EMNLP-IJCNLP 2019

# 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing

## Proceedings of the Conference

November 3–7, 2019
Hong Kong, China

The EMNLP-IJCNLP organizers gratefully acknowledge the support from the following sponsors.

**Diamond Level**




**Platinum Level**




**Gold Level**






## Silver Level







## Bronze Level




## Supporting Organizations



©2019 The Association for Computational Linguistics

Order copies of this and other ACL proceedings from:

Association for Computational Linguistics (ACL)
209 N. Eighth Street
Stroudsburg, PA 18360
USA
Tel: +1-570-476-8006
Fax: +1-570-476-0860
acl@aclweb.org

ISBN 978-1-950737-90-1

# Preface by the General Chair

Welcome to EMNLP-IJCNLP 2019 in Hong Kong! I hope that this will be a successful conference in the 25-year tradition of EMNLP and IJCNLP, as well as an enjoyable and enriching experience for all. Continuing recent trends, this year has seen overwhelming interest from researchers from all over the world. We received a total of 2,914 submissions, which is a 37% increase over EMNLP 2018 (2,117 submissions).

The three Program Co-Chairs, Jing Jiang, Vincent Ng, and Xiaojun Wan, oversaw the entire submission process, a truly Herculean feat. I am deeply grateful for your support. Managing such a large number of submissions would not have been possible without the hard work of a whopping 152 Area Chairs headed by 18 Senior Area Chairs as well, each responsible for overseeing a broad area. Under them, the combined effort of over 1,700 reviewers, each carefully examining submissions through the lens of their expertise, resulted in 465 long and 218 short papers to be presented at the main conference. Your efforts have been indispensable for making this conference happen.

I would like to express special thanks to Priscilla Rasmussen, the ACL Business Manager, who has been indispensable through many years of not only EMNLP but many other key NLP conferences. All of this would not have been possible without you. My deepest gratitude to David Yarowsky, Jian Su, and Noah Smith, and many others from the SIGDAT and AFNLP boards for their invaluable guidance in navigating difficult issues surrounding the daunting task I was faced with.

The hard work of our hosts, the Local Organizing Committee Chair Kam-Fai Wong and members Samuel Tam, Emmanuele Chersoni, Churen Huang, Wenjie Li, Derek F. Wong, and Ruifeng Xu, have especially been vital in bringing together EMNLP-IJCNLP 2019.

As many as 17 workshops and a co-located conference, CoNLL, have been coordinated by the Workshop Chairs Vera Demberg and Naoaki Okazaki. EMNLP-IJNLP 2019 will also be hosting a full-day tutorial and six half-day tutorials organized by the Tutorial Chairs Timothy Baldwin and Marine Carpuat. The combined efforts of the Demonstration Chairs Ruihong Huang and Sebastian Padó have culminated in a total of 44 accepted demo papers (out of 110 submissions) in addition to the overwhelming main conference papers. I am deeply grateful for your endless labor that brought to fruition an exciting program I am proud to be a part of.

We sought to make EMNLP 2019 a venue that is as welcoming and inclusive as possible to all. In this effort, we worked on continuing and expanding the diversity and inclusion (D&I) efforts initiated in recent NLP conferences as well as the Widening NLP workshop. The D&I committee, co-chaired by Chi-Kiu (Jackie) Lo and Vivek Srikumar, joined their forces with the Childcare Policy and Grant Coordinators Olivia Kwong and Sujian Li to help remove any obstacles to participating in a key event in the NLP community. Student Volunteer Coordinator and Student Scholarship Chair Wenjie Li and Marzieh Saeidi coordinated providing scholarships to students and non-students wanting to join us, who otherwise might not have had the means to. The D&I committee focused on several initiatives involving mentoring, both for first-time conference attendees and otherwise, providing accommodations and improving accessibility for participants if necessary, and generally making the conference experience a broadly comfortable one. For those unable to travel, we hope that the efforts of our Remote Presentation Chair, Derek F. Wong, enabled you to still feel included in the conference. We hope that these efforts to make EMNLP inclusive and welcoming will help enrich the conference.

A warm thank you to Micha Elsner, the General Publication Chair, Publication Chairs Fei Liu and Pontus Stenetorp, and also to Serena Villata, the Conference Handbook Chair, Kai-Wei Chang, the Conference Handbook Advisor, Natalie Schluter, the Conference Handbook Proofreader, for their strong sense of duty resulting in excellent supporting materials, arguably the most important contribution to many! I

must also thank Publicity Chairs Sebastian Ruder and Wei Xu, and Website and Conference App Chairs Kevin Duh and Henning Wachsmuth for your excellent promotion. We owe our success to reaching all those that could be interested, which cultivated a strong interest within the NLP community.

Last but not least, I would like to express my deepest gratitude to our sponsors, whose generous support has been invaluable in building up EMNLP and IJCNLP to what it is now, and to our Local Sponsorship Chair, Dongyan Zhao, for his assistance. A special thank you to Google, Facebook, Apple, ASAPP, and Salesforce, our Diamond sponsors. Many thanks to our Platinum sponsors Huawei, Baidu, DeepMind, and Amazon. We are also profoundly grateful to our Gold level sponsors PolyAI, Naver, ByteDance, Megagon Labs, Zhuiyi Technology, Verisk Analytics, and Xiaomi, Silver level sponsors Duolingo, SAP, Babelscape, eBay, and Cisco, Bronze level sponsor Shannon.AI, and finally, our supporting organization MEHK. Thank you for your monumental support towards hosting another (hopefully!) successful year of EMNLP-IJCNLP.

Finally, I would like to once again welcome you to EMNLP-IJCNLP 2019! We hope that this will be an exciting and memorable experience for you, especially if you are joining us for the first time. The NLP community is as thriving as ever, and I am honored to have had a part in hosting one of the leading conferences in the area of Natural Language Processing.

EMNLP-IJCNLP 2019 General Chair

*Kentaro Inui*, Tohoku University, Japan

# Preface by the Program Committee Co-Chairs

Welcome to EMNLP-IJCNLP 2019, the first joint EMNLP and IJCNLP conference! While this is the first time IJCNLP is held in Hong Kong, EMNLP left its footprints here in 2000 when it was still a tiny conference. There is nothing more exciting than seeing it return to Hong Kong after 19 years as one of the largest NLP conferences.

Owing to a significant increase in the number of submissions to recent NLP conferences, we, for the first time in the history of EMNLP and IJCNLP, attempted to reduce workload for reviewers by implementing a dual submission policy, where we disallowed authors to submit papers that are under review by a journal or another conference at the time of submission. Despite this policy, EMNLP-IJCNLP received 2914 submissions (excluding those withdrawn by the authors after initial submissions). This is an increase of over 30% compared with EMNLP 2018, making EMNLP-IJCNLP 2019 the largest NLP conference ever! Out of the 2914 submissions, 38 were desk-rejected for various reasons including formatting problems, length problems and violation of the dual submission policy. In spite of the record number of submissions, we managed to maintain a similar acceptance rate as past NLP conferences given the vast amount of space available to us at the AsiaWorld-Expo. In the end, we accepted 683 submissions. Some statistics of the accepted papers can be found below.

| | Long | Short | Total |
|---|---|---|---|
| Reviewed | 1,813 | 1,063 | 2,876 |
| Accepted as talk | 164 (9.0%) | 48 (4.5%) | 212 (7.3%) |
| Accepted as poster | 301 (16.6%) | 170 (16.0%) | 471 (16.4%) |
| Accepted (total) | 465 (25.6%) | 218 (20.5%) | 683 (23.7%) |

In addition, EMNLP-IJCNLP 2019 will feature 11 papers accepted by the Transactions of the Association for Computational Linguistics (TACL), out of which 8 will be presented orally and 3 as posters.

Handling close to 3000 submissions was a daunting task, but we were fortunate that a large team of volunteers from our community offered to help. Unlike last year's EMNLP, where a submission was reviewed in one of eight mega-areas, we organized this year's submissions into 18 areas, hoping that smaller areas would make things more manageable for our program committee. While traditionally large areas such as Information Extraction, Machine Learning for NLP, and Machine Translation and Multilinguality continued to receive a large number of submissions, areas such as Dialog and Interactive Systems and Summarization and Generation have grown significantly owing to the recent surge of interest in automated response generation.

We adopted a program committee structure similar to that of ACL 2019. For each area, we invited one Senior Area Chair, who worked with a team of Area Chairs (ranging from 4 to 18 per area) and an army of reviewers (1721 in total across all areas). Having a large number of ACs (152 in total) allowed us to assign each of them a reasonable number of papers, which in turn enabled them to better focus on evaluating each paper. Each submission was assigned to three reviewers and one AC. We allowed both the reviewers and the ACs to bid for papers, but used a combination of their bids and the TPMS (Toronto Paper Matching System) scores to assign papers. Although this lengthened the paper assignment process, we believe it allowed us to better match the submissions with reviewers. We also adopted a review form similar to what was used in ACL 2019 as we heard generally good feedback about less structured review forms. While NAACL HLT 2019 and ACL 2019 eliminated author response, we decided that it would be beneficial to keep it even though it put time pressure on our already tight reviewing schedule and resulted in additional work for our program committee members.

This year we received some submissions that raised ethical concerns from the reviewers, and we found that no existing guidelines could be applied. We decided to err on the side of acceptance, encouraging authors of otherwise acceptance-worthy papers to more deeply explore these issues in final drafts, and

encouraging the community to carry out further work.

We are extremely grateful to all the Senior Area Chairs, especially those who had a large number of submissions in their areas. The Senior Area Chairs did a fantastic job in nominating Area Chairs, recruiting reviewers and making final recommendations. We would also like to thank all the Area Chairs and reviewers for their hard work in writing meta-reviews and reviews, as well as leading and participating in the discussions. Special thanks to those emergency reviewers who offered help with short notice. Without the dedication of our program committee members, we would not be able to put together this conference program.

Award papers are an integral part of every NLP conference. Based on recommendations made by the ACs and the reviewers, we identified five candidates for the Best Paper award and another five for the Best Resource Paper award. We would like to thank Tim Baldwin, Claire Cardie, Dan Gildea, Qun Liu, Ellen Riloff, and Luke Zettlemoyer for serving in the Best Paper award committee, and Katrin Erk, Graeme Hirst, Gina-Anne Levow, Percy Liang, and Nianwen Xue for serving in the Best Resource Paper award committee. The award winners will be announced at the closing ceremony.

We are excited to have the following three keynote speakers: Noam Slonim (IBM Haifa), on automated debating technologies; Meeyoung Cha (KAIST), on research challenges in computational social science; and Kyunghyun Cho (NYU), on neural sequence modeling. We would like to thank them for traveling to Hong Kong to give the keynote speeches.

There are also many other people who contributed tremendously to the conference program, and we are very grateful for their help:

- Kentaro Inui, the General Conference Chair, who is always there to offer his help and advice;
- All the members of the Conference Coordinating Committee, who provided valuable advice on various issues that came up during the review process;
- David Chang and Julia Hockenmaier, Program Chairs of EMNLP 2018, who shared very helpful tips from their past experience;
- Rich Gerber from SoftConf, who helped us set up the conference submission site and always responded to our queries promptly;
- Other recent *ACL chairs who offered their help when we contacted them despite their busy schedule;
- TACL editors-in-chief Mark Johnson, Lillian Lee and Brian Roark, as well as TACL editorial assistant Cindy Robinson, for coordinating the TACL presentations with us;
- Micha Elsner, Fei Liu and Pontus Stenetorp, the Publication Chairs, who worked hard to compile the conference proceedings and kindly accommodated many last minute requests from authors;
- Kevin Duh, Henning Wachsmuth, Wei Xu and Sebastian Ruder, the Website Chairs and Publicity Chairs, who helped us make numerous announcements in a timely manner;
- Serena Villata, Natalie Schluter and Kai-Wei Chang for preparing and proofreading the conference handbook;
- Members of the Local Organizing Committee for making the local arrangements;
- Derek Wong, the Remote Presentation Chair, for taking care of remote presentations;
- Priscilla Rasmussen, whom we directed many inquiries to.

Again, welcome to EMNLP-IJCNLP 2019! We hope you will have a memorable conference experience!

EMNLP-IJCNLP 2019 Program Co-Chairs

*Jing Jiang*, Singapore Management University, Singapore
*Vincent Ng*, University of Texas at Dallas, USA
*Xiaojun Wan*, Peking University, China

# Organizing Committee

**General Chair**
Kentaro Inui, Tohoku University, Japan

**Program Committee Co-chairs**
Jing Jiang, Singapore Management University, Singapore
Vincent Ng, University of Texas at Dallas, USA
Xiaojun Wan, Peking University, China

**Local Arrangements Committee**
Kam-Fai Wong, Chinese University of Hong Kong, China (chair)
Samuel Tam, Chinese University of Hong Kong, China
Emmanuele Chersoni, Hong Kong Polytechnic University, China
Churen Huang, Hong Kong Polytechnic University, China
Wenjie Li, Hong Kong Polytechnic University, China
Derek F. Wong, University of Macau, China
Ruifeng Xu, Harbin Institute of Technology, Shenzhen, China

**Local Sponsorship Chair**
Dongyan Zhao, Peking University, China

**Workshop Co-chairs**
Vera Demberg, Saarland University, Germany
Naoaki Okazaki, Tokyo Institute of Technology, Japan

**Tutorial Co-chairs**
Timothy Baldwin, University of Melbourne, Australia
Marine Carpuat, University of Maryland, College Park, USA

**Demonstration Co-chairs**
Ruihong Huang, Texas A&M University, USA
Sebastian Padó, University of Stuttgart, Germany

**General Publication Chair**
Micha Elsner, Ohio State University, USA

**Publication Chairs**
Fei Liu, University of Central Florida, USA
Pontus Stenetorp, University College London, UK

**Conference Handbook Chair**
Serena Villata, CNRS, France

**Conference Handbook Advisor**
Kai-Wei Chang, University of California at Los Angeles, USA

**Conference Handbook Proofreader**

Natalie Schluter, IT University of Copenhagen, Denmark

**Publicity Chairs**
Sebastian Ruder, National University of Ireland, Ireland
Wei Xu, Ohio State University, USA

**Website and Conference App Chairs**
Kevin Duh, John Hopkins University, USA
Henning Wachsmuth, Paderborn University, Germany

**Remote Presentation Chair**
Derek F. Wong, University of Macau, China

**Student Volunteer Coordinator and Student Scholarship Chairs**
Wenjie Li, Hong Kong Polytechnic University, China
Marzieh Saeidi, Facebook AI Research, UK

**Childcare Policy and Grant Coordinators**
Olivia Kwong, Chinese University of Hong Kong, China
Sujian Li, Peking University, China

**Diversity and Inclusion Chairs**
Chi-Kiu (Jackie) Lo, National Research Council Canada, Canada
Vivek Srikumar, University of Utah, USA

# Program Committee

**Program Committee Co-chairs**
Jing Jiang, Singapore Management University, Singapore
Vincent Ng, University of Texas at Dallas, USA
Xiaojun Wan, Peking University, China

**Senior Area Chairs**
*Dialog and Interactive Systems*
Amanda Stent, Bloomberg, USA

*Discourse and Pragmatics*
Bonnie Webber, University of Edinburgh, UK

*Information Extraction*
Dan Roth, University of Pennsylvania, USA

*Information Retrieval and Document Analysis*
Hang Li, Bytedance Technology, China

*Lexical Semantics*
Mona Diab, George Washington University, USA

*Linguistic Theories, Cognitive Modeling and Psycholinguistics*
Andrew Kehler, University of California, San Diego, USA

*Machine Learning for NLP*
Kevin Duh, Johns Hopkins University, USA

*Machine Translation and Multilinguality*
Chris Quirk, Microsoft Research, USA

*Phonology, Morphology and Word Segmentation*
Yuji Matsumoto, Nara Institute of Science and Technology, Japan

*Question Answering*
Scott Wen-tau Yih, Facebook AI Research, USA

*Sentence-level Semantics*
Mirella Lapata, University of Edinburgh, UK

*Sentiment Analysis and Argument Mining*
Yulan He, University of Warwick, UK

*Social Media and Computational Social Science*
Dirk Hovy, Bocconi University, Italy

*Speech, Vision, Robotics, Multimodal and Grounding*
Haizhou Li, National University of Singapore, Singapore

*Summarization and Generation*
Michael White, Ohio State University, USA

*Tagging, Chunking, Syntax and Parsing*
Yue Zhang, Westlake University, China

*Text Mining and NLP Applications*
Jimmy Lin, University of Waterloo, Canada

*Textual Inference and Other Areas of Semantics*
Alessandro Moschitti, Amazon, USA

**Area Chairs**

*Dialog and Interactive Systems*
Luciana Benotti, Asli Celikyilmaz, Yun-Nung Chen, Milica Gasic, Ryuichiro Higashinaka, Casey Kennington, Kazunori Komatani, Sungjin Lee, Stefan Ultes

*Discourse and Pragmatics*
Yangfeng Ji, Shafiq Joty, Annie Louis, Michael Strube

*Information Extraction*
Snigdha Chaturvedi, Doug Downey, Radu Florian, Daniel Khashabi, Marius Pasca, Roi Reichart, Satoshi Sekine, Vivek Srikumar, Shyam Upadhyay, Jun Zhao

*Information Retrieval and Document Analysis*
Eugene Agichtein, Gerard de Melo, Oren Kurland, Craig Macdonald, Alice Oh, Xiang Ren, Quan Wang, Andrew Yates

*Lexical Semantics*
Marianna Apidianaki, Mohit Bansal, Manaal Faruqui, Qin Lu, Michael Roth

*Linguistic Theories, Cognitive Modeling and Psycholinguistics*
Leon Bergen, Richard Futrell, William Schuler, Suzanne Stevenson

*Machine Learning for NLP*
Isabelle Augenstein, Loïc Barrault, Danushka Bollegala, Shay B. Cohen, Yoav Goldberg, Gholamreza Haffari, Shou-de Lin, Nanyun Peng, Sujith Ravi, Laura Rimell, Sameer Singh, Jun Suzuki, Andreas Vlachos, William Yang Wang

*Machine Translation and Multilinguality*
Marine Carpuat, Daniel Cer, Boxing Chen, Colin Cherry, David Chiang, Ann Clifton, Marcello Federico, Orhan Firat, George Foster, Yang Liu, Minh-Thang Luong, Haitao Mi, Anders Søgaard, Jinsong Su, Zhaopeng Tu, Taro Watanabe, Deyi Xiong, Jiajun Zhang

*Phonology, Morphology and Word Segmentation*
Nizar Habash, Kemal Oflazer, Xipeng Qiu, Hiroyuki Shindo

*Question Answering*
Danqi Chen, Eunsol Choi, Yansong Feng, Matt Gardner, Sanda Harabagiu, Mohit Iyyer, Furu Wei, Caiming Xiong

*Sentence-level Semantics*

Jacob Andreas, Li Dong, Daniel Gildea, Wei Lu, Patrick Pantel

*Sentiment Analysis and Argument Mining*

Nikolaos Aletras, Alexandra Balahur, Dan Goldwasser, Minlie Huang, Saif Mohammad, Chris Reed, Rui Xia, Xiaodan Zhu

*Social Media and Computational Social Science*

David Bamman, David Jurgens, Dong Nguyen, Alan Ritter, Chenhao Tan, Oren Tsur, Yulia Tsvetkov, Svitlana Volkova, Wayne Xin Zhao

*Speech, Vision, Robotics, Multimodal and Grounding*

Yoav Artzi, Hannaneh Hajishirzi, Xiaodong He, Gina-Anne Levow, Yang Liu, Soujanya Poria, Kai Yu

*Summarization and Generation*

Anya Belz, Giuseppe Carenini, Jackie Chi Kit Cheung, Mark Dras, Michael Elhadad, Mary Ellen Foster, Claire Gardent, Albert Gatt, Chin-Yew Lin, Kathleen McKeown, Shashi Narayan, Lu Wang, Leo Wanner, Sina Zarrieß

*Tagging, Chunking, Syntax and Parsing*

Miguel Ballesteros, James Henderson, Emily Pitler, Barbara Plank, Kenji Sagae, Weiwei Sun, Meishan Zhang

*Text Mining and NLP Applications*

Kai-Wei Chang, Wei Gao, Lun-Wei Ku, Lei Li, Zhiyuan Liu, David Mimno, Preslav Nakov, Ming-Feng Tsai, Yoshimasa Tsuruoka, Byron C. Wallace, Min Zhang

*Textual Inference and Other Areas of Semantics*

Weiwei Cheng, Markus Dreyer, Cicero Nogueira dos Santos, Aliaksei Severyn, Kevin Small, Nianwen Xue, Fabio Massimo Zanzotto

**Reviewers**

Ahmed Abdelali, Asad Abdi, Mostafa Abdou, Muhammad Abdul-Mageed, Omri Abend, Abdalghani Abujabal, Manoj Acharya, Heike Adel, Stergos Afantenos, Oshin Agarwal, Apoorv Agarwal, Rodrigo Agerri, Željko Agić, Priyanka Agrawal, Roee Aharoni, Ali Ahmadvand, Natalie Ahn, Chaitanya Ahuja, Qingyao Ai, Alan Akbik, Nader Akoury, Khalid Al Khatib, Hussein Al-Olimat, Firoj Alam, Chris Alberti, Malihe Alikhani, Alexandre Allauzen, James Allen, Malik Altakrori, Kristen M. Altenburger, Fernando Alva-Manchego, Bharat Ram Ambati, Hessam Amini, Waleed Ammar, Reinald Kim Amplayo, Ashish Anand, Avishek Anand, Antonios Anastasopoulos, Peter Anderson, Nicholas Andrews, Gabor Angeli, Stefanos Angelidis, Jean-Yves Antoine, Emilia Apostolova, Kenji Araki, Masahiro Araki, Jun Araki, Rahul Aralikatte, Arturo Argueta, Piyush Arora, Mikel Artetxe, Masayuki Asahara, Akari Asai, Ehsaneddin Asgari, Nabiha Asghar, Ramón Astudillo, Eleftherios Avramidis, amittai axelrod, Wilker Aziz, JinYeong Bak, Mithun Balakrishna, Anusha Balakrishnan, Niranjan Balasubramanian, Livio Baldini Soares, Timothy Baldwin, Rafael E. Banchs, Siddhartha Banerjee, Jeesoo Bang, Sameer Bansal, Trapit Bansal, Forrest Sheng Bao, Ankur Bapna, Roy Bar-Haim, Libby Barak, Verginica Barbu Mititelu, Maria Barrett, Valentin Barriere, Joe Barrow, Alberto Barrón-Cedeño, Guntis Barzdins, Valerio Basile, Pierpaolo Basile, Roberto Basili, Joost Bastings, Riza Batista-Navarro, Vishwash Batra,

Timo Baumann, Daniel Beck, Barend Beekhuizen, John Beieler, Giannis Bekoulis, Núria Bel, Yonatan Belinkov, Eric Bell, Kedar Bellare, Meriem Beloucif, Yassine Benajiba, Farah Benamara, Emily M. Bender, Luciana Benotti, Adrian Benton, Toms Bergmanis, Thales Bertaglia, Dario Bertero, Robert Berwick, Laurent Besacier, Steven Bethard, Rahul Bhagat, Chandra Bhagavatula, Archna Bhatia, Parminder Bhatia, Sumit Bhatia, Wei Bi, Klinton Bicknell, Chris Biemann, Lidong Bing, Or Biran, Alexandra Birch, Arianna Bisazza, Yonatan Bisk, André Bittar, PRAKHAR BIYANI, Johannes Bjerva, Jari Björne, Eduardo Blanco, Su Lin Blodgett, Michael Bloodgood, Valts Blukis, Sravan Bodapati, Reihane Boghrati, Ben Bogin, Nikolay Bogoychev, Bernd Bohnet, Daniele Bonadiman, Kalina Bontcheva, Georgeta Bordea, Johan Bos, Antoine Bosselut, Florian Boudin, Samuel R. Bowman, Ryan Boyd, Jordan Boyd-Graber, Chloé Braud, Norbert Braunschweiler, Jonathan Brennan, Chris Brew, Chris Brockett, Thomas Brovelli (Meyer), Caroline Brun, Amar Budhiraja, Paweł Budzianowski, Alberto Bugarín Diz, Trung Bui, Laura Burdick, Evgeny Burnaev, Bill Byrne, Benjamin Börschinger, José G. C. de Souza, Elena Cabrio, Aoife Cahill, Deng Cai, Andrew Caines, Iacer Calixto, Erik Cambria, Wlliam Campbell, Burcu Can, Nicola Cancedda, Marie Candito, Yuan Cao, Kris Cao, Ziqiang Cao, Yixin Cao, Cornelia Caragea, Dallas Card, Iñigo Casanueva, Tommaso Caselli, Giuseppe Castellucci, Thiago Castro Ferreira, Asli Celikyilmaz, Mauro Cettolo, Arun Chaganty, Joyce Chai, Soumen Chakrabarti, Tanmoy Chakraborty, Yllias Chali, Hou Pong Chan, Yee Seng Chan, Senthil Chandramohan, Muthu Kumar Chandrasekaran, Baobao Chang, Yin-Wen Chang, Baobao Chang, Angel Chang, Soravit Changpinyo, Rajen Chatterjee, Stergios Chatzikyriakidis, Wanxiang Che, John Chen, Zhumin CHEN, Muhao Chen, Howard Chen, Sihao Chen, Hsin-Hsi Chen, Wenliang Chen, Yufei Chen, Yubo Chen, Yun-Nung Chen, Tao Chen, Mingda Chen, Tongfei Chen, Xinchi Chen, Kehai Chen, Lu Chen, Wenhu Chen, Chen Chen, Xilun Chen, Qian Chen, Lei Chen, Guanyi Chen, Fei Cheng, Jianpeng Cheng, Hao Cheng, Yong Cheng, Jen-Tzung Chien, Hai Leong Chieu, Maria Chinkina, Dhivya Chinnappa, Luis Chiruzzo, Laura Chiticariu, Yejin Choi, Heeyoul Choi, Shamil Chollampatt, Leshem Choshen, Prafulla Kumar Choubey, Shammur Absar Chowdhury, Chenhui Chu, Tagyoung Chung, Philipp Cimiano, Yagmur Gizem Cinar, Volkan Cirik, Elizabeth Clark, Alexander Clark, Christopher Clark, Chloé Clavel, Simon Clematide, Maximin Coavoux, Oana Cocarascu, Anne Cocos, Arman Cohan, Daniel Cohen, Reuben Cohn-Gordon, Guillem Collell, Costanza Conforti, Gao Cong, John Conroy, Mathieu Constant, Danish Contractor, Bonaventura Coppola, Francesco Corcoglioniti, Caio Corro, Marta R. Costa-jussà, Ryan Cotterell, Benoit Crabbé, Josep Crego, Danilo Croce, Paul Crook, Heriberto Cuayahuitl, Yiming Cui, Luis Fernando D'Haro, Raj Dabre, Zeyu Dai, XIN-YU DAI, Andrew Dai, Zhuyun Dai, Joachim Daiber, Jeff Dalton, Marco Damonte, Lena Dankin, Kareem Darwish, Abhishek Das, Rajarshi Das, Dipanjan Das, Amitava Das, Pradeep Dasigi, Vidas Daudaravicius, Hal Daumé III, Brian Davis, Cedric De Boom, Gaël de Chalendar, Orphee De Clercq, Éric de la Clergerie, Miryam de Lhoneux, Thierry Declerck, Luciano Del Corro, Louise Deléger, Thomas Demeester, David Demeter, Steve DeNeefe, Yuntian Deng, Lingjia Deng, Alexandre Denis, Matthew Denny, Valeria dePaiva, Leon Derczynski, Nina Dethlefs, Daniel Deutsch, Chris Develder, Barry Devereux, Jacob Devlin, Bhuwan Dhingra, Luigi Di Caro, Giorgio Maria Di Nunzio, Adji Bousso Dieng, Jana Diesner, Haibo Ding, Xiao Ding, Shuoyang Ding, Simon Dobnik, Tobias Domhan, Shichao Dong, Yue Dong, Daxiang Dong, Zhicheng Dou, Gabriel Doyle, Timothy Dozat, A. Seza Doğruöz, Eduard Dragut, Mark Dras, Rotem Dror, Xinya Du, Lan Du, Dheeru Dua, Nan Duan, Xiangyu Duan, Pablo Duboue, Shiran Dudy, Anca Dumitrache, Ewan Dunbar, Esin Durmus, Nadir Durrani, Greg Durrett, Ondřej Dušek, Chris Dyer, Marc Dymetman, Valery Dzutsati, Hiroshi Echizen'ya, Thomas Effland, Steffen Eger, Maud Ehrmann, Vladimir Eidelman, Arash Einolghozati, Andreas Eisele, Jacob Eisenstein, Asif Ekbal, Heba Elfardy, Ahmed Elgohary, Michael Elhadad, Desmond Elliott, Hady Elsahar, Micha Elsner, Aykut Erdem, Alexander Erdmann, Mihail Eric, Akiko Eriguchi, Ramy Eskander, Luis Espinosa Anke, Keelan Evanini, Ingrid

Falk, Tobias Falke, Angela Fan, Feifan Fan, James Fan, Kai Fan, Licheng Fang, Hao Fang, Meng Fang, Hui Fang, Farhood Farahnak, M. Amin Farajian, Maryam Fazel-Zarandi, Christian Federmann, Anna Feldman, Shi Feng, Shi Feng, Xiaocheng Feng, Yang Feng, Eraldo Fernandes, Raquel Fernández, Daniel Fernández-González, Elisa Ferracane, Francis Ferraro, Besnik Fetahu, Oluwaseyi Feyisetan, Elena Filatova, Mark Finlayson, Nicholas FitzGerald, Jeffrey Flanigan, Margaret Fleck, Lucie Flekova, Michael Flor, Antske Fokkens, José A. R. Fonollosa, April Foreman, Tommaso Fornaciari, Eric Fosler-Lussier, Anette Frank, Diego Frassinelli, Markus Freitag, Dayne Freitag, André Freitas, Jesse Freitas, Lea Frermann, Daniel Fried, Annemarie Friedrich, Zhenxin Fu, Lisheng Fu, Zuohui Fu, Liye Fu, Hagen Fuerstenau, Akinori Fujino, Kotaro Funakoshi, Matthias Gallé, Michael Gamon, Leilei Gan, Zhe Gan, Octavian-Eugen Ganea, Rashmi Gangadharaiah, Varun Gangal, Debasis Ganguly, Juri Ganitkevitch, Wei Gao, Qiaozi Gao, Xiang Gao, Shen Gao, Fei Gao, Mercedes García-Martínez, Siddhant Garg, Francesco Gargiulo, Ekaterina Garmash, Dan Garrette, Dragan Gasevic, Lorenzo Gatti, Tao Ge, Sebastian Gehrmann, Michaela Geierhos, Alexander Gelbukh, Lieke Gelderloos, Spandana Gella, Debela Gemechu, Kallirroi Georgila, Kim Gerdes, Ulrich Germann, Pablo Gervás, Daniela Gerz, Reza Ghaeini, Marjan Ghazvininejad, Deepanway Ghosal, Debanjan Ghosh, Sucheta Ghosh, Daniela Gifu, Emer Gilmartin, Kevin Gimpel, Filip Ginter, Roxana Girju, Goran Glavaš, Alfio Gliozzo, Lorraine Goeuriot, Darina Gold, Yeyun Gong, Jingjing Gong, Jesús González-Rubio, Hugo Gonçalo Oliveira, Mitchell Gordon, Kyle Gorman, Matthew R. Gormley, Prasoon Goyal, Pawan Goyal, Kartik Goyal, Anuj Goyal, Yvette Graham, Erin Grant, Scott Grimm, Yulia Grishina, Alvin Grissom II, Dagmar Gromann, Roman Grundkiewicz, Jiatao Gu, Lin Gui, Tao Gui, Camille Guinaudeau, Chulaka Gunasekara, Jiafeng Guo, Weiwei Guo, Jiang Guo, Arpit Gupta, Sonal Gupta, Pankaj Gupta, Deepak Gupta, Arshit Gupta, Nitish Gupta, Raghav Gupta, Iryna Gurevych, Francisco Guzmán, Jeremy Gwinnup, Carlos Gómez-Rodríguez, Masato Hagiwara, Udo Hahn, Dilek Hakkani-Tur, Keith Hall, William L. Hamilton, Michael Hammond, Ting Han, Bo Han, Lifeng Han, Jialong Han, Xianpei Han, Xu Han, Shuguang Han, Abram Handler, Christian Hardmeier, Daniel Hardt, Hardy Hardy, Mareike Hartmann, Sadid A. Hasan, Kazi Saidul Hasan, Chikara Hashimoto, Kazuma Hashimoto, Eva Hasler, Hany Hassan, Ahmed Hassan Awadallah, Claudia Hauff, Annette Hautli-Janisz, Catherine Havasi, Katsuhiko Hayashi, Devamanyu Hazarika, Shizhu He, Zhongjun He, Ji He, Hangfeng He, Hua He, Ruidan He, Yifan He, Luheng He, Ben He, Shexia He, Qi He, Kenneth Heafield, John Henderson, Matthew Henderson, Aron Henriksson, Aurélie Herbelot, Ulf Hermjakob, Delia Irazú Hernández Farías, Daniel Hershcovich, Raquel Hervas, Jonathan Herzig, Jack Hessel, Christopher Hidey, Ryuichiro Higashinaka, Gerold Hintz, Tsutomu Hirao, Sorami Hisamoto, Vu Cong Duy Hoang, Hieu Hoang, Julia Hockenmaier, Nathan Hodas, Johannes Hoffart, Chris Hokamp, Chester Holtz, Ari Holtzman, Yu Hong, Enamul Hoque, Ales Horak, Chiori Hori, Nabil Hossain, Mohammad Javad Hosseini, Yufang Hou, Eduard Hovy, Phu Mon Htut, Yuheng Hu, Junjie Hu, Guangneng Hu, Linmei Hu, Zhiting Hu, Yin Jou Huang, Jing Huang, Zhongqiang Huang, Guoping Huang, Qiuyuan Huang, Shujian Huang, Xuanjing Huang, Haoran Huang, Lifu Huang, Fei Huang, Hen-Hsen Huang, Po-Sen Huang, Minlie Huang, Xiaolei Huang, Patrick Huber, Matthias Huck, Kai Hui, Tim Hunter, Samar Husain, Seungwon Hwang, Ali Hürriyetoğlu, Ignacio Iacobacci, Adrian Iftene, Gonzalo Iglesias, Carlos A. Iglesias, Ryu Iida, Ilija Ilievski, Oana Inel, Takashi Inui, Radu Tudor Ionescu, Ozan Irsoy, Aminul Islam, Julia Ive, Srinivasan Iyer, Aaron Jaech, Kokil Jaidka, Prachi Jain, Shoaib Jameel, Hyeju Jang, Slava Jankin, Peter Jansen, Adam Jatowt, Sujay Kumar Jauhar, Sébastien Jean, Laura Jehl, Minwoo Jeong, Yacine Jernite, Elisabetta Jezek, Rahul Jha, Harsh Jhamtani, Zongcheng Ji, Feng Ji, Robin Jia, Meng Jiang, Yong Jiang, Xin Jiang, Zhuoren Jiang, Tianyu Jiang, Daxin Jiang, Jyun-Yu Jiang, Wenbin Jiang, Salud María Jiménez-Zafra, Hongxia Jin, Di Jin, Lifeng Jin, Yohan Jo, Charles Jochim, Richard Johansson, Kristen Johnson, Justin Johnson, Gareth Jones, Kenneth Joseph, Aditya Joshi, Prathyusha

Jwalapuram, Preethi Jyothi, Jad Kabbara, Kushal Kafle, Nobuhiro Kaji, Tomoyuki Kajiwara, Laura Kallmeyer, Christopher Kanan, Hiroshi Kanayama, Dongyeop Kang, Katharina Kann, Diptesh Kanojia, Sudipta Kar, Mladen Karan, Sarvnaz Karimi, Dimitri Kartsaklis, Arzoo Katiyar, Makoto P. Kato, David Kauchak, Daisuke Kawahara, Hideto Kazawa, Pei Ke, Chris Kedzie, Frank Keller, Aniruddha Kembhavi, Yova Kementchedjhieva, Ruth Kempson, Casey Kennington, Salam Khalifa, Alizishaan Khatri, Mikhail Khodak, Douwe Kiela, Halil Kilicoglu, Sun Kim, Seokhwan Kim, Yunsu Kim, Sunghwan Mac Kim, Young-Bum Kim, Joo-Kyung Kim, Jin-Dong Kim, Yoon Kim, Najoung Kim, Suin Kim, Irwin King, David King, Svetlana Kiritchenko, Andreas Søeborg Kirkedal, Julia Kiseleva, Norihide Kitaoka, Roman Klinger, Alistair Knott, Rebecca Knowles, Hayato Kobayashi, Sosuke Kobayashi, Thomas Kober, Ari Kobren, Simon Kocbek, Ekaterina Kochmar, Rob Koeling, Alexander Koller, Kazunori Komatani, Rik Koncel-Kedziorski, Grzegorz Kondrak, Xiang Kong, Ioannis Konstas, Parisa Kordjamshidi, Yannis Korkontzelos, Anastassia Kornilova, Bhushan Kotnis, Margarita Kotti, Sebastian Krause, Ralf Krestel, Julia Kreutzer, Kalpesh Krishna, Jayant Krishnamurthy, Nikhil Krishnaswamy, Canasai Kruengkrai, Udo Kruschwitz, Taku Kudo, Shankar Kumar, Anjishnu Kumar, Gaurav Kumar, Jonathan K. Kummerfeld, Adhiguna Kuncoro, Gourab Kundu, Tsung-Ting Kuo, Murathan Kurfalı, Sadao Kurohashi, Tom Kwiatkowski, Ákos Kádár, Arne Köhn, Gorka Labaka, Cyril Labbe, Matthieu Labeau, Ophélie Lacroix, Anirban Laha, Yuxuan Lai, Veronika Laippala, Wai Lam, Mathias Lambert, Patrik Lambert, Vasileios Lampos, Yanyan Lan, Ni Lao, Gabriella Lapesa, Stefan Larson, Michael A. Laurenzano, Alberto Lavelli, John Lawrence, Carolin Lawrence, Angeliki Lazaridou, Phong Le, Joseph Le Roux, Robert Leaman, Gianluca Lebani, Jason Lee, Young-Suk Lee, Yoong Keok Lee, Moontae Lee, Kenton Lee, Kuang-Huei Lee, Hung-yi Lee, Artuur Leeuwenberg, Els Lefever, Tao Lei, Wenqiang Lei, Jochen L. Leidner, Gaël Lejeune, Alessandro Lenci, Effi Levi, Ran Levy, Omer Levy, Martha Lewis, Xiang Li, Liangyou Li, Chenliang Li, Yaliang Li, Cheng-Te Li, Wenjie Li, Xiujun Li, Junyi Jessy Li, Chang Li, Shang-Wen Li, Quanzhi Li, Zichao Li, Yanran Li, Baoli LI, Yunyao Li, Zhixing Li, Xin Li, Yanen Li, Haibo Li, Zuchao Li, Sujian Li, Yitong Li, Juanzi Li, Jing Li, Junhui Li, Piji Li, Lei Li, Qing Li, Sheng Li, Chen Li, Lishuang Li, Zhenghua Li, Jery(Shaochun) Li, Maria Liakata, Paul Pu Liang, Chen Liang, Ming Liao, Jindřich Libovický, Patricia Lichtenstein, Anne-Laure Ligozat, Nut Limsopatham, Yankai Lin, Lucy H. Lin, Zhouhan Lin, Chenghua Lin, Chu-Cheng Lin, Bill Yuchen Lin, Chuan-Jie Lin, Ying Lin, Dekang Lin, Angela Lin, Tal Linzen, Marco Lippi, Zachary Lipton, Diane Litman, Marina Litvak, Lemao Liu, Wu Liu, Feifan Liu, Qi Liu, Xiaodong Liu, Qun Liu, Tong Liu, Yiqun Liu, Nelson F. Liu, Kang Liu, Xiaozhong Liu, Jiangming Liu, Tianyu Liu, Yang Liu, Shujie Liu, Kai Liu, Bing Liu, Liyuan Liu, Fei Liu, Yijia Liu, Pengfei Liu, Bing Liu, Nikola Ljubešić, Kyle Lo, Colin Lockard, Alexander Loeser, Robert Logan, Yunfei Long, Shangbang Long, Lucelene Lopes, Oier Lopez de Lacalle, Aurelio Lopez-Lopez, Daniel Loureiro, Pablo Loyola, Sharid Loáiciga, Jing Lu, Michal Lukasik, Liangchen Luo, Zhunchen Luo, Wencan Luo, Anh Tuan Luu, Chunchuan Lyu, Ji Ma, Xuezhe Ma, Mingbo Ma, Jing Ma, Jianqiang Ma, Wei-Yun Ma, Shuming Ma, Xutai Ma, Tengfei Ma, Sean MacAvaney, Mounica Maddela, Pranava Madhyastha, Andrea Madotto, Walid Magdy, Giorgio Magri, Saad Mahamood, Wolfgang Maier, Navonil Majumder, Peter Makarov, Prodromos Malakasiotis, Alfredo Maldonado, Andreas Maletti, Igor Malioutov, Jonathan Mallinson, Shervin Malmasi, Radhika Mamidi, Suresh Manandhar, Gideon Mann, Christopher D. Manning, Ramesh Manuvinakurike, Jiaxin Mao, Yuning Mao, Vladislav Maraev, Ana Marasovic, Diego Marcheggiani, Daniel Marcu, Katerina Margatina, Benjamin Marie, Alex Marin, Edison Marrese-Taylor, Marianna Martindale, André F. T. Martins, Bruno Martins, Héctor Martínez Alonso, Eugenio Martínez-Cámara, Luis Marujo, Ryo Masumura, Prashant Mathur, Shigeki Matsubara, Yuichiroh Matsubayashi, Takuya Matsuzaki, Austin Matthews, Arne Mauser, Jonathan May, Chandler May, Stephen Mayhew, Joshua Maynez, Alessandro Mazzei, David McAllester, David McClosky, John Philip Mc-

Crae, Philipp Meerkamp, Yashar Mehdad, Sanket Vaibhav Mehta, Sachin Mehta, Hongyuan Mei, Arul Menezes, Rui Meng, Helen Meng, Paola Merlo, Mohsen Mesgar, Florian Metze, Marie-Jean Meurs, Christian M. Meyer, Ivan Vladimir Meza Ruiz, Yishu Miao, Antonio Valerio Miceli Barone, Julian Michael, Sebastian J. Mielke, Margot Mieskes, Simon Mille, Timothy Miller, Tristan Miller, David Mimno, Sewon Min, Bonan Min, Anne-Lyse Minard, Pasquale Minervini, Michael Minock, Sabino Miranda-Jiménez, Seyed Abolghasem Mirroshandel, Paramita Mirza, Abhijit Mishra, Dipendra Misra, Jelena Mitrović, Arpit Mittal, Sudip Mittal, Daichi Mochihashi, Ashutosh Modi, Omid Mohamad Nezami, Karthik Mohan, Tasnim Mohiuddin, Michael Mohler, Luis Gerardo Mojica de la Vega, Diego Molla, Nicholas Monath, Taesun Moon, Nafise Sadat Moosavi, Roser Morante, Mohamed Morchid, Isabel Moreno, Véronique MORICEAU, Hajime Morita, David R. Mortensen, Diego Moussallem, Tingting Mu, Pramod Kaushik Mudrakarta, Animesh Mukherjee, Phoebe Mulcaire, Philippe Muller, Yugo Murawaki, Smaranda Muresan, Gabriel Murray, Kenton Murray, Shikhar Murty, Adrian Muscat, sunghyon myaeng, Jiří Mírovský, Maria Nadejde, Masaaki Nagata, Ajay Nagesh, Tetsuji Nakagawa, Yukiko Nakano, Ndapa Nakashole, Crystal Nakatsu, Toshiaki Nakazawa, Preslav Nakov, Nikita Nangia, Jason Naradowsky, Karthik Narasimhan, Shashi Narayan, Tahira Naseem, Alexis Nasr, Roberto Navigli, Adeline Nazarenko, Mark-Jan Nederhof, Matteo Negri, Aida Nematzadeh, Graham Neubig, Guenter Neumann, Mariana Neves, Denis Newman-Griffis, Hwee Tou Ng, Axel-Cyrille Ngonga Ngomo, Khanh Nguyen, Truc-Vien T. Nguyen, Huy Nguyen, Patrick Nguyen, Thien Huu Nguyen, Dat Quoc Nguyen, Jianmo Ni, Garrett Nicolai, Liqiang Nie, Jian-Yun NIE, Jan Niehues, Vassilina Nikoulina, Madhav Nimishakavi, Qiang Ning, Nobal Bikram Niraula, Hitoshi Nishikawa, Masaaki Nishino, Malvina Nissim, Xing Niu, Tong Niu, Zheng-Yu Niu, Joakim Nivre, Hiroshi Noji, Jekaterina Novikova, Pierre Nugues, Brendan O'Connor, Tim O'Gorman, Daniela Alejandra Ochoa, Yusuke Oda, Maciej Ogrodniczuk, Jong-Hoon Oh, Naoaki Okazaki, Tsuyoshi Okita, Oleg Okun, Ethel Ong, Constantin Orasan, Petya Osenova, Simon Ostermann, Myle Ott, Hiroki Ouchi, Iadh Ounis, Jessica Ouyang, Katja Ovchinnikova, Avinesh P.V.S, Ankur Padia, Aishwarya Padmakumar, Alexis Palmer, Martha Palmer, Alessio Palmero Aprosio, Yingwei Pan, Alexander Panchenko, Alexandros Papangelis, Nikos Papasarantopoulos, Nikolaos Pappas, Natalie Parde, Ankur Parikh, Joonsuk Park, Sungjoon Park, Yannick Parmentier, Patrick Paroubek, Carla Parra Escartín, Prasanna Parthasarathi, Tommaso Pasini, Ramakanth Pasunuru, Panupong Pasupat, Viviana Patti, Siddharth Patwardhan, Adam Pauls, Steffen Pauws, Umashanthi Pavalanathan, Ellie Pavlick, Adam Pease, Hao Peng, Haoruo Peng, Xiaochang Peng, Baolin Peng, Gerald Penn, Martín Pereira-Fariña, Ian Perera, Laura Perez-Beltrachini, Gabriele Pergola, Isaac Persing, Slav Petrov, Maxime Peyrard, Pouya Pezeshkpour, Sandro Pezzelle, Hieu Pham, Karl Pichotta, Mohammad Taher Pilehvar, Mārcis Pinnis, Yuval Pinter, Vassilis Plachouras, Emmanouil Antonios Platanios, Julien Plu, Massimo Poesio, Adam Poliak, Lucie Poláková, Edoardo Maria Ponti, Hanieh Poostchi, Andrei Popescu-Belis, Maja Popović, Christopher Potts, Amir Pouran Ben Veyseh, Vinodkumar Prabhakaran, Daniel Preoţiuc-Pietro, Danish Pruthi, Piotr Przybyła, Ratish Puduppully, Rajkumar Pujari, Matthew Purver, Yevgeniy Puzikov, Valentina Pyatkin, Sampo Pyysalo, Laura Pérez-Mayos, Peng Qi, Longhua Qian, Yanmin Qian, Jing Qian, Yujie Qian, Pengda Qin, Lianhui Qin, Guanghui Qin, Long Qiu, Minghui Qiu, Meng Qu, Lizhen Qu, Ella Rabinovich, Alexandre Rademaker, Muhammad Rafi, Dinesh Raghu, Afshin Rahimi, Muhammad Rahman, Jonathan Raiman, Dheeraj Rajagopal, Nazneen Fatema Rajani, Nitendra Rajput, Taraka Rama, Rohan Ramanath, Gabriela Ramirez-de-la-Rosa, Jinfeng Rao, Sudha Rao, Farzana Rashid, Hannah Rashkin, Mohammad Sadegh Rasooli, Pushpendre Rastogi, Abhinav Rastogi, Vinit Ravishankar, Livy Real, Siva Reddy, Georg Rehm, Marek Rei, Fiana Reiber, Julia Reinspach, David Reitter, Steffen Remus, Zhaochun Ren, Adithya Renduchintala, Rezvaneh Rezapour, Corentin Ribeyre, Matthew Richardson, Mark Riedl, Jason Riesa, German Rigau, Fabio Rinaldi, Kirk Roberts, Melissa Roemmele, Lina M. Rojas Barahona, Oleg Rokhlenko, Matteo

Romanello, Salvatore Romeo, Wenge Rong, Carolyn Rose, Andrew Rosenberg, Paolo Rosso, Benjamin Roth, Sascha Rothe, Aurko Roy, Sebastian Ruder, Alexander Rudnicky, Vasile Rus, Alexander Rush, Irene Russo, Derek Ruths, Mrinmaya Sachan, Kugatsu Sadamitsu, Fatiha Sadat, Mehrnoosh Sadrzadeh, Benoît Sagot, Monjoy Saha, Rishiraj Saha Roy, Saurav Sahay, Magnus Sahlgren, Sunil Kumar Sahu, Hassan Sajjad, Keisuke Sakaguchi, Sakriani Sakti, Mohammad Salameh, Iman Saleh, Elizabeth Salesky, Avneesh Saluja, Rajhans Samdani, Germán Sanchis-Trilles, Marina Santini, Diana Santos, Maarten Sap, Murat Saraclar, Ryohei Sasano, Hassan Sawaf, Asad Sayeed, Carolina Scarton, Shigehiko Schamoni, Odette Scharenborg, Niko Schenk, Yves Scherrer, Frank Schilder, David Schlangen, Natalie Schluter, Helmut Schmid, Steven Schockaert, Hannes Schulz, Anne-Kathrin Schumann, Sebastian Schuster, Roy Schwartz, H. Andrew Schwartz, Holger Schwenk, Djamé Seddah, Ethan Selfridge, Jean Senellart, Rico Sennrich, Minjoon Seo, Yeon Seonwoo, Christophe Servan, Lei Sha, Izhak Shafran, Kashif Shah, Pararth Shah, Samira Shaikh, Cory Shain, Azadeh Shakery, Igor Shalyminov, Jingbo Shang, Mingyue Shang, Lifeng Shang, Ori Shapira, Matthew Shardlow, Rebecca Sharp, Lanbo She, Ravi Shekhar, Jiaming Shen, Dinghan Shen, Baoxu Shi, Yunzhou Shi, Bei Shi, xiaodong shi, Hailin Shi, Peng Shi, Tianze Shi, Tomohide Shibata, Robik Shrestha, Manish Shrivastava, Lei Shu, Alexander Shvets, Vered Shwartz, Suzanna Sia, Carina Silberer, João Silva, Fabrizio Silvestri, Stefano Silvestri, Yanchuan Sim, Patrick Simianer, Dan Simonson, Kiril Simov, Edwin Simpson, Matthew Sims, Alberto Simões, Amando Jr. Singun, Steve Skiena, Andrii Skliar, Neil Smalheiser, Noah A. Smith, Otakar Smrž, Alison Sneyd, Artem Sokolov, Luca Soldaini, Hyun-Je Song, Ruihua Song, Wei Song, Linfeng Song, Yangqiu Song, Kaisong Song, Radu Soricut, Aitor Soroa, Ionut-Teodor Sorodoc, Daniil Sorokin, Matthias Sperber, Andreas Spitz, Richard Sproat, Balaji Vasan Srinivasan, Shashank Srivastava, Felix Stahlberg, Miloš Stanojević, Gabriel Stanovsky, Mark Steedman, Veselin Stoyanov, Karl Stratos, Kristina Striegnitz, Jannik Strötgen, Will Styler, Shang-Yu Su, Keh-Yih Su, Yu Su, Pei-Hao Su, Rajen Subba, Nishant Subramani, Katsuhito Sudoh, Kazunari Sugiyama, Alane Suhr, Elior Sulem, Md Arafat Sultan, Fei Sun, Shuo Sun, Yibo Sun, Ming Sun, Le Sun, Hanna Suominen, Mihai Surdeanu, Hisami Suzuki, Jun Suzuki, Swabha Swayamdipta, Ida Szubert, Jeniya Tabassum, Kaveh Taghipour, Hiroya Takamura, Sho Takase, David Talbot, Alon Talmor, Partha Talukdar, Akihiro Tamura, Jiwei Tan, Liling Tan, Hao Tan, Luchen Tan, Niket Tandon, Shuai Tang, Jiliang Tang, Raphael Tang, Duyu Tang, Siliang Tang, Jintao Tang, Hao Tang, Jian Tang, Gongbo Tang, Fei Tao, Chongyang Tao, Yi Tay, Selma Tekir, Damien Teney, Zhiyang Teng, Jesse Thomason, Sam Thomson, Ran Tian, Fei Tian, Jörg Tiedemann, Christoph Tillmann, Takenobu Tokunaga, Gaurav Singh Tomar, Marc Tomlinson, Sara Tonelli, Kentaro Torisawa, Paolo Torroni, Ke Tran, Quan Hung Tran, Trang Tran, Trieu Trinh, Rocco Tripodi, Adam Trischler, Chen-Tse Tsai, Yao-Hung Hubert Tsai, Yu Tsao, Reut Tsarfaty, Bo-Hsiang Tseng, Yuen-Hsien Tseng, Lifu Tu, Kewei Tu, Don Tuggener, Marco Turchi, Ferhan Ture, Francis Tyers, Kateryna Tymoshenko, Nicola Ueffing, Stefan Ultes, Lyle Ungar, Kartikeya Upasani, L. Alfonso Urena Lopez, Dmitry Ustalov, Masao Utiyama, Antonio Uva, Sowmya Vajjala, Tim Van de Cruys, Rob van der Goot, Chris van der Lee, Lonneke van der Plas, Benjamin Van Durme, Emiel van Miltenburg, Marten van Schijndel, Vincent Vandeghinste, Keith VanderLinden, Clara Vania, Shikhar Vashishth, Eva Maria Vecchi, Sumithra Velupillai, Alakananda Vempala, Sriram Venkatapathy, Ashish Venugopal, Subhashini Venugopalan, Suzan Verberne, Marc Verhagen, Rohil Verma, Yannick Versley, Guido Vetere, Natalia Viani, David Vilar, David Vilares, Jesús Vilares, Martin Villalba, Serena Villata, Aline Villavicencio, Jacky Visser, Elena Voita, Clare Voss, Thuy Vu, Ngoc Thang Vu, Slobodan Vucetic, Ivan Vulić, V.G.Vinod Vydiswaran, Henning Wachsmuth, Byron C. Wallace, Eric Wallace, Matthew Walter, Stephen Wan, Qing Wang, Shuai Wang, Wei Wang, Shaolei Wang, Hong Wang, Shuohang Wang, Han Wang, Longyue Wang, Xinyi Wang, Tong Wang, Chao Wang, Rui Wang, Xing Wang, Chuan Wang, Wenya Wang, Mingxuan Wang, Wen Wang, Yizhong Wang, Baoxun Wang, Chenguang

Wang, Dingquan Wang, Zhongqing Wang, Shufan Wang, Xin Wang, Cheng Wang, Hsin-Min Wang, Qingyun Wang, Wenhui Wang, Daling Wang, Zhichun Wang, Hai Wang, Yequan Wang, Wenlin Wang, Jakub Waszczuk, Shinji Watanabe, Ingmar Weber, Zhongyu Wei, Wei Wei, Gerhard Weikum, Ralph Weischedel, Johannes Welbl, Charles Welch, Tsung-Hsien Wen, Robert West, Aaron Steven White, Spencer Whitehead, Gregor Wiedemann, Michael Wiegand, Sarah Wiegreffe, John Wieting, Derry Tanti Wijaya, Gijs Wijnholds, Adina Williams, Tom Williams, Jake Williams, Shomir Wilson, Steven Wilson, Colin Wilson, Sam Wiseman, Guillaume Wisniewski, Travis Wolfe, Tak-Lam Wong, Derek F. Wong, Alina Wróblewska, Lijun Wu, Qi Wu, Zhaohui Wu, Mengyue Wu, Jiawei Wu, Xianchao Wu, Chien-Sheng Wu, Yu Wu, Fangzhao Wu, wei wu, Lingfei Wu, Hua Wu, Joern Wuebker, Yunqing Xia, Rong Xiang, Yanghua Xiao, Tong Xiao, Xinyan Xiao, Boyi Xie, Ruobing Xie, Shasha Xie, Jiateng Xie, Pengtao Xie, Qizhe Xie, Yingwei Xin, Chen Xing, Chenyan Xiong, Wenhan Xiong, Jia Xu, Jun Xu, Hainan Xu, Feiyu Xu, jiaming xu, Kun Xu, Tong Xu, Ruifeng Xu, Hu Xu, Wenduan Xu, Wei Xue, Yadollah Yaghoobzadeh, Takehiro Yamamoto, Rui Yan, Zhaojun Yang, Liu Yang, Cheng Yang, Shaohua Yang, Yaqin Yang, Jie Yang, Zichao Yang, Wei Yang, Qian Yang, Zi Yang, Yujiu Yang, Grace Hui Yang, Pengcheng Yang, Liner Yang, Baosong Yang, Yi Yang, Zhilin Yang, Diyi Yang, Helen Yannakoudakis, tae yano, Jin-ge Yao, Wenlin Yao, Ting Yao, Mahsa Yarmohammadi, Mark Yatskar, Lana Yeganova, Reyyan Yeniterzi, Seid Yimam, Qingyu Yin, Wenpeng Yin, Dawei Yin, Pengcheng Yin, Anssi Yli-Jyrä, Sho Yokoi, Seunghyun Yoon, Masaharu Yoshioka, Xiaodong Yu, Zhou Yu, Kai Yu, Bei Yu, Mo Yu, Tao Yu, Ning Yu, Dian Yu, Liang-Chih Yu, Jianfei Yu, François Yvon, Manzil Zaheer, Nasser Zalmout, Roberto Zamparelli, Marcos Zampieri, Guido Zarrella, Rowan Zellers, Daojian Zeng, Jiali Zeng, Xingshan Zeng, Chrysoula Zerva, Luke Zettlemoyer, Deniz Zeyrek, Feifei Zhai, Shuang (Sophie) Zhai, Richong Zhang, Yuhao Zhang, Wei Zhang, Yi Zhang, Yongfeng Zhang, Rui Zhang, Zhe Zhang, Sheng Zhang, Peng Zhang, Fuzheng Zhang, Yu Zhang, Hao Zhang, Xingxing Zhang, Yizhe Zhang, Lei Zhang, Jingyi Zhang, Tongtao Zhang, Chao Zhang, Yuchen Zhang, Meng Zhang, Biao Zhang, Boliang Zhang, Jiacheng Zhang, Qi Zhang, Yuan Zhang, Haichao Zhang, Xuan Zhang, Zhirui Zhang, Justine Zhang, Xiaodong Zhang, Dongdong Zhang, Yi Zhang, Zhou Zhao, Ran Zhao, Lin Zhao, Hai Zhao, Yang Zhao, Bing Zhao, Zhengli Zhao, Tiancheng Zhao, Kai Zhao, Dongyan Zhao, Sendong Zhao, Li Zhao, Guoqing Zheng, Alisa Zhila, Victor Zhong, Junsheng Zhou, Yu Zhou, Joey Tianyi Zhou, Guangyou Zhou, Deyu ZHOU, Qingyu Zhou, Ben Zhou, Wenxuan Zhou, Hao Zhou, Long Zhou, Hao Zhou, Li Zhou, Muhua Zhu, Hao Zhu, Lixing Zhu, Chenguang Zhu, Su Zhu, Huaiyu Zhu, Haichao Zhu, Junnan Zhu, Ayah Zirikly, Michael Zock, Chengqing Zong, Shi Zong, Markus Zopf, Arkaitz Zubiaga, Özlem Çetinoğlu, Çağrı Çöltekin, Diarmuid Ó Séaghdha, Robert Östling, Arzucan Özgür, Lilja Øvrelid, Gözde Gül Şahin, Jan Šnajder

**Outstanding ACs and Reviewers**

We would like to recognize the following outstanding Area Chairs.

Aliaksei Severyn, Andreas Vlachos, Andrew Yates, Ann Clifton, Annie Louis, Anya Belz, Barbara Plank, Chin-Yew Lin, Dan Goldwasser, Danqi Chen, Danushka Bollegala, David Jurgens, George Foster, Gholamreza Haffari, Isabelle Augenstein, Jacob Andreas, Jun Suzuki, Kemal Oflazer, Kevin Small, Laura Rimell, Leon Bergen, Li Dong, Loïc Barrault, Lu Wang, Marianna Apidianaki, Markus Dreyer, Matt Gardner, Michael Strube, Minlie Huang, Nanyun Peng, Nikolaos Aletras, Qin Lu, Quan Wang, Sameer Singh, Shafiq Joty, Shay B. Cohen, Shou-de Lin, Sujith Ravi, Suzanne Stevenson, Wei Gao, William Yang Wang, Xiaodan Zhu, Yoav Artzi, Yoav Goldberg, Yulia Tsvetkov

We would also like to recognize the following outstanding reviewers.

Abhijit Mishra, Adhiguna Kuncoro, Adina Williams, Ahmed Abdelali, Ahmed Hassan Awadallah, Aishwarya Padmakumar, Akihiro Tamura, Ákos Kádár, Alexander Rush, Aline Villavicencio, Amitava Das, Ana Marasovic, Ana Marasovic, Andrew Rosenberg, Angela Fan, Angeliki Lazaridou, Annemarie Friedrich, Annette frank, Antoine Bosselut, Antonio Uva, Artem Sokolov, Asad Abdi, Asad Sayeed, Asif ikbal, Barend Beekhuizen, Ben Bogin, Benjamin Börschinger, Benjamin Marie, Benoit Crabbé, Bhuwan Dhingra, Bill Yuchen Lin, Boyi Xie, Brendan O'Connor, Bruno Martins, Çağrı Çöltekin, Caio Filippo Corro, Carolin Lawrence, Chandler May, Chao Wang, Cheng Wang, Chenliang Li, Chester Holtz, Chris Kedzie, Christian M. Meyer, Christopher Potts, Chunchuan Lv, Cindy Xinyi Wang, Dagmar Gromann, Daichi Mochihashi, Dallas Card, Damien Teney, Dan Garrette, Dan Simonson, Daniel Hershcovich, Daniel Loureiro, Daniel Preoţiuc-Pietro, Daniele Bonadiman, Dario Bertero, David Kauchak, David Mortensen, David Reitter, David Vilares, Debanjan Ghosh, Deepanway Ghosal, Denis Newman-Griffis, Di Jin, Diego Moussallem, Diyi Yang, Djamé Seddah, Djamé Seddah, Don Tuggener, Edwin Simpson, Elena Voita, Elior Sulem, Elisa Ferracane, Elizabeth Salesky, Ella Rabinovich, Ellie Pavlick, Emiel van Miltenburg, Emily Bender, Eric Fosler-Lussier, Fernando Alva-Manchego, Francesco Corcoglioniti, Gabriel Stanovsky, Garrent Nicolai, Georgio Magri, Gijs Wijnholds, Goran Glavaš, Graham Neubig, Greg Durrett, Greg Durrett, Guanghui Qin, Guillaume Wisniewski, Guoqing Zheng, Hai Leong Chieu, Hai Wang, Hainan Xu, Hao Peng, Hao Tang, Hao Tang, Hayato Kobayashi, Henning Wachsmuth, Hiroki Ouchi, Hongyuan Mei, Hugo Gonçalo Oliveira, Hyeju Jang, Iacer Calixto, Ioannis Konstas, Ivan Vulic, Jack Hessel, Jacky Visser, Jannik Strötgen, Jason Naradowsky, Jeff Dalton, Jessica Ouyang, Jing Ma, Jinge Yao, John Lawrence, Jonathan K. Kummerfeld, Jonathan May, Jordan Boyd-Graber, Joseph Barrow, Julian Michael, Justine Zhang, Kazunari Sugiyama, Kenneth Heafield, Kevin Gimpel, Kirk Roberts, Klinton Bicknell, Kristen Johnson, Kristina Striegnitz, Kun Xu, Kyle Gorman, Laura Burdick, Laura Pérez-Mayos, Lea Frermann, Lifu Tu, Lijun Wu, Lucy H. Lin, Maarten Sap, Mahbub Rahman, Malvina Nissim, Manoj Acharya, Marco Damonte, Marco Turchi, Marcos Zampieri, Mariana Neves, Mark Finlayson, Martín Pereira-Fariña, Martha Palmer, Martin Villalba, Masaaki Nishino, Massimo Poesio, Mathieu Constant, Matthew Richardson, Matthew Walter, Maxime Peyrard, Melissa Roemmele, Meng Qu, Micha Elsner, Michael Gamon, Michael Wiegand, Mikel Artetxe, Mikhail Khodak, Minjoon Seo, Mo Yu, Moontae Lee, Mrinmaya Sachan, Muhammad Rafi, Muhao Chen, Nafise Sadat Moosavi, Naoki Otani, Nasser Zalmout, Natalie Ahn, nelson liu, Nguyen Xuan Khanh, Nikhil Krishnaswamy, Nikita Haduong, Nikita Nangia, Nikolaos Pappas, Noah Smith, Ofir Press, Omer Levy, Orphee De Clercq, Otakar Smrz, Panupong Pasupat, paola merlo, Pedro Rodriguez, Peng Qi, Pengcheng Yin, Peter Jansen, Philippe Muller, Piji Li, Prafulla Kumar Choubey, Pranava Madhyastha, Qun Liu, Ramakanth Pasunuru, Raquel Fernández, Ravi Shekhar, Reinald Kim Amplayo, Reut Tsarfaty, Richard Johansson, Rico Sennrich, Rob van der Goot, Robin Jia, Rotem Dror, Sachin Mehta, Sachin Mehta, Salam Khalifa, Sam Wiseman, Samuel R. Bowman, Sarvnaz Karimi, Sebastian Gehrmann, Selma Tekir, Sewon Min, Shankar Kumar, Sharid Loaiciga, Shikhar Murty, Shiran Dudy, Sho Takase, Shoaib Jameel, Shuai Tang, Simon Mille, Siva Reddy, Sorami Hisamoto, Sosuke Kobayashi, Spencer Whitehead, stefano silvestri, Stefanos Angelidis, Steven Wilson, Suzanna Sia, Tahira Naseem, Taku Kudo, Tal Linzen, Tanmoy Chakraborty, Tetsuji Nakagawa, Thomas Effland, Thuy Vu, Tianze Shi, Timothy Baldwin, Tobias Falke, Tom Kwiatkowski, Tom Williams, Trapit Bansal, Travis Wolfe, Tristan Miller, Udo Hahn, Valts Blukis, Victor Zhong, Wenlin Wang, Wenxuan Zhou, Wilker Ferreira Aziz, Will Styler, Xiaodong Liu, Xilun Chen, Xinya Du, Xuan Zhang, Yang Liu, Yejin Choi, Yiming Cui, Yitong Li, Yonatan Belinkov, Yoon Kim, Yova Kementched-

jhieva, Yue Dong, Yufang Hou, Yuhao Zhang, Yujiu Yang, Yulia Grishina, Yuxuan Lai, Yves Scherrer, Zhaochun Ren

# Invited Speaker: Meeyoung Cha, KAIST
## Current Challenges in Computational Social Science

**Abstract:** Artificial intelligence (AI) is reshaping business and science. Computational social science is an interdisciplinary field that solves complex societal problems by adopting AI-driven methods, processes, algorithms, and systems on data of various forms. This talk will review some of the latest advances in the research that focuses on fake news and legal liability. I will first discuss the structural, temporal, and linguistic traits of fake news propagation. One emerging challenge here is the increasing use of automated bots to generate and propagate false information. I will also discuss the current issues on the legal liability of AI and robots, particularly on how to regulate them (e.g., moral machine, punishment gap). This talk will suggest new opportunities to tackle these problems.

**Bio:** Meeyoung Cha is an associate professor at the School of Computing in KAIST. Dr. Cha's research interests are in analyzing complex network systems, including web and social media. Her research in the field of data science, artificial intelligence, and computational social science has gained more than 12,000 citations based on Google Scholar and has received the best paper awards at ACM IMC and ICWSM. Dr. Cha is currently in the editorial board member of PeerJ and ACM Transactions on Social Computing, and she has served as a program co-chair for ICWSM 2015. Dr. Cha has worked at Facebook's Data Science Team as a Visiting Professor in 2015−2016. Since 2019, she is jointly affiliated with the Institute for Basic Science (IBS) in Korea as a Chief Investigator.

# Invited Speaker: Kyunghyun Cho, New York University
## Curiosity-driven Journey into Neural Sequence Models

**Abstract:** In this talk, I take the audience on a tour of my earlier and recent experiences in building neural sequence models. I start from the earlier experience of using a recurrent net for sequence-to-sequence learning and talk about the attention mechanism. I discuss factors behind the success of these earlier approaches, and how these were embraced by the community even before they sota'd. I then move on to a more recent research direction in unconventional neural sequence models that automatically learn to decide on the order of generation.

**Bio:** Kyunghyun Cho is an associate professor of computer science and data science at New York University and a research scientist at Facebook AI Research. He was a postdoctoral fellow at University of Montreal until summer 2015 under the supervision of Prof. Yoshua Bengio, and received PhD and MSc degrees from Aalto University early 2014 under the supervision of Prof. Juha Karhunen, Dr. Tapani Raiko and Dr. Alexander Ilin. He tries his best to find a balance among machine learning, natural language processing, and life, but almost always fails to do so.

# Invited Speaker: Noam Slonim, IBM Haifa Research Lab

## Project Debater - How Persuasive can a Computer be?

**Abstract:** Project Debater is the first AI system that can meaningfully debate a human opponent. The system, an IBM Grand Challenge, is designed to build coherent, convincing speeches on its own, as well as provide rebuttals to the opponent's main arguments. In February 2019, Project Debater competed against Harish Natarajan, who holds the world record for most debate victories, in an event held in San Francisco that was broadcasted live world-wide. In this talk I will tell the story of Project Debater, from conception to a climatic final event, describe its underlying technology, and discuss how it can be leveraged for advancing decision making and critical thinking.

**Bio:** Noam Slonim is a Distinguished Engineer at IBM Research AI. He received his doctorate from the Interdisciplinary Center for Neural Computation at the Hebrew University and held a post-doc position at the Genomics Institute at Princeton University. During his PhD, Noam received the best paper award in UAI and ECIR, and the best presentation award at SIGIR. Noam joined the IBM Haifa Research Lab in 2007, and in 2011 he proposed to develop Project Debater. He has been serving as the Principal Investigator of the project since then. Noam published around 60 peer reviewed articles, focusing on the last few years on advancing the emerging field of Computational Argumentation. Finally, Noam used to have a secondary career as a TV script writer. Coincidentally, or not, in a sitcom he co-created back in 1998, the last episode was focused on competitive debates.

# Table of Contents

*Attending to Future Tokens for Bidirectional Sequence Generation*
Carolin Lawrence, Bhushan Kotnis and Mathias Niepert . . . . . 1

*Attention is not not Explanation*
Sarah Wiegreffe and Yuval Pinter . . . . . 11

*Practical Obstacles to Deploying Active Learning*
David Lowell, Zachary C. Lipton and Byron C. Wallace . . . . . 21

*Transfer Learning Between Related Tasks Using Expected Label Proportions*
Matan Ben Noach and Yoav Goldberg . . . . . 31

*Knowledge Enhanced Contextual Word Representations*
Matthew E. Peters, Mark Neumann, Robert Logan, Roy Schwartz, Vidur Joshi, Sameer Singh and Noah A. Smith . . . . . 43

*How Contextual are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings*
Kawin Ethayarajh . . . . . 55

*Room to Glo: A Systematic Comparison of Semantic Change Detection Approaches with Word Embeddings*
Philippa Shoemark, Farhana Ferdousi Liza, Dong Nguyen, Scott Hale and Barbara McGillivray 66

*Correlations between Word Vector Sets*
Vitalii Zhelezniak, April Shen, Daniel Busbridge, Aleksandar Savkov and Nils Hammerla . . . . . 77

*Game Theory Meets Embeddings: a Unified Framework for Word Sense Disambiguation*
Rocco Tripodi and Roberto Navigli . . . . . 88

*Guided Dialog Policy Learning: Reward Estimation for Multi-Domain Task-Oriented Dialog*
Ryuichi Takanobu, Hanlin Zhu and Minlie Huang . . . . . 100

*Multi-hop Selector Network for Multi-turn Response Selection in Retrieval-based Chatbots*
Chunyuan Yuan, Wei Zhou, Mingming Li, Shangwen Lv, Fuqing Zhu, Jizhong Han and Songlin Hu . . . . . 111

*MoEL: Mixture of Empathetic Listeners*
Zhaojiang Lin, Andrea Madotto, Jamin Shin, Peng Xu and Pascale Fung . . . . . 121

*Entity-Consistent End-to-end Task-Oriented Dialogue System with KB Retriever*
Libo Qin, Yijia Liu, Wanxiang Che, Haoyang Wen, Yangming Li and Ting Liu . . . . . 133

*Building Task-Oriented Visual Dialog Systems Through Alternative Optimization Between Dialog Policy and Language Generation*
Mingyang Zhou, Josh Arnold and Zhou Yu . . . . . 143

*DialogueGCN: A Graph Convolutional Neural Network for Emotion Recognition in Conversation*
Deepanway Ghosal, Navonil Majumder, Soujanya Poria, Niyati Chhaya and Alexander Gelbukh 154

*Knowledge-Enriched Transformer for Emotion Detection in Textual Conversations*
Peixiang Zhong, Di Wang and Chunyan Miao . . . . . 165

*Interpretable Relevant Emotion Ranking with Event-Driven Attention*
Yang Yang, Deyu ZHOU, Yulan He and Meng Zhang . . . . . 177

*Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects*
Jianmo Ni, Jiacheng Li and Julian McAuley . . . . . 188

*Using Customer Service Dialogues for Satisfaction Analysis with Context-Assisted Multiple Instance Learning*
Kaisong Song, Lidong Bing, Wei Gao, Jun Lin, Lujun Zhao, Jiancheng Wang, Changlong Sun, Xiaozhong Liu and Qiong Zhang . . . . . 198

*Leveraging Dependency Forest for Neural Medical Relation Extraction*
Linfeng Song, Yue Zhang, Daniel Gildea, Mo Yu, Zhiguo Wang and jinsong su . . . . . 208

*Open Relation Extraction: Relational Knowledge Transfer from Supervised Data to Unsupervised Data*
Ruidong Wu, Yuan Yao, Xu Han, Ruobing Xie, Zhiyuan Liu, Fen Lin, Leyu Lin and Maosong Sun 219

*Improving Relation Extraction with Knowledge-attention*
Pengfei Li, Kezhi Mao, Xuefeng Yang and Qi Li . . . . . 229

*Jointly Learning Entity and Relation Representations for Entity Alignment*
Yuting Wu, Xiao Liu, Yansong Feng, Zheng Wang and Dongyan Zhao . . . . . 240

*Tackling Long-Tailed Relations and Uncommon Entities in Knowledge Graph Completion*
Zihao Wang, Kwunping Lai, Piji Li, Lidong Bing and Wai Lam . . . . . 250

*Low-Resource Name Tagging Learned with Weakly Labeled Data*
Yixin Cao, Zikun Hu, Tat-seng Chua, Zhiyuan Liu and Heng Ji . . . . . 261

*Learning Dynamic Context Augmentation for Global Entity Linking*
Xiyuan Yang, Xiaotao Gu, Sheng Lin, Siliang Tang, Yueting Zhuang, Fei Wu, Zhigang Chen, Guoping Hu and Xiang Ren . . . . . 271

*Open Event Extraction from Online Text using a Generative Adversarial Network*
Rui Wang, Deyu ZHOU and Yulan He . . . . . 282

*Learning to Bootstrap for Entity Set Expansion*
Lingyong Yan, Xianpei Han, Le Sun and Ben He . . . . . 292

*Multi-Input Multi-Output Sequence Labeling for Joint Extraction of Fact and Condition Tuples from Scientific Text*
Tianwen Jiang, Tong Zhao, Bing Qin, Ting Liu, Nitesh Chawla and Meng Jiang . . . . . 302

*Cross-lingual Structure Transfer for Relation and Event Extraction*
Ananya Subburathinam, Di Lu, Heng Ji, Jonathan May, Shih-Fu Chang, Avirup Sil and Clare Voss 313

*Uncover the Ground-Truth Relations in Distant Supervision: A Neural Expectation-Maximization Framework*
Junfan Chen, Richong Zhang, Yongyi Mao, Hongyu Guo and Jie Xu . . . . . 326

*Doc2EDAG: An End-to-End Document-level Framework for Chinese Financial Event Extraction*
Shun Zheng, Wei Cao, Wei Xu and Jiang Bian . . . . . 337

*Event Detection with Trigger-Aware Lattice Neural Network*
Ning Ding, Ziran Li, Zhiyuan Liu, Haitao Zheng and Zibo Lin . . . . . 347

*A Boundary-aware Neural Model for Nested Named Entity Recognition*
Changmeng Zheng, Yi Cai, Jingyun Xu, Ho-fung Leung and Guandong Xu . . . . . 357

*Learning the Extraction Order of Multiple Relational Facts in a Sentence with Reinforcement Learning*
Xiangrong Zeng, Shizhu He, Daojian Zeng, Kang Liu, Shengping Liu and Jun Zhao . . . . . 367

*CaRe: Open Knowledge Graph Embeddings*
Swapnil Gupta, Sreyash Kenkre and Partha Talukdar . . . . . 378

*Self-Attention Enhanced CNNs and Collaborative Curriculum Learning for Distantly Supervised Relation Extraction*
Yuyun Huang and Jinhua Du . . . . . 389

*Neural Cross-Lingual Relation Extraction Based on Bilingual Word Embedding Mapping*
Jian Ni and Radu Florian . . . . . 399

*Leveraging 2-hop Distant Supervision from Table Entity Pairs for Relation Extraction*
xiang deng and Huan Sun . . . . . 410

*EntEval: A Holistic Evaluation Benchmark for Entity Representations*
Mingda Chen, Zewei Chu, Yang Chen, Karl Stratos and Kevin Gimpel . . . . . 421

*Joint Event and Temporal Relation Extraction with Shared Representations and Structured Prediction*
Rujun Han, Qiang Ning and Nanyun Peng . . . . . 434

*Hierarchical Text Classification with Reinforced Label Assignment*
Yuning Mao, Jingjing Tian, Jiawei Han and Xiang Ren . . . . . 445

*Investigating Capsule Network and Semantic Feature on Hyperplanes for Text Classification*
Chunning Du, Haifeng Sun, Jingyu Wang, Qi Qi, Jianxin Liao, Chun Wang and Bing Ma . . . . . 456

*Label-Specific Document Representation for Multi-Label Text Classification*
Lin Xiao, Xin Huang, Boli Chen and Liping Jing . . . . . 466

*Hierarchical Attention Prototypical Networks for Few-Shot Text Classification*
Shengli Sun, Qingfeng Sun, Kevin Zhou and Tengchao Lv . . . . . 476

*Many Faces of Feature Importance: Comparing Built-in and Post-hoc Feature Importance in Text Classification*
Vivian Lai, Zheng Cai and Chenhao Tan . . . . . 486

*Enhancing Local Feature Extraction with Global Representation for Neural Text Classification*
Guocheng Niu, Hengru Xu, Bolei He, Xinyan Xiao, Hua Wu and Sheng GAO . . . . . 496

*Latent-Variable Generative Models for Data-Efficient Text Classification*
Xiaoan Ding and Kevin Gimpel . . . . . 507

*PaRe: A Paper-Reviewer Matching Approach Using a Common Topic Space*
Omer Anjum, Hongyu Gong, Suma Bhat, Wen-Mei Hwu and JinJun Xiong . . . . . 518

*Linking artificial and human neural representations of language*
Jon Gauthier and Roger Levy . . . . . 529

*Neural Text Summarization: A Critical Evaluation*
Wojciech Kryscinski, Nitish Shirish Keskar, Bryan McCann, Caiming Xiong and Richard Socher 540

*Neural data-to-text generation: A comparison between pipeline and end-to-end architectures*
Thiago Castro Ferreira, Chris van der Lee, Emiel van Miltenburg and Emiel Krahmer . . . . . 552

*MoverScore: Text Generation Evaluating with Contextualized Embeddings and Earth Mover Distance*
Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M. Meyer and Steffen Eger . . . . . 563

*Select and Attend: Towards Controllable Content Selection in Text Generation*
Xiaoyu Shen, Jun Suzuki, Kentaro Inui, Hui Su, Dietrich Klakow and Satoshi Sekine . . . . . 579

*Sentence-Level Content Planning and Style Specification for Neural Text Generation*
Xinyu Hua and Lu Wang . . . . . 591

*Translate and Label! An Encoder-Decoder Approach for Cross-lingual Semantic Role Labeling*
Angel Daza and Anette Frank . . . . . 603

*Syntax-Enhanced Self-Attention-Based Semantic Role Labeling*
Yue Zhang, Rui Wang and Luo Si . . . . . 616

*VerbAtlas: a Novel Large-Scale Verbal Semantic Resource and Its Application to Semantic Role Labeling*
Andrea Di Fabio, Simone Conia and Roberto Navigli . . . . . 627

*Parameter-free Sentence Embedding via Orthogonal Basis*
Ziyi Yang, Chenguang Zhu and Weizhu Chen . . . . . 638

*Evaluation Benchmarks and Learning Criteria for Discourse-Aware Sentence Representations*
Mingda Chen, Zewei Chu and Kevin Gimpel . . . . . 649

*Extracting Possessions from Social Media: Images Complement Language*
Dhivya Chinnappa, Srikala Murugan and Eduardo Blanco . . . . . 663

*Learning to Speak and Act in a Fantasy Text Adventure Game*
Jack Urbanek, Angela Fan, Siddharth Karamcheti, Saachi Jain, Samuel Humeau, Emily Dinan, Tim Rocktäschel, Douwe Kiela, Arthur Szlam and Jason Weston . . . . . 673

*Help, Anna! Visual Navigation with Natural Multimodal Assistance via Retrospective Curiosity-Encouraging Imitation Learning*
Khanh Nguyen and Hal Daumé III . . . . . 684

*Incorporating Visual Semantics into Sentence Representations within a Grounded Space*
Patrick Bordes, Eloi Zablocki, Laure Soulier, Benjamin Piwowarski and patrick Gallinari . . . . . 696

*Neural Naturalist: Generating Fine-Grained Image Comparisons*
Maxwell Forbes, Christine Kaeser-Chen, Piyush Sharma and Serge Belongie . . . . . 708

*Fine-Grained Evaluation for Entity Linking*
Henry Rosales-Méndez, Aidan Hogan and Barbara Poblete . . . . . 718

*Supervising Unsupervised Open Information Extraction Models*
Arpita Roy, Youngja Park, Taesung Lee and Shimei Pan . . . . . 728

*Neural Cross-Lingual Event Detection with Minimal Parallel Resources*
Jian Liu, Yubo Chen, Kang Liu and Jun Zhao . . . . . 738

*KnowledgeNet: A Benchmark Dataset for Knowledge Base Population*
Filipe Mesquita, Matteo Cannaviccio, Jordan Schmidek, Paramita Mirza and Denilson Barbosa 749

*Effective Use of Transformer Networks for Entity Tracking*
Aditya Gupta and Greg Durrett . . . . . 759

*Explicit Cross-lingual Pre-training for Unsupervised Machine Translation*
Shuo Ren, Yu Wu, Shujie Liu, Ming Zhou and Shuai Ma . . . . . 770

*Latent Part-of-Speech Sequences for Neural Machine Translation*
Xuewen Yang, Yingru Liu, Dongliang Xie, Xin Wang and Niranjan Balasubramanian . . . . . 780

*Improving Back-Translation with Uncertainty-based Confidence Estimation*
Shuo Wang, Yang Liu, Chao Wang, Huanbo Luan and Maosong Sun . . . . . 791

*Towards Linear Time Neural Machine Translation with Capsule Networks*
Mingxuan Wang . . . . . 803

*Modeling Multi-mapping Relations for Precise Cross-lingual Entity Alignment*
Xiaofei Shi and Yanghua Xiao . . . . . 813

*Supervised and Nonlinear Alignment of Two Embedding Spaces for Dictionary Induction in Low Resourced Languages*
Masud Moshtaghi . . . . . 823

*Beto, Bentz, Becas: The Surprising Cross-Lingual Effectiveness of BERT*
Shijie Wu and Mark Dredze . . . . . 833

*Iterative Dual Domain Adaptation for Neural Machine Translation*
Jiali Zeng, Yang Liu, Jinsong SU, yubing Ge, Yaojie Lu, Yongjing Yin and jiebo luo . . . . . 845

*Multi-agent Learning for Neural Machine Translation*
tianchi bi, hao xiong, Zhongjun He, Hua Wu and Haifeng Wang . . . . . 856

*Pivot-based Transfer Learning for Neural Machine Translation between Non-English Languages*
Yunsu Kim, Petre Petrov, Pavel Petrushkov, Shahram Khadivi and Hermann Ney . . . . . 866

*Context-Aware Monolingual Repair for Neural Machine Translation*
Elena Voita, Rico Sennrich and Ivan Titov . . . . . 877

*Multi-Granularity Self-Attention for Neural Machine Translation*
Jie Hao, Xing Wang, Shuming Shi, Jinfeng Zhang and Zhaopeng Tu . . . . . 887

*Improving Deep Transformer with Depth-Scaled Initialization and Merged Attention*
Biao Zhang, Ivan Titov and Rico Sennrich . . . . . 898

*A Discriminative Neural Model for Cross-Lingual Word Alignment*
Elias Stengel-Eskin, Tzu-ray Su, Matt Post and Benjamin Van Durme . . . . . 910

*One Model to Learn Both: Zero Pronoun Prediction and Translation*
Longyue Wang, Zhaopeng Tu, Xing Wang and Shuming Shi . . . . . 921

*Dynamic Past and Future for Neural Machine Translation*
Zaixiang Zheng, Shujian Huang, Zhaopeng Tu, XIN-YU DAI and Jiajun CHEN . . . . . 931

*Revisit Automatic Error Detection for Wrong and Missing Translation – A Supervised Approach*
Wenqiang Lei, Weiwen Xu, Ai Ti Aw, Yuanxin Xiang and Tat Seng Chua . . . . . 942

*Towards Understanding Neural Machine Translation with Word Importance*
Shilin He, Zhaopeng Tu, Xing Wang, Longyue Wang, Michael Lyu and Shuming Shi . . . . . 953

*Multilingual Neural Machine Translation with Language Clustering*
Xu Tan, Jiale Chen, Di He, Yingce Xia, Tao QIN and Tie-Yan Liu . . . . . 963

*Don't Forget the Long Tail! A Comprehensive Analysis of Morphological Generalization in Bilingual Lexicon Induction*
Paula Czarnowska, Sebastian Ruder, Edouard Grave, Ryan Cotterell and Ann Copestake . . . . . 974

*Pushing the Limits of Low-Resource Morphological Inflection*
Antonios Anastasopoulos and Graham Neubig . . . . . 984

*Cross-Lingual Dependency Parsing Using Code-Mixed TreeBank*
Meishan Zhang, Yue Zhang and Guohong Fu . . . . . 997

*Hierarchical Pointer Net Parsing*
Linlin Liu, Xiang Lin, Shafiq Joty, Simeng Han and Lidong Bing . . . . . 1007

*Semi-Supervised Semantic Role Labeling with Cross-View Training*
Rui Cai and Mirella Lapata . . . . . 1018

*Low-Resource Sequence Labeling via Unsupervised Multilingual Contextualized Representations*
Zuyi Bao, Rui Huang, Chen Li and Kenny Zhu . . . . . 1028

*A Lexicon-Based Graph Neural Network for Chinese NER*
Tao Gui, Yicheng Zou, Qi Zhang, Minlong Peng, Jinlan Fu, Zhongyu Wei and Xuanjing Huang 1040

*CM-Net: A Novel Collaborative Memory Network for Spoken Language Understanding*
Yijin Liu, Fandong Meng, Jinchao Zhang, Jie Zhou, Yufeng Chen and Jinan Xu . . . . . 1051

*Tree Transformer: Integrating Tree Structures into Self-Attention*
Yaushian Wang, Hung-Yi Lee and Yun-Nung Chen . . . . . 1061

*Semantic Role Labeling with Iterative Structure Refinement*
Chunchuan Lyu, Shay B. Cohen and Ivan Titov . . . . . 1071

*Entity Projection via Machine Translation for Cross-Lingual NER*
Alankar Jain, Bhargavi Paranjape and Zachary C. Lipton . . . . . 1083

*A Bayesian Approach for Sequence Tagging with Crowds*
Edwin D. Simpson and Iryna Gurevych . . . . . 1093

*A systematic comparison of methods for low-resource dependency parsing on genuinely low-resource languages*
Clara Vania, Yova Kementchedjhieva, Anders Søgaard and Adam Lopez . . . . . 1105

*Target Language-Aware Constrained Inference for Cross-lingual Dependency Parsing*
Tao Meng, Nanyun Peng and Kai-Wei Chang . . . . . 1117

*Look-up and Adapt: A One-shot Semantic Parser*
Zhichu Lu, Forough Arabshahi, Igor Labutov and Tom Mitchell . . . . . 1129

*Similarity Based Auxiliary Classifier for Named Entity Recognition*
Shiyuan Xiao, Yuanxin Ouyang, Wenge Rong, Jianxin Yang and Zhang Xiong . . . . . 1140

*Variable beam search for generative neural parsing and its relevance for the analysis of neuro-imaging signal*
Benoit Crabbé, Murielle Fabre and Christophe Pallier . . . . . 1150

*Are We Modeling the Task or the Annotator? An Investigation of Annotator Bias in Natural Language Understanding Datasets*
Mor Geva, Yoav Goldberg and Jonathan Berant . . . . . 1161

*Robust Text Classifier on Test-Time Budgets*
Md Rizwan Parvez, Tolga Bolukbasi, Kai-Wei Chang and Venkatesh Saligrama . . . . . 1167

*Commonsense Knowledge Mining from Pretrained Models*
Joe Davison, Joshua Feldman and Alexander Rush . . . . . 1173

*RNN Architecture Learning with Sparse Regularization*
Jesse Dodge, Roy Schwartz, Hao Peng and Noah A. Smith . . . . . 1179

*Analytical Methods for Interpretable Ultradense Word Embeddings*
Philipp Dufter and Hinrich Schütze . . . . . 1185

*Investigating Meta-Learning Algorithms for Low-Resource Natural Language Understanding Tasks*
Zi-Yi Dou, Keyi Yu and Antonios Anastasopoulos . . . . . 1192

*Retrofitting Contextualized Word Embeddings with Paraphrases*
Weijia Shi, Muhao Chen, Pei Zhou and Kai-Wei Chang . . . . . 1198

*Incorporating Contextual and Syntactic Structures Improves Semantic Similarity Modeling*
Linqing Liu, Wei Yang, Jinfeng Rao, Raphael Tang and Jimmy Lin . . . . . 1204

*Neural Linguistic Steganography*
Zachary Ziegler, Yuntian Deng and Alexander Rush . . . . . 1210

*The Feasibility of Embedding Based Automatic Evaluation for Single Document Summarization*
Simeng Sun and Ani Nenkova . . . . . 1216

*Attention Optimization for Abstractive Document Summarization*
Min Gui, Junfeng Tian, Rui Wang and Zhenglu Yang . . . . . 1222

*Rewarding Coreference Resolvers for Being Consistent with World Knowledge*
Rahul Aralikatte, Heather Lent, Ana Valeria Gonzalez, Daniel Herschcovich, Chen Qiu, Anders Sandholm, Michael Ringaard and Anders Søgaard . . . . . 1229

*An Empirical Study of Incorporating Pseudo Data into Grammatical Error Correction*
Shun Kiyono, Jun Suzuki, Masato Mita, Tomoya Mizumoto and Kentaro Inui . . . . . 1236

*A Multilingual Topic Model for Learning Weighted Topic Links Across Corpora with Low Comparability*
Weiwei Yang, Jordan Boyd-Graber and Philip Resnik . . . . . 1243

*Measure Country-Level Socio-Economic Indicators with Streaming News: An Empirical Study*
Bonan Min and Xiaoxi Zhao . . . . . 1249

*Towards Extracting Medical Family History from Natural Language Interactions: A New Dataset and Baselines*
Mahmoud Azab, Stephane Dadian, Vivi Nastase, Larry An and Rada Mihalcea . . . . . 1255

*Multi-task Learning for Natural Language Generation in Task-Oriented Dialogue*
Chenguang Zhu, Michael Zeng and Xuedong Huang . . . . . 1261

*Dirichlet Latent Variable Hierarchical Recurrent Encoder-Decoder in Dialogue Generation*
Min Zeng, Yisen Wang and Yuan Luo . . . . . 1267

*Semi-Supervised Bootstrapping of Dialogue State Trackers for Task-Oriented Modelling*
Bo-Hsiang Tseng, Marek Rei, Paweł Budzianowski, Richard Turner, Bill Byrne and Anna Korhonen 1273

*A Progressive Model to Enable Continual Learning for Semantic Slot Filling*
Yilin Shen, Xiangyu Zeng and Hongxia Jin . . . . . 1279

*CASA-NLU: Context-Aware Self-Attentive Natural Language Understanding for Task-Oriented Chatbots*
Arshit Gupta, Peng Zhang, Garima Lalwani and Mona Diab . . . . . 1285

*Sampling Matters! An Empirical Study of Negative Sampling Strategies for Learning of Matching Models in Retrieval-based Dialogue Systems*
Jia Li, Chongyang Tao, wei wu, Yansong Feng, Dongyan Zhao and Rui Yan . . . . . 1291

*Zero-shot Cross-lingual Dialogue Systems with Transferable Latent Variables*
Zihan Liu, Jamin Shin, Yan Xu, Genta Indra Winata, Peng Xu, Andrea Madotto and Pascale Fung 1297

*Modeling Multi-Action Policy for Task-Oriented Dialogues*
Lei Shu, Hu Xu, Bing Liu and Piero Molino . . . . . 1304

*An Evaluation Dataset for Intent Classification and Out-of-Scope Prediction*
Stefan Larson, Anish Mahendran, Joseph J. Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael A. Laurenzano, Lingjia Tang and Jason mars . . . . 1311

*Automatically Learning Data Augmentation Policies for Dialogue Tasks*
Tong Niu and Mohit Bansal . . . . . 1317

*uniblock: Scoring and Filtering Corpus with Unicode Block Information*
Yingbo Gao, Weiyue Wang and Hermann Ney . . . . . 1324

*Multilingual word translation using auxiliary languages*
Hagai Taitelbaum, Gal Chechik and Jacob Goldberger . . . . . 1330

*Towards Better Modeling Hierarchical Structure for Self-Attention with Ordered Neurons*
Jie Hao, Xing Wang, Shuming Shi, Jinfeng Zhang and Zhaopeng Tu . . . . . 1336

*Vecalign: Improved Sentence Alignment in Linear Time and Space*
Brian Thompson and Philipp Koehn . . . . . 1342

*Simpler and Faster Learning of Adaptive Policies for Simultaneous Translation*
Baigong Zheng, Renjie Zheng, Mingbo Ma and Liang Huang . . . . . 1349

*Adversarial Learning with Contextual Embeddings for Zero-resource Cross-lingual Classification and NER*
Phillip Keung, yichao lu and Vikas Bhardwaj . . . . . 1355

*Recurrent Positional Embedding for Neural Machine Translation*
Kehai Chen, Rui Wang, Masao Utiyama and Eiichiro Sumita . . . . . 1361

*Machine Translation for Machines: the Sentiment Classification Use Case*
amirhossein tebbifakhr, Luisa Bentivogli, Matteo Negri and Marco Turchi . . . . . 1368

*Investigating the Effectiveness of BPE: The Power of Shorter Sequences*
Matthias Gallé . . . . . 1375

*HABLex: Human Annotated Bilingual Lexicons for Experiments in Machine Translation*
Brian Thompson, Rebecca Knowles, Xuan Zhang, Huda Khayrallah, Kevin Duh and Philipp Koehn 1382

*Handling Syntactic Divergence in Low-resource Machine Translation*
Chunting Zhou, Xuezhe Ma, Junjie Hu and Graham Neubig . . . . . 1388

*Speculative Beam Search for Simultaneous Translation*
Renjie Zheng, Mingbo Ma, Baigong Zheng and Liang Huang . . . . . 1395

*Self-Attention with Structural Position Representations*
Xing Wang, Zhaopeng Tu, Longyue Wang and Shuming Shi . . . . . 1403

*Exploiting Multilingualism through Multistage Fine-Tuning for Low-Resource Neural Machine Translation*
Raj Dabre, Atsushi Fujita and Chenhui Chu . . . . . 1410

*Unsupervised Domain Adaptation for Neural Machine Translation with Domain-Aware Feature Embeddings*
Zi-Yi Dou, Junjie Hu, Antonios Anastasopoulos and Graham Neubig . . . . . 1417

*A Regularization-based Framework for Bilingual Grammar Induction*
Yong Jiang, Wenjuan Han and Kewei Tu . . . . . 1423

*Encoders Help You Disambiguate Word Senses in Neural Machine Translation*
Gongbo Tang, Rico Sennrich and Joakim Nivre . . . . . 1429

*Korean Morphological Analysis with Tied Sequence-to-Sequence Multi-Task Model*
Hyun-Je Song and Seong-Bae Park . . . . . 1436

*Efficient Convolutional Neural Networks for Diacritic Restoration*
Sawsan Alqahtani, Ajay Mishra and Mona Diab . . . . . 1442

*Improving Generative Visual Dialog by Answering Diverse Questions*
Vishvak Murahari, Prithvijit Chattopadhyay, Dhruv Batra, Devi Parikh and Abhishek Das . . . 1449

*Cross-lingual Transfer Learning with Data Selection for Large-Scale Spoken Language Understanding*
Quynh Do and Judith Gaspers . . . . . 1455

*Multi-Head Attention with Diversity for Learning Grounded Multilingual Multimodal Representations*
Po-Yao Huang, Xiaojun Chang and Alexander Hauptmann . . . . . 1461

*Decoupled Box Proposal and Featurization with Ultrafine-Grained Semantic Labels Improve Image Captioning and Visual Question Answering*
Soravit Changpinyo, Bo Pang, Piyush Sharma and Radu Soricut . . . . . 1468

*REO-Relevance, Extraness, Omission: A Fine-grained Evaluation for Image Captioning*
Ming Jiang, Junjie Hu, Qiuyuan Huang, Lei Zhang, Jana Diesner and Jianfeng Gao . . . . . 1475

*WSLLN:Weakly Supervised Natural Language Localization Networks*
Mingfei Gao, Larry Davis, Richard Socher and Caiming Xiong . . . . . 1481

*Grounding learning of modifier dynamics: An application to color naming*
Xudong Han, Philip Schulz and Trevor Cohn . . . . . 1488

*Robust Navigation with Language Pretraining and Stochastic Sampling*
Xiujun Li, Chunyuan Li, Qiaolin Xia, Yonatan Bisk, Asli Celikyilmaz, Jianfeng Gao, Noah A. Smith and Yejin Choi . . . . . 1494

*Towards Making a Dependency Parser See*
Michalina Strzyz, David Vilares and Carlos Gómez-Rodríguez . . . . . 1500

*Unsupervised Labeled Parsing with Deep Inside-Outside Recursive Autoencoders*
Andrew Drozdov, Patrick Verga, Yi-Pei Chen, Mohit Iyyer and Andrew McCallum . . . . . 1507

*Dependency Parsing for Spoken Dialog Systems*
Sam Davidson, Dian Yu and Zhou Yu . . . . . 1513

*Span-based Hierarchical Semantic Parsing for Task-Oriented Dialog*
Panupong Pasupat, Sonal Gupta, Karishma Mandyam, Rushin Shah, Mike Lewis and Luke Zettlemoyer . . . . . 1520

*Enhancing Context Modeling with a Query-Guided Capsule Network for Document-level Translation*
Zhengxin Yang, Jinchao Zhang, Fandong Meng, Shuhao Gu, Yang Feng and Jie Zhou . . . . . 1527

*Simple, Scalable Adaptation for Neural Machine Translation*
Ankur Bapna and Orhan Firat . . . . . 1538

*Controlling Text Complexity in Neural Machine Translation*
Sweta Agrawal and Marine Carpuat . . . . . 1549

*Investigating Multilingual NMT Representations at Scale*
Sneha Kudugunta, Ankur Bapna, Isaac Caswell and Orhan Firat . . . . . 1565

*Hierarchical Modeling of Global Context for Document-Level Neural Machine Translation*
Xin Tan, Longyin Zhang, Deyi Xiong and Guodong Zhou . . . . . 1576

*Cross-Lingual Machine Reading Comprehension*
Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Shijin Wang and Guoping Hu . . . . . 1586

*A Multi-Type Multi-Span Network for Reading Comprehension that Requires Discrete Reasoning*
Minghao Hu, Yuxing Peng, Zhen Huang and Dongsheng Li . . . . . 1596

*Neural Duplicate Question Detection without Labeled Training Data*
Andreas Rücklé, Nafise Sadat Moosavi and Iryna Gurevych . . . . . 1607

*Asking Clarification Questions in Knowledge-Based Question Answering*
Jingjing Xu, Yuechen Wang, Duyu Tang, Nan Duan, Pengcheng Yang, Qi Zeng, Ming Zhou and Xu SUN . . . . . 1618

*Multi-View Domain Adapted Sentence Embeddings for Low-Resource Unsupervised Duplicate Question Detection*
Nina Poerner and Hinrich Schütze . . . . . 1630

*Multi-label Categorization of Accounts of Sexism using a Neural Framework*
Pulkit Parikh, Harika Abburi, Pinkesh Badjatiya, Radhika Krishnan, Niyati Chhaya, Manish Gupta and Vasudeva Varma . . . . . 1642

*The Trumpiest Trump? Identifying a Subject's Most Characteristic Tweets*
Charuta Pethe and Steve Skiena . . . . . 1653

*Finding Microaggressions in the Wild: A Case for Locating Elusive Phenomena in Social Media Posts*
Luke Breitfeller, Emily Ahn, David Jurgens and Yulia Tsvetkov . . . . . 1664

*Reinforced Product Metadata Selection for Helpfulness Assessment of Customer Reviews*
Miao Fan, Chao Feng, Mingming Sun and Ping Li . . . . . 1675

*Learning Invariant Representations of Social Media Users*
Nicholas Andrews and Marcus Bishop . . . . . 1684

*(Male, Bachelor) and (Female, Ph.D) have different connotations: Parallelly Annotated Stylistic Language Dataset with Multiple Personas*
Dongyeop Kang, Varun Gangal and Eduard Hovy . . . . . 1696

*Movie Plot Analysis via Turning Point Identification*
Pinelopi Papalampidi, Frank Keller and Mirella Lapata . . . . . 1707

*Latent Suicide Risk Detection on Microblog via Suicide-Oriented Word Embeddings and Layered Attention*
Lei Cao, Huijun Zhang, Ling Feng, Zihan Wei, Xin Wang, Ningyun Li and Xiaohao He . . . . . 1718

*Deep Ordinal Regression for Pledge Specificity Prediction*
Shivashankar Subramanian, Trevor Cohn and Timothy Baldwin . . . . . 1729

*Data-Efficient Goal-Oriented Conversation with Dialogue Knowledge Transfer Networks*
Igor Shalyminov, Sungjin Lee, Arash Eshghi and Oliver Lemon . . . . . 1741

*Multi-Granularity Representations of Dialog*
Shikib Mehri and Maxine Eskenazi . . . . . 1752

*Are You for Real? Detecting Identity Fraud via Dialogue Interactions*
Weikang Wang, Jiajun Zhang, Qian Li, Chengqing Zong and Zhifei Li . . . . . 1762

*Hierarchy Response Learning for Neural Conversation Generation*
Bo Zhang and Xiaoming Zhang . . . . . 1772

*Knowledge Aware Conversation Generation with Explainable Reasoning over Augmented Graphs*
zhibin liu, Zheng-Yu Niu, Hua Wu and Haifeng Wang . . . . . 1782

*Adaptive Parameterization for Neural Dialogue Generation*
Hengyi Cai, Hongshen Chen, Cheng Zhang, Yonghao Song, Xiaofang Zhao and Dawei Yin . . 1793

*Towards Knowledge-Based Recommender Dialog System*
Qibin Chen, Junyang Lin, Yichang Zhang, Ming Ding, Yukuo Cen, Hongxia Yang and Jie Tang 1803

*Structuring Latent Spaces for Stylized Response Generation*
Xiang Gao, Yizhe Zhang, Sungjin Lee, Michel Galley, Chris Brockett, Jianfeng Gao and Bill Dolan 1814

*Improving Open-Domain Dialogue Systems via Multi-Turn Incomplete Utterance Restoration*
Zhufeng Pan, Kun Bai, Yan Wang, Lianqiang Zhou and Xiaojiang Liu . . . . . 1824

*Unsupervised Context Rewriting for Open Domain Conversation*
Kun Zhou, Kai Zhang, Yu Wu, Shujie Liu and Jingsong Yu . . . . . 1834

*Dually Interactive Matching Network for Personalized Response Selection in Retrieval-Based Chatbots*
Jia-Chen Gu, Zhen-Hua Ling, Xiaodan Zhu and Quan Liu . . . . . 1845

*DyKgChat: Benchmarking Dialogue Generation Grounding on Dynamic Knowledge Graphs*
Yi-Lin Tuan, Yun-Nung Chen and Hung-yi Lee . . . . . 1855

*Retrieval-guided Dialogue Response Generation via a Matching-to-Generation Framework*
Deng Cai, Yan Wang, Wei Bi, Zhaopeng Tu, Xiaojiang Liu and Shuming Shi . . . . . 1866

*Scalable and Accurate Dialogue State Tracking via Hierarchical Sequence Generation*
Liliang Ren, Jianmo Ni and Julian McAuley . . . . . 1876

*Low-Resource Response Generation with Template Prior*
Ze Yang, wei wu, Jian Yang, Can Xu and zhoujun li . . . . . 1886

*A Discrete CVAE for Response Generation on Short-Text Conversation*
Jun Gao, Wei Bi, Xiaojiang Liu, Junhui Li, Guodong Zhou and Shuming Shi . . . . . 1898

*Who Is Speaking to Whom? Learning to Identify Utterance Addressee in Multi-Party Conversations*
Ran Le, Wenpeng Hu, Mingyue Shang, Zhenjun You, Lidong Bing, Dongyan Zhao and Rui Yan 1909

*A Semi-Supervised Stable Variational Network for Promoting Replier-Consistency in Dialogue Generation*
Jinxin Chang, Ruifang He, Longbiao Wang, Xiangyu Zhao, Ting Yang and Ruifang Wang . . . 1920

*Modeling Personalization in Continuous Space for Response Generation via Augmented Wasserstein Autoencoders*
Zhangming Chan, Juntao Li, Xiaopeng Yang, Xiuying Chen, Wenpeng Hu, Dongyan Zhao and Rui Yan . . . . . 1931

*Variational Hierarchical User-based Conversation Model*
JinYeong Bak and Alice Oh . . . . . 1941

*Recommendation as a Communication Game: Self-Supervised Bot-Play for Goal-oriented Dialogue*
Dongyeop Kang, Anusha Balakrishnan, Pararth Shah, Paul Crook, Y-Lan Boureau and Jason Weston . . . . . 1951

*CoSQL: A Conversational Text-to-SQL Challenge Towards Cross-Domain Natural Language Interfaces to Databases*
Tao Yu, Rui Zhang, Heyang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, Youxuan Jiang, Michihiro Yasunaga, Sungrok Shim, Tao Chen, Alexander Fabbri, Zifan Li, Luyao Chen, Yuwen Zhang, Shreya Dixit, Vincent Zhang, Caiming Xiong, Richard Socher, Walter Lasecki and Dragomir Radev . . . . . 1962

*A Practical Dialogue-Act-Driven Conversation Model for Multi-Turn Response Selection*
Harshit Kumar, Arvind Agarwal and Sachindra Joshi . . . . . 1980

*How to Build User Simulators to Train RL-based Dialog Systems*
Weiyan Shi, Kun Qian, Xuewei Wang and Zhou Yu . . . . . 1990

*Low-Rank HOCA: Efficient High-Order Cross-Modal Attention for Video Captioning*
Tao Jin, Siyu Huang, Yingming Li and Zhongfei Zhang . . . . . 2001

*Image Captioning with Very Scarce Supervised Data: Adversarial Semi-Supervised Learning Approach*
Dong-Jin Kim, Jinsoo Choi, Tae-Hyun Oh and In So Kweon . . . . . 2012

*Dual Attention Networks for Visual Reference Resolution in Visual Dialog*
Gi-Cheon Kang, Jaeseo Lim and Byoung-Tak Zhang . . . . . 2024

*Unsupervised Discovery of Multimodal Links in Multi-image, Multi-sentence Documents*
Jack Hessel, Lillian Lee and David Mimno . . . . . 2034

*UR-FUNNY: A Multimodal Language Dataset for Understanding Humor*
Md Kamrul Hasan, Wasifur Rahman, AmirAli Bagher Zadeh, Jianyuan Zhong, Md Iftekhar Tanveer, Louis-Philippe Morency and Mohammed (Ehsan) Hoque . . . . . 2046

*Partners in Crime: Multi-view Sequential Inference for Movie Understanding*
Nikos Papasarantopoulos, Lea Frermann, Mirella Lapata and Shay B. Cohen . . . . . 2057

*Guiding the Flowing of Semantics: Interpretable Video Captioning via POS Tag*
Xinyu Xiao, Lingfeng Wang, Bin Fan, Shinming Xiang and Chunhong Pan . . . . . 2068

*A Stack-Propagation Framework with Token-Level Intent Detection for Spoken Language Understanding*
Libo Qin, Wanxiang Che, Yangming Li, Haoyang Wen and Ting Liu . . . . . 2078

*Talk2Car: Taking Control of Your Self-Driving Car*
Thierry Deruyttere, Simon Vandenhende, Dusan Grujicic, Luc Van Gool and Marie-Francine Moens 2088

*Fact-Checking Meets Fauxtography: Verifying Claims About Images*
Dimitrina Zlatkova, Preslav Nakov and Ivan Koychev . . . . . 2099

*Video Dialog via Progressive Inference and Cross-Transformer*
Weike Jin, Zhou Zhao, Mao Gu, Jun Xiao, Furu Wei and Yueting Zhuang . . . . . 2109

*Executing Instructions in Situated Collaborative Interactions*
Alane Suhr, Claudia Yan, Jack Schluger, Stanley Yu, Hadi Khader, Marwa Mouallem, Iris Zhang and Yoav Artzi . . . . . 2119

*Fusion of Detected Objects in Text for Visual Question Answering*
Chris Alberti, Jeffrey Ling, Michael Collins and David Reitter . . . . . 2131

*TIGEr: Text-to-Image Grounding for Image Caption Evaluation*
Ming Jiang, Qiuyuan Huang, Lei Zhang, Xin Wang, Pengchuan Zhang, Zhe Gan, Jana Diesner and Jianfeng Gao . . . . . 2141

*Universal Adversarial Triggers for Attacking and Analyzing NLP*
Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner and Sameer Singh . . . . . 2153

*To Annotate or Not? Predicting Performance Drop under Domain Shift*
Hady Elsahar and Matthias Gallé . . . . . 2163

*Adaptively Sparse Transformers*
Gonçalo M. Correia, Vlad Niculae and André F. T. Martins . . . . . 2174

*Show Your Work: Improved Reporting of Experimental Results*
Jesse Dodge, Suchin Gururangan, Dallas Card, Roy Schwartz and Noah A. Smith . . . . . 2185

*A Deep Factorization of Style and Structure in Fonts*
Akshay Srivatsan, Jonathan Barron, Dan Klein and Taylor Berg-Kirkpatrick . . . . . 2195

*Cross-lingual Semantic Specialization via Lexical Relation Induction*
Edoardo Maria Ponti, Ivan Vulić, Goran Glavaš, Roi Reichart and Anna Korhonen . . . . . 2206

*Modelling the interplay of metaphor and emotion through multitask learning*
Verna Dankers, Marek Rei, Martha Lewis and Ekaterina Shutova . . . . . 2218

*How well do NLI models capture verb veridicality?*
Alexis Ross and Ellie Pavlick . . . . . 2230

*Modeling Color Terminology Across Thousands of Languages*
Arya D. McCarthy, Winston Wu, Aaron Mueller, William Watson and David Yarowsky . . . . . 2241

*Negative Focus Detection via Contextual Attention Mechanism*
Longxiang Shen, Bowei Zou, Yu Hong, Guodong Zhou, Qiaoming Zhu and AiTi Aw . . . . . 2251

*A Unified Neural Coherence Model*
Han Cheol Moon, Tasnim Mohiuddin, Shafiq Joty and Chi Xu . . . . . 2262

*Topic-Guided Coherence Modeling for Sentence Ordering by Preserving Global and Local Information*
Byungkook Oh, Seungmin Seo, Cheolheon Shin, Eunju Jo and Kyong-Ho Lee . . . . . 2273

*Neural Generative Rhetorical Structure Parsing*
Amandla Mabona, Laura Rimell, Stephen Clark and Andreas Vlachos . . . . . 2284

*Weak Supervision for Learning Discourse Structure*
Sonia Badene, Kate Thompson, Jean-Pierre Lorré and Nicholas Asher . . . . . 2296

*Predicting Discourse Structure using Distant Supervision from Sentiment*
Patrick Huber and Giuseppe Carenini . . . . . 2306

*The Myth of Double-Blind Review Revisited: ACL vs. EMNLP*
Cornelia Caragea, Ana Uban and Liviu P. Dinu . . . . . 2317

*Uncover Sexual Harassment Patterns from Personal Stories by Joint Key Element Extraction and Categorization*
Yingchi Liu, Quanzhi Li, Marika Cifor, Xiaozhong Liu, Qiong Zhang and Luo Si . . . . . 2328

*Identifying Predictive Causal Factors from News Streams*
Ananth Balashankar, Sunandan Chakraborty, Samuel Fraiberger and Lakshminarayanan Subramanian . . . . . 2338

*Training Data Augmentation for Detecting Adverse Drug Reactions in User-Generated Content*
Sepideh Mesbah, Jie Yang, Robert-Jan Sips, Manuel Valle Torre, Christoph Lofi, Alessandro Bozzon and Geert-Jan Houben . . . . . 2349

*Deep Reinforcement Learning-based Text Anonymization against Private-Attribute Inference*
Ahmadreza Mosallanezhad, Ghazaleh Beigi and Huan Liu . . . . . 2360

*Tree-structured Decoding for Solving Math Word Problems*
Qianying Liu, Wenyv Guan, Sujian Li and Daisuke Kawahara . . . . . 2370

*PullNet: Open Domain Question Answering with Iterative Retrieval on Knowledge Bases and Text*
Haitian Sun, Tania Bedrax-Weiss and William Cohen . . . . . 2380

*Cosmos QA: Machine Reading Comprehension with Contextual Commonsense Reasoning*
Lifu Huang, Ronan Le Bras, Chandra Bhagavatula and Yejin Choi . . . . . 2391

*Finding Generalizable Evidence by Learning to Convince Q&A Models*
Ethan Perez, Siddharth Karamcheti, Rob Fergus, Jason Weston, Douwe Kiela and Kyunghyun Cho 2402

*Ranking and Sampling in Open-Domain Question Answering*
Yanfu Xu, Zheng Lin, Yuanxin Liu, Rui Liu, Weiping Wang and Dan Meng . . . . . 2412

*A Non-commutative Bilinear Model for Answering Path Queries in Knowledge Graphs*
Katsuhiko Hayashi and Masashi Shimbo . . . . . 2422

*Generating Questions for Knowledge Bases via Incorporating Diversified Contexts and Answer-Aware Loss*
Cao Liu, Kang Liu, Shizhu He, Zaiqing Nie and Jun Zhao . . . . . 2431

*Multi-Task Learning for Conversational Question Answering over a Large-Scale Knowledge Base*
Tao Shen, Xiubo Geng, Tao QIN, Daya Guo, Duyu Tang, Nan Duan, Guodong Long and Daxin Jiang . . . . . 2442

*BiPaR: A Bilingual Parallel Dataset for Multilingual and Cross-lingual Reading Comprehension on Novels*
Yimin Jing, Deyi Xiong and Zhen Yan . . . . . 2452

*Language Models as Knowledge Bases?*
Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu and Alexander Miller . . . . . 2463

*NumNet: Machine Reading Comprehension with Numerical Reasoning*
Qiu Ran, Yankai Lin, Peng Li, Jie Zhou and Zhiyuan Liu . . . . . 2474

*Unicoder: A Universal Language Encoder by Pre-training with Multiple Cross-lingual Tasks*
Haoyang Huang, Yaobo Liang, Nan Duan, Ming Gong, Linjun Shou, Daxin Jiang and Ming Zhou 2485

*Addressing Semantic Drift in Question Generation for Semi-Supervised Question Answering*
Shiyue Zhang and Mohit Bansal . . . . . 2495

*Adversarial Domain Adaptation for Machine Reading Comprehension*
Huazheng Wang, Zhe Gan, Xiaodong Liu, Jingjing Liu, Jianfeng Gao and Hongning Wang . . 2510

*Incorporating External Knowledge into Machine Reading for Generative Question Answering*
Bin Bi, Chen Wu, Ming Yan, Wei Wang, Jiangnan Xia and Chenliang Li . . . . . 2521

*Answering questions by learning to rank - Learning to rank by answering questions*
George Sebastian Pirtoaca, Traian Rebedea and Stefan Ruseti . . . . . 2531

*Discourse-Aware Semantic Self-Attention for Narrative Reading Comprehension*
Todor Mihaylov and Anette Frank . . . . . 2541

*Revealing the Importance of Semantic Retrieval for Machine Reading at Scale*
Yixin Nie, Songhe Wang and Mohit Bansal . . . . . 2553

*PubMedQA: A Dataset for Biomedical Research Question Answering*
Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen and Xinghua Lu . . . . . 2567

*Quick and (not so) Dirty: Unsupervised Selection of Justification Sentences for Multi-hop Question Answering*
Vikas Yadav, Steven Bethard and Mihai Surdeanu . . . . . 2578

*Answering Complex Open-domain Questions Through Iterative Query Generation*
Peng Qi, Xiaowen Lin, Leo Mehr, Zijian Wang and Christopher D. Manning . . . . . 2590

*NL2pSQL: Generating Pseudo-SQL Queries from Under-Specified Natural Language Questions*
Fuxiang Chen, Seung-won Hwang, Jaegul Choo, Jung-Woo Ha and Sunghun Kim . . . . . 2603

*Leveraging Frequent Query Substructures to Generate Formal Queries for Complex Question Answering*
Jiwei Ding, Wei Hu, Qixin Xu and Yuzhong Qu . . . . . 2614

*Incorporating Graph Attention Mechanism into Knowledge Graph Reasoning Based on Deep Reinforcement Learning*
Heng Wang, Shuangyin Li, Rong Pan and Mingzhi Mao . . . . . 2623

*Learning to Update Knowledge Graphs by Reading News*
Jizhi Tang, Yansong Feng and Dongyan Zhao . . . . . 2632

*DIVINE: A Generative Adversarial Imitation Learning Framework for Knowledge Graph Reasoning*
Ruiping Li and Xiang Cheng . . . . . 2642

*Original Semantics-Oriented Attention and Deep Fusion Network for Sentence Matching*
Mingtong Liu, Yujie Zhang, Jinan Xu and Yufeng Chen . . . . . 2652

*Representation Learning with Ordered Relation Paths for Knowledge Graph Completion*
Yao Zhu, Hongzhi Liu, Zhonghai Wu, Yang Song and Tao Zhang . . . . . 2662

*Collaborative Policy Learning for Open Knowledge Graph Reasoning*
Cong Fu, Tong Chen, Meng Qu, Woojeong Jin and Xiang Ren . . . . . 2672

*Modeling Event Background for If-Then Commonsense Reasoning Using Context-aware Variational Autoencoder*
Li Du, Xiao Ding, Ting Liu and Zhongyang Li . . . . . 2682

*Asynchronous Deep Interaction Network for Natural Language Inference*
Di Liang, Fubao Zhang, Qi Zhang and Xuanjing Huang . . . . . 2692

*Keep Calm and Switch On! Preserving Sentiment and Fluency in Semantic Text Exchange*
Steven Y. Feng, Aaron W. Li and Jesse Hoey . . . . . 2701

*Query-focused Scenario Construction*
Su Wang, Greg Durrett and Katrin Erk . . . . . 2712

*Semi-supervised Entity Alignment via Joint Knowledge Embedding Model and Cross-graph Model*
Chengjiang Li, Yixin Cao, Lei Hou, Jiaxin Shi, Juanzi Li and Tat-Seng Chua . . . . . 2723

*Designing and Interpreting Probes with Control Tasks*
John Hewitt and Percy Liang . . . . . 2733

*Specializing Word Embeddings (for Parsing) by Information Bottleneck*
Xiang Lisa Li and Jason Eisner . . . . . 2744

*Deep Contextualized Word Embeddings in Transition-Based and Graph-Based Dependency Parsing - A Tale of Two Parsers Revisited*
Artur Kulmizev, Miryam de Lhoneux, Johannes Gontrum, Elena Fano and Joakim Nivre . . . . 2755

*Semantic graph parsing with recurrent neural network DAG grammars*
Federico Fancellu, Sorcha Gilroy, Adam Lopez and Mirella Lapata . . . . . 2769

*75 Languages, 1 Model: Parsing Universal Dependencies Universally*
Dan Kondratyuk and Milan Straka . . . . . 2779

*Interactive Language Learning by Question Answering*
Xingdi Yuan, Marc-Alexandre Côté, Jie Fu, Zhouhan Lin, Chris Pal, Yoshua Bengio and Adam Trischler . . . . . 2796

*What's Missing: A Knowledge Gap Guided Approach for Multi-hop Question Answering*
Tushar Khot, Ashish Sabharwal and Peter Clark . . . . . 2814

*KagNet: Knowledge-Aware Graph Networks for Commonsense Reasoning*
Bill Yuchen Lin, Xinyue Chen, Jamin Chen and Xiang Ren . . . . . 2829

*Learning with Limited Data for Multilingual Reading Comprehension*
Kyungjae Lee, Sunghyun Park, Hojae Han, Jinyoung Yeo, Seung-won Hwang and Juho Lee . 2840

*A Discrete Hard EM Approach for Weakly Supervised Question Answering*
Sewon Min, Danqi Chen, Hannaneh Hajishirzi and Luke Zettlemoyer . . . . . 2851

*Is the Red Square Big? MALeViC: Modeling Adjectives Leveraging Visual Contexts*
Sandro Pezzelle and Raquel Fernández . . . . . 2865

*Investigating BERT's Knowledge of Language: Five Analysis Methods with NPIs*
Alex Warstadt, Yu Cao, Ioana Grosu, Wei Peng, Hagen Blix, Yining Nie, Anna Alsop, Shikha Bordia, Haokun Liu, Alicia Parrish, Sheng-Fu Wang, Jason Phang, Anhad Mohananey, Phu Mon Htut, Paloma Jeretic and Samuel R. Bowman . . . . . 2877

*Representation of Constituents in Neural Language Models: Coordination Phrase as a Case Study*
Aixiu AN, Peng Qian, Ethan Wilcox and Roger Levy . . . . . 2888

*Towards Zero-shot Language Modeling*
Edoardo Maria Ponti, Ivan Vulić, Ryan Cotterell, Roi Reichart and Anna Korhonen . . . . . 2900

*What Gets Echoed? Understanding the "Pointers" in Explanations of Persuasive Arguments*
David Atkinson, Kumar Bhargav Srinivasan and Chenhao Tan . . . . . 2911

*Modeling Frames in Argumentation*
Yamen Ajjour, Milad Alshomary, Henning Wachsmuth and Benno Stein . . . . . 2922

*AMPERSAND: Argument Mining for PERSuAsive oNline Discussions*
Tuhin Chakrabarty, Christopher Hidey, Smaranda Muresan, Kathy McKeown and Alyssa Hwang 2933

*Evaluating adversarial attacks against multiple fact verification systems*
James Thorne, Andreas Vlachos, Christos Christodoulopoulos and Arpit Mittal . . . . . 2944

*Nonsense!: Quality Control via Two-Step Reason Selection for Annotating Local Acceptability and Related Attributes in News Editorials*
Wonsuk Yang, seungwon yoon, Ada Carpenter and Jong Park . . . . . 2954

*Evaluating Pronominal Anaphora in Machine Translation: An Evaluation Measure and a Test Suite*
Prathyusha Jwalapuram, Shafiq Joty, Irina Temnikova and Preslav Nakov . . . . . 2964

*A Regularization Approach for Incorporating Event Knowledge and Coreference Relations into Neural Discourse Parsing*
Zeyu Dai and Ruihong Huang . . . . . 2976

*Weakly Supervised Multilingual Causality Extraction from Wikipedia*
Chikara Hashimoto . . . . . 2988

*Attribute-aware Sequence Network for Review Summarization*
Junjie Li, Xuepeng Wang, Dawei Yin and Chengqing Zong . . . . . 3000

*Extractive Summarization of Long Documents by Combining Global and Local Context*
Wen Xiao and Giuseppe Carenini . . . . . 3011

*Enhancing Neural Data-To-Text Generation Models with External Background Knowledge*
Shuang Chen, Jinpeng Wang, Xiaocheng Feng, Feng Jiang, Bing Qin and Chin-Yew Lin . . . . . 3022

*Reading Like HER: Human Reading Inspired Extractive Summarization*
Ling Luo, Xiang Ao, Yan Song, Feiyang Pan, Min Yang and Qing He . . . . . 3033

*Contrastive Attention Mechanism for Abstractive Sentence Summarization*
Xiangyu Duan, Hongfei Yu, Mingming Yin, Min Zhang, Weihua Luo and Yue Zhang . . . . . 3044

*NCLS: Neural Cross-Lingual Summarization*
Junnan Zhu, Qian Wang, Yining Wang, Yu Zhou, Jiajun Zhang, Shaonan Wang and Chengqing Zong . . . . . 3054

*Clickbait? Sensational Headline Generation with Auto-tuned Reinforcement Learning*
Peng Xu, Chien-Sheng Wu, Andrea Madotto and Pascale Fung . . . . . 3065

*Concept Pointer Network for Abstractive Summarization*
Wenbo Wang, Yang Gao, Heyan Huang and Yuxiang Zhou . . . . . 3076

*Surface Realisation Using Full Delexicalisation*
Anastasia Shimorina and Claire Gardent . . . . . 3086

*IMaT: Unsupervised Text Attribute Transfer via Iterative Matching and Translation*
Zhijing Jin, Di Jin, Jonas Mueller, Nicholas Matthews and Enrico Santus . . . . . 3097

*Better Rewards Yield Better Summaries: Learning to Summarise Without References*
Florian Böhm, Yang Gao, Christian M. Meyer, Ori Shapira, Ido Dagan and Iryna Gurevych . . 3110

*Mixture Content Selection for Diverse Sequence Generation*
Jaemin Cho, Minjoon Seo and Hannaneh Hajishirzi . . . . . 3121

*An End-to-End Generative Architecture for Paraphrase Generation*
Qian Yang, Zhouyuan Huo, Dinghan Shen, Yong Cheng, Wenlin Wang, Guoyin Wang and Lawrence Carin . . . . . 3132

*Table-to-Text Generation with Effective Hierarchical Encoder on Three Dimensions (Row, Column and Time)*
Heng Gong, Xiaocheng Feng, Bing Qin and Ting Liu . . . . . 3143

*Subtopic-driven Multi-Document Summarization*
Xin Zheng, Aixin Sun, Jing Li and Karthik Muthuswamy . . . . . 3153

*Referring Expression Generation Using Entity Profiles*
Meng Cao and Jackie Chi Kit Cheung . . . . . 3163

*Exploring Diverse Expressions for Paraphrase Generation*
Lihua Qian, Lin Qiu, Weinan Zhang, Xin Jiang and Yong Yu . . . . . 3173

*Enhancing AMR-to-Text Generation with Dual Graph Representations*
Leonardo F. R. Ribeiro, Claire Gardent and Iryna Gurevych . . . . . 3183

*Keeping Consistency of Sentence Generation and Document Classification with Multi-Task Learning*
Toru Nishino, Shotaro Misawa, Ryuji Kano, Tomoki Taniguchi, Yasuhide Miura and Tomoko Ohkuma . . . . . 3195

*Toward a Task of Feedback Comment Generation for Writing Learning*
Ryo Nagata . . . . . 3206

*Improving Question Generation With to the Point Context*
Jingjing Li, Yifan Gao, Lidong Bing, Irwin King and Michael R. Lyu . . . . . 3216

*Deep Copycat Networks for Text-to-Text Generation*
Julia Ive, Pranava Madhyastha and Lucia Specia . . . . . 3227

*Towards Controllable and Personalized Review Generation*
Pan Li and Alexander Tuzhilin . . . . . 3237

*Answers Unite! Unsupervised Metrics for Reinforced Summarization Models*
Thomas Scialom, Sylvain Lamprier, Benjamin Piwowarski and Jacopo Staiano . . . . . 3246

*Long and Diverse Text Generation with Planning-based Hierarchical Variational Model*
Zhihong Shao, Minlie Huang, Jiangtao Wen, Wenfei Xu and xiaoyan zhu . . . . . 3257

*"Transforming" Delete, Retrieve, Generate Approach for Controlled Text Style Transfer*
Akhilesh Sudhakar, Bhargav Upadhyay and Arjun Maheswaran . . . . . 3269

*An Entity-Driven Framework for Abstractive Summarization*
Eva Sharma, Luyang Huang, Zhe Hu and Lu Wang . . . . . 3280

*Neural Extractive Text Summarization with Syntactic Compression*
Jiacheng Xu and Greg Durrett . . . . . 3292

*Domain Adaptive Text Style Transfer*
Dianqi Li, Yizhe Zhang, Zhe Gan, Yu Cheng, Chris Brockett, Bill Dolan and Ming-Ting Sun 3304

*Let's Ask Again: Refine Network for Automatic Question Generation*
Preksha Nema, Akash Kumar Mohankumar, Mitesh M. Khapra, Balaji Vasan Srinivasan and Balaraman Ravindran . . . . . 3314

*Earlier Isn't Always Better: Sub-aspect Analysis on Corpus and System Biases in Summarization*
Taehee Jung, Dongyeop Kang, Lucas Mentch and Eduard Hovy . . . . . 3324

*Lost in Evaluation: Misleading Benchmarks for Bilingual Dictionary Induction*
Yova Kementchedjhieva, Mareike Hartmann and Anders Søgaard . . . . . 3336

*Towards Realistic Practices In Low-Resource Natural Language Processing: The Development Set*
Katharina Kann, Kyunghyun Cho and Samuel R. Bowman . . . . . 3342

*Synchronously Generating Two Languages with Interactive Decoding*
Yining Wang, Jiajun Zhang, Long Zhou, Yuchen Liu and Chengqing Zong . . . . . 3350

*On NMT Search Errors and Model Errors: Cat Got Your Tongue?*
Felix Stahlberg and Bill Byrne . . . . . 3356

*"Going on a vacation" takes longer than "Going for a walk": A Study of Temporal Commonsense Understanding*
Ben Zhou, Daniel Khashabi, Qiang Ning and Dan Roth . . . . . 3363

*QAInfomax: Learning Robust Question Answering System by Mutual Information Maximization*
Yi-Ting Yeh and Yun-Nung Chen . . . . . 3370

*Adapting Meta Knowledge Graph Information for Multi-Hop Reasoning over Few-Shot Relations*
Xin Lv, Yuxian Gu, Xu Han, Lei Hou, Juanzi Li and Zhiyuan Liu . . . . . 3376

*How Reasonable are Common-Sense Reasoning Tasks: A Case-Study on the Winograd Schema Challenge and SWAG*
Paul Trichelair, Ali Emami, Adam Trischler, Kaheer Suleman and Jackie Chi Kit Cheung . . . . 3382

*Pun-GAN: Generative Adversarial Network for Pun Generation*
Fuli Luo, Shunyao Li, Pengcheng Yang, Lei Li, Baobao Chang, Zhifang Sui and Xu SUN . . . 3388

*Multi-Task Learning with Language Modeling for Question Generation*
Wenjie Zhou, Minghua Zhang and Yunfang Wu . . . . . 3394

*Autoregressive Text Generation Beyond Feedback Loops*
Florian Schmidt, Stephan Mandt and Thomas Hofmann . . . . . 3400

*The Woman Worked as a Babysitter: On Biases in Language Generation*
Emily Sheng, Kai-Wei Chang, Premkumar Natarajan and Nanyun Peng . . . . . 3407

*On the Importance of Delexicalization for Fact Verification*
Sandeep Suntwal, Mithun Paul, Rebecca Sharp and Mihai Surdeanu . . . . . 3413

*Towards Debiasing Fact Verification Models*
Tal Schuster, Darsh Shah, Yun Jie Serene Yeo, Daniel Roberto Filizzola Ortiz, Enrico Santus and Regina Barzilay . . . . . 3419

*Recognizing Conflict Opinions in Aspect-level Sentiment Classification with Dual Attention Networks*
Xingwei Tan, Yi Cai and Changxi Zhu . . . . . 3426

*Investigating Dynamic Routing in Tree-Structured LSTM for Sentiment Analysis*
Jin Wang, Liang-Chih Yu, K. Robert Lai and Xuejie Zhang . . . . . 3432

*A Label Informative Wide & Deep Classifier for Patents and Papers*
Muyao Niu and Jie Cai . . . . . 3438

*Text Level Graph Neural Network for Text Classification*
Lianzhe Huang, Dehong Ma, Sujian Li, Xiaodong Zhang and Houfeng WANG . . . . . 3444

*Semantic Relatedness Based Re-ranker for Text Spotting*
Ahmed Sabir, Francesc Moreno and Lluís Padró . . . . . 3451

*Delta-training: Simple Semi-Supervised Text Classification using Pretrained Word Embeddings*
Hwiyeol Jo and Ceyda Cinarel . . . . . 3458

*Visual Detection with Context for Document Layout Analysis*
Carlos Soto and Shinjae Yoo . . . . . 3464

*Evaluating Topic Quality with Posterior Variability*
Linzi Xing, Michael J. Paul and Giuseppe Carenini . . . . . 3471

*Neural Topic Model with Reinforcement Learning*
Lin Gui, Jia Leng, Gabriele Pergola, yu zhou, Ruifeng Xu and Yulan He . . . . . 3478

*Modelling Stopping Criteria for Search Results using Poisson Processes*
Alison Sneyd and Mark Stevenson . . . . . 3484

*Cross-Domain Modeling of Sentence-Level Evidence for Document Retrieval*
Zeynep Akkalyoncu Yilmaz, Wei Yang, Haotian Zhang and Jimmy Lin . . . . . 3490

*The Challenges of Optimizing Machine Translation for Low Resource Cross-Language Information Retrieval*
Constantine Lignos, Daniel Cohen, Yen-Chieh Lien, Pratik Mehta, W. Bruce Croft and Scott Miller 3497

*Rotate King to get Queen: Word Relationships as Orthogonal Transformations in Embedding Space*
Kawin Ethayarajh . . . . . 3503

*GlossBERT: BERT for Word Sense Disambiguation with Gloss Knowledge*
Luyao Huang, Chi Sun, Xipeng Qiu and Xuanjing Huang . . . . . 3509

*Leveraging Adjective-Noun Phrasing Knowledge for Comparison Relation Prediction in Text-to-SQL*
Haoyan Liu, Lei Fang, Qian Liu, Bei Chen, Jian-Guang LOU and Zhoujun Li . . . . . 3515

*Bridging the Defined and the Defining: Exploiting Implicit Lexical Semantic Relations in Definition Modeling*
Koki Washio, Satoshi Sekine and Tsuneaki Kato . . . . . 3521

*Don't Just Scratch the Surface: Enhancing Word Representations for Korean with Hanja*
Kang Min Yoo, Taeuk Kim and Sang-goo Lee . . . . . 3528

*SyntagNet: Challenging Supervised Word Sense Disambiguation with Lexical-Semantic Combinations*
Marco Maru, Federico Scozzafava, Federico Martelli and Roberto Navigli . . . . . 3534

*Hierarchical Meta-Embeddings for Code-Switching Named Entity Recognition*
Genta Indra Winata, Zhaojiang Lin, Jamin Shin, Zihan Liu and Pascale Fung . . . . . 3541

*Fine-tune BERT with Sparse Self-Attention Mechanism*
Baiyun Cui, Yingming Li, Ming Chen and Zhongfei Zhang . . . . . 3548

*Feature-Dependent Confusion Matrices for Low-Resource NER Labeling with Noisy Labels*
Lukas Lange, Michael A. Hedderich and Dietrich Klakow . . . . . 3554

*A Multi-Pairwise Extension of Procrustes Analysis for Multilingual Word Translation*
Hagai Taitelbaum, Gal Chechik and Jacob Goldberger . . . . . 3560

*Out-of-Domain Detection for Low-Resource Text Classification Tasks*
Ming Tan, Yang Yu, Haoyu Wang, Dakuo Wang, Saloni Potdar, Shiyu Chang and Mo Yu . . . . 3566

*Harnessing Pre-Trained Neural Networks with Rules for Formality Style Transfer*
Yunli Wang, Yu Wu, Lili Mou, Zhoujun Li and Wenhan Chao . . . . . 3573

*Multiple Text Style Transfer by using Word-level Conditional Generative Adversarial Network with Two-Phase Training*
Chih-Te Lai, Yi-Te Hong, Hong-You Chen, Chi-Jen Lu and Shou-De Lin . . . . . 3579

*Improved Differentiable Architecture Search for Language Modeling and Named Entity Recognition*
Yufan Jiang, Chi Hu, Tong Xiao, Chunliang Zhang and Jingbo Zhu . . . . . 3585

*Using Pairwise Occurrence Information to Improve Knowledge Graph Completion on Large-Scale Datasets*
Esma Balkir, Masha Naslidnyk, Dave Palfrey and Arpit Mittal . . . . . 3591

*Single Training Dimension Selection for Word Embedding with PCA*
Yu Wang . . . . . 3597

*A Surprisingly Effective Fix for Deep Latent Variable Modeling of Text*
Bohan Li, Junxian He, Graham Neubig, Taylor Berg-Kirkpatrick and Yiming Yang . . . . . 3603

*SciBERT: A Pretrained Language Model for Scientific Text*
Iz Beltagy, Kyle Lo and Arman Cohan . . . . . 3615

*Humor Detection: A Transformer Gets the Last Laugh*
Orion Weller and Kevin Seppi . . . . . 3621

*Combining Global Sparse Gradients with Local Gradients in Distributed Neural Network Training*
Alham Fikri Aji, Kenneth Heafield and Nikolay Bogoychev . . . . . 3626

*Small and Practical BERT Models for Sequence Labeling*
Henry Tsai, Jason Riesa, Melvin Johnson, Naveen Arivazhagan, Xin Li and Amelia Archer . . 3632

*Data Augmentation with Atomic Templates for Spoken Language Understanding*
Zijian Zhao, Su Zhu and Kai Yu . . . . . 3637

*PaLM: A Hybrid Parser and Language Model*
Hao Peng, Roy Schwartz and Noah A. Smith . . . . . 3644

*A Pilot Study for Chinese SQL Semantic Parsing*
Qingkai Min, Yuefeng Shi and Yue Zhang . . . . . 3652

*Global Reasoning over Database Structures for Text-to-SQL Parsing*
Ben Bogin, Matt Gardner and Jonathan Berant . . . . . 3659

*Transductive Learning of Neural Language Models for Syntactic and Semantic Analysis*
Hiroki Ouchi, Jun Suzuki and Kentaro Inui . . . . . 3665

*Efficient Sentence Embedding using Discrete Cosine Transform*
Nada Almarwani, Hanan Aldarmaki and Mona Diab . . . . . 3672

*A Search-based Neural Model for Biomedical Nested and Overlapping Event Detection*
Kurt Junshean Espinosa, Makoto Miwa and Sophia Ananiadou . . . . . 3679

*PAWS-X: A Cross-lingual Adversarial Dataset for Paraphrase Identification*
Yinfei Yang, Yuan Zhang, Chris Tar and Jason Baldridge . . . . . 3687

*Pretrained Language Models for Sequential Sentence Classification*
Arman Cohan, Iz Beltagy, Daniel King, Bhavana Dalvi and Dan Weld . . . . . 3693

*Emergent Linguistic Phenomena in Multi-Agent Communication Games*
Laura Harding Graesser, Kyunghyun Cho and Douwe Kiela . . . . . 3700

*TalkDown: A Corpus for Condescension Detection in Context*
Zijian Wang and Christopher Potts . . . . . 3711

*Summary Cloze: A New Task for Content Selection in Topic-Focused Summarization*
Daniel Deutsch and Dan Roth . . . . . 3720

*Text Summarization with Pretrained Encoders*
Yang Liu and Mirella Lapata . . . . . 3730

*How to Write Summaries with Patterns? Learning towards Abstractive Summarization through Prototype Editing*
Shen Gao, Xiuying Chen, Piji Li, Zhangming Chan, Dongyan Zhao and Rui Yan . . . . . 3741

*BottleSum: Unsupervised and Self-supervised Sentence Summarization using the Information Bottleneck Principle*
Peter West, Ari Holtzman, Jan Buys and Yejin Choi . . . . . 3752

*Improving Latent Alignment in Text Summarization by Generalizing the Pointer Generator*
Xiaoyu Shen, Yang Zhao, Hui Su and Dietrich Klakow . . . . . 3762

*Learning Semantic Parsers from Denotations with Latent Structured Alignments and Abstract Programs*
Bailin Wang, Ivan Titov and Mirella Lapata . . . . . 3774

*Broad-Coverage Semantic Parsing as Transduction*
Sheng Zhang, Xutai Ma, Kevin Duh and Benjamin Van Durme . . . . . 3786

*Core Semantic First: A Top-down Approach for AMR Parsing*
Deng Cai and Wai Lam . . . . . 3799

*Don't paraphrase, detect! Rapid and Effective Data Collection for Semantic Parsing*
Jonathan Herzig and Jonathan Berant . . . . . 3810

*Improving Distantly-Supervised Relation Extraction with Joint Label Embedding*
Linmei Hu, Luhao Zhang, Chuan Shi, Liqiang Nie, Weili Guan and Cheng Yang . . . . . 3821

*Leverage Lexical Knowledge for Chinese Named Entity Recognition via Collaborative Graph Network*
Dianbo Sui, Yubo Chen, Kang Liu, Jun Zhao and Shengping Liu . . . . . 3830

*Looking Beyond Label Noise: Shifted Label Distribution Matters in Distantly Supervised Relation Extraction*
Qinyuan Ye, Liyuan Liu, Maosen Zhang and Xiang Ren . . . . . 3841

*Easy First Relation Extraction with Information Redundancy*
Shuai Ma, Gang Wang, Yansong Feng and Jinpeng Huai . . . . . 3851

*Dependency-Guided LSTM-CRF for Named Entity Recognition*
Zhanming Jie and Wei Lu . . . . . 3862

*Cross-Cultural Transfer Learning for Text Classification*
Dor Ringel, Gal Lavee, Ido Guy and Kira Radinsky . . . . . 3873

*Combining Unsupervised Pre-training and Annotator Rationales to Improve Low-shot Text Classification*
Oren Melamud, Mihaela Bornea and Ken Barker . . . . . 3884

*ProSeqo: Projection Sequence Networks for On-Device Text Classification*
Zornitsa Kozareva and Sujith Ravi . . . . . 3894

*Induction Networks for Few-Shot Text Classification*
Ruiying Geng, Binhua Li, Yongbin Li, Xiaodan Zhu, Ping Jian and Jian Sun . . . . . 3904

*Benchmarking Zero-shot Text Classification: Datasets, Evaluation and Entailment Approach*
Wenpeng Yin, Jamaal Hay and Dan Roth . . . . . 3914

*A Logic-Driven Framework for Consistency of Neural Models*
Tao Li, Vivek Gupta, Maitrey Mehta and Vivek Srikumar . . . . . 3924

*Style Transfer for Texts: Retrain, Report Errors, Compare with Rewrites*
Alexey Tikhonov, Viacheslav Shibaev, Aleksander Nagaev, Aigul Nugmanova and Ivan P. Yamshchikov 3936

*Implicit Deep Latent Variable Models for Text Generation*
Le Fang, Chunyuan Li, Jianfeng Gao, Wen Dong and Changyou Chen . . . . . 3946

*Text Emotion Distribution Learning from Small Sample: A Meta-Learning Approach*
Zhenjie Zhao and Xiaojuan Ma . . . . . 3957

*Judge the Judges: A Large-Scale Evaluation Study of Neural Language Models for Online Review Generation*
Cristina Garbacea, Samuel Carton, Shiyan Yan and Qiaozhu Mei . . . . . 3968

*Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*
Nils Reimers and Iryna Gurevych . . . . . 3982

*Learning Only from Relevant Keywords and Unlabeled Documents*
Nontawat Charoenphakdee, Jongyeong Lee, Yiping Jin, Dittaya Wanvarie and Masashi Sugiyama 3993

*Denoising based Sequence-to-Sequence Pre-training for Text Generation*
Liang Wang, Wei Zhao, Ruoyu Jia, Sujian Li and Jingming Liu . . . . . 4003

*Dialog Intent Induction with Deep Multi-View Clustering*
Hugh Perkins and Yi Yang . . . . . 4016

*Nearly-Unsupervised Hashcode Representations for Biomedical Relation Extraction*
Sahil Garg, Aram Galstyan, Greg Ver Steeg and Guillermo Cecchi . . . . . 4026

*Auditing Deep Learning processes through Kernel-based Explanatory Models*
Danilo Croce, Daniele Rossini and Roberto Basili . . . . . 4037

*Enhancing Variational Autoencoders with Mutual Information Neural Estimation for Text Generation*
Dong Qian and William K. Cheung . . . . . 4047

*Sampling Bias in Deep Active Classification: An Empirical Study*
Ameya Prabhu, Charles Dognin and Maneesh Singh . . . . . 4058

*Don't Take the Easy Way Out: Ensemble Based Methods for Avoiding Known Dataset Biases*
Christopher Clark, Mark Yatskar and Luke Zettlemoyer . . . . . 4069

*Achieving Verified Robustness to Symbol Substitutions via Interval Bound Propagation*
Po-Sen Huang, Robert Stanforth, Johannes Welbl, Chris Dyer, Dani Yogatama, Sven Gowal, Krishnamurthy Dvijotham and Pushmeet Kohli . . . . . 4083

*Rethinking Cooperative Rationalization: Introspective Extraction and Complement Control*
Mo Yu, Shiyu Chang, Yang Zhang and Tommi Jaakkola . . . . . 4094

*Experimenting with Power Divergences for Language Modeling*
Matthieu Labeau and Shay B. Cohen . . . . . 4104

*Hierarchically-Refined Label Attention Network for Sequence Labeling*
Leyang Cui and Yue Zhang . . . . . 4115

*Certified Robustness to Adversarial Word Substitutions*
Robin Jia, Aditi Raghunathan, Kerem Göksel and Percy Liang . . . . . 4129

*Visualizing and Understanding the Effectiveness of BERT*
Yaru Hao, Li Dong, Furu Wei and Ke Xu . . . . . 4143

*Topics to Avoid: Demoting Latent Confounds in Text Classification*
Sachin Kumar, Shuly Wintner, Noah A. Smith and Yulia Tsvetkov . . . . . 4153

*Learning to Ask for Conversational Machine Learning*
Shashank Srivastava, Igor Labutov and Tom Mitchell . . . . . 4164

*Language Modeling for Code-Switching: Evaluation, Integration of Monolingual Data, and Discriminative Training*
Hila Gonen and Yoav Goldberg . . . . . 4175

*Using Local Knowledge Graph Construction to Scale Seq2Seq Models to Multi-Document Inputs*
Angela Fan, Claire Gardent, Chloé Braud and Antoine Bordes . . . . . 4186

*Fine-grained Knowledge Fusion for Sequence Labeling Domain Adaptation*
Huiyun Yang, Shujian Huang, XIN-YU DAI and Jiajun CHEN . . . . . 4197

*Exploiting Monolingual Data at Scale for Neural Machine Translation*
Lijun Wu, Yiren Wang, Yingce Xia, Tao QIN, Jianhuang Lai and Tie-Yan Liu . . . . . 4207

*Meta Relational Learning for Few-Shot Link Prediction in Knowledge Graphs*
Mingyang Chen, Wen Zhang, Wei Zhang, Qiang Chen and Huajun Chen . . . . . 4217

*Distributionally Robust Language Modeling*
Yonatan Oren, Shiori Sagawa, Tatsunori Hashimoto and Percy Liang . . . . . 4227

*Unsupervised Domain Adaptation of Contextualized Embeddings for Sequence Labeling*
Xiaochuang Han and Jacob Eisenstein . . . . . 4238

*Learning Latent Parameters without Human Response Patterns: Item Response Theory with Artificial Crowds*
John P. Lalor, Hao Wu and Hong Yu . . . . . 4249

*Parallel Iterative Edit Models for Local Sequence Transduction*
Abhijeet Awasthi, Sunita Sarawagi, Rasna Goyal, Sabyasachi Ghosh and Vihari Piratla . . . . . 4260

*ARAML: A Stable Adversarial Training Framework for Text Generation*
Pei Ke, Fei Huang, Minlie Huang and xiaoyan zhu . . . . . 4271

*FlowSeq: Non-Autoregressive Conditional Sequence Generation with Generative Flow*
Xuezhe Ma, Chunting Zhou, Xian Li, Graham Neubig and Eduard Hovy . . . . . 4282

*Compositional Generalization for Primitive Substitutions*
Yuanpeng Li, Liang Zhao, Jianyu Wang and Joel Hestness . . . . . 4293

*WikiCREM: A Large Unsupervised Corpus for Coreference Resolution*
Vid Kocijan, Oana-Maria Camburu, Ana-Maria Cretu, Yordan Yordanov, Phil Blunsom and Thomas Lukasiewicz . . . . . 4303

*Identifying and Explaining Discriminative Attributes*
Armins Stepanjans and André Freitas . . . . . 4313

*Patient Knowledge Distillation for BERT Model Compression*
Siqi Sun, Yu Cheng, Zhe Gan and Jingjing Liu . . . . . 4323

*Neural Gaussian Copula for Variational Autoencoder*
Prince Zizhuang Wang and William Yang Wang . . . . . 4333

*Transformer Dissection: An Unified Understanding for Transformer's Attention via the Lens of Kernel*
Yao-Hung Hubert Tsai, Shaojie Bai, Makoto Yamada, Louis-Philippe Morency and Ruslan Salakhutdinov . . . . . 4344

*Learning to Learn and Predict: A Meta-Learning Approach for Multi-Label Classification*
Jiawei Wu, Wenhan Xiong and William Yang Wang . . . . . 4354

*Revealing the Dark Secrets of BERT*
Olga Kovaleva, Alexey Romanov, Anna Rogers and Anna Rumshisky . . . . . 4365

*Machine Translation With Weakly Paired Documents*
Lijun Wu, Jinhua Zhu, Di He, Fei Gao, Tao QIN, Jianhuang Lai and Tie-Yan Liu . . . . . 4375

*Countering Language Drift via Visual Grounding*
Jason Lee, Kyunghyun Cho and Douwe Kiela . . . . . 4385

*The Bottom-up Evolution of Representations in the Transformer: A Study with Machine Translation and Language Modeling Objectives*
Elena Voita, Rico Sennrich and Ivan Titov . . . . . 4396

*Do We Really Need Fully Unsupervised Cross-Lingual Embeddings?*
Ivan Vulić, Goran Glavaš, Roi Reichart and Anna Korhonen . . . . . 4407

*Weakly-Supervised Concept-based Adversarial Learning for Cross-lingual Word Embeddings*
Haozhou Wang, James Henderson and Paola Merlo . . . . . 4419

*Aligning Cross-Lingual Entities with Multi-Aspect Information*
Hsiu-Wei Yang, Yanyan Zou, Peng Shi, Wei Lu, Jimmy Lin and Xu SUN . . . . . 4431

*Contrastive Language Adaptation for Cross-Lingual Stance Detection*
Mitra Mohtarami, James Glass and Preslav Nakov . . . . . 4442

*Jointly Learning to Align and Translate with Transformer Models*
Sarthak Garg, Stephan Peitz, Udhyakumar Nallasamy and Matthias Paulik . . . . . 4453

*Social IQa: Commonsense Reasoning about Social Interactions*
Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras and Yejin Choi . . . . . 4463

*Self-Assembling Modular Networks for Interpretable Multi-Hop Reasoning*
Yichen Jiang and Mohit Bansal . . . . . 4474

*Posing Fair Generalization Tasks for Natural Language Inference*
Atticus Geiger, Ignacio Cases, Lauri Karttunen and Christopher Potts . . . . . 4485

*Everything Happens for a Reason: Discovering the Purpose of Actions in Procedural Text*
Bhavana Dalvi, Niket Tandon, Antoine Bosselut, Wen-tau Yih and Peter Clark . . . . . 4496

*CLUTRR: A Diagnostic Benchmark for Inductive Reasoning from Text*
Koustuv Sinha, Shagun Sodhani, Jin Dong, Joelle Pineau and William L. Hamilton . . . . . 4506

*Taskmaster-1: Toward a Realistic and Diverse Dialog Dataset*
Bill Byrne, Karthik Krishnamoorthi, Chinnadhurai Sankar, Arvind Neelakantan, Ben Goodrich, Daniel Duckworth, Semih Yavuz, Amit Dubey, Kyu-Young Kim and Andy Cedilnik . . . . . 4516

*Multi-Domain Goal-Oriented Dialogues (MultiDoGO): Strategies toward Curating and Annotating Large Scale Dialogue Data*
Denis Peskov, Nancy Clarke, Jason Krone, Brigi Fodor, Yi Zhang, Adel Youssef and Mona Diab 4526

*Build it Break it Fix it for Dialogue Safety: Robustness from Adversarial Human Attack*
Emily Dinan, Samuel Humeau, Bharath Chintagunta and Jason Weston . . . . . 4537

*GECOR: An End-to-End Generative Ellipsis and Co-reference Resolution Model for Task-Oriented Dialogue*
Jun Quan, Deyi Xiong, Bonnie Webber and Changjian Hu . . . . . 4547

*Task-Oriented Conversation Generation Using Heterogeneous Memory Networks*
Zehao Lin, Xinjing Huang, Feng Ji, Haiqing Chen and Yin Zhang . . . . . 4558

*Aspect-based Sentiment Classification with Aspect-specific Graph Convolutional Networks*
Chen Zhang, Qiuchi Li and Dawei Song . . . . . 4568

*Coupling Global and Local Context for Unsupervised Aspect Extraction*
Ming Liao, Jing Li, Haisong Zhang, Lingzhi Wang, Xixin Wu and Kam-Fai Wong . . . . . 4579

*Transferable End-to-End Aspect-based Sentiment Analysis with Selective Adversarial Learning*
Zheng Li, Xin Li, Ying Wei, Lidong Bing, Yu Zhang and Qiang Yang . . . . . 4590

*CAN: Constrained Attention Networks for Multi-Aspect Sentiment Analysis*
Mengting Hu, Shiwan Zhao, Li Zhang, Keke Cai, Zhong Su, Renhong Cheng and Xiaowei Shen 4601

*Leveraging Just a Few Keywords for Fine-Grained Aspect Detection Through Weakly Supervised Co-Training*
Giannis Karamanolakis, Daniel Hsu and Luis Gravano . . . . . 4611

*Integrating Text and Image: Determining Multimodal Document Intent in Instagram Posts*
Julia Kruk, Jonah Lubin, Karan Sikka, Xiao Lin, Dan Jurafsky and Ajay Divakaran . . . . . 4622

*Neural Conversation Recommendation with Online Interaction Modeling*
Xingshan Zeng, Jing Li, Lu Wang and Kam-Fai Wong . . . . . 4633

*Different Absorption from the Same Sharing: Sifted Multi-task Learning for Fake News Detection*
Lianwei Wu, Yuan Rao, Haolin Jin, Ambreen Nazir and Ling Sun . . . . . 4644

*Text-based inference of moral sentiment change*
Jing Yi Xie, Renato Ferreira Pinto Junior, Graeme Hirst and Yang Xu . . . . . 4654

*Detecting Causal Language Use in Science Findings*
Bei Yu, Yingya Li and Jun Wang . . . . . 4664

*Multilingual and Multi-Aspect Hate Speech Analysis*
Nedjma Ousidhoum, Zizheng Lin, Hongming Zhang, Yangqiu Song and Dit-Yan Yeung . . . . . 4675

*MultiFC: A Real-World Multi-Domain Dataset for Evidence-Based Fact Checking of Claims*
Isabelle Augenstein, Christina Lioma, Dongsheng Wang, Lucas Chaves Lima, Casper Hansen, Christian Hansen and Jakob Grue Simonsen . . . . . 4685

*A Deep Neural Information Fusion Architecture for Textual Network Embeddings*
Zenan Xu, Qinliang Su, Xiaojun Quan and Weijia Zhang . . . . . 4698

*You Shall Know a User by the Company It Keeps: Dynamic Representations for Social Media Users in NLP*
Marco Del Tredici, Diego Marcheggiani, Sabine Schulte im Walde and Raquel Fernández . . . 4707

*Adaptive Ensembling: Unsupervised Domain Adaptation for Political Document Analysis*
Shrey Desai, Barea Sinno, Alex Rosenfeld and Junyi Jessy Li . . . . . 4718

*Macrocosm: Social Media Persona Linking for Open Source Intelligence Applications*
Graham Horwood, Ning Yu, Thomas Boggs, Changjiang Yang and Chad Holvenstot . . . . . 4731

*A Hierarchical Location Prediction Neural Network for Twitter User Geolocation*
Binxuan Huang and Kathleen Carley . . . . . 4732

*Trouble on the Horizon: Forecasting the Derailment of Online Conversations as they Develop*
Jonathan P. Chang and Cristian Danescu-Niculescu-Mizil . . . . . 4743

*A Benchmark Dataset for Learning to Intervene in Online Hate Speech*
Jing Qian, Anna Bethke, Yinyin Liu, Elizabeth Belding and William Yang Wang . . . . . 4755

*Detecting and Reducing Bias in a High Stakes Domain*
Ruiqi Zhong, Yanda Chen, Desmond Patton, Charlotte Selous and Kathy McKeown . . . . . 4765

*CodeSwitch-Reddit: Exploration of Written Multilingual Discourse in Online Discussion Forums*
Ella Rabinovich, Masih Sultani and Suzanne Stevenson . . . . . 4776

*Modeling Conversation Structure and Temporal Dynamics for Jointly Predicting Rumor Stance and Veracity*
Penghui Wei, Nan Xu and Wenji Mao . . . . . 4787

*Reconstructing Capsule Networks for Zero-shot Intent Classification*
Han Liu, Xiaotong Zhang, Lu Fan, Xuandi Fu, Qimai Li, Xiao-Ming Wu and Albert Y.S. Lam 4799

*Domain Adaptation for Person-Job Fit with Transferable Deep Global Match Network*
Shuqing Bian, Wayne Xin Zhao, Yang Song, Tao Zhang and Ji-Rong Wen . . . . . 4810

*Heterogeneous Graph Attention Networks for Semi-supervised Short Text Classification*
Hu Linmei, Tianchi Yang, Chuan Shi, Houye Ji and Xiaoli Li . . . . . 4821

*Comparing and Developing Tools to Measure the Readability of Domain-Specific Texts*
Elissa Redmiles, Lisa Maszkiewicz, Emily Hwang, Dhruv Kuchhal, Everest Liu, Miraida Morales, Denis Peskov, Sudha Rao, Rock Stevens, Kristina Gligorić, Sean Kross, Michelle Mazurek and Hal Daumé III . . . . . 4831

*News2vec: News Network Embedding with Subnode Information*
Ye Ma, Lu Zong, Yikang Yang and Jionglong Su . . . . . 4843

*Recursive Context-Aware Lexical Simplification*
Sian Gooding and Ekaterina Kochmar . . . . . 4853

*Leveraging Medical Literature for Section Prediction in Electronic Health Records*
Sara Rosenthal, Ken Barker and Zhicheng Liang . . . . . 4864

*Neural News Recommendation with Heterogeneous User Behavior*
Chuhan Wu, Fangzhao Wu, Mingxiao An, Tao Qi, Jianqiang Huang, Yongfeng Huang and Xing Xie . . . . . 4874

*Reviews Meet Graphs: Enhancing User and Item Representations for Recommendation with Hierarchical Attentive Graph Neural Network*
Chuhan Wu, Fangzhao Wu, Tao Qi, Suyu Ge, Yongfeng Huang and Xing Xie . . . . . 4884

*Event Representation Learning Enhanced with External Commonsense Knowledge*
Xiao Ding, Kuo Liao, Ting Liu, Zhongyang Li and Junwen Duan . . . . . 4894

*Learning to Discriminate Perturbations for Blocking Adversarial Attacks in Text Classification*
Yichao Zhou, Jyun-Yu Jiang, Kai-Wei Chang and Wei Wang . . . . . 4904

*A Neural Citation Count Prediction Model based on Peer Review Text*
Siqing Li, Wayne Xin Zhao, Eddy Jing Yin and Ji-Rong Wen . . . . . 4914

*Connecting the Dots: Document-level Neural Relation Extraction with Edge-oriented Graphs*
Fenia Christopoulou, Makoto Miwa and Sophia Ananiadou . . . . . 4925

*Semi-supervised Text Style Transfer: Cross Projection in Latent Space*
Mingyue Shang, Piji Li, Zhenxin Fu, Lidong Bing, Dongyan Zhao, Shuming Shi and Rui Yan 4937

*Question Answering for Privacy Policies: Combining Computational and Legal Perspectives*
Abhilasha Ravichander, Alan W Black, Shomir Wilson, Thomas Norton and Norman Sadeh . 4947

*Stick to the Facts: Learning towards a Fidelity-oriented E-Commerce Product Description Generation*
Zhangming Chan, Xiuying Chen, Yongliang Wang, Juntao Li, Zhiqiang Zhang, Kun Gai, Dongyan Zhao and Rui Yan . . . . . 4959

*Fine-Grained Entity Typing via Hierarchical Multi Graph Convolutional Networks*
Hailong Jin, Lei Hou, Juanzi Li and Tiansi Dong . . . . . 4969

*Learning to Infer Entities, Properties and their Relations from Clinical Conversations*
Nan Du, Mingqiu Wang, Linh Tran, Gang Lee and Izhak Shafran . . . . . 4979

*Practical Correlated Topic Modeling and Analysis via the Rectified Anchor Word Algorithm*
Moontae Lee, Sungjun Cho, David Bindel and David Mimno . . . . . 4991

*Modeling the Relationship between User Comments and Edits in Document Revision*
Xuchao Zhang, Dheeraj Rajagopal, Michael Gamon, Sujay Kumar Jauhar and ChangTien Lu 5002

*PRADO: Projection Attention Networks for Document Classification On-Device*
Karthik Krishnamoorthi, Sujith Ravi and Zornitsa Kozareva . . . . . 5012

*Subword Language Model for Query Auto-Completion*
Gyuwan Kim . . . . . 5022

*Enhancing Dialogue Symptom Diagnosis with Global Attention and Symptom Graph*
Xinzhu Lin, Xiahui He, Qin Chen, Huaixiao Tou, Zhongyu Wei and Ting Chen . . . . . 5033

*Counterfactual Story Reasoning and Generation*
Lianhui Qin, Antoine Bosselut, Ari Holtzman, Chandra Bhagavatula, Elizabeth Clark and Yejin Choi . . . . . 5043

*Encode, Tag, Realize: High-Precision Text Editing*
Eric Malmi, Sebastian Krause, Sascha Rothe, Daniil Mirylenka and Aliaksei Severyn . . . . . 5054

*Answer-guided and Semantic Coherent Question Generation in Open-domain Conversation*
Weichao Wang, Shi Feng, Daling Wang and Yifei Zhang . . . . . 5066

*Read, Attend and Comment: A Deep Architecture for Automatic News Comment Generation*
Ze Yang, Can Xu, wei wu and zhoujun li . . . . . 5077

*A Topic Augmented Text Generation Model: Joint Learning of Semantics and Structural Features*
hongyin tang, Miao Li and Beihong Jin . . . . . 5090

*LXMERT: Learning Cross-Modality Encoder Representations from Transformers*
Hao Tan and Mohit Bansal . . . . . 5100

*Phrase Grounding by Soft-Label Chain Conditional Random Field*
Jiacheng Liu and Julia Hockenmaier . . . . . 5112

*What You See is What You Get: Visual Pronoun Coreference Resolution in Dialogues*
Xintong Yu, Hongming Zhang, Yangqiu Song, Yan Song and Changshui Zhang . . . . . 5123

*YouMakeup: A Large-Scale Domain-Specific Multimodal Dataset for Fine-Grained Semantic Comprehension*
Weiying Wang, Yongcheng Wang, Shizhe Chen and Qin Jin . . . . . 5133

*DEBUG: A Dense Bottom-Up Grounding Approach for Natural Language Video Localization*
Chujie Lu, Long Chen, Chilie Tan, Xiaolin Li and Jun Xiao . . . . . 5144

*CrossWeigh: Training Named Entity Tagger from Imperfect Annotations*
Zihan Wang, Jingbo Shang, Liyuan Liu, Lihao Lu, Jiacheng Liu and Jiawei Han . . . . . 5154

*A Little Annotation does a Lot of Good: A Study in Bootstrapping Low-resource Named Entity Recognizers*
Aditi Chaudhary, Jiateng Xie, Zaid Sheikh, Graham Neubig and Jaime Carbonell . . . . . 5164

*Open Domain Web Keyphrase Extraction Beyond Language Modeling*
Lee Xiong, Chuan Hu, Chenyan Xiong, Daniel Campos and Arnold Overwijk . . . . . 5175

*TuckER: Tensor Factorization for Knowledge Graph Completion*
Ivana Balazevic, Carl Allen and Timothy Hospedales . . . . . 5185

*Human-grounded Evaluations of Explanation Methods for Text Classification*
Piyawat Lertvittayakumjorn and Francesca Toni . . . . . 5195

*A Context-based Framework for Modeling the Role and Function of On-line Resource Citations in Scientific Literature*
He Zhao, Zhunchen Luo, Chong Feng, Anqing Zheng and Xiaopeng Liu . . . . . 5206

*Adversarial Reprogramming of Text Classification Neural Networks*
Paarth Neekhara, Shehzeen Hussain, Shlomo Dubnov and Farinaz Koushanfar . . . . . 5216

*Document Hashing with Mixture-Prior Generative Models*
Wei Dong, Qinliang Su, Dinghan Shen and Changyou Chen . . . . . 5226

*On Efficient Retrieval of Top Similarity Vectors*
Shulong Tan, Zhixin Zhou, Zhaozhuo Xu and Ping Li . . . . . 5236

*Multiplex Word Embeddings for Selectional Preference Acquisition*
Hongming Zhang, Jiaxin Bai, Yan Song, Kun Xu, Changlong Yu, Yangqiu Song, Wilfred Ng and Dong Yu . . . . . 5247

*MulCode: A Multiplicative Multi-way Model for Compressing Neural Language Model*
Yukun Ma, Patrick H. Chen and Cho-Jui Hsieh . . . . . 5257

*It's All in the Name: Mitigating Gender Bias with Name-Based Counterfactual Data Substitution*
Rowan Hall Maudslay, Hila Gonen, Ryan Cotterell and Simone Teufel . . . . . 5267

*Examining Gender Bias in Languages with Grammatical Gender*
Pei Zhou, Weijia Shi, Jieyu Zhao, Kuan-Hao Huang, Muhao Chen, Ryan Cotterell and Kai-Wei Chang . . . . . 5276

*Weakly Supervised Cross-lingual Semantic Relation Classification via Knowledge Distillation*
Yogarshi Vyas and Marine Carpuat . . . . . 5285

*Improved Word Sense Disambiguation Using Pre-Trained Contextualized Word Representations*
Christian Hadiwinoto, Hwee Tou Ng and Wee Chung Gan . . . . . 5297

*Do NLP Models Know Numbers? Probing Numeracy in Embeddings*
Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh and Matt Gardner . . . . . 5307

*A Split-and-Recombine Approach for Follow-up Query Analysis*
Qian Liu, Bei Chen, Haoyan Liu, Jian-Guang LOU, Lei Fang, Bin Zhou and Dongmei Zhang 5316

*Text2Math: End-to-end Parsing Text into Math Expressions*
Yanyan Zou and Wei Lu . . . . . 5327

*Editing-Based SQL Query Generation for Cross-Domain Context-Dependent Questions*
Rui Zhang, Tao Yu, Heyang Er, Sungrok Shim, Eric Xue, Xi Victoria Lin, Tianze Shi, Caiming Xiong, Richard Socher and Dragomir Radev . . . . . 5338

*Syntax-aware Multilingual Semantic Role Labeling*
Shexia He, Zuchao Li and Hai Zhao . . . . . 5350

*Cloze-driven Pretraining of Self-attention Networks*
Alexei Baevski, Sergey Edunov, Yinhan Liu, Luke Zettlemoyer and Michael Auli . . . . . 5360

*Bridging the Gap between Relevance Matching and Semantic Matching for Short Text Similarity Modeling*
Jinfeng Rao, Linqing Liu, Yi Tay, Wei Yang, Peng Shi and Jimmy Lin . . . . . 5370

*A Syntax-aware Multi-task Learning Framework for Chinese Semantic Role Labeling*
Qingrong Xia, Zhenghua Li and Min Zhang . . . . . 5382

*Transfer Fine-Tuning: A BERT Case Study*
Yuki Arase and Jun'ichi Tsujii . . . . . 5393

*Data-Anonymous Encoding for Text-to-SQL Generation*
Zhen Dong, Shizhao Sun, Hongzhi Liu, Jian-Guang Lou and Dongmei Zhang . . . . . 5405

*Capturing Argument Interaction in Semantic Role Labeling with Capsule Networks*
Xinchi Chen, Chunchuan Lyu and Ivan Titov . . . . . 5415

*Learning Programmatic Idioms for Scalable Semantic Parsing*
Srinivasan Iyer, Alvin Cheung and Luke Zettlemoyer . . . . . 5426

*JuICe: A Large Scale Distantly Supervised Dataset for Open Domain Context-based Code Generation*
Rajas Agashe, Srinivasan Iyer and Luke Zettlemoyer . . . . . 5436

*Model-based Interactive Semantic Parsing: A Unified Framework and A Text-to-SQL Case Study*
Ziyu Yao, Yu Su, Huan Sun and Wen-tau Yih . . . . . 5447

*Modeling Graph Structure in Transformer for Better AMR-to-Text Generation*
Jie Zhu, Junhui Li, Muhua Zhu, Longhua Qian, Min Zhang and Guodong Zhou . . . . . 5459

*Syntax-Aware Aspect Level Sentiment Classification with Graph Attention Networks*
Binxuan Huang and Kathleen Carley . . . . . 5469

*Learning Explicit and Implicit Structures for Targeted Sentiment Analysis*
Hao Li and Wei Lu . . . . . 5478

*Capsule Network with Interactive Attention for Aspect-Level Sentiment Classification*
Chunning Du, Haifeng Sun, Jingyu Wang, Qi Qi, Jianxin Liao, Tong Xu and Ming Liu . . . . . 5489

*Emotion Detection with Neural Personal Discrimination*
Xiabing Zhou, Zhongqing Wang, Shoushan Li, Guodong Zhou and Min Zhang . . . . . 5499

*Specificity-Driven Cascading Approach for Unsupervised Sentiment Modification*
Pengcheng Yang, Junyang Lin, Jingjing Xu, Jun Xie, Qi Su and Xu SUN . . . . . 5508

*LexicalAT: Lexical-Based Adversarial Reinforcement Training for Robust Sentiment Classification*
Jingjing Xu, Liang Zhao, Hanqi Yan, Qi Zeng, Yun Liang and Xu SUN . . . . . 5518

*Leveraging Structural and Semantic Correspondence for Attribute-Oriented Aspect Sentiment Discovery*
Zhe Zhang and Munindar Singh . . . . . 5528

*From the Token to the Review: A Hierarchical Multimodal approach to Opinion Mining*
Alexandre Garcia, Pierre Colombo, Florence d'Alché-Buc, Slim Essid and Chloé Clavel . . . . . 5539

*Shallow Domain Adaptive Embeddings for Sentiment Analysis*
Prathusha K Sarma, Yingyu Liang and William Sethares . . . . . 5549

*Domain-Invariant Feature Distillation for Cross-Domain Sentiment Classification*
Mengting Hu, Yike Wu, Shiwan Zhao, Honglei Guo, Renhong Cheng and Zhong Su . . . . . 5559

*A Novel Aspect-Guided Deep Transition Model for Aspect Based Sentiment Analysis*
Yunlong Liang, Fandong Meng, Jinchao Zhang, Jinan Xu, Yufeng Chen and Jie Zhou . . . . . 5569

*Human-Like Decision Making: Document-level Aspect Sentiment Classification via Hierarchical Reinforcement Learning*
Jingjing Wang, Changlong Sun, Shoushan Li, Jiancheng Wang, Luo Si, Min Zhang, Xiaozhong Liu and Guodong Zhou . . . . . 5581

*A Dataset of General-Purpose Rebuttal*
Matan Orbach, Yonatan Bilu, Ariel Gera, Yoav Kantor, Lena Dankin, Tamar Lavee, Lili Kotlerman, Shachar Mirkin, Michal Jacovi, Ranit Aharonov and Noam Slonim . . . . . 5591

*Rethinking Attribute Representation and Injection for Sentiment Classification*
Reinald Kim Amplayo . . . . . 5602

*A Knowledge Regularized Hierarchical Approach for Emotion Cause Analysis*
Chuang Fan, Hongyu Yan, Jiachen Du, Lin Gui, Lidong Bing, Min Yang, Ruifeng Xu and Ruibin Mao . . . . . 5614

*Automatic Argument Quality Assessment - New Datasets and Methods*
Assaf Toledo, Shai Gretz, Edo Cohen-Karlik, Roni Friedman, Elad Venezian, Dan Lahav, Michal Jacovi, Ranit Aharonov and Noam Slonim . . . . . 5625

*Fine-Grained Analysis of Propaganda in News Article*
Giovanni Da San Martino, Seunghak Yu, Alberto Barrón-Cedeño, Rostislav Petrov and Preslav Nakov . . . . . 5636

*Context-aware Interactive Attention for Multi-modal Sentiment and Emotion Analysis*
Dushyant Singh Chauhan, Md Shad Akhtar, Asif Ekbal and Pushpak Bhattacharyya . . . . . 5647

*Sequential Learning of Convolutional Features for Effective Text Classification*
Avinash Madasu and Vijjini Anvesh Rao . . . . . 5658

*The Role of Pragmatic and Discourse Context in Determining Argument Impact*
Esin Durmus, Faisal Ladhak and Claire Cardie . . . . . 5668

*Aspect-Level Sentiment Analysis Via Convolution over Dependency Tree*
Kai Sun, Richong Zhang, Samuel Mensah, Yongyi Mao and Xudong Liu . . . . . 5679

*Understanding Data Augmentation in Neural Machine Translation: Two Perspectives towards Generalization*
Guanlin Li, Lemao Liu, Guoping Huang, Conghui Zhu and Tiejun Zhao . . . . . 5689

*Simple and Effective Noisy Channel Modeling for Neural Machine Translation*
Kyra Yee, Yann Dauphin and Michael Auli . . . . . 5696

*MultiFiT: Efficient Multi-lingual Language Model Fine-tuning*
Julian Eisenschlos, Sebastian Ruder, Piotr Czapla, Marcin Kadras, Sylvain Gugger and Jeremy Howard . . . . . 5702

*Hint-Based Training for Non-Autoregressive Machine Translation*
Zhuohan Li, Zi Lin, Di He, Fei Tian, Tao QIN, Liwei WANG and Tie-Yan Liu . . . . . 5708

*Working Hard or Hardly Working: Challenges of Integrating Typology into Neural Dependency Parsers*
Adam Fisch, Jiang Guo and Regina Barzilay . . . . . 5714

*Cross-Lingual BERT Transformation for Zero-Shot Dependency Parsing*
Yuxuan Wang, Wanxiang Che, Jiang Guo, Yijia Liu and Ting Liu . . . . . 5721

*Multilingual Grammar Induction with Continuous Language Identification*
Wenjuan Han, Ge Wang, Yong Jiang and Kewei Tu . . . . . 5728

*Quantifying the Semantic Core of Gender Systems*
Adina Williams, Damian Blasi, Lawrence Wolf-Sonkin, Hanna Wallach and Ryan Cotterell . . 5734

*Perturbation Sensitivity Analysis to Detect Unintended Model Biases*
Vinodkumar Prabhakaran, Ben Hutchinson and Margaret Mitchell . . . . . 5740

*Automatically Inferring Gender Associations from Language*
Serina Chang and Kathy McKeown . . . . . 5746

*Reporting the Unreported: Event Extraction for Analyzing the Local Representation of Hate Crimes*
Aida Mostafazadeh Davani, Leigh Yeh, Mohammad Atari, Brendan Kennedy, Gwenyth Portillo Wightman, Elaine Gonzalez, Natalie Delong, Rhea Bhatia, Arineh Mirinjian, Xiang Ren and Morteza Dehghani . . . . . 5753

*Minimally Supervised Learning of Affective Events Using Discourse Relations*
Jun Saito, Yugo Murawaki and Sadao Kurohashi . . . . . 5758

*Event Detection with Multi-Order Graph Convolution and Aggregated Attention*
Haoran Yan, Xiaolong Jin, Xiangbin Meng, Jiafeng Guo and Xueqi Cheng . . . . . 5766

*Coverage of Information Extraction from Sentences and Paragraphs*
Simon Razniewski, Nitisha Jain, Paramita Mirza and Gerhard Weikum . . . . . 5771

*HMEAE: Hierarchical Modular Event Argument Extraction*
Xiaozhi Wang, Ziqi Wang, Xu Han, Zhiyuan Liu, Juanzi Li, Peng Li, Maosong Sun, Jie Zhou and Xiang Ren . . . . . 5777

*Entity, Relation, and Event Extraction with Contextualized Span Representations*
David Wadden, Ulme Wennberg, Yi Luan and Hannaneh Hajishirzi . . . . . 5784

*Next Sentence Prediction helps Implicit Discourse Relation Classification within and across Domains*
Wei Shi and Vera Demberg . . . . . 5790

*Split or Merge: Which is Better for Unsupervised RST Parsing?*
Naoki Kobayashi, Tsutomu Hirao, Kengo Nakamura, Hidetaka Kamigaito, Manabu Okumura and Masaaki Nagata . . . . . 5797

*BERT for Coreference Resolution: Baselines and Analysis*
Mandar Joshi, Omer Levy, Luke Zettlemoyer and Daniel Weld . . . . . 5803

*Linguistic Versus Latent Relations for Modeling Coherent Flow in Paragraphs*
Dongyeop Kang and Eduard Hovy . . . . . 5809

*Event Causality Recognition Exploiting Multiple Annotators' Judgments and Background Knowledge*
Kazuma Kadowaki, Ryu Iida, Kentaro Torisawa, Jong-Hoon Oh and Julien Kloetzer . . . . . 5816

*What Part of the Neural Network Does This? Understanding LSTMs by Measuring and Dissecting Neurons*
Ji Xin, Jimmy Lin and Yaoliang Yu . . . . . 5823

*Quantity doesn't buy quality syntax with neural language models*
Marten van Schijndel, Aaron Mueller and Tal Linzen . . . . . 5831

*Higher-order Comparisons of Sentence Encoder Representations*
Mostafa Abdou, Artur Kulmizev, Felix Hill, Daniel M. Low and Anders Søgaard . . . . . 5838

*Text Genre and Training Data Size in Human-like Parsing*
John Hale, Adhiguna Kuncoro, Keith Hall, Chris Dyer and Jonathan Brennan . . . . . 5846

*Feature2Vec: Distributional semantic modelling of human property knowledge*
Steven Derby, Paul Miller and Barry Devereux . . . . . 5853

*Sunny and Dark Outside?! Improving Answer Consistency in VQA through Entailed Question Generation*
Arijit Ray, Karan Sikka, Ajay Divakaran, Stefan Lee and Giedrius Burachas . . . . . 5860

*GeoSQA: A Benchmark for Scenario-based Question Answering in the Geography Domain at High School Level*
Zixian Huang, Yulin Shen, Xiao Li, Yu'ang Wei, Gong Cheng, Lin Zhou, Xinyu Dai and Yuzhong Qu . . . . . 5866

*Revisiting the Evaluation of Theory of Mind through Question Answering*
Matthew Le, Y-Lan Boureau and Maximilian Nickel . . . . . 5872

*Multi-passage BERT: A Globally Normalized BERT Model for Open-domain Question Answering*
Zhiguo Wang, Patrick Ng, Xiaofei Ma, Ramesh Nallapati and Bing Xiang . . . . . 5878

*A Span-Extraction Dataset for Chinese Machine Reading Comprehension*
Yiming Cui, Ting Liu, Wanxiang Che, Li Xiao, Zhipeng Chen, Wentao Ma, Shijin Wang and Guoping Hu . . . . . 5883

*MICRON: Multigranular Interaction for Contextualizing RepresentatiON in Non-factoid Question Answering*
Hojae Han, Seungtaek Choi, Haeju Park and Seung-won Hwang . . . . . 5890

*Machine Reading Comprehension Using Structural Knowledge Graph-aware Network*
Delai Qiu, Yuanzhe Zhang, Xinwei Feng, Xiangwen Liao, Wenbin Jiang, Yajuan Lyu, Kang Liu and Jun Zhao . . . . . 5896

*Answering Conversational Questions on Structured Data without Logical Forms*
Thomas Mueller, Francesco Piccinno, Peter Shaw, Massimo Nicosia and Yasemin Altun . . . . . 5902

*Improving Answer Selection and Answer Triggering using Hard Negatives*
Sawan Kumar, shweta garg, Kartik Mehta and Nikhil Rasiwasia . . . . . 5911

*Can You Unpack That? Learning to Rewrite Questions-in-Context*
Ahmed Elgohary, Denis Peskov and Jordan Boyd-Graber . . . . . 5918

*Quoref: A Reading Comprehension Dataset with Questions Requiring Coreferential Reasoning*
Pradeep Dasigi, Nelson F. Liu, Ana Marasovic, Noah A. Smith and Matt Gardner . . . . . 5925

*Zero-shot Reading Comprehension by Cross-lingual Transfer Learning with Multi-lingual Language Representation Model*
Tsung-Yuan Hsu, Chi-Liang Liu and Hung-yi Lee . . . . . 5933

*QuaRTz: An Open-Domain Dataset of Qualitative Relationship Questions*
Oyvind Tafjord, Matt Gardner, Kevin Lin and Peter Clark . . . . . 5941

*Giving BERT a Calculator: Finding Operations and Arguments with Reading Comprehension*
Daniel Andor, Luheng He, Kenton Lee and Emily Pitler . . . . . 5947

*A Gated Self-attention Memory Network for Answer Selection*
Tuan Lai, Quan Hung Tran, Trung Bui and Daisuke Kihara . . . . . 5953

*Polly Want a Cracker: Analyzing Performance of Parroting on Paraphrase Generation Datasets*
Hong-Ren Mao and Hung-Yi Lee . . . . . 5960

*Query-focused Sentence Compression in Linear Time*
Abram Handler and Brendan O'Connor . . . . . 5969

*Generating Personalized Recipes from Historical User Preferences*
Bodhisattwa Prasad Majumder, Shuyang Li, Jianmo Ni and Julian McAuley . . . . . 5976

*Generating Highly Relevant Questions*
Jiazuo Qiu and Deyi Xiong . . . . . 5983

*Improving Neural Story Generation by Targeted Common Sense Grounding*
Huanru Henry Mao, Bodhisattwa Prasad Majumder, Julian McAuley and Garrison Cottrell . . 5988

*Abstract Text Summarization: A Low Resource Challenge*
Shantipriya Parida and Petr Motlicek . . . . . 5994

*Generating Modern Poetry Automatically in Finnish*
Mika Hämäläinen and Khalid Alnajjar . . . . . 5999

*SUM-QE: a BERT-based Summary Quality Estimation Model*
Stratos Xenouleas, Prodromos Malakasiotis, Marianna Apidianaki and Ion Androutsopoulos . 6005

*An Empirical Comparison on Imitation Learning and Reinforcement Learning for Paraphrase Generation*
Wanyu Du and Yangfeng Ji . . . . . 6012

*Countering the Effects of Lead Bias in News Summarization via Multi-Stage Training and Auxiliary Losses*
Matt Grenander, Yue Dong, Jackie Chi Kit Cheung and Annie Louis . . . . . 6019

*Learning Rhyming Constraints using Structured Adversaries*
Harsh Jhamtani, Sanket Vaibhav Mehta, Jaime Carbonell and Taylor Berg-Kirkpatrick . . . . . 6025

*Question-type Driven Question Generation*
Wenjie Zhou, Minghua Zhang and Yunfang Wu . . . . . 6032

*Deep Reinforcement Learning with Distributional Semantic Rewards for Abstractive Summarization*
Siyao Li, Deren Lei, Pengda Qin and William Yang Wang . . . . . 6038

*Clause-Wise and Recursive Decoding for Complex and Cross-Domain Text-to-SQL Generation*
Dongjun Lee . . . . . 6045

*Do Nuclear Submarines Have Nuclear Captains? A Challenge Dataset for Commonsense Reasoning over Adjectives and Objects*
James Mullenbach, Jonathan Gordon, Nanyun Peng and Jonathan May . . . . . 6052

*Aggregating Bidirectional Encoder Representations Using MatchLSTM for Sequence Matching*
Bo Shao, Yeyun Gong, Weizhen Qi, Nan Duan and Xiaola Lin . . . . . 6059

*What Does This Word Mean? Explaining Contextualized Embeddings with Natural Language Definition*
Ting-Yun Chang and Yun-Nung Chen . . . . . 6064

*Pre-Training BERT on Domain Resources for Short Answer Grading*
Chul Sung, Tejas Dhamecha, Swarnadeep Saha, Tengfei Ma, Vinay Reddy and Rishi Arora . . 6071

*WIQA: A dataset for "What if..." reasoning over procedural text*
Niket Tandon, Bhavana Dalvi, Keisuke Sakaguchi, Peter Clark and Antoine Bosselut . . . . . 6076

*Evaluating BERT for natural language inference: A case study on the CommitmentBank*
Nanjiang Jiang and Marie-Catherine de Marneffe . . . . . 6086

*Incorporating Domain Knowledge into Medical NLI using Knowledge Graphs*
Soumya Sharma, Bishal Santra, Abhik Jana, Santosh Tokala, Niloy Ganguly and Pawan Goyal6092

*The FLORES Evaluation Datasets for Low-Resource Machine Translation: Nepali–English and Sinhala–English*
Francisco Guzmán, Peng-Jen Chen, Myle Ott, Juan Pino, Guillaume Lample, Philipp Koehn, Vishrav Chaudhary and Marc'Aurelio Ranzato . . . . . 6098

*Mask-Predict: Parallel Decoding of Conditional Masked Language Models*
Marjan Ghazvininejad, Omer Levy, Yinhan Liu and Luke Zettlemoyer . . . . . 6112

*Learning to Copy for Automatic Post-Editing*
Xuancheng Huang, Yang Liu, Huanbo Luan, Jingfang Xu and Maosong Sun . . . . . 6122

*Exploring Human Gender Stereotypes with Word Association Test*
Yupei Du, Yuanbin Wu and Man Lan . . . . . 6133

*A Modular Architecture for Unsupervised Sarcasm Generation*
Abhijit Mishra, Tarun Tater and Karthik Sankaranarayanan . . . . . 6144

*Generating Classical Chinese Poems from Vernacular Chinese*
Zhichao Yang, Pengshan Cai, Yansong Feng, Fei Li, Weijiang Feng, Elena Suet-Ying Chiu and hong yu . . . . . 6155

*Set to Ordered Text: Generating Discharge Instructions from Medical Billing Codes*
Litton J Kurisinkel and Nancy Chen . . . . . 6165

*Constraint-based Learning of Phonological Processes*
Shraddha Barke, Rose Kunkel, Nadia Polikarpova, Eric Meinhardt, Eric Bakovic and Leon Bergen 6176

*Detect Camouflaged Spam Content via StoneSkipping: Graph and Text Joint Embedding for Chinese Character Variation Representation*
Zhuoren Jiang, Zhe Gao, Guoxiu He, Yangyang Kang, Changlong Sun, Qiong Zhang, Luo Si and Xiaozhong Liu . . . . . 6187

*An Attentive Fine-Grained Entity Typing Model with Latent Type Representation*
Ying Lin and Heng Ji . . . . . 6197

*An Improved Neural Baseline for Temporal Relation Extraction*
Qiang Ning, Sanjay Subramanian and Dan Roth . . . . . 6203

*Improving Fine-grained Entity Typing with Entity Linking*
Hongliang Dai, Donghong Du, Xin Li and Yangqiu Song . . . . . 6210

*Combining Spans into Entities: A Neural Two-Stage Approach for Recognizing Discontiguous Entities*
Bailin Wang and Wei Lu . . . . . 6216

*Cross-Sentence N-ary Relation Extraction using Lower-Arity Universal Schemas*
Kosuke Akimoto, Takuya Hiraoka, Kunihiko Sadamasa and Mathias Niepert . . . . . 6225

*Gazetteer-Enhanced Attentive Neural Networks for Named Entity Recognition*
Hongyu Lin, Yaojie Lu, Xianpei Han, Le Sun, Bin Dong and Shanshan Jiang . . . . . 6232

*"A Buster Keaton of Linguistics": First Automated Approaches for the Extraction of Vossian Antonomasia*
Michel Schwab, Robert Jäschke, Frank Fischer and Jannik Strötgen . . . . . 6238

*Multi-Task Learning for Chemical Named Entity Recognition with Chemical Compound Paraphrasing*
Taiki Watanabe, Akihiro Tamura, Takashi Ninomiya, Takuya Makino and Tomoya Iwakura . . 6244

*FewRel 2.0: Towards More Challenging Few-Shot Relation Classification*
Tianyu Gao, Xu Han, Hao Zhu, Zhiyuan Liu, Peng Li, Maosong Sun and Jie Zhou . . . . . 6250

*ner and pos when nothing is capitalized*
Stephen Mayhew, Tatiana Tsygankova and Dan Roth . . . . . 6256

*CaRB: A Crowdsourced Benchmark for Open IE*
Sangnie Bhardwaj, Samarth Aggarwal and Mausam Mausam . . . . . 6262

*Weakly Supervised Attention Networks for Entity Recognition*
Barun Patra and Joel Ruben Antony Moniz . . . . . 6268

*Revealing and Predicting Online Persuasion Strategy with Elementary Units*
Gaku Morio, Ryo Egawa and Katsuhide Fujita . . . . . 6274

*A Challenge Dataset and Effective Models for Aspect-Based Sentiment Analysis*
Qingnan Jiang, Lei Chen, Ruifeng Xu, Xiang Ao and Min Yang . . . . . 6280

*Learning with Noisy Labels for Sentence-level Sentiment Classification*
Hao Wang, Bing Liu, Chaozhuo Li, Yan Yang and Tianrui Li . . . . . 6286

*DENS: A Dataset for Multi-class Emotion Analysis*
Chen Liu, Muhammad Osama and Anderson De Andrade . . . . . 6293

*Multi-Task Stance Detection with Sentiment and Stance Lexicons*
Yingjie Li and Cornelia Caragea . . . . . 6299

*A Robust Self-Learning Framework for Cross-Lingual Text Classification*
Xin Dong and Gerard de Melo . . . . . 6306

*Learning to Flip the Sentiment of Reviews from Non-Parallel Corpora*
Canasai Kruengkrai . . . . . 6311

*Label Embedding using Hierarchical Structure of Labels for Twitter Classification*
Taro Miyazaki, Kiminobu Makino, Yuka Takei, Hiroki Okamoto and Jun Goto . . . . . 6317

*Interpretable Word Embeddings via Informative Priors*
Miriam Hurtado Bodell, Martin Arvidsson and Måns Magnusson . . . . . 6323

*Adversarial Removal of Demographic Attributes Revisited*
Maria Barrett, Yova Kementchedjhieva, Yanai Elazar, Desmond Elliott and Anders Søgaard . . 6330

*A deep-learning framework to detect sarcasm targets*
Jasabanta Patro, Srijan Bansal and Animesh Mukherjee . . . . . 6336

*In Plain Sight: Media Bias Through the Lens of Factual Reporting*
Lisa Fan, Marshall White, Eva Sharma, Ruisi Su, Prafulla Kumar Choubey, Ruihong Huang and Lu Wang . . . . . 6343

*Incorporating Label Dependencies in Multilabel Stance Detection*
William Ferreira and Andreas Vlachos . . . . . 6350

*Investigating Sports Commentator Bias within a Large Corpus of American Football Broadcasts*
Jack Merullo, Luke Yeh, Abram Handler, Alvin Grissom II, Brendan O'Connor and Mohit Iyyer 6355

*Charge-Based Prison Term Prediction with Deep Gating Network*
Huajie Chen, Deng Cai, Wei Dai, Zehui Dai and Yadong Ding . . . . . 6362

*Restoring ancient text using deep learning: a case study on Greek epigraphy*
Yannis Assael, Thea Sommerschield and Jonathan Prag . . . . . 6368

*Embedding Lexical Features via Tensor Decomposition for Small Sample Humor Recognition*
Zhenjie Zhao, Andrew Cattle, Evangelos Papalexakis and Xiaojuan Ma . . . . . 6376

*EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks*
Jason Wei and Kai Zou . . . . . 6382

*Neural News Recommendation with Multi-Head Self-Attention*
Chuhan Wu, Fangzhao Wu, Suyu Ge, Tao Qi, Yongfeng Huang and Xing Xie . . . . . 6389

*What Matters for Neural Cross-Lingual Named Entity Recognition: An Empirical Analysis*
Xiaolei Huang, Jonathan May and Nanyun Peng . . . . . 6395

*Telling the Whole Story: A Manually Annotated Chinese Dataset for the Analysis of Humor in Jokes*
Dongyu Zhang, Heting Zhang, Xikai Liu, Hongfei LIN and Feng Xia . . . . . 6402

*Generating Natural Anagrams: Towards Language Generation Under Hard Combinatorial Constraints*
Masaaki Nishino, Sho Takase, Tsutomu Hirao and Masaaki Nagata . . . . . 6408

*STANCY: Stance Classification Based on Consistency Cues*
Kashyap Popat, Subhabrata Mukherjee, Andrew Yates and Gerhard Weikum . . . . . 6413

*Cross-lingual intent classification in a low resource industrial setting*
Talaat Khalil, Kornel Kiełczewski, Georgios Christos Chouliaras, Amina Keldibek and Maarten Versteegh . . . . . 6419

*SoftRegex: Generating Regex from Natural Language Descriptions using Softened Regex Equivalence*
Jun-U Park, Sang-Ki Ko, Marco Cognetta and Yo-Sub Han . . . . . 6425

*Using Clinical Notes with Time Series Data for ICU Management*
Swaraj Khadanga, Karan Aggarwal, Shafiq Joty and Jaideep Srivastava . . . . . 6432

*Spelling-Aware Construction of Macaronic Texts for Teaching Foreign-Language Vocabulary*
Adithya Renduchintala, Philipp Koehn and Jason Eisner . . . . . 6438

*Towards Machine Reading for Interventions from Humanitarian-Assistance Program Literature*
Bonan Min, Yee Seng Chan, Haoling Qiu and Joshua Fasching . . . . . 6444

*RUN through the Streets: A New Dataset and Baseline Models for Realistic Urban Navigation*
Tzuf Paz-Argaman and Reut Tsarfaty . . . . . 6449

*Context-Aware Conversation Thread Detection in Multi-Party Chat*
Ming Tan, Dakuo Wang, Yupeng Gao, Haoyu Wang, Saloni Potdar, Xiaoxiao Guo, Shiyu Chang and Mo Yu . . . . . 6456

# Conference Program

**Tuesday, November 5, 2019**

08:45–09:00 *Opening Remarks*

09:00–10:00 *Keynote I: Noam Slonim*

10:00–10:30 *Coffee Break*

10:30–12:00 *Session 1*

**Session 1A: Machine Learning I**

10:30–10:48 *Attending to Future Tokens for Bidirectional Sequence Generation*
Carolin Lawrence, Bhushan Kotnis and Mathias Niepert

10:48–11:06 *Attention is not not Explanation*
Sarah Wiegreffe and Yuval Pinter

11:06–11:24 *Practical Obstacles to Deploying Active Learning*
David Lowell, Zachary C. Lipton and Byron C. Wallace

11:24–11:42 *Transfer Learning Between Related Tasks Using Expected Label Proportions*
Matan Ben Noach and Yoav Goldberg

11:42–12:00 [TACL] *Insertion-based Decoding with automatically Inferred Generation Order*
Jiatao Gu, Qi Liu and Kyunghyun Cho

### Session 1B: Lexical Semantics I

10:30–10:48 *Knowledge Enhanced Contextual Word Representations*
Matthew E. Peters, Mark Neumann, Robert Logan, Roy Schwartz, Vidur Joshi, Sameer Singh and Noah A. Smith

10:48–11:06 *How Contextual are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings*
Kawin Ethayarajh

11:06–11:24 *Room to Glo: A Systematic Comparison of Semantic Change Detection Approaches with Word Embeddings*
Philippa Shoemark, Farhana Ferdousi Liza, Dong Nguyen, Scott Hale and Barbara McGillivray

11:24–11:42 *Correlations between Word Vector Sets*
Vitalii Zhelezniak, April Shen, Daniel Busbridge, Aleksandar Savkov and Nils Hammerla

11:42–12:00 *Game Theory Meets Embeddings: a Unified Framework for Word Sense Disambiguation*
Rocco Tripodi and Roberto Navigli

### Session 1C: Dialog and Interactive Systems I

10:30–10:48 *Guided Dialog Policy Learning: Reward Estimation for Multi-Domain Task-Oriented Dialog*
Ryuichi Takanobu, Hanlin Zhu and Minlie Huang

10:48–11:06 *Multi-hop Selector Network for Multi-turn Response Selection in Retrieval-based Chatbots*
Chunyuan Yuan, Wei Zhou, Mingming Li, Shangwen Lv, Fuqing Zhu, Jizhong Han and Songlin Hu

11:06–11:24 *MoEL: Mixture of Empathetic Listeners*
Zhaojiang Lin, Andrea Madotto, Jamin Shin, Peng Xu and Pascale Fung

11:24–11:42 *Entity-Consistent End-to-end Task-Oriented Dialogue System with KB Retriever*
Libo Qin, Yijia Liu, Wanxiang Che, Haoyang Wen, Yangming Li and Ting Liu

11:42–12:00 *Building Task-Oriented Visual Dialog Systems Through Alternative Optimization Between Dialog Policy and Language Generation*
Mingyang Zhou, Josh Arnold and Zhou Yu

**Session 1D: Sentiment Analysis and Argument Mining I**

10:30–10:48 *DialogueGCN: A Graph Convolutional Neural Network for Emotion Recognition in Conversation*
Deepanway Ghosal, Navonil Majumder, Soujanya Poria, Niyati Chhaya and Alexander Gelbukh

10:48–11:06 *Knowledge-Enriched Transformer for Emotion Detection in Textual Conversations*
Peixiang Zhong, Di Wang and Chunyan Miao

11:06–11:24 *Interpretable Relevant Emotion Ranking with Event-Driven Attention*
Yang Yang, Deyu ZHOU, Yulan He and Meng Zhang

11:24–11:42 *Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects*
Jianmo Ni, Jiacheng Li and Julian McAuley

11:42–12:00 *Using Customer Service Dialogues for Satisfaction Analysis with Context-Assisted Multiple Instance Learning*
Kaisong Song, Lidong Bing, Wei Gao, Jun Lin, Lujun Zhao, Jiancheng Wang, Changlong Sun, Xiaozhong Liu and Qiong Zhang

**Poster and Demo Session 1: Information Extraction, Information Retrieval and Document Analysis, Linguistic Theories**

*Leveraging Dependency Forest for Neural Medical Relation Extraction*
Linfeng Song, Yue Zhang, Daniel Gildea, Mo Yu, Zhiguo Wang and jinsong su

*Open Relation Extraction: Relational Knowledge Transfer from Supervised Data to Unsupervised Data*
Ruidong Wu, Yuan Yao, Xu Han, Ruobing Xie, Zhiyuan Liu, Fen Lin, Leyu Lin and Maosong Sun

*Improving Relation Extraction with Knowledge-attention*
Pengfei Li, Kezhi Mao, Xuefeng Yang and Qi Li

*Jointly Learning Entity and Relation Representations for Entity Alignment*
Yuting Wu, Xiao Liu, Yansong Feng, Zheng Wang and Dongyan Zhao

*Tackling Long-Tailed Relations and Uncommon Entities in Knowledge Graph Completion*
Zihao Wang, Kwunping Lai, Piji Li, Lidong Bing and Wai Lam

*Low-Resource Name Tagging Learned with Weakly Labeled Data*
Yixin Cao, Zikun Hu, Tat-seng Chua, Zhiyuan Liu and Heng Ji

*Learning Dynamic Context Augmentation for Global Entity Linking*
Xiyuan Yang, Xiaotao Gu, Sheng Lin, Siliang Tang, Yueting Zhuang, Fei Wu, Zhigang Chen, Guoping Hu and Xiang Ren

*Open Event Extraction from Online Text using a Generative Adversarial Network*
Rui Wang, Deyu ZHOU and Yulan He

*Learning to Bootstrap for Entity Set Expansion*
Lingyong Yan, Xianpei Han, Le Sun and Ben He

*Multi-Input Multi-Output Sequence Labeling for Joint Extraction of Fact and Condition Tuples from Scientific Text*
Tianwen Jiang, Tong Zhao, Bing Qin, Ting Liu, Nitesh Chawla and Meng Jiang

*Cross-lingual Structure Transfer for Relation and Event Extraction*
Ananya Subburathinam, Di Lu, Heng Ji, Jonathan May, Shih-Fu Chang, Avirup Sil and Clare Voss

*Uncover the Ground-Truth Relations in Distant Supervision: A Neural Expectation-Maximization Framework*
Junfan Chen, Richong Zhang, Yongyi Mao, Hongyu Guo and Jie Xu

*Doc2EDAG: An End-to-End Document-level Framework for Chinese Financial Event Extraction*
Shun Zheng, Wei Cao, Wei Xu and Jiang Bian

*Event Detection with Trigger-Aware Lattice Neural Network*
Ning Ding, Ziran Li, Zhiyuan Liu, Haitao Zheng and Zibo Lin

*A Boundary-aware Neural Model for Nested Named Entity Recognition*
Changmeng Zheng, Yi Cai, Jingyun Xu, Ho-fung Leung and Guandong Xu

*Learning the Extraction Order of Multiple Relational Facts in a Sentence with Reinforcement Learning*
Xiangrong Zeng, Shizhu He, Daojian Zeng, Kang Liu, Shengping Liu and Jun Zhao

*CaRe: Open Knowledge Graph Embeddings*
Swapnil Gupta, Sreyash Kenkre and Partha Talukdar

*Self-Attention Enhanced CNNs and Collaborative Curriculum Learning for Distantly Supervised Relation Extraction*
Yuyun Huang and Jinhua Du

*Neural Cross-Lingual Relation Extraction Based on Bilingual Word Embedding Mapping*
Jian Ni and Radu Florian

*Leveraging 2-hop Distant Supervision from Table Entity Pairs for Relation Extraction*
xiang deng and Huan Sun

*EntEval: A Holistic Evaluation Benchmark for Entity Representations*
Mingda Chen, Zewei Chu, Yang Chen, Karl Stratos and Kevin Gimpel

*Joint Event and Temporal Relation Extraction with Shared Representations and Structured Prediction*
Rujun Han, Qiang Ning and Nanyun Peng

*Hierarchical Text Classification with Reinforced Label Assignment*
Yuning Mao, Jingjing Tian, Jiawei Han and Xiang Ren

*Investigating Capsule Network and Semantic Feature on Hyperplanes for Text Classification*
Chunning Du, Haifeng Sun, Jingyu Wang, Qi Qi, Jianxin Liao, Chun Wang and Bing Ma

*Label-Specific Document Representation for Multi-Label Text Classification*
Lin Xiao, Xin Huang, Boli Chen and Liping Jing

*Hierarchical Attention Prototypical Networks for Few-Shot Text Classification*
Shengli Sun, Qingfeng Sun, Kevin Zhou and Tengchao Lv

*Many Faces of Feature Importance: Comparing Built-in and Post-hoc Feature Importance in Text Classification*
Vivian Lai, Zheng Cai and Chenhao Tan

*Enhancing Local Feature Extraction with Global Representation for Neural Text Classification*
Guocheng Niu, Hengru Xu, Bolei He, Xinyan Xiao, Hua Wu and Sheng GAO

*Latent-Variable Generative Models for Data-Efficient Text Classification*
Xiaoan Ding and Kevin Gimpel

*PaRe: A Paper-Reviewer Matching Approach Using a Common Topic Space*
Omer Anjum, Hongyu Gong, Suma Bhat, Wen-Mei Hwu and JinJun Xiong

*Linking artificial and human neural representations of language*
Jon Gauthier and Roger Levy

[DEMO] *IFlyLegal: A Chinese Legal System for Consultation, Law Searching, and Document Analysis*
Ziyue Wang, Baoxin Wang, Xingyi Duan, Dayong Wu, Shijin Wang, Guoping Hu and Ting Liu

[DEMO] *TellMeWhy: Learning to Explain Corrective Feedback for Second Language Learners*
Yi-Huei Lai and Jason Chang

[DEMO] *Honkling: In-Browser Personalization for Ubiquitous Keyword Spotting*
Jaejun Lee, Raphael Tang and Jimmy Lin

[DEMO] *Redcoat: A Collaborative Annotation Tool for Hierarchical Entity Typing*
Michael Stewart, Wei Liu and Rachel Cardell-Oliver

[DEMO] *SEAGLE: A Platform for Comparative Evaluation of Semantic Encoders for Information Retrieval*
Fabian David Schmidt, Markus Dietsche, Simone Paolo Ponzetto and Goran Glavaš

[DEMO] *OpenNRE: An Open and Extensible Toolkit for Neural Relation Extraction*
Xu Han, Tianyu Gao, Yuan Yao, Deming Ye, Zhiyuan Liu and Maosong Sun

[DEMO] *Automatic Taxonomy Induction and Expansion*
Nicolas Rodolfo Fauceglia, Alfio Gliozzo, Sarthak Dash, Md. Faisal Mahbub Chowdhury and Nandana Mihindukulasooriya

[DEMO] *Applying BERT to Document Retrieval with Birch*
Zeynep Akkalyoncu Yilmaz, Shengjin Wang, Wei Yang, Haotian Zhang and Jimmy Lin

12:00–13:30 *Lunch*

13:30–15:00 *Session 2*

### Session 2A: Summarization and Generation

13:30–13:48 *Neural Text Summarization: A Critical Evaluation*
Wojciech Kryscinski, Nitish Shirish Keskar, Bryan McCann, Caiming Xiong and Richard Socher

13:48–14:06 *Neural data-to-text generation: A comparison between pipeline and end-to-end architectures*
Thiago Castro Ferreira, Chris van der Lee, Emiel van Miltenburg and Emiel Krahmer

14:06–14:24 *MoverScore: Text Generation Evaluating with Contextualized Embeddings and Earth Mover Distance*
Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M. Meyer and Steffen Eger

14:24–14:42 *Select and Attend: Towards Controllable Content Selection in Text Generation*
Xiaoyu Shen, Jun Suzuki, Kentaro Inui, Hui Su, Dietrich Klakow and Satoshi Sekine

14:42–15:00 *Sentence-Level Content Planning and Style Specification for Neural Text Generation*
Xinyu Hua and Lu Wang

### Session 2B: Sentence-level Semantics I

13:30–13:48 *Translate and Label! An Encoder-Decoder Approach for Cross-lingual Semantic Role Labeling*
Angel Daza and Anette Frank

13:48–14:06 *Syntax-Enhanced Self-Attention-Based Semantic Role Labeling*
Yue Zhang, Rui Wang and Luo Si

14:06–14:24 *VerbAtlas: a Novel Large-Scale Verbal Semantic Resource and Its Application to Semantic Role Labeling*
Andrea Di Fabio, Simone Conia and Roberto Navigli

14:24–14:42 *Parameter-free Sentence Embedding via Orthogonal Basis*
Ziyi Yang, Chenguang Zhu and Weizhu Chen

14:42–15:00 *Evaluation Benchmarks and Learning Criteria for Discourse-Aware Sentence Representations*
Mingda Chen, Zewei Chu and Kevin Gimpel

### Session 2C: Speech, Vision, Robotics, Multimodal and Grounding I

13:30–13:48 *Extracting Possessions from Social Media: Images Complement Language*
Dhivya Chinnappa, Srikala Murugan and Eduardo Blanco

13:48–14:06 *Learning to Speak and Act in a Fantasy Text Adventure Game*
Jack Urbanek, Angela Fan, Siddharth Karamcheti, Saachi Jain, Samuel Humeau, Emily Dinan, Tim Rocktäschel, Douwe Kiela, Arthur Szlam and Jason Weston

14:06–14:24 *Help, Anna! Visual Navigation with Natural Multimodal Assistance via Retrospective Curiosity-Encouraging Imitation Learning*
Khanh Nguyen and Hal Daumé III

14:24–14:42 *Incorporating Visual Semantics into Sentence Representations within a Grounded Space*
Patrick Bordes, Eloi Zablocki, Laure Soulier, Benjamin Piwowarski and patrick Gallinari

14:42–15:00 *Neural Naturalist: Generating Fine-Grained Image Comparisons*
Maxwell Forbes, Christine Kaeser-Chen, Piyush Sharma and Serge Belongie

### Session 2D: Information Extraction I

13:30–13:48 *Fine-Grained Evaluation for Entity Linking*
Henry Rosales-Méndez, Aidan Hogan and Barbara Poblete

13:48–14:06 *Supervising Unsupervised Open Information Extraction Models*
Arpita Roy, Youngja Park, Taesung Lee and Shimei Pan

14:06–14:24 *Neural Cross-Lingual Event Detection with Minimal Parallel Resources*
Jian Liu, Yubo Chen, Kang Liu and Jun Zhao

14:24–14:42 *KnowledgeNet: A Benchmark Dataset for Knowledge Base Population*
Filipe Mesquita, Matteo Cannaviccio, Jordan Schmidek, Paramita Mirza and Denilson Barbosa

14:42–15:00 *Effective Use of Transformer Networks for Entity Tracking*
Aditya Gupta and Greg Durrett

**Poster and Demo Session 2: Machine Translation and Mulitilinguality, Phonology, Morphology and Word Segmentation, Tagging, Chunking, Syntax and Parsing**

*Explicit Cross-lingual Pre-training for Unsupervised Machine Translation*
Shuo Ren, Yu Wu, Shujie Liu, Ming Zhou and Shuai Ma

*Latent Part-of-Speech Sequences for Neural Machine Translation*
Xuewen Yang, Yingru Liu, Dongliang Xie, Xin Wang and Niranjan Balasubramanian

*Improving Back-Translation with Uncertainty-based Confidence Estimation*
Shuo Wang, Yang Liu, Chao Wang, Huanbo Luan and Maosong Sun

*Towards Linear Time Neural Machine Translation with Capsule Networks*
Mingxuan Wang

*Modeling Multi-mapping Relations for Precise Cross-lingual Entity Alignment*
Xiaofei Shi and Yanghua Xiao

*Supervised and Nonlinear Alignment of Two Embedding Spaces for Dictionary Induction in Low Resourced Languages*
Masud Moshtaghi

*Beto, Bentz, Becas: The Surprising Cross-Lingual Effectiveness of BERT*
Shijie Wu and Mark Dredze

*Iterative Dual Domain Adaptation for Neural Machine Translation*
Jiali Zeng, Yang Liu, jinsong su, yubing Ge, Yaojie Lu, Yongjing Yin and jiebo luo

*Multi-agent Learning for Neural Machine Translation*
tianchi bi, hao xiong, Zhongjun He, Hua Wu and Haifeng Wang

*Pivot-based Transfer Learning for Neural Machine Translation between Non-English Languages*
Yunsu Kim, Petre Petrov, Pavel Petrushkov, Shahram Khadivi and Hermann Ney

*Context-Aware Monolingual Repair for Neural Machine Translation*
Elena Voita, Rico Sennrich and Ivan Titov

*Multi-Granularity Self-Attention for Neural Machine Translation*
Jie Hao, Xing Wang, Shuming Shi, Jinfeng Zhang and Zhaopeng Tu

*Improving Deep Transformer with Depth-Scaled Initialization and Merged Attention*
Biao Zhang, Ivan Titov and Rico Sennrich

*A Discriminative Neural Model for Cross-Lingual Word Alignment*
Elias Stengel-Eskin, Tzu-ray Su, Matt Post and Benjamin Van Durme

*One Model to Learn Both: Zero Pronoun Prediction and Translation*
Longyue Wang, Zhaopeng Tu, Xing Wang and Shuming Shi

*Dynamic Past and Future for Neural Machine Translation*
Zaixiang Zheng, Shujian Huang, Zhaopeng Tu, XIN-YU DAI and Jiajun CHEN

*Revisit Automatic Error Detection for Wrong and Missing Translation – A Supervised Approach*
Wenqiang Lei, Weiwen Xu, Ai Ti Aw, Yuanxin Xiang and Tat Seng Chua

*Towards Understanding Neural Machine Translation with Word Importance*
Shilin He, Zhaopeng Tu, Xing Wang, Longyue Wang, Michael Lyu and Shuming Shi

*Multilingual Neural Machine Translation with Language Clustering*
Xu Tan, Jiale Chen, Di He, Yingce Xia, Tao QIN and Tie-Yan Liu

*Don't Forget the Long Tail! A Comprehensive Analysis of Morphological Generalization in Bilingual Lexicon Induction*
Paula Czarnowska, Sebastian Ruder, Edouard Grave, Ryan Cotterell and Ann Copestake

*Pushing the Limits of Low-Resource Morphological Inflection*
Antonios Anastasopoulos and Graham Neubig

[TACL] *Morphological Analysis Using a Sequence Decoder*
Ekin Akyürek, Erenay Dayanık and Deniz Yuret

*Cross-Lingual Dependency Parsing Using Code-Mixed TreeBank*
Meishan Zhang, Yue Zhang and Guohong Fu

*Hierarchical Pointer Net Parsing*
Linlin Liu, Xiang Lin, Shafiq Joty, Simeng Han and Lidong Bing

*Semi-Supervised Semantic Role Labeling with Cross-View Training*
Rui Cai and Mirella Lapata

*Low-Resource Sequence Labeling via Unsupervised Multilingual Contextualized Representations*
Zuyi Bao, Rui Huang, Chen Li and Kenny Zhu

*A Lexicon-Based Graph Neural Network for Chinese NER*
Tao Gui, Yicheng Zou, Qi Zhang, Minlong Peng, Jinlan Fu, Zhongyu Wei and Xuanjing Huang

*CM-Net: A Novel Collaborative Memory Network for Spoken Language Understanding*
Yijin Liu, Fandong Meng, Jinchao Zhang, Jie Zhou, Yufeng Chen and Jinan Xu

*Tree Transformer: Integrating Tree Structures into Self-Attention*
Yaushian Wang, Hung-Yi Lee and Yun-Nung Chen

*Semantic Role Labeling with Iterative Structure Refinement*
Chunchuan Lyu, Shay B. Cohen and Ivan Titov

*Entity Projection via Machine Translation for Cross-Lingual NER*
Alankar Jain, Bhargavi Paranjape and Zachary C. Lipton

*A Bayesian Approach for Sequence Tagging with Crowds*
Edwin D. Simpson and Iryna Gurevych

*A systematic comparison of methods for low-resource dependency parsing on genuinely low-resource languages*
Clara Vania, Yova Kementchedjhieva, Anders Søgaard and Adam Lopez

*Target Language-Aware Constrained Inference for Cross-lingual Dependency Parsing*
Tao Meng, Nanyun Peng and Kai-Wei Chang

*Look-up and Adapt: A One-shot Semantic Parser*
Zhichu Lu, Forough Arabshahi, Igor Labutov and Tom Mitchell

*Similarity Based Auxiliary Classifier for Named Entity Recognition*
Shiyuan Xiao, Yuanxin Ouyang, Wenge Rong, Jianxin Yang and Zhang Xiong

*Variable beam search for generative neural parsing and its relevance for the analysis of neuro-imaging signal*
Benoit Crabbé, Murielle Fabre and Christophe Pallier

[DEMO] *MY-AKKHARA: A Romanization-based Burmese (Myanmar) Input Method*
Chenchen Ding, Masao Utiyama and Eiichiro Sumita

[DEMO] *LINSPECTOR WEB: A Multilingual Probing Suite for Word Representations*
Max Eichler, Gözde Gül Şahin and Iryna Gurevych

[DEMO] *Joey NMT: A Minimalist NMT Toolkit for Novices*
Julia Kreutzer, Joost Bastings and Stefan Riezler

[DEMO] *Multilingual, Multi-scale and Multi-layer Visualization of Intermediate Representations*
Carlos Escolano, Marta R. Costa-jussà, Elora Lacroux and Pere-Pau Vázquez

[DEMO] *A System for Diacritizing Four Varieties of Arabic*
Hamdy Mubarak, Ahmed Abdelali, Kareem Darwish, Mohamed Eldesouki, Younes Samih and Hassan Sajjad

[DEMO] *What's Wrong with Hebrew NLP? And How to Make it Right*
Reut Tsarfaty, Shoval Sadde, Stav Klein and Amit Seker

[DEMO] *INMT: Interactive Neural Machine Translation Prediction*
Sebastin Santy, Sandipan Dandapat, Monojit Choudhury and Kalika Bali

15:00–15:30 *Coffee Break*

15:30–16:18 *Session 3*

### Session 3A: Machine Learning II

15:30–15:42 *Are We Modeling the Task or the Annotator? An Investigation of Annotator Bias in Natural Language Understanding Datasets*
Mor Geva, Yoav Goldberg and Jonathan Berant

15:42–15:54 *Robust Text Classifier on Test-Time Budgets*
Md Rizwan Parvez, Tolga Bolukbasi, Kai-Wei Chang and Venkatesh Saligrama

15:54–16:06 *Commonsense Knowledge Mining from Pretrained Models*
Joe Davison, Joshua Feldman and Alexander Rush

16:06–16:18 *RNN Architecture Learning with Sparse Regularization*
Jesse Dodge, Roy Schwartz, Hao Peng and Noah A. Smith

### Session 3B: Semantics

15:30–15:42 *Analytical Methods for Interpretable Ultradense Word Embeddings*
Philipp Dufter and Hinrich Schütze

15:42–15:54 *Investigating Meta-Learning Algorithms for Low-Resource Natural Language Understanding Tasks*
Zi-Yi Dou, Keyi Yu and Antonios Anastasopoulos

15:54–16:06 *Retrofitting Contextualized Word Embeddings with Paraphrases*
Weijia Shi, Muhao Chen, Pei Zhou and Kai-Wei Chang

16:06–16:18 *Incorporating Contextual and Syntactic Structures Improves Semantic Similarity Modeling*
Linqing Liu, Wei Yang, Jinfeng Rao, Raphael Tang and Jimmy Lin

**Session 3C: Discourse, Summarization, and Generation**

15:30–15:42 *Neural Linguistic Steganography*
Zachary Ziegler, Yuntian Deng and Alexander Rush

15:42–15:54 *The Feasibility of Embedding Based Automatic Evaluation for Single Document Summarization*
Simeng Sun and Ani Nenkova

15:54–16:06 *Attention Optimization for Abstractive Document Summarization*
Min Gui, Junfeng Tian, Rui Wang and Zhenglu Yang

16:06–16:18 *Rewarding Coreference Resolvers for Being Consistent with World Knowledge*
Rahul Aralikatte, Heather Lent, Ana Valeria Gonzalez, Daniel Herschcovich, Chen Qiu, Anders Sandholm, Michael Ringaard and Anders Søgaard

**Session 3D: Text Mining and NLP Applications I**

15:30–15:42 *An Empirical Study of Incorporating Pseudo Data into Grammatical Error Correction*
Shun Kiyono, Jun Suzuki, Masato Mita, Tomoya Mizumoto and Kentaro Inui

15:42–15:54 *A Multilingual Topic Model for Learning Weighted Topic Links Across Corpora with Low Comparability*
Weiwei Yang, Jordan Boyd-Graber and Philip Resnik

15:54–16:06 *Measure Country-Level Socio-Economic Indicators with Streaming News: An Empirical Study*
Bonan Min and Xiaoxi Zhao

16:06–16:18 *Towards Extracting Medical Family History from Natural Language Interactions: A New Dataset and Baselines*
Mahmoud Azab, Stephane Dadian, Vivi Nastase, Larry An and Rada Mihalcea

**Poster and Demo Session 3: Dialog and Interactive Systems, Machine Translation and Multilinuality, Phonology, Morphology, and Word Segmentation, Speech, Vision, Robotics, Multimodal and Grounding, Tagging, Chunking, Syntax and Parsing**

*Multi-task Learning for Natural Language Generation in Task-Oriented Dialogue*
Chenguang Zhu, Michael Zeng and Xuedong Huang

*Dirichlet Latent Variable Hierarchical Recurrent Encoder-Decoder in Dialogue Generation*
Min Zeng, Yisen Wang and Yuan Luo

*Semi-Supervised Bootstrapping of Dialogue State Trackers for Task-Oriented Modelling*
Bo-Hsiang Tseng, Marek Rei, Paweł Budzianowski, Richard Turner, Bill Byrne and Anna Korhonen

*A Progressive Model to Enable Continual Learning for Semantic Slot Filling*
Yilin Shen, Xiangyu Zeng and Hongxia Jin

*CASA-NLU: Context-Aware Self-Attentive Natural Language Understanding for Task-Oriented Chatbots*
Arshit Gupta, Peng Zhang, Garima Lalwani and Mona Diab

*Sampling Matters! An Empirical Study of Negative Sampling Strategies for Learning of Matching Models in Retrieval-based Dialogue Systems*
Jia Li, Chongyang Tao, wei wu, Yansong Feng, Dongyan Zhao and Rui Yan

*Zero-shot Cross-lingual Dialogue Systems with Transferable Latent Variables*
Zihan Liu, Jamin Shin, Yan Xu, Genta Indra Winata, Peng Xu, Andrea Madotto and Pascale Fung

*Modeling Multi-Action Policy for Task-Oriented Dialogues*
Lei Shu, Hu Xu, Bing Liu and Piero Molino

*An Evaluation Dataset for Intent Classification and Out-of-Scope Prediction*
Stefan Larson, Anish Mahendran, Joseph J. Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael A. Laurenzano, Lingjia Tang and Jason mars

*Automatically Learning Data Augmentation Policies for Dialogue Tasks*
Tong Niu and Mohit Bansal

*uniblock: Scoring and Filtering Corpus with Unicode Block Information*
Yingbo Gao, Weiyue Wang and Hermann Ney

*Multilingual word translation using auxiliary languages*
Hagai Taitelbaum, Gal Chechik and Jacob Goldberger

*Towards Better Modeling Hierarchical Structure for Self-Attention with Ordered Neurons*
Jie Hao, Xing Wang, Shuming Shi, Jinfeng Zhang and Zhaopeng Tu

*Vecalign: Improved Sentence Alignment in Linear Time and Space*
Brian Thompson and Philipp Koehn

*Simpler and Faster Learning of Adaptive Policies for Simultaneous Translation*
Baigong Zheng, Renjie Zheng, Mingbo Ma and Liang Huang

*Adversarial Learning with Contextual Embeddings for Zero-resource Cross-lingual Classification and NER*
Phillip Keung, yichao lu and Vikas Bhardwaj

*Recurrent Positional Embedding for Neural Machine Translation*
Kehai Chen, Rui Wang, Masao Utiyama and Eiichiro Sumita

*Machine Translation for Machines: the Sentiment Classification Use Case*
amirhossein tebbifakhr, Luisa Bentivogli, Matteo Negri and Marco Turchi

*Investigating the Effectiveness of BPE: The Power of Shorter Sequences*
Matthias Gallé

*HABLex: Human Annotated Bilingual Lexicons for Experiments in Machine Translation*
Brian Thompson, Rebecca Knowles, Xuan Zhang, Huda Khayrallah, Kevin Duh and Philipp Koehn

*Handling Syntactic Divergence in Low-resource Machine Translation*
Chunting Zhou, Xuezhe Ma, Junjie Hu and Graham Neubig

*Speculative Beam Search for Simultaneous Translation*
Renjie Zheng, Mingbo Ma, Baigong Zheng and Liang Huang

*Self-Attention with Structural Position Representations*
Xing Wang, Zhaopeng Tu, Longyue Wang and Shuming Shi

*Exploiting Multilingualism through Multistage Fine-Tuning for Low-Resource Neural Machine Translation*
Raj Dabre, Atsushi Fujita and Chenhui Chu

*Unsupervised Domain Adaptation for Neural Machine Translation with Domain-Aware Feature Embeddings*
Zi-Yi Dou, Junjie Hu, Antonios Anastasopoulos and Graham Neubig

*A Regularization-based Framework for Bilingual Grammar Induction*
Yong Jiang, Wenjuan Han and Kewei Tu

*Encoders Help You Disambiguate Word Senses in Neural Machine Translation*
Gongbo Tang, Rico Sennrich and Joakim Nivre

*Korean Morphological Analysis with Tied Sequence-to-Sequence Multi-Task Model*
Hyun-Je Song and Seong-Bae Park

*Efficient Convolutional Neural Networks for Diacritic Restoration*
Sawsan Alqahtani, Ajay Mishra and Mona Diab

*Improving Generative Visual Dialog by Answering Diverse Questions*
Vishvak Murahari, Prithvijit Chattopadhyay, Dhruv Batra, Devi Parikh and Abhishek Das

*Cross-lingual Transfer Learning with Data Selection for Large-Scale Spoken Language Understanding*
Quynh Do and Judith Gaspers

*Multi-Head Attention with Diversity for Learning Grounded Multilingual Multimodal Representations*
Po-Yao Huang, Xiaojun Chang and Alexander Hauptmann

*Decoupled Box Proposal and Featurization with Ultrafine-Grained Semantic Labels Improve Image Captioning and Visual Question Answering*
Soravit Changpinyo, Bo Pang, Piyush Sharma and Radu Soricut

*REO-Relevance, Extraness, Omission: A Fine-grained Evaluation for Image Captioning*
Ming Jiang, Junjie Hu, Qiuyuan Huang, Lei Zhang, Jana Diesner and Jianfeng Gao

*WSLLN:Weakly Supervised Natural Language Localization Networks*
Mingfei Gao, Larry Davis, Richard Socher and Caiming Xiong

*Grounding learning of modifier dynamics: An application to color naming*
Xudong Han, Philip Schulz and Trevor Cohn

*Robust Navigation with Language Pretraining and Stochastic Sampling*
Xiujun Li, Chunyuan Li, Qiaolin Xia, Yonatan Bisk, Asli Celikyilmaz, Jianfeng Gao, Noah A. Smith and Yejin Choi

*Towards Making a Dependency Parser See*
Michalina Strzyz, David Vilares and Carlos Gómez-Rodríguez

*Unsupervised Labeled Parsing with Deep Inside-Outside Recursive Autoencoders*
Andrew Drozdov, Patrick Verga, Yi-Pei Chen, Mohit Iyyer and Andrew McCallum

*Dependency Parsing for Spoken Dialog Systems*
Sam Davidson, Dian Yu and Zhou Yu

*Span-based Hierarchical Semantic Parsing for Task-Oriented Dialog*
Panupong Pasupat, Sonal Gupta, Karishma Mandyam, Rushin Shah, Mike Lewis and Luke Zettlemoyer

16:18–16:30 *Mini-Break*

16:30–18:00 *Session 4*

**Session 4A: Neural Machine Translation**

16:30–16:48 *Enhancing Context Modeling with a Query-Guided Capsule Network for Document-level Translation*
Zhengxin Yang, Jinchao Zhang, Fandong Meng, Shuhao Gu, Yang Feng and Jie Zhou

16:48–17:06 *Simple, Scalable Adaptation for Neural Machine Translation*
Ankur Bapna and Orhan Firat

17:06–17:24 *Controlling Text Complexity in Neural Machine Translation*
Sweta Agrawal and Marine Carpuat

17:24–17:42 *Investigating Multilingual NMT Representations at Scale*
Sneha Kudugunta, Ankur Bapna, Isaac Caswell and Orhan Firat

17:42–18:00 *Hierarchical Modeling of Global Context for Document-Level Neural Machine Translation*
Xin Tan, Longyin Zhang, Deyi Xiong and Guodong Zhou

**Session 4B: Question Answering I**

16:30–16:48 *Cross-Lingual Machine Reading Comprehension*
Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Shijin Wang and Guoping Hu

16:48–17:06 *A Multi-Type Multi-Span Network for Reading Comprehension that Requires Discrete Reasoning*
Minghao Hu, Yuxing Peng, Zhen Huang and Dongsheng Li

17:06–17:24 *Neural Duplicate Question Detection without Labeled Training Data*
Andreas Rücklé, Nafise Sadat Moosavi and Iryna Gurevych

17:24–17:42 *Asking Clarification Questions in Knowledge-Based Question Answering*
Jingjing Xu, Yuechen Wang, Duyu Tang, Nan Duan, Pengcheng Yang, Qi Zeng, Ming Zhou and Xu SUN

17:42–18:00 *Multi-View Domain Adapted Sentence Embeddings for Low-Resource Unsupervised Duplicate Question Detection*
Nina Poerner and Hinrich Schütze

### Session 4C: Social Media and Computational Social Science

16:30–16:48 *Multi-label Categorization of Accounts of Sexism using a Neural Framework*
Pulkit Parikh, Harika Abburi, Pinkesh Badjatiya, Radhika Krishnan, Niyati Chhaya, Manish Gupta and Vasudeva Varma

16:48–17:06 *The Trumpiest Trump? Identifying a Subject's Most Characteristic Tweets*
Charuta Pethe and Steve Skiena

17:06–17:24 *Finding Microaggressions in the Wild: A Case for Locating Elusive Phenomena in Social Media Posts*
Luke Breitfeller, Emily Ahn, David Jurgens and Yulia Tsvetkov

17:24–17:42 *Reinforced Product Metadata Selection for Helpfulness Assessment of Customer Reviews*
Miao Fan, Chao Feng, Mingming Sun and Ping Li

17:42–18:00 *Learning Invariant Representations of Social Media Users*
Nicholas Andrews and Marcus Bishop

### Session 4D: Text Mining and NLP Applications II

16:30–16:48 *(Male, Bachelor) and (Female, Ph.D) have different connotations: Parallelly Annotated Stylistic Language Dataset with Multiple Personas*
Dongyeop Kang, Varun Gangal and Eduard Hovy

16:48–17:06 *Movie Plot Analysis via Turning Point Identification*
Pinelopi Papalampidi, Frank Keller and Mirella Lapata

17:06–17:24 *Latent Suicide Risk Detection on Microblog via Suicide-Oriented Word Embeddings and Layered Attention*
Lei Cao, Huijun Zhang, Ling Feng, Zihan Wei, Xin Wang, Ningyun Li and Xiaohao He

17:24–17:42 *Deep Ordinal Regression for Pledge Specificity Prediction*
Shivashankar Subramanian, Trevor Cohn and Timothy Baldwin

17:42–18:00 [TACL] *Enabling Robust Grammatical Error Correction in New Domains: Datasets, Metrics, and Analyses*
Courtney Napoles, Maria Nadejde and Joel Tetreault

**Poster and Demo Session 4: Dialog and Interactive Systems, Speech, Vision, Robotics, Multimodal and Grounding**

*Data-Efficient Goal-Oriented Conversation with Dialogue Knowledge Transfer Networks*
Igor Shalyminov, Sungjin Lee, Arash Eshghi and Oliver Lemon

*Multi-Granularity Representations of Dialog*
Shikib Mehri and Maxine Eskenazi

*Are You for Real? Detecting Identity Fraud via Dialogue Interactions*
Weikang Wang, Jiajun Zhang, Qian Li, Chengqing Zong and Zhifei Li

*Hierarchy Response Learning for Neural Conversation Generation*
Bo Zhang and Xiaoming Zhang

*Knowledge Aware Conversation Generation with Explainable Reasoning over Augmented Graphs*
zhibin liu, Zheng-Yu Niu, Hua Wu and Haifeng Wang

*Adaptive Parameterization for Neural Dialogue Generation*
Hengyi Cai, Hongshen Chen, Cheng Zhang, Yonghao Song, Xiaofang Zhao and Dawei Yin

*Towards Knowledge-Based Recommender Dialog System*
Qibin Chen, Junyang Lin, Yichang Zhang, Ming Ding, Yukuo Cen, Hongxia Yang and Jie Tang

*Structuring Latent Spaces for Stylized Response Generation*
Xiang Gao, Yizhe Zhang, Sungjin Lee, Michel Galley, Chris Brockett, Jianfeng Gao and Bill Dolan

*Improving Open-Domain Dialogue Systems via Multi-Turn Incomplete Utterance Restoration*
Zhufeng Pan, Kun Bai, Yan Wang, Lianqiang Zhou and Xiaojiang Liu

*Unsupervised Context Rewriting for Open Domain Conversation*
Kun Zhou, Kai Zhang, Yu Wu, Shujie Liu and Jingsong Yu

*Dually Interactive Matching Network for Personalized Response Selection in Retrieval-Based Chatbots*
Jia-Chen Gu, Zhen-Hua Ling, Xiaodan Zhu and Quan Liu

*DyKgChat: Benchmarking Dialogue Generation Grounding on Dynamic Knowledge Graphs*
Yi-Lin Tuan, Yun-Nung Chen and Hung-yi Lee

*Retrieval-guided Dialogue Response Generation via a Matching-to-Generation Framework*
Deng Cai, Yan Wang, Wei Bi, Zhaopeng Tu, Xiaojiang Liu and Shuming Shi

*Scalable and Accurate Dialogue State Tracking via Hierarchical Sequence Generation*
Liliang Ren, Jianmo Ni and Julian McAuley

*Low-Resource Response Generation with Template Prior*
Ze Yang, wei wu, Jian Yang, Can Xu and zhoujun li

*A Discrete CVAE for Response Generation on Short-Text Conversation*
Jun Gao, Wei Bi, Xiaojiang Liu, Junhui Li, Guodong Zhou and Shuming Shi

*Who Is Speaking to Whom? Learning to Identify Utterance Addressee in Multi-Party Conversations*
Ran Le, Wenpeng Hu, Mingyue Shang, Zhenjun You, Lidong Bing, Dongyan Zhao and Rui Yan

*A Semi-Supervised Stable Variational Network for Promoting Replier-Consistency in Dialogue Generation*
Jinxin Chang, Ruifang He, Longbiao Wang, Xiangyu Zhao, Ting Yang and Ruifang Wang

*Modeling Personalization in Continuous Space for Response Generation via Augmented Wasserstein Autoencoders*
Zhangming Chan, Juntao Li, Xiaopeng Yang, Xiuying Chen, Wenpeng Hu, Dongyan Zhao and Rui Yan

*Variational Hierarchical User-based Conversation Model*
JinYeong Bak and Alice Oh

*Recommendation as a Communication Game: Self-Supervised Bot-Play for Goal-oriented Dialogue*
Dongyeop Kang, Anusha Balakrishnan, Pararth Shah, Paul Crook, Y-Lan Boureau and Jason Weston

*CoSQL: A Conversational Text-to-SQL Challenge Towards Cross-Domain Natural Language Interfaces to Databases*
Tao Yu, Rui Zhang, Heyang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, Youxuan Jiang, Michihiro Yasunaga, Sungrok Shim, Tao Chen, Alexander Fabbri, Zifan Li, Luyao Chen, Yuwen Zhang, Shreya Dixit, Vincent Zhang, Caiming Xiong, Richard Socher, Walter Lasecki and Dragomir Radev

*A Practical Dialogue-Act-Driven Conversation Model for Multi-Turn Response Selection*
Harshit Kumar, Arvind Agarwal and Sachindra Joshi

*How to Build User Simulators to Train RL-based Dialog Systems*
Weiyan Shi, Kun Qian, Xuewei Wang and Zhou Yu

[TACL] *Graph Convolutional Network with Sequential Attention for Goal-Oriented Dialogue Systems*
Suman Banerjee and Mitesh M Khapra

*Low-Rank HOCA: Efficient High-Order Cross-Modal Attention for Video Captioning*
Tao Jin, Siyu Huang, Yingming Li and Zhongfei Zhang

*Image Captioning with Very Scarce Supervised Data: Adversarial Semi-Supervised Learning Approach*
Dong-Jin Kim, Jinsoo Choi, Tae-Hyun Oh and In So Kweon

*Dual Attention Networks for Visual Reference Resolution in Visual Dialog*
Gi-Cheon Kang, Jaeseo Lim and Byoung-Tak Zhang

*Unsupervised Discovery of Multimodal Links in Multi-image, Multi-sentence Documents*
Jack Hessel, Lillian Lee and David Mimno

*UR-FUNNY: A Multimodal Language Dataset for Understanding Humor*
Md Kamrul Hasan, Wasifur Rahman, AmirAli Bagher Zadeh, Jianyuan Zhong, Md Iftekhar Tanveer, Louis-Philippe Morency and Mohammed (Ehsan) Hoque

*Partners in Crime: Multi-view Sequential Inference for Movie Understanding*
Nikos Papasarantopoulos, Lea Frermann, Mirella Lapata and Shay B. Cohen

*Guiding the Flowing of Semantics: Interpretable Video Captioning via POS Tag*
Xinyu Xiao, Lingfeng Wang, Bin Fan, Shinming Xiang and Chunhong Pan

*A Stack-Propagation Framework with Token-Level Intent Detection for Spoken Language Understanding*
Libo Qin, Wanxiang Che, Yangming Li, Haoyang Wen and Ting Liu

*Talk2Car: Taking Control of Your Self-Driving Car*
Thierry Deruyttere, Simon Vandenhende, Dusan Grujicic, Luc Van Gool and Marie-Francine Moens

*Fact-Checking Meets Fauxtography: Verifying Claims About Images*
Dimitrina Zlatkova, Preslav Nakov and Ivan Koychev

*Video Dialog via Progressive Inference and Cross-Transformer*
Weike Jin, Zhou Zhao, Mao Gu, Jun Xiao, Furu Wei and Yueting Zhuang

*Executing Instructions in Situated Collaborative Interactions*
Alane Suhr, Claudia Yan, Jack Schluger, Stanley Yu, Hadi Khader, Marwa Mouallem, Iris Zhang and Yoav Artzi

*Fusion of Detected Objects in Text for Visual Question Answering*
Chris Alberti, Jeffrey Ling, Michael Collins and David Reitter

*TIGEr: Text-to-Image Grounding for Image Caption Evaluation*
Ming Jiang, Qiuyuan Huang, Lei Zhang, Xin Wang, Pengchuan Zhang, Zhe Gan, Jana Diesner and Jianfeng Gao

[DEMO] *Chameleon: A Language Model Adaptation Toolkit for Automatic Speech Recognition of Conversational Speech*
Yuanfeng Song, Di Jiang, Weiwei Zhao, Qian Xu, Raymond Chi-Wing Wong and Qiang Yang

[DEMO] *PyOpenDial: A Python-based Domain-Independent Toolkit for Developing Spoken Dialogue Systems with Probabilistic Rules*
Youngsoo Jang, Jongmin Lee, Jaeyoung Park, Kyeng-Hun Lee, Pierre Lison and Kee-Eung Kim

[DEMO] *PolyResponse: A Rank-based Approach to Task-Oriented Dialogue with Application in Restaurant Search and Booking*
Matthew Henderson, Ivan Vulić, Iñigo Casanueva, Paweł Budzianowski, Daniela Gerz, Sam Coope, Georgios Spithourakis, Tsung-Hsien Wen, Nikola Mrkšić and Pei-Hao Su

**Tuesday, November 5, 2019 (continued)**

[DEMO] *LIDA: Lightweight Interactive Dialogue Annotator*
Edward Collins, Nikolai Rozanov and Bingbing Zhang

[DEMO] *EGG: a toolkit for research on Emergence of lanGuage in Games*
Eugene Kharitonov, Rahma Chaabouni, Diane Bouchacourt and Marco Baroni

[DEMO] *Entity resolution for noisy ASR transcripts*
Arushi Raghuvanshi, Vijay Ramakrishnan, Varsha Embar, Lucien Carroll and Karthik Raghunathan

[DEMO] *Gunrock: A Social Bot for Complex and Engaging Long Conversations*
Dian Yu, Michelle Cohn, Yi Mang Yang, Chun Yen Chen, Weiming Wen, Jiaping Zhang, Mingyang Zhou, Kevin Jesse, Austin Chau, Antara Bhowmick, Shreenath Iyer, Giritheja Sreenivasulu, Sam Davidson, Ashwin Bhandare and Zhou Yu

[DEMO] *HARE: a Flexible Highlighting Annotator for Ranking and Exploration*
Denis Newman-Griffis and Eric Fosler-Lussier

**Wednesday, November 6, 2019**

09:00–10:00 *Keynote II: Meeyoung Cha*

10:00–10:30 *Coffee Break*

10:30–12:00 *Session 5*

**Session 5A: Machine Learning III**

10:30–10:48 *Universal Adversarial Triggers for Attacking and Analyzing NLP*
Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner and Sameer Singh

10:48–11:06 *To Annotate or Not? Predicting Performance Drop under Domain Shift*
Hady Elsahar and Matthias Gallé

11:06–11:24 *Adaptively Sparse Transformers*
Gonçalo M. Correia, Vlad Niculae and André F. T. Martins

11:24–11:42 *Show Your Work: Improved Reporting of Experimental Results*
Jesse Dodge, Suchin Gururangan, Dallas Card, Roy Schwartz and Noah A. Smith

11:42–12:00 *A Deep Factorization of Style and Structure in Fonts*
Akshay Srivatsan, Jonathan Barron, Dan Klein and Taylor Berg-Kirkpatrick

### Session 5B: Lexical Semantics II

10:30–10:48 *Cross-lingual Semantic Specialization via Lexical Relation Induction*
Edoardo Maria Ponti, Ivan Vulić, Goran Glavaš, Roi Reichart and Anna Korhonen

10:48–11:06 *Modelling the interplay of metaphor and emotion through multitask learning*
Verna Dankers, Marek Rei, Martha Lewis and Ekaterina Shutova

11:06–11:24 *How well do NLI models capture verb veridicality?*
Alexis Ross and Ellie Pavlick

11:24–11:42 *Modeling Color Terminology Across Thousands of Languages*
Arya D. McCarthy, Winston Wu, Aaron Mueller, William Watson and David Yarowsky

11:42–12:00 *Negative Focus Detection via Contextual Attention Mechanism*
Longxiang Shen, Bowei Zou, Yu Hong, Guodong Zhou, Qiaoming Zhu and AiTi Aw

### Session 5C: Discourse and Pragmatics

10:30–10:48 *A Unified Neural Coherence Model*
Han Cheol Moon, Tasnim Mohiuddin, Shafiq Joty and Chi Xu

10:48–11:06 *Topic-Guided Coherence Modeling for Sentence Ordering by Preserving Global and Local Information*
Byungkook Oh, Seungmin Seo, Cheolheon Shin, Eunju Jo and Kyong-Ho Lee

11:06–11:24 *Neural Generative Rhetorical Structure Parsing*
Amandla Mabona, Laura Rimell, Stephen Clark and Andreas Vlachos

11:24–11:42 *Weak Supervision for Learning Discourse Structure*
Sonia Badene, Kate Thompson, Jean-Pierre Lorré and Nicholas Asher

11:42–12:00 *Predicting Discourse Structure using Distant Supervision from Sentiment*
Patrick Huber and Giuseppe Carenini

### Session 5D: Text Mining and NLP Applications III

10:30–10:48 *The Myth of Double-Blind Review Revisited: ACL vs. EMNLP*
Cornelia Caragea, Ana Uban and Liviu P. Dinu

10:48–11:06 *Uncover Sexual Harassment Patterns from Personal Stories by Joint Key Element Extraction and Categorization*
Yingchi Liu, Quanzhi Li, Marika Cifor, Xiaozhong Liu, Qiong Zhang and Luo Si

11:06–11:24 *Identifying Predictive Causal Factors from News Streams*
Ananth Balashankar, Sunandan Chakraborty, Samuel Fraiberger and Lakshminarayanan Subramanian

11:24–11:42 *Training Data Augmentation for Detecting Adverse Drug Reactions in User-Generated Content*
Sepideh Mesbah, Jie Yang, Robert-Jan Sips, Manuel Valle Torre, Christoph Lofi, Alessandro Bozzon and Geert-Jan Houben

11:42–12:00 *Deep Reinforcement Learning-based Text Anonymization against Private-Attribute Inference*
Ahmadreza Mosallanezhad, Ghazaleh Beigi and Huan Liu

### Poster and Demo Session 5: Question Answering, Textual Inference and Other Areas of Semantics

*Tree-structured Decoding for Solving Math Word Problems*
Qianying Liu, Wenyv Guan, Sujian Li and Daisuke Kawahara

*PullNet: Open Domain Question Answering with Iterative Retrieval on Knowledge Bases and Text*
Haitian Sun, Tania Bedrax-Weiss and William Cohen

*Cosmos QA: Machine Reading Comprehension with Contextual Commonsense Reasoning*
Lifu Huang, Ronan Le Bras, Chandra Bhagavatula and Yejin Choi

*Finding Generalizable Evidence by Learning to Convince Q&A Models*
Ethan Perez, Siddharth Karamcheti, Rob Fergus, Jason Weston, Douwe Kiela and Kyunghyun Cho

*Ranking and Sampling in Open-Domain Question Answering*
Yanfu Xu, Zheng Lin, Yuanxin Liu, Rui Liu, Weiping Wang and Dan Meng

*A Non-commutative Bilinear Model for Answering Path Queries in Knowledge Graphs*
Katsuhiko Hayashi and Masashi Shimbo

*Generating Questions for Knowledge Bases via Incorporating Diversified Contexts and Answer-Aware Loss*
Cao Liu, Kang Liu, Shizhu He, Zaiqing Nie and Jun Zhao

*Multi-Task Learning for Conversational Question Answering over a Large-Scale Knowledge Base*
Tao Shen, Xiubo Geng, Tao QIN, Daya Guo, Duyu Tang, Nan Duan, Guodong Long and Daxin Jiang

*BiPaR: A Bilingual Parallel Dataset for Multilingual and Cross-lingual Reading Comprehension on Novels*
Yimin Jing, Deyi Xiong and Zhen Yan

*Language Models as Knowledge Bases?*
Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu and Alexander Miller

*NumNet: Machine Reading Comprehension with Numerical Reasoning*
Qiu Ran, Yankai Lin, Peng Li, Jie Zhou and Zhiyuan Liu

*Unicoder: A Universal Language Encoder by Pre-training with Multiple Cross-lingual Tasks*
Haoyang Huang, Yaobo Liang, Nan Duan, Ming Gong, Linjun Shou, Daxin Jiang and Ming Zhou

*Addressing Semantic Drift in Question Generation for Semi-Supervised Question Answering*
Shiyue Zhang and Mohit Bansal

*Adversarial Domain Adaptation for Machine Reading Comprehension*
Huazheng Wang, Zhe Gan, Xiaodong Liu, Jingjing Liu, Jianfeng Gao and Hongning Wang

*Incorporating External Knowledge into Machine Reading for Generative Question Answering*
Bin Bi, Chen Wu, Ming Yan, Wei Wang, Jiangnan Xia and Chenliang Li

*Answering questions by learning to rank - Learning to rank by answering questions*
George Sebastian Pirtoaca, Traian Rebedea and Stefan Ruseti

*Discourse-Aware Semantic Self-Attention for Narrative Reading Comprehension*
Todor Mihaylov and Anette Frank

*Revealing the Importance of Semantic Retrieval for Machine Reading at Scale*
Yixin Nie, Songhe Wang and Mohit Bansal

*PubMedQA: A Dataset for Biomedical Research Question Answering*
Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen and Xinghua Lu

*Quick and (not so) Dirty: Unsupervised Selection of Justification Sentences for Multi-hop Question Answering*
Vikas Yadav, Steven Bethard and Mihai Surdeanu

*Answering Complex Open-domain Questions Through Iterative Query Generation*
Peng Qi, Xiaowen Lin, Leo Mehr, Zijian Wang and Christopher D. Manning

*NL2pSQL: Generating Pseudo-SQL Queries from Under-Specified Natural Language Questions*
Fuxiang Chen, Seung-won Hwang, Jaegul Choo, Jung-Woo Ha and Sunghun Kim

*Leveraging Frequent Query Substructures to Generate Formal Queries for Complex Question Answering*
Jiwei Ding, Wei Hu, Qixin Xu and Yuzhong Qu

*Incorporating Graph Attention Mechanism into Knowledge Graph Reasoning Based on Deep Reinforcement Learning*
Heng Wang, Shuangyin Li, Rong Pan and Mingzhi Mao

*Learning to Update Knowledge Graphs by Reading News*
Jizhi Tang, Yansong Feng and Dongyan Zhao

*DIVINE: A Generative Adversarial Imitation Learning Framework for Knowledge Graph Reasoning*
Ruiping Li and Xiang Cheng

*Original Semantics-Oriented Attention and Deep Fusion Network for Sentence Matching*
Mingtong Liu, Yujie Zhang, Jinan Xu and Yufeng Chen

*Representation Learning with Ordered Relation Paths for Knowledge Graph Completion*
Yao Zhu, Hongzhi Liu, Zhonghai Wu, Yang Song and Tao Zhang

*Collaborative Policy Learning for Open Knowledge Graph Reasoning*
Cong Fu, Tong Chen, Meng Qu, Woojeong Jin and Xiang Ren

*Modeling Event Background for If-Then Commonsense Reasoning Using Context-aware Variational Autoencoder*
Li Du, Xiao Ding, Ting Liu and Zhongyang Li

*Asynchronous Deep Interaction Network for Natural Language Inference*
Di Liang, Fubao Zhang, Qi Zhang and Xuanjing Huang

*Keep Calm and Switch On! Preserving Sentiment and Fluency in Semantic Text Exchange*
Steven Y. Feng, Aaron W. Li and Jesse Hoey

*Query-focused Scenario Construction*
Su Wang, Greg Durrett and Katrin Erk

*Semi-supervised Entity Alignment via Joint Knowledge Embedding Model and Cross-graph Model*
Chengjiang Li, Yixin Cao, Lei Hou, Jiaxin Shi, Juanzi Li and Tat-Seng Chua

[DEMO] *EUSP: An Easy-to-Use Semantic Parsing PlatForm*
Bo An, Chen Bo, Xianpei Han and Le Sun

[DEMO] *ParaQG: A System for Generating Questions and Answers from Paragraphs*
vishwajeet kumar, Sivaanandh Muneeswaran, Ganesh Ramakrishnan and Yuan-Fang Li

[DEMO] *CFO: A Framework for Building Production NLP Systems*
Rishav Chakravarti, Cezar Pendus, Andrzej Sakrajda, Anthony Ferritto, Lin Pan, Michael Glass, Vittorio Castelli, J William Murdock, Radu Florian, Salim Roukos and Avi Sil

[DEMO] *ABSApp: A Portable Weakly-Supervised Aspect-Based Sentiment Extraction System*
Oren Pereg, Daniel Korat, Moshe Wasserblat, Jonathan Mamou and Ido Dagan

[DEMO] *Memory Grounded Conversational Reasoning*
Seungwhan Moon, Pararth Shah, Rajen Subba and Anuj Kumar

12:00–13:30 *Lunch*

13:30–15:00 *Session 6*

### Session 6A: Tagging, Chunking, Syntax and Parsing

13:30–13:48 *Designing and Interpreting Probes with Control Tasks*
John Hewitt and Percy Liang

13:48–14:06 *Specializing Word Embeddings (for Parsing) by Information Bottleneck*
Xiang Lisa Li and Jason Eisner

14:06–14:24 *Deep Contextualized Word Embeddings in Transition-Based and Graph-Based Dependency Parsing - A Tale of Two Parsers Revisited*
Artur Kulmizev, Miryam de Lhoneux, Johannes Gontrum, Elena Fano and Joakim Nivre

14:24–14:42 *Semantic graph parsing with recurrent neural network DAG grammars*
Federico Fancellu, Sorcha Gilroy, Adam Lopez and Mirella Lapata

14:42–15:00 *75 Languages, 1 Model: Parsing Universal Dependencies Universally*
Dan Kondratyuk and Milan Straka

### Session 6B: Question Answering II

13:30–13:48 *Interactive Language Learning by Question Answering*
Xingdi Yuan, Marc-Alexandre Côté, Jie Fu, Zhouhan Lin, Chris Pal, Yoshua Bengio and Adam Trischler

13:48–14:06 *What's Missing: A Knowledge Gap Guided Approach for Multi-hop Question Answering*
Tushar Khot, Ashish Sabharwal and Peter Clark

14:06–14:24 *KagNet: Knowledge-Aware Graph Networks for Commonsense Reasoning*
Bill Yuchen Lin, Xinyue Chen, Jamin Chen and Xiang Ren

14:24–14:42 *Learning with Limited Data for Multilingual Reading Comprehension*
Kyungjae Lee, Sunghyun Park, Hojae Han, Jinyoung Yeo, Seung-won Hwang and Juho Lee

14:42–15:00 *A Discrete Hard EM Approach for Weakly Supervised Question Answering*
Sewon Min, Danqi Chen, Hannaneh Hajishirzi and Luke Zettlemoyer

### Session 6C: Linguistic Theories, Cognitive Modeling and Psycholinguistics

13:30–13:48 *Is the Red Square Big? MALeViC: Modeling Adjectives Leveraging Visual Contexts*
Sandro Pezzelle and Raquel Fernández

13:48–14:06 *Investigating BERT's Knowledge of Language: Five Analysis Methods with NPIs*
Alex Warstadt, Yu Cao, Ioana Grosu, Wei Peng, Hagen Blix, Yining Nie, Anna Alsop, Shikha Bordia, Haokun Liu, Alicia Parrish, Sheng-Fu Wang, Jason Phang, Anhad Mohananey, Phu Mon Htut, Paloma Jeretic and Samuel R. Bowman

14:06–14:24 *Representation of Constituents in Neural Language Models: Coordination Phrase as a Case Study*
Aixiu AN, Peng Qian, Ethan Wilcox and Roger Levy

14:24–14:42 *Towards Zero-shot Language Modeling*
Edoardo Maria Ponti, Ivan Vulić, Ryan Cotterell, Roi Reichart and Anna Korhonen

14:42–15:00 [TACL] *Neural Network Acceptability Judgments*
Alex Warstadt, Amanpreet Singh and Samuel R. Bowman

### Session 6D: Sentiment Analysis and Argument Mining II

13:30–13:48 *What Gets Echoed? Understanding the "Pointers" in Explanations of Persuasive Arguments*
David Atkinson, Kumar Bhargav Srinivasan and Chenhao Tan

13:48–14:06 *Modeling Frames in Argumentation*
Yamen Ajjour, Milad Alshomary, Henning Wachsmuth and Benno Stein

14:06–14:24 *AMPERSAND: Argument Mining for PERSuAsive oNline Discussions*
Tuhin Chakrabarty, Christopher Hidey, Smaranda Muresan, Kathy McKeown and Alyssa Hwang

14:24–14:42 *Evaluating adversarial attacks against multiple fact verification systems*
James Thorne, Andreas Vlachos, Christos Christodoulopoulos and Arpit Mittal

14:42–15:00 *Nonsense!: Quality Control via Two-Step Reason Selection for Annotating Local Acceptability and Related Attributes in News Editorials*
Wonsuk Yang, seungwon yoon, Ada Carpenter and Jong Park

**Poster and Demo Session 6: Discourse and Pragmatics, Summarization and Generation**

*Evaluating Pronominal Anaphora in Machine Translation: An Evaluation Measure and a Test Suite*
Prathyusha Jwalapuram, Shafiq Joty, Irina Temnikova and Preslav Nakov

*A Regularization Approach for Incorporating Event Knowledge and Coreference Relations into Neural Discourse Parsing*
Zeyu Dai and Ruihong Huang

*Weakly Supervised Multilingual Causality Extraction from Wikipedia*
Chikara Hashimoto

*Attribute-aware Sequence Network for Review Summarization*
Junjie Li, Xuepeng Wang, Dawei Yin and Chengqing Zong

*Extractive Summarization of Long Documents by Combining Global and Local Context*
Wen Xiao and Giuseppe Carenini

*Enhancing Neural Data-To-Text Generation Models with External Background Knowledge*
Shuang Chen, Jinpeng Wang, Xiaocheng Feng, Feng Jiang, Bing Qin and Chin-Yew Lin

*Reading Like HER: Human Reading Inspired Extractive Summarization*
Ling Luo, Xiang Ao, Yan Song, Feiyang Pan, Min Yang and Qing He

*Contrastive Attention Mechanism for Abstractive Sentence Summarization*
Xiangyu Duan, Hongfei Yu, Mingming Yin, Min Zhang, Weihua Luo and Yue Zhang

*NCLS: Neural Cross-Lingual Summarization*
Junnan Zhu, Qian Wang, Yining Wang, Yu Zhou, Jiajun Zhang, Shaonan Wang and Chengqing Zong

*Clickbait? Sensational Headline Generation with Auto-tuned Reinforcement Learning*
Peng Xu, Chien-Sheng Wu, Andrea Madotto and Pascale Fung

*Concept Pointer Network for Abstractive Summarization*
Wenbo Wang, Yang Gao, Heyan Huang and Yuxiang Zhou

*Surface Realisation Using Full Delexicalisation*
Anastasia Shimorina and Claire Gardent

*IMaT: Unsupervised Text Attribute Transfer via Iterative Matching and Translation*
Zhijing Jin, Di Jin, Jonas Mueller, Nicholas Matthews and Enrico Santus

*Better Rewards Yield Better Summaries: Learning to Summarise Without References*
Florian Böhm, Yang Gao, Christian M. Meyer, Ori Shapira, Ido Dagan and Iryna Gurevych

*Mixture Content Selection for Diverse Sequence Generation*
Jaemin Cho, Minjoon Seo and Hannaneh Hajishirzi

*An End-to-End Generative Architecture for Paraphrase Generation*
Qian Yang, zhouyuan huo, Dinghan Shen, Yong Cheng, Wenlin Wang, Guoyin Wang and Lawrence Carin

*Table-to-Text Generation with Effective Hierarchical Encoder on Three Dimensions (Row, Column and Time)*
Heng Gong, Xiaocheng Feng, Bing Qin and Ting Liu

*Subtopic-driven Multi-Document Summarization*
Xin Zheng, Aixin Sun, Jing Li and Karthik Muthuswamy

*Referring Expression Generation Using Entity Profiles*
Meng Cao and Jackie Chi Kit Cheung

*Exploring Diverse Expressions for Paraphrase Generation*
Lihua Qian, Lin Qiu, Weinan Zhang, Xin Jiang and Yong Yu

*Enhancing AMR-to-Text Generation with Dual Graph Representations*
Leonardo F. R. Ribeiro, Claire Gardent and Iryna Gurevych

*Keeping Consistency of Sentence Generation and Document Classification with Multi-Task Learning*
Toru Nishino, Shotaro Misawa, Ryuji Kano, Tomoki Taniguchi, Yasuhide Miura and Tomoko Ohkuma

*Toward a Task of Feedback Comment Generation for Writing Learning*
Ryo Nagata

*Improving Question Generation With to the Point Context*
Jingjing Li, Yifan Gao, Lidong Bing, Irwin King and Michael R. Lyu

*Deep Copycat Networks for Text-to-Text Generation*
Julia Ive, Pranava Madhyastha and Lucia Specia

*Towards Controllable and Personalized Review Generation*
Pan Li and Alexander Tuzhilin

*Answers Unite! Unsupervised Metrics for Reinforced Summarization Models*
Thomas Scialom, Sylvain Lamprier, Benjamin Piwowarski and Jacopo Staiano

*Long and Diverse Text Generation with Planning-based Hierarchical Variational Model*
Zhihong Shao, Minlie Huang, Jiangtao Wen, Wenfei Xu and xiaoyan zhu

*"Transforming" Delete, Retrieve, Generate Approach for Controlled Text Style Transfer*
Akhilesh Sudhakar, Bhargav Upadhyay and Arjun Maheswaran

*An Entity-Driven Framework for Abstractive Summarization*
Eva Sharma, Luyang Huang, Zhe Hu and Lu Wang

*Neural Extractive Text Summarization with Syntactic Compression*
Jiacheng Xu and Greg Durrett

*Domain Adaptive Text Style Transfer*
Dianqi Li, Yizhe Zhang, Zhe Gan, Yu Cheng, Chris Brockett, Bill Dolan and Ming-Ting Sun

*Let's Ask Again: Refine Network for Automatic Question Generation*
Preksha Nema, Akash Kumar Mohankumar, Mitesh M. Khapra, Balaji Vasan Srinivasan and Balaraman Ravindran

*Earlier Isn't Always Better: Sub-aspect Analysis on Corpus and System Biases in Summarization*
Taehee Jung, Dongyeop Kang, Lucas Mentch and Eduard Hovy

[DEMO] *VizSeq: a visual analysis toolkit for text generation tasks*
Changhan Wang, Anirudh Jain, Danlu Chen and Jiatao Gu

[DEMO] *FAMULUS: Interactive Annotation and Feedback Generation for Teaching Diagnostic Reasoning*
Jonas Pfeiffer, Christian M. Meyer, Claudia Schulz, Jan Kiesewetter, Jan Zottmann, Michael Sailer, Elisabeth Bauer, Frank Fischer, Martin R. Fischer and Iryna Gurevych

[DEMO] *A Summarization System for Scientific Documents*
Shai Erera, Michal Shmueli-Scheuer, Guy Feigenblat, Ora Peled Nakash, Odellia Boni, Haggai Roitman, Doron Cohen, Bar Weiner, Yosi Mass, Or Rivlin, Guy Lev, Achiya Jerbi, Jonathan Herzig, Yufang Hou, Charles Jochim, Martin Gleize, Francesca Bonin, Francesca Bonin and David Konopnicki

[DEMO] *EASSE: Easier Automatic Sentence Simplification Evaluation*
Fernando Alva-Manchego, Louis Martin, Carolina Scarton and Lucia Specia

[DEMO] *ALTER: Auxiliary Text Rewriting Tool for Natural Language Generation*
Qiongkai Xu, Chenchen Xu and Lizhen Qu

15:00–15:30 *Coffee Break*

15:30–16:18 *Session 7*

**Session 7A: Machine Translation and Multilinguality I**

15:30–15:42 *Lost in Evaluation: Misleading Benchmarks for Bilingual Dictionary Induction*
Yova Kementchedjhieva, Mareike Hartmann and Anders Søgaard

15:42–15:54 *Towards Realistic Practices In Low-Resource Natural Language Processing: The Development Set*
Katharina Kann, Kyunghyun Cho and Samuel R. Bowman

15:54–16:06 *Synchronously Generating Two Languages with Interactive Decoding*
Yining Wang, Jiajun Zhang, Long Zhou, Yuchen Liu and Chengqing Zong

16:06–16:18 *On NMT Search Errors and Model Errors: Cat Got Your Tongue?*
Felix Stahlberg and Bill Byrne

**Session 7B: Reasoning and Question Answering**

15:30–15:42 *"Going on a vacation" takes longer than "Going for a walk": A Study of Temporal Commonsense Understanding*
Ben Zhou, Daniel Khashabi, Qiang Ning and Dan Roth

15:42–15:54 *QAInfomax: Learning Robust Question Answering System by Mutual Information Maximization*
Yi-Ting Yeh and Yun-Nung Chen

15:54–16:06 *Adapting Meta Knowledge Graph Information for Multi-Hop Reasoning over Few-Shot Relations*
Xin Lv, Yuxian Gu, Xu Han, Lei Hou, Juanzi Li and Zhiyuan Liu

16:06–16:18 *How Reasonable are Common-Sense Reasoning Tasks: A Case-Study on the Winograd Schema Challenge and SWAG*
Paul Trichelair, Ali Emami, Adam Trischler, Kaheer Suleman and Jackie Chi Kit Cheung

**Session 7C: Generation I**

15:30–15:42 *Pun-GAN: Generative Adversarial Network for Pun Generation*
Fuli Luo, Shunyao Li, Pengcheng Yang, Lei Li, Baobao Chang, Zhifang Sui and Xu SUN

15:42–15:54 *Multi-Task Learning with Language Modeling for Question Generation*
Wenjie Zhou, Minghua Zhang and Yunfang Wu

15:54–16:06 *Autoregressive Text Generation Beyond Feedback Loops*
Florian Schmidt, Stephan Mandt and Thomas Hofmann

16:06–16:18 *The Woman Worked as a Babysitter: On Biases in Language Generation*
Emily Sheng, Kai-Wei Chang, Premkumar Natarajan and Nanyun Peng

**Session 7D: Sentiment Analysis and Argument Mining III**

15:30–15:42 *On the Importance of Delexicalization for Fact Verification*
Sandeep Suntwal, Mithun Paul, Rebecca Sharp and Mihai Surdeanu

15:42–15:54 *Towards Debiasing Fact Verification Models*
Tal Schuster, Darsh Shah, Yun Jie Serene Yeo, Daniel Roberto Filizzola Ortiz, Enrico Santus and Regina Barzilay

15:54–16:06 *Recognizing Conflict Opinions in Aspect-level Sentiment Classification with Dual Attention Networks*
Xingwei Tan, Yi Cai and Changxi Zhu

16:06–16:18 *Investigating Dynamic Routing in Tree-Structured LSTM for Sentiment Analysis*
Jin Wang, Liang-Chih Yu, K. Robert Lai and Xuejie Zhang

**Poster and Demo Session 7: Information Retrieval and Document Analysis, Lexical Semantics, Sentence-level Semantics, Machine Learning**

*A Label Informative Wide & Deep Classifier for Patents and Papers*
Muyao Niu and Jie Cai

*Text Level Graph Neural Network for Text Classification*
Lianzhe Huang, Dehong Ma, Sujian Li, Xiaodong Zhang and Houfeng WANG

*Semantic Relatedness Based Re-ranker for Text Spotting*
Ahmed Sabir, Francesc Moreno and Lluís Padró

*Delta-training: Simple Semi-Supervised Text Classification using Pretrained Word Embeddings*
Hwiyeol Jo and Ceyda Cinarel

*Visual Detection with Context for Document Layout Analysis*
Carlos Soto and Shinjae Yoo

*Evaluating Topic Quality with Posterior Variability*
Linzi Xing, Michael J. Paul and Giuseppe Carenini

*Neural Topic Model with Reinforcement Learning*
Lin Gui, Jia Leng, Gabriele Pergola, yu zhou, Ruifeng Xu and Yulan He

*Modelling Stopping Criteria for Search Results using Poisson Processes*
Alison Sneyd and Mark Stevenson

*Cross-Domain Modeling of Sentence-Level Evidence for Document Retrieval*
Zeynep Akkalyoncu Yilmaz, Wei Yang, Haotian Zhang and Jimmy Lin

*The Challenges of Optimizing Machine Translation for Low Resource Cross-Language Information Retrieval*
Constantine Lignos, Daniel Cohen, Yen-Chieh Lien, Pratik Mehta, W. Bruce Croft and Scott Miller

*Rotate King to get Queen: Word Relationships as Orthogonal Transformations in Embedding Space*
Kawin Ethayarajh

*GlossBERT: BERT for Word Sense Disambiguation with Gloss Knowledge*
Luyao Huang, Chi Sun, Xipeng Qiu and Xuanjing Huang

*Leveraging Adjective-Noun Phrasing Knowledge for Comparison Relation Prediction in Text-to-SQL*
Haoyan Liu, Lei Fang, Qian Liu, Bei Chen, Jian-Guang LOU and Zhoujun Li

*Bridging the Defined and the Defining: Exploiting Implicit Lexical Semantic Relations in Definition Modeling*
Koki Washio, Satoshi Sekine and Tsuneaki Kato

*Don't Just Scratch the Surface: Enhancing Word Representations for Korean with Hanja*
Kang Min Yoo, Taeuk Kim and Sang-goo Lee

*SyntagNet: Challenging Supervised Word Sense Disambiguation with Lexical-Semantic Combinations*
Marco Maru, Federico Scozzafava, Federico Martelli and Roberto Navigli

*Hierarchical Meta-Embeddings for Code-Switching Named Entity Recognition*
Genta Indra Winata, Zhaojiang Lin, Jamin Shin, Zihan Liu and Pascale Fung

*Fine-tune BERT with Sparse Self-Attention Mechanism*
Baiyun Cui, Yingming Li, Ming Chen and Zhongfei Zhang

*Feature-Dependent Confusion Matrices for Low-Resource NER Labeling with Noisy Labels*
Lukas Lange, Michael A. Hedderich and Dietrich Klakow

*A Multi-Pairwise Extension of Procrustes Analysis for Multilingual Word Translation*
Hagai Taitelbaum, Gal Chechik and Jacob Goldberger

*Out-of-Domain Detection for Low-Resource Text Classification Tasks*
Ming Tan, Yang Yu, Haoyu Wang, Dakuo Wang, Saloni Potdar, Shiyu Chang and Mo Yu

*Harnessing Pre-Trained Neural Networks with Rules for Formality Style Transfer*
Yunli Wang, Yu Wu, Lili Mou, Zhoujun Li and Wenhan Chao

*Multiple Text Style Transfer by using Word-level Conditional Generative Adversarial Network with Two-Phase Training*
Chih-Te Lai, Yi-Te Hong, Hong-You Chen, Chi-Jen Lu and Shou-De Lin

*Improved Differentiable Architecture Search for Language Modeling and Named Entity Recognition*
Yufan Jiang, Chi Hu, Tong Xiao, Chunliang Zhang and Jingbo Zhu

*Using Pairwise Occurrence Information to Improve Knowledge Graph Completion on Large-Scale Datasets*
Esma Balkir, Masha Naslidnyk, Dave Palfrey and Arpit Mittal

*Single Training Dimension Selection for Word Embedding with PCA*
Yu Wang

*A Surprisingly Effective Fix for Deep Latent Variable Modeling of Text*
Bohan Li, Junxian He, Graham Neubig, Taylor Berg-Kirkpatrick and Yiming Yang

*SciBERT: A Pretrained Language Model for Scientific Text*
Iz Beltagy, Kyle Lo and Arman Cohan

*Humor Detection: A Transformer Gets the Last Laugh*
Orion Weller and Kevin Seppi

*Combining Global Sparse Gradients with Local Gradients in Distributed Neural Network Training*
Alham Fikri Aji, Kenneth Heafield and Nikolay Bogoychev

*Small and Practical BERT Models for Sequence Labeling*
Henry Tsai, Jason Riesa, Melvin Johnson, Naveen Arivazhagan, Xin Li and Amelia Archer

*Data Augmentation with Atomic Templates for Spoken Language Understanding*
Zijian Zhao, Su Zhu and Kai Yu

*PaLM: A Hybrid Parser and Language Model*
Hao Peng, Roy Schwartz and Noah A. Smith

*A Pilot Study for Chinese SQL Semantic Parsing*
Qingkai Min, Yuefeng Shi and Yue Zhang

*Global Reasoning over Database Structures for Text-to-SQL Parsing*
Ben Bogin, Matt Gardner and Jonathan Berant

*Transductive Learning of Neural Language Models for Syntactic and Semantic Analysis*
Hiroki Ouchi, Jun Suzuki and Kentaro Inui

*Efficient Sentence Embedding using Discrete Cosine Transform*
Nada Almarwani, Hanan Aldarmaki and Mona Diab

*A Search-based Neural Model for Biomedical Nested and Overlapping Event Detection*
Kurt Junshean Espinosa, Makoto Miwa and Sophia Ananiadou

*PAWS-X: A Cross-lingual Adversarial Dataset for Paraphrase Identification*
Yinfei Yang, Yuan Zhang, Chris Tar and Jason Baldridge

*Pretrained Language Models for Sequential Sentence Classification*
Arman Cohan, Iz Beltagy, Daniel King, Bhavana Dalvi and Dan Weld

*Emergent Linguistic Phenomena in Multi-Agent Communication Games*
Laura Harding Graesser, Kyunghyun Cho and Douwe Kiela

*TalkDown: A Corpus for Condescension Detection in Context*
Zijian Wang and Christopher Potts

**Wednesday, November 6, 2019 (continued)**

16:18–16:30 *Mini-Break*

16:30–18:00 *Session 8*

**Session 8A: Summarization**

16:30–16:48 *Summary Cloze: A New Task for Content Selection in Topic-Focused Summarization*
Daniel Deutsch and Dan Roth

16:48–17:06 *Text Summarization with Pretrained Encoders*
Yang Liu and Mirella Lapata

17:06–17:24 *How to Write Summaries with Patterns? Learning towards Abstractive Summarization through Prototype Editing*
Shen Gao, Xiuying Chen, Piji Li, Zhangming Chan, Dongyan Zhao and Rui Yan

17:24–17:42 *BottleSum: Unsupervised and Self-supervised Sentence Summarization using the Information Bottleneck Principle*
Peter West, Ari Holtzman, Jan Buys and Yejin Choi

17:42–18:00 *Improving Latent Alignment in Text Summarization by Generalizing the Pointer Generator*
Xiaoyu Shen, Yang Zhao, Hui Su and Dietrich Klakow

**Session 8B: Sentence-level Semantics II**

16:30–16:48 *Learning Semantic Parsers from Denotations with Latent Structured Alignments and Abstract Programs*
Bailin Wang, Ivan Titov and Mirella Lapata

16:48–17:06 *Broad-Coverage Semantic Parsing as Transduction*
Sheng Zhang, Xutai Ma, Kevin Duh and Benjamin Van Durme

17:06–17:24 *Core Semantic First: A Top-down Approach for AMR Parsing*
Deng Cai and Wai Lam

17:24–17:42 *Don't paraphrase, detect! Rapid and Effective Data Collection for Semantic Parsing*
Jonathan Herzig and Jonathan Berant

17:42–18:00 [TACL] *Massively Multilingual Sentence Embeddings for Zero-Shot Cross-Lingual Transfer and Beyond*
Mikel Artetxe and Holger Schwenk

### Session 8C: Information Extraction II

16:30–16:48 *Improving Distantly-Supervised Relation Extraction with Joint Label Embedding*
Linmei Hu, Luhao Zhang, Chuan Shi, Liqiang Nie, Weili Guan and Cheng Yang

16:48–17:06 *Leverage Lexical Knowledge for Chinese Named Entity Recognition via Collaborative Graph Network*
Dianbo Sui, Yubo Chen, Kang Liu, Jun Zhao and Shengping Liu

17:06–17:24 *Looking Beyond Label Noise: Shifted Label Distribution Matters in Distantly Supervised Relation Extraction*
Qinyuan Ye, Liyuan Liu, Maosen Zhang and Xiang Ren

17:24–17:42 *Easy First Relation Extraction with Information Redundancy*
Shuai Ma, Gang Wang, Yansong Feng and Jinpeng Huai

17:42–18:00 *Dependency-Guided LSTM-CRF for Named Entity Recognition*
Zhanming Jie and Wei Lu

### Session 8D: Information Retrieval and Document Analysis I

16:30–16:48 *Cross-Cultural Transfer Learning for Text Classification*
Dor Ringel, Gal Lavee, Ido Guy and Kira Radinsky

16:48–17:06 *Combining Unsupervised Pre-training and Annotator Rationales to Improve Low-shot Text Classification*
Oren Melamud, Mihaela Bornea and Ken Barker

17:06–17:24 *ProSeqo: Projection Sequence Networks for On-Device Text Classification*
Zornitsa Kozareva and Sujith Ravi

17:24–17:42 *Induction Networks for Few-Shot Text Classification*
Ruiying Geng, Binhua Li, Yongbin Li, Xiaodan Zhu, Ping Jian and Jian Sun

17:42–18:00 *Benchmarking Zero-shot Text Classification: Datasets, Evaluation and Entailment Approach*
Wenpeng Yin, Jamaal Hay and Dan Roth

**Poster and Demo Session 8: Machine Learning**

*A Logic-Driven Framework for Consistency of Neural Models*
Tao Li, Vivek Gupta, Maitrey Mehta and Vivek Srikumar

*Style Transfer for Texts: Retrain, Report Errors, Compare with Rewrites*
Alexey Tikhonov, Viacheslav Shibaev, Aleksander Nagaev, Aigul Nugmanova and Ivan P. Yamshchikov

*Implicit Deep Latent Variable Models for Text Generation*
Le Fang, Chunyuan Li, Jianfeng Gao, Wen Dong and Changyou Chen

*Text Emotion Distribution Learning from Small Sample: A Meta-Learning Approach*
Zhenjie Zhao and Xiaojuan Ma

*Judge the Judges: A Large-Scale Evaluation Study of Neural Language Models for Online Review Generation*
Cristina Garbacea, Samuel Carton, Shiyan Yan and Qiaozhu Mei

*Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*
Nils Reimers and Iryna Gurevych

*Learning Only from Relevant Keywords and Unlabeled Documents*
Nontawat Charoenphakdee, Jongyeong Lee, Yiping Jin, Dittaya Wanvarie and Masashi Sugiyama

*Denoising based Sequence-to-Sequence Pre-training for Text Generation*
Liang Wang, Wei Zhao, Ruoyu Jia, Sujian Li and Jingming Liu

*Dialog Intent Induction with Deep Multi-View Clustering*
Hugh Perkins and Yi Yang

*Nearly-Unsupervised Hashcode Representations for Biomedical Relation Extraction*
Sahil Garg, Aram Galstyan, Greg Ver Steeg and Guillermo Cecchi

*Auditing Deep Learning processes through Kernel-based Explanatory Models*
Danilo Croce, Daniele Rossini and Roberto Basili

*Enhancing Variational Autoencoders with Mutual Information Neural Estimation for Text Generation*
Dong Qian and William K. Cheung

*Sampling Bias in Deep Active Classification: An Empirical Study*
Ameya Prabhu, Charles Dognin and Maneesh Singh

*Don't Take the Easy Way Out: Ensemble Based Methods for Avoiding Known Dataset Biases*
Christopher Clark, Mark Yatskar and Luke Zettlemoyer

*Achieving Verified Robustness to Symbol Substitutions via Interval Bound Propagation*
Po-Sen Huang, Robert Stanforth, Johannes Welbl, Chris Dyer, Dani Yogatama, Sven Gowal, Krishnamurthy Dvijotham and Pushmeet Kohli

*Rethinking Cooperative Rationalization: Introspective Extraction and Complement Control*
Mo Yu, Shiyu Chang, Yang Zhang and Tommi Jaakkola

*Experimenting with Power Divergences for Language Modeling*
Matthieu Labeau and Shay B. Cohen

*Hierarchically-Refined Label Attention Network for Sequence Labeling*
Leyang Cui and Yue Zhang

*Certified Robustness to Adversarial Word Substitutions*
Robin Jia, Aditi Raghunathan, Kerem Göksel and Percy Liang

*Visualizing and Understanding the Effectiveness of BERT*
Yaru Hao, Li Dong, Furu Wei and Ke Xu

*Topics to Avoid: Demoting Latent Confounds in Text Classification*
Sachin Kumar, Shuly Wintner, Noah A. Smith and Yulia Tsvetkov

*Learning to Ask for Conversational Machine Learning*
Shashank Srivastava, Igor Labutov and Tom Mitchell

*Language Modeling for Code-Switching: Evaluation, Integration of Monolingual Data, and Discriminative Training*
Hila Gonen and Yoav Goldberg

*Using Local Knowledge Graph Construction to Scale Seq2Seq Models to Multi-Document Inputs*
Angela Fan, Claire Gardent, Chloé Braud and Antoine Bordes

*Fine-grained Knowledge Fusion for Sequence Labeling Domain Adaptation*
Huiyun Yang, Shujian Huang, XIN-YU DAI and Jiajun CHEN

*Exploiting Monolingual Data at Scale for Neural Machine Translation*
Lijun Wu, Yiren Wang, Yingce Xia, Tao QIN, Jianhuang Lai and Tie-Yan Liu

*Meta Relational Learning for Few-Shot Link Prediction in Knowledge Graphs*
Mingyang Chen, Wen Zhang, Wei Zhang, Qiang Chen and Huajun Chen

*Distributionally Robust Language Modeling*
Yonatan Oren, Shiori Sagawa, Tatsunori Hashimoto and Percy Liang

*Unsupervised Domain Adaptation of Contextualized Embeddings for Sequence Labeling*
Xiaochuang Han and Jacob Eisenstein

*Learning Latent Parameters without Human Response Patterns: Item Response Theory with Artificial Crowds*
John P. Lalor, Hao Wu and Hong Yu

*Parallel Iterative Edit Models for Local Sequence Transduction*
Abhijeet Awasthi, Sunita Sarawagi, Rasna Goyal, Sabyasachi Ghosh and Vihari Piratla

*ARAML: A Stable Adversarial Training Framework for Text Generation*
Pei Ke, Fei Huang, Minlie Huang and xiaoyan zhu

*FlowSeq: Non-Autoregressive Conditional Sequence Generation with Generative Flow*
Xuezhe Ma, Chunting Zhou, Xian Li, Graham Neubig and Eduard Hovy

*Compositional Generalization for Primitive Substitutions*
Yuanpeng Li, Liang Zhao, Jianyu Wang and Joel Hestness

*WikiCREM: A Large Unsupervised Corpus for Coreference Resolution*
Vid Kocijan, Oana-Maria Camburu, Ana-Maria Cretu, Yordan Yordanov, Phil Blunsom and Thomas Lukasiewicz

*Identifying and Explaining Discriminative Attributes*
Armins Stepanjans and André Freitas

*Patient Knowledge Distillation for BERT Model Compression*
Siqi Sun, Yu Cheng, Zhe Gan and Jingjing Liu

*Neural Gaussian Copula for Variational Autoencoder*
Prince Zizhuang Wang and William Yang Wang

*Transformer Dissection: An Unified Understanding for Transformer's Attention via the Lens of Kernel*
Yao-Hung Hubert Tsai, Shaojie Bai, Makoto Yamada, Louis-Philippe Morency and Ruslan Salakhutdinov

*Learning to Learn and Predict: A Meta-Learning Approach for Multi-Label Classification*
Jiawei Wu, Wenhan Xiong and William Yang Wang

*Revealing the Dark Secrets of BERT*
Olga Kovaleva, Alexey Romanov, Anna Rogers and Anna Rumshisky

*Machine Translation With Weakly Paired Documents*
Lijun Wu, Jinhua Zhu, Di He, Fei Gao, Tao QIN, Jianhuang Lai and Tie-Yan Liu

*Countering Language Drift via Visual Grounding*
Jason Lee, Kyunghyun Cho and Douwe Kiela

*The Bottom-up Evolution of Representations in the Transformer: A Study with Machine Translation and Language Modeling Objectives*
Elena Voita, Rico Sennrich and Ivan Titov

[DEMO] *NeuronBlocks: Building Your NLP DNN Models Like Playing Lego*
Ming Gong, Linjun Shou, Wutao Lin, Zhijie Sang, Quanjia Yan, Ze Yang, Feixiang Cheng and Daxin Jiang

**Wednesday, November 6, 2019 (continued)**

[DEMO] *Controlling Sequence-to-Sequence Models - A Demonstration on Neural-based Acrostic Generator*
Liang-Hsin Shen, Pei-Lun Tai, Chao-Chung Wu and Shou-De Lin

[DEMO] *AllenNLP Interpret: A Framework for Explaining Predictions of NLP Models*
Eric Wallace, Jens Tuyls, Junlin Wang, Sanjay Subramanian, Matt Gardner and Sameer Singh

[DEMO] *UER: An Open-Source Toolkit for Pre-training Models*
Zhe Zhao, Hui Chen, Jinbin Zhang, Xin Zhao, Tao Liu, Wei Lu, Xi Chen, Haotang Deng, Qi Ju and Xiaoyong Du

[DEMO] *MedCATTrainer: A Biomedical Free Text Annotation Interface with Active Learning and Research Use Case Specific Customisation*
Thomas Searle, Zeljko Kraljevic, Rebecca Bendayan, Daniel Bean and Richard Dobson

**Thursday, November 7, 2019**

09:00–10:00 *Keynote III: Kyunghyun Cho*

10:00–10:30 *Coffee Break*

10:30–12:00 *Session 9*

### Session 9A: Machine Translation and Multilinguality II

10:30–10:48 *Do We Really Need Fully Unsupervised Cross-Lingual Embeddings?*
Ivan Vulić, Goran Glavaš, Roi Reichart and Anna Korhonen

10:48–11:06 *Weakly-Supervised Concept-based Adversarial Learning for Cross-lingual Word Embeddings*
Haozhou Wang, James Henderson and Paola Merlo

11:06–11:24 *Aligning Cross-Lingual Entities with Multi-Aspect Information*
Hsiu-Wei Yang, Yanyan Zou, Peng Shi, Wei Lu, Jimmy Lin and Xu SUN

11:24–11:42 *Contrastive Language Adaptation for Cross-Lingual Stance Detection*
Mitra Mohtarami, James Glass and Preslav Nakov

11:42–12:00 *Jointly Learning to Align and Translate with Transformer Models*
Sarthak Garg, Stephan Peitz, Udhyakumar Nallasamy and Matthias Paulik

### Session 9B: Reasoning

10:30–10:48 *Social IQa: Commonsense Reasoning about Social Interactions*
Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras and Yejin Choi

10:48–11:06 *Self-Assembling Modular Networks for Interpretable Multi-Hop Reasoning*
Yichen Jiang and Mohit Bansal

11:06–11:24 *Posing Fair Generalization Tasks for Natural Language Inference*
Atticus Geiger, Ignacio Cases, Lauri Karttunen and Christopher Potts

11:24–11:42 *Everything Happens for a Reason: Discovering the Purpose of Actions in Procedural Text*
Bhavana Dalvi, Niket Tandon, Antoine Bosselut, Wen-tau Yih and Peter Clark

11:42–12:00 *CLUTRR: A Diagnostic Benchmark for Inductive Reasoning from Text*
Koustuv Sinha, Shagun Sodhani, Jin Dong, Joelle Pineau and William L. Hamilton

### Session 9C: Dialog and Interactive Systems II

10:30–10:48 *Taskmaster-1: Toward a Realistic and Diverse Dialog Dataset*
Bill Byrne, Karthik Krishnamoorthi, Chinnadhurai Sankar, Arvind Neelakantan, Ben Goodrich, Daniel Duckworth, Semih Yavuz, Amit Dubey, Kyu-Young Kim and Andy Cedilnik

10:48–11:06 *Multi-Domain Goal-Oriented Dialogues (MultiDoGO): Strategies toward Curating and Annotating Large Scale Dialogue Data*
Denis Peskov, Nancy Clarke, Jason Krone, Brigi Fodor, Yi Zhang, Adel Youssef and Mona Diab

11:06–11:24 *Build it Break it Fix it for Dialogue Safety: Robustness from Adversarial Human Attack*
Emily Dinan, Samuel Humeau, Bharath Chintagunta and Jason Weston

11:24–11:42 *GECOR: An End-to-End Generative Ellipsis and Co-reference Resolution Model for Task-Oriented Dialogue*
Jun Quan, Deyi Xiong, Bonnie Webber and Changjian Hu

11:42–12:00 *Task-Oriented Conversation Generation Using Heterogeneous Memory Networks*
Zehao Lin, Xinjing Huang, Feng Ji, Haiqing Chen and Yin Zhang

### Session 9D: Sentiment Analysis and Argument Mining IV

10:30–10:48 *Aspect-based Sentiment Classification with Aspect-specific Graph Convolutional Networks*
Chen Zhang, Qiuchi Li and Dawei Song

10:48–11:06 *Coupling Global and Local Context for Unsupervised Aspect Extraction*
Ming Liao, Jing Li, Haisong Zhang, Lingzhi Wang, Xixin Wu and Kam-Fai Wong

11:06–11:24 *Transferable End-to-End Aspect-based Sentiment Analysis with Selective Adversarial Learning*
Zheng Li, Xin Li, Ying Wei, Lidong Bing, Yu Zhang and Qiang Yang

11:24–11:42 *CAN: Constrained Attention Networks for Multi-Aspect Sentiment Analysis*
Mengting Hu, Shiwan Zhao, Li Zhang, Keke Cai, Zhong Su, Renhong Cheng and Xiaowei Shen

11:42–12:00 *Leveraging Just a Few Keywords for Fine-Grained Aspect Detection Through Weakly Supervised Co-Training*
Giannis Karamanolakis, Daniel Hsu and Luis Gravano

**Poster and Demo Session 9: Social Media and Computational Social Science, Text Mining and NLP Applications**

*Integrating Text and Image: Determining Multimodal Document Intent in Instagram Posts*
Julia Kruk, Jonah Lubin, Karan Sikka, Xiao Lin, Dan Jurafsky and Ajay Divakaran

*Neural Conversation Recommendation with Online Interaction Modeling*
Xingshan Zeng, Jing Li, Lu Wang and Kam-Fai Wong

*Different Absorption from the Same Sharing: Sifted Multi-task Learning for Fake News Detection*
Lianwei Wu, Yuan Rao, Haolin Jin, Ambreen Nazir and Ling Sun

*Text-based inference of moral sentiment change*
Jing Yi Xie, Renato Ferreira Pinto Junior, Graeme Hirst and Yang Xu

*Detecting Causal Language Use in Science Findings*
Bei Yu, Yingya Li and Jun Wang

*Multilingual and Multi-Aspect Hate Speech Analysis*
Nedjma Ousidhoum, Zizheng Lin, Hongming Zhang, Yangqiu Song and Dit-Yan Yeung

*MultiFC: A Real-World Multi-Domain Dataset for Evidence-Based Fact Checking of Claims*
Isabelle Augenstein, Christina Lioma, Dongsheng Wang, Lucas Chaves Lima, Casper Hansen, Christian Hansen and Jakob Grue Simonsen

*A Deep Neural Information Fusion Architecture for Textual Network Embeddings*
Zenan Xu, Qinliang Su, Xiaojun Quan and Weijia Zhang

*You Shall Know a User by the Company It Keeps: Dynamic Representations for Social Media Users in NLP*
Marco Del Tredici, Diego Marcheggiani, Sabine Schulte im Walde and Raquel Fernández

*Adaptive Ensembling: Unsupervised Domain Adaptation for Political Document Analysis*
Shrey Desai, Barea Sinno, Alex Rosenfeld and Junyi Jessy Li

*Macrocosm: Social Media Persona Linking for Open Source Intelligence Applications*
Graham Horwood, Ning Yu, Thomas Boggs, Changjiang Yang and Chad Holvenstot

*A Hierarchical Location Prediction Neural Network for Twitter User Geolocation*
Binxuan Huang and Kathleen Carley

*Trouble on the Horizon: Forecasting the Derailment of Online Conversations as they Develop*
Jonathan P. Chang and Cristian Danescu-Niculescu-Mizil

*A Benchmark Dataset for Learning to Intervene in Online Hate Speech*
Jing Qian, Anna Bethke, Yinyin Liu, Elizabeth Belding and William Yang Wang

*Detecting and Reducing Bias in a High Stakes Domain*
Ruiqi Zhong, Yanda Chen, Desmond Patton, Charlotte Selous and Kathy McKeown

*CodeSwitch-Reddit: Exploration of Written Multilingual Discourse in Online Discussion Forums*
Ella Rabinovich, Masih Sultani and Suzanne Stevenson

*Modeling Conversation Structure and Temporal Dynamics for Jointly Predicting Rumor Stance and Veracity*
Penghui Wei, Nan Xu and Wenji Mao

[TACL] *Measuring Online Debaters' Persuasive Skill from Text over Time*
Kelvin Luu, Chenhao Tan and Noah Smith

*Reconstructing Capsule Networks for Zero-shot Intent Classification*
Han Liu, Xiaotong Zhang, Lu Fan, Xuandi Fu, Qimai Li, Xiao-Ming Wu and Albert Y.S. Lam

*Domain Adaptation for Person-Job Fit with Transferable Deep Global Match Network*
Shuqing Bian, Wayne Xin Zhao, Yang Song, Tao Zhang and Ji-Rong Wen

*Heterogeneous Graph Attention Networks for Semi-supervised Short Text Classification*
Hu Linmei, Tianchi Yang, Chuan Shi, Houye Ji and Xiaoli Li

*Comparing and Developing Tools to Measure the Readability of Domain-Specific Texts*
Elissa Redmiles, Lisa Maszkiewicz, Emily Hwang, Dhruv Kuchhal, Everest Liu, Miraida Morales, Denis Peskov, Sudha Rao, Rock Stevens, Kristina Gligorić, Sean Kross, Michelle Mazurek and Hal Daumé III

*News2vec: News Network Embedding with Subnode Information*
Ye Ma, Lu Zong, Yikang Yang and Jionglong Su

*Recursive Context-Aware Lexical Simplification*
Sian Gooding and Ekaterina Kochmar

*Leveraging Medical Literature for Section Prediction in Electronic Health Records*
Sara Rosenthal, Ken Barker and Zhicheng Liang

*Neural News Recommendation with Heterogeneous User Behavior*
Chuhan Wu, Fangzhao Wu, Mingxiao An, Tao Qi, Jianqiang Huang, Yongfeng Huang and Xing Xie

*Reviews Meet Graphs: Enhancing User and Item Representations for Recommendation with Hierarchical Attentive Graph Neural Network*
Chuhan Wu, Fangzhao Wu, Tao Qi, Suyu Ge, Yongfeng Huang and Xing Xie

*Event Representation Learning Enhanced with External Commonsense Knowledge*
Xiao Ding, Kuo Liao, Ting Liu, Zhongyang Li and Junwen Duan

*Learning to Discriminate Perturbations for Blocking Adversarial Attacks in Text Classification*
Yichao Zhou, Jyun-Yu Jiang, Kai-Wei Chang and Wei Wang

*A Neural Citation Count Prediction Model based on Peer Review Text*
Siqing Li, Wayne Xin Zhao, Eddy Jing Yin and Ji-Rong Wen

*Connecting the Dots: Document-level Neural Relation Extraction with Edge-oriented Graphs*
Fenia Christopoulou, Makoto Miwa and Sophia Ananiadou

*Semi-supervised Text Style Transfer: Cross Projection in Latent Space*
Mingyue Shang, Piji Li, Zhenxin Fu, Lidong Bing, Dongyan Zhao, Shuming Shi and Rui Yan

*Question Answering for Privacy Policies: Combining Computational and Legal Perspectives*
Abhilasha Ravichander, Alan W Black, Shomir Wilson, Thomas Norton and Norman Sadeh

*Stick to the Facts: Learning towards a Fidelity-oriented E-Commerce Product Description Generation*
Zhangming Chan, Xiuying Chen, Yongliang Wang, Juntao Li, Zhiqiang Zhang, Kun Gai, Dongyan Zhao and Rui Yan

*Fine-Grained Entity Typing via Hierarchical Multi Graph Convolutional Networks*
Hailong Jin, Lei Hou, Juanzi Li and Tiansi Dong

*Learning to Infer Entities, Properties and their Relations from Clinical Conversations*
Nan Du, Mingqiu Wang, Linh Tran, Gang Lee and Izhak Shafran

*Practical Correlated Topic Modeling and Analysis via the Rectified Anchor Word Algorithm*
Moontae Lee, Sungjun Cho, David Bindel and David Mimno

*Modeling the Relationship between User Comments and Edits in Document Revision*
Xuchao Zhang, Dheeraj Rajagopal, Michael Gamon, Sujay Kumar Jauhar and ChangTien Lu

*PRADO: Projection Attention Networks for Document Classification On-Device*
Karthik Krishnamoorthi, Sujith Ravi and Zornitsa Kozareva

*Subword Language Model for Query Auto-Completion*
Gyuwan Kim

*Enhancing Dialogue Symptom Diagnosis with Global Attention and Symptom Graph*
Xinzhu Lin, Xiahui He, Qin Chen, Huaixiao Tou, Zhongyu Wei and Ting Chen

**Thursday, November 7, 2019 (continued)**

[DEMO] *TEASPN: Framework and Protocol for Integrated Writing Assistance Environments*
Masato Hagiwara, Takumi Ito, Tatsuki Kuribayashi, Jun Suzuki and Kentaro Inui

[DEMO] *Journalist-in-the-Loop: Continuous Learning as a Service for Rumour Analysis*
Twin Karmakharm, Nikolaos Aletras and Kalina Bontcheva

[DEMO] *MAssistant: A Personal Knowledge Assistant for MOOC Learners*
Lan Jiang, Shuhan Hu, Mingyu Huang, Zhichun Wang, Jinjian Yang, Xiaoju Ye and Wei Zheng

[DEMO] *A Stylometry Toolkit for Latin Literature*
Thomas J. Bolt, Jeffrey H. Flynt, Pramit Chaudhuri and Joseph P. Dexter

[DEMO] *Tanbih: Get To Know What You Are Reading*
Yifan Zhang, Giovanni Da San Martino, Alberto Barrón-Cedeño, Salvatore Romeo, Jisun An, Haewoon Kwak, Todor Staykovski, Israa Jaradat, Georgi Karadzhov, Ramy Baly, Kareem Darwish, James Glass and Preslav Nakov

[DEMO] *Visualizing Trends of Key Roles in News Articles*
Chen Xia, Haoxiang Zhang, Jacob Moghtader, Allen Wu and Kai-Wei Chang

12:00–12:30 *Lunch*

12:30–13:30 *SIGDAT Business Meeting*

13:30–15:00 *Session 10*

### Session 10A: Generation II

13:30–13:48 *Counterfactual Story Reasoning and Generation*
Lianhui Qin, Antoine Bosselut, Ari Holtzman, Chandra Bhagavatula, Elizabeth Clark and Yejin Choi

13:48–14:06 *Encode, Tag, Realize: High-Precision Text Editing*
Eric Malmi, Sebastian Krause, Sascha Rothe, Daniil Mirylenka and Aliaksei Severyn

14:06–14:24 *Answer-guided and Semantic Coherent Question Generation in Open-domain Conversation*
Weichao Wang, Shi Feng, Daling Wang and Yifei Zhang

14:24–14:42 *Read, Attend and Comment: A Deep Architecture for Automatic News Comment Generation*
Ze Yang, Can Xu, wei wu and zhoujun li

14:42–15:00 *A Topic Augmented Text Generation Model: Joint Learning of Semantics and Structural Features*
hongyin tang, Miao Li and Beihong Jin

### Session 10B: Speech, Vision, Robotics, Multimodal and Grounding II

13:30–13:48 *LXMERT: Learning Cross-Modality Encoder Representations from Transformers*
Hao Tan and Mohit Bansal

13:48–14:06 *Phrase Grounding by Soft-Label Chain Conditional Random Field*
Jiacheng Liu and Julia Hockenmaier

14:06–14:24 *What You See is What You Get: Visual Pronoun Coreference Resolution in Dialogues*
Xintong Yu, Hongming Zhang, Yangqiu Song, Yan Song and Changshui Zhang

14:24–14:42 *YouMakeup: A Large-Scale Domain-Specific Multimodal Dataset for Fine-Grained Semantic Comprehension*
Weiying Wang, Yongcheng Wang, Shizhe Chen and Qin Jin

14:42–15:00 *DEBUG: A Dense Bottom-Up Grounding Approach for Natural Language Video Localization*
Chujie Lu, Long Chen, Chilie Tan, Xiaolin Li and Jun Xiao

### Session 10C: Information Extraction III

13:30–13:48 *CrossWeigh: Training Named Entity Tagger from Imperfect Annotations*
Zihan Wang, Jingbo Shang, Liyuan Liu, Lihao Lu, Jiacheng Liu and Jiawei Han

13:48–14:06 *A Little Annotation does a Lot of Good: A Study in Bootstrapping Low-resource Named Entity Recognizers*
Aditi Chaudhary, Jiateng Xie, Zaid Sheikh, Graham Neubig and Jaime Carbonell

14:06–14:24 *Open Domain Web Keyphrase Extraction Beyond Language Modeling*
Lee Xiong, Chuan Hu, Chenyan Xiong, Daniel Campos and Arnold Overwijk

14:24–14:42 *TuckER: Tensor Factorization for Knowledge Graph Completion*
Ivana Balazevic, Carl Allen and Timothy Hospedales

14:42–15:00 [TACL] *Weakly Supervised Domain Detection*
Yumo Xu and Mirella Lapata

### Session 10D: Information Retrieval and Document Analysis II

13:30–13:48 *Human-grounded Evaluations of Explanation Methods for Text Classification*
Piyawat Lertvittayakumjorn and Francesca Toni

13:48–14:06 *A Context-based Framework for Modeling the Role and Function of On-line Resource Citations in Scientific Literature*
He Zhao, Zhunchen Luo, Chong Feng, Anqing Zheng and Xiaopeng Liu

14:06–14:24 *Adversarial Reprogramming of Text Classification Neural Networks*
Paarth Neekhara, Shehzeen Hussain, Shlomo Dubnov and Farinaz Koushanfar

14:24–14:42 *Document Hashing with Mixture-Prior Generative Models*
Wei Dong, Qinliang Su, Dinghan Shen and Changyou Chen

14:42–15:00 *On Efficient Retrieval of Top Similarity Vectors*
Shulong Tan, Zhixin Zhou, Zhaozhuo Xu and Ping Li

**Poster and Demo Session 10: Sentiment Analysis and Argument Mining, Lexical Semantics, Sentence-level Semantics**

*Multiplex Word Embeddings for Selectional Preference Acquisition*
Hongming Zhang, Jiaxin Bai, Yan Song, Kun Xu, Changlong Yu, Yangqiu Song, Wilfred Ng and Dong Yu

*MulCode: A Multiplicative Multi-way Model for Compressing Neural Language Model*
Yukun Ma, Patrick H. Chen and Cho-Jui Hsieh

*It's All in the Name: Mitigating Gender Bias with Name-Based Counterfactual Data Substitution*
Rowan Hall Maudslay, Hila Gonen, Ryan Cotterell and Simone Teufel

*Examining Gender Bias in Languages with Grammatical Gender*
Pei Zhou, Weijia Shi, Jieyu Zhao, Kuan-Hao Huang, Muhao Chen, Ryan Cotterell and Kai-Wei Chang

*Weakly Supervised Cross-lingual Semantic Relation Classification via Knowledge Distillation*
Yogarshi Vyas and Marine Carpuat

*Improved Word Sense Disambiguation Using Pre-Trained Contextualized Word Representations*
Christian Hadiwinoto, Hwee Tou Ng and Wee Chung Gan

*Do NLP Models Know Numbers? Probing Numeracy in Embeddings*
Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh and Matt Gardner

*A Split-and-Recombine Approach for Follow-up Query Analysis*
Qian Liu, Bei Chen, Haoyan Liu, Jian-Guang LOU, Lei Fang, Bin Zhou and Dongmei Zhang

*Text2Math: End-to-end Parsing Text into Math Expressions*
Yanyan Zou and Wei Lu

*Editing-Based SQL Query Generation for Cross-Domain Context-Dependent Questions*
Rui Zhang, Tao Yu, Heyang Er, Sungrok Shim, Eric Xue, Xi Victoria Lin, Tianze Shi, Caiming Xiong, Richard Socher and Dragomir Radev

*Syntax-aware Multilingual Semantic Role Labeling*
Shexia He, Zuchao Li and Hai Zhao

*Cloze-driven Pretraining of Self-attention Networks*
Alexei Baevski, Sergey Edunov, Yinhan Liu, Luke Zettlemoyer and Michael Auli

*Bridging the Gap between Relevance Matching and Semantic Matching for Short Text Similarity Modeling*
Jinfeng Rao, Linqing Liu, Yi Tay, Wei Yang, Peng Shi and Jimmy Lin

*A Syntax-aware Multi-task Learning Framework for Chinese Semantic Role Labeling*
Qingrong Xia, Zhenghua Li and Min Zhang

*Transfer Fine-Tuning: A BERT Case Study*
Yuki Arase and Jun'ichi Tsujii

*Data-Anonymous Encoding for Text-to-SQL Generation*
Zhen Dong, Shizhao Sun, Hongzhi Liu, Jian-Guang Lou and Dongmei Zhang

*Capturing Argument Interaction in Semantic Role Labeling with Capsule Networks*
Xinchi Chen, Chunchuan Lyu and Ivan Titov

*Learning Programmatic Idioms for Scalable Semantic Parsing*
Srinivasan Iyer, Alvin Cheung and Luke Zettlemoyer

*JuICe: A Large Scale Distantly Supervised Dataset for Open Domain Context-based Code Generation*
Rajas Agashe, Srinivasan Iyer and Luke Zettlemoyer

*Model-based Interactive Semantic Parsing: A Unified Framework and A Text-to-SQL Case Study*
Ziyu Yao, Yu Su, Huan Sun and Wen-tau Yih

*Modeling Graph Structure in Transformer for Better AMR-to-Text Generation*
Jie Zhu, Junhui Li, Muhua Zhu, Longhua Qian, Min Zhang and Guodong Zhou

*Syntax-Aware Aspect Level Sentiment Classification with Graph Attention Networks*
Binxuan Huang and Kathleen Carley

*Learning Explicit and Implicit Structures for Targeted Sentiment Analysis*
Hao Li and Wei Lu

*Capsule Network with Interactive Attention for Aspect-Level Sentiment Classification*
Chunning Du, Haifeng Sun, Jingyu Wang, Qi Qi, Jianxin Liao, Tong Xu and Ming Liu

*Emotion Detection with Neural Personal Discrimination*
Xiabing Zhou, Zhongqing Wang, Shoushan Li, Guodong Zhou and Min Zhang

*Specificity-Driven Cascading Approach for Unsupervised Sentiment Modification*
Pengcheng Yang, Junyang Lin, Jingjing Xu, Jun Xie, Qi Su and Xu SUN

*LexicalAT: Lexical-Based Adversarial Reinforcement Training for Robust Sentiment Classification*
Jingjing Xu, Liang Zhao, Hanqi Yan, Qi Zeng, Yun Liang and Xu SUN

*Leveraging Structural and Semantic Correspondence for Attribute-Oriented Aspect Sentiment Discovery*
Zhe Zhang and Munindar Singh

*From the Token to the Review: A Hierarchical Multimodal approach to Opinion Mining*
Alexandre Garcia, Pierre Colombo, Florence d'Alché-Buc, Slim Essid and Chloé Clavel

*Shallow Domain Adaptive Embeddings for Sentiment Analysis*
Prathusha K Sarma, Yingyu Liang and William Sethares

*Domain-Invariant Feature Distillation for Cross-Domain Sentiment Classification*
Mengting Hu, Yike Wu, Shiwan Zhao, Honglei Guo, Renhong Cheng and Zhong Su

*A Novel Aspect-Guided Deep Transition Model for Aspect Based Sentiment Analysis*
Yunlong Liang, Fandong Meng, Jinchao Zhang, Jinan Xu, Yufeng Chen and Jie Zhou

*Human-Like Decision Making: Document-level Aspect Sentiment Classification via Hierarchical Reinforcement Learning*
Jingjing Wang, Changlong Sun, Shoushan Li, Jiancheng Wang, Luo Si, Min Zhang, Xiaozhong Liu and Guodong Zhou

*A Dataset of General-Purpose Rebuttal*
Matan Orbach, Yonatan Bilu, Ariel Gera, Yoav Kantor, Lena Dankin, Tamar Lavee, Lili Kotlerman, Shachar Mirkin, Michal Jacovi, Ranit Aharonov and Noam Slonim

*Rethinking Attribute Representation and Injection for Sentiment Classification*
Reinald Kim Amplayo

*A Knowledge Regularized Hierarchical Approach for Emotion Cause Analysis*
Chuang Fan, Hongyu Yan, Jiachen Du, Lin Gui, Lidong Bing, Min Yang, Ruifeng Xu and Ruibin Mao

*Automatic Argument Quality Assessment - New Datasets and Methods*
Assaf Toledo, Shai Gretz, Edo Cohen-Karlik, Roni Friedman, Elad Venezian, Dan Lahav, Michal Jacovi, Ranit Aharonov and Noam Slonim

*Fine-Grained Analysis of Propaganda in News Article*
Giovanni Da San Martino, Seunghak Yu, Alberto Barrón-Cedeño, Rostislav Petrov and Preslav Nakov

*Context-aware Interactive Attention for Multi-modal Sentiment and Emotion Analysis*
Dushyant Singh Chauhan, Md Shad Akhtar, Asif Ekbal and Pushpak Bhattacharyya

*Sequential Learning of Convolutional Features for Effective Text Classification*
Avinash Madasu and Vijjini Anvesh Rao

*The Role of Pragmatic and Discourse Context in Determining Argument Impact*
Esin Durmus, Faisal Ladhak and Claire Cardie

*Aspect-Level Sentiment Analysis Via Convolution over Dependency Tree*
Kai Sun, Richong Zhang, Samuel Mensah, Yongyi Mao and Xudong Liu

15:00–15:30 *Coffee Break*

15:30–16:18 *Session 11*

### Session 11A: Machine Translation and Multilinguality III

15:30–15:42 *Understanding Data Augmentation in Neural Machine Translation: Two Perspectives towards Generalization*
Guanlin Li, Lemao Liu, Guoping Huang, Conghui Zhu and Tiejun Zhao

15:42–15:54 *Simple and Effective Noisy Channel Modeling for Neural Machine Translation*
Kyra Yee, Yann Dauphin and Michael Auli

15:54–16:06 *MultiFiT: Efficient Multi-lingual Language Model Fine-tuning*
Julian Eisenschlos, Sebastian Ruder, Piotr Czapla, Marcin Kadras, Sylvain Gugger and Jeremy Howard

16:06–16:18 *Hint-Based Training for Non-Autoregressive Machine Translation*
Zhuohan Li, Zi Lin, Di He, Fei Tian, Tao QIN, Liwei WANG and Tie-Yan Liu

### Session 11B: Syntax, Parsing, and Linguistic Theories

15:30–15:42 *Working Hard or Hardly Working: Challenges of Integrating Typology into Neural Dependency Parsers*
Adam Fisch, Jiang Guo and Regina Barzilay

15:42–15:54 *Cross-Lingual BERT Transformation for Zero-Shot Dependency Parsing*
Yuxuan Wang, Wanxiang Che, Jiang Guo, Yijia Liu and Ting Liu

15:54–16:06 *Multilingual Grammar Induction with Continuous Language Identification*
Wenjuan Han, Ge Wang, Yong Jiang and Kewei Tu

16:06–16:18 *Quantifying the Semantic Core of Gender Systems*
Adina Williams, Damian Blasi, Lawrence Wolf-Sonkin, Hanna Wallach and Ryan Cotterell

### Session 11C: Sentiment and Social Media

15:30–15:42 *Perturbation Sensitivity Analysis to Detect Unintended Model Biases*
Vinodkumar Prabhakaran, Ben Hutchinson and Margaret Mitchell

15:42–15:54 *Automatically Inferring Gender Associations from Language*
Serina Chang and Kathy McKeown

15:54–16:06 *Reporting the Unreported: Event Extraction for Analyzing the Local Representation of Hate Crimes*
Aida Mostafazadeh Davani, Leigh Yeh, Mohammad Atari, Brendan Kennedy, Gwenyth Portillo Wightman, Elaine Gonzalez, Natalie Delong, Rhea Bhatia, Arineh Mirinjian, Xiang Ren and Morteza Dehghani

16:06–16:18 *Minimally Supervised Learning of Affective Events Using Discourse Relations*
Jun Saito, Yugo Murawaki and Sadao Kurohashi

### Session 11D: Information Extraction IV

15:30–15:42 *Event Detection with Multi-Order Graph Convolution and Aggregated Attention*
Haoran Yan, Xiaolong Jin, Xiangbin Meng, Jiafeng Guo and Xueqi Cheng

15:42–15:54 *Coverage of Information Extraction from Sentences and Paragraphs*
Simon Razniewski, Nitisha Jain, Paramita Mirza and Gerhard Weikum

15:54–16:06 *HMEAE: Hierarchical Modular Event Argument Extraction*
Xiaozhi Wang, Ziqi Wang, Xu Han, Zhiyuan Liu, Juanzi Li, Peng Li, Maosong Sun, Jie Zhou and Xiang Ren

16:06–16:18 *Entity, Relation, and Event Extraction with Contextualized Span Representations*
David Wadden, Ulme Wennberg, Yi Luan and Hannaneh Hajishirzi

**Poster and Demo Session 11: Discourse and Pragmatics, Linguistic Theories, Textual Inference, Question Answering, Summarization and Generation**

*Next Sentence Prediction helps Implicit Discourse Relation Classification within and across Domains*
Wei Shi and Vera Demberg

*Split or Merge: Which is Better for Unsupervised RST Parsing?*
Naoki Kobayashi, Tsutomu Hirao, Kengo Nakamura, Hidetaka Kamigaito, Manabu Okumura and Masaaki Nagata

*BERT for Coreference Resolution: Baselines and Analysis*
Mandar Joshi, Omer Levy, Luke Zettlemoyer and Daniel Weld

*Linguistic Versus Latent Relations for Modeling Coherent Flow in Paragraphs*
Dongyeop Kang and Eduard Hovy

*Event Causality Recognition Exploiting Multiple Annotators' Judgments and Background Knowledge*
Kazuma Kadowaki, Ryu Iida, Kentaro Torisawa, Jong-Hoon Oh and Julien Kloetzer

*What Part of the Neural Network Does This? Understanding LSTMs by Measuring and Dissecting Neurons*
Ji Xin, Jimmy Lin and Yaoliang Yu

*Quantity doesn't buy quality syntax with neural language models*
Marten van Schijndel, Aaron Mueller and Tal Linzen

*Higher-order Comparisons of Sentence Encoder Representations*
Mostafa Abdou, Artur Kulmizev, Felix Hill, Daniel M. Low and Anders Søgaard

*Text Genre and Training Data Size in Human-like Parsing*
John Hale, Adhiguna Kuncoro, Keith Hall, Chris Dyer and Jonathan Brennan

*Feature2Vec: Distributional semantic modelling of human property knowledge*
Steven Derby, Paul Miller and Barry Devereux

*Sunny and Dark Outside?! Improving Answer Consistency in VQA through Entailed Question Generation*
Arijit Ray, Karan Sikka, Ajay Divakaran, Stefan Lee and Giedrius Burachas

*GeoSQA: A Benchmark for Scenario-based Question Answering in the Geography Domain at High School Level*
Zixian Huang, Yulin Shen, Xiao Li, Yu'ang Wei, Gong Cheng, Lin Zhou, Xinyu Dai and Yuzhong Qu

*Revisiting the Evaluation of Theory of Mind through Question Answering*
Matthew Le, Y-Lan Boureau and Maximilian Nickel

*Multi-passage BERT: A Globally Normalized BERT Model for Open-domain Question Answering*
Zhiguo Wang, Patrick Ng, Xiaofei Ma, Ramesh Nallapati and Bing Xiang

*A Span-Extraction Dataset for Chinese Machine Reading Comprehension*
Yiming Cui, Ting Liu, Wanxiang Che, Li Xiao, Zhipeng Chen, Wentao Ma, Shijin Wang and Guoping Hu

*MICRON: Multigranular Interaction for Contextualizing RepresentatiON in Non-factoid Question Answering*
Hojae Han, Seungtaek Choi, Haeju Park and Seung-won Hwang

*Machine Reading Comprehension Using Structural Knowledge Graph-aware Network*
Delai Qiu, Yuanzhe Zhang, Xinwei Feng, Xiangwen Liao, Wenbin Jiang, Yajuan Lyu, Kang Liu and Jun Zhao

*Answering Conversational Questions on Structured Data without Logical Forms*
Thomas Mueller, Francesco Piccinno, Peter Shaw, Massimo Nicosia and Yasemin Altun

*Improving Answer Selection and Answer Triggering using Hard Negatives*
Sawan Kumar, shweta garg, Kartik Mehta and Nikhil Rasiwasia

*Can You Unpack That? Learning to Rewrite Questions-in-Context*
Ahmed Elgohary, Denis Peskov and Jordan Boyd-Graber

*Quoref: A Reading Comprehension Dataset with Questions Requiring Coreferential Reasoning*
Pradeep Dasigi, Nelson F. Liu, Ana Marasovic, Noah A. Smith and Matt Gardner

*Zero-shot Reading Comprehension by Cross-lingual Transfer Learning with Multilingual Language Representation Model*
Tsung-Yuan Hsu, Chi-Liang Liu and Hung-yi Lee

*QuaRTz: An Open-Domain Dataset of Qualitative Relationship Questions*
Oyvind Tafjord, Matt Gardner, Kevin Lin and Peter Clark

*Giving BERT a Calculator: Finding Operations and Arguments with Reading Comprehension*
Daniel Andor, Luheng He, Kenton Lee and Emily Pitler

*A Gated Self-attention Memory Network for Answer Selection*
Tuan Lai, Quan Hung Tran, Trung Bui and Daisuke Kihara

*Polly Want a Cracker: Analyzing Performance of Parroting on Paraphrase Generation Datasets*
Hong-Ren Mao and Hung-Yi Lee

*Query-focused Sentence Compression in Linear Time*
Abram Handler and Brendan O'Connor

*Generating Personalized Recipes from Historical User Preferences*
Bodhisattwa Prasad Majumder, Shuyang Li, Jianmo Ni and Julian McAuley

*Generating Highly Relevant Questions*
Jiazuo Qiu and Deyi Xiong

*Improving Neural Story Generation by Targeted Common Sense Grounding*
Huanru Henry Mao, Bodhisattwa Prasad Majumder, Julian McAuley and Garrison Cottrell

*Abstract Text Summarization: A Low Resource Challenge*
Shantipriya Parida and Petr Motlicek

*Generating Modern Poetry Automatically in Finnish*
Mika Hämäläinen and Khalid Alnajjar

*SUM-QE: a BERT-based Summary Quality Estimation Model*
Stratos Xenouleas, Prodromos Malakasiotis, Marianna Apidianaki and Ion Androutsopoulos

*An Empirical Comparison on Imitation Learning and Reinforcement Learning for Paraphrase Generation*
Wanyu Du and Yangfeng Ji

*Countering the Effects of Lead Bias in News Summarization via Multi-Stage Training and Auxiliary Losses*
Matt Grenander, Yue Dong, Jackie Chi Kit Cheung and Annie Louis

*Learning Rhyming Constraints using Structured Adversaries*
Harsh Jhamtani, Sanket Vaibhav Mehta, Jaime Carbonell and Taylor Berg-Kirkpatrick

*Question-type Driven Question Generation*
Wenjie Zhou, Minghua Zhang and Yunfang Wu

*Deep Reinforcement Learning with Distributional Semantic Rewards for Abstractive Summarization*
Siyao Li, Deren Lei, Pengda Qin and William Yang Wang

*Clause-Wise and Recursive Decoding for Complex and Cross-Domain Text-to-SQL Generation*
Dongjun Lee

*Do Nuclear Submarines Have Nuclear Captains? A Challenge Dataset for Commonsense Reasoning over Adjectives and Objects*
James Mullenbach, Jonathan Gordon, Nanyun Peng and Jonathan May

*Aggregating Bidirectional Encoder Representations Using MatchLSTM for Sequence Matching*
Bo Shao, Yeyun Gong, Weizhen Qi, Nan Duan and Xiaola Lin

*What Does This Word Mean? Explaining Contextualized Embeddings with Natural Language Definition*
Ting-Yun Chang and Yun-Nung Chen

*Pre-Training BERT on Domain Resources for Short Answer Grading*
Chul Sung, Tejas Dhamecha, Swarnadeep Saha, Tengfei Ma, Vinay Reddy and Rishi Arora

*WIQA: A dataset for "What if..." reasoning over procedural text*
Niket Tandon, Bhavana Dalvi, Keisuke Sakaguchi, Peter Clark and Antoine Bosselut

*Evaluating BERT for natural language inference: A case study on the CommitmentBank*
Nanjiang Jiang and Marie-Catherine de Marneffe

*Incorporating Domain Knowledge into Medical NLI using Knowledge Graphs*
Soumya Sharma, Bishal Santra, Abhik Jana, Santosh Tokala, Niloy Ganguly and Pawan Goyal

16:18–16:30 *Mini-Break*

**Thursday, November 7, 2019 (continued)**

16:30–17:24 *Session 12*

**Session 12A: Machine Translation and Multilinguality IV**

16:30–16:48 *The FLORES Evaluation Datasets for Low-Resource Machine Translation: Nepali–English and Sinhala–English*
Francisco Guzmán, Peng-Jen Chen, Myle Ott, Juan Pino, Guillaume Lample, Philipp Koehn, Vishrav Chaudhary and Marc'Aurelio Ranzato

16:48–17:06 *Mask-Predict: Parallel Decoding of Conditional Masked Language Models*
Marjan Ghazvininejad, Omer Levy, Yinhan Liu and Luke Zettlemoyer

17:06–17:24 *Learning to Copy for Automatic Post-Editing*
Xuancheng Huang, Yang Liu, Huanbo Luan, Jingfang Xu and Maosong Sun

**Session 12B: Lexical Semantics III**

16:30–16:48 *Exploring Human Gender Stereotypes with Word Association Test*
Yupei Du, Yuanbin Wu and Man Lan

16:48–17:06 [TACL] *Still a Pain in the Neck: Evaluating Text Representations on Lexical Composition*
Vered Shwartz and Ido Dagan

17:06–17:24 [TACL] *Where's My Head? Definition, Dataset and Models for Numeric Fused-Heads Identification and Resolution*
Yanai Elazar and Yoav Goldberg

**Session 12C: Generation III**

16:30–16:48 *A Modular Architecture for Unsupervised Sarcasm Generation*
Abhijit Mishra, Tarun Tater and Karthik Sankaranarayanan

16:48–17:06 *Generating Classical Chinese Poems from Vernacular Chinese*
Zhichao Yang, Pengshan Cai, Yansong Feng, Fei Li, Weijiang Feng, Elena Suet-Ying Chiu and hong yu

17:06–17:24 *Set to Ordered Text: Generating Discharge Instructions from Medical Billing Codes*
Litton J Kurisinkel and Nancy Chen

### Session 12D: Phonology, Word Segmentation, and Parsing

16:30–16:48 *Constraint-based Learning of Phonological Processes*
Shraddha Barke, Rose Kunkel, Nadia Polikarpova, Eric Meinhardt, Eric Bakovic and Leon Bergen

16:48–17:06 *Detect Camouflaged Spam Content via StoneSkipping: Graph and Text Joint Embedding for Chinese Character Variation Representation*
Zhuoren Jiang, Zhe Gao, Guoxiu He, Yangyang Kang, Changlong Sun, Qiong Zhang, Luo Si and Xiaozhong Liu

17:06–17:24 [TACL] *A Generative Model for Punctuation in Dependency Trees*
Xiang Lisa Li, Dingquan Wang and Jason Eisner

### Poster and Demo Session 12: Information Extraction, Text Mining and NLP Applications, Social Media and Computational Social Science, Sentiment Analysis and Argument Mining

*An Attentive Fine-Grained Entity Typing Model with Latent Type Representation*
Ying Lin and Heng Ji

*An Improved Neural Baseline for Temporal Relation Extraction*
Qiang Ning, Sanjay Subramanian and Dan Roth

*Improving Fine-grained Entity Typing with Entity Linking*
Hongliang Dai, Donghong Du, Xin Li and Yangqiu Song

*Combining Spans into Entities: A Neural Two-Stage Approach for Recognizing Discontiguous Entities*
Bailin Wang and Wei Lu

*Cross-Sentence N-ary Relation Extraction using Lower-Arity Universal Schemas*
Kosuke Akimoto, Takuya Hiraoka, Kunihiko Sadamasa and Mathias Niepert

*Gazetteer-Enhanced Attentive Neural Networks for Named Entity Recognition*
Hongyu Lin, Yaojie Lu, Xianpei Han, Le Sun, Bin Dong and Shanshan Jiang

*"A Buster Keaton of Linguistics": First Automated Approaches for the Extraction of Vossian Antonomasia*
Michel Schwab, Robert Jäschke, Frank Fischer and Jannik Strötgen

*Multi-Task Learning for Chemical Named Entity Recognition with Chemical Compound Paraphrasing*
Taiki Watanabe, Akihiro Tamura, Takashi Ninomiya, Takuya Makino and Tomoya Iwakura

*FewRel 2.0: Towards More Challenging Few-Shot Relation Classification*
Tianyu Gao, Xu Han, Hao Zhu, Zhiyuan Liu, Peng Li, Maosong Sun and Jie Zhou

*ner and pos when nothing is capitalized*
Stephen Mayhew, Tatiana Tsygankova and Dan Roth

*CaRB: A Crowdsourced Benchmark for Open IE*
Sangnie Bhardwaj, Samarth Aggarwal and Mausam Mausam

*Weakly Supervised Attention Networks for Entity Recognition*
Barun Patra and Joel Ruben Antony Moniz

*Revealing and Predicting Online Persuasion Strategy with Elementary Units*
Gaku Morio, Ryo Egawa and Katsuhide Fujita

*A Challenge Dataset and Effective Models for Aspect-Based Sentiment Analysis*
Qingnan Jiang, Lei Chen, Ruifeng Xu, Xiang Ao and Min Yang

*Learning with Noisy Labels for Sentence-level Sentiment Classification*
Hao Wang, Bing Liu, Chaozhuo Li, Yan Yang and Tianrui Li

*DENS: A Dataset for Multi-class Emotion Analysis*
Chen Liu, Muhammad Osama and Anderson De Andrade

*Multi-Task Stance Detection with Sentiment and Stance Lexicons*
Yingjie Li and Cornelia Caragea

*A Robust Self-Learning Framework for Cross-Lingual Text Classification*
Xin Dong and Gerard de Melo

*Learning to Flip the Sentiment of Reviews from Non-Parallel Corpora*
Canasai Kruengkrai

*Label Embedding using Hierarchical Structure of Labels for Twitter Classification*
Taro Miyazaki, Kiminobu Makino, Yuka Takei, Hiroki Okamoto and Jun Goto

*Interpretable Word Embeddings via Informative Priors*
Miriam Hurtado Bodell, Martin Arvidsson and Måns Magnusson

*Adversarial Removal of Demographic Attributes Revisited*
Maria Barrett, Yova Kementchedjhieva, Yanai Elazar, Desmond Elliott and Anders Søgaard

*A deep-learning framework to detect sarcasm targets*
Jasabanta Patro, Srijan Bansal and Animesh Mukherjee

*In Plain Sight: Media Bias Through the Lens of Factual Reporting*
Lisa Fan, Marshall White, Eva Sharma, Ruisi Su, Prafulla Kumar Choubey, Ruihong Huang and Lu Wang

*Incorporating Label Dependencies in Multilabel Stance Detection*
William Ferreira and Andreas Vlachos

*Investigating Sports Commentator Bias within a Large Corpus of American Football Broadcasts*
Jack Merullo, Luke Yeh, Abram Handler, Alvin Grissom II, Brendan O'Connor and Mohit Iyyer

*Charge-Based Prison Term Prediction with Deep Gating Network*
Huajie Chen, Deng Cai, Wei Dai, Zehui Dai and Yadong Ding

*Restoring ancient text using deep learning: a case study on Greek epigraphy*
Yannis Assael, Thea Sommerschield and Jonathan Prag

*Embedding Lexical Features via Tensor Decomposition for Small Sample Humor Recognition*
Zhenjie Zhao, Andrew Cattle, Evangelos Papalexakis and Xiaojuan Ma

*EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks*
Jason Wei and Kai Zou

*Neural News Recommendation with Multi-Head Self-Attention*
Chuhan Wu, Fangzhao Wu, Suyu Ge, Tao Qi, Yongfeng Huang and Xing Xie

*What Matters for Neural Cross-Lingual Named Entity Recognition: An Empirical Analysis*
Xiaolei Huang, Jonathan May and Nanyun Peng

*Telling the Whole Story: A Manually Annotated Chinese Dataset for the Analysis of Humor in Jokes*
Dongyu Zhang, Heting Zhang, Xikai Liu, Hongfei LIN and Feng Xia

*Generating Natural Anagrams: Towards Language Generation Under Hard Combinatorial Constraints*
Masaaki Nishino, Sho Takase, Tsutomu Hirao and Masaaki Nagata

*STANCY: Stance Classification Based on Consistency Cues*
Kashyap Popat, Subhabrata Mukherjee, Andrew Yates and Gerhard Weikum

*Cross-lingual intent classification in a low resource industrial setting*
Talaat Khalil, Kornel Kiełczewski, Georgios Christos Chouliaras, Amina Keldibek and Maarten Versteegh

*SoftRegex: Generating Regex from Natural Language Descriptions using Softened Regex Equivalence*
Jun-U Park, Sang-Ki Ko, Marco Cognetta and Yo-Sub Han

*Using Clinical Notes with Time Series Data for ICU Management*
Swaraj Khadanga, Karan Aggarwal, Shafiq Joty and Jaideep Srivastava

*Spelling-Aware Construction of Macaronic Texts for Teaching Foreign-Language Vocabulary*
Adithya Renduchintala, Philipp Koehn and Jason Eisner

*Towards Machine Reading for Interventions from Humanitarian-Assistance Program Literature*
Bonan Min, Yee Seng Chan, Haoling Qiu and Joshua Fasching

*RUN through the Streets: A New Dataset and Baseline Models for Realistic Urban Navigation*
Tzuf Paz-Argaman and Reut Tsarfaty

*Context-Aware Conversation Thread Detection in Multi-Party Chat*
Ming Tan, Dakuo Wang, Yupeng Gao, Haoyu Wang, Saloni Potdar, Xiaoxiao Guo, Shiyu Chang and Mo Yu

17:24–17:30 *Mini-Break*

17:30–18:00 *Best Paper Awards and Closing*

# Attending to Future Tokens for Bidirectional Sequence Generation

**Carolin Lawrence** and **Bhushan Kotnis** and **Mathias Niepert**
NEC Laboratories Europe
{carolin.lawrence, bhushan.kotnis, mathias.niepert}@neclab.eu

## Abstract

Neural sequence generation is typically performed token-by-token and left-to-right. Whenever a token is generated only previously produced tokens are taken into consideration. In contrast, for problems such as sequence classification, *bidirectional* attention, which takes both past *and* future tokens into consideration, has been shown to perform much better. We propose to make the sequence generation process bidirectional by employing special placeholder tokens. Treated as a node in a fully connected graph, a placeholder token can take past and future tokens into consideration when generating the actual output token. We verify the effectiveness of our approach experimentally on two conversational tasks where the proposed bidirectional model outperforms competitive baselines by a large margin.

## 1 Introduction

When generating an output sequence, neural network models typically produce one token at a time. At each generation step, only the already produced sequence is taken into account. However, future and not-yet-produced tokens can also be highly relevant when choosing the current token. The importance of attending to both past and future tokens is apparent in self-attention architectures such as the Transformer (Vaswani et al., 2017). The self-attention module of a Transformer network treats a sequence *bidirectionally* as a fully connected graph of tokens – when a token is produced all other tokens are taken into consideration. However, this requires the entire sequence to be known a priori and when a Transformer is used for sequence generation, the self-attention process only includes previously produced tokens (Vaswani et al. (2017); Radford et al. (2019); *inter alia*). But the *bidirectional* self-attention is a crucial property of the highly successful language model BERT (Devlin et al., 2018). During the pre-training procedure of BERT, a fraction of input tokens is randomly masked out and the training objective is to predict these masked tokens correctly. BERT can then be fine-tuned for various classification tasks. Unfortunately, BERT cannot be directly used for sequence generation because the bidirectional nature of the approach requires the entire sequence to be known beforehand.

Inspired by BERT's masking-based objective, we propose to start out with a sequence of placeholder tokens which are iteratively replaced by tokens from the output vocabulary to eventually generate the full output sequence. For an example see Figure 1. With this novel model component, the self-attention of a Transformer can take both past and future tokens into consideration, leading to *Bi*directional *S*equence generati*on* (BISON). Furthermore, it allows us to directly incorporate the pre-trained language model BERT and, to the best of our knowledge, for the first time directly fine-tune it for sequence generation.

BISON makes two major contributions which we investigate in turn. First, we explore different stochastic placeholder replacement strategies to determine, at training time, where to position the placeholder tokens. This is crucial as we need the BISON models to be exposed to a large number of heterogeneous placeholder configurations. Second, we explore several strategies for iteratively generating, at inference time, a complete output sequence from an initial sequence of placeholders.

We evaluate our bidirectional sequence generation approach on two conversational tasks. BISON outperforms both competitive baselines and state of the art neural network approaches on both datasets by a significant margin.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1–10,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | $\bar{p}$ | $\bar{p}$ | $\bar{p}$ | $\bar{p}$ | $\bar{p}$ | $\bar{p}$ | $\bar{p}$ | $\bar{p}$ | $\bar{p}$ | $\bar{p}$ | $\bar{p}$ | $\bar{p}$ | $\bar{p}$ |
| 2 | $\bar{p}$ | you | been | $\bar{p}$ | or | enrolled | in | an | $\bar{p}$ | degree | or | program | ? |
| 3 | Have | you | been | accepted | or | enrolled | in | an | accredited | degree | or | program | ? |

Figure 1: Example generation, going from a sequence of the placeholder token $\bar{p}$ (1), to an intermediate representation (2) and to the final output (3).

## 2 Sequence Generation with Transformers

For sequence-to-sequence tasks, an input sequence $x = x_1, x_2, \ldots, x_{|x|}$ is to be mapped to an output sequence $y = y_1, y_2, \ldots, y_{|y|}$ by some model $\pi_\theta$ with learnable parameters $\theta$. For neural models, this is typically done by first encoding the input sequence $x$ and then calling a decoder $t$ times to produce a sequence $y$ token-by-token, from left-to-right.

A popular choice for both encoder and decoder is the transformer (Vaswani et al., 2017). It takes a sequence of embedded tokens $s = s_1, s_2, \ldots, s_{|s|}$ and treats it as a fully connected graph over which a self-attention module is applied: for each token $s_t$ in the sequence it assigns a probabilistic attention score $a_t$ to every other token in the sentence. For the full mathematical details we refer the reader to (Vaswani et al., 2017).

Typically a transformer *encoder* is employed to encode $x$, whereas a transformer *decoder* is used to produce $y$. In contrast to the encoder, the decoder at time step $t$ only has access to previously produced tokens $s_{<t} = s_1, s_2, \ldots, s_{t-1}$. Consequently, the attention module cannot take possible future tokens into account when making its decision at time $t$. Additionally, in this encoder-decoder framework, there is a disconnect between input $x$ and output $y$ because the self-attention modules are applied to $x$ and $y$ in isolation before they are combined.

The latter weakness has been overcome in recent work (Radford et al., 2018; Wolf et al., 2019; Radford et al., 2019) by feeding the concatenation $s = x \oplus y$ to a transformer decoder. At training time, given the current token $s_t$, the transformer is trained to predict the next word $s_{t+1}$ via maximum likelihood estimation. At test time, the transformer is conditioned on $x$ and then produces the output $y$ token-by-token. But because the model is a transformer decoder, it is unable to take possible future tokens into account.

## 3 Bidirectional Sequence Generation

During sequence generation, we want to take both past and future tokens into account. More formally, at time $t$, we want to attend to both $s_1, \ldots, s_{t-1}$ as well as $s_{t+1}, \ldots, s_{|s|}$. To do this, we give the sequence $s = x \oplus y$, the concatenation of the sequences $x$ and $y$, to a Transformer encoder, rather than a decoder. Of course, at inference time $y$ is unknown. Thus, we propose to replace each token $y_j$ with a *placeholder token* $\bar{p}$. Since the model needs to be exposed to heterogeneous placeholder token configurations during training time, we introduce a *placeholder strategy* that replaces some tokens $y_j$ with placeholder tokens $\bar{p}$ at training time. Hence, during training, a sequence $y$ is replaced by a sequence $p = p_1, p_2, \ldots, p_{|y|}$, where a token $p_j$ is either the original token $y_j$ or the placeholder token $\bar{p}$. We introduce two placeholder strategies in the following section. At inference time, $p$ contains only placeholder tokens up to some pre-determined maximum sequence length.

With the placeholder strategy in place, a Transformer encoder is given the sequence $s = x \oplus p$ as input. The self-attention module then computes hidden representations $r_t$ of each token $s_t$ by attending to every other token in the sequence $s$. Because the output sequence is already present in the form of placeholder tokens both past tokens as well as future, not-yet-produced, tokens can be taken into consideration for every token $s_t$. Following the self-attention step, placeholder tokens are converted into a token from the output vocabulary with a language model (LM) classification layer, where for each placeholder $p_t$, its hidden representation $r_t$ is mapped to a distribution $d_t$ over the output vocabulary.

At training time, each output sequence token is fed the gold label and updates to the model $\pi_\theta$ are performed using stochastic gradient descent with a cross-entropy loss, i.e.

$$\mathcal{L}_{\pi_\theta} = -\frac{1}{M}\sum_{m=1}^{M}\sum_{j=1}^{|y|}\log \pi_\theta(p_j = y_j|s),$$

where $M$ is the size of a minibatch.

At inference time, the placeholder tokens can be replaced iteratively based on the probability distribution $d_t$ over the output vocabulary for each placeholder $p_t$. Different sequence generation strategies are outlined in Section 3.2.

### 3.1 Placeholder Replacement Strategy

At inference time, the output sequence starts out with a sequence of placeholder tokens. To introduce this notion at training time, we require a strategy that replaces some output sequence tokens $y_j$ with the placeholder token $\bar{p}$. The simplest approach would be to replace all output sequence tokens $y_j$ with the placeholder token $\bar{p}$. However, with this approach the model is never confronted with a sequence containing a mix of output sequence tokens and placeholder tokens.

Due to the exponential number of possible replacement configurations per given token sequence, we introduce probabilistic generative models that we can use to draw diverse sequence replacements. To this end, we model the decision whether to use placeholder or input tokens with probabilistic models, two of which we propose in the following.

**Bernoulli Random Variables (RV).** We model each position of the input sequence with a binary random variable with a fixed mean. For every input sequence and every position $i$ in the sequence, we draw from a Bernoulli variable with mean $\mu$ to decide whether to use $y_i$ or $\bar{p}$. The expected number of placeholder tokens in a sequence of length $|y|$ is $|y|\mu$ and the variance is $\sigma^2 = |y|\mu(1-\mu)$. The variance of this strategy is determined by the mean and, therefore, the probabilistic model has one tunable parameter $\mu$.

**Gaussian Random Variables.** The *number of placeholder tokens* of the input sequence $p$ can be seen as drawn from an unknown optimal Binomial distribution. We can approximate this Binomial with a normal distribution $\mathcal{N}(\mu, \sigma^2)$, where $\mu$ is the mean and $\sigma$ the standard deviation and they are considered hyperparameters. More formally, for every input sequence, we draw a value $P \sim \mathcal{N}(\mu, \sigma^2)$. Multiplied with the sequence length $|y|$, the nearest integer value $\lfloor(|y| \cdot P)\rceil$ is used as the number of placeholder tokens for the given sequence. The positions of the placeholder tokens in the sequence are then determined at random. The resulting probabilistic model's two parameters (mean $\mu$ and standard deviation $\sigma$) are treated as hyperparameters and are not updated during training. Being able to tune the variance parameter independently from the mean parameter might provide an advantage over the parameterization with Bernoulli RVs.

### 3.2 Sequence Generation Strategies

Starting with a sequence of placeholder tokens at inference time, it is possible to generate output token sequences in arbitrary order. We experiment with the following strategies. The distribution in all of these strategies are the distributions $d_t$ ($t = 1, ..., n$) for the placeholders over the output vocabulary. We use the term *uncover* to mean that an output token is generated for a placeholder token.

**One-step greedy.** In a single time step, all placeholder tokens are uncovered simultaneously by picking the most probable token from the output vocabulary for each placeholder.

**Highest probability.** Placeholders are replaced iteratively and the placeholder to be uncovered is the placeholder that assigns the highest probability to a token from the output vocabulary, indicating the model is the most sure about this token.

**Lowest entropy.** Placeholders are replaced iteratively and the placeholder to be uncovered is the placeholder that exhibits the lowest entropy over its output vocabulary distribution and the most likely token at this position is chosen. Intuitively, the lowest entropy indicates the position where the uncertainty of the model to decide between tokens of the output vocabulary is the lowest.

**Left-to-right.** Placeholders are replaced iteratively, moving from left-to-right and thus mimicking the typical writing style for English. Note that this approach still differs from the Transformer decoders because future tokens are considered via the placeholder representations.

**No look ahead.** To test whether future placeholders hold useful information, we consider an adversarial sequence generation strategy: Again we iteratively uncover placeholders from left-to-right, but we suppress all attention flows from future placeholders. This imitates the behaviour of

a transformer decoder but follows the idea of predicting a token on a placeholder, rather than predicting the next word as is typically done in transformer decoders. If this performs worse than left-to-right, there is indeed valuable information in future placeholder tokens.

## 4 Experiments

We conduct a series of experiments to explore BISON's behavior. First, we want to compare two token replacement strategies for training as well as the four generation strategies for inference. Second, we want to compare BISON to state of the art methods and investigate the impact of its ability to attend to future tokens.

### 4.1 Datasets

We run experiments on the two following conversational datasets.

**Goal-oriented SHARC (Saeidi et al., 2018).** SHARC is a dialogue, text-based question-answering dataset. Unlike many popular QA datasets, answers cannot simply be extracted from the text. Given a regulatory text, such as a text from the UK government's website, and a user scenario with corresponding question, it is necessary to interpret the text in the context of the specific user's needs. Before generating its final answer, a system may generate clarification questions. Finally, the system decides if the answer to the user's original question is "*Yes*", "*No*" or "*Irrelevant*" where the latter means the question cannot be answered with the given text.

We perform the evaluation with the official SHARC script. For a set of generated clarification questions, it computes BLEU $n$-gram scores for $n = 1, 2, 3, 4$ using a set of clarification question in the set of gold responses. In each step of the conversation, the model under evaluation generates an output token sequence. This output is automatically assigned to the category "*More*" if it is a clarification question, and to "*Yes*", "*No*", and "*Irrelevant*" otherwise. Since this is a classification task we can compute micro and macro accuracy for it. The final model is chosen using the highest BLEU-4 score on the development set.

The SHARC dataset has a hidden test set and, therefore, it is not feasible to evaluate our various model variants. Hence, we take 30 unique rule texts and their corresponding training examples from the training set. This leads to a new development set of 2,465 instances and leaves the official development set to be used as a test set here. Finally we submitted our best model to be evaluated on the hidden test set.

**Free-form DAILY DIALOG (Li et al., 2017).** DAILY DIALOG is a dataset of written conversations occurring in daily life. Following the authors of the corpus, we report BLEU $n$-gram scores for $n = 1, 2, 3, 4$ for the generated output sequences with respect to the given gold responses. We tokenize these responses equally to ensure a fair comparison.

### 4.2 BISON Settings

We implement BISON based on the BERT Pytorch code[1] and initialize with the pre-trained BERT model BERT-BASE-UNCASED (Devlin et al., 2018). Consequently we employ the same model architecture and tokenisation as (Devlin et al., 2018) resulting in a model with about 110M parameters. To remain compatible with the BERT model, we prepend each sequence with a [CLS] token and place a [SEP] token after the input context. Similarly, producing a second [SEP] token indicates the end of sequence generation. For input context of SHARC, we follow Saeidi et al. (2018) and use the concatenation of question, rule text, scenario and history. The input context for DAILY DIALOG is the concatenation of all previous utterances.

On the SHARC and DAILY DIALOG training sets we train for 20 and 40 epochs, respectively, which equates in each case to about $200k$ seen examples. As optimizer we used ADAM (Kingma and Ba, 2015) with $\beta_1 = 0.9$, $\beta_2 = 0.999$, a L2 weight decay of $0.01$ and a learning rate warm-up over the first 10% of training steps. As learning rates we consider both the pre-training learning rate of BERT $1e$-$4$ and the fine-tuning learning rate $3e$-$5$. On preliminary experiments $3e$-$5$ proved to be best for SHARC, whereas it is $1e$-$4$ for DAILY DIALOG. We set the batch size to 15. Finally, the maximum sequence generation length, is set to 50 for SHARC and to 100 for DAILY DIALOG, which was chosen based on values observed in the training data. As the maximum sequence length of the BERT model is 512, longer input sequences are truncated accordingly. For the main results, we employ the sequence generation strategy left-

[1] https://github.com/huggingface/pytorch-pretrained-BERT

to-right, which we found to work best. Later on we also report results for the other strategies.

For the Bernoulli RV approach, we test $\mu \in [0.2, 0.8]$ with increments of $0.1$. For the Gaussian RV approach, we test all possible combinations for the following hyperparmeters $\mu = \{0.4, 0.5, 0.6\}$ and $\sigma = \{0.3, 0.6, 0.9\}$. The best combination on the SHARC dev set is $\mu = 0.5, \sigma = 0.6$. It outperforms the best Bernoulli approach ($\mu = 0.7$) by 3.4 point in BLEU-4 score. Some Bernoulli experiments in fact only produced a very small number of clarification question, e.g. $\mu = 0.5$ only generated 9 clarification questions on the development set, whereas in the ground truth responses 846 clarification questions occur. This suggests that a high variance is important, as the Bernoulli setups all have a variance of $0.25$ or lower and our best Gaussian approach has a variance of $0.6$. We directly employ the Gaussian distribution with $\mu = 0.5, \sigma = 0.6$ on the DAILY DIALOG task.

### 4.3 Baselines

To measure the success of our proposed approach, we consider the following three baselines.

**Encoder-Decoder Transformer (E&D).** First, we compare our bidirectional encoder to a standard encoder-decoder Transformer where the decoder only has access to tokens produced so far to compute its self-attention. We use the implementation of OpenNMT (Klein et al., 2017) and employ the parameters suggested by them, but adjust the learning rate to $0.1$, which we found to work better for both datasets. Additionally, we increased the word and hidden dimension size to 768 and the number of attention heads to 12 to match the capacity of our model. Training ran for 50 epochs. Needing both an encoder and a decoder, this leads to a total of about 270M parameters.

**Encoder-Decoder Transformer with BERT (E&D+B).** The power of our bidirectional decoder stems from two advantages. First, we can initialize our model with the pre-trained BERT-BASE-UNCASED model. Second, the decoding process is bidirectional. It would be possible to transfer the first advantage to an encoder-decoder framework by using BERT embeddings. This is however only possible for the input sequence, because the bidirectionality of BERT requires the entire sequence to be available beforehand. Thus, we modify implementation of OpenNMT to use the BERT model as the encoder. The weights are frozen when training the decoder, which produced better results than allowing the gradients to also flow through the BERT model. Again, with both an encoder and decoder, this leads to a total of about 270M parameters.

| | Model | Micro Acc. | Macro Acc. | B-1 | B-4 |
|---|---|---|---|---|---|
| SHARC | E&D | 31.9 | 38.9 | 17.1 | 1.9 |
| | E&D+B | 54.7 | 60.4 | 24.3 | 4.3 |
| | GPT2 | 60.4 | 65.1 | 53.7 | 33.9 |
| | BISON | **64.9** | **68.8** | **61.8** | **46.2** |

Table 1: Results on the SHARC test set, averaged over 3 independent runs for GPT2 and BISON, reporting micro accuracy and macro accuracy in terms of the classification task and BLEU-1 and BLEU-4 on instances for which a clarification question was generated. E&D uses no language model pre-training.

| | Model | Micro Acc. | Macro Acc. | B-1 | B-4 |
|---|---|---|---|---|---|
| SHARC | E3 | **67.6** | **73.3** | 54.1 | 38.7 |
| | BISON | 66.9 | 71.6 | **58.8** | **44.3** |

Table 2: Results on the official hidden SHARC test set of our model compared to the best model on the leaderboard, E3 (Zhong and Zettlemoyer, 2019).

**GPT2.** Radford et al. (2019) present a transformer decoder, GPT2, trained as a language model on large amounts of monolingual text. Radford et al. (2019) showed that it is possible to perform various tasks in a zero-shot setting by priming the language model with an input and letting it generate further words greedily. This setup can be transferred to a supervised setting, where the model is fine-tuned to a dataset by using maximum likelihood estimation to increase the probability of the gold output sequence (Wolf et al., 2019). As the starting point for the supervised learning, we initialize the model with the pre-trained model GPT-2-117M released by Radford et al. (2019)[2] and then fine-tune. With 117M parameters, this model is comparable to our model. Unlike baseline 2, this setup can directly employ a pre-trained model as our approach can, but it is not bidirectional.

[2] https://github.com/openai/gpt-2

### 4.4 Results

We report the results of our approach, the various baselines, as well as the previous state-of-the-art (SOTA) scores where applicable in Table 1 and 2 for SHARC and in Table 3 for DAILY DIALOG.

On the SHARC dataset, we observe very poor BLEU-4 performance for the encoder-decoder Transformer (E&D), which is consistent with results from Saeidi et al. (2018), who could not get a LSTM-based network to work without an additional classification head. Adding BERT (E&D+B) slightly improves performance. By directly leveraging a pre-trained model, GPT2 outperforms the previous models by a large margin, reaching 33.9% on BLEU-4 and a micro accuracy of 60.4%. BISON is able to take future tokens into consideration and outperforms GPT2 by 12.3 percentage points in BLEU-4 and by 4.5 points in micro accuracy.

We submitted the best BISON model out of the random three of Table 1 to be evaluated on the hidden test set and report results in comparison to the best model on the leaderboard,[3] E3 (Zhong and Zettlemoyer, 2019) in Table 2. BISON outperforms E3 by 5.6 BLEU-4 points, while it is only slightly worse than E3 in terms of accuracy.

On the DAILY DIALOG dataset the information retrieval-based method (IR in Table 3) introduced by Li et al. (2017) is very strong and outperforms the best end-to-end model (E2E) (Luo et al., 2018) by over 16 percentage points in BLEU-4. The best end-to-end model is based on LSTMs and Luo et al. (2018) report performance increases when adding an attention module to their setup. The encoder-decoder transformer (E&D) outperforms this setup by over 2 percentage points in BLEU-4 and we conjecture that this is due to the transformer making more effective use of the attention principle. Adding BERT (E&D+B) does not help much for this dataset. But again we observe a large increase of performance when directly employing pre-trained models. GPT2 performs on par with the IR SOTA, achieving a BLEU-4 score of 19.4%. Again, BISON can outperform GPT2, here with a difference of 6.2 points in BLEU-4 and even larger increases in the other scores.

**Effect of bidirectionality.** To investigate that our model benefits from bidirectionality, we consider a setup where BISON isn't allowed to attend to future tokens during prediction (see Table 4). It causes a drop in BLEU-4 performance of about 25 points on the SHARC dataset and a drop of 10 points on the DAILY DIALOG dataset. This showcases that BISON during training has learnt to rely on the ability to attend to future tokens.

[3] https://sharc-data.github.io/leaderboard.html, 19 August 2019

| | Model | B-1 | B-2 | B-3 | B-4 |
|---|---|---|---|---|---|
| DAILY DIALOG | IR | - | 25.8 | 20.4 | 19.4 |
| | E2E | 14.2 | 5.7 | 3.8 | 2.8 |
| | E&D | 22.3 | 6.8 | 5.7 | 5.2 |
| | E&D+B | 26.1 | 7.3 | 6.0 | 5.5 |
| | GPT2 | 42.3 | 23.6 | 20.7 | 19.4 |
| | BISON | **54.9** | **32.6** | **28.0** | **25.6** |

Table 3: BLEU $n$-gram scores for $n = 1, 2, 3, 4$ on the DailyDialog test set, averaged over 3 independent runs for GPT2 and BISON. Models before the line do not make use of a pre-trained language model. IR (SOTA) (Li et al., 2017) and E2E (SOTA) (Luo et al., 2018) are, to the best of our knowledge, the best previously published scores for information retrieval and end-to-end approaches.

| | Model | Micro Acc. | Macro Acc. | B-1 | B-4 |
|---|---|---|---|---|---|
| SHARC | BISON | **64.9** | **68.8** | **61.8** | **46.2** |
| | past only | 64.3 | 67.4 | 35.0 | 21.3 |
| | Model | B-1 | B-2 | B-3 | B-4 |
| DD | BISON | **54.9** | **32.6** | **28.0** | **25.6** |
| | past only | 48.0 | 24.6 | 18.5 | 14.8 |

Table 4: Comparison of BISON to a setup where BISON isn't allowed to attend to future tokens, i.e. past only, for SHARC and DAILY DIALOG (DD).

**Effect of pre-trained model.** We are curious how big the effect of the pre-trained model is. Thus, instead of starting with the BERT-BASE-UNCASED weights, we initialize BISON with random weights drawn from a normal distribution with mean 0.0 and standard deviation of 0.02. Results are presented in Table 5 for SHARC and DAILY DIALOG. Even without a pre-trained language model, our approach can outperform the standard encoder-decoder transformer framework (E&D) on both datasets, although we had to increase the number of epochs for the SHARC dataset to 40. On the DAILY DIALOG task, we are even able to outperform GPT2. This demonstrates

| | Model | Micro Acc. | Macro Acc. | B-1 | B-4 |
|---|---|---|---|---|---|
| SHARC | E&D | 31.9 | 38.9 | 17.1 | 1.9 |
| | BISON | **52.9** | **57.4** | **21.9** | **2.3** |
| | Model | B-1 | B-2 | B-3 | B-4 |
| DD | E&D | 22.3 | 6.8 | 5.7 | 5.2 |
| | BISON | **46.3** | **27.0** | **23.6** | **22.4** |

Table 5: Best end-to-end models that do not use a pre-trained language model in comparison with BISON that uses randomly initialized weights for SHARC and DAILY DIALOG (DD), averaged over 3 runs.

| Strategy | SHARC | DAILY DIALOG |
|---|---|---|
| one step greedy | 22.9 | 9.3 |
| lowest entropy | 40.3 | 16.8 |
| highest probability | **50.9** | 16.4 |
| left-to-right | 46.2 | **23.8** |

Table 6: BLEU-4 using various sequence generation strategies for BISON on SHARC and DAILY DIALOG.

| Dataset | $\alpha^1$ | $\alpha^2$ | $\alpha^3$ |
|---|---|---|---|
| SHARC | 92.6±3.1 | 5.2±2.4 | 2.2±1.8 |
| DD | 97.0±2.5 | 2.3±2.3 | 0.7±0.4 |
| | | $\bar{\alpha}^2$ | $\bar{\alpha}^3$ |
| SHARC | - | 71.7±13.7 | 28.3±13.7 |
| DD | - | 70.2±14.5 | 29.8±14.5 |

Table 7: Average attention weights and standard deviation when predicting from left-to-right on both SHARC and DAILY DIALOG (DD) for different parts of the sequence, where $\alpha^1$ is for the input sequence $x$, $\alpha^2/\bar{\alpha}^2$ is for the already produced sequence $y$ and $\alpha^3/\bar{\alpha}^3$ is for the sequence of remaining placeholder tokens $p$. $\alpha^k$ are the normalized attention weights across all three parts, whereas $\bar{\alpha}^k$ normalizes over the second and third part.

the effectiveness of our approach in itself, free of any pre-trained language model.

**Effect of sequence generation strategies.** We present the different sequence generation strategies in Table 6. The best overall sequence generation strategy is to predict from left to right which achieves good results on both datasets. On the SHARC dataset the highest probability approach performs better than left-to-right. However, on DAILY DIALOG this approach is not as successful. This suggests that it might be worth selecting the best sequence generation strategy for each dataset individually. However, we hypothesize that left-to-right works consistently well due to the left-to-right nature of the English language. A brief experiment with a right-to-left strategy gave poor results.

## 5 Analysis

We believe that the placeholders capture sequential information present in the language model learned during pre-training. After running a transformer encoder where each position can attend to every other position, a placeholder token will have a probability distribution over the output vocabulary and this distribution is informed by all other tokens in input and output. Thus, a placeholder could be seen as a mixture of tokens with varying probabilities. As placeholders are subsequently uncovered, the other placeholders can update their distribution by taking the newly revealed token into consideration.

For example, in Figure 2, for the sentence"is the animal an endangered animal ?", while generating "endangered", the self-attention head pays attention to the next placeholder token, which in the next step is revealed to be "animal". While producing "endangered", the distribution for the next position already placed a high probability on "animal", thus the current token can take this into consideration and produces "endangered". Further heat maps demonstrating this can be found in the appendix.

The quantify this intuition, we measure the average attention score on various parts of the sequence. For this, we use the left-to-right prediction strategy. Thus, at time $t$, we can decompose our sequence into three parts: $s = x \oplus y \oplus p$, where $x$ is the input, $y$ the already produced sequence and $p$ the remaining sequence of placeholder tokens. For each attention head, we can decompose the attention probabilities into the three parts,

1. attention on the input text,
2. attention on the current word and already generated words (left of the current word),
3. attention on words that are yet to be generated (right of the current word).

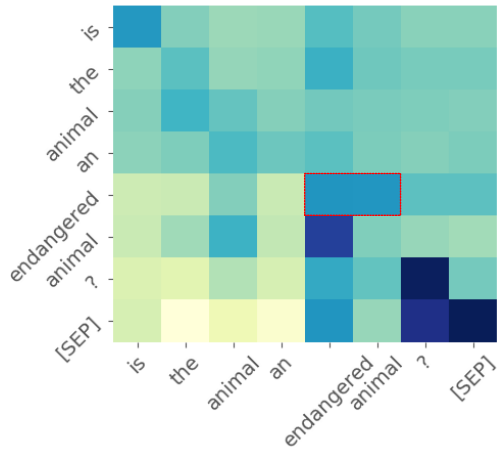


Figure 2: Heat map (darker hues indicate higher attention) that shows an example of where an attention head looks into the future while generating from left to right. Each row shows the attention over the output sequence for this row's placeholder token at that point in time. Word in previous rows have been produced already, whereas words of later rows still hold placeholder tokens. Thus the upper triangle of the matrix shows the attention that is paid to future tokens. The red square shows that while generating the token "endangered", the attention head already takes the next placeholder into account, which is revealed to be "animal" in the next step. Best viewed in color.

This is mathematically expressed as

$$a_t = a_{0:|x|} \oplus a_{|x|+1:|x|+t} \oplus a_{|x|+t+1,|s|},$$

where $|s|$ is the maximum possible sequence length. For each part we can calculate an average leading to three values, $a_t^1, a_t^2$ and $a_t^3$.

Averaged over all $T$ generation time steps and all $D$ data points, we can derive a score for each part $k$, $k = 1, 2, 3$ and each attention head $h$:

$$\alpha_h^k = \frac{1}{D}\frac{1}{T}\sum_{d=1}^{D}\sum_{t=1}^{T} a_{d,t}^k$$

Note that we use the attention heads in the final BERT self-attention layer. Averaging over all $H$ attention heads, $\alpha^k = \frac{1}{H}\sum_{h=1}^{H}\alpha_h^k$, leads to the results reported in Table 7 for both datasets. Unsurprisingly, we find that with scores of over 90% for both datasets the majority of the attention is focused on the first part, i.e. the conditioning input $x$ (see $\alpha^1$ in Table 7). The remaining attention is split between the already produced sequence ($\alpha^2$) and the future tokens ($\alpha^3$).

To directly compare the relationship within the sequence generation, we re-normalize over $\alpha^2$ and $\alpha^3$, leading to new values $\bar{\alpha}^2$ and $\bar{\alpha}^3$ (see Table 7). Here we can see that the past, already produced tokens are about twice as important as the future, not-yet-produced tokens. But with scores of just under 30% on both datasets, we see that a substantial amount of attention is also focused on the future, not-yet-produced tokens.

Interestingly, with a standard deviation of about 14%, the values of $\bar{\alpha}^2$ and $\bar{\alpha}^3$ vary strongly across the different attention heads. For example on the SHARC dataset, we find one attention head where only about 9% is focused on the future and another where it is about 64% and thus this attention head pays more attention to the future than the past. A graphical overview can be found in the appendix for both datasets.

## 6 Related Work

Transformers (Vaswani et al., 2017) model sequences as fully connected graphs and apply a bidirectional self-attention module where every token can attend to every other token. Because of this a Transformer is not restricted to sequential orderings. However, Vaswani et al. (2017); *inter alia* still restrict themselves to producing tokens from left-to-right and only allow a Transformer decoder to attend to previously produced tokens. Recently, several attempts have been made to lift the left-to-right restriction in Transformer or LSTM-based models (Gu et al., 2019; Stern et al., 2019; Welleck et al., 2019; Zhou et al., 2019), but in those approaches it is not possible to attend to future, not-yet-produced tokens.

Concurrently to our work, (Ghazvininejad et al., 2019) proposed a similar placeholder strategy approach for generating in the context of machine translation. However, they employ an encoder-decoder framework, whereas we only require an encoder, which more closely links input and output via a single shared attention module. Furthermore, they only consider uniform sampling of placeholders whereas we found that the higher variance, which we can control with the Gaussian random variable approach, leads to better results.

Bidirectionality is one of the crucial ingredients in the success of the recently proposed unsupervised language model BERT (Devlin et al., 2018). For this, Devlin et al. (2018) propose a Transformer *encoder* to take full advantage of the bidi-

rectional nature of the Transformer. Their resulting model, BERT, can directly be applied to various classification tasks but not to sequence generation tasks. Our approach shows how a Transformer encoder can be used for sequence generation and this allows us to directly incorporate BERT into our experiments.

GPT (Radford et al., 2018) and GPT2 (Radford et al., 2019) are both pre-trained language models that use a Transformer *decoder* instead, which can only attend to already produced tokens. For dialogue, the GPT model has been fine-tuned for the chit-chat dataset PersonaChat (Zhang et al., 2018) by Wolf et al. (2019). While GPT and GPT2 can immediately be used as a sequence generators, these models do not offer bidirectionality and they cannot attend to not-yet-produced tokens. Our bidirectional encoder for sequence generation can combine the best of both worlds.

## 7 Conclusion

We introduced bidirectional sequence generation by employing placeholders in the output sequence. These placeholder tokens are subsequently replaced by tokens of the output vocabulary. Crucially, this allows a transformer encoder to attend to both past and future, not-yet-produced token. Simply masking all placeholder tokens is not feasible. Instead we investigated two placeholder strategies, based on Bernoulli and Gaussian random variables. At prediction time, our approach is not restricted to produce the output sequence from left to right. However, this strategy proved to produce most consistent results in our experiments.

Our approach outperforms previous end-to-end approaches that do not make use of any pre-trained language models. In conjunction with the pre-trained language model BERT, our bidirectional sequence generation approach allows us to achieve new state-of-art results on both conversational tasks. In the future, we would like to apply our approach to other sequence generation tasks. Additionally, we wonder if a further performance increase could be achieved if the pre-training of BERT would employ our placeholder strategy.

## References


Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *ArXiv e-prints*, 1810.04805.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Constant-Time Machine Translation with Conditional Masked Language Models. *arXiv:1904.09324 [cs, stat]*. ArXiv: 1904.09324.

Jiatao Gu, Qi Liu, and Kyunghyun Cho. 2019. Insertion-based decoding with automatically inferred generation order. *CoRR*, abs/1902.01370.

Diederick P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, San Diego, CA, USA.

G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush. 2017. OpenNMT: Open-Source Toolkit for Neural Machine Translation. *ArXiv e-prints*, 1701.02810.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. Dailydialog: A manually labelled multi-turn dialogue dataset. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (IJCNLP)*, Taipei, Taiwan.

Liangchen Luo, Jingjing Xu, Junyang Lin, Qi Zeng, and Xu Sun. 2018. An auto-encoder matching model for learning utterance-level semantic dependency in dialogue generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving Language Understanding by Generative Pre-Training. Technical Report Technical report, OpenAI.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, and Dario Amodei. 2019. Language Models are Unsupervised Multitask Learners. Technical report, OpenAI.

Marzieh Saeidi, Max Bartolo, Patrick Lewis, Sameer Singh, Tim Rocktäschel, Mike Sheldon, Guillaume Bouchard, and Sebastian Riedel. 2018. Interpretation of natural language rules in conversational machine reading. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. 2019. Insertion transformer: Flexible sequence generation via insertion operations. *CoRR*, abs/1902.03249.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30 (NIPS)*.

Sean Welleck, Kianté Brantley, Hal Daumé III, and Kyunghyun Cho. 2019. Non-monotonic sequential text generation. *CoRR*, abs/1902.02192.

Thomas Wolf, Victor Sanh, Julien Chaumond, and Clement Delangue. 2019. TransferTransfo: A Transfer Learning Approach for Neural Network Based Conversational Agents. *ArXiv e-prints*, 1901.08149.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, Melbourne, Australia.

Victor Zhong and Luke Zettlemoyer. 2019. E3: Entailment-driven extracting and editing for conversational machine reading. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy.

Long Zhou, Jiajun Zhang, and Chengqing Zong. 2019. Synchronous bidirectional neural machine translation. *Transactions of the Association for Computational Linguistics*, 7:91–105.

# Attention is not not Explanation

**Sarah Wiegreffe***
School of Interactive Computing
Georgia Institute of Technology
saw@gatech.edu

**Yuval Pinter***
School of Interactive Computing
Georgia Institute of Technology
uvp@gatech.edu

## Abstract

Attention mechanisms play a central role in NLP systems, especially within recurrent neural network (RNN) models. Recently, there has been increasing interest in whether or not the intermediate representations offered by these modules may be used to explain the reasoning for a model's prediction, and consequently reach insights regarding the model's decision-making process. A recent paper claims that 'Attention is not Explanation' (Jain and Wallace, 2019). We challenge many of the assumptions underlying this work, arguing that such a claim depends on one's definition of explanation, and that testing it needs to take into account all elements of the model. We propose four alternative tests to determine when/whether attention can be used as explanation: a simple uniform-weights baseline; a variance calibration based on multiple random seed runs; a diagnostic framework using frozen weights from pretrained models; and an end-to-end adversarial attention training protocol. Each allows for meaningful interpretation of attention mechanisms in RNN models. We show that even when reliable adversarial distributions can be found, they don't perform well on the simple diagnostic, indicating that prior work does not disprove the usefulness of attention mechanisms for explainability.

## 1 Introduction

Attention mechanisms (Bahdanau et al., 2014) are nowadays ubiquitous in NLP, and their suitability for providing explanations for model predictions is a topic of high interest (Xu et al., 2015; Rocktäschel et al., 2015; Mullenbach et al., 2018; Thorne et al., 2019; Serrano and Smith, 2019). If they indeed offer such insights, many application areas would benefit by better understanding the internals of neural models that use attention as a means for, e.g., model debugging or architecture selection. A recent paper (Jain and Wallace, 2019) points to possible pitfalls that may cause researchers to misapply attention scores as explanations of model behavior, based on a premise that explainable attention distributions should be *consistent* with other feature-importance measures as well as *exclusive* given a prediction.[1] Its core argument, which we elaborate in §2, is that if alternative attention distributions exist that produce similar results to those obtained by the original model, then the original model's attention scores cannot be reliably used to "faithfully" explain the model's prediction. Empirically, the authors show that achieving such alternative distributions is easy for a large sample of English-language datasets.

We contend (§2.1) that while Jain and Wallace ask an important question, and raise a genuine concern regarding potential misuse of attention weights in explaining model decisions on English-language datasets, some key assumptions used in their experimental design leave an implausibly large amount of freedom in the setup, ultimately leaving practitioners without an applicable way for measuring the utility of attention distributions in specific settings.

We apply a more model-driven approach to this question, beginning (§3.2) with testing attention modules' **contribution** to a model by applying a simple baseline where attention weights are frozen to a uniform distribution. We demonstrate that for some datasets, a frozen attention distribution performs just as well as learned attention weights, concluding that randomly- or adversarially-perturbed distributions are not ev-

*Equal contributions.

[1] A preliminary version of our theoretical argumentation was published as a blog post on Medium at http://bit.ly/2OTzU4r. Following the ensuing online discussion, the authors uploaded a post-conference version of the paper to arXiv (v3) which addresses some of the issues in the post. We henceforth refer to this later version.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 11–20,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

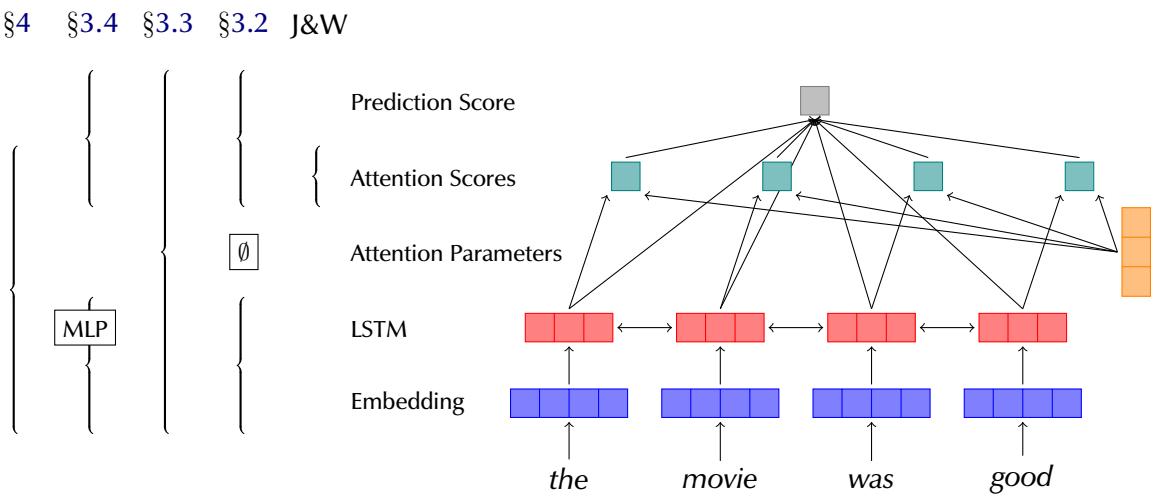


Figure 1: Schematic diagram of a classification LSTM model with attention, including the components manipulated or replaced in the experiments performed in Jain and Wallace (2019) and in this work (by section).

idence against attention as explanation in these cases. We next (§3.3) examine the **expected variance** in attention-produced weights by initializing multiple training sequences with different random seeds, allowing a better quantification of how much variance can be expected in trained models. We show that considering this background stochastic variation when comparing adversarial results with a traditional model allows us to better interpret adversarial results. In §3.4, we present a simple yet effective **diagnostic tool** which tests attention distributions for their usefulness by using them as frozen weights in a non-contextual multi-layered perceptron (MLP) architecture. The favorable performance of LSTM-trained weights provides additional support for the coherence of trained attention scores. This demonstrates a sense in which attention components indeed provide a meaningful model-agnostic interpretation of tokens in an instance.

In §4, we introduce a **model-consistent** training protocol for finding adversarial attention weights, correcting some flaws we found in the previous approach. We train a model using a modified loss function which takes into account the distance from an ordinarily-trained base model's attention scores in order to learn parameters for adversarial attention distributions. We believe these experiments are now able to support or refute a claim of faithful explainability, by providing a way for convincingly saying by construction that a plausible alternative 'explanation' can (or cannot) be constructed for a given dataset and model architecture. We find that while plausibly adversarial distributions of the consistent kind can indeed be found for the binary classification datasets in question, they are not as extreme as those found in the inconsistent manner, as illustrated by an example from the IMDB task in Figure 2. Furthermore, these outputs do not fare well in the diagnostic MLP, calling into question the extent to which we can treat them as equally powerful for explainability.

Finally, we provide a theoretical discussion (§5) on the definitions of interpretability and explainability, grounding our findings within the accepted definitions of these concepts.

Our four quantitative experiments are illustrated in Figure 1, where each bracket on the left covers the components in a standard RNN-with-attention architecture which we manipulate in each experiment. We urge NLP researchers to consider applying the techniques presented here on their models containing attention in order to evaluate its effectiveness at providing explanation. We offer our code for this purpose at `https://github.com/sarahwie/attention`.

## 2 Attention Might be Explanation

In this section, we briefly describe the experimental design of Jain and Wallace (2019) and look at the results they provide to support their claim that 'Attention is not explanation'. The authors select eight classification datasets, mostly binary, and two question answering tasks for their experiments (detailed in §3.1).

They first present a correlation analysis of attention scores and other interpretability measures.

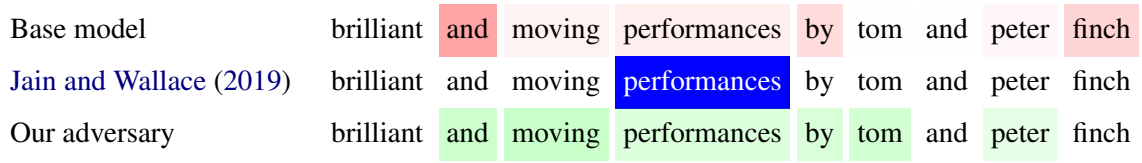


Figure 2: Attention maps for an IMDb instance (all predicted as positive with score $> 0.998$), showing that in practice it is difficult to learn a distant adversary which is consistent on all instances in the training set.

They find that attention is not strongly correlated with other, well-grounded feature importance metrics, specifically gradient-based and leave-one-out methods (which in turn correlate well with each other). This experiment evaluates the authors' claim of *consistency* – that attention-based methods of explainability cannot be valid if they do not correlate well with other metrics. We find the experiments in this part of the paper convincing and do not focus our analysis here. We offer our simple MLP diagnostic network (§3.4) as an additional way for determining validity of attention distributions, in a more *in vivo* setting.

Next, the authors present an adversarial search for alternative attention distributions which minimally change model predictions. To this end, they manipulate the attention distributions of trained models (which we will call **base** from now on) to discern whether alternative distributions exist for which the model outputs near-identical prediction scores. They are able to find such distributions, first by randomly permuting the base attention distributions on the test data during model inference, and later by adversarially searching for maximally different distributions that still produce a prediction score within $\epsilon$ of the base distribution. They use these experimental results as supporting evidence for the claim that attention distributions cannot be explainable because they are not *exclusive*. As stated, the lack of comparable change in prediction with a change in attention scores is taken as evidence for a lack of "faithful" explainability of the attention mechanism from inputs to output.

Notably, Jain and Wallace detach the attention distribution and output layer of their pretrained network from the parameters that compute them (see Figure 1), treating each attention score as a standalone unit independent of the model. In addition, they compute an independent adversarial distribution for each instance.

### 2.1 Main Claim

We argue that Jain and Wallace's counterfactual attention weight experiments do not advance their thesis, for the following reasons:

**Attention Distribution is not a Primitive.** From a modeling perspective, detaching the attention scores obtained by parts of the model (i.e. the attention mechanism) degrades the model itself. The base attention weights are not assigned arbitrarily by the model, but rather computed by an integral component whose parameters were trained alongside the rest of the layers; the way they work depends on each other. Jain and Wallace provide alternative distributions which may result in similar predictions, but in the process they remove the very linkage which motivates the original claim of attention distribution explainability, namely the fact that the model was *trained* to attend to the tokens it chose. A reliable adversary must take this consideration into account, as our setup in §4 does.

**Existence does not Entail Exclusivity.** On a more theoretical level, we hold that attention scores are used as providing *an* explanation; not *the* explanation. The final layer of an LSTM model may easily produce outputs capable of being aggregated into the same prediction in various ways, however the model still makes the choice of a specific weighting distribution using its trained attention component. This mathematically flexible production capacity is particularly evident in binary classifiers, where prediction is reduced to a single scalar, and an average instance (of e.g. the IMDB dataset) might contain 179 tokens, i.e. 179 scalars to be aggregated. This effect is greatly exacerbated when performed independently on each instance.[2] Thus, it is no surprise that Jain and Wal-

[2]Indeed, the most open-ended task, question answering over CNN data, produces considerable difficulty to manipulate its scores by random permutation (Figure 6e in Jain and Wallace (2019)). Similarly, the adversarial examples presented in Appendix C of the paper for the QA datasets select a different token of the correct word's type, which should not surprise us even under an LSTM assumption (encoder hidden

| Dataset | Avg. Length (tokens) | Train Size (neg/pos) | Test Size (neg/pos) |
|---|---|---|---|
| Diabetes | 1858 | 6381/1353 | 1295/319 |
| Anemia | 2188 | 1847/3251 | 460/802 |
| IMDb | 179 | 12500/12500 | 2184/2172 |
| SST | 19 | 3034/3321 | 863/862 |
| AgNews | 36 | 30000/30000 | 1900/1900 |
| 20News | 115 | 716/710 | 151/183 |

Table 1: Dataset statistics.

| Dataset | Attention (Base) | | Uniform |
|---|---|---|---|
| | Reported | Reproduced | |
| Diabetes | 0.79 | 0.775 | 0.706 |
| Anemia | 0.92 | 0.938 | 0.899 |
| IMDb | 0.88 | 0.902 | 0.879 |
| SST | 0.81 | 0.831 | 0.822 |
| AgNews | 0.96 | 0.964 | 0.960 |
| 20News | 0.94 | 0.942 | 0.934 |

Table 2: Classification F1 scores (1-class) on attention models, both as reported by Jain and Wallace and in our reproduction, and on models forced to use uniform attention over hidden states.

lace find what they are looking for given this degree of freedom.

In summary, due to the per-instance nature of the demonstration and the fact that model parameters have not been learned or manipulated directly, Jain and Wallace have **not shown the existence of an adversarial model** that produces the claimed adversarial distributions. Thus, we cannot treat these adversarial attentions as equally plausible or faithful explanations for model prediction. Additionally, they haven't provided a baseline of how much variation is to be expected in learned attention distributions, leaving the reader to question just how adversarial the found adversarial distributions are.

## 3 Examining Attention Distributions

In this section, we apply a careful methodological approach for examining the properties of attention distributions and propose alternatives. We begin by identifying the appropriate scope of the models' performance and variance, followed by implementing an empirical diagnostic technique which measures the model-agnostic usefulness of attention weights in capturing the relationship between inputs and output.

### 3.1 Experimental Setup

In order to make our many points in a succinct fashion as well as follow the conclusions drawn by Jain and Wallace, we focus on experimenting with the binary classification subset of their tasks, and on models with an LSTM architecture (Hochreiter and Schmidhuber, 1997), the only one the authors make firm conclusions on. Future work may extend our experiments to extractive tasks like question answering, as well as other attention-prone tasks, like seq2seq models.

We experiment on the following datasets: Stanford Sentiment Treebank (SST) (Socher et al., 2013), IMDB Large Movie Reviews Corpus (Maas et al., 2011), 20 NEWSGROUPS (hockey vs. baseball),[3] the AG NEWS Corpus,[4] and two prediction tasks from MIMIC-III ICD9 (Johnson et al., 2016): DIABETES and ANEMIA. The tasks are as follows: to predict positive or negative sentiment from sentences (SST) and movie reviews (IMDB), to predict the topic of news articles as either baseball (neg.) or hockey (pos.) in 20 NEWSGROUPS and either world (neg.) or business (pos.) in AG NEWS, to predict whether a patient is diagnosed with diabetes from their ICU discharge summary, and to predict whether the patient is diagnosed with acute (neg.) or chronic (pos.) anemia (both MIMIC-III ICD9). We use the dataset versions, including train-test split, provided by Jain and Wallace.[5] All datasets are in English.[6] Data statistics are provided in Table 1.

We use a single-layer bidirectional LSTM with $\tanh$ activation, followed by an additive attention layer (Bahdanau et al., 2014) and softmax prediction, which is equivalent to the LSTM setup of Jain and Wallace. We use the same hyperparameters found in that work to be effective in training, which we corroborated by reproducing its results to a satisfactory degree (see middle columns of Table 2). We refer to this architecture as the **main setup**, where training results in a **base model**.

Following Jain and Wallace, all analysis is performed on the test set. We report F1 scores on the positive class, and apply the same metrics they use for model comparison, namely Total Variation

states are typically affected by the input word to a noticeable degree).

[3] http://qwone.com/~jason/20Newsgroups/

[4] http://www.di.unipi.it/~gulli/AG_corpus_of_news_articles.html

[5] https://github.com/successar/AttentionExplanation

[6] We do not include the Twitter Adverse Drug Reactions (ADR) (Nikfarjam et al., 2015) dataset as the source tweets are no longer all available.

Distance (TVD) for comparing prediction scores $\hat{y}$ and Jensen-Shannon Divergence (JSD) for comparing weighting distributions $\alpha$:

$$\text{TVD}(\hat{y}_1, \hat{y}_2) = \frac{1}{2} \sum_{i=1}^{|\mathcal{Y}|} |\hat{y}_{1i} - \hat{y}_{2i}| \,;$$

$$\text{JSD}(\alpha_1, \alpha_2) = \frac{1}{2} \text{KL}[\alpha_1 \parallel \bar{\alpha}] + \frac{1}{2} \text{KL}[\alpha_2 \parallel \bar{\alpha}],$$

where $\bar{\alpha} = \frac{\alpha_1 + \alpha_2}{2}$.

### 3.2 Uniform as the Adversary

First, we test the validity of the classification tasks and datasets by examining whether attention is necessary in the first place. We argue that if attention models are not useful compared to very simple baselines, i.e. their parameter capacity is not being used, there is no point in using their outcomes for any type of explanation to begin with. We thus introduce a **uniform** model variant, identical to the main setup except that the attention distribution is frozen to uniform weights over the hidden states.

The results comparing this baseline with the base model are presented in Table 2. If attention was a necessary component for good performance, we would expect a large drop between the two rightmost columns. Somewhat surprisingly, for three of the classification tasks the attention layer appears to offer little to no improvement whatsoever. We conclude that these datasets, notably AG NEWS and 20 NEWSGROUPS, are not useful test cases for the debated question: attention is not explanation if you don't need it. We subsequently ignore the two News datasets, but keep SST, which we deem borderline.

### 3.3 Variance within a Model

We now test whether the variances observed by Jain and Wallace between trained attention scores and adversarially-obtained ones are unusual. We do this by repeating their analysis on eight models trained from the main setup using different initialization random seeds. The variance introduced in the attention distributions represents a baseline amount of variance that would be considered normal.

The results are plotted in Figure 3 using the same plane as Jain and Wallace's Figure 8 (with two of these reproduced as (e-f)). Left-heavy violins are interpreted as data classes for which the compared model produces attention distributions similar to the base model, and so having an adversary that manages to 'pull right' supports the argument that distributions are easy to manipulate. We see that SST distributions (c, e) are surprisingly robust to random seed change, validating our choice to continue examining this dataset despite its borderline F1 score. On the Diabetes dataset, the negative class is already subject to relatively arbitrary distributions from the different random seed settings (d), making the highly divergent results from the overly-flexible adversarial setup (f) seem less impressive. Our consistently-adversarial setup in §4 will further explore the difficulty of surpassing seed-induced variance between attention distributions.

### 3.4 Diagnosing Attention Distributions by Guiding Simpler Models

As a more direct examination of models, and as a complementary approach to Jain and Wallace (2019)'s measurement of backward-pass gradient flows through the model for gauging token importance, we introduce a post-hoc training protocol of a non-contextual model **guided** by pre-set weight distributions. The idea is to examine the prediction power of attention distributions in a 'clean' setting, where the trained parts of the model have no access to neighboring tokens of the instance. If pre-trained scores from an attention model perform well, we take this to mean they are helpful and consistent, fulfilling a certain sense of explainability. In addition, this setup serves as an effective diagnostic tool for assessing the utility of adversarial attention distributions: if such distributions are truly alternative, they should be equally useful as guides as their base equivalent, and thus perform comparably.

Our diagnostic model is created by replacing the main setup's LSTM and attention parameters with a token-level affine hidden layer with $\tanh$ activation (forming an MLP), and forcing its output scores to be weighted by a pre-set, per-instance distribution, during both training and testing. This setup is illustrated in Figure 4. The guide weights we impose are the following: **Uniform**, where we force the MLP outputs to be considered equally across each instance, effectively forming an unweighted baseline; **Trained MLP**, where we do not freeze the weights layer, instead allowing the

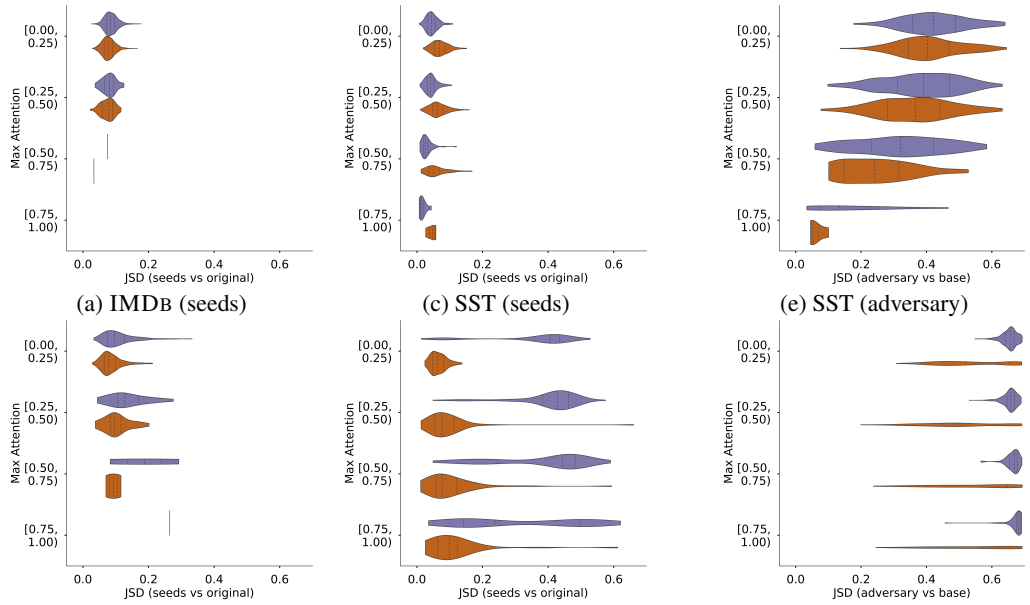


(a) IMDB (seeds) (c) SST (seeds) (e) SST (adversary)

(b) Anemia (seeds) (d) Diabetes (seeds) (f) Diabetes (adversary)

Figure 3: Densities of maximum JS divergences (x-axis) as a function of the max attention (y-axis) in each instance between the base distributions and: (a-d) models initialized on different random seeds; (e-f) models from a per-instance adversarial setup (replication of Figure 8a, 8c resp. in Jain and Wallace (2019)). In each max-attention bin, top (blue) is the negative-label instances, bottom (red) positive-label instances.

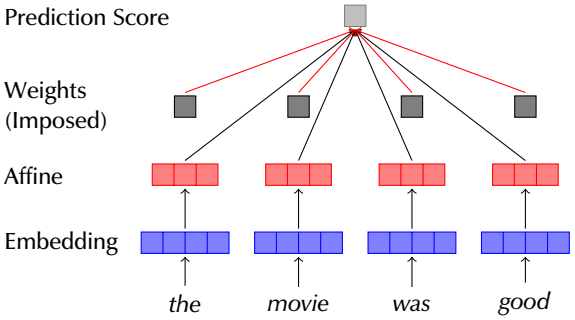


Figure 4: Diagram of the setup in §3.4 (except TRAINED MLP, which learns weight parameters).

| Guide weights | Diab. | Anemia | SST | IMDb |
|---|---|---|---|---|
| UNIFORM | 0.404 | 0.873 | 0.812 | 0.863 |
| TRAINED MLP | 0.699 | 0.920 | 0.817 | 0.888 |
| BASE LSTM | **0.753** | 0.931 | **0.824** | **0.905** |
| ADVERSARY (4) | 0.503 | **0.932** | 0.592 | 0.700 |

Table 3: F1 scores on the positive class for an MLP model trained on various weighting guides. For ADVERSARY, we set $\lambda \leftarrow 0.001$.

MLP to learn its own attention parameters;[7] **Base LSTM**, where we take the weights learned by the base LSTM model's attention layer; and **Adversary**, based on distributions found adversarially using the consistent training algorithm from §4 below (where their results will be discussed).

The results are presented in Table 3. The first important result, consistent across datasets, is that using pre-trained LSTM attention weights is better than letting the MLP learn them on its own, which is in turn better than the unweighted baseline. Comparing with results from §3.2, we see that this setup also outperforms the LSTM trained with uniform attention weights, suggesting that the attention module is more important than the word-level architecture for these datasets. These findings strengthen the case counter to the claim that attention weights are arbitrary: independent token-level models that have no access to contextual information find them useful, indicating that they encode some measure of token importance which is not model-dependent.

## 4 Training an Adversary

Having demonstrated three methods which test the meaningfulness of attention distributions as instruments of explainability with adequate control, we now propose a model-consistent training protocol for finding adversarial attention distributions through a coherent parameterization, which holds across all training instances. We believe this setup is able to advance the search for faithful explainability (see §5). Indeed, our results

[7] This is the same as Jain and Wallace's *average* setup.

will demonstrate that the extent to which a model-consistent adversary can be found varies across datasets, and that the dramatic reduction in degree of freedom compared to previous work allows for better-informed analysis.

**Model.** Given the base model $\mathcal{M}_\text{b}$, we train a model $\mathcal{M}_\text{a}$ whose explicit goal is to provide similar prediction scores for each instance, while distancing its attention distributions from those of $\mathcal{M}_\text{b}$. Formally, we train the adversarial model using stochastic gradient updates based on the following loss formula (summed over instances in the minibatch):

$$\mathcal{L}(\mathcal{M}_a, \mathcal{M}_b)^{(i)} = \text{TVD}(\hat{y}_a^{(i)}, \hat{y}_b^{(i)}) - \lambda \ \text{KL}(\boldsymbol{\alpha}_a^{(i)} \parallel \boldsymbol{\alpha}_b^{(i)}),$$

where $\hat{y}^{(i)}$ and $\boldsymbol{\alpha}^{(i)}$ denote predictions and attention distributions for an instance $i$, respectively.

$\lambda$ is a hyperparameter which we use to control the tradeoff between relaxing the prediction distance requirement (low TVD) in favor of more divergent attention distributions (high JSD), and vice versa. When this interaction is plotted on a two-dimensional axis, the shape of the plot can be interpreted to either support the 'attention is not explanation' hypothesis if it is convex (JSD is easily manipulable), or oppose it if it is concave (early increase in JSD comes at a high cost in prediction precision).

**Prediction performance.** By definition, our loss objective does not directly consider actual prediction performance. The TVD component pushes it towards the same score as the base model, but our setup does not ensure generalization from train to test. It would thus be interesting to inspect the extent of the implicit F1/TVD relationship. We report the highest F1 scores of models whose attention distributions diverge from the base, on average, by at least 0.4 in JSD, as well as their $\lambda$ setting and corresponding comparison metrics, in Table 4 (full results available in Appendix B). All F1 scores are on par with the original model results reported in Table 2, indicating the effectiveness of our adversarial models at imitating base model scores on the test sets.

**Adversarial weights as guides.** We next apply the diagnostic setup introduced in §3.4 by training a guided MLP model on the adversarially-trained attention distributions. The results, reported in the bottom line of Table 3, show that despite their local decision-imitation abilities, they are usually completely incapable of providing a non-contextual framework with useful guides.[8] We offer these results as evidence that adversarial distributions, even those obtained consistently for a dataset, deprive the underlying model from some form of understanding it gained over the data, one that it was able to leverage by tuning the attention mechanism towards preferring 'useful' tokens.

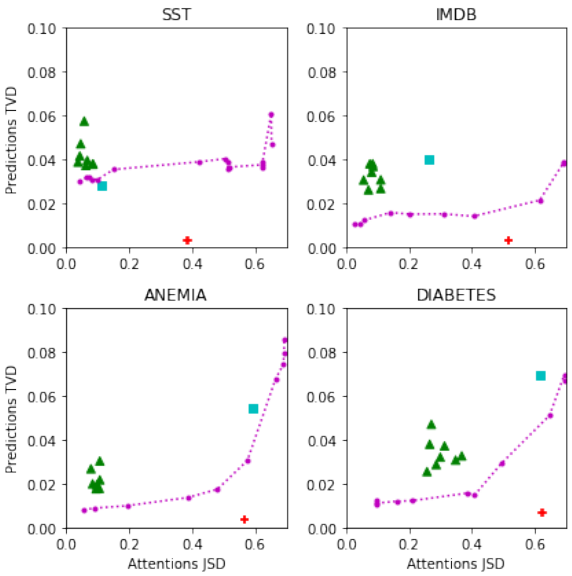


Figure 5: Averaged per-instance test set JSD and TVD from base model for each model variant. JSD is bounded at $\sim$ 0.693. ▲: random seed; ■: uniform weights; dotted line: our adversarial setup as $\lambda$ is varied; +: adversarial setup from Jain and Wallace (2019).

| Dataset | $\lambda$ | F1 (↑) | TVD (↓) | JSD (↑) |
|---|---|---|---|---|
| Diabetes | 2e-4 | 0.775 | 0.015 | 0.409 |
| Anemia | 5e-4 | 0.942 | 0.017 | 0.481 |
| SST | 5.25e-4 | 0.823 | 0.036 | 0.514 |
| IMDb | 8e-4 | 0.906 | 0.014 | 0.405 |

Table 4: Best-performing adversarial models with instance-average JSD $> 0.4$.

**TVD/JSD tradeoff.** In Figure 5 we present the levels of prediction variance (TVD) allowed

[8] We note the outlying result achieved on the Anemia dataset. This can be explained via the data distribution, which is heavily skewed towards positive examples (see Table 1 in the Appendix), together with the fact (conceded in Jain and Wallace (2019)'s section 4.2.1) that positive instances in detection datasets such as MIMIC tend to contain a handful of indicative tokens, making the particularly helpful distributions reached by a trained model hard to replace by an adversary. Together, this leads to the selected setting of $\lambda = 0.001$ producing average distributions substantially more similar to the base than in the other datasets (JSD $\sim$ 0.58 vs. $>$ 0.61) and thus more useful to the MLP setup.

by models achieving increased attention distance (JSD) on all four datasets. The convex shape of most curves does lend support to the claim that attention scores are easily manipulable; however the extent of this effect emerging from Jain and Wallace's per-instance setup is a considerable exaggeration, as seen by its position (+) well below the curve of our parameterized model set. Again, the SST dataset emerges as an outlier: not only can JSD be increased practically arbitrarily without incurring prediction variance cost, the uniform baseline (■) comes up under the curve, i.e. with a better adversarial score. We again include random seed initializations (▲) in order to quantify a baseline amount of variance.

TVD/JSD plots broken down by prediction class are available in Appendix C. In future work, we intend to inspect the potential of multiple adversarial attention models existing side-by-side, all distant enough from each other.

**Concrete Example.** Table 2 illustrates the difference between inconsistently-achieved adversarial heatmaps and consistently trained ones. Despite both adversaries approximating the desired prediction score to very high degree, the heatmaps show that Jain and Wallace's model has distributed all of the attention weight to an ad-hoc token, whereas our trained model could only distance itself from the base model distribution by so much, keeping multiple tokens in the $> 0.1$ score range.

## 5 Defining Explanation

The umbrella term of "Explainable AI" encompasses at least three distinct notions: *transparency*, *explainability*, and *interpretability*. Lipton (2016) categorizes transparency, or overall human understanding of a model, and post-hoc explainability as two competing notions under the umbrella of interpretability. The relevant sense of transparency, as defined by Lipton (2016) (§3.1.2), pertains to the way in which a specific portion of a model corresponds to a human-understandable construct (which Doshi-Velez and Kim (2017) refer to as a "cognitive chunk"). Under this definition, it should appear sensible of the NLP community to treat attention scores as a vehicle of (partial) transparency. Attention mechanisms do provide a look into the inner workings of a model, as they produce an easily-understandable weighting of hidden states.

Rudin (2018) defines explainability as simply a plausible (but not necessarily faithful) reconstruction of the decision-making process, and Riedl (2019) classifies explainable rationales as valuable in that they mimic what we as humans do when we rationalize past actions: we invent a story that plausibly justifies our actions, even if it is not an entirely accurate reconstruction of the neural processes that produced our behavior at the time. Distinguishing between interpretability and explainability as two separate notions, Rudin (2018) argues that interpretability is more desirable but more difficult to achieve than explainability, because it requires presenting humans with a big-picture understanding of the correlative relationship between inputs and outputs (citing the example of linear regression coefficients). Doshi-Velez and Kim (2017) break down interpretability into further subcategories, depending on the amount of human involvement and the difficulty of the task.

In prior work, Lei et al. (2016) train a model to simultaneously generate rationales and predictions from input text, using gold-label rationales to evaluate their model. Generally, many accept the notion of extractive methods such as Lei et al. (2016), in which explanations come directly from the input itself (as in attention), as plausible. Works such as Mullenbach et al. (2018) and Ehsan et al. (2019) use human evaluation to evaluate explanations; the former based on attention scores over the input, and the latter based on systems with additional rationale-generation capability. The authors show that rationales generated in a post-hoc manner increase user trust in a system.

Citing Ross et al. (2017), Jain and Wallace's requisite for attention distributions to be used as explanation is that there must only exist one or a few closely-related correct explanations for a model prediction. However, Doshi-Velez and Kim (2017) caution against applying evaluations and terminology broadly without clarifying task-specific explanation needs. If we accept the Rudin and Riedl definitions of explainability as providing a *plausible*, but not necessarily *faithful* rationale for model prediction, then the argument against attention mechanisms because they are not exclusive as claimed by Jain and Wallace is invalid, and human evaluation (which they do not consult) is necessary to evaluate the plausibility of generated rationales. Just because there exists another explanation does not mean that the one provided is false

or meaningless, and under this definition the existence of multiple different explanations is not necessarily indicative of the quality of a single one.

Jain and Wallace define attention and explanation as measuring the "responsibility" each input token has on a prediction. This aligns more closely with the more rigorous (Lipton, 2016, §3.1.1) definition of transparency, or Rudin (2018)'s definition of interpretability: human understanding of the model as a whole rather than of its respective parts. The ultimate question posed so far as 'is attention explanation?' seems to be: do high attention weights on certain elements in the input lead the model to make its prediction? This question is ultimately left largely unanswered by prior work, as we address in previous sections. However, under the given definition of transparency, the authors' exclusivity requisite is well-defined and we find value in their counterfactual framework as a concept – if a model is capable of producing multiple sets of diverse attention weights for the same prediction, then the relationship between inputs and outputs used to make predictions is not understood by attention analysis. This provides us with the motivation to implement the adversarial setup coherently and to derive and present conclusions from it. To this end, we additionally provide our §3.4 model to test the relationship between input tokens and output.

In the terminology of Doshi-Velez and Kim (2017), our proposed methods provide a *functionally-grounded* evaluation of attention as explanation, i.e. an analysis conducted on proxy tasks without human evaluation. We believe the proxies we have provided can be used to test the validity of attention as a form of explanation from the ground-up, based on the type of explanation one is looking for.

## 6 Attention is All you Need it to Be

Whether or not attention is explanation depends on the definition of explainability one is looking for: *plausible* or *faithful* explanations (or both). We believe that prior work focused on providing plausible rationales is not invalidated by Jain and Wallace's or our results. However, we have confirmed that adversarial distributions can be found for LSTM models in some classification tasks, as originally hypothesized by Jain and Wallace. This should provide pause to researchers who are looking to attention distributions for one true, faithful interpretation of the link their model has established between inputs and outputs. At the same time, we have provided a suite of experiments that researchers can make use of in order to make informed decisions about the quality of their models' attention mechanisms when used as explanation for model predictions.

We've shown that alternative attention distributions found via adversarial training methods perform poorly relative to traditional attention mechanisms when used in our diagnostic MLP model. These results indicate that trained attention mechanisms in RNNs on our datasets do in fact learn something meaningful about the relationship between tokens and prediction which cannot be easily 'hacked' adversarially.

We view the conditions under which adversarial distributions can actually be found in practice to be an important direction for future work. Additional future directions for this line of work include application on other tasks such as sequence modeling and multi-document analysis (NLI, QA); extension to languages other than English; and adding a human evaluation for examining the level of agreement with our measures. We also believe our work can provide value to theoretical analysis of attention models, motivating development of analytical methods to estimate the usefulness of attention as an explanation based on dataset and model properties.

## Acknowledgments

We thank Yoav Goldberg for preliminary comments on the idea behind the original Medium post. We thank the online community who participated in the discussion following the post, and particularly Sarthak Jain and Byron Wallace for their active engagement, as well as for the high-quality code they released which allowed fast reproduction and modification of their experiments. We thank Erik Wijmans for early feedback. We thank the members of the Computational Linguistics group at Georgia Tech for discussions and comments, particularly Jacob Eisenstein and Murali Raghu Babu. We thank the anonymous reviewers for many useful comments.

YP is a Bloomberg Data Science PhD Fellow.

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly

learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Finale Doshi-Velez and Been Kim. 2017. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*.

Upol Ehsan, Pradyumna Tambwekar, Larry Chan, Brent Harrison, and Mark O Riedl. 2019. Automated rationale generation: a technique for explainable ai and its effects on human perceptions. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*, pages 263–274. ACM.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Sarthak Jain and Byron C. Wallace. 2019. Attention is not Explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, Minneapolis, Minnesota. Association for Computational Linguistics.

Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3:160035.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 107–117, Austin, Texas. Association for Computational Linguistics.

Zachary C Lipton. 2016. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490*.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pages 142–150. Association for Computational Linguistics.

James Mullenbach, Sarah Wiegreffe, Jon Duke, Jimeng Sun, and Jacob Eisenstein. 2018. Explainable prediction of medical codes from clinical text. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1101–1111, New Orleans, Louisiana. Association for Computational Linguistics.

Azadeh Nikfarjam, Abeed Sarker, Karen O'connor, Rachel Ginn, and Graciela Gonzalez. 2015. Pharmacovigilance from social media: mining adverse drug reaction mentions using sequence labeling with word embedding cluster features. *Journal of the American Medical Informatics Association*, 22(3):671–681.

Mark O Riedl. 2019. Human-centered artificial intelligence and machine learning. *Human Behavior and Emerging Technologies*, 1(1):33–36.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*.

Andrew Slavin Ross, Michael C Hughes, and Finale Doshi-Velez. 2017. Right for the right reasons: training differentiable models by constraining their explanations. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 2662–2670. AAAI Press.

Cynthia Rudin. 2018. Please stop explaining black box models for high stakes decisions. *arXiv preprint arXiv:1811.10154*.

Sofia Serrano and Noah A. Smith. 2019. Is attention interpretable? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2931–2951, Florence, Italy. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2019. Generating token-level explanations for natural language inference. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 963–969, Minneapolis, Minnesota. Association for Computational Linguistics.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057.

# Practical Obstacles to Deploying Active Learning

**David Lowell**
Northeastern University
lowell.d@husky.neu.edu

**Zachary C. Lipton**
Carnegie Mellon University
zlipton@cmu.edu

**Byron C. Wallace**
Northeastern University
b.wallace@northeastern.edu

## Abstract

Active learning (AL) is a widely-used training strategy for maximizing predictive performance subject to a fixed annotation budget. In AL one iteratively selects training examples for annotation, often those for which the current model is most uncertain (by some measure). The hope is that active sampling leads to better performance than would be achieved under independent and identically distributed (i.i.d.) random samples. While AL has shown promise in retrospective evaluations, these studies often ignore practical obstacles to its use. In this paper we show that while AL may provide benefits when used with specific models and for particular domains, the benefits of current approaches do not generalize reliably across models and tasks. This is problematic because in practice one does not have the opportunity to explore and compare alternative AL strategies. Moreover, AL couples the training dataset with the model used to guide its acquisition. We find that subsequently training a *successor model* with an actively-acquired dataset does not consistently outperform training on i.i.d. sampled data. Our findings raise the question of whether the downsides inherent to AL are worth the modest and inconsistent performance gains it tends to afford.

## 1 Introduction

Although deep learning now achieves state-of-the-art results on a number of supervised learning tasks (Johnson and Zhang, 2016; Ghaddar and Langlais, 2018), realizing these gains requires large annotated datasets (Shen et al., 2018). This data dependence is problematic because labels are expensive. Several lines of research seek to reduce the amount of supervision required to achieve acceptable predictive performance, including semi-supervised (Chapelle et al., 2009), transfer (Pan and Yang, 2010), and *active learning* (AL) (Cohn et al., 1996; Settles, 2012).

In AL, rather than training on a set of labeled data sampled at i.i.d. random from some larger population, the learner engages the annotator in a cycle of learning, iteratively selecting training data for annotation and updating its model. *Pool-based* AL (the variant we consider) proceeds in rounds. In each, the learner applies a heuristic to score unlabeled instances, selecting the highest scoring instances for annotation.[1] Intuitively, by selecting training data cleverly, an active learner might achieve greater predictive performance than it would by choosing examples at random.

The more informative samples come at the cost of violating the standard i.i.d. assumption upon which supervised machine learning typically relies. In other words, the training and test data no longer reflect the same underlying data distribution. Empirically, AL has been found to work well with a variety of tasks and models (Settles, 2012; Ramirez-Loaiza et al., 2017; Gal et al., 2017a; Zhang et al., 2017; Shen et al., 2018). However, academic investigations of AL typically omit key real-world considerations that might overestimate its utility. For example, once a dataset is actively acquired with one model, it is seldom investigated whether this training sample will confer benefits if used to train a second model (vs i.i.d. data). Given that datasets often outlive learning algorithms, this is an important practical consideration.

[1] This may be done either deterministically, by selecting the top-$k$ instances, or stochastically, selecting instances with probabilities proportional to heuristic scores.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 21–30,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

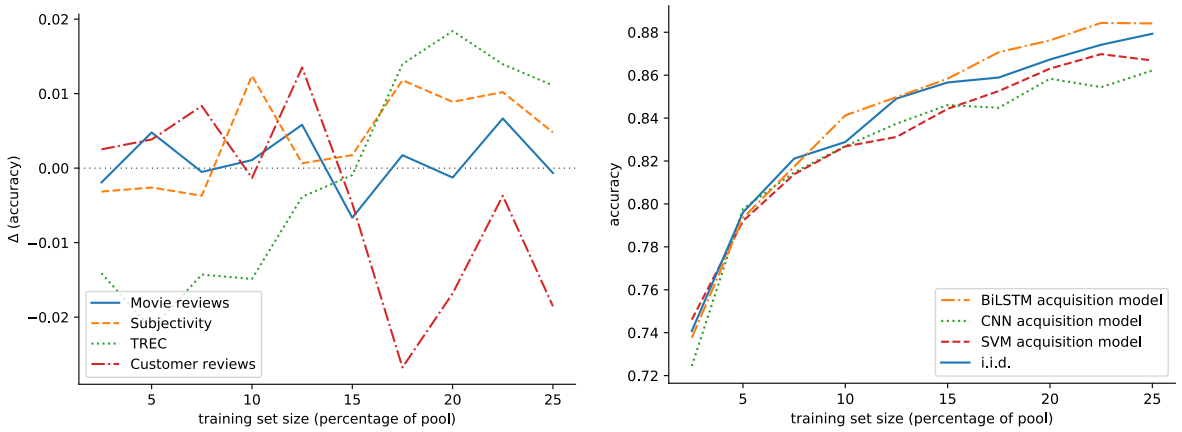


(a) Performance of AL relative to i.i.d. across corpora.

(b) Transferring actively acquired training sets.

Figure 1: We highlight practical issues in the use of AL. (a) AL yields inconsistent gains, relative to a baseline of i.i.d. sampling, across corpora. (b) Training a BiLSTM with training sets actively acquired based on the uncertainty of other models tends to result in worse performance than training on i.i.d. samples.

In contrast to experimental (retrospective) studies, in a real-world setting, an AL practitioner is not afforded the opportunity to retrospectively analyze or alter their scoring function. One would instead need to expend significant resources to validate that a given scoring function performs as intended for a particular model and task. This would require i.i.d. sampled data to evaluate the comparative effectiveness of different AL strategies. However, collection of such additional data would defeat the purpose of AL, i.e., obviating the need for a large amount of supervision. To confidently use AL in practice, one must have a reasonable belief that a given AL scoring (or *acquisition*) function will produce the desired results *before they deploy it* (Attenberg and Provost, 2011).

Most AL research does not explicitly characterize the circumstances under which AL may be expected to perform well. Practitioners must therefore make the implicit assumption that a given active acquisition strategy is likely to perform well under *any* circumstances. Our empirical findings suggest that this assumption is not well founded and, in fact, common AL algorithms behave inconsistently across model types and datasets, often performing no better than random (i.i.d.) sampling (1a). Further, while there is typically *some* AL strategy which outperforms i.i.d. random samples for a given dataset, *which* heuristic varies.

**Contributions**. We highlight important but often overlooked issues in the use of AL in practice. We report an extensive set of experimental results on classification and sequence tagging tasks that suggest AL typically affords only marginal performance gains at the somewhat high cost of non-i.i.d. training samples, which do not consistently transfer well to subsequent models.

## 2 The (Potential) Trouble with AL

We illustrate inconsistent comparative performance using AL. Consider Figure 1a, in which we plot the relative gains ($\Delta$) achieved by a BiLSTM model using a maximum-entropy active sampling strategy, as compared to the same model trained with randomly sampled data. Positive values on the $y$-axis correspond to cases in which AL achieves better performance than random sampling, 0 (dotted line) indicates no difference between the two, and negative values correspond to cases in which random sampling performs better than AL. Across the four datasets shown, results are decidedly mixed.

And yet realizing these equivocal gains using AL brings inherent drawbacks. For example, acquisition functions generally depend on the underlying model being trained (Settles, 2009, 2012), which we will refer to as the *acquisition model*. Consequently, the collected training data and the acquisition model are *coupled*. This coupling is problematic because manually labeled data tends to have a longer shelf life than models, largely because it is expensive to acquire. However, progress in machine learning is fast. Consequently, in many settings, an actively acquired dataset may remain in use (much) longer than the source model used to acquire it. In these cases, a few natural ques-

tions arise: How does a *successor* model $S$ fare, when trained on data collected via an acquisition model $A$? How does this compare to training $S$ on natively acquired data? How does it compare to training $S$ on i.i.d. data?

For example, if we use uncertainty sampling under a support vector machine (SVM) to acquire a training set $\mathcal{D}$, and subsequently train a Convolutional Neural Network (CNN) using $\mathcal{D}$, will the CNN perform better than it would have if trained on a dataset acquired via i.i.d. random sampling? And how does it perform compared to using a training corpus actively acquired using the CNN?

Figure 1b shows results for a text classification example using the Subjectivity corpus (Pang and Lee, 2004). We consider three models: a Bidirectional Long Short-Term Memory Network (BiLSTM) (Hochreiter and Schmidhuber, 1997), a Convolutional Neural Network (CNN) (Kim, 2014; Zhang and Wallace, 2015), and a Support Vector Machine (SVM) (Joachims, 1998). Training the LSTM with a dataset actively acquired using either of the other models yields predictive performance that is *worse* than that achieved under i.i.d. sampling. Given that datasets tend to outlast models, these results raise questions regarding the benefits of using AL in practice.

We note that in prior work, Tomanek and Morik (2011) also explored the transferability of actively acquired datasets, although their work did not consider modern deep learning models or share our broader focus on practical issues in AL.

## 3 Experimental Questions and Setup

We seek to answer two questions empirically: (1) How reliably does AL yield gains over sampling i.i.d.? And, (2) What happens when we use a dataset actively acquired using one model to train a different (successor) model? To answer these questions, we consider two tasks for which AL has previously been shown to confer considerable benefits: text classification and sequence tagging (specifically NER).[2]

To build intuition, our experiments address both linear models and deep networks more representative of the current state-of-the-art for these tasks. We investigate the standard strategy of acquiring data and training using a single model, and also the case of acquiring data using one model and subsequently using it to train a second model. Our experiments consider all possible (acquisition, successor) pairs among the considered models, such that the standard AL scheme corresponds to the setting in which the acquisition and successor models are same. For each pair $(A, S)$, we first simulate iterative active data acquisition with model $A$ to label a training dataset $\mathcal{D}_A$. We then train the successor model $S$ using $\mathcal{D}_A$.

In our evaluation, we compare the relative performance (accuracy or F1, as appropriate for the task) of the successor model trained with corpus $\mathcal{D}_A$ to the scores achieved by training on comparable amounts of native and i.i.d. sampled data. We simulate pool-based AL using labeled benchmark datasets by withholding document labels from the models. This induces a pool of unlabeled data $\mathcal{U}$. In AL, it is common to *warm-start* the acquisition model, training on some modest amount of i.i.d. labeled data $\mathcal{D}_w$ before using the model to score candidates in $\mathcal{U}$ (Settles, 2009) and commencing the AL process. We follow this convention throughout.

Once we have trained the acquisition model on the warm-start data, we begin the simulated AL loop, iteratively selecting instances for labeling and adding them to the dataset. We denote the dataset acquired by model $A$ at iteration $t$ by $\mathcal{D}_A^t$; $\mathcal{D}_A^0$ is initialized to $\mathcal{D}_w$ for all models (i.e., all values of $A$). At each iteration, the acquisition model is trained with $\mathcal{D}_A^t$. It then scores the remaining unlabeled documents in $\mathcal{U} \setminus \mathcal{D}_A^t$ according to a standard uncertainty AL heuristic. The top $n$ candidates $\mathcal{C}_A^t$ are selected for (simulated) annotation. Their labels are revealed and they are added to the training set: $\mathcal{D}_A^{t+1} \leftarrow \mathcal{D}_A^t \cup \mathcal{C}_A^t$. At the experiment's conclusion (time step $T$), each acquisition model $A$ will have selected a (typically distinct) subset of $\mathcal{U}$ for training.

Once we have acquired datasets from each acquisition model $\mathcal{D}_A$, we evaluate the performance of each possible successor model when trained on $\mathcal{D}_A$. Specifically, we train each successor model $S$ on the acquired data $\mathcal{D}_A^t$ for all $t$ in the range $[0, T]$, evaluating its performance on a held-out test set (distinct from $\mathcal{U}$). We compare the performance achieved in this case to that obtained using an i.i.d. training set of the same size.

We run this experiment ten times, averaging results to create summary learning curves, as shown

[2] Recent works have shown that AL is effective for these tasks even when using modern, neural architectures (Zhang et al., 2017; Shen et al., 2018), but do not address our primary concerns regarding replicability and transferability.

in Figure 1. All reported results, including i.i.d. baselines, are averages of ten experiments, each conducted with a distinct $\mathcal{D}_w$. These learning curves quantify the comparative performance of a particular model achieved using the same amount of supervision, but elicited under different acquisition models. For each model, we compare the learning curves of each acquisition strategy, including active acquisition using a *foreign* model and subsequent transfer, active acquisition without changing models (i.e., typical AL), and the baseline strategy of i.i.d. sampling.

## 4 Tasks

We now briefly describe the models, datasets, acquisition functions, and implementation details for the experiments we conduct with active learners for text classification (4.1) and NER (4.2).

### 4.1 Text Classification

**Models** We consider three standard models for text classification: Support Vector Machines (SVMs), Convolutional Neural Networks (CNNs) (Kim, 2014; Zhang and Wallace, 2015), and Bidirectional Long Short-Term Memory (BiLSTM) networks (Hochreiter and Schmidhuber, 1997). For SVM, we represent texts via sparse, TF-IDF bag-of-words (BoW) vectors. For neural models (CNN and BiLSTM), we represent each document as a sequence of word embeddings, stacked into an $l \times d$ matrix where $l$ is the length of the sentence and $d$ is the dimensionality of the word embeddings. We initialize all word embeddings with pre-trained GloVe vectors (Pennington et al., 2014).

We initialize vector representations for all words for which we do not have pre-trained embeddings uniformly at random. For the CNN, we impose a maximum sentence length of 120 words, truncating sentences exceeding this length and padding shorter sentences. We used filter sizes of 3, 4, and 5, with 128 filters per size. For BiLSTMs, we selected the maximum sentence length such that $90\%$ of sentences in $\mathcal{D}^t$ would be of equal or lesser length.[3] We trained all neural models using the Adam optimizer (Kingma and Ba, 2014), with a learning rate of $0.001$, $\beta_1 = 0.9$, $\beta_1 = 0.999$, and $\epsilon = 10^{-8}$.

**Datasets** We perform text classification experiments using four benchmark datasets. We reserve $20\%$ of each dataset (sampled at i.i.d. random) as test data, and use the remaining $80\%$ as the pool of unlabeled data $\mathcal{U}$. We sample $2.5\%$ of the remaining documents randomly from $\mathcal{U}$ for each $\mathcal{D}_w$. All models receive the same $\mathcal{D}_w$ for any given experiment.

- **Movie Reviews**: This corpus consists of sentences drawn from movie reviews. The task is to classify sentences as expressing positive or negative sentiment (Pang and Lee, 2005).
- **Subjectivity**: This dataset consists of statements labeled as either objective or subjective (Pang and Lee, 2004).
- **TREC**: This task entails categorizing questions into 1 of 6 categories based on the subject of the question (e.g., questions about people, locations, and so on) (Li and Roth, 2002). The TREC dataset defines standard train/test splits, but we generate our own for consistency in train/validation/test proportions across corpora.
- **Customer Reviews**: This dataset is composed of product reviews. The task is to categorize them as positive or negative (Hu and Liu, 2004).

### 4.2 Named Entity Recognition

**Models** We consider transfer between two NER models: Conditional Random Fields (CRF) (Lafferty et al., 2001) and Bidirectional LSTM-CNNs (BiLSTM-CNNs) (Chiu and Nichols, 2015).

For the CRF model we use a set of features including word-level and character-based embeddings, word suffix, capitalization, digit contents, and part-of-speech tags. The BiLSTM-CNN model[4] initializes word vectors to pre-trained GloVe vector embeddings (Pennington et al., 2014). We learn all word and character level features from scratch, initializing with random embeddings.

**Datasets** We perform NER experiments on the CoNLL-2003 and OntoNotes-5.0 English datasets. We used the standard test sets for both corpora, but merged training and validation sets to form $\mathcal{U}$. We initialize each $\mathcal{D}_w$ to $2.5\%$ of $\mathcal{U}$.

- **CoNLL-2003**: Sentences from Reuters news with words tagged as person, location, organization, or miscellaneous entities using an IOB

[3] Passing longer sentences to the BiLSTM degraded performance in preliminary experiments.

[4] Implementation of BiLSTM-CNN is based on https://github.com/asiddhant/Active-NLP.

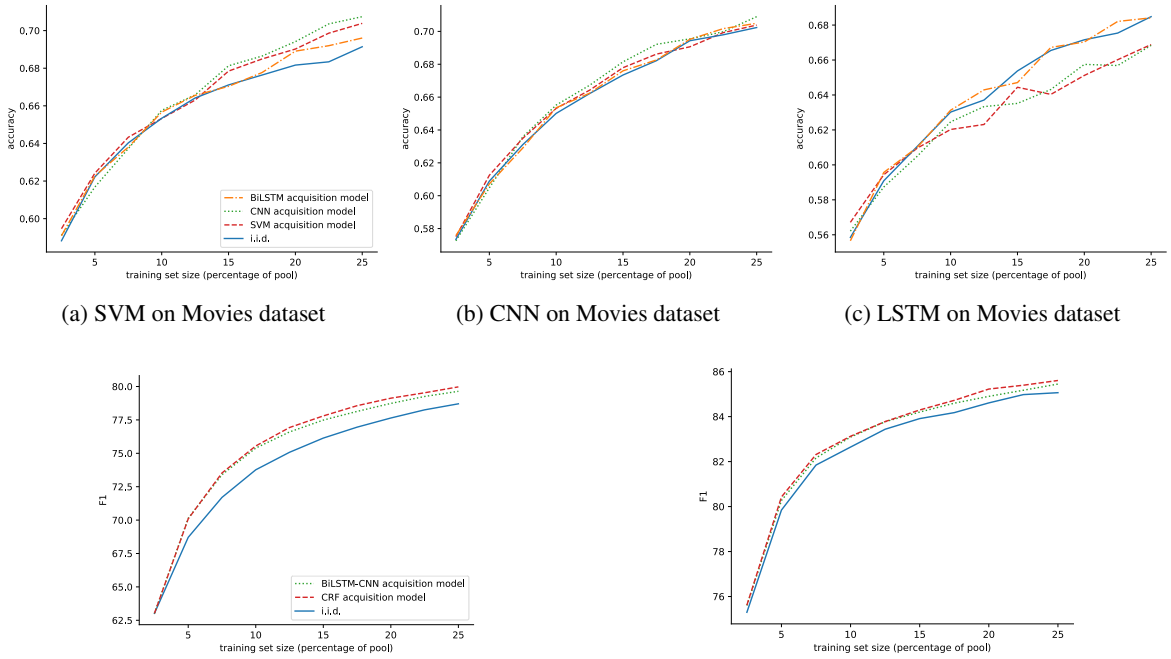


(a) SVM on Movies dataset

(b) CNN on Movies dataset

(c) LSTM on Movies dataset

(d) CRF on OntoNotes dataset

(e) BiLSTM-CNN on OntoNotes dataset

Figure 2: Sample learning curves for the text classification task on the Movie Reviews dataset and the NER task on the OntoNotes dataset using the maximum entropy acquisition function (we report learning curves for all models and datasets in the Appendix). Individual plots correspond to successor models. Each line corresponds to an acquisition model, with the blue line representing an i.i.d. baseline.

scheme (Tjong Kim Sang and De Meulder, 2003). The corpus contains 301,418 words.

- **OntoNotes-5.0**: A corpus of sentences drawn from a variety of sources including newswire, broadcast news, broadcast conversation, and web data. Words are categorized using eighteen entity categories annotated using the IOB scheme (Weischedel et al., 2013). The corpus contains 2,053,446 words.

### 4.3 Acquisition Functions

We evaluate these models using three common active learning acquisition functions: classical uncertainty sampling, query by committee (QBC), and Bayesian active learning by disagreement (BALD).

**Uncertainty Sampling** For text classification we use the entropy variant of uncertainty sampling, which is perhaps the most widely used AL heuristic (Settles, 2009). Documents are selected for annotation according to the function

$$\underset{\mathbf{x}\in\mathcal{U}}{\operatorname{argmax}} -\sum_{j} P(y_j|\mathbf{x}) \log P(y_j|\mathbf{x}),$$

where $\mathbf{x}$ are instances in the pool $\mathcal{U}$, $j$ indexes potential labels of these (we have elided the instance index here) and $P(y_j|\mathbf{x})$ is the predicted probability that $\mathbf{x}$ belongs to class $y_j$ (this estimate is implicitly conditioned on a model that can provide such estimates). For SVM, the equivalent form of this is to choose documents closest to the decision boundary.

For the NER task we use maximized normalized log-probability (MNLP) (Shen et al., 2018) as our AL heuristic, which adapts the least confidence heuristics to sequences by normalizing the log probabilities of predicted tag sequence by the sequence length. This avoids favoring selecting longer sentences (owing to the lower probability of getting the entire tag sequence right).

Documents are sorted in ascending order according to the function

$$\max_{y_1,\ldots,y_n} \frac{1}{n}\sum_{i=1}^{n} \log P(y_i|y_1,\ldots,y_{n-1},\mathbf{x})$$

Where the max over $y$ assignments denotes the most likely set of tags for instance $\mathbf{x}$ and $n$ is the sequence length. Because explicitly calculating the most likely tag sequence is computationally expensive, we follow (Shen et al., 2018) in using a greedy decoding (i.e., beam search with width 1) to determine the model's prediction.

**Text classification**

| | Acquisition model | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | 10% of pool | | | | 20% of pool | | |
| Successor | i.i.d. | SVM | CNN | LSTM | i.i.d. | SVM | CNN | LSTM |
| | | | | Movie reviews | | | | |
| SVM | 65.3 | **65.3** | 65.8 | 65.7 | 68.2 | **69.0** | 69.4 | 68.9 |
| CNN | 65.0 | 65.3 | **65.5** | 65.4 | 69.4 | 69.1 | **69.5** | 69.5 |
| LSTM | 63.0 | 62.0 | 62.5 | **63.1** | 67.2 | 65.1 | 65.8 | **67.0** |
| | | | | Subjectivity | | | | |
| SVM | 85.2 | **85.6** | 85.3 | 85.5 | 87.5 | **87.6** | 87.4 | 87.6 |
| CNN | 85.3 | 85.2 | **86.3** | 86.0 | 87.9 | 87.6 | **88.4** | 88.6 |
| LSTM | 82.9 | 82.7 | 82.7 | **84.1** | 86.7 | 86.3 | 85.8 | **87.6** |
| | | | | TREC | | | | |
| SVM | 68.5 | **68.3** | 66.8 | 68.5 | 74.1 | **74.7** | 73.2 | 74.3 |
| CNN | 70.9 | 70.5 | **69.0** | 70.0 | 76.1 | 77.7 | **77.3** | 78.0 |
| LSTM | 65.2 | 64.5 | 63.6 | **63.8** | 71.5 | 72.7 | 71.0 | **73.3** |
| | | | | Customer reviews | | | | |
| SVM | 68.8 | **70.5** | 70.3 | 68.5 | 73.6 | **74.2** | 72.9 | 71.1 |
| CNN | 70.6 | 70.9 | **71.7** | 68.2 | 74.1 | 74.5 | **74.8** | 71.5 |
| LSTM | 66.1 | 67.2 | 65.1 | **65.9** | 68.0 | 66.6 | 66.5 | **66.3** |

Table 1: Text classification accuracy, evaluated for each combination of acquisition and successor models using uncertainty sampling. Accuracies are reported for training sets composed of 10% and 20% of the document pool. Colors indicate performance relative to i.i.d. baselines: Blue indicates that a model fared better, red that it performed worse, and black that it performed the same.

**Named Entity Recognition**

| | Acquisition Model | | | | | |
|---|---|---|---|---|---|---|
| | | 10% of pool | | | 20% of pool | |
| Successor | i.i.d. | CRF | BiLSTM-CNN | i.i.d. | CRF | BiLSTM-CNN |
| | | | CoNLL | | | |
| CRF | 69.2 | **70.5** | 70.2 | 73.6 | **74.4** | 74.0 |
| BiLSTM-CNN | 87.4 | 87.4 | **87.8** | 89.1 | 89.6 | **89.6** |
| | | | OntoNotes | | | |
| CRF | 73.8 | **75.5** | 75.4 | 77.6 | **79.1** | 78.7 |
| BiLSTM-CNN | 82.6 | 83.1 | **83.1** | 84.6 | 85.2 | **84.9** |

Table 2: F1 measurements for the NER task, with training sets comprising 10% and 20% of the training pool.

**Query by Committee** For our QBC experiments, we use the bagging variant of QBC (Mamitsuka et al., 1998), in which a committee of $n$ models is assembled by sampling with replacement $n$ sets of $m$ documents from the training data ($\mathcal{D}^t$ at each $t$). Each model is then trained using a distinct resulting set, and the pool documents that maximize their disagreement are selected. We use 10 as our committee size, and set $m$ as equal to the number of documents in $\mathcal{D}^t$.

For the text classification task, we compute disagreement using Kullback-Leibler divergence (McCallum and Nigamy, 1998), selecting docu-

| Dataset | # Classes | # Documents | Examples per Class |
|---|---|---|---|
| Movie Reviews | 2 | 10662 | 5331, 5331 |
| Subjectivity | 2 | 10000 | 5000, 5000 |
| TREC | 6 | 5952 | 1300, 916, 95, 1288, 1344, 1009 |
| Customer Reviews | 2 | 3775 | 1368, 2407 |

Table 3: Text classification dataset statistics.

ments for annotation according to the function

$$\underset{\mathbf{x}\in\mathcal{U}}{\text{argmax}} \frac{1}{C}\sum_{c=1}^{C}\sum_{j} P_c(y_j|\mathbf{x}) \log \frac{P_c(y_j|\mathbf{x})}{P_C(y_j|\mathbf{x})}$$

where $\mathbf{x}$ are instances in the pool $\mathcal{U}$, $j$ indexes potential labels of these instances, and $C$ is the committee size. $P_c(y_j|\mathbf{x})$ is the probability that $\mathbf{x}$ belongs to class $y_j$ as predicted by committee member $c$. $P_C(y_j|\mathbf{x})$ represents the consensus probability that $\mathbf{x}$ belongs to class $y_j$, $\frac{1}{C}\sum_{c=1}^{C} P_c(y_j|\mathbf{x})$.

For NER, we compute disagreement using the average per word vote-entropy (Dagan and Engelson, 1995), selecting sequences for annotation which maximize the function

$$-\frac{1}{n}\sum_{i=1}^{n}\sum_{m}\frac{V(y_i,m)}{C}\log\frac{V(y_i,m)}{C}$$

where $n$ is the sequence length, $C$ is the committee size, and $V(y_i, m)$ is the number of committee members who assign tag $m$ to word $i$ in their most likely tag sequence. We do not apply the QBC acquisition function to the OntoNotes dataset, as training the committee for this larger dataset becomes impractical.

**Bayesian AL by Disagreement** We use the Monte Carlo variant of BALD, which exploits an interpretation of dropout regularization as a Bayesian approximation to a Gaussian process (Gal et al., 2017b; Siddhant and Lipton, 2018). This technique entails applying dropout at test time, and then estimating uncertainty as the disagreement between outputs realized via multiple passes through the model. We use the acquisition function proposed in (Siddhant and Lipton, 2018), which selects for annotation those instances that maximize the number of passes through the model that disagree with the most popular choice:

$$\underset{\mathbf{x}\in\mathcal{U}}{\text{argmax}}(1 - \frac{\text{count}(\text{mode}(y_{\mathbf{x}}^1, ..., y_{\mathbf{x}}^T))}{T})$$

where $\mathbf{x}$ are instances in the pool $\mathcal{U}$, $y_{\mathbf{x}}^i$ is the class prediction of the $i$th model pass on instance $x$, and $T$ is the number of passes taken through the model. Any ties are resolved using uncertainty sampling over the mean predicted probabilities of all $T$ passes.

In the NER task, agreement is measured across the entire sequence. Because this acquisition function relies on dropout, we do not consider it for non-neural models (SVM and CRF).

## 5 Results

We compare transfer between all possible (acquisition, successor) model pairs for each task. We report the performance of each model under all acquisition functions both in tables compiling results (Table 1 and Table 2 for classification and NER, respectively) and graphically via learning curves that plot predictive performance as a function of train set size (Figure 2).

We report additional results, including all learning curves (for all model pairs and for all tasks), and tabular results (for all acquisition functions) in the Appendix. We also provide in the Appendix plots resembling 1a for all (model, acquisition function) pairs that report the difference between performance under standard AL (in which acquisition and successor model are the same) and that under commensurate i.i.d. data, which affords further analysis of the gains offered by standard AL. For text classification tasks, we report accuracies; for NER tasks, we report F1.

To compare the learning curves, we select incremental points along the $x$-axis and report the performance at these points. Specifically, we report results with training sets containing 10% and 20% of the training pool.

## 6 Discussion

Results in Tables 1 and 2 demonstrate that standard AL — where the acquisition and successor models are one and the same — performs incon-

| | Successor | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Movie Reiews | | Subjectivity | | TREC | | Customer Reviews | |
| Acquisition Model | CNN | LSTM | CNN | LSTM | CNN | LSTM | CNN | LSTM |
| CNN | – | 0.961 | – | 0.968 | – | 0.988 | – | 0.973 |
| LSTM | 0.989 | – | 0.996 | – | 0.992 | – | 0.980 | – |
| SVM | 0.991 | 0.961 | 0.997 | 0.970 | 0.990 | 0.987 | 0.991 | 0.974 |

Table 4: Average Spearman's rank correlation coefficients (over five runs) of cosine distances between test set representations learned with native active learning and distances between those learned with transferred actively acquired datasets, at the end of the AL process. Uncertainty is used as the acquisition function in all cases.

sistently across text classification datasets. In 75% of all combinations of model, dataset, and training set size, there exists some acquisition function that outperforms i.i.d. data. This is consistent with the prior literature indicating the effectiveness of AL. However, when implementing AL in a real, live setting, a practitioner would choose a *single* acquisition function ahead of time. To accurately reflect this scenario, we must consider the performance of individual acquisition functions across multiple datasets. Results for individual AL strategies are more equivocal. In our reported classification datapoints, standard AL outperforms i.i.d. sampling in only a slight majority (60.9%) of cases.

AL thus seems to yield modest (though inconsistent) improvements over i.i.d. random sampling, but our results further suggest that this comes at an additional cost: the acquired dataset may not generalize well to new learners. Specifically, models trained on *foreign* actively acquired datasets tend to underperform those trained on i.i.d. datasets. We observe this most clearly in the classification task, where only a handful of (acquisition, successor, acquisition function) combinations lead to performance greater than that achieved using i.i.d. data. Specifically, only 37.5% of the tabulated data points representing dataset transfer (in which acquisition and successor models differ) outperform the i.i.d. baseline.

Results for NER are more favorable for AL. For this task we observe consistent improved performance versus the i.i.d. baseline in both standard AL data points and transfer data points. These results are consistent with previous findings on transferring actively acquired datasets for NER (Tomanek and Morik, 2011).

In standard AL for text classification, the only (model, acquisition function) pairs that we observe to produce better than i.i.d. results with any regularity are uncertainty with SVM or CNN, and BALD with CNN. When transferring actively acquired datasets, we do not observe consistently better than i.i.d. results with *any* combination of acquisition model, successor model, and acquisition function. The success of AL appears to depend very much on the dataset. For example, AL methods – both in the standard and acquisition/successor settings – perform much more reliably on the Subjectivity dataset than any other. In contrast, AL performs consistently poorly on the TREC dataset.

Our findings suggest that AL is brittle. During experimentation, we also found that performance often depends on factors that one may think are minor design decisions. For example, our setup largely resembles that of Siddhant and Lipton (2018), yet initially we observed large discrepancies in results. Digging into this revealed that much of the difference was due to our use of word2vec (Mikolov et al., 2013) rather than GloVe (Pennington et al., 2014) for word embedding initializations. That small decisions like this can result in relatively pronounced performance differences for AL strategies is disconcerting.

A key advantage afforded by neural models is representation learning. A natural question here is therefore whether the representations induced by the neural models differs as a function of the acquisition strategy. To investigate this, we measure pairwise distances between instances in the learned feature space after training. Specifically, for each test instance we calculate its cosine similarity to all other test instances, inducing a ranking. We do this in the three different feature spaces learned by the CNN and LSTM models, respectively, after sampling under the three acquisition models.

We quantify dissimilarities between the rankings induced under different representations via Spearman's rank correlation coefficients. We re-

peat this for all instances in the test set, and average over these coefficients to derive an overall similarity measure, which may be viewed as quantifying the similarity between learned feature spaces via average pairwise similarities within them. As reported in Table 4, despite the aforementioned differences in predictive performance, the learned representations seem to be similar. In other words, sampling under foreign acquisition models does not lead to notably different representations.

## 7 Conclusions

We extensively evaluated standard AL methods under varying model, domain, and acquisition function combinations for two standard NLP tasks (text classification and sequence tagging). We also assessed performance achieved when transferring an actively sampled training dataset from an acquisition model to a distinct successor model. Given the longevity and value of training sets and the frequency at which new ML models advance the state-of-the-art, this should be an anticipated scenario: Annotated data often outlives models.

Our findings indicate that AL performs unreliably. While a specific acquisition function and model applied to a particular task and domain may be quite effective, it is not clear that this can be predicted ahead of time. Indeed, there is no way to retrospectively determine the relative success of AL without collecting a relatively large quantity of i.i.d. sampled data, and this would undermine the purpose of AL in the first place. Further, even if such an i.i.d. sample were taken as a diagnostic tool early in the active learning cycle, relative success early in the AL cycle is not necessarily indicative of relative success later in the cycle, as illustrated by Figure 1a.

Problematically, even in successful cases, an actively sampled training set is linked to the model used to acquire it. We have found that training successor models with this set will often result in performance worse than that attained using an equivalently sized i.i.d. sample. Results are more favorable to AL for NER, as compared to text classification, which is consistent with prior work (Tomanek and Morik, 2011).

In short, the relative performance of individual active acquisition functions varies considerably over datasets and domains. While AL often does yield gains over i.i.d. sampling, these tend to be marginal and inconsistent. Moreover, this comes at a relatively steep cost: The acquired dataset may be disadvantageous for training subsequent models. Together these findings raise serious concerns regarding the efficacy of active learning in practice.

## 8 Acknowledgements

This work was supported in part by the Army Research Office (ARO), award W911NF1810328.

## References

Josh Attenberg and Foster Provost. 2011. Inactive learning?: difficulties employing active learning in practice. *ACM SIGKDD Explorations Newsletter*, 12(2):36–41.

Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. 2009. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542.

Jason PC Chiu and Eric Nichols. 2015. Named entity recognition with bidirectional lstm-cnns. *arXiv preprint arXiv:1511.08308*.

David A Cohn, Zoubin Ghahramani, and Michael I Jordan. 1996. Active learning with statistical models. *Journal of artificial intelligence research*, 4:129–145.

Ido Dagan and Sean P Engelson. 1995. Committee-based sampling for training probabilistic classifiers. In *Machine Learning Proceedings 1995*, pages 150–157. Elsevier.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. 2017a. Deep bayesian active learning with image data. In *International Conference on Machine Learning*, pages 1183–1192.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. 2017b. Deep bayesian active learning with image data. *CoRR*, abs/1703.02910.

Abbas Ghaddar and Phillippe Langlais. 2018. Robust lexical features for improved neural network named-entity recognition. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1896–1907. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer.

Rie Johnson and Tong Zhang. 2016. Supervised and semi-supervised text categorization using lstm for region embeddings. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 526–534. JMLR.org.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

Naoki Abe Hiroshi Mamitsuka et al. 1998. Query learning strategies using boosting and bagging. In *Machine learning: proceedings of the fifteenth international conference (ICML98)*, volume 1. Morgan Kaufmann Pub.

Andrew Kachites McCallum and Kamal Nigamy. 1998. Employing em and pool-based active learning for text classification. In *Proc. International Conference on Machine Learning (ICML)*, pages 359–367. Citeseer.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the ACL*.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the ACL*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Maria E. Ramirez-Loaiza, Manali Sharma, Geet Kumar, and Mustafa Bilgic. 2017. Active learning: an empirical study of common baselines. *Data Mining and Knowledge Discovery*, 31(2):287–313.

B. Settles. 2009. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison.

Burr Settles. 2012. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114.

Yanyao Shen, Hyokun Yun, Zachary C. Lipton, Yakov Kronrod, and Animashree Anandkumar. 2018. Deep active learning for named entity recognition. In *International Conference on Learning Representations*.

Aditya Siddhant and Zachary C Lipton. 2018. Deep bayesian active learning for natural language processing: Results of a large-scale empirical study. *arXiv preprint arXiv:1808.05697*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2003*, pages 142–147. Edmonton, Canada.

Katrin Tomanek and Katharina Morik. 2011. Inspecting sample reusability for active learning. In *Active Learning and Experimental Design workshop In conjunction with AISTATS 2010*, pages 169–181.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al. 2013. Ontonotes release 5.0 ldc2013t19. *Linguistic Data Consortium, Philadelphia, PA*.

Ye Zhang, Matthew Lease, and Byron C Wallace. 2017. Active discriminative text representation learning. In *AAAI*.

Ye Zhang and Byron Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*.

# Transfer Learning Between Related Tasks Using Expected Label Proportions

**Matan Ben Noach**[*†] and **Yoav Goldberg**[*‡]
[*]Computer Science Department, Bar-Ilan University, Ramat-Gan Israel
[†]Intel AI Lab, Petah-Tikva Israel
[‡]Allen Institute for Artificial Intelligence
matan.ben.noach@intel.com, yoav.goldberg@gmail.com

## Abstract

Deep learning systems thrive on abundance of labeled training data but such data is not always available, calling for alternative methods of supervision. One such method is expectation regularization (XR) (Mann and McCallum, 2007), where models are trained based on expected label proportions. We propose a novel application of the XR framework for transfer learning between related tasks, where knowing the labels of task A provides an estimation of the label proportion of task B. We then use a model trained for A to label a large corpus, and use this corpus with an XR loss to train a model for task B. To make the XR framework applicable to large-scale deep-learning setups, we propose a stochastic batched approximation procedure. We demonstrate the approach on the task of Aspect-based Sentiment classification, where we effectively use a sentence-level sentiment predictor to train accurate aspect-based predictor. The method improves upon fully supervised neural system trained on aspect-level data, and is also cumulative with LM-based pretraining, as we demonstrate by improving a BERT-based Aspect-based Sentiment model.

## 1 Introduction

Data annotation is a key bottleneck in many data driven algorithms. Specifically, deep learning models, which became a prominent tool in many data driven tasks in recent years, require large datasets to work well. However, many tasks require manual annotations which are relatively hard to obtain at scale. An attractive alternative is lightly supervised learning (Schapire et al., 2002; Jin and Liu, 2005; Chang et al., 2007; Graça et al., 2007; Quadrianto et al., 2009a; Mann and McCallum, 2010a; Ganchev et al., 2010; Hope and Shahaf, 2016), in which the objective function is supplemented by a set of domain-specific soft-constraints over the model's predictions on unlabeled data. For example, in *label regularization* (Mann and McCallum, 2007) the model is trained to fit the *true label proportions* of an unlabeled dataset. Label regularization is special case of *expectation regularization* (XR) (Mann and McCallum, 2007), in which the model is trained to fit the conditional probabilities of labels given features.

In this work we consider the case of correlated tasks, in the sense that knowing the labels for task A provides information on the expected label composition of task B. We demonstrate the approach using sentence-level and aspect-level sentiment analysis, which we use as a running example: knowing that a *sentence* has positive sentiment label (task A), we can expect that *most* aspects within this sentence (task B) will also have positive label. While this expectation may be noisy on the individual example level, it holds well in aggregate: given a *set* of positively-labeled sentences, we can robustly estimate the *proportion* of positively-labeled aspects within this set. For example, in a random set of positive sentences, we expect to find 90% positive aspects, while in a set of negative sentences, we expect to find 70% negative aspects. These proportions can be easily either guessed or estimated from a small set.

We propose a novel application of the XR framework for transfer learning in this setup. We present an algorithm (Sec 3.1) that, given a corpus labeled for task A (sentence-level sentiment), learns a classifier for performing task B (aspect-level sentiment) instead, *without* a direct supervision signal for task B. We note that the label information for task A is only used at training time. Furthermore, due to the stochastic nature of the estimation, the task A labels need not be fully accurate, allowing us to make use of *noisy* predictions which are assigned by an automatic classifier (Sections 3.1 and 4). In other words, given



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 31–42,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

a medium-sized sentiment corpus with sentence-level labels, and a large collection of *un-annotated* text from the same distribution, we can train an accurate aspect-level sentiment classifier.

The XR loss allows us to use task A labels for training task B predictors. This ability seamlessly integrates into other semi-supervised schemes: we can use the XR loss on top of a pre-trained model to fine-tune the pre-trained representation to the target task, and we can also take the model trained using XR loss and plentiful data and fine-tune it to the target task using the available small-scale annotated data. In Section 5.3 we explore these options and show that our XR framework improves the results also when applied on top of a pre-trained BERT-based model (Devlin et al., 2018).

Finally, to make the XR framework applicable to large-scale deep-learning setups, we propose a stochastic batched approximation procedure (Section 3.2). Source code is available at `https://github.com/MatanBN/XRTransfer`.

## 2 Background and Related Work

### 2.1 Lightly Supervised Learning

An effective way to supplement small annotated datasets is to use lightly supervised learning, in which the objective function is supplemented by a set of domain-specific soft-constraints over the model's predictions on unlabeled data. Previous work in lightly-supervised learning focused on training classifiers by using prior knowledge of label proportions (Jin and Liu, 2005; Chang et al., 2007; Musicant et al., 2007; Mann and McCallum, 2007; Quadrianto et al., 2009b; Liang et al., 2009; Ganchev et al., 2010; Mann and McCallum, 2010b; Chang et al., 2012; Wang et al., 2012; Zhu et al., 2014; Hope and Shahaf, 2016) or prior knowledge of features label associations (Schapire et al., 2002; Haghighi and Klein, 2006; Druck et al., 2008; Melville et al., 2009; Mohammady and Culotta, 2015). In the context of NLP, Haghighi and Klein (2006) suggested to use distributional similarities of words to train sequence models for part-of-speech tagging and a classified ads information extraction task. Melville et al. (2009) used background lexical information in terms of word-class associations to train a sentiment classifier. Ganchev and Das (2013); Wang and Manning (2014) suggested to exploit the bilingual correlations between a resource rich language and a resource poor language to train a classifier for the resource poor language in a lightly supervised manner.

### 2.2 Expectation Regularization (XR)

*Expectation Regularization* (XR) (Mann and McCallum, 2007) is a lightly supervised learning method, in which the model is trained to fit the conditional probabilities of labels given features. In the context of NLP, XR was used by Mohammady and Culotta (2015) to train twitter-user attribute prediction using hundreds of noisy distributional expectations based on census demographics. Here, we suggest using XR to train a target task (aspect-level sentiment) based on the output of a related source-task classifier (sentence-level sentiment).

**Learning Setup** The main idea of XR is moving from a fully supervised situation in which each data-point $x_i$ has an associated label $y_i$, to a setup in which sets of data points $U_j$ are associated with corresponding label proportions $\tilde{\mathbf{p}}_\mathbf{j}$ over that set.

Formally, let $X = \{x_1, x_2, \ldots, x_n\} \subseteq \mathcal{X}$ be a set of data points, $\mathcal{Y}$ be a set of $|\mathcal{Y}|$ class labels, $U = \{U_1, U_2, \ldots, U_m\}$ be a set of sets where $U_j \subseteq X$ for every $j \in \{1, 2, \ldots, m\}$, and let $\tilde{\mathbf{p}}_\mathbf{j} \in R^{|\mathcal{Y}|}$ be the label distribution of set $U_j$. For example, $\tilde{\mathbf{p}}_\mathbf{j} = \{.7, .2, .1\}$ would indicate that 70% of data points in $U_j$ are expected to have class 0, 20% are expected to have class 1 and 10% are expected to have class 2. Let $p_\theta(x)$ be a parameterized function with parameters $\theta$ from $\mathcal{X}$ to a vector of conditional probabilities over labels in $\mathcal{Y}$. We write $p_\theta(y|x)$ to denote the probability assigned to the $y$th event (the conditional probability of $y$ given $x$).

A typically objective when training on fully labeled data of $(x_i, y_i)$ pairs is to maximize likelihood of labeled data using the cross entropy loss,

$$L_{cross}(\theta) = -\sum_{i}^{n} \log p_\theta(y_i|x_i)$$

Instead, in XR our data comes in the form of pairs $(U_j, \tilde{\mathbf{p}}_\mathbf{j})$ of sets and their corresponding expected label proportions, and we aim to optimize $\theta$ to fit the label distribution $\tilde{\mathbf{p}}_\mathbf{j}$ over $U_j$, for all $j$.

**XR Loss** As counting the number of predicted class labels over a set $U$ leads to a non-differentiable objective, Mann and McCallum (2007) suggest to relax it and use instead the

model's posterior distribution $\hat{\mathbf{p}}_\mathbf{j}$ over the set:

$$\hat{\mathbf{q}}_\mathbf{j}(y) = \sum_{x \in U_j} p_\theta(y|x) \quad (1)$$

$$\hat{\mathbf{p}}_\mathbf{j}(y) = \frac{\hat{\mathbf{q}}_\mathbf{j}(y)}{\sum_{y'} \hat{\mathbf{q}}_\mathbf{j}(y')} \quad (2)$$

where $\mathbf{q}(y)$ indicates the $y$th entry in $\mathbf{q}$. Then, we would like to set $\theta$ such that $\hat{\mathbf{p}}_\mathbf{j}$ and $\tilde{\mathbf{p}}_\mathbf{j}$ are close. Mann and McCallum (2007) suggest to use KL-divergence for this. KL-divergence is composed of two parts:

$$D_{KL}(\tilde{\mathbf{p}}_\mathbf{j} || \hat{\mathbf{p}}_\mathbf{j}) = -\tilde{\mathbf{p}}_\mathbf{j} \cdot \log \hat{\mathbf{p}}_\mathbf{j} + \tilde{\mathbf{p}}_\mathbf{j} \cdot \log \tilde{\mathbf{p}}_\mathbf{j}$$

$$= H(\tilde{\mathbf{p}}_\mathbf{j}, \hat{\mathbf{p}}_\mathbf{j}) - H(\tilde{\mathbf{p}}_\mathbf{j})$$

Since $H(\tilde{\mathbf{p}}_\mathbf{j})$ is constant, we only need to minimize $H(\tilde{\mathbf{p}}_\mathbf{j}, \hat{\mathbf{p}}_\mathbf{j})$, therefore the loss function becomes:[1]

$$L_{XR}(\theta) = -\sum_{j=1}^{m} \tilde{\mathbf{p}}_\mathbf{j} \cdot \log \hat{\mathbf{p}}_\mathbf{j} \quad (3)$$

Notice that computing $\hat{\mathbf{q}}_\mathbf{j}$ requires summation over $p_\theta(x)$ for the entire set $U_j$, which can be prohibitive. We present batched approximation (Section 3.2) to overcome this.

**Temperature Parameter** Mann and McCallum (2007) find that XR might find a degenerate solution. For example, in a three class classification task, where $\tilde{p}_j = \{.5, .35, .15\}$, it might find a solution such that $\hat{p}_\theta(y) = \{.5, .35, .15\}$ for every instance, as a result, every instance will be classified the same. To avoid this, Mann and McCallum (2007) suggest to penalize flat distributions by using a temperature coefficient T likewise:

$$p_\theta(y|x) = \left( \frac{e^{\mathbf{z}W + \mathbf{b}}}{\sum_k e^{(\mathbf{z}W + \mathbf{b})_k}} \right)^{\frac{1}{T}} \quad (4)$$

Where $\mathbf{z}$ is a feature vector and W and $\mathbf{b}$ are the linear classifier parameters.

### 2.3 Aspect-based Sentiment Classification

In the aspect-based sentiment classification (ABSC) task, we are given a sentence and an aspect, and need to determine the sentiment that is expressed towards the aspect. For example the sentence *"Excellent food, although the interior could use some help."* has two aspects: *food* and *interior*, a positive sentiment is expressed about the food, but a negative sentiment is expressed about the interior. A sentence $\alpha = (w_1, w_2, \ldots, w_n)$, may contain 0 or more aspects $a_i$, where each aspect corresponds to a sub-sequence of the original sentence, and has an associated sentiment label (NEG, POS, or NEU). Concretely, we follow the task definition in the SemEval-2015 and SemEval-2016 shared tasks (Pontiki et al., 2015, 2016), in which the relevant aspects are given and the task focuses on finding the sentiment label of the aspects.

While sentence-level sentiment labels are relatively easy to obtain, aspect-level annotation are much more scarce, as demonstrated in the small datasets of the SemEval shared tasks.

[1] Note also that $\forall_j |U_j| = 1 \iff L_{XR}(\theta) = L_{cross}(\theta)$

**Algorithm 1** Stochastic Batched XR

**Inputs**: A dataset $(U_1, ..., U_m, \tilde{\mathbf{p}}_\mathbf{1}, ..., \tilde{\mathbf{p}}_\mathbf{m})$, batch size $k$, differentiable classifier $p_\theta(y|x)$

**while** not converged **do**
  $j \leftarrow$ random$(1, ..., m)$
  $U' \leftarrow$ random-choice$(U_j, k)$
  $\hat{\mathbf{q}}'_\mathbf{u} \leftarrow \sum_{x \in U'} p_\theta(x)$
  $\hat{\mathbf{p}}'_\mathbf{u} \leftarrow$ normalize$(\hat{\mathbf{q}}'_\mathbf{u})$
  $\ell \leftarrow -\tilde{p}_j \log \hat{p}_u$ ▷ Compute loss $\ell$ (eq (4))
  Compute gradients and update $\theta$
**end while**
**return** $\theta$

## 3 Technical Contributions

### 3.1 Transfer-training between related tasks with XR

Consider two classification tasks over a shared input space, a source task $s$ from $\mathcal{X}$ to $\mathcal{Y}^s$ and a target task $t$ from $\mathcal{X}$ to $\mathcal{Y}^t$, which are related through a conditional distribution $P(y^t = i | y^s = j)$. In other words, a labeling decision for task $s$ induces an expected label distribution over the task $t$. For a set of datapoints $x_1, ..., x_n$ that share a source label $y^s$, we expect to see a target label distribution of $P(y^t|y^s) = \tilde{\mathbf{p}}_{\mathbf{y^s}}$.

Given a large unlabeled dataset $D^u = (x^u_1, ..., x^u_{|D^u|})$, a small labeled dataset for the target task $D^t = ((x^t_1, y^t_1), ..., (x^t_{|D^t|}, y^t_{|D^t|}))$, classifier $C^s : \mathcal{X} \mapsto \mathcal{Y}^s$ (or sufficient training data to train one) for the source task,[2] we wish to use $C^s$

[2] Note that the classifier does not need to be trainable or differentiable. It can be a human, a rule based system, a non-parametric model, a probabilistic model, a deep learning net-

and $D^u$ to train a good classifier $C^t : \mathcal{X} \mapsto \mathcal{Y}^t$ for the target task. This can be achieved using the following procedure.

- Apply $C^s$ to $D^t$, resulting in a noisy source-side labels $\tilde{y}^s_i = C^s(x^t_i)$ for the target task.
- Estimate the conditional probability $P(y^t|\tilde{y}^s)$ table using MLE estimates over $D^t$

$$\tilde{p}_j(y^t = i|\tilde{y}^s = j) = \frac{\#(y^t = i, \tilde{y}^s = j)}{\#(\tilde{y}^s = j)}$$

  where $\#$ is a counting function over $D^t$.[3]
- Apply $C^s$ to the unlabeled data $D^u$ resulting in labels $C^s(x^u_i)$. Split $D^u$ into $|\mathcal{Y}^s|$ sets $U_j$ according to the labeling induced by $C^s$:

$$U_j = \{x^u_i \mid x^u_i \in D^u \wedge C^s(x^u_i) = j\}$$

- Use Algorithm 1 to train a classifier for the target task using input pairs $(U_j, \tilde{\mathbf{p}}_\mathbf{j})$ and the XR loss.

In words, by using XR training, we use the expected label proportions over the target task given predicted labels of the source task, to train a target-class classifier.

### 3.2 Stochastic Batched Training for Deep XR

Mann and McCallum (2007) and following work take the base classifier $p_\theta(y|x)$ to be a logistic regression classifier, for which they manually derive gradients for the XR loss and train with LBFGs (Byrd et al., 1995). However, nothing precludes us from using an arbitrary neural network instead, as long as it culminates in a softmax layer.

One complicating factor is that the computation of $\hat{\mathbf{q}}_\mathbf{j}$ in equation (1) requires a summation over $p_\theta(x)$ for the entire set $U_j$, which in our setup may contain hundreds of thousands of examples, making gradient computation and optimization impractical. We instead proposed a *stochastic batched approximation* in which, instead of requiring that the full constraint set $U_j$ will match the expected label posterior distribution, we require that sufficiently large random subsets of it will match the distribution. At each training step we compute the loss and update the gradient with respect to a different random subset. Specifically, in each training step we sample a random pair $(U_j, \tilde{\mathbf{p}}_\mathbf{j})$, sample a random subset $U'$ of $U_j$ of size $k$, and compute the local XR loss of set $U'$:

$$L_{XR}(\theta; j, U') = -\tilde{\mathbf{p}}_\mathbf{j} \cdot \log \hat{\mathbf{p}}_{\mathbf{u}'} \quad (5)$$

where $\hat{\mathbf{p}}_{\mathbf{u}'}$ is computed by summing over the elements of $U'$ rather than of $U_j$ in equations (1–2). The stochastic batched XR training algorithm is given in Algorithm 1. For large enough $k$, the expected label distribution of the subset is the same as that of the complete set.

## 4 Application to Aspect-based Sentiment

We demonstrate the procedure given above by training Aspect-based Sentiment Classifier (ABSC) using sentence-level[4] sentiment signals.

### 4.1 Relating the classification tasks

We observe that while the sentence-level sentiment does not *determine* the sentiment of individual aspects (a positive sentence may contain negative remarks about some aspects), it is very predictive of the *proportion* of sentiment labels of the fragments within a sentence. Positively labeled sentences are likely to have more positive aspects and fewer negative ones, and vice-versa for negatively-labeled sentences. While these proportions may vary on the individual sentence level, we expect them to be stable when aggregating fragments from several sentences: when considering a large enough sample of fragments that all come from positively labeled sentences, we expect the different samples to have roughly similar label proportions to each other. This situation is idealy suited for performing XR training, as described in section 3.1.

The application to ABSC is almost straightforward, but is complicated a bit by the decomposition of sentences into fragments: each sentence level decision now corresponds to multiple fragment-level decisions. Thus, we apply the sentence-level (task A) classifier $C^s$ on the aspect-level corpus $D^t$ by applying it on the sentence level and then associating the predicted sentence labels with each of the fragments, resulting in

work, etc. In this work, we use a neural classification model.

[3] In theory, we could estimate—or even "guess"—these $|\mathcal{Y}^s| \times |\mathcal{Y}^t|$ values without using $D^t$ at all. In practice, and in particular because we care about the target label proportions given *noisy* source labels $\tilde{y}^s$ assigned by $C^s$, we use MLE estimates over the tagged $D^t$.

[4] In practice, our "sentences" are in fact short documents, some of which are composed of two or more sentences.

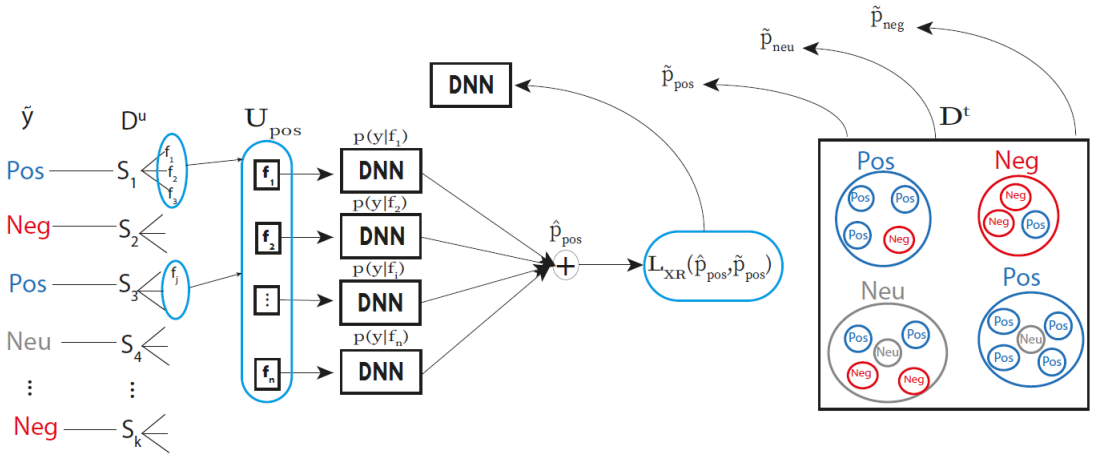


Figure 1: Illustration of the algorithm. $C^s$ is applied to $D^u$ resulting in $\tilde{y}$ for each sentence, $U_j$ is built according with the fragments of the same labelled sentences, the probabilities for each fragment in $U_j$ are summed and normalized, the XR loss in equation (4) is calculated and the network is updated.

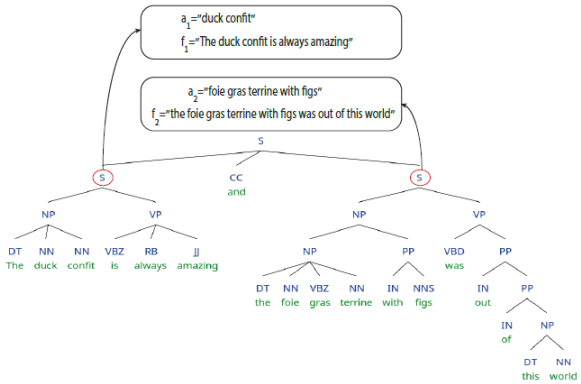


Figure 2: Illustration of the decomposition procedure, when given $a_1$="duck confit" and $a_2$= "foie gras terrine with figs" as the pivot phrases.

fragment-level labeling. Similarly, when we apply $C^s$ to the unlabeled data $D^u$ we again do it at the sentence level, but the sets $U_j$ are composed of fragments, not sentences:

$$U_j = \{f_i^\alpha \mid \alpha \in D^u \wedge f_i^\alpha \in \text{frags}(\alpha) \wedge C^s(\alpha) = j\}$$

We then apply algorithm 1 as is: at each step of training we sample a source label $j \in \{\text{POS},\text{NEG},\text{NEU}\}$, sample $k$ fragments from $U_j$, and use the XR loss to fit the expected fragment-label proportions over these $k$ fragments to $\tilde{\mathbf{p}}_\mathbf{j}$. Figure 1 illustrates the procedure.

### 4.2 Classification Architecture

We model the ABSC problem by associating each (sentence,aspect) pair with a *sentence-fragment*, and constructing a neural classifier from fragments to sentiment labels. We heuristically decompose a sentence into fragments. We use the same BiLSTM based neural architecture for both sentence classification and fragment classification.

**Fragment-decomposition** We now describe the procedure we use to associate a sentence fragment with each (sentence,aspect) pairs. The shared tasks data associates each aspect with a pivot-phrase $a$, where pivot phrase $(w_1, w_2, ...w_l)$ is defined as a pre-determined sequence of words that is contained within the sentence. For a sentence $\alpha$, a set of pivot phrases $A = (a_1, ..., a_m)$ and a specific pivot phrase $a_i$, we consult the constituency parse tree of $\alpha$ and look for tree nodes that satisfy the following conditions:[5]

1. The node governs the desired pivot phrase $a_i$.
2. The node governs either a verb (VB, VBD, VBN, VBG, VBP, VBZ) or an adjective (JJ, JJR, JJS), which is different than any $a_j \in A$.
3. The node governs a minimal number of pivot phrases from $(a_1, ..., a_m)$, ideally only $a_i$.

We then select the highest node in the tree that satisfies all conditions. The span governed by this node is taken as the fragment associated with as-

[5]Condition (2) coupled with selecting the highest node pushes towards complete phrases that contain opinions (which are usually expressed with adjectives or verbs), while the other conditions focus the attention on the desired pivot phrase.

pect $a_i$.[6] The decomposition procedure is demonstrated in Figure 2.

When aspect-level information is given, we take the pivot-phrases to be the requested aspects. When aspect-level information is *not available*, we take each noun in the sentence to be a pivot-phrase.

**Neural Classifier** Our classification model is a simple 1-layer BiLSTM encoder (a concatenation of the last states of a forward and a backward running LSTMs) followed by a linear-predictor. The encoder is fed either a complete sentence or a sentence fragment.

## 5 Experiments

**Data** Our *target* task is aspect-based fragment-classification, with small labeled datasets from the SemEval 2015 and 2016 shared tasks, each dataset containing aspect-level predictions for about 2000 sentences in the restaurants reviews domain. Our *source* classifier is based on training on up to 10,000 sentences from the same domain and 2000 sentences for validation, labeled for only for sentence-level sentiment. We additionally have an unlabeled dataset of up to 670,000 sentences from the same domain[7]. We tokenized all datasets using the Tweet Tokenizer from NLTK package[8] and parsed the tokenized sentences with AllenNLP parser.[9]

**Training Details** Both the sentence level classification models and the models trained with XR have a hidden state vector dimension of size 300, they use dropout (Hinton et al., 2012) on the sentence representation or fragment representation vector (rate=0.5) and optimized using Adam (Kingma and Ba, 2014). The sentence classification is trained with a batch size of 30 and XR models are trained with batch sizes $k$ that each contain 450 fragments[10]. We used a temperature parameter of 1[11]. We use pre-trained 300-dimensional GloVe embeddings[12] (Pennington et al., 2014), and fine-tune them during training. The XR training was validated with a validation set of 20% of SemEval-2015 training set, the sentence level BiLSTM classifiers were validated with a validation of 2000 sentences.[13] When fine-tuning to the aspect based task we used 20% of train in each dataset as validation and evaluated on this set. On each training method the models were evaluated on the validation set, after each epoch and the best model was chosen. The data is highly imbalanced, with only very few sentences receiving a NEU label. We do not deal with this imbalance directly and train both the sentence level and the XR aspect-based training on the imbalanced data. However, when training $C^s$, we trained five models and chose the best model that predicts correctly at least 20% of the neutral sentences. The models are implemented using DyNet[14] (Neubig et al., 2017).

**Baseline models** In recent years many neural network architectures with increasing sophistication were applied to the ABSC task (Nguyen and Shirai, 2015; Vo and Zhang, 2015; Tang et al., 2016a,b; Wang et al., 2016; Zhang et al., 2016; Ruder et al., 2016; Ma et al., 2017; Liu and Zhang, 2017; Chen et al., 2017; Liu et al., 2018; Yang et al., 2018; Wang et al., 2018b,a; Fan et al., 2018a,b; Li et al., 2018; Ouyang and Su, 2018). We compare to a series of state-of-the-art ABSC neural classifiers that participated in the shared tasks. TDLSTM-ATT (Tang et al., 2016a) encodes the information around an aspect using forward and backward LSTMs, followed by an attention mechanism. ATAE-LSTM (Wang et al., 2016) is an attention based LSTM variant. MM (Tang et al., 2016b) is a deep memory network with multiple-hops of attention layers. RAM (Chen et al., 2017) uses multiple attention mechanisms combined with a recurrent neural networks and a weighted memory mechanism. LSTM+SynATT+TarRep (He et al., 2018a) is an attention based LSTM which incorporates syn-

[6] On the rare occasions where we cannot find such a node, we take the root node of the tree (the entire sentence) as the fragment for the given aspect.

[7] All of the sentence-level sentiment data is obtained from the Yelp dataset challenge: https://www.yelp.com/dataset/challenge

[8] https://www.nltk.org/

[9] https://allennlp.org/

[10] We also increased the batch sizes of the baselines to match those of the XR setups. This decreased the performance of the baselines, which is consistent with the folk knowledge in the community according to which smaller batch sizes are more effective overall.

[11] Despite (Mann and McCallum, 2007) claim regarding the temperature parameter, we observed lower performance when using it in our setup. However, in other setups this parameter might be found to be beneficial.

[12] https://nlp.stanford.edu/projects/glove/

[13] We also tested the sentence BiLSTM baselines with a SemEval validation set, and received slightly lower results without a significant statistical difference.

[14] https://github.com/clab/dynet

| Data | Method | SemEval-15 | | SemEval-16 | |
|---|---|---|---|---|---|
| | | Acc. | Macro-F1 | Acc. | Macro-F1 |
| A | TDLSTM+ATT (Tang et al., 2016a) | 77.10 | 59.46 | 83.11 | 57.53 |
| A | ATAE-LSTM (Wang et al., 2016) | 78.48 | 62.84 | 83.77 | 61.71 |
| A | MM (Tang et al., 2016b) | 77.89 | 59.52 | 83.04 | 57.91 |
| A | RAM (Chen et al., 2017) | 79.98 | 60.57 | 83.88 | 62.14 |
| A | LSTM+SynATT+TarRep (He et al., 2018a) | 81.67 | 66.05 | 84.61 | 67.45 |
| S+A | Semisupervised (He et al., 2018b) | 81.30 | **68.74** | 85.58 | 69.76 |
| S | BiLSTM-$10^4$ Sentence Training | 80.24 $\pm$1.64 | 61.89 $\pm$0.94 | 80.89 $\pm$2.79 | 61.40 $\pm$2.49 |
| S+A | BiLSTM-$10^4$ Sentence Training $\rightarrow$Aspect Based Finetuning | 77.75 $\pm$2.09 | 60.83 $\pm$4.53 | 84.87$\pm$0.31 | 61.87 $\pm$5.44 |
| N | BiLSTM-XR-Dev Estimation | **83.31**$^*$ $\pm$ **0.62** | 62.24 $\pm$0.66 | **87.68**$^*$ $\pm$ **0.47** | 63.23 $\pm$1.81 |
| N | BiLSTM-XR | **83.31**$^*$ $\pm$ **0.77** | 64.42 $\pm$ 2.78 | **88.12**$^*$ $\pm$ **0.24** | 68.60 $\pm$1.79 |
| N+A | BiLSTM-XR $\rightarrow$Aspect Based Finetuning | **83.44**$^*$ $\pm$ **0.74** | **67.23** $\pm$ **1.42** | **87.66**$^*$ $\pm$ **0.28** | **71.19**†$\pm$ **1.40** |

Table 1: Average accuracies and Macro-F1 scores over five runs with random initialization along with their standard deviations. Bold: best results or within std of them. $^*$ indicates that the method's result is significantly better than all baseline methods, † indicates that the method's result is significantly better than all baselines methods that use the aspect-based data only, with $p < 0.05$ according to a one-tailed unpaired t-test. The data annotations **S**, **N** and **A** indicate training with Sentence-level, Noisy sentence-level and Aspect-level data respectively. Numbers for TDLSTM+Att,ATAE-LSTM,MM,RAM and LSTM+SynATT+TarRep are from (He et al., 2018a). Numbers for Semisupervised are from (He et al., 2018b).

tactic information into the attention mechanism and uses an auto-encoder structure to produce an aspect representations. All of these models are trained only on the small, fully-supervised ABSC datasets.

"Semisupervised" is the semi-supervised setup of (He et al., 2018b), it train an attention-based LSTM model on 30,000 documents additional to an aspect-based train set, 10,000 documents to each class. We consider additional two simple but strong semi-supervised baselines. Sentence-BiLSTM is our BiLSTM model trained on the $10^4$ sentence-level annotations, and applied as-is to the individual fragments. Sentence-BiLSTM+Finetuning is the same model, but finetuned on the aspect-based data as explained above. Finetuning is performed using our own implementation of the attention-based model of He et al. (2018b).[15] Both these models are on par with the fully-supervised ABSC models.

**Empirical Proportions** The proportion constraint sets $\tilde{\mathbf{p}}_\mathbf{j}$ based on the SemEval-2015 aspect-based train data are:
$\tilde{\mathbf{p}}_{\text{POS}} = \{\text{POS} : 0.93, \text{NEG} : 0.06, \text{NEU} : 0.01\}$
$\tilde{\mathbf{p}}_{\text{NEG}} = \{\text{POS} : 0.27, \text{NEG} : 0.7, \text{NEU} : 0.03\}$
$\tilde{\mathbf{p}}_{\text{NEU}} = \{\text{POS} : 0.45, \text{NEG} : 0.41, \text{NEU} : 0.14\}$

### 5.1 Main Results

Table 1 compares these baselines to three XR conditions.[16]

The first condition, BiLSTM-XR-Dev, performs XR training on the automatically-labeled sentence-level dataset. The only access it has to aspect-level annotation is for estimating the proportions of labels for each sentence-level label, which is done based on the validation set of SemEval-2015 (i.e., 20% of the train set). The XR setting is very effective: without using any in-task data, this model already surpasses all other models, both supervised and semi-supervised, except for the (He et al., 2018b,a) models which achieve higher F1 scores. We note that in contrast to XR, the competing models have complete access to the supervised aspect-based labels. The second condition, BiLSTM-XR, is similar but now the model is allowed to estimate the conditional label proportions based on the entire aspect-based training set (the classifier still does not have direct access to the labels beyond the aggregate proportion information). This improves results further, showing the importance of accurately estimating the proportions. Finally, in BiLSTM-XR+Finetuning, we follow the XR training with fully supervised fine-tuning on the small labeled dataset, using the attention-based model of He et al. (2018b). This achieves the best results, and surpasses also the semi-supervised He et al. (2018b) baseline on accuracy, and matching it on F1.[17]

We report significance tests for the robustness

[15] We changed the LSTM component to a BiLSTM.

[16] To be consistent with existing research (He et al., 2018b), aspects with conflicted polarity are removed.

[17] We note that their setup uses clean and more balanced annotations, i.e. they use 10,000 samples for each label, which helps predicting the infrequent neutral sentiment. We however, use noisy sentence sentiment labels which are automatically obtained from a trained classifier, which trains on 10,000 samples in their natural imbalanced distribution.

of the method under random parameter initialization. Our reported numbers are averaged over five random initialization. Since the datasets are unbalanced w.r.t the label distribution, we report both accuracy and macro-F1.

The XR training is also more stable than the other semi-supervised baselines, achieving substantially lower standard deviations across different runs.

### 5.2 Further experiments

In each experiment in this section we estimate the proportions using the SemEval-2015 train set.

**Effect of unlabeled data size** How does the XR training scale with the amount of unlabeled data? Figure 3a shows the macro-F1 scores on the entire SemEval-2016 dataset, with different unlabeled corpus sizes (measured in number of sentences). An unannotated corpus of $5 \times 10^4$ sentences is sufficient to surpass the results of the $10^4$ sentence-level trained classifier, and more unannotated data further improves the results.

**Effect of Base-classifier Quality** Our method requires a sentence level classifier $C^s$ to label both the target-task corpus and the unlabeled corpus. How does the quality of this classifier affect the overall XR training? We vary the amount of supervision used to train $C^s$ from 0 sentences (assigning the same label to all sentences), to 100, 1000, 5000 and 10000 sentences. We again measure macro-F1 on the entire SemEval 2016 corpus. The results in Figure 3b show that when using the prior distributions of aspects (0), the model struggles to learn from this signal, it learns mostly to predict the majority class, and hence reaches very low F1 scores of 35.28. The more data given to the sentence level classifier, the better the potential results will be when training with our method using the classifier labels, with a classifiers trained on 100,1000,5000 and 10000 labeled sentences, we get a F1 scores of 53.81, 58.84, 61.81, 65.58 respectively. Improvements in the source task classifier's quality clearly contribute to the target task accuracy.

**Effect of $k$** The Stochastic Batched XR algorithm (Algorithm 1) samples a batch of $k$ examples at each step to estimate the posterior label distribution used in the loss computation. How does the size of $k$ affect the results? We use $k = 450$ fragments in our main experiments, but smaller values of $k$ reduce GPU memory load and may train better in practice. We tested our method with varying values of $k$ on a sample of $5 \times 10^4$, using batches that are composed of fragments of 5, 25, 100, 450, 1000 and 4500 sentences. The results are shown in Figure 3c. Setting $k = 5$ result in low scores. Setting $k = 25$ yields better F1 score but with high variance across runs. For $k = 100$ fragments the results begin to stabilize, we also see a slight decrease in F1-scores with larger batch sizes. We attribute this drop despite having better estimation of the gradients to the general trend of larger batch sizes being harder to train with stochastic gradient methods.

### 5.3 Pre-training, BERT

The XR training can be performed also over pre-trained representations. We experiment with two pre-training methods: (1) pre-training by training the BiLSTM model to predict the noisy sentence-level predictions. (2) Using the pre-trained BERT representation (Devlin et al., 2018). For (1), we compare the effect of pre-train on unlabeled corpora of sizes of $5 \times 10^4$, $10^5$ and $6.7 \times 10^5$ sentences. Results in Figure 3d show that this form of pre-training is effective for smaller unlabeled corpora but evens out for larger ones.

**BERT** For the BERT experiments, we experiment with the BERT-base model[18] with $k = 450$ sets, 30 epochs for XR training or sentence level fine-tuning[19] and 15 epochs for aspect based fine-tuning, on each training method we evaluated the model on the dev set after each epoch and the best model was chosen[20]. We compare the following setups:

-BERT→Aspect Based Finetuning: pretrained BERT model finetuned to the aspect based task.

-BERT→ $10^4$: A pretrained BERT model finetuned to the sentence level task on the $10^4$ sentences, and tested by predicting fragment-level sentiment.

-BERT→$10^4$→Aspect Based Finetuning: pretrained BERT model finetuned to the sentence level task, and finetuned again to the aspect based one.

-BERT→XR: pretrained BERT model followed by

[18] We could not fit $k = 450$ sets of BERT-large on our GPU.

[19] When fine-tuning to the sentence level task, we provide the sentence as input. When fine-tuning to the aspect-level task, we provide the sentence, a seperator and then the aspect.

[20] The other configuration parameters were the default ones in https://github.com/huggingface/pytorch-pretrained-BERT

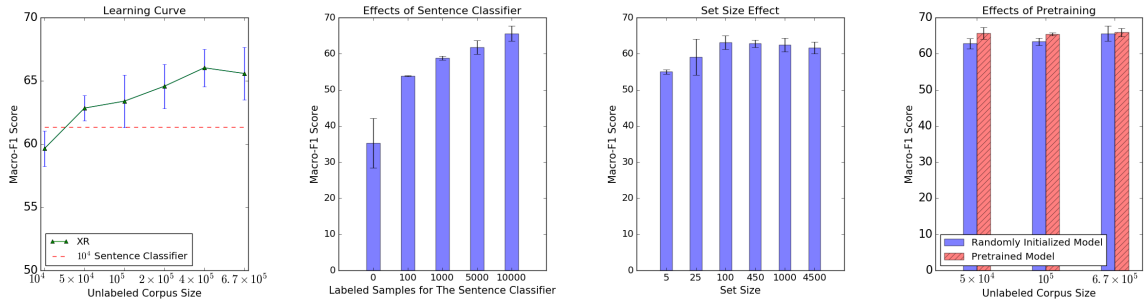


(a) (b) (c) (d)

Figure 3: Macro-F1 scores for the entire SemEval-2016 dataset of the different analyses. (a) the contribution of unlabeled data. (b) the effect of sentence classifier quality. (c) the effect of k. (d) the effect of sentence-level pretraining vs. corpus size.

| Data | Training | SemEval-15 | | SemEval-16 | |
|---|---|---|---|---|---|
| | | Acc. | Macro-F1 | Acc. | Macro-F1 |
| N | BiLSTM-XR | 83.31 ± 0.77 | 64.42 ± 2.78 | 88.12 ± 0.24 | 68.60 ±1.79 |
| N+A | BiLSTM-XR →Aspect Based Finetuning | 83.44 ± 0.74 | 67.23 ± 1.42 | 87.66 ± 0.28 | 71.19 ± 1.40 |
| A | BERT→Aspect Based Finetuning | 81.87 ±1.12 | 59.24 ±4.94 | 85.81 ±1.07 | 62.46 ±6.76 |
| S | BERT→$10^4$ Sent Finetuning | 83.29 ±0.77 | 66.79 ±1.99 | 84.53 ±1.66 | 65.53 ±3.03 |
| S+A | BERT→$10^4$ Sent Finetuning →Aspect Based Finetuning | 82.54 ±1.21 | 64.13 ±5.05 | 85.67 ±1.14 | 64.13 ±7.07 |
| N | BERT→XR | 85.46* ±0.59 | 66.86 ±2.8 | 89.5* ±0.55 | 70.86†±2.96 |
| N+A | BERT→XR →Aspect Based Finetuning | **85.78*** ± **0.65** | **68.74** ± **1.36** | **89.57*** ± **1.4** | **73.89*** ± **2.05** |

Table 2: BERT pre-training: average accuracies and Macro-F1 scores from five runs and their stdev. * indicates that the method's result is significantly better than all baseline methods, † indicates that the method's result is significantly better than all non XR baseline methods, with $p < 0.05$ according to a one-tailed unpaired t-test. The data annotations **S**, **N** and **A** indicate training with Sentence-level, Noisy sentence-level and Aspect-level data respectively.

XR training using our method.

-BERT→ XR → Aspect Based Finetuning: pre-trained BERT followed by XR training and then fine-tuned to the aspect level task.

The results are presented in Table 2. As before, aspect-based fine-tuning is beneficial for both SemEval-16 and SemEval-15. Training a BiLSTM with XR surpasses pre-trained BERT models and using XR training on top of the pre-trained BERT models substantially increases the results even further.

## 6 Discussion

We presented a transfer learning method based on expectation regularization (XR), and demonstrated its effectiveness for training aspect-based sentiment classifiers using sentence-level supervision. The method achieves state-of-the-art results for the task, and is also effective for improving on top of a strong pre-trained BERT model. The proposed method provides an additional data-efficient tool in the modeling arsenal, which can be applied on its own or together with another training method, in situations where there is a conditional relations between the labels of a source task for which we have supervision, and a target task for which we don't.

While we demonstrated the approach on the sentiment domain, the required conditional dependence between task labels is present in many situations. Other possible application of the method includes training language identification of tweets given geo-location supervision (knowing the geographical region gives a prior on languages spoken), training predictors for renal failure from textual medical records given classifier for diabetes (there is a strong correlation between the two conditions), training a political affiliation classifier from social media tweets based on age-group classifiers, zip-code information, or social-status classifiers (there are known correlations between all of these to political affiliation), training hate-speech detection based on emotion detection, and so on.

## Acknowledgements

The work was supported in part by The Israeli Science Foundation (grant number 1555/15).

## References


Richard H. Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. 1995. A limited memory algorithm for bound constrained optimization. *SIAM J. Scientific Computing*, 16(5):1190–1208.

Ming-Wei Chang, Lev Ratinov, and Dan Roth. 2007. Guiding semi-supervision with constraint-driven learning. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 280–287. Association for Computational Linguistics.

Ming-Wei Chang, Lev-Arie Ratinov, and Dan Roth. 2012. Structured learning with constrained conditional models. *Machine Learning*, 88(3):399–431.

Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. 2017. Recurrent attention network on memory for aspect sentiment analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 452–461. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Gregory Druck, Gideon S. Mann, and Andrew McCallum. 2008. Learning from labeled features using generalized expectation criteria. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2008, Singapore, July 20-24, 2008*, pages 595–602.

Chuang Fan, Qinghong Gao, Jiachen Du, Lin Gui, Ruifeng Xu, and Kam-Fai Wong. 2018a. Convolution-based memory network for aspect-based sentiment analysis. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, pages 1161–1164.

Feifan Fan, Yansong Feng, and Dongyan Zhao. 2018b. Multi-grained attention network for aspect-level sentiment classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3433–3442. Association for Computational Linguistics.

Kuzman Ganchev and Dipanjan Das. 2013. Cross-lingual discriminative learning of sequence models with posterior regularization. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1996–2006. Association for Computational Linguistics.

Kuzman Ganchev, João Graça, Jennifer Gillenwater, and Ben Taskar. 2010. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11:2001–2049.

João Graça, Kuzman Ganchev, and Ben Taskar. 2007. Expectation maximization and posterior constraints. In *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, pages 569–576.

Aria Haghighi and Dan Klein. 2006. Prototype-driven learning for sequence models. In *Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 4-9, 2006, New York, New York, USA*.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2018a. Effective attention modeling for aspect-level sentiment classification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1121–1131. Association for Computational Linguistics.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2018b. Exploiting document knowledge for aspect-level sentiment classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 579–585. Association for Computational Linguistics.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580.

Tom Hope and Dafna Shahaf. 2016. Ballpark learning: Estimating labels from rough group comparisons. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part II*, pages 299–314.

Rong Jin and Yi Liu. 2005. A framework for incorporating class priors into discriminative classification. In *Advances in Knowledge Discovery and Data Mining, 9th Pacific-Asia Conference, PAKDD 2005, Hanoi, Vietnam, May 18-20, 2005, Proceedings*, pages 568–577.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Lishuang Li, Yang Liu, and AnQiao Zhou. 2018. Hierarchical attention based position-aware network for aspect-level sentiment analysis. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 181–189. Association for Computational Linguistics.

Percy Liang, Michael I. Jordan, and Dan Klein. 2009. Learning from measurements in exponential families. In *Proceedings of the 26th Annual International Conference on Machine Learning, ICML*

*2009, Montreal, Quebec, Canada, June 14-18, 2009*, pages 641–648.

Fei Liu, Trevor Cohn, and Timothy Baldwin. 2018. Recurrent entity networks with delayed memory update for targeted aspect-based sentiment analysis. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 278–283. Association for Computational Linguistics.

Jiangming Liu and Yue Zhang. 2017. Attention modeling for targeted sentiment. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 572–577. Association for Computational Linguistics.

Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2017. Interactive attention networks for aspect-level sentiment classification. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 4068–4074.

Gideon S. Mann and Andrew McCallum. 2007. Simple, robust, scalable semi-supervised learning via expectation regularization. In *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*, pages 593–600.

Gideon S. Mann and Andrew McCallum. 2010a. Generalized expectation criteria for semi-supervised learning with weakly labeled data. *Journal of Machine Learning Research*, 11:955–984.

Gideon S. Mann and Andrew McCallum. 2010b. Generalized expectation criteria for semi-supervised learning with weakly labeled data. *Journal of Machine Learning Research*, 11:955–984.

Prem Melville, Wojciech Gryc, and Richard D. Lawrence. 2009. Sentiment analysis of blogs by combining lexical knowledge with text classification. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009*, pages 1275–1284.

Ardehaly Ehsan Mohammady and Aron Culotta. 2015. Inferring latent attributes of twitter users with label regularization. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 185–195. Association for Computational Linguistics.

David R. Musicant, Janara M. Christensen, and Jamie F. Olson. 2007. Supervised learning by training on aggregate outputs. In *Proceedings of the 7th IEEE International Conference on Data Mining (ICDM 2007), October 28-31, 2007, Omaha, Nebraska, USA*, pages 252–261.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. Dynet: The dynamic neural network toolkit. *CoRR*, abs/1701.03980.

Thien Hai Nguyen and Kiyoaki Shirai. 2015. Phrasernn: Phrase recursive neural network for aspect-based sentiment analysis. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2509–2514. Association for Computational Linguistics.

Zhifan Ouyang and Jindian Su. 2018. Dependency parsing and attention network for aspect-level sentiment classification. In *Natural Language Processing and Chinese Computing - 7th CCF International Conference, NLPCC 2018, Hohhot, China, August 26-30, 2018, Proceedings, Part I*, pages 391–403.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, Mohammad AL-Smadi, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphee De Clercq, Veronique Hoste, Marianna Apidianaki, Xavier Tannier, Natalia Loukachevitch, Evgeniy Kotelnikov, Núria Bel, Salud María Jiménez-Zafra, and Gülşen Eryiğit. 2016. Semeval-2016 task 5: Aspect based sentiment analysis. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 19–30. Association for Computational Linguistics.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Suresh Manandhar, and Ion Androutsopoulos. 2015. Semeval-2015 task 12: Aspect based sentiment analysis. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 486–495. Association for Computational Linguistics.

Novi Quadrianto, Alexander J. Smola, Tibério S. Caetano, and Quoc V. Le. 2009a. Estimating labels from label proportions. *Journal of Machine Learning Research*, 10:2349–2374.

Novi Quadrianto, Alexander J. Smola, Tibério S. Caetano, and Quoc V. Le. 2009b. Estimating labels from label proportions. *Journal of Machine Learning Research*, 10:2349–2374.

Sebastian Ruder, Parsa Ghaffari, and John G. Breslin. 2016. A hierarchical model of reviews for aspect-based sentiment analysis. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 999–1005. Association for Computational Linguistics.

Robert E. Schapire, Marie Rochery, Mazin G. Rahim, and Narendra K. Gupta. 2002. Incorporating prior knowledge into boosting. In *Machine Learning, Proceedings of the Nineteenth International Conference (ICML 2002), University of New South Wales, Sydney, Australia, July 8-12, 2002*, pages 538–545.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2016a. Effective lstms for target-dependent sentiment classification. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3298–3307. The COLING 2016 Organizing Committee.

Duyu Tang, Bing Qin, and Ting Liu. 2016b. Aspect level sentiment classification with deep memory network. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 214–224. Association for Computational Linguistics.

Duy-Tin Vo and Yue Zhang. 2015. Target-dependent twitter sentiment classification with rich automatic features. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1347–1353.

Jingjing Wang, Jie Li, Shoushan Li, Yangyang Kang, Min Zhang, Luo Si, and Guodong Zhou. 2018a. Aspect sentiment classification with both word-level and clause-level attention networks. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden.*, pages 4439–4445.

Mengqiu Wang and Christopher D. Manning. 2014. Cross-lingual projected expectation regularization for weakly supervised learning. *TACL*, 2:55–66.

Shuai Wang, Sahisnu Mazumder, Bing Liu, Mianwei Zhou, and Yi Chang. 2018b. Target-sensitive memory networks for aspect sentiment classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 957–967. Association for Computational Linguistics.

Yequan Wang, Minlie Huang, xiaoyan zhu, and Li Zhao. 2016. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 606–615. Association for Computational Linguistics.

Zuoguan Wang, Siwei Lyu, Gerwin Schalk, and Qiang Ji. 2012. Learning with target prior. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, pages 2240–2248.

Min Yang, Qiang Qu, Xiaojun Chen, Chaoxue Guo, Ying Shen, and Kai Lei. 2018. Feature-enhanced attention network for target-dependent sentiment classification. *Neurocomputing*, 307:91–97.

Meishan Zhang, Yue Zhang, and Duy-Tin Vo. 2016. Gated neural networks for targeted sentiment analysis. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 3087–3093.

Jun Zhu, Ning Chen, and Eric P. Xing. 2014. Bayesian inference with posterior regularization and applications to infinite latent svms. *Journal of Machine Learning Research*, 15(1):1799–1847.

# Knowledge Enhanced Contextual Word Representations

**Matthew E. Peters**[1], **Mark Neumann**[1], **Robert L. Logan IV**[2], **Roy Schwartz**[1,3], **Vidur Joshi**[1], **Sameer Singh**[2], **and Noah A. Smith**[1,3]

[1]Allen Institute for Artificial Intelligence, Seattle, WA, USA
[2]University of California, Irvine, CA, USA
[3]Paul G. Allen School of Computer Science & Engineering, University of Washington
{matthewp,markn,roys,noah}@allenai.org
{rlogan,sameer}@uci.edu

## Abstract

Contextual word representations, typically trained on unstructured, unlabeled text, do not contain any explicit grounding to real world entities and are often unable to remember facts about those entities. We propose a general method to embed multiple knowledge bases (KBs) into large scale models, and thereby enhance their representations with structured, human-curated knowledge. For each KB, we first use an integrated entity linker to retrieve relevant entity embeddings, then update contextual word representations via a form of word-to-entity attention. In contrast to previous approaches, the entity linkers and self-supervised language modeling objective are jointly trained end-to-end in a multitask setting that combines a small amount of entity linking supervision with a large amount of raw text. After integrating WordNet and a subset of Wikipedia into BERT, the knowledge enhanced BERT (KnowBert) demonstrates improved perplexity, ability to recall facts as measured in a probing task and downstream performance on relationship extraction, entity typing, and word sense disambiguation. KnowBert's runtime is comparable to BERT's and it scales to large KBs.

## 1 Introduction

Large pretrained models such as ELMo (Peters et al., 2018), GPT (Radford et al., 2018), and BERT (Devlin et al., 2019) have significantly improved the state of the art for a wide range of NLP tasks. These models are trained on large amounts of raw text using self-supervised objectives. However, they do not contain any explicit grounding to real world entities and as a result have difficulty recovering factual knowledge (Logan et al., 2019).

Knowledge bases (KBs) provide a rich source of high quality, human-curated knowledge that can be used to ground these models. In addition, they often include complementary information to that found in raw text, and can encode factual knowledge that is difficult to learn from selectional preferences either due to infrequent mentions of commonsense knowledge or long range dependencies.

We present a general method to insert multiple KBs into a large pretrained model with a Knowledge Attention and Recontextualization (KAR) mechanism. The key idea is to explicitly model *entity spans* in the input text and use an entity linker to retrieve relevant entity embeddings from a KB to form knowledge enhanced entity-span representations. Then, the model recontextualizes the entity-span representations with word-to-entity attention to allow long range interactions between contextual word representations and all entity spans in the context. The entire KAR is inserted between two layers in the middle of a pretrained model such as BERT.

In contrast to previous approaches that integrate external knowledge into task-specific models with task supervision (e.g., Yang and Mitchell, 2017; Chen et al., 2018), our approach learns the entity linkers with self-supervision on unlabeled data. This results in general purpose knowledge enhanced representations that can be applied to a wide range of downstream tasks.

Our approach has several other benefits. First, it leaves the top layers of the original model unchanged so we may retain the output loss layers and fine-tune on unlabeled corpora while training the KAR. This also allows us to simply swap out BERT for KnowBert in any downstream application. Second, by taking advantage of the existing high capacity layers in the original model, the KAR is lightweight, adding minimal additional parameters and runtime. Finally, it is easy to incorporate additional KBs by simply inserting them at other locations.

KnowBert is agnostic to the form of the



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 43–54,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

KB, subject to a small set of requirements (see Sec. 3.2). We experiment with integrating both WordNet (Miller, 1995) and Wikipedia, thus explicitly adding word sense knowledge and facts about named entities (including those unseen at training time). However, the method could be extended to commonsense KBs such as ConceptNet (Speer et al., 2017) or domain specific ones (e.g., UMLS; Bodenreider, 2004).

We evaluate KnowBert with a mix of intrinsic and extrinsic tasks. Despite being based on the smaller $\text{BERT}_{\text{BASE}}$ model, the experiments demonstrate improved masked language model perplexity and ability to recall facts over $\text{BERT}_{\text{LARGE}}$. The extrinsic evaluations demonstrate improvements for challenging relationship extraction, entity typing and word sense disambiguation datasets, and often outperform other contemporaneous attempts to incorporate external knowledge into BERT.

## 2 Related Work

**Pretrained word representations** Initial work learning word vectors focused on static word embeddings using multi-task learning objectives (Collobert and Weston, 2008) or corpus level co-occurence statistics (Mikolov et al., 2013a; Pennington et al., 2014). Recently the field has shifted toward learning context-sensitive embeddings (Dai and Le, 2015; Peters et al., 2018; Devlin et al., 2019). We build upon these by incorporating structured knowledge into these models.

**Entity embeddings** Entity embedding methods produce continuous vector representations from external knowledge sources. Knowledge graph-based methods optimize the score of observed triples in a knowledge graph. These methods broadly fall into two categories: translational distance models (Bordes et al., 2013; Wang et al., 2014b; Lin et al., 2015; Xiao et al., 2016) which use a distance-based scoring function, and linear models (Nickel et al., 2011; Yang et al., 2014; Trouillon et al., 2016; Dettmers et al., 2018) which use a similarity-based scoring function. We experiment with TuckER (Balazevic et al., 2019) embeddings, a recent linear model which generalizes many of the aforecited models. Other methods combine entity metadata with the graph (Xie et al., 2016), use entity contexts (Chen et al., 2014; Ganea and Hofmann, 2017), or a combination of contexts and the KB (Wang et al., 2014a; Gupta et al., 2017). Our approach is agnostic to the details of the entity embedding method and as a result is able to use any of these methods.

**Entity-aware language models** Some previous work has focused on adding KBs to generative language models (LMs) (Ahn et al., 2017; Yang et al., 2017; Logan et al., 2019) or building entity-centric LMs (Ji et al., 2017). However, these methods introduce latent variables that require full annotation for training, or marginalization. In contrast, we adopt a method that allows training with large amounts of unannotated text.

**Task-specific KB architectures** Other work has focused on integrating KBs into neural architectures for specific downstream tasks (Yang and Mitchell, 2017; Sun et al., 2018; Chen et al., 2018; Bauer et al., 2018; Mihaylov and Frank, 2018; Wang and Jiang, 2019; Yang et al., 2019). Our approach instead uses KBs to learn more generally transferable representations that can be used to improve a variety of downstream tasks.

## 3 KnowBert

KnowBert incorporates knowledge bases into BERT using the Knowledge Attention and Recontextualization component (KAR). We start by describing the BERT and KB components. We then move to introducing KAR. Finally, we describe the training procedure, including the multi-task training regime for jointly training KnowBert and an entity linker.

### 3.1 Pretrained BERT

We describe KnowBert as an extension to (and candidate replacement for) BERT, although the method is general and can be applied to any deep pretrained model including left-to-right and right-to-left LMs such as ELMo and GPT. Formally, BERT accepts as input a sequence of $N$ Word-Piece tokens (Sennrich et al., 2016; Wu et al., 2016), $(x_1, \ldots, x_N)$, and computes $L$ layers of $D$-dimensional contextual representations $\mathbf{H}_i \in \mathbb{R}^{N \times D}$ by successively applying non-linear functions $\mathbf{H}_i = \text{F}_i(\mathbf{H}_{i-1})$. The non-linear function is a multi-headed self-attention layer followed by a position-wise multilayer perceptron (MLP)

(Vaswani et al., 2017):

$$\begin{aligned}&\mathrm{F}_i(\mathbf{H}_{i-1}) = \\ &\quad \mathrm{TransformerBlock}(\mathbf{H}_{i-1}) = \\ &\quad \mathrm{MLP}(\mathrm{MultiHeadAttn}(\mathbf{H}_{i-1}, \mathbf{H}_{i-1}, \mathbf{H}_{i-1})).\end{aligned}$$

The multi-headed self-attention uses $\mathbf{H}_{i-1}$ as the query, key, and value to allow each vector to attend to every other vector.

BERT is trained to minimize an objective function that combines both next-sentence prediction (NSP) and masked LM log-likelihood (MLM):

$$\mathcal{L}_{\text{BERT}} = \mathcal{L}_{\text{NSP}} + \mathcal{L}_{\text{MLM}}.$$

Given two inputs $\mathbf{x}_A$ and $\mathbf{x}_B$, the next-sentence prediction task is binary classification to predict whether $\mathbf{x}_B$ is the next sentence following $\mathbf{x}_A$. The masked LM objective randomly replaces a percentage of input word pieces with a special `[MASK]` token and computes the negative log-likelihood of the missing token with a linear layer and softmax over all possible word pieces.

### 3.2 Knowledge Bases

The key contribution of this paper is a method to incorporate knowledge bases (KB) into a pretrained BERT model. To encompass as wide a selection of prior knowledge as possible, we adopt a broad definition for a KB in the most general sense as fixed collection of $K$ entity nodes, $e_k$, from which it is possible to compute entity embeddings, $\mathbf{e}_k \in \mathbb{R}^E$. This includes KBs with a typical `(subj, rel, obj)` graph structure, KBs that contain only entity metadata without a graph, and those that combine both a graph and entity metadata, as long as there is some method for embedding the entities in a low dimensional vector space. We also do not make any assumption that the entities are typed. As we show in Sec. 4.1 this flexibility is beneficial, where we compute entity embeddings from WordNet using both the graph and synset definitions, but link directly to Wikipedia pages without a graph by using embeddings computed from the entity description.

We also assume that the KB is accompanied by an entity candidate selector that takes as input some text and returns a list of $C$ potential entity links, each consisting of the start and end indices of the potential mention span and $M_m$ candidate entities in the KG:

$$\begin{aligned}\mathcal{C} = \{\langle(\text{start}_m, \text{end}_m), (e_{m,1}, \ldots, e_{m,M_m})\rangle \mid \\ m \in 1 \ldots C, e_k \in 1 \ldots K\}.\end{aligned}$$

In practice, these are often implemented using precomputed dictionaries (e.g., CrossWikis; Spitkovsky and Chang, 2012), KB specific rules (e.g., a WordNet lemmatizer), or other heuristics (e.g., string match; Mihaylov and Frank, 2018). Ling et al. (2015) showed that incorporating candidate priors into entity linkers can be a powerful signal, so we optionally allow for the candidate selector to return an associated prior probability for each entity candidate. In some cases, it is beneficial to over-generate potential candidates and add a special `NULL` entity to each candidate list, thereby allowing the linker to discriminate between actual links and false positive candidates. In this work, the entity candidate selectors are fixed but their output is passed to a learned context dependent entity linker to disambiguate the candidate mentions.

Finally, by restricting the number of candidate entities to a fixed small number (we use 30), KnowBert's runtime is independent of the size the KB, as it only considers a small subset of all possible entities for any given text. As the candidate selection is rule-based and fixed, it is fast and in our implementation is performed asynchronously on CPU. The only overhead for scaling up the size of the KB is the memory footprint to store the entity embeddings.

### 3.3 KAR

The Knowledge Attention and Recontextualization component (KAR) is the heart of KnowBert. The KAR accepts as input the contextual representations at a particular layer, $\mathbf{H}_i$, and computes knowledge enhanced representations $\mathbf{H}'_i = \mathrm{KAR}(\mathbf{H}_i, \mathcal{C})$. This is fed into the next pretrained layer, $\mathbf{H}_{i+1} = \mathrm{TransformerBlock}(\mathbf{H}'_i)$, and the remainder of BERT is run as usual.

In this section, we describe the KAR's key components: mention-span representations, retrieval of relevant entity embeddings using an entity linker, update of mention-span embeddings with retrieved information, and recontextualization of entity-span embeddings with word-to-entity-span attention. We describe the KAR for a single KB, but extension to multiple KBs at different layers is straightforward. See Fig. 1 for an overview.

**Mention-span representations** The KAR starts with the KB entity candidate selector that provides a list of candidate mentions which it uses to compute mention-span representations. $\mathbf{H}_i$ is first pro-

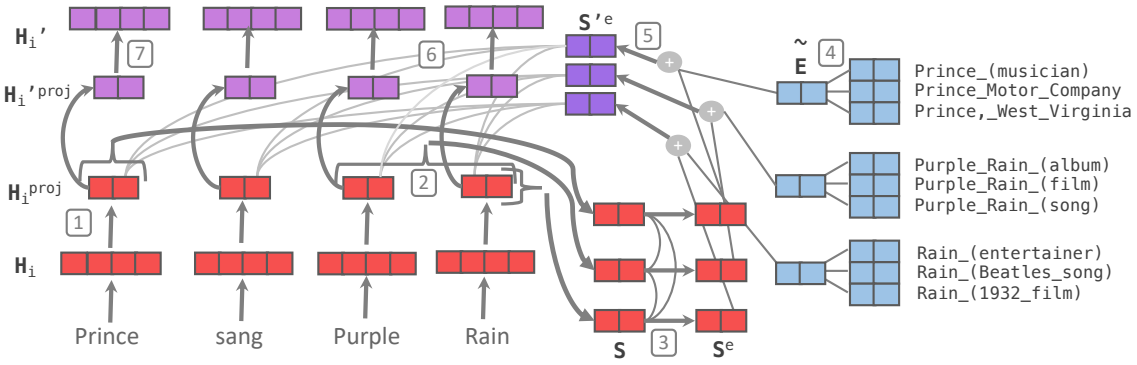


Figure 1: The Knowledge Attention and Recontextualization (KAR) component. BERT word piece representations ($\mathbf{H}_i$) are first projected to $\mathbf{H}_i^{\text{proj}}$ (1), then pooled over candidate mentions spans (2) to compute $\mathbf{S}$, and contextualized into $\mathbf{S}^e$ using mention-span self-attention (3). An integrated entity linker computes weighted average entity embeddings $\tilde{\mathbf{E}}$ (4), which are used to enhance the span representations with knowledge from the KB (5), computing $\mathbf{S}'^e$. Finally, the BERT word piece representations are recontextualized with word-to-entity-span attention (6) and projected back to the BERT dimension (7) resulting in $\mathbf{H}'_i$.

jected to the entity dimension ($E$, typically 200 or 300, see Sec. 4.1) with a linear projection,

$$\mathbf{H}_i^{\text{proj}} = \mathbf{H}_i \mathbf{W}_1^{\text{proj}} + \mathbf{b}_1^{\text{proj}}. \quad (1)$$

Then, the KAR computes $C$ mention-span representations $\mathbf{s}_m \in \mathbb{R}^E$, one for each candidate mention, by pooling over all word pieces in a mention-span using the self-attentive span pooling from Lee et al. (2017). The mention-spans are stacked into a matrix $\mathbf{S} \in \mathbb{R}^{C \times E}$.

**Entity linker** The entity linker is responsible for performing entity disambiguation for each potential mention from among the available candidates. It first runs mention-span self-attention to compute

$$\mathbf{S}^e = \text{TransformerBlock}(\mathbf{S}). \quad (2)$$

The span self-attention is identical to the typical transformer layer, exception that the self-attention is between mention-span vectors instead of word piece vectors. This allows KnowBert to incorporate global information into each linking decision so that it can take advantage of entity-entity co-occurrence and resolve which of several overlapping candidate mentions should be linked.[1]

Following Kolitsas et al. (2018), $\mathbf{S}^e$ is used to score each of the candidate entities while incorporating the candidate entity prior from the KB. Each candidate span $m$ has an associated mention-span vector $\mathbf{s}_m^e$ (computed via Eq. 2), $M_m$ candidate entities with embeddings $\mathbf{e}_{mk}$ (from the KB), and prior probabilities $p_{mk}$. We compute $M_m$ scores using the prior and dot product between the entity-span vectors and entity embeddings,

$$\psi_{mk} = \text{MLP}(p_{mk}, \mathbf{s}_m^e \cdot \mathbf{e}_{mk}), \quad (3)$$

with a two-layer MLP (100 hidden dimensions).

If entity linking (EL) supervision is available, we can compute a loss with the gold entity $e_{mg}$. The exact form of the loss depends on the KB, and we use both log-likelihood,

$$\mathcal{L}_{\text{EL}} = -\sum_m \log \left( \frac{\exp(\psi_{mg})}{\sum_k \exp(\psi_{mk})} \right), \quad (4)$$

and max-margin,

$$\mathcal{L}_{\text{EL}} = \max(0, \gamma - \psi_{mg}) + \sum_{e_{mk} \neq e_{mg}} \max(0, \gamma + \psi_{mk}), \quad (5)$$

formulations (see Sec. 4.1 for details).

**Knowledge enhanced entity-span representations** KnowBert next injects the KB entity information into the mention-span representations computed from BERT vectors ($\mathbf{s}_m^e$) to form entity-span representations. For a given span $m$, we first disregard all candidate entities with score $\psi$ below a fixed threshold, and softmax normalize the remaining scores:

$$\tilde{\psi}_{mk} = \begin{cases} \dfrac{\exp(\psi_{mk})}{\sum_{\psi_{mk} \geq \delta} \exp(\psi_{mk})}, & \psi_{mk} \geq \delta \\ 0, & \psi_{mk} < \delta. \end{cases}$$

[1] We found a small transformer layer with four attention heads and a 1024 feed-forward hidden dimension was sufficient, significantly smaller than each of the layers in BERT. Early experiments demonstrated the effectiveness of this layer with improved entity linking performance.

Then the weighted entity embedding is

$$\tilde{\mathbf{e}}_m = \sum_k \tilde{\psi}_{mk} \mathbf{e}_{mk}.$$

If all entity linking scores are below the threshold $\delta$, we substitute a special `NULL` embedding for $\tilde{\mathbf{e}}_m$. Finally, the entity-span representations are updated with the weighted entity embeddings

$$\mathbf{s}'^e_m = \mathbf{s}^e_m + \tilde{\mathbf{e}}_m, \tag{6}$$

which are packed into a matrix $\mathbf{S}'^e \in \mathbb{R}^{C \times E}$.

**Recontextualization** After updating the entity-span representations with the weighted entity vectors, KnowBert uses them to recontextualize the word piece representations. This is accomplished using a modified transformer layer that substitutes the multi-headed self-attention with a multi-headed attention between the projected word piece representations and knowledge enhanced entity-span vectors. As introduced by Vaswani et al. (2017), the contextual embeddings $\mathbf{H}_i$ are used for the query, key, and value in multi-headed self-attention. The word-to-entity-span attention in KnowBert substitutes $\mathbf{H}_i^{\text{proj}}$ for the query, and $\mathbf{S}'^e$ for both the key and value:

$$\mathbf{H}_i^{'\text{proj}} = \text{MLP}(\text{MultiHeadAttn}(\mathbf{H}_i^{\text{proj}}, \mathbf{S}'^e, \mathbf{S}'^e)).$$

This allows each word piece to attend to all entity-spans in the context, so that it can propagate entity information over long contexts. After the multi-headed word-to-entity-span attention, we run a position-wise MLP analogous to the standard transformer layer.[2]

Finally, $\mathbf{H}'^{\text{proj}}_i$ is projected back to the BERT dimension with a linear transformation and a residual connection added:

$$\mathbf{H}'_i = \mathbf{H}'^{\text{proj}}_i \mathbf{W}_2^{\text{proj}} + \mathbf{b}_2^{\text{proj}} + \mathbf{H}_i \tag{7}$$

**Alignment of BERT and entity vectors** As KnowBert does not place any restrictions on the entity embeddings, it is essential to align them with the pretrained BERT contextual representations. To encourage this alignment we initialize $\mathbf{W}_2^{\text{proj}}$ as the matrix inverse of $\mathbf{W}_1^{\text{proj}}$ (Eq. 1). The use of dot product similarity (Eq. 3) and residual connection (Eq. 7) further aligns the entity-span representations with entity embeddings.

[2] As for the multi-headed entity-span self-attention, we found a small transformer layer to be sufficient, with four attention heads and 1024 hidden units in the MLP.

**Algorithm 1:** KnowBert training method

**Input:** Pretrained BERT and $J$ KBs
**Output:** KnowBert
**for** $j = 1 \ldots J$ **do**
  Compute entity embeddings for $\text{KB}_j$
  **if** EL supervision available **then**
    Freeze all network parameters except those in (Eq. 1–3)
    Train to convergence using (Eq. 4) or (Eq. 5)
  **end**
  Initialize $\mathbf{W}_2^{\text{proj}}$ as $(\mathbf{W}_1^{\text{proj}})^{-1}$
  Unfreeze all parameters except entity embeddings
  Minimize $\mathcal{L}_{\text{KnowBert}} = \mathcal{L}_{\text{BERT}} + \sum_{i=1}^{j} \mathcal{L}_{\text{EL}_i}$
**end**

### 3.4 Training Procedure

Our training regime incrementally pretrains increasingly larger portions of KnowBert before fine-tuning all trainable parameters in a multitask setting with any available EL supervision. It is similar in spirit to the "chain-thaw" approach in Felbo et al. (2017), and is summarized in Alg. 1.

We assume access to a pretrained BERT model and one or more KBs with their entity candidate selectors. To add the first KB, we begin by pretraining entity embeddings (if not already provided from another source), then freeze them in all subsequent training, including task-specific fine-tuning. If EL supervision is available, it is used to pretrain the KB specific EL parameters, while freezing the remainder of the network. Finally, the entire network is fine-tuned to convergence by minimizing

$$\mathcal{L}_{\text{KnowBert}} = \mathcal{L}_{\text{BERT}} + \mathcal{L}_{\text{EL}}.$$

We apply gradient updates to homogeneous batches randomly sampled from either the unlabeled corpus or EL supervision.

To add a second KB, we repeat the process, inserting it in any layer above the first one. When adding a KB, the BERT layers above it will experience large gradients early in training, as they are subject to the randomly initialized parameters associated with the new KB. They are thus expected to move further from their pretrained values before convergence compared to parameters below the KB. By adding KBs from bottom to top, we

| System | PPL | Wikidata MRR | # params. masked LM | # params. KAR | # params. entity embed. | Fwd. / Bwd. time |
|---|---|---|---|---|---|---|
| $\text{BERT}_{\text{BASE}}$ | 5.5 | 0.09 | 110 | 0 | 0 | 0.25 |
| $\text{BERT}_{\text{LARGE}}$ | 4.5 | 0.11 | 336 | 0 | 0 | 0.75 |
| KnowBert-Wiki | 4.3 | 0.26 | 110 | 2.4 | 141 | 0.27 |
| KnowBert-WordNet | 4.1 | 0.22 | 110 | 4.9 | 265 | 0.31 |
| KnowBert-W+W | **3.5** | **0.31** | 110 | 7.3 | 406 | 0.33 |

Table 1: Comparison of masked LM perplexity, Wikidata probing MRR, and number of parameters (in millions) in the masked LM (word piece embeddings, transformer layers, and output layers), KAR, and entity embeddings for BERT and KnowBert. The table also includes the total time to run one forward and backward pass (in seconds) on a TITAN Xp GPU (12 GB RAM) for a batch of 32 sentence pairs with total length 80 word pieces. Due to memory constraints, the $\text{BERT}_{\text{LARGE}}$ batch is accumulated over two smaller batches.

minimize disruption of the network and decrease the likelihood that training will fail. See Sec. 4.1 for details of where each KB was added.

The entity embeddings and selected candidates contain lexical information (especially in the case of WordNet), that will make the masked LM predictions significantly easier. To prevent leaking into the masked word pieces, we adopt the BERT strategy and replace all entity candidates from the selectors with a special `[MASK]` entity if the candidate mention span overlaps with a masked word piece.[3] This prevents KnowBert from relying on the selected candidates to predict masked word pieces.

## 4 Experiments

### 4.1 Experimental Setup

We used the English uncased $\text{BERT}_{\text{BASE}}$ model (Devlin et al., 2019) to train three versions of KnowBert: KnowBert-Wiki, KnowBert-WordNet, and KnowBert-W+W that includes both Wikipedia and WordNet.

**KnowBert-Wiki** The entity linker in KnowBert-Wiki borrows both the entity candidate selectors and embeddings from Ganea and Hofmann (2017). The candidate selectors and priors are a combination of CrossWikis, a large, precomputed dictionary that combines statistics from Wikipedia and a web corpus (Spitkovsky and Chang, 2012), and the YAGO dictionary (Hoffart et al., 2011). The entity embeddings use a skip-gram like objective (Mikolov et al., 2013b) to learn 300-dimensional embeddings of Wikipedia page titles directly from Wikipedia descriptions without using any explicit graph structure between nodes. As such, nodes in the KB are Wikipedia page titles, e.g., `Prince_(musician)`. Ganea and Hofmann (2017) provide pretrained embeddings for a subset of approximately 470K entities. Early experiments with embeddings derived from Wikidata relations[4] did not improve results.

We used the AIDA-CoNLL dataset (Hoffart et al., 2011) for supervision, adopting the standard splits. This dataset exhaustively annotates entity links for named entities of person, organization and location types, as well as a miscellaneous type. It does not annotate links to common nouns or other Wikipedia pages. At both train and test time, we consider all selected candidate spans and the top 30 entities, to which we add the special `NULL` entity to allow KnowBert to discriminate between actual links and false positive links from the selector. As such, KnowBert models both entity mention detection and disambiguation in an end-to-end manner. Eq. 5 was used as the objective.

**KnowBert-WordNet** Our WordNet KB combines synset metadata, lemma metadata and the relational graph. To construct the graph, we first extracted all synsets, lemmas, and their relationships from WordNet 3.0 using the `nltk` interface. After disregarding certain symmetric relationships (e.g., we kept the `hypernym` relationship, but removed the inverse `hyponym` relationship) we were left with 28 synset-synset and lemma-lemma relationships. From these, we constructed a graph where each node is either a synset or lemma, and intro-

[3] Following BERT, for 80% of masked word pieces all candidates are replaced with `[MASK]`, 10% are replaced with random candidates and 10% left unmasked.

[4] https://github.com/facebookresearch/PyTorch-BigGraph

| System | $F_1$ |
|---|---|
| WN-first sense baseline | 65.2 |
| ELMo | 69.2 |
| $\text{BERT}_{\text{BASE}}$ | 73.1 |
| $\text{BERT}_{\text{LARGE}}$ | 73.9 |
| KnowBert-WordNet | 74.9 |
| KnowBert-W+W | **75.1** |

Table 2: Fine-grained WSD $F_1$.

| System | AIDA-A | AIDA-B |
|---|---|---|
| Daiber et al. (2013) | 49.9 | 52.0 |
| Hoffart et al. (2011) | 68.8 | 71.9 |
| Kolitsas et al. (2018) | **86.6** | **82.6** |
| KnowBert-Wiki | 80.2 | 74.4 |
| KnowBert-W+W | 82.1 | 73.7 |

Table 3: End-to-end entity linking strong match, micro averaged $F_1$.

duced the special `lemma_in_synset` relationship to link synsets and lemmas. The candidate selector uses a rule-based lemmatizer without part-of-speech (POS) information.[5]

Our embeddings combine both the graph and synset glosses (definitions), as early experiments indicated improved perplexity when using both vs. just graph-based embeddings. We used TuckER (Balazevic et al., 2019) to compute 200-dimensional vectors for each synset and lemma using the relationship graph. Then, we extracted the gloss for each synset and used an off-the-shelf state-of-the-art sentence embedding method (Subramanian et al., 2018) to produce 2048-dimensional vectors. These are concatenated to the TuckER embeddings. To reduce the dimensionality for use in KnowBert, the frozen 2248-dimensional embeddings are projected to 200-dimensions with a learned linear transformation.

For supervision, we combined the SemCor word sense disambiguation (WSD) dataset (Miller et al., 1994) with all lemma example usages from WordNet[6] and link directly to synsets. The loss function is Eq. 4. At train time, we did not provide gold lemmas or POS tags, so KnowBert must learn to implicitly model coarse grained POS tags to disambiguate each word. At test time when evaluating we restricted candidate entities to just those matching the gold lemma and POS tag, consistent with the standard WSD evaluation.

**Training details** To control for the unlabeled corpus, we concatenated Wikipedia and the Books Corpus (Zhu et al., 2015) and followed the data preparation process in BERT with the exception of heavily biasing our dataset to shorter sequences of 128 word pieces for efficiency. Both KnowBert-Wiki and KnowBert-WordNet insert the KB between layers 10 and 11 of the 12-layer $\text{BERT}_{\text{BASE}}$ model. KnowBert-W+W adds the Wikipedia KB between layers 10 and 11, with WordNet between layers 11 and 12. Earlier experiments with KnowBert-WordNet in a lower layer had worse perplexity. We generally followed the fine-tuning procedure in Devlin et al. (2019); see supplemental materials for details.

### 4.2 Intrinsic Evaluation

**Perplexity** Table 1 compares masked LM perplexity for KnowBert with $\text{BERT}_{\text{BASE}}$ and $\text{BERT}_{\text{LARGE}}$. To rule out minor differences due to our data preparation, the BERT models are fine-tuned on our training data before being evaluated. Overall, KnowBert improves the masked LM perplexity, with all KnowBert models outperforming $\text{BERT}_{\text{LARGE}}$, despite being derived from $\text{BERT}_{\text{BASE}}$.

**Factual recall** To test KnowBert's ability to recall facts from the KBs, we extracted 90K tuples from Wikidata (Vrandečić and Krötzsch, 2014) for 17 different relationships such as `companyFoundedBy`. Each tuple was written into natural language such as "Adidas was founded by Adolf Dassler" and used to construct two test instances, one that masks out the subject and one that masks the object. Then, we evaluated whether a model could recover the masked entity by computing the mean reciprocal rank (MRR) of the masked word pieces. Table 1 displays a summary of the results (see supplementary material for results across all relationship types). Overall, KnowBert-Wiki is significantly better at recalling facts than both $\text{BERT}_{\text{BASE}}$ and $\text{BERT}_{\text{LARGE}}$, with KnowBert-W+W better still.

**Speed** KnowBert is almost as fast as $\text{BERT}_{\text{BASE}}$ (8% slower for KnowBert-Wiki, 32% for KnowBert-W+W) despite adding a large number

[5] https://spacy.io/

[6] To provide a fair evaluation on the WiC dataset which is partially based on the same source, we excluded all WiC train, development and test instances.

| System | LM | P | R | $F_1$ |
|---|---|---|---|---|
| Zhang et al. (2018) | — | 69.9 | 63.3 | 66.4 |
| Alt et al. (2019) | GPT | 70.1 | 65.0 | 67.4 |
| Shi and Lin (2019) | $\text{BERT}_{\text{BASE}}$ | 73.3 | 63.1 | 67.8 |
| Zhang et al. (2019) | $\text{BERT}_{\text{BASE}}$ | 70.0 | 66.1 | 68.0 |
| Soares et al. (2019) | $\text{BERT}_{\text{LARGE}}$ | — | — | 70.1 |
| Soares et al. (2019) | $\text{BERT}_{\text{LARGE}}$† | — | — | **71.5** |
| KnowBert-W+W | $\text{BERT}_{\text{BASE}}$ | 71.6 | 71.4 | **71.5** |

Table 4: Single model test set results on the TACRED relationship extraction dataset. † with MTB pretraining.

| System | LM | $F_1$ |
|---|---|---|
| Wang et al. (2016) | — | 88.0 |
| Wang et al. (2019b) | $\text{BERT}_{\text{BASE}}$ | 89.0 |
| Soares et al. (2019) | $\text{BERT}_{\text{LARGE}}$ | 89.2 |
| Soares et al. (2019) | $\text{BERT}_{\text{LARGE}}$† | **89.5** |
| KnowBert-W+W | $\text{BERT}_{\text{BASE}}$ | 89.1 |

Table 5: Test set $F_1$ for SemEval 2010 Task 8 relationship extraction. † with MTB pretraining.

of (frozen) parameters in the entity embeddings (Table 1). KnowBert is much faster than $\text{BERT}_{\text{LARGE}}$. By taking advantage of the already high capacity model, the number of trainable parameters added by KnowBert is a fraction of the total parameters in BERT. The faster speed is partially due to the entity parameter efficiency in KnowBert as only as small fraction of parameters in the entity embeddings are used for any given input due to the sparse linker. Our candidate generators consider the top 30 candidates and produce approximately $O(\text{number tokens})$ candidate spans. For a typical 25 token sentence, approximately 2M entity embedding parameters are actually used. In contrast, $\text{BERT}_{\text{LARGE}}$ uses the majority of its 336M parameters for each input.

**Integrated EL** It is also possible to evaluate the performance of the integrated entity linkers inside KnowBert using diagnostic probes without any further fine-tuning. As these were trained in a multitask setting primarily with raw text, we do not *a priori* expect high performance as they must balance specializing for the entity linking task and learning general purpose representations suitable for language modeling.

Table 2 displays fine-grained WSD $F_1$ using the evaluation framework from Navigli et al. (2017) and the ALL dataset (combing SemEval 2007, 2013, 2015 and Senseval 2 and 3). By linking to nodes in our WordNet graph and restricting to gold lemmas at test time we can recast the WSD task under our general entity linking framework. The ELMo and BERT baselines use a nearest neighbor approach trained on the SemCor dataset, similar to the evaluation in Melamud et al. (2016), which has previously been shown to be competitive with task-specific architectures (Raganato et al., 2017). As can be seen, KnowBert provides competitive performance, and KnowBert-W+W is able to match the performance of KnowBert-WordNet despite incorporating both Wikipedia and WordNet.

Table 3 reports end-to-end entity linking performance for the AIDA-A and AIDA-B datasets. Here, KnowBert's performance lags behind the current state-of-the-art model from Kolitsas et al. (2018), but still provides strong performance compared to other established systems such as AIDA (Hoffart et al., 2011) and DBpedia Spotlight (Daiber et al., 2013). We believe this is due to the selective annotation in the AIDA data that only annotates named entities. The CrossWikis-based candidate selector used in KnowBert generates candidate mentions for all entities including common nouns from which KnowBert may be learning to extract information, at the detriment of specializing to maximize linking performance for AIDA.

### 4.3 Downstream Tasks

This section evaluates KnowBert on downstream tasks to validate that the addition of knowledge improves performance on tasks expected to benefit from it. Given the overall superior performance of KnowBert-W+W on the intrinsic evaluations, we focus on it exclusively for evaluation in this section. The main results are included in this section; see the supplementary material for full details.

The baselines we compare against are $\text{BERT}_{\text{BASE}}$, $\text{BERT}_{\text{LARGE}}$, the pre-BERT state of the art, and two contemporaneous papers that add similar types of knowledge to BERT. ERNIE (Zhang et al., 2019) uses TAGME (Ferragina and Scaiella, 2010) to link entities to Wikidata, retrieves the associated entity embeddings, and fuses them into $\text{BERT}_{\text{BASE}}$ by fine-tuning. Soares et al. (2019) learns relationship representations by fine-tuning $\text{BERT}_{\text{LARGE}}$ with large scale "matching the blanks" (MTB) pretraining using entity linked text.

| System | Accuracy |
|---|---|
| ELMo† | 57.7 |
| $\text{BERT}_{\text{BASE}}$† | 65.4 |
| $\text{BERT}_{\text{LARGE}}$† | 65.5 |
| $\text{BERT}_{\text{LARGE}}$†† | 69.5 |
| KnowBert-W+W | **70.9** |

Table 6: Test set results for the WiC dataset (v1.0).
†Pilehvar and Camacho-Collados (2019)
††Wang et al. (2019a)

**Relation extraction** Our first task is relation extraction using the TACRED (Zhang et al., 2017) and SemEval 2010 Task 8 (Hendrickx et al., 2009) datasets. Systems are given a sentence with marked a subject and object, and asked to predict which of several different relations (or no relation) holds. Following Soares et al. (2019), our KnowBert model uses special entity tokens `[E1]`, `[/E1]`, `[E2]`, `[/E2]` to mark the location of the subject and object in the input sentence, then concatenates the contextual word representations for `[E1]` and `[E2]` to predict the relationship. For TACRED, we also encode the subject and object types with special tokens and concatenate them to the end of the sentence.

For TACRED (Table 4), KnowBert-W+W significantly outperforms the comparable $\text{BERT}_{\text{BASE}}$ systems including ERNIE by 3.5%, improves over $\text{BERT}_{\text{LARGE}}$ by 1.4%, and is able to match the performance of the relationship specific MTB pretraining in Soares et al. (2019). For SemEval 2010 Task 8 (Table 5), KnowBert-W+W $F_1$ falls between the entity aware $\text{BERT}_{\text{BASE}}$ model from Wang et al. (2019b), and the $\text{BERT}_{\text{LARGE}}$ model from Soares et al. (2019).

**Words in Context (WiC)** WiC (Pilehvar and Camacho-Collados, 2019) is a challenging task that presents systems with two sentences both containing a word with the same lemma and asks them to determine if they are from the same sense or not. It is designed to test the quality of contextual word representations. We follow standard practice and concatenate both sentences with a `[SEP]` token and fine-tune the `[CLS]` embedding. As shown in Table 6, KnowBert-W+W sets a new state of the art for this task, improving over $\text{BERT}_{\text{LARGE}}$ by 1.4% and reducing the relative gap to 80% human performance by 13.3%.

| System | P | R | $F_1$ |
|---|---|---|---|
| UFET | 68.8 | 53.3 | 60.1 |
| $\text{BERT}_{\text{BASE}}$ | 76.4 | 71.0 | 73.6 |
| ERNIE | 78.4 | 72.9 | 75.6 |
| KnowBert-W+W | 78.6 | 73.7 | **76.1** |

Table 7: Test set results for entity typing using the nine general types from (Choi et al., 2018).

**Entity typing** We also evaluated KnowBert-W+W using the entity typing dataset from Choi et al. (2018). To directly compare to ERNIE, we adopted the evaluation protocol in Zhang et al. (2019) which considers the nine general entity types.[7] Our model marks the location of a target span with the special `[E]` and `[/E]` tokens and uses the representation of the `[E]` token to predict the type. As shown in Table 7, KnowBert-W+W shows an improvement of 0.6% $F_1$ over ERNIE and 2.5% over $\text{BERT}_{\text{BASE}}$.

## 5 Conclusion

We have presented an efficient and general method to insert prior knowledge into a deep neural model. Intrinsic evaluations demonstrate that the addition of WordNet and Wikipedia to BERT improves the quality of the masked LM and significantly improves its ability to recall facts. Downstream evaluations demonstrate improvements for relationship extraction, entity typing and word sense disambiguation datasets. Future work will involve incorporating a diverse set of domain specific KBs for specialized NLP applications.

## Acknowledgements

The authors acknowledge helpful feedback from anonymous reviewers and the AllenNLP team. This research was funded in part by the NSF under awards IIS-1817183 and CNS-1730158.

## References

Sungjin Ahn, Heeyoul Choi, Tanel Pärnamaa, and Yoshua Bengio. 2017. A neural knowledge language model. arXiv:1608.00318.

Christoph Alt, Marc Hübner, and Leonhard Hennig. 2019. Improving relation extraction by pre-trained language representations. In *AKBC*.

[7]Data obtained from https://github.com/thunlp/ERNIE

Ivana Balazevic, Carl Allen, and Timothy M. Hospedales. 2019. TuckER: Tensor factorization for knowledge graph completion. In *EMNLP*.

Lisa Bauer, Yicheng Wang, and Mohit Bansal. 2018. Commonsense for generative multi-hop question answering tasks. In *EMNLP*.

Olivier Bodenreider. 2004. The Unified Medical Language System (UMLS): integrating biomedical terminology. *Nucleic Acids Research*, 32 Database issue:D267–70.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NeurIPS*.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Diana Inkpen, and Si Wei. 2018. Neural natural language inference models enhanced with external knowledge. In *ACL*.

Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A unified model for word sense representation and disambiguation. In *EMNLP*.

Eunsol Choi, Omer Levy, Yejin Choi, and Luke Zettlemoyer. 2018. Ultra-fine entity typing. In *ACL*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*.

Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised sequence learning. In *NeurIPS*.

Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N. Mendes. 2013. Improving efficiency and accuracy in multilingual entity extraction. In *I-SEMANTICS*.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *AAAI*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *EMNLP*.

Paolo Ferragina and Ugo Scaiella. 2010. TAGME: on-the-fly annotation of short text fragments (by wikipedia entities). In *CIKM*.

Octavian-Eugen Ganea and Thomas Hofmann. 2017. Deep joint entity disambiguation with local neural attention. In *EMNLP*.

Nitish Gupta, Sameer Singh, and Dan Roth. 2017. Entity linking via joint encoding of types, descriptions, and context. In *EMNLP*.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2009. SemEval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *HLT-NAACL*.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *EMNLP*.

Yangfeng Ji, Chenhao Tan, Sebastian Martschat, Yejin Choi, and Noah A Smith. 2017. Dynamic entity representations in neural language models. In *EMNLP*.

Nikolaos Kolitsas, Octavian-Eugen Ganea, and Thomas Hofmann. 2018. End-to-end neural entity linking. In *CoNLL*.

Kenton Lee, Luheng He, Mike Lewis, and Luke S. Zettlemoyer. 2017. End-to-end neural coreference resolution. In *EMNLP*.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*.

Xiao Ling, Sameer Singh, and Daniel S. Weld. 2015. Design challenges for entity linking. *Transactions of the Association for Computational Linguistics*, 3:315–328.

Robert L Logan, Nelson F. Liu, Matthew E. Peters, Matthew Ph Gardner, and Sameer Singh. 2019. Barack's wife hillary: Using knowledge graphs for fact-aware language modeling. In *ACL*.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional LSTM. In *CoNLL*.

Todor Mihaylov and Anette Frank. 2018. Knowledgeable reader: Enhancing cloze-style reading comprehension with external commonsense knowledge. In *ACL*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. arXiv:1301.3781.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *NeurIPS*.

George A Miller. 1995. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41.

George A. Miller, Martin Chodorow, Shari Landes, Claudia Leacock, and Robert G. Thomas. 1994. Using a semantic concordance for sense identification. In *HLT*.

Roberto Navigli, José Camacho-Collados, and Alessandro Raganato. 2017. Word sense disambiguation: A unified evaluation framework and empirical comparison. In *EACL*.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *ICML*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *EMNLP*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*.

Mohammad Taher Pilehvar and José Camacho-Collados. 2019. WiC: the word-in-context dataset for evaluating context-sensitive meaning representations. In *NAACL-HLT*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Alessandro Raganato, Claudio Delli Bovi, and Roberto Navigli. 2017. Neural sequence learning models for word sense disambiguation. In *EMNLP*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL*.

Peng Shi and Jimmy Lin. 2019. Simple BERT models for relation extraction and semantic role labeling.

Livio B. Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. 2019. Matching the blanks: Distributional similarity for relation learning. In *ACL*.

Robert Speer, Joshua Chin, and Catherine Havasi. 2017. ConceptNet 5.5: An open multilingual graph of general knowledge. In *AAAI*.

Valentin I. Spitkovsky and Angel X. Chang. 2012. A cross-lingual dictionary for English Wikipedia concepts. In *LREC*.

Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J Pal. 2018. Learning general purpose distributed sentence representations via large scale multi-task learning. In *ICLR*.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan R. Salakhutdinov, and William W. Cohen. 2018. Open domain question answering using early fusion of knowledge bases and text. In *EMNLP*.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *ICML*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019a. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. arXiv:1905.00537.

Chao Wang and Hui Jiang. 2019. Explicit utilization of general knowledge in machine reading comprehension. In *ACL*.

Haoyu Wang, Ming Tan, Mo Yu, Shiyu Chang, Dakuo Wang, Kun Xu, Xiaoxiao Guo, and Saloni Potdar. 2019b. Extracting multiple-relations in one-pass with pre-trained transformers. In *ACL*.

Linlin Wang, Zhu Cao, Gerard de Melo, and Zhiyuan Liu. 2016. Relation classification via multi-level attention CNNs. In *ACL*.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014a. Knowledge graph and text jointly embedding. In *EMNLP*.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014b. Knowledge graph embedding by translating on hyperplanes. In *AAAI*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv:1609.08144.

Han Xiao, Minlie Huang, and Xiaoyan Zhu. 2016. From one point to a manifold: knowledge graph embedding for precise link prediction. In *AAAI*.

Ruobing Xie, Zhiyuan Liu, J. J. Jia, Huanbo Luan, and Maosong Sun. 2016. Representation learning of knowledge graphs with entity descriptions. In *AAAI*.

An Yang, Quan Wang, Jing Liu, Kai Liu, Yajuan Lyu, Hua Wu, Qiaoqiao She, and Sujian Li. 2019. Enhancing pre-trained language representations with rich knowledge for machine reading comprehension. In *ACL*.

Bishan Yang and Tom Michael Mitchell. 2017. Leveraging knowledge bases in LSTMs for improving machine reading. In *ACL*.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. arXiv:1412.6575.

Zichao Yang, Phil Blunsom, Chris Dyer, and Wang Ling. 2017. Reference-aware language models. In *EMNLP*.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *EMNLP*.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. 2017. Position-aware attention and supervised data improve slot filling. In *EMNLP*.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: Enhanced language representation with informative entities. In *ACL*.

Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan R. Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *ICCV*.

# How Contextual are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings

**Kawin Ethayarajh**[*]
Stanford University
kawin@stanford.edu

## Abstract

Replacing static word embeddings with contextualized word representations has yielded significant improvements on many NLP tasks. However, just how contextual are the contextualized representations produced by models such as ELMo and BERT? Are there infinitely many context-specific representations for each word, or are words essentially assigned one of a finite number of word-sense representations? For one, we find that the contextualized representations of all words are not isotropic in any layer of the contextualizing model. While representations of the same word in different contexts still have a greater cosine similarity than those of two different words, this self-similarity is much lower in upper layers. This suggests that upper layers of contextualizing models produce more context-specific representations, much like how upper layers of LSTMs produce more task-specific representations. In all layers of ELMo, BERT, and GPT-2, on average, less than 5% of the variance in a word's contextualized representations can be explained by a static embedding for that word, providing some justification for the success of contextualized representations.

## 1 Introduction

The application of deep learning methods to NLP is made possible by representing words as vectors in a low-dimensional continuous space. Traditionally, these word embeddings were *static*: each word had a single vector, regardless of context (Mikolov et al., 2013a; Pennington et al., 2014). This posed several problems, most notably that all senses of a polysemous word had to share the same representation. More recent work, namely deep neural language models such as ELMo (Peters et al., 2018) and BERT (Devlin et al., 2018), have successfully created *contextualized word representations*, word vectors that are sensitive to the context in which they appear. Replacing static embeddings with contextualized representations has yielded significant improvements on a diverse array of NLP tasks, ranging from question-answering to coreference resolution.

The success of contextualized word representations suggests that despite being trained with only a language modelling task, they learn highly transferable and task-agnostic properties of language. In fact, linear probing models trained on frozen contextualized representations can predict linguistic properties of words (e.g., part-of-speech tags) almost as well as state-of-the-art models (Liu et al., 2019a; Hewitt and Manning, 2019). Still, these representations remain poorly understood. For one, just how contextual are these contextualized word representations? Are there infinitely many context-specific representations that BERT and ELMo can assign to each word, or are words essentially assigned one of a finite number of word-sense representations?

We answer this question by studying the geometry of the representation space for each layer of ELMo, BERT, and GPT-2. Our analysis yields some surprising findings:

1. In all layers of all three models, the contextualized word representations of all words are not isotropic: they are not uniformly distributed with respect to direction. Instead, they are *anisotropic*, occupying a narrow cone in the vector space. The anisotropy in GPT-2's last layer is so extreme that two random words will on average have almost perfect cosine similarity! Given that isotropy has both theoretical and empirical benefits for static embeddings (Mu et al., 2018), the extent of anisotropy in contextualized represen-

[*]Work partly done at the University of Toronto.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 55–65,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

tations is surprising.

2. Occurrences of the same word in different contexts have non-identical vector representations. Where vector similarity is defined as cosine similarity, these representations are more dissimilar to each other in upper layers. This suggests that, much like how upper layers of LSTMs produce more task-specific representations (Liu et al., 2019a), upper layers of contextualizing models produce more context-specific representations.

3. Context-specificity manifests very differently in ELMo, BERT, and GPT-2. In ELMo, representations of words in the same sentence grow more similar to each other as context-specificity increases in upper layers; in BERT, they become more dissimilar to each other in upper layers but are still more similar than randomly sampled words are on average; in GPT-2, however, words in the same sentence are no more similar to each other than two randomly chosen words.

4. After adjusting for the effect of anisotropy, on average, less than 5% of the variance in a word's contextualized representations can be explained by their first principal component. This holds across all layers of all models. This suggests that contextualized representations do not correspond to a finite number of word-sense representations, and even in the best possible scenario, static embeddings would be a poor replacement for contextualized ones. Still, static embeddings created by taking the first principal component of a word's contextualized representations outperform GloVe and FastText embeddings on many word vector benchmarks.

These insights help justify why the use of contextualized representations has led to such significant improvements on many NLP tasks.

## 2 Related Work

**Static Word Embeddings** Skip-gram with negative sampling (SGNS) (Mikolov et al., 2013a) and GloVe (Pennington et al., 2014) are among the best known models for generating static word embeddings. Though they learn embeddings iteratively in practice, it has been proven that in theory, they both implicitly factorize a word-context matrix containing a co-occurrence statistic (Levy and Goldberg, 2014a,b). Because they create a single representation for each word, a notable problem with static word embeddings is that all senses of a polysemous word must share a single vector.

**Contextualized Word Representations** Given the limitations of static word embeddings, recent work has tried to create context-sensitive word representations. ELMo (Peters et al., 2018), BERT (Devlin et al., 2018), and GPT-2 (Radford et al., 2019) are deep neural language models that are fine-tuned to create models for a wide range of downstream NLP tasks. Their internal representations of words are called *contextualized word representations* because they are a function of the entire input sentence. The success of this approach suggests that these representations capture highly transferable and task-agnostic properties of language (Liu et al., 2019a).

ELMo creates contextualized representations of each token by concatenating the internal states of a 2-layer biLSTM trained on a bidirectional language modelling task (Peters et al., 2018). In contrast, BERT and GPT-2 are bi-directional and uni-directional transformer-based language models respectively. Each transformer layer of 12-layer BERT (base, cased) and 12-layer GPT-2 creates a contextualized representation of each token by attending to different parts of the input sentence (Devlin et al., 2018; Radford et al., 2019). BERT – and subsequent iterations on BERT (Liu et al., 2019b; Yang et al., 2019) – have achieved state-of-the-art performance on various downstream NLP tasks, ranging from question-answering to sentiment analysis.

**Probing Tasks** Prior analysis of contextualized word representations has largely been restricted to probing tasks (Tenney et al., 2019; Hewitt and Manning, 2019). This involves training linear models to predict syntactic (e.g., part-of-speech tag) and semantic (e.g., word relation) properties of words. Probing models are based on the premise that if a simple linear model can be trained to accurately predict a linguistic property, then the representations implicitly encode this information to begin with. While these analyses have found that contextualized representations encode semantic and syntactic information, they cannot answer *how contextual* these representations are, and to

what extent they can be replaced with static word embeddings, if at all. Our work in this paper is thus markedly different from most dissections of contextualized representations. It is more similar to Mimno and Thompson (2017), which studied the geometry of static word embedding spaces.

## 3 Approach

### 3.1 Contextualizing Models

The contextualizing models we study in this paper are ELMo, BERT, and GPT-2[1]. We choose the base cased version of BERT because it is most comparable to GPT-2 with respect to number of layers and dimensionality. The models we work with are all pre-trained on their respective language modelling tasks. Although ELMo, BERT, and GPT-2 have 2, 12, and 12 hidden layers respectively, we also include the input layer of each contextualizing model as its $0^{\text{th}}$ layer. This is because the $0^{\text{th}}$ layer is not contextualized, making it a useful baseline against which to compare the contextualization done by subsequent layers.

### 3.2 Data

To analyze contextualized word representations, we need input sentences to feed into our pre-trained models. Our input data come from the SemEval Semantic Textual Similarity tasks from years 2012 - 2016 (Agirre et al., 2012, 2013, 2014, 2015). We use these datasets because they contain sentences in which the same words appear in different contexts. For example, the word 'dog' appears in *"A panda dog is running on the road."* and *"A dog is trying to get bacon off his back."* If a model generated the same representation for 'dog' in both these sentences, we could infer that there was no contextualization; conversely, if the two representations were different, we could infer that they were contextualized to some extent. Using these datasets, we map words to the list of sentences they appear in and their index within these sentences. We do not consider words that appear in less than 5 unique contexts in our analysis.

### 3.3 Measures of Contextuality

We measure how contextual a word representation is using three different metrics: *self-similarity*, *intra-sentence similarity*, and *maximum explainable variance*.

[1]We use the pretrained models provided in an earlier version of the PyTorch-Transformers library.

**Definition 1** Let $w$ be a word that appears in sentences $\{s_1,...,s_n\}$ at indices $\{i_1,...,i_n\}$ respectively, such that $w = s_1[i_1] = ... = s_n[i_n]$. Let $f_\ell(s,i)$ be a function that maps $s[i]$ to its representation in layer $\ell$ of model $f$. The *self similarity* of $w$ in layer $\ell$ is

$$SelfSim_\ell(w) = \frac{1}{n^2-n}\sum_j\sum_{k\neq j}\cos(f_\ell(s_j,i_j),f_\ell(s_k,i_k)) \tag{1}$$

where cos denotes the cosine similarity. In other words, the *self-similarity* of a word $w$ in layer $\ell$ is the average cosine similarity between its contextualized representations across its $n$ unique contexts. If layer $\ell$ does not contextualize the representations at all, then $SelfSim_\ell(w) = 1$ (i.e., the representations are identical across all contexts). The more contextualized the representations are for $w$, the lower we would expect its self-similarity to be.

**Definition 2** Let $s$ be a sentence that is a sequence $\langle w_1,...,w_n\rangle$ of $n$ words. Let $f_\ell(s,i)$ be a function that maps $s[i]$ to its representation in layer $\ell$ of model $f$. The *intra-sentence similarity* of $s$ in layer $\ell$ is

$$\begin{aligned} IntraSim_\ell(s) &= \frac{1}{n}\sum_i\cos(\vec{s}_\ell, f_\ell(s,i)) \\ \text{where } \vec{s}_\ell &= \frac{1}{n}\sum_i f_\ell(s,i) \end{aligned} \tag{2}$$

Put more simply, the *intra-sentence similarity* of a sentence is the average cosine similarity between its word representations and the sentence vector, which is just the mean of those word vectors. This measure captures how context-specificity manifests in the vector space. For example, if both $IntraSim_\ell(s)$ and $SelfSim_\ell(w)$ are low $\forall\, w \in s$, then the model contextualizes words in that layer by giving each one a context-specific representation that is still distinct from all other word representations in the sentence. If $IntraSim_\ell(s)$ is high but $SelfSim_\ell(w)$ is low, this suggests a less nuanced contextualization, where words in a sentence are contextualized simply by making their representations converge in vector space.

**Definition 3** Let $w$ be a word that appears in sentences $\{s_1,...,s_n\}$ at indices $\{i_1,...,i_n\}$ respectively, such that $w = s_1[i_1] = ... = s_n[i_n]$. Let $f_\ell(s,i)$ be a function that maps $s[i]$ to its representation in layer $\ell$ of model $f$. Where $[f_\ell(s_1,i_1)...f_\ell(s_n,i_n)]$ is the *occurrence matrix* of $w$ and $\sigma_1...\sigma_m$ are the

first $m$ singular values of this matrix, the *maximum explainable variance* is

$$MEV_\ell(w) = \frac{\sigma_1^2}{\sum_i \sigma_i^2} \tag{3}$$

$MEV_\ell(w)$ is the proportion of variance in $w$'s contextualized representations for a given layer that can be explained by their first principal component. It gives us an upper bound on how well a static embedding could replace a word's contextualized representations. The closer $MEV_\ell(w)$ is to 0, the poorer a replacement a static embedding would be; if $MEV_\ell(w) = 1$, then a static embedding would be a perfect replacement for the contextualized representations.

### 3.4 Adjusting for Anisotropy

It is important to consider isotropy (or the lack thereof) when discussing contextuality. For example, if word vectors were perfectly isotropic (i.e., directionally uniform), then $SelfSim_\ell(w) = 0.95$ would suggest that $w$'s representations were poorly contextualized. However, consider the scenario where word vectors are so anisotropic that any two words have on average a cosine similarity of 0.99. Then $SelfSim_\ell(w) = 0.95$ would actually suggest the opposite – that $w$'s representations were well contextualized. This is because representations of $w$ in different contexts would on average be more dissimilar to each other than two randomly chosen words.

To adjust for the effect of anisotropy, we use three *anisotropic baselines*, one for each of our contextuality measures. For self-similarity and intra-sentence similarity, the baseline is the average cosine similarity between the representations of uniformly randomly sampled words from different contexts. The more anisotropic the word representations are in a given layer, the closer this baseline is to 1. For maximum explainable variance (MEV), the baseline is the proportion of variance in uniformly randomly sampled word representations that is explained by their first principal component. The more anisotropic the representations in a given layer, the closer this baseline is to 1: even for a random assortment of words, the principal component would be able to explain a large proportion of the variance.

Since contextuality measures are calculated for each layer of a contextualizing model, we calculate separate baselines for each layer as well. We then subtract from each measure its respective baseline to get the *anisotropy-adjusted contexuality measure*. For example, the anisotropy-adjusted self-similarity is

$$\begin{aligned} Baseline(f_\ell) &= \mathbb{E}_{x,y\sim U(\mathcal{O})}\left[\cos(f_\ell(x), f_\ell(y))\right] \\ SelfSim^*_\ell(w) &= SelfSim_\ell(w) - Baseline(f_\ell) \end{aligned} \tag{4}$$

where $\mathcal{O}$ is the set of all word occurrences and $f_\ell(\cdot)$ maps a word occurrence to its representation in layer $\ell$ of model $f$. Unless otherwise stated, references to contextuality measures in the rest of the paper refer to the anisotropy-adjusted measures, where both the raw measure and baseline are estimated with 1K uniformly randomly sampled word representations.

## 4 Findings

### 4.1 (An)Isotropy

**Contextualized representations are anisotropic in all non-input layers.** If word representations from a particular layer were isotropic (i.e., directionally uniform), then the average cosine similarity between uniformly randomly sampled words would be 0 (Arora et al., 2017). The closer this average is to 1, the more anisotropic the representations. The geometric interpretation of anisotropy is that the word representations all occupy a narrow cone in the vector space rather than being uniform in all directions; the greater the anisotropy, the narrower this cone (Mimno and Thompson, 2017). As seen in Figure 1, this implies that in almost all layers of BERT, ELMo and GPT-2, the representations of all words occupy a narrow cone in the vector space. The only exception is ELMo's input layer, which produces static character-level embeddings without using contextual or even positional information (Peters et al., 2018). It should be noted that not all static embeddings are necessarily isotropic, however; Mimno and Thompson (2017) found that skipgram embeddings, which are also static, are not isotropic.

**Contextualized representations are generally more anisotropic in higher layers.** As seen in Figure 1, for GPT-2, the average cosine similarity between uniformly randomly words is roughly 0.6 in layers 2 through 8 but increases exponentially from layers 8 through 12. In fact, word representations in GPT-2's last layer are so anisotropic that any two words have on average an almost perfect cosine similarity! This pattern holds for BERT and

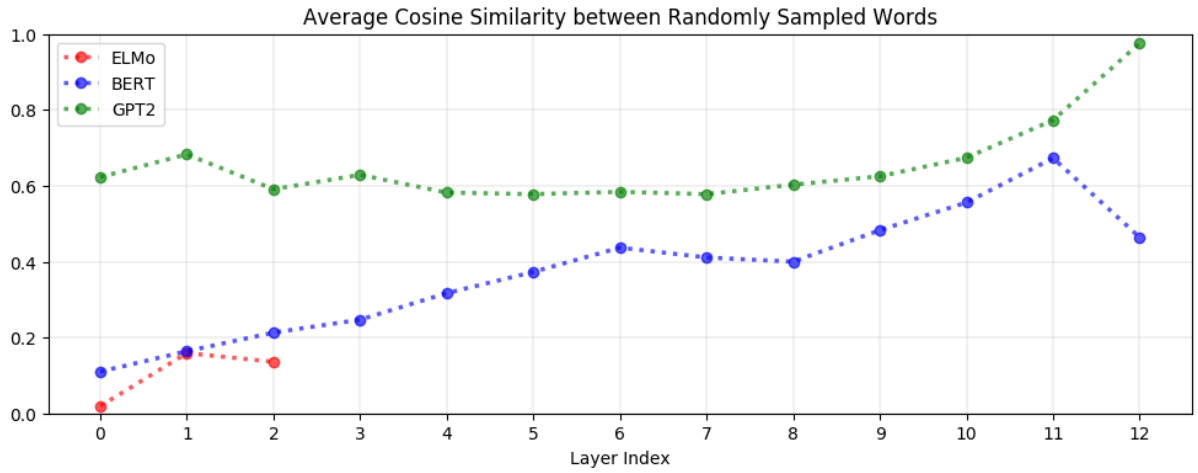


Figure 1: In almost all layers of BERT, ELMo, and GPT-2, the word representations are anisotropic (i.e., not directionally uniform): the average cosine similarity between uniformly randomly sampled words is non-zero. The one exception is ELMo's input layer; this is not surprising given that it generates character-level embeddings without using context. Representations in higher layers are generally more anisotropic than those in lower ones.

ELMo as well, though there are exceptions: for example, the anisotropy in BERT's penultimate layer is much higher than in its final layer.

Isotropy has both theoretical and empirical benefits for static word embeddings. In theory, it allows for stronger "self-normalization" during training (Arora et al., 2017), and in practice, subtracting the mean vector from static embeddings leads to improvements on several downstream NLP tasks (Mu et al., 2018). Thus the extreme degree of anisotropy seen in contextualized word representations – particularly in higher layers – is surprising. As seen in Figure 1, for all three models, the contextualized hidden layer representations are almost all more anisotropic than the input layer representations, which do not incorporate context. This suggests that high anisotropy is inherent to, or least a by-product of, the process of contextualization.

### 4.2 Context-Specificity

**Contextualized word representations are more context-specific in higher layers.** Recall from Definition 1 that the self-similarity of a word, in a given layer of a given model, is the average cosine similarity between its representations in different contexts, adjusted for anisotropy. If the self-similarity is 1, then the representations are not context-specific at all; if the self-similarity is 0, that the representations are maximally context-specific. In Figure 2, we plot the average self-similarity of uniformly randomly sampled words in each layer of BERT, ELMo, and GPT-2. For example, the self-similarity is 1.0 in ELMo's input layer because representations in that layer are static character-level embeddings.

In all three models, the higher the layer, the lower the self-similarity is on average. In other words, the higher the layer, the more context-specific the contextualized representations. This finding makes intuitive sense. In image classification models, lower layers recognize more generic features such as edges while upper layers recognize more class-specific features (Yosinski et al., 2014). Similarly, upper layers of LSTMs trained on NLP tasks learn more task-specific representations (Liu et al., 2019a). Therefore, it follows that upper layers of neural language models learn more context-specific representations, so as to predict the next word for a given context more accurately. Of all three models, representations in GPT-2 are the most context-specific, with those in GPT-2's last layer being almost maximally context-specific.

**Stopwords (e.g., *'the', 'of', 'to'*) have among the most context-specific representations.** Across all layers, stopwords have among the lowest self-similarity of all words, implying that their contextualized representations are among the most context-specific. For example, the words with the lowest average self-similarity across ELMo's layers are *'and', 'of', ''s', 'the'*, and *'to'*. This is relatively surprising, given that these words are not polysemous. This finding suggests that the variety

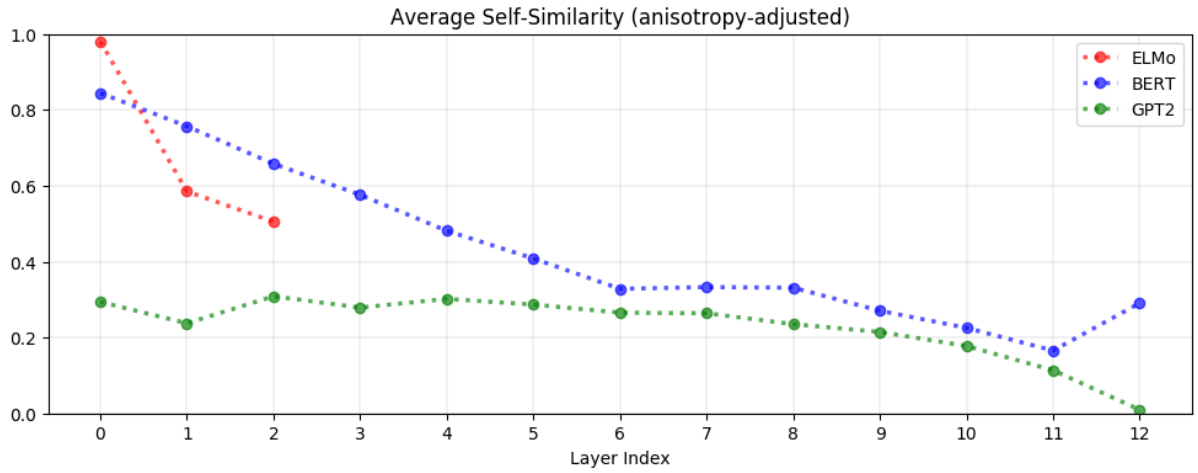


Figure 2: The average cosine similarity between representations of the same word in different contexts is called the word's *self-similarity* (see Definition 1). Above, we plot the average self-similarity of uniformly randomly sampled words after adjusting for anisotropy (see section 3.4). In all three models, the higher the layer, the lower the self-similarity, suggesting that contextualized word representations are more context-specific in higher layers.

of contexts a word appears in, rather than its inherent polysemy, is what drives variation in its contextualized representations. This answers one of the questions we posed in the introduction: ELMo, BERT, and GPT-2 are not simply assigning one of a finite number of word-sense representations to each word; otherwise, there would not be so much variation in the representations of words with so few word senses.

**Context-specificity manifests very differently in ELMo, BERT, and GPT-2.** As noted earlier, contextualized representations are more context-specific in upper layers of ELMo, BERT, and GPT-2. However, how does this increased context-specificity manifest in the vector space? Do word representations in the same sentence converge to a single point, or do they remain distinct from one another while still being distinct from their representations in other contexts? To answer this question, we can measure a sentence's intra-sentence similarity. Recall from Definition 2 that the intra-sentence similarity of a sentence, in a given layer of a given model, is the average cosine similarity between each of its word representations and their mean, adjusted for anisotropy. In Figure 3, we plot the average intra-sentence similarity of 500 uniformly randomly sampled sentences.

**In ELMo, words in the same sentence are more similar to one another in upper layers.** As word representations in a sentence become more context-specific in upper layers, the intra-sentence similarity also rises. This suggests that, in practice, ELMo ends up extending the intuition behind Firth's (1957) distributional hypothesis to the sentence level: that because words in the same sentence share the same context, their contextualized representations should also be similar.

**In BERT, words in the same sentence are more dissimilar to one another in upper layers.** As word representations in a sentence become more context-specific in upper layers, they drift away from one another, although there are exceptions (see layer 12 in Figure 3). However, in all layers, the average similarity between words in the same sentence is still greater than the average similarity between randomly chosen words (i.e., the anisotropy baseline). This suggests a more nuanced contextualization than in ELMo, with BERT recognizing that although the surrounding sentence informs a word's meaning, two words in the same sentence do not necessarily have a similar meaning because they share the same context.

**In GPT-2, word representations in the same sentence are no more similar to each other than randomly sampled words.** On average, the unadjusted intra-sentence similarity is roughly the same as the anisotropic baseline, so as seen in Figure 3, the anisotropy-adjusted intra-sentence similarity is close to 0 in most layers of GPT-2. In fact, the intra-sentence similarity is highest in the input layer, which does not contextualize words at all. This is in contrast to ELMo and BERT, where the

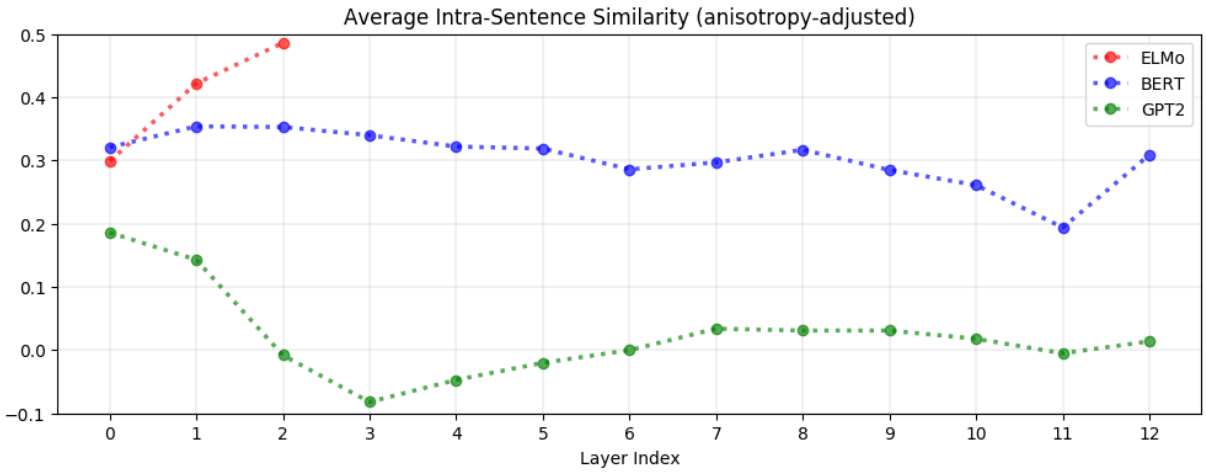


Figure 3: The *intra-sentence similarity* is the average cosine similarity between each word representation in a sentence and their mean (see Definition 2). Above, we plot the average intra-sentence similarity of uniformly randomly sampled sentences, adjusted for anisotropy. This statistic reflects how context-specificity manifests in the representation space, and as seen above, it manifests very differently for ELMo, BERT, and GPT-2.

average intra-sentence similarity is above 0.20 for all but one layer.

As noted earlier when discussing BERT, this behavior still makes intuitive sense: two words in the same sentence do not necessarily have a similar meaning simply because they share the same context. The success of GPT-2 suggests that unlike anisotropy, which accompanies context-specificity in all three models, a high intra-sentence similarity is not inherent to contextualization. Words in the same sentence can have highly contextualized representations without those representations being any more similar to each other than two random word representations. It is unclear, however, whether these differences in intra-sentence similarity can be traced back to differences in model architecture; we leave this question as future work.

### 4.3 Static vs. Contextualized

**On average, less than 5% of the variance in a word's contextualized representations can be explained by a static embedding.** Recall from Definition 3 that the *maximum explainable variance* (MEV) of a word, for a given layer of a given model, is the proportion of variance in its contextualized representations that can be explained by their first principal component. This gives us an upper bound on how well a static embedding could replace a word's contextualized representations. Because contextualized representations are anisotropic (see section 4.1), much of the variation *across all words* can be explained by a single vector. We adjust for anisotropy by calculating the proportion of variance explained by the first principal component of uniformly randomly sampled word representations and subtracting this proportion from the raw MEV. In Figure 4, we plot the average anisotropy-adjusted MEV across uniformly randomly sampled words.

In no layer of ELMo, BERT, or GPT-2 can more than 5% of the variance in a word's contextualized representations be explained by a static embedding, on average. Though not visible in Figure 4, the raw MEV of many words is actually below the anisotropy baseline: i.e., a greater proportion of the variance across all words can be explained by a single vector than can the variance across all representations of a single word. Note that the 5% threshold represents the best-case scenario, and there is no theoretical guarantee that a word vector obtained using GloVe, for example, would be similar to the static embedding that maximizes MEV. This suggests that contextualizing models are not simply assigning one of a finite number of word-sense representations to each word – otherwise, the proportion of variance explained would be much higher. Even the average raw MEV is below 5% for all layers of ELMo and BERT; only for GPT-2 is the raw MEV non-negligible, being around 30% on average for layers 2 to 11 due to extremely high anisotropy.

**Principal components of contextualized representations in lower layers outperform GloVe and FastText on many benchmarks.** As noted

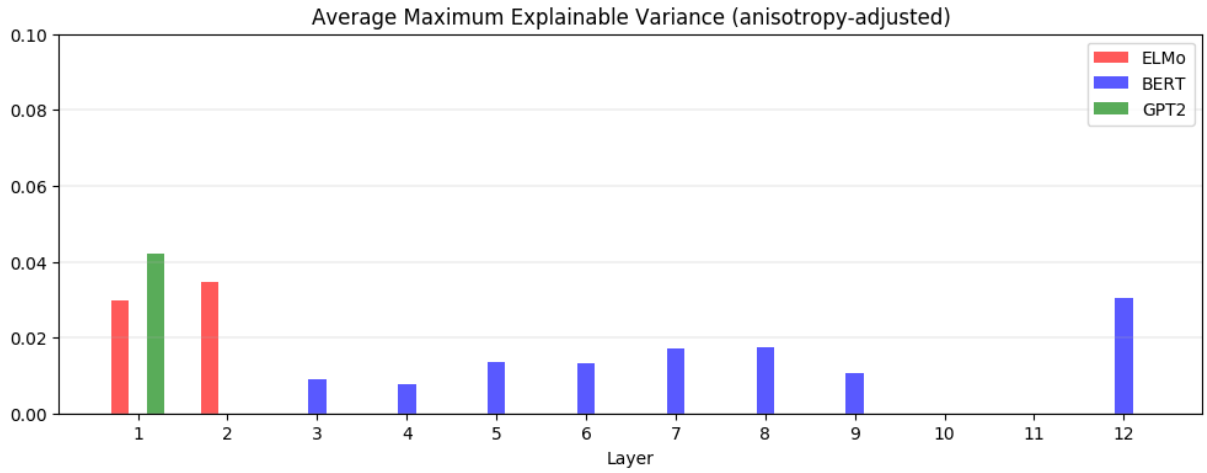


Figure 4: The *maximum explainable variance* (MEV) of a word is the proportion of variance in its contextualized representations that can be explained by their first principal component (see Definition 3). Above, we plot the average MEV of uniformly randomly sampled words after adjusting for anisotropy. In no layer of any model can more than 5% of the variance in a word's contextualized representations be explained by a static embedding.

| Static Embedding | SimLex999 | MEN | WS353 | RW | Google | MSR | SemEval2012(2) | BLESS | AP |
|---|---|---|---|---|---|---|---|---|---|
| GloVe | 0.194 | 0.216 | 0.339 | 0.127 | 0.189 | 0.312 | 0.097 | 0.390 | 0.308 |
| FastText | 0.239 | **0.239** | **0.432** | 0.176 | 0.203 | 0.289 | 0.104 | 0.375 | 0.291 |
| ELMo, Layer 1 | 0.276 | 0.167 | 0.317 | 0.148 | 0.170 | 0.326 | 0.114 | **0.410** | 0.308 |
| ELMo, Layer 2 | 0.215 | 0.151 | 0.272 | 0.133 | 0.130 | 0.268 | 0.132 | 0.395 | 0.318 |
| BERT, Layer 1 | 0.315 | 0.200 | 0.394 | **0.208** | **0.236** | **0.389** | **0.166** | 0.365 | **0.321** |
| BERT, Layer 2 | **0.320** | 0.166 | 0.383 | 0.188 | 0.230 | 0.385 | 0.149 | 0.365 | **0.321** |
| BERT, Layer 11 | 0.221 | 0.076 | 0.319 | 0.135 | 0.175 | 0.290 | 0.149 | 0.370 | 0.289 |
| BERT, Layer 12 | 0.233 | 0.082 | 0.325 | 0.144 | 0.184 | 0.307 | 0.144 | 0.360 | 0.294 |
| GPT-2, Layer 1 | 0.174 | 0.012 | 0.176 | 0.183 | 0.052 | 0.081 | 0.033 | 0.220 | 0.184 |
| GPT-2, Layer 2 | 0.135 | 0.036 | 0.171 | 0.180 | 0.045 | 0.062 | 0.021 | 0.245 | 0.184 |
| GPT-2, Layer 11 | 0.126 | 0.034 | 0.165 | 0.182 | 0.031 | 0.038 | 0.045 | 0.270 | 0.189 |
| GPT-2, Layer 12 | 0.140 | -0.009 | 0.113 | 0.163 | 0.020 | 0.021 | 0.014 | 0.225 | 0.172 |

Table 1: The performance of various static embeddings on word embedding benchmark tasks. The best result for each task is in bold. For the contextualizing models (ELMo, BERT, GPT-2), we use the first principal component of a word's contextualized representations in a given layer as its static embedding. The static embeddings created using ELMo and BERT's contextualized representations often outperform GloVe and FastText vectors.

earlier, we can create static embeddings for each word by taking the first principal component (PC) of its contextualized representations in a given layer. In Table 1, we plot the performance of these *PC static embeddings* on several benchmark tasks[2]. These tasks cover semantic similarity, analogy solving, and concept categorization: SimLex999 (Hill et al., 2015), MEN (Bruni et al., 2014), WS353 (Finkelstein et al., 2002), RW (Luong et al., 2013), SemEval-2012 (Jurgens et al., 2012), Google analogy solving (Mikolov et al., 2013a) MSR analogy solving (Mikolov et al., 2013b), BLESS (Baroni and Lenci, 2011) and AP (Almuhareb and Poesio, 2004). We leave out layers 3 - 10 in Table 1 because their performance is between those of Layers 2 and 11.

The best-performing PC static embeddings belong to the first layer of BERT, although those from the other layers of BERT and ELMo also outperform GloVe and FastText on most benchmarks. For all three contextualizing models, PC static embeddings created from lower layers are more effective those created from upper layers. Those created using GPT-2 also perform markedly worse than their counterparts from ELMo and BERT. Given that upper layers are much more context-specific than lower layers, and given that GPT-2's representations are more context-specific than ELMo and BERT's (see Figure 2), this suggests that the PCs of highly context-specific representations are less effective on traditional benchmarks. Those derived from less context-specific represen-

[2]The Word Embeddings Benchmarks package was used for evaluation.

tations, such as those from Layer 1 of BERT, are much more effective.

## 5 Future Work

Our findings offer some new directions for future work. For one, as noted earlier in the paper, Mu et al. (2018) found that making static embeddings more isotropic – by subtracting their mean from each embedding – leads to surprisingly large improvements in performance on downstream tasks. Given that isotropy has benefits for static embeddings, it may also have benefits for contextualized word representations, although the latter have already yielded significant improvements despite being highly anisotropic. Therefore, adding an anisotropy penalty to the language modelling objective – to encourage the contextualized representations to be more isotropic – may yield even better results.

Another direction for future work is generating static word representations from contextualized ones. While the latter offer superior performance, there are often challenges to deploying large models such as BERT in production, both with respect to memory and run-time. In contrast, static representations are much easier to deploy. Our work in section 4.3 suggests that not only it is possible to extract static representations from contextualizing models, but that these extracted vectors often perform much better on a diverse array of tasks compared to traditional static embeddings such as GloVe and FastText. This may be a means of extracting some use from contextualizing models without incurring the full cost of using them in production.

## 6 Conclusion

In this paper, we investigated how contextual contextualized word representations truly are. For one, we found that upper layers of ELMo, BERT, and GPT-2 produce more context-specific representations than lower layers. This increased context-specificity is always accompanied by increased anisotropy. However, context-specificity also manifests differently across the three models; the anisotropy-adjusted similarity between words in the same sentence is highest in ELMo but almost non-existent in GPT-2. We ultimately found that after adjusting for anisotropy, on average, less than 5% of the variance in a word's contextualized representations could be explained by a static embedding. This means that even in the best-case scenario, in all layers of all models, static word embeddings would be a poor replacement for contextualized ones. These insights help explain some of the remarkable success that contextualized representations have had on a diverse array of NLP tasks.

## Acknowledgments

We thank the anonymous reviewers for their insightful comments. We thank the Natural Sciences and Engineering Research Council of Canada (NSERC) for their financial support.

## References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel M Cer, Mona T Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Inigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, et al. 2015. Semeval-2015 task 2: Semantic textual similarity, English, Spanish and pilot on interpretability. In *Proceedings SemEval@ NAACL-HLT*. pages 252–263.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel M Cer, Mona T Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. Semeval-2014 task 10: Multilingual semantic textual similarity. In *Proceedings SemEval@ COLING*. pages 81–91.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. Sem 2013 shared task: Semantic textual similarity, including a pilot on typed-similarity. In *SEM 2013: The Second Joint Conference on Lexical and Computational Semantics*. Association for Computational Linguistics.

Eneko Agirre, Mona Diab, Daniel Cer, and Aitor Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*. Association for Computational Linguistics, pages 385–393.

Abdulrahman Almuhareb and Massimo Poesio. 2004. Attribute-based and value-based clustering: An evaluation. In *Proceedings of the 2004 conference on empirical methods in natural language processing*.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *International Conference on Learning Representations*.

Marco Baroni and Alessandro Lenci. 2011. How we blessed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*. Association for Computational Linguistics, pages 1–10.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *Journal of Artificial Intelligence Research* 49:1–47.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* .

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: The concept revisited. *ACM Transactions on information systems* 20(1):116–131.

John R Firth. 1957. A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis* .

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics* 41(4):665–695.

David A Jurgens, Peter D Turney, Saif M Mohammad, and Keith J Holyoak. 2012. Semeval-2012 task 2: Measuring degrees of relational similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*. Association for Computational Linguistics, pages 356–364.

Omer Levy and Yoav Goldberg. 2014a. Linguistic regularities in sparse and explicit word representations. In *Proceedings of the eighteenth conference on computational natural language learning*. pages 171–180.

Omer Levy and Yoav Goldberg. 2014b. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*. pages 2177–2185.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019a. Linguistic knowledge and transferability of contextual representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* .

Thang Luong, Richard Socher, and Christopher D Manning. 2013. Better word representations with recursive neural networks for morphology. In *SIGNLL Conference on Computational Natural Language Learning (CoNLL)*. pages 104–113.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013a. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*. pages 3111–3119.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 746–751.

David Mimno and Laure Thompson. 2017. The strange geometry of skip-gram with negative sampling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 2873–2878.

Jiaqi Mu, Suma Bhat, and Pramod Viswanath. 2018. All-but-the-top: Simple and effective postprocessing for word representations. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. pages 2227–2237.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners .

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R. Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R. Bowman, Dipanjan Das, and Ellie Pavlick. 2019. What do you learn from context? probing for sentence structure in contextualized word representations. In *International Conference on Learning Representations*. https://openreview.net/forum?id=SJzSgnRcKX.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237* .

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*. pages 3320–3328.

# Room to Glo: A Systematic Comparison of Semantic Change Detection Approaches with Word Embeddings

**Philippa Shoemark**[*♠♢], **Farhana Ferdousi Liza**[*♠♡], **Dong Nguyen**[♠♢],
**Scott A. Hale**[♠†], **Barbara McGillivray**[♠‡]
[♠]The Alan Turing Institute, UK, [♢]School of Informatics, University of Edinburgh, UK,
[♡]School of Computing, University of Kent, UK, [†]Oxford Internet Institute, University of Oxford, UK,
[‡]Faculty of Modern and Medieval Languages, University of Cambridge, UK
p.j.shoemark@ed.ac.uk, fl207@kent.ac.uk, dnguyen@turing.ac.uk,
scott.hale@oii.ox.ac.uk, bmcgillivray@turing.ac.uk

## Abstract

Word embeddings are increasingly used for the automatic detection of semantic change; yet, a robust evaluation and systematic comparison of the choices involved has been lacking. We propose a new evaluation framework for semantic change detection and find that (i) using the whole time series is preferable over only comparing between the first and last time points; (ii) independently trained and aligned embeddings perform better than continuously trained embeddings for long time periods; and (iii) that the reference point for comparison matters. We also present an analysis of the changes detected on a large Twitter dataset spanning 5.5 years.

## 1 Introduction

Semantic change, i.e., the change in the meanings of words, is inherent in language. A new meaning for a word can be added to the original one, become more or less prevalent, or even replace a former meaning (see Koch, 2016). An example is *lit*, which has gained a new sense of 'exciting' or 'awesome', via the extension of its long-established use as slang for 'intoxicated' to describe the vibrant environment in which acts of becoming intoxicated often occur.[1]

Automatically measuring semantic change can discover changes that would not be apparent from manual inspection. It can also facilitate the investigation of mechanisms driving semantic changes, e.g., how these changes are affected by language-internal and social factors. Moreover, there are direct benefits to applications, such as the detection of meaning shifts in polarized words to update sentiment lexicons and the detection of emerging word meanings to update dictionaries.

Word embeddings are increasingly used for automatic semantic change detection (Kutuzov et al., 2018). Words are mapped to low-dimensional vectors, and the semantic change of a word is then measured by comparing its vectors across time periods. Although word embeddings have emerged as one of the most popular approaches to measuring semantic change, researchers are faced with various decisions, including whether to train embeddings independently or continuously, which metric to use to measure change between two time periods, and which ranking approach to use for comparing semantic change candidates.

A major challenge in developing semantic change detection systems is obtaining ground truth data (Kutuzov et al., 2018), which has so far prevented a systematic evaluation of different approaches. Many studies rely on hand-picked examples (e.g., Wijaya and Yeniterzi, 2011; Rodda et al., 2017) or human judgements (e.g., Tredici et al., 2018). Some studies have performed evaluations based on dictionary data (e.g., Cook et al., 2014; Basile and McGillivray, 2018), manual annotation of dictionary senses in corpora (McGillivray et al., 2019), and manual annotation of word types (Kenter et al., 2015), but this approach is not well-suited for recent, yet-to-be-recorded changes.

In this paper, we present a new framework to evaluate semantic change detection systems (Section 5.1). We model multiple semantic change scenarios and compare the impact of different choices that are typical when using word embeddings to analyse semantic change (Section 5.2). Our framework is not specific to the use of word embeddings and can also support the evaluation of other approaches not considered in this paper.[2] We

*These authors contributed equally to the study.

[1]"What Does 'Lit' Mean?" Retrieved from https://www.merriam-webster.com/words-at-play/lit-meaning-origin. Accessed 2018-10-05.

[2]The dataset and all code for this paper is available at https://github.com/alan-turing-institute/room2glo.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 66–76,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

then apply the approaches to 5.5 years of Twitter data and provide an in-depth analysis of the top-ranked semantic change candidates (Section 6).

## 2 Related Work

There has been increasing interest in automatic semantic change detection (Tang, 2018; Kutuzov et al., 2018), using methods ranging from neural models to Bayesian learning (e.g., Frermann and Lapata, 2016), Temporal Random Indexing (e.g., Basile and McGillivray, 2018) and dynamic topic modelling (e.g., Blei and Lafferty, 2006). Word embeddings have been especially popular (e.g., Dubossarsky et al., 2017; Hamilton et al., 2016b), and recently Bamler and Mandt (2017) and Rudolph and Blei (2018) explored dynamic embeddings for semantic change detection by training a joint model over all time periods.

Most previous work analysed corpora spanning long time periods (e.g., a few centuries), such as the Google Books Ngrams corpus and the Corpus of Historical American English (e.g., Hamilton et al., 2016b). Recently short-term semantic changes have been studied, for example in Amazon Reviews (Kulkarni et al., 2015), scientific papers (Rudolph and Blei, 2018), news articles (Tang et al., 2016; Yao et al., 2018), and the UK Web Archive (Basile and McGillivray, 2018).

In this paper, we focus on social media: in particular, on Twitter. Semantic change in social media has only been lightly explored, with studies on Twitter (Kulkarni et al., 2015), the VKontake social network (Stewart et al., 2017), and Reddit (Tredici et al., 2018). In comparison to these studies, our data covers a longer time period and our evaluation more deeply explores the various choices involved in semantic change detection.

Much of the previous work on semantic change discovery has relied on qualitative evaluations of small samples from the output, case studies of a few well-known historical changes (e.g., Kim et al., 2014; Hamilton et al., 2016a,b; Stewart et al., 2017), or attested changes extracted from dictionaries (e.g., Rohrdantz et al., 2011; Cook et al., 2014; Basile and McGillivray, 2018). Some evaluations have been based on related tasks for which performance is expected to correlate, such as classifying the time period a text snippet belongs to (Mihalcea and Nastase, 2012) or predicting real-world events (Kutuzov et al., 2017).

Here we look for meaning changes over a short, recent time period. There is little existing literature on words that have undergone meaning change within the relevant time-frame, and language on social media is not always fully reflected in general language dictionaries. Moreover, even if we were able to obtain a substantial list of attested meaning changes, a system might still discover other valid meaning change candidates. Unfortunately, determining the validity of semantic change candidates is time-consuming, labour-intensive, and subjective; so, building on prior approaches (e.g., Kulkarni et al., 2015; Rosenfeld and Erk, 2018; Nguyen and Eisenstein, 2017), we introduce a new synthetic evaluation framework for semantic change detection.

Synthetic evaluation is especially important for short, recent time periods given the lack of other resources for evaluation, but it is also valuable for longer periods to detect hitherto unknown changes. Moreover, phenomena like seasonal trends are more likely to interfere with semantic change detection for short time periods, making this a challenging use case for semantic change detection. At the same time, it is an important use case in order to advance semantic change detection for contemporary data to be used to update lexicons, sentiment/polarity ratings, and other language resources.

## 3 Data

We collected tweets from Twitter's 'statuses/sample' streaming API endpoint from January 1, 2012, to June 30, 2017. There are a few minor gaps in our data due to occasional data collection issues. Most are a few minutes or at most a day, but one gap spans January and February 2015. Overall, our dataset consists of over 7 billion tweets sent during 1,889 days.

We use the Compact Language Detector version 2 (CLDv2),[3] following guidance from Graham et al. (2014), and we discard any tweets for which CLD detects less than 90% of the text to be in English, resulting in roughly 2.5 billion tweets. The remaining tweets are then lowercased, and usernames, urls, and non-alphanumeric characters (except emoji and hashtags) are removed. The text is then tokenized on whitespace. Digit-only tokens are replaced with '<NUM>'. Finally, we discard tweets that are duplicated within a given month, as tweets which are re-tweeted or copied verba-

[3] https://github.com/CLD2Owners/cld2

tim many times are not independent language samples and may exert undue influence on embeddings (Mikolov et al., 2018). Our final dataset consists of 1,696,142,020 tweets and 20,273,497,107 tokens.

## 4 Methods

Following the approach introduced by Kim et al. (2014) and adopted by Hamilton et al. (2016b) and others, we divide our dataset into discrete time periods, and for each time period $t$ we compute word embeddings, representing each word $w$ by a $d$-dimensional vector. We then compare the embeddings between different time periods to measure the semantic change of words. We use monthly bins, but the approach is applicable to time periods of any length, provided there is sufficient data in each bin to train quality embeddings.

### 4.1 Training Word Embeddings

We train word embeddings using gensim's (Řehůřek and Sojka, 2010) implementation of the continuous bag of words (CBOW; Mikolov et al., 2013) model.[4] Two evaluation tasks (word similarity using the dataset Wordsim353[5] and word analogy using the word test dataset[6]) were used to tune four hyperparameters, resulting in 200 dimensions, a window size of 10, 15 iterations, and a minimum frequency of 500 (per time-step). For all other hyperparameters we use gensim's default values.

### 4.2 Comparable Embeddings

To compare embeddings for a word between two time-points, the embeddings need to be in the same coordinate axes. We experiment with three approaches: (1) Training continuously by initializing the embeddings for a given time-step $t$ with the embeddings trained at the previous time-step $t-1$ (e.g., Kim et al., 2014); (2) Training embeddings for each time-step independently and post-hoc aligning them (e.g., Hamilton et al., 2016b; Kulkarni et al., 2015) using orthogonal Procrustes (as used by Hamilton et al., 2016b); and (3) combining continuous training and post-hoc alignment (as in Stewart et al., 2017).

### 4.3 Measuring Semantic Change

We compare two measures for quantifying a word's semantic change between two time points. The first is the cosine distance, a common approach in previous work (Hamilton et al., 2016b; Stewart et al., 2017; Dubossarsky et al., 2017; Kim et al., 2014). The second measure, introduced by Hamilton et al. (2016a), is based on comparing the neighbourhoods of the embeddings. For each time-step $t$, we first find the ordered set of word $w$'s $k$ nearest neighbours, based on cosine similarity. Following Hamilton et al. (2016a), we set $k = 25$. For any two time-steps, we then take the union $S$ of the two nearest neighbour sets and create a second-order vector $v_t$ where each entry $v_t^{(i)}$ contains the cosine similarity of target word $w$ to neighbouring word $S^{(i)}$ at time $t$. We then measure the cosine distance between these two second-order vectors.

### 4.4 Ranking Semantic Change Candidates

Our goal is not only to measure semantic changes for pre-selected words, but to identify which words out of the entire vocabulary have undergone the greatest or most significant semantic change. We compare several approaches to generating ranked lists of the 'most changed' words. The first only measures the change between two time-steps. The remaining approaches consider the whole time series. For the approaches that use the whole time-series, we limit the semantic change candidates to words that occur at least 500 times in at least 75% of the time-steps, simply condensing a word's time-series if there are gaps.

**Two-step approach** We first measure each word's semantic change between just two pre-selected time-steps (in this study, the first and final time-steps). This simple approach has been used in previous work, such as Kim et al. (2014).

**Change-point detection** Following Kulkarni et al. (2015), we choose one time-step $t_0$ as a reference and compute semantic change scores for each word with respect to $t_0$ at every other time-step $t_i$. Then, for each word $w$ and each time-step $t_i$, we compute a *mean-shift* score by partitioning $w$'s time series of semantic change scores at $t_i$, and

[4] We only report results using CBOW in this paper. We found similar trends when using the skip-gram model, which has been used in previous works on semantic change (e.g., Kim et al., 2014; Kulkarni et al., 2015; Hamilton et al., 2016b; Stewart et al., 2017; Tredici et al., 2018).

[5] http://www.cs.technion.ac.il/~gabr/resources/data/wordsim353/

[6] http://www.fit.vutbr.cz/~imikolov/rnnlm/word-test.v1.txt

calculating the difference between the means of the scores in the two partitions. Following Kulkarni et al. (2015), we use Monte Carlo permutation tests to estimate the statistical significance of mean-shift scores, and take the time-step with the lowest estimated p-value as the change point. Words are first sorted in descending order of the mean-shift scores of their estimated change points and then in ascending order of their p-values.

Kulkarni et al. standardized each word's cosine score for a given time-step relative to the mean score across all words at that time-step. This is meant to help control for corpus artefacts, e.g., shifting sampling biases over time, but its impact has not been demonstrated yet. We compare the results of ranking words without standardization (*raw* scores) and with standardization ($z$ scores).

**Global trend detection** We also compare three approaches to detect global trends in the same time series of semantic change scores as the change point detection methods. The first approach is fitting a linear regression model $d_i = \alpha + \beta t_i + \epsilon_i$, where $d_i$ is semantic change distance scores, $t_i$ is time periods $\{1, ..., n\}$, and $\epsilon_i$ is error. We rank the words based on their absolute $\beta$ values (slopes), which gives the semantic change per time period under the assumption of a linear relationship.

We also experiment with two correlation measures: Pearson's ($r$) and Kendall's rank ($\tau$) correlation coefficients. In contrast to linear regression and Pearson's correlation coefficient, Kendall's tau is non-parametric and resistant to outliers (Kendall, 1948). It is therefore often used for measuring trends in time series and change point detection (Quessy et al., 2013). We rank the words based on the absolute values of $\tau$ and $r$.

## 5 Synthetic Evaluation

To systematically compare the different methodological choices we introduce a new synthetic evaluation framework. We create seven schemas for how a word's distributional statistics may change. Three of these model scenarios in which a semantic change occurs, but crucially the remaining four model scenarios that we would *not* wish to classify as semantic change. Our framework builds on previous approaches that have modelled one type of semantic change—either a word gaining an additional sense (Kulkarni et al., 2015) or a word's original sense being completely replaced (Rosenfeld and Erk, 2018). Furthermore, although most work has focused on *recall*, our framework can also test *precision*, i.e., the ability to distinguish the injected changes from noise.

### 5.1 Dataset Construction

We first randomly sample 10% of the tweets from a single month from the middle of our empirical dataset (Dec. 2014). We then draw 66 random 70% samples with replacement from this sample. These 66 samples represent a dataset of 66 months (5.5 years) in which no semantic changes occur, but words' distributional statistics still vary from month to month due to sampling noise. This differs from, e.g., Kulkarni et al. (2015), who used a series of exact duplicates of an initial set of documents. Finally, we inject controlled changes by inserting made-up 'pseudowords', carefully changing their frequencies and co-occurrence distributions throughout the time series.

Our procedure for inserting pseudowords is as follows: we split the real words that occur in our empirical data for December 2014 into five equally sized frequency bins. For each pseudoword $\rho$ that we insert, we choose a frequency bin. To represent one of the senses of $\rho$, we sample a real word $w$ from the relevant frequency bin. For each synthetic month $m$, we insert $\rho$ replacing each token of $w$ with success probability $p_{(\rho,w,m)}$.

For example, we might insert one pseudoword replacing the instances of the word 'pudding' with a fixed probability throughout the whole time series, and then insert this same pseudoword replacing the instances of the word 'neon' with increasing probability over time. This would model a word that initially has a meaning related to 'pudding', but which then acquires a new sense related to 'neon'. We use seven different schemas: three model different kinds of semantic change (C1–C3), and four model ephemeral changes that we aim to avoid (D1–D4): see Figure 1.

C1: **Description:** This schema models a word that gradually acquires a new sense over time while retaining its original sense (e.g., *snowflake*, *lit*). This corresponds to what Koch (2016, 24) calls 'innovative meaning change' and Tahmasebi et al. (2018, 35) calls 'novel word sense'.

**Procedure:** Sample one real word $w_1$ to represent the original pseudosense[7] of the pseudoword $\rho$ and another real word $w_2$ to rep-

[7] A single pseudosense may in practice correspond to mul-

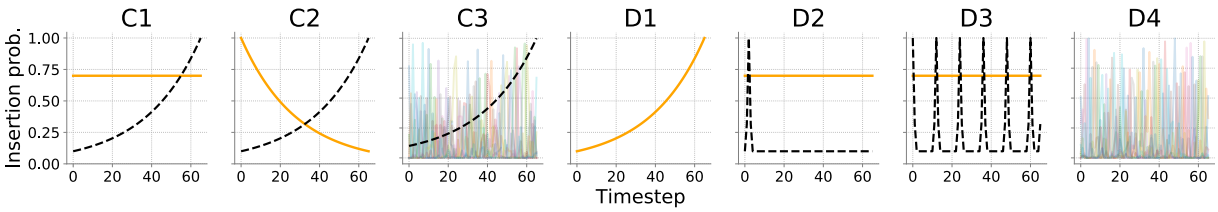


Figure 1: Illustration of our seven schemas for inserting pseudowords into the synthetic dataset. Each line represents a different pseudosense. Lines chart the probability of inserting a pseudoword token replacing a token representing the relevant pseudosense, as a function of 'time'. We vary whether the success probabilities change linearly or logarithmically and the time-steps at which the changes begin and end.

resent its new pseudosense. For each token of $w_1$ that occurs in the synthetic dataset for month $m$, insert a token of $\rho$ replacing it with success probability $p_{(\rho,w_1,m)}$, which remains constant throughout the time series. Insert $\rho$ replacing each token of $w_2$ with success probability $p_{(\rho,w_2,m)}$, which starts low and gradually increases over time.

C2: **Description:** This schema models a word that gradually acquires a new sense over time while its original sense gradually falls out of use (cf., *silly*, which originally meant 'happy' or 'lucky' and now means 'foolish'). This corresponds to the full cycle of genesis and disappearance of lexical polysemy as described by Koch (2016, 25), i.e., an 'innovative meaning change' and a 'reductive meaning change'.

**Procedure:** Sample one real word $w_1$ to represent the original sense of the pseudoword $\rho$, and another real word $w_2$ to represent its new sense. For each token of $w_1$ that occurs in the synthetic dataset for month $m$, insert a token of $\rho$ replacing it with success probability $p_{(\rho,w_1,m)}$, which starts relatively high and gradually decreases over time. Insert $\rho$ replacing each token of $w_2$ with success probability $p_{(\rho,w_2,m)} = 1 - p_{(\rho,w_1,m)}$.

C3: **Description:** This schema models a word with many senses, different random subsets of which are relatively frequent each month (e.g., an acronym that can refer to many different entities, which may trend at different times). Over time, the word acquires an additional, more stable sense whose frequency does not fluctuate so much from month to month. An example is *BLM*, which has been used to refer to a baseball magazine, a marketing company, a music label, the US Bureau of Land Management, etc., but since 2013 has been consistently associated with the Black Lives Matter movement. This could be considered a 'reductive' meaning change-in-progress, as we start out with multiple competing senses, and one sense gradually comes to dominate without the others having yet died out (see Koch, 2016, Fig. 2).

**Procedure:** Sample eight real words $\{w_1, w_2, ..., w_8\}$ to represent eight different pseudosenses for the pseudoword $\rho$. For each month $m$, draw a multinomial distribution $D^m$ over the first seven sampled words, using a Dirichlet prior with uniform, sparsity-inducing alpha. Replacing each token of a word $w_i, i \in [1, 7]$, insert a token of $\rho$ with success probability $D^m_{w_i}$. Let $w_8$ represent the new, more stable pseudosense, and for each month $m$, insert a token of $\rho$ replacing each token of $w_8$ with success probability $p_{(\rho,w_8,m)}$, which starts low and gradually increases over time.

D1: **Description:** This schema models a word that becomes more frequent over time, but *does not* change its co-occurrence distribution.

**Procedure:** Sample one real word $w$ to represent the meaning of the pseudoword $\rho$. For each token of $w$ that occurs in the synthetic dataset for month $m$, insert a token of $\rho$ replacing the token of $w$ with success probability $p_{(\rho,w,m)}$. $p_{(\rho,w,m)}$ starts relatively low and gradually increases over time.

D2: **Description:** This schema models a word with two senses. One sense is relatively infrequent, but suddenly spikes in frequency (e.g.,

tiple real senses, since the real word we use to represent this pseudosense may itself have multiple senses. There are few words in our dataset with only one sense according to WordNet; so, we restrict our choice to words for which WordNet lists no more than 10 senses. We also require that none of the real words chosen to represent different pseudosenses of a given pseudoword have any senses in common.

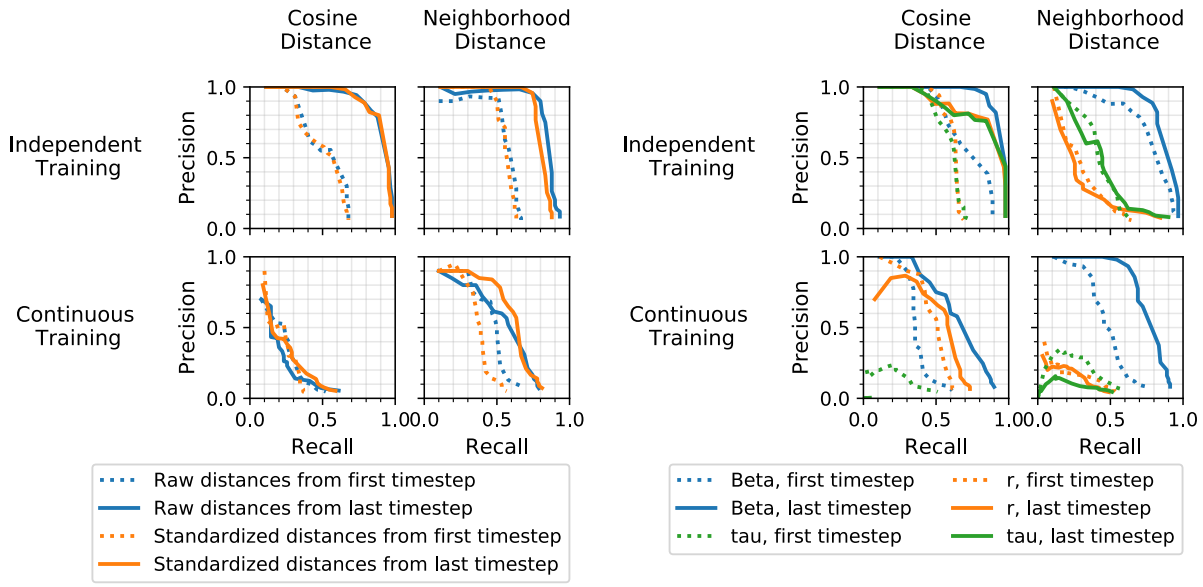


Figure 2: Precision–recall plots for times series approaches for $k$ in range $[0, 1000]$. Left: Change point methods with *Raw* and *Standardized* ($z$) scores. Right: global trend methods including linear regression (*Beta*), Pearson correlation coefficient (*r*), and Kendall rank correlation coefficient (*tau*). Dashed lines use the first timestep as the reference point for comparison while solid lines use the last timestep.

due to a trending topic), before becoming infrequent again.

**Procedure:** Sample two real words $w_1$ and $w_2$ to represent the two pseudosenses. Insert $\rho$ replacing each token of $w_1$ with probability $p_{(\rho,w_1,m)}$, which remains constant throughout the time series, and replacing each token of $w_2$ with probability $p_{(\rho,w_2,m)}$, which starts relatively low, rapidly increases and then rapidly decreases again.

D3: **Description:** This schema models a word with two senses: one is usually relatively infrequent, but spikes in frequency at periodic intervals (i.e., during the same month every year). An example is *turkey*, whose 'poultry' sense tends to be much more frequent around American Thanksgiving and Christmas.

**Procedure:** Sample two real words $w_1$ and $w_2$ to represent the two pseudosenses. Insert $\rho$ replacing each token of $w_1$ with probability $p_{(\rho,w_1,m)}$, which remains constant throughout the time series, and replace each token of $w_2$ with probability $p_{(\rho,w_2,m)}$, which is relatively low for most time-steps but rapidly spikes around the same month each year.

D4: **Description:** Like C3, this schema models words that can refer to many different entities, but in this case, an additional, more stable sense does not emerge.

**Procedure:** Sample seven real words $\{w_1, w_2, ..., w_7\}$ to represent different pseudosenses for $\rho$. For each month $m$, draw a multinomial distribution $D^m$ over these seven words, using a Dirichlet prior with uniform, sparsity-inducing alpha. Replacing each token of a word $w_i, i \in [1, 7]$, insert a token of $\rho$ with success probability $D^m_{w_i}$.

For each of these seven schemas, we create thirty pseudowords (six for each of our five frequency bins), and we vary whether the success probabilities change linearly or logarithmically and the time-steps at which the changes begin and end. In total, we insert 90 pseudowords using Schemas C1–C3, which model genuine semantic changes that we would like to be able to detect, and 120 pseudowords using Schemas D1–D4, which model real changes in words' use statistics but do *not* reflect semantic change. The synthetic dataset also contains 887,926 real words.

### 5.2 Evaluation Results

We evaluate systems by how highly they rank pseudowords from schemas C1–C3 using the Average Precision @ K, which approximates the area under a precision–recall curve over the interval from 0 to K. It is defined as the sum, over every rank $r$ in the top-K list of semantic change

candidates, of the precision at rank $r$ multiplied by the change in recall between ranks $r-1$ and $r$: $AP@K = \sum_{r=1}^{K} P(r)\Delta R(r)$, where $P(r)$ is the percentage of top-$r$ candidates which are pseudowords belonging to Schemas C1–C3, and $R(r)$ is the percentage of all C1–C3 pseudowords that appear in the top-$r$. The results are shown in Table 1 (two-step approach, comparing the first and last time steps) and Table 2 (whole time series). Precision–recall curves for the time series approaches are shown in Figure 2.

**Continuous training does not ensure that embeddings are comparable.** We experiment with three configurations for training the embeddings for the two-step approach: 1) training the embeddings for each time-step independently (ind.); 2) initializing the embeddings for the final time-step with those trained on the first time-step (cont.); and 3) continuous training throughout the whole series, so that the final time-step's embeddings are initialized with the data from all preceding time-steps (cont. whole series).

Continuous training has been used without separate alignment (e.g., Kim et al., 2014) as each time period is a continuation of the embeddings from the previous period. Table 1 shows, however, that alignment is necessary for continuously trained embeddings using the whole series as well as for independent ones when using the cosine distance measure. It is likely that the huge number of training updates in the entire time series causes the embeddings to drift considerably. For the time series approaches, we therefore did not apply the cosine measure without first aligning embeddings.

**Using the whole time series is more effective than comparing the first and the last time steps.** Overall, the approaches using the whole time series (Table 2) are more effective than the two-step approaches; particularly with regard to finding C1 pseudowords and avoiding D4 pseudowords.

**Continuous training provides no benefit for time series approaches.** For the time series approaches, independent training tends to perform better than continuous training (Table 2). The lack of improvement with continuous training is particularly noteworthy as independent training is more computationally efficient than continuous training, since different time periods can be trained in parallel. We did not explore the impact of different hyperparameter choices on continuous training, but note this would introduce another level of complexity in tuning model parameters.

| | ind. | cont. | cont. (whole series) |
|---|---|---|---|
| cosine (unaligned) | 0.00 | 0.32 | 0.00 |
| cosine (aligned) | 0.25 | 0.32 | 0.27 |
| neighbourhood | 0.28 | 0.34 | 0.30 |

Table 1: Average Precision @ 50 on the synthetic dataset of the two-step approach with CBOW

**Different time series approaches are best paired with different similarity measures.** Hamilton et al. (2016a) found that the neighbourhood-based measure tends to assign higher rates of semantic change to nouns, while the cosine measure tends to assign higher rates to verbs. However, they did not compare the overall effectiveness of these methods for semantic change detection.

We find that the neighbourhood-based measure[8] tends to outperform the cosine measure for the change point detection approaches; however, cosine tends to outperform the neighbourhood measure for correlation approaches (see Table 2). For change point detection, standardization of the time series does not have a consistent effect.

**The reference point for comparison matters.** For almost all configurations in Table 2, the AP @ 50 is better when the reference point is the last time-step. Figure 3 shows Recall@K broken down by pseudoword type. For types C1–C3, higher recall is better. Conversely, lower recall is better for types D1–D4, since these model changes that we do *not* consider to be lasting semantic changes. Recall is consistently low for types D1–D4, but strikingly, recall is also low for type C3 when we compare to the first-step. Schema C3 models words whose distributions change drastically from time-step to time-step, but which gradually become more stable as a new, consistently occurring sense emerges. The representation for the first time-step will thus be very different from subsequent representations, such that comparing to the first step is not effective. In contrast, comparing to the first time-step is expected to be more effective in finding words that become less stable over time.

**Correlation-based approaches perform worse than regression or change point detection approaches.** Pearson's correlation coefficient is

[8] We apply this to unaligned embeddings; alignment with orthogonal Procrustes has no effect on this measure.

| Measure | Training | Comparing to the first time-step | | | | | Comparing to the last time-step | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | *raw* | $z$ | $\beta$ | $r$ | $\tau$ | *raw* | $z$ | $\beta$ | $r$ | $\tau$ |
| cosine | independent | 0.37 | 0.36 | 0.51 | 0.52 | 0.47 | 0.54 | 0.56 | 0.56 | 0.48 | 0.49 |
| cosine | continuous | 0.17 | 0.18 | 0.34 | 0.40 | 0.02 | 0.13 | 0.15 | 0.45 | 0.34 | 0.00 |
| neighbourhood | independent | 0.47 | 0.50 | 0.48 | 0.23 | 0.35 | 0.53 | 0.56 | 0.56 | 0.20 | 0.30 |
| neighbourhood | continuous | 0.35 | 0.32 | 0.37 | 0.05 | 0.05 | 0.34 | 0.42 | 0.54 | 0.05 | 0.00 |

Table 2: Average Precision @ 50 on the synthetic dataset using time series approaches with CBOW. Change point methods are raw scores (*raw*) and standardized scores ($z$). Global trend methods are linear regression ($\beta$), Pearson correlation coefficient ($r$), and Kendall rank correlation coefficient ($\tau$).

maximized when the *magnitude* of the change between consecutive time periods is consistent over all time periods whereas maximizing Kendall's $\tau$ simply requires the change between consecutive time periods to be of a consistent *sign*. Both correlation measures therefore have particularly poor recall of words that have time periods without a consistent meaning as in the early time periods for pseudowords of type C3.

The $\beta$ value of the linear regression assumes a linear relationship, but is unfortunately sensitive to outliers (Chatterjee and Hadi, 1986), which likely explains why the regression approach has higher recall than change point approaches for schema D4 (Figure 3), in which a stable sense does not emerge. In general, however, the $\beta$ values produced for D4 pseudowords appear to be smaller in magnitude than genuine semantic changes (C1–C3) resulting in average precision measures that generally match or exceed change point approaches (Table 2). Regression is also more straightforward and computationally efficient to calculate than change point measures.

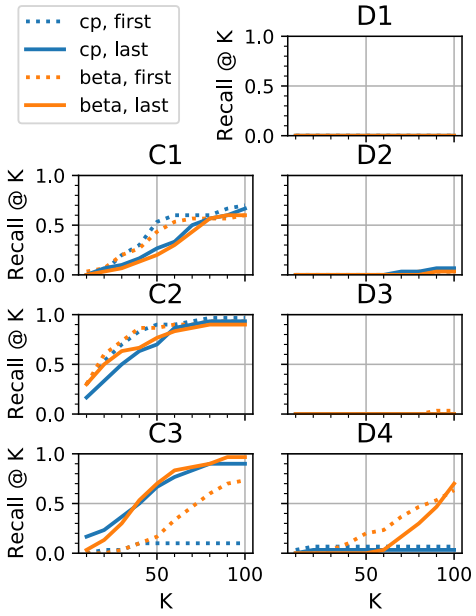


Figure 3: Recall@K of each schema, using independently trained embeddings and the neighbourhood-based measure. 'first'/ 'last' denotes the reference time-step, 'cp' the unstandardized change-point approach, and 'beta' the linear regression approach. C1–C3: higher recall is better, D1–D4: lower is better.

## 6 Results on Empirical Twitter Data

We now apply the approaches on our full empirical Twitter dataset. Table 3 shows the top 10 semantic change candidates using independent, aligned CBOW embeddings. When using continuously trained embeddings, the top-10 lists are similar.

In line with our synthetic results, we find different candidates when comparing to the first time-step or the last, but they appear to represent similar kinds of semantic change. Most have shifted due to associations with named entities. For example, *vine* (Figure 4a) acquired a new sense in January 2013 when the popular short-form video hosting service Vine was launched. Similarly, *ig*, initially shorthand for 'i guess', became shorthand for the social network Instagram as it gained popularity. The embedding for *shawn* shifted significantly around the beginning of 2014, when the singer Shawn Mendes signed a record deal.

There are also words whose embeddings have shifted due to waning associations with prominent named entities, e.g., *vow* was initially associated with The Vow, a high grossing movie released in Feb. 2012, but by the end of the time series it had shifted back towards synonyms like 'pledge' and 'urge'. Likewise the embedding for *temple* initially reflected the popularity of the video game Temple Run but gradually shifted to the word's canonical meaning, and the embedding for *bcs* initially reflected its usage as an acronym for Bowl Championship Series (a selection system in American college football), but then shifted towards

| | |
|---|---|
| **Comparing to first time-step** | vine, temple, unfollowers, favorited, mcm, glo, #ipadgames, shawn, retweeted, vow |
| **Comparing to last time-step** | isis, yasss, bcs, temple, 🐶, mcm, 😐, ig, mila, glo |

Table 3: Top 10 semantic change candidates of the change-point detection approach without standardization, using independently trained and aligned CBOW embeddings and the cosine distance measure.

'bcoz', 'bec', and other forms of 'because' after the selection system was ended in 2013.

There are also examples of neologisms: *mcm* is a lexicalized acronym for 'Man Crush Monday'. This initially referred to the meme of posting about a man one finds attractive each Monday, but then by metonymic extension came to be used to refer to the subject of the post himself. Another example is *glo*, which in the beginning of our data (Figure 4b) occurs mainly in reference to a Nigerian telecommunication company. A shift in its embedding is driven by the sudden emergence of the expression 'glo up', which was coined in August 2013 by rapper Chief Keef in the song "Gotta Glo Up One Day", and later gained traction as an expression to describe an impressive personal transformation.

Finally, there are words whose detected change-points reflect changes in automated activity. For example, the embedding for *yasss* shifts in early 2017 due to a sudden proliferation of tweets automatically posted to users' Twitter accounts by the live video streaming app LiveMe, which all begin with the text 'YASSS It's time for a great show' followed by the title and link to the video stream. Conversely, the detected change for *favorited* (Figure 4c) coincides with a sudden disappearance of automatically generated tweets about favorited YouTube videos.

## 7 Conclusion

In this paper, we presented a new evaluation framework and systematically compared the various choices involved in using word embeddings for semantic change detection. We then applied the approaches to a Twitter dataset spanning 5.5 years. Qualitative analysis found that the top ranked words have undergone genuine semantic change although some of the changes are restricted to social media or to Twitter specifically.

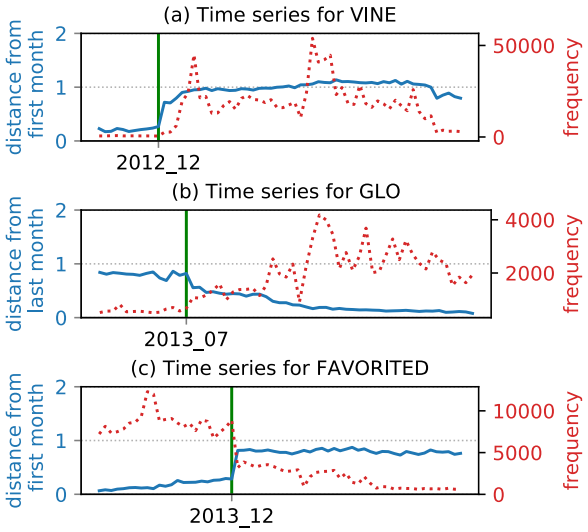


Figure 4: Neighbourhood-based distance (solid blue lines) and frequency (dotted red lines) over time, for three semantic change candidates. Vertical green lines indicate the automatically estimated change-points.

Our framework and dataset can also be used to evaluate approaches not considered in this paper. Moreover, our framework models different semantic change scenarios, and future work could focus on approaches that are able to distinguish between these different scenarios.

## Acknowledgements

This work was supported by The Alan Turing Institute under the UK Engineering and Physical Sciences Research Council (EPSRC) grant EP/N510129/1. P.S. was supported in part by the EPSRC Centre for Doctoral Training in Data Science, funded by the UK EPSRC (grant EP/L016427/1) and the University of Edinburgh. D.N. was supported by Turing award TU/A/000006 and B.McG. by Turing award TU/A/000010 (RG88751). S.A.H. was supported in part by The Volkswagen Foundation.

## References

Robert Bamler and Stephan Mandt. 2017. Dynamic word embeddings. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 380–389.

Pierpaolo Basile and Barbara McGillivray. 2018. *Discovery Science*, volume 11198 of *Lecture Notes in Computer Science*, chapter Exploiting the Web for Semantic Change Detection. Springer-Verlag.

D. M. Blei and J. D. Lafferty. 2006. Dynamic topic models. In *Proceedings of the 23rd international conference on Machine learning*, pages 113–120.

Samprit Chatterjee and Ali S. Hadi. 1986. Influential observations, high leverage points, and outliers in linear regression. *Statistical Science*, 1(3):379–393.

Paul Cook, Jey Han Lau, Diana McCarthy, and Timothy Baldwin. 2014. Novel word-sense identification. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1624–1635.

Haim Dubossarsky, Daphna Weinshall, and Eitan Grossman. 2017. Outta control: Laws of semantic change and inherent biases in word representation models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1136–1145.

Lea Frermann and Mirella Lapata. 2016. A bayesian model of diachronic meaning change. *Transactions of the Association for Computational Linguistics*, 4:31–45.

Mark Graham, Scott A. Hale, and Devin Gaffney. 2014. Where in the world are you? geolocation and language identification in Twitter. *The Professional Geographer*, 66(4):568–578.

William L Hamilton, Jure Leskovec, and Dan Jurafsky. 2016a. Cultural shift or linguistic drift? comparing two computational measures of semantic change. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing*, volume 2016, pages 2116–2121.

William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2016b. Diachronic word embeddings reveal statistical laws of semantic change. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1489–1501.

Maurice G. Kendall. 1948. *Rank correlation methods*. Griffin, London.

Tom Kenter, Melvin Wevers, Pim Huijnen, and Maarten de Rijke. 2015. Ad hoc monitoring of vocabulary shifts over time. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 1191–1200, New York, NY, USA. ACM.

Yoon Kim, Yi-I Chiu, Kentaro Hanaki, Darshan Hegde, and Slav Petrov. 2014. Temporal analysis of language through neural language models. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*, pages 61–65.

Peter Koch. 2016. Meaning change and semantic shifts. In Päivi Juvonen and Maria Koptjevskaja-Tamm, editors, *The Lexical Typology of Semantic Shifts*, pages 21–66. De Gruyter Mouton, Berlin/Boston.

Vivek Kulkarni, Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2015. Statistically significant detection of linguistic change. In *Proceedings of the 24th International Conference on World Wide Web*, pages 625–635.

Andrey Kutuzov, Lilja Øvrelid, Terrence Szymanski, and Erik Velldal. 2018. Diachronic word embeddings and semantic shifts: a survey. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1384–1397.

Andrey Kutuzov, Erik Velldal, and Lilja Øvrelid. 2017. Temporal dynamics of semantic relations in word embeddings: an application to predicting armed conflict participants. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1824–1829.

B. McGillivray, S. Hengchen, V. Läteenoja, M. Palma, and A. Vatri. 2019. A computational approach to lexical polysemy in Ancient Greek. *Digital Scholarship in the Humanities*.

Rada Mihalcea and Vivi Nastase. 2012. Word epoch disambiguation: Finding how words change over time. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 259–263.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations (ICLR) Workshop*.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Dong Nguyen and Jacob Eisenstein. 2017. A kernel independence test for geographical language variation. *Computational Linguistics*, 43(3):567–592.

Jean-François Quessy, Mériem Saïd, and Anne-Catherine Favre. 2013. Multivariate kendall's tau for change-point detection in copulas. *Canadian Journal of Statistics*, 41(1):65–82.

Radim Řehůřek and Petr Sojka. 2010. Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta.

Martina A. Rodda, Marco S.G. Senaldi, and Alessandro Lenci. 2017. Panta Rei: Tracking Semantic Change with Distributional Semantics in Ancient Greek. *Italian Journal of Computational Linguistics*, 3:11–24.

Christian Rohrdantz, Annette Hautli, Thomas Mayer, Miriam Butt, Daniel A. Keim, and Frans Plank. 2011. Towards tracking semantic change by visual analytics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 305–310.

Alex Rosenfeld and Katrin Erk. 2018. Deep neural models of semantic shift. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 474–484.

Maja R. Rudolph and David M. Blei. 2018. Dynamic embeddings for language evolution. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, pages 1003–1011.

Ian Stewart, Dustin Arendt, Eric Bell, and Svitlana Volkova. 2017. Measuring, predicting and visualizing short-term change in word representation and usage in vkontakte social network. In *Proceedings of the Eleventh International AAAI Conference on Web and Social Media (ICWSM 2017)*, pages 672–675.

Nina Tahmasebi, Lars Borin, and Adam Jatowt. 2018. Survey of Computational Approaches to Diachronic Conceptual Change.

Xuri Tang. 2018. A state-of-the-art of semantic change computation. *Natural Language Engineering*, 24(5):649–676.

Xuri Tang, Weiguang Qu, and Xiaohe Chen. 2016. Semantic change computation: A successive approach. *World Wide Web - Internet & Web Information Systems*, 19(3):375–415.

Marco Del Tredici, Raquel Fernández, and Gemma Boleda. 2018. Short-term meaning shift: an exploratory distributional analysis. *arXiv preprint arXiv:1809.03169*.

Derry Tanti Wijaya and Reyyan Yeniterzi. 2011. Understanding semantic change of words over centuries. In *Proceedings of the 2011 international workshop on DETecting and Exploiting Cultural diversiTy on the social web*, pages 35–40.

Zijun Yao, Yifan Sun, Weicong Ding, Nikhil Rao, and Hui Xiong. 2018. Dynamic word embeddings for evolving semantic discovery. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, WSDM '18, pages 673–681.

# Correlations between Word Vector Sets

**Vitalii Zhelezniak, April Shen, Daniel Busbridge,**
**Aleksandar Savkov, Nils Hammerla**
{firstname.lastname}@babylonhealth.com

## Abstract

Similarity measures based purely on word embeddings are comfortably competing with much more sophisticated deep learning and expert-engineered systems on unsupervised semantic textual similarity (STS) tasks. In contrast to commonly used geometric approaches, we treat a single word embedding as e.g. 300 observations from a scalar random variable. Using this paradigm, we first illustrate that similarities derived from elementary pooling operations and classic correlation coefficients yield excellent results on standard STS benchmarks, outperforming many recently proposed methods while being much faster and trivial to implement. Next, we demonstrate how to avoid pooling operations altogether and compare sets of word embeddings directly via correlation operators between reproducing kernel Hilbert spaces. Just like cosine similarity is used to compare individual word vectors, we introduce a novel application of the centered kernel alignment (CKA) as a natural generalisation of squared cosine similarity for sets of word vectors. Likewise, CKA is very easy to implement and enjoys very strong empirical results.

## 1 Introduction

Distributed representations of text have had a massive impact on the natural language processing (NLP), information retrieval (IR), and machine learning (ML) communities, thanks in part to their ability to capture rich notions of semantic similarity. While this work originally began with word embeddings (Bengio et al., 2003; Mikolov et al., 2013a; Pennington et al., 2014; Bojanowski et al., 2017; Joulin et al., 2017), there is now an ever-increasing number of representations for longer units of text based on simple aggregations of word vectors (Mitchell and Lapata, 2008; De Boom et al., 2016; Arora et al., 2017; Wieting et al., 2016; Wieting and Gimpel, 2018; Zhelezniak et al., 2019b) as well as complex neural architectures (Le and Mikolov, 2014; Kiros et al., 2015; Hill et al., 2016; Conneau et al., 2017; Gan et al., 2017; Tang et al., 2017; Pagliardini et al., 2018; Zhelezniak et al., 2018; Subramanian et al., 2018; Cer et al., 2018; Devlin et al., 2018).

By contrast, relatively little effort has been directed towards understanding the similarity measures used to compare these textual embeddings, for which cosine similarity remains a convenient and widespread, yet somewhat arbitrary default, despite some emerging research into the alternatives (Camacho-Collados et al., 2015; De Boom et al., 2015; Santus et al., 2018; Zhelezniak et al., 2019b,a). Part of the appeal of cosine similarity perhaps lies in the simple geometric interpretation behind it. However, as embeddings are ultimately just arrays of numbers, we are free to take alternative viewpoints other than the geometric ones, if they lead to illuminating insights or strong-performing methods.

Following Zhelezniak et al. (2019a), we treat a word embedding not as a geometric vector but as a statistical sample (of e.g. 300 observations) from a scalar random variable, and indeed find insights that are both intriguing and noteworthy. We first illustrate that similarities derived from elementary pooling operations and classic univariate correlation coefficients yield excellent results on standard semantic textual similarity (STS) benchmarks, outperforming many recently proposed methods while being much faster and simpler to implement. This empirically validates the advantages of the statistical perspective on word embeddings over the geometric interpretations. In the process, we provide more evidence that departures from normality, and in particular the presence of outliers, can have severe negative effects on the performance of some correlation coeffi-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 77–87,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

cients. We show how to overcome these complications, by selecting an outlier-removing pooling operation such as max-pooling, applying a more robust correlation coefficient such as Spearman's $\rho$, or simply clipping (winsorizing) the word vectors.

Next, we demonstrate how to avoid pooling operations completely and compare sets of word embeddings directly via correlation operators between reproducing kernel Hilbert spaces (RKHS). We introduce a novel application of the kernel alignment (KA) and the centered kernel alignment (CKA) as a natural generalisation of the squared cosine similarity and Pearson correlation for the sets of word embeddings. These multivariate correlation coefficients are very easy to implement and also enjoy very strong empirical results.

## 2 Related Work

Several lines of research seek to combine the strength of pretrained word embeddings and the elegance of set- or bag-of-words (BoW) representations. Any method that determines semantic similarity between sentences by comparing the corresponding sets of word embeddings is directly related to our work.

Perhaps the most obvious such approaches are based on elementary pooling operations such as average-, max- and min-pooling (Mitchell and Lapata, 2008; De Boom et al., 2015, 2016). While seemingly over-simplistic, numerous studies have confirmed their impressive performance on the downstream tasks (Arora et al., 2017; Wieting et al., 2016; Wieting and Gimpel, 2018; Zhelezniak et al., 2019b)

One step further, Zhao and Mao (2017); Zhelezniak et al. (2019b) introduce fuzzy bags-of-words (FBoW) where degrees of membership in a fuzzy set are given by the similarities between word embeddings. Zhelezniak et al. (2019b) show a close connection between FBoW and max-pooled word vectors.

Some approaches do not seek to build an explicit representation and instead focus directly on designing a similarity function between sets. Word Mover's Distance (WMD) (Kusner et al., 2015) is an instance of the Earth Mover's Distance (EMD) computed between normalised BoW, with the cost matrix given by Euclidean distances between word embeddings. In the soft cardinality framework of (Jimenez et al., 2010, 2015), the contribution of a word to the cardinality of a set depends on its similarities to other words in the same set. Such sets are then compared using an appropriately defined Jaccard index or related measures. DynaMax (Zhelezniak et al., 2019b) uses universe-constrained fuzzy sets designed explicitly for similarity computations.

Approaches that see word embeddings as statistical objects are very closely related to our work. Virtually all of them treat word embeddings as observations from some $D$-variate parametric family, where $D$ is the embedding dimension. Arora et al. (2016, 2017) introduce a latent discourse model and show the maximum likelihood estimate (MLE) for the discourse vector to be the weighted average of word embeddings in a sentence, where the weights are given by smooth inverse frequencies (SIF). Nikolentzos et al. (2017); Torki (2018) treat sets of word embeddings as observations from $D$-variate Gaussians, and compare such sets with cosine similarity between the parameters (means and covariances) estimated by maximum likelihood. Vargas et al. (2019) measure semantic similarity through penalised likelihood ratio between the joint and factorised models and explore Gaussian and von Mises–Fisher likelihoods.

Cosine similarity between covariances is an instance of the RV coefficient and its uncentered version was applied in the context of word embeddings before (Botev et al., 2017). We arrive at a similar coefficient (but with different centering) as a special case of CKA, which in the general case makes no parametric assumptions about disbtributions whatsoever. In particular our version is suitable for comparing sets containing just one word vector, whereas the method of Nikolentzos et al. (2017); Torki (2018) requires at least two vectors in each set. Very recently, Kornblith et al. (2019) used CKA to compare representations between layers of the same or different neural networks. This is again an instance of treating such representations as observations from a $D$-variate distribution, where $D$ is the dimension of the hidden layer in question. Our use of CKA is completely different from theirs.

Unlike all of the above approaches, (Zhelezniak et al., 2019a) see each word embedding itself as $D$ (e.g. 300) observations from some scalar random variable. They cast semantic similarity as correlations between these random variables and study

their properties using simple tools from univariate statistics. While they consider correlations between individual word vectors and averaged word vectors, they do not formally explore correlations between word vector sets. We review their framework in Section 3 and then proceed to formalise and generalise it to the case of sets of word embeddings.

## 3 Background: Correlation Coefficients and Semantic Similarity

Suppose we have a word embeddings matrix $\mathbf{W} \in \mathbb{R}^{N \times D}$, where $N$ is the number of words in the vocabulary and $D$ is the embedding dimension (usually 300). In other words, each row $\mathbf{w}^{(i)}$ of $\mathbf{W}$ is a $D$-dimensional word vector. When applying statistical analysis to these vectors, one might choose to treat each $\mathbf{w}^{(i)}$ as an observation from some $D$-variate distribution $P_D(E_1, \dots E_D)$ and model it with a Gaussian or a Gaussian Mixture. While such analysis helps in studying the overall geometry of the embedding space (how dimensions correlate and how embeddings cluster), $P_D$ is not directly useful for semantic similarity between individual words.

For the latter, Zhelezniak et al. (2019a) proposed to look at the transpose $\mathbf{W}^T$ and the corresponding distribution $P(W_1, W_2, \dots, W_N)$. Under this perspective, each word vector $\mathbf{w}_{(i)}$ is now a sample of $D$ (e.g. 300) observations from a scalar random variable $W_i$. Luckily, in applications we are usually not interested in the full joint distribution but only in the similarity between two words, i.e. the bivariate marginal $P(W_i, W_j)$. In practice, we make inferences about this marginal from the paired sample $(\mathbf{w}_{(i)}, \mathbf{w}_{(j)})$ through visualisations (histograms, Q-Q plots, scatter plots, etc.) as well as various statistics.

Zhelezniak et al. (2019a) found that for all common models (GloVe, fastText, word2vec) the means across word embeddings are tightly concentrated around zero (relative to their dimensions), thus making the widely used cosine similarity practically equivalent to Pearson correlation. However, while word2vec vectors seem mostly normal, GloVe and fastText vectors are highly non-normal, likely due to the presence of heavy univariate and bivariate outliers (as suggested by visualisations mentioned earlier). Quantitatively, the majority of GloVe and fastText vectors fail the Shapiro-Wilk normality test at significance level 0.05. Therefore, while Pearson's $r$ (and thus cosine similarity) may be acceptable for word2vec, it is preferable to resort to more robust non-parametic correlation coefficients such as Spearman's $\rho$ or Kendall's $\tau$ as a similarity measure between GloVe and fastText vectors.

Finally, very similar conclusions were shown to hold for sentence representations obtained by word vector averaging, also referred to as mean-pooling. In particular, averaged fastText vectors compared with rank correlation coefficients already show impressive results on standard STS tasks, rivaling much more sophisticated systems.

## 4 Correlations between Word Vector Sets

We are interested in applying the statistical framework from Section 3 to measure the semantic similarity between two sentences $s_1$ and $s_2$ given by the sets (or bags) $S_1$ and $S_2$ of word embeddings respectively. To formalise this new setup, we may see each set of word embeddings $S = \{\mathbf{w}_{(1)}, \mathbf{w}_{(2)}, \dots, \mathbf{w}_{(k)}\}$ as a sample (of e.g. 300 observations) from some theoretical set of scalar random variables $R = \{W_1, W_2, \dots, W_k\}$. In light of the above, our task then lies in finding correlation coefficients $\text{corr}(R_1, R_2)$ between $R_1$ and $R_2$ and their empirical estimates $\widehat{\text{corr}}(S_1, S_2)$ obtained from the paired sample $S_1, S_2$, hoping that such coefficients will serve as a good proxy for semantic similarity. Recall that for single-word sets $R_1 = \{W_i\}, R_2 = \{W_j\}$ the task simply reduces to computing a univariate correlation between word vectors $\mathbf{w}_{(i)}$ and $\mathbf{w}_{(j)}$, where the choice of the coefficient (Pearson's $r$, Spearman's $\rho$, etc.) is made based on the statistics exhibited by the word embeddings matrix. While generalising this to sets of more than one variable is not particularly hard, there are several ways to do so, each with its own advantages and downsides. In the present work, we group these approaches into two broad families: pooling-based and pooling-free correlation coefficients.

### 4.1 Correlations between Pooled Variables

Pooling-based approaches first reduce a set of random variables to a single scalar random variable $W_{\text{pool}} = f_{\text{pool}}(W_1, W_2, \dots, W_k)$ and then apply univariate correlation coefficients between the pooled variables. In practice this would correspond to pooling word embeddings $\mathbf{w}_{(1)}, \mathbf{w}_{(2)}, \dots, \mathbf{w}_{(k)}$ (along $i = 1{:}k$) into one

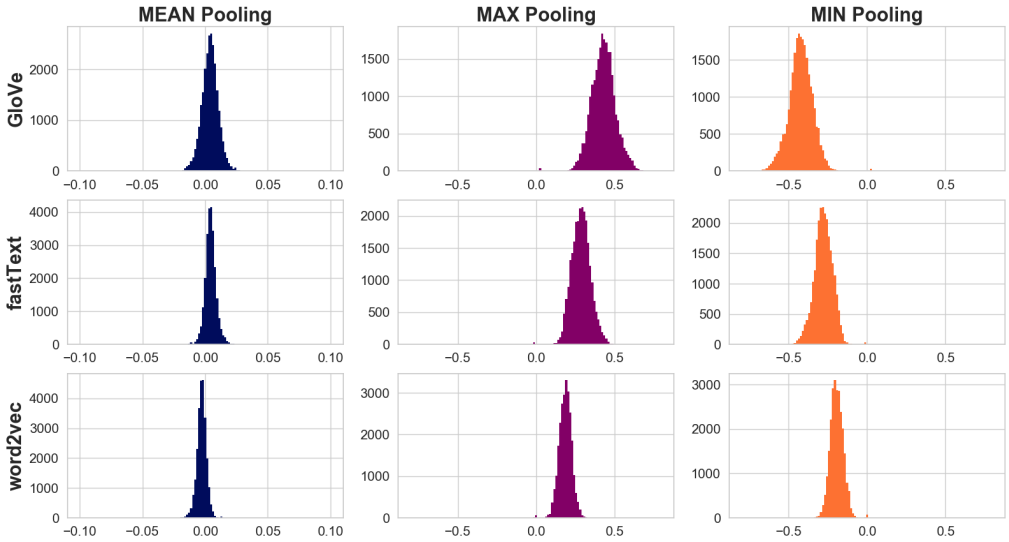


Figure 1: Normalised histograms of the mean distribution for sentence vectors generated by mean-, max- and min-pooling. Sentences were taken from the entire STS dataset (Agirre et al., 2012, 2013, 2014, 2015, 2016; Cer et al., 2017), and we utilise three commonly-used word embedding models: GloVe (Pennington et al., 2014), fastText (Bojanowski et al., 2017), and word2vec (Mikolov et al., 2013b,c).

**Algorithm 1** MaxPool-Spearman

**Require:** Word embeddings for the first sentence $\mathbf{x}^{(1)}, \mathbf{x}^{(2)} \ldots, \mathbf{x}^{(k)} \in \mathbb{R}^{1\times d}$

**Require:** Word embeddings for the second sentence $\mathbf{y}^{(1)}, \mathbf{y}^{(2)} \ldots, \mathbf{y}^{(l)} \in \mathbb{R}^{1\times d}$

**Ensure:** Similarity score $MS$

*# Max-pooling performed element-wise*

$\mathbf{x} \leftarrow \text{MAX\_POOL}(\mathbf{x}^{(1)}, \mathbf{x}^{(2)} \ldots, \mathbf{x}^{(k)})$

$\mathbf{y} \leftarrow \text{MAX\_POOL}(\mathbf{y}^{(1)}, \mathbf{y}^{(2)} \ldots, \mathbf{y}^{(l)})$

$\text{MS} \leftarrow \text{SPEARMANCORRELATION}(\mathbf{x}, \mathbf{y})$

fixed vector $\mathbf{w}_{\text{pool}}$, followed by computing univariate sample correlations. Certainly, these approaches are empirically attractive: not only are they very simple computationally (e.g. see Algorithm 1) but they also keep us in the realm of univariate statisics, where we have an entire arsenal of effective tools for making inferences about $W_{\text{pool}}$.

Unfortunately, it is not always clear *a priori* what should dictate our choice of the pooling function (though, as we will see shortly, for certain functions some statistical justifications do exist). By far the most common pooling operations for word embedding found in the literature are mean-, max- and min-pooling. It is also very common, with some exceptions, to treat these various pooled representation in a completely identical fashion, e.g. by comparing them all with cosine similarity. Intuitively, however, we suggest that the statistics of $W_{\text{pool}}$ must heavily depend on the pooling function $f_{\text{pool}}$ and thus each such pooled random variable should be studied in its own right. To illustrate this point, we would like to reveal the very different nature of mean- and max- and min-pooled sentence vectors though a practical example.

### 4.2 Statistics of the Pooled Representations: A Practical Analysis

Let us begin by examining sentence vectors obtained through mean-pooling. Recall that for common word embedding models, the mean across 300 dimensions of a single word embedding $\mathbf{w}_{(i)}$ happens to be close to zero (relative to the dimensions). By the linearity of expectation, we have that $\mathbb{E}[W_{\text{mean}}] = \mathbb{E}\left[\sum_{i=1}^{k} W_i\right] = \sum_{i=1}^{k} \mathbb{E}[W_i]$, and so the mean across $\mathbf{w}_{\text{mean}}$ will also be close to zero at least for small $k$. In practice, this seems to hold even for moderate $k$ in naturally occurring sentences, as seen in Figure 1. Based on this, we expect Pearson correlation and cosine similarity to have almost identical performance on the downstream tasks, which is confirmed in Figure 2.

On the other hand, intuition tells us that the means of the max-pooled vectors will be shifted

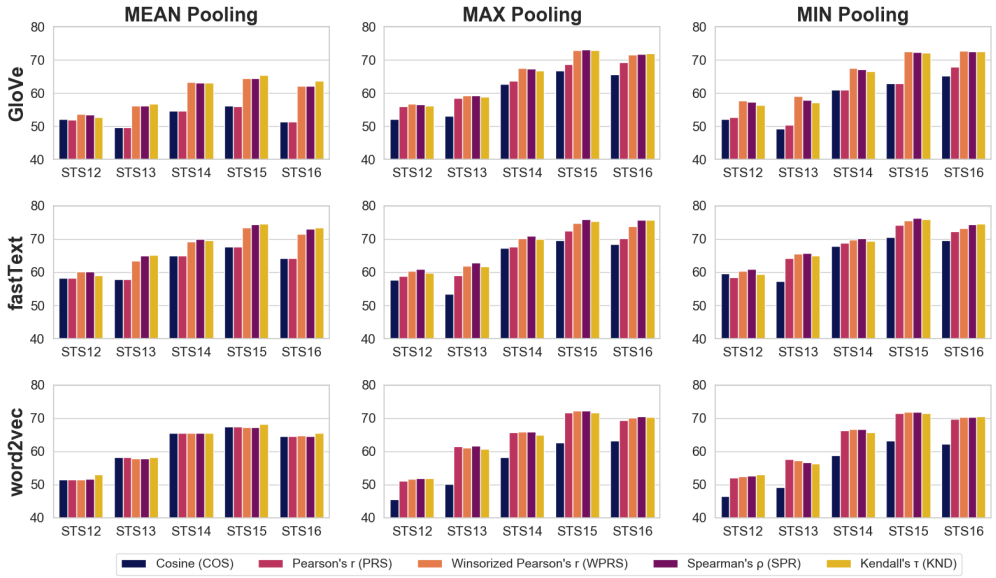


Figure 2: Bar plots of Pearson correlation on STS tasks between human scores and the following set-based similarity metrics: Cosine similarity (COS), Pearson's $r$ (PRS), Winsorized Pearson's $r$ (WPRS), Spearman's $\rho$ (SPR), and Kendall's $\tau$ (KND). Plots generated for three pooling methods and the following word embedding models: GloVe (Pennington et al., 2014), fastText (Bojanowski et al., 2017), and word2vec (Mikolov et al., 2013b,c).

to the right because of the max operation, which we see in Figure 1. In this case, cosine similarity and Pearson correlation will yield different results and, in fact, Pearson's $r$ considerably outperforms cosine on the downstream tasks (Figure 2). This in turn empirically adds weight to the statistical interpretation (correlation) over its geometrical counterpart (angle between vectors).

Recall also that unlike word2vec, GloVe and fastText vectors feature heavy univariate outliers, and the same can be expected to hold for the pooled representations; an example is shown in Figure 3. In case of mean-pooled vectors, this particular departure from normality can be successfully detected by the Shapiro-Wilk normality test, informing the appropriate choice of the correlation coefficient (Pearson's $r$ or robust rank correlation). By contrast, such procedure cannot be readily applied to max-pooled and min-pooled vectors as by construction they exhibit additional departures from normality, such as positive and negative skew respectively. It is always a good idea to consult visualisations for such vectors, such as the ones in Figure 3. Interestingly though, we do observe the some noteworthy regularities, which we describe further in Section 5.

The above example is meant to illustrate that even the simplest pooled random variables show strikingly different statistics depending on the aggregation. While the abundance of various pooling operations may be intimidating, the resulting vectors are always subject to the many tools of univariate statistics. As we hope to have shown, even crude analysis can shed light on the nature of these textual representations, which in turn has notable practical implications, as we will see in Section 5.

### 4.3 Correlations between Random Vectors

Exactly as before, suppose we have two sentences $S_1 = \{\mathbf{x}_{(1)}, \mathbf{x}_{(2)}, \ldots, \mathbf{x}_{(k)}\}$ and $S_2 = \{\mathbf{y}_{(1)}, \mathbf{y}_{(2)}, \ldots, \mathbf{y}_{(l)}\}$ and the corresponding random vectors $\mathbf{X} = (X_1, X_2, \ldots, X_k)$ and $\mathbf{Y} = (Y_1, Y_2, \ldots, Y_l)$. At this point it is important to emphasise again that we relate each word vector $\mathbf{x}_i$ to a random variable $X_i$ and treat the dimensions of $\mathbf{x}_i$ as $D$ observations from that variable, and similarly for $\mathbf{y}_i$ and $Y_i$. In contrast with the pooling-based approaches, our task here is to find a suitable correlation coefficient directly between the random vectors $\mathbf{X}$ and $\mathbf{Y}$. We begin by recalling the expression for the basic univariate Pearson's $r$:

$$r_{XY} = \frac{\mathbb{E}_{X,Y}\left[(X - \mu_X)(Y - \mu_Y)\right]}{\sigma_X \sigma_Y}, \quad (1)$$

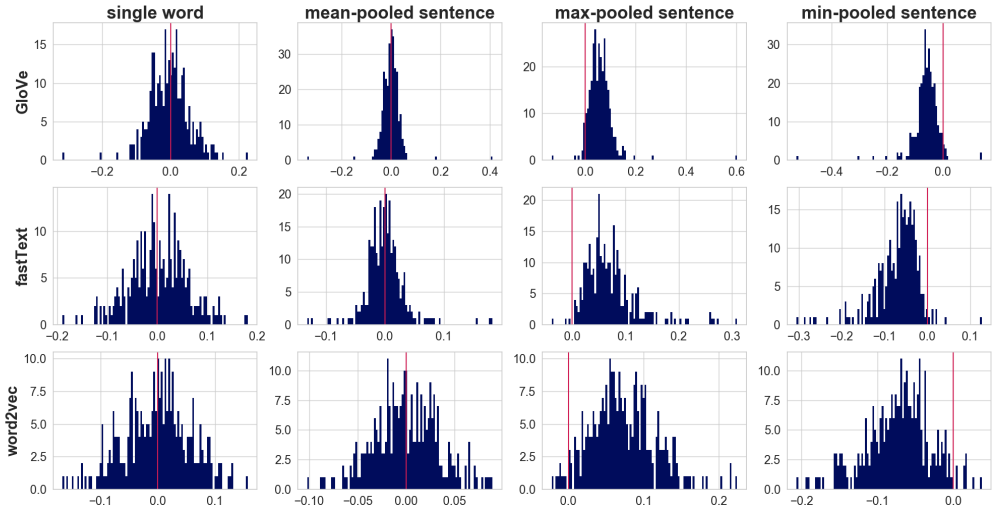


Figure 3: Histograms for word embeddings of the word "cats" and pooled representations of the embeddings for the words in the sentence "I like cats because they are very cute animals".

where

$$\mu_X = \mathbb{E}[X], \quad \sigma_X = \sqrt{\mathbb{E}\left[X^2\right] - \mu_X^2},$$

and similarly for $\mu_Y$ and $\sigma_Y$. The covariance term $\mathrm{cov}(X, Y)$ in the numerator is readily generalised to random vectors by the following cross-covariance operator between reproducing kernel Hilbert spaces (RKHS) $\mathcal{F}$ and $\mathcal{G}$

$$\mathcal{C}_{\mathbf{XY}} = \mathbb{E}_{\mathbf{X},\mathbf{Y}}\left[(\phi(\mathbf{X}) - \mu_{\mathbf{X}}) \otimes (\psi(\mathbf{Y}) - \mu_{\mathbf{Y}})\right], \tag{2}$$

where $\otimes$ denotes the tensor product and $\mu_{\mathbf{X}} = \mathbb{E}_{\mathbf{X}}\left[\phi(\mathbf{X})\right]$, $\mu_{\mathbf{Y}} = \mathbb{E}_{\mathbf{Y}}\left[\psi(\mathbf{Y})\right]$. Here $\phi$ and $\psi$ are the feature maps such that $\langle \phi(\mathbf{x}), \phi(\mathbf{x}')\rangle_{\mathcal{F}} = K(\mathbf{x}, \mathbf{x}')$ and $\langle \psi(\mathbf{y}), \psi(\mathbf{y}')\rangle_{\mathcal{G}} = L(\mathbf{y}, \mathbf{y}')$, where $K$ and $L$ are the kernels associated with RKHS $\mathcal{F}$ and $\mathcal{G}$ respectively. Note that if $\phi$ and $\psi$ are the identity maps, the cross-covariance operator (2) simply becomes the cross-covariance matrix

$$\mathbf{C}_{\mathbf{XY}} = \mathbb{E}_{\mathbf{X},\mathbf{Y}}\left[(\mathbf{X} - \mu_{\mathbf{X}})\left(\mathbf{Y} - \mu_{\mathbf{Y}}\right)^T\right].$$

Gretton et al. (2005a) define the Hilbert-Schmidt independence criterion (HSIC) to be the squared Hilbert-Schmidt norm $||\mathcal{C}_{\mathbf{XY}}||^2_{\mathrm{HS}}$ of (2) and derive an expression for it in terms of kernels $K$ and $L$

$$\begin{aligned}\mathrm{HSIC}(\mathbf{X}, \mathbf{Y}, K, L) = \\ \mathbb{E}_{\mathbf{X},\mathbf{X}',\mathbf{Y},\mathbf{Y}'}\left[K(\mathbf{X}, \mathbf{X}')L(\mathbf{Y}, \mathbf{Y}')\right] \\ +\mathbb{E}_{\mathbf{X},\mathbf{X}'}\left[K(\mathbf{X}, \mathbf{X}')\right]\mathbb{E}_{\mathbf{Y},\mathbf{Y}'}\left[L(\mathbf{Y}, \mathbf{Y}')\right] \\ -2\mathbb{E}_{\mathbf{X},\mathbf{Y}}\left[\mathbb{E}_{\mathbf{X}'}\left[K(\mathbf{X}, \mathbf{X}')\right]\mathbb{E}_{\mathbf{Y}'}\left[L(\mathbf{Y}, \mathbf{Y}')\right]\right].\end{aligned}$$

They also show the empirical estimate of it to be

$$\mathrm{HSIC}(\mathbf{K}, \mathbf{L}) = (D-1)^{-2}\,\mathrm{Tr}(\mathbf{KHLH}), \tag{3}$$

where $\mathbf{H} = \mathbf{I} - \frac{1}{D}\mathbf{1}\mathbf{1}^T$ is the centering matrix and $\mathbf{K} = K(\mathbf{X}^{(i)}, \mathbf{X}^{(j)}), \mathbf{L} = L(\mathbf{Y}^{(i)}, \mathbf{Y}^{(j)}), i, j = 1{:}D$ are the kernel (Gram) matrices of observations. Crucially, the kernel evaluations for $\mathbf{K}$ take place between $\mathbf{X}^{(i)} = (\mathbf{x}^i_{(1)}, \mathbf{x}^i_{(2)}, \ldots, \mathbf{x}^i_{(k)})$ and $\mathbf{X}^{(j)} = (\mathbf{x}^j_{(1)}, \mathbf{x}^j_{(2)}, \ldots, \mathbf{x}^j_{(k)})$ and not between the individual word embeddings $\mathbf{x}_{(i)}$ and $\mathbf{x}_{(j)}$, and similarly for $\mathbf{L}$. Thus, both $\mathbf{K}$ and $\mathbf{L}$ are square matrices of dimension $D \times D$ . Indeed, for (3) to make sense, the dimensions of $\mathbf{K}$ and $\mathbf{L}$ must match. The matching dimension in our case is the word embedding dimension $D$, while the number of words $k$ and $l$ in the sentences may vary. This is in line with our formalism, which models word vectors as random variables and their dimensions as observations.

Finally, the Centered Kernel Alignment (CKA) (Cortes et al., 2012) is simply defined as

$$\mathrm{CKA}(\mathbf{K}, \mathbf{L}) = \frac{\mathrm{HSIC}(\mathbf{K}, \mathbf{L})}{\sqrt{\mathrm{HSIC}(\mathbf{K}, \mathbf{K})}\sqrt{\mathrm{HSIC}(\mathbf{L}, \mathbf{L})}}. \tag{4}$$

We see now that CKA not only generalises the squared Pearson correlation to the multivariate case, it also allows it to operate in high-dimensional feature spaces, as commonly done in the kernel literature. The reason this is useful is that under certain conditions (when $K$ and $L$ are

| Approach STS | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|
| *Deep Learning Approaches* | | | | | |
| ELMo (BoW) | 55 | 53 | 63 | 68 | 60 |
| Skip-Thought | 41 | 29 | 40 | 46 | 52 |
| InferSent | **61** | 56 | 68 | 71 | 71 |
| USE (DAN) | 59 | 59 | 68 | 72 | 70 |
| USE (Transformer) | **61** | 64 | 71 | 74 | 74 |
| STN (multitask) | 60.6 | 54.7† | 65.8 | 74.2 | 66.4 |
| BERT-Base | 46.9 | 52.8 | 57.2 | 63.5 | 64.5 |
| BERT-Large | 42.6 | 47.4 | 49.3 | 55.6 | 60.4 |
| *Set-based Approaches* | | | | | |
| WMD | 53.0 | 45.9 | 57.2 | 66.9 | 63.1 |
| SoftCardinality | 55.9 | 50.5 | 59.0 | 66.2 | 65.1 |
| DynaMax | 60.9 | 60.3 | 69.5 | **76.7** | 74.6 |
| SIF+PCA | 58.6 | **67.3** | 70.5 | 73.5 | 71.7 |
| MeanPool+COS | 58.3 | 57.9 | 64.9 | 67.6 | 64.3 |
| MaxPool+COS | 57.7 | 53.5 | 67.2 | 69.5 | 68.5 |
| MeanPool+SPR | 60.2 | 65.1 | 70.1 | 74.4 | 73.0 |
| MaxPool+SPR⚒ | **61.0** | 62.9 | 70.9 | 75.9 | **75.8** |
| CKA Linear⚒ | 59.8 | 62.1 | 69.5 | 74.6 | 70.3 |
| CKA Gaussian⚒ | 60.5 | 63.8 | **71.6** | 76.3 | 73.7 |
| CKA dCor⚒ | **61.0** | 63.2 | 71.5 | 75.6 | 72.4 |

Table 1: Mean Pearson correlation on STS tasks for Deep Learning and Set-based methods using fastText. Methods proposed in this work are denoted with ⚒. Values in bold indicate best results per task. Previous results are taken from Perone et al. (2018), Subramanian et al. (2018) and Zhelezniak et al. (2019b,a). † indicates the only STS13 result (to our knowledge) that includes the SMT subtask.

characteristic kernels), HSIC can detect any existing dependence with high probability, as the sample size increases (Gretton et al., 2005b). One can also consider the Uncentered Kernel Alignment (or simply KA) (Cristianini et al., 2002), which can then be seen as a similar generalisation but for the univariate cosine similarity. To the best of our knowledge, KA and CKA in general have never been applied before to measure semantic similarity between sets of word embeddings; therefore this work seeks to introduce them as standard definitions for squared Pearson's $r$ and cosine similarity for such sets.

## 5 Experiments

We now empirically demonstrate the power of the methods and statistical analysis presented in Section 4, through a set of evaluations on the Semantic Textual Similarity (STS) tasks series 2012-2016 (Agirre et al., 2012, 2013, 2014, 2015, 2016; Cer et al., 2017). For methods involving pretrained word embeddings, we use fastText (Bojanowski et al., 2017) trained on Common Crawl (600B tokens), as previous evaluations have indicated that fastText vectors have uniformly the best performance on these tasks out of commonly-used pretrained unsupervised word vectors (Conneau et al., 2017; Perone et al., 2018; Zhelezniak et al., 2019a,b). We provide experiments and significance analysis for additional word vector in the Appendix. The success metric for the STS tasks is the Pearson correlation between the sentence similarity scores provided by human annotators and the scores generated by a candidate algorithm. Note that the dataset for the STS13 SMT subtask is no longer publicly available, so the mean Pearson correlation for STS13 reported in our experiments has been re-calculated accordingly. The code for our experiments builds on the SentEval toolkit (Conneau and Kiela, 2018) and is available on GitHub[1].

We first conduct a set of experiments to validate the observations of Sections 4.1 and 4.2 regarding the performance of cosine similarity and various univariate correlation coefficients when applied to pooled word vectors. These results are depicted in Figure 2, for which we can make the following observations.

First, max and min-pooled vectors consistently outperform mean-pooled vectors when all three representations are compared with Pearson correlation. We hypothesise that this is in part because max and min-pooling remove the outliers (to which Pearson's $r$ is very sensitive) from at least one tail of the distribution whereas mean-pooled vectors have outliers in both tails. This outlier-removing property, however, cannot be taken as a sole explanation behind excellent performance of max-pooled vectors, as max-pooling still tends to outperform mean-pooling when both are compared with correlations that are robust to outliers, as well as on word vectors that have very few outliers to begin with (e.g. word2vec).

In addition, the strong performance of rank correlation coefficients (Spearman's $\rho$ and Kendall's $\tau$) comes solely from their robustness to outliers, as clipping (winsorizing) the top and

[1] https://github.com/Babylonpartners/corrsim

bottom 5% of the values and then proceeding with Pearson's $r$ closes the gap almost completely. Consistently, on vectors with few outliers (word2vec), Pearson's $r$ achieves the same performance as rank correlations even without winsorization. However, unlike outliers, positive (negative) skew of max- (min-) pooled vectors does not seem to hurt Pearson's $r$ on STS tasks.

Next, we conduct evaluations of the methods proposed in this work alongside other deep learning and set-based similarity measures for STS from the literature. The methods we compare are as follows:

- *Deep representation approaches:* BoW with ELMo embeddings (Peters et al., 2018), Skip-Thought (Kiros et al., 2015), InferSent (Conneau et al., 2017), Universal Sentence Encoder both DAN and Transformer (Cer et al., 2018), STN multitask embeddings (Subramanian et al., 2018), and BERT 12- and 24-layer models (Devlin et al., 2018).

- *Set-based similarity measures:* Word Mover's Distance (WMD) (Kusner et al., 2015), soft-cardinality with Jaccard coefficient (Jimenez et al., 2012), DynaMax with Jaccard (Zhelezniak et al., 2019b), mean- and max-pooled word vectors with cosine similarity (COS), and mean-pooled word vectors with Spearman correlation (SPR) (Zhelezniak et al., 2019a).

- *Proposed set-based approaches:* max-pooled word vectors with Spearman correlation, CKA with linear kernel (also known as RV-coefficient), CKA with Gaussian kernel (median estimation for $\sigma^2$), and CKA with distance kernel (distance correlation).

Note that for BERT we evaluated all pooling strategies available in bert-as-service (Xiao, 2018) applied to either the last or second-to-last layers and report results for the best-performing combination, which was mean-pooling on the last layer for both model sizes. Our results are presented in Table 1. We can clearly see that deep learning-based methods do not shine on STS tasks, while simple compositions of word vectors can perform extremely well, especially when an appropriate correlation coefficient is used as the similarity measure. Indeed, the performance of max-pooled vectors with Spearman correlation approaches or exceeds that of more expensive or offline methods like that of Arora et al. (2017), which performs PCA computations on the entire test set. Additionally, while the multivariate correlation methods such as CKA are more computationally expensive than pooling-based approaches (see Table 2), they can provide performance boost on some tasks, making the cost worth it depending on the application. Finally, we conducted an exploratory error analysis and found that many errors are due to the well-known inherent weaknesses of word embeddings. For example, the proposed approaches heavily overestimate similarity when two sentences contain antonyms or when one sentence is the negation of the other. We illustrate these and other cases in the Appendix.

| **Approach** | **Time complexity** |
|---|---|
| MaxPool+SPR | $O(nd + d \log d)$ |
| CKA | $O(nd^2 + d^2)$ |
| DynaMax | $O(n^2 d)$ |
| SoftCard | $O(n^2 d)$ |
| WMD | $O(n^3 \log n \cdot d)$ |
| WMD (relaxed) | $O(n^2 d)$ |

Table 2: Computational complexity of some of the set-based STS methods discussed in this paper. Here $n$ is the sentence length and $d$ is the dimensionality of the word embeddings.

## 6 Conclusion

In this work we investigate the application of statistical correlation coefficients to sets of word vectors as a method for computing semantic textual similarity (STS). This can be done either by pooling these word vectors and computing univariate correlations between the resulting representations, or by applying multivariate correlation coefficients to the sets of vectors directly.

We provide further empirical evidence that outliers in word vector distributions disrupt performance of set-based similarity metrics as previously shown (Zhelezniak et al., 2019a). We also show working methods for solving or avoiding the issue through vector pooling operations, robust correlations or winsorization. In addition, we found that pooling operations in conjunction with univariate correlation coefficients yield one of the strongest results on downstream STS tasks, while being computationally much more efficient than competing set-based methods. Our findings are

supported by a combination of statistical analysis, practical examples and visualisations, and empirical evaluation on standard benchmark datasets.

Both proposed families of approaches serve as strong baselines for future research into STS, as well as useful algorithms for the practitioner, being efficient and simple to implement.

We believe our findings speak to the efficacy of the statistical perspective on word embeddings, which we hope will encourage others to explore further implications of not only this particular framework, but also completely novel interpretations of textual representations.

## Acknowledgements

We would like to thank the three anonymous reviewers for their useful feedback and suggestions.

## References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Inigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 2015. Semeval-2015 task 2: Semantic textual similarity, english, spanish and pilot on interpretability. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 252–263. Association for Computational Linguistics.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. Semeval-2014 task 10: Multilingual semantic textual similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 81–91. Association for Computational Linguistics.

Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016. Semeval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 497–511. Association for Computational Linguistics.

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 385–393. Association for Computational Linguistics.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *sem 2013 shared task: Semantic textual similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 32–43. Association for Computational Linguistics.

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. 2016. A latent variable model approach to pmi-based word embeddings. *Transactions of the Association for Computational Linguistics*, 4:385–399.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A Simple but Tough-to-Beat Baseline for Sentence Embeddings. *International Conference on Learning Representations*.

Yoshua Bengio, Rjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Viktor Botev, Kaloyan Marinov, and Florian Schäfer. 2017. Word importance-based similarity of documents metric (wisdm): Fast and scalable document similarity metric for analysis of scientific documents. In *Proceedings of the 6th International Workshop on Mining Scientific Publications*, WOSP 2017, pages 17–23, New York, NY, USA. ACM.

José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. Nasari: a novel approach to a semantically-aware representation of items. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 567–577. Association for Computational Linguistics.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14. Association for Computational Linguistics.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder for english. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174. Association for Computational Linguistics.

Alexis Conneau and Douwe Kiela. 2018. Senteval: An evaluation toolkit for universal sentence representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*. European Language Resource Association.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680. Association for Computational Linguistics.

Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. 2012. Algorithms for learning kernels based on centered alignment. *Journal of Machine Learning Research*, 13(Mar):795–828.

Nello Cristianini, John Shawe-Taylor, Andre Elisseeff, and Jaz S Kandola. 2002. On kernel-target alignment. In *Advances in neural information processing systems*, pages 367–373.

C. De Boom, S. Van Canneyt, S. Bohez, T. Demeester, and B. Dhoedt. 2015. Learning semantic similarity for very short texts. In *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*, pages 1229–1234.

Cedric De Boom, Steven Van Canneyt, Thomas Demeester, and Bart Dhoedt. 2016. Representation learning for very short texts using weighted word embedding aggregation. *Pattern Recogn. Lett.*, 80(C):150–156.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Zhe Gan, Yunchen Pu, Ricardo Henao, Chunyuan Li, Xiaodong He, and Lawrence Carin. 2017. Learning generic sentence representations using convolutional neural networks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2390–2400. Association for Computational Linguistics.

Arthur Gretton, Olivier Bousquet, Alex Smola, and Bernhard Schölkopf. 2005a. Measuring statistical dependence with hilbert-schmidt norms. In *International conference on algorithmic learning theory*, pages 63–77. Springer.

Arthur Gretton, Ralf Herbrich, Alexander Smola, Olivier Bousquet, and Bernhard Schölkopf. 2005b. Kernel methods for measuring independence. *Journal of Machine Learning Research*, 6(Dec):2075–2129.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1367–1377. Association for Computational Linguistics.

Sergio Jimenez, Claudia Becerra, and Alexander Gelbukh. 2012. Soft cardinality: A parameterized similarity function for text comparison. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics - Volume 1: Proceedings of the Main Conference and the Shared Task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, SemEval '12, pages 449–453, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sergio Jimenez, Fabio Gonzalez, and Alexander Gelbukh. 2010. Text comparison using soft cardinality. In *String Processing and Information Retrieval*, pages 297–302, Berlin, Heidelberg. Springer Berlin Heidelberg.

Sergio Jimenez, Fabio A. Gonzalez, and Alexander Gelbukh. 2015. Soft cardinality in semantic text processing: Experience of the SemEval international competitions. *Polibits*, 51:63–72.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-Thought Vectors. In *Advances in Neural Information Processing Systems*, pages 3294–3302.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. 2019. Similarity of neural network representations revisited. In *ICML*.

Matt J. Kusner, Yu Sun, Nicholas I. Kolkin, and Kilian Q. Weinberger. 2015. From word embeddings to document distances. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning*, volume 37 of *ICML'15*, pages 957–966. JMLR.org.

Quoc Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. In *International Conference on Machine Learning*, pages 1188–1196.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013c. Linguistic Regularities in Continuous Space Word Representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of ACL-08: HLT*, pages 236–244. Association for Computational Linguistics.

Giannis Nikolentzos, Polykarpos Meladianos, Francois Rousseau, Yannis Stavrakas, and Michalis Vazirgiannis. 2017. Multivariate Gaussian document representation from word embeddings for text categorization. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 450–455, Valencia, Spain. Association for Computational Linguistics.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2018. Unsupervised learning of sentence embeddings using compositional n-gram features. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 528–540. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics.

Christian S Perone, Roberto Silveira, and Thomas S Paula. 2018. Evaluation of sentence embeddings in downstream and linguistic probing tasks. *arXiv preprint arXiv:1806.06259*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Enrico Santus, Hongmin Wang, Emmanuele Chersoni, and Yue Zhang. 2018. A rank-based similarity metric for word embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 552–557. Association for Computational Linguistics.

Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J Pal. 2018. Learning general purpose distributed sentence representations via large scale multi-task learning. In *International Conference on Learning Representations*.

Shuai Tang, Hailin Jin, Chen Fang, Zhaowen Wang, and Virginia R. de Sa. 2017. Exploring asymmetric encoder-decoder structure for context-based sentence representation learning. *CoRR*, abs/1710.10380.

Marwan Torki. 2018. A document descriptor using covariance of word vectors. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 527–532, Melbourne, Australia. Association for Computational Linguistics.

Francisco Vargas, Kamen Brestnichki, and Nils Hammerla. 2019. Model comparison for semantic grouping. In *ICML*.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Towards Universal Paraphrastic Sentence Embeddings. In *International Conference on Learning Representations*.

John Wieting and Kevin Gimpel. 2018. Paranmt-50m: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462. Association for Computational Linguistics.

Han Xiao. 2018. bert-as-service. `https://github.com/hanxiao/bert-as-service`.

Rui Zhao and Kezhi Mao. 2017. Fuzzy bag-of-words model for document representation. *IEEE Transactions on Fuzzy Systems*, pages 1–1.

Vitalii Zhelezniak, Dan Busbridge, April Shen, Samuel L. Smith, and Nils Y. Hammerla. 2018. Decoding Decoders: Finding Optimal Representation Spaces for Unsupervised Similarity Tasks. *CoRR*, abs/1805.03435.

Vitalii Zhelezniak, Aleksandar Savkov, April Shen, and Nils Y. Hammerla. 2019a. Correlation coefficients and semantic textual similarity. In *NAACL-HLT*.

Vitalii Zhelezniak, Aleksandar Savkov, April Shen, Francesco Moramarco, Jack Flann, and Nils Y. Hammerla. 2019b. Don't settle for average, go for the max: Fuzzy sets and max-pooled word vectors. In *International Conference on Learning Representations*.

# Game Theory Meets Embeddings: a Unified Framework for Word Sense Disambiguation

**Rocco Tripodi**
Ca' Foscari University of Venice
rocco.tripodi@unive.it

**Roberto Navigli**
Sapienza University of Rome
navigli@di.uniroma1.it

## Abstract

Game-theoretic models, thanks to their intrinsic ability to exploit contextual information, have shown to be particularly suited for the Word Sense Disambiguation task. They represent ambiguous words as the players of a non-cooperative game and their senses as the strategies that the players can select in order to play the games. The interaction among the players is modeled with a weighted graph and the payoff as an embedding similarity function, which the players try to maximize. The impact of the word and sense embedding representations in the framework was tested and analyzed extensively: experiments on standard benchmarks show state-of-art performances and different tests hint at the usefulness of using disambiguation to obtain contextualized word representations.

## 1 Introduction

Word Sense Disambiguation (WSD), the task of linking the appropriate meaning from a sense inventory to words in a text, is an open problem in Natural Language Processing (NLP). It is particularly challenging because it deals with the semantics of words and, by their very nature, words are ambiguous and can be used with different meanings in different situations. Among the key tasks aimed at enabling Natural Language Understanding (Navigli, 2018), WSD provides a basic, solid contribution since it is able to identify the intended meaning of the words in a sentence (Kim et al., 2010).

WSD can be seen as a classification task in which words are the objects to be classified and senses are the classes into which the objects have to be classified (Navigli, 2009); therefore it is possible to use supervised learning techniques to solve the WSD problem. One drawback with this idea is that it requires large amounts of data that are difficult to obtain. Furthermore, in the WSD context, the production of annotated data is even more complicated and excessively time-consuming compared to other tasks. This arises because of the variability in lexical use. Furthermore, the number of different meanings to be considered in a WSD task is in the order of thousands, whereas classical classification tasks in machine learning have considerably fewer classes.

We decided to adopt a semi-supervised approach to overcome the knowledge acquisition bottleneck and innovate the strand of research introduced by Tripodi and Pelillo (2017). These researchers developed a flexible game-theoretic WSD model that exploits word and sense similarity information. This combination of features allows the textual coherence to be maintained: in fact, in this model the disambiguation process is relational, and the sense assigned to a word must always be compatible with the senses of the words in the same text. It can be seen as a constraint satisfaction model which aims to find the best configuration of senses for the words in context. This is possible because the payoff function of the games is modeled in a way in which, when a game is played between two players, they are emboldened to select the senses that have the highest compatibility with the senses that the co-player is choosing. Another appealing feature of this model is that it offers the possibility to configure many components of the system: it is possible to use any word and sense representation; also, one can model the interactions of the players in different ways by exploiting word similarity information, the syntactic structure of the sentence and the importance provided by specific relations. Furthermore, it is possible to use different priors on the sense distributions and to use different game dynamics to find the equilibrium state of the model. Traditional WSD methods have only some of these



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 88–99,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

properties.

The main difference between our model and the model proposed by Tripodi and Pelillo (2017) is that they did not use state-of-the-art models for word and sense representations. They used word co-occurrence measures for word similarity and *tf-idf* vectors for sense similarity, resulting in sparse graphs in which nodes can be disjoint or some semantic area is not covered. Instead, we are advocating the use of dense vectors, which provide a completely different perspective not only in terms of representation but also in terms of dynamics. Each player is involved in many more games and this affects the computation of the payoffs and the convergence of the system. The interaction among the players is defined in a different way and the priors are modeled with a more realistic distribution to avoid the skewness typical of word sense distributions. Furthermore, our model is evaluated on recent standard benchmarks, facilitating comparison with other models.

The main contributions of this paper are as follows:

1. the release of a general framework for WSD;
2. the evaluation of different word and sense embeddings;
3. state-of-the-art performances on standard benchmarks (in different cases performing better than recent supervised models);
4. the use of disambiguated sense vectors to obtain contextualized word representations.

## 2 Word Sense Disambiguation

WSD approaches can be divided into two main categories: supervised, which require human intervention in the creation of sense-annotated datasets, and the so-called knowledge-based approaches (Navigli, 2009), which require the construction of a task-independent lexical-semantic knowledge resource, but which, once that work is available, use models that are completely autonomous.

As regards supervised systems, a popular system is *It makes sense* (Zhong and Ng, 2010), a model which takes advantage of standard WSD features such as POS-tags, word co-occurrences, and collocations and creates individual support vector machine classifiers for each ambiguous word. Newer supervised models exploit deep neural networks and especially long short-term memory (LSTM) networks, a type of recurrent neural network particularly suitable for handling arbitrary-length sequences. Yuan et al. (2016) proposed a deep neural model trained with large amounts of data obtained in a semi-supervised fashion. This model was re-implemented by Le et al. (2018), reaching comparable results with a smaller training corpus. Raganato et al. (2017) introduced two approaches for neural WSD using models developed for machine translation and substituting translated words with sense-annotated ones. A recent work that combines labeled data and knowledge-based information has been proposed by Luo et al. (2018). Uslu et al. (2018) proposed *fastSense*, a model inspired by *fastText* (Joulin et al., 2017) which – rather than predicting context words – predicts word senses.

Knowledge-based models, instead, exploit the structural properties of a lexical-semantic knowledge base, and typically use the relational information between concepts in the semantic graph together with the lexical information contained therein (Navigli and Lapata, 2010). A popular algorithm used to select the sense of each word in this graph is PageRank (Page et al., 1999) that performs random walks over the network to identify the most important nodes (Haveliwala, 2002; Mihalcea et al., 2004; De Cao et al., 2010). An extension of these models was proposed by Agirre et al. (2014) in which the Personalized PageRank algorithm is applied. Another knowledge-based approach is Babelfy (Moro et al., 2014), which defines a semantic signature for a given context and compares it with all the candidate senses in order to perform the disambiguation task. Chaplot and Salakhutdinov (2018) proposed a method that uses the whole document as the context for the words to be disambiguated, exploiting topical information (Ferret and Grau, 2002). It models word senses using a variant of the Latent Dirichlet Allocation framework (Blei et al., 2003), in which the topic distributions of the words are replaced with sense distributions modeled with a logistic normal distribution according to the frequencies obtained from WordNet.

## 3 Word and Sense Embeddings

A good machine-interpretable representation of lexical features is fundamental for every NLP system. A system's performance, however, depends

on the quality of the input representations. Furthermore, the inclusion of semantic features, in addition to lexical ones, has been proven effective in many NLP approaches (Li and Jurafsky, 2015).

Word embeddings, the current paradigm for lexical representation of words, were popularized with word2vec (Mikolov et al., 2013). The main idea is to exploit a neural language model which learns to predict a word occurrence given its surroundings. Another well-known word embedding model was presented by Pennington et al. (2014), which shares the idea of word2vec, but with the difference that it uses explicit latent representations obtained from statistical calculation on word co-occurrences. However, all word embedding models share a common issue: they cannot capture polysemy since they conflate the various word senses into a single vector representation. Several efforts have been presented so far to deal with this problem. SensEmbed (Iacobacci et al., 2015) uses a knowledge-based disambiguation system to build a sense-annotated corpus that, in its turn, is used to train a vector space model for word senses with word2vec. AutoExtend (Rothe and Schütze, 2015), instead, is initialized with a set of pre-trained word embeddings, and induces sense and synset vectors in the same semantic space using an autoencoder. The vectors are induced by constraining their representation given the assumption that synsets are sums of their lexemes. Camacho-Collados et al. (2015) presented NASARI, an approach that learns sense vectors by exploiting the hyperlink structure of the English Wikipedia, linking their representations to the semantic network of BabelNet (Navigli and Ponzetto, 2012). More recent works, such as ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019), are based on language models learned using complex neural network architectures. The advantage of these models is that they can produce different representations of words according to the context in which they appear.

## 4 Game Theory and Game Dynamics

In this work we take a different approach to WSD by employing a model based on game theory (GT). This discipline was introduced by Neuman and Morgenstern (1944) in order to develop a mathematical framework able to model the essentials of decision making in interactive situations. In its *normal-form* representation (Weibull, 1997), it consists of a finite set of players $N = (1, .., n)$, a finite set of pure strategies $S_i = \{1, ..., m_i\}$ for each player $i \in N$, and a payoff (utility) function $u_i : S \rightarrow \mathbb{R}$, that associates a payoff with each combination of strategies in $S = S_1 \times S_2 \times ... \times S_n$. A fundamental assumption in game theory is that each player $i$ tries to maximize the value of $u_i$. Furthermore, in *non-cooperative games* the players choose their strategies independently, considering what choices other players can make and trying to find the best response to the strategy of the co-players.

A player $i$, in addition to playing single (pure) strategies from $S_i$, can also use *mixed strategies*, that are probability distributions over pure strategies. A mixed strategy over $S_i$ is defined as a vector $\mathbf{x_i} = (x_1, \ldots, x_{m_i})$, such that $x_j \geq 0$ and $\sum x_j = 1$. Each mixed strategy corresponds to a point in the simplex $\Delta_m$, whose corners correspond to pure strategies. The intuition is that player $i$ randomises over strategies according to the probabilities in $\mathbf{x_i}$. Each mixed strategy profile lives in the mixed strategy space of the game, given by the Cartesian product $\Theta = \Delta_{m_1} \times \Delta_{m_2} \times \cdots \times \Delta_{m_n}$.

In a *two-player game*, a strategy profile can be defined as a pair $(\mathbf{x}_i, \mathbf{x}_j)$. The expected payoff for this strategy profile is computed as:

$$u(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \cdot A_{ij}\mathbf{x}_j$$

where $A_{ij}$ is the $m_i \times m_j$ payoff matrix between players $i$ and $j$.

In evolutionary game theory (Weibull, 1997), the games are played repeatedly and the players update their mixed strategy distributions over time until no player can improve the payoff obtained with the current mixed strategy. This situation corresponds to the equilibrium of the system.

The payoff corresponding to the $h$-th pure strategy is computed as:

$$u(x_i^h) = x_i^h \cdot \sum_{j=1}^{n_i} (A_{ij}\mathbf{x}_j)^h \quad (1)$$

It is important to note here that the payoff in Equation 1 is additively separable, in fact, the summation is over all the $n_i$ players with whom $i$ is playing the games. The average payoff of player $i$ is calculated as:

$$u(\mathbf{x}_i) = \sum_{h=1}^{m_i} u(x_i^h) \quad (2)$$

To find the Nash equilibrium of the game it is common to use the discrete time version of the *replicator dynamics* equation (Weibull, 1997) for each player $i \in N$,

$$x_i^h(t+1) = x_i^h(t)\frac{u(x_i^h)}{u(\mathbf{x}_i)} \qquad \forall\, h \in \mathbf{x}_i \qquad (3)$$

This equation allows better than average strategies to grow at each iteration. It can be considered as an *inductive learning* process, in which the players learn from past experiences how to play their best strategy. We note that each player optimizes their individual strategy space, but this operation is done according to what other players simultaneously are doing, so the local optimization is the result of a global process.

Game-theoretic models are appealing because they are versatile, interpretable and have a solid mathematical foundation. Furthermore, it is always possible to find the Nash equilibrium in non-cooperative games in mixed strategies (Nash, 1951). In fact, starting from an interior point of $\Theta$, a point $\mathbf{x}$ is a Nash equilibrium only if it is the limit of a trajectory of Equation 3 (Weibull, 1997).

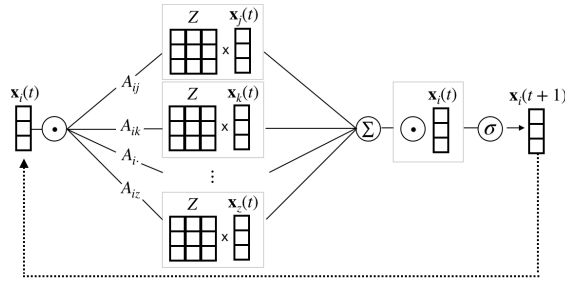


Figure 1: Generic scheme of the model. $\cdot$, $\times$ and $\sigma$ refer to elementwise multiplication, matrix multiplication and normalization, respectively.

## 5 The Model

The model used in this paper, Word Sense Disambiguation Games (*WSDG*), was introduced by Tripodi and Pelillo (2017). It is based on graph-theoretic principles to model the geometry of the data and on game theory to model the learning algorithm which disambiguates the words in a text. It represents the words as the players of a non-cooperative game and their senses as the strategy that the players can select in order to play the games. The players are arranged in a graph whose edges determine the interactions and carry word similarity information. The payoff matrix is encoded as a sense similarity function. The players play the games repeatedly and – at each iteration – update their strategy preferences according to what strategy has been effective in previous games. These preferences, as introduced previously, are encoded as a probability distribution over strategies (senses).

Formally, for a text $T$ we select its content words $W = (1, \ldots, n)$ as the players of the game $I = (1, \ldots, n)$. For each word we use a knowledge base to determine its possible senses. Each sense is represented as a strategy that the player can select from the set $S_i = \{1, ..., m_i\}$, where $m_i$ is the number of senses of word $w_i$. The set of all different senses in the text, $C = \{1, ..., m\}$, is the strategy space of the games. The strategy space is modeled, for each player, as a probability distribution, $\mathbf{x}_i$, of length $m$. It takes non-zero values only on the entries corresponding to the elements of $S_i$. It can be initialized with a normal distribution in the case of unsupervised learning or with information obtained from sense-labeled corpora in the case of semi-supervised learning.

The payoff of a game depends on a payoff matrix $Z$ in which the rows are indexed according to the strategies of player $i$ and the columns according to the strategies of player $j$. Its entries $Z_{r,t}$ are the payoff obtained when player $i$ selects strategy $r$ and player $j$ selects strategy $t$. It is important to note here that the payoff of a game does not depend on the single strategy taken individually by a player, but always by the combination of two simultaneous actions. In WSD this means that the sense selected by a word influences the choices of the other words in the text and this allows the textual coherence to be maintained.

The disambiguation games to build a payoff function require: a word similarity matrix $A$, a sense similarity matrix $Z$ and a sense distribution $\mathbf{x}_i$ for each player $i$. $A$ models the players' interactions, so that similar players play together and the more similar they are the more reciprocal influence they have. It can be interpreted as an attention mechanism (Vaswani et al., 2017) since it weights the payoffs. $Z$ is used to create the payoff matrices of the games so that the more similar the senses of the words are the more the corresponding players are encouraged to select them, since they give a high payoff. $A$ and $Z$ are obtained by computing vector representations of word and sense (see Section 3) and then calculating their pairwise sim-

ilarity.

The strategy space of each player, $i$, is represented as a column vector of length $m$. It is initialized with:

$$x_i^h = \begin{cases} |m_i|^{-1} & \text{if sense } h \text{ is in } S_i, \\ 0 & \text{otherwise.} \end{cases} \quad (4)$$

This initialization is used in the case of unsupervised WSD, since it does not use information from sense-tagged corpora. If instead this information is available, $|m_i|^{-1}$ in Equation 4 is substituted with the frequency of the corresponding sense and then $\mathbf{x}_i$ is normalized in order to sum up to one.

Once these sources of information are computed, the *WSDG* are run by using the replicator dynamic equation (Taylor and Jonker, 1978) in Equation 3, where the payoff of strategy $h$ for player $i$ is calculated as:

$$u(x_i^h) = x_i^h \cdot \sum_{j=1}^{n_i} (A_{ij} Z \mathbf{x}_j)^h \quad (5)$$

where $n_i$ are the neighbours of player $i$ as in the graph $A$. The average payoff is calculated as:

$$u(\mathbf{x}_i) = \sum_{h=1}^{m_i} u(x_i^h) \quad (6)$$

The complexity of *WSDG* scales linearly with the number of words to be disambiguated. Differently from other models based on PageRank, it is possible to disambiguate all the words at the same time. As an example, *WSDG* can disambiguate 200 words (1650 senses) in 7 seconds, on a single CPU core. A generic representation of the model is proposed in Figure 1.

**Implementation details** The cosine similarity was used as similarity measure for both words and senses. The $A$ matrix was treated as the adjacency matrix of an undirected weighted graph and, to reduce the complexity of the model, the edges with weight lower than 0.1 were removed. The symmetric normalized Laplacian of this graph was calculated as $D^{-1/2}AD^{-1/2}$, where $D$ is the degree matrix of graph $A$. Since the algorithm operates on an entire text, local information is added to matrix $A$. The mean value of the matrix is added to the $\lceil log(n) \rceil$ cells on the left of the main diagonal. For BERT, this operation was replaced with its attention layer, adding to matrix $A$ the mean attention distribution of all the heads of the last layer. The choice of the last layer is motivated by the fact that it stores semantic information and its attention distributions have high entropy (Clark et al., 2019). The first singular vector was removed from $A$ in the case of word vectors whose length exceeded 500. This was done to reduce the redundancy of the representations in line with Arora et al. (2017). The distributions for each $\mathbf{x}$ were computed according to SemCor (Miller et al., 1993) and normalized using the softmax function. The replicator dynamics were run until a maximum number of iterations was reached (100) or the difference between two consecutive iterations was below a small threshold ($10^{-3}$), calculated as $\sum_{i=1}^{n} \|\mathbf{x_i}(t-1) - \mathbf{x_i}(t)\|$. The code of the model is available at `https://github.com/roccotrip/wsd_games_emb`.

## 6 Evaluation

The evaluation of our model was conducted using the framework proposed by Raganato et al. (2017). This consists of five datasets which were unified to the same WordNet 3.0 inventory: Senseval-2 (S2), Senseval-3 (S3), SemEval-2007 (SE7), SemEval-2013 (SE13) and SemEval-2015 (SE15). These datasets have in total 26 texts and 10,619 words to be disambiguated. Our objective was to test our game-theoretic model with different settings and to evaluate its performances. To this end we performed experiments comparing 16 different sets of pretrained word embeddings and 7 sets of sense embeddings.

**Word embeddings** As word embedding models we included 4 pre-word2vec models: the *hierarchical log-bilinear* model (Mnih and Hinton, 2007, HLBL), a probabilistic linear neural model which aims to predict the embedding of a word given the concatenation of the previous words; CW (Collobert and Weston, 2008), an embeddings model with a deep unified architecture for multitask NLP; Distributional Memory (Baroni and Lenci, 2010, DM), a semantically enriched count-based model; leskBasile (Basile et al., 2014), a model based on Latent Semantic Analysis reduced via Singular-Value Decomposition; 3 models obtained with word2vec: GoogleNews, a set of 300-dimensions vectors trained with the Google News dataset; BNC-*, vectors of different dimensions trained on the British National Corpus including POS information during training; and w2vR, trained with word2vec on the 2014

dump of the English Wikipedia, enriched with retrofitting (Faruqui et al., 2015), a technique to enhance pre-trained embeddings with semantic information. The enrichment was performed using retrofitting's best configuration, based on the Paraphrase Database (Ganitkevitch et al., 2013, PPDB). We also tested GloVe (Pennington et al., 2014), trained with the concatenation of the 2014 dump of the English Wikipedia and Gigaword 5, and fastText (Bojanowski et al., 2017) trained on Wikipedia 2017, UMBC corpus and the statmt.org news dataset.

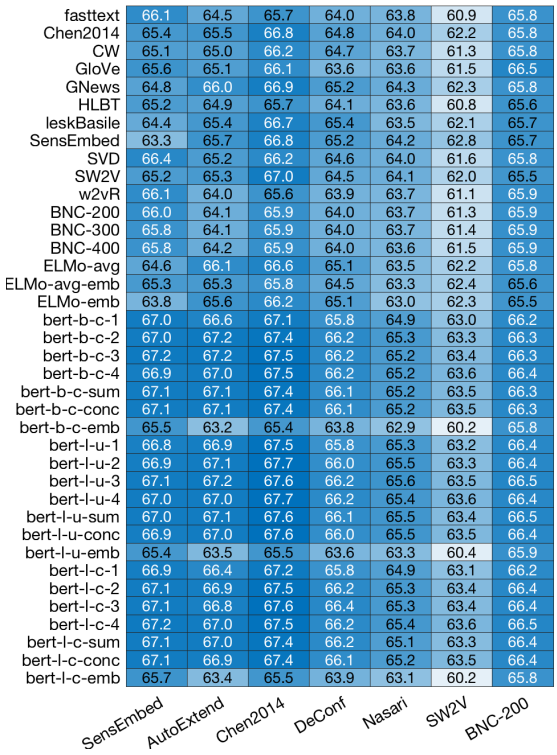


Figure 2: Performances of the model on the union of all datasets. The results are presented as F1 for all combinations of word and sense embeddings. Word vectors are on the rows and sense vectors on the columns.

**Contextualized word embeddings** As contextualized embeddings we used ELMo (Peters et al., 2018) in three different configurations, namely: ELMo-avg, weighted sum of its three layers; ELMo-avg emb, weighted sum of its three layers and the embeddings it produces; and ELMo-emb, the word embeddings produced by the model[1]. We also tested three implementations of BERT (Devlin et al., 2019): base cased (b-c); large uncased (l-u) and large cased (l-c). They offer pre-trained deep bidirectional representations of words in context[2]. We used seven configurations for each model: one for each of the last four layers (numbered from 1 to 4), the sum of these layers, their concatenation and the embedding layer. We fed all these models with the entire texts of the datasets. Since BERT uses WordPiece tokenization, we averaged sub-token embeddings to obtain token-level representations.

We also included three models which were built together with the sense embeddings introduced below.

**Sense embeddings** As sense embeddings, in addition to the three models introduced in Section 3 (AutoExtend, NASARI and SensEmbed), we included four models: Chen et al. (2014), a unified model which learns sense vectors by training a sense-annotated corpus disambiguated with a framework based on semantic similarity of WordNet sense definitions; meanBNC, created using a weighted combination of the words from WordNet glosses, using, as word vectors, the set of BNC-200 mentioned earlier; DeConf (Pilehvar and Collier, 2016), also linked to WordNet, a model where sense vectors are inferred in the same semantic space of pre-trained word embeddings by decomposing the given word representation into its constituent senses; and finally SW2V (Mancini et al., 2017), a model linked to BabelNet which uses a shallow disambiguation step and which, by extending the word2vec architecture, learns word and sense vectors jointly in the same semantic space as an emerging feature.

**Results** The results of these models are reported in Figure 2. One of the most interesting patterns that emerges from the heat map is that there are some combinations of word and sense embeddings that always work better than others. Sense vectors drive the performance of the system, contributing in great part to the accumulation of payoffs during the games. The sense vectors that maintain high performances are SensEmbed, AutoExtended and Chen2014. In particular Chen2014 has high performances with all the word embedding combinations. While these models are specific sense embedding techniques, the construction of BNC-200 follows a very simple method, which in view of these results can be refined using more principled gloss embedding techniques. The performances of

[1]TensorFlow models available at https://tfhub.dev/google/elmo/2.

[2]PyTorch models available at https://github.com/huggingface/pytorch-transformers.

| | model | S2 | S3 | SE07 | SE13 | SE15 | All | N | V | A | R |
|---|---|---|---|---|---|---|---|---|---|---|---|
| semi-sup. | *MFS* | 64.7* | 65.4 | 53.9 | 62.9 | 66.6* | 64.1 | 68.1 | 49.5 | 74.1 | 80.6 |
| | *Babelfy* | 67.0 | 63.5* | 51.6* | 66.4† | 70.3 | 65.5* | 68.6* | 49.9 | 73.2 | 79.8 |
| | *ppr_w2w* | 68.8 | 66.1 | 53.0 | **68.8** | 70.3 | 67.3 | - | - | - | - |
| | *WSD-TM* | 69.0 | 66.9 | 55.6 | 65.3* | 69.6 | 66.9 | 69.7† | 51.2 | **76.0** | **80.9** |
| | *WSDG*$_\alpha$ | 68.7 | **68.3** | **58.9** | 66.4 | 70.7 | **67.7** | **71.1** | **51.9**† | 75.4 | **80.9** |
| | *WSDG*$_\beta$ | 68.9 | 65.5 | 54.5 | 67.0 | **72.8** | 67.2 | 70.4 | 51.3 | 75.7 | 80.6 |
| | *WSDG*$_\gamma$ | **69.3** | 66.4 | 56.0† | 65.9 | 70.8 | 67.2 | 70.4 | 51.5 | 75.1 | 80.6 |
| sup. | *IMS (2010)* | 70.9† | 69.3 | 61.3 | 65.3 | 69.5† | 68.9† | 70.5 | 55.8 | 75.6 | 82.9 |
| | *IMS*$_{w2v}$ | 72.2 | 70.4 | 62.6 | 65.9 | 71.5 | 70.1 | 71.9 | 56.6 | 75.9 | 84.7 |
| | *Yuan*$_{LSTM}$ | 73.8 | 71.8 | 63.5 | 69.5 | 72.6 | 71.5 | - | - | - | - |
| | *Raganato*$_{BLSTM}$ | 72.0 | 69.1 | 64.8 | 66.9 | 71.5 | 69.9 | 71.5 | 57.5 | 75.0 | 83.8 |
| | *GAS* | 72.2 | 70.5† | - | 67.2 | 72.6 | - | - | - | - | - |
| | *fastSense* | 73.5 | 73.5 | 62.4 | 66.2 | 73.2 | - | - | - | - | - |

Table 1: Comparison with state-of-the-art algorithms: unsupervised or knowledge-based (*unsup.*), and supervised (*sup.*). *MFS* refers to the MFS heuristic computed on SemCor on each dataset. The results are provided as F1 and the first result of the semi supervised systems with a statistically significant difference from the best of each dataset is marked with * ($\chi^2$, $p < 0.1$). † indicates the same statistics but including also supervised models.

NASARI are lower compared to lexical vectors: this may be due to our choice to use NASARI-embed, whose vectors have low dimensionality.

The word vectors that have consistently high performances in association with the three sense vectors mentioned above are BERT, Chen2014, SensEmbed and SW2V. This is not surprising since they are able to produce contextualised word representations, performing, in fact, a preliminary disambiguation of the words. In particular, SW2V is specifically tailored for WSD. ELMo and fastText perform slightly worse. The vectors constructed using syntactic information and trained on the BNC corpus have similar performances to the their counterparts trained on larger corpora without the use of syntactic information. If we focus on BERT, we can see that it is able to maintain high performances ($F1 \approx 67$) with all its configurations, except for the embedding layers of all the models (*-emb). The contribution of the sum and concatenation operations is not significant.

**Comparison** We performed a comparison with 3 configurations of our model, one for each of the three best sense vectors: WSDG$_\alpha$, obtained using Chen2014 as sense vectors and BERT-l-u-4 as word vectors; WSDG$_\beta$, obtained using SensEmbed as sense vectors and BERT-l-c-4 as word vectors; and WSDG$_\gamma$, obtained using AutoExtended as sense vectors and BERT-l-u-3 as word vectors.

As comparison systems we included three semi-supervised approaches mentioned above, Babelfy (Moro et al., 2014), ppr$_{w2w}$, the best configuration of UKB (Agirre et al., 2018), and WSD-TM, introduced by Chaplot and Salakhutdinov (2018) (for this model we did not have the possibility to verify the results since its code is not available). In addition, we also report the performances of relevant supervised models, namely: It Makes Sense (Zhong and Ng, 2010, IMS), Iacobacci et al. (2016), Yuan et al. (2016), Raganato et al. (2017), Joulin et al. (2017) and Uslu et al. (2018).

The results of our evaluation are shown in Table 1. As we can see our model achieves state-of-the-art performances on four datasets and on S13 and S15 it performs better than many supervised systems. In general the gap between supervised and semi-supervised systems is reducing. This encourages new research in this direction. Our model fares particularly well on the disambiguation of nouns and verbs. However, the main gap between our models and supervised systems relies upon the disambiguation of verbs.

## 7 Analysis

**Polysemy** As expected, most of the errors made by WSDG$_\alpha$ are on highly polysemous words. Figure 3 shows that the number of wrong answers increases as the number of senses grows, and that the number of wrong answers starts to be higher than that of correct answers when the number of senses for a target word is in the range of 10-15 senses. The words with the highest number of errors are polysemous verbs such as: *say* (34), *make* (24), *find* (21), *have*, (17), *take* (15), *get*, (15) and *do* (13). These are words that in many NLP applications (especially those based on distributional models) are treated as stopwords.

**Sense rank** Mancini et al. (2017) show that senses which are not the most frequent ones are particularly challenging and most sense-based approaches fail to represent them properly. In Figure 4 we report the results of WSDG$_\alpha$ divided per sense rank, where it is possible to see how the performances of the system deteriorate as the rank of the correct sense increases. It is interesting to see that, in the first part of the plot, the performances follow a regular pattern that resembles a power-law distribution. This requires further analysis beyond the scope of this work, along the lines of Ferrer-i Cancho and Vitevitch (2018).

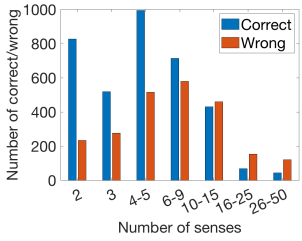


Figure 3: Correct and wrong answers given by WSDG$_\alpha$ grouped by number of senses

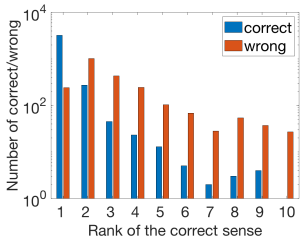


Figure 4: Correct and wrong answers given by WSDG$_\alpha$ per sense rank.

**Priors** Corroborating the findings of Pilehvar and Navigli (2014), Postma et al. (2016) conducted a series of experiments to study the effect that the variation of sense distributions in the training set has on the performances of *It makes sense* (Zhong and Ng, 2010). Specifically, they increased the volume of training examples (V) by enriching SemCor with senses inferred from BabelNet; increased the number of least frequent senses (LFS) (V+LFS); and overfitted the model constructing a training set proportional to the correct sense distribution of the test set (GOLD,GOLD+LFS). We used the same training sets to compute the priors for our system. The results of this analysis are reported in Table 2. These experiments show that increasing the number of training examples has a small beneficial effect. Increasing the number of LFS examples leads to worse results because this is a deviation from a real sense distribution. Further, to work with better semantic representations, this operation should also be accompanied by a similar selection on the training set of sense and word embeddings, otherwise LFS remain underrepresented. Finally, mimicking the distribution of the test set is more beneficial for *WSDG*$_\alpha$ than for IMS, especially when LFS examples are added, suggesting that semi-supervised systems can better adapt to specific domains than supervised systems.

| System | V | V+LFS | GOLD | GOLD+LFS |
|---|---|---|---|---|
| *IMS* | 68.9 | 62.0 | 86.8 | 85.4 |
| *WSDG*$_\alpha$ | 66.4 | 57.5 | 88.4 | 90.8 |

Table 2: Comparison using different priors.

## 8 Exploratory study

We now present three WSD applications in as many tasks: selection of context-sensitive embeddings; sentence similarity; paraphrases detection.

**Context-sensitive embeddings** We used the Word in Context (WiC) dataset (Pilehvar and Camacho-Collados, 2019) for this task. It contains 7466 sentence pairs in which a target word appears in two different contexts. The task consisted of predicting if a target word has the same sense in the two sentences or not. The aim of this experiment was twofold: we wanted to show the usefulness of contextualized word embeddings obtained from WSD systems and to demonstrate that the model was able to maintain the textual coherence. The experiments on this dataset were conducted on the development set (1400 sentence pairs). The comparison was conducted against state-of-the-art models for contextualized word embeddings and sense embeddings: Context2Vec (Melamud et al., 2016) based on a bidirectional LSTM language model; ELMo$_1$, the first LSTM hidden state; ELMo$_3$, the weighted sum of the 3 LSTM layers; BERT$_{\text{base}}$; BERT$_{\text{large}}$. The results of these systems were taken from Pilehvar and Camacho-Collados (2019). We note here that all these models, including WSDG$_\alpha$, do not use training data. They are based on a simple threshold-based classifier, tuned on the development set (638 sentence pairs). WSDG$_\alpha$ disambiguates all the words in each pair of sentences separately and, if the cosine

| C2V | ELMo$_1$ | ELMo$_3$ | BERT$_{base}$ | BERT$_{large}$ | WSDG$_\alpha$ |
|---|---|---|---|---|---|
| 59.7 | 57.1 | 56.3 | 63.6 | 63.8 | **66.2** |

Table 3: Performance on the WiC dataset.

| | Pearson | Spearman | MSE |
|---|---|---|---|
| sense | 46.5 | 43.9 | 7.9 |
| word | 39.8 | 39.9 | 8.6 |

Table 4: *WSDG*$_\alpha$ results on the SICK dataset.

similarity among the senses assigned to the target words is below a threshold (0.9), it classifies the pair as different senses, and as the same sense otherwise. As shown in Table 3 the disambiguation step has a big impact on the results.

**Sentence similarity** We used the SICK dataset (Marelli et al., 2014) for this task. It consists of 9841 sentence pairs that had been annotated with relatedness scores on a 5-point rating scale. We used the test split of this dataset that contains 4906 sentence pairs. The aim of this experiment was to test if disambiguated sense vectors can provide a better representation of sentences than word vectors. We used a simple method to test the two representations: it consisted of representing a sentence as the sum of the disambiguated sense vectors in one case and as the sum of word vectors in the other case. Once the sentence representations had been obtained for both methods the cosine similarity was used to measure their relatedness. The results of this experiment are reported in Table 4 as Pearson and Spearman correlation and Mean Squared Error (MSE). We used the $\alpha$ configuration of our model with Chen2014 to represent senses and BERT-l-u-4 to represent words. As we can see the simplicity of the method leads to low performances for both representations, but sense vectors correlate better than word vectors.

**Paraphrase detection** We used the test set of the Microsoft Research Paraphrase Corpus (Dolan et al., 2004, MRPC) for this task. The corpus contains 1621 sentence pairs that have been annotated with a binary label: 1 if the two sentences constitute a paraphrase and 0 otherwise. In this task we also used the sum of word vectors and the sum of disambiguated sense vectors to represent the sentences, and used part of the training set (10%) in order to tune the threshold parameter below which the sentences are not considered paraphrase. The classification accuracy for the word vectors used by *WSDG*$_\alpha$ was 58.1 whereas the disambiguated sense vectors obtained 66.9.

## 9 Conclusion

In this work we have presented *WSDG*, a flexible game-theoretic model for WSD. It combines game dynamics with most successful word and sense embeddings, therefore showing the potential of an effective combination of the two areas of game theory and word sense disambiguation.

Our approach achieves state-of-the-art performances on four datasets performing particularly well on the disambiguation of nouns and verbs. Beyond the numerical results, in this paper we have presented a model able to construct and evaluate word and sense representations. This is particularly useful since it can serve as a test bed for new word and sense embeddings. In particular, it will be interesting to test new sense embedding models based on contextual embeddings.

Thanks to the flexibility and scalability of our model, as future work we plan to explore in depth its use in different tasks, such as the creation of sentence (document) embeddings and lexical substitution. In fact, we believe that using disambiguated sense vectors, as shown in the context-sensitive embeddings and paraphrase detection studies, can offer a more accurate representation and improve the quality of downstream applications such as sentiment analysis and text classification (see, e.g., (Pilehvar et al., 2017)), machine translation and topic modelling. Encouraged by the good results achieved in our exploratory studies, we plan to develop a new model for contextualised word embeddings based on a game-theoretic framework.

## Acknowledgments

The authors gratefully acknowledge the support of the ODYCCEUS project No. 732942 (first author) and of the ERC Consolidator Grant MOUSSE No. 726487 (second author) under the European Union's Horizon 2020 research and innovation programme.

The experiments have been run on the SCSCF cluster of Ca' Foscari University. The authors thank Ignacio Iacobacci for preliminary work on this paper.

## References


Eneko Agirre, Oier López de Lacalle, and Aitor Soroa. 2018. The risk of sub-optimal use of open source NLP software: UKB is inadvertently state-of-the-art in knowledge-based WSD. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 29–33, Melbourne, Australia. ACL.

Eneko Agirre, Oier López de Lacalle, and Aitor Soroa. 2014. Random walks for knowledge-based word sense disambiguation. *Computational Linguistics*, 40(1):57–84.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *International Conference on Learning Representations*.

Marco Baroni and Alessandro Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4):673–721.

Pierpaolo Basile, Annalina Caputo, and Giovanni Semeraro. 2014. An enhanced Lesk word sense disambiguation algorithm through a distributional semantic model. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1591–1600, Dublin, Ireland. Dublin City University and ACL.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. NASARI: a novel approach to a semantically-aware representation of items. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 567–577, Denver, Colorado. ACL.

Ramon Ferrer-i Cancho and Michael S. Vitevitch. 2018. The origins of zipf's meaning-frequency law. *Journal of the Association for Information Science and Technology*, 69(11):1369–1379.

Devendra Singh Chaplot and Ruslan Salakhutdinov. 2018. Knowledge-based word sense disambiguation using topic models. In *AAAI Conference on Artificial Intelligence*.

Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A unified model for word sense representation and disambiguation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1025–1035, Doha, Qatar. ACL.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. ACL.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proc. ICML*, pages 160–167.

Diego De Cao, Roberto Basili, Matteo Luciani, Francesco Mesiano, and Riccardo Rossi. 2010. Robust and efficient page rank for word sense disambiguation. In *Proceedings of TextGraphs-5 - 2010 Workshop on Graph-based Methods for Natural Language Processing*, pages 24–32, Uppsala, Sweden. ACL.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. ACL.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 350–356, Geneva, Switzerland. COLING.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1606–1615, Denver, Colorado. ACL.

Olivier Ferret and Brigitte Grau. 2002. A bootstrapping approach for robust topic analysis. *Nat. Lang. Eng.*, 8(3):209–233.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764, Atlanta, Georgia. ACL.

Taher H. Haveliwala. 2002. Topic-sensitive pagerank. In *Proceedings of the 11th International Conference on World Wide Web*, WWW '02, pages 517–526, New York, NY, USA. ACM.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. SensEmbed: Learning sense embeddings for word and relational similarity. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the*

*7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 95–105, Beijing, China. ACL.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. Embeddings for word sense disambiguation: An evaluation study. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 897–907, Berlin, Germany. ACL.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, Valencia, Spain. ACL.

Doo Soon Kim, Ken Barker, and Bruce Porter. 2010. Improving the quality of text understanding by delaying ambiguity resolution. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 581–589, Beijing, China. Coling 2010 Organizing Committee.

Minh Le, Marten Postma, Jacopo Urbani, and Piek Vossen. 2018. A deep dive into word sense disambiguation with LSTM. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 354–365, Santa Fe, New Mexico, USA. ACL.

Jiwei Li and Dan Jurafsky. 2015. Do multi-sense embeddings improve natural language understanding? In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1722–1732, Lisbon, Portugal. ACL.

Fuli Luo, Tianyu Liu, Qiaolin Xia, Baobao Chang, and Zhifang Sui. 2018. Incorporating glosses into neural word sense disambiguation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Melbourne, Australia. ACL.

Massimiliano Mancini, Jose Camacho-Collados, Ignacio Iacobacci, and Roberto Navigli. 2017. Embedding words and senses together via joint knowledge-enhanced training. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 100–111, Vancouver, Canada. ACL.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*, pages 216–223, Reykjavik, Iceland. European Languages Resources Association (ELRA).

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional LSTM. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 51–61, Berlin, Germany. ACL.

Rada Mihalcea, Paul Tarau, and Elizabeth Figa. 2004. PageRank on semantic networks, with application to word sense disambiguation. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 1126–1132, Geneva, Switzerland. COLING.

Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013. Exploiting similarities among languages for machine translation. *CoRR*, abs/1309.4168.

George A Miller, Claudia Leacock, Randee Tengi, and Ross T Bunker. 1993. A semantic concordance. In *Proceedings of the workshop on Human Language Technology*, pages 303–308. ACL.

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 641–648, New York, NY, USA. ACM.

Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics*, 2:231–244.

John Nash. 1951. Non-cooperative games. *Annals of Mathematics*, 54(2):286–295.

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM Comput. Surv.*, 41(2):10:1–10:69.

Roberto Navigli. 2018. Natural Language Understanding: Instructions for (present and future) use. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI 2018)*, pages 5697–5702.

Roberto Navigli and Mirella Lapata. 2010. An experimental study of graph connectivity for unsupervised word sense disambiguation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(4):678–692.

Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artif. Intell.*, 193:217–250.

John von Neuman and Oskar Morgenstern. 1944. *Theory of games and economic behavior*. Princeton University Press.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab. Previous number = SIDL-WP-1999-0120.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. ACL.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. ACL.

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2019. WiC: the word-in-context dataset for evaluating context-sensitive meaning representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1267–1273, Minneapolis, Minnesota. ACL.

Mohammad Taher Pilehvar and Nigel Collier. 2016. De-conflated semantic representations. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1680–1690, Austin, Texas. ACL.

Mohammad Taher Pilehvar and Roberto Navigli. 2014. A large-scale pseudoword-based evaluation framework for state-of-the-art word sense disambiguation. *Computational Linguistics*, 40(4):837–881.

Mohammed Taher Pilehvar, José Camacho-Collados, Roberto Navigli, and Nigel Collier. 2017. Towards a Seamless Integration of Word Senses into Downstream NLP Applications. In *Proc. of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 1857–1869, Vancouver, Canada. ACL.

Marten Postma, Ruben Izquierdo Bevia, and Piek Vossen. 2016. More is not always better: balancing sense distributions for all-words word sense disambiguation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3496–3506, Osaka, Japan. The COLING 2016 Organizing Committee.

Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. 2017. Word sense disambiguation: A unified evaluation framework and empirical comparison. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 99–110, Valencia, Spain. ACL.

Sascha Rothe and Hinrich Schütze. 2015. AutoExtend: Extending word embeddings to embeddings for synsets and lexemes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1793–1803, Beijing, China. ACL.

Peter D. Taylor and Leo B. Jonker. 1978. Evolutionary stable strategies and game dynamics. *Mathematical Biosciences*, 40(1):145 – 156.

Rocco Tripodi and Marcello Pelillo. 2017. A game-theoretic approach to word sense disambiguation. *Computational Linguistics*, 43(1):31–70.

Tolga Uslu, Alexander Mehler, Daniel Baumartz, and Wahed Hemati. 2018. FastSense: An efficient word sense disambiguation classifier. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*, Miyazaki, Japan. ELRA.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Jörgen W. Weibull. 1997. *Evolutionary game theory*. MIT press.

Dayu Yuan, Julian Richardson, Ryan Doherty, Colin Evans, and Eric Altendorf. 2016. Semi-supervised word sense disambiguation with neural models. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1374–1385, Osaka, Japan. The COLING 2016 Organizing Committee.

Zhi Zhong and Hwee Tou Ng. 2010. It makes sense: A wide-coverage word sense disambiguation system for free text. In *Proceedings of the ACL 2010 System Demonstrations*, pages 78–83, Uppsala, Sweden. ACL.

# Guided Dialog Policy Learning: Reward Estimation for Multi-Domain Task-Oriented Dialog

**Ryuichi Takanobu**[1], **Hanlin Zhu**[2], **Minlie Huang**[1*]
Institute for AI, BNRist, [1]DCST, [2]IIIS, Tsinghua University, Beijing, China
gxly19@mails.tsinghua.edu.cn, aihuang@tsinghua.edu.cn

## Abstract

Dialog policy decides what and how a task-oriented dialog system will respond, and plays a vital role in delivering effective conversations. Many studies apply Reinforcement Learning to learn a dialog policy with the reward function which requires elaborate design and pre-specified user goals. With the growing needs to handle complex goals across multiple domains, such manually designed reward functions are not affordable to deal with the complexity of real-world tasks. To this end, we propose Guided Dialog Policy Learning, a novel algorithm based on Adversarial Inverse Reinforcement Learning for joint reward estimation and policy optimization in multi-domain task-oriented dialog. The proposed approach estimates the reward signal and infers the user goal in the dialog sessions. The reward estimator evaluates the state-action pairs so that it can guide the dialog policy at each dialog turn. Extensive experiments on a multi-domain dialog dataset show that the dialog policy guided by the learned reward function achieves remarkably higher task success than state-of-the-art baselines.

## 1 Introduction

Dialog policy, deciding the next action that the dialog agent should take at each turn, is a crucial component of a task-oriented dialog system. Among many models, Reinforcement Learning (RL) is commonly used to learn dialog policy (Fatemi et al., 2016; Peng et al., 2017; Chen et al., 2017; Yarats and Lewis, 2018; Lei et al., 2018; He et al., 2018; Su et al., 2018), where users are modeled as a part of the environment and the policy is learned through interactions with users.

While it is too expensive to learn directly from real users since RL requires a large number of samples to train, most existing studies use data-driven approaches to build a dialog system from conversational corpora (Zhao and Eskenazi, 2016; Dhingra et al., 2017; Shah et al., 2018; Shi and Yu, 2018), where a common strategy is to build a user simulator, and then to learn dialog policy through making simulated interactions between an agent and the simulator. A typical reward function on policy learning consists of a small negative penalty at each turn to encourage a shorter session, and a large positive reward when the session ends successfully if the agent completes the user goal.

| | |
|---|---|
| U: | I'm looking for a hotel to stay that has 5 stars and cheap price range. |
| S: | I am sorry that there is no such hotel, would you like to reserve a 3-star hotel as an alternative? |
| U: | I'd prefer a 4-star hotel even if it's a bit expensive. Oh, and I need parking. |
| S: | OK, I find a moderately priced 4-star hotel that includes parking and free wifi. |
| U: | Are there any places to eat around it? |
| S: | Many. Japanese, Indian, French, etc. What kind of food would you like? |

Table 1: An example of the multi-domain task-oriented dialog between the user (U) and the system (S). The dialog proceeds successfully because the system informs the user that no matching hotel exists (the first turn), identifies the new user goal about parking (the second turn), and shifts the topic to the restaurant domain (the third turn), which well understands the user's demand.

However, specifying an effective reward function is challenging in task-oriented dialog. On one hand, the short dialogs resulted from the negative constant rewards are not always efficient. The agent may end a session too quickly to complete the task properly. For example, it is inappropriate to book a 3-star hotel without confirming with the user at the first turn in Table 1. On the other hand, an explicit user goal is essential to evaluate the task success in the reward design, but user goals are hardly available in real situations (Su et al.,

*Corresponding author



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 100–110,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

2016). In addition, the user goal may change as the conversation proceeds. For instance, the user introduces a new requirement for the parking information at the second turn in Table 1.

Unlike a handcrafted reward function that only evaluates the task success at the end of a session, a good reward function should be able to guide the policy dynamically to complete the task during the conversation. We refer to this as the *reward sparsity* issue. Furthermore, the reward function is often manually tweaked until the dialog policy performs desired behaviors. With the growing needs for the system to handle complex tasks across multiple domains, a more sophisticated reward function would be designed, which poses a serious challenge to manually trade off those different factors.

In this paper, we propose a novel model for learning task-oriented dialog policy. The model includes a robust dialog reward estimator based on Inverse Reinforcement Learning (IRL). The main idea is to automatically infer the reward and goal that motivates human behaviors and interactions from the real human-human dialog sessions. Different from conventional IRL that learns a reward function first and then trains the policy, we integrate Adversarial Learning (AL) into the method so that the policy and reward estimator can be learned simultaneously in an alternate way, thus improving each other during training. To deal with reward sparsity, the reward estimator evaluates the generated dialog session using state-action pairs instead of the entire session, which provides reward signals at each dialog turn and guides dialog policy learning better.

To evaluate the proposed approach, we conduct our experiments on a multi-domain, multi-intent task-oriented dialog corpus. The corpus involves large state and action spaces, multiple decision making in one turn, which makes it more challenging for the reward estimator to infer the user goal. Furthermore, we experiment with two different user simulators. The contributions of our work are in three folds:

- We build a reward estimator via Inverse Reinforcement Learning (IRL) to infer an appropriate reward from multi-domain dialog sessions, in order to avoid manual design of reward function.
- We integrate Adversarial Learning (AL) to train the policy and estimator simultaneously, and evaluate the policy using state-action pairs to better guide dialog policy learning.
- We conduct experiments on the multi-domain, multi-intent task-oriented dialog corpus, with different types of user simulators. Results show the superiority of our model to the state-of-the-art baselines.

## 2 Related Work

### 2.1 Multi-Domain Dialog Policy Learning

Some recent efforts have been paid to multi-domain task-oriented dialog systems where users converse with the agent across multiple domains. A natural way to handle multi-domain dialog systems is to learn multiple independent single-domain sub-policies (Wang et al., 2014; Gašić et al., 2015; Cuayáhuitl et al., 2016). Multi-domain dialog completion was also addressed by hierarchical RL which decomposes the task into several sub-tasks in terms of temporal order (Peng et al., 2017) or space abstraction (Casanueva et al., 2018), but the hierarchical structure can be very complex and constraints between different domains should be considered if an agent conveys multiple intents.

### 2.2 Reward Learning in Dialog Systems

Handcrafted reward functions for dialog policy learning require elaborate design. Several reward learning algorithms have been proposed to find better rewards, including supervised learning on expert dialogs (Li et al., 2014), online active learning from user feedback (Su et al., 2016), multi-object RL to aggregate measurements of various aspects of user satisfaction (Ultes et al., 2017), etc. However, these methods still require some knowledge about user goals or annotations of dialog ratings from real users. Boularias et al. (2010) and Barahona and Cerisara (2014) learn the reward from dialogs using linear programming based on IRL, but do not scale well in real applications. Recently, Liu and Lane (2018) use adversarial rewards as the only source of reward signal. It trains a Bi-LSTM as a discriminator that works on the entire session to predict the task success.

### 2.3 Adversarial Inverse Reinforcement Learning

IRL aims to infer the reward function $\mathcal{R}$ underlying expert demonstrations sampled from humans

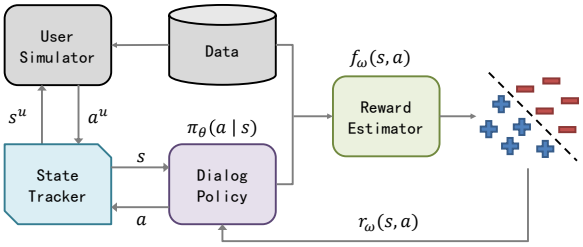


Figure 1: Architecture of GDPL. The dialog policy $\pi$ decides the dialog act $a$ according to the dialog state $s$ provided by the state tracker, and the reward estimator $f$ evaluates the dialog policy by comparing the generated state-action pair $(s, a)$ with the human dialog.

or the optimal policy $\pi^*$. This is similar to the discriminator network in AL that evaluates how realistic the sample looks. Finn et al. (2016) draw a strong connection between GAN and maximum entropy causal IRL (Ziebart et al., 2010) by replacing the estimated data density in AL with the Boltzmann distribution in IRL, i.e. $p(x) \propto \exp(-E(x))$. Several approaches (Ho and Ermon, 2016; Fu et al., 2018) obtain a promising result on automatic reward estimation in large, high-dimensional environments by combining AL with IRL. Inspired by this, we apply AIRL to complex, multi-domain task-oriented dialog, which faces new issues such as discrete action space and language understanding.

## 3 Guided Dialog Policy Learning

We propose Guided Dialog Policy Learning (GDPL), a flexible and practical method on joint reward learning and policy optimization for multi-domain task-oriented dialog systems.

### 3.1 Overview

The overview of the full model is depicted in Fig. 1. The framework consists of three modules: a multi-domain Dialog State Tracker (DST) at the dialog act level, a dialog policy module for deciding the next dialog act, and a reward estimator for policy evaluation.

Specifically, given a set of collected human dialog sessions $\mathcal{D} = \{\tau_1, \tau_2, \dots\}$, each dialog session $\tau$ is a trajectory of state-action pairs $\{s_0^u, a_0^u, s_0, a_0, s_1^u, a_1^u, s_1, a_1, \dots\}$. The user simulator $\mu(a^u, t^u|s^u)$ posts a response $a^u$ according to the user dialog state $s^u$ where $t^u$ denotes a binary terminal signal indicating whether the user wants to end the dialog session. The dialog policy $\pi_\theta(a|s)$ decides the action $a$ according to the current state $s$ and interacts with the simulator $\mu$. During the conversation, DST records the action from one dialog party and returns the state to the other party for deciding what action to take in the next step. Then, the reward estimator $f_\omega(s, a)$ evaluates the quality of the response from the dialog policy, by comparing it with sampled human dialog sessions from the corpus. The dialog policy $\pi$ and the reward estimator $f$ are MLPs parameterized by $\theta, \omega$ respectively. Note that our approach does not need any human supervision during training, and modeling a user simulator is beyond the scope of this paper.

In the subsequent subsections, we will first explain the state, action, and DST used in our algorithm. Then, the algorithm is introduced in a session level, and last followed by a decomposition of state-action pair level.

### 3.2 Multi-Domain Dialog State Tracker

A dialog state tracker keeps track of the dialog session to update the dialog state (Williams et al., 2016; Zhang et al., 2019). It records *informable* slots about the constraints from users and *requestable* slots that indicates what users want to inquiry. DST maintains a separate *belief state* for each slot. Given a user action, the belief state of its slot type is updated according to its slot value (Roy et al., 2000). Action and state in our algorithm are defined as follows:

**Action** : Each system action $a$ or user action $a^u$ is a subset of dialog act set $\mathcal{A}$ as there may be multiple intents in one dialog turn. A *dialog act* is an abstract representation of an intention (Stolcke et al., 2000), which can be represented in a quadruple composed of *domain*, *intent*, *slot type* and *slot value* in the multi-domain setting (e.g. [*restaurant*, *inform*, *food*, *Italian*]). In practice, dialog acts are delexicalized in the dialog policy. We replace the slot value with a count placeholder and refill it with the true value according to the entity selected from the external database, which allows the system to operate on unseen values.

**State** : At dialog turn $t$[1], the system state $s_t = [a_t^u; a_{t-1}; b_t; q_t]$ consists of (I) user action at current turn $a_t^u$; (II) system action at the last turn $a_{t-1}$; (III) all belief state $b_t$ from DST; and (IV) embedding vectors of the number of query results $q_t$ from the external database.

[1]We regard a user turn and a system turn as one dialog turn throughout the paper.

As our model works at the dialog act level, DST can be simply implemented by extracting the slots from actions.

### 3.3 Session Level Reward Estimation

Based on maximum entropy IRL (Ziebart et al., 2008), the reward estimator maximizes the log likelihood of observed human dialog sessions to infer the underlying goal,

$$\omega^* = \arg\max_{\omega} \mathbb{E}_{\tau\sim\mathcal{D}}[f_\omega(\tau)],$$

$$f_\omega(\tau) = \log p_\omega(\tau) = \log \frac{e^{R_\omega(\tau)}}{Z_\omega},$$

$$R_\omega(\tau) = \sum_{t=0}^{T} \gamma^t r_\omega(s_t, a_t),$$

$$Z_\omega = \sum_{\tau} e^{R_\omega(\tau)}.$$

where $f$ models human dialogs as a Boltzmann distribution (Ziebart et al., 2008), $R$ stands for the *return* of a session, i.e. $\gamma$-discounted cumulative rewards, and $Z$ is the corresponding partition function.

The dialog policy is encouraged to mimic human dialog behaviors. It maximizes the expected entropy-regularized return $\mathbb{E}_\pi[R] + H(\pi)$ (Ziebart et al., 2010) based on the *principle of maximum entropy* through minimizing the KL-divergence between the policy distribution and Boltzmann distribution,

$$\begin{aligned} J_\pi(\theta) &= -KL(\pi_\theta(\tau)||p_\omega(\tau)) \\ &= \mathbb{E}_{\tau\sim\pi}[f_\omega(\tau) - \log \pi_\theta(\tau)] \\ &= \mathbb{E}_{\tau\sim\pi}[R_\omega(\tau)] - \log Z_\omega + H(\pi_\theta), \end{aligned}$$

where the term $\log Z_\omega$ is independent to $\theta$, and $H(\cdot)$ denotes the entropy of a model. Intuitively, maximizing entropy is to resolve the ambiguity of language that many optimal policies can explain a set of natural dialog sessions. With the aid of the likelihood ratio trick, the gradient for the dialog policy is

$$\nabla_\theta J_\pi = \mathbb{E}_{\tau\sim\pi}[(f_\omega(\tau) - \log \pi_\theta(\tau))\nabla_\theta \log \pi_\theta(\tau)].$$

In the fashion of AL, the reward estimator aims to distinguish real human sessions and generated sessions from the dialog policy. Therefore, it minimizes KL-divergence with the empirical distribution, while maximizing the KL-divergence with the policy distribution,

$$\begin{aligned} J_f(\omega) &= -KL(p_\mathcal{D}(\tau)||p_\omega(\tau)) + KL(\pi_\theta(\tau)||p_\omega(\tau)) \\ &= \mathbb{E}_{\tau\sim\mathcal{D}}[f_\omega(\tau)] + H(p) - \mathbb{E}_{\tau\sim\pi}[f_\omega(\tau)] - H(\pi_\theta). \end{aligned}$$

**Algorithm 1:** Guided Dialog Policy Learning

**Require:** Dialog corpus $\mathcal{D}$, User simulator $\mu$

**foreach** *training iteration* **do**
- Sample human dialog sessions $D_H$ from $\mathcal{D}$ randomly
- Collect the dialog sessions $D_\Pi$ by executing the dialog policy $\pi$ and interacting with $\mu$, $a^u \sim \mu(\cdot|s^u)$, $a \sim \pi(\cdot|s)$, where $s$ is maintained by DST
- Update the reward estimator $f$ by maximizing $J_f$ w.r.t. $\omega$ adversarially (Eq. 2)
- Compute the estimated reward of each state-action pair in $D_\Pi$, $\hat{r} = f_\omega(s, a) - \log \pi_\theta(a|s)$
- Update $\pi$, $V$ using the estimated reward $\hat{r}$ by maximizing $J_\pi$, $J_V$ w.r.t. $\theta$ (Eq. 3 and Eq. 4)

**end**

Similarly, $H(p)$ and $H(\pi_\theta)$ is independent to $\omega$, so the gradient for the reward estimator yields

$$\nabla_\omega J_f = \mathbb{E}_{\tau\sim\mathcal{D}}[\nabla_\omega f_\omega(\tau)] - \mathbb{E}_{\tau\sim\pi}[\nabla_\omega f_\omega(\tau)].$$

### 3.4 State-Action Level Reward Estimation

So far, the reward estimation uses the entire session $\tau$, which can be very inefficient because of reward sparsity and may be of high variance due to the different lengths of sessions. Here we decompose a session $\tau$ into state-action pairs $(s, a)$ in the reward estimator to address the issues. Therefore, the loss functions for the dialog policy and the reward estimator become respectively as follows:

$$\begin{aligned} J_\pi(\theta) = \mathbb{E}_{s,a\sim\pi}[&\sum_{k=t}^{T} \gamma^{k-t}(f_\omega(s_k, a_k) \\ &- \log \pi_\theta(a_k|s_k))], \end{aligned} \tag{1}$$

$$J_f(\omega) = \mathbb{E}_{s,a\sim\mathcal{D}}[f_\omega(s, a)] - \mathbb{E}_{s,a\sim\pi}[f_\omega(s, a)], \tag{2}$$

where $T$ is the number of dialog turns. Since the reward estimator evaluates a state-action pair, it can guide the dialog policy at each dialog turn with the predicted reward $\hat{r}_\omega(s, a) = f_\omega(s, a) - \log \pi_\theta(a|s)$.

Moreover, the reward estimator $f_\omega$ can be transformed to a reward approximator $g_\omega$ and a shaping term $h_\omega$ according to (Fu et al., 2018) to recover

an interpretable and robust reward from real human sessions. Formally,

$$f_\omega(s_t, a_t, s_{t+1}) = g_\omega(s_t, a_t) + \gamma h_\omega(s_{t+1}) - h_\omega(s_t),$$

where we replace the state-action pair $(s_t, a_t)$ with the state-action-state triple $(s_t, a_t, s_{t+1})$ as the input of the reward estimator. Note that, different from the objective in (Fu et al., 2018) that learns a discriminator in the form $D_\omega(s,a) = \frac{p_\omega(s,a)}{p_\omega(s,a)+\pi(a|s)}$, GDPL directly optimizes $f_\omega$, which avoids unstable or vanishing gradient issue in vanilla GAN (Arjovsky et al., 2017).

In practice, we apply Proximal Policy Optimization (PPO) (Schulman et al., 2017), a simple and stable policy based RL algorithm using a constant clipping mechanism as the soft constraint for dialog policy optimization,

$$J_\pi(\theta) = \mathbb{E}_{s,a\sim\pi}[\min\{\beta_t \hat{A}_t, clip(\beta_t, 1-\epsilon, 1+\epsilon)\hat{A}_t\}], \tag{3}$$

$$\begin{aligned} \hat{A}_t &= \delta_t + \gamma\lambda\hat{A}_{t+1}, \\ \delta_t &= \hat{r}_t + \gamma V_\theta(s_{t+1}) - V_\theta(s_t), \end{aligned}$$

$$J_V(\theta) = -(V_\theta(s_t) - \sum_{k=t}^{T} \gamma^{k-t}\hat{r}_k)^2, \tag{4}$$

where $V_\theta$ is the approximate value function, $\beta_t = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ is the ratio of the probability under the new and old policies, $\hat{A}$ is the estimated advantage, $\delta$ is TD residual, $\lambda$ and $\epsilon$ are hyper-parameters.

In summary, a brief script for GPDL algorithm is shown in Algorithm 1.

# 4 Experimental Setting

## 4.1 Data and Simulators

We use MultiWOZ (Budzianowski et al., 2018), a multi-domain, multi-intent task-oriented dialog corpus that contains 7 domains, 13 intents, 25 slot types, 10,483 dialog sessions, and 71,544 dialog turns in our experiments. Among all the sessions, 1,000 each are used for validation and test. During the data collection process, a user is asked to follow a pre-specified user goal, but it encourages the user to change its goal during the session and the changed goal is also stored, so the collected dialogs are much closer to reality. The corpus also provides the ontology that defines all the entity attributes for the external database.

We apply two user simulators as the interaction environment for the agent. One is the agenda-based user simulator (Schatzmann et al., 2007) which uses heuristics, and the other is a data-driven neural model, namely, Variational Hierarchical User Simulator (VHUS) derived from (Gür et al., 2018). Both simulators initialize a user goal when the dialog starts[2], provide the agent with a simulated user response at each dialog turn, and work at the dialog act level. Since the original corpus only annotates the dialog acts at the system side, we use the annotation at the user side from ConvLab (Lee et al., 2019) to implement the two simulators.

## 4.2 Evaluation Metrics

Evaluation of a task-oriented dialog mainly consists of the cost (dialog turns) and task success (inform F1 & match rate). The definition of inform F1 and match rate is explained as follows.

**Inform F1** : This evaluates whether all the **requested information** (e.g. address, phone number of a hotel) has been informed. Here we compute the F1 score so that a policy which greedily answers all the attributes of an entity will only get a high *recall* but a low *precision*.

**Match rate** : This evaluates whether the booked entities match all the **indicated constraints** (e.g. Japanese food in the center of the city) for all domains. If the agent fails to book an entity in one domain, it will obtain 0 score on that domain. This metric ranges from 0 to 1 for each domain, and the average on all domains stands for the score of a session.

Finally, a dialog is considered successful only if all the information is provided (i.e. inform recall = 1) and the entities are correctly booked (i.e. match rate = 1) as well[3]. Dialog success is either 0 or 1 for each session.

## 4.3 Implementation Details

Both the dialog policy $\pi(a|s)$ and the value function $V(s)$ are implemented with two hidden layer MLPs. For the reward estimator $f(s,a)$, it is split into two networks $g(s,a)$ and $h(s)$ according to the proposed algorithm, where each is a one hidden layer MLP. The activation function is all *Relu*

[2]Refer to the appendix for user goal generation.

[3]If the user does not request any information in the session, this will just compute match rate, and similarly for inform recall.

| Hyper-parameter | Value |
|---|---|
| Learning rate | 1e-4 |
| Mini-batch size | 32 |
| Discount factor $\gamma$ | 0.99 |
| Clipping factor $\epsilon$ in PPO | 0.2 |
| GAE factor $\lambda$ in PPO | 0.95 |

Table 2: Hyper-parameter settings.

for MLPs. We use Adam as the optimization algorithm. The hyper-parameters of GDPL used in our experiments are shown in Table 2.

### 4.4 Baselines

First of all, we introduce three baselines that use handcrafted reward functions. Following (Peng et al., 2017), the agent receives a positive reward of $2L$ for success at the end of each dialog, or a negative reward of $-L$ for failure, where $L$ is the maximum number of turns in each dialog and is set to 40 in our experiments. Furthermore, the agent receives a reward of $-1$ at each turn so that a shorter dialog is encouraged.

**GP-MBCM** (Gašić et al., 2015): Multi-domain Bayesian Committee Machine for dialog management based on Gaussian process, which decomposes the dialog policy into several domain-specific policies.

**ACER** (Wang et al., 2017): Actor-Critic RL policy with Experience Replay, a sample efficient learning algorithm that has low variance and scales well with large discrete action spaces.

**PPO** (Schulman et al., 2017): The same as the dialog policy in GDPL.

Then, we also compare with another strong baseline that involves reward learning.

**ALDM** (Liu and Lane, 2018): Adversarial Learning Dialog Model that learns dialog rewards with a Bi-LSTM encoding the dialog sequence as the discriminator to predict the task success. The reward is only estimated at the end of the session and is further used to optimize the dialog policy.

For a fair comparison, each method is pre-trained for 5 epoches by simple imitation learning on the state-action pairs.

## 5 Result Analysis

### 5.1 Main Results

The performance of each approach that interacts with the agenda-based user simulator is shown in Table 3. GDPL achieves extremely high performance in the task success on account of the substantial improvement in inform F1 and match rate over the baselines. Since the reward estimator of GDPL evaluates state-action pairs, it can always guide the dialog policy during the conversation thus leading the dialog policy to a successful strategy, which also indirectly demonstrates that the reward estimator has learned a reasonable reward at each dialog turn. Surprisingly, GDPL even outperforms human in completing the task, and its average dialog turns are close to those of humans, though GDPL is inferior in terms of match rate. Humans almost manage to make a reservation in each session, which contributes to high task success. However, it is also interesting to find that human have low inform F1, and that may explain why the task is not always completed successfully. Actually, there have high recall (*86.75%*) but low precision (*54.43%*) in human dialogs when answering the requested information. This is possibly because during data collection human users forget to ask for all required information of the task, as reported in (Su et al., 2016).

| Method | Agenda | | | |
|---|---|---|---|---|
| | Turns | Inform | Match | Success |
| GP-MBCM | 2.99 | 19.04 | 44.29 | 28.9 |
| ACER | 10.49 | 77.98 | 62.83 | 50.8 |
| PPO | 9.83 | 83.34 | 69.09 | 59.1 |
| ALDM | 12.47 | 81.20 | 62.60 | 61.2 |
| GDPL-sess | **7.49** | 88.39 | 77.56 | 76.4 |
| GDPL-discr | 7.86 | 93.21 | 80.43 | 80.5 |
| GDPL | 7.64 | **94.97** | **83.90** | **86.5** |
| *Human* | *7.37* | *66.89* | *95.29* | *75.0* |

Table 3: Performance of different dialog agents on the multi-domain dialog corpus by interacting with the agenda-based user simulator. All the results except "dialog turns" are shown in percentage terms. Real human-human performance computed from the test set (i.e. the last row) serves as the upper bounds.

ACER and PPO obtain high performance in inform F1 and match rate as well. However, they obtain poor performance on the overall task success, even when they are provided with the designed reward that already knows the real user goals. This is because they only receive the reward about the success at the last turn and fail to understand what the user needs or detect the change of user goals.

Though ALDM obtains a lower inform F1 and match rate than PPO, it gets a slight improvement

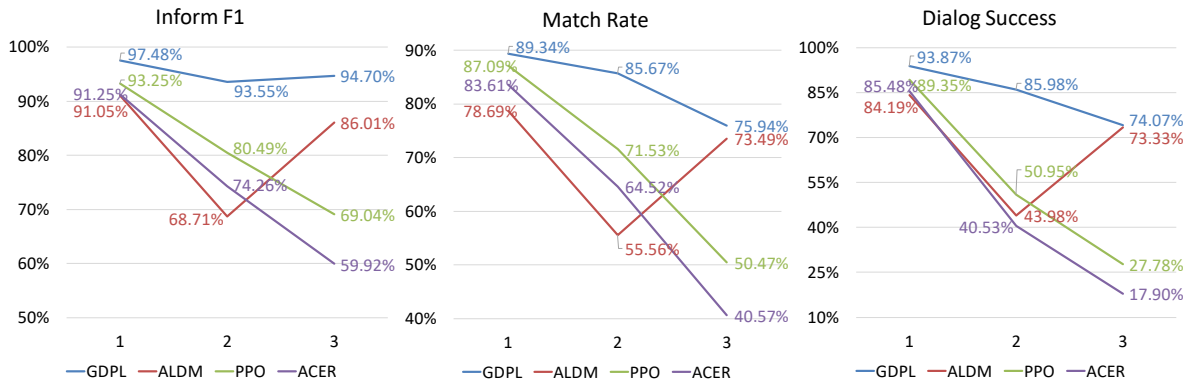


Figure 2: Performance of dialog agents according to the different number of domains in the dialog session. The ratio of the sessions with 1:2:3 domains is 310:528:162 respectively.

| GP-MBCM | ACER | PPO | ALDM | GDPL |
|---|---|---|---|---|
| 1.666 | 0.775 | 0.639 | 1.069 | **0.238** |

Table 4: KL-divergence between different dialog policy and the human dialog $KL(\pi_{turns}||p_{turns})$, where $\pi_{turns}$ denotes the discrete distribution over the number of dialog turns of simulated sessions between the policy $\pi$ and the agenda-based user simulator, and $p_{turns}$ for the real human-human dialog.

| Method | VHUS | | | |
|---|---|---|---|---|
| | Turns | Inform | Match | Success |
| ACER | 22.35 | 55.13 | 33.08 | 18.6 |
| PPO | **19.23** | **56.31** | 33.08 | 18.3 |
| ALDM | 26.90 | 54.37 | 24.15 | 16.4 |
| GDPL | 22.43 | 52.58 | **36.21** | **19.7** |

Table 5: Performance of different agents on the neural user simulator.

on task success by encoding the entire session in its reward estimator. This demonstrates that learning effective rewards can help the policy to capture user intent shift, but the reward sparsity issue remains unsolved. This may explain why the gain is limited, and ALDM even has longer dialog turns than others. In conclusion, the dialog policy benefits from the guidance of the reward estimator per dialog turn.

Moreover, GDPL can establish an efficient dialog thanks to the learned rewards that infer human behaviors. Table 4 shows that GDPL has the smallest KL-divergence to the human on the number of dialog turns over the baselines, which implies that GDPL behaves more like the human. It seems that all the approaches generate many more *short dialogs* (dialog turns less than 3) than human, but GDPL generates far less *long dialogs* (dialog turns larger than 11) than other baselines except GP-MBCM. Most of the long dialog sessions fail to reach a task success.

We also observe that GP-MBCM tries to provide many dialog acts to avoid the negative penalty at each turn, which results in a very low inform F1 and short dialog turns. However, as explained in the introduction, a shorter dialog is not always the best. The dialog generated by GP-MBCM is too short to complete the task successfully. GP-MBCM is a typical case that focuses too much on the cost of the dialog due to the handcrafted reward function and fails to realize the true target that helps the users to accomplish their goals.

### 5.2 Ablation Study

Ablation test is investigated in Table 3. GDPL-sess sums up all the rewards at each turn to the last turn and does not give any other reward before the dialog terminates, while GDPL-discr is to use the discriminator form as (Fu et al., 2018) in the reward estimator. It is perceptible that GDPL has better performance than GDPL-sess on the task success and is comparable regarding the dialog turns, so it can be concluded that GDPL does benefit from the guidance of the reward estimator at each dialog turn, and well addresses the reward sparsity issue. GDPL also outperforms GDPL-discr which means directly optimizing $f_\omega$ improves the stability of AL.

### 5.3 Interaction with Neural Simulator

The performance that the agent interacts with VHUS is presented in Table 5. VHUS has poor performance on multi-domain dialog. It sometimes becomes insensible about the dialog act so

| VS. | Efficiency | | | Quality | | | Success | | |
|---|---|---|---|---|---|---|---|---|---|
| | W | D | L | W | D | L | W | D | L |
| ACER | 55 | 25 | 20 | 44 | 32 | 24 | 52 | 30 | 18 |
| PPO | 74 | 13 | 13 | 56 | 26 | 18 | 59 | 31 | 10 |
| ALDM | 69 | 19 | 12 | 49 | 25 | 26 | 61 | 24 | 15 |

Table 6: The count of human preference on dialog session pairs that GDPL wins (W), draws with (D) or loses to (L) other methods based on different criteria. One method wins the other if the majority prefer the former one.

| Type | Inform | | Match | | Success | |
|---|---|---|---|---|---|---|
| | Mean | Num | Mean | Num | Mean | Num |
| Full | 8.413 | 903 | 10.59 | 450 | 11.18 | 865 |
| Other | -99.95 | 76 | -48.15 | 99 | -71.62 | 135 |

Table 7: Return distribution of GDPL on each metric. The first row counts the dialog sessions that get the full score of the corresponding metric, and the results of the rest sessions are included in the second row.

it often gives unreasonable responses. Therefore, it is more laborious for the dialog policy to learn a proper strategy with the neural user simulator. All the methods cause a significant drop in performance when interacting with VHUS. ALDM even gets worse performance than ACER and PPO. In comparison, GDPL is still comparable with ACER and PPO, obtains a better match rate, and even achieves higher task success. This indicates that GDPL has learned a more robust reward function than ALDM.

### 5.4 Goal across Multiple Domains

Fig. 2 shows the performance with the different number of domains in the user goal. In comparison with other approaches, GDPL is more scalable to the number of domains and achieves the best performance in all metrics. PPO suffers from the increasing number of the domain and has remarkable drops in all metrics. This demonstrates the limited capability for the handcrafted reward function to handle complex tasks across multiple domains in the dialog.

ALDM also has a serious performance degradation with 2 domains, but it is interesting to find that ALDM performs much better with 3 domains than with 2 domains. We further observe that ALDM performs well on the taxi domain, most of which appear in the dialogs with 3 domains. Taxi domain has the least slots for constraints and requests, which makes it easier to learn a reward about that domain, thus leading ALDM to a local optimal. In general, our reward estimator has higher effectiveness and scalability.

### 5.5 Human Evaluation

For human evaluation, we hire Amazon Mechanical Turkers to state their preferences between GDPL and other methods. Because all the policies work at dialog act level, we generate the texts from dialog acts using hand-crafted templates to make the dialog readable. Given a certain user goal, Turkers first read two simulated dialog sessions, one from the interaction between GDPL and the agenda-based user simulator, the other from another baseline with the same simulator. Then, they are asked to judge which dialog is better (win, draw or lose) according to different subjective assessments. In addition to *Task Success*, we examine another two measures concerning *Dialog Cost* in the human evaluation: *Efficiency* such as dialog turn cost or response delay, and *Quality* such as redundant information or inappropriate reply (Walker et al., 1997). Since the text is generated by templates for all policies, we do not evaluate language generation here (including grammar, diversity, etc.). We randomly sample 300 user goals from the test set, 100 each for one baseline, and each session pair is evaluated by 3 Turkers.

Table 6 presents the results of human evaluation. GDPL outperforms three baselines significantly in all aspects (sign test, p-value $< 0.01$) except for the quality compared with ACER. Among all the baselines, GDPL obtains the most preference against PPO. Note that the difference between PPO and GDPL is only in the reward signal. This again demonstrates the advantage of reward learning in GDPL over the handcrafted reward function. The agreement on the superiority of GDPL between objective rating in Table 3 and human preference here also indicates that the automatic metrics used in our experiments is reliable to reflect user satisfaction to some extent.

### 5.6 Reward Evaluation

To provide an insight into the learned reward function itself, Table 7 provides a quantitative evaluation on the learned rewards by showing the distribution of the return $R = \sum_t \gamma^t r_t$ according to each metric. It should be noted that some sessions do not have inform F1 because the user does not request any information, and similarly for match

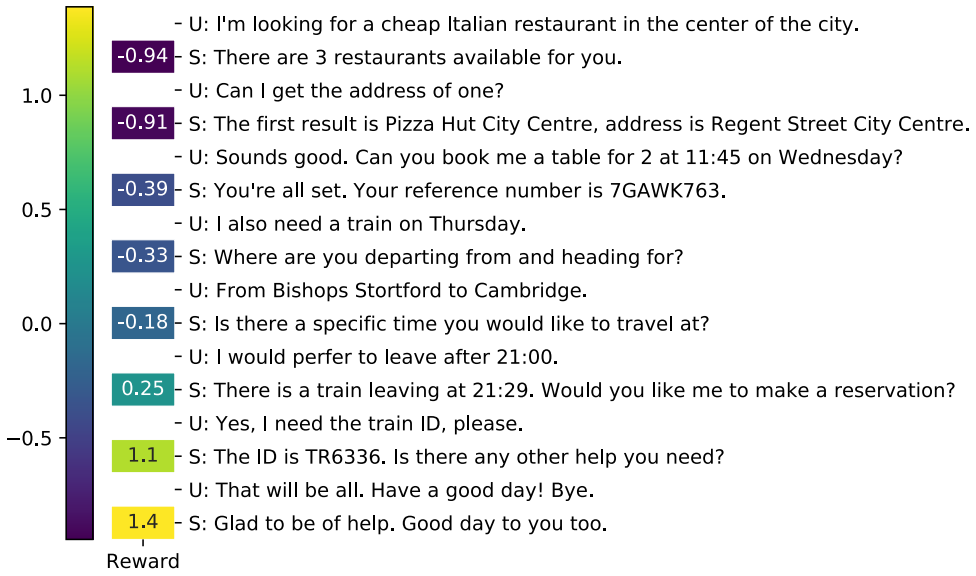


Figure 3: Reward visualization of a dialog session simulated between GDPL and the agenda-based user simulator that contains restaurant and train domains.[4]

rate. It can be observed that the learned reward function has good interpretability in that the reward is positive when the dialog gets a full score on each metric, and negative otherwise.

Fig. 3 gives an illustration of the learned reward at each dialog turn as a qualitative evaluation. In the beginning, the agent is unaware of the user goal thus it starts with a low reward. As the dialog proceeds, the agent has collected enough information from the user, then books the restaurant successfully and the reward remarkably increases at the third turn. The reward continues to grow stably after the topic shifts to the train domain. Again, the agent offers the correct train ID given sufficient information. Since the user has been informed all the requested information and the restaurant and train are both booked successfully, the user leaves the session with satisfaction at last, and the reward rises to the top as well. In brief, the learned reward can well reflect the current state of the dialog. It is also noticeable that the dialog policy manages to express multiple intents during the session.

## 6 Discussion

In this paper, we propose a guided policy learning method for joint reward estimation and policy optimization in multi-domain task-oriented dialog. The method is based on Adversarial Inverse Reinforcement Learning. Extensive experiments demonstrate the effectiveness of our proposed approach and that it can achieve higher task success and better user satisfaction than state-of-the-art baselines.

Though the action space $\mathcal{A}$ of the dialog policy is defined as the set of all dialog acts, it should be noted that GDPL can be equipped with NLU modules that identify the dialog acts expressed in utterance, and with NLG modules that generate utterances from dialog acts. In this way, we can construct the framework in an end-to-end scenario.

The agenda-based user simulator is powerful to provide a simulated interaction for the dialog policy learning, however, it needs careful design and is lack of generalization. While training a neural user simulator is quite challenging due to the high diversity of user modeling and the difficulty of defining a proper reward function, GDPL may offer some solutions for multi-agent dialog policy learning where the user is regarded as another agent and trained with the system agent simultaneously. We leave this as the future work.

[4]Refer to the appendix for the dialog acts.

## Acknowledgement

This work was supported by the National Science Foundation of China (Grant No. 61936010 / 61876096) and the National Key R&D Program of China (Grant No. 2018YFC0830200). We would like to thank THUNUS NExT Joint-Lab for the support, anonymous reviewers for their valuable suggestions, and our lab mate Qi Zhu for helpful discussions. The code is available at https://github.com/truthless11/GDPL.

## References


Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein generative adversarial networks. In *34th International Conference on Machine Learning*, pages 214–223.

Lina M Rojas Barahona and Christophe Cerisara. 2014. Bayesian inverse reinforcement learning for modeling conversational agents in a virtual environment. In *15th International Conference on Computational Linguistics and Intelligent Text Processing*, pages 503–514.

Abdeslam Boularias, Hamid R Chinaei, and Brahim Chaib-draa. 2010. Learning the reward model of dialogue pomdps from data. In *24th Annual Conference on Neural Information Processing Systems, Workshop on Machine Learning for Assistive Technologies*.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. Multiwoz: A large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In *2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026.

Iñigo Casanueva, Paweł Budzianowski, Pei-Hao Su, Stefan Ultes, Lina M Rojas Barahona, Bo-Hsiang Tseng, and Milica Gašić. 2018. Feudal reinforcement learning for dialogue management in large domains. In *2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 714–719.

Lu Chen, Xiang Zhou, Cheng Chang, Runzhe Yang, and Kai Yu. 2017. Agent-aware dropout dqn for safe and efficient on-line dialogue policy learning. In *2017 Conference on Empirical Methods in Natural Language Processing*, pages 2454–2464.

Heriberto Cuayáhuitl, Seunghak Yu, Ashley Williamson, and Jacob Carse. 2016. Deep reinforcement learning for multi-domain dialogue systems. In *30th Annual Conference on Neural Information Processing Systems, Deep Reinforcement Learning Workshop*.

Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2017. Towards end-to-end reinforcement learning of dialogue agents for information access. In *55th Annual Meeting of the Association for Computational Linguistics*, pages 484–495.

Mehdi Fatemi, Layla El Asri, Hannes Schulz, Jing He, and Kaheer Suleman. 2016. Policy networks with two-stage training for dialogue systems. In *17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 101–110.

Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. 2016. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. In *30th Annual Conference on Neural Information Processing Systems, Workshop on Adversarial Training*.

Justin Fu, Katie Luo, and Sergey Levine. 2018. Learning robust rewards with adversarial inverse reinforcement learning. In *6th International Conference on Learning Representations*.

Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2015. Policy committee for adaptation in multi-domain spoken dialogue systems. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 806–812.

Izzeddin Gür, Dilek Hakkani-Tür, Gokhan Tür, and Pararth Shah. 2018. User modeling for task oriented dialogues. In *2018 IEEE Spoken Language Technology Workshop*, pages 900–906.

He He, Derek Chen, Anusha Balakrishnan, and Percy Liang. 2018. Decoupling strategy and generation in negotiation dialogues. In *2018 Conference on Empirical Methods in Natural Language Processing*, pages 2333–2343.

Jonathan Ho and Stefano Ermon. 2016. Generative adversarial imitation learning. In *30th Annual Conference on Neural Information Processing Systems*, pages 4565–4573.

Sungjin Lee, Qi Zhu, Ryuichi Takanobu, Zheng Zhang, Yaoqin Zhang, Xiang Li, Jinchao Li, Baolin Peng, Xiujun Li, Minlie Huang, and Jianfeng Gao. 2019. Convlab: Multi-domain end-to-end dialog system platform. In *57th Annual Meeting of the Association for Computational Linguistics*, pages 64–69.

Wenqiang Lei, Xisen Jin, Min-Yen Kan, Zhaochun Ren, Xiangnan He, and Dawei Yin. 2018. Sequicity: Simplifying task-oriented dialogue systems with single sequence-to-sequence architectures. In *56th Annual Meeting of the Association for Computational Linguistics*, pages 1437–1447.

Lihong Li, He He, and Jason D Williams. 2014. Temporal supervised learning for inferring a dialog policy from example conversations. In *2014 IEEE Spoken Language Technology Workshop*, pages 312–317.

Bing Liu and Ian Lane. 2018. Adversarial learning of task-oriented neural dialog models. In *19th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 350–359.

Baolin Peng, Xiujun Li, Lihong Li, Jianfeng Gao, Asli Celikyilmaz, Sungjin Lee, and Kam-Fai Wong. 2017. Composite task-completion dialogue policy learning via hierarchical deep reinforcement learning. In *2017 Conference on Empirical Methods in Natural Language Processing*, pages 2231–2240.

Nicholas Roy, Joelle Pineau, and Sebastian Thrun. 2000. Spoken dialogue management using probabilistic reasoning. In *38th Annual Meeting of the Association for Computational Linguistics*, pages 93–100.

Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young. 2007. Agenda-based user simulation for bootstrapping a pomdp dialogue system. In *2007 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 149–152.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Pararth Shah, Dilek Hakkani-Tür, Bing Liu, and Gokhan Tür. 2018. Bootstrapping a neural conversational agent with dialogue self-play, crowdsourcing and on-line reinforcement learning. In *2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 41–51.

Weiyan Shi and Zhou Yu. 2018. Sentiment adaptive end-to-end dialog systems. In *56th Annual Meeting of the Association for Computational Linguistics*, pages 1509–1519.

Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, Paul Taylor, Rachel Martin, Carol Van Ess-Dykema, and Marie Meteer. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational linguistics*, 26(3):339–373.

Pei-Hao Su, Milica Gašić, Nikola Mrkšić, Lina M Rojas Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. On-line active reward learning for policy optimisation in spoken dialogue systems. In *54th Annual Meeting of the Association for Computational Linguistics*, pages 2431–2441.

Shang-Yu Su, Xiujun Li, Jianfeng Gao, Jingjing Liu, and Yun-Nung Chen. 2018. Discriminative deep dyna-q: Robust planning for dialogue policy learning. In *2018 Conference on Empirical Methods in Natural Language Processing*, pages 3813–3823.

Stefan Ultes, Paweł Budzianowski, Iñigo Casanueva, Nikola Mrkšić, Lina M Rojas Barahona, Pei-Hao Su, Tsung-Hsien Wen, Milica Gašić, and Steve Young. 2017. Reward-balancing for statistical spoken dialogue systems using multi-objective reinforcement learning. In *18th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 65–70.

Marilyn A Walker, Diane J Litman, Candace A Kamm, and Alicia Abella. 1997. Paradise: A framework for evaluating spoken dialogue agents. In *35th Annual Meeting of the Association for Computational Linguistics*, pages 271–280.

Zhuoran Wang, Hongliang Chen, Guanchun Wang, Hao Tian, Hua Wu, and Haifeng Wang. 2014. Policy learning for domain selection in an extensible multi-domain spoken dialogue system. In *2014 Conference on Empirical Methods in Natural Language Processing*, pages 57–67.

Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. 2017. Sample efficient actor-critic with experience replay. In *5th International Conference on Learning Representations*.

Jason D Williams, Antoine Raux, and Matthew Henderson. 2016. The dialog state tracking challenge series: A review. *Dialogue & Discourse*, 7(3):4–33.

Denis Yarats and Mike Lewis. 2018. Hierarchical text generation and planning for strategic dialogue. In *35th International Conference on Machine Learning*, pages 5587–5595.

Zheng Zhang, Minlie Huang, Zhongzhou Zhao, Feng Ji, Haiqing Chen, and Xiaoyan Zhu. 2019. Memory-augmented dialogue management for task-oriented dialogue systems. *ACM Transactions on Information Systems*, 37(3):34.

Tiancheng Zhao and Maxine Eskenazi. 2016. Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. In *17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 1–10.

Brian D Ziebart, J Andrew Bagnell, and Anind K Dey. 2010. Modeling interaction via the principle of maximum causal entropy. In *27th International Conference on Machine Learning*, pages 1255–1262.

Brian D Ziebart, Andrew Maas, J Andrew Bagnell, and Anind K Dey. 2008. Maximum entropy inverse reinforcement learning. In *23rd AAAI Conference on Artificial Intelligence*, pages 1433–1438.

# Multi-hop Selector Network for Multi-turn Response Selection in Retrieval-based Chatbots

**Chunyuan Yuan**[1,2,†], **Shangwen Lv**[1,2,†], **Mingming Li**[1,2], **Wei Zhou**[1,2,*], **Fuqing Zhu**[2], **Jizhong Han**[2] **and Songlin Hu**[1,2]
[1]School of Cyber Security, University of Chinese Academy of Sciences
[2]Institute of Information Engineering, Chinese Academy of Sciences
{yuanchunyuan, lvshangwen, limingming, zhouwei,
zhufuqing, hanjizhong, husonglin}@iie.ac.cn

## Abstract

Multi-turn retrieval-based conversation is an important task for building intelligent dialogue systems. Existing works mainly focus on matching candidate responses with every context utterance on multiple levels of granularity, which ignore the side effect of using excessive context information. Context utterances provide abundant information for extracting more matching features, but it also brings noise signals and unnecessary information.

In this paper, we will analyze the side effect of using too many context utterances and propose a multi-hop selector network (MSN) to alleviate the problem. Specifically, MSN firstly utilizes a multi-hop selector to select the relevant utterances as context. Then, the model matches the filtered context with the candidate response and obtains a matching score. Experimental results show that MSN outperforms some state-of-the-art methods on three public multi-turn dialogue datasets.

## 1 Introduction

Building a dialogue system that can naturally and consistently converse with humans has drawn increasing research interests in past years. Existing works on building dialogue systems include generation-based and retrieval-based methods. Compared with generation-based methods, retrieval-based methods have advantages in providing fluent and informative responses. Many industrial products have applied retrieval-based dialogue system, e.g., the E-commerce assistant AliMe Assist from Alibaba Group (Li et al., 2017) and the XiaoIce (Shum et al., 2018) from Microsoft.

Early studies (Tan et al., 2015; Yan et al., 2016; Wan et al., 2016) of retrieval-based dialogue system focus on response selection for single-turn conversation. Recently, researchers have begun to pay attention to the multi-turn conversation, aiming at selecting the most related response from a set of candidates given the context utterances of a conversation. Some effective models, such as Sequential Matching Network (SMN) (Wu et al., 2017), Deep Attention Matching network (DAM) (Zhou et al., 2018c), Multi-Representation Fusion Network (MFRN) (Tao et al., 2019), have been proposed to capture the matching features on multiple levels of granularity (words, phrases, sentences, etc.) and short-term and long-term dependencies among words.

Previous works have shown that utilizing multi-turn utterances can further improve the matching performance than only using single-turn utterance (i.e., last utterance). But context utterance is a "double-edged sword", it also provides a lot of noise while providing abundant information, which would influence the performance due to the sensitivity of these matching-based methods.

Table 1: An error case of SMN (Wu et al., 2017), DAM (Zhou et al., 2018c) from E-commerce Corpus. The scores in the table are matching scores predicted by the models.

| Turns | Dialogue Text | SMN | DAM |
|---|---|---|---|
| Turn-1 | A: Are there any **discounts** activities recently? | | |
| Turn-2 | B: No. Our product have been **cheaper** than before. | | |
| Turn-3 | A: Oh. | | |
| Turn-4 | B: Hum! | | |
| Turn-5 | A: I'll buy these nuts. Can you sell me **cheaper**? | | |
| Turn-6 | B: You can get some **coupons** on the homepage. | | |
| Turn-7 | A: Will you give me some nut **clips**? | | |
| Turn-8 | B: Of course **we will**. | | |
| **Turn-9** | A: **How many clips** will you give? | | |
| Resp-1 | **One clip** for every package. (True) | 0.832 | 0.854 |
| Resp-2 | OK, **we will** give you a **coupons** worth $1. (False) | 0.925 | 0.947 |

To illustrate the problem, we show an error case of SMN (Wu et al., 2017) and DAM (Zhou et al., 2018c) from E-commerce Corpus in Table 1. We can see that although "Resp-1" is the right answer for utterance "Turn-9", the SMN and DAM mod-

†Equally contributed.
* Corresponding author.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 111–120,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

els still choose "Resp-2". Because it has more words overlap with context utterances, thus accumulating a larger similarity score. We can easily observe that "Resp-2" is relevant to former utterances (Turn-1 to Turn-6), but the topic has changed after "Turn-6". Besides, we can see that "Turn-3" and "Turn-4" do not provide any useful information for selecting candidate response. From this example, irrelevant context utterances may cause the models making simple mistakes that humans would not make. Furthermore, we conduct several adversarial experiments and the results show that these matching-based models are very sensitive to the adversarial samples.

In this paper, we propose a multi-hop selector network to tackle the above problem. Intuitively, the closer the utterance to the response is, the more it reflects the intention of the last dialogue session. Thus, we firstly use the last utterance as key to select context utterances that are relevant to it on the word and sentence level. However, we find that there are many samples whose last utterance is very short and contains very limited information (such as "good", "ok"), which will cause the selectors to lose too much useful context information. Therefore, we propose multi-hop selectors to select more relevant context utterances, yielding $k$ different context. Then, we fuse these selected context utterances and match it with candidate response. During the matching stage, the convolution neural network (CNN) is applied to extract matching features and the gated recurrent unit (GRU) is applied to learn the temporal relationship of utterances.

The contributions of this paper are summarized as follows:

- We find the noises in context utterances could influence the matching performance and design adversarial experiments to verify it.
- We propose a unified network MSN to select relevant context utterances from word and utterance level and fuse the selected context to generate a better context representation.
- Experimental results on three public datasets achieve significant improvement, which shows the effectiveness of MSN.

The outline of the paper is as follows. Section 2 introduces related works. Section 3 describes adversarial experiment to check how sensitivity of previous models to the context utterances. Section 4 describes every component of MSN model. Section 5 discusses the experiments and corresponding results. Section 6 discusses some experiments to explore the influence of hyper-parameters on performance. We conclude our work in Section 7.

## 2 Related Work

With the development of natural language processing, building intelligent chatbots with data-driven approaches has drawn increasing attention in recent years. Existing works can be generally categorized into retrieval-based methods (Wan et al., 2016; Wu et al., 2017; Zhang et al., 2018; Tao et al., 2019) and generation-based methods (Shang et al., 2015; Serban et al., 2016; Xing et al., 2017; Wu et al., 2018; Zhou et al., 2018a,b). In this work, we focus on retrieval-based method and study context-based response selection.

Early retrieval-based chatbots are devoted to response selection for single-turn conversation (Wang et al., 2013; Tan et al., 2015; Yan et al., 2016). Recently, researchers have begun to turn to the multi-turn conversation. Lowe et al. (2015) use RNN to read context and response, use the last hidden states to represent context and response as two semantic vectors to measure their relevance. Zhou et al. (2016) perform context-response matching with a multi-view model on both word and utterance levels. Considering concatenating utterances in context may lose relationships among utterances or important contextual information, Wu et al. (2017) separately match the response with each utterance based on a convolutional neural network. This paradigm is applied in many subsequent works. Zhou et al. (2018c) consider the dependency relation among utterances based on the attention mechanism. Tao et al. (2019) fuse words, n-grams, and sub-sequences of utterances representations and capture both short-term and long-term dependencies among words.

Different from previous works, (i) we study the influence of using excessive context utterances, (ii) we explore how to filter out irrelevant context to improve the robustness of matching-based methods.

## 3 Adversarial experiments

To study how sensitive of the previous models (Wu et al., 2017; Zhang et al., 2018; Zhou et al., 2018c; Tao et al., 2019) to the context utterances, we conduct several adversarial experiments inspired

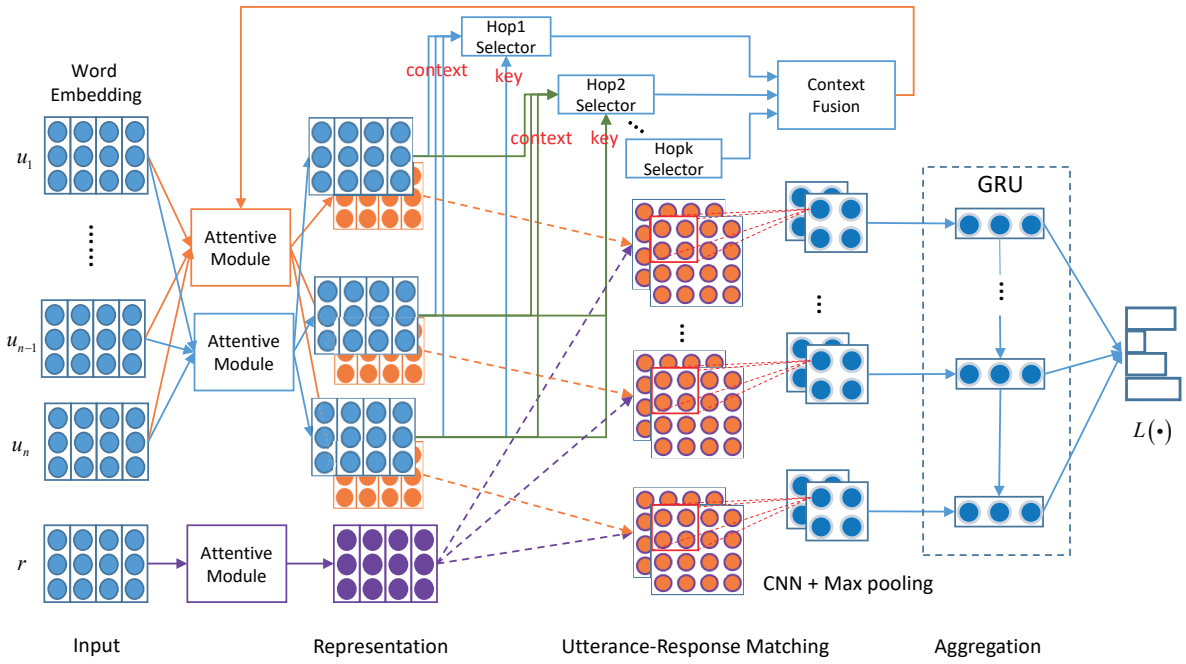


Figure 1: Architecture of multi-hop selector network.

by (Jia and Liang, 2017). We keep the training set unchanged and add some noises to the original test set. To be specific, we randomly sample 1∼3 words from context utterances and append them on every candidate response. In this way, we can obtain 3 different adversarial test sets: adversarial_set1, adversarial_set2, adversarial_set3.

Then, we evaluate the models again to see how much will the performance change. To ensure the fairness of the experiments, we use the results from their papers for the original test set. Moreover, we use their open source code for adversarial experiments. We employ recall at position k in n candidates ($R_n@k$) as the evaluation metric, which is the same as previous works.

Table 2: Adversarial experimental results on Ubuntu Dialogue Corpus. The results of SMN (Wu et al., 2017), DUA (Zhang et al., 2018), DAM (Zhou et al., 2018c), MFRN (Tao et al., 2019) on original test set are cited from their papers.

| Models | original test set | | adversarial_set1 | | adversarial_set2 | | adversarial_set3 | |
|---|---|---|---|---|---|---|---|---|
| | $R_{10}@1$ | $R_{10}@2$ | $R_{10}@1$ | $R_{10}@2$ | $R_{10}@1$ | $R_{10}@2$ | $R_{10}@1$ | $R_{10}@2$ |
| SMN | 72.6 | 84.7 | 66.2 | 82.1 | 63.8 | 79.4 | 57.1 | 75.0 |
| DUA | 75.2 | 86.8 | 64.0 | 80.4 | 58.0 | 75.6 | 52.7 | 70.8 |
| DAM | 76.7 | 87.4 | 67.5 | 82.3 | 61.2 | 76.8 | 54.3 | 71.6 |
| MRFN | 78.6 | 88.6 | 65.4 | 81.7 | 65.1 | 76.4 | 58.2 | 72.3 |
| MSN | 80.0 | 89.9 | 70.7 | 84.6 | 66.2 | 81.4 | 64.6 | 79.9 |

The experimental results are shown in Table 2. From the table, we can observe that the one-word noise will bring about 7% ∼ 13% absolute decrease on $R_{10}@1$ and the three-word noise brings about 20% $R_{10}@1$ decrease. Thus, we can see that matching-based models (Wu et al., 2017; Zhang et al., 2018; Zhou et al., 2018c; Tao et al., 2019) are very sensitive to small noises of the dataset. Moreover, using too many context utterances will greatly increase the probability of introducing noise. The results of MSN also show that filtering irrelevant utterances can effectively alleviate this problem and improve the robustness of matching-based models.

## 4 Model

### 4.1 Problem Formalization

Suppose that we have a data set $D = \{U_i, r_i, y_i\}_{i=1}^{N}$, where $U_i = \{u_{i1}, u_{i2}, \ldots, u_{iL}\}$ represents a conversation context with $L$ utterances and every utterance $u_{ij}$ contains $T$ words. $r_i$ is a response candidate and $y_i \in \{0, 1\}$ denotes a label. $y_i = 1$ means $r_i$ is a proper response for $U_i$, otherwise $y_i = 0$. Our goal is to learn a matching model $g(\cdot, \cdot)$ with $D$. For any context-response pair $(U_i, r_i)$, $g(U_i, r_i)$ measures the matching degree between $U_i$ and $r_i$.

To this end, we need to address two problems: (1) how to select proper context utterances from $U_i$; and (2) how to fuse these selected utterances together for a better representation.

### 4.2 Model Overview

We propose a multi-hop selector network (MSN) to model $g(\cdot, \cdot)$. Figure 1 gives the architecture, which generally follows the representation-matching-aggregation framework (Wu et al., 2017; Zhang et al., 2018; Zhou et al., 2018c; Tao et al., 2019) to match response with multi-turn context.

Different from previous works, we add a selection process before the above framework. MSN first constructs semantic representations at word level by an Attentive Module. Then, each utterance are packed as context or key and sent to the "Hopk Selector" to calculate relevance scores. The scores of $k$ different selectors are fused together by a Context Fusion module. Finally, the fused scores are performed over original context utterances to filter out irrelevant context. The rest context utterances are applied for response matching.

### 4.3 Attentive Module

We use the Attentive Module to learn the context information for word representation. Attentive Module is proposed in DAM (Zhou et al., 2018c) and it is a variant of Multi-head Attention (Vaswani et al., 2017). Figure 2 shows its structure.

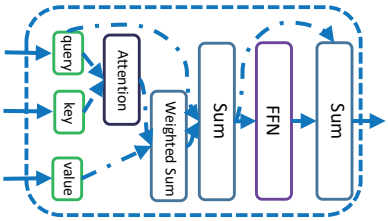


Figure 2: Architecture of Attentive Module.

The $\mathbf{AttentiveModule}(Q, K, V)$ has three input sentences: the query sentence, the key sentence and the value sentence, namely $Q \in \mathbb{R}^{n_q \times d}$, $K \in \mathbb{R}^{n_k \times d}$, and $V \in \mathbb{R}^{n_v \times d}$ respectively, where $n_q$, $n_k$, and $n_v$ denote the number of words in each sentence, and $d$ is the dimension of the embedding.

The Attentive Module first takes each word in the query sentence to attend to words in the key sentence via Scaled Dot-Product Attention (Vaswani et al., 2017), and then applies those attention weights upon the value sentence:

$$\mathbf{V}_{att} = softmax\left(\frac{QK^T}{\sqrt{d}}\right)V\,. \quad (1)$$

Then, a feed-forward network (FFN) with RELU (LeCun et al., 2015) activation is applied upon the normalization result, to further process the fused embeddings:

$$\mathbf{FFN}(x) = \max(0, x\mathbf{W}_1 + b_1)\mathbf{W}_2 + b_2\,, \quad (2)$$

where $x$ is a 2D matrix in the same shape of query sentence $Q$ and $\mathbf{W}_1$, $b_1$,$\mathbf{W}_2$, $b_2$ are learnt parameters. The result $\mathbf{FFN}(x)$ is a 2D matrix that has the same shape as $x$, $\mathbf{FFN}(x)$ is then residually added to $x$, and the fusion result is then normalized as the final outputs.

### 4.4 Context Selector

Given $\mathbf{U}_i = [\mathbf{u}_{i1}, \ldots, \mathbf{u}_{ij}, \ldots, \mathbf{u}_{iL}]$, the word-level embedding representations for utterance $\mathbf{u}_{ij} \in \mathbb{R}^{T \times d}$, where $d$ is the dimension of word vector, we use the Attentive Module to reconstruct the word representations of each utterance to encode the context and dependency information into word, which is formulated as:

$$\mathbf{u}'_{ij} = \mathbf{AttentiveModule}(\mathbf{u}_{ij}, \mathbf{u}_{ij}, \mathbf{u}_{ij})\,, \quad (3)$$

where $\mathbf{u}'_{ij} \in \mathbb{R}^{T \times d}$. $\mathbf{U}'_i = [\mathbf{u}'_{i1}, \mathbf{u}'_{i2}, \ldots, \mathbf{u}'_{iL}] \in \mathbb{R}^{L \times T \times d}$.

We first discuss how to construct "Hop1 Selector", which consists of word and utterance selector. To capture matching features at multiple levels of granularity, we leverage word and utterance level matching features to select relevant context.

#### 4.4.1 Word Selector

At word level, we utilize cross attention to obtain a matching feature map for each context utterance $\mathbf{u}'_{ij}$ and key $\mathbf{K}_1 = \mathbf{u}'_{iL}$, which is formulated as:

$$\mathbf{A} = \mathbf{v}^T \mathbf{tanh}(\mathbf{K}_1^T \mathbf{W} \mathbf{U}'_i + \mathbf{b})\,, \quad (4)$$

where $\mathbf{W} \in \mathbb{R}^{d \times d \times h}$, $\mathbf{b} \in \mathbb{R}^h$ and $\mathbf{v} \in \mathbb{R}^{h \times 1}$. And we get a word alignment matrix $\mathbf{A} \in \mathbb{R}^{L \times T \times T}$.

Then, we extract the most prominent matching features from $\mathbf{A}$ by max pooling over row and column. Then, they are concatenated together:

$$m_1(\mathbf{K}_1, \mathbf{U}'_i) = [\max_{dim=2} \mathbf{A}; \max_{dim=3} \mathbf{A}]\,, \quad (5)$$

where $m_1(\mathbf{K}_1, \mathbf{U}'_i) \in \mathbb{R}^{L \times 2T}$, which reflects which words have identical or similar meaning between utterances $\mathbf{u}_{ij}$ and key $\mathbf{u}_{iL}$. The matching features are transformed to the relevance score by a linear layer:

$$\mathbf{s}_1 = \mathbf{softmax}(m_1(\mathbf{K}_1, \mathbf{U}'_i)\mathbf{c} + b)\,, \quad (6)$$

where $\mathbf{c} \in \mathbb{R}^{2T\times 1}$ and $b \in \mathbb{R}^{L\times 1}$.

The word selector can only capture word-level relevance between key and utterances. It can not reflect whether key and context are compatible on the overall semantic level. Thus, we continue to evaluate the relevance on the utterance level.

#### 4.4.2 Utterance Selector

Firstly, the word-level representations $\mathbf{U}'_i$ are transformed to utterance-level representations by mean pooling over word dimension:

$$\widetilde{\mathbf{U}}_i = \mathbf{mean}(\mathbf{U}'_i)\,, \tag{7}$$

where $\widetilde{\mathbf{U}}_i \in \mathbb{R}^{L\times d}$.

We use cosine similarity to measure the relevance between key $\mathbf{K}_2 = \widetilde{\mathbf{U}}_{iL}$ and context utterances $\widetilde{\mathbf{U}}_i$, which is formulated as:

$$s_2 = \frac{\widetilde{\mathbf{U}}_i \mathbf{K}_2^T}{||\widetilde{\mathbf{U}}_i||_2 \, ||\mathbf{K}_2||_2}\,, \tag{8}$$

where $\mathbf{s}_2 \in \mathbb{R}^{L\times 1}$ is the relevance score at utterance level.

Both the scores of word selector and utterance selector are important to measure the relevance of last utterance and context. In order to make full use of word and utterance selectors, we design a combined strategy to fuse two scores. Specifically, we use the weighted sum of two scores for selection:

$$\mathbf{s}^{(1)} = \alpha * \mathbf{s}_1 + (1-\alpha) * \mathbf{s}_2\,, \tag{9}$$

where $\alpha$ is a hyper-parameter and $\mathbf{s}^{(1)}$ is the final score that hop1 selector produces. The default value of $\alpha$ is set to 0.5.

#### 4.4.3 Hopk Selector

Although "Hop1 Selector" can choose proper context utterances that are related to the last dialogue session, we find that there are many samples whose last utterance contains very little information (such as "good", "ok"), which will cause the selector lose too much useful context information. Thus, we combine it with $\widetilde{u}_{i,L-1}$, $\widetilde{u}_{i,L-2}$, ..., $\widetilde{u}_{i,L-k}$ by mean pooling. Then, we treat them as key to conduct the same process as "Hop1 Selector" for context selection. In this way, we can get $k$ different selectors, yielding $k$ different scores $\mathbf{S} = [\mathbf{s}^{(1)}, \mathbf{s}^{(2)}, \ldots, \mathbf{s}^{(k)}] \in \mathbb{R}^{L\times k}$.

### 4.5 Context Fusion

Then we fuse the similarity scores from different selectors and apply it to select relevant context utterances for matching.

Firstly, we combine the similarity scores $\mathbf{S} \in \mathbb{R}^{L\times k}$ to form the final scores for each context utterances and filter out irrelevant context, which is formulated as:

$$\begin{aligned} \mathbf{s}' &= \mathbf{S}\mathbf{W}^T\,, \\ \widetilde{\mathbf{s}} &= \mathbf{s}' \odot (\mathbf{sigmoid}(\mathbf{s}') \geq \gamma)\,, \end{aligned} \tag{10}$$

where $\mathbf{W} \in \mathbb{R}^{1\times k}$ is a dynamic weight vector and will be tuned by the gradient. $\gamma$ is the threshold and will be tuned according to the dataset. The default value of $\gamma$ can be set to 0.5. The utterances whose scores are below $\gamma$ will be allocated lower weights or filtered out.

Then, we multiply the mask weight $\widetilde{\mathbf{s}}$ and context utterances to filter irrelevant context:

$$\hat{\mathbf{U}}_i = \widetilde{\mathbf{s}} \odot \mathbf{U}_i\,, \tag{11}$$

and generate $\hat{\mathbf{U}}_i \in \mathbb{R}^{L\times T\times d}$, where $\mathbf{U}_i \in \mathbb{R}^{L\times T\times d}$ is the original utterances tensor.

### 4.6 Utterance-Response Matching

Similar to DAM (Zhou et al., 2018c), we utilize the self and cross matching paradigm to construct better matching feature maps.

#### 4.6.1 Origin Matching

Given the filtered utterances $\hat{\mathbf{U}}_i = [\hat{\mathbf{u}}_{i1}, \ldots, \hat{\mathbf{u}}_{ij}, \ldots, \hat{\mathbf{u}}_{iL}]$ and candidate response $\mathbf{r}_i \in \mathbb{R}^{T\times d}$, they are then used to construct a word-word similarity matrix $\mathbf{M_1} \in \mathbb{R}^{L\times 2\times T\times T}$ by dot product and cosine similarity. Both of them are stacked together as the channel dimension. The process can be formulated as:

$$\mathbf{M_1} = [\hat{\mathbf{U}}_i \mathbf{A_1} \mathbf{r}_i^T; \mathbf{cos}(\hat{\mathbf{U}}_i, \mathbf{r}_i)]\,. \tag{12}$$

where $\mathbf{A_1} \in \mathbb{R}^{d\times d}$ is a linear transformation matrix.

#### 4.6.2 Self Matching

Then, we use the Attentive Module over word dimension to construct multi-grained representations, which is formulated as:

$$\begin{aligned} \hat{\mathbf{U}}_i^{self} &= \mathbf{AttentiveModule}(\hat{\mathbf{U}}_i, \hat{\mathbf{U}}_i, \hat{\mathbf{U}}_i)\,, \\ \mathbf{r}_i^{self} &= \mathbf{AttentiveModule}(\mathbf{r}_i, \mathbf{r}_i, \mathbf{r}_i)\,. \end{aligned} \tag{13}$$

By this means, the words in each utterance or candidate response are connected together repeatedly to

combine more and more overall characterizations. Different from DAM (Zhou et al., 2018c), we do not stack many Attentive Module layers because it will drastically increase the computational expense.

Then, we use them to construct $\mathbf{M_2} \in \mathbb{R}^{L\times 2\times T\times T}$, whose element is

$$\mathbf{M}_2 = [\hat{\mathbf{U}}_i^{self}\mathbf{A_2}(\mathbf{r}_i^{self})^T; \mathbf{cos}(\hat{\mathbf{U}}_i^{self}, \mathbf{r}_i^{self})]\,, \tag{14}$$

where $\mathbf{A_2} \in \mathbb{R}^{d\times d}$ is a linear transformation matrix.

#### 4.6.3 Cross Matching

Similarly, we build the semantic association between every utterance and response by the attentive module:

$$\begin{aligned}\hat{\mathbf{U}}_i^{cross} &= \mathbf{AttentiveModule}(\hat{\mathbf{U}}_i, \mathbf{r}_i, \mathbf{r}_i)\,,\\ \mathbf{r}_i^{cross} &= \mathbf{AttentiveModule}(\mathbf{r}_i, \hat{\mathbf{U}}_i, \hat{\mathbf{U}}_i)\,.\end{aligned} \tag{15}$$

In this way, we can make the inter-dependent segment pairs close to each other, and aliment scores between those latently inter-dependent pairs could get increased, which will better encode the dependency relation into representation.

Finally, we use $\hat{\mathbf{U}}_i^{cross}$ and $\mathbf{r}_i^{cross}$ to construct $\mathbf{M}_3 \in \mathbb{R}^{L\times 2\times T\times T}$, whose element is

$$\mathbf{M}_3 = [\hat{\mathbf{U}}_i^{cross}\mathbf{A_3}(\mathbf{r}_i^{cross})^T; \mathbf{cos}(\hat{\mathbf{U}}_i^{cross}, \mathbf{r}_i^{cross})]\,, \tag{16}$$

where $\mathbf{A_3} \in \mathbb{R}^{d\times d}$ is a linear transformation.

### 4.7 Aggregation

MSN aggregates all the matching matrices together $\mathbf{M} = [\mathbf{M}_1; \mathbf{M}_2; \mathbf{M}_3] \in \mathbb{R}^{L\times 6\times T\times T}$ and applies 2D CNN and max pooling for matching feature extraction and use GRU to model the temporal relationship of utterances in the context, which is the same as SMN (Wu et al., 2017).

Then we compute matching score $g(U_i, r_i)$ based on the matching features. Specifically, we use the final state of GRU output $h_L$ as features and apply a single-layer perceptron to obtain score:

$$g(U_i, r_i) = \sigma(\mathbf{W}h_L + b)\,, \tag{17}$$

where $\mathbf{W}$ and $b$ are learnt parameters, $\sigma(\cdot)$ is sigmoid activation function.

Finally, the negative log-likelihood is used as a loss function to optimize the training process.

## 5 Experiment

### 5.1 Dataset

We test MSN on three widely used multi-turn response selection datasets, the Ubuntu Corpus (Lowe et al., 2015), the Douban Corpus (Wu et al., 2017) and the E-commerce Corpus (Zhang et al., 2018). Data statistics are in Table 3.

Table 3: Data statistics for Ubuntu, Douban and E-commerce datasets.

| Models | Ubuntu | | | Douban | | | E-commerce | | |
|---|---|---|---|---|---|---|---|---|---|
| | Train | Val | Test | Train | Val | Test | Train | Val | Test |
| #context-response pairs | 1M | 500K | 500K | 1M | 50K | 50K | 1M | 10K | 10K |
| #candidates per context | 2 | 10 | 10 | 2 | 2 | 10 | 2 | 2 | 10 |
| Avg #turns per context | 10.13 | 10.11 | 10.11 | 6.69 | 6.75 | 6.45 | 5.51 | 5.48 | 5.64 |
| Avg #words per utterance | 11.35 | 11.34 | 11.37 | 18.56 | 18.50 | 20.74 | 7.02 | 6.99 | 7.11 |

**Ubuntu Corpus** consists of English multi-turn conversations about technical support collected from chat logs of the Ubuntu forum.

**Douban Corpus** contains dyadic dialogs (conversation between two persons) longer than 2 turns from the Douban group [1] which is a popular social networking service in China.

**E-commerce Corpus** is collected from real-world conversations between customers and customer service staff from Taobao[2], the largest e-commerce platform in China. The dataset contains diverse types of conversations (e.g. commodity consultation, logistics express, recommendation, and chitchat) concerning various commodities.

### 5.2 Evaluation Metric

Following the previous works (Wu et al., 2017; Zhang et al., 2018; Chaudhuri et al., 2018; Tao et al., 2019), we employ recall at position k in n candidates ($R_n@k$) as evaluation metrics. Apart from $R_n@k$, we use MAP (Mean Average Precision), MRR (Mean Reciprocal Rank), and Precision-at-one P@1 especially for Douban corpus, which is the same as previous works (Wu et al., 2017; Tao et al., 2019). For some dialogues in Douban corpus have more than one true candidate response.

### 5.3 Baseline Models

**Single-turn matching models**: Basic models in (Lowe et al., 2015; Kadlec et al., 2015) including RNN, CNN are used in early works. Some advanced single-turn matching models, such as DL2R (Yan et al., 2016), Atten-LSTM (Tan et al.,

[1] https://www.douban.com/group
[2] https://www.taobao.com

Table 4: Experimental results on Ubuntu, Douban and E-commerce datasets. MRFN is the state-of-the-art model until this submission.

| Models | Ubuntu Corpus | | | Douban Corpus | | | | | | E-commerce Corpus | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $R_{10}$@1 | $R_{10}$@2 | $R_{10}$@5 | MAP | MRR | P@1 | $R_{10}$@1 | $R_{10}$@2 | $R_{10}$@5 | $R_{10}$@1 | $R_{10}$@2 | $R_{10}$@5 |
| TF-IDF (Lowe et al., 2015) | 41.0 | 54.5 | 70.8 | 33.1 | 35.9 | 18.0 | 9.6 | 17.2 | 40.5 | 15.9 | 25.6 | 47.7 |
| RNN (Lowe et al., 2015) | 40.3 | 54.7 | 81.9 | 39.0 | 42.2 | 20.8 | 11.8 | 22.3 | 58.9 | 32.5 | 46.3 | 77.5 |
| CNN (Kadlec et al., 2015) | 54.9 | 68.4 | 89.6 | 41.7 | 44.0 | 22.6 | 12.1 | 25.2 | 64.7 | 32.8 | 51.5 | 79.2 |
| LSTM (Kadlec et al., 2015) | 63.8 | 78.4 | 94.9 | 48.5 | 53.7 | 32.0 | 18.7 | 34.3 | 72.0 | 36.5 | 53.6 | 82.8 |
| BiLSTM (Kadlec et al., 2015) | 63.0 | 78.0 | 94.4 | 47.9 | 51.4 | 31.3 | 18.4 | 33.0 | 71.6 | 35.5 | 52.5 | 82.5 |
| DL2R (Yan et al., 2016) | 62.6 | 78.3 | 94.4 | 48.8 | 52.7 | 33.0 | 19.3 | 34.2 | 70.5 | 39.9 | 57.1 | 84.2 |
| Atten-LSTM (Tan et al., 2015) | 63.3 | 78.9 | 94.3 | 49.5 | 52.3 | 33.1 | 19.2 | 32.8 | 71.8 | 40.1 | 58.1 | 84.9 |
| MV-LSTM (Wan et al., 2016) | 65.3 | 80.4 | 94.6 | 49.8 | 53.8 | 34.8 | 20.2 | 35.1 | 71.0 | 41.2 | 59.1 | 85.7 |
| Match-LSTM (Wang and Jiang, 2016) | 65.3 | 79.9 | 94.4 | 50.0 | 53.7 | 34.5 | 20.2 | 34.8 | 72.0 | 41.0 | 59.0 | 85.8 |
| Multi-View (Zhou et al., 2016) | 66.2 | 80.1 | 95.1 | 50.5 | 54.3 | 34.2 | 20.2 | 35.0 | 72.9 | 42.1 | 60.1 | 86.1 |
| SMN (Wu et al., 2017) | 72.6 | 84.7 | 96.1 | 52.9 | 56.9 | 39.7 | 23.3 | 39.6 | 72.4 | 45.3 | 65.4 | 88.6 |
| DUA (Zhang et al., 2018) | 75.2 | 86.8 | 96.2 | 55.1 | 59.9 | 42.1 | 24.3 | 42.1 | 78.0 | 50.1 | 70.0 | 92.1 |
| DAM (Zhou et al., 2018c) | 76.7 | 87.4 | 96.9 | 55.0 | 60.1 | 42.7 | 25.4 | 41.0 | 75.7 | - | - | - |
| MRFN (Tao et al., 2019) | 78.6 | 88.6 | 97.6 | 57.1 | 61.7 | 44.8 | 27.6 | 43.5 | 78.3 | - | - | - |
| MSN | **80.0** | **89.9** | **97.8** | **58.7** | **63.2** | **47.0** | **29.5** | **45.2** | **78.8** | **60.6** | **77.0** | **93.7** |

2015), and MV-LSTM (Wan et al., 2016) are also explored in this work. These models concatenate all context utterances together to match a response.

**Multi-turn matching models**: Multi-view (Zhou et al., 2016) models utterances from word level view and utterance level view; DL2R model (Yan et al., 2016) reformulates the message with other utterances in the context; SMN (Wu et al., 2017) matches a response with each utterance in the context; DUA (Zhang et al., 2018) formulates previous utterances into context using a proposed deep utterance aggregation model; DAM (Zhou et al., 2018c) constructs representations of utterances in the context and the response with stacked self-attention and cross attention; MRFN (Tao et al., 2019) fuses multiple types of representations with a multi-representation fusion network for response matching.

### 5.4 Model Training

Our model was implemented by PyTorch (Paszke et al., 2017). Word embeddings were initialized by the results of word2vec (Mikolov et al., 2013) which ran on the dataset, and the dimensionality of word vectors is 200. The hyper-parameter $k$ of selectors is set to 3. We use three convolution layers to extract matching features. The 1st convolution layer has 16 [3,3] filters with [1,1] stride, and its max-pooling size is [2,2] with [2,2] stride. The 2nd convolution layer has 32 [3,3] filters with [1,1] stride, and its max pooling size is also [2,2] with [2,2] stride. The 3rd convolution layer has 64 [3,3] filters with [1,1] stride, and its max pooling size is also [3,3] with [3,3] stride. We set the dimension of the hidden states of GRU as 300. The parameters were updated by Adam algorithm (Kingma and Ba, 2014) and the parameters of Adam, $\beta_1$ and $\beta_2$ are 0.9 and 0.999 respectively. The learning rate is initialized as 1e-3 and gradually decreased during training. Same as previous works (Wu et al., 2017; Zhang et al., 2018), the maximum utterance length is 50 and the maximum context length (i.e., number of utterances) as 10.

### 5.5 Experiment Result

Table 4 shows the results of MSN and all baseline models on the datasets. All the experimental results are cited from previous works (Zhang et al., 2018; Chaudhuri et al., 2018; Tao et al., 2019).

Referring to the table, MSN significantly outperforms all other models in terms of most of the metrics on the three datasets, including MRFN, which is the state-of-the-art model until this submission. MSN extends from SMN (Wu et al., 2017) and DAM (Zhou et al., 2018c), and it achieves more than 3% absolute improvement on $R_{10}$@1 compared with SMN and DAM. The improvement also shows the importance of filtering irrelevant context before matching.

## 6 Further Analysis

### 6.1 Ablation Study

We perform a series of ablation experiments over the different parts of the model to investigate their relative importance. Firstly, we use the complete

MSN as the baseline. Then, we gradually remove its modules as follows:

- **w/o Word Selector**: A model that is trained using the utterance selector but without the word selector.
- **w/o Utterance Selector**: A model which is trained without the utterance selector.
- **Only Hop1 (Hop2, Hop3) Selector**: A model which is trained only with hop1 or hop2 or hop3 selector.
- **w/o Selector**: Removing all selector modules and only use the attention module for matching.

Table 5: Ablation study on E-commerce corpus.

| Model | $R_{10}@1$ | $R_{10}@2$ | $R_{10}@5$ |
|---|---|---|---|
| **MSN**$_{base}$ | 60.6 | 77.0 | 93.7 |
| **w/o Selector** | 55.4 | 74.2 | 92.5 |
| **w/o Word Selector** | 59.3 | 76.5 | 92.4 |
| **w/o Utterance Selector** | 58.6 | 75.3 | 92.8 |
| **Only Hop1 Selector** | 58.3 | 74.9 | 93.3 |
| **Only Hop2 Selector** | 56.8 | 76.7 | 94.6 |
| **Only Hop3 Selector** | 56.6 | 74.7 | 94.0 |

From experimental results in Table 5, we can observe that:

(1) Compared with MSN$_{base}$, removing selectors leads to performance degradation, which shows that the multi-hop selectors are indeed help to improve the selection performance.

(2) The performances decay a large margin when the word selector and utterance selector are removed, which proves that both word selector and utterance selector play an important role in selecting relevant context utterances.

(3) For E-commerce dataset, the context selected by Hop1 selector is more important than other selectors. We think the main reason is that the dialogs in E-commerce corpus happen between buyers and sellers on the Taobao platform. The intent of the dialogue is very clear and the dialogue is mainly in the form of one question and one answer. So the last dialogue session has little dependency on the very far context. However, the fusion of these hop selectors' results still brings more performance improvement.

### 6.2 Parameter Sensitivity

The choices of $k$ for selectors and threshold $\gamma$ in formula (10) may influence the performance. Thus, we conduct a series of sensitivity analysis experiments on the development dataset to study how different choices of parameters influence the performance of the model.

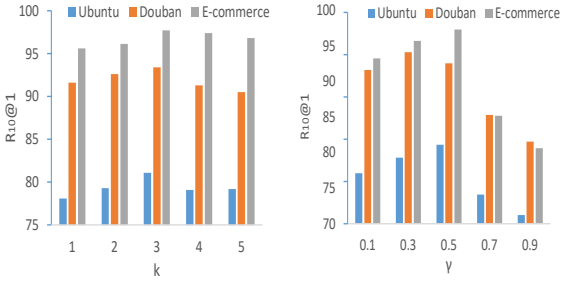


(a) Effects of k. (b) Effects of threshold $\gamma$.

Figure 3: Parameter sensitivity analysis on the development datasets of Ubuntu, Douban, and E-commerce Corpus.

The $k$ decides how many selectors that MSN uses to select relevant context utterances. Referring to Figure 3 (a), only using hop1 selector is not better than using multiple selectors. However, the performance does not increase when $k > 3$. It is easy to see that when $k$ is too large, the key will contain too many noises and cannot reflect the intention of the last dialogue session.

Figure 3 (b) shows the performance with different threshold $\gamma$. Intuitively, when $\gamma$ is too large, the selectors will filter out too much context, which may hurt performance. However, when $\gamma$ is too small, the selectors do not work very well. We can observe that MSN achieves the best performance when $\gamma = 0.3$ or $0.5$.

## 7 Conclusion and Future Work

In this paper, we analyze the side effect of using unnecessary context utterances and verify matching-based models are very sensitive to the context. We propose a multi-hop selector network to alleviate this problem. Empirical results on three large-scale datasets demonstrate the effectiveness of the model in multi-turn response selection and yield new state-of-the-art results at the same time.

In the future, we will study how to solve the logical consistency problem between utterances and candidate responses to improve selection performance.

## 8 Acknowledgement

We gratefully thank the anonymous reviewers for their insightful comments. This research is supported in part by the Beijing Municipal Science and Technology Project under Grant Z191100007119008 and Z181100002718004, the National Key Research and Development Program of China under Grant 2018YFC0806900 and 2017YFB1010000.

## References

Debanjan Chaudhuri, Agustinus Kristiadi, Jens Lehmann, and Asja Fischer. 2018. Improving response selection in multi-turn dialogue systems by incorporating domain knowledge. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 497–507.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Diana Inkpen, and Si Wei. 2018. Neural natural language inference models enhanced with external knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2406–2417.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031.

Rudolf Kadlec, Martin Schmid, and Jan Kleindienst. 2015. Improved deep learning baselines for ubuntu corpus dialogs. *arXiv preprint arXiv:1510.03753*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature*, 521(7553):436.

Feng-Lin Li, Minghui Qiu, Haiqing Chen, Xiongwei Wang, Xing Gao, Jun Huang, Juwei Ren, Zhongzhou Zhao, Weipeng Zhao, Lei Wang, et al. 2017. Alime assist: an intelligent assistant for creating an innovative e-commerce experience. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 2495–2498. ACM.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 285–294.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Natural language inference by tree-based convolution and heuristic matching. In *The 54th Annual Meeting of the Association for Computational Linguistics*, page 130.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NIPS-W*.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1577–1586.

Heung-Yeung Shum, Xiao-dong He, and Di Li. 2018. From eliza to xiaoice: challenges and opportunities with social chatbots. *Frontiers of Information Technology & Electronic Engineering*, 19(1):10–26.

Ming Tan, Cicero Dos Santos, Bing Xiang, and Bowen Zhou. 2015. Lstm-based deep learning models for nonfactoid answer selection. In *Proceedings of the International Conference on Learning Representations*.

Chongyang Tao, Wei Wu, Can Xu, Wenpeng Hu, Dongyan Zhao, and Rui Yan. 2019. Multi-representation fusion network for multi-turn response selection in retrieval-based chatbots. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 267–275. ACM.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Shengxian Wan, Yanyan Lan, Jun Xu, Jiafeng Guo, Liang Pang, and Xueqi Cheng. 2016. Match-srnn: modeling the recursive matching structure with spatial rnn. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 2922–2928. AAAI Press.

Hao Wang, Zhengdong Lu, Hang Li, and Enhong Chen. 2013. A dataset for research on short-text conversations. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 935–945.

Shuohang Wang and Jing Jiang. 2016. Learning natural language inference with lstm. In *Proceedings of NAACL-HLT*, pages 1442–1451.

Yu Wu, Wei Wu, Chen Xing, Ming Zhou, and Zhoujun Li. 2017. Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 496–505.

Yu Wu, Wei Wu, Dejian Yang, Can Xu, and Zhoujun Li. 2018. Neural response generation with dynamic vocabularies. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2017. Topic aware neural response generation. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Rui Yan, Yiping Song, and Hua Wu. 2016. Learning to respond with deep neural networks for retrieval-based human-computer conversation system. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 55–64. ACM.

Zhuosheng Zhang, Jiangtong Li, Pengfei Zhu, Hai Zhao, and Gongshen Liu. 2018. Modeling multi-turn conversation with deep utterance aggregation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3740–3752.

Hao Zhou, Minlie Huang, Tianyang Zhang, Xiaoyan Zhu, and Bing Liu. 2018a. Emotional chatting machine: Emotional conversation generation with internal and external memory. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Hao Zhou, Tom Young, Minlie Huang, Haizhou Zhao, Jingfang Xu, and Xiaoyan Zhu. 2018b. Commonsense knowledge aware conversation generation with graph attention. In *IJCAI*, pages 4623–4629.

Xiangyang Zhou, Daxiang Dong, Hua Wu, Shiqi Zhao, Dianhai Yu, Hao Tian, Xuan Liu, and Rui Yan. 2016. Multi-view response selection for human-computer conversation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 372–381.

Xiangyang Zhou, Lu Li, Daxiang Dong, Yi Liu, Ying Chen, Wayne Xin Zhao, Dianhai Yu, and Hua Wu. 2018c. Multi-turn response selection for chatbots with deep attention matching network. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1118–1127.

# MoEL: Mixture of Empathetic Listeners

**Zhaojiang Lin, Andrea Madotto, Jamin Shin, Peng Xu, Pascale Fung**
Center for Artificial Intelligence Research (CAiRE)
Department of Electronic and Computer Engineering
The Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong
{zlinao,amadotto,jmshinaa,pxuab}@connect.ust.hk,
pascale@ece.ust.hk

## Abstract

Previous research on empathetic dialogue systems has mostly focused on generating responses given certain emotions. However, being empathetic not only requires the ability of generating emotional responses, but more importantly, requires the understanding of user emotions and replying appropriately. In this paper, we propose a novel end-to-end approach for modeling empathy in dialogue systems: Mixture of Empathetic Listeners (MoEL). Our model first captures the user emotions and outputs an emotion distribution. Based on this, MoEL will *softly combine* the output states of the *appropriate* Listener(s), which are each optimized to react to certain emotions, and generate an empathetic response. Human evaluations on *empathetic-dialogues* (Rashkin et al., 2018) dataset confirm that MoEL outperforms multitask training baseline in terms of empathy, relevance, and fluency. Furthermore, the case study on generated responses of different Listeners shows high interpretability of our model.

## 1 Introduction

Neural network approaches for conversation models have shown to be successful in scalable training and generating fluent and relevant responses (Vinyals and Le, 2015). However, it has been pointed out by Li et al. (2016a,b,c); Wu et al. (2018b) that only using Maximum Likelihood Estimation as the objective function tends to lead to *generic* and *repetitive* responses like "I am sorry". Furthermore, many others have shown that the incorporation of additional inductive bias leads to a more engaging chatbot, such as understanding commonsense (Dinan et al., 2018), or modeling consistent persona (Li et al., 2016b; Zhang et al., 2018a; Mazare et al., 2018a).

Meanwhile, another important aspect of an engaging human conversation that received relatively less focus is emotional understanding and empathy (Rashkin et al., 2018; Dinan et al., 2019; Wolf et al., 2019). Intuitively, ordinary social conversations between two humans are often about their daily lives that revolve around happy or sad experiences. In such scenarios, people generally tend to respond in a way that acknowledges the feelings of their conversational partners.

| **Emotion: Angry** | |
|---|---|
| **Situation** | I was furious when I got in my first car wreck. |
| **Speaker** | I was driving on the interstate and another car ran into the back of me. |
| **Listener** | Wow. Did you get hurt? Sounds scary. |
| **Speaker** | No just the airbags went off and I hit my head and got a few bruises. |
| **Listener** | I am always scared about those airbags! I am so glad you are ok! |

Table 1: One conversation from empathetic dialogue, a speaker tells the situation he(she) is facing, and a listener try to understand speaker's feeling and respond accordingly

Table 1 shows an conversation from the *empathetic-dialogues* dataset (Rashkin et al., 2018) about how an empathetic person would respond to the stressful situation the *Speaker* has been through. However, despite the importance of empathy and emotional understanding in human conversations, it is still very challenging to train a dialogue agent able to recognize and respond with the correct emotion.

So far, to solve the problem of empathetic dialogue response generation, which is to understand the user emotion and respond appropriately (Bertero et al., 2016), there have been mainly two lines



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 121–132,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

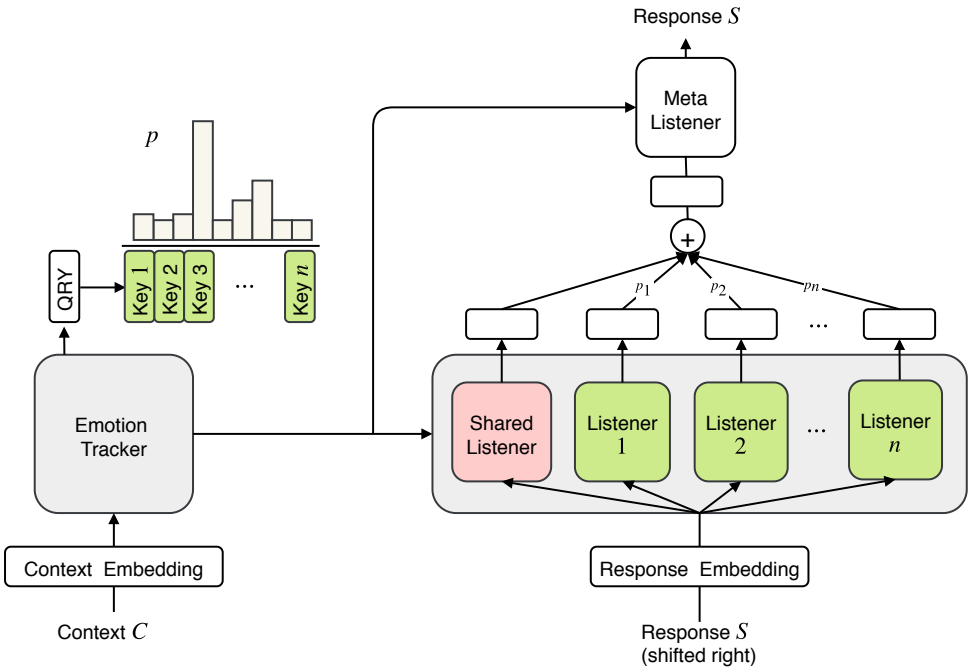


Figure 1: The proposed model Mixture of Empathetic Listeners, which has an emotion tracker, $n$ empathetic listeners along with a shared listener, and a meta listener to fuse the information from listeners and produce the empathetic response.

of work. The first is a multi-task approach that jointly trains a model to predict the current emotional state of the user and generate an appropriate response based on the state (Lubis et al., 2018; Rashkin et al., 2018). Instead, the second line of work focuses on conditioning the response generation to a certain fixed emotion (Hu et al., 2017; Wang and Wan, 2018; Zhou and Wang, 2018; Zhou et al., 2018).

Both cases have succeeded in generating empathetic and emotional responses, but have neglected some crucial points in empathetic dialogue response generation. 1) The first assumes that by understanding the emotion, the model implicitly learns how to respond appropriately. However, without any additional inductive bias, a single decoder learning to respond for all emotions will not only lose interpretability in the generation process, but will also promote more generic responses. 2) The second assumes that the emotion to condition the generation on is given as input, but we often do not know which emotion is appropriate in order to generate an empathetic response.

Therefore, in this paper, to address the above issues, we propose a novel end-to-end empathetic dialogue agent, called Mixture of Empathetic Listeners [1] (MoEL). Similar to Rashkin et al. (2018), we first encode the dialogue context and use it to recognize the emotional state ($n$ possible emotions). However, the main difference is that our model consists of $n$ decoders, further denoted as *listeners*, which are optimized to react to each context emotion accordingly. The listeners are trained along with a Meta-listener that *softly combines* the output decoder states of each listener according to the emotion classification distribution. Such design allows our model to explicitly learn how to choose an appropriate reaction based on its understanding of the context emotion. A detailed illustration of MoEL is shown in Figure 1.

The proposed model is tested against several competitive baseline settings (Vaswani et al., 2017; Rashkin et al., 2018), and evaluated with human judges. The experimental results show that our approach outperforms the baselines in both empathy and relevance. Finally, our analysis demonstrates that not only MoEL effectively attends to the right listener, but also each listener learns how to properly react to its corresponding emotion, hence allowing a more interpretable generative process.

[1] The code will be released at https://github.com/HLTCHKUST/MoEL

## 2 Related Work

**Conversational Models:** Open domain conversational models has been widely studied (Serban et al., 2016; Vinyals and Le, 2015; Wolf et al., 2019). A recent trend is to produce personalized responses by conditioning the generation on a persona profile to make the response more consistent through the dialogue (Li et al., 2016b). In particular, PersonaChat (Zhang et al., 2018b; Kulikov et al., 2018) dataset was created, and then extended in ConvAI 2 challenge (Dinan et al., 2019), to show that by adding persona information as input to the model, the produced responses elicit more consistent personas. Based on such, several follow-up work has been presented (Mazare et al., 2018b; Hancock et al., 2019; Joshi et al., 2017; Kulikov et al., 2018; Yavuz et al., 2018; Zemlyanskiy and Sha, 2018; Madotto et al., 2019). However, such personalized dialogue agents focus only on modeling a consistent persona and often neglect the feelings of their conversation partners.

Another line of work combines retrieval and generation to promote the response diversity (Cai et al., 2018; Weston et al., 2018; Wu et al., 2018b). However, only fewer works focus on emotion (Winata et al., 2017, 2019; Xu et al., 2018; Fan et al., 2018a,c,b; Lee et al., 2019) and empathy in the context of dialogues systems (Bertero et al., 2016; Chatterjee et al., 2019a,b; Shin et al., 2019). For generating emotional dialogues, Hu et al. (2017); Wang and Wan (2018); Zhou and Wang (2018) successfully introduce a framework of controlling the sentiment and emotion of the generated response, while (Zhou and Wang, 2018) also introduces a new Twitter conversation dataset and propose to distantly supervised the generative model with emojis. Meanwhile, (Lubis et al., 2018; Rashkin et al., 2018) also introduce new datasets for empathetic dialogues and train multi-task models on it.

**Mixture of Experts:** The idea of having specialized parameters, or so-called experts, has been widely studied topics in the last two decades (Jacobs et al., 1991; Jordan and Jacobs, 1994). For instance, different architectures and methodologies have been used such as SVM (Collobert et al., 2002), Gaussian Processes (Tresp, 2001; Theis and Bethge, 2015; Deisenroth and Ng, 2015), Dirichlet Processes (Shahbaba and Neal, 2009), Hierarchical Experts (Yao et al., 2009), Infinite Number of Experts (Rasmussen and Ghahramani, 2002) and sequential expert addition (Aljundi et al., 2017). More recently, the Mixture Of Expert (Shazeer et al., 2017; Kaiser et al., 2017) model was proposed which added a large number of experts in between of two LSTM (Schmidhuber, 1987) layers to enhance the capacity of the model. This idea of having independent specialized experts inspires our approach to model the reaction to each emotion with a separate expert.

## 3 Mixture of Empathetic Listeners

The dialogue context is an alternating set of utterances from speaker and listener. We denote the dialogue context as $C = \{U_1, S_1, U_2, S_2, \cdots, U_t\}$ and the speaker emotion state at each utterance as $Emo = \{e_1, e_2, \cdots, e_t\}$ where $\forall e_i \in \{1, \ldots, n\}$. Then, our model aims to track the speaker emotional state $e_t$ from the dialogue context $C$, and generates an empathetic response $S_t$.

Overall, MoEL is composed of three components: an *emotion tracker*, *emotion-aware listeners*, and a *meta listener* as shown in Figure 1. The emotion tracker (which is also the context encoder) encodes $C$ and computes a distribution over the possible user emotions. Then all the listeners independently attend to this distribution to compute their own representation. Finally, the meta listener takes the weighted sum of representations from the listeners and generates the final response.

### 3.1 Embedding

We define the context embedding $E^C \in \mathbb{R}^{|V| \times d_{emb}}$, and the response embedding $E^R \in \mathbb{R}^{|V| \times d_{emb}}$ which are used to convert tokens into embeddings. In multi-turn dialogues, ensuring that the model is able to distinguish among turns is essential, especially when multiple emotion are present in different turns. Hence, we incorporate a dialogue state embedding in the input. This is used to enable the encoder to distinguish speaker utterances and listener utterances (Wolf et al., 2019). As shown in Figure 2, our context embedding $E^C$ is the positional sum of the word embedding $E^W$, the positional embedding $E^P$ (Vaswani et al., 2017) and the dialogue state embedding $E^D$.

$$E^C(C) = E^W(C) + E^P(C) + E^D(C) \quad (1)$$

### 3.2 Emotion Tracker

MoEL uses a standard transformer encoder (Vaswani et al., 2017) for the emotion

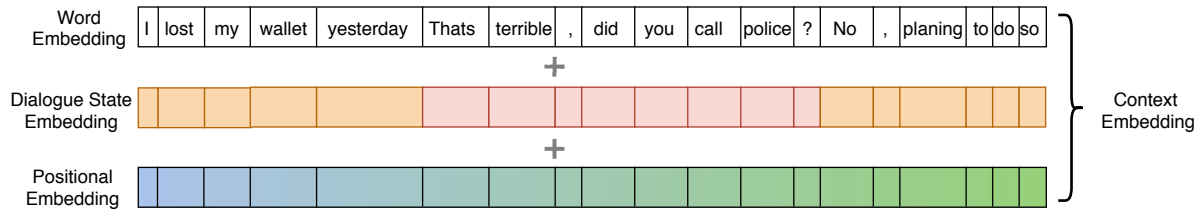


Figure 2: Context embedding is computed by summing up the word embedding, dialogue state embedding and positional embedding for each token.

tracker. We first flatten all dialogue turns in $C$, and map each token into its vectorial representation using the context embedding $E^C$. Then the encoder encodes the context sequence into a context representation. We add a query token $QRY$ at the beginning of each input sequence as in BERT (Devlin et al., 2018), to compute the weighted sum of the output tensor. Denoting a transformer encoder as $TRS_{Enc}$, then corresponding context representation become:

$$H = TRS_{Enc}(E^C([QRY; C])) \quad (2)$$

where $[;]$ denotes concatenation, $H \in \mathbb{R}^{L \times d_{model}}$ where $L$ is the sequence length. Then, we define the final representation of the token $QRY$ as

$$q = H_0 \quad (3)$$

where $q \in \mathbb{R}^{d_{model}}$, which is then used as the query for generating the emotion distribution.

### 3.3 Emotion Aware Listeners

The emotion aware listeners mainly consist of 1) a *shared listener* that learns shared information for all emotions and 2) $n$ independently parameterized Transformer decoders (Vaswani et al., 2017) that learn how to appropriately react given a particular emotional state. All the listeners are modeled by a standard transformer decoder layer block, denoted as $TRS_{Dec}$, which is made of three sub-components: a multi-head self-attention over the response input embedding, a multi-head attention over the output of the emotion tracker, and a position-wise fully connected feed-forward network.

Thus, we define the set of listeners as $L = [TRS^0_{Dec}, \ldots, TRS^n_{Dec}]$. Given the target sequence shifted by one $r_{0:t-1}$, each listener compute its own emotional response representation $V_i$:

$$V_i = TRS^i_{Dec}(H, E^R(r_{0:t-1})) \quad (4)$$

where $TRS^i_{Dec}$ refers to the $i$-th listener, including the shared one. Conceptually, we expect that the output from the shared listener, $TRS^0_{Dec}$, to be a general representation which can help the model to capture the dialogue context. On the other hand, we expect that each empathetic listener learns how to respond to a particular emotion. To model this behavior, we assign different weights to each empathetic listener according to the user emotion distribution, while assigning a fixed weight of 1 to the shared listener.

To elaborate, we construct a Key-Value Memory Network (Miller et al., 2016) and represent each memory slot as a vector pair $(k_i, V_i)$, where $k_i \in \mathbb{R}^{d_{model}}$ denotes the key vector and $V_i$ is from Equation 4. Then, the encoder informed query $q$ is used to address the key vectors $k$ by performing a dot product followed by a Softmax function. Thus, we have:

$$p_i = \frac{e^{q^\top k_i}}{\sum_{j=1}^{n} e^{q^\top k_j}} \quad (5)$$

each $p_i$ is the score assigned to $V_i$, thus used as the weight of each listener. During training, given the speaker emotion state $e_t$, we supervise each weight $p_i$ by maximizing the probability of the emotion state $e_t$ with a cross entropy loss function:

$$\mathcal{L}_1 = -\log p_{e_t} \quad (6)$$

Finally, the combined output representation is compute by the weighted sum of the memory values $V_i$ and the shared listener output $V_0$.

$$V_M = V_0 + \sum_{i=1}^{n} p_i V_i \quad (7)$$

### 3.4 Meta Listener

Finally, the Meta Listener is implemented using another transformer decoder layer, which further transform the representation of the listeners and

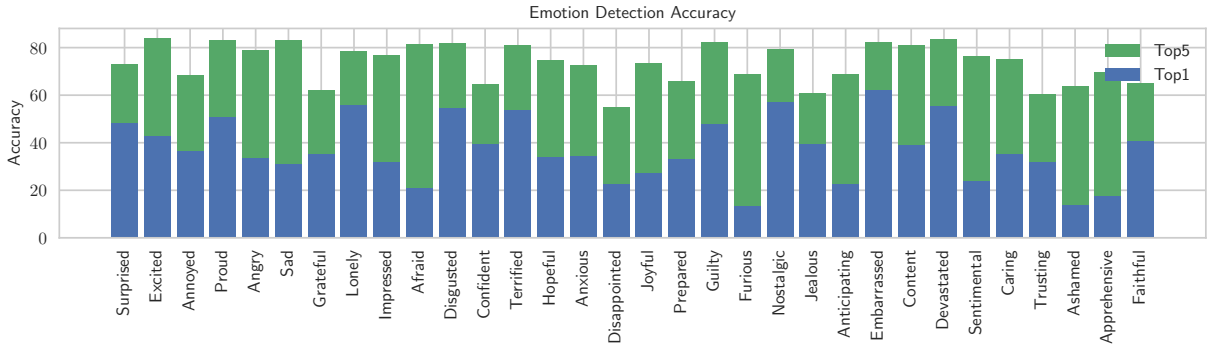


Figure 3: Top-1 and Top-5 emotion detection accuracy over 32 emotions at each turn

| | Params. | BLEU | Empathy | Relevance | Fluency |
|---|---|---|---|---|---|
| *Gold* | - | - | 3.93 | 3.93 | 3.35 |
| *TRS* | 16.94M | 3.02 | 3.32 | 3.47 | **3.52** |
| *MultiTRS* | 16.95M | 2.92 | 3.36 | 3.57 | 3.31 |
| *MoEL* | 23.1M | 2.90 | **3.44** | **3.70** | 3.47 |

Table 2: Comparison between our proposed methods and baselines. All of models receive close BLEU score. MoEL achieve highest *Empathy* and *Relevance* score, while TRS achieve better *Fluency* score. The number of parameters for each model is reported.

generates the final response. The intuition is that each listener specializes to a certain emotion and the Meta Listener gathers the opinions generated by multiple listeners to produce the final response. Hence, we define another $TRS^{Meta}_{Dec}$, and an affine transformation $W \in \mathbb{R}^{d_{model} \times |V|}$ to compute:

$$O = TRS^{Meta}_{Dec}(H, V_M) \tag{8}$$

$$p(r_{1:t}|C, r_{0:t-1}) = \text{softmax}(O^\top W) \tag{9}$$

where $O \in \mathbb{R}^{d_{model} \times t}$ is the output of meta listener and $p(r_{1:t}|C, r_{0:t-1})$ is a distribution over the vocabulary for the next tokens. We then use a standard maximum likelihood estimator (MLE) to optimize the response prediction:

$$\mathcal{L}_2 = -\log p\left(S_t|C\right) \tag{10}$$

Lastly, all the parameters are jointly trained end-to-end to optimize the listener selection and response generation by minimizing the weighted-sum of two losses:

$$\mathcal{L} = \alpha\mathcal{L}_1 + \beta\mathcal{L}_2 \tag{11}$$

Where $\alpha$ and $\beta$ are hyperparameters to balance two loss.

| Model | Win | Loss | Tie |
|---|---|---|---|
| *MoEL vs TRS* | 37.3% | 18.7% | 44% |
| *MoEL vs Multi-TRS* | 36.7% | 32.6% | 30.7% |

Table 3: Result of human A/B test. Tests are conducted pairwise between MoEL and baseline models

## 4 Experiment

### 4.1 Dataset

We conduct our experiment on the *empathetic-dialogues* (Rashkin et al., 2018) dataset which consist of 25k one-to-one open-domain conversation grounded in emotional situations. The dataset provides 32, evenly distributed, emotion labels. Table 1 shows an example from the training set. The speakers are talking about their situation and the listeners is trying to understand their feeling and reply accordingly. At training time the emotional labels of the speakers are given, while we hide the label in test time to evaluate the *empathy* of our model.

### 4.2 Training

We train our model using Adam optimizer (Kingma and Ba, 2014) and varied the learning rate during training following (Vaswani et al., 2017). The weight of both losses $\alpha$ and $\beta$ are set to 1 for simplicity. We use pre-trained Glove vectors (Pennington et al., 2014) to initialize the word embedding and we share it across the encoder and the decoder. The rest of the parameters are randomly initialized.

In the early training stage, emotion tracker randomly assign weights to the listeners, and may send noisy gradient flow back to the wrong listeners, which can make the model convergence harder. To stabilize the learning process, we replace the distribution $p$ of the listeners with the or-

acle emotion $e_t$ information using a certain probability $\epsilon_{oracle}$, and we gradually anneal it during the training. We set an annealing rate $\gamma = 1 \times 10^{-3}$, and a threshold $t_{thd}$ equal to $1 \times 10^4$, thus at each iteration $t$ iteration we compute:

$$\epsilon_{oracle} = \gamma + (1 - \gamma)e^{-\frac{t}{t_{thd}}} \quad (12)$$

### 4.3 Baseline

We compare our model with two baselines:

**Transformer (TRS)** The standard Transformer model (Vaswani et al., 2017) that is trained to minimize MLE loss as in Equation 10.

**Multitask Transformer (Multi-TRS)** A Multitask Transformer trained as (Rashkin et al., 2018) to incorporate additional supervised information about the emotion. The encoder of multitask transformer is the same as our emotion tracker, and the context representation $Q$, from Equation 3, is used as input to an emotion classifier. The whole model is jointy trained by optimizing both the classification and generation loss.

### 4.4 Hyperparameter

In all of our experiments we used 300 dimensional word embedding and 300 hidden size everywhere. We use 2 self-attention layers made up of 2 attention heads each with embedding dimension 40. We replace Positionwise Feedforward sub-layer with 1D convolution with 50 filters of width 3. We train all of models with batch size 16 and we use batch size 1 in the test time.

### 4.5 Evaluation Metrics

**BLEU** We compute BLEU scores (Papineni et al., 2002) to compare the generated response against human responses. However, in open-domain dialogue response generation, BLEU is not a good measurement of generation quality (Liu et al., 2016), so we use BLEU only as a reference.

**Human Ratings** In order to measure the quality of the generated responses, we conduct human evaluations with Amazon Mechanical Turk. Following Rashkin et al. (2018), we first randomly sample 100 dialogues and their corresponding generations from MoEL and the baselines. For each response, we assign three human annotators to score the following aspect of models: *Empathy*, *Relevance*, and *Fluency*. Note that we evaluate each metric independently and the scores range between 1 and 5, in which 1 is "not at all" and 5 is "very much".

We ask the human judges to evaluate each of the following categories from a 1 to 5 scale, where 5 is the best score.

- Empathy / Sympathy: Did the responses from the LISTENER show understanding of the feelings of the SPEAKER talking about their experience?
- Relevance: Did the responses of the LISTENER seem appropriate to the conversation? Were they on-topic?
- Fluency: Could you understand the responses from the LISTENER? Did the language seem accurate?

**Human A/B Test** In this human evaluation task, we aim to directly compare the generated responses with each other. We randomly sample 100 dialogues each for *MoEL vs* {*TRS, Multi-TRS*}. Three workers are given randomly ordered responses from either MoEL or {TRS, Multi-TRS}, and are prompted to choose the better response. They can either choose one of the responses or select *tie* when the provided options are either both good or both bad.

## 5 Results

**Emotion detection** To verify whether our model can attend to the appropriate listeners, we compute the emotion detection accuracy for each turn. Our model achieve 38%, 63%, 74% in terms of top-1, top-3, top-5 detection accuracy over 32 emotions. We notice that some emotions frequently appear in similar context (e.g., Annoyed, Angry, Furious) which might degrade the detection accuracy. Figure 3 shows the per class accuracy in the test set. We can see that by using top-5 the majority of the emotion achieve around 80% accuracy.

**Response evaluation** Both automatic and human evaluation results are shown in Table 2. TRS achieves the highest BLEU score and *Fluency* score but the lowest *Empathy* and *Relevance* score. This shows us that the responses generated by TRS are more generic but cannot accurately capture the user emotions. With the additional supervision on user emotions, multi-task training improves both *Empathy* and *Relevance* score, but it still degrades *Fluency*. In contrast, MoEL achieves the highest

| | |
|---|---|
| **Emotion** | Disgusted |
| **Context** | **Speaker**: i had to dog-sit for my dad about a week ago . i had to clean up dog poop at least 10 times , it was awful . |
| **TRS** | that is **horrible** . i hope you did not have to clean it up . |
| **Multi-TRS** | oh my goodness , that must have been so **scary** ! |
| **MoEL** | oh wow , that is pretty **gross** . |
| **Goal** | ohh , cleaning dog poop is always been annoying , yet petting the dog is always amusing |
| **Emotion** | Disappointed |
| **Context** | **Speaker**: i was turned down for a job i really thought i would be a good fit at .<br>**Listener**: did they give you any reason for not giving you the job ?<br>**Speaker**: someone else interviewed who had much more experience than i did . |
| **TRS** | that is good . i hope you have a good time . |
| **Multi-TRS** | that is a good idea . i hope it was a good idea . |
| **MoEL** | i am sorry to hear that . i am sure you will do great again |
| **Goal** | well , at least a candidate that deserved the job got it . |

| | |
|---|---|
| **Speaker**: so my friend 's baby fell down and scraped his knee yesterday and i was trying so hard to cheer him up.<br>**Listener**$_{caring}$: that was really nice of you. I am glad you were able to help him. | |
| **Listener**$_{terrified}$ | oh wow ! that must have been so scary . |
| **Listener**$_{sad}$ | oh no , i am so sorry to hear that . |
| **Listener**$_{excited}$ | that sounds like a good idea . |
| **Listener**$_{proud}$ | that was awesome ! did he get any good at you ? |
| **Speaker**: my husband lost a job but I am hoping he can find a full-time job soon.<br>**Listener**$_{hopeful}$: what kind of job is it? | |
| **Listener**$_{terrified}$ | oh that sucks . you should have been together . |
| **Listener**$_{sad}$ | i am so sorry . i am sure he will be fine . |
| **Listener**$_{excited}$ | that sounds awesome . what kind of job did he get you for ? |
| **Listener**$_{proud}$ | oh wow ! congratulations to him . you must be proud of him . |

Table 4: Generated responses from TRS, Multi-TRS and MoEL in 2 different user emotion states (**top**) and comparing generation from different listeners (**bottom**). We use hard attention on Terrified, Sad, Excited and Proud listeners.

*Empathy* and *Relevance* score. This suggests that the multi-expert strategy helps to capture the user emotional states and context simultaneously, and elicits a more appropriate response. The human A/B tests also confirm that the responses from our model are more preferred by human judges.

## 6 Analysis

In order to understand whether or how MoEL can effectively improve other baselines, learn each emotion, and properly react to them, we conduct three different analyses: model response comparison, listener analysis, and visualization of the emotion distribution $p$.

**Model response comparison** The top part of Table 4 compares the generated responses from MoEL and the two baselines on two different speaker emotional states. In the first example, MoEL captures the exact emotion of the speaker, by replying with "cleaning up dog poop is pretty **gross**", instead of "**horrible**" and "**scary**". In the second example, both TRS and Multi-TRS fail to understand that the speaker is disappointed about the failure of his interview, and they generate inappropriate responses. On the other hand, MoEL shows an empathetic response by comforting the speaker with "I am sure you will do great again". More examples can be find in the Appendix.

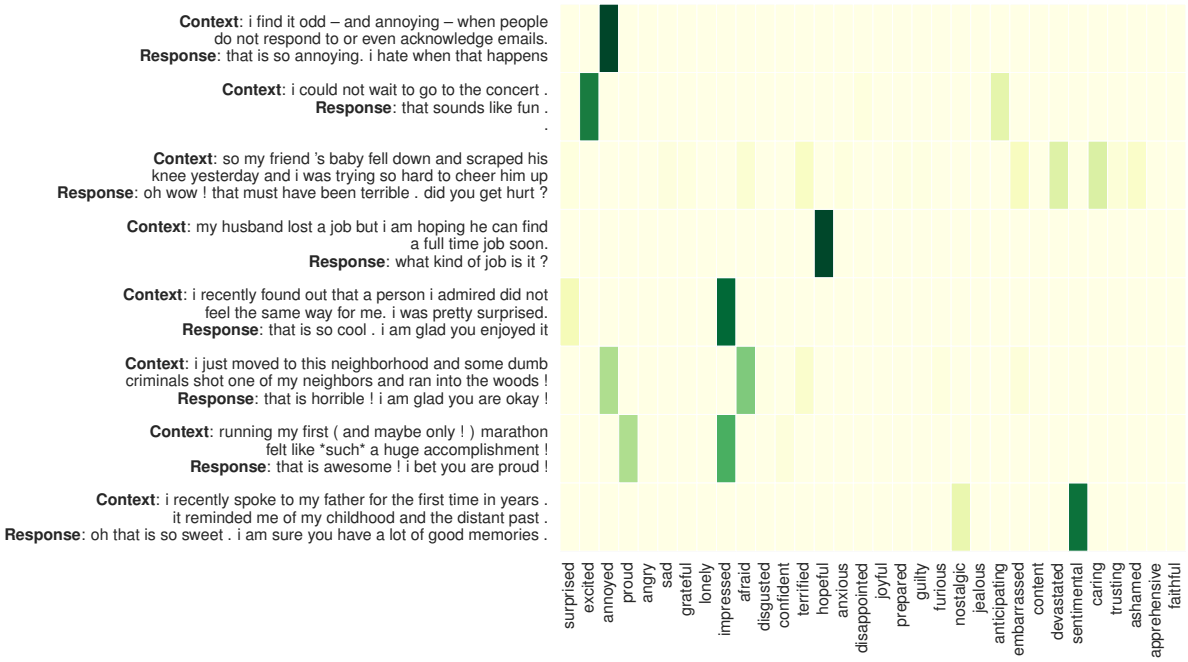


Figure 4: The visualization of attention on the listeners: The left side is the context followed by the responses generated by MoEL. The heat map illustrate the attention weights on 32 listeners

**Listener analysis** To have a better understanding of how each listener learned to react to different context, we conduct a study of comparing responses produced by different listeners. To do so, we *fix* the input dialogue context and we manually modify the attention vector distribution $p$ used to produce the response. We experiment with the correct listener and four other listeners: **Listener**$_{terrified}$, **Listener**$_{sad}$, **Listener**$_{excited}$, **Listener**$_{proud}$. Given the same context, we expect that different listeners will react differently, as this is our inductive bias. For example, **Listener**$_{sad}$ is optimized to comfort sad people, and **Listener**$_{\{excited,proud\}}$ share the positive emotions from the user. From the generation results in the bottom parts of Table 4 we can see that the corresponding listeners can produce empathetic and relevant responses when they reasonably match the speaker emotions. However, when the expected emotion label is opposite to the selected listener, such as *caring* and *sad*, the response becomes emotionally inappropriate.

Interestingly, in the last example, the *sad* listener actually produces a more meaningful response by encouraging the speaker. This is due to the first part of the context which conveys a sad emotion. On the other hand, for the same example, the *excited* listener responds with very relevant yet unsympathetic response. In addition, as many dialogue contexts contain multiple emotions, being able to capture them would lead to a better understanding of the speaker emotional state.

**Visualization of Emotion Distribution** Finally, to understand how MoEL chooses the listener according to the context, we visualize the emotion distribution $p$ in Figure 4. In most of the cases, the model attends to the proper listeners (emotions), and generate a proper responses. This is confirmed also by the accuracy results shown in Figure 3. However, our model is sometimes focuses on parts of the dialogue context. For example, in the fifth example in Figure 4, the model fails to detect the real emotion of speaker as the context contains "I was pretty **surprised**" in its last turn.

On the other hand, the last three rows of the heatmap indicate that the model learns to leverage **multiple** listeners to produce an empathetic response. For example, when the speaker talks about some criminals that shot one of his neighbors, MoEL successfully detects both *annoyed* and *afraid* emotions from the context, and replies with an appropriate response "that is horrible! i am glad you are okay!" that addresses both emotions. However, in the third row, the model produces "you" instead of "he" by mistake. Although the model is able to capture relevant emotions for this case, other emotions also have non-negligible weights which results in a smooth emotion distri-

bution $p$ that confuses the meta listener from accurately generating a response.

## 7 Conclusion & Future Work

In this paper, we propose a novel way to generate empathetic dialogue responses by using Mixture of Empathetic Listeners (MoEL). Differently from previous works, our model understands the user feelings and responds accordingly by learning specific listeners for each emotion. We benchmark our model in *empathetic-dialogues* dataset (Rashkin et al., 2018), which is a multi-turn open-domain conversation corpus grounded on emotional situations. Our experimental results show that MoEL is able to achieve competitive performance in the task with the advantage of being more interpretable than other conventional models. Finally, we show that our model is able to automatically select the correct emotional decoder and effectively generate an empathetic response.

One of the possible extensions of this work would be incorporating it with Persona (Zhang et al., 2018a) and task-oriented dialogue systems (Gao et al., 2018; Madotto et al., 2018; Wu et al., 2019, 2017, 2018a; Reddy et al., 2018; Raghu et al., 2019). Having a persona would allow the system to have more consistent and personalized responses, and combining open-domain conversations with task-oriented dialogue systems would equip the system with more engaging conversational capabilities, hence resulting in a more versatile dialogue system.

## Acknowledgments

This work has been partially funded by ITF/319/16FP and MRP/055/18 of the Innovation Technology Commission, the Hong Kong SAR Government. We sincerely thank the three anonymous reviewers for their insightful comments.

## References

Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. 2017. Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3366–3375.

Dario Bertero, Farhad Bin Siddique, Chien-Sheng Wu, Yan Wan, Ricky Ho Yin Chan, and Pascale Fung. 2016. Real-time speech emotion and sentiment recognition for interactive dialogue systems. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1042–1047.

Deng Cai, Yan Wang, Victoria Bi, Zhaopeng Tu, Xiaojiang Liu, Wai Lam, and Shuming Shi. 2018. Skeleton-to-response: Dialogue generation guided by retrieval memory. *arXiv preprint arXiv:1809.05296*.

Ankush Chatterjee, Umang Gupta, Manoj Kumar Chinnakotla, Radhakrishnan Srikanth, Michel Galley, and Puneet Agrawal. 2019a. Understanding emotions in text using deep learning and big data. *Computers in Human Behavior*, 93:309–317.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019b. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

Ronan Collobert, Samy Bengio, and Yoshua Bengio. 2002. A parallel mixture of svms for very large scale problems. In *Advances in Neural Information Processing Systems*, pages 633–640.

Marc Deisenroth and Jun Wei Ng. 2015. Distributed gaussian processes. In *International Conference on Machine Learning*, pages 1481–1490.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Emily Dinan, Varvara Logacheva, Valentin Malykh, Alexander Miller, Kurt Shuster, Jack Urbanek, Douwe Kiela, Arthur Szlam, Iulian Serban, Ryan Lowe, et al. 2019. The second conversational intelligence challenge (convai2). *arXiv preprint arXiv:1902.00098*.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2018. Wizard of wikipedia: Knowledge-powered conversational agents. *ICLR*.

Yingruo Fan, Jacqueline CK Lam, and Victor OK Li. 2018a. Multi-region ensemble convolutional neural network for facial expression recognition. In *International Conference on Artificial Neural Networks*, pages 84–94. Springer.

Yingruo Fan, Jacqueline CK Lam, and Victor OK Li. 2018b. Unsupervised domain adaptation with generative adversarial networks for facial emotion recognition. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 4460–4464. IEEE.

Yingruo Fan, Jacqueline CK Lam, and Victor OK Li. 2018c. Video-based emotion recognition using deeply-supervised neural networks. In *Proceedings of the 2018 on International Conference on Multimodal Interaction*, pages 584–588. ACM.

Jianfeng Gao, Michel Galley, and Lihong Li. 2018. Neural approaches to conversational ai. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1371–1374. ACM.

Braden Hancock, Antoine Bordes, Pierre-Emmanuel Mazare, and Jason Weston. 2019. Learning from dialogue after deployment: Feed yourself, chatbot! *arXiv preprint arXiv:1901.05415*.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *International Conference on Machine Learning*, pages 1587–1596.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, Geoffrey E Hinton, et al. 1991. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87.

Michael I Jordan and Robert A Jacobs. 1994. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214.

Chaitanya K Joshi, Fei Mi, and Boi Faltings. 2017. Personalization in goal-oriented dialog. *arXiv preprint arXiv:1706.07503*.

Lukasz Kaiser, Aidan N Gomez, Noam Shazeer, Ashish Vaswani, Niki Parmar, Llion Jones, and Jakob Uszkoreit. 2017. One model to learn them all. *arXiv preprint arXiv:1706.05137*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ilya Kulikov, Alexander H Miller, Kyunghyun Cho, and Jason Weston. 2018. Importance of a search strategy in neural dialogue modelling. *arXiv preprint arXiv:1811.00907*.

Nayeon Lee, Zihan Liu, and Pascale Fung. 2019. Team yeon-zi at semeval-2019 task 4: Hyperpartisan news detection by de-noising weakly-labeled data. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 1052–1056.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119.

Jiwei Li, Michel Galley, Chris Brockett, Georgios Spithourakis, Jianfeng Gao, and Bill Dolan. 2016b. A persona-based neural conversation model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 994–1003.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016c. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2122–2132.

Nurul Lubis, Sakriani Sakti, Koichiro Yoshino, and Satoshi Nakamura. 2018. Eliciting positive emotion through affect-sensitive dialogue response generation: A neural network approach. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Andrea Madotto, Zhaojiang Lin, Chien-Sheng Wu, and Pascale Fung. 2019. Personalizing dialogue agents via meta-learning. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 5454–5459.

Andrea Madotto, Chien-Sheng Wu, and Pascale Fung. 2018. Mem2seq: Effectively incorporating knowledge bases into end-to-end task-oriented dialog systems. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1468–1478.

Pierre-Emmanuel Mazare, Samuel Humeau, Martin Raison, and Antoine Bordes. 2018a. Training millions of personalized dialogue agents. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2775–2779.

Pierre-Emmanuel Mazare, Samuel Humeau, Martin Raison, and Antoine Bordes. 2018b. Training millions of personalized dialogue agents. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2775–2779. Association for Computational Linguistics.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1409.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Dinesh Raghu, Nikhil Gupta, and Mausam. 2019. Disentangling Language and Knowledge in Task-Oriented Dialogs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1239–1255, Minneapolis, Minnesota. Association for Computational Linguistics.

Hannah Rashkin, Eric Michael Smith, Margaret Li, and Y-Lan Boureau. 2018. I know the feeling: Learning to converse with empathy. *arXiv preprint arXiv:1811.00207*.

Carl E Rasmussen and Zoubin Ghahramani. 2002. Infinite mixtures of gaussian process experts. In *Advances in neural information processing systems*, pages 881–888.

Revanth Reddy, Danish Contractor, Dinesh Raghu, and Sachindra Joshi. 2018. Multi-level memory for task oriented dialogs. *arXiv preprint arXiv:1810.10647*.

Jurgen Schmidhuber. 1987. Evolutionary principles in self-referential learning. on learning now to learn: The meta-meta-meta...-hook. Diploma thesis, Technische Universitat Munchen, Germany, 14 May.

Iulian Vlad Serban, Ryan Lowe, Laurent Charlin, and Joelle Pineau. 2016. Generative deep neural networks for dialogue: A short review. *arXiv preprint arXiv:1611.06216*.

Babak Shahbaba and Radford Neal. 2009. Nonlinear models using dirichlet process mixtures. *Journal of Machine Learning Research*, 10(Aug):1829–1850.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

Jamin Shin, Peng Xu, Andrea Madotto, and Pascale Fung. 2019. Happybot: Generating empathetic dialogue responses by improving user experience lookahead. *arXiv preprint arXiv:1906.08487*.

Lucas Theis and Matthias Bethge. 2015. Generative image modeling using spatial lstms. In *Advances in Neural Information Processing Systems*, pages 1927–1935.

Volker Tresp. 2001. Mixtures of gaussian processes. In *Advances in neural information processing systems*, pages 654–660.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. In *International Conference on Machine Learning*.

Ke Wang and Xiaojun Wan. 2018. Sentigan: generating sentimental texts via mixture adversarial networks. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 4446–4452. AAAI Press.

Jason Weston, Emily Dinan, and Alexander Miller. 2018. Retrieve and refine: Improved sequence generation models for dialogue. In *Proceedings of the 2018 EMNLP Workshop SCAI: The 2nd International Workshop on Search-Oriented Conversational AI*, pages 87–92.

Genta Indra Winata, Onno Kampman, Yang Yang, Anik Dey, and Pascale Fung. 2017. Nora the empathetic psychologist. *Proc. Interspeech 2017*, pages 3437–3438.

Genta Indra Winata, Andrea Madotto, Zhaojiang Lin, Jamin Shin, Yan Xu, Peng Xu, and Pascale Fung. 2019. Caire_hkust at semeval-2019 task 3: Hierarchical attention for dialogue emotion classification. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 142–147.

Thomas Wolf, Victor Sanh, Julien Chaumond, and Clement Delangue. 2019. Transfertransfo: A transfer learning approach for neural network based conversational agents. *arXiv preprint arXiv:1901.08149*.

Chien-Sheng Wu, Andrea Madotto, Genta Winata, and Pascale Fung. 2017. End-to-end recurrent entity network for entity-value independent goal-oriented dialog learning. In *Dialog System Technology Challenges Workshop, DSTC6*.

Chien-Sheng Wu, Andrea Madotto, Genta Indra Winata, and Pascale Fung. 2018a. End-to-end dynamic query memory network for entity-value independent task-oriented dialog. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6154–6158. IEEE.

Chien-Sheng Wu, Richard Socher, and Caiming Xiong. 2019. Global-to-local memory pointer networks for task-oriented dialogue. *arXiv preprint arXiv:1901.04713*.

Yu Wu, Furu Wei, Shaohan Huang, Zhoujun Li, and Ming Zhou. 2018b. Response generation by context-aware prototype editing. *arXiv preprint arXiv:1806.07042*.

Peng Xu, Andrea Madotto, Chien-Sheng Wu, Ji Ho Park, and Pascale Fung. 2018. Emo2vec: Learning generalized emotion representation by multi-task training. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 292–298.

Bangpeng Yao, Dirk Walther, Diane Beck, and Li Fei-Fei. 2009. Hierarchical mixture of classification experts uncovers interactions between brain regions. In *Advances in Neural Information Processing Systems*, pages 2178–2186.

Semih Yavuz, Abhinav Rastogi, Guanlin Chao, Dilek Hakkani-Tür, and Amazon Alexa AI. 2018. Deepcopy: Grounded response generation with hierarchical pointer networks. *ConvAI Workshop@NIPS*.

Yury Zemlyanskiy and Fei Sha. 2018. Aiming to know you better perhaps makes me a more engaging dialogue partner. *CoNLL 2018*, page 551.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018a. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2204–2213.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018b. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2204–2213. Association for Computational Linguistics.

Hao Zhou, Minlie Huang, Tianyang Zhang, Xiaoyan Zhu, and Bing Liu. 2018. Emotional chatting machine: Emotional conversation generation with internal and external memory. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Xianda Zhou and William Yang Wang. 2018. Mojitalk: Generating emotional responses at scale. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1128–1137.

# Entity-Consistent End-to-end Task-Oriented Dialogue System with KB Retriever

**Libo Qin, Yijia Liu, Wanxiang Che*, Haoyang Wen, Yangming Li, Ting Liu**
Research Center for Social Computing and Information Retrieval
Harbin Institute of Technology, China
{lbqin,yjliu,car,hywen,yangmingli,tliu}@ir.hit.edu.cn

## Abstract

Querying the knowledge base (KB) has long been a challenge in the end-to-end task-oriented dialogue system. Previous sequence-to-sequence (Seq2Seq) dialogue generation work treats the KB query as an attention over the entire KB, without the guarantee that the generated entities are consistent with each other. In this paper, we propose a novel framework which queries the KB in two steps to improve the consistency of generated entities. In the first step, inspired by the observation that a response can usually be supported by a single KB row, we introduce a KB retrieval component which explicitly returns the most relevant KB row given a dialogue history. The retrieval result is further used to filter the irrelevant entities in a Seq2Seq response generation model to improve the consistency among the output entities. In the second step, we further perform the attention mechanism to address the most correlated KB column. Two methods are proposed to make the training feasible without labeled retrieval data, which include distant supervision and Gumbel-Softmax technique. Experiments on two publicly available task oriented dialog datasets show the effectiveness of our model by outperforming the baseline systems and producing entity-consistent responses.

## 1 Introduction

*Task-oriented dialogue system*, which helps users to achieve specific goals with natural language, is attracting more and more research attention. With the success of the sequence-to-sequence (Seq2Seq) models in text generation (Sutskever et al., 2014; Bahdanau et al., 2014; Luong et al., 2015a; Nallapati et al., 2016b,a), several works tried to model the task-oriented dialogue as the Seq2Seq generation of response from the dialogue history (Eric and Manning, 2017; Eric et al., 2017; Madotto et al., 2018). This kind of modeling scheme frees the task-oriented dialogue system from the manually designed pipeline modules and heavy annotation labor for these modules. Different from typical text generation, the successful conversations for task-oriented dialogue system heavily depend on accurate knowledge base (KB) queries. Taking the dialogue in Figure 1 as an example, to answer the driver's query on the gas station, the dialogue system is required to retrieve the entities like "`200 Alester Ave`" and "`Valero`". For the task-oriented system based on Seq2Seq generation, there is a trend in recent study towards modeling the KB query as an attention network over the entire KB entity representations, hoping to learn a model to pay more attention to the relevant entities (Eric et al., 2017; Madotto et al., 2018; Reddy et al., 2018; Wen et al., 2018).

| Address | Distance | POI type | POI | Traffic info |
|---|---|---|---|---|
| 638 Amherst St | 3 miles | grocery store | Sigona Farmers Market | car collision nearby |
| 269 Alger Dr | 1 miles | coffee or tea place | Cafe Venetia | car collision nearby |
| 5672 barringer street | 5 miles | certain address | 5672 barringer street | no traffic |
| 200 Alester Ave | 2 miles | gas station | Valero | road block nearby |
| 899 Ames Ct | 5 miles | hospital | Stanford Childrens Health | moderate traffic |
| 481 Amaranta Ave | 1 miles | parking garage | Palo Alto Garage R | moderate traffic |
| 145 Amherst St | 1 miles | coffee or tea place | Teavana | road block nearby |
| 409 Bollard St | 5 miles | grocery store | Willows Market | no traffic |

**Driver** Address to the gas station.
**Car** Valero is located at 200 Alester Ave.
**Driver** OK , please give me directions via a route that avoids all heavy traffic.
**Car** Since there is a road block nearby, I found another route for you and I sent it on your screen.
**Driver** Awesome thank you.

Figure 1: An example of a task-oriented dialogue that incorporates a knowledge base (KB). The fourth row in KB supports the second turn of the dialogue. A dialogue system will produce a response with *conflict entities* if it includes the POI in the fourth row and the address in the fifth row, like "*Valero is located at 899 Ames Ct*".

Though achieving good end-to-end dialogue generation with over-the-entire-KB attention mechanism, these methods do not guarantee the generation consistency regarding KB entities and

* Email corresponding.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 133–142, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

sometimes yield responses with conflict entities, like "*Valero is located at 899 Ames Ct*" for the gas station query (as shown in Figure 1). In fact, the correct address for *Valero* is *200 Alester Ave*. A consistent response is relatively easy to achieve for the conventional pipeline systems because they query the KB by issuing API calls (Bordes and Weston, 2017; Wen et al., 2017b,a), and the returned entities, which typically come from a single KB row, are consistently related to the object (like the "gas station") that serves the user's request. This indicates that *a response can usually be supported by a single KB row*. It's promising to incorporate such observation into the Seq2Seq dialogue generation model, since it encourages KB relevant generation and avoids the model from producing responses with conflict entities.

To achieve entity-consistent generation in the Seq2Seq task-oriented dialogue system, we propose a novel framework which query the KB in two steps. In the first step, we introduce a retrieval module — KB-retriever to explicitly query the KB. Inspired by the observation that a single KB row usually supports a response, given the dialogue history and a set of KB rows, the KB-retriever uses a memory network (Sukhbaatar et al., 2015) to select the most relevant row. The retrieval result is then fed into a Seq2Seq dialogue generation model to filter the irrelevant KB entities and improve the consistency within the generated entities. In the second step, we further perform attention mechanism to address the most correlated KB column. Finally, we adopt the copy mechanism to incorporate the retrieved KB entity.

Since dialogue dataset is not typically annotated with the retrieval results, training the KB-retriever is non-trivial. To make the training feasible, we propose two methods: 1) we use a set of heuristics to derive the training data and train the retriever in a distant supervised fashion; 2) we use Gumbel-Softmax (Jang et al., 2017) as an approximation of the non-differentiable selecting process and train the retriever along with the Seq2Seq dialogue generation model. Experiments on two publicly available datasets (**Camrest** (Wen et al., 2017b) and **InCar Assistant** (Eric et al., 2017)) confirm the effectiveness of the KB-retriever. Both the retrievers trained with distant-supervision and Gumbel-Softmax technique outperform the compared systems in the automatic and human evaluations. Analysis empirically verifies our assumption that more than 80% responses in the dataset can be supported by a single KB row and better retrieval results lead to better task-oriented dialogue generation performance.

## 2 Definition

In this section, we will describe the input and output of the end-to-end task-oriented dialogue system, and the definition of Seq2Seq task-oriented dialogue generation.

### 2.1 Dialogue History

Given a dialogue between a user ($u$) and a system ($s$), we follow Eric et al. (2017) and represent the $k$-turned *dialogue utterances* as $\{(u_1, s_1), (u_2, s_2), ..., (u_k, s_k)\}$. At the $i^{\text{th}}$ turn of the dialogue, we aggregate dialogue context which consists of the tokens of $(u_1, s_1, ..., s_{i-1}, u_i)$ and use $\mathbf{x} = (x_1, x_2, ..., x_m)$ to denote the whole *dialogue history* word by word, where $m$ is the number of tokens in the dialogue history.

### 2.2 Knowledge Base

In this paper, we assume to have the access to a relational-database-like KB $B$, which consists of $|\mathcal{R}|$ rows and $|\mathcal{C}|$ columns. The value of entity in the $j^{\text{th}}$ row and the $i^{\text{th}}$ column is noted as $v_{j,i}$.

### 2.3 Seq2Seq Dialogue Generation

We define the Seq2Seq task-oriented dialogue generation as finding the most likely response $\mathbf{y}$ according to the input dialogue history $\mathbf{x}$ and KB $B$. Formally, the probability of a response is defined as

$$p(\mathbf{y} \mid \mathbf{x}, B) = \prod_{t=1}^{n} p(y_t \mid y_1, ..., y_{t-1}, \mathbf{x}, B),$$

where $y_t$ represents an output token.

## 3 Our Framework

In this section, we describe our framework for end-to-end task-oriented dialogues. The architecture of our framework is demonstrated in Figure 2, which consists of two major components including an memory network-based retriever and the seq2seq dialogue generation with KB Retriever. Our framework first uses the KB-retriever to select the most relevant KB row and further filter the irrelevant entities in a Seq2Seq response generation model to improve the consistency among the

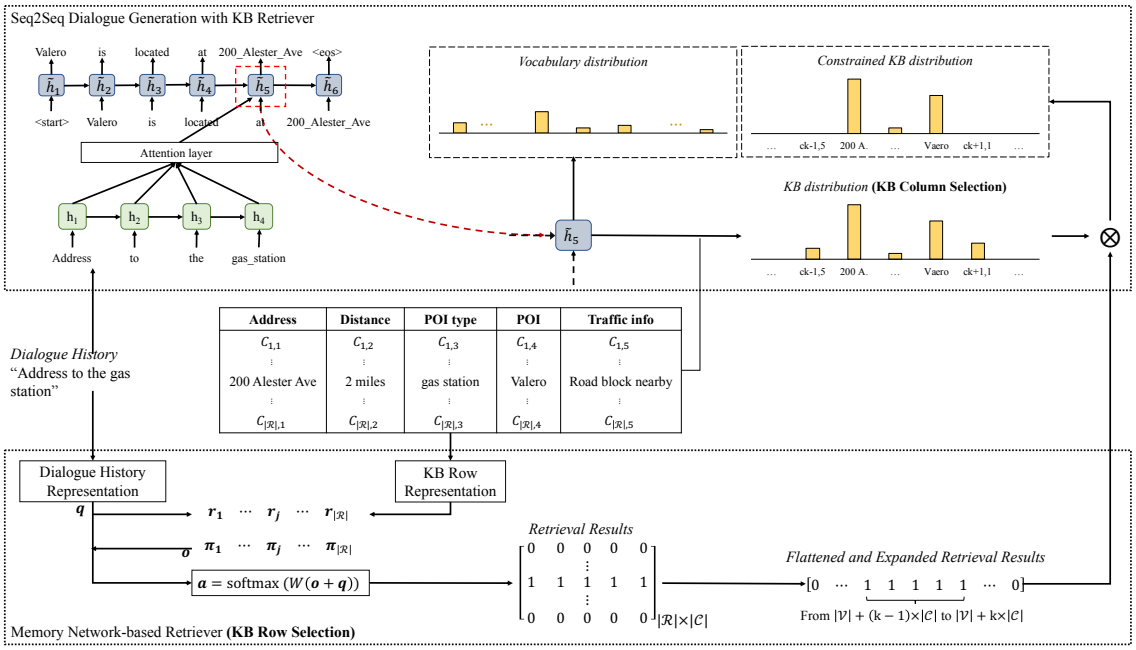


Figure 2: The workflow of our Seq2Seq task-oriented dialogue generation model with KB-retriever. For simplification, we draw the single-hop memory network instead of the multiple-hop one we use in our model.

output entities. While in decoding, we further perform the attention mechanism to choose the most probable KB column. We will present the details of our framework in the following sections.

### 3.1 Encoder

In our encoder, we adopt the bidirectional LSTM (Hochreiter and Schmidhuber, 1997, BiLSTM) to encode the dialogue history $\mathbf{x}$, which captures temporal relationships within the sequence. The encoder first map the tokens in $\mathbf{x}$ to vectors with embedding function $\phi^{\text{emb}}$, and then the BiLSTM read the vector forwardly and backwardly to produce context-sensitive hidden states $(\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_m)$ by repeatedly applying the recurrence $\mathbf{h}_i = \text{BiLSTM}\left(\phi^{\text{emb}}\left(x_i\right), \mathbf{h}_{i-1}\right)$.

### 3.2 Vanilla Attention-based Decoder

Here, we follow Eric et al. (2017) to adopt the attention-based decoder to generation the response word by word. LSTM is also used to represent the partially generated output sequence $(y_1, y_2, ..., y_{t-1})$ as $(\tilde{\mathbf{h}}_1, \tilde{\mathbf{h}}_2, ..., \tilde{\mathbf{h}}_t)$. For the generation of next token $y_t$, their model first calculates an attentive representation $\tilde{\mathbf{h}}_t'$ of the dialogue history as

$$\mathbf{u}_i^t = W_2 \tanh(W_1 [\mathbf{h}_i, \tilde{\mathbf{h}}_t]),$$
$$\mathbf{a}_i^t = \text{softmax}(\mathbf{u}_i^t),$$
$$\tilde{\mathbf{h}}_t' = \sum_{i=1}^{m} \mathbf{a}_i^t \cdot \mathbf{h}_i.$$

Then, the concatenation of the hidden representation of the partially outputted sequence $\tilde{\mathbf{h}}_t$ and the attentive dialogue history representation $\tilde{\mathbf{h}}_t'$ are projected to the vocabulary space $\mathcal{V}$ by $U$ as

$$\mathbf{o}_t = U\,[\tilde{\mathbf{h}}_t, \tilde{\mathbf{h}}_t'],$$

to calculate the score (logit) for the next token generation. The probability of next token $y_t$ is finally calculated as

$$p(y_t \mid y_1, ..., y_{t-1}, \mathbf{x}, B) = \text{softmax}(\mathbf{o}_t).$$

### 3.3 Entity-Consistency Augmented Decoder

As shown in section 3.2, we can see that the generation of tokens are just based on the dialogue history attention, which makes the model ignorant to the KB entities. In this section, we present how to query the KB explicitly in two steps for improving the entity consistence, which first adopt the KB-retriever to select the most relevant KB row and the generation of KB entities from the entities-augmented decoder is constrained to the entities

within the most probable row, thus improve the entity generation consistency. Next, we perform the column attention to select the most probable KB column. Finally, we show how to use the copy mechanism to incorporate the retrieved entity while decoding.

#### 3.3.1 KB Row Selection

In our framework, our KB-retriever takes the dialogue history and KB rows as inputs and selects the most relevant row. This selection process resembles the task of selecting one word from the inputs to answer questions (Sukhbaatar et al., 2015), and we use a memory network to model this process. In the following sections, we will first describe how to represent the inputs, then we will talk about our memory network-based retriever

**Dialogue History Representation:** We encode the dialogue history by adopting the neural bag-of-words (BoW) followed the original paper (Sukhbaatar et al., 2015). Each token in the dialogue history is mapped into a vector by another embedding function $\phi^{\text{emb}'}(x)$ and the dialogue history representation $\mathbf{q}$ is computed as the sum of these vectors: $\mathbf{q} = \sum_{i=1}^{m} \phi^{\text{emb}'}(x_i)$.

**KB Row Representation:** In this section, we describe how to encode the KB row. Each KB cell is represented as the cell value $v$ embedding as $\mathbf{c}_{j,k} = \phi^{\text{value}}(v_{j,k})$, and the neural BoW is also used to represent a KB row $\mathbf{r}_j$ as $\mathbf{r}_j = \sum_{k=1}^{|\mathcal{C}|} \mathbf{c}_{j,k}$.

**Memory Network-Based Retriever:** We model the KB retrieval process as selecting the row that most-likely supports the response generation. Memory network (Sukhbaatar et al., 2015) has shown to be effective to model this kind of selection. For a $n$-hop memory network, the model keeps a set of input matrices $\{R^1, R^2, ..., R^{n+1}\}$, where each $R^i$ is a stack of $|\mathcal{R}|$ inputs $(\mathbf{r}_1^i, \mathbf{r}_2^i, ..., \mathbf{r}_{|\mathcal{R}|}^i)$. The model also keeps query $\mathbf{q}^1$ as the input. A single hop memory network computes the probability $\mathbf{a}_j$ of selecting the $j^{\text{th}}$ input as

$$\boldsymbol{\pi}^1 = \text{softmax}((\mathbf{q}^1)^T R^1),$$
$$\mathbf{o}^1 = \sum_i \boldsymbol{\pi}_i^1 \mathbf{r}_i^2,$$
$$\mathbf{a} = \text{softmax}(W^{\text{mem}} (\mathbf{o}^1 + \mathbf{q}^1)).$$

For the multi-hop cases, layers of single hop memory network are stacked and the query of the $(i+1)^{\text{th}}$ layer network is computed as

$$\mathbf{q}^{i+1} = \mathbf{q}^i + \mathbf{o}^i,$$

and the output of the last layer is used as the output of the whole network. For more details about memory network, please refer to the original paper (Sukhbaatar et al., 2015).

After getting $\mathbf{a}$, we represent the retrieval results as a 0-1 matrix $T \in \{0,1\}^{|\mathcal{R}|\times|\mathcal{C}|}$, where each element in $T$ is calculated as

$$T_{j,*} = \mathbb{1}[j = \underset{i}{\text{argmax}}\, \mathbf{a}_i]. \quad (1)$$

In the retrieval result, $T_{j,k}$ indicates whether the entity in the $j^{\text{th}}$ row and the $k^{\text{th}}$ column is relevant to the final generation of the response. In this paper, we further flatten T to a 0-1 vector $\mathbf{t} \in \{0,1\}^{|\mathcal{E}|}$ (where $|\mathcal{E}|$ equals $|\mathcal{R}| \times |\mathcal{C}|$) as our retrieval row results.

#### 3.3.2 KB Column Selection

After getting the retrieved row result that indicates which KB row is the most relevant to the generation, we further perform column attention in decoding time to select the probable KB column. For our KB column selection, following the Eric et al. (2017) we use the decoder hidden state $(\tilde{\mathbf{h}}_1, \tilde{\mathbf{h}}_2, ..., \tilde{\mathbf{h}}_t)$ to compute an attention score with the embedding of column attribute name. The attention score $\mathbf{c} \in R^{|\mathcal{E}|}$ then become the logits of the column be selected, which can be calculated as

$$\mathbf{c}_j = W_2^{'} \tanh(W_1^{'} [\mathbf{k}_j, \tilde{\mathbf{h}}_t]),$$

where $\mathbf{c}_j$ is the attention score of the $j^{\text{th}}$ KB column, $\mathbf{k}_j$ is represented with the embedding of word embedding of KB column name. $W_1^{'}$, $W_2^{'}$ and $\mathbf{t}^T$ are trainable parameters of the model.

#### 3.3.3 Decoder with Retrieved Entity

After the row selection and column selection, we can define the final retrieved KB entity score as the element-wise dot between the row retriever result and the column selection score, which can be calculated as

$$\mathbf{v}^t = \mathbf{t} * \mathbf{c}, \quad (2)$$

where the $v^t$ indicates the final KB retrieved entity score. Finally, we follow Eric et al. (2017) to

use copy mechanism to incorporate the retrieved entity, which can be defined as

$$\mathbf{o}_t = U\,[\tilde{\mathbf{h}}_t, \tilde{\mathbf{h}'}_t] + \mathbf{v}^t,$$

where $\mathbf{o}_t$'s dimensionality is $|\mathcal{V}| + |\mathcal{E}|$. In $\mathbf{v}^t$ , lower $|\mathcal{V}|$ is zero and the rest $|\mathcal{E}|$ is retrieved entity scores.

## 4 Training the KB-Retriever

As mentioned in section 3.3.1, we adopt the memory network to train our KB-retriever. However, in the Seq2Seq dialogue generation, the training data does not include the annotated KB row retrieval results, which makes supervised training the KB-retriever impossible. To tackle this problem, we propose two training methods for our KB-row-retriever. 1) In the first method, inspired by the recent success of distant supervision in information extraction (Zeng et al., 2015; Mintz et al., 2009; Min et al., 2013; Xu et al., 2013), we take advantage of the similarity between the surface string of KB entries and the reference response, and design a set of heuristics to extract training data for the KB-retriever. 2) In the second method, instead of training the KB-retriever as an independent component, we train it along with the training of the Seq2Seq dialogue generation. To make the retrieval process in Equation 1 differentiable, we use Gumbel-Softmax (Jang et al., 2017) as an approximation of the argmax during training.

### 4.1 Training with Distant Supervision

Although it's difficult to obtain the annotated retrieval data for the KB-retriever, we can "guess" the most relevant KB row from the reference response, and then obtain the weakly labeled data for the retriever. Intuitively, for the current utterance in the same dialogue which usually belongs to one topic and the KB row that contains the largest number of entities mentioned in the whole dialogue should support the utterance. In our training with distant supervision, we further simplify our assumption and assume that one dialogue which is usually belongs to one topic and can be supported by the most relevant KB row, which means for a $k$-turned dialogue, we construct $k$ pairs of training instances for the retriever and all the inputs $(u_1, s_1, ..., s_{i-1}, u_i \mid i \leq k)$ are associated with the same weakly labeled KB retrieval result $T^*$.

In this paper, we compute each row's similarity to the whole dialogue and choose the most similar row as $T^*$. We define the similarity of each row as the number of matched spans with the surface form of the entities in the row. Taking the dialogue in Figure 1 for an example, the similarity of the 4th row equals to 4 with "`200 Alester Ave`", "`gas station`", "`Valero`", and "`road block nearby`" matching the dialogue context; and the similarity of the 7th row equals to 1 with only "`road block nearby`" matching.

In our model with the distantly supervised retriever, the retrieval results serve as the input for the Seq2Seq generation. During training the Seq2Seq generation, we use the weakly labeled retrieval result $T^*$ as the input.

### 4.2 Training with Gumbel-Softmax

In addition to treating the row retrieval result as an input to the generation model, and training the kb-row-retriever independently, we can train it along with the training of the Seq2Seq dialogue generation in an end-to-end fashion. The major difficulty of such a training scheme is that the discrete retrieval result is not differentiable and the training signal from the generation model cannot be passed to the parameters of the retriever. Gumbel-softmax technique (Jang et al., 2017) has been shown an effective approximation to the discrete variable and proved to work in sentence representation. In this paper, we adopt the Gumbel-Softmax technique to train the KB retriever. We use

$$T^{\text{approx}}_{j,*} = \frac{\exp((\log(\mathbf{a}_j) + \mathbf{g}_j)/\tau)}{\sum_i \exp((\log(\mathbf{a}_i) + \mathbf{g}_i)/\tau)},$$

as the approximation of $T$, where $\mathbf{g}_j$ are i.i.d samples drawn from Gumbel$(0, 1)$[1] and $\tau$ is a constant that controls the smoothness of the distribution. $T^{\text{approx}}_j$ replaces $T_j$ in equation 1 and goes through the same flattening and expanding process as $\mathbf{V}$ to get $\mathbf{v}^{\mathbf{t}^{\text{approx}'}}$ and the training signal from Seq2Seq generation is passed via the logit

$$\mathbf{o}^{\text{approx}}_t = U\,[\tilde{\mathbf{h}}_t, \tilde{\mathbf{h}}'_t] + \mathbf{v}^{\mathbf{t}^{\text{approx}'}}.$$

To make training with Gumbel-Softmax more stable, we first initialize the parameters by pre-training the KB-retriever with distant supervision and further fine-tuning our framework.

[1] We sample $\mathbf{g}$ by drawing $u \sim \text{Uniform}(0, 1)$ then computing $\mathbf{g} = -\log(-\log(u))$.

### 4.3 Experimental Settings

We choose the InCar Assistant dataset (Eric et al., 2017) including three distinct domains: navigation, weather and calendar domain. For weather domain, we follow Wen et al. (2018) to separate the highest temperature, lowest temperature and weather attribute into three different columns. For calendar domain, there are some dialogues without a KB or incomplete KB. In this case, we padding a special token "-" in these incomplete KBs. Our framework is trained separately in these three domains, using the same train/validation/test split sets as Eric et al. (2017).[2] To justify the generalization of the proposed model, we also use another public CamRest dataset (Wen et al., 2017b) and partition the datasets into training, validation and testing set in the ratio 3:1:1.[3] Especially, we hired some human experts to format the CamRest dataset by equipping the corresponding KB to every dialogues.

All hyper-parameters are selected according to validation set. We use a three-hop memory network to model our KB-retriever. The dimensionalities of the embedding is selected from $\{100, 200\}$ and LSTM hidden units is selected from $\{50, 100, 150, 200, 350\}$. The dropout we use in our framework is selected from $\{0.25, 0.5, 0.75\}$ and the batch size we adopt is selected from $\{1, 2\}$. L2 regularization is used on our model with a tension of $5 \times 10^{-6}$ for reducing overfitting. For training the retriever with distant supervision, we adopt the weight typing trick (Liu and Perez, 2017). We use Adam (Kingma and Ba, 2014) to optimize the parameters in our model and adopt the suggested hyper-parameters for optimization.

We adopt both the automatic and human evaluations in our experiments.

### 4.4 Baseline Models

We compare our model with several baselines including:

- **Attn seq2seq** (Luong et al., 2015b): A model with simple attention over the input context at each time step during decoding.
- **Ptr-UNK** (Gulcehre et al., 2016): Ptr-UNK is the model which augments a sequence-to-sequence architecture with attention-based copy mechanism over the encoder context.
- **KV Net** (Eric et al., 2017): The model adopted and argumented decoder which decodes over the concatenation of vocabulary and KB entities, which allows the model to generate entities.
- **Mem2Seq** (Madotto et al., 2018): Mem2Seq is the model that takes dialogue history and KB entities as input and uses a pointer gate to control either generating a vocabulary word or selecting an input as the output.
- **DSR** (Wen et al., 2018): DSR leveraged dialogue state representation to retrieve the KB implicitly and applied copying mechanism to retrieve entities from knowledge base while decoding.

In InCar dataset, for the *Attn seq2seq*, *Ptr-UNK* and *Mem2seq*, we adopt the reported results from Madotto et al. (2018). In CamRest dataset, for the *Mem2Seq*, we adopt their open-sourced code to get the results while for the *DSR*, we run their code on the same dataset to obtain the results.[4]

## 5 Results

Follow the prior works (Eric et al., 2017; Madotto et al., 2018; Wen et al., 2018), we adopt the *BLEU* and the *Micro Entity F1* to evaluate our model performance. The experimental results are illustrated in Table 1.

In the first block of Table 1, we show the Human, rule-based and KV Net (with*) result which are reported from Eric et al. (2017). We argue that their results are not directly comparable because their work uses the entities in thier canonicalized forms, which are not calculated based on real entity value. It's noticing that our framework with two methods still outperform KV Net in InCar dataset on whole BLEU and Entity F metrics, which demonstrates the effectiveness of our framework.

[2]We obtain the BLEU and Entity F1 score on the whole InCar dataset by mixing all generated response and evaluating them together.

[3]The dataset can be available at: https://github.com/yizhen20133868/Retriever-Dialogue

[4]We adopt the same pre-processed dataset from Madotto et al. (2018). We can find that experimental results is slightly different with their reported performance (Wen et al., 2018) because of their different tokenized utterances and normalization for entities.

| Model | InCar | | | | | CamRest | |
|---|---|---|---|---|---|---|---|
| | BLEU | F1 | Navigate F1 | Weather F1 | Calendar F1 | BLEU | F1 |
| Human* (Eric et al., 2017) | 13.5 | 60.7 | 55.2 | 61.6 | 64.3 | - | - |
| Rule-Based* (Eric et al., 2017) | 6.6 | 43.8 | 40.4 | 39.5 | 61.3 | - | - |
| KV Net* (Eric et al., 2017) | 13.2 | 48.0 | 41.3 | 47.0 | 62.9 | - | - |
| Attn seq2seq (Luong et al., 2015b) | 9.3 | 11.9 | 10.8 | 25.6 | 23.4 | - | - |
| Ptr-UNK (Gulcehre et al., 2016) | 8.3 | 22.7 | 14.9 | 26.7 | 26.9 | - | - |
| Mem2Seq (Madotto et al., 2018) | 12.6 | 33.4 | 20.0 | 32.8 | 49.3 | 16.6 | 42.4 |
| DSR (Wen et al., 2018) | 12.7 | 51.9 | 52.0 | 50.4 | 52.1 | 18.3 | 53.6 |
| w/ distant supervision | **14.1** | 51.9 | 51.6 | 49.6 | 54.2 | 17.4 | 58.0 |
| w/ Gumbel-Softmax | 13.9 | **53.7** | **54.5** | **52.2** | **55.6** | **18.5** | **58.6** |

Table 1: Comparison of our model with baselines

In the second block of Table 1, we can see that our framework trained with both the distant supervision and the Gumbel-Softmax beats all existing models on two datasets. Our model outperforms each baseline on both BLEU and F1 metrics. In InCar dataset, Our model with Gumbel-Softmax has the highest BLEU compared with baselines, which which shows that our framework can generate more fluent response. Especially, our framework has achieved 2.5% improvement on navigate domain, 1.8% improvement on weather domain and 3.5% improvement on calendar domain on F1 metric. It indicates that the effectiveness of our KB-retriever module and our framework can retrieve more correct entity from KB. In CamRest dataset, the same trend of improvement has been witnessed, which further show the effectiveness of our framework.

Besides, we observe that the model trained with Gumbel-Softmax outperforms with distant supervision method. We attribute this to the fact that the KB-retriever and the Seq2Seq module are fine-tuned in an end-to-end fashion, which can refine the KB-retriever and further promote the dialogue generation.

### 5.1 The proportion of responses that can be supported by a single KB row

In this section, we verify our assumption by examining the proportion of responses that can be supported by a single row.

We define a response being supported by the most relevant KB row as all the responded entities are included by that row. We study the proportion of these responses over the test set. The number is 95% for the navigation domain, 90% for the CamRest dataset and 80% for the weather domain. This confirms our assumption that most responses can be supported by the relevant KB row.

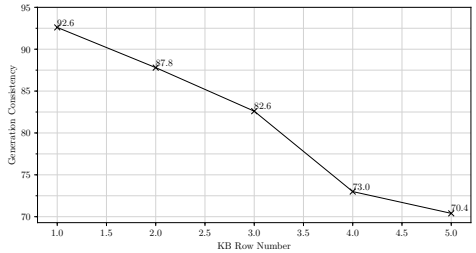


Figure 3: Correlation between the number of KB rows and generation consistency on navigation domain.

Correctly retrieving the supporting row should be beneficial.

We further study the weather domain to see the rest 20% exceptions. Instead of being supported by multiple rows, most of these exceptions cannot be supported by any KB row. For example, there is one case whose reference response is "*It 's not rainy today*", and the related KB entity is `sunny`. These cases provide challenges beyond the scope of this paper. If we consider this kind of cases as being supported by a single row, such proportion in the weather domain is 99%.

### 5.2 Generation Consistency

In this paper, we expect the consistent generation from our model. To verify this, we compute the consistency recall of the utterances that have multiple entities. An utterance is considered as consistent if it has multiple entities and these entities belong to the same row which we annotated with distant supervision.

The consistency result is shown in Table 2. From this table, we can see that incorporating retriever in the dialogue generation improves the consistency.

### 5.3 Correlation between the number of KB rows and generation consistency

To further explore the correlation between the number of KB rows and generation consistency, we conduct experiments with distant manner to study the correlation between the number of KB rows and generation consistency.

We choose KBs with different number of rows on a scale from 1 to 5 for the generation. From Figure 3, as the number of KB rows increase, we can see a decrease in generation consistency. This indicates that irrelevant information would harm the dialogue generation consistency.

### 5.4 Visualization

To gain more insights into how the our retriever module influences the whole KB score distribution, we visualized the KB entity probability at the decoding position where we generate the entity `200_Alester_Ave`. From the example (Fig 4), we can see the $4^{th}$ row and the $1^{th}$ column has the highest probabilities for generating `200_Alester_Ave`, which verify the effectiveness of firstly selecting the most relevant KB row and further selecting the most relevant KB column.

### 5.5 Human Evaluation

We provide human evaluation on our framework and the compared models. These responses are based on distinct dialogue history. We hire several human experts and ask them to judge the quality of the responses according to correctness, fluency, and humanlikeness on a scale from 1 to 5. In each judgment, the expert is presented with the dialogue history, an output of a system with the name anonymized, and the gold response.

The evaluation results are illustrated in Table 2. Our framework outperforms other baseline models on all metrics according to Table 2. The most significant improvement is from correctness, indicating that our model can retrieve accurate entity from KB and generate more informative information that the users want to know.

## 6 Related Work

Sequence-to-sequence (Seq2Seq) models in text generation (Sutskever et al., 2014; Bahdanau et al., 2014; Luong et al., 2015a; Nallapati et al., 2016b,a) has gained more popular and they are applied for the open-domain dialogs (Vinyals and Le, 2015; Serban et al., 2016) in the end-to-end training method. Recently, the Seq2Seq can be used for learning task oriented dialogs and how to query the structured KB is the remaining challenges.

Properly querying the KB has long been a challenge in the task-oriented dialogue system. In the pipeline system, the KB query is strongly correlated with the design of language understanding, state tracking, and policy management. Typically, after obtaining the dialogue state, the policy management module issues an API call accordingly to query the KB. With the development of neural network in natural language processing, efforts have been made to replacing the discrete and predefined dialogue state with the distributed representation (Bordes and Weston, 2017; Wen et al., 2017b,a; Liu and Lane, 2017). In our framework, our retrieval result can be treated as a numeric representation of the API call return.

Instead of interacting with the KB via API calls, more and more recent works tried to incorporate KB query as a part of the model. The most popular way of modeling KB query is treating it as an attention network over the entire KB entities (Eric et al., 2017; Dhingra et al., 2017; Reddy et al., 2018; Raghu et al., 2019; Wu et al., 2019) and the return can be a fuzzy summation of the entity representations. Madotto et al. (2018)'s practice of modeling the KB query with memory network can also be considered as learning an attentive prefer-

| Model | Cons. | Human Evaluation | | |
|---|---|---|---|---|
| | | Cor. | Flu. | Hum. |
| Copy Net | 21.2 | 4.14 | 4.40 | 4.36 |
| Mem2Seq | 38.1 | 4.29 | 4.29 | 4.27 |
| DSR | 70.3 | 4.59 | 4.71 | 4.65 |
| w/ distant supervision | 65.8 | 4.53 | 4.71 | 4.64 |
| w/ Gumble-Softmax | **72.1** | **4.64** | **4.73** | **4.69** |

Table 2: The generation consistency and Human Evaluation on navigation domain. *Cons.* represents *Consistency*. *Cor.* represents *Correctness*. *Flu.* represents *Fluency* and *Hum.* represents *Humanlikeness.*

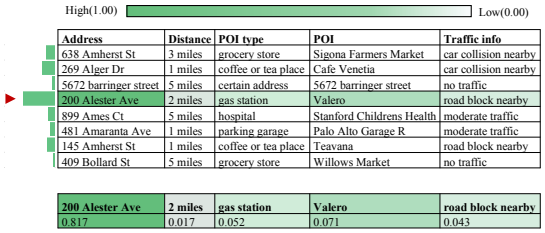


Figure 4: KB score distribution. The distribution is the timestep when generate entity `200_Alester_Ave` for response " *Valero is located at 200_Alester_Ave*"

ence over these entities. Wen et al. (2018) propose the implicit dialogue state representation to query the KB and achieve the promising performance. Different from their modes, we propose the KB-retriever to explicitly query the KB, and the query result is used to filter the irrelevant entities in the dialogue generation to improve the consistency among the output entities.

## 7 Conclusion

In this paper, we propose a novel framework to improve entities consistency by querying KB in two steps. In the first step, inspired by the observation that a response can usually be supported by a single KB row, we introduce the KB retriever to return the most relevant KB row, which is used to filter the irrelevant KB entities and encourage consistent generation. In the second step, we further perform attention mechanism to select the most relevant KB column. Experimental results show the effectiveness of our method. Extensive analysis further confirms the observation and reveal the correlation between the success of KB query and the success of task-oriented dialogue generation.

## Acknowledgments

We thank the anonymous reviewers for their helpful comments and suggestions. This work was supported by the National Natural Science Foundation of China (NSFC) via grant 61976072, 61632011 and 61772153.

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Antoine Bordes and Jason Weston. 2017. Learning end-to-end goal-oriented dialog. In *Proc. of ICLR*.

Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2017. Towards end-to-end reinforcement learning of dialogue agents for information access. In *Proc. of ACL*.

Mihail Eric, Lakshmi Krishnan, Francois Charette, and Christopher D Manning. 2017. Key-value retrieval networks for task-oriented dialogue. In *Proc. of SIGDial*.

Mihail Eric and Christopher Manning. 2017. A copy-augmented sequence-to-sequence architecture gives good performance on task-oriented dialogue. In *Proc. of EACL*.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proc. of ACL*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *ICLR*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Bing Liu and Ian Lane. 2017. An end-to-end trainable neural network model with belief tracking for task-oriented dialog. In *Interspeech 2017*.

Fei Liu and Julien Perez. 2017. Gated end-to-end memory networks. In *Proc. of ACL*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015a. Effective approaches to attention-based neural machine translation. In *Proc. of EMNLP*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015b. Effective approaches to attention-based neural machine translation. In *Proc. of EMNLP*.

Andrea Madotto, Chien-Sheng Wu, and Pascale Fung. 2018. Mem2seq: Effectively incorporating knowledge bases into end-to-end task-oriented dialog systems. In *Proc. of ACL*.

Bonan Min, Ralph Grishman, Li Wan, Chang Wang, and David Gondek. 2013. Distant supervision for relation extraction with an incomplete knowledge base. In *Proc. of ACL*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proc. of ACL*.

Ramesh Nallapati, Bing Xiang, and Bowen Zhou. 2016a. Sequence-to-sequence rnns for text summarization.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016b. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proc. of SIGNLL*.

Dinesh Raghu, Nikhil Gupta, and Mausam. 2019. Disentangling Language and Knowledge in Task-Oriented Dialogs. In *Proc. of NAACL*.

Revanth Reddy, Danish Contractor, Dinesh Raghu, and Sachindra Joshi. 2018. Multi-level memory for task oriented dialogs. *arXiv preprint arXiv:1810.10647*.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proc. of AAAI*.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *NIPS*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Haoyang Wen, Yijia Liu, Wanxiang Che, Libo Qin, and Ting Liu. 2018. Sequence-to-sequence learning for task-oriented dialogue with dialogue state representation. In *Proc. of COLING*.

Tsung-Hsien Wen, Yishu Miao, Phil Blunsom, and Steve Young. 2017a. Latent intention dialogue models. In *ICML*.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gasic, Lina M. Rojas Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017b. A network-based end-to-end trainable task-oriented dialogue system. In *Proc. of EACL*.

Chien-Sheng Wu, Richard Socher, and Caiming Xiong. 2019. Global-to-local memory pointer networks for task-oriented dialogue. *arXiv preprint arXiv:1901.04713*.

Wei Xu, Raphael Hoffmann, Le Zhao, and Ralph Grishman. 2013. Filling knowledge base gaps for distant supervision of relation extraction. In *Proc. of ACL*.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proc. of EMNLP*.

# Building Task-Oriented Visual Dialog Systems Through Alternative Optimization Between Dialog Policy and Language Generation

**Mingyang Zhou** **Josh Arnold** **Zhou Yu**
Department of Computer Science
University of California, Davis
{minzhou, jarnold, joyu}@ucdavis.edu

## Abstract

Reinforcement learning (RL) is an effective approach to learn an optimal dialog policy for task-oriented visual dialog systems. A common practice is to apply RL on a neural sequence-to-sequence (seq2seq) framework with the action space being the output vocabulary in the decoder. However, it is difficult to design a reward function that can achieve a balance between learning an effective policy and generating a natural dialog response. This paper proposes a novel framework that alternatively trains a RL policy for image guessing and a supervised seq2seq model to improve dialog generation quality. We evaluate our framework on the GuessWhich task and the framework achieves the state-of-the-art performance in both task completion and dialog quality.

## 1 Introduction

Visually-grounded conversational artificial intelligence (AI) is an important field that explores the extent intelligent systems are able to hold meaningful conversations regarding visual content. Visually-grounded conversational AI can be applied to a wide range of real-world tasks, including assisting blind people to navigate their surroundings, online recommendation systems, and analysing mass amounts of visual media through natural language. Current approaches to these tasks involve an end-to-end framework that maps the multi-modal context to a deep vector and in order to decode a natural dialog response. This framework can be trained through supervised learning (SL) with the objective to maximize the distribution of the response given a human-human dialog history. Given a large conversational data, the neural end-to-end system can effectively learn to generate coherent and natural language.

While much success has been achieved by applying neural sequence to sequence models to open visual grounding conversation, the visual dialog system also needs to learn an optimal strategy to efficiently accomplish an external goal through natural conversations. To address this issue, various image guessing tasks such us GuessWhich (Chattopadhyay et al., 2017) and GuessWhat (de Vries et al., 2016) are proposed to evaluate a visual-grounded conversational agent on its ability to retrieve visual content via conversing in natural language. To obtain an optimal dialog policy, reinforcement learning (RL) is introduced to enable the neural end-to-end framework to model a more effective action distribution by exploring different dialog strategies. With the application of RL, the visual dialog system can generate more consistent responses and achieve a higher level of engagement in the conversation when compared to a dialog system trained via SL (Das et al., 2017b; Zhang et al., 2017). A typical way to apply RL on a dialog system is to assign a task-related reward to influence the utterance generation process by treating each output word as the action step. A significant limitation of this approach is that it is difficult to achieve an optimal dialog policy that can both effectively complete the external goal and generate natural utterances (Zhao et al., 2019; Das et al., 2017b). As there is no ground truth reference during the RL training stage, the dialog system can only leverage the reward signal when generating the response. However, this approach often deviates from natural language as it is challenging to define a comprehensive reward that considers all aspects of the dialog quality, and in addition, assigns appropriate rewards to the large word vocabulary action space.

In this paper we propose a novel learning curriculum to address the challenge of joint learning between the dialog policy and language generation for task-oriented dialog systems. In our framework, we separate the training of the image re-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 143–153,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

trieval policy from dialog generation by applying RL, with the goal of achieving an optimal policy for guessing the target image at every turn. In addition, we apply a language model objective function to optimize the utterance generator to mitigate language degeneration. We specifically study this framework in the image guessing task GuessWhich, where a conversational agent attempts to guess a target image by asking a series of questions. When compared to state-of-art RL visual dialog systems, our method achieves superior performance in both task-accomplishment and dialog quality.

## 2 Related Work

### 2.1 Visual Dialog System

Visual dialog systems are an emerging area of interdisciplinary research that attracts both the vision and language communities due to the potential applications. Das et al. (2017a) proposed a visual dialog task in which a conversational agent attempts to answer questions regarding an assigned image based on a dialog history. To approach this task, they initially collected data by having two people chat about an image with one person acting as the questioner and the other as the answerer. GuessWhich (Chattopadhyay et al., 2017) extends VisDial with the goal to build an agent that learns how to identify a target image through question and answers. (de Vries et al., 2016) additionally introduced a game in which a series of yes-or-no questions are asked by an agent in order to locate an object in an image. Many researchers approached these tasks via reinforcement learning (RL), with the goal of obtaining an optimal dialog policy. Zhang et al. (2017), for example, designed three rewards with respect to the goals of task achievement, efficiency, and question informativeness, in order to help the agent to achieve an effective question generation policy for GuessWhat game. Das et al. (2017b) applies reinforcement learning in the GuessWhich task and demonstrates a moderate improvement in accuracy compared to the supervised learning approach. Both methods apply RL on a neural end-to-end pipeline to jointly influence the language generation and dialog policy. Due the challenge of designing an appropriate reward for language generation, these methods generate responses that deviate from human natural language. Zhang et al. (2018), proposed an approach involving hierarchical reinforcement learning and state-adaptation techniques that enable the agent to learn an optimal and efficient multi-modal policy. The bottleneck of (Zhang et al., 2018)'s method, however, is that the system response is retrieved from a predefined human-written or system-generated utterance. The number of predefined responses are limited, therefore, this method does not easily generalize to other tasks in real-world settings. We address these limitations by applying RL on a reduced, yet more relevant action space, while optimizing the dialog generator in a supervised fashion. We alternatively optimize policy learning to language generation to combine the two tasks together.

### 2.2 RL on Task-oriented Dialog System

Various RL-based models have been proposed to train task-oriented dialog systems (Williams and Young, 2007). In order to build a traditional modular-based dialog system, researchers first identify the semantic representation, such as the dialog acts and slots in user utterances. Then they accumulate these semantic representations over time to track the dialog state. Finally they apply RL to learn an optimized dialog policy given the dialog state (Raux et al., 2005; Shi and Yu, 2018). Such modular-based dialog systems are effective in narrow task domains, such as searching a bus route schedule and reserving a restaurant through natural language, but they fail to generalize to complex settings where the size of the action space increases. Owing to the development of deep learning, RL on neural sequence-to-sequence models has been explored in more complex dialog domains such as open-domain conversation (Li et al., 2016) and negotiation (Lewis et al., 2017). However, due to the difficulty of assigning appropriate rewards when operating in a large action space, these frameworks cannot generate fluent dialog utterances. Zhao et al. (2019) proposed a novel latent action RL framework to marry the advantage of a module based approach and sequence-to-sequence approach. They learned the optimal dialog policy in a complex task-oriented dialog domain while achieving decent conversation quality. We study the similar issue in a multi-modal task-oriented dialog scenario. We propose an iterative approach to optimize dialog policy using RL methods and system response generation via SL.

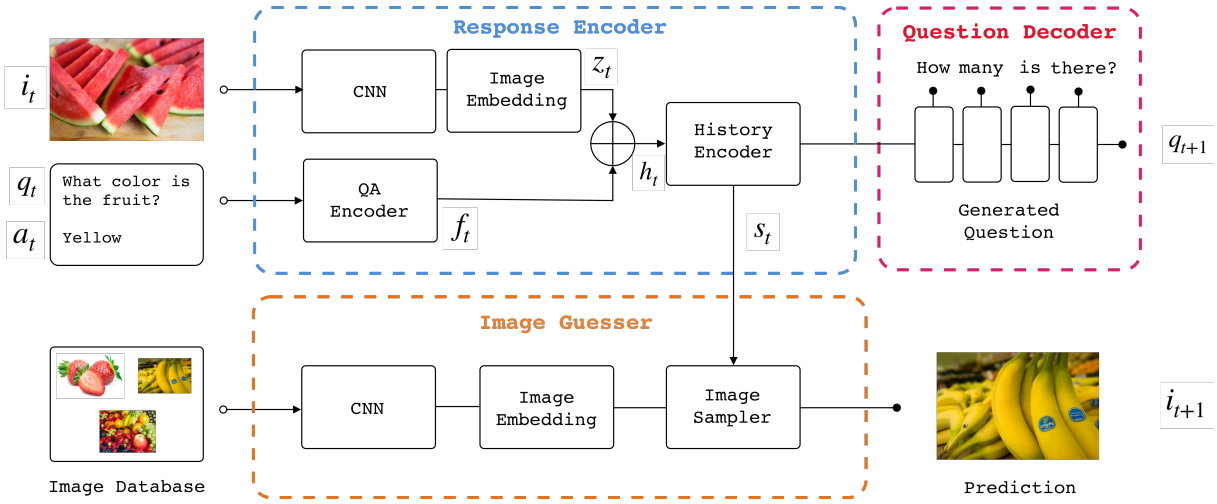


Figure 1: The proposed end-to-end framework of the conversation agent for GuessWhich task-oriented visual dialog task

## 3 Method

### 3.1 Problem Setting

In the GuessWhich problem, we aim to build an agent (Q-Bot) that attempts to guess an image $i_{tgt}$ that another agent (A-Bot) knows by asking it a series of questions. At the beginning of the conversation, the Q-Bot is primed with a short caption $c$ of the target image that is only known by A-Bot. At every round $t$, the Q-Bot generates a question $q_t$ to elicit as much information as possible about the target image and the A-Bot provides an appropriate answer $a_t$ with regard to $q_t$ and the target image. In the end, the agent guesses the target image among a set of images considering the entire conversation.

In addition, our dialog system also guesses a candidate image $i_t$ out of an image database $\mathcal{I} = \{i_k\}_{k=0}^{m}$ at every turn. This action models the process of sequentially updating the visual belief state on the target image based on the latest dialog history. Conditioned on the current guessed image and the prior dialog contexts, the system will generate an optimal question in order to get the maximum information from A-Bot that can strengthen the system's belief on the target image. At the end of the conversation, our Q-Bot will guess the target image based on the multimodal contexts $s_n = (q_{1:n}, a_{1:n}, i_{1:n}, c)$ consisting of the dialog history and the trajectory of guessed images.

### 3.2 Model Architecture

Our Q-Bot is constructed on top of a hierarchical encoder-decoder framework (Serban et al., 2015), which consists of three major components: The **Response Encoder**, the **Question Decoder**, and the **Image Guesser**. We introduce each component as follows:

**Response Encoder** The goal of the response encoder is to append the question $q_t$, the answer $a_t$, and the guessed image $i_t$ received at current round to the dialog history and obtain an updated vector representation of the multimodal context $s_t$. The image $i_t$ is encoded with a pre-trained convolutional neural network VGG-16 (Simonyan and Zisserman, 2015) followed by a linear embedding layer and the image feature vector denoted as $z_t$. For the question and answer pair at the current round $(q_t, a_t)$, we map them to a hidden state vector $f_t$ through the LSTM based *QA Encoder*. We then apply a linear projection on the concatenation of $f_t$ and $z_t$ in order to obtain the multi-modal context vector $h_t$ for the current round. The context vector is then passed through another LSTM encoder: *History Encoder* generates an updated dialog history representation $s_t = \text{HistoryEnc}(h_t, s_{t-1})$. We denote the trainable parameters for Response Encoder as $\theta_e$.

**Question Decoder** The question decoder is a two-layer LSTM network initialized with the most updated dialog history representation vector $s_t$ from the response encoder. It will sequentially sample the words to come up with the next question $q_t$. The learned parameters for question decoder are denoted as $\theta_d$.

**Image Guesser** The Image Guesser attempts to identify the candidate image that best aligns with the dialog history. Given a image database $\mathcal{I} = \{i_k\}_{k=0}^{m}$ where we sample the candidate image, we first extract the image feature representations $\{z_k\}_{k=0}^{m}$ for all candidate images with the convolutional neural network and image embedding layer defined in response encoder. Then, we can sample a candidate image $i_k$ for the current turn based on the euclidean distance $d(z_k, s_t)$ between the image feature of the candidate image and the current dialog history vector. The image with the smallest euclidean distance is selected as the guess $i_t$ at the current round. The associated parameters for image guesser are defined as $\theta_g$.

### 3.3 Training Dialog System

We follow a two-stage training fashion as introduced in many previous end-to-end RL dialog systems (Das et al., 2017b; Zhang et al., 2017; Zhao et al., 2019), where we first pre-train the dialog framework with a supervised objective then apply reinforcement learning to learn an optimal policy to retrieve the target image. The Supervised pre-training is a critical step that facilitates an effective policy exploration for RL training, as it is difficult to explore a complex action space with limited prior knowledge. During RL training, we introduce an alternative learning method between dialog policy exploration and natural utterance generation that addresses the issue of language degeneration in previous RL based visual dialog systems (Das et al., 2017b). We introduce each training method as follows.

#### 3.3.1 Supervised Pre-training

During the supervised pre-training process, we jointly optimize the objective to generate questions and also predict target image features from dialog contexts. The task of question generation is optimized by maximizing the log conditional probability of the next question dependent on a ground truth dialog for every round of the conversation. For the image feature prediction, we minimize the mean square error (MSE) between the target image feature $z_{tgt}$ and the dialog context vector $s_t$ at each round. The joint loss function for supervised pre-training is:

$$\mathcal{L}_{SL}(\theta_r, \theta_d, \theta_g) = \alpha \sum_{t=0}^{n} \log p(q_t|s_t) + \beta \sum_{t=0}^{n} \text{MSE}(z_{tgt}, s_t) \quad (1)$$

Where $\alpha$ and $\beta$ are weights assigned to the objective function of each task in the joint objective function. With SL pre-training process, the dialog system is facilitated with the ability to estimate a visual object and emit a natural language sentence given a dialog context.

#### 3.3.2 Reinforcement Learning on Image Retrieval

In our framework, we treat the sequence of image guess through out the conversation as a partially observable markov decision process and train a policy network via RL to obtain an optimal strategy to retrieve the target image. We formally describe state, policy, action, rewards, and the training procedures in our pipeline.

**State** The dialog states in our framework consist of a combination of multimodal contexts, including the image caption $c$, the dialog history with A-Bot $[q_1, a_2, \ldots, q_t, a_t]$, and the image guessing trajectories $[i_1, i_2, \ldots, i_t]$.

**Policy** The dialog policy $\pi_{\theta_r,\theta_g}(i_t|S_t)$ is a stochastic policy that samples the candidate image to guess from an image set based on the previous dialog histories. The policy is learned from response encoder and image generator which is parameterized via $\theta_r$ and $\theta_g$.

**Action** The full action space is the number of images in the database that we can sample to guess an image. As the pre-trained process already enables the system to approximate a target image feature $z_{tgt}$ with the dialog history representation vector $s_t$, we reduce the action space to the top-K nearest images, $s_t$, based upon the euclidean distance. The probability to sample an image $i_j$ is gained by applying a softmax function over the top-K candidates on their distance to $s_t$: $\pi(j) = \frac{e^{-d_j}}{\sum_{k=1}^{K} e^{-d_k}}$. $d_j$ represents the mean-square-distance between the $j$-th image and the dialog history state vector $s_t$.

**Rewards** We use the ranking percentile of the target image with respect to the dialog history vector $s_t$ as the reward signal to credit the guess at each turn. The goal is to maximize the expectation value of the discounted return $\mathbb{E}[\sum_{t=1}^{n} \gamma^t r_t]$ over the n-round conversation. $r_t$ is the ranking percentile of target image at round $t$ and $\gamma$ is the discounted factor between $(0, 1)$.

**Training Procedure** Inspired from the RL training process on the iterative image retrieval framework (Guo et al., 2018), we apply the policy improvement theory (Sutton and Barto, 1998) to estimate an improved policy $\pi^*(s_t)$ from an existing policy $\pi(s_t)$ obtained from the pre-trained dialog system. Given a dialog state $s_t$ and the action $a_t$ derived from the existing policy, the value estimated by the current policy for taking the action $a_t$ is $Q_\pi(s_t, a_t) = \mathbb{E}[\sum_{t'=t}^{n} \gamma^t r_t]$. To improve this, we explore a different action $a_t^* \neq a_t$ such that a larger policy value $Q_\pi(s_t, a_t^*) > Q_\pi(s_t, a_t)$ estimated with the current policy is achieved. Then we can adjust the existing policy $\pi(s_t)$ to a new policy $\pi^*(s_t)$ that executes that optimal action $a_t^*$ given the current dialog state. The parameters of the policy can be effectively optimized via a cross entropy loss function conditioned on the derived optimal action $a_t^*$:

$$\mathcal{L}_{RL}(\theta_r, \theta_g) = \mathbb{E}[-\sum_{t=1}^{n} \log(\pi_{\theta_r, \theta_g}(a_t^*|s_t))] \quad (2)$$

Compared to the previous RL visual-grounded conversational agent, (Das et al., 2017b), there are several advantages of conducting policy learning on the action level of guessing the image. First, the action space of the top-k nearest neighbors are much smaller compared to the vocabulary size of the output words which reduces the difficulty to explore optimal strategies. Second, only the parameters of response encoder and image generator will be optimized during the RL training stage. The question decoder stays intact so that it is less likely for the dialog system to suffer from language deviation.

#### 3.3.3 Alternating Policy Learning and Language Generation

Although the parameters of the decoder won't be impacted during the RL training stage, the shared response encoder of the dialog context is still optimized with policy learning. The language distribution captured by both the response encoder and question decoder will gradually be differentiated from the original human dialog distribution. To prevent the potential language degeneration behavior, we alternatively optimize the dialog system with a policy learning objective in equation 2 and the language model objective function in equation 1 at every other epoch. It assures the dialog system maintains a good estimation of the human language distribution while also effectively exploring various dialog actions in order to achieve the task of guessing the right image.

## 4 Experiments

### 4.1 AI-AI Image Guessing Game

We evaluate the performance of our task-oriented dialog system by playing the image guessing game, GuessWhich with an automatic answer bot. Our conversational agent's goal is to locate the target image out of the 9,628 test images by interacting with the other player in five conversation exchanges. We evaluate agent on both goal achievement and utterance generation quality using two automatic evaluation metrics Percentile Mean Rank (PMR) and perplexity respectively. PMR estimates how good the agent can rank the target image against other candidates in the test database based on its current dialog state. Perplexity estimates the closeness of the generated response to a reference utterance given a dialog context from the VisDial dataset.

### 4.2 Human-AI Image Guessing Game

To evaluate the ability of our task-oriented dialog system in a realistic conversational scenario, we also make our agent play the image guessing game with human users. The games are set up as 20-image guessing games where the agent attempts to guess a target image outside of a pool of 20 candidate images by asking a human player 5 rounds of questions. The objective of the human player is to play the role of answer bot and answer agent's question with respect to the target image.

In this setting, the performance of the agent on task accomplishment is evaluated by the game win rates. The quality of the dialogs are manually rated on four criteria: fluency, comprehension, diversity and relevance. Fluency defines the naturalness and readability of the generated question in English. Comprehension represents the consistency of the generated question with respect to the previous dialog context. Diversity evaluates the uniqueness

| Image | RL-Q-IG | SL-Q-IG | RL-Q |
|---|---|---|---|
|  | Q: are bananas in bunch?<br>A: yes it's in bunch<br>Q: are they on table?<br>A: yes they are<br>Q: is this in kitchen?<br>A: I'm not sure<br>Q: any people?<br>A: Yes there are several people<br>Q: any other fruits?<br>A: NO, only bananas | Q: what color are planes?<br>A: no planes there<br>Q: are they in city?<br>A: yes it is<br>Q: are they in baskets?<br>A: yes they are<br>Q: any people?<br>A: yes there are<br>Q: animals?<br>A: no animals | Q: is this outside?<br>A: yes it is<br>Q: is there any people?<br>A: yes ...<br>Q: is this in zoo?<br>A: no it is not<br>Q: are there any people?<br>A: yes there are<br>Q: is there any people?<br>A: yes ... |

Table 1: A dialog example with the ground truth caption: **bunches of bananas hang on a wall and arranged for sale.** blue indicates ideal relevant questions and orange indicates less relevant questions.

of the questions generated within one game. Relevance presents how well the asked question is related to the target image and the given caption.

### 4.3 Comparative Models

We compare the performance of our model with state-of-the-art task-oriented visual dialog systems. Meanwhile we also perform an ablation study to evaluate the contribution of different designs in our framework. We introduce each model as follows:

**SL-Q**: The dialog agent from (Das et al., 2017b), which is trained with a joint supervised learning objective function for language generation and image prediction.

**RL-Q**: The dialog agent from (Das et al., 2017b) which is fine-tuned on a trained SL-Q by applying RL to the action space of output word vocabulary.

**SL-Q-IG**: The dialog agent from this framework is build on top of the SL-Q. Compared to SL-Q, SL-Q-IG has an additional image guesser module that makes a guess on target image at every round. SL-Q-IG also has an image encoder which fuses the guessed candidate image into the dialog history tracker. We only train this model with the supervised learning objective introduced equation 1.

**RL-Q-IG**: We use RL method to fine-tune SL-Q-IG. The RL method used is applied on action space of guessing candidate image. We alternate the model to optimize towards dialog policy learning and language generation.

**RL-Q-IG-NA**: We fine-tune SL-Q-IG by applying RL to the action space of guessing candidate image and only optimized with policy learning objective function alone.

**RL-Q-IG-W**: The dialog agent from our framework, which is fine-tuned on a trained SL-Q-IG by applying reinforcement learning on output word vocabulary. It follows the same training procedures as RL-Q to conduct policy learning.

All the SL dialog agents are trained on the VisDial Dataset with the default setting from (Das et al., 2017b) for 40 epochs. The RL dialog agents are then fine-tuned on their corresponding SL dialog agents for another 20 epochs. We evaluate every model on AI-AI image guessing games with the same answer bot, trained on the Visdial Dataset with the objective of visual question answering. We only evaluate RL-Q, SL-Q-IG and RL-Q-IG in human evaluation.

### 4.4 Human-AI Evaluation Implementation

In order to evaluate the effectiveness of the model, we designed three human evaluation tasks. Six college students were recruited to conduct the evaluation. Each student evaluated 100 games using the ground truth captions and 30 games using human generated captions. An additional three evaluators each completed 30 rounds of the relevancy experiment.

**Ground Truth Captions** We generated 100 image guessing games that used the ground truth captions to ensure a consistent amount of information is supplied across all human evaluators. Each game consists of a randomly selected set of 20 images from the VisDial Dataset, with one image randomly chosen as the target. For each game, we test three different models, each twice, resulting in a total of 600 evaluated games from the 100 generated games. We keep the identity of the models anonymous to the evaluator.

During each game, the human evaluator is pre-

sented with a target image the agent is trying to guess. Five rounds of Q&A take place in which the agent asks a question to elicit information and the human evaluator responds with a relevant truthful answer. At the end of each game, the evaluator is asked to rate the conversation on four criteria: fluency, relevance, comprehension and diversity.

**Human Generated Captions** In order to distinguish SL-Q-IG and RL-Q-IG in a more natural setting, we generate an additional 30 games, similar to the previous human evaluation task, except when beginning the game, the evaluator is asked to provide the caption for the target image instead of using the ground truth.

**Relevance Experiment** We noticed that the human evaluators found rating dialogues on the relevance criteria challenging and nuanced. In order to reduce the difficulty of rating dialogues using the relevance criteria, we designed a separate experiment in which, using the conversations obtained from the previous 600 evaluated ground truth games, a human evaluator is presented with three complete conversations side by side at each round. The evaluator then selects the most relevant conversation out of the three that corresponds to the image caption. Each of the three conversations have the same caption, however, correspond to a different model, thus allowing for an effective comparison between the relevancy of each model.

## 5 Results

### 5.1 Results on AI-AI Image Guess Game

**Image Retrieval** It is clear from Fig 2 that our dialog system significantly outperforms the baseline models from (Das et al., 2017b) in terms of PMR on every round of the dialog. PMR estimates how good the agent can rank the target image against other candidates in the test database. The biggest improvement gap is observed between SL-Q-IG and SL-Q. In comparison to SL-Q, SL-Q-IG tracks the additional context from the previously guessed images which leads to a better estimation of the target image. RL-Q-IG has better performance compared to SL-Q-IG in terms of PMR. This suggests that fine-tuning dialog systems with RL can further improve the success of guessing the correct image. The best image retrieval result is achieved by RL-Q-IG-NA, as the objective function of RL-Q-IG-NA is based solely on policy learning without consideration for the dialog generation quality.

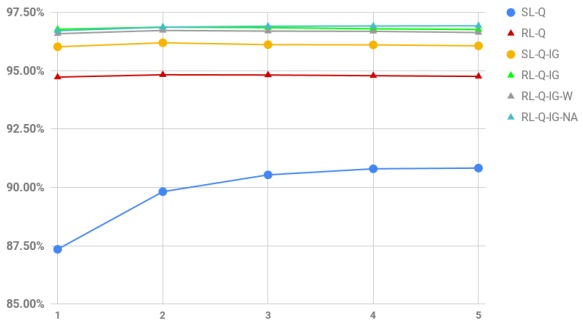


Figure 2: The percentile mean rank (PMR) over the 5-round dialog in the AI-AI image guessing game

| Model | PMR | Perplexity |
|---|---|---|
| SL-Q | 90.07% | 79.49 |
| SL-Q-IG | 96.09% | 61.42 |
| RL-Q | 94.78% | 544.97 |
| RL-Q-IG | 96.81% | **54.66** |
| RL-Q-IG-NA | **96.88%** | 363.88 |
| RL-Q-IG-W | 96.65% | 227.35 |

Table 2: RL-Q-IG-NA performs best in PMR and RL-Q-IG perform best in perplexity

Although our framework achieved an improved image retrieval accuracy, we observed, however, that there is little improvement gained in PMR after additional rounds of conversation. We suspect this is partially due to the fact that images from MSCOCO are composed of a diverse selection objects and background scenes, thus making images easily distinguishable with a detailed caption. In cases where candidate images are visually similar or the given caption is not informative, additional rounds of dialog are necessary to identify the target image.

**Language Generation** We observe a marginal increase of perplexity from SL-Q to RL-Q in Table 2, thus demonstrating that there is a bottleneck when applying RL to improve the language generation. By decoupling the policy learning from the language generation and alternatively optimizing the dialog policy and language model, our RL-Q-IG avoids language deviation while still achieving an optimal dialog policy for the image retrieval task. To further evaluate the contribution from the RL and alternative training curriculum, we conduct two ablation studies. RL-Q-IG-NA is fine-tuned with a policy learning objective that excludes alternatively applying the language model loss. While RL-Q-IG-NA only achieves an in-

| Model | Win | Fluency | Relevance | Comprehension | Diversity |
|---|---|---|---|---|---|
| RL-Q | 59.6 | 4.19 | 3.22 | 2.60 | 2.50 |
| SL-Q-IG | 62.7 | 4.18 | 3.96 | 3.18 | 3.22 |
| **RL-Q-IG** | **67.5** | **4.40** | **4.02** | **3.50** | **3.25** |

Table 3: Evaluation results on the human-AI image guessing game initialized with ground truth captions

| Model | Win | Fluency | Relevance | Comprehension | Diversity |
|---|---|---|---|---|---|
| RL-Q | 29.2 | 4.04 | 2.88 | 2.71 | 2.29 |
| SL-Q-IG | 40.6 | 4.16 | 3.19 | 2.75 | 2.69 |
| **RL-Q-IG** | **67.6** | **4.23** | **3.74** | **3.32** | **3.06** |

Table 4: Evaluation results on the human-AI image guessing game initialized with human generated captions

cremental improvement over the full framework RL-Q-IG in terms of the PMR rate with less than $0.1\%$, it suffers from a dramatic increase of perplexity from 61.42 to 363.88, thus suggesting that alternatively applying the supervised learning objective can prevent the language model from deviating from the human language distribution. We additionally apply policy learning on the question decoder of SL-Q-IG and follow the RL fine-tuning process in (Das et al., 2017b) to train the agent, RL-Q-IG-W. While applying word-level RL enables RL-Q to achieve a moderate improvement over SL-Q in terms of PMR, we did not observe, the same degree of advantage in RL-Q-IG-W over SL-Q-IG. Additionally, RL-Q-IG-W is affected by a marginal increase in perplexity in comparison to the SL pre-trained agent, which approves the drawbacks of applying RL on a large action space in language generation.

### 5.2 Results on Human-AI Image Guess Game

The performance of a dialog agent evaluated with a user simulator does not necessarily reflect its performance on real human (de Vries et al., 2016). We conduct human evaluation on different dialog agents. From the results summarized in Table 3 and Table 4, we observe a consistent optimal performance of our method from conversations with AI agent to conversations with real human. Our RL-Q-IG significantly outperforms the baseline RL agent in all criteria for both settings. RL-Q-IG's advantage over SL-Q-IG is not significant in the game when agents are primed with ground truth image caption. This observation correlates with the result in the Human-AI game, as both RL-Q-IG and SL-Q-IG achieve superior PMR over $96\%$ when presented with the ground truth caption. However, if a human generated caption is given, the performance of the SL pre-trained agent suffers a big drop in all metrics except fluency while our RL agent maintains similar performance. Applying RL to fine-tune the dialog system enables the agent to generate more consistent dialogs in unseen scenarios. We also notice a degradation of the baseline RL agent from its performance with the user simulator, which suggests deviation from natural language is due to the sub-optimal RL training on a large action space.

We conduct a qualitative analysis on the generated dialogs from the three models with human players. Besides a marginal improvement over the RL baseline model and SL pretrained agent in terms of decreased repetition and grammar mistakes, there is a distinct superiority in regards to the relevance to the image caption in the questions generated from our RL agent. For example, in Table 9, we demonstrate the three dialogs generated by RL-Q-IG, SL-Q-IG and RL-Q on one game. Given the image caption *bunches of bananas hang on a wall and arranged for sale.*, RL-Q and SL-Q-IG ask very general questions that are not related to the caption such as *"planes"*, *"zoo"* and *"animals"*. In comparison, our agent asks high-quality questions regarding the caption that covers *"bananas"* and *"fruits"*. These questions help our RL agent obtain useful information to guess the target image. This advantage is also evident from the results of comparative evaluation on the degree of relevance of the questions in Table 5. We credit the positive result to the dialog policy, which explores multiple paths to conduct the conversation. The optimal path will involve a set of questions that obtains the maximum information of the target image such that it can construct the best estimation of the target image.

| Model | Prefered (%) |
|---|---|
| RL-Q | 8.93 |
| SL-Q-ImGuess | 39.90 |
| **RL-Q-IG** | **51.20** |

Table 5: Results on comparative evaluation of relevance on the human-AI image guessing dialogs

## 6 Conclusion and Future Work

We present a novel framework for building a task-oriented visual dialog system. We model the agent to simultaneously optimize two actions: guessing the image and generating effective questions. We achieve this simultaneous optimization through alternatively applying reinforcement learning to obtain an effective image guessing policy, whilst also applying supervised learning to enhance the quality of generated questions. By decoupling the policy learning from language generation, we overcome language degeneration in the word-level reinforcement learning framework. Both analytical and human evaluation suggests our proposed framework leads to a higher task completion rate and an improved dialog quality.

In the future, we plan to collect a fashion retrieval visual dialog dataset which simulates a realistic application for multi-modal dialog systems. To address the limitation of a high image retrieval rate with just the use of captions from the VisDial dataset, we plan to format a challenging candidate image pool in which images are visually similar to each other. This will incentivize the dialog system to conduct multiple rounds of dialog in order to retrieve the target image successfully. Furthermore, we will explore additional task-oriented settings where we can decouple task accomplishment from language generation to evaluate the extent our framework can generalize to other conversational tasks.

## References


Prithvijit Chattopadhyay, Deshraj Yadav, Viraj Prabhu, Arjun Chandrasekaran, Abhishek Das, Stefan Lee, Dhruv Batra, and Devi Parikh. 2017. Evaluating visual conversational agents via cooperative human-ai games. *CoRR*, abs/1708.05122.

Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José M.F. Moura, Devi Parikh, and Dhruv Batra. 2017a. Visual Dialog. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Abhishek Das, Satwik Kottur, José M.F. Moura, Stefan Lee, and Dhruv Batra. 2017b. Learning cooperative visual dialog agents with deep reinforcement learning. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Xiaoxiao Guo, Hui Wu, Yu Cheng, Steven Rennie, and Rogério Schmidt Feris. 2018. Dialog-based interactive image retrieval. *CoRR*, abs/1805.00145.

Mike Lewis, Denis Yarats, Yann N. Dauphin, Devi Parikh, and Dhruv Batra. 2017. Deal or no deal? end-to-end learning for negotiation dialogues. *CoRR*, abs/1706.05125.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016. Deep reinforcement learning for dialogue generation. *CoRR*, abs/1606.01541.

Antoine Raux, Brian Langner, Dan Bohus, Alan W Black, and Maxine Eskenazi. 2005. Lets go public! taking a spoken dialog system to the real world. In *in Proc. of Interspeech 2005*.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. 2015. Hierarchical neural network generative models for movie dialogues. *CoRR*, abs/1507.04808.

Weiyan Shi and Zhou Yu. 2018. Sentiment adaptive end-to-end dialog systems. *CoRR*, abs/1804.10731.

K. Simonyan and A. Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*.

Richard S. Sutton and Andrew G. Barto. 1998. *Introduction to Reinforcement Learning*, 1st edition. MIT Press, Cambridge, MA, USA.

Harm de Vries, Florian Strub, Sarath Chandar, Olivier Pietquin, Hugo Larochelle, and Aaron C. Courville. 2016. Guesswhat?! visual object discovery through multi-modal dialogue. *CoRR*, abs/1611.08481.

Jason D. Williams and Steve Young. 2007. Partially observable markov decision processes for spoken dialog systems. *Comput. Speech Lang.*, 21(2):393–422.

Jiaping Zhang, Tiancheng Zhao, and Zhou Yu. 2018. Multimodal hierarchical reinforcement learning policy for task-oriented visual dialog. *CoRR*, abs/1805.03257.

Junjie Zhang, Qi Wu, Chunhua Shen, Jian Zhang, Jianfeng Lu, and Anton van den Hengel. 2017. Asking the difficult questions: Goal-oriented visual question generation via intermediate rewards. *CoRR*, abs/1711.07614.

Tiancheng Zhao, Kaige Xie, and Maxine Eskénazi. 2019. Rethinking action spaces for reinforcement learning in end-to-end dialog agents with latent variable models. *CoRR*, abs/1902.08858.

## A Human Evaluation Interface

We demonstrate our web interface for our Human-AI guessing game in Fig 3. At the beginning of every game the human player is presented with a random sampled target image. For the trained Q-bot, 20 images are sampled as the candidate pool with the objective of retrieving the target image. At the end of the conversation, the Q-Bot's guess is revealed to the human player followed by a request to evaluate the quality of the conversation in the four aspects introduced in section 4.4 on a scale from 1 to 5.

Our human relevancy evaluation interface is demonstrated in Fig 4. We present side-by-side questions generated in the same game from each of the three different models and ask the human evaluators to pick the conversation that is most semantically relevant to the target image caption.

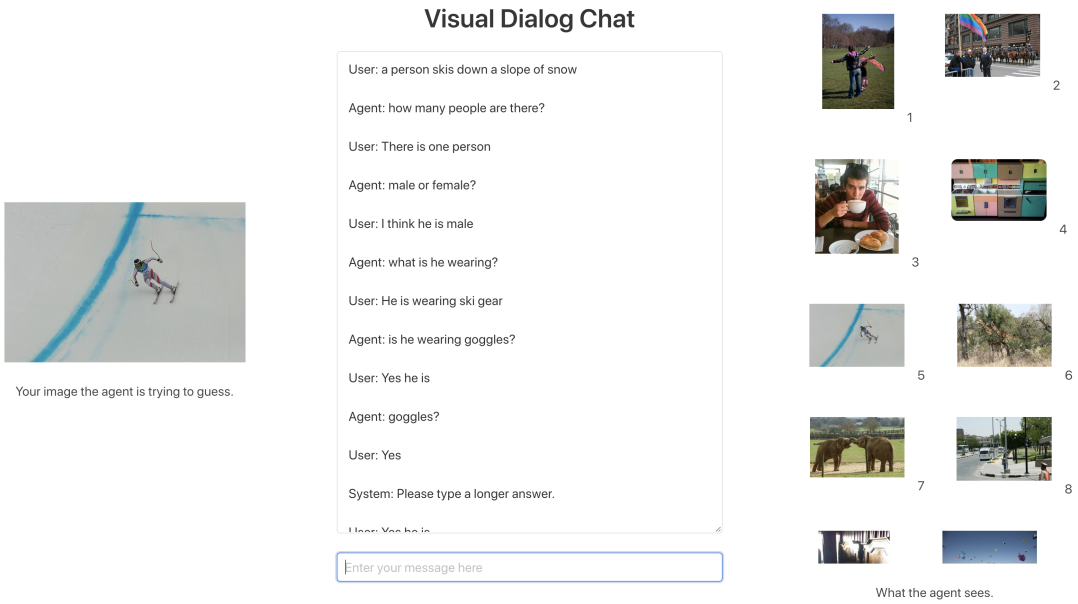


Figure 3: The web interface for human-AI guessing game. The left image is a target image randomly sampled from (Das et al., 2017a). The center section is a chat platform for human to communicate with a trained Q-Bot. On the right hand side are the 20 candidate images sampled for the Q-Bot to retrieve the target image.

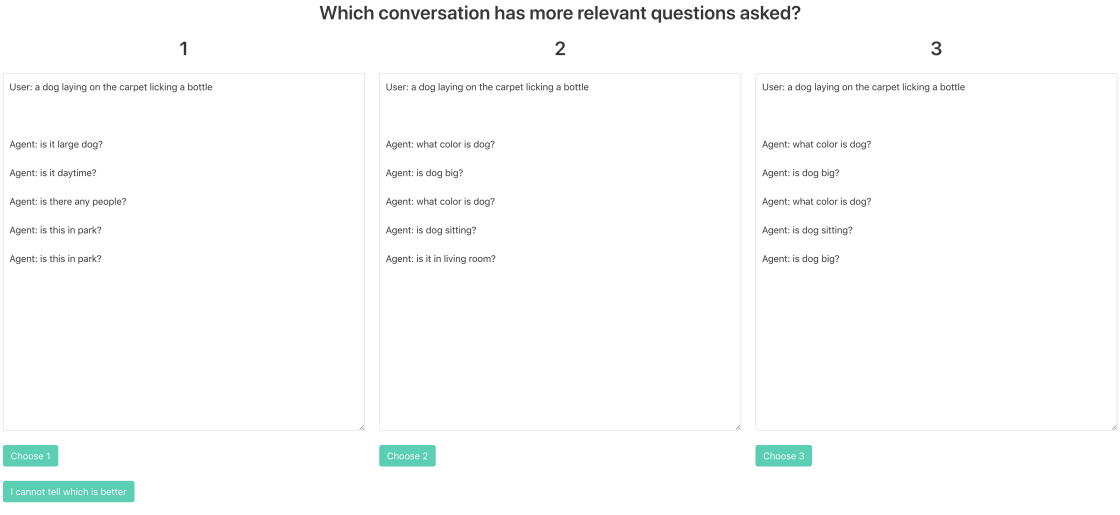


Figure 4: The user interface for human relevancy experiment. The conversation from the three different models on the same game setting are presented in the three boxes. On top of every conversation is the caption for the target image. Followed by that are the five questions asked by the Q-Bot in that game.

## B Qualitative Examples

| Image | RL-Q-ImGuess | SL-Q-ImGuess | RL-Q |
|---|---|---|---|
|  | Q: what color is vase?<br>A: it is transparent<br>Q: is it on table?<br>A: yes it is<br>Q: is it in kitchen?<br>A: I do not know<br>Q: any people?<br>A: no, there is no people<br>Q: what color is the table?<br>A: yes there is | Q: what color is vase?<br>A: it is brown<br>Q: is it in vase?<br>yes it is<br>Q: is it in vase?<br>A: yes it is<br>Q: is it in vase?<br>A: yes it is<br>Q: is it in vase?<br>A: yes it is | Q: what color is table?<br>A: it is orange<br>Q: is there anything else on table?<br>A: yes there are flowers<br>Q: what color is table?<br>A: it is orange<br>Q: is there anything else on table?<br>A: yes there are<br>Q: is there anything else on table?<br>A: yes there is |

Table 6: A dialog example with the human generated caption: **there is a plant in a vase and cookies.** Blue highlights diverse questions and orange indicates poor diversity.

| Image | RL-Q-ImGuess | SL-Q-ImGuess | RL-Q |
|---|---|---|---|
|  | Q: are men old?<br>A: No they are not<br>Q: are they in uniform?<br>A: I'm not sure<br>Q: is it sunny?<br>A: yes it is<br>Q: are they on road?<br>A: yes they are<br>Q: are they in parking lot?<br>A: No they're not | Q: how old are men?<br>A: 30 years old<br>Q: are they in city?<br>A: yes they are<br>Q: is it sunny?<br>A: yes it is<br>Q: any other people?<br>A: no other people<br>Q: animals?<br>A: no other animals | Q: what color is hat?<br>A: there is no hat<br>Q: is it sunny?<br>A: it seems yes<br>Q: is this in city?<br>A: it is in city<br>Q: are there any people in picture?<br>A: there are two people<br>Q: is this in home or restaurant?<br>A: it is outside |

Table 7: A dialog example with the human generated caption: **two men in formal wear standing next to a monster truck.** Blue highlights ideal relevant questions and orange indicates less relevant questions.

| Image | RL-Q-ImGuess | SL-Q-ImGuess | RL-Q |
|---|---|---|---|
|  | Q: how many horses are there?<br>A: there are no horses<br>Q: are they in wild?<br>A: no, they are in village<br>Q: is there fence?<br>A: yes, there is fence<br>Q: any people?<br>A: yes, a person<br>Q: animals?<br>A: yes, several bulls | Q: how many horses are there?<br>A: There are no horses, only bulls<br>Q: what color are horses?<br>A: they are white and black<br>Q: are they in field?<br>A: Yes they are<br>Q: is it sunny?<br>A: Yes it is<br>Q: is it sunny?<br>A: Yes it is | Q: is this in park?<br>A: maybe it is in park<br>Q: are there any people in picture?<br>A: yes there is one picture<br>Q: is this in home or restaurant?<br>A: it is in neither of those<br>Q: are there any people in picture?<br>A: yes there is one person<br>Q: is this in restaurant?<br>A: no it is not |

Table 8: A dialog example with the human generated caption: **an image of running with the bulls outside.** Blue indicates ideal relevant questions and orange highlights irrelevant/repeating questions.

| Image | RL-Q-ImGuess | SL-Q-ImGuess | RL-Q |
|---|---|---|---|
|  | Q: what color is kite?<br>A: It is orange<br>Q: is it sunny?<br>A: Yes it is<br>Q: any other people?<br>A: Yes there are two people<br>Q: any animals?<br>A: No there is not<br>Q: any trees?<br>A: Yes, there are several trees | Q: what color is horse?<br>A: there is no horse<br>Q: how old is man?<br>A: he is young<br>Q: is he wearing hat?<br>A: no, he is not wearing hat<br>Q: what color is horse?<br>A: there is no horse<br>Q: is it sunny?<br>A: yes it is | Q: is there only 1 person?<br>A: no there is not<br>Q: is this in park?<br>A: yes it is<br>Q: are there any people in picture?<br>A: yes there are<br>Q: is there any other food on table?<br>A: no food on the table<br>Q: is there anything else on table?<br>A: nothing . . |

Table 9: A dialog example with the human generated caption: **a man holding a kite while a girl tries to fly it.** Blue indicates ideal relevant questions and orange indicates poor relevance.

# DialogueGCN: A Graph Convolutional Neural Network for Emotion Recognition in Conversation

**Deepanway Ghosal**[†], **Navonil Majumder**[‡], **Soujanya Poria**[†*], **Niyati Chhaya**[▽] **and Alexander Gelbukh**[‡]
[†] Singapore University of Technology and Design, Singapore
[‡]Instituto Politécnico Nacional, CIC, Mexico
[▽] Adobe Research, India

{1004721@mymail.,sporia@}sutd.edu.sg, navo@nlp.cic.ipn.mx, nchhaya@adobe.com, gelbukh@gelbukh.com

## Abstract

Emotion recognition in conversation (ERC) has received much attention, lately, from researchers due to its potential widespread applications in diverse areas, such as health-care, education, and human resources. In this paper, we present Dialogue Graph Convolutional Network (DialogueGCN), a graph neural network based approach to ERC. We leverage self and inter-speaker dependency of the interlocutors to model conversational context for emotion recognition. Through the graph network, DialogueGCN addresses context propagation issues present in the current RNN-based methods. We empirically show that this method alleviates such issues, while outperforming the current state of the art on a number of benchmark emotion classification datasets.

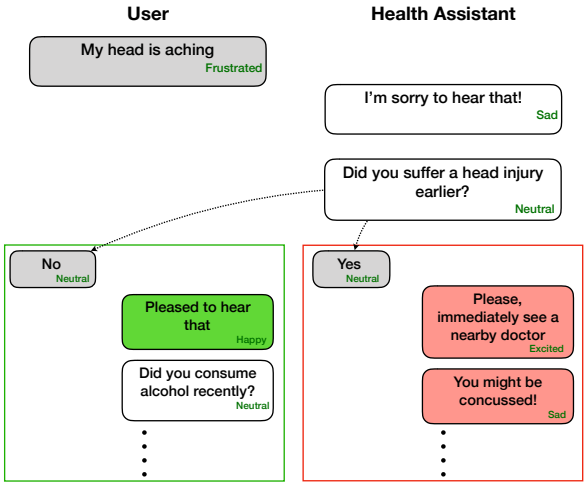


Figure 1: Illustration of an affective conversation where the emotion depends on the context. Health assistant understands affective state of the user in order to generate affective and empathetic responses.

## 1 Introduction

Emotion recognition has remained an active research topic for decades (K. D'Mello et al., 2006; Busso et al., 2008; Strapparava and Mihalcea, 2010). However, the recent proliferation of open conversational data on social media platforms, such as Facebook, Twitter, Youtube, and Reddit, has warranted serious attention (Poria et al., 2019b; Majumder et al., 2019; Huang et al., 2019) from researchers towards emotion recognition in conversation (ERC). ERC is also undeniably important in affective dialogue systems (as shown in Fig. 1) where bots understand users' emotions and sentiment to generate emotionally coherent and empathetic responses.

Recent works on ERC process the constituent utterances of a dialogue in sequence, with a recurrent neural network (RNN). Such a scheme is illustrated in Fig. 2 (Poria et al., 2019b), that relies on propagating contextual and sequential information to the utterances. Hence, we feed the conversation to a bidirectional gated recurrent unit (GRU) (Chung et al., 2014). However, like most of the current models, we also ignore intent modelling, topic, and personality due to lack of labelling on those aspects in the benchmark datasets. In theory, RNNs like long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) and GRU should propagate long-term contextual information. However, in practice it is not always the case (Bradbury et al., 2017). This affects the efficacy of RNN-based models in various tasks, including ERC.

To mitigate this issue, some variants of the state-of-the-art method, DialogueRNN (Majumder et al., 2019), employ attention mechanism that pools information from entirety or part of the conversation per target utterance. However, this pooling mechanism does not consider speaker information of the utterances and the relative position of other utterances from the target utterance. Speaker information is necessary for mod-

* Corresponding author



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 154–164, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

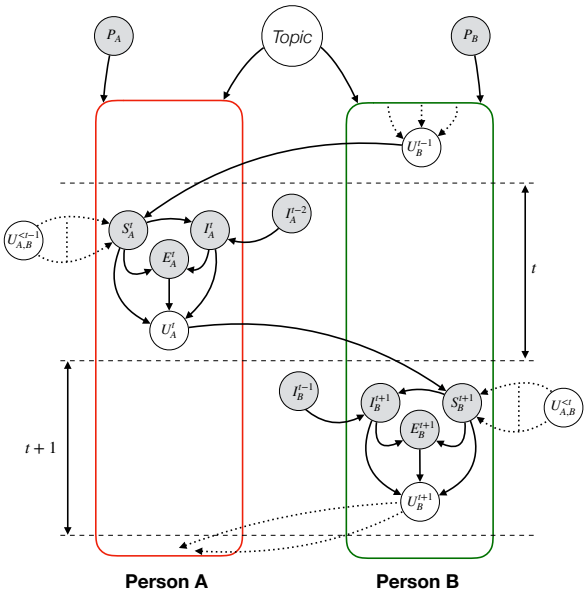


Figure 2: Interaction among different controlling variables during a dyadic conversation between persons A and B. Grey and white circles represent hidden and observed variables, respectively. $P$ represents personality, $U$ represents utterance, $S$ represents interlocutor state, $I$ represents interlocutor intent, $E$ represents emotion and *Topic* represents topic of the conversation. This can easily be extended to multi-party conversations.

elling inter-speaker dependency, which enables the model to understand how a speaker coerces emotional change in other speakers. Similarly, by extension, intra-speaker or self-dependency aids the model with the understanding of emotional inertia of individual speakers, where the speakers resist the change of their own emotion against external influence. On the other hand, consideration of relative position of target and context utterances decides how past utterances influence future utterances and vice versa. While past utterances influencing future utterances is natural, the converse may help the model fill in some relevant missing information, that is part of the speaker's background knowledge but explicitly appears in the conversation in the future. We leverage these two factors by modelling conversation using a directed graph. The nodes in the graph represent individual utterances. The edges between a pair of nodes/utterances represent the dependency between the speakers of those utterances, along with their relative positions in the conversation. By feeding this graph to a graph convolution network (GCN) (Defferrard et al., 2016), consisting of two consecutive convolution operations, we propagate contextual information among distant utterances. We surmise that these representations hold richer context relevant to emotion than DialogueRNN. This is empirically shown in Section 5.

The remainder of the paper is organized as follows — Section 2 briefly discusses the relevant and related works on ERC; Section 3 elaborates the method; Section 4 lays out the experiments; Section 5 shows and interprets the experimental results; and finally, Section 6 concludes the paper.

## 2 Related Work

Emotion recognition in conversation is a popular research area in natural language processing (Kratzwald et al., 2018; Colneriĉ and Demsar, 2018) because of its potential applications in a wide area of systems, including opinion mining, health-care, recommender systems, education, etc.

However, emotion recognition in conversation has attracted attention from researchers only in the past few years due to the increase in availability of open-sourced conversational datasets (Chen et al., 2018; Zhou et al., 2018; Poria et al., 2019a). A number of models has also been proposed for emotion recognition in multimodal data i.e. datasets with textual, acoustic and visual information. Some of the important works include (Poria et al., 2017; Chen et al., 2017; Zadeh et al., 2018a,b; Hazarika et al., 2018a,b), where mainly deep learning-based techniques have been employed for emotion (and sentiment) recognition in conversation, in only textual and multimodal settings. The current state-of-the-art model in emotion recognition in conversation is (Majumder et al., 2019), where authors introduced a party state and global state based recurrent model for modelling the emotional dynamics.

Graph neural networks have also been very popular recently and have been applied to semi-supervised learning, entity classification, link prediction, large scale knowledge base modelling, and a number of other problems (Kipf and Welling, 2016; Schlichtkrull et al., 2018; Bruna et al., 2013). Early work on graph neural networks include (Scarselli et al., 2008). Our graph model is closely related to the graph relational modelling work introduced in (Schlichtkrull et al., 2018).

## 3 Methodology

One of the most prominent strategies for emotion recognition in conversations is contextual mod-

elling. We identify two major types of context in ERC – sequential context and speaker-level context. Following Poria et al. (2017), we model these two types of context through the neighbouring utterances, per target utterance.

Computational modeling of context should also consider emotional dynamics of the interlocutors in a conversation. Emotional dynamics is typically subjected to two major factors in both dyadic and multiparty conversational systems — inter-speaker dependency and self-dependency. Inter-speaker dependency refers to the emotional influence that counterparts produce in a speaker. This dependency is closely related to the fact that speakers tend to mirror their counterparts to build rapport during the course of a dialogue (Navaretta et al., 2016). However, it must be taken into account, that not all participants are going to affect the speaker in identical way. Each participant generally affects each other participants in unique ways. In contrast, self-dependency, or emotional inertia, deals with the aspect of emotional influence that speakers have on themselves during conversations. Participants in a conversation are likely to stick to their own emotional state due to their emotional inertia, unless the counterparts invoke a change. Thus, there is always a major interplay between the inter-speaker dependency and self-dependency with respect to the emotional dynamics in the conversation.

We surmise that combining these two distinct yet related contextual information schemes (sequential encoding and speaker level encoding) would create enhanced context representation leading to better understanding of emotional dynamics in conversational systems.

### 3.1 Problem Definition

Let there be $M$ speakers/parties $p_1, p_2, \ldots, p_M$ in a conversation. The task is to predict the emotion labels (*happy*, *sad*, *neutral*, *angry*, *excited*, *frustrated*, *disgust*, and *fear*) of the constituent utterances $u_1, u_2, \ldots, u_N$, where utterance $u_i$ is uttered by speaker $p_{s(u_i)}$, while $s$ being the mapping between utterance and index of its corresponding speaker. We also represent $u_i \in \mathbb{R}^{D_m}$ as the feature representation of the utterance, obtained using the feature extraction process described below.

### 3.2 Context Independent Utterance-Level Feature Extraction

A convolutional neural network (Kim, 2014) is used to extract textual features from the transcript of the utterances. We use a single convolutional layer followed by max-pooling and a fully connected layer to obtain the feature representations for the utterances. The input to this network is the 300 dimensional pretrained 840B GloVe vectors (Pennington et al., 2014). We use filters of size 3, 4 and 5 with 50 feature maps in each. The convoluted features are then max-pooled with a window size of 2 followed by the ReLU activation (Nair and Hinton, 2010). These are then concatenated and fed to a 100 dimensional fully connected layer, whose activations form the representation of the utterance. This network is trained at utterance level with the emotion labels.

### 3.3 Model

We now present our Dialogue Graph Convolutional Network (DialogueGCN[1]) framework for emotion recognition in conversational setups. DialogueGCN consists of three integral components — Sequential Context Encoder, Speaker-Level Context Encoder, and Emotion Classifier. An overall architecture of the proposed framework is illustrated in Fig. 3.

#### 3.3.1 Sequential Context Encoder

Since, conversations are sequential by nature, contextual information flows along that sequence. We feed the conversation to a bidirectional gated recurrent unit (GRU) to capture this contextual information: $g_i = \overleftrightarrow{GRU_S}(g_{i(+,-)1}, u_i)$, for i = 1, 2,..., N, where $u_i$ and $g_i$ are context-independent and sequential context-aware utterance representations, respectively.

Since, the utterances are encoded irrespective of its speaker, this initial encoding scheme is speaker agnostic, as opposed to the state of the art, DialogueRNN (Majumder et al., 2019).

#### 3.3.2 Speaker-Level Context Encoder

We propose the Speaker-Level Context Encoder module in the form of a graphical network to capture speaker dependent contextual information in a conversation. Effectively modelling speaker level context requires capturing the inter-dependency

[1]Implementation available at https://github.com/SenticNet/conv-emotion

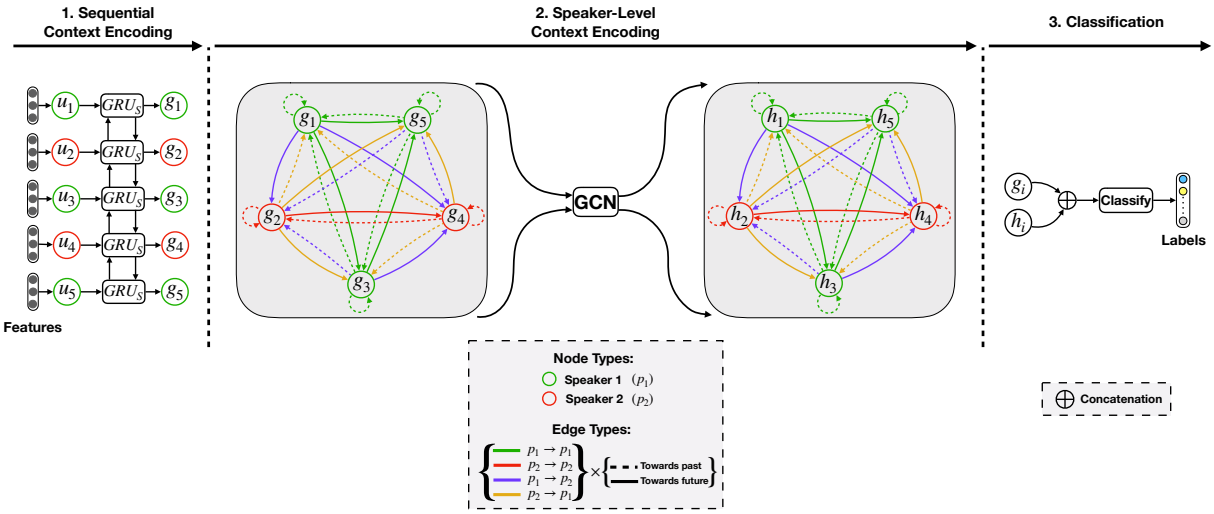


Figure 3: Overview of DialogueGCN, congruent to the illustration in Table 1.

and self-dependency among participants. We design a directed graph from the sequentially encoded utterances to capture this interaction between the participants. Furthermore, we propose a local neighbourhood based convolutional feature transformation process to create the enriched speaker-level contextually encoded features. The framework is detailed here.

First, we introduce the following notation: a conversation having $N$ utterances is represented as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{R}, \mathcal{W})$, with vertices/nodes $v_i \in \mathcal{V}$, labeled edges (relations) $r_{ij} \in \mathcal{E}$ where $r \in \mathcal{R}$ is the relation type of the edge between $v_i$ and $v_j$ and $\alpha_{ij}$ is the weight of the labeled edge $r_{ij}$, with $0 \leqslant \alpha_{ij} \leqslant 1$, where $\alpha_{ij} \in \mathcal{W}$ and $i, j \in [1, 2, ..., N]$.

**Graph Construction:** The graph is constructed from the utterances in the following way,

**Vertices:** Each utterance in the conversation is represented as a vertex $v_i \in \mathcal{V}$ in $\mathcal{G}$. Each vertex $v_i$ is initialized with the corresponding sequentially encoded feature vector $g_i$, for all $i \in [1, 2, ..., N]$. We denote this vector as the vertex feature. Vertex features are subject to change downstream, when the neighbourhood based transformation process is applied to encode speaker-level context.

**Edges:** Construction of the edges $\mathcal{E}$ depends on the context to be modeled. For instance, if we hypothesize that each utterance (vertex) is contextually dependent on all the other utterances in a conversation (when encoding speaker level information), then a fully connected graph would be constructed. That is each vertex is connected to all the other vertices (including itself) with an edge. However, this results in $O(N^2)$ number of edges, which is computationally very expensive for graphs with large number of vertices. A more practical solution is to construct the edges by keeping a past context window size of $p$ and a future context window size of $f$. In this scenario, each utterance vertex $v_i$ has an edge with the immediate $p$ utterances of the past: $v_{i-1}, v_{i-2}, ..v_{i-p}$, $f$ utterances of the future: $v_{i+1}, v_{i+2}, ..v_{i+f}$ and itself: $v_i$. For all our experiments in this paper, we consider a past context window size of 10 and future context window size of 10.

As the graph is directed, two vertices can have edges in both directions with different relations.

**Edge Weights:** The edge weights are set using a similarity based attention module. The attention function is computed in a way such that, for each vertex, the incoming set of edges has a sum total weight of 1. Considering a past context window size of $p$ and a future context window size of $f$, the weights are calculated as follows,

$$\alpha_{ij} = \text{softmax}(g_i^T W_e [g_{i-p}, \ldots, g_{i+f}]), \quad \text{for } j = i-p, \ldots, i+f. \tag{1}$$

This ensures that, vertex $v_i$ which has incoming edges with vertices $v_{i-p}, \ldots, v_{i+f}$ (as speaker-level context) receives a total weight contribution of 1.

**Relations:** The relation $r$ of an edge $r_{ij}$ is set depending upon two aspects:

***Speaker dependency*** — The relation depends on both the speakers of the constituting vertices: $p_{s(u_i)}$ (speaker of $v_i$) and $p_{s(u_j)}$ (speaker of $v_j$).

***Temporal dependency*** — The relation also de-

pends upon the relative position of occurrence of $u_i$ and $u_j$ in the conversation: whether $u_i$ is uttered before $u_j$ or after. If there are $M$ distinct speakers in a conversation, there can be a maximum of $M$ (speaker of $u_i$) $*M$ (speaker of $u_j$) $*2$ ($u_i$ occurs before $u_j$ or after) = $2M^2$ distinct relation types $r$ in the graph $\mathcal{G}$.

Each speaker in a conversation is uniquely affected by each other speaker, hence we hypothesize that explicit declaration of such relational edges in the graph would help in capturing the inter-dependency and self-dependency among the speakers in play, which in succession would facilitate speaker-level context encoding.

As an illustration, let two parties $p_1, p_2$ participate in a dyadic conversation having 5 utterances, where $u_1, u_3, u_5$ are uttered by $p_1$ and $u_2, u_4$ are uttered by $p_2$. If we consider a fully connected graph, the edges and relations will be constructed as shown in Table 1.

**Feature Transformation:** We now describe the methodology to transform the sequentially encoded features using the graph network. The vertex feature vectors ($g_i$) are initially speaker independent and thereafter transformed into a speaker dependent feature vector using a two-step graph convolution process. Both of these transformations can be understood as special cases of a basic differentiable message passing method (Gilmer et al., 2017).

In the first step, a new feature vector $h_i^{(1)}$ is computed for vertex $v_i$ by aggregating local neighbourhood information (in this case neighbour utterances specified by the past and future context window size) using the relation specific transformation inspired from (Schlichtkrull et al., 2018):

$$h_i^{(1)} = \sigma\Big(\sum_{r\in\mathcal{R}}\sum_{j\in N_i^r}\frac{\alpha_{ij}}{c_{i,r}}W_r^{(1)}g_j + \alpha_{ii}W_0^{(1)}g_i\Big), \quad \text{for } i = 1,2,\ldots,N, \tag{2}$$

where, $\alpha_{ij}$ and $\alpha_{ii}$ are the edge weights, $N_i^r$ denotes the neighbouring indices of vertex $i$ under relation $r \in \mathcal{R}$. $c_{i,r}$ is a problem specific normalization constant which either can be set in advance, such that, $c_{i,r} = |N_i^r|$, or can be automatically learned in a gradient based learning setup. $\sigma$ is an activation function such as ReLU, $W_r^{(1)}$ and $W_0^{(1)}$ are learnable parameters of the transformation. In the second step, another local neighbourhood based transformation is applied over the output of the first step,

$$h_i^{(2)} = \sigma\Big(\sum_{j\in N_i^r} W^{(2)}h_j^{(1)} + W_0^{(2)}h_i^{(1)}\Big), \quad \text{for } i = 1,2,\ldots,N, \tag{3}$$

where, $W^{(2)}$ and $W_0^{(2)}$ are parameters of these transformation and $\sigma$ is the activation function.

This stack of transformations, Eqs. (2) and (3), effectively accumulates normalized sum of the local neighbourhood (features of the neighbours) i.e. the neighbourhood speaker information for each utterance in the graph. The self connection ensures self dependent feature transformation.

**Emotion Classifier:** The contextually encoded feature vectors $g_i$ (from sequential encoder) and $h_i^{(2)}$ (from speaker-level encoder) are concatenated and a similarity-based attention mechanism is applied to obtain the final utterance representation:

$$h_i = [g_i, h_i^{(2)}], \tag{4}$$

$$\beta_i = \text{softmax}(h_i^T W_\beta[h_1, h_2 \ldots, h_N]), \tag{5}$$

$$\tilde{h}_i = \beta_i[h_1, h_2, \ldots, h_N]^T. \tag{6}$$

Finally, the utterance is classified using a fully-connected network:

$$l_i = \text{ReLU}(W_l\tilde{h}_i + b_l), \tag{7}$$

$$\mathcal{P}_i = \text{softmax}(W_{smax}l_i + b_{smax}), \tag{8}$$

$$\hat{y}_i = \underset{k}{\text{argmax}}(\mathcal{P}_i[k]). \tag{9}$$

| Relation | $p_s(u_i), p_s(u_j)$ | $i < j$ | $(i, j)$ |
|---|---|---|---|
| 1 | $p_1, p_1$ | Yes | (1,3), (1,5), (3,5) |
| 2 | $p_1, p_1$ | No | (1,1), (3,1), (3,3) (5,1), (5,3), (5,5) |
| 3 | $p_2, p_2$ | Yes | (2,4) |
| 4 | $p_2, p_2$ | No | (2,2), (4,2), (4, 4) |
| 5 | $p_1, p_2$ | Yes | (1,2), (1,4), (3,4) |
| 6 | $p_1, p_2$ | No | (3,2), (5,2), (5,4) |
| 7 | $p_2, p_1$ | Yes | (2,3), (2,5), (4,5) |
| 8 | $p_2, p_1$ | No | (2,1), (4,1), (4,3) |

Table 1: $p_s(u_i)$ and $p_s(u_j)$ denotes the speaker of utterances $u_i$ and $u_j$. 2 distinct speakers in the conversation implies $2 * M^2 = 2 * 2^2 = 8$ distinct relation types. The rightmost column denotes the indices of the vertices of the constituting edge which has the relation type indicated by the leftmost column.

**Training Setup:** We use categorical cross-entropy along with L2-regularization as the measure of loss ($L$) during training:

$$L = -\frac{1}{\sum_{s=1}^{N} c(s)} \sum_{i=1}^{N} \sum_{j=1}^{c(i)} \log \mathcal{P}_{i,j}[y_{i,j}] + \lambda \|\theta\|_2 , \tag{10}$$

where $N$ is the number of samples/dialogues, $c(i)$ is the number of utterances in sample $i$, $\mathcal{P}_{i,j}$ is the probability distribution of emotion labels for utterance $j$ of dialogue $i$, $y_{i,j}$ is the expected class label of utterance $j$ of dialogue $i$, $\lambda$ is the L2-regularizer weight, and $\theta$ is the set of all trainable parameters.

We used stochastic gradient descent based Adam (Kingma and Ba, 2014) optimizer to train our network. Hyperparameters were optimized using grid search.

## 4 Experimental Setting

### 4.1 Datasets Used

We evaluate our DialogueGCN model on three benchmark datasets — IEMOCAP (Busso et al., 2008), AVEC (Schuller et al., 2012), and MELD (Poria et al., 2019a). All these three datasets are multimodal datasets containing textual, visual and acoustic information for every utterance of each conversation. However, in this work we focus on conversational emotion recognition only from the textual information. Multimodal emotion recognition is outside the scope of this paper, and is left as future work.

**IEMOCAP** (Busso et al., 2008) dataset contains videos of two-way conversations of ten unique speakers, where only the first eight speakers from session one to four belong to the trainset. Each video contains a single dyadic dialogue, segmented into utterances. The utterances are annotated with one of six emotion labels, which are happy, sad, neutral, angry, excited, and frustrated.

**AVEC** (Schuller et al., 2012) dataset is a modification of SEMAINE database (McKeown et al., 2012) containing interactions between humans and artificially intelligent agents. Each utterance of a dialogue is annotated with four real valued affective attributes: valence ($[-1, 1]$), arousal ($[-1, 1]$), expectancy ($[-1, 1]$), and power ($[0, \infty)$). The annotations are available every 0.2 seconds in the original database. However, in order to adapt the annotations to our need of utterance-level annotation, we averaged the attributes over the span of an utterance.

**MELD** (Poria et al., 2019a) is a multimodal emotion/sentiment classification dataset which has been created by the extending the EmotionLines dataset (Chen et al., 2018). Contrary to IEMOCAP and AVEC, MELD is a multiparty dialog dataset. MELD contains textual, acoustic and visual information for more than 1400 dialogues and 13000 utterances from the Friends TV series. Each utterance in every dialog is annotated as one of the seven emotion classes: anger, disgust, sadness, joy, surprise, fear or neutral.

| Dataset | # dialogues | | | # utterances | | |
|---|---|---|---|---|---|---|
| | train | val | test | train | val | test |
| IEMOCAP | 120 | | 31 | 5810 | | 1623 |
| AVEC | 63 | | 32 | 4368 | | 1430 |
| MELD | 1039 | 114 | 280 | 9989 | 1109 | 2610 |

Table 2: Training, validation and test data distribution in the datasets. No predefined train/val split is provided in IEMOCAP and AVEC, hence we use 10% of the training dialogues as validation split.

### 4.2 Baselines and State of the Art

For a comprehensive evaluation of DialogueGCN, we compare our model with the following baseline methods:

**CNN (Kim, 2014)** This is the baseline convolutional neural network based model which is identical to our utterance level feature extractor network (Section 3.2). This model is context independent as it doesn't use information from contextual utterances.

**Memnet (Sukhbaatar et al., 2015)** This is an end-to-end memory network baseline (Hazarika et al., 2018b). Every utterance is fed to the network and the memories, which correspond to the previous utterances, is continuously updated in a multi-hop fashion. Finally the output from the memory network is used for emotion classification.

**c-LSTM (Poria et al., 2017)** Context-aware utterance representations are generated by capturing the contextual content from the surrounding utterances using a Bi-directional LSTM (Hochreiter and Schmidhuber, 1997) network. The context-aware utterance representations are then used for emotion classification. The contextual-LSTM

model is speaker independent as it doesn't model any speaker level dependency.

**c-LSTM+Att (Poria et al., 2017)** In this variant of c-LSTM, an attention module is applied to the output of c-LSTM at each timestamp by following Eqs. (5) and (6). Generally this provides better context to create a more informative final utterance representation.

**CMN (Hazarika et al., 2018b)** CMN models utterance context from dialogue history using two distinct GRUs for two speakers. Finally, utterance representation is obtained by feeding the current utterance as query to two distinct memory networks for both speakers. However, this model can only model conversations with two speakers.

**ICON (Hazarika et al., 2018b)** ICON which is an extension of CMN, connects outputs of individual speaker GRUs in CMN using another GRU for explicit inter-speaker modeling. This GRU is considered as a memory to track the overall conversational flow. Similar to CMN, ICON can not be extended to apply on multiparty datasets e.g., MELD.

**DialogueRNN (Majumder et al., 2019)** This is the state-of-the-art method for ERC. It is a recurrent network that uses two GRUs to track individual speaker states and global context during the conversation. Further, another GRU is employed to track emotional state through the conversation. DialogueRNN claims to model inter-speaker relation and it can be applied on multiparty datasets.

## 5 Results and Discussions

### 5.1 Comparison with State of the Art and Baseline

We compare the performance of our proposed DialogueGCN framework with the state-of-the-art DialogueRNN and baseline methods in Tables 3 and 4. We report all results with average of 5 runs. Our DialogueGCN model outperforms the SOTA and all the baseline models, on all the datasets, while also being statistically significant under the paired t-test ($p < 0.05$).

**IEMOCAP and AVEC:** On the IEMOCAP dataset, DialogueGCN achieves new state-of-the-art average F1-score of 64.18% and accuracy of 65.25%, which is around 2% better than DialogueRNN, and at least 5% better than all the other baseline models. Similarly, on AVEC dataset, DialogueGCN outperforms the state-of-the-art on all the four emotion dimensions: valence, arousal, expectancy, and power.

To explain this gap in performance, it is important to understand the nature of these models. DialogueGCN and DialogueRNN both try to model speaker-level context (albeit differently), whereas, none of the other models encode speaker-level context (they only encode sequential context). This is a key limitation in the baseline models, as speaker-level context is indeed very important in conversational emotion recognition.

As for the difference of performance between DialogueRNN and DialogueGCN, we believe that this is due to the different nature of speaker-level context encoding. DialogueRNN employs a gated recurrent unit (GRU) network to model individual speaker states. Both IEMOCAP and AVEC dataset has many conversations with over 70 utterances (the average conversation length is 50 utterances in IEMOCAP and 72 in AVEC). As recurrent encoders have long-term information propagation issues, speaker-level encoding can be problematic for long sequences like those found in these two datasets. In contrast, DialogueGCN tries to overcome this issue by using neighbourhood based convolution to model speaker-level context.

**MELD:** The MELD dataset consists of multiparty conversations and we found that emotion recognition in MELD is considerably harder to model than IEMOCAP and AVEC - which only consists of dyadic conversations. Utterances in MELD are much shorter and rarely contain emotion specific expressions, which means emotion modelling is highly context dependent. Moreover, the average conversation length is 10 utterances, with many conversations having more than 5 participants, which means majority of the participants only utter a small number of utterances per conversation. This makes inter-dependency and self-dependency modeling difficult. Because of these reasons, we found that the difference in results between the baseline models and DialogueGCN is not as contrasting as it is in the case of IEMOCAP and AVEC. Memnet, CMN, and ICON are not suitable for this dataset as they exclusively work in dyadic conversations. Our DialogueGCN model achieves new state-of-the-art F1 score of 58.10% outperforming DialogueRNN by more than 1%.

| Methods | IEMOCAP | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Happy | | Sad | | Neutral | | Angry | | Excited | | Frustrated | | **Average(w)** | |
| | Acc. | F1 | Acc. | F1 | Acc. | F1 | Acc. | F1 | Acc. | F1 | Acc. | F1 | Acc. | F1 |
| CNN | 27.77 | 29.86 | 57.14 | 53.83 | 34.33 | 40.14 | 61.17 | 52.44 | 46.15 | 50.09 | 62.99 | 55.75 | 48.92 | 48.18 |
| Memnet | 25.72 | 33.53 | 55.53 | 61.77 | 58.12 | 52.84 | 59.32 | 55.39 | 51.50 | 58.30 | 67.20 | 59.00 | 55.72 | 55.10 |
| bc-LSTM | 29.17 | 34.43 | 57.14 | 60.87 | 54.17 | 51.81 | 57.06 | 56.73 | 51.17 | 57.95 | 67.19 | 58.92 | 55.21 | 54.95 |
| bc-LSTM+Att | 30.56 | 35.63 | 56.73 | 62.90 | 57.55 | 53.00 | 59.41 | 59.24 | 52.84 | 58.85 | 65.88 | 59.41 | 56.32 | 56.19 |
| CMN | 25.00 | 30.38 | 55.92 | 62.41 | 52.86 | 52.39 | 61.76 | 59.83 | 55.52 | 60.25 | 71.13 | 60.69 | 56.56 | 56.13 |
| ICON | 22.22 | 29.91 | 58.78 | 64.57 | 62.76 | 57.38 | 64.71 | 63.04 | 58.86 | 63.42 | 67.19 | 60.81 | 59.09 | 58.54 |
| DialogueRNN | 25.69 | 33.18 | 75.10 | 78.80 | 58.59 | 59.21 | 64.71 | **65.28** | 80.27 | **71.86** | 61.15 | 58.91 | 63.40 | 62.75 |
| **DialogueGCN** | 40.62 | **42.75** | 89.14 | **84.54** | 61.92 | **63.54** | 67.53 | 64.19 | 65.46 | 63.08 | 64.18 | **66.99** | 65.25 | **64.18** |

Table 3: Comparison with the baseline methods on IEMOCAP dataset; Acc. = Accuracy; bold font denotes the best performances. Average(w) = Weighted average.

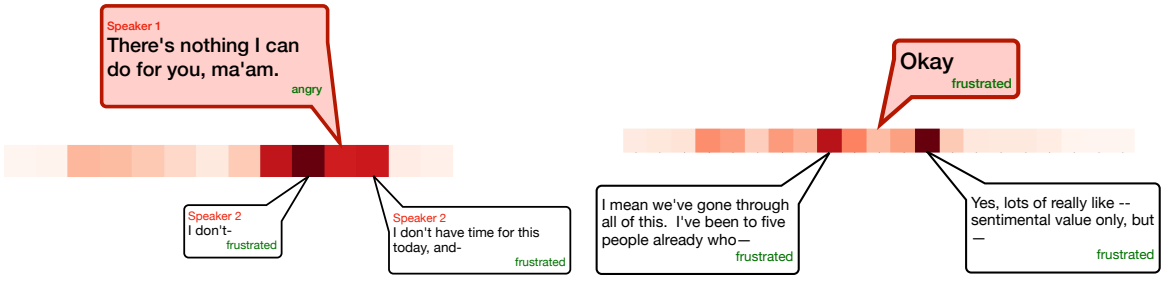


(a) (b)

Figure 4: Visualization of edge-weights in Eq. (1) — (a) Target utterance attends to other speaker's utterance for correct context; (b) Short utterance attends to appropriate contextual utterances to be classified correctly.

We surmise that this improvement is a result of the speaker dependent relation modelling of the edges in our graph network which inherently improves the context understanding over DialogueRNN.

### 5.2 Effect of Context Window

We report results for DialogueGCN model in Tables 3 and 4 with a past and future context window size of (10, 10) to construct the edges. We also carried out experiments with decreasing context window sizes of (8, 8), (4, 4), (0, 0) and found that performance steadily decreased with F1 scores of 62.48%, 59.41% and 55.80% on IEMOCAP. DialogueGCN with context window size of (0, 0) is equivalent to a model with only sequential encoder (as it only has self edges), and performance is expectedly much worse. We couldn't perform extensive experiments with larger windows because of computational constraints, but we expect performance to improve with larger context sizes.

### 5.3 Ablation Study

We perform ablation study for different level of context encoders, namely sequential encoder and speaker-level encoder, in Table 5. We remove them one at a time and found that the speaker-level encoder is slightly more important in overall performance. This is due to speaker-level encoder mitigating long distance dependency issue of sequential encoder and DialogueRNN. Removing both of them results in a very poor F1 score of 36.7 %, which demonstrates the importance of contextual modelling in conversational emotion recognition.

Further, we study the effect of edge relation modelling. As mentioned in Section 3.3.2, there are total $2M^2$ distinct edge relations for a conversation with $M$ distinct speakers. First we removed only the temporal dependency (resulting in $M^2$ distinct edge relations), and then only the speaker dependency (resulting in 2 distinct edge relations) and then both (resulting in a single edge relation all throughout the graph). The results of these tests in Table 6 show that having these different relational edges is indeed very important for modelling emotional dynamics. These results support our hypothesis that each speaker in a conversation is uniquely affected by the others, and hence, modelling interlocutors-dependency is rudimentary. Fig. 4a illustrates one such instance where target utterance attends to other speaker's utterance for context. This phenomenon is com-

| Methods | AVEC | | | | MELD |
|---|---|---|---|---|---|
| | Valence | Arousal | Expectancy | Power | |
| CNN | 0.545 | 0.542 | 0.605 | 8.71 | 55.02 |
| Memnet | 0.202 | 0.211 | 0.216 | 8.97 | - |
| bc-LSTM | 0.194 | 0.212 | 0.201 | 8.90 | 56.44 |
| bc-LSTM+Att | 0.189 | 0.213 | 0.190 | 8.67 | 56.70 |
| CMN | 0.192 | 0.213 | 0.195 | 8.74 | - |
| ICON | 0.180 | 0.190 | 0.180 | 8.45 | - |
| DialogueRNN | 0.168 | 0.165 | 0.175 | 7.90 | 57.03 |
| **DialogueGCN** | **0.157** | **0.161** | **0.168** | **7.68** | **58.10** |

Table 4: Comparison with the baseline methods on AVEC and MELD dataset; MAE and F1 metrics are user for AVEC and MELD, respectively.

| Sequential Encoder | Speaker-Level Encoder | F1 |
|---|---|---|
| ✓ | ✓ | 64.18 |
| ✓ | ✗ | 55.30 |
| ✗ | ✓ | 56.71 |
| ✗ | ✗ | 36.75 |

Table 5: Ablation results w.r.t the contextual encoder modules on IEMOCAP dataset.

| Speaker Dependency Edges | Temporal Dependency Edges | F1 |
|---|---|---|
| ✓ | ✓ | 64.18 |
| ✓ | ✗ | 62.52 |
| ✗ | ✓ | 61.03 |
| ✗ | ✗ | 60.11 |

Table 6: Ablation results w.r.t the edge relations in speaker-level encoder module on IEMOCAP dataset.

monly observable for DialogueGCN, as compared to DialogueRNN.

### 5.4 Performance on Short Utterances

Emotion of short utterances, like "okay", "yeah", depends on the context it appears in. For example, without context "okay" is assumed 'neutral'. However, in Fig. 4b, DialogueGCN correctly classifies "okay" as 'frustration', which is apparent from the context. We observed that, overall, DialogueGCN correctly classifies short utterances, where DialogueRNN fails.

### 5.5 Error Analysis

We analyzed our predicted emotion labels and found that misclassifications are often among similar emotion classes. In the confusion matrix, we observed that our model misclassifies several samples of 'frustrated' as 'angry' and 'neutral'. This is due to subtle difference between frustration and anger. Further, we also observed similar misclassification of 'excited' samples as 'happy' and 'neutral'. All the datasets that we use in our experiment are multimodal. A few utterances e.g., 'ok. yes' carrying *non-neutral* emotions were misclassified as we do not utilize audio and visual modality in our experiments. In such utterances, we found audio and visual (in this particular example, high pitched audio and frowning expression) modality providing key information to detect underlying emotions (*frustrated* in the above utterance) which DialogueGCN failed to understand by just looking at the textual context.

## 6 Conclusion

In this work, we present Dialogue Graph Convolutional Network (DialogueGCN), that models inter and self-party dependency to improve context understanding for utterance-level emotion detection in conversations. On three benchmark ERC datasets, DialogueGCN outperforms the strong baselines and existing state of the art, by a significant margin. Future works will focus on incorporating multimodal information into DialogueGCN, speaker-level emotion shift detection, and conceptual grounding of conversational emotion reasoning. We also plan to use DialogueGCN in dialogue systems to generate affective responses.

## References


James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. 2017. Quasi-Recurrent Neural Networks. In *International Conference on Learning Representations (ICLR 2017)*.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2013. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*.

Carlos Busso, Murtaza Bulut, Chi-Chun Lee, Abe Kazemzadeh, Emily Mower, Samuel Kim, Jeannette N Chang, Sungbok Lee, and Shrikanth S Narayanan. 2008. IEMOCAP: Interactive emotional dyadic motion capture database. *Language resources and evaluation*, 42(4):335–359.

Minghai Chen, Sen Wang, Paul Pu Liang, Tadas Baltrušaitis, Amir Zadeh, and Louis-Philippe Morency. 2017. Multimodal sentiment analysis with word-level fusion and reinforcement learning. In *Proceedings of the 19th ACM International Conference on Multimodal Interaction*, pages 163–171. ACM.

Sheng-Yeh Chen, Chao-Chun Hsu, Chuan-Chun Kuo, Lun-Wei Ku, et al. 2018. Emotionlines: An emotion corpus of multi-party conversations. *arXiv preprint arXiv:1802.08379*.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *CoRR*, abs/1412.3555.

Niko Colneriĉ and Janez Demsar. 2018. Emotion recognition on twitter: comparative study and training a unison model. *IEEE Transactions on Affective Computing*.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3844–3852. Curran Associates, Inc.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1263–1272. JMLR. org.

Devamanyu Hazarika, Soujanya Poria, Rada Mihalcea, Erik Cambria, and Roger Zimmermann. 2018a. Icon: Interactive conversational memory network for multimodal emotion detection. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2594–2604.

Devamanyu Hazarika, Soujanya Poria, Amir Zadeh, Erik Cambria, Louis-Philippe Morency, and Roger Zimmermann. 2018b. Conversational Memory Network for Emotion Recognition in Dyadic Dialogue Videos. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2122–2132, New Orleans, Louisiana. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Chenyang Huang, Amine Trabelsi, and Osmar R Zaïane. 2019. Ana at semeval-2019 task 3: Contextual emotion detection in conversations through hierarchical lstms and bert. *arXiv preprint arXiv:1904.00132*.

Sidney K. D'Mello, Scotty Craig, Jeremiah Sullins, and Arthur Graesser. 2006. Predicting affective states expressed through an emote-aloud procedure from autotutor's mixed-initiative dialogue. *I. J. Artificial Intelligence in Education*, 16:3–28.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP 2014*, pages 1746–1751.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Bernhard Kratzwald, Suzana Ilic, Mathias Kraus, Stefan Feuerriegel, and Helmut Prendinger. 2018. Decision support with text-based emotion recognition: Deep learning for affective computing. *arXiv preprint arXiv:1803.06397*.

Navonil Majumder, Soujanya Poria, Devamanyu Hazarika, Rada Mihalcea, Alexander Gelbukh, and Erik Cambria. 2019. DialogueRNN: An Attentive RNN for Emotion Detection in Conversations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6818–6825.

G. McKeown, M. Valstar, R. Cowie, M. Pantic, and M. Schroder. 2012. The SEMAINE Database: Annotated Multimodal Records of Emotionally Colored Conversations between a Person and a Limited Agent. *IEEE Transactions on Affective Computing*, 3(1):5–17.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

Costanza Navarretta, K Choukri, T Declerck, S Goggi, M Grobelnik, and B Maegaard. 2016. Mirroring facial expressions and emotions in dyadic conversations. In *LREC*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Soujanya Poria, Erik Cambria, Devamanyu Hazarika, Navonil Majumder, Amir Zadeh, and Louis-Philippe Morency. 2017. Context-Dependent Sentiment Analysis in User-Generated Videos. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 873–883, Vancouver, Canada. Association for Computational Linguistics.

Soujanya Poria, Devamanyu Hazarika, Navonil Majumder, Gautam Naik, Erik Cambria, and Rada Mihalcea. 2019a. MELD: A multimodal multi-party dataset for emotion recognition in conversations. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 527–536, Florence, Italy. Association for Computational Linguistics.

Soujanya Poria, Navonil Majumder, Rada Mihalcea, and Eduard Hovy. 2019b. Emotion recognition in conversation: Research challenges, datasets, and recent advances. *IEEE Access*, 7:100943–100953.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2008. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer.

Björn Schuller, Michel Valster, Florian Eyben, Roddy Cowie, and Maja Pantic. 2012. AVEC 2012: The Continuous Audio/Visual Emotion Challenge. In *Proceedings of the 14th ACM International Conference on Multimodal Interaction*, ICMI '12, pages 449–456, New York, NY, USA. ACM.

Carlo Strapparava and Rada Mihalcea. 2010. Annotating and identifying emotions in text. In *Intelligent Information Access*, pages 21–38. Springer.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end Memory Networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, pages 2440–2448, Cambridge, MA, USA. MIT Press.

Amir Zadeh, Paul Pu Liang, Navonil Mazumder, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. 2018a. Memory Fusion Network for Multi-view Sequential Learning. In *AAAI Conference on Artificial Intelligence*, pages 5634–5641.

Amir Zadeh, Paul Pu Liang, Soujanya Poria, Prateek Vij, Erik Cambria, and Louis-Philippe Morency. 2018b. Multi-attention recurrent network for human communication comprehension. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 5642–5649.

Hao Zhou, Minlie Huang, Tianyang Zhang, Xiaoyan Zhu, and Bing Liu. 2018. Emotional chatting machine: Emotional conversation generation with internal and external memory. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

# Knowledge-Enriched Transformer for Emotion Detection in Textual Conversations

**Peixiang Zhong**[1,2], **Di Wang**[1], **Chunyan Miao**[1,2,3]
[1]Joint NTU-UBC Research Centre of Excellence in Active Living for the Elderly
[2]Alibaba-NTU Singapore Joint Research Institute
[3]School of Computer Science and Engineering
Nanyang Technological University, Singapore
peixiang001@e.ntu.edu.sg, {wangdi, ascymiao}@ntu.edu.sg

## Abstract

Messages in human conversations inherently convey emotions. The task of detecting emotions in textual conversations leads to a wide range of applications such as opinion mining in social networks. However, enabling machines to analyze emotions in conversations is challenging, partly because humans often rely on the context and commonsense knowledge to express emotions. In this paper, we address these challenges by proposing a Knowledge-Enriched Transformer (KET), where contextual utterances are interpreted using hierarchical self-attention and external commonsense knowledge is dynamically leveraged using a context-aware affective graph attention mechanism. Experiments on multiple textual conversation datasets demonstrate that both context and commonsense knowledge are consistently beneficial to the emotion detection performance. In addition, the experimental results show that our KET model outperforms the state-of-the-art models on most of the tested datasets in F1 score.

## 1 Introduction

Emotions are "generated states in humans that reflect evaluative judgments of the environment, the self and other social agents" (Hudlicka, 2011). Messages in human communications inherently convey emotions. With the prevalence of social media platforms such as Facebook Messenger, as well as conversational agents such as Amazon Alexa, there is an emerging need for machines to understand human emotions in natural conversations. This work addresses the task of detecting emotions (e.g., happy, sad, angry, etc.) in textual conversations, where the emotion of an utterance is detected in the conversational context. Being able to effectively detect emotions in conversations leads to a wide range of applications ranging from opinion mining in social media platforms (Chatterjee et al., 2019) to building emotion-aware conversational agents (Zhou et al., 2018a).

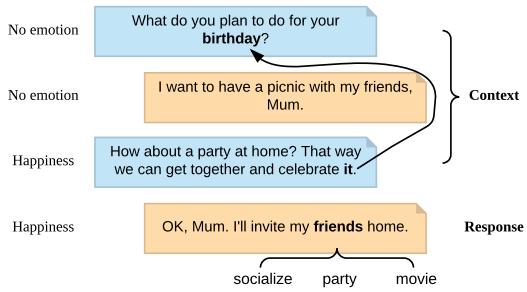


Figure 1: An example conversation with annotated labels from the DailyDialog dataset (Li et al., 2017). By referring to the context, "it" in the third utterance is linked to "birthday" in the first utterance. By leveraging an external knowledge base, the meaning of "friends" in the forth utterance is enriched by associated knowledge entities, namely "socialize", "party", and "movie". Thus, the implicit "happiness" emotion in the fourth utterance can be inferred more easily via its enriched meaning.

However, enabling machines to analyze emotions in human conversations is challenging, partly because humans often rely on the context and commonsense knowledge to express emotions, which is difficult to be captured by machines. Figure 1 shows an example conversation demonstrating the importance of context and commonsense knowledge in understanding conversations and detecting implicit emotions.

There are several recent studies that model contextual information to detect emotions in conversations. Poria et al. (2017) and Majumder et al. (2019) leveraged recurrent neural networks (RNN) to model the contextual utterances in sequence, where each utterance is represented by a feature vector extracted by convolutional neural networks (CNN) at an earlier stage. Similarly, Hazarika et al. (2018a,b) proposed to use extracted CNN



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 165–176, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

features in memory networks to model contextual utterances. However, these methods require separate feature extraction and tuning, which may not be ideal for real-time applications. In addition, to the best of our knowledge, no attempts have been made in the literature to incorporate commonsense knowledge from external knowledge bases to detect emotions in textual conversations. Commonsense knowledge is fundamental to understanding conversations and generating appropriate responses (Zhou et al., 2018b).

To this end, we propose a Knowledge-Enriched Transformer (KET) to effectively incorporate contextual information and external knowledge bases to address the aforementioned challenges. The Transformer (Vaswani et al., 2017) has been shown to be a powerful representation learning model in many NLP tasks such as machine translation (Vaswani et al., 2017) and language understanding (Devlin et al., 2018). The self-attention (Cheng et al., 2016) and cross-attention (Bahdanau et al., 2014) modules in the Transformer capture the intra-sentence and inter-sentence correlations, respectively. The shorter path of information flow in these two modules compared to gated RNNs and CNNs allows KET to model contextual information more efficiently. In addition, we propose a hierarchical self-attention mechanism allowing KET to model the hierarchical structure of conversations. Our model separates context and response into the encoder and decoder, respectively, which is different from other Transformer-based models, e.g., BERT (Devlin et al., 2018), which directly concatenate context and response, and then train language models using only the encoder part.

Moreover, to exploit commonsense knowledge, we leverage external knowledge bases to facilitate the understanding of each word in the utterances by referring to related knowledge entities. The referring process is dynamic and balances between relatedness and affectiveness of the retrieved knowledge entities using a context-aware affective graph attention mechanism.

In summary, our contributions are as follows:

- For the first time, we apply the Transformer to analyze conversations and detect emotions. Our hierarchical self-attention and cross-attention modules allow our model to exploit contextual information more efficiently than existing gated RNNs and CNNs.
- We derive dynamic, context-aware, and emotion-related commonsense knowledge from external knowledge bases and emotion lexicons to facilitate the emotion detection in conversations.
- We conduct extensive experiments demonstrating that both contextual information and commonsense knowledge are beneficial to the emotion detection performance. In addition, our proposed KET model outperforms the state-of-the-art models on most of the tested datasets across different domains.

## 2 Related Work

**Emotion Detection in Conversations:** Early studies on emotion detection in conversations focus on call center dialogs using lexicon-based methods and audio features (Lee and Narayanan, 2005; Devillers and Vidrascu, 2006). Devillers et al. (2002) annotated and detected emotions in call center dialogs using unigram topic modelling. In recent years, there is an emerging research trend on emotion detection in conversational videos and multi-turn Tweets using deep learning methods (Hazarika et al., 2018b,a; Zahiri and Choi, 2018; Chatterjee et al., 2019; Zhong and Miao, 2019; Poria et al., 2019). Poria et al. (2017) proposed a long short-term memory network (LSTM) (Hochreiter and Schmidhuber, 1997) based model to capture contextual information for sentiment analysis in user-generated videos. Majumder et al. (2019) proposed the DialogueRNN model that uses three gated recurrent units (GRU) (Cho et al., 2014) to model the speaker, the context from the preceding utterances, and the emotions of the preceding utterances, respectively. They achieved the state-of-the-art performance on several conversational video datasets.

**Knowledge Base in Conversations:** Recently there is a growing number of studies on incorporating knowledge base in generative conversation systems, such as open-domain dialogue systems (Han et al., 2015; Asghar et al., 2018; Ghazvininejad et al., 2018; Young et al., 2018; Parthasarathi and Pineau, 2018; Liu et al., 2018; Moghe et al., 2018; Dinan et al., 2019; Zhong et al., 2019), task-oriented dialogue systems (Madotto et al., 2018; Wu et al., 2019; He et al., 2019) and question answering systems (Kiddon et al., 2016; Hao et al., 2017; Sun et al., 2018; Mihaylov and Frank, 2018). Zhou et al. (2018b) adopted structured

knowledge graphs to enrich the interpretation of input sentences and help generate knowledge-aware responses using graph attentions. The graph attention in the knowledge interpreter (Zhou et al., 2018b) is static and only related to the recognized entity of interest. By contrast, our graph attention mechanism is dynamic and selects context-aware knowledge entities that balances between relatedness and affectiveness.

**Emotion Detection in Text:** There is a trend moving from traditional machine learning methods (Pang et al., 2002; Wang and Manning, 2012; Seyeditabari et al., 2018) to deep learning methods (Abdul-Mageed and Ungar, 2017; Zhang et al., 2018b) for emotion detection in text. Khanpour and Caragea (2018) investigated the emotion detection from health-related posts in online health communities using both deep learning features and lexicon-based features.

**Incorporating Knowledge in Sentiment Analysis:** Traditional lexicon-based methods detect emotions or sentiments from a piece of text based on the emotions or sentiments of words or phrases that compose it (Hu et al., 2009; Taboada et al., 2011; Bandhakavi et al., 2017). Few studies investigated the usage of knowledge bases in deep learning methods. Kumar et al. (2018) proposed to use knowledge from WordNet (Fellbaum, 2012) to enrich the text representations produced by LSTM and obtained improved performance.

**Transformer:** The Transformer has been applied to many NLP tasks due to its rich representation and fast computation, e.g., document machine translation (Zhang et al., 2018a), response matching in dialogue system (Zhou et al., 2018c), language modelling (Dai et al., 2019) and understanding (Radford et al., 2018). A very recent work (Rik Koncel-Kedziorski and Hajishirzi, 2019) extends the Transformer to graph inputs and propose a model for graph-to-text generation.

## 3 Our Proposed KET Model

In this section we present the task definition and our proposed KET model.

### 3.1 Task Definition

Let $\{X_j^i, Y_j^i\}, i = 1, ...N, j = 1, ...N_i$ be a collection of $\{$*utterance*, *label*$\}$ pairs in a given dialogue dataset, where $N$ denotes the number of conversations and $N_i$ denotes the number of utterances in the $i$th conversation. The objective of the task is to maximize the following function:

$$\Phi = \prod_{i=1}^{N} \prod_{j=1}^{N_i} p(Y_j^i | X_j^i, X_{j-1}^i, ..., X_1^i; \theta), \quad (1)$$

where $X_{j-1}^i, ..., X_1^i$ denote contextual utterances and $\theta$ denotes the model parameters we want to optimize.

We limit the number of contextual utterances to $M$. Discarding early contextual utterances may cause information loss, but this loss is negligible because they only contribute the least amount of information (Su et al., 2018). This phenomenon can be further observed in our model analysis regarding context length (see Section 5.2). Similar to (Poria et al., 2017), we clip and pad each utterance $X_j^i$ to a fixed $m$ number of tokens. The overall architecture of our KET model is illustrated in Figure 2.

### 3.2 Knowledge Retrieval

We use a commonsense knowledge base ConceptNet (Speer et al., 2017) and an emotion lexicon NRC_VAD (Mohammad, 2018a) as knowledge sources in our model.

ConceptNet is a large-scale multilingual semantic graph that describes general human knowledge in natural language. The nodes in ConceptNet are concepts and the edges are relations. Each $\langle$*concept1*, *relation*, *concept2*$\rangle$ triplet is an assertion. Each assertion is associated with a confidence score. An example assertion is $\langle$*friends*, *CausesDesire*, *socialize*$\rangle$ with confidence score of 3.46. Usually assertion confidence scores are in the $[1, 10]$ interval. Currently, for English, ConceptNet comprises 5.9M assertions, 3.1M concepts and 38 relations.

NRC_VAD is a list of English words and their VAD scores, i.e., valence (negative-positive), arousal (calm-excited), and dominance (submissive-dominant) scores in the $[0, 1]$ interval. The VAD measure of emotion is culture-independent and widely adopted in Psychology (Mehrabian, 1996). Currently NRC_VAD comprises around 20K words.

In general, for each non-stopword token $t$ in $X_j^i$, we retrieve a connected knowledge graph $g(t)$ comprising its immediate neighbors from ConceptNet. For each $g(t)$, we remove concepts that are stopwords or not in our vocabulary. We further remove concepts with confidence scores less

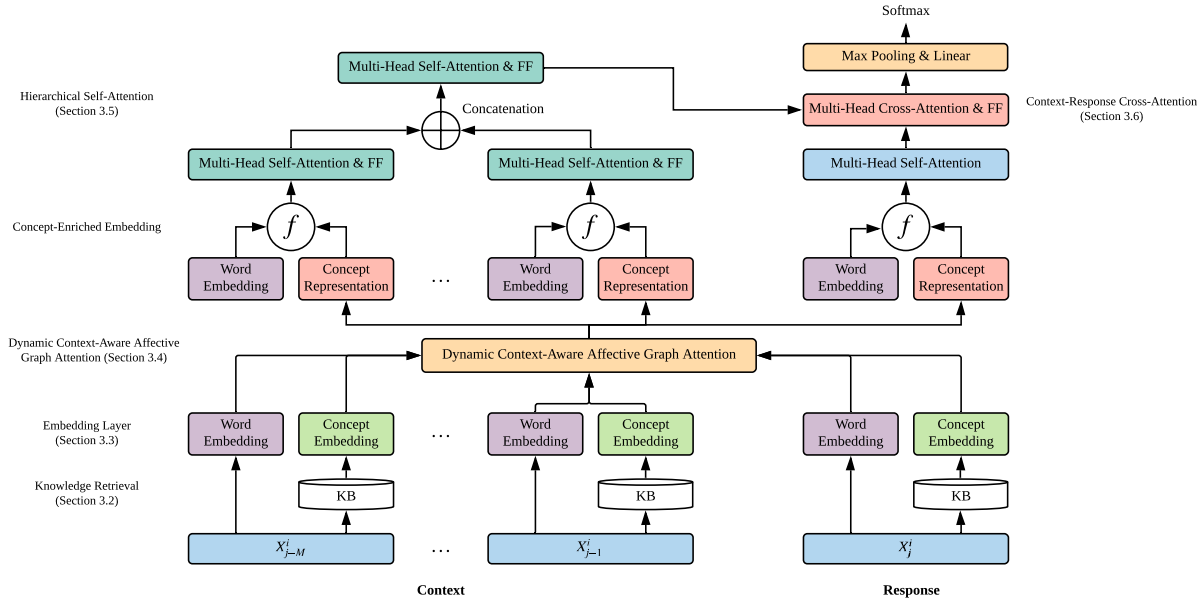


Figure 2: Overall architecture of our proposed KET model. The positional encoding, residual connection, and layer normalization are omitted in the illustration for brevity.

than 1 to reduce annotation noises. For each concept, we retrieve its VAD values from NRC_VAD. The final knowledge representation for each token $t$ is a list of tuples: $(c_1, s_1, \mathit{VAD}(c_1))$, $(c_2, s_2, \mathit{VAD}(c_2))$, ..., $(c_{|g(t)|}, s_{|g(t)|}, \mathit{VAD}(c_{|g(t)|}))$, where $c_k \in g(t)$ denotes the $k$th connected concept, $s_k$ denotes the associated confidence score, and $\mathit{VAD}(c_k)$ denotes the VAD values of $c_k$. The treatment for tokens that are not associated with any concept and concepts that are not included in NRC_VAD are discussed in Section 3.4. We leave the treatment on relations as future work.

### 3.3 Embedding Layer

We use a word embedding layer to convert each token $t$ in $X^i$ into a vector representation $\mathbf{t} \in \mathbb{R}^d$, where $d$ denotes the size of word embedding. To encode positional information, the position encoding (Vaswani et al., 2017) is added as follows:

$$\mathbf{t} = \mathit{Embed}(t) + \mathit{Pos}(t). \tag{2}$$

Similarly, we use a concept embedding layer to convert each concept $c$ into a vector representation $\mathbf{c} \in \mathbb{R}^d$ but without position encoding.

### 3.4 Dynamic Context-Aware Affective Graph Attention

To enrich word embedding with concept representations, we propose a dynamic context-aware affective graph attention mechanism to compute the concept representation for each token. Specifically, the concept representation $\mathbf{c}(t) \in \mathbb{R}^d$ for token $t$ is computed as

$$\mathbf{c}(t) = \sum_{k=1}^{|g(t)|} \alpha_k * \mathbf{c_k}, \tag{3}$$

where $\mathbf{c_k} \in \mathbb{R}^d$ denotes the concept embedding of $c_k$ and $\alpha_k$ denotes its attention weight. If $|g(t)| = 0$, we set $\mathbf{c}$(t) to the average of all concept embeddings. The attention $\alpha_k$ in Equation 3 is computed as

$$\alpha_k = \mathit{softmax}(w_k), \tag{4}$$

where $w_k$ denotes the weight of $c_k$.

The derivation of $w_k$ is crucial because it regulates the contribution of $\mathbf{c}_k$ towards enriching $\mathbf{t}$. A standard graph attention mechanism (Velikovi et al., 2018) computes $w_k$ by feeding $\mathbf{t}$ and $\mathbf{c}_k$ into a single-layer feedforward neural network. However, not all related concepts are equal in detecting emotions given the conversational context. In our model, we make the assumption that important concepts are those that relate to the conversational context and have strong emotion intensity. To this end, we propose a context-aware affective graph attention mechanism by incorporating two factors when computing $w_k$, namely relatedness and affectiveness.

**Relatedness:** Relatedness measures the strength of the relation between $c_k$ and the conversational context. The relatedness factor in $w_k$ is computed as

$$rel_k = \textit{min-max}(s_k) * \textit{abs}(\cos(\mathbf{CR}(X^i), \mathbf{c_k})), \quad (5)$$

where $s_k$ is the confidence score introduced in Section 3.2, *min-max* denotes min-max scaling for each token $t$, *abs* denotes the absolute function, cos denotes the cosine similarity function, and $\mathbf{CR}(X^i) \in \mathbb{R}^d$ denotes the context representation of the $i$th conversation $X^i$. Here we compute $\mathbf{CR}(X^i)$ as the average of all sentence representations in $X^i$ as follows:

$$\mathbf{CR}(X^i) = \textit{avg}(\mathbf{SR}(X^i_{j-M}), ..., \mathbf{SR}(X^i_j)), \quad (6)$$

where $\mathbf{SR}(X^i_j) \in \mathbb{R}^d$ denotes the sentence representation of $X^i_j$. We compute $\mathbf{SR}(X^i_j)$ via hierarchical pooling (Shen et al., 2018) where $n$-gram ($n \leq 3$) representations in $X^i_j$ are first computed by max-pooling and then all $n$-gram representations are averaged. The hierarchical pooling mechanism preserves word order information to certain degree and has demonstrated superior performance than average pooling or max-pooling on sentiment analysis tasks (Shen et al., 2018).

**Affectiveness:** Affectiveness measures the emotion intensity of $c_k$. The affectiveness factor in $w_k$ is computed as

$$\textit{aff}_k = \textit{min-max}(||[V(c_k)-1/2, A(c_k)/2]||_2), \quad (7)$$

where $||.||_k$ denotes $l_k$ norm, $V(c_k) \in [0, 1]$ and $A(c_k) \in [0, 1]$ denote the valence and arousal values of $VAD(c_k)$, respectively. Intuitively, $\textit{aff}_k$ considers the deviations of valence from neutral and the level of arousal from calm. There is no established method in the literature to compute the emotion intensity based on VAD values, but empirically we found that our method correlates better with an emotion intensity lexicon comprising 6K English words (Mohammad, 2018b) than other methods such as taking dominance into consideration or taking $l_1$ norm. For concept $c_k$ not in NRC_VAD, we set $\textit{aff}_k$ to the mid value of 0.5.

Combining both $rel_k$ and $\textit{aff}_k$, we define the weight $w_k$ as follows:

$$w_k = \lambda_k * rel_k + (1 - \lambda_k) * \textit{aff}_k, \quad (8)$$

where $\lambda_k$ is a model parameter balancing the impacts of relatedness and affectiveness on computing concept representations. Parameter $\lambda_k$ can be fixed or learned during training. The analysis of $\lambda_k$ is discussed in Section 5.2.

Finally, the concept-enriched word representation $\hat{\mathbf{t}}$ can be obtained via a linear transformation:

$$\hat{\mathbf{t}} = \mathbf{W}[\mathbf{t}; \mathbf{c}(t)], \quad (9)$$

where $[;]$ denotes concatenation and $\mathbf{W} \in \mathbb{R}^{d \times 2d}$ denotes a model parameter. All $m$ tokens in each $X^i_j$ then form a concept-enriched utterance embedding $\hat{\mathbf{X}}^i_j \in \mathbb{R}^{m \times d}$.

### 3.5 Hierarchical Self-Attention

We propose a hierarchical self-attention mechanism to exploit the structural representation of conversations and learn a vector representation for the contextual utterances $X^i_{j-1}, ..., X^i_{j-M}$. Specifically, the hierarchical self-attention follows two steps: 1) each utterance representation is computed using an utterance-level self-attention layer, and 2) a context representation is computed from $M$ learned utterance representations using a context-level self-attention layer.

At step 1, for each utterance $X^i_n$, $n$=$j-1$, ..., $j-M$, its representation $\hat{\mathbf{X}}'^i_n \in \mathbb{R}^{m \times d}$ is learned as follows:

$$\hat{\mathbf{X}}'^i_n = \textit{FF}(L'(\textit{MH}(L(\hat{\mathbf{X}}^i_n), L(\hat{\mathbf{X}}^i_n), L(\hat{\mathbf{X}}^i_n)))), \quad (10)$$

where $L(\hat{\mathbf{X}}^i_n) \in \mathbb{R}^{m \times h \times d_s}$ is linearly transformed from $\hat{\mathbf{X}}^i_n$ to form $h$ heads ($d_s = d/h$), $L'$ linearly transforms from $h$ heads back to 1 head, and

$$\textit{MH}(Q, K, V) = \textit{softmax}(\frac{QK^T}{\sqrt{d_s}})V, \quad (11)$$

$$\textit{FF}(x) = \textit{max}(0, xW_1 + b_1)W_2 + b_2, \quad (12)$$

where $Q$, $K$, and $V$ denote sets of queries, keys and values, respectively, $W_1 \in \mathbb{R}^{d \times p}, b_1 \in \mathbb{R}^p, W_2 \in \mathbb{R}^{p \times d}$ and $b_2 \in \mathbb{R}^d$ denote model parameters, and $p$ denotes the hidden size of the point-wise feedforward layer (FF) (Vaswani et al., 2017). The multi-head self-attention layer (MH) enables our model to jointly attend to information from different representation subspaces (Vaswani et al., 2017). The scaling factor $\frac{1}{\sqrt{d_s}}$ is added to ensure the dot product of two vectors do not get overly large. Similar to (Vaswani et al., 2017), both MH and FF layers are followed by residual connection and layer normalization, which are omitted in Equation 10 for brevity.

| Dataset | Domain | #Conv. (Train/Val/Test) | #Utter. (Train/Val/Test) | #Classes | Evaluation |
|---|---|---|---|---|---|
| EC | Tweet | 30160/2755/5509 | 90480/8265/16527 | 4 | Micro-F1 |
| DailyDialog | Daily Communication | 11118/1000/1000 | 87170/8069/7740 | 7 | Micro-F1 |
| MELD | TV Show Scripts | 1038/114/280 | 9989/1109/2610 | 7 | Weighted-F1 |
| EmoryNLP | TV Show Scripts | 659/89/79 | 7551/954/984 | 7 | Weighted-F1 |
| IEMOCAP | Emotional Dialogues | 100/20/31 | 4810/1000/1523 | 6 | Weighted-F1 |

Table 1: Dataset descriptions.

At step 2, to effectively combine all utterance representations in the context, the context-level self-attention layer is proposed to hierarchically learn the context-level representation $\mathbf{C}^i \in \mathbb{R}^{M\times m\times d}$ as follows:

$$\mathbf{C}^i = FF(L^{'}(MH(L(\hat{\mathbf{X}}^i), L(\hat{\mathbf{X}}^i), L(\hat{\mathbf{X}}^i)))), \quad (13)$$

where $\hat{\mathbf{X}}^i$ denotes $[\hat{\mathbf{X}}^{'i}_{j-M}; ...; \hat{\mathbf{X}}^{'i}_{j-1}]$, which is the concatenation of all learned utterance representations in the context.

### 3.6 Context-Response Cross-Attention

Finally, a context-aware concept-enriched response representation $\mathbf{R}^i \in \mathbb{R}^{m\times d}$ for conversation $X^i$ is learned by cross-attention (Bahdanau et al., 2014), which selectively attends to the concept-enriched context representation as follows:

$$\mathbf{R}^i = FF(L^{'}(MH(L(\hat{\mathbf{X}}^{'i}_j), L(\mathbf{C}^i), L(\mathbf{C}^i)))), \quad (14)$$

where the response utterance representation $\hat{\mathbf{X}}^{'i}_j \in \mathbb{R}^{m\times d}$ is obtained via the MH layer:

$$\hat{\mathbf{X}}^{'i}_j = L^{'}(MH(L(\hat{\mathbf{X}}^i_j), L(\hat{\mathbf{X}}^i_j), L(\hat{\mathbf{X}}^i_j))), \quad (15)$$

The resulted representation $\mathbf{R}^i \in \mathbb{R}^{m\times d}$ is then fed into a max-pooling layer to learn discriminative features among the positions in the response and derive the final representation $\mathbf{O} \in \mathbb{R}^d$:

$$\mathbf{O} = max\_pool(\mathbf{R}^i). \quad (16)$$

The output probability $p$ is then computed as

$$p = softmax(\mathbf{O}W_3 + b_3), \quad (17)$$

where $W_3 \in \mathbb{R}^{d\times q}$ and $b_3 \in \mathbb{R}^q$ denote model parameters, and $q$ denotes the number of classes. The entire KET model is optimized in an end-to-end manner as defined in Equation 1. Our model is available at here[1].

[1]https://github.com/zhongpeixiang/KET

## 4 Experimental Settings

In this section we present the datasets, evaluation metrics, baselines, our model variants, and other experimental settings.

### 4.1 Datasets and Evaluations

We evaluate our model on the following five emotion detection datasets of various sizes and domains. The statistics are reported in Table 1.

**EC** (Chatterjee et al., 2019): Three-turn Tweets. The emotion labels include happiness, sadness, anger and other.

**DailyDialog** (Li et al., 2017): Human written daily communications. The emotion labels include neutral and Ekman's six basic emotions (Ekman, 1992), namely happiness, surprise, sadness, anger, disgust and fear.

**MELD** (Poria et al., 2018): TV show scripts collected from *Friends*. The emotion labels are the same as the ones used in DailyDialog.

**EmoryNLP** (Zahiri and Choi, 2018): TV show scripts collected from *Friends* as well. However, its size and annotations are different from MELD. The emotion labels include neutral, sad, mad, scared, powerful, peaceful, and joyful.

**IEMOCAP** (Busso et al., 2008): Emotional dialogues. The emotion labels include neutral, happiness, sadness, anger, frustrated, and excited.

In terms of the evaluation metric, for EC and DailyDialog, we follow (Chatterjee et al., 2019) to use the micro-averaged F1 excluding the majority class (neutral), due to their extremely unbalanced labels (the percentage of the majority class in the test set is over 80%). For the rest relatively balanced datasets, we follow (Majumder et al., 2019) to use the weighted macro-F1.

### 4.2 Baselines and Model Variants

For a comprehensive performance evaluation, we compare our model with the following baselines:

**cLSTM**: A contextual LSTM model. An utterance-level bidirectional LSTM is used to encode each utterance. A context-level unidirectional LSTM is used to encode the context.

| Model | EC | DailyDialog | MELD | EmoryNLP | IEMOCAP |
|---|---|---|---|---|---|
| cLSTM | 0.6913 | 0.4990 | 0.4972 | 0.2601 | 0.3484 |
| CNN (Kim, 2014) | 0.7056 | 0.4934 | 0.5586 | 0.3259 | 0.5218 |
| CNN+cLSTM (Poria et al., 2017) | 0.7262 | 0.5024 | 0.5687 | 0.3289 | 0.5587 |
| BERT_BASE (Devlin et al., 2018) | 0.6946 | 0.5312 | 0.5621 | 0.3315 | 0.6119 |
| DialogueRNN (Majumder et al., 2019) | 0.7405 | 0.5065 | 0.5627 | 0.3170 | **0.6121** |
| KET_SingleSelfAttn (ours) | 0.7285 | 0.5192 | 0.5624 | 0.3251 | 0.5810 |
| KET_StdAttn (ours) | **0.7413** | 0.5254 | 0.5682 | 0.3353 | 0.5861 |
| KET (ours) | 0.7348 | **0.5337** | **0.5818** | **0.3439** | 0.5956 |

Table 2: Performance comparisons on the five test sets. Best values are highlighted in bold.

| Dataset | M | m | d | p | h |
|---|---|---|---|---|---|
| EC | 2 | 30 | 200 | 100 | 4 |
| DailyDialog | 6 | 30 | 300 | 400 | 4 |
| MELD | 6 | 30 | 200 | 100 | 4 |
| EmoryNLP | 6 | 30 | 100 | 200 | 4 |
| IEMOCAP | 6 | 30 | 300 | 400 | 4 |

Table 3: Hyper-parameter settings for KET. $M$: context length. $m$: number of tokens per utterance. $d$: word embedding size. $p$: hidden size in FF layer. $h$: number of heads.

**CNN** (Kim, 2014): A single-layer CNN with strong empirical performance. This model is trained on the utterance-level without context.

**CNN+cLSTM** (Poria et al., 2017): An CNN is used to extract utterance features. An cLSTM is then applied to learn context representations.

**BERT_BASE** (Devlin et al., 2018): Base version of the state-of-the-art model for sentiment classification. We treat each utterance with its context as a single document. We limit the document length to the last 100 tokens to allow larger batch size. We do not experiment with the large version of BERT due to memory constraint of our GPU.

**DialogueRNN** (Majumder et al., 2019): The state-of-the-art model for emotion detection in textual conversations. It models both context and speakers information. The CNN features used in DialogueRNN are extracted from the carefully tuned CNN model. For datasets without speaker information, i.e., EC and DailyDialog, we use two speakers only. For MELD and EmoryNLP, which have 260 and 255 speakers, respectively, we additionally experimented with clipping the number of speakers to the most frequent ones (6 main speakers + an universal speaker representing all other speakers) and reported the best results.

**KET_SingleSelfAttn**: We replace the hierarchical self-attention by a single self-attention layer to learn context representations. Contextual utterances are concatenated together prior to the single self-attention layer.

**KET_StdAttn**: We replace the dynamic context-aware affective graph attention by the standard graph attention (Velikovi et al., 2018).

### 4.3 Other Experimental Settings

We preprocessed all datasets by lower-casing and tokenization using Spacy[2]. We keep all tokens in the vocabulary[3]. We use the released code for BERT_BASE and DialogueRNN. For each dataset, all models are fine-tuned based on their performance on the validation set.

For our model in all datasets, we use Adam optimization (Kingma and Ba, 2014) with a batch size of 64 and learning rate of 0.0001 throughout the training process. We use GloVe embedding (Pennington et al., 2014) for initialization in the word and concept embedding layers[4]. For the class weights in cross-entropy loss for each dataset, we set them as the ratio of the class distribution in the validation set to the class distribution in the training set. Thus, we can alleviate the problem of unbalanced dataset. The detailed hyper-parameter settings for KET are presented in Table 3.

## 5 Result Analysis

In this section we present model evaluation results, model analysis, and error analysis.

### 5.1 Comparison with Baselines

We compare the performance of KET against that of the baseline models on the five afore-introduced datasets. The results are reported in Table 2. Note that our results for CNN, CNN+cLSTM and DialogueRNN on EC, MELD and IEMOCAP are slightly different from the reported results in (Majumder et al., 2019; Poria et al., 2019).

[2]https://spacy.io/

[3]We keep tokens with minimum frequency of 2 for DailyDialog due to its large vocabulary size

[4]We use GloVe embeddings from Magnitude Medium: https://github.com/plasticityai/magnitude

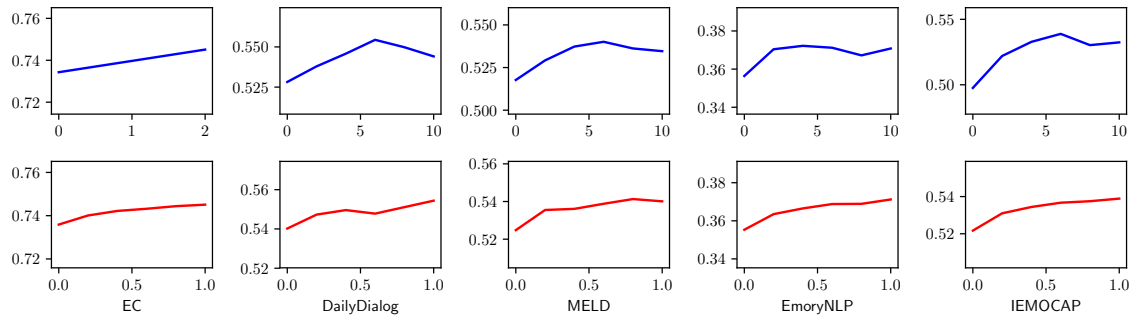


Figure 3: Validation performance by KET. Top: different context length ($M$). Bottom: different sizes of random fractions of ConceptNet.

cLSTM performs reasonably well on short conversations (i.e., EC and DailyDialog), but the worst on long conversations (i.e., MELD, EmoryNLP and IEMOCAP). One major reason is that learning long dependencies using gated RNNs may not be effective enough because the gradients are expected to propagate back through inevitably a huge number of utterances and tokens in sequence, which easily leads to the vanishing gradient problem (Bengio et al., 1994). In contrast, when the utterance-level LSTM in cLSTM is replaced by features extracted by CNN, i.e., the CNN+cLSTM, the model performs significantly better than cLSTM on long conversations, which further validates that modelling long conversations using only RNN models may not be sufficient. BERT_BASE achieves very competitive performance on all datasets except EC due to its strong representational power via bi-directional context modelling using the Transformer. Note that BERT_BASE has considerably more parameters than other baselines and our model (110M for BERT_BASE versus 4M for our model), which can be a disadvantage when deployed to devices with limited computing power and memory. The state-of-the-art DialogueRNN model performs the best overall among all baselines. In particular, DialogueRNN performs better than our model on IEMOCAP, which may be attributed to its detailed speaker information for modelling the emotion dynamics in each speaker as the conversation flows.

It is encouraging to see that our KET model outperforms the baselines on most of the datasets tested. This finding indicates that our model is robust across datasets with varying training sizes, context lengths and domains. Our KET variants KET_SingleSelfAttn and KET_StdAttn perform comparably with the best baselines on all datasets except IEMOCAP. However, both variants perform noticeably worse than KET on all datasets except EC, validating the importance of our proposed hierarchical self-attention and dynamic context-aware affective graph attention mechanism. One observation worth mentioning is that these two variants perform on a par with the KET model on EC. Possible explanations are that 1) hierarchical self-attention may not be critical for modelling short conversations in EC, and 2) the informal linguistic styles of Tweets in EC, e.g., misspelled words and slangs, hinder the context representation learning in our graph attention mechanism.

### 5.2 Model Analysis

We analyze the impact of different settings on the validation performance of KET. All results in this section are averaged over 5 random seeds.

**Analysis of context length:** We vary the context length $M$ and plot model performance in Figure 3 (top portion). Note that EC has only a maximum number of 2 contextual utterances. It is clear that incorporating context into KET improves performance on all datasets. However, adding more context is contributing diminishing performance gain or even making negative impact in some datasets. This phenomenon has been observed in a prior study (Su et al., 2018). One possible explanation is that incorporating long contextual information may introduce additional noises, e.g., polysemes expressing different meanings in different utterances of the same context. More thorough investigation of this diminishing return phenomenon is a worthwhile direction in the future.

**Analysis of the size of ConceptNet:** We vary the size of ConceptNet by randomly keeping only a fraction of the concepts in ConceptNet when train-

| Dataset | 0 | 0.3 | 0.7 | 1 |
|---|---|---|---|---|
| EC | 0.7345 | 0.7397 | **0.7426** | 0.7363 |
| DailyDialog | 0.5365 | 0.5432 | **0.5451** | 0.5383 |
| MELD | 0.5321 | **0.5395** | 0.5366 | 0.5306 |
| EmoryNLP | 0.3528 | **0.3624** | 0.3571 | 0.3488 |
| IEMOCAP | 0.5344 | **0.5367** | 0.5314 | 0.5251 |

Table 4: Analysis of the relatedness-affectiveness tradeoff on the validation sets. Each column corresponds to a fixed $\lambda_k$ for all concepts (see Equation 8).

| Dataset | KET | -context | -knowledge |
|---|---|---|---|
| EC | **0.7451** | 0.7343 | 0.7359 |
| DailyDialog | **0.5544** | 0.5282 | 0.5402 |
| MELD | **0.5401** | 0.5177 | 0.5248 |
| EmoryNLP | **0.3712** | 0.3564 | 0.3553 |
| IEMOCAP | **0.5389** | 0.4976 | 0.5217 |

Table 5: Ablation study for KET on the validation sets.

ing and evaluating our model. The results are illustrated in Figure 3 (bottom portion). Adding more concepts consistently improves model performance before reaching a plateau, validating the importance of commonsense knowledge in detecting emotions. We may expect the performance of our KET model to improve with the growing size of ConceptNet in the future.

**Analysis of the relatedness-affectiveness tradeoff:** We experiment with different values of $\lambda_k \in [0, 1]$ (see Equation 8) for all $k$ and report the results in Table 4. It is clear that $\lambda_k$ makes a noticeable impact on the model performance. Discarding relatedness or affectiveness completely will cause significant performance drop on all datasets, with one exception of IEMOCAP. One possible reason is that conversations in IEMOCAP are emotional dialogues, therefore, the affectiveness factor in our proposed graph attention mechanism can provide more discriminative power.

**Ablation Study:** We conduct ablation study to investigate the contribution of context and knowledge as reported in Table 5. It is clear that both context and knowledge are essential to the strong performance of KET on all datasets. Note that removing context has a greater impact on long conversations than short conversations, which is expected because more contextual information is lost in long conversations.

### 5.3 Error Analysis

Despite the strong performance of our model, it still fails to detect certain emotions on certain datasets. We rank the F1 score of each emotion per dataset and investigate the emotions with the worst scores. We found that disgust and fear are generally difficult to detect and differentiate. For example, the F1 score of fear emotion in MELD is as low as 0.0667. One possible cause is that these two emotions are intrinsically similar. The VAD values of both emotions have low valence, high arousal and low dominance (Mehrabian, 1996). Another cause is the small amount of data available for these two emotions. How to differentiate intrinsically similar emotions and how to effectively detect emotions using limited data are two challenging directions in this field.

## 6 Conclusion

We present a knowledge-enriched transformer to detect emotions in textual conversations. Our model learns structured conversation representations via hierarchical self-attention and dynamically refers to external, context-aware, and emotion-related knowledge entities from knowledge bases. Experimental analysis demonstrates that both contextual information and commonsense knowledge are beneficial to model performance. The tradeoff between relatedness and affectiveness plays an important role as well. In addition, our model outperforms the state-of-the-art models on most of the tested datasets of varying sizes and domains.

Given that there are similar emotion lexicons to NRC_VAD in other languages and ConceptNet is a multilingual knowledge base, our model can be easily adapted to other languages. In addition, given that NRC_VAD is the only emotion-specific component, our model can be adapted as a generic model for conversation analysis.

## Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments. This research is supported, in part, by the National Research Foundation, Prime Ministers Office, Singapore under its AI Singapore Programme (Award Number: AISG-GC-2019-003) and under its NRF Investigatorship Programme (NRFI Award No. NRF-NRFI05-2019-0002). This research is also supported, in part, by the Alibaba-NTU Singapore Joint Research Institute, Nanyang Technological University, Singapore.

## References


Muhammad Abdul-Mageed and Lyle Ungar. 2017. Emonet: Fine-grained emotion detection with gated recurrent neural networks. In *ACL*, volume 1, pages 718–728.

Nabiha Asghar, Pascal Poupart, Jesse Hoey, Xin Jiang, and Lili Mou. 2018. Affective neural response generation. In *ECIR*, pages 154–166. Springer.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Anil Bandhakavi, Nirmalie Wiratunga, Stewart Massie, and Deepak Padmanabhan. 2017. Lexicon generation for emotion detection from text. *IEEE Intelligent Systems*, 32(1):102–108.

Yoshua Bengio, Patrice Simard, Paolo Frasconi, et al. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

Carlos Busso, Murtaza Bulut, Chi-Chun Lee, Abe Kazemzadeh, Emily Mower, Samuel Kim, Jeannette N Chang, Sungbok Lee, and Shrikanth S Narayanan. 2008. IEMOCAP: Interactive emotional dyadic motion capture database. *Language Resources and Evaluation*, 42(4):335.

Ankush Chatterjee, Umang Gupta, Manoj Kumar Chinnakotla, Radhakrishnan Srikanth, Michel Galley, and Puneet Agrawal. 2019. Understanding emotions in text using deep learning and big data. *Computers in Human Behavior*, 93:309 – 317.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. In *EMNLP*, pages 551–561.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP*, pages 1724–1734.

Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

Laurence Devillers, Ioana Vasilescu, and Lori Lamel. 2002. Annotation and detection of emotion in a task-oriented human-human dialog corpus. In *Proceedings of ISLE Workshop*.

Laurence Devillers and Laurence Vidrascu. 2006. Real-life emotions detection with lexical and paralinguistic cues on human-human call center dialogs. In *Ninth International Conference on Spoken Language Processing*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2019. Wizard of wikipedia: Knowledge-powered conversational agents. In *ICLR*.

Paul Ekman. 1992. An argument for basic emotions. *Cognition & emotion*, 6(3-4):169–200.

Christiane Fellbaum. 2012. Wordnet. *The Encyclopedia of Applied Linguistics*.

Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen-tau Yih, and Michel Galley. 2018. A knowledge-grounded neural conversation model. In *AAAI*.

Sangdo Han, Jeesoo Bang, Seonghan Ryu, and Gary Geunbae Lee. 2015. Exploiting knowledge base to generate responses for natural language dialog listening agents. In *Proceedings of the 16th SIGDIAL*, pages 129–133.

Yanchao Hao, Yuanzhe Zhang, Kang Liu, Shizhu He, Zhanyi Liu, Hua Wu, and Jun Zhao. 2017. An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge. In *ACL*, pages 221–231.

Devamanyu Hazarika, Soujanya Poria, Rada Mihalcea, Erik Cambria, and Roger Zimmermann. 2018a. Icon: Interactive conversational memory network for multimodal emotion detection. In *EMNLP*, pages 2594–2604.

Devamanyu Hazarika, Soujanya Poria, Amir Zadeh, Erik Cambria, Louis-Philippe Morency, and Roger Zimmermann. 2018b. Conversational memory network for emotion recognition in dyadic dialogue videos. In *NAACL*, volume 1, pages 2122–2132.

Junqing He, Bing Wang, Mingming Fu, Tianqi Yang, and Xuemin Zhao. 2019. Hierarchical attention and knowledge matching networks with information enhancement for end-to-end task-oriented dialog systems. *IEEE Access*, 7:18871–18883.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Yajie Hu, Xiaoou Chen, and Deshun Yang. 2009. Lyric-based song emotion detection with affective lexicon and fuzzy clustering method. In *ISMIR*, pages 123–128.

Eva Hudlicka. 2011. Guidelines for designing computational models of emotions. *International Journal of Synthetic Emotions*, 2(1):26–79.

Hamed Khanpour and Cornelia Caragea. 2018. Fine-grained emotion detection in health-related online posts. In *EMNLP*, pages 1160–1166.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally coherent text generation with neural checklist models. In *EMNLP*, pages 329–339.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Abhishek Kumar, Daisuke Kawahara, and Sadao Kurohashi. 2018. Knowledge-enriched two-layered attention network for sentiment analysis. In *NAACL*, volume 2, pages 253–258.

Chul Min Lee and Shrikanth S Narayanan. 2005. Toward detecting emotions in spoken dialogs. *IEEE Transactions on Speech and Audio Processing*, 13(2):293–303.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. Dailydialog: A manually labelled multi-turn dialogue dataset. In *IJCNLP*, volume 1, pages 986–995.

Shuman Liu, Hongshen Chen, Zhaochun Ren, Yang Feng, Qun Liu, and Dawei Yin. 2018. Knowledge diffusion for neural dialogue generation. In *ACL*, pages 1489–1498.

Andrea Madotto, Chien-Sheng Wu, and Pascale Fung. 2018. Mem2seq: Effectively incorporating knowledge bases into end-to-end task-oriented dialog systems. In *ACL*, volume 1, pages 1468–1478.

Navonil Majumder, Soujanya Poria, Devamanyu Hazarika, Rada Mihalcea, Alexander Gelbukh, and Erik Cambria. 2019. Dialoguernn: An attentive rnn for emotion detection in conversations. In *AAAI*.

Albert Mehrabian. 1996. Pleasure-arousal-dominance: A general framework for describing and measuring individual differences in temperament. *Current Psychology*, 14(4):261–292.

Todor Mihaylov and Anette Frank. 2018. Knowledgeable reader: Enhancing cloze-style reading comprehension with external commonsense knowledge. In *ACL*, pages 821–832.

Nikita Moghe, Siddhartha Arora, Suman Banerjee, and Mitesh M Khapra. 2018. Towards exploiting background knowledge for building conversation systems. In *EMNLP*, pages 2322–2332.

Saif Mohammad. 2018a. Obtaining reliable human ratings of valence, arousal, and dominance for 20,000 english words. In *ACL*, pages 174–184.

Saif M. Mohammad. 2018b. Word affect intensities. In *LREC*.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *EMNLP*, pages 79–86. Association for Computational Linguistics.

Prasanna Parthasarathi and Joelle Pineau. 2018. Extending neural generative conversational model using external knowledge sources. In *EMNLP*, pages 690–695.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Soujanya Poria, Erik Cambria, Devamanyu Hazarika, Navonil Majumder, Amir Zadeh, and Louis-Philippe Morency. 2017. Context-dependent sentiment analysis in user-generated videos. In *ACL*, volume 1, pages 873–883.

Soujanya Poria, Devamanyu Hazarika, Navonil Majumder, Gautam Naik, Erik Cambria, and Rada Mihalcea. 2018. Meld: A multimodal multi-party dataset for emotion recognition in conversations. *arXiv preprint arXiv:1810.02508*.

Soujanya Poria, Navonil Majumder, Rada Mihalcea, and Eduard Hovy. 2019. Emotion recognition in conversation: Research challenges, datasets, and recent advances. *arXiv preprint arXiv:1905.02947*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Yi Luan Mirella Lapata Rik Koncel-Kedziorski, Dhanush Bekal and Hannaneh Hajishirzi. 2019. Text Generation from Knowledge Graphs with Graph Transformers. In *NAACL*.

Armin Seyeditabari, Narges Tabari, and Wlodek Zadrozny. 2018. Emotion detection in text: a review. *arXiv preprint arXiv:1806.00674*.

Dinghan Shen, Guoyin Wang, Wenlin Wang, Martin Renqiang Min, Qinliang Su, Yizhe Zhang, Chunyuan Li, Ricardo Henao, and Lawrence Carin. 2018. Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms. In *ACL*, pages 440–450.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *AAAI*.

Shang-Yu Su, Pei-Chieh Yuan, and Yun-Nung Chen. 2018. How time matters: Learning time-decay attention for contextual spoken language understanding in dialogues. In *NAACL*, volume 1, pages 2133–2142.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen. 2018. Open domain question answering using early fusion of knowledge bases and text. In *EMNLP*, pages 4231–4242.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-based methods for sentiment analysis. *Computational Linguistics*, 37(2):267–307.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*, pages 5998–6008.

Petar Velikovi, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Li, and Yoshua Bengio. 2018. Graph attention networks. In *ICLR*.

Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *ACL*, pages 90–94. Association for Computational Linguistics.

Chien-Sheng Wu, Richard Socher, and Caiming Xiong. 2019. Global-to-local memory pointer networks for task-oriented dialogue. In *ICLR*.

Tom Young, Erik Cambria, Iti Chaturvedi, Hao Zhou, Subham Biswas, and Minlie Huang. 2018. Augmenting end-to-end dialogue systems with commonsense knowledge. In *AAAI*.

Sayyed M Zahiri and Jinho D Choi. 2018. Emotion detection on tv show transcripts with sequence-based convolutional neural networks. In *Workshops at AAAI*.

Jiacheng Zhang, Huanbo Luan, Maosong Sun, Feifei Zhai, Jingfang Xu, Min Zhang, and Yang Liu. 2018a. Improving the transformer translation model with document-level context. In *EMNLP*, pages 533–542.

Yuxiang Zhang, Jiamei Fu, Dongyu She, Ying Zhang, Senzhang Wang, and Jufeng Yang. 2018b. Text emotion distribution learning via multi-task convolutional neural network. In *IJCAI*, pages 4595–4601.

Peixiang Zhong and Chunyan Miao. 2019. ntuer at SemEval-2019 task 3: Emotion classification with word and sentence representations in RCNN. In *SemEval*, pages 282–286.

Peixiang Zhong, Di Wang, and Chunyan Miao. 2019. An affect-rich neural conversational model with biased attention and weighted cross-entropy loss. In *AAAI*, pages 7492–7500.

Hao Zhou, Minlie Huang, Tianyang Zhang, Xiaoyan Zhu, and Bing Liu. 2018a. Emotional chatting machine: Emotional conversation generation with internal and external memory. In *AAAI*.

Hao Zhou, Tom Young, Minlie Huang, Haizhou Zhao, Jingfang Xu, and Xiaoyan Zhu. 2018b. Commonsense knowledge aware conversation generation with graph attention. In *IJCAI*, pages 4623–4629.

Xiangyang Zhou, Lu Li, Daxiang Dong, Yi Liu, Ying Chen, Wayne Xin Zhao, Dianhai Yu, and Hua Wu. 2018c. Multi-turn response selection for chatbots with deep attention matching network. In *ACL*, volume 1, pages 1118–1127.

# Interpretable Relevant Emotion Ranking with Event-Driven Attention

**Yang Yang**[†] **Deyu Zhou**[*†] **Yulan He**[§] **Meng Zhang**[†]
[†]School of Computer Science and Engineering, Key Laboratory of Computer Network and Information Integration, Ministry of Education, Southeast University, China
[§]Department of Computer Science, University of Warwick, UK
{yyang, d.zhou}@seu.edu.cn, y.he@cantab, m.zhang@seu.edu.cn.

## Abstract

Multiple emotions with different intensities are often evoked by events described in documents. Oftentimes, such event information is hidden and needs to be discovered from texts. Unveiling the hidden event information can help to understand how the emotions are evoked and provide explainable results. However, existing studies often ignore the latent event information. In this paper, we proposed a novel interpretable relevant emotion ranking model with the event information incorporated into a deep learning architecture using the event-driven attentions. Moreover, corpus-level event embeddings and document-level event distributions are introduced respectively to consider the global events in corpus and the document-specific events simultaneously. Experimental results on three real-world corpora show that the proposed approach performs remarkably better than the state-of-the-art emotion detection approaches and multi-label approaches. Moreover, interpretable results can be obtained to shed light on the events which trigger certain emotions.

## 1 Introduction

The advent and prosperity of social media enable users to share their opinions, feelings and attitudes online. Apart from directly expressing their opinions on social media posts, users can also vote for their emotional states after reading an article online. An example of a news article crawled from Sina News Society Channel together with its associated emotion votes received from readers is illustrated in Figure 1. Treating these emotion votes as labels for the news article, we can define the emotion detection problem as an emotion ranking problem that ranks emotions based on their intensities. Moreover, some of the emotion labels

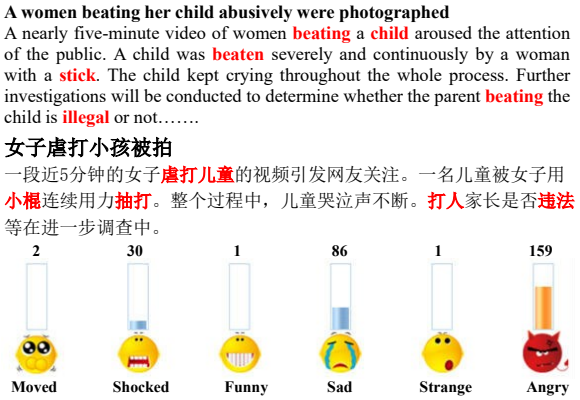


Figure 1: An example of an online news article with the readers' votes on various emotion categories. Words highlighted in red are event-indicative words.

could be considered as irrelevant emotions. For example, the emotion categories '*Moved*', '*Funny*' and '*Strange*' in Figure 1 only received one or two votes. These emotion votes could be noises (e.g., readers accidentally clicked on a wrong emotion button) and hence can be considered as irrelevant emotions. We need to separate the relevant emotions from irrelevant ones and only predict the ranking results for the relevant emotion labels. Therefore, the task we need to perform is the relevant emotion ranking. Understanding and automatically ranking users' emotional states would be potentially useful for downstream applications such as dialogue systems (Picard and Picard, 1997). Multiple emotion detection from texts has been previously addressed in (Zhou et al., 2016) which predicted multiple emotions with different intensities based on emotion distribution learning. A relevant emotion ranking framework was proposed in (Yang et al., 2018) to predict multiple relevant emotions as well as the rankings based on their intensities. However, existing emotion detection approaches do not model the events in texts

*Corresponding author



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 177–187, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

which are crucial for emotion detection. Moreover, most of the existing approaches only produce emotion classification or ranking results, and they do not provide interpretations such as identifying which event triggers a certain emotion.

We argue that emotions may be evoked by latent events in texts. Let us refer back to the example shown in Figure 1 and read the text more carefully. We notice that words such as '*beat*', '*child*' and '*stick*' marked in red are event-related words indicating the event of "*child abuse*" which may evoke the emotions of "*Anger*", "*Sadness*" and "*Shock*".

The above example shows that it is important to simultaneously consider the latent events in texts for relevant emotion ranking. In this paper we proposed an interpretable relevant emotion ranking model with event-driven attention (IRER-EA). We focus on relevant emotion ranking (RER) by discriminating relevant emotions from irrelevant ones and only learn the rankings of the relevant emotions based on their intensities.

Our main contributions are summarized below:

- A novel interpretable relevant emotion ranking model with event-driven attention (IRER-EA) is proposed. The latent event information is incorporated into a deep learning architecture through event-driven attentions which can provide clues of how the emotions are evoked with interpretable results. To the best of our knowledge, it is the first deep event-driven neural approach for RER.

- To consider event information comprehensively, corpus-level event embeddings are incorporated to consider global events in corpus and document-level event distributions are incorporated to learn document-specific event-related attention respectively.

- Experimental results on three different real-world corpora show that the proposed method performs better than the state-of-the-art emotion detection methods and multi-label learning methods. Moreover, the event-driven attention enables dynamically highlighting important event-related parts evoking the emotions in texts.

## 2 Related Work

In general, emotion detection methods can mainly be categorized into two classes: lexicon-based methods and learning-based methods. Lexicon-based approaches utilize emotion lexicons including emotion words and their emotion labels for detecting emotions from texts. For example, emotion lexicons are used in (Aman and Szpakowicz, 2007) to distinguish emotional and non-emotional sentences. Emotion dictionaries could also be used to predict the readers' emotion of new articles (Rao et al., 2012; Lei et al., 2014). Wang et al. (2015) proposed a model with several constraints using non-negative matrix factorization based on emotion lexicon for multiple emotion detection. However, these approaches often suffer from low recall.

Learning-based approaches can be further categorized into unsupervised and supervised learning methods. Unsupervised learning approaches do not require labeled training data (Blei et al., 2003). Supervised learning methods typically frame emotion detection as a classification problem by training supervised classifiers from texts with emotion categories (Rao et al., 2014; Wang and Pal, 2015; Rao, 2016). Lin et al. (2008) studied the readers' emotion detection with various kinds of feature sets on news articles. Quan et al. (2015) detected emotions from texts with a logistic regression model introducing the intermediate hidden variables to model the latent structure of input text corpora. Zhou et al. (2016) predicted multiple emotions with intensities based on emotion distribution learning. A relevant label ranking framework for emotion detection was proposed to predict multiple relevant emotions as well as the rankings of emotions based on their intensities (Yang et al., 2018; Zhou et al., 2018). However, these approaches do not model the latent events in texts.

In recent years, deep neural network models have been widely used for text classification. In particular, the attention-based recurrent neural networks (RNNs) (Schuster and Paliwal, 2002; Yang et al., 2016) prevail in text classification. However, these approaches ignore the latent events in texts thus fail to attend on event-related parts. Moreover, they are lack of interpretation.

Our work is partly inspired by (Yang et al., 2018) for relevant emotion ranking, but with the following significant differences: (1) our model incorporates corpus-level event embeddings and document-level event distributions by an event-driven attention mechanism attending to event-related words, which are ignored in the mod-

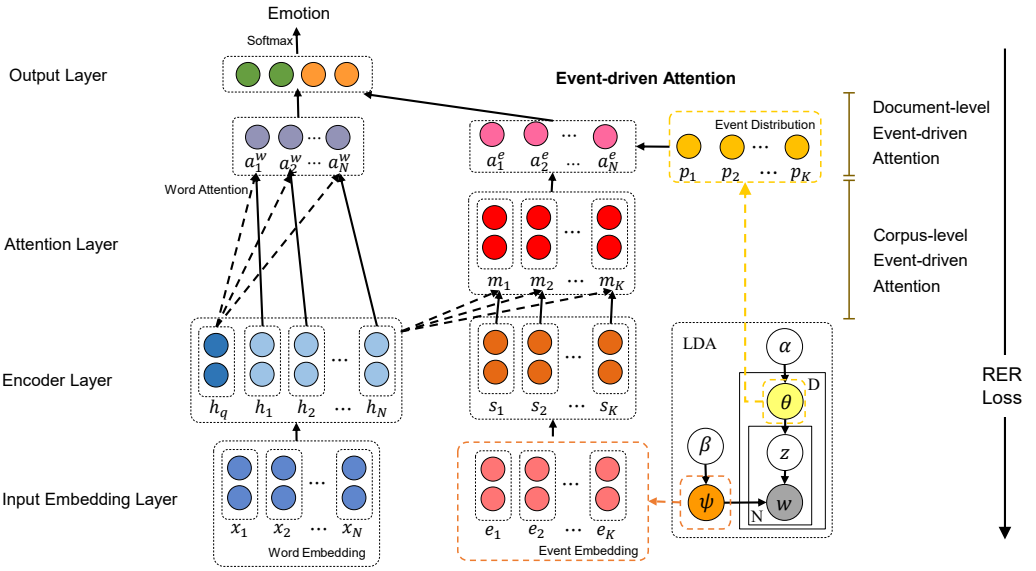


Figure 2: The IRER-EA model.

el (Yang et al., 2018) simply using a Kullback-Leibler (KL) divergence to approximatively learning the documents' topic distributions; (2) our model incorporates the event information into a deep learning architecture thus can consider the sequential information of texts which is ignored in the model (Yang et al., 2018) based on shallow bag-of-words representations.

## 3 The IRER-EA Model

### 3.1 Problem Setting

Assuming a set of $T$ emotions, $L = \{l_1, l_2, ...l_T\}$, and a set of $Q$ document instances, $\mathbb{D} = \{d_1, d_2, d_3, ..., d_Q\}$, each instance $d_i$ is associated with a list of its relevant emotions $R_i \subseteq L$ ranked by their intensities and also a list of irrelevant emotions $\overline{R_i} = L - R_i$. Relevant emotion ranking aims to learn a score function $\mathbf{g}(d_i) = [g_1(d_i), ..., g_T(d_i)]$ which assigns a score $g_j(d_i)$ to each emotion $l_j, (j \in \{1, ..., T\})$. Relevant emotions and their rankings can be obtained simultaneously according to the scores assigned by the learned ranking function $\mathbf{g}$.

The learning objective of relevant emotion ranking (RER) is to both discriminate relevant emotions from irrelevant ones and to rank relevant emotions according to their intensities. Therefore, to fulfil the requirements of RER, the global objective function is defined as follows:

$$E = \sum_{i=1}^{Q} \sum_{l_t \in R_i} \sum_{l_s \in \prec(l_t)} \frac{1}{norm_{t,s}} [\exp(-(g_t(d_i) - g_s(d_i))) + \omega_{ts}(g_t(d_i) - g_s(d_i))^2] \quad (1)$$

Here, $l_s \in \prec (l_t)$ represents that emotion $l_s$ is less relevant than emotion $l_t$. The normalization term $norm_{t,s}$ is used to avoid terms dominated by the sizes of emotion pairs. The term $g_t(d_i) - g_s(d_i)$ measures the difference between two emotions. $\omega_{ts}$ represents the relationship between two emotions $l_t$ and $l_s$ which is calculated by Pearson correlation coefficient (Nicewander, 1988).

We present the overall architecture of the proposed interpretable relevant emotion ranking with event-driven attention (IRER-EA) model in Figure 2. It consists of four layers: (1) the *Input Embedding Layer* including both word embeddings and event embeddings; (2) the *Encoder Layer* including both the word encoder and the event encoder; (3) the *Attention Layer* which computes the word-level attention scores and the event-driven attention scores taking into account of the corpus-level and document-level event information, respectively; (4) the *Output Layer* which generates the emotion ranking results.

### 3.2 Input Embedding Layer

The *Input Embedding Layer* contains word embeddings and event embeddings. Assuming a document $d_i$ consisting of $N$ words represented as

$d_i = \{w_1, w_2, ..., w_N\}$, the pre-trained word vector, GloVe (Pennington et al., 2014), is used to obtain the fixed word embedding of each word and $d_i$ can be represented as $d_i = \{x_1, x_2, ..., x_N\}$ as shown in Figure 2.

Since nouns and verbs are more important than other word types in referring to specific events, they are utilized as inputs of topic model such as Latent Dirichlet Allocation (LDA) (Blei et al., 2003) to generate events automatically. Therefore, the granularity of extracted events is controlled by the predefined K, the number of events. For the corpus $\mathbb{D}$ consisting of $K$ events $\{e_1, e_2, ..., e_K\}$, the event embedding of the $k$th event $e_k$ can be obtained from the output event-word distribution matrix $E$ of the topic model. For the single document $d_i$, the event distribution $p = (p_1, p_2, ..., p_K)$ obtained from the topic model represents the probability of the text expressing each event.

### 3.3 Encoder Layer

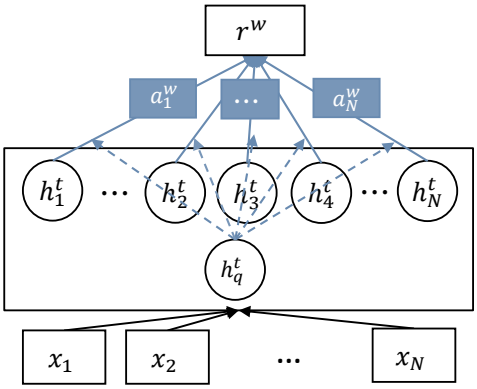


Figure 3: Word Encoder.

The *Encoder Layer* contains both the word encoder and event encoder. As for the word encoder, an alternative RNN structure (Zhang et al., 2018) is used to encode texts into semantic representations since it has been shown to be more effective in encoding longer texts. For document $d_i$, formally, a state at time step $t$ can be denoted by:

$$H^t = < h_1^t, ..., h_N^t, h_q^t > \quad (2)$$

which consists of sub-states $h_i^t$ for the $i$th word $w_i$ in document $d_i$ and a document-level sub-state $h_q^t$ as shown in Figure 3. The hidden states are independent of each other at the present recurrent step and are connected across recurrent steps, which can capture long-range dependencies. The recurrent state transition process is used to model information exchange between those sub-states to enrich state representations incrementally. The state transition is similar to LSTM (Hochreiter and Schmidhuber, 1997) and a recurrent cell $c_i^t$ for each word $w_i$ and a cell $c_q^t$ for document-level sub-state $h_q$ are used. The value of each $h_i^t$ is computed based on the values of $x_i$, $h_{i-1}^{t-1}$, $h_i^{t-1}$, $h_{i+1}^{t-1}$, $h_q^{t-1}$ at two adjacent recurrent time steps. Note that the number of window size between two adjacent steps can be set manually. Hence, the hidden sub-states $h_i$ for individual words $w_i$ and a global document hidden state $h_q$ for $d_i$ are obtained.

As for event encoder, event representations are produced by the ReLU-actived neural perceptrons taking the event-word weight matrix $E \in V \times K$ as inputs. Hence, each event representation $s_k$ representing event $k$ is obtained according to the event embedding $e_k$, $k \in \{1, 2, 3, ..., K\}$.

### 3.4 Attention Layer

Given a word $w_n$ in document $d_i$, $h_n$ is the hidden representation of $w_n$ after encoder. Given an event embedding $e_k$ in the corpus, $s_k$ is the event representation of $e_k$ generated by the event encoder. Then we utilize attention weights to enhance the word representations and event representations from different aspects.

Our model contains two kinds of attention mechanisms including the word-level attentions and the event-driven attentions.

#### 3.4.1 Word-Level Attention

As for word-level attentions, since not all words contribute equally to the meaning of a document, we introduce an attention mechanism to extract words with greater importance and aggregate the representations of those informative words to form the document representation, which is shown in the left part of Figure 2. More concretely,

$$\begin{aligned} \varphi_{w_i} &= tanh(W_w(h_i + h_q) + b_w) \\ a_i^w &= \frac{exp(\varphi_{w_i}^\top u_w)}{\sum_i exp(\varphi_{w_i}^\top u_w)} \\ r^w &= \sum_i a_i^w h_i \end{aligned} \quad (3)$$

where the weight $a_i^w$ is the attention of the word $w_i$ and $W_w, b_w$ and $u_w$ are parameters similar to (Pappas and Popescu-Belis, 2017). Note that we further incorporate the global information of the document representation $h_q$ obtained from the encoder to strengthen the word attention.

### 3.4.2 Event-Driven Attention

In our model, we use the event-driven attention mechanism to attend to event-related words, which can discover words more important for text-related events. The event-driven attention leverages the corpus-level event information based on each event representation $s_k, k \in \{1, 2, 3, ..., K\}$ obtained from the corpus and the document-level event information based on the document's event distribution $p = (p_1, p_2, ..., p_K)$.

**Corpus-level Event-Driven Attention**

The model utilizes the corpus-level event information by a joint attention mechanism to consider global events in corpus, which aggregates the semantic representations $h = (h_1, h_2, ..., h_N)$ of an input text obtained and measures the interaction the words in the text with the event representations $s = (s_1, s_2, ..., s_K)$ by the event-driven attention. The corpus-level event-driven attention is calculated as follows:

$$\begin{aligned} \varphi^c &= tanh(W_c h + b_c) \\ m_k^c &= (\varphi^c)^\top s_k \\ a^c &= softmax(\sum_{k=1}^{K} m_k^c) \\ r^c &= (a^c)^\top h \end{aligned} \tag{4}$$

where $h = (h_1, h_2, ..., h_N)^\top$stands for the combination of all the hidden states of words in the document and $W_c$ and $b_c$ are parameters needed to be learnt for corpus-level event-driven attention. $\varphi^c = (\varphi_1^c, \varphi_2^c, ..., \varphi_N^c)$ refers to the hidden representation of state $h$ through a fully connected layer. Given the event representation $s_k$, we measure the interaction of the words in the document and the event by an attention weight vector $m_k^c$ which can be computed as the inner product of event $s_k$ and $\varphi^c$ followed by a softmax layer. $a^c = (a_1^c, a_2^c, ..., a_N^c)$ stands for the average attention weights of all the events for words which contribute to discover event keywords of a document according to different events in corpus. Then we construct the text representation $r^c$ with the sum of hidden states weighted by $a^c$.

**Document-level Event-driven Attention**

We further incorporate the document-level event-driven attention mechanism. Our model can attend to the event distributions of the current document in order to strengthen the effect of the current document expressing each event and learn document-specific event related attention. For each document, $p = (p_1, p_2, ..., p_K)$ denotes the event distributions of the document, with each dimension representing the level of prominence of the corresponding event occurred in the document. The corpus-level event-driven attention weights can be further strengthened by including document-level event distributions. The document-level event-driven attention is calculated as follows:

$$\begin{aligned} \varphi^d &= tanh(W_d h + b_d) \\ m_k^d &= (\varphi^d)^\top s_k \\ a^e &= softmax(\sum_{k=1}^{K} m_k^d p_k) \\ r^e &= (a^e)^\top h \end{aligned} \tag{5}$$

where $h = (h_1, h_2, ..., h_N)^\top$ stands for the aggregation of all the hidden states of words in the document and $W_d$ and $b_d$ are parameters needed to be learnt for the document-level event-driven attention. $\varphi^d = (\varphi_1^d, \varphi_2^d, ..., \varphi_N^d)$ refers to the hidden representation of state $h$ through a fully connected layer. $m_k^d$ represents the interaction of the words in the document and the event which can be computed as the inner product of event $s_k$ and $\varphi^d$. Then $m_k^d$ is weighted by the document-level event distribution, $p = (p_1, p_2, ..., p_K)$, followed by a softmax layer, and $a^e = (a_1^e, a_2^e, ..., a_N^e)$ stands for the attention weight after incorporating the document-level event distributions. Then we construct the text representation $r^e$ with the sum of hidden states weighted by $a^e$. Finally, $r^e$ is used as the final text representation obtained by the event-driven attention which simultaneously takes into account both the corpus-level event information and the document-level event information.

### 3.5 Output Layer

At last, we concatenate both the representations calculated by the word-level attention and the event-driven attention to obtain the final representation $r = [r^w, r^e]$, which is fed to a multi-layer perceptron and a softmax layer for identifying relevant emotions and their rankings.

## 4 Experiments

To evaluate our proposed approach, we conducted experiments on the following three corpora:

**Sina Social News (News)** (Zhou et al., 2018) consists of 5,586 news articles collected from the Sina

news *Society* channel. Each document was kept together with the readers' emotion votes of the six emotions including *Funny*, *Moved*, *Angry*, *Sad*, *Strange*, and *Shocked*.

**Ren-CECps corpus (Blogs)** (Quan and Ren, 2010) is a Chinese data set containing 1,487 blogs annotated with eight basic emotions from writer's perspective, including *Anger*, *Anxiety*, *Expect*, *Hate*, *Joy*, *Love*, *Sorrow* and *Surprise*. The emotions are represented by their emotion scores in the range of $[0, 1]$. Higher scores represent higher emotion intensities.

**SemEval** (Strapparava and Mihalcea, 2007) contains 1,250 English news headlines extracted from Google news, CNN, and many other portals, which are manually annotated with a fine-grained valence scale of 0 to 100 across 6 emotions, including *Anger*, *Disgust*, *Fear*, *Joy*, *Sad* and *Surprise*.

| News | | Blogs | | SemEval | |
|---|---|---|---|---|---|
| Category | #Votes | Category | #Scores | Category | #Scores |
| Touching | 694,006 | Joy | 349.2 | anger | 12042 |
| Shock | 572,651 | Hate | 174.2 | disgust | 7634 |
| Amusement | 869,464 | Love | 610.6 | fear | 20306 |
| Sadness | 837,431 | Sorrow | 408.4 | joy | 23613 |
| Curiosity | 212,559 | Anxiety | 422.6 | sad | 24039 |
| Anger | 1,109,315 | Surprise | 59.2 | surprise | 21495 |
| | | Anger | 116.4 | | |
| | | Expect | 385.5 | | |
| All | 4,295,426 | All | 2526.1 | All | 109,129 |

Table 1: Statistics for the three corpora used in our experiments.

The statistics for the three corpora used in our experiments are shown in Table 1.

In our experiments, the News and Blog corpora were preprocessed using the python jieba segmenter[1] for word segmentation and filtering. The third corpus SemEval is in English and was tokenized by white space. Stanford CoreNLP[2] was applied for parts of speech tagging to obtain the nouns and verbs of the documents. Stop words and words appeared less than twice were removed from documents. We used the pre-trained Chinese GloVe and English GloVe[3] vectors as the word embeddings in the experiments and the dimension of the word embeddings was 300.

[1] https://github.com/fxsjy/jieba

[2] https://stanfordnlp.github.io/CoreNLP/index.html

[3] https://nlp.stanford.edu/projects/glove/

| Name | Definition |
|---|---|
| PRO Loss | $\frac{1}{n}\sum_{i=1}^{n}\sum_{e_t\in R_i\cup\{\Theta\}}\sum_{e_s\in\prec(e_t)}\frac{1}{norm_{t,s}}l_{t,s}$<br>$l_{t,s}$ is a modified 0-1 error;<br>$norm_{t,s}$is the set size of label pair$(e_t, e_s)$ |
| Hamming Loss | $\frac{1}{nT}\sum_{i=1}^{n}\lvert\hat{R}_i\triangle R_i\rvert$<br>The predicted relevant emotions: $\hat{R}_i$. |
| Ranking Loss | $\frac{1}{n}\sum_{i=1}^{n}\frac{(\sum_{(e_t,e_s)\in R_i\times\overline{R_i}}\delta[g_t(x_i)<g_s(x_i)])}{(\lvert R_i\rvert\times\lvert\overline{R_i}\rvert)}$<br>where $\delta$ is the indicator function. |
| One Error | $\frac{1}{n}\sum_{i=1}^{n}\delta[\underset{e_t}{\operatorname{argmax}}\, g_t(x_i)\notin R_i]$ |
| Average Precision | $\frac{1}{n}\sum_{i=1}^{n}\frac{1}{\lvert R_i\rvert}\times\sum_{t:e_t\in R_i}\frac{\lvert\{e_s\in R_i\lvert g_s(x_i)>g_t(x_i)\}\rvert}{\lvert\{e_s\lvert g_s(x_i)>g_t(x_i)\}\rvert}$ |
| Coverage | $\frac{1}{n}\sum_{i=1}^{n}\max_{t:e_t\in R_i}\lvert\{e_s\lvert g_s(x_i)>g_t(x_i)\}\rvert$ |
| $F1_{exam}$ | $\frac{1}{n}\sum_{i=1}^{n}\frac{2\lvert R_i\cap\hat{R}_i\rvert}{(\lvert R_i\rvert+\lvert\hat{R}_i\rvert)}$ |
| MicroF1 | $F1(\sum_{t=1}^{T}TP_t,\sum_{t=1}^{T}FP_t,\sum_{t=1}^{T}TN_t,\sum_{t=1}^{T}FN_t)$ |
| MacroF1 | $\frac{1}{T}\sum_{t=1}^{T}F1(TP_t,FP_t,TN_t,FN_t)$ |

Table 2: Evaluation criteria for the Multi-Label Learning (MLL) methods. $TP_t, FP_t, TN_t, FN_t$ represent the number of true positive, false positive, true negative, and false negative test examples with respect to emotion $t$ respectively. $F1(TP_t, FP_t, TN_t, FN_t)$ represent specific binary classification metric F1 (Manning et al., 2008).

The event embeddings and event distributions used in the proposed method are derived in different ways. For long documents including News and Blogs, LDA was employed to generate event embeddings and event distributions using verbs and nouns as the input. For short texts in SemEval with the sparsity problem, Bi-term Topic Model (BTM) (Cheng et al., 2014) was chosen. The number of topics was 60. The parameters were chosen from the validation set which is 10% of the training set. The encoder was trained using a learning rate of 0.001, a dropout rate of 0.5, a window size of 1 and a layer number of 3. The number of epochs was 10 and the mini batch (Cotter et al., 2011) size was 16. For each method, 10-fold cross validation was conducted to get the final results.

The baselines can be categorized into two classes, emotion detection methods and multi-label methods. Most these baselines are either re-implemented or cited from published papers. For instance, the results of multi-label methods are re-implemented, since they are not proposed for relevant emotion ranking. The performances of some emotion detection methods, such as EDL, EmoDetect, RER and INNRER, are cited from the pub-

| Corpus | Category | Method | Criteria | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | PL(↓) | HL(↓) | RL(↓) | OE(↓) | AP(↑) | Cov(↓) | F1(↑) | MiF1(↑) | MaF1(↑) |
| News | Emotion Detection | EDL | 0.2348 | 0.2510 | 0.1616 | 0.2243 | 0.8372 | 2.1940 | 0.6260 | 0.6454 | 0.5703 |
| | | EmoDetect | 0.2157 | 0.2575 | 0.1538 | 0.1627 | 0.8605 | 2.1761 | 0.6697 | 0.6739 | 0.5359 |
| | | RER | 0.2142 | 0.2498 | 0.1491 | 0.1513 | 0.8633 | 2.1989 | 0.6820 | 0.6919 | 0.6198 |
| | | INN-RER | 0.1973 | 0.2312 | 0.1353 | 0.1331 | 0.8764 | 2.1339 | 0.7108 | 0.7161 | 0.6282 |
| | Multi-label | LIFT | 0.2224 | 0.3363 | 0.1382 | 0.1411 | 0.8234 | 2.1394 | 0.6646 | 0.6801 | 0.6151 |
| | | Rank-SVM | 0.2842 | 0.2872 | 0.2114 | 0.2034 | 0.7967 | 2.5358 | 0.5066 | 0.5656 | 0.5298 |
| | | MLLOC | 0.4458 | 0.4206 | 0.4500 | 0.4193 | 0.6531 | 3.9032 | 0.3000 | 0.4060 | 0.3327 |
| | | BP-MLL | 0.2118 | 0.2399 | 0.1443 | 0.1544 | 0.8677 | 2.1738 | 0.6881 | 0.6915 | 0.6013 |
| | Our model | **IRER-EA** | **0.1674** | **0.1958** | **0.0989** | **0.0846** | **0.9137** | **1.9108** | **0.7475** | **0.7379** | **0.6358** |
| Blogs | Emotion Detection | EDL | 0.3385 | 0.3916 | 0.2550 | 0.4206 | 0.6962 | 4.2491 | 0.5060 | 0.5396 | 0.4131 |
| | | EmoDetect | 0.3115 | 0.3848 | 0.2123 | 0.2880 | 0.7617 | 4.1650 | 0.5340 | 0.5492 | 0.4387 |
| | | RER | 0.3007 | 0.3657 | 0.2043 | 0.2728 | 0.7746 | 4.1638 | 0.5957 | 0.6084 | **0.5342** |
| | | INN-RER | 0.2829 | 0.3209 | 0.1924 | 0.2626 | 0.7784 | 3.6418 | 0.6187 | 0.6225 | 0.5133 |
| | Multi-label | LIFT | 0.3452 | 0.3817 | 0.3089 | 0.3306 | 0.7557 | **3.1290** | 0.6053 | 0.6113 | 0.5155 |
| | | Rank-SVM | 0.3888 | 0.3786 | 0.3356 | 0.3219 | 0.7030 | 4.0801 | 0.3489 | 0.3686 | 0.3210 |
| | | MLLOC | 0.4999 | 0.5135 | 0.5115 | 0.6554 | 0.5569 | 5.7432 | 0.4104 | 0.4411 | 0.4391 |
| | | BP-MLL | 0.2987 | 0.3281 | 0.2141 | 0.2727 | 0.7267 | 3.9802 | 0.5844 | 0.6065 | 0.4833 |
| | Our model | **IRER-EA** | **0.2609** | **0.2874** | **0.1638** | **0.1845** | **0.8149** | 3.6510 | **0.6304** | **0.6320** | 0.4804 |
| SemEval | Emotion Detection | EDL | 0.4130 | 0.4291 | 0.3401 | 0.3875 | 0.7345 | 3.3433 | 0.4002 | 0.413 | 0.3813 |
| | | EmoDetect | 0.3176 | 0.3167 | 0.2411 | 0.2308 | 0.8241 | 3.0439 | 0.6275 | 0.6245 | 0.5385 |
| | | RER | 0.2907 | 0.3128 | 0.2389 | 0.2220 | 0.8302 | 2.9963 | 0.6839 | 0.6898 | 0.6283 |
| | | INN-RER | 0.3194 | **0.3005** | 0.2302 | 0.2261 | 0.8379 | **2.8632** | 0.7081 | 0.7156 | 0.6093 |
| | Multi-label | LIFT | 0.4279 | 0.4651 | 0.3627 | 0.4113 | 0.7344 | 3.2823 | 0.6299 | 0.6469 | 0.6112 |
| | | Rank-SVM | 0.3452 | 0.3617 | 0.3083 | 0.3006 | 0.7557 | 3.1290 | 0.6253 | 0.6472 | 0.5955 |
| | | MLLOC | 0.4458 | 0.4206 | 0.4500 | 0.4193 | 0.6531 | 3.9032 | 0.3000 | 0.4060 | 0.3327 |
| | | BP-MLL | 0.3790 | 0.3656 | 0.3605 | 0.3790 | 0.7945 | 3.2097 | 0.5868 | 0.6101 | 0.5402 |
| | Our model | **IRER-EA** | **0.2754** | 0.3065 | **0.1999** | **0.1286** | **0.8976** | 3.5799 | **0.7394** | **0.7538** | **0.6841** |

Table 3: Comparison with Emotion Detection Methods and Multi-label Methods. 'PL' represent Pro Loss, 'HL' represents Hamming Loss, 'RL' represents ranking loss, 'OE' represents one error, 'AP' represent average precision, 'Cov' represent coverage, 'F1' represents $F1_{exam}$, 'MiF1' represents MicroF1, 'MaF1' represents MacroF1. "↓" indicates "the smaller the better", while "↑" indicates "the larger the better". The best performance on each evaluation measure is highlighted by boldface.

lished paper (Yang et al., 2018) as they use the same experimental data as ours.

Evaluation metrics typically used in multi-label learning and label ranking are employed in our experiments which are different from those of classical single-label learning systems (Sebastiani, 2001). The detailed explanations of evaluation metrics are presented in Table 2.

### 4.1 Compared Methods

There are several emotion detection approaches addressing multiple emotions detection from texts.

- **Emotion Distribution Learning(EDL)** (Zhou et al., 2016) learns a mapping function from sentences to their emotion distributions.
- **EmoDetect** (Wang and Pal, 2015) employs a constraint optimization framework with several constraints to obtain multiple emotions.
- **RER** (Zhou et al., 2018) uses support vector machines to predict relevant emotions and rankings in one text based on their intensities.
- **INN-RER** (Yang et al., 2018) designs a three-layer network combined with topic model to solve relevant emotion ranking.

Relevant emotion ranking can be treated as an extension of multi-label learning, so we also com-

| Corpus | Category | Method | Criteria | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | PL(↓) | HL(↓) | RL(↓) | OE(↓) | AP(↑) | Cov(↓) | F1(↑) | MiF1(↑) | MaF1(↑) |
| News | Baselines | IRER-EA(-EA) | 0.1786 | 0.1981 | 0.1023 | 0.0854 | 0.9122 | 1.9294 | 0.7423 | 0.7299 | 0.6116 |
| | | IRER-EA(-DEA) | 0.1700 | **0.1947** | 0.1092 | **0.0835** | 0.9086 | 2.0014 | 0.7437 | 0.7358 | 0.6038 |
| | Our model | **IRER-EA** | **0.1674** | 0.1958 | **0.0989** | 0.0846 | **0.9137** | **1.9108** | **0.7475** | **0.7379** | **0.6358** |
| Blogs | Baselines | IRER-EA(-EA) | 0.2860 | 0.3010 | 0.1819 | 0.2004 | 0.7816 | 3.8320 | 0.5534 | 0.5422 | 0.3805 |
| | | IRER-EA(-DEA) | 0.2852 | 0.3077 | 0.1844 | 0.1986 | 0.7936 | 3.8290 | 0.6027 | 0.6053 | 0.4220 |
| | Our model | **IRER-EA** | **0.2609** | **0.2874** | **0.1638** | **0.1845** | **0.8149** | **3.6510** | **0.6304** | **0.6320** | **0.4804** |
| SemEval | Baselines | IRER-EA(-EA) | 0.2760 | **0.2968** | 0.2076 | 0.1916 | 0.8565 | **2.8235** | 0.6494 | 0.6491 | 0.5480 |
| | | IRER-EA(-DEA) | 0.2879 | 0.3220 | 0.2205 | 0.1534 | 0.8795 | 3.2893 | 0.6852 | 0.6891 | 0.5927 |
| | Our model | **IRER-EA** | **0.2754** | 0.3065 | **0.1999** | **0.1286** | **0.8976** | 3.5799 | **0.7394** | **0.7538** | **0.6841** |

Table 4: Comparison of different IRER-EA components."↓" indicates "the smaller the better", while "↑" indicates "the larger the better". The best performance on each evaluation measure is highlighted by boldface.

pare the proposed IRER-EA with several widely-used multi-label learning methods.

- **LIFT** (Zhang, 2011) constructs features specific to each label.
- **Rank-SVM** (Zhang and Zhou, 2014) distinguishes relevant labels from irrelevant ones with large margin strategy.
- **MLLOC** (Huang and Zhou, 2012) exploits local emotion correlations in expression data.
- **BP-MLL** (Zhang and Zhou, 2006) employs a novel error function into back propagation algorithm to capture the characteristics of multi-label learning.

For the MLL methods, linear kernel is used in LIFT. Rank-SVM uses the RBF kernel with the width $\sigma$ equals to 1.

Experimental results on three corpora are shown in Table 3. It can be summarized from the table that: (1) IRER-EA performs better than state-of-art emotion detection baselines on almost all evaluation metrics across three corpora, which obviously shows the effectiveness of incorporating event information to obtain event-driven attentions for relevant emotion ranking; (2) IRER-EA achieves remarkably better results than MLL methods. It further confirms the effectiveness of IRER-EA, which uses a deep learning architecture incorporating event-driven attention for better performance.

### 4.2 Model Analysis

To further validate the effectiveness of event-driven attention components, we compare IRER-EA with two sub-networks based on our architectures.

- **IRER-EA(-EA)**: removes event-driven attention from IRER-EA.
- **IRER-EA(-DEA)**: removes document-level event-driven attention from IRER-EA.

Experimental results on three corpora are shown in Table 4. It can be summarized from the table that: (1) All the sub-networks cannot compete with IRER-EA on three corpora, which indicates the corpus-level and document-level event information are effective for relevant emotion ranking task; (2) IRER-EA(-DEA) performs better than IRER-EA(-EA) on most of the evaluation metrics, which verifies the effectiveness of incorporating corpus-level event-driven attention; (3) IRER-EA achieves better results than IRER-EA(-DEA) on almost all the evaluation metrics which further proves the effectiveness of document-level event-driven attention.

### 4.3 Case Study of Interpretability

To further investigate whether the event-driven attention is able to capture the event-associated words in a given document and provide interpretable results, we compare the results of the attention mechanisms of the word-level attentions and event-driven attentions by visualizing the weights of words in the same documents as shown in Figure 4. As the document of the News corpus and Blogs corpus are too long, we manually simplified the texts for better visualization results and provided English translations of the texts. The words marked in red represent highly-weighted ones according to the event-driven attentions, while the words with blue underlines are the

| Corpora | Texts | Emotions |
|---|---|---|
| News | 古琴台附近一名女子**轻生溺亡**。从她走进水里到最终**溺亡**，持续时间将近两分钟。当时有 10 余人**围观**，竟**无人**下水**施救**。这种现象让我们**寒心**。<br>A woman **drowned** near Guqin Terrace. It took nearly two minutes from she stepping into the water to her final **death**. More than 10 people were **watching** her, but **no one** came to **rescue**. Such phenomenon **chills** us. | Angry<br>Sad<br>Shocked |
| Blog | **地震**过后两百个小时里，每一张前沿的照片，每一段**获救**的视频，每个爸爸妈妈发自心底的痛哭，都令我心痛。<br>During the two hundred hours after the **earthquake**, each photo and video about **rescue** process and the **crying** from the parents make my heart broken. | Sorrow<br>Love |
| SemEval | teacher in hide after **attack** on islam stir **threat** | Fear<br>Sad |

Figure 4: Case Study of Interpretability on Three Corpora.

ones with higher attention weights according to the word-level attentions.

From the visualization results on an example News article, it can be observed that different from word-level attention that pays more attentions to emotion-associated words, such as '**chill**' which may only evoke the emotion "*Sad*", the event-driven attentions can find words indicating latent events in the document, such as '**drown**', '**death**', '**no one rescue**' which are all closely related to the event "*Suicide without rescue*', which may evoke emotions such as "*Angry*" and "*Shocked*". In an example Blog article, word-level attentions highlight emotion-associated words such as '**crying**' and '**broken**' which may evoke the emotion "*Sorrow*", while event-driven attentions focus on the event-related words such as '**earthquake**' and '**rescued**' representing the event "*Earthquake Relief*'. Finally, in an example from the SemEval corpus, we can see that the word-level attention mechanism only gives a higher attention weight to the word '**threat**' and ignores the word '**attack**', which is also an important indicator of the emotions "*Fear*" and "*Sad*". On the contrary, the event-driven attention mechanism highlights both '**attack**' and '**threat**', representing the event "*Terrorist attack*'.

In summary, we can observe from Figure 4 that: (1) Event-driven attention can capture words representing latent events in texts; (2) Compared with the word-level attention which is prone to attend on emotion-associated keywords, event-driven attention can find words representing one or more hidden events in a document, which can provide more explainable clues of which event triggers certain emotions; (3) Event-driven attention can achieve better performance especially in documents without any emotion-associated words.

## 5 Conclusion

In this paper, we have proposed an interpretable relevant emotion ranking model with event-driven attention. The event information is incorporated into a neural model through event-driven attentions which can provide clues of how the emotions are evoked with explainable results. Moreover, corpus-level event embeddings and document-level event distributions are incorporated respectively to consider event information comprehensively. Experimental results show that the proposed method performs better than the state-of-the-art emotion detection methods and multi-label learning methods.

## Acknowledgments

We would like to thank anonymous reviewers for their valuable comments and helpful suggestions. This work was funded by the National Key Research and Development Program of China (2017YFB1002801), the National Natural Science Foundation of China (61772132), the Natural Science Foundation of Jiangsu Province of China (BK20161430) and Innovate UK (grant no. 103652).

## References


Saima Aman and Stan Szpakowicz. 2007. Identifying expressions of emotion in text. *Lecture Notes in Computer Science*, 4629:196–205.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Xueqi Cheng, Xiaohui Yan, Yanyan Lan, and Jiafeng Guo. 2014. Btm: Topic modeling over short texts. *IEEE Transactions on Knowledge and Data Engineering*, 26(12):2928–2941.

Andrew Cotter, Ohad Shamir, Nathan Srebro, and Karthik Sridharan. 2011. Better mini-batch algorithms via accelerated gradient methods. *Advances in Neural Information Processing Systems*, pages 1647–1655.

S Hochreiter and J Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Sheng Jun Huang and Zhi Hua Zhou. 2012. Multi-label learning by exploiting label correlations locally. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, pages 949–955.

Jingsheng Lei, Yanghui Rao, Qing Li, Xiaojun Quan, and Liu Wenyin. 2014. Towards building a social emotion detection system for online news. *Future Generation Computer Systems*, 37:438–448.

Kevin Hsin-Yih Lin, Changhua Yang, and Hsin-Hsi Chen. 2008. Emotion classification of online news articles from the reader's perspective. In *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology-Volume 01*, pages 220–226. IEEE Computer Society.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. 2008. An introduction to information retrieval. *Journal of the American Society for Information Science and Technology*, 43(3):824–825.

W. Alan Nicewander. 1988. Thirteen ways to look at the correlation coefficient. *American Statistician*, 42(1):59–66.

Nikolaos Pappas and Andrei Popescu-Belis. 2017. Multilingual hierarchical attention networks for document classification. *CoRR*, abs/1707.00896.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing*, pages 1532–1543.

Rosalind W Picard and Roalind Picard. 1997. *Affective computing*, volume 252. MIT press Cambridge.

Changqin Quan and Fuji Ren. 2010. Sentence emotion analysis and recognition based on emotion words using ren-cecps. *International Journal of Advanced Intelligence Paradigms*, 2(1):105–117.

Xiaojun Quan, Qifan Wang, Ying Zhang, Luo Si, and Liu Wenyin. 2015. Latent discriminative models for social emotion detection with emotional dependency. *ACM Trans. Inf. Syst.*, 34(1):2:1–2:19.

Yanghui Rao. 2016. Contextual sentiment topic model for adaptive social emotion classification. *IEEE Intelligent Systems*, 31(1):41–47.

Yanghui Rao, Qing Li, Xudong Mao, and Wenyin Liu. 2014. Sentiment topic models for social emotion mining. *Information Sciences*, 266(5):90–100.

Yanghui Rao, Xiaojun Quan, Liu Wenyin, Qing Li, and Mingliang Chen. 2012. Building word-emotion mapping dictionary for online news. In *SDAD 2012 The 1st International Workshop on Sentiment Discovery from Affective Data*, page 28.

M. Schuster and K.K. Paliwal. 2002. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Fabrizio Sebastiani. 2001. Machine learning in automated text categorization. *Acm Computing Surveys*, 34(1):1–47.

Carlo Strapparava and Rada Mihalcea. 2007. Semeval-2007 task 14: Affective text. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 70–74. Association for Computational Linguistics.

Yichen Wang and Aditya Pal. 2015. Detecting emotions in social media: A constrained optimization approach. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)*, pages 996–1002.

Yang Yang, Deyu Zhou, and Yulan He. 2018. An interpretable neural network with topical information for relevant emotion ranking. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3423–3432.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. pages 1480–1489.

Min Ling Zhang. 2011. Lift: multi-label learning with label-specific features. In *International Joint Conference on Artificial Intelligence*, pages 1609–1614.

Min Ling Zhang and Zhi Hua Zhou. 2006. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE Transactions on Knowledge Data Engineering*, 18(10):1338–1351.

Min-Ling Zhang and Zhi-Hua Zhou. 2014. A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering*, 26(8):1819–1837.

Yue Zhang, Qi Liu, and Linfeng Song. 2018. Sentence-state lstm for text representation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 317–327. Association for Computational Linguistics.

Deyu Zhou, Yang Yang, and Yulan He. 2018. Relevant emotion ranking from text constrained with emotion relationships. In *Meeting of the North American Chapter of the Association for Computation Linguistics*, pages 561–571.

Deyu Zhou, Xuan Zhang, Yin Zhou, Quan Zhao, and Xin Geng. 2016. Emotion distribution learning from texts. In *Conference on Empirical Methods in Natural Language Processing*, pages 638–647.

# Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects

**Jianmo Ni, Jiacheng Li, Julian McAuley**
University of California, San Diego
{jin018,j9li,jmcauley}@ucsd.edu

## Abstract

Several recent works have considered the problem of generating reviews (or 'tips') as a form of *explanation* as to why a recommendation might match a user's interests. While promising, we demonstrate that existing approaches struggle (in terms of both quality and content) to generate *justifications* that are relevant to users' decision-making process. We seek to introduce new datasets and methods to address this *recommendation justification* task. In terms of data, we first propose an 'extractive' approach to identify review segments which justify users' intentions; this approach is then used to distantly label massive review corpora and construct large-scale personalized recommendation justification datasets. In terms of generation, we design two personalized generation models with this data: (1) a reference-based Seq2Seq model with aspect-planning which can generate justifications covering different aspects, and (2) an aspect-conditional masked language model which can generate diverse justifications based on templates extracted from justification histories. We conduct experiments on two real-world datasets which show that our model is capable of generating convincing and diverse justifications.

## 1 Introduction

Explaining, or justifying, recommendations to users has the potential to increase their transparency and reliability. However providing meaningful interpretations remains a difficult task, partly due to the black-box nature of many recommendation models, but also because we simply lack ground-truth datasets specifying what 'good' justifications ought to look like.

Previous work has sought to learn user preferences and writing styles from crowd-sourced reviews (Dong et al., 2017; Ni and McAuley, 2018) to generate explanations in the form of natural language, e.g. generating synthesized reviews similar to those that users would write about a product. However, a large portion of review text (or text from 'tips') is often of little relevance to most users' decision making (e.g. they describe verbose experiences or general endorsements) and may not be appropriate to use as explanations in terms of content and language style. As a result, existing models that learn directly from reviews (or tips) may not capture crucial information that explains users' purchases. Table 1 shows examples of reviews, tips and ideal justifications. More recently, there has been work studying the task of tip generation where tips are concise summaries of reviews (Li et al., 2017). Though tips are concise and some subset of them might be suitable as candidates for recommendation justifications, only a few e-commerce systems provide tips accompanied with reviews. Even in systems where tips are available, the number of tips is usually far smaller than the number of reviews. These approaches hence suffer from generalizability issues, especially in settings

| Review examples: |
|---|
| I love this little stand! **The coconut mocha chiller and caramel macchiato are delicious.** |
| Wow what a special find. One of the most unique and special date nights my husband and I have had. |
| **Tip examples:** |
| **Great food. Nice ambiance. Gnocchi were very good.** |
| I can't get enough of this place. |
| **Justification examples:** |
| **The food portions were huge.** |
| **Plain cheese quesadilla is very good and very cheap.** |

Table 1: In contrast to reviews and tips, we seek to automatically generate *recommendation justifications* that are more concise, concrete, and helpful for decision making. Examples of justifications from reviews, tips, and our annotated dataset are marked in bold.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 188–197, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

where user interactions are highly sparse.

On the other hand, generating *diverse* responses is essential in personalized content generation scenarios such as justification generation. Instead of always predicting the most popular reasons, it's preferable to present diverse justifications for different users based on their personal interests. Recent work has shown that incorporating prior knowledge into generation frameworks can greatly improve diversity. Prior knowledge could include story-lines in story generation (Yao et al., 2019), or historical responses in dialogue systems (Weston et al., 2018).

**In this work**, our goal is to generate convincing and diverse justifications. To address the challenge of lacking ground-truth data about 'good' justifications, we propose a pipeline that can identify justifications from massive corpora of reviews or tips. We extract fine-grained aspects from justifications and build user personas and item profiles consisting of sets of representative aspects. To improve generation quality and diversity, we propose two generation models (1) a reference-based Seq2Seq model with aspect-planning, which takes previous justifications as a reference and can produce justifications based on different aspects, and (2) an aspect-conditional masked language model that can generate diverse justifications from templates extracted from previous justifications.

Our contributions are threefold:

- To facilitate recommendation justification generation, we propose a pipeline to identify justification candidates and build aspect-based user personas and item profiles from massive corpora of reviews. With this approach, we are able to build large-scale personalized justification datasets. We use these extractive justification segments in the task of explainable recommendation and show that these are better training sources instead of whole reviews.
- We propose two models based on reference attention, aspect-planning techniques and a persona-conditional masked language model. We show that adding such personalized information enables the models to generate justifications with high quality and diversity.
- We conduct extensive experiments on two real-world datasets from *Yelp* and *Amazon Clothing*. We provide an annotated dataset about 'good' justifications on the Yelp dataset and show that the binary classifier trained on this dataset generalizes well to the Amazon Clothing dataset. We study different decoding strategies and compare their effect on generation performance.

## 2 Dataset Generation

In this section, we introduce the pipeline to extract high quality justifications from raw user reviews. Specifically, our goal here is to identify review segments that can be used as justifications and build a personalized justification dataset upon them. Our pipeline consists of three steps:

1. Annotating a set of review segments with binary labels, *i.e.,* to determine whether they are 'good' or 'bad' justifications.
2. Training a classifier on the annotated subset and applying it to distantly label all the review segments to extract 'good' justifications for each user and item pair.
3. Applying fine-grained aspect extraction for the extracted justifications, and building user personas and item profiles.

### 2.1 Identifying Justifications From Reviews

The first step is to extract text segments from reviews that are appropriate to use as justifications. Instead of a complete sentence or phrase, we define each segment as an Elementary Discourse Unit (EDU; (Mann and Thompson, 1988)) which corresponds to a sequence of clauses. We use the model of Wang et al. (2018) to obtain EDUs from reviews. Recent works have shown that EDUs can improve the performance of document-level summarization (Bhatia et al., 2015) and opinion summarizaiton (Angelidis and Lapata, 2018).

After preprocessing the reviews into EDUs, we analyzed the linguistic differences between recommendation justifications and reviews, and built two rules to filter the segments that are unlikely to be suitable justifications: (1) segments with first-person or third-person pronouns, and (2) too long or short. Next, two expert annotators were exposed to 1,000 segments among those not filtered out and asked to determine whether they are 'good' justifications. Labeling was performed iteratively, followed by feedback and discussion, until the quality was aligned between the two annotators. At the end of the process, the inter-annotator agreement for the binary labeling task (*good* vs. *bad*), measured by Cohen's kappa (Cohen, 1960), was 0.927 after alignment. Then, the annotators further labeled 600 segments. Overall,

| Method | F1 | Recall | Precision |
|---|---|---|---|
| BOW-Xgboost | 0.559 | 0.679 | 0.475 |
| CNN | 0.644 | 0.596 | 0.700 |
| LSTM-MaxPool | 0.675 | 0.703 | 0.650 |
| BERT | **0.747** | 0.700 | **0.800** |
| BERT-SA (one epoch) | 0.481 | 0.975 | 0.320 |
| BERT-SA (three epoch) | 0.491 | **1.000** | 0.325 |

Table 2: Performance for classifying review segments as good or bad for recommendation justification.

24.8% of the segments were labeled *good*.

### 2.2 Automatic Classification

Our next step is to propagate labels to the complete review corpus. Here we adopt BERT (Devlin et al., 2019) to fine-tune on our classification task, where a [CLS] token is added to the beginning of each segment and the final hidden state (i.e., output of BERT) corresponding to this token is fed into a linear layer to obtain the binary prediction. Cross entropy is used as the training loss.

We split the annotated dataset into Train, Dev, and Test sets with a 0.8/0.1/0.1 ratio, fine-tune the BERT classifier on the Train set and choose the best model on the Dev set. After three epochs of fine-tuning, BERT can achieve an F1-score of 0.80 on the Test set. We compare the performance of BERT with multiple baseline models: (1) a XGBoost model which uses Bags-of-Words as sentence features (2) a convolutional neural network (CNN) with three convolution layers and one linear layer (3) a long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) network with a max-pooling layer, and a linear layer (4) a BERT sentiment classifier (BERT-SA) trained on the complete Yelp dataset for one epoch and three epochs. To obtain the pre-trained word embeddings for the CNN and LSTM models, we applied fastText (Bojanowski et al., 2016) on the *Yelp* Review dataset. We set the embedding dimension to 200 and used default values for other hyper-parameters.

Table 2 presents results for our binary classification task. The BERT classifier has higher F1-score and precision than other classifiers. The BERT-SA model after three epochs only achieves an F1-score of 0.491, which confirms the difference between sentiment analysis and our *good/bad* task, i.e., even if the segment has positive sentiment, it might be not suitable as a justification.

| Yelp |
|---|
| The *Tuna* is pretty amazing |
| *Appetizers* and *pasta* are excellent here |
| An excellent *selection* of both *sweet* and savory *crepes* |
| It was filled with delicious *food*, fantastic *music* and *dancing* |
| **Amazon-Cloth** |
| The *quality* of the *material* is great |
| Great *shirt*, especially for the *price*. |
| The *seams* and *stitching* are really nice |
| *Fit* the bill for a *Halloween costume*. |

Table 3: Examples of justifications with fine-grained aspects in our annotated dataset. The fine-grained aspects are italic and underlined.

### 2.3 Fine-grained Aspect Extraction

Finally, we extract the fine-grained aspects that each justification covers. Fine-grained aspects are properties of products that appear among a user's opinions. We adopt the method proposed by Zhang et al. (2014) to build a sentiment lexicon which includes a set of fine-grained aspects from the whole dataset. We then use simple rules to determine which aspects appear in each justification.[1] Table 3 presents a set of examples from our dataset. Each example consists of a justification that a user has written about an item, and multiple fine-grained aspects mentioned in the justification. **Note that** we only annotated the Yelp dataset, trained a classifer on that and applied the model on both Yelp and Amazon Clothing dataset. As shown in Table 3, the trained classifier works well on both datasets.

## 3 Approach

### 3.1 Problem Definition

For each user $u$ (or item $i$), we build a justification reference $D = \{d_1, \ldots, d_{l_r}\}$ consisting of justifications that the user has written (or justifications about the item) on the training set, where $l_r$ is the maximum number of justifications. We also obtain a user persona (or item profile) $A = \{a_1, \ldots, a_K\}$ based on the fine-grained aspects that the user's (or item's) previous justifications have covered, where $K$ is the maximum number of aspects.

Given a user $u$ and an item $i$, as well as the their justification reference $D_u$ and $D_i$, and $u$'s persona $A_u$ and $i$'s profile $A_i$, **our target** is to predict the

[1]For each aspect, if its singular or plural exists in the tokenized justification, then we consider that this aspect exists in that justification.

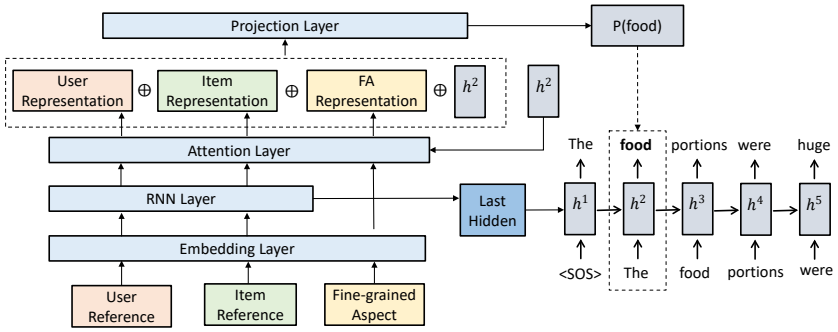


Figure 1: Structure of the reference-based Seq2Seq model with Aspect Planning

justifications $J_{u,i} = \{w_1, w_2, \ldots, w_T\}$ that would explain why item $i$ fits user $u$'s interests, where $T$ is the length of the justification.

### 3.2 Reference-based Seq2Seq Model

Our base model follows the structure of a standard Seq2Seq (Sutskever et al., 2014) model. Our framework, called 'Ref2Seq', views the historical justifications of users and items as references and learns latent personalized features from them. Figure 1 shows the structure of our Reference-based Seq2Seq Model. It includes two components: (1) two sequence encoders that learn user and item latent representations by taking previous justifications as references; (2) a sequence decoder incorporating representations from users and items to generate personalized justifications.

**Sequence Encoders**. Our user encoder and sequence encoder share the same structure, which includes an embedding layer, a two-layer bi-directional GRU (Cho et al., 2014), and a projection layer. The inputs are a user (or item) reference $D$ consisting of a set of historical justifications. These justifications pass a word embedding layer, then go through the GRU and yield a sequence of hidden states $\mathbf{e} \in \mathbb{R}^{l_s \times l_r \times n}$:

$$\mathrm{E} = \mathrm{Embedding}(D), \mathbf{e} = \mathrm{GRU}(\mathrm{E}) = \overrightarrow{\mathrm{e}} + \overleftarrow{\mathrm{e}}, \tag{1}$$

where $l_s$ denotes the length of the sequence, $n$ is the hidden size of the encoder GRU, $\mathrm{E} \in \mathbb{R}^{l_s \times l_r \times n}$ is the embedded sequence representation, and $\overrightarrow{e}$ and $\overleftarrow{e}$ are the hidden vectors produced by a forward and a backward GRU (respectively).

To combine information from different 'references' (i.e. justifications), the hidden states are then projected via a linear layer:

$$\hat{\mathbf{e}} = W_e \cdot \mathbf{e} + \mathbf{b}_e, \tag{2}$$

where $\hat{\mathbf{e}} \in \mathbb{R}^{l_s \times n}$ is the final output of the encoder, and $W_e \in \mathbb{R}^{l_r}$, $\mathbf{b}_e \in \mathbb{R}$ are learned parameters.

**Sequence decoder**. The decoder is a two-layer GRU that predicts the target words given a start token. The hidden state of the decoder is initialized using the sum of the last hidden state of the user and item encoders. The hidden state at time-step $t$ is updated via the GRU unit based on the previous hidden state and the input word. Specifically:

$$\mathbf{h}_0 = \mathbf{e}^u_{l_s} + \mathbf{e}^i_{l_s}, \mathbf{h}_t = \mathbf{GRU}(w_t, \mathbf{h}_{t-1}), \tag{3}$$

where $\mathbf{e}^u_{l_s}$ and $\mathbf{e}^i_{l_s}$ are the last hidden states of the user and item encoder output $\hat{\mathbf{e}}_u$ and $\hat{\mathbf{e}}_i$.

To explore the relation between the reference and generation, we apply an attention fusion layer to summarize the output of each encoder. For the user and item reference encoder, the attention vector is defined as:

$$\mathbf{a}^1_t = \sum_{j=1}^{l_s} \alpha^1_{tj} \mathbf{e}_j,$$
$$\alpha^1_{tj} = \exp(\tanh({\mathbf{v}^1_\alpha}^\top (W^1_\alpha [\mathbf{e}_j; \mathbf{h}_t] + \mathbf{b}^1_\alpha)))/Z, \tag{4}$$

where $\mathbf{a}^1_t \in \mathbb{R}^n$ is an attention vector on the sequence encoder at time-step $t$, $\alpha^1_{tj}$ is an attention score over the encoder hidden state $\mathbf{e}_j$ and decoder hidden state $\mathbf{h}_t$, and $Z$ is a normalization term.

**Aspect-Planning Generation**. One of the challenges for generating justifications is how to improve controllability, i.e., directly manipulate the content being generated. Inspired by 'plan-and-write' (Yao et al., 2019), we extend the base model to an Aspect-Planning Ref2Seq (AP-Ref2Seq) model where we plan a fine-grained aspect before generation. This aspect planning can be considered as an extra form of supervision instead of a hard constraint to make justification generation more controllable.

When generating the justification for user $u$ and item $i$, we first provide a fine-grained aspect $a$ as a plan. The aspect $a$ is fed into the word embedding layer to obtain the aspect embedding $\mathrm{E_a}$. Then, we compute the scores between the embedding of the aspect and the decoder hidden state as:

$$\mathbf{a}_t^2 = \alpha_t^2 \mathrm{E}_a,$$
$$\alpha_t^2 = \exp(\tanh(\mathbf{v}_\alpha^2{}^\top (W_\alpha^2[\mathrm{E}_a; \mathbf{h}_t] + \mathbf{b}_\alpha^2)))/Z, \quad (5)$$

where $\mathbf{a}_t^2 \in \mathbb{R}^n$ is an attention vector and $\alpha_t^2$ is an attention score.

The attention vectors $\mathbf{a}_{ut}^1$ of user $u$, $\mathbf{a}_{it}^1$ of item $i$, and $\mathbf{a}_t^2$ of fine-grained aspect $a$, are concatenated with the decoder hidden state at time-step $t$ and projected to obtain the output word distribution $P$. The output probability for word $w$ at time-step $t$ is given by:

$$p(w_t) = \tanh(W_1[\mathbf{h}_t; \mathbf{a}_{ut}^1; \mathbf{a}_{it}^1; \mathbf{a}_t^2] + \mathbf{b}_1), \quad (6)$$

where $w_t$ is the target word at time-step $t$. Given the probability $p(w_t)$ at each time step $t$, the model is trained using a cross-entropy loss compared against the ground-truth sequence.

### 3.3 Aspect Conditional Masked Language Model

Though Seq2Seq-based models can achieve high quality output, they often fail to generate *diverse* content. Recent works in natural language generation (NLG) tried to combine generation methods with information retrieval techniques to increase the generation diversity (Li et al., 2018; Baheti et al., 2018). The basic idea follows the paradigm of retrieve-and-edit—which is to first retrieve historical responses as templates, and then edit the template into new content. Since our data is annotated with fine-grained aspects, it naturally fits into this type of retrieve-and-edit paradigm. Meanwhile, masked language models have shown great performance in language modeling. Recent work (Wang and Cho, 2019; Mansimov et al., 2019) has shown that by sampling from the masked language model (e.g. BERT), it is able to generate coherent sentences.

Inspired by this work, we want to extend such an approach into a conditional version—we explore the use of an Aspect Conditional Masked Language Model (ACMLM) to generate diverse personalized justifications. Figure 2 shows the structure of our Aspect Conditional Masked Language Model. For a justification $J_{u,i}$ that user $u$ wrote about item $i$, we adapt the pre-trained BERT model (Devlin et al., 2019) into an encoder-decoder network with (1) an aspect encoder which encodes the user persona and item profile into latent representations and (2) a masked language model sequence decoder that takes in a masked justification and predicts the masked tokens.

**Aspect Encoder**. Our aspect encoder shares the same WordPiece embeddings (Wu et al., 2016) as BERT. The encoder feeds the intersection of fine-grained aspects from the user persona and item profile $A_{ui} = \{a_1, \ldots, a_{K'}\}$ into the embedding layer and obtains the aspect embedding $\mathbf{A}_{ui} \in \mathbb{R}^{K' \times n}$, where $K'$ is the number of common fine-grained aspects and $n$ is the dimension of the WordPiece embeddings.

**Masked Language Model Sequence Decoder**. We use the masked language model in the pre-trained BERT model as our sequence decoder and add attention over the aspect encoder's output. As shown in Figure 2, the input to the decoder is a masked justification $J_{u,i}^M = \{w_1, \ldots, w_T\}$ with multiple tokens be replaced as [MASK]. The decoder's output $\mathbf{T} \in \mathbb{R}^{T \times n}$ is then fed to the attention layer to calculate an attention score with the output of the encoder:

$$\mathbf{a}_t^3 = \sum_{j=1}^{K'} \alpha_{tj}^3 \mathbf{A}_j,$$
$$\alpha_{tj}^3 = \exp(\tanh(\mathbf{v}_\alpha^3{}^\top (W_\alpha^3[\mathbf{A}_j; \mathbf{T}_t] + \mathbf{b}_\alpha^3)))/Z. \quad (7)$$

The attention vector $\mathbf{a}_t^3$ is then concatenated with the decoder hidden state at time-step $t$ and sent to a linear projection layer to obtain the output word distribution $P$. The output probability for word $w$ at time-step $t$ is given by:

$$p(w_t) = \tanh(W_2[\mathbf{T}_t; \mathbf{a}_t^3] + \mathbf{b}_2) \quad (8)$$

where $w_t$ is the target word at time-step $t$.

**Masking Procedure**. The original BERT paper applies a flat rate (15%) to decide whether to mask a token. Unlike their approach, we adopt a higher rate to mask fine-grained aspects since they are more important in justifications. Specifically, if we encounter a fine-grained aspect, we will replace it with a [MASK] token 30% of the time; while for other words, we will replace them with a [MASK] token 15% of the time.

During training, the model will only predict those masked tokens and calculate a cross-entropy loss on them.

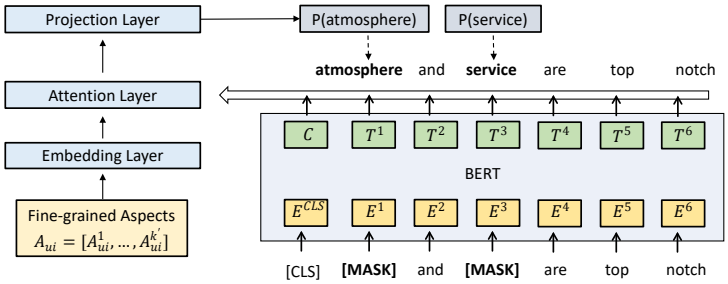


Figure 2: Structure of the Aspect Conditional Masked Language Model

| | |
|---|---|
| Iter 0 | universe [MASK] is extremely friendly and persona ##ble |
| Iter 5 | the [MASK] is extremely friendly and persona ##ble |
| Iter 10 | the [MASK] is extremely friendly and persona ##ble |
| Iter 15 | the staff are extremely cool and persona ##ble |
| Iter 20 | the staff are extra kind , persona ##ble |

Table 4: Examples of the generation output of ACMLM at different iterations.

**Generation by Sampling from Masked Templates**. We next discuss how to generate justifications from the trained ACMLM. We follow the sampling strategy of Wang and Cho (2019) to generate justifications. Instead of generating from a sequence of all [MASK] tokens, we start with masked templates generated from historical justifications about the target item. These masked templates include prior knowledge about the item and can increase the speed of sampling convergence.

Table 4 shows an example of the generation process. We initialize the template sequence $X^0$ as (universe, [MASK], ..., ##ble) with length $T$. At each iteration $i$, a position $t_i$ is sampled uniformly at random from $\{1, \ldots, T\}$ and the token at $t_i$ (i.e. $x^i_{t_i}$) of the current sequence $X^i$ is replaced by [MASK]. After that, we obtain the conditional probability of $x_{t_i}$ as

$$p(x_{t_i}|X^i_{\setminus t_i} = \frac{1}{Z(X^i_{\setminus t_i})} \exp(1\mathrm{h}(x_{t_i})^\top f_\theta(X^i_{\setminus t_i}))), \quad (9)$$

where $1\mathrm{h}(x_{t_i})$ is a one-hot vector with index $x_{t_i}$ set to 1, $X^i_{\setminus t_i}$ is the sequence we obtain after replacing the token at position $t_i$ of $X^i$ by [MASK], $f_\theta(X^i_{\setminus t_i})$ is the output after feeding $X^i_{\setminus t_i}$ into the ACMLM as in Equation (8), and $Z$ is the normalization term. We then sample $\tilde{x}_{ti}$ from Equation (9), and construct the next sequence by $X^{i+1} = (x^i_1, \ldots, \tilde{x}_{ti}, \ldots, x^i_T)$. After repeating this procedure $N$ times, the final output is considered as the generation output.[2]

## 4 Experiments

### 4.1 Datasets

With our proposed pipeline (Section 2), we construct two personalized justification datasets from existing review data—Yelp and Amazon Clothing.[3][4] We further filter those users with fewer than five justifications. For each user, we randomly hold out two samples from all of their justifications to construct the Dev and Test sets. Table 5 shows the statistics of our two datasets.

### 4.2 Baselines

For automatic evaluation, we consider three baselines: Item-Rand is a baseline which randomly chooses a justification from the item's historical justifications. LexRank is a strong unsupervised baseline that is widely used in text summarization (Erkan and Radev, 2004). Given all historical justifications about an item, LexRank can select one justification as the summary. We then use that as the justification for all users. Attr2Seq (Dong et al., 2017) is a Seq2Seq baseline that uses attributes (i.e. user and item identity) as input.

By default, all models use beam search during generation. Recently, there have been works showing that the generation output of sampling methods is more diverse and suitable on high-entropy tasks (Holtzman et al., 2019). To this end, we explore another decoding strategy— 'Top-k sampling' (Radford et al., 2019) in experiments

[2]We set $N$ proportional to the length $T$ of the initial masked template to prevent the generation diverging too much from the original template.

[3]https://www.yelp.com/dataset/challenge

[4]http://jmcauley.ucsd.edu/data/amazon

| Dataset | Train | Dev | Test | # Users | # Items | # Aspects |
|---|---|---|---|---|---|---|
| Yelp | 1,219,962 | 115,907 | 115,907 | 115,907 | 51,948 | 2,041 |
| Amazon Clothing | 202,528 | 57,947 | 57,947 | 57,947 | 50,240 | 581 |

Table 5: Statistics of our datasets.

| Dataset | Yelp | | | | Amazon Clothing | | | |
|---|---|---|---|---|---|---|---|---|
| Model | BLEU-3 | BLEU-4 | Distinct-1 | Distinct-2 | BLEU-3 | BLEU-4 | Distinct-1 | Distinct-2 |
| Item-Rand | 0.440 | 0.150 | 2.766 | 20.151 | 1.620 | 0.680 | 2.400 | 11.853 |
| LexRank | 2.290 | 0.920 | 1.738 | 8.509 | 3.480 | 2.250 | 2.407 | 14.956 |
| Attr2seq | **7.890** | 0.000 | 0.049 | 0.095 | 1.720 | 0.560 | 0.076 | 0.352 |
| Ref2Seq | 4.380 | **2.450** | 0.188 | 1.163 | 8.780 | 5.670 | 0.141 | 1.240 |
| AP-Ref2Seq | 3.390 | 1.830 | 0.326 | 2.094 | **13.910** | **12.500** | 0.557 | 3.661 |
| Ref2Seq (Top-k) | 1.630 | 0.700 | 0.818 | 11.927 | 3.960 | 2.130 | 0.697 | **10.858** |
| ACMLM | 0.700 | 0.280 | **1.322** | **14.319** | 2.420 | 1.590 | **0.942** | 9.312 |

Table 6: Performance on Automatic Evaluation.

and include a variant of our model: Ref2Ref (Top-k).[5]

For human evaluation, we include two baselines: Ref2Seq (Review) and Ref2Seq (Tip), both of which are the same model as Ref2Seq model but trained on the original review and tip data, respectively. Comparisons with these two baselines demonstrates that training on our annotated dataset tends to generate text more suitable as justifications.

| Model | R | I | D |
|---|---|---|---|
| Ref2Seq (Review) | 3.02 | 2.39 | 2.10 |
| Ref2Seq (Tip) | 3.25 | 2.35 | 2.34 |
| Ref2Seq | 3.87 | 3.13 | 2.96 |
| Ref2Seq (Top-k) | **3.95** | **3.34** | 3.39 |
| ACMLM | 3.23 | 3.29 | **3.42** |

Table 7: Performance on Human Evaluation, where R,I,D represents Relevance, Informativeness and Diversity, respectively.

### 4.3 Implementation Detail

We use PyTorch[6] to implement our models.

For Req2Seq and AP-Ref2Seq, we set the hidden size and word embedding size as 256. We apply a dropout rate of 0.5 for the encoder and 0.2 for the decoder. The size of the justification reference $l_r$ is set to 5 and the number of fine-grained aspects $K$ in the user persona and item profile is set to 30. We train the model using Adam with learning rate $2e^{-4}$ and stop training either when it reaches 20 epochs or the perplexity does not improve (on the Dev set). For ACMLM, we build our model based on the BERT implementation from HuggingFace.[7] We initialize our decoder using the pre-trained 'Bert-base' model and set the max sequence length to 30. We train the model for 5 epochs using Adam with learning rate $2e^{-5}$. For models using beam search, we set the beam size as 10. For models using 'top-k' sampling, we set k to 5. For ACMLM, we use a burn-in step equal to the length of the initial sequence. Our data and code are available online.[8]

### 4.4 Automatic Evaluation

For automatic evaluation, we use BLEU, Distinct-1, and Distinct-2 (Li et al., 2015) to measure the performance of our model. As shown in Table 6, our reference-based models achieve the highest BLEU scores on both datasets except for BLEU-3 on Yelp. This confirms that Ref2Seq is able to capture user and item content to generate the most relevant content, compared with unpersonalized models such as LexRank and personalized models that do not leverage historical justifications such as Attr2Seq.

On the other hand, recent works have reported that models achieving higher diversity scores will have lower scores on overlap-based metrics (e.g. BLEU) for open-domain generation tasks (Baheti et al., 2018; Gao et al., 2018). We make a similar observation for our personalized justifi-

[5] At each time step, the next word is sampled from the top $k$ possible next tokens, according to their probabilities.

[6] http://pytorch.org/docs/master/index.html

[7] https://github.com/huggingface/pytorch-pretrained-BERT

[8] https://github.com/nijianmo/recsys_justification.git

| Model | Shake Shack | Teharu Sushi | MGM Grand Hotel |
|---|---|---|---|
| Ground Truth | The burger was good | The rolls are pretty great , typical rolls not that many specials | Room was very clean comfortable |
| LexRank | A great burger and fries. | Sushi ? | Great rooms. |
| Ref2Seq (Review) | i love trader joe 's , i love trader joe 's | the food was good and the service was great | i love this place ! the food is always good and the service is always great |
| Ref2Seq (Tip) | this place is awesome | love this place | come here |
| Ref2Seq | this place has some of the best burgers | the sushi is delicious | the room was nice |
| Ref2Seq (Top-k) | the fries are amazing | fresh and delicious sushi | open hotel for hours |
| ACMLM | breakfast sandwiches are overall very filling | overall fun experience with half price sushi | family style dinner , long time shopping trip to vegas, family dining , cheap lunch |

Table 8: Comparisons of the generated justifications from different models for three businesses on the Yelp dataset.

cation generation task. As shown in Table 6, both sampling-based methods Ref2Seq (Top-k) and ACMLM achieve higher Distinct-1 and Distinct-2, while their BLEU scores are lower than Seq2Seq based models using beam search. Therefore, we also perform human evaluation to validate the generation quality of our proposed methods.

### 4.5 Human Evaluation

We conduct human evaluation on three aspects: (1) *Relevance* measures whether the generated output contains information relevant to an item; (2) *Informativeness* measures whether the generated justification includes specific information that is helpful to users; and (3) *Diversity* measures how distinct the generated output is compared with other justifications.

We focus on the Yelp dataset and sample 100 generated examples from each of the five models as shown in Table 7. Human annotators are asked to give a score in the range [1,5] (lowest to highest) for each metric. Each example is rated by at least three annotators. The results show that both Ref2Seq (Top-k) and ACMLM achieve higher scores on Diversity and Informativeness compared to other models.

### 4.6 Qualitative Analysis

Here we study the following two qualitative questions:

**RQ1: How do training data and methods affect generation?** As Table 8 shows, models trained on reviews and tips tend to generate generic phrases (such as 'i love this place') which often do not include information that helps users to make decisions. Other models trained on the justification datasets tend to mention concrete information (e.g. different aspects). LexRank tends to generate relevant but short content. Meanwhile, sampling-based models are able to generate more diverse content.

| Dataset | Aspects | Generated Output |
|---|---|---|
| Yelp | dining | the dining room is nice |
| | pastry | the pastries were pretty good |
| | chicken | the chicken fried rice is the best |
| | sandwich | the pulled pork sandwich is the best thing on the menu |
| Amazon-Clothing | product | great product , fast shippong |
| | price | design is nice , good price |
| | leather | comfortable leather sneakers . classic |
| | walking | sturdy , great city walking shoes |

Table 9: Generated justifications from AP-Ref2Seq. The planned aspects are randomly selected from users' personas.

**RQ2: How does aspect planning affect generation?** To mitigate the trade-off between diversity and relevance, one approach is to add more constraints during generation such as constrained Beam Search (Anderson et al., 2017). In our work, we extend our base model Ref2Seq by incorporating aspect-planning to guide generation. As shown in Table 9, most planned aspects are present in the generated outputs of AP-Req2Seq.

## 5 Related Work

**Explainable Recommendation** There has been a line of work that studies how to improve the

explainability of recommender systems. Catherine and Cohen (2017) learn latent representations of review text to predict ratings. These representations are then used to find the most helpful reviews for given a particular user and item pair. Another popular direction is to generate text to justify recommendations. Dong et al. (2017) proposed an attribute-to-sequence model to generate product reviews which utilizes categorical attributes. Ni et al. (2017) developed a multi-task learning method that considers collaborative filter and review generation. Li et al. (2019b) generated tips by considering 'persona' information which can capture the language style of users and characteristics of items. However, these works use whole reviews or tips as training examples, which may not be appropriate due to the quality of review text. More recently, Liu et al. (2019) proposed a framework to generate fine-grained explanations for text classification. To achieve labels for human-readable explanations, they constructed a dataset from a website which provides ratings and fine-grained summaries written by users. Unfortunately, most websites do not provide such fine-grained information. On the other hand, our work identifies justifications from reviews, uses them as training examples and shows these are better data source for explainable recommendation via extensive experiments.

**Diversity-aware NLG** Diversity is an important aspect of NLG systems. Recent works have focused on digesting prior knowledge to improve generation diversity. Yao et al. (2019) proposed a method to incorporate planned story-lines in story generation. Li et al. (2019a) developed an aspect-aware coarse-to-fine review generation method. They predict an aspect for each sentence in the review to capture the content flow. Given the aspects, a sequence of sentence sketches is generated and a decoder will fill in the slots of each sketch. In dialogue systems, several works have studied frameworks to extract templates from historical responses, which are then edited to form new responses (Weston et al., 2018; Wu et al., 2018). Similarly, the extract-and-edit paradigm has been studied in style transfer tasks in NLG (Li et al., 2018). Wu et al. (2019) proposed an attribute aware masked language model for non-parallel sentiment transfer. They first mask out the sentimental tokens and then train a masked language model to infill the masked positions for target sentiment. In this work, we also introduces a conditional masked language model but considers more fine-grained aspects.

## 6 Conclusion

In this work, we studied the problem of personalized justification generation. To build high quality justification datasets, we provided an annotated dataset and proposed a pipeline to extract justifications from massive review corpora. To generate convincing and diverse justifications, we developed two models: (1) Ref2Seq which leverages historical justifications as references during generation; and (2) ACMLM, which is an aspect conditional model built on a pre-trained masked language model. Our experiments showed that Ref2Seq achieves higher scores (in terms of BLEU) and ACMLM achieves higher diversity scores compared with baselines. Human evaluation showed that reference-based models obtain high relevance scores and sampling based methods led to more diverse and informative outputs. Finally, we showed that aspect-planning is a promising way to guide generation to produce personlized and relevant justifications.

**Acknowledgements.** This work is partly supported by NSF #1750063. We thank all the reviewers for their constructive suggestions.

## References

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2017. Guided open vocabulary image captioning with constrained beam search. In *EMNLP*.

Stefanos Angelidis and Mirella Lapata. 2018. Summarizing opinions: Aspect extraction meets sentiment prediction and they are both weakly supervised. In *EMNLP*.

Ashutosh Baheti, Alan Ritter, Jiwei Li, and William B. Dolan. 2018. Generating more interesting responses in neural conversation models with distributional constraints. In *EMNLP*.

Parminder Bhatia, Yangfeng Ji, and Jacob Eisenstein. 2015. Better document-level sentiment analysis from rst discourse parsing. In *EMNLP*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Rose Catherine and William W. Cohen. 2017. Transnets: Learning to transform for recommendation. In *RecSys*.

Kyunghyun Cho, Bart van Merrienboer, aglar Gülehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*.

Jacob Willem Cohen. 1960. A coefficient of agreement for nominal scales.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Li Dong, Shaohan Huang, Furu Wei, Mirella Lapata, Ming Zhou, and Ke Xu. 2017. Learning to generate product reviews from attributes. In *EACL*.

Günes Erkan and Dragomir R. Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Intell. Res.*, 22:457–479.

Jun Gao, Wei Bi, Xiaojiang Liu, Junhui Li, and Shuming Shi. 2018. Generating multiple diverse responses for short-text conversation. In *AAAI*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9:1735–1780.

Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. *CoRR*, abs/1904.09751.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and William B. Dolan. 2015. A diversity-promoting objective function for neural conversation models. In *HLT-NAACL*.

Juncen Li, Robin Jia, He He, and Percy S. Liang. 2018. Delete, retrieve, generate: A simple approach to sentiment and style transfer. In *NAACL-HLT*.

Junyi Li, Wayne Xin Zhao, Ji-Rong Wen, and Yang Song. 2019a. Generating long and informative reviews with aspect-aware coarse-to-fine decoding. In *ACL*.

Piji Li, Zihao Wang, Lidong Bing, and Wai Lam. 2019b. Persona-aware tips generation. In *WWW*.

Piji Li, Zihao Wang, Zhaochun Ren, Lidong Bing, and Wai Lam. 2017. Neural rating regression with abstractive tips generation for recommendation. In *SIGIR*.

Hui Liu, Qingyu Yin, and William Yang Wang. 2019. Towards explainable nlp: A generative explanation framework for text classification. In *ACL*.

William C. Mann and Sandra A. Thompson. 1988. Rhetorical structure theory: toward a functional theory of text.

Elman Mansimov, Alex Wang, and Kyunghyun Cho. 2019. A generalized framework of sequence generation with application to undirected sequence models. *ArXiv*, abs/1905.12790.

Jianmo Ni, Zachary C. Lipton, Sharad Vikram, and Julian J. McAuley. 2017. Estimating reactions and recommending products with generative models of reviews. In *IJCNLP*.

Jianmo Ni and Julian McAuley. 2018. Personalized review generation by expanding phrases and attending on aspect-aware representations. In *ACL*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Alex Wang and Kyunghyun Cho. 2019. Bert has a mouth, and it must speak: Bert as a markov random field language model. *CoRR*, abs/1902.04094.

Yizhong Wang, Sujian Li, and Jingfeng Yang. 2018. Toward fast and accurate neural discourse segmentation. In *EMNLP*.

Jason Weston, Emily Dinan, and Alexander H. Miller. 2018. Retrieve and refine: Improved sequence generation models for dialogue. In *SCAI@EMNLP*.

Xing Wu, Tao Zhang, Liangjun Zang, Jizhong Han, and Songlin Hu. 2019. Mask and infill: Applying masked language model to sentiment transfer. In *IJCAI*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Gregory S. Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Yu Wu, Furu Wei, Shaohan Huang, Yunli Wang, Zhoujun Li, and Ming Zhou. 2018. Response generation by context-aware prototype editing. In *AAAI*.

Lili Yao, Nanyun Peng, Ralph Weischedel, Kevin Knight, Dongyan Zhao, and Rui Yan. 2019. Plan-and-write: Towards better automatic storytelling. In *AAAI*.

Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. 2014. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *SIGIR*.

# Using Customer Service Dialogues for Satisfaction Analysis with Context-Assisted Multiple Instance Learning

**Kaisong Song**[1], **Lidong Bing**[1], **Wei Gao**[2], **Jun Lin**[1], **Lujun Zhao**[1]
**Jiancheng Wang**[3], **Changlong Sun**[1], **Xiaozhong Liu**[4], **Qiong Zhang**[1]
[1]Alibaba Group, Hangzhou, China
[2]Victoria University of Wellington, Wellington, New Zealand
[3]Soochow University, Suzhou, China
[4]Indiana University Bloomington, USA
{kaisong.sks,l.bing,linjun.lj,lujun.zlj,qz.zhang}@alibaba-inc.com,wei.gao@vuw.ac.nz
jiancheng.wang@qq.com, changlong.scl@taobao.com, liu237@indiana.edu

## Abstract

Customers ask questions and customer service staffs answer their questions, which is the basic service model via multi-turn customer service (CS) dialogues on E-commerce platforms. Existing studies fail to provide comprehensive service satisfaction analysis, namely satisfaction polarity classification (e.g., well satisfied, met and unsatisfied) and sentimental utterance identification (e.g., positive, neutral and negative). In this paper, we conduct a pilot study on the task of service satisfaction analysis (SSA) based on multi-turn CS dialogues. We propose an extensible Context-Assisted Multiple Instance Learning (CAMIL) model to predict the sentiments of all the customer utterances and then aggregate those sentiments into service satisfaction polarity. After that, we propose a novel Context Clue Matching Mechanism (CCMM) to enhance the representations of all customer utterances with their matched context clues, i.e., sentiment and reasoning clues. We construct two CS dialogue datasets from a top E-commerce platform. Extensive experimental results are presented and contrasted against a few previous models to demonstrate the efficacy of our model. [1]

## 1 Introduction

In the past decades, E-commerce platforms, such as Amazon.com and Taobao.com[2], have evolved into the most comprehensive and prosperous business ecosystems. They not only deeply involve other traditional businesses such as payment and logistics, but also largely transform every aspect of retailing. Taking the customer service on Taobao as an example, third-party retailers are always online to answer any question at any stage of pre-sale, sale and after-sale, through an instant messenger within the platform. The topics of relevant customer service dialogues involve various aspects of online shopping, such as product information, return or exchange, logistics, etc. Based on a previous survey, over 77% of buyers on Taobao communicated with sellers before placing an order (Gao and Zhang, 2011). Therefore, such service dialogue data contain very important clues for sellers to improve their service quality.

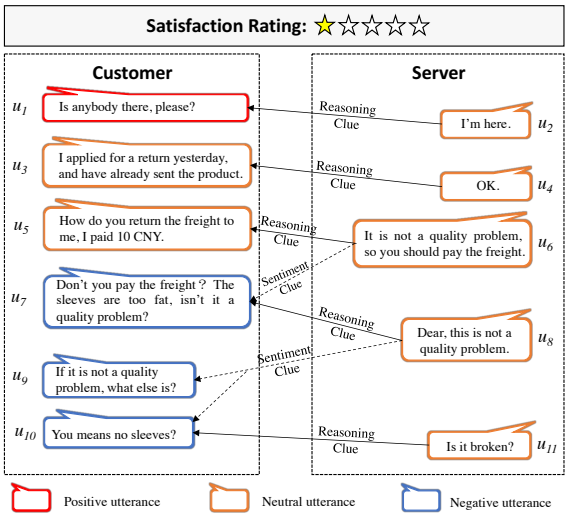


Figure 1: Customer utterance and context clues (i.e., sentiment and reasoning clues) alignments in multi-turn dialogue utterances of an *unsatisfied* customer service. The utterances ($u_i$) with *positive/neutral/negative* sentiments are denoted by *red*/*orange*/*blue* boxes.

Figure 1 depicts an exemplar dialogue of online customer service, which has a form of multi-turn dialogue between the customer and the customer service staff (or "the server" for short). In this dialogue, the customer is asking for refunding the freight he/she paid for sending back the product. At the end of service dialogue, the E-commerce platform invites the customer to score the service quality (e.g., using 1-5 stars denoting the extent of

[1]We have released the dataset at https://github.com/songkaisong/ssa.
[2]Taobao is a top E-commerce platform in China.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 198–207, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

satisfaction from "*very unsatisfied*" to "*very satisfied*") via instant messages or a grading interface. Evidently, the customer feels unsatisfied with the response. Automatically detecting such unsatisfactory service is important. For the retail shopkeepers, they can quickly locate such service dialogue and find out the reason to take remedial actions. For the platform, by detecting and analyzing such cases, the platform can define clear-cut rules, say "*not fitting well is not a quality issue, and the buyers should pay the freight for freight.*"

In this paper, we define a new task named **Service Satisfaction Analysis (SSA)**: *Given a service dialogue between the customer and the service staff, the task aims at predicting the customer's satisfaction, i.e., if the customer is satisfied by the responses from the service staff, meanwhile locating possible sentiment reasons, i.e., sentiment identification of the customer utterances*. For example, Figure 1 gives the satisfaction prediction of the service as "*unsatisfied*" and identifies the detailed sentiments of all customer utterances. Obviously, SSA focuses on two special cases of text classification over predefined satisfaction labels ("*well satisfied/met/unsatisfied*") and predefined sentiment labels ("*positive/neutral/negative*"). Text classification has been widely studied for decades, such as sentiment classification on product reviews (Song et al., 2017; Chen et al., 2017; Li et al., 2018, 2019), stance classification on tweets or blogs (Du et al., 2017; Liu, 2010), emotion classification for chit-chat (Majumder et al., 2018), etc. However, all these methods cannot deal with these two classification tasks simultaneously in a unified framework. Although recent studies on multi-task learning framework suggest that closely related tasks can improve each other mutually from separated supervision information (Ma et al., 2018; Cerisara et al., 2018; Wang et al., 2018b), the acquisition of sentence (or utterance)-level sentiment labels, which is required by multi-task learning, remains a laborious and expensive endeavor. In contrast, coarse-grained document (or dialogue)-level annotations are relatively easy to obtain due to the widespread use of opinion grading interfaces (e.g., ratings).

Recently, *Multiple Instance Learning* (MIL) framework is adopted for performing document-level and sentence-level sentiment classification simultaneously while only using document-level sentiment annotations (Zhou et al., 2009; Wang and Wan, 2018). However, these models are trained based on plain textual data which are in a much simpler form than our multi-turn dialogue structure. Specifically, customer service dialogue has unique characteristics. Customer utterances tend to have more sentiment changes during the customer service dialogue which affect customer's final satisfaction. Figure 1 illustrates that satisfaction polarity ("*unsatisfied*") is mostly embedded in the last few customer utterances (i.e., $u_7$, $u_9$ and $u_{10}$)[3]. On the other hand, a well-trained server varies less by always expressing positive/neutral utterances which contain helpful sentiment clues and reasoning clues. In this work, both sentiment clue and reasoning clue are called context clues which can directly or indirectly influence satisfaction polarity and need to be given special treatments in the model.

To deal with the issues, we propose a novel and extensible Context-Assisted Multiple Instance Learning (CAMIL) model for the new SSA task, and utterance-level sentiment classification and dialogue-level satisfaction classification will be done simultaneously only under the supervision of satisfaction labels. We motivate the idea of our context-assisted modeling solution based on the hypothesis that if a customer utterance does not have enough information to create a sound vector representation for sentiment prediction, we try to enhance it with a complementary representation derived from context clues via our position-guided Context Clue Matching Mechanism (CCMM). Overall, our contributions are three-fold:

- We introduce a new SSA task based on customer service dialogues. We thus propose a novel CAMIL model to predict the sentiment distributions of all customer utterances, and then aggregate those distributions to determine the final satisfaction polarity.
- We further propose an automatic CCMM to associate each customer utterance with its most relevant context clues, and then generate a complementary vector which enhances the customer utterance representation for better sentiment classification to boost the final satisfaction classification.

[3]Given a specific customer utterance ($u_7$) in Figure 1, the server utterance ($u_6$) triggers the change of customer sentiment from "*neutral*" to "*negative*", and the server utterance ($u_8$) answers the question "*this is not a quality problem*".

- Two real-world CS dialogue datasets are collected from a top E-commerce platform. The experimental results demonstrate that our model is effective for the SSA task.

## 2 Related Work

Service satisfaction analysis (SSA) is closely related to sentiment analysis (SA), because the sentiment of the customer utterances is a basic clue signaling the customer's satisfaction. Existing SA works aim to predict sentiment polarities (*positive*, *neutral* and *negative*) for subjective texts in different granularities, such as word (Feng et al., 2015; Song et al., 2016), sentence (Ma et al., 2017), short text (Song et al., 2015) and document (Yang et al., 2018a). In these studies, subjective texts are always considered as a sequence of words[4]. More recently, some researchers started to explore the utterance-level structure for sentiment classification, such as modeling dialogues via a hierarchical RNN in both word level and utterance level (Cerisara et al., 2018) or keeping track of sentiment states of dialogue participants (Majumder et al., 2018). However, none of these works can do dialogue-level satisfaction classification and utterance-level sentiment classification simultaneously. Recent studies (Cerisara et al., 2018; Ma et al., 2018; Wang et al., 2018b) employing multi-task learning open a possibility to address this issue. However, these models must be trained under the supervision of both document-level and sentence-level sentiment labels in which the later are generally not easy to obtain.

Sentiment classification based on Multiple Instance Learning (MIL) frameworks (Wang and Wan, 2018; Angelidis and Lapata, 2018) aims to perform document-level and sentence-level sentiment classification tasks simultaneously with the supervision of document labels only. Angelidis and Lapata (2018) proposed an MIL model for fine-grained sentiment analysis. Wang and Wan (2018) further applied the model to peer-reviewed research papers by integrating a memory built from abstracts. However, their models are not suitable for our SSA task because they ignore the dialogue structure of arbitrary interactions between customers and servers. In contrast, we consider complex multi-turn interactions within dialogues and explore context clue matching between customer utterances and server utterances for multi-tasking in the SSA task. Specifically, we improve the basic MIL models by proposing a position-guided automatic context clue matching mechanism (CCMM) to conduct customer utterance and context clues alignments for better sentiment classification to boost satisfaction classification. Other related work related to sentiment analysis for subjective texts in different granularities include (Yang et al., 2016; Wu and Huang, 2016; Yang et al., 2018b; Du et al., 2017; Wang et al., 2018a; Song et al., 2019).

[4]From plain sentiment analysis viewpoint, sentence and utterance are treated the same since a dialogue is considered as a chunk of plain texts and the matching between utterances is ignored. So, an utterance is seen as a sentence and a dialogue as a document. Here, we use sentence and utterance interchangeably when it comes to plain sentiment analysis models.

## 3 Context-Assisted MIL Network

In order to predict service satisfaction and identify sentiments of all customer utterances with available satisfaction labels, we propose a CAMIL model based on multiple instance learning approach. Figure 2 shows the architecture of our model which consists of three layers: *Input Representation Layer*, *Sentiment Classification Layer* and *Satisfaction Classification Layer*. In this section, we will describe the model in detail.

### 3.1 Input Representation Layer

Let each utterance $u_i = \left[w_1, ..., w_{|u_i|}\right]$ be a sequence of words. By adopting word embeddings and semantic composition models such as Recurrent Neural Network (RNN), we can learn the utterance representation. In this work, we adopt a standard LSTM model (Hochreiter and Schmidhuber, 1997) to learn a fixed-size utterance representation $\boldsymbol{v}_{u_i} \in \mathbb{R}^k$, where $k$ is the size of LSTM hidden state. Specifically, we first convert the words in each utterance $u_i$ to the corresponding word embeddings $\boldsymbol{E}_{u_i} \in \mathbb{R}^{d \times |u_i|}$ which are then fed into a LSTM for obtaining the last hidden state as the $\boldsymbol{v}_{u_i}$, where $d$ is the dimensionality of word embeddings. Formally, we have $\boldsymbol{v}_{u_i} = \mathrm{LSTM}(\boldsymbol{E}_{u_i})$.

We conjecture that the participants (i.e., customer and server) play different roles in CS dialogue. Our hypothesis is that satisfaction polarity can be more or less conveyed by the sentiments of key customer utterances, and meanwhile the sentiments of server utterances are generally polite or non-negative and contain text with context clues which complement the target customer's ut-

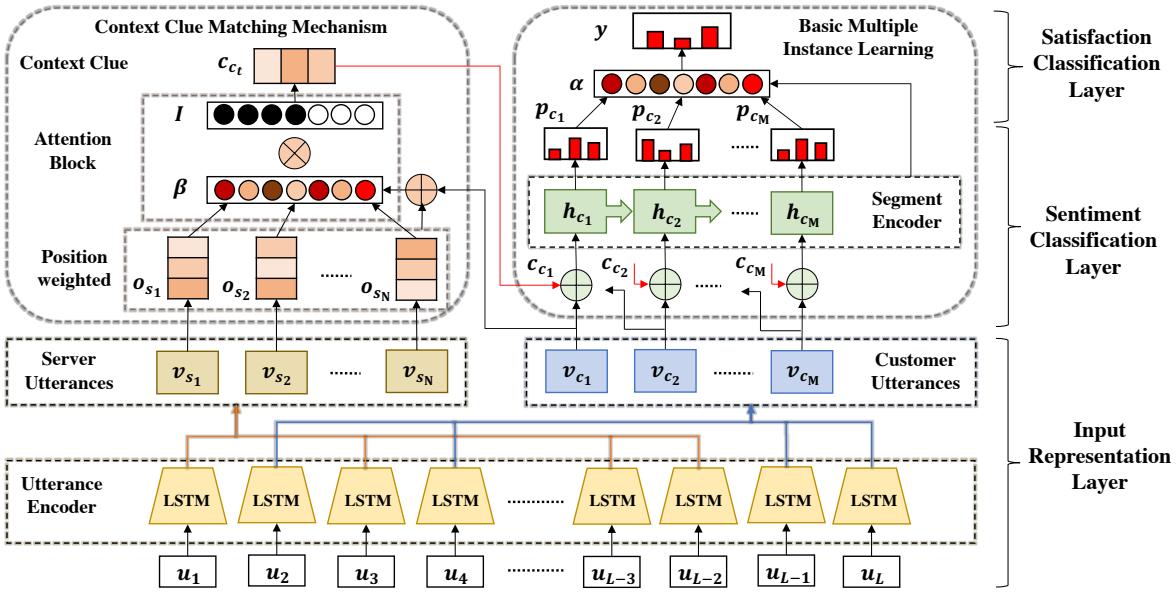


Figure 2: The architecture of our Context-Assisted Multiple Instance Learning (CAMIL) network. The model consists of four modules: Input Representation Layer, Sentiment Classification Layer, Satisfaction Classification Layer and Context Clue Matching Mechanism.

terances and indirectly affect satisfaction polarity. Thus, we separately denote the customer utterance representations as $\{\boldsymbol{v}_{c_1}, \boldsymbol{v}_{c_2}, ..., \boldsymbol{v}_{c_M}\}$ and server utterance representations as $\{\boldsymbol{v}_{s_1}, \boldsymbol{v}_{s_2}, ..., \boldsymbol{v}_{s_N}\}$, where $M + N = L$ and $L$ is the total number of utterances in the dialogue.

### 3.2 Sentiment Classification Layer

Customer utterances tend to have more direct impact on the dominating satisfaction polarity. However, short utterance texts may not contain enough information for semantic representation. Thus, considering context to enhance utterance representation is a natural and reasonable choice. Given a specific customer utterance vector $\boldsymbol{v}_{c_t}$, we use a context clue matching mechanism, namely CCMM (see Section 4), to produce matched context representation $\boldsymbol{c}_{c_t} \in \mathbb{R}^k$ as below:

$$\boldsymbol{c}_{c_t} = \text{CCMM}\left(\boldsymbol{v}_{c_t}, \{\boldsymbol{v}_{s_{t'}} | 1 \leq t' \leq N\}\right) \quad (1)$$

where $\boldsymbol{v}_{s_{t'}}$ is any server utterance representation.

After that, $\boldsymbol{v}_{c_t}$ can be enhanced by $\boldsymbol{c}_{c_t}$ via concatenation for a combined representation $\bar{\boldsymbol{v}}_{c_t} = \boldsymbol{v}_{c_t} \oplus \boldsymbol{c}_{c_t}$. Compared to $\boldsymbol{v}_{c_t}$, $\bar{\boldsymbol{v}}_{c_t} \in \mathbb{R}^{2k}$ contains more evidence for sentiment prediction. Then, we feed the representation sequence $\{\bar{\boldsymbol{v}}_{c_1}, \bar{\boldsymbol{v}}_{c_2}, ..., \bar{\boldsymbol{v}}_{c_M}\}$ into a standard LSTM for obtaining a segment representation $\boldsymbol{h}_{c_t} \in \mathbb{R}^k$ at each time step $t$, i.e., $\boldsymbol{h}_{c_t} = \text{LSTM}\left(\bar{\boldsymbol{v}}_{c_t}\right)$.

Finally, each segment representation $\boldsymbol{h}_{c_t}$ is fed into a linear layer and then a softmax function for predicting its sentiment distribution over sentiment labels $\mathcal{G} = \{\textit{positive, neutral, negative}\}$:

$$\boldsymbol{p}_{c_t} = \text{softmax}(\mathbf{W}^s \boldsymbol{h}_{c_t} + \boldsymbol{b}^s) \quad (2)$$

where $\mathbf{W}^s \in \mathbb{R}^{|\mathcal{G}| \times k}$ and $\boldsymbol{b}^s \in \mathbb{R}^{|\mathcal{G}|}$ are trainable parameters shared across all segments, and $\boldsymbol{p}_{c_t} \in \mathbb{R}^{|\mathcal{G}|}$ is the sentiment distribution for utterance $u_{c_t}$.

### 3.3 Satisfaction Classification Layer

In the simplest case, satisfaction polarities $\mathcal{C} = \{\textit{well satisfied, met, unsatisfied}\}$ can be computed by averaging all predicted sentiment distributions of customer utterances as $\boldsymbol{y} = \frac{1}{M} \sum_{t \in [1,M]} \boldsymbol{p}_{c_t}$. However, it is a crude way of combining sentiment distributions uniformly, as not all distributions convey equally important sentiment clues. In Figure 1, for example, the satisfaction polarity ("unsatisfied") is mostly determined by customer utterances $u_7$, $u_9$ and $u_{10}$ which are relatively more crucial than other ones. We opt for an attention mechanism to reward segments that are more likely to be good sentiment predictors. Therefore, we measure the importance of each segment representation $\boldsymbol{h}_{c_t}$ through a scoring function using feed forward neural network as below:

$$\alpha_{c_t} = \text{softmax}\left(\boldsymbol{v}^{\mathrm{T}} \tanh(\boldsymbol{W}^u \boldsymbol{h}_{c_t} + \boldsymbol{b}^u)\right) \quad (3)$$

where $\mathbf{W}^u \in \mathbb{R}^{k\times k}$, $\boldsymbol{b}^u \in \mathbb{R}^k$ and $\boldsymbol{v} \in \mathbb{R}^k$ are trainable parameters, $\boldsymbol{v}$ can be seen as a high-level representation of a fixed query "*what is the informative segment*" like that used in (Yang et al., 2016). Finally, we obtain the satisfaction distribution $\boldsymbol{y} \in \mathbb{R}^{|\mathcal{C}|}$ as the weighted sum of sentiment distributions of all the customer utterances by:

$$\boldsymbol{y} = \sum_{t\in[1,\mathrm{M}]} \alpha_{c_t} \boldsymbol{p}_{c_t} \tag{4}$$

### 3.4 Training and Parameter Learning

Note that in the training dataset, our approach only needs the dialogue's satisfaction labels while the utterance sentiment labels are unobserved. Therefore, we use the categorical cross-entropy loss to minimize the error between the distribution of the output satisfaction polarity and that of the gold-standard satisfaction label of the dialogue by:

$$\mathcal{L}(\Theta) = -\sum_{j\in[1,T]} \sum_{i\in\mathcal{C}} g_i^j \log(y_i^j) \tag{5}$$

where $g_i^j$ is 1 or 0 indicating whether the $i^{th}$ class is a correct answer for the $j^{th}$ training instance, $y_i^j$ is the predicted satisfaction probability distribution, and $\Theta$ denotes the trainable parameter set.

After learning $\Theta$, we feed each test instance into the final model, and the label with the highest probability stands for the predicted satisfaction polarity. We use back propagation to calculate the gradients of all the model parameters, and update them with Momentum optimizer (Qian, 1999).

## 4 Context Clue Matching Mechanism

Server utterances provide helpful context clues which can be defined as sentiment and reasoning clues by the positions of server utterances. Thus, we introduce the position-guided automatic context clue matching mechanism (CCMM) used to match each customer utterance with its most related server utterances, which contain two layers: the *position attention layer* and the *utterance attention layer*.

**Sentiment and Reasoning Clues:** Server utterances provide helpful context clues for each targeted customer utterance. Here, we aim to locate helpful context clues in server utterances which are categorized as sentiment clues and reasoning clues. Sentiment clues refer to the server utterances that appear preceding the targeted customer utterance and trigger its sentiment expression, such as server utterance $u_6$ leading to customer displeasure of the utterance $u_7$ in the Figure 1. Reasoning clues are the server utterances that appear following the targeted customer utterance and respond to its concerns, such as server utterance $u_6$ responding to the customer utterance $u_5$ in the Figure 1. Both types of clues are identified by the proposed attention layers along with position information.

**Position Attention Layer:** Typically, customer utterances are more likely to be triggered or answered by the server utterances near them. Let $p(\cdot)$ denote the position function of any utterance in the original dialogue, such as $p(u_{c_2}) = 3$ in Figure 1. For any customer utterance $u_{c_t}$, the preceding server utterances $\{u_{s_{t'}} | p(u_{s_{t'}}) < p(u_{c_t})\}$ may provide sentiment clues, and the following server utterances $\{u_{s_{t'}} | p(u_{s_{t'}}) > p(u_{c_t})\}$ may contain reasoning clues. By considering both directions, we compute the position attention weight $g(\cdot)$ by:

$$g(u_{c_t}, u_{s_{t'}}) = 1 - \frac{|p(u_{c_t}) - p(u_{s_{t'}})|}{L} \tag{6}$$

Then, the weighted output after this layer is formulated as below:

$$\boldsymbol{o}_{s_{t'}} = g(u_{c_t}, u_{s_{t'}}) * \boldsymbol{h}_{s_{t'}} * I(u_{c_t}, u_{s_{t'}}) \tag{7}$$

where $\boldsymbol{o}_{s_{t'}} \in \mathbb{R}^k$ is the weighted $\boldsymbol{h}_{s_{t'}}$ for $t' \in [1, N]$, the notation $I(\cdot)$ denotes a masking function which can be used to reserve either only sentiment clues (i.e., if $p(u_{c_t}) > p(u_{s_{t'}})$, $I(u_{c_t}, u_{s_{t'}})$ equals to 1, or 0 otherwise) or only reasoning clues (i.e., if $p(u_{c_t}) < p(u_{s_{t'}})$, $I(u_{c_t}, u_{s_{t'}})$ equals to 1, or 0 otherwise). Here, we suggest to consider both sentiment and reasoning clues, so $I(u_{c_t}, u_{s_{t'}})$ is a constant 1. Finally, we construct memory $\boldsymbol{O} \in \mathbb{R}^{k\times N}$ as below:

$$\boldsymbol{O} = [\boldsymbol{o}_{s_1}; \boldsymbol{o}_{s_2}; \ldots; \boldsymbol{o}_{s_\mathrm{N}}] \tag{8}$$

**Utterance Attention Layer:** Only a fraction of server utterances can match every customer utterance in sentiment or content, such as the exemplar dialogue in Figure 1. So, we introduce an attention strategy which enables our model to attend on server utterances of different importance when constructing a complementary context representation for any customer utterance. Considering customer utterance representation $\boldsymbol{h}_{c_t}$ as an index, we can produce a context vector $\boldsymbol{c}_{c_t} \in \mathbb{R}^k$ using a weighted sum of each piece $\boldsymbol{o}_{s_{t'}}$ of memory $\mathbf{O}$:

$$\boldsymbol{c}_{c_t} = \sum_{t'\in[1,N]} \beta_{s_{t'}} \boldsymbol{o}_{s_{t'}} \tag{9}$$

| Statistics items | Clothes | Makeup |
|---|---|---|
| # Dialogues | 10,000 | 3,540 |
| # US (unsatisfied) | 2,302 | 1,180 |
| # MT (met) | 6,399 | 1,180 |
| # WS (well satisfied) | 1,299 | 1,180 |
| Avg# Utterances | 25.99 | 26.67 |
| Avg# Words | 7.11 | 7.32 |
| # Segments | 123,242 | 46,255 |
| # NG (negative) | 12,619 | 6,130 |
| # NE (neutral) | 97,380 | 33,158 |
| # PO (positive) | 13,243 | 6,976 |

Table 1: Statistics of the datasets we collected.

where $\beta_{s_{t'}} \in [0, 1]$ is the attention weight calculated based on a scoring function using a feed forward neural network as follow:

$$\beta_{s_{t'}} = \text{softmax}\left(\bar{\mathbf{v}}^{\mathrm{T}} \tanh(\mathbf{W}^c \begin{bmatrix} \boldsymbol{o}_{s_{t'}} \\ \boldsymbol{h}_{c_t} \end{bmatrix} + \boldsymbol{b}^c)\right) \tag{10}$$

where $\mathbf{W}^c \in \mathbb{R}^{2k \times 2k}$, $\bar{\mathbf{v}} \in \mathbb{R}^{2k}$ and $\boldsymbol{b}^c \in \mathbb{R}^{2k}$ are trainable parameters.

## 5 Experiments and Results

### 5.1 Dataset and Experimental Settings

Our experiments are conducted based on two Chinese CS dialogue datasets, namely **Clothes** and **Makeup**, collected from a top E-commerce platform. Note that our proposed method is language independent and can be applied to other languages directly. **Clothes** is a corpus with 10K dialogues in the *Clothes* domain and **Makeup** is a balanced corpus with 3,540 dialogues in the *Makeup* domain. Both datasets have service satisfaction ratings in 1-5 stars from customer feedbacks. Meanwhile, we also annotate all the utterances in both datasets with sentiment labels for testing. In this study, we conduct two classification tasks: one is to predict in three satisfaction classes, i.e., "*unsatisfied*" (1-2 stars), "*met*" (3 stars) and "*satisfied*" (4-5 stars), and the other is to predict in three sentiment classes, i.e., "*negative/neutral/positive*". All texts are tokenized by a popular Chinese word segmentation utility called jieba[5]. After preprocessing, the datasets are partitioned for training, validation and test with a 80/10/10 split. A summary of statistics for both datasets are given in Table 1.

For all the methods, we apply fine-tuning for the word vectors, which can improve the performance. The word vectors are initialized by word embeddings that are trained on both datasets with CBOW (Mikolov et al., 2013), where the dimension is 300 and the vocabulary size is 23.3K. Other trainable model parameters are initialized by sampling values from a uniform distribution $\mathcal{U}(-0.01, 0.01)$. The size of LSTM hidden states $k$ is set as 128. The hyper-parameters are tuned on the validation set. Specifically, the initial learning rate is fixed as 0.1, the dropout rate is 0.2, the batch size is 32 and the number of epochs is 20. The performances of both satisfaction and sentiment classifications are evaluated using standard *Macro F1* and *Accuracy*.

### 5.2 Comparative Study

We compare our proposed approach with the following state-of-the-art Sentiment Analysis (SA) methods which can be grouped into two types: plain SA models and dialogue SA models.

**Plain SA models** consider dialogue as plain text and ignore utterance matching, say, utterance is seen as sentence and dialogue as document.

1) **LSTM**: We use word vectors as the input of a standard LSTM (Hochreiter and Schmidhuber, 1997) and feed the last hidden state into a softmax layer for satisfaction prediction.

2) **HAN**: A hierarchical attention network for document classification (Yang et al., 2016), which has two levels of attention mechanisms applied at word- and utterance-level, enabling it to attend differentially to more and less important content when constructing the dialogue representation and feeding it into a softmax layer for classification.

3) **HRN**: A hierarchical recurrent network for joint sentiment and act sequence recognition (Cerisara et al., 2018). It uses a bi-directional LSTM to represent utterances which are then fed into a standard LSTM for dialogue representation as the input of a softmax layer for classification.

4) **MILNET**: A multiple instance learning network for document-level and sentence-level sentiment analysis (Angelidis and Lapata, 2018). The original method is designed for plain textual data, which does not consider CS dialogue structure. In addition, their method ignores long-range dependencies among customer sentiments (i.e., without segment encoder in Figure 2).

**Dialogue SA models** consider utterance matching between customer and server utterances.

5) **HMN**: A hierarchical matching network for sentiment analysis, which uses a question-answer bidirectional matching layer to learn the matching

[5] https://pypi.org/project/jieba/

vector of each QA pair (i.e., customer utterance, server utterance) and then characterizes the importance of the generated matching vectors via a self-matching attention layer (Shen et al., 2018). However, the amount of pairs within a dialogue is huge, which leads to expensive calculations. Meanwhile, it considers the sentiments of server utterances, which will mislead final prediction.

6) **CAMIL**$_s$, **CAMIL**$_r$ and **CAMIL**$_{full}$: Our CAMIL models with only sentiment clues, only reasoning clues, and both of them, respectively, by setting masking function (see Equation 7).

All the methods are implemented by ourselves with TensorFlow[6] and run on a server configured with a Tesla V100 GPU, 2 CPU and 32G memory.

**Results and Analysis:** The results of comparisons are reported in Table 2. It indicates that LSTM cannot compete with other methods because it simply considers dialogues as word sequences but ignores the utterance matching. HAN and HRN perform much better by using a two-layer architecture (i.e., utterance and dialogue), but they ignore the utterance interactions. Besides, HRN treats the sentiment analysis task and the service satisfaction analysis task separately, and ignores their sentiment dependence. HMN uses a heuristic question-answering matching strategy, which is not enough flexible and easily causes mismatching issues. MILNET is the most related work, but its simplistic alignment model weakens prediction performance when facing on our complex customer service dialogue structure. MILNET however does not consider the dialogue structure and introduces unrelated sentiments from server utterances. CAMIL$_r$ and CAMIL$_s$ only consider either sentiment or reasoning clues, so they cannot compete with CAMIL$_{full}$ which considers both in dialogues. Partially configured model CAMI$L_r$ (or CAMI$L_s$) only considers sentiment (or reasoning) clues and performs worse than our full model CAMIL. This verifies that both types of clues are helpful and complementary, and they should be employed simultaneously.

On Clothes corpus, compared to the *met* class, the performances of all models on the *satisfied* class are much worse, because when the two classes cannot be well distinguished the models tend to predict the majority class (i.e., *met*) to minimize the loss. On Makeup corpus which is a balanced dataset, the performances on the *met* and *satisfied* classes are less distinctive, but both are consistently worse than the *unsatisfied* class.

[6] https://www.tensorflow.org/

| Methods | Clothes | | | | |
|---|---|---|---|---|---|
| | WS F1 | MT F1 | US F1 | MacroF1 | Acc. |
| LSTM | 0.264 | 0.772 | 0.634 | 0.557 | 0.684 |
| HAN | 0.515 | 0.817 | 0.704 | 0.679 | 0.755 |
| HRN | 0.508 | 0.835 | 0.676 | 0.673 | 0.766 |
| MILNET | 0.382 | 0.823 | 0.708 | 0.638 | 0.753 |
| HMN | 0.441 | 0.833 | 0.696 | 0.657 | 0.763 |
| CAMIL$_s$ | 0.450 | 0.822 | 0.713 | 0.665 | 0.767 |
| CAMIL$_r$ | 0.448 | 0.827 | 0.704 | 0.659 | 0.764 |
| CAMIL$_{full}$ | **0.554** | **0.844** | **0.715** | **0.704** | **0.783** |
| **Methods** | **Makeup** | | | | |
| | WS F1 | MT F1 | US F1 | MacroF1 | Acc. |
| LSTM | 0.585 | 0.597 | 0.724 | 0.635 | 0.632 |
| HAN | 0.684 | 0.713 | 0.848 | 0.748 | 0.748 |
| HRN | 0.720 | 0.686 | 0.858 | 0.755 | 0.754 |
| MILNET | 0.720 | 0.689 | 0.849 | 0.753 | 0.751 |
| HMN | 0.735 | 0.731 | 0.834 | 0.766 | 0.768 |
| CAMIL$_s$ | 0.693 | 0.710 | 0.840 | 0.747 | 0.748 |
| CAMIL$_r$ | 0.687 | 0.705 | 0.861 | 0.751 | 0.754 |
| CAMIL$_{full}$ | **0.738** | **0.745** | **0.874** | **0.786** | **0.785** |

Table 2: Results of different satisfaction classification methods. The best results are highlighted.

| Methods | Clothes | | | | |
|---|---|---|---|---|---|
| | WS F1 | MT F1 | US F1 | MacroF1 | Acc. |
| Server | 0.346 | 0.785 | 0.589 | 0.573 | 0.689 |
| Customer | 0.553 | 0.824 | 0.663 | 0.681 | 0.759 |
| NoPos | 0.554 | 0.838 | 0.673 | 0.688 | 0.771 |
| Average | 0.070 | 0.789 | 0.347 | 0.402 | 0.671 |
| Voting | 0.059 | 0.776 | 0.046 | 0.293 | 0.633 |
| CAMIL$_{full}$ | **0.554** | **0.844** | **0.715** | **0.704** | **0.783** |
| **Methods** | **Makeup** | | | | |
| | WS F1 | MT F1 | US F1 | MacroF1 | Acc. |
| Server | 0.597 | 0.630 | 0.745 | 0.657 | 0.655 |
| Customer | 0.735 | 0.687 | 0.790 | 0.737 | 0.734 |
| NoPos | 0.731 | 0.742 | 0.864 | 0.779 | 0.779 |
| Average | 0.578 | 0.708 | 0.834 | 0.706 | 0.714 |
| Voting | 0.231 | 0.016 | 0.553 | 0.267 | 0.387 |
| CAMIL$_{full}$ | **0.738** | **0.745** | **0.874** | **0.786** | **0.785** |

Table 3: Results of different model configurations.

### 5.3 Ablation Study

Different model configurations can largely affect the performance. We implement several model variants for ablation tests: **Server** and **Customer** consider only server and customer utterances in a dialogue, respectively. **NoPos** ignores the prior position information. **Average** takes the average of all the sentiment distributions for classification. **Voting** directly maps the majority sentiment into satisfaction prediction, i.e., negative $\rightarrow$ unsatisfied, neutral $\rightarrow$ met, positive $\rightarrow$ well-satisfied. The results of comparisons are reported in Table 3.

In Table 3, we can observe that **Customer** outperforms **Server** by a large margin, which indicates that service satisfaction is mostly related to

| Dataset | Class | positive | neutral | negative |
|---|---|---|---|---|
| **Clothes** | WS | 0.278 | 0.712 | 0.009 |
| | MT | 0.020 | 0.893 | 0.086 |
| | US | 0.042 | 0.513 | 0.443 |
| **Makeup** | WS | 0.273 | 0.718 | 0.009 |
| | MT | 0.012 | 0.893 | 0.094 |
| | US | 0.046 | 0.598 | 0.355 |

Table 4: Sentiment distribution in satisfaction classes.

| Methods | Clothes | | | | |
|---|---|---|---|---|---|
| | PO F1 | NE F1 | NG F1 | MacroF1 | Acc. |
| MILNET | 0.441 | 0.814 | 0.404 | 0.553 | 0.713 |
| CAMIL$_s$ | **0.545** | 0.867 | 0.506 | 0.639 | 0.787 |
| CAMIL$_r$ | 0.470 | 0.870 | 0.529 | 0.623 | 0.792 |
| CAMIL$_{full}$ | 0.484 | **0.893** | **0.555** | **0.644** | **0.824** |
| **Methods** | **Makeup** | | | | |
| | PO F1 | NE F1 | NG F1 | MacroF1 | Acc. |
| MILNET | 0.447 | 0.387 | 0.416 | 0.417 | 0.410 |
| CAMIL$_s$ | **0.566** | 0.672 | 0.501 | 0.580 | 0.609 |
| CAMIL$_r$ | 0.556 | 0.600 | 0.488 | 0.548 | 0.561 |
| CAMIL$_{full}$ | 0.544 | **0.725** | **0.516** | **0.595** | **0.647** |

Table 5: Results of sentiment classification by different models. The best results are highlighted.

$\mathbf{C}_1$: I have been waiting too long time! Negative [$NG$;$NE$]
$\mathbf{S}_1$ : Dear, I am the customer service agent. May I help you?
$\mathbf{C}_2$: I haven't received the goods yet! Why? Negative [$NG$;$NG$]
$\mathbf{S}_2$ : Dear, I'm reading your messages.
$\mathbf{C}_3$: As a rule, it should be received today. Neutral [$NE$;$NE$]
$\mathbf{S}_3$ : Dear, I feel very sorry about it.
$\mathbf{S}_4$ : I will help you push the workers.
$\mathbf{C}_4$: You deserve bad ratings! Negative [$NG$;$NG$]
$\mathbf{C}_5$: It is supposed to be shipped here in 3 days. Many days have passed! Negative [$NG$;$NE$]
$\mathbf{S}_5$ : Dear, I have already pushed them.
$\mathbf{C}_6$: Tell me when the goods will arrive? Negative [$NG$;$NE$]
$\mathbf{C}_7$: I will apply for refund if I don't get it tomorrow. Neutral [$NE$;$NE$]
$\mathbf{S}_6$ : I have urged them.
(*A few hours passed.*)
$\mathbf{C}_8$: I don't want it. I will apply for refund, immediately! Negative [$NG$;$NG$]
$\mathbf{S}_7$ : I'm sorry for the inconvenience.
$\mathbf{S}_8$ : Please wait for a few more days.
$\mathbf{S}_9$ : I have pushed them twice!
$\mathbf{C}_9$: However, I know this is useless. Negative [$NG$;$NG$]

**Satisfaction**: Unsatisfied [$US$;$MT$]

Figure 3: An example dialogue with predictions. $\mathbf{C}_i$ ($\mathbf{S}_j$) are customer (server) utterances. True labels are underlined. The predictions by our model and MILNET are colored in *red* and *blue* in the brackets, respectively.

the sentiments embedded in the customer utterances. However, its performance is still lower than CAMIL$_{full}$, suggesting that server utterances can provide helpful context clues. NoPos performs well but worse than CAMIL$_{full}$ since the position information provides prior knowledge for guiding context clue matching. Average and Voting are sub-optimal choices because not all the sentiment distributions contribute equally to the satisfaction polarity and the majority sentiment polarity also does not correlate strongly with it.

### 5.4 Results on Sentiment Classification

Table 4 shows another statistics of our datasets, i.e., the distribution of sentiment labels over each service satisfaction polarity, which reflects the imbalanced situation of utterance-level sentiments in real customer service dialogues.

In Table 5, we compare the sentiment prediction results of MILNET, CAMIL$_r$, CAMIL$_s$ and CAMIL$_{full}$. CAMIL$_r$ and CAMIL$_s$ perform worse than CAMIL$_{full}$ because they only consider partial context information. CAMIL$_{full}$ is the best mainly due to its accurate context clue matching. Thus, our proposed approach is more adaptive to the service satisfaction analysis task based on the customer service dialogues.

### 5.5 Case Study

Figure 1 illustrates our prediction results with an example dialogue which is translated from Chinese text. For brevity, we use **C/S** to denote customer/server utterance. Our model predicts the label "*unsatisfied*" correctly and also predicts reasonable sentiment polarities for customer utterances. Considering the context, customer utterances $\mathbf{C}_{1,5,6}$ are "*negative*" but predicted as "*neutral*" by MILNET because MILNET predicts sentiments only from target utterance itself and ignores context information. In addition, the sentiments of the customer utterances $\mathbf{C}_{4,5}$ and $\mathbf{C}_9$ tend to have larger influences on deciding the satisfaction polarity because $\mathbf{C}_4$ clearly conveys "*unsatisfied*" attitude, $\mathbf{C}_5$ complains about delay and $\mathbf{C}_9$ criticizes the low service quality.

We also visualize the attention weights in Figure 4 to explain our prediction results. For each customer utterance $\mathbf{C}_i$, we give the attention weights $\beta_{s_{t'}}$ on all the server utterances (see Formula 10). Furthermore, we also visualize the attention rates $\alpha_{c_t}$ on the customer utterances (see Formula 3). *Lighter colors denote smaller values*. From Figure 4, we can see that the customer utterances $\mathbf{C}_{4,5,9}$ have higher attention weights because customer attitudes are intuitively formed at the end of the dialogues (i.e., $\mathbf{C}_9$) or determined by explicit sentiments (i.e., $\mathbf{C}_4$). In this example, the customer is finally unhappy with the provided solution, and the sentiments did not change through the whole dialogue. We can also see that customer

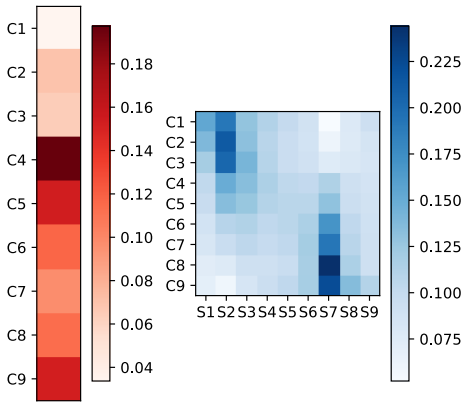


Figure 4: The visualization of attention rates $\alpha_{c_t}$ (Left in *red*) and $\beta_{s_{t'}}$ (Right in *blue*) for the given example.

utterances are influenced by server utterances. For example, $\mathbf{C}_{1-3}$ are related to $\mathbf{S}_2$, $\mathbf{C}_{4,5}$ are related to $\mathbf{S}_{2,3,7}$, and $\mathbf{C}_{6-9}$ are related to $\mathbf{S}_7$. This again validates the fact that customer utterances are related to the server utterances near them. Meanwhile, customer utterances may provide different types of context clues (i.e., sentiment and reasoning). For a specific server utterance $\mathbf{S}_7$, it provides explicit sentiment clue for $\mathbf{C}_9$ and also gives reasoning clue for $\mathbf{C}_8$.

**In-depth Analysis**: CAMIL$_{full}$ is only trained based on satisfaction labels, thus the laborious acquisition of sentiment labels is unnecessary. However, we would point out that lack of sentiment labels will inevitably lead to difficulties on identifying positive/negative utterances from those neutral ones. We will study to alleviate it in the future.

Our general observation is that the sentiment of customers at the beginning cannot largely determine the service satisfaction at the end. This is because the sentiment of the customers can vary with different quality of service during the dialogue, and the final service satisfaction results from the overall sentiments of important customer utterances in the dialogue (see the attention weights in Figure 4). To verify this, we design a heuristic baseline called Mapping which directly maps the initial *negative*, *neutral* and *positive* sentiment of customer to the corresponding service satisfaction, i.e., *unsatisfied*, *met* and *satisfied*. The satisfaction classification results are displayed in the Table 6.

In Table 6, we can observe that the Mapping method is far worse than our model. One reason is that the service dialogues in our datasets have more than 25 utterances in average (See the statistics in Table 1) and contain a large proportion

| Methods | Clothes | | Makeup | |
|---|---|---|---|---|
| | MacroF1 | Acc. | MacroF1 | Acc. |
| Mapping | 0.625 | 0.487 | 0.471 | 0.430 |
| CAMIL$_{full}$ | **0.704** | **0.783** | **0.786** | **0.785** |

Table 6: Satisfaction classification comparison between our method and a heuristic mapping method.

of complex interactions. Besides, the sentiment change is closely related to the quality of service and it is very common in our datasets. Thus, using such simple correlation does not work well in our complex dialogue scenarios.

## 6 Conclusions and Future Work

In this paper, we propose a novel CAMIL model for the SSA task. We first propose a basic MIL approach with the inputs of context-matched customer utterances, then predict the utterance-level sentiment polarities and dialogue-level satisfaction polarities simultaneously. In addition, we propose a context clue matching mechanism (CCMM) to match any customer utterance with the most related server utterances. Experimental results on two real-world datasets indicate our method clearly outperforms some state-of-the-art baseline models on the two SSA subtasks, i.e., service satisfaction polarity classification and utterance sentiment classification, which are performed simultaneously. We have made our datasets publicly available.

In the future, we will further improve our method by learning the correlation between the customer utterances and the server utterances. In addition, we will study other interesting tasks in customer service dialogues, such as outcome prediction or opinion change.

## 7 Acknowledgements

We thank all the reviewers for their insightful comments and helpful suggestions. This work is supported by National Key R&D Program of China (2018YFC0830200; 2018YFC0830206).

## References

Stefanos Angelidis and Mirella Lapata. 2018. Multiple instance learning networks for fine-grained sentiment analysis. *TACL*, 6:17–31.

Christophe Cerisara, Somayeh Jafaritazehjani, Adedayo Oluokun, and Hoa T. Le. 2018. Multi-task dialog act and sentiment recognition on mastodon. In *Proceedings of COLING*, pages 745–754.

Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. 2017. Recurrent attention network on memory for aspect sentiment analysis. In *Proceedings of EMNLP*, pages 452–461.

Jiachen Du, Ruifeng Xu, Yulan He, and Lin Gui. 2017. Stance classification with target-specific neural attention. In *Proceedings of IJCAI*, pages 3988–3994.

Shi Feng, Kaisong Song, Daling Wang, and Ge Yu. 2015. A word-emoticon mutual reinforcement ranking model for building sentiment lexicon from massive collection of microblogs. *World Wide Web*, 18(4):949–967.

Jie Gao and Zhenghua Zhang. 2011. User satisfaction of ali wangwang, an instant messenger tool. In *Theory, Methods, Tools and Practice - DUXU 2011, Part of HCI*, pages 414–420.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Xin Li, Lidong Bing, Wai Lam, and Bei Shi. 2018. Transformation networks for target-oriented sentiment classification. In *Proceedings of ACL*, pages 946–956.

Xin Li, Lidong Bing, Piji Li, and Wai Lam. 2019. A unified model for opinion target extraction and target sentiment prediction. In *Proceedings of AAAI*, pages 6714–6721.

Bing Liu. 2010. Sentiment analysis and subjectivity. In *Handbook of Natural Language Processing, Second Edition.*, pages 627–666.

Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2017. Interactive attention networks for aspect-level sentiment classification. In *Proceedings of IJCAI*, pages 4068–4074.

Jing Ma, Wei Gao, and Kam-Fai Wong. 2018. Detect rumor and stance jointly by neural multi-task learning. In *Proceedings of WWW*, pages 585–593.

Navonil Majumder, Soujanya Poria, Devamanyu Hazarika, Rada Mihalcea, Alexander F. Gelbukh, and Erik Cambria. 2018. Dialoguernn: An attentive RNN for emotion detection in conversations. *CoRR*, abs/1811.00405.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of ICLR Workshop*.

Ning Qian. 1999. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145–151.

Chenlin Shen, Changlong Sun, Jingjing Wang, Yangyang Kang, Shoushan Li, Xiaozhong Liu, Luo Si, Min Zhang, and Guodong Zhou. 2018. Sentiment classification towards question-answering with hierarchical matching network. In *Proceedings of EMNLP*, pages 3654–3663.

Kaisong Song, Shi Feng, Wei Gao, Daling Wang, Ge Yu, and Kam-Fai Wong. 2015. Personalized sentiment classification based on latent individuality of microblog users. In *Proceedings of IJCAI*, pages 2277–2283.

Kaisong Song, Wei Gao, Ling Chen, Shi Feng, Daling Wang, and Chengqi Zhang. 2016. Build emotion lexicon from the mood of crowd via topic-assisted joint non-negative matrix factorization. In *Proceedings of SIGIR*, pages 773–776.

Kaisong Song, Wei Gao, Shi Feng, Daling Wang, Kam-Fai Wong, and Chengqi Zhang. 2017. Recommendation vs sentiment analysis: A text-driven latent factor model for rating prediction with cold-start awareness. In *Proceedings of IJCAI*, pages 2744–2750.

Kaisong Song, Wei Gao, Lujun Zhao, Jun Lin, Changlong Sun, and Xiaozhong Liu. 2019. Cold-start aware deep memory network for multi-entity aspect-based sentiment analysis. In *Proceedings of IJCAI*, pages 5197–5203.

Jingjing Wang, Jie Li, Shoushan Li, Yangyang Kang, Min Zhang, Luo Si, and Guodong Zhou. 2018a. Aspect sentiment classification with both word-level and clause-level attention networks. In *Proceedings of IJCAI*, pages 4439–4445.

Ke Wang and Xiaojun Wan. 2018. Sentiment analysis of peer review texts for scholarly papers. In *Proceedings of SIGIR*, pages 175–184.

Weichao Wang, Shi Feng, Wei Gao, Daling Wang, and Yifei Zhang. 2018b. Personalized microblog sentiment classification via adversarial cross-lingual multi-task learning. In *Proceedings of EMNLP*, pages 338–348.

Fangzhao Wu and Yongfeng Huang. 2016. Personalized microblog sentiment classification via multi-task learning. In *Proceedings of AAAI*, pages 3059–3065.

Jun Yang, Runqi Yang, Chongjun Wang, and Junyuan Xie. 2018a. Multi-entity aspect-based sentiment analysis with context, entity and aspect memory. In *Proceedings of AAAI*, pages 6029–6036.

Jun Yang, Runqi Yang, Chongjun Wang, and Junyuan Xie. 2018b. Multi-entity aspect-based sentiment analysis with context, entity and aspect memory. In *Proceedings of AAAI*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of NAACL*, pages 1480–1489.

Zhi-Hua Zhou, Yu-Yin Sun, and Yu-Feng Li. 2009. Multi-instance learning by treating instances as non-i.i.d. samples. In *Proceedings of ICML*, pages 1249–1256.

# Leveraging Dependency Forest for Neural Medical Relation Extraction

**Linfeng Song**[1,2], **Yue Zhang**[3,4]*, **Daniel Gildea**[1], **Mo Yu**[5], **Zhiguo Wang**[6] and **Jinsong Su**[7]
[1]University of Rochester, Rochester, NY, USA
[2]Tencent AI Lab, Bellevue, WA, USA
[3]Institute of Advanced Technology, Westlake Institute for Advanced Study
[4]School of Engineering, Westlake Universtiy
[5]IBM T.J. Watson Research, Yorktown Heights, NY, USA
[6]Amazon AWS, New York, NY, USA
[7]Xiamen University, Xiamen, China

## Abstract

Medical relation extraction discovers relations between entity mentions in text, such as research articles. For this task, dependency syntax has been recognized as a crucial source of features. Yet in the medical domain, 1-best parse trees suffer from relatively low accuracies, diminishing their usefulness. We investigate a method to alleviate this problem by utilizing dependency forests. Forests contain many possible decisions and therefore have higher recall but more noise compared with 1-best outputs. A graph neural network is used to represent the forests, automatically distinguishing the useful syntactic information from parsing noise. Results on two biomedical benchmarks show that our method outperforms the standard tree-based methods, giving the state-of-the-art results in the literature.

## 1 Introduction

The sheer amount of medical articles and their rapid growth prevent researchers from receiving comprehensive literature knowledge by direct reading. This can hamper both medical research and clinical diagnosis. NLP techniques have been used for automating the knowledge extraction process from the medical literature (Friedman et al., 2001; Yu and Agichtein, 2003; Hirschman et al., 2005; Xu et al., 2010; Sondhi et al., 2010; Abacha and Zweigenbaum, 2011). Along this line of work, a long-standing task is relation extraction, which mines factual knowledge from free text by labeling relations between entity mentions. As shown in Figure 1, the sub-clause "previously observed cytochrome P450 3A4 ( CYP3A4 ) interaction of the dual orexin receptor antagonist almorexant" contains two entities, namely "orexin receptor" and "almorexant". There is an "adversary" relation between these two entities, denoted as"CPR:6".

*Yue Zhang is the corresponding author

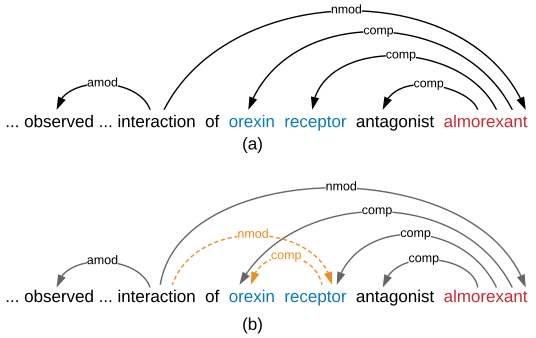


Figure 1: (a) 1-best dependency tree and (b) dependency forest for a medical-domain sentence, where edge label "comp" represents "compound". Associated mentions are in different colors. Some irrelevant words and edges are omitted for simplicity.

Previous work has shown that dependency syntax is important for guiding relation extraction (Culotta and Sorensen, 2004; Bunescu and Mooney, 2005; Liu et al., 2015; Gormley et al., 2015; Xu et al., 2015a,b; Miwa and Bansal, 2016; Zhang et al., 2018b), especially in biological and medical domains (Quirk and Poon, 2017; Peng et al., 2017; Song et al., 2018b). Compared with sequential surface-level structures, such as POS tags, dependency trees help to model word-to-word relations more easily by drawing direct connections between distant words that are syntactically correlated. Take the phrase "effect on the medicine" for example; "effect" and "medicine" are directly connected in a dependency tree, regardless of how many modifiers are added in between.

Dependency parsing has achieved an accuracy over 96% in the news domain (Liu and Zhang, 2017; Kitaev and Klein, 2018). However, for the medical literature domain, parsing accuracies can drop significantly (Lease and Charniak, 2005; McClosky and Charniak, 2008; Sagae et al., 2008; Candito et al., 2011). This can lead to severe er-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 208–218, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

ror propagation in downstream relation extraction tasks, offsetting much of the benefit that relation extraction models can obtain by exploiting dependency trees as a source of external features.

We address the low-accuracy issue in biomedical dependency parsing by considering dependency forests as external features. Instead of 1-best trees, dependency forests consist of dependency arcs and labels that a parser is relatively confident about, therefore having better recall of gold-standard arcs by offering more candidate choices with noise. Our main idea is to let a relation extraction system learn automatically from a forest which arcs are the most relevant through end-task training, rather than relying solely on the decisions of a noisy syntactic parser. To this end, a graph neural network is used for encoding a forest, which in turn provides features for relation extraction. Back-propagation passes loss gradients from the relation extraction layer to the graph encoder, so that the more relevant edges can be chosen automatically for better relation extraction.

Results on BioCreative VI ChemProt (CPR) (Krallinger et al., 2017) and a recent dataset focused on phenotype-gene relations (PGR) (Sousa et al., 2019) show that our method outperforms a strong baseline that uses 1-best dependency trees as features, giving the state-of-the-art accuracies in the literature. To our knowledge, we are the first to study dependency forests for medical information extraction, showing their advantages over 1-best tree structures. Our code is available at `http://github.com/freesunshine/dep-forest-re`.

## 2 Related work

**Syntactic forests** There have been previous studies leveraging constituent forests for machine translation (Mi et al., 2008; Ma et al., 2018; Zaremoodi and Haffari, 2018), sentiment analysis (Le and Zuidema, 2015) and text generation (Lu and Ng, 2011). However, the usefulness of dependency forests is relatively rarely studied, with one exception being Tu et al. (2010), who use dependency forests to enhance long-range word-to-word dependencies for statistical machine translation. To our knowledge, we are the first to study the usefulness of dependency forests for relation extraction under a strong neural framework.

**Graph neural network** Graph neural networks (GNNs) have been successful in encoding dependency trees for downstream tasks, such as semantic role labeling (Marcheggiani and Titov, 2017), semantic parsing (Xu et al., 2018), machine translation (Song et al., 2019; Bastings et al., 2017), relation extraction (Song et al., 2018b) and sentence ordering (Yin et al., 2019). In particular, Song et al. (2018b) showed that GNNs are more effective than DAG networks (Peng et al., 2017) for modeling syntactic trees in relation extraction, which cause loss of important structural information. We are the first to exploit GNNs for encoding search spaces in the form of dependency forests.

## 3 Task

Formally, the input to our task is a sentence $\boldsymbol{s} = w_1, w_2, \ldots, w_N$, where $N$ is the number of words in the sentence and $w_i$ represents the $i$-th input word. $\boldsymbol{s}$ is annotated with boundary information ($\xi_1 : \xi_2$ and $\zeta_1 : \zeta_2$) of target entity mentions ($\xi$ and $\zeta$). We focus on the classic binary relation extraction setting (Quirk and Poon, 2017), where the number of associated mentions is two. The output is a relation from a predefined relation set $\boldsymbol{R} = (r_1, \ldots, r_M, \text{None})$, where "None" means that no relation holds for the entities.

Two steps are taken for predicting the correct relation given an input sentence. First, a dependency parser is used to label the syntactic structure of the input. Here our baseline system takes the standard approach, using the 1-best parser output tree $\boldsymbol{D}_T$ as features. In contrast, our proposed model uses the most confident parser forest $\boldsymbol{D}_F$ as features. Given $\boldsymbol{D}_T$ or $\boldsymbol{D}_F$, the second step is to encode both $s$ and $\boldsymbol{D}_T/\boldsymbol{D}_F$ using a neural network, before making a prediction.

We make use of the same graph neural network encoder structure to represent dependency syntax information for both the baseline and our model. In particular, a graph recurrent neural network architecture (Beck et al., 2018; Song et al., 2018a; Zhang et al., 2018a) is used, which has been shown effective in encoding graph structures (Song et al., 2019), giving competitive results with alternative graph networks such as graph convolutional neural networks (Marcheggiani and Titov, 2017; Bastings et al., 2017).

## 4 Baseline: DEPTREE

As shown in Figure 2, our baseline model stacks a bidirectional LSTM layer to encode an input sentence $w_1, \ldots, w_N$ with a graph recurrent network

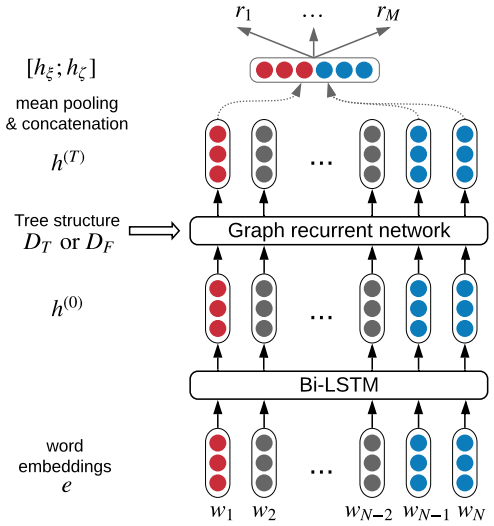


Figure 2: Framework of our baseline and model.

(GRN) to encode a 1-best dependency tree, which extracts features from the sentence and the dependency tree $\boldsymbol{D}_T$, respectively. Similar model frameworks have shown highly competitive performances in previous relation extraction studies (Peng et al., 2017; Song et al., 2018b).

### 4.1 Bi-LSTM layer

Given the input sentence $w_1, w_2, \ldots, w_N$, we represent each word with its embedding to generate a sequence of embeddings $\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_N$. A Bi-LSTM layer is used to encode the sentences:

$$\overleftarrow{\boldsymbol{h}}_i^{(0)} = \text{LSTM}_l(\overleftarrow{\boldsymbol{h}}_{i+1}^{(0)}, \boldsymbol{e}_i) \quad (1)$$

$$\overrightarrow{\boldsymbol{h}}_i^{(0)} = \text{LSTM}_r(\overrightarrow{\boldsymbol{h}}_{i-1}^{(0)}, \boldsymbol{e}_i), \quad (2)$$

where the state of each word $w_i$ is generated by concatenating the states of both directions:

$$\boldsymbol{h}_i^{(0)} = [\overleftarrow{\boldsymbol{h}}_i^{(0)}; \overrightarrow{\boldsymbol{h}}_i^{(0)}] \quad (3)$$

### 4.2 GRN layer

A 1-best dependency tree can be represented as a directed graph $\boldsymbol{D}_T = \langle \boldsymbol{V}, \boldsymbol{E} \rangle$, where $\boldsymbol{V}$ includes all words $w_1, w_2, \ldots, w_N$ and $\boldsymbol{E} = \{(w_j, l, w_i)\}|_{w_j \in V, w_i \in V}$ represents all dependency edges (Marcheggiani and Titov, 2017). Each triple $(w_j, l, w_i)$ corresponds to a dependency edge, where $w_j$ modifies $w_i$ with an arc label $l$. Each word $w_i$ is associated with a hidden state that is initialized with the Bi-LSTM output $\boldsymbol{h}_i^{(0)}$. The state representation of the entire tree consists of all word states:

$$\boldsymbol{h}^{(0)} = \{\boldsymbol{h}_i^{(0)}\}_{w_i \in V} \quad (4)$$

In order to capture non-local interactions between words, the GRN layer adopts a message passing framework that performs iterative information exchange between directly connected words. As a result, each word state is updated by absorbing larger contextual information through the message passing process, and a sequence of state transitions $\boldsymbol{h}^{(0)}, \boldsymbol{h}^{(1)}, \ldots$ is generated for the entire tree. The final state $\boldsymbol{h}^{(T)} = \text{GRN}(\boldsymbol{h}^{(0)}, T)$, where $T$ is a hyperparameter representing the number of state transitions.

**Message passing** The message passing framework takes two main steps within each iteration: message calculation and state update. Take $w_i$ and iteration $t$ as the example. In the first step, separate messages $\boldsymbol{m}_i^{\uparrow}$ and $\boldsymbol{m}_i^{\downarrow}$ are calculated by summing up the messages of its children and parent in the dependency tree, respectively:

$$\boldsymbol{m}_i^{\uparrow} = \sum_{(w_j, l, w_i) \in \boldsymbol{E}_{(\cdot,\cdot,i)}} [\boldsymbol{h}_j^{(t-1)}; \boldsymbol{e}_l] \quad (5)$$

$$\boldsymbol{m}_i^{\downarrow} = \sum_{(w_i, l, w_k) \in \boldsymbol{E}_{(i,\cdot,\cdot)}} [\boldsymbol{h}_k^{(t-1)}; \boldsymbol{e}_{l_{rev}}], \quad (6)$$

where $\boldsymbol{E}_{(\cdot,\cdot,i)}$ and $\boldsymbol{E}_{(i,\cdot,\cdot)}$ represent all edges with a head word $w_i$ and a modifier word $w_i$, respectively, and $\boldsymbol{e}_{l_{rev}}$ represents the embedding of label $l_{rev}$, the reverse version of original label $l$ (such as "amod-rev" is the reverse version of "amod"). The message from a child or a parent is obtained by simply concatenating its hidden state with the corresponding edge label embedding.

In the second step, GRN uses standard gated operations of LSTM (Hochreiter and Schmidhuber, 1997) to update hidden state $\boldsymbol{h}_i^{(t-1)}$ with the previously integrated message. In particular, a cell $\boldsymbol{c}_i^{(t)}$ is taken to record memory for $\boldsymbol{h}_i^{(t)}$; an input gate $\boldsymbol{i}_i^{(t)}$, an output gate $\boldsymbol{o}_i^{(t)}$ and a forget gate $\boldsymbol{f}_i^{(t)}$ are used to control information flow from the inputs and to the output $\boldsymbol{h}_i^{(t)}$:

$$\begin{aligned}
\boldsymbol{i}_i^{(t)} &= \sigma(\boldsymbol{W}_1^{\uparrow}\boldsymbol{m}_i^{\uparrow} + \boldsymbol{W}_1^{\downarrow}\boldsymbol{m}_i^{\downarrow} + \boldsymbol{b}_1) \\
\boldsymbol{o}_i^{(t)} &= \sigma(\boldsymbol{W}_2^{\uparrow}\boldsymbol{m}_i^{\uparrow} + \boldsymbol{W}_2^{\downarrow}\boldsymbol{m}_i^{\downarrow} + \boldsymbol{b}_2) \\
\boldsymbol{f}_i^{(t)} &= \sigma(\boldsymbol{W}_3^{\uparrow}\boldsymbol{m}_i^{\uparrow} + \boldsymbol{W}_3^{\downarrow}\boldsymbol{m}_i^{\downarrow} + \boldsymbol{b}_3) \\
\boldsymbol{u}_i^{(t)} &= \tanh(\boldsymbol{W}_4^{\uparrow}\boldsymbol{m}_i^{\uparrow} + \boldsymbol{W}_4^{\downarrow}\boldsymbol{m}_i^{\downarrow} + \boldsymbol{b}_4) \\
\boldsymbol{c}_i^{(t)} &= \boldsymbol{f}_i^{(t)} \odot \boldsymbol{c}_i^{(t-1)} + \boldsymbol{i}_i^{(t)} \odot \boldsymbol{u}_i^{(t)} \\
\boldsymbol{h}_i^{(t)} &= \boldsymbol{o}_i^{(t)} \odot \tanh(\boldsymbol{c}_i^{(t)}),
\end{aligned} \quad (7)$$

where $\boldsymbol{W}_x^{\uparrow}$, $\boldsymbol{W}_x^{\downarrow}$, and $\boldsymbol{b}_x$ ($x \in \{1, 2, 3, 4\}$) are

model parameters, and $\boldsymbol{c}_i^{(0)}$ is initialized as a vector of zeros.

The same process repeats for $T$ iterations. Starting from $\boldsymbol{h}^{(0)}$ of the Bi-LSTM layer, increasingly more informed hidden states $\boldsymbol{h}^{(t)}$ are obtained as the iteration increases, and $\boldsymbol{h}^{(T)}$ is used as the final representation of each word.

### 4.3 Relation prediction

Given $\boldsymbol{h}^{(T)}$ of the GRN encoding, we calculate the representation vector of the two related entity mentions $\xi$ and $\zeta$ (such as "almorexant" and "orexin receptor" in Figure 1) with mean pooling:

$$\boldsymbol{h}_\xi = f_{\text{mean}}(\boldsymbol{h}^{(T)}_{\xi_1:\xi_2}) \quad (8)$$

$$\boldsymbol{h}_\zeta = f_{\text{mean}}(\boldsymbol{h}^{(T)}_{\zeta_1:\zeta_2}) \quad (9)$$

where $\xi_1 : \xi_2$ and $\zeta_1 : \zeta_2$ represent the span of $\xi$ and $\zeta$, respectively, and $f_{\text{mean}}$ is the mean-pooling function.

Finally, the representations of both mentions are concatenated to be the input of a logistic regression classifier:

$$\boldsymbol{y} = \text{softmax}(\boldsymbol{W}_5[\boldsymbol{h}_\xi; \boldsymbol{h}_\zeta] + \boldsymbol{b}_5), \quad (10)$$

where $\boldsymbol{W}_5$ and $\boldsymbol{b}_5$ are model parameters.

## 5 Model

In this section, we first discuss how to generate high-quality dependency forests, before showing how to adapt GRN to consider the parser probability of each dependency edge.

### 5.1 Forest generation

Given a dependency parser, generating dependency forests with high recall and low noise is a non-trivial problem. On the one hand, keeping the whole search space gives 100% recall, but introduces maximum noise. On the other hand, using the 1-best dependency tree can result in low recall given an imperfect parser. We investigate two algorithms to generate high-quality forests by judging "quality" from different perspectives: one focusing on arcs, and the other focusing on trees.

**EDGEWISE** This algorithm focuses on the local relation of each individual edge and uses parser probabilities as confidence scores to assess edge qualities. Starting from the whole parser search space, it keeps all the edges with scores greater than a threshold $\gamma$. The time complexity is $O(N^2)$, where $N$ represents the sentence length.[1]

**KBESTEISNER** This algorithm extends the Eisner algorithm (Eisner, 1996) with cube pruning (Huang and Chiang, 2005) for finding $K$ highest-scored tree structures. The Eisner algorithm is a standard method for decoding 1-best trees for graph-based dependency parsing. Based on bottom-up dynamic programming, it stores the 1-best subtree for each span and takes $O(N^3)$ time complexity for decoding a sentence of $N$ words.

KBESTEISNER keeps a sorted list of $K$-best hypotheses for each span. Cube pruning (Huang and Chiang, 2005) is adopted to generate the $K$-best list for each larger span from the $K$-best lists of its sub-spans. After the bottom-up decoding, we merge the final $K$-bests by combining identical dependency edges to make the forest. As a result, KBESTEISNER takes $O(N^3K \log K)$ time.

**Discussions** EDGEWISE is much simpler and faster than KBESTEISNER. Compared with the $O(N^3K \log K)$ time complexity of KBESTEISNER, EDGEWISE only takes $O(N^2)$ running time, and each step (storing an edge) runs faster than KBESTEISNER (making a new hypothesis by combining two from sub-spans). Besides, the forests of EDGEWISE can be denser and provide richer information than those from KBESTEISNER. This is because KBESTEISNER only merges $K$ trees, where many edges are shared among them. Also, $K$ cannot be set to a large number (such as 100), because that will cause a dramatic increase of running time.

Compared with KBESTEISNER, EDGEWISE suffers from two potential problems. First, EDGEWISE does not guarantee to produce a 1-best tree in a generated forest, as it makes decisions by considering the individual edges. Second, it does not guarantee to generate spanning forests, which can happen when the threshold $\gamma$ is high. On the other hand, no previous work has shown that the information from the whole tree is crucial for relation extraction. In fact, many previous studies use only the dependency path between the target entity mentions (Bunescu and Mooney, 2005; Airola et al., 2008; Chowdhury et al., 2011; Gormley et al., 2015; Mehryary et al., 2016). We study

[1] More accurately, it is $O(N^2L)$ and $L$ s a *constant* factor, denoting the number of distinct dependency labels. We omit it for simplicity.

the effectiveness of both algorithms in our experiments.

### 5.2 GRN encoding with parser confidence

As illustrated by Figure 1(b), our dependency forests are directed graphs that can be consumed by GRN without any structural changes. For fair comparison, we use the same model as the baseline to encode sentences and forests. Thus our model uses **the same number of parameters** as our baseline taking 1-best trees.

Since forests contain more than one tree, it is intuitive to consider parser confidence scores for potentially better feature extraction. To this end, we slightly adjust the GRN encoding process without introducing additional parameters. In particular, we enhance the original message sum function (Equations 5 and 6) by applying the edge probabilities in calculating weighted message sums:

$$\boldsymbol{m}_i^{\uparrow} = \sum_{\epsilon \in \boldsymbol{E}_{(\cdot,\cdot,i)}} p_\epsilon [\boldsymbol{h}_j^{(t-1)}; \boldsymbol{e}_l] \quad (11)$$

$$\boldsymbol{m}_i^{\downarrow} = \sum_{\epsilon \in \boldsymbol{E}_{(i,\cdot,\cdot)}} p_\epsilon [\boldsymbol{h}_k^{(t-1)}; \boldsymbol{e}_{l_{rev}}], \quad (12)$$

where $\epsilon$ (instead of a triple) is used to represent an edge for simplicity, and $p_\epsilon$ is the parser probability for edge $\epsilon$. The edge probabilities are *not* adjusted during end-task training.

## 6 Training

**Relation loss** Given a set of training instances, each containing a sentence $\boldsymbol{s}$ with two target mentions $\xi$ and $\zeta$, and a dependency structure $\boldsymbol{D}$ (tree or forest), we train our models with a cross-entropy loss between the gold-standard relations $r$ and model distribution:

$$l_R = -\log p(r|\boldsymbol{s}, \xi, \zeta, \boldsymbol{D}; \boldsymbol{\theta}), \quad (13)$$

where $\boldsymbol{\theta}$ represents the model parameters.

**Using additional NER loss** For training on BioCreative VI CPR, we follow previous work (Liu et al., 2017; Verga et al., 2018) to take NER loss as additional supervision, though the mention boundaries are known during testing.

$$l_{NER} = -\frac{1}{N} \sum_{n=1}^{N} \log p(t_n|\boldsymbol{s}, \boldsymbol{D}; \boldsymbol{\theta}), \quad (14)$$

where $t_n$ is the gold NE tag of $w_n$ with the "BIO" scheme. Both losses are conditionally independent given the deep features produced by our model, and the final loss for BioCreative VI CPR training is $l = l_R + l_{NER}$.

## 7 Experiments

We conduct experiments on two medical benchmarks to test the usefulness of dependency forest.

### 7.1 Data

**BioCreative VI CPR (Krallinger et al., 2017)** This task[2] focuses on the relations between chemical compounds (such as drugs) and proteins (such as genes). The full corpus contains 1020, 612 and 800 extracted PubMed[3] abstracts for training, development and testing, respectively. All abstracts are manually annotated with the boundaries of entity mentions and the relations. The data provides three types of NEs: "CHEMICAL", "GENE-Y" and "GENE-N", and the relation set $\boldsymbol{R}$ contains 5 regular relations ("CPR:3", "CPR:4", "CPR:5", "CPR:6" and "CPR:9") and the "None" relation.

For efficient generation of dependency structures, we segment each abstract into sentences, keeping only the sentences that contain at least a chemical mention and a protein mention. For any sentence containing several chemical mentions or protein mentions, we keep multiple copies of it with each copy having different target mention pairs. As a result, we only consider the relations of mentions in the same sentence, assigning all cross-sentence chemical-protein pairs as "None" relation. By doing this, we effectively sacrifice cross-sentence relations, which has a negative effect on our systems; but this is necessary for efficient generation of dependency structures since directly parsing a short paragraph is slow and erroneous.[4] In general, we obtain 16,107 training, 10,030 development and 14,269 testing instances, in which around 23% have regular relations. The highest recalls for relations on our development and test sets are 92.25 and 92.54, respectively, because of the exclusion of cross-sentence relations in preprocessing. We report F1 scores of the full test set for a fair comparison, using all gold regular relations to calculate recalls.

**Phenotype-Gene relation (PGR) (Sousa et al., 2019)** This dataset concerns the relations between

[2] https://biocreative.bioinformatics.udel.edu/tasks/biocreative-vi/track-5/

[3] https://www.ncbi.nlm.nih.gov/pubmed/

[4] Peng et al. (2017) describe a solution for cross-sentence cases, which joins different dependency structures by connecting their roots. We leave it for future work.

human phenotypes (such as diseases) with human genes, where the relation set is a *binary* class on whether a phenotype is related to a gene. It has 18,451 silver training instances and 220 high-quality test instances, with each containing mention boundary annotations. We separate the first 15% training instances as our development set. Unlike *BioCreative VI CPR*, almost every relation of PGR is within a single sentence.

### 7.2 Models

We compare the following models:

- TEXTONLY: It does not take dependency structures and directly uses the Bi-LSTM outputs ($\boldsymbol{h}^{(0)}$ in Eq. 3) to make predictions.
- DEPTREE: Our baseline using 1-best dependency trees, as shown in Section 4.
- EDGEWISEPS and EDGEWISE: Our models using the forests generated by our EDGEWISE algorithm with or without parser scores.
- KBESTEISNERPS and KBESTEISNER: Our model using the forests generated by our KBESTEISNER algorithm with or without parser scores, respectively.

### 7.3 Settings

We take a state-of-the-art deep biaffine parser (Dozat and Manning, 2017), trained on the Penn Treebank (PTB) (Marcus and Marcinkiewicz, 1993) converted to Universal Dependency, to obtain 1-best trees and full search spaces for generating forests. Using standard PTB data split (02–21 for training, 22 for development and 23 for testing), it gives UAS and LAS scores of 95.7 and 94.6, respectively.

For the other hyper-parameters, word embeddings are initialized with the 200-dimensional BioASQ vectors[5], pretrained on 10M abstracts of biomedical articles, and are fixed during training. The dimension of hidden vectors in Bi-LSTM is set to 200, and the number of message passing steps $T$ is set to 2 based on Zhang et al. (2018b). We use Adam (Kingma and Ba, 2014), with a learning rate of 0.001, as the optimizer. The batch size, coefficient for $l2$ normalization loss and dropout rate are 20, $10^{-8}$ and 0.1, respectively.

[5]http://bioasq.lip6.fr/tools/BioASQword2vec/

| $\gamma$ | #Edge/#Node | LAS | Conn. Ratio(%) |
|---|---|---|---|
| 0.05 | 2.09 | 92.5 | 100.0 |
| 0.1 | 1.57 | 91.2 | 99.5 |
| 0.2 | 1.34 | 90.5 | 94.2 |
| 0.3 | 1.04 | 88.0 | 77.6 |
| $K$ | #Edge/#Node | LAS | Conn. Ratio(%) |
| 1 | 1.00 | 86.4 | 100.0 |
| 2 | 1.03 | 87.3 | 100.0 |
| 5 | 1.09 | 89.1 | 100.0 |
| 10 | 1.14 | 89.8 | 100.0 |

Table 1: Statistics on forests generated with various $\gamma$ (upper half) and $K$ (lower half) on the development set.

### 7.4 Analyses of generated forests

Table 1 demonstrates several characteristics of the generated forests of both the EDGEWISE and KBESTEISNER algorithms in Section 5.1, where "#Edge/#Sent" measures the forest density with the number of edges divided by the sentence length, "LAS" represents the oracle LAS score on 100 biomedical sentences with manually annotated dependency trees, and "Conn. Ratio (%)" shows the percentage of forests where both related entity mentions are connected.

Regarding the forest density, forests produced by EDGEWISE generally contain more edges than those from KBESTEISNER. Due to the combinatorial property of forests, EDGEWISE can give much more candidate trees (and sub-trees) for the whole sentence (and each sub-span). This coincides with the fact that the forests generated by EDGEWISE have higher oracle scores than these generated by KBESTEISNER.

For connectivity, KBESTEISNER guarantees to generate spanning forests. On the other hand, the connectivity ratio for the forests produced by EDGEWISE drops when increasing the threshold $\gamma$. We can have more than 94% being connected with $\gamma \leq 0.2$. Later we will show that good end-task performance can still be achieved with the 94% connectivity ratio. This indicates that losing connectivity for a small potion of the data may not hurt the overall performance.

### 7.5 Development results

Figure 3 shows the development experiments for our forest generation algorithms, where both EDGEWISE and KBESTEISNER give consistent

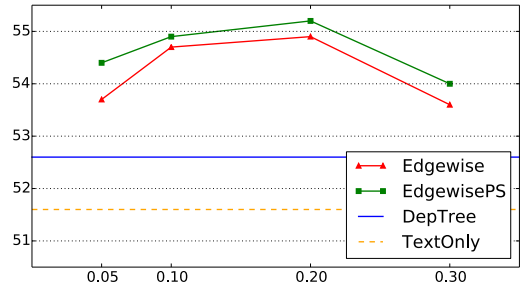


(a) EDGEWISE

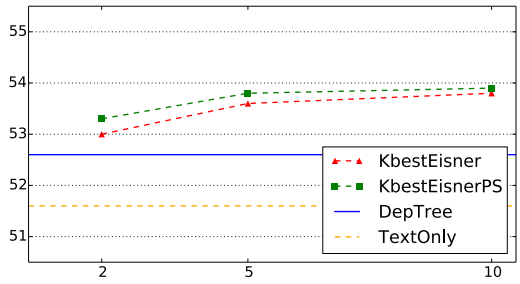


(b) KBESTEISNER

Figure 3: Development results (F1 score) for our forest generation methods.

improvements over DEPTREE and TEXTONLY. Generally, EDGEWISE gives more improvements than KBESTEISNER. The main reason may be that EDGEWISE generates denser forests, providing richer features. On the other hand, KBESTEISNER shows a marginal improvement by increasing $K$ from 5 to 10. This indicates that only merging 10-best trees may be far from sufficient. However, using a much larger $K$ (such as 100) is not practical due to dramatically increased computation time. In particular, the running time of KBESTEISNER with $K = 10$ is already much longer than that of EDGEWISE. As a result, EDGEWISE better serves our goal compared to KBESTEISNER. This may sound surprising, as EDGEWISE does not consider tree-level scores. It suggests that relation extraction may not require full dependency tree features. This coincides with previous relation extraction research (Bunescu and Mooney, 2005; Airola et al., 2008), which utilizes the shortest path connecting the two candidate entities in the dependency tree.

Leveraging parser confidence scores also consistently helps both methods. It is especially effective for EDGEWISE when $\gamma = 0.05$. This is likely because the parser confidence scores are useful for distinguishing some erroneous dependency arcs, when noise is large (e.g. when $\gamma$ is too small). Following the development results, we directly report the performances of EDGEWISEPS and KBESTEISNERPS, setting $\gamma$ and $K$ to 0.2 and 10, respectively, in our remaining experiments.

| Model | F1 score |
|---|---|
| GRU+Attn (Liu et al., 2017)† | 49.5 |
| Bran (Verga et al., 2018)† | 50.8 |
| TEXTONLY | 50.6 |
| DEPTREE | 51.4 |
| KBESTEISNERPS | 52.4** |
| EDGEWISEPS | **53.4**** |

Table 2: Test results of Biocreative VI CPR. † indicates previously reported numbers. ** means significant over DEPTREE at $p < 0.01$ with 1000 bootstrap tests (Efron and Tibshirani, 1994).

### 7.6 Main results on BioCreative VI CPR

Table 2 shows the main comparison results on the BioCreative CPR testset, with comparisons to the previous state-of-the-art and our baselines. *GRU+Attn* (Liu et al., 2017) stacks a self-attention layer on top of GRU (Cho et al., 2014) and embedding layers; *Bran* (Verga et al., 2018) adopts a biaffine self-attention model to simultaneously extract the relations of all mention pairs. Both methods use only textual knowledge.

TEXTONLY gives a performance comparable with *Bran*. With 1-best dependency trees, our DEPTREE baseline gives better performances than the previous state of the art. This confirms the usefulness of dependency structures and the effectiveness of GRN on encoding these structures. Using dependency forests and parser confidence scores, both KBESTEISNERPS and EDGEWISEPS obtain significantly higher numbers than DEPTREE. Consistent with the development experiments, EDGEWISEPS has a higher testset performance than KBESTEISNERPS.

### 7.7 Analysis

**Effectiveness on parsing accuracy** We have shown in Sections 7.5 and 7.6 that a dependency parser trained using a domain-general treebank can produce high-quality dependency forests in a target domain (biomedical) for helping relation extraction. This is based on the assumption of there being a high-quality treebank in a descent scale, which may not be true for low-resource languages. We simulate this low-resource effect by training our parser in much smaller treebanks of 1K or 5K

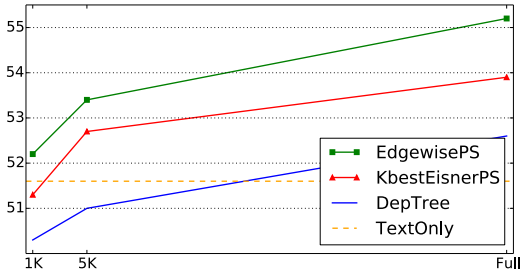


Figure 4: DEV results of BioCreative CPR regarding the dependency parsers trained on different number (1K, 5K or Full) of dependency trees.

dependency trees, respectively. The LAS scores for the resulting parsers on our 100 manually annotated biomedical dependency trees are 79.3 and 84.2, respectively, while the LAS score for the parser trained with the full treebank is 86.4, as shown in Table 1.

Figure 4 shows the results on the Biocreative CPR development set, where the performance of TEXTONLY is 51.6. DEPTREE fails to outperform TEXTONLY when only 1K or 5K dependency trees are available for training our parser. This is due to the low parsing recall and subsequent noise caused by the weak parsers. It confirms the previous conclusion that dependency structures are highly influential to the performance of relation extraction. Both EDGEWISEPS and KBESTEISNERPS are still more effective than DEPTREE. In particular, KBESTEISNERPS significantly improves TEXTONLY with 5K dependency trees, and EDGEWISEPS is helpful even with 1K dependency trees.

KBESTEISNER shows relatively smaller gaps than EDGEWISE when only a limited number of dependency trees are available. This is probably because considering whole-tree quality helps to better eliminate noise.

**Case study** Figure 5 illustrates two major types of errors in BioCreative CPR, which are caused by inaccurate 1-best dependency trees. As shown in Figure 5(a), the baseline system mistakenly predicts a "None" relation for that instance. This is mainly because "STAT3" is incorrectly linked to the main verb "inhibited" with a "punct" relation, but it should be linked to "AKT". In contrast, our forest contains the correct relation and with a probability of 0.18. This is possibly because "AKT and STAT3" fits the common pattern of "A and B" that conjunct two nouns.

Figure 5(b) shows another type of parsing errors that cause end-task mistakes. In this example, the multi-token mention "calcium modulated cyclases" is incorrectly segmented in the 1-best dependency tree, where "modulated" is used as the main verb of the whole sentence, leaving "cyclases" and "calcium" as the object and the modifier of the subject, respectively. However, this mention ought to be a noun phrase with "cyclases" being the head. Our forest helps in this case by providing a more reasonable structure (shown as the yellow dashed arcs), where both "calcium" and "modulated" modify "cyclases". This is likely because "modulated" can be interpreted as an adjective in addition to being a verb. It shows the advantage of keeping multiple candidate syntactic arcs.

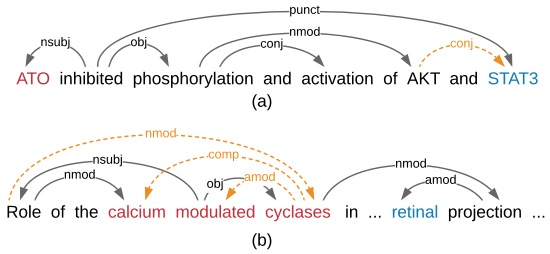


Figure 5: Two representative cases in BioCreative CPR, contrasting 1-best trees and forests, where irrelevant content and arcs are omitted for simplicity.

| Model | F1 score |
|---|---|
| BO-LSTM (Lamurias et al., 2019)† | 52.3 |
| BioBERT (Lee et al., 2019)† | 67.2 |
| TEXTONLY | 76.0 |
| DEPTREE | 78.9 |
| KBESTEISNERPS | 83.6* |
| EDGEWISEPS | **85.7**** |

Table 3: Main results on PGR testest. † denotes previous numbers rounded into 3 significant digits. * and ** indicate significance over DEPTREE at $p < 0.05$ and $p < 0.01$ with 1000 bootstrap tests.

### 7.8 Main results on PGR

Table 3 shows the comparison with previous work on the PGR testset, where our models are significantly better than the existing models. This is likely because the previous models do not utilize all the information from inputs: *BO-LSTM* only takes the words (without arc labels) along the shortest dependency path between the target mentions; the pretrained weights of *BioBERT* are kept constant during training for relation extraction.

With 1-best trees, DEPTREE is 2.9 points bet-

| Model | F1 score |
|---|---|
| C-GCN (Zhang et al., 2018b)† | 84.8 |
| C-AGGCN (Guo et al., 2019)† | 85.7 |
| DEPTREE | 84.6 |
| KBESTEISNERPS | 85.8 |
| EDGEWISEPS | 86.3 |

Table 4: Main results on SemEval-2010 task 8 testest. † denotes previous numbers.

ter than TEXTONLY, confirming the usefulness of dependency structures. Leveraging dependency forests, both KBESTEISNERPS and EDGEWISEPS significantly outperform DEPTREE with $p$-values of 0.003 and 0.024, respectively. This further confirms the usefulness of dependency forests for medical relation extraction.

### 7.9 Main results on SemEval-2010 task 8

In addition to the biomedical domain, leveraging dependency forests applies to other domains as well. As shown in Table 4, we conduct a preliminary study on SemEval-2010 task 8 (Hendrickx et al., 2009), a widely used benchmark for news-domain relation extraction. It is a public dataset, containing 10,717 instances (8000 for training and development, 2717 for testing) with 19 relations: 9 directed relations and a special "Other" class. Both *C-GCN* and *C-AGGCN* take a similar network as ours by stacking a graph neural network for encoding trees on top of a Bi-LSTM layer for encoding sentences.

DEPTREE achieves similar performance as *C-GCN* and is slightly worse than *C-AGGCN*, with one potential reason being that *C-AGGCN* takes more parameters. Using forests, both KBESTEISNERPS and EDGEWISEPS outperform DEPTREE with the same number of parameters, and they show comparable and slightly better performances than *C-AGGCN*. Again, EDGEWISEPS is better than KBESTEISNERPS, showing that the former is a better way for generating forests.

## 8 Conclusion

We have proposed two algorithms for generating high-quality dependency forests for relation extraction, and studied a graph recurrent network for effectively distinguishing useful features from noise in parsing forests. Experiments on two biomedical relation extraction benchmarks show the superiority of forests versus tree structures, without introducing any additional model parameters. Our deep analyses indicate that the main advantage comes from alleviating out-of-domain parsing errors.

**Acknowledgments** Research supported by NSF award IIS-1813823.

## References

Asma Ben Abacha and Pierre Zweigenbaum. 2011. Automatic extraction of semantic relations between medical entities: a rule based approach. *Journal of biomedical semantics*, 2(5).

Antti Airola, Sampo Pyysalo, Jari Björne, Tapio Pahikkala, Filip Ginter, and Tapio Salakoski. 2008. All-paths graph kernel for protein-protein interaction extraction with evaluation of cross-corpus learning. *BMC bioinformatics*, 9(11):S2.

Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Simaan. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Daniel Beck, Gholamreza Haffari, and Trevor Cohn. 2018. Graph-to-sequence learning using gated graph neural networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Razvan Bunescu and Raymond Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*.

Marie Candito, Enrique Henestroza Anguiano, and Djamé Seddah. 2011. A word clustering approach to domain adaptation: Effective parsing of biomedical texts. In *Proceedings of the 12th International Conference on Parsing Technologies*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Faisal Md. Chowdhury, Alberto Lavelli, and Alessandro Moschitti. 2011. A study on dependency tree kernels for automatic extraction of protein-protein interaction. In *Proceedings of BioNLP 2011 Workshop*.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*.

Timothy Dozat and Christopher D Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proceedings of International Conference on Learning Representations*.

Bradley Efron and Robert J Tibshirani. 1994. *An introduction to the bootstrap*. CRC press.

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th conference on Computational linguistics-Volume 1*.

Carol Friedman, Pauline Kra, Hong Yu, Michael Krauthammer, and Andrey Rzhetsky. 2001. Genies: a natural-language processing system for the extraction of molecular pathways from journal articles. *Bioinformatics*, 17(1).

Matthew R Gormley, Mo Yu, and Mark Dredze. 2015. Improved relation extraction with feature-rich compositional embedding models. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1774–1784.

Zhijiang Guo, Yan Zhang, and Wei Lu. 2019. Attention guided graph convolutional networks for relation extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 241–251.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2009. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of SemEval*, pages 94–99.

Lynette Hirschman, Alexander Yeh, Christian Blaschke, and Alfonso Valencia. 2005. Overview of biocreative: critical assessment of information extraction for biology.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8).

Liang Huang and David Chiang. 2005. Better k-best parsing. In *Proceedings of the Ninth International Workshop on Parsing Technology*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Martin Krallinger, Obdulia Rabal, Saber A Akhondi, et al. 2017. Overview of the biocreative vi chemical-protein interaction track. In *Proceedings of the VI BioCreative challenge evaluation workshop*.

Andre Lamurias, Diana Sousa, Luka A Clarke, and Francisco M Couto. 2019. BO-LSTM: classifying relations via long short-term memory networks along biomedical ontologies. *BMC bioinformatics*, 20(1):10.

Phong Le and Willem Zuidema. 2015. The forest convolutional network: Compositional distributional semantics with a neural chart and without binarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Matthew Lease and Eugene Charniak. 2005. Parsing biomedical literature. In *International Conference on Natural Language Processing*.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. Biobert: pre-trained biomedical language representation model for biomedical text mining. *arXiv preprint arXiv:1901.08746*.

Jiangming Liu and Yue Zhang. 2017. In-order transition-based constituent parsing. *Transactions of the Association for Computational Linguistics*, 5.

Sijia Liu, Feichen Shen, Yanshan Wang, Majid Rastegar-Mojarad, Ravikumar Komandur Elayavilli, Vipin Chaudhary, and Hongfang Liu. 2017. Attention-based neural networks for chemical protein relation extraction. In *Proceedings of the BioCreative VI Workshop*.

Yang Liu, Furu Wei, Sujian Li, Heng Ji, Ming Zhou, and Houfeng WANG. 2015. A dependency-based neural network for relation classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*.

Wei Lu and Hwee Tou Ng. 2011. A probabilistic forest-to-string model for language generation from typed lambda calculus expressions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Chunpeng Ma, Akihiro Tamura, Masao Utiyama, Tiejun Zhao, and Eiichiro Sumita. 2018. Forest-based neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Mitchell P Marcus and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19(2).

David McClosky and Eugene Charniak. 2008. Self-training for biomedical parsing. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*.

Farrokh Mehryary, Jari Björne, Sampo Pyysalo, Tapio Salakoski, and Filip Ginter. 2016. Deep learning with minimal training data: TurkuNLP entry in the BioNLP shared task 2016. In *Proceedings of the 4th BioNLP Shared Task Workshop*.

Haitao Mi, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *Proceedings of ACL-08: HLT*.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen-tau Yih. 2017. Cross-sentence n-ary relation extraction with graph LSTMs. *Transactions of the Association for Computational Linguistics*, 5:101–115.

Chris Quirk and Hoifung Poon. 2017. Distant supervision for relation extraction beyond the sentence boundary. In *Proceedings of the 15th Conference of the European Chapter of the ACL (EACL-17)*.

Kenji Sagae, Yusuke Miyao, Rune Sætre, and Jun'ichi Tsujii. 2008. Evaluating the effects of treebank size in a practical application for parsing. In *Software Engineering, Testing, and Quality Assurance for Natural Language Processing*.

Parikshit Sondhi, Manish Gupta, ChengXiang Zhai, and Julia Hockenmaier. 2010. Shallow information extraction from medical forum data. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*.

Linfeng Song, Daniel Gildea, Yue Zhang, Zhiguo Wang, and Jinsong Su. 2019. Semantic neural machine translation using amr. *Transactions of the Association for Computational Linguistics*, 7:19–31.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018a. A graph-to-sequence model for amr-to-text generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1616–1626.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018b. N-ary relation extraction using graph-state lstm. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2226–2235.

Diana Sousa, André Lamúrias, and Francisco M Couto. 2019. A silver standard corpus of human phenotype-gene relations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Zhaopeng Tu, Yang Liu, Young-Sook Hwang, Qun Liu, and Shouxun Lin. 2010. Dependency forest for statistical machine translation. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*.

Patrick Verga, Emma Strubell, and Andrew McCallum. 2018. Simultaneously self-attending to all mentions for full-abstract biological relation extraction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*.

Hua Xu, Shane P Stenner, Son Doan, Kevin B Johnson, Lemuel R Waitman, and Joshua C Denny. 2010. Medex: a medication information extraction system for clinical narratives. *Journal of the American Medical Informatics Association*, 17(1).

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2015a. Semantic relation classification via convolutional neural networks with simple negative sampling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Kun Xu, Lingfei Wu, Zhiguo Wang, Mo Yu, Liwei Chen, and Vadim Sheinin. 2018. Exploiting rich syntactic information for semantic parsing with graph-to-sequence model. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015b. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Yongjing Yin, Linfeng Song, Jinsong Su, Jiali Zeng, Chulun Zhou, and Jiebo Luo. 2019. Graph-based neural sentence ordering. In *Proceedings of IJCAI*.

Hong Yu and Eugene Agichtein. 2003. Extracting synonymous gene and protein terms from biological literature. *Bioinformatics*, 19.

Poorya Zaremoodi and Gholamreza Haffari. 2018. Incorporating syntactic uncertainty in neural machine translation with a forest-to-sequence model. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1421–1429.

Yue Zhang, Qi Liu, and Linfeng Song. 2018a. Sentence-state lstm for text representation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 317–327.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018b. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

# Open Relation Extraction: Relational Knowledge Transfer from Supervised Data to Unsupervised Data

**Ruidong Wu**[1*], **Yuan Yao**[1*], **Xu Han**[1], **Ruobing Xie**[2],
**Zhiyuan Liu**[1†], **Fen Lin**[2], **Leyu Lin**[2], **Maosong Sun**[1]
[1]Department of Computer Science and Technology, Tsinghua University, Beijing, China
Institute for Artificial Intelligence, Tsinghua University, Beijing, China
State Key Lab on Intelligent Technology and Systems, Tsinghua University, Beijing, China
[2]Search Product Center, WeChat Search Application Department, Tencent, China
mooninsiderain@gmail.com

## Abstract

Open relation extraction (OpenRE) aims to extract relational facts from the open-domain corpus. To this end, it discovers relation patterns between named entities and then clusters those semantically equivalent patterns into a united relation cluster. Most OpenRE methods typically confine themselves to unsupervised paradigms, without taking advantage of existing relational facts in knowledge bases (KBs) and their high-quality labeled instances. To address this issue, we propose Relational Siamese Networks (RSNs) to learn similarity metrics of relations from labeled data of pre-defined relations, and then transfer the relational knowledge to identify novel relations in unlabeled data. Experiment results on two real-world datasets show that our framework can achieve significant improvements as compared with other state-of-the-art methods. Our code is available at https://github.com/thunlp/RSN.

## 1 Introduction

Relation extraction (RE) aims to extract relational facts between two entities from plain texts. For example, with the sentence *"Hayao Miyazaki is the director of the film 'The Wind Rises'"*, we can extract a relation "director_of" between two entities "*Hayao Miyazaki*" and "*The Wind Rises*".

Recent progress in supervised methods to RE has achieved great successes. Supervised methods can effectively learn significant relation semantic patterns based on existing labeled data, but the data constructions are time-consuming and human-intensive. To lower the level of supervision, several semi-supervised approaches have been developed, including bootstrapping, active learning, label propagation (Pawar et al., 2017).

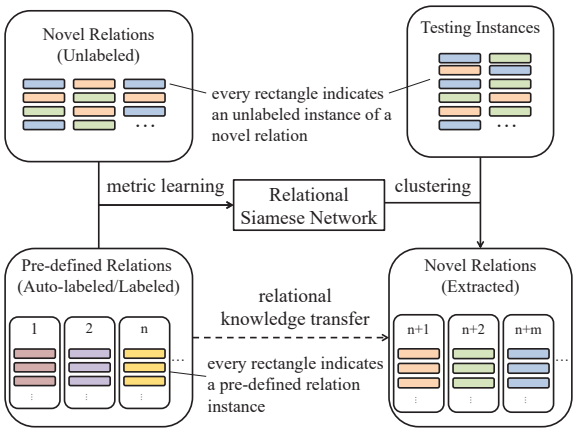


Figure 1: A flowchart of our framework. Our model RSN learns from both labeled instances of pre-defined relations and unlabeled instances of new relations, and tries to cluster testing instances of new relations.[1]

Mintz (2009) also proposes distant supervision to generate training data automatically. It assumes that if two entities have a relation in KBs, all sentences that contain these two entities will express this relation. Still, all these approaches can only extract pre-defined relations that have already appeared either in human-annotated datasets or KBs. It is hard for them to cover the great variety of novel relational facts in the open-domain corpora.

Open relation extraction (OpenRE) aims to extract relational facts on the open-domain corpus, where the relation types may not be predefined. There are some efforts concentrating on extracting triples with new relation types. Banko (2008) directly extracts words or phrases in sentences to represent new relation types. However, some relations cannot be explicitly represented with tokens in sentences, and it is hard to align different relational tokens that exactly have the same meanings. Yao (2011) consid-

[*] indicates equal contribution
[†] Corresponding author: Z.Liu(liuzy@tsinghua.edu.cn)

[1]To highlight our model's ability to extract new relations, testing instances only contain new relations.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 219–228,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

ers OpenRE as a clustering task for extracting triples with new relation types. However, previous clustering-based OpenRE methods (Yao et al., 2011, 2012; Marcheggiani and Titov, 2016; Elsahar et al., 2017) are mostly unsupervised, and cannot effectively select meaningful relation patterns and discard irrelevant information.

In this paper, we propose to take advantage of high-quality supervised data of pre-defined relations for OpenRE. The approach is non-trivial, however, due to the considerable gap between the pre-defined relations and novel relations of interest in open domain. To bridge the gap, we propose **Relational Siamese Networks** (RSNs) to learn transferable relational knowledge from supervised data for OpenRE. Specifically, RSNs learn relational similarity metrics from labeled data of pre-defined relations, and then transfer the metrics to measure the similarity of unlabeled sentences for open relation clustering. We describe the flowchart of our framework in Figure 1.

Moreover, we show that RSNs can also be generalized to various weakly-supervised scenarios. We propose **Semi-supervised RSN** to learn from both supervised data of pre-defined relations and unsupervised data with novel relations, and **Distantly-supervised RSN** to learn from distantly-supervised data and unsupervised data.

We conduct experiments on real-world RE datasets, FewRel and FewRel-distant, by splitting relations into seen and unseen set, and evaluate our models in supervised, semi-supervised, and distantly-supervised scenarios. The results demonstrate that our models significantly outperform state-of-the-art baseline methods in all scenarios without using external linguistic tools. To summarize, the main contributions of this work are as follows:

(1) We develop a novel relational knowledge transfer framework RSN for OpenRE, which can effectively transfer existing relational knowledge to novel-relation data and accurately identify novel relations. To the best of our knowledge, RSN is the first model to consider knowledge transfer in clustering-based OpenRE task.

(2) We further propose Semi-supervised RSNs and Distantly-supervised RSNs that can learn from various weakly supervised scenarios. The experimental results show that all these RSN models achieve significant improvements in F-measure compared with state-of-the-art baselines.

## 2 Related Work

**Open Relation Extraction**. Relation extraction (RE) is an important task in NLP. Traditional RE methods mainly concentrate on classifying relational facts into pre-defined relation types (Mintz et al., 2009; Yu et al., 2017). Zeng (2014) utilizes CNN encoders to build sentence representations with the help of position embeddings. Lin (2016) further improves RE performance on distantly-supervised data via instance-level attention. These methods take advantage of supervised or distantly-supervised data to learn neural sentence encoders for distributed representations, and have achieved promising results. However, these methods cannot handle the open-ended growth of new relation types in the open-domain corpora.

To solve this problem, recently many efforts have been invested in exploring methods for open relation extraction (OpenRE), which aims to discover new relation types from unsupervised open-domain corpora. OpenRE methods can be roughly divided into two categories: tagging-based and clustering-based. Tagging-based methods cast OpenRE as a sequence labeling problem, and extract relational phrases consisting of words from sentences in unsupervised (Banko et al., 2007; Banko and Etzioni, 2008) or supervised paradigms (Jia et al., 2018; Cui et al., 2018; Stanovsky et al., 2018). However, tagging-based methods often extract multiple overly-specific relational phrases for the same relation type, and cannot be readily utilized for downstream tasks.

In comparison, conventional clustering-based OpenRE methods extract rich features for relation instances via external linguistic tools, and cluster semantic patterns into several relation types (Lin and Pantel, 2001; Yao et al., 2011, 2012). Marcheggiani (2016) proposes a reconstruction-based model discrete-state variational autoencoder for OpenRE via unlabeled instances. Elsahar (2017) utilizes a clustering algorithm over linguistic features. In this paper, we focus on the clustering-based OpenRE methods, which have the advantage of discovering highly distinguishable relation types.

**Few-shot Learning**. Few-shot learning aims to classify instances with a handful of labeled samples. Many efforts are devoted to few-shot image classification (Koch et al., 2015) and relation classification (Yuan et al., 2017; Han et al., 2018). Notably, (Koch et al., 2015) introduces Convolu-

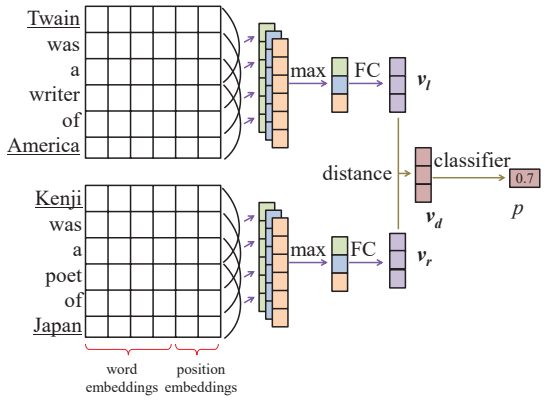


Figure 2: The architecture of Relational Siamese Networks. The output is the similarity between two relational instances.

tional Siamese Neural Network for image metric learning, which inspires us to learn relational similarity metrics for OpenRE.

**Semi-supervised Clustering**. Semi-supervised clustering aims to cluster semantic patterns given instance seeds of target categories (Bair, 2013; Hongtao Lin, 2019). Differently, our proposed Semi-supervised RSN only leverages labeled instances of pre-defined relations, and does not need any seed of new relations.

## 3 Methodology

Our OpenRE framework mainly consists of two modules, the relation similarity calculation module and the relation clustering module. For relation similarity calculation, we propose Relational Siamese Networks (RSNs), which learn to predict whether two sentences mention the same relation. To utilize large-scale unsupervised data and distantly-supervised data, we further propose Semi-supervised RSN and Distantly-supervised RSN. Finally, in the relation clustering module, with the learned relation metric, we utilize hierarchical agglomerative clustering (HAC) and Louvain clustering algorithms to cluster target relation instances of new relation types.

### 3.1 Relational Siamese Network (RSN)

The architecture of our Relational Siamese Networks is shown in Figure 2. CNN modules encode a pair of relational instances into vectors, and several shared layers compute their similarity.

**Sentence Encoder**. We use a CNN module as the sentence encoder. The CNN module includes an embedding layer, a convolutional layer, a max-pooling layer, and a fully-connected (FC) layer. The embedding layer transforms the words in a sentence $x$ and the positions of entities $e_{head}$ and $e_{tail}$ into pre-trained word embeddings and random-initialized position embeddings. Following (Zeng et al., 2014), we concatenate these embeddings to form a vector sequence. Next, a one-dimensional convolutional layer and a max-pooling layer transform the vector sequence into features. Finally, an FC layer with sigmoid activation maps features into a relational vector $\boldsymbol{v}$. To summarize, we obtain a vector representation $\boldsymbol{v}$ for a relational sentence with our CNN module:

$$\boldsymbol{v} = \mathrm{CNN}(\mathrm{s}), \tag{1}$$

in which we denote the joint information of a sentence $x$ and two entities in it $e_{head}$ and $e_{tail}$ as a data sample $s$. And with paired input relational instances, we have:

$$\boldsymbol{v_l} = \mathrm{CNN}(\mathrm{s_l}), \boldsymbol{v_r} = \mathrm{CNN}(\mathrm{s_r}), \tag{2}$$

in which two CNN modules are identical and share all the parameters.

**Similarity Computation**. Next, to measure the similarity of two relational vectors, we calculate their absolute distance and transform it into a real-number similarity $p \in [0, 1]$. First, a distance layer computes the element-wise absolute distance of two vectors:

$$\boldsymbol{v_d} = |\boldsymbol{v_l} - \boldsymbol{v_r}|. \tag{3}$$

Then, a classifier layer calculates a metric $p$ for relation similarity. The layer is a one-dimensional-output FC layer with sigmoid activation:

$$p = \sigma(\boldsymbol{k}\boldsymbol{v_d} + b), \tag{4}$$

in which $\sigma$ denotes the sigmoid function, $\boldsymbol{k}$ and $b$ denote the weights and bias. To summarize, we obtain a good similarity metric $p$ of relational instances.

**Cross Entropy Loss.** The output of RSN $p$ can also be explained as the probability of two sentences mentioning two different relations. Thus, we can use binary labels $q$ and binary cross entropy loss to train our RSN:

$$\mathcal{L}_l = \mathbb{E}_{d_l \sim \mathcal{D}_l}[q \ln(p_\theta(d_l)) + (1-q)\ln(1-p_\theta(d_l))], \tag{5}$$

in which $\theta$ indicates all the parameters in the RSN.

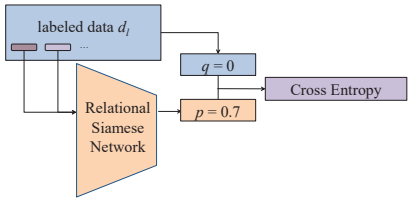


(a) Supervised RSN

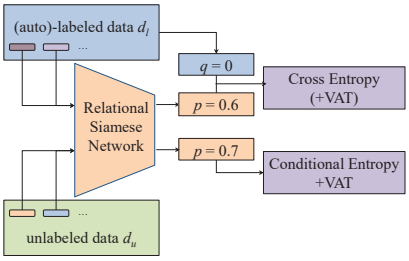


(b) Weakly-supervised RSNs

Figure 3: The comparison of (a) Supervised RSN and (b) Weakly-supervised RSNs. Weakly-supervised RSNs, including Semi-supervised RSN and Distantly-supervised RSN, further learn from unlabeled data with conditional entropy minimization and virtual adversarial training (VAT). In figures, $p$ indicates the predicted similarity of two relational sentences, while $q$ indicates the ground-truth label between them.

### 3.2 Semi-supervised RSN

To discover relation clusters in the open-domain corpus, it is beneficial to not only learn from labeled data, but also capture the manifold of unlabeled data in the semantic space. To this end, we need to push the decision boundaries away from high-density areas, which is known as the cluster assumption (Chapelle and Zien, 2005).

We try to achieve this goal with several additional loss functions. In the following paragraphs, we denote the labeled training dataset as $\mathcal{D}_l$ and a couple of labeled relational instances as $d_l$. Similarly, we denote the unlabeled training dataset as $\mathcal{D}_u$ and a couple of unlabeled instances as $d_u$.

**Conditional Entropy Loss.** In classification problems, a well-classified embedding space usually reserves large margins between different classified clusters, and optimizing margin can be a promising way to facilitate training. However, in clustering problems, type labels are not available during training. To optimize margin without explicit supervision, we can push the data points away from the decision boundaries. Intuitively, when the distance similarity $p$ between two relational instances equals 0.5, there is a high probability that at least one of two instances is near the decision boundary between relation clusters. Thus, we use the conditional entropy loss (Grandvalet and Bengio, 2005), which reaches the maximum when $p = 0.5$, to penalize close-boundary distribution of data points:

$$\begin{aligned}\mathcal{L}_u = \mathbb{E}_{d_u \sim \mathcal{D}_u}[p_\theta(d_u)\ln(p_\theta(d_u)) + \\ (1 - p_\theta(d_u))\ln(1 - p_\theta(d_u))].\end{aligned} \tag{6}$$

**Virtual Adversarial Loss.** Despite its theoretical promise, conditional entropy minimization suffers from shortcomings in practice. Due to neural networks' strong fitting ability, a very complex decision hyperplane might be learned so as to keep away from all the training samples, which lacks generalizability. As a solution, we can smooth the relational representation space with locally-Lipschitz constraint.

To satisfy this constraint, we introduce virtual adversarial training (Miyato et al., 2016) on both branches of RSN. Virtual adversarial training can search through data point neighborhoods, and penalize most sharp changes in distance prediction. For labeled data, we have

$$\mathcal{L}_{vl} = \mathbb{E}_{d_l \sim \mathcal{D}_l}[\mathrm{D}_{\mathrm{KL}}(p_\theta(d_l)||p_\theta(d_l, t_1, t_2))], \tag{7}$$

in which $\mathrm{D}_{\mathrm{KL}}$ indicates the Kullback-Leibler divergence, $p_\theta(d_l, t_1, t_2)$ indicates a new distance estimation with perturbations $t_1$ and $t_2$ on both input instances respectively. Specifically, $t_1$ and $t_2$ are worst-case perturbations that maximize the KL divergence between $p_\theta(d_l)$ and $p_\theta(d_l, t_1, t_2)$ with a limited length. Empirically, we approximate the perturbations the same as the original paper (Miyato et al., 2016). Specifically, we first add a random noise to the input, and calculate the gradient of the KL-divergence between the outputs of the original input and the noisy input. We then add the normalized gradient to the original input and get the perturbed input. And for unlabeled data, we have

$$\mathcal{L}_{vu} = \mathbb{E}_{d_u \sim \mathcal{D}_u}[\mathrm{D}_{\mathrm{KL}}(p_\theta(d_u)||p_\theta(d_u, t_1, t_2))], \tag{8}$$

in which the perturbations $t_1$ and $t_2$ are added to word embeddings rather than the words themselves.

To summarize, we use the following loss function to train Semi-supervised RSN, which learns from both labeled and unlabeled data:

$$\mathcal{L}_{all} = \mathcal{L}_l + \lambda_v \mathcal{L}_{vl} + \lambda_u(\mathcal{L}_u + \lambda_v \mathcal{L}_{vu}), \tag{9}$$

in which $\lambda_v$ and $\lambda_u$ are two hyperparameters.

### 3.3 Distantly-supervised RSN

To alleviate the intensive human labor for annotation, the topic of distantly-supervised learning has attracted much attention in RE. Here, we propose Distantly-supervised RSN, which can learn from both distantly-supervised data and unsupervised data for relational knowledge transfer. Specifically, we use the following loss function:

$$\mathcal{L}_{all} = \mathcal{L}_l + \lambda_u(\mathcal{L}_u + \lambda_v \mathcal{L}_{vu}), \tag{10}$$

which treats auto-labeled data as labeled data but removes the virtual adversarial loss on the auto-labeled data.

The reason to remove the loss is simple: virtual adversarial training on auto-labeled data can amplify the noise from false labels. Indeed, we do find that the virtual adversarial loss on auto-labeled data can harm our model's performance in experiments.

We do not use more denoising methods, since we think RSN has some inherent advantages of tolerating such noise. Firstly, the noise will be overwhelmed by the large proportion of negative sampling during training. Secondly, during clustering, the prediction of a new relation cluster is based on areas where the density of relational instances is high. Outliers from noise, as a result, will not influence the prediction process so much.

### 3.4 Open Relation Clustering

After RSN is learned, we can use RSN to calculate the similarity matrix of testing instances. With this matrix, several clustering methods can be applied to extract new relation clusters.

**Hierarchical Agglomerative Clustering**. The first clustering method we adopt is hierarchical agglomerative clustering (HAC). HAC is a bottom-up clustering algorithm. At the start, every testing instance is regarded as a cluster. For every step, it agglomerates two closest instances. There are several criteria to evaluate the distance between two clusters. Here, we adopt the complete-linkage criterion, which is more robust to extreme instances.

However, there is a significant shortcoming of HAC: it needs the exact number of clusters in advance. A potential solution is to stop agglomerating according to an empirical distance threshold, but it is hard to determine such a threshold. This problem leads us to consider another clustering algorithm Louvain (Blondel et al., 2008).

**Louvain**. Louvain is a graph-based clustering algorithm traditionally used for detecting communities. To construct the graph, we use the binary approximation of RSN's output, with $0$ indicating an edge between two nodes. The advantage of Louvain is that it does not need the number of potential clusters beforehand. It will automatically find proper sizes of clusters by optimizing community modularity. According to the experiments we conduct, Louvain performs better than HAC.

After running, Louvain might produce a number of singleton clusters with few instances. It is not proper to call these clusters new relation types, so we label these instances the same as their closest labeled neighbors.

Finally, we want to explain the reason why we do not use some other common clustering methods like K-Means, Mean-Shift and Ward's (Ward Jr, 1963) method of HAC: these methods calculate the centroid of several points during clustering by merely averaging them. However, the relation vectors in our model are high-dimensional, and the distance metric described by RSN is non-linear. Consequently, it is not proper to calculate the centroid by simply averaging the vectors.

## 4 Experiments

In this section, we conduct several experiments on real-world RE datasets to show the effectiveness of our models, and give a detailed analysis to show its advantages.

### 4.1 Dataset

In experiments, we use FewRel (Han et al., 2018) as our first dataset. FewRel is a human-annotated dataset containing $80$ types of relations, each with $700$ instances. An advantage of FewRel is that every instance contains a unique entity pair, so RE models cannot choose the easy way to memorize the entities.

We use the original train set of FewRel, which contains $64$ relations, as labeled set with predefined relations, and the original validation set of FewRel, which contains $16$ new relations, as the unlabeled set with novel relations to extract. We then randomly choose $1,600$ instances from the unlabeled set as the test set, with the rest labeled and unlabeled instances considered as the train set.

The second dataset we use is FewRel-distant, which contains the distantly-supervised data obtained by the authors of FewRel before human an-

notation. We follow the split of FewRel to obtain the auto-labeled train set and unlabeled train set. For evaluation, we use the human-annotated test set of FewRel with $1,600$ instances. Unlabeled instances already existing in this test set are removed from the unlabeled train set of FewRel-distant. Finally, the auto-labeled train set contains $323,549$ relational instances, and the unlabeled train set contains $60,581$ instances.

A previous OpenRE work reports performance on an unpublic dataset called NYT-FB (Marcheggiani and Titov, 2016). However, it has several shortcomings compared with FewRel-distant. First, NTY-FB's test set is distantly-supervised and is noisy for instance-level RE. Moreover, instances in NYT-FB often share entity pairs or relational phrases, which makes it much easier for relation clustering. Therefore, we think the results on FewRel-distant are convincing enough for Distantly-supervised OpenRE.

### 4.2 Implementation Details

**Data Sampling**. The input of RSN should be a pair of sampled instances. For the unlabeled set, the only possible sampling method is to select two instances randomly. For the labeled set, however, random selection would result in too many different-relation pairs, and cause severe biases for RSN. To solve this problem, we use down-sampling. In our experiments, we fix the percentage of same-relation pairs in every labeled data batch as $6\%$.

Let us denote this percentage number as the sample ratio for convenience. Experimental results show that the sample ratio decides RSN's tendency to predict larger or smaller clusters. In other words, it controls the granularity of the predicted relation types. This phenomenon suggests a potential application of our model in hierarchical relation extraction. However, we leave any serious discussion to future work.

**Hyperparameter Settings**. Following (Lin et al., 2016) and (Zeng et al., 2014), we fix the less influencing hyperparameters for sentence encoding as their reported optimal values. For word embeddings, we use pre-trained $50$-dimensional Glove (Pennington et al., 2014) word embeddings. For position embeddings, we use random-initialized $5$-dimensional position embeddings. During training, all the embeddings are trainable. For the neural network, the number of feature maps in the convolutional layer is $230$. The filter length is $3$. The activation function after the max-pooling layer is ReLU, and the activation functions after FC layers are sigmoid. Besides, we adopt two regularization methods in the CNN module. We put a dropout layer right after the embedding layer as (Miyato et al., 2016). The dropout rate is $0.2$. We also impose L2 regularization on the convolutional layer and the FC layer, with parameters of $0.0002$ and $0.001$ respectively. Hyperparameters for virtual adversarial training are just the same as (Miyato et al., 2016) proposed.

At the same time, major hyperparameters are selected with grid search according to the model performance on a validation set. Specifically, the validation set contains 10,000 randomly chosen sentence pairs from the unlabeled set (i.e. 16 novel relations) and does not overlap with the test set. The model is evaluated according to the precision of binary classification of sentence pairs on the validation set, which is an estimation for models' clustering ability. We do not use F1 during model validation because the clustering steps are time-consuming.

For optimization, we use Adam optimizer (Kingma and Ba, 2014) with a learning rate of 0.0001, which is selected from $\{0.1, 0.01, 0.001, 0.0001, 0.00001\}$. The batch size is 100 selected from $\{25, 50, 100\}$. For hyperparameters in Equation 9 and Equation 10, $\lambda_v$ is $1.0$ selected from $\{0.1, 0.5, 1.0, 2.0\}$ and $\lambda_u$ is $0.03$ selected from $\{0.01, 0.02, 0.03, 0.04, 0.05\}$.

For baseline models, original papers do grid search for all possible hyperparameters and report the best result during testing. We follow their settings and do grid search directly on the test set.

### 4.3 Experiment Results on OpenRE

In this section, we demonstrate the effectiveness of our RSN models by comparing our models with state-of-the-art clustering-based OpenRE methods. We also conduct ablation experiments to detailedly investigate the contributions of different mechanisms of Semi-supervised RSN and Distantly-supervised RSN.

**Baselines**. Conventional clustering-based OpenRE models usually cluster instances by either clustering their linguistic features (Lin and Pantel, 2001; Yao et al., 2012; Elsahar et al., 2017) or imposing reconstruction constraints (Yao et al., 2011; Marcheggiani and Titov, 2016). To demonstrate

the effectiveness of our RSN models, we compare our models with two state-of-the-art models:

(1) HAC with re-weighted word embeddings (RW-HAC) (Elsahar et al., 2017): RW-HAC is the state-of-the-art feature clustering model for OpenRE. The model first extracts KB types and NER tags of entities as well as re-weighted word embeddings from sentences, then adopts principal component analysis (PCA) to reduce feature dimensionality, and finally uses HAC to cluster the concatenation of reduced feature representations.

(2) Discrete-state variational autoencoder (VAE) (Marcheggiani and Titov, 2016): VAE is the state-of-the-art reconstruction-based model for OpenRE via unlabeled instances. It optimizes a relation classifier by reconstructing entities from pairing entities and predicted relation types. Rich features including entity words, context words, trigger words, dependency paths, and context POS tags are used to predict the relation type.

RW-HAC and VAE both rely on external linguistic tools to extract rich features from plain texts. Specifically, we first align entities to Wikidata and get their KB types. Next, we preprocess the instances with part-of-speech (POS) tagging, named-entity recognition (NER), and dependency parsing with Stanford CoreNLP (Manning et al., 2014). It is worth noting that these features are only used by baseline models. Our models, in contrast, only use sentences and entity pairs as inputs.

**Evaluation Protocol**. In evaluation, we use $B^3$ metric (Bagga and Baldwin, 1998) as the scoring function. $B^3$ metric is a standard measure to balance the precision and recall of clustering tasks, and is commonly used in previous OpenRE works (Marcheggiani and Titov, 2016; Elsahar et al., 2017). To be specific, we use $F_1$ measure, the harmonic mean of precision and recall.

First, we report the result of supervised RSN with different clustering methods. Specifically, **SN** represents the original RSN structure, **HAC** and **L** indicate HAC and Louvain clustering introduced in Sec. 3.3. The result shows that Louvain performs better than HAC, so in the following experiments we focus on using Louvain clustering.

Next, for Semi-supervised and Distantly-supervised RSN, we conduct various combinations of different mechanisms to verify the contribution of each part. **(+C)** indicates that the model is powered up with conditional entropy minimization, while **(+V)** indicates that the model is powered up with virtual adversarial training.

| Approach | FewRel $P$ | FewRel $R$ | FewRel $F_1$ | FewRel-distant $P$ | FewRel-distant $R$ | FewRel-distant $F_1$ |
|---|---|---|---|---|---|---|
| VAE | 17.9 | 69.7 | 28.5 | 17.9 | 69.7 | 28.5 |
| RW-HAC | 31.8 | 46.0 | 37.6 | 31.8 | 46.0 | 37.6 |
| SN-HAC | 36.2 | 53.3 | 43.1 | 34.5 | 53.3 | 41.5 |
| SN-L | 36.5 | 69.2 | 47.8 | 34.6 | 59.8 | 43.9 |
| SN-L+V | 46.1 | 77.3 | 57.8 | 40.7 | 52.4 | 45.8 |
| SN-L+C | 47.1 | **78.1** | 58.8 | **42.3** | 66.0 | 51.5 |
| SN-L+CV[1] | **48.9** | 77.5 | **59.9** | 40.8 | **74.0** | **52.6** |

Table 1: Precision, recall and F1 results (%) for different models. The first two models are baselines. The next five models are different variants of our model.

**Experimental Result Analysis**. Table 1 shows the experimental results, from which we can observe that:

(1) RSN models outperform all baseline models on precision, recall, and F1-score, among which Weakly-supervised RSN (SN-L+CV) achieves state-of-the-art performances. This indicates that RSN is capable of understanding new relations' semantic meanings within sentences.

(2) Supervised and distantly-supervised relational representations improve clustering performances. Compared with RW-HAC, SN-HAC achieves better clustering results because of its supervised relational representation and similarity metric. Specifically, unsupervised baselines mainly use sparse one-hot features. RW-HAC uses word embeddings, but integrates them in a rule-based way. In contrast, RSN uses distributed feature representations, and can optimize information integration process according to supervision.

(3) Louvain outperforms HAC for clustering with RSN, comparing SN-HAC with SN-L. One explanation is that our model does not put additional constraints on the prior distribution of relational vectors, and therefore the relation clusters might have odd shapes in violation of HAC's assumption. Moreover, when representations are not distinguishable enough, forcing HAC to find fine-grained clusters may harm recall while contributing minimally to precision. In practice, we do observe that the number of relations SN-L extracts is constantly less than the true number 16.

(4) Both SN-L+V and SN-L+C improve the performance of supervised or distantly-supervised

[1]Here for FewRel-distant we use Equation 10 rather than Equation 9 as loss, which corresponds to Distantly-supervised RSN, and this brings a minor improvement on $F_1$ from 52.0% to 52.6%.

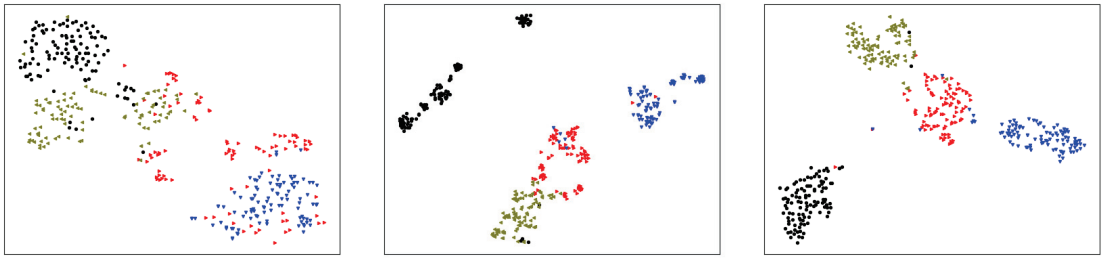

(a) RSN (b) Semi-supervised RSN (c) Supervised CNN

Figure 4: The t-SNE visualization of the output vectors of CNN modules in our (a) OpenRE model RSN, (b) Semi-supervised RSN facilitated by unlabeled novel-relation data and in (c) a classical RE baseline trained with labeled novel-relation data. All figures visualize the clustering result for 402 instances of 4 novel relations.

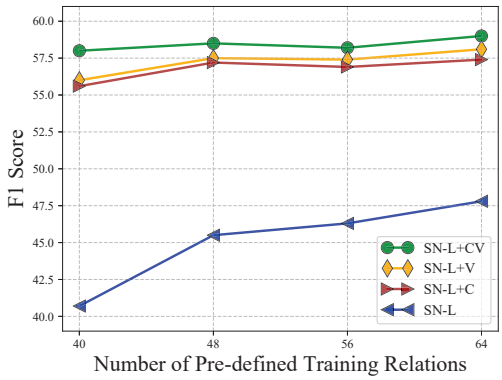


Figure 5: The clustering results with different numbers of pre-defined training relations.

RSN by further utilizing unsupervised corpora. Both semi-supervised approaches bring significant improvements for $F_1$ scores by increasing the precision and recall, and combining both can further increase the $F_1$ score.

(5) One interesting observation is that SN-L+V does not outperform SN-L so much on FewRel-distant. This is probably because VAT on the noisy data might amplify the noise. In further experiments, we perform VAT only on unlabeled set and observe improvements on $F_1$, with SN-L+V from $45.8\%$ to $49.2\%$ and SN-L+CV from $52.0\%$ to $52.6\%$, which proves this conjecture.

### 4.4 The Influence of Pre-defined Relation Diversity on Generalizability

In this subsection, we mainly focus on analyzing the influence of pre-defined relation diversity, i.e., the number of relations in the labeled train set. To study this influence, we use FewRel for evaluation and change the number of relations in the labeled train set from 40 to 64 while fixing the total number of labeled instances to $25,000$, and report the clustering results in Figure 5.

Several conclusions can be drawn according to Figure 5. Firstly, a rich variety of labeled relations do improve the performance of our models, especially RSN. The models trained on 64 relations perform better than those trained on 40 relations constantly. Secondly, while the performance of supervised RSN is very sensitive to pre-defined relation diversity, its semi-supervised counterparts suffer much less from the relation number limit. This phenomenon suggests that Semi-supervised RSNs succeed in learning from unlabeled novel-relation data and are more generalizable to novel relations.

### 4.5 Relational Knowledge Representation Visualization

To intuitively evaluate the knowledge transfer effects of RSN and Semi-supervised RSN, we visualize their relational knowledge representation spaces in the last layer of CNN encoders with t-SNE(Maaten and Hinton, 2008) in Figure 4. We also compare with a supervised CNN trained on $9,600$ labeled instances of novel relations, which suggests the optimal relational knowledge representation. In each figure, we plot 402 relation instances of 4 randomly-chosen relation types in the test set, and points are colored according to their ground-truth labels.

As we can see from Figure 4, RSN is able to roughly distinguish different relations, and Semi-supervised RSN further facilitated knowledge transfer by optimizing the margin between potential relation clusters during training. As a result, Semi-supervised RSN can extract more distinguishable novel relations, and gains comparable

relational knowledge representation ability with supervised CNN.

## 5 Conclusions and Future Work

In this paper, we propose a new model Relational Siamese Network (RSN) for OpenRE. Different from conventional unsupervised models, our model learns to measure relational similarity from supervised/distantly-supervised data of predefined relations, as well as unsupervised data of novel relations. There are mainly two innovative points in our model. First, we propose to transfer relational similarity knowledge with RSN structure. To the best of our knowledge, we are the first to propose knowledge transfer for OpenRE. Second, we propose Semi/Distantly-supervised RSN, to further perform semi-supervised and distantly-supervised transfer learning. Experiments show that our models significantly surpass conventional OpenRE models and achieve new state-of-the-art performance.

For future research, we plan to explore the following directions: (1) Besides CNN, there are some other popular sentence encoder structures like piecewise convolutional neural network (PCNN) and Long Short-Term Memory (LSTM) for RE. In the future, we can try different sentence encoders in our model. (2) As mentioned above, our model has the potential ability to discover the hierarchical structure of relations. In the future, we will try to explore this application with additional experiments.

## 6 Acknowledgement

This work is supported by the National Key Research and Development Program of China (No. 2018YFB1004503) and the National Natural Science Foundation of China (NSFC No. 61572273, 61661146007). Ruidong Wu is also supported by Tsinghua University Initiative Scientific Research Program.

## References

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *Proceedings of the First Iternational Conference on Language Resources and Evaluation Workshop on Linguistics Coreference*.

Eric Bair. 2013. Semi-supervised clustering methods. *Wiley Interdisciplinary Reviews: Computational Statistics*.

Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *Proceedings of IJCAI*.

Michele Banko and Oren Etzioni. 2008. The tradeoffs between open and traditional relation extraction. In *Proceedings of ACL-HLT*.

Vincent D Blondel, Jean Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics*.

Olivier Chapelle and Alexander Zien. 2005. Semi-supervised classification by low density separation. In *Proceedings of AISTATS*.

Lei Cui, Furu Wei, and Ming Zhou. 2018. Neural open information extraction. *arXiv*.

Hady Elsahar, Elena Demidova, Simon Gottschalk, Christophe Gravier, and Frederique Laforest. 2017. Unsupervised open relation extraction. In *Proceedings of European Semantic Web Conference*.

Yves Grandvalet and Yoshua Bengio. 2005. Semi-supervised learning by entropy minimization. In *Proceedings of NIPS*.

Xu Han, Hao Zhu, Pengfei Yu, Ziyun Wang, Yuan Yao, Zhiyuan Liu, and Maosong Sun. 2018. Fewrel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. In *Proceedings of EMNLP*.

Meng Qu Xiang Ren Hongtao Lin, Jun Yan. 2019. Learning dual retrieval module for semi-supervised relation extraction. In *Proceedings of WWW*.

Shengbin Jia, Yang Xiang, and Xiaojun Chen. 2018. Supervised neural models revitalize the open relation extraction. *arXiv*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv*.

Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. 2015. Siamese neural networks for one-shot image recognition. In *Proceedings of ICML Deep Learning Workshop*, volume 2.

Dekang Lin and Patrick Pantel. 2001. Dirt - discovery of inference rules from text. In *Proceedings of KDD*.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of ACL*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *JMLRJournal of Statistical Mechanics*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of ACL*.

Diego Marcheggiani and Ivan Titov. 2016. Discrete-state variational autoencoders for joint discovery and factorization of relations. *Transactions of ACL*.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of ACL-IJCNLP*.

Takeru Miyato, Andrew M Dai, and Ian Goodfellow. 2016. Adversarial training methods for semi-supervised text classification. *arXiv*.

Sachin Pawar, Girish K. Palshikar, and Pushpak Bhattacharyya. 2017. Relation extraction : A survey. *arXiv*.

Jeffrey Pennington, Richard Socher, and Christoper Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*.

Gabriel Stanovsky, Julian Michael, Luke Zettlemoyer, and Ido Dagan. 2018. Supervised open information extraction. In *Proceedings of NAACL*.

Joe H Ward Jr. 1963. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*.

Limin Yao, Aria Haghighi, Sebastian Riedel, and Andrew Mccallum. 2011. Structured relation discovery using generative models. In *Proceedings of EMNLP*.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2012. Unsupervised relation discovery with sense disambiguation. In *Proceedings of ACL*.

Dian Yu, Lifu Huang, and Heng Ji. 2017. Open relation extraction and grounding. In *Proceedings of IJCNLP*.

Jianbo Yuan, Han Guo, Zhiwei Jin, Hongxia Jin, Xianchao Zhang, and Jiebo Luo. 2017. One-shot learning for fine-grained relation extraction via convolutional siamese neural network. In *Proceedings of BigData*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, Jun Zhao, et al. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING*.

# Improving Relation Extraction with Knowledge-attention

Pengfei Li[1], Kezhi Mao[1]*, Xuefeng Yang[3], and Qi Li[2]

[1]School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore
[2]Interdisciplinary Graduate School, Nanyang Technological University, Singapore
[3]ZhuiYi Technology, Shenzhen, China
{pli006,ekzmao,liqi0024}@ntu.edu.sg, ryan@wezhuiyi.com

## Abstract

While attention mechanisms have been proven to be effective in many NLP tasks, majority of them are data-driven. We propose a novel knowledge-attention encoder which incorporates prior knowledge from external lexical resources into deep neural networks for relation extraction task. Furthermore, we present three effective ways of integrating knowledge-attention with self-attention to maximize the utilization of both knowledge and data. The proposed relation extraction system is end-to-end and fully attention-based. Experiment results show that the proposed knowledge-attention mechanism has complementary strengths with self-attention, and our integrated models outperform existing CNN, RNN, and self-attention based models. State-of-the-art performance is achieved on TACRED, a complex and large-scale relation extraction dataset.

## 1 Introduction

Relation extraction aims to detect the semantic relationship between two entities in a sentence. For example, given the sentence: "*James Dobson* has resigned as chairman of *Focus On The Family*, which he founded thirty years ago.", the goal is to recognize the organization-founder relation held between "*Focus On The Family*" and "*James Dobson*". The various relations between entities extracted from large-scale unstructured texts can be used for ontology and knowledge base population (Chen et al., 2018a; Fossati et al., 2018), as well as facilitating downstream tasks that requires relational understanding of texts such as question answering (Yu et al., 2017) and dialogue systems (Young et al., 2018).

Traditional feature-based and kernel-based approaches require extensive feature engineering (Suchanek et al., 2006; Qian et al., 2008; Rink and Harabagiu, 2010). Deep neural networks such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) have the ability of exploring more complex semantics and extracting features automatically from raw texts for relation extraction tasks (Xu et al., 2016; Vu et al., 2016; Lee et al., 2017). Recently, attention mechanisms have been introduced to deep neural networks to improve their performance (Zhou et al., 2016; Wang et al., 2016; Zhang et al., 2017). Especially, the Transformer proposed by Vaswani et al. (2017) is based solely on self-attention and has demonstrated better performance than traditional RNNs (Bilan and Roth, 2018; Verga et al., 2018). However, deep neural networks normally require sufficient labeled data to train their numerous model parameters. The scarcity or low quality of training data will limit the model's ability to recognize complex relations and also cause overfitting issue.

A recent study (Li and Mao, 2019) shows that incorporating prior knowledge from external lexical resources into deep neural network can reduce the reliance on training data and improve relation extraction performance. Motivated by this, we propose a novel knowledge-attention mechanism, which transforms texts from word semantic space into relational semantic space by attending to relation indicators that are useful in recognizing different relations. The relation indicators are automatically generated from lexical knowledge bases which represent keywords and cue phrases of different relation expressions. While the existing self-attention encoder learns internal semantic features by attending to the input texts themselves, the proposed knowledge-attention encoder captures the linguistic clues of different relations based on external knowledge. Since the two attention mechanisms complement each other, we integrate them into a single model to maximize the uti-

*Corresponding author.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 229–239,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

lization of both knowledge and data, and achieve optimal performance for relation extraction.

In summary, the main contributions of the paper are: (1) We propose knowledge-attention encoder, a novel attention mechanism which incorporates prior knowledge from external lexical resources to effectively capture the informative linguistic clues for relation extraction. (2) To take the advantages of both knowledge-attention and self-attention, we propose three integration strategies: multi-channel attention, softmax interpolation, and knowledge-informed self-attention. Our final models are fully attention-based and can be easily set up for end-to-end training. (3) We present detailed analysis on knowledge-attention encoder. Results show that it has complementary strengths with self-attention encoder, and the integrated models achieve start-of-the-art results for relation extraction.

## 2 Related Works

We focus here on deep neural networks for relation extraction since they have demonstrated better performance than traditional feature-based and kernel-based approaches.

Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) are the earliest and commonly used approaches for relation extraction. Zeng et al. (2014) showed that CNN with position embeddings is effective for relation extraction. Similarly, CNN with multiple filter sizes (Nguyen and Grishman, 2015), pairwise ranking loss function (dos Santos et al., 2015) and auxiliary embeddings (Lee et al., 2017) were proposed to improve performance. Zhang and Wang (2015) proposed bi-directional RNN with max pooling to model the sequential relations. Instead of modeling the whole sentence, performing RNN on sub-dependency trees (e.g. shortest dependency path between two entities) has demonstrated to be effective in capturing long-distance relation patterns (Xu et al., 2016; Miwa and Bansal, 2016). Zhang et al. (2018) proposed graph convolution over dependency trees and achieved state-of-the-art results on TACRED dataset.

Recently, attention mechanisms have been widely applied to CNNs (Wang et al., 2016; Han et al., 2018) and RNNs (Zhou et al., 2016; Zhang et al., 2017; Du et al., 2018). The improved performance demonstrated the effectiveness of attention mechanisms in deep neural networks. Particularly, Vaswani et al. (2017) proposed a solely self-attention-based model called Transformer, which is more effective than RNNs in capturing long-distance features since it is able to draw global dependencies without regard to their distances in the sequences. Bilan and Roth (2018) first applied self-attention encoder to relation extraction task and achieved competitive results on TACRED dataset. Verga et al. (2018) used self-attention to encode long contexts spanning multiple sentences for biological relation extraction. However, more attention heads and layers are required for self-attention encoder to capture complex semantic and syntactic information since learning is solely based on training data. Hence, more high quality training data and computational power are needed. Our work utilizes the knowledge from external lexical resources to improve deep neural network's ability of capturing informative linguistic clues.

External knowledge has shown to be effective in neural networks for many NLP tasks. Existing works focus on utilizing external knowledge to improve embedding representations (Chen et al., 2015; Liu et al., 2015; Sinoara et al., 2019), CNNs (Toutanova et al., 2015; Wang et al., 2017; Li and Mao, 2019), and RNNs (Ahn et al., 2016; Chen et al., 2016, 2018b; Shen et al., 2018). Our work is the first to incorporate knowledge into Transformer through a novel knowledge-attention mechanism to improve its performance on relation extraction task.

## 3 Knowledge-attention Encoder

We present the proposed knowledge-attention encoder in this section. Relation indicators are first generated from external lexical resources (Section 3.1); Then the input texts are transformed from word semantic space into relational semantic space by attending to the relation indicators using knowledge-attention mechanism (Section 3.2); Finally, position-aware attention is used to summarize the input sequence by taking both relation semantics and relative positions into consideration (Section 3.3).

### 3.1 Relation Indicators Generation

Relation indicators represent the keywords or cue phrases of various relation types, which are essential for knowledge-attention encoder to capture the linguistic clues of certain relation from texts.

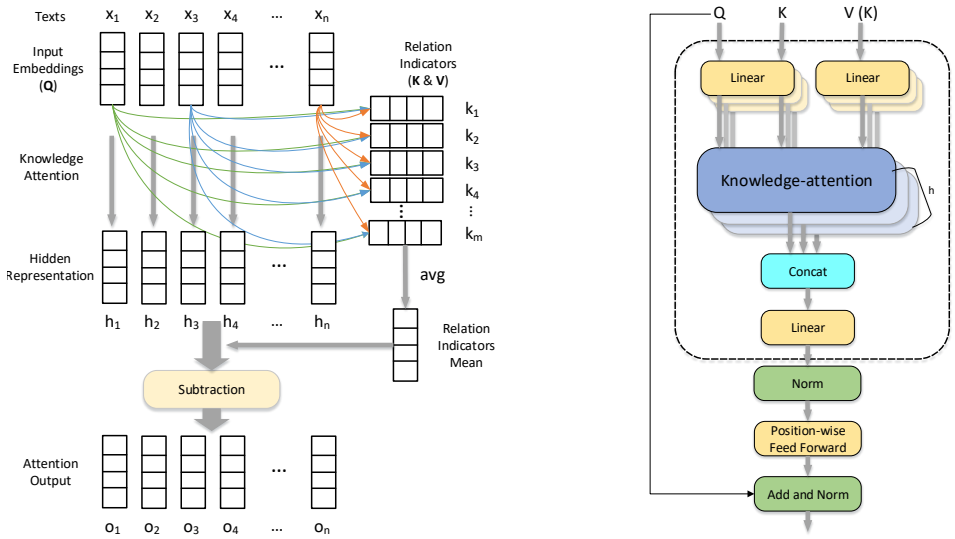


Figure 1: Knowledge-attention process (left) and multi-head structure (right) of knowledge-attention encoder.

We utilize two publicly available lexical resources including FrameNet[1] and Thesaurus.com[2] to find such lexical units.

FrameNet is a large lexical knowledge base which categorizes English words and sentences into higher level semantic frames (Ruppenhofer et al., 2006). Each frame is a conceptual structure describing a type of event, object or relation. FrameNet contains over 1200 semantic frames, many of which represent various semantic relations. For each relation type in our relation extraction task, we first find all the relevant semantic frames by searching from FrameNet (refer Appendix for detailed semantic frames used). Then we extract all the lexical units involved in these frames, which are exactly the keywords or phrases that often used to express such relation. Thesaurus.com is the largest online thesaurus which has over 3 million synonyms and antonyms. It also has the flexibility to filter search results by relevance, POS tag, word length, and complexity. To broaden the coverage of relation indicators, we utilize the synonyms in Thesaurus.com to extend the lexical units extracted from FrameNet. To reduce noise, only the most relevant synonyms with the same POS tag are selected.

Relation indicators are generated based on the word embeddings and POS tags of lexical units. Formally, given a word in a lexical unit, we find its word embedding $\mathbf{w}_i \in \mathbb{R}^{d_w}$ and POS embedding $\mathbf{t}_i \in \mathbb{R}^{d_t}$ by looking up the word embedding matrix $\mathbf{W}^{wrd} \in \mathbb{R}^{d_w \times V^{wrd}}$ and POS embedding matrix $\mathbf{W}^{pos} \in \mathbb{R}^{d_t \times V^{pos}}$ respectively, where $d_w$ and $d_t$ are the dimensions of word and POS embeddings, $V^{wrd}$ is vocabulary size[3] and $V^{pos}$ is total number of POS tags. The corresponding relation indicator is formed by concatenating word embedding and POS embedding, $\mathbf{k}_i = [\mathbf{w}_i, \mathbf{t}_i]$. If a lexical unit contains multiple words (i.e. phrase), the corresponding relation indicator is formed by averaging the embeddings of all words. Eventually, around 3000 relation indicators (including 2000 synonyms) are generated: $\mathbf{K} = \{\mathbf{k}_1, \mathbf{k}_2, ..., \mathbf{k}_m\}$.

## 3.2 Knowledge Attention

### 3.2.1 Knowledge-attention process

In a typical attention mechanism, a query ($q$) is compared with the keys ($K$) in a set of key-value pairs and the corresponding attention weights are calculated. The attention output is weighted sum of values ($V$) using the attention weights. In our proposed knowledge-attention encoder, the queries are input texts and the key-value pairs are both relation indicators. The detailed process of knowledge-attention is shown in Figure 1 (left).

Formally, given text input $x = \{x_1, x_2, ..., x_n\}$, the input embeddings $\mathbf{Q} = \{\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_n\}$ are generated by concatenating each word's word embedding and POS embedding in the same way as relation indicator generation in Section 3.1. The

[1] https://framenet.icsi.berkeley.edu/fndrupal

[2] https://www.thesaurus.com

[3] Same word embedding matrix is used for relation indicators and input texts, hence the vocabulary also includes all the words in the training corpus.

hidden representations $\mathbf{H} = \{\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_n\}$ are obtained by attending to the knowledge indicators $\mathbf{K}$, as shown in Equation 1. The final knowledge-attention outputs are obtained by subtracting the hidden representations with the relation indicators mean, as shown in Equation 2.

$$\mathbf{H} = softmax(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}})\mathbf{V} \quad (1)$$

$$knwl(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathbf{H} - \sum \mathbf{K}/m \quad (2)$$

where $knwl$ indicates knowledge-attention process, $m$ is the number of relation indicators, and $d_k$ is dimension of key/query vectors which is a scaling factor same as in Vaswani et al. (2017).

The subtraction of relation indicators mean will result in small outputs for irrelevant words. More importantly, the resulted output will be close to the related relation indicators and further apart from other relation indicators in relational semantic space. Therefore, the proposed knowledge-attention mechanism is effective in capturing the linguistic clues of relations represented by relation indicators in the relational semantic space.

#### 3.2.2 Multi-head knowledge-attention

Inspired by the multi-head attention in Transformer (Vaswani et al., 2017), we also have multi-head knowledge-attention which first linearly transforms $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ $h$ times, and then perform $h$ knowledge-attentions simultaneously, as shown in Figure 1 (right).

Different from the Transformer encoder, we use the same linear transformation for $\mathbf{Q}$ and $\mathbf{K}$ in each head to keep the correspondence between queries and keys.

$$head_i = knwl(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^Q, \mathbf{V}\mathbf{W}_i^V) \quad (3)$$

where $\mathbf{W}_i^Q, \mathbf{W}_i^V \in \mathbb{R}^{d_k \times (d_k/h)}$ and $i \in [1, 2, ...h]$. Besides, only one residual connection from input embeddings to outputs of position-wise feed forward networks is used. We also mask the outputs of padding tokens using zero vectors.

The multi-head structure in knowledge-attention allows the model to jointly attend inputs to different relational semantic subspaces with different contributions of relation indicators. This is beneficial in recognizing complex relations where various compositions of relation indicators are needed.

### 3.3 Position-aware Attention

It has been proven that the relative position information of each token with respective to the two target entities is beneficial for relation extraction task (Zeng et al., 2014). We modify the position-aware attention originally proposed by Zhang et al. (2017) to incorporate such relative position information and find the importance of each token to the final sentence representation.

Assume the relative position of token $x_i$ to target entity is $\hat{p}_i$. We apply position binning function (Equation 4) to make it easier for the model to distinguish long and short relative distances.

$$p_i = \begin{cases} \hat{p}_i & |\hat{p}_i| \leq 2 \\ \frac{\hat{p}_i}{|\hat{p}_i|} \lceil log_2 |\hat{p}_i| + 1 \rceil & |\hat{p}_i| > 2 \end{cases} \quad (4)$$

After getting the relative positions $p_i^s$ and $p_i^o$ to the two entities of interest (subject and object respectively), we map them to position embeddings base on a shared position embedding matrix $\mathbf{W}^p$. The two embeddings are concatenated to form the final position embedding for token $x_i$: $\mathbf{p}_i = [\mathbf{p}_i^s, \mathbf{p}_i^o]$.

Position-aware attention is performed on the outputs of knowledge-attention $\mathbf{O} \in \mathbb{R}^{n \times d_k}$, taking the corresponding relative position embeddings $\mathbf{P} \in \mathbb{R}^{n \times d_p}$ into consideration:

$$\mathbf{f} = O^T softmax(tanh(\mathbf{O}\mathbf{W}_o + \mathbf{P}\mathbf{W}_p)\mathbf{c}) \quad (5)$$

where $\mathbf{W}_o \in \mathbb{R}^{d_k \times d_a}$, $\mathbf{W}_p \in \mathbb{R}^{d_p \times d_a}$, $d_a$ is attention dimension, and $\mathbf{c} \in \mathbb{R}^{d_a}$ is a context vector learned by the neural network.

## 4 Integrate Knowledge-attention with Self-attention

The self-attention encoder proposed by Vaswani et al. (2017) learns internal semantic features by modeling pair-wise interactions within the texts themselves, which is effective in capturing long-distance dependencies. Our proposed knowledge-attention encoder has complementary strengths of capturing the linguistic clues of relations precisely based on external knowledge. Therefore, it is beneficial to integrate the two models to maximize the utilization of both external knowledge and training data. In this section, we propose three integration approaches as shown in Figure 2, and each approach has its own advantages.

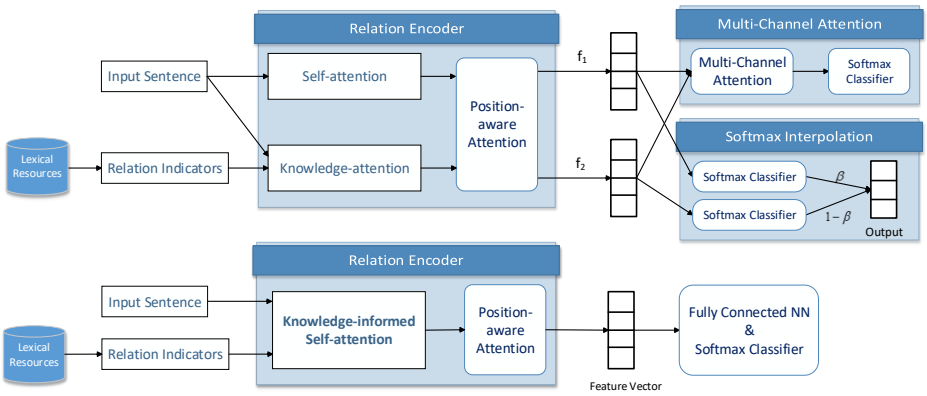


Figure 2: Three ways of integrating knowledge-attention with self-attention: multi-channel attention and softmax interpolation (top), as well as knowledge-informed self-attention (bottom).

### 4.1 Multi-channel Attention

In this approach, self-attention and knowledge-attention are treated as two separate channels to model sentence from different perspectives. After applying position-aware attention, two feature vectors $\mathbf{f}_1$ and $\mathbf{f}_2$ are obtained from self-attention and knowledge-attention respectively. We apply another attention mechanism called multi-channel attention to integrate the feature vectors.

In multi-channel attention, feature vectors are first fed into a fully connected neural network to get their hidden representations $\mathbf{h}_i$. Then attention weights are calculated using a learnable context vector $\mathbf{c}$, which reflects the importance of each feature vector to final relation classification. Finally, the feature vectors are integrated based on attention weights, as shown in Equation 6.

$$\mathbf{r} = \sum_i softmax(\mathbf{h}_i^T \mathbf{c})\mathbf{h}_i \tag{6}$$

After obtaining the integrated feature vector $\mathbf{r}$, we pass it to a softmax classifier to determine the relation class. The model is trained using stochastic gradient descent with momentum and learning rate decay to minimize the cross-entropy loss.

The main advantage of this approach is flexibility. Since the two channels process information independently, the input components are not necessary to be the same. Besides, we can add more features from other sources (e.g. subject and object categories) to multi-channel attention to make final decision based on all the information sources.

### 4.2 Softmax Interpolation

Similar as multi-channel attention, we also use two independent channels for self-attention and knowledge-attention in softmax interpolation. Instead of integrating the feature vectors, we make two independent predictions using two softmax classifiers based on the feature vectors from the two channels. The loss function is defined as total cross-entropy loss of the two classifiers. The final prediction is obtained using an interpolation function of the two softmax distributions:

$$\mathbf{p} = \beta \cdot \mathbf{p}_1 + (1 - \beta) \cdot \mathbf{p}_2 \tag{7}$$

where $\mathbf{p}_1$, $\mathbf{p}_2$ are the softmax distributions obtained form self-attention and knowledge-attention respectively, and $\beta$ is the priority weight assigned to self-attention.

Since knowledge-attention focuses on capturing the keywords and cue phrases of relations, the precision will be higher than self-attention while the recall is lower. The proposed softmax interpolation approach is able to take the advantages of both attention mechanisms and balance the precision and recall by adjusting the priority weight $\beta$.

### 4.3 Knowledge-informed Self-attention

Since knowledge-attention and self-attention share similar structures, it is also possible to integrate them into a single channel. We propose knowledge-informed self-attention encoder which incorporates knowledge-attention into every self-attention head to jointly model the semantic relations based on both knowledge and data.

The structure of knowledge-informed self-attention is shown in Figure 3. Formally, given texts input matrix $\mathbf{Q} \in \mathbb{R}^{n \times d_k}$ and knowledge indicators $\mathbf{K} \in \mathbb{R}^{m \times d_k}$. The output of each attention head is calculated as follows:

$$\begin{aligned} head_i = \; & knwl(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^V) + \\ & self(\mathbf{Q}\mathbf{W}_i^{Q_s}, \mathbf{Q}\mathbf{W}_i^{K_s}, \mathbf{Q}\mathbf{W}_i^{V_s}) \end{aligned} \tag{8}$$

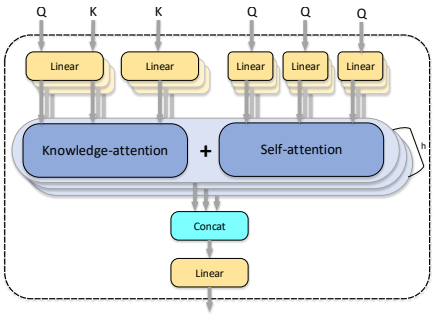


Figure 3: Knowledge-informed self-attention structure. Q, K represent input matrix and knowledge indicators respectively, h is the number of attention heads.

where $knwl$ and $self$ indicate knowledge-attention and self-attention respectively, and all the linear transformation weight matrices have the dimensionality of $\mathbf{W} \in \mathbb{R}^{d_k \times (d_k/h)}$.

Since each self-attention head is aided with prior knowledge in knowledge-attention, the knowledge-informed self-attention encoder is able to capture more lexical and semantic information than single attention encoder.

## 5 Experiment and Analysis

### 5.1 Baseline Models

To study the performance of our proposed models, the following baseline models are used for comparison:

**CNN-based models** including: (1) CNN: the classical convolutional neural network for sentence classification (Kim, 2014). (2) CNN-PE: CNN with position embeddings dedicated for relation classification (Nguyen and Grishman, 2015). (3) GCN: a graph convolutional network over the pruned dependency trees of the sentence (Zhang et al., 2018).

**RNN-based models** including: (1) LSTM: long short-term memory network to sequentially model the texts. Classification is based on the last hidden output. (2) PA-LSTM: Similar position-aware attention mechanism as our work is used to summarize the LSTM outputs (Zhang et al., 2017).

**CNN-RNN hybrid model** including contextualized GCN (C-GCN) where the input vectors are obtained using bi-directional LSTM network (Zhang et al., 2018).

**Self-attention-based model** (Self-attn) which uses self-attention encoder to model the input sentence. Our implementation is based on Bilan and Roth (2018) where several modifications are made on the original Transformer encoder, including the use of relative positional encodings instead of absolute sinusoidal encodings, as well as other configurations such as residual connection, activation function and normalization.

For our model, we evaluate both the proposed knowledge-attention encoder (Knwl-attn) as well as the integrated models with self-attention including multi-channel attention (MCA), softmax interpolation (SI) and knowledge-informed self-attention (KISA).

### 5.2 Experiment Settings

We conduct our main experiments on TACRED, a large-scale relation extraction dataset introduced by Zhang et al. (2017). TACRED contains over 106k sentences with hand-annotated subject and object entities as well as the relations between them. It is a very complex relation extraction dataset with 41 relation types and a *no_relation* class when no relation is hold between entities. The dataset is suited for real-word relation extraction since it is unbalanced with 79.5% *no_relation* samples, and multiple relations between different entity pairs can be exist in one sentence. Besides, the samples are normally long sentences with an average of 36.2 words.

Since the dataset is already partitioned into train (68124 samples), dev (22631 samples) and test (15509 samples) sets, we tune model hyperparameters using dev set and evaluate model using test set. The evaluation metrics are micro-averaged precision, recall and $F_1$ score. For fair comparison, we select the model with median $F_1$ score on dev set from 5 independent runs, same as Zhang et al. (2017). The same "entity mask" strategy is used which replaces subject (or object) entity with special $\langle$NER$\rangle$-SUBJ (or $\langle$NER$\rangle$-OBJ) tokens to avoid overfittting on specific entities and provide entity type information.

Besides TACRED, another dataset called SemEval2010-Task8 (Hendrickx et al., 2009) is used to evaluate the generalization ability of our proposed model. The dataset is significantly smaller and simpler than TACRED, which has 8000 training samples and 2717 testing samples. It contains 9 directed relations and 1 other relation (19 relation classes in total). We use the official macro-averaged $F_1$ score as evaluation metric.

We use one layer encoder with 6 attention heads for both knowledge-attention and self-attention

since further increasing the number of layers and attention heads will degrade the performance. For softmax interpolation, we choose $\beta = 0.8$ to balance precision and recall. Word embeddings are fine-tuned based on pre-trained GloVe (Pennington et al., 2014) with dimensionality of 300. Dropout (Srivastava et al., 2014) is used during trianing to alleviate overfitting. Other model hyperparameters and training details are described in Appendix due to space limitations.

### 5.3 Results and Analysis

#### 5.3.1 Results on TACRED dataset

Table 1 shows the results of baseline as well as our proposed models on TACRED dataset. It is observed that our proposed knowledge-attention encoder outperforms all CNN-based and RNN-based models by at least 1.3 $F_1$. Meanwhile, it achieves comparable results with C-GCN and self-attention encoder, which are the current start-of-the-art single-model systems.

Comparing with self-attention encoder, it is observed that knowledge-attention encoder results in higher precision but lower recall. This is reasonable since knowledge-attention encoder focuses on capturing the significant linguistic clues of relations based on external knowledge, it will result in high precision for the predicted relations similar to rule-based systems. Self-attention encoder is able to capture more long-distance dependency features by learning from data, resulting in better recall. By integrating self-attention and knowledge-attention using the proposed approaches, a more balanced precision and recall can be obtained, suggesting the complementary effects of self-attention and knowledge-attention mechanisms. The integrated models improve performance by at least 0.9 $F_1$ score and achieve new state-of-the-art results among all the single end-to-end models.

Comparing the three integrated models, softmax interpolation (SI) achieves the best performance. More interestingly, we found that the precision and recall can be controlled by adjusting the priority weight $\beta$. Figure 4 shows impact of $\beta$ on precision, recall and $F_1$ score. As $\beta$ increases, precision decreases and recall increases. Therefore, we can choose a small $\beta$ for relation extraction system which requires high precision, and a large $\beta$ for the system requiring better recall. $F_1$ score reaches the highest value when precision and recall are balanced ($\beta = 0.8$).

| Model | P | R | $F_1$ |
|---|---|---|---|
| CNN† | **72.1** | 50.3 | 59.2 |
| CNN-PE† | 68.2 | 55.4 | 61.1 |
| GCN‡ | 69.8 | 59.0 | 64.0 |
| LSTM† | 61.4 | 61.7 | 61.5 |
| PA-LSTM† | 65.7 | 64.5 | 65.1 |
| C-GCN‡ | 69.9 | 63.3 | 66.4 |
| Self-attn†† | 64.6 | **68.6** | 66.5 |
| Knwl-attn | 70.0 | 63.1 | 66.4 |
| Knwl+Self (MCA) | 68.4 | 66.1 | 67.3* |
| Knwl+Self (SI) | 67.1 | 68.4 | **67.8*** |
| Know+Self (KISA) | 69.4 | 66.0 | 67.7* |

Table 1: Micro-averaged precision (P), recall (R) and $F_1$ score on TACRED dataset. †, ‡ and †† mark the results reported in (Zhang et al., 2017), (Zhang et al., 2018) and (Bilan and Roth, 2018) respectively. $*$ marks statistically significant improvements over Self-attn with $p < 0.01$ under one-tailed t-test.

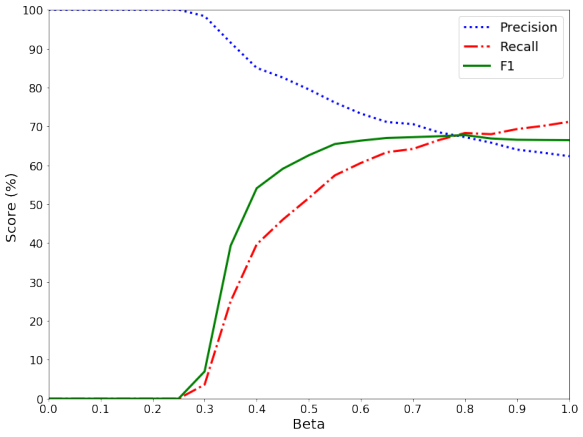


Figure 4: Change of precision, recall and $F_1$ score on dev set as the priority weight $\beta$ in softmax interpolation changes.

Knowledge-informed self-attention (KISA) has comparable performance with softmax interpolation, and without the need of hyper-parameter tuning since knowledge-attention and self-attention are integrated into a single channel. The performance gain over self-attention encoder is 1.2 $F_1$ with much improved precision, demonstrating the effectiveness of incorporating knowledge-attention into self-attention to jointly model the sentence based on both knowledge and data.

Performance gain is the lowest for multi-channel attention (MCA). However, the model is more flexible in the way that features from other information sources can be easily added to the

| Model | P | R | $F_1$ |
|---|---|---|---|
| MCA | 68.4 | 66.1 | 67.3 |
| + NER | 68.7 | **66.2** | 67.5 |
| + Entity category | 70.0 | 65.8 | 67.8 |
| + Both | **70.1** | 66.0 | **67.9** |

Table 2: Results of adding NER embeddings and entity categorical embeddings to the multi-channel attention (MCA) integrated model.

model to further improve its performance. Table 2 shows the results of adding NER embeddings of each token to self-attention channel, and entity (subject and object) categorical embeddings to multi-channel attention as additional feature vectors. We use dimensionality of 30 and 60 for NER and entity categorical embeddings respectively, and the two embedding matrixes are learned by the neural network. Results show that adding NER and entity categorical information to MCA integrated model improves $F_1$ score by 0.2 and 0.5 respectively, and adding both improves precision significantly, resulting a new best $F_1$ score.

#### 5.3.2 Results on SemEval2010-Task8 dataset

We use SemEval2010-Task8 dataset to evaluate the generalization ability of our proposed model. Experiments are conducted in two manners: mask or keep the entities of interest. Results in Table 3 show that the "entity mask" strategy degrades the performance, indicating that there exist strong correlations between entities of interest and relation classes in SemEval2010-Task8 dataset. Although the results of keeping the entities are better, the model tends to remember these entities instead of focusing on learning the linguistic clues of relations. This will result in bad generalization for sentences with unseen entities.

Regardless of whether the entity mask is used, by incorporating knowledge-attention mechanism, our model improves the performance of self-attention by a statistically significant margin, especially the softmax interpolation integrated model. The results on SemEval2010-Task8 are consistent with that of TACRED, demonstrating the effectiveness and robustness of our proposed method.

#### 5.3.3 Ablation study

To study the contributions of specific components of knowledge-attention encoder, we perform ablation experiments on the dev set of TACRED. The results of knowledge-attention encoder with and without certain components are shown in Table 4.

| Model | mask entity | keep entity |
|---|---|---|
| Self-attn | 76.8 | 83.1 |
| Knwl-attn | 76.1 | 82.3 |
| Knwl+Self (MCA) | 77.4* | 84.0* |
| Knwl+Self (SI) | **78.0*** | **84.3*** |
| Know+Self (KISA) | 77.5* | 84.0* |

Table 3: Macro-averaged $F_1$ score on SemEval2010-Task8 dataset. $*$ marks statistically significant improvements over Self-attn with $p < 0.01$ under one-tailed t-test.

| Model | Dev $F_1$ |
|---|---|
| Knwl-attn Encoder | 66.5 |
| 1. − Multi-head structure | 64.6 |
| 2. − Synonym relation indicators | 64.7 |
| 3. − Relation indicators mean | 65.0 |
| 4. − Output masking | 65.8 |
| 5. − Entity masking | 65.4 |
| 6. − Relative positions | 63.0 |

Table 4: Ablation study on knowledge-attention encoder. Results are the median $F_1$ scores of 5 independent runs on dev set of TACRED.

It is observed that: (1) The proposed multi-head knowledge-attention structure outperforms single-head significantly. This demonstrates the effectiveness of jointly attending texts to different relational semantic subspaces in the multi-head structure. (2) The synonyms improve the performance of knowledge-attention since they are able to broaden the coverage of relation indicators and form a robust relational semantic space. (3) The subtraction of relation indicators mean vector from attention hidden representations helps to suppress the activation of irrelevant words and results in a better representation for each word to capture the linguistic clues of relations. (4-5) The two masking strategies are helpful for our model: the output masking eliminates the effects of the padding tokens and the entity masking avoids entity overfitting while providing entity type information. (6) The relative position embedding term in position-aware attention contributes a significant amount of $F_1$ score. This shows that positional information is particularly important for relation extraction task.

#### 5.3.4 Attention visualization

To verify the complementary effects of knowledge-attention encoder and self-attention

| Sample Sentences | True Relation | Predict |
|---|---|---|
| SUBJ-PERSON graduated in 1992 from the OBJ-ORGANIZATION OBJ-ORGANIZATION OBJ-ORGANIZATION with a degree in computer science and had worked as a systems analyst at a Pittsburgh law firm since 1999 . | per:schools_attended | correct |
| SUBJ-PERSON graduated in 1992 from the OBJ-ORGANIZATION OBJ-ORGANIZATION OBJ-ORGANIZATION with a degree in computer science and had worked as a systems analyst at a Pittsburgh law firm since 1999 . | | correct |
| OBJ-PERSON OBJ-PERSON , a public affairs and government relations strategist , was executive director of the SUBJ-ORGANIZATION SUBJ-ORGANIZATION Policy Institute from 2005 to 2010 . | org:top_members/employees | correct |
| OBJ-PERSON OBJ-PERSON , a public affairs and government relations strategist , was executive director of the SUBJ-ORGANIZATION SUBJ-ORGANIZATION Policy Institute from 2005 to 2010 . | | wrong |
| Founded in 1992 in Schaumburg , Illinois , the SUBJ-ORGANIZATION is one of the largest Chinese - American associations of professionals in the OBJ-COUNTRY OBJ-COUNTRY . | org:country_of_headquarters | wrong |
| Founded in 1992 in Schaumburg , Illinois , the SUBJ-ORGANIZATION is one of the largest Chinese - American associations of professionals in the OBJ-COUNTRY OBJ-COUNTRY . | | correct |

Table 5: Attention visualization for knowledge-attention encoder (first) and self-attention encoder (second). Words are highlighted based on the attention weights assigned to them. Best viewed in color.

encoder, we compare the attention weights assigned to words from the two encoders. Table 5 presents the attention visualization results on sample sentences. For each sample sentence, attention weights from knowledge-attention encoder are visualized first, followed by self-attention encoder. It is observed that knowledge-attention encoder focuses more on the specific keywords or cue phrases of certain relations, such as "graduated", "executive director" and "founded"; while self-attention encoder attends to a wide range of words in the sentence and pays more attention to the surrounding words of target entities especially the words indicating the syntactic structure, such as "is", "in" and "of". Therefore, knowledge-attention encoder and self-attention encoder have complementary strengths that focus on different perspectives for relation extraction.

#### 5.3.5 Error analysis

To investigate the limitations of our proposed model and provide insights for future research, we analyze the errors produced by the system on the test set of TACRED. For knowledge-attention encoder, 58% errors are false negative (FN) due to the limited ability in capturing long-distance dependencies and some unseen linguistic clues during training. For our integrated model[4] that takes the benefits of both self-attention and knowledge-attention, FN is reduced by 10%. However, false positive (FP) is not improved due to overfitting that leads to wrong predictions. Many errors are caused by multiple entities with different relations co-occurred in one sentence. Our model may mistake irrelevant entities as a relation pair. We also observed that many FP errors are due to the confusions between related relations such as "city_of_death"and "city_of_residence". More data or knowledge is needed to distinguish "death" and "residence". Besides, some errors are caused by imperfect annotations.

## 6 Conclusion and Future Work

We introduce knowledge-attention encoder which effectively incorporates prior knowledge from external lexical resources for relation extraction. The proposed knowledge-attention mechanism transforms texts from word space into relational semantic space and captures the informative linguistic clues of relations effectively. Furthermore, we show the complementary strengths of knowledge-attention and self-attention, and propose three different ways of integrating them to maximize the utilization of both knowledge and data. The proposed models are fully attention-based end-to-end systems and achieve state-of-the-art results on TACRED dataset, outperforming existing CNN, RNN, and self-attention based models.

In future work, besides lexical knowledge, we will incorporate conceptual knowledge from encyclopedic knowledge bases into knowledge-attention encoder to capture the high-level semantics of texts. We will also apply knowledge-attention in other tasks such as text classification, sentiment analysis and question answering.

[4]We observed similar error behaviors of the three proposed integrated models.

## References


Sungjin Ahn, Heeyoul Choi, Tanel Pärnamaa, and Yoshua Bengio. 2016. A neural knowledge language model. *arXiv preprint arXiv:1608.00318*.

Ivan Bilan and Benjamin Roth. 2018. Position-aware self-attention with relative positional encodings for slot filling. *arXiv preprint arXiv:1807.03052*.

Muhao Chen, Yingtao Tian, Xuelu Chen, Zijun Xue, and Carlo Zaniolo. 2018a. On2vec: Embedding-based relation prediction for ontology population. In *Proceedings of the 2018 SIAM International Conference on Data Mining*, pages 315–323. SIAM.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Diana Inkpen, and Si Wei. 2018b. Neural natural language inference models enhanced with external knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2406–2417.

Yun-Nung Chen, Dilek Hakkani-Tur, Gokhan Tur, Asli Celikyilmaz, Jianfeng Gao, and Li Deng. 2016. Knowledge as a teacher: Knowledge-guided structural attention networks. *arXiv preprint arXiv:1609.03286*.

Zhigang Chen, Wei Lin, Qian Chen, Xiaoping Chen, Si Wei, Hui Jiang, and Xiaodan Zhu. 2015. Revisiting word embedding for contrasting meaning. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 106–115.

Jinhua Du, Jingguang Han, Andy Way, and Dadong Wan. 2018. Multi-level structured self-attentions for distantly supervised relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2216–2225.

Marco Fossati, Emilio Dorigatti, and Claudio Giuliano. 2018. N-ary relation extraction for simultaneous t-box and a-box knowledge base augmentation. *Semantic Web*, 9(4):413–439.

Xu Han, Pengfei Yu, Zhiyuan Liu, Maosong Sun, and Peng Li. 2018. Hierarchical relation extraction with coarse-to-fine grained attention. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2236–2245.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2009. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, pages 94–99. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

Ji Young Lee, Franck Dernoncourt, and Peter Szolovits. 2017. Mit at semeval-2017 task 10: Relation extraction with convolutional neural networks. *arXiv preprint arXiv:1704.01523*.

Pengfei Li and Kezhi Mao. 2019. Knowledge-oriented convolutional neural network for causal relation extraction from natural language texts. *Expert Systems with Applications*, 115:512–523.

Quan Liu, Hui Jiang, Si Wei, Zhen-Hua Ling, and Yu Hu. 2015. Learning semantic word embeddings based on ordinal knowledge constraints. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1501–1511.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1105–1116.

Thien Huu Nguyen and Ralph Grishman. 2015. Relation extraction: Perspective from convolutional neural networks. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 39–48.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Longhua Qian, Guodong Zhou, Fang Kong, Qiaoming Zhu, and Peide Qian. 2008. Exploiting constituent dependencies for tree kernel-based semantic relation extraction. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 697–704. Association for Computational Linguistics.

Bryan Rink and Sanda Harabagiu. 2010. Utd: Classifying semantic relations by combining lexical and semantic resources. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 256–259. Association for Computational Linguistics.

Josef Ruppenhofer, Michael Ellsworth, Myriam Schwarzer-Petruck, Christopher R Johnson, and Jan Scheffczyk. 2006. Framenet ii: Extended theory and practice.

Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proceedings of the*

*53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 626–634.

Ying Shen, Yang Deng, Min Yang, Yaliang Li, Nan Du, Wei Fan, and Kai Lei. 2018. Knowledge-aware attentive neural network for ranking question answer pairs. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 901–904. ACM.

Roberta A Sinoara, Jose Camacho-Collados, Rafael G Rossi, Roberto Navigli, and Solange O Rezende. 2019. Knowledge-enhanced document embeddings for text classification. *Knowledge-Based Systems*, 163:955–971.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Fabian M Suchanek, Georgiana Ifrim, and Gerhard Weikum. 2006. Combining linguistic and statistical analysis to extract relations from web documents. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 712–717. ACM.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Patrick Verga, Emma Strubell, and Andrew McCallum. 2018. Simultaneously self-attending to all mentions for full-abstract biological relation extraction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 872–884.

Ngoc Thang Vu, Heike Adel, Pankaj Gupta, et al. 2016. Combining recurrent and convolutional neural networks for relation classification. In *Proceedings of NAACL-HLT*, pages 534–539.

Jin Wang, Zhongyuan Wang, Dawei Zhang, and Jun Yan. 2017. Combining knowledge with deep convolutional neural networks for short text classification. In *IJCAI*, pages 2915–2921.

Linlin Wang, Zhu Cao, Gerard De Melo, and Zhiyuan Liu. 2016. Relation classification via multi-level attention cnns.

Yan Xu, Ran Jia, Lili Mou, Ge Li, Yunchuan Chen, Yangyang Lu, and Zhi Jin. 2016. Improved relation classification by deep recurrent neural networks with data augmentation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1461–1470.

Tom Young, Erik Cambria, Iti Chaturvedi, Hao Zhou, Subham Biswas, and Minlie Huang. 2018. Augmenting end-to-end dialogue systems with commonsense knowledge. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Mo Yu, Wenpeng Yin, Kazi Saidul Hasan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2017. Improved neural relation detection for knowledge base question answering. *arXiv preprint arXiv:1704.06194*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, Jun Zhao, et al. 2014. Relation classification via convolutional deep neural network.

Dongxu Zhang and Dong Wang. 2015. Relation classification via recurrent neural network. *arXiv preprint arXiv:1508.01006*.

Yuhao Zhang, Peng Qi, and Christopher D Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2205–2215.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D Manning. 2017. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 35–45.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 207–212.

# Jointly Learning Entity and Relation Representations for Entity Alignment

**Yuting Wu**[1], **Xiao Liu**[1], **Yansong Feng**[1,2]*, **Zheng Wang**[3] and **Dongyan Zhao**[1,2]
[1]Wangxuan Institute of Computer Technology, Peking University, China
[2]The MOE Key Laboratory of Computational Linguistics, Peking University, China
[3]School of Computing, University of Leeds, U.K.
{wyting,lxlisa,fengyansong,zhaodongyan}@pku.edu.cn
z.wang5@leeds.ac.uk

## Abstract

Entity alignment is a viable means for integrating heterogeneous knowledge among different knowledge graphs (KGs). Recent developments in the field often take an embedding-based approach to model the structural information of KGs so that entity alignment can be easily performed in the embedding space. However, most existing works do not explicitly utilize useful relation representations to assist in entity alignment, which, as we will show in the paper, is a simple yet effective way for improving entity alignment. This paper presents a novel joint learning framework for entity alignment. At the core of our approach is a Graph Convolutional Network (GCN) based framework for learning both entity and relation representations. Rather than relying on pre-aligned relation seeds to learn relation representations, we first approximate them using entity embeddings learned by the GCN. We then incorporate the relation approximation into entities to iteratively learn better representations for both. Experiments performed on three real-world cross-lingual datasets show that our approach substantially outperforms state-of-the-art entity alignment methods.

## 1 Introduction

Knowledge graphs (KGs) transform unstructured knowledge into simple and clear triples of <*head entity, relation, tail entity*> for rapid response and reasoning of knowledge. They are an effective way for supporting various NLP-enabled tasks like machine reading (Yang and Mitchell, 2017), information extraction (Wang et al., 2018a), and question-answering (Zhang et al., 2018b).

Even though many KGs originate from the same resource, e.g., Wikipedia, they are usually created independently. Therefore, different KGs often use different expressions and surface forms to indicate equivalent entities and relations – let alone those built from different resources or languages. This common problem of heterogeneity makes it difficult to integrate knowledge among different KGs. A powerful technique to address this issue is *Entity Alignment*, the task of linking entities with the same real-world identity from different KGs.

Classical methods for entity alignment typically involve a labor-intensive and time-consuming process of feature construction (Mahdisoltani et al., 2013) or rely on external information constructed by others (Suchanek et al., 2011). Recently, efforts have been devoted to the so-called embedding-based approaches. Representative works of this direction include JE (Hao et al., 2016), MTransE (Chen et al., 2017), JAPE (Sun et al., 2017), IPTransE (Zhu et al., 2017), and BootEA (Sun et al., 2018). More recent work (Wang et al., 2018b) uses the Graph Convolutional Network (GCN) (Kipf and Welling, 2017) to jointly embed multiple KGs.

Most of the recent works (e.g., JE, MTransE, JAPE, IPTransE and BootEA) rely on the translation-based models, such as TransE (Bordes et al., 2013), which enable these approaches to encode both entities and relations of KGs. These methods often put more emphasis on the entity embeddings, but do not explicitly utilize relation embeddings to help with entity alignment. Another drawback of such approaches is that they usually rely on pre-aligned relations (JAPE and IPTransE) or triples (MTransE). This limits the scale at which the model can be effectively performed due to the overhead for constructing seed alignments for large KGs. Alternative methods like GCN-based models, unfortunately, cannot directly obtain relation representations, leaving much room for improvement.

Recent studies have shown that jointly modeling entities and relations in a single framework can

* Corresponding author.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 240–249, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

improve tasks like information extraction (Miwa and Bansal, 2016; Bekoulis et al., 2018). We hypothesize that this will be the case for entity alignment too; that is, the rich relation information could be useful for improving entity alignment as entities and their relations are usually closely related. Our experiments show that this is even a conservative target: by jointly learning entity and relation representations, we can promote the results of both entity and relation alignment.

In this work, we aim to build a learning framework that jointly learns entity and relation representations for entity alignment; and we want to achieve this with only a small set of pre-aligned entities but not relations. Doing so will allow us to utilize relation information to improve entity alignment without paying extra cost for constructing seed relation alignments.

Our work is enabled by the recent breakthrough effectiveness of GCNs (Kipf and Welling, 2017) in extracting useful representations from graph structures. Although GCNs provide a good starting point, applying it to develop a practical and efficient framework to accurately capture relation information across KGs is not trivial. Because a vanilla GCN operates on the undirected and unlabeled graphs, a GCN-based model like (Wang et al., 2018b) would ignore the useful relation information of KGs. While the Relational Graph Convolutional Network (R-GCN) (Schlichtkrull et al., 2018) can model multi-relational graphs, existing R-GCNs use a weight matrix for each relation. This means that an R-GCN would require an excessive set of parameters to model thousands of relations in a typical real-world KG, making it difficult to learn an effective model on large KGs.

A key challenge of our joint learning framework is how to generate useful relation representations at the absence of seed relation alignments, and to ensure the framework can scale to a large number of types of relations. We achieve this by first approximating the relation representations using entity embeddings learned through a small amount of seed entity alignments. We go further by constructing a new joint entity representation consisting of both relation information and neighboring structural information of an entity. The joint representations allow us to iteratively improve the model's capability of generating better entity and relation representations, which lead to not only better entity alignment, but also more accurate relation alignment as a by-product.

We evaluate our approach by applying it to three real-world datasets. Experimental results show that our approach delivers better and more robust results when compared with state-of-the-art methods for entity and relation alignments. The key contribution of this paper is a novel joint learning model for entity and relation alignments. Our approach reduces the human involvement and the associated cost in constructing seed alignments, but yields better performance over prior works.

## 2 Related Work

### 2.1 Entity Alignment

Until recently, entity alignment would require intensive human participation (Vrandečić and Krötzsch, 2014) to design hand-crafted features (Mahdisoltani et al., 2013), rules, or rely on external sources (Wang et al., 2017). In a broader context, works in schema and ontology matching also seek help from additional information by using e.g., extra data sources (Nguyen et al., 2011), entity descriptions (Lacoste-Julien et al., 2013; Yang et al., 2015), or semantics of the web ontology language (Hu et al., 2011). Performance of such schemes is bounded by the quality and availability of the extra information about the target KG, but obtaining sufficiently good-quality annotated data could be difficult for large KGs.

Recently, embedding-based entity alignment methods were proposed to reduce human involvement. JE (Hao et al., 2016) was among the first attempts in this direction. It learns embeddings of different KGs in a uniform vector space where entity alignment can be performed. MTransE (Chen et al., 2017) encodes KGs in independent embeddings and learns transformation between KGs. BootEA (Sun et al., 2018) exploits a bootstrapping process to learn KG embeddings. SEA (Pei et al., 2019) proposes a degree-aware KG embedding model to embed KGs. KDCoE (Chen et al., 2018) is a semi-supervised learning approach for co-training embeddings for multilingual KGs and entity descriptions. They all use translation-based models as the backbone to embed KGs.

Non-translational embedding-based methods include recent works on a GCN-based model (Wang et al., 2018b) and NTAM (Li et al., 2018). Additionally, most recent work, RDGCN (Wu et al., 2019), introduces the dual relation graph to model the relation information of KGs. Through

multiple rounds of interactions between the primal and dual graphs, RDGCN can effectively incorporate more complex relation information into entity representations and achieve promising results for entity alignment. However, existing methods only focus on entity embeddings and ignore the help that relation representations can provide on this task.

MTransE and NTAM are two of a few methods that try to perform both relation and entity alignments. However, both approaches require high-quality seed alignments, such as pre-aligned triples or relations, for relation alignment. Our approach advances prior works by jointly modeling entities and relations by using only a *small set* of pre-aligned entities (*but not relations*) to simultaneously perform entity and relation alignments.

### 2.2 Graph Convolutional Networks

GCNs (Duvenaud et al., 2015; Kearnes et al., 2016; Kipf and Welling, 2017) are neural networks operating on unlabeled graphs and inducing features of nodes based on the structures of their neighborhoods. Recently, GCNs have demonstrated promising performance in tasks like node classification (Kipf and Welling, 2017), relation extraction (Zhang et al., 2018a), semantic role labeling (Marcheggiani and Titov, 2017), etc. As an extension of GCNs, the R-GCNs (Schlichtkrull et al., 2018) have recently been proposed to model relational data for link prediction and entity classification. However, R-GCNs usually require a large number of parameters that are often hard to train, when applied to multi-relational graphs.

In this work, we choose to use GCNs to first encode KG entities and to approximate relation representations based on entity embeddings. Our work is the first to utilize GCNs for jointly aligning entities and relations for heterogeneous KGs.

## 3 Problem Formulation

We now introduce the notations used in this paper and define the scope of this work.

A KG is formalized as $G = (E, R, T)$, where $E, R, T$ are the sets of entities, relations and triples, respectively. Let $G_1 = (E_1, R_1, T_1)$ and $G_2 = (E_2, R_2, T_2)$ be two different KGs. Usually, some equivalent entities between KGs are already known, defined as *alignment seeds* $\mathbb{L} = \{(e_{i_1}, e_{i_2})|e_{i_1} \in E_1, e_{i_2} \in E_2\}$.

We define the task of entity or relation alignment as automatically finding more equivalent entities or relations based on known alignment seeds. In our model, we only use known *aligned entity pairs* as training data for both entity and relation alignments. The process of relation alignment in our framework is unsupervised, which does not need pre-aligned relation pairs for training.

## 4 Our Approach

Given two target KGs, $G_1$ and $G_2$, and a set of known aligned entity pairs $\mathbb{L}$, our approach uses GCNs (Kipf and Welling, 2017) with highway network (Srivastava et al., 2015) gates to embed entities of the two KGs and approximate relation semantics based on entity representations. By linking entity representations with relation representations, they promote each other in our framework and ultimately achieve better alignment results.

### 4.1 Overall Architecture

As illustrated in Figure 1, our approach consists of three stages: (1) preliminary entity alignment, (2) approximating relation representations, and (3) joint entity and relation alignment.

In the first stage, we utilize GCNs to embed entities of various KGs in a unified vector space for preliminary entity alignment. Next, we use the entity embeddings to approximate relation representations which can be used to align relations across KGs. In the third stage, we incorporate the relation representations into entity embeddings to obtain the joint entity representations, and continue using GCNs to iteratively integrate neighboring structural information to achieve better entity and relation representations.

### 4.2 Preliminary Entity Alignment

As shown in Figure 1, we put $G_1$ and $G_2$ in one graph $G_a = (E_a, R_a, T_a)$ to form our model's input. We utilize pre-aligned entity pairs to train our model and then discover latent aligned entities.

**Graph Convolutional Layers.** Our entity alignment model utilizes GCNs to embed entities in $G_a$. Our model consists of multiple stacked GCN layers so that it can incorporate higher degree neighborhoods. The input for GCN layer $l$ is a node feature matrix, $\mathbf{X}^{(l)} = \{\boldsymbol{x}_1^{(l)}, \boldsymbol{x}_2^{(l)}, ..., \boldsymbol{x}_n^{(l)}|\boldsymbol{x}_i^{(l)} \in \mathbb{R}^{d^{(l)}}\}$, where $n$ is the number of nodes (entities) of $G_a$, and $d^{(l)}$ is the number of features in layer $l$. $\mathbf{X}^{(l)}$ is updated us-

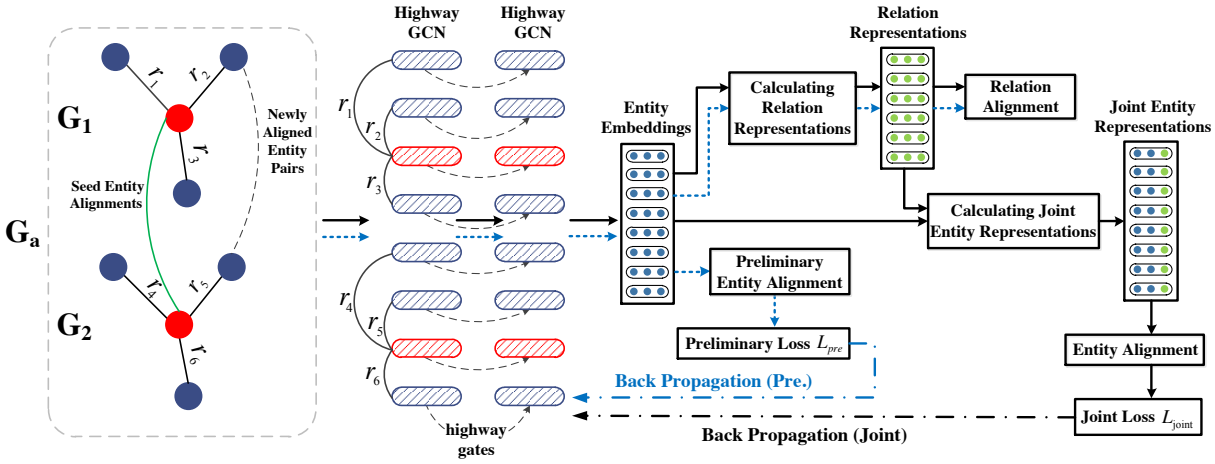


Figure 1: Overall architecture of our model. The blue dotted lines denote the process of preliminary entity alignment and preliminary relation alignment using approximate relation representations, and the black solid lines denote the process of continuing using GCNs to iteratively learn better entity and relation representations.

ing forward propagation as:

$$\mathbf{X}^{(l+1)} = \mathrm{ReLU}(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}\mathbf{X}^{(l)}\mathbf{W}^{(l)}), \quad (1)$$

where $\tilde{A} = A + I$ is the adjacency matrix of $G_a$ with self-connections, $I$ is an identity matrix, $\tilde{D}_{jj} = \sum_k \tilde{A}_{jk}$, and $\mathbf{W}^{(l)} \in \mathbb{R}^{d^{(l)} \times d^{(l+1)}}$ is a layer-specific trainable weight matrix.

Inspired by (Rahimi et al., 2018) that uses highway gates (Srivastava et al., 2015) to control the noise propagation in GCNs for geographic localization, we also employ layer-wise highway gates to build a *Highway*-GCN (HGCN) model. Our layer-wise gates work as follow:

$$T(\mathbf{X}^{(l)}) = \sigma(\mathbf{X}^{(l)}\mathbf{W}_T^{(l)} + \boldsymbol{b}_T^{(l)}), \quad (2)$$

$$\mathbf{X}^{(l+1)} = T(\mathbf{X}^{(l)}) \cdot \mathbf{X}^{(l+1)} + (\mathbf{1} - T(\mathbf{X}^{(l)})) \cdot \mathbf{X}^{(l)} \quad (3)$$

where $\mathbf{X}^{(l)}$ is the input to layer $l+1$; $\sigma$ is the sigmoid function; $\cdot$ is element-wise multiplication; $\mathbf{W}_T^{(l)}$ and $\boldsymbol{b}_T^{(l)}$ are the weight matrix and bias vector for the transform gate $T(\mathbf{X}^{(l)})$, respectively.

**Alignment.** In our work, entity alignment is performed by simply measuring the distance between two entity nodes on their embedding space. With the output entity representations $\mathbf{X}' = \{\boldsymbol{x}'_1, \boldsymbol{x}'_2, ..., \boldsymbol{x}'_n | \boldsymbol{x}'_i \in \mathbb{R}^d\}$, for entities $e_1$ from $G_1$ and $e_2$ from $G_2$, their distance is calculated as:

$$d(e_1, e_2) = \|\boldsymbol{x}'_{e_1} - \boldsymbol{x}'_{e_2}\|_{L_1}. \quad (4)$$

**Training.** We use a margin-based scoring function as the training objective, to make the distance between aligned entity pairs to be as close as possible, and the distance between positive and negative alignment pairs to be as large as possible. The loss function is defined as:

$$L = \sum_{(p,q)\in\mathbb{L}} \sum_{(p',q')\in\mathbb{L}'} \max\{0, d(p,q) - d(p',q') + \gamma\}, \quad (5)$$

where $\gamma > 0$ is a margin hyper-parameter; $\mathbb{L}'$ stands for the negative alignment set of $\mathbb{L}$.

Rather than simply random sampling for negative instances, we look for more challenging negative samples, e.g., those with subtle differences from the positive ones, to train our model. Given a positive aligned pair $(p, q)$, we choose the $\mathcal{K}$-nearest entities of $p$ (or $q$) according to Eq. 4 in the embedding space to replace $q$ (or $p$) as the negative instances.

### 4.3 Approximating Relation Representations

At this stage, we expect to obtain relation representations, which can be used in the next stage for constructing joint representations and can also be used for preliminary relation alignment. Since we are unable to explicitly modeling relations within our GCN-based framework, we thus approximate the relation representations based on their head and tail entity representations produced by the entity alignment model described in Section 4.2. This strategy is based on our observation that the statistical information of the head and tail entities of a relation can more or less reflect the shallow semantics of the relation itself, such as the head or tail entities' type requirements of a relation. Our

experiments in Section 6 suggest that this is a reasonable assumption.

Given a relation $r \in R_a$, there is a set of triples of $r$, $\mathbb{T}_r = \{(h_i, r, t_j)|h_i \in H_r, t_j \in T_r\}$, where $H_r$ and $T_r$ are the sets of head entities and tail entities of relation $r$, respectively. For a relation $r$, its representation can be approximated as:

$$\boldsymbol{r} = f(\mathbf{H}_r, \mathbf{T}_r), \tag{6}$$

where $\boldsymbol{r}$ is the approximated representation of relation $r$. $\mathbf{H}_r$ and $\mathbf{T}_r$ are the sets of HGCN-output embeddings of head entities and tail entities of relation $r$. $f(\cdot)$ is a function to produce relation representations with input entity vectors, which can take many forms such as *mean*, *adding*, *concatenation* or more complex models. In our model, we compute the relation representation for $r$ by first concatenating its averaged head and tail entity representations, and then introducing a matrix $\mathbf{W}_R \in \mathbb{R}^{2\tilde{d}\times m}$ as a learnable shared linear transformation on relation vectors. Here, $\tilde{d}$ is the number of features in each HGCN-output entity embedding and $m$ is the number of features in each relation representation.

With the relation representations in place, relation alignment can be performed by measuring the similarity between two relation vectors. For relation $r_1$ from $G_1$ and $r_2$ from $G_2$, their similarity is computed as:

$$s(r_1, r_2) = \|\boldsymbol{r}_1 - \boldsymbol{r}_2\|_{L_1} - \beta\frac{|P_{r_1r_2}|}{|HT_{r_1} \cup HT_{r_2}|}, \tag{7}$$

where $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ are the relation representations for $r_1$ and $r_2$. In addition to calculating the distance between the two relation vectors, we believe that the more aligned entities exist in the entities that are connected to the two relations, the more likely the two relations are equivalent. Thus, for $r_1$ and $r_2$, we collect the pre-aligned entities existing in the head/tail entities of these two relations as the set $P_{r_1r_2} = \{(e_{i_1}, e_{i_2})|e_{i_1} \in HT_{r_1}, e_{i_2} \in HT_{r_2}, (e_{i_1}, e_{i_2}) \in \mathbb{L}\}$. $HT_{r_1}$ and $HT_{r_2}$ are the sets of head/tail entities for relation $r_1$ and $r_2$ respectively. $\beta$ is a hyper-parameter for balance.

In our framework, relation alignment is explored in an unsupervised fashion, in which we do not have any pre-aligned relations as training data.

### 4.4 Joint Entity and Relation Alignment

The first two stages of our approach could already produce a set of entity and relation alignments, but we do not stop here. Instead, we attentively fuse the entity and relation representations and further jointly optimize them using the seed entity alignments. Our key insight is that entity and relation alignment tasks are inherently closely related. This is because aligned entities tend to have some relations in common, and similar relations should have similar categories of head and tail entities.

Specifically, we first pre-train the entity alignment model (Section 4.2) until its entity alignment performance has converged to be stable. We assume that both the pre-trained entity and approximate relation representations can provide rich information for themselves. Next, for each entity, we aggregate the representations of its relevant relations into a relation context vector, which is further combined with its pre-trained entity representation to form a new joint entity representation.

Formally, for each entity $e \in E_a$, its new joint representation $\boldsymbol{e}_{joint}$ can be calculated as:

$$\boldsymbol{e}_{joint} = g(\boldsymbol{e}, \mathbf{R}_e), \tag{8}$$

where $\boldsymbol{e}$ is the HGCN-output representation of entity $e$. $\mathbf{R}_e$ is the set of relation representations of $e$'s relevant relations. $g(\cdot)$ is a function to produce the new joint entity representation by taking $\boldsymbol{e}$ and $\mathbf{R}_e$ as input, which can also take many forms of operations. In our model, we calculate $\boldsymbol{e}_{joint}$ by first summing all relation representations in $\mathbf{R}_e$ and then concatenating $\boldsymbol{e}$ with the summed relation context vector.

After getting the new joint entity representations, $\mathbf{X}_{joint}$, we continue optimizing our model against the seed entity alignments, where we use the joint entity representations to calculate the training loss according to Eq. 5 to continue updating HGCNs[1]. Note that the joint entity representations are composed of entity embeddings and relation representations, while the relation representations are also constructed based on the entity embeddings. Hence, after backpropagation of the loss calculated using the joint entity representations, we optimize the entity embeddings.

## 5 Experimental Setup

### 5.1 Datasets

We use DBP15K datasets from (Sun et al., 2017) to evaluate our approach. DBP15K contains three

[1]The training procedure is detailed in Appendix A.

| DBP15K | | #Ent. | #Rel. | #Rel tr. | Alignments #Ent. | Alignments #Rel. |
|---|---|---|---|---|---|---|
| ZH-EN | ZH | 66,469 | 2,830 | 153,929 | 1,5000 | 890 |
| | EN | 98,125 | 2,317 | 237,674 | | |
| JA-EN | JA | 65,744 | 2,043 | 164,373 | 1,5000 | 529 |
| | EN | 95,680 | 2,096 | 233,319 | | |
| FR-EN | FR | 66,858 | 1,379 | 192,191 | 1,5000 | 212 |
| | EN | 105,889 | 2,209 | 278,590 | | |

Table 1: Summary of the DBP15K datasets.

cross-lingual datasets that were built from the English version to Chinese, Japanese and French versions of DBpedia. Each contains data from two KGs in different languages and provides 15K pre-aligned entity pairs. Besides, each dataset also provides some pre-aligned relations. We manually aligned more relations from the three datasets and removed the ambiguously aligned relation pairs to construct the test sets for relation alignment. Table 1 shows the statistics of the three datasets. We stress that our approach achieves entity and relation alignments simultaneously using only a small number of pre-aligned entities, and relation alignments are only used for testing. Following the previous works (Sun et al., 2017; Wang et al., 2018b; Sun et al., 2018), we use 30% of the pre-aligned entity pairs as training data and 70% for testing. Our source code and datasets are freely available online[2].

### 5.2 Implementation Details

We set $\gamma = 1$, $\beta = 20$, and learning rate to 0.001. We sample $\mathcal{K} = 125$ negative pairs every 50 epochs. We use entity names in different KGs for better model initialization. We translate non-English entity names to English via Google Translate, and the entity features are initialized with pre-trained English word vectors *glove.840B.300d*[3] in our model. Note that Google Translate does not always give accurate translations for named entities. We inspected 100 English translations for Japanese and Chinese entity names, and discovered that around 20% of the translations are wrong. The errors are mainly attributed to the missing of titles/modifications and wrong interpretations for person/location names. The inaccurate translation poses further challenges for our model.

### 5.3 Competitive Approaches

**Entity Alignment.** For entity alignment, we compare our approach against six embedding-based entity alignment methods discussed in Section 1: JE (Hao et al., 2016), MTransE (Chen et al., 2017), JAPE (Sun et al., 2017)[4], IPTransE (Zhu et al., 2017), BootEA (Sun et al., 2018) and GCN (Wang et al., 2018b). Among those, BootEA is the best-performing model on DBP15K.

**Relation Alignment.** For relation alignment, we compare our approach with the state-of-the-art BootEA (denoted by BootEA-*R*), and MTransE (denoted by MTransE-*R*). Note that MTransE provides five implementation variants for its alignment model. To provide a fair comparison, we choose the one that does not use pre-aligned relations but gives the best performance for a triple-wise alignment verification (Chen et al., 2017) - a closely related task for relation alignment. Since BootEA and MTransE are translation-based models that encode both entities and relations, relation alignment can be done by measuring the similarities between two relation representations. Furthermore, to evaluate the effectiveness of our proposed relation approximation method, we also build BootEA-*PR* and MTransE-*PR* for relation alignment according to Section 4.3.

### 5.4 Evaluation Methodology

**Model Variants.** To evaluate our design choices, we provide different implementation variants with the following denotations. HGCN is our base GCN model with highway gates and entity name initialization. It has several variants, described as follows. HGCN-*PE* (Section 4.2) and HGCN-*PR* (Section 4.3) are our preliminary models for entity and relation alignments, respectively. HGCN-*JE* and HGCN-*JR* are our complete models that use joint representations to further improve entity alignment and relation alignment (Section 4.4). Finally, GCN-*PE* and GCN-*PR* are the preliminary GCN-based models for entity and relation alignments respectively, which use entity name initialization *but no highway gates*; GCN-*JE* and GCN-*JR* are the corresponding joint learning models; and GCN-*JE-r* is the randomly initialized version of GCN-*JE* without entity name initialization.

[2]https://github.com/StephanieWyt/HGCN-JE-JR

[3]http://nlp.stanford.edu/projects/glove/

[4]We note that (Sun et al., 2017) also provides analysis by considering the outputs of a machine translator and JAPE, and using a theoretically perfect *oracle* predictor to correctly choose in between the results given by the machine translator and JAPE. As this only serves as an interesting up-bound analysis, but does not reflect the capability of JAPE (because it is impossible to build such a perfect predictor in the first place), we do not compare to this oracle implementation.

**Metrics.** Like prior works (Sun et al., 2017; Wang et al., 2018b; Sun et al., 2018), we use Hits@k as our evaluation metric. A Hits@k score is computed by measuring the proportion of correctly aligned entities ranked in the top $k$ list. Hence, we prefer higher Hits@k scores that indicate better performance.

## 6 Experiment Results

In this section, we first show that our complete model consistently outperforms all alternative methods across datasets, metrics and alignment tasks. We then analyze the impact of prior alignment data size on model performance, showing that our approach requires significantly less training data but achieves better performance over the best-performing prior method. Finally, we use a concrete example to discuss how jointly learned entity and relation representations can be used to improve both entity and relation alignments.

### 6.1 Entity Alignment

Table 2 reports the performance for entity alignment of all compared approaches. The top part of the table shows the performance of prior approaches. By using a bootstrapping process to expand the training data, BootEA clearly outperforms all prior methods. By capturing the rich neighboring structural information, GCN outperforms all other translation-based models on Hits@1, and over IPTransE, MTransE and JE on Hits@10.

The bottom part of Table 2 shows how our proposed techniques, i.e., entity name initialization, joint embeddings and layer-wise highway gates, can be used within a GCN framework to improve entity alignment. After initialized with the machine-translated entity names, GCN-PE considerably improves GCN on all datasets. The improvement suggests that even rough translations of entity names (see Section 5.2) can still provide important evidence for entity alignment and finally boost the performance. By employing layer-wise highway gates, HGCN-PE further improves GCN-PE, giving a 34.31% improvement on Hits@1 on *DBP15K*$_{FR-EN}$, and also outperforms the strongest baseline BootEA. This substantial improvement indicates that highway gates can effectively control the propagation of noisy information. Our complete framework HGCN-JE gives the best performance across all metrics and datasets. Comparing HGCN-JE with HGCN-PE and GCN-JE with GCN-PE (2.36% and 4.19% improvements of Hits@1 on *DBP15K*$_{ZH-EN}$ respectively), we observe that joining entity and relation alignments improves the model performance. Even without entity name initialization, GCN-JE-r still has obvious advantages over JE, MTransE, JAPE, IPTransE and GCN. The results reinforce our claim that merging the relation information into entities can produce better entity representations. We stress that our proposed methods are not restricted to GCNs or HGCNs, but can be flexibly integrated with other KG representation models as well.

| Models | ZH-EN | | JA-EN | | FR-EN | |
|---|---|---|---|---|---|---|
| | Hits@1 | Hits@10 | Hits@1 | Hits@10 | Hits@1 | Hits@10 |
| JE | 21.27 | 42.77 | 18.92 | 39.97 | 15.38 | 38.84 |
| MTransE | 30.83 | 61.41 | 27.86 | 57.45 | 24.41 | 55.55 |
| JAPE | 41.18 | 74.46 | 36.25 | 68.50 | 32.39 | 66.68 |
| IPTransE | 40.59 | 73.47 | 36.69 | 69.26 | 33.30 | 68.54 |
| BootEA | 62.94 | 84.75 | 62.23 | 85.39 | 65.30 | 87.44 |
| GCN | 41.25 | 74.38 | 39.91 | 74.46 | 37.29 | 74.49 |
| **GCN-JE-r** | 45.92 | 72.91 | 46.62 | 74.57 | 48.27 | 77.82 |
| **GCN-PE** | 50.70 | 79.23 | 53.01 | 82.83 | 54.42 | 84.65 |
| **GCN-JE** | 54.89 | 81.30 | 56.35 | 83.73 | 58.30 | 86.34 |
| **HGCN-PE** | 69.67 | 82.56 | 75.50 | 87.89 | 88.73 | 95.52 |
| **HGCN-JE** | **72.03** | **85.70** | **76.62** | **89.73** | **89.16** | **96.11** |

Table 2: Performance on entity alignment.

| Models | ZH-EN | | JA-EN | | FR-EN | |
|---|---|---|---|---|---|---|
| | Hits@1 | Hits@10 | Hits@1 | Hits@10 | Hits@1 | Hits@10 |
| MTransE-R | 3.03 | 8.88 | 2.65 | 10.21 | 3.30 | 14.62 |
| MTransE-PR | 32.81 | 57.64 | 31.00 | 56.14 | 18.87 | 44.34 |
| BootEA-R | 55.17 | 70.00 | 47.83 | 67.67 | 36.79 | 58.49 |
| BootEA-PR | 45.28 | 85.37 | 41.40 | 79.77 | 30.19 | **60.38** |
| **GCN-PR** | 66.18 | 82.81 | 60.87 | 81.47 | 38.21 | 52.83 |
| **GCN-JR** | 70.22 | 82.81 | 63.89 | 81.10 | 41.98 | 53.77 |
| **HGCN-PR** | 69.33 | 84.49 | 63.14 | 81.26 | 41.51 | 54.25 |
| **HGCN-JR** | **70.34** | **85.39** | **65.03** | **83.55** | **42.45** | 56.60 |

Table 3: Performance on relation alignment.

### 6.2 Relation Alignment

Table 3 reports the results of relation alignment. Directly using the relation embeddings learned by MTransE to perform relation alignment leads to rather poor performance for MTransE-R, less than 4% for Hits@1 for all datasets. This is because the translation assumption, $head + relation \approx tail$, used by MTransE focuses on modeling the overall relationship among heads, tails, and relations, but capturing little neighboring information and relation semantics. After approximating the relation representations using entity embeddings according to Eq 6, MTransE-PR substantially improves MTransE-R. This confirms our assumption that it is feasible to approximate a relation using the in-

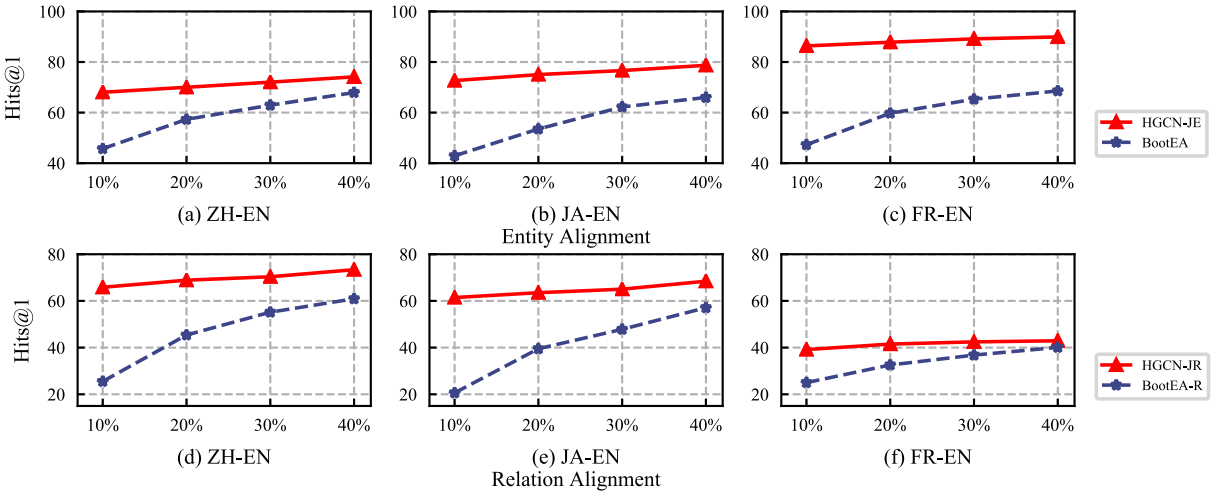


Figure 2: (a)-(c) report the performance for entity alignment of HGCN-JE and BootEA when they are trained with different proportions of seed entity alignments on the three DBP15K datasets. (d)-(f) show the relation alignment performance of HGCN-JR and BootEA-R under corresponding conditions. The x-axes are the proportions of seed alignments, and the y-axes are Hits@1 scores.

formation of its head and tail entities.

The strong entity alignment model BootEA also performs well for relation alignment. Using the relation embeddings from BootEA, BootEA-R delivers the best Hits@1 in MTransE and BootEA variants. Using our approximation strategy hurts BootEA-R in Hits@1, but we see improvements on Hits@10 across all datasets. This suggests that our approximation method can bring more related candidates, but may lack precision to select top-ranked candidates, comparing to explicitly relation modeling in translation-based models.

Our framework, HGCN-JR, delivers the best relation alignment results across datasets and metrics, except for Hits@10 on $DBP15K_{FR-EN}$. Like entity alignment, we also observe that joining entity and relation alignments improves relation alignment, as evidenced by the better performance of HGCN-JR and GCN-JR over HGCN-PR and GCN-PR, respectively. That is, joint modeling produces better entity representations, which in turn provide better relation approximations. This can promote the results of both alignment tasks.

### 6.3 Analysis

**Impact of Available Seed Alignments.** To explore the impact of the size of seed alignments on our model, we compare our HGCN with BootEA by varying the proportion of pre-aligned entities from 10% to 40% with a step of 10%. Figure 2 (a-c) illustrate the Hits@1 for entity alignment of HGCN-JE and BootEA on three datasets. As the amount of seed alignments increases, the performances of both models on all three data sets gradually improve. HGCN-JE consistently obtains superior results compared to BootEA, and seems to be insensitive to the proportion of seed alignments. For example, HGCN-JE still achieves 86.40% for Hits@1 on $DBP15K_{FR-EN}$ when only using 10% of training data. This Hits@1 score is 17.84% higher than that of BootEA when BootEA uses 40% of seed alignments.

Figure 2 (d-f) show the Hits@1 for relation alignment of HGCN-JR and BootEA-R. HGCN-JR also consistently outperforms BootEA-R, and gives more stable results with different ratios of seed entity alignments. These results further confirm the robustness of our model, especially with limited seed entity alignments.

**Case Study.** Figure 3 shows an example from $DBP15K_{FR-EN}$. In the stages of preliminary entity alignment and relation alignment, our model correctly predicts the aligned entity pair ($v_2$, $v_5$) and relation pair ($r_2$, $r_5$). After examining the full experimental data[5], we find that the entities with more neighbors, such as $v_2$ and $v_5$ (indicating *Norway*), and the high-frequency relations, such as $r_2$ and $r_5$ (indicating *country*), are easier to align, since such entities and relations have rich structural information that can be exploited by a GCN. After jointly learning entity and relation representations, the extra neighboring relation information

[5] A more detailed analysis of our experimental results can be found in Appendix B in the supplementary material.

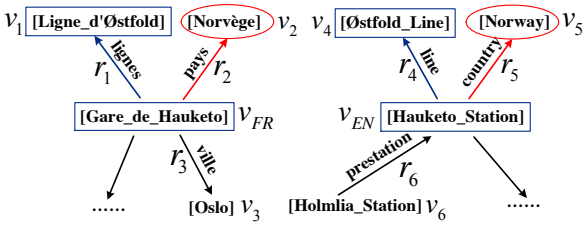


Figure 3: A real-world example from $DBP15K_{FR-EN}$. $[v_2; v_5]$ and $[r_2; r_5]$ are respectively the aligned entities and aligned relations after performing preliminary entity and relation alignments. $[v_{FR}; v_{EN}]$ and $[v_1; v_4]$ are the newly aligned entity pairs, and $r_1$ and $r_4$ are the newly aligned relations, which are discovered using jointly learned entity and relation representations. Jointly optimizing alignment tasks leads to the sucessful discovery of new aligned relation and entity pairs.

(e.g., the aligned relations ($r_2$, $r_5$)) enables our model to successfully align $v_{FR}$ and $v_{EN}$. If we keep updating the model to learn better entity and relation representations, our alignment framework can successfully uncover more entity and relation alignments such as ($v_1$, $v_4$) and ($r_1$, $r_4$). This shows that joint representations can improve both entity and relation alignments.

## 7 Conclusions

This paper presents a novel framework for entity alignment by jointly modeling entities and relations of KGs. Our approach does not require pre-aligned relations as training data, yet it can simultaneously align entities and relations of heterogeneous KGs. We achieve this by employing gated GCNs to automatically learn high-quality entity and relation representations. As a departure from prior work, our approach constructs joint entity representations that contain both relation information and entity information. We demonstrate that *the whole is greater than the sum of its parts*, as the joint representations allow our model to iteratively improve the learned representations for both entities and relations. Extensive experiments on three real-world datasets show that our approach delivers better and more robust performance when compared to state-of-the-art methods.

## Acknowledgments

This work is supported in part by the National Hi-Tech R&D Program of China (No. 2018YFB1005100), the NSFC Grants (No. 61672057, 61672058, 61872294), and a UK Royal Society International Collaboration Grant (IE161012). For any correspondence, please contact Yansong Feng.

## References

Giannis Bekoulis, Johannes Deleu, Thomas Demeester, and Chris Develder. 2018. Adversarial training for multi-context joint entity and relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2830–2836.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

Muhao Chen, Yingtao Tian, Kai-Wei Chang, Steven Skiena, and Carlo Zaniolo. 2018. Co-training embeddings of knowledge graphs and entity descriptions for cross-lingual entity alignment. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*.

Muhao Chen, Yingtao Tian, Mohan Yang, and Carlo Zaniolo. 2017. Multilingual knowledge graph embeddings for cross-lingual knowledge alignment. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*.

David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alan Aspuru-Guzik, and Ryan P Adams. 2015. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in Neural Information Processing Systems 28*, pages 2224–2232.

Yanchao Hao, Yuanzhe Zhang, Shizhu He, Kang Liu, and Jian Zhao. 2016. A joint embedding method for entity alignment of knowledge bases. In *Proceedings of China Conference on Knowledge Graph and Semantic Computing (CCKS2016)*, pages 3–14.

Wei Hu, Jianfeng Chen, and Yuzhong Qu. 2011. A self-training approach for resolving object coreference on the semantic web. In *Proceedings of the 20th international conference on World wide web*.

Steven Kearnes, Kevin Mccloskey, Marc Berndl, Vijay Pande, and Patrick Riley. 2016. Molecular graph convolutions: moving beyond fingerprints. *Journal of Computer-Aided Molecular Design*, 30(8):1–14.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

Simon Lacoste-Julien, Konstantina Palla, Alex Davies, Gjergji Kasneci, Thore Graepel, and Zoubin Ghahramani. 2013. Sigma: Simple greedy matching for aligning large knowledge bases. In *Proceedings*

*of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*.

Shengnan Li, Xin Li, Rui Ye, Mingzhong Wang, Haiping Su, and Yingzi Ou. 2018. Non-translational alignment for multi-relational networks. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 4180–4186.

Farzaneh Mahdisoltani, Joanna Biega, and Fabian M Suchanek. 2013. Yago3: A knowledge base from multilingual wikipedias. In *CIDR*.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1506–1515.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using LSTMs on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1105–1116.

Thanh Nguyen, Viviane Moreira, Huong Nguyen, Hoa Nguyen, and Juliana Freire. 2011. Multilingual schema matching for wikipedia infoboxes. *Proc. VLDB Endow.*

Shichao Pei, Lu Yu, Robert Hoehndorf, and Xiangliang Zhang. 2019. Semi-supervised entity alignment via knowledge graph embedding with awareness of degree difference. In *The World Wide Web Conference*, WWW '19, pages 3130–3136.

Afshin Rahimi, Trevor Cohn, and Timothy Baldwin. 2018. Semi-supervised user geolocation via graph convolutional networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 2009–2019.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *arXiv preprint arXiv:1505.00387*.

Fabian M. Suchanek, Serge Abiteboul, and Pierre Senellart. 2011. Paris: Probabilistic alignment of relations, instances, and schema. *Proc. VLDB Endow.*, 5(3):157–168.

Zequn Sun, Wei Hu, and Chengkai Li. 2017. Cross-lingual entity alignment via joint attribute-preserving embedding. In *International Semantic Web Conference*, pages 628–644.

Zequn Sun, Wei Hu, Qingheng Zhang, and Yuzhong Qu. 2018. Bootstrapping entity alignment with knowledge graph embedding. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 4396–4402.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A free collaborative knowledgebase. *Communications of the ACM*, 57:78–85.

Guanying Wang, Wen Zhang, Ruoxu Wang, Yalin Zhou, Xi Chen, Wei Zhang, Hai Zhu, and Huajun Chen. 2018a. Label-free distant supervision for relation extraction via knowledge graph embedding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2246–2255.

Xuepeng Wang, Kang Liu, Shizhu He, Shulin Liu, Yuanzhe Zhang, and Jun Zhao. 2017. Multi-source knowledge bases entity alignment by leveraging semantic tags. *Chinese Journal of Computers*, 40(3):701–711.

Zhichun Wang, Qingsong Lv, Xiaohan Lan, and Yu Zhang. 2018b. Cross-lingual knowledge graph alignment via graph convolutional networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 349–357.

Yuting Wu, Xiao Liu, Yansong Feng, Zheng Wang, Rui Yan, and Dongyan Zhao. 2019. Relation-aware entity alignment for heterogeneous knowledge graphs. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 5278–5284.

Bishan Yang and Tom Mitchell. 2017. Leveraging knowledge bases in lstms for improving machine reading. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Yang Yang, Yizhou Sun, Jie Tang, Bo Ma, and Juanzi Li. 2015. Entity matching across heterogeneous sources. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018a. Graph convolution over pruned dependency trees improves relation extraction. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J Smola, and Le Song. 2018b. Variational reasoning for question answering with knowledge graph. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Hao Zhu, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2017. Iterative entity alignment via joint knowledge embeddings. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 4258–4264.

# Tackling Long-Tailed Relations and Uncommon Entities in Knowledge Graph Completion *

**Zihao Wang**[1], **Kwun Ping Lai**[1], **Piji Li**[12], **Lidong Bing**[3], **Wai Lam**[1],
[1]The Chinese University of Hong Kong
[2]Tencent AI Lab, Shenzhen, China
[3]R&D Center Singapore, Machine Intelligence Technology, Alibaba DAMO Academy
zihaowangbupt@gmail.com
{kplai, wlam}@se.cuhk.edu.hk
pijili@tencent.com
l.bing@alibaba-inc.com

## Abstract

For large-scale knowledge graphs (KGs), recent research has been focusing on the large proportion of infrequent relations which have been ignored by previous studies. For example few-shot learning paradigm for relations has been investigated. In this work, we further advocate that handling uncommon entities is inevitable when dealing with infrequent relations. Therefore, we propose a meta-learning framework that aims at handling infrequent relations with few-shot learning and uncommon entities by using textual descriptions. We design a novel model to better extract key information from textual descriptions. Besides, we also develop a novel generative model in our framework to enhance the performance by generating extra triplets during the training stage. Experiments are conducted on two datasets from real-world KGs, and the results show that our framework [1] outperforms previous methods when dealing with infrequent relations and their accompanying uncommon entities.

## 1 Introduction

Modern knowledge graphs (KGs)(Bollacker et al., 2008; Lehmann et al., 2015; Vrandečić and Krötzsch, 2014) consist of a large number of facts, where each fact is represented as a triplet consisting of two entities and a binary relation between them. KGs provide rich information and it has been widely adopted in different tasks, such as question answering (Yih et al., 2015), information extraction (Bing et al., 2017, 2015, 2016) and image classification (Marino et al., 2017). However, KGs still have the issue of incomplete facts. To deal with the problem, Knowledge Graph Completion (KGC) task is introduced to automatically deduce and fill the missing facts. There exist many previous works focusing on this task and embedding-based methods (Bordes et al., 2013; Wang et al., 2014; Trouillon et al., 2016) achieves the best performance among them. Recent works such as (Xiong et al., 2018) have pointed out that relations in KGs follow a long-tailed distribution. To be more precise, a large proportion of relations have only a few facts in KGs. However, previous works of KGC usually focused on small proportions of frequent relations and ignored the remaining ones. One observation is that they often conducted experiments on small datasets such as FB15k and WN18 (Bordes et al., 2013) where a relation typically possesses thousands of facts. Moreover, after analyzing real-world KGs, we find that the more infrequently a relation appears, the entities within its facts are also more uncommon. Figure 1 depicts the relationship between the relation frequency and the proportion of uncommon entities that appear in the facts of these relations in a KG, where an entity is treated as uncommon when it appears less or equal than 5 times in all triplets of the KG. From Figure 1, it is obvious that less frequent relations involve more uncommon entities than frequent relations. Therefore, when dealing with the problem of infrequent relations, the issue of uncommon entities should also be considered simultaneously, where they are two sides of a coin.

Previous works such as (Xiong et al., 2018) only focused on those infrequent relations and ignored the accompanying problem of uncommon entities. When handling uncommon entities, relying only on the structural information of KGs would lead to inferior performance due to data insufficiency,

*The work described in this paper is substantially supported by a grant from the Research Grant Council of the Hong Kong Special Administrative Region, China (Project Code: 14204418).

[1]The implementation of our framework can be found in https://github.com/ZihaoWang/Few-shot-KGC.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 250–260, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

and thus additional information is required. Some works (Toutanova et al., 2015; Xie et al., 2016) utilize textual description of entities, but they cannot extract different information from entity description if the entity is involved in more than one relations. A recent work (Shi and Weninger, 2018) tries to tackle this problem by using an attention mechanism considering both entity description and relation, but it adopts a heuristic method that cannot generalize well.

In this paper, we consider performing KGC for infrequent relations and uncommon entities as a few-shot learning problem, and we propose a framework that consists of three main components: description encoder, triplet generator, and meta-learner. In the description encoder, we design a novel structure to handle entities involved with multiple relations by automatically locating and extracting relation-specific information. We also simultaneously learn a triplet generator that is able to generate extra triplets in order to relieve the problem of data sparsity in few-shot learning. Moreover, a meta-learner is further adopted to learn a initial representation of the model that can be easily adapted to unseen relations and entities. As a result, our work has three main contributions as follows:

- We formulate the problem of infrequent relations and uncommon entities as a few-shot learning problem and propose a meta-learning framework to solve it.
- We propose a novel model to extract relation-specific information from entity description for entities with multiple relations.
- We propose a generative model that can enhance the performance of few-shot learning by generating extra triplets during the training stage.

## 2 Related Work

### 2.1 Knowledge Graph Completion

The knowledge graph completion (KGC) task focuses on automatically inferring and filling the missing facts in KG. The most successful method of KGC is the embedding-based method that learns a latent embedding in a common space for entities and relations. This method only relies on the structural information of entities and relations in KG. There exist a variety of methods (Bordes et al., 2013; Socher et al., 2013; Wang et al., 2014; Trouillon et al., 2016; Nguyen et al., 2018) that have been proposed to learn good embeddings for entities and relations.

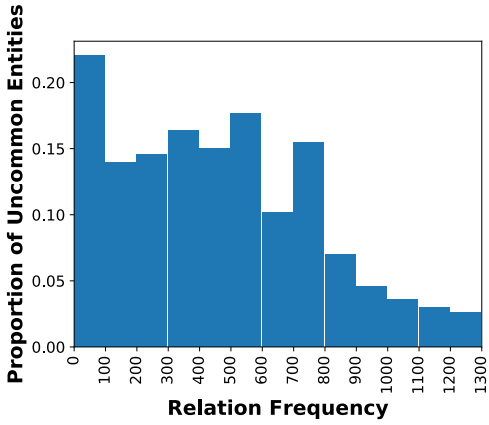


Figure 1: A histogram about relation frequencies and the corresponding proportions of uncommon entities in DBpedia.

However, embedding of uncommon relation or entities can not learn a good representation due to the data insufficiency. Some research has proposed that additional information can be introduced to enhance the learning performance. Among different types of information, textual descriptions is commonly considered by previous works (Zhong et al., 2015; Toutanova et al., 2015; Xie et al., 2016; Shi and Weninger, 2018). Recently, meta-learning is also proposed by (Xiong et al., 2018) to learn infrequent long-tailed relations in KG.

### 2.2 Meta-Learning

Meta-learning (Lemke et al., 2015) aims at learning common experiences across different tasks and easy adapting the existing model to new tasks. One interesting application of meta-learning is few-shot learning problem where each task has only a few training data available.

Some research focus on learning a general policy for different tasks using a neural network. An early work (Santoro et al., 2016) proposes that the learning policy can be learned by using a global memory network. Recently, temporal convolution and attention have been considered to learn a common representation and pinpoint common experiences (Mishra et al., 2018).

Another direction is to learn a good initial representation where the learned model can be easily adapted to new data. Prototypical Network (Snell

et al., 2017) is proposed to learn a prototype for each category, and thus new data can be classified by distances between data and prototypes. Model-Agnostic Meta-Learning (MAML) (Finn et al., 2017) focuses on learning a good initial point in parameter space of model, hence a trained model can be quickly adapted to new tasks with several updates. More recently, Reptile (Nichol et al., 2018) proposes to be an approximation of MAML. In the Reptile model parameters are updated after a number of steps of inner iteration that can maximize within-task generalization.

## 3 Background and Overview

### 3.1 Problem Setting

Knowledge graph (KG) consists of a set of facts. Each fact has the form of a triplet $(h, r, t)$ where $h$ is a head entity, $r$ is a relation and $t$ is a tail entity. KGs are usually sparse, incomplete, and noisy. Therefore Knowledge Graph Completion (KGC) becomes an important task. Given arbitrary two out of three elements within a triplet, the goal of KGC is to predict the remaining one. We focus on predicting $t$ given $h$ and $r$ in this work since our purpose is to deal with uncommon entities.

Previous works usually considered performing KGC given a set of common relations with lots of triplets. On the contrary, we concentrate on performing KGC on those relations that have only a small number of triplets, which can be viewed as a $k$-shot learning problem for relations when $k$ is a small number. In the limiting case where $k$ equals to 1, we deal with one-shot learning problem in our framework. Besides, unlike the previous work (Xiong et al., 2018) that focuses on common entities, we also consider uncommon entities during operational or testing phase, which means that some entities could appear only several times or be absent before. Moreover, when dealing with uncommon entities, relying only on the structural information of KG would lead to inferior performance due to data insufficiency, and thus additional information is necessary. Textual descriptions have been widely considered in KGC. Typically, textual descriptions are used to describe an entity or a relation, and each of them can be either a short sentence or a paragraph consisting of several sentences. We utilize textual descriptions of entities and relations in our framework.

### 3.2 Overview of Learning Method

Meta learning is a popular paradigm for solving the few-shot learning problem, and we adopt it to perform KGC. Given a KG, we treat each relation as a task, and the triplets of each relation can be viewed as specific data of each task. We further divide all tasks into three disjoint sets $R_{train}$, $R_{val}$ and $R_{test}$. Hence meta-training, meta-validation and meta-testing can be performed on each set respectively. In each iteration of the meta-training phase, we randomly sample $B$ tasks from $R_{train}$ where $B$ is batch size, and then for each task $r$ in the batch we sample some triplets of $r$ to train the model. After meta-training finishes, we obtain a trained model with model parameters $W$. Next, we follow the procedure in the previous work (Xiong et al., 2018) to perform meta-validation and meta-testing on $R_{val}$ and $R_{test}$, where the settings are the same. So we only describe meta-testing for short. In the meta-testing phase, given a new task $r^{'} \in R_{test}$ with $H_{r^{'}}$ triplets, we randomly sample $k$ out of $H_{r^{'}}$ triplets. The trained model is further improved via another training stage with only these $k$ samples and the parameters of model become $W^{'}$. Then we keep parameters being fixed as $W^{'}$ and evaluate the performance of model on the remaining $H_{r^{'}} - k$ triplets. These procedures are repeated for all tasks in $R_{test}$.

Given a triplet, the textual descriptions of $h$, $r$ and $t$ are respectively $d_h$, $d_r$ and $d_t$. With textual descriptions, entities and relations can be mapped into a common semantic space. Therefore, uncommon entities can be tackled in this common space as usual.

## 4 Model Description

In this section we present the architecture and the learning procedure of our proposed framework.

First, given the textual descriptions of a triplet $(h, r, t)$, the description encoder extracts key information from descriptions and produces corresponding embeddings $O = (o_h, o_r, o_t)$, where $o$ is a $u$-dimensional vector in the latent semantic space.

Next, the triplet generator participates in the learning procedure. During meta-training phase, it takes $O$ as inputs and learns latent patterns for triplets. However, during the training stage of the meta-testing phase, instead of learning latent patterns, the triplet generator performs triplet aug-

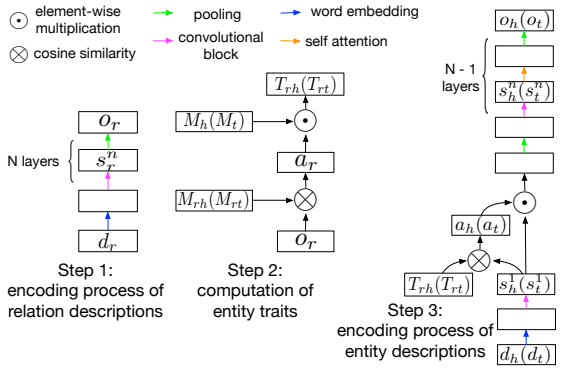


Figure 2: Structure and learning procedures of our description encoder. This figure should be viewed in color.

mentation by generating extra $K$ sets of embeddings $G = \{(g_h, g_r, g_t)\}$. Each set of embeddings $(g_h, g_r, g_t)$ can be viewed as an artificial imitation of $O$. In the few-shot setting, the size of $O$ is usually too small to learn a good representation for a new task, and thus extra embeddings $G$ is generated for data augmentation.

After previous procedures, we are able to obtain a set of embeddings of triplets $E = \{(e_h, e_r, e_t)\}$, where $E = O$ during meta-training phase and $E = O \cup G$ during training stage of meta-testing phase. With $E$ prepared, a score function $F$ takes $E$ as input and computes the score $C$ for each group of embeddings $(e_h, e_r, e_t) \in E$. Although more sophisticated score functions might be designed, in our framework we adopt a simple formula as follows:

$$C = F(h, r, t) = |e_h + e_r - e_t| \qquad (1)$$

from TransE (Bordes et al., 2013), where $L_1$-norm is used.

Finally, during the meta-training phase or the training stage of the meta-testing phase, a loss function $L$ related to $C$ is computed, and the meta-learner is adopted to optimize $L$ so that the framework can be easily adapted to new relations and entities. Otherwise, during the testing stage of the meta-testing phase, we collect scores of the correct triplet and other candidates, and then we compute metrics based on the rank of correct triplet within all scores for evaluation.

### 4.1 Description Encoder

In KG, if an entity is involved in multiple relations, it is natural that different relations are more relevant to different parts in the description of the entity. However, existing works using textual descriptions have not tackled this issue effectively. In order to deal with this issue, we define a new concept "entity trait" ("trait" for short) that represents the common characteristics of some entities related to a special relation. In another word, an entity owns different traits for different relations it involved. In a sense, a trait is similar to an entity type ("type" for short), but it has more advantages when handling KGC. First, types are not relation-specific but traits are. Besides, a trait may consist of semantics of several different types and hence it is more expressive. Moreover, we cannot easily obtain types in some situations, but traits can always be learned properly since they are latent and data-driven. Formally, we assume that a relation $r$ has two traits $T_{rh}$ and $T_{rt}$, where the previous one for all the head entities of $r$ and the latter one for all the tail entities of $r$. In our description encoder, a simple but effective method is adopted to learn and utilize traits to extract relation-specific information from description.

The overall structure and learning process of our description encoder are given in Figure 2. Given the descriptions $(d_h, d_r, d_t)$ of a triplet $(h, r, t)$, there are three steps to obtain the embeddings of triplet $O$ as depicted in the figure. For entity, we only describe the process of $h$ for simplicity, but everything stays the same for $t$. In Step 1, the encoding process of relation descriptions takes $d_r$ as input and outputs a relation embedding $o_r$. Next, $o_r$ is used to learn the trait $T_{rh}$ for all the head entities of the relation $r$ in Step 2. Finally, in Step 3 both $d_h$ and $T_{rh}$ are fed to the encoding process of the entity descriptions, and the output $o_h$ is the embedding of the head entity. Note that the word embedding layer, convolutional blocks and pooling layers in Step 1 share the same parameters and architectures with the corresponding ones in Step 3.

The core of the encoding process is a $N$-layer convolutional neural network (CNN) (Conneau et al., 2017), which is shown to have excellent ability of extracting information. In our CNN, the basic convolutional block consists of three consecutive operations: two 1-d convolutions, an instance normalization (Ulyanov et al., 2016), and a non-linear mapping. For the pooling strategy, max pooling with a proper stride is used to distill the key information in the previous $N - 1$ layers, and

mean pooling is used to gather the information in the last layer. Moreover, in Step 3, we also apply self-attention mechanism (Vaswani et al., 2017) before each pooling layer in the last $N-1$ layers. Unlike Step 1, self-attention is necessary here since entity descriptions are often more complex and noisier than relation descriptions according to our observation. Self-attention can assign lower weights to noise, and then those noise would be filtered out in the subsequent pooling layer.

Furthermore, we demonstrate how to compute the trait in Step 2, where the external memories $M$ play an important role. These memories record the global information of relations and entities that can generalize well when encountering new relations and entities. In detail, the $o_r$ computed in Step 1 is transformed to a probability distribution $a_r$ by using $m$ relation memories $M_{rh}$

$$a_r = softmax(o_r \otimes M_{rh}), \tag{2}$$

where $a_r$ is $m$-dimensional, $\otimes$ denotes the cosine similarity, $softmax$ is the commonly used softmax function and $M_{rh}$ is a matrix with shape $(m, u)$. After that, the $u$-dimensional trait $T_{rh}$ can be obtained by computing a linear combination of $m$ latent entity memories $M_h$

$$T_{rh} = \sum_{i=1}^{m} M_h^i \odot a_r^i, \tag{3}$$

where $\odot$ is the element-wise multiplication between two vectors and $M_h^i$ is a matrix with shape $(m, u)$. Note that each pair of $m$ latent relation memories $M_{rh}$ and $m$ latent entity memories $M_h$ has a one-to-one correspondence.

Finally, we describe how the trait $T_{rh}$ is used to extract the relation-specific information in Step 3. Given the description $d_h$, the hidden states $s_h^1$ can be obtained after the first convolutional block, and then the trait $T_{rh}$ is used to locate important hidden states in $s_h^1$ that have high relevance to $r$ by assigning them higher weights. The procedure here is the same as before: a probability distribution $a_h$ over $s_h^1$ is computed by

$$a_h = softmax(T_{rh} \otimes s_h^1), \tag{4}$$

and then $a_h$ multiplies with $s_h^1$ element-wise to weight different hidden states. In this way, the hidden states that are not relevant to $r$ are assigned lower weights, and thus they are more likely to be filtered in the subsequent max-pooling layer.

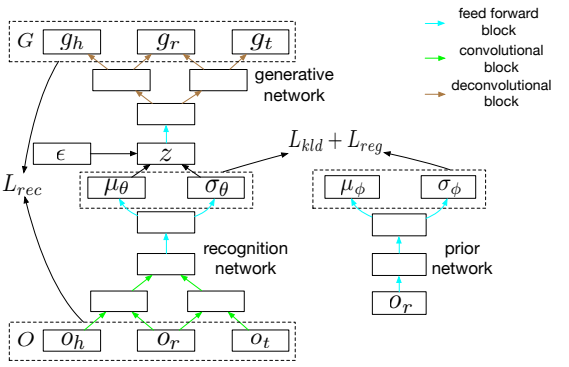


Figure 3: Structure of TCVAE. This figure should be viewed in color.

### 4.2 Triplet Generator

When handling KGC, learning good representation for infrequent relations and uncommon entities is difficult due to data sparsity. However, recently some research has focused on relieving the data sparsity in few-shot learning by generating extra data with a generative model (Schwartz et al., 2018; Wang et al., 2018). Inspired by these works, we propose a deep generative model that aims at triplet augmentation for $k$-shot learning. Although the generative adversarial network (Goodfellow et al., 2014) is a popular model that can generate high-quality samples (Frid-Adar et al., 2018), it suffers from an unstable learning process in our framework because of the difficult nature of Nash equilibrium and the influence of meta-learner. On the other hand, VAE is often applied to generate samples and extract latent semantics (Pu et al., 2016; Li et al., 2017) due to its smooth learning procedure. To cope with this issue, we design our triplet generator on the basis of CVAE (Sohn et al., 2015) and we name it triplet CVAE (TCVAE) in this paper. Figure 3 depicts the overall structure of TCVAE that is composed of three important probability distributions:

- Variational posterior distribution $q_\theta(z|O)$ parameterized by $\theta$ of the recognition network.
- Conditional prior distribution $p_\phi(z|o_r)$ parameterized by $\phi$ of the prior network.
- Likelihood distribution $p_\psi(G|z, o_r)$ parameterized by $\psi$ of the generative network.

In the recognition network, there exist two layer of convolutional blocks. Each convolutional block takes two $u$-dimensional inputs and concatenates

them to form a matrix with shape $(2, u)$, so that 1-d convolution with filter width 2 can be applied to it. Instead of directly concatenate $O = (o_h, o_r, o_t)$ to form a matrix with shape $(3, u)$ and adopt only one layer of convolutional block, such a tree structure of two consecutive layers can better capture the pairwise semantics between any two embeddings in $O$. Likewise, two layers of the deconvolutional blocks that takes a $u$-dimensional vector as input and outputs a matrix with shape $(2, u)$ are placed in the generative network. Besides, all feed forward blocks in TCVAE consist of an affine transformation and a non-linear mapping.

During the meta-training phase, the recognition network takes $O$ as input and learns the variational parameters $\mu_\theta$ and $\sigma_\theta$ of the variational posterior $q_\theta(z|O)$, where the latent semantics $z$ is assumed to follow a Gaussian distribution. Besides, the prior network conditioning on $o_r$ computes the parameters $\mu_\phi$ and $\sigma_\phi$ of the prior distribution $p_\phi(z|o_r)$. After that, the generative network samples three $u$-dimensional embeddings $G = (g_h, g_r, g_t)$ from the likelihood distribution $p_\psi(G|z, o_r)$. In the generative network, firstly, the latent variable $z$ is transformed into a $u$-dimensional hidden state with the feed forward block after $z$. Next, the first deconvolutional block receives the hidden state and outputs a matrix with shape $(2, u)$. Finally, the second deconvolutional block receives the matrix before and outputs a matrix with shape $(3, u)$ which is denoted to $G$. $G$ consists of three u-dimensional embeddings $(g_h, g_r, g_t)$ corresponding to three elements $(h, r, t)$ in the triplet.

Given the procedure, we are able to write down the loss terms of TCVAE: $L_{rec}$, $L_{kld}$ and $L_{rec}$. More formally, $L_{rec}$ is the expected log-likelihood that is also the reconstruction loss between the input $O$ and the output $G$

$$L_{rec} = E_{q_\theta(z|O)} \log p_\psi(G|z, o_r). \quad (5)$$

$L_{kld}$ is the KL-divergence between variational posterior distribution and conditional prior distribution

$$L_{kld} = KL(q_\theta(z|O) \| p_\phi(z|o_r)). \quad (6)$$

And $L_{rec}$ is the regularization term for the prior network proposed in (Ivanov et al., 2019)

$$L_{reg} = -\frac{\mu_\phi^2}{2\sigma_\mu^2} + \sigma_\sigma(\log \sigma_\psi - \sigma_\psi), \quad (7)$$

where $\sigma_\mu = 10000$ and $\sigma_\sigma = 0.0001$ are two hyper-parameters. There terms are jointly optimized with the loss function of KGC that we would demonstrate in the following subsection.

During the training stage of meta-testing phase, TCVAE uses prior network to compute $\mu_\phi$ and $\sigma_\phi$ given only the relation embedding $o_r^{'}$ within $O$ where $r^{'} \in R_{test}$. Then it obtains $K$ latent variables $z$ with the following transformation

$$z = \mu_\phi + \sigma_\phi \odot \epsilon \quad \epsilon \sim Gaussian(0, 1). \quad (8)$$

After that, $K$ embeddings of triplet $G$ can be generated from the likelihood distribution $p_\psi(G|z, o_r^{'})$, and $G$ is merged with $O$ to form the final embeddings $E$. Please note that $E$ is subsequently used to compute the score $C$ with the score function $F$ in Equation 1.

### 4.3 Loss Function and Meta-Learner

Following previous works, we adopt a simple strategy to compute the loss function of KGC $L_{KGC}$ in both meta-training phase and the training stage of meta-testing phase. Given a randomly sampled relation $r$, first a positive triplet $(h, r, t)$ is sampled from all triplets of $r$. Next, a negative triplet $(h, r, t^{'})$ can be produced by replacing $t$ with another entity $t^{'}$ in KG, where the replacement is based on a uniform negative sampling. Note that if the negative triplet exists in KG, the negative sampling needs to be performed again. With a positive triplet and a negative triplet, embeddings of the positive triplet $E^+$ and embeddings of the negative triplet $E^-$ can be obtained, and then two scores $C^+$ and $C^-$ can be computed respectively. Finally, a hinge loss related to both scores is minimized for performing KGC

$$L_{KGC} = max(0, \gamma + C^+ - C^-), \quad (9)$$

where $\gamma$ is a margin hyper-parameter greater than 0. Moreover, during meta-training phase, $L_{KGC}$ is also jointly optimized with TCVAE so that the overall loss $L$ is

$$L = L_{KGC} - L_{rec} - \lambda_1 L_{kld} - \lambda_2 L_{reg}, \quad (10)$$

where the negative loss terms of TCVAE are minimized, and $\lambda_1$, $\lambda_2$ are two hyper-parameters for weighting terms in the overall loss.

In order to ensure that both description encoder and TCVAE have a good generalization ability when handling infrequent relations and uncommon entities, a meta-learner is further used

to optimize the overall loss $L$. Among different directions of meta-learning, we construct the meta-learner based on Reptile since we find it has the best performance in our task. In the context of KGC, learning with Reptile is different from previous KGC works. During the meta-training phase, Reptile searches for an initial point $W$ within the parameter space of our framework, but such a framework may not perform well when directly used for performing KGC in $R_{test}$. Instead, during training stage of the meta-testing phase, the framework parameters can be quickly adapted to a new point $W^{'}$ that is suitable for performing KGC given a new relation $r^{'} \in R_{test}$, and such an adaptation only needs a few training triplets of $r^{'}$ available. The procedure of learning with the meta-learner is depicted in Algorithm 1, where Adam (Kingma and Ba, 2015) is used during the $S$ inner-training steps.

**Algorithm 1** Learning Procedure of Meta-Learner

**for** iteration $= 1, 2, \ldots$ **do**
  Save parameters of our framework $W$
  **for** $i = 1, \ldots, B$ **do**
    Sample a relation $r \in R_{train}$, then sample a positive and a negative triplet of $r$
    Train our framework for $S$ steps using Adam with learning rate $\alpha_1$
    Save current parameters of our framework $W_i$ and reset them to $W$
  **end for**
  Update $W \leftarrow W + \alpha_2(\frac{1}{B}\sum_{i=1}^{B} W_i - W)$
**end for**

# 5 Experiments

## 5.1 Datasets

| | WDtext | DBPtext |
|---|---|---|
| #Entity | 60304 | 51768 |
| #Relation | 178 | 319 |
| #Word | 131796 | 170844 |
| $\#R_{train}$ | 130 | 220 |
| $\#R_{val}$ | 16 | 30 |
| $\#R_{test}$ | 32 | 69 |
| Avg #words | 5.3 | 170.8 |

Table 1: Statistics of datasets, where "Avg #words" means the average number of words in descriptions.

Existing datasets for KGC usually select triplets consisting of frequently appearing relations and common entities. Recently two datasets focused on infrequent relations are proposed in (Xiong et al., 2018), but they do not contain textual descriptions for relations and entities. To obtain datasets that fulfill the practical few-shot learning situations as investigated in the problem setting, we manually harvest triplets and their textual descriptions from Wikidata (Vrandečić and Krötzsch, 2014) and DBpedia (Lehmann et al., 2015), and then we construct two datasets called WDtext and DBPtext respectively. The statistics of the two datasets are shown in Table 1. Specifically, the average numbers of words in the textual descriptions have a large variation between two datasets. In WDtext, descriptions are usually short phrases with only several words, on the other hand, descriptions in DBPtext consists of thousands of words, and we use first 200 words so that our model can be processed on GPUs. Such a variation of description length can better reveal the performance of our model in different situations. Besides, in order to collect enough triplets for evaluation, we select those relations whose numbers of triplets are greater than 5 and less than 1000, where the contained entities may exist only several times or even unseen during the meta-training phase. In this way, the problem setting of infrequent relations and uncommon entities is fulfilled in our datasets.

## 5.2 Experiment Setting

We compare our model with previous KGC models that can make use of textual descriptions, namely, DKRL (Xie et al., 2016) and ConMask (Shi and Weninger, 2018). We adapt their codes with our implementation. Following the experimental protocol of (Shi and Weninger, 2018), we also remove structural features of DKRL so that it can tackle unseen entities. To facilitate fair comparisons, even though these models are designed without using meta-learning, for each relation in $R_{val}$ and $R_{test}$, we also sample $k$ triplets and put them into $R_{train}$ to ensure that all models make use of similar training data. Besides, the few-shot KGC model GMatching (Xiong et al., 2018) is also used for comparison. We only enable its "neighbor encoder" on WDtext because we cannot collect neighbor information from DBPedia. Moreover, we also design two additional baselines for ablation study. These baselines are constructed by removing specific components and keeping re-

| | WDtext | | | | DBPtext | | | |
|---|---|---|---|---|---|---|---|---|
| Model | Hits@10 | Hits@5 | Hits@1 | MRR | Hits@10 | Hits@5 | Hits@1 | MRR |
| DKRL | 0.180 | 0.143 | 0.104 | 0.137 | 0.100 | 0.041 | 0.010 | 0.054 |
| ConMask | **0.279** | **0.207** | 0.085 | 0.156 | 0.304 | 0.213 | 0.059 | 0.147 |
| GMatching | 0.095 | 0.092 | 0.090 | 0.093 | 0.194 | 0.141 | 0.100 | 0.138 |
| Ours-trait | 0.179 | 0.168 | 0.081 | 0.126 | 0.182 | 0.124 | 0.060 | 0.107 |
| Ours-TCVAE | 0.178 | 0.155 | 0.138 | 0.149 | 0.343 | 0.258 | 0.110 | 0.187 |
| Ours | 0.198 | 0.189 | **0.148** | **0.168** | **0.376** | **0.321** | **0.224** | **0.281** |

Table 2: One-shot KGC results on WDtext and DBPedia, where **bold** numbers indicate best results over different models on the same metric.

| | WDtext | | | | DBPtext | | | |
|---|---|---|---|---|---|---|---|---|
| Model | Hits@10 | Hits@5 | Hits@1 | MRR | Hits@10 | Hits@5 | Hits@1 | MRR |
| DKRL | 0.202 | 0.173 | 0.122 | 0.151 | 0.099 | 0.053 | 0.012 | 0.052 |
| ConMask | **0.303** | 0.211 | 0.107 | 0.173 | 0.333 | 0.225 | 0.077 | 0.163 |
| GMatching | 0.089 | 0.088 | 0.085 | 0.088 | 0.185 | 0.146 | 0.097 | 0.138 |
| Ours-trait | 0.277 | 0.221 | 0.113 | 0.170 | 0.252 | 0.174 | 0.088 | 0.146 |
| Ours-TCVAE | 0.214 | 0.198 | 0.159 | 0.184 | 0.303 | 0.227 | 0.110 | 0.174 |
| Ours | 0.258 | **0.227** | **0.180** | **0.210** | **0.409** | **0.319** | **0.186** | **0.255** |

Table 3: Four-shot KGC results on WDtext and DBPedia, where **bold** numbers indicate best results over different models on the same metric.

| WDtext | | DBPtext | |
|---|---|---|---|
| #Gen | MRR | #Gen | MRR |
| 0 | 0.149 | 0 | 0.187 |
| 2 | 0.166 | 16 | 0.247 |
| 4 | 0.172 | 32 | 0.274 |
| 8 | 0.179 | 64 | 0.304 |
| 16 | 0.146 | 128 | 0.330 |
| 32 | 0.144 | 256 | 0.256 |

Table 4: MRR results of our framework when using different number of augmentation in TCVAE, where #Gen means the number of triplets being generated.

maining parts in our framework. Specifically, trait is removed in the baseline "Ours - trait" and TCVAE is removed in the baseline "Ours - TCVAE" respectively. The experimental setting of hyperparameters and initialization of our framework and baselines can be found in Appendix A.

To make a fair comparison, we use two categories of common metrics: mean reciprocal rank (MRR) and hits@P which is the percentage of correct tail entities ranked in the top P. Besides, experiments are conducted using four different random seeds and we report the average results of four trials. For each model, we select the epoch that has the best performance when evaluating on $R_{val}$ and report the corresponding results on $R_{test}$.

### 5.3 Results of One-Shot Learning

Firstly, we compare our overall framework with baselines by conducting an one-shot KGC experiment, which is the most difficult case in few-shot learning. The results of our overall framework and baselines are shown in Table 2, where we find that our overall framework outperforms the baselines on most metrics. For WDtext, ConMask is a strong baseline and has a better result on Hits@10 and Hits@5, but it performs worse on Hits@1 and MRR compared to our framework. On the other hand, our overall framework outperforms for a large margin compared to all baselines on DBPtext. Since the main difference between the two datasets are the average length of descriptions, we can observe that our framework has a better performance when dealing with long textual descriptions. Besides, the results of the two ablation baselines are significantly worse than our overall framework, and thus we can see both components play an important role in our framework.

### 5.4 Results of Few-Shot Learning

In real-world KGs, few-shot learning of infrequent relations and uncommon entities are more common than one-shot scenario, so we also conduct four-shot KGC as another experiment. We use the same baselines and metrics as that in the one-shot

KGC experiment. The results are shown in Table 3, where we can see that our overall framework also has the best performance on different comparisons except for Hits@10 on WDtext. By comparing our overall framework with two ablation baselines, the importance of traits and triplet augmentation is demonstrated again. Furthermore, when compared with the previous one-shot KGC results, all baselines in this experiment are able to make use of the extra training data to improve their performances. Our framework also performs better on all metrics in WDtext, but it performs inferior on some metrics in DBPtext when compared with the one-shot scenario. One possible reason is descriptions in DBPtext is longer and more complex than that in WDtext, and thus four descriptions of training triplets are too diverse to learn a good representation.

### 5.5 Analysis of Triplet Generation

As shown in the previous KGC experiments, the performance of our framework heavily depends on the triplet generation provided by TCVAE. In this subsection, we further explore the effect of triplet generation by comparing the MRR result of our overall framework with different number of generated triplets in one-shot scenario, where the triplet augmentation is particularly helpful. For this analysis, we only conduct experiment and report the result with one trial for simplicity. The results are shown in Table 4, and we can conclude that a proper data augmentation does enhance the performance of our framework when training data available is scarce. Besides, the appropriate number of triplets being generated varies from one dataset to another, and too many or too few generation leads to an inferior performance. In WDtext, generating 8 extra triplets enhances the performance most, but generating 128 triplets is better in DBPtext. One reason for such a difference is that DBPtext has longer descriptions which also increase the variance of the generated triplets, and thus generating more triplets is necessary to learn a stable representation.

## 6 Conclusions

We consider a new type of KGC where infrequent relations and uncommon entities need to be jointly handled, and we formulate it as a few-shot KGC problem. To tackle the problem, we propose a novel concept ”trait” and adopt it to extract relation-specific information from entity descriptions. Besides, we also design a triplet generator and a meta-learning framework based on Reptile to deal with the issue of few-shot KGC. Moreover, we also conduct two new datasets that focus on this problem setting. The experiments of both one-shot and four-shot scenarios show that our framework has a better performance compared to other baselines.

## References


Lidong Bing, Sneha Chaudhari, Richard C. Wang, and William W. Cohen. 2015. Improving distant supervision for information extraction using label propagation through lists. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 524–529.

Lidong Bing, Bhuwan Dhingra, Kathryn Mazaitis, Jong Hyuk Park, and William W. Cohen. 2017. Bootstrapping distantly supervised IE using joint learning and small well-structured corpora. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3408–3414.

Lidong Bing, Mingyang Ling, Richard C. Wang, and William W. Cohen. 2016. Distant IE by bootstrapping using lists and document structure. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 2899–2905.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 1247–1250. ACM.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 2787–2795. Curran Associates Inc.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2017. Very deep convolutional networks for text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1107–1116. Association for Computational Linguistics.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 1126–1135.

Maayan Frid-Adar, Eyal Klang, Michal Amitai, Jacob Goldberger, and Hayit Greenspan. 2018. Synthetic data augmentation using gan for improved liver lesion classification. *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pages 289–293.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 2672–2680. MIT Press.

Oleg Ivanov, Michael Figurnov, and Dmitry P. Vetrov. 2019. Variational autoencoder with arbitrary conditioning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195.

Christiane Lemke, Marcin Budka, and Bogdan Gabrys. 2015. Metalearning: a survey of trends and technologies. *Artificial Intelligence Review*, 44(1):117–130.

Piji Li, Zihao Wang, Wai Lam, Zhaochun Ren, and Lidong Bing. 2017. Salience estimation via variational auto-encoders for multi-document summarization. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3497–3503.

K. Marino, R. Salakhutdinov, and A. Gupta. 2017. The more you know: Using knowledge graphs for image classification. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20–28.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. 2018. A simple neural attentive meta-learner. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. 2018. A novel embedding model for knowledge base completion based on convolutional neural network. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 327–333. Association for Computational Linguistics.

Alex Nichol, Joshua Achiam, and John Schulman. 2018. On first-order meta-learning algorithms. *CoRR*, abs/1803.02999.

Yunchen Pu, Zhe Gan, Ricardo Henao, Xin Yuan, Chunyuan Li, Andrew Stevens, and Lawrence Carin. 2016. Variational autoencoder for deep learning of images, labels and captions. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2352–2360.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. 2016. Meta-learning with memory-augmented neural networks. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 1842–1850. JMLR.org.

Eli Schwartz, Leonid Karlinsky, Joseph Shtok, Sivan Harary, Mattias Marder, Abhishek Kumar, Rogerio Feris, Raja Giryes, and Alex Bronstein. 2018. Delta-encoder: an effective sample synthesis method for few-shot object recognition. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 2845–2855. Curran Associates, Inc.

Baoxu Shi and Tim Weninger. 2018. Open-world knowledge graph completion. In *Proceedings of the Thirty-second AAAI Conference on Artificial Intelligence*, pages 1957–1964. AAAI Press.

Jake Snell, Kevin Swersky, and Richard S. Zemel. 2017. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 4077–4087.

Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'13, pages 926–934. Curran Associates Inc.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. 2015. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 3483–3491.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509. Association for Computational Linguistics.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 2071–2080. JMLR.org.

Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. 2016. Instance normalization: The missing ingredient for fast stylization. *CoRR*, abs/1607.08022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

Yu-Xiong Wang, Ross B. Girshick, Martial Hebert, and Bharath Hariharan. 2018. Low-shot learning from imaginary data. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7278–7286.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, pages 1112–1119. AAAI Press.

Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation learning of knowledge graphs with entity descriptions. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 2659–2665. AAAI Press.

Wenhan Xiong, Mo Yu, Shiyu Chang, Xiaoxiao Guo, and William Yang Wang. 2018. One-shot relational learning for knowledge graphs. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 1980–1990.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1321–1331.

Huaping Zhong, Jianwen Zhang, Zhen Wang, Hai Wan, and Zheng Chen. 2015. Aligning knowledge and text embeddings by entity descriptions. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 267–272.

# Low-Resource Name Tagging Learned with Weakly Labeled Data

**Yixin Cao**[1] **Zikun Hu**[1] **Tat-Seng Chua**[1]
**Zhiyuan Liu**[2] **Heng Ji**[3]
[1]School of Computing, National University of Singapore, Singapore
[2]Department of CST, Tsinghua University, Beijing, China
[3]Department of CS, University of Illinois Urbana-Champaign, U.S.A.
caoyixin2011@gmail.com,zikunhu@u.nus.edu,dcscts@nus.edu.sg
liuzy@tsinghua.edu.cn,hengji@illinois.edu

## Abstract

Name tagging in low-resource languages or domains suffers from inadequate training data. Existing work heavily relies on additional information, while leaving those noisy annotations unexplored that extensively exist on the web. In this paper, we propose a novel neural model for name tagging solely based on weakly labeled (WL) data, so that it can be applied in any low-resource settings. To take the best advantage of all WL sentences, we split them into high-quality and noisy portions for two modules, respectively: (1) a classification module focusing on the large portion of noisy data can efficiently and robustly pre-train the tag classifier by capturing textual context semantics; and (2) a costly sequence labeling module focusing on high-quality data utilizes Partial-CRFs with non-entity sampling to achieve global optimum. Two modules are combined via shared parameters. Extensive experiments involving five low-resource languages and fine-grained food domain demonstrate our superior performance (6% and 7.8% F1 gains on average) as well as efficiency[1].

## 1 Introduction

Name tagging[2] is the task of identifying the boundaries of entity mentions in texts and classifying them into the pre-defined entity types (e.g., person). It serves as a fundamental role as providing the essential inputs for many IE tasks, such as Entity Linking (Cao et al., 2018a) and Relation Extraction (Lin et al., 2017).

Many recent methods utilize a neural network (NN) with Conditional Random Fields (CRFs) (Lafferty et al., 2001) by treating name tagging as a sequence labeling problem (Lample et al., 2016), which has became a basic architecture due to its superior performance. Nevertheless, NN-CRFs require exhaustive human efforts for training annotations, and may not perform well in low-resource settings (Ni et al., 2017). Many approaches thus focus on transferring cross-domain, cross-task and cross-lingual knowledge into name tagging (Yang et al., 2017; Peng and Dredze, 2016; Mayhew et al., 2017; Pan et al., 2017; Lin et al., 2018; Xie et al., 2018). However, they are usually limited by the extra knowledge resources that are effective only in specific languages or domains.

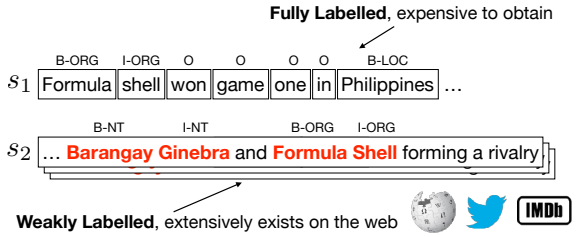


Figure 1: Example of weakly labeled data. B-NT and I-NT denote incomplete labels without types.

Actually, in many low-resource settings, there are extensive noisy annotations that naturally exist on the web yet to be explored (Ni et al., 2017). In this paper, we propose a novel model for name tagging that maximizes the potential of weakly labeled (WL) data. As shown in Figure 1, $s_2$ is weakly labeled, since only *Formula shell* and *Barangay Ginebra* are annotated, leaving the remaining words unannotated.

WL data is more practical to obtain, since it is difficult for people to accurately annotate those entities that they do not know or are not interested in. We can construct them from online resources, such as the anchors in Wikipedia. However, the following natures of WL data make learning name tagging from them more challenging:

**Partially-Labeled Sequence** Automatically

[1]Our project can be found in https://github.com/zig-kwin-hu/Low-Resource-Name-Tagging.
[2]Someone may call it Named Entity Recognition (NER).



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 261–270, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

derived WL data does not contain complete annotations, thus can not be directly used for training. Ni et al. (2017) select the sentences with highest confidence, and assume missing labels as O (i.e., non-entity), but it will introduce a bias to recognize mentions as non-entity. Another line of work is to replace CRFs with Partial-CRFs (Täckström et al., 2013), which assign unlabeled words with all possible labels and maximize the total probability (Yang et al., 2018; Shang et al., 2018). However, they still rely on seed annotations or domain dictionaries for high-quality training.

**Massive Noisy Data** WL corpora are usually generated with massive noisy data including missing labels, incorrect boundaries and types. Previous work filtered out WL sentences by statistical methods (Ni et al., 2017) or the output of a trainable classifier (Yang et al., 2018). However, abandoning training data may exacerbate the issue of inadequate annotation. Therefore, maximizing the potential of massive noisy data as well as high-quality part, yet being efficient, is challenging.

To address these issues, we first differentiate noisy data from high-quality WL sentences via a lightweight scoring strategy, which accounts for the annotation confidence as well as the coverage of all mentions in one sentence. To take best advantages of all WL data, we then propose a unified neural framework that solves name tagging from two perspectives: sequence labeling and classification for two types of data, respectively.

Specifically, the classification module focuses on noisy data to efficiently pretrain the tag classifier by capturing textual context semantics. It is trained only using annotated words without noisy unannotated words, and thus it is robust and efficient during training. The costly sequence labeling module is to achieve sequential optimum among word tags. It further alleviates the burden of seed annotations in Partial-CRFs and increases randomness via Non-entity Sampling strategy, which samples O words according to some linguistic natures. These two modules are combined via shared parameters. Our main contributions are as follows:

- We propose a novel neural name tagging model that merely relies on WL data without feature engineering. It can thus be adapted for both low-resource languages and domains, while no previous work deals with them at the same time.
- We consider name tagging from two perspectives of sequence labeling and classification, to efficiently take the best advantage of both high-quality and noisy WL data.
- We conduct extensive experiments in five low-resource languages and a fine-grained domain. Since few work has been done in two types of low-resource settings simultaneously, we arrive at two types of baselines from state-of-the-art methods. Our model achieves significant improvements (6% and 7.8% F1 on average), yet being efficient demonstrated in further ablation studies.

## 2 Related Work

Name tagging is an fundamental task of extracting entity information, which shall benefit many applications, such as information extraction (Zhang et al., 2017; Kuang et al., 2019; Cao et al., 2019a) and recommendation (Wang et al., 2019; Cao et al., 2019b). It can be treated as either a multi-class classification problem (Hammerton, 2003; Xu et al., 2017) or a sequence labeling problem (Collobert et al., 2011), but very little work combined them together. The difference between them mainly lies in whether the method models sequential label constraints, which have been demonstrated effective in many NN-CRFs models (Lample et al., 2016; Ma and Hovy, 2016; Chiu and Nichols, 2016). However, they require a large amount of human annotated corpora, which are usually expensive to obtain.

The above issue motivates a lot of work on name tagging in low-resource languages or domains. A typical line of effort focuses on introducing external knowledge via transfer learning (Fritzler et al., 2018; Hofer et al., 2018), such as the use of cross-domain (Yang et al., 2017), cross-task (Peng and Dredze, 2016; Lin et al., 2018) and cross-lingual resources (Ni et al., 2017; Xie et al., 2018; Zafarian et al., 2015; Zhang et al., 2016; Mayhew et al., 2017; Tsai et al., 2016; Feng et al., 2018; Pan et al., 2017). Although they achieve promising results, there are a large amount of weak annotations on the Web, which have not been well studied (Nothman et al., 2008; Ehrmann et al., 2011). Yang et al. (2018); Shang et al. (2018) utilized Partial-CRFs (Täckström et al., 2013) to model incomplete annotations for specific domains, but they still rely on seed annotations or a domain dictionary. Therefore, we aim at filling the gap in low-resource name tagging research by using only WL

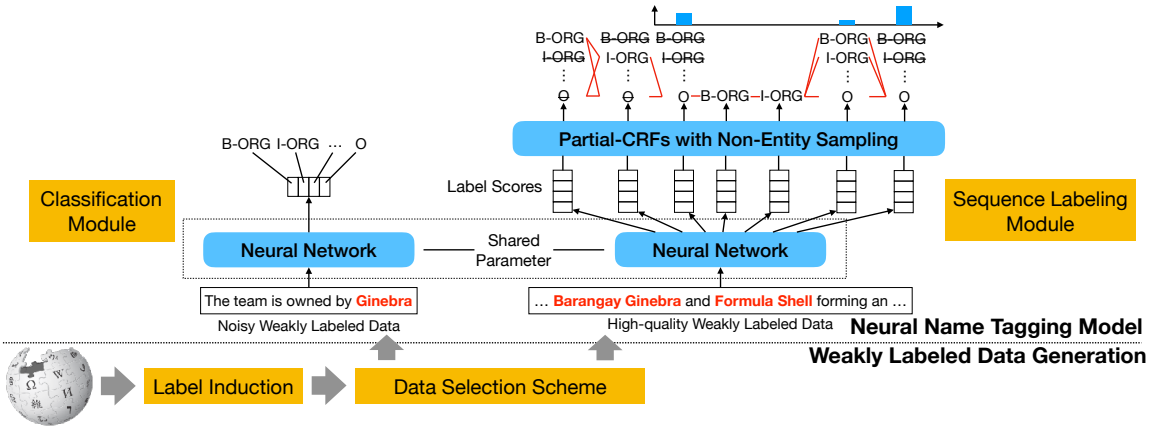


Figure 2: Framework. Rectangles denote the main components for two steps, and rounded rectangles consist of two modules of the neural model. In input sentences, bold fonts denote labeled words, and at the top is corresponding outputs. We use Partial-CRFs to model all possible label sequences (red paths from left to right by picking up one label per column) controlled by non-entity sampling (strikethrough labels according to the distribution). We replace "UN" and "x-NT" label with corresponding possible labels to clarify the principle of PCRF.

data, and adapt it to arbitrary low-resource languages or domains, which can be further improved by the above transfer-based methods.

## 3 Preliminaries and Framework

### 3.1 Preliminaries

We formally define the name tagging task as follows: given a sequence of words $X = \langle x_1, \cdots, x_i, \cdots, x_{|X|} \rangle$, it aims to infer a sequence of labels $Y = \langle y_1, \cdots, y_i, \cdots, y_{|X|} \rangle$, where $|X|$ is the length of the sequence, $y_i \in \mathcal{Y}$ is the label of the word $x_i$, each label consists of the boundary and type information, such as B-ORG indicating that the word is **B**egin of an **ORG**anization entity. To make notations consistent, we use $\tilde{\mathcal{Y}} = \mathcal{Y} \bigcup \{\text{UN,B-NT,I-NT}\}$ to denote the label set of WL data, where UN indicates that the word is unlabeled, and NT denote only the type is unlabeled. In other words, the word with UN may be any one of the label in $\mathcal{Y}$, and the word with NT may be any type. We define $\tilde{\mathcal{Y}}$ for notation clarity.

To deal with the issue of limited annotations, we construct WL data $\mathcal{D} = \{(X, \tilde{Y})\}$ based on Wikipedia anchors and taxonomy, where $\tilde{Y} = \langle \tilde{y}_1, \cdots, \tilde{y}_i, \cdots, \tilde{y}_{|X|} \rangle$ and $\tilde{y}_i \in \tilde{\mathcal{Y}}$. An **anchor** $\langle m, e \rangle \in \mathcal{A}$ links a mention $m$ to an entity $e \in E$, where $m$ contains one or several consecutive words of length $|m|$. Particularly, we define $\mathcal{A}(X)$ as the set of anchors in $X$. Most entities are mapped to hierarchically organized categories, namely **taxonomy** $\mathcal{T}$, which provides category information $C = \{c\}$. We define $C(e)$ as the category set of $e$, and $\mathcal{T}_{\downarrow}(c)$ as the children of $c$.

### 3.2 Framework

The goal of our method is to extract WL data from Wikipedia and use them as training corpora for name tagging. As shown in Figure 2, there are two steps in our framework:

**Weakly Labeled Data Generation** generates as many WL data as possible for higher tagging recall. It contains two components of label induction and data selection scheme. First, the label induction assigns each word a label based on Wikipedia anchors and taxonomy. Then, the data selection scheme computes the quality scores for the WL sentences by considering the coverage of mentions as well as the label confidence. According to the scores, we split the entire set into two parts: a small set of high-quality data for the sequence labeling module, and a large amount of noisy data for the classification module.

**Neural Name Tagging Model** aims at efficiently and robustly utilizing both high-quality and noisy WL data, ensuring satisfying tagging precision. It is to make best use of labeled words via the sequence labeling module and the classification module. More specifically, we pre-train the classification module to capture the textual context semantics from massive noisy data, and then the sequence labeling module further fine-tunes the shared neural networks using a Partial-CRFs layer with Non-Entity Sampling.

## 4 Weakly Labeled Data Generation

Existing methods use Wikipedia (Ni et al., 2017; Pan et al., 2017; Geiß et al., 2017) to train an extra

classifier to predict entity categories for name tagging training. Instead, we aim at lowering the requirements of additional resources in order to support more low-resource settings. We thus utilize a lightweight strategy to generate WL data including label induction and data selection scheme.

### 4.1 Label Induction

Given a sentence $X$ including anchors $\mathcal{A}(X)$ and taxonomy $\mathcal{T}$, we aim at inducing a label $\tilde{y} \in \tilde{\mathcal{Y}}$ for each word $x \in X$. Obviously, the words outside of anchors should be labeled with UN, indicating that it is unlabeled and could be O or unannotated mentions. For the words in an anchor $\langle m, e \rangle$, we label it according to the entity categories. For example, words *Formula* and *Shell* (Figure 1) in $s_2$ are labeled as B-ORG and I-ORG, respectively, because mention *Formula Shell* is linked to entity *Shell Turbo Chargers*, which belongs to category *Basketball teams*. We trace it along the taxonomy $\mathcal{T}$: *Basketball teams*→...→*Organizations*, and find that it is a child of *Organizations*. According to a manually defined mapping $\Gamma(\mathcal{Y}) \rightarrow C$ (e.g., $\Gamma(ORG)$ =*Organizations*), we denote all the classes and their children with the same type (e.g., ORG).

However, there are two minor issues. First, for the entities without category information $C(e) = \emptyset$, we label them as B-NT or I-NT, indicating that they have no type information. Second, for the entities referring to multiple categories, we induce labels that maximizes the conditional probability:

$$\underset{y^*}{\text{argmax}}\, p(y^*|C(e)) = \frac{\sum_{c \in C(e)} \mathbf{1}(c \in \mathcal{T}_{\downarrow}(\Gamma(y^*)))}{|C(e)|} \tag{1}$$

where $\mathbf{1}(\cdot)$ indicates 1 if holds true, otherwise 0.

By doing so, we obtain a set of WL sentences $\mathcal{D} = \{(X, \tilde{Y})\}$. However, the induction process may introduce incorrect boundaries and types of labels due to the crowdsourcing nature of source data. We thus design a data selection scheme to deal with the issues.

### 4.2 Data Selection Scheme

Following Ni et al. (2017), we compute quality scores for sentences to distinguish high-quality and noisy data from two aspects: the annotation confidence and the annotation coverage.

The annotation confidence measures the likelihood of the text spans being mentions (i.e., correctness of boundaries), and being assigned with the types. We define it as follows:

$$q(X, \tilde{Y}) = \frac{\sum\limits_{(x_i, \tilde{y}_i)} \mathbf{1}(\tilde{y}_i \in \mathcal{Y}) p(\tilde{y}_i|C(e)) p(C(e)|x_i)}{|X|} \tag{2}$$

where $p(C(e)|x_i)$ is the conditional probability of $x_i$ linking to an entity belong to category $C(e)$, we compute it based on its statistical frequency among Wikipedia anchors.

The annotation coverage measures to which ratio the words are being labeled in the sentence:

$$n(X, \tilde{Y}) = \frac{\sum\limits_{(x_i, \tilde{y}_i)} \mathbf{1}(\tilde{y}_i \in \mathcal{Y})}{|X|} \tag{3}$$

We select high-quality sentences $\mathcal{D}^{hq}$ satisfying:

$$q(X, \tilde{Y}) \geq \theta_q; n(X, \tilde{Y}) \geq \theta_n \tag{4}$$

where $\theta_q$ and $\theta_n$ are the hyperparameters. Thus, the remaining sentences are noisy $\mathcal{D}^{noise}$.

For example (Figure 2), the sentence ... *Barangay Ginebra and Formula Shell* ... is high-quality, and *The team is owned by Ginebra* is noisy. This is because there are more anchors that link *Formula Shell* to an organization entity and the anchors within the sentence account for a large proportion, leading to a higher quality score. Note that *Barangay* and *Ginebra* are labeled with B-NT and I-NT, indicating the type information is missing. Our model may learn the textual semantics for classifying *Ginebra* to ORG from the noisy sentence, where *Ginebra* is labeled with B-ORG.

## 5 Neural Name Tagging Model

Our neural model contains two modules that share the same NN architecture except the Partial-CRFs layer. Given $\mathcal{D}^{hq}$ and $\mathcal{D}^{noise}$, we first pre-train the classification module using massive noisy data $\mathcal{D}^{noise}$ to efficiently capture the textual semantics. Then, we use the sequence labeling module to fine-tune the classification module on high-quality data $\mathcal{D}^{hq}$ by considering the transitional constraints among sequential labels.

### 5.1 Sequence Labeling Module

Before describing the NN of the classification module, we first introduce the sequence labeling module. Different from conventional NN-CRFs

models, we utilize the Partial CRFs layer to maximize the probability of all possible sequential labels for the sentence with transitional constraints, where the probability of missing word labels is controlled by non-entity sampling.

**Partial-CRFs**

Partial-CRFs (PCRFs) was first proposed in the field of Part-of-Speech Tagging (Täckström et al., 2013). It can be trained when the coupled word and label constraints provide only a partial signal by assuming that the uncoupled words may refer to multiple labels. Given $(X, \tilde{Y})$, we traverse all possible labels $\mathcal{Y}$ for each unannotated word $\{x_i|\tilde{y}_i \in$ UN,B-NT,I-NT$\}$ (e.g., the red paths in Figure 2), and compute the total probability of possible fully labeled sentences $\mathcal{Y}(X, \tilde{Y}) = \{(X, Y)\}$:

$$p(\tilde{Y}|X) = \sum_{(X,Y)}^{\mathcal{Y}(X,\tilde{Y})} p(Y|X) \tag{5}$$

where $p(Y|X) = \text{softmax}(s(X, Y))$, the same as in CRFs, and the score function $s(X, Y)$ is:

$$s(X, Y) = \sum_{i=0}^{|X|} A_{y_i,y_{i+1}} + \sum_{i=1}^{|X|} P_{x_i,y_i} \tag{6}$$

where $P_{x_i,y_i}$ is the score indicating how possible $x_i$ is labeled with $y_i$, which is defined as the output of NN and will be detailed in the next section. $A_{y_i,y_{i+1}}$ is the transition score from label $y_i$ to $y_{i+1}$ that is learned in this layer.

Instead of the single correct label sequence in CRFs, the loss function of partial-CRFs is to minimize the negative log-probability of ground truth over all possible labeled sequences:

$$\mathcal{L} = -\sum_{(X,\tilde{Y})}^{\mathcal{D}^{nq}} \log p(\tilde{Y}|X) \tag{7}$$

**Non-entity Sampling**

A crucial drawback of using partial CRFs for WL sentences is that there are no words labeled with O (i.e., non-entity words) for training (Section 6.5).

To further alleviate the reliance on seed annotations, we introduce non-entity sampling that samples O from unlabeled words as follows:

$$p(y_i = \text{O}|x_i, \tilde{y}_i = \text{N}) = \frac{\alpha}{3}(\lambda_1 f_1 + \lambda_2(1 - f_2) + \lambda_3 f_3) \tag{8}$$

where $\alpha$ is non-entity ratio to balance how many unlabeled words are sampled as O, we set $\alpha = 0.9$ in experiments according to Augenstein et al. (2017). Weighting parameters satisfy $0 \leq \lambda_1, \lambda_2, \lambda_3 \leq 1$, and $f_1, f_2, f_3$ are feature scores. We define $f_1 = \mathbf{1}(x_i \text{ adjoins an entity})$, which implies that the words around a mention are possible to be O; $f_2$ is the ratio of the number of $x_i$ labeled with entities to its total occurrences, reflecting how frequent a word is in a mention; and $f_3 = tf * df$, where $tf$ is term frequency and $df$ is document frequency in Wikipedia articles.

As shown in Figure 2, three words *and*, *forming* and *an* are labeled with N since they are outside of anchors. During training, they should be regarded as all labels of $\mathcal{Y}$ in Partial CRFs, while we sample some of them as O words according to Equation 8. Thus, *and* and *an* are instead treated as O words, because they do not appear in any anchor, and are too general due to a high $f_3$ value.

### 5.2 Classification Module

To efficiently utilize the noisy WL sentences, this module regards name tagging as a multi-label classification problem. On one hand, it predicts each word's label separately, naturally addressing the issue of inconsecutive labels. On the other hand, we only focus on the labeled words, so that the module is robust to the noise since most noise arises from the unlabeled words, and enjoy an efficient training procedure.

Formally, given a noisy sentence $(X, \tilde{Y}) \in \mathcal{D}^{noise}$, we classify words $\{x_i|\tilde{y}_i \in \mathcal{Y}\}$ by capturing textual semantics within the context. Independently of languages and domains, we combine the character and word embeddings for each word, then feed them into an encoder layer to capture contextual information for the classification layer.

**Character and Word Embeddings**

As inputs, we introduce character information to enhance word representations to improve the robustness to morphological and misspelling noise following (Ma and Hovy, 2016). Concretely, we represent a word $\mathbf{x}$ by concatenating word embedding $\mathbf{w}$ and Convolutional Neural Networks (CNN) (LeCun et al., 1989) based character embedding $\mathbf{c}$, which is obtained through convolution operations over characters in a word followed by max pooling and drop out techniques.

**Encoder Layer**

Given an arbitrary length of sentence $X$, this component encodes the semantics of words as well as their compositionality into a low-dimensional vector space. The most common encoders are CNN, Long-Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Transformer (Vaswani et al., 2017). We use the bi-directional LSTM (Bi-LSTM) due to its superior performance. We discuss it in Section 6.2.

Bi-LSTM (Graves et al., 2013) has been widely used in modeling sequential words, so as to capture both past and future input features for a given word. It stacks a forward LSTM and a backward LSTM, so that the output of a word $x_i$ is $\mathbf{h}_i = [\overleftarrow{\mathbf{h}}_i; \overrightarrow{\mathbf{h}}_i]$, where $\overrightarrow{\mathbf{h}}_i = \text{LSTM}(X_{1:i})$ and $\overleftarrow{\mathbf{h}}_i = \text{LSTM}(X_{i:|X|})$.

**Classification Layer**

The classification layer makes independent labeling decisions for each word, so that we can only focus on labeled words, while robustly and efficiently skip the noisy unlabeled words.

In this layer, we estimate the score $P_{x_i,y_i}$ (Equation 6) for word $x_i$ being the label $y_i$. We use a fully connected layer followed by softmax to output a probability-like score:

$$P_{x_i,y_i} = Softmax(W\mathbf{h}_i + b) \tag{9}$$

where $W \in \mathbb{R}^{|\mathcal{Y}|}$. Note that we have no training instance for O words. Thus, we also use the non-entity sampling (Section 5.1). Given $(X, \tilde{Y}) \in \mathcal{D}^{noise}$, this module is trained to minimize cross-entropy of the predicted and ground truth:

$$\mathcal{L}_c = -\sum_{(X,\tilde{Y})}^{\mathcal{D}^{noise}} \mathbf{1}(\tilde{y}_i \in \mathcal{Y}) \tilde{y}_i \log P_{x_i,y_i} \tag{10}$$

### 5.3 Training and Inference

To distill the knowledge derived from noisy data, we first pre-train the classification module, then share the overall NN with the sequence labeling module. If we choose a loose threshold $\theta_p$ and $\theta_n$, there is no noisy data and our model shall degrade to the sequential model without the pre-trained classifier. When the threshold is strict, there is no high-quality data and our model will degrade to the classification module only (Section 6.4).

For inference, we use the sequence labeling module to predict the output label sequence with the largest score as in Equation 6.

## 6 Experiment

We verify our model using five low-resource languages and a specific domain. Furthermore, we investigate the impacts of the main components as well as hyperparameters in the ablation study.

### 6.1 Experiment Settings

**Datasets** Since most datasets on low-resource languages are not publicly available, we use Wikipedia data as the "ground truth" following Pan et al. (2017). Thus, we can test name tagging in low-resource languages as well as domains. We choose five languages: Welsh, Bengali, Yoruba, Mongolian and Egyptian Arabic (or CY, BN, YO, MN and ARZ for short), at different low-resource levels, and select 3 types: Person, Location and Organization. For food domain, we reorganized the entities in Wikipedia category "Food and drink" into 5 types: Drinks, Meat, Vegetables, Condiments and Breads, for name tagging, and extract sentences containing those entities from all English Wikipedia articles for as many data as possible.

| | Train | | Test | |
|---|---|---|---|---|
| | #Sent | #Ment | #Sent | # Ment |
| CY | 106,541 | 146,524 | 1,193 | 3,256 |
| BN | 66,915 | 127,932 | 870 | 2838 |
| YO | 36,548 | 10,405 | 77 | 232 |
| MN | 19,250 | 27,820 | 173 | 439 |
| ARZ | 18,700 | 28,928 | 195 | 377 |
| Food Domain | 27,798 | 32,155 | 207 | 253 |
| Drinks | 8,615 | 9,218 | 62 | 67 |
| Meat | 7,685 | 8,841 | 53 | 68 |
| Vegetables | 6,155 | 7,235 | 45 | 58 |
| Condiments | 3,737 | 4,084 | 27 | 30 |
| Breads | 2,515 | 2,777 | 24 | 30 |

Table 1: The statistics of weakly labeled dataset.

We use 20190120 Wikipedia dump for WL data construction, where the ratio of words in anchors to the whole sentence is nearly 0.12, 0.07, 0.14, 0.07 and 0.06 for languages CY, BN, YO, MN and ARZ, and 0.13 for food domain, demonstrating that unlabeled words are dominant. By heuristically setting $\theta_q = 0.1, \theta_n = 0.9$, we obtain 56,571, 16,718, 4,131, 8,332, 6,266, 11,297 high-quality and 49,970, 50,197, 32,417, 10,918, 12,434, 16,501 noisy WL sentences for language

| | CY | | | BN | | | YO | | | MN | | | ARZ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **P** | **R** | **F1** | **P** | **R** | **F1** | **P** | **R** | **F1** | **P** | **R** | **F1** | **P** | **R** | **F1** |
| CNN-CRFs | 84.4 | 76.2 | 80.1 | 92.0 | 89.1 | 90.5 | **80.9** | 68.9 | 74.4 | 87.3 | 85.5 | 86.3 | 88.6 | 86.7 | 87.6 |
| BiLSTM-CRFs | **86.0** | 77.8 | 81.6 | 93.3 | 91.5 | 92.3 | 74.1 | 68.9 | 71.3 | **89.0** | 85.5 | 87.1 | **89.5** | 88.5 | 89.0 |
| Trans-CRFs | 83.7 | 73.2 | 78.1 | 93.0 | 85.9 | 89.3 | 80.2 | 60.5 | 69.0 | 88.0 | 80.0 | 83.8 | 88.9 | 83.2 | 85.9 |
| BiLSTM-PCRFs | 85.2 | 79.6 | 82.3 | 91.2 | 92.7 | 91.9 | 68.1 | 70.2 | 69.1 | 82.5 | 91.2 | 86.6 | 84.0 | 90.7 | 87.1 |
| Ours | 82.8 | **82.5** | **82.6** | **93.4** | **93.5** | **93.4** | 73.5 | **76.8** | **75.1** | 86.9 | **93.6** | **90.1** | 87.7 | **91.5** | **89.5** |

Table 2: Performance (%) on low-resource languages.

CY, BN, YO, MN and ARZ, and food domain, respectively. For correctness, we then pick up test data of 25% sentences that has highest annotation confidence and exceed 0.3 coverage. We randomly choose 25% of high-quality data as validation for early stop, and the rest for training. The statistics[3] is in Table 1.

**Training Details**

For tuning of hyper-parameters, we set non-entity feature weights to $\lambda_1 = 0, \lambda_2 = 0.9, \lambda_3 = 0.1$ heuristically. We pre-train word embeddings using Glove (Pennington et al., 2014), and fine-tune embeddings during training. We set the dimension of words and characters as 100 and 30, respectively. We use 30 filter kernels, where each kernel has the size of 3 in character CNN, and dropout rate is set to 0.5. For bi-LSTM, the hidden state has 150 dimensions. The batch size is set to 32 and 64 for sequence labeling module and classification module. We adopt Adam with L2 regularization for optimization, and set the learning rate and weight decay to 0.001 and $1e^{-9}$.

**Baselines** Since most low-resource name tagging methods introduce external knowledge (Section 2), which has limited availability and is out of the scope for this paper, we arrive at two types of baselines from weakly supervised models:

**Typical NN-CRFs models** (Ni et al., 2017) by selecting high-quality WL data and regarding unlabeled words as O, which usually achieve very competitive results. NN denotes CNN, Transformer (Trans for short) or Bi-LSTM.

**NN-PCRFs model** (Yang et al., 2018; Shang et al., 2018). Although they achieves state-of-the-art performance, methods of this type are only evaluated in specific domains and require a small set of seed annotations or a domain dictionary. We thus carefully adapt them to low-resource languages and domains by selecting the highest-quality WL data ($\theta_n > 0.3$) as seeds[4].

[3]The statistics includes noisy data, which greatly increases the size but cannot be used for evaluation.

[4]We adopt the common part of their models related to handling weakly labeled data, removing the other parts that are specifically designed for domains, such as instance selector (Yang et al., 2018) which makes it worse since we have already selected the high-quality data.

## 6.2 Results on Low-Resource Languages

Table 2 shows the overall performance of our proposed model as well as the baseline methods (P and R denote Precision and Recall). We can see:

Our method consistently outperforms all baselines in five languages w.r.t F1, mainly because we greatly improve recall (2.7% to 9.34% on average) by taking best advantage of WL data and being robust to noise via two modules. As for the precision, partial-CRFs perform poorly compared with CRFs due to the uncertainty of unlabeled words, while our method alleviates this issue by introducing linguistic features in non-entity sampling. An exception occurs in CY, because it has the most training data, which may bring more accurate information than sampling. Actually, we can tune hyper-parameter non-entity ratio $\alpha$ to improve precision[5], more studies can be found in Section 6.5. Besides, the sampling technique can utilize more prior features if available, we leave it in future.

Among all encoders, Bi-LSTM has greater ability for feature abstraction and achieves the highest precision in most languages. An unexpected exception is Yoruba, where CNN achieves the higher performance. This indicates that the three encoders capture textual semantics from different perspectives, thus it is better to choose the encoder by considering the linguistic natures.

As for the impacts of resources, all the models perform the worst in Yoruba. Interestingly, we conclude that the performance for name tagging in low-resource languages doesn't depend entirely on the absolute number of mentions in the training data, but largely on the average number of annotations per sentence. For example, Bengali has 1.9 mentions per sentence and all methods achieve their best results, while the opposite is Welsh with

[5]In this table, we show the performance using the same hyper-parameters in different languages for fairness.

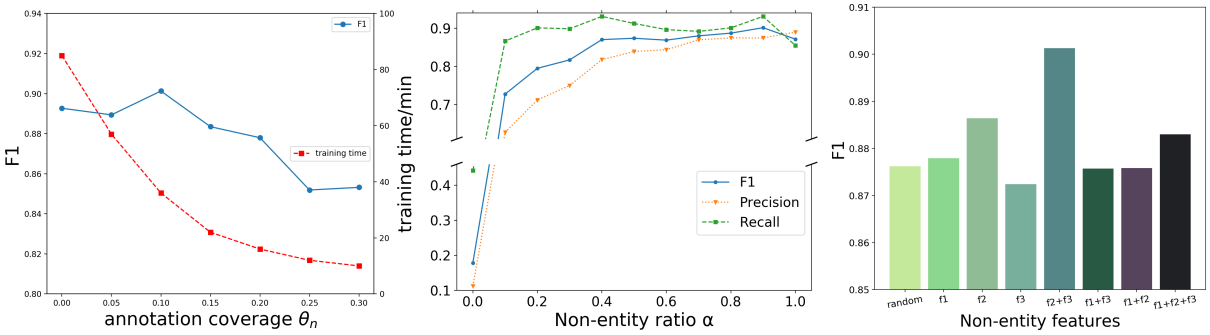


(a) Efficiency analysis. (b) Impact of non-entity sampling ratio. (c) Impact of non-entity features.

Figure 3: Ablation study of our model in Mongolian.

1.4 mentions per sentence. This verifies our data selection scheme (e.g., annotation coverage $n(\cdot)$), and we will give more discussion in Section 6.4.

### 6.3 Results on Low-Resource Domains

| | Food | | | | | |
|---|---|---|---|---|---|---|
| | **D** | **M** | **V** | **C** | **B** | **All** |
| CNN-CRFs | 67.8 | 69.8 | 57.9 | 42.8 | 46.5 | 60.9 |
| BiLSTM-CRFs | 64.9 | 69.0 | 62.8 | 50.0 | 62.2 | 63.5 |
| Trans-CRFs | 62.1 | 68.9 | 59.6 | 43.4 | 54.5 | 60.6 |
| BiLSTM-PCRFs | 66.1 | 70.7 | 67.2 | 44.4 | 58.3 | 64.4 |
| Ours | **72.0** | **76.4** | **70.0** | **53.5** | **66.6** | **70.1** |

Table 3: F1-score (%) on food domain.

Table 3 shows the overall performance in food domain, where D, M, V, C and B denote: Drink, Meat, Vegetables, Condiments and Breads. We can observe that there is a performance drop compared to that in low-resource languages, mainly because of more types and sparse training data. Our model outperforms all of the baselines in all food types by 7.8% on average. The performance in condiments is relatively low, because most of them are composed of meat or vegetables, such as steak sauce, which is overlapped with other types and make the recognition more difficult.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Sentence: | *Maize* | and | *yams* | are | grown | on | its | banks | . |
| Labels: | B-V | UN | UN | UN | UN | UN | UN | UN | UN |
| Predictions: | B-V | O | B-V | O | O | O | O | O | O |

Figure 4: Our predictions on a noisy WL sentence.

Here is a representative case demonstrating that our model is robust to noise induced by unlabeled words. In Figure 4, the sentence is from the noisy WL training data of food domain, and only *Maize* is labeled as B-V. Although our model is trained on this sentence, it successfully predicts *yams* as B-V. This example shows that our two-modules design can utilize the noisy data while avoiding side effects caused by incomplete annotation.

### 6.4 Efficiency Analysis

We utilize $\theta_n$, the main factor to annotation quality (Section 6.2), to trade off between high-quality and noisy WL data. As shown in Figure 3(a), the red curve denotes the training time and the blue curve denotes F1. We can see that the performance of our model is relatively stable when $\theta_n \in [0, 0.15)$, while the time cost drops dramatically (from 90 to 20 minutes), demonstrating the robustness and efficiency of two-modules design. When $\theta_n \in [0.15, 0.3]$, the performance decreases greatly due to less available high-quality data for sequence labeling module; meanwhile, little time is saved through classification module. Thus, we pick up $\theta_n = 0.1$ in experiments. A special case happens when $\theta_n = 0$, our model degrades to sequence labeling without pre-trained classifier. We can see the performance is worse than that of $\theta_n = 0.1$ due to massive noisy data.

### 6.5 Impact of Non-Entity Sampling Ratio

We use non-entity ratio $\alpha$ to control sampling, and a higher $\alpha$ denotes that more unlabeled words are labeled with O. As shown in Figure 3(b), the precision increases as more words are assigned with labels, while the recall achieves two peaks ($\alpha = 0.4, 0.9$), leading to the highest F1 when $\alpha = 0.9$, which conforms to the statistics in Augenstein et al. (2017). There are two special cases. When $\alpha = 0$, our model degrades to a NN-PCRFs model without non-entity sampling and there is no seed annotations for training. We can see the model performs poorly due to the dominant unlabeled words (Section 5.1). When $\alpha = 1$ indicating all unlabeled words are sampled as O, our model

degrades to NN-CRFs model, which has higher precision at the cost of recall. Clearly, the model suffers from the bias to O labeling.

### 6.6 Impact of Non-Entity Features

We propose three features for non-entity samples: nearby entities ($f_1$), ever within entities ($f_2$) and term/document frequency ($f_3$). We now investigate how effective each feature is. Figure 3(c) shows the performance of our model that samples non-entity words using each feature as well as their combinations. The first bar denotes the performance of sampling without any features. It is not satisfying but competitive, indicating the importance of non-entity sampling to partial-CRFs. The single $f_2$ contributes the most, and gets enhanced with $f_3$ because they provide complementary information. Surprisingly, $f_1$ seems better than $f_3$, but makes the model worse if we use it combined with $f_2, f_3$, thus we set $\lambda_1 = 0$.

## 7 Conclusions

In this paper, we propose a novel name tagging model that consists of two modules of sequence labeling and classification, which are combined via shared parameters. We automatically construct WL data from Wikipedia anchors and split them into high-quality and noisy portions for training each module. The sequence labeling module focuses on high-quality data and is costly due to the partial-CRFs layer with non-entity sampling, which models all possible label combinations. The classification module focuses on the annotated words in noisy data to pretrain the tag classifier efficiently. The experimental results in five low-resource languages and a specific domain demonstrate the effectiveness and efficiency.

In the future, we are interested in incorporating entity structural knowledge to enhance text representation (Cao et al., 2017, 2018b), or transfer learning (Sun et al., 2019) to deal with massive rare words and entities for low-resource name tagging, or introduce external knowledge for further improvement.

## Acknowledgments

NExT++ research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its IRC@SG Funding Initiative.

## References

Isabelle Augenstein, Leon Derczynski, and Kalina Bontcheva. 2017. Generalisation in named entity recognition: A quantitative analysis. *Computer Speech & Language*.

Yixin Cao, Lei Hou, Juanzi Li, and Zhiyuan Liu. 2018a. Neural collective entity linking. In *COLING*.

Yixin Cao, Lei Hou, Juanzi Li, Zhiyuan Liu, Chengjiang Li, Xu Chen, and Tiansi Dong. 2018b. Joint representation learning of cross-lingual words and entities via attentive distant supervision. In *EMNLP*.

Yixin Cao, Lifu Huang, Heng Ji, Xu Chen, and Juanzi Li. 2017. Bridge text and knowledge by learning multi-prototype entity mention embedding. In *ACL*.

Yixin Cao, Zhiyuan Liu, Chengjiang Li, Juanzi Li, and Tat-Seng Chua. 2019a. Multi-channel graph neural network for entity alignment. In *ACL*.

Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. 2019b. Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In *WWW*.

Jason Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *TACL*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *JMLR*.

Maud Ehrmann, Marco Turchi, and Ralf Steinberger. 2011. Building a multilingual named entity-annotated corpus using annotation projection. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*.

Xiaocheng Feng, Xiachong Feng, Bing Qin, Zhangyin Feng, and Ting Liu. 2018. Improving low resource named entity recognition using cross-lingual knowledge transfer. In *IJCAI*.

Alexander Fritzler, Varvara Logacheva, and Maksim Kretov. 2018. Few-shot classification in named entity recognition task. *arXiv preprint arXiv:1812.06158*.

Johanna Geiß, Andreas Spitz, and Michael Gertz. 2017. Neckar: a named entity classifier for wikidata. In *International Conference of the German Society for Computational Linguistics and Language Technology*. Springer.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*.

James Hammerton. 2003. Named entity recognition with long short-term memory. In *NAACL*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*.

Maximilian Hofer, Andrey Kormilitzin, Paul Goldberg, and Alejo Nevado-Holgado. 2018. Few-shot learning for named entity recognition in medical text. *arXiv preprint arXiv:1811.05468*.

Jun Kuang, Yixin Cao, Jianbing Zheng, Xiangnan He, Ming Gao, and Aoying Zhou. 2019. Improving neural relation extraction with implicit mutual relations. *arXiv preprint arXiv:1907.05333*.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *NAACL*.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. 1989. Backpropagation applied to handwritten zip code recognition. *Neural computation*.

Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2017. Neural relation extraction with multi-lingual attention. In *ACL*.

Ying Lin, Shengqi Yang, Veselin Stoyanov, and Heng Ji. 2018. A multi-lingual multi-task architecture for low-resource sequence labeling. In *ACL*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *ACL*.

Stephen Mayhew, Chen-Tse Tsai, and Dan Roth. 2017. Cheap translation for cross-lingual named entity recognition. In *EMNLP*.

Jian Ni, Georgiana Dinu, and Radu Florian. 2017. Weakly supervised cross-lingual named entity recognition via effective annotation and representation projection. In *ACL*.

Joel Nothman, James R Curran, and Tara Murphy. 2008. Transforming wikipedia into named entity training data. In *Proceedings of the Australasian Language Technology Association Workshop 2008*.

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017. Cross-lingual name tagging and linking for 282 languages. In *ACL*.

Nanyun Peng and Mark Dredze. 2016. Improving named entity recognition for chinese social media with word segmentation representation learning. In *ACL*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Jingbo Shang, Liyuan Liu, Xiaotao Gu, Xiang Ren, Teng Ren, and Jiawei Han. 2018. Learning named entity tagger using domain-specific dictionary. In *EMNLP*.

Qianru Sun, Yaoyao Liu, Tat-Seng Chua, and Bernt Schiele. 2019. Meta-transfer learning for few-shot learning. In *CVPR*.

Oscar Täckström, Dipanjan Das, Slav Petrov, Ryan McDonald, and Joakim Nivre. 2013. Token and type constraints for cross-lingual part-of-speech tagging. *TACL*.

Chen-Tse Tsai, Stephen Mayhew, and Dan Roth. 2016. Cross-lingual named entity recognition via wikification. In *CoNLL*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*.

Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. 2019. Explainable reasoning over knowledge graphs for recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5329–5336.

Jiateng Xie, Zhilin Yang, Graham Neubig, Noah A Smith, and Jaime Carbonell. 2018. Neural cross-lingual named entity recognition with minimal resources. In *EMNLP*.

Mingbin Xu, Hui Jiang, and Sedtawut Watcharawittayakul. 2017. A local detection approach for named entity recognition and mention detection. In *ACL*.

Yaosheng Yang, Wenliang Chen, Zhenghua Li, Zhengqiu He, and Min Zhang. 2018. Distantly supervised ner with partial annotation learning and reinforcement learning. In *COLING*.

Zhilin Yang, Ruslan Salakhutdinov, and William W Cohen. 2017. Transfer learning for sequence tagging with hierarchical recurrent networks. In *ICLR*.

Atefeh Zafarian, Ali Rokni, Shahram Khadivi, and Sonia Ghiasifard. 2015. Semi-supervised learning for named entity recognition using weakly labeled training data. In *AISP*.

Boliang Zhang, Xiaoman Pan, Tianlu Wang, Ashish Vaswani, Heng Ji, Kevin Knight, and Daniel Marcu. 2016. Name tagging for low-resource incident languages based on expectation-driven learning. In *NAACL*.

Jing Zhang, Yixin Cao, Lei Hou, Juanzi Li, and Hai-Tao Zheng. 2017. Xlink: An unsupervised bilingual entity linking system. In *CCL*.

# Learning Dynamic Context Augmentation for Global Entity Linking

**Xiyuan Yang**[1], **Xiaotao Gu**[2], **Sheng Lin**[1], **Siliang Tang**[1]*, **Yueting Zhuang**[1], **Fei Wu**[1], **Zhigang Chen**[3], **Guoping Hu**[3] **& Xiang Ren**[4]
[1]Zhejiang University, [2]University of Illinois at Urbana Champaign
[3]iFLYTEK Research, [4]University of Southern California
{yangxiyuan, shenglin, siliang, yzhuang, wufei}@zju.edu.cn,
xiaotao2@illinois.edu, {zgchen, gphu}@iflytek.com,
xiangren@usc.edu

## Abstract

Despite of the recent success of collective entity linking (EL) methods, these "global" inference methods may yield sub-optimal results when the "all-mention coherence" assumption breaks, and often suffer from high computational cost at the inference stage, due to the complex search space. In this paper, we propose a simple yet effective solution, called Dynamic Context Augmentation (DCA), for collective EL, which requires only one pass through the mentions in a document. DCA *sequentially* accumulates context information to make efficient, collective inference, and can cope with different local EL models as a plug-and-enhance module. We explore both supervised and reinforcement learning strategies for learning the DCA model. Extensive experiments[1] show the effectiveness of our model with different learning settings, base models, decision orders and attention mechanisms.

## 1 Introduction

Linking mentions of entities in text to knowledge base entries (i.e., entity linking, or EL) is critical to understanding and structuring text corpora. In general, EL is approached by first obtaining candidate entities for each mention, and then identifying the true referent among the candidate entities. Prior distribution and local contexts, either in the form of hand-crafted features (Ratinov et al., 2011; Shen et al., 2015) or dense embeddings (He et al., 2013; Nguyen et al., 2016; Francis-Landau et al., 2016), play key roles in distinguishing different candidates. However, in many cases, local features can be too sparse to provide sufficient information for disambiguation.

To alleviate this problem, various collective EL models have been proposed to globally optimize the inter-entity coherence between mentions in the same document (Hoffart et al., 2011; Cheng and Roth, 2013; Nguyen et al., 2014; Alhelbawy and Gaizauskas, 2014; Pershina et al., 2015). Despite of their success, existing global EL models try to optimize the entire linking configuration of all mentions, with extra assumptions of either all-mention coherence or pairwise coherence (Phan et al., 2018). Such assumptions are against human intuitions, as they imply that no inference can

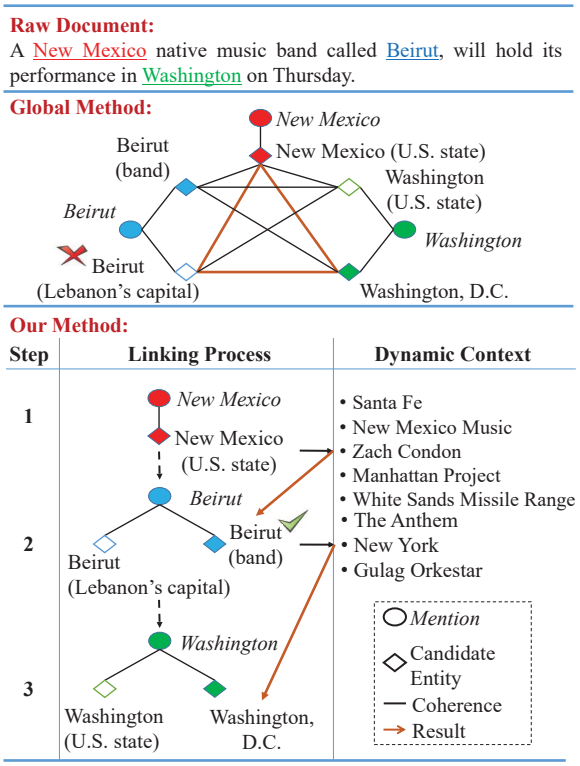


Figure 1: **An Illustration of the Dynamic Context Augmentation process.** A traditional global EL model jointly optimizes the linking configuration after iterative calculations over all mentions, which is computationally expensive. In contrast, the DCA process only requires one pass of the document to accumulate knowledge from previously linked mentions to enhance fast future inference.

*Corresponding author.
[1]Code and data is available at https://github.com/YoungXiyuan/DCA



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 271–281, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

be made until all mentions in a document have been observed. Also, there usually exists a trade-off between accuracy and efficiency: state-of-the-art collective/global models suffer from high time complexity. From the perspective of computational efficiency, optimal global configuration inference is NP-hard. Approximation methods, such as loopy belief propagation (Ganea and Hofmann, 2017) or iterative substitutions (Shen et al., 2015), are still computationally expensive due to the huge hypothesis space, and thus can hardly be scaled to handle large corpus. Many previous works have discussed the urgent needs of more efficient linking system for production, both in time complexity (Hughes et al., 2014) and memory consumption (Blanco et al., 2015).

In this paper, we propose a simple yet effective **D**ynamic **C**ontext **A**ugmentation (DCA) process to incorporate global signal for EL. As Figure 1 shows, in contrast to traditional global models, DCA only requires one pass through all mentions to achieve comparable linking accuracy. The basic idea is to accumulate knowledge from previously linked entities as dynamic context to enhance later decisions. Such knowledge come from not only the inherent properties (e.g., description, attributes) of previously linked entities, but also from their closely related entities, which empower the model with important associative abilities. In real scenarios, some previously linked entities may be irrelevant to the current mention. Some falsely linked entities may even introduce noise. To alleviate error propagation, we further explore two strategies: (1) soft/hard attention mechanisms that favour the most relevant entities; (2) a reinforcement learning-based ranking model, which proves to be effective as reported in other information extraction tasks.

**Contributions.** The DCA model forms a new linking strategy from the perspective of data augmentation and thus can serve as a plug-and-enhance module of existing linking models. The major contributions of this work are as follows: (1) DCA can introduce topical coherence into local linking models without reshaping their original designs or structures; (2) Comparing to global EL models, DCA only requires one pass through all mentions, yielding better efficiency in both training and inference; (3) Extensive experiments show the effectiveness of our model under different learning settings, base models, decision orders and attention mechanisms.

## 2 Background

### 2.1 Problem Definition

Given a set of entity mentions $\mathcal{M} = \{m_1, ..., m_T\}$ in corpus $\mathcal{D}$, Entity Linking aims to link each mention $m_t$ to its corresponding gold entity $e_t^*$. Such a process is usually divided into two steps: *Candidate generation* first collects a set of possible (candidate) entities $\mathcal{E}_t = \{e_t^1, ..., e_t^{|\mathcal{E}_t|}\}$ for $m_t$; *Candidate ranking* is then applied to rank all candidates by likelihood. The linking system selects the top ranked candidate as the predicted entity $\hat{e}_t$. The key challenge is to capture high-quality features of each entity mention for accurate entity prediction, especially when local contexts are too sparse to disambiguate all candidates.

We build our DCA model based on two existing local EL models. In this section, we first introduce the architecture of the base models, then present the proposed DCA model under the standard supervised learning framework. Since the DCA process can be naturally formed as a sequential decision problem, we also explore its effectiveness under the Reinforcement Learning framework. Detailed performance comparison and ablation studies are reported in Section 6.

### 2.2 Local Base Models for Entity Linking

We apply the DCA process in two popular local models with different styles: the first is a neural attention model named ETHZ-Attn (Ganea and Hofmann, 2017), the other is the Berkeley-CNN (Francis-Landau et al., 2016) model which is made up of multiple convolutional neural networks (CNN).

**ETHZ-Attn.** For each mention $m_t$ and a candidate $e_t^j \in \mathcal{E}_t$, three local features are considered: (1) *Mention-entity Prior* $\hat{P}(e_t^j|m_t)$ is the empirical distribution estimated from massive corpus (e.g.Wikipedia); (2) *Context Similarity* $\Psi_C(m_t, e_t^j)$ measures the textual similarity between $e_t^j$ and the local context of $m_t$; (3) *Type Similarity* $\Psi_T(m_t, e_t^j)$ considers the similarity between the type of $e_t^j$ and contexts around $m_t$. $\hat{P}(e_t^j|m_t)$ and $\Psi_C(m_t, e_t^j)$ are calculated in the same way as (Ganea and Hofmann, 2017). For $\Psi_T(m_t, e_t^j)$, we first train a typing system proposed by (Xu and Barbosa, 2018) on AIDA-train dataset, yielding 95% accuracy on AIDA-A dataset. In the testing phase, the typing system

predicts the probability distribution over all types (PER, GPE, ORG and UNK) for $m_t$, and outputs $\Psi_T(m_t, e_t^j)$ for each candidate accordingly. All local features are integrated by a two-layer feed-forward neural network with 100 hidden units, as described in (Ganea and Hofmann, 2017).

**Berkeley-CNN.** The only difference between **ETHZ-Attn** and **Berkeley-CNN** is that, this model utilizes CNNs at different granularities to capture *context similarity* $\Psi_C(m_t, e_t^j)$ between a mention's context and its target candidate entities.

## 3 Dynamic Context Augmentation

As Figure 1 demonstrates, the basic idea of DCA is to accumulate knowledge from previously linked entities as dynamic context to enhance later decisions. Formally, denote the list of previously linked entities as $S_t = \{\hat{e}_1, ..., \hat{e}_t\}$, where each $\hat{e}_i$ is represented as an embedding vector. The augmented context can be represented by accumulated features of all previous entities and their neighbors (e.g. by averaging their embeddings, in the simplest way). In actual scenarios, some entities in $S_t$ are irrelevant, if not harmful, to the linking result of $m_{t+1}$. To highlight the importance of relevant entities while filtering noises, we also try to apply a neural attention mechanism on dynamic contexts (Figure 2). For mention $m_{t+1}$, candidates that are more coherent with $S_t$ are preferred. More specifically, we calculate the relevance score for each $\hat{e}_i \in S_t$ as

$$u(\hat{e}_i) = \max_{e_{t+1}^j \in \mathcal{E}_{t+1}} {e_{t+1}^j}^\top \cdot A \cdot \hat{e}_i, \quad (1)$$

where $A$ is a parameterized diagonal matrix. Top $K$ entities in $S_t$ are left to form dynamic context while the others are pruned. The relevance scores are transformed to attention weights with

$$a(\hat{e}_i) = \frac{\exp[u(\hat{e}_i)]}{\sum_{\hat{e}_j \in S_t} \exp[u(\hat{e}_j)]}. \quad (2)$$

Thus, we can define a weighted coherence score between $e_{t+1}^j \in \mathcal{E}_{t+1}$ and $S_t$ as

$$\Phi(e_{t+1}^j, S_t) = \sum_{\hat{e}_i \in S_t} a(\hat{e}_i) \cdot {e_{t+1}^j}^\top \cdot R \cdot \hat{e}_i, \quad (3)$$

where $R$ is a learnable diagonal matrix. Such a coherence score will be later incorporated in the final representation of $e_{t+1}^j$.

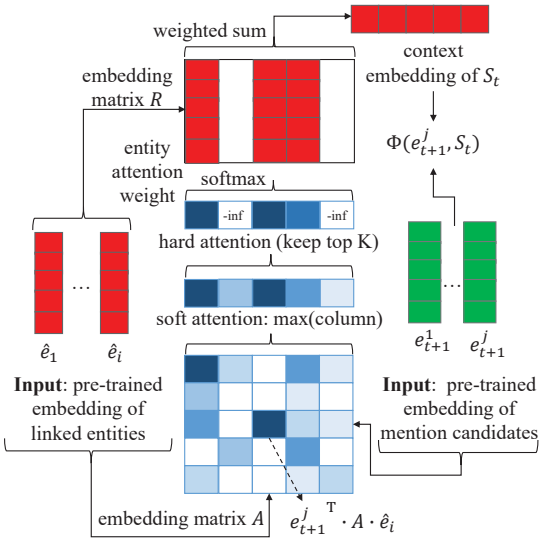


Figure 2: Neural attention mechanism on the dynamic context. The soft attention module assigns higher weights to entities that are more relevant to the target mention. The hard attention module only considers top $K$ entities as dynamic contexts.

To empower the linking model with associative ability, aside from previously linked entities, we also incorporate entities that are closely associated with entities in $S_t$. Specifically, for each $\hat{e}_i \in S_t$, we collect its neighborhood $\mathcal{N}(\hat{e}_i)$ consisting of Wikipedia entities that have inlinks pointing to $\hat{e}_i$. Denoting $S_t'$ as the union of $\{\mathcal{N}(\hat{e}_i)|\hat{e}_i \in S_t\}$, we define a similar weighted coherence score between $e_{t+1}^j \in \mathcal{E}_{t+1}$ and $S_t'$ as

$$\Phi'(e_{t+1}^j, S_t') = \sum_{\hat{e}_i \in S_t'} a'(\hat{e}_i) \cdot {e_{t+1}^j}^\top \cdot R' \cdot \hat{e}_i, \quad (4)$$

where $a'$ is defined similarly to $a$, and $R'$ is a learnable diagonal matrix. The final representation $\vec{h}_0(m_{t+1}, e_{t+1}^j)$ is the concatenation of $\Phi(e_{t+1}^j, S_t)$, $\Phi'(e_{t+1}^j, S_t')$, $\Psi_T(m_t, e_{t+1}^j)$, $\Psi_C(m_t, e_{t+1}^j)$ and $\log \hat{P}(e_{t+1}^j|m_{t+1})$.

## 4 Model Learning for DCA

In this section, we explore different learning strategies for the linking model. Specifically, we present a Supervised Learning model, where the model is given all gold entities for training, and a Reinforcement Learning model, where the model explores possible linking results by itself in a long-term planning task.

### 4.1 Supervised Ranking Method

Given a mention-candidate pair $(m_t, e_t^j)$, the ranking model parameterized by $\theta$ accepts the feature

vector $\vec{h_0}(m_t, e_t^j)$ as input, and outputs the probability $P_\theta(e_t^j|m_t)$. In this work, we use a two-layer feedforward neural network as the ranking model. We apply the max-margin loss as

$$\mathcal{L}_\theta = \sum_{D\in\mathcal{D}} \sum_{m_t\in D} \sum_{e_t^j\in\mathcal{E}_t} g_\theta(e_t^j, m_t),$$

$$g_\theta(e_t^j, m_t) = \max(0, \gamma - P_\theta(e_t^*|m_t) + P_\theta(e_t^j|m_t)).$$

The learning process is to estimate the optimal parameter such that $\theta^* = \arg\min_\theta \mathcal{L}_\theta$.

Note that, in the Supervised Ranking model, dynamic contexts are provided by previous gold entities: $S_t = \{e_1^*, ..., e_t^*\}$, $S'_t = \bigcup_{i=1}^{t} \mathcal{N}(e_i^*)$. In the testing phase, however, we do not have access to gold entities. Wrongly linked entities can introduce noisy contexts to future linking steps. To consider such long-term influences, we introduce an alternative Reinforcement Learning model in the next section.

### 4.2 Reinforcement Learning Method

Naturally, the usage of dynamic context augmentation forms a sequential decision problem, as each linking step depends on previous linking decisions. Correct linking results provide valuable information for future decisions, while previous mistakes can lead to error accumulation. Reinforcement Learning (RL) algorithms have proven to be able to alleviate such accumulated noises in the decision sequence in many recent works (Narasimhan et al., 2016; Feng et al., 2018). In this work, we propose an RL ranking model for DCA-enhanced entity linking.

**Agent**: The Agent is a candidate ranking model that has a similar architecture to (Clark and Manning, 2016), aiming to output the action preference $H_\theta(S_{t-1}, S'_{t-1}, A_t^j)$ of each linking action $A_t^j = (m_t \rightarrow e_t^j)$. It is a 2-layer feedforward neural network with following components:

**Input Layer**: For each $(m_t, e_t^j)$ pair, DCA-RL extracts context-dependent features from $S_{t-1}, S'_{t-1}$, and concatenates them with other context-independent features to produce an $I$-dimensional input vector $\vec{h_0}(m_t, e_t^j)$.

**Hidden Layers**: Let $Drop(\vec{x})$ be the dropout operation (Srivastava et al., 2014) and $ReLU(\vec{x})$ be the rectifier nonlinearity (Nair and Hinton, 2010). So the output $\vec{h}_1$ of the hidden layer is defined as:

$$\vec{h}_1 = Drop(ReLU(\vec{W}_1 \cdot \vec{h}_0 + \vec{b}_1)), \quad (5)$$

where $\vec{W}_1$ is a $H_1 \times I$ weight matrix.

**Output Layers**: This scoring layer is also fully connected layer of size 1.

$$\vec{h}_2 = \vec{W}_2 \cdot \vec{h}_1 + \vec{b}_2, \quad (6)$$

where $\vec{W}_2$ is a $1 \times H_1$ weight matrix. In the end, all action preference would be normalized together using an exponential softmax distribution, getting their action probabilities $\pi_\theta(A_t^j|S_{t-1}, S'_{t-1})$:

According to policy approximating methods, the best approximate policy may be stochastic. So we randomly sample the actions based on the softmax distribution during the training time, whereas deliberately select the actions with the highest ranking score at the test time.

**Reward.** The reward signals are quite sparse in our framework. For each trajectory, the Agent can only receive a reward signal after it finishes all the linking actions in a given document. Therefore the immediate reward of action $t$, $R_t = 0$, where $0 \leq t < T$, and $R_T = -(|\mathbb{M}_e|/T)$, where $T$ is total number of mentions in the source document, and $|\mathbb{M}_e|$ is the number of incorrectly linked mentions. Then the value $G_t$ (expected reward) of each previous state $S_t$ can be retraced back with a discount factor $\rho$ according to $R_T$:

$$G_t = \rho^{T-t} R_T \quad (7)$$

To maximize the expected reward of all trajectories, the Agent utilizes the REINFORCE algorithm (Sutton and Barto, 1998) to compute Monte Carlo policy gradient over all trajectories, and perform gradient ascent on its parameters:

$$\theta \leftarrow \theta + \alpha \sum_t G_t \nabla_\theta \ln \pi_\theta(A_t^j|S_{t-1}, S'_{t-1}) \quad (8)$$

In following sections, to fully investigate the effectiveness of the proposed method, we report and compare the performances of both the Supervised-learning model and the Reinforcement-learning model.

## 5 Analysis of Computational Complexity

For each document $D$, the train and inference of the global EL models are heavily relied on the inter-entity coherence graph $\Phi_g$. Many studies (Ratinov et al., 2011; Globerson et al., 2016; Yamada et al., 2016; Ganea and Hofmann, 2017; Le and Titov, 2018) obtain $\Phi_g$ by calculating all

| Dataset | # mention | # doc | Mentions per doc | Gold recall |
|---|---|---|---|---|
| AIDA-train | 18448 | 946 | 19.5 | - |
| AIDA-A | 4791 | 216 | 22.1 | 97.3 |
| AIDA-B | 4485 | 231 | 19.4 | 98.3 |
| MSNBC | 656 | 20 | 32.8 | 98.5 |
| AQUAINT | 727 | 50 | 14.5 | 94.2 |
| ACE2004 | 257 | 36 | 7.1 | 90.6 |
| CWEB | 11154 | 320 | 34.8 | 91.1 |
| WIKI | 6821 | 320 | 21.3 | 92.4 |

Table 1: **Dataset Statistics.** *Gold recall* is the percentage of mentions for which the candidate entities contain the ground truth entity.

pairwise scores between two arbitrary elements ${e_i}^x$ and ${e_j}^y$ sampled independently from candidate sets $\mathcal{E}_i$ and $\mathcal{E}_j$ in the given document. It is obvious that $\Phi$ is intractable, and the computational complexity of $\Phi_g$ is

$$\mathcal{O}(\Phi_g) = \mathcal{O}(\sum_{i=1}^{T} \sum_{j=1, j\neq i}^{T} \sum_{{e_i}^x \in \mathcal{E}_i}^{|\mathcal{E}_i|} \sum_{{e_j}^y \in \mathcal{E}_j}^{|\mathcal{E}_j|} \Phi({e_i}^x, {e_j}^y)) \tag{9}$$

, where $\Phi({e_i}^x, {e_j}^y)$ is a learnable score function. Thus, $\mathcal{O}(\Phi_g)$ is approximate to $\mathcal{O}(T^2 \times |\mathcal{E}|^2 \times \mathcal{I})$, where $|\mathcal{E}|$ is the average number of candidates per mention and $\mathcal{I}$ is the unit cost of pairwise function $\Phi$. In order to reduce $\mathcal{O}(\Phi_g)$, most previous models (Hoffart et al., 2011; Ganea and Hofmann, 2017; Le and Titov, 2018; Fang et al., 2019) have to hard prune their candidates into an extremely small size (e.g. $|\mathcal{E}|$=5). This will reduce the gold recall of candidate sets and also unsuitable for large scale production (e.g. entity disambiguation for dynamic web data like Twitter).

In contrast, the computational complexity of our model is $\mathcal{O}(T \times |\mathcal{E}| \times \mathcal{I} \times K)$, where $K$ is the key hyper-parameter described in Section 3 and is usually set to a small number. This indicates the response time of our method grow linearly as a function of $T \times |\mathcal{E}|$.

## 6 Experiment

### 6.1 Experiment Setup

**Datasets.** Following our predecessors, we train and test all models on the public and widely used AIDA CoNLL-YAGO dataset (Hoffart et al., 2011). The target knowledge base is Wikipedia. The corpus consists of 946 documents for training, 216 documents for development and 231 documents for testing (AIDA-train/A/B respectively).

| System | In-KB acc. (%) |
|---|---|
| AIDA-light (Nguyen et al., 2014) | 84.8 |
| WNED (Guo and Barbosa, 2016) | 89.0 |
| Global-RNN (Nguyen et al., 2016) | 90.7 |
| MulFocal-Att (Globerson et al., 2016) | 91.0 |
| Deep-ED (Ganea and Hofmann, 2017) | 92.22 |
| Ment-Norm (Le and Titov, 2018) | 93.07 |
| Prior ($p(e\|m)$) (Ganea and Hofmann, 2017) | 71.51 |
| *Berkeley-CNN (Sec. 2.2)* | 84.21 |
| *Berkeley-CNN + DCA-SL* | 92.72 $\pm$ 0.3 |
| *Berkeley-CNN + DCA-RL* | 92.37 $\pm$ 0.1 |
| *ETHZ-Attn (Sec. 2.2)* | 90.88 |
| *ETHZ-Attn + DCA-SL* | **94.64** $\pm$ 0.2 |
| *ETHZ-Attn + DCA-RL* | 93.73 $\pm$ 0.2 |

Table 2: **In-domain Performance Comparison on the AIDA-B Dataset.** For our method we show 95% confidence intervals obtained over 5 runs. DCA-based models achieve the best reported scores on this benchmark.

To evaluate the generalization ability of each model, we apply cross-domain experiments following the same setting in (Ganea and Hofmann, 2017; Le and Titov, 2018; Yang et al., 2018). Models are trained on AIDA-train, and evaluated on five popular public datasets: AQUAINT (Milne and Witten, 2008), MSNBC (Cucerzan, 2007), ACE2004 (Ratinov et al., 2011), CWEB (Guo and Barbosa, 2016) and WIKI (Guo and Barbosa, 2016). The statistics of these datasets are available in Table 1. In the candidate generation step, we directly use the candidates provided by the Ment-Norm system (Le and Titov, 2018)[2], and their quality is also listed in Table 1.

**Compared Methods.** We compare our methods with following existing systems that report state-of-the-art results on the test datasets: **AIDA-light** (Nguyen et al., 2014) uses a kind of two-stage collective mapping algorithm and designs several domain or category related coherence features. **WNED** (Guo and Barbosa, 2016) applies random walks on carefully built disambiguation graphs and uses a greedy, iterative and global disambiguation algorithm based on Information Theory. **Global-RNN** (Nguyen et al., 2016) develops a framework based on convolutional neural networks and recurrent neural networks to simultaneously model the local and global features. **MulFocal-Att** (Globerson et al., 2016) adopts a coherence model with a multi-focal attention mechanism. **Deep-ED** (Ganea and Hofmann, 2017) leverages learned neural representa-

[2]https://github.com/lephong/mulrel-nel

| System | MSBNC | AQUAINT | ACE2004 | CWEB | WIKI |
|---|---|---|---|---|---|
| AIDA (Hoffart et al., 2011) | 79 | 56 | 80 | 58.6 | 63 |
| GLOW (Ratinov et al., 2011) | 75 | 83 | 82 | 56.2 | 67.2 |
| RI (Cheng and Roth, 2013) | 90 | **90** | 86 | 67.5 | 73.4 |
| WNED (Guo and Barbosa, 2016) | 92 | 87 | 88 | 77 | **84.5** |
| Deep-ED (Ganea and Hofmann, 2017) | 93.7 | 88.5 | 88.5 | **77.9** | 77.5 |
| Ment-Norm (Le and Titov, 2018) | 93.9 | 88.3 | 89.9 | 77.5 | 78.0 |
| Prior ($p(e\|m)$) (Ganea and Hofmann, 2017) | 89.3 | 83.2 | 84.4 | 69.8 | 64.2 |
| *Berkeley-CNN (Section 2.2)* | 89.05 | 80.55 | 87.32 | 67.97 | 60.27 |
| *Berkeley-CNN + DCA-SL* | 93.38 $\pm$ 0.2 | 85.63 $\pm$ 0.3 | 88.73 $\pm$ 0.3 | 71.01 $\pm$ 0.1 | 72.55 $\pm$ 0.2 |
| *Berkeley-CNN + DCA-RL* | 93.65 $\pm$ 0.2 | 88.53 $\pm$ 0.3 | 89.73 $\pm$ 0.4 | 72.66 $\pm$ 0.4 | 73.98 $\pm$ 0.2 |
| *ETHZ-Attn (Section 2.2)* | 91.97 | 84.06 | 86.92 | 70.07 | 74.37 |
| *ETHZ-Attn + DCA-SL* | **94.57** $\pm$ 0.2 | 87.38 $\pm$ 0.5 | 89.44 $\pm$ 0.4 | 73.47 $\pm$ 0.1 | 78.16 $\pm$ 0.1 |
| *ETHZ-Attn + DCA-RL* | 93.80 $\pm$ 0.0 | 88.25 $\pm$ 0.4 | **90.14** $\pm$ 0.0 | 75.59 $\pm$ 0.3 | 78.84 $\pm$ 0.2 |

Table 3: **Performance Comparison on Cross-domain Datasets using F1 score (%)**. The best results are in bold. Note that our own results all retain two decimal places. Other results with uncertain amount of decimal places are directly retrieved from their original paper.

tions, and uses a deep learning model combined with a neural attention mechanism and graphical models. **Ment-Norm** (Le and Titov, 2018) improving the **Deep-ED** model by modeling latent relations between mentions.

For a fair comparison with prior work, we use the same input as the **WNED**, **Deep-ED** and **Ment-Norm** (models proposed after 2016), and report the performance of our model with both Supervised Learning (**DCA-SL**) and Reinforcement Learning (**DCA-RL**). We won't compare our models with the **RLEL** (Fang et al., 2019) which is a deep reinforcement learning based LSTM model. There are two reasons: 1) **RLEL** uses optimized candidate sets with smaller candidate size and higher gold recall than ours and the listed baselines. 2) **RLEL** uses additional training set from Wikipedia data. (Fang et al., 2019) doesn't release either their candidate sets or updated training corpus, so the comparison with their work would be unfair for us.

**Hyper-parameter Setting.** We coarsely tune the hyper-parameters according to model performance on AIDA-A. We set the dimensions of word embedding and entity embedding to 300, where the word embedding and entity embedding are publicly released by (Pennington et al., 2014) and (Ganea and Hofmann, 2017) respectively. Hyper-parameters of the best validated model are: $K = 7$, $I = 5$, $H_1 = 100$, and the probability of dropout is set to 0.2. Besides, the rank margin $\gamma = 0.01$ and the discount factor $\rho = 0.9$. We also regularize the *Agent* model as adopted in (Ganea and Hofmann, 2017) by constraining the sum of squares of all weights in the linear layer with $MaxNorm = 4$. When training the model, we use Adam (Kingma and Ba, 2014) with learning rate of 2e-4 until validation accuracy exceeds 92.8%, afterwards setting it to 5e-5.

| System | In-KB acc. (%) | |
|---|---|---|
| | SL | RL |
| ETHZ-Attn (Section 2.2) | 90.88 | - |
| ETHZ-Attn + 1-hop DCA | 93.69 | 93.20 |
| ETHZ-Attn + 2-hop DCA | 94.47 | 93.76 |

Table 4: **Ablation Study on Neighbor Entities.** We compare the performance of DCA with or without neighbor entities (i.e., 2-hop vs. 1-hop).

### 6.2 Overall Performance Comparison

Starting with an overview of the end-task performance, we compare DCA (using SL or RL) with several state-of-the-art systems on in-domain and cross-domain datasets. We follow prior work and report in-KB accuracy for AIDA-B and micro F1 scores for the other test sets.

Table 2 summarizes results on the AIDA-B dataset, and shows that DCA-based models achieve the highest in-KB accuracy and outperforms the previous state-of-the-art neural system by near 1.6% absolute accuracy. Moreover, compared with the base models, dynamic context augmentation significantly improve absolute in-KB accuracy in models Berkeley-CNN (more than

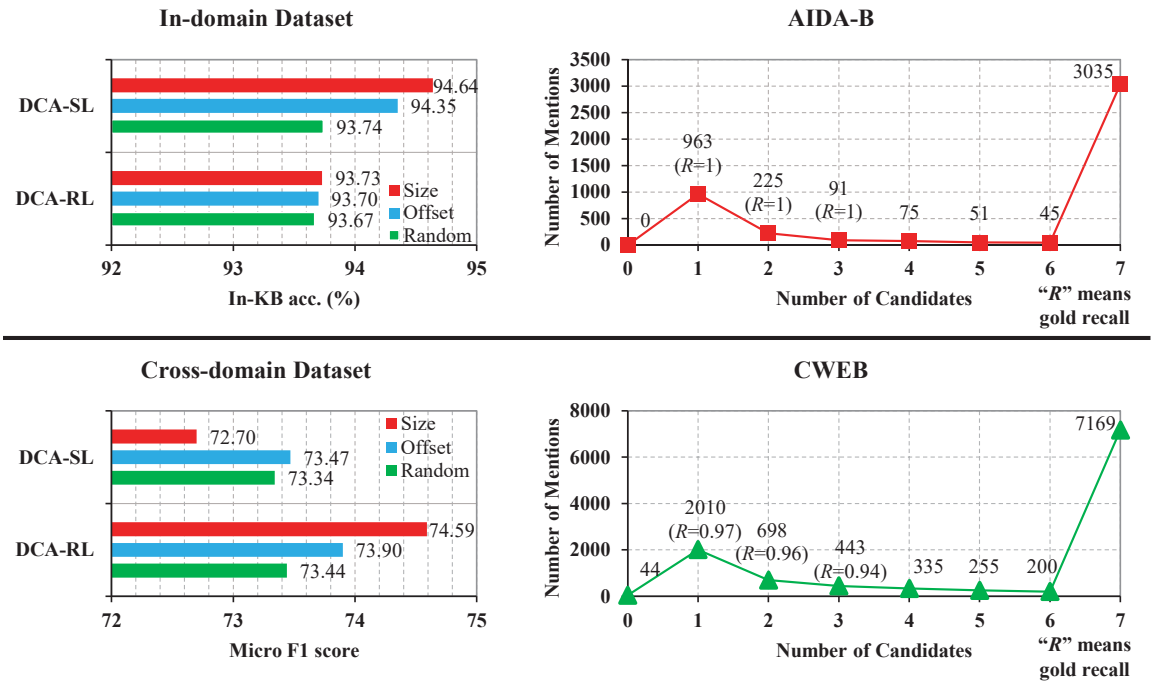


Figure 3: **Ablation Study on Different Decision Orders.** We test on both in-domain (AIDA-B) and cross-domain (CWEB) datasets, using ETHZ-Attn as the local model.

8%) and ETHZ-Attn (3.3% on average). Note that, our DCA model outperforms existing global models with the same local model (**Global-RNN** uses Berkeley-CNN as base model, **Deep-ED** and **Ment-Norm** use ETHZ-Attn as the local model).

Table 3 shows the results on the five cross-domain datasets. As shown, none of existing methods can consistently win on all datasets. DCA-based models achieve state-of-the-art performance on the MSBNC and the ACE2004 dataset. On remaining datasets, DCA-RL achieves comparable performance with other complex global models. In addition, RL-based models show on average 1.1% improvement on F1 score over the SL-based models across all the cross-domain datasets. At the same time, DCA-based methods are much more efficient, both in time complexity and in resource requirement. Detailed efficiency analysis will be presented in following sections.

### 6.3 Performance Analysis

**1. Impact of decision order.** As the DCA model consecutively links and adds all the mentions in a document, the linking order may play a key role in the final performance. In this work, we try three different linking orders: *Offset* links all mentions by their natural orders in the original document;

| System | In-KB acc. (%) | |
|---|---|---|
| | SL | RL |
| DCA + Average Sum | 94.05 | 93.56 |
| DCA + Soft Attention | 94.23 | 93.69 |
| DCA + Soft&Hard Attention | 94.47 | 93.76 |

Table 5: **Study on Different Attention Mechanisms on the Dynamic Context.** Results (using ETHZ-Attn as local model) show that different attention mechanisms have similar impact on the performance.

*Size* first links mentions with smaller candidate sizes, as they tend to be easier to link; The baseline method is to link all mentions in a *Random* order. Figure 3 shows the performance comparison on the AIDA-B and the CWEB dataset. As shown, in general, *Size* usually leads to better performance than *Offset* and *Random*. However, the DCA-SL model shows poor performance on the CWEB dataset with *Size* order. This is mainly because the CWEB dataset is automatically generated rather than curated by human, and thus contains many noisy mentions. Some mentions in CWEB with less than three candidates are actually bad cases , where none of the candidates is the actual gold entity. Thus, such mentions will always introduce wrong information to the model, which leads to a worse performance. In contrast, the AIDA-B dataset does not have such situations.

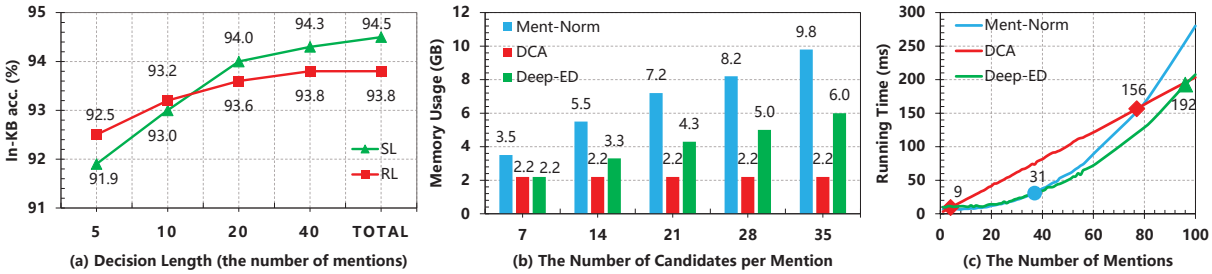


Figure 4: (a) **In-KB accuracy** as a function of decision length on AIDA-B dataset; (b) **Memory usage** as a function of the number of candidates on AIDA-B dataset; (c) **Runtime cost** (at inference time) as a function of the number of mentions. (each dot represents a document of AIDA-B with $|\mathcal{E}| = 35$).

The DCA-RL model, however, still has strong performance on the CWEB dataset, which highlights its robustness to potential noises.

**2. Effect of neighbor entities.** In contrast to traditional global models, we include both previously linked entities and their close neighbors for global signal. Table 4 shows the effectiveness of this strategy. We observe that incorporating these neighbor entities (2-hop) significantly improve the performance (compared to 1-hop) by introducing more related information. And our analysis shows that on average 0.72% and 3.56% relative improvement of 2-hop DCA-(SL/RL) over 1-hop DCA-(SL/RL) or baseline-SL (without DCA) is statistically significant (with P-value $<$ 0.005). This is consistent with our design of DCA.

**3. Study of different attention mechanisms.** Table 5 shows the performance comparison by replacing the attention module described in Section 3 with different variants. *Average Sum* treats all previously linked entities equally with a uniform distribution. *Soft Attention* skips the pruning step for entities with low weight scores. *Soft&Hard Attention* stands for the strategy used in our model. It is obvious that the attention mechanism does show positive influence on the linking performance compared with *Average Sum*. Hard pruning brings slight further improvement.

**4. Impact of decision length.** As wrongly linked entities can introduce noise to the model, there exists a trade-off in DCA: involving more previous entities (longer historical trajectory) provides more information, and also more noise. Figure (4.a) shows how the performance of DCA changes with the number of previous entities involved. We observe that longer historical trajectories usually have a positive influence on the performance of DCA. The reason is that our attention mechanism could effectively assess and select relevant contexts for each entity mention on the fly, thus reducing potential noise.

### 6.4 Analysis on Time Complexity

As discussed in Sec. 5, the running time of a DCA enhanced model may rise linearly when the average number of candidates per mention (i.e., $|\mathcal{E}|$) increases, while the global EL model increases exponentially. To validate the theory we empirically investigate the scalability of DCA, and carefully select two global EL models Ment-Norm (Le and Titov, 2018) and Deep-ED (Ganea and Hofmann, 2017) as our baselines. The reason for this choice is that our final model shares the same local model as their models, which excludes other confounding factors like implementation details. As Figure (4.c) shows, when $|\mathcal{E}|$ increases, the running time of these two global EL models increases shapely, while our DCA model grows linearly. On the other hand, we also observed that the resources required by the DCA model are insensitive to $|\mathcal{E}|$. For example, as shown in Figure (4.b), the memory usage of Ment-Norm and Deep-ED significantly rises as more candidates are considered, while the DCA model remains a relatively low memory usage all the time. We also measure the power consumption of Ment-Norm and DCA models, and we find that the DCA model saves up to 80% of the energy consumption over the Ment-Norm, which is another advantage for large scale production.

## 7 Related Work

Local EL methods disambiguate each mention independently according to their local contexts (Yamada et al., 2016; Chen et al., 2017; Globerson et al., 2016; Raiman and Raiman, 2018). The per-

formance is limited when sparse local contexts fail to provide sufficient disambiguation evidence.

To alleviate this problem, global EL models jointly optimize the entire linking configuration. The key idea is to maximize a global coherence/similarity score between all linked entities (Hoffart et al., 2011; Ratinov et al., 2011; Cheng and Roth, 2013; Nguyen et al., 2014; Alhelbawy and Gaizauskas, 2014; Pershina et al., 2015; Guo and Barbosa, 2016; Globerson et al., 2016; Ganea and Hofmann, 2017; Le and Titov, 2018; Yang et al., 2018; Fang et al., 2019; Xue et al., 2019). Despite of its significant improvement in accuracy, such global methods suffer from high complexity. To this end, some works try to relax the assumption of all-mention coherence, e.g. with pairwise coherence, to improve efficiency (Phan et al., 2018), but exact inference remains an NP-hard problem. Approximation methods are hence proposed to achieve reasonably good results with less cost. (Shen et al., 2012) propose the iterative substitution method to greedily substitute linking assignment of one mention at a time that can improve the global objective. Another common practice is to use Loopy Belief Propagation for inference (Ganea and Hofmann, 2017; Le and Titov, 2018). Both approximation methods iteratively improve the global assignment, but are still computationally expensive with unbounded number of iterations. In contrast, the proposed DCA method only requires one pass through the document. Global signals are accumulated as dynamic contexts for local decisions, which significantly reduces computational complexity and memory consumption.

## 8 Conclusions

In this paper we propose Dynamic Context Augmentation as a plug-and-enhance module for local Entity Linking models. In contrast to existing global EL models, DCA only requires one pass through the document. To incorporate global disambiguation signals, DCA accumulates knowledge from previously linked entities for fast inference. Extensive experiments on several public benchmarks with different learning settings, base models, decision orders and attention mechanisms demonstrate both the effectiveness and efficiency of DCA-based models. The scalability of DCA-based models make it possible to handle large-scale data with long documents. Related code and data has been published and may hopefully benefit the community.

## Acknowledgments

This work has been supported in part by National Key Research and Development Program of China (SQ2018AAA010010), NSFC (No.61751209, U1611461), Zhejiang University-iFLYTEK Joint Research Center, Zhejiang University-Tongdun Technology Joint Laboratory of Artificial Intelligence, Hikvision-Zhejiang University Joint Research Center, Chinese Knowledge Center of Engineering Science and Technology (CKCEST), Engineering Research Center of Digital Library, Ministry of Education. Xiang Ren's research has been supported in part by National Science Foundation SMA 18-29268, DARPA MCS and GAILA, IARPA BETTER, Schmidt Family Foundation, Amazon Faculty Award, Google Research Award, Snapchat Gift and JP Morgan AI Research Award. Finally, we would like to thank Qi Zhang from Sogou, Inc. and all the collaborators in INK research lab for their constructive feedback on this work.

## References

Ayman Alhelbawy and Robert Gaizauskas. 2014. Graph ranking for collective named entity disambiguation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 75–80.

Roi Blanco, Giuseppe Ottaviano, and Edgar Meij. 2015. Fast and space-efficient entity linking for queries. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, pages 179–188. ACM.

Hui Chen, Baogang Wei, Yonghuai Liu, Yiming Li, Jifang Yu, and Wenhao Zhu. 2017. Bilinear joint learning of word and entity embeddings for entity linking. *Neurocomputing*.

Xiao Cheng and Dan Roth. 2013. Relational inference for wikification. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1787–1796.

Kevin Clark and Christopher D Manning. 2016. Deep reinforcement learning for mention-ranking coreference models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2256–2262.

Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on wikipedia data. In *Proceedings of the 2007 Joint Conference on Empirical*

*Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Zheng Fang, Yanan Cao, Qian Li, Dongjie Zhang, Zhenyu Zhang, and Yanbing Liu. 2019. Joint entity linking with deep reinforcement learning. In *The World Wide Web Conference*, pages 438–447. ACM.

Jun Feng, Minlie Huang, Yijie Zhang, Yang Yang, and Xiaoyan Zhu. 2018. Relation mention extraction from noisy data with hierarchical reinforcement learning. *arXiv preprint arXiv:1811.01237*.

Matthew Francis-Landau, Greg Durrett, and Dan Klein. 2016. Capturing semantic similarity for entity linking with convolutional neural networks. *arXiv preprint arXiv:1604.00734*.

Octavian-Eugen Ganea and Thomas Hofmann. 2017. Deep joint entity disambiguation with local neural attention. *arXiv preprint arXiv:1704.04920*.

Amir Globerson, Nevena Lazic, Soumen Chakrabarti, Amarnag Subramanya, Michael Ringaard, and Fernando Pereira. 2016. Collective entity resolution with multi-focal attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 621–631.

Zhaochen Guo and Denilson Barbosa. 2016. Robust named entity disambiguation with random walks. *Semantic Web*, (Preprint):1–21.

Zhengyan He, Shujie Liu, Mu Li, Ming Zhou, Longkai Zhang, and Houfeng Wang. 2013. Learning entity representation for entity disambiguation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 30–34.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 782–792. Association for Computational Linguistics.

Kristy Hughes, Joel Nothman, and James R Curran. 2014. Trading accuracy for faster named entity linking. In *Proceedings of the Australasian Language Technology Association Workshop 2014*, pages 32–40.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Phong Le and Ivan Titov. 2018. Improving entity linking by modeling latent relations between mentions. *arXiv preprint arXiv:1804.10637*.

David Milne and Ian H Witten. 2008. Learning to link with wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 509–518. ACM.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

Karthik Narasimhan, Adam Yala, and Regina Barzilay. 2016. Improving information extraction by acquiring external evidence with reinforcement learning. *arXiv preprint arXiv:1603.07954*.

Dat Ba Nguyen, Johannes Hoffart, Martin Theobald, and Gerhard Weikum. 2014. Aida-light: High-throughput named-entity disambiguation. *LDOW*, 1184.

Thien Huu Nguyen, Nicolas Fauceglia, Mariano Rodriguez Muro, Oktie Hassanzadeh, Alfio Massimiliano Gliozzo, and Mohammad Sadoghi. 2016. Joint learning of local and global features for entity linking via neural networks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2310–2320.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Maria Pershina, Yifan He, and Ralph Grishman. 2015. Personalized page rank for named entity disambiguation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 238–243.

Minh C Phan, Aixin Sun, Yi Tay, Jialong Han, and Chenliang Li. 2018. Pair-linking for collective entity disambiguation: Two could be better than all. *IEEE Transactions on Knowledge and Data Engineering*.

Jonathan Raiman and Olivier Raiman. 2018. Deeptype: Multilingual entity linking by neural type system evolution. *arXiv preprint arXiv:1802.01021*.

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and global algorithms for disambiguation to wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1375–1384. Association for Computational Linguistics.

Wei Shen, Jianyong Wang, and Jiawei Han. 2015. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Transactions on Knowledge and Data Engineering*, 27(2):443–460.

Wei Shen, Jianyong Wang, Ping Luo, and Min Wang. 2012. Linden: linking named entities with knowledge base via semantic knowledge. In *Proceedings of the 21st international conference on World Wide Web*, pages 449–458. ACM.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

Peng Xu and Denilson Barbosa. 2018. Neural fine-grained entity type classification with hierarchy-aware loss. *north american chapter of the association for computational linguistics*, 1:16–25.

Mengge Xue, Weiming Cai, Jinsong Su, Linfeng Song, Yubin Ge, Yubao Liu, and Bin Wang. 2019. Neural collective entity linking based on recurrent random walk network learning. *arXiv preprint arXiv:1906.09320*.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint learning of the embedding of words and entities for named entity disambiguation. *arXiv preprint arXiv:1601.01343*.

Yi Yang, Ozan Irsoy, and Kazi Shefaet Rahman. 2018. Collective entity disambiguation with structured gradient tree boosting. *arXiv preprint arXiv:1802.10229*.

# Open Event Extraction from Online Text using a Generative Adversarial Network

**Rui Wang**[†] **Deyu Zhou**[†*] **Yulan He**[§]
[†]School of Computer Science and Engineering, Key Laboratory of Computer Network and Information Integration, Ministry of Education, Southeast University, China
[§]Department of Computer Science, University of Warwick, UK
{rui_wang, d.zhou}@seu.edu.cn, yulan.he@warwick.ac.uk

## Abstract

To extract the structured representations of open-domain events, Bayesian graphical models have made some progress. However, these approaches typically assume that all words in a document are generated from a single event. While this may be true for short text such as tweets, such an assumption does not generally hold for long text such as news articles. Moreover, Bayesian graphical models often rely on Gibbs sampling for parameter inference which may take long time to converge. To address these limitations, we propose an event extraction model based on Generative Adversarial Nets, called Adversarial-neural Event Model (AEM). AEM models an event with a Dirichlet prior and uses a generator network to capture the patterns underlying latent events. A discriminator is used to distinguish documents reconstructed from the latent events and the original documents. A byproduct of the discriminator is that the features generated by the learned discriminator network allow the visualization of the extracted events. Our model has been evaluated on two Twitter datasets and a news article dataset. Experimental results show that our model outperforms the baseline approaches on all the datasets, with more significant improvements observed on the news article dataset where an increase of 15% is observed in F-measure.

## 1 Introduction

With the increasing popularity of the Internet, online texts provided by social media platform (e.g. Twitter) and news media sites (e.g. Google news) have become important sources of real-world events. Therefore, it is crucial to automatically extract events from online texts.

Due to the high variety of events discussed online and the difficulty in obtaining annotated data for training, traditional template-based or supervised learning approaches for event extraction are no longer applicable in dealing with online texts. Nevertheless, newsworthy events are often discussed by many tweets or online news articles. Therefore, the same event could be mentioned by a high volume of redundant tweets or news articles. This property inspires the research community to devise clustering-based models (Popescu et al., 2011; Abdelhaq et al., 2013; Xia et al., 2015) to discover new or previously unidentified events without extracting structured representations.

To extract structured representations of events such as *who* did *what*, *when*, *where* and *why*, Bayesian approaches have made some progress. Assuming that each document is assigned to a single event, which is modeled as a joint distribution over the named entities, the date and the location of the event, and the event-related keywords, Zhou *et al.* (2014) proposed an unsupervised Latent Event Model (LEM) for open-domain event extraction. To address the limitation that LEM requires the number of events to be pre-set, Zhou *et al.* (2017) further proposed the Dirichlet Process Event Mixture Model (DPEMM) in which the number of events can be learned automatically from data. However, both LEM and DPEMM have two limitations: (1) they assume that all words in a document are generated from a single event which can be represented by a quadruple <entity, location, keyword, date>. However, long texts such news articles often describe multiple events which clearly violates this assumption; (2) During the inference process of both approaches, the Gibbs sampler needs to compute the conditional posterior distribution and assigns an event for each document. This is time consuming and takes long time to converge.

To deal with these limitations, in this paper, we propose the Adversarial-neural Event Model

*corresponding author



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 282–291, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

(AEM) based on adversarial training for open-domain event extraction. The principle idea is to use a generator network to learn the projection function between the document-event distribution and four event related word distributions (entity distribution, location distribution, keyword distribution and date distribution). Instead of providing an analytic approximation, AEM uses a discriminator network to discriminate between the reconstructed documents from latent events and the original input documents. This essentially helps the generator to construct a more realistic document from a random noise drawn from a Dirichlet distribution. Due to the flexibility of neural networks, the generator is capable of learning complicated nonlinear distributions. And the supervision signal provided by the discriminator will help generator to capture the event-related patterns. Furthermore, the discriminator also provides low-dimensional discriminative features which can be used to visualize documents and events.

The main contributions of the paper are summarized below:

- We propose a novel Adversarial-neural Event Model (AEM), which is, to the best of our knowledge, the first attempt of using adversarial training for open-domain event extraction.
- Unlike existing Bayesian graphical modeling approaches, AEM is able to extract events from different text sources (short and long). And a significant improvement on computational efficiency is also observed.
- Experimental results on three datasets show that AEM outperforms the baselines in terms of accuracy, recall and F-measure. In addition, the results show the strength of AEM in visualizing events.

## 2 Related Work

Our work is related to two lines of research, event extraction and Generative Adversarial Nets.

### Event Extraction

Recently there has been much interest in event extraction from online texts, and approaches could be categorized as domain-specific and open-domain event extraction.

Domain-specific event extraction often focuses on the specific types of events (e.g. sports events or city events). Panem *et al.* (2014) devised a novel algorithm to extract attribute-value pairs and mapped them to manually generated schemes for extracting the natural disaster events. Similarly, to extract the city-traffic related event, Anantharam *et al.* (2015) viewed the task as a sequential tagging problem and proposed an approach based on the conditional random fields. Zhang (2018) proposed an event extraction approach based on imitation learning, especially on inverse reinforcement learning.

Open-domain event extraction aims to extract events without limiting the specific types of events. To analyze individual messages and induce a canonical value for each event, Benson *et al.* (2011) proposed an approach based on a structured graphical model. By representing an event with a binary tuple which is constituted by a named entity and a date, Ritter *et al.* (2012) employed some statistic to measure the strength of associations between a named entity and a date. The proposed system relies on a supervised labeler trained on annotated data. In (Abdelhaq et al., 2013), Abdelhaq *et al.* developed a real-time event extraction system called EvenTweet, and each event is represented as a triple constituted by time, location and keywords. To extract more information, Wang *el al.* (2015) developed a system employing the links in tweets and combing tweets with linked articles to identify events. Xia *el al.* (2015) combined texts with the location information to detect the events with low spatial and temporal deviations. Zhou *et al.* (2014; 2017) represented event as a quadruple and proposed two Bayesian models to extract events from tweets.

### Generative Adversarial Nets

As a neural-based generative model, Generative Adversarial Nets (Goodfellow et al., 2014) have been extensively researched in natural language processing (NLP) community.

For text generation, the sequence generative adversarial network (SeqGAN) proposed in (Yu et al., 2017) incorporated a policy gradient strategy to optimize the generation process. Based on the policy gradient, Lin *et al.* (2017) proposed RankGAN to capture the rich structures of language by ranking and analyzing a collection of human-written and machine-written sentences. To overcome mode collapse when dealing with discrete data, Fedus *et al.* (2018) pro-

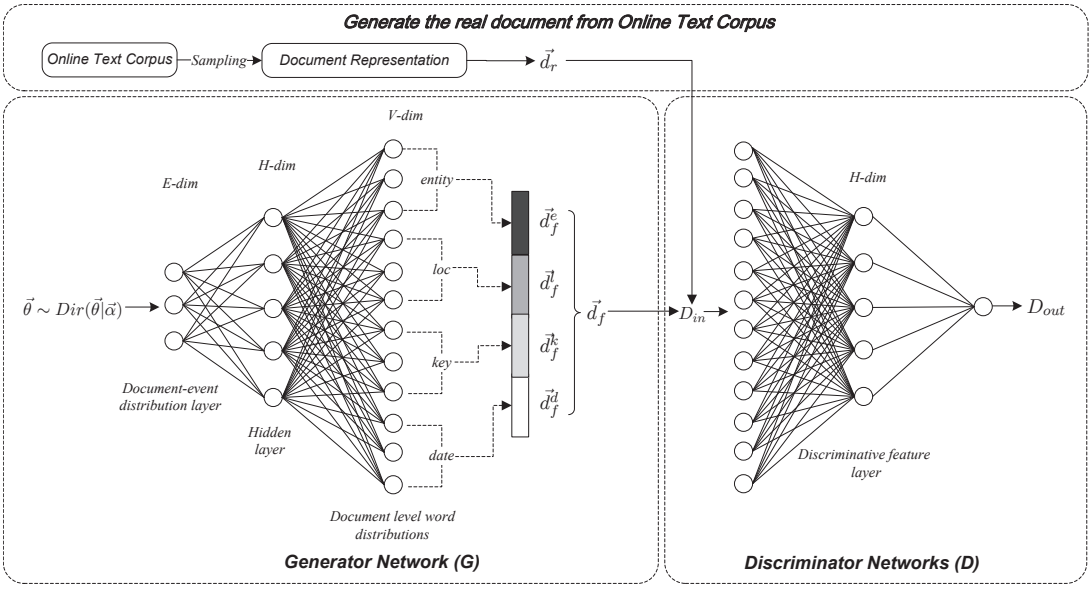


Figure 1: The framework of the Adverarial-neural Event Model (AEM), and $\vec{d}_f^e$, $\vec{d}_f^l$, $\vec{d}_f^k$ and $\vec{d}_f^d$ denote the generated entity distribution, location distribution, keyword distribution and date distribution corresponding to event distribution $\vec{\theta}$.

posed MaskGAN which used an actor-critic conditional GAN to fill in missing text conditioned on the surrounding context. Along this line, Wang *et al.* (2018) proposed SentiGAN to generate texts of different sentiment labels. Besides, Li et al. (2018) improved the performance of semi-supervised text classification using adversarial training, (Zeng et al., 2018; Qin et al., 2018) designed GAN-based models for distance supervision relation extraction.

Although various GAN based approaches have been explored for many applications, none of these approaches tackles open-domain event extraction from online texts. We propose a novel GAN-based event extraction model called AEM. Compared with the previous models, AEM has the following differences: (1) Unlike most GAN-based text generation approaches, a generator network is employed in AEM to learn the projection function between an event distribution and the event-related word distributions (entity, location, keyword, date). The learned generator captures event-related patterns rather than generating text sequence; (2) Different from LEM and DPEMM, AEM uses a generator network to capture the event-related patterns and is able to mine events from different text sources (short and long). Moreover, unlike traditional inference procedure, such as Gibbs sampling used in LEM and DPEMM, AEM could extract the events more efficiently due to the CUDA acceleration; (3) The discriminative features learned by the discriminator of AEM provide a straightforward way to visualize the extracted events.

## 3 Methodology

We describe Adversarial-neural Event Model (AEM) in this section. An event is represented as a quadruple $<e, l, k, d>$, where $e$ stands for non-location named entities, $l$ for a location, $k$ for event-related keywords, $d$ for a date, and each component in the tuple is represented by component-specific representative words.

AEM is constituted by three components: (1) *The document representation module*, as shown at the top of Figure 1, defines a document representation approach which converts an input document from the online text corpus into $\vec{d}_r \in \mathbb{R}^V$ which captures the key event elements; (2) *The generator* $G$, as shown in the lower-left part of Figure1, generates a fake document $\vec{d}_f$ which is constituted by four multinomial distributions using an event distribution $\vec{\theta}$ drawn from a Dirichlet distribution as input; (3) *The discriminator* $D$, as shown in the lower-right part of Figure1, distinguishes the real documents from the fake ones and its output is subsequently employed as a learning signal to update the $G$ and $D$. The details of each component are presented below.

### 3.1 Document Representation

Each document $doc$ in a given corpus $C$ is represented as a concatenation of 4 multinomial distributions which are entity distribution ($\vec{d}_r^e$), location distribution ($\vec{d}_r^l$), keyword distribution ($\vec{d}_r^k$) and date distribution ($\vec{d}_r^d$) of the document. As four distributions are calculated in a similar way, we only describe the computation of the entity distribution below as an example.

The entity distribution $\vec{d}_r^e$ is represented by a normalized $V_e$-dimensional vector weighted by TF-IDF, and it is calculated as:

$$tf^e_{i,doc} = \frac{n^e_{i,doc}}{\sum_{v_e} n^e_{v_e,doc}}$$
$$idf^e_i = \log \frac{|C^e|}{|C^e_i|}$$
$$tf\text{-}idf^e_{i,doc} = tf^e_{i,doc} \times idf^e_i$$
$$d^e_{r,i} = \frac{tf\text{-}idf^e_{i,doc}}{\sum_{v_e} tf\text{-}idf^e_{v_e,doc}}$$

where $C^e$ is the pseudo corpus constructed by removing all non-entity words from $C$, $V_e$ is the total number of distinct entities in a corpus, $n^e_{i,doc}$ denotes the number of $i$-th entity appeared in document $doc$, $|C^e|$ represents the number of documents in the corpus, and $|C^e_i|$ is the number of documents that contain $i$-th entity, and the obtained $d^e_{r,i}$ denotes the relevance between $i$-th entity and document $doc$.

Similarly, location distribution $\vec{d}_r^l$, keyword distribution $\vec{d}_r^k$ and date distribution $\vec{d}_r^d$ of $doc$ could be calculated in the same way, and the dimensions of these distributions are denoted as $V_l$, $V_k$ and $V_d$, respectively. Finally, each document $doc$ in the corpus is represented by a $V$-dimensional ($V$=$V_e$+$V_l$+$V_k$+$V_d$) vector $\vec{d}_r$ by concatenating four computed distributions.

### 3.2 Network Architecture

#### 3.2.1 Generator

The generator network $G$ is designed to learn the projection function between the document-event distribution $\vec{\theta}$ and the four document-level word distributions (entity distribution, location distribution, keyword distribution and date distribution).

More concretely, $G$ consists of a $E$-dimensional document-event distribution layer, $H$-dimensional hidden layer and $V$-dimensional event-related word distribution layer. Here, $E$ denotes the event number, $H$ is the number of units in the hidden layer, $V$ is the vocabulary size and equals to $V_e$+$V_l$+$V_k$+$V_d$. As shown in Figure 1, $G$ firstly employs a random document-event distribution $\vec{\theta}$ as an input. To model the multinomial property of the document-event distribution, $\vec{\theta}$ is drawn from a Dirichlet distribution parameterized with $\vec{\alpha}$ which is formulated as:

$$p(\vec{\theta}|\vec{\alpha}) = Dir(\vec{\theta}|\vec{\alpha}) \triangleq \frac{1}{\triangle(\vec{\alpha})} \prod_{t=1}^{E} \theta_t^{\alpha_t - 1} \quad (1)$$

where $\vec{\alpha}$ is the hyper-parameter of the dirichlet distribution, $E$ is the number of events which should be set in AEM, $\triangle(\vec{\alpha}) = \frac{\prod_{t=1}^{E}\Gamma(\alpha_t)}{\Gamma(\sum_{t=1}^{E}\alpha_t)}$, $\theta_t \in [0,1]$ represents the proportion of event $t$ in the document and $\sum_{t=1}^{E}\theta_t = 1$.

Subsequently, $G$ transforms $\vec{\theta}$ into a $H$-dimensional hidden space using a linear layer followed by layer normalization, and the transformation is defined as:

$$\vec{s}_h = LN(W_h\vec{\theta} + \vec{b}_h) \quad (2)$$
$$\vec{o}_h = \max(\vec{s}_h, l_p \times \vec{s}_h) \quad (3)$$

where $W_h \in \mathbb{R}^{H\times E}$ represents the weight matrix of hidden layer, and $\vec{b}_h$ denotes the bias term, $l_p$ is the parameter of LeakyReLU activation and is set to 0.1, $\vec{s}_h$ and $\vec{o}_h$ denote the normalized hidden states and the outputs of the hidden layer, and $LN$ represents the layer normalization.

Then, to project $\vec{o}_h$ into four document-level event related word distributions ($\vec{d}_f^e$, $\vec{d}_f^l$, $\vec{d}_f^k$ and $\vec{d}_f^d$ shown in Figure 1), four subnets (each contains a linear layer, a batch normalization layer and a softmax layer) are employed in $G$. And the exact transformation is based on the formulas below:

$$\vec{h}^e_w = W^e_w\vec{o}_h + \vec{b}^e_w,\ \vec{d}^e_f = SM(BN(\vec{h}^e_w)) \quad (4)$$
$$\vec{h}^l_w = W^l_w\vec{o}_h + \vec{b}^l_w,\ \vec{d}^l_f = SM(BN(\vec{h}^l_w)) \quad (5)$$
$$\vec{h}^k_w = W^k_w\vec{o}_h + \vec{b}^k_w,\ \vec{d}^k_f = SM(BN(\vec{h}^k_w)) \quad (6)$$
$$\vec{h}^d_w = W^d_w\vec{o}_h + \vec{b}^d_w,\ \vec{d}^d_f = SM(BN(\vec{h}^d_w)) \quad (7)$$

where $SM$ means softmax layer, $W^e_w \in \mathbb{R}^{V_e\times H}$, $W^l_w \in \mathbb{R}^{V_l\times H}$, $W^k_w \in \mathbb{R}^{V_k\times H}$ and $W^d_w \in \mathbb{R}^{V_d\times H}$ denote the weight matrices of the linear layers in subnets, $\vec{b}^e_w$, $\vec{b}^l_w$, $\vec{b}^k_w$ and $\vec{b}^d_w$ represent the corresponding bias terms, $\vec{h}^e_w$, $\vec{h}^l_w$, $\vec{h}^k_w$ and $\vec{h}^d_w$ are state vectors. $\vec{d}^e_f$, $\vec{d}^l_f$, $\vec{d}^k_f$ and $\vec{d}^d_f$ denote the generated entity distribution, location distribution, keyword

distribution and date distribution, respectively, that correspond to the given event distribution $\vec{\theta}$. And each dimension represents the relevance between corresponding entity/location/keyword/date term and the input event distribution.

Finally, four generated distributions are concatenated to represent the generated document $\vec{d}_f$ corresponding to the input $\vec{\theta}$:

$$\vec{d}_f = [\vec{d}_f^e; \vec{d}_f^l; \vec{d}_f^k; \vec{d}_f^d] \tag{8}$$

#### 3.2.2 Discriminator

The discriminator network $D$ is designed as a fully-connected network which contains an input layer, a discriminative feature layer (discriminative features are employed for event visualization) and an output layer. In AEM, $D$ uses fake document $\vec{d}_f$ and real document $\vec{d}_r$ as input and outputs the signal $D_{out}$ to indicate the source of the input data (lower value denotes that $D$ is prone to predict the input data as a fake document and vice versa).

As have previously been discussed in (Arjovsky et al., 2017; Gulrajani et al., 2017), lipschitz continuity of $D$ network is crucial to the training of the GAN-based approaches. To ensure the lipschitz continuity of $D$, we employ the spectral normalization technique (Miyato et al., 2018). More concretely, for each linear layer $l_d(\vec{h}) = W\vec{h}$ (bias term is omitted for simplicity) in $D$, the weight matrix $W$ is normalized by $\sigma(W)$. Here, $\sigma(W)$ is the spectral norm of the weight matrix $W$ with the definition below:

$$\sigma(W) := \max_{\vec{h}:\vec{h}\neq\vec{0}} \frac{\|W\vec{h}\|_2}{\|\vec{h}\|_2} = \max_{\|\vec{h}\|_2\leq 1} \|W\vec{h}\|_2 \tag{9}$$

which is equivalent to the largest singular value of $W$. The weight matrix $W$ is then normalized using:

$$\hat{W_{SN}} := W/\sigma(W) \tag{10}$$

Obviously, the normalized weight matrix $\hat{W_{SN}}$ satisfies that $\sigma(\hat{W_{SN}}) = 1$ and further ensures the lipschitz continuity of the $D$ network (Miyato et al., 2018). To reduce the high cost of computing spectral norm $\sigma(W)$ using singular value decomposition at each iteration, we follow (Yoshida and Miyato, 2017) and employ the power iteration method to estimate $\sigma(W)$ instead. With this substitution, the spectral norm can be estimated with very small additional computational time.

### 3.3 Objective and Training Procedure

The real document $\vec{d}_r$ and fake document $\vec{d}_f$ shown in Figure 1 could be viewed as random samples from two distributions $\mathbb{P}_r$ and $\mathbb{P}_g$, and each of them is a joint distribution constituted by four Dirichlet distributions (corresponding to entity distribution, location distribution, keyword distribution and date distribution). The training objective of AEM is to let the distribution $\mathbb{P}_g$ (produced by $G$ network) to approximate the real data distribution $\mathbb{P}_r$ as much as possible.

To compare the different GAN losses, Kurach (2018) takes a sober view of the current state of GAN and suggests that the Jansen-Shannon divergence used in (Goodfellow et al., 2014) performs more stable than variant objectives. Besides, Kurach also advocates that the gradient penalty (GP) regularization devised in (Gulrajani et al., 2017) will further improve the stability of the model. Thus, the objective function of the proposed AEM is defined as:

$$L_d = -\mathop{\mathbb{E}}_{\vec{d}_r\sim\mathbb{P}_r}[\log(D(\vec{d}_r))] - \mathop{\mathbb{E}}_{\vec{d}_f\sim\mathbb{P}_g}[\log(1-D(\vec{d}_f))] \tag{11}$$

$$L_{gp} = \mathop{\mathbb{E}}_{\vec{d^*}\sim\mathbb{P}_{d^*}}[(\| \nabla_{\vec{d^*}} D(\vec{d^*}) \|_2 - 1)^2] \tag{12}$$

$$L = L_d + \lambda L_{gp} \tag{13}$$

where $L_d$ denotes the discriminator loss, $L_{gp}$ represents the gradient penalty regularization loss, $\lambda$ is the gradient penalty coefficient which trade-off the two components of objective, $\vec{d^*}$ could be obtained by sampling uniformly along a straight line between $\vec{d}_r$ and $\vec{d}_f$, $\mathbb{P}_{d^*}$ denotes the corresponding distribution.

The training procedure of AEM is presented in Algorithm 1, where $E$ is the event number, $n_d$ denotes the number of discriminator iterations per generator iteration, $m$ is the batch size, $\alpha'$ represents the learning rate, $\beta_1$ and $\beta_2$ are hyperparameters of Adam (Kingma and Ba, 2014), $p_a$ denotes $\{\alpha', \beta_1, \beta_2\}$. In this paper, we set $\lambda = 10$, $n_d = 5$, $m = 32$. Moreover, $\alpha'$, $\beta_1$ and $\beta_2$ are set as 0.0002, 0.5 and 0.999.

### 3.4 Event Generation

After the model training, the generator $G$ learns the mapping function between the document-event distribution and the document-level event-related word distributions (entity, location, keyword and date). In other words, with an event distribution

**Algorithm 1** Training procedure for AEM

**Input:** $E$, $\lambda$, $n_d$, $m$, $\alpha'$, $\beta_1$, $\beta_2$
**Output:** the trained $G$ and $D$.

Initial $D$ parameters $\omega_d$ and $G$ parameter $\omega_g$
**while** $\omega_g$ has not converged **do**
  **for** $t = 1, ..., n_d$ **do**
    **for** $j = 1, ..., m$ **do**
      Sample $\vec{d}_r \sim \mathbb{P}_r$,
      Sample a random $\vec{\theta} \sim Dir(\vec{\theta}|\vec{\alpha})$
      Sample a random number $\epsilon \sim U[0, 1]$
      $\vec{d}_f \leftarrow G(\vec{\theta})$
      $\vec{d^*} \leftarrow \epsilon\vec{d}_r + (1-\epsilon)\vec{d}_f$
      $L_d^{(j)} = -\log[D(\vec{d}_r)] - \log[1 - D(\vec{d}_f)]$
      $L_{gp}^{(j)} = (\| \nabla_{\vec{d^*}} D(\vec{d^*}) \| - 1)^2$
      $L^{(j)} \leftarrow L_d^{(j)} + \lambda L_{gp}^{(j)}$
    **end for**
    $\omega_d \leftarrow Adam(\nabla_{\omega_d} \frac{1}{m} \sum_{j=1}^{m} L^{(j)}, \omega_d, p_a)$
  **end for**
  Sample $m$ noise $\left\{\vec{\theta}'^{(j)} \sim Dir(\vec{\theta}|\vec{\alpha})\right\}$
  $\omega_g \leftarrow Adam(\nabla_{\omega_g} \frac{1}{m} \sum_{j=1}^{m} \log[1 - D(G(\vec{\theta}'^{(j)}))], \omega_g, p_a)$
**end while**

$\vec{\theta}'$ as input, $G$ could generate the corresponding entity distribution, location distribution, keyword distribution and date distribution.

In AEM, we employ event seed $\vec{s}_{t\in\{1,...,E\}}$, an $E$-dimensional vector with one-hot encoding, to generate the event related word distributions. For example, in ten event setting, $\vec{s}_1 = [1, 0, 0, 0, 0, 0, 0, 0, 0, 0]^\intercal$ represents the event seed of the first event. With the event seed $\vec{s}_1$ as input, the corresponding distributions could be generated by $G$ based on the equation below:

$$[\vec{\phi}_e^1; \vec{\phi}_l^1; \vec{\phi}_k^1; \vec{\phi}_d^1] = G(\vec{s}_1) \tag{14}$$

where $\vec{\phi}_e^1$, $\vec{\phi}_l^1$, $\vec{\phi}_k^1$ and $\vec{\phi}_d^1$ denote the entity distribution, location distribution, keyword distribution and date distribution of the first event respectively.

## 4 Experiments

In this section, we firstly describe the datasets and baseline approaches used in our experiments and then present the experimental results.

### 4.1 Experimental Setup

To validate the effectiveness of AEM for extracting events from social media (e.g. Twitter) and news media sites (e.g. Google news), three datasets (FSD (Petrovic et al., 2013), Twitter, and Google datasets[1]) are employed. Details are summarized below:

- *FSD dataset* (social media) is the first story detection dataset containing 2,499 tweets. We filter out events mentioned in less than 15 tweets since events mentioned in very few tweets are less likely to be significant. The final dataset contains 2,453 tweets annotated with 20 events.

- *Twitter dataset* (social media) is collected from tweets published in the month of December in 2010 using Twitter streaming API. It contains 1,000 tweets annotated with 20 events.

- *Google dataset* (news article) is a subset of GDELT Event Database[1], documents are retrieved by event related words. For example, documents which contain 'malaysia', 'airline', 'search' and 'plane' are retrieved for event *MH370*. By combining 30 events related documents, the dataset contains 11,909 news articles.

We choose the following three models as the baselines:

- **K-means** is a well known data clustering algorithm, we implement the algorithm using sklearn[2] toolbox, and represent documents using bag-of-words weighted by TF-IDF.

- **LEM** (Zhou et al., 2014) is a Bayesian modeling approach for open-domain event extraction. It treats an event as a latent variable and models the generation of an event as a joint distribution of its individual event elements. We implement the algorithm

[1] http://data.gdeltproject.org/events/index.html
[2] https://scikit-learn.org/

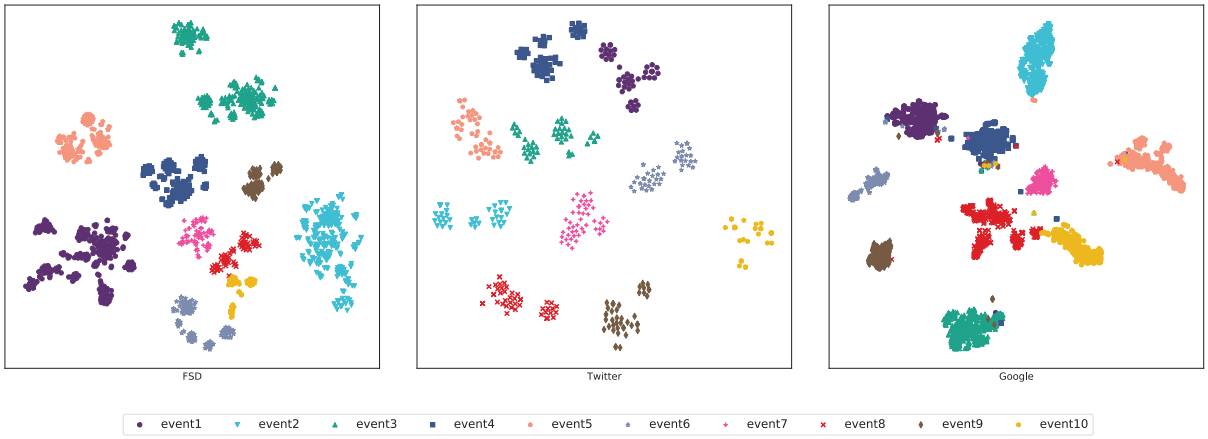


Figure 2: Visualization of the ten randomly selected events on each dataset. Each point denotes a document. Different color denotes different events.

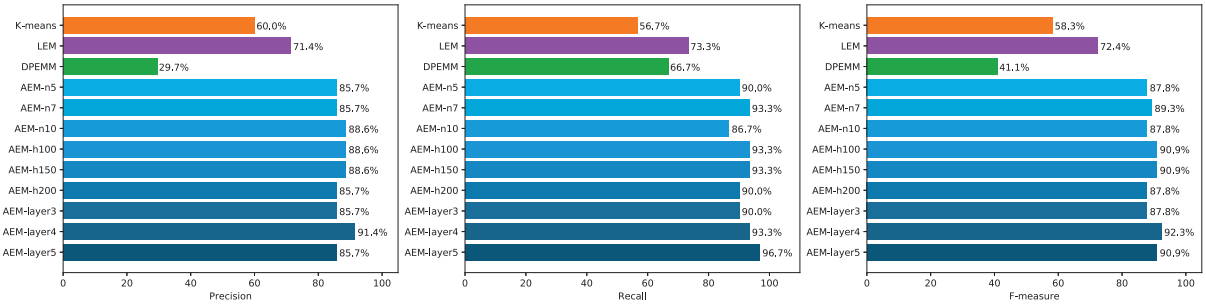


Figure 3: Comparison of methods and parameter settings,'n' and 'h' denote parameter $n_d$ and $H$, other parameters follow the default setting. The vertical axis represents methods/parameter settings, the horizontal axis denotes the corresponding performance value. All blue histograms with different intensity are those obtained by AEM.

with the default configuration.

- **DPEMM** (Zhou et al., 2017) is a non-parametric mixture model for event extraction. It addresses the limitation of LEM that the number of events should be known beforehand. We implement the model with the default configuration.

For social media text corpus (FSD and Twitter), a named entity tagger[3] specifically built for Twitter is used to extract named entities including locations from tweets. A Twitter Part-of-Speech (POS) tagger (Gimpel et al., 2010) is used for POS tagging and only words tagged with nouns, verbs and adjectives are retained as keywords. For the Google dataset, we use the Stanford Named Entity Recognizer[4] to identify the named entities (organization, location and person). Due to the 'date' information not being provided in the Google dataset, we further divide the non-location named entities into two categories ('person' and 'organization') and employ a quadruple <organization, location, person, keyword> to denote an event in news articles. We also remove common stopwords and only keep the recognized named entities and the tokens which are verbs, nouns or adjectives.

### 4.2 Experimental Results

To evaluate the performance of the proposed approach, we use the evaluation metrics such as precision, recall and F-measure. Precision is defined as the proportion of the correctly identified events out of the model generated events. Recall is defined as the proportion of correctly identified true events. For calculating the precision of the 4-tuple, we use following criteria:

- (1) Do the entity/organization, location, date/person and keyword that we have extracted refer to the same event?
- (2) If the extracted representation contains

[3]http://fithub.com/aritter/twitter-nlp
[4]https://nlp.stanford.edu/software/CRF-NER.html

keywords, are they informative enough to tell us what happened?

| Dataset | Method | Precision (%) | Recall (%) | F-measure (%) |
|---|---|---|---|---|
| FSD | K-means | 84.0 | 55.0 | 66.5 |
| | LEM | 80.0 | 80.0 | 80.0 |
| | DPEMM | 84.6 | 85.0 | 84.8 |
| | AEM | **88.0** | **85.0** | **86.5** |
| Twitter | K-means | 68.0 | 75.0 | 71.3 |
| | LEM | 68.0 | 80.0 | 73.5 |
| | DPEMM | 69.2 | 80.0 | 74.2 |
| | AEM | **72.0** | **85.0** | **77.9** |
| Google | K-Means | 60.0 | 56.7 | 58.3 |
| | LEM | 71.4 | 73.3 | 72.4 |
| | DPEMM | 29.7 | 66.7 | 41.3 |
| | AEM | **85.7** | **90.0** | **87.8** |

Table 1: Comparison of the performance of event extraction on the three datasets.

Table 1 shows the event extraction results on the three datasets. The statistics are obtained with the default parameter setting that $n_d$ is set to 5, number of hidden units $H$ is set to 200, and $G$ contains three fully-connected layers. The event number $E$ for three datasets are set to 25, 25 and 35, respectively. The examples of extracted events are shown in Table.2.

It can be observed that K-means performs the worst over all three datasets. On the social media datasets, AEM outpoerforms both LEM and DPEMM by 6.5% and 1.7% respectively in F-measure on the FSD dataset, and 4.4% and 3.7% in F-measure on the Twitter dataset. We can also observe that apart from K-means, all the approaches perform worse on the Twitter dataset compared to FSD, possibly due to the limited size of the Twitter dataset. Moreover, on the Google dataset, the proposed AEM performs significantly better than LEM and DPEMM. It improves upon LEM by 15.5% and upon DPEMM by more than 30% in F-measure. This is because: (1) the assumption made by LEM and DPEMM that all words in a document are generated from a single event is not suitable for long text such as news articles; (2) DPEMM generates too many irrelevant events which leads to a very low precision score. Overall, we see the superior performance of AEM across all datasets, with more significant improvement on the for Google datasets (long text).

We next visualize the detected events based on the discriminative features learned by the trained $D$ network in AEM. The t-SNE (Maaten and Hinton, 2008) visualization results on the datasets are shown in Figure 2. For clarity, each subplot is plotted on a subset of the dataset containing ten randomly selected events. It can be observed that documents describing the same event have been grouped into the same cluster.

To further evaluate if a variation of the parameters $n_d$ (the number of discriminator iterations per generator iteration), $H$ (the number of units in hidden layer) and the structure of generator $G$ will impact the extraction performance, additional experiments have been conducted on the Google dataset, with $n_d$ set to 5, 7 and 10, $H$ set to 100, 150 and 200, and three $G$ structures (3, 4 and 5 layers). The comparison results on precision, recall and F-measure are shown in Figure 3. From the results, it could be observed that AEM with the 5-layer generator performs the best and achieves 96.7% in F-measure, and the worst F-measure obtained by AEM is 85.7%. Overall, the AEM outperforms all compared approaches acorss various parameter settings, showing relatively stable performance.

Finally, we compare in Figure 4 the training time required for each model, excluding the constant time required by each model to load the data. We could observe that K-means runs fastest among all four approaches. Both LEM and DPEMM need to sample the event allocation for each document and update the relevant counts during Gibbs sampling which are time consuming. AEM only requires a fraction of the training time compared to LEM and DPEMM. Moreover, on a larger dataset such as the Google dataset, AEM appears to be far more efficient compared to LEM and DPEMM.

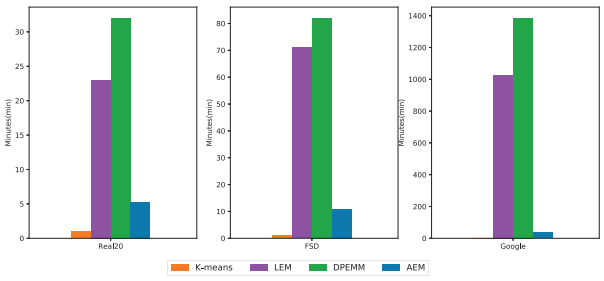


Figure 4: Comparison of training time of models.

## 5 Conclusions and Future Work

In this paper, we have proposed a novel approach based on adversarial training to extract the structured representation of events from online text. The experimental comparison with the state-of-the-art methods shows that AEM achieves improved extraction performance, especially on

| | | | |
|---|---|---|---|
| **FSD dataset** | | | |
| Earthquake in Viriginia | e: nbc coast tremor east eastern<br>l: virginia russian eal croydon washington<br>k: earthquak feel center magnitud hit<br>d: 2011/8/23 2011/7/23 2011/8/06 2011/9/07 2011/9/12 | US debt ceiling | e: hous gifford us gabriell repres<br>l: virginia russian eal croydon washington<br>k: debt bill hous ceil vote<br>d: 2011/8/01 2011/7/23 2011/8/23 2011/8/06 2011/9/13 |
| South sudan independent | e: south sudan independ earthquak tremor<br>l: earth senat congress york nyc<br>k: independ celebr countri congrat challeng<br>d: 2011/7/09 2011/8/06 2011/8/23 2011/7/23 2011/9/07 | US credit downgrade | e: aaa aa yahoo standard obama<br>l: state america tottenham congress seattl<br>k: credit rate downgrad histori lose<br>d: 2011/8/06 2011/7/23 2011/8/23 2011/9/07 2011/9/12 |
| Somalia declare famine | e: somalia africa bakool southern nation<br>l: somalia africa rome independ southern<br>k: declar famin drought part region<br>d: 2011/7/20 2011/7/23 2011/8/06 2011/8/23 2011/9/07 | Norway youth camp attack | e: eyewit norway norweigan rock us<br>l: norway island germani state libya<br>k: camp attack youth bomb shoot<br>d: 2011/7/22 2011/7/23 2011/8/23 2011/8/06 2011/8/10 |
| **Twitter dataset** | | | |
| Russia hosts world cup | e: world cup william russia sport<br>l: qatar russia china europ beij<br>k: host cup reaction world triumph<br>d: 2010/9/3 2010/9/10 2010/9/9 2010/9/8 2010/9/17 | Larry King's last show | e: king larri cnn red vega<br>l: uk state richardson unit south<br>k: final show broadcast night year<br>d: 2010/9/17 2010/9/10 2010/9/8 2010/9/9 2010/9/26 |
| Coach Urban Meyer step down | e: meyer urban reid florida gator<br>l: florida univers senat europ hous<br>k: coach step univers footbal accord<br>d: 2010/9/8 2010/9/10 2010/9/9 2010/9/18 2010/9/17 | Boxer floyd Maweath is arrested | e: boxer floyd mayweath vega obama<br>l: vega las beij europ itali<br>k: guard boxer secur assault arrest<br>d: 2010/9/17 2010/9/9 2010/9/18 2010/9/8 2010/9/26 |
| Christian violence in nigeria | e: christian muslim polit concord eve<br>l: nigeria jos congress christian of<br>k: religion church violenc plagu peopl<br>d: 2010/9/25 2010/9/28 2010/9/26 2010/9/6 2010/9/8 | Xiaobo Liu award nobel prize | e: xiaobo liu nobel prize china<br>l: china oslo congress continent europ<br>k: award live nobel ceremoni dissid<br>d: 2010/9/10 2010/9/8 2010/9/17 2010/9/9 2010/9/18 |
| **Google dataset** | | | |
| Sexual assault in campus | o: university federal college department white<br>l: obama princeton ohio columbia harvard<br>p: mccaskill rose catherine brown duncan<br>k: sexual assault campus title colleges | Lockett is executed death penalty in Oklahoma | o: warner state department cohen robert<br>l: lockett oklahoma states texas ohio<br>p: lockett clayton patton stephanie charles<br>k: execution death penalty lethal minutes |
| Apple & Samsung patent jury | o: apple samsung google inc motorola<br>l: california south santa us calif<br>p: judge steve dunham schmidt mueller<br>k: patent jury smartphone verdict trial | MH370 | o: airlines air transport boeing najib<br>l: malaysia australia beijing malacca houston<br>p: najib hishammuddin hussein clark dolan<br>k: search plane flight aircraft ocean |
| Afghanistan landslide | o: afghanistan united taliban kabul un<br>l: afghanistan badakhshan kabul tajikistan pakistan<br>p: karzai shah hill mark angela<br>k: landslide village rescue mud province | South Africa election | o: anc national mandela congress eff<br>l: zuma africa south africans nkandla<br>p: zuma jacob president nelson malema<br>k: election apartheid elections voters economic |

Table 2: The event examples extracted by AEM.

long text corpora with an improvement of 15% observed in F-measure. AEM only requires a fraction of training time compared to existing Bayesian graphical modeling approaches. In future work, we will explore incorporating external knowledge (e.g. word relatedness contained in word embeddings) into the learning framework for event extraction. Besides, exploring nonparametric neural event extraction approaches and detecting the evolution of events over time from news articles are other promising future directions.

## 6 Acknowledgments

We would like to thank anonymous reviewers for their valuable comments and helpful suggestions. This work was funded by the National Key Research and Development Program of China (2017YFB1002801), the National Natural Science Foundation of China (61772132), the Natural Science Foundation of Jiangsu Province of China (BK20161430) and Innovate UK (grant no. 103652).

## References

Hamed Abdelhaq, Christian Sengstock, and Michael Gertz. 2013. Eventweet: Online localized event detection from twitter. *Proceedings of the VLDB Endowment*, 6(12):1326–1329.

Pramod Anantharam, Payam Barnaghi, Krishnaprasad Thirunarayan, and Amit Sheth. 2015. Extracting city traffic events from social streams. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 6(4):43.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein gan. *arXiv preprint arXiv:1701.07875*.

Edward Benson, Aria Haghighi, and Regina Barzilay. 2011. Event discovery in social media feeds. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 389–398. Association for Computational Linguistics.

William Fedus, Ian Goodfellow, and Andrew M Dai. 2018. Maskgan: Better text generation via filling in the _. *arXiv preprint arXiv:1801.07736*.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein,

Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A Smith. 2010. Part-of-speech tagging for twitter: Annotation, features, and experiments. Technical report, Carnegie-Mellon Univ Pittsburgh Pa School of Computer Science.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. 2017. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5769–5779.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Karol Kurach, Mario Lucic, Xiaohua Zhai, Marcin Michalski, and Sylvain Gelly. 2018. The gan landscape: Losses, architectures, regularization, and normalization. *arXiv preprint arXiv:1807.04720*.

Yan Li and Jieping Ye. 2018. Learning adversarial networks for semi-supervised text classification via policy gradient. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1715–1723. ACM.

Kevin Lin, Dianqi Li, Xiaodong He, Zhengyou Zhang, and Ming-Ting Sun. 2017. Adversarial ranking for language generation. In *Advances in Neural Information Processing Systems*, pages 3155–3165.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. 2018. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*.

Sandeep Panem, Manish Gupta, and Vasudeva Varma. 2014. Structured information extraction from natural disaster events on twitter. In *Proceedings of the 5th International Workshop on Web-scale Knowledge Representation Retrieval & Reasoning*, pages 1–8. ACM.

Sasa Petrovic, Miles Osborne, Richard McCreadie, Craig Macdonald, Iadh Ounis, and Luke Shrimpton. 2013. Can twitter replace newswire for breaking news? In *Seventh international AAAI conference on weblogs and social media*.

Ana-Maria Popescu, Marco Pennacchiotti, and Deepa Paranjpe. 2011. Extracting events and event descriptions from twitter. In *Proceedings of the 20th international conference companion on World wide web*, pages 105–106. ACM.

Pengda Qin, Weiran Xu, and William Yang Wang. 2018. Dsgan: Generative adversarial training for distant supervision relation extraction. *arXiv preprint arXiv:1805.09929*.

Alan Ritter, Oren Etzioni, Sam Clark, et al. 2012. Open domain event extraction from twitter. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1104–1112. ACM.

Ke Wang and Xiaojun Wan. 2018. Sentigan: Generating sentimental texts via mixture adversarial networks. In *IJCAI*, pages 4446–4452.

Yu Wang, David Fink, and Eugene Agichtein. 2015. Seeft: Planned social event discovery and attribute extraction by fusing twitter and web content. In *ICWSM*, pages 483–492.

Chaolun Xia, Jun Hu, Yan Zhu, and Mor Naaman. 2015. What is new in our city? a framework for event extraction using social media posts. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 16–32. Springer.

Yuichi Yoshida and Takeru Miyato. 2017. Spectral norm regularization for improving the generalizability of deep learning. *arXiv preprint arXiv:1705.10941*.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*, pages 2852–2858.

Daojian Zeng, Yuan Dai, Feng Li, R Simon Sherratt, and Jin Wang. 2018. Adversarial learning for distant supervised relation extraction. *Computers, Materials & Continua*, 55(1):121–136.

Tongtao Zhang and Heng Ji. 2018. Event extraction with generative adversarial imitation learning. *arXiv preprint arXiv:1804.07881*.

Deyu Zhou, Liangyu Chen, and Yulan He. 2014. A simple bayesian modelling approach to event extraction from twitter. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 700–705.

Deyu Zhou, Xuan Zhang, and Yulan He. 2017. Event extraction from twitter using non-parametric bayesian mixture model with word embeddings. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 808–817.

# Learning to Bootstrap for Entity Set Expansion

**Lingyong Yan**[1,3], **Xianpei Han**[1,2,*], **Le Sun**[1,2], **Ben He**[3,1,*]
[1]Chinese Information Processing Laboratory [2]State Key Laboratory of Computer Science
Institute of Software, Chinese Academy of Sciences, Beijing, China
[3]University of Chinese Academy of Sciences, Beijing, China
{lingyong2014, xianpei, sunle}@iscas.ac.cn, benhe@ucas.ac.cn

## Abstract

Bootstrapping for Entity Set Expansion (ESE) aims at iteratively acquiring new instances of a specific target category. Traditional bootstrapping methods often suffer from two problems: 1) *delayed feedback*, i.e., the pattern evaluation relies on both its direct extraction quality and the extraction quality in later iterations. 2) *sparse supervision*, i.e., only few seed entities are used as the supervision. To address the above two problems, we propose a novel bootstrapping method combining the Monte Carlo Tree Search (MCTS) algorithm with a deep similarity network, which can efficiently estimate delayed feedback for pattern evaluation and adaptively score entities given sparse supervision signals. Experimental results confirm the effectiveness of the proposed method.

## 1 Introduction

Bootstrapping is widely used for Entity Set Expansion (ESE), which acquires new instances of a specific category by iteratively evaluating and selecting patterns, while extracting and scoring entities. For example, given seeds {*London, Paris, Beijing*} for capital entity expansion, a bootstrapping system for ESE iteratively selects effective patterns, e.g., "*the US Embassy in* *", and extracts new capital entities, e.g., *Moscow*, using the selected patterns.

The main challenges of effective bootstrapping for ESE owe to the *delayed feedback* and the *sparse supervision*. Firstly, bootstrapping is an iterative process, where noises brought by currently selected patterns can affect successive iterations (Movshovitz-Attias and Cohen, 2012; Qadir et al., 2015). Indeed, the pattern evaluation relies on not only its direct extraction quality but also the extraction quality in later iterations, which are correspondingly denoted as *instant feedback* and *delayed feedback* in this paper. For instance, as shown in Figure 1, although "* *is a big city*" and "*the US Embassy in* *" have equal direct extraction quality, the former is considered less useful since its later extracted entities are mostly unrelated. As a result, selecting patterns with high instant feedback but low delayed feedback can cause semantic drift problem (Curran et al., 2007), where the later extracted entities belong to other categories. Secondly, the above difficulty is further compounded by sparse supervision, i.e., using only seed entities as supervision, since it provides little evidence for deciding whether an extracted entity belongs to the same category of seed entities or not.

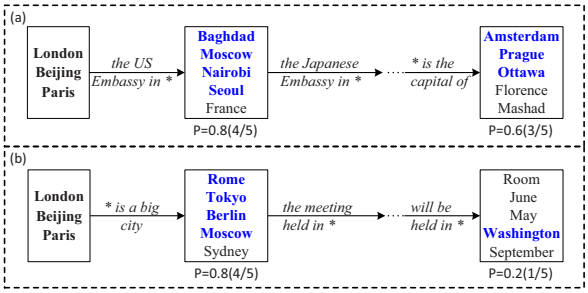


Figure 1: Example of the delayed feedback problem, which expands the capital seeds {*London, Paris, Beijing*}. We demonstrate the top entities extracted by a pattern (correct ones are shown in blue), where $P$ is the precision of extracted entities.

To address the above two challenges, we propose a novel bootstrapping method combining the Monte Carlo Tree Search (MCTS) algorithm with a deep similarity network, aiming to effectively evaluate the delayed feedback of patterns and adaptively score entities given sparse supervision signals. Specifically, our method tackles the delayed feedback problem by enhancing the traditional bootstrapping method using the MCTS algorithm, which effectively estimates each pattern's delayed feedback via efficient multi-step lookahead search. In this way, our method can select the pattern based on its delayed feedback

*Corresponding authors.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 292–301, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

rather than instant feedback, which is beneficial in that the former feedback is regarded more reliable and accurate for bootstrapping. To resolve the sparse supervision problem, we propose a deep similarity network—pattern mover similarity network (PMSN), which uniformly embeds entities and patterns using the distribution vectors on context pattern embeddings, and measures their semantic similarity to seeds as their ranking scores based on those embeddings; furthermore, we combine the PMSN with the MCTS, and fine-tune the distribution vectors using the estimated delayed feedback. In this way, our method can adaptively embed and score entities.

Major contributions of this paper are tri-fold.

- Enhanced bootstrapping via the MCTS algorithm to estimate delayed feedback in bootstrapping. To our best knowledge, this is the first work to combine bootstrapping with MCTS for entity set expansion.
- A novel deep similarity network to evaluate different categories of entities in the bootstrapping for Entity Set Expansion.
- Adaptive entity scoring by combining the deep similarity network with MCTS.

## 2 Background

Traditional bootstrapping systems for ESE are usually provided with sparse supervision, i.e., only a few seed entities, and iteratively extract new entities from corpus by performing the following steps, as demonstrated in Figure 2(a).

**Pattern generation**. Given seed entities and the extracted entities (known entities), a bootstrapping system for ESE firstly generates patterns from the corpus. In this paper, we use lexicon-syntactic surface words around known entities as patterns.

**Pattern evaluation**. This step evaluates generated patterns using sparse supervision and other sources of evidence, e.g., pattern embedding similarity. Many previous studies (Riloff and Jones, 1999; Curran et al., 2007; Gupta and Manning, 2014) use the RlogF function or its variants to evaluate patterns, which usually estimate the instant feedback of each pattern.

**Entity expansion**. This step selects top patterns to match new candidate entities from the corpus.

**Entity scoring**. This step scores candidate entities using sparse supervision, bootstrapping or other external sources of evidence. The top scored entities are then added to the extracted entity set.

As aforementioned, traditional bootstrapping systems for ESE do not consider the delayed feedback when evaluating patterns, leaving considerable potential in further improvement. To estimate the delayed feedback of a pattern, a simple solution is to perform lookahead search for a fixed number of steps and estimate the quality of its successive extracted entities. However, lookahead search may suffer from efficiency issue brought by the enormous search space since there are many candidate patterns at each step. Besides, due to the sparse supervision problem, entity scoring is often unreliable, which in turn influences the delayed feedback estimation.

To address the above two problems, this paper enhances the bootstrapping system using the Monte Carlo Tree Search (MCTS) algorithm for lookahead search, combined with a pattern mover similarity network (PMSN) for better entity scoring. Specifically, we use MCTS for the efficient lookahead search by pruning bad patterns. We additionally estimate the delayed feedback using the PMSN to score entities, given the sparse supervision signals, and fine-tune the PMSN using the delayed feedback estimated by MCTS. In this way, both the MCTS algorithm and the PMSN are devised to enhance each other, resulting in efficient delayed feedback estimation for pattern evaluation and reliable entity scoring.

## 3 Enhancing Bootstrapping via Monte Carlo Tree Search

In this section, we describe how to enhance the traditional bootstrapping for ESE using the MCTS algorithm for efficient delayed feedback estimation.

### 3.1 Efficient Delayed Feedback Estimation via MCTS

To estimate the delayed feedback, this paper uses the Monte Carlo Tree Search (MCTS) algorithm for efficient lookahead search.

At each bootstrapping iteration, the MCTS algorithm performs multiple lookahead search procedures (MCTS simulations). Starting from the same root node, each MCTS simulation performs forward search by iteratively constructing sub-nodes, and moving to one of these sub-nodes. Therefore, the whole MCTS algorithm looks like a tree structure (see Figure 2(b)), where the node $s$ represents for known entities, i.e., both seed entities and previously extracted entities, and the edge,

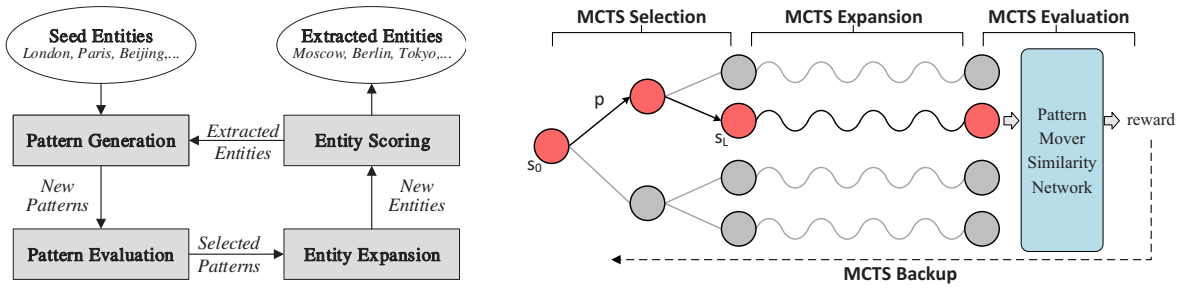


(a) Traditional bootstrapping system for Entity set expansion. The main difference between our method and traditional methods is that we enhance the **pattern evaluation** by the MCTS algorithm, as demonstrated in Figure (b).

(b) The Monte Carlo Tree Search for the pattern evaluation in a bootstrapping system for entity set expansion. The red circles refer to the search node (i.e., seed entities plus extracted entities). A unidirectional edge represents the selection of a pattern to match new entities, which results in a new node.

Figure 2: Traditional Bootstrapping for ESE (a) and the enhanced pattern evaluation using MCTS (b).

linking from a node $s$ to one of its sub-nodes $s'$, represents for selecting a pattern $p$ to expand $s$. At the very beginning, the root tree node $s_0$ is constructed using seed entities and the extracted entities from previous iterations, and is fixed among different simulations. Besides, for each $(s, p)$ pair, we store a cumulative reward $Q(s, p)$ and a visit count $N(s, p)$ during tree search for the subsequent reward function defined in Section 3.3.

Specifically, each MCTS simulation contains four stages, as demonstrated in Figure 2(b):

**Selection**. Starting from $s_0$, each simulation first traverses the MCTS search tree until reaching the leaf node $s_L$ (which is never reached before) or reaching a fixed depth. Each traversing step $i$ corresponding to selecting a pattern $p^i$ by:

$$p^i = \arg\max_p Q(s, p) + \mu(s, p) \tag{1}$$

where $\mu(s, p) \propto \frac{p_\sigma(s,p)}{1+N(s,p)}$, $p_\sigma(s, p)$ is the prior probability of $p$ returned by the policy network $p_\sigma$, which is described in detail in Section 3.2.

Among an MCTS simulation, the lookahead search space is reduced mainly in this stage. Specifically, according to e.q. (1), the lookahead search is more likely to select the potential patterns with high cumulative rewards or high prior probabilities, rather than all patterns.

**Expansion**. When the traversal reaches a leaf node $s_L$, we expand this node by selecting a new pattern to match more entities. Due to the lack of the cumulative rewards on the new patterns, we select the new pattern with the highest prior probability returned by $p_\sigma$.

**Evaluation**. Once finishing the above expansion stage or the simulation procedure reaches a fixed depth, the reward $R$ for the leaf node is returned by first quickly selecting multiple patterns to expand the leaf node and then evaluating the quality of all newly extracted entities in this simulation. We herein use the RlogF function to quickly select patterns rather than the policy network (Running RlogF function is much faster than the policy network). And the reward function is described in Section 3.3 in detail.

**Backup**. At this stage, the returned reward is used to update the cumulative rewards and visit counts of previous $(s, p)$ pairs by:

$$\begin{aligned} N(s, p) &= \sum_j^n \mathbf{1}(s, p, j) \\ Q(s, p) &= \frac{1}{N(s, p)} \sum_{j=1}^n \mathbf{1}(s, p, j) \cdot R \end{aligned} \tag{2}$$

where $\mathbf{1}(s, p, j)$ indicates whether an edge $(s, p)$ was traversed during the $j^{th}$ simulation.

After finishing all MCTS simulations, we use the cumulative reward of each $(s_0, p)$ pair as the delayed feedback for pattern $p$. Because the cumulative rewards are updated many times using the quality evaluation of their future extracted entities, the cumulative reward of each $(s_0, p)$ pair can be regarded as the precise approximation of the delayed feedback if we have a proper reward function. As a result, our method selects the top patterns with the highest cumulative rewards, which are more likely to extract correct entities.

### 3.2 Prior Policy using Pattern Mover Similarity Network

The prior policy network is mainly used to prune bad patterns and therefore reduce the search space in the MCTS. Intuitively, if a pattern is not similar

to other patterns around some entities, it is likely a general pattern or noisy pattern to these entities, and therefore should be pruned.

To this end, this paper proposes a novel deep similarity network, namely Pattern Mover Similarity Network (PMSN), which uniformly embeds patterns, entities and entity sets, and estimates their embedding similarities. We describe this model in detail in Section 4.

Particularly, for a pattern $p$ linked from a tree node $s$, the PMSN is used as the prior policy network to compute the prior probability by:

$$p_\sigma(s,p) = \frac{\text{SIM}(p,E)}{\sum_{p'} \text{SIM}(p',E)} \tag{3}$$

where $E$ is the set of known entities included in node $s$, and $\text{SIM}(p,E)$ is the similarity of the pattern $p$ to the entity set $E$ using the PMSN.

To further reduce the search space (note that the number of patterns at each step can be tens of thousands), we use the RlogF function to retain only the top $k$ for lookahead search. In the experiments, we set $k$ to a reasonably large value, i.e., 200, to balance between efficiency and effectiveness.

### 3.3 Reward Function in MCTS

The reward function is critical for efficiently estimating the real delayed feedback of each pattern. Intuitively, a pattern should have higher delayed feedback if it extracts more similar entities and less unrelated entities. Base on this intuition, we devise the reward function as follows:

$$R = \frac{\sum_{e \in E'} \text{SIM}(e,E_0)}{|E'|} \sigma(\frac{|E'|}{a}) \tag{4}$$

where $E_0$ is the set of known entities in root node, $E'$ is the set of new entities, $\text{SIM}(e,E)$ is the similarity score of newly extracted entity $e$ to known entities, $\sigma(\cdot)$ is the sigmoid function, and $a$ is a "temperature" hyperparameter. The above reward function can be regarded as a trade-off between the extraction quality (the first term) and the extraction amount (the second term). To compute the similarity score, we also exploit the PMSN.

## 4 Pattern Mover Similarity Network

The pattern mover similarity network (PMSN) is a unified model for adaptively scoring the similarity of entities or patterns to seed entities. Specifically, the pattern mover similarity network contains two components: 1) the adaptive distributional pattern embeddings (ADPE) that adaptively represent patterns, entities, and entity sets in a unified way; 2) the pattern mover similarity (PMS) measurement that estimates the similarity of two ADPEs.

The PMSN model is mainly used in three aspects as follows. 1) The PMSN is used as the prior policy network in the MCTS algorithm to evaluate the similarity of patterns. 2) The PMSN is used to evaluate the newly extracted entities within the MCTS simulation, whose evaluation scores are subsequently used to update rewards. 3) The PMSN is also used as the entity scoring function at the Entity Scoring stage in the bootstrapping process as mentioned in Section 2.

### 4.1 Adaptive Distributional Pattern Embeddings

In this section, we first describe how to embed patterns; then, we introduce how to obtain the basic distributional pattern embeddings for uniformly representing entities and patterns without adaptation; finally, we introduce the adaptation mechanism combined with the MCTS algorithm.

**Pattern Embedding**. As a basic step of our PMSN model, we first embed a context pattern of an entity as a single embedding vector. Specifically, we use the average word embeddings, from the pre-trained GloVe (Pennington et al., 2014) embeddings, of a pattern surface text as the pattern embeddings. We filter out patterns containing at least two OOV terms. According to our pilot experiments, replacing the average GloVe word embeddings with alternatives such as Convolutional Neural Network and Recurrent Neural Network does not influence performance.

**Basic distributional pattern embeddings without adaptation**. Based on the single embedding of each context pattern, this paper represents an entity using a distributional vector on its context pattern embeddings, called distributional pattern embeddings (DPE). The intuition behind is that each context pattern represents one aspect of meanings of an entity according to the distributional hypothesis in linguistics (Harris, 1954), while the importance of different patterns to the semantic of an entity varies from each other. Therefore, we use a distributional vector, which stores the importance score of different patterns, together with the context pattern embeddings to represent the semantic of an entity.

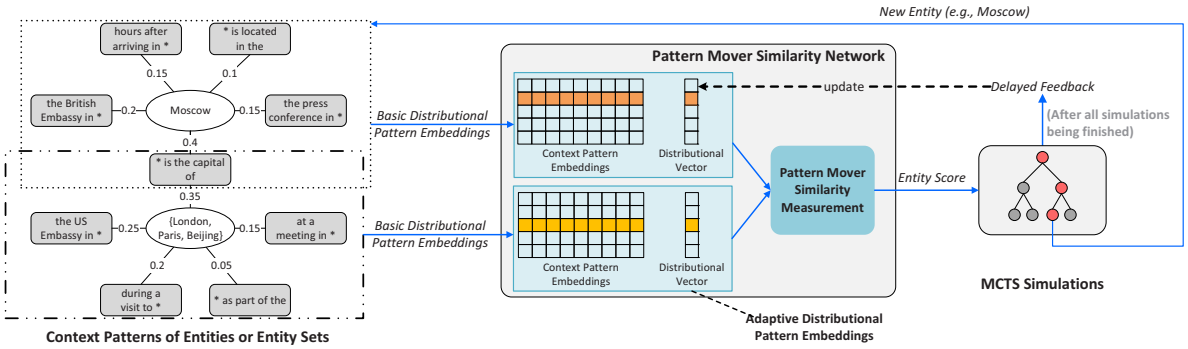


Figure 3: Overall architecture of the Pattern Mover Similarity Network (PMSN), as shown in the rounded square in the middle of the figure.

To estimate each pattern's importance score, we suggest that a context pattern is important to the semantic of an entity if: the pattern matches the entity frequently; the pattern matches as few other entities as possible. Therefore, we design a importance score function for each context pattern $p$ of an entity $e$ as follows:

$$w(p,e) = \frac{N(e,p) \times \log N(e,p)}{C(p)} \tag{5}$$

where $C(p)$ is the number of different entities matched by $p$, $N(e,p)$ is the frequency of $p_i$ matching entity $e$. Ultimately, the above importance scores are mapped to the distributional probabilities of each entity's context pattern such that all probabilities sum up to 1.

In addition, we estimate the basic DPE for a single pattern and a whole entity set. Specifically, a single pattern can be regarded as a special "entity", whose context pattern is only the pattern itself. Similarly, as shown in Figure 3, an entity set can be regarded as another special "entity", whose context patterns are the union of all context patterns of entities in this set. Besides, the importance score function for context patterns of an entity set is slightly different from the one in e.q. (5):

$$w(p,E) = \frac{N(E,p) \times \log N(E,p)}{C(p)} \tag{6}$$

where $E$ is the entity set, $N(E,p)$ is the frequency of $p$ matching all entities in $E$.

Finally, we denote the DPE as $< X, w >$, where $X$ is the context pattern embedding matrix, and $w$ is the vector of distribution probabilities. For efficiency, we only select the top $n$ important patterns. Therefore, $X$ is actually an $n \times d$ matrix, where $d$ is the dimension of each pattern embedding, and $w$ is an $n$-dimensional vector.

**Adaptive distributional pattern embeddings combined with MCTS**. Although the basic DPE can provide unified representations for both patterns and entities, it could still fail to represent the underlying semantics of seed entities as unrelated patterns may match many other entities. To address this problem, we combine the MCTS algorithm with the PMSN, to fine-tune the distributional vector of the basic DPE, resulting in an Adaptive Distributional Pattern Embedding (ADPE). As shown in Figure 3, at each iteration, multiple MCTS simulations are performed to accumulate the long-term feedback of selecting a pattern, where the PMSN is used for entity scoring; after that, the delayed feedback can be efficiently calculated for each pattern. Apart from being used to evaluate the reward of selecting a pattern, the delayed feedback can also be used to estimate the importance score of a pattern, since patterns with the high delayed feedback can extract more correct entities and less incorrect entities in the future. Therefore, at each iteration, we fine-tune the distributional probabilities after the MCTS simulations as following:

$$w_t(p,e) \propto w_{t-1}(p,e) \cdot Q(s_0,p) \tag{7}$$

where $w_{t-1}(p,E)$ is the probability of $p$ at iteration $t-1$, $Q(s_0,p)$ is returned cumulative reward of $p_i$ in iteration $t$.

## 4.2 Pattern Mover Similarity

The similarity measurement for two ADPEs is important, since it is used for delayed feedback estimation and entity scoring. Intuitively, given two entities embedded by two ADPEs, they can be regarded similarly to each other, if they have similar context patterns and similar distributions on these patterns. Therefore, the similarity measurement should take both context pattern embeddings

and corresponding distributional vectors into consideration. Inspired by Kusner et al. (2015) on the sentence similarity measurement, we devise a similarity measurement for two ADPEs as follows:

$$\begin{aligned} \max_{T \geq 0} \quad & \sum_{i,j=1}^{n} T_{ij} \cdot \text{SIM}(i,j) \\ \text{s.t.} \quad & \sum_{j=1}^{n} T_{ij} = w_i, \quad \forall i \in 1,...,n \\ & \sum_{i=1}^{n} T_{ij} = w_j, \quad \forall j \in 1,...,n \end{aligned} \tag{8}$$

where $\text{SIM}(i,j)$ is the cosine similarity between the $i$-th pattern embedding of one entity and the $j$-th pattern embedding of the other entity. We denote the above measurement as the Pattern Mover Similarity (PMS) measurement.

## 5 Experiments

### 5.1 Experimental Settings

| Category | Description | Category | Description |
|---|---|---|---|
| **CAP** | Capital name | **FAC** | Man-made structures |
| **ELE** | Chemical element | **ORG** | Organizations |
| **FEM** | Female first name | **GPE** | Geo-political entities |
| **MALE** | Male first name | **LOC** | Non-GPE locations |
| **LAST** | Last name | **DAT** | A date or period |
| **TTL** | Honorific title | **LANG** | Any named language |
| **NORP** | Nationality, Religion, Political | | |

Table 1: Target categories used on Google Web 1T.

**Datasets**. We conduct experiments on three public datasets: Google Web 1T (Brants and Franz, 2006), APR, and Wiki (Shen et al., 2017). 1) Google Web 1T contains a large scale of $n$-grams compiled from a one trillion words corpus. Following Shi et al. (2014), we use 5-grams as the entity context and filter out those 5-grams containing all stopwords or common words. We use 13 categories of entities (see Table 1) list in Shi et al. (2014) and compare our method with traditional bootstrapping methods for ESE on this corpus. 2) APR (2015 news from AP and Reuters) and Wiki (a subset of English Wikipedia) are two datasets published by Shen et al. (2017). Each of them contains about 1 million sentences. We use totally 12 categories of entities as listed in Shen et al. (2017) and compare the final entity scoring performance on both datasets.

**Baselines**. To evaluate the efficiency of the MCTS and the PMSN, we use several baselines:

1) POS: bootstrapping method which only uses positive seeds without any other constraint;

2) MEB(Curran et al., 2007): mutual exclusion bootstrapping method, which uses the category exclusion as the constraints of bootstrapping;

3) COB(Shi et al., 2014): a probabilistic bootstrapping method which uses both positive and negative seeds.

4) SetExpan(Shen et al., 2017): corpus-based entity set expansion method, which adaptively selects context features and unsupervisedly ensembles them to score entities.

Specifically, we compare baselines (1)-(3) and our method on Google Web 1T; we compare baseline (4) and our method on APR and Wiki [1].

**Metrics**. We use P@n (precision at top $n$), and the mean average precision (MAP) on Google Web 1T as in Shi et al. (2014). As for the APR and the Wiki , we use MAP@n (n=10,20,50) to evaluate entity scoring performance of our method. In our experiments, we manually select frequent entities as the seeds from these datasets; the correctness of all extracted entities is manually judged with external supporting resources, e.g., the entity list collected from Wikipedia[2].

### 5.2 Experimental Results

**Comparison with three baseline methods on Google Web 1T**. Table 2 shows the performance of different bootstrapping methods on Google Web 1T. We can see that our full model outperforms three baseline methods: comparing with POS, our method achieves 41% improvement in P@100, 35% improvement in P@200 and 45% improvement in MAP; comparing with MEB, our method achieves 24% improvement in P@100 and 18% improvement in P@200; comparing with COB, our method achieves 3% improvement in both P@100 and P@200 metrics, and 2% improvement in MAP. The above findings indicate that our method can extract more correct entities with higher ranking scores than the baselines.

**Comparison with SetExpan on APR and Wiki**. To further verify that our method can learn a better representation and adaptively score entities in ESE, we compare our method with the state-

[1]Due to scalability issue, we omit the result of SetExpan on Google Web 1T. On the two smaller datasets, results of the under-performing baselines (1-3) are also omitted for space reason. The incompetitive performance of those 3 baselines could be caused by the difficulty in deriving robust statistical features out of the sparse context patterns on the small ARP and Wiki datasets.

[2]The code is released at https://www.github.com/lingyongyan/mcts-bootstrapping.

| Method | P@10 | P@20 | P@50 | P@100 | P@200 | MAP |
|---|---|---|---|---|---|---|
| POS | 0.84 | 0.74 | 0.55 | 0.41 | 0.34 | 0.42 |
| MEB | 0.83 | 0.79 | 0.68 | 0.58 | 0.51 | - |
| COB* | 0.97 | 0.96 | 0.90 | 0.79 | 0.66 | 0.85 |
| Ours$^{full}$ | **0.97** | **0.96** | **0.92** | **0.82** | **0.69** | **0.87** |
| Ours$^{-MCTS}$ | 0.85 | 0.81 | 0.73 | 0.63 | 0.52 | 0.75 |
| Ours$^{-PMSN}$ | 0.63 | 0.60 | 0.56 | 0.48 | 0.42 | 0.61 |

Table 2: Overall results for entity set expansion on Google Web 1T,where Ours$^{full}$ is the full version of our method, Ours$^{-MCTS}$ is our method with the MCTS disabled, and Ours$^{-PMSN}$ is our method but replacing the PMSN with fixed word embeddings. * indicates COB using the human feedback for seed entity selection.

| Method | APR | | | Wiki | | |
|---|---|---|---|---|---|---|
| | MAP@10 | MAP@20 | MAP@50 | MAP@10 | MAP@20 | MAP@50 |
| SetExpan | 0.90 | 0.86 | 0.79 | 0.96 | 0.90 | 0.75 |
| Ours | **0.96** | **0.90** | **0.81** | **0.98** | **0.93** | **0.79** |

Table 3: The adaptive entity scoring performance of different methods on the APR and Wiki.

| **Category** | **P@20** | **P@50** | **P@100** | **P@200** |
|---|---|---|---|---|
| CAP | 1.00 | 1.00 | 0.94 | 0.69 |
| ELE | 1.00 | 0.84 | 0.51 | 0.36 |
| FEN | 1.00 | 1.00 | 1.00 | 0.95 |
| MALE | 1.00 | 1.00 | 1.00 | 0.98 |
| LAST | 1.00 | 1.00 | 1.00 | 1.00 |
| TTL | 0.85 | 0.64 | 0.49 | 0.34 |
| NORP | 0.95 | 0.96 | 0.89 | 0.60 |
| FAC | 0.95 | 0.86 | 0.59 | 0.42 |
| ORG | 1.00 | 1.00 | 1.00 | 0.94 |
| GPE | 1.00 | 1.00 | 1.00 | 0.94 |
| LOC | 0.80 | 0.76 | 0.70 | 0.67 |
| DAT | 1.00 | 1.00 | 0.82 | 0.56 |
| LANG | 1.00 | 0.98 | 0.81 | 0.52 |

Table 4: The performance of our *full* method on different categories on Google Web 1T.

of-the-art entity set expansion method–SetExpan, which is a non-bootstrapping method on the APR and Wik (see Table 3). From Table 3, we can see that our method outperforms SetExpan on both datasets: on the APR, our method achieves 6% improvement in MAP@10 and 2% improvement in MAP@50 ; on the Wiki, our method can achieve 2% improvement in MAP@10 and 4% improvement in MAP@50. The above results further confirm that our method can improve the performance of bootstrapping for Entity Set Expansion.

### 5.3 Detailed Analysis

**Comparison with the Ours$^{-MCTS}$ method and the Ours$^{-PMSN}$ method**. From Table 2, we can also see that if we replace the Monte Carlo Tree Search by selecting top-$n$ patterns, the performance decreases by 19% in P@100 and 17% in P@200; if we replace the PMSN with word embedding, the performance decreases by 34% in P@100 and 27% in P@200. The results show that both the PMSN and the MCTS algorithm are critical for our model's performance. Remarkably, the PMSN and the MCTS algorithms can enhance each other in that the PMSN can learn a better representation by combing with the MCTS algorithm, and the MCTS can in turn effectively estimate delayed feedback using the PMSN.

**Performance of our full method on different categories of Google Web 1T.** From Table 4, We can see that our method achieves high performance in most categories except for ELE, TTL and FAC entities. The lesser performance of our method on ELE entities is likely caused by the data sparseness — there are fewer than 150 elements of the ELE entities in total. The lower performance on TTL and FAC entities is likely due to the fact that the context patterns of TTL and FAC entities are similar to those of person and location names respectively, which makes them easily be regarded as special person names and location names respectively.

**Influence of the number of context patterns.** Figure 4 shows the performance of our full method under different context pattern numbers. It can be seen that the number of context patterns used to embed entity and entity set heavily influences the performance. Comparing with the settings using fewer context patterns (e.g., 10 and 20, 50), using more context patterns, i.e., 100, in the PMSN has superior performance since it can provide more context information to compare two entities. Besides, further adding context patterns in the PMSN causes the performance decrease for 3%, which is

| Iters | Patterns selected by Ours$^{full}$ | Patterns selected by Ours$^{-MCTS}$ | Patterns selected by Ours$^{-PMSN}$ |
|---|---|---|---|
| 1 | Embassy of Sweden in * | held a meeting in * | * and New York in |
| 2 | Embassy of Belgium in * | * meeting was held on | in New York or * |
| 3 | 's capital city of * | * Meeting to be held | between New York and * |
| 4 | * is the capital city | * held its first meeting | * hotel reservations with discount |
| 5 | * was the capital of | * meeting to be held | * is a great city |

Table 5: The top patterns selected by different methods when expanding capital entities in the first 5 iterations.

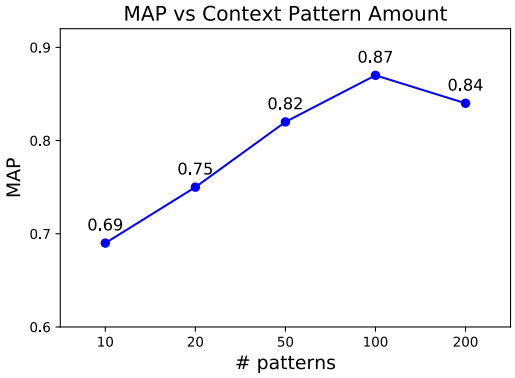


Figure 4: Performance of our *full* method using different context patterns in the PMSN.

likely due to the case that noises can be included when considering too many patterns.

**Top patterns selected by different methods**. To demonstrate the effectiveness of delayed feedback in our method, we illustrate the top 1 pattern in the first five iterations of three methods in Table 5. From Table 5, we can see that the top patterns by our *full* method are more related to seed entities than other two baselines. Besides, we can see that without the MCTS algorithm or the PMSN, most top patterns are less related and easily semantically drifted to other categories.

## 6 Related Work

Entity set expansion (ESE) is a weakly supervised task, which is often given seed entities as supervision and tries to expand new entities related to them. According to the used corpus, there are two types of ESE: limited corpus (Shi et al., 2014; Shen et al., 2017) and large open corpus, e.g. making use of a search engine for the web search (Wang and Cohen, 2007).

Weakly supervised methods for information extraction (IE) are often provided insufficient supervision signals, such as knowledge base facts as distant supervision (Mintz et al., 2009; Hoffmann et al., 2011; Zeng et al., 2015; Han and Sun, 2016), and light amount of supervision samples in bootstrapping(Riloff and Jones, 1999). As a classical technique, bootstrapping usually exploits pattern (Curran et al., 2007), document (Liao and Grishman, 2010) or syntactic and semantic contextual features (He and Grishman, 2015) to extract and classify new instances.

Limited to the sparse supervision, previous work estimate patterns mainly based on its direct extraction features, e.g., the matching statistics with known entities (Riloff and Jones, 1999; Agichtein and Gravano, 2000), which often suffers from the semantic drift problem. To avoid semantic drift, most existing approaches exploit extra constraints, such as parallel multiple categories (Thelen and Riloff, 2002; Yangarber, 2003; McIntosh, 2010), coupling constraints (Carlson et al., 2010), and mutual exclusion bootstrapping (Curran et al., 2007; McIntosh and Curran, 2008). Besides, graph-based methods (Li et al., 2011; Tao et al., 2015) and the probability-based method (Shi et al., 2014) are also used to improve the bootstrapping performance.

To address the *sparse supervision* problem, many previous studies score entities by leveraging lexical and statistical features (Yangarber et al., 2000; Stevenson and Greenwood, 2005; Pantel and Pennacchiotti, 2006; Paşca, 2007; Pantel et al., 2009), which, despite the promising effectiveness, could often fail since the sparse statistical features provide little semantic information to evaluate entities. Recently word embedding based methods (Batista et al., 2015; Gupta and Manning, 2015) use fixed word embedding learned on external resources and evaluate entities by their similarity to seeds. Recently, Berger et al. (2018) propose to learn custom embeddings at each bootstrapping iteration, to trade efficiency for effectiveness.

## 7 Conclusions

In this paper, we propose a deep similarity network-based model combined with the MCTS algorithm to bootstrap Entity Set Expansion. Specifically, we leverage the Monte Carlo Tree Search (MCTS) algorithm to efficiently estimate the delayed feedback of each pattern in the bootstrapping; we propose a Pattern Mover Similarity Network (PMSN) to uniformly embed entities

and patterns using a distribution on context pattern embeddings; we combine the MCTS and the PMSN to adaptively learn a better embedding for evaluating both patterns and entities. Experimental results confirm the superior performance of our PMSN combined with the MCTS algorithm.

## Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grants no. 61433015, 61572477 and 61772505; the Young Elite Scientists Sponsorship Program no. YESS20160177; and University of Chinese Academy of Sciences.

## References

Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting Relations from Large Plain-text Collections. In *Proceedings of the Fifth ACM Conference on Digital Libraries*, pages 85–94, NY, USA.

David S. Batista, Bruno Martins, and Mário J. Silva. 2015. Semi-Supervised Bootstrapping of Relationship Extractors with Distributional Semantics. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 499–504, Lisbon, Portugal. Association for Computational Linguistics.

Matthew Berger, Ajay Nagesh, Joshua Levine, Mihai Surdeanu, and Helen Zhang. 2018. Visual Supervision in Bootstrapped Information Extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2043–2053, Brussels, Belgium. Association for Computational Linguistics.

Thorsten Brants and Alex Franz. 2006. The google web 1t 5-gram corpus version 1.1. *LDC2006T13*.

Andrew Carlson, Justin Betteridge, Richard C. Wang, Estevam R. Hruschka, Jr., and Tom M. Mitchell. 2010. Coupled Semi-supervised Learning for Information Extraction. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, pages 101–110, NY, USA.

James R. Curran, Tara Murphy, and Bernhard Scholz. 2007. Minimising semantic drift with mutual exclusion bootstrapping. In *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics*, volume 6, pages 172–180. Citeseer.

Sonal Gupta and Christopher Manning. 2014. Improved Pattern Learning for Bootstrapped Entity Extraction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 98–108, Ann Arbor, Michigan. Association for Computational Linguistics.

Sonal Gupta and Christopher D. Manning. 2015. Distributed Representations of Words to Guide Bootstrapped Entity Classifiers. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1215–1220, Denver, Colorado. Association for Computational Linguistics.

Xianpei Han and Le Sun. 2016. Global Distant Supervision for Relation Extraction. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Yifan He and Ralph Grishman. 2015. ICE: Rapid Information Extraction Customization for NLP Novices. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 31–35, Denver, Colorado. Association for Computational Linguistics.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based Weak Supervision for Information Extraction of Overlapping Relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 541–550, Stroudsburg, PA, USA. Association for Computational Linguistics.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pages 957–966, Lille, France.

Haibo Li, Danushka Bollegala, Yutaka Matsuo, and Mitsuru Ishizuka. 2011. Using Graph Based Method to Improve Bootstrapping Relation Extraction. In *Computational Linguistics and Intelligent Text Processing*, volume 6609, pages 127–138. Springer Berlin Heidelberg, Berlin, Heidelberg.

Shasha Liao and Ralph Grishman. 2010. Filtered Ranking for Bootstrapping in Event Extraction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 680–688, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tara McIntosh. 2010. Unsupervised Discovery of Negative Categories in Lexicon Bootstrapping. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 356–365, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tara McIntosh and James R. Curran. 2008. Weighted Mutual Exclusion Bootstrapping for Domain Independent Lexicon and Template Acquisition. In *Proceedings of the Australasian Language Technology*

*Association Workshop 2008*, pages 97–105, Hobart, Australia.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant Supervision for Relation Extraction Without Labeled Data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dana Movshovitz-Attias and William W. Cohen. 2012. Bootstrapping Biomedical Ontologies for Scientific Text Using NELL. pages 11–19.

Patrick Pantel, Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu, and Vishnu Vyas. 2009. Web-scale Distributional Similarity and Entity Set Expansion. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2 - Volume 2*, pages 938–947, Stroudsburg, PA, USA. Association for Computational Linguistics.

Patrick Pantel and Marco Pennacchiotti. 2006. Espresso: Leveraging Generic Patterns for Automatically Harvesting Semantic Relations. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 113–120, Stroudsburg, PA, USA. Association for Computational Linguistics.

Marius Paşca. 2007. Weakly-supervised Discovery of Named Entities Using Web Search Queries. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, pages 683–690, New York, NY, USA. ACM.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Ashequl Qadir, Pablo N Mendes, Daniel Gruhl, and Neal Lewis. 2015. Semantic Lexicon Induction from Twitter with Pattern Relatedness and Flexible Term Length. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2432–2439.

Ellen Riloff and Rosie Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *AAAI/IAAI*, pages 474–479.

Jiaming Shen, Zeqiu Wu, Dongming Lei, Jingbo Shang, Xiang Ren, and Jiawei Han. 2017. SetExpan: Corpus-Based Set Expansion via Context Feature Selection and Rank Ensemble. In *ECML PKDD*, volume 10534, pages 288–304, Cham. Springer International Publishing.

Bei Shi, Zhenzhong Zhang, Le Sun, and Xianpei Han. 2014. A probabilistic co-bootstrapping method for entity set expansion. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics*, pages 2280–2290.

Mark Stevenson and Mark A. Greenwood. 2005. A Semantic Approach to IE Pattern Induction. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 379–386, Stroudsburg, PA, USA. Association for Computational Linguistics.

Fangbo Tao, Bo Zhao, Ariel Fuxman, Yang Li, and Jiawei Han. 2015. Leveraging Pattern Semantics for Extracting Entities in Enterprises. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1078–1088, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Michael Thelen and Ellen Riloff. 2002. A Bootstrapping Method for Learning Semantic Lexicons Using Extraction Pattern Contexts. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing*, pages 214–221, Stroudsburg, PA, USA. Association for Computational Linguistics.

Richard C. Wang and William W. Cohen. 2007. Language-independent set expansion of named entities using the web. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference On*, pages 342–350. IEEE.

Roman Yangarber. 2003. Counter-training in Discovery of Semantic Patterns. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, pages 343–350, Stroudsburg, PA, USA. Association for Computational Linguistics.

Roman Yangarber, Ralph Grishman, Pasi Tapanainen, and Silja Huttunen. 2000. Automatic Acquisition of Domain Knowledge for Information Extraction. In *Proceedings of the 18th Conference on Computational Linguistics - Volume 2*, pages 940–946, Stroudsburg, PA, USA. Association for Computational Linguistics.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant Supervision for Relation Extraction via Piecewise Convolutional Neural Networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1753–1762, Lisbon, Portugal. Association for Computational Linguistics.

# Multi-Input Multi-Output Sequence Labeling for Joint Extraction of Fact and Condition Tuples from Scientific Text

**Tianwen Jiang**[‡†,1], **Tong Zhao**[†], **Bing Qin**[‡], **Ting Liu**[‡], **Nitesh V. Chawla**[†], **Meng Jiang**[†]
[‡]Harbin Institute of Technology, Harbin, Heilongjiang, China
[†]University of Notre Dame, Notre Dame, Indiana, USA
[‡]{twjiang,bqin,tliu}@ir.hit.edu.cn
[†]{tzhao2,nchawla,mjiang2}@nd.edu

## Abstract

Condition is essential in scientific statement. Without the conditions (e.g., equipment, environment) that were precisely specified, facts (e.g., observations) in the statements may no longer be valid. Existing ScienceIE methods, which aim at extracting factual tuples from scientific text, do not consider the conditions. In this work, we propose a new sequence labeling framework (as well as a new tag schema) to jointly extract the fact and condition tuples from statement sentences. The framework has (1) a multi-output module to generate one or multiple tuples and (2) a multi-input module to feed in multiple types of signals as sequences. It improves F1 score relatively by 4.2% on BioNLP2013 and by 6.2% on a new bio-text dataset for tuple extraction.

## 1 Introduction

Conditions such as environment and equipment provide validation supports for facts, while the facts focus on scientific observation and hypothesis in scientific literature (Miller, 1947). Existing ScienceIE methods, which extract (subject, relational phrase, object)-tuples from scientific text, do not distinguish the roles of fact and condition. Simply adding a tuple classification module has two weak points: (1) one tuple may have different roles in different sentences; (2) the tuples in one sentence have high dependencies with each other, for example, given a statement sentence in a biochemistry paper (Tomilin et al., 2016):

"*We observed that ... alkaline pH increases the activity of TRPV5/V6 channels in Jurkat T cells.*"

an existing system (Stanovsky et al., 2018) would return one tuple as below:

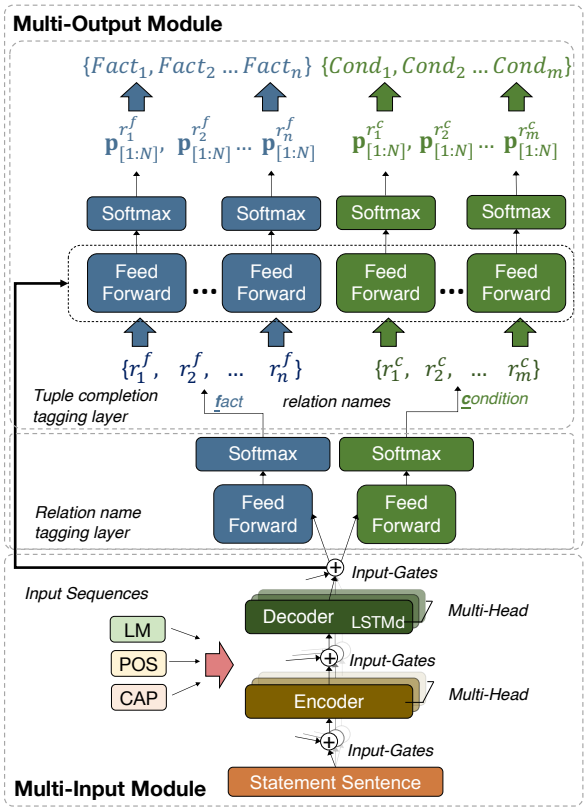


Figure 1: Our framework has two modules: (1) a multi-input module (bottom) based on a multi-head encoder-decoder model with multi-input gates; (2) a multi-output module (top) of a relation name tagging layer and a tuple completion tagging layer.

(alkaline_pH, increases, activity_of_TRPV5/V6 _channels_in_Jurkat_T_cells).

where (a) the object should just be the channel's activity and (b) the condition tuple (TRPV5/V6 _channels, in, Jurkat_T_cells) was not found. Note that the term "TRPV5/V6 _channels" is *not only* the concept in the fact tuple's object *but also* the condition tuple's subject.

In this work, we define the joint tuple extraction task as a *multi-output* sequence labeling problem. First, we create a new tag schema: Non-"O" tags

[1]This work was done when the first author was visiting the University of Notre Dame.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 302–312, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

are formatted as "B/I-XYZ", where

- X ∈ {fact, condition};
- Y ∈ {1: subject; 2: relation; 3: object};
- Z ∈ {concept, attribute, relational phrase}.

Note that if Y="2" then Z="p". So, the number of non-"O" tags is 20. Now each fact/condition tuple can be represented as a tag sequence. Moreover, it is the first work in sequence labeling that concepts and attributes are separated. The fact tuple in the example will ideally be: (alkaline_pH, increases, {TRPV5/V6_channels : activity}).

Figure 1 shows our framework. Multiple tag sequences are generated after the LSTMd decoder, each of which represents a fact or condition tuple. This multi-output module has two layers: one is a relation name tagging layer that predicts the tags of relational phrases and determines the number of output sequences; the other is a tuple completion tagging layer that generates the tag sequences for completing the fact and condition tuples.

To address the challenge of modeling the complex tag schema, besides language model, we incorporate as much information as possible from upstream tools such as Part-of-Speech tagging (POS), Concept detection, Attribute name extraction, and Phrase mining (CAP). And we transform them into tag sequences as the model input. We observe strong dependencies between the token's POS/CAP tags and target tags. We appreciate the high accuracy of existing techniques making the multi-input sequences available for new datasets.

The multi-input multi-output sequence labeling framework is named as MIMO. Experiments demonstrate that it improves F1 score relatively by 6.2% over state-of-the-art models for tuple extraction on a new bio-text dataset we will introduce in the later section. When transferred to the BioNLP2013 dataset without additional training, it improves F1 score relatively by 4.2%. We apply MIMO to a large set of 15.5M MEDLINE papers and construct a knowledge graph: An example can be found in Figure 4.

## 2 A New Dataset

We built a system with GUI (Figure 2) to collect a new dataset for the joint tuple extraction purpose, named Biomedical Conditional Fact Extraction (BioCFE). Three participants (experts in biomedical domain) manually annotated the fact and condition tuples from statement sentences from 31 paper abstracts in the MEDLINE database. The annotation procedure took over 30 minutes on average for each paper. Here is a brief guide to the system. First, the users merged the token(s) into a span. Second, they gave a proper number of fact and/or condition tuple(s), where the proper number is not fixed but depends on the concrete sentence. Each tuple has five slots (subject's concept, subject's attribute, relation phrase, object's concept, and object's attribute). Third, they dragged the spans filling into the slots. If the three annotations are inconsistent, we filtered out the case. Eventually we have 756 fact tuples and 654 condition tuples from 336 annotated sentences. It is common to see one sentence having multiple facts and/or conditions, and actually 61%/52% statement sentences have more than one fact/condition tuples.

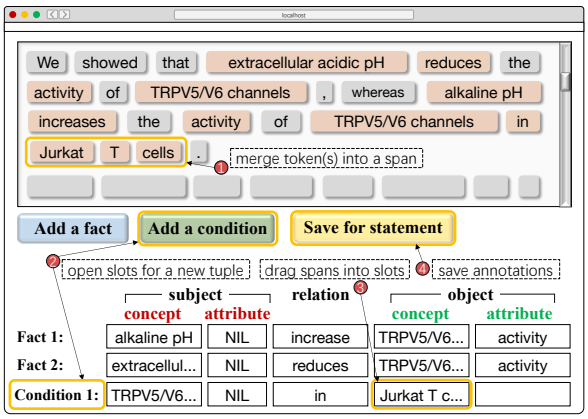


Figure 2: Annotation by four steps: (1) merge token(s) into a span; (2) make slots for a new tuple; (3) drag spans into the slots; (4) save annotations.

## 3 The Proposed Approach

Our approach has two modules: (1) a multi-input module that harnesses recent NLP development to process the text for input sequences from multiple tasks and feeds them into a multi-head encoder-decoder model with multi-input gates; (2) a multi-output module that generates multiple tuple tag sequences for fact and condition tuples, which consists of a relation name tagging layer and a tuple completion tagging layer, as shown in Figure 1.

### 3.1 The Multi-Input Module

**Preprocessing for input sequences:** Following fundamental NLP techniques have achieved high accuracy requiring no additional training with labeled data: Language Model (LM) (Howard and Ruder, 2018), POS (Labeau et al., 2015), CAP (Luan et al., 2018; Jiang et al., 2017; Shang

et al., 2018; Wang et al., 2018a). For any given input sentence, we tokenize it and represent each token by its word embedding (pre-trained GloVe vector in this paper). Then we get another three input sequences by the input sentence and the above three fundamental NLP techniques. (1) A pre-trained LSTM-based language model takes the sentence as input and returns semantic embedding sequence, where the dependencies between a token and its predecessors in distant contexts are preserved. (2) We employ NLTK tool to generate the POS tag sequence for the given sentence. The POS tag sequence indicates syntactic patterns of the words in a sentence, that is the dependencies between POS tags and output tags, like verbs (e.g., VBD) and predicates (e.g., B-f2p). (3) Multiple complementary IE techniques are used to detect concepts, attributes and phrases from the given sentences, being merged and resulting a CAP sequence. We make tags in the format of "B/I - c/a/p" for the tokens of concepts, attributes, and phrases.

Each sequence encodes a specific type of dependencies. A combination of multi-type dependencies learns the complicated dependencies on the 21 tuple tags better than any sole type. LM learns the dependencies between a token and its predecessors in distant contexts, which helps predict the position of subject, relation, and object. POS encodes the syntactic features of words. Dependencies between the POS tag and tuple tag (e.g., "VBD" and "B-f2p") can be modeled. We also spot high dependencies between the CAP tag and tuple tag. For example, the tokens of "B/I-c" (concept) and "B/I-a" (attribute) tags have high probability of being labeled as "B/I-XYc" and "B/I-XYa" in the output sequences, respectively.

**Multi-head Encoder-Decoder**: We investigate two neural models as encoder: one is bidirectional LSTM (BiLSTM), the other is the renown, bidirectional encoder representations from Transformers (BERT). We adopt a LSTM structure as the decoding layer (LSTMd) (Zheng et al., 2017). We observe that the input sequences may have different tag predictability on different sentences. For short sentences, POS and CAP are more useful (modeling local dependencies); for long sentences, LM is more effective (modeling distant dependencies). In order to secure the model's robustness on massive data, we apply a multi-head mechanism to the encoder-decoder model. Each head of the encoder-decoder is fed with one type of input sequence, and they are combined at the end of decoder layer. Thus, the tag prediction becomes more stable than using a simple encoder-decoder without the multi-head.

**Multi-input gates**: We adopt the multi-input gates in ResNet (He et al., 2016) to take the most use of the multi-input sequences. We add the gates to the input of BiLSTM or BERT encoder, the input of LSTMd decoder, and the multi-output module.

### 3.2 Multi-Output Module

We propose to generate multiple output sequences. As annotating multiple tuples from one sentence is common, a token may have different expected tags in the tuples. On BioCFE, we observe that 93.8% statement sentences make multiple tuples: 21.7% of the sentences have at least one token that appears in at least one fact tuple and at least one condition tuple, expecting tags "B/I-fYZ" and "B/I-cYZ"; 18.1% of the sentences have at least one token that appears in one condition tuple as a part of subject and in another condition tuple as a part of object, expecting tags "B/I-c1Z" and "B/I-c3Z". Therefore, we extend the typical one-output sequence labeling to a multi-output design.

Then what is the number of output sequences? We reveal the significant role of relation names in making tuples. If we tagged the relation names out, for each relation name, of tags beginning with "B-f2p" as a fact's and "B-c2p" as a condition's, the module would generate an output sequence, respectively. Then we extract all possible tuples, whose relation has been specified, from every output sequence. Two observations on the annotated data support this idea: We transform each of the 1,410 tuples into a tag sequence. For the same sentence, if the tuples' relation names are the same, we merge their tag sequences into one and then use the matching function in (Stanovsky et al., 2018) to recover the tuples. First, 0 token has conflicting tags among the 240 merged sequences. Second, the recovery has 0 missing or wrong tuple. So, generating one output sequence and completing the tuples per relation name is practical.

The multi-output module has two layers: one is a relation name tagging layer and the other is a tuple completion tagging layer.

**Relation name tagging (RNT) layer**: It consists of feed-forward neural networks (FFNs) and softmax layers. Decoded vectors are fed into the FFNs

and the softmax predict the probability distribution of tags on fact and condition, respectively:

$$\mathbf{p}_i^f = \text{softmax}(\text{FFN}_{RNT}^f(\mathbf{d}_i)), \quad (1)$$
$$\mathbf{p}_i^c = \text{softmax}(\text{FFN}_{RNT}^c(\mathbf{d}_i)). \quad (2)$$

where $f$ is for fact and $c$ for condition. $\mathbf{d}_i$ denotes the $i$-th token's vector given by the LSTMd.

Now we have two tag sequences, one for fact and the other for condition. As we have argued with one-output, extracting tuples from the "two-output" sequences cannot resolve the tag conflicts, either. Here we extract only the relation names: $\{r_1^f, r_2^f, \cdots, r_n^f\}$ denotes the $n$ relation names (beginning with "B-f2p" tag) in fact tuples and $\{r_1^c, r_2^c, \cdots, r_m^c\}$ denotes the $m$ relation names (beginning with "B-c2p" tag) in condition tuples.

**Tuple completion tagging (TCT) Layer**: This layer predicts $n$ fact tag sequences and $m$ condition tag sequences. Each sequence is generated by a FFN and a softmax layer. The FFN obtains the relation name from the RNT layer. The FFN's input also includes the token's vectors from the encoder-decoder model of the multi-input module.

Here we take condition sequences as an example to describe the details of the method. When predicting the $j$-th tag sequence, we define the position embedding of the $i$-th token as follows, representing the relative position to the $j$-th relation name's tag "B-c2p":

$$\mathbf{v}_{i,j}^c = g_{\text{emb}}(r_j^c, i). \quad (3)$$

Thus, the tag probability distributions of the $i$-th token in the condition tag sequences are:

$$\begin{cases} \mathbf{p}_i^{(r_1^c)} = \text{softmax}(\text{FFN}_{TCT}^c(\mathbf{v}_{i,1}^c + \mathbf{d}_i)), \\ \mathbf{p}_i^{(r_2^c)} = \text{softmax}(\text{FFN}_{TCT}^c(\mathbf{v}_{i,2}^c + \mathbf{d}_i)), \\ \cdots, \\ \mathbf{p}_i^{(r_m^c)} = \text{softmax}(\text{FFN}_{TCT}^c(\mathbf{v}_{i,m}^c + \mathbf{d}_i)). \end{cases} \quad (4)$$

Similarly, we have the following tag distributions for the $i$-th token in the fact tag sequences:

$$\begin{cases} \mathbf{p}_i^{(r_1^f)} = \text{softmax}(\text{FFN}_{TCT}^f(\mathbf{v}_{i,1}^f + \mathbf{d}_i)), \\ \mathbf{p}_i^{(r_2^f)} = \text{softmax}(\text{FFN}_{TCT}^f(\mathbf{v}_{i,2}^f + \mathbf{d}_i)), \\ \cdots, \\ \mathbf{p}_i^{(r_n^f)} = \text{softmax}(\text{FFN}_{TCT}^f(\mathbf{v}_{i,n}^f + \mathbf{d}_i)), \end{cases} \quad (5)$$

where $\mathbf{v}_{i,j}^f$ is the position embedding of the $i$-th token in the $j$-th fact sequence, representing the relative position to the relation name's tag "B-f2p".

Finally, we apply the matching function in (Stanovsky et al., 2018) to complete and extract the tuples (i.e., the concepts and/or attributes in the subjects and objects) for each output sequence.

### 3.3 Loss Function and Training

Given a sentence $s$, the loss function of the relation name tagging layer can be written as below:

$$\ell_{RNT}^s = -\sum_{i=1}^{N^s} (\log(p_{i,y_i^f}^f) + \log(p_{i,y_i^c}^c)), \quad (6)$$

where $\mathbf{p}_{i,y}^f$ and $\mathbf{p}_{i,y}^c$ are the probability of predicting $y$ as the tag of the $i$-th token in the fact and condition tag sequences, respectively. $y_i^f$ and $y_i^c$ are the observed tag of the $i$-th token in the fact and condition tuple, respectively. $N^s$ is the length of the sentence $s$.

The loss function of the tuple completion tagging layer is consisted of two parts, loss on fact tuples and loss on condition tuples:

$$\begin{cases} \ell_{TCT}^s = \ell_{fact}^s + \ell_{cond.}^s, \\ \ell_{fact}^s = -\sum_{i=1}^{N^s} \sum_{j=1}^{n} \log(p_{i,y_{i,j}^f}^{(r_j^f)}), \\ \ell_{cond.}^s = -\sum_{i=1}^{N^s} \sum_{j=1}^{m} \log(p_{i,y_{i,j}^c}^{(r_j^c)}), \end{cases} \quad (7)$$

where $n$ and $m$ are the number of fact and condition tag sequences for the sentence $s$, respectively. $p_{i,y_{i,j}^f}^{(r_j^f)}$ and $p_{i,y_{i,j}^c}^{(r_j^c)}$ are the probability of predicting $y_{i,j}^f$ and $y_{i,j}^c$ as the tag of the $i$-th token in the $j$-th fact and condition tag sequence, respectively.

The overall loss function for optimization is:

$$\ell = \ell_{RNT} + \ell_{TCT} = \sum_{s \in \mathcal{S}} (\ell_{RNT}^s + \ell_{TCT}^s), \quad (8)$$

where $\mathcal{S}$ is the set of statement sentences.

**Training details:** On one hand, Equations (6) and (7) show that the error signal can be propagated from the RNT/TCT layers to the encoder-decoder model. On the other hand, the RNT layer specifies the relation names, or say, the tokens that have tags "B/I-f2p" and "B/I-c2p" for each tag sequence in the TCT layer. So we cannot have smooth gradients for back propagation from the TCT layer to the RNT layer. So, in order to have good learning effectiveness, the quality of predicting relation names has been secured beforehand. We pre-train the RNT layer with the multi-input module till the relation name's tag prediction achieves a higher-than-0.8 F1 score. Then we plug the TCT layer onto the RNT layer and train the entire framework to generate the multi-output tag sequences.

## 4 Experiments

We evaluate the performance of condition/fact tag prediction and tuple extraction by the proposed MIMO model, its variants, and state-of-the-art models on the newly annotated BioCFE dataset and transferred to the BioNLP2013 dataset.

### 4.1 Experimental Setup

**Datasets:** Statistics of BioCFE has been given in the Section 2. Additionally, the attribute-related tags take 11.7% and 9.4% of non-"O" tags in fact and condition tuples, respectively. So, it is important to distinguish concept and attribute. To the best of our knowledge, it is the first time that conditional information was carefully annotated on biomedical literature.

We use the system in Figure 2 to annotate a subset of BioNLP2013 Cancer Genetics (CG) task dataset (Nédellec et al., 2013). We have 197 fact tuples and 173 condition tuples. We use this BioNLP dataset as an extra test set for task of fact and condition tuples extraction, but the model will not be trained on this dataset.

**Validation:** The ratio of training:validation:test is 60:8:32. For BioCFE, the evaluation set has 242 fact tuples and 209 condition tuples (on average from 108 sentences). We repeat five times, evaluate the performance, and report average results.

**Evaluation metrics:** For tag prediction, We use standard metrics, precision, recall, and F1 scores. We have similar observations on Micro F1 scores as Macro F1 scores, so we report Macro F1 only. For evaluating tuple extraction, we use pair-wise comparison to match the extracted and ground-truth tuples. We evaluate the correctness on the tuple's five slots using the same metrics.

**Baselines:** We compare with statistical sequence labeling methods: Structured Support Vector Machine (SVM) (Tsochantaridis et al., 2005) and Conditional random field (CRF) (Lafferty et al., 2001). We compare with a neural sequence labeling method, BiLSTM-LSTMd (Zheng et al., 2017). We replace its encoder with BERT (Devlin et al., 2018) to make it a more competitive baseline. We also compare against two renown OpenIE systems, Stanford OpenIE (Angeli et al., 2015) and AllenNLP OpenIE (Stanovsky et al., 2018) followed by a condition/fact classification.

We enhance statistical sequence labeling models with multi-input signals for fairness, and train them for fact tuple and condition tuple extraction separately. In the neural baselines (BiLSTM-LSTMd and BERT-LSTMd), fact extraction and condition extraction share the encoder-decoder model and use different, proper parameters in the linear-softmax layer.

**Hyperparameters:** The multi-input module has a BiLSTM/BERT encoder and a LSTM decoder. The word embeddings were obtained from GloVe (Pennington et al., 2014) with the dimension size $d_{WE}$ = 50. The language model dimension size ns $d_{LM}$ = 200. The size of POS tag embedding is $d_{POS}$ = 6. The size of CAP tag embedding is $d_{CAP}$ = 3. The number of LSTM units in the encoding layer is 300. The number of transformer units in the BERT encoding layer is 768.

### 4.2 Results on BioCFE

In this section, we present overall performance, ablation study, error analysis, and efficiency.

#### 4.2.1 Overall Performance

Table 1 shows that the proposed multi-input multi-output sequence labeling model with a BERT encoder consistently performs the best over all the baselines on tag prediction and tuple extraction. Compared to BiLSTM-LSTMd, BiLSTM-based MIMO improves F1 score relatively by 7.1% on tag prediction and by 8.8% on tuple extraction; compared to BERT-LSTMd, BERT-based MIMO improve F1 by **4.7%** and **6.2%** on the two tasks, respectively. Apparently the BERT encoder significantly improves the performance (by 16.9–17.2% on tag prediction and 7.7–10.3% on tuple extraction). And the MIMO design can further improve it. Neural sequence labeling models perform better than OpenIE systems and statistical methods. Neural sequence labeling models are more adaptive to learning structures with the new tag schema. Open IE method plus a condition/fact classification is not effective.

Compared to BERT-LSTMd, the BERT-based MIMO improves precision and recall relatively by 8.3% and 1.3% on tag prediction; and relatively by 3.1% and 9.3% on tuple extraction, respectively. When the tags were more precisely predicted, the tuple's five slots would be more accurately filled, and we would have more complete tuples.

We also observe that the improvements on condition's tags/tuples are consistently bigger than the improvements on fact's tag/tuples. It demonstrates that the MIMO design recognizes the role of conditions in the statement sentences better.

| Methods | Tag Prediction (%) | | | Tuple Extraction(%) | | |
|---|---|---|---|---|---|---|
| | ***Prec.*** | ***Rec.*** | ***F1 / F1***$_{Fact}$***, F1***$_{Cond.}$ | ***Prec.*** | ***Rec.*** | ***F1 / F1***$_{Fact}$***, F1***$_{Cond.}$ |
| Allennlp OpenIE (Stanovsky et al., 2018) | - | - | - | 42.60 | 38.22 | 40.29 / -, - |
| Stanford OpenIE (Angeli et al., 2015) | - | - | - | 47.11 | 41.62 | 44.19 / -, - |
| Structured SVM (Tsochantaridis et al., 2005) | 32.68 | 25.80 | 28.83 / 32.76, 24.71 | 47.62 | 46.15 | 46.87 / 45.01, 48.72 |
| CRF (Lafferty et al., 2001) | 60.07 | 41.92 | 49.37 / 56.23, 41.87 | 65.19 | 62.44 | 63.78 / 64.07, 63.44 |
| BiLSTM-LSTMd (Zheng et al., 2017) | 61.00 | 56.26 | 58.53 / 65.16, 51.78 | 71.57 | 66.55 | 68.97 / 69.51, 68.41 |
| BERT-LSTMd | 70.07 | 70.19 | 70.13 / 74.30, 65.88 | 78.64 | 73.67 | 76.08 / 76.14, 75.99 |
| MIMO (BiLSTM based) | 67.80 | 58.24 | 62.66 / 66.67, 58.58 | 75.35 | 74.67 | 75.01 / 74.91, 75.10 |
| MIMO (BERT based) | **75.91** | **71.08** | **73.41 / 76.01**, **70.75** | **81.06** | **80.53** | **80.79 / 79.94**, **81.64** |

Table 1: The proposed MIMO outperforms existing methods on tag prediction and tuple extraction in the BioCFE dataset. The MIMO with BERT-based encoder performs the best. Higher score performs better.

#### 4.2.2 Ablation Study

Table 2 compares variants of the proposed model to evaluate the effectiveness of the following components: (1) *multi-input sequences*, such as none, or one (in LM, POS, and CAP), double combination, or triple combination; (2) *multi-input encoder model*, BiLSTM or BERT; (3) *multi-output module*, with the RNT layer only (generating one fact tag sequence and one condition tag sequence) or a combination of RNT and TCT layers (generating multiple sequences for each tuple type).

**Multi-input sequences:** When the choices of the encoder model and multi-output layers are specified, we observe that triple combination of input sequences performs better than double combinations and the double combinations win over the sole input. An additional sequence makes a relative F1 improvement by 1.0–2.4%. The triple combination improves F1 relatively by 3.2–4.1%. This demonstrates that the three types of input sequences encode *complementary information* for learning dependencies in the proposed tag schema.

First, the language model learns the dependencies between a token and its predecessors in distant contexts. Having the LM sequence recognizes subjects and objects relative to the relation names and reduces the false positives of "B/I-X1Z" and "B/I-X3Z". Second, the POS tag encodes the token's syntactic feature. Having the POS sequence improves the precision of tag prediction. For example, verbs and prepositions (e.g., "in", "during") often act as the relation name of facts and conditions, respectively; conjunction words (e.g., "that", "which") indicate subordinate clauses, so the noun phrase before the conjunction word is likely to be the subject of the tuple given by the clause. Third, the formerly-detected concepts, attribute names, and phrases are absolutely useful for tagging the slots of subjects and objects. In other words, the tags "B/I-c" and "B/I-a" in the CAP sequence are strongly associated with the target tags "B/I-XYc" and "B/I-XYa", respectively.

**Encoder in the multi-input module:** Comparing the middle three columns (BiLSTM-based encoder) and the right-hand three columns (BERT-based encoder), one can easily tell the significant improvement brought by the BERT model.

**Layers in the multi-output module:** If the multi-output models have both RNT and TCT layers, the F1 score is relatively 1.4–5.0% higher than the models that have the RNT layer only. Moreover, the recall is improved relatively by 1.5–9.0%. So the TCT layer, which generates multiple tag sequences for each type of tuple (i.e., fact and condition), plays a very important role in recognizing the multiple tuples from one statement sentence.

#### 4.2.3 Error Analysis

Table 3 presents the confusion matrices made by the BERT-based MIMO on predicting non-"O" tags for facts and conditions, respectively. The columns are predicted tags and the rows are actual ones. Perfect results would be diagonal matrices.

We observe that the numbers at the diagonal are consistently bigger than the numbers on the corresponding row and column. The accuracy scores are 0.905 for predicting fact tags and 0.908 for predicting condition tags. Of the 182 actual "B-f2p", the model predicted that 175 were "B-f2p"; of the 186 actual "B-c2p", it predicted that one was "I-c1c" and one was "I-c3c". It demonstrates the high accuracy (0.961 and 0.989) of extracting relation names for multi-output generation.

| Settings | | | | BiLSTM-based Encoder (%) | | | BERT-based Encoder (%) | | |
|---|---|---|---|---|---|---|---|---|---|
| LM | POS | CAP | MO | *Prec.* | *Rec.* | *F1 / F1$_{Fact}$, F1$_{Cond.}$* | *Prec.* | *Rec.* | *F1 / F1$_{Fact}$, F1$_{Cond.}$* |
| | | | ✗ | 71.57 | 66.55 | 68.97 / 69.51, 68.41 | 78.64 | 73.67 | 76.08 / 76.14, 75.99 |
| ✔ | | | ✗ | 72.84 | 67.36 | 69.99 / 69.22, 70.75 | 79.57 | 74.77 | 77.10 / 77.47, 76.71 |
| | ✔ | | ✗ | 72.68 | 68.11 | 70.32 / 71.85, 68.78 | 79.66 | 74.59 | 77.04 / 76.80, 77.27 |
| | | ✔ | ✗ | 72.84 | 67.69 | 70.17 / 69.42, 70.91 | 79.01 | 74.02 | 76.43 / 77.43, 75.43 |
| ✔ | ✔ | | ✗ | 73.38 | 68.81 | 71.02 / 71.86, 70.15 | 80.66 | 75.52 | 78.01 / 77.91, 78.09 |
| ✔ | | ✔ | ✗ | 73.71 | 68.14 | 70.82 / 70.34, 71.27 | 80.90 | 76.20 | 78.48 / 78.35, 78.60 |
| | ✔ | ✔ | ✗ | 74.17 | 69.12 | 71.56 / 70.89, 72.21 | 81.13 | 76.20 | 78.59 / 78.42, 78.73 |
| ✔ | ✔ | ✔ | ✗ | 74.63 | 69.21 | 71.82 / 71.68, 71.94 | **81.74** | 76.29 | 78.92 / 78.67, 79.16 |
| | | | ✔ | 71.80 | 72.34 | 72.07 / 72.39, 71.73 | 77.38 | 79.19 | 78.27 / 76.64, 79.89 |
| ✔ | | | ✔ | 72.41 | 73.35 | 72.88 / 71.99, 73.77 | 79.04 | 79.87 | 79.45 / 79.09, 79.81 |
| | ✔ | | ✔ | 73.85 | 73.74 | 73.80 / 72.64, 74.96 | 79.40 | 79.50 | 79.45 / 78.66, 80.24 |
| | | ✔ | ✔ | 72.69 | 74.27 | 73.47 / 72.19, 74.75 | 79.05 | 79.72 | 79.39 / 78.41, 80.36 |
| ✔ | ✔ | | ✔ | 74.43 | 73.73 | 74.08 / 73.19, 74.96 | 79.67 | **80.65** | 80.16 / 79.16, 81.14 |
| ✔ | | ✔ | ✔ | 74.31 | 74.33 | 74.32 / 74.45, 74.19 | 79.97 | 79.56 | 79.76 / 79.06, 80.47 |
| | ✔ | ✔ | ✔ | 75.15 | 74.12 | 74.63 / 74.69, 74.57 | 79.41 | 79.98 | 79.70 / 79.49, 79.90 |
| ✔ | ✔ | ✔ | ✔ | **75.35** | **74.67** | **75.01 / 74.91, 75.10** | 81.06 | 80.53 | **80.79 / 79.94, 81.64** |

Table 2: The proposed MIMO that employs (a) multi-input Language Models, POS tags, and Concept-Attribute-Phrase sequences, (b) multi-output tag sequences, (c) BERT-based encoder performs the best on tuple extraction.

The ovals in each confusion matrix present the most significant type of error. Of a small set of actual subjects, the model predicted them as objects, and vise versa, though the fact/condition role and concept role were correctly predicted.

The dashed circles show the second frequent type of error. Of the actual "I-f2p" tokens, the model predicted that 7 were "B-f2p"; for the actual "I-c2p", it predicted that 6 were "B-c2p". Basically, it was because of missing the beginning word of the relational phrases. Of the actual "B-f3a" tokens, the model predicted 6 were "I-f2p". Future work will aim at improving the prediction of the boundaries of long relational phrases.

#### 4.2.4 Efficiency

All the experiments were conducted on 16 Graphics Cards (GeForce GTX 1080 Ti), where one individual model only used 1 GPU. Each model was trained for 1,000 epochs. For the BiLSTM-LSTMd MIMOs, the pre-training took 2.4 hours and the re-training (TCT layer) took 0.4 hour. For the BERT-LSTMd MIMOs of the best performance, the pre-training took 3.5 hours and the re-training took 0.9 hour. It took 5.7 hours to extract fact and condition tuples from 141 million sentences in the MEDLINE text data. It is comparable with existing approaches in terms of scalability.

### 4.3 Results on BioNLP2013

As shown in Table 3, the BERT-LSTMd MIMO model achieves an F1 score of 0.790 on tuple extraction from BioNLP2013. Note that the model was trained on BioCFE that has no overlapping sentence with BioNLP2013. This score is comparable with the testing F1 score on the BioCFE (0.808), which demonstrates the effectiveness and reliability of the proposed model.

Our model improves the F1 score relatively by 4.2% over the best baseline BERT-LSTMd. The improvement on recall is more substantial: It improves recall relatively by 5.8%. It was because of the design of the multi-output module: the TCT layer generates multiple tag sequences based on the relation names predicted by the RNT layer. A token in a statement sentence may have different roles in different tuples of the same type (fact or condition). For example, given the following statement sentence:

"*Immunohistochemical staining of the tumors demonstrated a decreased number of blood vessels in the treatment group versus the controls.*"

The proposed model is able to find one fact tuple and two condition tuples precisely:

-*Fact 1:* ({tumors:immunohistochemical_staining}, demonstrated, {blood_vessels:decreased_number})

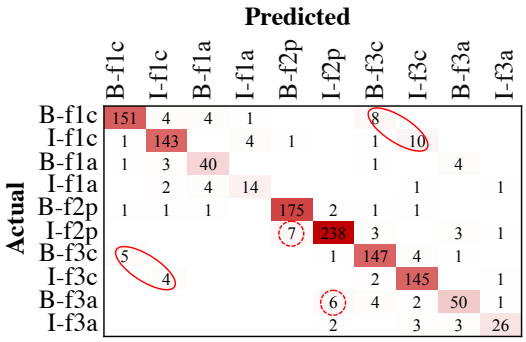


(a) Fact tags

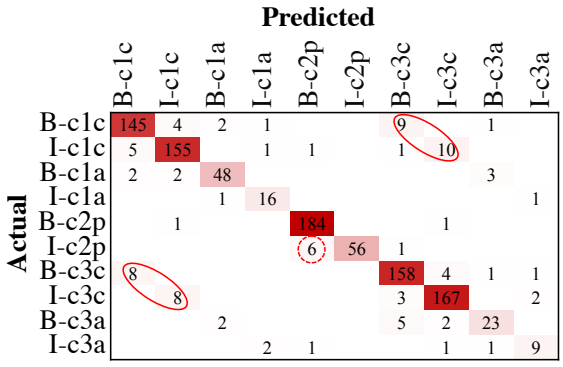


(b) Condition tags

Figure 3: Confusion matrices on predicting fact tags (Top) and condition tags (Bottom) in BioCFE data.

-*Condition 1:* (blood_vessels,in,**treatment_group**)
-*Condition 2:* (**treatment_group**,versus,controls)

Note that the concept "treatment_group" acts as the *object* of Condition Tuple 1 (having tags "B/I-c3c") and the *subject* of Condition Tuple 2 (having tags "B/I-c1c"). The multi-output design tackled this issue while other models could not.

**Compared with BioCFE**: On BioCFE, the F1 score on condition tuple extraction is a bit higher than that on fact tuple extraction (81.64 vs 79.94). On BioNLP2013, we have the opposite observation (78.58 vs 79.42). They are still comparable but if we look at the error cases, we find that most of the false predictions of condition tuple come from long sentences (having more than 30 words). And 35% of the sentences in BioNLP are long sentences, while only 5% in Bio CFE are long. Long dependency modeling is always challenging for IE, especially condition extraction. We will study it in the future work.

### 4.4 A Visualized Case Study

Scientific knowledge graph enables effective search and exploration. It is certainly important to represent the conditions of the corresponding fact

| Methods | *Prec. (%)* | *Rec. (%)* | *F1 (%)* |
|---|---|---|---|
| Allennlp OpenIE | 41.84 | 37.87 | 39.75 |
| Stanford OpenIE | 45.04 | 42.99 | 43.99 |
| Structured SVM | 49.69 | 47.54 | 48.59 |
| CRF | 59.55 | 56.29 | 57.88 |
| BiLSTM-LSTMd | 65.33 | 63.09 | 64.19 |
| BERT-LSTMd | 76.99 | 74.74 | 75.85 |
| MIMO (BERT based) | **78.93** | **79.07** | **79.00** |

Table 3: The BERT-LSTMd MIMO model performs the best on tuple extraction in BioNLP2013.

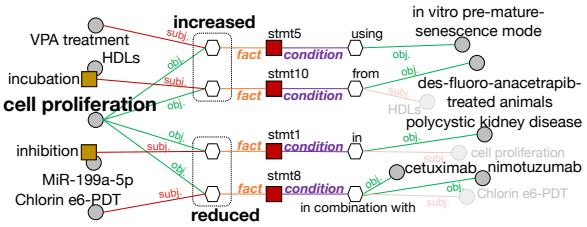


Figure 4: Structuring tuples detected from four statement sentences that mention "cell proliferation" into a snapshot of scientific knowledge graph with fact tuples on the left and condition tuples on the right.

being valid in the graph. As we have applied our model to the large MEDLINE dataset, Figure 4 visualizes the fact and condition tuples extracted from four statement sentences about "cell proliferation". On the left side, we find (1) "VPA treatment" and the "incubation" of "HDLs" increased cell proliferation, while (2)"Chlorin e6-PDT" and the "inhibition" of "MiR-199a-5p" decreased cell proliferation. On the right, we are aware of the conditions of the factual claims. They describe the methodology of the observation (e.g., "using", "in combination with") or the context (e.g., "in" a specific disease or "from" specific animals). In some other cases, we find the temperature and pH values are detected as the conditions of observations.

## 5 Related Work

### 5.1 Scientific Information Extraction

Information extraction in scientific literature, e.g., computer science, biology and chemistry, has been receiving much attention in recent years. ScienceIE in computer science focus on concept recognition and factual relation extraction (Luan et al., 2017; Gábor et al., 2018; Luan et al., 2018). ScienceIE in biological literature aims at identifying the relationships between biological concepts

(i.e., proteins, diseases, drugs and genes) (Kang et al., 2012; Liu et al., 2018; Xu et al., 2018). Rule-based approaches were used in early studies (Rindflesch and Fiszman, 2003; Kang et al., 2012). Recently, a wide line of neural network models have been proposed and outperformed traditional methods (Wang et al., 2018b; Liu et al., 2018; Xu et al., 2018; Jiang et al., 2019). Wang *et al.* (2018b) investigated different kinds of word embeddings on different NLP tasks in the biological domain. Liu *et al.* (2018) employed attention-based neural networks to extract chemical-protein relations. Xu *et al.* (2018) used the BiLSTM model to recognize the drug interaction. In our work, we extract biological relational facts as well as their conditions. The condition tuples are essential to interpreting the factual claims.

### 5.2 Open-Domain IE

Open IE refers to the extraction of (subject, relation, object)-triples from plain text (Angeli et al., 2015; Stanovsky et al., 2018; Saha et al., 2018; Wang et al., 2018a). The schema for the relations does not need to be specified in advance. Distant supervision has been widely used because the size of the benchmark data is often limited (Banko et al., 2007; Wu and Weld, 2010). Stanovsky *et al.* (2018) proposed supervised neural methods for OpenIE. The idea was to transform annotated tuples into tags and learn via sequence tagging. We create a new tag schema and propose a novel sequence labeling framework.

### 5.3 Sequence Labeling for IE

Statistical models have been studied for long, including Hidden Markov Models (HMM), Support Vector Machine (SVM), and Conditional Random Fields (CRF) (Lafferty et al., 2001; Tsochantaridis et al., 2005; Passos et al., 2014; Luo et al., 2015; Li et al., 2018). However, these methods rely heavily on hand-crafted features. Then neural network models become popular and obtain more promising performance than traditional statistical methods (Yang and Mitchell, 2017; Zheng et al., 2017; Wang et al., 2019; Yu et al., 2019). So, we use them as strong baselines.

## 6 Conclusions

We present a new problem to find conditional information in scientific statements. We created a new tag schema for jointly extracting condition and fact tuples from scientific text. We proposed a multi-input multi-output sequence labeling model to utilize results from well-established related tasks and extract an uncertain number of fact(s)/condition(s). Our model yields improvement over all the baselines on a newly annotated dataset BioCFE and a public dataset BioNLP2013. We argue that structured representations of knowledge, such as fact/condition tuple, for scientific statements will enable more intelligent downstream applications. In the future work, we will explore the use of the structured tuples to bridge the gap between text content and knowledge-based applications, such as knowledge-based scientific literature search.

## Acknowledgments

he authors would like to thank the anonymous reviewers for their valuable comments and insightful suggestions. This work was supported in part by National Key Research and Development Program of China No. 2018YFB1005103, National Natural Science Foundation of China (NSFC) No. 61632011, and NSF Grant CCF-1901059.

## References

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 344–354.

Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *IJCAI*, volume 7, pages 2670–2676.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Kata Gábor, Davide Buscaldi, Anne-Kathrin Schumann, Behrang QasemiZadeh, Haifa Zargayouna, and Thierry Charnois. 2018. Semeval-2018 task 7: Semantic relation extraction and classification in scientific papers. In *SemEval*, pages 679–688.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*, pages 770–778.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In

*Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 328–339.

Meng Jiang, Jingbo Shang, Taylor Cassidy, Xiang Ren, Lance M Kaplan, Timothy P Hanratty, and Jiawei Han. 2017. Metapad: Meta pattern discovery from massive text corpora. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 877–886. ACM.

Tianwen Jiang, Tong Zhao, Bing Qin, Ting Liu, Nitesh V Chawla, and Meng Jiang. 2019. The role of "condition": A novel scientific knowledge graph representation and construction model. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1634–1642. ACM.

Ning Kang, Bharat Singh, Zubair Afzal, Erik M van Mulligen, and Jan A Kors. 2012. Using rule-based natural language processing to improve disease normalization in biomedical text. *Journal of the American Medical Informatics Association*, 20(5):876–881.

Matthieu Labeau, Kevin Löser, and Alexandre Allauzen. 2015. Non-lexical neural architecture for fine-grained pos tagging. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 232–237.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Qi Li, Meng Jiang, Xikun Zhang, Meng Qu, Timothy P Hanratty, Jing Gao, and Jiawei Han. 2018. Truepie: Discovering reliable patterns in pattern-based information extraction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1675–1684. ACM.

Sijia Liu, Feichen Shen, Ravikumar Komandur Elayavilli, Yanshan Wang, Majid Rastegar-Mojarad, Vipin Chaudhary, and Hongfang Liu. 2018. Extracting chemical–protein relations using attention-based neural networks. *Database*, 2018:bay102.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreferencefor scientific knowledge graph construction. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*.

Yi Luan, Mari Ostendorf, and Hannaneh Hajishirzi. 2017. Scientific information extraction with semi-supervised neural tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2641–2651, Copenhagen, Denmark. Association for Computational Linguistics.

Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie. 2015. Joint entity recognition and disambiguation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 879–888.

David L Miller. 1947. The nature of scientific statements. *Philosophy of Science*, 14(3):219–223.

Claire Nédellec, Robert Bossy, Jin-Dong Kim, Jung-Jae Kim, Tomoko Ohta, Sampo Pyysalo, and Pierre Zweigenbaum. 2013. Overview of bionlp shared task 2013. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 1–7. Association for Computational Linguistics.

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. *arXiv preprint arXiv:1404.5367*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Thomas C Rindflesch and Marcelo Fiszman. 2003. The interaction of domain knowledge and linguistic structure in natural language processing: interpreting hypernymic propositions in biomedical text. *Journal of biomedical informatics*, 36(6):462–477.

Swarnadeep Saha et al. 2018. Open information extraction from conjunctive sentences. In *COLING*, pages 2288–2299.

Jingbo Shang, Jialu Liu, Meng Jiang, Xiang Ren, Clare R Voss, and Jiawei Han. 2018. Automated phrase mining from massive text corpora. *IEEE Transactions on Knowledge and Data Engineering*, 30(10):1825–1837.

Gabriel Stanovsky, Julian Michael, Luke Zettlemoyer, and Ido Dagan. 2018. Supervised open information extraction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 885–895.

Victor N Tomilin, Alena L Cherezova, Yuri A Negulyaev, and Svetlana B Semenova. 2016. Trpv5/v6 channels mediate ca2+ influx in jurkat t cells under the control of extracellular ph. *Journal of cellular biochemistry*, 117(1):197–206.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. 2005. Large margin methods for structured and interdependent output variables. *Journal of machine learning research*, 6(Sep):1453–1484.

Xuan Wang, Yu Zhang, Qi Li, Yinyin Chen, and Jiawei Han. 2018a. Open information extraction with meta-pattern discovery in biomedical literature. In *Proceedings of the 2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, pages 291–300. ACM.

Xueying Wang, Haiqiao Zhang, Qi Li, Yiyu Shi, and Meng Jiang. 2019. A novel unsupervised approach for precise temporal slot filling from incomplete and noisy temporal contexts. In *The World Wide Web Conference*, pages 3328–3334. ACM.

Yanshan Wang, Sijia Liu, Naveed Afzal, Majid Rastegar-Mojarad, Liwei Wang, Feichen Shen, Paul Kingsbury, and Hongfang Liu. 2018b. A comparison of word embeddings for the biomedical natural language processing. *Journal of biomedical informatics*, 87:12–20.

Fei Wu and Daniel S Weld. 2010. Open information extraction using wikipedia. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 118–127. Association for Computational Linguistics.

Bo Xu, Xiufeng Shi, Zhehuan Zhao, and Wei Zheng. 2018. Leveraging biomedical resources in bi-lstm for drug-drug interaction extraction. *IEEE Access*, 6:33432–33439.

Bishan Yang and Tom Mitchell. 2017. Leveraging knowledge bases in lstms for improving machine reading. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1436–1446.

Wenhao Yu, Zongze Li, Qingkai Zeng, and Meng Jiang. 2019. Tablepedia: Automating pdf table reading in an experimental evidence exploration and analytic system. In *The World Wide Web Conference*, pages 3615–3619. ACM.

Suncong Zheng, Feng Wang, Hongyun Bao, Yuexing Hao, Peng Zhou, and Bo Xu. 2017. Joint extraction of entities and relations based on a novel tagging scheme. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1227–1236.

# Cross-lingual Structure Transfer for Relation and Event Extraction

**Ananya Subburathinam[1]*, Di Lu[1]* , Heng Ji[2],**
**Jonathan May[3], Shih-Fu Chang[4], Avirup Sil[5], Clare Voss[6]**
[1]Computer Science Department, Rensselaer Polytechnic Institute
[2]Computer Science Department, University of Illinois at Urbana-Champaign
[3]University of Southern California Information Sciences Institute
[4]Electrical Engineering Department, Columbia University
[5]IBM T.J. Watson Research Center
[6]Army Research Laboratory
hengji@illinois.edu

## Abstract

The identification of complex semantic structures such as events and entity relations, already a challenging Information Extraction task, is doubly difficult from sources written in under-resourced and under-annotated languages. We investigate the suitability of cross-lingual structure transfer techniques for these tasks. We exploit relation- and event-relevant language-universal features, leveraging both symbolic (including part-of-speech and dependency path) and distributional (including type representation and contextualized representation) information. By representing all entity mentions, event triggers, and contexts into this complex and structured multilingual common space, using graph convolutional networks, we can train a relation or event extractor from source language annotations and apply it to the target language. Extensive experiments on cross-lingual relation and event transfer among English, Chinese, and Arabic demonstrate that our approach achieves performance comparable to state-of-the-art supervised models trained on up to 3,000 manually annotated mentions: up to 62.6% F-score for Relation Extraction, and 63.1% F-score for Event Argument Role Labeling. The event argument role labeling model transferred from English to Chinese achieves similar performance as the model trained from Chinese. We thus find that language-universal symbolic and distributional representations are complementary for cross-lingual structure transfer.

## 1 Introduction

Advanced Information Extraction (IE) tasks entail predicting structures, such as relations between entities, and events involving entities. Given a pair of entity mentions, *Relation Extraction* aims to identify the relation between the mentions and classify it by predefined type. *Event Extraction* aims to identify event triggers and their arguments in unstructured texts and classify them respectively by predefined types and roles. As Figure 1 illustrates, both tasks entail predicting an information network (Li et al., 2014) for each sentence, where the entity mentions and event triggers are nodes, and the relations and event-argument links are edges labeled with their relation and argument roles, respectively.

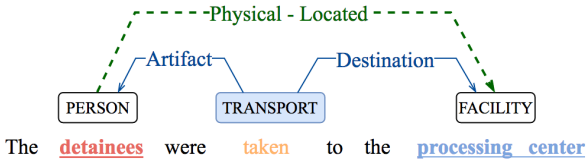


Figure 1: Information Network example constructed by Event (blue) and Relation (green dashed) Extraction.

There are certain relations and events that are of primary interest to a given community and so are reported predominantly in the low-resource language data sources available to that community. For example, though English language news will occasionally discuss the *Person* Aung San Suu Kyi, the vast majority of *Physical-Located* relations and *Meeting* events involving her are only reported locally in Burmese news, and thus, without Burmese relation and event extraction, a knowledge graph of this person will lack this available information. Unfortunately, publicly-available gold-standard annotations for relation and event extraction exist for only a few languages (Doddington et al., 2004; Getman et al., 2018), and Burmese is not among them. Compared to other IE tasks such as name tagging, the annotations for Relation and Event Extraction are also more costly to obtain, because they are structured and require a rich label space.

Recent research (Lin et al., 2017) has found that relational facts are typically expressed by identifiable patterns within languages and has shown



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 313–325,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

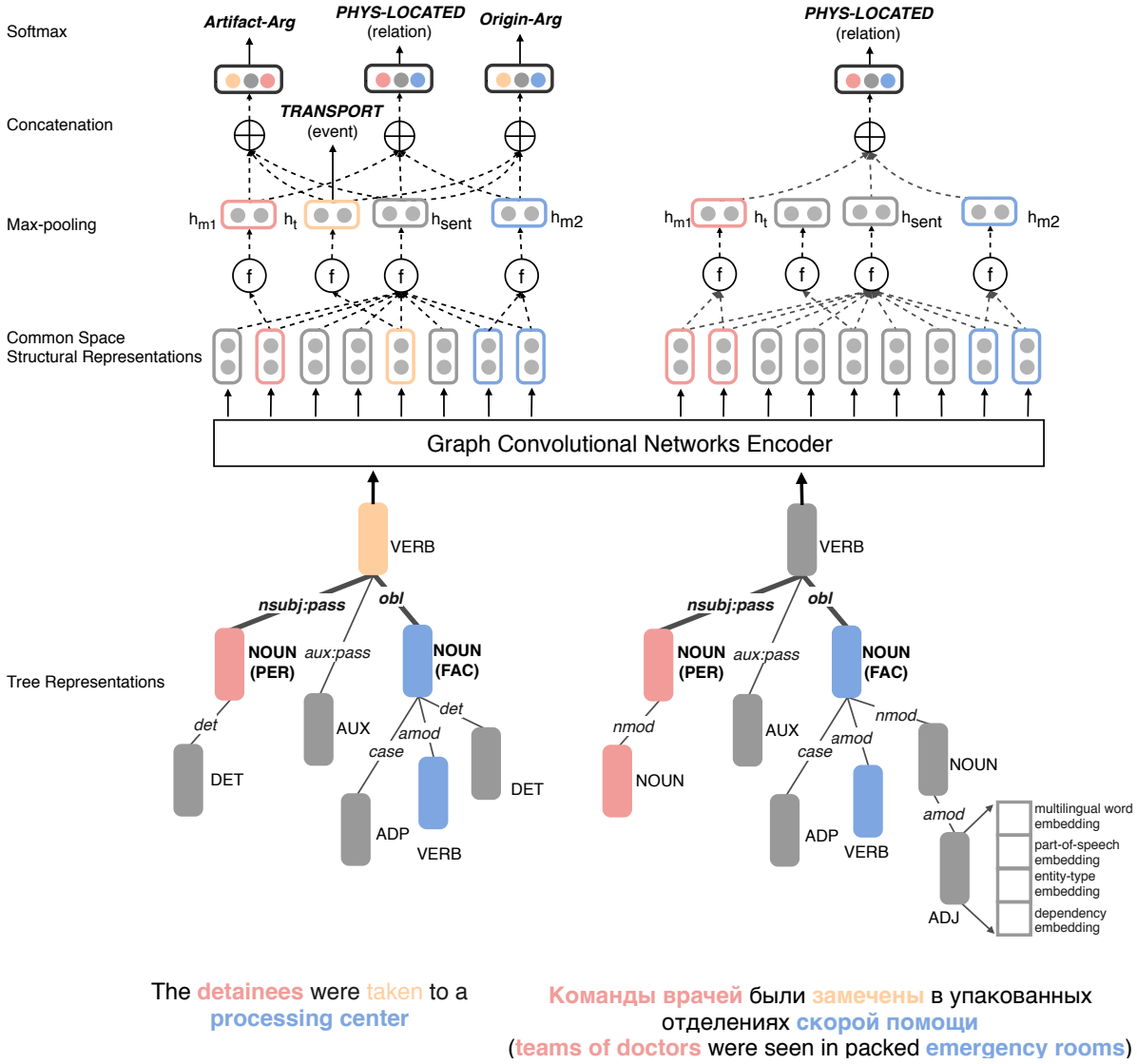


Figure 2: Multilingual common semantic space and cross-lingual structure transfer.

that the consistency in patterns observed across languages can be used to improve relation extraction. Inspired by their results, we exploit language-universal features relevant to relation and event argument identification and classification, by way of both symbolic and distributional representations. For example, language-universal POS tagging and universal dependency parsing is available for 76 languages (Nivre et al., 2018), entity extraction is available for 282 languages (Pan et al., 2017), and multi-lingual word embeddings are available for 44 languages (Bojanowski et al., 2017; Joulin et al., 2018). As shown in Figure 2, even for distinct pairs of entity mentions (colored pink and blue, in both English and Russian), the structures share similar language-universal symbolic features, such as a common labeled dependency path, as well as distributional features, such as multilingual word embeddings.

Based on these language-universal representations, we then project all entity mentions, event triggers and their contexts into one multilingual common space. Unlike recent work on multilingual common space construction that makes use of linear mappings (Mikolov et al., 2013; Rothe et al., 2016; Zhang et al., 2016; Lazaridou et al., 2015; Xing et al., 2015; Smith et al., 2017) or canonical correlation analysis (CCA) (Ammar et al., 2016; Faruqui and Dyer, 2014; Lu et al., 2015) to transfer *surface features* across languages, our major innovation is to convert the text data into structured representations derived from universal dependency parses and enhanced with distributional information to capture individual entities as well

as the relations and events involving those entities, so we can *share structural representations* across multiple languages.

Then we construct a novel cross-lingual structure transfer learning framework to project source language (SL) training data and target language (TL) test data into the common semantic space, so that we can train a relation or event extractor from SL annotations and apply the resulting extractor to TL texts. We adopt graph convolutional networks (GCN) (Kipf and Welling, 2017; Marcheggiani and Titov, 2017) to encode graph structures over the input data, applying graph convolution operations to generate entity and word representations in a latent space. In contrast to other encoders such a Tree-LSTM (Tai et al., 2015), GCN can cover more complete contextual information from dependency parses because, for each word, it captures all parse tree neighbors of the word, rather than just the child nodes of the word. Using this shared encoder, we treat the two tasks of relation extraction and event argument role labeling as mappings from the latent space to relation type and to event type and argument role, respectively.

Extensive experiments on cross-lingual relation and event transfer among English, Chinese, and Arabic show that our approach achieves highly promising performance on both tasks.

## 2 Model

### 2.1 Overview

Our cross-lingual structure transfer approach (see Figure 2) consists of four steps: (1) Convert each sentence in any language into a language-universal tree structure based on universal dependency parsing. (2) For each node in the tree structure, create a representation from the concatenation of multilingual word embedding, language-universal POS embedding, dependency role embedding and entity-type embedding, so that all sentences, independent of their language, are represented within one shared semantic space. (3) Adopt GCN to generate contextualized word representations by leveraging information from neighbors derived from the dependency parsing tree, and (4) Using this shared semantic space, train relation and event argument extractors with high-resource language training data, and apply the resulting extractors to texts of low-resource languages that do not have any relation or event argument annotations.

### 2.2 Tree Representations

Most previous approaches regard a sentence as a linear sequence of words, and incorporate language-specific information such as word order. Unlike sequence representations, tree representations such as constituency trees and dependency trees are typically constructed following a combination of syntactic principles and annotation guidelines designed by linguists. The resulting structures, such as the `verb` – `subject` relation and the `verb` – `object` relation, are found across languages. In this paper, we choose dependency trees as the sentence representations because the community has made great efforts at developing language-universal dependency parsing resources across 83 languages (Nivre et al., 2016).

We define the dependency-based tree representation for a sentence as $G = (V, E)$, where $V = \{v_1, v_2, ..., v_N\}$ is a set of words, and $E = \{e_1, e_2, ..., e_M\}$ is a set of language-universal syntactic relations. $N$ is the number of words in the sentence and $M$ is the number of dependency relations between words. To make this tree representation language-universal, we first convert each tree node into a vector which is a concatenation of three language-universal representations at word-level: multilingual word embedding, POS embedding (Nivre et al., 2016), entity-type embedding, and dependency relation embedding. More details are reported in Section 3.2.

### 2.3 GCN Encoder

Structural information is important for relation extraction and event argument role labeling, thus we aim to generate contextualized word representations by leveraging neighbors in dependency trees for each node.

Our GCN encoder is based on the monolingual design by Zhang et al. (2018b). The graphical sentence representation obtained from dependency parsing of a sentence with $N$ tokens is converted into an $N \times N$ adjacency matrix $\boldsymbol{A}$, with added self-connections at each node to help capture information about the current node itself, as in Kipf and Welling (2017). Here, $A_{i,j} = 1$ denotes the presence of a directed edge from node $i$ to node $j$ in the dependency tree. Initially, each node contains distributional information about the $i^{th}$ word, including word embedding $\boldsymbol{x}_i^w$, embeddings for symbolic information including its POS tag $\boldsymbol{x}_i^p$, dependency relation $\boldsymbol{x}_i^d$ and entity type $\boldsymbol{x}_i^e$.

We represent this initial representation $\boldsymbol{h}_i^0$ as follows:

$$\boldsymbol{h}_i^{(0)} = \boldsymbol{x}_i^w \oplus \boldsymbol{x}_i^p \oplus \boldsymbol{x}_i^d \oplus \boldsymbol{x}_i^e$$

Then, at the $k^{th}$ layer of convolution, the hidden representation is derived from the representations of its neighbors at the $(k-1)^{th}$ layer. Thus, the hidden representation at the $k^{th}$ layer for the $i^{th}$ node is computed as:

$$\boldsymbol{h}_i^{(k)} = \text{ReLU}\left(\sum_{j=0}^{N} \frac{\boldsymbol{A_{ij}} \boldsymbol{W}^{(k)} \boldsymbol{h}_j^{(k-1)}}{d_i + b^{(k)}}\right)$$

where $\boldsymbol{W}$ is the weight vector, $b$ is the bias term, and $d_i$ represents the degree of the $i^{th}$ node. The denominator in the equation denotes the normalization factor to neutralize the negative impact of node degree (Zhang et al., 2018b). The final hidden representation of each node after the $k^{th}$ layer is the encoding of each word $\boldsymbol{h}_i^{(k)}$ in our language universal common space, and incorporates information about neighbors up to $k$ hops away in the dependency tree.

### 2.4 Application on Relation Extraction

The GCN encoder generates the final hidden representation, $h_i^{(k)}$, for each of the $n$ nodes. We perform max-pooling over these final node representations to obtain a single vector representation for the sentence, $h^s$. Similar to previous work (Zhang et al., 2018b), we also use the following recipe to obtain relation type classification results for each mention pair in a sentence: (1) max-pooling over the final representations of the nodes representing entity mentions, to get a single vector representation for each mention in a pair under consideration, $\boldsymbol{h}^{m_1}$ and $\boldsymbol{h}^{m_2}$, (2) a concatenation of three results of max-pooling: $([\boldsymbol{h}^{m_1}; \boldsymbol{h}^s; \boldsymbol{h}^{m_2}])$ to combine contextual sentence information with entity mention information (Santoro et al., 2017; Lee et al., 2017; Zhang et al., 2018b), (3) a linear layer to generate a combined representation of these concatenated results, and (4) a Softmax output layer for relation type classification. The objective function used here is as follows:

$$L^r = \sum_{i=1}^{N} \sum_{j=1}^{L_i} y_{ij} \ \log(\sigma(\boldsymbol{U}^r \cdot [\boldsymbol{h}_i^{m1}; \boldsymbol{h}_{ij}^s; \boldsymbol{h}_j^{m2}])) \quad (1)$$

where $\boldsymbol{U}^r$ is a weight matrix.

### 2.5 Application on Event Argument Role Labeling

Event argument role labeling distinguishes arguments from non-arguments and classifies arguments by argument role. To label the role of an argument candidate ($x_j^a$) for an event trigger ($x_i^t$), we first generate the sentence representation $\boldsymbol{h}^s$, argument candidate representation $\boldsymbol{h}^a$, and trigger representation $\boldsymbol{h}^t$ by applying pooling on sentence, argument candidate $x_j^a$ and event trigger $x_i^t$ respectively, which is the same as that for relation extraction. The mapping function from the latent space to argument roles is composed of a concatenation operation ($[\boldsymbol{h}^t; \boldsymbol{h}^s; \boldsymbol{h}^a]$), a linear layer ($\boldsymbol{U}^a$) and a Softmax output layer:

$$L^a = \sum_{i=1}^{N} \sum_{j=1}^{L_i} y_{ij} \ \log(\sigma(\boldsymbol{U}^a \cdot [\boldsymbol{h}_i^t; \boldsymbol{h}_{ij}^s; \boldsymbol{h}_j^a])) \quad (2)$$

where $N$ is the number of event mentions, $L_i$ is the number of argument candidates for $i^{th}$ event mention and $\sigma$ is the Sigmoid function.

## 3 Experiments

### 3.1 Data and Evaluation Metrics

| | Relation Mentions | Event Mentions | Event Arguments |
|---|---|---|---|
| English | 8,738 | 5,349 | 9,793 |
| Chinese | 9,317 | 3,333 | 8,032 |
| Arabic | 4,731 | 2,270 | 4,975 |

Table 1: Data statistics.

We choose the Automatic Content Extraction (ACE) 2005 corpus (Walker et al., 2006) for our experiments because it contains the most comprehensive gold-standard relation and event annotations for three distinct languages: English, Chinese and Arabic (see Table 1). Our target ontology includes the 7 entity types, 18 relation subtypes and 33 event subtypes defined in ACE. We randomly choose 80% of the corpus for training, 10% for development and 10% for blind test. We downsample the negative training instances by limiting the number of negative samples to be no more than the number of positive samples for each document.

For data preprocessing, we apply the Stanford CoreNLP toolkit (Manning et al., 2014) for Chinese word segmentation and English tokenization, and the API provided by UDPipe (Straka and

Straková, 2017) for Arabic tokenization. We use UDPipe[1] for POS tagging and dependency parsing for all three languages.

We follow the following criteria in previous work (Ji and Grishman, 2008; Li et al., 2013; Li and Ji, 2014) for evaluation:

- A relation mention is considered correct if its relation type is correct, and the head offsets of the two related entity mention arguments are both correct.
- An event argument is correctly labeled if its event type, offsets, and role label match any of the reference argument mentions.

### 3.2 Training Details

We use the multilingual word embeddings released by (Joulin et al., 2018) which are based on aligning monolingual embeddings learned by fastText (Bojanowski et al., 2017) from Wikipedia. We adopt the universal POS tag set (17 word categories) and 37 categories of dependency relations defined by the *Universal Dependencies* program (Nivre et al., 2016), and seven entity types defined in ACE: person, geopolitical entity, organization, facility, location, weapon and vehicle. Table 2 shows the hyperparameters of our model.

| **Hyperparameter** | **Value** |
|---|---|
| word embedding size | 300 |
| POS embedding size | 30 |
| entity embedding size | 30 |
| dependency relation embedding size | 30 |
| hidden dimension size | 200 |
| dropout | 0.5 |
| number of layers | 2 |
| pooling function | max pooling |
| mlp layers | 2 |
| learning rate | 0.1 |
| learning rate decay | 0.9 |
| batch size | 50 |
| optimization | SGD |

Table 2: GCN hyperparameters.

### 3.3 Overall Performance

In order to fully analyze the cross-lingual learning capabilities of our framework, we assess its performance by applying models trained with various combinations of training and test data from these three languages, as shown in Tables 3 and 4. We can see that the results are promising. For both tasks, the models trained from English are best, followed by Chinese, and then Arabic. We find that extraction task performance degrades as the accuracy of language-dependent tools (for sentence segmentation, POS tagging, dependency parsing) degrades.

Using English as training data, our cross-lingual transfer approach achieves similar performance on Chinese event argument role labeling (59.0%), compared to the model trained from Chinese annotations (59.3%), which is much higher than the best reported English-to-Chinese transfer result on event argument role labeling (47.7%) (Hsi et al., 2016).

We also show polyglot results for event argument role labeling in Table 4, by combing the training data from multiple languages. We observe that our model benefits from the combination of training data of multiple languages. The polyglot transfer learning does not provide further gains to relation extraction because the model converges quickly on a small amount of training data.

| Train \ Test | English | Chinese | Arabic |
|---|---|---|---|
| English | 68.2 | **42.5** | **58.7** |
| Chinese | **62.6** | 69.4 | 54.0 |
| Arabic | 58.6 | 35.2 | 67.4 |

Table 3: Relation Extraction: overall performance (F-score %) using perfect entity mentions.

| Train \ Test | English | Chinese | Arabic |
|---|---|---|---|
| English | 63.9 | **59.0** | **61.8** |
| Chinese | **51.6** | 59.3 | 60.6 |
| Arabic | 43.1 | 50.1 | 64.0 |
| English + Chinese | – | – | **63.1** |
| English + Arabic | – | **60.1** | – |
| Chinese + Arabic | **51.9** | – | – |

Table 4: Event Argument Role Labeling results (F1 %) with perfect event triggers.

[1]Universal Dependencies 2.3 models: english-ewt-ud-2.3-181115.udpipe, chinese-gsd-ud-2.3-181115.udpipe, arabic-padt-ud-2.3-181115.udpipe

| Feature | Chinese | Arabic |
|---|---|---|
| Multilingual Word Embedding | 40.2 | 53.8 |
| + POS Embedding | 40.1 | 54.9 |
| + Entity Embedding | 40.8 | 56.9 |
| + Dependency Relation Embedding | **42.5** | **58.7** |

Table 5: Relation Extraction: Contribution of adding each language-universal feature consecutively for relation extraction on Arabic and Chinese test sets, using the model trained on English (F1 scores (%)).

### 3.4 Ablation Study

Tables 5 and 6 show the impact of each feature category. We can see that all categories help except Chinese POS features for Relation Extraction and Arabic POS features for Event Argument Role Labeling.

Many Chinese word segmentation errors occur on tokens involved in names. For example, "总统(NOUN, president) 萨姆(PROPN, Samuel) ·(PUNCT) 努乔马(PROPN, Nujoma)" is mistakenly tagged as "总(NOUN) 统萨姆(PROPN) ·(PUNCT) 努乔(NOUN) 马(NOUN)".

In Arabic, sometimes one word is a combination of *Noun* and *Verb*. For example, the single word "الإسرائيلي" means *"Israeli conflict"* in English, including both a trigger and an argument, which are not separated by our tokenizer. In contrast, two entity mentions are unlikely to be combined into one word in Arabic, thus Relation Extraction does not suffer from tokenization errors and corresponding POS features.

| Feature | Chinese | Arabic |
|---|---|---|
| Multilingual Word Embedding | 52.4 | 56.6 |
| + POS Embedding | 57.1 | 37.8 |
| + Entity Embedding | 58.3 | 41.4 |
| + Dependency Relation Embedding | **59.0** | **61.8** |

Table 6: Event Extraction: Contributions of adding each language-universal feature consecutively for event argument role labeling on Arabic and Chinese test datasets, using a model trained on English (F1 scores (%)).

### 3.5 Comparison with Supervised Approach

We also compare the results with supervised monolingual models trained from manual annotations in the same language. Figures 3 and 4 show the learning curves of these supervised models. For event argument role labeling, we can see that without using any annotations for the target language, our approach achieves performance comparable to the supervised models trained from more than 3,000 manually annotated event argument mentions, which equal to approximately 1,326 Chinese sentences and 1,141 Arabic sentences based on the statistics of ACE data.

Our model performs particularly well on relation types or argument roles that require deep understanding of wide contexts involving complex syntactic and semantic structures, such as *PART-WHOLE:Artifact*, *PART-WHOLE:Geographical*, *GEN-AFF:Org-Location* and *ORG-AFF:Employment* relations, and *Injure: Victim* argument role. Despite only having 14 training instances, our model achieves near 100% F-score on *PART-WHOLE:Artifact* relations when transferred from English to Chinese. Our model achieves 86% F1 score on *PART-WHOLE:Geographical* relations when transferred from English to Arabic, and 73% and 79% F1 scores on *GEN-AFF:Org-Location* relations when transferred from English to Chinese and to Arabic, respectively. In an *Injure* event, a *Person* can either be an *Agent* or a *Victim*. Surface lexical embedding features are often not sufficient to disambiguate them. Our model is effective at transferring structural information such as dependency relations between words, and obtains 72.97% F1 score on labeling *Injure: Victim* when transferred from English to Chinese, and 75.43 % from English to Arabic.

In addition, our model achieves very high performance on event argument roles for which entity type is a strong indicator. For example, a *weapon* is much more likely to play as an *Instrument* rather than a *Target* in an *Attack*. Our model achieves 89.9% F1 score on *Attack: Instrument* and 91.4% F1 score on *Personnel: POSITION* argument roles when transferred from English to Chinese.

### 3.6 Using System Extracted Name Mentions

| Target Language | F1 score |
|---|---|
| Chinese | 56.9 |
| Arabic | 60.1 |

Table 7: Event Argument Role Labeling results (F1 %) on Chinese and Arabic using English as training data (with system generated entity mentions)

Table 7 shows the results of event argument role labeling on Chinese and Arabic entity mentions automatically extracted by Stanford CoreNLP instead of manually annotated mentions. The system extracted entity mentions introduce noise and

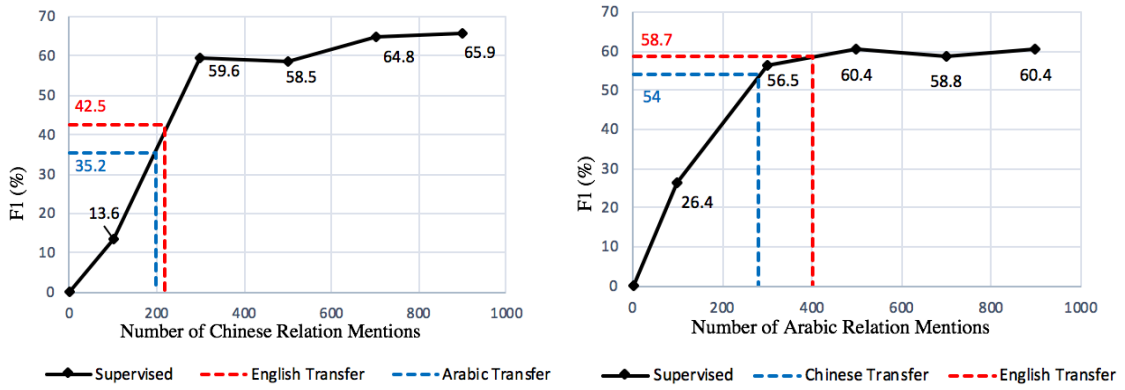


(a) Target Language: Chinese (b) Target Language: Arabic

Figure 3: Relation Extraction: Comparison between supervised monolingual GCN model and cross-lingual transfer learning.

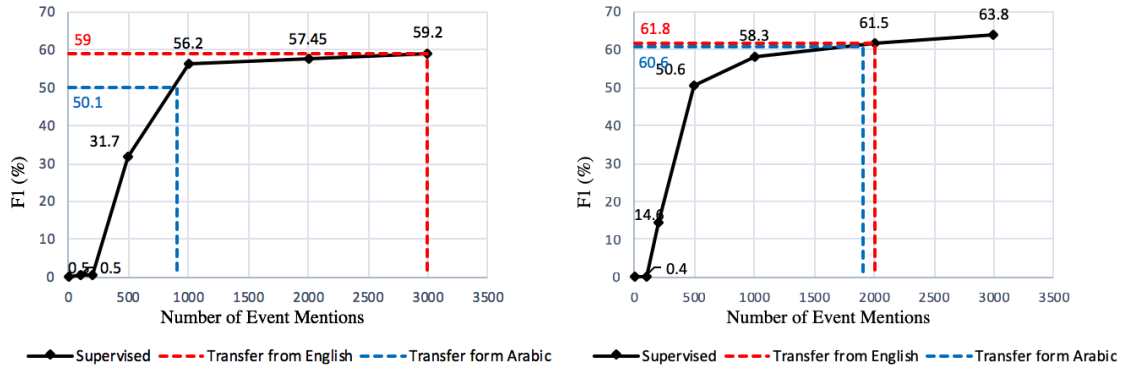


(a) Target Language: Chinese (b) Target Language: Arabic

Figure 4: Event Argument Role Labeling: Comparison between supervised monolingual GCN model and cross-lingual transfer learning.

thus decrease the performance of the model, but the overall results are still promising.

## 4 Related Work

A large number of supervised machine learning techniques have been used for English event extraction, including traditional techniques based on symbolic features (Ji and Grishman, 2008; Liao and Grishman, 2011), joint inference models (Li et al., 2014; Yang and Mitchell, 2016), and recently with neural networks (Nguyen and Grishman, 2015a; Nguyen et al., 2016; Chen et al., 2015; Nguyen and Grishman, 2018; Liu et al., 2018b; Lu and Nguyen, 2018; Liu et al., 2018a; Zhang et al., 2018a, 2019). English relation extraction in the early days also followed supervised paradigms (Li and Ji, 2014; Zeng et al., 2014; Nguyen and Grishman, 2015b; Miwa and Bansal, 2016; Pawar et al., 2017; Bekoulis et al., 2018; Wang et al., 2018b). Recent efforts have attempted to reduce annotation costs using distant supervision (Mintz et al., 2009; Surdeanu et al., 2012; Min et al., 2013; Angeli et al., 2014; Zeng et al., 2015; Quirk and Poon, 2017; Qin et al., 2018; Wang et al., 2018a) through knowledge bases (KBs), where entities and static relations are plentiful. Distant supervision is less applicable for the task of event extraction because very few dynamic events are included in KBs. These approaches, however, incorporate language-specific characteristics and thus are costly in requiring substantial amount of annotations to adapt to a new language (Chen and Vincent, 2012; Blessing and Schütze, 2012; Li et al., 2012; Danilova et al., 2014; Agerri et al., 2016; Hsi et al., 2016; Feng et al., 2016).

Regardless of the recent successes in applying cross-lingual transfer learning to sequence labeling tasks, such as name tagging (e.g., (Mayhew et al., 2017; Lin et al., 2018; Huang et al., 2019)), only limited work has explored cross-lingual relation and event structure transfer. Most previous efforts working with cross-lingual structure trans-

fer rely on bilingual dictionaries (Hsi et al., 2016), parallel data (Chen and Ji, 2009; Kim et al., 2010; Qian et al., 2014) or machine translation (Faruqui and Kumar, 2015; Zou et al., 2018). Recent methods (Lin et al., 2017; Wang et al., 2018b) aggregate consistent patterns and complementary information across languages to enhance Relation Extraction, but they do so exploiting only distributional representations.

Event extraction shares with Semantic Role Labeling (SRL) the task of assigning to each event argument its event role label, in the process of completing other tasks to extract the full event structure (assigning event types to predicates and more specific roles to arguments). Cross-lingual transfer has been very successful for SRL. Early attempts relied on word alignment (Van der Plas et al., 2011) or bilingual dictionaries (Kozhevnikov and Titov, 2013). Recent work incorporates universal dependencies (Prazák and Konopík, 2017) or multilingual word embeddings for Polyglot SRL (Mulcaire et al., 2018). Liu et al. (2019) and Mulcaire et al. (2019) exploit multi-lingual contextualized word embedding for SRL and other Polyglot NLP tasks including dependency parsing and name tagging. To the best of our knowledge, our work is the first to construct a cross-lingual structure transfer framework that combines language-universal symbolic representations and distributional representations for relation and event extraction over texts written in a language without any training data.

GCN has been successfully applied to several individual monolingual NLP tasks, including relation extraction (Zhang et al., 2018b), event detection (Nguyen and Grishman, 2018), SRL (Marcheggiani and Titov, 2017) and sentence classification (Yao et al., 2019). We apply GCN to construct multi-lingual structural representations for cross-lingual transfer learning.

## 5 Conclusions and Future Work

We show how cross-lingual relation and event argument structural representations may be transferred between languages without any training data for the target language, and conclude that language-universal symbolic and distributional representations are complementary for cross-lingual structure transfer. In the future we will explore more language-universal representations such as visual features from topically-related images and videos and external background knowledge.


## Acknowledgement

This research is based upon work supported in part by U.S. DARPA LORELEI Program HR0011-15-C-0115, the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via contract FA8650-17-C-9116, and ARL NS-CTA No. W911NF-09-2-0053. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of DARPA, ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.


## References


Rodrigo Agerri, Itziar Aldabe, Egoitz Laparra, German Rigau Claramunt, Antske Fokkens, Paul Huijgen, Rubén Izquierdo Beviá, Marieke van Erp, Piek Vossen, Anne-Lyse Minard, et al. 2016. Multilingual event detection using the newsreader pipelines. In *Proceedings of International Conference on Language Resources and Evaluation 2016*.

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A Smith. 2016. Massively multilingual word embeddings. *arXiv preprint arXiv:1602.01925*.

Gabor Angeli, Julie Tibshirani, Jean Wu, and Christopher D Manning. 2014. Combining distant and partial supervision for relation extraction. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Giannis Bekoulis, Johannes Deleu, Thomas Demeester, and Chris Develder. 2018. Adversarial training for multi-context joint entity and relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2830–2836, Brussels, Belgium. Association for Computational Linguistics.

Andre Blessing and Hinrich Schütze. 2012. Crosslingual distant supervision for extracting relations of different complexity. In *Proceedings of the 21st ACM international conference on Information and knowledge management*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*.

Chen Chen and Ng Vincent. 2012. Joint modeling for Chinese event extraction with rich linguistic features. In *Proceedings of the 24th International Conference on Computational Linguistics*.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Zheng Chen and Heng Ji. 2009. Can one language bootstrap the other: a case study on event extraction. In *Proceedings of the NAACL HLT 2009 Workshop on Semi-Supervised Learning for Natural Language Processing*.

Vera Danilova, Mikhail Alexandrov, and Xavier Blanco. 2014. A survey of multilingual event extraction from text. In *Proceedings of the International Conference on Applications of Natural Language to Data Bases/Information Systems*.

George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie M Strassel, and Ralph M Weischedel. 2004. The automatic content extraction (ACE) program-tasks, data, and evaluation. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation*.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*.

Manaal Faruqui and Shankar Kumar. 2015. Multilingual open relation extraction using cross-lingual projection. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Xiaocheng Feng, Lifu Huang, Duyu Tang, Heng Ji, Bing Qin, and Ting Liu. 2016. A language-independent neural network for event detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.

Jeremy Getman, Joe Ellis, Stephanie Strassel, Zhiyi Song, and Jennifer Tracey. 2018. Laying the groundwork for knowledge base population: Nine years of linguistic resources for TAC KBP. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*.

Andrew Hsi, Yiming Yang, Jaime Carbonell, and Ruochen Xu. 2016. Leveraging multilingual training for limited resource event extraction. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*.

Lifu Huang, Heng Ji, and Jonathan May. 2019. Cross-lingual multi-level adversarial transfer to enhance low-resource name tagging. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

Armand Joulin, Piotr Bojanowski, Tomas Mikolov, Hervé Jégou, and Edouard Grave. 2018. Loss in translation: Learning bilingual word mapping with a retrieval criterion. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Seokhwan Kim, Minwoo Jeong, Jonghoon Lee, and Gary Geunbae Lee. 2010. A cross-lingual annotation projection approach for relation detection. In *Proceedings of the 23rd International Conference on Computational Linguistics*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

Mikhail Kozhevnikov and Ivan Titov. 2013. Cross-lingual transfer of semantic role labeling models. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*.

Angeliki Lazaridou, Georgiana Dinu, and Marco Baroni. 2015. Hubness and pollution: Delving into cross-space mapping for zero-shot learning. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Peifeng Li, Guodong Zhou, Qiaoming Zhu, and Libin Hou. 2012. Employing compositional semantics and discourse consistency in chinese event extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Qi Li, Heng Ji, HONG Yu, and Sujian Li. 2014. Constructing information networks using one single model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Shasha Liao and Ralph Grishman. 2011. Acquiring topic features to improve event extraction: in preselected and balanced collections. In *Proceedings of the International Conference Recent Advances in Natural Language Processing 2011*.

Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2017. Neural relation extraction with multi-lingual attention. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Ying Lin, Shengqi Yang, Veselin Stoyanov, and Heng Ji. 2018. A multi-lingual multi-task architecture for low-resource sequence labeling. In *Proc. The 56th Annual Meeting of the Association for Computational Linguistics*.

Jian Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2018a. Event detection via gated multilingual attention mechanism. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. Linguistic knowledge and transferability of contextual representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*.

Xiao Liu, Zhunchen Luo, and Heyan Huang. 2018b. Jointly multiple events extraction via attention-based graph information aggregation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Ang Lu, Weiran Wang, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Deep multilingual correlation for improved word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Weiyi Lu and Thien Huu Nguyen. 2018. Similar but not the same: Word sense disambiguation improves event detection via neural representation matching. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd annual meeting of the Association for Computational Linguistics: system demonstrations*.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Stephen Mayhew, Chen-Tse Tsai, and Dan Roth. 2017. Cheap translation for cross-lingual named entity recognition. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Bonan Min, Ralph Grishman, Li Wan, Chang Wang, and David Gondek. 2013. Distant supervision for relation extraction with an incomplete knowledge base. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using LSTMs on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Phoebe Mulcaire, Jungo Kasai, and Noah A. Smith. 2019. Polyglot contextual representations improve crosslingual transfer. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*.

Phoebe Mulcaire, Swabha Swayamdipta, and Noah A. Smith. 2018. Polyglot semantic role labeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Thien Huu Nguyen and Ralph Grishman. 2015a. Event detection and domain adaptation with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*.

Thien Huu Nguyen and Ralph Grishman. 2015b. Relation extraction: Perspective from convolutional neural networks. In *Proceedings of North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Thien Huu Nguyen and Ralph Grishman. 2018. Graph convolutional networks with argument-aware pooling for event detection. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Joakim Nivre, Mitchell Abrams, Željko Agić, Lars Ahrenberg, Lene Antonsen, Katya Aplonova, Maria Jesus Aranzabe, Gashaw Arutie, Masayuki Asahara, Luma Ateyah, Mohammed Attia, Aitziber Atutxa, Liesbeth Augustinus, Elena Badmaeva, Miguel Ballesteros, Esha Banerjee, Sebastian Bank, Verginica Barbu Mititelu, Victoria Basmov, John Bauer, Sandra Bellato, Kepa Bengoetxea, Yevgeni Berzak, Irshad Ahmad Bhat, Riyaz Ahmad Bhat, Erica Biagetti, Eckhard Bick, Rogier Blokland, Victoria Bobicev, Carl Börstell, Cristina Bosco, Gosse Bouma, Sam Bowman, Adriane Boyd, Aljoscha Burchardt, Marie Candito, Bernard Caron, Gauthier Caron, Gülşen Cebiroğlu Eryiğit, Flavio Massimiliano Cecchini, Giuseppe G. A. Celano, Slavomír Čéplö, Savas Cetin, Fabricio Chalub, Jinho Choi, Yongseok Cho, Jayeol Chun, Silvie Cinková, Aurélie Collomb, Çağrı Çöltekin, Miriam Connor, Marine Courtin, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Arantza Diaz de Ilarraza, Carly Dickerson, Peter Dirix, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Marhaba Eli, Ali Elkahky, Binyam Ephrem, Tomaž Erjavec, Aline Etienne, Richárd Farkas, Hector Fernandez Alcalde, Jennifer Foster, Cláudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Moa Gärdenfors, Sebastian Garza, Kim Gerdes, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökırmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Céline Guillot-Barbance, Nizar Habash, Jan Hajič, Jan Hajič jr., Linh Hà Mỹ, Na-Rae Han, Kim Harris, Dag Haug, Barbora Hladká, Jaroslava Hlaváčová, Florinel Hociung, Petter Hohle, Jena Hwang, Radu Ion, Elena Irimia, Ọlájídé Ishola, Tomáš Jelínek, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşıkara, Sylvain Kahane, Hiroshi Kanayama, Jenna Kanerva, Boris Katz, Tolga Kayadelen, Jessica Kenney, Václava Kettnerová, Jesse Kirchner, Kamil Kopacewicz, Natalia Kotsyba, Simon Krek, Sookyoung Kwak, Veronika Laippala, Lorenzo Lambertino, Lucia Lam, Tatiana Lando, Septina Dian Larasati, Alexei Lavrentiev, John Lee, Phương Lê Hồng, Alessandro Lenci, Saran Lertpradit, Herman Leung, Cheuk Ying Li, Josie Li, Keying Li, KyungTae Lim, Nikola Ljubešić, Olga Loginova, Olga Lyashevskaya, Teresa Lynn, Vivien Macketanz, Aibek Makazhanov, Michael Mandl, Christopher Manning, Ruli Manurung, Cătălina Mărănduc, David Mareček, Katrin Marheinecke, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Gustavo Mendonça, Niko Miekka, Margarita Misirpashayeva, Anna Missilä, Cătălin Mititelu, Yusuke Miyao, Simonetta Montemagni, Amir More, Laura Moreno Romero, Keiko Sophie Mori, Shinsuke Mori, Bjartur Mortensen, Bohdan Moskalevskyi, Kadri Muischnek, Yugo Murawaki, Kaili Müürisep, Pinkey Nainwani, Juan Ignacio Navarro Horñiacek, Anna Nedoluzhko, Gunta Nešpore-Bērzkalne, Lương Nguyễn Thị, Huyền Nguyễn Thị Minh, Vitaly Nikolaev, Rattima Nitisaroj, Hanna Nurmi, Stina Ojala, Adédayọ̀ Olúòkun, Mai Omura, Petya Osenova, Robert Östling, Lilja Øvrelid, Niko Partanen, Elena Pascual, Marco Passarotti, Agnieszka Patejuk, Guilherme Paulino-Passos, Siyao Peng, Cenel-Augusto Perez, Guy Perrier, Slav Petrov, Jussi Piitulainen, Emily Pitler, Barbara Plank, Thierry Poibeau, Martin Popel, Lauma Pretkalniņa, Sophie Prévost, Prokopis Prokopidis, Adam Przepiórkowski, Tiina Puolakainen, Sampo Pyysalo, Andriela Rääbis, Alexandre Rademaker, Loganathan Ramasamy, Taraka Rama, Carlos Ramisch, Vinit Ravishankar, Livy Real, Siva Reddy, Georg Rehm, Michael Rießler, Larissa Rinaldi, Laura Rituma, Luisa Rocha, Mykhailo Romanenko, Rudolf Rosa, Davide Rovati, Valentin Ro□ca, Olga Rudina, Jack Rueter, Shoval Sadde, Benoît Sagot, Shadi Saleh, Tanja Samardžić, Stephanie Samson, Manuela Sanguinetti, Baiba Saulīte, Yanin Sawanakunanon, Nathan Schneider, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Mo Shen, Atsuko Shimada, Muh Shohibussirri, Dmitry Sichinava, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Isabela Soares-Bastos, Carolyn Spadine, Antonio Stella, Milan Straka, Jana Strnadová, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Dima Taji, Yuta Takahashi, Takaaki Tanaka, Isabelle Tellier, Trond Trosterud, Anna Trukhina, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Zdeňka Urešová, Larraitz Uria, Hans Uszkoreit, Sowmya Vajjala, Daniel van Niekerk, Gertjan van Noord, Viktor Varga, Eric Villemonte de la Clergerie, Veronika Vincze, Lars Wallin, Jing Xian Wang, Jonathan North Washington, Seyi Williams, Mats Wirén, Tsegay Woldemariam, Tak-sum Wong, Chunxiao Yan, Marat M. Yavrumyan, Zhuoran Yu, Zdeněk Žabokrtský, Amir Zeldes, Daniel Zeman, Manying Zhang, and Hanzhi Zhu. 2018. Universal dependencies 2.3. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Joakim Nivre, Marie-Catherine De Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan T McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation*.

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017. Cross-lingual name tagging and linking for 282 languages. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Sachin Pawar, Pushpak Bhattacharyya, and Girish K. Palshikar. 2017. End-to-end relation extraction using neural networks and Markov logic networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*.

Lonneke Van der Plas, Paola Merlo, and James Henderson. 2011. Scaling up automatic cross-lingual semantic role annotation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

Ondrej Prazák and Miloslav Konopík. 2017. Cross-lingual SRL based upon universal dependencies. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*.

Longhua Qian, Haotian Hui, Ya'nan Hu, Guodong Zhou, and Qiaoming Zhu. 2014. Bilingual active learning for relation classification via pseudo parallel corpora. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Pengda Qin, Weiran Xu, and William Yang Wang. 2018. Dsgan: Generative adversarial training for distant supervision relation extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Chris Quirk and Hoifung Poon. 2017. Distant supervision for relation extraction beyond the sentence boundary. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*.

Sascha Rothe, Sebastian Ebert, and Hinrich Schütze. 2016. Ultradense word embeddings by orthogonal transformation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. 2017. A simple neural network module for relational reasoning. In *Proceedings of Advances in Neural Information Processing Systems*.

Samuel L Smith, David HP Turban, Steven Hamblin, and Nils Y Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. *arXiv preprint arXiv:1702.03859*.

Milan Straka and Jana Straková. 2017. Tokenizing, POS tagging, lemmatizing and parsing UD 2.0 with UDPipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. Ace 2005 multilingual training corpus. *Linguistic Data Consortium, Philadelphia*, 57.

Guanying Wang, Wen Zhang, Ruoxu Wang, Yalin Zhou, Xi Chen, Wei Zhang, Hai Zhu, and Huajun Chen. 2018a. Label-free distant supervision for relation extraction via knowledge graph embedding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Xiaozhi Wang, Xu Han, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2018b. Adversarial multi-lingual neural relation extraction. In *Proceedings of the 27th International Conference on Computational Linguistics*.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Bishan Yang and Tom M. Mitchell. 2016. Joint extraction of events and entities within a document context. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of the 25th International Conference on Computational Linguistics*.

Jingli Zhang, Wenxuan Zhou, Yu Hong, Jianmin Yao, and Min Zhang. 2018a. Using entity relation to improve event detection via attention mechanism. In *Proceedings of NLPCC 2018*.

Tongtao Zhang, Heng Ji, and Avirup Sil. 2019. Joint entity and event extraction with generative adversarial imitation learning. *Data Intelligence Vol 1 (2)*.

Yuan Zhang, David Gaddy, Regina Barzilay, and Tommi S Jaakkola. 2016. Ten pairs to tag-multilingual pos tagging via coarse mapping between embeddings. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018b. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Bowei Zou, Zengzhuang Xu, Yu Hong, and Guodong Zhou. 2018. Adversarial feature adaptation for cross-lingual relation classification. In *Proceedings of the 27th International Conference on Computational Linguistics*.

# Uncover the Ground-Truth Relations in Distant Supervision: A Neural Expectation-Maximization Framework

**Junfan Chen**
BDBC and SKLSDE
Beihang University, China
chenjf@act.buaa.edu.cn

**Richong Zhang***
BDBC and SKLSDE
Beihang University, China
zhangrc@act.buaa.edu.cn

**Yongyi Mao**
School of EECS
University of Ottawa, Canada
ymao@uottawa.ca

**Hongyu Guo**
Digital Technologies Research Center
National Research Council Canada, Canada
hongyu.guo@nrc-cnrc.gc.ca

**Jie Xu**
School of Computing
University of Leeds, United Kingdom
j.xu@leeds.ac.uk

## Abstract

Distant supervision for relation extraction enables one to effectively acquire structured relations out of very large text corpora with less human efforts. Nevertheless, most of the prior-art models for such tasks assume that the given text can be noisy, but their corresponding labels are clean. Such unrealistic assumption is contradictory with the fact that the given labels are often noisy as well, thus leading to significant performance degradation of those models on real-world data. To cope with this challenge, we propose a novel label-denoising framework that combines neural network with probabilistic modelling, which naturally takes into account the noisy labels during learning. We empirically demonstrate that our approach significantly improves the current art in uncovering the ground-truth relation labels.

## 1 Introduction

Relation extraction aims at automatically extracting semantic relationships from a piece of text. Consider the sentence "Larry Page, the chief executive officer of Alphabet, Google's parent company, was born in East Lansing, Michigan.". The knowledge triple (*Larry_Page*, *employed_by*, *Google*) can be extracted. Despite various efforts in building relation extraction models (Zelenko et al., 2002; Zhou et al., 2005; Bunescu and Mooney, 2005; Zeng et al., 2014; dos Santos et al., 2015; Yang et al., 2016; Liu et al., 2015; Xu et al., 2015; Miwa and Bansal, 2016; Gábor et al., 2018), the difficulty of obtaining abundant training data with labelled relations remains a challenge, and thus motivates the development of *Distant Supervision* relation extraction (Mintz et al., 2009; Riedel et al., 2010; Zeng et al., 2015; Lin et al., 2016; Ji et al., 2017; Zeng et al., 2018; Feng et al., 2018; Wang et al., 2018b,a; Hoffmann et al., 2011; Surdeanu et al., 2012; Jiang et al., 2016; Ye et al., 2017; Su et al., 2018; Qu et al., 2018). Distant supervision (DS) relation extract methods collect a large dataset with "distant" supervision signal and learn a relation predictor from such data. In detail, each example in the dataset contains a collection, or *bag*, of sentences all involving the same pair of entities extracted from some corpus (e.g., news reports). Although such a dataset is expected to be very noisy, one hopes that when the dataset is large enough, useful correspondence between the semantics of a sentence and the relation label it implies still reinforces and manifests itself. Despite their capability of learning from large scale data, we show in this paper that these DS relation extraction strategies fail to adequately model the characteristic of the noise in the data. Specifically, most of the works fail to recognize that the labels can be noisy in a bag and directly use bag labels as training targets.

This aforementioned observation has inspired us to study a more realistic setting for DS relation extraction. That is, we treat the bag labels as the noisy observations of the ground-truth labels. To that end, we develop a novel framework that jointly models the semantics representation of the bag, the latent ground truth labels, and the noisy observed labels. The framework, probabilistic in nature, allows any neural network, that encodes the bag semantics, to nest within. We show that the well-known Expectation-Maximization (EM) algorithm can be applied to the end-to-end learning of the models built with this framework. As such, we term the framework *Neural Expectation-Maximization*, or the *nEM*.

Since our approach deviates from the conventional models and regards bag labels not as the ground truth, bags with the real ground-truth la-

*Corresponding author



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 326–336,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

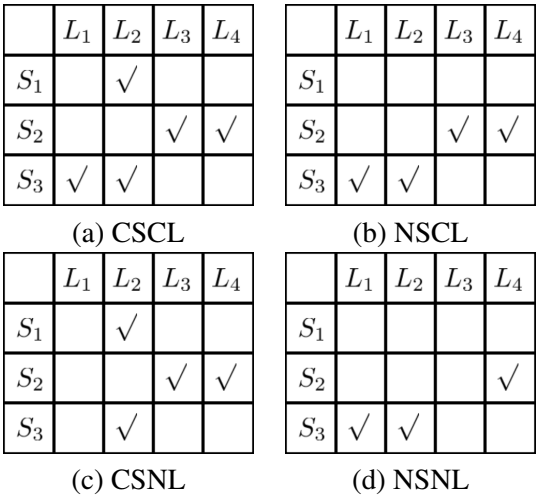


Figure 1: Assumptions on the cleanness of DS dataset. In each matrix, rows correspond to the sentences in a bag, and columns correspond to the labels assigned to the bag. A check mark on $(S_i, Lj)$ indicates that the label $L_j$ is supported by sentence $S_i$.

bels are required for evaluating our model. To that end, we manually re-annotate a fraction of the testing bags in a standard DS dataset with their real ground-truth labels. We then perform extensive experiments and demonstrate that the proposed nEM framework improves the current state-of-the-art models in uncovering the ground truth relations. To the best of our knowledge, this work is the first that combines a neural network model with EM training under the "noisy-sentence noisy-label" assumption. The re-annotated testing dataset [1], containing the ground-truth relation labels, would also benefit the research community.

## 2 Problem Statement

Let set $\mathcal{R}$ contain all relation labels of interest. Specifically, each label $r$ in $\mathcal{R}$ corresponds to a candidate relation in which any considered pair $(e, e')$ of entities may participate. Since $\mathcal{R}$ cannot contain all relations that are implied in a corpus, we include in $\mathcal{R}$ an additional relation label "NA", which refers to any relation that cannot be regarded as the other relations in $\mathcal{R}$.

Any subset of $\mathcal{R}$ will be written as a $\{0, 1\}$-valued vector of length $|\mathcal{R}|$, for example, $z$, where each element of the vector $z$ corresponds to a label $r \in \mathcal{R}$. Specifically, if and only if label $r$ is contained in the subset, its corresponding element $z[r]$ of $z$ equals 1. If two entities $e$ and $e'$ participate in a relation $r$, we say that the triple $(e, r, e')$ is factual. Let $\mathcal{B}$ be a finite set, in which each $b \in \mathcal{B}$ is a pair $(e, e')$ of entities. Each $b = (e, e') \in \mathcal{B}$ serves as the index for a *bag* $x_b$ of sentences.

The objective of DS is to use a large but noisy training set to learn a predictor that predicts the relations involving two arbitrary (possibly unseen) entities; the predictor takes as input a bag of sentences each containing the two entities of interest, and hopefully outputs the set of all relations in which the two entities participate.

## 3 Prior Art and Related Works

Relation extraction is an important task in natural language processing. Many approaches with supervised methods have been proposed to complete this task. These works, such as (Zelenko et al., 2002; Zhou et al., 2005; Bunescu and Mooney, 2005), although achieving good performance, rely on carefully selected features and well labelled dataset. Recently, neural network models, have been used in (Zeng et al., 2014; dos Santos et al., 2015; Yang et al., 2016; Liu et al., 2015; Xu et al., 2015; Miwa and Bansal, 2016) for supervised relation extraction. These models avoid feature engineering and are shown to improve upon previous models. But having a large number of parameters to estimate, these models rely heavily on costly human-labeled data.

Distant supervision was proposed in (Mintz et al., 2009) to automatically generate large dataset through aligning the given knowledge base to text corpus. However, such dataset can be quite noisy. To articulate the nature of noise in DS dataset, a sentence is said to be *noisy* if it supports no relation labels of the bag, and a label of the bag is said to be *noisy* if it is not supported by any sentence in the bag. A sentence or label that is not noisy will be called *clean*. The cleanness of a training example may obey the following four assumptions, for each of which an example is given In Figure 1.

- **Clean-Sentence Clean-Label (CSCL)**: All sentences and all labels are clean (Figure 1(a)).
- **Noisy-Sentence Clean-Label (NSCL)**: Some sentences may be noisy but all labels are clean (Figure 1(b)). Note that CSCL is a special case of NSCL.
- **Clean-Sentence Noisy-Label (CSNL)**: All sentences are clean but some labels may be noisy(Figure 1(c)). Note that CSNL includes CSCL as a special case.

[1] Will be released upon the acceptance of the paper

- **Noisy-Sentence Noisy-Label (NSNL)**: Some sentences may be noisy and some labels may also be noisy (Figure 1(d)).

Obviously, CSCL, NSCL, CSNL are all special cases of NSNL. Thus NSNL is the most general among all these assumptions.

The author of (Mintz et al., 2009) creates a model under the CSCL assumption, which is however pointed out a too strong assumption (Riedel et al., 2010). To alleviate this issue, many studies adopt NSCL assumption. Some of them, including (Riedel et al., 2010; Zeng et al., 2015; Lin et al., 2016; Ji et al., 2017; Zeng et al., 2018; Feng et al., 2018; Wang et al., 2018b,a), formulate the task as a multi-instance learning problem where only one label is allowed for each bag. These works take sentence denoising through selecting the max-scored sentence (Riedel et al., 2010; Zeng et al., 2015, 2018), applying sentence selection with soft attention (Lin et al., 2016; Ji et al., 2017), performing sentence level prediction as well as filtering noisy bags (Feng et al., 2018) and redistributing the noisy sentences into negative bags (Wang et al., 2018b,a). Other studies complete this task with multi-instance multi-label learning (Hoffmann et al., 2011; Surdeanu et al., 2012; Jiang et al., 2016; Ye et al., 2017; Su et al., 2018), which allow overlapping relations in a bag. Despite the demonstrated successes, these models ignore the fact that relation labels can be noisy and "noisy" sentences that indeed point to factual relations may also be ignored. Two recent approaches (Liu et al., 2017; Luo et al., 2017) using the NSNL assumption have also been introduced, but these methods are evaluated based on the assumption that the evaluation labels are clean.

We note that this paper is not the first work that combines neural networks with EM training. Very recently, a model also known as Neural Expectation-Maximization or NEM (Greff et al., 2017) has been presented to learn latent representations for clusters of objects (e.g., images) under a complete unsupervised setting. The NEM model is not directly applicable to our problem setting which deals with noisy supervision signals from categorical relation labels. Nonetheless, given the existence of the acronym NEM, we choose to abbreviate our Neural Expecation-Maximization model as the nEM model.

## 4 The nEM Framework

We first introduce the nEM architecture and its learning strategy, and then present approaches to encode a bag of sentences (i.e., the Bag Encoding Models) needed by the framework.

### 4.1 The nEM Architecture

Let random variable $X$ denote a random bag and random variable $Z$ denote the label set assigned to $X$. Under the NSNL assumption, $Z$, or some labels within, may not be clean for $X$. We introduce another latent random variable $Y$, taking values as a subset of $\mathcal{R}$, indicating the set of ground-truth (namely, clean) labels for $X$. We will write $Y$ again as an $|\mathcal{R}|$-dimensional $\{0,1\}$-valued vector. From here on, we will adopt the convention that a random variable will be written using a capitalized letter, and the value it takes will be written using the corresponding lower-cased letter.

A key modelling assumption in nEM is that random variables $X$, $Y$ and $Z$ form a Markov chain $X \rightarrow Y \rightarrow Z$. Specifically, the dependency of noisy labels $Z$ on the bag $X$ is modelled as

$$p_{Z|X}(z|x) := \sum_{y \in \{0,1\}^{|\mathcal{R}|}} p_{Y|X}(y|x) p_{Z|Y}(z|y) \quad (1)$$

The conditional distribution $p_{Z|Y}$ is modelled as

$$p_{Z|Y}(z|y) \quad := \quad \prod_{r \in \mathcal{R}} p_{Z[r]|Y[r]}\left(z[r]|y[r]\right) \quad (2)$$

That is, for each ground-truth label $r \in \mathcal{R}$, $Z[r]$ depends only on $Y[r]$. Furthermore, we assume that for each $r$, there are two parameters $\phi_r^0$ and $\phi_r^1$ governing the dependency of $Z[r]$ on $Y[r]$ via

$$p_{Z[r]|Y[r]}(z[r]|y[r]) := \begin{cases} \phi_r^0 & y[r]=0, z[r]=1 \\ 1-\phi_r^0 & y[r]=0, z[r]=0 \\ \phi_r^1 & y[r]=1, z[r]=0 \\ 1-\phi_r^1 & y[r]=1, z[r]=1 \end{cases} \quad (3)$$

We will denote by $\{\phi_r^0, \phi_r^1 : r \in \mathcal{R}\}$ collectively by $\phi$. .

On the other hand, we model $p_{Y|X}$ by

$$p_{Y|X}(y|x) := \prod_{r \in \mathcal{R}} p_{Y[r]|X}\left(y[r]|\overline{x}\right) \quad (4)$$

where $\overline{x}$ is the encoding of bag $x$, implemented using any suitable neural network. Postponing explaining the details of bag encoding to a later

section (namely Section 4.3), we here specify the form of $p_{Y[r]|X}\left(y[r]|\overline{x}\right)$ for each $r$:

$$p_{Y[r]|X}\left(1|\overline{x}\right) = \sigma\left(\overline{r}^T\overline{x} + b_r\right) \tag{5}$$

where $\sigma(\cdot)$ is the sigmoid function, $\overline{r}$ is a $|\mathcal{R}|$-dimensional vector and $b_r$ is bias. That is, $p_{Y[r]|X}$ is essentially a logistic regression (binary) classifier based on the encoding $\overline{x}$ of bag $x$. We will denote by $\theta$ the set of all parameters $\{\overline{r} : r \in \mathcal{R}\}$ and the parameters for generating encoding $\overline{x}$.

At this end, we have defined the overall structure of the nEM framework (summarized in Figure 2). Next, we will discuss the learning of it.

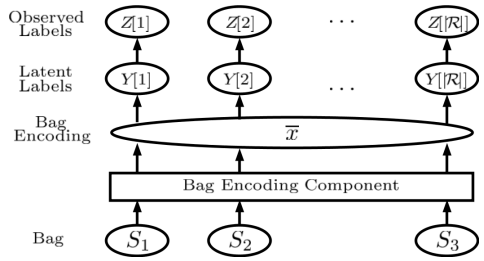


Figure 2: The probabilistic framework of nEM

### 4.2 Learning with the EM Algorithm

Let $\ell(z|x;\phi,\theta)$ be the log-likelihood of observing the label set $z$ given the bag $x$, that is,

$$\ell(z|x;\phi,\theta) := \log p_{Z|X}(z|x). \tag{6}$$

The structure of the framework readily enables a principled learning algorithm based on the EM algorithm (Dempster et al., 1977).

Let $\ell_{\text{total}}(\phi,\theta)$ be the log-likelihood defined as

$$\ell_{\text{total}}(\phi,\theta) \;:=\; \sum_{b\in\mathcal{B}} \ell(z_b|x_b;\phi,\theta). \tag{7}$$

The learning problem can then be formulated as maximizing this objective function over its parameters $(\phi,\theta)$, or solving

$$(\widehat{\phi},\widehat{\theta}) := \arg\max_{\phi,\theta} \ell_{\text{total}}(\phi,\theta). \tag{8}$$

Let $Q$ be an arbitrary distribution over $\{0,1\}^{|\mathcal{R}|}$. Then it is possible to show

$$\begin{aligned}
&\ell(z|x;\phi,\theta)\\
&= \log \sum_{y\in\{0,1\}^{|\mathcal{R}|}} Q(y) \frac{\prod_{r\in\mathcal{R}} p_{Y[r]|X}(y[r]|\overline{x}) p_{Z[r]|Y[r]}(z[r]|y[r])}{Q(y)}\\
&\geqslant \sum_{y\in\{0,1\}^{|\mathcal{R}|}} Q(y) \log \frac{\prod_{r\in\mathcal{R}} p_{Y[r]|X}(y[r]|\overline{x}) p_{Z[r]|Y[r]}(z[r]|y[r])}{Q(y)}\\
&= \mathcal{L}(z|x;\phi,\theta,Q)
\end{aligned} \tag{9}$$

where the lower bound $\mathcal{L}(z|x;\phi,\theta,Q)$, often referred to as the variational lower bound. Now we define $\mathcal{L}_{\text{total}}$ as

$$\mathcal{L}_{\text{total}}(\phi,\theta,\{Q_b : b\in\mathcal{B}\}) := \sum_{b\in\mathcal{B}} \mathcal{L}\left(z_b|x_b;\phi,\theta,Q_b\right) \tag{10}$$

where we have introduced a $Q_b$ for each $b \in \mathcal{B}$. Instead of solving the original optimization problem (8), we can turn to solving a different optimization problem by maximizing $\mathcal{L}_{\text{total}}$

$$(\widehat{\phi},\widehat{\theta},\{\widehat{Q_b}\}) := \arg\max_{\phi,\theta,\{Q_b\}} \mathcal{L}_{\text{total}}(\phi,\theta,\{Q_b\}) \tag{11}$$

The EM algorithm for solving the optimization problem (11) is essentially the coordinate ascent algorithm on objective function $\mathcal{L}_{\text{total}}$, where we iterate over two steps, the E-Step and the M-Step. In the E-Step, we maximizes $\mathcal{L}_{\text{total}}$ over $\{Q_b\}$ for the current setting of $(\phi,\theta)$ and in the M-Step, we maximize $\mathcal{L}_{\text{total}}$ $(\phi,\theta)$ for the current setting of $\{Q_b\}$. We now describe the two steps in detail, where we will use superscript $t$ to denote the iteration number.

**E-step**: In this step, we hold $(\phi,\theta)$ fixed and update $\{Q_b : b\in\mathcal{B}\}$ to maximize the lower bound $\mathcal{L}_{\text{total}}(\phi,\theta,\{Q_b\})$. This boils down to update each factor $Q_{b,r}$ of $Q_b$ according to:

$$\begin{aligned}
Q_{b,r}^{t+1}(y[r]) &:= p_{Y[r]|X,Z}(y_b[r]|x_b, z_b[r]; \theta^t, \phi_r^t)\\
&= \frac{p_{Z|Y}(z_b[r]|y_b[r];\phi_r^t) p_{Y|X}(y_b[r]|x_b;\theta^t)}{p_{Z|X}(z_b[r]|x_b;\theta^t,\phi_r^t)}\\
&= \frac{p_{Z|Y}(z_b[r]|y_b[r];\phi_r^t) p_{Y|X}(y_b[r]|x_b;\theta^t)}{\sum_{y[r]\in\{0,1\}} p_{Z|Y}(z_b[r]|y[r];\phi_r^t) p_{Y|X}(y[r]|x_b;\theta^t)}
\end{aligned} \tag{12}$$

**M-step**: In this step, we hold $\{Q_b\}$ fixed and update $(\phi,\theta)$, to maximize the lower bound $\mathcal{L}_{\text{total}}(\phi,\theta,\{Q_b\})$. Let

$$M = \prod_{r\in\mathcal{R}} p_{Y[r]|X}(y_b[r]|\overline{x}_b) p_{Z[r]|Y[r]}(z_b[r]|y_b[r]), \tag{13}$$

then $(\phi,\theta)$ is updated according to:

$$\begin{aligned}
&(\theta^{t+1},\phi^{t+1}) := \operatorname*{argmax}_{\theta,\phi} \mathcal{L}_{\text{total}}(\theta,\phi,\{Q_b^{t+1}\})\\
&= \operatorname*{argmax}_{\theta,\phi} \sum_{b\in\mathcal{B}} \sum_{y_b\in\{0,1\}^{|\mathcal{R}|}} Q_b^{t+1}(y) \log \frac{M}{Q_b^{t+1}(y)}\\
&= \operatorname*{argmax}_{\theta,\phi} \sum_{b\in\mathcal{B}} \sum_{y_b\in\{0,1\}^{|\mathcal{R}|}} Q_b^{t+1}(y) \log M - \sum_{b\in\mathcal{B}} \sum_{y_b\in\{0,1\}^{|\mathcal{R}|}} Q_b^{t}(y) \log Q_b^{t+1}(y)\\
&= \operatorname*{argmax}_{\theta,\phi} \sum_{b\in\mathcal{B}} \sum_{r\in\mathcal{R}} \sum_{y_b[r]\in\{0,1\}} Q_{b,r}^{t+1}(y[r]) \big(\log p_{Z|Y}(z_b[r]|y_b[r];\phi_r)\\
&\quad + \log p_{Y|X}(y_b[r]|\overline{x}_b;\theta)\big)
\end{aligned} \tag{14}$$

Overall the EM algorithm starts with initializing $(\phi, \theta, \{Q_b\})$, and then iterates over the two steps until convergence or after some prescribed number of iterations is reached. There are however several caveats on which caution is needed. First, the optimization problem in the M-Step cannot be solved in closed form. As such we take a stochastic gradient descent (SGD) approach[2].

In each M-Step, we perform $\Delta$ updates, where $\Delta$ is a hyper-parameter. Such an approach is sometimes referred to as "Generalized EM" (Wu, 1983; Jojic and Frey, 2001; Greff et al., 2017). Note that since the parameters $\theta$ are parameters for the neural network performing bag encoding, the objective function $\mathcal{L}_{\text{total}}$ is highly non-convex with respect to $\theta$. This makes it desirable to choose an appropriate $\Delta$. Too small $\Delta$ results in little change in $\theta$ and hence provides insufficient signal to update $\{Q_b\}$; too large $\Delta$, particularly at the early iterations when $\{Q_b\}$ has not been sufficiently optimized, tends to make the optimization stuck in undesired local optimum. In practice, one can try several values of $\Delta$ by inspecting their achieved value of $\mathcal{L}_{\text{total}}$ and select the $\Delta$ giving rise to the highest $\mathcal{L}_{\text{total}}$. Note that such a tuning of $\Delta$ requires no access of the testing data. The second issue is that the EM algorithm is known to be sensitive to initialization. In our implementation, in order to provide a good initial parameter setting, we set each $Q_b^0$ to $z_b$. Despite the fact that $z$ contains noise, this is a much better approximation of the posterior of true labels than any random initialization.

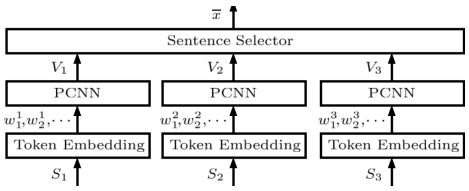


Figure 3: The structure of Bag Encoding component.

The nEM framework needs the encoding of a bag of $x$ as discussed in Equation 4. Any suitable neural network can be deployed to achieve this goal. Next, we present the widely used methods for DS relation extraction strategies: the Bag Encoding Models.

[2]In fact, the approach should be called "stochastic gradient ascent", since we are maximizing the objective function not minimizing it.

### 4.3 Bag Encoding Models

As illustrated in Figure 3, the Bag Encoding Models include three components: Word-Position Embedding, Sentence Encoding, and Sentence Selectors.

#### 4.3.1 Word-Position Embedding

For the $j^{\text{th}}$ word in a sentence, Word-Position Embedding generates a vector representation $\mathbf{w}_j$ as concatenated three components $\left[\mathbf{w}_j^w, \mathbf{w}_j^{p1}, \mathbf{w}_j^{p2}\right]$. Specifically, $\mathbf{w}_j^w$ is the word embedding of the word and $\mathbf{w}_j^{p1}$ and $\mathbf{w}_j^{p2}$ are two position embeddings. Here $\mathbf{w}_j^{p1}$ (resp. $\mathbf{w}_j^{p2}$) are the embedding of the relative location of the word with respect to the first (resp. second) entity in the sentence. The dimensions of word and position embeddings are denoted by $d_w$ and $d_p$ respectively.

#### 4.3.2 Sentence Encoding

Sentence encoding uses Piecewise Convolutional Neural Networks (PCNN) (Zeng et al., 2015; Lin et al., 2016; Ye et al., 2017; Ji et al., 2017), which consists of convolution followed by Piecewise Max-pooling. The convolution operation uses a list of matrix kernels such as $\{\mathbf{K}_1, \mathbf{K}_2, \cdots, \mathbf{K}_{l_{ker}}\}$ to extract *n-gram* features through sliding windows of length $l_{win}$. Here $\mathbf{K}_i \in \mathbb{R}^{l_{win} \times d_e}$ and $l_{ker}$ is the number of kernels. Let $\mathbf{w}_{j-l_{win}+1:j} \in \mathbb{R}^{l_{win} \times d_e}$ be the concatenated vector of token embeddings in the $j^{\text{th}}$ window. The output of convolution operation is a matrix $\mathbf{U} \in \mathbb{R}^{l_{ker} \times (m+l_{win}-1)}$ where each element is computed by

$$\mathbf{U}_{ij} = \mathbf{K}_i \odot \mathbf{w}_{j-l_{win}+1:j} + b_i \qquad (15)$$

where $\odot$ denotes inner product. In Piecewise Max-Pooling, $\mathbf{U}_i$ is segmented to three parts $\left\{\mathbf{U}_i^1, \mathbf{U}_i^2, \mathbf{U}_i^3\right\}$ depending on whether an element in $\mathbf{U}_i$ is on the left or right of the two entities, or between the two entities. Then max-pooling is applied to each segment, giving rise to

$$\mathbf{g}^{ip} = \max(\mathbf{U}_i^p), 1 \leqslant i \leqslant l_{ker}, 1 \leqslant p \leqslant 3 \qquad (16)$$

Let $\mathbf{g} = \left[\mathbf{g}^1, \mathbf{g}^2, \mathbf{g}^3\right]$. Then sentence encoding outputs

$$\mathbf{v} = \text{ReLU}(\mathbf{g}) \circ \mathbf{h}, \qquad (17)$$

where $\circ$ is element-wise multiplication and $\mathbf{h}$ is a vector of Bernoulli random variables, representing dropouts.

#### 4.3.3 Sentence Selectors

Let $n$ be the number of sentences in a bag. We denote a matrix $\mathbf{V} \in \mathbb{R}^{n\times(l_{ker}\times 3)}$consisting of each sentence vector $\mathbf{v}_T$. $\mathbf{V}_{k:}$ and $\mathbf{V}_{:j}$ are used to index the $k^{\text{th}}$ row vector and $j^{\text{th}}$ column vector of $\mathbf{V}$ respectively. Three kinds of sentence-selectors are used to construct the bag encoding. Mean-selector (Lin et al., 2016; Ye et al., 2017): The bag encoding is computed as

$$\overline{x} = \frac{1}{n}\sum_{k=1}^{n}\mathbf{V}_{k:} \tag{18}$$

Max-selector (Jiang et al., 2016): The $j^{\text{th}}$ element of bag encoding $\overline{x}$ is computed as

$$\overline{x}_j = \max(\mathbf{V}_{:j}) \tag{19}$$

Attention-selector Attention mechanism is extensively used for sentence selection in relation extraction by weighted summing of the sentence vectors, such as in (Lin et al., 2016; Ye et al., 2017; Su et al., 2018). However, all these works assume that the labels are *correct* and only use the golden label embeddings to select sentences at training stage. We instead selecting sentences using all label embeddings $\overline{r}$ and construct a bag encoding for each label $r \in \mathcal{R}$. The output is then a list of vectors $\{\overline{x}^r\}$, in which the $r^{\text{th}}$ vector is calculated through attention mechanism as

$$e_l = \mathbf{V}_{l:}^T\mathbf{A}\overline{r},\ \alpha_k = \frac{\exp(e_k)}{\sum_{l=1}^{n}\exp(e_l)},\ \overline{x}^r = \sum_{k=1}^{n}\alpha_k\mathbf{V}_{k:} \tag{20}$$

where $\mathbf{A} \in \mathbb{R}^{d_r\times d_r}$ is a diagonal matrix and $d_r$ is the dimension of relation embedding.

## 5 Experimental Study

We first conduct experiments on the widely used benchmark data set Riedel (Riedel et al., 2010), and then on the TARCED (Zhang et al., 2017) data set. The latter allows us to control the noise level in the labels to observe the behavior and working mechanism of our proposed method. The code for our model is found on the Github page [3].

### 5.1 Evaluation on the Riedel Dataset

The Riedel dataset[2] is a widely used DS dataset for relation extraction. It was developed in (Riedel et al., 2010) through aligning entity pairs from Freebase with the New York Times (NYT) corpus. There are 53 relations in the Riedel dataset, including "*NA*". The bags collected from the 2005-2006 corpus are used as the training set, and the bags from the 2007 corpus are used as the test set. The training data contains $281,270$ entity pairs and $522,611$ sentences; the testing data contains $96,678$ entity pairs and $172,448$ sentences.

#### 5.1.1 Ground-Truth Annotation

The Riedel dataset contains no ground-truth labels. The held-out evaluation (Mintz et al., 2009) is highly unreliable in measuring a model's performance against the ground truth. Since this study is concerned with discovering the ground-truth labels, the conventional held-out evaluation is no longer appropriate. For this reason, we annotate a subset of the original test data for evaluation purpose. Specifically, we annotate a bag $x_b$ by its all correct labels, and if no such labels exist, we label the bag NA.

In total 3762 bags are annotated, which include all bags originally labelled as non-NA ("Part 1") and 2000 bags ("Part 2") selected from the bags originally labelled as NA but have relatively low score for NA label under a pre-trained PCNN+ATT (Lin et al., 2016) model. The rationale here is that we are interested in inspecting the model's performance in detecting the labels of known relations rather than NA relation.

Table 1: The statistics of ground-truth annotation. *origin* denotes the total number of originally assigned labels of these bags. *correct* and *wrong* denote the total number of correctly assigned and wrongly assigned labels. *added* denote the number of missing labels we added into these bags.

| | bags | origin | correct | wrong | added |
|---|---|---|---|---|---|
| Part 1 | 1762 | 1953 | 1245 | 708 | 804 |
| Part 2 | 2000 | 2000 | 938 | 1062 | 1211 |

The statistics of annotation is shown in Table 1. Through the annotation, we notice that, about 36% of the original labels in original non-NA bags and 53% of labels in original NA bags are wrongly assigned. Similar statistics has been reported in previous works (Riedel et al., 2010; Feng et al., 2018).

#### 5.1.2 Evaluation Metrics and Baselines

For the Riedel dataset, we train the models on the noisy training set and test the models on the manually labeled test set. The precision-recall (PR)

[3] https://github.com/AlbertChen1991/nEM
[2] http://iesl.cs.umass.edu/riedel/ecml/

curve is reported to compare performance of models. Three baselines are considered:

- PCNN+MEAN (Lin et al., 2016; Ye et al., 2017): A model using PCNN to encode sentences and a mean-selector to generate bag encoding.
- PCNN+MAX (Jiang et al., 2016): A model using PCNN to encode sentences and a max-selector to generate bag encoding.
- PCNN+ATT (Lin et al., 2016; Ye et al., 2017; Su et al., 2018): A model using PCNN to encode sentences and an attention-selector to generate bag encoding.

We compare the three baselines with their nEM versions (namely using them as the Bag Encoding component in nEM), which are denoted with a "+nEM" suffix.

#### 5.1.3 Implementation Detail

Following previous work, we tune our models using three-fold validation on the training set. As the best configurations, we set $d_w = 50$, $d_p = 5$ and $d_r = 230$. For PCNN, we set $l_{win} = 3$, $l_{ker} = 230$ and set the probability of dropout to 0.5. We use Adadelta (Zeiler, 2012) with default setting ($\rho = 0.95, \varepsilon = 1e^{-6}$) to optimize the models and the initial learning rate is set as 1. The batch size is fixed to 160 and the max length of a sentence is set as 120.

For the noise model $p_{Z|Y}$, we set $\phi^0_{NA} = 0.3, \phi^1_{NA} = 0$ for the NA label and $\phi^0_r = 0, \phi^1_r = 0.9$ for other labels $r \neq NA$. In addition, the number $\Delta$ of SGD updates in M-step is set to 2000.

#### 5.1.4 Predictive performance

The evaluation results on manually labeled test set are shown in Figure 4. From which, we observe that the PR-curves of the nEM models are above their corresponding baselines by significant margins, especially in the low-recall regime. This observation demonstrates the effectiveness of the nEM model on improving the extracting performance upon the baseline models. We also observe that models with attention-selector achieve better performance than models with mean-selector and max-selector, demonstrating the superiority of the attention mechanism on sentence selection.

We then analyze the predicting probability of PCNN+ATT and PCNN+ATT+nEM on the ground-truth labels in the test set. We divide the predicting probability values into 5 bins and

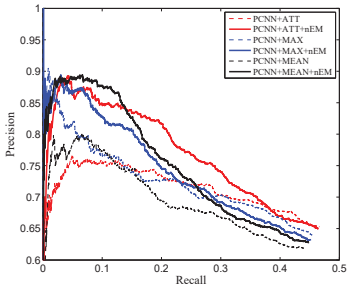


Figure 4: PR curves on DS test set.

count the number of label within each bin. The result is shown in Figure 5(a). We observe that the count for PCNN+ATT in bins $0.0 - 0.2$, $0.2 - 0.4$, $0.4 - 0.6$ and $0.6 - 0.8$ are all greater than PCNN+ATT+nEM. But in bin $0.8 - 1.0$, the count for PCNN+ATT+nEM is about 55% larger than PCNN+ATT. This observation indicates that nEM can promote the overall predicting scores of ground-truth labels.

Figure 5(b) compares PCNN+ATT and PCNN+ATT+nEM in their predictive probabilities on the frequent relations. The result shows that PCNN+ATT+nEM achieves higher average predicting probability on all these relations, except for the NA relation, on which PCNN+ATT+nEM nearly levels with PCNN+ATT. This phenomenon demonstrates that nEM tends to be more confident in predicting the correct labels. In this case, raising the predictive probability on the correct label does increase the models ability to make the correct decision, thereby improving performance.

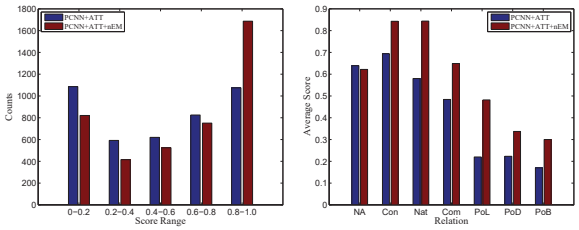


(a) Score distribution (b) Scores on main relations

Figure 5: Predictive probabilities on the test set. CON, Nat, Com, PoL, PoD and PoB represent 'contains', 'nationality', 'company', 'place lived', 'place of death' and 'place of birth' relations respectively.

### 5.2 Evaluation on the TACRED Dataset

TACRED is a large supervised relation extraction dataset collected in (Zhang et al., 2017). The text corpus is generated from the TAC KBP evaluation of years 2009 – 2015. Corpus from years

$2009 - 2012$, year 2013 and year 2014 are used as training, validation and testing sets respectively. In all annotated sentences, $68,124$ are used for training, $22,631$ for validation and $15,509$ for testing. There are in total $42,394$ *distinct* entity pairs, where $33,079$ of these entity pairs appear in exactly one sentence. Since these $33,079$ entity pairs dominate the training set, we simply treat each sentence as a bag for training and testing. Another advantage of constructing single-sentence bags is that it allows us to pinpoint the correspondence between the correct label and its supporting sentence. The number of relations in TACRED dataset is 42, including a special relation "*no relation*", which is treated as NA in this study.

#### 5.2.1 Constructing Semi-synthetic Dataset

To obtain insight into the working of nEM, we create a simulated DS dataset by inserting noisy labels into the training set of a supervised dataset, TACRED. Since the training set of TACRED was manually annotated, the labels therein may be regarded as the ground-truth labels. Training using this semi-synthetic dataset allows us to easily observe models' behaviour with respect to the noisy labels and the true labels.

We inject artificial noise into the TACRED training data through the following precedure. For each bag $x_b$ in the training set, we generate a noisy label vector $\tilde{z}_b$ from the observed ground-truth label vector $y_b$. Specifically, $\tilde{z}_b$ is generated by flipping each element $y_b[r]$ from 0 to 1 or 1 to 0 with a probability $p_f$. This precedure simulates a DS dataset through introducing wrong labels into training bags, thus corrupts the training dataset.

#### 5.2.2 Experimental Settings

Following (Zhang et al., 2017), the common relation classification metrics Precision, Recall and F1 are used for evaluation. The PCNN model is used to generate the bag encoding since sentence selection is not needed in this setting. The same hyperparameter settings as in Section 5.1.3 are used in this experiment. For the noise model $p_{Z[r]|Y[r]}$ of PCNN +nEM, we set $\phi_r^0 = 0.1, \phi_r^1 = 0.1$ for each label $r \in \mathcal{R}$. The number $\Delta$ of SGD updates in each M-step is set to 1600.

#### 5.2.3 Test Results

From Table 2, we see that PCNN+nEM achieves better recall and F1 score than the PCNN model under various noise levels. Additionally, the recall and F1 margins between PCNN and PCNN+nEM increase with noise levels. This suggests that nEM keeps better performance than the corresponding baseline model under various level of training noise. We also observe that the precision of nEM is consistently lower than that of PCNN when noise is injected to TACRED. This is a necessary trade-off present in nEM. The training of nEM regards the training labels with less confidence based on noisy-label assumption. This effectively lowers the probability of seen training labels and considers the unseen labels also having some probability to occur. When trained this way, nEM, at the prediction time, tends to predict more labels than its baseline (PCNN) does. Note that in TACRED, each instance contains only a single ground truth label. Thus the tendency of nEM to predict more labels leads to the reduced precision. However, despite this tendency, nEM, comparing with PCNN, has a stronger capability in detecting the correct label and gives the better recall of nEM. The gain in recall dominates the loss in precision.

Table 2: Test performance on the TACRED dataset.

| $p_f$ | PCNN | | | PCNN+nEM | | |
|---|---|---|---|---|---|---|
| | **P** | **R** | **F1** | **P** | **R** | **F1** |
| 0.02 | 65.2 | 30.0 | 41.1 | 61.8 | 34.8 | 44.5 |
| 0.04 | 63.5 | 28.9 | 39.7 | 60.2 | 33.8 | 43.3 |
| 0.06 | 70.5 | 25.2 | 37.1 | 59.9 | 31.1 | 41.0 |
| 0.08 | 63.5 | 23.6 | 34.4 | 58.8 | 29.8 | 39.6 |
| 0.10 | 66.1 | 22.7 | 33.9 | 56.5 | 28.9 | 38.3 |

#### 5.2.4 Training Label Probabilities

The predicting probabilities for the noise labels and the original true labels are also evaluated under the trained models. The results are shown in Figure 6 (left) which reveals that with increasing noise, the average probability for noisy label sets of PCNN and PCNN+nEM both increase and average scores for original label sets of PCNN and PCNN+nEM both decrease. The performance degradation of PCNN and PCNN+nEM under noise appears different. The average probability for noisy label sets rises with a higher slope in PCNN than in PCNN+nEM. Additionally, the average probability for the original label sets of PCNN+nEM is higher than or equal to PCNN at all noise levels. These observations confirms that the denoising capability of nEM is learned from effectively denoising the training set.

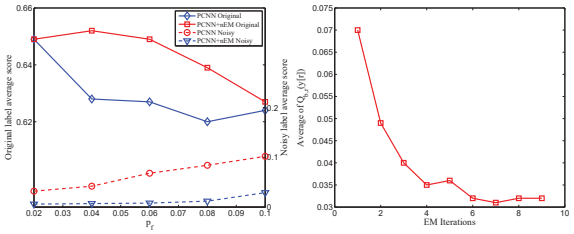


Figure 6: Average probabilities for noisy and original labels (left) and average $Q_{b,r}(y[r])$ over EM iterations (right)

#### 5.2.5 Effectiveness of EM Iterations

For each training bag $x_b$ and each artificial noisy label $r$, we track the probability $Q_{b,r}(y[r]) = 1$ over EM iterations. This probability, measuring the likelihood of the noise label $r$ being correct, is then averaged over $r$ and over all bags $x_b$. It can be seen in Figure 6 (right) that the average value of $Q_{b,r}(y[r])$ decreases as the training progresses, leading the model to gradually ignore noisy labels. This demonstrates the effectiveness of EM iterations and validates the proposed EM-based framework.

## 6 Concluding Remarks

We proposed a nEM framework to deal with the noisy-label problem in distance supervision relation extraction. We empirically demonstrated the effectiveness of the nEM framework, and provided insights on its working and behaviours through data with controllable noise levels. Our framework is a combination of latent variable models in probabilistic modelling with contemporary deep neural networks. Consequently, it naturally supports a training algorithm which elegantly nests the SGD training of any appropriate neural network inside an EM algorithm. We hope that our approach and the annotated clean testing data would inspire further research along this direction.

## Acknowledgment

This work is supported partly by China 973 program (No. 2015CB358700), by the National Natural Science Foundation of China (No. 61772059, 61421003), by the Beijing Advanced Innovation Center for Big Data and Brain Computing (BDBC), by State Key Laboratory of Software Development Environment (No. SKLSDE-2018ZX-17) and by the Fundamental Research Funds for the Central Universities.

## References

Razvan C. Bunescu and Raymond J. Mooney. 2005. Subsequence kernels for relation extraction. In *Advances in Neural Information Processing Systems 18 [Neural Information Processing Systems, NIPS 2005, December 5-8, 2005, Vancouver, British Columbia, Canada]*, pages 171–178.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38.

Jun Feng, Minlie Huang, Li Zhao, Yang Yang, and Xiaoyan Zhu. 2018. Reinforcement learning for relation classification from noisy data. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5779–5786.

Kata Gábor, Davide Buscaldi, Anne-Kathrin Schumann, Behrang QasemiZadeh, Haïfa Zargayouna, and Thierry Charnois. 2018. Semeval-2018 task 7: Semantic relation extraction and classification in scientific papers. In *SemEval@NAACL-HLT*, pages 679–688. Association for Computational Linguistics.

Klaus Greff, Sjoerd van Steenkiste, and Jürgen Schmidhuber. 2017. Neural expectation maximization. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6694–6704.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke S. Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*, pages 541–550.

Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. 2017. Distant supervision for relation extraction with sentence-level attention and entity descriptions. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3060–3066.

Xiaotian Jiang, Quan Wang, Peng Li, and Bin Wang. 2016. Relation extraction with multi-instance multi-label convolutional neural networks. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 1471–1480.

Nebojsa Jojic and Brendan J. Frey. 2001. Learning flexible sprites in video layers. In *2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001), with CD-ROM, 8-14 December 2001, Kauai, HI, USA*, pages 199–206.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Tianyu Liu, Kexiang Wang, Baobao Chang, and Zhifang Sui. 2017. A soft-label method for noise-tolerant distantly supervised relation extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 1790–1795.

Yang Liu, Furu Wei, Sujian Li, Heng Ji, Ming Zhou, and Houfeng Wang. 2015. A dependency-based neural network for relation classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*, pages 285–290.

Bingfeng Luo, Yansong Feng, Zheng Wang, Zhanxing Zhu, Songfang Huang, Rui Yan, and Dongyan Zhao. 2017. Learning with noise: Enhance distantly supervised relation extraction with dynamic transition matrix. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 430–439.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACL 2009, Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP, 2-7 August 2009, Singapore*, pages 1003–1011.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Meng Qu, Xiang Ren, Yu Zhang, and Jiawei Han. 2018. Weakly-supervised relation extraction by pattern-enhanced embedding learning. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, pages 1257–1266.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD 2010, Barcelona, Spain, September 20-24, 2010, Proceedings, Part III*, pages 148–163.

Cícero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 626–634.

Sen Su, Ningning Jia, Xiang Cheng, Shuguang Zhu, and Ruiping Li. 2018. Exploring encoder-decoder model for distant supervised relation extraction. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden.*, pages 4389–4395.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2012, July 12-14, 2012, Jeju Island, Korea*, pages 455–465.

William Yang Wang, Weiran Xu, and Pengda Qin. 2018a. DSGAN: generative adversarial training for distant supervision relation extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 496–505.

William Yang Wang, Weiran Xu, and Pengda Qin. 2018b. Robust distant supervision relation extraction via deep reinforcement learning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 2137–2147.

CF Jeff Wu. 1983. On the convergence properties of the em algorithm. *The Annals of statistics*, pages 95–103.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1785–1794.

Yunlun Yang, Yunhai Tong, Shulei Ma, and Zhi-Hong Deng. 2016. A position encoding convolutional

neural network based on dependency tree for relation classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 65–74.

Hai Ye, Wenhan Chao, Zhunchen Luo, and Zhoujun Li. 2017. Jointly extracting relations with class ties via effective deep ranking. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1810–1820.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2002. Kernel methods for relation extraction. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing, EMNLP 2002, Philadelphia, PA, USA, July 6-7, 2002*.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1753–1762.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, August 23-29, 2014, Dublin, Ireland*, pages 2335–2344.

Xiangrong Zeng, Shizhu He, Kang Liu, and Jun Zhao. 2018. Large scaled relation extraction with reinforcement learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5658–5665.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. 2017. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 35–45.

Guodong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring various knowledge in relation extraction. In *ACL 2005, 43rd Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, 25-30 June 2005, University of Michigan, USA*, pages 427–434.

# Doc2EDAG: An End-to-End Document-level Framework for Chinese Financial Event Extraction

**Shun Zheng**[†*], **Wei Cao**[‡], **Wei Xu**[†], **Jiang Bian**[‡]
[†]Institute of Interdisciplinary Information Sciences, Tsinghua University
[‡]Microsoft Research
zhengs14@mails.tsinghua.edu.cn;
{Wei.Cao, Jiang.Bian}@microsoft.com;
weixu@mail.tsinghua.edu.cn

## Abstract

Most existing event extraction (EE) methods merely extract event arguments within the sentence scope. However, such sentence-level EE methods struggle to handle soaring amounts of documents from emerging applications, such as finance, legislation, health, etc., where event arguments always scatter across different sentences, and even multiple such event mentions frequently co-exist in the same document. To address these challenges, we propose a novel end-to-end model, Doc2EDAG, which can generate an entity-based directed acyclic graph to fulfill the document-level EE (DEE) effectively. Moreover, we reformalize a DEE task with the no-trigger-words design to ease document-level event labeling. To demonstrate the effectiveness of Doc2EDAG, we build a large-scale real-world dataset consisting of Chinese financial announcements with the challenges mentioned above. Extensive experiments with comprehensive analyses illustrate the superiority of Doc2EDAG over state-of-the-art methods. Data and codes can be found at https://github.com/dolphin-zs/Doc2EDAG.

## 1 Introduction

Event extraction (EE), traditionally modeled as detecting trigger words and extracting corresponding arguments from plain text, plays a vital role in natural language processing since it can produce valuable structured information to facilitate a variety of tasks, such as knowledge base construction, question answering, language understanding, etc.

In recent years, with the rising trend of digitalization within various domains, such as finance, legislation, health, etc., EE has become an increasingly important accelerator to the development of business in those domains. Take the financial domain as an example, continuous economic growth has witnessed exploding volumes of digital financial documents, such as financial announcements in a specific stock market as Figure 1 shows, specified as *Chinese financial announcements* (ChFinAnn). While forming up a gold mine, such large amounts of announcements call EE for assisting people in extracting valuable structured information to sense emerging risks and find profitable opportunities timely.

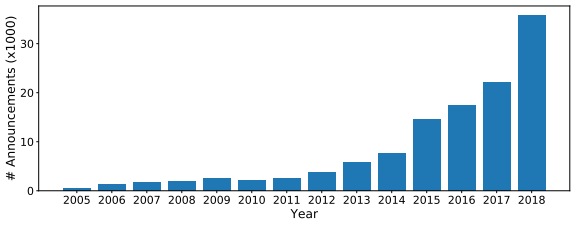


Figure 1: The rapid growth of event-related announcements considered in this paper.

Given the necessity of applying EE on the financial domain, the specific characteristics of financial documents as well as those within many other business fields, however, raise two critical challenges to EE, particularly *arguments-scattering* and *multi-event*. Specifically, the first challenge indicates that arguments of one event record may scatter across multiple sentences of the document, while the other one reflects that a document is likely to contain multiple such event records. To intuitively illustrate these challenges, we show a typical ChFinAnn document with two *Equity Pledge* event records in Figure 2. For the first event, the entity[1] "[SHARE1]" is the correct *Pledged Shares* at the sentence level (*ID 5*). However, due to the capital stock increment (*ID 7*),

[*] This work was done during the internship of Shun Zheng at Microsoft Research Asia, Beijing, China.

[1] In this paper, we use "entity" as a general notion that includes named entities, numbers, percentages, etc., for brevity.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 337–346,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

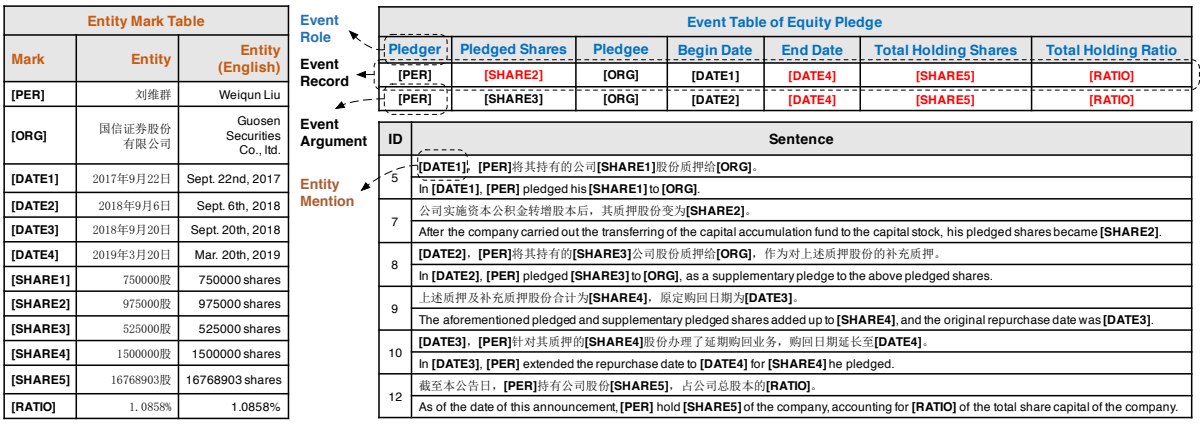


Figure 2: A document example with two *Equity Pledge* event records whose arguments scatter across multiple sentences, where we use *ID* to denote the sentence index, substitute entity mentions with corresponding marks, and color event arguments outside the scope of key-event sentences as red.

the correct *Pledged Shares* at the document level should be "[SHARE2]". Similarly, "[DATE3]" is the correct *End Date* at the sentence level (*ID 9*) but incorrect at the document level (*ID 10*). Moreover, some summative arguments, such as "[SHARE5]" and "[RATIO]", are often stated at the end of the document.

Although a great number of efforts (Ahn, 2006; Ji and Grishman, 2008; Liao and Grishman, 2010; Hong et al., 2011; Riedel and McCallum, 2011; Li et al., 2013, 2014; Chen et al., 2015; Yang and Mitchell, 2016; Nguyen et al., 2016; Liu et al., 2017; Sha et al., 2018; Zhang and Ji, 2018; Nguyen and Nguyen, 2019; Wang et al., 2019) have been put on EE, most of them are based on ACE 2005[2], an expert-annotated benchmark, which only tagged event arguments within the sentence scope. We refer to such task as the **sentence-level EE** (SEE), which obviously overlooks the *arguments-scattering* challenge. In contrast, EE on financial documents, such as ChFinAn, requires **document-level EE** (DEE) when facing *arguments-scattering*, and this challenge gets much harder when coupled with *multi-event*.

The most recent work, DCFEE (Yang et al., 2018), attempted to explore DEE on ChFinAnn, by employing *distant supervision* (DS) (Mintz et al., 2009) to generate EE data and performing a two-stage extraction: 1) a sequence tagging model for SEE, and 2) a key-event-sentence detection model to detect the key-event sentence, coupled with a heuristic strategy that padded missing arguments from surrounding sentences, for DEE. However, the sequence tagging model for SEE cannot handle multi-event sentences elegantly, and even worse, the context-agnostic arguments-completion strategy fails to address the arguments-scattering challenge effectively.

In this paper, we propose a novel end-to-end model, Doc2EDAG, to address the unique challenges of DEE. The key idea of Doc2EDAG is to transform the event table into an *entity-based directed acyclic graph* (EDAG). The EDAG format can transform the hard table-filling task into several sequential path-expanding sub-tasks that are more tractable. To support the EDAG generation efficiently, Doc2EDAG encodes entities with document-level contexts and designs a memory mechanism for path expanding. Moreover, to ease the DS-based document-level event labeling, we propose a novel DEE formalization that removes the trigger-words labeling and regards DEE as directly filling event tables based on a document. This no-trigger-words design does not rely on any predefined trigger-words set or heuristic to filter multiple trigger candidates, and still perfectly matches the ultimate goal of DEE, mapping a document to underlying event tables.

To evaluate the effectiveness of our proposed Doc2EDAG, we conduct experiments on a real-world dataset, consisting of large scales of financial announcements. In contrast to the dataset used by DCFEE where 97%[3] documents just contained one event record, our data collection is ten times larger where about 30% documents include multiple event records. Extensive experiments demonstrate that Doc2EDAG can significantly outper-

[2] https://www.ldc.upenn.edu/collaborations/past-projects/ace

[3] Estimated by their Table 1 as $\frac{2*\text{NO.ANN}-\text{NO.POS}}{\text{NO.ANN}}$.

form state-of-the-art methods when facing DEE-specific challenges.

In summary, our contributions include:

- We propose a novel model, Doc2EDAG, which can directly generate event tables based on a document, to address unique challenges of DEE effectively.
- We reformalize a DEE task without trigger words to ease the DS-based document-level event labeling.
- We build a large-scale real-world dataset for DEE with the unique challenges of *arguments-scattering* and *multi-event*, the extensive experiments on which demonstrate the superiority of Doc2EDAG.

Note that though we focus on ChFinAnn data in this work, we tackle those DEE-specific challenges without any domain-specific assumption. Therefore, our general labeling and modeling strategies can directly benefit many other business domains with similar challenges, such as criminal facts and judgments extraction from legal documents, disease symptoms and doctor instructions identification from medical reports, etc.

## 2 Related Work

Recent development on information extraction has been advancing in building the joint model that can extract entities and identify structures (relations or events) among them simultaneously. For instance, (Ren et al., 2017; Zheng et al., 2017; Zeng et al., 2018a; Wang et al., 2018) focused on jointly extracting entities and inter-entity relations. In the meantime, the same to the focus of this paper, a few studies aimed at designing joint models for the entity and event extraction, such as handcrafted-feature-based (Li et al., 2014; Yang and Mitchell, 2016; Judea and Strube, 2016) and neural-network-based (Zhang and Ji, 2018; Nguyen and Nguyen, 2019) models. Nevertheless, these models did not present how to handle argument candidates beyond the sentence scope. (Yang and Mitchell, 2016) claimed to handle event-argument relations across sentences with the prerequisite of well-defined features, which, unfortunately, is nontrivial.

In addition to the modeling challenge, another big obstacle for democratizing EE is the lack of training data due to the enormous cost to obtain expert annotations. To address this problem, some researches attempted to adapt distant supervision (DS) to the EE setting, since DS has shown promising results by employing knowledge bases to automatically generate training data for relation extraction (Mintz et al., 2009). However, the vanilla EE required the trigger words that were absent on factual knowledge bases. Therefore, (Chen et al., 2017; Yang et al., 2018) employed either linguistic resources or predefined dictionaries for trigger-words labeling. On the other hand, another recent work (Zeng et al., 2018b) showed that directly labeling event arguments without trigger words was also feasible. However, they only considered the SEE setting and their methods cannot be directly extended to the DEE setting, which is the main focus of this work.

Traditionally, when applying DS to relation extraction, researchers put huge efforts into alleviating labeling noises (Riedel et al., 2010; Lin et al., 2016; Feng et al., 2018; Zheng et al., 2019). In contrast, this work shows that combining DS with some simple constraints can obtain pretty good labeling quality for DEE, where the reasons are two folds: 1) both the knowledge base and text documents are from the same domain; 2) an event record usually contains multiple arguments, while a common relational fact only covers two entities.

## 3 Preliminaries

We first clarify several key notions: 1) **entity mention**: an entity mention is a text span that refers to an entity object; 2) **event role**: an event role corresponds to a predefined field of the event table; 3) **event argument**: an event argument is an entity that plays a specific event role; 4) **event record**: an event record corresponds to an entry of the event table and contains several arguments with required roles. For example, Figure 2 shows two event records, where the entity "[PER]" is an event argument with the *Pledger* role.

To better elaborate and evaluate our proposed approach, we leverage the ChFinAnn data in this paper. ChFinAnn documents contain firsthand official disclosures of listed companies in the Chinese stock market and have hundreds of types, such as annual reports and earnings estimates. While in this work, we focus on those event-related ones that are frequent, influential, and mainly expressed by the natural language.

## 4 Document-level Event Labeling

As a prerequisite to DEE, we first conduct the DS-based event labeling at the document level. More specifically, we map tabular records from an event knowledge base to document text and regard well-matched records as events expressed by that document. Moreover, we adopt a no-trigger-words design and reformalize a novel DEE task accordingly to enable end-to-end model designs.

**Event Labeling.** To ensure the labeling quality, we set two constraints for matched records: 1) arguments of predefined key event roles must exist (non-key ones can be empty) and 2) the number of matched arguments should be higher than a certain threshold. Configurations of these constraints are event-specific, and in practice, we can tune them to directly ensure the labeling quality at the document level. We regard records that meet these two constraints as the well-matched ones, which serve as distantly supervised ground truths. In addition to labeling event records, we assign roles of arguments to matched tokens as token-level entity tags. Note that we do not label trigger words explicitly. Besides not affecting the DEE functionality, an extra benefit of such no-trigger-words design is a much easier DS-based labeling that does not rely on predefined trigger-words dictionaries or manually curated heuristics to filter multiple potential trigger words.

**DEE Task Without Trigger Words.** We reformalize a novel task for DEE as directly filling event tables based on a document, which generally requires three sub-tasks: 1) **entity extraction**, extracting entity mentions as argument candidates, 2) **event detection**, judging a document to be triggered or not for each event type, and 3) **event table filling**, filling arguments into the table of triggered events. This novel DEE task is much different from the vanilla SEE with trigger words but is consistent with the above simplified DS-based event labeling.

## 5 Doc2EDAG

The key idea of Doc2EDAG is to transform tabular event records into an EDAG and let the model learn to generate this EDAG based on document-level contexts. Following the example in Figure 2, Figure 3 typically depicts an EDAG generation process and Figure 4 presents the overall workflow of Doc2EDAG, which consists of two key stages:

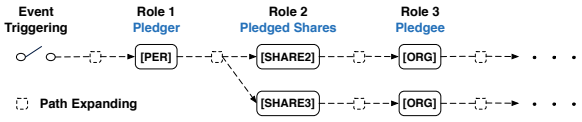


Figure 3: An EDAG generation example that starts from event triggering and expands sequentially following the predefined order of event roles.

document-level entity encoding (Section 5.1) and EDAG generation (Section 5.2). Before elaborating each of them in this section, we first describe two preconditioned modules: input representation and entity recognition.

**Input Representation.** In this paper, we denote a document as a sequence of sentences. Formally, after looking up the token embedding table $\boldsymbol{V} \in \mathbb{R}^{d_w \times |V|}$, we denote a document $\boldsymbol{d}$ as a sentence sequence $[\boldsymbol{s_1}; \boldsymbol{s_2}; \cdots; \boldsymbol{s_{N_s}}]$ and each sentence $\boldsymbol{s_i} \in \mathbb{R}^{d_w \times N_w}$ is composed of a sequence of token embeddings as $[\boldsymbol{w_{i,1}}, \boldsymbol{w_{i,2}}, \cdots, \boldsymbol{w_{i,N_w}}]$, where $|V|$ is the vocabulary size, $N_s$ and $N_w$ are the maximum lengths of the sentence sequence and the token sequence, respectively, and $\boldsymbol{w_{i,j}} \in \mathbb{R}^{d_w}$ is the embedding of $j^{th}$ token in $i^{th}$ sentence with the embedding size $d_w$.

**Entity Recognition.** Entity recognition is a typical sequence tagging task. We conduct this task at the sentence level and follow a classic method, BI-LSTM-CRF (Huang et al., 2015), that first encodes the token sequence and then adds a conditional random field (CRF) layer to facilitate the sequence tagging. The only difference is that we employ the Transformer (Vaswani et al., 2017) instead of the original encoder, LSTM (Hochreiter and Schmidhuber, 1997). Transformer encodes a sequence of embeddings by the multi-headed self-attention mechanism to exchange contextual information among them. Due to the superior performance of the Transformer, we employ it as a primary context encoder in this work and name the Transformer module used in this stage as Transformer-1. Formally, for each sentence tensor $\boldsymbol{s_i} \in \mathbb{R}^{d_w \times N_w}$, we get the encoded one as $\boldsymbol{h_i} = \text{Transformer-1}(\boldsymbol{s_i})$, where $\boldsymbol{h_i} \in \mathbb{R}^{d_w \times N_w}$ shares the same embedding size $d_w$ and sequence length $N_w$. During training, we employ roles of matched arguments as entity labels with the classic `BIO` (Begin, Inside, Other) scheme and wrap $\boldsymbol{h_i}$ with a CRF layer to get the entity-recognition loss $L_{er}$. As for the inference, we use the Viterbi

decoding to get the best tagging sequence.

### 5.1 Document-level Entity Encoding

To address the arguments-scattering challenge efficiently, it is indispensable to leverage global contexts to better identify whether an entity plays a specific event role. Consequently, we utilize document-level entity encoding to encode extracted entity mentions with such contexts and produce an embedding of size $d_w$ for each entity mention with a distinct surface name.

**Entity & Sentence Embedding.** Since an entity mention usually covers multiple tokens with a variable length, we first obtain a fixed-sized embedding for each entity mention by conducting a max-pooling operation over its covered token embeddings. For example, given $l^{th}$ entity mention covering $j^{th}$ to $k^{th}$ tokens of $i^{th}$ sentence, we conduct the max-pooling over $[\boldsymbol{h_{i,j}}, \cdots, \boldsymbol{h_{i,k}}]$ to get the entity mention embedding $\boldsymbol{e}_l \in \mathbb{R}^{d_w}$. For each sentence $\boldsymbol{s_i}$, we also take the max-pooling operation over the encoded token sequence $[\boldsymbol{h_{i,1}}, \cdots, \boldsymbol{h_{i,N_w}}]$ to obtain a single sentence embedding $\boldsymbol{c_i} \in \mathbb{R}^{d_w}$. After these operations, both the mention and the sentence embeddings share the same embedding size $d_w$.

**Document-level Encoding.** Though we get embeddings for all sentences and entity mentions, these embeddings only encode local contexts within the sentence scope. To enable the awareness of document-level contexts, we employ the second Transformer module, Transformer-2, to facilitate the information exchange between all entity mentions and sentences. Before feeding them into Transformer-2, we add them with sentence position embeddings to inform the sentence order. After the Transformer encoding, we utilize the max-pooling operation again to merge multiple mention embeddings with the same entity surface name into a single embedding. Formally, after this stage, we obtain document-level context-aware entity mention and sentence embeddings as $\boldsymbol{e^d} = [\boldsymbol{e^d_1}, \cdots, \boldsymbol{e^d_{N_e}}]$ and $\boldsymbol{c^d} = [\boldsymbol{c^d_1}, \cdots, \boldsymbol{c^d_{N_s}}]$, respectively, where $N_e$ is the number of distinct entity surface names. These aggregated embeddings serve the next stage to fill event tables directly.

### 5.2 EDAG Generation

After the document-level entity encoding stage, we can obtain the document embedding $\boldsymbol{t} \in \mathbb{R}^{d_w}$ by operating the max-pooling over the sentence tensor $\boldsymbol{c^d} \in \mathbb{R}^{d_w \times N_s}$ and stack a linear classifier over $\boldsymbol{t}$ to conduct the event-triggering classification for each event type. Next, for each triggered event type, we learn to generate an EDAG.

**EDAG Building.** Before the model training, we need to build the EDAG from tabular event records. For each event type, we first manually define an event role order. Then, we transform each event record into a linked list of arguments following this order, where each argument node is either an entity or a special empty argument `NA`. Finally, we merge these linked lists into an EDAG by sharing the same prefix path. Since every complete path of the EDAG corresponds to one row of the event table, recovering the table format from a given EDAG is simple.

**Task Decomposition.** The EDAG format aims to simplify the hard table-filling task into several tractable path-expanding sub-tasks. Then, a natural question is how the task decomposition works, which can be answered by the following EDAG recovering procedure. Assume the event triggering as the starting node (the initial EDAG), there comes a series of path-expanding sub-tasks following a predefined event role order. When considering a certain role, for every leaf node of the current EDAG, there is a path-expanding sub-task that decides which entities to be expanded. For each entity to be expanded, we create a new node of that entity for the current role and expand the path by connecting the current leaf node to the new entity node. If no entity is valid for expanding, we create a special `NA` node. When all sub-tasks for the current role finish, we move to the next role and repeat until the last. In this work, we leverage the above logic to recover the EDAG from path-expanding predictions at inference and to set associated labels for each sub-task when training.

**Memory.** To better fulfill each path-expanding sub-task, it is crucial to know entities already contained by the path. Hence, we design a memory mechanism that initializes a memory tensor $\boldsymbol{m}$ with the sentence tensor $\boldsymbol{c^d}$ at the beginning and updates $\boldsymbol{m}$ when expanding the path by appending either the associated entity embedding or the zero-padded one for the `NA` argument. With this design, each sub-task can own a distinct memory tensor, corresponding to the unique path history.

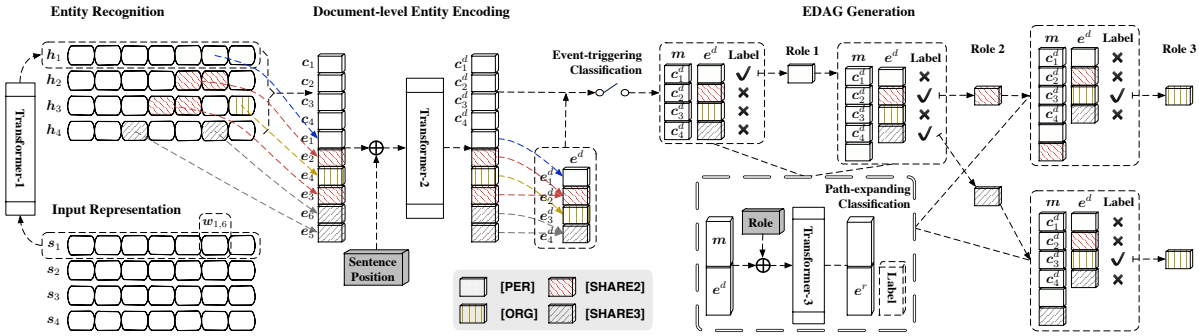


Figure 4: The overall workflow of Doc2EDAG, where we follow the example in Figure 2 and the EDAG structure in Figure 3, and use stripes to differentiate different entities (note that the number of input tokens and entity positions are imaginary, which do not match previous ones strictly, and here we only include the first three event roles and associated entities for brevity).

**Path Expanding.** For each path-expanding subtask, we formalize it as a collection of multiple binary classification problems, that is predicting *expanding* (1) or *not* (0) for all entities. To enable the awareness of the current path state, history contexts and the current event role, we first concatenate the memory tensor $\boldsymbol{m}$ and the entity tensor $\boldsymbol{e^d}$, then add them with a trainable event-role-indicator embedding, and encode them with the third Transformer module, Transformer-3, to facilitate the context-aware reasoning. Finally, we extract the enriched entity tensor $\boldsymbol{e^r}$ from outputs of Transformer-3 and stack a linear classifier over $\boldsymbol{e^r}$ to conduct the path-expanding classification.

**Optimization.** For the event-triggering classification, we calculate the cross-entropy loss $L_{tr}$. During the EDAG generation, we calculate a cross-entropy loss for each path-expanding subtask, and sum these losses as the final EDAG-generation loss $L_{dag}$. Finally, we sum $L_{tr}$, $L_{dag}$ and the entity-recognition loss $L_{er}$ together as the final loss, $L_{all} = \lambda_1 L_{er} + \lambda_2 L_{tr} + \lambda_3 L_{dag}$, where $\lambda_1$, $\lambda_2$ and $\lambda_3$ are hyper-parameters.

**Inference.** Given a document, Doc2EDAG first recognizes entity mentions from sentences, then encodes them with document-level contexts, and finally generates an EDAG for each triggered event type by conducting a series of path-expanding sub-tasks.

**Practical Tips.** During training, we can utilize both ground-truth entity tokens and the given EDAG structure. While at inference, we need to first identify entities and then expand paths sequentially based on embeddings of those entities to recover the EDAG. This gap between training and inference can cause severe error-propagation problems. To mitigate such problems, we utilize the scheduled sampling (Bengio et al., 2015) to gradually switch the inputs of document-level entity encoding from ground-truth entity mentions to model recognized ones. Moreover, for path-expanding classifications, false positives are more harmful than false negatives, because the former can cause a completely wrong path. Accordingly, we can set $\gamma(> 1)$ as the negative class weight of the associated cross-entropy loss.

## 6 Experiments

In this section, we present thorough empirical studies to answer the following questions: 1) to what extent can Doc2EDAG improve over state-of-the-art methods when facing DEE-specific challenges? 2) how do different models behave when facing both *arguments-scattering* and *multi-event* challenges? 3) how important are various components of Doc2EDAG?

### 6.1 Experimental Setup

**Data Collection with Event Labeling.** We utilize ten years (2008-2018) ChFinAnn[4] documents and human-summarized event knowledge bases to conduct the DS-based event labeling. We focus on five event types: *Equity Freeze* (EF), *Equity Repurchase* (ER), *Equity Underweight* (EU), *Equity Overweight* (EO) and *Equity Pledge* (EP), which belong to major events required to be disclosed by the regulator and may have a huge impact on the company value. To ensure the labeling quality, we set constraints for matched document-record pairs

[4] Crawling from http://www.cninfo.com.cn/new/index

| Event | #Train | #Dev | #Test | #Total | MER (%) |
|---|---|---|---|---|---|
| EF | 806 | 186 | 204 | 1,196 | 32.0 |
| ER | 1,862 | 297 | 282 | 3,677 | 16.1 |
| EU | 5,268 | 677 | 346 | 5,847 | 24.3 |
| EO | 5,101 | 570 | 1,138 | 6,017 | 28.0 |
| EP | 12,857 | 1,491 | 1,254 | 15,602 | 35.4 |
| All | 25,632 | 3,204 | 3,204 | 32,040 | 29.0 |

Table 1: Dataset statistics about the number of documents for the train (#Train), development (#Dev) and test (#Test), the number (#Total) and the multi-event ratio (MER) of all documents.

| Precision | Recall | F1 | MER (%) |
|---|---|---|---|
| 98.8 | 89.7 | 94.0 | 31.0 |

Table 2: The quality of the DS-based event labeling evaluated on 100 manually annotated documents (randomly select 20 for each event type).

as Section 4 describes. Moreover, we directly use the character tokenization to avoid error propagations from Chinese word segmentation tools.

Finally, we obtain 32,040 documents in total, and this number is ten times larger than 2,976 of DCFEE and about 53 times larger than 599 of ACE 2005. We divide these documents into train, development, and test set with the proportion of 8 : 1 : 1 based on the time order. In Table 1, we show the number of documents and the multi-event ratio (MER) for each event type on this dataset. Note that a few documents may contain multiple event types at the same time.

**Data Quality.** To verify the quality of DS-based event labeling, we randomly select 100 documents and manually annotate them. By regarding DS-generated event tables as the prediction and human-annotated ones as the ground-truth, we evaluate the labeling quality based on the metric introduced below. Table 2 shows this approximate evaluation, and we can observe that DS-generated data are pretty good, achieving high precision and acceptable recall. In later experiments, we directly employ the automatically generated test set for evaluation due to its much broad coverage.

**Evaluation Metric.** The ultimate goal of DEE is to fill event tables with correct arguments for each role. Therefore, we evaluate DEE by directly comparing the predicted event table with the ground-truth one for each event type. Specifically, for each document and each event type, we pick one predicted record and one most similar ground-truth record (at least one of them is non-empty) from associated event tables without replacement to calculate event-role-specific true positive, false positive and false negative statistics until no record left. After aggregating these statistics among all evaluated documents, we can calculate role-level precision, recall, and F1 scores (all reported in percentage format). As an event type often includes multiple roles, we calculate micro-averaged role-level scores as the final event-level metric that reflects the ability of end-to-end DEE directly.

**Hyper-parameter Setting.** For the input, we set the maximum number of sentences and the maximum sentence length as 64 and 128, respectively. During training, we set $\lambda_1 = 0.05$, $\lambda_2 = \lambda_3 = 0.95$ and $\gamma = 3$. We employ the Adam (Kingma and Ba, 2015) optimizer with the learning rate $1e^{-4}$, train for at most 100 epochs and pick the best epoch by the validation score on the development set. Besides, we leverage the decreasing order of the non-empty argument ratio as the event role order required by Doc2EDAG, because more informative entities in the path history can better facilitate later path-expanding classifications.

Note that, due to the space limit, we leave other detailed hyper-parameters, model structures, data preprocessing configurations, event type specifications and pseudo codes for EDAG generation to the appendix.

### 6.2 Performance Comparisons

**Baselines.** As discussed in the related work, the state-of-the-art method applicable to our setting is DCFEE. We follow the implementation described in (Yang et al., 2018), but they did not illustrate how to handle multi-event sentences with just a sequence tagging model. Thus, we develop two versions, *DCFEE-O* and *DCFEE-M*, where *DCFEE-O* only produces one event record from one key-event sentence, while *DCFEE-M* tries to get multiple possible argument combinations by the closest relative distance from the key-event sentence. To be fair, the SEE stages of both versions share the same neural architecture as the entity recognition part of Doc2EDAG. Besides, we further employ a simple decoding baseline of Doc2EDAG, *Greedy-Dec*, that only fills one event table entry greedily by using recognized entity roles to verify the necessity of end-to-end modeling.

**Main Results.** As Table 3 shows, Doc2EDAG achieves significant improvements over all base-

| Model | EF | | | ER | | | EU | | | EO | | | EP | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | P. | R. | F1 | P. | R. | F1 | P. | R. | F1 | P. | R. | F1 | P. | R. | F1 |
| DCFEE-O | 66.0 | 41.6 | 51.1 | 84.5 | 81.8 | 83.1 | 62.7 | 35.4 | 45.3 | 51.4 | 42.6 | 46.6 | 64.3 | 63.6 | 63.9 |
| DCFEE-M | 51.8 | 40.7 | 45.6 | 83.7 | 78.0 | 80.8 | 49.5 | 39.9 | 44.2 | 42.5 | 47.5 | 44.9 | 59.8 | 66.4 | 62.9 |
| GreedyDec | **79.5** | 46.8 | 58.9 | 83.3 | 74.9 | 78.9 | 68.7 | 40.8 | 51.2 | 69.7 | 40.6 | 51.3 | **85.7** | 48.7 | 62.1 |
| Doc2EDAG | 77.1 | **64.5** | **70.2** | **91.3** | **83.6** | **87.3** | **80.2** | **65.0** | **71.8** | **82.1** | **69.0** | **75.0** | 80.0 | **74.8** | **77.3** |

Table 3: Overall event-level precision (P.), recall (R.) and F1 scores evaluated on the test set.

| Model | EF | | ER | | EU | | EO | | EP | | Avg. | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | S. | M. | S. | M. | S. | M. | S. | M. | S. | M. | S. | M. | S. & M. |
| DCFEE-O | 56.0 | 46.5 | 86.7 | 54.1 | 48.5 | 41.2 | 47.7 | 45.2 | 68.4 | 61.1 | 61.5 | 49.6 | 58.0 |
| DCFEE-M | 48.4 | 43.1 | 83.8 | 53.4 | 48.1 | 39.6 | 47.1 | 42.0 | 67.0 | 60.6 | 58.9 | 47.7 | 55.7 |
| GreedyDec | 75.9 | 40.8 | 81.7 | 49.8 | 62.2 | 34.6 | 65.7 | 29.4 | **88.5** | 42.3 | 74.8 | 39.4 | 60.5 |
| Doc2EDAG | **80.0** | **61.3** | **89.4** | **68.4** | **77.4** | **64.6** | **79.4** | **69.5** | 85.5 | **72.5** | **82.3** | **67.3** | **76.3** |

Table 4: F1 scores for all event types and the averaged ones (Avg.) on single-event (S.) and multi-event (M.) sets.

| Model | EF | ER | EU | EO | EP | Avg. |
|---|---|---|---|---|---|---|
| Doc2EDAG | 70.2 | 87.3 | 71.8 | 75.0 | 77.3 | 76.3 |
| -PathMem | -11.2 | -0.2 | -10.1 | -16.3 | -10.9 | -9.7 |
| -SchSamp | -5.3 | -4.8 | -5.3 | -6.6 | -3.0 | -5.0 |
| -DocEnc | -4.7 | -1.5 | -1.6 | -1.1 | -1.5 | -2.1 |
| -NegCW | -1.4 | -0.4 | -0.7 | -1.3 | -0.4 | -0.8 |

Table 5: Performance differences of Doc2EDAG variants for all event types and the averaged ones (Avg.).

lines for all event types. Specifically, Doc2EDAG improves 19.1, 4.2, 26.5, 28.4 and 13.4 F1 scores over DCFEE-O, the best baseline, on EF, ER, EU, EO and EP events, respectively. These vast improvements mainly owe to the document-level end-to-end modeling of Doc2EDAG. Moreover, since we work on automatically generated data, the direct document-level supervision can be more robust than the extra sentence-level supervision used in DCFEE, which assumes the sentence containing most event arguments as the key-event one. This assumption does not work well on some event types, such as EF, EU and EO, on which DCFEE-O is even inferior to the most straightforward baseline, GreedyDec. Besides, DCFEE-O achieves better results than DCFEE-M, which demonstrates that naively guessing multiple events from the key-event sentence cannot work well. By comparing Doc2EDAG with GreedyDec that owns high precision but low recall, we can clearly see the benefit of document-level end-to-end modeling.

**Single-Event vs. Multi-Event.** We divide the test set into a single-event set, containing documents with just one event record, and a multi-event set, containing others, to show the extreme difficulty when *arguments-scattering* meets *multi-event*. Table 4 shows F1 scores for different scenarios. Although Doc2EDAG still maintains the highest extraction performance for all cases, the multi-event set is extremely challenging as the extraction performance of all models drops significantly. Especially, GreedyDec, with no mechanism for the *multi-event* challenge, decreases most drastically. DCFEE-O decreases less, but is still far away from Doc2EDAG. On the multi-event set, Doc2EDAG increases by 17.7 F1 scores over DCFEE-O, the best baseline, on average.

**Ablation Tests.** To demonstrate key designs of Doc2EDAG, we conduct ablation tests by evaluating four variants: 1) *-PathMem*, removing the memory mechanism used during the EDAG generation, 2) *-SchSamp*, dropping the scheduled sampling strategy during training, 3) *-DocEnc*, removing the Transformer module used for document-level entity encoding, and 4) *-NegCW*, keeping the negative class weight as 1 when doing path-expanding classifications. From Table 5, we can observe that 1) the memory mechanism is of prime importance, as removing it can result in the most drastic performance declines, over 10 F1 scores on four event types except for the ER type whose MER is very low on the test set; 2) the scheduled sampling strategy that alleviates the mismatch of entity candidates for event table filling between training and inference also contributes greatly, improving by 5 F1 scores on average; 3) the document-level entity encoding that enhances global entity representations contributes

2.1 F1 scores on average; 4) the larger negative class weight to penalize false positive path expanding can also make slight but stable contributions for all event types.

**Case Studies.** Let us follow the example in Figure 2, Doc2EDAG can successfully recover the correct EDAG, while DCFEE inevitably makes many mistakes even with a perfect SEE model, as discussed in the introduction. Due to the space limit, we leave another three fine-grained case studies to the appendix.

## 7 Conclusion and Future Work

Towards the end-to-end modeling for DEE, we propose a novel model, Doc2EDAG, associated with a novel task formalization without trigger words to ease DS-based labeling. To validate the effectiveness of the proposed approach, we build a large-scale real-world dataset in the financial domain and conduct extensive empirical studies. Notably, without any domain-specific assumption, our general labeling and modeling strategies can benefit practitioners in other domains directly.

As this work shows promising results for the end-to-end DEE, expanding the inputs of Doc2EDAG from pure text sequences to richly formatted ones (Wu et al., 2018) is appealing, and we leave it as future work to explore.

## Acknowledgements

This work is supported in part by the National Natural Science Foundation of China (NSFC) Grant 61532001 and the Zhongguancun Haihua Institute for Frontier Information Technology.

## References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *NIPS*.

Yubo Chen, Shulin Liu, Xiang Zhang, Kang Liu, and Jun Zhao. 2017. Automatically labeled data generation for large scale event extraction. In *ACL*.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *ACL*.

Jun Feng, Minlie Huang, Li Zhao, Yang Yang, and Xiaoyan Zhu. 2018. Reinforcement learning for relation classification from noisy data. In *AAAI*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*.

Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. 2011. Using cross-entity inference to improve event extraction. In *ACL*.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *ACL*.

Alex Judea and Michael Strube. 2016. Incremental global event extraction. In *COLING*.

Diederick P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *ACL*.

Qi Li, Heng Ji, HONG Yu, and Sujian Li. 2014. Constructing information networks using one single model. In *EMNLP*.

Shasha Liao and Ralph Grishman. 2010. Using document level cross-event inference to improve event extraction. In *ACL*.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *ACL*.

Shulin Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2017. Exploiting argument information to improve event detection via supervised attention mechanisms. In *ACL*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACL*.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *NAACL*.

Trung Minh Nguyen and Thien Huu Nguyen. 2019. One for all: Neural joint modeling of entities and events. In *AAAI*.

Xiang Ren, Zeqiu Wu, Wenqi He, Meng Qu, Clare R Voss, Heng Ji, Tarek F Abdelzaher, and Jiawei Han. 2017. CoType: Joint extraction of typed entities and relations with knowledge bases. In *WWW*.

Sebastian Riedel and Andrew McCallum. 2011. Fast and robust joint models for biomedical event extraction. In *EMNLP*.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *ECML*.

Lei Sha, Feng Qian, Baobao Chang, and Zhifang Sui. 2018. Jointly extracting event triggers and arguments by dependency-bridge rnn and tensor-based argument interaction. In *AAAI*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Shaolei Wang, Yue Zhang, Wanxiang Che, and Ting Liu. 2018. Joint extraction of entities and relations based on a novel graph scheme. In *IJCAI*.

Xiaozhi Wang, Xu Han, Zhiyuan Liu, Maosong Sun, and Peng Li. 2019. Adversarial training for weakly supervised event detection. In *NAACL*.

Sen Wu, Luke Hsiao, Xiao Cheng, Braden Hancock, Theodoros Rekatsinas, Philip Levis, and Christopher Ré. 2018. Fonduer: Knowledge base construction from richly formatted data. In *SIGMOD*.

Bishan Yang and Tom M. Mitchell. 2016. Joint extraction of events and entities within a document context. In *NAACL*.

Hang Yang, Yubo Chen, Kang Liu, Yang Xiao, and Jun Zhao. 2018. DCFEE: A document-level chinese financial event extraction system based on automatically labeled training data. In *Proceedings of ACL 2018, System Demonstrations*.

Xiangrong Zeng, Daojian Zeng, Shizhu He, Kang Liu, and Jun Zhao. 2018a. Extracting relational facts by an end-to-end neural model with copy mechanism. In *ACL*.

Ying Zeng, Yansong Feng, Rong Ma, Zheng Wang, Rui Yan, Chongde Shi, and Dongyan Zhao. 2018b. Scale up event extraction learning via automatic training data generation. In *AAAI*.

Tongtao Zhang and Heng Ji. 2018. Event extraction with generative adversarial imitation learning. *arXiv preprint arXiv:1804.07881*.

Shun Zheng, Xu Han, Yankai Lin, Peilin Yu, Lu Chen, Ling Huang, Zhiyuan Liu, and Wei Xu. 2019. DIAG-NRE: A neural pattern diagnosis framework for distantly supervised neural relation extraction. In *ACL*.

Suncong Zheng, Feng Wang, Hongyun Bao, Yuexing Hao, Peng Zhou, and Bo Xu. 2017. Joint extraction of entities and relations based on a novel tagging scheme. In *ACL*.

# Event Detection with Trigger-Aware Lattice Neural Network

**Ning Ding**[1,2*], **Ziran Li**[1,2*], **Zhiyuan Liu**[2,3,4], **Hai-Tao Zheng**[1,2†], **Zibo Lin**[1,2]
[1]Tsinghua Shenzhen International Graduate School, Tsinghua University
[2]Department of Computer Science and Technology, Tsinghua University, Beijing, China
[3]Institute for Artificial Intelligence, Tsinghua University, Beijing, China
[4]State Key Lab on Intelligent Technology and Systems, Tsinghua University, Beijing, China
{dingn18}@mails.tsinghua.edu.cn

## Abstract

Event detection (ED) aims to locate trigger words in raw text and then classify them into correct event types. In this task, neural network based models became mainstream in recent years. However, two problems arise when it comes to languages without natural delimiters, such as Chinese. First, word-based models severely suffer from the problem of word-trigger mismatch, limiting the performance of the methods. In addition, even if trigger words could be accurately located, the ambiguity of polysemy of triggers could still affect the trigger classification stage. To address the two issues simultaneously, we propose the Trigger-aware Lattice Neural Network (TLNN). (1) The framework dynamically incorporates word and character information so that the trigger-word mismatch issue can be avoided. (2) Moreover, for polysemous characters and words, we model all senses of them with the help of an external linguistic knowledge base, so as to alleviate the problem of ambiguous triggers. Experiments on two benchmark datasets show that our model could effectively tackle the two issues and outperforms previous state-of-the-art methods significantly, giving the best results. The source code of this paper can be obtained from https://github.com/thunlp/TLNN.

## 1 Introduction

Event Detection (ED) is a pivotal part of Event Extraction, which aims to detect the position of event triggers in raw text and classify them into corresponding event types. Conventionally, the stage of locating trigger words is known as **Trigger Identification (TI)**, and the stage of classifying trigger words into particular event types

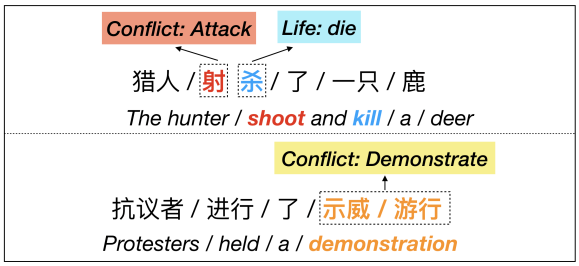


(a) An example of trigger-word mismatch.




(b) An example of polysemous trigger.

Figure 1: Examples about trigger-word mismatch and polysemous trigger in Event Detection.

is called **Trigger Classification (TC)**. Although neural network methods have achieved significant progress in event detection (Nguyen and Grishman, 2015; Chen et al., 2015; Zeng et al., 2016), both steps are still exposed to the following two issues.

In the TI stage, the problem of **trigger-word mismatch** could severely impact the performance of event detection systems. Because in languages without natural delimiters such as Chinese, mainstream approaches are mostly word-based models, in which the segmentation should be firstly performed as a necessary preprocessing step. Unfortunately, these word-wise methods neglect an important problem that a trigger could be a specific part of one word or contain multiple words. As shown in Figure 1(a), "射" (shoot) and "杀" (kill)

* indicates equal contribution
† Corresponding author: Hai-Tao Zheng. ( E-mail: zheng.haitao@sz.tsinghua.edu.cn )



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 347–356, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| Datasets | Trigger-word Match | | Trigger Polysemy | |
|---|---|---|---|---|
| | Match | Mismatch | Polysemy | Univocal |
| ACE 2005 | 85.39% | 14.61% | 41.64% | 58.36% |
| KBP 2017 | 76.28% | 23.72% | 52.14% | 47.86% |

Table 1: Proportion of Trigger-word mismatch and Polysemous Triggers on ACE 2005 and KBP 2017.

are two triggers that both are parts of the word "射杀" (shoot and kill) . In the other case, "示威游行" (demonstration) is a trigger that crosses two words. Under this circumstance, triggers could not be located correctly with word-based methods, thereby becoming a serious limitation of the task. Some feature-based methods are proposed (Chen and Ji, 2009; Qin et al., 2010; Li and Zhou, 2012) to alleviate the issue, but they heavily rely on the hand-crafted features. Lin et al. (2018) proposes the nugget proposal networks (NPN) in terms of this issue, which uses a neural network to model character compositional structure of trigger words in a fix-sized window. However, the mechanism of the NPN limits the scope of trigger candidates within a fix-sized window, which is inflexible and suffering from the problem of trigger overlaps.

Even if the locations of triggers can be correctly detected in the TI step, the TC step could still be severely affected by the inherent problem of **ambiguity of polysemy**. Because a trigger word with multiple word senses could be classified into different event types. Take 1(b) as an example, a polysemous trigger word "释放" (release) could represent two distinctly different event types. In the first case, the word 'release' triggers an *Attack* event (release tear gas). But in the second case, the event triggered by 'release' becomes *Release-Parole* (release a man in court).

To further illustrate that the two problems mentioned above do exist, we make manual statistics on the proportion of mismatch triggers and polysemous triggers on two widely used datasets. The statistics are illustrated in Table 1, and we can observe that data with trigger-word mismatch and trigger polysemy do account for a considerable proportion and then affect the task.

In this paper, we propose the Trigger-aware Lattice Network (TLNN), a comprehensive model that can simultaneously tackle both issues. To avoid error propagation by NLP tools like segmentor, we take characters as the basic units of the input sequence. Moreover, we utilize HowNet (Dong and Dong, 2003), an external knowledge base that manually annotates polysemous Chinese and English words, to obtain the sense-level information. Further, we develop the trigger-aware lattice LSTM as the feature extractor of our model, which could leverage character-level, word-level and sense-level information at the same time. More specifically, in order to address the trigger-word mismatch issue, we construct short cut paths to link the cell state between the start and the end characters for each word. It is worth mentioning that the paths are sense-level, which means all the sense information of words that end in one specific character will flow into the memory cell of the character. Hence, with the utilization of multiple granularity of information (character, word and word sense), the problem of polysemous triggers could be effectively alleviated.

We conduct sets of experiments on two real-world datasets in the task of event detection. Empirical results of the main experiments show that our model can efficiently address both mentioned issues. With comprehensive comparisons with other proposed methods, our model achieves the state-of-the-art results on both datasets. Further, sets of subsidiary experiments are conducted to further analyze how TLNN addresses the two issues.

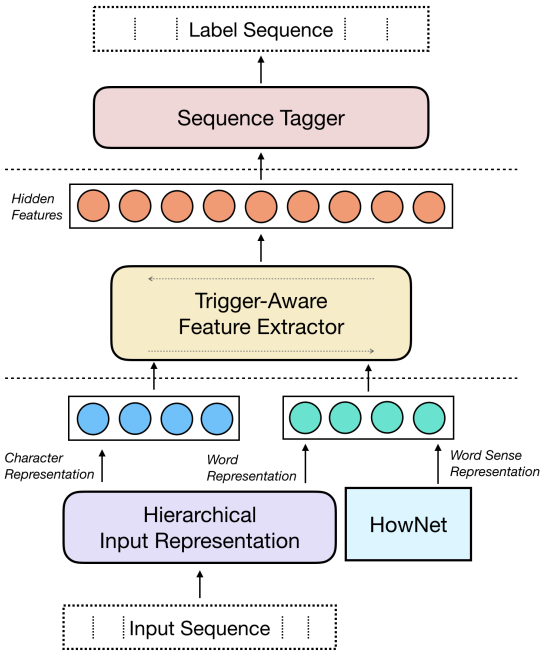


Figure 2: The architecture of TLNN.

## 2 Methodology

In the paper, event detection is regarded as a sequence labelling task. For each character, the model should identify if it is a part of one trigger and correctly classify the trigger into one specific event type.

The architecture of our model is shown in Figure 2, which primarily includes the following three parts: (1) **Hierarchical Representation Learning**, which reveals the character-level, word-level and sense-level embedding vectors in an unsupervised way. (2) **Trigger-aware Feature Extractor**, which automatically extracts different levels of semantic features by a tree structure LSTM model. (3) **Sequence Tagger**, which calculates the probability of being a trigger for each character candidate.

### 2.1 Hierarchical Representation Learning

Given an input sequence $S = \{c_1, c_2, ..., c_N\}$, where $c_i$ represents the $i$th character in the sequence. In character level, each character will be represented as an embedding vector $\boldsymbol{x^c}$ by Skip-Gram method (Mikolov et al., 2013).

$$\boldsymbol{x}_i^c = \boldsymbol{e}(c_i) \tag{1}$$

In the word level, the input sequence $S$ could also be $S = \{w_1, w_2, ..., w_M\}$, where the basic unit is a single word $w_i$. In this paper, we will use two indexes $b$ and $e$ to represent the start and the end of a word. In this case, the word embeddings are:

$$\boldsymbol{x}_{b,e}^w = \boldsymbol{e}(w_{b,e}) \tag{2}$$

However, the Skip-Gram method maps each word to only one single embedding, ignoring the fact that many words have multiple senses. Hence representation of finer granularity is still necessary to represent deep semantics. With the help of HowNet (Dong and Dong, 2003), we can obtain the representation of each sense of a character or a word. For each character $c$, there are possible multiple senses $sen^{(c_i)} \in S^{(c)}$ annotated in HowNet. Similarly, for each word $w$, the senses could be $sen^{(w_i)} \in S^{(w)}$. Consequently, we can obtain the embeddings of senses by jointly learning word and sense embeddings via Skip-gram manner. This mechanism is also applied to (Niu et al., 2017).

$$\boldsymbol{s}_j^{c_i} = \boldsymbol{e}(sen_j^{(c_i)}) \tag{3}$$

$$\boldsymbol{s}_j^{w_{b,e}} = \boldsymbol{e}(sen_j^{(w_{b,e})}) \tag{4}$$

where $sen_j^{(c_i)}$ and $sen_j^{(w_{b,e})}$ represents the $j$th sense for the character $c_i$ and word $w_{b,e}$ in the sequence. And then $\boldsymbol{s_j^{c_i}}$ and $\boldsymbol{s_j^{w_{b,e}}}$ are the embeddings of $c_i$ and $w_{b,e}$.

### 2.2 Trigger-Aware Feature Extractor

The trigger-aware feature extractor is the core component of our model. After training, the outputs of the extractor are the hidden state vectors **h** of an input sentence.

**Conventional LSTM.** LSTM (Hochreiter and Schmidhuber, 1997) is an extension of the recurrent neural network (RNN) with additional gates to control the information. Traditionally, there are following basic gates in LSTM: input gate $\boldsymbol{i}$, output gate $\boldsymbol{o}$ and forget gate $\boldsymbol{f}$. They collectively controls which information will be reserved, forgotten and output. All three gates are accompanied by corresponding weight matrix $\boldsymbol{W}$. Current cell state $\boldsymbol{c}$ records all historical information flow up to the current time. Therefore, the character-based LSTM functions are:

$$\begin{bmatrix} \boldsymbol{i}_i^c \\ \boldsymbol{o}_i^c \\ \boldsymbol{f}_i^c \\ \tilde{\boldsymbol{c}}_i^c \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{bmatrix} (\boldsymbol{W}^{c\top} \begin{bmatrix} \boldsymbol{x}_i^c \\ \boldsymbol{h}_{i-1}^c \end{bmatrix} + \boldsymbol{b}^c) \tag{5}$$

$$\boldsymbol{c}_i^c = \boldsymbol{f}_i^c \odot \boldsymbol{c}_{i-1}^c + \boldsymbol{i}_i^c \odot \tilde{\boldsymbol{c}}_i^c \tag{6}$$

$$\boldsymbol{h}_i^c = \boldsymbol{o}_i^c \odot tanh(\boldsymbol{c}_i^c) \tag{7}$$

where $h_i^c$ is the hidden state vector.

**Trigger-Aware Lattice LSTM.** Trigger-Aware Lattice LSTM is the core feature extractor of our framework, which is an extension of LSTM and lattice LSTM. In this subsection, We will derive and theoretically analyze the model in detail.

In this section, characters and words are assumed to have $K$ senses. As mentioned in 2.1, for the $j$th sense of the $i$th character $c_i$, the embedding would be $\boldsymbol{s_j^{c_i}}$ . Then an additional LSTMCell is utilized to integrate all senses of the character, hence the calculation of the cell gate of the multi-sense character $\boldsymbol{c_i}$ would be:

$$\begin{bmatrix} \boldsymbol{i}_j^{c_i} \\ \boldsymbol{f}_j^{c_i} \\ \tilde{\boldsymbol{c}}_j^{c_i} \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ tanh \end{bmatrix} (\boldsymbol{W}^{c\top} \begin{bmatrix} \boldsymbol{s}_j^{c_i} \\ \boldsymbol{h}^{c_{i-1}} \end{bmatrix} + \boldsymbol{b}^c) \tag{8}$$

$$\boldsymbol{c}_j^{c_i} = \boldsymbol{f}_j^{c_i} \odot \boldsymbol{c}^{c_{i-1}} + \boldsymbol{i}_j^{c_i} \odot \tilde{\boldsymbol{c}}_j^{c_i} \tag{9}$$

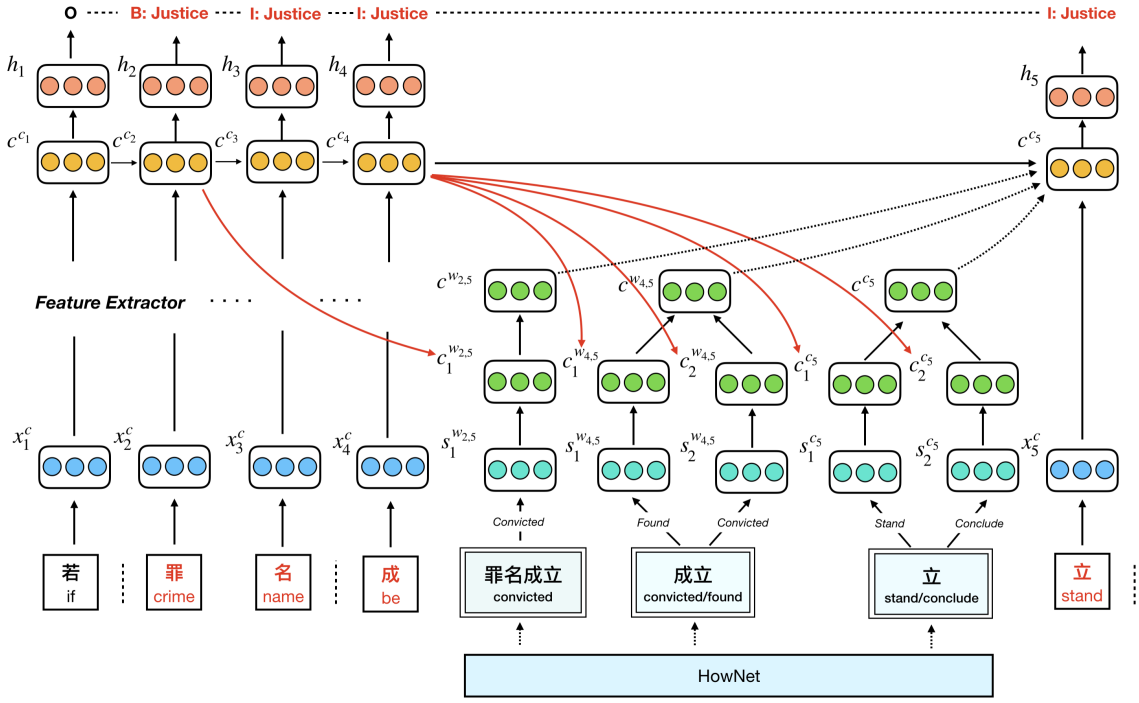


Figure 3: The structure of **Trigger-Aware Feature Extractor**, the input of the example is a part of the sentence "若罪名成立，他将被逮捕" (If convicted, he will be arrested) . In this case, "罪名成立" (convicted) is a trigger with event type *Justice: Sentence*. "成立" (convicted/found) and "立" (stand/conclude) are polysemous words. To keep the figure concise, we (1) only show two senses for each polysemous word; (2) only show the forward direction.

where $\boldsymbol{c}_j^{c_i}$ is the cell state of the $j$th sense of the $i$th character, $\boldsymbol{c}^{c_{i-1}}$ is the final cell state of the $i-1$th character. In order to obtain the cell state of the character, an additional gate is used:

$$\boldsymbol{g}_j^{c_i} = \sigma(\boldsymbol{W}^\top \begin{bmatrix} \boldsymbol{x}_i^c \\ \boldsymbol{c}_j^{c_i} \end{bmatrix} + \boldsymbol{b}) \tag{10}$$

Then all the senses should be dynamically integrated into the temporary cell state:

$$\boldsymbol{c}^{*c_i} = \sum_j^K \boldsymbol{\alpha}_j^{c_i} \odot \boldsymbol{c}_j^{c_i} \tag{11}$$

where $\boldsymbol{\alpha}_j^{c_i}$ is the character sense gate after normalization:

$$\boldsymbol{\alpha}_j^{c_i} = \frac{exp(\boldsymbol{g}_j^{c_i})}{\sum_k^K exp(\boldsymbol{g}_k^{c_i})} \tag{12}$$

Eq.11 obtains the temporary cell state of the character $c^{*c_i}$ by incorporating all the senses information of the character. However, word-level information needs to be considered as well. As mentioned in 2.1, $\boldsymbol{s}_j^{w_{b,e}}$ is the embedding for the $j$th sense of the word out $w_{b,e}$. Similar to characters, extra LSTMCell is used to calculate the cell state of each word that matches the lexicon $\mathbb{D}$.

$$\begin{bmatrix} \boldsymbol{i}_j^{w_{b,e}} \\ \boldsymbol{f}_j^{w_{b,e}} \\ \boldsymbol{c}_j^{w_{b,e}} \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ tanh \end{bmatrix} (\boldsymbol{W}^{c\top} \begin{bmatrix} \boldsymbol{s}_j^{w_{b,e}} \\ \boldsymbol{h}^{c_{b-1}} \end{bmatrix} + \boldsymbol{b}^c) \tag{13}$$

$$\boldsymbol{c}_j^{w_{b,e}} = \boldsymbol{f}_j^{w_{b,e}} \odot \boldsymbol{c}^{c_{b-1}} + \boldsymbol{i}_j^{w_{b,e}} \odot \tilde{\boldsymbol{c}}_j^{w_{b,e}} \tag{14}$$

Similar to Eq.11, the cell state of the word could be computed by incorporating all the cells of senses.

$$\boldsymbol{g}_j^{w_{b,e}} = \sigma(\boldsymbol{W}^\top \begin{bmatrix} \boldsymbol{x}_b^c \\ \boldsymbol{c}_j^{w_{b,e}} \end{bmatrix} + \boldsymbol{b}) \tag{15}$$

$$\boldsymbol{c}^{w_{b,e}} = \sum_j^K \boldsymbol{\alpha}_j^{w_{b,e}} \odot \boldsymbol{c}_j^{w_{b,e}} \tag{16}$$

where $\boldsymbol{\alpha}_j^{w_{b,e}}$ is the word sense gate after normalization:

$$\boldsymbol{\alpha}_j^{w_{b,e}} = \frac{exp(\boldsymbol{g}_j^{w_{b,e}})}{\sum_k^K exp(\boldsymbol{g}_k^{w_{b,e}})} \tag{17}$$

For a character $c_i$, the temporary cell state $\boldsymbol{c}^{*c_i}$ that contains sense information is calculated by Eq.11. Moreover, we could calculate all the cell states of words that end in the index $i$ by Eq.16, which are represented as $\{\boldsymbol{c}^{w_{b,i}}|b \in [1,i], w_{b,i} \in \mathbb{D}\}$. In order to ensure the corresponding information could flow into the final cell state of $c_i$, an extra gate $\boldsymbol{g}^m_{b,i}$ is used to merge character and word cells:

$$\boldsymbol{g}^m_{b,i} = \sigma(\boldsymbol{W}^{l\top} \begin{bmatrix} \boldsymbol{x}^c_i \\ \boldsymbol{c}^w_{b,i} \end{bmatrix} + \boldsymbol{b}^l) \tag{18}$$

and the computation of the final cell state of the character $\boldsymbol{c}^{c_i}$ is:

$$\boldsymbol{c}^{c_i} = \sum_{b \in \{b'|w^d_{b',i} \in \mathbb{D}\}} \boldsymbol{\alpha}^{w_{b,i}} \odot \boldsymbol{c}^{w_{b,i}} + \boldsymbol{\alpha}^{c_i} \odot \boldsymbol{c}^{*c_i} \tag{19}$$

where $\boldsymbol{\alpha}^{w_{b,i}}$ and $\boldsymbol{\alpha}^{c_i}$ are word gate and character gate after normalization. The computation is similar to Eq. 12 and Eq. 17.

Therefore, the final cell state $\boldsymbol{c}^{c_i}$ could represent the ambiguous characters and words in a dynamic manner. Similar to Eq. 7, hidden state vectors could be calculated to transmit to the sequence tagger layer.

### 2.3 Sequence Tagger

In this paper, the event detection task is regarded as a sequence tagging problem. For an input sequence $S = \{c_1, c_2, ..., c_N\}$, there is a corresponding label sequence $L = \{y_1, y_2, ..., y_N\}$. Hidden vectors $\boldsymbol{h}$ for each character obtained in 2.2 are used as the input. We use a classic CRF layer to perform the sequence tagging, thus the probability distribution is:

$$P(L|S) = \frac{exp(\sum_{i=1}^{N}(S(y_i) + T(y_{i-1}, y_i)))}{\sum_{L' \in \mathbb{C}} exp(\sum_{i=1}^{N}(S(y'_i) + T(y'_{i-1}, y'_i)))}, \tag{20}$$

where $S$ is the score function to compute the emission score from hidden vector $\boldsymbol{h_i}$ to the label $y_i$:

$$S(y_i) = \boldsymbol{W}^{y_i}_{CRF} \boldsymbol{h}_i + b^{y_i}_{CRF}. \tag{21}$$

$\boldsymbol{W}^{y_i}_{CRF}$ and $b^{y_i}_{CRF}$ are learned parameters specific to $y_i$. And in Eq. 20, $T$ is the transition function to compute the transition score from $y_{i-1}$ to $y_i$. $\mathbb{C}$ contains all the possible label sequences on sequence $S$ and $L'$ is a random label sequence in $\mathbb{C}$.

We use standard Viterbi (Viterbi, 1967) algorithm as a decoder to decode the highest scored label sequence. The loss function of our model is log-likelihood in sentence-level.

$$Loss = \sum_{i=1}^{M} log(P(L_i|S_i)) \tag{22}$$

where $M$ is the number of sentences, $L_i$ is the correct label for the sentence $S_i$.

## 3 Experiments

### 3.1 Datasets and Experimental Settings

**Datasets.** In this paper, we conduct a series of experiments on two real-world datasets: ACE 2005 Chinese dataset (ACE2005) and TAC KBP 2017 Event Nugget Detection Evaluation dataset (KBP2017). For better comparison, we use the same data split as previous works (Chen and Ji, 2009; Zeng et al., 2016; Feng et al., 2018; Lin et al., 2018). Specifically, the ACE2005 (LDC2006T06) contains 697 articles, with 569 articles for training, 64 for validating and the rest 64 for testing. For KBP2017 Chinese dataset (LDC2017E55), we follow the same setup as Lin et al. (2018), using 506/20/167 documents as training/development/test set respectively.

**Evaluation Metrics.** Standard micro-averaged Precision, Recall and F1 are used as evaluation metrics. For ACE2005 the computation is the same as Chen and Ji (2009). To remain rigorous, we use the official evaluation toolkit [1] to perform the metrics for KBP2017.

**Hyper-Parameter Settings.** We tune the parameters of our models by grid searching on the validation dataset. Adam (Kingma and Ba, 2014) with a learning rate decay is utilized as the optimizer. The embedding sizes of characters and senses are all 50. To avoid overfitting, Dropout mechanism (Srivastava et al., 2014) is used in the system, and the dropout rate is set to 0.5. We select the best models by early stopping using the F1 results on the validation dataset. Because of the limited influence, we follow empirical settings for other hyper-parameters.

### 3.2 Overall Results

In this section, we compare our model with previous state-of-the-art methods. The proposed models are as follows:

[1] github.com/hunterhector/EvmEval

| Model | | ACE2005 | | | | | | KBP2017 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Trigger Identification | | | Trigger Classification | | | Trigger Identification | | | Trigger Classification | | |
| | | P | R | F1 | P | R | F1 | P | R | F1 | P | R | F1 |
| Char | DMCNN | 60.10 | 61.60 | 60.90 | 57.10 | 58.50 | 57.80 | 53.67 | 49.92 | 51.73 | 50.03 | 46.53 | 48.22 |
| | C-BiLSTM* | 65.60 | 66.70 | 66.10 | 60.00 | 60.90 | 60.40 | - | - | - | - | - | - |
| | HBTNGMA | 41.67 | 59.29 | 48.94 | 38.74 | 55.13 | 45.50 | 40.52 | 46.76 | 43.41 | 35.93 | 41.47 | 38.50 |
| Word | DMCNN | 66.60 | 63.60 | 65.10 | 61.60 | 58.80 | 60.20 | 60.43 | 51.64 | 55.69 | 54.81 | 46.84 | 50.51 |
| | HNN* | **74.20** | 63.10 | 68.20 | **77.10** | 53.10 | 63.00 | - | - | - | - | - | - |
| | HBTNGMA | 54.29 | 62.82 | 58.25 | 49.86 | 57.69 | 53.49 | 46.92 | 53.57 | 50.02 | 37.54 | 42.86 | 40.03 |
| Feature | Rich-C* | 62.20 | 71.90 | 66.70 | 58.90 | 68.10 | 63.20 | - | - | - | - | - | - |
| | KBP2017 Best* | - | - | - | - | - | - | **67.76** | 45.92 | 54.74 | **62.69** | 42.48 | 50.64 |
| Hibird | NPN | 70.63 | 64.74 | 67.56 | 67.13 | 61.54 | 64.21 | 58.03 | **59.91** | 58.96 | 52.04 | 53.73 | 52.87 |
| | **TLNN (Ours)** | 67.34 | **74.68** | **70.82** | 64.45 | **71.47** | **67.78** | 65.93 | 59.07 | **62.31** | 60.72 | **54.41** | **57.39** |

Table 2: Overall results of proposed methods and TLNN on ACE2005 and KBP2017. * indicates the results adapted from the original paper. For KBP2017, "Trigger Identification" and "Trigger Classification" correspond to the "Span" and "Type" metrics in the official evaluation.

**DMCNN** (Chen et al., 2015) put forward a dynamic Multi-pooling CNN as a sentence-level feature extractor. Moreover, we add a classifier to DMCNN using IOB encoding.

**C-BiLSTM** (Zeng et al., 2016) put forward the Convolutional Bi-LSTM model for the event detection task.

**HNN** (Feng et al., 2018) designed a Hybrid Neural Network model which combines CNN with Bi-LSTM.

**HBTNGMA** (Chen et al., 2018) put forward a Hierarchical and Bias Tagging Networks with Gated Multi-level Attention Mechanisms to integrate sentence-level and document-level information collectively.

**NPN** (Lin et al., 2018) proposed a comprehensive model by automatically learning the inner compositional structures of triggers to solve the trigger mismatch problem.

The results of all the models are shown in Table 2. From the results, we can observe that:

(1) Both for ACE2005 and KBP2017, TLNN outperform other proposed models significantly, achieving the best results on two datasets. This demonstrates that the trigger-aware lattice structure could enhance the accuracy of locating triggers. Further, thanks to the usage of sense-level information, triggers could be more precisely classified into correct event types.

(2) On the TI stage, TLNN gives the best performance. By linking shortcut paths of all word candidates with the current character, the model could effectively exploit both character and word information, and then alleviates the issue of trigger-word mismatch.

| Model | ACE2005 | | KBP2017 | |
|---|---|---|---|---|
| | TI | TC | TI | TC |
| Word baseline | 61.13 | 57.81 | 56.68 | 50.98 |
| +char CNN | 61.11 | 57.52 | 57.14 | 51.66 |
| +char LSTM | 61.51 | 58.15 | 56.14 | 50.46 |
| Char baseline | 64.97 | 61.78 | 53.95 | 49.63 |
| +bigram | 66.89 | 62.95 | 57.01 | 52.71 |
| +softword | 67.79 | 64.11 | 60.68 | 54.57 |
| +softword+bigram | 68.22 | 64.23 | 60.42 | 54.85 |
| **TLNN** | **70.82** | **67.78** | **62.31** | **57.39** |

Table 3: F1-score of Word-based and Character-based baselines and TLNN on ACE2005 and KBP2017.

(3) On the TC stage, TLNN still maintain its advantages. The results indicate that the linguistic knowledge of HowNet and the unique structure to dynamically utilize sense-level information could enhance the performance on the TC stage. More located triggers could be classified into correct event types by considering the ambiguity of triggers.

### 3.3 Effect of Trigger-aware Feature Extractor

In this section, we design a set of experiments to explore the effect of the trigger-aware feature extractor. We implement strong character-based and word-based baselines by replacing the trigger-aware lattice LSTM with the standard Bi-LSTM.

For word-based baselines, the input is segmented into word sequences firstly. Furthermore, we implement extra CNN and LSTM to learn character-level features as additional modules. For character-based baselines, the basic units of the

| Model | ACE2005 | | KBP2017 | |
|---|---|---|---|---|
| | Match | Mismatch | Match | Mismatch |
| Char Baseline | 65.58 | 63.89 | 55.65 | 42.76 |
| Word Baseline | 66.30 | 2.78 | 64.43 | 17.63 |
| NPN | 65.94 | 55.56 | 56.47 | 41.53 |
| **TLNN** | **69.93** | **72.22** | **65.52** | **48.56** |

Table 4: Recall rates of two trigger-word match splits of two datasets on Trigger Identification task.

input sequence are characters. Then we enhance the character representation by adding external word-level features including bigram and softword (word in which the current character is located). Hence, both baselines could collectively utilize character and word information.

As shown in Table 3, experiments of two types of baselines and our model are conducted on ACE2005 and KBP2017. For the word baseline, although adding character-level features can improve the performance, the effects are relative limited. For the char baseline, it gains considerable improvements when word-level features are taken into account. The results of baselines indicate that integrating different level of information is an effective strategy to improve the performance of models. Compared with baselines, the TLNN achieves the best F1-score compared to all the baselines on both datasets, showing remarkable superiority and robustness. The results show that by dynamically combining multi-grained information, the trigger-aware feature extractor could effectively explore deeper semantic features than the feature-based strategies used in baselines.

### 3.4 Influence of Trigger Mismatch

In order to explore the influence of trigger mismatch problem, we split the test data of ACE2005 and KBP2017 into two types: match and mismatch. Table 1 shows the proportion of word-trigger match and mismatch on two datasets.

The recall of different methods of each split on Trigger Identification task is shown in Table 4. We can observe that:

(1) The result indicates that the word-trigger mismatch problem could severely impact the performance of the task. All approaches except ours give lower recall rates in the trigger-mismatch part than in the trigger-match part. In contrast, our model could robustly address the word-trigger mismatch problem, reaching the best results on both parts of the two datasets.

| Model | ACE2005 | | KBP2017 | |
|---|---|---|---|---|
| | **TI** | **TC** | **TI** | **TC** |
| NPN | 67.56 | 64.21 | 58.96 | 52.87 |
| TLNN | **70.82** | **67.78** | **62.31** | **57.39** |
| - w/o Sense info | 68.65 | 65.83 | 61.17 | 56.55 |

Table 5: F1-score of NPN, TLNN and TLNN - w/o Sense info on ACE2005 and KBP2017.

| Model | ACE2005 | | KBP2017 | |
|---|---|---|---|---|
| | Poly | Mono | Poly | Mono |
| NPN | 66.27 | 61.08 | 51.15 | 48.63 |
| TLNN | **69.11** | 62.56 | **58.61** | **56.56** |
| - w/o Sense info | 67.53 | **62.89** | 56.01 | 55.96 |

Table 6: F1-score of two splits of two datasets on Trigger Classification task. The splits are based on the polysemy of triggers. "Poly" and "Mono" correspond to polysemous and monosemous trigger splits.

(2) To a certain extent, the NPN model could alleviate the problem by utilizing hybrid representation learning and nugget generator in a fix-sized window. However, the mechanism is still not flexible and robust to integrate character and word information.

(3) The word-based baseline is most severely affected by the trigger-word mismatch problem. This phenomenon is explainable because if one trigger could not be segmented as a specific word in the preprocessing stage, it is impossible to be located correctly.

### 3.5 Influence of Trigger Polysemy

In this section, we mainly focus on the influence of polysemous triggers. We select NPN model for comparison. And we implement a version of TLNN without sense information, which is denoted as **TLNN - w/o Sense info** in Table 5 and Table 6.

Empirical results in Table 5 show the overall performance on ACE2005 and KBP2017. We can observe that the TLNN is weakened by removing sense information, which indicates the effectiveness of the usage of sense-level information. Even without sense information, our model could still outperform the NPN model on both two datasets.

To further explore and analyze the effect of word sense information, we split the KBP2017 dataset into two parts based on the polysemy of triggers and their contexts. The F1-score of each split is shown in Table 6, in which the TLNN yields the best results on both "Poly" parts. With-

| Sentence 1 | Word baseline | NPN | TLNN | Answer |
|---|---|---|---|---|
| 立刻/**抗**敌援友<br>**Resist** the enemies and aid the allies at once | (抗敌援友,Attack) | (刻抗,Attack) | (抗,Attack) | (抗,Attack) |
| **Sentence 2** | **NPN** | **TLNN - Sense info** | **TLNN** | **Answer** |
| **送**/他/一笔/赴/欧洲/的/旅费<br>**Send** him money for European tourism | (送,TransferPerson) | (送,TransferPerson) | (送,TransferMoney) | (送,TransferMoney) |

Table 7: Two model prediction examples. The first sentence is an example of trigger-word mismatch, while the second one is about polysemous triggers. For each prediction result, (A,B) indicates that A is a trigger with event type B.

out sense information, TLNN - w/o sense info could give comparable F1-scores with TLNN on the "Mono" parts. The results indicate that the trigger-aware feature extractor could dynamically learn all the senses of characters and words, gaining significant improvements under the condition of polysemy.

### 3.6 Case Study

Table 7 shows two examples comparing the TLNN model with other ED methods. The former example is about trigger-word mismatch, in which the correct trigger "抗"(resist) is part of the idiom word "抗敌援友" (resist the enemies and aid the allies). In this case, the word baseline gives the whole word "抗敌援友" as prediction because it is impossible for word-based methods to detect part-of-word triggers. Additionally, the NPN model recognizes a non-existent word "刻抗". The reason is that the NPN enumerates the combinations of all characters within a window as trigger candidates, which is likely to generate invalid words. In contrast, our model detects the event trigger "抗" accurately.

In the latter example, the trigger "送"(send) is a polysemous word with two different meanings: "送行"(see him off) and "送钱"(give him money). Without considering multiple word senses of polysemes, the NPN and TLNN (w/o Sense info) classify trigger "送" into wrong event type *TransferPerson*. On the contrary, the TLNN can dynamically select word sense for polysemous triggers by utilizing context information. Thus the correct event type *TransferMoney* is predicted.

## 4 Related Work

Event Detection (ED) is a crucial subtask in Event Extraction task. Feature-based methods (Ahn, 2006; Ji and Grishman, 2008; Liao and Grishman, 2010; Huang and Riloff, 2012; Patwardhan and Riloff, 2009; McClosky et al., 2011) were widely used in the ED task, but these traditional methods are heavily rely on the manual features, limiting the scalability and robustness.

Recent developments in deep learning have led to a renewed interest in neural event detection. Neural networks can automatically learn features of the input sequence and conduct token-level classification. CNN-based models are the seminal neural network models in ED (Nguyen and Grishman, 2015; Chen et al., 2015; Nguyen and Grishman, 2016). However, these models can only capture the local context features in a fixed size window. Some approaches design comprehensive models to explore the interdependency among trigger words (Chen et al., 2018; Feng et al., 2018). To further improve the ED task, some joint models are designed (Nguyen et al., 2016; Lu and Nguyen, 2018; Yang and Mitchell, 2016). These methods have achieved great success in English datasets.

However, in languages without delimiters, such as Chinese, the mismatch of word-trigger become significantly severe. Some feature-based methods are proposed to solve the problem (Chen and Ji, 2009; Qin et al., 2010; Li and Zhou, 2012), but they heavily rely on the hand-crafted features. Lin et al. (2018) proposes NPN, a neural network based method to address the issue. However, the mechanism of NPNs limits the scope of trigger candidates within a fix-sized window, which will cause two problems in the progress. First, the NPNs still cannot take all the possible trigger candidates into account, leading to meaningless computation. Furthermore, the overlap of triggers is serious in NPNs. Lattice-based models were used in other fields to combine character and word information (Li et al., 2019; Zhang and Yang, 2018; Yang et al., 2018). Mainstream methods also suffer from the problem of trigger polysemy. Lu and Nguyen (2018) proposes a multi-task learning model which uses word sense disambiguation to alleviate the effect of the trigger polysemy

problem. But in this work, word disambiguation datasets are necessary. In contrast, our model can solve both word-trigger mismatch and trigger polysemy problems at the same time.

## 5 Conclusion and Future Work

We propose a novel framework TLNN for event detection, which can simultaneously address the problems of trigger-word mismatch and polysemous triggers. With the hierarchical representation learning and the trigger-aware feature extractor, TLNN efficaciously exploits multi-grained information and learn deep semantic features. Sets of experiments on two real-world datasets show that TLNN could efficiently address the two issues and yield better empirical results than a variety of neural network models.

In future work, we will conduct experiments on more languages with and without explicit word delimiters. In addition, we will try developing a dynamic mechanism to selectively consider the sense-level information rather than take all the senses of characters and words into account.

## 6 Acknowledgement

This work is supported by National Natural Science Foundation of China (Grant No. 61773229, 61572273, 61661146007), National Key Research and Development Program of China (No. 2018YFB1004503), Basic Scientific Research Program of Shenzhen City (Grant No. JCYJ20160331184440545), and Overseas Cooperation Research Fund of Graduate School at Shenzhen, Tsinghua Univeristy (Grant No. HW2018002), Shenzhen Giiso Information Technology Co. Ltd. Finally, we would like to thank the anonymous reviewers for their helpful feedback and suggestions.

## References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, pages 1–8.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 167–176.

Yubo Chen, Hang Yang, Kang Liu, Jun Zhao, and Yantao Jia. 2018. Collective event detection via a hierarchical and bias tagging networks with gated multi-level attention mechanisms. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1267–1276.

Zheng Chen and Heng Ji. 2009. Language specific issue and feature exploration in chinese event extraction. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 209–212.

Zhendong Dong and Qiang Dong. 2003. Hownet-a hybrid language and knowledge resource. In *Proceedings of NLP-KE*.

Xiaocheng Feng, Bing Qin, and Ting Liu. 2018. A language-independent neural network for event detection. *Science China Information Sciences*, 61(9):092106.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Ruihong Huang and Ellen Riloff. 2012. Modeling textual cohesion for event extraction. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. *Proceedings of ACL-08: HLT*, pages 254–262.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Peifeng Li and Guodong Zhou. 2012. Employing morphological structures and sememes for chinese event extraction. *Proceedings of COLING 2012*, pages 1619–1634.

Ziran Li, Ning Ding, Zhiyuan Liu, Haitao Zheng, and Ying Shen. 2019. Chinese relation extraction with multi-grained information and external linguistic knowledge. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 4377–4386.

Shasha Liao and Ralph Grishman. 2010. Using document level cross-event inference to improve event extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 789–797. Association for Computational Linguistics.

Hongyu Lin, Yaojie Lu, Xianpei Han, and Le Sun. 2018. Nugget proposal networks for chinese event detection. *arXiv preprint arXiv:1805.00249*.

Weiyi Lu and Thien Huu Nguyen. 2018. Similar but not the same: Word sense disambiguation improves event detection via neural representation matching.

In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4822–4828.

David McClosky, Mihai Surdeanu, and Christopher D Manning. 2011. Event extraction as dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1626–1635. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 300–309.

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 365–371.

Thien Huu Nguyen and Ralph Grishman. 2016. Modeling skip-grams for event detection with convolutional neural networks. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 886–891, Austin, Texas. Association for Computational Linguistics.

Yilin Niu, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2017. Improved word representation learning with sememes. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2049–2058, Vancouver, Canada. Association for Computational Linguistics.

Siddharth Patwardhan and Ellen Riloff. 2009. A unified model of phrasal and sentential evidence for information extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 151–160. Association for Computational Linguistics.

Bing Qin, Yanyan Zhao, Xiao Ding, Ting Liu, and Guofu Zhai. 2010. Event type recognition based on trigger expansion. *Tsinghua Science and Technology*, 15(3):251–258.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Andrew Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*, 13(2):260–269.

Bishan Yang and Tom Mitchell. 2016. Joint extraction of events and entities within a document context. *arXiv preprint arXiv:1609.03632*.

Jie Yang, Yue Zhang, and Shuailong Liang. 2018. Subword encoding in lattice lstm for chinese word segmentation. *arXiv preprint arXiv:1810.12594*.

Ying Zeng, Honghui Yang, Yansong Feng, Zheng Wang, and Dongyan Zhao. 2016. A convolution bilstm neural network model for chinese event extraction. In *Natural Language Understanding and Intelligent Applications*, pages 275–287. Springer.

Yue Zhang and Jie Yang. 2018. Chinese ner using lattice lstm. *arXiv preprint arXiv:1805.02023*.

# A Boundary-aware Neural Model for Nested Named Entity Recognition

**Changmeng Zheng**[1], **Yi Cai**[1]*, **Jingyun Xu**[1], **Ho-fung Leung**[2] and **Guandong Xu**[3]
[1]School of Software Engineering, South China University of Technology, Guangzhou, China
[2]The Chinese University of Hong Kong, Hong Kong SAR, China
[3]Advanced Analytics Institute, University of Technology Sydney, Australia
sethecharm@mail.scut.edu.cn, ycai@scut.edu.cn

## Abstract

In natural language processing, it is common that many entities contain other entities inside them. Most existing works on named entity recognition (NER) only deal with flat entities but ignore nested ones. We propose a boundary-aware neural model for nested NER which leverages entity boundaries to predict entity categorical labels. Our model can locate entities precisely by detecting boundaries using sequence labeling models. Based on the detected boundaries, our model utilizes the boundary-relevant regions to predict entity categorical labels, which can decrease computation cost and relieve error propagation problem in layered sequence labeling model. We introduce multitask learning to capture the dependencies of entity boundaries and their categorical labels, which helps to improve the performance of identifying entities. We conduct our experiments on nested NER datasets and the experimental results demonstrate that our model outperforms other state-of-the-art methods.

## 1 Introduction

Named entity recognition (NER) is a task that seeks to locate and classify named entities in unstructured texts into pre-defined categories such as person names, locations or medical codes. NER is generally treated as single-layer sequence labeling problem (Lafferty et al., 2001; Lample et al., 2016) where each token is tagged with one label. The label is composed by an entity boundary label and a categorical label. For example, a token can be tagged with $B\text{-}PER$, where $B$ indicates the boundary of an entity and $PER$ indicates the corresponding entity categorical label. However, when entities are nested within one another, single-layer sequence labeling models can not extract both entities simultaneously. A token contained inside many entities has more than one categorical label. Consider an example in Figure 1 from GENIA corpus (Kim et al., 2003), "Human TR Beta 1" is an $protein$ and it is also a part of a $DNA$ "Human TR Beta 1 mRNA". Both entities contain the same token "Human". Thus the token should have two different categorical labels. In that case, assigning a single categorical label for "Human" is improper.

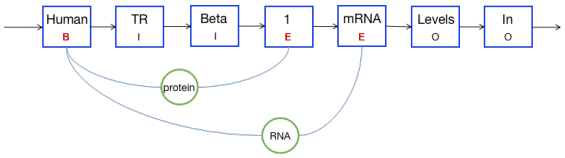


Figure 1: An example of nested entities and their boundary labels. "B" and "E" indicate the beginning and end of an entity. They are the boundary labels. "I" and "O" denote tokens inside and outside entities, respectively. $protein$ and $RNA$ are categories of entities.

Traditional methods coping with nested entities rely on hand-craft features (Shen et al., 2003; Alex et al., 2007) and suffer from heavy feature engineering. Recent studies tackle the nested NER using neural models without relying on linguistics features or external knowledge resources. Ju et al. (2018) propose a layered sequence labeling model and Sohrab and Miwa (2018) propose a exhaustive region classification model.

- Layered sequence labeling model will first extract the inner entities (contained by other entities) and feed them into the next layer to extract outer entities. Thus, this model suffers from error propagation. When the previous layer extracts wrong entities, the performance of next layer will be affected. Moreover, when an outer entity is extracted first, the inner one will not be detected.

*Corresponding author



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 357–366, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

- Exhaustive region classification model enumerates all possible regions or spans in sentences to predict entities in a single layer. One issue of their method is the explicit boundary information is ignored, leading to extraction of some non-entities. **We consider an example**. In a sequence of tokens in GENIA dataset, "novel TH protein" is an entity and "a novel TH protein" is not an entity. However, since they share many tokens, the merged region representations of them are similar to each other. "novel" and "protein" are the boundary of the entity. Without the boundary information, both candidate regions are extracted as the entities.

Despite their shortcomings, layered sequence labeling model and exhaustive region classification model are complementary to each other. Therefore, we can combine them to improve the performance of nested NER. We leverage the sequence labeling model to consider the boundary information into locating entities. In the example mentioned above, "novel" is the boundary of the entity "novel TH protein", while "a" is a general token whose representation is different from "novel". With the guidance of boundary information, the model can detect "novel" as a boundary of the entity rather than token "a". We also utilize the region classification model to predict entities without considering the dependencies of inner and outer entities. In such case, Our model will not suffer error propagation problem.

In this paper, we propose a boundary-aware neural model that makes the fusion of sequence labeling model and region classification model. We apply a single-layer sequence labeling model to identify entity boundaries because the tokens in nested entities can share the same boundary labels. For example, as shown in Figure 1, "Human" can be tagged with the label $B$ although it is the beginning of two different entities. Based on the detected entity boundaries, we predict entity categorical labels by classifying boundary-relevant regions. As shown in Figure 1, we match each token with label $B$ to tokens with label $E$. The regions between them are considered as candidate entities. The representation of candidate entities will be utilized to classify categorical labels.

Our model is advanced than exhaustive region classification model in two ways: (1) we leverage the explicit boundary information to guide the model to locate and classify entities precisely. Exhaustive region classification model classifies entity regions individually, however, our model can consider the context information of boundary tokens with a sequence labeling model. That facilitates the detection of boundaries. (2) Our model only classifies the boundary-relevance regions which are much fewer than all possible regions. That decreases the time cost. Our model is advanced than layered sequence labeling model because we extract entities without distinguishing inner and outer entities.

Multitask learning is considered good at optimising the overall goal via alternatively tuning 2+ objectives, which are reinforced each other (Ruder, 2017). Considering our boundary detection module and entity categorical label prediction module share the same entity boundaries, we apply a multitask loss for training the two tasks simultaneously. The shared features of two modules are extracted by a bidirectional long short-term memory (LSTM) layer. Extensive experiments show the framework of multitask learning improves final performance in a large margin.

In summary, we make the following major contributions in this paper:

- We propose a boundary-aware neural model which leverages entity boundaries to predict categorical labels. Our model can locate entities precisely by detecting boundaries using sequence labeling models. Based on the detected boundaries, our model utilizes boundary-relevant regions to predict entity categorical labels, which can decrease computation cost and relieve error propagation problem.

- We introduce the multitask learning to capture the dependencies of entity boundaries and their categorical labels, which helps to improve the performance of identifying entities.

- We conduct our experiments on public nested NER datasets. The experimental results demonstrate our model outperforms previous state-of-the-art methods and our model is much faster in inference speed.

## 2 Related Work

NER has drawn the attention of NLP researchers because several downstream tasks such as entity

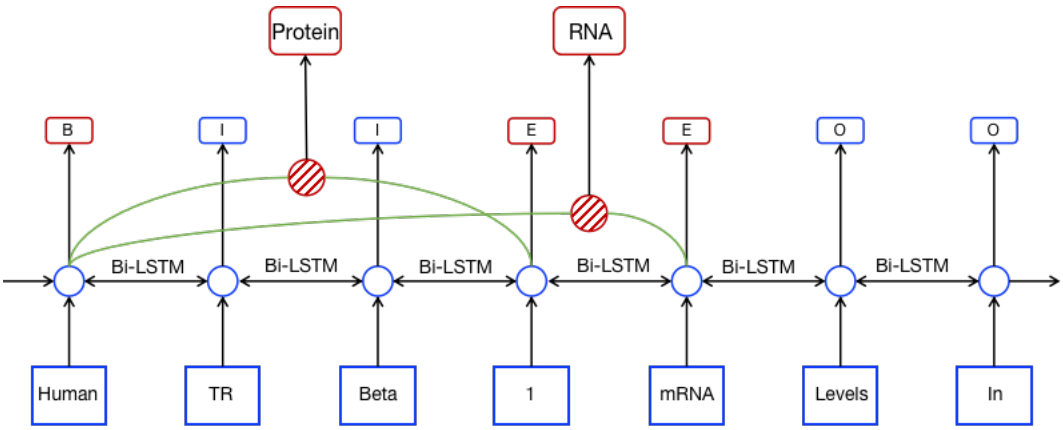


Figure 2: The Architecture of our boundary-aware model. The representation of each token in sentence "Human TR Beta 1 mRNA Levels in.." is feed into a shared bidirectional LSTM layer. We leverage the outputs of Bi-LSTM to detect entity boundaries and their categorical labels. The red circle indicates entity region representations between entity boundaries.

linking (Gupta et al., 2017), relation extraction (Mintz et al., 2009; Liu et al., 2017), co-reference resolution (Chang et al., 2013) and conversation system (Ren et al., 2019) rely on it. Several methods have been proposed on flat named entity recognition (Lample et al., 2016; Ma and Hovy, 2016; Strubell et al., 2017) while few of them address nested entities. Early work on nested entities rely on hand-craft features or rule-based post-processing (Zhang et al., 2004; Zhou et al., 2004; Zhou, 2006). They detect the innermost flat entities with a Hidden Markov Model and then use rule-based post-processing to extract the outer entities.

While most work concerns about named entities, Lu and Roth (2015) present a novel hypergraph-based method to tackle the problem of entity mention detection. One issue of their method is the spurious structure of hyper-graphs. Muis and Lu (2017) improve the method of Lu and Roth (2015) by incorporating mention separators along with features.

Recent studies reveal that stacking sequence model like conditional random filed(CRF) layer can extract entities from inner to outer. Alex et al. (2007) propose several CRF-based methods for the GENIA dataset. However, their approach can not recognize nested entities of the same type. Finkel and Manning (2009) present a chart-based parsing method where each named entity is a constituent in the parsing tree. However, their method is not scalable to larger corpus with a cubic time complexity. Ju et al. (2018) dynamically stack flat NER layers to extract nested entities, each flat layer is based on a Bi-LSTM layer and then a cascaded CRF layer. Their model suffers error propagation from layer to layer, an inner entity can not be detected when a outer entity is identified first.

It is difficult for sequence model, like CRF, to extract nested entities where a token can be included in several entities. Wang et al. (2018) present a transition-based model for nested mention detection using a forest representation. One drawback of their model is the greedy training and decoding. Sohrab and Miwa (2018) consider all possible regions in a sentence and classify them into their entity type or non-entity. However, their exhaustive method considers too many irrelevant regions(non-entity regions) into detecting entity types and the regions are classified individually, without considering the contextual information. Our model focuses on the boundary-relevant regions which is much fewer and the explicit leveraging of boundary information helps to locate entities more precisely.

## 3 Method

In this paper, we propose a boundary-aware neural model which considers the boundary information into locating and classifying entities. The architecture is illustrated in Figure 2.

Our model is built upon a shared bidirectional LSTM layer. It uses the outputs of LSTM layer to detect entity boundaries and predict categorical labels. We extract entity boundaries as paired tokens with label $B$ and label $E$, "B" indicates the beginning of an entity and "E" means the end of an entity. We match every detected token with label

$B$ and its corresponding token with label $E$, the regions between them are identified as candidate entities. We represent entities using the corresponding region outputs of shared LSTM and classify them into categorical labels.

The boundary detection module and entity categorical label prediction module are training simultaneously with a multitask loss function, which can capture the underlying dependencies of entity boundaries and their categorical labels. We will describe each part of our model in detail.

### 3.1 Token Representation

We represent each token in the sentence following the success of Ma and Hovy (2016) and Lample et al. (2016) that leverages character embedding for the flat NER task.

For a given sentence consisting of $n$ tokens ($t_1$,$t_2$,....$t_n$), we represent the word embedding of $i$-th token $t_i$ as equation(1):

$$\mathbf{x}_i^w = \mathbf{e}^w(t_i) \tag{1}$$

where $\mathbf{e}^w$ denotes a word embedding lookup table. We use pre-trained word embedding (Chiu et al., 2016) to initialize it.

We capture the orthographic and morphological features of the word by integrating character representations. Denoting the representation of characters within $t_i$ as $\mathbf{x}_i^c$, The embedding of each character within token $t_i$ is denoted as $\mathbf{e}^c(c_j)$. $\mathbf{e}^c$ is the character embedding lookup which is initialized randomly. Then we feed them into a bidirectional LSTM layer to learn hidden states. The forward and backward outputs are concatenated to construct character representations:

$$\mathbf{x}_i^c = [\overrightarrow{\mathbf{h}_i^c}; \overleftarrow{\mathbf{h}_i^c}] \tag{2}$$

where $\overleftarrow{\mathbf{h}_i^c}$ and $\overrightarrow{\mathbf{h}_i^c}$ denote the forward and backward outputs of bidirectional LSTM.

The final token representation is obtained as equation (3), where [;] denotes concatenation.

$$\mathbf{x}_i^t = [\mathbf{x}_i^w; \mathbf{x}_i^c] \tag{3}$$

### 3.2 Shared Feature Extractor

As shown in Figure 2, we apply the hard parameter sharing mechanism (Ruder, 2017) for multitask training using bidirectional LSTM as shared feature extractor. Hard parameter sharing greatly reduces the risk of overfitting (Baxter, 1997) and increases the correlation of our boundary detection module and categorical label prediction module. Specifically, the hidden state of bidirectional LSTM can be expressed as following:

$$\overrightarrow{\mathbf{h}}_i^t = \overrightarrow{\mathrm{LSTM}}(\mathbf{x}_i^t, \overrightarrow{\mathbf{h}}_{i-1}^t) \tag{4}$$

$$\overleftarrow{\mathbf{h}}_i^t = \overleftarrow{\mathrm{LSTM}}(\mathbf{x}_i^t, \overleftarrow{\mathbf{h}}_{i-1}^t) \tag{5}$$

$$\mathbf{h}_i^t = [\overrightarrow{\mathbf{h}}_i^t; \overleftarrow{\mathbf{h}}_i^t] \tag{6}$$

where $\mathbf{x}_i^t$ is the token representation which is mentioned in section 3.1. We feed $\mathbf{x}_i^t$ into a Dropout layer to prevent overfitting. $\overrightarrow{\mathbf{h}}_i^t$ and $\overleftarrow{\mathbf{h}}_i^t$ denote the $i$-th forward and backward hidden state of Bi-LSTM layer. Formally, we extract the shared features of each token in a sentence as $\mathbf{h}_i^t$.

### 3.3 Entity Boundary Detection

Previous works (Lample et al., 2016; Ma and Hovy, 2016) on flat NER (non-nested named entities recognition) predict entity boundaries and categorical labels jointly. However, when entities are nested in other entities, one individual token can be included in many different entities. This means assigning one single categorical label for each token is inappropriate.

We divide nested NER into two subtasks: entity boundary detection and categorical label prediction tasks. Unlike assigning an entity categorical label for each token, we predict the boundary labels first. Formally, given a sentence ($t_1$,$t_2$,...$t_n$), and one entity in the sentence. we represent the entity as $R(i, j)$, which denotes the entity is composed by a continuous token sequence ($t_i$,$t_{i+1}$,...$t_j$). Specially, we tag the boundary token $t_i$ as "B" and $t_j$ as "E". The tokens inside entities are assigned with label "I" and non-entity tokens are assigned with "O" labels.

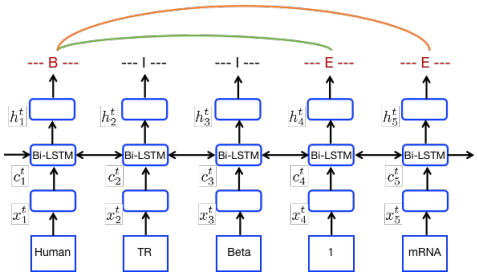


Figure 3: The architecture of entity boundary detection module. We feed the representation of each token in the sentence into a bidirectional LSTM layer, the outputs of LSTM layer are utilized to predict boundary labels.

We detect entity boundaries as shown in Fig-

ure 3. For each token $t_i$ in a sentence, we predict a boundary label by feeding its corresponding shared feature representation $\mathbf{h}_i^t$ (described in section 3.2) into a ReLU activation function and a softmax classifier:

$$\mathbf{o}_i^t = \mathbf{U}\mathbf{h}_i^t + \mathbf{b} \quad (7)$$

$$\mathbf{d}_i^t = \text{softmax}(\mathbf{o}_i^t) \quad (8)$$

where $\mathbf{U}$ and $\mathbf{b}$ are trainable parameters. We compute the KL-divergence multi-label loss between the true distribution $\hat{\mathbf{d}}_i^t$ and the predicted distribution $\mathbf{d}_i^t$ as equation (9):

$$L_{bcls} = -\sum(\hat{\mathbf{d}}_i^t)\log(\mathbf{d}_i^t) \quad (9)$$

Conditional random field (CRF) (Lafferty et al., 2001) is considered good at modeling sequence label dependencies (e.g., label $I$ must be after $B$). We make a comparison of choosing softmax or CRF as output layer because our sequence labels are different from flat NER models.

### 3.4 Entity Categorical Label Prediction

Given an input sentence sequence X = ($x_1$,$x_2$, ... $x_n$), and a corresponding boundary label sequence L = ($l_1$,$l_2$, ... $l_n$), we match each token with label $B$ to the token with label $E$ to construct candidate entity regions. Especially, considering there are entities containing one single token, we match tokens with label $B$ to themselves firstly. The representation of entity $R(i,j)$ is obtained as following:

$$\mathrm{R}_{i,j} = \left[\frac{1}{j-i+1}\sum_{k=i}^{j}\mathbf{h}_k^t\right] \quad (10)$$

where $\mathbf{h}_k^t$ denotes the outputs of the shared bidirectional LSTM layer for $k$-th token in sentence. We simply average the representations for each token within boundary regions. The final representation of entities will be sent into a ReLU activation function and the softmax layer to predict entity categorical labels. We compute the loss of categorical label prediction in equation (11-12):

$$\mathbf{d}_{i,j}^e = \text{softmax}(\mathbf{U}_{i,j}^e\mathrm{R}_{i,j} + \mathbf{b}_{i,j}^e) \quad (11)$$

$$L_{ecls} = -\sum(\hat{\mathbf{d}}_{i,j}^e)\log(\mathbf{d}_{i,j}^e) \quad (12)$$

where $\mathbf{U}_{i,j}^e$ and $\mathbf{b}_{i,j}^e$ are trainable parameters. $\hat{\mathbf{d}}_{i,j}^e$ and $\mathbf{d}_{i,j}^e$ denote the true distribution and predicted distribution of entity categorical labels, respectively.

### 3.5 Multitask Training

In our model, it is inconvenient and inefficient for the reason that we predict entity categorical labels after all boundary-relative regions have been detected. Considering our boundary detection module and entity categorical label prediction module share the same entity boundaries, we apply a multitask loss for training the two tasks simultaneously.

During training phase, we feed the ground-truth boundary labels into entity categorical label prediction module so that the classifier will be trained without affection from error boundary detection. As for testing phase, the outputs of boundary detection will be collected. The detected boundaries will indicate which entity regions should be considered into predicting categorical labels. The multitask loss function is defined as follows:

$$L_{multi} = \alpha\sum L_{bcls} + (1-\alpha)\sum L_{ecls} \quad (13)$$

where $L_{bcls}$ and $L_{ecls}$ denote the categorical cross-entropy loss for boundary detection module and entity categorical label prediction module, respectively. $\alpha$ is a hyper-parameter which is assigned to control the degree of importance for each task.

## 4 Experimental Settings

### 4.1 Dataset

To provide empirical evidence for effectiveness of the proposed model, we employ our experiments on three nested NER datasets: GENIA (Kim et al., 2003), JNLPBA (Kim et al., 2004) and GermEval 2014 (Benikova et al., 2014).

**GENIA** dataset is constructed based on the GENIA v3.0.2 corpus. We preprocess the dataset following the same settings of (Finkel and Manning, 2009) and (Lu and Roth, 2015). The dataset is split into 8.1:0.9:1 for training, development and testing. The statistics of GENIA dataset is shown as Table 1.

| Item | Train | Dev | Test | Overall | Nested |
|---|---|---|---|---|---|
| Document | 1599 | 189 | 212 | 2000 | - |
| Sentences | 15023 | 1669 | 1854 | 18546 | - |
| Percentage | 81% | 9% | 10% | 100% | - |
| DNA | 7650 | 1026 | 1257 | 9933 | 1744 |
| RNA | 692 | 132 | 109 | 933 | 407 |
| Protein | 28728 | 2303 | 3066 | 34097 | 1902 |
| Cell Line | 3027 | 325 | 438 | 3790 | 347 |
| Cell Type | 5832 | 551 | 604 | 6987 | 389 |
| Overall | 45929 | 4337 | 5474 | 55740 | 4789 |

Table 1: Statistics of GENIA dataset

**JNLPBA** dataset is originally from GENIA corpus. It contains a training and testing datasets.

However, only the flat and top-most entities are preserved. We collapse the sub-categories into 5 categories following the same settings as GENIA dataset.

**GermEval 2014** dataset contains German nested named entities. The dataset covers over 31,000 sentences corresponding to over 590,000 tokens.

### 4.2 Baseline Methods

We compare our model with several state-of-the-art models on GENIA dataset. These methods can be divided into three groups:

Finkel and Manning (2009) and Ju et al. (2018) propose CRF-based sequence labeling approaches for nested named entity recognition. Finkel and Manning (2009) leverage entity-level features while Ju et al. (2018) propose neural-based method. We rerun the codes of Ju et al. (2018) because they have not shared their pre-processed dataset.

Sohrab and Miwa (2018) propose an exhaustive region classification model for nested NER. We re-implement their method according to their paper because they have not shared the codes.

Lu and Roth (2015) and Muis and Lu (2017) build hyper-graphs to represent both the nested entities and their mentions. Muis and Lu (2017) improve the method of Lu and Roth (2015).

### 4.3 Parameter Settings

Our model is implemented by PyTorch framework[1] [2]. We use Adam optimizer for training our model. We initialize word vectors with a 200-dimension pre-trained word embedding the same as Ju et al. (2018) and Sohrab and Miwa (2018) while the char embedding is set to 50-dimension and initialized randomly. The learning rate is set to 0.005. We set a 0.5 dropout rate for the Dropout layer employed after token-level LSTM during training phase. The output dimension of our shared bidirectional LSTM is 200. The coefficient $\alpha$ of multitask loss is tuned during development process. All of our experiments are performed on the same machine (NVIDIA 1080ti GPU and Intel i7-8700 CPU).

[1] https://pytorch.org/

[2] Code is available at https://github.com/thecharm/boundary-aware-nested-ner

### 4.4 Evaluation Metrics

We use a strict evaluation metrics that an entity is confirmed correct when the entity boundary and the entity categorical label are correct simultaneously. We employ precision, recall and F-score to evaluate the performance.

## 5 Results and Discussion

### 5.1 Overall Evaluation

We conduct our experiments on GENIA test dataset for nested named entity recognition. Table 2 shows our method outperforms the compared methods both in recall and F-score metrics. The CRF-based model is considered as more efficient in sequence labeling task, we compare the utilization of softmax and CRF as output layer of boundary detection module. The results show they gain comparable scores in precision, recall and F-score. However, the CRF-based model is time-consuming, about 3-5 times slower than the softmax-based model in inference speed.

| Model | P(%) | R(%) | F(%) |
|---|---|---|---|
| Finkel and Manning (2009)[3] | 75.4 | 65.9 | 70.3 |
| Lu and Roth (2015)[3] | 72.5 | 65.2 | 68.7 |
| Muis and Lu (2017)[3] | 75.4 | 66.8 | 70.8 |
| Sohrab and Miwa (2018) | 73.3 | 68.3 | 70.7 |
| Ju et al. (2018) | **76.1** | 66.8 | 71.1 |
| Our model(softmax) | 75.9 | **73.6** | **74.7** |
| Our model(CRF) | 74.6 | 73.2 | 73.9 |

Table 2: Performance on GENIA test set. Our models with softmax and CRF outperform other state-of-the-art methods.

Our model achieves a recall value of 73.6% and outperforms compared methods in Recall value with a large margin. We think that our model extract entities with a more accurate boundaries comparing to other methods. We evaluate it in experiments on boundary detection module.

| Model | P(%) | R(%) | F(%) |
|---|---|---|---|
| Sohrab and Miwa (2018) | **75.0** | 60.8 | 67.2 |
| Ju et al. (2018) | 72.9 | 61.5 | 66.7 |
| Our model | 74.5 | **69.1** | **71.7** |

Table 3: Performance on GermEval 2014 test set. Our model outperforms two state-of-the-art methods in nested NER.

The GermEval 2014 dataset from KONVENS 2014 shared task is a German NER dataset. It

[3] The results are taken from their papers.

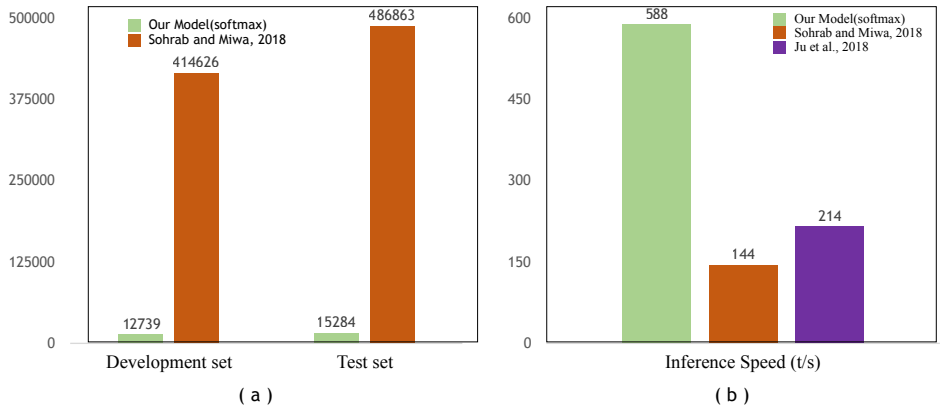


Figure 4: (a): The number of candidate entity regions in our model with softmax and the approach of Sohrab and Miwa (2018) when evaluating on GENIA test and development set; (b): The inference speed of our model and compared models on GENIA test set. t/s indicates token per second.

contains few nested entities. Previous works in this dataset ignore nested entities or extract inner and outer entities in two independent models. Our method can extract nested entities in an end-to-end way. We compare our method with two state-of-the-art approaches in Table 3. Our method outperforms their approaches both in Recall and F-score metrics.

Table 4 describes the performances of our model on the five categories on the test dataset. Our model outperforms the model described in Ju et al. (2018) and Sohrab and Miwa (2018) with F-score value on all categories.

| Category | P(%) | R(%) | F(%) | Ju. F(%) | Soh. F(%) |
|---|---|---|---|---|---|
| DNA | 73.6 | 67.8 | **70.6** | 70.1 | 67.8 |
| RNA | 82.2 | 80.7 | **81.5** | 80.8 | 75.9 |
| protein | 76.7 | 76.0 | **76.4** | 72.7 | 72.9 |
| cell line | 77.8 | 65.8 | **71.3** | 66.9 | 63.6 |
| cell type | 73.9 | 71.2 | **72.5** | 71.3 | 69.8 |
| overall | 75.8 | 73.6 | **74.7** | 71.1 | 70.7 |

Table 4: Our results on five categories compared to Ju et al. (2018) and Sohrab and Miwa (2018) on GENIA test set.

## 5.2 Performance of Boundary Detection

We conduct experiments on boundary detection to illustrate that our model extract entity boundaries more precisely comparing to Sohrab and Miwa (2018) and Ju et al. (2018). Table 5 shows the results of boundary detection on GENIA test dataset. Our model locates entities more accurately with a higher recall value (76.9%) than the comparing methods. It gives a reason why our model outperforms other state-of-the-art methods in recall value. We exploit boundary information explicitly and consider the dependencies of boundaries and entity categorical labels with a multitask loss. While in the method of Sohrab and Miwa (2018), candidate entity regions are classified individually.

| Model | Boundary Detection | | |
|---|---|---|---|
| | P(%) | R(%) | F(%) |
| Sohrab and Miwa (2018) | 76.6 | 69.2 | 72.7 |
| Ju et al. (2018) | 79.9 | 67.08 | 73.4 |
| Our model(softmax) | 79.7 | **76.9** | **78.3** |

Table 5: Performance of Boundary Detection on GENIA test set.

Table 6 describes the performance of our model in detecting boundary labels for each token in sentences. The results are based on the shared bidirectional LSTM and a softmax classifier. Our model extracts entity boundaries with a relatively high performance. This facilitates the prediction of entity categorical labels because the candidate entity regions are more likely to be true entities.

| Boundary Label | P(%) | R(%) | F(%) |
|---|---|---|---|
| O (non-entity) | 99.3 | 99.0 | 99.2 |
| B (beginning) | 84.4 | 84.3 | 84.3 |
| E (end) | 86.0 | 87.2 | 86.6 |
| I (inner-entity) | 82.8 | 88.6 | 85.6 |

Table 6: Performance of Boundary Label Prediction with softmax classifier on GENIA test set.

## 5.3 Inference Time

Figure 4(a) shows the number of candidate entity regions in our model with softmax and the approach of Sohrab and Miwa (2018). Comparing to classifying all possible regions in sentences, our

model only concerns about boundary-relevant regions which is much fewer. We compare the inference speed of our model and the approaches of Sohrab and Miwa (2018) and Ju et al. (2018) in Figure 4(b). Our model is about 4 times faster than Sohrab and Miwa (2018) and about 3 times faster than Ju et al. (2018). The cascaded CRF layers of Ju et al. (2018) are the limitation in inference speed.

### 5.4 Performance of Multitask Learning

Table 7 shows the performance of our pipeline model and multitask model on GENIA development set and test set. For pipeline model, we train the boundary detection module and entity categorical label prediction module separately. Our multitask model has a higher F value both in development set and test set.

| Model | Development Set | | | Test Set | | |
|---|---|---|---|---|---|---|
| | P(%) | R(%) | F(%) | P(%) | R(%) | F(%) |
| Pipeline | 74.5 | 74.8 | 74.6 | 75.4 | 72.2 | 73.8 |
| Multitask | 74.5 | 75.6 | **75.0** | 75.9 | 73.4 | **74.7** |

Table 7: Performance Comparison of our pipeline model and multitask model on GENIA development set and test set.

Multitask learning can capture the underlying dependencies of boundaries and entity categorical labels. It helps the model focus its attention on those features that actually matter (Ruder, 2017). In pipeline model, entity categorical prediction module will not share information with boundary detection module because they are trained separately. However, entity categorical prediction module and boundary detection module share the same entity boundaries. We assign a shared feature extractor (the bidirectional LSTM layer) to extract the features utilized in both entity categorical prediction and boundary detection. The results have demonstrated that the framework of multitask learning improves final performance.

### 5.5 Ablation Study and Flat NER

We conduct ablation experiments on GENIA development set to evaluate the contributions of neural components including dropout layer, pretrained word embedding and the character-level LSTM. The results are listed in Table 8. All these components contribute to the effectiveness of our model. Dropout layer contributes significantly for both precision and recall values.

| Setting | P(%) | R(%) | F(%) |
|---|---|---|---|
| Our Model(softmax) | 74.5 | 75.6 | **75.0** |
| without Dropout | 72.6 | 73.1 | 72.9 |
| without Pre-trained | 73.8 | **75.7** | 74.7 |
| without Char repr. | **75.3** | 73.9 | 74.6 |

Table 8: Results of Ablation Tests on GENIA development set.

To prove our model can work on nested NER and also flat NER task, we perform experiments on the JNLPBA dataset. We achieve 73.6 in term of F-score which is comparable with the state-of-the-art result of Gridach (2017).

## 6 Case Study

Table 9 shows a case study comparing our model with exhaustive model (Sohrab and Miwa, 2018) and Layered model (Ju et al., 2018). In the example, "human TATA binding factor" is an entity nested in entity "transcriptionally active human TATA binding factor". Our model with multitask learning extracts both entities with exact boundaries and entity categorical labels. Exhaustive model gets the error boundaries and misses the token "human" in entities. Comparing to layered model only detecting an outer entity, our model extract both inner and outer entities. It demonstrates that our combination of sequence labeling models and region classification models can locate entities precisely and extract both inner and outer entities.

| | |
|---|---|
| Sentence | Cloning of a transcriptionally active human TATA binding factor. |
| Gold Label | protein: {human TATA binding factor; transcriptionally active human TATA binding factor} |
| Exhaustive model | protein: {TATA binding factor; transcriptionally active human TATA binding factor} |
| Layered model | protein: {transcriptionally active human TATA binding factor} |
| Our model(pipeline) | protein: {human TATA binding factor;} |
| Our model(multitask) | protein: {human TATA binding factor; transcriptionally active human TATA binding factor} |

Table 9: An example of predicted results in GENIA test dataset.

For our pipeline model, without the dependencies information from entity categorical labels, it misses the outer entity boundaries and only extracts the inner one. It verifies that the multitask learning can share boundary information between boundary detection module and entity categorical

label prediction module, which is very effective for nested NER.

## 7 Conclusion

This paper presents a boundary-aware model which leverages boundaries to predict entity categorical labels. Our model combines sequence labeling model and region classification model to locate and classify nested entities with high performance. To capture the underlying dependencies of boundary detection module and entity categorical prediction module, we apply a multitask loss for training the two tasks simultaneously. Our model outperforms existing nested models in terms of F-score.

For future work, we consider to model the dependencies among entity regions explicitly and improve the performance of boundary detection module which is important for entity categorical label prediction.

## Acknowledgement

This work was supported by the Fundamental Research Funds for the Central Universities, SCUT (No. 2017ZD048, D2182480), the Science and Technology Planning Project of Guangdong Province (No.2017B050506004), the Science and Technology Programs of Guangzhou (No. 201704030076,201802010027,201902010046) and a CUHK Research Committee Funding (Direct Grants) (Project Code: EE16963).

## References

Beatrice Alex, Barry Haddow, and Claire Grover. 2007. Recognising nested named entities in biomedical text. In *Proceedings of the Workshop on BioNLP 2007: Biological, Translational, and Clinical Language Processing*, pages 65–72. Association for Computational Linguistics.

Jonathan Baxter. 1997. A bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine learning*, 28(1):7–39.

Darina Benikova, Chris Biemann, and Marc Reznicek. 2014. Nosta-d named entity annotation for german: Guidelines and dataset. In *LREC*, pages 2524–2531.

Kai-Wei Chang, Rajhans Samdani, and Dan Roth. 2013. A constrained latent variable model for coreference resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 601–612.

Billy Chiu, Gamal Crichton, Anna Korhonen, and Sampo Pyysalo. 2016. How to train good word embeddings for biomedical nlp. In *Proceedings of the 15th workshop on biomedical natural language processing*, pages 166–174.

Jenny Rose Finkel and Christopher D Manning. 2009. Nested named entity recognition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 141–150. Association for Computational Linguistics.

Mourad Gridach. 2017. Character-level neural network for biomedical named entity recognition. *Journal of biomedical informatics*, 70:85–91.

Nitish Gupta, Sameer Singh, and Dan Roth. 2017. Entity linking via joint encoding of types, descriptions, and context. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2681–2690.

Meizhi Ju, Makoto Miwa, and Sophia Ananiadou. 2018. A neural layered model for nested named entity recognition. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1446–1459.

J-D Kim, Tomoko Ohta, Yuka Tateisi, and Jun‘‘ichi Tsujii. 2003. Genia corpus‘‘a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(suppl_1):i180–i182.

Jin-Dong Kim, Tomoko Ohta, Yoshimasa Tsuruoka, Yuka Tateisi, and Nigel Collier. 2004. Introduction to the bio-entity recognition task at jnlpba. In *Proceedings of the international joint workshop on natural language processing in biomedicine and its applications*, pages 70–75. Citeseer.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Liyuan Liu, Xiang Ren, Qi Zhu, Shi Zhi, Huan Gui, Heng Ji, and Jiawei Han. 2017. Heterogeneous supervision for relation extraction: A representation learning approach. *arXiv preprint arXiv:1707.00166*.

Wei Lu and Dan Roth. 2015. Joint mention extraction and classification with mention hypergraphs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 857–867.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Aldrian Obaja Muis and Wei Lu. 2017. Labeling gaps between words: Recognizing overlapping mentions with mention separators. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2608–2618, Copenhagen, Denmark. Association for Computational Linguistics.

Da Ren, Yi Cai, Xue Lei, Jingyun Xu, Qing Li, and Ho-fung Leung. 2019. A multi-encoder neural conversation model. *Neurocomputing*, 358:344–354.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

Dan Shen, Jie Zhang, Guodong Zhou, Jian Su, and Chew-Lim Tan. 2003. Effective adaptation of a hidden markov model-based named entity recognizer for biomedical domain. In *Proceedings of the ACL 2003 workshop on Natural language processing in biomedicine-Volume 13*, pages 49–56. Association for Computational Linguistics.

Mohammad Golam Sohrab and Makoto Miwa. 2018. Deep exhaustive model for nested named entity recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2843–2849.

Emma Strubell, Patrick Verga, David Belanger, and Andrew McCallum. 2017. Fast and accurate entity recognition with iterated dilated convolutions. *arXiv preprint arXiv:1702.02098*.

Bailin Wang, Wei Lu, Yu Wang, and Hongxia Jin. 2018. A neural transition-based model for nested mention recognition. *arXiv preprint arXiv:1810.01808*.

Jie Zhang, Dan Shen, Guodong Zhou, Jian Su, and Chew-Lim Tan. 2004. Enhancing hmm-based biomedical named entity recognition by studying special phenomena. *Journal of biomedical informatics*, 37(6):411–422.

GD Zhou. 2006. Recognizing names in biomedical texts using mutual information independence model and svm plus sigmoid. *International Journal of Medical Informatics*, 75(6):456–467.

Guodong Zhou, Jie Zhang, Jian Su, Dan Shen, and Chewlim Tan. 2004. Recognizing names in biomedical texts: a machine learning approach. *Bioinformatics*, 20(7):1178–1190.

# Learning the Extraction Order of Multiple Relational Facts in a Sentence with Reinforcement Learning

**Xiangrong Zeng**[1,2,3], **Shizhu He**[1,2], **Daojian Zeng**[4], **Kang Liu**[1,2], **Jun Zhao**[1,2]
[1]NLPR, Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, China
[2]University of Chinese Academy of Sciences, Beijing, 100049, China
[3]Unisound AI Technology Co.,Ltd, Beijing, 100000, China
[4]Changsha University of Science & Technology, Changsha, 410114, China
{xiangrong.zeng, shizhu.he, kliu, jzhao}@nlpr.ia.ac.cn
zengdj@csust.edu.cn

## Abstract

The multiple relation extraction task tries to extract all relational facts from a sentence. Existing works didn't consider the extraction order of relational facts in a sentence. In this paper we argue that the extraction order is important in this task. To take the extraction order into consideration, we apply the reinforcement learning into a sequence-to-sequence model. The proposed model could generate relational facts freely. Widely conducted experiments on two public datasets demonstrate the efficacy of the proposed method.

## 1 Introduction

Relation extraction (RE) is a core task in natural language processing (NLP). RE can be used in information extraction (Wu and Weld, 2010), question answering (Yih et al., 2015; Dai et al., 2016) and other NLP tasks. Most existing works assumed that a sentence only contains one relational facts (a relational fact, or a triplet, contains a relation and two entities). But in fact, a sentence often contains multiple relational facts (Zeng et al., 2018b). The multiple relation extraction task tries to extract all relational facts from a sentence.

Existing works on multiple relation extraction task can be divided into five genres. 1) PseudoPipeline genre, including Miwa and Bansal (2016); Sun et al. (2018). They first recognized all the entities of the sentence, then extracted features for each entity pair and predicted their relation. They trained the entity recognition model and relation prediction model together instead of separately. Therefore, we call them PseudoPipeline methods. 2) TableFilling genre, including Miwa and Sasaki (2014); Gupta et al. (2016) and Zhang et al. (2017). They maintained a entity-relation table and predicted a semantic tag (either entity tags or relation tags) for each cell in the table. According to the predicted tags, they can recognize the entities and the relation between each entity pair. 3) NovelTagging genre, including Zheng et al. (2017). This method can be seen as a development of TableFilling method. They assigned a pre-defined semantic tag to each word of the sentence and collected triplets based on the tags. Their tags include both entity and relation information. Therefore, they don't need to maintain a entity-relation table. 4) MultiHeadSelection genre, including Bekoulis et al. (2018a) and Bekoulis et al. (2018b). They first recognized the entities, then they formulated the relation extraction task as a multi-head selection problem. For each entity, they calculated the score between it and every other entities for a given relation. The combination of the entity pair and relation with the score exceeding a threshold will be kept as a triplet. 5) Generative genre, including Zeng et al. (2018b). They directly generate triplets one by one by a sequence-to-sequence model with copy mechanism (Gu et al., 2016; Vinyals et al., 2015). To generate a triplet, they first generated the relation, then they copy the first entity and the second entity from the source sentence.

However, none of them have considered the extraction order of multiple triplets in a sentence. Given a sentence, the PseudoPipeline methods extract relations of different entity pairs separately. Although they jointly training the entity model and relation model, they ignore the influence between triplets actually. The TableFilling, NovelTagging and MultiHeadSelection methods extract the triplets in the word order of this sentence. They firstly deal with the first word, then the second one and so on. The generative method could generate triplets in any order actually. However, Zeng et al. (2018b) randomly choose the extraction order of the triplets in each sentence. Sorting the triplets in the sentences of training data beforehand with



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 367–377, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| | | |
|---|---|---|
| Sentence | Cubanelle is in Arros negre, a dish from the Catalonia region. | |
| Relational Facts | #1 | $F_2$: <Arros negre, food_region, Catalonia> |
| | #2 | $F_1$: <Arros negre, ingredient, Cubanelle> |

Figure 1: Example of multiple relation extraction. In this example, it's easier to extract $F_2$ first. The extraction of $F_1$ can be benefit from $F_2$.

global rules (e.g., alphabetical order) is straightforward. But one global sorting rule may not fit every sentences.

In this paper, we argue that the extraction order of triplets in a sentence is important. Take Figure 1 as example. It's difficult to extract $F_1$ first because we don't know what "Arros negre" is in the first place. Extracting $F_2$ is more straightforward as the key words "dish", "region" in the sentence is helpful. $F_2$ can help us to extract $F_1$ because now we are confident that "Arros negre" is some kind of food, so that "ingredient" is a suitable relation between "Arros negre" and "Cubanelle". From this intuitive example, we can see that the extracted triplets could influence the extraction of the remaining triplets.

To automatically learning the extraction order of multiple relational facts in a sentence, we propose a sequence-to-sequence model and apply reinforcement learning (RL) on it. we follow the Generative genre because such a model could extract triplets in various order, which is convenient for us to explore the influence of triplets extraction order. Our model reads in a raw sentence and generates triplets one by one. Thus, all triplets in a sentence could be extracted. To take the triplets extraction order into consideration, we convert the triplets generation process as a RL process. The sequence-to-sequence model is regarded as the RL policy. The action is what we generate in each time step. We assume that a better generation order could lead to more valid generated triplets. The RL reward is related to the generated triplets. In general, the more triplets are correctly generated, the higher the reward. Unlike supervised learning with negative log likelihood (NLL) loss, which forces the model to generate triplets in the order of the ground truth, reinforcement learning allows the model generate triplets freely to achieve higher reward.

The main contributions of this work are:

- We discuss the triplets extraction order problem in the multiple relation extraction task. In our knowledge, this problem has never been addressed before.
- We apply reinforcement learning method on a sequence-to-sequence model to handle this problem.
- We conduct widely experiments on two public datasets. Experimental results show that the proposed method outperform the strong baselines with 3.4% and 5.5% improvements respectively.

## 2 Related Work

Given a sentence with two annotated entities (an entity pair), the relation classification task aims to identify the predefined relation between these two entities. Zeng et al. (2014) was among the first to apply neural networks in relation classification task. They adopted the Convolutional Neural Network (CNN) to learn the sentence representation automatically. In the following, dos Santos et al. (2015); Xu et al. (2015a) also applied CNN to extract relation. Xu et al. (2015b) utilized shortest dependency path between two entities with a LSTM (Hochreiter and Schmidhuber, 1997) based recurrent neural network. Zhou et al. (2016) applied attention mechanism to learn different weights for each word and used LSTM to represent sentence. These methods all assumed that the entity pair is given beforehand and a sentence only contains two entities.

To extract both entities and relation from sentence, early works like Zelenko et al. (2003); Chan and Roth (2011) adopted pipeline methods. However, such pipeline methods neglect the relevance between entities and relation. Latter works focused on joint models that extract entities and relation jointly. Yu and Lam (2010); Li and Ji (2014); Miwa and Bansal (2016) relied on NLP tools to do feature engineering, which suffered from the error propagation problem. Miwa and Sasaki (2014); Gupta et al. (2016); Zhang et al. (2017) applied neural networks to jointly extract entities and relations. They converted the relation extraction task into a table filling task. Zheng et al. (2017) took a step further and converted this task into a tagging task. They assigned a semantic tag to each word in the sentence and collected triplets according to the tag information. Bekoulis et al. (2018b,a) model the relation extraction task as a multi-head selec-

tion problem. However, these models can not take triplet's extraction order into consideration. Sun et al. (2018) proposed a joint learning paradigm based on minimum risk training. Their method ignore the influence between relational facts. Zeng et al. (2018b) proposed an sequence-to-sequence model with copy mechanism to handle the overlapping problem in multiple relation extraction. They randomly choose a extraction order for each sentence.

RL has attracted lot of attention recently. It has been successfully applied in many games (Mnih et al., 2015; Silver et al., 2016). Narasimhan et al. (2015); He et al. (2016) applied RL on text based games. Narasimhan et al. (2016) employed deep Q-network to optimize a reward function that reflects the extraction accuracy while penalizing extra effort. Li et al. (2016) applied policy gradient method to model future reward in chatbot dialogue. They designed a reward to promote three conversational properties: informativity, coherence and ease of answering. Su et al. (2016) using on-line activate reward learning for policy optimization in spoken dialogue systems because the user feedback is often unreliable and costly to collect. Yu et al. (2017) applied RL method to overcome the limitations that the Generative Adversarial Net (GAN) in generating sequences of discrete tokens. Our work is related to Li et al. (2016); Yu et al. (2017) since we also apply RL to generate better sequences.

There are several works that related to both relation extraction and RL, which are also related to our work. Zeng et al. (2018a); Feng et al. (2018); Qin et al. (2018) applied RL to distantly supervised relation extraction task. Zeng et al. (2018a) turned the bag relation prediction into an RL process. They assumed that the relation of the bag is determined by the relation of sentences from the bag. They set the final reward to +1 or -1 by comparing the predict bag relation with the gold relation. Feng et al. (2018) adopted policy gradient method to select high-quality sentences from the bag. The selected sentences are feed to the relation classifier and the relation classifier provides rewards to the instance selector. Similarly, Qin et al. (2018) explored a deep RL strategy to generate the false-positive indicator. Our work is different from them since we focus on supervised relation extraction task.

## 3 Method

We first introduce our basic model and then introduce how to apply RL on it. Similar to Zeng et al. (2018b), our neural model is also a sequence-to-sequence model with copy mechanism. It reads in a raw sentence and generates triplets one by one. Instead of training the model with NLL loss, we regard the triplets generation process as a RL process and optimize the model with REINFORCE (Williams, 1992) algorithm. Therefore, we don't have to determine the triplets order of each sentence beforehand, we let the model generate triplets freely. We show the RL process in Figure 2.

### 3.1 Sequence-to-Sequence Model with Copy Mechanism

The sequence-to-sequence model with copy mechanism is a kind of CopyNet (Gu et al., 2016) or PointerNetwork (Vinyals et al., 2015). Two components included in this model: encoder and decoder. The encoder is a bi-directional recurrent neural network, which is used to encode a variable-length sentence into a fixed-length vector. We denote the outputs of encoder as $\mathbf{O}^E = [\mathbf{o}_1^E, ..., \mathbf{o}_n^E]$, where $\mathbf{o}_i^E$ denotes the output of $i$-th word of the encoder and $n$ is the sentence length.

The decoder is another recurrent neural network, which is used to generate triplets one by one. The *NA*-triplets will be generated if the valid triplets number is less than the maximum triplets number.[1] It takes three-time steps to generate one triplet. That is, in time step $t$ ($t = 1, 2, 3, ..., T$), if $t\%3 = 1$, we predict the relation. If $t\%3 = 2$, we copy the first entity and if $t\%3 = 0$, we copy the second entity. $T$ is the maximum decode time step. Note that $T$ is always divisible by 3.

Suppose there are $m$ predefined valid relations, in time step $t$ ($t = 1, 4, 7, ...$), we calculate the confidence score for each valid relation:

$$\mathbf{q}_t^r = selu(\mathbf{o}_t^D \cdot \mathbf{W}_t^r + \mathbf{b}_t^r) \quad (1)$$

where $\mathbf{o}_t^D$ is the output of decoder in time step $t$; $\mathbf{W}_t^r$ is the weight matrix and $\mathbf{b}_t^r$ is the bias in time step $t$; $selu(\cdot)$ (Klambauer et al., 2017) is activation function. To allow the model to generate *NA*-triplet, we also calculate the confidence score for

[1] *NA*-triplet is a special triplet proposed in Zeng et al. (2018b). It's similar to the "eos" symbol in neural sentence generation.

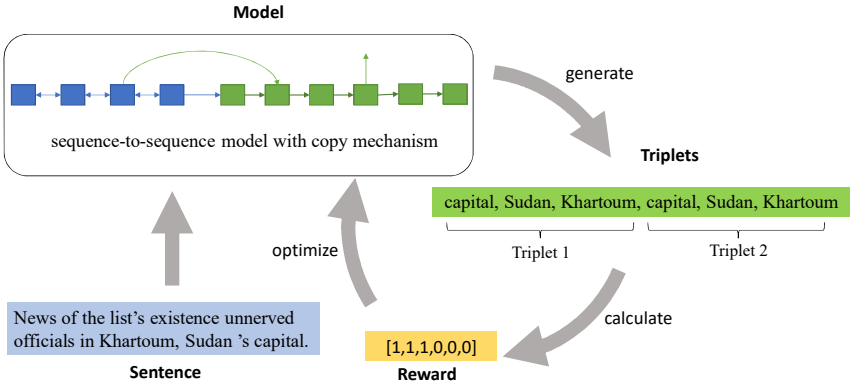


Figure 2: The RL process. The model reads in a raw sentence and generates triplets. Then, a reward is assigned to each time step based on the generated triplets. Lastly, the rewards is used to optimize the model.

*NA* relation:

$$\mathbf{q}_t^{NA} = selu(\mathbf{o}_t^D \cdot \mathbf{W}_t^{NA} + \mathbf{b}_t^{NA}) \quad (2)$$

where $\mathbf{W}_t^{NA}$ and $\mathbf{b}_t^{NA}$ are parameters in time step $t$. Then we concatenate $\mathbf{q}_t^r$ and $\mathbf{q}_t^{NA}$ and perform softmax to obtain the probability distribution:

$$\mathbf{p}_t^r = softmax([\mathbf{q}_t^r; \mathbf{q}_t^{NA}]) \quad (3)$$

To copy the first entity in time step $t$ ($t = 2, 5, 8, ...$), we calculate the confidence score of each word in source sentence:

$$q_{ti}^e = selu([\mathbf{o}_t^D; \mathbf{o}_i^E] \cdot \mathbf{w}_t^e) \quad (4)$$

where $q_{ti}^e$ is the confidence score of $i$-th word and $\mathbf{w}_t^e$ is the weight vector, in time step $t$. Similarly, to take the *NA*-triplet into consideration, we also calculate the confidence score for *NA* entity with Eq 2. We concatenate them and perform softmax to obtain the probability distribution:

$$\mathbf{p}_t^e = softmax([[q_{t1}^e, ..., q_{tn}^e]; \mathbf{q}_t^{NA}]) \quad (5)$$

Copy the second entity in time step $t$ ($t = 3, 6, 9, ...$) is almost the same as the first entity. The only difference is we also apply the mask (Zeng et al., 2018b) to avoid the copied two entities are the same.

Our model is similar to OneDecoder model and MultiDecoder model in Zeng et al. (2018b). Compared with OneDecoder model, our model using different linear transformation parameters in different decoding time step. Compared with MultiDecoder model, our model using only one decoder cell to decode all triplets. In our model, we didn't using attention mechanism because we found that the attention mechanism makes no difference to the results.

### 3.2 Reinforcement Learning Process

We regard the triplets generation process as RL process. The loop in Figure 2 represents a RL episode. In each RL episode, the model reads in the raw sentence and generate output sequence. Then we gain triplets from the output sequence and calculate rewards based on them. Finally, we optimize the model with REINFORCE algorithm.

#### State

We use $s_t$ to denote the state of sentence $x$ in decoding time step $t$. The state $s_t$ contains the already generated tokens $\hat{y}^{<t}$, the information of source sentence $x$ and the model parameters $\theta$.

$$s_t = (\hat{y}^{<t}, x, \theta) \quad (6)$$

#### Action

The action is what we predict (or copy) in each time step. In time step $t$ and $t\%3 = 1$, the model (policy) is required to determine the relation of the triplet; In time step $t$ where $t\%3 = 2$ or 0, the model is required to determine the first or second entity, which is copied from the source sentence. Therefore, the action space $A$ is varied in different time step $t$.

$$A = \begin{cases} R, & t\%3 = 1 \\ P, & t\%3 = 2, 0 \end{cases} \quad (7)$$

where $R$ is the predefined relations and $P$ is the positions of source sentence. We denote the action sequence of the source sentence as $a = [a_1, ..., a_T]$.

**Algorithm 1** Reward Assignment

**Input:** Sampled action sequence $a = [a_1, ..., a_T]$; Gold triplets set $G$; *NA*-triplet *NA*.
**Output:** Rewards of each action $r = [r_1, ..., r_T]$.

```
Number of generated triplets K = T/3;
Generated triplets list F = [F_1, ..., F_K];
Already generated triplets set V = {}.
for each i ∈ [1, K] do
    F_i = [a_{3*i-2}, a_{3*i-1}, a_{3*i}]
end for
for each i ∈ [1, K] do
    r_{3*i-2} = r_{3*i-1} = r_{3*i} = 0
    if i ≤ |G| then
        if F_i ∈ G and F_i ∉ V then
            Add F_i to V
            r_{3*i-2} = r_{3*i-1} = r_{3*i} = 1
        end if
    else if F_i = NA then
        r_{3*i-2} = r_{3*i-1} = r_{3*i} = 0.5
    end if
end for
```

### Reward

The reward is used to guide the training, which is critical to RL training. However, we can't assign a reward to each step directly during the generation since we don't know whether each action we choose is good or not before we finish the generation. Remind that we could obtain a triplet in every three steps. Once we obtained a triplet, we can compare it with the gold triplets and know if this triplet is good or not. A well generated triplet means it's the same with one of the gold triplets and not the same with any already generated triplets.

When we obtained a good triplet after three steps, we assign reward 1 to each of these three steps. Otherwise, we assign reward 0 to them. After generating valid triplets, we may need to generate *NA*-triplets. We assign reward 0.5 to each of these three steps if we correctly generate *NA*-triplet and reward 0 otherwise. We show the details of the reward assignment in Algorithm 1[2].

## 3.3 Training

The model can be trained with either supervised learning loss or reinforcement learning loss. However, the supervised learning forces the model to generate triplets in the order of the ground truth while the reinforcement learning allows the model generate triplets freely.

### NLL Loss

Training the model with NLL loss requires a predefined ground truth sequence for each sentence. Suppose $T$ is the maximum time step of decoder, we denote the ground truth sequence as $[y_1, ..., y_t, ..., y_T]$. Them the NLL loss for sentence $x$ can be defined as:

$$L = \frac{1}{T}\sum_{t=1}^{T} -log(p(y_t|\hat{y}^{<t}, x, \theta)) \quad (8)$$

where $\hat{y}^{<t}$ is the already generated tokens; $p(\cdot|\cdot)$ is the conditional probability; $\theta$ is the parameters of the entire model.

### RL Loss

Training the model with reinforcement learning only require the ground truth triplets for each sentence. The RL loss for sentence $x$ is:

$$L = \frac{1}{T}\sum_{t=1}^{T} -log(p(\hat{y}_t|\hat{y}^{<t}, x, \theta)) * r_t \quad (9)$$

where $\hat{y}_t$ is the sampled action and $r_t$ is the reward, in time step $t$.

# 4 Experiments

## 4.1 Dataset

Following Zeng et al. (2018b), we test our method on two open datasets: New York Times (NYT) dataset and WebNLG dataset.

NYT dataset is proposed by Riedel et al. (2010). This dataset is produced by distant supervision method which automatically aligns Freebase with New York Times news articles. Like Zheng et al. (2017); Zeng et al. (2018b) do, we ignore the noise in this dataset and use it as a supervised dataset. We use the pre-processed dataset used in Zeng et al. (2018b), which contains 5000 sentences in the test set and 5000 sentences in the validation set and 56195 sentences in the train set. In the train set, there are 36868 sentences that contain one triplet, 19327 sentences that contain multiple triplets. In the test set, the sentence number are 3244 and 1756, respectively. There are 24 relations in total.

WebNLG dataset is proposed by Gardent et al. (2017). This data set is originally created for Natural Language Generation (NLG) task. Given a

[2]How to determine the reward in RL is difficult. We tried several different reward assignments but only this one works.

group of triplets, annotators are asked to write a sentence which contains the information of all triplets in this group. We use the dataset preprocessed by Zeng et al. (2018b) and the train set contains 5019 sentences, the test set contains 703 sentences and the validation set contains 500 sentences. In the train set, there are 1596 sentences that contain one triplet, 3423 sentences contain multiple triplets. In the test set, the sentence number are 266 and 437, respectively. There are 246 different relations.

### 4.2 Settings

Zeng et al. (2018b) only use LSTM as the model cell. In this paper, we report the results of both LSTM and GRU (Cho et al., 2014). We follow the most settings from Zeng et al. (2018b). The cell unit number is set to 1000; The embedding dimension is set to 100; The batch size is 100; The maximum time step $T$ is 15, that is, we will extract 5 triplets for each sentence; We use Adam (Kingma and Ba, 2015) to optimize parameters and stop the training when we find the best result in validation set. For the NLL training, the learning rate in both dataset is 0.001. For the RL training, we first pretrain the model with NLL training (pretrain model achieves 80%-90% of the best NLL training performance), then training the model with RL. The RL learning rate is 0.0005.

### 4.3 Evaluation Metrics

We follow the evaluation metrics in Zeng et al. (2018b). Our model can only copy one word for each entity and we use the last word of each entity to represent them. Triplet is regarded as correct when its relation, the first entity and the second entity are all correct. For example, suppose the gold triplets is $< Barack\ Obama, president, USA >$, $< Obama, president, USA >$ is regarded as correct while $< Obama, locate, USA >$ and $< Barack, prsident, USA >$ are not. A triplet is regarded as *NA*-triplet when and only when its relation is NA relation and it has a NA entity pair. The predicted *NA*-triplet will be excluded. We use the standard micro Precision, Recall and F1 score to evaluate the results.

### 4.4 Results of Different Extraction Order

To find out if the triplets extraction order of a sentence can make difference in multiple relation extraction task, we conduct widely experiments on both NYT and WebNLG dataset. We show the results of different extraction order of different models with LSTM cell in Table 1. The results of models with GRU cell are shown in Appendix B. We box the best results of a model and the bold values are the best results in this dataset.

**CNN** denotes the baseline with CNN classifier. We use the NLTK toolkit[3] to recognize the entities first. Then we combine every two entities as an entity pair. Every two entities can lead to two different entity pairs. For each entity pair, we apply a CNN classifier (Zeng et al., 2014) to determine the relation. We leave the details of this model in Appendix A. **ONE** and **MULTI** denotes the OneDecoder model and MultiDecoder model in Zeng et al. (2018b).[4] **NLL** means the model is trained with NLL loss, which requires a predefined ground truth sequence for each sentence. For a sentence with $N$ triplets, there are $N!$ (the factorial of $N$) possible extraction order, which lead to $N!$ valid sequences. **Shuffle** means we randomly select one valid sequence as the ground truth sequence in every training epoch for a sentence. **FixUnsort** means we randomly select one valid sequence before training, and use the selected one as ground truth sequence during training. This strategy is used in Zeng et al. (2018b). **Alphabetical** means we sort the triplets of a sentence in alphabetical order and build ground truth sequence based on the sorted triplets. **Frequency** means we sort the triplets of a sentence based on the relation frequency. We count the relation frequency from the training set. **RL** means the model is trained with reinforcement learning. In NYT dataset, we using Alphabetical strategy to pretrain the model, and in WebNLG dataset, we pretrain the model wtih Frequency strategy.

Form Table 1, we can observe that: (a) The CNN baseline is not performing well because this model neglect the influence between triplets. (b) Compared with FixUnsort strategy, simply change the ground truth sequence in different training epoch (the Shuffle strategy) is also not good. The performance of OneDecoder drops from 0.566 to 0.552 in NYT dataset and 0.305 to 0.283 in WebNLG dataset. (c) In both dataset and for all models trained with NLL loss, sort the triplets in some order (Alphabetical or Frequency order)

[3] https://www.nltk.org/

[4] We reimplement these two models and find that the attention mechanism is not important. Therefore, we report the results without applying the attention mechanism.

| Model | NYT | | | WebNLG | | |
|---|---|---|---|---|---|---|
| | Precision | Recall | F1 | Precision | Recall | F1 |
| CNN | 0.468 | 0.620 | 0.533 | 0.304 | 0.417 | 0.352 |
| ONE+NLL (Shuffle) | 0.635 | 0.488 | 0.552 | 0.400 | 0.219 | 0.283 |
| ONE+NLL (FixUnsort) | 0.606 | 0.530 | 0.566 | 0.320 | 0.291 | 0.305 |
| ONE+NLL (Alphabetical) | 0.616 | 0.553 | 0.583 | 0.305 | 0.292 | 0.298 |
| ONE+NLL (Frequency) | 0.609 | 0.546 | 0.576 | 0.323 | 0.311 | 0.317 |
| ONE+RL | 0.715 | 0.533 | 0.610 | 0.668 | 0.307 | 0.421 |
| MULTI+NLL (Shuffle) | 0.629 | 0.522 | 0.570 | 0.363 | 0.278 | 0.315 |
| MULTI+NLL (FixUnsort) | 0.600 | 0.562 | 0.580 | 0.434 | 0.431 | 0.432 |
| MULTI+NLL (Alphabetical) | 0.685 | 0.647 | 0.665 | 0.482 | 0.481 | 0.481 |
| MULTI+NLL (Frequency) | 0.648 | 0.604 | 0.625 | 0.519 | 0.518 | 0.518 |
| MULTI+RL | 0.742 | 0.639 | 0.687 | 0.611 | 0.523 | 0.564 |
| Our+NLL (Shuffle) | 0.670 | 0.545 | 0.601 | 0.417 | 0.310 | 0.356 |
| Our+NLL (FixUnsort) | 0.645 | 0.591 | 0.617 | 0.490 | 0.488 | 0.489 |
| Our+NLL (Alphabetical) | 0.717 | **0.678** | 0.697 | 0.533 | 0.537 | 0.535 |
| Our+NLL (Frequency) | 0.688 | 0.652 | 0.669 | 0.581 | 0.587 | 0.584 |
| Our+RL | **0.779** | 0.672 | **0.721** | **0.633** | **0.599** | **0.616** |

Table 1: Results of different extraction order of models with LSTM cell.

can lead to better performance. For example, our model achieves 0.617 F1 score under FixUnsort strategy, while 0.697 and 0.669 F1 score under Alphabetical and Frequency strategy in NYT dataset. This observation verifies that the triplets extracting order of a sentence is important in multiple relation extraction task. (d) Another interesting observation is that the NLL trained model can achieve the best F1 score in NYT dataset if we sort the triplets in alphabetical order, while in WebNLG dataset, we need to sort the triplets in relation frequency order. This observation demonstrates that a global sorting rule may not fit for every dataset. The Alphabetical strategy is better for NYT dataset while the Frequency strategy is better for WebNLG dataset. (e) We can also observe that the model trained with RL can achieve better result than any NLL sorting strategy. For example, in WebNLG dataset, MultiDecoder model only achieves 0.481 and 0.518 F1 score with Alphabetical and Frequency strategy, it achieves 0.564 F1 score with RL. Our model trained with RL achieves the best performance on both NYT and WebNLG dataset, which is 0.721 and 0.616. It's 3.4% and 5.5% improvements compared with the best global sorting rule on these two datasets.

### 4.5 Extraction Order Comparison

This experiment compares the extraction order of our model trained with RL. We test our model on WebNLG dataset. The generated triplets (excluding NA-triplets) sequence of our model which is pretrained with FixUnsort strategy then trained with RL, is denoted as **FURL**. Similarly, the triplets sequence of our model which is pretrained with Frequency strategy then trained with RL is denoted as **FreqRL**. And the triplets sequence of our model which is trained with FixUnsort strategy is denoted as **FUNLL**.

| Comparison | LSTM | GRU |
|---|---|---|
| FUNLL-FURL | 0.326 | 0.390 |
| FreqRL-FURL | 0.446 | 0.435 |

Table 2: The extraction order comparison.

Suppose $A = [F_a, F_b, F_c]$ is the generated triplets sequence of FURL for sentence $x$, $B = [F_a, F_c]$ is the generated triplets sequence of FUNLL for the same sentence $x$. $F_a$, as well as $F_b$ and $F_c$, is a triplet. The first triplet of the sequence $A$ is $F_a$, which is the same with the first triplet of the sequence $B$. But the second triplet of $A$ is different from $B$. Therefore, there are only 1 triplet is in the same position for $A$ and $B$. The triplets number is the maximum triplets number of $A$ and $B$, which is 3 in this example. We calculate the order comparison of sentence $x$ as $1/3 = 0.333$. The order comparison of FUNLL and FURL (denote as FUNLL-FURL) is the mean value of all

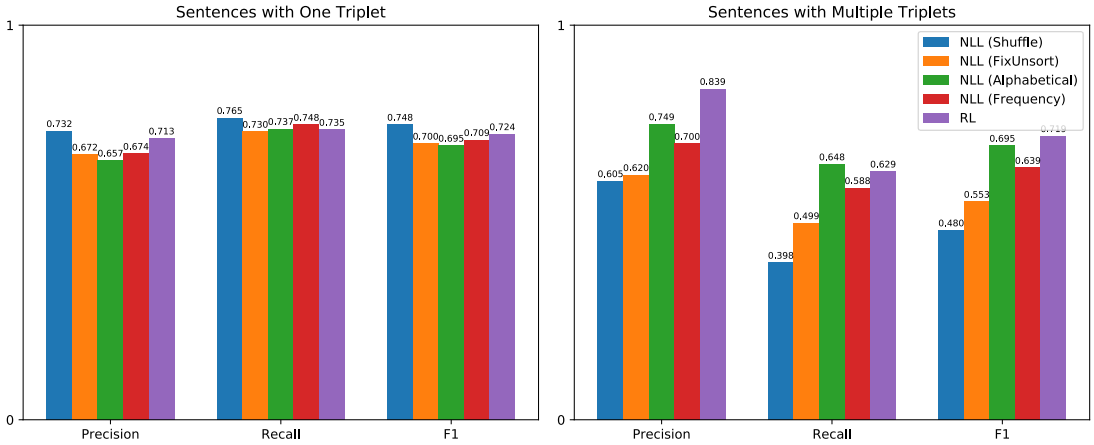


Figure 3: Results of our model with LSTM cell under different training strategies on NYT dataset.

sentences.

We show the order comparison results of our model with LSTM and GRU cell in Table 2. As we can see, although FURL model is pretrained by FUNLL, FURL is more alike FreqRL (0.446 and 0.435), rather then FUNLL (0.326 and 0.390).

This experiment verified that after RL training, the model trend to generate triplets in the same order.

### 4.6 Multiple Relation Extraction

To verify the ability of extracting multiple relational facts, we conduct the experiment on NYT dataset of our model with LSTM cell. We show the results in Figure 3. The left part of this figure shows the performance of sentences with one triplet. The right part shows the performance of sentences with multiple triplets.

As we can see, when the sentence only contains one triplet, our model trained with RL can achieve comparative performance with the strong baselines. When there are multiple triplets for a sentence, our model trained with RL outperform all baselines significantly. By training with RL, our model could extract triplets more precisely. Although the recall value is slightly lower then NLL training with Frequency strategy, it exceeds other baselines significantly. These observations demonstrate that RL training is effective to handle the multiple relation extraction task.

## 5 Weakness

Although we overcome all strong baselines by training the model with RL, there are still some weaknesses in our method.

The first weakness is the decrease in recall. Table 1 shows that NLL training with Alphabetical or Frequency strategy achieves the highest recall in most cases. Training the model with RL achieves the highest precision and relatively low recall. This phenomenon demonstrates that the model trained with RL generates relatively fewer triplets. Although we can extract triplets more accurate, it is still a weakness of our method since we try to extract all triplets from a sentence.

The second weakness is our model can only copy one word for each entity. Following Zeng et al. (2018b), we only copy the last word of an entity. But in reality, most entities contains more than one word. In the future, we will consider how to extract the complete entity. For example, we could add the BIO tag prediction in the encoder and train the BIO loss together with current loss. Therefore, we can recognize the complete entity with the help of BIO tags. Or, we can take two steps to generate one entity, one step for the head word and the other for the tail word.

## 6 Conclusions

In this paper, we discuss the multiple triplets extraction order problem in the multiple relation extraction task. We propose a sequence-to-sequence model with reinforcement learning to take the extraction order into consideration. Widely experiments on NYT dataset and WebNLG dataset are conducted and verified that the proposed method is effective in handling this problem.

## 7 Acknowledgments

This work is supported by the National Natural Science Foundation of China (No.61533018, No.61702512) and the independent research project of National Laboratory of Pattern Recognition.

## References

Giannis Bekoulis, Johannes Deleu, Thomas Demeester, and Chris Develder. 2018a. Adversarial training for multi-context joint entity and relation extraction.

Giannis Bekoulis, Johannes Deleu, Thomas Demeester, and Chris Develder. 2018b. Joint entity recognition and relation extraction as a multi-head selection problem.

Yee Seng Chan and Dan Roth. 2011. Exploiting syntactico-semantic structures for relation extraction. In *Proceedings of ACL*, pages 551–560.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the EMNLP*, pages 1724–1734.

Zihang Dai, Lei Li, and Wei Xu. 2016. Cfo: Conditional focused neural question answering with large-scale knowledge bases. In *Proceedings of ACL*, pages 800–810.

Jun Feng, Minlie Huang, Li Zhao, Yang Yang, and Xiaoyan Zhu. 2018. Reinforcement learning for relation classification from nosy data. In *Proceedings of AAAI*.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. Creating training corpora for nlg micro-planners. In *Proceedings of ACL*, pages 179–188.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of ACL*, pages 1631–1640.

Pankaj Gupta, Hinrich Schtze, and Bernt Andrassy. 2016. Table filling multi-task recurrent neural network for joint entity and relation extraction. In *Proceedings of COLING*, pages 2537–2547.

Ji He, Jianshu Chen, Xiaodong He, Jianfeng Gao, Lihong Li, Li Deng, and Mari Ostendorf. 2016. Deep reinforcement learning with a natural language action space. In *Proceedings of ACL*, pages 1621–1630.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: a Method for Stochastic Optimization. In *Proceedings of ICLR*, pages 1–15.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. 2017. Self-normalizing neural networks. In *Advances in NIPS*, pages 971–980.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016. Deep Reinforcement Learning for Dialogue Generation. In *Proceedings of EMNLP*, pages 1192–1202.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of ACL*, pages 402–412.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *Proceedings of ACL*, pages 1105–1116.

Makoto Miwa and Yutaka Sasaki. 2014. Modeling joint entity and relation extraction with table representation. In *Proceedings of EMNLP*, pages 1858–1869.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. 2015. Language understanding for text-based games using deep reinforcement learning. In *Proceedings of EMNLP*, pages 1–11.

Karthik Narasimhan, Adam Yala, and Regina Barzilay. 2016. Improving information extraction by acquiring external evidence with reinforcement learning. In *Proceedings of EMNLP*, pages 2355–2365.

Pengda Qin, Weiran Xu, and William Yang Wang. 2018. Robust distant supervision relation extraction via deep reinforcement learning. In *Proceedings of ACL*.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Proceedings of ECML PKDD*, pages 148–163.

Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proceedings of ACL*, pages 626–634.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

Pei-Hao Su, Milica Gasic, Nikola Mrki, Lina M. Rojas Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. On-line active reward learning for policy optimisation in spoken dialogue systems. In *Proceedings of ACL*, pages 2431–2441.

Changzhi Sun, Yuanbin Wu, Man Lan, Shiliang Sun, Wenting Wang, Kuang-Chih Lee, and Kewen Wu. 2018. Extracting entities and relations with joint minimum risk training. In *Proceedings of EMNLP*, pages 2256–2265.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in NIPS*, pages 2692–2700.

Ronald J Williams. 1992. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 8:229–256.

Fei Wu and Daniel S. Weld. 2010. Open information extraction using wikipedia. In *Proceedings of ACL*, pages 118–127.

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2015a. Semantic relation classification via convolutional neural networks with simple negative sampling. In *Proceedings of EMNLP*, pages 536–540.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015b. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of EMNLP*, pages 1785–1794.

Wentau Yih, Mingwei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of ACL*, pages 1321–1331.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of AAAI*, pages 2852–2858.

Xiaofeng Yu and Wai Lam. 2010. Jointly identifying entities and extracting relations in encyclopedia text via a graphical model approach. In *Proceedings of COLING*, pages 1399–1407.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *J. Mach. Learn. Res.*, 3:1083–1106.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING*, pages 2335–2344.

Xiangrong Zeng, Shizhu He, Kang Liu, and Jun Zhao. 2018a. Large scaled relation extraction with reinforcement learning. In *Proceedings of AAAI*.

Xiangrong Zeng, Daojian Zeng, Shizhu He, Kang Liu, and Jun Zhao. 2018b. Extracting relational facts by an end-to-end neural model with copy mechanism. In *Proceedings of the ACL*.

Meishan Zhang, Yue Zhang, and Guohong Fu. 2017. End-to-end neural relation extraction with global optimization. In *Proceedings of EMNLP*, pages 1730–1740.

Suncong Zheng, Feng Wang, Hongyun Bao, Yuexing Hao, Peng Zhou, and Bo Xu. 2017. Joint extraction of entities and relations based on a novel tagging scheme. In *Proceedings of ACL*, pages 1227–1236.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of ACL*, pages 207–212.

## A The Details of the CNN Baseline

In this section, we will describe the details of the CNN baseline. This baseline is a pipeline method. For a sentence, we use the NLTK toolkit to recognize the entities first. Then, we combine each two entities as an entity pair and use a CNN relation classifier to predict their relation.

For example, suppose we recognize 3 entities in sentence $s$, which denoted as $e_1, e_2, e_3$. There are 6 different entity pairs (remind that $< e_1, e_2 >$ and $< e_2, e_1 >$ are different).

The CNN classifier is basically the same with Zeng et al. (2014). Each word is turned into a embedding which including it's word embedding and position embedding. After the convolution layer, we apply a maxpooling layer on it. Then we apply a two layer softmax classify layer to obtain the final results. We train the model with NLL loss.

Specifically, the word embedding dimension is 100, the position embedding dimension is 5, we use 128 filters and the filter size is 3. The hidden layer size of softmax classifier is 100 and we use tanh as the activation function. We optimize the model with Adam optimizer (Kingma and Ba, 2015).

During evaluation, if the entity pair is classified into *NA* relation, we will exclude this triplet. Otherwise, the triplet will be regarded as a predict triplet. If the predicted triplet is the same as one of the gold triples, it will be regarded as correct triplet. To be fair, when comparing the entities in the triplets, we only compare the last word of each entity. As long as the last word of the extract entity is the same as the gold one, we regard it as correct.

| Model | NYT | | | WebNLG | | |
|---|---|---|---|---|---|---|
| | Precision | Recall | F1 | Precision | Recall | F1 |
| ONE+NLL (Shuffle) | 0.617 | 0.471 | 0.534 | 0.360 | 0.214 | 0.268 |
| ONE+NLL (FixUnsort) | 0.597 | 0.511 | 0.551 | 0.311 | 0.294 | 0.302 |
| ONE+NLL (Alphabetical) | 0.570 | 0.526 | 0.547 | 0.310 | 0.292 | 0.300 |
| ONE+NLL (Frequency) | 0.609 | 0.531 | 0.567 | 0.311 | 0.302 | 0.306 |
| ONE+RL | 0.723 | 0.540 | 0.618 | 0.613 | 0.300 | 0.403 |
| MULTI+NLL (Shuffle) | 0.617 | 0.517 | 0.562 | 0.377 | 0.275 | 0.318 |
| MULTI+NLL (FixUnsort) | 0.586 | 0.544 | 0.564 | 0.404 | 0.403 | 0.403 |
| MULTI+NLL (Alphabetical) | 0.652 | 0.635 | 0.643 | 0.470 | 0.471 | 0.471 |
| MULTI+NLL (Frequency) | 0.636 | 0.593 | 0.614 | 0.500 | 0.499 | 0.500 |
| MULTI+RL | 0.738 | 0.636 | 0.683 | 0.615 | 0.478 | 0.538 |
| OUR+NLL (Shuffle) | 0.644 | 0.545 | 0.591 | 0.419 | 0.312 | 0.358 |
| OUR+NLL (FixUnsort) | 0.657 | 0.577 | 0.615 | 0.488 | 0.485 | 0.486 |
| OUR+NLL (Alphabetical) | 0.701 | **0.683** | 0.692 | 0.521 | 0.516 | 0.518 |
| OUR+NLL (Frequency) | 0.680 | 0.642 | 0.660 | 0.569 | **0.576** | 0.572 |
| OUR+RL | **0.770** | 0.671 | **0.717** | **0.628** | 0.555 | **0.589** |

Table 3: Results of different extraction order of models with GRU cell.

## B Results of GRU Cell

We show the results of different extraction order of models with GRU cell in Table 3.

# CaRe: Open Knowledge Graph Embeddings

**Swapnil Gupta**
Indian Institute of Science
Bangalore, India
swapnilgupta.229@gmail.com

**Sreyash Kenkre**
Walmart Labs
Bangalore, India
sreyash@gmail.com

**Partha Talukdar**
Indian Institute of Science
Bangalore, India
ppt@iisc.ac.in

## Abstract

Open Information Extraction (OpenIE) methods are effective at extracting (noun phrase, relation phrase, noun phrase) triples from text, e.g., *(Barack Obama, took birth in, Honolulu)*. Organization of such triples in the form of a graph with noun phrases (NPs) as nodes and relation phrases (RPs) as edges results in the construction of Open Knowledge Graphs (OpenKGs). In order to use such OpenKGs in downstream tasks, it is often desirable to learn embeddings of the NPs and RPs present in the graph. Even though several Knowledge Graph (KG) embedding methods have been recently proposed, all of those methods have targeted *Ontological KGs*, as opposed to OpenKGs. Straightforward application of existing Ontological KG embedding methods to OpenKGs is challenging, as unlike Ontological KGs, OpenKGs are not *canonicalized*, i.e., a real-world entity may be represented using multiple nodes in the OpenKG, with each node corresponding to a different NP referring to the entity. For example, nodes with labels *Barack Obama*, *Obama*, and *President Obama* may refer to the same real-world entity *Barack Obama*. Even though canonicalization of OpenKGs has received some attention lately, output of such methods has not been used to improve OpenKG embeddings. We fill this gap in the paper and propose Canonicalization-infused Representations (CaRe) for OpenKGs. Through extensive experiments, we observe that CaRe enables existing models to adapt to the challenges in OpenKGs and achieve substantial improvements for the link prediction task.

## 1 Introduction

Open Information Extraction (OpenIE) methods such as ReVerb (Fader et al., 2011), OLLIE (Mausam et al., 2012), BONIE (Saha et al., 2017) and CALMIE (Saha and Mausam, 2018) can automatically extract *(noun phrase, relation phrase, noun phrase)* triples from text, e.g., *(Barack Obama, took birth in, Honolulu), (Michelle Obama, wife of, Barack), etc*. An *Open Knowledge Graph* (*OpenKG*) can be constructed out of such triples by representing noun phrases (NPs) as nodes and relation phrases (RPs) as edges connecting them. Example of an OpenKG is shown in Figure 1. OpenKGs do not require pre-specified ontology, making them highly adaptable. In order to use OpenKGs in downstream tasks such as Question Answering, Document Classification, etc., it is often necessary to learn embeddings of NPs and RPs present as nodes and edges in an OpenKG.

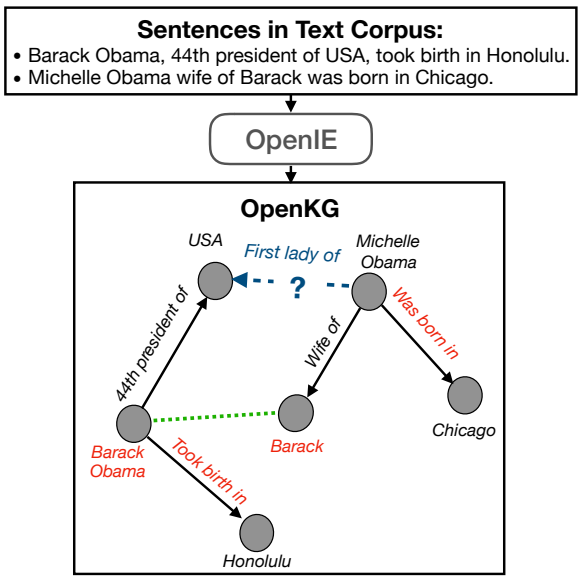


Figure 1: **Challenges in OpenKG**. KG embedding methods are effective in downstream tasks such as Link Prediction. Existing KG embedding methods are ineffective on OpenKGs as they are not canonicalized. CaRe can effectively utilize canonicalization information to learn better embeddings in OpenKGs. Green dotted line represents the missing NP canonicalization information. Please see Section 1 for more details.

Even though Knowledge Graph (KG) embed-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 378–388, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

ding has been an active area of research (Bordes et al., 2013; Yang et al., 2014), all the proposed KG embedding methods have focused on embedding *Ontological KGs*, such as WikiData (Vrandečić and Krötzsch, 2014), DBpedia (Auer et al., 2007), YAGO (Suchanek et al., 2007), NELL (Mitchell et al., 2018), and Freebase (Bollacker et al., 2008). Existing KG embedding models train representation of each node and edge label based on the context of triples they are present in. Doing this is suitable for ontological KGs as they are *canonicalized*. However, in OpenKGs, the same latent entity may be represented in different nodes labelled with different NPs. Similarly, the same latent relation can be represented with different RPs. For example, in Figure 1, *Barack Obama* the entity is represented using two NP nodes: *Barack Obama* and *Barack*. Similarly, the two RPs – *took birth in* and *was born in* – refer to the same underlying relation. Hence, the paradigm of learning embeddings for each node and edge label only from the context of the triples they appear in is ineffective for OpenKGs.

A possible solution is to canonicalize the OpenKGs. This involves identifying NPs and RPs that refer to the same entity and relation, and assigning them unique IDs. Nodes in the OpenKG having the same ID are *merged*, leading to a clean and canonicalized graph. Recent works that automatically canonicalize OpenKGs, including (Galárraga et al., 2014) and CESI (Vashishth et al., 2018), pose canonicalization as a clustering task of the NPs and RPs. However, due to automatic generation, the output clusters often contain some incorrectly canonicalized elements. Thus, directly merging nodes or relations with the same IDs would result in the propagation of errors in the canonicalization step to down-stream tasks.

Our premise is that the output of these automatic canonicalization models can be utilized to improve OpenKG embedding. Instead of explicitly merging nodes with common IDs, KG embedding models can be designed to judiciously account for mistakes during the canonicalization step. Towards establishing this premise, we propose a flexible OpenKG embedding approach to integrate and utilize the output of a canonicalization model in an error-conscious manner. Our contributions are as follows:

- We draw attention to an important but relatively unexplored problem of learning representations for OpenKGs.
- We propose Canonicalization-infused Representations (CaRe) for Open KGs - a novel approach to enrich OpenKG embedding models with the output of a canonicalization model. To the best of our knowledge, this is the first model of its kind.
- Through extensive experiments on real-world datasets, we establish CaRe's effectiveness in embedding OpenKGs.

CaRe source code is available at `https://github.com/malllabiisc/CaRE`.

## 2 Related Work

**OpenKG extraction and canonicalization**: Multiple OpenIE systems have been developed over the years. These include TextRunner (Yates et al., 2007), (Angeli et al., 2015), ReVerb (Fader et al., 2011), OLLIE (Mausam et al., 2012), SRLIE (Christensen et al., 2011), RelNoun (Pal and Mausam, 2016) and ClauseIE (Del Corro and Gemulla, 2013). A recent survey on the progress in OpenIE systems is presented in (Mausam, 2016).

To perform automatic canonicalization of OpenKGs, (Galárraga et al., 2014) first cluster NPs over manually defined feature spaces. These are then passed to AMIE (Galárraga et al., 2013) for RP clustering. Whereas, CESI (Vashishth et al., 2018) jointly learns vector representations of NPs and RPs by infusing side information in KG embedding models which are then used to cluster NPs and RPs. In this work, we use CESI to generate canonicalization clusters for our datasets.

**KG Embedding Methods**: KG embedding methods aim to learn low dimensional vector representations for the nodes and edge labels encoding the graph topology. All these methods train the embeddings by optimizing a link prediction based objective. They primarily differ in their way of mathematically modelling the likelihood of a triple being true. Translation-based hypothesis which regards relation vector for any *(subject, relation, object)* triple as a translation from the subject vector to the object vector is used in methods like TransE (Bordes et al., 2013) and TransH (Wang et al., 2014). Semantic matching models, such as DistMult (Yang et al., 2014), ComplEx (Trouillon et al., 2016) and HolE (Nickel et al., 2016), use

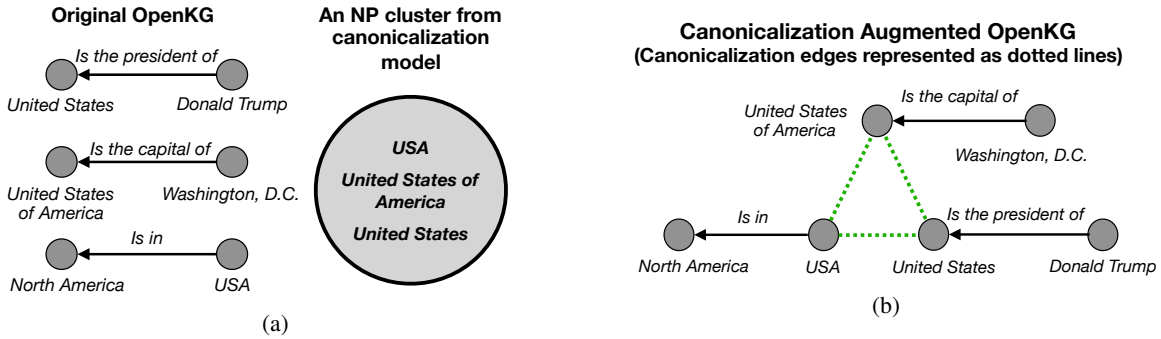


Figure 2: **CaRe Step 1:** First CaRe augments the original OpenKG with the output of a canonicalization model. (a) OpenKG and NP clusters from a canonicalization model. (b) Augmented OpenKG by adding undirected edges between canonical NPs (represented as dotted lines). Please refer to Section 4.2 for more details.

similarity-based scoring functions to measure the likelihood of a fact. Multi-Layer neural network models, ConvE (Dettmers et al., 2018) and R-GCN (Schlichtkrull et al., 2018) have shown better expressive strength. R-GCN adapts graph convolutional network (GCN) (Kipf and Welling, 2016) to a relational graph proposing an auto-encoder model for the link prediction task. Implicit in all these approaches is the assumption that each node in the graph is a different entity, and distinct edge labels refer to distinct relations. This assumption does not hold in OpenKGs. Existing KG embeddings are thus unsuitable for a task like link prediction on OpenKGs. CaRe addresses these limitations by infusing the output of a canonicalization model with the KG embedding models.

## 3 Background

In this section, we present a quick overview of a few basic methods which are useful for understanding the rest of the paper, especially the experiments section. We first start with the notations.

**Notations**: OpenKG is denoted as $\mathrm{G} = (\mathrm{N}, \mathrm{R}, \mathrm{T}^+)$, where N and R are the set of NPs and RPs, respectively, and $\mathrm{T}^+ = \{(s, r, o) | s \in \mathrm{N}, r \in \mathrm{R}, o \in \mathrm{N}\}$ is the set of observed triples. Here $s, o$ are the subject and object phrase, respectively.

**TransE:** Given a triple $(s, r, o)$, consider $e_s, r_r, e_o \in \mathbb{R}^d$ as the $d$-dimensional vector representations of subject $(s)$, relation $(r)$ and object $(o)$ respectively. TransE follows a translation based triple scoring function $\psi(.)$:

$$\psi(s, r, o) = -\left\| e_s + r_r - e_o \right\|_p .$$

Here, $\|.\|$ denotes norm and $p$ is either 1 or 2. For training parameters, TransE uses margin-based pairwise ranking loss with negative sampling.

**ConvE:** ConvE (Dettmers et al., 2018) first reshapes $e_s$ and $r_r$ to $\bar{e_s}$ and $\bar{r_r}$ , respectively, and passes them through a 2D convolution layer to compute the score corresponding to a triple:

$$\psi(s, r, o) = f(vec(f([\bar{e_s}; \bar{r_r}] * w))W)e_o.$$

Here, $*$ and $w$ denote the convolution operator and convolution filters, $f$ represents an activation function and $W$, the weights of the final linear layer. For training, ConvE uses binary cross-entropy loss with correct samples considered as positive instances while negative instances are generated through negative sampling.

**Graph Neural Networks (GNN):** GNNs were introduced in (Gori et al., 2005) and (Scarselli et al., 2009) as a generalization of recursive neural networks. Later, the generalization of CNN to graph-structured data, popularly known as Graph Convolution Network (GCN), was proposed in (Bruna et al., 2014). A first-order formulation of GCN as proposed in (Kipf and Welling, 2016). Under GCN formulation, the representation of a node $n$ after $l^{th}$ layer is defined as follows.

$$e_n^{l+1} = f\left(\sum_{i \in \mathcal{N}(n)} \left(W^l e_i^l + b^l\right)\right), \forall n \in \mathrm{N} \quad (1)$$

Here, $W^l, b^l$ are layer parameters, $\mathcal{N}(n)$ corresponds to the immediate neighborhood of $n$ and $f$ denotes an activation function.

**GAT:** GAT uses the attention mechanism to determine the weights of a node's neighbors (Veličković et al., 2018). Further, it uses multi-head attentions and defines the representation of

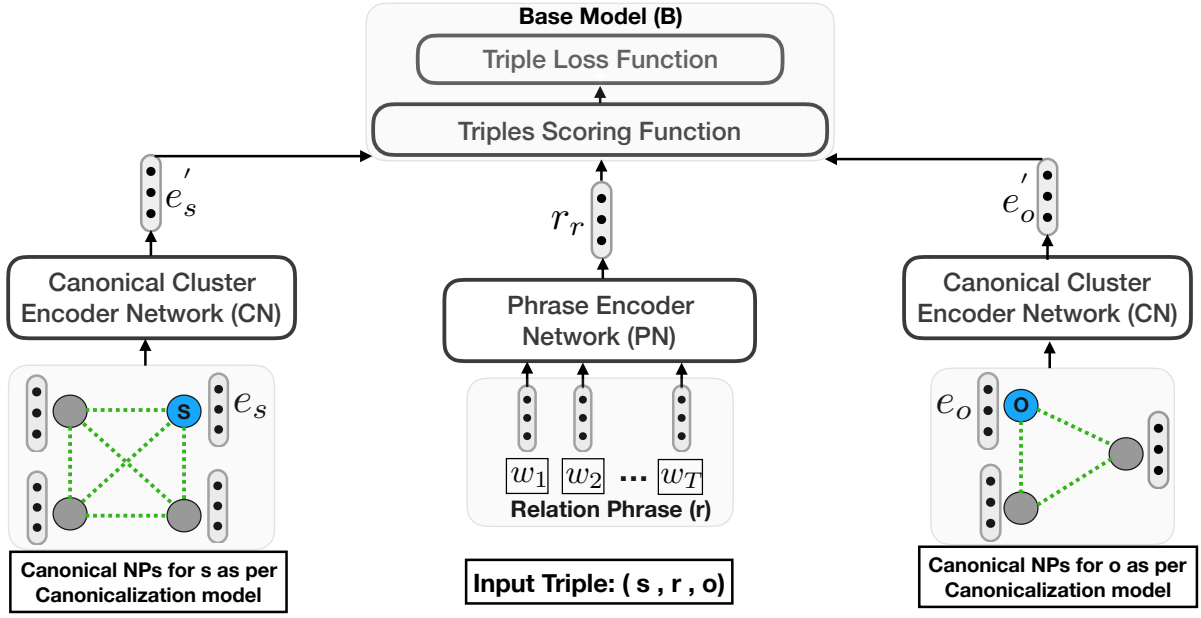


Figure 3: **CaRe Step 2:** In this step, CaRe learns KG embeddings from the augmented OpenKG (Figure 2). Base model can be any existing KG embedding model (e.g., TransE, ConvE). RP embeddings are parameterized by encoding vector representations of the word sequence composing them. This enables CaRe to capture semantic similarity of RPs. Embeddings of NPs are made more context rich by updating them with the represenations of canonical NPs (connected with dotted lines). Please see Section 3 and 4 for details.

$n^{th}$ the node as follows

$$e_n^{l+1} = ||_{k=1}^{K} f\left(\sum_{i \in \mathcal{N}(n)} \left(\alpha_k(e_n^l, e_i^l) W_k^l e_i^l\right)\right). \quad (2)$$

Here, $\alpha(.)$ denotes attention weights and $||$ denotes concatenation.

# 4 CaRe: Proposed Method

## 4.1 Overview

CaRe consists of two steps. We first give an overview of the two steps below and then provide a detailed description of each step.

- **Step 1**: In this step, the given OpenKG is augmented by adding edges from a canonicalization model. This step is outlined in Figure 2 and described in detail in Section 4.2.
- **Step 2**: In this step, embeddings of nodes and edges present in the augmented OpenKG obtained in Step 1 are learned. This step is outlined in Figure 3. The system consists of three components: Base Model, Phrase Encoder Network, and Canonical Cluster Encoder Network. We describe these components in Section 4.3, Section 4.4, and Section 4.5, respectively.

CaRe architecture and its components are depicted in Figure 3.

## 4.2 Step 1: Canonicalization Augmented OpenKG

As discussed in Section 1, based on the automatic clusters generated by a canonicalization model, merging NPs with a single ID can lead to the propagation of errors in the graph. Hence, we propose a soft integration scheme by adding an unlabeled and undirected edge between any two NPs which are canonical as per the canonicalization model as shown in Figure 2. More formally, suppose $n_1, n_2$ are two NPs that have been identified to be the same in the canonicalization step. We add an undirected and unlabelled edge $(n_1, n_2)$ between the nodes $n_1$ and $n_2$, and call them canonicalization edges. If the canonicalization step produces a confidence score for $n_1$ and $n_2$ being the same, then this score can be incorporated as a weight on the canonicalization edge. If no such score is produced, then the edges are kept unweighted. We then collect all these canonicalization edges into the set C. Finally, these edges are added to the original OpenKG $G$ to get a *canonicalization augmented OpenKG*, $\mathrm{G}' = (\mathrm{N}, \mathrm{R}, \mathrm{T}^{+} \cup \mathrm{C})$

### 4.3 Step 2: Base Model (B)

This component decides the way parameters are trained in CaRe based on the relational edges $T^+$. While CaRe is flexible to accommodate any KG embedding model as the base model, for the experiments in paper, we used TransE and ConvE (please see Section 3).

### 4.4 Step 2: Phrase Encoder Network (PN)

We generate embeddings of RPs by encoding the vector representations of the words composing them. Consider an RP as a sequence of T words $(w_1, w_2, ..., w_T)$ and their corresponding word vectors as $(x_1, x_2, ..., x_T)$. They are passed through a Phrase Encoder Network module (Figure 3) for which we use a bidirectional GRU (Cho et al., 2014) model with last pooling as described below.

$$(\overrightarrow{h_1}, \overrightarrow{h_2}, ..., \overrightarrow{h_T}) = \overrightarrow{GRU}(x_1, x_2, ..., x_T)$$
$$(\overleftarrow{h_1}, \overleftarrow{h_2}, ..., \overleftarrow{h_T}) = \overleftarrow{GRU}(x_1, x_2, ..., x_T).$$

Finally, $r_r = [\overrightarrow{h_T} : \overleftarrow{h_1}]$ is concatenation of final hidden states in both the directions. This approach allows parameter sharing across RPs with word overlaps while leveraging the rich semantic information from pre-trained word embeddings.

### 4.5 Step 2: Canonical Cluster Encoder Network (CN)

The canonicalization information is incorporated in the NP embeddings by utilizing the canonicalization-induced edges C.

Each NP $n \in$ N is assigned a vector $e_n$. As NP canonicalization is expressed as a clustering step, for each NP, its canonical NPs are a single edge away in C (see Figure 3). Hence, a single layer of network aggregation is sufficient. We propose a non-parametric message passing and update network which works in the following two steps:

**Context vector:** First for each NP, $n \in$ N a context vector $e_n^c$ is generated by the following message passing scheme from its canonical neighbors.

$$e_n^c = \left( \sum_{i \in \mathcal{N}(n)} \frac{1}{|\mathcal{N}(n)|} e_i \right), \forall n \in \text{N}.$$

Here, $\mathcal{N}(n) = \{i | i \in \text{N}, (n, i) \in \text{C}\}$, is the canonical neighborhood of $n$.

**Updating NP embeddings:** The updated embeddings for each NP is computed as below:

$$e_n^{'} = \frac{e_n}{2} + \frac{e_n^c}{2}$$

which are passed to the decoding stage. For the case of weighted canonicalization edges, the weights can be used while generating the context vector of an NP, by making the contribution of its canonical NPs proportional to the edge weights joining them. The proposed approach can be interpreted as a *local embedding smoothening* of canonical NPs and we call this network as Local Averaging Network **(LAN)**. We also experimented with more sophisticated attention-based context vector generation and gating mechanism for embedding update stage, but they resulted in a performance drop.

Note that, our approach to embed NPs is inspired by the auto-encoder framework used in R-GCN (Schlichtkrull et al., 2018) but with a key difference. In R-GCN, the same set of edges are utilized at both the encoding and decoding stages. Whereas in CaRe, the canonicalization edges C are used at the encoder (Canonical Cluster Encoder Network) while the decoder (Base Model) operates on the relational edges $T^+$.

**CaRe nomenclature:** We introduce a generic nomenclature for the possible variants of the CaRe framework based on the choice of the Base Model (B), Phrase Encoder Network (PN) and Canonical Cluster Encoder Network (CN) as **CaRe(B, PN, CN)**. For e.g., CaRe(B=ConvE, PN=Bi-GRU, CN=LAN) corresponds to a CaRe model with ConvE as base model, Bi-GRU network with last pooling (Section 4.4) as the Phrase Encoder Network and Local Averaging Network (Section 4.5) as the Canonical Cluster Encoder Network. We define Bi-GRU and LAN as default values for the PN and CN arguments respectively. In case, values for these arguments are not explicitly mentioned, they take the default values. Thus, CaRe(B=ConvE) represents the same network as CaRe(B=ConvE, PN=Bi-GRU, CN=LAN).

## 5 Experiments

### 5.1 Datasets

Statistics of the two datasets used in the experiments of this paper are summarized in Table 1. In order to build these datasets, we first obtained the three graph datasets – Base, Ambiguous and ReVerb45K. While Base and Ambiguous datasets are created in (Galárraga et al., 2014), ReVerb45K is

| Datasets | ReVerb45K | ReVerb20K |
|---|---|---|
| # NPs | 27K | 11.1K |
| # RPs | 21.6K | 11.1K |
| # Gold NP Clusters | 18.6K | 10.8K |
| # Train Triples | 36K | 15.5K |
| # Test Triples | 5.4K | 2.4K |
| # Validation Triples | 3.6K | 1.6K |

Table 1: Details of datasets used. Please see Section 5.1 for details.

introduced in (Vashishth et al., 2018). ReVerb20K is created by combining the two smaller datasets Base and Ambiguous. All the datasets are constructed through ReVerb Open KB (Fader et al., 2011). We refer the readers to the respective papers for the construction details.

To generate the train, validation and test triples the following procedure is adopted. First, the entire set of triples is divided in $80 : 20$ ratio ensuring that each NP and RP in the triples of the smaller set is at least present once in the triples of the bigger set. This consideration is essential because of the transductive nature of existing KG embedding models. The bigger set is considered as the train dataset and the smaller set is further randomly divided in $30 : 70$ to get the validation and test datasets respectively. Both datasets contain gold canonicalization clusters for the NPs extracted through the Freebase entity linking information (Gabrilovich et al., 2013) which enables automatic evaluation in the canonicalization task.

### 5.2 Open KG Link Prediction Evaluation

In the typical link prediction evaluation, an unseen triple $(s, r, o)$ is taken and partial triples $(s, r, ?)$ and $(?, r, o)$ are shown to the model. It ranks all the entities in the graph for their likelihood to be the missing entity and the rank assigned to the true missing entity is considered.

However, while this is suitable for ontological KGs, it is not valid for our setting. In OpenKGs, instead of entities, NPs are present and several of them can refer to the same entity. This means that, even when predicting correct entity, a model will be unfairly penalized if the prediction is a different canonical form of the entity than the one present in the considered triple. We, therefore propose to rank gold NP clusters, available in the dataset, instead of ranking each NP. We do this in the following manner:

- List all the NPs in decreasing order of their likelihood to be the missing part of the triple.
- Prune this list by keeping the best ranked NPs for each gold cluster. This gives us the ranked list of gold clusters.
- Consider the rank of the cluster to which the true missing NP belongs.

We provide results using three commonly used evaluation metrics: *mean rank* (MR), *mean reciprocal rank* (MRR) and *Hits@n* with $n = \{10, 30, 50\}$. Filtered setting as introduced in (Bordes et al., 2013) is followed.

### 5.3 Experimental Setup

We run CESI on both the datasets to generate the NP canonical clusters. Note that, none of the existing automatic canonicalization models output edge weights. Hence, for all the experiments, the canonicalization-induced edges are kept unweighted. For Phrase Encoder Network, we found single layer in the Bi-GRU model works best. CaRe allows the use of any pre-trained embeddings. However, (Vashishth et al., 2018) demonstrates the effectiveness of pre-trained GloVE (Pennington et al., 2014) vectors for initializing representations for the same datasets. Hence, the word vectors are initialized with 300-dimensional pre-trained GloVE embeddings and are kept trainable. We use PyTorch Geometric library (Fey and Lenssen, 2019) for the Canonical Cluster Encoder Network module.

The choice of optimizer and regularization based hyper-parameters is directly adopted from the ones proposed in the original work of the base models. Both the NP and RP embedding size is kept fixed at 300, while the learning rate is selected through a grid search over $\{0.1, 0.01, 0.001\}$.

Models like ConvE introduce inverse relations. For our experiments we generate inverse phrase for an RP by adding a phrase *"inverse of"* to that RP. In ComplEx the embeddings have both real and imaginary parts. For this, we use two separate Phrase Encoder Network and Graph Neural Network modules for the real and imaginary parts of RP and NP embeddings respectively.

## 6 Results

In this section we evaluate the following questions:

| Method | ReVerb45K | | | | | ReVerb20K | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | MR | MRR | Hits@10 | Hits@30 | Hits@50 | MR | MRR | Hits@10 | Hits@30 | Hits@50 |
| TransE | 2955.8 | .193 | .361 | .446 | .478 | 1425.8 | .126 | .299 | .411 | .468 |
| TransH | 2998.2 | .194 | .362 | .442 | .478 | 1464.4 | .129 | .303 | .409 | .467 |
| DistMult | 8988.8 | .051 | .051 | .052 | .065 | 6260.0 | .033 | .044 | .055 | .060 |
| ComlEx | 7786.5 | .047 | .047 | .048 | .073 | 5502.2 | .037 | .058 | .075 | .085 |
| R-GCN | 2866.8 | .042 | .046 | .091 | .113 | 1204.3 | .122 | .187 | .263 | .305 |
| ConvE | 2650.8 | .233 | .338 | .401 | .429 | 1014.5 | .294 | .402 | .491 | .541 |
| **CaRe(B=ConvE)** | **1308.0** | **.324** | **.456** | **.543** | **.579** | **973.2** | **.318** | **.439** | **.525** | **.566** |

Table 2: **Link Prediction results**. CaRe(B=ConvE) substantially outperforms all the existing KG embedding models. For all the experiments, evaluation strategy described in Section 5.2 is followed. B, CN, PN are the arguments of CaRe framework (Section 4 and Figure 3). For the reported results: PN=Bi-GRU and CN=LAN.

| Method | ReVerb45K | | | | | ReVerb20K | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | MR | MRR | Hits@10 | Hits@30 | Hits@50 | MR | MRR | Hits@10 | Hits@30 | Hits@50 |
| TransE | 2955.8 | .193 | .361 | .446 | .478 | 1425.8 | .126 | .299 | .411 | .468 |
| CaRe(B=TransE, CN=$\phi$) | 2522.3 | .195 | .378 | .457 | .488 | 978.2 | .286 | .411 | .515 | .565 |
| ComplEx | 7786.5 | .047 | .047 | .048 | .073 | 5502.2 | .037 | .058 | .075 | .085 |
| CaRe(B=ComplEx, CN=$\phi$) | 5205.4 | .185 | .225 | .235 | .325 | 3010.0 | .216 | .288 | .345 | .376 |
| R-GCN | 2866.8 | .042 | .046 | .091 | .113 | 1204.3 | .122 | .187 | .263 | .305 |
| CaRe(B=R-GCN, CN=$\phi$) | 2508.1 | .145 | .203 | .260 | .305 | 1210.3 | .195 | .275 | .340 | .370 |
| ConvE | 2650.8 | .233 | .338 | .401 | .429 | 1014.5 | .294 | .402 | .491 | .541 |
| CaRe(B=ConvE, CN=$\phi$) | 1656.1 | .293 | .401 | .477 | .509 | 966.9 | .307 | .419 | .514 | .556 |

Table 3: Impact of parameterizing RP embeddings. B, CN, PN are the arguments of CaRe framework (Section 4 and Figure 3). $\phi$ value for CN argument implies that Canonical Cluster Encoder Network module is not used in these experiments. PN=Bi-GRU in all these experiments. Please refer to Section 6.2.

Q1. Is CaRe effective for the link prediction task in OpenKGs? (Section 6.1)

Q2. What is the quantitave and qualitative impact of parameterizing RP embeddings in CaRe? (Section 6.2 and Section 6.4)

Q3. How does the local embedding smoothening approach adopted in CaRe compare against other competitive baselines? (Section 6.3)

### 6.1 Overall Performance

As shown in Figure 3, any existing KG embedding model can be used as the base model in the CaRe framework. We experimented with several KG embedding models and found the CaRe achieved substantial improvements in comparison to standalone use of the KG embedding model in each case. We achieved overall best performance working with ConvE as the base model, CaRe(B=ConvE). The experimental results are presented in Table 2.

Table 1 shows that in both the datasets, on an average, the number of train triples for each NP and RP is less than 2. In contrast, FB15k (Bordes et al., 2013), an ontological KG, has on an average 32 triples for each entity and 360 triples for each relation. This highlights the extremely sparse and fragmented nature of OpenKGs. Hence, the superior performance of CaRe supports the hypothesis that the flow of information while learning the representations of canonical NPs and RPs in OpenKGs is beneficial.

Also, a comparison of the number of unique NPs and gold NP clusters for the two datasets (Table 1) shows that the number of NPs canonical to each other is more significant in ReVerb45K than in ReVerb20K. Hence, a more prominent improvement due to CaRe in ReVerb45K as compared to ReVerb20K proves the effectiveness of CaRe in utilizing the canonicalization information.

### 6.2 Impact of parameterizing RP embeddings

As described in Section 4.4, a Phrase Encoder Network is used to parametrize RP embeddings, which allows parameter sharing while learning embeddings. Table 3 provides a quantitative anal-

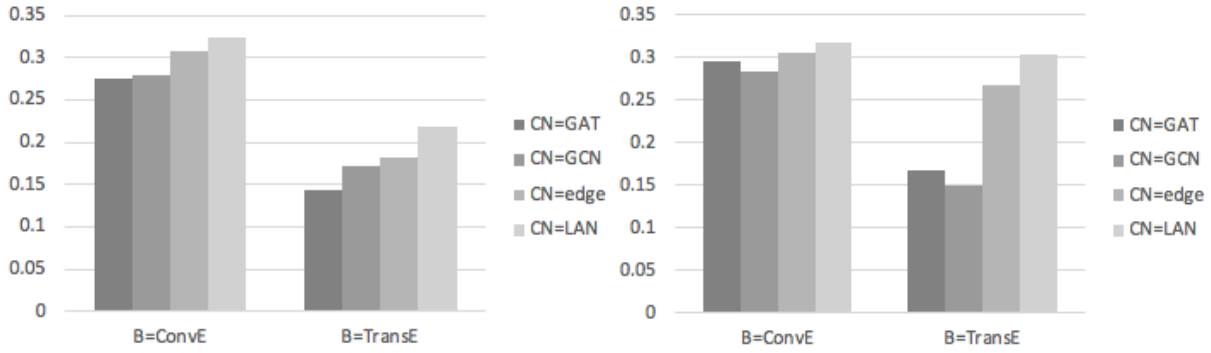


(a) ReVerb45K (b) ReVerb20K

Figure 4: Performance comparison of CaRe framework for diffrent values of the CN (as described in Section 6.3). B=ConvE (left group) and B=TransE (right group) in both the plots. PN=Bi-GRU. **Mean Reciprocal Rank (MRR)** is plotted on the y axis (higher is better).

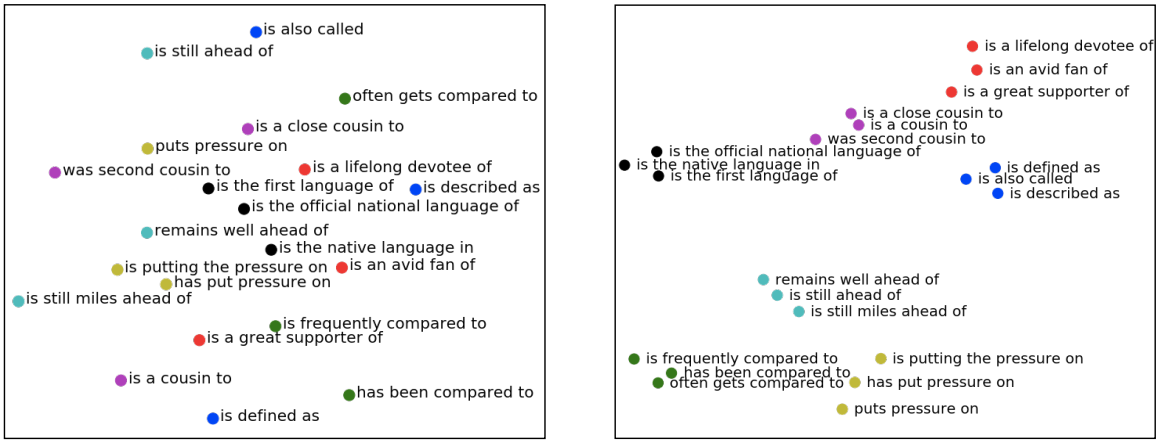


(a) ConvE (b) CaRe(B=ConvE)

Figure 5: **t-SNE visualization of RP embeddings**. RP embeddings learned by CaRe(B=ConvE) are able to capture the semantic similarity of the RPs whereas in ConvE this information is lost. Please refer to Section 6.4.

ysis of the impact of parameterizing RP embeddings across several KG embedding models. In these experiments, the CN argument of CaReis kept $\phi$, which implies no canonicalization information is used. The NP embeddings are trained in the same manner as the base KG embedding model.

Comparing the performance of CaRe(B=ConvE,CN=$\phi$) in Table 3 with the performance of CaRe(B=ConvE) in Table 2, it can be noticed that a major part of the performance boost of CaRe can be attributed to parameterization of RP embeddings.

### 6.3 Different ways to utilize Canonicalization edges

There can be several methods through which the canonicalization edges integrated into the graph (as described in Figure 2), can be utilized in the model. In Section 4.5, we described a local embedding smoothening method adopted in CaRe. In this section, we present a comparative analysis with the following competitive baselines:

- **CaRe(CN=GCN)**: A single layer of GCN (Kipf and Welling, 2016) for the Canonical Cluster Encoder Network. Refer to Equation 1.
- **CaRe(CN=GAT)**: A single layer of GAT (Veličković et al., 2018) for the Canonical Cluster Encoder Network. Refer to Equation 2.
- **CaRe(CN=edge)**: Here the canonicalization edges in $G'$ are labelled with a symmetric RP *"is canonical to"*. This adds a new edge type in the graph and is treated by KG embedding models like any other edge type.

We show these comparisons using both TransE and ConvE as base models in Figure 4. The GCN and GAT architectures have an adverse effect on the performance. We believe this can be attributed to the complex nature of these architectures in an already over-parameterized and noisy setting. In CaRe(CN=edge), the model is expected to learn the meaning of two NPs being canonical and encode it in the vector embedding of the new edge type. The local averaging network CaRe(CN=LAN) used in CaRe is a GNN architecture with fixed weights. The choice of weights follows from the prior belief that canonical NP embeddings should be close in the vector space, inducing a useful inductive bias. The above results indicate that this leads to better performance. Additionally, unlike the GNN based methods, CaRe(CN=edge) has a limitation in its modelling capacity as it provides no way to handle the case where canonicalization edges are weighted.

### 6.4 Qualitative Analysis

Figure 5 demonstrates a qualitative comparison of RP embeddings between ConvE and CaRe(B=ConvE). For this experiment, we selected seven RPs, and for each RP, through human judgement, we selected two more RPs with similar meaning. Thus, there are seven different clusters. The figure shows t-SNE (van der Maaten and Hinton, 2008) visualization of the embeddings learnt for these RPs by CaRe(B=ConvE) and ConvE. t-SNE is a non-linear transformation which tries to map points in high dimensional space to a lower dimension preserving the local relationships between the points. The figure verifies the hypothesis that due to the parameterization of RP embeddings and utilizing pre-trained word embeddings, CaRe is able to better capture the semantic similarity of the RPs in comparison to the base models. Due to the explicit integration of NP canonicalization in CaRe, we observed a similar desirable impact on the embeddings learned for NPs as well.

## 7 Conclusion

Open Information Extraction (OpenIE) provides an effective way to bootstrap Open Knowledge Graphs (OpenKGs) from text corpus. OpenKGs consist of noun phrases (NPs) as nodes and relations phrases (RPs) as edges. In spite of this advantage, OpenKGs are often sparse and non-canonicalized, i.e., the same entity could be expressed using multiple nodes in the graph (and similarly for relations). This renders existing Ontological KG embedding methods ineffective in learning embeddings of NPs and RPs in OpenKGs. In spite of this limitation, there has been no prior work which has focused on OpenKG embedding. We fill this gap in the paper and propose CaRe. CaRe infuses canonicalization information combined with the neighborhood graph structure to learn rich representations of NPs. Further, it captures the semantic similarity of RPs by utilizing the word sequence information in these relation phrases to parameterize the RP embeddings. Through extensive experiments on real-world datasets, we demonstrate the effectiveness of embeddings learned by CaRe.

As part of future work, we hope to extend CaRe to also utilize RP canonicalization information. Utilizing OpenKG embeddings in tasks beyond link prediction is another avenue of further work.

## Acknowledgments

We thank the anonymous reviewers for their constructive comments. This work is supported by the Ministry of Human Resource Development (MHRD), Government of India. Finally, we thank all the members of MALL Lab, IISc for their invaluable suggestions.

## References

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 344–354, Beijing, China. Association for Computational Linguistics.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The Semantic Web*, pages 722–735, Berlin, Heidelberg. Springer Berlin Heidelberg.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 1247–1250, New York, NY, USA. ACM.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 2787–2795, USA. Curran Associates Inc.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann Lecun. 2014. Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representations (ICLR2014), CBLS, April 2014*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Janara Christensen, Mausam, Stephen Soderland, and Oren Etzioni. 2011. An analysis of open information extraction based on semantic role labeling. In *Proceedings of the Sixth International Conference on Knowledge Capture*, K-CAP '11, pages 113–120, New York, NY, USA. ACM.

Luciano Del Corro and Rainer Gemulla. 2013. Clausie: Clause-based open information extraction. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 355–366, New York, NY, USA. ACM.

Tim Dettmers, Minervini Pasquale, Stenetorp Pontus, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the 32th AAAI Conference on Artificial Intelligence*, pages 1811–1818.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Matthias Fey and Jan E. Lenssen. 2019. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. 2013. Facc1: Freebase annotation of clueweb corpora, version 1 (release date 2013-06-26, format version 1, correction level 0). *Note: http://lemurproject.org/clueweb09/FACC1/Cited by*, 5.

Luis Galárraga, Geremy Heitz, Kevin Murphy, and Fabian M Suchanek. 2014. Canonicalizing open knowledge bases. In *Proceedings of the 23rd acm international conference on conference on information and knowledge management*, pages 1679–1688. ACM.

Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. 2013. Amie: Association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 413–422, New York, NY, USA. ACM.

Marco Gori, Gabriele Monfardini, and Franco Scarselli. 2005. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734. IEEE.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605.

Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 523–534, Jeju Island, Korea. Association for Computational Linguistics.

Mausam Mausam. 2016. Open information extraction systems and downstream applications. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 4074–4077. AAAI Press.

T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, B. Yang, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. 2018. Never-ending learning. *Commun. ACM*, 61(5):103–115.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2016. Holographic embeddings of knowledge graphs. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 1955–1961. AAAI Press.

Harinder Pal and Mausam. 2016. Demonyms and compound relational nouns in nominal open IE. In *Proceedings of the 5th Workshop on Automated Knowledge Base Construction*, pages 35–39, San Diego, CA. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language*

*Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Swarnadeep Saha and Mausam. 2018. Open information extraction from conjunctive sentences. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2288–2299, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Swarnadeep Saha, Harinder Pal, and Mausam. 2017. Bootstrapping for numerical open IE. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 317–323, Vancouver, Canada. Association for Computational Linguistics.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *ESWC*, volume 10843 of *Lecture Notes in Computer Science*, pages 593–607. Springer.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, pages 2071–2080.

Shikhar Vashishth, Prince Jain, and Partha Talukdar. 2018. Cesi: Canonicalizing open knowledge bases using embeddings and side information. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 1317–1327. International World Wide Web Conferences Steering Committee.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. *International Conference on Learning Representations*. Accepted as poster.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Twenty-Eighth AAAI conference on artificial intelligence*.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*.

Alexander Yates, Michael Cafarella, Michele Banko, Oren Etzioni, Matthew Broadhead, and Stephen Soderland. 2007. Textrunner: open information extraction on the web. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 25–26. Association for Computational Linguistics.

# Self-Attention Enhanced CNNs and Collaborative Curriculum Learning for Distantly Supervised Relation Extraction

**Yuyun Huang**[†] **Jinhua Du**[‡§]
[†]University College Dublin, Ireland
[‡]Investments AI, AIG (American International Group, Inc.)
[§]ADAPT Centre, School of Computing, Dublin City University, Ireland
{yuyun.huang}@ucd.ie
{jinhua.du}@aig.com

## Abstract

Distance supervision is widely used in relation extraction tasks, particularly when large-scale manual annotations are virtually impossible to conduct. Although Distantly Supervised Relation Extraction (DSRE) benefits from automatic labelling, it suffers from serious mislabelling issues, i.e. some or all of the instances for an entity pair (head and tail entities) do not express the labelled relation. In this paper, we propose a novel model that employs a collaborative curriculum learning framework to reduce the effects of mislabelled data. Specifically, we firstly propose an internal self-attention mechanism between the convolution operations in convolutional neural networks (CNNs) to learn a better sentence representation from the noisy inputs. Then we define two sentence selection models as two relation extractors in order to collaboratively learn and regularise each other under a curriculum scheme to alleviate noisy effects, where the curriculum could be constructed by conflicts or small loss. Finally, experiments are conducted on a widely-used public dataset and the results indicate that the proposed model significantly outperforms baselines including the state-of-the-art in terms of P@N and PR curve metrics, thus evidencing its capability of reducing noisy effects for DSRE.

## 1 Introduction

Relation Extraction (RE) is vital for NLP tasks such as information extraction, question answering and knowledge base completion. RE aims to identify the relationship between an entity pair $(e_1, e_2)$ in a sentence. For example, in the sentence "$[Bill\_Gates_{e_1}]$ is the principal founder of $[Microsoft_{e_2}]$", the relation extractor decodes the relation of *founder* for the entity pair *Bill_Gates* (the person) and *Microsoft* (the company).

Recent supervised relation extraction research can be roughly categorised into two areas: fully supervised and distantly supervised relation extraction. Fully supervised relation extraction mainly depends on manually annotated training dataset (Zeng et al., 2014; dos Santos et al., 2015). Ordinarily, human annotation on large-scale datasets is costly and often practicably impossible. Distance supervision addresses this problem by using an existing knowledge base (e.g. DBpedia) to automatically annotating large-scale datasets, thus reducing the burden of manual annotation (Mintz et al., 2009; Hoffmann et al., 2011; Surdeanu et al., 2012). However, distance supervision often suffers from mislabelling problems.

Figure 1 illustrates this incorrect labelling issue, which shows that not all sentences in a bag with the same entity pair express the labelled relation of *person/company/founder*. The worst case is that all sentences in a bag are mislabelled. Thus, one primary challenge in DSRE is to minimize the noisy labelling effects, which in turn, would let model learn from incorrect labelled datasets.

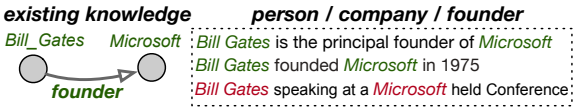


Figure 1: Mislabeling issue example in DSRE: the entity pairs bag containing three sentences is labeled as *person/company/founder*. However, the last sentence marked in red has no extractable pre-defined relation.

In order to design a solution to mitigate the effects of noisy data, we can treat our DSRE model learning procedure as analogous to *two students* training with a *curriculum* to answer a list of multiple-choice questions, where some difficult multiple-choice questions may have no correct answers (false positive). We base our proposed solution on the intuition that two students will compare and rethink their different answers during the learning process, thus regularising each



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 389–398, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

other and improving their final grades. We call this process of two students learning together with a curriculum as *Collaborative Curriculum Learning* (CCL), where each student represents a network.

Inspired by the above intuition, we propose to use a bag-level selective sentence attention model (NetAtt) (Lin et al., 2016) and a maximum probability sentence model (NetMax) (Zeng et al., 2015) as two students learning collaboratively. Meanwhile, highly organised human education methods using a multilevel curriculum, from easy to hard, is an analogy for curriculum learning training, which advocates the adoption of similar multi-stage strategies (Bengio et al., 2009). The curriculum learning training method for DSRE is used with the assumption that entity pair bags contain corrupted labelled sentences, which are difficult components to learn in the curriculum. By disregarding the effects of noisy samples (meaningless knowledge) during the training, the expectation is to boost the model's learning capability. Moreover, in order to accurately obtain the semantic representation of each sentence for our curriculum learning approach, we propose an internal CNNs self-attention mechanism to learn a better sentence representation in the DSRE setting.

The main contributions are summarised as:

(1) We make the first attempt to use the concept of curriculum learning for denoising DSRE and present a novel collaborative curriculum learning model to alleviate the effects of noisy sentences in an entity pair bag. In this model, we define two collaborative relation extractors to regularize each other and boost the model's learning capability.

(2) We propose conflicts and small loss tricks for our collaborative curriculum learning. Instead of using a separated complex noisy sentence filter and two-step training in baseline models, our model can alleviate noise effects during a single training and is easy to implement.

(3) We are the first to apply an internal CNNs self-attention mechanism to enhance a multilayer CNNs model for DSRE.

(4) We conduct thorough experiments on the widely-used NYT dataset, and achieve significant improvements over state-of-the-art models.

## 2 Related Work

Most DSRE approaches fall under the framework of Multi-Instance Learning (MIL) (Riedel et al., 2010; Surdeanu et al., 2012; Zeng et al., 2015; Lin et al., 2016; Ji et al., 2017; Qin et al., 2018; Feng et al., 2018). At the encoding step, a sentence representation is learned using handcrafted features or neural network models. Afterwards, in the sentence selection step, one or several sentences from an entity pair bag are chosen for further bag representation learning. Previously, statistical models (Mintz et al., 2009; Hoffmann et al., 2011; Surdeanu et al., 2012) have used designed features, such as syntactic and lexical features, and have then been trained by logistic regression or expectation maximization.

When adopting deep learning approaches, a single layer CNNs based model (Zeng et al., 2014) was exploited to extract sentence level features to attain fully supervised relation classification. For DSRE, Zeng et al. (2015) proposed an extended Piece-wise CNN (PCNN) approach and selected the most probable valid sentence to represent an entity pair bag, while the remaining sentences in the bag were ignored. Lin et al. (2016) and Ji et al. (2017) used all the sentences in a bag by assigning higher weights to valid labeled sentences and lower weights to noisy sentences. The selective sentence attention mechanism combined all weighted sentences as a bag representation. In addition, Ji et al. (2017) made use of entity description background knowledge and fused the external information into their PCNN-based model.

The self-attention mechanism (Cheng et al., 2016; Parikh et al., 2016; Vaswani et al., 2017), also called intra-attention, relates to different positions of a single sequence to learn the sequence representation. An internal CNNs states self-attention approach was proposed by Zhang et al. (2018) to improve Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) performance in generating high-quality images. Alt et al. (2019) extended Generative Pre-trained Transformer (GPT) to learn semantic and syntactic features for DSRE. Wang et al. (2019) used pre-trained Transformer for multiple entity-relations extraction task. Du et al. (2018) utilized self-attention mechanisms for better MIL sentence-level and bag-level representations. However, previous work has not considered to use self-attention over the internal CNNs model states for DSRE.

Deeper CNNs have positive effects on noisy NLP tasks (Conneau et al., 2017). Huang and Wang (2017) used residual learning for multilayer deep CNNs to improve DSRE performance.

To better address issues relating to mislabelling, e.g., when *some or all sentence labels* in a bag are falsely positive, schemes to filter out noisy instances have been developed. Takamatsu et al. (2012) proposed a wrong label sentence filter method using linguistic features. Feng et al. (2018) proposed a model comprising of a relation classifier and an instance selector based on reinforcement learning (RL). The instance selector was designed to select possible correctly labelled sentences and was regularised by the rewards from the relation classifier. Qin et al. (2018) used the idea of GANs to develop a separated correct label indicator, which filters high confidence scoring instances for training on existing PCNN/CNN-based relation extractors (DSGAN). *Unlike previous work*, which filters out incorrectly labelled *sentences* to generate an approximate clean training dataset and then retrains on the filtered data to improve models, we instead train our model to actively and purposefully forget noisy entity pair *bags*, based on a collaborative curriculum learning strategy in a single training process. In doing so, we develop an approach to building the curriculum – the identified disagreements and losses of two collaborative student networks in our model.

## 3 Methodology

The mislabelled sentences from the distance supervision method are normally regarded as unwanted noise that can reduce the performance of relation extraction. To alleviate noisy effects, we propose a collaborative curriculum learning framework for DSRE. The *architecture* is shown in Figure 3, consisting of three main components: (1) input representation; (2) CNNs that are composed of convolution, self-attention and pooling; and (3) collaborative curriculum learning module.

### 3.1 Inputs: Word and Position Embeddings

To represent each input token, we use word2vec[1] (Mikolov et al., 2013) to obtain its embedding. Each word embedding ($w_t$) contains syntactic and semantic information.




Figure 2: relative distances from 'founder' to entities

Similar to (Zeng et al., 2015), we use position embedding to assist the CNNs in measuring distances from the current word to the head and tail entities. As illustrated in Figure 2, in the sentence, the distance of the word *founder* to the head entity is 4 and $-2$ to the tail entity.

Figure 3(a) further illustrates the use of these embeddings as input representation in CNNs, i.e. the input representation is a concatenation of *word* embedding and *position* embedding. In Figure 3(a), it is assumed that the dimensions of word embedding $d_w$ and position embedding $d_p$ are 3 and 1 (as a simplified example for the sake of the figure), respectively. The total vector representation dimension $d$ is $d_w + 2 \times d_p$.

### 3.2 Contextualised Representation: Self-attention for CNNs

The sentence embedding matrix is formed by concatenating every vector representation horizontally (left panel Figure 3) and is represented as $s_n = [w_1 : \cdots : w_t : \cdots : w_T]$, where $s_n$ is the input that feeds CNNs to learn a sentence representation. For an input sentence, we use a convolution filter $W_p$ to slide along $s_n$ as $[w_t : \cdots : w_{t+u-1}] * W_p + b$, where $*$ is the convolution symbol, $b \in \mathbb{R}$ is a bias and $p$ represents the $p^{th}$ filter in a filter set. $W_p \subset \mathbb{R}^{u \times d}$, where $u$ is the length of the filter and $d$ is the dimension of a word vector consisting of word and position embeddings.

In order to learn a better sentence representation, we propose a self-attention mechanism which is performed directly over internal CNNs states as shown in Figure 3(a). The self-attention function maps queries (Q) and corresponding keys (K) to compute a weight map. The output is a multiplication of the values (V) and the weight vector. We place this internal CNNs self-attention module after the first convolution state ($C$). Values, queries, and keys are computed by applying convolution again on $C$, where $V = C * W^V$, $Q = C * W^Q$ and $K = C * W^K$. The attention map, is calculated using the softmax function, as shown in Equation (1), where $Q^T$ is the transpose of $Q$ and $\otimes$ is the matrix multiplication operator.

$$map = \frac{\exp(Q^T \otimes K)}{\sum \exp(Q^T \otimes K)} \quad (1)$$

Subsequently, the weighted value is computed as $\tau V \otimes map$. $\tilde{C} = cov(C + \tau V \otimes map)$ is then fed into a piece-wise max pooling layer. Where, $\tau$ learns gradually from 0 to 1 to assign more weight to the model.

[1] https://code.google.com/p/word2vec/

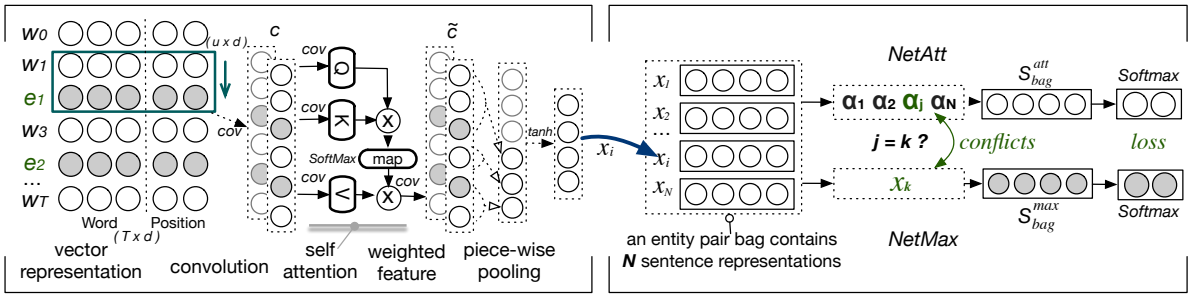


*(a) Self-Attention Piece-wise pooling CNN* *(b) Collaborative Learning with Conflicts and Loss*

Figure 3: ***(a)*** A sentence embedding matrix with the entity pair $(e_1, e_2)$ consists of a word vector set of dimension $(T \times d)$. $T$ is the length of the given sentence and $d$ is the length of the word and position vector. A convolution filter with a dimension of $(u \times d)$ is sliding along the sentence representation. $cov$ represents a convolution operation. The internal CNNs states $(C)$ from the first convolution operation are fed into the self-attention module. A piece-wise max-pooling is employed at outputs $(\tilde{C})$ from last convolution layer. A sentence representation $x_i$ is learned after a nonlinear function. ***(b)*** For an entity pair bag $B$ containing $N$ sentence representations, the $j^{th}$ sentence with maximum weight score $\alpha_j$ is selected by NetAtt, while $x_k$ is the $k^{th}$ sentence selected by NetMax. Conflicts between the two subnets are used to form a conflict loss $L_{jk}$. Each network uses the same sentence representation and generates different bag representations to feed a softmax layer separately. The *conflicts* and *loss* from collaborative training are used as cues for curriculum building in section 3.5.2.

### 3.3 Entity Position-aware Sentence Representation: Piece-wise Max Pooling

Following the self-attention operations, we use a piece-wise max pooling operation to form the final sentence representation. The max pooling is a variant of pooling in standard CNNs, which applies the pooling to three convolution segments separated by head and tail entities (Zeng et al., 2015). As shown in the column of "piece-wise pooling" in Figure 3(a), the grey dots in $\tilde{c}$, representing head and tail entities, split each vector into three pieces, which are denoted as $\tilde{C}_1, \tilde{C}_2, \tilde{C}_3$, respectively. Thus, the piece-wise max pooling is expressed as $\{x_i^1, x_i^2, x_i^3\} = \text{max-pooling}(\tilde{C}_1, \tilde{C}_2, \tilde{C}_3)$. These three pooled vectors are then concatenated together to form a vector $x_i$ and a nonlinear function is applied to the output vector, such that $x_i = tanh(x_i)$ is the final representation of a sentence.

### 3.4 Bag Representation for Entity Pairs

After learning representations of all sentences in an entity pair bag, we then use two approaches to form the entity pair bag representation, as illustrated in Figure 3(b), i.e., we employ a mechanism of averaging all sentences using attention scores (NetAtt) (Lin et al., 2016), and a maximum probability sentence selection method (NetMax) (Zeng et al., 2015), respectively, to learn the bag representation. Specifically, NetAtt assigns weight scores $\{\alpha_1, \ldots, \alpha_N\}$ to all sentences in an entity pair bag, while NetMax will select the most reasonable sentence that has the highest probability.

Ideal student networks are equipped with SOTA MIL selection mechanisms and could empirically generate conflicts during the selection, based on this criterion we use NetAtt and NetMax. NetAtt works by considering all sentences in a bag, but it also introduces noisy sentences while learning a bag representation. NetMax works by selecting the sentence with the highest probability in a bag as the bag representation, but it overlooks other valid sentences. Moreover, the two networks use different bag representations to feed classifiers and eventually generate *disagreements*. Therefore, we are motivated by the idea of combining their advantages and disagreements over sentence selection in a single framework to learn better bag representations and to reduce the noise effects.

Before introducing the proposed single framework, we first describe below how NetAtt and NetMax work in bag representation learning. The sentence bag of an entity pair is denoted as $B$, which consists of representations of $N$ sentences $\{x_1, x_2, \ldots, x_N\}$, where each sentence representation is learned from our self-attention based PCNN. The entity pair is expressed as $(e_1, e_2)$ and the bag's relation is $r$.

#### 3.4.1 NetAtt: Sentence-Level Attention

To extract information from all sentences in a bag, a sentence-level attention mechanism is used to learn a weight score $\alpha_i$ for each sentence. Subse-

quently, the bag representation $S^{att}_{bag}$ for the entity pair's bag $B$ is computed as: $S^{att}_{bag} = \sum_{i=1}^{N} \alpha_i x_i$.

We can see that the purpose of the weighted factor $\alpha_i$ is to give higher weights to correctly labeled instances and lower weights to wrongly labeled ones. Given the score $\beta_i$ of a given sentence representation $x_i$ with a relation $r$, which is measured as: $\beta_i(x_i|r) = x_i A r$ (where $A$ is a weighted diagonal matrix), the attention scores in a given bag $B$ can be calculated as: $\alpha_i(x_i|r, B) = softmax(\gamma\beta_i(x_i|r))$, where $\gamma$ is set empirically and borrowed from the work by (Sutton and Barto, 1998). Smaller $\gamma$ will lead to equalisation in all sentences and larger $\gamma$ will increase bias of the high scored sentence.

#### 3.4.2 NetMax: Maximum Probability

NetMax assumes that at least one sentence in an entity pair bag reflects the bag's relation, and only one sentence with the maximum probability is selected to represent the bag, which is denoted as $S^{max}_{bag} = x_k$, where $x_k$ is the $k^{th}$ sentence representation with maximum probability. As only one sentence is selected, the input $o$ for the *softmax* classifier can be expressed as $o = Kx_k + b$, where $K$ is the transformation matrix and $b$ is the bias. For a bag with relation $r$, the conditional probability is $p(r|x_k;\theta) = softmax(o_r)$. For all sentences in a bag, the index $k$ is computed by $k = arg\max_k p(r|x_k;\theta)$.

### 3.5 Collaborative Curriculum Learning

As mentioned above, we consider the advantages and disagreements of sentence selection of NetAtt and NetMax in a single framework so that they can learn to regularise each other so as to reduce the effects of noisy sentences. We propose a collaborative curriculum learning framework where NetAtt and NetMax are defined as two student networks and they learn together under a curriculum scheme. For DSRE, we assume that entity pair bags with wrongly labeled sentences are *hard samples* to be learned, while bags with correctly labeled sentences are *easy samples*. Figure 3(b) shows the architecture of our collaborative curriculum learning framework, where NetAtt and NetMax are trained collaboratively and regularised by each other. The curriculum vector $v_i$ for collaborative learning could be built by various schemes, for example, the *conflicts* ($v^c_i$) of selecting the valid sentence in a bag between the two student networks, which will be detailed later.

#### 3.5.1 Objective Function

The objective function is defined as $J(S_i;\theta) = \frac{1}{m}\sum_{i=1}^{m} j(S_i;\theta)$, where $m$ is the total number of entity pair's bags in a mini-batch. $S_i$ is a set of $\{S^{att}_{bag\_i}, S^{max}_{bag\_i}\}$. $j(S_i;\theta)$ is the objective function of one entity pair's bag defined in Equation (2):

$$\begin{aligned} j(S_i;\theta) = \eta \log p(r_i|S^{att}_{bag\_i};\theta) + \\ (1-\eta)\log p(r_i|S^{max}_{bag\_i};\theta) \end{aligned} \quad (2)$$

where, $0 < \eta < 1$ is an empirical value to assign weights to NetAtt and NetMax. With a curriculum, the model's minimisation problem can be formulated as in Equation (3):

$$\min_{\theta,v} E(\theta, v, \lambda) = \frac{1}{m}\sum_{i=1}^{m} v_i l_i + L_{jk} \quad (3)$$

where, $l_i = l(r_i, j(S_i;\theta))$ is the loss of each bag; $r_i$ is the relation of a bag; $v_i$ is the curriculum weight variable; $L_{jk}$ is the cross entropy loss of conflicts between the highest probability sentence indexes from NetAtt and NetMax, which aims to let them regularise each other during sentence selection; $\theta$ and $\lambda$ are the optimisation parameters of relation extractor and curriculum. The weighted loss is minimised by stochastic gradient descent.

**Algorithm 1** Update using conflicts

**inputs:** mini batch size $m$; $j^i_{att}$ and $k^i_{max}$; two students: $p_{att}$ (*NetAtt*), $p_{max}$ (*NetMax*); bag representation: $S^{att}_i$, $S^{max}_i$

**for** *i in* {1,2,3, ..., $m$} **do**
  **if** $j^i_{att} = k^i_{max}$ **then**
    # *no conflicts* $v^c_i = 1$
    add $S^{att}_i$ to set $\{S^{att}\}$
    add $S^{max}_i$ to set $\{S^{max}\}$
  **end**
**end**
**update** $p_{att}$ *with* $\{S^{att}\}$ **&** $p_{max}$ *with* $\{S^{max}\}$

#### 3.5.2 Curriculum Construction for Collaborative Learning

**Conflicts trick** of two students are utlised to build a curriculum ($v^c_i$): In a mini-batch of size $m$, where the $i^{th}$ entity pair's bag contains $N$ sentences, $j^i_{att}$, $k^i_{max} \in [0\ to\ N)$ are the indexes of highest probably sentence selected by NetAtt and NetMax, respectively. For each entity pair bag in the batch, if $j^i_{att}$ is not equal to $k^i_{max}$, then $v^c_i = 0$, representing a *'hard' sample* (with conflicts). Otherwise $v^c_i = 1$, representing an *'easy' sample*. The

conflicts that occur during the collaborative training between NetAtt and NetMax are shown in Figure 3(b). There is no extra curriculum network ($f(v^c;\lambda)$) required to learn a $v_i^c$.

When $v_i^c$ is assigned to 0, the training procedure will forget the effects of *'hard'* sample ($i^{th}$ bag) by multiplying 0 and its loss $l_i$ as shown in Eq. (3). Algorithm 1 illustrates the logic to update the training using the conflicts curriculum. In a mini-batch, entity pair bags with conflicts in sentence selection are dropped, the remaining bags are used to update the network parameters.

Furthermore, various curriculum types could be used in CCL to alleviate the noise in DSRE. We utilize the **small loss trick** to build a curriculum which inspired by Jiang et al. (2018). Specifically, to build a curriculum ($v_i^l$), the loss ($l_i$) is used as an input constant feature to learn a curriculum vector. We use the MentorNet framework[2] (Jiang et al., 2018) as the curriculum network to learn an approximate predefined curriculum $f(v^l;\lambda)$. The approximation process is to train a curriculum model using synthetic data generated according to the predefined curriculum. The trained model is then used as the curriculum to guide the further model training. We use the predefined curriculum $f(v^l;\lambda) = v_i^l l_i + \frac{1}{2}\lambda_2 (v_i^l)^2 - (\lambda_1 + \lambda_2) v_i^l$ (Jiang et al., 2015) to guide our training.

## 4 Experiments

### 4.1 Dataset

We evaluate our model on the widely used New York Times (NYT) DSRE dataset[3], which aligns Freebase entity relation with NYT corpus (Riedel et al., 2010). The dataset uses the data from 2005 to 2006 as the training set and the remaining data, taken from 2007, as the test set. The processed dataset[4] was released by Lin et al. (2016). We use the cleaned version of the processed dataset, which has removed duplicated sentences in training and test sets. In total, the training set consists of *522,611* sentences, 281,270 entity pairs and 18,252 relation facts. The testing set contains 172,448 sentences, 96,678 entity pairs, and 1,950 relation facts. The dataset contains 39,528 unique entities and 53 relations in total including an *NA* relation which represents no existing relation for given entity pairs in sentences.

[2] https://github.com/google/mentornet
[3] http://iesl.cs.umass.edu/riedel/ecml/
[4] https://github.com/thunlp/NRE

### 4.2 Evaluation Metrics

Instead of obtaining a costly human annotated test data, we conduct a held-out evaluation (Riedel et al., 2010) in the experiments, as in previous work (Riedel et al., 2010; Surdeanu et al., 2012; Zeng et al., 2015; Lin et al., 2016). Held-out evaluation compares relation facts predicted in test data with those relations identified in Freebase. It gives an approximate evaluation of the proposed model. As with the evaluation metrics used in the literature, we report our results using Precision-Recall curve (PR-curve) and Precision at N (P@N) metrics. *PR-curve* is used to understand the trade-off between precision and recall. Using all the test data, the plotted curve expresses the precision as a function of recall. *P@N* considers the cutoff top-most *N* precision values as a set. Each entity pair bag contains one or more instances and P@N considers *all* the multiple instances.

### 4.3 Baseline Models

We compare our model with both statistical and deep learning baseline models. These baselines were evaluated on the same cleaned dataset. We exclude some recent models due to the dataset and reproducibility issues. Namely, recent models that were trained on a dataset, which was released by mistake, obtained higher results. These results in fact may be inaccurate. From the Github repository commit history and comments[5]: *"It has not deleted the mix part of testing data. The training sentence is 570000+, but in the paper is 520000+."*. The incorrect training dataset was replaced with the corrected one in March 2018, which might be the main reason as to why some MIL DSRE papers in 2018 reported non-reproducible results on the corrected training data. In all probability these works had commenced prior to March 2018 and were likely relying on the erroneous dataset.

(1) Statistical Models: *Mintz* (Mintz et al., 2009) extracted sentence syntactic and lexical features and trained a multiclass logistic regression classifier. *Hoffmann* (Hoffmann et al., 2011) is a probabilistic, graphic model with MIL. *MIMLRE* (Surdeanu et al., 2012) is a MIL model that uses expectation maximization for classification.

(2) Deep Learning Models: both CNN (Zeng et al., 2014) and PCNN (Zeng et al., 2015) based

[5] https://github.com/thunlp/NRE/commit/77025e5cc6b42bc1adf3ec46835101d162013659

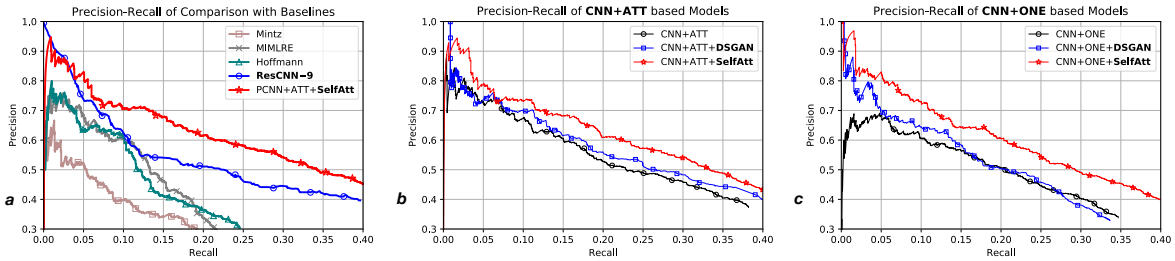


Figure 4: PR curves comparison of CNN/PCNN based models with SelfAtt module and baselines

relation extraction models with bag level selective attention mechanism (*PCNN+ATT*, *CNN+ATT*) (Lin et al., 2016) and multi-instance learning (*PCNN+ONE*, *CNN+ONE*) (Zeng et al., 2015).

(3) ResCNN-9 (Huang and Wang, 2017): 9-layers CNN model with residual identity shortcuts and three fully connected layers. The model outperforms CNN+ATT/ONE models.

(4) State-of-the-art DSRE Noise Filter Systems: DSGAN (Qin et al., 2018) is a model to filter out noise instances. It unitizes GANs to remove potentially inaccurate sentences from original training data and further trains PCNN/CNN models on the filtered data to improve performance.

| | |
|---|---|
| Number of CNN filters | 128, 230, 256 |
| Batch size $B$ | 120 |
| Learning rate | 0.1, 0.4 |
| Weight decay | 0.00001 |
| Burn-in epoch | 5 |
| Dropout probability | 0.3 |

Table 1: Parameters setting for best results

We fine-tune our models by validating and selecting the best model parameters. To accomplish this we set the gradient descent learning rate among $\{0.4, 0.2, 0.1, 0.01, 0.001\}$. Batch size $B$ is set to $\{60, 120, 160\}$. The amount of CNN filter is set among $\{64, 128, 230, 256\}$. A dropout rate is in the set $\{0.1, 0.3\}$. All the parameters fine-tuned in our experiments are shown in Table 1. Specifically, the best performance of [CNN+ATT/ONE+SelfAtt] with training parameters of learning rate:0.4 and CNN filters:230; the best performance of [PCNN+ATT/ONE+SelfAtt] with training parameters of learning rate:0.4 and CNN filters:128; and the best performance of [PCNN+ATT/ONE+SelfAtt+CCL] with training parameters of learning rate:0.1 and CNN filters:256. For other parameters, we follow the settings in the work of (Lin et al., 2016), e.g. the maximum sentence length is limited to 70, the word and position embedding size is fixed to 50 and 5 respectively, and the CNNs filter window size is 3.

### 4.4 Effects of Internal CNNs Self-Attention

Armed by the internal multilayer CNNs self-attention mechanism (named as **SelfAtt**), our model can learn a better representation from noisy data compared with the conventional CNN encoder used in PCNN/CNN. Each model has more than one convolution operation, as illustrated in Figure 3(a). Prior to feeding these outputs from the first convolution operation into the self-attention module, we add a batch normalisation followed by an ReLU activation function.

Figures 4 & 5 show the PR-curve results attained by applying SelfAtt to CNN/PCNN based models. We also report the results of P@100, P@200, P@300 and the Mean for CNN/PCNN+ONE and CNN/PCNN+ATT with SelfAtt in the held-out evaluation. Table 3 shows the P@N values with test settings where all sentences in an entity pair's bag are taken into account. PR-curve and P@N results demonstrate that CNN/PCNN based approaches achieve improved results with the SelfAtt. PCNN/CNN models with SelfAtt also outperform ResCNN-9 in terms of P@N and PR-curve, which indicates that the proposed SelfAtt is beneficial for boosting the performance of the models learning from noisy inputs. The state-of-the-art DSGAN system demonstrates its ability to improve PCNN/CNN + ATT/ONE by filtering out noisy data. By comparing our SelfAtt with DSGAN, Figure 4 shows that SelfAtt significantly outperforms the DSGAN system in terms of CNN-based models.

The intuition of adding an internal CNN states self-attention module to help DSRE task is that, 1/ a deeper CNN has positive effects on noisy NLP tasks (Conneau et al., 2017), 2/ attention enhanced PCNN/CNN is expected to assign various weight scores to different sentence portions and will form a better representation in the DSRE setting.

By applying the SelfAtt it could add more

convolution layers into a model, as the internal self-attention used is placed between two convolution operations. Huang and Wang (2017) demonstrated that multi-layer ResCNNs network does achieve performance improvement by adding residual identity shortcuts, which aligns with the study that deeper CNN has positive effects on noisy NLP tasks (Conneau et al., 2017). However, previous DSRE researches overlooked multilayer CNN/PCNN with ONE/ATT. To investigate multilayer effects of DSRE, we present multilayer CNN with ATT results in Table 2, we observe that with ATT several multilayer models (e.g., 2 layers CNN+ATT) have improvements compared with single layer models, and overfitting occurs when larger convolution layers are employed.

| P@N(%) | 100 | 200 | 300 | Mean | AUC |
|---|---|---|---|---|---|
| CNN-1+ATT | 76.2 | 68.6 | 59.8 | 68.2 | 0.327 |
| **CNN-2+ATT** | 76.2 | **72.1** | **68.4** | **72.2** | **0.348** |
| **CNN-5+ATT** | 74.2 | **72.6** | **66.1** | **71.0** | **0.338** |
| CNN-9+ATT | 66.3 | 64.2 | 63.7 | 64.7 | 0.315 |

Table 2: Multilayer CNNs+ATT

Thus, our SelfAtt design is expected to boost a sentence encoder with attention scores and alleviate noise effects by benefiting from a multilayer CNN network. From empirical testing, to apply attention more subtly, placing self-attention after the second convolution operation and followed by one convolution operation works well for all PCNN/CNN + ONE/ATT based models generally. We report results with this setting.

avaya , which was once a division of lucent technologies and att before that , is one of the nation 's top makers of phone equipment , rivaling cisco , nortel and alcatel-lucent in providing internet-based communications to corporations .

By looking at the weight scores from SeftAtt, we observe that different parts of a sentence obtain different attentions. For example, as the heat-map illustrates using attention scores, entities *avaya* and *cisco* obtain more attention than others.

### 4.5 Effects of Collaborative Curriculum Learning

We refer to the Collaborative Curriculum Learning using Conflicts Trick as CCL-CT, and using the Small Loss trick as CCL-SL. For the following experiments, we expect that by applying our CCL strategies, it will result in further improvements by reducing the undesirable effects of noise. In our framework of integrating collaborative curriculum learning, both NetAtt and NetMax could be used for testing. We report a comparison of NetAtt/NetMax+SelfAtt+CCL, PCNN+ONE/ATT and state-of-the-art DSGAN noise reduction as shown in Table 3 and Figure 5.

| P@N (%) | 100 | 200 | 300 | Mean |
|---|---|---|---|---|
| *CNN-based Models* | | | | |
| CNN+ONE | 67.3 | 64.7 | 58.1 | 63.4 |
| **ResCNN-9** | 79.0 | 69.0 | 61.0 | 69.7 |
| CNN+ONE+**SelfAtt** | **81.1** | **75.1** | **70.4** | **75.5** |
| CNN+ATT | 76.2 | 68.6 | 59.8 | 68.2 |
| CNN+ATT+**SelfAtt** | **81.1** | **74.1** | **72.4** | **75.9** |
| *PCNN-based Models* | | | | |
| PCNN+ONE | 72.3 | 69.7 | 64.1 | 68.7 |
| PCNN+ONE+**SelAtt** | 84.1 | 75.1 | 69.1 | **76.1** |
| [NetMax+**SelfAtt**]+**CCL-CT** | **85.1** | **78.6** | **74.4** | **79.4** |
| PCNN+ATT | 76.2 | 73.1 | 67.4 | 72.2 |
| PCNN+ATT+**SelfAtt** | 81.1 | 71.6 | 70.4 | **74.4** |
| [NetAtt+**SelfAtt**]+**CCL-CT** | **82.2** | **79.1** | **73.1** | **78.1** |

Table 3: P@N results for models with internal CNNs self-attention and curriculum learning

*CCL-CT* utilises the conflicts from the collaborative training of NetMax and NetAtt to form a curriculum to guide the training. The two students form different bag representations to feed to Softmax layer separately, and they generate disagreements on the bag-level selection during training. In our experiment, each epoch reveals less than 10.9% disagreement, thus we drop less than 10.9% of the total entity pair bags during each epoch and the drop ratios work well in our experiments.

| **Model** | | **+SelfAtt** | **CCL-CT** |
|---|---|---|---|
| PCNN+ATT | 0.341 | **0.368** | **0.381** |
| PCNN+ONE | 0.325 | **0.352** | **0.380** |

Table 4: Comparison of AUC Results

From Figures 5(a) and 5(b), we can see that the CCL based models have further improvements in terms of PR-curves compared with PCNN+ATT/ONE+SelfAtt. The P@N results in Table 3 indicate that CCL further improves the model's performance when compared to PCNN+ATT/ONE+SelfAtt as well. Table 4 gives another comparison using AUC with all p-values less than 5e-02 from $t$-test evaluation. The results indicate that the larger AUC, the better performance. A simple ensemble model of two networks (AUC: 0.371) has a similar result as a single model (NetAtt, AUC: 0.368). The main purpose of adding an additional student network is to introduce conflicts and to build the collaborative curriculum learning scheme. With the CCL strategies, our models improve performances by removing *'hard'* (noisy) entity pair bags during

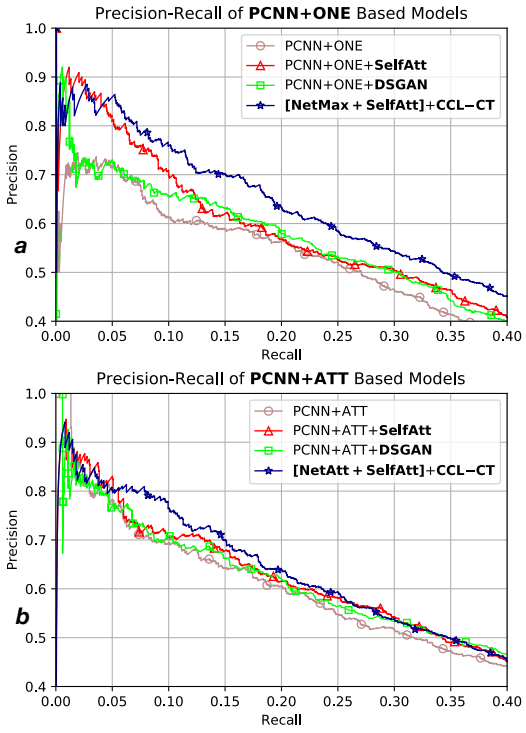


Figure 5: PR curves comparison of PCNN based noise removal models

training. When compared with the state-of-the-art DSGAN system, our models outperform both DSGAN based ONE and ATT models.

We examine the small loss trick based curriculum using MentorNet (Jiang et al., 2018). *CCL-SL* uses the loss from the collaborative training to build the curriculum vector, which commences by guiding the collaborative training from the burn-in epoch ($5^{th}$) and the optimal epoch result of (AUC:0.382, P@N mean:77.3%), which is reported for held-out evaluation starting at the burn-in epoch. The results also demonstrate that various curriculum types (the conflict and loss tricks) could help to alleviate the noise in DSRE.

Overall, our experimental results demonstrate that the proposed SelfAtt and CCL strategies for PCNN/CNN models significantly outperform baselines in terms of PR-curve and P@N.

## 5 Conclusion

To deal with the mislabelling issue in distantly supervised relation extraction, this paper details the development of a novel model based on a multi-layer self-attention mechanism for CNNs and collaborative curriculum learning strategies with two students (NetAtt and NetMax). The internal self-attention model can learn a better sentence representation by taking advantage of deeper CNNs in terms of positive effects on noisy inputs. The CCL strategies can perform a collaborative training on NetAtt and NetMax by allowing them to regularize each other, in tandem with the removal of noisy sample effects. Two different tricks, namely conflicts tricks and small loss tricks, are utilized in the CCL framework. Experimental results on the commonly-used NYT dataset indicate that our proposed approaches significantly outperform state-of-the-art baseline models in terms of P@N and PR-curve evaluation metrics.

## 6 Acknowledgement

We thank co-authors Jingguang Han and Sha Liu ({jingguang.han, sha.liu}@ucd.ie) and anonymous reviewers for these insightful comments and suggestions. We would like to thank Emer Gilmartin (gilmare@tcd.ie) for helpful comments and presentation improvements. This research is funded by the Enterprise-Ireland Innovation Partnership Programme (Grant IP2017626).

## References

Christoph Alt, Marc Hübner, and Leonhard Hennig. 2019. Fine-tuning pre-trained transformer language models to distantly supervised relation extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1388–1398, Florence, Italy. ACL.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 551–561.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2017. Very deep convolutional networks for text classification. In *European Chapter of the Association for Computational Linguistics EACL'17*.

Jinhua Du, Jingguang Han, Andy Way, and Dadong Wan. 2018. Multi-level structured self-attentions for distantly supervised relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2216–2225.

Jun Feng, Minlie Huang, Li Zhao, Yang Yang, and Xiaoyan Zhu. 2018. Reinforcement learning for relation classification from noisy data. In *Proceedings of AAAI*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*, pages 2672–2680.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 541–550. ACL.

YiYao Huang and William Yang Wang. 2017. Deep residual learning for weakly-supervised relation extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1803–1807.

Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. 2017. Distant supervision for relation extraction with sentence-level attention and entity descriptions. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3060–3066.

Lu Jiang, Deyu Meng, Qian Zhao, Shiguang Shan, and Alexander Hauptmann. 2015. Self-paced curriculum learning. In *AAAI*.

Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. 2018. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *ICML*.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2124–2133.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th ACL and the 4th IJCNLP of the AFNLP*, pages 1003–1011. Association for Computational Linguistics.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255.

Pengda Qin, Weiran Xu, and William Yang Wang. 2018. Dsgan: Generative adversarial training for distant supervision relation extraction. *arXiv preprint arXiv:1805.09929*.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer.

Cícero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *ACL*. Association for Computational Linguistics.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, pages 455–465. Association for Computational Linguistics.

Richard S Sutton and Andrew G Barto. 1998. Reinforcement learning: An introduction.

Shingo Takamatsu, Issei Sato, and Hiroshi Nakagawa. 2012. Reducing wrong labels in distant supervision for relation extraction. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 721–729. ACL.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Haoyu Wang, Ming Tan, Mo Yu, Shiyu Chang, Dakuo Wang, Kun Xu, Xiaoxiao Guo, and Saloni Potdar. 2019. Extracting multiple-relations in one-pass with pre-trained transformers. *arXiv preprint arXiv:1902.01030*.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on EMNLP*, pages 1753–1762.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344.

Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. 2018. Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318*.

# Neural Cross-Lingual Relation Extraction Based on Bilingual Word Embedding Mapping

**Jian Ni** and **Radu Florian**
IBM Research AI
1101 Kitchawan Road, Yorktown Heights, NY 10598, USA
{nij, raduf}@us.ibm.com

## Abstract

Relation extraction (RE) seeks to detect and classify semantic relationships between entities, which provides useful information for many NLP applications. Since the state-of-the-art RE models require large amounts of manually annotated data and language-specific resources to achieve high accuracy, it is very challenging to transfer an RE model of a resource-rich language to a resource-poor language. In this paper, we propose a new approach for cross-lingual RE model transfer based on bilingual word embedding mapping. It projects word embeddings from a target language to a source language, so that a well-trained source-language neural network RE model can be directly applied to the target language. Experiment results show that the proposed approach achieves very good performance for a number of target languages on both in-house and open datasets, using a small bilingual dictionary with only 1K word pairs.

## 1 Introduction

*Relation extraction* (RE) is an important information extraction task that seeks to detect and classify semantic relationships between entities like persons, organizations, geo-political entities, locations, and events. It provides useful information for many NLP applications such as knowledge base construction, text mining and question answering. For example, the entity *Washington, D.C.* and the entity *United States* have a *CapitalOf* relationship, and extraction of such relationships can help answer questions like "What is the capital city of the United States?"

Traditional RE models (e.g., Zelenko et al. (2003); Kambhatla (2004); Li and Ji (2014)) require careful feature engineering to derive and combine various lexical, syntactic and semantic features. Recently, neural network RE models (e.g., Zeng et al. (2014); dos Santos et al. (2015); Miwa and Bansal (2016); Nguyen and Grishman (2016)) have become very successful. These models employ a certain level of automatic feature learning by using word embeddings, which significantly simplifies the feature engineering task while considerably improving the accuracy, achieving the state-of-the-art performance for relation extraction.

All the above RE models are supervised machine learning models that need to be trained with large amounts of manually annotated RE data to achieve high accuracy. However, annotating RE data by human is expensive and time-consuming, and can be quite difficult for a new language. Moreover, most RE models require language-specific resources such as dependency parsers and part-of-speech (POS) taggers, which also makes it very challenging to transfer an RE model of a resource-rich language to a resource-poor language.

There are a few existing weakly supervised cross-lingual RE approaches that require no human annotation in the target languages, e.g., Kim et al. (2010); Kim and Lee (2012); Faruqui and Kumar (2015); Zou et al. (2018). However, the existing approaches require aligned parallel corpora or machine translation systems, which may not be readily available in practice.

In this paper, we make the following contributions to cross-lingual RE:

- We propose a new approach for direct cross-lingual RE model transfer based on bilingual word embedding mapping. It projects word embeddings from a target language to a source language (e.g., English), so that a well-trained source-language RE model can be directly applied to the target language, with no manually annotated RE data needed for the target language.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 399–409, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

- We design a deep neural network architecture for the source-language (English) RE model that uses word embeddings and generic language-independent features as the input. The English RE model achieves the-state-of-the-art performance without using language-specific resources.

- We conduct extensive experiments which show that the proposed approach achieves very good performance (up to 79% of the accuracy of the supervised target-language RE model) for a number of target languages on both in-house and the ACE05 datasets (Walker et al., 2006), using a small bilingual dictionary with only 1K word pairs. To the best of our knowledge, this is the first work that includes empirical studies for cross-lingual RE on several languages across a variety of language families, without using aligned parallel corpora or machine translation systems.

We organize the paper as follows. In Section 2 we provide an overview of our approach. In Section 3 we describe how to build monolingual word embeddings and learn a linear mapping between two languages. In Section 4 we present a neural network architecture for the source-language (English). In Section 5 we evaluate the performance of the proposed approach for a number of target languages. We discuss related work in Section 6 and conclude the paper in Section 7.

## 2 Overview of the Approach

We summarize the main steps of our neural cross-lingual RE model transfer approach as follows.

1. Build word embeddings for the source language and the target language separately using monolingual data.

2. Learn a linear mapping that projects the target-language word embeddings into the source-language embedding space using a small bilingual dictionary.

3. Build a neural network source-language RE model that uses word embeddings and generic language-independent features as the input.

4. For a target-language sentence and any two entities in it, project the word embeddings of the words in the sentence to the source-language word embeddings using the linear mapping, and then apply the source-language RE model on the projected word embeddings to classify the relationship between the two entities. An example is shown in Figure 1, where the target language is Portuguese and the source language is English.

We will describe each component of our approach in the subsequent sections.

## 3 Cross-Lingual Word Embeddings

In recent years, vector representations of words, known as *word embeddings*, become ubiquitous for many NLP applications (Collobert et al., 2011; Mikolov et al., 2013a; Pennington et al., 2014).

A monolingual word embedding model maps words in the vocabulary $\mathcal{V}$ of a language to real-valued vectors in $\mathbb{R}^{d\times 1}$. The dimension of the vector space $d$ is normally much smaller than the size of the vocabulary $V = |\mathcal{V}|$ for efficient representation. It also aims to capture semantic similarities between the words based on their distributional properties in large samples of monolingual data.

Cross-lingual word embedding models try to build word embeddings across multiple languages (Upadhyay et al., 2016; Ruder et al., 2017). One approach builds monolingual word embeddings separately and then maps them to the same vector space using a bilingual dictionary (Mikolov et al., 2013b; Faruqui and Dyer, 2014). Another approach builds multilingual word embeddings in a shared vector space simultaneously, by generating mixed language corpora using aligned sentences (Luong et al., 2015; Gouws et al., 2015).

In this paper, we adopt the technique in (Mikolov et al., 2013b) because it only requires a small bilingual dictionary of aligned word pairs, and does not require parallel corpora of aligned sentences which could be more difficult to obtain.

### 3.1 Monolingual Word Embeddings

To build monolingual word embeddings for the source and target languages, we use a variant of the Continuous Bag-of-Words (CBOW) word2vec model (Mikolov et al., 2013a).

The standard CBOW model has two matrices, the input word matrix $\tilde{\mathbf{X}} \in \mathbb{R}^{d\times V}$ and the output word matrix $\mathbf{X} \in \mathbb{R}^{d\times V}$. For the $i$th word $w_i$ in $\mathcal{V}$, let $\mathbf{e}(w_i) \in \mathbb{R}^{V\times 1}$ be a one-hot vector with 1 at index $i$ and 0s at other indexes, so that $\tilde{\mathbf{x}}_i = \tilde{\mathbf{X}}\mathbf{e}(w_i)$

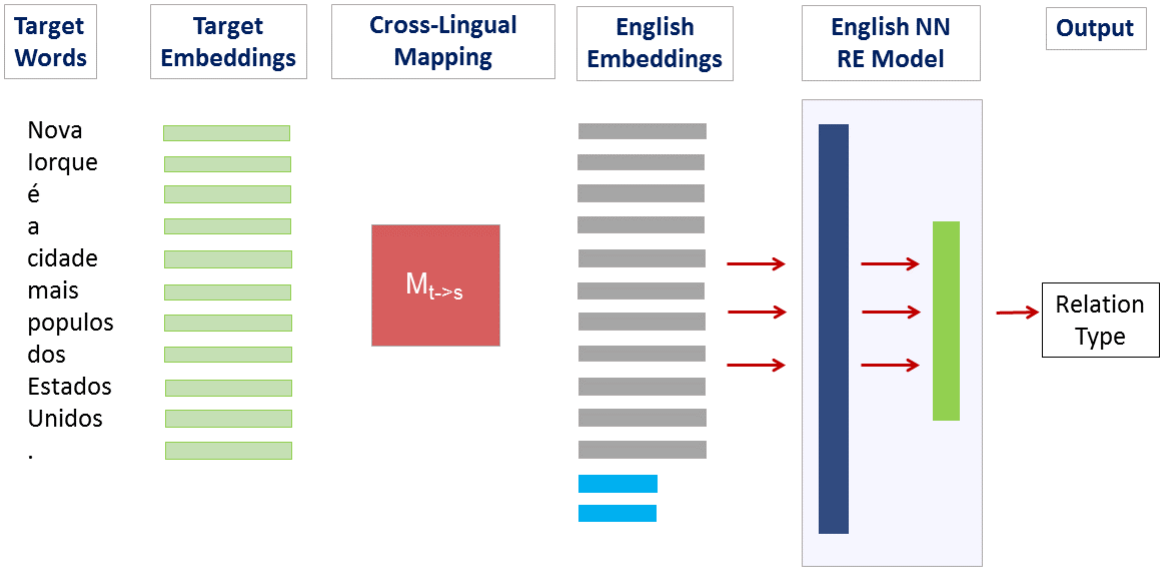


Figure 1: Neural cross-lingual relation extraction based on bilingual word embedding mapping - target language: Portuguese, source language: English.

(the $i$th column of $\tilde{\mathbf{X}}$) is the input vector representation of word $w_i$, and $\mathbf{x}_i = \mathbf{X}\mathbf{e}(w_i)$ (the $i$th column of $\mathbf{X}$) is the output vector representation (i.e., word embedding) of word $w_i$.

Given a sequence of training words $w_1, w_2, ..., w_N$, the CBOW model seeks to predict a target word $w_t$ using a window of $2c$ context words surrounding $w_t$, by maximizing the following objective function:

$$\mathcal{L} = \frac{1}{N}\sum_{t=1}^{N} \log P(w_t | w_{t-c}, ..., w_{t-1}, w_{t+1}, ..., w_{t+c})$$

The conditional probability is calculated using a softmax function:

$$P(w_t | w_{t-c}, ..., w_{t+c}) = \frac{\exp(\mathbf{x}_t^{\mathrm{T}} \tilde{\mathbf{x}}_{c(t)})}{\sum_{i=1}^{V} \exp(\mathbf{x}_i^{\mathrm{T}} \tilde{\mathbf{x}}_{c(t)})} \quad (1)$$

where $\mathbf{x}_t = \mathbf{X}\mathbf{e}(w_t)$ is the output vector representation of word $w_t$, and

$$\tilde{\mathbf{x}}_{c(t)} = \sum_{-c \le j \le c, j \neq 0} \tilde{\mathbf{X}}\mathbf{e}(w_{t+j}) \quad (2)$$

is the sum of the input vector representations of the context words.

In our variant of the CBOW model, we use a separate input word matrix $\tilde{\mathbf{X}}_j$ for a context word at position $j, -c \le j \le c, j \neq 0$. In addition, we employ weights that decay with the distances of the context words to the target word. Under these modifications, we have

$$\tilde{\mathbf{x}}_{c(t)}^{new} = \sum_{-c \le j \le c, j \neq 0} \frac{1}{|j|} \tilde{\mathbf{X}}_j \mathbf{e}(w_{t+j}) \quad (3)$$

We use the variant to build monolingual word embeddings because experiments on named entity recognition and word similarity tasks showed this variant leads to small improvements over the standard CBOW model (Ni et al., 2017).

### 3.2 Bilingual Word Embedding Mapping

Mikolov et al. (2013b) observed that word embeddings of different languages often have similar geometric arrangements, and suggested to learn a linear mapping between the vector spaces.

Let $\mathcal{D}$ be a bilingual dictionary with aligned word pairs $(w_i, v_i)_{i=1,...,D}$ between a source language $s$ and a target language $t$, where $w_i$ is a source-language word and $v_i$ is the translation of $w_i$ in the target language. Let $\mathbf{x}_i \in \mathbb{R}^{d \times 1}$ be the word embedding of the source-language word $w_i$, $\mathbf{y}_i \in \mathbb{R}^{d \times 1}$ be the word embedding of the target-language word $v_i$.

We find a linear mapping (matrix) $\mathbf{M}_{t \to s}$ such that $\mathbf{M}_{t \to s}\mathbf{y}_i$ approximates $\mathbf{x}_i$, by solving the fol-

lowing least squares problem using the dictionary as the training set:

$$\mathbf{M}_{t\rightarrow s} = \arg\min_{\mathbf{M}\in\mathbb{R}^{d\times d}} \sum_{i=1}^{D} ||\mathbf{x}_i - \mathbf{M}\mathbf{y}_i||^2 \quad (4)$$

Using $\mathbf{M}_{t\rightarrow s}$, for any target-language word $v$ with word embedding $\mathbf{y}$, we can project it into the source-language embedding space as $\mathbf{M}_{t\rightarrow s}\mathbf{y}$.

#### 3.2.1 Length Normalization and Orthogonal Transformation

To ensure that all the training instances in the dictionary $\mathcal{D}$ contribute equally to the optimization objective in (4) and to preserve vector norms after projection, we have tried length normalization and orthogonal transformation for learning the bilingual mapping as in (Xing et al., 2015; Artetxe et al., 2016; Smith et al., 2017).

First, we normalize the source-language and target-language word embeddings to be unit vectors: $\mathbf{x}' = \frac{\mathbf{x}}{||\mathbf{x}||}$ for each source-language word embedding $\mathbf{x}$, and $\mathbf{y}' = \frac{\mathbf{y}}{||\mathbf{y}||}$ for each target-language word embedding $\mathbf{y}$.

Next, we add an orthogonality constraint to (4) such that $\mathbf{M}$ is an orthogonal matrix, i.e., $\mathbf{M}^{\mathrm{T}}\mathbf{M} = \mathbf{I}$ where $\mathbf{I}$ denotes the identity matrix:

$$\mathbf{M}^{O}_{t\rightarrow s} = \arg\min_{\mathbf{M}\in\mathbb{R}^{d\times d}, \mathbf{M}^{\mathrm{T}}\mathbf{M}=\mathbf{I}} \sum_{i=1}^{D} ||\mathbf{x}'_i - \mathbf{M}\mathbf{y}'_i||^2 \quad (5)$$

$\mathbf{M}^{O}_{t\rightarrow s}$ can be computed using singular-value decomposition (SVD).

#### 3.2.2 Semi-Supervised and Unsupervised Mappings

The mapping learned in (4) or (5) requires a seed dictionary. To relax this requirement, Artetxe et al. (2017) proposed a self-learning procedure that can be combined with a dictionary-based mapping technique. Starting with a small seed dictionary, the procedure iteratively 1) learns a mapping using the current dictionary; and 2) computes a new dictionary using the learned mapping.

Artetxe et al. (2018) proposed an unsupervised method to learn the bilingual mapping without using a seed dictionary. The method first uses a heuristic to build an initial dictionary that aligns the vocabularies of two languages, and then applies a robust self-learning procedure to iteratively improve the mapping. Another unsupervised method based on adversarial training was proposed in Conneau et al. (2018).

We compare the performance of different mappings for cross-lingual RE model transfer in Section 5.3.2.

## 4 Neural Network RE Models

For any two entities in a sentence, an RE model determines whether these two entities have a relationship, and if yes, classifies the relationship into one of the pre-defined relation types. We focus on neural network RE models since these models achieve the state-of-the-art performance for relation extraction. Most importantly, neural network RE models use word embeddings as the input, which are amenable to cross-lingual model transfer via cross-lingual word embeddings. In this paper, we use English as the source language.

Our neural network architecture has four layers. The first layer is the *embedding layer* which maps input words in a sentence to word embeddings. The second layer is a *context layer* which transforms the word embeddings to context-aware vector representations using a recurrent or convolutional neural network layer. The third layer is a *summarization layer* which summarizes the vectors in a sentence by grouping and pooling. The final layer is the *output layer* which returns the classification label for the relation type.

### 4.1 Embedding Layer

For an English sentence with $n$ words $\mathbf{s} = (w_1, w_2, ..., w_n)$, the *embedding layer* maps each word $w_t$ to a real-valued vector (word embedding) $\mathbf{x}_t \in \mathbb{R}^{d\times 1}$ using the English word embedding model (Section 3.1). In addition, for each entity $m$ in the sentence, the embedding layer maps its entity type to a real-valued vector (entity label embedding) $\mathbf{l}_m \in \mathbb{R}^{d_m\times 1}$ (initialized randomly). In our experiments we use $d = 300$ and $d_m = 50$.

### 4.2 Context Layer

Given the word embeddings $\mathbf{x}_t$'s of the words in the sentence, the *context layer* tries to build a sentence-context-aware vector representation for each word. We consider two types of neural network layers that aim to achieve this.

#### 4.2.1 Bi-LSTM Context Layer

The first type of context layer is based on Long Short-Term Memory (LSTM) type recurrent neural networks (Hochreiter and Schmidhuber, 1997;

Graves and Schmidhuber, 2005). Recurrent neural networks (RNNs) are a class of neural networks that operate on sequential data such as sequences of words. LSTM networks are a type of RNNs that have been invented to better capture long-range dependencies in sequential data.

We pass the word embeddings $\mathbf{x}_t$'s to a forward and a backward LSTM layer. A forward or backward LSTM layer consists of a set of recurrently connected blocks known as *memory blocks*. The memory block at the $t$-th word in the forward LSTM layer contains a memory cell $\overrightarrow{\mathbf{c}}_t$ and three *gates*: an input gate $\overrightarrow{\mathbf{i}}_t$, a forget gate $\overrightarrow{\mathbf{f}}_t$ and an output gate $\overrightarrow{\mathbf{o}}_t$ ($\overrightarrow{\cdot}$ indicates the forward direction), which are updated as follows:

$$\begin{aligned}
\overrightarrow{\mathbf{i}}_t &= \sigma(\overrightarrow{W}_i \mathbf{x}_t + \overrightarrow{U}_i \overrightarrow{\mathbf{h}}_{t-1} + \overrightarrow{\mathbf{b}}_i) \\
\overrightarrow{\mathbf{f}}_t &= \sigma(\overrightarrow{W}_f \mathbf{x}_t + \overrightarrow{U}_f \overrightarrow{\mathbf{h}}_{t-1} + \overrightarrow{\mathbf{b}}_f) \\
\overrightarrow{\mathbf{o}}_t &= \sigma(\overrightarrow{W}_o \mathbf{x}_t + \overrightarrow{U}_o \overrightarrow{\mathbf{h}}_{t-1} + \overrightarrow{\mathbf{b}}_o) \\
\overrightarrow{\mathbf{c}}_t &= \overrightarrow{\mathbf{f}}_t \odot \overrightarrow{\mathbf{c}}_{t-1} + \\
&\quad \overrightarrow{\mathbf{i}}_t \odot \tanh\left(\overrightarrow{W}_c \mathbf{x}_t + \overrightarrow{U}_c \overrightarrow{\mathbf{h}}_{t-1} + \overrightarrow{\mathbf{b}}_c\right) \\
\overrightarrow{\mathbf{h}}_t &= \overrightarrow{\mathbf{o}}_t \odot \tanh(\overrightarrow{\mathbf{c}}_t) \qquad (6)
\end{aligned}$$

where $\sigma$ is the element-wise sigmoid function and $\odot$ is the element-wise multiplication.

The hidden state vector $\overrightarrow{\mathbf{h}}_t$ in the forward LSTM layer incorporates information from the left (past) tokens of $w_t$ in the sentence. Similarly, we can compute the hidden state vector $\overleftarrow{\mathbf{h}}_t$ in the backward LSTM layer, which incorporates information from the right (future) tokens of $w_t$ in the sentence. The concatenation of the two vectors $\mathbf{h}_t = [\overrightarrow{\mathbf{h}}_t, \overleftarrow{\mathbf{h}}_t]$ is a good representation of the word $w_t$ with both left and right contextual information in the sentence.

#### 4.2.2 CNN Context Layer

The second type of context layer is based on Convolutional Neural Networks (CNNs) (Zeng et al., 2014; dos Santos et al., 2015), which applies *convolution-like* operation on successive windows of size $k$ around each word in the sentence. Let $\mathbf{z}_t = [\mathbf{x}_{t-(k-1)/2}, ..., \mathbf{x}_{t+(k-1)/2}]$ be the concatenation of $k$ word embeddings around $w_t$. The convolutional layer computes a hidden state vector

$$\mathbf{h}_t = \tanh(\mathbf{W}\mathbf{z}_t + \mathbf{b}) \qquad (7)$$

for each word $w_t$, where $\mathbf{W}$ is a weight matrix and $\mathbf{b}$ is a bias vector, and $\tanh(\cdot)$ is the element-wise hyperbolic tangent function.

### 4.3 Summarization Layer

After the context layer, the sentence $(w_1, w_2, ..., w_n)$ is represented by $(\mathbf{h}_1, ...., \mathbf{h}_n)$. Suppose $m_1 = (w_{b_1}, .., w_{e_1})$ and $m_2 = (w_{b_2}, .., w_{e_2})$ are two entities in the sentence where $m_1$ is on the left of $m_2$ (i.e., $e_1 < b_2$). As different sentences and entities may have various lengths, the *summarization layer* tries to build a *fixed-length* vector that best summarizes the representations of the sentence and the two entities for relation type classification.

We divide the hidden state vectors $\mathbf{h}_t$'s into 5 groups:

- $G_1 = \{\mathbf{h}_1, .., \mathbf{h}_{b_1-1}\}$ includes vectors that are left to the first entity $m_1$.
- $G_2 = \{\mathbf{h}_{b_1}, .., \mathbf{h}_{e_1}\}$ includes vectors that are in the first entity $m_1$.
- $G_3 = \{\mathbf{h}_{e_1+1}, .., \mathbf{h}_{b_2-1}\}$ includes vectors that are between the two entities.
- $G_4 = \{\mathbf{h}_{b_2}, .., \mathbf{h}_{e_2}\}$ includes vectors that are in the second entity $m_2$.
- $G_5 = \{\mathbf{h}_{e_2+1}, .., \mathbf{h}_n\}$ includes vectors that are right to the second entity $m_2$.

We perform element-wise max pooling among the vectors in each group:

$$\mathbf{h}_{G_i}(j) = \max_{\mathbf{h} \in G_i} \mathbf{h}(j), 1 \le j \le d_h, 1 \le i \le 5 \qquad (8)$$

where $d_h$ is the dimension of the hidden state vectors. Concatenating the $\mathbf{h}_{G_i}$'s we get a fixed-length vector $\mathbf{h}_s = [\mathbf{h}_{G_1}, ..., \mathbf{h}_{G_5}]$.

### 4.4 Output Layer

The output layer receives inputs from the previous layers (the summarization vector $\mathbf{h}_s$, the entity label embeddings $\mathbf{l}_{m_1}$ and $\mathbf{l}_{m_2}$ for the two entities under consideration) and returns a probability distribution over the relation type labels:

$$\mathbf{p} = \text{softmax}\left(\mathbf{W}_s \mathbf{h}_s + \mathbf{W}_{m_1} \mathbf{l}_{m_1} + \mathbf{W}_{m_2} \mathbf{l}_{m_2} + \mathbf{b}_o\right) \qquad (9)$$

### 4.5 Cross-Lingual RE Model Transfer

Given the word embeddings of a sequence of words in a target language $t$, $(\mathbf{y}_1, ..., \mathbf{y}_n)$, we project them into the English embedding space by applying the linear mapping $\mathbf{M}_{t \to s}$ learned in Section 3.2: $(\mathbf{M}_{t \to s}\mathbf{y}_1, \mathbf{M}_{t \to s}\mathbf{y}_2, ..., \mathbf{M}_{t \to s}\mathbf{y}_n)$. The

neural network English RE model is then applied on the projected word embeddings and the entity label embeddings (which are language independent) to perform relationship classification.

Note that our models do not use language-specific resources such as dependency parsers or POS taggers because these resources might not be readily available for a target language. Also our models do not use precise word position features since word positions in sentences can vary a lot across languages.

## 5 Experiments

In this section, we evaluate the performance of the proposed cross-lingual RE approach on both in-house dataset and the ACE (Automatic Content Extraction) 2005 multilingual dataset (Walker et al., 2006).

### 5.1 Datasets

Our in-house dataset includes manually annotated RE data for 6 languages: English, German, Spanish, Italian, Japanese and Portuguese. It defines 56 entity types (e.g., *Person*, *Organization*, *Geo-Political Entity*, *Location*, *Facility*, *Time*, *Event_Violence*, etc.) and 53 relation types between the entities (e.g., *AgentOf*, *LocatedAt*, *PartOf*, *TimeOf*, *AffectedBy*, etc.).

The ACE05 dataset includes manually annotated RE data for 3 languages: English, Arabic and Chinese. It defines 7 entity types (*Person*, *Organization*, *Geo-Political Entity*, *Location*, *Facility*, *Weapon*, *Vehicle*) and 6 relation types between the entities (*Agent-Artifact*, *General-Affiliation*, *ORG-Affiliation*, *Part-Whole*, *Personal-Social*, *Physical*).

For both datasets, we create a class label "O" to denote that the two entities under consideration do not have a relationship belonging to one of the relation types of interest.

### 5.2 Source (English) RE Model Performance

We build 3 neural network English RE models under the architecture described in Section 4:

- The first neural network RE model does not have a context layer and the word embeddings are directly passed to the summarization layer. We call it **Pass-Through** for short.
- The second neural network RE model has a Bi-LSTM context layer. We call it **Bi-LSTM** for short.
- The third neural network model has a CNN context layer with a window size 3. We call it **CNN** for short.

| Model | $F_1$ |
|---|---|
| FCM (S) (Gormley et al., 2015) | 55.06 |
| Hybrid FCM (E) (Gormley et al., 2015) | 58.26 |
| BIDIRECT (S) (Nguyen and Grishman, 2016) | 57.73 |
| VOTE-BW (E) (Nguyen and Grishman, 2016) | 60.60 |
| Pass-Through (S) | 54.99 |
| Bi-LSTM (S) | 58.92 |
| CNN (S) | 57.91 |

Table 1: Comparison with the state-of-the-art RE models on the ACE05 English data (S: Single Model; E: Ensemble Model).

| **In-House** | **Training** | **Dev** | **Test** |
|---|---|---|---|
| English (Source) | 1137 | 140 | 140 |
| German (Target) | 280 | 35 | 35 |
| Spanish (Target) | 451 | 55 | 55 |
| Italian (Target) | 322 | 40 | 40 |
| Japanese (Target) | 396 | 50 | 50 |
| Portuguese (Target) | 390 | 50 | 50 |
| **ACE05** | **Training** | **Dev** | **Test** |
| English (Source) | 479 | 60 | 60 |
| Arabic (Target) | 323 | 40 | 40 |
| Chinese (Target) | 507 | 63 | 63 |

Table 2: Number of documents in the training/dev/test sets of the in-house and ACE05 datasets.

First we compare our neural network English RE models with the state-of-the-art RE models on the ACE05 English data. The ACE05 English data can be divided to 6 different domains: broadcast conversation (bc), broadcast news (bn), telephone conversation (cts), newswire (nw), usenet (un) and webblogs (wl). We apply the same data split in (Plank and Moschitti, 2013; Gormley et al., 2015; Nguyen and Grishman, 2016), which uses news (the union of bn and nw) as the training set, a half of bc as the development set and the remaining data as the test set.

We learn the model parameters using Adam (Kingma and Ba, 2015). We apply dropout (Srivastava et al., 2014) to the hidden layers to reduce overfitting. The development set is used for tuning the model hyperparameters and for early stopping.

In Table 1 we compare our models with the best models in (Gormley et al., 2015) and (Nguyen and Grishman, 2016). Our Bi-LSTM model outperforms the best model (single or ensemble) in (Gormley et al., 2015) and the best single model in (Nguyen and Grishman, 2016), without using any language-specific resources such as dependency

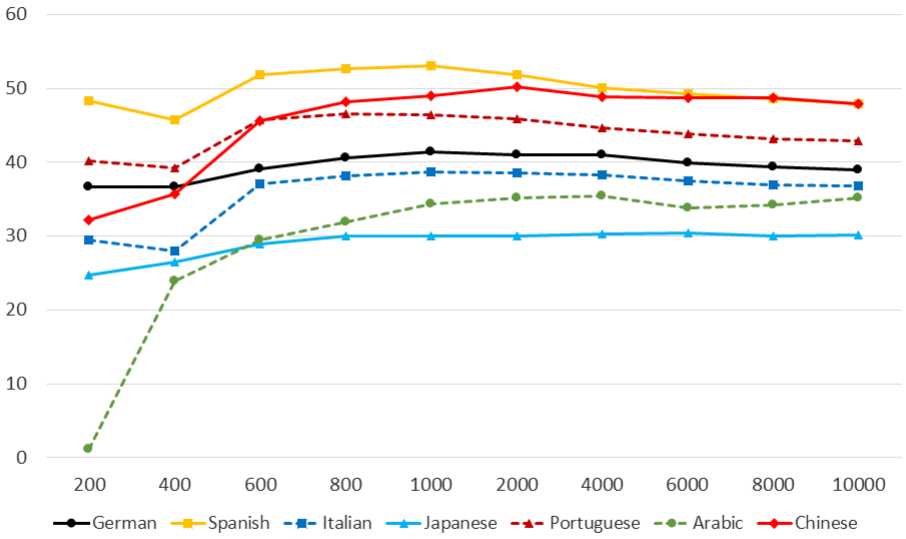


Figure 2: Cross-lingual RE performance ($F_1$ score) vs. dictionary size (number of bilingual word pairs for learning the mapping (4)) under the Bi-LSTM English RE model on the target-language development data.

| Model | In-House | | | ACE05 | | |
|---|---|---|---|---|---|---|
| | P | R | $F_1$ | P | R | $F_1$ |
| Pass-Through | 60.8 | 63.1 | 61.9 | 59.4 | 55.1 | 57.2 |
| Bi-LSTM | 66.5 | 67.7 | 67.1 | 65.1 | 65.8 | 65.5 |
| CNN | 63.4 | 68.8 | 66.0 | 61.7 | 67.1 | 64.3 |

Table 3: Performance of the supervised English RE models on the in-house and ACE05 English test data.

parsers.

While the data split in the previous works was motivated by domain adaptation, the focus of this paper is on cross-lingual model transfer, and hence we apply a random data split as follows. For the source language English and each target language, we randomly select 80% of the data as the training set, 10% as the development set, and keep the remaining 10% as the test set. The sizes of the sets are summarized in Table 2.

We report the Precision, Recall and $F_1$ score of the 3 neural network English RE models in Table 3. Note that adding an additional context layer with either Bi-LSTM or CNN significantly improves the performance of our English RE model, compared with the simple Pass-Through model. Therefore, we will focus on the Bi-LSTM model and the CNN model in the subsequent experiments.

### 5.3 Cross-Lingual RE Performance

We apply the English RE models to the 7 target languages across a variety of language families.

#### 5.3.1 Dictionary Size

The bilingual dictionary includes the most frequent target-language words and their translations in English. To determine how many word pairs are needed to learn an effective bilingual word embedding mapping for cross-lingual RE, we first evaluate the performance ($F_1$ score) of our cross-lingual RE approach on the target-language development sets with an increasing dictionary size, as plotted in Figure 2.

We found that for most target languages, once the dictionary size reaches 1K, further increasing the dictionary size may not improve the transfer performance. Therefore, we select the dictionary size to be 1K.

#### 5.3.2 Comparison of Different Mappings

We compare the performance of cross-lingual RE model transfer under the following bilingual word embedding mappings:

- **Regular-1K**: the regular mapping learned in (4) using 1K word pairs;

| Mapping | German | Spanish | Italian | Japanese | Portuguese | Arabic | Chinese | Average |
|---|---|---|---|---|---|---|---|---|
| Regular-1K | 41.4 | 53.0 | 38.7 | 30.0 | 46.4 | 34.4 | 49.0 | 41.8 |
| Orthogonal-1K | 41.0 | 49.2 | 35.7 | 27.1 | 44.0 | 32.1 | 47.2 | 39.5 |
| Semi-Supervised-1K | 38.1 | 46.7 | 31.2 | 25.3 | 43.1 | 37.3 | 47.1 | 38.4 |
| Unsupervised | 37.8 | 45.5 | 28.9 | 21.2 | 40.8 | 37.5 | 49.1 | 37.3 |

Table 4: Comparison of the performance ($F_1$ score) using different mappings on the target-language development data under the Bi-LSTM model.

- **Orthogonal-1K**: the orthogonal mapping with length normalization learned in (5) using 1K word pairs (in this case we train the English RE models with the normalized English word embeddings);
- **Semi-Supervised-1K**: the mapping learned with 1K word pairs and improved by the self-learning method in (Artetxe et al., 2017);
- **Unsupervised**: the mapping learned by the unsupervised method in (Artetxe et al., 2018).

The results are summarized in Table 4. The regular mapping outperforms the orthogonal mapping consistently across the target languages. While the orthogonal mapping was shown to work better than the regular mapping for the word translation task (Xing et al., 2015; Artetxe et al., 2016; Smith et al., 2017), our cross-lingual RE approach directly maps target-language word embeddings to the English embedding space without conducting word translations. Moreover, the orthogonal mapping requires length normalization, but we observed that length normalization adversely affects the performance of the English RE models (about 2.0 $F_1$ points drop).

We apply the vecmap toolkit[1] to obtain the semi-supervised and unsupervised mappings. The unsupervised mapping has the lowest average accuracy over the target languages, but it does not require a seed dictionary. Among all the mappings, the regular mapping achieves the best average accuracy over the target languages using a dictionary with only 1K word pairs, and hence we adopt it for the cross-lingual RE task.

#### 5.3.3 Performance on Test Data

The cross-lingual RE model transfer results for the in-house test data are summarized in Table 5 and the results for the ACE05 test data are summarized in Table 6, using the regular mapping learned with a bilingual dictionary of size 1K. In the tables, we also provide the performance of the supervised RE model (Bi-LSTM) for each target language, which is trained with a few hundred thousand tokens of manually annotated RE data in the target-language, and may serve as an upper bound for the cross-lingual model transfer performance.

Among the 2 neural network models, the Bi-LSTM model achieves a better cross-lingual RE performance than the CNN model for 6 out of the 7 target languages. In terms of *absolute* performance, the Bi-LSTM model achieves over 40.0 $F_1$ scores for German, Spanish, Portuguese and Chinese. In terms of *relative* performance, it reaches over 75% of the accuracy of the supervised target-language RE model for German, Spanish, Italian and Portuguese. While Japanese and Arabic appear to be more difficult to transfer, it still achieves 55% and 52% of the accuracy of the supervised Japanese and Arabic RE model, respectively, without using any manually annotated RE data in Japanese/Arabic.

We apply model ensemble to further improve the accuracy of the Bi-LSTM model. We train 5 Bi-LSTM English RE models initiated with different random seeds, apply the 5 models on the target languages, and combine the outputs by selecting the relation type labels with the highest probabilities among the 5 models. This **Ensemble** approach improves the single model by 0.6-1.9 $F_1$ points, except for Arabic.

#### 5.3.4 Discussion

Since our approach projects the target-language word embeddings to the source-language embedding space preserving the word order, it is expected to work better for a target language that has more similar word order as the source language. This has been verified by our experiments. The source language, English, belongs to the SVO (Subject, Verb, Object) language family where in a sentence the subject comes first, the verb second, and the object third. Spanish, Italian, Portuguese, German (in conventional typology) and Chinese

[1] https://github.com/artetxem/vecmap

| Model | German | | | Spanish | | | Italian | | | Japanese | | | Portuguese | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **P** | **R** | **$F_1$** | **P** | **R** | **$F_1$** | **P** | **R** | **$F_1$** | **P** | **R** | **$F_1$** | **P** | **R** | **$F_1$** |
| Bi-LSTM | 39.6 | 48.9 | 43.8 | 54.5 | 47.6 | 50.8 | 41.8 | 34.2 | 37.6 | 33.9 | 25.1 | 28.9 | 52.9 | 44.5 | 48.4 |
| CNN | 32.5 | 50.5 | 39.5 | 49.3 | 48.3 | 48.8 | 36.6 | 34.9 | 35.7 | 27.3 | 31.5 | 29.3 | 49.0 | 44.0 | 46.3 |
| Ensemble | 39.6 | 50.5 | 44.4 | 56.9 | 49.1 | 52.7 | 42.6 | 35.3 | 38.6 | 35.3 | 26.4 | 30.2 | 54.9 | 45.2 | 49.6 |
| *Supervised* | *59.3* | *56.4* | *57.8* | *68.4* | *65.4* | *66.8* | *51.4* | *48.3* | *49.8* | *52.7* | *52.0* | *52.4* | *64.0* | *61.3* | *62.6* |

Table 5: Performance of the cross-lingual RE approach on the in-house target-language test data.

| Model | Arabic | | | Chinese | | |
|---|---|---|---|---|---|---|
| | **P** | **R** | **$F_1$** | **P** | **R** | **$F_1$** |
| Bi-LSTM | 30.3 | 45.7 | 36.4 | 61.7 | 37.8 | 46.8 |
| CNN | 24.0 | 39.7 | 29.9 | 56.4 | 33.8 | 42.3 |
| Ensemble | 27.5 | 48.7 | 35.2 | 61.0 | 40.4 | 48.6 |
| *Supervised* | *70.0* | *69.1* | *69.5* | *66.9* | *69.4* | *68.1* |

Table 6: Performance of the cross-lingual RE approach on the ACE05 target-language test data.

also belong to the SVO language family, and our approach achieves over 70% relative accuracy for these languages. On the other hand, Japanese belongs to the SOV (Subject, Object, Verb) language family and Arabic belongs to the VSO (Verb, Subject, Object) language family, and our approach achieves lower relative accuracy for these two languages.

## 6 Related Work

There are a few weakly supervised cross-lingual RE approaches. Kim et al. (2010) and Kim and Lee (2012) project annotated English RE data to Korean to create weakly labeled training data via aligned parallel corpora. Faruqui and Kumar (2015) translates a target-language sentence into English, performs RE in English, and then projects the relation phrases back to the target-language sentence. Zou et al. (2018) proposes an adversarial feature adaptation approach for cross-lingual relation classification, which uses a machine translation system to translate source-language sentences into target-language sentences. Unlike the existing approaches, our approach does not require aligned parallel corpora or machine translation systems. There are also several multilingual RE approaches, e.g., Verga et al. (2016); Min et al. (2017); Lin et al. (2017), where the focus is to improve monolingual RE by jointly modeling texts in multiple languages.

Many cross-lingual word embedding models have been developed recently (Upadhyay et al., 2016; Ruder et al., 2017). An important application of cross-lingual word embeddings is to enable cross-lingual model transfer. In this paper, we apply the bilingual word embedding mapping technique in (Mikolov et al., 2013b) to cross-lingual RE model transfer. Similar approaches have been applied to other NLP tasks such as dependency parsing (Guo et al., 2015), POS tagging (Gouws and Søgaard, 2015) and named entity recognition (Ni et al., 2017; Xie et al., 2018).

## 7 Conclusion

In this paper, we developed a simple yet effective neural cross-lingual RE model transfer approach, which has very low resource requirements (a small bilingual dictionary with 1K word pairs) and can be easily extended to a new language. Extensive experiments for 7 target languages across a variety of language families on both in-house and open datasets show that the proposed approach achieves very good performance (up to 79% of the accuracy of the supervised target-language RE model), which provides a strong baseline for building cross-lingual RE models with minimal resources.

## Acknowledgments

We thank Mo Yu for sharing their ACE05 English data split and the anonymous reviewers for their valuable comments.

## References

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2289–2294, Austin, Texas. Association for Computational Linguistics.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462, Vancouver, Canada. Association for Computational Linguistics.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018. A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 789–798, Melbourne, Australia. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. In *International Conference on Learning Representations*.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 462–471, Gothenburg, Sweden. Association for Computational Linguistics.

Manaal Faruqui and Shankar Kumar. 2015. Multilingual open relation extraction using cross-lingual projection. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1351–1356. Association for Computational Linguistics.

Matthew R. Gormley, Mo Yu, and Mark Dredze. 2015. Improved relation extraction with feature-rich compositional embedding models. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1774–1784, Lisbon, Portugal. Association for Computational Linguistics.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. Bilbowa: Fast bilingual distributed representations without word alignments. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 748–756. JMLR Workshop and Conference Proceedings.

Stephan Gouws and Anders Søgaard. 2015. Simple task-specific bilingual word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1386–1390, Denver, Colorado. Association for Computational Linguistics.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *NEURAL NETWORKS*, 18(5-6):602–610.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1234–1244, Beijing, China. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *Proceedings of the ACL 2004 on Interactive Poster and Demonstration Sessions*, Stroudsburg, PA, USA. Association for Computational Linguistics.

Seokhwan Kim, Minwoo Jeong, Jonghoon Lee, and Gary Geunbae Lee. 2010. A cross-lingual annotation projection approach for relation detection. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 564–571, Stroudsburg, PA, USA. Association for Computational Linguistics.

Seokhwan Kim and Gary Geunbae Lee. 2012. A graph-based cross-lingual projection approach for weakly supervised relation extraction. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ACL '12, pages 48–53, Stroudsburg, PA, USA. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, ICLR '15.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 402–412. Association for Computational Linguistics.

Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2017. Neural relation extraction with multi-lingual attention. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 34–43. Association for Computational Linguistics.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Bilingual word representations with monolingual quality in mind. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 151–159. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for machine translation. *CoRR*, abs/1309.4168.

Bonan Min, Zhuolin Jiang, Marjorie Freedman, and Ralph Weischedel. 2017. Learning transferable representation for bilingual relation extraction via convolutional neural networks. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 674–684, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using LSTMs on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1116. Association for Computational Linguistics.

Thien Huu Nguyen and Ralph Grishman. 2016. Combining neural networks and log-linear models to improve relation extraction. In *Proceedings of IJCAI Workshop on Deep Learning for Artificial Intelligence (DLAI)*.

Jian Ni, Georgiana Dinu, and Radu Florian. 2017. Weakly supervised cross-lingual named entity recognition via effective annotation and representation projection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1470–1480. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Barbara Plank and Alessandro Moschitti. 2013. Embedding semantic similarity in tree kernels for domain adaptation of relation extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1498–1507, Sofia, Bulgaria. Association for Computational Linguistics.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2017. A survey of cross-lingual embedding models. *CoRR*, abs/1706.04902.

Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 626–634. Association for Computational Linguistics.

Samuel L. Smith, David H. P. Turban, Steven Hamblin, and Nils Y. Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Shyam Upadhyay, Manaal Faruqui, Chris Dyer, and Dan Roth. 2016. Cross-lingual models of word embeddings: An empirical comparison. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1661–1670. Association for Computational Linguistics.

Patrick Verga, David Belanger, Emma Strubell, Benjamin Roth, and Andrew McCallum. 2016. Multilingual relation extraction using compositional universal schema. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 886–896. Association for Computational Linguistics.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. ACE 2005 multilingual training corpus. Philadelphia: Linguistic Data Consortium.

Jiateng Xie, Zhilin Yang, Graham Neubig, Noah A. Smith, and Jaime Carbonell. 2018. Neural cross-lingual named entity recognition with minimal resources. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 369–379, Brussels, Belgium. Association for Computational Linguistics.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1006–1011, Denver, Colorado. Association for Computational Linguistics.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *The Journal of Machine Learning Research*, 3:1083–1106.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344. Dublin City University and Association for Computational Linguistics.

Bowei Zou, Zengzhuang Xu, Yu Hong, and Guodong Zhou. 2018. Adversarial feature adaptation for cross-lingual relation classification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 437–448, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

# Leveraging *2-hop* Distant Supervision from Table Entity Pairs for Relation Extraction

**Xiang Deng**
The Ohio State University
deng.595@buckeyemail.osu.edu

**Huan Sun**
The Ohio State University
sun.397@osu.edu

## Abstract

Distant supervision (DS) has been widely used to automatically construct (noisy) labeled data for relation extraction (RE). Given two entities, distant supervision exploits sentences that directly mention them for predicting their semantic relation. We refer to this strategy as 1-hop DS, which unfortunately may not work well for long-tail entities with few supporting sentences. In this paper, we introduce a new strategy named *2-hop* DS to enhance distantly supervised RE, based on the observation that there exist a large number of relational tables on the Web which contain entity pairs that share common relations. We refer to such entity pairs as *anchors* for each other, and collect all sentences that mention the anchor entity pairs of a given target entity pair to help relation prediction. We develop a new neural RE method REDS2 in the multi-instance learning paradigm, which adopts a hierarchical model structure to fuse information respectively from 1-hop DS and 2-hop DS. Extensive experimental results on a benchmark dataset show that REDS2 can consistently outperform various baselines across different settings by a substantial margin.[1]

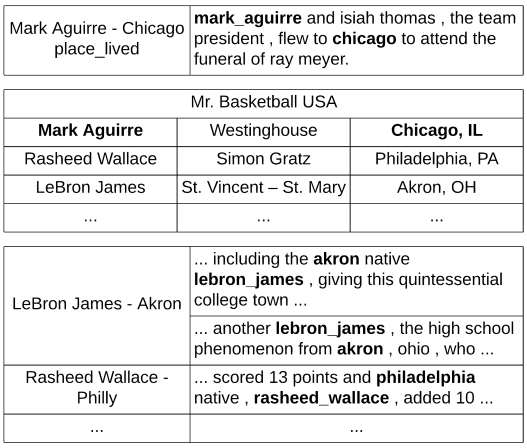


Figure 1: Illustration of *2-hop* distant supervision. The top panel shows a target entity pair, one sentence that mentions it, and the relation under study which cannot be inferred from the sentence. The middle gives part of a table from Wikipedia page "*Mr. Basketball USA*", where we can extract *anchors* for the target entity pair. The bottom shows some sentences that are associated with the anchors, which more clearly indicate the underinvestigated relation and can be utilized to extract relations between the target entity pair.

## 1 Introduction

Relation extraction (RE) aims to extract semantic relations between two entities from unstructured text and is an important task in natural language processing (NLP). Formally, given an entity pair $(e_1, e_2)$ from a knowledge base (KB) and a sentence (instance) that mentions them, RE tries to predict if a relation $r$ from a predefined relation set exists between $e_1$ and $e_2$. A special relation *NA* is used if none of the predefined relations holds.

Given that it is costly to construct large-scale labeled instances for RE, distant supervision (DS) has been a popular strategy to automatically construct (noisy) training data. It assumes that if two entities hold a relation in a KB, all sentences mentioning them express the same relation. Noticing that the DS assumption does not always hold and has the wrong labeling problem, many efforts including (Riedel et al., 2010; Hoffmann et al., 2011; Surdeanu et al., 2012) have adopted the multi-instance learning paradigm to tackle the challenge, and more recently, neural models with attention mechanism have been proposed to de-emphasize the noisy instances (Lin et al., 2016; Ji et al., 2017; Han et al., 2018). Such models tend to work well when there are a large number of sentences talking about the target entity pair (Lin et al., 2016).

[1]Our source code and datasets are at https://github.com/sunlab-osu/REDS2.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 410–420, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

However, we observe that there can be a large portion of entity pairs that have very few supporting sentences (e.g., nearly 75% of entity pairs in the Riedel et al. (2010) dataset only have one single sentence mentioning them), which makes distantly supervised RE even more challenging.

The conventional distant supervision strategy only exploits instances that directly mention a target entity pair, and because of this, we refer to it as *1-hop* distant supervision. On the other hand, there are a large number of Web tables that contain relational facts about entities (Cafarella et al., 2008; Venetis et al., 2011; Wang et al., 2012). Owing to the semi-structured nature of tables, we can extract from them sets of entity pairs that share common relations, and sentences mentioning these entity pairs often have similar semantic meanings. Under this observation, we introduce a new strategy named *2-hop* distant supervision: We define entity pairs that potentially have the same relation with a given target entity pair as *anchors*, which can be found through Web tables, and aim to fully exploit the sentences that mention those anchor entity pairs to augment RE for the target entity pair. Figure 1 illustrates the 2-hop DS strategy.

The intuition behind 2-hop DS is if the target entity pair holds a certain relation, one of its anchors is likely to have that relation too and at least one sentence mentioning the anchors should express the relation. Despite being noisy, the 2-hop DS can provide extra, informative supporting sentences for the target entity pair. One straightforward approach is to merge the two bags of sentences respectively derived from 1-hop and 2-hop DS as one single set and apply existing multi-instance learning models. However, the 2-hop DS strategy also has the wrong labeling problem that already exists in 1-hop DS. Simply mixing the two sets of sentences together may mislead the prediction, especially when there is a great disparity in their size. In this paper, we propose REDS2[2], a new neural relation extraction method in the multi-instance learning paradigm, and design a hierarchical model structure to fuse information from 1-hop and 2-hop DS. We evaluate REDS2 on a widely used benchmark dataset and show that it consistently outperforms various baseline models by a large margin.

We summarize our contributions as three-fold:

- We introduce 2-hop distant supervision as an extension to the conventional distant supervision, and leverage entity pairs in Web tables as anchors to find additional supporting sentences to further improve RE.
- We propose REDS2, a new neural relation extraction method based on 2-hop DS and has achieved new state-of-the-art performance in the benchmark dataset (Riedel et al., 2010).
- We release both our source code and an augmented benchmark dataset that has entity pairs aligned with those in Web tables, to facilitate future work.

[2]stands for relation extraction with 2-hop DS.

## 2 Related Work

**Distant Supervision.** One main drawback of traditional supervised relation extraction models (Zelenko et al., 2003; Mooney and Bunescu, 2006) is they require adequate amounts of annotated training data, which is time consuming and labor intensive. To address this issue, Mintz et al. (2009) proposes distant supervision (DS) to automatically label data by aligning plain text with Freebase. However, DS inevitably accompanies with the wrong labeling problem. To alleviate the noise brought by DS, Riedel et al. (2010) and Hoffmann et al. (2011) introduce multi-instance learning mechanism, which is originally used to combat the problem of ambiguously-labeled training data when predicting the activity of different drugs (Dietterich et al., 1997).

**Neural Relation Extraction.** Early stage relation extraction (RE) methods use features extracted by NLP tools and strongly rely on the quality of features. Due to the recent success of neural models in different NLP tasks, many researchers have investigated the possibility of using neural networks to build end-to-end relation extraction models. Zeng et al. (2014) uses convolutional neural network (CNN) to encode sentences, which is further improved through piecewise-pooling (Zeng et al., 2015). Adel and Schütze (2017) and Gupta et al. (2016) use neural networks for joint entity and relation extraction. More advanced network architectures like Tree-LSTM (Miwa and Bansal, 2016) and Graph Convolution Network (Vashishth et al., 2018) are also adopted to learn better representations by using syntactic features like dependency trees. Most recent models also incorporate neural attention technology (Lin et al., 2016) as an

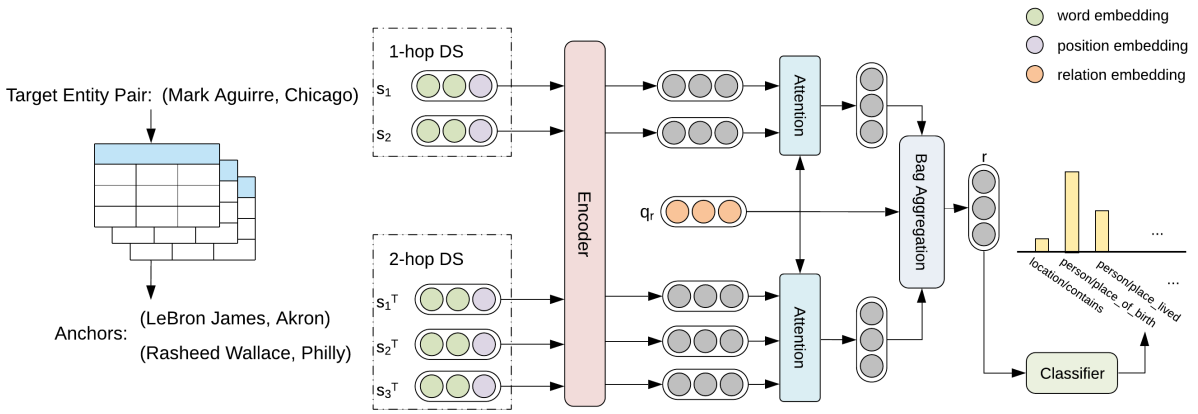


Figure 2: Overview of our method REDS2. REDS2 first obtains *anchors* of the target entity pair and constructs a 2-hop DS bag. Sentences in the 1-hop and 2-hop DS bag are individually encoded with a PCNN sentence encoder (Zeng et al., 2015). We then use selective attention and bag aggregation to get the final representation, based on which a classifier predicts scores for each candidate relation.

improvement to at-least-one multi-instance learning (Zeng et al., 2015). Han et al. (2018) further develops a hierarchical attention scheme to utilize the relation correlations and help predictions for long-tail relations.

**Web Table Understanding.** Aside from plain texts, there are large amounts of factual knowledge in the Web expressed in hundreds of millions of tables and other structured lists (Cafarella et al., 2008; Venetis et al., 2011), which have not been fully explored yet. Table understanding tries to match tables to KB and parse the schemas. Existing methods for table understanding mainly fall into two categories. One is based on local evidence (Venetis et al., 2011; Muñoz et al., 2014; Ritze et al., 2015). Given one table, the main idea is to first link cells to entities in KB. We can then use existing relations between linked entities to infer relations between columns and extract new facts by generalizing to all rows. However, this method requires a high overlap between table and KB, which is hampered by KB incompleteness. The other approach tries to leverage features extracted from the table header and column names (Ritze and Bizer, 2017; Cannaviccio et al., 2018). Unfortunately, a large portion of Web tables miss such metadata or contain limited information, and the second approach will fail in such cases. Although the focus of this paper is the RE task, we believe the idea of connecting Web tables and plain texts using DS can potentially benefit table understanding as well.

## 3 Methodology

Given a set of sentences $\mathrm{S} = \{s_1, s_2, ...\}$ and a target entity pair $(h, t)$, we will leverage the directly associated sentence bag $S_{h,t} \subseteq \mathrm{S}$ by 1-hop distant supervision (1-hop DS bag), and the table expanded sentence bag $S^T_{h,t} \subseteq \mathrm{S}$ by 2-hop distant supervision (2-hop DS bag), for relation extraction. $S_{h,t}$ contains all instances mentioning both $h$ and $t$, while $S^T_{h,t}$ is obtained indirectly through the anchors of $(h, t)$ found in Web tables. Following previous work (Riedel et al., 2010; Hoffmann et al., 2011), we adopt the multi-instance learning paradigm to measure the probability of $(h, t)$ having relation $r$.

Figure 2 gives an overview of our framework with three major components:

- **Table-aided Instance Expansion:** Given a target entity pair $(h, t)$, we find its *anchor* entity pairs $\{(h_1, t_1), (h_2, t_2), ...\}$ through Web tables. We define an anchor entity pair as two entities co-occurring with $(h, t)$ in some table columns at least once. $S^T_{h,t} = S_{h_1,t_1} \cup S_{h_2,t_2} \cup ...$ is then exploited to augment the directly associated bag $S_{h,t}$.

- **Sentence Encoding:** For each sentence $s$ in bag $S_{h,t}$ or $S^T_{h,t}$, a sentence encoder is used to obtain its semantic representation $\mathbf{s}$.

- **Hierarchical Bag Aggregation:** Once the embedding of each sentence is learned, we first use a sentence-level attention mechanism to get bag representation $\mathbf{h}$ and $\mathbf{h}^T$, and

then aggregate them for final relation prediction.

### 3.1 Table-aided Instance Expansion

Now we introduce how to construct the table expanded sentence bag $S^T_{h,t}$ for a given target entity pair $(h, t)$ by 2-hop distant supervision.

#### 3.1.1 Web Tables

Web tables have been found to contain rich facts of entities and relations. It is estimated that out of a total of 14.1 billion tables on the Web, 154 million tables contain relational data (Cafarella et al., 2008) and Wikipedia alone is the source of nearly 1.6 million relational tables (Bhagavatula et al., 2015). Columns of a Wikipedia table can be classified into one of the following data types: 'empty', 'named entity', 'number', 'date expression', 'long text' and 'other' (Zhang, 2017). Here we only focus on named entity columns (NE-columns) and the Wikipedia page title, which can be easily linked to KB entities. These entities can be further categorized as:

**A topic entity $e^t$** that the table is centered around. We refer to the Wikipedia article where the table is found and take the entity it describes as $e^t$.

**Subject entities $E^s = \{e^s_1, e^s_2, ...\}$** that can act as primary keys of the table. Following previous work on Web table analysis (Venetis et al., 2011), we select the leftmost NE-column as subject column and its entities as $E^s$.

**Body entities $E = \{e_{1,1}, e_{1,2}, ...\}$** that compose the rest of the table. All entities in non-subject NE-columns are considered as $E$.

#### 3.1.2 2-hop Distant Supervision

In the conventional distant supervision setting, each entity pair $(h, t)$ is associated with a bag of sentences $S_{h,t}$ that directly mention $h$ and $t$. The intuition behind 2-hop distant supervision is, if $(h_i, t_i)$ and $(h_j, t_j)$ potentially hold the same relation, we can treat them as **anchor entity pairs** for each other, and then use the 1-hop DS bag $S_{h_j,t_j}$ to help with the prediction for $(h_i, t_i)$ and vice versa. In this paper, we extract anchor entity pairs with the help of Web tables.

We notice that owing to the semi-structured nature of tables, (1) subject entities can usually be connected with the topic entity by the same relation. (2) Non-subject columns of a table usually have binary relationships to or are properties of the subject column. Body entities in the same column share common relations with their corresponding subject entities. For example, in Figure 1, the topic entity is "*Mr. Basketball USA*"; column 1 is the subject column and contains a list of winners of "*Mr. Basketball USA*"; column 2 and column 3 are high school and city of the subject entity.

Formally, we consider two entity pairs $(h_i, t_i)$ and $(h_j, t_j)$ as anchored if there exists a Web table such that either criterion below is met:

- $h_i = h_j = e^t$ and $t_i,\ t_j \in E^s$.
- $h_i \in E^s$ or $t_i \in E^s$, $(h_i, h_j)$ is in the same column (and so is $(t_i, t_j)$), and $(h_i, t_i)$ is in the same row (and so is $(h_j, t_j)$)

The 2-hop DS bag $S^T_{h,t}$ is then constructed as the union of $S_{h_i,t_i}$'s, where $(h_i, t_i)$ is an anchor entity pair of $(h, t)$.

### 3.2 Sentence Encoding

Given a sentence $s$ consisting of $n$ words $s = \{w_1, w_2, ..., w_n\}$, we use a neural network with an embedding layer and an encoding layer to obtain its low-dimensional vector representation.

#### 3.2.1 Embedding Layer

Each token is first fed into an embedding layer to embed both semantic and positional information.

**Word Embedding** maps words to vectors of real numbers which preserve syntactic and semantic information of words. Here we get a vector representation $\mathbf{w}_i \in \mathbb{R}^{k_w}$ for each word from a pre-trained word embedding matrix.

**Position Embedding** was proposed by Zeng et al. (2014). Position embedding is used to embed the positional information of each word relative to the head and tail mention. A position embedding matrix is learned in training to compute position representation $\mathbf{p}_i \in \mathbb{R}^{k_p \times 2}$.

Finally, we concatenate the word representation $\mathbf{w}_i$ and position representation $\mathbf{p}_i$ to build the input representation $\mathbf{x}_i \in \mathbb{R}^{k_i}$ (where $k_i = k_w + k_p \times 2$) for each word $w_i$.

#### 3.2.2 Encoding Layer

A sequence of input representations $\mathbf{x} = \{\mathbf{x}_1, \mathbf{x}_2, ...\}$ with a variable length is then fed through the encoding layer and converted to a fixed-sized sentence representation $\mathbf{s} \in \mathbb{R}^{k_h}$. There are many existing neural architectures that can serve as the encoding layer, such as CNN

(Zeng et al., 2014), PCNN (Zeng et al., 2015) and LSTM-RNN (Miwa and Bansal, 2016). We simply adopt PCNN here, which has been shown very powerful and efficient by a number of previous RE works.

PCNN is an extension to CNN, which first slides a convolution kernel with a window size $m$ over the input sequence to get the hidden vectors:

$$\mathbf{h}_i = \text{CNN}(\mathbf{x}_{i-\frac{m-1}{2}:i+\frac{m-1}{2}}), \tag{1}$$

A piecewise max-pooling is then applied over the hidden vectors:

$$\begin{aligned}
[\mathbf{s}^{(1)}]_j &= \max_{1\le i\le i_1}\{[\mathbf{h}_i]_j\},\\
[\mathbf{s}^{(2)}]_j &= \max_{i_i+1\le i\le i_2}\{[\mathbf{h}_i]_j\},\\
[\mathbf{s}^{(3)}]_j &= \max_{i_2+1\le i\le n}\{[\mathbf{h}_i]_j\},
\end{aligned} \tag{2}$$

where $i_1$ and $i_2$ are head and tail positions. The final sentence representation $\mathbf{s}$ is composed by concatenating these three pooling results $\mathbf{s} = [\mathbf{s}^{(1)}; \mathbf{s}^{(2)}; \mathbf{s}^{(3)}]$.

### 3.3 Hierarchical Bag Aggregation

After we get sentence representations $\{\mathbf{s}_1, \mathbf{s}_2, ...\}$ and $\{\mathbf{s}_1^T, \mathbf{s}_2^T, ...\}$ for $S$ and $S^T$, to fuse key information from these two bags, we adopt a hierarchical aggregation design to obtain the final representation $\mathbf{r}$ for prediction. We first get bag representation $\mathbf{h}$ and $\mathbf{h}^T$ using a sentence-level selective attention, and then employ a bag-level aggregation to compute $\mathbf{r}$.

#### 3.3.1 Sentence-level Selective Attention

Since the wrong labeling problem inevitably exists in both 1-hop and 2-hop distant supervision, here we use selective attention to assign different weights to different sentences given relation $r$ and de-emphasize the noisy sentences. The attention is caculated as follows:

$$\begin{aligned}
e_i &= \mathbf{q}_r^\top \mathbf{s}_i,\\
\alpha_i &= \frac{\exp(e_i)}{\sum_{j=1}^{n}\exp(e_j)},\\
\mathbf{h} &= \sum_{i=1}^{n}\alpha_i \mathbf{s}_i,
\end{aligned} \tag{3}$$

where $\mathbf{q}_r$ is a query vector assigned to relation $r$. $\mathbf{h}$ and $\mathbf{h}^T$ are computed respectively for the two bags $S$ and $S^T$.

#### 3.3.2 Bag-level Aggregation

Since 2-hop DS bag $S^T$ is collected indirectly through anchor entity pairs in Web tables, despite that it brings abundant information, it also contains a massive amount of noise. Thus treating $S^T$ equally as $S$ may mislead the prediction, especially when their sizes are extremely imbalanced.

To automatically decide how to balance between $S$ and $S^T$, we utilize information from $\mathbf{h}$, $\mathbf{h}^T$ and $\mathbf{q}_r$ to predict a weight $\beta$:

$$\beta = \sigma(\mathbf{W}[\mathbf{h}; \mathbf{h}^T; \mathbf{q}_r] + \mathbf{b}), \tag{4}$$

where vector $\mathbf{W}$ and scalar $\mathbf{b}$ are learnable variables and $\sigma$ is the sigmoid function. Next, $\beta$ is used as a weight to fuse information from 1-hop DS and 2-hop DS, determined by $S$ and $S^T$ of the current target entity pair and relation $r$. We then obtain the final representation $\mathbf{r}$ as:

$$\mathbf{r} = \beta\mathbf{h} + (1-\beta)\mathbf{h}^T, \tag{5}$$

Finally, we define the conditional probability $P(r|S, S^T, \theta)$ as follows,

$$P(r|S, S^T, \theta) = \frac{\exp(\mathbf{o}_r)}{\sum_{k=1}^{n_r}\exp(\mathbf{o}_k)} \tag{6}$$

where $\mathbf{o}$ is the score vector for current target entity pair having each relation,

$$\mathbf{o} = \mathbf{M}\mathbf{r} + \mathbf{d}, \tag{7}$$

here $\mathbf{M}$ is the representation matrix of relations, which shares weights with $\mathbf{q}_r$'s. $\mathbf{d}$ is a learnable bias term.

### 3.4 Optimization

We adopt the cross-entropy loss as the training objevtive function. Given a set of target entity pairs with relations $\pi = \{(h_1, t_1, r_1), (h_2, t_2, r_2), ...\}$, we define the loss function as follows:

$$J(\theta) = -\frac{1}{|\pi|}\sum_{i=1}^{|\pi|}\log P(r_i|S_{h_i,t_i}, S^T_{h_i,t_i}, \theta). \tag{8}$$

All models are trained with stochastic gradient descent (SGD) to minimize the objective function. The same sentence encoder is used to encode $S$ and $S^T$.

| | Train | | Test | |
|---|---|---|---|---|
| | Overall | Non-NA | Overall | Non-NA |
| # Entity Pairs | 291699 | 18144 | 96678 | 1761 |
| # Entity Pairs with $\|S^T\| > 0$ | 17565 | 6928 | 4832 | 824 |
| % Entity Pairs with $\|S^T\| > 0$ | 6.02 | **38.18** | 5.00 | **46.79** |
| mean $\|S\|$ | 1.69 | 5.24 | 1.62 | 2.78 |
| mean $\|S^T\|$ | 147.65 | 190.61 | 131.65 | 217.23 |

Table 1: Dataset statistics. We show statistics of entity pairs that hold non-NA relations separately from overall, as they are important relational facts to discover. Among non-NA entity pairs, 38.18% in training and 46.79% in testing have nonempty $S^T$, which respectively have 190.61 and 217.23 sentences on average.

# 4 Experiments

## 4.1 Datasets and Evaluation

We evaluate our model on the New York Times (NYT) dataset developed by Riedel et al. (2010), which is widely used in recent works. The dataset has 53 relations including a special relation NA which indicates none of the other 52 relations exists between the head and tail entity.

We use the WikiTable corpus collected by Bhagavatula et al. (2015) as our table source. It originally contains around 1.65M tables extracted from Wikipedia pages. Since the NYT dataset is already linked to Freebase, we perform entity linking on the table cells and the Wikipedia page titles using existing mapping from Wikipedia URL to Freebase MID (Machine Identifier). We then align the table corpus with NYT and construct $S^T$ for entity pairs as detailed in section 3.1. *For both training and testing, we only use entity pairs and sentences in the original NYT training data for table-aided instance expansion.* We set the max size of $S^T$ as 300, and randomly sample 300 sentences if $|S^T| > 300$. Statistics of our final dataset is summarized in Table 1. One can see that 38.18% and 46.79% of relational facts (i.e., entity pairs holding non-NA relations) respectively in the training and testing set can potentially benefit from leveraging 2-hop DS.

Following prior work (Mintz et al., 2009), we use the testing set for held-out evaluation, and evaluate models by comparing the predicted relational facts with those in Freebase. For evaluation, we rank the extracted relational facts based on model confidence and plot precision-recall curves. In addition, we also show the area under the curve (AUC) and precision values at specific recall rates to conduct a more comprehensive comparison.

| | |
|---|---|
| Window size $m$ | 3 |
| Sentence Representation Size $k^h$ | 230 |
| Word Dimension $k^w$ | 50 |
| Position Dimension $k^p$ | 5 |
| Pre-train Learning Rate $\lambda_P$ | 0.005 |
| Fine-tune Learning Rate $\lambda_F$ | 0.002 |
| Dropout Probability $p$ | 0.5 |

Table 2: Parameter settings in REDS2.

## 4.2 Baselines

We compare REDS2 with the following baselines:
**PCNN+ATT** (Lin et al., 2016). This model uses a PCNN encoder combined with selective attention over sentences. Since this is also the base block of our model, we also refer to it as BASE in this paper.
**PCNN+HATT** (Han et al., 2018). This is another PCNN based relation extraction model, where the authors use hierarchical attention to model the semantic correlations among relations.
**RESIDE** (Vashishth et al., 2018). It uses Graph Convolutional Networks (GCN) for sentence encoding, and also leverages relevant side information like relation alias and entity type.

Results of PCNN+HATT and RESIDE are directly taken from the code repositories released by the authors. For PCNN+ATT, we report results obtained by our reproduced model, which are close to those shown in (Lin et al., 2016). To simply verify the effectiveness of adding extra supporting sentences from 2-hop DS, we also compare the following vanilla method with PCNN+ATT:
**BASE+MERGE**. For each target entity pair $(h, t)$, we simply merge $S$ and $S^T$ as one sentence bag, and apply the trained PCNN+ATT (or, BASE) model.

## 4.3 Implementation Details

We preprocess the WikiTable corpus with PySpark to build index for anchor entity pairs. On a single machine with two 8-core E5 CPUs and 256 GB memory, this processing takes around 20 minutes.

We use word embeddings from (Lin et al., 2016) for initialization, which are learned by word2vec tool[3] on NYT corpus. The vocabulary is composed of words that appear more than 100 times in the corpus and words in an entity mention are concatenated as a single word.

[3]https://code.google.com/archive/p/word2vec/

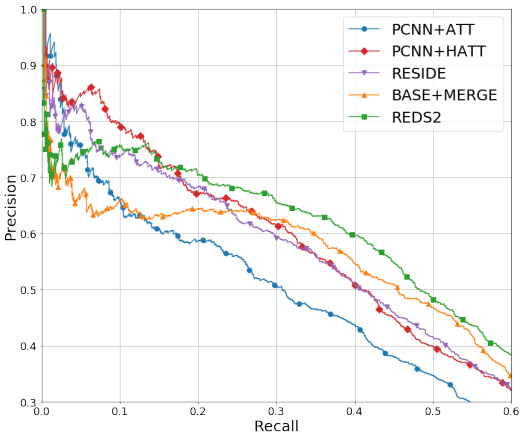


Figure 3: Precision-recall curves for the proposed model and various baselines.

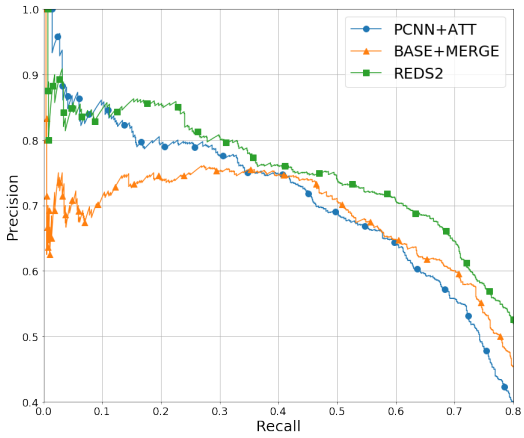


Figure 4: Precision-recall curves on the subset of test entity pairs whose $S^T$ is not empty, to better show the effect of hierarchical bag aggregation design.

| Method | P@0.1 | P@0.2 | P@0.3 | AUC |
|---|---|---|---|---|
| PCNN+ATT | 65.9 | 58.7 | 50.4 | 35.8 |
| PCNN+HATT | **79.3** | 67.2 | 61.6 | 42.0 |
| RESIDE | 73.6 | 68.4 | 59.5 | 41.5 |
| BASE+MERGE | 65.9 | 64.5 | 62.4 | 41.2 |
| **REDS2** | 75.9 | **70.4** | **65.5** | **44.7** |

Table 3: Comparison on Precision@recall and AUC.

To see the effect of 2-hop DS more directly, we set most parameters in REDS2 following Lin et al. (2016). Since the original NYT dataset only contains training and testing set, we randomly sample 20% training data for development. We first pre-train a PCNN+ATT model with only $S$ and sentence-level selective attention. This BASE model converges in around 100 epochs. We then fine-tune the entire model with $S^T$ and bag-level aggregation added, which can finish within 50 epochs. Some key parameter settings in REDS2 are summarized in Table 2.

In testing phase, inference using 2-hop DS is slower, because the average size of $S^T$ is about 100 times that of $S$. With single 2080ti GPU, one full pass of testing data takes around 37s using REDS2, compared with 12s using BASE model.

## 4.4 Results

### 4.4.1 Overall Evaluation Results

Evaluation results on all target entity pairs in testing set are shown in Figure 3 and Table 3, from which we make the following observations:

(1) Figure 3 shows all models obtain a reasonable precision when recall is smaller than 0.05. With the recall gradually increasing, the performance of models with 2-hop DS drops slower than those existing methods without. From Figure 3, we can see simply merging $S^T$ with $S$ in BASE+MERGE can boost the performance of basic PCNN+ATT model, and even achieves higher precision than state-of-the-art models like PCNN+HATT when recall is greater than 0.3. This demonstrates that models utilizing 2-hop DS are more robust and remain a reasonable precision when including more lower-ranked relational facts which tend to be more challenging to predict because of insufficient evidence.

(2) As shown in both Figure 3 and Table 3, REDS2 achieves the best results among all the models. Even when compared with PCNN+HATT and RESIDE which adopt extra relation hierarchy and side information from KB, our model still enjoys a significant performance gain. This is because our method can take advantage of the rich entity pair correlations in Web tables and leverage the extra information brought by 2-hop DS. We anticipate our REDS2 model can be further improved by using more advanced sentence encoders and extra mechanisms like reinforcement learning (Feng et al., 2018) and adversarial training (Wu et al., 2017), which we leave for future work.

### 4.4.2 Effect of Hierarchical Bag Aggregation

To further show the effect of our hierarchical bag aggregation design, here we also plot precision-recall curves in Figure 4 on a subset of entity pairs in the test set (i.e., 4832 in total according to Table 1) whose table expanded sentence bag $S^T$ is not empty.

One main challenge of using 2-hop DS is it brings more noise. As shown in Table 1, for en-

| Test Mode | SINGLE | | | | MULTIPLE | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | ONE | | | | TWO | | | | ALL | | | |
| Metric | P@0.1 | P@0.2 | P@0.3 | AUC | P@0.1 | P@0.2 | P@0.3 | AUC | P@0.1 | P@0.2 | P@0.3 | AUC | P@0.1 | P@0.2 | P@0.3 | AUC |
| PCNN+ATT | 57.3 | 53.0 | 41.3 | 30.0 | 72.9 | 66.4 | 57.5 | 39.0 | 75.3 | 69.7 | 63.8 | 45.9 | 80.5 | 68.6 | 64.2 | 48.6 |
| RESIDE | 66.9 | 60.5 | 51.8 | 36.2 | 79.7 | 70.4 | 57.4 | 42.1 | 77.7 | 76.1 | 67.6 | 46.5 | 83.2 | **80.3** | 73.2 | 51.9 |
| PCNN+HATT | **70.1** | 59.1 | 50.8 | 34.8 | 74.5 | 67.4 | 57.2 | 40.1 | 78.4 | 68.9 | 65.9 | 44.5 | **86.4** | 75.5 | 70.0 | 49.7 |
| BASE+MERGE | 62.2 | 58.5 | 55.2 | 35.6 | 71.4 | 74.9 | 73.2 | 51.3 | 70.0 | 71.8 | 73.9 | 52.0 | 70.7 | 71.1 | 73.2 | 52.0 |
| **REDS2** | 69.1 | **61.1** | **57.4** | **37.5** | **82.4** | **81.9** | **78.1** | **56.3** | **81.4** | **79.6** | **76.9** | **57.2** | 82.4 | 79.6 | **76.6** | **57.6** |

Table 4: Comparison on Precision@recall and AUC under different testing settings, detailed in Section 4.4.3.

| max $\|S^T\|$ | P@0.1 | P@0.2 | P@0.3 | AUC |
|---|---|---|---|---|
| 10 | 57.9 | 55.8 | 51.5 | 36.2 |
| 50 | 69.4 | 65.7 | 60.8 | 42.2 |
| 100 | 70.4 | 66.7 | 62.3 | 43.2 |
| 200 | 72.8 | 68.4 | 63.4 | 44.0 |
| 300 | **75.9** | **70.4** | **65.5** | **44.7** |

Table 5: Effect of the table expanded sentence bag size $|S^T|$ on Precision@recall and AUC.

| Relation: `country.capital` | |
|---|---|
| 1-hop | ... the golden gate bridge and the petronas towers in **kuala_lumpur**, **malaysia**, was experienced ... |
| 2-hop | a friend from **cardiff** , the capital city of **wales** , lives for complex ... |

Table 6: An example for case study, where the sentence with the highest attention weight is selected respectively from 1-hop and 2-hop sentence bag.

tity pair with nonempty $S^T$, the size of $S^T$ is usually tens of times the size of $S$. From Figure 4 we can see BASE+MERGE performs much worse compared with PCNN+ATT when recall is smaller than 0.2. This is because 2-hop DS bag tends to be much larger than 1-hop DS bag, and the model has a larger chance to attend to the noisy sentences obtained from 2-hop DS. While ignoring the information in its directly associated sentences. We alleviate this problem by introducing hierarchical structure to first aggregate the two sets separately and then weight and sum them together. The proposed `REDS2` model has a comparable precision with PCNN+ATT in the beginning and gradually outperform it.

### 4.4.3 Effect of Sentence Number

**Number of sentences from 1-hop DS**. In the originally testing set, there are 79176 entity pairs that are associated with only one sentence, out of which 1149 actually have relations. We hope our model can improve performance on these long-tail entities. Following Lin et al. (2016), we design the following test settings to evaluate the effect of sentence number: the "SINGLE" test setting contains all entity pairs that correspond to only one sentence; the "MULTIPLE" test setting contains the rest of entity pairs that have at least two sentences associated. We further construct the "ONE" testing setting where we randomly select one sentence for each entity pair; the "TWO" setting where we randomly select two sentences for each entity pair and the "ALL" setting where we use all the associated sentences from MULTIPLE. We use all sentences in $S^T$ for each entity pair if it is nonempty. Results are shown in Table 4, from which we can see that `REDS2` and BASE+MERGE have 25.0% and 18.7% improvements under AUC compared with PCNN+ATT in the SINGLE setting. Although the performance of all models generally improves as the sentence number increases in MULTIPLE setting, models leveraging 2-hop DS are more stable and have smaller changes. These observations indicate that 2-hop DS is helpful when information obtained by 1-hop DS is insufficient.

**Number of sentences from 2-hop DS**. We also evaluate how the number of sentences obtained by 2-hop DS will affect the performance of our proposed model. In Table 5, we show the performance of `REDS2` with different numbers of sentences sampled from $S^T$. We observe that: (1) Performance of `REDS2` improves as the number of sentences sampled increases. This shows that the selective attention over $S^T$ can effectively take advantage of the extra information from 2-hop DS while filtering out noisy sentences. (2) Even with 50 randomly sampled sentences, our model `REDS2` still has a higher AUC than all baselines in Table 3. This indicates information obtained by 2-hop DS is redundant, even a small portion can be beneficial to relation extraction. How to sample a representative set effectively is worth further exploring in future work.

### 4.5 RE for Entity Pairs with Empty 1-hop Sentence Bag

We observe that there are large amounts of entity pairs in the table corpus that have no associated sentences but have anchor entity pairs mentioned in the text corpus. By leveraging 2-hop distant supervision, we can do relation extraction for this set of entity pairs.

We extract a total number of 251917 entity pairs from the WikiTable dataset which do not exist in the NYT dataset but have at least one anchor entity pair that appear in the original NYT training data. We randomly sample 10000 examples and evaluate our trained model on them. Surprisingly, the relation extraction result is even better than the result on the NYT test data in Table 3, with an overall AUC of 54.7 and a P@0.3 of 71.1. This can be explained partly by two observations: (1) The table corpus generates higher-quality entity pairs, 18% of extracted entity pairs have non-NA relations, compared with only 1.8% in NYT test data. (2) The newly extracted entity pairs have 14 useful anchor entity pairs and 175 2-hop DS sentences on average, which give ample information for prediction. This study shows that for two entities that have no directly associated sentences, it is possible to utilize the 2-hop DS to predict their relations accurately.

### 4.6 Case Study and Error Analysis

In addition to the motivating example from the training set shown in Figure 1, we also demonstrate how 2-hop DS helped relation extraction using an example from the testing set in Table 6. As we can see, the sentence with the highest attention weight in 1-hop DS bag does not express the desired relation between the target entity pair whereas that in 2-hop DS bag clearly indicates the `country.capital` relation.

We also conduct an error analysis by analyzing examples where REDS2 gives worse predictions than BASE (e.g., assigns a lower score to a correct relation or a higher score to a wrong relation), and 50 examples with most disparity in the two methods' scores are selected. We find that 29 examples have wrong labels caused by KB incompleteness and our model in fact makes the right prediction. 11 examples are due to errors in column processing (e.g., errors in NE/subject column selection and entity linking), 9 are caused by anchor entity pairs with differet relations (e.g., **(Greece, Atlanta)** and **(Mexico, Xalapa)** are in the same table *"National Records in High Jump"* under columns **(Nation, Place)**, but only the latter has relation `location.contains`), and 1 is because of wrong information in the original table.

## 5 Conclusion and Future Work

This paper introduces 2-hop distant supervision for relation extraction, based on the intuition that entity pairs in relational Web tables often share common relations. Given a target entity pair, we define and find its anchor entity pairs via Web tables and collect all sentences that mention the anchor entity pairs to help relation prediction. We develop a new neural RE method `REDS2` in the multi-instance learning paradigm which fuses information from 1-hop DS and 2-hop DS using a hierarchical model structure, and substantially outperforms existing RE methods on a benchmark dataset. Interesting future work includes: (1) Given that information from 2-hop DS is redundant and noisy, we can explore smarter sampling and/or better bag-level aggregation methods to capture the most representative information. (2) Metadata in Web tables like headers and column names also contain rich information, which can be incorporated to further improve RE performance.

## 6 Acknowledgments

This research was sponsored in part by the Army Research Office under cooperative agreements NSF Grant IIS1815674, W911NF-17-1-0412, Fujitsu gift grant, and Ohio Supercomputer Center [8]. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Office or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notice herein.

## References

Heike Adel and Hinrich Schütze. 2017. Global normalization of convolutional neural networks for joint entity and relation classification. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1723–1729.

Chandra Sekhar Bhagavatula, Thanapon Noraset, and Doug Downey. 2015. Tabel: entity linking in web

tables. In *International Semantic Web Conference*, pages 425–441.

Michael J Cafarella, Alon Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. 2008. Webtables: exploring the power of tables on the web. *Proceedings of the VLDB Endowment*, 1(1):538–549.

Matteo Cannaviccio, Lorenzo Ariemma, Denilson Barbosa, and Paolo Merialdo. 2018. Leveraging wikipedia table schemas for knowledge graph augmentation. In *Proceedings of the 21st International Workshop on the Web and Databases*, page 5. ACM.

Thomas G Dietterich, Richard H Lathrop, and Tomás Lozano-Pérez. 1997. Solving the multiple instance problem with axis-parallel rectangles. *Artificial intelligence*, 89(1-2):31–71.

Jun Feng, Minlie Huang, Li Zhao, Yang Yang, and Xiaoyan Zhu. 2018. Reinforcement learning for relation classification from noisy data. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Pankaj Gupta, Hinrich Schütze, and Bernt Andrassy. 2016. Table filling multi-task recurrent neural network for joint entity and relation extraction. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2537–2547.

Xu Han, Pengfei Yu, Zhiyuan Liu, Maosong Sun, and Peng Li. 2018. Hierarchical relation extraction with coarse-to-fine grained attention. In *Proceedings of EMNLP*.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 541–550. Association for Computational Linguistics.

Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. 2017. Distant supervision for relation extraction with sentence-level attention and entity descriptions. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2124–2133.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1105–1116.

Raymond J Mooney and Razvan C Bunescu. 2006. Subsequence kernels for relation extraction. In *Advances in neural information processing systems*, pages 171–178.

Emir Muñoz, Aidan Hogan, and Alessandra Mileo. 2014. Using linked data to mine rdf from wikipedia's tables. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 533–542. ACM.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer.

Dominique Ritze and Christian Bizer. 2017. Matching web tables to dbpedia - A feature utility study. In *Proceedings of the 20th International Conference on Extending Database Technology, EDBT 2017, Venice, Italy, March 21-24, 2017.*, pages 210–221.

Dominique Ritze, Oliver Lehmberg, and Christian Bizer. 2015. Matching html tables to dbpedia. In *Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics*, page 10. ACM.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 455–465.

Shikhar Vashishth, Rishabh Joshi, Sai Suman Prayaga, Chiranjib Bhattacharyya, and Partha Talukdar. 2018. Reside: Improving distantly-supervised neural relation extraction using side information. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1257–1266.

Petros Venetis, Alon Halevy, Jayant Madhavan, Marius Paşca, Warren Shen, Fei Wu, Gengxin Miao, and Chung Wu. 2011. Recovering semantics of tables on the web. *Proceedings of the VLDB Endowment*, 4(9):528–538.

Jingjing Wang, Haixun Wang, Zhongyuan Wang, and Kenny Q Zhu. 2012. Understanding tables on the web. In *International Conference on Conceptual Modeling*, pages 141–155. Springer.

Yi Wu, David Bamman, and Stuart Russell. 2017. Adversarial training for relation extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1778–1783.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *Journal of machine learning research*, 3(Feb):1083–1106.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1753–1762.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344.

Ziqi Zhang. 2017. Effective and efficient semantic table interpretation using tableminer+. *Semantic Web*, 8(6):921–957.

# EntEval: A Holistic Evaluation Benchmark for Entity Representations

**Mingda Chen**[3*] **Zewei Chu**[1*] **Yang Chen**[2] **Karl Stratos**[4†] **Kevin Gimpel**[3]
[1]University of Chicago, IL, USA
[2]Ohio State University, OH, USA
[3]Toyota Technological Institute at Chicago, IL, USA
[4]Rutgers University, NJ, USA
{mchen,kgimpel}@ttic.edu,stratos@cs.rutgers.edu,zeweichu@uchicago.edu,chen.9279@osu.edu

## Abstract

Rich entity representations are useful for a wide class of problems involving entities. Despite their importance, there is no standardized benchmark that evaluates the overall quality of entity representations. In this work, we propose EntEval: a test suite of diverse tasks that require nontrivial understanding of entities including entity typing, entity similarity, entity relation prediction, and entity disambiguation. In addition, we develop training techniques for learning better entity representations by using natural hyperlink annotations in Wikipedia. We identify effective objectives for incorporating the contextual information in hyperlinks into state-of-the-art pretrained language models (Peters et al., 2018a) and show that they improve strong baselines on multiple EntEval tasks.[1]

## 1 Introduction

Entity representations play a key role in numerous important problems including language modeling (Ji et al., 2017), dialogue generation (He et al., 2017), entity linking (Gupta et al., 2017), and story generation (Clark et al., 2018). One successful line of work on learning entity representations has been learning *static* embeddings: that is, assign a unique vector to each entity in the training data (Gupta et al., 2017; Yamada et al., 2016, 2017). While these embeddings are useful in many applications, they have the obvious drawback of not accommodating unknown entities. Another limiting factor is the lack of an evaluation benchmark: it is often difficult to know which entity representations are better for which tasks.

We introduce EntEval: a carefully designed benchmark for holistically evaluating entity representations. It is a test suite of diverse tasks that require nontrivial understanding of entities, including entity typing, entity similarity, entity relation prediction, and entity disambiguation. Motivated by the recent success of contextualized word representations (henceforth: CWRs) from pretrained models (McCann et al., 2017; Peters et al., 2018a; Devlin et al., 2018; Yang et al., 2019; Liu et al., 2019b), we propose to encode the mention context or the description to dynamically represent an entity. In addition, we perform an in-depth comparison of ELMo and BERT-based embeddings and find that they show different characteristics on different tasks. We analyze each layer of the CWRs and make the following observations:

- The dynamically encoded entity representations show a strong improvement on the entity disambiguation task compared to prior work using static entity embeddings.
- BERT-based entity representations require further supervised training to perform well on downstream tasks, while ELMo-based representations are more capable of performing zero-shot tasks.
- In general, higher layers of ELMo and BERT-based CWRs are more transferable to EntEval tasks.

To further improve contextualized and descriptive entity representations (CER/DER), we leverage natural hyperlink annotations in Wikipedia. We identify effective objectives for incorporating the contextual information in hyperlinks and improve ELMo-based CWRs on a variety of entity related tasks.

*Equal contribution. Listed in alphabetical order.
†Work done while the author was at Toyota Technological Institute at Chicago.
[1]Data processing and evaluation scripts are available at https://github.com/ZeweiChu/EntEval



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 421–433, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

## 2 Related Work

EntEval and the training objectives considered in this work are built on previous works that involve reasoning over entities. We give a brief overview of relevant works.

**Entity linking/disambiguation.** Entity linking is a fundamental task in information extraction with a wealth of literature (He et al., 2013; Guo and Barbosa, 2014; Ling et al., 2015; Huang et al., 2015; Francis-Landau et al., 2016; Le and Titov, 2018; Martins et al., 2019). The goal of this task is to map a mention in context to the corresponding entity in a database. A natural approach is to learn entity representations that enable this mapping. Recent works focused on learning a fixed embedding for each entity using Wikipedia hyperlinks (Yamada et al., 2016; Ganea and Hofmann, 2017; Le and Titov, 2019). Gupta et al. (2017) additionally train context and description embeddings jointly, but this mainly aims to improve the quality of the fixed entity embeddings rather than using the context and description embeddings directly; we find that their context and description encoders perform poorly on EntEval tasks.

A closely related concurrent work by (Logeswaran et al., 2019) jointly encodes a mention in context and an entity description from Wikia to perform zero-shot entity linking. In contrast, here we seek to pretrain a general purpose entity representations that can function well either given or not given entity descriptions or mention contexts.

Other entity-related tasks involve entity typing (Yaghoobzadeh and Schütze, 2015; Murty et al., 2017; Del Corro et al., 2015; Rabinovich and Klein, 2017; Choi et al., 2018; Onoe and Durrett, 2019; Obeidat et al., 2019) and coreference resolution (Durrett and Klein, 2013; Wiseman et al., 2016; Lee et al., 2017; Webster et al., 2018; Kantor and Globerson, 2019).

**Evaluating pretrained representations.** Recent work has sought to evaluate the knowledge acquired by pretrained language models (Shi et al., 2016; Adi et al., 2017; Belinkov et al., 2017; Peters et al., 2018b; Conneau et al., 2018; Conneau and Kiela, 2018; Wang et al., 2018; Liu et al., 2019a; Chen et al., 2019a, *inter alia*). In this work, we focus on evaluating their capabilities in modeling entities.

Part of EntEval involves evaluating world knowledge about entities, relating them to fact checking (Vlachos and Riedel, 2014; Wang, 2017; Thorne et al., 2018; Yin and Roth, 2018; Chen et al., 2019b), and commonsense learning (Angeli and Manning, 2014; Bowman et al., 2015; Li et al., 2016; Mihaylov et al., 2018; Zellers et al., 2018; Trinh and Le, 2018; Talmor et al., 2019; Zellers et al., 2019; Sap et al., 2019; Rajani et al., 2019). Another related line of work is to integrate entity-related knowledge into the training of language models (Logan et al., 2019; Zhang et al., 2019; Sun et al., 2019).

**Contextualized word representations.** Contextualized word representations and pretrained language representation models, such as ELMo (Peters et al., 2018a) and BERT (Devlin et al., 2018), are powerful pretrained models that have been shown to be effective for a variety of downstream tasks such as text classification, sentence relation prediction, named entity recognition, and question answering. Recent work has sought to evaluate the knowledge acquired by such models (Shi et al., 2016; Adi et al., 2017; Belinkov et al., 2017; Conneau et al., 2018; Conneau and Kiela, 2018; Liu et al., 2019a). In this work, we focus on evaluating their capabilities in modeling entities.

## 3 EntEval

We are interested in two approaches: contextualized entity representations (henceforth: CER) and descriptive entity representations (henceforth: DER), both encoding fixed-length vector representations for entities.

The contextualized entity representations encodes an entity based on the context it appears regardless of whether the entity is seen before. The motivation behind contextualized entity representations is that we want an entity encoder that does not depend on entries in a knowledge base, but is capable of inferring knowledge about an entity from the context it appears.

As opposed to contextualized entity representations, descriptive entity representations do rely on entries in Wikipedia. We use a model-specific function $f$ to obtain a fixed-length vector representation from the entity's textual description.

To evaluate CERs and DERs, we propose a wide range of entity related tasks. Since our purpose is for examining the learned entity representations, we only use a linear classifier and freeze the entity representations when performing the follow-

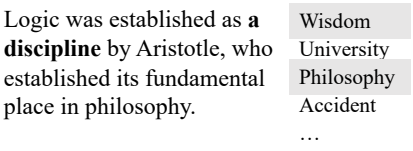


Figure 1: An example taken from ET. Targeted entity mention is bold. Candidate categories are on the right. Gold standard categories are in gray.

ing tasks. Unless otherwise noted, when the task involves a pair of entities, the input to the classifier are the entity representations $x_1$ and $x_2$, concatenated with their element-wise product and absolute difference: $[x_1, x_2, x_1 \odot x_2, |x_1 - x_2|]$. This input format has been used in SentEval (Conneau and Kiela, 2018).

The datasets used in EntEval tasks are summarized in table 1. It shows the number of instances in train/valid/test split for each dataset, and the number of target classes if this is a classification task. We describe the proposed tasks in the following subsections.

### 3.1 Entity Typing (ET)

The task of entity typing (ET) is to assign types to an entity given only the context of the entity mention. ET is context-sensitive, making it an effective approach to probe the knowledge of context encoded in pretrained representations. For example, in the sentence "Bill Gates has donated billions to eradicate malaria", "Bill Gates" has the type of "philanthropist" instead of "inventor" (Choi et al., 2018). In this task, we will contextualized entity representations, followed by a linear layer to make predictions. We use the annotated ultra-fine entity typing dataset of Choi et al. (2018) with standard data splits. As shown in Figure 1, there can be multiple labels for an instance. We use binary log loss for training using all positive and negative entity types, and report $F_1$ score. Thresholds are tuned based on validation set accuracy.

### 3.2 Coreference Arc Prediction (CAP)

Given two entities and the associated context, the task is to determine whether they refer to the same entity. Solving this task may require the knowledge of entities. For example, in the sentence "Revenues of \$14.5 billion were posted by $\text{Dell}_1$. The $\text{company}_1$ ...", there is no prior context of "Dell", so having known "Dell" is a company instead of the people "Michael Dell" will surely benefit the model (Durrett and Klein, 2014). Unlike other tasks, coreference typically involves longer context. To restrict the effect of broad context, we only keep two groups of coreference arcs from smaller context. One includes mentions that are in the same sentence ("same") for examining the model capability of encoding local context. The other includes mentions that are in consecutive sentences ("next") for the broader context. We create this task from the PreCo dataset (Chen et al., 2018), which has mentions annotated even when they are not part of coreference chains. We filter out instances in which both mentions are pronouns. All non-coreferent mention pairs are considered to be negative samples.

To make this task more challenging, for each instance we compute cosine similarity of mentions by averaging GloVe word vectors. We group the instances into bins by cosine similarity, and randomly select the same number of positive and negative instances from each bin to ensure that models do not solve this task by simply comparing similarity of mention names.

We use the contextualized entity representations of the two mentions to infer coreference arcs with supervised training and report the averaged accuracy of "same" and "next".

### 3.3 Entity Factuality Prediction (EFP)

The entity factuality prediction (EFP) task involves determining the correctness of statements regarding entities. We use the manually-annotated FEVER dataset (Thorne et al., 2018) for this task. FEVER is a task to verify whether a statement is supported by evidences. The original FEVER dataset includes three classes, namely "Supports", "Refutes", and "NotEnoughInfo" and evidences are additionally available for each instance. As our purpose is to examine the knowledge encoded in entity representations, we discard the last category ("NotEnoughInfo") and the evidence. In rare cases, instances in FEVER may include multiple entity mentions, so we randomly pick one. We randomly sample 10000, 2000, and 2000 instances for our training, validation, and test sets, respectively.

In this task, entity representations can be obtained either by contextualized entity representations or descriptive entity representations. In practice, we observe descriptive entity representations give better performance, which presumably is be-

| | CAP | | CERP | EFP | ET | KORE | WikiSRS | | ERT | Rare | CoNLL |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | same | next | | | | | Rel | Sim | | | |
| #train | 3982 | 3982 | 10000 | 10000 | 1998 | N/A | N/A | N/A | 3130 | 10000 | 18538 |
| #valid | 3806 | 3828 | 2000 | 2000 | 1998 | N/A | N/A | N/A | 6260 | 4000 | 4790 |
| #test | 3938 | 3850 | 2000 | 2000 | 1998 | $20 \times 20$ | 688 | 688 | 6260 | 4000 | 4481 |
| #classes | 2 | | 2 | 2 | 10331 | N/A | N/A | N/A | 626 | 4 | up to 30 |

Table 1: Statistics of datasets used in EntEval tasks. CAP: coreference arc prediction, CERP: contexualized entity relationship prediction, EFP: entity factuality prediction, ET: entity typing, ESR: entity similarity and relatedness, ERT: entity relationship typing, NED: named entity disambiguation, Rare: rare entity prediction, CoNLL: CoNLL-YAGO named entity disambiguation.

*REFUTES*: The **New York City Landmarks Preservation Commission** consists of zero commissioners.
*SUPPORTS*: **TD Garden** has held Bruins games.

Figure 2: Two examples from the EFP.

*TRUE*: **Gin and vermouth** can **make a martini**
*FALSE*: **Connecticut** is not **a state**

Figure 3: Examples from the CERP.

| Score | Entity Name |
|---|---|
| - | Apple Inc. |
| 20 | Steve Jobs |
| ... | ... |
| 11 | Microsoft |
| ... | ... |
| 1 | Ford Motor Company |

Table 2: An example from KORE. The task is to rank the candidate entities by similarity.

cause these statements are more similar to descriptions than entity mentions. As shown in Figure 2, without providing additional evidences, solving this task requires knowledge of entities encoded in representations. We directly use entity representations as input to the classifier.

### 3.4 Contexualized Entity Relationship Prediction (CERP)

The task of contexualized entity relationship prediction (CERP) modeling determines the connection between two entities appeared in the same context. We use sentences from ConceptNet (Speer et al., 2017) with automatically parsed mentions and templates used to construct the dataset. We filter out non-English concepts and relations such as 'related', 'translation', 'synonym', and 'likely to find' since we seek to evaluate more complicated knowledge of entities encoded in representations. We further filter out non-entity mentions and entities with type 'DATE', 'TIME', 'PERCENT', 'MONEY', 'QUANTITY', 'ORDINAL', and 'CARDINAL' according to SpaCy (Honnibal and Montani, 2017). After filtering, we have 13374 assertions.

Negative samples are generated based on the following rules:

1. For each relationship, we replace an entity with similar negative entities based on cosine similarity of averaged GloVe embeddings (Pennington et al., 2014).
2. We change the relationship in positive samples from affirmation to negation (e.g., 'is' to 'is not'). These serve as negative samples.
3. We further sample positive samples from (1) in an attempt to prevent the 'not' token from being biased towards negative samples. Therefore, for negative samples we get from (1), we change the relationship from affirmation to negation as in (2) to get positive samples.

For example, let 'A is B' be the positive sample. (1) changes it to 'C is B' which serves as a negative sample and (2) changes it to 'A is not B' as another negative sample. (3) changes it to 'C is not B' as a positive example. In the end, we randomly sample 7000 instances from each class. This ends up yielding a 10000/2000/2000 train/dev/test dataset. As shown in Figure 3, this task cannot be solved by relying on surface form of sentences, instead it requires the input representations to encode knowledge of entities based on the context.

We use contextualized entity representations in this task.

### 3.5 Entity Similarity and Relatedness (ESR)

Given two entities with their descriptions from Wikipedia, the task is to determine their similarity or relatedness. After the entity descriptions are encoded into vector representations, we compute their cosine similarity as predictions. We use the KORE (Hoffart et al., 2012) and Wik-

iSRS (Newman-Griffis et al., 2018) datasets in this task. Since the original datasets only provide entity names, we automatically add Wikipedia descriptions to each entity and manually ensure that every entity is matched to a Wikipedia description. We use Spearman's rank correlation coefficient between our computed cosine similarity and the gold standard similarity/relatedness scores to measure the performance of entity representations.

As KORE does not provide similarity scores of entity pairs, but simply ranks the candidate entities by their similarities to a target entity, we assign scores from 20 to 1 accordingly to each entity in the order of similarity. Table 2 shows an example from KORE. The fact that "Apple Inc." is more related to "Steve Jobs" than "Microsoft" requires multiple steps of inference, which motivates this task. Since the predictor we use is cosine similarity, which does not introduce additional parameters, we directly use encoded representations on the test set without any supervised training.

### 3.6 Entity Relationship Typing (ERT)

As another popular resource for common knowledge, we consider using Freebase (Bollacker et al., 2008) for probing the encoded knowledge by classifying the types of relations between pair of entities. First, we extract entity relation tuples (entity1, relation, entity2) from Freebase and then filter out easy tuples based on training a classifier using averaged GloVe vectors of entity names as input, which leaves us 626 types of relations, including "internet.website.owner", "film.film_art_director.films_art_directed", and "comic_books.comic_book_series.genre". We randomly sample 5 instances for each relation type to form our training set and 10 instances per type the for validation and test sets. We use Wikipedia descriptions for each entity in the pair whose relation we are predicting and we use descriptive entity representations for each entity with supervised training.

### 3.7 Named Entity Disambiguation (NED)

Named entity disambiguation is the task of linking a named-entity mention to its corresponding instance in a knowledge base such as Wikipedia. In this task, we consider CoNLL-YAGO (CoNLL; Hoffart et al., 2011) and Rare Entity Prediction (Rare; Long et al., 2017).

For CoNLL-YAGO, following Hoffart et al. (2011) and Yamada et al. (2016), we used the 27,816 mentions with valid entries in the knowledge base. For each entity mention $m$ in its context, we generate a set of (at most) its top 30 candidate entities $C_m = \{c_j\}$ using CrossWikis (Spitkovsky and Chang, 2012). Some gold standard candidates $c$ are not present in CrossWikis, so we set the prior probability $p_{\text{prior}}(y)$ for those to 1e-6 and normalize the resulting priors for the candidate entities. When adding Wikipedia descriptions, we manually ensure gold standard mentions are attached to a description, however, we discard candidate mentions that cannot be aligned to a Wikipedia page. We use contextualized entity representations for entity mentions and use descriptive entity representations for candidate entities. Training minimizes binary log loss using all negative examples. At test time, we use $\arg\max_{c \in C_m} [p_{\text{prior}}(c) + p_{\text{classifier}}(c)]$ as the prediction. We note that directly using prior as predictions yields an accuracy of 58.2%.




Figure 4: An example from CoNLL-YAGO. Only four candidates are shown due to space constraints. The target mention is underlined. Sentences in gray are Wikipedia descriptions. The gold standard is boldfaced.

Long et al. (2017) introduce the task of *rare entity prediction*. The task has a similar format to CoNLL-YAGO entity linking. Given a document with a blank in it, the task is to select an entity from a provided list of entities with descriptions. Only rare entities are used in this dataset so that performing well on the task requires the ability to effectively represent entity descriptions. We randomly select 10k/4k/4k examples to construct train/valid/test sets. For simplicity, we only keep instances with four candidate entities.

Figure 4 shows an example from CoNLL-YAGO, where the "China" in context has many deceptive meanings. Here the candidate "China" has exact string match of the entity name but it should not be selected as it is an after-game report on soccer. To match the entities, this task requires both effective contextualize entity representations and descriptive entity representation.

Practically, we encode the context using CER to be $x_1$, and encode each entity description using DER to be $x_2$, and pass $[x_1, x_2, x_1 \odot x_2, |x_1 - x_2|]$ to a linear model to predict whether it is the correct entity to fill in. The model is trained with cross entropy loss.

## 4 Methods

We first describe how we define encoders for contextualized entity representations (Section 4.1) and descriptive entity representations (Section 4.2), then we discuss how we train new encoders tailored to capture information from the hyperlink structure of Wikipedia (Section 4.3).

### 4.1 Encoders for Contextualized Entity Representations

For defining these encoders, we assume we have a sentence $s = (w_1, \ldots, w_T)$ where span $(w_i, \ldots, w_j)$ refers to an entity mention. When using ELMo, we first encode the sentence: $(c_1, \ldots, c_T) = \text{ELMo}(w_1, \cdots, w_T)$, and we use the average of contextualized hidden states corresponding to the entity span as the contextualized entity representation. That is, $f_{\text{ELMo}}(w_{1:T}, i, j) = \frac{\sum_{k=i}^{j} c_k}{j-i+1}$.

With BERT, following Onoe and Durrett (2019), we concatenate the full sentence with the entity mention, starting with [CLS] and separating the two by [SEP], i.e., $[\text{CLS}], w_1, \ldots, w_T, [\text{SEP}], w_i, \ldots, w_j, [\text{SEP}]$. We encode the full sequence using BERT and use the output from the [CLS] token as the entity mention representation.

### 4.2 Encoders for Descriptive Entity Representations

We encode an entity description by treating the entity description as a sentence, and use the average of the hidden states from ELMo as the entity description representation. With BERT, we use the output from the [CLS] token as the description representation.

### 4.3 Hyperlink-Based Training

An entity mentioned in a Wikipedia article is often linked to its Wikipedia page, which provides a useful description of the mentioned entity. The same Wikipedia page may correspond to many different entity mentions. Likewise, the same entity mention may refer to different Wikipedia pages depending on its context. For instance, as shown in Figure 5, based on the context, "France" is linked to the Wikipedia page of "France national football team" instead of the country. The specific entity in the knowledge base can be inferred from the context information. In such cases, we believe Wikipedia provides valuable complementary information to the current pretrained CWRs such as BERT and ELMo.

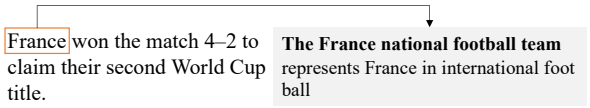


Figure 5: An example of hyperlinks in Wikipedia. "France" is linked to the Wikipedia page of "France national football team" instead of the country France.

To incorporate such information during training, we automatically construct a hyperlink-enriched dataset from Wikipedia that we will refer to as WIKIENT. Prior work has used similar resources (Singh et al., 2012; Gupta et al., 2017), but we aim to standardize the process and will release the dataset.

The WIKIENT dataset consists of sentences with contextualized entity mentions and their corresponding descriptions obtained via hyperlinked Wikipedia pages. When processing descriptions, we only keep the first 100 word tokens at most as the description of a Wikipedia page; similar truncation has been done in prior work (Gupta et al., 2017). For context sentences, we remove those without hyperlinks from the training data and duplicate those with multiple hyperlinks. We also remove context sentences for which we cannot find matched Wikipedia descriptions. These processing steps result in a training set of approximately 92 million instances and over 3 million unique entities.

We define a hyperlink-based training objective and add it to ELMo. In particular, we use contextualized entity representations to decode the hyperlinked Wikipedia description, and also use the descriptive entity representations to decode the linked context. We use bag-of-words decoders in both decoding processes. More specifically, given a context sentence $x_{1:T_x}$ with mention span $(i, j)$ and a description sentence $y_{1:T_y}$, we use the same bidirectional language modeling

loss $l_{\text{lang}}(x_{1:T_x}) + l_{\text{lang}}(y_{1:T_y})$ in ELMo where

$$l_{\text{lang}}(u_{1:T}) = -\sum_{t=1}^{T} \log p(u_{t+1}|u_1, \ldots, u_t) + \log p(u_{t-1}|u_t, \ldots, u_T)$$

and $p$ is defined by the ELMo parameters. In addition, we define the two bag-of-words reconstruction losses:

$$l_{\text{ctx}} = -\sum_t \log q(x_t | f_{\text{ELMo}}([\text{BOD}]y_{1:T_y}, 1, T_y))$$

$$l_{\text{desc}} = -\sum_t \log q(y_t | f_{\text{ELMo}}([\text{BOC}]x_{1:T_x}, i, j))$$

where [BOD] and [BOC] are special symbols prepended to sentences to distinguish descriptions from contexts. The distribution $q$ is parameterized by a linear layer that transforms the conditioning embedding into weights over the vocabulary. The final training loss is

$$l_{\text{lang}}(x_{1:T_x}) + l_{\text{lang}}(y_{1:T_y}) + l_{\text{ctx}} + l_{\text{desc}} \quad (1)$$

Same as the original ELMo, each log loss is approximated with negative sampling (Jean et al., 2015). We write EntELMo to denote the model trained by Eq. (1). When using EntELMo for contextualized entity representations and descriptive entity representations, we use it analogously to ELMo.

## 5 Experiments

### 5.1 Setup

As a baseline for hyperlink-based training, we train EntELMo on the WIKIENT dataset with only a bidirectional language model loss. Due to the limitation of computational resources, both variants of EntELMo are trained for one epoch (3 weeks time) with smaller dimensions than ELMo. We set the hidden dimension of each directional long short-term memory network (LSTM; Hochreiter and Schmidhuber, 1997) layer to be 600, and project it to 300 dimensions. The resulting vectors from each layer are thus 600 dimensional. We use 1024 as the negative sampling size for each positive word token. For bag-of-words reconstruction, we randomly sample at most 50 word tokens as positive samples from the the target word tokens. Other hyperparameters are the same as ELMo. EntELMo is implemented based on the official ELMo implementation.[2]

[2]Our implementation is available at https://github.com/mingdachen/bilm-tf

As a baseline for contextualized and descriptive entity representations, we use GloVe word averaging of the entity mention as the "contextualized" entity representation, and use word averaging of the truncated entity description text as its description representation. We also experiment two variants of EntELMo, namely EntELMo w/o $l_{\text{ctx}}$ and EntELMo with $l_{\text{etn}}$. For second variant, we replace $l_{\text{ctx}}$ with $l_{\text{etn}}$, where we only decode entity mentions instead of the whole context from descriptions. We lowercased all training data as well as the evaluation benchmarks.

We evaluate the transferrability of ELMo, EntELMo, and BERT by using trainable mixing weights for each layer. For ELMo and EntELMo, we follow the recommendation from Peters et al. (2018a) to first pass mixing weights through a softmax layer and then multiply the weighted-summed representations by a scalar. For BERT, we find it better to just use unnormalized mixing weights. In addition, we investigate per-layer performance for both models in Section 6.

### 5.2 Results

Table 3 shows the performance of our models on the EntEval tasks. Our findings are detailed below:

- Pretrained CWRs (ELMo, BERT) perform the best on EntEval overall, indicating that they capture knowledge about entities in contextual mentions or as entity descriptions.
- BERT performs poorly on entity similarity and relatedness tasks. Since this task is zero-shot, it validates the recommended setting of finetuning BERT (Devlin et al., 2018) on downstream tasks, while the embedding of the [CLS] token does not necessarily capture the semantics of the entity.
- BERT Large is better than BERT Base on average, showing large improvements in ERT and NED. To perform well at ERT, a model must either glean particular relationships from pairs of lengthy entity descriptions or else leverage knowledge from pretraining about the entities considered. Relatedly, performance on NED is expected to increase with both the ability to extract knowledge from descriptions and by starting with increased knowledge from pretraining. The Large model appears to be handling these capabilities better than the Base model.
- EntELMo improves over the EntELMo baseline (trained without the hyperlinking loss) on some

| | CAP | CERP | EFP | ET | ESR | ERT | NED | Average |
|---|---|---|---|---|---|---|---|---|
| GloVe | 71.9 | 52.6 | 67.0 | 10.3 | 50.9 | 40.8 | 41.2 | 47.8 |
| BERT Base | **80.6** | 65.6 | 74.8 | 32.0 | 28.8 | 42.2 | 50.6 | 53.5 |
| BERT Large | 79.1 | **66.9** | **76.7** | 32.3 | 32.6 | **48.8** | **54.3** | 55.8 |
| ELMo | 80.2 | 61.2 | 75.8 | **35.6** | 60.3 | 46.8 | 51.6 | **58.8** |
| EntELMo baseline | 78.0 | 59.6 | 71.5 | 31.3 | **61.6** | 46.5 | 48.5 | 56.7 |
| EntELMo | 76.9 | 59.9 | 72.4 | 32.2 | 59.7 | 45.7 | 49.0 | 56.5 |
| EntELMo w/o $l_{\text{ctx}}$ | 73.5 | 59.4 | 71.1 | 33.2 | 53.3 | 44.6 | 48.9 | 54.9 |
| EntELMo w/ $l_{\text{etn}}$ | 76.2 | 60.4 | 70.9 | 33.6 | 49.0 | 42.9 | 49.3 | 54.6 |

Table 3: Performances of entity representations on EntEval tasks. Best performing model in each task is boldfaced. CAP: coreference arc prediction, CERP: contexualized entity relationship prediction, EFP: entity factuality prediction, ET: entity typing, ESR: entity similarity and relatedness, ERT: entity relationship typing, NED: named entity disambiguation. EntELMo baseline is trained on the same dataset as EntELMo but not using the hyperlink-based training. EntELMo w/ $l_{\text{etn}}$ is trained with a modified version of $l_{\text{ctx}}$, where we only decode entity mentions instead of the whole context.

| | Rare | | CoNLL | | ERT | |
|---|---|---|---|---|---|---|
| | Des. | Name | Des. | Name | Des. | Name |
| ELMo | 38.1 | 36.7 | 63.4 | 71.2 | 46.8 | 31.5 |
| BERT Base | 42.2 | 36.6 | 64.7 | 74.3 | 42.2 | 34.3 |
| BERT Large | 48.8 | 44.0 | 64.6 | 74.8 | 48.8 | 32.6 |

Table 4: Accuracies (%) in comparing the use of description encoder (Des.) to entity name (Name).

| | CoNLL |
|---|---|
| ELMo | 71.2 |
| Gupta et al. (2017) | 65.1 |
| Deep ED | 66.7 |

Table 5: Accuracies (%) on CoNLL-YAGO with static or non-static entity representations.

tasks but suffers on others. The hyperlink-based training helps on CERP, EFP, ET, and NED. Since the hyperlink loss is closely-associated to the NED problem, it is unsurprising that NED performance is improved. Overall, we believe that hyperlink-based training benefits contextualized entity representations but does not benefit descriptive entity representations (see, for example, the drop of nearly 2 points on ESR, which is based solely on descriptive representations). This pattern may be due to the difficulty of using descriptive entity representations to reconstruct their appearing context.

## 6 Analysis

**Is descriptive entity representation necessary?** A natural question to ask is whether the entity description is needed, as for humans, the entity names carry sufficient amount of information for a lot of tasks. To answer this question, we experiment with encoding entity names by the descriptive entity encoder for ERT (entity relationship typing) and NED (named entity disambiguation) tasks. The results in Table 4 show that encoding the entity names by themselves already captures a great deal of knowledge regarding entities, especially for CoNLL-YAGO. However, in tasks like ERT, the entity descriptions are crucial as the names do not reveal enough information to categorize their relationships.

Table 5 reports the performance of different descriptive entity representations on the CoNLL-YAGO task. The three models all use ELMo as the context encoder. "ELMo" encodes the entity name with ELMo as descriptive encoder, while both Gupta et al. (2017) and Deep ED (Ganea and Hofmann, 2017) use their trained static entity embeddings. [3] As Gupta et al. (2017) and Deep ED have different embedding sizes from ELMo, we add an extra linear layer after them to map to the same dimension. These two models are designed for entity linking, which gives them potential advantages. Even so, ELMo outperforms them both by a wide margin.

**Per-Layer Analysis.** We evaluate each ELMo and EntELMo layer, i.e., the character CNN layer and two bidirectional LSTM layers, as well as each BERT layer on the EntEval tasks. Figure 6 reveals that for ELMo models, the first and second LSTM layers capture most of the entity knowledge from context and descriptions. The BERT layers show more diversity. Lower layers perform better on ESR (entity similarity and relatedness), while

[3] We note that the numbers reported here are not strictly comparable to the ones in their original paper since we keep all the top 30 candidates from Crosswiki while prior work employs different pruning heuristics.

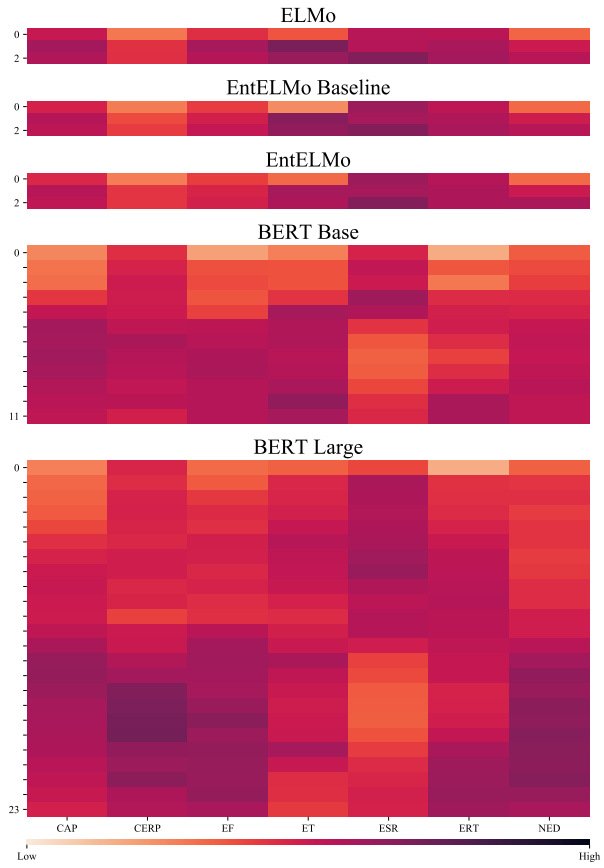


Figure 6: Heatmap showing per-layer performances for ELMo, EntELMo baseline, EntELMo, BERT Base, and BERT Large.

for other tasks higher layers are more effective.

## 7 Conclusion

Our proposed EntEval test suite provides a standardized evaluation method for entity representations. We demonstrate that EntEval tasks can benefit from the success of contextualized word representations such as ELMo and BERT. Augmenting encoding-decoding loss leveraging natural hyperlinks from Wikipedia further improves ELMo on some EntEval tasks. As shown by our experimental results, the contextualized entity encoder benefits more from this hyperlink-based training objective, suggesting future works to prioritize encoding entity description from its mention context.

## Acknowledgments

This research was supported in part by a Bloomberg data science research grant to K. Stratos and K. Gimpel.

## References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2017. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. In *ICLR*.

Gabor Angeli and Christopher D. Manning. 2014. NaturalLI: Natural logic inference for common sense reasoning. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 534–545, Doha, Qatar. Association for Computational Linguistics.

Yonatan Belinkov, Lluís Màrquez, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2017. Evaluating layers of representation in neural machine translation on part-of-speech and semantic tagging tasks. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. AcM.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Hong Chen, Zhenhua Fan, Hao Lu, Alan Yuille, and Shu Rong. 2018. PreCo: A large-scale dataset in preschool vocabulary for coreference resolution. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 172–181, Brussels, Belgium. Association for Computational Linguistics.

Mingda Chen, Zewei Chu, and Kevin Gimpel. 2019a. Evaluation benchmarks and learning criteriafor discourse-aware sentence representations. In *Proc. of EMNLP*.

Sihao Chen, Daniel Khashabi, Wenpeng Yin, Chris Callison-Burch, and Dan Roth. 2019b. Seeing things from a different angle:discovering diverse perspectives about claims. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 542–557, Minneapolis, Minnesota. Association for Computational Linguistics.

Eunsol Choi, Omer Levy, Yejin Choi, and Luke Zettlemoyer. 2018. Ultra-fine entity typing. In *Proceedings of the 56th Annual Meeting of the Association*

*for Computational Linguistics (Volume 1: Long Papers)*, pages 87–96, Melbourne, Australia. Association for Computational Linguistics.

Elizabeth Clark, Yangfeng Ji, and Noah A. Smith. 2018. Neural text generation in stories using entity representations as context. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2250–2260, New Orleans, Louisiana. Association for Computational Linguistics.

Alexis Conneau and Douwe Kiela. 2018. Senteval: An evaluation toolkit for universal sentence representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*.

Alexis Conneau, Germán Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics.

Luciano Del Corro, Abdalghani Abujabal, Rainer Gemulla, and Gerhard Weikum. 2015. FINET: Context-aware fine-grained named entity typing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 868–878, Lisbon, Portugal. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Greg Durrett and Dan Klein. 2013. Easy victories and uphill battles in coreference resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1971–1982, Seattle, Washington, USA. Association for Computational Linguistics.

Greg Durrett and Dan Klein. 2014. A joint model for entity analysis: Coreference, typing, and linking. *Transactions of the Association for Computational Linguistics*, 2:477–490.

Matthew Francis-Landau, Greg Durrett, and Dan Klein. 2016. Capturing semantic similarity for entity linking with convolutional neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1256–1261, San Diego, California. Association for Computational Linguistics.

Octavian-Eugen Ganea and Thomas Hofmann. 2017. Deep joint entity disambiguation with local neural attention. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2619–2629, Copenhagen, Denmark. Association for Computational Linguistics.

Zhaochen Guo and Denilson Barbosa. 2014. Robust entity linking via random walks. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '14, pages 499–508, New York, NY, USA. ACM.

Nitish Gupta, Sameer Singh, and Dan Roth. 2017. Entity linking via joint encoding of types, descriptions, and context. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2681–2690, Copenhagen, Denmark. Association for Computational Linguistics.

He He, Anusha Balakrishnan, Mihail Eric, and Percy Liang. 2017. Learning symmetric collaborative dialogue agents with dynamic knowledge graph embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1766–1776, Vancouver, Canada. Association for Computational Linguistics.

Zhengyan He, Shujie Liu, Mu Li, Ming Zhou, Longkai Zhang, and Houfeng Wang. 2013. Learning entity representation for entity disambiguation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 30–34, Sofia, Bulgaria. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Johannes Hoffart, Stephan Seufert, Dat Ba Nguyen, Martin Theobald, and Gerhard Weikum. 2012. Kore: keyphrase overlap relatedness for entity disambiguation. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 545–554. ACM.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 782–792. Association for Computational Linguistics.

Matthew Honnibal and Ines Montani. 2017. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. *To appear*.

Hongzhao Huang, Larry Heck, and Heng Ji. 2015. Leveraging deep neural networks and knowledge graphs for entity disambiguation. *arXiv preprint arXiv:1504.07678*.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, Beijing, China. Association for Computational Linguistics.

Yangfeng Ji, Chenhao Tan, Sebastian Martschat, Yejin Choi, and Noah A. Smith. 2017. Dynamic entity representations in neural language models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1830–1839, Copenhagen, Denmark. Association for Computational Linguistics.

Ben Kantor and Amir Globerson. 2019. Coreference resolution with entity equalization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 673–677, Florence, Italy. Association for Computational Linguistics.

Phong Le and Ivan Titov. 2018. Improving entity linking by modeling latent relations between mentions. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1595–1604, Melbourne, Australia. Association for Computational Linguistics.

Phong Le and Ivan Titov. 2019. Boosting entity linking performance by leveraging unlabeled documents. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1935–1945, Florence, Italy. Association for Computational Linguistics.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197, Copenhagen, Denmark. Association for Computational Linguistics.

Xiang Li, Aynaz Taheri, Lifu Tu, and Kevin Gimpel. 2016. Commonsense knowledge base completion. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1445–1455, Berlin, Germany. Association for Computational Linguistics.

Xiao Ling, Sameer Singh, and Daniel S. Weld. 2015. Design challenges for entity linking. *Transactions of the Association for Computational Linguistics*, 3:315–328.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019a. Linguistic knowledge and transferability of contextual representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Robert Logan, Nelson F. Liu, Matthew E. Peters, Matt Gardner, and Sameer Singh. 2019. Barack's wife hillary: Using knowledge graphs for fact-aware language modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5962–5971, Florence, Italy. Association for Computational Linguistics.

Lajanugen Logeswaran, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, Jacob Devlin, and Honglak Lee. 2019. Zero-shot entity linking by reading entity descriptions. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3449–3460, Florence, Italy. Association for Computational Linguistics.

Teng Long, Emmanuel Bengio, Ryan Lowe, Jackie Chi Kit Cheung, and Doina Precup. 2017. World knowledge for reading comprehension: Rare entity prediction with hierarchical lstms using external descriptions. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 825–834.

Pedro Henrique Martins, Zita Marinho, and André F. T. Martins. 2019. Joint learning of named entity recognition and entity linking. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 190–196, Florence, Italy. Association for Computational Linguistics.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6294–6305. Curran Associates, Inc.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium. Association for Computational Linguistics.

Shikhar Murty, Patrick Verga, Luke Vilnis, and Andrew McCallum. 2017. Finer grained entity typing with typenet. *arXiv preprint arXiv:1711.05795*.

Denis Newman-Griffis, Albert M. Lai, and Eric Fosler-Lussier. 2018. Jointly embedding entities and text with distant supervision. In *Proceedings of The Third Workshop on Representation Learning for NLP*, pages 195–206, Melbourne, Australia. Association for Computational Linguistics.

Rasha Obeidat, Xiaoli Fern, Hamed Shahbazi, and Prasad Tadepalli. 2019. Description-based zero-shot fine-grained entity typing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 807–814, Minneapolis, Minnesota. Association for Computational Linguistics.

Yasumasa Onoe and Greg Durrett. 2019. Learning to denoise distantly-labeled data for entity typing. In *NAACL-HLT*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018a. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. 2018b. Dissecting contextual word embeddings: Architecture and representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509, Brussels, Belgium. Association for Computational Linguistics.

Maxim Rabinovich and Dan Klein. 2017. Fine-grained entity typing with high-multiplicity assignments. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 330–334.

Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Explain yourself! leveraging language models for commonsense reasoning. *arXiv preprint arXiv:1906.02361*.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. 2019. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural MT learn source syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1526–1534, Austin, Texas. Association for Computational Linguistics.

Sameer Singh, Amarnag Subramanya, Fernando Pereira, and Andrew McCallum. 2012. Wikilinks: A large-scale cross-document coreference corpus labeled via links to Wikipedia. Technical Report UM-CS-2012-015.

Robert Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Valentin I. Spitkovsky and Angel X. Chang. 2012. A cross-lingual dictionary for English Wikipedia concepts. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*, pages 3168–3175, Istanbul, Turkey. European Language Resources Association (ELRA).

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. 2019. Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. Fever: a large-scale dataset for fact extraction and verification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819.

Trieu H Trinh and Quoc V Le. 2018. A simple method for commonsense reasoning. *arXiv preprint arXiv:1806.02847*.

Andreas Vlachos and Sebastian Riedel. 2014. Fact checking: Task definition and dataset construction. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*, pages 18–22, Baltimore, MD, USA. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

William Yang Wang. 2017. "liar, liar pants on fire": A new benchmark dataset for fake news detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 422–426, Vancouver, Canada. Association for Computational Linguistics.

Kellie Webster, Marta Recasens, Vera Axelrod, and Jason Baldridge. 2018. Mind the GAP: A balanced corpus of gendered ambiguous pronouns. *Transactions of the Association for Computational Linguistics*, 6:605–617.

Sam Wiseman, Alexander M. Rush, and Stuart M. Shieber. 2016. Learning global features for coreference resolution. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 994–1004, San Diego, California. Association for Computational Linguistics.

Yadollah Yaghoobzadeh and Hinrich Schütze. 2015. Corpus-level fine-grained entity typing using contextual information. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 715–725, Lisbon, Portugal. Association for Computational Linguistics.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint learning of the embedding of words and entities for named entity disambiguation. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 250–259, Berlin, Germany. Association for Computational Linguistics.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2017. Learning distributed representations of texts and entities from knowledge base. *Transactions of the Association for Computational Linguistics*, 5(1):397–411.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Wenpeng Yin and Dan Roth. 2018. TwoWingOS: A two-wing optimization strategy for evidential claim verification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 105–114, Brussels, Belgium. Association for Computational Linguistics.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. Swag: A large-scale adversarial dataset for grounded commonsense inference. *arXiv preprint arXiv:1808.05326*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451, Florence, Italy. Association for Computational Linguistics.

# Joint Event and Temporal Relation Extraction with Shared Representations and Structured Prediction

**Rujun Han,**[1,2] **Qiang Ning,**[3] **Nanyun Peng**[1,2]
[1]Department of Computer Science, University of Southern California
[2]Information Sciences Institute, University of Southern California
[3]Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign
rujunhan@isi.edu, qning2@illinois.edu, npeng@isi.edu

## Abstract

We propose a joint event and temporal relation extraction model with shared representation learning and structured prediction. The proposed method has two advantages over existing work. First, it improves event representation by allowing the event and relation modules to share the same contextualized embeddings and neural representation learner. Second, it avoids error propagation in the conventional pipeline systems by leveraging structured inference and learning methods to assign both the event labels and the temporal relation labels jointly. Experiments show that the proposed method can improve both event extraction and temporal relation extraction over state-of-the-art systems, with the end-to-end $F_1$ improved by 10% and 6.8% on two benchmark datasets respectively.

## 1 Introduction

The extraction of temporal relations among events is an important natural language understanding (NLU) task that can benefit many downstream tasks such as question answering, information retrieval, and narrative generation. The task can be modeled as building a graph for a given text, whose nodes represent events and edges are labeled with temporal relations correspondingly. Figure 1a illustrates such a graph for the text shown therein. The nodes **assassination**, **slaughtered**, **rampage**, **war**, and **Hutu** are the candidate events, and different types of edges specify different temporal relations between them: **assassination** is *BEFORE* **rampage**, **rampage** *INCLUDES* **slaughtered**, and the relation between **slaughtered** and **war** is *VAGUE*. Since "Hutu" is actually not an event, a system is expected to annotate the relations between "Hutu" and all other nodes in the graph as *NONE* (i.e., no relation).

As far as we know, all existing systems treat this task as a pipeline of two separate subtasks,

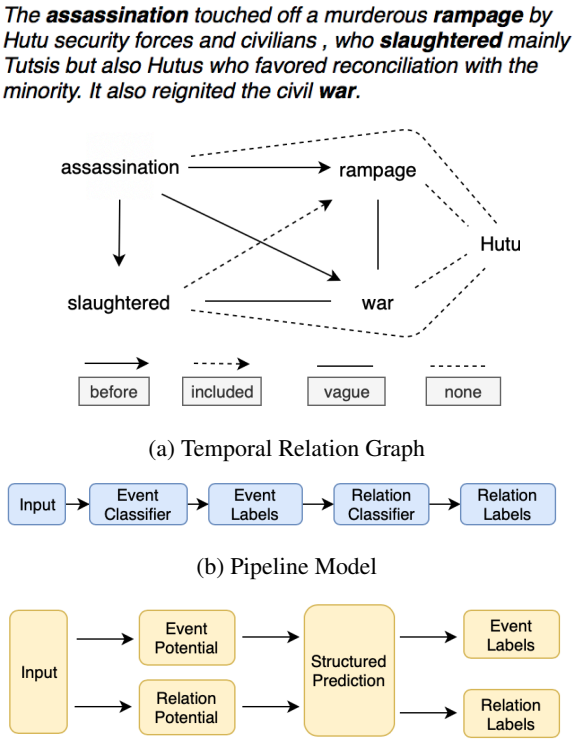


(a) Temporal Relation Graph

(b) Pipeline Model

(c) Structured Joint Model

Figure 1: An illustration of event and relation models in our proposed joint framework. (a) is a (partial) graph of the output of the relation extraction model. "Hutu" is not an event and hence all relations including it should be annotated as *NONE*. (b) and (c) are comparisons between a pipeline model and our joint model.

i.e., event extraction and temporal relation classification, and they also assume that gold events are given when training the relation classifier (Verhagen et al., 2007, 2010; UzZaman et al., 2013; Chambers et al., 2014; Ning et al., 2017; Meng and Rumshisky, 2018). Specifically, they built end-to-end systems that extract events first and then predict temporal relations between them (Fig. 1b). In these pipeline models, event extraction errors will propagate to the relation classification step and cannot be corrected afterwards. **Our first contribution** is the proposal of a joint model that ex-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 434–444, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

tracts both events and temporal relations simultaneously (see Fig. 1c). The motivation is that if we train the relation classifier with *NONE* relations between non-events, then it will potentially have the capability of correcting event extraction mistakes. For instance in Fig. 1a, if the relation classifier predicts *NONE* for (**Hutu**, **war**) with a high confidence, then this is a strong signal that can be used by the event classifier to infer that at least one of them is not an event.

**Our second contribution** is that we improve event representations by sharing the same contextualized embeddings and neural representation learner between the event extraction and temporal relation extraction modules for the first time. On top of the shared embeddings and neural representation learner, the proposed model produces a graph-structured output representing all the events and relations in the given sentences.

A valid graph prediction in this context should satisfy two structural constraints. First, the temporal relation should always be *NONE* between two non-events or between one event and one non-event. Second, for those temporal relations among events, no loops should exist due to the transitive property of time (e.g., if **A** is before **B** and **B** is before **C**, then **A** must be before **C**). The validity of a graph is guaranteed by solving an integer linear programming (ILP) optimization problem with those structural constraints, and our joint model is trained by structural support vector machines (SSVM) in an end-to-end fashion.

Results show that, according to the end-to-end $F_1$ score for temporal relation extraction, the proposed method improves CAEVO (Chambers et al., 2014) by 10% on TB-Dense, and improves CogCompTime (Ning et al., 2018c) by 6.8% on MATRES. We further show ablation studies to confirm that the proposed joint model with shared representations and structured learning is very effective for this task.

## 2 Related Work

In this section we briefly summarize the existing work on event extraction and temporal relation extraction. To the best of our knowledge, there is no prior work on joint *event* and relation extraction, so we will review joint *entity* and relation extraction works instead.

Existing **event extraction** methods in the temporal relation domain, as in the TempEval3 workshop (UzZaman et al., 2013), all use conventional machine learning models (logistic regression, SVM, or Max-entropy) with hand-engineered features (e.g., ClearTK (Bethard, 2013) and NavyTime (Chambers, 2013)). While other domains have shown progress on event extraction using neural methods (Nguyen and Grishman, 2015; Nguyen et al., 2016; Feng et al., 2016), recent progress in the temporal relation domain is focused more on the setting where gold events are provided. Therefore, we first show the performance of a neural event extractor on this task, although it is not our main contribution.

Early attempts on **temporal relation extraction** use local pair-wise classification with hand-engineered features (Mani et al., 2006; Verhagen et al., 2007; Chambers et al., 2007; Verhagen and Pustejovsky, 2008). Later efforts, such as ClearTK (Bethard, 2013), UTTime (Laokulrat et al., 2013), NavyTime (Chambers, 2013), and CAEVO (Chambers et al., 2014) improve earlier work with better linguistic and syntactic rules. Yoshikawa et al. (2009); Ning et al. (2017); Leeuwenberg and Moens (2017) explore structured learning for this task, and more recently, neural methods have also been shown effective (Tourille et al., 2017; Cheng and Miyao, 2017; Meng et al., 2017; Meng and Rumshisky, 2018).

In practice, we need to extract both events and those temporal relations among them from raw text. All the works above treat this as two subtasks that are solved in a pipeline. To the best of our knowledge, there has been no existing work on **joint event-temporal relation extraction**. However, the idea of "joint" has been studied for *entity*-relation extraction in many works. Miwa and Sasaki (2014) frame their joint model as table filling tasks, map tabular representation into sequential predictions with heuristic rules, and construct global loss to compute the best joint predictions. Li and Ji (2014) define a global structure for joint entity and relation extraction, encode local and global features based on domain and linguistic knowledge. and leverage beam-search to find global optimal assignments for entities and relations. Miwa and Bansal (2016) leverage LSTM architectures to jointly predict both entity and relations, but fall short on ensuring prediction consistency. Zhang et al. (2017) combine the benefits of both neural net and global optimization with beam

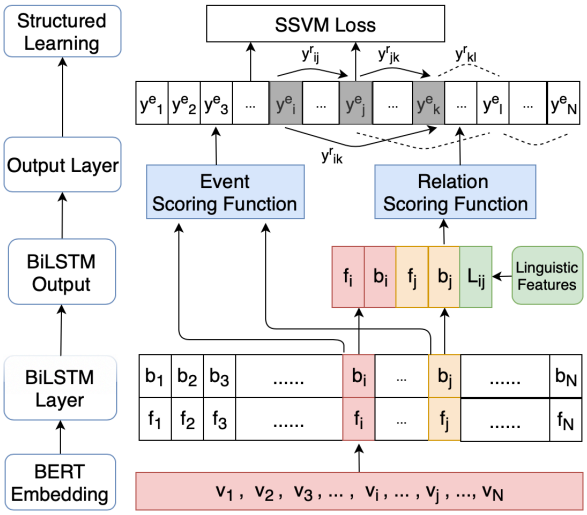


Figure 2: Deep neural network architecture for joint structured learning. Note that on the structured learning layer, grey bars denote tokens being predicted as events. Edge types between events follow the same notations as in 1a. $y_l^e = 0$ (non-event), so all edges connecting to $y_l^e$ are *NONE*. $y_i^e = 1, y_j^e = 1, y_k^e = 1$ (events) and hence edges between them are forced to be the same ($y_{ij}^r = y_{jk}^r = y_{ik}^r =$ *BEFORE* in this example) by transitivity. These global assignments are input to compute the SSVM loss.

search. Motivated by these works, we propose an end-to-end trainable neural structured support vector machine (neural SSVM) model to *simultaneously* extract events and their relations from text and ensure the global structure via ILP constraints. Next, we will describe in detail our proposed method.

## 3 Joint Event-Relation Extraction Model

In this section we first provide an overview of our neural SSVM model, and then describe each component in our framework in detail (i.e., the multi-tasking neural scoring module, and how inference and learning are performed). We denote the set of all possible relation labels (including *NONE*) as $\mathcal{R}$, all event candidates (both events and non-events) as $\mathcal{E}$, and all relation candidates as $\mathcal{EE}$.

### 3.1 Neural SSVM

Our neural SSVM adapts the SSVM loss as:

$$\mathcal{L} = \sum_{n=1}^{l} \frac{C}{M^n} \big[ \max_{\hat{\boldsymbol{y}}^n \in \mathcal{Y}} \big(0, \Delta(\boldsymbol{y}^n, \hat{\boldsymbol{y}}^n) + \bar{S}_{\mathcal{R}}^n + C_{\mathcal{E}} \bar{S}_{\mathcal{E}}^n\big)\big] + ||\Phi||^2, \quad (1)$$

where $\bar{S}_{\mathcal{E}}^n = S(\hat{\boldsymbol{y}}_{\mathcal{E}}^n; \boldsymbol{x}^n) - S(\boldsymbol{y}_{\mathcal{E}}^n; \boldsymbol{x}^n)$ and $\bar{S}_{\mathcal{R}}^n = S(\hat{\boldsymbol{y}}_{\mathcal{R}}^n; \boldsymbol{x}^n) - S(\boldsymbol{y}_{\mathcal{R}}^n; \boldsymbol{x}^n)$; $\Phi$ denotes model parameters, $n$ indexes instances, $M^n = |\mathcal{E}|^n + |\mathcal{EE}|^n$ denotes the total number of relations $|\mathcal{E}|^n$ and events $|\mathcal{EE}|^n$ in instance $n$. $\boldsymbol{y}^n, \hat{\boldsymbol{y}}^n$ denote the gold and predicted global assignments of events and relations for instance $n$—each of which consists of either one hot vector representing true and predicted relation labels $\boldsymbol{y}_{\mathcal{R}}^n, \hat{\boldsymbol{y}}_{\mathcal{R}}^n \in \{0,1\}^{|\mathcal{EE}|}$, or entity labels $\boldsymbol{y}_{\mathcal{E}}^n, \hat{\boldsymbol{y}}_{\mathcal{E}}^n \in \{0,1\}^{|\mathcal{E}|}$. A maximum a posteriori probability (MAP) inference is needed to find $\hat{\boldsymbol{y}}^n$, which we formulate as an interger linear programming (ILP) problem and describe more details in Section 3.3. $\Delta(\boldsymbol{y^n}, \hat{\boldsymbol{y}}^n)$ is a distance measurement between the gold and the predicted assignments; we simply use the Hamming distance. $C$ and $C_{\mathcal{E}}$ are the hyper-parameters to balance the losses between event, relation and the regularizer, and $S(\boldsymbol{y}_{\mathcal{E}}^n; \boldsymbol{x}^n), S(\boldsymbol{y}_{\mathcal{R}}^n; \boldsymbol{x}^n)$ are scoring functions, which we design a multi-tasking neural architecture to learn.

The intuition behind the SSVM loss is that it requires the score of gold output structure $\boldsymbol{y}^n$ to be greater than the score of the best output structure under the current model $\hat{\boldsymbol{y}}^n$ with a margin $\Delta(\boldsymbol{y}^n, \hat{\boldsymbol{y}}^n)$[1] or else there will be some loss. The training objective is to minimize the loss.

The major difference between our neural-SSVM and the traditional SSVM model is the scoring function. Traditional SSVM uses a linear function over hand-crafted features to compute the scores, whereas we propose to use a recurrent neural network to estimate the scoring function and train the entire architecture end-to-end.

### 3.2 Multi-Tasking Neural Scoring Function

The recurrent neural network (RNN) architecture has been widely adopted by prior temporal extraction work to encode context information (Tourille et al., 2017; Cheng and Miyao, 2017; Meng et al., 2017). Motivated by these works, we adopt a RNN-based scoring function for both event and relation prediction in order to learn features in a data driven way and capture long-term contexts in the input. In Fig. 2, we skip the input layer for simplicity.[2]

The bottom layer corresponds to contextualized word representations denoted as $v_k$. We use $(i, j) \in \mathcal{EE}$ to denote a candidate relation and $i \in \mathcal{E}$ to

[1] Note that if the best prediction is the same as the gold structure, the margin is zero; there will be no loss.

[2] Following the convention of event relation prediction literature (Chambers et al., 2014; Ning et al., 2018a,b), we only consider event pairs that occur in the same or neighboring sentences, but the architecture can be easily adapted to the case where inputs are longer than two sentences.

indicate a candidate event in the input sentences of length N. We fix word embeddings computed by a pre-trained BERT-base model (Devlin et al., 2018). They are then fed into a BiLSTM layer to further encode task-specific contextual information. Both event and relation tasks share this layer.

The event scorer is illustrated by the left two branches following the BiLSTM layer. We simply concatenate both forward and backward hidden vectors to encode the context of each token. As for the relation scorer shown in the right branches, for each pair $(i, j)$ we take the forward and backward hidden vectors corresponding to them, $f_i, b_i, f_j, b_j$, and concatenate them with linguistic features as in previous event relation prediction research. We denote linguistic features as $L_{i,j}$ and only use simple features provided in the original datasets: token distance, tense, and polarity of events.

Finally, all hidden vectors and linguistic features are concatenated to form the input to compute the probability of being an event or a softmax distribution over all possible relation labels—which we refer to as the RNN-based **scoring function** in the following sections.

### 3.3 MAP Inference

A MAP inference is needed both during training to obtain $\hat{\boldsymbol{y}}^n$ in the loss function (Equation 1), as well as during the test time to get globally coherent assignments. We formulate the inference problem as an ILP problem. The inference framework is established by constructing a global objective function using scores from local scorers and imposing several global constraints: 1) one-label assignment, 2) event-relation consistency, and 3) symmetry and transitivity as in Bramsen et al. (2006); Chambers and Jurafsky (2008); Denis and Muller (2011); Do et al. (2012); Ning et al. (2017).

#### 3.3.1 Objective Function

The objective function of the global inference is to find the global assignment that has the highest probability under the current model, as specified in Equation 2:

$$\hat{y} = \arg\max \sum_{(i,j)\in\mathcal{EE}} \sum_{r\in\mathcal{R}} y^r_{i,j} S(y^r_{i,j}, \boldsymbol{x}) + C_{\mathcal{E}} \sum_{k\in\mathcal{E}} \sum_{e\in\{0,1\}} y^e_k S(y^e_k, \boldsymbol{x}) \quad (2)$$

$$\textbf{s.t.}\ y^r_{i,j}, y^e_k \in \{0,1\}\,,\ \sum_{r\in\mathcal{R}} y^r_{i,j} = 1,\ \sum_{e\in\{0,1\}} y^e_k = 1,$$

where $y^e_k$ is a binary indicator of whether the $k$-th candidate is an event or not, and $y^r_{i,j}$ is a binary indicator specifying whether the global prediction of the relation between $(i, j)$ is $r \in \mathcal{R}$. $S(y^e_k, \boldsymbol{x}), \forall e \in \{0, 1\}$ and $S(y^r_{i,j}, \boldsymbol{x}), \forall r \in \mathcal{R}$ are the scoring functions obtained from the event and relation scoring functions, respectively. The output of the global inference $\hat{\mathbf{y}}$ is a collection of optimal label assignments for all events and relation candidates in a fixed context. $C_{\mathcal{E}}$ is a hyper-parameter controlling weights between relation and event. The constraint that follows immediately from the objective function is that the global inference should only assign one label for all entities and relations.

#### 3.3.2 Constraints

We introduce several additional constraints to ensure the resulting optimal output graph forms a valid and plausible event graph.

**Event-Relation Consistency.** Event and relation prediction consistency is defined with the following property: **a pair of input tokens have a positive temporal relation if and only if both tokens are events**. The following global constraints will satisfy this property,

$$\forall(i,j) \in \mathcal{EE}, e^P_i \geq r^P_{i,j}\,,\ e^P_j \geq r^P_{i,j}$$

$$\text{and } e^N_i + e^N_j \geq r^N_{i,j}$$

where $e^P_i$ denotes an event and $e^N_i$ denotes a non-event token. $r^P_{i,j}$ indicates positive relations: *BEFORE*, *AFTER*, *SIMULTANEOUS*, *INCLUDES*, *IS_INCLUDED*, *VAGUE* and $r^N_{i,j}$ indicate a negative relation, i.e., *NONE*. A formal proof of this property can be found in Appendix A.

**Symmetry and Transitivity Constraint.** We also explore the symmetry and transitivity constraints of relations. They are specified as follows:

$$\forall(i,j),(j,k) \in \mathcal{EE}, y^r_{i,j} = y^{\bar{r}}_{j,i},\ \text{(symmetry)}$$

$$y^{r_1}_{i,j} + y^{r_2}_{j,k} - \sum_{r_3\in Trans(r_1,r_2)} y^{r_3}_{i,k} \leq 1,\ \text{(transitivity)}$$

Intuitively, the symmetry constraint forces two pairs of events with flipping orders to have reversed relations. For example, if $r_{i,j}$ = *BEFORE*, then $r_{j,i}$ = *AFTER*. The transitivity constraint rules that if $(i, j)$, $(j, k)$ and $(i, k)$ pairs exist in the graph, the label (relation) prediction of $(i, k)$ pair

has to fall into the transitivity set specifyed by $(i, j)$ and $(j, k)$ pairs. The full transitivity table can be found in Ning et al. (2018a).

### 3.4 Learning

We begin by experimenting with optimizing SSVM loss directly, but model performance degrades.[3] Therefore, we develop a two-state learning approach which first trains a pipeline version of the joint model without feedback from global constraints. In other words, the local neural scoring functions are optimized with cross-entropy loss using gold events and relation candidates that are constructed directly from the outputs of the event model. During the second stage, we switch to the global SSVM loss function in Equation 1 and re-optimize the network to adjust for global properties. We will provide more details in Section 4.

## 4 Implementation Details

In this section we describe implementation details of the baselines and our four models to build an end-to-end event temporal relation extraction system with an emphasis on the structured joint model. In Section 6 we will compare and contrast them and show why our proposed structured joint model works the best.

### 4.1 Baselines

We run two event and relation extraction systems, CAEVO[4] (Chambers et al., 2014) and CogCompTime[5] (Ning et al., 2018c), on TB-Dense and MATRES, respectively. These two methods both leverage conventional learning algorithms (i.e., MaxEnt and averaged perceptron, respectively) based on manually designed features to obtain separate models for events and temporal relations, and conduct end-to-end relation extraction as a pipeline. Note Chambers et al. (2014) does not report event and end-to-end temporal relation extraction performances, so we calculate the scores per our implementation.

[3] We leave further investigation for future work.

[4] https://www.usna.edu/Users/cs/nchamber/caevo/

[5] http://cogcomp.org/page/publication_view/844

### 4.2 End-to-End Event Temporal Relation Extraction

**Single-Task Model.** The most basic way to build an end-to-end system is to train separate event detection and relation prediction models with gold labels, as we mentioned in our introduction. In other words, the BiLSTM layer is not shared as in Fig. 2. During evaluation and test time, we use the outputs from the event detection model to construct relation candidates and apply the relation prediction model to make the final prediction.

**Multi-Task Model.** This is the same as the single-task model except that the BiLSTM layer is now shared for both event and relation tasks. Note that both single-task and multi-task models are not trained to tackle the *NONE* relation directly. They both rely on the predictions of the event model to annotate relations as either positive pairs or *NONE*.

**Pipeline Joint Model.** This shares the same architecture as the multi-task model, except that during training, we use the predictions of the event model to construct relation candidates to train the relation model. This strategy will generate *NONE* pairs during training if one argument of the relation candidate is not an event. These *NONE* pairs will help the relation model to distinguish negative relations from positive ones, and thus become more robust to event prediction errors. We train this model with gold events and relation candidates during the first several epochs in order to obtain a relatively accurate event model and switch to a pipeline version afterwards inspired by Miwa and Bansal (2016).

**Structured Joint Model.** This is described in detail in Section 3. However, we experience difficulties in training the model with SSVM loss from scratch. This is due to large amounts of non-event tokens, and the model is not capable of distinguishing them in the beginning. We thus adopt a two-stage learning procedure where we take the best pipeline joint model and re-optimize it with the SSVM loss.

To restrict the search space for events in the ILP inference of the SSVM loss, we use the predicted probabilities from the event detection model to filter out non-events since the event model has a strong performance, as shown in Section 6. Note that this is very different from the pipeline

model where events are first predicted and relations are constructed with *predicted* events. Here, we only leverage an additional hyper-parameter $T_{evt}$ to filter out highly unlikely event candidates. Both event and relation labels are assigned *simultaneously* during the global inference with ILP, as specified in Section 3.3. We also filter out tokens with POS tags that do not appear in the training set as most of the events are either nouns or verbs in TB-Dense, and all events are verbs in MATRES.

**Hyper-Parameters.** All single-task, multi-task and pipeline joint models are trained by minimizing cross-entropy loss. We observe that model performances vary significantly with dropout ratio, hidden layer dimensions of the BiLSTM model and entity weight in the loss function (with relation weight fixed at 1.0). We leverage a pre-trained BERT model to compute word embedding[6] and all MLP scoring functions have one hidden layer.[7] In the SSVM loss function, we fix the value of $C = 1$, but fine-tune $C_{\mathcal{E}}$ in the objective function in Equation 2. Hyper-parameters are chosen using a standard development set for TB-Dense and a random holdout-set based on an 80/20 split of training data for MATRES. To solve ILP in the inference process, we leverage an off-the-shelf solver provided by Gurobi optimizer; i.e. the best solutions from the Gurobi optimizer are inputs to the global training. The best combination of hyper-parameters can be found in Table 9 in our appendix.[8]

## 5 Experimental Setup

In this section we first provide a brief overview of temporal relation data and describe the specific datasets used in this paper. We also explain the evaluation metrics at the end.

### 5.1 Temporal Relation Data

Temporal relation corpora such as TimeBank (Pustejovsky et al., 2003) and RED (O'Gorman et al., 2016) facilitate the research in temporal relation extraction. The common issue in these corpora is missing annotations. Collecting densely annotated temporal relation corpora with all events and relations fully annotated is reported to be a challenging task as annotators could easily overlook some facts (Bethard et al., 2007; Cassidy et al., 2014; Chambers et al., 2014; Ning et al., 2017), which made both modeling and evaluation extremely difficult in previous event temporal relation research.

| | TB-Dense | MATRES |
|---|---|---|
| **# of Documents** | | |
| Train | 22 | 183 |
| Dev | 5 | - |
| Test | 9 | 20 |
| **# of Pairs** | | |
| Train | 4032 | 6332 |
| Dev | 629 | - |
| Test | 1427 | 827 |

Table 1: Data overview. Note that the numbers reported for MATRES do not include the AQUAINT section.

The TB-Dense dataset mitigates this issue by forcing annotators to examine all pairs of events within the same or neighboring sentences, and it has been widely evaluated on this task (Chambers et al., 2014; Ning et al., 2017; Cheng and Miyao, 2017; Meng and Rumshisky, 2018). Recent data construction efforts such as MATRES (Ning et al., 2018a) further enhance the data quality by using a multi-axis annotation scheme and adopting a start-point of events to improve inter-annotator agreements. We use TB-Dense and MATRES in our experiments and briefly summarize the data statistics in Table 1.

### 5.2 Evaluation Metrics

To be consistent with previous research, we adopt two different evaluation metrics. The first one is the standard **micro-average scores**. For densely annotated data, the micro-average metric should share the same precision, recall and F1 scores. However, since our joint model includes *NONE* pairs, we follow the convention of IE tasks and exclude them from evaluation. The second one is similar except that we exclude both *NONE* and *VAGUE* pairs following (Ning et al., 2018c). Please refer to Figure 4 in the appendix for a visualizations of the two metrics.

## 6 Results and Analysis

The main results of this paper can be found in Table 2. All best-recall and F1 scores are achieved by our structured joint model, and the results outperform the baseline systems by 10.0% and 6.8% on end-to-end relation extraction per F1 scores and

[6] We use a pre-trained BERT-Base model with 768 hidden size, 12 layers, 12 heads implemented by https://github.com/huggingface/pytorch-pretrained-BERT

[7] Let $H, K$ denotes the dimension of (concatenated) vector from BiLSTM and number of output classes. MLP layer consists of $|H| * |K| + |K| * |K|$ parameters

[8] PyTorch code will be made available upon acceptance.

| Corpus | Models | Event | | | Relation | | |
|---|---|---|---|---|---|---|---|
| | | P | R | F1 | P | R | F1 |
| TB-Dense | Structrued Joint Model (Ours) | 89.2 | **92.6** | **90.9** | **52.6** | **46.5** | **49.4** |
| | Chambers et al. (2014) | **97.2** | 79.4 | 87.4 | 43.8 | 35.7 | 39.4 |
| MATRES | Structrued Joint Model (Ours) | **87.1** | **88.5** | **87.8** | **59.0** | **60.2** | **59.6** |
| | Ning et al. (2018c) | 83.5 | 87.0 | 85.2 | 48.4 | 58.0 | 52.8 |

Table 2: Event and Relation Extraction Results on TB-Dense and MATRES

| Micro-average F1 (%) | TB-Dense | | | MATRES | | |
|---|---|---|---|---|---|---|
| | Event | Relation (G) | Relation (E) | Event | Relation (G) | Relation (E) |
| Baselines | 87.4† | 57.0‡ | 39.4† | 85.2 | 65.9 | 52.8 |
| Single-task | 88.6 | 61.9 | 44.3 | 86.9 | 75.3 | 57.2 |
| Multi-task | 89.2 | **64.5** | 48.4 | 86.4 | **75.5** | 58.7 |
| Pipeline Joint | 90.3 | - | 48.5 | 87.2 | - | 58.5 |
| Structured Joint | **90.9** | - | **49.4** | **87.8** | - | **59.6** |

Table 3: Further ablation studies on event and relation extractions. Relation (G) denotes train and evaluate using gold events to compose relation candidates, whereas Relation (E) means end-to-end relation extraction. † is the event extraction and pipeline relation extraction F1 scores for CAEVO (Chambers et al., 2014). 57.0‡ is the best previously reported micro-average score for temporal relation extraction based on gold events by Meng and Rumshisky (2018). All MATRES baseline results are provided by Ning et al. (2018c).

3.5% and 2.6% on event extraction per F1 scores. The best precision score for the TB-Dense dataset is achieved by CAEVO, which indicates that the linguistic rule-based system can make highly precise predictions by being conservative.

Table 3 shows a more detailed analysis, in which we can see that our single-task models with BERT embeddings and a BiLSTM encoder already outperform the baseline systems on end-to-end relation extraction tasks by 4.9% and 4.4% respectively. In the following sections we discuss step-by-step improvement by adopting multi-task, pipeline joint, and structured joint models on end-to-end relation extraction, event extraction, and relation extraction on gold event pairs.

### 6.1 End-to-End Relation Extraction

**TB-Dense.** The improvements over the single-task model per F1 score are 4.1% and 4.2% for the multi-task and pipeline joint model respectively. This indicates that the pipeline joint model is helpful only marginally. Table 4 shows that the structured joint model improves both precision and recall scores for *BEFORE* and *AFTER* and achieves the best end-to-end relation extraction performance at 49.4%—which outperforms the baseline system by 10.0% and the single-task model by 5.1%.

**MATRES.** Compared to the single-task model, the multi-task model improves F1 scores by 1.5%, while the pipeline joint model improves F1 scores by 1.3%—which means that pipeline joint training does not bring any gains for MATRES. The structured joint model reaches the best end-to-end F1 score at 59.6%, which outperforms the baseline system by 6.8% and the single-task model by 2.4%. We speculate that the gains come from the joint model's ability to help deal with *NONE* pairs, since recall scores for *BEFORE* and *AFTER* increase by 1.5% and 1.1% respectively (Table 10 in our appendix).

### 6.2 Event Extraction

**TB-Dense.** Our structured joint model outperforms the CAEVO baseline by 3.5% and the single-task model by 1.3%. Improvements on event extraction can be difficult because our single-task model already works quite well with a close-to 89% F1 score, while the inter-annotator agreement for events in TimeBank documents is merely 87% (UzZaman et al., 2013).

**MATRES.** The structured model outperforms the the baseline model and the single-task model by 2.6% and 0.9% respectively. However, we observe that the multi-task model has a slight drop in event extraction performance over the single-task model (86.4% vs. 86.9%). This indicates

| | CAEVO | | | Pipeline Joint | | | Structure Joint | | |
|---|---|---|---|---|---|---|---|---|---|
| | P | R | F1 | P | R | F1 | P | R | F1 |
| **B** | 41.4 | 19.5 | 26.5 | 59.0 | 46.9 | 52.3 | 59.8 | 46.9 | 52.6 |
| **A** | 42.1 | 17.5 | 24.7 | 69.3 | 45.3 | 54.8 | 71.9 | 46.7 | 56.6 |
| **I** | 50.0 | 3.6 | 6.7 | - | - | - | - | - | - |
| **II** | 38.5 | 9.4 | 15.2 | - | - | - | - | - | - |
| **S** | 14.3 | 4.5 | 6.9 | - | - | - | - | - | - |
| **V** | 44.9 | 59.4 | 51.1 | 45.1 | 55.0 | 49.5 | 45.9 | 55.8 | 50.4 |
| **Avg** | 43.8 | 35.7 | 39.4 | 51.5 | 45.9 | 48.5 | 52.6 | 46.5 | **49.4** |

Table 4: Model performance breakdown for TB-Dense. "-" indicates no predictions were made for that particular label, probably due to the small size of the training sample. *BEFORE* (**B**), *AFTER* (**A**), *INCLUDES* (**I**), *IS_INCLUDED* (**II**), *SIMULTANEOUS* (**S**), *VAGUE* (**V**)

that incorporating relation signals are not particularly helpful for event extraction on MATRES. We speculate that one of the reasons could be the unique event characteristics in MATERS. As we described in Section 5.1, all events in MATRES are verbs. It is possible that a more concentrated single-task model works better when events are homogeneous, whereas a multi-task model is more powerful when we have a mixture of event types, e.g., both verbs and nouns as in TB-Dense.

### 6.3 Relation Extraction with Gold Events

**TB-Dense.** There is much prior work on relation extraction based on gold events in TB-Dense. Meng and Rumshisky (2018) proposed a neural model with global information that achieved the best results as far as we know. The improvement of our single-task model over that baseline is mostly attributable to the adoption of BERT embedding. We show that sharing the LSTM layer for both events and relations can help further improve performance of the relation classification task by 2.6%. For the joint models, since we do not train them on gold events, the evaluation would be meaningless. We simply skip this evaluation.

**MATRES.** Both single-task and multi-task models outperform the baseline by nearly 10%, while the improvement of multi-task over single task is marginal. In MATRES, a relation pair is equivalent to a verb pair, and thus the event prediction task probably does not provide much more information for relation extraction.

In Table 4 we further show the breakdown performances for each positive relation on TB-Dense. The breakdown on MATRES is shown in Table 10 in the appendix. *BEFORE*, *AFTER* and *VAGUE* are the three dominant label classes in TB-Dense. We observe that the linguistic rule-based model, CAEVO, tends to have a more evenly spread-out

| Labels | TB-Dense | MATRES |
|---|---|---|
| *BEFORE* | 384 | 417 |
| *AFTER* | 274 | 266 |
| *INCLUDES* | 56 | – |
| *IS_INCLUDED* | 53 | – |
| *SIMULTANEOUS* | 22 | 31 |
| *VAGUE* | 638 | 113 |

Table 5: Label Size Breakdown in the Test Data

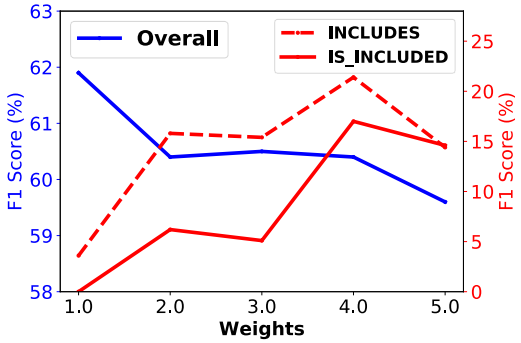


Figure 3: Performances from a single-task relation model under different class weights. Left-axis: overall model; right-axis: two minority relations.

performance, whereas our neural network-based models are more likely to have concentrated predictions due to the imbalance of the training sample across different label classes.

### 6.4 Discussion

**Label Imbalance.** One way to mitigate the label imbalance issue is to increase the sample weights for small classes during model training. We investigate the impact of class weights by refitting our single-task model with larger weights on *INCLUDES*, *IS_INCLUDED* and *VAGUE* in the cross-entropy loss.

Figure 3 shows that increasing class weights up to 4 times can significantly improve the F1 scores of *INCLUDES* and *IS_INCLUDED* classes with a decrease less than 2% for the overall F1 score. Performance of *INCLUDES* and *IS_INCLUDED* eventually degrades when class weights are too large. These results seem to suggest that more labels are needed in order to improve the performance on both of these two classes and the overall model. For *SIMULTANEOUS*, our model does not make any correct predictions for both TB-DENSE and MATRES by increasing class weight up to 10 times, which implies that *SIMULTANEOUS* could be a hard temporal relation to predict in general.

**Global Constraints.** In Table 6 we conduct an ablation study to understand the contributions from the event-relation prediction consis-

| Micro-average | TB-Dense | MATRES |
|---|---|---|
| No Structure | 48.5 | 58.5 |
| + $\mathcal{ER}$ Consistency | 49.4 | 59.5 |
| + Transitivity | 49.4 | 59.6 |

Table 6: Ablation Study on Global Constraints

tency constraint and the temporal relation transitivity constraint for the structured joint model. As we can see, the event-relation consistency helps improve the F1 scores by 0.9% and 1% for TB-Dense and MATRES, respectively, but the gain by using transitivity is either non-existing or marginal. We hypothesize two potential reasons: 1) We leveraged BERT contextualized embedding as word representation, which could tackle transitivity in the input context; 2) *NONE* pairs could make transitivity rule less useful, as positive pairs can be predicted as *NONE* and transitivity rule does not apply to *NONE* pairs.

**Error Analysis.** By comparing gold and predicted labels for events and temporal relations and examining predicted probabilities for events, we identified three major sources of mistakes made by our structured model, as illustrated in Table 7 with examples.

**Type 1.** Both events in Ex 1 are assigned low scores by the event module ($<< 0.01$). Although the structured joint model is designed to predict events and relations jointly, we leverage the event module to filter out tokens with scores lower than a threshold. Consequently, some true events can be mistakenly predicted as non-events, and the relation pairs including them are automatically assigned *NONE*.

**Type 2.** In Ex 2 the event module assigns high scores to tokens **happened** (0.97) and **according** (0.89), but **according** is not an event. When the structured model makes inference jointly, the decision will weigh heavily towards assigning **1** (event) to both tokens. With the event-relation consistency constraint, this pair is highly likely to be predicted as having a positive temporal relation. Nearly all mistakes made in this category follow the same pattern illustrated by this example.

**Type 3.** The existence of *VAGUE* makes temporal relation prediction challenging as it can be easily confused with other temporal relations, as shown in Ex 3. This challenge is compounded with *NONE* in our end-to-end extraction task.

Type 1 and Type 2 errors suggest that building a stronger event detection module will be helpful for both event and temporal relation extraction tasks. To improve the performance on *VAGUE* pairs, we could either build a stronger model that incorporates both contextual information and commonsense knowledge or create datasets with annotations that better separate *VAGUE* from other positive temporal relations.

| **Type 1**: Event predicted as non-event | 189 pairs |
|---|---|
| **Ex 1.** What Microsoft gets are developers around the world **working** on ideas that could potentially **open** up Kinect for Windows ... | |
| **Type 2**: *NONE* predicted as true relation | 135 pairs |
| **Ex 2.** Mr. Netanyahu told Mr. Erdogan that what **happened** on board the Mavi Marmara was "unintentional" ... , **according** to the statement. | |
| **Type 3**: *VAGUE* relation | 87 pairs |
| **Ex 3.** Microsoft **said** it has identified 3 companies for the China program to run through June. The company **gives** each participating startup $ 20,000 to create ... | |

Table 7: Error Types Based on MATRES Test Data

## 7 Conclusion

In this paper we investigate building an end-to-end event temporal relation extraction system. We propose a novel neural structured prediction model with joint representation learning to make predictions on events and relations simultaneously; this can avoid error propagation in previous pipeline systems. Experiments and comparative studies on two benchmark datasets show that the proposed model is effective for end-to-end event temporal relation extraction. Specifically, we improve the performances of previously published systems by 10% and 6.8% on the TB-Dense and MATRES datasets, respectively.

Future research can focus on creating more robust structured constraints between events and relations, especially considering event types, to improve the quality of global assignments using ILP. Since a better event model is generally helpful for relation extraction, another promising direction would be to incorporate multiple datasets to enhance the performance of our event extraction systems.

## Acknowledgements

This work is supported in part by Contracts W911NF-15-1-0543 and HR0011-18-2-0052 with the US Defense Advanced Research Projects Agency (DARPA). Approved for Public Release, Distribution Unlimited. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

## References


Steven Bethard. 2013. Cleartk-TimeML: A minimalist approach to TempEval 2013. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 10–14. Association for Computational Linguistics.

Steven Bethard, James H. Martin, and Sara Klingenstein. 2007. Timelines from text: Identification of syntactic temporal relations. In *Proceedings of the International Conference on Semantic Computing*, ICSC '07, pages 11–18, Washington, DC, USA. IEEE Computer Society.

Philip Bramsen, Pawan Deshpande, Yoong Keok Lee, and Regina Barzilay. 2006. Inducing temporal graphs. In *EMNLP*, Sydney, Australia.

Taylor Cassidy, Bill McDowell, Nathanael Chambers, and Steven Bethard. 2014. An annotation framework for dense event ordering. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 501–506. Association for Computational Linguistics.

Nate Chambers. 2013. Navytime: Event and time ordering from raw text. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 73–77, Atlanta, Georgia, USA. Association for Computational Linguistics.

Nathanael Chambers, Taylor Cassidy, Bill McDowell, and Steven Bethard. 2014. Dense event ordering with a multi-pass architecture. In *ACL*.

Nathanael Chambers and Dan Jurafsky. 2008. Jointly combining implicit constraints improves temporal ordering. In *EMNLP*, Honolulu, United States.

Nathanael Chambers, Shan Wang, and Dan Jurafsky. 2007. Classifying temporal relations between events. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 173–176, Stroudsburg, PA, USA. Association for Computational Linguistics.

Fei Cheng and Yusuke Miyao. 2017. Classifying temporal relations by bidirectional LSTM over dependency paths. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 1–6.

Pascal Denis and Philippe Muller. 2011. Predicting globally-coherent temporal structures from texts via endpoint inference and graph decomposition. In *IJCAI*, Barcelone, Spain.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Quang Xuan Do, Wei Lu, and Dan Roth. 2012. Joint inference for event timeline construction. In *EMNLP*, Jeju, Korea.

Xiaocheng Feng, Lifu Huang, Duyu Tang, Heng Ji, Bing Qin, and Ting Liu. 2016. A language-independent neural network for event detection. pages 66–71.

Natsuda Laokulrat, Makoto Miwa, Yoshimasa Tsuruoka, and Takashi Chikayama. 2013. UTTime: Temporal relation classification using deep syntactic features. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 88–92, Atlanta, Georgia, USA. Association for Computational Linguistics.

Artuur Leeuwenberg and Marie-Francine Moens. 2017. Structured learning for temporal relation extraction from clinical records. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 1150–1158.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 402–412, Baltimore, Maryland. Association for Computational Linguistics.

Inderjeet Mani, Marc Verhagen, Ben Wellner, Chong Min Lee, and James Pustejovsky. 2006. Machine learning of temporal relations. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, pages 753–760, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yuanliang Meng and Anna Rumshisky. 2018. Context-Aware neural model for temporal information extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Yuanliang Meng, Anna Rumshisky, and Alexey Romanov. 2017. Temporal information extraction for question answering using syntactic dependencies in an lstm-based architecture. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 887–896.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using LSTMs on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1116, Berlin,

Germany. Association for Computational Linguistics.

Makoto Miwa and Yutaka Sasaki. 2014. Modeling joint entity and relation extraction with table representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1858–1869, Doha, Qatar. Association for Computational Linguistics.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 300–309, San Diego, California. Association for Computational Linguistics.

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *ACL*.

Qiang Ning, Zhili Feng, and Dan Roth. 2017. A structured learning approach to temporal relation extraction. In *EMNLP*.

Qiang Ning, Zhili Feng, Hao Wu, and Dan Roth. 2018a. Joint reasoning for temporal and causal relations. In *ACL*.

Qiang Ning, Hao Wu, and Dan Roth. 2018b. A Multi-Axis annotation scheme for event temporal relations. In *ACL*.

Qiang Ning, Ben Zhou, Zhili Feng, Haoruo Peng, and Dan Roth. 2018c. CogCompTime: A tool for understanding time in natural language. In *EMNLP*.

Tim O'Gorman, Kristin Wright-Bettner, and Martha Palmer. 2016. Richer Event Description: Integrating event coreference with temporal, causal and bridging annotation. In *Proceedings of 2nd Workshop on Computing News Storylines*, pages 47–56. Association for Computational Linguistics.

James Pustejovsky, Patrick Hanks, Roser Sauri, Andrew See, Robert Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, and Lisa Ferro. 2003. The TIMEBANK corpus. In *Corpus linguistics*, pages 647–656.

Julien Tourille, Olivier Ferret, Aurelie Neveol, and Xavier Tannier. 2017. Neural architecture for temporal relation extraction: a bi-lstm approach for detecting narrative containers. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 224–230.

Naushad UzZaman, Hector Llorens, Leon Derczynski, James Allen, Marc Verhagen, and James Pustejovsky. 2013. SemEval-2013 task 1: Tempeval-3: Evaluating time expressions, events, and temporal relations. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 1–9. Association for Computational Linguistics.

Marc Verhagen, Robert Gaizauskas, Frank Schilder, Mark Hepple, Graham Katz, and James Pustejovsky. 2007. SemEval-2007 task 15: TempEval temporal relation identification. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, SemEval '07, pages 75–80, Stroudsburg, PA, USA. Association for Computational Linguistics.

Marc Verhagen and James Pustejovsky. 2008. Temporal processing with the TARSQI toolkit. In *22Nd International Conference on on Computational Linguistics: Demonstration Papers*, COLING '08, pages 189–192, Stroudsburg, PA, USA. Association for Computational Linguistics.

Marc Verhagen, Roser Saurí, Tommaso Caselli, and James Pustejovsky. 2010. SemEval-2010 task 13: Tempeval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval '10, pages 57–62, Stroudsburg, PA, USA. Association for Computational Linguistics.

Katsumasa Yoshikawa, Sebastian Riedel, Masayuki Asahara, and Yuji Matsumoto. 2009. Jointly identifying temporal relations with Markov logic. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 405–413. Association for Computational Linguistics.

Meishan Zhang, Yue Zhang, and Guohong Fu. 2017. End-to-end neural relation extraction with global optimization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1730–1740, Copenhagen, Denmark. Association for Computational Linguistics.

# Hierarchical Text Classification with Reinforced Label Assignment

**Yuning Mao**[1], **Jingjing Tian**[2], **Jiawei Han**[1], **Xiang Ren**[3]
[1]Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA
[3]Department of Computer Science, Peking University, Beijing, China
[3]Department of Computer Science, University of Southern California, CA, USA
[1]{yuningm2, hanj}@illinois.edu [2]tianjj97@pku.edu.cn [3]xiangren@usc.edu

## Abstract

While existing hierarchical text classification (HTC) methods attempt to capture label hierarchies for model training, they either make local decisions regarding each label or completely ignore the hierarchy information during inference. To solve the mismatch between training and inference as well as modeling label dependencies in a more principled way, we formulate HTC as a Markov decision process and propose to learn a **L**abel **A**ssignment **P**olicy via deep reinforcement learning to determine *where to place* an object and *when to stop* the assignment process. The proposed method, **HiLAP**, explores the hierarchy during both training and inference time in a *consistent* manner and makes *inter-dependent* decisions. As a general framework, HiLAP can incorporate different neural encoders as *base models* for end-to-end training. Experiments on five public datasets and four base models show that HiLAP yields an average improvement of 33.4% in Macro-F1 over flat classifiers and outperforms state-of-the-art HTC methods by a large margin.[1]

## 1 Introduction

In recent years there has been a surge of interest in leveraging hierarchies (taxonomies) to organize objects (*e.g.*, documents), leading to the development of hierarchical text classification (HTC)—a task that aims to predict for an object multiple appropriate labels in a given label hierarchy, which together constitute a sub-tree. HTC methods have found a wide range of applications such as question answering (Qu et al., 2012), online advertising (Agrawal et al., 2013), and scientific literature organization (Peng et al., 2016). In contrast to "flat" classification, the key challenges of HTC lie in modeling the large-scale, imbalanced, and in particular, structured label space.

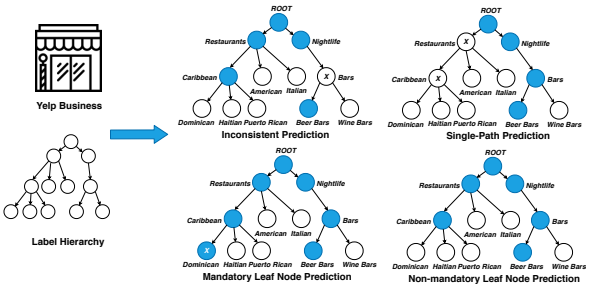


Figure 1: We aim at **consistent, multi-path, and non-mandatory leaf node prediction**. For a Caribbean restaurant with a beer bar, inconsistent prediction may place it to node "Beer Bars" but not "Bars", which contradicts with each other; Single-path prediction may only recognize that it is a beer bar; Mandatory leaf node prediction would have to assign a leaf node "Dominican" even if the nation of the cuisine is uncertain.

Based on how the hierarchy is explored, HTC methods can be summarized into *flat*, *local*, and *global* approaches (Silla and Freitas, 2011). Flat approaches (Hayete and Bienkowska, 2005; Johnson and Zhang, 2014) assume all the labels in the given hierarchy are independent. Some predict labels at the leaf nodes and heuristically add their ancestor labels, which is problematic as the labels of some objects may not be at the leaf nodes (*non-mandatory leaf node prediction*, see Fig. 1) and all the non-leaf nodes are completely neglected. Some simply ignore the hierarchy and perform standard multi-label classification, in which *label inconsistencies* (one label is predicted positive but its ancestors are not) may occur and post-processing is needed to correct such contradictions. Local approaches (Koller and Sahami, 1997; Cesa-Bianchi et al., 2006) train a set of local classifiers that function *independently* and predictions are usually made in a top-down order: one node is visited if and only if its ancestors have

[1]Data and code can be found at https://github.com/morningmoni/HiLAP.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 445–455, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

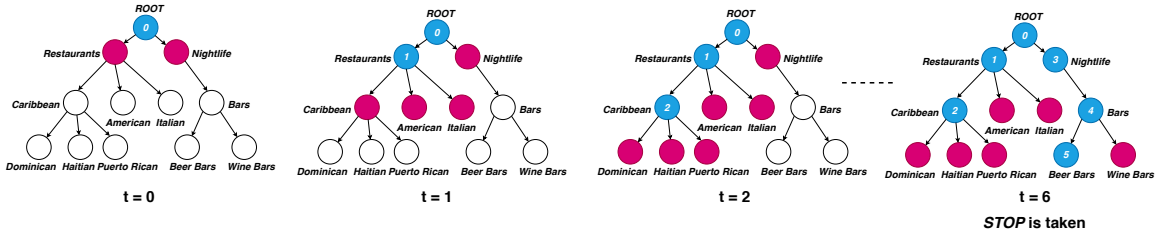


Figure 2: **An illustrative example of the label assignment policy.** At $t = 0$, $x_i$ is placed at the root label and the policy would decide if $x_i$ should be placed to its two children (*red*). At $t = 1$, $x_i$ is placed at label "Restaurants", which adds its three children as the candidates. At $t = 6$, the *stop* action is taken and the label assignment is thus terminated. We then take all the labels where $x_i$ has been placed (*blue*) as $x_i$'s labels.

been predicted positive. One critical issue is that the number of local classifiers depends on the size of the label hierarchy, making local approaches infeasible to scale.

Global approaches use one single classifier and model the label hierarchy more explicitly. Traditional global approaches (Wang et al., 2001; Silla Jr and Freitas, 2009) are largely based on specific flat models and often make unrealistic assumptions (Cai and Hofmann, 2004) as in flat approaches. Recent neural approaches (Kim, 2014; Yang et al., 2016) mainly focus on flat classification while their performance in HTC is relatively less studied. Even if the classification is supposed to be hierarchical, prior work (Gopal and Yang, 2013; Johnson and Zhang, 2014; Peng et al., 2018) still make *flat* and *independent* predictions or utilize simple constraints without considering the holistic quality of label assignment. One recent framework (Wehrmann et al., 2018) attempts to leverage both local and global information but it uses static features as input and its inference process is still flat.

In this paper, we formulate HTC as a Markov decision process to better capture *label dependencies* and measure the *holistic quality* of label assignment. We present HiLAP, a global framework that learns a label assignment policy to determine *where to place* the objects and *when to stop* the assignment process. HiLAP explores the label hierarchy during both training and inference in a consistent manner, which alleviates the exposure bias often found in prior local and global approaches. By learning when to stop, HiLAP is more flexible than approaches that only support mandatory leaf node prediction or require thresholding. In addition, HiLAP supports multi-path prediction and its predictions of one object on different paths are *inter-dependent*, which not only guarantees *label consistency* but matches the nature of HTC. Furthermore, HiLAP estimates the holistic quality of all the labels assigned to one object via reinforcement learning instead of evaluating each label independently via maximum likelihood as in prior studies. To summarize, HiLAP achieves better effectiveness compared to flat and local approaches as it examines the label hierarchy during both training and inference. HiLAP has more flexibility and generalization capacity than previous global approaches in that it has no constraints on the structure of the hierarchy or the labels of the objects (Cai and Hofmann, 2004), generalizes to neural representation learning models (Gopal and Yang, 2013), and makes inter-dependent predictions while ensuring label consistency (Wehrmann et al., 2018; Peng et al., 2018).

HiLAP can be combined with various neural encoding models and trained in an end-to-end fashion. In our experiments, we select four representative encoding models as the *base models* to evaluate the effectiveness of HiLAP. Experimental results on five public datasets from different domains show that combining the base models with HiLAP yields an average performance improvement of 33.4% in Macro-F1 over corresponding flat classifiers and outperforms state-of-the-art HTC methods by a large margin. In particular, ablation study shows that HiLAP is especially beneficial to those unpopular labels at the bottom levels.

## 2 Hierarchical Label Assignment

### 2.1 Overview

**Problem Formulation.** We define a label hierarchy $H = (L, E)$ as a tree or DAG (directed acyclic graph)-structured hierarchy with a set of nodes (labels) $L$ and a set of edges $E$ indicating the parent-child relation between the labels. Taking a set of objects $\mathcal{X} = \{x_1, x_2, ..., x_N\}$ and their labels $\mathcal{L} = \{L_1, L_2, ..., L_N\}$ as input, we aim to

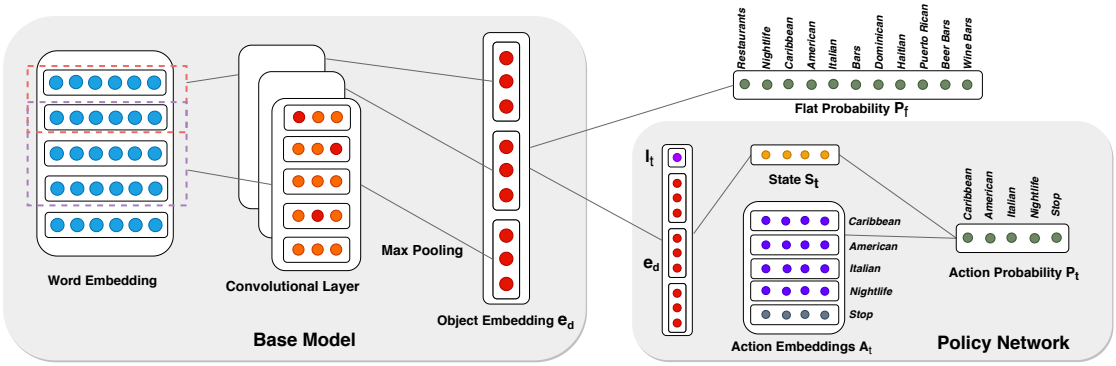


Figure 3: **The architecture of the proposed framework HiLAP**. One CNN model (Kim, 2014) is used as the base model for illustration. The object embedding $\mathbf{e}_d$ generated by the base model is combined with the embedding of currently assigned label $\mathbf{l}_t$ and used as the state representation $\mathbf{s}_t$, based on which actions are taken by the policy network. The time corresponds to $t = 1$ in Fig. 2.

learn a label assignment policy $\mathcal{P}$ to place each object $x_i$ to its labels $L_i$ on the label hierarchy $H$. The label assignment is supposed to be *consistent, multi-path, and non-mandatory leaf node prediction* (refer to Figs. 1 and 2). We define one *base model* $\mathcal{B}$ as a mapping $f$ that converts raw object $x_i$ to a finite dimensional vector, *i.e.*, the object embedding $\mathbf{e}_d \in \mathbb{R}^D$. $\mathcal{B}$ can be any neural representation learning model and its output $\mathbf{e}_d$ is used as the input of $\mathcal{P}$ for policy learning. The major challenge, compared to standard classification setup, is that we need to model $E$, *i.e.*, the relation between labels.

**Our Framework.** Prior studies either have a mismatch between training and inference as different routines are followed in the two phases, or compute losses with respect to each individual label and make flat predictions during inference time. In contrast, we learn a policy that (1) makes consistent, inter-dependent predictions by traversing the label hierarchy and maintaining state representation; (2) measures the holistic quality of label assignment via reinforcement learning. Specifically, the policy $\mathcal{P}$ puts $x_i$ at the root label in the beginning. At each time step, $\mathcal{P}$ decides which label $x_i$ should be further placed to, among all the *children* labels of where $x_i$ has been placed, until a special *stop* action is taken. An illustration of how HiLAP labels one object is shown in Fig. 2 and the overall architecture of HiLAP is shown in Fig. 3.

## 2.2 Reinforcement Learning for Hierarchical Label Assignment

We describe the details of policy learning including its actions, rewards, states, and the policy network in this section. We formulate HTC as a Markov decision process (MDP): at each time step, the agent observes current state, takes an action, and receives a reward. The end goal is to train a policy network to determine where to place the objects and when to stop.

**Actions.** Specifically, we regard the process of placing an object $x_i$ to the right positions on the label hierarchy as making a sequence of actions, where an action $a_t$ at time step $t$ is to select one label $l_{t+1}$ from the action space $\mathcal{A}_t$ and place $x_i$ to that label $l_{t+1}$. We denote the children of label $l_t$ as $\mathcal{C}(l_t)$. At the beginning of each episode, $x_i$ is placed at the root label $l_0$ and the action space $\mathcal{A}_0 = \mathcal{C}(l_0)$, *i.e.*, all the labels at level 1. When $x_i$ is placed at another label $l_1$, its children $\mathcal{C}(l_1)$ are then added to the action space $\mathcal{A}_1$ while $l_1$ itself is removed. In addition, one *stop* action with embedding $\mathbf{e}_{\text{stop}} \in \mathbb{R}^C$ is included in the action space so that the model can automatically learn when to stop placing object $x_i$ to new labels. Intuitively, when the confidence of placing $x_i$ to another label is lower than the *stop* action, the label assignment process would be terminated.

In short, the action space $\mathcal{A}_t$ consists of all the *unvisited* children labels of where the object $x_i$ *has been placed* and the *stop* action. One distinction of HiLAP is that it takes the inter-dependencies of labels across different paths and levels into consideration while previous approaches make independent predictions on different paths. For example, HiLAP can first place $x_i$ to a label at level 3 if the probability of that label is high and then place it to another label at level 1 on another path.

**Rewards.** The agent receives scalar rewards as feedback for its actions. Different from exist-

ing work where each label of one example[2] is treated independently, HiLAP measures the quality of all the labels assigned to each example $x_i$ by rewarding the agent with the *Example-based F1* (see Sec. 4.1 for details of this metric). Intuitively, the agent would realize how similar the assigned and the ground-truth labels of *one example* are. Instead of waiting until the end of the label assignment process and comparing the predicted labels with the gold labels, we use reward shaping (Mao et al., 2018), *i.e.*, giving intermediate rewards at each time step, to accelerate the learning process. Specifically, we set the reward $r$ of $x_i$ at time step $t$ to be the difference of Example-based F1 scores between current and the last time step: $r_t^{x_i} = \text{F1}_t^{x_i} - \text{F1}_{t-1}^{x_i}$.

If current F1 is better than that at the last time step, the reward would be positive, and vice versa. The cumulative reward from current time step to the end of an episode would cancel the intermediate rewards and thus reflect whether the current action improves the holistic label assignment or not. As a result, the learned policy would not focus on the current placement but have a long-term view that takes following actions into account.

**States and Policy Network.** We parameterize action $a_t$ by a policy network $\pi(\mathbf{a} \mid \mathbf{s}; \mathbf{W})$. For each object, its representation $\mathbf{e}_d$ is generated by the base model $\mathcal{B}$. For each label, a label embedding $\mathbf{l} \in \mathbb{R}^C$ is randomly initialized and updated during training. The embeddings of the object $\mathbf{e}_d$ and currently assigned label $\mathbf{l}_t$ are concatenated and projected to a vector $\mathbf{s}_t \in \mathbb{R}^C$ via a two-layer feed-forward network. $\mathbf{s}_t$ has the same size as the label embedding $\mathbf{l}$ and is used as the state representation at time step $t$. By stacking the action embeddings (*i.e.*, the embeddings of candidate labels and *stop* action), we obtain an action matrix $\mathbf{A}_t$ with size $|\mathcal{A}| \times C$. $\mathbf{A}_t$ is multiplied with the state embedding $\mathbf{s}_t$, which outputs the probability distribution of actions. Finally, an action $a_t$ is sampled based on the probability distribution of the action space.

$$\begin{aligned} \mathbf{s}_t &= \text{ReLU}(\mathbf{W}_l^1 \text{ReLU}(\mathbf{W}_l^2[\mathbf{e}_d; \mathbf{l}_t])), \\ \pi(\mathbf{a}_t \mid \mathbf{s}; \mathbf{W}) &= \text{softmax}(\mathbf{A}_t \mathbf{s}_t), \\ a_t &\sim \pi(\mathbf{a}_t \mid \mathbf{s}; \mathbf{W}). \end{aligned}$$

We use policy gradient (Williams, 1992) as the optimization algorithm. In addition, we adopt a *self-critical* training approach (Rennie et al., 2017). For each object $x_i$, two label assignments are generated: $\tilde{L}_{x_i}$ is sampled from the probability distribution, and $\hat{L}_{x_i}$, the baseline label assignment, is greedily obtained by choosing the action with the highest probability at each time step. We use $\tilde{r}_t^{x_i} = r_t^{\tilde{L}_{x_i}} - r_t^{\hat{L}_{x_i}}$ as the actual reward, which ensures that the policy network learns to place the object to positions with higher F1 score than the greedy baseline. Formally, we measure the global loss $\mathcal{O}_g$ as follows.

$$\mathcal{O}_g = -\sum_{i=1}^{N}\sum_{t=1}^{T} log\pi^{x_i}(a_t \mid \mathbf{s}; \mathbf{W}) \times v_t^{x_i},$$

where $v_j^{x_i} = \sum_{t=j}^{T} \gamma^{t-j} \tilde{r}_t^{x_i}$ is the cumulative future reward at time $j$ and $\gamma \in [0, 1]$ is the discount factor. At the time of inference, we greedily select labels with the highest probability as $\hat{L}_{x_i}$.

## 3 End-to-End Model Learning

### 3.1 Top-Down Supervised Pre-Training

Instead of learning from scratch, we use supervised learning to pre-train HiLAP. We denote the supervised variant as HiLAP-SL. While most parameters of HiLAP-SL are shared and used to initialize HiLAP (except that $\mathbf{e}_{\text{stop}}$ is randomly initialized), its way of exploring the label hierarchy $H$ is dissimilar.

The major difference is that HiLAP-SL explores the label hierarchy $H$ in a top-down manner independently. At each time step $t$, the object goes down one level on the hierarchy and the labels under the same parent are discriminated locally. Specifically, the local per-parent label probability distribution $\mathbf{p}_t^{\text{Local}}$ is estimated as $\mathbf{p}_t^{\text{Local}} = \sigma(\mathbf{C}_t \mathbf{s}_t)$, where $\sigma$ denotes the sigmoid function, and $\mathbf{C}_t \in \mathbb{R}^{|\mathcal{C}(l_t)| \times C}$ denotes the candidate embeddings of HiLAP-SL, *i.e.*, an embedding matrix consisting of the children of *current* label $l_t$, rather than *all* the labels where $x_i$ has been placed.

Another difference is that in HiLAP the actions are sampled and thus might place the objects to incorrect labels, while in HiLAP-SL only the ground-truth positions are traversed during training. Specifically, if there are $K(\geq 1)$ *ground-truth* labels at the same level, the object embedding $\mathbf{e}_d$ would be copied $K$ times and losses on the $K$ different paths would be measured independently (see Fig. 6 in Appendix for illustration). The local loss of HiLAP-SL is defined as $\mathcal{O}_l = \sum_{t=0}^{T} \mathcal{O}_t$, where $T$ is the lowest label's

[2] We use "example" and "object" interchangeably.

level of one example and $\mathcal{O}_t$ estimates the binary cross entropy over the candidate labels $\mathcal{C}(l_t)$: $\mathcal{O}_t = -\sum_{i=1}^{N}\sum_{l\in\mathcal{C}(l_{t,i})} L_i(l) \times log\mathbf{p}_{t,i}^{\text{Local}}(l) + (1 - L_i(l)) \times log(1 - \mathbf{p}_{t,i}^{\text{Local}}(l))$, where $L_i(l)$ and $\mathbf{p}_{t,i}^{\text{Local}}(l)$ evaluate label $l$ of $x_i$. Intuitively, HiLAP-SL works as if there were a set of local classifiers, although most of its parameters (except for the label embedding $\mathbf{l}$) are shared by all the labels so that there is no need to train multiple classifiers.

### 3.2 Combining Flat, Local, and Global Information for Policy Learning

We further add a *flat component* to HiLAP as a regularization of the base model. Specifically, the flat component is a feed-forward network that projects the object embedding $\mathbf{e}_d$ to a label probability distribution $\mathbf{p}^{\text{Flat}}$ of all the labels on the hierarchy: $\mathbf{p}^{\text{Flat}} = \sigma(\mathbf{W}^{\text{Flat}}\mathbf{e}_d)$. The combination of the base model and the flat component functions the same as a flat model and ensures that the object representation $\mathbf{e}_d$ has the capability of flat classification. We denote the flat loss that measures the binary cross entropy over all the labels by $\mathcal{O}_f = -\sum_{i=1}^{N}\sum_{l\in L} L_i(l) \times log\mathbf{p}_i^{\text{Flat}}(l) + (1 - L_i(l)) \times log(1 - \mathbf{p}_i^{\text{Flat}}(l))$. Combining the flat and local losses, the supervised loss in HiLAP-SL is defined as $\mathcal{O}_{\text{SL}} = \lambda\mathcal{O}_f + (1 - \lambda)\mathcal{O}_l$, where $\lambda \in [0, 1]$ is the mixing ratio. Similar to Celikyilmaz et al. (2018), we also found that mixing a proportion of the supervised loss is beneficial to the learning process of HiLAP. Further combining the global information $\mathcal{O}_g$ (*i.e.*, $\mathcal{O}_{\text{RL}}$), the total loss of HiLAP is defined as $\mathcal{O} = \mathcal{O}_{\text{RL}} + \alpha\mathcal{O}_{\text{SL}}$, where $\alpha$ is a scaling factor accounting for the difference in magnitude between $\mathcal{O}_{\text{RL}}$ and $\mathcal{O}_{\text{SL}}$. While we do not use the flat component during inference, it helps the representation learning of the base model and improves the performance of both HiLAP-SL and HiLAP (see Sec. 4.5).

## 4 Experiments

### 4.1 Experiment Setup

**Datasets.** We conduct extensive experiments on five public datasets from various domains (summarized in Table 1 and detailed in Appendix A). The first two datasets are related to news categorization, including RCV1 (Lewis et al., 2004) and the NYT annotated corpus (Sandhaus, 2008). The third dataset is the Yelp Dataset Challenge 2018[3]. We hypothesize that one business can be represented by its reviews and use the reviews to predict business categories. The last two datasets are related to protein functional catalogue (FunCat) and gene ontology (GO) prediction (Vens et al., 2008), which are used to test the generalization ability of HiLAP to non-textual data. For all the datasets, the lowest labels of one example may not be at the leaf nodes and there could be multiple labels at each level, making them harder and more realistic than *mandatory-leaf* or *single-path* datasets such as IPC (WIPO, 2014) and LSHTC (Partalas et al., 2015).

Table 1: **Statistics of the datasets.** $|L|$ denotes the number of labels in the label hierarchy. Avg($|L_i|$) and Max($|L_i|$) denote the average and maximum number of labels of one object, respectively.

| Dataset | Hierarchy | $\|L\|$ | Avg($\|L_i\|$) | Max($\|L_i\|$) | Training | Validation | Test |
|---|---|---|---|---|---|---|---|
| RCV1 | Tree | 103 | 3.24 | 17 | 23,149 | 2,315 | 781,265 |
| NYT | Tree | 115 | 2.52 | 14 | 25,279 | 2,528 | 10,828 |
| Yelp | DAG | 539 | 3.77 | 32 | 87,375 | 8,737 | 37,265 |
| FunCat | Tree | 499 | 8.76 | 45 | 1,628 | 848 | 1,281 |
| GO | DAG | 4,125 | 34.9 | 141 | 1,625 | 848 | 1,278 |

**Evaluation Metrics.** We use standard metrics (Johnson and Zhang, 2014; Meng et al., 2018; Peng et al., 2018) for HTC, including **Micro-F1**, **Macro-F1**, and **Example-based F1 (EBF)** (Partalas et al., 2015; Peng et al., 2016). Let $TP_i$, $FP_i$, $FN_i$ denote the true positive, false positive, and false negative for the i-th *example* $x_i$ in object set $X$, respectively. EBF calculates the F1 scores of all the *examples* independently and averages them. $P^{x_i} = \frac{TP_i}{TP_i+FP_i}$, $R^{x_i} = \frac{TP_i}{TP_i+FN_i}$, $\text{F1}^{x_i} = \frac{2P^{x_i}\times R^{x_i}}{P^{x_i}+R^{x_i}}$, and $\text{EBF} = \frac{1}{N}\sum_{i=1}^{N}\text{F1}^{x_i}$. Recall that $\text{F1}^{x_i}$ is used as the reward in HiLAP.

**Base Models for Feature Encoding.** Different from most of existing *global* HTC methods that rely on pre-specified features (Gopal and Yang, 2013) as input or build on specific models (Cai and Hofmann, 2004; Vens et al., 2008; Silla Jr and Freitas, 2009), our framework is trained in an end-to-end manner by leveraging a differentiable feature representation learning model as the *base model*. Specifically, we use **TextCNN** (Kim, 2014), **HAN** (Yang et al., 2016), **bow-CNN** (Johnson and Zhang, 2014) on the three textual datasets, and a **feed-forward network** on the two non-textual datasets. The details of the base models

[3] https://www.yelp.com/dataset/challenge

Table 2: **Performance comparison on RCV1.** * denotes the results reported in Peng et al. (2018) on the same dataset split. Note that the results of HR-SVM reported in Gopal and Yang (2013) are not comparable as they use a different hierarchy with 137 labels.

| | Method | Micro-F1 | Macro-F1 | EBF |
|---|---|---|---|---|
| Flat | Leaf-SVM* | 69.1 | 33.0 | - |
| | SVM | 80.4 | 46.2 | 80.5 |
| | TextCNN | 76.6 | 43.0 | 75.8 |
| | HAN | 75.3 | 40.6 | 76.1 |
| | bow-CNN | 82.7 | 44.7 | 83.3 |
| Local & Global | TD-SVM | 80.1 | 50.7 | 80.5 |
| | HSVM* | 69.3 | 33.3 | - |
| | HR-SVM* | 72.8 | 38.6 | - |
| | HR-DGCNN* | 76.1 | 43.2 | - |
| | HMCN | 80.8 | 54.6 | 82.2 |
| | **HiLAP** (TextCNN) | 78.6 | 50.5 | 80.1 |
| | **HiLAP** (HAN) | 75.4 | 45.5 | 77.4 |
| | **HiLAP** (bow-CNN) | **83.3** | **60.1** | **85.0** |

Table 3: **Performance comparison on the NYT and Yelp datasets.** We mainly compare with competitive baselines that perform well on RCV1.

| Method | NYT | | | Yelp | | |
|---|---|---|---|---|---|---|
| | Micro-F1 | Macro-F1 | EBF | Micro-F1 | Macro-F1 | EBF |
| SVM | 72.4 | 37.1 | 74.0 | 66.9 | 36.3 | 68.0 |
| TextCNN | 69.5 | 39.5 | 71.6 | 62.8 | 27.3 | 63.1 |
| HAN | 62.8 | 22.8 | 65.5 | 66.7 | 29.0 | 67.9 |
| bow-CNN | 72.9 | 33.4 | 74.1 | 63.6 | 23.9 | 63.9 |
| TD-SVM | 73.7 | 43.7 | 75.0 | 67.2 | 40.5 | 67.8 |
| HMCN | 72.2 | 47.4 | 74.2 | 66.4 | 42.7 | 67.6 |
| **HiLAP** (TextCNN) | 69.9 | 43.2 | 72.8 | 65.5 | 37.3 | 68.4 |
| **HiLAP** (HAN) | 65.2 | 28.7 | 68.0 | **69.7** | 38.1 | **72.4** |
| **HiLAP** (bow-CNN) | **74.6** | **51.6** | **76.6** | 68.9 | **42.8** | 71.5 |

are provided in Appendix C due to limited space.

To incorporate one base model into our framework, we remove its final feed-forward layer that projects the object representation $\mathbf{e}_d$ to a flat probability distribution of all labels ($\mathbf{p}^{\text{Flat}}$), and use $\mathbf{e}_d$ directly as the input of HiLAP. As one will see in the later experiments, HiLAP consistently improves the base model by modeling the label hierarchy in an effective manner.

### 4.2 Compared Methods

**1. Traditional HTC Methods.** A major line of work for HTC is Support Vector Machines (SVM) and its hierarchical variants. Specifically, **SVM** performs standard multi-label classification using one-vs-the-rest (OvR) strategy. **Leaf-SVM** treats each leaf node as a label and adds the ancestors of predicted leaf nodes. Variants such as **HSVM** (Tsochantaridis et al., 2005), Top-Down SVM (**TD-SVM**) (Liu et al., 2005), and Hierarchically Regularized SVM (**HR-SVM**) (Gopal and Yang, 2013) are also tested. Other state-of-the-art HTC methods that we compare with include **Clus-HMC** (Vens et al., 2008) and **CSSA** (Bi and Kwok, 2011).

**2. Neural HTC Methods.** There are not many neural methods that specifically target HTC. We mainly compare with two *latest* neural models: **HR-DGCNN** (Peng et al., 2018), which extends hierarchical regularization (Gopal and Yang, 2013) to Graph-CNN and compares favorably to flat models like RCNN (Lai et al., 2015) and XML-CNN (Liu et al., 2017), and **HMCN** (Wehrmann et al., 2018), which outperforms state-of-the-art HTC methods such as HMC-LMLP (Cerri et al., 2016). We also compare with the base models that we use for feature encoding. The main aim is to see how much gain they could obtain by combining each one of them with HiLAP.

### 4.3 Implementation Details

For datasets without held-out set, we randomly sample 10% from the training set as the validation set following Johnson and Zhang (2014); Peng et al. (2018). We only use the first 256 tokens of each document for representation learning. All the models are trained using an Adam optimizer with initial learning rate 1e-3 and weight decay 1e-6. We use GloVe (Pennington et al., 2014) with size 50 as word embeddings for TextCNN (Kim, 2014) and HAN (Yang et al., 2016). We create a vocabulary of the most frequent 30,000 words in the training data and generate multi-hot vectors as the input of bow-CNN (Johnson and Zhang, 2014). For our framework, since the parameter updates are performed after $T$ steps, we cache the object representation $\mathbf{e}_d$ and reuse it at each step for better efficiency. More details are provided in Appendix D for reproducibility.

### 4.4 Performance Comparison

**1. Comparison with State-of-the-art Methods.** We compare the performance of HiLAP to state-of-the-art HTC methods and show the results in Tables 2 and 3. On RCV1, HiLAP (HAN) achieves similar performance to HR-DGCNN even though the corresponding base model HAN is originally worse than HR-DGCNN. HiLAP (TextCNN) outperforms most baselines in Macro-F1 and perform similarly to TD-SVM despite that it uses one global classifier while TD-

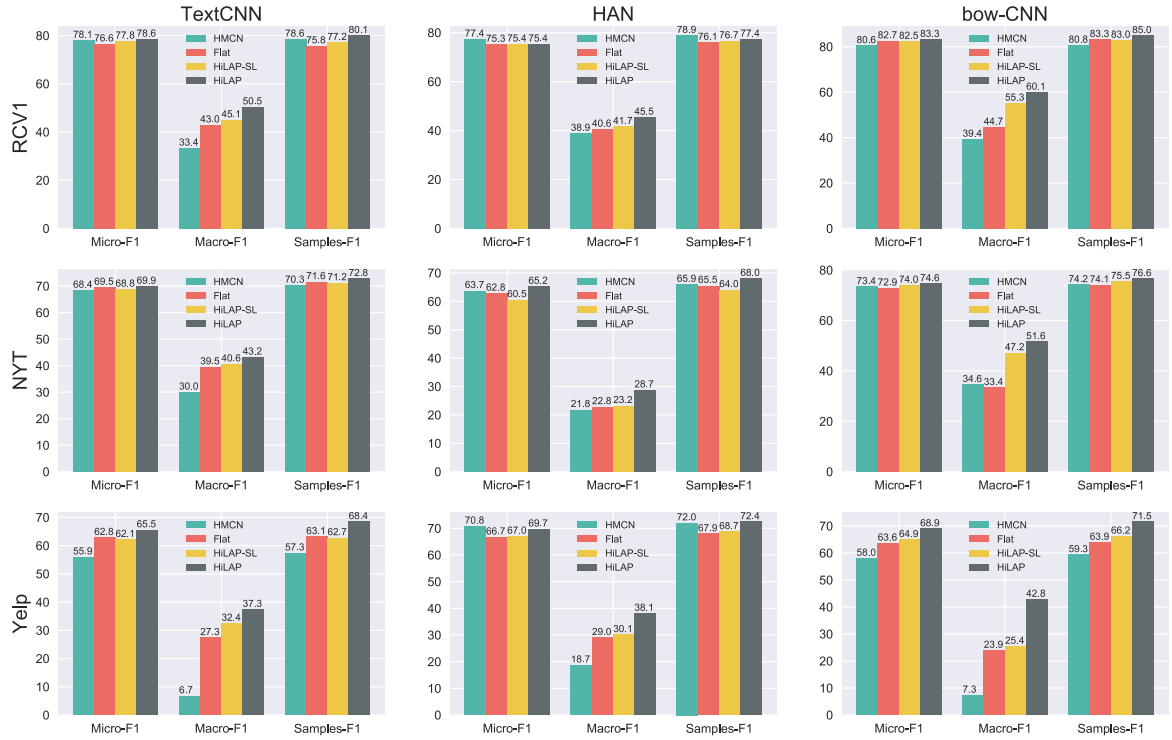


Figure 4: **Performance comparison of different classification frameworks using the same base models.** We compare HiLAP with its flat, supervised variants, and HMCN. Results show that HiLAP exhibits consistent improvement over flat classifiers and larger gains than HMCN.

SVM uses a set of classifiers. Among all compared methods, HiLAP (bow-CNN) achieves the best performance on all the three metrics.[4] On NYT, similar results are observed: TextCNN and HAN are both improved when combining with HiLAP and HiLAP (bow-CNN) again achieves the best performance. On Yelp, HiLAP (HAN) achieves the best Micro-F1 and EBF, while HiLAP (bow-CNN) obtains the highest Macro-F1.

**2. Comparison using Same Base Models.** We compare the performance of different frameworks that support the use of *exactly the same* base models and summarize the results in Fig. 4.[5] Due to the extreme imbalance of the data, directly applying a flat model may suffer from low Macro-F1, *i.e.*, the predictions of flat models are inevitably biased to the most popular labels. HMCN also has the same issue, resulting in Macro-F1 lower than 10 when combining with some base models. In contrast, HiLAP outperforms the baselines significantly in Macro-F1, which implies that our method is better at tackling labels with relatively few examples. It is also observed that HiLAP-SL sometimes may have a negative effect in terms of Micro-F1, although it is usually marginal compared with the gains in Macro-F1. However, such negative effects are eliminated by HiLAP through better exploration of the label hierarchy. Overall, HiLAP achieves the highest performance on 24 of 27 results among the combinations of three datasets, three base models, and three evaluation metrics. In particular, HiLAP yields an average improvement of 33.4% in Macro-F1 compared to corresponding base models.

Table 4: **Performance comparison on Functional Catalogue and Gene Ontology.** We compare with state-of-the-art hierarchical classification methods that take exactly the same raw features as input (*i.e.*, we exclude models designed specifically for text objects).

| Method | FunCat | | | GO | | |
|---|---|---|---|---|---|---|
| | Micro-F1 | Macro-F1 | EBF | Micro-F1 | Macro-F1 | EBF |
| SVM | 2.72 | 1.21 | 3.42 | 34.1 | 1.46 | 36.8 |
| CSSA | 16.0 | 4.60 | 14.8 | 11.6 | 0.76 | 11.5 |
| CLUS-HMC | 25.2 | 4.14 | 24.1 | 41.4 | 3.01 | 40.3 |
| HMCN | 21.3 | 5.07 | 21.5 | 43.2 | 3.81 | 43.3 |
| **HiLAP** | **26.5** | **7.50** | **27.4** | **45.4** | **5.87** | **45.2** |

**3. Results on Functional Genomics Prediction.** We compare HiLAP with CSSA (Bi and

[4]The results are not comparable with Johnson and Zhang (2014) due to implementation details and the fact that they tune the threshold for each label using k-fold cross-validation. See Appendix B for more discussions.

[5]For HMCN, we replace its static features with the same base model for fair comparison.

Table 5: **Ablation study of HiLAP.** We evaluate variants of HiLAP using bow-CNN (Johnson and Zhang, 2014) on RCV1 (Lewis et al., 2004).

| Method | Micro-F1 | Macro-F1 | EBF |
|---|---|---|---|
| Flat-Only | 82.7 | 44.7 | 83.3 |
| HiLAP-SL-NoFlat | 81.0 | 52.1 | 81.7 |
| HiLAP-SL | 82.5 | 55.3 | 83.0 |
| HiLAP-NoSL | 83.2 | 59.3 | 85.0 |
| HiLAP-NoFlat | 83.0 | 59.8 | 84.7 |
| HiLAP | **83.3** | **60.1** | **85.0** |

Kwok, 2011), CLUS-HMC (Vens et al., 2008), and HMCN (Wehrmann et al., 2018) on the FunCat and GO datasets, as they represent the state-of-the-art on these datasets. An SVM classifier is also evaluated to better understand the difficulties of the task. We use the same raw features as the input of all the methods for apples-to-apples comparison and list the results in Table 4. Note that the metric *area under the average precision-recall curve (AUPRC)* (Wehrmann et al., 2018) is not applicable because HiLAP does not use a flat probability distribution of all the labels. As one can see, HiLAP outperforms all the baselines on both datasets by a large margin. In particular, we observe significant improvement on Macro-F1 over the best baseline (47.9% and 53.9%, respectively), which shows that our method is especially better at classifying sparse labels than previous approaches.

### 4.5 Performance Analysis

**1. Ablation Study on Different Framework Components.** We show the ablation analysis of HiLAP in Table 5. Using *Flat-Only* degenerates HiLAP to the flat baseline. By comparing the results of *Flat-Only* and HiLAP-SL-NoFlat (a variant of HiLAP-SL without flat loss), we further confirm that flat approaches are likely to neglect sparse labels, which results in low Macro-F1. Local approaches (HiLAP-SL-NoFlat), on the other hand, are slightly worse in terms of Micro-F1 and EBF but significantly better on Macro-F1. By combining flat and local information, HiLAP-SL achieves performance close to *Flat-Only* on Micro-F1 and EBF, and even higher Macro-F1 than HiLAP-SL-NoFlat. HiLAP-NoSL is initialized by the pre-trained HiLAP-SL model without mixing the supervised loss during its training. We can see that using the reinforced loss alone still improves the performance on all the three metrics. After removing the flat loss during the training of HiLAP, HiLAP-NoFlat shows slightly lower performance than the full HiLAP model, indicating that the flat component serves as a regularization of the base model and is beneficial to the overall performance.

**2. Performance Study on Label Granularity and Popularity.** We analyze the sources of performance gains by dividing the labels based on their levels and number of supporting examples. Fig. 5 shows the absolute Macro-F1 differences between several methods and the base model. We observe similar results for other setups and omit them for a clearer view. As depicted in Fig. 5, HiLAP and HiLAP-SL are especially beneficial to unpopular labels (P3) at the bottom levels (L3).

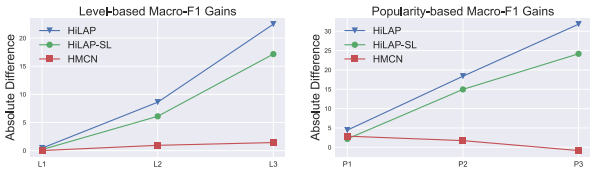


Figure 5: **Performance Study on Label Granularity and Popularity.** We compute level-based and popularity-based Macro-F1 gains on NYT with bow-CNN as base model. We denote the levels of the hierarchy with L1, L2, and L3 (*left*) and divide the labels into three equal sized categories (P1, P2, and P3) in a descending order by their number of examples (*right*).

**3. Analysis of Label Inconsistency.** Label inconsistencies often happen in approaches that perform flat inference, but they are not measured by standard evaluation metrics like F1 scores. To provide a picture of how severe the issue is, we further conduct experiments to check the percentage of objects that are predicted with inconsistent labels (Table 6). We found, for example, 29,186/781,265 (3.74%) predictions of TextCNN have inconsistent on RCV1. In contrast, HiLAP ensures **0%** label inconsistency without the need of post-processing, because its predictions are always valid sub-trees of the label hierarchy (refer to Fig. 2).

Table 6: **Analysis of Label Inconsistency.** We compare various methods by the percentage of predictions with inconsistent labels on RCV1 (Lewis et al., 2004).

| SVM | TextCNN | HMCN | HiLAP |
|---|---|---|---|
| 4.83% | 3.74% | 3.84% | **0%** |

## 5 Related Work

Hierarchical classification approaches have been developed for many applications. For text classification, both traditional methods (Lewis et al., 2004; Gopal and Yang, 2013) and neural methods (Johnson and Zhang, 2014; Peng et al., 2018) have been proposed to classify, *e.g.*, the topics of newswire and web content (Sun and Lim, 2001) or categories of laws and patents (Bi and Kwok, 2015; Cai and Hofmann, 2004; Rousu et al., 2005). Many previous studies (Liu et al., 2005; Sun and Lim, 2001) train a set of local classifiers and make predictions in a top-down manner. In particular, Bi and Kwok (2015) develop Bayes-optimal predictions that minimize the global risks but their model is still locally trained. Such local approaches are not popularly used among recent neural-based HTC models (Johnson and Zhang, 2014; Peng et al., 2018) since it is usually infeasible to train many neural classifiers locally.

Global methods, on the other hand, train only one classifier. Although global methods are desirable, they are relatively less studied due to the complexity of the problem. Existing global models are generally modified based on specific flat models. Hierarchical-SVM (Cai and Hofmann, 2004; Qiu et al., 2009) generalizes Support Vector Machine (SVM) learning based on discriminant functions that are structured in a way that mirrors the label hierarchy. One limitation is that Hierarchical-SVM only supports balanced tree (all possible labels are presumed to be at the same level in their experiments). Hierarchical naive Bayes (Silla Jr and Freitas, 2009) modifies naive Bayes by updating weights of one's ancestors as well whenever one label's weights are updated. There are other global methods that are based on association rules (Wang et al., 2001), C4.5 (Clare and King, 2003), kernel machines (Rousu et al., 2005), and decision tree (Vens et al., 2008). Constraints such as the regularization that enforces the parameters of one node and its parent to be similar (Gopal and Yang, 2013) are also proposed to leverage the label hierarchy while maintaining scalability. However, their use of the label hierarchies is somewhat limited compared with HiLAP.

## 6 Conclusions

We proposed an end-to-end reinforcement learning approach to hierarchical text classification (HTC) where objects are labeled by placing them at the proper positions in the label hierarchy. The proposed framework makes *consistent* and *inter-dependent* predictions, in which any neural-based representation learning model can be used as a base model and a label assignment policy is learned to determine where to place the objects and when to stop the assignment process. Experiments on five public datasets and four base models showed that our approach outperforms state-of-the-art HTC methods significantly. For future work, we will explore the effectiveness of the proposed framework on other base models and forms of data (*e.g.*, images). We will introduce more losses covering other aspects in the objective function to further improve the performance of our framework.

## Acknowledgments

Research was sponsored in part by U.S. Army Research Lab under Cooperative Agreement No. W911NF-09-2-0053 (NSCTA), DARPA under Agreement No. W911NF-17-C-0099, National Science Foundation IIS 16-18481, IIS 17-04532, and IIS-17-41317, grant 1U54GM-114838 awarded by NIGMS, National Science Foundation SMA 18-29268, DARPA MCS and GAILA, IARPA BETTER, Schmidt Family Foundation, Amazon Faculty Award, Google Research Award, Snapchat Gift, and JP Morgan AI Research Award. We thank Chao Zhang, Xiao-Yang Liu, Qingrong Chen, Jun Yan, collaborators in the INK research lab, and anonymous reviewers for their help and valuable feedback.

## References

Rahul Agrawal, Archit Gupta, Yashoteja Prabhu, and Manik Varma. 2013. Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. In *WWW*, pages 13–24. ACM.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Wei Bi and Jame T Kwok. 2015. Bayes-optimal hierarchical multilabel classification. *TKDE*, 27(11):2907–2918.

Wei Bi and James T Kwok. 2011. Multi-label classification on tree-and dag-structured hierarchies. In *ICML-11*, pages 17–24.

Lijuan Cai and Thomas Hofmann. 2004. Hierarchical document categorization with support vector machines. In *CIKM*, pages 78–87. ACM.

Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep communicating agents for abstractive summarization. In *NAACL*, pages 1662–1675.

Ricardo Cerri, Rodrigo C Barros, André CPLF de Carvalho, and Yaochu Jin. 2016. Reduction strategies for hierarchical multi-label classification in protein function prediction. *BMC bioinformatics*, 17(1):373.

Nicolò Cesa-Bianchi, Claudio Gentile, and Luca Zaniboni. 2006. Hierarchical classification: combining bayes with svm. In *ICML*, pages 177–184. ACM.

Amanda Clare and Ross D King. 2003. Predicting gene function in saccharomyces cerevisiae. *Bioinformatics*, 19(suppl_2):ii42–ii49.

Siddharth Gopal and Yiming Yang. 2013. Recursive regularization for large-scale classification with hierarchical and graphical dependencies. In *KDD*, pages 257–265. ACM.

Boris Hayete and Jadwiga R Bienkowska. 2005. Gotrees: predicting go associations from protein domain composition using decision trees. In *Biocomputing 2005*, pages 127–138. World Scientific.

Rie Johnson and Tong Zhang. 2014. Effective use of word order for text categorization with convolutional neural networks. *arXiv preprint arXiv:1412.1058*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Daphne Koller and Mehran Sahami. 1997. Hierarchically classifying documents using very few words. In *ICML*, pages 170–178. Morgan Kaufmann Publishers Inc.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *AAAI*, volume 333, pages 2267–2273.

David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. 2004. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397.

Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. 2017. Deep learning for extreme multi-label text classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 115–124. ACM.

Tie-Yan Liu, Yiming Yang, Hao Wan, Hua-Jun Zeng, Zheng Chen, and Wei-Ying Ma. 2005. Support vector machines classification with a very large-scale taxonomy. *Acm Sigkdd Explorations Newsletter*, 7(1):36–43.

Yuning Mao, Xiang Ren, Jiaming Shen, Xiaotao Gu, and Jiawei Han. 2018. End-to-end reinforcement learning for automatic taxonomy induction. In *ACL*, pages 2462–2472. Association for Computational Linguistics.

Yu Meng, Jiaming Shen, Chao Zhang, and Jiawei Han. 2018. Weakly-supervised neural text classification. In *CIKM*.

Ioannis Partalas, Aris Kosmopoulos, Nicolas Baskiotis, Thierry Artières, George Paliouras, Éric Gaussier, Ion Androutsopoulos, Massih-Reza Amini, and Patrick Gallinari. 2015. LSHTC: A benchmark for large-scale text classification. *CoRR*, abs/1503.08581.

Hao Peng, Jianxin Li, Yu He, Yaopeng Liu, Mengjiao Bao, Lihong Wang, Yangqiu Song, and Qiang Yang. 2018. Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In *WWW*, pages 1063–1072.

Shengwen Peng, Ronghui You, Hongning Wang, Chengxiang Zhai, Hiroshi Mamitsuka, and Shanfeng Zhu. 2016. Deepmesh: deep semantic representation for improving large-scale mesh indexing. *Bioinformatics*, 32(12):i70–i79.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Xipeng Qiu, Wenjun Gao, and Xuanjing Huang. 2009. Hierarchical multi-class text categorization with global margin maximization. In *acl-ijcnlp 2009*, pages 165–168.

Bo Qu, Gao Cong, Cuiping Li, Aixin Sun, and Hong Chen. 2012. An evaluation of classification models for question topic categorization. *JASIST*, 63:889–903.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *CVPR*, page 3.

Juho Rousu, Craig Saunders, Sandor Szedmak, and John Shawe-Taylor. 2005. Learning hierarchical multi-category text classification models. In *ICML*, pages 744–751. ACM.

Evan Sandhaus. 2008. The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752.

Carlos N Silla and Alex A Freitas. 2011. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, 22(1-2):31–72.

Carlos N Silla Jr and Alex A Freitas. 2009. A global-model naive bayes approach to the hierarchical prediction of protein functions. In *ICDM'09*, pages 992–997. IEEE.

Aixin Sun and Ee-Peng Lim. 2001. Hierarchical text classification and evaluation. In *ICDM*.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. 2005. Large margin methods for structured and interdependent output variables. *Journal of machine learning research*, 6(Sep):1453–1484.

Celine Vens, Jan Struyf, Leander Schietgat, Sašo Džeroski, and Hendrik Blockeel. 2008. Decision trees for hierarchical multi-label classification. *Machine Learning*, 73(2):185.

Ke Wang, Senqiang Zhou, and Yu He. 2001. Hierarchical classification of real life documents. In *SDM*, pages 1–16. SIAM.

Jonatas Wehrmann, Ricardo Cerri, and Rodrigo Barros. 2018. Hierarchical multi-label classification networks. In *ICML*, pages 5225–5234.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

IPC WIPO. 2014. International patent classification (ipc). *World Intellectual Property Organization, Geneve*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *NAACL*, pages 1480–1489.

# Investigating Capsule Network and Semantic Feature on Hyperplanes for Text Classification

[12]Chunning Du [*], [12]Haifeng Sun[*], [12]Jingyu Wang[†], [12]Qi Qi, [12]Jianxin Liao
[12]Chun Wang, [12]Bing Ma
[1]State Key Laboratory of Networking and Switching Technology,
Beijing University of Posts and Telecommunications, Beijing 100876, China
[2]EBUPT Information Technology Co., Ltd., Beijing 100191, China
{duchunning,sunhaifeng_1,wangjingyu,qiqi}@ebupt.com

## Abstract

As an essential component of natural language processing, text classification relies on deep learning in recent years. Various neural networks are designed for text classification on the basis of word embedding. However, polysemy is a fundamental feature of the natural language, which brings challenges to text classification. One polysemic word contains more than one sense, while the word embedding procedure conflates different senses of a polysemic word into a single vector. Extracting the distinct representation for the specific sense could thus lead to fine-grained models with strong generalization ability. It has been demonstrated that multiple senses of a word actually reside in linear superposition within the word embedding so that specific senses can be extracted from the original word embedding. Therefore, we propose to use capsule networks to construct the vectorized representation of semantics and utilize hyperplanes to decompose each capsule to acquire the specific senses. A novel dynamic routing mechanism named 'routing-on-hyperplane' will select the proper sense for the downstream classification task. Our model is evaluated on 6 different datasets, and the experimental results show that our model is capable of extracting more discriminative semantic features and yields a significant performance gain compared to other baseline methods.

## 1 Introduction

Text classification is a crucial task in natural language processing, which has many applications, such as sentiment analysis, intent identification and topic labeling[Aggarwal and Zhai, 2012; Wang and Manning, 2012a]. Recent years, many studies rely on neural networks and have shown promising performance.

The success of deep learning model for NLP is based on the progress in learning distributed word representations in semantic vector space, where each word is mapped to a vector called a word embedding. The word's representation is calculated relying on the *distributional hypothesis* - the assumption that semantically similar or related words appear in similar contexts [Mikolov *et al.*, 2013; Langendoen, 1959]. Normally, each word's representation is constructed by counting all its context features. However, for the polysemic word which contains multiple senses, the context features of different senses are mixed together, leading to inaccurate word representation. As demonstrated in [Arora *et al.*, 2018], multiple senses of a word actually reside in linear superposition within the word embedding:

$$v \approx \alpha_1 v_{sense1} + \alpha_2 v_{sense2} + \alpha_3 v_{sense3} + \cdots , \quad (1)$$

where coefficients $\alpha_i$ are nonnegative and $v_{sense1}, v_{sense2}...$ are the hypothetical embeddings of different senses. As a result, the word embedding $v$ deviates from any sense, which brings ambiguity for the subsequent task. It demands us to extract the separate senses from the overall word representation to avoid the ambiguity.

Similar to the word embedding, the recently proposed capsule network constructs vectorized representations for different entities[Hinton *et al.*, 2018; Sabour *et al.*, 2017; Hinton *et al.*, 2011]. A dynamic routing mechanism, 'routing-by-agreement', is implemented to ensure that the output of the capsule gets sent to an appropriate parent in the layer above. Very recently, capsule network is applied in the field of NLP where each capsule is obtained from the word embedding. Compared with the standard neural nets using a single scalar (the output of a neural unit) to represent the detected semantics, the vectorized rep-

[*] Authors contributed equally.
[†] Corresponding author.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 456–465,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

resentation in capsule network enables us to utilize hyperplanes to extract the component from the overall representation and get the specific sense.

Therefore, we propose to attach different hyperplanes to capsules to tackle the ambiguity caused by polysemy. Each capsule is decomposed by projecting the output vector on the hyperplanes, which can extract the specific semantic feature. The projected capsule denotes a specific sense of the target words, and a novel dynamic routing mechanism named 'routing-on-hyperplane' will decide which specific senses are selected for the downstream classification and which senses are ignored. Similar to routing-by-agreement [Sabour *et al.*, 2017], we aim to activate a higher-level capsule whose output vector is agreed with the predictions from the lower-level capsules. Differently, before the active capsule at a level makes predictions for the next-level capsules, the capsule's output vector will be projected on the trainable hyperplane. The hyperplanes will be trained discriminatively to extract specific senses. Moreover, in order to encourage the diversity of the hyperplanes, a well-designed penalization term is implemented in our model. We define the cosine similarity between the normal vectors of the hyperplanes as a measure of redundancy, and minimize it together with the original loss.

We test our model (HCapsNet) on the text classification task and conduct extensive experiments on 6 datasets. Experimental results show that the proposed model could learn more discriminative features and outperform other baselines. Our main contributions are summarized as follows:

- We explore the capsule network for text classification and propose to decompose capsules by means of projecting on hyperplanes to tackle the polysemy problem in natural language.
- Propose routing-on-hyperplane to dynamically select specific senses for the subsequent classification. A penalization term is designed to obtain diversified hyperplanes and offer multiple senses representations of words.
- Our work is among the few studies which prove that the idea of capsule networks have promising applications on natural language processing tasks.

## 2 Related Work

### 2.1 Neural Networks for Text Classification

Various neural networks for text classification have been proposed based on the word embedding. Commonly used models include convolutional neural networks [Kim, 2014], recursive neural network [Socher *et al.*, 2013] and recurrent neural networks. There have been several recent studies of CNN for text classification in the large training dataset and deep complex model structures [Schwenk *et al.*, 2017; Johnson and Zhang, 2017]. Some models were proposed to combine the strength of CNN and RNN [Lai *et al.*, 2015; Zhang *et al.*, 2016]. Moreover, the accuracy was further improved by attention-based neural networks[Lin *et al.*, 2017; Vaswani *et al.*, 2017; Yang *et al.*, 2016]. However, these models are less efficient than capsule networks.

As a universal phenomenon of language, polysemy calls much attention of linguists. It has been demonstrated that learning a distinct representation for each sense of an ambiguous word could lead to more powerful and fine-grained models based on vector-space representations [Li and Jurafsky, 2015].

### 2.2 Capsule Network

Capsule network was proposed to improve the representational limitations of CNN and RNN by extracting features in the form of vectors. The technique was firstly proposed in [Hinton *et al.*, 2011] and improved in [Sabour *et al.*, 2017] and [Hinton *et al.*, 2018]. Vector-based representation is able to encode latent inter-dependencies between groups of input features during the learning process. Introducing capsules also allows us to use routing mechanism to generate high-level features which is a more efficient way for feature encoding.

Several types of capsule networks have been proposed for natural language processing. Yang *et al.* [2018] investigated capsule networks with routing-by-agreement for text classification. They also found that capsule networks exhibit significant improvement when transfer single-label to multi-label text classification. Capsule networks also show a good performance in multi-task learning [Xiao *et al.*, 2018]. Xia *et al.* [2018] discovered the capsule-based model's potential on zero-shot learning. However, existing capsule networks for natural language processing cannot model the

polysemic words or phrases which contain multiple senses.

## 3 Model

In this section, we begin by introducing the idea of routing-on-hyperplane and formulate it in details. Then the architecture of the HCapsNet is formally presented in the second subsection. Finally, the penalization term and loss function implemented in this paper are explained.

### 3.1 Routing-on-hyperplane

Suppose that we have already decided on the output vectors of all the capsules in the layer $L$ and we now want to decide which capsules to activate in the layer $L+1$. We should also consider how to assign each active capsule in the layer $L$ to one active capsule in the layer $L+1$. The output vector of capsule $i$ in the layer $L$ is denoted by $\boldsymbol{u}_i$, and the output vector of capsule $j$ in the layer $L+1$ is denoted by $\boldsymbol{v}_j$.

Firstly, for all the capsules in the layer $L$, we attach the trainable hyperplane to each capsule. Capsule's output vectors will be projected on the hyperplanes before making predictions. More specifically, for capsule $i$, we define the trainable matrix $\boldsymbol{W}_i^h$, which is used to decide the normal vector $\boldsymbol{w}_i$ of the attached hyperplane. By restricting $\|\boldsymbol{w}_i\|_2 = 1$, we can get the projected capsule's output vector $\boldsymbol{u}_{\perp i}$:

$$\boldsymbol{w}_i = \boldsymbol{W}_i^h \boldsymbol{u}_i. \tag{2}$$

$$\boldsymbol{u}_{\perp i} = \boldsymbol{u}_i - \boldsymbol{w}_i^{\mathsf{T}} \boldsymbol{u}_i \boldsymbol{w}_i. \tag{3}$$

In this way, the output vectors of capsules will be projected on the specific hyperplanes to get different components which denote the specific senses in our task. To retain or ignore a specific sense (projected capsule) will be decided through an iterative procedure. The procedure contains making predictions and calculating the agreement. When one projected capsule's prediction is highly agreed with one target parent capsule, the probability of retaining the projected capsule gets gradually larger. In another word, when a specific sense is highly relevant with the subsequent classification, we choose to keep it and ignore others.

Therefore, the $\boldsymbol{u}_{\perp i}$ will then be used to make predictions for the $L+1$ layer's capsules and calculate coupling coefficients $c_{ij}$. When making predictions, the capsules in the layer $L$ will multiply their projected output vector $\boldsymbol{u}_{\perp i}$ by a weight matrix $\boldsymbol{W}_{ij}$:

$$\hat{\boldsymbol{u}}_{\perp j|i} = \boldsymbol{W}_{ij} \boldsymbol{u}_{\perp i}, \tag{4}$$

where $\hat{\boldsymbol{u}}_{\perp j|i}$ denotes the 'vote' of the capsule $i$ for the capsule $j$. The agreement between the prediction vector $\hat{\boldsymbol{u}}_{\perp j|i}$ with current output vector of parent capsule $j$ will be fed back to the coupling coefficients $c_{ij}$ between the two capsules: increase $c_{ij}$ if highly agreed. Similar with [Sabour *et al.*, 2017] we define the agreement as scalar product between the two vectors. $b_{ij}$ is the accumulation of the agreement after each iteration and the softmax function is implemented to ensure the coupling coefficients between the capsule $i$ and all the capsules in the layer above sum to one:

$$b_{ij} \leftarrow b_{ij} + \hat{\boldsymbol{u}}_{\perp j|i} \cdot \boldsymbol{v}_j \tag{5}$$

$$c_{ij} = \frac{exp(b_{ij})}{\sum_k exp(b_{ik})} \tag{6}$$

**Algorithm 1** Routing-on-hyperplane returns the output vector of capsule $j$ in the layer $L+1$ given the output vector of capsule $i$ in the layer $L$. $\boldsymbol{W}_{ij}$ are trainable parameters denoting the transformation matrix between the two adjacent layers. $\boldsymbol{W}_i^h$ are trainable parameters for each capsule $i$ to calculate the proposed hyperplane's normal vectors $\boldsymbol{w}_i$, we restrict that $\|\boldsymbol{w}_i\|_2 = 1$.

initialize the routing logits:
for all capsule $i$ in the layer $L$ and capsule $j$ in the layer $L+1$: $b_{ij} \leftarrow 0$;
for every capsule $i$ in the layer $L$: $\boldsymbol{w}_i = \boldsymbol{W}_i^h \boldsymbol{u}_i$
for every capsule $i$ in the layer $L$: $\boldsymbol{u}_{\perp i} \leftarrow \boldsymbol{u}_i - \boldsymbol{w}_i^{\mathsf{T}} \boldsymbol{u}_i \boldsymbol{w}_i$
for every capsule $i$ in the layer $L$: $\hat{\boldsymbol{u}}_{\perp j|i} = \boldsymbol{W}_{ij} \boldsymbol{u}_{\perp i}$
**for** r iterations **do**
  for all capsule $i$ and $j$: $c_{ij} \leftarrow \frac{exp(b_{ij})}{\sum_k exp(b_{ik})}$
  for all capsule $j$ in the layer $L+1$: $\boldsymbol{s}_{\perp j} \leftarrow \sum_i c_{ij} \hat{\boldsymbol{u}}_{\perp j|i}$
  for all capsule $j$ in the layer $L+1$: $\boldsymbol{v}_j \leftarrow \frac{\|\boldsymbol{s}_{\perp j}\|^2}{1+\|\boldsymbol{s}_{\perp j}\|^2} \frac{\boldsymbol{s}_{\perp j}}{\|\boldsymbol{s}_{\perp j}\|}$
  for all capsule $i$ and capsule $j$: $b_{ij} \leftarrow b_{ij} + \hat{\boldsymbol{u}}_{\perp j|i} \cdot \boldsymbol{v}_j$
**end for**
**return** $\boldsymbol{v}_j$

Each iteration will result in a temporary output vector of the capsule $j$: the weighted sum over all prediction vectors $\hat{\boldsymbol{u}}_{\perp j|i}$ using coefficient $c_{ij}$. Moreover, to ensure the length of the output vector of capsule $j$ is able to represent the probability and prevent it from being too big, we use a non-linear 'squashing' function to make the vector's length range from zero to one without changing the vector's direction:

$$\boldsymbol{s}_{\perp j} = \sum_i c_{ij} \hat{\mathbf{u}}_{\perp j|i}. \tag{7}$$

$$\boldsymbol{v}_j = \frac{\|\boldsymbol{s}_{\perp j}\|^2}{1 + \|\boldsymbol{s}_{\perp j}\|^2} \frac{\boldsymbol{s}_{\perp j}}{\|\boldsymbol{s}_{\perp j}\|}. \tag{8}$$

The $\boldsymbol{v}_j$ will then be returned as input to calculate the agreement for the next iteration. The coupling coefficients $c_{ij}$ and the output vector of capsule $j$ gradually converge after several iterations. After the last iteration of the routing process, the coupling coefficients $c_{ij}$ is determined. Hyperplane plays the role to extract specific senses and assist to route the lower-level capsules to the right parent capsules. We detail the whole routing-on-hyperplane algorithm in Algorithm 1.

### 3.2 HCapsNet Model Architecture

We propose a model named HCapsNet for text classification based on the theory of capsule network and routing-on-hyperplane. The architecture is illustrated in Figure 1. The model consists of three layers: one bi-directional recurrent layer, one convolutional capsule layer, and one fully connected capsule layer. The input of the model is a sentence $S$ consisting of a sequence of word tokens $t_1, t_2, ..., t_n$. The output of the model contains a series of capsules. Each top-level capsule corresponds to a sentence category. The length of the top-level capsule's output vector is the probability $p$ that the input sentence $S$ belongs to the corresponding category.

The recurrent neural network can capture long-distance dependencies within a sentence. For this strength, a bi-directional recurrent neural network is the first layer of HCapsNet. We concatenate the left context and the right context as the word's elementary representation $x_i$, which is the input to the second layer:

$$c_l(t_i) = \overrightarrow{RNN}(t_i), \tag{9}$$

$$c_r(t_i) = \overleftarrow{RNN}(t_i), \tag{10}$$

$$x_i = [c_l(t_i), c_r(t_i)]. \tag{11}$$

The second layer is a convolutional capsule layer. This is the first layer consisting of capsules, we call capsules in this layer as primary capsules. Primary capsules are groups of detected features which means piecing instantiated parts together to make familiar wholes. Since the output of the bi-directional recurrent neural network is not in the form of capsules, no routing method is used in this layer.

The final layer is fully connected capsule layer. Each capsule corresponds to a sentence class. All the capsules in this layer receive the output of the lower-level capsules by the routing-on-hyperplane method as we described in Section 3.1. The length of the top-level capsule's output vector represents the probability that the input sentence belongs to the corresponding category.

### 3.3 Penalization Term

The HCapsNet may suffer from redundancy problem if the output vectors of capsules are always getting projected on the similar hyperplanes at the routing-on-hyperplane procedure. Thus, we need a penalization term to encourage the diversity of the hyperplanes. We introduce an easy penalization term with low time complexity and space cost. Firstly, we construct a matrix $X_i$ the columns of which is the normal vectors $w$ of the hyperplanes for the $i$th word. The dot product of $X_i$ and its transpose, subtracted by an identity matrix is defined as a measure of redundancy. The penalization term is the sum of all the words' redundancy:

$$P_i = ||(X_i X_i^{\mathsf{T}} - I)||_F^2. \tag{12}$$

$$P = \sum_i P_i, \tag{13}$$

where $||\bullet||_F$ stands for the Frobenius norm of a matrix. Similar to adding the L2 regularization term, this penalization term $P$ will be multiplied by a coefficient, and we minimize it together with the original loss.

Let's consider the two columns $w^a$ and $w^b$ in $X_i$, which are two normal vectors of hyperplanes for the $i$th word. We have restricted that $||w|| = 1$ as described in Algorithm 1. For any non-diagonal elements $x_{ab}$ $(a \neq b)$ in the $X_i X_i^{\mathsf{T}}$ matrix, it corresponds to the cosine similarity between the two

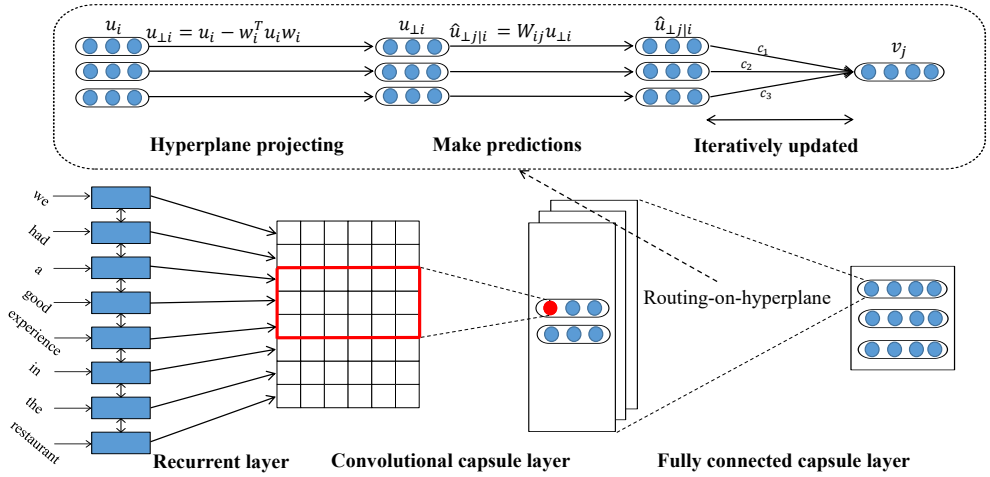


Figure 1: The architecture of HCapsNet

normal vectors:

$$-1 < x_{ab} = \sum_k w_k^a w_k^b < 1. \quad (14)$$

where $w_k^a$ and $w_k^b$ are $k$-th element in the $w^a$ and $w^b$ vectors, respectively. In the most extreme case, where the two normal vectors of hyperplanes are orthometric with each other, i.e. the word is projected to two extremely different meanings, the corresponding $x_{ab}$ is 0. Otherwise, the absolute value will be positive. In the other most extreme case, where the two normal vectors of hyperplanes are identical, i.e. the word is projected to the same vector, the corresponding absolute value of $x_{ab}$ is 1. The diagonal elements $x_{ab}$ $(a = b)$ in the $X_i X_i^\mathsf{T}$ matrix is the normal vectors' cosine similarity with themselves, so they are all 1. The $X_i X_i^\mathsf{T}$ is subtracted by an identity matrix $I$ so as to eliminate the meaningless elements. We minimize the Frobenius norm of $P_i$ to encourage the non-diagonal elements in $P_i$ to converge to 0, in another word, to encourage word vector to be projected on orthometric hyperplanes and get diversified explanation.

### 3.4 Loss Function

In HCapsNet, each top-level capsule corresponds to a sentence category. The length of the top-level capsule's output vector represents the probability that the input sentence belongs to the corresponding category. We would like the top-level capsule for the category $k$ to have a long output vector if the input sentence belongs to the category $k$ and have a short output vector if the input sentence does not belong to the category $k$. Similar with [Sabour *et al.*, 2017], We use a separate margin loss, $L_k$ for each top-level capsule $k$. The total loss $L$ is simply the sum of the losses of all top-level capsules:

$$L = \sum_k \{T_k \max(0, m^+ - ||\boldsymbol{v_k}||)^2 + \lambda_1 (1 - T_k) \max(0, ||\boldsymbol{v_k}|| - m^-)^2\} + \lambda_2 P, \quad (15)$$

where $T_k$ will be 1 if the sentence belongs to the k class, or else $T_k$ will be 0. $||\boldsymbol{v_k}||$ is the length of the output vector of capsule $k$. We introduce $\lambda_1$ to reduce the penalization to avoid shrinking the length of the capsules' output vectors in the initial learning stage. $P$ is the penalization term introduced in Section 3.3. In our experiments, $m^+ = 0.9$, $m^- = 0.1$, $\lambda_1 = 0.5$.

## 4 Experiments

We compare our method with the widely used text classification methods and baseline models (listed in Table 1).

### 4.1 Datasets

HCapsNet is evaluated on six widely studied datasets including three common text classification tasks: sentiment analysis, question classification and topic classification. These datasets are Stanford Sentiment Treebank [Socher *et al.*, 2013], Movie Review Data [Pang and Lee, 2005], Subjectivity dataset [Pang and Lee, 2004], TREC [Li and Roth, 2002] and AG's corpus of news articles [Zhang *et al.*, 2015b]. Summary statistics of the datasets are listed in Table2.

| Method | SST-2 | SST-5 | MR | Subj | TREC | AG's news |
|---|---|---|---|---|---|---|
| SVM [Socher *et al.*, 2013] | 79.4 | 40.7 | - | - | - | - |
| NB [Socher *et al.*, 2013] | 81.8 | 41.0 | - | - | - | - |
| NBSVM-bi [Wang and Manning, 2012b] | - | - | 79.4 | 93.2 | - | - |
| Standard-LSTM | 80.6 | 45.3 | 75.9 | 89.3 | 86.8 | 86.1 |
| bi-LSTM | 83.2 | 46.7 | 79.3 | 90.5 | 89.6 | 88.2 |
| RCNN [Lai *et al.*, 2015] | - | 47.21 | - | - | - | |
| SNN [Zhao *et al.*, 2018] | - | 50.4 | 82.1 | 93.9 | **96** | - |
| CNN-non-static [Kim, 2014] | 87.2 | 48.0 | 81.5 | 93.4 | 93.6 | 92.3 |
| VD-CNN [Schwenk *et al.*, 2017] | - | - | - | 88.2 | 85.4 | 91.3 |
| CL-CNN [Zhang *et al.*, 2015a] | - | - | - | 88.4 | 85.7 | 92.3 |
| Capsule-B [Yang *et al.*, 2018] | 86.8 | - | 82.3 | 93.8 | 93.2 | 92.6 |
| HCapsNet | **88.7** | **50.8** | **83.5** | **94.2** | 94.2 | **93.5** |

Table 1: Experimental results of our model compared with other models. Performance is measured in accuracy (%). Models are divided into 3 categories. The first part is baseline methods including SVM and Naive Bayes and their variations. The second part contains models about recurrent neural networks. The third part contains models about convolutional neural networks.

| Dataset | Class | Len | V | Train | Dev | Test |
|---|---|---|---|---|---|---|
| SST-2 | 2 | 54 | 16185 | 6920 | 872 | 1821 |
| SST-5 | 5 | 54 | 17836 | 8544 | 1101 | 2210 |
| MR | 2 | 58 | 18765 | 9596 | - | 1066 |
| Subj | 2 | 121 | 21323 | 9000 | - | 1000 |
| TREC | 6 | 37 | 9592 | 5452 | - | 500 |
| AG's news | 4 | 197 | 51379 | 120k | - | 7.6k |

Table 2: Summary statistics for the datasets.

### 4.2 Hyperparameters

In our experiments, we use 300-dimensional word2vec [Mikolov *et al.*, 2013] vectors to initialize word representations. In the first bi-directional RNN layer of HCapsNet, we use Long Short Term Memory network, the dimension of the hidden state is 256. The second layer contains 32 channels of primary capsules and the number of capsules in one channel depends on the sentence length. Each primary capsule contains 8 atoms which means that the dimension of the primary capsules is 8. The top-level capsules are obtained after 3 routing iterations. The dimension of the output vector of top-level capsules is 16. For all the datasets, we conduct mini-batch with size 25. We use Adam [Kingma and Ba, 2014] as our optimization method with $1e-3$ learning rate. $\lambda_2$ is 0.01.

### 4.3 Results and Discussions

Table 1 reports the results of our model on different datasets comparing with the widely used text classification methods and state-of-the-art approaches. We can have the following observations.

Our HCapsNet achieves the best results on 5 out of 6 datasets, which verifies the effectiveness of our model. In particular, HCapsNet outperforms vanilla capsule network Capsule-B[Yang *et al.*, 2018] by a remarkable margin, which only utilizes the dynamic routing mechanism without hyperplane projecting.

HCapNet does not perform best on the TREC dataset. One main reason maybe TREC dataset is used for question type classification, where samples are all question sentences. The task is mainly determined by interrogative words. For example, the sentence containing 'where' will probably be classified to 'location'. The ability to tackle polysemy doesn't play an important role. So, our model gets a similar result with Capsule-B.

### 4.4 Ablation Study

To analyze the effect of different components including hyperplane projection, penalization term, and routing iterations, we report the results of variants of HCapsNet in Table 4.

The results show that capsule network performs best when conducting 3 routing iterations, which stays in line with the conclusion in [Sabour *et*

| Sentence | Projected representation for polyseme | Capsule-B | HCapsNet |
|---|---|---|---|
| like old myths and **[wonder]** tales spun afresh. | [0.4632, 0.1429, -0.2311, 0.4664, 0.2448, -0.6256, -0.0975, 0.1784] | P$^{\checkmark}$ | P$^{\checkmark}$ |
| as your relatives swap one mundane story after another , you begin to **[wonder]** if they are ever going to depart. | [-0.059, -0.3657, -0.3788, -0.0716, 0.4276, -0.5122, 0.5092, 0.092] | N$^{\times}$ | P$^{\checkmark}$ |
| you **[wonder]** why enough was n't just a music video rather than a full-length movie . | [0.0137, -0.2311, -0.4361, 0.0194, 0.5012, -0.5311, 0.4617, -0.0082] | P$^{\times}$ | N$^{\checkmark}$ |
| trouble every day is a success in some sense , but it 's hard to like a film so **[cold]** and dead | [0.1758, -0.3037, -0.3679, 0.0992, 0.3996, -0.6481, 0.3668, 0.1267] | N$^{\checkmark}$ | N$^{\checkmark}$ |
| the **[cold]** and dreary weather is a perfect metaphor for the movie itself , which contains few laughs and not much drama | [-0.3810, -0.3923, -0.3016, -0.3045, 0.2417, -0.3109, 0.5999, -0.0391] | N$^{\times}$ | P$^{\checkmark}$ |

Table 3: Projected primary capsule's representations for polysemic words. P and N denote positive and negative classification results, respectively. ✓ denotes the right classification and × denotes the incorrect classification.

- Almost every scene in this film is a gem that could stand alone, a perfectly realized observation of mood, behavior and intent.
- A spunky, original take on a theme that will resonate with singles of many ages.
- The story drifts so inexorable into cliches about tortured lrb and torturing rrb artists and consuming but impossible love that you can't help but become more disappointed as each overwrought new sequence plods on.
- The premise is in extremely bad taste, and the film's supposed insights are so poorly thought out and substance free that even a high school senior taking his or her first psychology class could dismiss them.

Figure 2: Examples of routing results for SST-2.

*al.*, 2017; Yang *et al.*, 2018]. Compared with the vanilla capsule network (row 3), applying routing-on-hyperplane brings a noticeable improvement (row 2). This demonstrates the necessity of integrating hyperplane projecting at the routing procedure to tackle the polysemy problems. Moreover, the penalization term described in Section 3.3 also marginally improves the accuracy, which proves that the orthogonal constraint on hyperplane is beneficial for text classification.

| Hyperplane | Pena. | Iterations | Accuracy |
|---|---|---|---|
| ✓ | ✓ | 3 | **83.5** |
| ✓ | × | 3 | 83.2 |
| × | × | 3 | 82.5 |
| ✓ | ✓ | 1 | 81.5 |
| ✓ | ✓ | 5 | 82.6 |

Table 4: Ablation study on MR dataset. "Pena." denotes Penalization Term in Section 3.3.

### 4.5 Case Study

Table 3 shows some sample cases from SST validation dataset, which are movie reviews for sentiment analysis. We analyze the attended primary capsule representation for the polysemic words in brackets. Specifically, we report the output vectors of the projected primary capsule, which is mostly attended by the routing mechanism.

The word 'wonder' in the first sample sentence means something that fills you with surprise and admiration, which shows a very positive sentiment. However, the polysemic word 'wonder' in the second and third sentences means to think about something and try to decide what is true, which is neutral in sentiment. We can observe that for the same word, the attended projected capsule representations are quite different according to different word senses. The projected representations for the same sense are similar, the Euclidean distance is 0.23 (row 2,3). On the contrary, for the different senses, the Euclidean distance is 1.12 (row 1,2). This property helps our model to make the predictions all correctly, while Capsule-B [Yang *et al.*, 2018] can not handle the latter two sentences. Similarly, the word 'cold' conveys two different senses in the last two samples (row 4-5), which means cruel and low temperature, respectively. The corresponding projected vectors are also quite different, which verifies the ability to tackle polysemy by routing-on-hyperplane.

### 4.6 Visualizing Routing Results

After several iterations of the routing algorithm, each primary capsule and the top-level capsule will be connected via a calculated coupling coefficient. The coupling coefficient corresponds to how much contribution of a low-level capsule to a specific high-level capsule. Routing-on-hyperplane can also be viewed as a parallel attention mech-

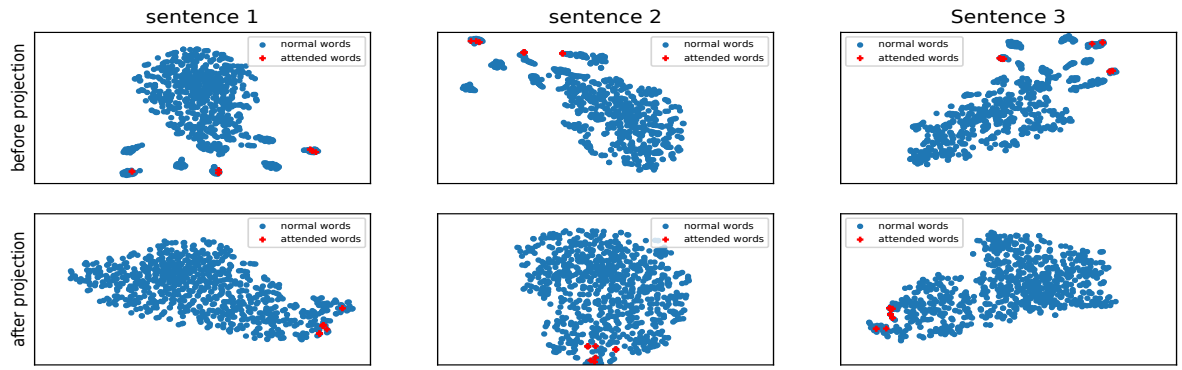


Figure 3: Illustrating the effect of the hyperplane.

anism that allows each capsule at one level to attend to some active capsules at the level below and to ignore others. We can thus draw a heat map to figure which phrases are taken into account a lot, and which ones are skipped by the routing-on-hyperplane in the task of text classification.

We randomly select 4 examples of reviews from the test set of SST, when the model has a high confidence (>0.8) in predicting the label. As shown in Figure 2, the words whose coupling coefficient greater than 0.7 are marked. It is easy to conclude that our routing method can effectively extract the sentimental words that indicate strongly on the sentiment behind the sentence and assign a greater coupling coefficient between the corresponding capsules. For example, '*gem*', '*spunky*', '*disappointed*', '*bad taste*' etc.

### 4.7 Visualizing Effects of The Hyperplane

In order to assess the effect of the hyperplane, we randomly select 3 examples in SST dataset and draw the distribution maps of primary capsules' output vectors before and after the projection operation respectively. As the dimension of the primary capsule's output vector is 8, T-Distributed Stochastic Neighbor Embedding (t-SNE) is performed on the vectors to reduce the dimension for visualization. As illustrated in Figure 3, the three pictures in the first line show the distribution before the projection operation for the three example sentences respectively. And the three pictures in the second line show the distribution after the projection. The blue points in the distribution maps denote the normal words and the red crosses denote the words attended by the routing algorithm which are defined in Section 4.6.

The relationship between the semantic capsules can be estimated by analyzing the distribution of the low-dimensional data in Figure 3. We find that originally scattered points which denote attended words converge after the projection. The attended words' projected vectors are close with each other, showing that they contain similar senses which are beneficial for the subsequent task. On the contrary, the capsules before projection contain multiple senses and show a scattered pattern. This demonstrates that the hyperplanes can effectively extract the guided senses and get attended by the routing-on-hyperplane mechanism.

## 5 Conclusion and Future Work

In this paper, we explore the capsule network for text classification and propose to decompose the capsule by means of projecting on hyperplanes to tackle the polysemy problem in natural language. Routing-on-hyperplane, a dynamic routing method, is implemented to select the sense-specific projected capsules for the subsequent classification task. We assess the effect of the hyperplane by case study and analyzing the distribution of the capsules' output vectors. The experiments demonstrate the superiority of HCapsNet and our proposed routing-on-hyperplane method outperforms the existing routing method in the text classification task.

In future, we would like to investigate the application of our theory in various tasks including reading comprehension and machine translating. We believe that capsule networks have broad applicability on the natural language processing tasks. Our core idea that decomposing the semantic capsules by projecting on hyperplanes is a necessary complement to capsule network to tackle the polysemy problem in various natural language processing tasks.

## Acknowledgements

This work was jointly supported by: (1) National Natural Science Foundation of China (No. 61771068, 61671079, 61471063, 61372120, 61421061); (2) Beijing Municipal Natural Science Foundation (No.4182041, 4152039); (3) the National Basic Research Program of China (No. 2013CB329102); (4) Fundamental Research Funds for the Central Universities under Grant 2018RC20; (5) BUPT Excellent Ph.D. Students Foundation.

## References

Charu C. Aggarwal and ChengXiang Zhai. A survey of text classification algorithms. In *Mining Text Data*. 2012.

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. Linear algebraic structure of word senses, with applications to polysemy. *TACL*, 2018.

Geoffrey E. Hinton, Alex Krizhevsky, and Sida D. Wang. Transforming auto-encoders. In *Artificial Neural Networks and Machine Learning*, 2011.

Geoffrey Hinton, Nicholas Frosst, and Sara Sabour. Matrix capsules with em routing. 2018.

Rie Johnson and Tong Zhang. Deep pyramid convolutional neural networks for text categorization. In *ACL*, 2017.

Yoon Kim. Convolutional neural networks for sentence classification. In *EMNLP*, 2014.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, 2014.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. Recurrent convolutional neural networks for text classification. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

D. Terence Langendoen. Studies in linguistic analysis. *International Journal of American Linguistics*, 1959.

Jiwei Li and Dan Jurafsky. Do multi-sense embeddings improve natural language understanding? In *EMNLP*, 2015.

Xin Li and Dan Roth. Learning question classifiers. In *19th International Conference on Computational Linguistics, COLING*, 2002.

Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. *international conference on learning representations*, 2017.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *Computer Science*, 2013.

Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, 2004.

Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL 2005*, 2005.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *NAACL HLT*, 2018.

Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. Dynamic routing between capsules. In *NIPS*, 2017.

Holger Schwenk, Loïc Barrault, Alexis Conneau, and Yann LeCun. Very deep convolutional networks for text classification. In *EACL*, 2017.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, 2013.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.

Sida I. Wang and Christopher D. Manning. Baselines and bigrams: Simple, good sentiment and topic classification. In *ACL*, 2012.

Sida I. Wang and Christopher D. Manning. Baselines and bigrams: Simple, good sentiment and topic classification. In *ACL*, 2012.

Congying Xia, Chenwei Zhang, Xiaohui Yan, Yi Chang, and Philip S. Yu. Zero-shot user intent detection via capsule neural networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

Liqiang Xiao, Honglun Zhang, Wenqing Chen, Yongkun Wang, and Yaohui Jin. Mcapsnet: Capsule network for text with multi-task learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. Hierarchical attention networks for document classification. In *NAACL HLT*, 2016.

Min Yang, Wei Zhao, Jianbo Ye, Zeyang Lei, Zhou Zhao, and Soufei Zhang. Investigating capsule networks with dynamic routing for text classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems*, 2015.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *NIPS*, 2015.

Rui Zhang, Honglak Lee, and Dragomir R. Radev. Dependency sensitive convolutional neural networks for modeling sentences and documents. In *NAACL HLT 2016*, 2016.

Jianyu Zhao, Zhiqiang Zhan, Qichuan Yang, Yang Zhang, Changjian Hu, Zhensheng Li, Liuxin Zhang, and Zhiqiang He. Adaptive learning of local semantic and global structure representations for text classification. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, 2018.

# Label-Specific Document Representation for Multi-Label Text Classification

**Lin Xiao, Xin Huang, Boli Chen, Liping Jing**
Beijing Key Lab of Traffic Data Analysis and Mining
Beijing Jiaotong University, China
{17112079, 18120367, 18120345, lpjing}@bjtu.edu.cn

## Abstract

Multi-label text classification (MLTC) aims to tag most relevant labels for the given document. In this paper, we propose a **L**abel-**S**pecific **A**ttention **N**etwork (**LSAN**) to learn the new document representation. LSAN takes advantage of label semantic information to determine the semantic connection between labels and document for constructing label-specific document representation. Meanwhile, the self-attention mechanism is adopted to identify the label-specific document representation from document content information. In order to seamlessly integrate the above two parts, an adaptive fusion strategy is designed, which can effectively output the comprehensive document representation to build multi-label text classifier. Extensive experimental results on four benchmark datasets demonstrate that LSAN consistently outperforms the state-of-the-art methods, especially on the prediction of low-frequency labels. The code and hyper-parameter settings are released to facilitate other researchers [1].

## 1 Introduction

Text classification is a fundamental text mining task including multi-class classification and multi-label classification. The former only assigns one label to the given document, while the latter classifies one document into different topics. In this paper, we focus on multi-label text classification (MLTC) because it has become one of the core tasks in natural language processing and has been widely applied in topic recognition (Yang et al., 2016), question answering (Kumar et al., 2016), sentimental analysis (Cambria et al., 2014) and so on. With the boom of big data, MLTC becomes significantly challenging because it has to handle the massive documents, words and labels simultaneously. Therefore, it is an emergency to develop effective multi-label text classifier for various practical applications.

Multi-label text classification allows for the coexistence of more than one label in a single document, thus, there are semantical correlations among labels because they may share the same subsets of document. Meanwhile, the document may be long and complicated semantic information may be hidden in the noisy or redundant content. Furthermore, most documents fall into few labels while a large number of "tail labels" only contain very few positive documents. To handle these issues, researchers pay much attention on three facets: 1) how to sufficiently capture the semantic patterns from the original documents, 2) how to extract the discriminative information related to the corresponding labels from each document, and 3) how to accurately mine the correlation among labels.

Till now, in the community of machine learning and natural language processing, researchers have paid tremendous efforts on developing MLTC methods in each facet. Among them, deep learning-based methods such as CNN (Liu et al., 2017; Kurata et al., 2016), RNN (Liu et al., 2016), combination of CNN and RNN (Lai et al., 2015; Chen et al., 2017), attention mechanism (Yang et al., 2016; You et al., 2018), (Adhikari et al., 2019) and etc., have achieved great success in document representation. However, most of them only focus on document representation but ignore the correlation among labels. Recently, some methods including DXML(Zhang et al., 2018), EXAM(Du et al., 2018), SGM(Yang et al., 2018), GILE(Pappas and Henderson, 2019) are proposed to capture the label correlations by exploiting label structure or label content. Although they obtained promising results in some cases, they still

[1] https://github.com/EMNLP2019LSAN/LSAN/



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 466–475, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

cannot work well when there is no big difference between label texts (e.g., the categories *Management* vs *Management moves* in Reuters News), which makes them hard to distinguish.

In MLTC task, one document may contain multiple labels, and each label can be taken as one aspect or component of the document, thus, the overall semantics of the whole document can be formed by multiple components. Motivated by the above-mentioned observations, we propose a novel **L**abel-**S**pecific **A**ttention **N**etwork model (**LSAN**) to learn document representation by sufficiently exploiting the document content and label content. To capture the label-related component from each document, we adopt the self-attention mechanism to measure the contribution of each word to each label. Meanwhile, **LSAN** takes advantage of label texts to embed each label into a vector like word embedding, so that the semantic relations between document words and labels can be explicitly computed. Thereafter, an adaptive fusion strategy is designed to extract the proper amount of information from these two aspects and construct the label-specific representation for each document. We summarize the main contributions:

- A label-specific attention network model is proposed to handle multi-label text classification task by considering document content and label texts simultaneously.
- An adaptive fusion strategy is first designed to adaptively extract the proper semantical information to construct label-specific document representation.
- The performance of **LSAN** is thoroughly investigated on four widely-used benchmark datasets in terms of several evaluation metrics, indicating its advantage over the state-of-the-art baselines.

The rest of the paper is organized as follows. Section 2 describes the proposed **LSAN** model for multi-label text classification. The experiments on real-word datasets are conducted in Section 3 and their results are discussed in detail. Section 4 lists the related work. The brief conclusions and future work are given in Section 5.

## 2 Proposed Method

In this section, we introduce the proposed label-specific attention network, as shown in Figure 1. **LSAN** consists of two main parts. The first part is to capture the label-related components from each document by exploiting both document content and label texts. The second part aims to adaptively extract the proper information from two aspects. Finally, the classification model can be trained on the fused label-specific document representations.

### 2.1 Preliminaries

**Problem Definition:** Let $D = \{(x_i, y_i)\}_{i=1}^{N}$ denote the set of documents, which consists of N documents with corresponding labels $Y = \{y_i \in \{0,1\}^l\}$, here $l$ is the total number of labels. Each document contains a sequence of words. Each word can be encoded to a low-dimensional space and represented as a $d$-dimension vector via word2vector technique (Pennington et al., 2014). Let $x_i = \{w_1, \cdots, w_p, \cdots, w_n\}$ denote the $i$-th document, $w_p \in \mathbb{R}^k$ is the $p$-th word vector in the document, $n$ is the number of words in document.

For text classification, each label contains textual information. Thus, similar to the document word, one label can be represented as an embedding vector and the label set will be encoded by a trainable matrix $C \in \mathbb{R}^{l \times k}$. Given the input documents and their associated labels $D$, MLTC aims to train a classifier to assign the most relevant labels to the new coming documents.

**Input Text Representation:** To capture the forward and backward sides contextual information of each word, we adopt the bidirectional long short-term memory (Bi-LSTM) (Zhou et al., 2016) language model to learn the word embedding for each input document. At time-step $p$, the hidden state can be updated with the aid of input and $(p-1)$-th step output.

$$\begin{aligned} \overrightarrow{h_p} &= LSTM(\overrightarrow{h_{p-1}}, w_p) \\ \overleftarrow{h_p} &= LSTM(\overleftarrow{h_{p-1}}, w_p) \end{aligned} \tag{1}$$

where $w_p$ is the embedding vector of the $p$-th word in the corresponding document, and $\overrightarrow{h_p}, \overleftarrow{h_p} \in \mathbb{R}^k$ indicate the forward and backward word context representations respectively. Then, the whole document can be represented by Bi-LSTM as follows.

$$\begin{aligned} H &= (\overrightarrow{H}, \overleftarrow{H}) \\ \overrightarrow{H} &= (\overrightarrow{h_1}, \overrightarrow{h_2}, \cdots, \overrightarrow{h_n}) \\ \overleftarrow{H} &= (\overleftarrow{h_1}, \overleftarrow{h_2}, \cdots, \overleftarrow{h_n}) \end{aligned} \tag{2}$$

In this case, the whole document set can be taken as a matrix $H \in \mathbb{R}^{2k \times n}$.

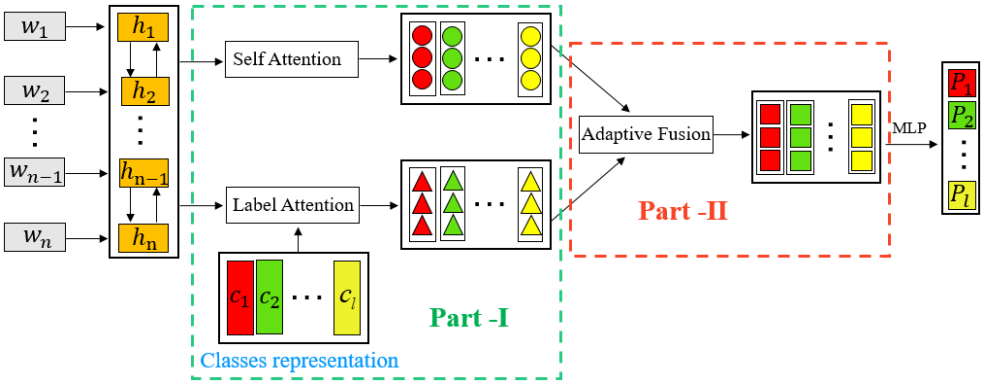


Figure 1: The architecture of the proposed label-specific attention network model (LSAN).

### 2.2 Label-Specific Attention Network

In this subsection, we will give the proposed attention network model for label-specific document representation learning. It aims to determin the label-related component from each document. Actually, this strategy is intuitive for text classification. For example, regarding the text *"June a friday, in the lawn, a war between the young boys of the football game starte"*, it is assigned into two categories *youth* and *sports*. Obviously, the content *"young boy"* is much more related to *youth* than to *sports*, while *"football game"* should be directly related to *sports*. Next, we will show how to capture this characteristic with our model.

#### 2.2.1 Self-attention Mechanism

As mentioned above, the multi-label document may be tagged by more than one label, and each document should have the most relative contexts with its corresponding labels. In other words, each document may contain multiple components, and the words in one document make different contributions to each label. To capture different components for each label, we adopt the self-attention mechanism (Lin et al., 2017), which has been successful used in various text mining tasks (Tan et al., 2018; Al-Sabahi et al., 2018; You et al., 2018). The label-word attention score ($A^s \in \mathbb{R}^{l \times n}$) can be obtained by

$$A^{(s)} = softmax(W_2 tanh(W_1 H)) \tag{3}$$

where $W_1 \in \mathbb{R}^{d_a \times 2k}$ and $W_2 \in \mathbb{R}^{l \times d_a}$ are the so-called self-attention parameters to be trained. $d_a$ is a hyper-parameter we can set arbitrarily. Each row $A^{(s)}_{j\cdot}$ (an $n$-dim row vector where $n$ is the total number of words) indicates the contribution of all words to the $j$-th label. Then, we can obtain the the linear combination of the context words for each label with the aid of label-word attention score ($A^{(s)}$) as follows.

$$M^{(s)}_{j\cdot} = A^{(s)}_{j} H^T \tag{4}$$

which can be taken as a new representation of the input document along the $j$-th label. Then the whole matrix $M^{(s)} \in \mathbb{R}^{l \times 2k}$ is the label-specific document representation under the self-attention mechanism.

#### 2.2.2 Label-Attention Mechanism

Self-attention mechanism can be taken as content-based attention because it only considers the document content. As we all know, labels have specific semantics in text classification, which is hidden in the label texts or descriptions. To make use of the semantic information of labels, they are preprocessed and represented as a trainable matrix $C \in \mathbb{R}^{l \times k}$ in the same latent $k$-dim space with the words.

Once having the word embedding from Bi-LSTM in (1) and the label embedding in $C$, we can explicitly determine the semantic relation between each pair of word and label. A simple way is calculating the dot product between $\overrightarrow{h}_p$ (or $\overleftarrow{h}_p$) and $C_{j\cdot}$ as follows.

$$\begin{aligned} \overrightarrow{A}^{(l)} &= C\overrightarrow{H} \\ \overleftarrow{A}^{(l)} &= C\overleftarrow{H} \end{aligned} \tag{5}$$

where $\overrightarrow{A}^{(l)} \in \mathbb{R}^{l \times n}$ and $\overleftarrow{A}^{(l)} \in \mathbb{R}^{l \times n}$ indicate the forward and backward sides semantic relation between words and labels. Similar to the previous self-attention mechanism, the label-specific document representation can be constructed by linear combining the label's context words as follows.

$$\begin{aligned} \overrightarrow{M}^{(l)} &= \overrightarrow{A}^{(l)} \overrightarrow{H}^T \\ \overleftarrow{M}^{(l)} &= \overleftarrow{A}^{(l)} \overleftarrow{H}^T \end{aligned} \tag{6}$$

Finally, the document can be re-represented along all labels via $M^{(l)} = (\overrightarrow{M}^{(l)}, \overleftarrow{M}^{(l)}) \in \mathbb{R}^{l \times 2k}$. This representation is based on the label texts, thus, we called it as label-attention mechanism.

### 2.3 Adaptive Attention Fusion Strategy

Both $M^{(s)}$ and $M^{(l)}$ are label-specific document representation, but they are different. The former focuses on the document content, while the later prefers to the semantic correlation between document content and label text. In order to take advantage these two parts, in this subsection, an attention fusion strategy is proposed to adaptively extract proper amount of information from them and build comprehensive label-specific document representation.

More specifically, two weight vectors ($\alpha, \beta \in \mathbb{R}^l$) are introduced to determine the importances of the above two mechanisms, which can obtained by a fully connected layer on the input $M^{(s)}$ and $M^{(l)}$.

$$\begin{aligned} \alpha &= sigmoid(M^{(s)} W_3) \\ \beta &= sigmoid(M^{(l)} W_4) \end{aligned} \tag{7}$$

$W_3, W_4 \in \mathbb{R}^{2k}$ are the parameters to be trained. $\alpha_j$ and $\beta_j$ indicate the importances of self-attention and label-attention to construct the final document representation along the $j$-th label respectively. Therefore, we add the constraint on them:

$$\alpha_j + \beta_j = 1 \tag{8}$$

Then, we can obtain the final document representation along the $j$-th label based on fusion weights as follows.

$$M_{j\cdot} = \alpha_j M^{(s)}_{j\cdot} + \beta_j M^{(l)}_{j\cdot} \tag{9}$$

The label-specific document representation along all labels can be described as a matrix $M \in \mathbb{R}^{l \times 2k}$.

### 2.4 Label Prediction

Once having the comprehensive label-specific document representation, we can build the multi-label text classifier via a multilayer perceptron with two fully connected layers. Mathematically, the predicted probability of each label for the coming document can be estimated via

$$\hat{y} = sigmoid(W_6 f(W_5 M^T)) \tag{10}$$

Here $W_5 \in \mathbb{R}^{b \times 2k}, W_6 \in \mathbb{R}^b$ are the trainable parameters of the fully connected layer and output layer respectively. $f$ is the ReLU nonlinear activation function. The sigmoid function is used to transfer the output value into a probability, in this case, the cross-entropy loss can be used as the loss function which has been proved suitable for multi-label text classification task (Nam et al., 2014).

$$\begin{aligned} \mathcal{L} = -\sum_{i=1}^{N}\sum_{j=1}^{l} &(y_{ij} \log(\hat{y}_{ij})) \\ &+ (1 - y_{ij}) \log(1 - \hat{y}_{ij}) \end{aligned} \tag{11}$$

where $N$ is the number of training documents, $l$ is the number of labels, $\hat{y}_{ij} \in [0, 1]$ is the predicted probability, and $y_{ij} \in \{0, 1\}$ indicates the ground truth of the $i$-th document along the $j$-th label.

## 3 Experiments

In this section, we evaluate the proposed model on four datasets (with various number of labels from 54 to 3956) by comparing with the state-of-the-art methods in terms of widely used metrics.

### 3.1 Experimental Setting

**Datasets:** There are several multi-label text datasets, however, only few of them have label text information. Thus, in this paper, four benchmark multi-label datasets including three small-scale datasets (RCV1, AAPD and EUR-Lex) and one medium-scale dataset (KanShan-Cup) are used to construct the experiments.

- **Reuters Corpus Volume I (RCV1)** (Lewis et al., 2004) contains more than 80K manually categorized news belonging to 103 classes.
- **AAPD**[2] (Yang et al., 2018) collects the abstract and the corresponding subjects of 55840 papers from arXiv in the filed of computer science.
- **EUR-Lex** (Mencia and Fürnkranz, 2008) is a collection of documents about European Union law belonging to 3956 subjects. The public version[3] contains 11585 training instances and 3865 testing instances.
- **KanShan-Cup**[4] is released by the largest Chinese community question answering platform, Zhihu. It contains near 3 million questions about 1999 topics.

For the first three data sets, only last 500 words were kept for each document, while the last 50

[2] https://github.com/lancopku/SGM

[3] https://drive.google.com/drive/folders/1KQMBZgACUm-ZZcSrQpDPlB6CFKvf9Gfb

[4] https://www.biendata.com/competition/zhihu/data/

Table 1: Summary of Experimental Datasets.

| Datasets | $N$ | $M$ | $D$ | $L$ | $\bar{L}$ | $\tilde{L}$ | $\bar{W}$ | $\tilde{W}$ |
|---|---|---|---|---|---|---|---|---|
| RCV1 | 23,149 | 781,265 | 47,236 | 103 | 3.18 | 729.67 | 259.47 | 269.23 |
| AAPD | 54,840 | 1,000 | 69,399 | 54 | 2.41 | 2444.04 | 163.42 | 171.65 |
| EUR-Lex | 11,585 | 3,865 | 171,120 | 3,956 | 5.32 | 15.59 | 1,225.20 | 1,248.07 |
| Kanshan-Cup | 2,799,967 | 200,000 | 411,721 | 1999 | 2.34 | 3513.13 | 38.06 | 35.48 |

$N$ is the number of training instances, $M$ is the number of test instances, $D$ is the total number of words, $L$ is the total number of classes, $\bar{L}$ is the average number of labels per document, $\tilde{L}$ is the average number of documents per label, $\bar{W}$ is the average number of words per document in the training set, $\tilde{W}$ is the average number of words per document in the testing set.

words were used in KanShan-Cup dataset. Once the document has less than the predifined number of words, we extend it by padding zeros. All methods are trained and tested on the given training and testing datasets which are summarized in Table 1.

**Evaluation Metrics:** We use two kinds of metric, precision at top K ($P@k$) and the Normalized Discounted Cumulated Gains at top $K$ ($nDCG@k$) to evaluate the prediction performance. $P@k$ and $nDCG@k$ are defined according to the predicted score vector $\hat{y} \in \mathbb{R}^l$ and the ground truth label vector $y \in \{0,1\}^l$ as follows.

$$P@k = \frac{1}{k} \sum_{l \in rank_k(\hat{y})} y_l$$

$$DCG@k = \sum_{l \in rank_k(\hat{y})} \frac{y_l}{\log(l+1)}$$

$$nDCG@k = \frac{DCG@k}{\sum_{l=1}^{\min(k, \|y\|_0)} \frac{1}{\log(l+1)}}$$

where $rank_k(y)$ is the label indexes of the top $k$ highest scores of the current prediction result. $\|y\|_0$ counts the number of relevant labels in the ground truth label vector $y$.

**Baseline Models:** The proposed LSAN is a deep neural network model, thus the recent state-of-the-art deep learning-based MLTC methods are selected as baselines.

- **XML-CNN:** (Liu et al., 2017) adopts Convolutional Neural Network (CNN) and a dynamic pooling technique to extract high-level feature for multi-label text classification.
- **SGM:** (Yang et al., 2018) applies a sequence generation model from input document to output label to construct the multi-label text classifier.
- **DXML:** (Zhang et al., 2018) tries to explore the label correlation by considering the label structure from the label co-occurrence graph.
- **AttentionXML:** (You et al., 2018) builds the label-aware document representation only based on the document content, thus, it can be taken as one special case of our proposed **LSAN** with arbitrarily setting $\alpha = 0$.
- **EXAM:** (Du et al., 2018) is the most similar work to **LSAN** because both of them exploit the label text to learn the interaction between words and labels. However, EXAM suffers from the situation where different labels have similar text.

**Parameter Setting:** For the KanShan-Cup dataset, we use the pre-trained word embedding and label embedding public in the official website, where the embedding space size is 256, i.e., $k = 256$. The parameters corresponding to the weights between neurals are $d_a = 200$ for $W_1$ and $W_2$, $b = 256$ for $W_5$ and $W_6$. For other three datasets, $k = 300$, $d_a = 200$ and $b = 300$. The whole model is trained via Adam (Kingma and Ba, 2014) with the learning rate being 0.001. The parameters of all baselines are either adopted from their original papers or determined by experiments.

### 3.2 Comparison Results and Discussion

In this section, the proposed LSAN is evaluated on four benchmark datasets by comparing with five baselines in terms of $P@K$ and $nDCG@K(K = 1, 3, 5)$. Table 2 and Table 3 show the averaged performance of all test documents. According to the formula of $P@K$ and $nDCG@K$, we know $P@1 = nDCG@1$, thus only $nDCG@3$ and $nDCG@5$ are listed in Table 3. In each line, the best result is marked in bold.

From Table 2 and 3, we can make a number of observations about these results. Firstly, XML-CNN is worse than other four methods because it only considers the document content but ignores the label correlation which has been proven very important for multi-label learning. Secondly, AttentionXML is superior to EXAM on datasets RCV1 and Kanshan-Cup, because these two datases have hierarchical label structures. In this case, parent label and child label may contain similar text, which makes them hard to distinguish according to the text-based embedding and further reduce the performance of EXAM. By compar-

| Datasets | Metrics | XML-CNN | DXML | SGM | AttentionXML | EXAM | LSAN(ours) |
|---|---|---|---|---|---|---|---|
| RCV1 | P@1 | 95.75% | 94.04% | 95.37% | 96.41% | 93.67% | **96.81%** |
| | P@3 | 78.63% | 78.65% | 81.36% | 80.91% | 75.80% | **81.89%** |
| | P@5 | 54.94% | 54.38% | 53.06% | 56.38% | 52.73% | **56.92%** |
| AAPD | P@1 | 74.38% | 80.54% | 75.67% | 83.02% | 83.26% | **85.28%** |
| | P@3 | 53.84% | 56.30% | 56.75% | 58.72% | 59.77% | **61.12%** |
| | P@5 | 37.79% | 39.16% | 35.65% | 40.56% | 40.66% | **41.84%** |
| EUR-Lex | P@1 | 70.40% | 75.53% | 70.45% | 67.34% | 74.40% | **79.17%** |
| | P@3 | 54.98% | 60.13% | 60.37% | 52.52% | 61.93% | **64.99%** |
| | P@5 | 44.86% | 48.65% | 43.88% | 47.72% | 50.98% | **53.67%** |
| Kanshan-Cup | P@1 | 49.68% | 50.84% | 50.32% | 53.69% | 51.41% | **54.46%** |
| | P@3 | 32.27% | 32.69% | 31.83% | 34.10% | 32.81% | **34.56%** |
| | P@5 | 24.17% | 24.07% | 23.95% | 25.16% | 24.29% | **25.54%** |

Table 2: Comparing LSAN with five baselines in terms of $P@K$ (K=1,3,5)on four benchmark datasets.

| Datasets | Metrics | XML-CNN | DXML | SGM | AttentionXML | EXAM | LSAN(ours) |
|---|---|---|---|---|---|---|---|
| RCV1 | nDCG@3 | 89.89% | 89.83% | 91.76% | 91.88% | 86.85% | **92.83%** |
| | nDCG@5 | 90.77% | 90.21% | 90.69% | 92.70% | 87.71% | **93.43%** |
| AAPD | nDCG@3 | 71.12% | 77.23% | 72.36% | 78.01% | 79.10% | **80.84%** |
| | nDCG@5 | 75.93% | 80.99% | 75.35% | 82.31% | 82.79% | **84.78%** |
| EUR-Lex | nDCG@3 | 58.62% | 63.96% | 60.72% | 56.21% | 65.12% | **68.32%** |
| | nDCG@5 | 53.10% | 57.60% | 55.24% | 50.78% | 59.43% | **62.47%** |
| Kanshan-Cup | nDCG@3 | 46.65% | 49.54% | 46.90% | 51.03% | 49.32% | **51.43%** |
| | nDCG@5 | 49.60% | 52.16% | 50.47% | 53.96% | 49.74% | **54.36%** |

Table 3: Comparing LSAN with five baselines in terms of $nDCG@K$ (K=3,5) on four benchmark datasets.

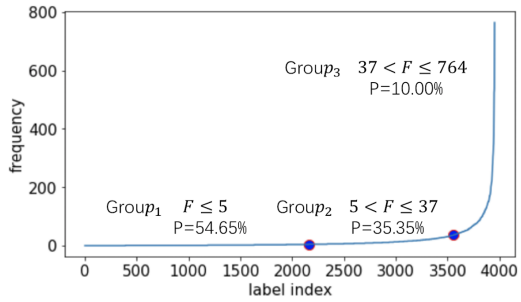


Figure 2: The label distribution of EUR-Lex

ing with EXAM and the proposed LSAN, however, AttentionXML performs worse on EUR-Lex dataset The main reason is that AttentionXML only focuses on the document content, which will make it not sufficiently trained once there are only few documents in some labels. Fortunately, EXAM and LSAN benefit from the label texts. Last one, as expected, is that LSAN consistently outperforms all baselines on all experimental datasets. This result further confirms that the proposed adaptive attention fusion strategy is much helpful to learn the label-specific document representation for multi-label text classification.

### 3.3 Comparison on Sparse Data

In order to verify the performance of LSAN on low-frequency labels, we divided labels in EUR-Lex into three groups according to their occurring frequency. Figure 2 shows the distribution of label frequency on EUR-Lex, $F$ is the frequency of label. Among it, nearly 55% of labels occur between 1 and 5 times to form the first label group (Group1). The labels appearing 5-37 times are assigned into Group2, which is 35% of the whole label set. The remaining 10% frequent labels form the last group (Group3). Obviously, Group1 is much harder than other two groups due to the lack of training documents.

Figure 3 shows the prediction results in terms of $P@1$, $P@3$ and $P@5$ obtained by AttentionXML, EXAM and LSAN. Three methods become better and better from Group1 to Group3, which is reasonable because more and more documents are included in each label from Group1 to Group3. LSAN significantly improves the prediction performance on Group1. Especially, LSAN obtains an average of more than 83.82%, 182.55%, 244.62% gain on three metrices for group 1 to AttentionXML, and 3.85%, 27.19%, 58.27% gain to EXAM. This result demonstrates the superiority of the proposed model on multi-label text data with tail labels.

### 3.4 Ablation Test

The proposed LSAN can be taken as a joint attention strategy including three parts. One is self-attention based on document content (denoted as A). The second one is label-attention based on label text (denoted as L). Another one is fusion attention by adaptively integrating A and L with proper weights (denoted as W). In this section, we

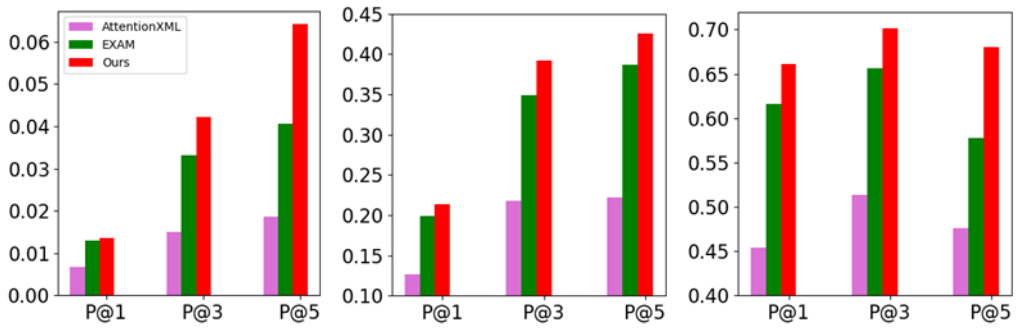


(a) Gropu1 ($F \leq 5$) (b) Gropu2 ($5 < F \leq 37$) (c) Gropu3 ($F > 37$)

Figure 3: Precision@k for three groups on EUR-Lex

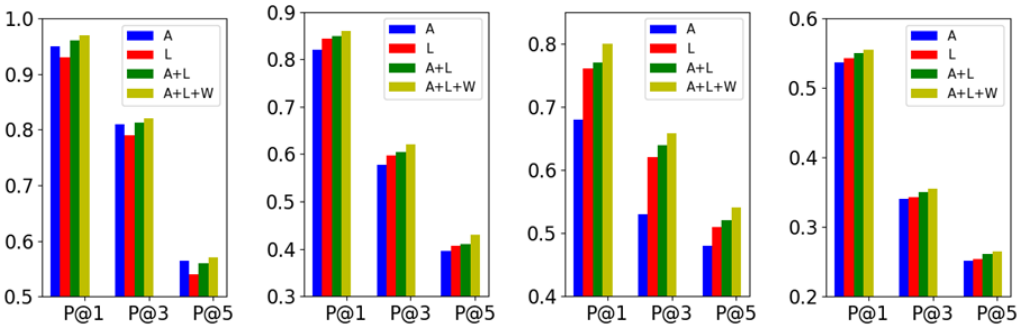


(a) RCV1 (b) AAPD (c) EUR-Lex (d) Kanshan-Cup

Figure 4: Result of the ablation test. 'A' denotes the Self-Attention, 'L' denotes the Label-Attention, 'W' denotes the Fusion Attention with Adaptive Weights.

try to demonstrate the effection of each component via an ablation test.

Figure 4 lists the prediction results on four datasets in terms of $P@1, P@3$ and $P@5$. (i.e., S+L gets better results than S and L). S prefers to finding the useful content when constructing the label-specific document representation, but it ignores the label information. L takes adantage of label text to explicitly determine the semantic relation between documents and labels, however, label text is not easy to distinguish the difference between labels (e.g., *Management* vs. *Management movies*). Therefore, coupling with both attentions is really reasonable. Furthermore, adaptively extracting proper amount of information from these two attentions benefits the final multi-label text classification.

To further verify the effectiveness of attention adaptive fusion, Figure 5 lists the distribution of weights on Self-attention and Label-attention on two representative datasets, one for sparse data (EUR-Lex) and the other for dense data (AAPD). As expected, the label-attention is much more useful than self-attention for sparse data, vice versa for dense data. In dense data, each label has sufficient documents, therefore, self-attention can sufficiently obtain label-specific document representation. On the other hand, label text is helpful to extract the semantic relations between labels and documents. Results on other two datasets have the similar trends which are omitted due to page limitation.

For investigating the effect of label-attention, we visualize the attention weights on the original document using heat map, as shown in Figure 6. Among it, the example AAPD document belongs to two categories *Computer Vision* and *Neural and Evolutionary Computing*. From the attention weights, we can see that each category has its own most related words, which confirms that the proposed label specific attention network is able to extract the label-aware content and further construct label-specific document representation.

## 4 Related work

In the line of MLTC, most works focus on two issues, one is document representation learning and

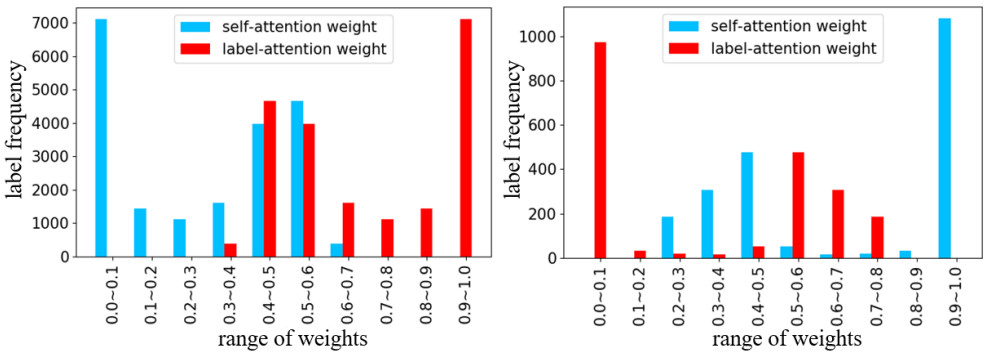


Figure 5: Weight distribution for two components on EUR-Lex (left subfigure) and AAPD (right subfigure). Horizontal axis is the range of weight from 0 to 1 with 0.1 gap. Vertical axis is the frequency that the specific range occurs in current label group.

Various hand crafted features and metric learning methods prevail in the field of person reidentification compared to these methods , this paper proposes a more general way that can learn a similarity metric from image pixels directly by using a siamese deep neural network , the proposed method can jointly learn the color feature , texture feature and metricin a unified framework the network has a symmetry structure with two sub networks which are connected by cosine function to deal with the big variations of person images

(a) Visual analysis the label "*Computer Vision*"

Various hand crafted features and metric learning methods prevail in the field of person reidentification compared to these methods , this paper proposes a more general way that can learn a similarity metric from image pixels directly by using a siamese deep neural network , the proposed method can jointly learn the color feature , texture feature and metricin a unified framework the network has a symmetry structure with two sub networks which are connected by cosine function to deal with the big variations of person images

(b) Visual analysis the label "*Neural and Evolutionary Computing*"

Figure 6: Demonstration of words with largest label-attention weights ($A^{(l)}$) in one AAPD document belonging to two categories:*Computer Vision and Neural and Evolutionary Computing*.

the other is label correlation detection.

For document representation, along with the recent success in CNN, many works are proposed based on CNN (Kim, 2014; Liu et al., 2017; Chen et al., 2017), which can capture the local correlations from the consecutive context windows. Although they obain promising results, these methods suffer from the limitation of window size so that they cannot determine the long-distance dependency of text. Meanwhile, they treat all words equally no matter how noisy the word is. Later, RNN and attention mechanism are introduced to get brilliant results (Yang et al., 2016). To implicitly learn the document representation for each label, the self-attention mechanism (Lin et al., 2017) is adopted for multi-label classification (You et al., 2018).

To determine the label correlation among multi-label data, in literatures, researchers proposed various methods. Kurata et al. (2016) adopt an initialization method to leverage label co-occurrence information. SLEEC (Bhatia et al., 2015) divides dataset into several clusters, and in each cluster it detects embedding vectors by capturing non-linear label correlation. DXML (Zhang et al., 2018) establishes an explicit label co-occurrence graph to explore label embedding in low-dimension latent space. Yang et al. (2018) use sequence-to-sequence(Seq2Seq) model to consider the correlations between labels. Recently, the textual information of labels are used to guide MLTC. EXAM (Du et al., 2018) introduces the interaction mechanism to incorporate word-level matching signals into the text classification task. GILE (Pappas and Henderson, 2019) proposes a joint input-label embedding model for neural text classification. Unfortunately, they cannot work well when there is no big difference between label texts.

## 5 Conclusions and Future Work

A new label-specific attention network, in this paper, is proposed for multi-label text classification. It makes use of document content and label text to learn the label-specific document representation with the aid of self-attention and label-attention mechanisms. An adaptive fusion is designed to effectively integrate these two attention mechanisms to improve the final prediction performance. Extensive experiments on four benchmark datasets prove the superiority of LSAN by comparing with the state-of-the-art methods, especially on the dataset with large subset of low-frequency labels.

In real applications, more precious information can be collected, such as label description, la-

bel topology (e.g., hierarchical structure) and etc. Therefore, it is interesting to extend the current model with such extra information.

## Acknowledgments

We would like to thank the outstanding anonymous reviewers for their helpful comments to improve our manuscript. This work was supported in part by the National Natural Science Foundation of China under Grant 61822601, 61773050, and 61632004; the Beijing Natural Science Foundation under Grant Z180006; the Beijing Municipal Science & Technology Commission under Grant Z181100008918012; National Key Research and Development Program (2017YFC1703506).

## References

Ashutosh Adhikari, Achyudh Ram, Raphael Tang, and Jimmy Lin. 2019. Docbert: Bert for document classification. *arXiv preprint arXiv:1904.08398*.

Kamal Al-Sabahi, Zhang Zuping, and Mohammed Nadher. 2018. A hierarchical structured self-attentive model for extractive document summarization (hssas). *IEEE Access*, 6:24205–24212.

Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. 2015. Sparse local embeddings for extreme multi-label classification. In *Advances in neural information processing systems*, pages 730–738.

Erik Cambria, Daniel Olsher, and Dheeraj Rajagopal. 2014. Senticnet 3: a common and common-sense knowledge base for cognition-driven sentiment analysis. In *Twenty-eighth AAAI conference on artificial intelligence*.

Guibin Chen, Deheng Ye, Zhenchang Xing, Jieshan Chen, and Erik Cambria. 2017. Ensemble application of convolutional and recurrent neural networks for multi-label text categorization. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2377–2383. IEEE.

Cunxiao Du, Zhaozheng Chin, Fuli Feng, Lei Zhu, Tian Gan, and Liqiang Nie. 2018. Explicit interaction model towards text classification. *arXiv preprint arXiv:1811.09386*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *International Conference on Machine Learning*, pages 1378–1387.

Gakuto Kurata, Bing Xiang, and Bowen Zhou. 2016. Improved neural network-based multi-label classification with better initialization leveraging label co-occurrence. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 521–526.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Twenty-ninth AAAI conference on artificial intelligence*.

David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. 2004. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. 2017. Deep learning for extreme multi-label text classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 115–124. ACM.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101*.

Eneldo Loza Mencia and Johannes Fürnkranz. 2008. Efficient pairwise multilabel classification for large-scale problems in the legal domain. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 50–65. Springer.

Jinseok Nam, Jungi Kim, Eneldo Loza Mencía, Iryna Gurevych, and Johannes Fürnkranz. 2014. Large-scale multi-label text classificationłrevisiting neural networks. In *Joint european conference on machine learning and knowledge discovery in databases*, pages 437–452. Springer.

Nikolaos Pappas and James Henderson. 2019. Gile: A generalized input-label embedding for text classification. *Transactions of the Association for Computational Linguistics (TACL)*, 7.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2018. Deep semantic role labeling with self-attention. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Pengcheng Yang, Xu Sun, Wei Li, Shuming Ma, Wei Wu, and Houfeng Wang. 2018. Sgm: sequence generation model for multi-label classification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3915–3926.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Ronghui You, Suyang Dai, Zihan Zhang, Hiroshi Mamitsuka, and Shanfeng Zhu. 2018. Attentionxml: Extreme multi-label text classification with multi-label attention based recurrent neural networks. *arXiv preprint arXiv:1811.01727*.

Wenjie Zhang, Junchi Yan, Xiangfeng Wang, and Hongyuan Zha. 2018. Deep extreme multi-label learning. In *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval*, pages 100–107. ACM.

Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. 2016. Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. *arXiv preprint arXiv:1611.06639*.

# Hierarchical Attention Prototypical Networks for Few-Shot Text Classification

**Shengli Sun**[1*] **Qingfeng Sun**[1*] **Kevin Zhou**[2 †] **Tengchao Lv**[1]
[1]Peking University [2]Microsoft
slsun@ss.pku.edu.cn {sunqingfeng, lvtengchao}@pku.edu.cn
kezhou@microsoft.com

## Abstract

Most of the current effective methods for text classification task are based on large-scale labeled data and a great number of parameters, but when the supervised training data are few and difficult to be collected, these models are not available. In this paper, we propose a hierarchical attention prototypical networks (HAPN) for few-shot text classification. We design the feature level, word level, and instance level multi cross attention for our model to enhance the expressive ability of semantic space. We verify the effectiveness of our model on two standard benchmark few-shot text classification datasets – FewRel and CSID, and achieve the state-of-the-art performance. The visualization of hierarchical attention layers illustrates that our model can capture more important features, words, and instances separately. In addition, our attention mechanism increases support set augmentability and accelerates convergence speed in the training stage.

## 1 Introduction

The dominant text classification models in deep learning (Kim, 2014; Zhang et al., 2015a; Yang et al., 2016; Wang et al., 2018) require a considerable amount of labeled data to learn a large number of parameters. However, such methods may have difficulty in learning the semantic space in the case that only few data are available. Few-shot learning has became an effective approach to solve this challenge, it can train a neural network with a few parameters using few data but achieve good performance. A typical example of this approach is prototypical networks (Snell et al., 2017), which averages the vector of few support instances as the class prototype and computes distance between target query and each prototype, then classify the query to the nearest prototype's class. However, prototypical networks is rough and does not consider the adverse effects of various noises in the data, which weakens the discrimination and expressiveness of the prototype.

In this paper, we propose a hierarchical attention prototypical networks for few-shot text classification by using attention mechanism in three levels. For feature level attention, we use convolutional neural networks to get the feature scores which is different for various classes. For word level attention, we adopt an attention mechanism to learn the importance of each word hidden state in an instance. For instance level multi cross attention, with the help of multi cross attention between support set and target query, we can determine the importance of different instances in the same class and enable the model to get a more discriminative prototype of each class.

In the actual scenario, we apply HAPN on intention detection of our open domain chatbots with different character. If we create a chatbot for old people, the user intentions will focus on children, health or expectation, so we can define specific intentions and supply related responses. And because of only few data are needed, we can expand the number of classes quickly. The model helps chatbot to identify user intentions precisely, makes the dialogue process smoother, more knowledgeable and more controllable.

There are three main parts of our contribution: first of all, we propose a hierarchical attention prototypical networks for few-shot text classification, then we achieve state-of-the-art performance on FewRel and CSID datasets, and the experiments prove our model is faster and more extensible.

## 2 Related Works

### 2.1 Text Classification

Text Classification is an important task in Natural Language Processing, and many models are

[*] They contribute equally to this work.
[†] Corresponding author.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 476–485, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

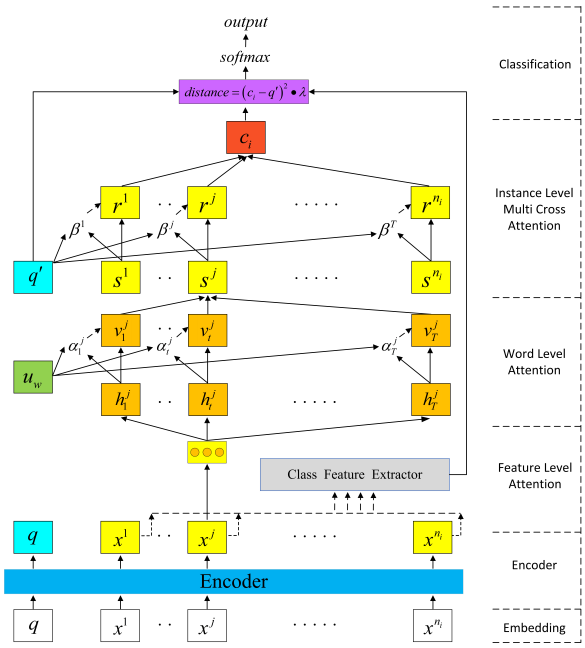


Figure 1: Hierarchical Attention Prototypical Networks architecture

proposed to solve it. The traditional methods mainly focus on feature engineerings such as bag-of-words or n-grams (Wang and Manning, 2012) or SVMs (Tang et al., 2015). The neural network based methods like Kim (2014) applies convolutional neural networks for sentence classification. Then, Johnson and Zhang (2015) use a one-hot word order CNN, and Zhang et al. (2015b) apply a character level CNN. C-LSTM (Zhou et al., 2015) combines CNN and RNN for sentence representation and text classification. Yang et al. (2016) explore the hierarchical structure of documents classification, they use a GRU-based attention to build representations of sentences and another GRU-based attention to aggregate them into a document representation. But above supervised learning methods require large-scale labeled data and can't classify unseen classes.

## 2.2 Few-Shot Learning

Few-Shot Learning (FSL) aims to solve classification problems by training a classifier with few instances in each class, and it can apply to unseen classes. The early works aim to use transfer learning approaches, Caruana (1994) and Bengio (2011) adopt the target task from the pre-trained models. Then Koch et al. (2015) explore a method for learning siamese neural networks which employs an unique structure to rank similarity between inputs. Vinyals et al. (2016) use matching networks to map a small labeled support set and an unlabelled example to its label, and obviate the need for fine-tuning to adapt to new class types. Prototypical networks (Snell et al., 2017) learns a metric space in which the model can perform well by computing distance between query and prototype representations of each class and classify the query to the nearest prototype's class. Sung et al. (2018) propose a two-branch relation networks, which learns to compare query against few-shot labeled sample support data. Dual TriNet structure (Chen et al., 2018) can efficiently and directly augment multi-layer visual features to boost the few-shot classification.But all of the above works mainly concentrate on computer vision field, the research and applications in NLP field are extremely limited. Recently, Yu et al. (2018) propose an adaptive metric learning approach that automatically determines the best weighted combination from a set of metrics obtained from meta-training tasks for a newly seen few-shot task such as intention classification, Han et al. (2018) present a relation classification dataset – FewRel, and adapt most recent state-of-the-art few-shot learning methods for it, Gao et al. (2019) propose a hybrid attention-based prototypical networks for noisy few-shot relation classification. However, these methods do not consider mining semantic information or reducing the impact of noise more precisely. And in most of the realistic settings, we may increase the number of instances gradually, so model capacity needs more attention.

## 3 Task Definition

In few-shot text classification task, our goal is to learn a function : $G(D, S, x) \rightarrow y$. $D$ is the labeled data, we divide $D$ into three parts: $D_{train}$, $D_{validation}$, and $D_{test}$, and each part has specific label space. We use $D_{train}$ to optimize parameters, $D_{validation}$ to select best hyper parameters, and $D_{test}$ to evaluate the model.

The "episode" training strategy that Vinyals et al. (2016) proposed has proved to be effective. For each training episode, we first sample a label set $\mathcal{L}$ from $D_{train}$, then use $\mathcal{L}$ to sample the support set $S$ and the query set $Q$, finally, we feed $S$ and $Q$ to the model and minimize the loss. If $\mathcal{L}$ includes $N$ different classes and each class of $S$ contains $K$ instances, we call the target problem $N$-way $K$-shot learning. For this paper, we consider $N$ = 5 or 10, and $K$ = 5 or 10.

For exactly, in an episode, we are given a support set $S$

$$\begin{aligned} S = &\{(x_1^1, l_1), (x_1^2, l_1), \ldots, (x_1^{n_1}, l_1), \\ &\cdots, \\ &(x_m^1, l_m), (x_m^2, l_m), \ldots, (x_m^{n_m}, l_m)\}, \\ &l_1, l_2, \cdots, l_m \in \mathcal{L} \end{aligned} \tag{1}$$

consists of $n_i$ text instances for each class $l_i \in \mathcal{L}$, $x_i^j$ means it is the $j$ support instance belonging to calss $l_i$, and instance $x_i^j$ includes $T_{i,j}$ words $\{w_1, w_2, \ldots, w_{T_{i,j}}\}$.

Then $x$ is an unlabeled instance of query set $Q$ to classify, and $y \in \mathcal{L}$ is the output label followed by the prediction of $G$.

## 4 Method

### 4.1 Model Overview

The overall architecture of the Hierarchical Attention Prototypical Networks is shown in Figure 1. We introduce different components in the following subsections:

**Instance Encoder** Each instance in support set or query set will be first represented to a input vector by transforming each word into embeddings. Considering the lightweight and speed of the model, we achieve this part with one layer convolutional neural networks (CNN). For ease of comparison, its details are the same as Han et al. (2018) proposed.

**Hierarchical Attention** In order to get more important information from rare data, we adopt a hierarchical attention mechanism. Feature level attention enhances or reduces the importance of different feature in each class, word level attention highlight the important words for meaning of the instance, and instance level multi cross attention can extract the important support instances for different query instances, these three attention mechanisms work together to improve the classification performance of our model.

**Prototypical Networks** Prototypical networks compute a prototype vector as the representation of each class, and this vector is the mean vector of the embedded support instances belonging to its class. We compare the distance between all prototype vectors and a target query vector, then classify this query to the nearest one.

### 4.2 Instance Encoder

The instance encoder part consists of two layers: embedding layer and instance encoding layer.

#### 4.2.1 Embedding Layer

Given an instance $x = \{w_t, w_2, \ldots, w_T\}$ with $T$ words. We use an embedding matrix $W_E$,$\boldsymbol{w}_t = W_E w_t$ to embed each word to a vector

$$\{\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_T\}, \boldsymbol{w}_t \in \mathbb{R}^d \tag{2}$$

where $d$ is the word embedding dimension.

#### 4.2.2 Encoding Layer

Following we apply a convolutional neural network Zeng et al. (2014) as encoding layer to get the hidden annotations of each word by a convolution kernel with the window size $m$

$$\boldsymbol{h}_t = \text{CNN}(\boldsymbol{w}_{t-\frac{m-1}{2}}, \ldots, \boldsymbol{w}_{t-\frac{m+1}{2}}) \tag{3}$$

Especially, if the word $w_t$ has a position embedding $\boldsymbol{p}_t$, we should concat $\boldsymbol{w}_t$ and $\boldsymbol{p}_t$

$$\boldsymbol{wp}_t = [\boldsymbol{w}_t \oplus \boldsymbol{p}_t] \tag{4}$$

where $\oplus$ is a concatation, the $\boldsymbol{h}_t$ will be as follow

$$\boldsymbol{h}_t = \text{CNN}(\boldsymbol{wp}_{t-\frac{m-1}{2}}, \ldots, \boldsymbol{wp}_{t-\frac{m+1}{2}}) \tag{5}$$

Then, we aggregate all $\boldsymbol{h}_t$ to get the overall representation of instance $x$

$$\boldsymbol{x} = \{\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_t\} \tag{6}$$

Finally, we define those two layers as a comprehensive function

$$\boldsymbol{x} = g_\theta(x) \tag{7}$$

$\theta$ in this function are the networks parameters to be learned.

### 4.3 Prototypical Networks

The prototypical networks (Snell et al., 2017) has achieved excellent performance in few-shot image classification and few-shot text classification (Han et al., 2018; Gao et al., 2019) tasks respectively, so our model is based on prototypical networks and aims to get promotion.

The fundamental idea of prototypical networks is simple but efficient: we can use a prototype vector $\boldsymbol{c}_i$ as the representative feature of class $l_i$, each prototype vector can be calculated by averaging all the embedded instances in its support set

$$\boldsymbol{c}_i = \frac{1}{n_i} \sum_{j=1}^{n_i} g_\theta(x_i^j) \tag{8}$$

Then the probability distribution over the classes in $\mathcal{L}$ can be produced by a softmax function over distances between all prototypes vector and the target query $q$

$$p_\theta(y = l_i|q) = \frac{\exp(-d(g_\theta(q), \boldsymbol{c}_i)}{\Sigma_{l=1}^{|\mathcal{L}|} \exp(-d(g_\theta(q), \boldsymbol{c}_l)} \tag{9}$$

As Snell et al. (2017) mentioned, squared Euclidean distance is a reasonable choice, however, we will introduce a more effective method in section 4.4.1, which combines squared Euclidean distance with class feature scores, and achieves definite improvement.

### 4.4 Hierarchical Attention

We focus on sentence-level text classification in this work. The proposed model gets a feature scores vector and transfers the support set of each class into a vector representation, on which we build a classifier to perform few-shot text classification.

#### 4.4.1 Feature Level Attention

Obviously, the same dimension belonging to different classes has different importance when we calculate the euclidean distance. In other words, some feature dimensions are more discriminative for distinguishing specific class in the feature level space, and other features are confusing and useless at the same time.

So we apply a CNN-based feature attention mechanism similar to Gao et al. (2019) proposed as a class feature extractor. It depends on all the instances in the support set of each class and will dynamiclly change with different classes.

Given a support set $\boldsymbol{S}_i \in \mathbb{R}^{n_i \times T \times d}$ of class $l_i$ as the output of above instance encoder part

$$\boldsymbol{S}_i = \{\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^{n_i}\} \tag{10}$$

we apply a max pooling layer over each instance in $\boldsymbol{S}_i$ to get a new feature map $\boldsymbol{S}_{\boldsymbol{c}i} \in \mathbb{R}^{n_i \times d}$. Then we use three convolution layers to obtain $\boldsymbol{\lambda}_i \in \mathbb{R}^d$, which is the scores vector of class $l_i$. The specific structure of above class feature extractor is shown in Table 1.

| layer name | kernel size | stride | output size |
|---|---|---|---|
| pool | $T \times 1$ | $1 \times 1$ | $K \times d \times 1$ |
| conv_1 | $K \times 1$ | $1 \times 1$ | $K \times d \times 32$ |
| | ReLU | | |
| conv_2 | $K \times 1$ | $1 \times 1$ | $K \times d \times 64$ |
| | ReLU | | |
| conv_3 | $K \times 1$ | $K \times 1$ | $1 \times d \times 1$ |
| | ReLU | | |

Table 1: Class feature extractor architecture

So we get a new distance calculation method as follow

$$d(\boldsymbol{c}_i, \boldsymbol{q}^{'}) = (\boldsymbol{c}_i - \boldsymbol{q}^{'})^2 \cdot \boldsymbol{\lambda}_i \tag{11}$$

where $\boldsymbol{q}^{'}$ is the query vector passed through the word level attention mechanism which will be introduced in the next subsection.

#### 4.4.2 Word Level Attention

The importance of different words to the meanings of an instance is unequal, thus it is worth pointing out which words are useful and which words are useless. Therefore, we apply an attention mechanism (Yang et al., 2016) to get those important words and assemble them to compose a more informative instance vector $\boldsymbol{s}^j$, and the definitions are as follows

$$\boldsymbol{u}_t^j = \tanh(\boldsymbol{W}_w \boldsymbol{h}_t^j + \boldsymbol{b}_w) \tag{12}$$

$$v_t^j = \boldsymbol{u}_t^{j\top} \boldsymbol{u}_w \tag{13}$$

$$\alpha_t^j = \frac{\exp(v_t^j)}{\Sigma_t \exp(v_t^j)} \tag{14}$$

$$\boldsymbol{s}^j = \sum_t \alpha_t^j \boldsymbol{h}_t^j \tag{15}$$

where $\boldsymbol{h}_t^j$ is the $t$ hidden word embedding of instance $\boldsymbol{x}^j$, it was encoded through the instance encoder, and has the same hidden size with $\boldsymbol{x}^j$.

Firstly, the $\boldsymbol{W}_w$ and $\boldsymbol{b}_w$ followed by activation function $\tanh$ make up a MLP layer to transform $\boldsymbol{h}_t^j$ to the new hidden representation $\boldsymbol{u}_t^j$. Immediately, we apply a dot product operation between $\boldsymbol{u}_t^j$ and a word level weight vector $\boldsymbol{u}_w$ to compute similarity $v_t^j$ as the importance weight of $\boldsymbol{u}_t^j$. Then we use a softmax function to normalize $v_t^j$ to $\alpha_t^j$. Finally, we calculate the instance level vector $\boldsymbol{s}^j$ through the weighted sum of $\alpha_t^j$ and $\boldsymbol{h}_t^j$. As memory networks (Sukhbaatar et al., 2015) proposed, $\boldsymbol{u}_w$ can help us to select the important words in each instance, it will be randomly initialized at the beginning of the training stage, and be optimized together with the networks parameters $\theta$.

#### 4.4.3 Instance Level Multi Cross Attention

The previous prototypical networks use the mean vector of support instances as the class prototype. Because of the diversity and lack of the support instances, the gap between each support vector and prototype maybe wide, meanwhile, different query instances can be expressed in several ways, so not every instance in a support set contributes equally to the class prototype when they face a target query instance. To highlight the importance of support instances which are useful clues to classify a query instance correctly, we propose a multi cross attention mechanism.

Given a support set $\boldsymbol{S}_i^{'} \in \mathbb{R}^{n_i \times d}$ for class $l_i$ and a query vector $\boldsymbol{q}^{'} \in \mathbb{R}^d$, they are all encoded through the instance encoder and word level attention. We consider each support vector $\boldsymbol{s}_i^j$ in $\boldsymbol{S}_i^{'}$ has its own weight $\beta_i^j$ to query $\boldsymbol{q}^{'}$. So the formula (8) will be rewritten as follow

$$\boldsymbol{c}_i = \sum_{j=1}^{n_i} \beta_i^j \boldsymbol{s}_i^j \tag{16}$$

where we define $\boldsymbol{r}_i^j = \beta_i^j \boldsymbol{s}_i^j$ as the weighted prototype vector and the definitions of $\beta_i^j$ are as follows

$$\beta_i^j = \frac{\exp(\gamma_i^j)}{\Sigma_{j=1}^{n_i} \exp(\gamma_i^j)} \tag{17}$$

$$\gamma_i^j = \text{sum}\{\sigma(f_\varphi(mca))\} \tag{18}$$

$$mca = [\boldsymbol{s}_{i\phi}^j \oplus \boldsymbol{q}_\phi^{'} \oplus \boldsymbol{\tau}_1 \oplus \boldsymbol{\tau}_2] \tag{19}$$

$$\boldsymbol{\tau}_1 = |\boldsymbol{s}_{i\phi}^j - \boldsymbol{q}_\phi^{'}|, \boldsymbol{\tau}_2 = \boldsymbol{s}_{i\phi}^j \odot \boldsymbol{q}_\phi^{'} \tag{20}$$

$$\boldsymbol{s}_{i\phi}^j = f_\phi(\boldsymbol{s}_i^j), \boldsymbol{q}_\phi^{'} = f_\phi(\boldsymbol{q}^{'}) \tag{21}$$

where $f_\phi$ is a linear layer, $|\cdot|$ is element-wise absolute value and $\odot$ is element-wise product, we use these two operation to get the difference information $\boldsymbol{\tau}_1$ and $\boldsymbol{\tau}_2$ between $\boldsymbol{s}_i^j$ and $\boldsymbol{q}^{'}$, then concatenate them all as the multi cross attention information $mca$, then $f_\varphi(\cdot)$ is a linear layer, $\sigma(\cdot)$ is a $tanh$ activation function, sum$\{\cdot\}$ means a sum operation of all elements in the vector. Finally, $\gamma_i^j$ is the weight of $j$ instance in support set $\boldsymbol{s}_i$, and we use a softmax function to nomalize it to $\beta_i^j$.

Through the multi cross attention mechanism, the prototype can pay more attention to those query-related support instances and improve the capacity of support set.

## 5 Experiments

In this section, we will introduce the experiment results of our model. Firstly, we evaluate our model on FewRel dataset and CSID dataset, and achieve state-of-the-art results, our model outperforms the best baselines models by 1.11% and 1.64% respectively on 10 way 5 shot setting. Then we will show how our model works by case study and visualization of attention layers. We further demonstrate that the hierarchical attention increases the augmentability of support set and the convergence speed of the model.

### 5.1 Datasets

**FewRel** Few-Shot Relation Classification (Han et al., 2018) is a new large-scale supervised dataset[1]. It consists of 70000 instances on 100 relations derived from Wikipedia, and each relation includes 700 instances. It also marks the head and tail entities in each instance, and the average number of tokens is 24.99. FewRel has 64 relations for training, 16 relations for validation, and 20 relations for test separately.

**CSID** Character Studio Intention Detection is a dataset extracted from a real-world open domain chatbot. In character studio platform, this chatbot should transform its character style sometime so it can adapt to different user group and environment, thus dialog query intention detection turns into an important task. CSID consists of 24596 instances for 128 intentions, and each intention includes 30 to 260 instances, the average number of tokens in each instance is 11.52. We use 80, 18 and 30 intentions for training, validation, and test respectively.

[1] https://github.com/thunlp/FewRel

| Model | 5 Way 5 Shot | 5 Way 10 Shot | 10 Way 5 Shot | 10 Way 10 Shot |
|---|---|---|---|---|
| Finetune | 69.43 ± 0.30 | 71.56 ± 0.33 | 58.31 ± 0.26 | 62.12 ± 0.38 |
| kNN | 71.94 ± 0.33 | 73.20 ± 0.35 | 61.69 ± 0.36 | 66.49 ± 0.42 |
| MetaN | 79.46 ± 0.35 | 83.63 ± 0.38 | 70.49 ± 0.52 | 72.84 ± 0.69 |
| GNN | 80.42 ± 0.56 | 83.50 ± 0.63 | 63.08 ± 0.70 | 64.81 ± 0.85 |
| SNAIL | 78.64 ± 0.19 | 81.22 ± 0.23 | 69.46 ± 0.25 | 71.29 ± 0.27 |
| Proto | 83.79 ± 0.12 | 86.05 ± 0.10 | 75.25 ± 0.14 | 77.14 ± 0.19 |
| PHATT | 86.96 ± 0.05 | 87.56 ± 0.08 | 78.62 ± 0.11 | 80.94 ± 0.14 |
| HAPN-FA | 86.53 ± 0.07 | 87.05 ± 0.07 | 78.23 ± 0.09 | 80.73 ± 0.12 |
| HAPN-WA | 87.91 ± 0.09 | 87.83 ± 0.12 | 79.31 ± 0.10 | 81.06 ± 0.17 |
| HAPN-IMCA | 88.07 ± 0.09 | 88.96 ± 0.11 | 80.02 ± 0.12 | 81.87 ± 0.15 |
| HAPN | **88.45 ± 0.06** | **89.72 ± 0.08** | **80.26 ± 0.11** | **82.68 ± 0.13** |

Table 2: Accuracies (%) of different models on the CSID dataset on four different settings.

### 5.2 Baselines

Firstly, we compare our model with several traditional models such as Finetune and kNN, Then we compare our model with five state-of-the-art few-shot learning models based on neural networks, they are MetaN (Munkhdalai and Yu, 2017), GNN (Garcia and Bruna, 2018), SNAIL (Mishra et al., 2018), Proto (Snell et al., 2017) and PHATT (Gao et al., 2019) respectively.

### 5.3 Implementation details

We compare our models with seven baselines, and the implementation details are as follows.

For FewRel dataset, we cite the results reported by Snell et al. (2017) which includes Finetune, kNN, MetaN, GNN, and SNAIL, then we cite the results reported by Gao et al. (2019) which includes Proto and PHATT. For a fair comparison, in our model, we use the same word embeddings and hyperparameters of instance encoder as PHATT proposed. In detail, we use the Glove (Pennington et al., 2014) consisting of 6B tokens and 400K vocabulary as our initialized word representation, and each word has a 50 dimensions vector. In addition, the position embedding dimension of a word is 10, the max length of each instance is 40. Finally, we evaluate all models on 5 way 5 shot and 10 way 5 shot settings.

For CSID dataset, we implement all above seven baseline models and our models. we use the Baidu Encyclopedia (Li et al., 2018) as our initialized word representation, it includes 745M tokens and 5422K vocabulary, and each word has a 300d dimensions vector, the max length of each instance is 20. Finally, we evaluate all models on 5 way 5 shot, 5 way 10 shot, 10 way 5 shot and 10 way 10 shot settings.

For the Finetune and kNN baselines, they learn the parameters on the support set with the CNN encoder. For the neural networks based baselines, we use the same hyper parameters as Han et al. (2018) proposed.

For our hierarchical attention prototypical networks, the window size of the CNN instance encoder is 3, the dimension of the hidden layer is 230, the learning rate is 0.1, the learning rate decay step is 3000 and the decay rate is 0.1. In addition, we train our model 12000 episodes and each episode consists of 20 classes.

In order to study the effects of different components, we refer to our models as HAPN-{FA,WA, IMCA}, FA indicates feature level attention, WA indicates word level attention and IMCA indicates instance level multi cross attention.

### 5.4 Results and analysis

The experimental accuracies on CSID and FewRel are shown in Tabel 2 and Table 4 respectively. In this subsection, we will show the effects of hierarchical attention and support set augmentability of three Proto-based models and the convergence speed comparison.

#### 5.4.1 Effects of hierarchical attention

Benefit from hierarchical attention, our model achieves excellent performance.

The case study of word level attention and instance level multi cross attention are shown in Table 3, this is a 2 way 3 shot task on FewRel dataset. The query instance is an instance of “mother” class in fact, and our model should classify it into “mother” class or “child” class. It is a difficult

| Class | Word Attention | IMCAS |
|---|---|---|
| | Support Set | |
| (1) mother | **Cherie Gil** is the daughter of Filipino actors Eddie Mesa and **Rosemarie Gil**, and sister of fellow actors, Michael de Mesa and the late Mark Gil. | |
| | When they reached adulthood, Pelias and **Neleus** found their mother **Tyro** and then killed her stepmother, Sidero, for having mistreated her. | |
| | It was here that the Queen Consort **Jetsun Pema** gave birth to a son on 5 February 2016, **Jigme Namgyel Wangchuck**. | |
| (2) child | In 1421 **Mehmed** died and his son **Murad II** refused to honour his father's obligations to the Byzantines. | |
| | **Henry Norreys** was a lifelong friend of Queen Elizabeth and was the father of six sons, who included Sir **John Norreys**, a famous English soldier. | |
| | **Jim Henson** and his son **Brian** were impressed enough with Barron's style to offer him a job directing the pilot episode of "The Storyteller". | |
| | Query | |
| (1) or (2) | From 1922 to 1963, **Princess Dagmar of Demark**, the daughter of Frederick VIII of Denmark and **Louise of Sweden**, lived on Kongestlund. | |

Table 3: Visualization of word level and instance level multi cross attention scores (IMCAS) for 2 way 3 shot setting, the bold words are head entities and tail entities.

| Model | 5 Way 5 Shot | 10 Way 5 Shot |
|---|---|---|
| Finetune* | 68.66 ± 0.41 | 55.04 ± 0.31 |
| kNN* | 68.77 ± 0.41 | 55.87 ± 0.31 |
| MetaN* | 80.57 ± 0.48 | 69.23 ± 0.52 |
| GNN* | 81.28 ± 0.62 | 64.02 ± 0.77 |
| SNAIL* | 79.40 ± 0.22 | 68.33 ± 0.25 |
| Proto◇ | 89.05 ± 0.09 | 81.46 ± 0.13 |
| PHATT◇ | 90.12 ± 0.04 | 83.05 ± 0.05 |
| HAPN-FA | 89.79 ± 0.13 | 82.47 ± 0.20 |
| HAPN-WA | 90.86 ± 0.12 | 83.79 ± 0.19 |
| HAPN-IMCA | 90.92 ± 0.11 | 84.07 ± 0.19 |
| HAPN | **91.02 ± 0.11** | **84.16 ± 0.18** |

Table 4: Accuracies (%) for 5 way 5 shot and 10 way 5 shot settings on FewRel test set. * reported by Han et al. (2018) and ◇ reported by Gao et al. (2019).

task because of there are many similarities between the expressions of two classes. With the help of word level attention, we highlight the importance of the word "daughter", which appears in the query instance and the first support instance of class "mother" at the same time, then this support instance get the highest attention score and contributes more to the prototype vector of "mother" class, finally our model can classify the query instance into the correct class in this confusing task.

As shown in Figure 2, by using the feature level attention, we also get the feature attention scores of "mother" class and "child" class respectively. The features with high scores have deep color, and the features with low scores have light color. Obviously, different classes may have different feature score vector, in other words, the same feature of different classes have different importance. So our feature level attention can highlight importance of the useful features and weaken the importance of the noise features, then the distance between the prototype vector and the query vector will measure the difference between them more efficiently.



(a) Feature attention scores of "mother" class



(b) Feature attention scores of "child" class

Figure 2: Feature attention scores of different classes

We treat the final prototype embedding vector as the features of each instance, then we can get the distribution of features by principal pomponent analysis in feature space as shown in Figure 3. As we can see, the instances without hierarchical attention are more distributed and may cross with each other, but the instances with hierarchical attention are more centralized and discriminative, which proves that our model learns a better semantic space, which helps to distinguish confus-

ing data..

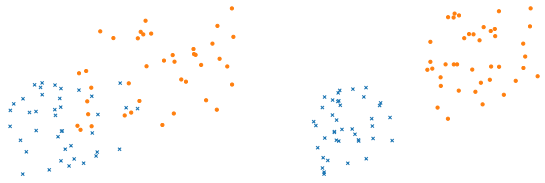

(a) Instances without hierarchical attention

(b) Instances with hierarchical attention

Figure 3: Instances distribution of embedding vector without hierarchical attention (a) and with hierarchical attention (b). The left blue points marked × are instances of "mother" class and the right orange points marked • are instances of "child" class.

### 5.4.2 Augmentability of support set

More support instances can contribute more useful information to the prototype vector, meanwhile, more noise will be added in.

In this section, we define the support set augmentability (SSA) as the additive value of accuracy when we increase the same number of the support set for different models. So we compare our model's SSA with other models such as Proto and PHATT on the 10 way FewRel task, and the shot number ranges from 5 to 25.

By using the hierarchical attention, our model obtaines a strong robustness and can pay more attention to the important information of support set and reduce those negative effects of noisy data, thus as shown in Figure 4, the support set augmentability of our model is larger than other models. Benefit from the above advantages, we can deploy our model in the cold start stage, and gradually accumulate labeled support data in practical applications, then improve the performance of the model day by day, and thus improve the utilization rate of few data in realistic settings.

### 5.4.3 Convergence speed comparison

At the training stage, we also compare the convergence speed between Proto, PHATT, and HAPN on the 10 way 5 shot and 10 way 15 shot FewRel task. As shown in Figure 5, our model can be optimized more quickly than the other models. From 10 way 5 shot task to 10 way 15 shot settings, the Proto model takes almost twice time to achieve 70% accuracy on validation set, in other words, the convergence speed will decrease sharply when we increase the number of support instances, but this problem can be effectively alleviated when we use hierarchical attention mechanism.

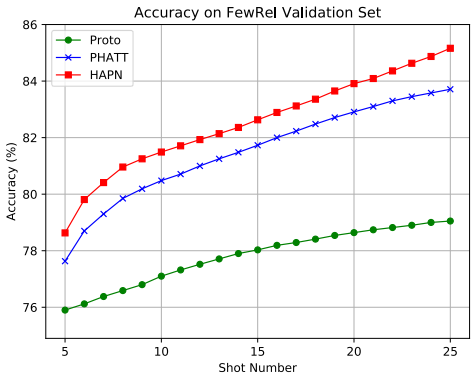


Figure 4: Support set augmentability of Proto, PHATT and HAPN on FewRel validation set.

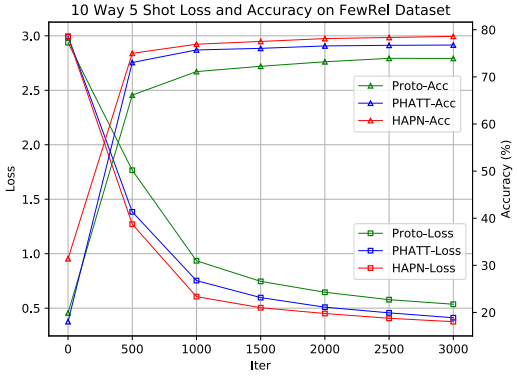


Figure 5: Training Proto, PHATT and HAPN on FewRel dataset. Lines marked □ denote loss on the training set and lines marked △ denote accuracy on the validation set.

## 6 Conclusion

Previous few-shot learning models for text classification roughly apply text representations or neglect the noisy information. We propose to do hierarchical attention prototypical networks consisting of feature level, word level and instance level multi cross attention, which highlight the important information of few data and learn a more discriminative prototype representation. In the experiments, our model achieves the state-of-the-art performance on FewRel and CSID datasets. HAPN not only increases support set augmentability but also accelerates convergence speed in the training stage.

In the future, we will contribute new text dataset to few-shot learning, explore better feature extrac-

tor networks and do some industrial application.

## Acknowledgements

We would like to thank Sawyer Zeng and Yue Liu for providing valuable hardware support and useful advice, and thank Xuexiang Xu and Yang Bai for helping us test online FewRel dataset. This work is also supported by the National Key Research and Development Program of China (No. 2018YFB1402902 and No. 2018YFB1403002) and the Natural Science Foundation of Jiangsu Province (No. BK20151132).

## References

Yoshua Bengio. 2011. Deep learning of representations for unsupervised and transfer learning. In *Unsupervised and Transfer Learning - Workshop held at ICML 2011, Bellevue, Washington, USA, July 2, 2011*, pages 17–36.

Rich Caruana. 1994. Learning many related tasks at the same time with backpropagation. In *Advances in Neural Information Processing Systems 7, [NIPS Conference, Denver, Colorado, USA, 1994]*, pages 657–664.

Zitian Chen, Yanwei Fu, Yinda Zhang, Yu-Gang Jiang, Xiangyang Xue, and Leonid Sigal. 2018. Semantic feature augmentation in few-shot learning. volume abs/1804.05298.

Tianyu Gao, Zhiyuan Liu Xu Han, and Maosong Sun. 2019. Hybrid attention-based prototypical networks for noisy few-shot relation classification. In *Proceedings of the Association for the Advancement of Artificial Intelligence*.

Victor Garcia and Joan Bruna. 2018. Few-shot learning with graph neural networks. In *Proceedings of ICLR*.

Xu Han, Hao Zhu, Pengfei Yu, Ziyun Wang, Yuan Yao, Zhiyuan Liu, and Maosong Sun. 2018. Fewrel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 4803–4809.

Rie Johnson and Tong Zhang. 2015. Effective use of word order for text categorization with convolutional neural networks. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 103–112.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. arXiv:1408.5822.

Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. 2015. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning workshop*.

Shen Li, Zhe Zhao, Renfen Hu, Wensi Li, Tao Liu, and Xiaoyong Du. 2018. Analogical reasoning on chinese morphological and semantic relations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 138–143. Association for Computational Linguistics.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. 2018. A simple neural attentive meta-learner. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Tsendsuren Munkhdalai and Hong Yu. 2017. Meta networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 2554–2563.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543.

Jake Snell, Kevin Swersky, and Richard S. Zemel. 2017. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 4080–4090.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. Yara parser: A fast and accurate dependency parser. *End-to-end memory networks*, arXiv:1503.08895.

Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H. S. Torr, and Timothy M. Hospedales. 2018. Learning to compare: Relation network for few-shot learning. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 1199–1208.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1422–1432.

Oriol Vinyals, Charles Blundell, Tim Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. 2016. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems 29: Annual*

*Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3630–3638.

Guoyin Wang, Chunyuan Li, Wenlin Wang, Yizhe Zhang, Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. 2018. Joint embedding of words and labels for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 2321–2331.

Sida I. Wang and Christopher D. Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, July 8-14, 2012, Jeju Island, Korea - Volume 2: Short Papers*, pages 90–94.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. 2016. Hierarchical attention networks for document classification. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 1480–1489.

Mo Yu, Xiaoxiao Guo, Jinfeng Yi, Shiyu Chang, Saloni Potdar, Yu Cheng, Gerald Tesauro, Haoyu Wang, and Bowen Zhou. 2018. Diverse few-shot text classification with multiple metrics. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1206–1215.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, August 23-29, 2014, Dublin, Ireland*, pages 2335–2344.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015a. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 649–657.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015b. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 649–657.

Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis C. M. Lau. 2015. A C-LSTM neural network for text classification. volume abs/1511.08630.

# Many Faces of Feature Importance: Comparing Built-in and Post-hoc Feature Importance in Text Classification

**Vivian Lai** and **Zheng Cai** and **Chenhao Tan**
Department of Computer Science
University of Colorado Boulder
Boulder, CO
vivian.lai, jon.z.cai, chenhao.tan@colorado.edu

## Abstract

Feature importance is commonly used to explain machine predictions. While feature importance can be derived from a machine learning model with a variety of methods, the consistency of feature importance via different methods remains understudied. In this work, we systematically compare feature importance from built-in mechanisms in a model such as attention values and post-hoc methods that approximate model behavior such as LIME. Using text classification as a testbed, we find that 1) no matter which method we use, important features from traditional models such as SVM and XGBoost are more similar with each other, than with deep learning models; 2) post-hoc methods tend to generate more similar important features for two models than built-in methods. We further demonstrate how such similarity varies across instances. Notably, important features do *not* always resemble each other better when two models agree on the predicted label than when they disagree.

## 1 Introduction

As machine learning models are adopted in societally important tasks such as recidivism prediction and loan approval, explaining machine predictions has become increasingly important (Doshi-Velez and Kim, 2017; Lipton, 2016). Explanations can potentially improve the trustworthiness of algorithmic decisions for decision makers, facilitate model developers in debugging, and even allow regulators to identify biased algorithms.

A popular approach to explaining machine predictions is to identify important features for a particular prediction (Luong et al., 2015; Ribeiro et al., 2016; Lundberg and Lee, 2017). Typically, these explanations assign a value to each feature (usually a word in NLP), and thus enable visualizations such as highlighting top $k$ features.

In general, there are two classes of methods: 1) built-in feature importance that is embedded in the machine learning model such as coefficients in linear models and attention values in attention mechanisms; 2) post-hoc feature importance through credit assignment based on the model such as LIME. It is well recognized that robust evaluation of feature importance is challenging (Jain and Wallace, 2019; Nguyen, 2018, inter alia), which is further complicated by different use cases of explanations (e.g., for decision makers vs. for developers). Throughout this work, we refer to machine learning models that learn from data as *models* and methods to obtain local explanations (i.e., feature importance in this work) for a prediction by a model as *methods*.

While prior research tends to focus on the internals of models in designing and evaluating methods of explanations, e.g., how well explanations reflect the original model (Ribeiro et al., 2016), we view feature importance itself as a subject of study, and aim to provide a systematic characterization of important features obtained via different methods for different models. This view is particularly important when explanations are used to support decision making because they are the only exposure to the model for decision makers. It would be desirable that explanations are consistent across different instances. In comparison, debugging represents a distinct use case where developers often know the mechanism of the model beyond explanations. Our view also connects to studying explanation as a *product* in cognitive studies of explanations (Lombrozo, 2012), and is complementary to the model-centric perspective.

Given a wide variety of models and methods to generate feature importance, there are basic open questions such as how similar important features are between models and methods, how important features distribute across instances, and what lin-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 486–495,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

One of favorite places to eat on the King W side, simple and relatively quick. I typically always get the chicken burrito and the small is enough for me for dinner. Ingredients are always fresh and watch out for the hot sauce cause it's skull scratching hot. Seating is limited so be prepared to take your burrito outside or you can even eat at Metro Hall Park.

| methods \ models | SVM ($\ell_2$) | XGBoost | LSTM with attention | BERT |
|---|---|---|---|---|
| built-in | sauce, seating, park, prepared, even, always, can, fresh, quick, favorite | is, can, quick, fresh, at, to, always, even, favorite, and | me, be, relatively, enough, always, fresh, ingredients, prepared, quick, favorite | ., ingredients, relatively, quick, places, enough, dinner, typically, me, i |
| LIME | the, dinner, be, quick, and, even, you, always, fresh, favorite | you, to, fresh, quick, at, can, even, always, and, favorite | dinner, ingredients, typically, fresh, places, cause, quick, and, favorite, always | one, watch, to, enough, limited, cause, and, fresh, hot, favorite |

Table 1: 10 most important features (separated by comma) identified by different methods for different models for the given review. In the interest of space, we only show built-in and LIME here.

guistic properties important features tend to have. We use text classification as a testbed to answer these questions. We consider built-in importance from both traditional models such as linear SVM and neural models with attention mechanisms, as well as post-hoc importance based on LIME and SHAP. Table 1 shows important features for a Yelp review in sentiment classification. Although most approaches consider "fresh" and "favorite" important, there exists significant variation.

We use three text classification tasks to characterize the overall similarity between important features. Our analysis reveals the following insights:

- (Comparison between approaches) Deep learning models generate more different important features from traditional models such as SVM and XGBoost. Post-hoc methods tend to reduce the dissimilarity between models by making important features more similar than the built-in method. Finally, different approaches do not generate more similar important features even if we focus on the most important features (e.g., top one feature).
- (Heterogeneity between instances) Similarity between important features is not always greater when two models agree on the predicted label, and longer instances are less likely to share important features.
- (Distributional properties) Deep models generate more diverse important features with higher entropy, which indicates lower consistency across instances. Post-hoc methods bring the POS distribution closer to background distributions.

In summary, our work systematically compares important features from different methods for different models, and sheds light on how different models/methods induce important features. Our work takes the first step to understand important features as a product and helps inform the adoption of feature importance for different purposes. Our code is available at https://github.com/BoulderDS/feature-importance.

## 2 Related Work

To provide further background for our work, we summarize current popular approaches to generating and evaluating explanations of machine predictions, with an emphasis on feature importance.

**Approaches to generating explanations.** A battery of approaches have been recently proposed to explain machine predictions (see Guidotti et al. (2019) for an overview), including example-based approaches that identifies "informative" examples in the training data (e.g., Kim et al., 2016) and rule-based approaches that reduce complex models to simple rules (e.g., Malioutov et al., 2017). Our work focuses on characterizing properties of feature-based approaches. Feature-based approaches tend to identify important features in an instance and enable visualizations with important features highlighted. We discuss several directly related post-hoc methods here and introduce the built-in methods in §3. A popular approach, LIME, fits a sparse linear model to approximate model predictions locally (Ribeiro et al., 2016); Lundberg and Lee (2017) present a unified framework based on Shapley values, which can be computed with different approximation methods for different models. Gradients are popular for identifying important features in deep learning models since these models are usually differentiable (Shrikumar et al., 2017), for instance, Li et al. (2016) uses gradient-based saliency to compare LSTMs with simple recurrent networks.

**Definition and evaluation of explanations.** De-

spite a myriad of studies on approaches to explaining machine predictions, explanation is a rather overloaded term and evaluating explanations is challenging. Doshi-Velez and Kim (2017) lays out three levels of evaluations: functionally-grounded evaluations based on proxy automatic tasks, human-grounded evaluations with laypersons on proxy tasks, and application-grounded based on expert performance in the end task. In text classification, Nguyen (2018) shows that automatic evaluation based on word deletion moderately correlate with human-grounded evaluations that ask crowdworkers to infer machine predictions based on explanations. However, explanations that help humans infer machine predictions may not actually help humans make better decisions/predictions. In fact, recent studies find that feature-based explanations alone have limited improvement on human performance in detecting deceptive reviews and media biases (Lai and Tan, 2019; Horne et al., 2019).

In another recent debate, Jain and Wallace (2019) examine attention as an explanation mechanism based on how well attention values correlate with gradient-based feature importance and whether they exclusively lead to the predicted label, and conclude that attention is not explanation. Similarly, Serrano and Smith (2019) show that attention is not a fail-safe indicator for explaining machine predictions based on intermediate representation erasure. However, Wiegreffe and Pinter (2019) argue that attention can be explanation depending on the definition of explanations (e.g., plausibility and faithfulness).

In comparison, we treat feature importance itself as a subject of study and compare different approaches to obtaining feature importance from a model. Instead of providing a normative judgment with respect to what makes good explanations, our goal is to allow decision makers or model developers to make informed decisions based on properties of important features using different models and methods.

## 3 Approach

In this section, we first formalize the problem of obtaining feature importance and then introduce the models and methods that we consider in this work. Our main contribution is to compare important features identified for a particular instance through different methods for different models.

**Feature importance.** For any instance $t$ and a machine learning model $m : t \rightarrow y \in \{0, 1\}$, we use method $h$ to obtain feature importance on an interpretable representation of $t$, $\mathcal{I}_t^{m,h} \in \mathbb{R}^d$, where $d$ is the dimension of the interpretable representation. In the context of text classification, we use unigrams as the interpretable representation. Note that the machine learning model does not necessarily use the interpretable representation. Next, we introduce the models and methods in this work.

**Models ($m$).** We include both recent deep learning models for NLP and popular machine learning models that are not based on neural networks. In addition, we make sure that the chosen models have some built-in mechanism for inducing feature importance and describe the built-in feature importance as we introduce the model.[1]

- Linear SVM with $\ell_2$ (or $\ell_1$) regularization. Linear SVM has shown strong performance in text categorization (Joachims, 1998). The absolute value of coefficients in these models is typically considered a measure of feature importance (e.g., Ott et al., 2011). We also consider $\ell_1$ regularization because $\ell_1$ regularization is often used to induce sparsity in the model.

- Gradient boosting tree (XGBoost). XGBoost represents an ensembled tree algorithm that shows strong performance in competitions (Chen and Guestrin, 2016). We use the default option in XGBoost to measure feature importance with the average training loss gained when using a feature for splitting.

- LSTM with attention (often shortened as LSTM in this work). Attention is a commonly used technique in deep learning models for NLP (Bahdanau et al., 2015). The intuition is to assign a weight to each token before aggregating into the final prediction (or decoding in machine translation). We use the dot product formulation in Luong et al. (2015). The weight on each token has been commonly used to visualize the importance of each token. To compare with the previous bag-of-words models, we use the average weight of each type (unique token) in this work to measure feature importance.

- BERT. BERT represents an example architecture based on Transformers, which could show different behavior from LSTM-style recurrent

[1] For instance, we do not consider LSTM as a model here due to the lack of commonly-accepted built-in mechanisms.

networks (Devlin et al., 2019; Vaswani et al., 2017; Wolf, 2019). It also achieves state-of-the-art performance in many NLP tasks. Similar to LSTM with attention, we use the average attention values of 12 heads used by the first token at the final layer (the representation passed to fully connected layers) to measure feature importance for BERT.[2] Since BERT uses a subword tokenizer, for each word, we aggregate the attention on related subparts. BERT also requires special processing due to the length constraint; please refer to the supplementary material for details. As a result, we focus on presenting LSTM with attention in the main paper for ease of understanding.

**Methods ($h$).** For each model, in addition to the built-in feature importance that we described above, we consider the following two popular methods for extracting post-hoc feature importance (see the supplementary material for details of using the post-hoc methods).

- LIME (Ribeiro et al., 2016). LIME generates post-hoc explanations by fitting a local sparse linear model to approximate model predictions. As a result, the explanations are sparse.
- SHAP (Lundberg and Lee, 2017). SHAP unifies several interpretations of feature importance through Shapley values. The main intuition is to account the importance of a feature by examining the change in prediction outcomes for all the combinations of other features. Lundberg and Lee (2017) propose various approaches to approximate the computation for different classes of models (including gradient-based methods for deep models).

Note that feature importances obtained via all approaches are all local, because the top features are conditioned on an instance (i.e., words present in an instance) even for the built-in method for SVM and XGBoost.

**Comparing feature importance.** Given $\mathcal{I}_t^{m,h}$ and $\mathcal{I}_t^{m',h'}$, we use Jaccard similarity based on the top $k$ features with the greatest absolute feature importance, $\frac{|\text{TopK}(\mathcal{I}_t^{m,h}) \cap \text{TopK}(\mathcal{I}_t^{m',h'})|}{|\text{TopK}(\mathcal{I}_t^{m,h}) \cup \text{TopK}(\mathcal{I}_t^{m',h'})|}$, as our main similarity metric for two reasons. First, the most typical way to use feature importance for interpretation purposes is to show the most important features (Lai and Tan, 2019; Ribeiro et al., 2016; Horne et al., 2019). Second, some models and methods inherently generate sparse feature importance, so most feature importance values are 0.

It is useful to discuss the implication of similarity before we proceed. On the one hand, it is possible that different models/methods identify the same set of important features (high similarity) and the performance difference in prediction is due to how different models weigh these important features. If this were true, the choice of model/method would have mattered little for visualizing important features. On the other hand, a low similarity poses challenges for choosing which model/method to use for displaying important features. In that case, this work aims to develop an understanding of how the similarity varies depending on models and methods, instances, and features. We leave it to future work for examining the impact on human interaction with feature importance. Low similarity may enable model developers to understand the differences between models, but may lead to challenges for decision makers to get a consistent picture of what the model relies on.

## 4 Experimental Setup and Hypotheses

Our goal is to characterize the similarities and differences between feature importances obtained with different methods and different models. In this section, we first present our experimental setup and then formulate our hypotheses.

**Experimental setup.** We consider the following three text classification tasks in this work. We choose to focus on classification because classification is the most common scenario used for examining feature importance and the associated human interpretation (e.g., Jain and Wallace, 2019).

- Yelp (Yelp, 2019). We set up a binary classification task to predict whether a review is positive (rating $\geq$ 4) or negative (rating $\leq$ 2). As the original dataset is huge, we subsample 12,000 reviews for this work.
- SST (Socher et al., 2013). It is a sentence-level sentiment classification task and represents a common benchmark. We only consider the binary setup here.
- Deception detection (Ott et al., 2013, 2011).

[2] We also tried to use the max of 12 heads and previous layers, and the average of the final layer is more similar to SVM ($\ell_2$) than the average of first layer. Results are in the supplementary material. Vig (2019) show that attention in BERT tends to be on first words, neighboring words, and even separators. The complex choices for BERT further motivate our work to view feature importance as a subject of study.

| Model | Yelp | SST | Deception |
|---|---|---|---|
| SVM ($\ell_2$) | 92.3 | 80.8 | 86.3 |
| SVM ($\ell_1$) | 91.5 | 79.2 | 84.4 |
| XGBoost | 88.8 | 75.9 | 83.4 |
| LSTM w/ attention | 93.9 | 82.6 | 88.4 |
| BERT | 95.5 | 92.2 | 90.9 |

Table 2: Accuracy on the test set.

This dataset was created by extracting genuine reviews from TripAdvisor and collecting deceptive reviews using Turkers. It is relatively small with 1,200 reviews and represents a distinct task from sentiment classification.

For all the tasks, we use 20% of the dataset as the test set. For SVM and XGBoost, we use cross validation on the other 80% to tune hyperparameters. For LSTM with attention and BERT, we use 10% of the dataset as a validation set, and choose the best hyperparameters based on the validation performance. We use spaCy to tokenize and obtain part-of-speech tags for all the datasets (Honnibal and Montani, 2017). Table 2 shows the accuracy on the test set and our results are comparable to prior work. Not surprisingly, BERT achieves the best performance in all three tasks. For important features, we use $k \leq 10$ for Yelp and deception detection, and $k \leq 5$ for SST as it is a sentence-level task. See supplementary materials for details of preprocessing, learning, and dataset statistics.

**Hypotheses.** We aim to examine the following three research questions in this work: 1) How similar are important features between models and methods? 2) What factors relate to the heterogeneity across instances? 3) What words tend to be chosen as important features?

*Overall similarity.* Here we focus on discussing comparative hypotheses, but we would like to note that it is important to understand to what extent important features are similar across models (i.e., the value of similarity score). First, as deep learning models and XGBoost are non-linear, we hypothesize that built-in feature importance is more similar between SVM ($\ell_1$) and SVM ($\ell_2$) than other model pairs (**H1a**). Second, LIME generates more similar important features to SHAP than to built-in feature importance because both LIME and SHAP make additive assumptions, while built-in feature importance is based on drastically different models (**H1b**). It also follows that post-hoc explanations of different models show higher similarity than built-in explanations across models. Third, the similarity with small $k$ is higher (**H1c**) because hopefully, all models and methods agree what the most important features are.

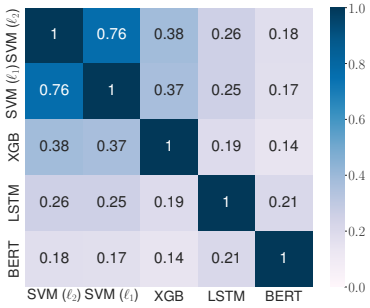


Figure 1: Jaccard similarity between the top 10 features of different models based on built-in feature importance on Yelp. The similarity is the greatest between SVM ($\ell_2$) and SVM ($\ell_1$), while LSTM with attention and BERT pay attention to quite different features from other models.

*Heterogeneity between instances.* Given a pair of (model, method) combinations, our second question is concerned with how instance-level properties affect the similarity in important features between different combinations. We hypothesize that 1) when two models agree on the predicted label, the similarity between important features is greater (**H2a**); 2) longer instances are less likely to share similar important features (**H2b**). 3) instances with higher type-token ratio,[3] which might be more complex, are less likely to share similar important features (**H2c**).

*Distribution of important features.* Finally, we examine what words tend to be chosen as important features. This question certainly depends on the nature of the task, but we would like to understand how consistent different models and methods are. We hypothesize that 1) deep learning models generate more diverse important features (**H3a**); 2) adjectives are more important in sentiment classification, while pronouns are more important in deception detection as shown in prior work (**H3b**).

## 5 Similarity between Instance-level Feature Importance

We start by examining the overall similarity between different models using different methods. In a nutshell, we compute the average Jaccard similarity of top $k$ features for each pair of $(m, h)$ and

[3]Type-token ratio is defined as the number of unique tokens divided by the number of tokens.

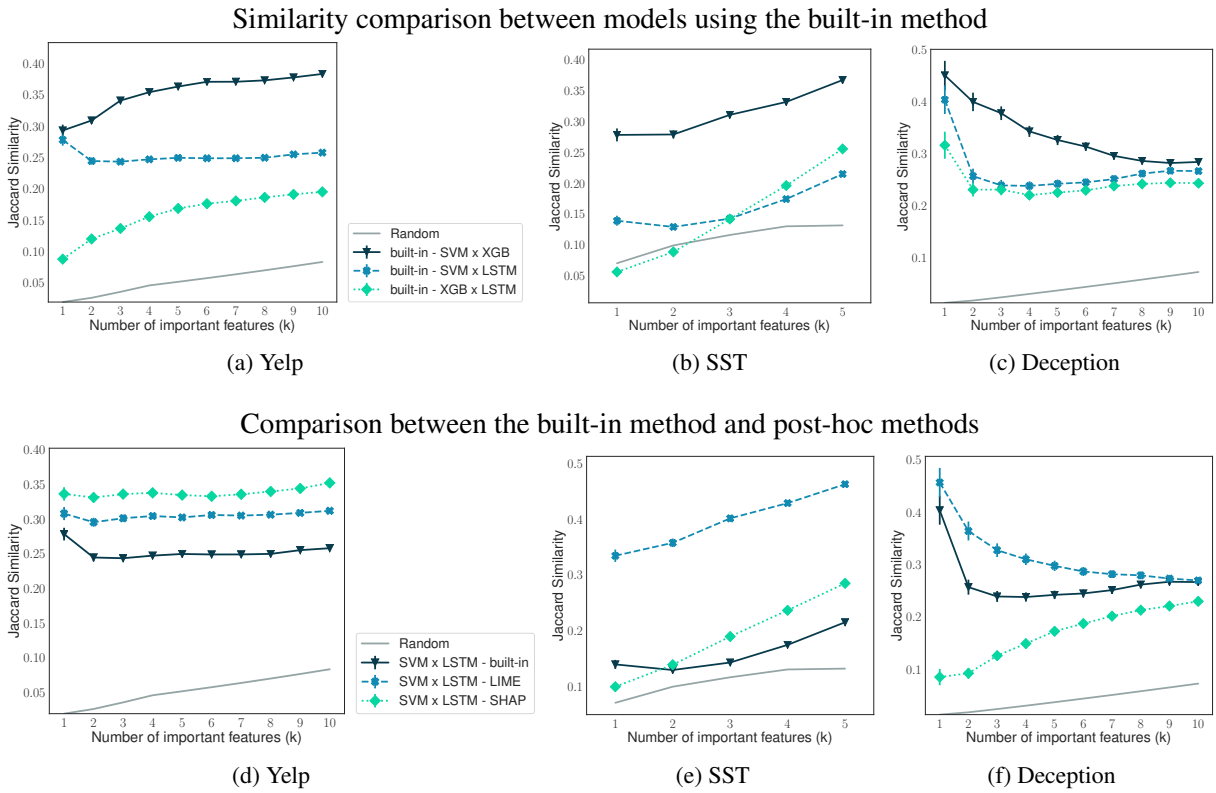


Figure 2: Similarity comparison between models with the same method. $x$-axis represents the number of important features that we consider, while $y$-axis shows the Jaccard similarity. *Error bars represent standard error throughout the paper.* The top row compares three pairs of models using the built-in method, while the second row compares three methods on SVM and LSTM with attention (LSTM in figure legends always refers to LSTM with attention in this work). The random line is derived using the average similarity between two random samples of $k$ features from 100 draws.

$(m', h')$. To facilitate effective comparisons, we first fix the method and compare the similarity of different models, and then fix the model and compare the similarity of different methods. Figure 1 shows the similarity between different models using the built-in feature importance for the top 10 features in Yelp ($k = 10$). Consistent with **H1a**, SVM ($\ell_2$) and SVM ($\ell_1$) are very similar to each other, and LSTM with attention and BERT clearly lead to quite different top 10 features from the other models. As the number of important features ($k$) can be useful for evaluating the overall trend, we thus focus on line plots as in Figure 2 in the rest of the paper. This heatmap visualization represents a snapshot for $k = 10$ using the built-in method. Also, we only include SVM ($\ell_2$) in the main paper for ease of visualization and sometimes refer to it in the rest of the paper as SVM.

**No matter which method we use, important features from SVM and XGBoost are more similar with each other, than with deep learning models (Figure 2).** First, we compare the similarity of feature importance between different models using the same method. Using the built-in method (first row in Figure 2), the solid line (SVM x XGBoost) is always above the other lines, usually by a significant margin, suggesting that deep learning models such as LSTM with attention are less similar to traditional models. In fact, the similarity between XGBoost and LSTM with attention is lower than random samples for $k = 1, 2$ in SST. Similar results also hold for BERT (see supplementary materials). Another interesting observation is that post-hoc methods tend to generate greater similarity than built-in methods, especially for LIME (the dashed line (LIME) is always above the solid line (built-in) in the second row of Figure 2). This is likely because LIME only depends on the model behavior (i.e., what the model predicts) and does not account for how the model works.

**The similarity between important features from different methods tends to be lower for LSTM with attention (Figure 3).** Second, we compare the similarity of feature importance derived from the same model with different methods. For deep learning models such as LSTM with at-

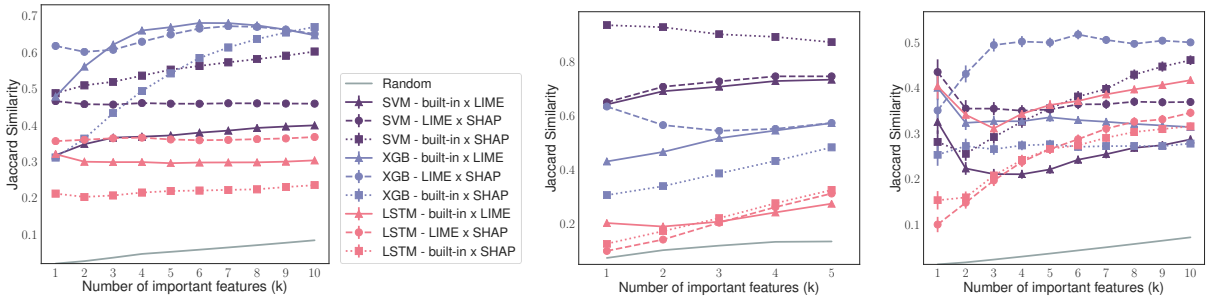


(a) Yelp (b) SST (c) Deception

Figure 3: Similarity comparison between methods using the same model. The similarity between different methods based on LSTM with attention is generally lower than other methods. Similar results hold for BERT (see the supplementary material).

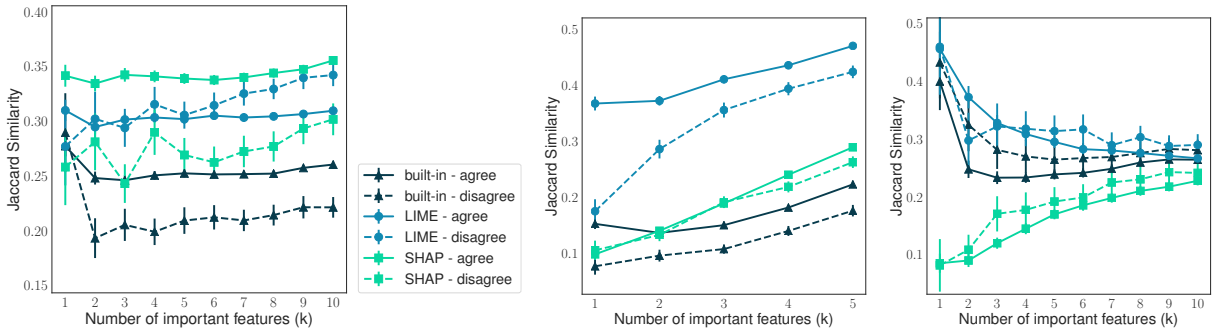


(a) Yelp (b) SST (c) Deception

Figure 4: Similarity between SVM ($\ell_2$) and LSTM with attention with different methods grouped by whether these two models agree on the predicted label. The similarity is not always greater when they agree on the predicted labels than when they disagree.

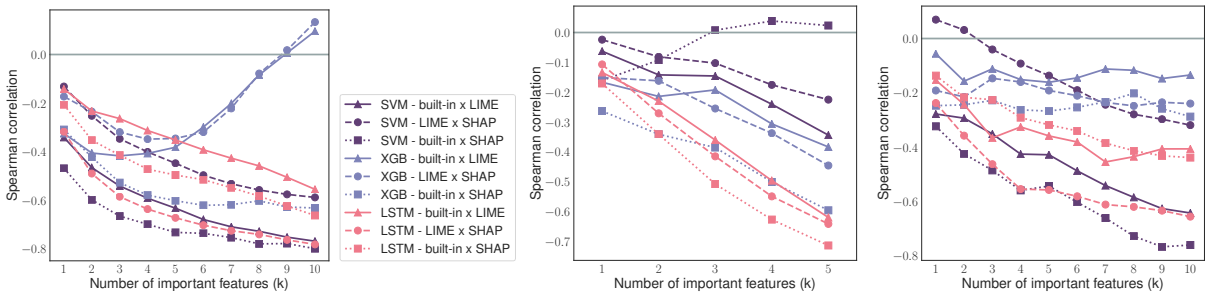


(a) Yelp (b) SST (c) Deception

Figure 5: In most cases, the similarity between feature importance is negatively correlated with length. Here we only show the comparison between different methods based on the same model. Similar results hold for comparison between different models using the same method. For ease of comparison, the gray line marks the value 0. Generally as $k$ grows, relationship becomes even more negatively correlated.

tention, the similarity between feature importance generated by different methods is the lowest, especially comparing LIME with SHAP. Notably, the results are much more cluttered in deception detection. Contrary to **H1b**, we do not observe that LIME is more similar to SHAP than built-in. The order seems to depend on both the task and the model: even within SST, the similarity between built-in and LIME can rank as third, second, or first. In other words, post-hoc methods generate more similar important features when we compare different models, but that is not the case when we fix the model. It is reassuring that that similarity between any pairs is above random, with a sizable margin in most cases (BERT on SHAP is an exception; see supplementary materials).

**Relation with $k$.** As the relative order between different approaches can change with $k$, we have

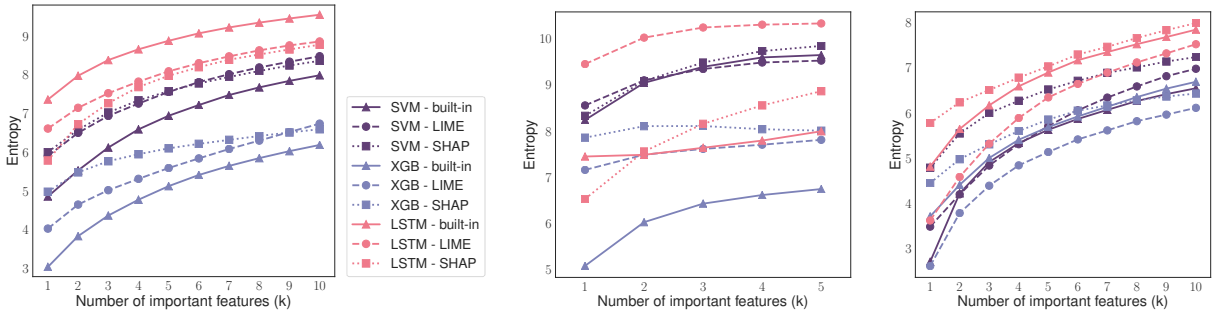


(a) Yelp (b) SST (c) Deception

Figure 6: The entropy of important features. LSTM with attention generates more diverse important features than SVM and XGBoost.

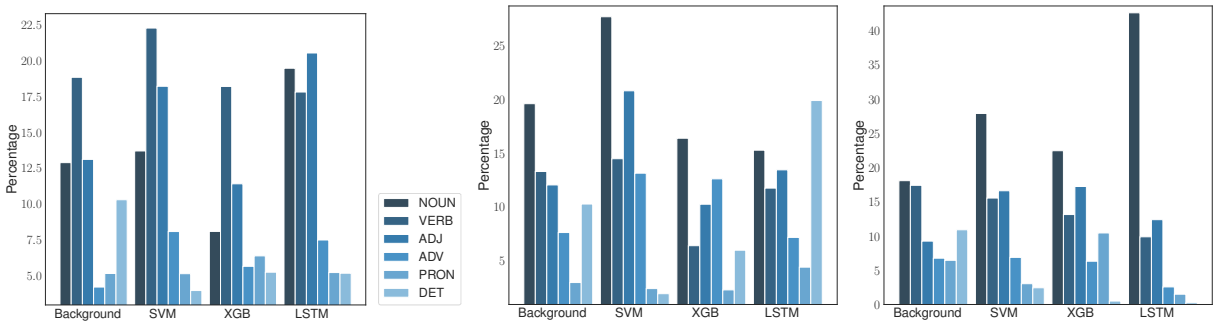


(a) Yelp (b) SST (c) Deception

Figure 7: Part-of-speech tag distribution with the built-in method. "Background" shows the distribution of all words in the test set. LSTM with attention puts a strong emphasis on nouns in deception detection, but is not necessarily more different from the background than other models.

so far only focused on relatively consistent patterns over $k$ and classification tasks. Contrary to **H1c**, the similarity between most approaches is not drastically greater for small $k$, which suggests that different approaches may not even agree on the most important features. In fact, there is no consistent trend as $k$ grows: similarity mostly *increases* in SST (while our hypothesis is that it decreases), increases or stays level in Yelp, and shows varying trends in deception detection.

## 6 Heterogeneity between Instances

Given the overall low similarity between different methods/models, we next investigate how the similarity may vary across instances.

**The similarity between models is not always greater when two models agree on the predicted label (Figure 4).** One hypothesis for the overall low similarity between models is that different models tend to give different predictions therefore they choose different features to support their decisions. However, we find that the similarity between models is not particularly high when they agree on the predicted label, and are sometimes even lower than when they disagree. This is true for LIME in Yelp and for all methods in deception detection. In SST, the similarity when the models agree on the predicted label is generally greater than when they disagree. We show the comparison between SVM ($\ell_2$) and LSTM here, and similar results hold for other combinations (see supplementary materials). This observation suggests that feature importance may not connect with the predicted labels: different models agree for different reasons and also disagree for different reasons.

**The similarity between models and methods is generally negatively correlated with length but positively correlated with type-token ratio (Figure 5).** Our results support **H2b**: Spearman correlation between length and similarity is mostly below 0, which indicates that the longer an instance is, the less similar the important features are. The negative correlation becomes stronger as $k$ grows, indicating that length has a stronger effect on similarity when we consider more top features. However, this is not true in the case of LIME and SHAP where the correlation between length and similarity are occasionally above 0 and sometimes even

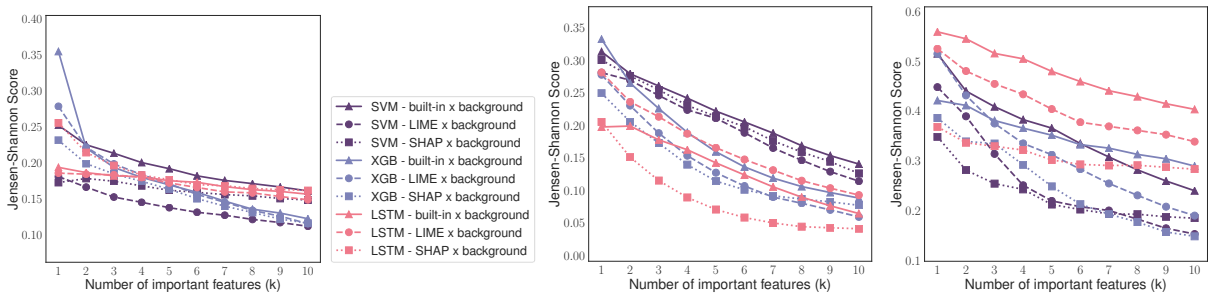


(a) Yelp (b) SST (c) Deception

Figure 8: Distance of the part-of-speech tag distributions between important features and all words (background). Distance is generally smaller with post-hoc methods for all models, although some exceptions exist for LSTM with attention and BERT.

the declining relationship with $k$ does not hold. Our result on type-token ratio is opposite to **H2c**: the greater the type-token ratio, the higher the similarity (see supplementary materials). We believe that the reason is that type-token ratio is strongly negatively correlated with length (the Spearman correlation for Yelp, SST and deception dataset is -0.92, -0.59 and -0.84 respectively). In other words, type-to-token ratio becomes redundant to length and fails to capture text complexity beyond length.

## 7 Distribution of Important Features

Finally, we examine the distribution of important features obtained from different approaches. These results may partly explain our previously observed low similarity in feature importance.

**Important features show higher entropy using LSTM with attention and lower entropy with XGBoost (Figure 6).** As expected from **H3a**, LSTM with attention (the pink lines) are usually at the top (similar results for BERT in the supplementary material). Such a high entropy can contribute to the low similarity between LSTM with attention and other models. However, as the order in similarity between SVM and XGBoost is less stable, entropy cannot be the sole cause.

**Distribution of POS tags (Figure 7 and Figure 8).** We further examine the linguistic properties of important words. Consistent with **H3b**, adjectives are more important in sentiment classification than in deception detection. On the contrary to our hypothesis, we found that pronouns do not always play an important role in deception detection. Notably, LSTM with attention puts a strong emphasis on nouns in deception detection. In all cases, determiners are under-represented among important words. With respect to the distance of part-of-speech tag distributions between important features and all words (background), post-hoc methods tend to bring important words closer to the background words, which echoes the previous observation that post-hoc methods tend to increase the similarity between important words (Figure 8).

## 8 Concluding Discussion

In this work, we provide the first systematic characterization of feature importance obtained from different approaches. Our results show that different approaches can sometimes lead to very different important features, but there exist some consistent patterns between models and methods. For instance, deep learning models tend to generate diverse important features that are different from traditional models; post-hoc methods lead to more similar important features than built-in methods.

As important features are increasingly adopted for varying use cases (e.g., decision making vs. model debugging), we hope to encourage more work in understanding the space of important features, and how they should be used for different purposes. While we focus on consistent patterns across classification tasks, it is certainly interesting to investigate how properties related to tasks and data affect the findings. Another promising direction is to understand whether more concentrated important features (lower entropy) lead to better human performance in supporting decision making.

## Acknowledgments

We thank Jim Martin, anonymous reviewers, and members of the NLP+CSS research group at CU Boulder for their insightful comments and discussions. This work was supported in part by NSF grant IIS-1849931.

## References


Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*.

Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A scalable tree boosting system. In *Proceedings of KDD*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL*.

Finale Doshi-Velez and Been Kim. 2017. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*.

Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. 2019. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, 51(5):93.

Matthew Honnibal and Ines Montani. 2017. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing.

Benjamin D Horne, Dorit Nevo, John O'Donovan, Jin-Hee Cho, and Sibel Adali. 2019. Rating reliability and bias in news articles: Does ai assistance help everyone? In *Proceedings of ICWSM*.

Sarthak Jain and Byron C. Wallace. 2019. Attention is not explanation. In *Proceedings of NAACL*.

Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of ECML*.

Been Kim, Rajiv Khanna, and Oluwasanmi O Koyejo. 2016. Examples are not enough, learn to criticize! criticism for interpretability. In *Proceedings of NeurIPS*.

Vivian Lai and Chenhao Tan. 2019. On human predictions with explanations and predictions of machine learning models: A case study on deception detection. In *Proceedings of FAT**.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in nlp. In *Proceedings of NAACL*.

Zachary C Lipton. 2016. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490*.

Tania Lombrozo. 2012. Explanation and abductive inference. *Oxford handbook of thinking and reasoning*, pages 260–276.

Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Proceedings of NeurIPS*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of EMNLP*.

Dmitry M Malioutov, Kush R Varshney, Amin Emad, and Sanjeeb Dash. 2017. Learning interpretable classification rules with boolean compressed sensing. In *Transparent Data Mining for Big and Small Data*, pages 95–121. Springer.

Dong Nguyen. 2018. Comparing automatic and human evaluation of local explanations for text classification. In *Proceedings of NAACL*.

Myle Ott, Claire Cardie, and Jeffrey T Hancock. 2013. Negative deceptive opinion spam. In *Proceedings of NAACL*.

Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T Hancock. 2011. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of ACL*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of KDD*.

Sofia Serrano and Noah A. Smith. 2019. Is Attention Interpretable? In *Proceedings of ACL*.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017. Learning important features through propagating activation differences. In *Proceedings of ICML*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of NeurIPS*.

Jesse Vig. 2019. Deconstructing BERT: Distilling 6 Patterns from 100 Million Parameters. https://towardsdatascience.com/\deconstructing-bert-distilling-6-patterns-from\-100-million-parameters-b49113672f77. [Online; accessed 27-Apr-2019].

Sarah Wiegreffe and Yuval Pinter. 2019. Attention is not not Explanation. In *Proceedings of EMNLP*.

Thomas Wolf. 2019. huggingface/pytorch-pretrained-bert. https://github.com/huggingface/pytorch-pretrained-BERT.

Yelp. 2019. Yelp dataset 2019. https://www.yelp.com/dataset. Accessed: 2019-01-01.

# Enhancing Local Feature Extraction with Global Representation for Neural Text Classification

**Guocheng Niu[1*], Hengru Xu[2*†], Bolei He[1], Xinyan Xiao[1], Hua Wu[1], Sheng Gao[2]**

[1] Baidu Inc., Beijing, China
[2] Beijing University of Posts and Telecommunications
[1]{niuguocheng, hebolei, xiaoxinyan, wu_hua}@baidu.com
[2]{xuhengru, gaosheng}@bupt.edu.cn

## Abstract

For text classification, traditional local feature driven models learn long dependency by deeply stacking or hybrid modeling. This paper proposes a novel Encoder1-Encoder2 architecture, where global information is incorporated into the procedure of local feature extraction from scratch. In particular, Encoder1 serves as a global information provider, while Encoder2 performs as a local feature extractor and is directly fed into the classifier. Meanwhile, two modes are also designed for their interactions. Thanks to the awareness of global information, our method is able to learn better instance specific local features and thus avoids complicated upper operations. Experiments conducted on eight benchmark datasets demonstrate that our proposed architecture promotes local feature driven models by a substantial margin and outperforms the previous best models in the fully-supervised setting.

## 1 Introduction

Text classification is a fundamental task in natural language processing, which is widely used in various applications such as spam detection, sentiment analysis and topic classification. One of the mainstream approaches firstly utilizes explicit local extractors to identity key local patterns and classifies based on them afterwards. In this paper, we call this line of research as *local feature driven models*.

Lots of proposed methods can be grouped into this scope. Ngrams have been traditionally exploited in statistical machine learning approaches (Pang et al., 2002; Wang and Manning, 2012). For deep neural networks, encoding local features into low-dimensional distributed ngrams embeddings (Joulin et al., 2016; Qiao et al., 2018) and simply bagging of them have been proved effective and highly efficient. Convolutional Neural Networks (CNN) (LeCun et al., 2010) are promising methods for their strong capacities in capturing local invariant regularities (Kim, 2014). More recently, Wang (2018) proposes the Disconnected Recurrent Neural Network (DRNN), which utilizes RNN to extract local features for larger windows and has reported best results on several benchmarks.

| |
|---|
| Case1: ***Apple*** *is really amazing! I am fed up to carry my clunky camera.* |
| Case2: ***Apple*** *is famous around world and deserves to be called "nutritional powerhouses".* |

Table 1: Topic classification examples for *Technology* and *Health*, where ***Apple*** is ambiguous within local context.

Despite having good interpretability and remarkable performance, current local feature extraction still has one shortcoming. As shown in Table 1, the real meaning of *Apple* can only be correctly recognized from overall view instead of narrow window. If the local extractor in charge of *Apple* cannot receive *camera* and *nutritional* from the very beginning, it would require complicated and costly upper structures to help revise the imprecisely local representation and create newer high-level features, such as deeply stacking (Johnson and Zhang, 2017; Conneau et al., 2016) and hybrid integration (Xiao and Cho, 2016). To a certain extend, it is inefficient and hard to train especially in the case of insufficient corpus.

To address this issue, we believe a more efficient approach is to optimize the local extraction process directly. In this paper, we propose

*These authors contributed equally to this work.
†This work was done while the author was an intern at Baidu Inc.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 496–506, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

a novel architecture named Encoder1-Encoder2[1], which innovatively contains two encoders for the identical input sequence respectively, instead of using only one single encoder in previous work. Concretely, the Encoder1 can be any kind of neural network models designed for briefly grasping global background, while the Encoder2 should be a typical local feature driven model. The key point is, the earlier generated global representations from Encoder1 is then incorporated into the local extraction procedure of Encoder2. In this way, local extractors can notice more long-range information on the basis of its natural advantages. As a result, better instance specific local features can be captured and directly utilized for classification owing to global awareness, which means further upper complicated operations can be avoided.

We conduct experiments on eight public text classification datasets introduced by Zhang et al. (2015). The experimental results show that our proposed architecture promotes local feature driven models by a substantial margin. In fully-supervised settings, our best models achieves new state-of-the-art performances on all benchmark datasets. We further demonstrate the ability and generalization of our architecture in the semi-supervised domain.

Our contributions can be concluded as follows:

1. We propose a novel Encoder1-Encoder2 architecture, where better instance specific local features are captured by incorporating global representations into local extraction procedure.

2. Our architecture has great flexibility. Different associations among Encoder1, Encoder2 and Interaction Modes are studied, where any kind of combination promotes vanilla CNN or DRNN significantly.

3. Our architecture is more robust to the window size of local extractors and the corpus scale.

## 2 Related Work

**Local Feature Driven Models** FastText uses bag of n-grams embeddings as text representation (Joulin et al., 2016), which has been proved effective and efficient. Qiao et al. (2018) propose a new method of learning and utilizing task specific n-grams embeddings to conquer data sparsity.

CNN (LeCun et al., 2010) are representative methods of this category. Convolution operators are performed at every window based location to extract local features, interleaved with pooling layer for capturing invariant regularities. From Kim (2014), CNN are widely used in text classification. In addition to shallow structure, very deep and more complex CNN based models have also been studied to establish long distance association. Examples are deep character-level CNNs Zhang et al. (2015); Conneau et al. (2016), deep pyramid CNN Johnson and Zhang (2017) and convolution-recurrent networks Xiao and Cho (2016), in which recurrent layers are designed on top of convolutional layers for learning long-term dependencies between local features.

CNN use simple linear operations on n-gram vectors of each window, which enlightens researchers to capture higher order local non-linear feature using RNN. Shi et al. (2016) first replace convolution filters with LSTM for query classification. Wang (2018) proposes DRNN, which exploits large window size equipped with GRU.

To make full use of local and global information, Zhao et al. (2018) propose a sandwich network by carding a CNN in the middle of two LSTM layers, where the output of CNN provides local semantic representations while the top LSTM supplies global structure representations. However, the global information they mainly focus on is the syntax part, which is produced by reorganizing the already obtained local features. Besides, both of them are directly used for final classification, while we use pre-acquired global representations to help capture better local features. To the best of our knowledge, we are the first to incorporate global representation into the extraction procedure of local features for text classification.

**Other Neural Network Models** Recurrent Neural Networks (RNN) are naturally good at modeling variable-length sequential data and capturing long-term dependencies (Hochreiter and Schmidhuber, 1997; Chung et al., 2014). Global features are encoded by semantically synthesizing each word in the sequence in turn and there is no explicit small regions feature extraction procedure in this process. Lai et al. (2015) equip RNN with max-pooling to tackle the bias problem where later words are more dominant than earlier words. Tang et al. (2015) utilize LSTM to encode semantics of sentences and their relations in doc-

[1]Our code will be available at https://github.com/PaddlePaddle/models/tree/develop/PaddleNLP/Research/EMNLP2019-GELE. "GELE" is the abbreviation for Global Encoder and Local Encoder, i.e., Encoder1 and Encoder2 respectively.

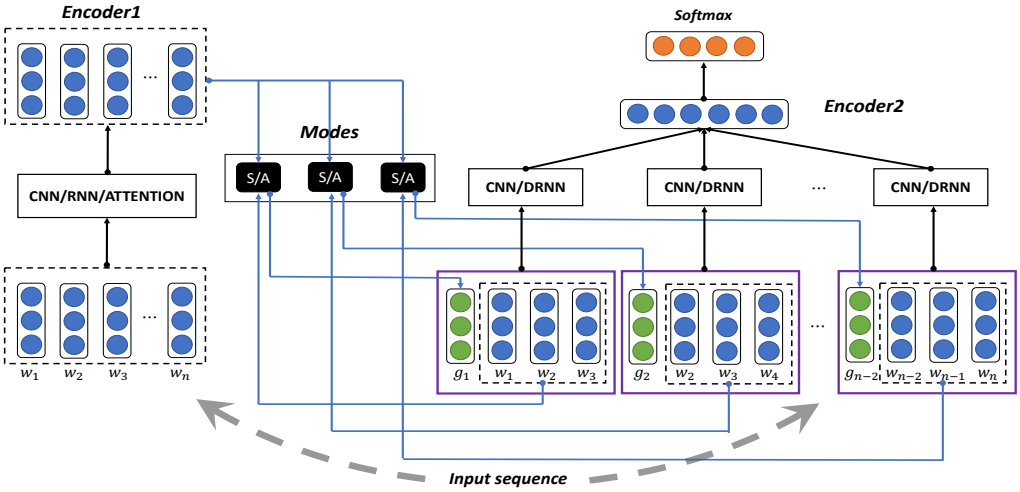


Figure 1: Encoder1-Encoder2 architecture mainly contains three components. (1) Encoder1 serves as a global information provider. (2) Encoder2 is a local feature driven model whose output is directly fed into the classifier. (3) Mode is the interaction manner between them. S and A are abbreviation of SAME and ATTEND respectively.

ument representation. Tai et al. (2015) introduce a tree-structured LSTM for sentiment classification.

The attention mechanism proposed by Bahdanau et al. (2014) has achieved great success in machine translation (Vaswani et al., 2017). For text classification which only has single input sequence, attention based models mainly focus on applying attention mechanism on top of CNN or RNN for selecting the more important information (Yang et al., 2016; Er et al., 2016). Letarte et al. (2018) and Shen et al. (2018) also explore self-attention networks which is CNN/RNN free.

## 3 Encoder1-Encoder2 Architecture

### 3.1 Overview

In this paper, we propose a novel neural network architecture named Encoder1-Encoder2 for text classification, which is illustrated in Figure 1. The identical input sequence will be encoded twice by two encoders respectively, but only the output of Encoder2 is used directly for the classifier. In particular, the Encoder1 serves as a pioneer for providing global information, while the Encoder2 focuses on extracting better local features by incorporating the former into the local extraction procedure. Besides, two Interaction Modes are developed for more targeted absorption of global information.

### 3.2 Encoder1: Global Information Provider

Without loss of generality, we introduce three types of models for Encoder1 in our architecture, each of which can be an independent global information provider and they are compared in our experiments.

**CNN** Let $\mathbf{x}_t$ be the $d$-dimensional word vector corresponding to the $t$-th word in a sequence of length $n$, $\mathbf{x}_{t-h+1:t}$ refers to the concatenation of words $\mathbf{x}_{t-h+1}, \mathbf{x}_{t-h+2}, \ldots, \mathbf{x}_t$ with size $h$ and $k$ number of filters are applied to the input sequence to generate features. Formally, filters $\mathbf{W}_f$ are applied to window $\mathbf{x}_{t-h+1:t}$ to compute $\mathbf{h}_t$:

$$\mathbf{h}_t = Conv(\mathbf{x}_{t-h+1}, \mathbf{x}_{t-h+2}, \ldots, \mathbf{x}_t) \quad (1)$$

$$= relu(\mathbf{W}_f \mathbf{x}_{t-h+1:t} + \mathbf{b}_f) \quad (2)$$

By same padding, filters are applied to $n$ possible windows in the sequence and the global representation can be represented as $\mathbf{enc}_1$:

$$\mathbf{enc}_1 = [\mathbf{h}_1; \mathbf{h}_2; \ldots; \mathbf{h}_n] \quad (3)$$

**GRU** Gated recurrent units (GRU) are a gating mechanism in RNN (Cho et al., 2014). Two types of gates are used in GRU: reset gate decides how much new information is updated, while update gate controls the flow of previous information. The hidden state $\mathbf{h}_t$ is computed iteratively based on $\mathbf{h}_{t-1}$ and $\mathbf{x}_t$. As a result, the all previous information can be encoded. For saving space, here we abbreviate it as:

$$\mathbf{h}_t = GRU(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t) \quad (4)$$

The global representation produced by GRU is hidden states of all time steps:

$$\mathbf{enc}_1 = [\mathbf{h}_1; \mathbf{h}_2; \ldots; \mathbf{h}_n] \quad (5)$$

**Attention** We also introduce attention mechanism on GRU for enhancing valuable information following Zhou et al. (2016). Define a context vector $\mathbf{u}_w$ to measure the importance of each hidden state $\mathbf{h}_t$ in GRU, which is randomly initialized and learned during training. A normalized importance weight $\alpha_t$ is obtained through a softmax function:

$$\alpha_t = \frac{\exp(\tanh(\mathbf{h}_t)^\top \mathbf{u}_w)}{\sum_t \exp(\tanh(\mathbf{h}_t)^\top \mathbf{u}_w)} \quad (6)$$

The global representation produced by this attention mechanism is expressed as:

$$\mathbf{enc}_1 = [\alpha_1 \mathbf{h}_1; \alpha_2 \mathbf{h}_2; \ldots; \alpha_n \mathbf{h}_n] \quad (7)$$

### 3.3 Encoder2: Variant Local Extractor

Vanilla local feature extractor strictly focuses on a limited size region. Here we propose a variant method. Apart from the expected local context, global information distilled by Encoder1 is also absorbed by a local extractor. In this way, the local features extracted by Encoder2 can notice the global background while still maintaining the position-invariant local patterns.

For Encoder2, we introduce two kinds of local feature driven models, i.e., CNN and DRNN. The former is good at capturing local spatial structure, while the latter is highlighted in capturing local temporal part. Set $\mathbf{g_t}$ as the required global information for a certain size window starting from $\mathbf{x}_t$, which will be introduced in 3.4 in detail.

**CNN** Here we treat each $\mathbf{g_t} \in \mathbb{R}^d$ as a faked extra global word, and do convolution with window words together. Based on Equation 1, features produced by filters for window $\mathbf{x}_{t-h+1:t}$ can be represented as:

$$\mathbf{h}_t = Conv(\mathbf{g_t}, \mathbf{x}_{t-h+1}, \mathbf{x}_{t-h+2}, \ldots, \mathbf{x}_t) \quad (8)$$

**DRNN** Different from CNN, DRNN utilizes RNN to extract local features for each window (Wang, 2018). To introduce global information into DRNN, faked global word $\mathbf{g_t}$ is filled in the head of each window like CNN does. Because of the sequential nature of RNN, even for a limited window, global information can be encoded into RNN from scratch and motivate the latter words. Here we use GRU as the local feature extractor, and features produced for window $\mathbf{x}_{t-h+1:t}$ can be represented as:

$$\mathbf{h}_t = GRU(\mathbf{g_t}, \mathbf{x}_{t-h+1}, \mathbf{x}_{t-h+2}, \ldots, \mathbf{x}_t) \quad (9)$$

To maintain translation invariant, a max-over-time pooling layer is then applied to CNN or DRNN layer, the pooling result is regarded as the output of Encoder2:

$$\mathbf{enc}_2 = maxpool([\mathbf{h}_1; \mathbf{h}_2; \ldots; \mathbf{h}_n]) \quad (10)$$

### 3.4 Interaction Modes between Encoders

Set $\mathbf{enc}_1$ as the global representation produced by Encoder1, required information for a certain window $\mathbf{x}_{t-h+1:t}$ with size $h$ is defined as $\mathbf{g_t}$:

$$\mathbf{g_t} = G(\mathbf{enc}_1, \mathbf{x}_{t-h+1}, \mathbf{x}_{t-h+2}, \ldots, \mathbf{x}_t) \quad (11)$$

where $G$ is a function of interaction mode. Two modes are devised from different point of views.

**SAME** Treat $\mathbf{enc}_1$ as a "reference book" provided by Encoder1. The basic idea of SAME Mode is each window in Encoder2 will get indiscriminate guidance regardless of the local information itself. For this purpose, max-over-time pooling is operated on $\mathbf{enc}_1$ directly to extract the most important information:

$$\hat{\mathbf{g}}_\mathbf{t} = maxpool(\mathbf{enc}_1) \quad (12)$$

**ATTEND** Mode ATTEND utilizes global information from another perspective. According to different local contexts, the guidances from Encoder1 can be be more targeted. Specifically, we use attention mechanism.

For window $\mathbf{x}_{t:t+h-1}$ with size $h$, the context vector is the average pooling of local words embeddings and the importance weight $\alpha_t$ for each hidden state $\mathbf{h}_t$ in Encoder1 can be computed as:

$$\alpha_t = \frac{\exp(\tanh(\mathbf{h}_t)^\top avg(\mathbf{x}_{t:t+h-1}))}{\sum_t \exp(\tanh(\mathbf{h}_t)^\top avg(\mathbf{x}_{t:t+h-1}))} \quad (13)$$

To maximize the profits obtained from Encoder1, we concatenate both of maxpooling results and attention results. Then $\hat{\mathbf{g}}_\mathbf{t}$ in ATTEND mode can be represented as:

$$\hat{\mathbf{g}}_\mathbf{t} = Concat(maxpool(\mathbf{enc}_1), \sum_t \alpha_t \mathbf{h}_t) \quad (14)$$

Finally, to keep consistent dimensions with words in the text, we compress $\hat{\mathbf{g}}_\mathbf{t}$ using MLP and formalized as $\mathbf{g_t}$, which can be easily embedded into the local feature extraction in Encoder2.

| Dataset | C | L | N | Test | Tasks |
|---|---|---|---|---|---|
| Yelp P. | 2 | 156 | 560k | 38k | SA |
| Yelp F. | 5 | 158 | 650k | 50k | SA |
| Amz. P. | 2 | 91 | 3M | 650k | SA |
| Amz. F | 5 | 93 | 3.6M | 400k | SA |
| AG | 4 | 44 | 120k | 7.6k | News |
| Sogou | 5 | 579 | 450k | 60k | News |
| Yah. A. | 10 | 112 | 1.4M | 60k | QA |
| DBP | 14 | 55 | 560k | 70k | Ontology |

Table 2: Datasets summary. C: Number of target classes. L: Average sentence length. N: Dataset size. Test: Test set size. In tasks, SA refers to sentiment analysis, and QA refers to question answering.

| Corpus | Vocabulary | Length | Window size |
|---|---|---|---|
| Yelp P. | 150k | 300 | 20 |
| Yelp F. | 200k | 300 | 20 |
| Amz. P. | 400k | 256 | 15 |
| Amz .F. | 400k | 256 | 15 |
| AG | 50k | 128 | 15 |
| Sogou | 56k | 400 | 15 |
| Yah. A. | 400k | 256 | 20 |
| DBP | 400k | 128 | 15 |

Table 3: Model settings. We limit the vocabulary size and set maximum sequence length. We also show the window size in DRNN following Wang (2018).

### 3.5 Classification Layer

After incorporating the global information obtained from Encoder1 into the local feature extraction of Encoder2, the output vector of latter can be regarded as the representation of the entire text. The vector is then fed into a softmax classifier to predict the probability of each category and cross entropy is used as loss function:

$$\hat{y} = softmax(\mathbf{W}_c\mathbf{enc}_2 + \mathbf{b}_c) \quad (15)$$

$$H(y, \hat{y}) = \sum_i y_i \log \hat{y}_i \quad (16)$$

where $\hat{y}_i$ is the predicted probability and $y_i$ is the true probability of class $i$.

## 4 Experiments

We report experiments with proposed models in comparison with previous methods.

### 4.1 Experiments Settings

**Datasets** Publicly available datasets from Zhang et al. (2015) are used to evaluate our models. These datasets contain various domains and sizes, corresponding to sentiment analysis, news classification, question answering, and ontology extraction, which are summarized in Table 2.

**Model Settings** For data preprocessing, all the texts of datasets are tokenized by NLTKs tokenizer (Loper and Bird, 2002). For model hyperparameters, notations are following in 3.2 and 3.3. We adopt CNN, GRU and Attention for Encoder1. In CNN (Encoder1), we use filter windows ($h$) of $[3, 5, 7]$ with 128 feature maps ($k$) each. The hidden unit number is 128 in GRU (Encoder1) and Attention (Encoder1). For Encoder2, we conduct experiments based on two types of local feature extractor, corresponding to CNN and DRNN respectively. In CNN (Encoder2), window sizes of filters are of $[3, 5, 7]$ with 128 feature maps each. In DRNN (Encoder2), all the dimensions of hidden states are set to 300. Other settings are shown in Table 3, all trainable parameters including embeddings of words are initialized randomly without any pre-trained techniques (Mikolov et al., 2013; Peters et al., 2018; Devlin et al., 2018).

**Training and Validation** For each dataset, we randomly split the full training corpus into training and validation set, where the validation size is the same as the corresponding test size. Then the validation set is fixed for all models for fair comparison. The reported test accuracy is evaluated in the model which has lowest validation error.

AdaDelta (Zeiler, 2012) with $\rho = 0.95$ and $\epsilon = 1e - 6$ is chosen to optimize all the trainable parameters. Gradient norm clipping is employed to avoid the gradient explosion problem. L2 normalization is used in all models which include RNN structures. The batch size is set to 64 for Yelp P. and Yelp F. while 128 for other datasets. We train all the models using early stopping with timedelay 10.

### 4.2 Experimental Results and Analysis

Table 4 is the summary of the experimental results. We use underscores to represent the best published models, and bold the best records. Best models in our proposed architecture beat previous state-of-the-art models on all eight text classification benchmarks.

For published models, best results are achieved almost all by local feature driven models including Region-emb, VDCNN and DRNN. Self-Attention model SANet performs well, but does not achieve advantageous results as in sequence to sequence

| Model | Yelp P. | Yelp F. | Amz. P. | Amz. F. | AG | Sogou | Yah. A. | DBP |
|---|---|---|---|---|---|---|---|---|
| bigram-FastText (Joulin et al., 2016) | 95.7 | 63.9 | 94.6 | 60.2 | 92.5 | 96.8 | 72.3 | 98.6 |
| Region-emb (Qiao et al., 2018) | 96.2 | 64.5 | 95.3 | 60.8 | 92.8 | 97.3 | 73.4 | 98.9 |
| SANet(big) (Letarte et al., 2018) | 95.2 | 64.0 | 95.5 | 61.3 | 92.6 | - | 74.1 | 98.8 |
| LSTM (Zhang et al., 2015) | 94.7 | 58.2 | 93.9 | 59.4 | 86.1 | 95.2 | 70.8 | 98.6 |
| D-LSTM (Yogatama et al., 2017) | 92.6 | 59.6 | - | - | 92.1 | 94.9 | 73.7 | 98.7 |
| char-CNN (Zhang et al., 2015) | 94.7 | 62.0 | 94.5 | 59.6 | 87.2 | 95.1 | 71.2 | 98.3 |
| VDCNN (Conneau et al., 2016) | 95.7 | 64.7 | 95.7 | 63.0 | 91.3 | 96.8 | 73.4 | 98.7 |
| CNN (Kim, 2014) | 95.8 | 64.7 | 95.2 | 60.9 | 91.9 | 97.1 | 72.6 | 98.8 |
| *Encoder1*-CNN-S (Ours) | 96.5 | 66.2 | 95.9 | **63.3** | 92.5 | **97.5** | 74.5 | 98.8 |
| *Encoder1*-CNN-A (Ours) | 96.5 | 66.5 | **96.0** | **63.3** | 93.0 | **97.5** | 74.6 | 98.9 |
| DRNN (Wang, 2018) | 96.3 | 66.4 | 95.6 | 63.0 | 92.9 | 96.9 | 74.3 | 98.9 |
| *Encoder1*-DRNN-S (Ours) | 96.6 | 66.8 | **96.0** | 63.2 | 93.0 | 97.2 | 74.8 | **99.0** |
| *Encoder1*-DRNN-A (Ours) | **96.7** | **67.0** | **96.0** | 63.1 | **93.2** | 97.3 | **75.0** | **99.0** |

Table 4: Test accuracy [%] on several Datasets. We use Encoder1-Encoder2-Mode to represent our architecture, where S indicates mode SAME while A indicates ATTEND. For each specialized Encoder2, we test three different Encoder1. For the sake of brevity, *Encoder1* here does not specifically refer to a certain model but the best performance of the three combinations. Detailed experiments results for all combinations of Encoder1-Encoder2-Mode are further reported and compared in Table 5. For compared previous models, first block lists n-grams based models including bigram-FastText (Joulin et al., 2016) and region embedding (Qiao et al., 2018). Self-attention Networks SANet (Letarte et al., 2018) is reported in the second block. RNN based models LSTM (Zhang et al., 2015), D-LSTM (Yogatama et al., 2017) and CNN based models char-CNN (Zhang et al., 2015) and VDCNN (Conneau et al., 2016) are listed in third and forth block respectively. Strong local feature driven models CNN (Kim, 2014) and DRNN (Wang, 2018) are chosen as base model and directly compared with our architecture in last two blocks.

scenes, neither do RNN based methods. We argue that it is because key phrases and word order play an important role in text classification.

For our models, the experimental results show that enhanced local extractors with global encoder outperform vanilla local models by a advantageous margin. When CNN is chosen as local extractor, the performance gains are particularly significant for relatively difficult tasks such as Amz. F.(+2.4%) and Yah. A.(+2.0%). Encoder1-CNN performs even better than VDCNN with 29 convolutional layers. The results are satisfying considering that our CNN used as local extractor here is a shallow model with only one layer. Moreover, complicated VDCNN performs best among published models on larger datasets Amz. P.(95.7%) and Amz. F.(63.0%) but not as expected on smaller AG(91.3%), while our Encoder1-CNN has stable superior performance on all datasets. When DRNN is chosen as local extractor, the bonus from the global encoder is not so big like CNN, but still considerable and stable. Encoder1-DRNN beats DRNN on all datasets with a highest gain up to 0.7%.

To better analyze the impact of specific Encoder1 (global encoder) and different Interaction Modes on architecture performance, Table 5 details all combinations results of Encoder1-Encoder2-Mode on three datasets. We find the local extractor benefits quite a lot for any introduced global encoders. Overall, RNN and Attention based global encoders perform well-matched for local extractor, and both of them often perform better than CNN based global encoder. For example, Attention-CNN wins CNN-CNN 1.0% on Yelp F. and RNN-DRNN wins CNN-DRNN 0.4% on Yah. A. This is intuitive since RNN and Attention are more appropriate in capturing global information compared with CNN, which is critical for local extractor. The structures which specialize in modeling long-term dependency are more recommended as the global encoder.

For two Interaction Modes, we find ATTEND performs slightly better than SAME up to 0.4%, which can verify the differentiated motivation. Encoder1 (global encoder) can be viewed as a "reference book" about the whole text. Two Modes utilize the information from different perspectives to

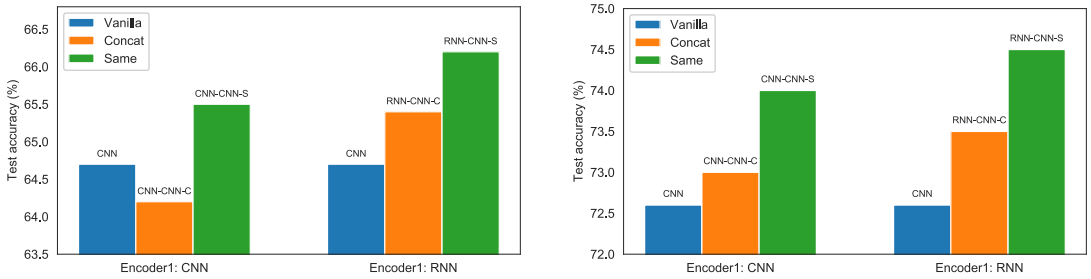


(a) Experiments on Yelp F.

(b) Experiments on Yah. A.

Figure 2: Model ablation experiments. "Vanilla" is the traditional CNN (Kim, 2014). "Concat" simply concatenates the output of Encoder1 and Encoder2 and directly classifies according to the concatenated result. "Same" indicates the Interaction Mode in our architecture.

| Model | Yelp P. | Yelp F. | Yah. A. |
|---|---|---|---|
| CNN | 95.8 | 64.7 | 72.6 |
| CNN-CNN-S | 96.0 | **65.5** | 73.9 |
| CNN-CNN-A | **96.3** | **65.5** | **74.1** |
| RNN-CNN-S | **96.5** | 66.2 | 74.5 |
| RNN-CNN-A | **96.5** | **66.4** | **74.6** |
| Attention-CNN-S | 96.4 | 66.2 | **74.2** |
| Attention-CNN-A | **96.5** | **66.5** | 74.1 |
| DRNN | 96.3 | 66.4 | 74.3 |
| CNN-DRNN-S | 96.5 | 66.6 | **74.6** |
| CNN-DRNN-A | **96.7** | **67.0** | 74.4 |
| RNN-DRNN-S | **96.6** | 66.6 | 74.7 |
| RNN-DRNN-A | **96.6** | **66.9** | **75.0** |
| Attention-DRNN-S | **96.6** | 66.8 | **74.9** |
| Attention-DRNN-A | **96.6** | **67.0** | **74.9** |

Table 5: Effect of Encoder1 and Interaction Mode.

assist the local extractor. SAME Mode selects the most important information of global encoder and provides same guidance for each window in Encoder2, while the ATTEND Mode tends to make use of the "reference" with purpose based on different local contexts as if we refer to a reference book with initiative questions.

### 4.3 Model Ablation

In addition to introducing another encoder into vanilla local feature driven models, the greatest novelty of our architecture lies in that the global encoding is used to generate local features directly. Based on this motivation, the local features have the global awareness from the very beginning. To verify that our novel architecture makes key contribution to the performance improvement, we carry out model ablation experiments.

Without loss of generality, we use CNN as local extractor here and validate on Yelp F. and Yah. A. datasets. The results are illustrated in Figure 2.

Firstly, we list the results of Vanilla CNN, which is regarded as the most primitive state. Secondly, another additional encoder is introduced but they both process inputs independently and then their output representations are concatenated for classification. We call it "Concat", abbreviated as "C". For example, RNN-CNN-C stands for concatenating another RNN. Finally, we upgrade the way to use the introduced encoder as our proposed architecture. Here we list Mode SAME, abbreviated as "S".

We find CNN-CNN-C loses 0.5% on Yelp P. but wins 0.4% on Yah. A. compared with vanilla CNN. CNN-CNN-C can be viewed as doubling convolution filters and we can observe that introducing more parameters does not always perform better. Meanwhile, RNN-CNN-C wins vanilla CNN 0.7% on Yelp P. and 0.9% on Yah. A. It makes sense since the classifier could use features from CNN and RNN simultaneously and different model structures complement each other for classification. In particular, our architecture performs best for both cases. CNN-CNN-S wins CNN-CNN-C 1.3% and 0.8%, and RNN-CNN-S wins RNN-CNN-C 0.6% and 1.0% on Yelp P. and Yah. A. respectively. In fact, CNN-CNN-S does not introduce new model structure or complicated operations and the number of parameters are almost the same. We attribute the great improvement to our novel mechanism where the global representation conduces to the local extraction.

### 4.4 Effect of Window Size

As an important hyperparameter, window size determines how much information can be seen in a specific window and often requires carefully tun-

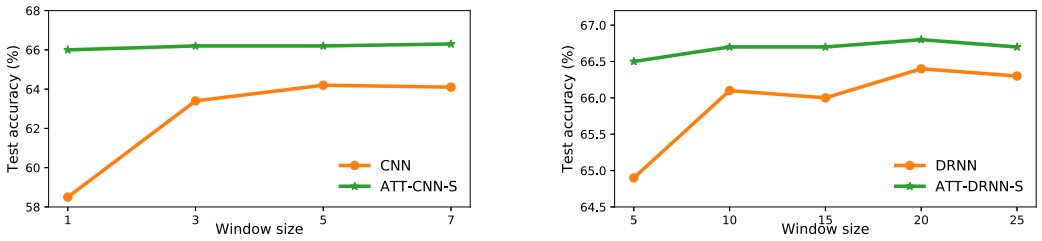


(a) The impact of window size on CNN.

(b) The impact of window size on DRNN.

Figure 3: Window size experiments on Yelp F. ATT is the abbreviation of Attention.

| Model | Sentence Samples |
|---|---|
| CNN | *"Commission backs 5bn British Energy deal "" British Energy, the nuclear generator yesterday welcomed a decision by the European commission to approve a government-backed 5bn rescue plan ."* **World ✗** |
| ATT-CNN | *"Commission backs 5bn British Energy deal "" British Energy, the nuclear generator, yesterday welcomed a decision by the European commission to approve a government-backed 5bn rescue plan ."* **Business ✓** |
| CNN | *The mac and cheese sticks were amazing ... highly recommend them . Overall, for the high price price pay here, I would rather be across the casino with at least a great view of the fountains.* **Positive ✗** |
| ATT-CNN | *The mac and cheese sticks were amazing ... highly recommend them . Overall, for the high price you pay here, I would rather be across the casino with at least a great view of the fountains.* **Negative ✓** |

Table 6: Visualization of chosen samples on AG News and Yelp Review Polarity dataset. We use SAME Interaction Mode in ATT-CNN where ATT is the abbreviation of Attention.

ing in traditional method. Small window sizes may result in the loss of some critical information whereas large windows result in an enormous parameter space, which could be difficult to train (Lai et al., 2015). In this section, we analyze the impact of different window sizes on model performances.

As shown in Figure 3, both CNN and DRNN are very sensitive to window size, the optimal window size in DRNN can be much larger than CNN due to the sequential memory in RNN structure. Tuning these models is often challenging. In contrast, our Encoder1-Encoder2 architecture is insensitive to the parameter and achieves stable satisfied performance in various window sizes. We believe it is because the local extraction has been enhanced by global information and not strictly dependent on large windows to capture long range information.

### 4.5 Case Study and Visualization

To investigate how our architecture makes a difference in details, we visualize the attending phrases by the neural model in Table 6. Qualitatively, we display the contribution of phrases in Encoder2 to classification via max-pooling. The most important phrases are highlighted red where the intensity of the color indicates the contribution. Meanwhile, we use waves to roughly indicate the key phrases with high attention scores in Encoder1. Detailed visualization techniques have been introduced in Li et al. (2015).

The first two lines compare CNN with our Attention-CNN on an example from AG News. CNN wrongly captures key phrases *British Energy* and *the nuclear generator* and thus misclassifies the example into *World*. In contrast, our Attention-CNN is able to correctly classify it into *Business*. The Encoder1 firstly captures the global description by *Energy deal*, *nuclear*, and *rescue plan*. Informed with these global information, Encoder2 reduces its attention to *nuclear*, which implies label *World* while captures key phrases *British Energy deal* and *5bn rescue plan*. Accordingly the model makes a correct prediction labeled as *Business*. For the second example, the global representations include phrases *high price* and conjunction *Overall*, making Encoder2 activate *I would rather* while reduce its sensitivity to *highly recommend them* compared with CNN.

In short, the global representations learned by Encoder1 provide a brief overall grasp of the whole text, which includes both semantic and structure information. It effectively helps Encoder2 capture better instance specific local features and improve model performance.

### 4.6 Comparison to Pre-trained Models

As shown above, our paper mainly focuses on fully-supervised domain where all model parameters are trained from scratch. Alternatively, substantial work has shown that pre-trained models are beneficial for various NLP tasks. Typically, they first pre-train neural networks on large-scale unlabeled text corpora, and then finetune the models or representations on downstream tasks.

One kind of pre-trained models is the word embeddings, such as word2vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014). More recently, by utilizing larger-scale unsupervised corpus and deeper architecture, pre-trained language models have shown to be effective in learning common language representations and have achieved great success. Among them, OpenAI GPT (Radford et al., 2018), BERT (Devlin et al., 2018), XLNet (Yang et al., 2019) and ERNIE 2.0 (Sun et al., 2019) are the most remarkable examples.

In this section, we generalize our architecture to semi-supervised domain which is equipped with pre-trained word embeddings and then compare with popular pre-trained based models. Specifically, we use GloVe vectors[2] with 300 dimensions to initialize the word embeddings in our architecture. BERT$_{BASE}$[3] and ERNIE 2.0$_{BASE}$[4] with 12-layer Transformer (Vaswani et al., 2017) are chosen for comparison. Here we report best model for each specialized Encoder2 with SAME Mode. Results on three datasets are listed in Table 7.

Overall, our architecture can be further boosted a lot by utilizing pre-trained word embeddings. For example, *Encoder1*-DRNN-S obtains a new score of 76.2%(+1.4%) on Yah. A. and *Encoder1*-CNN-S gets 94.1%(+1.6%) on AG. Vanilla local extractors also achieve better performance as expected in most instances while our models are still much better than them. *Encoder1*-CNN-S outperforms CNN by 0.9%, 1.8% and 2.0% on three datasets respectively, and *Encoder1*-DRNN-S outperforms DRNN by 0.4%, 0.6% and 0.7%. It shows that our architecture is well generalized and compatible with pre-training techniques.

It is interesting to compare with stronger pre-trained models. Although we obtain close scores on AG, BERT and ERNIE 2.0 indeed achieve more advanced results on others and the latter performs best on all three datasets. Despite their superb accuracy, we argue that the huge models are resource-hungry in practice. Lightweight models still have advantages under some circumstances such as limited memory, longer text data to be processed and requirements of faster inference time.

| Model | Yelp F. | AG | Yah. A. |
|---|---|---|---|
| CNN(n/y) | 64.7/64.5 | 91.9/92.3 | 72.6/73.7 |
| *Encoder1*-CNN-S(n/y) | 66.2/66.6 | 92.5/94.1 | 74.5/75.7 |
| DRNN(n/y) | 66.4/66.8 | 92.9/93.6 | 74.3/75.5 |
| *Encoder1*-DRNN-S(n/y) | 66.8/67.2 | 93.0/94.2 | 74.8/76.2 |
| BERT$_{BASE}$ | 67.9 | 94.2 | 76.4 |
| ERNIE 2.0$_{BASE}$ | **69.1** | **94.3** | **77.0** |

Table 7: Semi-supervised generalization of our architecture and comparison with popular pre-trained models. Here "n" and "y" stand for initializing word embeddings randomly and with pre-trained GloVe vectors separately.

## 5 Conclusion

In this work, we demonstrate the local feature extraction can be significantly enhanced with global information. Instead of traditionally exploiting deeper and complicated operations in upper neural layers, our work innovatively provides another lightweight way for improving the ability of neural model. Specifically, we propose a novel architecture named Encoder1-Encoder2 with two Interaction Modes for their interacting. The architecture has high flexibility and our best models achieve new state-of-the-art performance in fully-supervised setting on all benchmark datasets. We also find that our architecture is insensitive to window size and enjoy a better robustness. In future work, we plan to validate its effectiveness for multi-label classification. Besides, we are interested in incorporating more powerful unsupervised methods into our architecture.

## Acknowledgments

This work is supported by the Natural Science Foundation of China (No. 61533018). We gratefully thank the anonymous reviewers for their insightful comments.

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly

[2]http://nlp.stanford.edu/projects/glove
[3]https://github.com/google-research/bert
[4]https://github.com/PaddlePaddle/ERNIE

learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2016. Very deep convolutional networks for text classification. *arXiv preprint arXiv:1606.01781*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Meng Joo Er, Yong Zhang, Ning Wang, and Mahardhika Pratama. 2016. Attention pooling-based convolutional neural network for sentence modelling. *Information Sciences*, 373:388–403.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Rie Johnson and Tong Zhang. 2017. Deep pyramid convolutional neural networks for text categorization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 562–570.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Twenty-ninth AAAI conference on artificial intelligence*.

Yann LeCun, Koray Kavukcuoglu, and Clément Farabet. 2010. Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 253–256. IEEE.

Gaël Letarte, Frédérik Paradis, Philippe Giguère, and François Laviolette. 2018. Importance of self-attention for sentiment analysis. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 267–275.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2015. Visualizing and understanding neural models in nlp. *arXiv preprint arXiv:1506.01066*.

Edward Loper and Steven Bird. 2002. Nltk: the natural language toolkit. *arXiv preprint cs/0205028*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Chao Qiao, Bo Huang, Guocheng Niu, Daren Li, Daxiang Dong, Wei He, Dianhai Yu, and Hua Wu. 2018. Anew method of region embedding for text classification. In *International Conference on Learning Representations*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018. Disan: Directional self-attention network for rnn/cnn-free language understanding. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Yangyang Shi, Kaisheng Yao, Le Tian, and Daxin Jiang. 2016. Deep lstm based feature mapping for query classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1501–1511.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2019. Ernie 2.0: A continual pre-training framework for language understanding. *arXiv preprint arXiv:1907.12412*.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1422–1432.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Baoxin Wang. 2018. Disconnected recurrent neural networks for text categorization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2311–2320.

Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 90–94. Association for Computational Linguistics.

Yijun Xiao and Kyunghyun Cho. 2016. Efficient character-level document classification by combining convolution and recurrent layers. *arXiv preprint arXiv:1602.00367*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Dani Yogatama, Chris Dyer, Wang Ling, and Phil Blunsom. 2017. Generative and discriminative text classification with recurrent neural networks. *arXiv preprint arXiv:1703.01898*.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Jianyu Zhao, Zhiqiang Zhan, Qichuan Yang, Yang Zhang, Changjian Hu, Zhensheng Li, Liuxin Zhang, and Zhiqiang He. 2018. Adaptive learning of local semantic and global structure representations for text classification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2033–2043.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 207–212.

# Latent-Variable Generative Models for Data-Efficient Text Classification

**Xiaoan Ding**[1] **Kevin Gimpel**[2]

[1]University of Chicago, Chicago, IL, 60637, USA
[2]Toyota Technological Institute at Chicago, Chicago, IL, 60637, USA

xiaoanding@uchicago.edu, kgimpel@ttic.edu

## Abstract

Generative classifiers offer potential advantages over their discriminative counterparts, namely in the areas of data efficiency, robustness to data shift and adversarial examples, and zero-shot learning (Ng and Jordan, 2002; Yogatama et al., 2017; Lewis and Fan, 2019). In this paper, we improve generative text classifiers by introducing discrete latent variables into the generative story, and explore several graphical model configurations. We parameterize the distributions using standard neural architectures used in conditional language modeling and perform learning by directly maximizing the log marginal likelihood via gradient-based optimization, which avoids the need to do expectation-maximization. We empirically characterize the performance of our models on six text classification datasets. The choice of where to include the latent variable has a significant impact on performance, with the strongest results obtained when using the latent variable as an auxiliary conditioning variable in the generation of the textual input. This model consistently outperforms both the generative and discriminative classifiers in small-data settings. We analyze our model by using it for controlled generation, finding that the latent variable captures interpretable properties of the data, even with very small training sets.

## 1 Introduction

The most widely-used neural network classifiers are **discriminative**, that is, they are trained to explicitly favor the gold standard label over others. The alternative is to design classifiers that are **generative**; these follow a generative story that includes predicting the label and then the data conditioned on the label. Discriminative classifiers are preferred because they generally outperform their generative counterparts on standard benchmarks. These benchmarks typically assume large annotated training sets, little mismatch between training and test distributions, relatively clean data, and a lack of adversarial examples (Zue et al., 1990; Marcus et al., 1993; Deng et al., 2009; Lin et al., 2014).

However, when conditions are not ideal for discriminative classifiers, generative classifiers can actually perform better. Ng and Jordan (2002) showed theoretically that linear generative classifiers approach their asymptotic error rates more rapidly than discriminative ones. Based on this finding, Yogatama et al. (2017) empirically characterized the performance of RNN-based generative classifiers, showing advantages in sample complexity, zero-shot learning, and continual learning. Recent work in generative question answering models (Lewis and Fan, 2019) demonstrates better robustness to biased training data and adversarial testing data than state-of-the-art discriminative models.

In this paper, we focus on settings with small amounts of annotated data and improve generative text classifiers by introducing discrete latent variables into the generative story. Accordingly, the training objective is changed to log marginal likelihood of the data as we marginalize out the latent variables during learning. We parameterize the distributions with standard neural architectures used in conditional language models and include the latent variable by concatenating its embedding to the RNN hidden state before computing the softmax over words. While traditional latent variable learning in NLP uses the expectation-maximization (EM) algorithm (Dempster et al., 1977), we instead simply perform direct optimization of the log marginal likelihood using gradient-based methods. At inference time, we similarly marginalize out the latent variables while maximizing over the label.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 507–517, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

We characterize the performance of our latent-variable generative classifiers on six text classification datasets introduced by Zhang et al. (2015). We observe that introducing latent variables leads to large and consistent performance gains in the small-data regime, though the benefits of adding latent variables reduce as the training set becomes larger.

To better understand the modeling space of latent variable classifiers, we explore several graphical model configurations. Our experimental results demonstrate the importance of including a direct dependency between the label and the input in the model. We study the relationship between the label, latent, and input variables in our strongest latent generative classifier, finding that the label and latent capture complementary information about the input. Some information about the textual input is encoded in the latent variable to help with generation.

We analyze our latent generative model by generating samples when controlling the label and latent variables. Even with small training data, the samples capture the salient characteristics of the label space while also conforming to the values of the latent variable, some of which we find to be interpretable. While discriminative classifiers excel at separating examples according to labels, generative classifiers offer certain advantages in practical settings that benefit from a richer understanding of the data-generating process.

## 2 Discriminative and Generative Text Classifiers

We begin by defining our baseline generative and discriminative text classifiers for document classification. Our models are essentially the same as those from Yogatama et al. (2017); we describe them in detail here because our latent-variable models will extend them.[1] Our classifiers are trained on datasets $D$ of annotated documents. Each instance $\langle x, y \rangle \in D$ consists of a textual input $x = \{x_1, x_2, ..., x_T\}$, where $T$ is the length of the document, and a label $y \in \mathcal{Y}$.

[1]The main difference between our baselines and the models in Yogatama et al. (2017) are: (1) their discriminative classifier uses an LSTM with "peephole connections"; (2) they evaluate a label-based generative classifier ("Independent LSTMs") that uses a separate LSTM for each label. They also evaluate the model we described here, which they call "Shared LSTMs". Their Independent and Shared LSTMs perform similarly across training set sizes.

The discriminative classifier is trained to maximize the conditional probability of labels given documents: $\sum_{\langle x,y \rangle \in D} \log p(y \mid x)$. For our discriminative model, we encode a document $x$ using an LSTM (Hochreiter and Schmidhuber, 1997), and use the average of the LSTM hidden states as the document representation. The classifier is built by adding a softmax layer on top of the LSTM state average to get a probability distribution over labels.

The generative classifier is trained to maximize the joint probability of documents and labels: $\sum_{\langle x,y \rangle \in D} \log p(x, y)$. The generative classifier uses the following factorization:

$$p(x, y) = p(x \mid y)p(y) \tag{1}$$

We parameterize $\log p(x \mid y)$ as a conditional LSTM language model using the standard sequential factorization:

$$\log p(x \mid y) = \sum_{t=1}^{T} \log p(x_t \mid x_{<t}, y) \tag{2}$$

We define a label embedding matrix $V_{\mathcal{Y}} \in \mathbb{R}^{d_1 \times |\mathcal{Y}|}$. To predict the next word $x_{t+1}$, we concatenate the LSTM hidden state $h_t$ with the label embedding $v_y$ (a column of $V_{\mathcal{Y}}$), and feed it to a softmax layer to get the probability distribution over the vocabulary. More details about the factorization and parameterization are discussed in Section 3. The label prior $p(y)$ is acquired via maximum likelihood estimation and fixed during training of the remaining parameters.

At inference time, the prediction is made by maximizing $p(y \mid x)$ with respect to $y$ for the discriminative classifier and maximizing $p(x \mid y)p(y)$ for the generative classifier.

## 3 Latent-Variable Generative Classifiers

We now introduce discrete latent variables into the standard generative classifier as shown in Figure 1. We refer to the latent-variable model as an auxiliary latent generative model, as we expect the latent variable to contain auxiliary information that can help with the generation of the input.

Following the graphical model structure in Figure 1(b), we factorize the joint probability $p(x, y, c)$ as follows:

$$p(x, y, c) = p_\Theta(x \mid c, y)p_\Phi(c)p_\Psi(y) \tag{3}$$

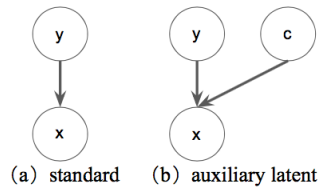


Figure 1: Graphical models of (a) standard generative classifier and (b) auxiliary latent generative classifier.

We parameterize $p_\Theta(x \mid c, y)$ as a conditional LSTM language model using the same factorization as above:

$$\log p_\Theta(x \mid c, y) = \sum_{t=1}^{T} \log p_\Theta(x_t \mid x_{<t}, c, y) \quad (4)$$

where $\Theta$ is the set of parameters of the language model. As in the generative classifier, we use a label embedding matrix $V_\mathcal{Y}$. In addition, we define a latent variable embedding matrix $V_\mathcal{C} \in \mathbb{R}^{d_2 \times |\mathcal{C}|}$ where $\mathcal{C}$ is the set of values for the discrete latent variable. Also like the generative classifier, we use an LSTM to predict each word with a softmax layer:

$$p_\Theta(x_t \mid x_{<t}, c, y) \propto \exp\{u_{x_t}^\top([h_t; v_y; v_c]) + b_{x_t}\} \quad (5)$$

where $h_t$ is the hidden representation of $x_{<t}$ from the LSTM, $v_y$ and $v_c$ are the embeddings for the label and the latent variable, respectively, $[u; v]$ denotes vertical concatenation, $u_{x_t}$ is the output word embedding, and $b_{x_t}$ is a bias parameter.

The prior distribution of the latent variable is parameterized as follows:

$$p_\Phi(c) \propto \exp\{w_c^\top v_c + b_c\} \quad (6)$$

where $\Phi$ is the set of parameters for this distribution which includes the vector $w_c$ and scalar $b_c$ for each $c$.

As in the standard generative model, the label prior $p_\Psi(y)$ is acquired from the empirical label distribution in the training data and remains fixed during training.

**Training.** As is standard in latent-variable modeling, we train our models by maximizing the log marginal likelihood:

$$\max_{\Theta,\Phi,V_\mathcal{C},V_\mathcal{Y}} \sum_{\langle x,y\rangle \in \mathcal{D}} \log \sum_{c \in \mathcal{C}} p_\Theta(x \mid c, y) p_\Phi(c) p_\Psi(y) \quad (7)$$

In NLP, these sorts of optimization problems are traditionally solved with the EM algorithm. However, we instead directly optimize the above quantity using automatic differentiation. This is natural because we use softmax-transformed parameterizations; a more traditional parameterization would assign parameters directly to individual probabilities, which then requires constrained optimization.

**Inference.** The prediction is made by marginalizing out the latent variables as follows:

$$\hat{y} = \operatorname*{argmax}_{y \in \mathcal{Y}} \sum_{c \in \mathcal{C}} p_\Theta(x \mid c, y) p_\Phi(c) p_\Psi(y) \quad (8)$$

We experimented with other inference objectives and found similar results. More details can be found in Appendix C.

**Differences with ensembles.** Our latent-variable model resembles an ensemble of multiple generative classifiers, but there are two main differences. First, all parameters in the latent generative classifier are trained jointly, while a standard ensemble combines predictions from multiple, independently-trained models. Joint training leads to complementary information being captured by latent variable values, as shown in our analysis. Moreover, a standard ensemble will lead to far more parameters (10, 30, or 50 times as many in our experimental setup) since each generative classifier is a completely separate model. Our approach simply conditions on the embedding of the latent variable value and therefore does not add many parameters.

## 4 Experiments

### 4.1 Datasets

We present our results on six publicly available text classification datasets introduced by Zhang et al. (2015), which include news categorization, sentiment analysis, question/answer topic classification, and article ontology classification.[2]

To compare classifiers across training set sizes, we follow the setup of Yogatama et al. (2017) and construct multiple training sets by randomly sampling 5, 20, 100, 1k, 2k, 5k, and 10k instances per label from each dataset.

### 4.2 Training Details

In all experiments, the word embedding dimension and the LSTM hidden state dimension are set to

[2] A more detailed dataset description is in Appendix E.

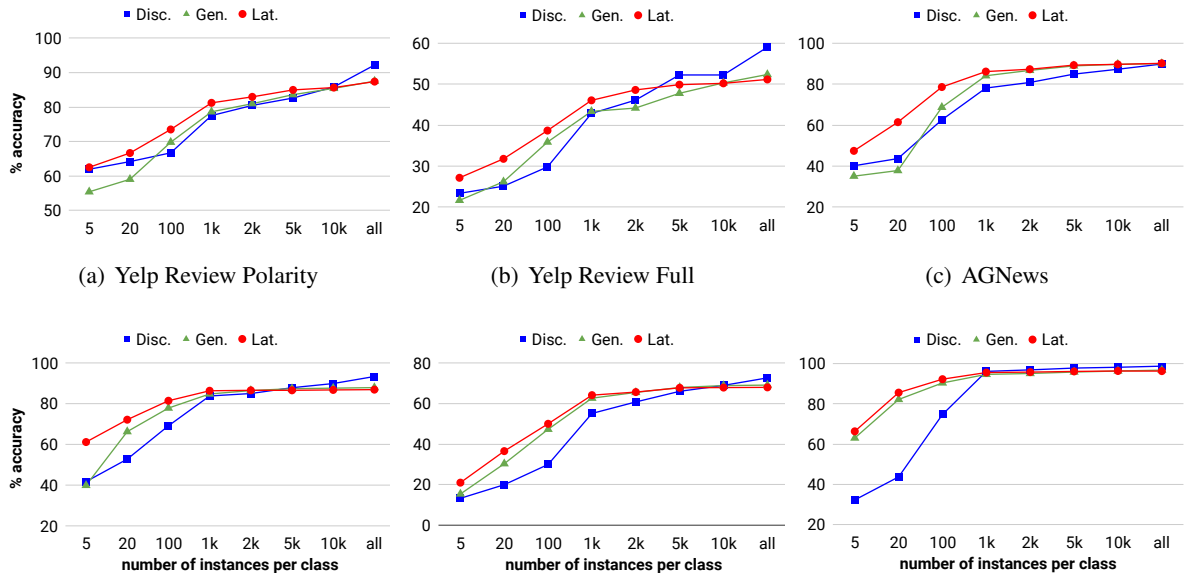


(a) Yelp Review Polarity (b) Yelp Review Full (c) AGNews

(d) Sogou (e) Yahoo (f) DBpedia

Figure 2: Comparison of classification accuracy of the discriminative (**Disc.**), standard generative (**Gen.**), and latent generative (**Lat.**) classifiers training across training set sizes.

100. All LSTMs use one layer and are unidirectional. The label dimensionality of all generative classifiers is set to 100. We adopt the same parameter settings as Yogatama et al. (2017) to ensure the results are comparable. For the latent-variable generative classifiers, we choose 10 or 30 latent variable values with embeddings of dimensionality 10, 50, or 100.

For optimization, we use Adam (Kingma and Ba, 2015) with learning rate 0.001. We do early stopping by evaluating the classification accuracy on the development set.

Due to memory limitations and computational costs, we truncate the length of the input sequences to 80 tokens before adding <s> and </s> to indicate the start and end of the document. Though truncation decreases the performance of the models, all models use the same truncated inputs, so the comparison is still fair.[3]

### 4.3 Baselines

To confirm we have built strong baselines, we first compare our implementation of the generative and discriminative classifiers to prior work. Our results in Appendix A show that our baselines are comparable to those of Yogatama et al. (2017).

[3]In other experiments, we compared performance with different truncation limits across training set sizes, finding the trends to be consistent with those presented here.

## 5 Results

### 5.1 Data Efficiency

Figure 2 shows results for the discriminative, generative, and latent generative classifiers in terms of data efficiency. Data efficiency is measured by comparing the accuracies of the classifiers when trained across varying sizes of training sets. Numerical comparisons on two datasets are shown in Table 1.

With small training sets, the latent generative classifier consistently outperforms both the generative and discriminative classifiers. When the generative classifier is better than the discriminative one, as in DBpedia, the latent classifier resembles the generative classifier. When the discriminative classifier is better, as in Yelp Polarity, the latent classifier patterns after the discriminative classifier. However, when the number of training examples is in the range of approximately 5,000 to 10,000 per class, the discriminative classifier tends to perform best.

With small training sets, the generative classifier outperforms the discriminative one in most cases except the very smallest training sets. For example, in the Yelp Review Polarity dataset, the first two points are from classifiers trained with only 10 and 40 instances in total. The other case in which generative classifiers underperform is when training over large training sets, which

| | Δ(Lat., Gen.) | | Δ(Lat., Disc.) | |
|---|---|---|---|---|
| | **AGNews** | **DBpedia** | **AGNews** | **DBpedia** |
| **5** | +12.3 | +3.3 | +7.2 | +34.0 |
| **20** | +23.5 | +3.3 | +17.7 | +41.8 |
| **100** | +9.8 | +1.8 | +16.0 | +17.5 |
| **1k** | +2.0 | +0.9 | +8.0 | +0.0 |
| **all** | +0.1 | -0.4 | +0.3 | -2.4 |

Table 1: **Δ(Lat., Gen.)**: change in accuracy when moving from generative to latent generative classifier; **Δ(Lat., Disc.)**: change in accuracy when moving from discriminative to latent generative classifier. The first column shows the number of training instances per class.

| | **5** | **20** | **100** | **1k** | **all** |
|---|---|---|---|---|---|
| **Yelp P** | +6.4 | +8.2 | +6.6 | +8.6 | +3.4 |
| **Yelp F** | +4.7 | +7.7 | +11.3 | +9.0 | +11.8 |
| **AGNews** | +13.4 | +19.9 | +27.9 | +1.8 | +0.4 |
| **Sogou** | +22.3 | +24.0 | +13.9 | +3.3 | +2.8 |
| **Yahoo** | +8.4 | +17.7 | +22.5 | +11.1 | +3.6 |
| **DBpedia** | +44.6 | +44.6 | +28.4 | +8.8 | +2.3 |

Table 2: Changes in accuracy when adding a directed edge from the label to the input, i.e., the improvement in accuracy when moving from the middle to the hierarchical latent generative classifier. Each column shows a different number of training instances per class.

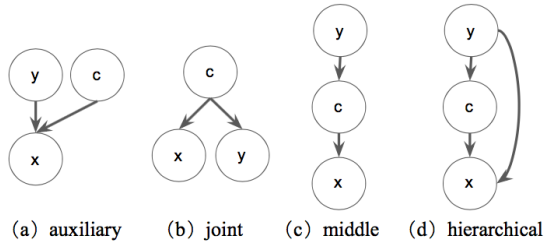


Figure 3: Graphical models of (a) auxiliary, (b) joint, (c) middle, and (d) hierarchical latent generative classifiers.

agrees with the theoretical and empirical findings in prior work (Yogatama et al., 2017; Ng and Jordan, 2002).

### 5.2 Effect of Graphical Model Structure

There are multiple choices to factorize the joint probability of the variables $x$, $y$, and $c$, which correspond to different graphical models. Here we consider other graphical model structures, namely those shown in Figure 3. We refer to the model in Figure 3(b) as the "joint" latent generative classifier since it uses the latent variable to jointly generate $x$ and $y$. We refer to the model in Figure 3(c) as the "middle" latent generative classifier as the latent variable separates the textual input from the label. We use similar parameterizations for these models as for the auxiliary latent classifier, with conditional language models to generate $x$ where the embedding of the latent variable is concatenated to the hidden state as in Section 3.

Figure 4 shows the comparison of the standard and the three latent generative classifiers on Yelp Review Polarity, AGNews, and DBpedia.[4] We observe that the auxiliary model consistently performs best, while the other two latent generative classifiers do not consistently improve over the standard generative classifier. On DBpedia, we see surprisingly poor performance when adding latent variables suboptimally. This suggests that the choice of where to include latent variables has a significant impact on performance.

**Dependency between label and input variable.** We observe that the most prominent difference between the auxiliary and the other two latent-variable models is that the label variable $y$ is directly linked to the input variable $x$ in the auxiliary model, which is also the case in the standard generative model.

In order to verify the importance of this direct dependency between the label and input, we create a new latent-variable model by adding a directed edge between $y$ and $x$ to the middle latent generative model. We refer to this model as the "hierarchical" latent generative classifier, which is shown in Figure 3(d). The results in Table 2 show the performance gains after adding this edge, which are all positive and sometimes very large. The resulting hierarchical model is very close in performance to the auxiliary model, which is unsurprising because these two models differ only in the presence of the edge from $y$ to $c$.

### 5.3 Effect of Latent Variables

We conduct a comparison to demonstrate that the performance gains are due to the latent-variable structure instead of an increased number of parameters when adding the latent variables.[5]

For the latent generative classifier, we choose 10 latent variable values with embeddings of di-

[4] Similar trends are observed for all datasets, so we only show three for brevity.

[5] The results in the preceding sections use the models with configurations tuned on the development sets. We follow the practice of Yogatama et al. (2017) and fix label dimensionality to 100, as described in Section 4.2. The only tuned hyperparameters are the number of latent variable values and the dimensions of their embeddings.

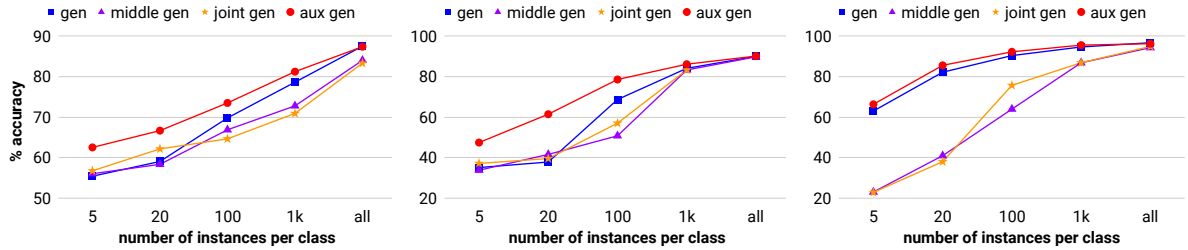


(a) Yelp Review Polarity (b) AGNews (c) DBpedia

Figure 4: Comparison of generative classifier (**gen**) and latent generative classifiers (**middle gen, joint gen, aux gen**).

| | | 5 | 20 | 100 | 1k | all |
|---|---|---|---|---|---|---|
| **Yelp P** | **Gen.** | 55.4 | 59.0 | 69.8 | 78.6 | 87.5 |
| | **Gen. PC** | 55.4 | 58.4 | 69.5 | 78.5 | 87.5 |
| | **Lat.** | 62.5 | 66.6 | 73.5 | 81.2 | 87.3 |
| | **Lat. PC** | 62.2 | 66.2 | 73.0 | 80.8 | 87.1 |
| **AGNews** | **Gen.** | 35.1 | 37.8 | 68.7 | 84.0 | 90.0 |
| | **Gen. PC** | 33.7 | 37.8 | 68.4 | 83.2 | 89.8 |
| | **Lat.** | 47.4 | 61.4 | 78.5 | 86.1 | 90.1 |
| | **Lat. PC** | 47.2 | 61.4 | 78.3 | 84.5 | 90.1 |

Table 3: Accuracy comparison of standard generative (**Gen.**) and latent (**Lat.**) classifiers under earlier experimental configurations and parameter-comparison configurations (**PC**). When controlling for the number of parameters, the latent classifier still outperforms the standard generative classifier, which indicates the performance gains are due to the latent variables instead of an increased number of parameters.

| | AGNews | | Sogou | |
|---|---|---|---|---|
| | **Direct** | **EM** | **Direct** | **EM** |
| **5** | 47.4 (62) | 47.4 (62) | 61.1 (144) | 61.2 (147) |
| **20** | 61.4 (72) | 61.1 (69) | 72.1 (30) | 72.2 (29) |
| **100** | 78.5 (10) | 78.5 (10) | 81.4 (11) | 81.4 (11) |
| **1k** | 86.1 (18) | 86.1 (18) | 86.4 (9) | 85.9 (9) |
| **all** | 90.1 (7) | 90.0 (5) | 86.9 (0) | 86.9 (0) |

Table 4: Comparison of the classification accuracy and convergence speed of the classifiers trained with direct optimization (**Direct**) of the log marginal likelihood and the EM algorithm (**EM**). The numbers inside the parentheses are the numbers of epochs required to reach the classification accuracies listed outside the parentheses.

mensionality 10, and a label dimensionality of 100 (**Lat. PC** in Table 3). For the standard generative classifier, we choose a label dimensionality of 110 (**Gen. PC** in Table 3). So, the numbers of parameters are comparable, since we ensure the same number of parameters in the "output" word embeddings in the softmax layer of the language model, which is the decision that most strongly affects the number of parameters.

Table 3 shows the results with different configurations, including the choices mentioned above as well as the results from earlier configurations mentioned in the paper. We observe that the latent generative classifiers still perform better in terms of data efficiency, which shows that the latent-variable structure accounts for the performance gains.

### 5.4 Learning via Expectation-Maximization

The results reported before are evaluated on the classifiers trained by directly maximizing the log marginal likelihood via gradient-based optimization. In addition, we train our latent generative classifiers with the EM algorithm (Salakhutdinov et al., 2003). More training details can be found in Appendix B.

To speed convergence, we use a mini-batch version of EM, updating the parameters after each mini-batch. Our results in Table 4 show that the direct approach and the EM algorithm have similar performance in terms of classification accuracy and convergence speed in optimizing the parameters of our latent models. Similar trends appear for the other datasets.

## 6 Analysis

### 6.1 Interpretation of Latent Variables

We take the strongest latent-variable model, the auxiliary latent generative classifier, and analyze the relationship among the latent, input, and label variables. We use the AGNews dataset, which contains 4 categories: world, sports, business, and sci/tech. The classifier we analyze has 10 values for the latent variable and is trained on a training set containing 1k instances per class.

We first investigate the relationship between

the latent variable and the label by counting co-occurrences. For each instance in the development set, we calculate the posterior probability distribution over the latent variable, and pick the value with the highest probability as the preferred latent variable value for that instance. This is reasonable since in our trained model, the posterior distribution over latent variable values is peaked. Then we categorize the data by their preferred latent variable values and count the gold standard labels in each group. We observe that the labels are nearly uniformly distributed in each latent variable value, suggesting that the latent variables are not obviously being used to encode information about the label.

Thus, we hypothesize there should be information other than that pertaining to the label that causes the data to cluster into different latent variable values. We study the differences of the input texts among the 10 clusters by counting frequent words, manually scanning through instances, and looking for high-level similarities and differences. We report our manual labeling for the latent variable values in Table 5.

For example, value 1 is mostly associated with future and progressive tenses; the words "will", "next", and "new" appear frequently. Value 2 tends to contain past and perfect verb tenses (the phrases "has been" and "have been" appear frequently). Value 3 contains region names like "VANCOUVER", "LONDON", and "New Brunswick", while value 7 contains country-oriented terms like "Indian", "Russian", "North Korea", and "Ireland". Our choice of only 10 latent variable values causes them to capture the coarse-grained patterns we observe here. It is possible that more fine-grained differences would appear with a larger number of values.

### 6.2 Generation with Latent Variables

Another advantage of generative models is that they can be used to generate data in order to better understand what they have learned, especially in seeking to understand latent variables. We use our auxiliary latent generative classifier to generate multiple samples by setting the latent variable and the label. Instead of the soft mixture of discrete latent variable values that is used in classification (since we marginalize over the latent variable at test time), here we choose a single latent variable value when generating a textual sample.

To increase generation diversity, we use temperature-based sampling when choosing the next word, where higher temperature leads to higher variety and more noise. We set the temperature to 0.6. Note that the latent-variable model here is trained on only 4000 instances (1k for each label) from AGNews, so the generated samples do suffer from the small size of data used in training the language model. Table 6 shows some generated examples. We observe that different combinations of the latent variable and label lead to generations that comport with both the labels and our interpretations of the latent variable values.

We speculate that the reason our generative classifiers perform well in the data-efficient setting is that they are better able to understand the data via language modeling rather than directly optimizing the classification objective. Our generated samples testify to the ability of generative classifiers to model the underlying data distribution even with only 4000 instances.

## 7 Related Work

**Supervised Generative Models.** Generative models have traditionally been used in supervised settings for many NLP tasks, including naive Bayes and other models for text classification (Maron, 1961; Yogatama et al., 2017), Markov models for sequence labeling (Church, 1988; Bikel et al., 1999; Brants, 2000; Zhou and Su, 2002), and probabilistic models for parsing (Magerrnan and Marcus, 1991; Black et al., 1993; Eisner, 1996; Collins, 1997; Dyer et al., 2016). Recent work in generative models for question answering (Lewis and Fan, 2019) learns to generate questions instead of directly penalizing prediction errors, which encourages the model to better understand the input data. Our work is directly inspired by that of Yogatama et al. (2017), who build RNN-based generative text classifiers and show scenarios where they can be empirically useful.

**Text Classification.** Traditionally, linear classifiers (McCallum and Nigam, 1998; Joachims, 1998; Fan et al., 2008) have been used for text classification. Recent work has scaled up text classification to larger datasets with models based on logistic regression (Joulin et al., 2017), convolutional neural networks (Kim, 2014; Zhang et al., 2015; Conneau et al., 2017), and recurrent neural networks (Xiao and Cho, 2016; Yogatama et al.,

| id | description | examples |
|---|---|---|
| 1 | future/progressive tenses | ... Commission **is likely to** follow opinion in the U.S. on the merger suit ...<br>... to increase computer software exports **is beginning to** show results ... |
| 2 | past/perfect tense | A screensaver targeting spam-related websites appears to **have been** too successful .<br>Universal **has signed** a handful of artists to a digital-only record label . .. |
| 3 | region names, locations | **Newcastle** manager Bobby Robson ... relieved of his duties ... **Newcastle** announced ...<br>**ABUJA** ... its militias **in Darfur** before they would sign ... |
| 4 | mixture | |
| 5 | abbreviations | **St.** Louis advanced to the **N.L.** championship series for the third time in five years ...<br>**UAL ( UALAQ.OB : OTC BB** - news - research ) ... ( **UAIRQ.OB : OTC BB** ... |
| 6 | numbers, money-related | ...challenge larger rivals in the fast-growing **2.1 billion-a-year** sleep aid market ....<br>... to a **$ 25,000 prize** , and more importantly , into the history books ... |
| 7 | dates | ... an Egyptian diplomat said **on Friday**, and the abduction of ... **earlier this month** .<br>... expected **Monday or Tuesday** , ... doctors and nurses off for the **holiday weekend** ... |
| 8 | country-oriented terms | **Rwandan** President ... in the **Democratic Republic of the Congo** after ...<br>Pope John Paul II issued a new appeal for peace **in Iraq and the Middle East** ... |
| 9 | mixure | |
| 10 | symbols, links | ... **A HREF = " http : / / www.reuters.co.uk / financeQuoteLookup.jhtml** ...<br>**& lt ;** strong **& gt ;** Analysis **& lt ; /** strong **& gt ;** Contracting out the blame ... |

Table 5: Latent variable values ("id"), our manually-defined descriptions, and examples of instances associated to them. Boldface is used to highlight cues to our labeling. We use the term "mixture" when we did not find clear signals to interpret the latent variable value.

| | |
|---|---|
| **Latent variable id = 3: region names, locations** | |
| world | **BEIJING** ( Reuters ) - **Oklahoma** supporters unemployment claims that he plans to trying to restore access next season 's truce by ruling , saying a major parliament . |
| sport | The **Dallas** Cowboys today continued advantage today with **Miami** and the Hurricanes had to get the big rotation for the first time this year . |
| business | **Las Vegas** took one more high-stepping kick across the pond as casino operator Caesars Entertainment Inc . |
| sci/tech | **SAN FRANCISCO** - Sun Microsystems on Monday will surely offer the deal to sell up pioneer members into two years and archiving . |
| **Latent variable id = 6: numbers, money-related** | |
| world | An Israeli helicopter gunship fired a missile among **$ 5** million in to Prime Minister Ariel Sharon on the streets of U.S. warming may not be short-lived . |
| sport | On Wednesday it would win the disgruntled one of the season opener in a # 36 ; **8.75 billion** of World Cup final day for second . |
| business | Reuters - U.S. drug company Biogen Idec is considering an all-share bid of more than **8.5 billion euros** ( # 36 ; **10.6 billion** ) for Irish peer Elan , a newspaper reported on Sunday . |
| sci/tech | The JVC Everio GZ-MC100 ( **$ 1199.95** ) and GZ-MC200 ( **$ 1299.95** ) will use **4GB** Microdrive cards , which are removable hard drives measuring **1.5** inches square , but will also lost vital " fans to recently over . |
| **Latent variable id = 10: symbols, links** | |
| world | A German court is set to hear all its secular oil , and western Kerik in Fallujah . **& lt ; A HREF = " http : / / www.investor.reuters.com / FullQuote.aspx ? ticker = Agency target = Army** ... |
| sport | White Sox to an overpowering 49-0 victory over The world championship game . **& lt ; br & gt ; & lt ; br & gt ;** Comcast SportsNet |
| business | NEW YORK ( Reuters ) - U.S. stocks climbed on Monday , with a steep decline in commodity prices and lower crude oil dented shares of Alcoa Inc . **& lt ; A HREF = " http : / / www.investor.reuters.com / FullQuote.aspx ? ticker = GDT.N target = / stocks / quickinfo / fullquote " & gt ; & lt ; / A & gt ;**. |
| sci/tech | Spyware problems introduced a radio frequency code Thursday . **& lt ; FONT face = " verdana , MS Sans Serif , arial , helvetica " size = " -2 " color = " # 666666 " & gt ; & lt ; B & gt ; -washingtonpost.com & lt ; / B & gt ; & lt** |

Table 6: Generated examples by controlling the latent variables and labels (world, sport, business, sci/tech) with our latent classifier trained on a small subset of the AGNews dataset.

2017), the latter of which is most closely-related to our models.

**Latent-variable Models.** Latent variables have been widely used in both generative and discriminative models to learn rich structure from data (Petrov and Klein, 2007, 2008; Blunsom et al., 2008; Yu and Joachims, 2009; Morency et al., 2008). Recent work in neural networks has shown that introducing latent variables leads to higher representational capacity (Kingma and Welling, 2014; Chung et al., 2015; Burda et al., 2016; Ji et al., 2016). However, unlike variational autoencoders (Kingma and Ba, 2015) and related work that use continuous latent variables,

our model is more similar to recent efforts that combine neural architectures with discrete latent variables and end-to-end training (Ji et al., 2016; Kim et al., 2017; Kong et al., 2017; Chen and Gimpel, 2018; Wiseman et al., 2018, *inter alia*).

## 8 Discussion and Future Work

An alternative solution to the small-data setting is to use language representations pretrained on large, unannotated datasets (Mikolov et al., 2013; Pennington et al., 2014; Devlin et al., 2019). In other experiments not reported here, we found that using pretrained word embeddings leads to larger performance improvements for the discriminative classifiers than the generative ones.

Future work will explore the performance of latent generative classifiers in other challenging experimental conditions, including testing robustness to data shift and adversarial examples as well as zero-shot learning. Another thread of future work is to explore the performance of discriminative models with latent variables, and investigate combining pretrained representations with both generative and discriminative classifiers.

## 9 Conclusion

We focused in this paper on improving the data efficiency of generative text classifiers by introducing discrete latent variables into the generative story. Our experimental results demonstrate that, with small annotated training data, latent generative classifiers have larger and more stable performance gains over discriminative classifiers than their standard generative counterparts. Analysis reveals interpretable latent variable values and generated samples, even with very small training sets.

## Acknowledgments

We would like to thank Lingyu Gao, Qingming Tang, and Lifu Tu for helpful discussions, Michael Maire and Janos Simon for their useful feedback, the anonymous reviewers for their comments that improved this paper, and Google for a faculty research award to K. Gimpel that partially supported this research.

## References

Daniel M. Bikel, Richard Schwartz, and Ralph M. Weischedel. 1999. An algorithm that learns what's in a name. *Machine learning*, 34(1-3):211–231.

Ezra Black, Fred Jelinek, John Lafrerty, David M. Magerman, Robert Mercer, and Salim Roukos. 1993. Towards history-based grammars: Using richer models for probabilistic parsing. In *31st Annual Meeting of the Association for Computational Linguistics*, pages 31–37, Columbus, Ohio, USA. Association for Computational Linguistics.

Phil Blunsom, Trevor Cohn, and Miles Osborne. 2008. A discriminative latent variable model for statistical machine translation. In *Proceedings of ACL-08: HLT*, pages 200–208, Columbus, Ohio. Association for Computational Linguistics.

Thorsten Brants. 2000. TnT – a statistical part-of-speech tagger. In *Sixth Applied Natural Language Processing Conference*, pages 224–231, Seattle, Washington, USA. Association for Computational Linguistics.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. 2016. Importance weighted autoencoders. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Mingda Chen and Kevin Gimpel. 2018. Smaller text classifiers with discriminative cluster embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 739–745, New Orleans, Louisiana. Association for Computational Linguistics.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. 2015. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, pages 2980–2988.

Kenneth Ward Church. 1988. A stochastic parts program and noun phrase parser for unrestricted text. In *Second Conference on Applied Natural Language Processing*, pages 136–143, Austin, Texas, USA. Association for Computational Linguistics.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics*, pages 16–23, Madrid, Spain. Association for Computational Linguistics.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2017. Very deep convolutional networks for text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1107–1116, Valencia, Spain. Association for Computational Linguistics.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209, San Diego, California. Association for Computational Linguistics.

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Yangfeng Ji, Gholamreza Haffari, and Jacob Eisenstein. 2016. A latent variable recurrent neural network for discourse-driven language models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 332–342, San Diego, California. Association for Computational Linguistics.

Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, Valencia, Spain. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. 2017. Structured attention networks. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational Bayes. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Lingpeng Kong, Chris Dyer, and Noah A. Smith. 2017. Segmental recurrent neural networks. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Mike Lewis and Angela Fan. 2019. Generative question answering: Learning to answer the whole question. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft COCO: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

David M. Magerrnan and Mitchell P. Marcus. 1991. Pearl: A probabilistic chart parser. In *Fifth Conference of the European Chapter of the Association for Computational Linguistics*, Berlin, Germany. Association for Computational Linguistics.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

M. E. Maron. 1961. Automatic indexing: An experimental inquiry. *J. ACM*, 8(3):404–417.

Andrew McCallum and Kamal Nigam. 1998. A comparison of event models for naive Bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*, volume 752, pages 41–48.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Louis-Philippe Morency, Xu Sun, Daisuke Okanoharay, and Jun'ichi Tsujii. 2008. Modeling Latent-Dynamic in Shallow Parsing: A Latent

Conditional Model with Improved Inference. In *The 22nd International Conference on Computational Linguistics (COLING 2008)*, Manchester, UK.

Andrew Y. Ng and Michael I. Jordan. 2002. On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. In *Advances in neural information processing systems*, pages 841–848.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 404–411, Rochester, New York. Association for Computational Linguistics.

Slav Petrov and Dan Klein. 2008. Discriminative log-linear grammars with latent variables. In *Advances in neural information processing systems*, pages 1153–1160.

Ruslan Salakhutdinov, Sam Roweis, and Zoubin Ghahramani. 2003. Relationship between gradient and EM steps in latent variable models.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2018. Learning neural templates for text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3174–3187, Brussels, Belgium. Association for Computational Linguistics.

Yijun Xiao and Kyunghyun Cho. 2016. Efficient character-level document classification by combining convolution and recurrent layers. *arXiv preprint arXiv:1602.00367*.

Dani Yogatama, Chris Dyer, Wang Ling, and Phil Blunsom. 2017. Generative and discriminative text classification with recurrent neural networks. *arXiv preprint arXiv:1703.01898*.

Chun-Nam John Yu and Thorsten Joachims. 2009. Learning structural SVMs with latent variables. In *ICML*, volume 2, page 5.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 649–657, Cambridge, MA, USA. MIT Press.

GuoDong Zhou and Jian Su. 2002. Named entity recognition using an HMM-based chunk tagger. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 473–480, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Victor Zue, Stephanie Seneff, and James Glass. 1990. Speech database development at MIT: TIMIT and beyond. *Speech communication*, 9(4):351–356.

# PaRe: A Paper-Reviewer Matching Approach Using a Common Topic Space

**Omer Anjum**[*,1], **Hongyu Gong**[*,1], **Suma Bhat**[1], **Jinjun Xiong**[2], **Wen-Mei Hwu**[1]
[1]University of Illinois at Urbana-Champaign, USA
[2] IBM Thomas J. Watson Research Center, USA
{oanjum, hgong6, spbhat2, w-hwu}@illinois.edu, jinjun@us.ibm.com

## Abstract

Finding the right reviewers to assess the quality of conference submissions is a time consuming process for conference organizers. Given the importance of this step, various automated reviewer-paper matching solutions have been proposed to alleviate the burden. Prior approaches, including bag-of-words models and probabilistic topic models have been inadequate to deal with the vocabulary mismatch and partial topic overlap between a paper submission and the reviewer's expertise. Our approach, the common topic model, jointly models the topics common to the submission and the reviewer's profile while relying on abstract topic vectors. Experiments and insightful evaluations on two datasets demonstrate that the proposed method achieves consistent improvements compared to available state-of-the-art implementations of paper-reviewer matching.

## 1 Introduction

The peer review mechanism constitutes the bedrock of today's academic research landscape spanning submissions to conferences, journals, and funding bodies across numerous disciplines. Matching a paper (or a proposal) to an expert in the topic presented in the paper requires the knowledge of diverse topics of both the submission as well as that of the reviewer's expertise in addition to knowing recent affiliations and co-authorship to resolve conflict of interest. Considering the scale of current conference submissions, performing the task of paper-reviewer matching manually incurs significant overheads to the program committee (PC) and calls for automating the process. Faced with record number of paper submissions essentially interdisciplinary in nature, the inadequacy of available reviewer matching systems to scale to the current needs is being expressed by many conference program committees. It is also notable that the approaches to address the challenges seem ad-hoc and non-scalable, as described in a few of the PC blogs; "Looking at the abstracts for many of the submissions it also quickly became clear that there was disparity in how authors chose topic keywords for submissions with many only using a single keyword and others using over half a dozen keywords. As such relying on the keywords for submissions became difficult. The combined effect of these problems made any automatic or semi-automatic assignment using HotCRP suboptimal...So, we chose to hand assign the papers." (Falsafi et al., 2018), and again in "Our plan was to rely on the Toronto Paper Matching System (TPMS) in allocating papers to reviewers. Unfortunately, this system didnt prove as useful as we had hoped for (it requires more extensive reviewer profiles for optimal performance than what we had available) and the work had to rely largely on the manual effort..." (ACL, 2019). Noting the urgent need to advance research to address this problem, we study this challenge of matching a paper with a reviewer from a list of potential reviewers for the purpose of assessing the quality of the submission.

Aside from the long precedence of research in the related area of expertise retrieval – that of expert finding and expert profiling (Balog et al., 2012), several recent attempts have been made to automate the process (Price and Flach, 2017). These include, the Toronto paper matching system (Laurent and Zemel, 2013), the IEEE INFOCOM review assignment system (Li and Hou, 2016), and the online reviewer recommendation system (Qian et al., 2018). Central to these systems is a module that performs the paper-reviewer assignment, which can be broken down into its matching and constraint satisfaction constituents. The constraint

*Omer Anjum and Hongyu Gong have equal contribution.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 518–528, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

satisfaction component typically handles the constraints that each paper be reviewed by at least a few reviewers, each reviewer be assigned no more than a few papers, and thaat reviewers not be assigned papers for which they have a conflict of interest. A second constituent is that of finding a reviewer from a list of reviewers based on the relevance of the person's expertise with the topic of the submission. This latter aspect will be the focus of our study.

Available approaches to solve this matching problem can be broadly classified into the following categories (Price and Flach, 2017): a) Feature-based matching, where a set of topic keywords are collected for each paper and each reviewer. The reviewers are then ranked in order of the number of keyword matches with the paper; b) Automatic feature construction with profile-based matching, where the relevance is decided by building automatic topic representations of both papers and reviewers; c) Bidding, a more recent method, involves giving the reviewers access to all the papers and asking them to bid on papers of their choice. The approaches used in this study are of the profile-based matching kind, where we rely on the use of abstract topic vectors and word embeddings (Mikolov et al., 2013b) to derive the semantic representations of the paper and the expertise area of the reviewer. This is a departure from the bag-of-words approach taken in related prior approaches, e.g. (Laurent and Zemel, 2013), relying on automatic topic extraction using keywords – a ranked list of terms taken from the paper and the reviewer's profile.

In general, we assume that a reviewer can be represented by a collection of the abstracts of her past publications (termed as the reviewer's profile) and a submission by its abstract. While attempting to match the paper with the reviewer via their profile representations, the obvious difference in the document lengths gives rise to a mismatch due to the small overlap in vocabulary and a consequent scarcity of shared contexts of these overlapping terms. This is because, while past publications of a reviewer may be sufficient to provide a reasonable context for a topical word, a submission abstract provides a very limited context for that topical word.

To alleviate this problem of mismatched 'profiles', we use the idea of a shared topic space between the submission and the reviewer's profile. In our experiments we compare our approach to match the profiles using abstract vectors with that using the hidden topic model (also a set of abstract topic vectors) (Gong et al., 2018). The two approaches primarily differ in the way the shared topic space is constructed, which we describe in Section 4. We also include other baseline comparisons where the matching is done on the basis of common topic words (keywords) and word- or document-embeddings.

This study makes the following contributions:
(1) Instead of relying on a collection of topic words (keywords chosen by the authors or experts), our approach relies on abstract topic vectors to represent the common topics shared by the submission and the reviewer.
(2) We propose a model that outperforms state-of-the-art approaches in the task of paper-reviewer matching on a benchmark dataset (Mimno and McCallum, 2007). Additionally, a field evaluation of our approach performed by the program committee of a tier-1 conference showed that it was highly useful.

## 2 Related Work

The paper-reviewer matching task lays the basis for the peer review process ubiquitous in academic conferences and journals. Existing automatic approaches can be broadly categorized into the following types according to the type of models for comparing documents: *feature-based models*, *probabilistic models*, *embedding-based models*, *graph models* and *neural network models*.

**Feature-based models**. A list of keywords which summarizes the topics of a submission is used as informative features in the matching process (Dumais and Nielsen, 1992; Basu et al., 1999). Automatic extraction of these features achieves higher efficiency and one commonly-used feature is a bag-of-words weighted by the words' TF-IDF scores (Jin et al., 2018; Tang et al., 2010; Li and Hou, 2016; Nguyen et al., 2018).

**Probabilistic models**. The Latent Dirichlet Allocation (LDA) model is the most commonly used probabilistic model in expertise matching, where each topic is represented as a distribution over a given vocabulary and each document is a mixture of hidden topics (Blei et al., 2003). The popular Toronto Paper Matching System (TPMS) (Laurent and Zemel, 2013) uses LDA to generate the similarity score between a reviewer and a submis-

sion (Li and Hou, 2016). One limitation of LDA is that it does not make use of the potential semantic relatedness between words in a topic because of its assumption that words are generated independently (Xie et al., 2015). Variants of LDA have been proposed to incorporate notions of semantic coherence for more effective topic modeling (Hu and Tsujii, 2016; Das et al., 2015; Xun et al., 2017). Beyond having probabilistic models for topics, Jin et al. sought to capture the temporal changes of reviewer interest as well as the stability of their interest trend with probabilistic modeling (Jin et al., 2017).

In addition to their inherent limiting assumptions such as independence of semantically related words, probabilistic models, including LDA, require a large corpus to accurately identify the topics and topic distribution in each document, which can be problematic when applied to short documents, such as abstracts.

**Embedding-based models**. Latent Semantic Indexing (LSI) proposes to represent a document as a single dense vector (Deerwester et al., 1990). The documents corresponding to reviewers and submissions can thus be transformed into their respective vector representations. The relevance of a reviewer to a given submission would then be measured using a distance metric in the vector space, such as the cosine similarity. Other approaches have used word or document embeddings as document representations in order to compare two documents.

Kou et al. derived topic vectors by treating each topic as a distribution of words (Kou et al., 2015). In comparison, the key improvement in our work is that the topics are derived based on word embeddings instead of word distributions. Moreover, we derive common topics for each submission-reviewer pair, and as a result, the topics can vary from pair to pair.

Another approach to capture similarity between documents is by the use of the Word Mover's Distance (WMD). It relies on the alignment of word pairs from two texts, and the textual dissimilarity is measured as the total distance between the vectors of the word pairs (Kusner et al., 2015). More recently, a hidden topic model has been used to compare two documents via extracted abstract topic vectors, which showed a strong performance in comparing document for semantic similarity (Gong et al., 2018).

Extending the models in this category, we propose the common topic model. Similar to the hidden topic model, we extract topic vectors using word embeddings and match documents at the topic level. The hidden topic model extracts topics purely relying on the reviewer profile, so the topic vectors can be regarded as a summary of the reviewers' research interest. In contrast, the common topic vectors are selected based on the knowledge of both the submission and the reviewer's profile, which are expected to capture the topical overlap between the two. As we will show in the qualitative evaluation, the hidden topic model is likely to miss some important topics when a reviewer has broad research interests, resulting in an underestimation of the paper-reviewer relevance. The common topic model is able to overcome this limitation by extracting topics with reference to submissions.

**Graph models**. All of the models mentioned above only assume access to the texts of submissions and reviewers' publications. Some works also make use of external information such as coauthorship to improve the matching performance. For instance, Liu et al. capture academic connections between reviewers using a graph model, and show that such information improves the matching quality. Each node in their graph model represents a reviewer (Liu et al., 2014). There is an edge between two nodes if the corresponding reviewers have co-authored papers, and the edge weight is the number of publications. This work also uses LDA to measure the similarity between the submission and the reviewer.

**Neural network models**. Dense vectors are learned by neural networks as the semantic representation of documents (Socher et al., 2011; Le and Mikolov, 2014; Lin et al., 2015; Lau and Baldwin, 2016). When it comes to the task of expertise matching, the reviewer-submission relevance can thus be measured by the similarity of the vector representations of their textual descriptions.

## 3 System Overview

The different stages of our system, together called PaRe, are briefly explained as below:

*Data collection*. At this stage, we collect previous publications from one or more tier-1 conferences in the same domain as the one to which reviewer-submission matching is applied. This data is used to create our pool of candidate reviewers and do-

main knowledge of the research area. The source of the data is Microsoft Academic Graph (MSG) (Sinha et al., 2015). All the abstracts of a reviewer are concatenated as one document, which is then used to profile the reviewer. Reviewers' profiles reflect their research topics, which are later used in the reviewer-submission matching process.

*Data processing*. Since our proposed model is based on word embeddings, we pre-train embeddings using CBOW model of word2vec on the collected publications (Mikolov et al., 2013a). The dense word representations are intended to capture domain-specific lexical semantics. The data collection and processing are detailed in Section 5.

*Reviewer-submission matching*. A common topic modeling approach is proposed in this work to match reviewers with submissions. The model compares the abstracts of submissions and reviewers' past abstracts during the matching process to decide the reviewer-submission relevance by finding their common research topics. The algorithm is described in Section 4.

## 4 Modeling

For the purpose of our study, we consider a reviewer's profile to be the concatenation of the abstracts from their previous publications. Let $m$ be the number of words in the reviewer's profile and let the *normalized* word embeddings of these words be stacked as a reviewer matrix $\mathbf{R} \in \mathbb{R}^{\mathbf{d}\times\mathbf{m}}$, where $d$ is the embedding dimension. Since the embedding is normalized, we have $\|\mathbf{R_i}\|_\mathbf{2} = \mathbf{1}$ for each column in $\mathbf{R}$. Next suppose that the submission is represented by an $n-$word sequence of its abstract. Similar to the case of the reviewer, we stack its normalized embeddings as a submission matrix $\mathbf{S} \in \mathbb{R}^{\mathbf{d}\times\mathbf{n}}$. Also we have for each column $\|\mathbf{S_j}\|_\mathbf{2} = \mathbf{1}, \forall \mathbf{1} \leq \mathbf{j} \leq \mathbf{n}$.

**Common topic selection**. Inspired by the compositionality of embeddings (Gong et al., 2017) and the hidden topic model in document matching (Gong et al., 2018), our intention is to extract topics from reviewer profiles and submissions to summarize their topical overlap. We would like to remind the reader that the topics extracted are neither words nor distributions, but only abstractions and constitute a set of numeric vectors that do not necessarily have a textual representation. To establish the connection between topics, reviewer profiles and submissions, we assume that the topic vectors can be written as a linear combination of the embeddings of component words in either the reviewer profiles or the submissions. This assumption is supported by the geometric property of word embeddings that the weighted sum of the component word embeddings have been shown to be a robust and efficient representation of sentences and documents (Mikolov et al., 2013b). Intuitively, the extracted common topics would be highly correlated with the subset of the words in the reviewer profile or that of the submission in terms of semantic similarity.

Let both the reviewer and the submission have $K$ research topics, with each topic represented by a $d-$dimensional vector. This vector is an abstract topic vector and does not necessarily correspond to a specific word or a word distribution as in LDA (Blei et al., 2003). Suppose that these topic vectors of the reviewer are stacked as a matrix $\mathbf{P} \in \mathbb{R}^{d\times K}$, and those of the submission as $\mathbf{Q} \in \mathbb{R}^{d\times K}$. Therefore, these matrices can be represented as a linear combination of the underlying word vectors.

$$\begin{aligned} \mathbf{P} &= \mathbf{Ra}, \\ \mathbf{Q} &= \mathbf{Sb}, \end{aligned} \tag{1}$$

where $\mathbf{a} \in \mathbb{R}^{m\times K}, \mathbf{b} \in \mathbb{R}^{n\times K}$ are the coefficients in the linear combinations.

Our goal is to find the common topics shared by a given reviewer and a given submission, to account for the overlap of their research areas. We consider a pair of topics from a reviewer and a submission respectively to constitute a pair of common topics if they are semantically similar. For example, if the reviewer's research areas are *machine learning* and *theory of computation*, and the submission is about *classification* in *natural language processing*, then (*machine learning, classification*) can be regarded as a pair of common topics, while the other pairs corresponding to the areas *theory of computation* and *natural language processing* are much less similar. We used cosine similarity to measure the semantic similarity of two topic vectors [*]. The similarity $\text{sim}(\mathbf{P}_k, \mathbf{Q}_k)$ between reviewer topic $\mathbf{P}_k$ and submission topic $\mathbf{Q}_k$ is shown below.

$$\text{sim}(\mathbf{P}_k, \mathbf{Q}_k) = \frac{\mathbf{P}_k^T \mathbf{Q}_k}{\|\mathbf{P}_k\| \cdot \|\mathbf{Q}_k\|}. \tag{2}$$

[*]We leave it to future work to experiment with other useful measures of semantic similarity.

For $K$ pairs of topic vectors $\{\mathbf{P_k}, \mathbf{Q_k}\}_{k=1}^{K}$, their similarity is the sum of the pairwise similarities:

$$\text{sim}(\mathbf{P}, \mathbf{Q}) = \sum_{k=1}^{K} \text{sim}(\mathbf{P}_k, \mathbf{Q}_k). \quad (3)$$

This in turn translates to identifying the common research topics between the reviewer and the submission, i.e., we need to find $K$ such pairs of topics that have the maximum similarity. Based on our discussions above, the approach of common topic extraction can be formulated as an optimization problem:

$$\begin{aligned} &\max_{\mathbf{a},\mathbf{b}} \text{sim}(\mathbf{P}, \mathbf{Q}) \\ \text{s.t.} \quad &\mathbf{P} = \mathbf{R}\mathbf{a}, \\ &\mathbf{Q} = \mathbf{S}\mathbf{b}, \\ &\mathbf{P}^T\mathbf{P} = \mathbf{Q}^T\mathbf{Q} = \mathbf{I} \end{aligned} \quad (4)$$

The first two constraints are based on the linear assumption shown in Eq. 1. Without loss of generality, we add the third constraint that the topic vectors are orthogonal as shown in Eq. 4 to avoid generating multiple similar topic vectors. The closed-form solution to this optimization problem can be derived via singular value decomposition on the correlation matrix of $\mathbf{R}$ and $\mathbf{S}$ (Wegelin, 2000).

Let topic vectors $\mathbf{P}^*$ and $\mathbf{Q}^*$ be the optimal solution, both describing the common topics shared by the reviewer and the submission. In the following discussions, we use $\mathbf{P}^*$ as the common topic vectors.

**Common topic scoring**. To further quantify the reviewer-submission relevance, we need to evaluate how significant these common topics are for the reviewer and the submission respectively. Reusing the example where a reviewer's area are *machine learning* and *theory of computation*, we know that *machine learning* is the common topic between the reviewer and the submission. If the topic of *machine learning* were only a small part of the reviewer's publications, the reviewer may not be a good match for the submission since reviewer is more of an expert in *theory of computation* than in *machine learning*.

To evaluate how well the topics reflect a reviewer's expertise, we define the importance of common topics $\mathbf{P}^*$ for both the reviewer and the submission. Consider the vector of the $i-$th word in the reviewer's profile, $\mathbf{R}_i$, and the $k$-th topic vector $\mathbf{P}_k^*$. The relevance between $\mathbf{R}_i$ and $\mathbf{P}_k^*$ is defined as the their squared cosine similarity.

$$\text{rel}(\mathbf{R}_i, \mathbf{P}_k^*) = \cos^2(\mathbf{R}_i, \mathbf{P}_k^*) = (\mathbf{R}_i^T\mathbf{P}_k^*)^2. \quad (5)$$

Note that we do not use cosine similarity as is, since $\mathbf{R}_i$ and $\mathbf{P}_k^*$ might be negatively correlated and the cosine similarity can be negative. Instead, we use the square of the cosine similarity to reflect the strength of their correlation.

The relevance between word $\mathbf{R}_i$ and a set of topic vectors $\mathbf{P}^*$ is defined as the sum of the relevance between the word and each topic vector.

$$\text{rel}(\mathbf{R}_i, \mathbf{P}^*) = \sum_{k=1}^{K} \text{rel}(\mathbf{R}_i, \mathbf{P}_k^*). \quad (6)$$

We can think of word vector $\mathbf{R}_i$ to be projected along the $K$ dimensions of a linear subspace spanned by topic vectors in $\mathbf{P}^*$. If $\mathbf{R}_i$ lies in this linear subspace, then it can be represented as a linear combination of the topic vectors. In this case, $\text{rel}(\mathbf{R}_i, \mathbf{P})$ achieves the maximum of 1. If the word vector is orthogonal to all topic vectors in $\mathbf{P}^*$, the relevance results in the minimum relevance of 0. Thus, the range of $\text{rel}(\mathbf{R}_i, \mathbf{P})$ is from 0 to 1.

Furthermore, we define the relevance between the reviewer and the topics as the average of the relevance between the words and the topics.

$$\text{rel}(\mathbf{R}, \mathbf{P}^*) = \frac{1}{m} \sum_{i=1}^{m} \text{rel}(\mathbf{R}_i, \mathbf{P}^*). \quad (7)$$

The reviewer-topic relevance $\text{rel}(\mathbf{R}, \mathbf{P})$ also ranges from 0 to 1.

Similarly, we measure the relevance between a submission and a set of common topics, $\text{rel}(\mathbf{S}, \mathbf{P}^*)$ by measuring the relevance between words in the submission and common topics. The submission-topic relevance reflects the importance of the common topics for a submission. We define the reviewer-submission matching score as a harmonic mean (f-measure) of the reviewer-topic and submission-topic relevance (Powers, 2015).

$$\text{rel}(\mathbf{R}, \mathbf{S}) = \frac{2 \cdot \text{rel}(\mathbf{R}, \mathbf{P}^*) \cdot \text{rel}(\mathbf{S}, \mathbf{P}^*)}{\text{rel}(\mathbf{R}, \mathbf{P}^*) + \text{rel}(\mathbf{S}, \mathbf{P}^*)}. \quad (8)$$

The reviewer-submission relevance is high when the common topic vectors $\mathbf{P}^*$ are highly relevant to both the reviewer and the submission.

It indicates that the submission has a substantial overlap with reviewer's research area, and that the reviewer is considered to be a good match for the submission.

## 5 Experiments and Results

In this section, we empirically compare our proposed common topic model approach against a variety of models in the task of expertise matching.

### 5.1 Dataset

For our experiments, we use the two datasets described below.

**NIPS dataset**. This is a benchmark dataset described in (Mimno and McCallum, 2007) and commonly used in the evaluation of expertise matching. It consists of 148 NIPS papers accepted in 2006 and abstracts from the publications of 364 reviewers. It includes annotations from 9 annotators on the relevance of 650 reviewer-paper pairs. Each pair is rated on a scale from 0 to 3. Here "0" means irrelevant, "1" means slightly relevant, "2" means relevant and "3" means very relevant.

**A new dataset.** Our proposed paper-reviewer matching system is applied to a tier-1 conference in the area of computer architecture. We created a new dataset for the evaluation of expertise matching from the submissions to this conference. We first collected a pool of 2284 candidate reviewers with publications in top conferences of computer architecture. A reviewer selection policy was adopted by the conference program committee to select reviewers still active in relevant areas. Reviewers were excluded if

1) they started publishing 40 years ago, but had no publications for the last ten years;

2) they did not have publications for the last ten years and have fewer than three papers before that.

The publications of these reviewers were collected from Microsoft Academic Graph (Sinha et al., 2015). Each reviewer had at least one publication, and some reviewers had as many as 34 publications. Again the abstracts were used as reviewers' profile.

We then used our proposed common topic model to assist the program committee of the conference on computer architecture, and recommended most relevant reviewers to all submissions in the conference. We randomly selected 20 submissions and with the help of the committee, we collected feedbacks from 33 reviewers on their relevance to the assigned submissions. These 33 reviewers were among the top reviewers recommended by our system for each of 20 submissions. The relevance was rated on a scale from 1 to 5, where a score of "1" meant that the paper was not relevant at all, "2" meant that the reviewer had passing familiarity with the topic of the submission, "3" meant that the reviewer knew the material well, "4" meant that the reviewer had a deep understanding of the submission, and "5" means that the reviewer was a champion candidate for the submission.

### 5.2 Baselines

We include previous approaches to paper-reviewer matching as our baselines.

- **APT 200**. Author-Person-Topic (Mimno and McCallum, 2007) is a generative probabilistic topic model which groups documents of an author into different clusters with the author's topic distribution. Clusters represent different areas of a reviewer's research.
- **Single Doc**. The Single Doc model is a probabilistic model which takes the idea of language modeling and estimates the likelihood that a submission is assigned to a reviewer given the reviewer's previous works (Mimno and McCallum, 2007).
- **Latent Dirichlet Allocation (LDA)**: LDA and its variants are the most popular topic models in expertise matching systems (Blei et al., 2003). LDA models assume that each document is a mixture of topics where each topic is a multinomial distribution over the words.
- **Hierarchical Dirichlet Process (HDP)**. HDP model is an extension of LDA (Teh et al., 2006). It is a non-parametric mixed-membership Bayesian model with variable number of topics. It is effective in choosing the number of topics to characterize a given corpus.
- **Random Walk with Restart (RWR)**. RWR is a graph model with sparsity constraints in expertise matching (Liu et al., 2014). It relies on LDA to capture reviewer-submission relevance and also takes diversity into consideration in the matching process.
- **Word Mover's Distance (WMD)**. WMD is a distance metric between two documents on the basis of pre-trained word embeddings (Kusner et al., 2015). It calculates the dissmilarity be-

| Method | Number of Topics | P@5 | P@10 | P@5 | P@10 | P@5 | P@10 |
|---|---|---|---|---|---|---|---|
| | | GT1 | | GT2 | | GT3 | |
| **Common Topic Model** | 5 | 53.7 | 43.0 | 58.5 | 49.1 | 63.7 | **55.8** |
| | 10 | **56.6** | **44.6** | **63.2** | **50.4** | **67.2** | 55.2 |
| | 20 | 54.4 | 43.4 | 59.2 | 49.5 | 63.6 | 54.7 |
| **Hidden Topic Model** | 10 | 43.4 | 41.1 | 46.4 | 45.4 | 61.8 | 46.3 |
| | 20 | 51.3 | 43.4 | 58.4 | 49.0 | 63.6 | 53.6 |
| | 30 | 47.5 | 40.0 | 49.6 | 44.0 | 56.3 | 49.4 |
| **APT200** | 200 | 41.18 | 29.71 | - | - | - | - |
| **Single Doc** | - | 44.71 | 27.35 | - | - | - | - |
| **LDA** | 50 | 41.3 | 38.4 | 51.2 | 45.0 | 55.4 | 51.5 |
| | 200 | 47.5 | 37.3 | 53.6 | 43.6 | 50.9 | 50.0 |
| | 300 | 46.2 | 38.8 | 52.0 | 46.8 | 50.0 | 45.2 |
| **HDP** | - | 45.5 | 38.0 | 48.0 | 44.5 | 55.4 | 50.0 |
| **RWR** | - | 45.3 | 43 | - | - | - | - |
| **Doc2Vec** | - | 51.7 | 41.1 | 59.2 | 46.8 | 64.5 | 51.1 |
| **WMD** | - | 36.1 | 32.4 | 42.2 | 36.8 | 46.8 | 41.8 |

Table 1: The mean precision of different baselines with optimal hyperparamters on the NIPS dataset. A reviewer is classified as relevant with a TREC score $\geq 2$.

tween two documents, which is measured by the embedding distance of aligned words in these documents.

- **Hidden Topic Model**. This model proposes to learn hidden topic vectors to measure document similarity based on word embeddings (Gong et al., 2018).
- **Doc2Vec**. Doc2Vec is a neural network model which trains document embeddings to predict component words in the documents (Le and Mikolov, 2014). In expertise matching, the Doc2Vec model is pre-trained on the corpus consisting of reviewers' previous publications. We use the trained model to generate representations for reviewers and submissions respectively. The reviewer-submission relevance is quantified by the cosine similarity of their embeddings.

**Setting**. Since our model relies on word embeddings, we pre-train embeddings on all papers published in the NIPS conference until 2017 for the matching task in NIPS dataset. Similarly for our new dataset, we collected a corpus of publications until 2018 from top computer architecture conferences for embedding training. The embedding dimension was set as 100, and these word embeddings were also used in two embedding-based baselines: word mover's distance and hidden topic model. For a fair comparison, the corpora used for word embedding training were also used to train Doc2Vec model to generate document embeddings.

### 5.3 Results on NIPS Data

The NIPS dataset provides ground truth relevance for reviewer-submission pairs, and the relevance scales from 0 to 3. A score of 0 is assigned when that the reviewer is considered to be *not relevant* and a score of 3 is assigned when the reviewer is considered to be *highly relevant*. We set a relevance threshold of 2, and considered reviewers with a score equal to or higher than this threshold to be relevant reviewers to the given submission. In our matching system, we sorted reviewers in decreasing order of the predicted relevance score for a given submission.

**Evaluation Metric**. Precision at k (P@k) is a commonly-used ranking metric on NIPS dataset. P@k is defined to be the percentage of relevant reviewers in the top-k recommendations made by the model to a submission. It is likely that the top-k recommendations made by the model contain reviewers whose relevance information is not available in the ground truth. To address this issue, we first discard reviewers that do not have a relevance information prior to calculating P@k. In our experiments, we set k to be 5 and 10. We report the average P@k over all submissions in Table 1.

We note that not all submissions in NIPS dataset have the same number of relevant reviewers and a

failure to account for this discrepancy would negatively impact the performance of a system. For example, a submission with only one relevant reviewer would result in a P@5 no higher than 20% for any model. In order to take this discrepancy into consideration, we report the performance only on submissions with at least two relevant reviewers in the columns of "GT2", and on submissions with at least three relevant reviewers in column "GT3". In "GT1", we report the performance without making this distinction.

The reviewer-submission matching results of our model on the NIPS dataset are presented in Table 1 alongside those of our chosen baselines. We note that the results for APT 200 and Single Doc were only available for GT1 and we report them as such. Some approaches including Common Topic Model, Hidden Topic Model and LDA required a hyperparameter (number of topics) to be specified. We performed experiments on NIPS data with different number of topics in Table 1. As is shown, our proposed approach consistently outperforms the strong baselines. We also note that Hidden Topic Model and Doc2Vec are competitive approaches in expertise matching compared against probabilistic models.

| Expertise Level | % of Reviewers Predicted by the System |
|---|---|
| $\geq$5 | 15.2 |
| $\geq$4 | 63.6 |
| $\geq$3 | 87.9 |
| $\geq$2 | 100 |

Table 2: Percentage of reviewers in levels of expertise to the submissions recommended by our model.

### 5.4 Results on the New Dataset

Our proposed approach has been used to assist in the paper-reviewer matching process in a tier-1 conference of computer architecture. We evaluated our approach on a new dataset constructed with reviewers' feedbacks on their assigned submissions. Based on the optimal number of topics on the NIPS dataset, we set the number of common topics to be 10 in this experiment.

We report the percentage of reviewers whose reported expertise level falls in the given range in Table 2. We note that all recommendations made by our system are reasonable considering that all reviewers had expertise levels no lower than 2. The majority (87.9%) of reviewers reported that they were familiar with the topics of the submissions assigned to them, and 63.6% of the reviewers had deep understanding of the submissions.

## 6 Error Analysis

We perform a qualitative analysis on NIPS dataset to analyze the difference of different algorithms on expertise matching. For the clarity of our discussion, we sample a submission whose abstract is shown in Table 4. We consider five models: common topic modeling (CT), hidden topic modeling (HT), LDA, Doc2Vec and WMD. We list reviewers who were considered as top candidates for this submission by the five models in Table 3. For the analyses, we used research topics from the publications of the reviewers as well as their relevance scores assigned by human annotators (i.e., their TREC scores in NIPS dataset). Reviewers are sorted in decreasing order of their relevance to the submission by five models. For example, rank 1 corresponds to the highest relevance. In Table 3, We also present the rank of each reviewer given by the models.

*Common topic model*. According to the common topic model, reviewer 3, 4 and 5 are included as its top 3 recommendations. But we note that it ranks reviewer 3 higher than reviewer 4 and 5. The relevance scoring of common topic model is based on the relevance between the common topic "Bayesian method" and reviewers' profile. Since reviewer 3 is more focused on Bayesian model, it's topic-reviewer relevance is higher than reviewer 4 and 5 who have broader research interests beyond Bayesian model and more publications. One limitation of common topic model reflected in this case is that it does not capture the authority and experience of reviewers.

*Hidden topic model*. It incorrectly considered reviewer 1 more relevant to the submission compared to reviewer 5. We note that reviewer 5 works on a broad set of research topics ranging from Bayesian model to active learning. Since the hidden topic model extracts reviewer's topics based on the topic importance without any knowledge of the submission, it is likely that Bayesian model was not selected into representative hidden topics, which results in low relevance of reviewer 5 to the given submission.

*LDA model*. LDA assigns higher relevance to reviewer 1 than reviewer 4. Reviewer 1 used

| Reviewer | TREC | Research topics | CT | HT | LDA | Doc2Vec | WMD |
|---|---|---|---|---|---|---|---|
| 1 | 1 | Speech recognition with Bayesian approach, Neural network | 6 | 2 | 4 | 1 | 7 |
| 2 | 0 | Online learning, Sequential prediction, Bayes point machine | 10 | 10 | 8 | 9 | 1 |
| 3 | 2 | Bayesian network, Variational Bayes estimation, Mixture models | 1 | 4 | 3 | 2 | 9 |
| 4 | 3 | Variational method, Bayesian learning, Markov model | 2 | 3 | 9 | 4 | 8 |
| 5 | 3 | Bayesian learning, Variational method, Active learning | 3 | 5 | 1 | 7 | 6 |

Table 3: Examples of reviewers and their relevance to the submission ranked by different algorithms.

| |
|---|
| Dirichlet Process (DP) mixture models are candidates for clustering applications where the number of clusters is unknown a priori. [...] The speedup is achieved by incorporating kd-trees into a variational Bayesian algorithm for DP mixture [...] |

Table 4: An example of abstract from a submission.

Bayesian approach, whereas it was not his research focus according to his publications. Reviewer 4 had done extensive research in general graphical models including Bayesian model. We observed that LDA fails to capture the relevance between graph model and Bayesian model since it ignores the semantic similarity between words.

*Doc2Vec model*. Doc2Vec assigned the highest relevance to reviewer 1 among all reviewers. The document representation it generates for reviewer 1's profile is similar to the representation for the submission, possibly because the key word "Bayesian" and "mixture" in the submission also occurs frequently in the profile. It suggests that Doc2Vec model might be limited to lexical overlap.

*WMD*. Reviewer 2 is included as WMD's top recommendation, whereas the research focus of reviewer 2 is sequential prediction which is irrelevant to the submission. Moreover, actually relevant reviewers 4 and 5 were excluded from WMD's top recommendations. This may have resulted from WMD's word-level similarity measure. Reviewer 2's publications had some lexical overlap with the submission (e.g., words "Bayes", "algorithm" and "learning", which have high frequency in the submission). WMD tends to assign high relevance scores due to such lexical overlap.

## 7 Future Work

This study used a basic version of a reviewer's profile to be the concatenation of the abstracts from their previous publications. A concrete direction for future work would be to consider enhancements in representing reviewers' profiles. Such efforts could consider, for instance, the temporal variation of research interests in order to capture the relevance of a given reviewer to a given topic based on the recency of the contributions to a given area. Other efforts could involve the use of a variable number of research topics for each reviewer and exploring ways to render reviewer profiles human interpretable.

## 8 Conclusion

We proposed an automated reviewer-paper matching algorithm via jointly finding the common research topics between submissions and reviewers' publications. Our model is based on word embeddings and efficiently captures the reviewer-paper relevance. It is robust to cases of vocabulary mismatch and partial topic overlap between submissions and reviewers – factors that have posed problems for previous approaches. The common topic model showed strong empirical performance on a benchmark and a newly collected dataset.

## Acknowledgments

This work is supported by the IBM-ILLINOIS Center for Cognitive Computing Systems Research (C3SR) - a research collaboration as part of the IBM AI Horizons Network. We thank the EMNLP anonymous reviewers for their constructive suggestions.

## References

PC Chairs ACL. 2019. Whats new, different and challenging in acl 2019? http://acl2019pcblog.fileli.unipi.it/?p=156. Accessed: 2019-05-19.

Krisztian Balog, Yi Fang, Maarten de Rijke, Pavel Serdyukov, Luo Si, et al. 2012. Expertise retrieval. *Foundations and Trends® in Information Retrieval*, 6(2–3):127–256.

Chumki Basu, Haym Hirsh, William W. Cohen, and Craig Nevill-Manning. 1999. Recommending papers by mining the web. In *In: Proceedings of the*

*IJCAI99 Workshop on Learning about Users*, pages 1–11.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022.

Rajarshi Das, Manzil Zaheer, and Chris Dyer. 2015. Gaussian lda for topic models with word embeddings. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 795–804.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407.

Susan T. Dumais and Jakob Nielsen. 1992. Automating the assignment of submitted manuscripts to reviewers. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '92, pages 233–244, New York, NY, USA. ACM.

Babak Falsafi, Mario Drumond, and Mark Sutherland. 2018. Isca'18 review process reflections. https://www.sigarch.org/isca18-review-process-reflections/. Accessed: 2019-05-19.

Hongyu Gong, Suma Bhat, and Pramod Viswanath. 2017. Geometry of compositionality. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Hongyu Gong, Tarek Sakakini, Suma Bhat, and Jinjun Xiong. 2018. Document similarity for texts of varying lengths via hidden topics. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2341–2351.

Weihua Hu and Jun'ichi Tsujii. 2016. A latent concept topic model for robust topic inference using word embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 380–386, Berlin, Germany. Association for Computational Linguistics.

Jian Jin, Qian Geng, Haikun Mou, and Chong Chen. 2018. Author-subject-topic model for reviewer recommendation. *Journal of Information Science*, 45(4):554–570.

Jian Jin, Qian Geng, Qian Zhao, and Lixue Zhang. 2017. Integrating the trend of research interest for reviewer assignment. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 1233–1241. International World Wide Web Conferences Steering Committee.

Ngai Meng Kou, Nikos Mamoulis, Yuhong Li, Ye Li, Zhiguo Gong, et al. 2015. A topic-based reviewer assignment system. *Proceedings of the VLDB Endowment*, 8(12):1852–1855.

Matt J. Kusner, Yu Sun, Nicholas I. Kolkin, and Kilian Q. Weinberger. 2015. From word embeddings to document distances. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*.

Jey Han Lau and Timothy Baldwin. 2016. An empirical evaluation of doc2vec with practical insights into document embedding generation. *arXiv preprint arXiv:1607.05368*.

Charlin Laurent and Richard S. Zemel. 2013. The toronto paper matching system: An automated paper-reviewer assignment system. In *Proceedings of 30th International Conference on Machine Learning*.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196.

Baochun Li and Y Thomas Hou. 2016. The new automated ieee infocom review assignment system. *IEEE Network*, 30(5):18–24.

Rui Lin, Shujie Liu, Muyun Yang, Mu Li, Ming Zhou, and Sheng Li. 2015. Hierarchical recurrent neural network for document modeling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 899–907.

Xiang Liu, Torsten Suel, and Nasir Memon. 2014. A robust model for paper reviewer assignment. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 25–32, New York, NY, USA. ACM.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv: 1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA. Curran Associates Inc.

David Mimno and Andrew McCallum. 2007. Expertise modeling for matching papers with reviewers. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, pages 500–509, New York, NY, USA. ACM.

Jennifer Nguyen, Germn Snchez-Hernndez, Nria Agell, Xari Rovira, and Cecilio Angulo. 2018. A decision support tool using order weighted averaging for conference review assignment. *Pattern Recogn. Lett.*, 105(C):114–120.

David MW Powers. 2015. What the f-measure doesn't measure: Features, flaws, fallacies and fixes. *arXiv preprint arXiv:1503.06410*.

Simon Price and Peter A Flach. 2017. Computational support for academic peer review: A perspective from artificial intelligence. *Communications of the ACM*, 60(3):7079.

Yujie Qian, Jie Tang, and Kan Wu. 2018. Weakly learning to match experts in online community. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, pages 3841–3847. AAAI Press.

Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June (Paul) Hsu, and Kuansan Wang. 2015. An overview of microsoft academic service (ma) and applications. WWW '15 Companion.

Richard Socher, Cliff C Lin, Chris Manning, and Andrew Y Ng. 2011. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 129–136.

Wenbin Tang, Jie Tang, and Chenhao Tan. 2010. Expertise matching via constraint-based optimization. In *Proceedings of the 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 01*, WI-IAT '10, pages 34–41, Washington, DC, USA. IEEE Computer Society.

Yee Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei. 2006. Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581.

Jacob A Wegelin. 2000. A survey of partial least squares (pls) methods, with emphasis on the two-block case. *Technical Report 371, Department of Statistics, University of Washington*.

Pengtao Xie, Diyi Yang, and Eric Xing. 2015. Incorporating word correlation knowledge into topic modeling. In *Proceedings of the 2015 conference of the north American chapter of the association for computational linguistics: human language technologies*, pages 725–734.

Guangxu Xun, Yaliang Li, Wayne Xin Zhao, Jing Gao, and Aidong Zhang. 2017. A correlated topic model using word embeddings. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, pages 4207–4213. AAAI Press.

# Linking artificial and human neural representations of language

**Jon Gauthier** and **Roger P. Levy**
Massachusetts Institute of Technology
Department of Brain and Cognitive Sciences
jon@gauthiers.net, rplevy@mit.edu

## Abstract

What information from an act of sentence understanding is robustly represented in the human brain? We investigate this question by comparing sentence encoding models on a brain decoding task, where the sentence that an experimental participant has seen must be predicted from the fMRI signal evoked by the sentence. We take a pre-trained BERT architecture as a baseline sentence encoding model and fine-tune it on a variety of natural language understanding (NLU) tasks, asking which lead to improvements in brain-decoding performance.

We find that none of the sentence encoding tasks tested yield significant increases in brain decoding performance. Through further task ablations and representational analyses, we find that tasks which produce syntax-light representations yield significant improvements in brain decoding performance. Our results constrain the space of NLU models that could best account for human neural representations of language, but also suggest limits on the possibility of decoding fine-grained syntactic information from fMRI human neuroimaging.

What are the neural representations which support human language understanding? Language neuroscience has converged on a set of reliable physiological markers related to language processing (Kutas and Federmeier, 2011), and a picture of where in the brain different aspects of language processing take place (Fedorenko et al., 2010). But we still largely lack a coherent picture of the *structure* and *format* of the neural representations driving language understanding.

Part of this struggle is due to a scarcity of candidate representational structures in the first place. While there are certainly enough representational theories of language understanding, many are specified at too high a level of analysis to plausibly map onto neural structures without serious further revision (Poeppel, 2012).

Studies which draw on these high-level representations must therefore also assume some link between such representations and measures of neural activity — for example, that the magnitude of neural activations should match up with the probability values derived from a computational model (Brennan et al., 2016), or measures derived from syntactic representations of the input (Pallier et al., 2011). While the success of these mapping studies demonstrates that specific summary statistics of linguistic representations have correlates in the mind, these summary statistics are not themselves candidate representations for the fundamental operations underlying language understanding.

In the meantime, research in natural language processing has produced neural network models that capture many different sorts of intelligent language understanding behavior (Collobert et al., 2011; Goldberg, 2016). These models accomplish this behavior in an implementation better matched with that of the brain, with information about their inputs distributed across a high-dimensional continuous space. Could these models be taken seriously as candidate hypotheses of how language processing could be implemented in neural hardware? Under the assumption that both human brain and neural network representations are optimally suited to solve some *task* (Anderson et al., 1990), linking these two computational systems should reveal parallel task-optimal structure shared within their representations.

This paradigm linking brain and machine has already seen substantial success in vision science. Yamins et al. (2013) first demonstrated that the activations of a convolutional neural network trained on ImageNet in response to natural images could

Source code for all analyses reported in this paper is available at http://bit.ly/nn-decoding.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 529–539,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

predict activations in a macaque monkey's visual cortex in response to the same images. This result and others have led to an increasingly detailed understanding of the contents of brain representations (Schrimpf et al., 2018) and novel artificial neural network architectures (Kubilius et al., 2018) in the domain of vision.

In language understanding, several authors have exploited neural network representations as proxies for sentence meaning, and demonstrated that human brain activations in response to sentences can match with these meaning representations at well above chance performance (see e.g. Mitchell et al., 2008; Wehbe et al., 2014; Huth et al., 2016; Pereira et al., 2018). Our aim in this paper is to further understand why these mappings are successful, uncovering the parallel *representational contents* shared between human brains and neural networks.

We evaluate the link between human brain activity and neural network models as the models are optimized for different tasks. We find that neural network models quickly diverge in their capacity to match human brain activations as they are optimized for different NLU objectives. We further locate correlates of these changes in representational content, finding that the granularity of a model's *syntactic* representations is at least partially responsible for their differences in brain decoding performance. Overall, this approach allows us to generate and validate hypotheses about the representational contents of both human brain and neural network activity.

## 1 Related work

Several papers have begun to explore the brain–machine link in language understanding, asking whether human brain activations can be matched with the activations of computational language models. Mitchell et al. (2008) first demonstrated that distributional word representations could be used to predict human brain activations, when subjects were presented with individual words in isolation. Huth et al. (2016) replicated and extended these results using distributed word representations, and Pereira et al. (2018) extended these results to sentence stimuli. Wehbe et al. (2014), Qian et al. (2016), Jain and Huth (2018), and Abnar et al. (2019) next introduced more complex word and sentence meaning representations, demonstrating that neural network language models could better account for brain activation by incorporating representations of longer-term linguistic context. Gauthier and Ivanova (2018) and Sun et al. (2019) further demonstrated that optimizing model representations for different objectives yielded substantial differences in brain decoding performance. This paper extends the neural network brain decoding paradigm both in **breadth**, studying a wide class of different task-optimal models, and in **depth**, exploring the particular representational contents of each model responsible for its brain decoding performance.

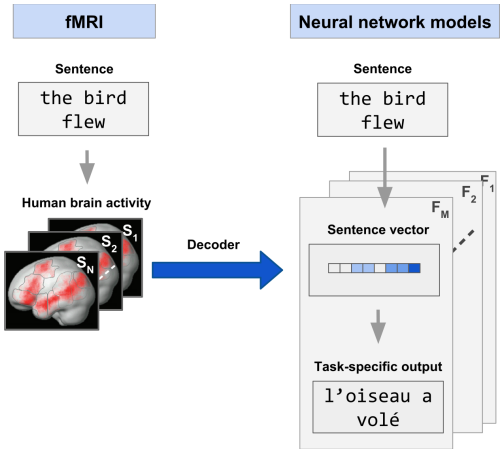


Figure 1: Brain decoding methodology. We use human brain activations in response to sentences to predict how neural networks represent those same sentences.

## 2 Methods

Figure 1 describes the high-level design of our experiments, which attempt to match human neuroimaging data with different candidate model representations of sentence inputs. Using a dataset of human brain activations recorded in response to complete sentences, we learn linear regression models which map from human brain activity to representations of the same sentences produced by different natural language understanding models.

To the extent that this linear decoding is successful, it can reveal parallel structure between brain and model representations. Consider a softmax neural network classifier model optimized for some task $T$, mapping input sentences $\mathbf{x}$ to class outputs $y$. We can factor this neural network classifier into the composition of two operations, a representational function $r(\mathbf{x})$, and an affine operator $A$:

$$p(y \mid \mathbf{x}) \propto \exp(A(r(\mathbf{x}))) \quad (1)$$

Under this factorization, $r(\mathbf{x})$ must compute representations which are linearly separable with respect

| |
|---|
| A man is a male human adult. |
| Fingers are used for grasping, writing, grooming and other activities. |
| Piranhas have very sharp teeth and are opportunistic carnivores. |
| Pressing a piano key causes a felt-tipped hammer to hit a vibrating steel string. |
| A range of mountains forms due to tectonic plate collision. |

Table 1: Randomly sampled sentence stimuli from the human neuroimaging dataset of Pereira et al. (2018).

to the classes of the task $T$.

Research in cognitive neuroscience has shown that surprisingly many features of perceptual and cognitive states are likewise linearly separable from images of human brain activity, even at the coarse spatial and temporal resolution afforded by functional magnetic resonance imaging (fMRI; see e.g. Haxby et al., 2001; Kriegeskorte et al., 2006). However, the full power of linear decoding with fMRI remains unknown within language neuroscience and elsewhere. One possibility (1) is that the representational distinctions intrinsically required to describe language understanding behavior are linearly decodable from fMRI data. If this were the case, we could use performance in brain decoding to gauge the similarity between the mental representations underlying human language understanding and those deployed within artificial neural network models. Conversely (2), if the representations supporting language understanding in the brain are not linearly decodable from fMRI, we should be able to demonstrate this fact by showing specific ablations of sentence representation models do not degrade in brain decoding performance. Thus, the brain decoding framework offers possibilities both for (1) discriminating among NLU tasks as faithful characterizations of human language understanding, and for (2) understanding potential limitations of fMRI imaging and linear decoding methods. We explore both of these possibilities in this paper.

Section 2.1 describes the human neuroimaging data used as the source of this learned mapping. Section 2.2 next describes how we derive the target representations of sentence inputs from different natural language understanding models. Finally, Section 2.3 describes our method for deriving and evaluating mappings between the two representational spaces.

### 2.1 Human neuroimaging data

We use the human brain images collected by Pereira et al. (2018, experiment 2).[1] Pereira et al. visually presented 384 natural-language sentences to 8 adult subjects. The sentences (examples in Table 1) consisted of simple encyclopedic statements about natural kinds, written by the authors. The subjects were instructed to carefully read each sentence, presented one at a time, and think about its meaning. As they read the sentences, the subjects' brain activity was recorded with functional magnetic resonance imaging (fMRI). For each subject and each sentence, the fMRI images consist of a $\sim 200,000$-dimensional vector describing the approximate neural activity within small 3D patches of the brain, known as voxels.[2] We collect these vectors in a single matrix and compress them to $d_B = 256$ dimensions using PCA, yielding a matrix $B_i \in \mathbb{R}^{384 \times d_B}$ for each subject $i$.[3]

| Task | Dataset | Domain | # train | Avg sent len. | # types |
|---|---|---|---|---|---|
| Paraphrase classification | Quora Question Pairs | social QA | 364k | 22.1 | 103k |
| Question answering | SQuAD 2.0 (Rajpurkar et al., 2018) | wiki | 130k | 11.2 | 43.9k |
| Natural language inference | MNLI (Williams et al., 2017) | mixed | 393k | 16.8 | 83.3k |
| Sentiment analysis | SST-2 (Socher et al., 2013) | movie reviews | 67.3k | 9.41 | 14.8k |

Table 2: Details of the tasks used for fine-tuning.

### 2.2 Sentence representation models

We will match the human brain activation data described above with a suite of different sentence representations. Our primary concern in this evaluation is to compare alternative *tasks* and the representational contents they demand, rather than comparing neural network architectures. For this reason, we draw sentence representations from a unified neural network architecture — the bidirectional Transformer model BERT (Devlin et al., 2018) — as we optimize it to perform different tasks.

The BERT model uses a series of multi-head attention operations to compute context-sensitive representations for each token in an input sentence. The model is pre-trained on two tasks: (1) a cloze language modeling task, where the model is given a complete sentence containing several masked words and asked to predict the identity of a particu-

[1] This dataset is publicly available at https://osf.io/crwz7/.

[2] For more information on data collection and fMRI analysis, please see Pereira et al. (2018).

[3] The PCA projections retain more than 95% of the variance among sentence responses within each subject.

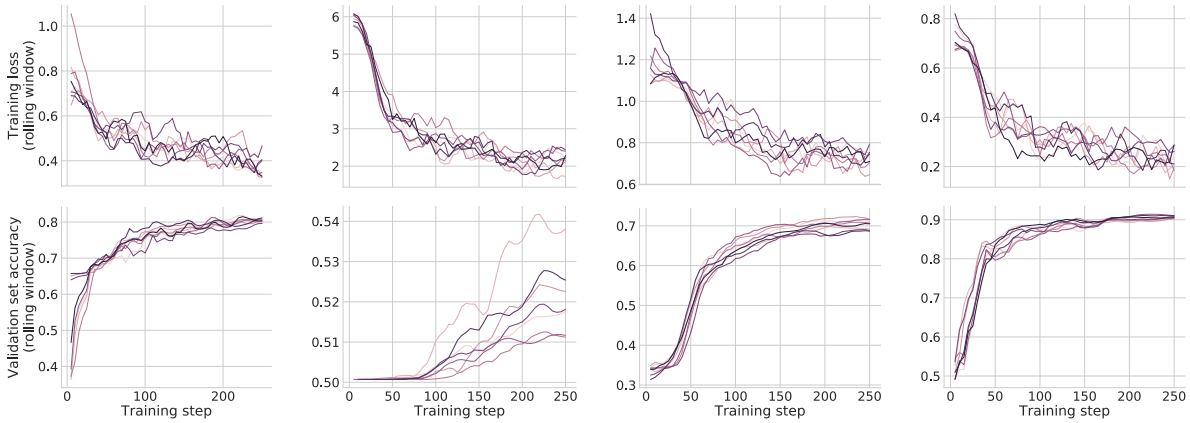


(a) QQP (b) SQuAD (c) MNLI (d) SST

Figure 2: Learning curves (training set loss and evaluation set accuracy) for NLU model fine-tuning.

lar masked word; and (2) a next-sentence prediction task, where the model is given two sentences and asked to predict whether the sentences are immediately adjacent in the original language modeling data.[4] For our purposes, this pre-training process produces a set of BERT parameters $\Theta_{\text{LM}}$ jointly optimized for these two objectives, consisting of word embeddings and attention mechanism parameters.

For an input token sequence $w_1, \ldots, w_T$, the output of the BERT model is a corresponding sequence of contextualized representations of each token. We derive a single sentence representation vector by prepending a constant token $w_0 = \texttt{[CLS]}$, and extracting its corresponding output vector at the final layer, following Devlin et al. (2018).[5]

We first extract sentence representations from the pre-trained model parameterized by $\Theta_{\text{LM}}$. We let $C_{\text{LM}} \in \mathbb{R}^{384 \times d_H}$ refer to the matrix of sentence representations drawn from this pre-trained BERT model, where $d_H$ is the dimensionality of the BERT hidden layer representation.

### 2.2.1 Fine-tuned models

We use the code and pre-trained weights released by Devlin et al. (2018) to fine-tune the BERT model on a suite of natural language processing classification tasks, shown in Table 2.[6] This fine-tuning process jointly optimizes the pre-trained word embeddings and attention weights drawn from $\Theta_{\text{LM}}$, along with a task-specific classification model which accepts the sentence representations produced by the BERT model (the vector corresponding to the prepended `[CLS]` token) as input.

We fine-tune the pretrained BERT model on a set of popular shared NLU tasks, shown in Table 2, with fixed hyperparameters across tasks (available in Appendix B). Each fine-tuning operation is run for 250 iterations, before which all models show substantial improvements on the fine-tuning task. Figure 2 shows the learning curves for each of the models fine-tuned by this procedure.

We execute 8 different runs for each fine-tuning task. Each run $\ell$ of each fine-tuning task $j$ produces a set of final parameters $\Theta_{j\ell}$. We can derive a set of sentence representations $C_{j\ell}$ from each fine-tuned model by the same logic as above, feeding in each sentence with a prepended `[CLS]` token and extracting the contextualized `[CLS]` representation.

For our purposes, the product of each fine-tuning run is a set of sentence representations $C_{j\ell} \in \mathbb{R}^{384 \times d_H}$.

**Custom tasks** In order to better understand *why* models might fail or succeed at brain decoding, we also produced several custom fine-tuning tasks. Each task was a modified form of the standard BERT cloze language modeling task, manipulated to strongly select for or against some particular aspect of linguistic representation.[7]

[4] See Devlin et al. (2018); Vaswani et al. (2017) for further details on model architecture and training.

[5] We also repeated the experiments of this paper using sentence representations computed by uniformly averaging BERT's contextualized token representations of each input sentence, and found similar qualitative results as will be presented below.

[6] Details on the fine-tuning procedure are available in Devlin et al. (2018).

[7] The training and test data for these tasks were generated from the Toronto Books Corpus (Zhu et al., 2015), a subset of the data used to train the original BERT model. Further details and examples for these tasks are given in Appendix C.

**Scrambled language modeling** We first design two language modeling tasks to select against fine-grained syntactic representation of inputs. We randomly shuffle words from the corpus samples used for language modeling, to remove all first-order cues to syntactic structure. Our first custom task, `LM-scrambled`, deals with sentence inputs where words are shuffled within sentences; our second task, `LM-scrambled-para`, uses inputs where words are shuffled within their containing paragraphs in the corpus. [8]

By shuffling inputs in this way, we effectively turn the cloze task into a bag-of-words language modeling task: given a set of words from a sentence or a random draw of words from a paragraph, the model must predict a missing word. After optimizing models on these scrambled tasks, we design a probe to validate the effects of the task on the model's syntactic representations. This probe is detailed in Section 3.2.2.

**Part-of-speech language modeling** We next design a task, `LM-pos`, to select against fine-grained semantic representation of inputs. We do this by requiring a model to predict only the *part of speech* of a masked word, rather than the word itself. This manipulation removes pressure for the model to distinguish predictions between target words in the same syntactic class.

We repeat fine-tuning runs on each of these custom tasks 4 times per task, and see substantial improvements in held-out task performance for each of these custom tasks after just 250 steps of training. After fine-tuning, we extract sentence representation matrices $C_{j\ell}$ from each run $\ell$ of each these models.

**Language modeling control** As a control, we also continue training on the original BERT language modeling objectives using text drawn from the Books Corpus (Zhu et al., 2015). We run 4 fine-tuning runs of this task, yielding representations $C_{\text{LM},\ell}$.

#### 2.2.2 Word vector baseline

As a baseline comparison, we also include sentence representations computed from GloVe word vectors (Pennington et al., 2014). Unlike BERT's word representations, these word vectors are insensitive to their surrounding sentential context. These word vectors have nevertheless successfully served as sentence meaning representations in prior studies (Pereira et al., 2018; Gauthier and Ivanova, 2018). We let $C_{\text{GloVe}}(w_1, \ldots, w_T) = \frac{1}{T} e(w_t)$, where $e(w_i)$ retrieves the GloVe embedding for word $w_i$.[9]

### 2.3 Brain decoding

We next learn a suite of *decoders*: regression models mapping from descriptions of human brain activation to model activations in response to sentences. Let $B_i \in \mathbb{R}^{384 \times d_B}$ represent the brain activations of subject $i$ in response to the 384 sentences in our evaluation set. For each subject and each sentence representation available (including both language model representations $C_{\text{LM}}$ and fine-tuned model representations $C_{j\ell}$), we learn a linear map $G_{i \to j\ell} : \mathbb{R}^{d_H \times d_B}$ between the two spaces which minimizes the regularized regression loss

$$J_{ij\ell} = ||G_{i \to j\ell} B_i - C_{j\ell}||_2^2 + \beta \, ||G_{i \to j\ell}||_2^2 \quad (2)$$

where $\beta$ is a regularization hyperparameter. For each subject's collection of brain images and each target model representation, we train and evaluate the above regression model with nested 8-fold cross-validation (Cawley and Talbot, 2010). The regression models are evaluated under two metrics: mean squared error (MSE) in prediction of model activations, and average rank (AR):

$$\text{AR}(i, j, \ell) = \frac{1}{384} \sum_k \text{rank}((G_{i \to j\ell} B_i)[k], C_{j\ell}[k]) \quad (3)$$

where $\text{rank}((G_{i \to j\ell} B_i)[k], C_{j\ell}[k])$ gives the rank of a ground-truth sentence representation $C_{j\ell}[k]$ in the list of nearest neighbors of a predicted sentence representation $(G_{i \to j\ell} B_i)[k]$, ordered by increasing cosine distance.

These two metrics serve complementary roles: the MSE metric strictly evaluates the ability of human brain activations to exactly match the representational geometry of model activations, while the AR metric simply requires that the brain activations be able to support the relevant meaning contrasts between the 384 sentences tested.

[8] This shuffling method removes first-order cues to constituency: for example, the fact that *the table* appears directly to the right of *on* suggests that they are members of a single constituent. It is still possible that the model can exploit second-order cues to structure. For example, if two nouns *ranch* and *rancher* appear in the same sentence as *dressing*, we can still guess that *ranch* is more likely to modify *dressing*.

[9] We use publicly available GloVe vectors computed on Common Crawl, available in the spaCy toolkit as `en_vectors_web_lg`.

## 3 Results

We first present the performance of all of the BERT models tested and the GloVe baseline in Figure 3. This figure makes apparent a number of surprising findings, which we validate by paired $t$-tests.[10]

1. On average, fine-tuning on the standard NLU tasks yields *increased error* in brain decoding under both metrics relative to the BERT baseline (MSE, $t \approx 14.8, p < 3 \times 10^{-36}$; AR, $t \approx 17, p < 3 \times 10^{-42}$). This trend is significant for each model individually, except QQP (MSE, $t \approx -2.2, p > 0.03$; AR, $t \approx 0.79, p > 0.4$).

2. Fine-tuning on the LM-scrambled-para custom task yields *decreased error* in brain decoding under both metrics relative to the BERT baseline (MSE, $t \approx -33.6, p < 3 \times 10^{-45}$; AR, $t \approx -26.9, p < 5 \times 10^{-39}$) and GloVe vectors (MSE, $t \approx -23.4, p < 4 \times 10^{-35}$; AR, $t \approx -12.1, p < 6 \times 10^{-19}$).

3. Fine-tuning on both the control language-modeling task and the LM-pos custom task yields ambiguous results: *decreased MSE* and *no significant change in AR* relative to the BERT baseline (LM-pos: MSE, $t \approx -2.90, p < 0.007$; AR, $t \approx 2.10, p > 0.04$; LM: MSE, $t \approx -5.68, p < 4 \times 10^{-7}$; AR, $t \approx -2.36, p > 0.02$).

### 3.1 Learning dynamics

When during training do these models diverge in brain decoding performance? We repeat our brain decoding evaluation on model snapshots taken every 5 steps during fine-tuning, and chart brain decoding performance over time for each model in Figure 4. We find that models rapidly diverge in brain decoding performance, but remain mostly stable after about 100 fine-tuning steps. This phase of rapid change in brain decoding performance is generally matched with a phase of rapid change in task performance (compare each line in Figure 4 with the learning curves in Figure 2).

[10]Each sample in our statistical tests compares the brain decoding performance matching a subject's brain image with model representations before and after fine-tuning on a particular task. See Figures 8 and 9 of Appendix A for further visualizations. Results are reported throughout with a significance level $\alpha = 0.01$.

### 3.2 Representational analysis

We next investigate the structure of the model representations, and find that differing fidelity of *syntactic* representation can explain some major qualitative differences between the models in brain decoding performance.

#### 3.2.1 Representational similarity

We first investigate coarse-grained model similarity with representational similarity analysis (Kriegeskorte et al., 2008), which measures models' pairwise distance judgments as a proxy for how well the underlying representations are aligned in their contents. For each fine-tuning run $\ell$ on each task $j$, we compute pairwise cosine similarities between each pair of sentence representation rows in $C_{j\ell}$, yielding a vector $D_{j\ell} \in \mathbb{R}^{(384 \text{ choose } 2)}$.

We measure the similarity between the representations derived from a run $(j, \ell)$ and some other run $(j', \ell')$ by computing the Spearman correlation coefficient $\rho(D_{j\ell}, D_{j'\ell'})$. These correlation values are graphed as a heatmap in Figure 5, where each cell averages over different runs of the two corresponding models. This heatmap yields several clear findings:

1. The language modeling fine-tuning runs (especially the two LM-scrambled tasks) are the only models which have reliable high correlations between one another.

2. Language modeling tasks yield representations which make similar sentence-sentence distance predictions between different runs on the same task, while the rest of the models are less coherent across runs (see matrix diagonal).

The scrambled LM tasks produce sentence representations which are reliably coherent across runs (Figure 5), and produce reliable improvements in brain decoding performance (Figure 3). What is it about this task which yields such reliable results? We attempt to answer this question in the following section.

#### 3.2.2 Syntactic probe

Because the scrambled LM tasks were designed to remove all first-order cues to syntactic constituency from the input, we hypothesized that the models trained on this task were succeeding due to their resulting coarse syntactic representations. We tested

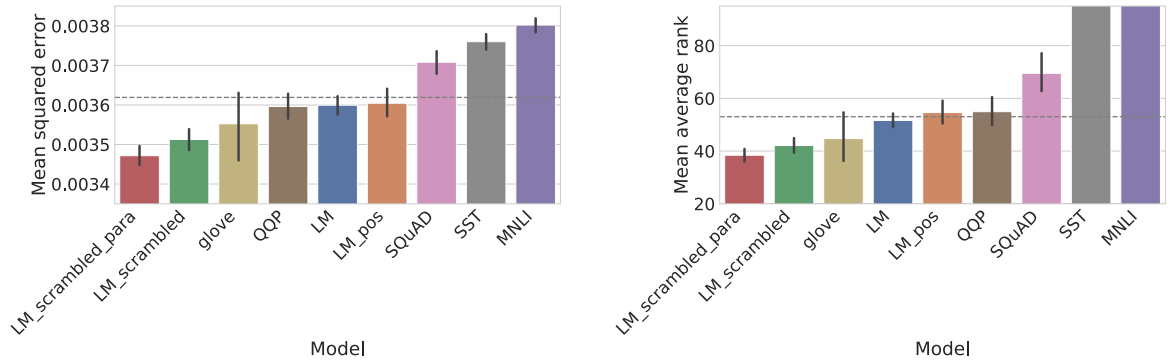


(a) Mean squared error metric

(b) Average rank metric

Figure 3: Brain decoding performance for fine-tuned BERT models and the GloVe baseline. Error bars are bootstrapped 95% confidence intervals, pooling across decoders matching different subjects (8 total) and different BERT fine-tuning runs (up to 8 runs per model). Dashed gray line shows the average brain decoding performance of the pretrained BERT language model.

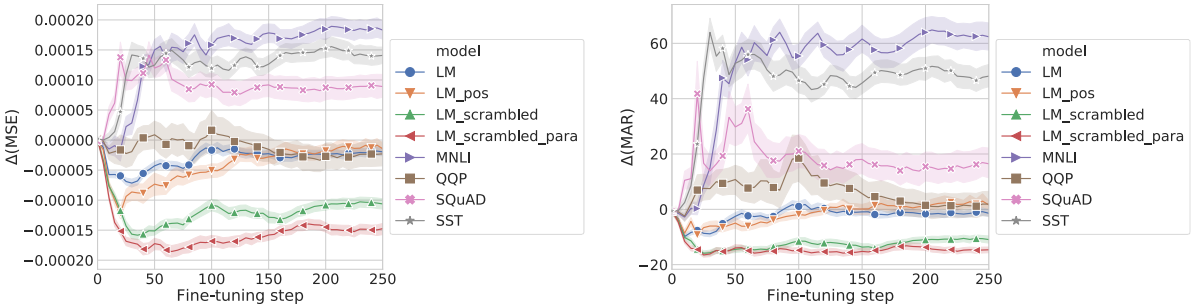


(a) Mean squared-error metric

(b) Average rank metric

Figure 4: Brain decoding performance trajectories over fine-tuning time, graphed relative to the brain decoding performance of the pre-trained BERT language model. Performance rapidly diverges and then stabilizes within tens of fine-tuning steps. Shaded regions represent 95% confidence intervals, pooling across 8 subjects and up to 8 fine-tuning runs per model.

this idea using the structural probe method of Hewitt and Manning (2019). This method measures the degree to which word representations can reproduce syntactic analyses of sentences. We used dependency-parsed sentences from the Universal Dependencies (UD) English Web Treebank corpus (Silveira et al., 2014) to evaluate the performance of each fine-tuned BERT model with a structural probe.

For each fine-tuned BERT model (task $j$, fine-tuning run $\ell$) and each sentence $w_1, \ldots, w_T$, let $\tilde{w}_i$ denote the context-sensitive representation of word $w_i$ under the model parameters $\Theta_{j\ell}$. The structural probe method attempts to derive a distance measure between context-sensitive word representations, parameterized by a transformation matrix $B$, which approximates the number of grammatical dependencies separating the two words (Hewitt and Manning, 2019). Concretely, for any two words $w_i, w_j$ in a parsed sentence, we learn a parameter matrix $B$ such that

$$\left((B(\tilde{w}_i - \tilde{w}_j))^T (B(\tilde{w}_i - \tilde{w}_j))\right)^2 \approx |w_i \leftrightarrow w_j| \quad (4)$$

where $|w_i \leftrightarrow w_j|$ denotes the number of edges separating $w_i$ and $w_j$ in a dependency parse of the sentence.

We learn this parameter matrix $B$ for a set of training sentences randomly sampled from the UD corpus, and then apply the distance measure above to model representations for a set of held-out test sentences. For any sentence $w_1, \ldots, w_T$, the measure induces a $T \times T$ pairwise distance matrix, where each entry $(i, j)$ predicts the distance (in grammatical dependencies) between words $i$ and $j$. By applying a minimum spanning tree algorithm to this matrix, we derive an (undirected) parse tree for

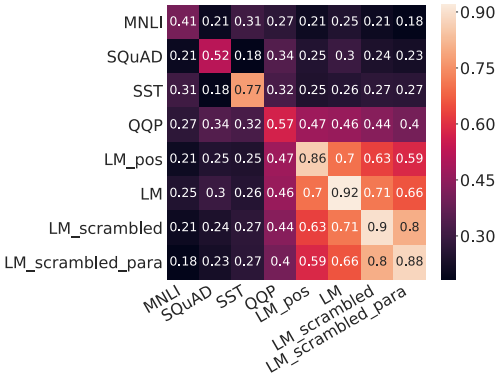


Figure 5: Representational similarity of the sentence encodings produced by each model (between -1 and 1; higher is more similar). See Section 3.2.1 for details.

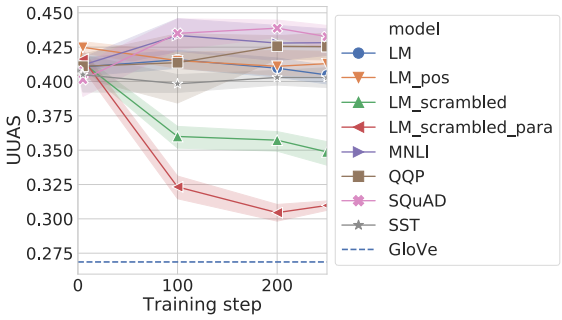


Figure 6: Syntactic probe evaluation across fine-tuning time (section 3.2.2).

the sentence which best matches the predictions of the distance measure. We measure the accuracy of the reconstructed tree by calculating its unlabeled attachment score (UAS) relative to ground-truth parses from the UD corpus.

We apply the probe described above to every fine-tuning run of each model, and to baseline GloVe representations. We expected the GloVe representations to perform worst, since they cannot encode any context-sensitive features of input words. The probe results are graphed over fine-tuning time in Figure 6, relative to a probe induced from the GloVe representations (dashed blue line). This analysis shows that the models optimized for LM-scrambled and LM-scrambled-para — the models which improve in brain decoding performance — progressively worsen under this syntactic probe measure during fine-tuning. Their probe performance remains well above the performance of the GloVe baseline, however.

Figure 7 shows a representative sample sentence with parses induced from the syntactic probes of

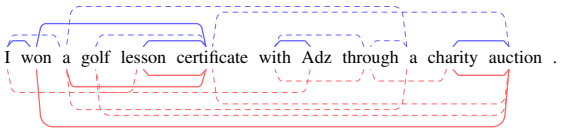


Figure 7: Example sentence with undirected syntactic probe parses induced from LM-scrambled representations (blue, above) and GloVe representations (red, below). Solid arcs represent correct dependency predictions; dashed lines represent errors.

LM-scrambled (after 250 fine-tuning steps) and the GloVe baseline. While both parses make many mistaken attachments (dashed arcs), the parse induced from LM-scrambled (blue arcs) makes better guesses about local attachment decisions than the parse from GloVe (red arcs), which seems to simply link identical and thematically related words. This is the case even though LM-scrambled is never able to exploit information about the relative positions of words during its training. Overall, this suggests that much (but not all) of the syntactic information initially represented in the baseline BERT model is *discarded* during training on the scrambled language modeling tasks. Surprisingly, this loss of syntactic information seems to yield improved performance in brain decoding.

## 4 Discussion

The brain decoding paradigm presented in this paper has led us to a set of scrambled language modeling tasks which best match the structure of brain activations among the models tested. Optimizing for these scrambled LM tasks produces a rapid but stable divergence in representational contents, yielding improvements in brain decoding performance (Figures 3 and 4) and reliably coherent predictions in pairwise sentence similarity (Figure 5). These changes are matched with a clear loss of syntactic information (Figure 6), though some minimal information about local grammatical dependencies is decodable from the model's context-sensitive word representations (Figure 7).

We do not take these results to indicate that human neural representations of language do not encode syntactic information. Rather, we see several possible explanations for these results:

**Limitations of fMRI.** Functional magnetic resonance imaging — the brain imaging method used to collect the dataset studied in this paper — may be too temporally coarse to detect traces of the syntac-

tic computations powering language understanding in the human brain.

This idea may conflict with several findings in the neuroscience of language. For example, Brennan et al. (2016) compared how language models with different granularities of syntactic representation map onto human brain activations during passive listening of English fiction. They derived word-level surprisal estimates from n-gram models (which have no explicit syntactic representation) and PCFG models (which explicitly represent syntactic constituency). In a stepwise regression analysis, they demonstrated that the surprisal estimates drawn from the PCFG model explain variance in fMRI measures of brain activation not already explained by estimates drawn from the n-gram model.

Pallier et al. (2011) examined a different hypothesis linking mental and neural representations of language. They presented subjects with strings of words which contain valid syntactic constituents of different lengths. They assumed that, since subjects will attempt to construct syntactic analyses of the word strings, the length of the possible syntactic constituents in any stimulus should have some correlate in subjects' neural activations. They found a reliable relationship between the size of the available constituents in the input and region-specific brain activations as measured by fMRI.

Our results are compatible with the idea that *specific syntactic features* like those discussed above are still represented in the brain at the time scale of fMRI. Figures 6 and 7 demonstrate, in fact, that the `LM-scrambled` models still retain some syntactic information (or correlates thereof), in that they clearly outperform a baseline model in predicting the syntactic parses of sentences.

While these brain mapping studies have detected particular summary features of syntactic computation in the brain, these summary features do not constitute complete proposals of syntactic processing. In contrast, each of the models trained in this paper constitutes an independent candidate algorithmic description of sentence representation. These candidate descriptions can be probed (as in Section 3.2.2) to reveal exactly *why* brain decoding fails or succeeds in any case.

Our paradigm thus enables us to next ask: what specific syntactic features are responsible for the improved performance of the `LM-scrambled` models? By further probing the models and designing ablated datasets, we plan to narrow down the particular phenomena responsible for the findings presented here. These results should act as a source of finer-grained hypotheses about what sort of syntactic information is preserved at coarse temporal resolutions, and allow us to resolve the conflict between our results those of Pallier et al. (2011) and Brennan et al. (2016), among others.

**Limitations of the linking hypothesis.** Our linear linking hypothesis (presented in Section 2.3) that representations of syntactic structure are encoded entirely in the linear geometry of both neural networks and human brains. It is likely that some syntactic information — among other features of the input — *are* conserved in the fMRI signal, but not readable by a linear decoder. Future work should investigate how more complex transforms linking brain and machine can reveal parallel structure between the two systems.

**Limitations of the data.** The fMRI data used in this study (presented in Section 2.1) was collected as subjects read sentences and were asked to simply think about their meaning. This vague task specification may have led subjects to engage only superficially with the sentences. If this were the case, these shallow mental representations might present us with correspondingly shallow neural representations — just the sort of representations which might be optimal for the simple tasks such as `LM-scrambled` and `LM-scrambled-para`. Future work should integrate brain images derived from different behavioral tasks, and study which model–brain relationships are conserved across these behaviors. Such studies could illuminate the degree to which there are genuinely task-general language representations in the mind.

Our broader framework of analysis promises to reveal further insights about the parallel contents between artificial and human neural representations of language. In this spirit, we have released our complete analysis framework as open source code for the research community, available at `http://bit.ly/nn-decoding`.

## Acknowledgments

JG gratefully acknowledges support from the Open Philanthropy Project, and RPL gratefully acknowledges support from a Newton Brain Science Research Seed Award and from the MIT–SenseTime Alliance on Artificial Intelligence.

## References


Samira Abnar, Lisa Beinborn, Rochelle Choenni, and Willem Zuidema. 2019. Blackbox meets blackbox: Representational similarity and stability analysis of neural language models and brains. *arXiv preprint arXiv:1906.01539*.

John Robert Anderson et al. 1990. *The Adaptive Character of Thought*. Psychology Press.

Jonathan R Brennan, Edward P Stabler, Sarah E Van Wagenen, Wen-Ming Luh, and John T Hale. 2016. Abstract linguistic structure correlates with temporal activity during naturalistic comprehension. *Brain and language*, 157:81–94.

Gavin C Cawley and Nicola LC Talbot. 2010. On overfitting in model selection and subsequent selection bias in performance evaluation. *Journal of Machine Learning Research*, 11(Jul):2079–2107.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(Aug):2493–2537.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Evelina Fedorenko, Po-Jang Hsieh, Alfonso Nieto-Castañón, Susan Whitfield-Gabrieli, and Nancy Kanwisher. 2010. New method for fmri investigations of language: defining rois functionally in individual subjects. *Journal of neurophysiology*, 104(2):1177–1194.

Jon Gauthier and Anna Ivanova. 2018. Does the brain represent words? an evaluation of brain decoding studies of language understanding. In *Proceedings of the 2nd Conference on Cognitive Computational Neuroscience*, Philadelphia, Pennsylvania.

Yoav Goldberg. 2016. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, 57:345–420.

James V Haxby, M Ida Gobbini, Maura L Furey, Alumit Ishai, Jennifer L Schouten, and Pietro Pietrini. 2001. Distributed and overlapping representations of faces and objects in ventral temporal cortex. *Science*, 293(5539):2425–2430.

John Hewitt and Christopher D Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, Minneapolis, Minnesota, USA*, volume 2.

Matthew Honnibal and Ines Montani. 2017. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. *To appear*.

Alexander G Huth, Wendy A de Heer, Thomas L Griffiths, Frédéric E Theunissen, and Jack L Gallant. 2016. Natural speech reveals the semantic maps that tile human cerebral cortex. *Nature*, 532(7600):453.

Shailee Jain and Alexander Huth. 2018. Incorporating context into language encoding models for fMRI. *bioRxiv*, page 327601.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Nikolaus Kriegeskorte, Rainer Goebel, and Peter Bandettini. 2006. Information-based functional brain mapping. *Proceedings of the National Academy of Sciences*, 103(10):3863–3868.

Nikolaus Kriegeskorte, Marieke Mur, and Peter A Bandettini. 2008. Representational similarity analysis-connecting the branches of systems neuroscience. *Frontiers in systems neuroscience*, 2:4.

Jonas Kubilius, Martin Schrimpf, Aran Nayebi, Daniel Bear, Daniel LK Yamins, and James J DiCarlo. 2018. Cornet: Modeling the neural mechanisms of core object recognition. *BioRxiv*, page 408385.

Marta Kutas and Kara D Federmeier. 2011. Thirty years and counting: finding meaning in the n400 component of the event-related brain potential (erp). *Annual review of psychology*, 62:621–647.

Tom M Mitchell, Svetlana V Shinkareva, Andrew Carlson, Kai-Min Chang, Vicente L Malave, Robert A Mason, and Marcel Adam Just. 2008. Predicting human brain activity associated with the meanings of nouns. *Science*, 320(5880):1191–1195.

Christophe Pallier, Anne-Dominique Devauchelle, and Stanislas Dehaene. 2011. Cortical representation of the constituent structure of sentences. *Proceedings of the National Academy of Sciences*, 108(6):2522–2527.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Francisco Pereira, Bin Lou, Brianna Pritchett, Samuel Ritter, Samuel J Gershman, Nancy Kanwisher, Matthew Botvinick, and Evelina Fedorenko. 2018. Toward a universal decoder of linguistic meaning from brain activation. *Nature communications*, 9(1):963.

David Poeppel. 2012. The maps problem and the mapping problem: two challenges for a cognitive neuroscience of speech and language. *Cognitive neuropsychology*, 29(1-2):34–55.

Peng Qian, Xipeng Qiu, and Xuanjing Huang. 2016. Bridging lstm architecture and the neural dynamics during reading. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 1953–1959. AAAI Press.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*.

Martin Schrimpf, Jonas Kubilius, Ha Hong, Najib J Majaj, Rishi Rajalingham, Elias B Issa, Kohitij Kar, Pouya Bashivan, Jonathan Prescott-Roy, Kailyn Schmidt, et al. 2018. Brain-score: which artificial neural network for object recognition is most brain-like? *BioRxiv*, page 407007.

Natalia Silveira, Timothy Dozat, Marie-Catherine de Marneffe, Samuel Bowman, Miriam Connor, John Bauer, and Christopher D. Manning. 2014. A gold standard dependency corpus for English. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Jingyuan Sun, Shaonan Wang, Jiajun Zhang, Chengqing Zong, et al. 2019. Towards sentence-level brain decoding with distributed representations. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI-2019)*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Leila Wehbe, Ashish Vaswani, Kevin Knight, and Tom Mitchell. 2014. Aligning context-based statistical models of language with brain activity during reading. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 233–243.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Daniel L Yamins, Ha Hong, Charles Cadieu, and James J DiCarlo. 2013. Hierarchical modular optimization of convolutional networks achieves representations similar to macaque it and human ventral stream. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3093–3101. Curran Associates, Inc.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *The IEEE International Conference on Computer Vision (ICCV)*.

# Neural Text Summarization: A Critical Evaluation

**Wojciech Kryściński, Nitish Shirish Keskar, Bryan McCann,**
**Caiming Xiong, Richard Socher**
Salesforce Research
{kryscinski,nkeskar,bmccann,cxiong,rsocher}@salesforce.com

## Abstract

Text summarization aims at compressing long documents into a shorter form that conveys the most important parts of the original document. Despite increased interest in the community and notable research effort, progress on benchmark datasets has stagnated. We critically evaluate key ingredients of the current research setup: datasets, evaluation metrics, and models, and highlight three primary shortcomings: 1) automatically collected datasets leave the task underconstrained and may contain noise detrimental to training and evaluation, 2) current evaluation protocol is weakly correlated with human judgment and does not account for important characteristics such as factual correctness, 3) models overfit to layout biases of current datasets and offer limited diversity in their outputs.

## 1 Introduction

Text summarization aims at compressing long textual documents into a short, human readable form that contains the most important information from the source. Two strategies of generating summaries are *extractive* (Dorr et al., 2003; Nallapati et al., 2017), where salient fragments of the source document are identified and directly copied into the summary, and *abstractive* (Rush et al., 2015; See et al., 2017), where the salient parts are detected and paraphrased to form the final output.

The number of summarization models introduced every year has been increasing rapidly. Advancements in neural network architectures (Sutskever et al., 2014; Bahdanau et al., 2015; Vinyals et al., 2015; Vaswani et al., 2017) and the availability of large scale data (Sandhaus, 2008; Nallapati et al., 2016a; Grusky et al., 2018) enabled the transition from systems based on expert knowledge and heuristics to data-driven approaches powered by end-to-end deep neural models. Current approaches to text summarization utilize advanced attention and copying mechanisms (See et al., 2017; Tan et al., 2017; Cohan et al., 2018), multi-task and multi-reward training techniques (Guo et al., 2018; Pasunuru and Bansal, 2018; Kryściński et al., 2018), reinforcement learning strategies (Paulus et al., 2017; Narayan et al., 2018b; Dong et al., 2018; Wu and Hu, 2018), and hybrid extractive-abstractive models (Liu et al., 2018; Hsu et al., 2018; Gehrmann et al., 2018; Chen and Bansal, 2018). Many of the introduced models are trained on the CNN/DailyMail (Nallapati et al., 2016a) news corpus, a popular benchmark for the field, and are evaluated based on $n$-gram overlap between the generated and target summaries with the ROUGE package (Lin, 2004).

Despite substantial research effort, the progress on these benchmarks has stagnated. State-of-the-art models only slightly outperform the Lead-3 baseline, which generates summaries by extracting the first three sentences of the source document. We argue that this stagnation can be partially attributed to the current research setup, which involves uncurated, automatically collected datasets and non-informative evaluations protocols. We critically evaluate our hypothesis, and support our claims by analyzing three key components of the experimental setting: datasets, evaluation metrics, and model outputs. Our motivation is to shift the focus of the research community into developing a more robust research setup for text summarization.

## 2 Related Work

### 2.1 Datasets

To accommodate the requirements of modern data-driven approaches, several large-scale datasets have been proposed. The majority of



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 540–551,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

available corpora come from the news domain. Gigaword (Graff and Cieri, 2003) is a set of articles and corresponding titles that was originally used for headline generation (Takase et al., 2016), but it has also been adapted to single-sentence summarization (Rush et al., 2015; Chopra et al., 2016). NYT (Sandhaus, 2008) is a collection of articles from the New York Times magazine with abstracts written by library scientists. It has been primarily used for extractive summarization (Hong and Nenkova, 2014; Li et al., 2016) and phrase-importance prediction (Yang and Nenkova, 2014; Nye and Nenkova, 2015). The CNN/DailyMail (Nallapati et al., 2016a) dataset consists of articles with summaries composed of highlights from the article written by the authors themselves. It is commonly used for both *abstractive* (See et al., 2017; Paulus et al., 2017; Kryściński et al., 2018) and *extractive* (Dong et al., 2018; Wu and Hu, 2018; Zhou et al., 2018) neural summarization. The collection was originally introduced as a Cloze-style QA dataset by Hermann et al. (2015). XSum (Narayan et al., 2018a) is a collection of articles associated with one, single-sentence summary targeted at abstractive models. Newsroom (Grusky et al., 2018) is a diverse collection of articles sourced from 38 major online news outlets. This dataset was released together with a leaderboard and held-out testing split.

Outside of the news domain, several datasets were collected from open discussion boards and other portals offering structure information. Reddit TIFU (Kim et al., 2018) is a collection of posts scraped from Reddit where users post their daily stories and each post is required to contain a Too Long; Didn't Read (TL;DR) summary. WikiHow (Koupaee and Wang, 2018) is a collection of articles from the WikiHow knowledge base, where each article contains instructions for performing procedural, multi-step tasks covering various areas, including: arts, finance, travel, and health.

### 2.2 Evaluation Metrics

Manual and semi-automatic (Nenkova and Passonneau, 2004; Passonneau et al., 2013) evaluation of large-scale summarization models is costly and cumbersome. Much effort has been made to develop automatic metrics that would allow for fast and cheap evaluation of models.

The ROUGE package (Lin, 2004) offers a set of automatic metrics based on the lexical overlap between candidate and reference summaries. Overlap can be computed between consecutive ($n$-grams) and non-consecutive (skip-grams) subsequences of tokens. ROUGE scores are based on exact token matches, meaning that computing overlap between synonymous phrases is not supported.

Many approaches have extended ROUGE with support for synonyms and paraphrasing. ParaEval (Zhou et al., 2006) uses a three-step comparison strategy, where the first two steps perform optimal and greedy paraphrase matching based on paraphrase tables before reverting to exact token overlap. ROUGE-WE (Ng and Abrecht, 2015) replaces exact lexical matches with a soft semantic similarity measure approximated with the cosine distances between distributed representations of tokens. ROUGE 2.0 (Ganesan, 2018) leverages synonym dictionaries, such as WordNet, and considers all synonyms of matched words when computing token overlap. ROUGE-G (ShafieiBavani et al., 2018) combines lexical and semantic matching by applying graph analysis algorithms to the WordNet semantic network. Despite being a step in the direction of a more comprehensive evaluation protocol, none of these metrics gained sufficient traction in the research community, leaving ROUGE as the default automatic evaluation toolkit for text summarization.

### 2.3 Models

Existing summarization models fall into three categories: *abstractive*, *extractive*, and *hybrid*.

*Extractive* models select spans of text from the input and copy them directly into the summary. Non-neural approaches (Neto et al., 2002; Dorr et al., 2003; Filippova and Altun, 2013; Colmenares et al., 2015) utilized domain expertise to develop heuristics for summary content selection, whereas more recent, neural techniques allow for end-to-end training. In the most common case, models are trained as word- or sentence-level classifiers that predict whether a fragment should be included in the summary (Nallapati et al., 2016b, 2017; Narayan et al., 2017; Liu et al., 2019; Xu and Durrett, 2019). Other approaches apply reinforcement learning training strategies to directly optimize the model on task-specific, non-differentiable reward functions (Narayan et al., 2018b; Dong et al., 2018; Wu and Hu, 2018) .

*Abstractive* models paraphrase the source doc-

uments and create summaries with novel phrases not present in the source document. A common approach in *abstractive* summarization is to use attention and copying mechanisms (See et al., 2017; Tan et al., 2017; Cohan et al., 2018). Other approaches include using multi-task and multi-reward training (Paulus et al., 2017; Jiang and Bansal, 2018; Guo et al., 2018; Pasunuru and Bansal, 2018; Kryściński et al., 2018), and unsupervised training strategies (Chu and Liu, 2018; Schumann, 2018).

*Hybrid* models (Hsu et al., 2018; Liu et al., 2018; Gehrmann et al., 2018; Chen and Bansal, 2018) include both *extractive* and *abstractive* modules and allow to separate the summarization process into two phases – content selection and paraphrasing.

For the sake of brevity we do not describe details of different models, we refer interested readers to the original papers.

### 2.4 Analysis and Critique

Most summarization research revolves around new architectures and training strategies that improve the state of the art on benchmark problems. However, it is also important to analyze and question the current methods and research settings.

Zhang et al. (2018) conducted a quantitative study of the level of abstraction in *abstractive* summarization models and showed that word-level, copy-only *extractive* models achieve comparable results to fully *abstractive* models in the measured dimension. Kedzie et al. (2018) offered a thorough analysis of how neural models perform content selection across different data domains, and exposed data biases that dominate the learning signal in the news domain and architectural limitations of current approaches in learning robust sentence-level representations. Liu and Liu (2010) examine the correlation between ROUGE scores and human judgments when evaluating meeting summarization data and show that the correlation strength is low, but can be improved by leveraging unique meeting characteristics, such as available speaker information. Owczarzak et al. (2012) inspect how inconsistencies in human annotator judgments affect the ranking of summaries and correlations with automatic evaluation metrics. The results showed that system-level rankings, considering all summaries, were stable despite inconsistencies in judgments, however, summary-level rankings and automatic metric correlations benefit from improving annotator consistency. Graham (2015) compare the fitness of the BLEU metric (Papineni et al., 2002) and a number of different ROUGE variants for evaluating summarization outputs. The study reveals superior variants of ROUGE that are different from the commonly used recommendations and shows that the BLEU metric achieves strong correlations with human assessments of generated summaries. Schulman et al. (2015) study the problems related to using ROUGE as an evaluation metric with respect to finding optimal solutions and provide proof of NP-hardness of global optimization with respect to ROUGE.

Similar lines of research, where the authors put under scrutiny existing methodologies, datasets, or models were conducted by Callison-Burch et al. (2006, 2007); Tan et al. (2015); Post (2018) in machine translation, (Gkatzia and Mahamood, 2015; Reiter and Belz, 2009; Reiter, 2018) in natural language generation, Lee et al. (2016); Chen et al. (2016); Kaushik and Lipton (2018) in reading comprehension, Gururangan et al. (2018); Poliak et al. (2018); Glockner et al. (2018) in natural language inference, Goyal et al. (2017) in visual question answering, and Xian et al. (2017) in zero-shot image classification. Comments on the general state of scholarship in the field of machine learning were presented by Sculley et al. (2018); Lipton and Steinhardt (2019) and references therein.

## 3 Datasets

### 3.1 Underconstrained task

The task of summarization is to compress long documents by identifying and extracting the most important information from the source documents. However, assessing the importance of information is a difficult task in itself, that highly depends on the expectations and prior knowledge of the target reader.

We show that the current setting in which models are simply given a document with one associated reference summary and no additional information, leaves the task of summarization underconstrained and thus too ambiguous to be solved by end-to-end models.

To quantify this effect, we conducted a human study which measured the agreement between different annotators in selecting important sentences

| **Article** | |
|---|---|
| The glowing blue letters that once lit the Bronx from above Yankee stadium failed to find a buyer at an auction at Sotheby's on Wednesday. While the 13 letters were expected to bring in anywhere from $300,000 to $600,000, the only person who raised a paddle - for $260,000 - was a Sotheby's employee trying to jump start the bidding. The current owner of the signage is Yankee hall-of-famer Reggie Jackson, who purchased the 10-feet-tall letters for an undisclosed amount after the stadium saw its final game in 2008. No love: 13 letters that hung over Yankee stadium were estimated to bring in anywhere from $300,000 to $600,000, but received no bids at a Sotheby's auction Wednesday. The 68-year-old Yankee said he wanted 'a new generation to own and enjoy this icon of the Yankees and of New York City.', The letters had beamed from atop Yankee stadium near grand concourse in the Bronx since 1976, the year before Jackson joined the team. (...) | |
| **Summary Questions** | |
| When was the auction at Sotheby's?<br>Who is the owner of the signage?<br>When had the letters been installed on the stadium? | |
| **Constrained Summary A** | **Unconstrained Summary A** |
| Glowing letters that had been hanging above the Yankee stadium from 1976 to 2008 were placed for auction at Sotheby's on Wednesday, but were not sold, The current owner of the sign is Reggie Jackson, a Yankee hall-of-famer. | There was not a single buyer at the auction at Sotheby's on Wednesday for the glowing blue letters that once lit the Bronx's Yankee Stadium. Not a single non-employee raised their paddle to bid. Jackson, the owner of the letters, was surprised by the lack of results. The venue is also auctioning off other items like Mets memorabilia. |
| **Constrained Summary B** | **Unconstrained Summary B** |
| An auction for the lights from Yankee Stadium failed to produce any bids on Wednesday at Sotheby's. The lights, currently owned by former Yankees player Reggie Jackson, lit the stadium from 1976 until 2008. | The once iconic and attractive pack of 13 letters that was placed at the Yankee stadium in 1976 and later removed in 2008 was unexpectedly not favorably considered at the Sotheby's auction when the 68 year old owner of the letters attempted to transfer its ownership to a member the younger populace. Thus, when the minimum estimate of $300,000 was not met, a further attempt was made by a former player of the Yankees to personally visit the new owner as an |

Table 1: Example summaries collected from human annotators in the *constrained* (left) and *unconstrained* (right) task. In the *unconstrained* setting, annotators were given a news article and asked to write a summary covering the parts they considered most important. In the *constrained* setting, annotators were given a news article with three associated questions and asked to write a summary that contained the answers to the given questions.

from a fragment of text. We asked workers to write summaries of news articles and highlight sentences from the source documents that they based their summaries on. The experiment was conducted in two settings: *unconstrained*, where the annotators were instructed to summarize the content that they considered most important, and *constrained*, where annotators were instructed to write summaries that would contain answers to three questions associated with each article. This is similar to the construction of the TAC 2008 Opinion Summarization Task [1]. The questions associated with each article where collected from human workers through a separate assignment. Experiments were conducted on 100 randomly sampled articles, further details of the human study can be found in Appendix A.1.

Table 2 shows the average number of sentences, per-article, that annotators agreed were important. The rows show how the average changes with the human vote threshold needed to reach consensus about the importance of any sentence. For example, if we require that three or more human votes are necessary to consider a sentence important, annotators agreed on average on the importance of 0.627 and 1.392 sentences per article in the *unconstrained* and *constrained* settings respectively. The average length (in sentences) of sampled articles was 16.59, with a standard deviation of 5.39. The study demonstrates the difficulty and ambiguity of content selection in text summarization.

We also conducted a qualitative study of summaries written by annotators. Examples comparing summaries written in the *constrained* and *unconstrained* setting are shown in Table 1. We noticed that in both cases the annotators correctly identified the main topic and important fragments of the source article. However, *constrained* summaries were more succinct and targeted, without sacrificing the natural flow of sentences. *Un-*

[1] https://tac.nist.gov/2008/summarization/op.summ.08.guidelines.html

| Human vote threshold | Sent. per article considered important | |
|---|---|---|
| | *Unconstrained* | *Constrained* |
| $= 5$ | 0.028 | 0.251 |
| $\geq 4$ | 0.213 | 0.712 |
| $\geq 3$ | 0.627 | 1.392 |
| $\geq 2$ | 1.695 | 2.404 |
| $\geq 1$ | 5.413 | 4.524 |

Table 2: Average number of sentences, per-article, which annotators agreed were important. The human vote threshold investigates how the average agreement changes with the threshold of human votes required to consider any sentence important. Rows $= 5$ and $\geq 1$ correspond to the set intersection and union of selected sentences accordingly.

*constrained* writers tended to write more verbose summaries that did not add information. The study also highlights the *abstractive* nature of human written summaries in that similar content can be described in unique ways.

### 3.2 Layout bias in news data

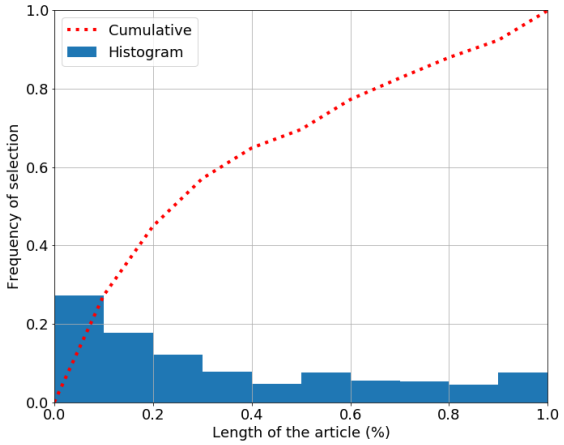


Figure 1: The distribution of important sentences over the length of the article according to human annotators (blue) and its cumulative distribution (red).

News articles adhere to a writing structure known in journalism as the "Inverted Pyramid" (PurdueOWL, 2019). In this form, initial paragraphs contain the most newsworthy information, which is followed by details and background information.

To quantify how strongly articles in the CNN/DM corpus follow this pattern we conducted a human study that measured the importance of different sections of the article. Annotators read news articles and selected sentences they found most important. Experiments were conducted on 100 randomly sampled articles, further details of the human study are described in Appendix A.3. Figure 1 presents how annotator selections were distributed over the length of the article. The distribution is skewed towards the first quarter of the length of articles. The cumulative plot shows that nearly 60% of the important information was present in the first third of the article, and approximately 25% and 15% of selections pointing to the second and last third, respectively.

It has become standard practice to exploit such biases during training to increase performance of models (See et al., 2017; Paulus et al., 2017; Kryściński et al., 2018; Gehrmann et al., 2018; Jiang and Bansal, 2018; Pasunuru and Bansal, 2018), but the importance of these heuristics has been accepted without being quantified. These same heuristics would not apply to books or legal documents, which lack the Inverted Pyramid layout so common in the news domain, so it is important that these heuristics be part of ablation studies rather than accepted as default pre-processing step.

### 3.3 Noise in scraped datasets

Given the data requirements of deep neural networks and the vast amounts of diverse resources available online, automatically scraping web content is a convenient way of collecting data for new corpora. However, adapting scraped content to the needs of end-to-end models is problematic. Given that manual inspection of data is infeasible and human annotators are expensive, data curation is usually limited to removing any markup structure and applying simple heuristics to discard obviously flawed examples. This, in turn, makes the quality of the datasets heavily dependent on how well the scraped content adheres to the assumptions made by the authors about its underlying structure.

This issue suggests that available summarization datasets would be filled with noisy examples. Manual inspection of the data, particularly the reference summaries, revealed easily detectable, consistent patterns of flawed examples Many such examples can be isolated using simple regular expressions and heuristics, which allows approximation of how widespread these flaws are in the dataset.

We investigated this issue in two large summarization corpora scraped from the internet:

| *CNN/DM - Links to other articles* |
|---|
| Michael Carrick has helped Manchester United win their last six games. Carrick should be selected alongside Gary Cahill for England. Carrick has been overlooked too many times by his country. READ : Carrick and Man United team-mates enjoy second Christmas party. |
| *Newsroom - Links to news sources* |
| Get Washington DC, Virginia, Maryland and national news. Get the latest/breaking news, featuring national security, science and courts. Read news headlines from the nation and from The Washington Post. Visit www.washingtonpost.com/nation today. |

Table 3: Examples of noisy reference summaries found in the CNN/DM and Newsroom datasets.

| *Article* |
|---|
| Quick-thinking: Brady Olson, a teacher at North Thurston High, took down a gunman on Monday. A Washington High School teacher is being hailed a hero for tackling a 16-year-old student to the ground after he opened fire on Monday morning (...) |
| *Summary - Factually incorrect* |
| Brady Olson, a Washington High School teacher at North Thurston High, opened fire on Monday morning. No one was injured after the boy shot twice toward the ceiling in the school commons before classes began at North Thurston High School in Lacey (...) |

Table 4: Example of a factually incorrect summary generated by an abstractive model. Top: ground-truth article. Bottom: summary generated by model.

CNN/DM (Nallapati et al., 2016a) and the Newsroom (Grusky et al., 2018). The problem of noisy data affects 0.47%, 5.92%, and 4.19% of the training, validation, and test split of the CNN/DM dataset, and 3.21%, 3.22%, and 3.17% of the respective splits of the Newsroom dataset. Examples of noisy summaries are shown in Table 3. Flawed examples contained links to other articles and news sources, placeholder texts, unparsed HTML code, and non-informative passages in the reference summaries.

## 4 Evaluation Metrics

### 4.1 Weak correlation with human judgment

The effectiveness of ROUGE was previously evaluated (Lin, 2004; Graham, 2015) through statistical correlations with human judgment on the DUC datasets (Over and Yen, 2001, 2002, 2003). However, their setting was substantially different from the current environment in which summarization models are developed and evaluated.

To investigate the robustness of ROUGE in the setting in which it is currently used, we evaluate how its scores correlate with the judgment of an average English-speaker using examples from the CNN/DM dataset. Following the human evaluation protocol from Gehrmann et al. (2018), we asked annotators to rate summaries across four dimensions: *relevance* (selection of important content from the source), *consistency* (factual alignment between the summary and the source), *fluency* (quality of individual sentences), and *coherence* (collective quality of all sentences). Each summary was rated by 5 distinct judges with the final score obtained by averaging the individual scores. Experiments were conducted on 100 randomly sampled articles with the outputs of 13 summarization systems provided by the original authors. Correlations were computed between all pairs of Human-, ROUGE-scores, for all systems. Additional summaries were collected from annotators to inspect the effect of using multiple ground-truth labels on the correlation with automatic metrics. Further details of the human study can be found in Appendix A.2.

Results are shown in Table 5. The left section of the table presents Pearson's correlation coefficients and the right section presents Kendall rank correlation coefficients. In terms of Pearsons's coefficients, the study showed minimal correlation with any of the annotated dimensions for both *abstractive* and *extractive* models together and for *abstractive* models individually. Weak correlation was discovered for *extractive* models primarily with the *fluency* and *coherence* dimensions.

We hypothesized that the noise contained in the fine-grained scores generated by both human annotators and ROUGE might have affected the correlation scores. We evaluated the relation on a higher level of granularity by means of correlation between rankings of models that were obtained from the fine-grained scores. The study showed weak correlation with all measured dimensions, when evaluated for both *abstractive* and *extractive* models together and for *abstractive* models individually. Moderate correlation was found for *extractive* models across all dimensions. A surprising result was that correlations grew weaker with the increase of ground truth references.

Our results align with the observations from Liu and Liu (2010) who also evaluated ROUGE outside of its original setting. The study highlights the limited utility in measuring progress of the field

| | Pearson correlation | | | | | | | | | Kendall rank correlation | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 Reference | | | 5 References | | | 10 References | | | 1 Reference | | | 5 References | | | 10 References | | |
| | **R-1** | **R-2** | **R-L** | **R-1** | **R-2** | **R-L** | **R-1** | **R-2** | **R-L** | **R-1** | **R-2** | **R-L** | **R-1** | **R-2** | **R-L** | **R-1** | **R-2** | **R-L** |
| *All Models* | | | | | | | | | | | | | | | | | | |
| Relevance | 0.07 | 0.03 | 0.06 | 0.03 | 0.02 | 0.02 | 0.05 | 0.03 | 0.04 | 0.29 | 0.30 | 0.29 | 0.28 | 0.29 | 0.27 | 0.28 | 0.28 | 0.27 |
| Consistency | 0.08 | 0.03 | 0.07 | 0.02 | 0.01 | 0.01 | 0.03 | 0.01 | 0.02 | 0.27 | 0.27 | 0.28 | 0.28 | 0.28 | 0.27 | 0.28 | 0.30 | 0.28 |
| Fluency | 0.08 | 0.06 | 0.08 | 0.05 | 0.03 | 0.04 | 0.05 | 0.04 | 0.05 | 0.28 | 0.27 | 0.29 | 0.26 | 0.28 | 0.27 | 0.26 | 0.26 | 0.25 |
| Coherence | 0.06 | 0.05 | 0.07 | 0.05 | 0.04 | 0.05 | 0.04 | 0.03 | 0.04 | 0.29 | 0.31 | 0.29 | 0.27 | 0.27 | 0.27 | 0.28 | 0.28 | 0.27 |
| *Abstractive Models* | | | | | | | | | | | | | | | | | | |
| Relevance | 0.04 | 0.01 | 0.05 | 0.01 | 0.00 | 0.00 | 0.04 | 0.02 | 0.03 | 0.31 | 0.28 | 0.29 | 0.25 | 0.28 | 0.27 | 0.28 | 0.26 | 0.24 |
| Consistency | 0.07 | 0.01 | 0.06 | 0.00 | −0.02 | −0.01 | 0.03 | 0.01 | 0.03 | 0.30 | 0.30 | 0.29 | 0.26 | 0.25 | 0.25 | 0.24 | 0.27 | 0.24 |
| Fluency | 0.06 | 0.04 | 0.07 | 0.03 | 0.01 | 0.02 | 0.05 | 0.04 | 0.04 | 0.28 | 0.28 | 0.28 | 0.24 | 0.25 | 0.24 | 0.27 | 0.26 | 0.24 |
| Coherence | 0.04 | 0.02 | 0.04 | 0.02 | 0.01 | 0.02 | 0.03 | 0.02 | 0.03 | 0.30 | 0.32 | 0.31 | 0.25 | 0.27 | 0.26 | 0.28 | 0.29 | 0.26 |
| *Extractive Models* | | | | | | | | | | | | | | | | | | |
| Relevance | 0.14 | 0.09 | 0.13 | 0.09 | 0.05 | 0.07 | 0.06 | 0.03 | 0.04 | 0.45 | 0.44 | 0.49 | 0.38 | 0.38 | 0.38 | 0.44 | 0.43 | 0.42 |
| Consistency | 0.10 | 0.09 | 0.11 | 0.07 | 0.07 | 0.07 | 0.00 | −0.03 | −0.02 | 0.48 | 0.49 | 0.49 | 0.38 | 0.39 | 0.38 | 0.42 | 0.41 | 0.42 |
| Fluency | 0.13 | 0.14 | 0.13 | 0.10 | 0.10 | 0.08 | 0.06 | 0.03 | 0.04 | 0.47 | 0.47 | 0.48 | 0.35 | 0.37 | 0.35 | 0.39 | 0.34 | 0.37 |
| Coherence | 0.15 | 0.17 | 0.15 | 0.13 | 0.13 | 0.13 | 0.08 | 0.05 | 0.06 | 0.42 | 0.43 | 0.46 | 0.40 | 0.39 | 0.39 | 0.38 | 0.34 | 0.36 |

Table 5: Correlations between human annotators and ROUGE scores along different dimensions and multiple reference set sizes. Left: Pearson's correlation coefficients. Right: Kendall's rank correlation coefficients.

solely by means of ROUGE scores.

### 4.2 Insufficient evaluation protocol

The goal of text summarization is to automatically generate succinct, fluent, relevant, and factually consistent summaries. The current evaluation protocol depends primarily on the exact lexical overlap between reference and candidate summaries measured by ROUGE. In certain cases, ROUGE scores are complemented with human studies where annotators rate the *relevance* and *fluency* of generated summaries. Neither of the methods explicitly examines the factual consistency of summaries, leaving this important dimension unchecked.

To evaluate the factual consistency of existing models, we manually inspected randomly sampled articles with summaries coming from randomly chosen, *abstractive* models. We focused exclusively on factual incorrectness and ignored any other issues, such as low fluency. Out of 200 article-summary pairs that were reviewed manually, we found that 60 (30%) contained consistency issues. Table 4 shows examples of discovered inconsistencies. Some of the discovered inconsistencies, despite being factually incorrect, could be rationalized by humans. However, in many cases, the errors were substantial and could have severe repercussions if presented as-is to target readers.

## 5 Models

### 5.1 Layout bias in news data

We revisit the problem of layout bias in news data from the perspective of models. Kedzie et al. (2018) showed that in the case of news articles, the layout bias dominates the learning signal for neural models. In this section, we approximate the degree with which generated summaries rely on the leading sentences of news articles.

We computed ROUGE scores for collected models in two settings: first using the CNN/DM reference summaries as the ground-truth, and second where the leading three sentences of the source article were used as the ground-truth, i.e. the Lead-3 baseline. We present the results in Table 6.

For all examined models we noticed a substantial increase of overlap across all ROUGE variants. Results suggest that performance of current models is strongly affected by the layout bias of news corpora. Lead-3 is a strong baseline that exploits the described layout bias. However, there is still a large gap between its performance and an upper bound for extractive models (extractive oracle).

### 5.2 Diversity of model outputs

Models analyzed in this paper are considerably different from each other in terms of architectures, training strategies, and underlying approaches. We inspected how the diversity in approaches translates into the diversity of model outputs.

We computed ROUGE-1 and ROUGE-4 scores between pairs of model outputs to compare them by means of token and phrase overlap. Results are visualized in Figure 2, where the values above and below the diagonal are ROUGE-1 and -4 scores accordingly, and model names (M-) follow the order from Table 6.

| | Target Reference | | | | | Lead-3 Reference | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **R-1** | **R-2** | **R-3** | **R-4** | **R-L** | **R-1** | **R-2** | **R-3** | **R-4** | **R-L** |
| Extractive Oracle (Grusky et al., 2018) | 93.36 | 83.19 | - | - | 93.36 | - | - | - | - | - |
| Lead-3 Baseline | 40.24 | 17.53 | 9.94 | 6.50 | 36.49 | - | - | - | - | - |
| *Abstractive Models* | | | | | | | | | | |
| Model Hsu et al. (2018) | 40.68 | 17.97 | 10.43 | 6.97 | 37.13 | 69.66 | 62.60 | 60.33 | 58.72 | 68.42 |
| Model Gehrmann et al. (2018) | 41.53 | 18.77 | 10.68 | 6.98 | 38.39 | 52.25 | 39.03 | 33.40 | 29.61 | 50.21 |
| Model Jiang and Bansal (2018) | 40.05 | 17.66 | 10.34 | 6.99 | 36.73 | 62.32 | 52.93 | 49.95 | 47.98 | 60.72 |
| Model Chen and Bansal (2018) | 40.88 | 17.81 | 9.79 | 6.19 | 38.54 | 55.87 | 41.30 | 34.69 | 29.88 | 53.83 |
| Model See et al. (2017) | 39.53 | 17.29 | 10.05 | 6.75 | 36.39 | 58.15 | 47.60 | 44.11 | 41.82 | 56.34 |
| Model Kryściński et al. (2018) | 40.23 | 17.30 | 9.33 | 5.70 | 37.76 | 57.22 | 42.30 | 35.26 | 29.95 | 55.13 |
| Model Li et al. (2018) | 40.78 | 17.70 | 9.76 | 6.19 | 38.34 | 56.45 | 42.36 | 35.97 | 31.39 | 54.51 |
| Model Pasunuru and Bansal (2018) | 40.44 | 18.03 | 10.56 | 7.12 | 37.02 | 62.81 | 53.57 | 50.25 | 47.99 | 61.27 |
| Model Zhang et al. (2018) | 39.75 | 17.32 | 10.11 | 6.83 | 36.54 | 58.82 | 47.55 | 44.07 | 41.84 | 56.83 |
| Model Guo et al. (2018) | 39.81 | 17.64 | 10.40 | 7.08 | 36.49 | 56.42 | 45.88 | 42.39 | 40.11 | 54.59 |
| *Extractive Models* | | | | | | | | | | |
| Model Dong et al. (2018) | 41.41 | 18.69 | 10.87 | 7.22 | 37.61 | 73.10 | 66.98 | 65.49 | 64.66 | 72.05 |
| Model Wu and Hu (2018) | 41.25 | 18.87 | 11.05 | 7.38 | 37.75 | 78.68 | 74.74 | 73.74 | 73.12 | 78.08 |
| Model Zhou et al. (2018) | 41.59 | 19.00 | 11.13 | 7.45 | 38.08 | 69.32 | 61.00 | 58.51 | 56.98 | 67.85 |

Table 6: ROUGE (R-) scores computed for different models on the test set of the CNN/DM dataset. Left: Scores computed with the original reference summaries. Right: Scores computed with Lead-3 used as the reference.

We notice that the ROUGE-1 scores vary considerably less than ROUGE-4 scores. This suggests that the models share a large part of the vocabulary on the token level, but differ on how they organize the tokens into longer phrases.

Comparing results with the $n$-gram overlap between models and reference summaries (Table 6) shows a substantially higher overlap between any model pair than between the models and reference summaries. This might imply that the training data contains easy to pick up patterns that all models overfit to, or that the information in the training signal is too weak to connect the content of the source articles with the reference summaries.

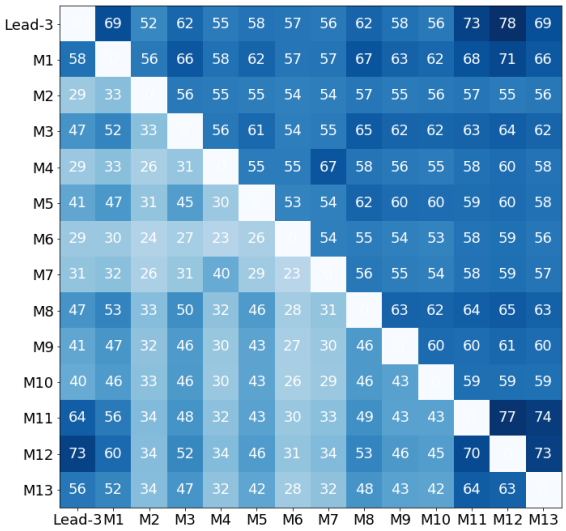


Figure 2: Pairwise similarities between model outputs computed using ROUGE. Above diagonal: Unigram overlap (ROUGE-1). Below diagonal: 4-gram overlap (ROUGE-4). Model order (M-) follows Table 6.

## 6 Conclusions

This critique has highlighted the weak points of the current research setup in text summarization. We showed that text summarization datasets require additional constraints to have well-formed summaries, current state-of-the-art methods learn to rely too heavily on layout bias associated with the particular domain of the text being summarized, and the current evaluation protocol reflects human judgments only weakly while also failing to evaluate critical features (e.g. factual correctness) of text summarization.

We hope that this critique provides the summarization community with practical insights for future research directions that include the construction of datasets, models less fit to a particular domain bias, and evaluation that goes beyond current metrics to capture the most important features of summarization.

## 7 Acknowledgements

We thank all the authors listed in Table 6 for sharing their model outputs and thus contributing to this work. We also thank Shafiq Rayhan Joty for reviewing this manuscript and providing valuable feedback.

## References


Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Chris Callison-Burch, Cameron S. Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. 2007. (meta-) evaluation of machine translation. In *WMT@ACL*, pages 136–158. Association for Computational Linguistics.

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. 2006. Re-evaluation the role of bleu in machine translation research. In *EACL*. The Association for Computer Linguistics.

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. In *ACL (1)*. The Association for Computer Linguistics.

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. In *ACL (1)*, pages 675–686. Association for Computational Linguistics.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*.

Eric Chu and Peter J. Liu. 2018. Unsupervised neural multi-document abstractive summarization. *CoRR*, abs/1810.05739.

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. 2018. A discourse-aware attention model for abstractive summarization of long documents. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers)*.

Carlos A Colmenares, Marina Litvak, Amin Mantrach, and Fabrizio Silvestri. 2015. Heads: Headline generation as sequence prediction using an abstract feature-rich space. In *HLT-NAACL*, pages 133–142.

Yue Dong, Yikang Shen, Eric Crawford, Herke van Hoof, and Jackie Chi Kit Cheung. 2018. Banditsum: Extractive summarization as a contextual bandit. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*.

Bonnie Dorr, David Zajic, and Richard Schwartz. 2003. Hedge trimmer: A parse-and-trim approach to headline generation. In *HLT-NAACL*.

Katja Filippova and Yasemin Altun. 2013. Overcoming the lack of parallel data in sentence compression. In *Proceedings of EMNLP*, pages 1481–1491. Citeseer.

Kavita Ganesan. 2018. ROUGE 2.0: Updated and improved measures for evaluation of summarization tasks. *CoRR*, abs/1803.01937.

Sebastian Gehrmann, Yuntian Deng, and Alexander M. Rush. 2018. Bottom-up abstractive summarization. In *EMNLP*, pages 4098–4109. Association for Computational Linguistics.

Dimitra Gkatzia and Saad Mahamood. 2015. A snapshot of NLG evaluation practices 2005 - 2014. In *ENLG 2015 - Proceedings of the 15th European Workshop on Natural Language Generation, 10-11 September 2015, University of Brighton, Brighton, UK*, pages 57–60.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking NLI systems with sentences that require simple lexical inferences. In *ACL (2)*, pages 650–655. Association for Computational Linguistics.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the V in VQA matter: Elevating the role of image understanding in visual question answering. In *CVPR*, pages 6325–6334. IEEE Computer Society.

David Graff and C Cieri. 2003. English gigaword, linguistic data consortium.

Yvette Graham. 2015. Re-evaluating automatic summarization with BLEU and 192 shades of ROUGE. In *EMNLP*, pages 128–137. The Association for Computational Linguistics.

Max Grusky, Mor Naaman, and Yoav Artzi. 2018. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*.

Han Guo, Ramakanth Pasunuru, and Mohit Bansal. 2018. Soft layer-specific multi-task summarization with entailment and question generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R. Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *NAACL-HLT (2)*, pages 107–112. Association for Computational Linguistics.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *NIPS*.

Kai Hong and Ani Nenkova. 2014. Improving the estimation of word importance for news multi-document summarization. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2014, April 26-30, 2014, Gothenburg, Sweden*.

Wan Ting Hsu, Chieh-Kai Lin, Ming-Ying Lee, Kerui Min, Jing Tang, and Min Sun. 2018. A unified model for extractive and abstractive summarization using inconsistency loss. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*.

Yichen Jiang and Mohit Bansal. 2018. Closed-book training to improve summarization encoder memory. In *EMNLP*, pages 4067–4077. Association for Computational Linguistics.

Divyansh Kaushik and Zachary C. Lipton. 2018. How much reading does reading comprehension require? A critical investigation of popular benchmarks. In *EMNLP*, pages 5010–5015. Association for Computational Linguistics.

Chris Kedzie, Kathleen R. McKeown, and Hal Daumé III. 2018. Content selection in deep learning models of summarization. In *EMNLP*, pages 1818–1828. Association for Computational Linguistics.

Byeongchang Kim, Hyunwoo Kim, and Gunhee Kim. 2018. Abstractive summarization of reddit posts with multi-level memory networks. *CoRR*, abs/1811.00783.

Mahnaz Koupaee and William Yang Wang. 2018. Wikihow: A large scale text summarization dataset. *CoRR*, abs/1810.09305.

Wojciech Kryściński, Romain Paulus, Caiming Xiong, and Richard Socher. 2018. Improving abstraction in text summarization. In *EMNLP*, pages 1808–1817. Association for Computational Linguistics.

Moontae Lee, Xiaodong He, Wen-tau Yih, Jianfeng Gao, Li Deng, and Paul Smolensky. 2016. Reasoning in vector space: An exploratory study of question answering. In *ICLR*.

Junyi Jessy Li, Kapil Thadani, and Amanda Stent. 2016. The role of discourse units in near-extractive summarization. In *Proceedings of the SIGDIAL 2016 Conference, The 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 13-15 September 2016, Los Angeles, CA, USA*.

Wei Li, Xinyan Xiao, Yajuan Lyu, and Yuanzhuo Wang. 2018. Improving neural abstractive document summarization with structural regularization. In *EMNLP*, pages 4078–4087. Association for Computational Linguistics.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Proc. ACL workshop on Text Summarization Branches Out*, page 10.

Zachary C. Lipton and Jacob Steinhardt. 2019. Troubling trends in machine learning scholarship. *ACM Queue*, 17(1):80.

Feifan Liu and Yang Liu. 2010. Exploring correlation between ROUGE and human evaluation on meeting summaries. *IEEE Trans. Audio, Speech & Language Processing*, 18(1):187–196.

Jingyun Liu, Jackie Chi Kit Cheung, and Annie Louis. 2019. What comes next? extractive summarization by next-sentence prediction. *CoRR*, abs/1901.03859.

Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating wikipedia by summarizing long sequences. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *AAAI*.

Ramesh Nallapati, Bowen Zhou, Çağlar Gülçehre, Bing Xiang, et al. 2016a. Abstractive text summarization using sequence-to-sequence rnns and beyond. *Proceedings of SIGNLL Conference on Computational Natural Language Learning*.

Ramesh Nallapati, Bowen Zhou, and Mingbo Ma. 2016b. Classify or select: Neural architectures for extractive document summarization. *CoRR*, abs/1611.04244.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018a. Don't give me the details, just the summary! Topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018b. Ranking sentences for extractive summarization with reinforcement learning. In *NAACL-HLT*, pages 1747–1759. Association for Computational Linguistics.

Shashi Narayan, Nikos Papasarantopoulos, Mirella Lapata, and Shay B. Cohen. 2017. Neural extractive summarization with side information. *CoRR*, abs/1704.04530.

Ani Nenkova and Rebecca J. Passonneau. 2004. Evaluating content selection in summarization: The pyramid method. In *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, HLT-NAACL 2004, Boston, Massachusetts, USA, May 2-7, 2004*.

Joel Larocca Neto, Alex A Freitas, and Celso AA Kaestner. 2002. Automatic text summarization using a machine learning approach. In *Brazilian Symposium on Artificial Intelligence*, pages 205–215. Springer.

Jun-Ping Ng and Viktoria Abrecht. 2015. Better summarization evaluation with word embeddings for ROUGE. *CoRR*, abs/1508.06034.

Benjamin Nye and Ani Nenkova. 2015. Identification and characterization of newsworthy verbs in world news. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*.

Paul Over and James Yen. 2001. An introduction to duc-2001: Intrinsic evaluation of generic news text summarization systems.

Paul Over and James Yen. 2002. An introduction to duc-2002: Intrinsic evaluation of generic news text summarization systems.

Paul Over and James Yen. 2003. An introduction to duc-2003: Intrinsic evaluation of generic news text summarization systems.

Karolina Owczarzak, Peter A. Rankel, Hoa Trang Dang, and John M. Conroy. 2012. Assessing the effect of inconsistent assessors on summarization evaluation. In *ACL (2)*, pages 359–362. The Association for Computer Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318. ACL.

Rebecca J. Passonneau, Emily Chen, Weiwei Guo, and Dolores Perin. 2013. Automated pyramid scoring of summaries using distributional semantics. In *ACL (2)*, pages 143–147. The Association for Computer Linguistics.

Ramakanth Pasunuru and Mohit Bansal. 2018. Multi-reward reinforced summarization with saliency and entailment. *CoRR*, abs/1804.06451.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. In *ICLR*.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis only baselines in natural language inference. In **SEM@NAACL-HLT*, pages 180–191. Association for Computational Linguistics.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *WMT*, pages 186–191. Association for Computational Linguistics.

PurdueOWL. 2019. Journalism and journalistic writing: The inverted pyramid structure. Accessed: 2019-05-15.

Ehud Reiter. 2018. A structured review of the validity of BLEU. *Computational Linguistics*, 44(3).

Ehud Reiter and Anja Belz. 2009. An investigation into the validity of some metrics for automatically evaluating natural language generation systems. *Computational Linguistics*, 35(4):529–558.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *Proceedings of EMNLP*.

Evan Sandhaus. 2008. The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752.

John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. 2015. Gradient estimation using stochastic computation graphs. In *NIPS*.

Raphael Schumann. 2018. Unsupervised abstractive sentence summarization using length controlled variational autoencoder. *CoRR*, abs/1809.05233.

D. Sculley, Jasper Snoek, Alexander B. Wiltschko, and Ali Rahimi. 2018. Winner's curse? on pace, progress, and empirical rigor. In *ICLR (Workshop)*. OpenReview.net.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *ACL*.

Elaheh ShafieiBavani, Mohammad Ebrahimi, Raymond K. Wong, and Fang Chen. 2018. A graph-theoretic summary evaluation for rouge. In *EMNLP*, pages 762–767. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Sho Takase, Jun Suzuki, Naoaki Okazaki, Tsutomu Hirao, and Masaaki Nagata. 2016. Neural headline generation on abstract meaning representation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*.

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*.

Liling Tan, Jon Dehdari, and Josef van Genabith. 2015. An awkward disparity between BLEU / RIBES scores and human judgements in machine translation. In *WAT*, pages 74–81. Workshop on Asian Translation.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6000–6010.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *NIPS*.

Yuxiang Wu and Baotian Hu. 2018. Learning to extract coherent summary via deep reinforcement learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*.

Yongqin Xian, Bernt Schiele, and Zeynep Akata. 2017. Zero-shot learning - the good, the bad and the ugly. In *CVPR*, pages 3077–3086. IEEE Computer Society.

Jiacheng Xu and Greg Durrett. 2019. Neural extractive text summarization with syntactic compression. *CoRR*, abs/1902.00863.

Yinfei Yang and Ani Nenkova. 2014. Detecting information-dense texts in multiple news domains. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*

Fangfang Zhang, Jin-ge Yao, and Rui Yan. 2018. On the abstractiveness of neural document summarization. In *EMNLP*, pages 785–790. Association for Computational Linguistics.

Liang Zhou, Chin-Yew Lin, Dragos Stefan Munteanu, and Eduard H. Hovy. 2006. Paraeval: Using paraphrases to evaluate summaries automatically. In *Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 4-9, 2006, New York, New York, USA*.

Qingyu Zhou, Nan Yang, Furu Wei, Shaohan Huang, Ming Zhou, and Tiejun Zhao. 2018. Neural document summarization by jointly learning to score and select sentences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*.

# Neural data-to-text generation: A comparison between pipeline and end-to-end architectures

**Thiago Castro Ferreira**[1,2] **Chris van der Lee**[1] **Emiel van Miltenburg**[1] **Emiel Krahmer**[1]
[1]Tilburg center for Cognition and Communication (TiCC), Tilburg University, The Netherlands
[2]Federal University of Minas Gerais (UFMG), Brazil
{tcastrof,c.vdrlee}@tilburguniversity.edu
{c.w.j.vanmiltenburg,e.j.krahmer}@tilburguniversity.edu

## Abstract

Traditionally, most data-to-text applications have been designed using a modular pipeline architecture, in which non-linguistic input data is converted into natural language through several intermediate transformations. By contrast, recent neural models for data-to-text generation have been proposed as end-to-end approaches, where the non-linguistic input is rendered in natural language with much less explicit intermediate representations in between. This study introduces a systematic comparison between neural pipeline and end-to-end data-to-text approaches for the generation of text from RDF triples. Both architectures were implemented making use of the encoder-decoder Gated-Recurrent Units (GRU) and Transformer, two state-of-the art deep learning methods. Automatic and human evaluations together with a qualitative analysis suggest that having explicit intermediate steps in the generation process results in better texts than the ones generated by end-to-end approaches. Moreover, the pipeline models generalize better to unseen inputs. Data and code are publicly available.[1]

## 1 Introduction

Data-to-text Natural Language Generation (NLG) is the computational process of generating meaningful and coherent natural language text to describe non-linguistic input data (Gatt and Krahmer, 2018). Practical applications can be found in domains such as weather forecasts (Mei et al., 2016), health care (Portet et al., 2009), feedback for car drivers (Braun et al., 2018), diet management (Anselma and Mazzei, 2018), election results (Leppänen et al., 2017) and sportscasting news (van der Lee et al., 2017).

Traditionally, most of data-to-text applications have been designed in a modular fashion, in which the non-linguistic input data (be it, say, numerical weather information or game statistics) are converted into natural language (e.g., weather forecast, game report) through several explicit intermediate transformations. The most prominent example is the 'traditional' pipeline architecture (Reiter and Dale, 2000) that performs tasks related to document planning, sentence planning and linguistic realization in sequence. Many of the traditional, rule-based NLG systems relied on modules because (a) these modules could be more easily reused across applications, and (b) because going directly from input to output using rules was simply too complex in general (see Gatt and Krahmer 2018 for a discussion of different architectures).

The emergence of neural methods changed this: provided there is enough training data, it does become possible to learn a direct mapping from input to output, as has also been shown in, for example, neural machine translation. As a result, in NLG more recently, neural end-to-end data-to-text models have been proposed, which directly learn input-output mappings and rely much less on explicit intermediate representations (Wen et al., 2015; Dušek and Jurcicek, 2016; Mei et al., 2016; Lebret et al., 2016; Gehrmann et al., 2018).

However, the fact that neural end-to-end approaches are *possible* does not necessarily entail that they are *better* than (neural) pipeline models. On the one hand, cascading of errors is a known problem of pipeline models in general (an error in an early module will impact all later modules in the pipeline), which (almost by definition) does not apply to end-to-end models. On the other hand, it is also conceivable that developing dedicated neural modules for specific tasks leads to better performance on each of these successive tasks, and combining them might lead to better,

[1] https://github.com/ThiagoCF05/DeepNLG/



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 552–562,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

and more reusable, output results. In fact, this has never been systematically studied, and this is the main goal of the current paper.

We present a systematic comparison between neural pipeline and end-to-end data-to-text approaches for the generation of output text from RDF input triples, relying on an augmented version of the WebNLG corpus (Gardent et al., 2017b). Using two state-of-the-art deep learning techniques, GRU (Cho et al., 2014) and Transformer (Vaswani et al., 2017), we develop both a neural pipeline and an end-to-end architecture. The former, which also makes use of the NeuralREG approach (Castro Ferreira et al., 2018a), tackles standard NLG tasks (discourse ordering, text structuring, lexicalization, referring expression generation and textual realization) in sequence, while the latter does not address these individual tasks, but directly tries to learn how to map RDF triples into corresponding output text.

Using a range of evaluation techniques, including both automatic and human measures, combined with a qualitative analysis, we provide answers to our two main research questions: (RQ1) How well do deep learning methods perform as individual modules in a data-to-text pipeline architecture? And (RQ2) How well does a neural pipeline architecture perform compared to a neural end-to-end one? Our results show that adding supervision during the data-to-text generation process, by distinguishing separate modules and combining them in a pipeline, leads to better results than full end-to-end approaches. Moreover, the pipeline architecture offers somewhat better generalization to unseen domains and compares favorably to the current state-of-the-art.

## 2 Data

The experiments presented in this work were conducted on the WebNLG corpus (Gardent et al., 2017a,b), which consists of sets of ⟨ Subject, Predicate, Object ⟩ RDF triples and their target texts. In comparison with other popular NLG benchmarks (Belz et al., 2011; Novikova et al., 2017; Mille et al., 2018), WebNLG is the most semantically varied corpus, consisting of 25,298 texts describing 9,674 sets of up to 7 RDF triples in 15 domains. Out of these domains, 5 are exclusively present in the test set, being unseen during the training and validation processes. Figure 1 depicts an example of a set of 3 RDF triples and its related text.

| | | |
|---|---|---|
| A.C._Cesena | **manager** | Massimo_Drago |
| Massimo_Drago | **club** | S.S.D._Potenza_Calcio |
| Massimo_Drago | **club** | Calcio_Catania |

↓

Massimo Drago played for the club SSD Potenza Calcio and his own club was Calcio Catania. He is currently managing AC Cesena.

Figure 1: Example of a set of triples (top) and the corresponding text (bottom).

To evaluate the intermediate stages between the triples and the target text, we use the augmented version of the WebNLG corpus (Castro Ferreira et al., 2018b), which provides gold-standard representations for traditional pipeline steps, such as discourse ordering (i.e., the order in which the source triples are verbalized in the target text), text structuring (i.e., the organization of the triples into paragraph and sentences), lexicalization (i.e., verbalization of the predicates) and referring expression generation (i.e., verbalization of the entities).

## 3 Pipeline Architecture

Based on Reiter and Dale (2000), we propose a pipeline architecture which converts a set of RDF triples into text in 5 sequential steps.

### 3.1 Discourse Ordering

Originally designed to be performed when the document is planned, Discourse Ordering is the process of determining the order in which the communicative goals should be verbalized in the target text. In our case, the communicative goals are the RDF triples received as input by the model.

Given a set of linearized triples, this step determines the order in which they should be verbalized. For example, given the triple set in Figure 1 in the linearized format:

<TRIPLE> A.C._Cesena **manager** Massimo_Drago </TRIPLE> <TRIPLE> Massimo_Drago **club** S.S.D._Potenza_Calcio </TRIPLE> <TRIPLE> Massimo_Drago **club** Calcio_Catania </TRIPLE>

Our discourse ordering model would ideally return the set `club club manager`, which later is used to retrieve the input triples on the predicted order. In case of triples with the same predicates, as `club`, our implementation will randomly retrieve the triples.

### 3.2 Text Structuring

Text Structuring is the step which intends to organize the ordered triples into paragraphs and sentences. Since the WebNLG corpus only contains single-paragraph texts, this step will be only evaluated on sentence planning, being closer to the Aggregation task of the original architecture (Reiter and Dale, 2000). However, it can be easily extended to predict paragraph structuring in multi-paragraph datasets.

Given a linearized set of ordered triples, this step works by generating the predicates segmented by sentences based on the tokens <SNT> and </SNT>. For example, given the ordered triple set in Figure 1 in the same linearized format as in Discourse Ordering, the module would generate <SNT> club club </SNT> <SNT> manager </SNT>, where predicates are replaced by the proper triples for the next step.

### 3.3 Lexicalization

Lexicalization involves finding the proper phrases and words to express the content to be included in each sentence (Reiter and Dale, 2000). In this study, given a linearized ordered set of triples segmented by sentences, the Lexicalization step aims to predict a template which verbalizes the predicates of the triples. For our example based on Figure 1, given the ordered triple set segmented by sentences in the following format:

---

<SNT> <TRIPLE> Massimo_Drago **club** S.S.D._Potenza_Calcio </TRIPLE> <TRIPLE> Massimo_Drago **club** Calcio_Catania </TRIPLE> </SNT> <SNT> <TRIPLE> A.C._Cesena **manager** Massimo_Drago </TRIPLE> </SNT>

---

This step would ideally return a template like:

---

**ENTITY-1** VP[*aspect=simple, tense=past, voice=active, person=null, number=null*] play for DT[*form=defined*] the club **ENTITY-2** and **ENTITY-1** own club VP[*aspect=simple, tense=past, voice=active, person=null, number=singular*] be **ENTITY-3** . **ENTITY-1** VP[*aspect=simple, tense=present, voice=active, person=3rd, number=singular*] be currently VP[*aspect=progressive, tense=present, voice=active, person=null, number=null*] manage **ENTITY-4** .

---

The template format not only selects the proper phrases and words to verbalize the predicates, but also does indications for the further steps. The general tags ENTITY-[0-9] indicates where references should be realized. The number in an entity tag indicates the entity to be realized based on its occurrence in the ordered triple set. For instance, ENTITY-3 refers to the entity Calcio_Catania, the third mentioned entity in the ordered triple set.

Information for the further textual realization step is stored in the tags VP, which contains the aspect, mood, tense, voice and number of the subsequent lemmatized verb, and DT, which depicts the form of the subsequent lemmatized determiner.[2]

### 3.4 Referring Expression Generation

Referring Expression Generation (REG) is the pipeline task responsible for generating the references to the entities of the discourse (Krahmer and van Deemter, 2012). As previously explained, the template created in the previous step depicts where and to which entities such references should be generated. Given our example based in Figure 1, the result of the REG step for the template predicted in the previous step would be:

---

**Massimo Drago** VP[...] play for DT[...] the club **SSD Potenza Calcio** and **his** own club VP[...] be **Calcio Catania** . **He** VP[...] be currently VP[...] manage **AC Cesena** .

---

To perform the task, we used the NeuralREG algorithm (Castro Ferreira et al., 2018a). Given a reference to be realized, this algorithm works by encoding the template before (pre-context) and after (post-contex) the reference using two different Bidirectional LSTMs (Hochreiter and Schmidhuber, 1997). Attention vectors are then computed for both vectors and concatenated together with the embedding of the entity. Finally, this representation is decoded into the referring expression to the proper entity in the given context.[3]

### 3.5 Textual Realization

Textual Realization aims to perform the last steps of converting the non-linguistic data into text. In our pipeline architecture this includes setting the verbs (e.g., *VP[aspect=simple, tense=past, voice=passive, person=3rd, number=singular] locate → was located*) and determiners (*DT[form=undefined] a American national → an American national*) to their right formats. Both verbs and determiners are solved in a

[2]Both kind of tags with their respective information are treated as a single token.

[3]NeuralREG works with the Wikipedia representation of the entities (e.g., Massimo_Drago) in the templates instead of general tags (e.g., ENTITY-1).

rule-based strategy, where the implications are extracted from the training set. This step will not be individually evaluated as the other ones.

## 4 End-to-End Architecture

Our end-to-end architecture is similar to novel data-to-text models (Wen et al., 2015; Dušek and Jurcicek, 2016; Mei et al., 2016; Lebret et al., 2016; Gehrmann et al., 2018), which aim to convert a non-linguistic input into natural language without explicit intermediate representations, making use of Neural Machine Translation techniques. In this study, our end-to-end architecture intends to directly convert an unordered (linearized) set of RDF triples into text.

## 5 Models Set-Up

Both pipeline steps and the end-to-end architecture were modelled using two deep learning encoder-decoder approaches: Gated-Recurrent Units (GRU; Cho et al. 2014) and Transformer (Vaswani et al., 2017). These models differ in the way they encode their input. GRUs encode the input data by going over the tokens one-by-one, while Transformers (which do not have a recurrent structure) may encode the entire source sequence as a whole, using position embeddings to keep track of the order. We are particularly interested in the capacity of such approaches to learn order and structure in the process of text generation. The model settings are explained in the supplementary materials.

## 6 Experiment 1: Learning the pipeline

Most of the data-to-text pipeline applications have their steps implemented using rule-based or statistical data-driven models. However, these techniques have shown to be outperformed by deep neural networks in other Computational Linguistics subfields and in particular pipeline steps like Referring Expression Generation. NeuralREG (Castro Ferreira et al., 2018a), for instance, outperforms other techniques in generating references and co-references along a single-paragraph text. Given this context, our first experiment intends to analyze how well deep learning methods perform particular steps of the pipeline architecture, like Discourse Ordering, Text Structuring, Lexicalization and Referring Expression Generation, in comparison with simpler data-driven baselines.

### 6.1 Data

We used version 1.5 of the augmented WebNLG corpus (Castro Ferreira et al., 2018b)[4] to evaluate the steps of our pipeline approach. Based on its intermediate representations, we extracted gold-standards to train and evaluate the different steps.

**Discourse Ordering** We used pairs of RDF triple sets and their ordered versions to evaluate our Discourse Ordering approaches. For the cases in the training set where a triple set was verbalized in more than one order, we added one entry per verbalization taking the proper order as the target. To make sure the source set followed a pattern, we ordered the input according to the alphabetic order of its predicates, followed by the alphabetical order of its subjects and objects in case of similar predicates. In total, our Discourse Ordering data consists of 13,757, 1,730 and 3,839 ordered triple sets for 5,152, 644 and 1,408 training, development and test input triple sets, respectively.

**Text Structuring** 14,010, 1,752 and 3,955 structured triple sets were extracted for 10,281, 1,278 and 2,774 training, development and test ordered triple sets, respectively.

**Lexicalization** 18,295, 2,288 and 5,012 lexicalization templates were used for 12,814, 1,601 and 3,463 training, development and test structured triple sets, respectively.

**Referring Expression Generation** To evaluate the performance of the REG models, we extracted 67,144, 8,294 and 19,210 reference instances from training, development and test part of the corpus. Each instance consists of the cased tokenized referring expression, the identifier of the target entity and the uncased tokenized pre- and post-contexts.

### 6.2 Metrics

Discourse Ordering and Text Structuring approaches were evaluated based on their accuracy to predict one of the gold-standards given the input (many of the RDF triple sets in the corpus were verbalized in more than one order and structure). Referring Expression Generation approaches were also evaluated based on their accuracy to predict the uncased tokenized gold-standard referring expressions. Lexicalization was evaluated based on the BLEU score of the predicted templates in their uncased tokenized form.

[4] https://github.com/ThiagoCF05/webnlg

### 6.3 Baselines

We proposed *random* and *majority* baselines for the steps of Discourse Ordering, Text Structuring and Lexicalization. In comparison with NeuralREG, we used the *OnlyNames* baseline, also introduced in Castro Ferreira et al. (2018a).

**Discourse Ordering** The random baseline returns the triple set in a random order, whereas the majority one returns the most frequent order of the input predicates in the training set. For unseen sets of predicates, the *majority* model returns the triple set in the same order as the input.

**Text Structuring** The random baseline for this step chooses a random split of triples in sentences, inserting the tags <SNT> and </SNT> in aleatory positions among them. The majority baseline returns the most frequent sentence intervals in the training set based on the input predicates. In case of an unseen set, the model looks for sentence intervals in subsets of the input.

**Lexicalization** Algorithm 1 depicts our baseline approach for Lexicalization. As in Text Structuring, given a set of input triples structured in sentences, the random and majority models return a random and the most frequent template that describes the input predicates, respectively (line 6). If the set of predicates is unseen, the model returns a template that describes a subset of the input.

**Algorithm 1** Lexicalization Pseudocode

```
Require: struct, model
 1: start, end ← 0, |struct|
 2: template ← ∅
 3: while start < |struct| do
 4:     snts ← struct[start,end)
 5:     if snts ∈ model then
 6:         template ← template ∪ model[snts]
 7:         start ← end
 8:         end ← |struct|
 9:     else
10:         end ← end −1
11:         if start = end then
12:             start ← start + 1
13:             end ← |struct|
14:         end if
15:     end if
16: end while
17: return template
```

**Referring Expression Generation** We used *OnlyNames*, a baseline introduced in Castro Ferreira et al. (2018a), in contrast to NeuralREG. Given an entity to be referred to, this model returns the entity Wikipedia identifier with underscores replaced by spaces (`Massimo_Drago` → `Massimo Drago`).

| | All | Seen | Unseen |
|---|---|---|---|
| **Discourse Ordering** | | | |
| Random | 0.31 | 0.29 | 0.35 |
| Majority | 0.48 | 0.51 | 0.44 |
| GRU | 0.35 | 0.56 | 0.10 |
| Transformer | 0.34 | 0.56 | 0.09 |
| **Text Structuring** | | | |
| Random | 0.29 | 0.29 | 0.30 |
| Majority | 0.27 | 0.45 | 0.06 |
| GRU | 0.39 | 0.63 | 0.13 |
| Transformer | 0.36 | 0.59 | 0.12 |
| **Lexicalization** | | | |
| Random | 39.49 | 40.46 | 33.79 |
| Majority | 44.82 | 45.65 | 39.43 |
| GRU | 37.43 | 49.26 | 23.63 |
| Transformer | 38.12 | 48.14 | 24.15 |
| **Referring Expression Generation** | | | |
| *OnlyNames* | 0.51 | 0.53 | 0.50 |
| NeuralREG | 0.39 | 0.70 | 0.07 |

Table 1: Accuracy of Discourse Ordering, Text Structuring and Referring Expression models, as well as BLEU score of Lexicalization approaches.

### 6.4 Results

Table 1 shows the results for our models for each of the 4 evaluated pipeline steps. In general, the results show a clear pattern in all of these steps: both neural models (GRU and Transformer) introduced higher results on domains seen during training, but their performance drops substantially on unseen domains in comparison with the baselines (Random and Majority). The only exception is found in Text Structuring, where the neural models outperforms the Majority baseline on unseen domains, but are still worse than the Random baseline. Between both neural models, recurrent networks seem to have an advantage over the Transformer in Discourse Ordering and Text Structuring, whereas the latter approach performs better than the former one in Lexicalization.

## 7 Experiment 2: Pipeline vs. End-to-End

In this experiment, we contrast our pipeline with our end-to-end implementation and state-of-the-art models for RDF-to-text. The models were evaluated in automatic and human evaluations, followed by a qualitative analysis.

### 7.1 Approaches

**Pipeline** We evaluated 4 implementations of our pipeline architecture, where the output of the previous step is fed into the next one. We call these implementations *Random*, *Majority*, *GRU* and *Transformer*, where each one has its steps solved by one of the proposed baselines or deep learning implementations. In Random and Majority, the referring expressions were generated by the *OnlyNames* baseline, whereas for GRU and Transformer, NeuralREG was used for the seen entities, *OnlyNames* for the unseen ones and special rules to realize dates and numbers.

**End-to-End** We aimed to convert a set of RDF-triples into text using a GRU and a Transformer implementation without explicit intermediate representations in-between.

### 7.2 Models for Comparison

To ground this study with related work, we compared the performance of the proposed approaches with 4 state-of-the-art RDF-to-text models.

**Melbourne** is the approach which obtained the highest performance in the automatic evaluation of the WebNLG Challenge. The approach consists of a neural encoder-decoder approach, which encodes a linearized triple set, with predicates split on camel case (e.g. `floorArea` → `floor area`) and entities represented by general (e.g., `ENTITY-1`) and named entity recognition (e.g., `PERSON`) tags, into a template where references are also represented with general tags. The referring expressions are later generated in the template simply by replacing these general tags with an approach similar to *OnlyNames*.

**UPF-FORGe** obtained the highest ratings in the human evaluation of the WebNLG challenge, having a performance similar to texts produced by humans. It also follows a pipeline architecture, which maps predicate-argument structures onto sentences by applying a series of rule-based graph-transducers (Mille et al., 2019).

**Marcheggiani and Perez (2018)** proposed a graph convolutional network that directly encodes the input triple set in contrast with previous model that first linearize the input to then decode it into text.

**Moryossef et al. (2019)** proposed an approach which converts an RDF triple set into text in two steps: text planning, a non-neural method where the input will be ordered and structured, followed by a neural realization step, where the ordered and structured input is converted into text.

### 7.3 Evaluation

**Automatic Evaluation** We evaluated the textual outputs of each system using the BLEU (Papineni et al., 2002) and METEOR (Lavie and Agarwal, 2007) metrics. The evaluation was done on the entire test data, as well as only in their seen and unseen domains.

**Human Evaluation** We conducted a human evaluation, selecting the same 223 samples used in the evaluation of the WebNLG challenge (Gardent et al., 2017b). For each sample, we used the original texts and the ones generated by our 6 approaches and by the Melbourne and UPF-FORGe reference systems, totaling 2,007 trials. Each trial displayed the triple set and the respective text. The goal of the participants was to rate the trials based on the fluency (i.e., does the text flow in a natural, easy to read manner?) and semantics (i.e., does the text clearly express the data?) of the text in a 1-7 Likert scale.

We recruited 35 raters from Mechanical Turk to participate in the experiment. We first familiarized them with the set-up of the experiment, depicting a trial example in the introduction page accompanied by an explanation. Then each participant had to rate 60 trials, randomly chosen by the system, making sure that each trial was rated at least once.[5]

**Qualitative Analysis** To have a better understanding of the positive and negative aspects of each model, we also performed a qualitative analysis, where the second and third authors of this study analyzed the original texts and the ones generated by the previous 8 models for 75 trials extracted from the human evaluation sample for each combination between size and domain of the corpus. The trials were displayed in a similar way to the human evaluation, where the annotators did not know which model produced the text. The only difference was the additional display of the predicted structure by the pipeline approaches (a

[5]The raters had an average age of 32.29 and 40% were female. 17 participants indicated they were fluent in English, while 18 were native. The experiment, which obtained ethical clearance from the university, took around 20-30 minutes to be completed and each rater received a pay in U.S. dollars for participation.

| | All | Seen | Unseen | All | Seen | Unseen |
|---|---|---|---|---|---|---|
| | | BLEU | | | METEOR | |
| Random | 41.68 | 41.72 | **41.51** | 0.20 | 0.27 | - |
| Majority | 43.82 | 44.79 | 41.13 | 0.33 | 0.41 | 0.22 |
| GRU | 50.55 | 55.75 | 38.55 | 0.33 | **0.42** | 0.22 |
| Transformer | **51.68** | 56.35 | 38.92 | 0.32 | 0.41 | 0.21 |
| E2E GRU | 33.49 | **57.20** | 6.25 | 0.25 | 0.41 | 0.09 |
| E2E Transformer | 31.88 | 50.79 | 5.88 | 0.25 | 0.39 | 0.09 |
| Melbourne | 45.13 | 54.52 | 33.27 | 0.37 | 0.41 | 0.33 |
| UPF-FORGe | 38.65 | 40.88 | 35.70 | **0.39** | 0.40 | **0.37** |
| (Marcheggiani and Perez, 2018) | - | 55.90 | - | - | 0.39 | - |
| (Moryossef et al., 2019) | 47.40 | - | - | **0.39** | - | - |
| | | Fluency | | | Semantic | |
| Random | $4.55^{E}$ | $4.79^{D}$ | $4.07^{D}$ | $4.44^{D}$ | $4.73^{D}$ | $3.86^{C}$ |
| Majority | $5.00^{CD}$ | $5.25^{CD}$ | $4.49^{CD}$ | $5.02^{BC}$ | $5.41^{BC}$ | $4.25^{BC}$ |
| GRU | $5.31^{B}$ | $5.51^{AB}$ | $4.91^{BC}$ | $5.21^{BC}$ | $5.48^{AB}$ | $4.67^{B}$ |
| Transformer | $5.03^{BC}$ | $5.53^{AB}$ | $4.05^{D}$ | $4.87^{C}$ | $5.49^{AB}$ | $3.64^{C}$ |
| E2E GRU | $4.73^{DE}$ | $5.40^{BC}$ | $3.45^{E}$ | $4.47^{D}$ | $5.21^{CD}$ | $3.03^{D}$ |
| E2E Transformer | $5.02^{BC}$ | $5.38^{BC}$ | $4.32^{CD}$ | $4.70^{CD}$ | $5.15^{BCD}$ | $3.81^{C}$ |
| Melbourne | $5.04^{CD}$ | $5.23^{BC}$ | $4.65^{CD}$ | $4.94^{C}$ | $5.33^{BC}$ | $4.15^{C}$ |
| UPF-FORGe | $5.46^{B}$ | $5.43^{BC}$ | $5.51^{AB}$ | $5.31^{B}$ | $5.35^{BC}$ | $5.24^{A}$ |
| Original | $5.76^{A}$ | $5.82^{A}$ | $5.63^{A}$ | $5.74^{A}$ | $5.80^{A}$ | $5.63^{A}$ |

Table 2: (1) BLEU and METEOR scores of the models in the automatic evaluation, and (2) Fluency and Semantic obtained in the human evaluation. In the first part, best results are boldfaced and second best ones are underlined. In the second part, ranking was determined by pair-wise Mann-Whitney statistical tests with $p < 0.05$.

| | Ord. | Struct. | Txt. | Ovr. | Keep. |
|---|---|---|---|---|---|
| | Semantic | | | | |
| Random | 1.00 | 1.00 | 0.43 | 0.05 | 0.41 |
| Majority | 1.00 | 1.00 | 0.75 | 0.01 | 0.69 |
| GRU | 0.77 | 0.73 | 0.67 | 0.01 | 0.81 |
| Transformer | 0.75 | 0.69 | 0.68 | 0.08 | 0.80 |
| E2E GRU | - | - | 0.47 | 0.41 | - |
| E2E Trans. | - | - | 0.39 | 0.53 | - |
| Melbourne | - | - | 0.73 | 0.19 | - |
| UPF-FORGe | - | - | 0.91 | 0.00 | - |
| Original | - | - | 0.99 | 0.12 | - |

| | Verb | Det. | Reference |
|---|---|---|---|
| | Grammaticality | | |
| Random | 0.95 | 0.91 | 0.89 |
| Majority | 1.00 | 1.00 | 0.99 |
| GRU | 1.00 | 0.99 | 0.80 |
| Transformer | 0.95 | 1.00 | 0.93 |
| E2E GRU | 0.97 | 1.00 | 0.91 |
| E2E Trans. | 0.95 | 0.97 | 0.79 |
| Melbourne | 0.96 | 0.87 | 0.77 |
| UPF-FORGe | 1.00 | 1.00 | 1.00 |
| Original | 0.95 | 0.95 | 0.92 |

Table 3: Qualitative analysis. The first part shows the percentage of trials that keeps the input predicates over Discourse Ordering (Ord.), Text Structuring (Struct.) and in the final text (Txt.). It also shows the ratio of text trials with more predicates than in the input (Ovr.) and the pipeline texts which keep the decisions of previous steps (Keep.). The second part shows the number of trials without verb, determiner and reference mistakes.

fake structure was displayed for the other models). Both annotators analyzed grammaticality aspects, like whether the texts had mistakes involving the determiners, verbs and references, and semantic ones, like whether the text followed the predicted order and structure, and whether it verbalizes less or more information than the input triples.[6]

### 7.4 Results

Table 2 depicts the results of automatic and human evaluations, whereas Table 3 shows the results of the qualitative analysis.

**Automatic Evaluation** In terms of BLEU, our neural pipeline models (GRU and Transformer) outperformed all the reference approaches in all domains, whereas our end-to-end GRU and Random pipeline obtained the best results on seen and unseen domains, respectively.

Regarding METEOR, which includes synonymy matching to score the inputs, reference methods introduced the best scores in all domains. In seen and unseen domains, our neural GRU pipeline and reference approach UPF-FORGe obtained the best results, respectively.

[6]Inter-annotator agreement for the evaluated aspects ranged from 0.26 (Reference) to 0.93 (Input Triples), with an average Krippendorff $\alpha$ of 0.67.

**Human Evaluation** In all domains, neural GRU pipeline and UPF-FORGe were rated the highest in fluency by the participants of the evaluation. In seen ones, both our neural pipeline approaches (GRU and Transformer) were rated the best, whereas UPF-FORGe was considered the most fluent approach in unseen domains.

UPF-FORGe was also rated the most semantic approach in all domains, followed by neural GRU and Majority pipeline approaches. For seen domains, similar to the fluency ratings, both our neural pipeline approaches were rated the highest, whereas UPF-FORGe was considered the most semantic approach in unseen domains.

**Qualitative Analysis** In general, UPF-FORGe emerges as the system which follows the input the best: 91% of the evaluated trials verbalized the input triples. Moreover, the annotators did not find any grammatical mistakes in the output of this approach.

When focusing on the neural pipeline approaches, we found that in all the steps up to Text Structuring, the recurrent networks retained more information than the Transformer. However, 68% of the Transformer's text trials contained all the input triples, against 67% of the GRU's trials. As in Experiment 1, we see that recurrent networks as GRUs are better in ordering and structuring the discourse, but is outperformed by the Transformer in the Lexicalization step. In terms of fluency, we did not see a substantial difference between both kinds of approaches.

Regarding our end-to-end trials, different from the pipeline ones, less than a half verbalized all the input triples. Moreover, the end-to-end outputs also constantly contained more information than there were in the non-linguistic input.

## 8 Discussion

This study introduced a systematic comparison between pipeline and end-to-end architectures for data-to-text generation, exploring the role of deep neural networks in the process. In this section we answer the two introduced research questions and additional topics based on our findings.

**How well do deep learning methods perform as individual modules in a data-to-text pipeline?** In comparison with Random and Majority baselines, we observed that our deep learning implementations registered a higher performance in the pipeline steps on domains seen during training, but their performance dropped considerably on unseen domains, being lower than the baselines.

In the comparison between our GRU and Transformer, the former seems to be better at ordering and structuring the non-linguistic input, whereas the latter performs better in verbalizing an ordered and structured set of triples. The advantage of GRUs over the Transformer in Discourse Ordering and Text Structuring may be its capacity to implicitly take order information into account. On the other hand, the Transformer could have had difficulties caused by the task's design, where triples and sentences were segmented by tags (e.g. `<TRIPLE>` and `<SNT>`), rather than positional embeddings, which suits this model better. In sum, more research needs to be done to set this point.

**How well does a neural pipeline architecture perform compared to a neural end-to-end one?** Our neural pipeline approaches were superior to the end-to-end ones in most tested circumstances: the former generates more fluent texts which better describes data on all domains of the corpus. The difference is most noticeable for unseen domains, where the performance of end-to-end approaches drops considerably. This shows that end-to-end approaches do not generalize as well as the pipeline ones. In the qualitative analysis, we also found that end-to-end generated texts have the problem of describing non-linguistic representations which are not present in the input, also known as Hallucination (Rohrbach et al., 2018).

The example in Figure 2 shows the advantage of our pipeline approaches in comparison with the end-to-end ones. It depicts the texts produced by the proposed approaches for an unseen set during training of 4 triples, where 2 out of the 4 predicates are present in the WebNLG training set (e.g., *birthPlace* and *occupation*). In this context, the pipeline approaches managed to generate a semantic text based on the two seen predicates, whereas the end-to-end approaches hallucinated texts which have no semantic relation with the non-linguistic input.

**Related Work** We compared the proposed approaches with 4 state-of-the-art RDF-to-text systems. Except for Marcheggiani and Perez (2018), all the others are not end-to-end approaches, already directing the field to pipeline architectures. UPF-FORGe is a proper pipeline system with sev-

| | | |
|---|---|---|
| Ace_Wilder | **background** | "solo_singer" |
| Ace_Wilder | ***birthPlace*** | Sweden |
| Ace_Wilder | **birthYear** | 1982 |
| Ace_Wilder | ***occupation*** | Songwriter |

↓

| | |
|---|---|
| GRU | Ace Wilder, born in Sweden, performs as Songwriter. |
| Transformer | Ace Wilder (born in Sweden) was Songwriter. |
| E2E GRU | The test pilot who was born in Willington, who was born in New York, was born in New York and is competing in the competing in the U.S.A. The construction of the city is produced in Mandesh. |
| E2E Trans. | Test pilot Elliot See was born in Dallas and died in St. Louis. |

Figure 2: Example of a set of triples from an unseen domain during training (top) and the corresponding texts produced by our pipeline (e.g., GRU and Transformer) and end-to-end approaches (e.g., E2E GRU and E2E Trans.) (bottom). In the top set of triples, predicates seen during training are highlighted in *italic*, whereas the unseen ones are underlined.

eral sequential steps, Melbourne first generates a delexicalized template to later realize the referring expressions, and Moryossef et al. (2019) splits the process up into Planning, where ordering and structuring are merged, and Realization.

Besides the approach of Marcheggiani and Perez (2018), the ADAPT system, introduced in the WebNLG challenge (Gardent et al., 2017b), is another full end-to-end approach to the task. It obtained the highest results in the seen part of the WebNLG corpus (BLEU = 60.59; METEOR = 0.44). However, the results drastically dropped on the unseen part of the dataset (BLEU = 10.53; METEOR = 0.19). Such results correlate with our findings showing the difficult of end-to-end approaches to generalize to new domains.

By obtaining the best results in almost all the evaluated metrics, UPF-FORGe emerges as the best reference system, showing again the advantage of generating text from non-linguistic data in several explicit intermediate representations. However, it is important to observe that the advantage of UPF-FORGe over our pipeline approaches is the fact that it was designed taking the seen and unseen domains of the corpus into account. So in practice, there was no "unseen" domains for UPF-FORGe. In a fair comparison between this reference system with our neural pipeline approaches in only seen domains, we may see that ours are rated higher in almost all the evaluated metrics.

**General Applicability** Besides our findings, the results for other benchmarks, such as Elder et al. (2019), suggest that pipeline approaches can work well in the context of neural data-to-text generation. Concerning our pipeline approach specifically, although it was designed to convert RDF triples to text, we assume it can be adapted to other domains (and languages) where communicative goals can be linearized and split in units, as in the E2E dataset (Novikova et al., 2017). In future work, we plan to study this in more detail.

**Conclusion** In a systematic comparison, we show that adding supervision during the data-to-text process leads to more fluent text that better describes the non-linguistic input data than full end-to-end approaches, confirming the trends in related work in favor of pipeline architectures.

## Acknowledgments

This work is part of the research program "Discussion Thread Summarization for Mobile Devices" (DISCOSUMO) which is financed by the Netherlands Organization for Scientific Research (NWO). We also acknowledge the three reviewers for their insightful comments.

## References

Luca Anselma and Alessandro Mazzei. 2018. Designing and testing the messages produced by a virtual dietitian. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 244–253, Tilburg University, The Netherlands. Association for Computational Linguistics.

Anja Belz, Mike White, Dominic Espinosa, Eric Kow, Deirdre Hogan, and Amanda Stent. 2011. The first surface realisation shared task: Overview and evaluation results. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 217–226, Nancy, France. Association for Computational Linguistics.

Daniel Braun, Ehud Reiter, and Advaith Siddharthan. 2018. Saferdrive: An NLG-based behaviour change support system for drivers. *Natural Language Engineering*, 24(4):551–588.

Thiago Castro Ferreira, Diego Moussallem, Ákos Kádár, Sander Wubben, and Emiel Krahmer. 2018a. NeuralREG: An end-to-end approach to referring expression generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1959–1969. Association for Computational Linguistics.

Thiago Castro Ferreira, Diego Moussallem, Emiel Krahmer, and Sander Wubben. 2018b. Enriching the WebNLG corpus. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 171–176. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar.

Ondřej Dušek and Filip Jurcicek. 2016. Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 45–51, Berlin, Germany. Association for Computational Linguistics.

Henry Elder, Jennifer Foster, James Barry, and Alexander O'Connor. 2019. Designing a symbolic intermediate representation for neural surface realization. In *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation*, pages 65–73, Minneapolis, Minnesota. Association for Computational Linguistics.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017a. Creating training corpora for NLG micro-planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, ACL'17, pages 179–188, Vancouver, Canada. Association for Computational Linguistics.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017b. The WebNLG challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation*, INLG'17, pages 124–133, Santiago de Compostela, Spain. Association for Computational Linguistics.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61:65–170.

Sebastian Gehrmann, Falcon Dai, Henry Elder, and Alexander Rush. 2018. End-to-end content and plan selection for data-to-text generation. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 46–56, Tilburg University, The Netherlands. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Emiel Krahmer and Kees van Deemter. 2012. Computational generation of referring expressions: A survey. *Computational Linguistics*, 38(1):173–218.

Alon Lavie and Abhaya Agarwal. 2007. Meteor: An automatic metric for MT evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT'07, pages 228–231, Prague, Czech Republic.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, EMNLP'16, pages 1203–1213, Austin, Texas. Association for Computational Linguistics.

Chris van der Lee, Emiel Krahmer, and Sander Wubben. 2017. PASS: A Dutch data-to-text system for soccer, targeted towards specific audiences. In *Proceedings of the 10th International Conference on Natural Language Generation*, INLG'2017, pages 95–104, Santiago de Compostela, Spain. Association for Computational Linguistics.

Leo Leppänen, Myriam Munezero, Mark Granroth-Wilding, and Hannu Toivonen. 2017. Data-driven news generation for automated journalism. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 188–197. Association for Computational Linguistics.

Diego Marcheggiani and Laura Perez. 2018. Deep graph convolutional encoders for structured data to text generation. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 1–9. Association for Computational Linguistics.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. What to talk about and how? selective generation using LSTMs with coarse-to-fine alignment. In *Proceedings of the 2016 Conference of the North*

*American Chapter of the Association for Computational Linguistics: Human Language Technologies*, HLT-NAACL'16, pages 720–730, San Diego, California. Association for Computational Linguistics.

Simon Mille, Anja Belz, Bernd Bohnet, Yvette Graham, Emily Pitler, and Leo Wanner. 2018. The first multilingual surface realisation shared task (SR'18): Overview and evaluation results. In *Proceedings of the First Workshop on Multilingual Surface Realisation*, pages 1–12, Melbourne, Australia. Association for Computational Linguistics.

Simon Mille, Stamatia Dasiopoulou, and Leo Wanner. 2019. A portable grammar-based NLG system for verbalization of structured data. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, pages 1054–1056. ACM.

Amit Moryossef, Yoav Goldberg, and Ido Dagan. 2019. Step-by-step: Separating planning from realization in neural data-to-text generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2267–2277, Minneapolis, Minnesota. Association for Computational Linguistics.

Jekaterina Novikova, Ondrej Dusek, and Verena Rieser. 2017. The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 201–206, Saarbrücken, Germany.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, ACL'02, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Franois Portet, Ehud Reiter, Albert Gatt, Jim Hunter, Somayajulu Sripada, Yvonne Freer, and Cindy Sykes. 2009. Automatic generation of textual summaries from neonatal intensive care data. *Artificial Intelligence*, 173(78):789 – 816.

Ehud Reiter and Robert Dale. 2000. *Building natural language generation systems*. Cambridge University Press, New York, NY, USA.

Anna Rohrbach, Lisa Anne Hendricks, Kaylee Burns, Trevor Darrell, and Kate Saenko. 2018. Object hallucination in image captioning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4035–4045, Brussels, Belgium. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, EMNLP'15, pages 1711–1721, Lisbon, Portugal. Association for Computational Linguistics.

# MoverScore: Text Generation Evaluating with Contextualized Embeddings and Earth Mover Distance

**Wei Zhao**[†], **Maxime Peyrard**[†], **Fei Liu**[‡], **Yang Gao**[†], **Christian M. Meyer**[†], **Steffen Eger**[†]
[†] Computer Science Department, Technische Universität Darmstadt, Germany
[‡] Computer Science Department, University of Central Florida, US
zhao@aiphes.tu-darmstadt.de, maxime.peyrard@epfl.ch
feiliu@cs.ucf.edu, yang.gao@rhul.ac.uk
meyer@ukp.informatik.tu-darmstadt.de
eger@aiphes.tu-darmstadt.de

## Abstract

A robust evaluation metric has a profound impact on the development of text generation systems. A desirable metric compares system output against references based on their semantics rather than surface forms. In this paper we investigate strategies to encode system and reference texts to devise a metric that shows a high correlation with human judgment of text quality. We validate our new metric, namely MoverScore, on a number of text generation tasks including summarization, machine translation, image captioning, and data-to-text generation, where the outputs are produced by a variety of neural and non-neural systems. Our findings suggest that metrics combining contextualized representations with a distance measure perform the best. Such metrics also demonstrate strong generalization capability across tasks. For ease-of-use we make our metrics available as web service.[1]

## 1 Introduction

The choice of evaluation metric has a significant impact on the assessed quality of natural language outputs generated by a system. A desirable metric assigns a single, real-valued score to the system output by comparing it with one or more reference texts for content matching. Many natural language generation (NLG) tasks can benefit from robust and unbiased evaluation, including text-to-text (*machine translation* and *summarization*), data-to-text (*response generation*), and image-to-text (*captioning*) (Gatt and Krahmer, 2018). Without proper evaluation, it can be difficult to judge on system competitiveness, hindering the development of advanced algorithms for text generation.

It is an increasingly pressing priority to develop better evaluation metrics given the recent advances in neural text generation. Neural models provide the flexibility to copy content from source text as well as generating unseen words (See et al., 2017). This aspect is hardly covered by existing metrics. With greater flexibility comes increased demand for unbiased evaluation. Diversity-promoting objectives make it possible to generate diverse natural language descriptions (Li et al., 2016; Wiseman et al., 2018). But standard evaluation metrics including BLEU (Papineni et al., 2002) and ROUGE (Lin, 2004) compute the scores based primarily on n-gram co-occurrence statistics, which are originally proposed for diagnostic evaluation of systems but not capable of evaluating text quality (Reiter, 2018), as they are not designed to measure if, and to what extent, the system and reference texts with distinct surface forms have conveyed the same meaning. Recent effort on the applicability of these metrics reveals that while compelling text generation system ascend on standard metrics, the text quality of system output is still hard to be improved (Böhm et al., 2019).

Our goal in this paper is to devise an automated evaluation metric assigning a single holistic score to any system-generated text by comparing it against human references for content matching. We posit that it is crucial to provide a holistic measure attaining high correlation with human judgments so that various neural and non-neural text generation systems can be compared directly. Intuitively, the metric assigns a perfect score to the system text if it conveys the same meaning as the reference text. Any deviation from the reference content can then lead to a reduced score, e.g., the system text contains more (or less) content than the reference, or the system produces ill-formed text that fails to deliver the intended meaning.

We investigate the effectiveness of a spectrum of distributional semantic representations to encode system and reference texts, allowing them to be compared for semantic similarity across

[1] Our code is publicly available at http://tiny.cc/vsqtbz



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 563–578, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

multiple natural language generation tasks. Our new metric quantifies the semantic distance between system and reference texts by harnessing the power of contextualized representations (Peters et al., 2018; Devlin et al., 2018) and a powerful distance metric (Rubner et al., 2000) for better content matching. Our contributions can be summarized as follows:

- We formulate the problem of evaluating generation systems as measuring the semantic distance between system and reference texts, assuming powerful continuous representations can encode any type of semantic and syntactic deviations.
- We investigate the effectiveness of existing contextualized representations and Earth Mover's Distance (Rubner et al., 2000) for comparing system predictions and reference texts, leading to our new automated evaluation metric that achieves high correlation with human judgments of text quality.
- Our metric outperforms or performs comparably to strong baselines on four text generation tasks including summarization, machine translation, image captioning, and data-to-text generation, suggesting this is a promising direction moving forward.

## 2 Related Work

It is of fundamental importance to design evaluation metrics that can be applied to natural language generation tasks of similar nature, including summarization, machine translation, data-to-text generation, image captioning, and many others. All these tasks involve generating texts of sentence or paragraph length. The system texts are then compared with one or more reference texts of similar length for semantic matching, whose scores indicate how well the systems perform on each task. In the past decades, however, evaluation of these natural language generation tasks has largely been carried out independently within each area.

**Summarization** A dominant metric for summarization evaluation is ROUGE (Lin, 2004), which measures the degree of lexical overlap between a system summary and a set of reference summaries. Its variants consider overlap of unigrams (-1), bigrams (-2), unigrams and skip bigrams with a maximum gap of 4 words (-SU4), longest common subsequences (-L) and its weighted version (-W-1.2), among others. Metrics such as Pyramid (Nenkova and Passonneau, 2004) and BE (Hovy et al., 2006; Tratz and Hovy, 2008) further compute matches of content units, e.g., (head-word, modifier) tuples, that often need to be manually extracted from reference summaries. These metrics achieve good correlations with human judgments in the past. However, they are not general enough to account for the relatedness between abstractive summaries and their references, as a system abstract can convey the same meaning using different surface forms. Furthermore, large-scale summarization datasets such as CNN/Daily Mail (Hermann et al., 2015) and Newsroom (Grusky et al., 2018) use a *single reference* summary, making it harder to obtain unbiased results when only lexical overlap is considered during summary evaluation.

**Machine Translation** A number of metrics are commonly used in MT evaluation. Most of these metrics compare system and reference translations based on surface forms such as word/character n-gram overlaps and edit distance, but not the meanings they convey. BLEU (Papineni et al., 2002) is a precision metric measuring how well a system translation overlaps with human reference translations using n-gram co-occurrence statistics. Other metrics include SentBLEU, NIST, chrF, TER, WER, PER, CDER, and METEOR (Lavie and Agarwal, 2007) that are used and described in the WMT metrics shared task (Bojar et al., 2017; Ma et al., 2018). RUSE (Shimanaka et al., 2018) is a recent effort to improve MT evaluation by training sentence embeddings on large-scale data obtained in other tasks. Additionally, preprocessing reference texts is crucial in MT evaluation, e.g., normalization, tokenization, compound splitting, etc. If not handled properly, different preprocessing strategies can lead to inconsistent results using word-based metrics (Post, 2018).

**Data-to-text Generation** BLEU can be poorly suited to evaluating data-to-text systems such as dialogue response generation and image captioning. These systems are designed to generate texts with lexical and syntactic variation, communicating the same information in many different ways. BLEU and similar metrics tend to reward systems that use the same wording as reference texts, causing repetitive word usage that is deemed undesirable to humans (Liu et al., 2016). In a similar vein, evaluating the quality of image captions can be challenging. CIDEr (Vedantam et al., 2015) uses tf-idf weighted n-grams for similarity estimation; and SPICE (Anderson et al., 2016) incorporates

synonym matching over scene graphs. Novikova et al. (2017) examine a large number of word- and grammar-based metrics and demonstrate that they only weakly reflect human judgments of system outputs generated by data-driven, end-to-end natural language generation systems.

**Metrics based on Continuous Representations** Moving beyond traditional metrics, we envision a new generation of automated evaluation metrics comparing system and reference texts based on semantics rather than surface forms to achieve better correlation with human judgments. A number of previous studies exploit static word embeddings (Ng and Abrecht, 2015; Lo, 2017) and trained classifers (Peyrard et al., 2017; Shimanaka et al., 2018) to improve semantic similarity estimation, replacing lexical overlaps.

In contemporaneous work, Zhang et al. (2019) describe a method comparing system and reference texts for semantic similarity leveraging the BERT representations (Devlin et al., 2018), which can be viewed as a special case of our metrics and will be discussed in more depth later. More recently, Clark et al. (2019) present a semantic metric relying on sentence mover's similarity and the ELMo representations (Peters et al., 2018) and apply them to summarization and essay scoring. Mathur et al. (2019) introduce unsupervised and supervised metrics based on the BERT representations to improve MT evaluation, while Peyrard (2019a) provides a composite score combining redundancy, relevance and informativeness to improve summary evaluation.

In this paper, we seek to accurately measure the (dis)similarity between system and reference texts drawing inspiration from contextualized representations and Word Mover's Distance (WMD; Kusner et al., 2015). WMD finds the "traveling distance" of moving from the word frequency distribution of the system text to that of the reference, which is essential to capture the (dis)similarity between two texts. Our metrics differ from the contemporaneous work in several facets: (i) we explore the granularity of embeddings, leading to two variants of our metric, word mover and sentence mover; (ii) we investigate the effectiveness of diverse pretrained embeddings and finetuning tasks; (iii) we study the approach to consolidate layer-wise information within contextualized embeddings; (iii) our metrics demonstrate strong generalization capability across four tasks, oftentimes outperforming the supervised ones. We now describe our method in detail.

## 3 Our MoverScore Meric

We have motivated the need for better metrics capable of evaluating disparate NLG tasks. We now describe our metric, namely MoverScore, built upon a combination of (i) contextualized representations of system and reference texts and (ii) a distance between these representations measuring the semantic distance between system outputs and references. It is particularly important for a metric to not only capture the amount of *shared content* between two texts, i.e., intersect(A,B), as is the case with many semantic textual similarity measures (Peters et al., 2018; Devlin et al., 2018); but also to accurately reflect to what extent the system text has *deviated* from the reference, i.e., union(A,B) - intersect(A,B), which is the intuition behind using a distance metric.

### 3.1 Measuring Semantic Distance

Let $\boldsymbol{x} = (\mathrm{x}_1, \ldots, \mathrm{x}_m)$ be a sentence viewed as a sequence of words. We denote by $\boldsymbol{x}^n$ the sequence of $n$-grams of $\boldsymbol{x}$ (i.e., $\boldsymbol{x}^1 = \boldsymbol{x}$ is the sequence of words and $\boldsymbol{x}^2$ is the sequence of bigrams). Furthermore, let $\boldsymbol{f}_{\boldsymbol{x}^n} \in \mathbb{R}_+^{|\boldsymbol{x}^n|}$ be a vector of weights, one weight for each $n$-gram of $\boldsymbol{x}^n$. We can assume $\boldsymbol{f}_{\boldsymbol{x}^n}^\intercal \mathbf{1} = 1$, making $\boldsymbol{f}_{\boldsymbol{x}^n}$ a distribution over $n$-grams. Intuitively, the effect of some $n$-grams like those including function words can be downplayed by giving them lower weights, e.g., using Inverse Document Frequency (IDF).

Word Mover's Distance (WMD) (Kusner et al., 2015), a special case of Earth Mover's Distance (Rubner et al., 2000), measures semantic distance between texts by aligning semantically similar words and finding the amount of flow traveling between these words. It was shown useful for text classification and textual similarity tasks (Kusner et al., 2015). Here, we formulate a generalization operating on $n$-grams. Let $\boldsymbol{x}$ and $\boldsymbol{y}$ be two sentences viewed as sequences of $n$-grams: $\boldsymbol{x}^n$ and $\boldsymbol{y}^n$. If we have a distance metric $d$ between $n$-grams, then we can define the *transportation cost* matrix $\boldsymbol{C}$ such that $\boldsymbol{C}_{ij} = d(\mathrm{x}_i^n, \mathrm{y}_j^n)$ is the distance between the $i$-th $n$-gram of $\boldsymbol{x}$ and the $j$-th $n$-gram of $\boldsymbol{y}$. The WMD between the two sequences of $n$-grams $\boldsymbol{x}^n$ and $\boldsymbol{y}^n$ with associated $n$-gram weights $\boldsymbol{f}_{\boldsymbol{x}^n}$ and $\boldsymbol{f}_{\boldsymbol{y}^n}$ is then given by:

$$\mathrm{WMD}(\boldsymbol{x}^n, \boldsymbol{y}^n) := \min_{\boldsymbol{F} \in \mathbb{R}^{|\boldsymbol{x}^n| \times |\boldsymbol{y}^n|}} \langle \boldsymbol{C}, \boldsymbol{F} \rangle,$$
$$\text{s.t. } \boldsymbol{F}\mathbf{1} = \boldsymbol{f}_{\boldsymbol{x}^n}, \quad \boldsymbol{F}^\intercal \mathbf{1} = \boldsymbol{f}_{\boldsymbol{y}^n}.$$

where $\boldsymbol{F}$ is the *transportation flow* matrix with $\boldsymbol{F}_{ij}$ denoting the amount of flow traveling from the $i$-th $n$-gram $\mathrm{x}_i^n$ in $\boldsymbol{x}^n$ to the $j$-th $n$-gram $\mathrm{y}_j^n$ in $\boldsymbol{y}^n$. Here, $\langle \boldsymbol{C}, \boldsymbol{F} \rangle$ denotes the sum of all matrix entries of the matrix $\boldsymbol{C} \odot \boldsymbol{F}$, where $\odot$ denotes element-wise multiplication. Then $\mathrm{WMD}(\boldsymbol{x}^n, \boldsymbol{y}^n)$ is the minimal transportation cost between $\boldsymbol{x}^n$ and $\boldsymbol{y}^n$ where $n$-grams are weighted by $\boldsymbol{f}_{\boldsymbol{x}^n}$ and $\boldsymbol{f}_{\boldsymbol{y}^n}$.

In practice, we compute the Euclidean distance between the embedding representations of $n$-grams: $d(\mathrm{x}_i^n, \mathrm{y}_j^n) = ||E(\mathrm{x}_i^n) - E(\mathrm{y}_j^n)||_2$ where $E$ is the embedding function which maps an $n$-gram to its vector representation. Usually, *static* word embeddings like word2vec are used to compute $E$ but these cannot capture word order or compositionality. Alternatively, we investigate contextualized embeddings like ELMo and BERT because they encode information about the whole sentence into each word vector.

We compute the $n$-gram embeddings as the weighted sum over its word embeddings. Formally, if $\boldsymbol{x}_i^n = (\mathrm{x}_i, \ldots, \mathrm{x}_{i+n-1})$ is the $i$-th $n$-gram from sentence $\boldsymbol{x}$, its embedding is given by:

$$E(\boldsymbol{x}_i^n) = \sum_{k=i}^{i+n-1} \mathrm{idf}(\mathrm{x}_k) \cdot E(\mathrm{x}_k) \qquad (1)$$

where $\mathrm{idf}(\mathrm{x}_k)$ is the IDF of word $\mathrm{x}_k$ computed from all sentences in the corpus and $E(\mathrm{x}_k)$ is its word vector. Furthermore, the weight associated to the $n$-gram $\boldsymbol{x}_i^n$ is given by:

$$\boldsymbol{f}_{\boldsymbol{x}_i^n} = \frac{1}{Z} \sum_{k=i}^{i+n-1} \mathrm{idf}(\mathrm{x}_k) \qquad (2)$$

where $Z$ is a normalizing constant s.t. $\boldsymbol{f}_{\boldsymbol{x}^n}^\top \mathbf{1} = 1$,

In the limiting case where $n$ is larger than the sentence's size, $\boldsymbol{x}^n$ contains only one $n$-gram: the whole sentence. Then $\mathrm{WMD}(\boldsymbol{x}^n, \boldsymbol{y}^n)$ reduces to computing the distance between the two sentence embeddings, namely Sentence Mover's Distance (SMD), denoted as:

$$\mathrm{SMD}(\boldsymbol{x}^n, \boldsymbol{y}^n) := ||E(\boldsymbol{x}_1^{l_x}) - E(\boldsymbol{y}_1^{l_y})||$$

where $l_x$ and $l_y$ are the size of sentences.

**Hard and Soft Alignments** In contemporaneous work, BERTScore (Zhang et al., 2019) also models the semantic distance between system and reference texts for evaluating text generation systems. As shown in Figure 1, BERTScore (precision/recall) can be intuitively viewed as hard alignments (one-to-one) for words in a sentence pair, where each word in one sequence travels to the most semantically similar word in the other sequence. In contrast, MoverScore goes beyond BERTScore as it relies on soft alignments (many-to-one) and allows to map semantically related words in one sequence to the respective word in the other sequence by solving a constrained optimization problem: finding the minimum effort to transform between two texts.

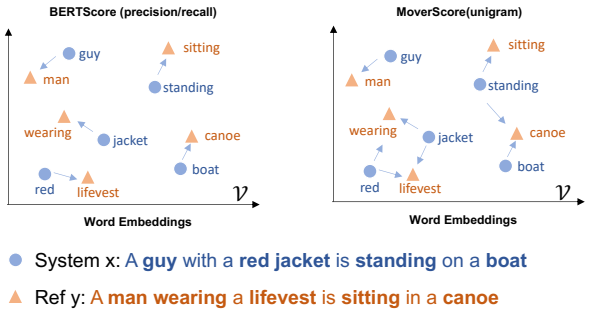


Figure 1: An illustration of MoverScore and BERTScore.

The formulation of Word Mover's Distance provides an important possibility to bias the metric towards precision or recall by using an asymmetric transportation cost matrix, which bridges a gap between MoverScore and BERTScore:

**Proposition 1** *BERTScore (precision/recall) can be represented as a (non-optimized) Mover Distance $\langle \boldsymbol{C}, \boldsymbol{F} \rangle$, where $\boldsymbol{C}$ is a transportation cost matrix based on BERT and $\boldsymbol{F}$ is a uniform transportation flow matrix.*[2]

### 3.2 Contextualized Representations

The task formulation naturally lends itself to deep contextualized representations for inducing word vectors $E(\mathrm{x}_i)$. Despite the recent success of multi-layer attentive neural architectures (Devlin et al., 2018; Peters et al., 2018), consolidating layer-wise information remains an open problem as different layers capture information at disparate scales and task-specific layer selection methods may be limited (Liu et al., 2018, 2019). Tenney et al. (2019) found that a scalar mix of output layers trained from task-dependent supervisions would be effective in a deep transformer-based model. Instead, we investigate aggregation functions to consolidate layer-wise information, forming stationary representations of words without supervision.

Consider a sentence $\boldsymbol{x}$ passed through contextualized encoders such as ELMo and BERT with $L$ layers. Each layer of the encoders produces a vec-

[2] See the proof in the appendix.

tor representation for each word $x_i$ in $\boldsymbol{x}$. We denote by $\boldsymbol{z}_{i,l} \in \mathbb{R}^d$ the representation given by layer $l$, a $d$-dimensional vector. Overall, $x_i$ receives $L$ different vectors $(\boldsymbol{z}_{i,1}, \ldots, \boldsymbol{z}_{i,L})$. An aggregation $\phi$ maps these $L$ vectors to one final vector:

$$E(x_i) = \phi(\boldsymbol{z}_{i,1}, \ldots, \boldsymbol{z}_{i,L}) \quad (3)$$

where $E(x_i)$ is the aggregated representation of the word $x_i$.

We study two alternatives for $\phi$: (i) the concatenation of power means (Rücklé et al., 2018) as a generalized pooling mechanism, and (ii) a routing mechanism for aggregation (Zhao et al., 2018, 2019). We relegate the routing method to appendix, as it does not yield better results than power means.

**Power Means** Power means is an effective generalization of pooling techniques for aggregating information. It computes a non-linear average of a set of values with an exponent $p$ (Eq. (4)). Following Rücklé et al. (2018), we exploit power means to aggregate vector representations $(\boldsymbol{z}_{i,l})_{l=1}^{L}$ pertaining to the $i$-th word from all layers of a deep neural architecture. Let $p \in \mathbb{R} \cup \{\pm\infty\}$, the $p$-mean of $(\boldsymbol{z}_{i,1}, \ldots, \boldsymbol{z}_{i,L})$ is:

$$\boldsymbol{h}_i^{(p)} = \left( \frac{\boldsymbol{z}_{i,1}^p + \cdots + \boldsymbol{z}_{i,L}^p}{L} \right)^{1/p} \in \mathbb{R}^d \quad (4)$$

where exponentiation is applied elementwise. This generalized form can induce common named means such as arithmetic mean ($p = 1$) and geometric mean ($p = 0$). In extreme cases, a power mean reduces to the minimum value of the set when $p = -\infty$, and the maximum value when $p = +\infty$. The concatenation of $p$-mean vectors we use in this paper is denoted by:

$$E(x_i) = \mathbf{h}_i^{(p_1)} \oplus \cdots \oplus \mathbf{h}_i^{(p_K)} \quad (5)$$

where $\oplus$ is vector concatenation; $\{p_1, \ldots, p_K\}$ are exponent values, and we use $K = 3$ with $p = 1, \pm\infty$ in this work.

### 3.3 Summary of MoverScore Variations

We investigate our MoverScore along four dimensions: (i) the granularity of embeddings, i.e., the size of $n$ for $n$-grams, (ii) the choice of pretrained embedding mechanism, (iii) the fine-tuning task used for BERT[3] (iv) the aggregation technique ($p$-means or routing) when applicable.

[3]ELMo usually requires heavy layers on the top, which restricts the power of fine-tuning tasks for ELMo.

**Granularity** We used $n = 1$ and $n = 2$ as well as full sentences ($n =$ size of the sentence).

**Embedding Mechanism** We obtained word embeddings from three different methods: *static* embedding with word2vec as well as contextualized embedding with ELMo and BERT. If $n > 1$, $n$-gram embeddings are calculated by Eq. (1). Note that they represent sentence embeddings when $n =$ size of the sentence.

**Fine-tuning Tasks** Natural Language Inference (NLI) and paraphrasing pose high demands in understanding sentence meaning. This motivated us to fine-tune BERT representations on two NLI datasets, MultiNLI and QANLI, and one Paraphrase dataset, QQP—the largest datasets in GLUE (Wang et al., 2018). We fine-tune BERT on each of these, yielding different contextualized embeddings for our general evaluation metrics.

**Aggregation** For ELMo, we aggregate word representations given by all three ELMo layers, using $p$-means or routing (see the appendix). Word representations in BERT are aggregated from the last five layers, using $p$-means or routing since the representations in the initial layers are less suited for use in downstream tasks (Liu et al., 2019).

## 4 Empirical Evaluation

In this section, we measure the quality of different metrics on four tasks: *machine translation*, *text summarization*, *image captioning* and *dialogue generation*. Our major focus is to study the correlation between different metrics and human judgment. We employ two text encoders to embed $n$-grams: $\text{BERT}_{\text{base}}$, which uses a 12-layer transformer, and $\text{ELMO}_{\text{original}}$, which uses a 3-layer BiLSTM. We use Pearson's $r$ and Spearman's $\rho$ to measure the correlation. We consider two variants of MoverScore: *word mover* and *sentence mover*, described below.

**Word Mover** We denote our word mover notation containing four ingredients as: *WMD-Granularity+Embedding+Finetune+Aggregation*. For example, WMD-1+BERT+MNLI+PMEANS represents the semantic metric using word mover distance where unigram-based word embeddings fine-tuned on MNLI are aggregated by $p$-means.

**Sentence Mover** We denote our sentence mover notation with three ingredients as: *SMD+Embedding+Finetune+Aggregation*. For example, SMD+W2V represents the semantic

metric using sentence mover distance where two sentence embeddings are computed as the weighted sum over their word2vec embeddings by Eq. (1).

**Baselines** We select multiple strong baselines for each task for comparison: SentBLEU, METEOR++ (Guo et al., 2018), and a supervised metric RUSE for machine translation; ROUGE-1 and ROUGE-2 and a supervised metric $S^3_{best}$ (Peyrard et al., 2017) for text summarization; BLEU and METEOR for dialogue response generation, CIDEr, SPICE, METEOR and a supervised metric LEIC (Cui et al., 2018) for image captioning. We also report BERTScore (Zhang et al., 2019) for all tasks (see §2). Due to the page limit, we only compare with the strongest baselines, the rest can be found in the appendix.

### 4.1 Machine Translation

**Data** We obtain the source language sentences, their system and reference translations from the WMT 2017 news translation shared task (Bojar et al., 2017). We consider 7 language pairs: from German (de), Chinese (zh), Czech (cs), Latvian (lv), Finnish (fi), Russian (ru), and Turkish (tr), resp. to English. Each language pair has approximately 3,000 sentences, and each sentence has one reference translation and multiple system translations generated by participating systems. For each system translation, at least 15 human assessments are independently rated for quality.

**Results** Table 1: In all language pairs, the best correlation is achieved by our word mover metrics that use a BERT pretrained on MNLI as the embedding generator and PMeans to aggregate the embeddings from different BERT layers, i.e., WMD-1/2+BERT+MNLI+PMeans. Note that our unsupervised word mover metrics even outperforms RUSE, a supervised metric. We also find that our word mover metrics outperforms the sentence mover. We conjecture that important information is lost in such a sentence representation while transforming the whole sequence of word vectors into one sentence embedding by Eq. (1).

### 4.2 Text Summarization

We use two summarization datasets from the Text Analysis Conference (TAC)[4]: TAC-2008 and TAC-2009, which contain 48 and 44 *clusters*, respectively. Each cluster includes 10 news articles (on the same topic), four reference summaries, and 57 (in TAC-2008) or 55 (in TAC-2009) system summaries generated by the participating systems. Each summary (either reference or system) has fewer than 100 words, and receives two human judgment scores: the *Pyramid* score (Nenkova and Passonneau, 2004) and the *Responsiveness* score. Pyramid measures how many important semantic content units in the reference summaries are covered by the system summary, while Responsiveness measures how well a summary responds to the overall quality combining both content and linguistic quality.

**Results** Tables 2: We observe that lexical metrics like ROUGE correlate above-moderate on TAC 2008 and 2009 datasets. In contrast, these metrics perform poorly on other tasks like Dialogue Generation (Novikova et al., 2017) and Image Captioning (Anderson et al., 2016). Apparently, strict matches on surface forms seems reasonable for *extractive* summarization datasets. However, we still see that our word mover metrics, i.e., WMD-1+BERT+MNLI+PMeans, perform better than or come close to even the supervised metric $S^3_{best}$.

### 4.3 Data-to-text Generation

We use two task-oriented dialogue datasets: BAGEL (Mairesse et al., 2010) and SFHOTEL (Wen et al., 2015), which contains 202 and 398 instances of Meaning Representation (MR). Each MR instance includes multiple references, and roughly two system utterances generated by different neural systems. Each system utterance receives three human judgment scores: *informativeness*, *naturalness* and *quality* score (Novikova et al., 2017). Informativeness measures how much information a system utterance provides with respect to an MR. Naturalness measures how likely a system utterance is generated by native speakers. Quality measures how well a system utterance captures fluency and grammar.

**Results** Tables 3: Interestingly, no metric produces an even moderate correlation with human judgments, including our own. We speculate that current contextualizers are poor at representing named entities like hotels and place names as well as numbers appearing in system and reference texts. However, best correlation is still achieved by our word mover metrics combining contextualized representations.

[4] http://tac.nist.gov

| Setting | Metrics | Direct Assessment cs-en | de-en | fi-en | lv-en | ru-en | tr-en | zh-en | Average |
|---|---|---|---|---|---|---|---|---|---|
| BASELINES | METEOR++ | 0.552 | 0.538 | 0.720 | 0.563 | 0.627 | 0.626 | 0.646 | 0.610 |
| | RUSE(*) | 0.624 | 0.644 | 0.750 | 0.697 | 0.673 | 0.716 | 0.691 | 0.685 |
| | BERTSCORE-F1 | 0.670 | 0.686 | 0.820 | 0.710 | 0.729 | 0.714 | 0.704 | 0.719 |
| SENT-MOVER | SMD + W2V | 0.438 | 0.505 | 0.540 | 0.442 | 0.514 | 0.456 | 0.494 | 0.484 |
| | SMD + ELMO + PMEANS | 0.569 | 0.558 | 0.732 | 0.525 | 0.581 | 0.620 | 0.584 | 0.595 |
| | SMD + BERT + PMEANS | 0.607 | 0.623 | 0.770 | 0.639 | 0.667 | 0.641 | 0.619 | 0.652 |
| | SMD + BERT + MNLI + PMEANS | 0.616 | 0.643 | 0.785 | 0.660 | 0.664 | 0.668 | 0.633 | 0.667 |
| WORD-MOVER | WMD-1 + W2V | 0.392 | 0.463 | 0.558 | 0.463 | 0.456 | 0.485 | 0.481 | 0.471 |
| | WMD-1 + ELMO + PMEANS | 0.579 | 0.588 | 0.753 | 0.559 | 0.617 | 0.679 | 0.645 | 0.631 |
| | WMD-1 + BERT + PMEANS | 0.662 | 0.687 | 0.823 | 0.714 | 0.735 | 0.734 | 0.719 | 0.725 |
| | WMD-1 + BERT + MNLI + PMEANS | 0.670 | 0.708 | **0.835** | **0.746** | **0.738** | 0.762 | **0.744** | **0.743** |
| | WMD-2 + BERT + MNLI + PMEANS | **0.679** | **0.710** | 0.832 | 0.745 | 0.736 | **0.763** | 0.740 | **0.743** |

Table 1: Absolute Pearson correlations with segment-level human judgments in 7 language pairs on WMT17 dataset.

| Setting | Metrics | TAC-2008 Responsiveness $r$ | Responsiveness $\rho$ | Pyramid $r$ | Pyramid $\rho$ | TAC-2009 Responsiveness $r$ | Responsiveness $\rho$ | Pyramid $r$ | Pyramid $\rho$ |
|---|---|---|---|---|---|---|---|---|---|
| BASELINES | $S^3_{best}$ (*) | 0.715 | 0.595 | 0.754 | 0.652 | 0.738 | **0.595** | **0.842** | **0.731** |
| | ROUGE-1 | 0.703 | 0.578 | 0.747 | 0.632 | 0.704 | 0.565 | 0.808 | 0.692 |
| | ROUGE-2 | 0.695 | 0.572 | 0.718 | 0.635 | 0.727 | 0.583 | 0.803 | 0.694 |
| | BERTSCORE-F1 | 0.724 | 0.594 | 0.750 | 0.649 | 0.739 | 0.580 | 0.823 | 0.703 |
| SENT-MOVER | SMD + W2V | 0.583 | 0.469 | 0.603 | 0.488 | 0.577 | 0.465 | 0.670 | 0.560 |
| | SMD + ELMO + PMEANS | 0.631 | 0.472 | 0.631 | 0.499 | 0.663 | 0.498 | 0.726 | 0.568 |
| | SMD + BERT + PMEANS | 0.658 | 0.530 | 0.664 | 0.550 | 0.670 | 0.518 | 0.731 | 0.580 |
| | SMD + BERT + MNLI + PMEANS | 0.662 | 0.525 | 0.666 | 0.552 | 0.667 | 0.506 | 0.723 | 0.563 |
| WORD-MOVER | WMD-1 + W2V | 0.669 | 0.549 | 0.665 | 0.588 | 0.698 | 0.520 | 0.740 | 0.647 |
| | WMD-1 + ELMO + PMEANS | 0.707 | 0.554 | 0.726 | 0.601 | 0.736 | 0.553 | 0.813 | 0.672 |
| | WMD-1 + BERT + PMEANS | 0.729 | 0.595 | 0.755 | 0.660 | 0.742 | 0.581 | 0.825 | 0.690 |
| | WMD-1 + BERT + MNLI + PMEANS | **0.736** | **0.604** | **0.760** | **0.672** | **0.754** | 0.594 | 0.831 | 0.701 |
| | WMD-2 + BERT + MNLI + PMEANS | 0.734 | 0.601 | 0.752 | 0.663 | 0.753 | 0.586 | 0.825 | 0.694 |

Table 2: Pearson $r$ and Spearman $\rho$ correlations with summary-level human judgments on TAC 2008 and 2009.

### 4.4 Image Captioning

We use a popular image captioning dataset: MS-COCO (Lin et al., 2014), which contains 5,000 images. Each image includes roughly five reference captions, and 12 system captions generated by the participating systems from 2015 COCO Captioning Challenge. For the system-level human correlation, each system receives five human judgment scores: M1, M2, M3, M4, M5 (Anderson et al., 2016). The M1 and M2 scores measure overall quality of the captions while M3, M4 and M5 scores measure correctness, detailedness and saliency of the captions. Following Cui et al. (2018), we compare the Pearson correlation with two system-level scores: M1 and M2, since we focus on studying metrics for the overall quality of the captions, leaving metrics understanding captions in different aspects (correctness, detailedness and saliency) to future work.

**Results** Table 4: Word mover metrics outperform all baselines except for the supervised metric LEIC, which uses more information by considering *both* images and texts.

### 4.5 Further Analysis

**Hard and Soft Alignments** BERTScore is the harmonic mean of BERTScore-Precision and BERTScore-Recall, where both two can be decomposed as a combination of "Hard Mover Distance" (HMD) and BERT (see Prop. 1).

We use the representations in the 9-th BERT layer for fair comparison of BERTScore and MoverScore and show results on the machine translation task in Table 5. MoverScore outperforms both asymmetric HMD factors, while if they are combined via harmonic mean, BERTScore is on par with MoverScore. We conjecture that BERT softens hard alignments of BERTScore as contextualized embeddings encode information about the whole sentence into each word vector. We also observe that WMD-BIGRAMS slightly outperforms WMD-UNIGRAMS on 3 out of 4 language pairs.

| Setting | Metrics | BAGEL Inf | BAGEL Nat | BAGEL Qual | SFHOTEL Inf | SFHOTEL Nat | SFHOTEL Qual |
|---|---|---|---|---|---|---|---|
| BASELINES | BLEU-1 | 0.225 | 0.141 | 0.113 | 0.107 | 0.175 | 0.069 |
| | BLEU-2 | 0.211 | 0.152 | 0.115 | 0.097 | 0.174 | 0.071 |
| | METEOR | 0.251 | 0.127 | 0.116 | 0.111 | 0.148 | 0.082 |
| | BERTSCORE-F1 | 0.267 | 0.210 | **0.178** | 0.163 | 0.193 | 0.118 |
| SENT-MOVER | SMD + W2V | 0.024 | 0.074 | 0.078 | 0.022 | 0.025 | 0.011 |
| | SMD + ELMO + PMEANS | 0.251 | 0.171 | 0.147 | 0.130 | 0.176 | 0.096 |
| | SMD + BERT + PMEANS | 0.290 | 0.163 | 0.121 | 0.192 | 0.223 | 0.134 |
| | SMD + BERT + MNLI + PMEANS | 0.280 | 0.149 | 0.120 | 0.205 | 0.239 | 0.147 |
| WORD-MOVER | WMD-1 + W2V | 0.222 | 0.079 | 0.123 | 0.074 | 0.095 | 0.021 |
| | WMD-1 + ELMO + PMEANS | 0.261 | 0.163 | 0.148 | 0.147 | 0.215 | 0.136 |
| | WMD-1 + BERT + PMEANS | **0.298** | **0.212** | 0.163 | 0.203 | 0.261 | 0.182 |
| | WMD-1 + BERT + MNLI + PMEANS | 0.285 | 0.195 | 0.158 | **0.207** | **0.270** | **0.183** |
| | WMD-2 + BERT + MNLI + PMEANS | 0.284 | 0.194 | 0.156 | 0.204 | 0.270 | 0.182 |

Table 3: Spearman correlation with utterance-level human judgments for BAGEL and SFHOTEL datasets.

| Setting | Metric | M1 | M2 |
|---|---|---|---|
| BASELINES | LEIC(*) | **0.939** | **0.949** |
| | METEOR | 0.606 | 0.594 |
| | SPICE | 0.759 | 0.750 |
| | BERTSCORE-RECALL | 0.809 | 0.749 |
| SENT-MOVER | SMD + W2V | 0.683 | 0.668 |
| | SMD + ELMO + P | 0.709 | 0.712 |
| | SMD + BERT + P | 0.723 | 0.747 |
| | SMD + BERT + M + P | 0.789 | 0.784 |
| WORD-MOVER | WMD-1 + W2V | 0.728 | 0.764 |
| | WMD-1 + ELMO + P | 0.753 | 0.775 |
| | WMD-1 + BERT + P | 0.780 | 0.790 |
| | WMD-1 + BERT + M + P | **0.813** | **0.810** |
| | WMD-2 + BERT + M + P | 0.812 | 0.808 |

Table 4: Pearson correlation with system-level human judgments on MSCOCO dataset. 'M' and 'P' are short names.

| Metrics | cs-en | de-en | fi-en | lv-en |
|---|---|---|---|---|
| RUSE | 0.624 | 0.644 | 0.750 | 0.697 |
| HMD-F1 + BERT | 0.655 | 0.681 | 0.821 | 0.712 |
| HMD-RECALL + BERT | 0.651 | 0.658 | 0.788 | 0.681 |
| HMD-PREC + BERT | 0.624 | 0.669 | 0.817 | 0.707 |
| WMD-UNIGRAM + BERT | 0.651 | 0.686 | **0.823** | 0.710 |
| WMD-BIGRAM + BERT | **0.665** | **0.688** | 0.821 | **0.712** |

Table 5: Comparison on hard and soft alignments.

**Distribution of Scores** In Figure 2, we take a closer look at sentence-level correlation in MT. Results reveal that the lexical metric SENTBLEU can correctly assign lower scores to system translations of low quality, while it struggles in judging system translations of high quality by assigning them lower scores. Our finding agrees with the observations found in Chaganty et al. (2018); Novikova et al. (2017): lexical metrics correlate better with human judgments on texts of low quality than high quality. Peyrard (2019b) further show that lexical metrics cannot be trusted because

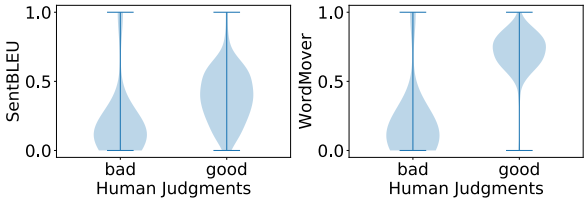


Figure 2: Score distribution in German-to-English pair.

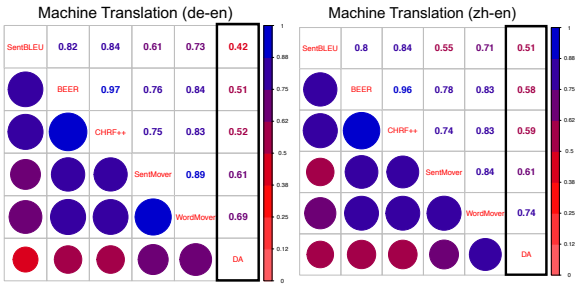


Figure 3: Correlation in similar language (de-en) and distant language (zh-en) pair, where bordered area shows correlations between human assessment and metrics, the rest shows inter-correlations across metrics and DA is direct assessment rated by language experts.

they strongly disagree on high-scoring system outputs. Importantly, we observe that our word mover metric combining BERT can clearly distinguish texts of two polar qualities.

**Correlation Analysis** In Figure 3, we observe existing metrics for MT evaluation attaining medium correlations (0.4-0.5) with human judgments but high inter-correlations between themselves. In contrast, our metrics can attain high correlations (0.6-0.7) with human judgments, performing robust across different language pairs. We believe that our improvements come from clearly distinguishing translations that fall on two extremes.

**Impact of Fine-tuning Tasks** Figure 4 com-

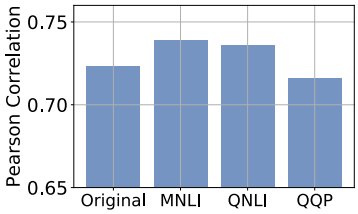


Figure 4: Correlation is averaged over 7 language pairs.

pares Pearson correlations with our word mover metrics combining BERT fine-tuned on three different tasks. We observe that fine-tuning on closely related tasks improves correlations, especially fine-tuning on MNLI leads to an impressive improvement by 1.8 points on average.

### 4.6 Discussions

We showed that our metric combining contextualized embeddings and Earth Mover's Distance outperforms strong unsupervised metrics on 3 out of 4 tasks, i.e., METEOR++ on machine translation by 5.7 points, SPICE on image captioning by 3.0 points, and METEOR on dialogue response generation by 2.2 points. The best correlation we achieved is combining contextualized word embeddings and WMD, which even rivals or exceeds SOTA task-dependent *supervised* metrics across different tasks. Especially in machine translation, our word mover metric pushes correlations in machine translation to 74.3 on average (5.8 points over the SOTA supervised metric and 2.4 points over contemporaneous BERTScore). The major improvements come from contextualized BERT embeddings rather than word2vec and ELMo, and from fine-tuning BERT on large NLI datasets. However, we also observed that soft alignments (MoverScore) marginally outperforms hard alignments (BERTScore). Regarding the effect of $n$-grams in word mover metrics, unigrams slightly outperforms bigrams on average. For the effect of aggregation functions, we suggested effective techniques for layer-wise consolidations, namely $p$-means and routing, both of which are close to the performance of the best layer and on par with each other (see the appendix).

## 5 Conclusion

We investigated new unsupervised evaluation metrics for text generation systems combining contextualized embeddings with Earth Mover's Distance. We experimented with two variants of our metric, sentence mover and word mover. The latter has demonstrated strong generalization ability across four text generation tasks, oftentimes even outperforming supervised metrics. Our metric provides a promising direction towards a holistic metric for text generation and a direction towards more 'human-like' (Eger et al., 2019) evaluation of text generation systems.

In future work, we plan to avoid the need for costly human references in the evaluation of text generation systems, and instead base evaluation scores on source texts and system predictions only, which would allow for 'next-level', unsupervised (in a double sense) and unlimited evaluation (Louis and Nenkova, 2013; Böhm et al., 2019).

## Acknowledgments

We thank the anonymous reviewers for their comments, which greatly improved the final version of the paper. This work has been supported by the German Research Foundation as part of the Research Training Group Adaptive Preparation of Information from Heterogeneous Sources (AIPHES) at the Technische Universität Darmstadt under grant No. GRK 1994/1. Fei Liu is supported in part by NSF grant IIS-1909603.

## References

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2016. SPICE: semantic propositional image caption evaluation. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part V*, pages 382–398.

Florian Böhm, Yang Gao, Christian M. Meyer, Ori Shapira, Ido Dagan, and Iryna Gurevych. 2019. Better rewards yield better summaries: Learning to summarise without references. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, Hong Kong, China.

Ondrej Bojar, Yvette Graham, and Amir Kamran. 2017. Results of the WMT17 metrics shared task. In *Proceedings of the Conference on Machine Translation (WMT)*.

Arun Chaganty, Stephen Mussmann, and Percy Liang. 2018. The price of debiasing automatic metrics in natural language evalaution. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 643–653.

Elizabeth Clark, Asli Celikyilmaz, and Noah A. Smith. 2019. Sentence mover's similarity: Automatic evaluation for multi-sentence texts. In *Proceedings of*

*the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2748–2760, Florence, Italy. Association for Computational Linguistics.

Dorin Comaniciu and Peter Meer. 2002. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (5):603–619.

Yin Cui, Guandao Yang, Andreas Veit, Xun Huang, and Serge Belongie. 2018. Learning to evaluate image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5804–5812.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *arXiv:1810.04805*.

Steffen Eger, Gözde Gül Şahin, Andreas Rücklé, Ji-Ung Lee, Claudia Schulz, Mohsen Mesgar, Krishnkant Swarnkar, Edwin Simpson, and Iryna Gurevych. 2019. Text processing like humans do: Visually attacking and shielding NLP systems. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1634–1647, Minneapolis, Minnesota. Association for Computational Linguistics.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research (JAIR)*.

Max Grusky, Mor Naaman, and Yoav Artzi. 2018. NEWSROOM: A dataset of 1.3 million summaries with diverse extractive strategies. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Yinuo Guo, Chong Ruan, and Junfeng Hu. 2018. Meteor++: Incorporating copy knowledge into machine translation evaluation. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 740–745.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proceedings of Neural Information Processing Systems (NIPS)*.

Eduard Hovy, Chin-Yew Lin, Liang Zhou, and Junichi Fukumoto. 2006. Automated summarization evaluation with basic elements. In *Proceedings of the Fifth Conference on Language Resources and Evaluation (LREC 2006)*, pages 604–611.

Matt J. Kusner, Yu Sun, Nicholas I. Kolkin, and Kilian Q. Weinberger. 2015. From word embeddings to document distances. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Alon Lavie and Abhaya Agarwal. 2007. Meteor: An Automatic Metric for MT Evaluation with High Levels of Correlation with Human Judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07, pages 228–231, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of summaries. In *Proceedings of ACL workshop on Text Summarization Branches Out*, pages 74–81, Barcelona, Spain.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Chia-Wei Liu, Ryan Lowe, Iulian V. Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Liyuan Liu, Xiang Ren, Jingbo Shang, Xiaotao Gu, Jian Peng, and Jiawei Han. 2018. Efficient contextualized representation: Language model pruning for sequence labeling. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Nelson F Liu, Matt Gardner, Yonatan Belinkov, Matthew Peters, and Noah A Smith. 2019. Linguistic knowledge and transferability of contextual representations. *arXiv preprint arXiv:1903.08855*.

Chi-kiu Lo. 2017. MEANT 2.0: Accurate semantic MT evaluation for any output language. In *Proceedings of the Second Conference on Machine Translation, WMT 2017, Copenhagen, Denmark, September 7-8, 2017*, pages 589–597.

Annie Louis and Ani Nenkova. 2013. Automatically assessing machine summary content without a gold standard. *Computational Linguistics*, 39(2):267–300.

Qingsong Ma, Ondrej Bojar, and Yvette Graham. 2018. Results of the WMT18 metrics shared task. In *Proceedings of the Third Conference on Machine Translation (WMT)*.

François Mairesse, Milica Gašić, Filip Jurčíček, Simon Keizer, Blaise Thomson, Kai Yu, and Steve Young. 2010. Phrase-based statistical language generation

using graphical models and active learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1552–1561. Association for Computational Linguistics.

Nitika Mathur, Timothy Baldwin, and Trevor Cohn. 2019. Putting evaluation in context: Contextual embeddings improve machine translation evaluation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2799–2808, Florence, Italy. Association for Computational Linguistics.

Ani Nenkova and Rebecca J. Passonneau. 2004. Evaluating content selection in summarization: The pyramid method. In *Proceedings of the 2004 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 145–152. Association for Computational Linguistics.

Jun-Ping Ng and Viktoria Abrecht. 2015. Better summarization evaluation with word embeddings for rouge. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1925–1930, Lisbon, Portugal. Association for Computational Linguistics.

Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. 2017. Why We Need New Evaluation Metrics for NLG. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2241–2252, Copenhagen, Denmark. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Maxime Peyrard. 2019a. A simple theoretical model of importance for summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1059–1073, Florence, Italy. Association for Computational Linguistics.

Maxime Peyrard. 2019b. Studying summarization evaluation metrics in the appropriate scoring range. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5093–5100, Florence, Italy. Association for Computational Linguistics.

Maxime Peyrard, Teresa Botschen, and Iryna Gurevych. 2017. Learning to score system summaries for better content selection evaluation. In *Proceedings of the Workshop on New Frontiers in Summarization*.

Matt Post. 2018. A call for clarity in reporting bleu scores. In *Proceedings of the Third Conference on Machine Translation (WMT)*.

Ehud Reiter. 2018. A structured review of the validity of BLEU. *Computational Linguistics*, 44(3):393–401.

Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. 2000. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*.

Andreas Rücklé, Steffen Eger, Maxime Peyrard, and Iryna Gurevych. 2018. Concatenated power mean word embeddings as universal cross-lingual sentence representations. *CoRR*, abs/1803.01400.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Hiroki Shimanaka, Tomoyuki Kajiwara, and Mamoru Komachi. 2018. RUSE: Regressor using sentence embeddings for automatic machine translation evaluation. In *Proceedings of the Third Conference on Machine Translation (WMT)*.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R Bowman, Dipanjan Das, et al. 2019. What do you learn from context? probing for sentence structure in contextualized word representations. *arXiv preprint arXiv:1905.06316*.

Stephen Tratz and Eduard H Hovy. 2008. Summarization Evaluation Using Transformed Basic Elements. In *Proceedings of the text analysing conference, (TAC 2008)*.

Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. 2015. CIDEr: Consensus-based Image Description Evaluation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 4566–4575.

Matt P Wand and M Chris Jones. 1994. *Kernel smoothing*. Chapman and Hall/CRC.

Alex Wang, Amapreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745*.

Sam Wiseman, Stuart M. Shieber, and Alexander M. Rush. 2018. Learning neural templates for text generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Suofei Zhang, Wei Zhao, Xiaofu Wu, and Quan Zhou. 2018. Fast dynamic routing based on weighted kernel density estimation. *CoRR*, abs/1805.10807.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with BERT. *CoRR*, abs/1904.09675.

Wei Zhao, Haiyun Peng, Steffen Eger, Erik Cambria, and Min Yang. 2019. Towards scalable and reliable capsule networks for challenging NLP applications. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1549–1559, Florence, Italy. Association for Computational Linguistics.

Wei Zhao, Jianbo Ye, Min Yang, Zeyang Lei, Suofei Zhang, and Zhou Zhao. 2018. Investigating capsule networks with dynamic routing for text classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

# A Supplemental Material

## A.1 Proof of Prop. 1

In this section, we prove Prop. 1 in the paper about viewing BERTScore (precision/recall) as a (non-optimized) Mover Distance.

As a reminder, the WMD formulation is:

$$\text{WMD}(\boldsymbol{x}^n, \boldsymbol{y}^n) := \min_{\boldsymbol{F} \in \mathbb{R}^{|\boldsymbol{x}^n| \times |\boldsymbol{y}^n|}} \sum_{i,j} \boldsymbol{C}_{ij} \cdot \boldsymbol{F}_{ij}$$
$$\text{s.t. } \mathbf{1}^\intercal \boldsymbol{F}^\intercal \mathbf{1} = 1, \quad \mathbf{1}^\intercal \boldsymbol{F} \mathbf{1} = 1.$$

where $\boldsymbol{F}^\intercal \mathbf{1} = \boldsymbol{f}_x^n$ and $\boldsymbol{F}\mathbf{1} = \boldsymbol{f}_y^n$. Here, $\boldsymbol{f}_x^n$ and $\boldsymbol{f}_y^n$ denote vectors of weights for each $n$-gram of $\boldsymbol{x}^n$ and $\boldsymbol{y}^n$.

BERTScore is defined as:

$$R_{\text{BERT}} = \frac{\sum_{y_i^1 \in \boldsymbol{y}^1} \text{idf}(y_i^1) \max_{x_j^1 \in \boldsymbol{x}^1} E(x_j^1)^\intercal E(y_i^1)}{\sum_{y_i^1 \in \boldsymbol{y}^1} \text{idf}(y_i^1)}$$
$$P_{\text{BERT}} = \frac{\sum_{x_j^1 \in \boldsymbol{x}^1} \text{idf}(x_j^1) \max_{y_i^1 \in \boldsymbol{y}^1} E(y_i^1)^\intercal E(x_j^1)}{\sum_{x_j^1 \in \boldsymbol{x}^1} \text{idf}(x_j^1)}$$
$$F_{\text{BERT}} = 2\frac{P_{\text{BERT}} \cdot R_{\text{BERT}}}{P_{\text{BERT}} + R_{\text{BERT}}} \quad .$$

Then, $R_{\text{BERT}}$ can be formulated in a "quasi" WMD form:

$$R_{\text{BERT}}(\boldsymbol{x}^1, \boldsymbol{y}^1) := \sum_{i,j} \boldsymbol{C}_{ij} \cdot \boldsymbol{F}_{ij}$$
$$\boldsymbol{F}_{ij} = \begin{cases} \frac{1}{M} & \text{if } x_j = \arg\max_{\hat{x}_j^1 \in \boldsymbol{x}^1} E(y_i^1)^\intercal E(\hat{x}_j^1) \\ 0 & \text{otherwise} \end{cases}$$
$$\boldsymbol{C}_{ij} = \begin{cases} \frac{M}{Z} \text{idf}(y_i^1) E(x_j^1)^\intercal E(y_i^1) & \text{if } x_j = \arg\max_{\hat{x}_j^1 \in \boldsymbol{x}^1} E(y_i^1)^\intercal E(\hat{x}_j^1) \\ 0 & \text{otherwise} \end{cases}$$

where $Z = \sum_{y_i^1 \in \boldsymbol{y}^1} \text{idf}(y_i^1)$ and $M$ is the size of $n$-grams in $\boldsymbol{x}^1$. Similarly, we can have $P_{\text{BERT}}$ in a quasi WMD form (omitted). Then, $F_{\text{BERT}}$ can be formulated as harmonic-mean of two WMD forms of $P_{\text{BERT}}$ and $R_{\text{BERT}}$.

## A.2 Routing

In this section, we study the aggregation function $\phi$ with a routing scheme, which has achieved good results in other NLP tasks (Zhao et al., 2018, 2019). Specifically, we introduce a nonparametric clustering with Kernel Density Estimation (KDE) for routing since KDE bridges a family of kernel functions with underlying empirical distributions, which often leads to computational efficiency (Zhang et al., 2018), defined as:

$$\min_{\boldsymbol{v}, \gamma} f(\boldsymbol{z}) = \sum_{i=1}^{L} \sum_{j=1}^{T} \gamma_{ij} k(d(\boldsymbol{v}_j - \boldsymbol{z}_{i,j}))$$
$$\textit{s.t.} \quad \forall i, j : \gamma_{ij} > 0, \sum_{j=1}^{L} \gamma_{ij} = 1.$$

where $d(\cdot)$ is a distance function, $\gamma_{ij}$ denotes the underlying closeness between the aggregated vector $\boldsymbol{v}_j$ and vector $\boldsymbol{z}_i$ in the $i$-th layer, and $k$ is a kernel function. Some instantiations of $k(\cdot)$ (Wand and Jones, 1994) are:

$$Gaussian: k(x) \triangleq \exp\left(-\frac{x}{2}\right), \quad Epanechnikov: k(x) \triangleq \begin{cases} 1-x & x \in [0,1) \\ 0 & x \geq 1. \end{cases}$$

One typical solution for KDE clustering to minimize $f(\boldsymbol{z})$ is taking Mean Shift (Comaniciu and Meer, 2002), defined as:

$$\nabla f(\boldsymbol{z}) = \sum_{i,j} \gamma_{ij} k'(d(\boldsymbol{v}_j, \boldsymbol{z}_{i,j})) \frac{\partial d(\boldsymbol{v}_j, \boldsymbol{z}_{i,j})}{\partial \boldsymbol{v}}$$

Firstly, $\boldsymbol{v}_j^{\tau+1}$ can be updated while $\gamma_{ij}^{\tau+1}$ is fixed:

$$\boldsymbol{v}_j^{\tau+1} = \frac{\sum_i \gamma_{ij}^{\tau} k'(d(\boldsymbol{v}_j^{\tau}, \boldsymbol{z}_{i,j}))\boldsymbol{z}_{i,j}}{\sum_{i,j} k'(d(\boldsymbol{v}_j^{\tau}, \boldsymbol{z}_{i,j}))}$$

Intuitively, $\boldsymbol{v}_j$ can be explained as a final aggregated vector from $L$ contextualized layers. Then, we adopt SGD to update $\gamma_{ij}^{\tau+1}$:

$$\gamma_{ij}^{\tau+1} = \gamma_{ij}^{\tau} + \alpha \cdot k(d(\boldsymbol{v}_j^{\tau}, \boldsymbol{z}_{i,j}))$$

where $\alpha$ is a hyperparameter to control step size. The routing process is summarized in Algorithm 1.

**Algorithm 1** Aggregation by Routing

**procedure** ROUTING($\boldsymbol{z}_{ij}, \ell$)
Initialize $\forall i, j : \gamma_{ij} = 0$
**while** true **do**
  **foreach** representation $i$ and $j$ in layer $\ell$ and $\ell+1$ **do** $\gamma_{ij} \leftarrow softmax(\gamma_{ij})$
  **foreach** representation $j$ in layer $\ell+1$ **do**
    $\boldsymbol{v}_j \leftarrow \sum_i \gamma_{ij} k'(\boldsymbol{v}_j, \boldsymbol{z}_i)\boldsymbol{z}_i / \sum_i k'(\boldsymbol{v}_i, \boldsymbol{z}_i)$
  **foreach** representation $i$ and $j$ in layer $\ell$ and $\ell+1$ **do** $\gamma_{ij} \leftarrow \gamma_{ij} + \alpha \cdot k(\boldsymbol{v}_j, \boldsymbol{z}_i)$
  loss $\leftarrow \log(\sum_{\mathrm{i,j}} \gamma_{\mathrm{ij}} \mathrm{k}(\boldsymbol{v}_{\mathrm{j}}, \boldsymbol{z}_{\mathrm{i}}))$
  **if** $|\text{loss} - \text{preloss}| < \epsilon$ **then**
    **break**
  **else**
    preloss $\leftarrow$ loss
**return** $v_j$

**Best Layer and Layer-wise Consolidation** Table 6 compares our word mover based metric combining BERT representations on different layers with stronger BERT representations consolidated from these layers (using $p$-means and routing). We often see that which layer has best performance is task-dependent, and our word mover based metrics (WMD) with $p$-means or routing schema come close to the oracle performance obtained from the best layers.

**Experiments** Table 7, 8 and 9 show correlations between metrics (all baseline metrics and word mover based metrics) and human judgments on machine translation, text summarization and dialogue response generation, respectively. We find that word mover based metrics combining BERT fine-tuned on MNLI have highest correlations with humans, outperforming all of the unsupervised metrics and even supervised metrics like RUSE and $S^3_{full}$. Routing and $p$-means perform roughly equally well.

| Metrics | Direct Assessment cs-en | de-en | fi-en | lv-en | ru-en | tr-en | zh-en |
|---|---|---|---|---|---|---|---|
| WMD-1 + BERT + LAYER 8 | .6361 | .6755 | .8134 | .7033 | .7273 | .7233 | .7175 |
| WMD-1 + BERT + LAYER 9 | .6510 | .6865 | .8240 | .7107 | .7291 | **.7357** | **.7195** |
| WMD-1 + BERT + LAYER 10 | .6605 | **.6948** | **.8231** | .7158 | **.7363** | .7317 | .7168 |
| WMD-1 + BERT + LAYER 11 | **.6695** | .6845 | .8192 | .7048 | .7315 | .7276 | .7058 |
| WMD-1 + BERT + LAYER 12 | .6677 | .6825 | .8194 | **.7188** | .7326 | .7291 | .7064 |
| WMD-1 + BERT + ROUTING | .6618 | .6897 | .8225 | .7122 | .7334 | .7301 | .7182 |
| WMD-1 + BERT + PMEANS | .6623 | .6873 | .8234 | .7139 | .7350 | .7339 | .7192 |

Table 6: Absolute Pearson correlations with segment-level human judgments on WMT17 to-English translations.

| Setting | Metrics | Direct Assessment cs-en | de-en | fi-en | lv-en | ru-en | tr-en | zh-en | Average |
|---|---|---|---|---|---|---|---|---|---|
| BASELINES | BLEND | 0.594 | 0.571 | 0.733 | 0.594 | 0.622 | 0.671 | 0.661 | 0.635 |
| | RUSE | 0.624 | 0.644 | 0.750 | 0.697 | 0.673 | 0.716 | 0.691 | 0.685 |
| | SENTBLEU | 0.435 | 0.432 | 0.571 | 0.393 | 0.484 | 0.538 | 0.512 | 0.481 |
| | CHRF++ | 0.523 | 0.534 | 0.678 | 0.520 | 0.588 | 0.614 | 0.593 | 0.579 |
| | METEOR++ | 0.552 | 0.538 | 0.720 | 0.563 | 0.627 | 0.626 | 0.646 | 0.610 |
| | BERTSCORE-F1 | 0.670 | 0.686 | 0.820 | 0.710 | 0.729 | 0.714 | 0.704 | 0.719 |
| WORD-MOVER | WMD-1 + W2V | 0.392 | 0.463 | 0.558 | 0.463 | 0.456 | 0.485 | 0.481 | 0.471 |
| | WMD-1 + BERT + ROUTING | 0.658 | 0.689 | 0.823 | 0.712 | 0.733 | 0.730 | 0.718 | 0.723 |
| | WMD-1 + BERT + MNLI + ROUTING | 0.665 | 0.705 | 0.834 | 0.744 | 0.735 | 0.752 | 0.736 | 0.739 |
| | WMD-2 + BERT + MNLI + ROUTING | 0.676 | 0.706 | 0.831 | 0.743 | 0.734 | 0.755 | 0.732 | 0.740 |
| | WMD-1 + BERT + PMEANS | 0.662 | 0.687 | 0.823 | 0.714 | 0.735 | 0.734 | 0.719 | 0.725 |
| | WMD-1 + BERT + MNLI + PMEANS | 0.670 | 0.708 | **0.835** | **0.746** | **0.738** | 0.762 | **0.744** | **0.743** |
| | WMD-2 + BERT + MNLI + PMEANS | **0.679** | **0.710** | 0.832 | 0.745 | 0.736 | **0.763** | 0.740 | **0.743** |

Table 7: Absolute Pearson correlations with segment-level human judgments on WMT17 to-English translations.

| Setting | Metrics | TAC-2008 Responsiveness $r$ | Responsiveness $\rho$ | Pyramid $r$ | Pyramid $\rho$ | TAC-2009 Responsiveness $r$ | Responsiveness $\rho$ | Pyramid $r$ | Pyramid $\rho$ |
|---|---|---|---|---|---|---|---|---|---|
| BASELINES | $S^3_{full}$ | 0.696 | 0.558 | 0.753 | 0.652 | 0.731 | 0.552 | 0.838 | 0.724 |
| | $S^3_{best}$ | 0.715 | 0.595 | 0.754 | 0.652 | 0.738 | **0.595** | **0.842** | **0.731** |
| | TF∗IDF-1 | 0.176 | 0.224 | 0.183 | 0.237 | 0.187 | 0.222 | 0.242 | 0.284 |
| | TF∗IDF-2 | 0.047 | 0.154 | 0.049 | 0.182 | 0.047 | 0.167 | 0.097 | 0.233 |
| | ROUGE-1 | 0.703 | 0.578 | 0.747 | 0.632 | 0.704 | 0.565 | 0.808 | 0.692 |
| | ROUGE-2 | 0.695 | 0.572 | 0.718 | 0.635 | 0.727 | 0.583 | 0.803 | 0.694 |
| | ROUGE-1-WE | 0.571 | 0.450 | 0.579 | 0.458 | 0.586 | 0.437 | 0.653 | 0.516 |
| | ROUGE-2-WE | 0.566 | 0.397 | 0.556 | 0.388 | 0.607 | 0.413 | 0.671 | 0.481 |
| | ROUGE-L | 0.681 | 0.520 | 0.702 | 0.568 | 0.730 | 0.563 | 0.779 | 0.652 |
| | FRAME-1 | 0.658 | 0.508 | 0.686 | 0.529 | 0.678 | 0.527 | 0.762 | 0.628 |
| | FRAME-2 | 0.676 | 0.519 | 0.691 | 0.556 | 0.715 | 0.555 | 0.781 | 0.648 |
| | BERTSCORE-F1 | 0.724 | 0.594 | 0.750 | 0.649 | 0.739 | 0.580 | 0.823 | 0.703 |
| WORD-MOVER | WMD-1 + W2V | 0.669 | 0.559 | 0.665 | 0.611 | 0.698 | 0.520 | 0.740 | 0.647 |
| | WMD-1 + BERT + ROUTING | 0.729 | 0.601 | 0.763 | 0.675 | 0.740 | 0.580 | 0.831 | 0.700 |
| | WMD-1 + BERT + MNLI + ROUTING | 0.734 | **0.609** | **0.768** | **0.686** | 0.747 | 0.589 | 0.837 | 0.711 |
| | WMD-2 + BERT + MNLI + ROUTING | 0.731 | 0.593 | 0.755 | 0.666 | 0.753 | 0.583 | 0.827 | 0.698 |
| | WMD-1 + BERT + PMEANS | 0.729 | 0.595 | 0.755 | 0.660 | 0.742 | 0.581 | 0.825 | 0.690 |
| | WMD-1 + BERT + MNLI + PMEANS | **0.736** | 0.604 | 0.760 | 0.672 | **0.754** | 0.594 | 0.831 | 0.701 |
| | WMD-2 + BERT + MNLI + PMEANS | 0.734 | 0.601 | 0.752 | 0.663 | 0.753 | 0.586 | 0.825 | 0.694 |

Table 8: Correlation of automatic metrics with summary-level human judgments for TAC-2008 and TAC-2009.

| Setting | Metrics | BAGEL | | | SFHOTEL | | |
|---|---|---|---|---|---|---|---|
| | | **Inf** | **Nat** | **Qual** | **Inf** | **Nat** | **Qual** |
| BASELINES | BLEU-1 | 0.225 | 0.141 | 0.113 | 0.107 | 0.175 | 0.069 |
| | BLEU-2 | 0.211 | 0.152 | 0.115 | 0.097 | 0.174 | 0.071 |
| | BLEU-3 | 0.191 | 0.150 | 0.109 | 0.089 | 0.161 | 0.070 |
| | BLEU-4 | 0.175 | 0.141 | 0.101 | 0.084 | 0.104 | 0.056 |
| | ROUGE-L | 0.202 | 0.134 | 0.111 | 0.092 | 0.147 | 0.062 |
| | NIST | 0.207 | 0.089 | 0.056 | 0.072 | 0.125 | 0.061 |
| | CIDER | 0.205 | 0.162 | 0.119 | 0.095 | 0.155 | 0.052 |
| | METEOR | 0.251 | 0.127 | 0.116 | 0.111 | 0.148 | 0.082 |
| | BERTSCORE-F1 | 0.267 | 0.210 | **0.178** | 0.163 | 0.193 | 0.118 |
| WORD-MOVER | WMD-1 + W2V | 0.222 | 0.079 | 0.123 | 0.074 | 0.095 | 0.021 |
| | WMD-1 + BERT + ROUTING | 0.294 | 0.209 | 0.156 | 0.208 | 0.256 | 0.178 |
| | WMD-1 + BERT + MNLI + ROUTING | 0.278 | 0.180 | 0.144 | **0.211** | 0.252 | 0.175 |
| | WMD-2 + BERT + MNLI + ROUTING | 0.279 | 0.182 | 0.147 | 0.204 | 0.252 | 0.172 |
| | WMD-1 + BERT + PMEANS | **0.298** | **0.212** | 0.163 | 0.203 | 0.261 | 0.182 |
| | WMD-1 + BERT + MNLI + PMEANS | 0.285 | 0.195 | 0.158 | 0.207 | **0.270** | **0.183** |
| | WMD-2 + BERT + MNLI + PMEANS | 0.284 | 0.194 | 0.156 | 0.204 | **0.270** | 0.182 |

Table 9: Spearman correlation with utterance-level human judgments for BAGEL and SFHOTEL datasets.

# Select and Attend: Towards Controllable Content Selection in Text Generation

**Xiaoyu Shen**[1,2]*, **Jun Suzuki**[3,4], **Kentaro Inui**[3,4], **Hui Su**[5]
**Dietrich Klakow**[1] **and Satoshi Sekine**[4]
[1]Spoken Language Systems (LSV), Saarland University, Germany
[2]Max Planck Institute for Informatics, Saarland Informatics Campus, Germany
[3]Tohoku University [4]RIKEN AIP, Japan
[5]Pattern Recognition Center, Wechat AI, Tencent Inc, China

## Abstract

Many text generation tasks naturally contain two steps: content selection and surface realization. Current neural encoder-decoder models conflate both steps into a black-box architecture. As a result, the content to be described in the text cannot be explicitly controlled. This paper tackles this problem by decoupling content selection from the decoder. The decoupled content selection is human interpretable, whose value can be manually manipulated to control the content of generated text. The model can be trained end-to-end without human annotations by maximizing a lower bound of the marginal likelihood. We further propose an effective way to trade-off between performance and controllability with a single adjustable hyperparameter. In both data-to-text and headline generation tasks, our model achieves promising results, paving the way for controllable content selection in text generation.[1]

## 1 Introduction

Many text generation tasks, e.g., data-to-text, summarization and image captioning, can be naturally divided into two steps: content selection and surface realization. The generations are supposed to have two levels of diversity: (1) content-level diversity reflecting multiple possibilities of content selection (what to say) and (2) surface-level diversity reflecting the linguistic variations of verbalizing the selected contents (how to say) (Reiter and Dale, 2000; Nema et al., 2017). Recent neural network models handle these tasks with the encoder-decoder (Enc-Dec) framework (Sutskever et al., 2014; Bahdanau et al., 2015), which simultaneously performs selecting and verbalizing in a black-box way. Therefore, both levels of diversity are entangled within the generation. This entanglement, however, sacrifices the controllability and interpretability, making it diffifcult to specify the content to be conveyed in the generated text (Qin et al., 2018; Wiseman et al., 2018).

| |
|---|
| **Source Sentence:** The sri lankan government on Wednesday announced the closure of government schools with immediate effect as a military campaign against tamil separatists escalated in the north of the country. |
| **Selected :** sri lankan, closure, schools<br>**Text:** sri lanka closes schools . |
| **Selected :** sri lankan, Wednesday, closure, schools<br>**Text:** sri lanka closes schools on Wednesday. |
| **Selected :** sri lankan, closure, schools, military campaign<br>**Text:** sri lanka shuts down schools amid war fears. |
| **Selected :** sri lankan, announced, closure, schools<br>**Text:** sri lanka declares closure of schools. |

Table 1: Headline generation examples from our model. We can generate text describing various contents by sampling different content selections. The selected source word and its corresponding realizations in the text are highlighted with the same color.

With this in mind, this paper proposes decoupling content selection from the Enc-Dec framework to allow finer-grained control over the generation. Table 1 shows an example. We can easily modify the content selection to generate text with various focuses, or sample multiple paraphrases by fixing the content selection.

Though there has been much work dealing with content selection for the Enc-Dec, none of them is able to address the above concerns properly. Current methods can be categorized into the following three classes and have different limits:

1. **Bottom-up**: Train a separate content selector to constrain the attention to source tokens (Gehrmann et al., 2018), but the separate training of selector/generator might lead to

*Work mostly done while at RIKEN AIP. Correspondence to xshen@mpi-inf.mpg.de

[1]The source code is available on https://github.com/chin-gyou/controllable-selection



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 579–590, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

discrepancy when integrating them together.

2. **Soft-select**: Learn a soft mask to filter useless information (Mei et al., 2016; Zhou et al., 2017). However, the mask is *deterministic* without any probabilistic variations, making it hard to model the content-level diversity.

3. **Reinforce-select**: Train the selector with reinforcement learning (Chen and Bansal, 2018), which has high training variance and low diversity on content selection.

In this paper, we treat the content selection as latent variables and train with amortized variational inference (Kingma and Welling, 2014; Mnih and Gregor, 2014). This provides a lower training variance than Reinforce-select. The selector and generator are co-trained within the same objective, the generations are thus more faithful to the selected contents than Bottom-up methods. Our model is task-agnostic, end-to-end trainable and can be seamlessly inserted into any encoder-decoder architecture. On both the data-to-text and headline generation task, we show our model outperforms others regarding content-level diversity and controllability while maintaining comparable performance. The performance/controllability trade-off can be effectively adjusted by adjusting a single hyperparameter in the training stage, which constrains an upper bound of the conditional mutual information (CMI) between the selector and generated text (Alemi et al., 2018; Zhao et al., 2018). A higher CMI leads to stronger controllability with a bit more risk of text disfluency.

In summary, our contributions are **(1)** systematically studying the problem of controllable content selection for Enc-Dec text generation, **(2)** proposing a task-agnostic training framework achieving promising results and **(3)** introducing an effective way to achieve the trade-off between performance and controllability.

## 2 Background and Notation

Let $X, Y$ denote a source-target pair. $X$ is a sequence of $x_1, x_2, \ldots, x_n$ and can be either some structured data or unstructured text/image depending on the task. $Y$ corresponds to $y_1, y_2, \ldots, y_m$ which is a text description of $X$. The goal of text generation is to learn a distribution $p(Y|X)$ to automatically generate proper text.

The Enc-Dec architecture handles this task with an encode-attend-decode process (Bahdanau et al., 2015; Xu et al., 2015). The encoder first encodes each $x_i$ into a vector $h_i$. At each time step, the decoder pays attentions to some source embeddings and outputs the probability of the next token by $p(y_t|y_{1:t-1}, C_t)$. $C_t$ is a weighted average of source embeddings:

$$C_t = \sum_i \alpha_{t,i} h_i$$
$$\alpha_{t,i} = \frac{e^{f(h_i, d_t)}}{\sum_j e^{f(h_j, d_t)}} \quad (1)$$

$d_t$ is the hidden state of the decoder at time step $t$. $f$ is a score function to compute the similarity between $h_i$ and $d_t$ (Luong et al., 2015).

## 3 Content Selection

Our goal is to decouple the content selection from the decoder by introducing an extra content selector. We hope the content-level diversity can be fully captured by the content selector for a more interpretable and controllable generation process. Following Gehrmann et al. (2018); Yu et al. (2018), we define content selection as a sequence labeling task. Let $\beta_1, \beta_2, \ldots, \beta_n$ denote a sequence of binary selection masks. $\beta_i = 1$ if $h_i$ is selected and 0 otherwise. $\beta_i$ is assumed to be independent from each other and is sampled from a bernoulli distribution $\mathbf{B}(\gamma_i)$[2]. $\gamma_i$ is the bernoulli parameter, which we estimate using a two-layer feedforward network on top of the source encoder. Text are generated by first sampling $\beta$ from $\mathbf{B}(\gamma)$ to decide which content to cover, then decode with the conditional distribution $p_\theta(Y|X, \beta)$. The text is expected to faithfully convey all selected contents and drop unselected ones. Fig. 1 depicts this generation process. Note that the selection is based on the token-level *context-aware* embeddings $h$ and will maintain information from the surrounding contexts. It encourages the decoder to stay faithful to the original information instead of simply fabricating random sentences by connecting the selected tokens. For each source-target pair, the ground-truth selection mask is unknown, so training is challenging. In the following session, we discuss several training possibilities and introduce the proposed model in detail.

[2]Devlin et al. (2019) have shown that excellent performance can be obtained by assuming such conditionally independence given a sufficiently expressive representation of $x$, though modelling a richer inter-label dependency is for sure beneficial (Lei et al., 2016; Nallapati et al., 2017).

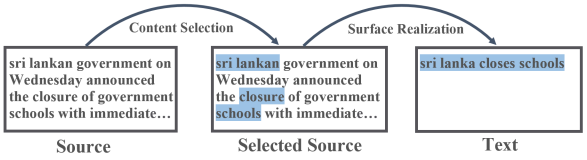


Figure 1: Model will select contents based on $\mathbf{B}(\gamma)$, then decode with $p_\theta(Y|X,\beta)$. Source-text pairs are available for training, but the ground-truth content selection for each pair is unknown.

### 3.1 Bottom-up

The most intuitive way is training the content selector to target some heuristically extracted contents. For example, we can train the selector to select overlapped words between the source and target (Gehrmann et al., 2018), sentences with higher tf-idf scores (Li et al., 2018) or identified image objects that appear in the caption (Wang et al., 2017). A standard encoder-decoder model is independently trained. In the testing stage, the prediction of the content selector is used to hard-mask the attention vector to guide the text generation in a bottom-up way. Though easy to train, Bottom-up generation has the following two problems: **(1)** The heuristically extracted contents might be coarse and cannot reflect the variety of human languages and **(2)** The selector and decoder are independently trained towards different objectives thus might not adapt to each other well.

***β* as Latent Variable:** Another way is to treat $\beta$ as a latent variable and co-train selector and generator by maximizing the marginal data likelihood. By doing so, the selector has the potential to automatically explore optimal selecting strategies best fit for the corresponding generator component.

With this in mind. We design $p_\theta(Y|X,\beta)$ by changing the original decoder in the following way: (1) We initialize hidden states of the decoder from a mean pooling over selected contents to inform the decoder which contents to cover and (2) Unselected contents will be prohibited from being attended to:

$$d_0 = \text{MLP}\left(\frac{1}{n}\left(\sum_i^n \beta_i h_i\right)\right)$$
$$\alpha_{t,i} = \frac{e^{f(h_i,d_t)}\beta_i}{\sum_j e^{f(h_j,d_t)}\beta_j} \quad (2)$$

$d_0$ is the initial decoder hidden state and MLP denotes multi-layer-perceptron.

Since computing the exact marginal likelihood $\log \mathbb{E}_{\beta\sim\mathbf{B}(\gamma)} p_\theta(Y|X,\beta)$ requires enumerating over all possible combinations of $\beta$ (complexity $\mathbf{O}(2^n)$), we need some way to efficiently estimate the likelihood.

### 3.2 Soft-Select

Soft-select falls back on a *deterministic* network to output the likelihood function's first-order Taylor series approximation expanded at $\mathbb{E}_{\beta\sim\mathbf{B}(\gamma)}\beta$:

$$\begin{aligned}&\log \mathbb{E}_{\beta\sim\mathbf{B}(\gamma)} p_\theta(Y|X,\beta)\\ \approx &\log[p_\theta(Y|X,\gamma) + \mathbb{E}_{\beta\sim\mathbf{B}(\gamma)}(\beta-\gamma)p'_\theta(Y|X,\gamma)]\\ = &\log p_\theta(Y|X,\gamma)\end{aligned}$$

By moving the expectation into the decoding function, we can deterministically compute the likelihood by setting $\beta_i = \gamma_i$, reducing complexity to $\mathbf{O}(1)$. Each attention weight will first be "soft-masked" by $\gamma$ before being passed to the decoder. soft-select is fully differentiable and can be easily trained by gradient descent. However, this soft-approximation is normally inaccurate, especially when $\mathbf{B}(\gamma)$ has a high entropy, which is common in one-to-many text generation tasks. The gap between $\log \mathbb{E}_{\beta\sim\mathbf{B}(\gamma)} p_\theta(Y|X,\beta)$ and $\log p_\theta(Y|X,\mathbb{E}_{\beta\sim\mathbf{B}(\gamma)})$ will be large (Ma et al., 2017; Deng et al., 2018). In practice, this would lead to unrealistic generations when sampling $\beta$ from the deterministically trained distribution.

### 3.3 Reinforce-Select

Reinforce-select (RS) (Ling and Rush, 2017; Chen and Bansal, 2018) utilizes reinforcement learning to approximate the marginal likelihood. Specifically, it is trained to maximize a lower bound of the likelihood by applying the Jensen inequalily:

$$\log \mathbb{E}_{\beta\sim\mathbf{B}(\gamma)} p_\theta(Y|X,\beta) \geq \mathbb{E}_{\beta\sim\mathbf{B}(\gamma)} \log p_\theta(Y|X,\beta)$$

The gradient to $\gamma$ is approximated with Monte-Carlo sampling by applying the REINFORCE algorithm (Williams, 1992; Glynn, 1990). To speed up convergence, we pre-train the selector by some distant supervision, which is a common practice in reinforcement learning. REINFORCE is unbiased but has a high variance. Many research have proposed sophisticated techniques for variance reduction (Mnih and Gregor, 2014; Tucker et al., 2017; Grathwohl et al., 2018). In text generation, the high-variance problem is aggravated because there exists multiple valid selections. Accurately estimating the likelihood becomes difficult. Another

issue is its tendency to avoid stochasticity (Raiko et al., 2015), which we will show in Sec 5.2 that it results in low content-level diversity.

### 3.4 Variational Reinforce-Select

We propose Variational Reinforce-Select (VRS) which applies variational inference (Kingma and Welling, 2014) for variance reduction. Instead of directly integrating over $\mathbf{B}(\gamma)$, it imposes a proposal distribution $q_\phi$ for importance sampling. The marginal likelihood is lower bounded by:

$$\begin{aligned}
&\log \mathbb{E}_{\beta\sim\mathbf{B}(\gamma)} p_\theta(Y|X,\beta) \\
&= \log \mathbb{E}_{\beta\sim q_\phi} \frac{p_\theta(Y,\beta|X)}{q_\phi(\beta)} \\
&\geq \mathbb{E}_{\beta\sim q_\phi} \log \frac{p_\theta(Y,\beta|X)}{q_\phi(\beta)} \\
&= \mathbb{E}_{\beta\sim q_\phi} \log p_\theta(Y|X,\beta) - KL(q_\phi||\mathbf{B}(\gamma))
\end{aligned} \tag{3}$$

By choosing a proper $q_\phi$, the bound will be improved and the variance can be largely reduced compared with REINFORCE. If $q_\phi$ equals the posterior distribution $p_\theta(\beta|X,Y)$, the bound is tight and the variance would be zero (Mnih and Rezende, 2016). We define $q_\phi(\beta|X,Y)$ as a mean-field distribution parameterized by a set of global parameters $\phi$ to approach the true posterior distribution. $\phi$, $\theta$ and $\gamma$ are simultaneously trained by minimizing the last tine of Eq. 3. $q_\phi(\beta|X,Y)$ also allows us to further perform posterior inference: Given an arbitrary text $Y$ for a source $X$, we can infer which source contents are included in $Y$ (An example is given in Appendix C).

In Eq.3, the KL divergence term can be computed analytically. As for the independence assumption, it can be summed over each individual $\beta_i$. The likelihood term is differentiable to $\theta$ but not to $\phi$, we estimate the gradient to $\phi$ in Eq 3 by applying the REINFORCE estimator:

$$\nabla_\phi \mathbb{E}_{\beta\sim q_\phi} \log p_\theta(Y|X,\beta) =$$
$$\mathbb{E}_{\beta\sim q_\phi} \nabla_\phi \log q_\phi(\beta|X,Y)(\log p_\theta(Y|X,\beta) - B)$$

$B$ is the control variate (Williams, 1992). The optimal $B$ would be (Weaver and Tao, 2001):

$$B^* = \mathbb{E}_{\beta\sim q_\phi} \log p_\theta(Y|X,\beta)$$

which we set as a soft-select approximation:

$$B = \log p_\theta(Y|X, \mathbb{E}_{\beta\sim q_\phi}\beta)$$

We estimate Eq. 3.4 with a single sample from $q_\phi$ for efficiency. Though multiple-sample could potentially further tighten the bound and reduce the variance (Burda et al., 2016; Lawson et al., 2018; Tucker et al., 2019), it brings significant computational overhead, especially in text generation tasks where the whole sentence needs to be decoded.

### 3.5 Degree of Controllability

In practice, when treating content selection as latent variables, the model tends to end up with a trivial solution of always selecting all source tokens (Shen et al., 2018a; Ke et al., 2018). This behavior is understandable since Eq. 2 strictly masks unselected tokens. Wrongly unselecting one token will largely deteriorate the likelihood. Under the maximum likelihood (MLE) objective, this high risk pushes the selector to take a conservative strategy of always keeping all tokens, then the whole model degenerates to the standard Enc-Dec and the selection mask loses effects on the generation. Usually people apply a penalty term to the selecting ratio when optimizing the likelihood:

$$\mathcal{L} + \lambda\big|(\bar{\gamma} - \alpha)\big| \tag{4}$$

$\mathcal{L}$ is the MLE loss function, $\bar{\gamma}$ is the mean of $\gamma$ and $\alpha$ is the target selecting ratio. This forces the selector to select the most important $\alpha$ tokens for each source input instead of keeping all of them.

In our VRS model, we can easily adjust the degree of controllability by limiting an upper bound of the conditional mutual information (CMI) $I(\beta, Y|X)$ (Zhao et al., 2018). Specifically, we can change our objective into:

$$\begin{aligned}
&\max_{\phi,\theta,\gamma} \mathbb{E}_{\beta\sim q_\phi} \log p_\theta(Y|X,\beta) \\
&\quad -\lambda\big|KL(q_\phi||\mathbf{B}(\gamma)) - \epsilon)\big|
\end{aligned} \tag{5}$$

$\lambda$ is a fixed lagrangian multiplier. Eq. 5 can be proved equal to maximum likelihood with the constraint $I(\beta, Y|X) = \epsilon$ given proper $\lambda$ (Alemi et al., 2018). A higher $\epsilon$ indicates $\beta$ has more influences to $Y$ (higher controllability) while always safely selecting all tokens will lead $I(\beta, Y|X) = 0$.[3] It is preferred over Eq. 4 because (a) CMI directly considers the dependency between the selection and *multiple*-possible text while limiting the ratio aims at finding the *single* most salient parts for each source. (b) Unlike CMI, limiting the ratio is coarse. It considers only the total selected size and ignores its internal distribution.

[3] We also tried adding a coverage constraint to ensure the decoder covers all the selected tokens (Wen et al., 2015; Wang et al., 2019), but we find it brings no tangible help since a higher CMI can already discourage including redundant tokens into the selection.

**Algorithm 1** Variational Reinforce-Select (VRS)

**Parameters:** $\theta, \phi, \gamma$
$pretrain \leftarrow$ TRUE
**repeat**
  Sample X,Y from the corpus;
  Encode X into $(h_1, h_2, \ldots, h_n)$;
  **if** $pretrain$ **then**
    Update $\phi$ with distant supervision;
    Update $\theta, \gamma$ by $\nabla_{\theta,\gamma}$Eq. 3;
  **else**
    Update $\theta, \gamma, \phi$ by $\nabla_{\theta,\gamma,\phi}$Eq. 5;
  **end if**
  $pretrain \leftarrow$ FALSE if Eq. 3 degrades
**until** convergence and $pretrain$ is False

In practice, we can set $\epsilon$ to adjust the degree of controllability we want. Later we will show it leads to a trade-off with performance. The final algorithm is detailed in Algorithm 1. To keep fairness, we trian RS and VRS with the same control variate and pre-training strategy.[4]

## 4 Related Work

Most content selection models train the selector with heuristic rules (Hsu et al., 2018; Li et al., 2018; Yu et al., 2018; Gehrmann et al., 2018; Yao et al., 2019; Moryossef et al., 2019), which fail to fully capture the relation between selection and generation. Mei et al. (2016); Zhou et al. (2017); Lin et al. (2018); Li et al. (2018) "soft-select" word or sentence embeddings based on a gating function. The output score from the gate is a deterministic vector without any probabilistic variations, so controlling the selection to generate diverse text is impossible. Very few works explicitly define a bernoulli distribution for the selector, then train with the REINFORCE algorithm (Ling and Rush, 2017; Chen and Bansal, 2018), but the selection targets at a high recall regardless of the low precision, so the controllability over generated text is weak. Fan et al. (2018) control the generation by manually concatenating entity embeddings, while our model is much more flexible by explicitly defining the selection probability over all source tokens.

Our work is closely related with learning discrete representations with variational inference (Wen et al., 2017; van den Oord et al., 2017; Kaiser et al., 2018; Lawson et al., 2018), where we treat content selection as the latent representation. Limiting the KL-term is a common technique to deal with the "posterior collapse" problem (Kingma et al., 2016; Yang et al., 2017; Shen et al., 2018b). We adopt a similar approach and use it to further control the selecting strategy.

[4]The only extra parameter of VRS is $\phi$ which is a simple MLP structure. The actual training complexity is similar to RS because they both use the REINFORCE algorithm for gradient estimation.

## 5 Experiments

For the experiments, we focus on comparing (1) Bottom-up generation (Bo.Up.), (2) soft-select (SS), (3) Reinforce-select (RS) and (4) Variational-Reinforce-select (VRS) regarding their performance on content selection. SS and RS are trained with the selecting ratio constraint in Eq. 4. For the SS model, we further add a regularization term to encourage the maximum value of $\gamma$ to be close to 1 as in Mei et al. (2016). We first briefly introduce the tasks and important setup, then present the evaluation results.

### 5.1 Tasks and Setup

We test content-selection models on the headline and data-to-text generation task. Both tasks share the same framework with the only difference of source-side encoders.

**Headline Generation**: We use English Gigaword preprocessed by Rush et al. (2015), which pairs first sentences of news articles with their headlines. We keep most settings same as in Zhou et al. (2017), but use a vocabulary built by byte-pair-encoding (Sennrich et al., 2016). We find it speeds up training with superior performance.

**Data-to-Text Generation**: We use the Wikibio dataset (Lebret et al., 2016). The source is a Wikipedia infobox and the target is a one-sentence biography description. Most settings are the same as in Liu et al. (2018), but we use a bi-LSTM encoder for better performance.

**Heuristically extracted content**: This is used to train the selector for bottom up models and pretrain the RS and VRS model. For wikibio, we simply extract overlapped words between the source and target. In Gigaword, as the headline is more abstractive, we select the closest source word for each target word in the embedding space. Stop words and punctuations are prohibited from being selected.

**Choice of $\alpha/\epsilon$**: As seen in Sec 3.5, we need to set the hyperparameter $\alpha$ for RS/SS and $\epsilon$ for

| Gigaword | Oracle upper bound of | | | % unique | % unique | % Effect of | Entropy of |
|---|---|---|---|---|---|---|---|
| | ROUGE-1 | ROUGE-2 | ROUGE-L | Generation | Mask | Selector | Selector |
| Bo.Up. | 42.61 | 22.32 | 38.37 | 84.28 | 95.87 | 87.91 | 0.360 |
| SS | 33.15 | 14.63 | 30.68 | 82.06 | **96.23** | 85.27 | **0.392** |
| RS | 36.62 | 18.34 | 34.60 | 3.01 | 6.23 | 48.31 | 0.018 |
| VRS | **54.73** | **33.28** | **51.62** | **89.23** | 92.51 | **96.45** | 0.288 |
| Wikibio | ROUGE-4 | BLEU-4 | NIST | Generation | Mask | Selector | Selector |
| Bo.Up. | 47.28 | 49.95 | 11.06 | 31.57 | 77.42 | 40.78 | 0.177 |
| SS | 41.73 | 43.94 | 9.82 | **63.09** | **89.42** | 70.55 | **0.355** |
| RS | 44.07 | 46.89 | 10.31 | 4.55 | 43.83 | 10.38 | 0.105 |
| VRS | **52.41** | **55.03** | **11.89** | 57.62 | 77.83 | **74.03** | 0.181 |

Table 2: Diversity of content selection. The % effect of selector is defined as the ratio of unique generation and mask, which reflects the rate that changing the selector will lead to corresponding changes of the generated text.

VRS. $\alpha$ corresponds to the selecting ratio. We set them as $\alpha = 0.35$ for Wikibio and $0.25$ for Gigaword. The value is decided by running a human evaluation to get the empirical estimation. To keep comparison fairness, we tune $\epsilon$ to make VRS select similar amount of tokens with RS. The values we get are $\epsilon = 0.15n$ for Wikibio and $\epsilon = 0.25n$ for Gigaword. $n$ is the number of source tokens. [5]

### 5.2 Results and Analysis

Ideally we would expect the learned content selector to (1) have reasonable diversity so that text with various contents can be easily sampled, (2) properly control the contents described in the generated text and (3) not hurt performance. The following section will evaluate these three points in order.

**Diversity**: We first look into the diversity of content selection learned by different models. For each test data, 50 selection masks are randomly sampled from the model's learned distribution. Greedy decoding is run to generate the text for each mask. We measure the entropy of the selector, proportion of unique selection masks and generated text in the 50 samples. We further define the "effect" of the selector as the ratio of sampled unique text and mask. This indicates how often changing the selection mask will also lead to a change in the generated text. The results are averaged over all test data. Following Rush et al. (2015) and Lebret et al. (2016), we measure the quality of generated text with ROUGE-1, 2, L F-score for Gigaword and ROUGE-4, BLEU-4, NIST for Wikibio. As there is only one reference text for each source, we report an oracle upper bound of these scores by assuming an "oracle" that can choose the best text among all the candidates (Mao et al., 2015; Wang et al., 2017). Namely, out of each 50 sampled text, we pick the one with the maximum metric score. The final metric score is evaluated on these "oracle" picked samples. The intuition is that if the content selector is properly trained, at least one out of the 50 samples should describe similar contents with the reference text, the metric score between it and the reference text should be high. Table 2 lists the results. We can have the following observations:

- RS model completely fails to capture the content-level diversity. Its selector is largely deterministic, with a lowest entropy value among all models. In contrast, the selector from SS, VRS and Bo.Up. have reasonable diversity, with over $90\%$ and $75\%$ unique selection masks for Gigaword and Wikibio respectively.

- The selector from VRS has the strongest effect to the generator, especially on the Gigaword data where modifying the content selection changes the corresponding text in more than 95% of the cases. RS has the lowest effect value, which indicates that even with the selecting ratio constraint, its generator still ignores the selection mask to a large extent.

- The oracle metric score of VRS is much higher than the other two. This is beneficial when people want to apply the model to generate a few candidate text then hand-pick the suitable one. VRS has more potential than the other three to contain the expected text. SS performs worst. The gap between the

[5] $\epsilon$ corresponds to the KL divergence of the selection mask, which scales linearly with the number of source tokens, so we set it proportinally w.r.t. $n$.

soft approximation and the real distribution, as mentioned before, indeed results in a large drop of performance.

In short, compared with others, the content selector of VRS is *(1) diverse, (2) has stronger effect on the text generation and (3) with a larger potential of producing an expected text*.

**Controllability**: We have shown the content selector of VRS is diverse and has strong effect on the text generation. This section aims at examining whether such effect is desirable, i.e., whether the selector is able to properly control the contents described in the text. We measure it based on the self-bleu metric and a human evaluation.

The self-bleu metric measures the controllability by evaluating the "intra-selection" similarity of generated text. Intuitively, by fixing the selection mask, multiple text sampled from the decoder are expected to describe the same contents and thereby should be highly similar to each other. The decoder should only model surface-level diversity without further modifying the selected contents. With this in mind, for each test data, we randomly sample a selection mask from the selector's distribution, then fix the mask and run the decoder to sample 10 different text. The self-BLEU-1 score (Zhu et al., 2018) on the sampled text is reported, which is the average BLEU score between each text pair. A higher self-BLEU score indicates the sampled text are more similar with each other. The results are shown in Table 3. We can see generations from VRS have a clearly higher intra-selection similarity. SS performs even worse than RS, despite having a high effect score in Table 2. The selector from SS affects the generation in an undesirable way, which also explain why SS has a lowest oracle metric score though with a high score on content diversity and effect.

| Method | Bo.Up. | SS | RS | VRS |
|---|---|---|---|---|
| Gigaword | 46.58 | 37.20 | 48.13 | **61.14** |
| Wikibio | 38.30 | 13.92 | 25.99 | **43.81** |

Table 3: Self-Bleu score by fixing selection mask. Higher means better controllability of content selection

We further run a human evaluation to measure the text-content consistency among different models. 100 source text are randomly sampled from the human-written DUC 2004 data for task 1&2 (Over et al., 2007). Bo.Up, SS, RS and VRS are applied to generate the target text by first sampling a selection mask, then run beam search decoding with beam size 10. We are interested in seeing (1) if multiple generations from the same selection mask are paraphrases to each other (intra-consistent) and (2) if generations from different selection masks do differ in the content they described (inter-diverse). The results in Table 4 show that VRS significantly outperforms the other two in both intra-consistency and inter-diversity. RS has the lowest score on both because the selector has very weak effects on the generation as measured in the last section. Bo.Up and SS lay between them. Overall VRS is able to *maintain the highest content-text consistency* among them.

| Method | Fluency | intra-consistency | inter-diversity |
|---|---|---|---|
| Reference | 0.96 | - | - |
| Enc-Dec | 0.83 | - | - |
| Bo.Up. | 0.46 | 0.48 | 0.61 |
| SS | 0.27 | 0.41 | 0.54 |
| RS | **0.78** | 0.39 | 0.47 |
| VRS | 0.74 | **0.72** | **0.87** |

Table 4: Human evaluation on fluency, intra-consistency and inter-diversity of content selection on DUC 2004.

| Method | R-1 | R-2 | R-L | %Word |
|---|---|---|---|---|
| Zhou et al. (2017) | 36.15 | 17.54 | 33.63 | 100 |
| Enc-Dec | 35.92 | 17.43 | 33.42 | 100 |
| SS | 20.35 | 4.78 | 16.53 | 24.82 |
| Bo.Up | 28.17 | 10.32 | 26.68 | 24.54 |
| RS | 35.45 | 16.38 | 32.71 | 25.12 |
| VRS($\epsilon = 0$)-pri | **36.42** | **17.81** | **33.86** | 78.63 |
| VRS($\epsilon = 0.25$)-pri | 34.26 | 15.11 | 31.69 | 24.36 |
| VRS($\epsilon = 0$)-post | 37.14 | 18.03 | 34.26 | 78.66 |
| VRS($\epsilon = 0.25$)-post | **56.72** | **33.24** | **51.88** | 24.53 |

Table 5: Gigaword best-select results. Larger $\epsilon$ leads to more controllable selector with a bit degrade of performance. (-post) means selecting from the posterior $q_\phi(\beta|X, Y)$, (-pri) is from the prior $\mathbf{B}(\gamma_i)$.

**Performance** & **Trade-off**: To see if the selector affects performance, we also ask human annotators to judge the text fluency. The fluency score is computed as the average number of text being judged as fluent. We include generations from the standard Enc-Dec model. Table 4 shows the best fluency is achieved for Enc-Dec. Imposing

| Method | R-4 | B-4 | NIST | %Word |
|---|---|---|---|---|
| Liu et al. (2018) | 41.65 | 44.71 | | 100 |
| Enc-Dec | 42.07 | 44.80 | 9.82 | 100 |
| SS | 5.10 | 5.73 | 0.24 | 35.12 |
| Bo.Up | 8.07 | 9.52 | 0.42 | 38.79 |
| RS | 42.64 | 45.08 | 10.01 | 34.53 |
| VRS($\epsilon = 0$)-pri | **43.01** | **46.01** | **10.24** | 84.56 |
| VRS($\epsilon = 0.15$)-pri | 42.13 | 44.51 | 9.84 | 34.04 |
| VRS($\epsilon = 0$)-post | 43.84 | 46.60 | 10.27 | 85.34 |
| VRS($\epsilon = 0.15$)-post | **49.68** | **52.26** | **11.48** | 34.57 |

Table 6: Wikibio best-select results.

a content selector always affects the fluency a bit. The main reason is that when the controllability is strong, the change of selection will directly affect the text realization so that a tiny error of content selection might lead to unrealistic text. If the selector is not perfectly trained, the fluency will inevitably be influenced. When the controllability is weaker, like in RS, the fluency is more stable because it will not be affected much by the selection mask. For SS and Bo.Up, the drop of fluency is significant because of the gap of soft approximation and the independent training procedure. In general, VRS does properly decouple content selection from the enc-dec architecture, with only tiny degrade on the fluency.

Table 5/6 further measure the metric scores on Gigaword/Wikibio by decoding text from the best selection mask based on the selector's distribution (set $\beta_i = 1$ if $\mathbf{B}(\gamma_i) > 0.5$ and 0 otherwise). We include results from VRS model with $\epsilon = 0$, which puts no constraint on the mutual information. We further report the score by generating the best selection mask from the learned posterior distribution $q_\phi(\beta|X, Y)$ for VRS model. Two current SOTA results from Zhou et al. (2017) and Liu et al. (2018) and the proportion of selected source words for each model are also included. We have the following observations:

- As the value of $\epsilon$ decreases, the performance of VRS improves, but the selector loses more controllability because the model tends to over-select contents (over 75% source words selected). The text-content consistency will become low.

- Increasing $\epsilon$ sacrifices a bit performance, but still comparable with SOTA. Especially on Wikibio where the performance drop is minor. The reason should be that Wikibio is relatively easier to predict the selection but Gigaword has more uncertainty.

- Increasing $\epsilon$ improves the accuracy of the posterior selection. This would be useful when we want to perform posterior inference for some source-target pair.

- Setting $\epsilon = 0$ can actually outperform SOTA seq2seq which keeps all tokens, suggesting it is still beneficial to use the VRS model even if we do not care about the controllability.

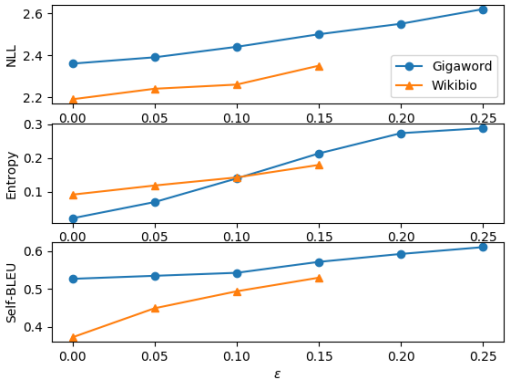


Figure 2: Negative log likelihood (NLL), selection entropy and self-BLEU as $\epsilon$ changes. NLL and self-bleu on Wikibio are added by 1 for better visualization. Lower NLL suggests higher performance. Higher entropy/self-BLEU means higher diversity/controllability.

Figure 2 visualizes how changing the value of $\epsilon$ affects the negative log likelihood (NLL), entropy of the selector and self-bleu score, which roughly correlates with performance, diversity and controllability. NLL is evaluated based on the lower bound in Eq 3 (Sohn et al., 2015). We can see as $\epsilon$ increases, the performance decreases gradually but the content selection gains more diversity and controllability. In practice we can tune the $\epsilon$ value to achieve a trade-off.

**Generation Example**: Figure 3 shows some examples from Gigaword. As can be seen, decodings from the VRS model are largely consistent with each other, in most cases only replacing one or two words with corresponding synonyms. Samples are able to faithfully convey all selected contents. In contrast, generations from SS. Bo.Up. and RS are unpredictable, differing in both selected

**Source:** indian prime minister p.v. narasimha rao 's promise of more autonomy for troubled kashmir and his plea for early state elections has sparked a violent reaction from provincial moslem and opposition parties .
**Samples from SS:**
**t1:** indian indian calls for end to violence in kashmir .
**t2:** indian pm calls for end to violence in afghanistan .
**t3:** indian pm calls for boycott of pakistan 's ruling party .
**Samples from Bo.Up:**
**t1:** india promises more autonomous more autonomy .
**t2:** indian pm promises autonomy for kashmir autonomy .
**t3:** indian pm 's promise sparks violent reaction .
**Samples from RS:**
**t1:** indian pm 's kashmir promises sparks violent reaction.
**t2:** indian pm 's promise sparks violent reaction .
**t3:** kashmir parties blast pm 's promise .
**Samples from VRS:**
**t1:** indian pm 's promise on kashmir sparks uproar .
**t2:** indian pm 's promise on kashmir sparks protests .
**t3:** indian pm 's promise for kashmir sparks controversy .

---

**Source:** factory orders for manufactured goods rose #.# percent in september , the commerce department said here thursday .
**Samples from SS:**
**t1:** u.s. consumer confidence down in january in january.
**t2:** u.s. wholesale prices up #.# percent in october .
**t3:** u.s. jobless rate rises to #.# percent in march .
**Samples from Bo.Up.:**
**t1:** september u.s. factory orders up #.# percent .
**t2:** september u.s. factory orders increase .
**t3:** factory orders up in september .
**Samples from RS:**
**t1:** u.s. factory orders up #.# percent in september .
**t2:** factory orders for manufactured goods rise .
**t3:** factory orders up in september from the year .
**Samples from VRS:**
**t1:** september factory orders up #.# percent .
**t2:** september factory orders rise #.# percent .
**t3:** september factory orders increase #.# pct .

Figure 3: Text generation examples from Gigaword. Highlighted words are selected. t1-3 are sampled from the decoder based on the selected content. Generations from VRS are more faithful to selected contents.

contents and also the way of saying. SS and Bo.Up also suffer more from the text disfluency. The generations from them are largely uncertain.

## 6 Conclusion

In this paper, we tackle the unaddressed problem of controllable content selection in text generation. We propose a general framework based on variational inference that can be potentiall applied to arbitrary tasks. On both the headline generation and data-to-text tasks, our model outperforms state-of-the-art models regarding the diversity and controllability of content selection. We further introduce an effective way to achieve a performance/controllability trade-off, which can be easily tuned to meet specific requirement.


## Acknowledgments

We thank anonymous reviewers for valuable comments, thank Aditya Mogadala, Shun Kiyono, Thomas Mclachlan and other members of the LIAT team at RIKEN AIP for useful discussions. Xiaoyu Shen is supported by IMPRS-CS fellowship. The work of J. Suzuki was partly supported by JSPS KAKENHI Grant Number JP19104418 and AIRPF Grant Number 30AI036-8. This work is also partially funded by DFG collaborative research center SFB 1102.


## References


Alexander Alemi, Ben Poole, Ian Fischer, Joshua Dillon, Rif A. Saurous, and Kevin Murphy. 2018. Fixing a broken ELBO. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 159–168, Stockholmsmssan, Stockholm Sweden. PMLR.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. 2016. Importance weighted autoencoders. In *International Conference on Learning Representations*.

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 675–686. Association for Computational Linguistics.

Marcella Cornia, Lorenzo Baraldi, and Rita Cucchiara. 2018. Show, control and tell: A framework for generating controllable and grounded captions. *arXiv preprint arXiv:1811.10652*.

Yuntian Deng, Yoon Kim, Justin Chiu, Demi Guo, and Alexander M Rush. 2018. Latent alignment and variational attention. *Nips*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *NAACL*.

Angela Fan, David Grangier, and Michael Auli. 2018. Controllable abstractive summarization. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 45–54. Association for Computational Linguistics.

Sebastian Gehrmann, Yuntian Deng, and Alexander M Rush. 2018. Bottom-up abstractive summarization. *EMNLP*.

Peter W. Glynn. 1990. Likelihood ratio gradient estimation for stochastic systems. *Commun. ACM*, 33(10):75–84.

Will Grathwohl, Dami Choi, Yuhuai Wu, Geoff Roeder, and David Duvenaud. 2018. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. In *International Conference on Learning Representations*.

Anne-Claire Haury, Pierre Gestraud, and Jean-Philippe Vert. 2011. The influence of feature selection methods on accuracy, stability and interpretability of molecular signatures. *PloS one*, 6(12):e28210.

Wan-Ting Hsu, Chieh-Kai Lin, Ming-Ying Lee, Kerui Min, Jing Tang, and Min Sun. 2018. A unified model for extractive and abstractive summarization using inconsistency loss. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 132–141. Association for Computational Linguistics.

Peter Jackson. 1998. *Introduction to expert systems*. Addison-Wesley Longman Publishing Co., Inc.

Lukasz Kaiser, Samy Bengio, Aurko Roy, Ashish Vaswani, Niki Parmar, Jakob Uszkoreit, and Noam Shazeer. 2018. Fast decoding in sequence models using discrete latent variables. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2390–2399, Stockholmsmssan, Stockholm Sweden. PMLR.

Nan Rosemary Ke, Konrad Żołna, Alessandro Sordoni, Zhouhan Lin, Adam Trischler, Yoshua Bengio, Joelle Pineau, Laurent Charlin, and Christopher Pal. 2018. Focused hierarchical RNNs for conditional sequence processing. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2554–2563, Stockholmsmssan, Stockholm Sweden. PMLR.

Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. 2016. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, pages 4743–4751.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *International Conference on Learning Representations*.

Dieterich Lawson, Chung-Cheng Chiu, George Tucker, Colin Raffel, Kevin Swersky, and Navdeep Jaitly. 2018. Learning hard alignments with variational inference. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5799–5803. IEEE.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1203–1213. Association for Computational Linguistics.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 107–117.

Wei Li, Xinyan Xiao, Yajuan Lyu, and Yuanzhuo Wang. 2018. Improving neural abstractive document summarization with explicit information selection modeling. *EMNLP*.

Junyang Lin, Xu SUN, Shuming Ma, and Qi Su. 2018. Global encoding for abstractive summarization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 163–169. Association for Computational Linguistics.

Jeffrey Ling and Alexander Rush. 2017. Coarse-to-fine attention models for document summarization. In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 33–42.

Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. 2018. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 4881–4888. AAAI Press.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421. Association for Computational Linguistics.

Xuezhe Ma, Yingkai Gao, Zhiting Hu, Yaoliang Yu, Yuntian Deng, and Eduard Hovy. 2017. Dropout with expectation-linear regularization. In *International Conference on Learning Representations*.

Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan Yuille. 2015. Deep captioning with multimodal recurrent neural networks (m-rnn). *ICLR*.

Hongyuan Mei, TTI UChicago, Mohit Bansal, and Matthew R Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *Proceedings of NAACL-HLT*, pages 720–730.

Andriy Mnih and Karol Gregor. 2014. Neural variational inference and learning in belief networks. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–1791–II–1799. JMLR.org.

Andriy Mnih and Danilo J. Rezende. 2016. Variational inference for monte carlo objectives. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 2188–2196. JMLR.org.

Amit Moryossef, Yoav Goldberg, and Ido Dagan. 2019. Step-by-step: Separating planning from realization in neural data-to-text generation. *NAACL*.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *AAAI*.

Preksha Nema, Mitesh M Khapra, Anirban Laha, and Balaraman Ravindran. 2017. Diversity driven attention model for query-based abstractive summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1063–1072.

Aaron van den Oord, Oriol Vinyals, and koray kavukcuoglu. 2017. Neural discrete representation learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6306–6315. Curran Associates, Inc.

Paul Over, Hoa Dang, and Donna Harman. 2007. Duc in context. *Information Processing & Management*, 43(6):1506–1520.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Shrimai Prabhumoye, Chris Quirk, and Michel Galley. 2019. Towards content transfer through grounded text generation. *NAACL*.

Guanghui Qin, Jin-Ge Yao, Xuening Wang, Jinpeng Wang, and Chin-Yew Lin. 2018. Learning latent semantic annotations for grounding natural language to structured data. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3761–3771.

Tapani Raiko, Mathias Berglund, Guillaume Alain, and Laurent Dinh. 2015. Techniques for learning binary stochastic feedforward neural networks. *ICLR*.

Ehud Reiter and Robert Dale. 2000. *Building natural language generation systems*. Cambridge university press.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1715–1725.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Sen Wang, and Chengqi Zhang. 2018a. Reinforced self-attention network: a hybrid of hard and soft attention for sequence modeling. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 4345–4352. International Joint Conferences on Artificial Intelligence Organization.

Xiaoyu Shen, Hui Su, Shuzi Niu, and Vera Demberg. 2018b. Improving variational encoder-decoders in dialogue generation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. 2015. Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems*, pages 3483–3491.

Amanda Stent, Rashmi Prasad, and Marilyn Walker. 2004. Trainable sentence planning for complex information presentation in spoken dialog systems. In *Proceedings of the 42nd annual meeting on association for computational linguistics*, page 79. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

George Tucker, Dieterich Lawson, Shixiang Gu, and Chris J Maddison. 2019. Doubly reparameterized gradient estimators for monte carlo objectives. In *International Conference on Learning Representations*.

George Tucker, Andriy Mnih, Chris J Maddison, John Lawson, and Jascha Sohl-Dickstein. 2017. Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 2627–2636. Curran Associates, Inc.

Liwei Wang, Alexander Schwing, and Svetlana Lazebnik. 2017. Diverse and accurate image description using a variational auto-encoder with an additive gaussian encoding space. In *Advances in Neural Information Processing Systems*, pages 5756–5766.

Wentao Wang, Zhiting Hu, Zichao Yang, Haoran Shi, Frank Xu, and Eric Xing. 2019. Toward unsupervised text content manipulation. *arXiv preprint arXiv:1901.09501*.

Lex Weaver and Nigel Tao. 2001. The optimal reward baseline for gradient-based reinforcement learning. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, UAI'01, pages 538–545, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721.

Tsung-Hsien Wen, Yishu Miao, Phil Blunsom, and Steve Young. 2017. Latent intention dialogue models. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3732–3741, International Convention Centre, Sydney, Australia. PMLR.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256.

Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2018. Learning neural templates for text generation. *EMNLP*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057.

Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. 2017. Improved variational autoencoders for text modeling using dilated convolutions. *ICML*.

Lili Yao, Nanyun Peng, Weischedel Ralph, Kevin Knight, Dongyan Zhao, and Rui Yan. 2019. Plan-and-write: Towards better automatic storytelling. *AAAI*.

Naitong Yu, Jie Zhang, Minlie Huang, and Xiaoyan Zhu. 2018. An operation network for abstractive sentence compression. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1065–1076. Association for Computational Linguistics.

Shengjia Zhao, Jiaming Song, and Stefano Ermon. 2018. The information autoencoding family: A lagrangian perspective on latent variable generative models. *UAI*.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2017. Selective encoding for abstractive sentence summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1095–1104. Association for Computational Linguistics.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: a benchmarking platform for text generation models. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1097–1100. ACM.

# Sentence-Level Content Planning and Style Specification for Neural Text Generation

**Xinyu Hua** and **Lu Wang**
Khoury College of Computer Sciences
Northeastern University
Boston, MA 02115
hua.x@husky.neu.edu luwang@ccs.neu.edu

## Abstract

Building effective text generation systems requires three critical components: content selection, text planning, and surface realization, and traditionally they are tackled as separate problems. Recent all-in-one style neural generation models have made impressive progress, yet they often produce outputs that are incoherent and unfaithful to the input. To address these issues, we present an end-to-end trained two-step generation model, where a sentence-level content planner first decides on the keyphrases to cover as well as a desired language style, followed by a surface realization decoder that generates relevant and coherent text. For experiments, we consider three tasks from domains with diverse topics and varying language styles: persuasive argument construction from Reddit, paragraph generation for normal and simple versions of Wikipedia, and abstract generation for scientific articles. Automatic evaluation shows that our system can significantly outperform competitive comparisons. Human judges further rate our system generated text as more fluent and correct, compared to the generations by its variants that do not consider language style.

## 1 Introduction

Automatic text generation is a long-standing challenging task, as it needs to solve at least three major problems: (1) *content selection* ("what to say"), identifying pertinent information to present, (2) *text planning* ("when to say what"), arranging content into ordered sentences, and (3) *surface realization* ("how to say it"), deciding words and syntactic structures that deliver a coherent output based on given discourse goals (McKeown, 1985). Traditional text generation systems often handle each component separately, thus requiring extensive effort on data acquisition and system engineering (Reiter and Dale, 2000). Recent progress has been made by developing end-to-end trained neural models (Rush et al., 2015; Yu et al., 2018; Fan et al., 2018), which naturally excel at producing fluent text. Nonetheless, limitations of model structures and training objectives make them suffer from low interpretability and substandard generations which are often incoherent and unfaithful to the input material (See et al., 2017; Wiseman et al., 2017; Li et al., 2017).

*Topic:* **US should cut off foreign aid completely.**

`/r/ChangeMyView` *Counter-argument:* It can be a useful **political bargaining chip**. A few years ago, the US **cut financial aid** to **Uganda** due to its plans to **make homosexuality a crime** punishable by death. *Please consider changing your mind!*

*Topic:* **Artificial Intelligence**

*English Wikipedia:* … **Computer science** defines **AI research** as … any **device** that **perceives its environment** and **takes actions** that **maximize its chance** of successfully **achieving its goals**. ...

*Simple Wikipedia:* **Artificial Intelligence** is **the ability** of a **computer program** or **a machine** to **think and learn**. ...

Figure 1: [Upper] Sample counter-argument from Reddit. Argumentative stylistic language for persuasion is in italics. [Bottom] Excerpts from Wikipedia, where sophisticated concepts and language of higher complexity used in the standard version are not present in the corresponding simplified version. Both: key concepts are in bold.

To address the problems, we believe it is imperative for neural models to gain adequate control on content planning (i.e., content selection and ordering) to produce coherent output, especially for long text generation. We further argue that, in order to achieve desired discourse goals, it is beneficial to enable style-controlled surface realization by explicitly modeling and specifying proper linguistic styles. Consider the task of producing counter-arguments to the topic "US should cut off foreign aid completely". A sample argument in Figure 1 demonstrates how human selects



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 591–602, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

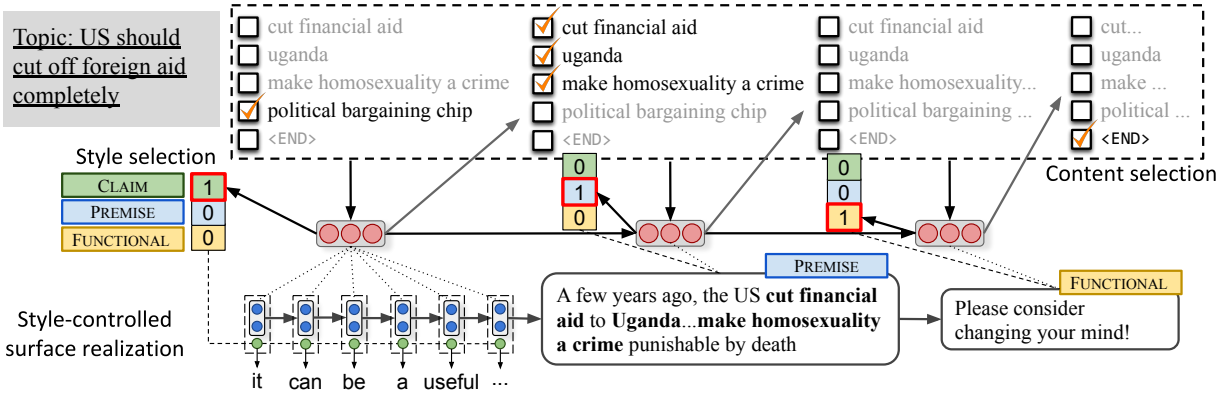


Figure 2: Overview of our framework. The LSTM content planning decoder (§ 3.2) first identifies a set of keyphrases from the memory bank conditional on previous selection history, based on which, a style is specified. During surface realization, the hidden states of the planning decoder and the predicted style encoding are fed into the realizer, which generates the final output (§ 3.3). Best viewed in color.

a series of talking points and a proper style based on the argumentative function for each sentence. For instance, the argument starts with a *proposition* on "foreign aid as a political bargaining chip", followed by a concrete *example* covering several key concepts. It ends with *argumentative stylistic language*, which differs in both content and style from the previous sentences. Figure 1 shows another example on Wikipedia articles: compared to a topic's standard version where longer sentences with complicated concepts are constructed, its simplified counterpart tends to explain the same subject with plain language and simpler concepts, indicating the interplay between content selection and language style.

We thus present an end-to-end trained neural text generation framework that includes the modeling of traditional generation components, to promote the control of content and linguistic style of the produced text.[1] Our model performs *sentence-level content planning* for information selection and ordering, and *style-controlled surface realization* to produce the final generation. We focus on conditional text generation problems (Lebret et al., 2016; Colin et al., 2016; Dušek et al., 2018): As shown in Figure 2, the input to our model consists of a topic statement and a set of keyphrases. The output is a relevant and coherent paragraph to reflect the salient points from the input. We utilize two separate decoders: for each sentence, (1) a planning decoder selects relevant keyphrases and a desired style conditional on previous selections, and (2) a realization decoder produces the text in the specified style.

We demonstrate the effectiveness of our framework on three challenging datasets with diverse topics and varying linguistic styles: persuasive argument generation on Reddit ChangeMyView (Hua and Wang, 2018); introduction paragraph generation on a *newly* collected dataset from Wikipedia and its simple version; and scientific paper abstract generation on AGENDA dataset (Koncel-Kedziorski et al., 2019).

Experimental results on all three datasets show that our models that consider content planning and style selection achieve significantly better BLEU, ROUGE, and METEOR scores than non-trivial comparisons that do not consider such information. Human judges also rate our model generations as more fluent and correct compared to the outputs produced by its variants without style modeling.

## 2 Related Work

Content selection and text planning are critical components in traditional text generation systems (Reiter and Dale, 2000). Early approaches separately construct each module and mainly rely on hand-crafted rules based on discourse theory (Scott and de Souza, 1990; Hovy, 1993) and expert knowledge (Reiter et al., 2000), or train statistical classifiers with rich features (Duboue and McKeown, 2003; Barzilay and Lapata, 2005). Advances in neural generation models have alleviated human efforts on system engineering, by combining all components into an end-to-end trained conditional text generation framework (Mei et al., 2016; Wiseman et al., 2017). However, without

[1]Data and code are available at xinyuhua.github.io/Resources/emnlp19/.

proper planning and control (Rambow and Korelsky, 1992; Stone and Doran, 1997; Walker et al., 2001), the outputs are often found to be incoherent and hallucinating. Recent work (Moryossef et al., 2019) separates content selection from the neural generation process and shows improved generation quality. However, their method requires an exhaustive search for content ordering and is therefore hard to generalize and scale. In this work, we improve the content selection by incorporating past selection history and directly feeding the predicted language style into the realization module.

Our work is also inline with concept-to-text generation, where sentences are produced from *structured* representations, such as database records (Konstas and Lapata, 2013; Lebret et al., 2016; Wiseman et al., 2017; Moryossef et al., 2019), knowledge base items (Luan et al., 2018; Koncel-Kedziorski et al., 2019), and AMR graphs (Konstas et al., 2017; Song et al., 2018; Koncel-Kedziorski et al., 2019). Shared tasks such as WebNLG (Colin et al., 2016) and E2E NLG challenges (Dušek et al., 2019) have been designed to evaluate single sentence planning and realization from the given structured inputs with a small set of fixed attribute types. Planning for multiple sentences in the same paragraph is nevertheless much less studied; it poses extra challenges for generating coherent long text, which is addressed in this work. Moreover, structured inputs are only available in a limited number of domains (Tanaka-Ishii et al., 1998; Chen and Mooney, 2008; Belz, 2008; Liang et al., 2009; Chisholm et al., 2017). The emerging trend is to explore less structured data (Kiddon et al., 2016; Fan et al., 2018; Martin et al., 2018). In our work, keyphrases are used as input to our generation system, which offer flexibility for concept representation and generalizability to broader domains.

## 3 Model

Our model tackles conditional text generation tasks where the input is comprised of two major parts: (1) a topic statement, $\mathbf{x} = \{x_i\}$, which can be an argument, the title of a Wikipedia article, or a scientific paper title, and (2) a keyphrase memory bank, $\mathcal{M}$, containing a list of talking points, which plays a critical role in content planning and style selection. We aim to produce a sequence of words, $\mathbf{y} = \{y_t\}$, to comprise the output, which can be a counter-argument, a paragraph as in Wikipedia articles, or a paper abstract.

### 3.1 Input Encoding

The input text $\mathbf{x}$ is encoded via a bidirectional LSTM (biLSTM), with its last hidden state used as the initial states for both content planning decoder and surface realization decoder. To encode keyphrases in the memory bank $\mathcal{M}$, each keyphrase is first converted into a vector $\boldsymbol{e}_k$ by summing up all its words' embeddings from GloVe (Pennington et al., 2014). A biLSTM-based *keyphrase reader*, with hidden states $\boldsymbol{h}^e_k$, is used to encode all keyphrases in $\mathcal{M}$. We also insert entries of <START> and <END> into $\mathcal{M}$ to facilitate learning to start and finish selection.

### 3.2 Sentence-Level Content Planning and Style Specification

**Content Planning: Context-Aware Keyphrase Selection.** Our content planner selects a set of keyphrases from the memory bank $\mathcal{M}$ for each sentence, indexed with $j$, conditional on keyphrases that have been selected in previous sentences, allowing topical coherence and content repetition avoidance. The decisions are denoted as a *selection vector* $\boldsymbol{v}_j \in \mathbb{R}^{|\mathcal{M}|}$, with each dimension $\boldsymbol{v}_{j,k} \in \{0, 1\}$, indicating whether the $k$-th phrase is selected for the $j$-th sentence generation. Starting with a <START> tag as the input for the first step, our planner predicts $\boldsymbol{v}_1$ for the first sentence, and recurrently makes predictions per sentence until <END> is selected, as depicted in Figure 2.

Formally, we utilize a sentence-level LSTM $f$, which consumes the summation embedding of selected keyphrases, $\boldsymbol{m}_j$, to produce a hidden state $\boldsymbol{s}_j$ for the $j$-th sentence step:

$$\boldsymbol{s}_j = f(\boldsymbol{s}_{j-1}, \boldsymbol{m}_j) \tag{1}$$

$$\boldsymbol{m}_j = \sum_{k=1}^{|\mathcal{M}|} \boldsymbol{v}_{j,k} \boldsymbol{h}^e_k \tag{2}$$

where $\boldsymbol{v}_{j,k} \in \{0, 1\}$ is the selection decision for the $k$-th keyphrase in the $j$-th sentence.

Our recent work (Hua et al., 2019) utilizes a similar formulation for sentence representations. However, the prediction of $\boldsymbol{v}_{j+1}$ is estimated by a bilinear product between $\boldsymbol{h}^e_k$ and $\boldsymbol{s}_j$, which is agnostic to what have been selected so far. While in reality, content selection for a new sentence should depend on previous selections. For in-

stance, keyphrases that have already been utilized many times are less likely to be picked again; topically related concepts tend to be mentioned closely. We therefore propose a vector $\boldsymbol{q}_j$ that keeps track of what keyphrases have been selected up to the $j$-th sentence:

$$\boldsymbol{q}_j = (\sum_{r=0}^{j} \boldsymbol{v}_r)^T \times \mathbb{E} \quad (3)$$

where $\mathbb{E} = [\boldsymbol{h}^e_1, \boldsymbol{h}^e_2, \dots \boldsymbol{h}^e_{|\mathcal{M}|}]^T \in \mathbb{R}^{|\mathcal{M}| \times H}$ is the matrix of keyphrase representations, $H$ is the hidden dimension of the keyphrase reader LSTM.

Then $\boldsymbol{v}_{j+1}$ is calculated in an attentive manner with $\boldsymbol{q}_j$ as the attention query:

$$P(\boldsymbol{v}_{j+1,k} = 1 | \boldsymbol{v}_{1:j}) = \sigma(\mathbf{w}_v^T \boldsymbol{s}_j + \boldsymbol{q}_j \mathbf{W}^c \boldsymbol{h}^e_k) \quad (4)$$

where $\sigma$ is the sigmoid funciton, and $\mathbf{w}_*$, $\mathbf{W}^*$, and $\mathbf{W}^{**}$ are trainable parameters throughout the paper. Bias terms are all omitted for simplicity.

As part of the learning objective, we utilize the binary cross-entropy loss with the gold-standard selection $\boldsymbol{v}^*_j$ as criterion over the training set $D$:

$$\mathcal{L}_{\text{sel}} = -\sum_{(\mathbf{x},\mathbf{y}) \in D} \sum_{j=1}^{J} (\sum_{k=1}^{|\mathcal{M}|} \log(P(\boldsymbol{v}^*_{j,k}))) \quad (5)$$

**Style Specification.** As discussed in § 1, depending on the content (represented as selected keyphrases in our model), humans often choose different language styles adapted for different discourse goals. Our model characterizes such stylistic variations by assigning a categorical style type $\boldsymbol{t}_j$ for each sentence, which is predicted as follows:

$$\hat{\boldsymbol{t}}_j = \text{softmax}(\mathbf{w}_s^T(\tanh(\mathbf{W}^s[\boldsymbol{m}_j; \boldsymbol{s}_j])) \quad (6)$$

$\hat{\boldsymbol{t}}_j$ is the estimated distribution over all types. We select the one with the highest probability and use a one-hot encoding vector, $\boldsymbol{t}_j$, as the input to our realization decoder (§ 3.3). The estimated distributions $\hat{\boldsymbol{t}}_j$ are compared against the gold-standard labels $\boldsymbol{t}^*_j$ to calculate the cross-entropy loss $\mathcal{L}_{\text{style}}$:

$$\mathcal{L}_{\text{style}} = -\sum_{(\mathbf{x},\mathbf{y}) \in D} \sum_{j=1}^{J} \boldsymbol{t}^*_j \log \hat{\boldsymbol{t}}_j \quad (7)$$

### 3.3 Style-Controlled Surface Realization

Our surface realization decoder is implemented with an LSTM with state calculation function $g$ to get each hidden state $\boldsymbol{z}_t$ for the $t$-th generated token. To compute $\boldsymbol{z}_t$, we incorporate the content planning decoder hidden state $\boldsymbol{s}_{J(t)}$ for the sentence to be generated, with $J(t)$ as the sentence index, and previously generated token $\boldsymbol{y}_{t-1}$:

$$\boldsymbol{z}_t = g(\boldsymbol{z}_{t-1}, \tanh(\mathbf{W}^{ws}\boldsymbol{s}_{J(t)} + \mathbf{W}^{ww}\boldsymbol{y}_{t-1})) \quad (8)$$

For word prediction, we calculate two attentions, one over the input statement $\mathbf{x}$, which produces a context vector $\boldsymbol{c}^w_t$ (Eq. 10), the other over the keyphrase memory bank $\mathcal{M}$, which generates $\boldsymbol{c}^e_t$ (Eq. 11). To better reflect the control over word choice by language styles, we directly append the predicted style $\boldsymbol{t}_{J(t)}$ to the context vectors and hidden state $\boldsymbol{z}_t$, to compute the distribution over the vocabulary[2]:

$$P(y_t | y_{1:t-1}) = \text{softmax}(\tanh(\mathbf{W}^o[\boldsymbol{z}_t; \boldsymbol{c}^w_t; \boldsymbol{c}^e_t; \boldsymbol{t}_{J(t)}])) \quad (9)$$

$$\boldsymbol{c}^w_t = \sum_{i=1}^{L} \alpha^w_i \boldsymbol{h}_i, \quad \alpha^w_i = \text{softmax}(\boldsymbol{z}_t \mathbf{W}^{wa} \boldsymbol{h}_i) \quad (10)$$

$$\boldsymbol{c}^e_t = \sum_{k=1}^{|\mathcal{M}|} \alpha_k \boldsymbol{h}^e_k, \quad \alpha^e_k = \text{softmax}(\boldsymbol{z}_t \mathbf{W}^{we} \boldsymbol{h}^e_k) \quad (11)$$

We further adopt a copying mechanism from See et al. (2017) to enable direct reuse of words from the input $\mathbf{x}$ and keyphrase bank $\mathcal{M}$ to allow out-of-vocabulary words to be included.

### 3.4 Training Objective

We jointly learn to conduct content planning and surface realization by aggregating the losses over (i) word generation: $\mathcal{L}_{\text{gen}} = -\sum_D \sum_{t=1}^{T} \log P(y^*_t | \mathbf{x}; \theta)$, (ii) keyphrase selection: $\mathcal{L}_{\text{sel}}$ (Eq. 5), and (iii) style prediction $\mathcal{L}_{\text{style}}$ (Eq. 7):

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{gen}}(\theta) + \gamma \cdot \mathcal{L}_{\text{style}}(\theta) + \eta \cdot \mathcal{L}_{\text{sel}}(\theta) \quad (12)$$

where $\theta$ denotes the trainable parameters. $\gamma$ and $\eta$ are set to 1.0 in our experiments for simplicity.

## 4 Tasks and Datasets

### 4.1 Task I: Argument Generation

Our first task is to generate a counter-argument for a given statement on a controversial issue. The input keyphrases are extracted from automatically retrieved and reranked passages with queries constructed from the input statement.

[2]The inclusion of style variables is different from our prior style-aware generation model (Hua et al., 2019), where styles are predicted but not encoded for word production.

| | Argument | Wikipedia | AGENDA |
|---|---|---|---|
| | # Args (# Threads) | (Nor. / Sim.) | |
| # Train | 272,147 (11,434) | 125,136 | 38,720 |
| # Dev | 40,291 (1,784) | 21,004 | 1,000 |
| # Test | 46,757 (1,706) | 23,534 | 1,000 |
| # Tokens | 54.87 | 70.57 / 48.60 | 141.34 |
| # Sent. | 2.48 | 3.15 / 3.20 | 5.59 |
| # KP (candidates) | 55.80 | 23.56 | 12.23 |
| # KP (selected) | 11.61 | 16.01/11.11 | 12.23 |

Table 1: Statistics of the three datasets. Average numbers are reported. For argument dataset, number of unique threads is also shown. On AGENDA, entities are extracted from abstract as keyphrases, hence all candidates are "selected".

| | CLAIM | PREMISE | FUNCTIONAL |
|---|---|---|---|
| # Arguments | 29.1% | 62.2% | 8.7% |
| # Tokens | 17.0 | 26.2 | 10.0 |

| Length $\in$ | (0, 10] | (10, 20] | (20, 30] | (30, $\infty$) |
|---|---|---|---|---|
| Normal Wikipedia | 9.9% | 40.5% | 29.8% | 19.8% |
| Simple Wikipedia | 29.3% | 51.7% | 14.6% | 4.4% |

Table 2: Sentence style distribution for argument and Wikipedia datasets.

We reuse the dataset from our previous work (Hua et al., 2019), but annotate with newly designed style scheme. We first briefly summarize the procedures for data collection, keyphrase extraction and selection, and passage reranking; more details can be found in our prior work. Then we describe how to label argument sentences with style types that capture argumentative structures.

The dataset is collected from Reddit `/r/ChangeMyView` subcommunity, where each thread consists of a multi-paragraph original post (OP), followed by user replies with the intention to change the opinion of the OP user. Each OP is considered as the input, and the root replies awarded with delta ($\Delta$), or with positive `karma` (# upvotes $>$ # downvotes) are target counter-arguments to be generated. A domain classifier is further adopted to select politics related threads. Since users often have separate arguments in different paragraphs, we treat each paragraph as one *target argument* by itself. Statistics are shown in Table 1.

**Input Keyphrases and Label Construction.** To obtain the input keyphrase candidates and their sentence-level selection labels, we first construct queries to retrieve passages from Wikipedia and news articles collected from `commoncrawl.org`.[3] For training, we construct a query per target argument sentence using its content words for retrieval, and keep top 5 passages per query. For testing, the queries are constructed from the sentences in OP (input statement).

We then extract keyphrases from the retrieved passages based on topic signature words (Lin and Hovy, 2000) calculated over the given OP. These words, together with their related terms from WordNet (Miller, 1994), are used to determine whether a phrase in the passage is a *keyphrase*. Specifically, a keyphrase is (1) a noun phrase or verb phrase that is shorter than 10 tokens; (2) contains at least one content word; (3) has a topic signature or a Wikipedia title. For each keyphrase candidate, we match them with the sentences in the target counter-argument, and we consider it to be "selected" for the sentence if there is any overlapping content word.

During test time, we further adopt a stance classifier from Bar-Haim et al. (2017) to produce a stance score for each passage. We retain passages that have a negative stance towards OP, and a greater than 5 stance score. They are further ordered based on the number of overlapping keyphrases with the OP. Top 10 passages are used to construct the input keyphrase bank, and as optional input to our model.

**Sentence Style Label Construction.** For argument generation, we define three sentence styles based on their argumentative discourse functions (Persing and Ng, 2016; Lippi and Torroni, 2016): CLAIM is a proposition, usually containing one or two talking points, e.g., "*I believe foreign aid is a useful bargaining chip*"; PREMISE contains supporting arguments with reasoning or examples; FUNCTIONAL is usually a generic statement, e.g., "*I understand what you said*". For training, we employ a list of rules extended from the claim detection method by Levy et al. (2018) to automatically construct a style label for each sentence. Statistics are displayed in Table 2, and sample rules are shown below, with the complete list in the Supplementary:

- CLAIM: must be shorter than 20 tokens and matches any of the following patterns: (a) `i (don't)? (believe|agree|...);` (b) `(anyone|all|everyone|nobody...) (should|could|need|must|might...);`

[3]The choice of news portals, statistics of the dataset, and preprocerssing steps are described in Hua et al. (2019), §4.1.

(c) `(in my opinion|my view|...)`

- PREMISE: must be longer than 5 tokens, contains at least one noun or verb content word, and matches any of the following patterns: (a) `(for (example|instance)|e.g.)`; (b) `(increase|reduce|improve|...)`
- FUNCTIONAL: contains fewer than 5 alphabetical words and no noun or verb content word

Paragraphs that only contain FUNCTIONAL sentences are removed from our dataset.

### 4.2 Task II: Paragraph Generation for Normal and Simple Wikipedia

The second task is generating introduction paragraphs for Wikipedia articles. The input consists of a title, a user-specified global style (normal or simple), and a list of keyphrases collected from the gold-standard paragraphs of both normal and simple Wikipedia. During training and testing, the global style is encoded as one extra bit appended to $\boldsymbol{m}_j$ (Eq. 2).

We construct a *new* dataset with topically-aligned paragraphs from normal and simple English Wikipedia.[4] For alignment, we consider it a match if two articles share exactly the same title with at most two non-English words. We then extract the first paragraphs from both and filter out the pair if one of the paragraphs is shorter than 10 words or is followed by a table.

**Input Keyphrases and Label Construction.** Similar to argument generation, we extract noun phrases and verb phrases and consider the ones with at least one content word as keyphrase candidates. After de-duplication, there are on average 5.4 and 3.7 keyphrases per sentence for the normal and simple Wikipedia paragraphs, respectively. For each sample, we merge the keyphrases from the aligned paragraphs as the input. The model is then trained to select the appropriate ones conditioned on the global style.

**Sentence Style Label Construction.** We distinguish sentence-level styles based on language complexity, which is approximated by sentence length. The distribution of sentence styles is displayed in Table 2.

### 4.3 Task III: Paper Abstract Generation

We further consider a task of generating abstracts for scientific papers (Ammar et al., 2018), where the input contains a paper title and scientific entities mentioned in the abstract. We use the AGENDA data processed by Koncel-Kedziorski et al. (2019), where entities and their relations in the abstracts are extracted by SciIE (Luan et al., 2018). All entities appearing in the abstract are included in our keyphrase bank. The state-of-the-art system (Koncel-Kedziorski et al., 2019) exploits the scientific entities, their relations, and the relation types. In our setup, we ignore the relation graph, and focus on generating the abstract with only entities and title as the input. Due to the dataset's relatively uniform language style and smaller size, we do not experiment with our style specification component.

[4] We download the dumps of 2019/04/01 for both dataset.

## 5 Experiments

### 5.1 Implementation Details

For argument generation, we truncate the input OP and retrieved passages to 500 and 400 words. Passages are optionally appended to OP as our encoder input. The keyphrase bank size is limited to 70 for argument, and 30 for Wikipedia and AGENDA data (based on the average numbers in Table 1), with keyphrases truncated to 10 words. We use a vocabulary size of 50K for all tasks.

**Training Details.** Our models use a two-layer LSTM for both decoders. They all have 512-dimensional hidden states per layer and dropout probabilities (Gal and Ghahramani, 2016) of 0.2 between layers. Wikipedia titles are encoded with the summation of word embeddings due to their short length. The learning process is driven by AdaGrad (Duchi et al., 2011) with 0.15 as the learning rate and 0.1 as the initial accumulator. We clip the gradient norm to a maximum of 2.0. The mini-batch size is set to 64. And the optimal weights are chosen based on the validation loss.

For argument generation, we also pre-train the encoder and the lower layer of realization decoder using language model losses. We collect all the OP posts from the training set, and an extended set of reply paragraphs, which includes additional counter-arguments that have non-negative `karma`. For Wikipedia, we consider the large collection of 1.9 million unpaired normal English Wikipedia paragraphs to pre-train the model for both normal and simple Wikipedia generation.

**Beam Search Decoding.** For inference, we utilize beam search with a beam size of 5. We disallow the repetition of trigrams, and replace the `UNK`

with the keyphrase of the highest attention score.

### 5.2 Baselines and Comparisons

For all three tasks, we consider a SEQ2SEQ with attention baseline (Bahdanau et al., 2015), which encodes the input text and keyphrase bank as a sequence of tokens, and generates the output.

For argument generation, we implement a RETRIEVAL baseline, which returns the highest reranked passage retrieved with OP as the query. We also compare with our prior model (Hua and Wang, 2018), which is a multi-task learning framework to generate both keyphrases and arguments.

For Wikipedia generation, a RETRIEVAL baseline obtains the most similar paragraph from the training set with input title and keyphrases as the query, measured with bigram cosine similarity. We further train a logistic regression model (LOGREGSEL), which takes the summation of word embeddings in a phrase and predicts its inclusion in the output for a normal or simple Wiki paragraph.

For abstract generation, we compare with the state-of-the-art system GRAPHWRITER (Koncel-Kedziorski et al., 2019), which is a transformer model enabled with knowledge graph encoding mechanism to handle both the entities and their structural relations from the input.

We also report results by our model variants to demonstrate the usefulness of content planning and style control: (1) with gold-standard[5] keyphrase selection for each sentence (Oracle Plan.), and (2) without style specification.

## 6 Results and Analysis

### 6.1 Automatic Evaluation

We report precesion-oriented BLEU (Papineni et al., 2002), recall-oriented ROUGE-L (Lin, 2004) that measures the longest common subsequence, and METEOR (Denkowski and Lavie, 2014), which considers both precision and recall.

**Argument Generation.** For each input OP, there can be multiple possible counter-arguments. We thus consider the best matched (i.e., highest scored) reference when reporting results in Table 3. Our models yield significantly higher BLEU and ROUGE scores than all comparisons while producing longer arguments than generation-based approaches. Furthermore, among our model variants, oracle content planning further improves the performance, indicating the importance of content selection and ordering. Taking out style specification decreases scores, indicating the influence of style control on generation.[6]

| | BLEU | ROUGE | MTR | Len. |
|---|---|---|---|---|
| RETRIEVAL | 7.81 | 15.68 | **10.59** | 150.0 |
| SEQ2SEQ | 3.64 | 19.00 | 9.85 | 51.7 |
| H&W (2018) | 5.73 | 14.44 | 3.82 | 36.5 |
| OURS (Oracle Plan.) | 16.30* | 20.25* | 11.61 | 65.5 |
| OURS | **13.19*** | 20.15* | 10.42 | 65.2 |
| w/o Style | 12.61* | **20.28*** | 10.15 | 64.5 |
| w/o Passage | 11.84* | 19.90* | 9.03 | 62.6 |

Table 3: Results on argument generation with BLEU (up to bigrams), ROUGE-L, and METEOR (MTR). Best systems without oracle planning are in **bold** per metric. Our models that are significantly better than all comparisons are marked with $*$ ($p < 0.001$, approximate randomization test (Noreen, 1989)).

**Wikipedia Generation.** Results on Wikipedia (Table 4) show similar trends, where our models almost always outperform all comparisons across metrics. The significant performance drop on ablated models without style prediction proves the effectiveness of style usage. Our model, if guided with oracle keyphrase selection per sentence, again achieves the best performance.

We further show the *effect of content selection on generation* on Wikipedia and abstract data in Figure 3, where we group the test samples into 10 bins based on F1 scores on keyphrase selection.[7] We observe a strong correlation between keyphrase selection and generation performance, e.g., for BLEU, Pearson correlations of $0.95$ ($p < 10^{-4}$) and $0.85$ ($p < 10^{-2}$) are established for Wikipedia and abstract. For ROUGE, the values are $0.99$ ($p < 10^{-8}$) and $0.72$ ($p < 10^{-1}$).

**Abstract Generation.** Lastly, we compare with the state-of-the-art GRAPHWRITER model on AGENDA dataset in Table 5. Although our model does not make use of the relational graph encod-

[5] "Gold-standard" indicates the keyphrases that have content word overlap with the reference sentence.

[6] We do not compare with our recent model in Hua et al. (2019) due to the training data difference caused by our new sentence style scheme. However, the newly proposed model generates arguments with lengths closer to human arguments, benefiting from the improved content planning module.

[7] We calculate F1 by aggregating the selections across all sentences. For argument generation, keyphrases are often paraphrased, making it difficult to calculate F1 reliably, therefore omitted here.

| | BLEU | ROUGE | METEOR | Length | BLEU | ROUGE | METEOR | Length |
|---|---|---|---|---|---|---|---|---|
| | Normal Wikipedia | | | | Simple Wikipedia | | | |
| RETRIEVAL | 20.10 | 28.60 | 12.23 | 44.5 | 21.99 | 33.44 | 12.97 | 34.7 |
| SEQ2SEQ | 22.62 | 27.49 | 14.74 | 52.9 | 21.98 | 29.36 | 16.94 | 52.8 |
| LOGREGSEL | 29.28 | 28.65 | **27.76** | 34.3 | 5.59 | 23.21 | 13.27 | 13.0 |
| OURS (Oracle Plan.) | 37.70* | 45.41* | 31.65* | 79.8 | 34.22* | 45.48* | 32.84* | 70.5 |
| OURS | **33.76*** | **40.08*** | 25.70 | 65.4 | **31.22*** | **40.76*** | **26.76*** | 58.7 |
| w/o Style | 31.06* | 37.72* | 24.56 | 71.0 | 27.94* | 38.20* | 25.87* | 64.5 |

Table 4: Results on Wikipedia generation. Best results without oracle planning are in **bold**. $*$: Our models that are significantly better than all comparisons ($p < 0.001$, approximate randomization test).

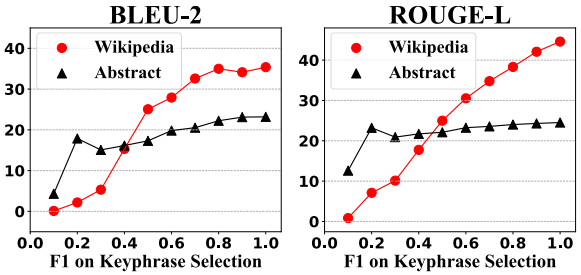


Figure 3: Effect of keyphrase selection (F1 score) on generation performance, measured by BLEU and ROUGE. Positive correlations are observed.

| | BLEU | ROUGE | MTR | Len. |
|---|---|---|---|---|
| GRAPHWRITER | 29.95 | 28.56 | 19.90 | 130.1 |
| SEQ2SEQ | 18.13 | 21.03 | 13.95 | 134.8 |
| OURS (Oracle Plan.) | 25.03 | 26.18 | 19.21 | 125.8 |
| OURS | 20.32 | 23.30 | 15.95 | 128.3 |

Table 5: Results on paper abstract generation. Notice that GRAPHWRITER models rich information about relations and relation types among entities, which is not utilized by our model.

ing, we achieve competitive ROUGE-L and METEOR scores given the oracle plans. Our model also outperforms the seq2seq baseline, which has the same input, indicating the applicability of our method across different domains.

### 6.2 Human Evaluation

We further ask three proficient English speakers to assess the quality of generated arguments and Wikipedia paragraphs. Human subjects are asked to rate on a scale of 1 (worst) to 5 (best) on **grammaticality**, **correctness** of the text (for arguments, the stance is also considered), and **content richness** (i.e., coverage of relevant points). Detailed guidelines for different ratings are provided to the raters (see Supplementary). For both tasks, we randomly choose 30 samples from the test set; outputs from two variants of our models and a human written text are presented in random order.

| | Argument | | | Wikipedia | | |
|---|---|---|---|---|---|---|
| | Gram. | Corr. | Cont. | Gram. | Corr. | Cont. |
| HUMAN | 4.81 | 3.90 | 3.48 | 4.84 | 4.73 | 4.49 |
| OURS | **3.99*** | **2.78*** | **2.61*** | **3.38** | **3.24*** | 3.43 |
| w/o Style | 3.03 | 2.26 | 2.03 | 2.99 | 2.89 | **3.50** |
| *Krippendorff's* $\alpha$ | 0.75 | 0.69 | 0.33 | 0.70 | 0.56 | 0.55 |

Table 6: Human evaluation on argument generation (Upper) and Wikipedia generation (Bottom). Grammaticality (**Gram**), correctness (**Corr**), and content richness (**Cont**) are rated on Likert scale ($1-5$). We mark our model with $*$ to indicate statistically significantly better ratings over the variant without style specification ($p < 0.001$, approximate randomization test).

According to Krippendorff's $\alpha$, the raters achieve substantial agreement on grammaticality and correctness, while the agreement on content richness is only moderate due to its subjectivity. As shown in Table 6, on both tasks, our models with style specification produce more fluent and correct generations, compared to the ones without such information. However, there is still a gap between system generations and human edited text.

We further show **sample outputs** in Figure 4. The first example is on the topic of abortion, our model captures the relevant concepts such as "*fetuses are not fully developed*" and "*illegal to kill*". It also contains fewer repetitions than the seq2seq baseline. For Wikipedia, our model is not only better at controlling the global simplicity style, but also more grammatical and coherent than the seq2seq output.

### 6.3 Further Analysis and Discussions

We further investigate the usage of different styles, and show the top frequent patterns for each argument style from human arguments and our system generation (Table 7). We first calculate the most

| | Human | Our model |
|---|---|---|
| C | It doesn't mean that; everyone should be able to | I don't believe that it is necessary; don't need to be able to |
| P | have the freedom to; is leagal in the US; imagine for a moment if; Let's say (you/your partner/a friend) | have the right to (bear arms/cast a ballot vote); For example, (if you look at/let's look at/I don't think) |
| F | Why is that?; that's ok; Would it change your mind? | I'm not sure (why/if) this is; TLDR: I don't care about this |

Table 7: Top frequent patterns captured in style CLAIM (C), PREMISE (P), and FUNCTIONAL (F) from arguments by human and our model.

frequent 4-grams per style, then extend it with context. We manually cluster and show the representative ones. For both columns, the popular patterns reflect the corresponding discourse functions: CLAIM is more evaluative, PREMISE lists out details, and FUNCTIONAL exhibits argumentative stylistic languages. Interestingly, our model also learns to paraphrase popular patterns, e.g., "*have the freedom to*" vs. "*have the right to*".

For Wikipedia, the sentence style is defined by length. To validate its effect on content selection, we calculate the average number of keyphrases per style type. The results on human written paragraphs are 2.0, 3.8, 5.8, and 9.0 from the simplest to the most complex. A similar trend is observed in our model outputs, which indicates the challenge of content selection in longer sentences.

For future work, improvements are needed in both model design and evaluation. As shown in Figure 4, system arguments appear to overfit on stylistic languages and rarely create novel concepts like humans do. Future work can lead to improved model guidance and training methods, such as reinforcement learning-based explorations, and better evaluation to capture diversity.

## 7 Conclusion

We present an end-to-end trained neural text generation model that considers sentence-level content planning and style specification to gain better control of text generation quality. Our content planner first identifies salient keyphrases and a proper language style for each sentence, then the realization decoder produces fluent text. We consider three tasks of different domains on persuasive argument generation, paragraph generation for normal and simple versions of Wikipedia, and abstract generation for scientific papers. Experimental results demonstrate the effectiveness of our model, where it obtains significantly better BLEU, ROUGE, and METEOR scores than non-trivial comparisons. Human subjects also rate our model generations as more grammatical and correct when language style is considered.

**Topic**: Aborting a fetus has some non-zero negative moral implications

**Human**: It's not the birthing process that changes things. It's the existence of the baby. Before birth, the baby only exists inside another human being. After birth, it exists on its own in the world like every other person in the world.

**Seq2seq**: i 'm not going to try to change your view here , but i do n't want to change your position . i do n't think it 's fair to say that a fetus is not a person . it 's not a matter of consciousness .

**Our model**: tl ; dr : i agree with you , but i think it 's important to note that fetuses are not fully developed . i do n't know if this is the case , but it does n't seem to be a compelling argument to me at all , so i 'm not going to try to change your view by saying that it should be illegal to kill

**Topic**: Moon Jae-in

**Simple Wikipedia**: Moon Jae-in is a South Korean politician. He is the 12th and current President of South Korea since 10 May 2017 after winning the majority vote in the 2017 presidential election.

**Seq2seq:** moon election park is a election politician who served as prime minister of korea from 2007 to 2013 . he was elected as a member of the house of democratic party in the moon 's the the moon the first serving president of jae-in , in office since 2010 .

**Our model**: moon jae-in is a south korean politician and current president of south korea from 2012 to 2017 and again from 2014 to 2017.

**Normal Wikipedia**: Moon Jae-in is a South Korean politician serving as the 19th and current President of South Korea since 2017. He was elected after the impeachment of Park Geun-hye as the candidate of the Democratic Party of Korea.

**Seq2seq**: moon winning current is a current politician who served as prime minister of korea from 2007 to 2013 . he was elected as a member of the house of democratic party in the moon 's the the current the first president of pakistan , in office . prior to that , he also served on the democratic republic of germany .

**Our model**: moon jae-in is a south korean politician serving as the 19th and current president of south korea , since 2019 to 2019 and 2019 to 2017 respectively he has been its current president ever since .

Figure 4: Sample outputs for argument generation and Wikipedia generation.

## Acknowledgements

This research is supported in part by National Science Foundation through Grants IIS-1566382 and IIS-1813341, and Nvidia GPU gifts. We are grateful to Rik Koncel-Kedziorski and Hannaneh Hajishirzi for sharing their system outputs. We also thank anonymous reviewers for their valuable suggestions.

## References


Waleed Ammar, Dirk Groeneveld, Chandra Bhagavatula, Iz Beltagy, Miles Crawford, Doug Downey, Jason Dunkelberger, Ahmed Elgohary, Sergey Feldman, Vu Ha, Rodney Kinney, Sebastian Kohlmeier, Kyle Lo, Tyler Murray, Hsu-Han Ooi, Matthew Peters, Joanna Power, Sam Skjonsberg, Lucy Wang, Chris Wilhelm, Zheng Yuan, Madeleine van Zuylen, and Oren Etzioni. 2018. Construction of the literature graph in semantic scholar. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*, pages 84–91, New Orleans - Louisiana. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Roy Bar-Haim, Indrajit Bhattacharya, Francesco Dinuzzo, Amrita Saha, and Noam Slonim. 2017. Stance classification of context-dependent claims. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 251–261. Association for Computational Linguistics.

Regina Barzilay and Mirella Lapata. 2005. Collective content selection for concept-to-text generation. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 331–338, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Anja Belz. 2008. Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Natural Language Engineering*, 14(4):431–455.

David L Chen and Raymond J Mooney. 2008. Learning to sportscast: a test of grounded language acquisition. In *Proceedings of the 25th international conference on Machine learning*, pages 128–135. ACM.

Andrew Chisholm, Will Radford, and Ben Hachey. 2017. Learning to generate one-sentence biographies from Wikidata. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 633–642, Valencia, Spain. Association for Computational Linguistics.

Emilie Colin, Claire Gardent, Yassine M'rabet, Shashi Narayan, and Laura Perez-Beltrachini. 2016. The WebNLG challenge: Generating text from DBPedia data. In *Proceedings of the 9th International Natural Language Generation conference*, pages 163–167, Edinburgh, UK. Association for Computational Linguistics.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 376–380, Baltimore, Maryland, USA. Association for Computational Linguistics.

Pablo Ariel Duboue and Kathleen R. McKeown. 2003. Statistical acquisition of content selection rules for natural language generation. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 121–128.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2018. Findings of the E2E NLG challenge. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 322–328, Tilburg University, The Netherlands. Association for Computational Linguistics.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2019. Evaluating the state-of-the-art of end-to-end natural language generation: The E2E NLG Challenge. *arXiv preprint arXiv:1901.11528*.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 1019–1027. Curran Associates, Inc.

Eduard H Hovy. 1993. Automated discourse generation using discourse structure relations. *Artificial intelligence*, 63(1-2):341–385.

Xinyu Hua, Zhe Hu, and Lu Wang. 2019. Argument generation with retrieval, planning, and realization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2661–2672, Florence, Italy. Association for Computational Linguistics.

Xinyu Hua and Lu Wang. 2018. Neural argument generation augmented with externally retrieved evidence. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 219–230, Melbourne, Australia. Association for Computational Linguistics.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally coherent text generation with neural

checklist models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 329–339, Austin, Texas. Association for Computational Linguistics.

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text Generation from Knowledge Graphs with Graph Transformers. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2284–2293, Minneapolis, Minnesota. Association for Computational Linguistics.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: Sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 146–157, Vancouver, Canada. Association for Computational Linguistics.

Ioannis Konstas and Mirella Lapata. 2013. Inducing document plans for concept-to-text generation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1503–1514, Seattle, Washington, USA. Association for Computational Linguistics.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1203–1213, Austin, Texas. Association for Computational Linguistics.

Ran Levy, Ben Bogin, Shai Gretz, Ranit Aharonov, and Noam Slonim. 2018. Towards an argumentative content search engine using weak supervision. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2066–2081, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2157–2169, Copenhagen, Denmark. Association for Computational Linguistics.

Percy Liang, Michael Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 91–99, Suntec, Singapore. Association for Computational Linguistics.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*.

Chin-Yew Lin and Eduard Hovy. 2000. The automated acquisition of topic signatures for text summarization. In *COLING 2000 Volume 1: The 18th International Conference on Computational Linguistics*.

Marco Lippi and Paolo Torroni. 2016. Argumentation mining: State of the art and emerging trends. *ACM Transactions on Internet Technology (TOIT)*, 16(2):10.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3219–3232, Brussels, Belgium. Association for Computational Linguistics.

Lara J Martin, Prithviraj Ammanabrolu, Xinyu Wang, William Hancock, Shruti Singh, Brent Harrison, and Mark O Riedl. 2018. Event representations for automated story generation with deep neural nets. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Kathleen R. McKeown. 1985. *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text*. Cambridge University Press, New York, NY, USA.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. What to talk about and how? selective generation using LSTMs with coarse-to-fine alignment. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 720–730, San Diego, California. Association for Computational Linguistics.

George A. Miller. 1994. Wordnet: A lexical database for english. In *HUMAN LANGUAGE TECHNOLOGY: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*.

Amit Moryossef, Yoav Goldberg, and Ido Dagan. 2019. Step-by-step: Separating planning from realization in neural data-to-text generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2267–2277, Minneapolis, Minnesota. Association for Computational Linguistics.

Eric W Noreen. 1989. *Computer-intensive methods for testing hypotheses*. Wiley New York.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Isaac Persing and Vincent Ng. 2016. End-to-end argumentation mining in student essays. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1384–1394, San Diego, California. Association for Computational Linguistics.

Owen Rambow and Tanya Korelsky. 1992. Applied text generation. In *Proceedings of the Third Conference on Applied Natural Language Processing*, pages 40–47, Trento, Italy. Association for Computational Linguistics.

Ehud Reiter and Robert Dale. 2000. *Building applied natural language generation systems*. Cambridge University Press.

Ehud Reiter, Roma Robertson, and Liesl Osman. 2000. Knowledge acquisition for natural language generation. In *INLG'2000 Proceedings of the First International Conference on Natural Language Generation*, pages 217–224, Mitzpe Ramon, Israel. Association for Computational Linguistics.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal. Association for Computational Linguistics.

Donia Scott and Clarisse Sieckenius de Souza. 1990. Getting the message across in rst-based text generation. *Current research in natural language generation*, 4:47–73.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. A graph-to-sequence model for AMR-to-text generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1616–1626, Melbourne, Australia. Association for Computational Linguistics.

Matthew Stone and Christine Doran. 1997. Sentence planning as description using tree adjoining grammar. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 198–205, Madrid, Spain. Association for Computational Linguistics.

Kumiko Tanaka-Ishii, Kôiti Hasida, and Itsuki Noda. 1998. Reactive content selection in the generation of real-time soccer commentary. In *Proceedings of the 17th international conference on Computational linguistics-Volume 2*, pages 1282–1288. Association for Computational Linguistics.

Marilyn A. Walker, Owen Rambow, and Monica Rogati. 2001. SPoT: A trainable sentence planner. In *Second Meeting of the North American Chapter of the Association for Computational Linguistics*.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263, Copenhagen, Denmark. Association for Computational Linguistics.

Zhiwei Yu, Jiwei Tan, and Xiaojun Wan. 2018. A neural approach to pun generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1650–1660, Melbourne, Australia. Association for Computational Linguistics.

# Translate and Label! An Encoder-Decoder Approach for Cross-lingual Semantic Role Labeling

**Angel Daza and Anette Frank**
Leibniz ScienceCampus "Empirical Linguistics and Computational Language Modeling"
Department of Computational Linguistics
Heidelberg University
69120 Heidelberg, Germany
{daza,frank}@cl.uni-heidelberg.de

## Abstract

We propose a Cross-lingual Encoder-Decoder model that simultaneously translates and generates sentences with Semantic Role Labeling annotations in a resource-poor target language. Unlike annotation projection techniques, our model does not need parallel data during inference time. Our approach can be applied in monolingual, multilingual and cross-lingual settings and is able to produce dependency-based and span-based SRL annotations. We benchmark the labeling performance of our model in different monolingual and multilingual settings using well-known SRL datasets. We then train our model in a cross-lingual setting to generate new SRL labeled data. Finally, we measure the effectiveness of our method by using the generated data to augment the training basis for resource-poor languages and perform manual evaluation to show that it produces high-quality sentences and assigns accurate semantic role annotations. Our proposed architecture offers a flexible method for leveraging SRL data in multiple languages.

## 1 Introduction

Semantic Role Labeling (SRL) extracts semantic predicate-argument structure from sentences. This has proven to be useful in Neural Machine Translation (NMT) (Marcheggiani et al., 2018), Multi-document-summarization (Khan et al., 2015), AMR parsing (Wang et al., 2015) and Reading Comprehension (Mihaylov and Frank, 2019). SRL consists of three steps: i) predicate detection, ii) argument identification and iii) role classification. In this work we focus on PropBank SRL (Palmer et al., 2005), which has proven its validity across languages (van der Plas et al., 2010). While former SRL systems rely on syntactic features (Punyakanok et al., 2008; Täckström et al., 2015), recent neural approaches learn to model both argument detection and role classification given a predicate (Marcheggiani et al., 2017; He et al., 2017), and even jointly predict predicates inside sentences (He et al., 2018; Cai et al., 2018). While these approaches alleviate the need for pipeline models, they require sufficient amounts of training data to perform adequately. To date, such models have been tested primarily for English, which offers a considerable amount of high-quality training data compared to other languages. The lack of sufficiently large SRL datasets makes it hard to straightforwardly apply the same architectures to other languages and calls for methods to augment the training data in lower-resource languages.

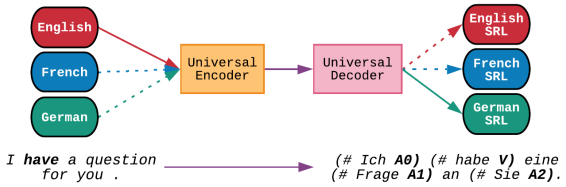


Figure 1: We propose an Encoder-Decoder model that translates a sentence into a target language and applies SRL labeling to the translated words. In this example we translate from English to German and label roles for the predicate *have*.

There is significant prior work on SRL data augmentation (Hartmann et al., 2017), annotation projection for monolingual (Fürstenau and Lapata, 2012; Hartmann et al., 2016), and cross-lingual SRL (Padó and Lapata, 2009; van der Plas et al., 2011; Akbik et al., 2015, 2016). A drawback of cross-lingual projection is that even at prediction time it requires parallel sentences, a semantic role labeler on the source side, as well as syntactic information for both language sides. Thus, it is desirable to design an architecture that can make use of existing annotations in more than one lan-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 603–615, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

guage and that learns to translate input sentences to another language while transferring semantic role annotations from the source to the target.

Techniques for low-resource Neural Machine Translation (NMT) show the positive impact on target predictions by adding more than one language during training, such as Multi-source NMT (Zoph and Knight, 2016) and Multilingual NMT (Johnson et al., 2017; Firat et al., 2016a), whereas Mulcaire et al. (2018) show the advantages of training a single polyglot SRL system that improves over monolingual baselines in lower-resource settings. In this work, we propose a general Encoder-Decoder (Enc-Dec) architecture for SRL (see Figure 1). We extend our previous Enc-Dec approach for SRL (Daza and Frank, 2018) to a cross-lingual model that translates sentences from a source language to a (lower-resource) target language, and during decoding jointly labels it with SRL annotations.[1]

Our contributions are as follows:

- We propose the first cross-lingual multilingual Enc-Dec model for PropBank SRL.
- We show that our cross-lingual model can generate new labeled sentences in a target language without the need of explicit syntactic or semantic annotations at inference time.
- Cross-lingual evaluation against a labeled gold standard achieves good performance, comparable to monolingual SRL results.
- Augmenting the training set of a lower-resource language with sentences generated by the cross-lingual model achieves improved F1 scores on the benchmark dataset.
- Our universal Enc-Dec model lends itself to monolingual, multilingual and crosslingual SRL and yields competitive performance.

## 2 An Extensible Model for SRL

### 2.1 One Model to Treat Them All

We define the SRL task as a sequence transduction problem: given an input sequence of tokens $X = x_1, ..., x_i$, the system is tasked to generate a sequence $Y = y_1, ..., y_j$ consisting of words interleaved with SRL annotations. Defining the task in this fashion allows $X$ and $Y$ to be of different lengths and therefore target sequences may also

[1]Code is available at: https://github.com/Heidelberg-NLP/SRL-S2S.

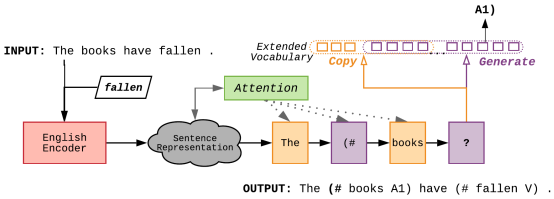


Figure 2: Monolingual Enc-Dec model for SRL with copying (Daza and Frank, 2018). We generalize this architecture to multilingual and cross-lingual SRL.

contain word tokens of different languages if desired. This means that we could train an Enc-Dec model that learns not only to label a sentence, but to jointly translate it while applying SRL annotations directly to the target language. Moreover, following conceptually the multilingual Enc-Dec model proposed by Johnson et al. (2017), we can train a single model that allows for joint training with multiple language pairs while sharing parameters among them. We apply a similar joint multilingual learning method to produce structured output sequences in the form of translations enriched with SRL annotations on the (lower-resource) target language (cf. Figure 3). We will apply this universal structure-inducing Enc-Dec model to the Semantic Role Labeling task, and show that it can be deployed in three different settings:

i) **monolingual**: encode a sentence in a given language and learn to decode a labeled sequence by reproducing the source words and inserting the appropriate structure-indicating labels in the output (cf. Figure 2). A copying mechanism (Gu et al., 2016) allows this model to reproduce the input sentence as faithfully as possible.

ii) **one-to-one multilingual**: train a single, joint model to generate $n$ different structure-enriched target languages given inputs in the same language. For example: Labeled English (*EN-SRL*) given an *EN* sentence or Labeled German (*DE-SRL*) given a *DE* sentence. This multilingual model still relies on copying to relate each labeled output sentence to its corresponding input counterpart. However, unlike (i), it has the advantage of sharing parameters among languages.

iii) **cross-lingual**: generate outputs in $n$ different target languages given inputs in $m$ different source languages, for example: Labeled German (*DE-SRL*) and Labeled French (*FR-SRL*) given an *EN* sentence (see Figure 3). In this setting, we do not restrict the model to *copy* words from the

source sentence but train it to *translate* them.

In Section 2.2 we describe how the basic Enc-Dec model for SRL is constructed and in Section 2.3 we describe the additional components that allow us to generalize this architecture to the one-to-one multilingual and cross-lingual scenarios.

### 2.2 Encoder-Decoder Architecture

We reimplement and extend the Enc-Dec model with attention (Bahdanau et al., 2015) and copying (Gu et al., 2016) mechanisms for SRL proposed by Daza and Frank (2018). This model encodes the source sentence and decodes the input sequence of words (in the same language) interleaved with SRL labels.

**Data Representation.** Similar to other prior work (Liu et al., 2018) and our own (Daza and Frank, 2018), we linearize the SRL structure in order to process it as a sequence of symbols suitable for the Enc-Dec architecture. We restrict ourselves to argument identification and labeling of one predicate at a time. We feed the gold predicate in training and inference, and process each sentence as many times as it has predicates. An opening bracket *(#* indicates the start of a labeled-argument region; a closing labeled bracket, e.g. *A0)*, indicates the ending and the tag of the labeled region (see Figure 2).

**Vocabulary.** We define a shared vocabulary consisting of all source and target words $\mathcal{V} = \{v_1, ..., v_N\} \cup \{UNK\}$ and the SRL labels $\mathcal{L} = \{l_1, ..., l_M\}$. In addition, we employ a per-instance extension set $\mathcal{X} = \{x_1 ..., x_{T_x}\}$ containing all words from the source sequence. Our final vocabulary is $\mathcal{V} \cup \mathcal{L} \cup \mathcal{X}$.

**Encoder.** In our prior work (Daza and Frank, 2018) we used a 2-layer BiLSTM as encoder. In this paper, we adopt the Deep BiLSTM Encoder from He et al. (2017) which has been shown to work well for SRL models. Again following He et al. (2017), we define the encoder input vector $x_i$ as the concatenation of a word embedding $w_i$ and a binary predicate-feature embedding $p_i$ indicating at each time-step whether the current word is a predicate or not [2]. The encoder outputs a series of hidden states $h_1, ..., h_{T_x}$ representing each token. We refer to this series of states as $\mathbf{H}$.

**Attention.** To improve the access to the source sentence representation, we include the attention mechanism proposed by Bahdanau et al. (2015), which computes a context vector at each time step *t* based on $\mathbf{H}$ and the current decoder state.

**Decoder.** We use a single-layer Decoder with LSTM cells (Hochreiter and Schmidhuber, 1997) and a copying mechanism. It emits an output token $y_t$ from a learned score $\psi_g$ over the vocabulary at each time step *t* given its state $s_t$, the previous output token $y_{t-1}$, and the attention context vector $c_t$. In addition, a copying score $\psi_c$ is calculated. The decoder learns from these scores when to generate a new token and when to copy from the encoded hidden states $\mathbf{H}$. Formally we compute the scores as:

$$\begin{aligned} \psi_g(y_t = v_i) &= W_o[s_t; c_t], \quad v_i \, \epsilon \, \mathcal{V} \cup \mathcal{L} \\ \psi_c(y_t = x_j) &= \sigma(h_j^T W_c) s_t, \quad x_j \, \epsilon \, \mathcal{X} \end{aligned} \tag{1}$$

where $W_o \epsilon \mathbb{R}^{N \times 2d_s}$ and $W_c \epsilon \mathbb{R}^{d_h \times d_s}$ are learnable parameters and $s_t$, $c_t$ are the current decoder state and context vector, respectively. These scores are used to compute two distributions: one for the likelihood of copying ($\mathbf{c}$) $y_t$ and another for the likelihood of generating ($\mathbf{g}$) $y_t$. Formally:

$$\begin{aligned} p(y_t | s_t, y_{t-1}, c_t, \mathbf{H}) = p(y_t, \mathbf{g} | s_t, y_{t-1}, c_t) + \\ p(y_t, \mathbf{c} | s_t, y_{t-1}, \mathbf{H}) \end{aligned} \tag{2}$$

The two distributions are then normalized by a final *softmax* layer from which we compute a joint likelihood of $y_t$ and choose the token with the highest score within this joint likelihood.

### 2.3 Multilingual Extensions

We generalize the monolingual Enc-Dec model for SRL to a multilingual SRL system by adding two main components:

**Translation Token.** Like Johnson et al. (2017), we prefix the source sequence with a special token that indicates the expected language of the target sequence. If the source is in *EN* and the target is a German sentence with SRL labels, the source sentence will be preceded by the token *<2DE-SRL>*.

**Language Indicator Embeddings.** We want the model to profit from the common role label inventory used across languages, yet at the same time there are subtle differences in role labeling and how roles are linguistically marked in the different languages[3]. Hence, we define *N* different language indicators (e.g., *FR, DE*) and represent

[2] These two additions already show improvements compared to the reported results in Daza and Frank (2018).

[3] e.g. the role A2 (Beneficiary) can be PP in *EN* and *FR*, but dative NP in *DE* (DativeNP)

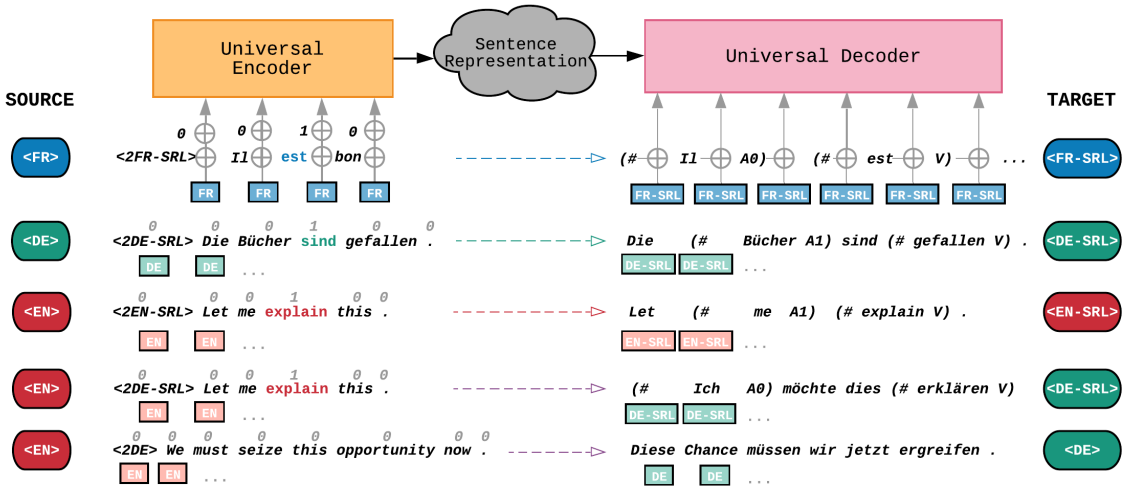


Figure 3: A universal structure-inducing Enc-Dec model with copying and sharing of parameters across languages that jointly translates and labels sentences. It can also be trained cross-lingually and be augmented with classic machine translation data (as shown in the two bottom rows of the figure).

each of them with a randomly initialized language indicator vector that we tune during training. The model can use these language indicator embedding vectors to leverage language-specific properties when generating SRL annotations. Also, by using these embeddings in the decoder, we can help it to stay consistent regarding the language it generates[4].

Thus, in all multilingual settings, at each time step $t$ we feed the **Encoder** with a concatenation of the previous encoder state $h_{t-1}$, the word embedding $w_t$ of the current token, the embedded predicate indicator $p_t$ and the language indicator embedding $l_t$. The Encoder state update is defined as:

$$h_t = LSTM([h_{t-1}; w_t; p_t; l_t]) \quad (3)$$

Likewise, on the **Decoder** side we concatenate the representations for both word tokens and label tokens with the language indicator vector to produce tokens in a specific language. For SRL-labeled output sentences the indicator token for the language embedding is *DE-SRL, FR-SRL, ...* depending on the target language. Formally, at each time step the decoder updates its state by taking into account the previous decoder state $s_{t-1}$, the previous generated token[5] $y_{t-1}$, the language indicator embedding $l_{t-1}$ and the attention context vector $c_t$:

$$s_t = LSTM([s_{t-1}; y_{t-1}; l_{t-1}; c_t]) \quad (4)$$

## 3 Data

### 3.1 SRL Monolingual Datasets

Two labeling schemes have been established for PropBank SRL: span-based and dependency-based. In the former, arguments are characterized as word-spans. This scheme was introduced in the CoNLL-05 Shared-task (Carreras and Màrquez, 2005) and is available only for English. In the dependency-based SRL format, only the syntactic heads of arguments are labeled. This format was defined in the CoNLL-09 Shared task (Hajič et al., 2009), which includes SRL labeled data for seven languages.[6]

We will use span-based and dependency-based data for English to benchmark the monolingual system. For the multilingual experiments, we use the dependency-based annotations, given that there is labeled data available in different languages on this format. Specifically, we use the English and German portions of CoNLL-09 and the automatically annotated French SRL corpus of van der Plas et al. (2011) for training and the human-labeled sentences from van der Plas et al. (2010) for testing. Both corpora contain a similar

[4] Johnson et al. (2017) only use a translation token, but our training data is significantly smaller than theirs.

[5] During training we use teacher forcing, feeding the gold target token instead of the previously generated token

[6] Note that CoNLL-09 is not a parallel corpus. All data was annotated independently and later ported to CoNLL.

| Mono-lingual | Language | Train | | Test |
|---|---|---|---|---|
| | | # Sents | w/ 1-Pred | w/ 1-Pred |
| CoNLL-05 | EN [Span] | 75,187 | 94,497 | 5,476 |
| CoNLL-09 | EN [Dep] | 39,279 | 180,446 | 10,626 |
| CoNLL-09 | DE [Dep] | 36,020 | 39,138 | 2,044 |
| v.d. Plas | FR [Dep] | 40,075 | 73,094 | 2,036 |

Table 1: Train and Test Data for Monolingual Models. We show the original number of sentences and the size of the "expanded" data with one copy per predicate.

| Cross-lingual Model | # Sentences |
|---|---|
| EN - DE-SRL (Akbik, 2015) | 63,397 |
| EN - FR-SRL (Akbik, 2015) | 40,827 |
| EN - FR (UN) | 100,000 |
| EN - DE (Europarl) | 100,000 |

Table 2: Data used for Cross-lingual Models: From the SRL parallel data available we take 90% for training and use the rest as a *Dev* set for our experiments. We add the non-labeled data (from UN and Europarl) during training to enforce translation knowledge.

label set as the English PropBank[7]. For statistics on the size of the datasets see Table 1.

### 3.2 Datasets for Cross-lingual SRL

We use the dependency-based labeled German and French SRL corpus from Akbik et al. (2015) which was produced via annotation projection. These sentences are already pre-filtered to ensure that the predicate sense of the source predicate is preserved in the target sentence. Since the role labels are projected from automatically PropBank-parsed English sentences, all languages share the same label set. The underlying corpus for this dataset is composed of Machine Translation (MT) parallel corpora: Europarl (Koehn, 2005) for *EN-DE* (about 63K sents), and UN (Ziemski et al., 2016) for *EN-FR* (about 40K sents).

Since we only had access to the labeled sentences (target-side), we constructed our parallel training pairs *EN* to *FR-SRL* and *EN* to *DE-SRL* by finding the original source English counterparts. We use Flair (Akbik et al., 2018) to predict PropBank frames on the English source sentences and find the alignment to the labeled predicate on the target side using fast-align (Dyer et al., 2013).

In addition to the parallel SRL-labeled data, we choose a subset of 100K parallel (non-labeled) sentences for each language pair from the mentioned MT datasets (Europarl and UN corpora) to improve the translation quality of the model, we use 90% for training and the rest as a development set. The data is summarized in Table 2.

[7]French data was directly annotated using the English labelset but German CoNLL-09 contains additional core labels A5-A9 and does not contain *AM-* modifier labels

## 4 Experiments and Results

### 4.1 General Settings

We use the AllenNLP (Gardner et al., 2018) Enc-Dec model as a basis for our implementation. Our model is trained to minimize the negative log-likelihood of the next token. Hyperparameters and model sizes are provided in Supplement A.1. We use pre-trained word embeddings (fine-tuned during training) for the 3 languages: GloVe (Pennington et al., 2014) for *EN* and the pre-trained vectors from Grave et al. (2018) for *FR* and *DE*. We also train versions with contextual word representations: pre-trained English 1024-dimensional ELMo (Peters et al., 2018) and multilingual 768-dimensional BERT-small (Devlin et al., 2019) representations.

### 4.2 Monolingual Experiments and Results

We train three separate monolingual versions for *EN*, *DE* and *FR*. We first benchmark our system against a wide variety of English models (span- and dependency-based) that perform the role classification task with gold predicates to show that our labeling performance is competitive with the existent SOTA neural models for English. This is shown in Table 3. The performance of *DE* and *FR* is shown in Table 4 where we compare all monolingual systems for the three languages (top half), against the one-to-one multilingual versions (bottom half). Results for *EN* show that the Enc-Dec architecture is competitive with the GloVe-based models (although still 4 F1 points below SOTA in most cases), however it benefits more from ELMo, achieving SOTA results for span-based and dependency-based SRL.

### 4.3 Multilingual Experiments and Results

We train a single multilingual model with the concatenation of the training data for the three languages *EN*, *DE* and *FR* that we previously used on the monolingual experiments. We use a common vocabulary for the three languages and keep all tokens that occur more than 5 times in the combined dataset. We train the model with batches containing instances randomly chosen from the individual languages (this means that each batch might contain examples from different language pairs).

| Type | Model | Word Repres. | CoNLL-05 WSJ | CoNLL-05 OOD | CoNLL-09 WSJ | CoNLL-09 OOD |
|---|---|---|---|---|---|---|
| Span SRL | He 2017 | GloVe | 84.6 | 73.6 | - | - |
| | Daza 2018 | GloVe | 79.2 | 68.4 | - | - |
| | He, 2018 | ELMo | 83.9 | 73.7 | - | - |
| | Tan, 2018 | GloVe | 84.8 | 74.1 | - | - |
| | Strubell 18 [LISA] | GloVe | 84.6 | 74.5 | - | - |
| | Strubell 18 [LISA*] | ELMo | 86.5 | 78 | - | - |
| | Ouchi 2018 | ELMo | **88.5** | 79.6 | - | - |
| Dep SRL | Roth 2016 | DPE* | - | - | 87.7 | 76.1 |
| | Marcheggiani 2017 | Dyer* | - | - | 87.7 | 77.7 |
| | Cai et al 2018 | GloVe | - | - | 89.6 | 79 |
| Dep and Span SRL | FitzGerald 2015 | GloVe | 80.3 | 72.2 | 87.8 | 75.5 |
| | Li 2019 | ELMo | 87.7 | - | 90.4 | - |
| | Ours [Mono] | GloVe | 80.4 | 70.5 | 85.5 | 75.7 |
| | Ours [Mono] | ELMo | 88.3 | **80.9** | **90.8** | **84.1** |

Table 3: CoNLL-09 and CoNLL-05 Test Sets for English. Our model with ELMo shows SOTA performance on both types of SRL. LISA* only reports ELMo with predicted predicates; DPE*: dependency path embeddings; Dyer*: Dyer et al. 2015.

| Model | EN-Test | DE-Test | FR-Test |
|---|---|---|---|
| SOTA models* | 90.4 | 80.1 | 73 |
| Ours-EN [Mono + GloVe] | 85.5 | - | - |
| Ours-DE [Mono + GloVe] | - | 61.9 | - |
| Ours-FR [Mono + GloVe] | - | - | 70.3 |
| Mulcaire 2018 [Multi + GloVe] | 86.5 | 69.9 | - |
| Ours [Multi + GloVe] | 87 | 68.2 | 70.5 |
| Ours [Multi + ELMo] | 91.1 | 75.7 | 70.7 |
| Ours [Multi + BERT] | 89.7 | 77.2 | 72.4 |

Table 4: F1 scores for role labeling on dependency-based SRL data. EN and DE Tests: CoNLL-09; FR-Test: van der Plas et al. (2011). State of the art (SOTA) models* are: Cai et al. (2018) [GloVe] for EN, Roth and Lapata (2016) [Dependency-path Embeddings] for DE and van der Plas et al. (2014) [Non-neural] for FR, respectively.

Multilingual training yields improvement on the three languages studied in this paper when compared to our monolingual baselines, particularly for German, which shows more than 6 points (F1) of improvement. In addition, we compare with the polyglot SRL system of Mulcaire et al. (2018) (which also leverages data from multiple languages during training), obtaining better results for English using GloVe. We then show that adding contextual representations to our model results in bigger improvements across the board.

### 4.4 Cross-Lingual Experiments and Results

**Training.** After validating the robustness of our architecture when handling different languages at the same time, we now train a cross-lingual SRL version. This setting differs from the previous two because the model needs to learn two tasks: besides generating appropriate SRL labels, it needs to translate from source into a target language. To do so, we train a single model using the concatenation of the parallel datasets listed in Table 2 and described in Section 3.2. We further include Machine Translation (MT) data to reinforce the translation knowledge of the model, so that it can generate fluent (labeled) target sentences. As in the multilingual experiments, we train the model with alternating batches of instances randomly chosen from the individual language pairs. Note that the amount of MT data that we can add is restricted: the labeled multilingual data is relatively small and labeling performance suffers when the MT data gets too dominant.

**Evaluating Cross-lingual SRL.** As in classical MT, evaluation is difficult, since the system outputs will approximate a target reference but will never be guaranteed to match it. Hence in this setting we do not have a proper gold standard to evaluate the labeled outputs, since we are generating labeled target sentences from scratch. Similar to MT research, we apply BLEU score (Papineni et al., 2002) to measure the closeness of the outputs against our *Dev* Set.

The upper part of Table 5 compares the scores of two versions of the Enc-Dec model trained on the cross-lingual data from Table 2 systems, one using GloVe embeddings and the second using BERT, respectively. To better distinguish translation vs. labeling quality, we compute BLEU scores for the system outputs against labeled reference sentences in three different ways: on *words only*, *labels only*, and on *full labeled sequences* (both word and label outputs). We see that the prediction of words is similar in the two languages, but labeling is more difficult for *DE* than for *FR* for both systems. Also we observe that adding multilingual BERT is very helpful to obtain even more fluent and correct labeled outputs (according to BLEU) resulting in ca. +9 points in German and +5 in French on the full sequences. This is very important given that we have a small training set compared to classic NMT scenarios.

The bottom part of Table 5 shows the scores when restricting the evaluation to sentences with score $\geq 10$. We observed that this threshold[8] is a good trade-off in both the amount of kept sentences (above the threshold) and average BLEU

[8]We tried with thresholds of 5, 10, 20 and 30.

| | German | | | French | | |
|---|---|---|---|---|---|---|
| **Model [Filter]** | **F-Seq** | **Word** | **Label** | **F-Seq** | **Word** | **Label** |
| XL-GloVe [All] | 18.86 | 17.17 | 25.52 | 28.99 | 17.36 | 32.76 |
| XL-BERT [All] | 27.22 | 27.36 | 29.59 | 33.59 | 22.48 | 37.17 |
| XL-GloVe [$\geq$ 10] | 30.58 | 36.71 | 51.68 | 38.99 | 43.79 | 61.73 |
| XL-BERT [$\geq$ 10] | 36.95 | 41.36 | 55.73 | 42.66 | 46.52 | 65.32 |

Table 5: Cross-lingual (XL) system results using BLEU score on individual languages inside the *Dev* set. We compute BLEU on labeled sequences (F-Seq), and separately for words and only labels. We also show scores when pre-filtering on F-Seq with BLEU $\geq$ 10.

score increase (presumably sentence quality). By keeping only the filtered subset of sentences we get an improvement on average of approx. 10 BLEU points on the full sequences (*F-Seq*), and almost double the score for *labels only*. This holds for GloVe and BERT versions on both languages.

**Output Filtering and Data Generation.** We use our cross-lingual model as a labeled data generator by applying it on *EN* sentences from Europarl (100K) and UN corpora (100K)[9] and let the model predict *DE-SRL* and *FR-SRL* as target languages. This results in *unseen* German and French labeled sentences. Since we cannot guarantee that the generated sentences preserve the source predicate meaning, we filter all outputs by keeping only those that come close to the original sentence meaning. We approximate this by back-translating the generated outputs (stripping the labels and keeping only the words) using the pre-trained *DE-EN* model from OpenNMT (Klein et al., 2017).

We compare the back-translations to the sentences that we originally presented to the system and, using the previously described filtering heuristic, we keep only those whose BLEU score is equal or greater than 10. The logic behind this is that if the back-translation is close enough to the source, the generated sentence preserves a fair amount of the original sentence meaning[10]. With this strategy, after applying the *BLEU filter*, we end up with a parallel dataset of 44K generated sentences for *(EN, DE-SRL)* and 32K for (*EN*, *FR-SRL*). In the next section we show more detailed evaluation measures of the system outputs, focusing on the filtered dataset that we just described.

[9]Note that these are taken from a different subset than the parallel sentences used during training.

[10]BLEU score is used as a naive approach to avoid excessively noisy data but we could also develop, for example, a semantic similarity metric to also keep sentences that are close enough to the original predicate sense meaning.

### 4.5 Cross-Lingual Detailed Evaluation

We are aware that BLEU score gives only a rough estimate of the actual quality of the outputs, therefore we propose to measure the performance of our system in two more detailed evaluation settings: (i) a small-scale **human evaluation** where we evaluate the assigned SRL labels against 226 sentences that were manually judged and annotated to give an estimation of the quality of the generated data, (ii) an **extrinsic evaluation** using labeled sentences generated by our system to augment the training set for a resource-poor language. We conduct the extrinsic evaluation on German and French and the manual evaluation only on the German data, which proved to be the more challenging language compared to French.

#### 4.5.1 Human Evaluation

To provide an in-depth quality assessment of the generated sentences, we create a small-scale gold standard consisting of 226 sentences. To select a representative sample from our newly generated labeled sentences,[11] we analyze the distribution of labels in the data and apply stratified sampling to cover as many predicates as possible and as many role label variants as possible. We judge these sentences on the quality of the generated language and annotate them with PropBank roles.

**SRL Gold Standard.** As we are lacking trained PropBank annotators, we mimic the question-based role annotation method of He et al. (2015), who constructed QA pairs in order to label the predicate-argument structure of verbs. The annotation involves several subtasks: The first is to generate questions targeting a specific verb in a sentence and to mark as answers a subset of words from the same sentence. The next subtask is to choose the head word of each selected subset and to assign a PropBank label to this head according to a table that correlates WH-phrases with the most likely label.[12]

We ask two linguistically trained annotators to perform the whole task independently and compute Krippendorff's Alpha (Krippendorff, 1980) on the role labels, which results in an inter-annotator agreement score of 82.83. We resolved conflicting annotations through discussion among

[11]i.e., the generated sentences for which we measured a BLEU score $\geq$ 10 against the source using back-translation.

[12]We provide this correlation table and the full annotation guidelines in appendix A.2 in the Supplement.

the annotators. The resulting gold standard contains 737 annotated roles. Notably, the most prominent roles (as in the CoNLL datasets) are A0 and A1 which are normally related to the agent and the patient in sentences, but the annotated data also includes modifier roles such as temporal, modal, discourse markers, among others[13].

**Translation Quality.** We ask two different annotators to score each output sentence (they see only the words, not the labels) on a scale of 1-5 for *Quality* (1: 'is completely ungrammatical'; 5: 'is perfectly grammatical') and for *Naturalness* (1: 'The sentence is not what a native speaker would write'; 5: 'The sentence could have been written by a native speaker'). We obtain a high average score of 4.4 for *Quality* and 4.2 for *Naturalness*.

**SRL Performance on Gold Standard.** We use our human-annotated sentences to measure the automatic labeling performance of our cross-lingual SRL model which we call XL-BERT). We obtain 73.21 F1 score (73.33 precision, 73.1 recall). We also measure the performance of the ZAP label projection system of Akbik and Vollgraf (2018) on this data (we only consider arguments of the predicates that were annotated). ZAP obtains a low F1 score of 56.03 (42.65 precision, 81.7 recall). Thus, XL-BERT shows much better, and more precise results compared to this baseline and achieves overall very acceptable and stable labeling quality. This shows that the joint translation-labeling task is successful. ZAP, by contrast, shows more unstable results, which might be due to word alignment noise. Although we train on such data, our model can also loose some of this noise, given that the same model is trained to produce more than one labeled language, namely *FR-SRL* and *DE-SRL*.

#### 4.5.2 Extrinsic Task: Data Augmentation

Finally, we augment the training sets of our two resource-poor languages *DE* and *FR*, in portions of 10K until we cover the complete generated data. We compare the increase in F1 score when training models with different amounts of additional data. We also add a comparison of the improvement achieved when adding the same amount of sentences produced by the labeled projection method of Akbik et al. (2015). We see in Table 6 that adding our German data shows improvement in F1 score, despite the fact that the CoNLL-09 label scheme has arguments not seen in our training data (namely A5-A9). Presumably we see this improvement because the frequency of the major roles is more prominent. In the case of French, we don't see significant improvement, however also here the addition of projected data shows a similar trend.

[13]The label distribution is given in the Supplement, A.3.

| Model + Training Data | Data Size | F1 Test |
|---|---|---|
| DE [Mono] *(Original)* | 39K | 61.9 |
| DE [Mono] + *LabelProj* | 83K | 62.37 |
| DE [Mono] + OurGen10K | 49K | 62.4 |
| DE [Mono] + OurGen20K | 59K | 62.46 |
| DE [Mono] + OurGen30K | 69K | 62.81 |
| **DE [Mono] + OurGenALL** | **83K** | **63.57** |
| FR [Mono] *(Original)* | 73K | 70.3 |
| FR [Mono] + *LabelProj* | 105K | 70.45 |
| FR [Mono] + OurGen10K | 83K | 70.33 |
| **FR [Mono] + OurGen20K** | **93K** | **70.52** |
| FR [Mono] + OurGenALL | 105K | 70.39 |

Table 6: We retrain the monolingual systems *DE, FR* using the original training sets (BL: *Original*) shown in Table 1 and inject our generated data in different sizes. We also compare to the stronger baseline *LabelProj* where we add data created by label projection (Akbik et al., 2015).

## 5 Related Work

**Encoder-Decoder Models**. A wide range of NMT models are based on the Encoder-Decoder approach (Sutskever et al., 2014) with attention mechanism (Bahdanau et al., 2015; Luong et al., 2015). More recent architectures (Zoph and Knight, 2016; Firat et al., 2016a) show that training with multiple languages performs better than one-to-one NMT. Multilingual models have also been trained to perform Zero-shot translation (Johnson et al., 2017; Firat et al., 2016b). The Enc-Dec approach has been tested in many tasks that can be formulated as a sequence transduction problem: syntactic parsing (Vinyals et al., 2015), AMR and Semantic Parsing (Konstas et al., 2017; Dong and Lapata, 2016) and SRL (Daza and Frank, 2018). The most similar approach to ours is Zhang et al. (2017), who propose a cross-lingual Enc-Dec that produces OpenIE-annotated English given a Chinese sentence. However, their setup is easier than ours since they have a reliable labeler on the target side, facilitating the generation of more training data unlike us who are interested in labeling the resource-poor language.

**Cross-lingual Annotation Projection.** A common approach to address the lack of annotations is projecting labels from English to a lower-resource language of interest. This has shown good results in the transfer of semantic information to target languages. Kozhevnikov and Titov (2013) propose an unsupervised method to transfer SRL labels to another language by training on the source side and using shared feature representations for predicting on the target side. Padó and Lapata (2009) project FrameNet (Baker et al., 1998) SRL labels by searching for the best alignment in source and target constituent trees, defining label transfer as an optimization problem in a bipartite graph. van der Plas et al. (2011) use intersective word alignments between English and French with additional filtering heuristics to determine whether a PropBank label should be transferred and then use this to train a joint syntactic-semantic parser for both languages. Akbik et al. (2015) proposes a higher-confidence projection by first creating a system with high precision and low recall and then using a bootstrap approach to augment the labeled data.

Separately, Minard et al. (2016) generated a multilingual event and time parallel corpus including SRL annotations. Their corpus was manually annotated on the English side and automatically projected to Italian, Spanish, and Dutch based on the manual alignment of the annotated elements. Unfortunately, the authors do not report the performance of the SRL task, making it difficult for us to use their data for benchmarking.

**Semantic Role Labeling.** Span-based SRL only exists on English data (Zhou and Xu, 2015; He et al., 2018; Strubell et al., 2018; Ouchi et al., 2018). Dependency-based SRL models such as (Marcheggiani and Titov, 2017; Cai et al., 2018; Li et al., 2019) are the state-of-the-art for English. For French, we compare against van der Plas et al. (2014) since we did not find more recent work for that language. Roth and Lapata (2016) show a model based on dependency path embeddings that achieved SOTA in English and German. The Polyglot SRL model of Mulcaire et al. (2018) shows some improvement over monolingual baselines when aggregating all multilingual data available from CoNLL-09, while more refined integration did not show further improvement. Their system does not perform better than our multilingual models for English and German.

## 6 Conclusions

We presented the first cross-lingual SRL system that translates a sentence and concurrently labels it with PropBank roles. The proposed Enc-Dec architecture is flexible: as a *monolingual* system the model achieves SOTA for English PropBank role labeling, the *multilingual* SRL system shows that joining multiple languages improves SRL performance over the monolingual baselines, and a *cross-lingual* system can be used to generate SRL-labeled data for lower-resource languages. Evaluation of the cross-lingual system shows that the quality-filtered sentences are highly grammatical and natural, and that the generated PropBank labels can be more precise than label projection. Using our labeled data beats a label projection baseline when using it to augment the training set of a lower-resource language.

An advantage of our proposed model is that it does not need parallel data at inference time. Our current model can possibly be further improved by adding more automatically generated data in the data augmentation scenario, or by targeted selection in an active learning setting. Current limitations of the system may be alleviated by pretraining the model to acquire better translation knowledge from larger training data, and by developing more refined filtering methods.

In future work we also aim to make the system more flexible, by extending it to few-shot or zero-shot learning, to alleviate the need for an initial big annotated set, and thus to be able to generate SRL data for truly resource-poor languages. Further challenges for this novel architecture are to extend it to joint predicate and role labeling for more than one predicate at a time.

## Acknowledgements

We thank the reviewers for their insightful comments. This research is funded by the Leibniz ScienceCampus Empirical Linguistics & Computational Language Modeling, supported by Leibniz Association grant no. SAS2015-IDS-LWC and by the Ministry of Science, Research, and Art of Baden-Wurttemberg. We thank NVIDIA Corporation for donating GPUs used in this research. We are grateful to our annotators and to Éva Mújdricza-Maydt for her assistance with the human evaluation setup.

## References


Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*, pages 1638–1649.

Alan Akbik, L.b Chiticariu, M.b Danilevsky, Y.b Li, S.b Vaithyanathan, and H.b Zhu. 2015. Generating high quality proposition banks for multilingual semantic role labeling. *ACL-IJCNLP 2015*, 1:397–418.

Alan Akbik, Vishwajeet Kumar, and Yunyao Li. 2016. Towards Semi-Automatic Generation of Proposition Banks for Low-Resource Languages. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 993–998.

Alan Akbik and Roland Vollgraf. 2018. ZAP: An open-source multilingual annotation projection framework. In *Proceedings of the 11th Language Resources and Evaluation Conference*, Miyazaki, Japan. European Language Resource Association.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Collin F. Baker, Charles J. Fillmore, John B. Lowe, Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of the 17th international conference on Computational linguistics*, volume 1, page 86, Morristown, NJ, USA. Association for Computational Linguistics.

Jiaxun Cai, Shexia He, Zuchao Li, and Hai Zhao. 2018. A full end-to-end semantic role labeler, syntactic-agnostic over syntactic-aware? In *COLING*, pages 2753–2765. Association for Computational Linguistics.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the conll-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, CONLL '05, pages 152–164, Stroudsburg, PA, USA. Association for Computational Linguistics.

Angel Daza and Anette Frank. 2018. A sequence-to-sequence model for semantic role labeling. In *Proceedings of The Third Workshop on Representation Learning for NLP*, pages 207–216. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43. Association for Computational Linguistics.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648, Atlanta, Georgia. Association for Computational Linguistics.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016a. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *HLT-NAACL*, pages 866–875. The Association for Computational Linguistics.

Orhan Firat, Baskaran Sankaran, Yaser Al-Onaizan, Fatos T. Yarman Vural, and Kyunghyun Cho. 2016b. Zero-resource translation with multi-lingual neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 268–277, Austin, Texas. Association for Computational Linguistics.

Hagen Fürstenau and Mirella Lapata. 2012. Semi-Supervised Semantic Role Labeling via Structural Alignment. *Journal of Computational Linguistics*, 38(1):135–171.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6, Melbourne, Australia. Association for Computational Linguistics.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany. Association for Computational Linguistics.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu,

Nianwen Xue, and Yi Zhang. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, CoNLL '09, pages 1–18, Stroudsburg, PA, USA. Association for Computational Linguistics.

Silvana Hartmann, Judith Eckle-Kohler, and Iryna Gurevych. 2016. Generating training data for semantic role labeling based on label transfer from linked lexical resources. *Transactions of the Association for Computational Linguistics*, 4:197–213.

Silvana Hartmann, Éva Mújdricza-Maydt, Ilia Kuznetsov, Iryna Gurevych, and Anette Frank. 2017. Assessing SRL frameworks with automatic training data expansion. In *Proceedings of the 11th Linguistic Annotation Workshop*, pages 115–121, Valencia, Spain. Association for Computational Linguistics.

Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2018. Jointly predicting predicates and arguments in neural semantic role labeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 364–369. Association for Computational Linguistics.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and whats next. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Luheng He, Mike Lewis, and Luke Zettlemoyer. 2015. Question-answer driven semantic role labeling: Using natural language to annotate natural language. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 643–653, Lisbon, Portugal. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Melvin Johnson, Mike Schuster, Quoc Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernand a Vigas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Atif Khan, Naomie Salim, and Yogan Jaya Kumar. 2015. A framework for multi-document abstractive summarization based on semantic role labelling. *Appl. Soft Comput.*, 30(C):737–747.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada. Association for Computational Linguistics.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: Sequence-to-Sequence Models for Parsing and Generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 146–157, Vancouver, Canada.

Mikhail Kozhevnikov and Ivan Titov. 2013. Cross-lingual transfer of semantic role labeling models. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1190–1200. Association for Computational Linguistics.

Klaus Krippendorff. 1980. *Content Analysis: An Introduction to Its Methodology*. Sage.

Zuchao Li, Shexia He, Hai Zhao, Yiqing Zhang, Zhuosheng Zhang, Xi Zhou, and Xiaoping Zhou. 2019. Dependency or span, end-to-end uniform semantic role labeling. In *AAAI*.

Jiangming Liu, Shay B. Cohen, and Mirella Lapata. 2018. Discourse representation structure parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 429–439, Melbourne, Australia. Association for Computational Linguistics.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1412–1421.

Diego Marcheggiani, Joost Bastings, and Ivan Titov. 2018. Exploiting semantics in neural machine translation with graph convolutional networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 486–492, New Orleans, Louisiana. Association for Computational Linguistics.

Diego Marcheggiani, Anton Frolov, and Ivan Titov. 2017. A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 411–420, Vancouver, Canada. Association for Computational Linguistics.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017*

*Conference on Empirical Methods in Natural Language Processing*, pages 1506–1515. Association for Computational Linguistics.

Todor Mihaylov and Anette Frank. 2019. Discourse-Aware Semantic Self-Attention For Narrative Reading Comprehension. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019*.

Anne-Lyse Minard, Manuela Speranza, Ruben Urizar, Begoña Altuna, Marieke van Erp, Anneleen Schoen, and Chantal van Son. 2016. MEANTIME, the NewsReader Multilingual Event and Time Corpus. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 4417–4422, Portorož, Slovenia. European Language Resources Association (ELRA).

Phoebe Mulcaire, Swabha Swayamdipta, and Noah A. Smith. 2018. Polyglot semantic role labeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 667–672, Melbourne, Australia. Association for Computational Linguistics.

Hiroki Ouchi, Hiroyuki Shindo, and Yuji Matsumoto. 2018. A span selection model for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 1630–1642.

Sebastian Padó and Mirella Lapata. 2009. Cross-lingual annotation projection of semantic roles. *J. Artif. Int. Res.*, 36(1):307–340.

Martha Palmer, Paul Kingsbury, and Daniel Gildea. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *EMNLP*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the NAACL-HLT, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Lonneke van der Plas, Marianna Apidianaki, and Chenhua Chen. 2014. Global methods for cross-lingual semantic role and predicate labelling. In *COLING*.

Lonneke van der Plas, Paola Merlo, and James Henderson. 2011. Scaling up automatic cross-lingual semantic role annotation. In *ACL (Short Papers)*, pages 299–304. The Association for Computer Linguistics.

Lonneke van der Plas, Tanja Samardzic, and Paola Merlo. 2010. Cross-lingual validity of propbank in the manual annotation of french. In *Linguistic Annotation Workshop*, pages 113–117. Association for Computational Linguistics.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Comput. Linguist.*, 34(2):257–287.

Michael Roth and Mirella Lapata. 2016. Neural semantic role labeling with dependency path embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1192–1202, Berlin, Germany. Association for Computational Linguistics.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5027–5038, Brussels, Belgium. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3104–3112, Cambridge, MA, USA. MIT Press.

Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Efficient inference and structured learning for semantic role labeling. *Transactions of the Association for Computational Linguistics*, 3:29–41.

Oriol Vinyals, Ł ukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2773–2781. Curran Associates, Inc.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015. A Transition-based Algorithm for AMR Parsing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 366–375. Association for Computational Linguistics.

Sheng Zhang, Kevin Duh, and Benjamin Van Durme. 2017. MT/IE: Cross-lingual open information extraction with neural sequence-to-sequence models.

In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, Valencia, Spain. Association for Computational Linguistics.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1127–1137, Beijing, China. Association for Computational Linguistics.

Micha Ziemski, Marcin Junczys-Dowmunt, and Bruno Pouliquen. 2016. The united nations parallel corpus v1.0. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Barret Zoph and Kevin Knight. 2016. Multi-source neural translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 30–34, San Diego, California. Association for Computational Linguistics.

# Syntax-Enhanced Self-Attention-Based Semantic Role Labeling

**Yue Zhang, Rui Wang, Luo Si**
Alibaba Group, China
{shiyu.zy, masi.wr, luo.si}@alibaba-inc.com

## Abstract

As a fundamental NLP task, semantic role labeling (SRL) aims to discover the semantic roles for each predicate within one sentence. This paper investigates how to incorporate syntactic knowledge into the SRL task effectively. We present different approaches of encoding the syntactic information derived from dependency trees of different quality and representations; we propose a syntax-enhanced self-attention model and compare it with other two strong baseline methods; and we conduct experiments with newly published deep contextualized word representations as well. The experiment results demonstrate that with proper incorporation of the high quality syntactic information, our model achieves a new state-of-the-art performance for the Chinese SRL task on the CoNLL-2009 dataset.

## 1 Introduction

The task of semantic role labeling (SRL) is to recognize arguments for a given predicate in one sentence and assign labels to them, including "who" did "what" to "whom", "when", "where", etc. Figure 1 is an example sentence with both semantic roles and syntactic dependencies. Since the nature of semantic roles is more abstract than the syntactic dependencies, SRL has a wide range of applications in different areas, e.g., text classification (Sinoara et al., 2016), text summarization (Genest and Lapalme, 2011; Khan et al., 2015), recognizing textual entailment (Burchardt et al., 2007; Stern and Dagan, 2014), information extraction (Surdeanu et al., 2003), question answering (Shen and Lapata, 2007; Yih et al., 2016), and so on.

Traditionally, syntax is the bridge to reach semantics. However, along with the popularity of the end-to-end models in the NLP community, various recent studies have been discussing the necessity of syntax in the context of SRL. For instance, He et al. (2017) have observed that only good syntax helps with the SRL performance. Xia et al. (2019) have explored what kind of syntactic information or structure is better suited for the SRL model. Cai et al. (2018) have compared syntax-agnostic and syntax-aware approaches and claim that the syntax-agnostic model surpasses the syntax-aware ones.

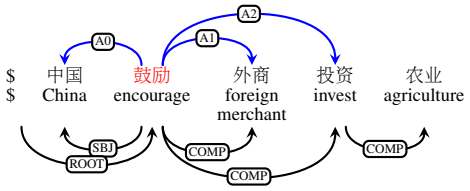


Figure 1: An example of one sentence with its syntactic dependency tree and semantic roles. Arcs above the sentence are semantic role annotations for the predicate "鼓励 (encourage)" and below the sentence are syntactic dependency annotations of the whole sentence. The meaning of this sentence is "China encourages foreign merchants to invest in agriculture".

In this paper, we focus on analyzing the relationship between the syntactic dependency information and the SRL performance. In particular, we investigate the following four aspects: 1) *Quality* of the syntactic information: whether the performance of the syntactic parser output affects the SRL performance; 2) *Representation* of the syntactic information: how to represent the syntactic dependencies to better preserve the original structural information; 3) *Incorporation* of the syntactic information: at which layer of the SRL model and how to incorporate the syntactic information; and 4) the *Relationship* with other external resources: when we append other external resources into the SRL model, whether their contributions are orthogonal to the syntactic dependencies.

For the main architecture of the SRL model, many neural-network-based models use BiLSTM



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 616–626, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

as the encoder (e.g., Cai et al. (2018); Li et al. (2018); He et al. (2018)), while recently self-attention-based encoder becomes popular due to both the effectiveness and the efficiency (Vaswani et al., 2017; Tan et al., 2017; Strubell et al., 2018). By its nature, the self-attention-based model directly captures the relation between words in the sentence, which is convenient to incorporate syntactic dependency information. Strubell et al. (2018) replace one attention head with pre-trained syntactic dependency information, which can be viewed as a *hard* way to inject syntax into the neural model. Enlightened by the machine translation model proposed by Shaw et al. (2018), we introduce the *Relation-Aware* method to incorporate syntactic dependencies, which is a *softer* way to encode richer structural information.

Various experiments for the Chinese SRL on the CoNLL-2009 dataset are conducted to evaluate our hypotheses. From the empirical results, we observe that: 1) The quality of the syntactic information is essential when we incorporate structural information into the SRL model; 2) Deeper integration of the syntactic information achieves better results than the simple concatenation to the inputs; 3) External pre-trained contextualized word representations help to boost the SRL performance further, which is not entirely overlapping with the syntactic information.

In summary, the contributions of our work are:

- We present detailed experiments on different aspects of incorporating syntactic information into the SRL model, in what quality, in which representation and how to integrate.
- We introduce the relation-aware approach to employ syntactic dependencies into the self-attention-based SRL model.
- We compare our approach with previous studies, and achieve state-of-the-art results with and without external resources, i.e., in the so-called *closed* and *open* settings.

## 2 Related work

Traditional semantic role labeling task (Gildea and Jurafsky, 2002) presumes that the syntactic structure of the sentence is given, either being a constituent tree or a dependency tree, like in the CoNLL shared tasks (Carreras and Màrquez, 2005; Surdeanu et al., 2008; Hajič et al., 2009). Recent neural-network-based approaches can be roughly categorized into two classes: 1) making use of the syntactic information (FitzGerald et al., 2015; Roth and Lapata, 2016; Qian et al., 2017; Marcheggiani and Titov, 2017), and 2) pure end-to-end learning from tokens to semantic labels, e.g., Zhou and Xu (2015); Marcheggiani et al. (2017).

Roth and Lapata (2016) utilize an LSTM model to obtain embeddings from the syntactic dependency paths; while Marcheggiani and Titov (2017) construct Graph Convolutional Networks to encode the dependency structure. Although He et al. (2017)'s approach is a pure end-to-end learning, they have included an analysis of adding syntactic dependency information into English SRL in the discussion section. Cai et al. (2018) have compared syntax-agnostic and syntax-aware approaches and Xia et al. (2019) have compared different ways to represent and encode the syntactic knowledge.

In another line of research, Tan et al. (2017) utilize the Transformer network for the encoder instead of the BiLSTM. Strubell et al. (2018) present a novel and effective multi-head self-attention model to incorporate syntax, which is called LISA (Linguistically-Informed Self-Attention). We follow their approach of replacing one attention head with the dependency head information, but use a softer way to capture the pairwise relationship between input elements (Shaw et al., 2018).

For the datasets and annotations of the SRL task, most of the previous research focuses on 1) PropBank (Palmer et al., 2005) and NomBank (Meyers et al., 2004) annotations, i.e., the CoNLL 2005 (Carreras and Màrquez, 2005) and CoNLL 2009 (Hajič et al., 2009) shared tasks; 2) OntoNotes annotations (Weischedel et al., 2011), i.e., the CoNLL 2005 and CoNLL 2012 datasets and more; 3) and FrameNet (Baker et al., 1998) annotations. For the non-English languages, not all of them are widely available. Apart from these, in the broad range of semantic processing, other formalisms non-exhaustively include abstract meaning representation (Banarescu et al., 2013), universal decompositional semantics (White et al., 2016), and semantic dependency parsing (Oepen et al., 2015). Abend and Rappoport (2017) give a better overview of various semantic representations. In this paper, we primarily work on the Chinese and English datasets from the

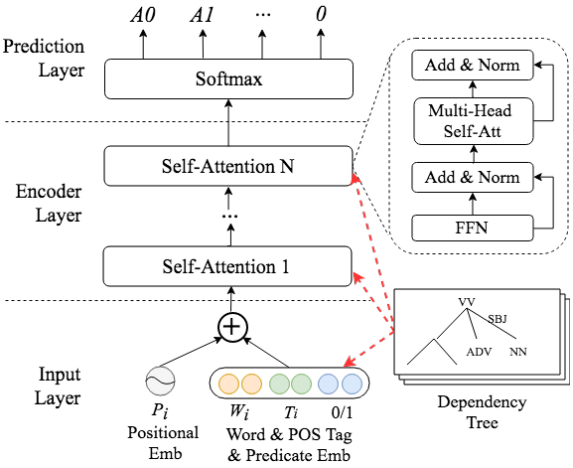


Figure 2: Architecture of our syntax-enhanced self-attention-based SRL model. Red dotted arrows indicate different locations where we incorporate linguistic knowledge in different forms. The dotted box on the upper right is the detailed composition of the self-attention block.

CoNLL-2009 shared task and focus on the effectiveness of incorporating syntax into the Chinese SRL task.

## 3 Approaches

In this section, we first introduce the basic architecture of our self-attention-based SRL model, and then present two different ways to encode the syntactic dependency information. Afterwards, we compare three approaches to incorporate the syntax into the base model, concatenation to the input embedding, LISA, and our proposed relation-aware method.

### 3.1 The Basic Architecture

Our basic model is a multi-head self-attention-based model, which is effective in SRL task as previous work proves (Tan et al., 2018). The model consists of three layers: the input layer, the encoder layer and the prediction layer as shown in Figure 2.

#### 3.1.1 Input Layer

The input layer contains three types of embeddings: token embedding, predicate embedding, and positional embedding.

**Token Embedding** includes word embedding, part-of-speech (POS) tag embedding.

**Predicate Embedding** has been proposed by He et al. (2017), and its binary embedding is used to indicate the predicates indices in each sentence.

**Positional Embedding** encodes the order of the input word sequence. We follow Vaswani et al. (2017) to use time positional embedding, which is formulated as follows:

$$\begin{aligned} PE(t, 2i) &= sin(t/10000^{2i/d}) \\ PE(t, 2i+1) &= cos(t/10000^{2i/d}) \end{aligned} \quad (1)$$

where $t$ is the position, $i$ means the dimension, and $d$ is the dimension of the model input embedding.

#### 3.1.2 Encoder Layer

The self-attention block is almost the same as Transformer encoder proposed by Vaswani et al. (2017). Specifically the Transformer encoder contains a feed-forward network (**FFN**) and a multi-head attention network. The former is followed by the latter. In this work, we exchange their order, so that the multi-head attention module is moved behind the FFN module[1] as Figure 2 shows.

**FFN** The FFN module consists of two affine layers with a ReLU activation in the middle. Formally, we have the following equation:

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2 \quad (2)$$

**Multi-Head Attention** The basic attention mechanism used in the multi-head attention function is called "Scaled Dot-Product Attention", which is formulated as follows:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \quad (3)$$

where $Q$ is queries, $K$ is keys, and $V$ is values.

In the multi-head attention setting, it first maps the input matrix $X$ into queries, keys and values matrices by using $h$ different learned linear projections. Taking queries $Q$ as an example:

$$Linear_i^Q(X) = XW_i^Q + b_i^Q \quad (4)$$

where $0 \leq i < h$. Keys and values use similar projections.

On each of these projections, we perform the scaled dot-product attention in parallel. These parallel output values are concatenated and once again projected into the final values. Equation 5 depicts the above operations.

$$MultiHead(X) = Concat(head_1, \ldots, head_h)W^O \quad (5)$$

[1]Changing the order delivers better empirical results (Tan et al., 2018) and our experiments show the same trend. The details of the experiments are not listed in this paper.

where

$$head_i = Attention(Linear_i^Q(X), Linear_i^K(X), Linear_i^V(X)) \quad (6)$$

More details about multi-head attention can be found in Vaswani et al. (2017).

**Add & Norm** We employ a residual connection to each module, followed by a layer normalization (Ba et al., 2016) operation. The output of each module is formulated as

$$x = LayerNorm(x + f(x)) \quad (7)$$

where $f(x)$ is implemented by each above module.

## 3.2 Representation of the Syntactic Dependencies

### 3.2.1 Dependency Head & Relation

The most intuitive way to represent syntactic information is to use individual dependency relations directly, like dependency head and dependency relation label, denoted as DEP and REL for short.

Except for LISA, where DEP is a one-hot matrix of dependency head word index described in 3.3.2, in other cases, we use the corresponding head word. REL is the dependency relation between the word and its syntactic head. We take both DEP and REL as common strings and map them into dense vectors in the similar way of word embedding.

### 3.2.2 Dependency Path & Relation Path

In order to preserve the structural information of dependency trees as much as possible, we take the syntactic path between candidate arguments and predicates in dependency trees as linguistic knowledge. Referring to Xia et al. (2019), we use the Tree-based Position Feature (TPF) as Dependency Path (DEPPATH) and use the Shortest Dependency Path (SDP) as Relation Path (RELPATH).

To generate DEPPATH & RELPATH between candidate argument and predicate, we firstly find their lowest common ancestor. Then we get two sub-paths, one is from the ancestor to the predicate and the other is from the ancestor to the argument. For DEPPATH, we compute distance from ancestor to predicate and argument respectively and then concatenate two distances with the separator ‘,’.

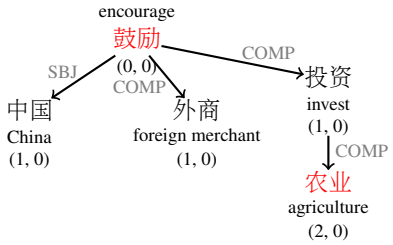


Figure 3: The syntactic dependency tree of the sentence “中国鼓励外商投资农业” (China encourages foreign merchants to invest in agriculture). Numbers in brackets are the DEPPATH for each candidate argument with the predicate “鼓励 (encourage)”. Light grey labels on the arcs are the syntactic dependency labels.

For RELPATH, we concatenate the labels appearing in each sub-path with the separator “_” respectively to get two label paths, and then concatenate the two label paths with the separator ‘,’.

As shown in Figure 3, the lowest common ancestor of the predicate “鼓励 (encourage)” and the candidate argument “农业 (agriculture)” is “鼓励 (encourage)”, so their DEPPATH is “2,0” and its RELPATH is “COMP_COMP,”[2].

We take both DEPPATH and RELPATH as common strings and map them into dense vectors in the similar way of DEP and REL.

## 3.3 Incorporation Methods

### 3.3.1 Input Embedding Concatenation

To incorporate syntactic knowledge, one simple method is to take it as part of the neural network input, denoted as INPUT. We represent the syntactic information with dense vectors, and concatenate it with other information like word embedding:

$$input = E_W \oplus E_S. \quad (8)$$

where $\oplus$ means concatenation; $E_W$ means the original inputs of the neural model and $E_S$ means the embedding of syntax information, such as DEP/REL or DEPPATH/RELPATH.

### 3.3.2 LISA

Strubell et al. (2018) propose the linguistically-informed self-attention model (LISA for short) to combine SRL and dependency parsing as multi-task learning in a subtle way. Based on the multi-head self-attention model, LISA uses one attention head to predict the dependency results and it

[2] When predicate is the ancestor of argument, sub-path from ancestor to predicate is none. We use ‘0’ to represent distance and use empty string to represent the label path, and vice versa.

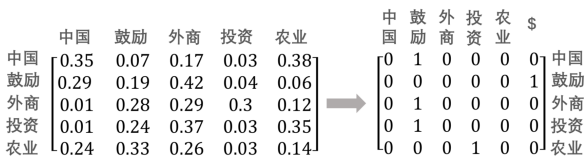


Figure 4: Attention matrix of the replaced attention head in the LISA model. The left matrix is the original $softmax$ attention, and the right is a one-hot matrix copied from the syntactic dependency head results.

can also directly use pre-trained dependency head results to replace the attention matrix during testing.

Being different from their multi-task learning, we make the replacement of one attention head during both training and testing. Instead of the original $softmax$ attention matrix, we use a one-hot matrix, generated by mapping the dependency head index of each word into a 0-1 vector of the sentence length as Figure 4 shows.

We add the dependency relation information with $V$ in the replaced head so that we can make full use of the syntactic knowledge. The replaced attention head is formulated as follows:

$$Attention(Q, K, V) = M_D\ (V \oplus E_R) \quad (9)$$

where $M_D$ is the one-hot dependency head matrix and $E_R$ means the embedding of dependency relation information, such as REL or RELPATH.

#### 3.3.3 Relation-Aware Self-Attention

Relation-aware self-attention model (RELAWE for brevity) incorporates external information into the attention. By this way, the model considers the pairwise relationships between input elements, which highly agrees with the task of SRL, i.e., aiming to find the semantic relations between the candidate argument and predicate in one sentence.

Compared to the standard attention, in this paper, we add the dependency information into $Q$ and $V$ in each attention head, like equation (10) shows:

$$Attention(Q, K, V) = softmax(\frac{(Q + E_D + E_R)K^T}{\sqrt{d_k}})(V + E_D + E_R) \quad (10)$$

where $E_D$ and $E_R$ mean the syntactic dependency head and relation information respectively. For our multi-layer multi-head self-attention model, we make this change to each head of the first $N$ self-attention layers.

| | dev | | test | |
|---|---|---|---|---|
| | **UAS** | **LAS** | **UAS** | **LAS** |
| AUTO | 80.50 | 78.34 | 80.70 | 78.46 |
| BIAFFINE | 89.00 | 85.86 | 89.05 | 85.60 |
| BIAFFINEBERT | 91.76 | 89.08 | 92.14 | 89.23 |

Table 1: Syntactic dependency performance for different parsers. AUTO indicates the automatic dependency trees provided by the CoNLL-09 Chinese dataset. BIAFFINE means the trees are generated by BiaffineParser with pre-trained word embedding on the Gigaword corpus while BIAFFINEBERT is the same parser with BERT. We use the labeled accuracy score (LAS) and unlabeled accuracy score (UAS) to measure the quality of syntactic dependency trees.

## 4 Experiment

### 4.1 Settings

**Datasets & Evaluation Metrics** Our experiments are conducted on the CoNLL-2009 shared task dataset (Hajič et al., 2009). We use the official evaluation script to compare the output of different system configurations, and report the labeled precision (P), labeled recall (R) and labeled f-score (F1) for the semantic dependencies.

**Word Representations** Most of our experiments are conducted in the *closed* setting without any external word embeddings or data resources than those provided by the CoNLL-2009 datasets. In the *closed* setting, word embedding is initialized by a Gaussian distribution with mean 0 and variance $\frac{1}{\sqrt{d}}$, where $d$ is the dimension of embedding size of each layer.

For the experiments with external resources in the *open* setting, we utilize 1) word embeddings pre-trained with GloVe (Pennington et al., 2014) on the Gigaword corpus for Chinese and the published embeddings with 100 dimensions pre-trained on Wikipedia and Gigaword for English; and 2) ELMo[3] (Peters et al., 2018) and BERT[4] (Devlin et al., 2018), two recently proposed effective deep contextualized word representations[5].

[3]We use the released model on their website: https://github.com/allenai/allennlp/blob/master/tutorials/how_to/elmo.md

[4]We generate our pre-trained BERT embedding with the released model in https://github.com/google-research/bert. The model uses character-based tokenization for Chinese, which require us to maintain alignment between our input text and output text of Bert. So we take take embedding of the first word piece as the whole word representation.

[5]In *open* setting, we use pre-trained word embedding instead of random initialized embedding. If using BERT and Elmo additionally, we project them into representation vectors of same dimension as word embedding and concatenate

Other embeddings, i.e., POS embedding, linguistic knowledge embedding, and so on are initialized in same way as random word embedding no matter in *closed* or *open* setting.

**Syntactic Parsers** In Table 1, both AUTO and GOLD syntactic dependencies are provided by the dataset. Since the performance of the AUTO is far behind the state-of-the-art BiaffineParser (Dozat and Manning, 2016), we generate more dependency results by training BiaffineParser[6] with different external knowledge, including pre-trained word embedding and BERT. Performance for different parsers is listed in Table 1.

**Parameters** In this work, we set word embedding size $d_w = 100$, POS embedding size $d_t = 50$. The predicate embedding size is set as $d_p = 100$. The syntax-related embedding size varies along with different configurations, so as the feature embedding size $d_f$.

To facilitate residual connections, all sublayers in the model produce outputs of dimension $d_{model} = d_f + d_p$. The hidden dimension $d_{ff} = 800$ is applied for all the experiments. We set the number of shared self-attention blocks $N = 10$. The number of heads varies with $d_{model}$, but dimension of each head is 25. Besides, LISA incorporates syntax knowledge in the 5-th self-attention layer while RELAWE incorporates in the first 5 layers.

We apply the similar dropout strategy as Vaswani et al. (2017), i.e., the attention and residual dropout values are 0.2 and 0.3 respectively. The dropout is also applied in the middle layer of FFN with value 0.2. We also employ label smoothing (Szegedy et al., 2015) of value 0.1 during training.

We use softmax-cross-entropy as our loss function, and use the Adadelta optimizer (Zeiler, 2012) with $\epsilon = 10^{-6}$ and $\rho = 0.95$. For all experiments, we train the model $200,000$ steps with learning rate $lr = 1.0$, and each batch has 4096 words.

All the hyper-parameters are tuned on the development set.

**Configurations** We use different abbreviations to represent the parsing results, syntactic dependency representations, and incorporation methods. All the system configurations in our experiments are listed in Table 2.

them with other input.

[6]We split train data into 5-fold, and train model with 4-fold to generate automatic trees of the left 1-fold train data.

| Abbreviation | Description |
|---|---|
| NONE | No syntactic knowledge |
| **Syn. Parser** | |
| AUTO | Parsing result from CoNLL09 |
| AUTODEL | Remove syntax errors of AUTO |
| BIAFFINE | BiaffineParser result |
| BIAFFINEBERT | BiaffineParser with BERT |
| GOLD | Gold syntax from CoNLL09 |
| **Syn. Representation** | |
| DEP | Dependency head |
| REL | Dependency relation label |
| DEPPATH | Tree-based position feature |
| RELPATH | Shortest relation label path |
| **Incorporation** | |
| INPUT | Add to the input embedding |
| LISA | From Strubell et al. (2018) |
| RELAWE | Relation-aware self-attention |

Table 2: A glossary of abbreviations for different system configurations in our experiments.

| | P | R | F1 |
|---|---|---|---|
| NONE | 83.97 | 82.94 | 83.45 |
| AUTO | 85.92 | 84.62 | 85.26 |
| BIAFFINE | 86.04 | 85.43 | 85.73 |
| BIAFFINEBERT | 87.32 | 87.10 | **87.21** |
| AUTODEL | 88.33 | 87.67 | 88.00 |
| GOLD | 91.00 | 91.43 | 91.22 |

Table 3: SRL results with dependency trees of different quality on the Chinese dev set. These experiments are conducted on the RELAWE model with DEP&REL representations.

### 4.2 Quality of the Syntactic Dependencies

We use the above-mentioned dependency trees of different quality for comparison, with DEP&REL representation on our RELAWE model. In addition, we generate one more data AUTODEL by deleting all the erroneous dependency heads and relations from the provided AUTO data according to the gold heads and relations, and we do not replace them with any alternative heads and relations[7]. We take this setting as another reference (along with GOLD) to indicate that erroneous syntax information may hurt the performance of the SRL model. We take the GOLD as the upperbound reference of our task setting. Experiment results in Table 3 demonstrate that, incorporating syntactic knowledge into the SRL model can achieve better performance and overall, the better the quality is, the better the SRL model performs. This is consistent with the previous study by He et al. (2017) on the English dataset.

[7]For the AUTODEL data, we cannot guarantee that there exists a syntactic path between the two words. Therefore, we do not conduct experiments under the DepPath&RelPath setting.

| | P | R | F1 |
|---|---|---|---|
| **AUTO** | | | |
| DEPPATH&RELPATH | 84.76 | 81.85 | 83.28 |
| **BIAFFINE** | | | |
| DEP | 84.33 | 84.47 | 84.40 |
| REL | 85.84 | 85.23 | 85.54 |
| DEP&REL | 86.04 | 85.43 | 85.73 |
| DEPPATH | 85.48 | 84.17 | 84.82 |
| RELPATH | 86.85 | 83.92 | 85.36 |
| DEPPATH&RELPATH | 86.40 | 85.52 | **85.96** |
| **GOLD** | | | |
| DEPPATH&RELPATH | 92.20 | 92.53 | 92.37 |

Table 4: SRL results with different syntactic representations on the Chinese dev set. Experiments are conducted on the RELAWE method.

Closer observation reveals two additional interesting phenomena. Firstly, SRL performance improvement is not proportionate to the improvement of dependency quality. When switching syntactic dependency trees from AUTO to BIAFFINE, SRL performance improves 0.5%, although syntactic dependency improves about 8%. In contrast, the difference between BIAFFINE and BIAFFINEBERT shows more significant improvement of 1.5%. The possible reason is that BIAFFINEBERT provides key dependency information which is missing in other configurations. Secondly, the SRL performance gap between AUTODEL and AUTO is large though they provide the same correct syntactic information. This may indicate that incorporating erroneous syntactic knowledge hurts the SRL model, and even providing more correct dependencies cannot make up for the harm (cf. BIAFFINEBERT).

### 4.3 Representation of the Syntactic Dependencies

Apart from DEP and REL, we also use DEPPATH and RELPATH to encode the syntactic knowledge. In this subsection, we conduct experiments to compare different syntactic encoding in our SRL model. We base the experiments on our RELAWE model, since it is easier to incorporate different representations for comparison. When generating the RELPATH, we filter the paths 1) when the dependency distance between the predicate and the candidate argument is more than 4, and 2) when the RELPATH's frequency is less than 10[8].

No matter in which representation, dependency label information is more important than the head and the combination of the two achieves better performance as our experiment results in Table 4 show. Furthermore, using BIAFFINE dependency trees, DEPPATH and RELPATH perform better than DEP and REL. This is because of the capability of DEPPATH and RELPATH to capture more structural information of the dependency trees.

Comparing Table 3 and 4, when using gold dependencies, DEPPATH&RELPATH can achieve much better result than DEP&REL. But with the AUTO trees, DEPPATH&RELPATH is much worse. Therefore, structural information is much more sensitive to the quality of dependency trees due to error propagation.

[8]As we know, dependency parsers cannot deal with long distance dependency well and it is unlikely to deliver a reliable result. And our experiments show that filtration achieves empirically better results.

| | P | R | F1 |
|---|---|---|---|
| **INPUT** | | | |
| DEP | 83.89 | 83.61 | 83.75 |
| DEP&REL | 86.21 | 85.00 | 85.60 |
| DEP&RELPATH | 86.01 | 85.38 | 85.69 |
| DEPPATH&RELPATH | 85.84 | 85.54 | **85.69** |
| **LISA** | | | |
| DEP | 84.68 | 85.38 | 85.03 |
| DEP&REL | 85.56 | 85.89 | 85.73 |
| DEP&RELPATH | 85.84 | 85.64 | **85.74** |
| DEPPATH&RELPATH[9] | na | na | na |
| **RELAWE** | | | |
| DEP | 84.33 | 84.47 | 84.40 |
| DEP&REL | 86.04 | 85.43 | 85.73 |
| DEP&RELPATH | 86.21 | 85.01 | 85.60 |
| DEPPATH&RELPATH | 86.40 | 85.52 | **85.96** |

Table 5: SRL results with different incorporation methods of the syntactic information on the Chinese dev set. Experiments are conducted on the BIAFFINE parsing results.

| | P | R | F1 |
|---|---|---|---|
| **BIAFFINE** | | | |
| RANDOM | 86.40 | 85.52 | 85.96 |
| GIGA | 86.73 | 85.58 | 86.15 |
| ELMO | 86.84 | 86.74 | 86.79 |
| BERT | 88.16 | 89.57 | **88.86** |
| **BIAFFINEBERT** | | | |
| BERT | 88.05 | 89.65 | 88.84 |

Table 6: SRL results with different external knowledge on the Chinese dev set. We use the RELAWE model and DEPPATH&RELPATH syntax representation.

### 4.4 Incorporation Methods

This subsection discusses the effectiveness of different incorporation methods of the syntactic knowledge. We take BIAFFINE's output as our de-

[9]From the mechanism of LISA, we can find that the replaced attention head can't copy the syntactic dependency heads from DEPPATH.

| Chinese | P | R | F1 |
|---|---|---|---|
| **NONE** | 81.99 | 80.65 | 81.31 |
| **Closed** | | | |
| CoNLL09 SRL Only | na | na | 78.6 |
| INPUT(DEPPATH&RELPATH) | 84.19 | 83.65 | 83.92 |
| LISA(DEP&RELPATH) | 83.84 | 83.54 | 83.69 |
| RELAWE(DEPPATH&RELPATH) | 84.77 | 83.68 | **84.22** |
| **Open** | | | |
| Marcheggiani and Titov (2017) | na | na | 82.5 |
| Cai et al. (2018) | 84.7 | 84.0 | 84.3 |
| INPUT(DEPPATH&RELPATH) + BERT | 86.89 | 87.75 | 87.32 |
| LISA(DEP&RELPATH) + BERT | 86.45 | 87.90 | 87.17 |
| RELAWE(DEPPATH&RELPATH) + BERT | 86.73 | 87.98 | **87.35** |
| **GOLD** | 91.93 | 92.36 | 92.14 |

Table 7: SRL results on the Chinese test set. We choose the best settings for each configuration of our model.

pendency information for the comparison.

Firstly, results in Table 5 show that with little dependency information (DEP), LISA performs better, while incorporating richer syntactic knowledge (DEP&REL or DEP&RELPATH), three methods achieve similar performance. Overall, RELAWE achieves best results given enough syntactic knowledge.

Secondly, INPUT and LISA achieve much better performance when we combine the dependency head information and the relation, while Strubell et al. (2018) have not introduced relation information to the LISA model and Xia et al. (2019) have not combined the head and relation information either. Our proposed RELAWE method with DEPPATH&RELPATH representation performs the best, which encodes the richest syntactic knowledge.

Lastly, under the same settings, LISA and RELAWE perform better than INPUT, which indicates the importance of the location where the model incorporates the syntax, the input layer vs. the encoder layer.

### 4.5 External Resources

Apart from the experiments with syntactic knowledge itself, we also compare different external resources to discover their relationship with the syntax, including pre-trained word embeddings, ELMo, and BERT. We conduct experiments with our best setting, the RELAWE model with DEPPATH & RELPATH and the results are listed in Table 6.

The plain word embedding improves a little in such settings with syntactic information, while for the newly proposed ELMO and BERT, both of them can boost the models further.

### 4.6 Final Results on the Chinese Test Data

Based on the above experiments and analyses, we present the overall results of our model in this subsection. We train the three models (INPUT, LISA, and RELAWE) with their best settings without any external knowledge as CLOSED, and we take the same models with BERT as OPEN. The DEPPATH&RELPATH from GOLD without external knowledge serves as the GOLD for reference. Since we have been focusing on the task of argument identification and labeling, for both CLOSED and OPEN, we follow Roth and Lapata (2016) to use existing systems' predicate senses (Johansson and Nugues, 2008) to exclude them from comparison.

Table 7 shows that our OPEN model achieves more than 3 points of f1-score than the state-of-the-art result, and RELAWE with DEPPATH&RELPATH achieves the best in both CLOSED and OPEN settings. Notice that our best CLOSED model can almost perform as well as the state-of-the-art model while the latter utilizes pre-trained word embeddings. Besides, performance gap between three models under OPEN setting is very small. It indicates that the representation ability of BERT is so powerful and may contains rich syntactic information. At last, the GOLD result is much higher than the other models, indicating that there is still large space for improvement for this task.

| English | P | R | F1 |
|---|---|---|---|
| LISA(DEP)[10] | 89.26 | 85.46 | 87.32 |
| LISA(DEP&RELPATH) | 89.74 | 85.38 | 87.51 |
| INPUT(DEPPATH) | 89.33 | 85.60 | 87.42 |
| INPUT(DEPPATH&RELPATH) | 89.70 | 85.48 | 87.54 |
| RELAWE(DEPPATH&RELPATH) | 89.55 | 85.92 | **87.70** |

Table 8: SRL results on the English test set. We use syntactic dependency results generated by BiaffineParser (On test set, syntactic performance is: UAS = 94.35%, and LAS = 92.54%, which improves about 6% compared to automatic trees in CoNLL-2009.).

### 4.7 Results on the English Data

We also conduct several experiments on the English dataset to validate the effectiveness of our approaches on other languages than Chinese and the results are in Table 8. Although both configurations are not exactly the same as their original papers, we tried our best to reproduce their methods on the CoNLL2009 dataset for our comparison. Overall, the results are consistent with the Chinese experiments, while the improvement is not as large as the Chinese counterparts. The RELAWE model with DEPPATH&RELPATH still achieves the best performance. Applying our syntax-enhanced model to more languages will be an interesting research direction to work on in the future.

## 5 Conclusion and Future Work

This paper investigates how to incorporate syntactic dependency information into semantic role labeling in depth. Firstly, we confirm that dependency trees of better quality are more helpful for the SRL task. Secondly, we present different ways to encode the trees and the experiments show that keeping more (correct) structural information during encoding improves the SRL performance. Thirdly, we compare three incorporation methods and discover that our proposed relation-aware self-attention-based model is the most effective one.

Although our experiments are primarily on the Chinese dataset, the approach is largely language independent. Apart from our tentative experiments on the English dataset, applying the approach to other languages will be an interesting research direction to work on in the future.

[10]We reimplement LISA in Strubell et al. (2018) as LISA(DEP), and Xia et al. (2019)'s best DEPPATH approach as INPUT(DEPPATH). Therefore, we can compare with their work as fairly as possible. Other settings are the best configurations for their corresponding methods.

## References


Omri Abend and Ari Rappoport. 2017. The state of the art in semantic representation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 77–89, Vancouver, Canada. Association for Computational Linguistics.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. abs/1607.06450.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pages 86–90, Stroudsburg, PA, USA. Association for Computational Linguistics.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria. Association for Computational Linguistics.

Aljoscha Burchardt, Nils Reiter, Stefan Thater, and Anette Frank. 2007. A semantic approach to textual entailment: System evaluation and task analysis. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, RTE '07, pages 10–15, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jiaxun Cai, Shexia He, Zuchao Li, and Hai Zhao. 2018. A full end-to-end semantic role labeler, syntax-agnostic over syntax-aware? *CoRR*, abs/1808.03815.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the conll-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 152–164, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Timothy Dozat and Christopher D. Manning. 2016. Deep biaffine attention for neural dependency parsing. *CoRR*, abs/1611.01734.

Nicholas FitzGerald, Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic role labeling with neural network factors. In *Proceedings of the 2015 Conference on Empirical Methods in*

*Natural Language Processing*, pages 960–970, Lisbon, Portugal. Association for Computational Linguistics.

Pierre-Etienne Genest and Guy Lapalme. 2011. Framework for abstractive summarization using text-to-text generation. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 64–73, Portland, Oregon. Association for Computational Linguistics.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Comput. Linguist.*, 28(3):245–288.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 1–18, Boulder, Colorado. Association for Computational Linguistics.

Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2018. Jointly predicting predicates and arguments in neural semantic role labeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 364–369.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 473–483. Association for Computational Linguistics.

Richard Johansson and Pierre Nugues. 2008. The effect of syntactic representation on semantic role labeling. In *COLING 2008, 22nd International Conference on Computational Linguistics, Proceedings of the Conference, 18-22 August 2008, Manchester, UK*, pages 393–400.

Atif Khan, Naomie Salim, and Yogan Jaya Kumar. 2015. A framework for multi-document abstractive summarization based on semantic role labelling. *Appl. Soft Comput.*, 30(C):737–747.

Zuchao Li, Shexia He, Jiaxun Cai, Zhuosheng Zhang, Hai Zhao, Gongshen Liu, Linlin Li, and Luo Si. 2018. A unified syntax-aware framework for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2401–2411.

Diego Marcheggiani, Anton Frolov, and Ivan Titov. 2017. A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 411–420, Vancouver, Canada. Association for Computational Linguistics.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing(EMNLP2017)*, pages 1506–1515, Copenhagen, Denmark. Association for Computational Linguistics.

A. Meyers, R. Reeves, C. Macleod, R. Szekely, V. Zielinska, B. Young, and R. Grishman. 2004. The nombank project: An interim report. In *HLT-NAACL 2004 Workshop: Frontiers in Corpus Annotation*, pages 24–31, Boston, Massachusetts, USA. Association for Computational Linguistics.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinkova, Dan Flickinger, Jan Hajic, and Zdenka Uresova. 2015. Semeval 2015 task 18: Broad-coverage semantic dependency parsing. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 915–926, Denver, Colorado. Association for Computational Linguistics.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Comput. Linguist.*, 31(1):71–106.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018*, pages 2227–2237.

Feng Qian, Lei Sha, Baobao Chang, LuChen Liu, and Ming Zhang. 2017. Syntax aware lstm model for semantic role labeling. In *Proceedings of the 2nd Workshop on Structured Prediction for Natural Language Processing*, pages 27–32, Copenhagen, Denmark. Association for Computational Linguistics.

Michael Roth and Mirella Lapata. 2016. Neural semantic role labeling with dependency path embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1192–1202, Berlin, Germany. Association for Computational Linguistics.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT*, pages 464–468.

Dan Shen and Mirella Lapata. 2007. Using semantic roles to improve question answering. In *Proceedings of the 2007 Joint Conference on Empirical*

*Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 12–21, Prague, Czech Republic. Association for Computational Linguistics.

R. A. Sinoara, R. G. Rossi, and S. O. Rezende. 2016. Semantic role-based representations in text classification. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2313–2318.

Asher Stern and Ido Dagan. 2014. Recognizing implied predicate-argument relationships in textual inference. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 739–744, Baltimore, Maryland. Association for Computational Linguistics.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5027–5038.

Mihai Surdeanu, Sanda Harabagiu, John Williams, and Paul Aarseth. 2003. Using predicate-argument structures for information extraction. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 8–15, Sapporo, Japan. Association for Computational Linguistics.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The conll 2008 shared task on joint parsing of syntactic and semantic dependencies. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 159–177, Manchester, England. Coling 2008 Organizing Committee.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2015. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567.

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2017. Deep semantic role labeling with self-attention.

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2018. Deep semantic role labeling with self-attention. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence,*, pages 4929–4936.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6000–6010. Curran Associates, Inc.

Ralph Weischedel, Eduard Hovy, Mitchell Marcus, Martha Palmer, Robert Belvin, Sameer Pradhan, Lance Ramshaw, and Nianwen Xue. 2011. Ontonotes: A large training corpus for enhanced processing. *Handbook of Natural Language Processing and Machine Translation. Springer*.

Aaron Steven White, Drew Reisinger, Keisuke Sakaguchi, Tim Vieira, Sheng Zhang, Rachel Rudinger, Kyle Rawlins, and Benjamin Van Durme. 2016. Universal decompositional semantics on universal dependencies. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1713–1723, Austin, Texas. Association for Computational Linguistics.

Qingrong Xia, Zhenghua Li, Min Zhang, Meishan Zhang, Guohong Fu, Rui Wang, and Luo Si. 2019. Syntax-aware neural semantic role labeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 201–206, Berlin, Germany. Association for Computational Linguistics.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1127–1137, Beijing, China. Association for Computational Linguistics.

# VerbAtlas: a Novel Large-Scale Verbal Semantic Resource and Its Application to Semantic Role Labeling

**Andrea Di Fabio**[◇♡], **Simone Conia**[◇], **Roberto Navigli**[◇]
[◇]Department of Computer Science
[♡]Department of Literature and Modern Cultures
Sapienza University of Rome, Italy
{difabio,conia,navigli}@di.uniroma1.it

## Abstract

We present VerbAtlas, a new, hand-crafted lexical-semantic resource whose goal is to bring together all verbal synsets from WordNet into semantically-coherent frames. The frames define a common, prototypical argument structure while at the same time providing new concept-specific information. In contrast to PropBank, which defines enumerative semantic roles, VerbAtlas comes with an explicit, cross-frame set of semantic roles linked to selectional preferences expressed in terms of WordNet synsets, and is the first resource enriched with semantic information about implicit, shadow, and default arguments. We demonstrate the effectiveness of VerbAtlas in the task of dependency-based Semantic Role Labeling and show how its integration into a high-performance system leads to improvements on both the in-domain and out-of-domain test sets of CoNLL-2009. VerbAtlas is available at http://verbatlas.org.

## 1 Introduction

During the last two decades, we have witnessed increased attention to Natural Language Understanding, a core goal of Natural Language Processing (NLP). Several challenges, however, are yet to be overcome when it comes to performing sentence-level semantic tasks (Navigli, 2018). In order to understand the meaning of sentences, the semantics of verbs plays a crucial role, since verbs define the argument structure roughly in terms of "who" did "what" to "whom", with the arguments being the constituents that bear a semantic relation (called semantic role) with the verb. In the following example, "Joe" and "lunch" are arguments of "eat", whose argument structure identifies them, respectively, as the Agent and the Patient of the scenario evoked by the verb:

[Joe]$_{\texttt{Agent}}$ is [eating]$_{\texttt{Verb}}$ his [lunch]$_{\texttt{Patient}}$

The automatic identification and labeling of argument structures is a task pioneered by Gildea and Jurafsky (2002) called Semantic Role Labeling (SRL). SRL has become very popular thanks to its integration into other related NLP tasks such as machine translation (Liu and Gildea, 2010), visual semantic role labeling (Silberer and Pinkal, 2018) and information extraction (Bastianelli et al., 2013).

In order to be performed, SRL requires the following core elements: 1) a verb inventory, and 2) a semantic role inventory. However, the current verb inventories used for this task, such as PropBank (Palmer et al., 2005) and FrameNet (Baker et al., 1998), are language-specific and lack high-quality interoperability with existing knowledge bases. Furthermore, such resources provide low to medium coverage of the verbal lexicon (cf. Table 1), with PropBank showing the best figures, but still lower than other lexical inventories like WordNet (Fellbaum et al., 1998). Finally, the informativeness of the semantic roles defined in the various resources ranges from underspecified, as in PropBank's roles, to overspecified, as in FrameNet's frame elements. This poses multiple issues in terms of interpretability or cross-domain applicability.

To overcome the above limitations, in this paper we present VerbAtlas, a manually-crafted inventory of verbs and argument structures which provides several contributions: 1) full coverage of the English verbal lexicon, 2) prototypical argument structures for each cluster of synsets that define a semantically-coherent frame, 3) cross-domain explicit semantic roles, 4) the specification of refined semantic information and selectional preferences for the argument structure of frames, 5) linkage to WordNet and, as a result, to BabelNet (Navigli and Ponzetto, 2010) and Open Multilingual Wordnet (Bond and Foster, 2013), which in turn enable scalability across languages. Furthermore, to



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 627–637, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| | Cluster types | # | Argument roles | # | Meaning units | # |
|---|---|---|---|---|---|---|
| **FrameNet** | Frames | 1,224 | Frame elements | 10,542 | Lexical units | 5,200 |
| **VerbNet** | Levin's classes | 329 | Thematic roles | 39 | Senses | 6,791 |
| **PropBank** | Verbs | 5,649 | Proto-roles | 6 | Framesets | 10,687 |
| **WordNet** | – | – | – | – | Synsets | 13,767 |
| **VerbAtlas** | Frames | 466 | Semantic roles | 25 | Synsets | 13,767 |

Table 1: Quantitative analysis of popular verbal resources. Since each of these resources is based on different linguistic assumptions, in the top row we use general labels to encompass resource-specific terms that can be viewed as homologous: cluster type, argument roles and meaning units. WordNet does not provide argument structures except for sentence frames which, however, are syntactic and do not specify any roleset.

make VerbAtlas suitable for NLP tasks that rely on PropBank, we also provide a mapping to its framesets. Finally, we prove through an SRL experiment that VerbAtlas is robust and enables state-of-the-art performances on the CoNLL-2009 dataset.

## 2 Related work

The most popular English verbal resources are FrameNet (Baker et al., 1998), PropBank (Palmer et al., 2005), and VerbNet (Kipper-Schuler, 2005). Each resource is based on a different linguistic theory, which leads to different information being provided for each verb (cf. Table 1). FrameNet, in particular, was the first resource to be used for SRL (Gildea and Jurafsky, 2002): it is based on frame semantics, theorized by Fillmore (1976), which assumes different roles, i.e., frame elements, for different frames[1]. This led to a proliferation of thousands of roles for only 5200 verbs. Such domain specificity makes it difficult to scale to open-text SRL (Hartmann et al., 2017).

PropBank challenges the issue of FrameNet's roles with a repository of only 6 different core roles plus 19 modifiers for 10,687 framesets[2]. This resource is the most widely adopted for SRL, as also attested by the popularity of datasets such as CoNLL-2005 (Carreras and Màrquez, 2005) and CoNLL-2009 (Hajič et al., 2009). PropBank's methodology was also used for other languages, such as Arabic (Palmer et al., 2008), Chinese (Xue and Palmer, 2003), Spanish and Catalan (Taulé et al., 2008), Hindi-Urdu (Bhatt et al., 2009), Brazilian Portuguese (Duran and Aluísio, 2011), Finnish (Haverinen et al., 2015), Turkish (Şahin and Adalı, 2018), Basque (Aldezabal et al., 2010), among others. Its application goes well beyond the annotation of corpora: in fact, it was also adopted for the Abstract Meaning Representation (Banarescu et al., 2013), a semantic language that aims at abstracting away from cross-lingual syntactic idiosyncrasies, and NomBank (Meyers et al., 2004), a resource which provides argument structures for nouns. However, PropBank's major drawback is that its roles do not explicitly mark the type of semantic relation with the verb, instead they just enumerate the arguments (i.e., `Arg0`, `Arg1`, etc.). Due to this, role labels do not preserve the same type of semantic relation across verbs, e.g., the first arguments of "eat" and "feel" are both labeled with `Arg0` even if they express different relations (`Agent` and `Experiencer`, respectively).

VerbNet addresses this limit by providing explicit, human-readable roles such as `Agent`, `Patient`, `Experiencer`, etc. Yet, VerbNet suffers from low coverage, in that it includes only 6791 verbs, which makes it a suboptimal resource for wide-coverage SRL. Another drawback of VerbNet is its organization into Levin's classes (Levin, 1993), namely, 329 groups of verbs sharing the same syntactic behavior, independently of their meaning. As a consequence, its classes cannot be used straightforwardly in a semantic task.

On top of their individual limitations, the above resources also have some common drawbacks. One of these is language specificity, which implies a considerable amount of work will be needed for the creation of a corresponding resource for each new language of interest. Another common problem is the lack of coverage, as shown in Table 1. The highest-coverage inventory is PropBank, with its 10,687 framesets and 5,649 verbs. However, PropBank's coverage is still limited when compared to computational lexicons like WordNet, which contains 13,767 verbal concepts and 11,529 distinct verbs.

[1] "By the term 'frame' I have in mind any system of concepts related in such a way that to understand any one of them you have to understand the whole structure in which it fits." (Fillmore, 1982).

[2] "A frameset corresponds to a coarse-grained sense of the verb which has a specific set of semantic arguments." (Babko-Malaya, 2005).

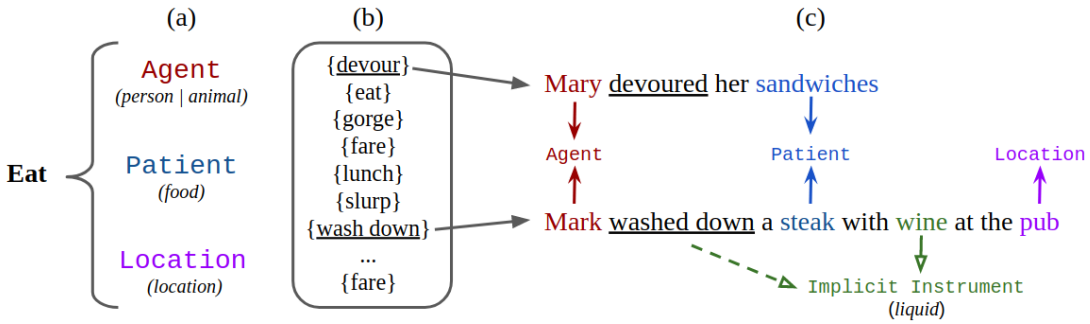


Figure 1: Structure of the EAT frame. The colored roles on the left (a) constitute the Predicate Argument Structure of the frame and their respective selectional preferences within round brackets. Box (b) shows a sample of the frame's synsets (a synonym per synset is shown). In the example sentences (c), we show the head words colored according to their semantic roles. While in the first sentence the synset {devour} does not project the `Location`, the synset {wash down} projects an implicit `Instrument` with its synset-level selectional preference *(liquid)*.

To get the best of the three worlds, there have been attempts to map the aforementioned resources to each other. To our knowledge the most popular endeavors are SemLink (Palmer, 2009; Bonial et al., 2013) and Predicate Matrix (De Lacalle et al., 2014), the latter being an extension of the former via automatic methods. However, while the main drawback of SemLink is coverage, the Predicate Matrix suffers from quality issues.

Both the foregoing limitations are addressed in VerbAtlas, the manually-curated resource that we present in this paper. With VerbAtlas we provide the community with a resource that improves the main features of the existing inventories of verbs, while also adding new semantic information. Compared to FrameNet, VerbAtlas has fewer frames and roles while at the same time presenting full coverage in terms of concepts (cf. Table 1). VerbNet's roles, instead, provided inspiration for our role repository, but we reduced the number of roles from 39 to 25 in order to alleviate data sparsity issues. PropBank was mapped to VerbAtlas synsets and roles in order to enable SRL systems to exploit its additional semantic information and improve verb coverage. Moreover, in contrast to the other resources, the use of WordNet synsets makes VerbAtlas able to scale multilingually through resources such as BabelNet.

In Section 3 we introduce VerbAtlas and its features; in Section 4 we explain how this resource was built and organized. Finally, Section 5 validates VerbAtlas experimentally in the SRL task.

## 3 VerbAtlas

We now introduce VerbAtlas, a new verbal semantic resource structured into frames which group semantically-coherent synsets from WordNet (v3.0). A VerbAtlas frame is a cluster of verb meanings expressing similar semantics, which expands upon the frame notion of FrameNet. Each frame is provided with an argument structure that generalizes over all the synsets in the frame plus preferential selections for each semantic role. Furthermore, synsets are enriched with novel semantic information. In Figure 1 we show the structure of the EAT frame in VerbAtlas, which we use as a running example to illustrate the new features in our work and compare them with current verbal resources.

### 3.1 Frame organization

We define a frame in VerbAtlas as a cluster of WordNet synsets that, with different shades of meaning, express a certain scenario. For instance, the EAT frame, an excerpt of which is shown in Figure 1(b), comprises all the synsets specializing the general scenario of "eating", including synsets such as {eat}, {devour, guttle, raven, pig}, etc.

This organization of frames is intended to overcome the limitations affecting VerbNet and FrameNet. In fact, while the former organizes verbs by syntactic rather than semantic behavior, the latter is affected by the sparsity of 5,200 verbal senses (i.e., lexical units) distributed across frames. Instead, PropBank's framesets correspond to a coarse-grained sense of a verb, but each frameset expresses its own separate argument structure.

Differently from the other resources, VerbAtlas frames are organized into 466 wide-coverage and semantically-coherent clusters (cf. Table 1) which provide cross-frame argument structures.

### 3.2 Semantic roles

A limit in PropBank is its inventory of so-called proto-roles (`Arg0`, `Arg1`, etc., with the first two roughly corresponding, respectively, to a proto-agent and a proto-patient (Dowty, 1991)), which does not provide human-readable labels and is predicate-specific (i.e., the same label does not necessarily depict the same semantic relation across predicates).

While PropBank roles do not provide clear explicit semantics, FrameNet roles are explicit but frame-specific (e.g., `Ingestor` and `Ingestibles` for the "Ingestion" frame, or `Impactor` and `Impactee` for the "Impact" frame). This produces a fine-grained role inventory that makes it difficult for SRL systems to generalize across frames. In VerbAtlas we follow VerbNet and take a middle-ground approach: our inventory of 25 semantic roles is inspired by VerbNet, whose 39 labels (like `Agent`, `Patient`, `Time`, etc.) are explicit, cross-frame and domain-general. The rationale is that these features enable neural networks employed in the SRL task to generalize across frames in a consistent way, as we show in Section 5.2. For example, we can tag the arguments of different VerbAtlas frames – such as EAT, HIT and CONQUER – with just two roles, namely: `Agent` and `Patient`.

### 3.3 Prototypical Argument Structure

Each VerbAtlas frame expresses a Prototypical Argument Structure (PAS) that generalizes over all the synsets in a particular frame. The PAS specifies a roleset that defines the frame's overall meaning (e.g., in Figure 1(a) `Agent`, `Patient` and `Location`). In Figure 1(c) we show two sentences annotated with argument roles.

Since we do not distinguish between core roles and adjuncts, we decided to also include in the various PAS roles which might be projected optionally by argument structures that are nonetheless present in the scenario evoked by the frame. For instance, the inclusion of the `Location` role in the PAS of the EAT example ensures a robust descriptiveness of the PAS across synsets in the same frame.

### 3.4 Selectional preferences

To narrow down the number of candidates for a particular argument slot and provide further semantic structure, the semantic role of the PAS has been manually labeled with selectional preferences from a set of 116 macro-concepts. Our selectional preferences are defined by WordNet synsets whose hyponyms are expected to be likely candidates to the corresponding argument slot, a strategy similar to that of Agirre and Martinez (2002) which, instead, was algorithm-based.

Consider again the EAT frame and its PAS: `Agent`, `Patient`, `Location` (Figure 1 (a)). In this frame, most of the example sentences from the WordNet synsets express a `Patient` like "cake", "meat" or "banana", thus, since their common hypernym is "food", we provided the PAS with the information that the `Patient` prototypically expects hyponyms of the {food, solid food} synset. The `Location` role is labeled with its homonymous {location} synset given its generality.

### 3.5 Synset-level semantic information

To enrich the semantic representation of synsets, VerbAtlas provides semantic and pragmatic (English-specific) information regarding implicit (1,028 labels), shadow and default arguments (2,979 labels) inspired by Pustejovsky (1995) and inferred from synset's glosses and examples. To our knowledge, the following information is new in a large-scale verbal resource such as ours:

- **implicit arguments**, i.e., implicit in the argument structure of the verb but not always syntactically expressed. Consider the synset {overleap, vault} in our JUMP frame: since its gloss is "Jump across or leap over (an obstacle)", we know that the `Patient` of this verb can be a hyponym of {obstacle}, therefore implying a selectional preference on the role with the {obstacle} synset.

- **shadow and default arguments**: the former is incorporated in the meaning of the verb but not syntactically expressed. An example from the EAT frame is {eat in, dine in} ("Eat at home"). This synset is tagged with the shadow argument `Location` = {home}, since the latter is not expressed syntactically. On the other hand, default arguments are logically implied but not syntactically expressed. These are also tagged as shadow arguments. For instance, the synset {deliver} (as in "Our local supermarket delivers") has the label `Patient` = {grocery} to provide the commonsense information that what a supermarket usually delivers is groceries.

We are aware of the problems raised about Pustejovsky's framework of analysis (Fodor and Lepore, 1998), but we believe that this new information, if properly exploited, would be fruitful for better meaning representations.

### 3.6 Linkage to PropBank and multilingual resources

To allow a straightforward applicability to and evaluation of VerbAtlas on semantic tasks based on PropBank, such as SRL, each frameset and roleset in the CoNLL-2009 dataset was mapped to the corresponding VerbAtlas frame and roleset. This mapping was done starting from the Predicate Matrix (De Lacalle et al., 2014) and then manually corrected and augmented. Finally, we aligned PropBank's roles with the corresponding roles in VerbAtlas PAS (e.g., `Arg0` and `Arg1` of PropBank's eat.01 frameset were mapped, respectively, to the VerbAtlas PAS roles `Agent` and `Patient` of the frame EAT).

Moreover, thanks to the use of WordNet and its semantic nature, VerbAtlas can easily scale to arbitrary languages. This can be achieved by leveraging BabelNet (Navigli and Ponzetto, 2010), a lexical-semantic resource that provides multilingual synsets in 284 different languages linked to WordNet itself. As a result, VerbAtlas can be used in virtually any language, in contrast to PropBank and other resources which are inherently language-specific and require considerable human intervention in each new language.

Finally, the above implies that if there is a verbal repository for one of the languages linked to the aforementioned resources, its argument structures can be seamlessly aligned with the PAS of VerbAtlas, as well as with VerbAtlas frames.

## 4 Methodology

### 4.1 Bottom-up approach

The manual construction of VerbAtlas was performed in a bottom-up fashion. Rather than forcing synsets into a predetermined set of frames, we started the clustering process from the full inventory of 13,767 WordNet synsets, which represent the building blocks of VerbAtlas frames. This allowed us to induce the semantics of 466 frames (with an average of 29.5 synsets per frame) and make VerbAtlas consistent both in terms of lexical semantics and the frame's argument structures.

| | | |
|---|---|---|
| EAT | CLASSIFY | SWITCH |
| KILL | CLEAN | MOVE SOMETHING |
| DRINK | AMELIORATE | JUDGE |
| POUR | TRANSPORT | PERFORM |
| OBTAIN | BEHAVE | STOP |
| GIVE | HIT | COMBINE |

Table 2: Frames with the highest number of synsets.

### 4.2 Creation of frames

VerbAtlas frames were induced via semantic similarity between synsets. During multiple iterations over the verb inventory, if two or more synsets were perceived as similar, namely, if they shared features like the purpose of the action and the participants in the action (Hill et al., 2015) (e.g., "dine" and "lunch"), they were clustered together to form a new semantically-coherent frame. A one-synset-one-frame strategy was used to avoid any future mapping problem with other resources. The process is based on synset-by-synset human inspection. Two synsets are clustered together in the same frame if they express semantically-similar scenarios. For example, the verbs "kill" and "slaughter" share similar participants in the action (an `Agent` who kills/slaughters; a `Patient` who is killed/slaughtered) and purpose (`Agent` makes `Patient` die), so they are clustered into the KILL frame.

At the end of the first iteration, we checked the resulting frames and named them according to the common action implied by the synsets contained therein. For example, the EAT frame (Figure 1(b)) is composed of synsets that depict different kinds of action implying eating, like {devour, ..., pig} and {gorge, ..., glut}. In Table 2 we report the frames with the highest number of verb synsets.

To validate the resulting frames we adopted a strategy similar to Hovy et al. (2006): we provided 3 linguists not involved in the clustering with a random sample of 1000 frame-synset pairs and asked them if the action expressed by the synset was implied by that expressed by the frame. We iterated over the inventory various times and moved the synsets from one frame to another until the Cohen's Kappa coefficient (Eugenio and Glass, 2004) of their (yes or no) agreement was $\kappa \geq 0.80$. Once this value was attained, we finalized the overall clustering and the resulting frames.

### 4.3 Establishing explicit semantic roles

Inspired by existing research on semantic roles (Bonial et al., 2011; Allen and Teng, 2018),

| | |
|---|---|
| Agent | Material |
| Attribute | Patient |
| Beneficiary | Product |
| Cause | Purpose |
| Co-Agent | Recipient |
| Co-Patient | Result |
| Co-Theme | Source |
| Destination | Stimulus |
| Experiencer | Theme |
| Extent | Time |
| Goal | Topic |
| Instrument | Value |
| Location | |

Table 3: List of VerbNet roles which make up the VerbAtlas role inventory.

| | |
|---|---|
| Affector | Manner |
| Axis | Path |
| Context | Pivot |
| Duration | Precondition |
| Final_Time | Predicate |
| Initial_Location | Reflexive |
| Initial_State | Trajectory |

Table 4: List of VerbNet roles unused in VerbAtlas.

VerbAtlas implements VerbNet's human-readable role labels across its frames and merges together some VerbNet roles which can be seen as complementary. For example, Initial_Location and Initial_State are subsumed by Source. The hunch is that we can consider them as playing the same role but in different scenarios: the former is the Source for verbs of change of location, while the latter is the Source for verbs of change of state. Furthermore, in VerbAtlas we did not want to use poorly instantiated roles (e.g., Affector), therefore 14 of the roles (Table 4) were subsumed by coarser and more common roles (e.g., Agent in place of Affector, see Table 3).

### 4.4 Rationale of a Prototypical Argument Structure

We defined as "prototypical" an argument structure capable of being applied to all the synsets in a frame. To achieve this, each PAS was defined at the end of the first iteration over the inventory of synsets and constantly adjusted until the last iteration. The initial PAS was inspired by the argument structure of the common verbal concept implied by the synsets constituting the frame. For instance, in the case of BETRAY, Agent and Patient. The PAS was later expanded due to synsets inside the frame that projected additional arguments (e.g., the Goal role of {defect, desert} as in the sentence "The reporter defected to another *network*").

### 4.5 Rationale of selectional preferences

Each semantic role of a PAS was provided with selectional preferences from a set of 116 synsets. The set was created by generalizing across the arguments in the example sentences of each synset in a given frame. The result for each PAS argument was one or more hypernyms in the WordNet taxonomy which were common across the frame synsets.

In contrast to VerbNet selectional restrictions, the selectional preferences in VerbAtlas do not restrict the occurrence of words with a particular feature (e.g., solid, liquid, etc.), rather, they suggest the most prototypical hypernym(s) for a given argument. We opted for preferences instead of restrictions to not exclude the potential metaphorical use of a verb.

## 5 Experiments

In previous sections we discussed the qualitative and structural advantages of VerbAtlas compared to existing resources. Because, in principle, a new linguistic resource might seem to provide an arbitrary contribution, we also provide here an experimental validation of its usefulness in an SRL task. First, we present our experimental setup (Section 5.1) and then discuss our results (Section 5.2).

### 5.1 Experimental setup

**Goal** We argue that the quality and wealth of information in VerbAtlas can effectively improve the performance of an existing, high-performance SRL system like Cai et al. (2018).

**Datasets** We performed our experiments on the English part of CoNLL-2009 in-domain and out-of-domain datasets (Hajič et al., 2009) designed for the dependency-based SRL task and whose annotations concern predicate argument structures from PropBank and nominal argument structures from NomBank (Meyers et al., 2004). The dataset consists of 39,279 sentences and 958,167 tokens (18.7% of which are argument bearers).

Through this experiment we aim to show that the additional information provided by VerbAtlas frames and semantic roles improves the performance of a neural network on the in-domain test set. We also aim to demonstrate a better ability to generalize on the out-of-domain dataset.

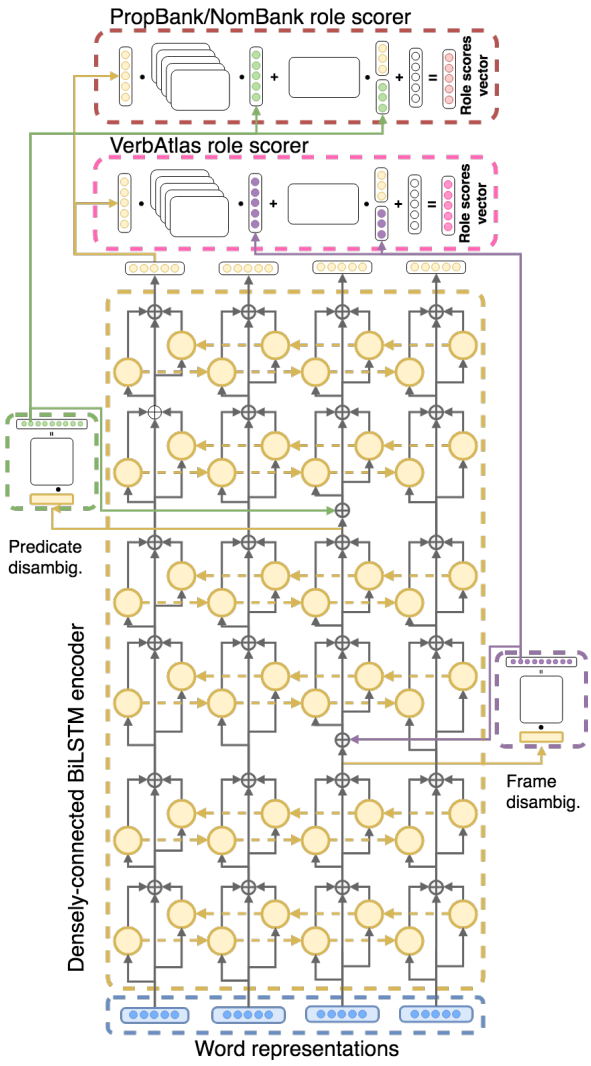


Figure 2: Overview of the model architecture. From bottom to top, a sequence of word embeddings is fed to a densely-connected BiLSTM encoder where the output of each encoding layer is concatenated with its input. The predicate hidden representation from the second BiLSTM layer is used for VerbAtlas frame disambiguation (right), whereas the predicate hidden representation from the fourth BiLSTM layer is used for PropBank predicate disambiguation (left). The topmost output of the BiLSTM encoder is used together with the output of the PropBank predicate disambiguation layer to obtain a PropBank-style SRL output, and together with the output of the VerbAtlas frame disambiguation layer to obtain a VerbAtlas-style SRL output.

**Baseline model** Our baseline model in Figure 2 is built on top of the syntax-agnostic model proposed by Cai et al. (2018) in that it is mainly composed of a word representation layer, a sequence encoder and a biaffine attentional role scorer. A key difference is that our model features a multi-output layer that returns PropBank (PB) and NomBank (NB) labels (i.e., framesets and their roles) and, if the predicate is a verb, also VerbAtlas labels (i.e., frames and their roles). With this design choice, the output of our model can be directly compared to the output of previous SRL models. At the same time, our model achieves a deeper and more general understanding of the relations between a PB or NB predicate and its arguments by learning from the corresponding VerbAtlas frame and its semantically-coherent roles.

Formally, our model is built on top of the following components:

- A **word representation layer** that, given a sentence $\mathbf{s} = \langle w_1, w_2, \ldots, w_n \rangle$, builds a sequence of word representations $\mathbf{x} = \langle \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \rangle$ where $\mathbf{x}_i$ is an embedding representing $w_i$. Each $\mathbf{x}_i$ is the result of the concatenation of a pre-trained word embedding $\mathbf{e}^{pt}$, and the following trainable vectors: a word embedding $\mathbf{e}^{w}$, a lemma embedding $\mathbf{e}^{l}$, a POS embedding $\mathbf{e}^{pos}$, and a predicate lemma embedding $\mathbf{e}^{pred}$ (active only if $w_i$ is a predicate). Formally: $\mathbf{x}_i = \mathbf{e}^{pt} \oplus \mathbf{e}^{w} \oplus \mathbf{e}^{l} \oplus \mathbf{e}^{pos} \oplus \mathbf{e}^{pred}$. We use GloVe embeddings (Pennington et al., 2014) as our underlying pre-trained word embeddings.

- A **densely-connected BiLSTM encoder** that, given a sequence $\mathbf{x}$ of word representations, returns a sequence of encodings $\mathbf{y} = \langle \mathbf{y}_i = \text{BiLSTM}(\mathbf{x}_i; \mathbf{x}) : \forall i \in \{1, \ldots, n\} \rangle$, where $\mathbf{y}_i$ is a dynamic representation of $\mathbf{x}_i$ with respect to the context defined by the whole sequence $\mathbf{x}$. In a densely-connected BiLSTM encoder, the output of each layer is concatenated with the input of the same layer to mitigate the vanishing gradient problem. If $\mathbf{h}_i^k$ is the encoding of the $k$-th layer for $\mathbf{x}_i$, then $\mathbf{h}_i^{k+1} = \mathbf{h}_i^k \oplus \text{LSTM}^f(\mathbf{h}_i^k; \mathbf{h}_{1:i-1}^k) \oplus \text{LSTM}^b(\mathbf{h}_i^k; \mathbf{h}_{i+1:n}^k)$, and $\mathbf{y}_i = \mathbf{h}_i^m$ where $m = 6$ is the final BiLSTM layer, while $\text{LSTM}^f$ and $\text{LSTM}^b$ are the forward and backward LSTM transformations.

- A **frame disambiguation layer** that, given the BiLSTM encoding $\mathbf{y}_{pred}^i$ of a predicate $w_{pred}$ at the $i$-th encoder layer (with $i = 2$), disambiguates $w_{pred}$ with a VerbAtlas frame $f$, returning a trainable frame embedding $\mathbf{e}^f$:

$$\mathbf{h}_{pred}^i = \text{ReLU}(\mathbf{W}' \cdot \mathbf{y}_{pred}^i + \mathbf{b}')$$

$$f = \text{argmax}(\mathbf{W}^f \cdot \mathbf{h}_{pred}^i + \mathbf{b}^f)$$

- A **predicate disambiguation layer** that, given the BiLSTM encoding $\mathbf{y}^j_{pred}$ of a predicate $w_{pred}$ at the $j$-th encoder layer (with $j = 4$) and the frame embedding $\mathbf{e}^f$, disambiguates $w_{pred}$ with a PB or NB frameset $p$, returning a trainable predicate embedding $\mathbf{e}^p$:

$$\mathbf{h}^j_{pred} = \text{ReLU}(\mathbf{W}'' \cdot (\mathbf{y}^j_{pred} \oplus \mathbf{e}^f) + \mathbf{b}'')$$

$$p = \text{argmax}(\mathbf{W}^p \cdot \mathbf{h}^j_{pred} + \mathbf{b}^p)$$

- A **biaffine attentional PB and NB role scorer** that, given a BiLSTM encoding $\mathbf{y}_i$ for a word $w_i$ and a predicate embedding $\mathbf{e}^p$, returns a vector $\mathbf{s}^p_i$ of PB/NB role scores for $w_i$ with respect to $p$ in a similar fashion to Cai et al. (2018). Formally:

$$\mathbf{s}^p_i = \mathbf{y}_i^\top \cdot \mathbf{W}^{role}_{\text{PB}} \cdot \mathbf{e}^p + \mathbf{U}^{role}_{\text{PB}}(\mathbf{y}_i \oplus \mathbf{e}^p) + \mathbf{b}^{role}_{\text{PB}}$$

- A **biaffine attentional VerbAtlas role scorer** that, given a BiLSTM encoding $\mathbf{y}_i$ for a word $w_i$ and a frame embedding $\mathbf{e}^f$, returns a vector $\mathbf{s}^f_i$ of VerbAtlas role scores for $w_i$ with respect to $f$. Formally:

$$\mathbf{s}^f_i = \mathbf{y}_i^\top \cdot \mathbf{W}^{role}_{\text{VA}} \cdot \mathbf{e}^f + \mathbf{U}^{role}_{\text{VA}}(\mathbf{y}_i \oplus \mathbf{e}^f) + \mathbf{b}^{role}_{\text{VA}}$$

We found it beneficial to interleave the frame disambiguation layer and the predicate disambiguation layer within the BiLSTM encoder layers, enhancing the input of the upper encoder layers with the corresponding frame and predicate embeddings. Finally, our model loss is defined as:

$$l_{\text{total}} = l_{\text{PropBank/NomBank roles}} + l_{\text{VerbAtlas roles}} + l_{\text{predicate disambiguation}} + l_{\text{frame disambiguation}}$$

Since the choice of hyperparameters for an LSTM-based model can significantly affect results (Reimers and Gurevych, 2017), we trained our baseline using the same values as in Cai et al. (2018) unless otherwise stated.

### 5.2 Results

**In-domain SRL** Table 5 reports the results of our syntax-agnostic baseline model on the CoNLL-2009 in-domain test set. Not only does our model outperform the syntax-agnostic model of Cai et al. (2018) by a significant margin (according to a $\chi^2$ test, $p < 0.05$, on precision and recall) with a 0.4% $F_1$ improvement, but it also slightly outperforms the syntax-aware model of Li et al. (2018) with a 0.2% $F_1$ improvement. We recall that the main difference between our baseline model and that of Cai et al. (2018) lies in the additional VerbAtlas frame disambiguation layer and role scorer. This demonstrates the boost provided by VerbAtlas in PropBank/NomBank-based SRL.

| *Syntax-aware system* | P | R | $F_1$ |
|---|---|---|---|
| Roth and Lapata (2016) | 88.1 | 85.3 | 86.7 |
| Marcheggiani and Titov (2017) | 89.1 | 86.8 | 88.0 |
| He et al. (2018) | 89.7 | **89.3** | 89.5 |
| Li et al. (2018) | **90.3** | **89.3** | **89.8** |
| *Syntax-agnostic system* | P | R | $F_1$ |
| Marcheggiani et al. (2017) | 88.7 | 86.8 | 87.7 |
| He et al. (2018) | 89.5 | 87.9 | 88.7 |
| Cai et al. (2018) | 89.9 | 89.2 | 89.6 |
| This work | **90.5** | **89.5** | **90.0** |

Table 5: Results on the English in-domain test.

| *Syntax-agnostic system* | P |
|---|---|
| He et al. (2018) | 95.5 |
| Cai et al. (2018) | 95.0 |
| This work | **96.0** |

Table 6: Predicate disambiguation results on the English in-domain test. In CoNLL-2009 predicates are already identified, hence we only report predicate disambiguation precision.

| *Syntax-agnostic system* | P | R | $F_1$ |
|---|---|---|---|
| Marcheggiani et al. (2017) | 79.4 | 76.2 | 77.7 |
| He et al. (2018) | **81.7** | 76.1 | 78.8 |
| Cai et al. (2018) | 79.8 | 78.3 | 79.0 |
| This work | 81.1 | **78.4** | **79.7** |

Table 7: Results on the English out-of-domain test.

**In-domain predicate disambiguation** Remarkably, as shown in Table 6, our model outperforms the previously reported best scores in the predicate disambiguation subtask, scoring 96.0% in precision for the in-domain test set and reporting a 1.0% improvement over Cai et al. (2018) and a 0.5% improvement over He et al. (2018).

**Out-of-domain SRL** The results of our baseline model on the out-of-domain SRL CoNLL-2009 dataset are shown in Table 7. Our system improves the results reported by Cai et al. (2018) by 0.7% $F_1$, proving that using VerbAtlas improves the model's ability to generalize across domains.

| $F_1$ evaluated on | verbs + nouns | only verbs |
|---|---|---|
| This work | 90.0 | 91.1 |
| *without* VerbAtlas | 89.5 | 90.5 |
| *without* PB *nor* NB | – | 90.9 |

Table 8: The model shows a significant $F_1$ decrease when it is trained without exploiting VerbAtlas frames and roles (second row) falling in line with Cai et al. (2018) (89.5% vs 89.6%, respectively). On the contrary, removing the PropBank role scorer leads to a negligible performance drop (last row).

### 5.3 Analysis

Our main experiment proved that a semantics-focused resource for verbal predicates such as VerbAtlas can be successfully employed for CoNLL-like SRL datasets that include a mix of nominal and verbal predicates. But what is the contribution of VerbAtlas (i.e., of its semantically-clustered frames and semantically-coherent roles) to the overall performance? How does a VerbAtlas-only model fare when evaluated solely on verbal predicates against a PropBank-only model? To answer these questions, we evaluated the model in two further settings.

A first study aimed at quantifying the boost in performance the model gets from the use of VerbAtlas. Removing the VerbAtlas frame disambiguation layer and role scorer from our model significantly decreases (according to a $\chi^2$ test, $p < 0.05$, on precision and recall) the overall performance in $F_1$ score by 0.5%, as reported in Table 8 (left column), with results that are comparable to those of Cai et al. (2018) (89.5% vs 89.6% $F_1$, Tables 8 and 5, respectively). This proves that the model with VerbAtlas achieves a better understanding of semantic roles, thanks to using the VerbAtlas frame disambiguation layer and role scorer.

A second study aimed at comparing PropBank and VerbAtlas on their common ground, i.e., their coverage and organization of verb meanings into framesets and frames, respectively. Table 8 (right column) shows that removing the PropBank bi-affine role scorer leads to a negligible performance drop (0.2% $F_1$) in how it performs on argument labeling of verb predicates using only the semantic roles of VerbAtlas mapped at the output layer to PropBank roles (see Section 3.6); we note that the higher number of roles in VerbAtlas makes the argument labeling problem potentially harder. In contrast, when VerbAtlas is removed, the drop in performance is noteworthy, with a 0.6% decrease in $F_1$ on SRL of verbal predicates.

## 6 Conclusions and Future Work

In this paper we presented VerbAtlas, a new large-scale verbal semantic resource which provides generalizing argument structures with cross-frame semantic roles. The resource is available at `http://verbatlas.org`.

In contrast to other verb repositories, VerbAtlas offers full coverage of the English verbs and addresses the issues of current predicate resources, while at the same time providing linkage to WordNet and PropBank. This makes the resource fully compatible with previous datasets and scalable to arbitrary languages thanks to BabelNet.

While the frame creation process resulted in a strong agreement between annotators, we further validated the quality of VerbAtlas experimentally by showing that the integration of its frame information together with its explicit semantic roles enables a neural architecture to improve its performance on the Semantic Role Labeling task. This improvement translates across domains, demonstrating the robustness and variety of the knowledge provided in our resource.

As future work, we plan to take full advantage of the novel semantic features available in VerbAtlas, such as wide-coverage selectional preferences and synset-level information, by exploiting them in multilingual SRL and Word Sense Disambiguation tasks. Our plans include integrating the selectional preferences from SyntagNet (Maru et al., 2019), a new, large-scale lexical-semantic combination resource. We also plan to extend our methodology to nouns and adjectives, in a similar fashion to (O'Gorman et al., 2018) and connect the resulting frames to those in VerbAtlas.

## Acknowledgments

The authors gratefully acknowledge the support of the ERC Consolidator Grant MOUSSE No. 726487 and the ELEXIS project No. 731015 under the European Union's Horizon 2020 research and innovation programme.

This work was supported in part by the MIUR under grant "Dipartimenti di eccellenza 2018-2022" of the Department of Computer Science of Sapienza University.

## References


Eneko Agirre and David Martinez. 2002. Integrating selectional preferences in WordNet. In *Proceedings of First International WordNet Conference. Mysore (India)*, pages 1–9.

Izaskun Aldezabal, María Jesús Aranzabe, Arantza Díaz de Ilarraza Sánchez, and Ainara Estarrona. 2010. Building the Basque PropBank. In *Proceedings of LREC*.

James Allen and Choh Man Teng. 2018. Putting semantics into semantic roles. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 235–244.

Olga Babko-Malaya. 2005. PropBank annotation guidelines. *URL: http://verbs.colorado.edu*.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *Proceedings of the 17th International Conference on Computational Linguistics*, pages 86–90. Association for Computational Linguistics.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for Sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186.

Emanuele Bastianelli, Giuseppe Castellucci, Danilo Croce, and Roberto Basili. 2013. Textual inference and meaning representation in human robot interaction. In *Proceedings of the Joint Symposium on Semantic Processing. Textual Inference and Structures in Corpora*, pages 65–69.

Rajesh Bhatt, Bhuvana Narasimhan, Martha Palmer, Owen Rambow, Dipti Sharma, and Fei Xia. 2009. A multi-representational and multi-layered treebank for Hindi-Urdu. In *Proceedings of the Third Linguistic Annotation Workshop (LAW III)*, pages 186–189.

Francis Bond and Ryan Foster. 2013. Linking and extending an open multilingual Wordnet. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1352–1362.

Claire Bonial, William Corvey, Martha Palmer, Volha V Petukhova, and Harry Bunt. 2011. A hierarchical unification of LIRICS and verbnet semantic roles. In *2011 IEEE Fifth International Conference on Semantic Computing*, pages 483–489. IEEE.

Claire Bonial, Kevin Stowe, and Martha Palmer. 2013. Renewing and revising SemLink. In *Proceedings of the 2nd Workshop on Linked Data in Linguistics (LDL-2013): Representing and linking lexicons, terminologies and other language data*, pages 9–17.

Jiaxun Cai, Shexia He, Zuchao Li, and Hai Zhao. 2018. A full end-to-end semantic role labeler, syntactic-agnostic over syntactic-aware? In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2753–2765.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, CONLL '05, pages 152–164, Stroudsburg, PA, USA. Association for Computational Linguistics.

Maddalen Lopez De Lacalle, Egoitz Laparra, and German Rigau. 2014. Predicate Matrix: extending SemLink through WordNet mappings. In *Proceedings of LREC*, pages 903–909.

David Dowty. 1991. Thematic proto-roles and argument selection. *Language*, 67(3):547–619.

Magali Sanches Duran and Sandra Maria Aluísio. 2011. PropBank-Br: a Brazilian Portuguese corpus annotated with semantic role labels. In *Proceedings of the 8th Brazilian Symposium in Information and Human Language Technology*.

Barbara Di Eugenio and Michael Glass. 2004. The kappa statistic: A second look. *Computational linguistics*, 30(1):95–101.

Christiane Fellbaum et al. 1998. *WordNet: An electronic database*. MIT Press, Cambridge, MA.

Charles J. Fillmore. 1976. Frame semantics and the nature of language. *Annals of the New York Academy of Sciences*, 280(1):20–32.

Charles J. Fillmore. 1982. Frame semantics. *Linguistics in the Morning Calm*, pages 111–137.

Jerry A. Fodor and Ernie Lepore. 1998. The emptiness of the lexicon: reflections on James Pustejovsky's The Generative Lexicon. *Linguistic Inquiry*, 29(2):269–288.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational linguistics*, 28(3):245–288.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, et al. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–18. Association for Computational Linguistics.

Silvana Hartmann, Ilia Kuznetsov, Teresa Martin, and Iryna Gurevych. 2017. Out-of-domain FrameNet semantic role labeling. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 471–482.

Katri Haverinen, Jenna Kanerva, Samuel Kohonen, Anna Missilä, Stina Ojala, Timo Viljanen, Veronika Laippala, and Filip Ginter. 2015. The Finnish proposition bank. *Language Resources and Evaluation*, 49(4):907–926.

Shexia He, Zuchao Li, Hai Zhao, and Hongxiao Bai. 2018. Syntax for semantic role labeling, to be, or not to be. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2061–2071.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: the 90% solution. In *Proceedings of the human language technology conference of the NAACL, Companion Volume: Short Papers*, pages 57–60. Association for Computational Linguistics.

Karin Kipper-Schuler. 2005. *VerbNet: A broad-coverage, comprehensive verb lexicon*. University of Pensylvania.

Beth Levin. 1993. *English verb classes and alternations: A preliminary investigation*. University of Chicago press.

Zuchao Li, Shexia He, Jiaxun Cai, Zhuosheng Zhang, Hai Zhao, Gongshen Liu, Linlin Li, and Luo Si. 2018. A unified syntax-aware framework for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2401–2411.

Ding Liu and Daniel Gildea. 2010. Semantic role features for machine translation. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 716–724.

Marco Maru, Federico Scozzafava, Federico Martelli, and Roberto Navigli. 2019. SyntagNet: Challenging Supervised Word Sense Disambiguation with Lexical-Semantic Combinations. In *Proceedings of EMNLP-IJCNLP 2019*.

Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. 2004. The NomBank project: An interim report. In *Proceedings of the Workshop Frontiers in Corpus Annotation at HLT-NAACL 2004*.

Roberto Navigli. 2018. Natural Language Understanding: Instructions for (present and future) use. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI 2018)*, pages 5697–5702, Stockholm, Sweden.

Roberto Navigli and Simone Paolo Ponzetto. 2010. BabelNet: Building a very large multilingual semantic network. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 216–225. Association for Computational Linguistics.

Tim O'Gorman, Sameer Pradhan, Martha Palmer, Julia Bonn, Kathryn Conger, and James Gung. 2018. The new Propbank: Aligning Propbank with AMR through POS unification. In *Proceedings of LREC*.

Martha Palmer. 2009. Semlink: Linking Propbank, VerbNet and FrameNet. In *Proceedings of the generative lexicon conference*, pages 9–15. GenLex-09, Pisa, Italy.

Martha Palmer, Olga Babko-Malaya, Ann Bies, Mona T. Diab, Mohamed Maamouri, Aous Mansouri, and Wajdi Zaghouani. 2008. A pilot arabic Propbank. In *LREC*.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014*, pages 1532–1543. ACL.

James Pustejovsky. 1995. *The Generative Lexicon*. MIT Press, Cambridge MA.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of LSTM-networks for sequence tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 338–348.

Gözde Gül Şahin and Eşref Adalı. 2018. Annotation of semantic roles for the Turkish proposition bank. *Language Resources and Evaluation*, 52(3):673–706.

Carina Silberer and Manfred Pinkal. 2018. Grounding semantic roles in images. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2616–2626.

Mariona Taulé, Maria Antònia Martí, and Marta Recasens. 2008. Ancora: Multilevel annotated corpora for Catalan and Spanish. In *Proceedings of LREC*.

Nianwen Xue and Martha Palmer. 2003. Annotating the propositions in the Penn Chinese Treebank. In *Proceedings of the Second Workshop on Chinese Language Processing, SIGHAN 2003, Sapporo, Japan, July 11-12, 2003*.

# Parameter-free Sentence Embedding via Orthogonal Basis

Ziyi Yang[1]*, Chenguang Zhu[2], and Weizhu Chen[3]

[1]Department of Mechanical Engineering, Stanford university
[2]Microsoft Speech and Dialogue Research Group
[3]Microsoft Dynamics 365 AI
ziyi.yang@stanford.edu, {chezhu, wzchen}@microsoft.com

## Abstract

We propose a simple and robust non-parameterized approach for building sentence representations. Inspired by the Gram-Schmidt Process in geometric theory, we build an orthogonal basis of the subspace spanned by a word and its surrounding context in a sentence. We model the semantic meaning of a word in a sentence based on two aspects. One is its relatedness to the word vector subspace already spanned by its contextual words. The other is the word's novel semantic meaning which shall be introduced as a new basis vector perpendicular to this existing subspace. Following this motivation, we develop an innovative method based on orthogonal basis to combine pre-trained word embeddings into sentence representations. This approach requires zero parameters, along with efficient inference performance. We evaluate our approach on 11 downstream NLP tasks. Our model shows superior performance compared with non-parameterized alternatives and it is competitive to other approaches relying on either large amounts of labelled data or prolonged training time.

## 1 Introduction

The concept of word embeddings has been prevalent in NLP community in recent years, as they can characterize semantic similarity between any pair of words, achieving promising results in a large number of NLP tasks (Mikolov et al., 2013; Pennington et al., 2014; Salle et al., 2016). However, due to the hierarchical nature of human language, it is not sufficient to comprehend text solely based on isolated understanding of each word. This has prompted a recent rise in search for semantically robust embeddings for longer pieces of text, such as sentences and paragraphs.

Based on learning paradigms, the existing approaches to sentence embeddings can be categorized into two categories: i) parameterized methods and ii) non-parameterized methods.

**Parameterized sentence embeddings.** These models are parameterized and require training to optimize their parameters. SkipThought (Kiros et al., 2015) is an encoder-decoder model that predicts adjacent sentences. Pagliardini et al. (2018) proposes an unsupervised model, Sent2Vec, to learn an n-gram feature in a sentence to predict the center word from the surrounding context. Quick thoughts (QT) (Logeswaran and Lee, 2018) replaces the encoder with a classifier to predict context sentences from candidate sequences. Khodak et al. (2018) proposes *à la carte* to learn a linear mapping to reconstruct the center word from its context. Conneau et al. (2017) generates the sentence encoder InferSent using Natural Language Inference (NLI) dataset. Universal Sentence Encoder (Yang et al., 2018; Cer et al., 2018) utilizes the emerging transformer structure (Vaswani et al., 2017; Devlin et al., 2018) that has been proved powerful in various NLP tasks. The model is first trained on large scale of unsupervised data from Wikipedia and forums, and then trained on the Stanford Natural Language Inference (SNLI) dataset. Wieting and Gimpel (2017b) propose the gated recurrent averaging network (GRAN), which is trained on Paraphrase Database (PPDB) and English Wikipedia. Subramanian et al. (2018) leverages a multi-task learning framework to generate sentence embeddings. Wieting et al. (2015a) learns the paraphrastic sentence representations as the simple average of updated word embeddings.

**Non-parameterized sentence embedding.** Recent work (Arora et al., 2017) shows that, surprisingly, a weighted sum or transformation of word representations can outperform many sophisticated neural network structures in sen-

* Most of the work was done during internship at Microsoft



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 638–648, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

tence embedding tasks. These methods are parameter-free and require no further training upon pre-trained word vectors. Arora et al. (2017) constructs a sentence embedding called SIF as a sum of pre-trained word embeddings, weighted by reverse document frequency. Ethayarajh (2018) builds upon the random walk model proposed in SIF by setting the probability of word generation inversely related to the angular distance between the word and sentence embeddings. Rücklé et al. (2018) concatenates different power mean word embeddings as a sentence vector in $p$-mean. As these methods do not have a parameterized model, they can be easily adapted to novel text domains with both fast inference speed and high-quality sentence embeddings. In view of this trend, our work aims to further advance the frontier of this group and make its new state-of-the-art.

In this paper, we propose a novel sentence embedding algorithm, Geometric Embedding (GEM), based entirely on the geometric structure of word embedding space. Given a $d$-dim word embedding matrix $\boldsymbol{A} \in \mathbb{R}^{d \times n}$ for a sentence with $n$ words, any linear combination of the sentence's word embeddings lies in the subspace spanned by the $n$ word vectors. We analyze the geometric structure of this subspace in $\mathbb{R}^d$. When we consider the words in a sentence one-by-one in order, each word may bring in a novel orthogonal basis to the existing subspace. This new basis can be considered as the new semantic meaning brought in by this word, while the length of projection in this direction can indicate the intensity of this new meaning. It follows that a word with a strong intensity should have a larger influence in the sentence's meaning. Thus, these intensities can be converted into weights to linearly combine all word embeddings to obtain the sentence embedding. In this paper, we theoretically frame the above approach in a QR factorization of the word embedding matrix $\boldsymbol{A}$. Furthermore, since the meaning and importance of a word largely depends on its close neighborhood, we propose the sliding-window QR factorization method to capture the context of a word and characterize its significance within the context.

In the last step, we adapt a similar approach as Arora et al. (2017) to remove top principal vectors before generating the final sentence embedding. This step is to ensure commonly shared background components, e.g. stop words, do not bias sentence similarity comparison. As we build a new orthogonal basis for each sentence, we propose to have disparate background components for each sentence. This motivates us to put forward a sentence-specific principal vector removal method, leading to better empirical results.

We evaluate our algorithm on 11 NLP tasks. Our algorithm outperforms all non-parameterized methods and many parameterized approaches in 10 tasks. Compared to SIF (Arora et al., 2017), the performance is boosted by 5.5% on STS benchmark dataset, and by 2.5% on SST dataset. Plus, the running time of our model compares favorably with existing models.

The rest of this paper is organized as following. In Section 2, we describe our sentence embedding algorithm GEM. We evaluate our model on various tasks in Section 3 and Section 4. Finally, we summarize our work in Section 5. Our implementation is available online[1].

## 2 Approach

We introduce three scores to quantify the importance of a word, as will be explained in this section. First, **novelty score** $\alpha_n$ measures the portion of the new semantic meaning in a word. Second, **significance score** $\alpha_s$ describes the alignment between the new semantic meaning and the sentence-level meaning. Finally, **uniqueness score** $\alpha_u$ examines the uniqueness of the new semantic meaning in the corpus level.

### 2.1 Quantify New Semantic Meaning

Let us consider the idea of word embeddings (Mikolov et al., 2013), where a word $w_i$ is projected as a vector $\boldsymbol{v}_{w_i} \in \mathbb{R}^d$. Any sequence of words can be viewed as a subspace in $\mathbb{R}^d$ spanned by its word vectors. Before the appearance of the $i$th word, $\boldsymbol{S}$ is a subspace in $\mathbb{R}^d$ spanned by $\{\boldsymbol{v}_{w_1}, \boldsymbol{v}_{w_2}, ..., \boldsymbol{v}_{w_{i-1}}\}$. Its orthonormal basis is $\{\boldsymbol{q}_1, \boldsymbol{q}_2, ..., \boldsymbol{q}_{i-1}\}$. The embedding $\boldsymbol{v}_{w_i}$ of the $i$th word $w_i$ can be decomposed into

$$\begin{aligned} \boldsymbol{v}_{w_i} &= \sum_{j=1}^{i-1} r_j \boldsymbol{q}_j + r_i \boldsymbol{q}_i \\ r_j &= \boldsymbol{q}_j^T \boldsymbol{v}_{w_i} \\ r_i &= \|\boldsymbol{v}_{w_i} - \sum_{j=1}^{i-1} r_j \boldsymbol{q}_j\|_2 \end{aligned} \tag{1}$$

[1] https://github.com/ziyi-yang/GEM

where $\sum_{j=1}^{i-1} r_j \boldsymbol{q}_j$ is the part in $\boldsymbol{v}_{w_i}$ that resides in subspace $\boldsymbol{S}$, and $\boldsymbol{q}_i$ is orthogonal to $\boldsymbol{S}$ and is to be added to $\boldsymbol{S}$. The above algorithm is also known as **Gram-Schmidt Process**. In the case of rank deficiency, i.e., $\boldsymbol{v}_{w_i}$ is already a linear combination of $\{\boldsymbol{q}_1, \boldsymbol{q}_2, ...\boldsymbol{q}_{i-1}\}$, $\boldsymbol{q}_i$ is a zero vector and $r_i = 0$. In matrix form, this process is also known as *QR factorization*, defined as follows.

**QR factorization.** Define an embedding matrix of $n$ words as $\boldsymbol{A} = [\boldsymbol{A}_{:,1}, \boldsymbol{A}_{:,2}, ..., \boldsymbol{A}_{:,n}] \in \mathbb{R}^{d\times n}$, where $\boldsymbol{A}_{:,i}$ is the embedding of the $i$th word $w_i$ in a word sequence $(w_1, \ldots, w_i, \ldots, w_n)$. $\boldsymbol{A} \in \mathbb{R}^{d\times n}$ can be factorized into $\boldsymbol{A} = \boldsymbol{QR}$, where the non-zero columns in $\boldsymbol{Q} \in \mathbb{R}^{d\times n}$ are the orthonormal basis, and $\boldsymbol{R} \in \mathbb{R}^{n\times n}$ is an upper triangular matrix.

The process above computes the novel semantic meaning of a word w.r.t all preceding words. As the meaning of a word influences and is influenced by its close neighbors, we now calculate the novel orthogonal basis vector $\boldsymbol{q}_i$ of each word $w_i$ in its neighborhood, rather than only w.r.t the preceding words.

**Definition 1 (Contextual Window Matrix)** *Given a word $w_i$, and its $m$-neighborhood window inside the sentence $(w_{i-m}, \ldots, w_{i-1}, w_i, w_{i+1}, \ldots, w_{i+m})$ , define the contextual window matrix of word $w_i$ as:*

$$\boldsymbol{S}^i = [\boldsymbol{v}_{w_{i-m}} ... \boldsymbol{v}_{w_{i-1}}, \boldsymbol{v}_{w_{i+1}} ... \boldsymbol{v}_{w_{i+m}}, \boldsymbol{v}_{w_i}] \quad (2)$$

Here we shuffle $\boldsymbol{v}_{w_i}$ to the end of $\boldsymbol{S}^i$ to compute its novel semantic information compared with its context. Now the QR factorization of $\boldsymbol{S}^i$ is

$$\boldsymbol{S}^i = \boldsymbol{Q}^i \boldsymbol{R}^i \quad (3)$$

Note that $\boldsymbol{q}_i$ is the last column of $\boldsymbol{Q}^i$, which is also the new orthogonal basis vector to this contextual window matrix.

Next, in order to generate the embedding for a sentence, we will assign a weight to each of its words. This weight should characterize how much new and important information a word brings to the sentence. The previous process yields the orthogonal basis vector $\boldsymbol{q}_i$. We propose that $\boldsymbol{q}_i$ represents the novel semantic meaning brought by word $w_i$. We will now discuss how to quantify i) the novelty of $\boldsymbol{q}_i$ to other meanings in $w_i$, ii) the significance of $\boldsymbol{q}_i$ to its context, and iii) the corpus-wise uniqueness of $\boldsymbol{q}_i$ w.r.t the whole corpus.

### 2.2 Novelty

We propose that a word $w_i$ is more important to a sentence if its novel orthogonal basis vector $\boldsymbol{q}_i$ is a large component in $\boldsymbol{v}_{w_i}$, quantified by the proposed novelty score $\alpha_n$. Let $\boldsymbol{r}$ denote the last column of $\boldsymbol{R}^i$, and $\boldsymbol{r}_{-1}$ denote the last element of $\boldsymbol{r}$, $\alpha_n$ is defined as:

$$\alpha_n = \exp(\frac{\|\boldsymbol{q}_i\|_2}{\|\boldsymbol{v}_{w_i}\|_2}) = \exp(\frac{\boldsymbol{r}_{-1}}{\|\boldsymbol{r}\|_2}) \quad (4)$$

Note that $\|\boldsymbol{q}_i\|_2 = \boldsymbol{r}_{-1}$ and $\|\boldsymbol{v}_{w_i}\|_2 = \|\boldsymbol{r}\|_2$. One can show that $\alpha_n$ is the exponential of the normalized distance between $\boldsymbol{v}_{w_i}$ and the subspace spanned by its context.

### 2.3 Significance

The significance of a word is related to how semantically aligned it is to the meaning of its context. To identify principal directions, i.e. meanings, in the contextual window matrix $\boldsymbol{S}^i$, we employ *Singular Value Decomposition*.

**Singular Value Decomposition.** Given a matrix $\boldsymbol{A} \in \mathbb{R}^{d\times n}$, there exists $\boldsymbol{U} \in \mathbb{R}^{d\times n}$ with orthogonal columns, diagonal matrix $\boldsymbol{\Sigma} = \text{diag}(\sigma_1, ..., \sigma_n)$, $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n \geq 0$, and orthogonal matrix $\boldsymbol{V} \in \mathbb{R}^{n\times n}$, such that $\boldsymbol{A} = \boldsymbol{U\Sigma V}^T$.

The columns of $\boldsymbol{U}$, $\{\boldsymbol{U}_{:,j}\}_{j=1}^n$, are an orthonormal basis of $\boldsymbol{A}$'s columns subspace and we propose that they represent a set of semantic meanings from the context. Their corresponding singular values $\{\sigma_j\}_{j=1}^n$, denoted by $\sigma(\boldsymbol{A})$, represent the importance associated with $\{\boldsymbol{U}_{:,j}\}_{j=1}^n$. The SVD of $w_i$'s contextual window matrix is $\boldsymbol{S}^i = \boldsymbol{U}^i\boldsymbol{\Sigma}^i\boldsymbol{V}^{iT} \in \mathbb{R}^{d\times(2m+1)}$. It follows that $\boldsymbol{q}_i^T\boldsymbol{U}^i$ is the coordinate of $\boldsymbol{q}_i$ in the basis of $\{\boldsymbol{U}^i_{:,j}\}_{j=1}^{2m+1}$.

Intuitively, a word is more important if its novel semantic meaning has a better alignment with more principal meanings in its contextual window. This can be quantified as $\|\sigma(\boldsymbol{S}^i) \odot (\boldsymbol{q}_i^T\boldsymbol{U}^i)\|_2$, where $\odot$ denotes element-wise product. Therefore, we define the significance of $w_i$ in its context to be:

$$\alpha_s = \frac{\|\sigma(\boldsymbol{S}^i) \odot (\boldsymbol{q}_i^T\boldsymbol{U}^i)\|_2}{2m+1} \quad (5)$$

It turns out $\alpha_s$ can be rewritten as

$$\begin{aligned}\alpha_s &= \frac{\|\boldsymbol{q}_i^T\boldsymbol{U}^i\boldsymbol{\Sigma}^i\|_2}{2m+1} = \frac{\|\boldsymbol{q}_i^T\boldsymbol{U}^i\boldsymbol{\Sigma}^i\boldsymbol{V}^i\|_2}{2m+1} \\ &= \frac{\|\boldsymbol{q}_i^T\boldsymbol{S}^i\|_2}{2m+1} = \frac{\boldsymbol{q}_i^T\boldsymbol{v}_{w_i}}{2m+1} = \frac{\boldsymbol{r}_{-1}}{2m+1}\end{aligned} \quad (6)$$

and we use the fact that $\boldsymbol{V}^i$ is an orthogonal matrix and $\boldsymbol{q}_i$ is orthogonal to all but the last column of $\boldsymbol{S}^i$, $\boldsymbol{v}_{w_i}$. Therefore, $\alpha_s$ is essentially the distance between $\boldsymbol{w}_i$ and the context hyper-plane, normalized by the context size.

Although $\alpha_s$ and $\alpha_n$ look alike in mathematics form, they model distinct quantities in word $w_i$ against its contextual window. $\alpha_n$ is a function of $\|\boldsymbol{q}_i\|_2$ divided by $\|\boldsymbol{w}_i\|_2$, i.e., the portion of the new semantic meaning in word $w_i$. In contrast, eq. (6) shows that $\alpha_s$ equals $\|\boldsymbol{q}_i\|_2$ divided by a constant, namely $\alpha_s$ quantifies the absolute magnitude of the new semantic meaning $\boldsymbol{q}_i$.

### 2.4 Corpus-wise Uniqueness

Similar to the idea of inverse document frequency (IDF) (Sparck Jones, 1972), a word that is commonly present in the corpus is likely to be a stop word, thus its corpus-wise uniqueness is small. In our solution, we compute the principal directions of the corpus and then measure their alignment with the novel orthogonal basis vector $\boldsymbol{q}_i$. If there is a high alignment, $w_i$ will be assigned a relatively low corpus-wise uniqueness score, and vice versa.

#### 2.4.1 Compute Principal Directions of Corpus

In Arora et al. (2017), given a corpus containing a set of $N$ sentences, an embedding matrix $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N] \in \mathbb{R}^{d \times N}$ is generated, where $\boldsymbol{x}_i$ is the sentence embedding for the $i$-th sentence in the corpus, computed by SIF algorithm. Then principal vectors of $\boldsymbol{X}$ are computed and projections onto the principal vectors are removed from each sentence embedding $\boldsymbol{x}_i$.

In contrast to Arora et al. (2017), we do not form the embedding matrix after we obtain the final sentence representation. Instead, we obtain an intermediate coarse-grained embedding matrix $\boldsymbol{X}^c = [\boldsymbol{g}_1, \ldots, \boldsymbol{g}_N]$ as follows. Suppose the SVD of the sentence matrix of the $i$th sentence is $\boldsymbol{S} = [\boldsymbol{v}_{w_1}, \ldots, \boldsymbol{v}_{w_n}] = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$. Then the coarse-grained embedding for the $i$th sentence is defined as:

$$\boldsymbol{g}_i = \sum_{j=1}^{n} f(\sigma_j)\boldsymbol{U}_{:,j} \tag{7}$$

where $f(\sigma_j)$ is a monotonically increasing function. We then compute the top $K$ principal vectors $\{\boldsymbol{d}_1, ..., \boldsymbol{d}_K\}$ of $\boldsymbol{X}^c$, with singular values $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_K$.

#### 2.4.2 Uniqueness Score

In contrast to Arora et al. (2017), we select different principal vectors of $\boldsymbol{X}^c$ for each sentence, as different sentences may have disparate alignments with the corpus. For each sentence, $\{\boldsymbol{d}_1, ..., \boldsymbol{d}_K\}$ are re-ranked in descending order of their correlation with sentence matrix $\boldsymbol{S}$. The correlation is defined as:

$$o_i = \sigma_i \|\boldsymbol{S}^T \boldsymbol{d}_i\|_2, 1 \leq i \leq K \tag{8}$$

Next, the top $h$ principal vectors after re-ranking based on $o_i$ are selected: $\boldsymbol{D} = \{\boldsymbol{d}_{t_1}, ..., \boldsymbol{d}_{t_h}\}$, with $o_{t_1} \geq o_{t_2} \geq ... \geq o_{t_h}$ and their singular values in $\boldsymbol{X}^c$ are $\boldsymbol{\sigma}_d = [\sigma_{t_1}, ..., \sigma_{t_h}] \in \mathbb{R}^h$.

Finally, a word $w_i$ with new semantic meaning vector $\boldsymbol{q}_i$ in this sentence will be assigned a corpus-wise uniqueness score:

$$\alpha_u = \exp\left(-\|\boldsymbol{\sigma}_d \odot (\boldsymbol{q}_i^T \boldsymbol{D})\|_2 / h\right) \tag{9}$$

This ensures that common stop words will have their effect diminished since their embeddings are closely aligned with the corpus' principal directions.

### 2.5 Sentence Vector

A sentence vector $\boldsymbol{c}_s$ is computed as a weighted sum of its word embeddings, where the weights come from three scores: a novelty score ($\alpha_n$), a significance score ($\alpha_s$) and a corpus-wise uniqueness score ($\alpha_u$).

$$\begin{aligned} \alpha_i &= \alpha_n + \alpha_s + \alpha_u \\ \boldsymbol{c}_s &= \sum_i \alpha_i \boldsymbol{v}_{w_i} \end{aligned} \tag{10}$$

We provide a theoretical explanation of Equation (10) in Appendix.

**Sentence-Dependent Removal of Principal Components.** Arora et al. (2017) shows that given a set of sentence vectors, removing projections onto the principal components of the spanned subspace can significantly enhance the performance on semantic similarity task. However, as each sentence may have a different semantic meaning, it could be sub-optimal to remove the same set of principal components from all sentences.

Therefore, we propose the *sentence-dependent principal component removal* (SDR), where we re-rank top principal vectors based on correlation

with each sentence. Using the method from Section 2.4.2, we obtain $\boldsymbol{D} = \{\boldsymbol{d}_{t_1}, ..., \boldsymbol{d}_{t_r}\}$ for a sentence $s$. The final embedding of this sentence is then computed as:

$$\boldsymbol{c}_s \leftarrow \boldsymbol{c}_s - \sum_{j=1}^{r} (\boldsymbol{d}_{t_j}^T \boldsymbol{c}_s) \boldsymbol{d}_{t_j} \quad (11)$$

Ablation experiments show that sentence-dependent principal component removal can achieve better result. The complete algorithm is summarized in Algorithm 1 with an illustration in Figure 1.

### 2.6 Handling of out-of-vocabulary Words

In many NLP algorithms, the out-of-vocabulary (OOV) words are projected to a special "UNK" token. However, in this way, different OOV words with drastically different meanings will share the same embedding. To fix this problem, we change this projection method by mapping OOVs to pre-trained in-vocabulary words, based on a hash function SHA-256 of its characters. In this way, two different OOV words will almost certainly have different embeddings. In the experiments, we apply this OOV projection technique in both STS-B and CQA tasks.

## 3 Experiments

### 3.1 Semantic Similarity Tasks: STS Benchmark

We evaluate our model on the STS Benchmark (Cer et al., 2017), a sentence-level semantic similarity dataset. The goal for a model is to predict a similarity score of two sentences given a sentence pair. The evaluation is by the Pearson's coefficient $r$ between human-labeled similarity (0 - 5 points) and predictions.

**Experimental settings.** We report two versions of our model, one only using GloVe word vectors (GEM + GloVe), and the other using word vectors concatenated from LexVec, fastText and PSL (Wieting et al., 2015b) (GEM + L.F.P). The final similarity score is computed as an inner product of normalized sentence vectors. Since our model is non-parameterized, it does not utilize any information from the dev set when evaluating on the test set and vice versa. Hyper-parameters are chosen at $m = 7$, $h = 17$, $K = 45$, and $t = 3$ by conducing hyper-parameters search on dev set. Results on the dev and test set are reported in Table 1. As shown,

| Non-parameterized models | dev | test |
|---|---|---|
| **GEM + L.F.P (ours)** | 83.5 | 78.4 |
| **GEM + LexVec (ours)** | 81.9 | 76.5 |
| SIF (Arora et al., 2017) | 80.1 | 72.0 |
| uSIF (Ethayarajh, 2018) | **84.2** | 79.5 |
| LexVec | 58.78 | 50.43 |
| L.F.P | 62.4 | 52.0 |
| word2vec skipgram | 70.0 | 56.5 |
| Glove | 52.4 | 40.6 |
| ELMo | 64.6 | 55.9 |
| Parameterized models | | |
| PARANMT-50M (Wieting and Gimpel, 2017a) | - | **79.9** |
| Reddit + SNLI (Yang et al., 2018) | 81.4 | 78.2 |
| GRAN (Wieting and Gimpel, 2017b) | 81.8 | 76.4 |
| InferSent (Conneau et al., 2017) | 80.1 | 75.8 |
| Sent2Vec (Pagliardini et al., 2018) | 78.7 | 75.5 |
| Paragram-Phrase (Wieting et al., 2015a) | 73.9 | 73.2 |

Table 1: Pearson's $r \times 100$ on STSB. Best results are in bold.

| | |
|---|---|
| **GEM + L.F.P (ours)** | **49.11** |
| Reddit + SNLI tuned | 47.44 |
| KeLP-contrastive1 | 49.00 |
| SimBow-contrastive2 | 47.87 |
| SimBow-primary | 47.22 |

Table 2: MAP on CQA subtask B.

on the test set, our model has a $6.4\%$ higher score compared with another non-parameterized model SIF, and $26.4\%$ higher than the baseline of averaging L.F.P word vectors. It also outperforms all parameterized models including GRAN, InferSent, Sent2Vec and Reddit+SNLI.

### 3.2 Semantic Similarity Tasks: CQA

We evaluate our model on subtask B of the SemEval Community Question Answering (CQA) task, another semantic similarity dataset. Given an original question $Q_o$ and a set of the first ten related questions $(Q_1, ..., Q_{10})$ retrieved by a search engine, the model is expected to re-rank the related questions according to their similarity with respect to the original question. Each retrieved question $Q_i$ is labelled "PerfectMatch", "Relevant" or "Irrelevant", with respect to $Q_o$. Mean average precision (MAP) is used as the evaluation measure.

We encode each question text into a unit vector $\boldsymbol{u}$. Retrieved questions $\{Q_i\}_{i=1}^{10}$ are ranked according to their cosine similarity with $Q_o$. Results are shown in Table 2. For comparison, we include results from the best models in 2017 competition: SimBow (Charlet and Damnati, 2017),

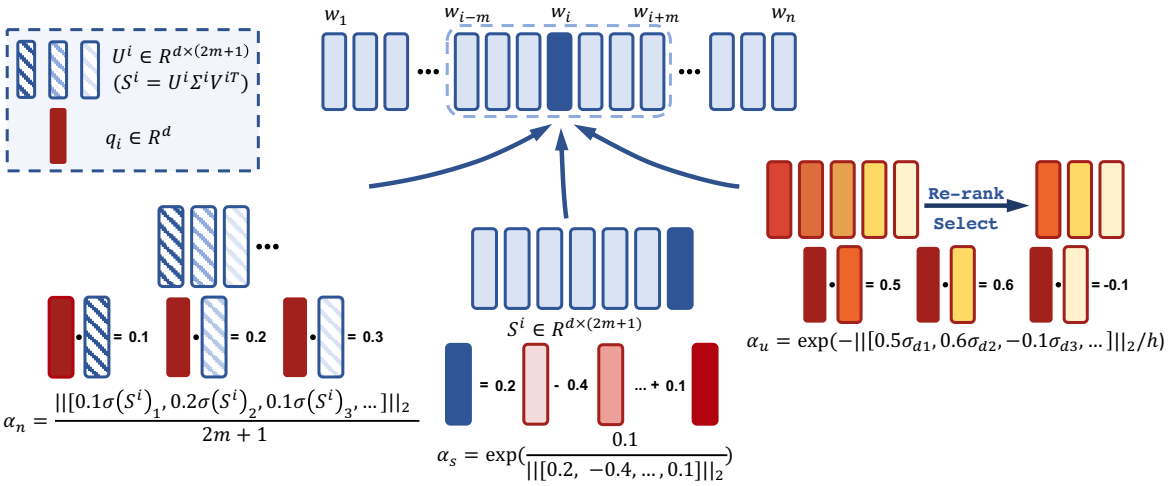


Figure 1: An illustration of GEM algorithm. Top middle: The sentence to encode, with words $w_1$ to $w_n$. The contextual window of word $w_i$ is inside the dashed line. Bottom middle: Form contextual window matrix $\boldsymbol{S}^i$ for $w^i$, compute $\boldsymbol{q}_i$ and novelty score $\alpha_n$ (Section 2.1 and Section 2.2). Bottom left: SVD of $\boldsymbol{S}^i$ and compute the significance score $\alpha_s$ (Section 2.3). Bottom right: Re-rank and select from principal components (orange blocks) and compute uniqueness score $\alpha_u$ (Section 2.4).

**Algorithm 1** Geometric Embedding (GEM)

**Inputs:**

A set of sentences $\mathcal{S}$, vocabulary $\mathcal{V}$, word embeddings $\{\boldsymbol{v}_w \in \mathbb{R}^d \,|\, w \in \mathcal{V}\}$

**Outputs:**

Sentence embeddings $\{\boldsymbol{c}_s \in \mathbb{R}^d \,|\, s \in \mathcal{S}\}$

**for** $i$th sentence $s$ in $\mathcal{S}$ **do**

Form matrix $\boldsymbol{S} \in \mathbb{R}^{d\times n}$, $\boldsymbol{S}_{:,j} = \boldsymbol{v}_{w_j}$ and $w_j$ is the $j$th word in $s$

The SVD is $\boldsymbol{S} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$

The $i$th column of the coarse-grained sentence embedding matrix $\boldsymbol{X}^c_{:,i}$ is $\boldsymbol{U}(\sigma(\boldsymbol{S}))^3$

**end for**

Take first $K$ singular vectors $\{\boldsymbol{d}_1, ..., \boldsymbol{d}_K\}$ and singular values $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_K$ of $\boldsymbol{X}^c$

**for** sentence $s$ in $\mathcal{S}$ **do**

Re-rank $\{\boldsymbol{d}_1, ..., \boldsymbol{d}_K\}$ in descending order by $o_i = \sigma_i \|\boldsymbol{S}^T\boldsymbol{d}_i\|_2, 1 \leq i \leq K$.

Select top $h$ principal vectors as $\boldsymbol{D} = [\boldsymbol{d}_{t_1}, ..., \boldsymbol{d}_{t_h}]$, with singular values $\boldsymbol{\sigma}_d = [\sigma_{t_1}, ..., \sigma_{t_h}]$.

**for** word $w_i$ in $s$ **do**

$\boldsymbol{S}^i = [\boldsymbol{v}_{w_{i-m}}, ..., \boldsymbol{v}_{w_{i-1}}, \boldsymbol{v}_{w_{i+1}}, ..., \boldsymbol{v}_{w_{i+m}}, \boldsymbol{v}_{w_i}]$ is the contextual window matrix of $w_i$.

Do QR decomposition $\boldsymbol{S}^i = \boldsymbol{Q}^i\boldsymbol{R}^i$, let $\boldsymbol{q}_i$ and $\boldsymbol{r}$ denote the last column of $\boldsymbol{Q}^i$ and $\boldsymbol{R}^i$

$\alpha_n = \exp(\boldsymbol{r}_{-1}/\|\boldsymbol{r}\|_2), \alpha_s = \boldsymbol{r}_{-1}/(2m+1), \alpha_u = \exp\left(-\|\boldsymbol{\sigma}_d \odot (\boldsymbol{q}_i^T\boldsymbol{D})\|_2/h\right)$

$\alpha_i = \alpha_n + \alpha_s + \alpha_u$

**end for**

$\boldsymbol{c}_s = \sum_{\boldsymbol{v}_i \in \boldsymbol{s}} \alpha_i \boldsymbol{v}_{w_i}$

Principal vectors removal: $\boldsymbol{c}_s \leftarrow \boldsymbol{c}_s - \boldsymbol{D}\boldsymbol{D}^T\boldsymbol{c}_s$

**end for**

KeLP (Filice et al., 2017), and Reddit + SNLI tuned. Note that all three benchmark models require learning on CQA training set, and SimBow and KeLP leverage optional features including usage of comments and user profiles. In comparison, our model only uses the question text without any training. Our model clearly outperforms Reddit + SNLI tuned, SimBow-primary and KeLP model.

### 3.3 Supervised tasks

We further test our model on nine supervised tasks, including seven classification tasks: movie review (MR) (Pang and Lee, 2005), Stanford Sentiment Treebank (SST) (Socher et al., 2013),

question-type classification (TREC) (Voorhees and Dang, 2003), opinion polarity (MPQA) (Wiebe et al., 2005), product reviews (CR) (Hu and Liu, 2004), subjectivity/objectivity classification (SUBJ) (Pang and Lee, 2004) and paraphrase identification (MRPC) (Dolan et al., 2004). We also evaluate on two entailment and semantic relatedness tasks: SICK similarity (SICK-R) and the SICK entailment (SICK-E) (Marelli et al., 2014). The sentence embeddings generated are fixed and only the downstream task-specific neural structure is learned. For classification tasks, a linear classifier is trained on top, following Kiros et al. (2015), and classification accuracy are reported. For relatedness tasks, we follow Tai et al. (2015) to train a logistic regression to learn the probability distribution of relatedness scores, and we report Pearson's correlation. The four hyper-parameters are chosen the same as those in STS benchmark experiment. For fair comparison, embeddings models are divided into two categories: non-parameterized and parameterized ones, as described in section 1. Results are shown in Table 3.

GEM outperforms all other non-parameterized sentence embedding models, including SIF, p-mean (Rücklé et al., 2018), and BOW on GloVe. The consistent superior performance again demonstrates GEM's advantage on weighting scheme. It also compares favorably with most of parameterized models, including *à la carte* (Khodak et al., 2018), FastSent (Hill et al., 2016), InferSent, QT, Sent2Vec, SkipThought-LN (with layer normalization) (Kiros et al., 2015), SDAE (Hill et al., 2016), STN (Subramanian et al., 2018) and USE (Yang et al., 2018). Note that sentence representations generated by GEM have much smaller dimension compared to most of benchmark models, and the subsequent neural structure has fewer trainable parameters. This observation suggests that local multi-word level information in sentences has already provided revealing information for sophisticated downstream tasks. The fact that GEM does well on several classification tasks (e.g. TREC and SUBJ) indicates that the proposed weight scheme is able to recognize important words sentences. Also, GEM's competitive performance on sentiment tasks shows that exploiting the geometric structures of two sentence subspaces is semantically informative.

## 4 Discussion

**Comparison with Arora et al. (2017).** We would like to point out that although sharing the idea of modelling the sentence as the weighted sum of its word vectors, GEM is substantially different from Arora et al. (2017). First, we adopt well-established numerical linear algebra to quantify the semantic meaning and importance of words in the sentences context. And this new approach proves to be effective. Second, the weights in SIF (and uSIF) are calculated from the statistic of vocabularies on very large corpus (wikipedia). In contrast, the weights in GEM are directly computed from the sentences themselves along with the dataset, independent with prior statistical knowledge of language or vocabularies. Furthermore, the components in GEM's weights are derived from numerical linear algebra eq. (4) to (9), while SIF directly includes a hyper-parameter term in its weight scheme, i.e. its smooth term.

**Robustness and Effectiveness.** Besides experiments mentioned above, we also test the robustness and effectiveness GEM on several simple but non-trivial examples. These experiments demonstrate that GEM is quite insensitive against the removal of non-important stop words. Also GEM can correctly assign higher weights for words with more significant semantic meanings in the sentence.

We first test the robustness of GEM by removing one non-important stop word in a sentence and computed the similarity between the original sentence and the one after removal. For example:

- original sentence: "the first tropical system to slam the US this year is expected to make landfall as a hurricane"
- remove 7 stop words: "first tropical system slam US this year expected make landfall hurricane"

The cosine similarity between these two sentences given by GEM is 0.954. Even though aggressively removing 7 stop words, GEM still assigns pretty similar embeddings for these two sentences.

We further demonstrate that GEM does assign higher weights to words with more significant semantic meanings. Consider the following sentence: "there are two ducks swimming in the river". Weights assigned by GEM are (sorted from high to low): [ducks: 4.93, river:4.72 , swimming: 4.70, two: 3.87, are: 3.54, there:

| Model | Dim | Training time (h) | MR | CR | SUBJ | MPQA | SST | TREC | MRPC | SICK-R | SICK-E |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *Non-parameterized models* | | | | | | | | | | | |
| **GEM + L.F.P (ours)** | 900 | 0 | **79.8** | **82.5** | **93.8** | **89.9** | **84.7** | **91.4** | **75.4/82.9** | **86.5** | **86.2** |
| **GEM + GloVe (ours)** | 300 | 0 | 78.8 | 81.1 | 93.1 | 89.4 | 83.6 | 88.6 | 73.4/82.3 | 86.3 | 85.3 |
| SIF | 300 | 0 | 77.3 | 78.6 | 90.5 | 87.0 | 82.2 | 78.0 | - | 86.0 | 84.6 |
| uSIF | 300 | 0 | - | - | - | - | 80.7 | - | - | 83.8 | 81.1 |
| p-mean | 3600 | 0 | 78.4 | 80.4 | 93.1 | 88.9 | 83.0 | 90.6 | - | - | - |
| GloVe BOW | 300 | 0 | 78.7 | 78.5 | 91.6 | 87.6 | 79.8 | 83.6 | 72.1/80.9 | 80.0 | 78.6 |
| *Paraemterized models* | | | | | | | | | | | |
| InferSent | 4096 | 24 | 81.1 | 86.3 | 92.4 | 90.2 | 84.6 | 88.2 | 76.2/83.1 | 88.4 | 86.3 |
| Sent2Vec | 700 | 6.5 | 75.8 | 80.3 | 91.1 | 85.9 | - | 86.4 | 72.5/80.8 | - | - |
| SkipThought-LN | 4800 | 336 | 79.4 | 83.1 | 93.7 | 89.3 | 82.9 | 88.4 | - | 85.8 | 79.5 |
| FastSent | 300 | 2 | 70.8 | 78.4 | 88.7 | 80.6 | - | 76.8 | 72.2/80.3 | - | - |
| *à la carte* | 4800 | N/A | 81.8 | 84.3 | 93.8 | 87.6 | 86.7 | 89.0 | - | - | - |
| SDAE | 2400 | 192 | 74.6 | 78.0 | 90.8 | 86.9 | - | 78.4 | 73.7/80.7 | - | - |
| QT | 4800 | 28 | 82.4 | 86.0 | **94.8** | 90.2 | **87.6** | 92.4 | 76.9/84.0 | 87.4 | - |
| STN | 4096 | 168 | **82.5** | **87.7** | 94.0 | **90.9** | 83.2 | 93.0 | **78.6/84.4** | **88.8** | **87.8** |
| USE | 512 | N/A | 81.36 | 86.08 | 93.66 | 87.14 | 86.24 | **96.60** | - | - | - |

Table 3: Results on supervised tasks. Sentence embeddings are fixed for downstream supervised tasks. Best results for each task are underlined, best results from models in the same category are in bold. SIF results are extracted from Arora et al. (2017) and Rücklé et al. (2018), and training time is collected from Logeswaran and Lee (2018).

3.23, in:3.04, the:2.93]. GEM successfully assigns higher weight to informative words like ducks and river, and downplay stop words like the and there. More examples can be found in the Appendix.

**Ablation Study.** As shown in in Table 4, every GEM weight $(\alpha_n, \alpha_s, \alpha_u)$ and proposed principal components removal methods contribute to the performance. As listed on the left, adding GEM weights improves the score by 8.6% on STS dataset compared with averaging three concatenated word vectors. The sentence-dependent principal component removal (SDR) proposed in GEM improves 1.7% compared to directly removing the top $h$ corpus principal components (SIR). Using GEM weights and SDR together yields an overall improvement of 21.1%. As shown on the right in Table 4, every weight contributes to the performance of our model. For example, three weights altogether improve the score in SUBJ task by 0.38% compared with only using $\alpha_n$.

**Sensitivity Study.** We evaluate the effect of all four hyper-parameters in our model: the window size $m$ in the contextual window matrix, the number of candidate principal components $K$, the number of principal components to remove $h$, and the power of the singular value in coarse sentence embedding, i.e. the power $t$ in $f(\sigma_j) = \sigma_j^t$ in Equation (7). We sweep the hyper-parameters and test on STSB dev set, SUBJ, and MPQA. Unspecified parameters are fixed at $m = 7$, $K = 45$, $h = 17$ and $t = 3$. As shown in Figure 2, our model is quite robust with respect to hyper-parameters.

| Configurations | STSB dev | SUBJ |
|---|---|---|
| Mean of L.F.P | 62.4 | - |
| GEM weights | 71.0 | - |
| GEM weights + SIR | 81.8 | - |
| GEM weights + SDR | 83.5 | - |
| $\alpha_n$ + SDR | 81.6 | 93.42 |
| $\alpha_n, \alpha_s$ + SDR | 81.9 | 93.6 |
| $\alpha_n, \alpha_s, \alpha_u$ + SDR | 83.5 | 93.8 |

Table 4: Comparison of different configurations demonstrates the effectiveness of our model on STSB dev set and SUBJ. SDR stands for sentence-dependent principal component removal in Section 2.4.2. SIR stands for sentence-independent principal component removal, i.e. directly removing top $h$ corpus principal components from the sentence embedding.

**Inference speed.** We also compare the inference speed of our algorithm on the STSB test set with the benchmark models SkipThought and InferSent. SkipThought and InferSent are run on a NVIDIA Tesla P100 GPU, and our model is run on a CPU (Intel Xeon CPU E5-2690 v4 @2.60GHz). For fair comparison, batch size in InferSent and SkipThought is set to be 1. The results are shown in Table 5. It shows that without acceleration from GPU, our model is still faster than InferSent and is 54% faster than SkipThought.

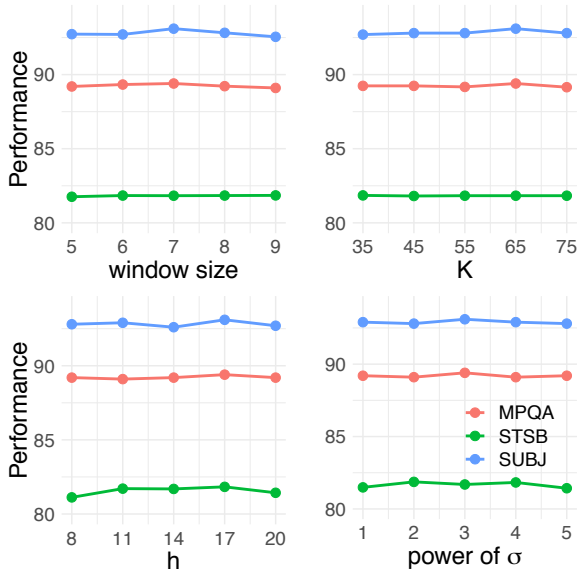


Figure 2: Sensitivity tests on four hyper-parameters, the window size $m$ in contextual window matrix, the number of candidate principal components $K$, the number of principal components to remove $h$, and the exponential power of singular value in coarse sentence embedding.

| | Average run time (s) | Variance |
|---|---|---|
| GEM (CPU) | **20.08** | 0.23 |
| InferSent(GPU) | 21.24 | 0.15 |
| SkipThought (GPU) | 43.36 | 0.10 |

Table 5: Run time of GEM, InferSent and SkipThought on encoding sentences in STSB test set. GEM is run on CPU, InferSent and SkipThought is run on GPU. Data are collected from 5 trials.

## 5 Conclusions

We proposed a simple non-parameterized method to generate sentence embeddings, based entirely on the geometric structure of the subspace spanned by word embeddings. Our sentence embedding evolves from the new orthogonal basis vector brought in by each word, which represents novel semantic meaning. The evaluation shows that our method not only sets up the new state-of-the-art of non-parameterized models but also performs competitively when compared with models requiring either large amount of training data or prolonged training time. In future work, we plan to consider subwords into the model and explore more geometric structures in sentences.

## Acknowledgments

We would like to thank Jade Huang for proofreading the paper and helpful writing suggestions. We also acknowledge the anonymous reviewers for their valuable feedback.

## References

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. *International Conference on Learning Representations*.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Delphine Charlet and Geraldine Damnati. 2017. Simbow at semeval-2017 task 3: Soft-cosine semantic similarity between questions for community question answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 315–319.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th international conference on Computational Linguistics*, page 350. Association for Computational Linguistics.

Kawin Ethayarajh. 2018. Unsupervised random walk sentence embeddings: A strong but simple baseline. In *Proceedings of The Third Workshop on Representation Learning for NLP*, pages 91–100.

Simone Filice, Giovanni Da San Martino, and Alessandro Moschitti. 2017. Kelp at semeval-2017 task 3: Learning pairwise patterns in community question answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 326–333.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. *arXiv preprint arXiv:1602.03483*.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Mikhail Khodak, Nikunj Saunshi, Yingyu Liang, Tengyu Ma, Brandon Stewart, and Sanjeev Arora. 2018. A la carte embedding: Cheap but effective induction of semantic feature vectors. *arXiv preprint arXiv:1805.05388*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Lajanugen Logeswaran and Honglak Lee. 2018. An efficient framework for learning sentence representations. *International Conference on Learning Representations*.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, pages 1–8.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2018. Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features. In *NAACL 2018 - Conference of the North American Chapter of the Association for Computational Linguistics*.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 115–124. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

A. Rücklé, S. Eger, M. Peyrard, and I. Gurevych. 2018. Concatenated $p$-mean Word Embeddings as Universal Cross-Lingual Sentence Representations. *ArXiv e-prints*.

Alexandre Salle, Aline Villavicencio, and Marco Idiart. 2016. Matrix factorization using window sampling and negative sampling for improved word representations. *CoRR*, abs/1606.00819.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21.

Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J Pal. 2018. Learning general purpose distributed sentence representations via large scale multi-task learning. *arXiv preprint arXiv:1804.00079*.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Ellen M Voorhees and Hoa Trang Dang. 2003. Overview of the trec 2003 question answering track. In *TREC*, volume 2003, pages 54–68.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39(2-3):165–210.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015a. Towards universal paraphrastic sentence embeddings. In *International Conference on Learning Representation*.

John Wieting, Mohit Bansal, Kevin Gimpel, Karen Livescu, and Dan Roth. 2015b. From paraphrase database to compositional paraphrase model and back. *arXiv preprint arXiv:1506.03487*.

John Wieting and Kevin Gimpel. 2017a. Paranmt-50m: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. *arXiv preprint arXiv:1711.05732*.

John Wieting and Kevin Gimpel. 2017b. Revisiting recurrent networks for paraphrastic sentence embeddings. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Y. Yang, S. Yuan, D. Cer, S.-y. Kong, N. Constant, P. Pilar, H. Ge, Y.-H. Sung, B. Strope, and R. Kurzweil. 2018. Learning Semantic Textual Similarity from Conversations. *ArXiv e-prints*.

# Evaluation Benchmarks and Learning Criteria for Discourse-Aware Sentence Representations

**Mingda Chen**[2*] **Zewei Chu**[1*] **Kevin Gimpel**[2]
[1]University of Chicago, IL, USA
[2]Toyota Technological Institute at Chicago, IL, USA
{mchen,kgimpel}@ttic.edu,zeweichu@uchicao.edu

## Abstract

Prior work on pretrained sentence embeddings and benchmarks focuses on the capabilities of representations for stand-alone sentences. We propose DiscoEval, a test suite of tasks to evaluate whether sentence representations include information about the role of a sentence in its discourse context. We also propose a variety of training objectives that make use of natural annotations from Wikipedia to build sentence encoders capable of modeling discourse information. We benchmark sentence encoders trained with our proposed objectives, as well as other popular pretrained sentence encoders, on DiscoEval and other sentence evaluation tasks. Empirically, we show that these training objectives help to encode different aspects of information from the surrounding document structure. Moreover, BERT (Devlin et al., 2019) and ELMo (Peters et al., 2018a) demonstrate strong performance across DiscoEval tasks with individual hidden layers showing different characteristics.[1]

## 1 Introduction

Pretrained sentence representations have been found useful in various downstream tasks such as visual question answering (Tapaswi et al., 2016), script inference (Pichotta and Mooney, 2016), and information retrieval (Le and Mikolov, 2014; Palangi et al., 2016). Benchmark datasets (Adi et al., 2017; Conneau and Kiela, 2018; Wang et al., 2018a, 2019) have been proposed to evaluate the encoded knowledge, where the focus has been primarily on natural language understanding capabilities of the representation of a stand-alone sentence, such as its semantic roles, rather than the broader context in which it is situated.

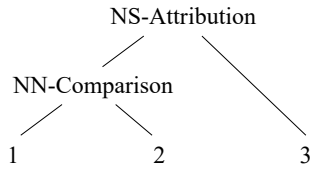


[The European Community's consumer price index rose a provisional 0.6% in September from August]$_1$ [and was up 5.3% from September 1988,]$_2$ [according to Eurostat, the EC's statistical agency.]$_3$

Figure 1: An RST discourse tree from the RST Discourse Treebank. "N" represents "nucleus", containing basic information for the relation. "S" represents "satellite", containing additional information about the nucleus.

In this paper, we seek to incorporate and evaluate discourse knowledge in general purpose sentence representations. A discourse is a coherent, structured group of sentences that acts as a fundamental type of structure in natural language (Jurafsky and Martin, 2009). A discourse structure is often characterized by the arrangement of semantic elements across multiple sentences, such as entities and pronouns. The simplest such arrangement (i.e., linearly-structured) can be understood as sentence ordering, where the structure is manifested in the timing of introducing entities. Deeper discourse structures use more complex relations among sentences (e.g., tree-structured; see Figure 1).

Theoretically, discourse structures have been approached through Centering Theory (Grosz et al., 1995) for studying distributions of entities across text and Rhetorical Structure Theory (RST; Mann and Thompson, 1988) for modelling the logical structure of natural language via discourse trees. Researchers have found modelling discourse useful in a range of tasks (Guzmán et al., 2014; Narasimhan and Barzilay, 2015; Liu and Lapata, 2018; Pan et al., 2018), including summarization (Gerani et al., 2014), text classification (Ji

*Equal contribution. Listed in alphabetical order.
[1]Data processing and evaluation scripts are available at https://github.com/ZeweiChu/DiscoEval.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 649–662, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

and Smith, 2017), and text generation (Bosselut et al., 2018).

In this paper, we propose DiscoEval, a task suite designed to evaluate discourse-related knowledge in pretrained sentence representations. DiscoEval comprises 7 task groups covering multiple domains, including Wikipedia, stories, dialogues, and scientific literature. The tasks are probing tasks (Shi et al., 2016; Adi et al., 2017; Belinkov et al., 2017; Peters et al., 2018b; Conneau et al., 2018; Poliak et al., 2018; Tenney et al., 2019; Liu et al., 2019a; Ettinger, 2019; Chen et al., 2019, *inter alia*) based on sentence ordering, annotated discourse relations, and discourse coherence. The data is either generated semi-automatically or based on human annotations (Carlson et al., 2001; Prasad et al., 2008; Lin et al., 2009; Kummerfeld et al., 2019).

We also propose a set of novel multi-task learning objectives building upon standard pretrained sentence encoders, which rely on the assumption of distributional semantics of text. These objectives depend only on the natural structure in structured document collections like Wikipedia.

Empirically, we benchmark our models and several popular sentence encoders on DiscoEval and SentEval (Conneau and Kiela, 2018). We find that our proposed training objectives help the models capture different characteristics in the sentence representations. Additionally, we find that ELMo shows strong performance on SentEval, whereas BERT performs the best among the pretrained embeddings on DiscoEval. Both BERT and Skip-thought vectors (Kiros et al., 2015), which have training losses explicitly related to surrounding sentences, perform much stronger compared to their respective prior work, demonstrating the effectiveness of incorporating losses that make use of broader context. Through per-layer analysis, we also find that for both BERT and ELMo, deep layers consistently outperform shallower ones on DiscoEval, showing different trends from SentEval where the shallow layers have the best performance.

## 2 Related Work

Discourse modelling and discourse parsing have a rich history (Marcu, 2000; Barzilay and Lapata, 2008; Zhou et al., 2010; Kalchbrenner and Blunsom, 2013; Ji and Eisenstein, 2015; Li and Jurafsky, 2017; Wang et al., 2018c; Liu et al., 2018; Lin et al., 2019, *inter alia*), much of it based on recovering linguistic annotations of discourse structure.

Several researchers have defined tasks related to discourse structure, including sentence ordering (Chen et al., 2016; Logeswaran et al., 2016; Cui et al., 2018), sentence clustering (Wang et al., 2018b), and disentangling textual threads (Elsner and Charniak, 2008, 2010; Lowe et al., 2015; Mehri and Carenini, 2017; Jiang et al., 2018; Kummerfeld et al., 2019).

There is a great deal of prior work on pretrained representations (Le and Mikolov, 2014; Kiros et al., 2015; Hill et al., 2016; Wieting et al., 2016; McCann et al., 2017; Gan et al., 2017; Peters et al., 2018a; Logeswaran and Lee, 2018; Devlin et al., 2019; Tang and de Sa, 2019; Yang et al., 2019; Liu et al., 2019b, *inter alia*). Skip-thought vectors form an effective architecture for general-purpose sentence embeddings. The model encodes a sentence to a vector representation, and then predicts the previous and next sentences in the discourse context. Since Skip-thought performs well in downstream evaluation tasks, we use this neighboring-sentence objective as a starting point for our models.

There is also work on incorporating discourse related objectives into the training of sentence representations. Jernite et al. (2017) propose binary sentence ordering, conjunction prediction (requiring manually-defined conjunction groups), and next sentence prediction. Similarly, Sileo et al. (2019) and Nie et al. (2019) create training datasets automatically based on discourse relations provided in the Penn Discourse Treebank (PDTB; Lin et al., 2009).

Our work differs from prior work in that we propose a general-purpose pretrained sentence embedding evaluation suite that covers multiple aspects of discourse knowledge and we propose novel training signals based on document structure, including sentence position and section titles, without requiring additional human annotation.

## 3 Discourse Evaluation

We propose DiscoEval, a test suite of 7 tasks to evaluate whether sentence representations include semantic information relevant to discourse processing. Below we describe the tasks and datasets, as well as the evaluation framework. We closely follow the SentEval sentence embedding evaluation suite, in particular its supervised sentence and

sentence pair classification tasks, which use predefined neural architectures with slots for fixed-dimensional sentence embeddings. All DiscoEval tasks are modelled by logistic regression unless otherwise stated in later sections.

We also experimented with adding hidden layers to the DiscoEval classification models. However, we find simpler linear classifiers to provide a clearer comparison among sentence embedding methods. More complex classification models lead to noisier results, as more of the modelling burden is shifted to the optimization of the classifiers. Hence we decide to evaluate the sentence embeddings with simple classification models.

In the rest of this section, we will use $[\cdot, \cdot, \cdots]$ to denote concatenation of vectors, $\odot$ for element-wise multiplication, and $|\cdot|$ for element-wise absolute value.

### 3.1 Discourse Relations

As the most direct way to probe discourse knowledge, we consider the task of predicting annotated discourse relations among sentences. We use two human-annotated datasets: the RST Discourse Treebank (RST-DT; Carlson et al., 2001) and the Penn Discourse Treebank (PDTB; Prasad et al., 2008). They have different labeling schemes. PDTB provides discourse markers for adjacent sentences, whereas RST-DT offers document-level discourse trees, which recently was used to evaluate discourse knowledge encoded in document-level models (Ferracane et al., 2019). The difference allows us to see if the pretrained representations capture local or global information about discourse structure.

More specifically, as shown in Figure 1, in RST-DT, text is segmented into basic units, elementary discourse units (EDUs), upon which a discourse tree is built recursively. Although a relation can take multiple units, we follow prior work (Ji and Eisenstein, 2014) to use right-branching trees for non-binary relations to binarize the tree structure and use the 18 coarse-grained relations defined by Carlson et al. (2001).

When evaluating pretrained sentence encoders on RST-DT, we first encode EDUs into vectors, then use averaged vectors of EDUs of subtrees as the representation of the subtrees. The target prediction is the label of nodes in discourse trees and the input to the classifier is $[x_{\text{left}}, x_{\text{right}}, x_{\text{left}} \odot x_{\text{right}}, |x_{\text{left}} - x_{\text{right}}|]$, where $x_{\text{left}}$ and $x_{\text{right}}$ are vector representations of the left and right subtrees respectively. For example, the input for target "NN-Attribution" in Figure 1 would be $x_{\text{left}} = \frac{x_1+x_2}{2}$, $x_{\text{right}} = x_3$, where $x_i$ is the encoded representation for the $i$th EDU in the text. We use the standard data splits, where there are 347 documents for training and 38 documents for testing. We choose 35 documents from the training set to serve as a validation set.

1. In any case, the brokerage firms are clearly moving faster to create new ads than they did in the fall of 1987.
2. [But] it remains to be seen whether their ads will be any more effective.
label: Comparison.Contrast

Figure 2: Example in the PDTB explicit relation task. The words in [] are taken out from input sentence 2.

1. "A lot of investor confidence comes from the fact that they can speak to us," he says.
2. [so] "To maintain that dialogue is absolutely crucial."
label: Contingency.Cause

Figure 3: Example in the PDTB implicit relation task.

For PDTB, we use a pair of sentences to predict discourse relations. Following Lin et al. (2009), we focus on two kinds of relations from PDTB: explicit (PDTB-E) and implicit (PDTB-I). The sentence pairs with explicit relations are two consecutive sentences with a particular connective word in between. Figure 2 is an example of an explicit relation.

In the PDTB, annotators insert an implicit connective between adjacent sentences to reflect their relations, if such an implicit relation exists. Figure 3 shows an example of an implicit relation. The PDTB provides a three-level hierarchy of relation tags. In DiscoEval, we use the second level of types (Lin et al., 2009), as they provide finer semantic distinctions compared to the first level. To ensure there is a reasonable amount of evaluation data, we use sections 2-14 as training set, 15-18 as development set, and 19-23 as test set. In addition, we filter out categories that have less than 10 instances. This leaves us 12 categories for explicit relations and 11 for implicit ones. Category names are listed in the supplementary material.

We use the sentence embeddings to infer sentence relations with supervised training. As input to the classifier, we encode both sentences to vector representations $x_1$ and $x_2$, concatenated with their element-wise product and absolute difference: $[x_1, x_2, x_1 \odot x_2, |x_1 - x_2|]$.

- She was excited thinking she must have lost weight.
- Bonnie hated trying on clothes.
- Then she realized they actually size 14s, and 12s.
- She picked up a pair of size 12 jeans from the display.
- When she tried them on they were too big!

Figure 4: Example from the ROC Stories domain of the Sentence Position task. The first sentence should be in the fourth position.

### 3.2 Sentence Position (SP)

We create a task that we call Sentence Position. It can be seen as way to probe the knowledge of linearly-structured discourse, where the ordering corresponds to the timings of events. When constructing this dataset, we take five consecutive sentences from a corpus, randomly move one of these five sentences to the first position, and ask models to predict the true position of the first sentence in the modified sequence.

We create three versions of this task, one for each of the following three domains: the first five sentences of the introduction section of a Wikipedia article (Wiki), the ROC Stories corpus (ROC; Mostafazadeh et al., 2016), and the first 5 sentences in the abstracts of arXiv papers (arXiv; Chen et al., 2016). Figure 4 shows an example of this task for the ROC Stories domain. The first sentence should be in the fourth position among these sentences. To make correct predictions, the model needs to be aware of both typical orderings of events as well as how events are described in language. In the example shown, Bonnie's excitement comes from her imagination so it must happen after she picked up the jeans and tried them on but right before she realized the actual size.

To train classifiers for these tasks, we do the following. We first encode the five sentences to vector representations $x_i$. As input to the classifier, we include $x_1$ and the concatenation of $x_1 - x_i$ for all $i$: $[x_1, x_1 - x_2, x_1 - x_3, x_1 - x_4, x_1 - x_5]$.

### 3.3 Binary Sentence Ordering (BSO)

Similar to sentence position prediction, Binary Sentence Ordering (BSO) is a binary classification task to determine the order of two sentences. The fact that BSO only has a pair of sentences as input makes it different from Sentence Position, where there is more context, and we hope that BSO can evaluate the ability of capturing local discourse coherence in the given sentence representations. The data comes from the same three domains as Sentence Position, and each instance is a pair of consecutive sentences.

1. These functions include fast and synchronized response to environmental change, or long-term memory about the transcriptional status.
2. Focusing on the collective behaviors on a population level, we explore potential regulatory functions this model can offer.

Figure 5: Example from the arXiv domain of the Binary Sentence Ordering task (incorrect ordering shown).

Figure 5 shows an example from the arXiv domain of the Binary Sentence Ordering task. The order of the sentences in this instance is incorrect, as the "functions" are referenced before they are introduced. To detect the incorrect ordering in this example, the encoded representations need to be able to provide information about new and old information in each sentence.

To form the input when training classifiers, we concatenate the embeddings of both sentences with their element-wise difference: $[x_1, x_2, x_1 - x_2]$.

### 3.4 Discourse Coherence (DC)

Inspired by prior work on chat disentanglement (Elsner and Charniak, 2008, 2010) and sentence clustering (Wang et al., 2018b), we propose a sentence disentanglement task. The task is to determine whether a sequence of six sentences forms a coherent paragraph. We start with a coherent sequence of six sentences, then randomly replace one of the sentences (chosen uniformly among positions 2-5) with a sentence from another discourse. This task, which we call Discourse Coherence (DC), is a binary classification task and the datasets are balanced between positive and negative instances.

We use data from two domains for this task: Wikipedia and the Ubuntu IRC channel.[2] For Wikipedia, we begin by choosing a sequence of six sentences from a Wikipedia article. For purposes of choosing difficult distractor sentences, we use the Wikipedia categories of each document as an indication of its topic. To create a negative instance, we randomly sample a sentence from another document with a similar set of categories (measured by the percentage of overlapping categories). This sampled sentence replaces one of the six consecutive sentences in the original sequence. When splitting the train, development,

[2] irclogs.ubuntu.com/

1. It is possible he was the youngest of the family as the name "Sextus" translates to sixth in English implying he was the sixth of two living and three stillborn brothers.
2. According to Roman tradition, his rape of Lucretia was the precipitating event in the overthrow of the monarchy and the establishment of the Roman Republic.
3. Tarquinius Superbus was besieging Ardea, a city of the Rutulians.
4. The place could not be taken by force, and the Roman army lay encamped beneath the walls.
5. **He was soon elected to the Academy's membership (although he had to wait until 1903 to be elected to the Society of American Artists), and in 1883 he opened a New York studio, dividing his time for several years between Manhattan and Boston.**
6. As nothing was happening in the field, they mounted their horses to pay a surprise visit to their homes.

Figure 6: An example from the Wikipedia domain of the Discourse Coherence task. This sequence is not coherent; the boldface sentence was substituted in for the true fifth sentence from another article.

and test sets, we ensure there are no overlapping documents among them.

Our proposed dataset differs from the sentence clustering task of Wang et al. (2018b) in that it preserves sentence order and does not anonymize or lemmatize words, because they play an important role in conveying information about discourse coherence.

For the Ubuntu domain, we use the human annotations of conversation thread structure from Kummerfeld et al. (2019) to provide us with a coherent sequence of utterances. We filter out sentences by heuristic rules to avoid overly technical and unsolvable cases. The negative sentence is randomly picked from other conversations. Similarly, when splitting the train, development, and test sets, we ensure there are no overlapping conversations among them.

Figure 6 is an instance of the Wikipedia domain of the Discourse Coherence task. This instance is not coherent and the boldfaced text is from a different document. The incoherence can be found either by comparing characteristics of the entity being discussed or by the topic of the sentence group. Solving this task is non-trivial as it may require the ability to perform inference across multiple sentences.

In this task, we encode all sentences to vector representations and concatenate all of them ($[x_1, x_2, x_3, x_4, x_5, x_6]$) as input to the classification model. Note that in this task, we use a hidden layer of 2000 dimensions with sigmoid activation in the classification model, as this is necessary for the classifier to use features based on multiple inputs simultaneously given the simple concatenation as input. We could have developed richer ways to encode the input so that a linear classifier would be feasible (e.g., use the element-wise products of all pairs of sentence embeddings), but we wish to keep the input dimensionality of the classifier small enough that the classifier will be learnable given fixed sentence embeddings and limited training data.

1. The theory behind the SVM and the naive Bayes classifier is explored.
2. This relocation of the active target may be repeated an arbitrary number of times.

Figure 7: Examples from Sentence Section Prediction. The first is from an Abstract while the second is not.

| Task | PDTB-E | PDTB-I | Ubuntu | RST-DT | Others |
|---|---|---|---|---|---|
| Train | 9383 | 8693 | 5816 | 17051 | 10000 |
| Dev. | 3613 | 2972 | 1834 | 2045 | 4000 |
| Test | 3758 | 3024 | 2418 | 2308 | 4000 |

Table 1: Size of datasets in DiscoEval.

### 3.5 Sentence Section Prediction (SSP)

The Sentence Section Prediction (SSP) task is defined as determining the section of a given sentence. The motivation behind this task is that sentences within certain sections typically exhibit similar patterns because of the way people write coherent text. The pattern can be found based on connectives or specificity of a sentence. For example, "Empirically" is usually used in the abstract or introduction sections in scientific writing.

We construct the dataset from PeerRead (Kang et al., 2018), which consists of scientific papers from a variety of fields. The goal is to predict whether or not a sentence belongs to the Abstract section. After eliminating sentences that are too easy for the task (e.g., equations), we randomly sample sentences from the Abstract or from a section in the middle of a paper.[3] Figure 7 shows two sentences from this task, where the first sentence is more general and from an Abstract whereas the second is more specific and is from another section. In this task, the input to the classifier is simply the sentence embedding.

Table 1 shows the number of instances in each DiscoEval task introduced above.

[3]We avoid sentences from the Introduction or Conclusion sections to make the task more solvable.

## 4 Models and Learning Criteria

Having described DiscoEval, we now discuss methods for incorporating discourse information into sentence embedding training. All models in our experiments are composed of a single encoder and multiple decoders. The encoder, parameterized by a bidirectional Gated Recurrent Unit (Bi-GRU; Chung et al., 2014), encodes the sentence, either in training or in evaluation of the downstream tasks, to a fixed-length vector representation (i.e., the average of the hidden states across positions).

The decoders take the aforementioned encoded sentence representation, and predict the targets we define in the sections below. We first introduce Neighboring Sentence Prediction, the loss for our baseline model. We then propose additional training losses to encourage our sentence embeddings to capture other context information.

### 4.1 Neighboring Sentence Prediction (NSP)

Similar to prior work on sentence embeddings (Kiros et al., 2015; Hill et al., 2016), we use an encoded sentence representation to predict its surrounding sentences. In particular, we predict the immediately preceding and succeeding sentences. All of our sentence embedding models use this loss. Formally, the loss is defined as

$$\text{NSP} = -\log p_\theta(s_{t-1}|s_t) - \log p_\phi(s_{t+1}|s_t)$$

where we parameterize $p_\theta$ and $p_\phi$ as separate feedforward neural networks and compute the log-probability of a target sentence using its bag-of-words representation.

### 4.2 Nesting Level (NL)

A table of contents serves as a high level description of an article, outlining its organizational structure. Wikipedia articles, for example, contain rich tables of contents with many levels of hierarchical structure. The "nesting level" of a sentence (i.e., how many levels deep it resides) provides information about its role in the overall discourse. To encode this information into our sentence representations, we introduce a discriminative loss to predict a sentence's nesting level in the table of contents:

$$\text{NL} = -\log p_\theta(l_t|s_t)$$

where $l_t$ represents the nesting level of the sentence $s_t$ and $p_\theta$ is parameterized by a feedforward neural network. Note that sentences within the same paragraph share the same nesting level. In Wikipedia, there are up to 7 nesting levels.

### 4.3 Sentence and Paragraph Position (SPP)

Similar to nesting level, we add a loss based on using the sentence representation to predict its position in the paragraph and in the article. The position of the sentence can be a strong indication of the relations between the topics of the current sentence and the topics in the entire article. For example, the first several sentences often cover the general topics to be discussed more thoroughly in the following sentences. To encourage our sentence embeddings to capture such information, we define a position prediction loss

$$\text{SPP} = -\log p_\theta(sp_t|s_t) - \log p_\phi(pp_t|s_t)$$

where $sp_t$ is the sentence position of $s_t$ within the current paragraph and $pp_t$ is the position of the current paragraph in the whole document.

### 4.4 Section and Document Title (SDT)

Unlike the previous position-based losses, this loss makes use of section and document titles, which gives the model more direct access to the topical information at different positions in the document. The loss is defined as

$$\text{SDT} = -\log p_\theta(st_t|s_t) - \log p_\phi(dt_t|s_t)$$

Where $st_t$ is the section title of sentence $s_t$, $dt_t$ is the document title of sentence $s_t$, and $p_\theta$ and $p_\phi$ are two different bag-of-words decoders.

## 5 Experiments

### 5.1 Setup

We train our models on Wikipedia as it is a knowledge rich textual resource and has consistent structures over all documents. Details on hyperparameters are in the supplementary material. When evaluating on DiscoEval, we encode sentences with pretrained sentence encoders. Following SentEval, we freeze the sentence encoders and only learn the parameters of the downstream classifier. The "Baseline" row in Table 2 are embeddings trained with only the NSP loss. The subsequent rows are trained with extra losses defined in Section 4 in addition to the NSP loss.

| | SentEval | | | | DiscoEval | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | USS | SSS | SC | Probing | SP | BSO | DC | SSP | PDTB-E | PDTB-I | RST-DT | avg. |
| Skip-thought | 41.7 | 81.2 | 78.4 | 70.1 | 44.6 | 63.0 | 54.9 | 77.6 | 39.6 | 38.7 | **59.7** | 53.8 |
| InferSent | **63.4** | **83.3** | 79.7 | 71.8 | 43.6 | 62.9 | 56.3 | 77.2 | 37.1 | 38.0 | 53.2 | 49.8 |
| DisSent | 50.0 | 79.2 | 80.5 | 74.0 | 44.9 | 64.9 | 54.8 | 77.6 | 41.6 | 39.9 | 57.8 | 54.6 |
| ELMo | 60.9 | 77.6 | 80.8 | 74.7 | 46.4 | 66.6 | 59.4 | 78.4 | 41.5 | 41.5 | 58.8 | 56.1 |
| BERT-Base | 30.1 | 66.3 | 81.4 | 73.9 | 49.1 | 68.1 | 58.9 | **80.6** | 44.0 | 42.5 | 58.7 | 57.4 |
| BERT-Large | 43.6 | 70.7 | **83.4** | **75.0** | **49.9** | **69.3** | 60.5 | 80.4 | **44.1** | **43.8** | 58.8 | **58.1** |
| Baseline (NSP) | 57.8 | 77.1 | 77.0 | 70.6 | 44.1 | 63.8 | **61.2** | 78.2 | 36.9 | 38.0 | 57.0 | 54.5 |
| + SDT | 59.0 | 77.3 | 76.8 | 69.7 | 43.9 | 62.9 | 60.0 | 78.0 | 37.0 | 37.7 | 56.2 | 53.7 |
| + SPP | 56.0 | 77.5 | 77.4 | 70.7 | 45.6 | 64.3 | 60.8 | 78.6 | 37.1 | 37.7 | 57.1 | 54.6 |
| + NL | 56.7 | 78.2 | 77.2 | 70.6 | 44.7 | 64.0 | **61.2** | 78.3 | 37.2 | 37.8 | 56.4 | 54.4 |
| + SPP + NL | 55.4 | 76.7 | 77.0 | 70.4 | 45.7 | 64.9 | 60.9 | 79.2 | 37.9 | 39.3 | 56.7 | 55.1 |
| + SDT + NL | 58.5 | 76.9 | 77.2 | 70.2 | 44.4 | 63.2 | 61.0 | 78.4 | 36.6 | 37.9 | 57.4 | 54.2 |
| + SDT + SPP | 58.4 | 77.4 | 76.6 | 70.2 | 44.4 | 63.9 | 60.5 | 78.0 | 37.3 | 36.9 | 56.2 | 54.1 |
| ALL | 58.8 | 76.3 | 77.0 | 70.2 | 43.9 | 63.3 | 60.3 | 78.7 | 36.5 | 38.2 | 55.8 | 54.1 |

Table 2: Results for SentEval and DiscoEval. The highest number in each column is boldfaced. The highest number for our models in each column is underlined. "All" uses all four losses. "avg." is the averaged accuracy for all tasks in DiscoEval.

Additionally, we benchmark several popular pretrained sentence encoders on DiscoEval, including Skip-thought,[4] InferSent (Conneau et al., 2017),[5] DisSent (Nie et al., 2019),[6] ELMo,[7] and BERT.[8] For ELMo, we use the averaged vector of all three layers and time steps as the sentence representations. For BERT, we use the averaged vector at the position of the "[CLS]" token across all layers. We also evaluate per-layer performance for both models in Section 6.

When reporting results for SentEval, we compute the averaged Pearson correlations for Semantic Textual Similarity tasks from 2012 to 2016 (Agirre et al., 2012, 2013, 2014, 2015, 2016). We refer to the average as unsupervised semantic similarity (USS) since those tasks do not require training data. We compute the averaged results for the STS Benchmark (Cer et al., 2017), textual entailment, and semantic relatedness (Marelli et al., 2014) and refer to the average as supervised semantic similarity (SSS). We compute the average accuracy for movie review (Pang and Lee, 2005); customer review (Hu and Liu, 2004); opinion polarity (Wiebe et al., 2005); subjectivity classification (Pang and Lee, 2004); Stanford sentiment treebank (Socher et al., 2013); question classification (Li and Roth, 2002); and paraphrase detection (Dolan et al., 2004), and refer to it as sentence classification (SC). For the rest of the linguistic probing tasks (Conneau et al., 2018), we report the average accuracy and report it as "Probing".

### 5.2 Results

Table 2 shows the experiment results over all SentEval and DiscoEval tasks. Different models and training signals have complex effects when performing various downstream tasks. We summarize our findings below:

- On DiscoEval, Skip-thought performs best on RST-DT. DisSent performs strongly for PDTB tasks but it requires discourse markers from PDTB for generating training data. BERT has the highest average by a large margin, but ELMo has competitive performance on multiple tasks.
- The NL or SPP loss alone has complex effects across tasks in DiscoEval, but when they are combined, the model achieves the best performance, outperforming our baseline by 0.6% on average. In particular, it yields 39.3% accuracy on PDTB-I, outperforming Skip-thought by 0.6%. This is presumably caused by the differing, yet complementary, effects of these two losses (NL and SPP).
- The SDT loss generally hurts performance on DiscoEval, especially on the position-related tasks (SP, BSO). This can be explained by the notion that consecutive sentences in the same section are encouraged to have the same sentence representations when using the SDT loss. However, the SP and BSO tasks involve differentiating neighboring sentences in terms of their position and ordering information.

[4] github.com/ryankiros/skip-thoughts
[5] github.com/facebookresearch/InferSent
[6] github.com/windweller/DisExtract
[7] github.com/allenai/allennlp
[8] github.com/huggingface/pytorch-pretrained-BERT

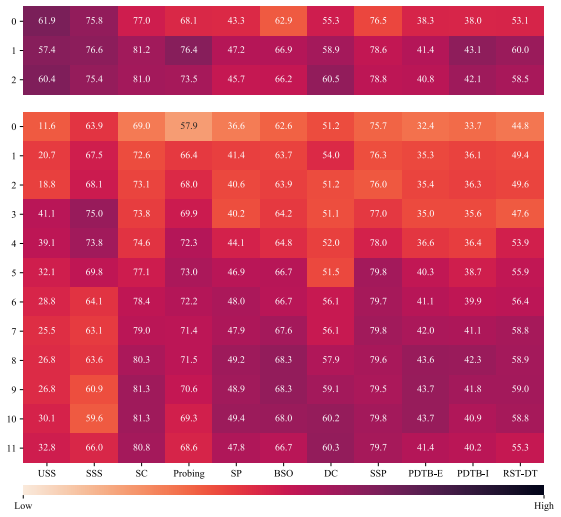


Figure 8: Heatmap for individual hidden layers of BERT-Base (lower part) and ELMo (upper part).

| | ELMo | BERT-Base |
|---|---|---|
| SentEval | 0.8 | 5.0 |
| DiscoEval | 1.3 | 8.9 |

Table 3: Average of the layer number for the best layers in SentEval and DiscoEval.

- On SentEval, SDT is most helpful for the USS tasks, presumably because it provides the most direct information about the topic of each sentence, which is a component of semantic similarity. SDT helps slightly on the SSS tasks. NL gives the biggest improvement in SSS.
- In comparing BERT to ELMo and Skip-thought to InferSent on DiscoEval, we can see the benefit of adding information about neighboring sentences. Our proposed training objectives show complementary improvements over NSP, which suggests that they can potentially benefit these pretrained representations.

## 6 Analysis

**Per-Layer analysis.** To investigate the performance of individual hidden layers, we evaluate ELMo and BERT on both SentEval and DiscoEval using each hidden layer. For ELMo, we use the averaged vector from the targeted layer. For BERT-Base, we use the vector from the position of the "[CLS]" token. Figure 8 shows the heatmap of performance for individual hidden layers. We note that for better visualization, colors in each column are standardized. On SentEval, BERT-Base performs better with shallow layers on USS, SSS, and Probing (though not on SC), but on DiscoEval, the results using BERT-Base gradually increase with deeper layers. To evaluate this phenomenon quantitatively, we compute the average of the layer number for the best layers for both ELMo and BERT-Base and show it in Table 3. From the table, we can see that DiscoEval requires deeper layers to achieve better performance. We assume this is because deeper layers can capture higher-level structure, which aligns with the information needed to solve the discourse tasks.

| | |
|---|---|
| Baseline w/o hidden layer | 52.0 |
| Baseline w/ hidden layer | 61.2 |

Table 4: Accuracies with baseline encoder on Discourse Coherence task, with or without a hidden layer in the classifier.

**DiscoEval architectures.** In all DiscoEval tasks except DC, we use no hidden layer in the neural architectures, following the example of SentEval. However, some tasks are unsolvable with this simple architecture. In particular, the DC tasks have low accuracies with all models unless a hidden layer is used. As shown in Table 4, when adding a hidden layer of 2000 to this task, the performance on DC improves dramatically. This shows that DC requires more complex comparison and inference among input sentences. Our human evaluation below on DC also shows that human accuracies exceed those of the classifier based on sentence embeddings by a large margin.

**Human Evaluation.** We conduct a human evaluation on the Sentence Position, Binary Sentence Ordering, and Discourse Coherence datasets. A native English speaker was provided with 50 examples per domain for these tasks. While the results in Table 5 show that the overall human accuracies exceed those of the classifier based on BERT-Large by a large margin, we observe that within some specific domains, for example Wiki in BSO, BERT-Large demonstrates very strong performance.

**Does context matter in Sentence Position?** In the SP task, the inputs are the target sentence together with 4 surrounding sentences. We study the effect of removing the surrounding 4 sentences, i.e., only using the target sentence to predict its position from the start of the paragraph.

| | Sentence Position | | | Binary Sentence Ordering | | | Discourse Coherence | |
|---|---|---|---|---|---|---|---|---|
| Human | 77.3 | | | 84.7 | | | 87.0 | |
| BERT-Large | 49.9 | | | 69.3 | | | 60.5 | |
| | Wiki | arXiv | ROC | Wiki | arXiv | ROC | Wiki | Ubuntu |
| Human | 84.0 | 76.0 | 94.0 | 64.0 | 72.0 | 96.0 | 98.0 | 74.0 |
| BERT-Large | 43.0 | 56.0 | 50.9 | 70.3 | 66.8 | 70.9 | 64.9 | 56.1 |

Table 5: Accuracies (%) for a human annotator and BERT-Large on Sentence Position, Binary Sentence Ordering, and Discourse Coherence tasks.

| | |
|---|---|
| Random | 20 |
| Baseline w/o context | 43.2 |
| Baseline w/ context | 46.0 |

Table 6: Accuracies (%) for baseline encoder on Sentence Position task when using downstream classifier with or without context.

Table 6 shows the comparison of the baseline model performance on Sentence Position with or without the surrounding sentences and a random baseline. Since our baseline model is already trained with NSP, it is expected to see improvements over a random baseline. The further improvement from using surrounding sentences demonstrates that the context information is helpful in determining the sentence position.

## 7 Conclusion

We proposed DiscoEval, a test suite of tasks to evaluate discourse-related knowledge encoded in pretrained sentence representations. We also proposed a variety of training objectives to strengthen encoders' ability to incorporate discourse information. We benchmarked several pretrained sentence encoders and demonstrated the effects of the proposed training objectives on different tasks. While our learning criteria showed benefit on certain classes of tasks, our hope is that the DiscoEval evaluation suite can inspire additional research in capturing broad discourse context in fixed-dimensional sentence embeddings.

## Acknowledgments

We thank Jonathan Kummerfeld for helpful discussions about the IRC Disentanglement dataset, Davis Yoshida for discussions about BERT, and the anonymous reviewers for their feedback that improved this paper. This research was supported in part by a Bloomberg data science research grant to K. Gimpel.

## References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2017. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. In *Proceedings of ICLR*.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Inigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 2015. Semeval-2015 task 2: Semantic textual similarity, english, spanish and pilot on interpretability. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 252–263. Association for Computational Linguistics.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. Semeval-2014 task 10: Multilingual semantic textual similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 81–91. Association for Computational Linguistics.

Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016. Semeval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 497–511. Association for Computational Linguistics.

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 385–393. Association for Computational Linguistics.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *sem 2013 shared task: Semantic textual similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 32–43. Association for Computational Linguistics.

Regina Barzilay and Mirella Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34.

Yonatan Belinkov, Lluís Màrquez, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2017. Evaluating layers of representation in neural machine translation on part-of-speech and semantic tagging tasks. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Antoine Bosselut, Asli Celikyilmaz, Xiaodong He, Jianfeng Gao, Po-Sen Huang, and Yejin Choi. 2018. Discourse-aware neural rewards for coherent text generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 173–184, New Orleans, Louisiana. Association for Computational Linguistics.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurovsky. 2001. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Proceedings of the Second SIGdial Workshop on Discourse and Dialogue*.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14. Association for Computational Linguistics.

Mingda Chen, Zewei Chu, Karl Stratos, and Kevin Gimpel. 2019. Enteval: A holistic evaluation benchmark for entity representations. In *Proc. of EMNLP*.

Xinchi Chen, Xipeng Qiu, and Xuanjing Huang. 2016. Neural sentence ordering. *arXiv preprint arXiv:1607.06952*.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Alexis Conneau and Douwe Kiela. 2018. SentEval: An evaluation toolkit for universal sentence representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*, Miyazaki, Japan. European Languages Resources Association (ELRA).

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680. Association for Computational Linguistics.

Alexis Conneau, Germán Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics.

Baiyun Cui, Yingming Li, Ming Chen, and Zhongfei Zhang. 2018. Deep attentive sentence ordering network. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4340–4349. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING)*, pages 350–356, Geneva, Switzerland.

Micha Elsner and Eugene Charniak. 2008. You talking to me? a corpus and algorithm for conversation disentanglement. In *Proceedings of ACL-08: HLT*, pages 834–842, Columbus, Ohio. Association for Computational Linguistics.

Micha Elsner and Eugene Charniak. 2010. Disentangling chat. *Computational Linguistics*, 36(3):389–409.

Allyson Ettinger. 2019. What bert is not: Lessons from a new suite of psycholinguistic diagnostics for language models. *arXiv preprint arXiv:1907.13528*.

Elisa Ferracane, Greg Durrett, Junyi Jessy Li, and Katrin Erk. 2019. Evaluating discourse in structured text representations. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 646–653, Florence, Italy. Association for Computational Linguistics.

Zhe Gan, Yunchen Pu, Ricardo Henao, Chunyuan Li, Xiaodong He, and Lawrence Carin. 2017. Learning generic sentence representations using convolutional neural networks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2390–2400, Copenhagen, Denmark. Association for Computational Linguistics.

Shima Gerani, Yashar Mehdad, Giuseppe Carenini, Raymond T. Ng, and Bita Nejat. 2014. Abstractive summarization of product reviews using discourse structure. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1602–1613, Doha, Qatar. Association for Computational Linguistics.

Barbara J. Grosz, Scott Weinstein, and Aravind K. Joshi. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2).

Francisco Guzmán, Shafiq Joty, Lluís Màrquez, and Preslav Nakov. 2014. Using discourse structure improves machine translation evaluation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 687–698, Baltimore, Maryland. Association for Computational Linguistics.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1367–1377. Association for Computational Linguistics.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Yacine Jernite, Samuel R Bowman, and David Sontag. 2017. Discourse-based objectives for fast unsupervised sentence representation learning. *arXiv preprint arXiv:1705.00557*.

Yangfeng Ji and Jacob Eisenstein. 2014. Representation learning for text-level discourse parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13–24, Baltimore, Maryland. Association for Computational Linguistics.

Yangfeng Ji and Jacob Eisenstein. 2015. One vector is not enough: Entity-augmented distributed semantics for discourse relations. *Transactions of the Association for Computational Linguistics*, 3:329–344.

Yangfeng Ji and Noah A. Smith. 2017. Neural discourse structure for text categorization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 996–1005, Vancouver, Canada. Association for Computational Linguistics.

Jyun-Yu Jiang, Francine Chen, Yan-Ying Chen, and Wei Wang. 2018. Learning to disentangle interleaved conversational threads with a siamese hierarchical network and similarity ranking. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1812–1822. Association for Computational Linguistics.

Daniel Jurafsky and James H. Martin. 2009. *Speech and Language Processing (2Nd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent convolutional neural networks for discourse compositionality. In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality*, pages 119–126. Association for Computational Linguistics.

Dongyeop Kang, Waleed Ammar, Bhavana Dalvi, Madeleine van Zuylen, Sebastian Kohlmeier, Eduard Hovy, and Roy Schwartz. 2018. A dataset of peer reviews (peerread): Collection, insights and nlp applications. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1647–1661. Association for Computational Linguistics.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, pages 3294–3302, Cambridge, MA, USA. MIT Press.

Jonathan K. Kummerfeld, Sai R. Gouravajhala, Joseph J. Peper, Vignesh Athreya, Chulaka Gunasekara, Jatin Ganhotra, Siva Sankalp Patel, Lazaros C Polymenakos, and Walter Lasecki. 2019. A large-scale corpus for conversation disentanglement. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3846–3856, Florence, Italy. Association for Computational Linguistics.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196.

Jiwei Li and Dan Jurafsky. 2017. Neural net models of open-domain discourse coherence. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 198–209. Association for Computational Linguistics.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

Xiang Lin, Shafiq Joty, Prathyusha Jwalapuram, and M Saiful Bari. 2019. A unified linear-time framework for sentence-level discourse parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4190–4200, Florence, Italy. Association for Computational Linguistics.

Ziheng Lin, Min-Yen Kan, and Hwee Tou Ng. 2009. Recognizing implicit discourse relations in the penn discourse treebank. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 343–351. Association for Computational Linguistics.

Jiangming Liu, Shay B. Cohen, and Mirella Lapata. 2018. Discourse representation structure parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 429–439, Melbourne, Australia. Association for Computational Linguistics.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019a. Linguistic knowledge and transferability of contextual representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.

Yang Liu and Mirella Lapata. 2018. Learning structured text representations. *Transactions of the Association for Computational Linguistics*, 6:63–75.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Lajanugen Logeswaran and Honglak Lee. 2018. An efficient framework for learning sentence representations. In *International Conference on Learning Representations*.

Lajanugen Logeswaran, Honglak Lee, and Dragomir Radev. 2016. Sentence ordering and coherence modeling using recurrent neural networks.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The Ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 285–294, Prague, Czech Republic. Association for Computational Linguistics.

William C Mann and Sandra A Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8(3):243–281.

Daniel Marcu. 2000. *The Theory and Practice of Discourse Parsing and Summarization*. MIT Press, Cambridge, MA, USA.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014. SemEval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6294–6305. Curran Associates, Inc.

Shikib Mehri and Giuseppe Carenini. 2017. Chat disentanglement: Identifying semantic reply relationships with random forests and recurrent neural networks. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 615–623, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849, San Diego, California. Association for Computational Linguistics.

Karthik Narasimhan and Regina Barzilay. 2015. Machine comprehension with discourse relations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1253–1262, Beijing, China. Association for Computational Linguistics.

Allen Nie, Erin Bennett, and Noah Goodman. 2019. DisSent: Learning sentence representations from explicit discourse relations. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4497–4510, Florence, Italy. Association for Computational Linguistics.

Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. 2016. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 24(4):694–707.

Boyuan Pan, Yazheng Yang, Zhou Zhao, Yueting Zhuang, Deng Cai, and Xiaofei He. 2018. Discourse marker augmented network with reinforcement learning for natural language inference. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 989–999, Melbourne, Australia. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 115–124. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018a. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. 2018b. Dissecting contextual word embeddings: Architecture and representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509, Brussels, Belgium. Association for Computational Linguistics.

Karl Pichotta and Raymond J. Mooney. 2016. Using sentence-level LSTM language models for script inference. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 279–289, Berlin, Germany. Association for Computational Linguistics.

Adam Poliak, Aparajita Haldar, Rachel Rudinger, J. Edward Hu, Ellie Pavlick, Aaron Steven White, and Benjamin Van Durme. 2018. Collecting diverse natural language inference problems for sentence representation evaluation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 67–81, Brussels, Belgium. Association for Computational Linguistics.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The penn discourse treebank 2.0. In *In Proceedings of LREC*.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural MT learn source syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1526–1534, Austin, Texas. Association for Computational Linguistics.

Damien Sileo, Tim Van De Cruys, Camille Pradel, and Philippe Muller. 2019. Mining discourse markers for unsupervised sentence representation learning. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3477–3486, Minneapolis, Minnesota. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642.

Shuai Tang and Virginia R. de Sa. 2019. Exploiting invertible decoders for unsupervised sentence representation learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4050–4060, Florence, Italy. Association for Computational Linguistics.

Makarand Tapaswi, Yukun Zhu, Rainer Stiefelhagen, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2016. Movieqa: Understanding stories in movies through question-answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4631–4640.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Sam Bowman, Dipanjan Das, and Ellie Pavlick. 2019. What do you learn from context? probing for sentence structure in contextualized word representations. In *International Conference on Learning Representations*.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *arXiv preprint arXiv:1905.00537*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018a. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355. Association for Computational Linguistics.

Su Wang, Eric Holgate, Greg Durrett, and Katrin Erk. 2018b. Picking apart story salads. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1455–1465. Association for Computational Linguistics.

Yizhong Wang, Sujian Li, and Jingfeng Yang. 2018c. Toward fast and accurate neural discourse segmentation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 962–967, Brussels, Belgium. Association for Computational Linguistics.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39(2):165–210.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Towards universal paraphrastic sentence embeddings. In *Proceedings of International Conference on Learning Representations*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Zhi-Min Zhou, Yu Xu, Zheng-Yu Niu, Man Lan, Jian Su, and Chew Lim Tan. 2010. Predicting discourse connectives for implicit discourse relation recognition. In *Coling 2010: Posters*, pages 1507–1514. Coling 2010 Organizing Committee.

# Extracting Possessions from Social Media: Images Complement Language

**Dhivya Chinnappa**, **Srikala Murugan** and **Eduardo Blanco**
Human Intelligence and Language Technologies Lab
University of North Texas
Denton, Texas 76203
{dhivyainfantchinnappa,srikalamurugan}@my.unt.edu, eduardo.blanco@unt.edu

## Abstract

This paper describes a new dataset and experiments to determine whether authors of tweets possess the objects they tweet about. We work with 5,000 tweets and show that both humans and neural networks benefit from images in addition to text. We also introduce a simple yet effective strategy to incorporate visual information into any neural network beyond weights from pretrained networks. Specifically, we consider the tags identified in an image as an additional textual input, and leverage pretrained word embeddings as usually done with regular text. Experimental results show this novel strategy is beneficial.

## 1 Introduction

Social media are platforms for sharing information online. Social media posts and online behavior in general (e.g., Facebook likes, following other users in Twitter) have been shown to predict human traits (Burke et al., 2010; Schwartz et al., 2013). Many social media posts include an image alongside text, and the percentage keeps growing as doing so boosts user engagement (Patel, 2016).

Pictures and text in social media usually complement each other. Thus, even if the information of interest can be understood from either communication modality, considering both is beneficial. Consider the tweet in Figure 1. The text indicates that *Arnold* (the author of the tweet) goes on *bike* rides when he travels. The image shows him riding a bike, indicating that he was riding a *bike* when he tweeted thus he was in possession of a *bike*. On the other hand, if the picture were a screenshot of his Twitter posting statistics, *Arnold* most likely would not be in possession of a *bike* when tweeting, but rather sharing a log of his previous trips with his followers. In this paper, we extract possession relations from social media




Figure 1: Sample tweet with text and an image. We can infer that *Arnold* possessed a *bike* when he tweeted.

posts containing both text and images. Possession is an asymmetric semantic relation between two entities, where one entity (the possessee) *belongs* to the other entity (the possessor) (Stassen, 2009). Following the literature, we consider not only ownership, but also control possessions. In control possessions, the possessor has temporary control of the possessee but not necessarily ownership (Tham, 2004), e.g., *Bill borrowed the ozone generator from John*.

While we do not explore any, extracting possessions has many potential applications. For example, possessions could help to reveal hobbies and to find people with similar interests. Possessions could also improve recommender systems. For example, people without cars are unlikely to be interested in oil changes and auto mechanics. Similarly, people who recently purchased a home may be interested in moving and remodeling services. Extracting possessions could also be useful to identify skills. For example, peo-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 663–672,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

ple who possess a bike are likely to be able to ride bikes, and those who have control possession of an 18-wheeler are typically able to drive large trucks. The main contributions of this paper are: (a) a corpus of 5,000 tweets (text and images) annotated with possession relations including type (alienable or control), temporal anchors with respect to the tweet timestamp, and interest in something,[1] (b) detailed corpus analysis showing, among others, that humans understand more possession relations when they have access to both the text and images, (c) experimental results showing that the task can be automated and features extracted from the images improve results. Regarding visual features, we show that incorporating weights from pretrained networks—a common practice in previous work—is beneficial, but we obtain more substantial improvements incorporating the objects and events identified in an image as an additional textual input and leveraging word embeddings.

## 2 Previous Work

**Possession relations** have primarily been studied in efforts targeting large relation repositories between arguments connected with some lexico-syntactic pattern. Tratz and Hovy (2013) work with 17 semantic relations realized by possessive constructions, Badulescu and Moldovan (2009) with 36 relations realized by genitives, and Nakov and Hearst (2013) and Tratz and Hovy (2010) target noun compounds. Blodgett and Schneider (2018) annotate 50 supersenses (including roles and relations between entities) for possessives. These efforts extract possessions from text, and target possessors and possessees connected by specific patterns. Unlike them, we extract possessions using both text and images. In addition to possession existence, we also extract types, temporal anchors, and interest in the possessee.

Two recent efforts target possession relation extraction from text without strict syntactic constraints. In our previous work, we extract intra-sentential possessions from OntoNotes (Chinnappa and Blanco, 2018). In the work described here, we use the list of synsets from our previous work to select possessees (Section 3). Banea and Mihalcea (2018) work with blogs and annotate possession existence at the time of utterance. Unlike these previous works, we (a) leverage both text and images, (b) work with informal tweets (instead of standard English), (c) temporally anchor possessions before, during and after the tweet timestamp, and (d) also extract whether somebody has an interest in a concrete object regardless of possession existence.

**Using multiple modalities** (e.g., text and images) to better solve some task is not new. Among many others, Specia et al. (2016) propose multimodal machine translation and Moon et al. (2018) show that named entity recognition benefits from taking into account both text and images. Our innovation is twofold. First, we show that humans understand more possession information when they have access to the image accompanying a text, as opposed to only reporting improvements on (automatically) solving some task. Second, our neural image component includes two subcomponents. The first one—weights from InceptionNet—is common in previous work, but the second one is novel. Specifically, the second component considers the objects and events identified in an image as an additional textual input. This allows us to leverage pretrained word embeddings and recurrent neural networks, a strategy that we prove beneficial.

## 3 A Corpus of Possession Relations

We start with a collection of English tweets consisting of text and images (Hu et al., 2018). First, we discard tweets that do not contain *I*, *me*, *my*, or *mine* in order to maximize the amount of tweets published by individuals and avoid tweets by organizations as well as advertisements. Second, we select as potential possessors the authors of tweets, and as potential possesses the nouns subsumed by the WordNet synsets (Miller, 1995) proposed in previous work (Section 2) except the following nouns: *fan*, *filter*, *launch*, *mini*, *release* and *safe*. We eliminate them because they almost never yield possession relations in social media. For example, *fan* almost always refers to a person (e.g., *This Bucks fan put on a show*) instead of to an "apparatus with rotating blades," and *filter* almost always refers to a photo effect (e.g., *Bare face plus a snap filter*) instead of to a "porous device for removing impurities." Finally, we randomly select 5,000 possessor-possessee pairs.

### 3.1 Annotation Tasks And Guidelines

In addition to possession existence (i.e., whether the potential possessor possesses the potential pos-

[1] Available at dhivyachinnappa.com

| | Only Text | | Text and Image | |
|---|---|---|---|---|
| | % | $\kappa$ | % | $\kappa$ |
| alienable, control, never, unk | 87.1 | 0.83 | 86.5 | 0.82 |
| before yes, before no | 90.0 | 0.80 | 91.6 | 0.81 |
| during yes, during no | 89.5 | 0.78 | 98.1 | 0.78 |
| after yes, after no | 90.5 | 0.80 | 97.2 | 0.79 |
| interest_yes, interest_no | 82.5 | 0.76 | 90.3 | 0.78 |

Table 1: Inter-annotator agreements (observed and Cohen's $\kappa$) having access to (a) the text and (b) the text and image. $\kappa$ values between 0.60 and 0.80 are considered *substantial*, and above 0.80 *nearly perfect*.

sessee), we also annotate possession type (alienable or control), temporal anchors with respect to the tweet timestamp (before, during and after), and whether the potential possessor has an interest in the potential possessee regardless of possession existence and possession type.

**Possession Existence.** The first annotation task is to determine whether the potential possessor ($x$) possesses the potential possessee ($y$). Annotators choose between the following labels:

- `yes` if a possession exists (i.e., $x$ possesses $y$) at some point of time;
- `never` if a possession does not exist (i.e., $x$ does not possess $y$) at any point of time; or
- `unk` (unknown) if it is sound to ask whether $x$ possesses $y$, but there is not enough information to choose `yes` or `never`.

**Possession Type.** If a possession relation exists (existence: `yes`), annotators also indicate the type:

- `alienable`: if $x$ can be separated from $y$ and $x$ is the owner of $y$, regardless of spatial proximity or other variables; or
- `control`: if $x$ can be separated from $y$ and $x$ has control over $y$, regardless of ownership, spatial proximity or other variables.

Note that according to these definitions, control possessions, unlike alienable possessions, do not require ownership. For example, people driving a rental car have control possession of the car but not alienable possession. Control possession and alienable possession are mutually exclusive labels. We study possession types (`alienable` and `control`) to understand the strength of the possession relation between the possessor and the possessee. We do not consider inalienable possessions because they are uncommon in social media.

**Temporal Anchors.** If a possession relation exists (existence: `yes`), annotators also indicate when it is true with respect to the tweet timestamp:

- `before yes` or `no`: whether $x$ possesses $y$ the day before tweeting or earlier;
- `during yes` or `no`: whether $x$ possesses $y$ the day he tweeted; and
- `after yes` or `no`: whether $x$ possesses $y$ the day after tweeting or later.

**Interest in the Possessee.** Finally, annotators also indicate whether $x$ has an interest in $y$ (`interest_yes` or `interest_no`), regardless of the labels for possession existence and type. Interest does not entail past, current or future possession existence. It indicates that $x$ shows curiosity or excitement about $y$. Let us consider John Doe and a tweet about eating more vegetables (with a fork) because of a doctor's recommendation. In this context, John would have possession of the *fork* but no interest in it.

### 3.2 Annotation Set-Up and Quality

Annotations were done in-house by two graduate students. We developed a simple interface that showed one tweet at a time, and annotators were instructed to use world knowledge and common sense. In order to minimize biases, the interface did not show the twitter handle, profile picture or any other user information. In addition, we sampled 200 tweets and only in 7% of them the image possibly provided information about the tweet author (e.g., gender, age, race, ethnicity).

In a first round of annotations, annotators had access only to the text in the tweet. In a second round, they had access to both the text and the image. Our rationale is that we are interested in comparing human judgments depending on whether the image is available or not (Section 4.1).

**Inter-Annotator Agreement.** Table 1 shows inter-annotator agreements (observed and Cohen's $\kappa$) when annotators have access to (a) only the text and (b) the text and image. Agreements are very similar regardless of whether the image is available, and as we shall see (Section 4.1), doing so results in more possession information. Cohen's

| | | |
|---|---|---|
| Possession existence and type | yes, alienable | 38.4% |
| | yes, control | 7.5% |
| | never | 38.6% |
| | unk | 15.5% |
| Interest | interest_yes | 40.2% |
| | interest_no | 59.8% |

Table 2: Label distributions. We divide the percentage of yes into alienable and control.

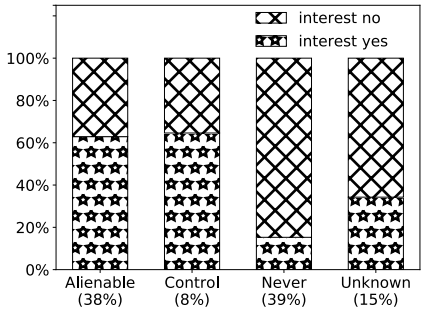


Figure 2: Distribution of *interest in the possessee* labels depending on possession existence and type. People often have an interest in objects they do not possess and when possessions cannot be determined (never, unk).

$\kappa$ for possession existence and type (first row) is 0.82 with text and images, it ranges from 0.78 to 0.81 for temporal anchors (rows 2–4), and it is 0.78 for interest in the possessee (row 5). $\kappa$ values between 0.60 and 0.80 are *substantial*, and above 0.80 *nearly perfect* (Artstein and Poesio, 2008).

## 4 Corpus Analysis

We start describing the final corpus, which was annotated with both text and images. Then, we compare the corpus obtained when annotators have access to (a) only the text and (b) the text and image.

**Label Distributions.** Table 2 shows the label distributions for the three annotation tasks: possession existence, possession type (yes: alienable or control) and interest in the possessee.

Overall, the percentage of unknown label (unk) is low (15.5%), indicating that possession existence can almost always be determined. More importantly, the procedure described in Section 3 allows us to reveal useful knowledge in 84.5% of all generated possessor-possessee pairs (alienable, control and never). Most possessions are alienable (83.6%, 38.4% of all possessor-possesee pairs) and the percentage of control possession is low. The never label is somewhat common (38.6), indicating that people often tweet about objects that they have never possessed.

| | | alienable | control |
|---|---|---|---|
| before | yes | 79.5% | 45.8% |
| | never | 20.5% | 54.2% |
| during | yes | 95.9% | 76.3% |
| | no | 4.1% | 23.7% |
| after | yes | 90.5% | 29.5% |
| | no | 9.5% | 70.5% |

Table 3: Distribution of temporal anchor labels depending on the possession type (alienable or control).

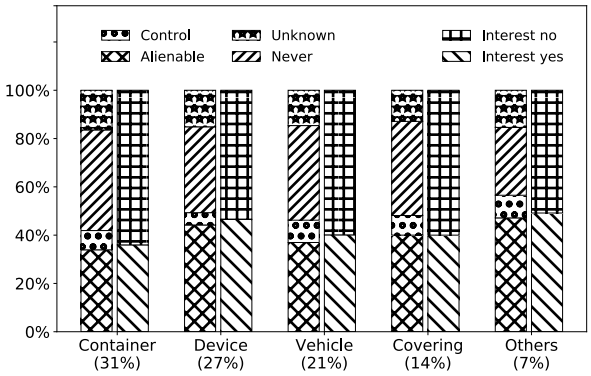


Figure 3: Label distribution (left column: possession existence and type, right column: interest) depending on the WordNet synset of the possessee.

Regarding interest, the possessor has an interest in the possessee in 40.2% of the 5,000 generated pairs. Figure 2 provides insights regarding possession existence and interest for the generated pairs. First, the possessor has an interest in the possessee in (a) 35% of pairs for which possession could not be determined (unk) and (b) 15% of pairs for which no possession exists (never). Second, regardless of possession type, the percentage of interest_yes remains at approximately 60%.

The distributions of temporal anchor labels (before, during and after) per possession type (Table 3) show that possession type substantially influences *when* the possession is true with respect to the tweet timestamp. Regarding alienable possessions, people tweet more about what they own or will own in the future than what they owned in the past (95.9% and 90.5% vs. 79.5%). Control possessions show mostly uniform distribution with anchor before, and are unlikely to be true the day after tweeting (29.5%).

Finally, Figure 3 presents the distribution of possession existence, possession type and interest in the possessee depending on the WordNet synset of the possessee. Regarding possession existence and possession type labels (left columns), we observe similar distributions across synsets, although

<table>
<tr><td>a)<br>  <br></td><td>b)<br>  <br></td><td>c)<br>  <br></td></tr>
<tr><td colspan="3">Possessee: bike</td><td colspan="3">Possessee: jacket</td><td colspan="3">Possessee: bag</td></tr>
<tr><td>Ex., Type</td><td>Temporal</td><td>Interest</td><td>Ex., Type</td><td>Temporal</td><td>Interest</td><td>Ex., Type</td><td>Temporal</td><td>Interest</td></tr>
<tr><td>alienable</td><td>B</td><td>yes</td><td>control</td><td>D</td><td>no</td><td>alienable</td><td>B, D, A</td><td>yes</td></tr>
<tr><td>d)<br>  <br></td><td>e)<br>  <br></td><td>f)<br>  <br></td></tr>
<tr><td colspan="3">Possessee: sunglasses</td><td colspan="3">Possessee: boot</td><td colspan="3">Possessee: pants</td></tr>
<tr><td>Ex., Type</td><td>Temporal</td><td>Interest</td><td>Ex., Type</td><td>Temporal</td><td>Interest</td><td>Ex., Type</td><td>Temporal</td><td>Interest</td></tr>
<tr><td>never</td><td>n/a</td><td>no</td><td>unk</td><td>n/a</td><td>yes</td><td>never</td><td>n/a</td><td>no</td></tr>
</table>

Table 4: Annotation examples when annotators have access to the text and image. We indicate possession existence and type (alienable, control, never, unk), temporal anchors (B: before, D: during, A: after) and interest.

*devices* (e.g., watch, guitar, cell phone) yield more possessions (`alienable` and `control` labels) and most of the possessions are `alienable`. In other words, people tend to tweet about *devices* they own, and rarely about *devices* they only have control over. Regarding interest in the possessee (right columns), people are slightly more likely to have an interest if the possessee is a *device*.

**Annotation Examples.** We present annotation examples in Table 4. Note that unlike in these examples, the annotation interface did not show the Twitter handle and profile picture in an effort to minimize potential biases (Section 3.2).

In Example (a), annotators understood that the author of the tweet was a competitive biker (bike stunt, racing flag), and world knowledge tells us that competitive bikers own their bikes. Thus, the author had an alienable possession with the bike. The text clearly indicates that the possession was true in the past (*2 years ago*), and hints that the author has an interest in bikes (*miss being on X*).

Example (b) is a straightforward example of control possession: the author does not own the *jacket*. While weekends last for two days, it is unknown when the author tweeted, so annotators chose only *during* temporal anchor. Additionally, neither the text or image indicate that the author has any interest in the *jacket*.

Example (c) illustrates an alienable possession in which the author possesses the possessee (i.e., the *bag*) before, during and after tweeting. While there is no specific cue indicating that the author will own the *bag* for an extended period of time, common sense indicates so. Additionally, the text (*my cutest bag ever*) indicates that the author is excited and has an interest in the *bag*.

Example (d) illustrates `never` label. In this case, the author is talking about the baby's *sun-*

| | | Only Text | | | |
|---|---|---|---|---|---|
| | | alienable | control | never | unk |
| Text and Image | alienable | 80.5% | 7.8% | 10.0% | 38.7% |
| | control | 9.4% | 82.8% | 3.0% | 12.2% |
| | never | 8.3% | 6.3% | 83.3% | 23.4% |
| | unk | 1.8% | 3.1% | 4.7% | 25.7% |
| Total | | 100.0% | 100.0% | 100.0% | 100.0% |

Table 5: Changes in labels depending on whether annotators have access to the image. Note that 74.3% of instances annotated unk with only text become alienable (38.7%), control (12.2%) or never (23.4%).

| | (a) | | | (b) | | | (c) | | |
|---|---|---|---|---|---|---|---|---|---|
| | 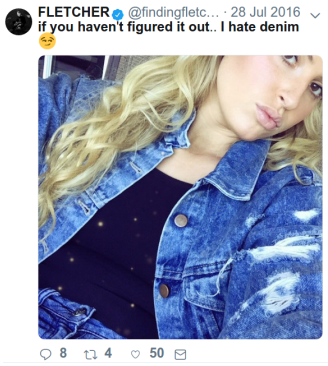  | | | 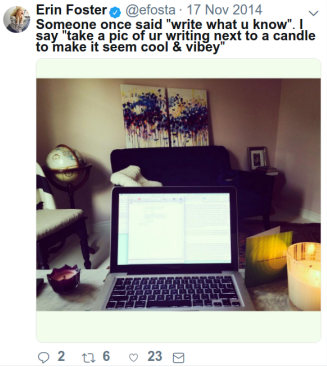  | | |   | | |
| | Possessee: *denim* | | | Possessee: *candle* | | | Possessee: *hat* | | |
| | Ex., Type | Temporal | Interest | Ex., Type | Temporal | Interest | Ex., Type | Temporal | Interest |
| T | unk | n/a | no | unk | n/a | no | alienable | B, D, A | no |
| T+I | alienable | B, D, A | yes | alienable | B, D, A | no | never | n/a | no |

Table 6: Examples of tweets which are annotated different depending on whether annotators have access to only the text (first row of labels, T) or the text and image (second row of labels, T+I). Note that the image allows to annotate more possessions (Examples (a) and (b)) as well as fix mistakes (Example (c)).

*glasses*. Additionally, there is no indication of the author having an interest about the *sunglasses*.

Examples (e, f) illustrate unk and never labels. The author of tweet (d) is sharing his favorite *boots*, and there is not enough information to determine whether she owns any. The author is, however, interested in *boots*, as she went through the task of choosing her favorite *boot* picks. Finally, in Example (f), *pants* cannot be a possessee as it is part of the name of a movie character.

### 4.1 Text vs. Text and Image

Annotators chose different labels depending on whether they had access to (a) only the text or (b) the text and image. Table 5 summarizes the changes in annotations. Most labels remain the same (alienable: 80.5%, control: 82.8%, never: 83.3%), however, most instances labeled unk when annotators have access only to text (74.3%) become alienable (38.7%), control (12.2%), or never (23.4%) when they also have access to the image (last column).

Table 6 shows examples of changes in annotation. Given only the text in Tweet (a), it appears that the author dislikes *denim*. Looking at the image, however, it becomes clear that the author is being sarcastic and not only owns *denim* clothes but actually has a strong interest in *denim*. The text in Tweet (b) is basically two quotes, and looking only at the text one cannot determine whether the author owns any candles. Taking into account the picture, however, one can conclude that the author does own a *candle* (the picture illustrates the advice from the quote) although she does not have an interest in it. The last example, Tweet (c), illustrates how images also help discarding possessions that appear obvious from the text. The *hat* is not an actual object (it is a drawing on top of the picture) thus no alienable possession exists.

## 5 Experiments and Results

We experiment primarily with neural networks. Regarding libraries, we use Keras (Chollet et al., 2015) with TensorFlow as a backend (Abadi et al.,

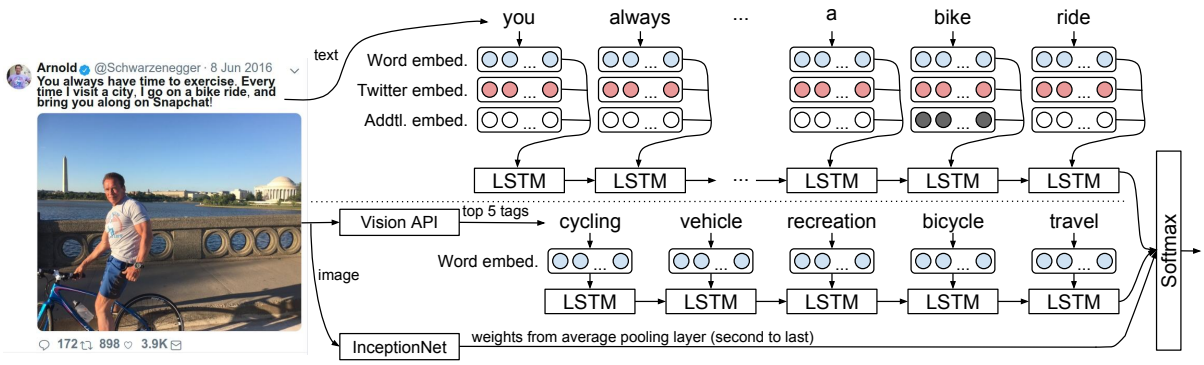


Figure 4: Neural network architecture to predict possession existence, type, temporal anchors and interest. We include a text component (above dotted line) and two image components (below dotted line). Note that the top 5 tags from the Vision API become a textual input, and we use pretrained word embeddings and an LSTM for them.

2015). Each possessor-possessee pair (and corresponding tweet) becomes an instance, and we create stratified training (80%) and test (20%) sets. We train the neural network for up to 200 epochs using the Adam optimizer (Kingma and Ba, 2014), categorical cross entropy, and batch size 32. We stop the training process before 200 epochs if no improvement occurs in the validation set (15% of the training set) for 10 epochs. More specifically, we train six classifiers. The first classifier predicts possession existence (`yes`, `never` or `unk`). The second classifier predicts possession types, i.e., classifies pairs between which a possession exists (`yes`) into `alienable` or `control`. The third, fourth and fifth classifiers predict temporal anchors, i.e., classify pairs between which a possession holds—either alienable or control—into `before yes` or `before no`, `during yes` or `during no`, and `after yes` or `after no`. Finally, the sixth classifier predicts interest in the possessee (`interest_yes` or `interest_no`).[2]

### 5.1 Neural Network Architecture

Figure 4 shows the neural network architecture, which includes components for the text and image (above and below dotted line respectively).

**Text Component.** The text component is an LSTM (Hochreiter and Schmidhuber, 1997). Each token is represented with the concatenation of three embeddings. The first two are Glove word embeddings pretrained with Common Crawl and Twitter (Pennington et al., 2014). The third embedding only takes two possible values (dark grey: possessee, white: non-possessee) and it is used to indicate the potential possessee. Only the additional embeddings are tuned along with other network parameters. Intuitively, the additional embedding allows the LSTM to focus on the context surrounding the potential possessee.

**Image Component.** The image component leverages two pretrained neural networks: InceptionNet (Szegedy et al., 2015) and Cloud Vision API.[3] Generally speaking, InceptionNet is pretrained to identify objects, and the Vision API outputs tags describing images including not only objects but also events (e.g., *cycling*, *recreation*, *travel* from the image in the tweet in Figure 4). Regarding InceptionNet, we follow previous work (Section 2) and include the weights of the average pooling layer (second to last layer). This incorporates features (real numbers) capturing characteristics of the image to the output Softmax layer, where the features are useful for object prediction.

More interestingly, we also incorporate the top 5 tags identified in the image by the Cloud Vision API. The main novelty of our architecture is the strategy to incorporate them. Rather than using one-hot encodings or training special-purpose embeddings, we consider the top 5 tags as an additional textual input and leverage pretrained GloVe word embeddings and an LSTM. Word embeddings allow us to bring meaning to the tags. Intuitively, this is more beneficial than incorporating weights from InceptionNet because the embeddings are a distributed representation of word meaning, and they are useful to, among others, determining word similarity and solving analogies (Pennington et al., 2014). The LSTM is useful because tags may be more than one token (e.g., *whipped cream*, *electronic device*) thus the top 5

[2]Code available at `dhivyachinnappa.com`

[3]`https://cloud.google.com/vision/`

| | Maj. baseline | | | NN only text | | | NN text + IN | | | NN text + Itags | | | NN text + img | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | P | R | F1 | P | R | F1 | P | R | F1 | P | R | F1 | P | R | F1 |
| yes | .46 | 1.0 | .63 | .70 | .69 | .69 | .70 | .72 | .71 | .70 | .78 | .74 | .73 | .78 | .76 |
| never | .00 | .00 | .00 | .57 | .55 | .56 | .57 | .57 | .57 | .62 | .56 | .59 | .64 | .63 | .63 |
| unk | .00 | .00 | .00 | .57 | .55 | .56 | .51 | .47 | .49 | .64 | .54 | .59 | .64 | .53 | .58 |
| Macro Avg. | .15 | .33 | .21 | .61 | .60 | .60 | .59 | .59 | .59 | .65 | .63 | .64 | .67 | .65 | **.66** |
| alienable | .84 | 1.0 | .91 | .84 | .94 | .89 | .84 | .95 | .89 | .84 | .96 | .90 | .83 | .92 | .88 |
| control | .00 | .00 | .00 | .19 | .07 | .10 | .21 | .07 | .10 | .29 | .09 | .14 | .75 | .82 | .77 |
| Macro Avg. | .52 | .50 | .46 | .52 | .51 | .50 | .53 | .51 | .50 | .57 | .53 | .52 | .79 | .87 | **.83** |
| interest_yes | .59 | 1.0 | .74 | .54 | .39 | .45 | .52 | .48 | .50 | .52 | .43 | .47 | .52 | .43 | .47 |
| interest_no | .00 | .00 | .00 | .64 | .77 | .70 | .65 | .68 | .67 | .64 | .72 | .68 | .64 | .72 | .68 |
| Macro Avg. | .30 | .50 | .37 | .61 | .58 | **.58** | .59 | .58 | **.59** | .58 | .58 | **.58** | .58 | .58 | **.58** |

Table 7: Results for predicting possession existence, possession type and interest in possessee. We report results with the full network (last column, NN text + img) as well as with selected components: only text, text + weights from InceptionNet (IN), and text + LSTM encoding of top-5 tags detected in the image (text + Itags).

| | | Before | | | During | | | After | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | P | R | F1 | P | R | F1 | P | R | F1 |
| NN, only text | yes | 0.74 | 0.96 | 0.83 | 0.92 | 0.98 | 0.95 | 0.82 | 0.88 | 0.85 |
| | no | 0.35 | 0.07 | 0.11 | 0.00 | 0.00 | 0.00 | 0.29 | 0.21 | 0.24 |
| | Macro Avg. | 0.55 | 0.52 | 0.47 | 0.46 | 0.49 | 0.48 | 0.56 | 0.55 | 0.55 |
| NN, text + img | yes | 0.70 | 0.78 | 0.74 | 0.88 | 0.97 | 0.92 | 0.84 | 0.89 | 0.87 |
| | no | 0.48 | 0.38 | 0.43 | 0.25 | 0.08 | 0.12 | 0.53 | 0.41 | 0.46 |
| | Macro Avg. | 0.59 | 0.58 | **0.59** | 0.57 | 0.53 | **0.52** | 0.69 | 0.65 | **0.67** |

Table 8: Results for predicting temporal anchors with the neural network (only text, and text and image).

tags are a sequence of at least five tokens—not necessarily five tokens. To the best of our knowledge, this strategy to incorporate information extracted from an image is novel. While simple, we show that it is effective at determining possession existence despite our dataset is relatively small.

### 5.2 Experimental Results

Tables 7 and 8 present the experimental results (Precision, Recall and F1-score). We present results per label and the macro average.

**Baselines.** We use the majority baseline and logistic regression using bag-of-words features (not shown, F1-scores are 0.48 (existence), 0.56 (types) and 0.58 (interest)). The full LSTM (NN text + img) outperforms the baselines predicting possession existence and types (existence F1: 0.66 vs. 0.21–0.64; types F1: 0.83 vs. 0.46–0.52)), but all models except the majority baseline perform roughly the same predicting interest in the possessee (F1: 0.58–0.59).

#### 5.2.1 Neural Network

Table 7 presents the results obtained with four versions of the neural network: using (a) only the text component, (b) the text component and the weights from InceptionNet (text + IN), (c) the text component and the tags from the Vision API as an additional textual input (text + Itags), and (d) the full network (text + img). We also obtained results with only the image components, but do not report the results because they were much worse.

**Possession Existence.** All variations of the neural network outperform the baselines (logistic regression obtains 0.48 F1, not shown). Weights from InceptionNet do not bring any improvement by themselves, but the tags from the Vision API used as an additional textual input do (F1: 0.64 vs. 0.60). More importantly, combining both of them yields 10% improvement (F1: 0.66 vs. 0.60). Further examination revealed that this is due to leveraging pretrained word embeddings with the tags from the Vision API—using one-hot encodings does not bring improvements (not shown).

**Possession Type and Interest in the Possessee.** Regarding possession type, we observe a similar trend than with possession existence. This time, however, the differences in results are larger (F1: 0.50 vs. 0.83) and the network with both image components (NN text + img) is the only model predicting control reliably (0.77 vs. 0.10–0.14).

Regarding interest in the possessee, all models but the majority baseline (including logistic re-

gression) obtain similar F1s (0.58–0.59). While there is certainly room for improvement, the current results lead to the conclusion that a few keywords are sufficient to obtain 0.58 F1: neither images nor word embeddings bring improvements.

**Temporal Anchors.** Table 8 presents results obtained with the neural network when predicting temporal anchors. The image components are beneficial with all anchors, especially `before` (F1: 0.47 vs. 0.59, +25%) and `after` (0.55 vs. 0.67, +22%), and to a lesser degree `during` (0.48 vs. 0.52; 8%). F1 scores are higher for `yes` label than `no` label across all temporal anchors.

## 6 Conclusions

We have presented a corpus of 5,000 tweets and experimental results to extract possession relations. Specifically, we work with text and images in order to reveal the possesees of the author of a tweet. Beyond possession existence, we also consider possession type, temporal anchors with respect to the tweet timestamp, and whether the author has an interest in the potential possessor regardless of possession existence.

The corpus analysis shows that humans understand more possessions when they have access to both the text and images. Authors of tweets often have an interest in potential possessees when there is no possession relation or there is not enough information to determine whether a possession exists (`never` and `unk` labels). Finally, experimental results show that incorporating pretrained networks for object identification and image understanding complement neural components that consider text. Crucially, we show that considering the top 5 tags identified in images (objects and events) as an additional textual input and leveraging word embeddings and recurrent neural networks yields better results than incorporating only weights from intermediate layers, as previous work does.

## References


Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Ron Artstein and Massimo Poesio. 2008. Inter-coder agreement for computational linguistics. *Comput. Linguist.*, 34(4):555–596.

Adriana Badulescu and Dan Moldovan. 2009. A semantic scattering model for the automatic interpretation of english genitives. *NLE*.

Carmen Banea and Rada Mihalcea. 2018. Possession identification in text. *Natural Language Engineering*, 24(4):589610.

Austin Blodgett and Nathan Schneider. 2018. Semantic Supersenses for English Possessives. In *LREC*.

Moira Burke, Cameron Marlow, and Thomas Lento. 2010. Social network activity and social well-being. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 1909–1912, New York, NY, USA. ACM.

Dhivya Chinnappa and Eduardo Blanco. 2018. Mining possessions: Existence, type and temporal anchors. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 496–505, New Orleans, Louisiana, USA. Association for Computational Linguistics.

François Chollet et al. 2015. Keras. `https://github.com/fchollet/keras`.

Ben Eisner, Tim Rocktäschel, Isabelle Augenstein, Matko Bošnjak, and Sebastian Riedel. 2016. emoji2vec: Learning emoji representations from their description. In *Proceedings of The Fourth International Workshop on Natural Language Processing for Social Media*, pages 48–54, Austin, TX, USA. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Yuting Hu, Liang Zheng, Yi Yang, and Yongfeng Huang. 2018. Twitter100k: A real-world dataset for weakly supervised cross-media retrieval. *IEEE Trans. Multimedia*, 20(4):927–938.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

George A. Miller. 1995. Wordnet: A lexical database for english. In *Communications of the ACM*, volume 38, pages 39–41.

Seungwhan Moon, Leonardo Neves, and Vitor Carvalho. 2018. Multimodal named entity recognition for short social media posts. In *Proceedings of the*

*2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 852–860. Association for Computational Linguistics.

Preslav I. Nakov and Marti A. Hearst. 2013. Semantic interpretation of noun compounds using verbal and other paraphrases. *ACM Trans. Speech Lang. Process.*, 10(3):13:1–13:51.

Niel Patel. 2016. 3 Simple Tools To Double Your Twitter Engagement This Week. https://www.forbes.com/sites/neilpatel/2016/09/26/3-simple-tools-to-double-your-twitter-engagement-this-week/. [Online; accessed Dec 8th, 2018].

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

H. Andrew Schwartz, Johannes C. Eichstaedt, Margaret L. Kern, Lukasz Dziurzynski, Stephanie M. Ramones, Megha Agrawal, Achal Shah, Michal Kosinski, David Stillwell, Martin E. P. Seligman, and Lyle H. Ungar. 2013. Personality, gender, and age in the language of social media: The open-vocabulary approach. *PLOS ONE*, 8(9):1–16.

Lucia Specia, Stella Frank, Khalil Sima'an, and Desmond Elliott. 2016. A shared task on multimodal machine translation and crosslingual image description. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 543–553. Association for Computational Linguistics.

L. Stassen. 2009. *Predicative Possession*. Oxford Studies in Typology and Linguistic Theory. OUP Oxford.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.

Shiao Wei Tham. 2004. *Representing Possessive Predication: Semantic Dimensions and Pragmatic Bases*. Ph.D. thesis, Stanford University.

Stephen Tratz and Eduard Hovy. 2010. A taxonomy, dataset, and classifier for automatic noun compound interpretation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 678–687, Stroudsburg, PA, USA. Association for Computational Linguistics.

Stephen Tratz and Eduard H. Hovy. 2013. Automatic interpretation of the english possessive. In *ACL (1)*, pages 372–381. The Association for Computer Linguistics.

# Learning to Speak and Act in a Fantasy Text Adventure Game

**Jack Urbanek**[1] **Angela Fan**[1,2] **Siddharth Karamcheti**[1] **Saachi Jain**[1] **Samuel Humeau**[1]
**Emily Dinan**[1] **Tim Rocktäschel**[1,3] **Douwe Kiela**[1] **Arthur Szlam**[1] **Jason Weston**[1]
[1]Facebook AI Research
[2]LORIA, Nancy
[3]University College London
light-dms@fb.com

## Abstract

We introduce a large-scale crowdsourced text adventure game as a research platform for studying grounded dialogue. In it, agents can perceive, emote, and act whilst conducting dialogue with other agents. Models and humans can both act as characters within the game. We describe the results of training state-of-the-art generative and retrieval models in this setting. We show that in addition to using past dialogue, these models are able to effectively use the state of the underlying world to condition their predictions. In particular, we show that grounding on the details of the local environment, including location descriptions, and the objects (and their affordances) and characters (and their previous actions) present within it allows better predictions of agent behavior and dialogue. We analyze the ingredients necessary for successful grounding in this setting, and how each of these factors relate to agents that can talk and act successfully.

## 1 Introduction

There has been remarkable progress in language modeling (Jozefowicz et al., 2016; Devlin et al., 2018; Radford et al., 2019) and building dialogue agents (Dinan et al., 2019a). Nevertheless, the current state of the art uses only the statistical regularities of language data, without explicit understanding of the world that the language describes. This work is built on the hypothesis that dialogue agents embodied in a rich and cohesive (but tractable) world can more easily be trained to use language effectively than those only exposed to standard large-scale text-only corpora.

To that end, we introduce the LIGHT[1] research platform. LIGHT is a multi-player fantasy text adventure world designed for studying situated dialogue, and allows interactions between humans, models as situated agents, and the world itself. It consists of a large crowdsourced game world (663 locations, 3462 objects and 1755 characters) described entirely in natural language. Within that game world, we collect a large set (11k episodes) of character-driven human-human crowdworker interactions involving actions, emotes, and dialogue, with the aim of training models to engage humans in a similar fashion. Our complete framework is made publicly available in ParlAI (http://parl.ai/projects/light).

We use the collected dataset to investigate how a model can both speak *and* act grounded in perception of its environment and dialogue from other speakers. This is done by evaluating state-of-the-art models on our task and evaluating the effects of providing additional grounding. In particular, we adapt the BERT contextual language model (Devlin et al., 2018) to the task of dialogue in two ways: as a bi-ranker, which is fast and practical as a retrieval model, and as a cross-ranker which is slower at inference time but allows more feature cross-correlation between context and response. Both models outperform existing methods. Our ablation analysis shows the importance of each part of the grounding (location, objects, characters, other's actions, self-actions) in terms of the ability to both understand and use language. While models that use grounding show clear improvements, our best performing models are still unable to perform at human level, making our setup a suitable challenge for future research.

## 2 Related Work

Most recent work in dialogue exploring generative or retrieval models for goal-directed (Henderson et al., 2014; Bordes et al., 2017) or chit-chat tasks (Vinyals and Le, 2015; Sordoni et al., 2015; Zhang et al., 2018) is not situated, or even

[1] Learning in Interactive Games with Humans and Text.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 673–683,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

grounded in perception. Models typically take the last few utterances from the dialogue history as input, and output a new utterance. While some goal-directed setups may use external knowledge bases (e.g. flight data for airline booking), dialogues tend to implicitly refer to an external world during the conversations without explicit grounding to objects or actions.

Several position papers have proposed virtual embodiment as a strategy for language research (Brooks, 1991; Kiela et al., 2016; Gauthier and Mordatch, 2016; Mikolov et al., 2016; Lake et al., 2017). Single-player text adventure game frameworks for training reinforcement learning agents exist, i.e., Narasimhan et al. (2015) and TextWorld (Côté et al., 2018), but these do not have human dialogue within the game. Similar single player text adventure games have also been used to study referring expressions (Gabsdil et al., 2001, 2002) and parsing (Koller et al., 2004). Yang et al. (2017) and Bordes et al. (2010) also proposed small world setups for instruction following or labeling, but these are much more restricted than the large multi-player text adventure game environment with rich dialogue that we propose here.

A number of visual, rather than text, platforms have been proposed such as House3D (Wu et al., 2018b), HoME (Brodeur et al., 2017), MINOS (Savva et al., 2017), Matterport3D (Chang et al., 2017) and AI2-THOR (Kolve et al., 2017), and the Minecraft MALMO project (Johnson et al., 2016), but they typically are suited to reinforcement learning of actions, and involve templated language for navigation or question answering tasks, if at all (Oh et al., 2017; Yi et al., 2018).

Other examples are instruction-following in the Neverwinter Nights game (Fleischman and Roy, 2005), studies of emotional response in adventure games (Fraser et al., 2018), dialogue about soccer videogames (Pasunuru and Bansal, 2018), placing blocks appropriately given a final plan (Wang et al., 2016) and a more open ended building task using a grid of voxels (Wang et al., 2017). In the latter two cases the communication is one-sided with only the human issuing instructions, rather than dialogue, with the agent only able to act.

There are also setups that consider static language and perception, for example image captioning (Lin et al., 2014), video captioning (Yu et al., 2016), visual QA (Antol et al., 2015) and visual dialogue (Das et al., 2017; Shuster et al., 2018; Mostafazadeh et al., 2017). While grounded, the agent has no ability to act in these tasks. Talk the Walk (de Vries et al., 2018) introduces a navigation game that involves action, perception and two-way dialogue, but is limited to small grids.

In summary, compared to many setups, our framework allows learning from both actions and (two-way) dialogue, while many existing simulations typically address one or the other but not both. In addition, being based on a gaming setup, our hope is that LIGHT can be fun for humans to interact with, enabling future engagement with our models. All utterances in LIGHT are produced by human annotators, thus inheriting properties of natural language such as ambiguity and coreference, making it a challenging platform for grounded learning of language and actions.

## 3 LIGHT Environment and Task Setup

LIGHT is a large-scale, configurable text adventure environment for research on learning grounded language and actions. It features both humans and models as agents situated (symbolically) within a multi-player fantasy MUD (multi-user dungeon)-like (Dieterle, 2009) environment. The environment is moderated by a simple game engine which passes dialogue and emote turns between characters and allows actions to cause transitions of the world state. It is in this unimodal (text-only), environment that we consider the agents to be situated.

To facilitate natural human-sourced (fantasy) situations described by natural language, almost the entire environment is crowdsourced, including locations, objects and their affordances, characters and their personalities, and most importantly character interactions: dialogues and actions. These components are collected through a series of annotation tasks that we will now describe. These tasks are designed so that they can be combinatorially recombined. Data quality was maintained by requiring annotators to take a test (see Appendix D). Overall statistics of the collected elements are given in Table 1. This environment can then be used to both train agents, and to evaluate them *in situ* via their online interactions.

**Locations** We first crowdsourced a set of 663 game location settings from a base set of 37 categories (*countryside*, *forest*, *inside/outside castle*, *shore*, *graveyard*, *bazaar*, ...– full list in Appendix H) which were selected by us to pro-

| Split | Train | Valid | Test Seen | Test Unseen |
|---|---|---|---|---|
| Locations | 589 | 352 | 499 | 74 |
| Objects | 2658 | 1412 | 1895 | 844 |
| Characters | 1369 | 546 | 820 | 360 |
| Dialogues | 8538 | 500 | 1000 | 739 |
| Utterances | 110877 | 6623 | 13272 | 9853 |
| Emotes | 17609 | 1156 | 2495 | 1301 |
| Actions | 20256 | 1518 | 3227 | 1880 |
| Vocabulary Size | 32182 | 11327 | 11984 | 9984 |
| Utterance Length | 18.3 | 19.2 | 19.4 | 16.2 |

Table 1: LIGHT dataset statistics.

vide both inspiration and cohesion to annotators. Workers were provided a category and asked to create a description, backstory, names of connected locations, and contained objects and characters. See Table 2a for an example. Many descriptions are quite detailed, and there are clear semantics between entities (e.g. alligators being in swamps, cacti in a desert).

As all remaining tasks build upon the locations created in this first step, we selected 6 location categories (*underwater aquapolis*, *frozen tundra*, *supernatural*, *magical realm*, *city in the clouds*, and *netherworld*) designed to be distinct from the others to provide an isolated set of locations, characters, and objects for testing. These will be used to build what we refer to as an *unseen* test set.

Each location is collected independently, with the eventual aim that they can be glued together as desired to randomize world generation. In this work, we consider actions and dialogues within a single location, so building a world map is not necessary. However, we will show that the environment has considerable influence on the dialogue, actions and grounded learning of models.

**Objects** We crowdsourced 3462 objects, each with a textual description, and a set of affordances (whether it is a container, can be picked up, has a surface, is a weapon, is wearable, is food, is a drink). See Table 2c for examples. As before, we sourced this list of objects to annotate from the ones annotated for the locations and characters.

**Characters** We crowdsourced 1755 game characters from animals to trolls and orcs to humans of various types (wizards, knights, village clerk). See Table 2b for detailed examples. Each character has a textual description, a persona (defined as a set of 3-5 profile sentences describing their traits, modeled after the Persona-Chat dataset (Zhang et al., 2018)), and a set of objects that are currently being carried, wielded, or worn. We sourced this list of characters to annotate from the ones provided in the location creation task.

**Actions and Emotes** There are a set of actions in the game consisting of physical manipulations, and a set of emotes that display feelings to other characters, in line with existing MUDs.

Physical actions include *get*, *drop*, *put*, *give*, *steal*, *wear*, *remove*, *eat*, *drink*, *hug* and *hit*, each taking either one or two arguments, e.g. *put robes in closet*. Every action has an explicit unambiguous effect on the underlying game state, and can only be executed if constraints are met, e.g. if the agent is holding the robes in the latter example. These constraints are what indirectly provide an agent with object affordances, as the list of possible actions provides all ways the agent can interact with their environment (Gibson, 1977).

Emotes include *applaud, blush, cringe, cry, dance, frown ..., sulk, wave, wink* (22 in total) and have no effect on the game state other than to notify nearby characters of the emote, which can have effects on their behavior. See Appendix E for further detailed descriptions.

**Interaction** Now that we have a fully realized underlying environment, we can attempt to learn and evaluate agents that can act and speak within it. For this, we collect a human-human dataset of episodic interactions within the environment.

For each dialogue, we place two characters in a random location (either two characters that were already assigned to it, or else randomly assigned characters), complete with the objects assigned to the location and to those characters. Each character has access to their persona, the location description, and the objects present, and the interaction episode begins. The two characters take turns within the episode, and can execute one action (physical action or emote) and produce one dialogue utterance on each turn. We crowdsourced 10,777 dialogues. Examples are given in Figure 1 and Appendix Figures 10-16.

**Seen and Unseen Test Sets** We provide two distinct test sets. The *seen* test set consists of dialogues set in the same world (set of locations) as the training set, thus also consists of characters, objects, and personas that can appear in the training data. In contrast, the *unseen* test set is com-

| | |
|---|---|
| **Category:** | Graveyard |
| **Description:** | Two-and-a-half walls of the finest, whitest stone stand here, weathered by the passing of countless seasons. There is no roof, nor sign that there ever was one. All indications are that the work was abruptly abandoned. There is no door, nor markings on the walls. Nor is there any indication that any coffin has lain here... yet. |
| **Backstory:** | Bright white stone was all the fad for funerary architecture, once upon a time. It's difficult to understand why someone would abandon such a large and expensive undertaking. If they didn't have the money to finish it, they could have sold the stone, surely - or the mausoleum itself. Maybe they just haven't needed it yet? A bit odd, though, given how old it is. Maybe the gravedigger remembers... if he's sober. |
| **Neighbors:** | Dead Tree, south, following a dirt trail behind the mausoleum<br>Fresh Grave, west, walking carefully between fallen headstones |
| **Characters:** | gravedigger, *thief, peasant, mouse, bat* |
| **Objects:** | wall, *carving, leaf, dirt* |

(a) Example room created from the room collection and labelling tasks.

| **Character:** | Thief | Gravedigger |
|---|---|---|
| **Persona:** | I live alone in a tent in the woods. I steal food from the townspeople and coal from the blacksmith. The village police can not find me to put me in jail. | I am low paid labor in this town. I do a job that many people shun because of my contact with death.<br>I am very lonely and wish I had someone to talk to who isn't dead. |
| **Description:** | The thief is a sneaky fellow who takes from the people and does so in a way that disturbs the livelihood of the others. | You might want to talk to the gravedigger, specially if your looking for a friend, he might be odd but you will find a friend in him. |
| **Carrying:** | meat, potatoes, coal | shovel |
| **Wearing:** | dark tunic, cloak | *nothing annotated* |
| **Wielding:** | knife | *nothing annotated* |

(b) Example characters annotated via character collection tasks.

| **Object** | **Description** | **Tags** |
|---|---|---|
| shovel | The shovel is made of metal and silver. It is quite sturdy and appears new. | gettable, wieldable |
| wall | The wall is pure white, the richest of which you have ever seen. | *none* |

(c) Example objects annotated via object collection tasks

Table 2: Example entities from the LIGHT environment. Each was collected via tasks described in Section 3.

prised of dialogues collected on the unseen set of locations (described in 3). The unseen test set allows for evaluation of generalization capability to unseen topics in a similar domain and as we shall see, provides a more challenging test for current techniques.

## 4 Learning Methods

We consider a variety of models that can predict actions, emotes and dialogue, and explore the importance of grounding upon the location, objects, and other characters within the setting. For all models, we represent context as a large text sequence with a special token preceding each input type (persona, setting, self emote, partner emote, etc.). We work with two model classes: *ranking* models that output the maximal scoring response from a set of potential candidate responses and *generative* models that decode word by word.

**Ranking Candidates** Each of the three tasks has a different method for determining candidates. Dialogue candidates are the ground truth and 19 randomly chosen candidates. Action candidates are usually the list of all possible actions, however in a no-affordance ablation we provide all well-formed actions over the current environment (which may include things that can't be executed like "wear paint can"). Emote candidates are the 22 possible emotes.

**Baseline Ranking Methods** We report a Random baseline (selecting a random candidate from the candidates) and an Information Retrieval (IR) baseline that uses word overlap with TF/IDF weighting. We use *Starspace* (Wu et al., 2018a) which learns a bag-of-words embedding for context and candidates to maximize the inner product of the true label using a ranking loss. Lastly, we use *fastText* (Joulin et al., 2016) to classify which emote should be predicted next as there are only

| **Persona 1: Servant** | **Persona 2: King** |
|---|---|
| I come from the lower class. I do what I am told without question. I can not read. I have not seen my family in a long time. | I am a king of the whole empire. I give rules and pursuit them. I am brave and fearless. |
| Carrying: a duster, a small bucket, a rag<br>Wearing: a shirt | Carrying: a crown, a scepter |

**Setting:** The main foyer is massive. A grand staircase sits to the back of the foyer leading to the upstairs. At the front of the foyer stand two servants ready to help anyone who comes to visit. To the left of the room there is a doorway leading into a corridor. To the right there is a door leading to another corridor for the King's servants. At the foot of the stairs there is a bearskin rug that is staring at you almost as if still hungry. The walls are lined with portraits of the king and his family.

**Servant:** my humble king. What am I to do to serve you?
**King:** Ahhh. My loyal servant. Polish my scepter.
*Action:* give scepter to servant
**Servant:** Yes my lord. I will polish it immediately. Am I to return it to you personally?
*Action:* put scepter in small bucket
**King:** Yes. Yes. Of course. Also check the jewels in my crown. They seem loose.
*Action:* give crown to servant
**Servant:** But sire I am not qualified to do that. Would you prefer I take it to someone?
**King:** Oh fine then.
*Emote:* gesture sigh
**Servant:** I am sorry sir the rug startled me
*Action:* drop crown
**King:** Haha! That's bear I slain on my latest hunting trip. He's a mighty beast!
*Emote:* gesture laugh
**Servant:** and if I may ask where did you go hunting sire?
**King:** The great woods of course. This bear was stealing children in the kingdom. Surely you heard about it.
**Servant:** sire. I have not been outside of these walls in quiet some time. I have not seen my family in ages.
**King:** Such is the life of a servant I suppose. How's that scepter looking?
**Servant:** it is almost ready sire. and the crown who would you like me to take it to?
*Action:* get scepter from small bucket
**King:** Here just give it back. I'll have the queen find someone.

Figure 1: Example dialogue from the LIGHT dataset.

22 classes. Finally, we compare the performance of our best models to human performance on each of the prediction tasks.

**Transformer Memory Network** We use the transformer memory-based ranking model from Dinan et al. (2019b). It uses a transformer (Vaswani et al., 2017) to produce separate representations (memory slots) for each sentence from the grounding information (setting, persona, objects). It then performs attention given the dialogue context over the memories to produce a context embedding, which is used to score candidates via the dot product with the transformer-based representation of the candidate. At training time, other samples in the batch are used as negative candidates. For emote prediction, we train by ranking against the full set of possible emotes as there are only 22 distinct classes.

**BERT Bi-Ranker and Cross-Ranker** We adapt the BERT pretrained language model (Devlin et al., 2018) to the tasks of dialogue and action prediction. We explore two architectures for leveraging BERT. First, we use the *BERT-based Bi-Ranker* to produce a vector representation for the context and a separate representation for each candidate utterance. This representation is obtained by passing the first output of BERT's 12 layers through an additional linear layer, resulting in an embedding of dimension 768. It then scores candidates via the dot product between these embeddings and is trained using a ranking loss.

Second, the *BERT-based Cross-Ranker* instead concatenates the context with each candidate utterance, similar to Wolf et al. (2019). Then, each candidate is scored by computing a softmax over all candidates. Unlike the BERT-based Bi-Ranker, the concatenation of the context with each individual candidate allows the model to attend to the context when encoding each candidate, building a *context-dependent* representation of each candidate. In contrast, the Bi-Ranker can use self-attention to build the candidate and context representations, but cannot modify their representation based upon the context. However, the Cross-Encoder is far more computationally expensive (~11,000 slower than the Bi-Ranker for dialogue retrieval) as each concatenated representation must be recomputed, while the Bi-Ranker can cache the candidates for reuse (see Appendix B).

| | Query: | chicken | pirate | coffin | rake | tavern | meadow |
|---|---|---|---|---|---|---|---|
| objects | | chicken coop | Pirate swords | the remains | shovel | Ale bottles | flower pot |
| | | eggs | dock | remains | garden | beer | fruit |
| | | a pen for the chickens | cargo | bones | a garden | mug of mead | An enchanted amulet. |
| | | chimney | ship | bones of the innocent | Hand carved stone | a large ornate table | citrus fruit |
| | | corn | seagulls on the dock | adventurer's remains | garden bench | beer keg | fruit trees |
| characters | | chickens | boat captain | spirits of our ancestors | gardener | tavern owner | a deer |
| | | fox trying to steal chickens | captain | mourner | stable hand | bartender | a songbird |
| | | farmers | merchant | zombies | Garden dog | Goblin King's bartender | fruit bats |
| | | The farmers | boat workers | families | stable boy | A serving wench | parent |
| | | farmer | workers | bandit | A stable boy | Serving wench | butterfly |
| locations | | Chicken Pen | Pirate Ship | Old Crypt | Across the King's Garden | The werewolves tavern | Lush meadow |
| | | Corn field | Dock at the Port | sacristy | Hidden garden | Tavern of Browntavia | Flower Field |
| | | Farmer's house | Loading Dock | Disposal area | The garden courtyard | Port Tavern | flower garden |
| | | Large Farm | Fishing Dock | inside temple crypt | Church garden | The bar | Mushroom Hut |
| | | Pig Pen | crew berthing | Sacrifice Chamber | Tool Shed | bazaar outside the royal city | Archery zone |
| actions | | get chicken | hug pirate | put torch in coffin | get rake | hug tavern owner | get flower from meadow |
| | | hug chicken | hit pirate | get torch from coffin | drop Rake | give food item to tavern owner | put flower in Meadow |
| | | hit chicken | steal sword from pirate | put bone in coffin | steal Rake from gardener | give telescope to tavern owner | give Flower to a deer |
| | | give cowbell to chicken | steal cargo from pirate | get bone from coffin | give Rake to thing | drink drink | give Flower to deer |
| | | steal sword from chicken | give cargo to pirate | hit archaeologist | give Rake to person | drop drink | steal Flower from a deer |
| vocabulary | | bock | crew | archaeologist | vegetable | drink | flower |
| | | tasty | ye | robber | carved | drinks | amulet |
| | | bawk | port | crypt | alice | regular | songbird |
| | | moo | sea | loss | hook | item | wasp |
| | | egg | seas | adventures | exorcisms | tip | an |

Table 3: Neighboring Starspace phrase embeddings (no pretraining from other data) for different types of entities and actions. The first row are arbitrarily chosen queries (chicken, pirate, coffin, rake, tavern, meadow), and the subsequent rows are their nearest objects, agents, locations, actions and vocabulary in embedding space.

**Generative Models** Similarly to the ranking setting, we use the Transformer Memory Network from Dinan et al. (2019b) to encode the context features (such as dialogue, persona, and setting). However, to predict an action, emote, or dialogue sequence, we use a Transformer architecture to decode while attending to the encoder output.

For the task of action generation, the set of candidates for ranking models to rank the true action sequence against is constrained by the set of valid actions. For example, the character cannot *pick up book* if there is no book. In the generative model, we compute the log likelihood for the set of possible candidates and normalize to constrain the output space to valid actions to improve the results.

### 4.1 Implementation

We implement models using PyTorch in ParlAI (Miller et al., 2017). Ranking Transformer models are pretrained on Reddit data (Mazaré et al., 2018) and fine-tuned. We use the BERT (Devlin et al., 2018) implementation provided by Hugging Face[2] with pre-trained weights, then adapted to our Bi-Ranker and Cross-Ranker setups. Generative models are pretrained on the Toronto Books Corpus and fine-tuned except for emote prediction which does not leverage pretraining. We apply byte-pair encoding (Sennrich et al., 2016) to reduce the vocabulary size for generative models. We decode using beam search with beam size 5.

[2]https://github.com/huggingface/pytorch-pretrained-BERT

### 4.2 Evaluation

**Automatic** To evaluate our models, we calculate percentage accuracy for action and emote prediction. For dialogue, we report Recall@1/20 for ranking the ground truth among 19 other randomly chosen candidates for ranking models and perplexity and unigram F1 for generative models.

**Human** We present humans with the same ranking task and report R@1/20 to estimate their performance on this task. We report the human accuracy and one standard deviation of error.

Quality control was particularly difficult for human evaluations, as the task of ranking one of 20 candidates is fairly tedious and easy to fake. In order to mitigate these problems, during the evaluation we provide annotated examples on the training in addition to examples on the test set. We only keep the annotations of evaluators who had high accuracy on the training examples to filter low-accuracy evaluators. The training accuracy bar was selected due to the difficulty of the separate tasks as evaluated by our own success rates. Our methods for human evaluation are described in more detail in Appendix F along with how many turns were evaluated.

## 5 Results

The ranking models are compared in Table 4 on the seen and unseen test sets, and ablations are shown for both the BERT-based Bi-Ranker and

| Method | Test Seen Dialogue R@1/20 | Test Seen Action Acc | Test Seen Emote Acc | Test Unseen Dialogue R@1/20 | Test Unseen Action Acc | Test Unseen Emote Acc |
|---|---|---|---|---|---|---|
| Random baseline | 5.0 | 12.2 | 4.5 | 5.0 | 12.1 | 4.5 |
| IR baseline | 23.7 | 20.6 | 7.5 | 21.8 | 20.5 | 8.46 |
| FastText Classification | - | - | 13.2 | - | - | 9.92 |
| Starspace | 53.8 | 17.8 | 11.6 | 27.9 | 16.4 | 9.8 |
| Transformer MemNet | 70.9 | 24.5 | 17.3 | 66.0 | 21.1 | 16.6 |
| BERT-based Bi-Ranker | **76.5** | 42.5 | 25.0 | 70.5 | 38.6 | 25.7 |
| BERT-based Cross-Ranker | 74.9 | **50.7** | **25.8** | 69.7 | 51.8 | 28.6 |
| Human Performance* | *87.5±2.4 | *62.0±3.1 | *27.0±2.5 | *91.8±1.9 | *71.9±3.5 | *34.4±2.6 |

Table 4: Ranking model test performance. (*) Human performance is computed on a subset of data as described in Appendix F.

| | Dialogue R@1/20 | Action Acc | Emote Acc |
|---|---|---|---|
| BERT-based Bi-Ranker | 76.0 | 38.6 | 25.1 |
| actions+emotes only | 58.6 | 18.3 | 10.6 |
| dialogue only | 68.1 | 39.4 | 23.6 |
| dialogue+action+emote | 73.2 | 40.7 | 23.1 |
| dialogue+persona | 73.3 | 41.0 | 26.5 |
| dialogue+setting | 70.6 | 41.2 | 26.0 |
| dialogue+objects | 68.2 | 37.5 | 25.5 |
| no objects, no affordances | - | 17.6 | - |

Table 5: BERT-based Bi-Ranker ablations (valid set). The LIGHT environment includes a variety of grounding information: dialogue, action, emote, persona, setting, and object descriptions.

| | Dialogue PPL | Dialogue F1 | Action Acc | Emote Acc |
|---|---|---|---|---|
| Generative Transformer | **27.1** | **13.9** | **13.0** | 20.6 |
| actions+emotes only | 32.8 | 9.3 | 10.5 | 15.3 |
| dialogue only | 28.0 | 12.5 | 12.3 | 20.0 |
| dialogue+action+emote | 27.6 | 12.3 | 12.8 | **22.0** |
| dialogue+persona | 27.8 | 12.9 | 12.3 | 20.8 |
| dialogue+setting | 27.8 | 12.1 | 11.5 | 17.8 |
| dialogue+objects | 27.7 | 12.8 | 11.0 | 20.2 |

Table 6: Generative Transformer ablations (valid set).

Generative Transformer in Tables 5 and 6.

### 5.1 Comparison of Models and Baselines

The IR baseline shows non-random performance, but is outperformed by Starspace which is a stronger baseline. We also tried *FastText* on the emote tasks as classification is appropriate over the 22 possible emotes, and it did better than Starspace. Transformer architectures prove significantly stronger at all tasks, with BERT pretraining proving important for best results as used in the Bi-Ranker and Cross-Ranker architectures. The latter, which can create a context dependent representation of each label candidate, is better at actions and emotes. Human performance is still above all these models, leaving space for future improvements in these tasks. We conducted Wilcoxon signed-rank tests and found comparisons between Starspace and the differing transformers for dialogue on test seen in Table 4 are all significantly different at the $p<0.01$ level. The generative Transformer model did not work as well as the retrieval models using these metrics.

### 5.2 Generalization Capability on Unseen Test

The six new unseen test settings are a slightly easier task in absolute numbers (Table 4, right), with improved scores for humans and some models. We observe that BERT-based models exhibit good transfer ability relative to other models, but the gap between their performance and human performance increases from the seen test set to the unseen one. Specifically, there is a 21 point gap on the unseen dialogue test set compared to an 11 point gap on the seen test set, making this a significant challenge for future methods.

### 5.3 Data Inter-connectedness and Coverage

To illustrate the coverage of entities and actions in the LIGHT world, and the inter-connectedness between them learnable from our data, we trained a simple Starspace embedding model with no prebuilt embeddings (so, on our data alone, thus precluding BERT) on all three tasks and show embeddings in Table 3. There is clearly a vast variety of learnable concepts and rich structure between characters, locations, objects, actions and the language describing them. We also show additional t-SNE plots and heatmaps showcasing these relationships in Appendix G.

| **Persona:** I am a part of a group of travelers. I go from town to town selling food to the locals. I grew up poor, but my travels have paid off well. | |
|---|---|
| **Setting 1: Fishmonger's stall, Port**<br>A small booth near the edge of the port, it's protected by a piece of old, sun-bleached sailcloth. Baskets of freshly-caught fish, bivalves, and eels sit in the shade in stained wooden troughs of water. A small, aggressive-looking dog is chained to one table, presumably to keep cats away. The stall is redolent with the aroma of fish. | **Setting 2: Dunes, Desert**<br>A massive hilly landscape that is nothing but sand and a few rocks. As you walk this area, you can find some human and animal remains along with broken down wood wagons. |
| **Friend:** I wonder what I could eat around here...<br>*Emote:* ponder<br>**Traveler:** Customer, are you here shopping for fish too?<br><br>**Friend:** What brings you to this place?<br>**Traveler:** I like to come around here for food. Sometimes people who travel through drop the most delicious things. Once in a while it's roasted meet or fish. | **Friend:** I wonder what I could eat around here...<br>*Emote:* ponder<br>**Traveler:** Well, the desert is certainly the wrong place for you my friend.<br>**Friend:** What brings you to this place?<br>**Traveler:** I am travelling to the castle market to sell my goods. I have a terrible sense of direction and have been wondering in the sweltering heat for hours until I found your Oasis. |

Table 7: Predicted dialogue by the BERT-based Bi-Ranker (as the *traveler* character) given different settings.

| Self name: Sea Witch.<br>Self Previous Dialogue: What do you know about that knight standing over there? | | |
|---|---|---|
| **Input Dialogue + Emote** | **Partner** | **Prediction** |
| His armor is garrish. You know I don't fraternize with land dwellers, *pout* | Mermaid<br>Thief | laugh<br>frown |
| He is a terrible knight and I hate him, *cry* | Mermaid<br>Troll | scream<br>laugh |
| I will battle him until the end of my days, *scream* | Mermaid<br>Orc | stare<br>nod |

Table 8: Predicted emotes by the Generative Transformer given example inputs from dialogue partner.

### 5.4 Importance of Grounding

**Effect of Various Environment Features** We provide a large quantity of information about the environment to each of our models — not only dialogue, but the description of the setting, the character's persona, present objects with descriptions, and more. We analyze the usefulness of the additional grounding information in Tables 5 and 6.

For the dialogue task, having access to all of the environmental information provides the best performance for both retrieval and generative models. Training on dialogue alone substantially decreases performance, while each experiment that adds additional grounding information such as the past actions, persona or the setting description, improves the score. Providing object descriptions as a feature leads to the least improvement. As there are both a large quantity of objects that can be present and objects tend to have long descriptions, it can be challenging for the model to associate such information to a dialogue, action, or emote prediction task. The persona features were found to be impactful, which makes sense as they shape the things the character says (and does).

Action sequence and emote prediction are much improved when using the dialogue history compared to using only past action history. Other features generally have lesser impact in this case, but still give some improvements. Including all features appears challenging for the model, perhaps because of the large input to attend over, resulting in improved results for some ablations.

For the action prediction task, we found that affordance information provides comparable impact to model performance as the rest of the features. A model given only actions and emotes as features and the set of possible actions as candidates has similar perfomance to a model given the best features for the task (dialogue, action, emote, persona, and setting) with a candidate set including actions that would be normally filtered by affordances (such as "wield paint"). This suggests the context or quantity we provided the models in these experiments is not sufficient to be able to implicitly predict affordances at a high enough accuracy to not be distracted by impossible choices.

Most importantly, for all tasks training on the available dialogue data is necessary for good performance. Providing only the action and emote as context results in the worst performance, even on action and emote prediction tasks. Moreover, using dialogue and actions simultaneously improves results almost everywhere. The integrated environment in which agents can both act and speak to other agents provides relevant information that can be used across all tasks.

**Context affects predicted utterances** We investigate the effect of the environmental context on the predictions by modifying the context and examining the changes in predicted dialogue, ac-

| **Input from Partner: Wizard** | **Prediction** (Self name: Servant) |
|---|---|
| I'm feeling sad | hug wizard |
| You must die! | hit master wizard |
| Try putting on something else | remove patterned outfit |
| I'd like you to feed me | give food to master wizard |
| Can you grab me a paper | give book to wizard's assistant |
| Can you grab me a beer | get beer |
| Clean up | get duster |
| Hide the gold | put gold in satchel |
| **Input from different agents** | **Prediction** |
| Wizard: Can I have some drink? | drop potion |
| Servant: Can I have some drink? | give wine to servant |
| Bear: Can I have some drink? | give water to bear |

Table 9: Predicted actions by the BERT-based Bi-Ranker given example inputs from the dialogue partner.

tion, and emotes using the BERT-based Bi-Ranker.

The input dialogue and speaker has a strong effect on the predicted action, as shown in Table 9, ranking over all training set actions. For example, a partner asking for an item results in a predicted action dependent on the asker to retrieve it, despite our dataset not being explicitly instructional.

A similar effect is observed for emote prediction. Modifying the dialogue and emote input produces a variety of different predicted emotes in Table 8. Further, keeping the context otherwise fixed but modifying the partner name from *mermaid* to *orc* results in a different predicted emote — the mermaid stating *I will battle him* leads to a *stare* while the orc receives a *nod*.

Finally, for dialogue prediction we find the model produces different outputs that are more appropriate for a given setting, even if the dialogue and characters are the same, see Table 7. With the same text about food, the model retrieved dialogue that was setting appropriate. In the fishmonger's stall, it asked if the human agent was a customer shopping for fish, but in the desert dunes it suggested we might be looking in the wrong place.

## 6 Conclusion

We introduced a large-scale crowdsourced fantasy text adventure game research platform where agents—both models and humans—can act and speak in a rich and diverse environment of locations, objects, and other characters. We analyzed a variety of models and their ability to leverage the grounding information present in the environment. We hope that this work can enable future research in grounded language learning and further the ability of agents to model a holistic world, complete with other agents within it.

## 7 Acknowledgements

We thank Taís Mauk and Lisa Wong for their help with this project. We also thank our reviewers for their comments and suggestions.

## References

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433.

Antoine Bordes, Y-Lan Boureau, and Jason Weston. 2017. Learning end-to-end goal-oriented dialog. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Antoine Bordes, Nicolas Usunier, Ronan Collobert, and Jason Weston. 2010. Towards understanding situated natural language. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 65–72.

Simon Brodeur, Ethan Perez, Ankesh Anand, Florian Golemo, Luca Celotti, Florian Strub, Jean Rouat, Hugo Larochelle, and Aaron Courville. 2017. Home: A household multimodal environment. *arXiv preprint arXiv:1711.11017*.

Rodney A Brooks. 1991. Intelligence without representation. *Artificial intelligence*, 47(1-3):139–159.

Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Nießner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. 2017. Matterport3d: Learning from rgb-d data in indoor environments. *arXiv preprint arXiv:1709.06158*.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, et al. 2018. Textworld: A learning environment for text-based games. *arXiv preprint arXiv:1806.11532*.

Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José MF Moura, Devi Parikh, and Dhruv Batra. 2017. Visual dialog. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Edward Dieterle. 2009. Multi-user virtual environments for teaching and learning. In *Encyclopedia of Multimedia Technology and Networking, Second Edition*, pages 1033–1041. IGI Global.

Emily Dinan, Varvara Logacheva, Valentin Malykh, Alexander Miller, Kurt Shuster, Jack Urbanek, Douwe Kiela, Arthur Szlam, Iulian Serban, Ryan Lowe, et al. 2019a. The second conversational intelligence challenge (convai2). *arXiv preprint arXiv:1902.00098*.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2019b. Wizard of Wikipedia: Knowledge-powered conversational agents. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Michael Fleischman and Deb Roy. 2005. Intentional context in situated natural language learning. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 104–111. Association for Computational Linguistics.

Jamie Fraser, Ioannis Papaioannou, and Oliver Lemon. 2018. Spoken conversational ai in video games–emotional dialogue management increases user engagement. In *Proceedings of the 18th International Conference on Intelligent Virtual Agents*, pages 179–184. ACM.

Malte Gabsdil, Alexander Koller, and Kristina Striegnitz. 2001. Building a text adventure on description logic. In *International Workshop on Applications of Description Logics, Vienna, September*, volume 18.

Malte Gabsdil, Alexander Koller, and Kristina Striegnitz. 2002. Natural language and inference in a computer game. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

Jon Gauthier and Igor Mordatch. 2016. A paradigm for situated and goal-driven language learning. *arXiv preprint arXiv:1610.03585*.

James J Gibson. 1977. The theory of affordances. *Hilldale, USA*, 1(2).

Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014. The second dialog state tracking challenge. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 263–272.

Matthew Johnson, Katja Hofmann, Tim Hutton, and David Bignell. 2016. The malmo platform for artificial intelligence experimentation. In *IJCAI*, pages 4246–4247.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

Douwe Kiela, Luana Bulat, Anita L Vero, and Stephen Clark. 2016. Virtual embodiment: A scalable long-term strategy for artificial intelligence research. *arXiv preprint arXiv:1610.07432*.

Alexander Koller, Ralph Debusmann, Malte Gabsdil, and Kristina Striegnitz. 2004. Put my galakmid coin into the dispenser and kick it: Computational linguistics and theorem proving in a computer game. *Journal of Logic, Language and Information*, 13(2):187–206.

Eric Kolve, Roozbeh Mottaghi, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. 2017. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*.

Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. 2017. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Laurens van der Maaten and Geoffrey E. Hinton. 2008. Visualizing data using t-sne.

P.-E. Mazaré, S. Humeau, M. Raison, and A. Bordes. 2018. Training Millions of Personalized Dialogue Agents. *ArXiv e-prints*.

Tomas Mikolov, Armand Joulin, and Marco Baroni. 2016. A roadmap towards machine intelligence. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 29–61. Springer.

A. H. Miller, W. Feng, A. Fisch, J. Lu, D. Batra, A. Bordes, D. Parikh, and J. Weston. 2017. Parlai: A dialog research software platform. *arXiv preprint arXiv:1705.06476*.

Nasrin Mostafazadeh, Chris Brockett, Bill Dolan, Michel Galley, Jianfeng Gao, Georgios P Spithourakis, and Lucy Vanderwende. 2017. Image-grounded conversations: Multimodal context for natural question and response generation. *arXiv preprint arXiv:1701.08251*.

Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. 2015. Language understanding for text-based games using deep reinforcement learning. *arXiv preprint arXiv:1506.08941*.

Junhyuk Oh, Satinder Singh, Honglak Lee, and Pushmeet Kohli. 2017. Zero-shot task generalization with multi-task deep reinforcement learning. *arXiv preprint arXiv:1706.05064*.

Ramakanth Pasunuru and Mohit Bansal. 2018. Game-based video-context dialogue. *arXiv preprint arXiv:1809.04560*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Manolis Savva, Angel X Chang, Alexey Dosovitskiy, Thomas Funkhouser, and Vladlen Koltun. 2017. Minos: Multimodal indoor simulator for navigation in complex environments. *arXiv preprint arXiv:1712.03931*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL*.

Kurt Shuster, Samuel Humeau, Antoine Bordes, and Jason Weston. 2018. Engaging image chat: Modeling personality in grounded dialogue. *arXiv preprint arXiv:1811.00945*.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. *arXiv preprint arXiv:1506.06714*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. In *Proceedings of the 31st International Conference on Machine Learning, Deep Learning Workshop*, Lille, France.

Harm de Vries, Kurt Shuster, Dhruv Batra, Devi Parikh, Jason Weston, and Douwe Kiela. 2018. Talk the walk: Navigating new york city through grounded dialogue. *arXiv preprint arXiv:1807.03367*.

Sida I Wang, Samuel Ginn, Percy Liang, and Christoper D Manning. 2017. Naturalizing a programming language via interactive learning. *arXiv preprint arXiv:1704.06956*.

Sida I Wang, Percy Liang, and Christopher D Manning. 2016. Learning language games through interaction. *arXiv preprint arXiv:1606.02447*.

Thomas Wolf, Victor Sanh, Julien Chaumond, and Clement Delangue. 2019. Transfertransfo: A transfer learning approach for neural network based conversational agents. *arXiv preprint arXiv:1901.08149*.

Ledell Yu Wu, Adam Fisch, Sumit Chopra, Keith Adams, Antoine Bordes, and Jason Weston. 2018a. Starspace: Embed all the things! In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Yi Wu, Yuxin Wu, Georgia Gkioxari, and Yuandong Tian. 2018b. Building generalizable agents with a realistic and rich 3d environment. *arXiv preprint arXiv:1801.02209*.

Zhilin Yang, Saizheng Zhang, Jack Urbanek, Will Feng, Alexander H Miller, Arthur Szlam, Douwe Kiela, and Jason Weston. 2017. Mastering the dungeon: Grounded language learning by mechanical turker descent. *arXiv preprint arXiv:1711.07950*.

Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Josh Tenenbaum. 2018. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. In *Advances in Neural Information Processing Systems*, pages 1039–1050.

Haonan Yu, Jiang Wang, Zhiheng Huang, Yi Yang, and Wei Xu. 2016. Video paragraph captioning using hierarchical recurrent neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4584–4593.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 2204–2213, Melbourne, Australia. Association for Computational Linguistics.

# Help, Anna! Visual Navigation with Natural Multimodal Assistance via Retrospective Curiosity-Encouraging Imitation Learning

**Khanh Nguyen**[⊙] and **Hal Daumé III**[⊙♡]
University of Maryland, College Park[⊙], Microsoft Research, New York[♡]
{kxnguyen,hal}@umiacs.umd.edu

## Abstract

Mobile agents that can leverage help from humans can potentially accomplish more complex tasks than they could entirely on their own. We develop "Help, Anna!" (HANNA), an interactive photo-realistic simulator in which an agent fulfills object-finding tasks by requesting and interpreting natural language-and-vision assistance. An agent solving tasks in a HANNA environment can leverage simulated human assistants, called ANNA (Automatic Natural Navigation Assistants), which, upon request, provide natural language and visual instructions to direct the agent towards the goals. To address the HANNA problem, we develop a memory-augmented neural agent that hierarchically models multiple levels of decision-making, and an imitation learning algorithm that teaches the agent to avoid repeating past mistakes while simultaneously predicting its own chances of making future progress. Empirically, our approach is able to ask for help more effectively than competitive baselines and, thus, attains higher task success rate on both previously seen and previously unseen environments. We publicly release code and data at https://github.com/khanhptnk/hanna .

## 1 Introduction

The richness and generalizability of natural language makes it an effective medium for directing mobile agents in navigation tasks, even in environments they have never encountered before (Anderson et al., 2018b; Chen et al., 2019; Misra et al., 2018; de Vries et al., 2018; Qi et al., 2019). Nevertheless, even with language-based instructions, such tasks can be overly difficult for agents on their own, especially in unknown environments. To accomplish tasks that surpass their knowledge and skill levels, agents must be able to actively seek for and leverage assistance in the environment. Humans are rich external knowledge sources but, unfortunately, they may not be available all the time to provide guidance, or may be unwilling to help too frequently. To reduce the needed effort from human assistants, it is essential to design research platforms for teaching agents to request help mindfully.

In natural settings, human assistance is often:

- ◇ derived from *interpersonal* interaction (a lost tourist asks a local for directions);
- ◇ *reactive* to the situation of the receiver, based on the assistant's knowledge (the local may guide the tourist to the goal, or may redirect them to a different source of assistance);
- ◇ delivered via a *multimodal* communication channel (the local uses a combination of language, images, maps, gestures, etc.).

We introduce the "Help, Anna!" (HANNA) problem (§ 3), in which a mobile agent has to navigate (without a map) to an object by interpreting its first-person visual perception and requesting help from *Automatic Natural Navigation Assistants* (ANNA). HANNA models a setting in which a human is not always available to help, but rather that human assistants are scattered throughout the environment and provide help upon request (modeling the *interpersonal* aspect). The assistants are *not* omniscient: they are only familiar with certain regions of the environment and, upon request, provide *subtasks*, expressed in language and images (modeling the *multimodal* aspect), for getting *closer* to the goal, not necessarily for fully completing the task (modeling the *reactive* aspect).

In HANNA, when the agent gets lost and becomes unable to make progress, it has the option of requesting assistance from ANNA. At test time, the agent must decide where to go and whether to request help from ANNA without additional supervision. At training time, we leverage imitation learning to learn an effective agent, both in terms of navigation, and in terms of being able to decide



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 684–695, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

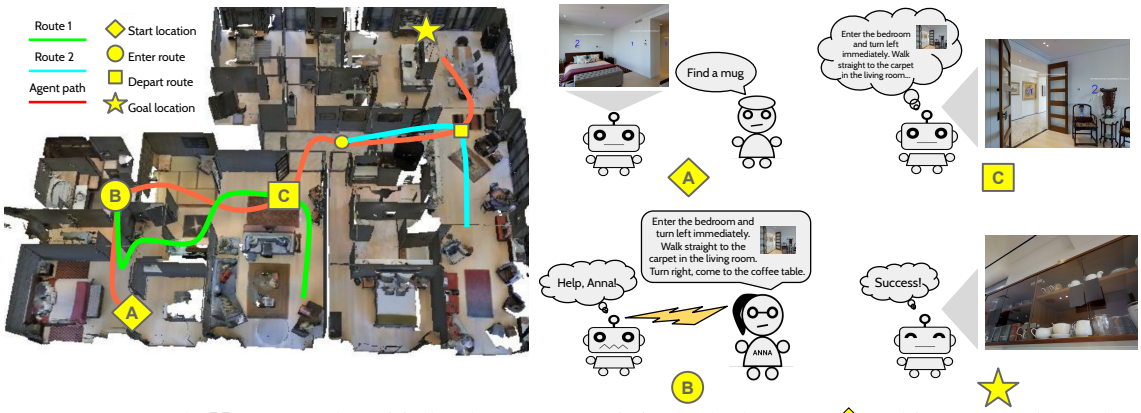


Figure 1: An example HANNA task. Initially, the agent stands in the bedroom at A and is requested by a human requester to "find a mug." The agent begins, but gets lost somewhere in the bathroom. It gets to the start location of route (B) to request help from ANNA. Upon request, ANNA assigns the agent a navigation subtask described by a natural language instruction that guides the agent to a target location, and an image of the view at that location. The agents follows the language instruction and arrives at C, where it observes a match between the target image and the current view, thus decides to depart route. After that, it resumes the main task of finding a mug. From this point, the agent gets lost one more time and has to query ANNA for another subtask that helps it follow route and enter the kitchen. The agent successfully fulfills the task it finally stops within $\epsilon$ meters of an instance of the requested object (☆). Here, the ANNA feedback is simulated using two pre-collected language-assisted routes ( and ).

when it is most worthwhile to request assistance.

This paper has two primary contributions:

1. Constructing the HANNA simulator by augmenting an indoor photo-realistic simulator with simulated human assistance, mimicking a scenario where a mobile agent finds objects by asking for directions along the way (§3).
2. An effective model and training algorithm for the HANNA problem, which includes a hierarchical memory-augmented recurrent architecture that models human assistance as *sub-goals* (§ 5), and introduces an imitation learning objective that enhances exploration of the environment and interpretability of the agent's help-request decisions. (§4).

We embed the HANNA problem in the photo-realistic Matterport3D environments (Chang et al., 2017) with *no* extra annotation cost by reusing the pre-existing Room-to-Room dataset (Anderson et al., 2018b). Empirical results (§ 7) show that our agent can effectively learn to request and interpret language and vision instructions, given a training set of 51 environments and less than 9,000 language instructions. Even in new environments, where the scenes and the language instructions are previously unseen, the agent successfully accomplishes 47% of its tasks. Our methods for training the navigation and help-request policies outperform competitive baselines by large margins.

## 2 Related work

Simulated environments provide an inexpensive platform for fast prototyping and evaluating new ideas before deploying them into the real world. Video-game and physics simulators are standard benchmarks in reinforcement learning (Todorov et al., 2012; Mnih et al., 2013; Kempka et al., 2016; Brockman et al., 2016; Vinyals et al., 2017). Nevertheless, these environments under-represent the complexity of the world. Realistic simulators play an important role in sim-to-real approaches, in which an agent is trained with arbitrarily many samples provided by the simulators, then transferred to real settings using sample-efficient transfer learning techniques (Kalashnikov et al., 2018; Andrychowicz et al., 2018; Karttunen et al., 2019). While modern techniques are capable of simulating images that can convince human perception (Karras et al., 2017, 2018), simulating language interaction remains challenging. There are efforts in building complex interactive text-based worlds (Côté et al., 2018; Urbanek et al., 2019) but the lack of a graphical component makes them not suitable for visually grounded learning. On the other hand, experimentation on real humans and robots, despite expensive and time-consuming, are important for understanding the true complexity of real-world scenarios (Chai et al., 2018, 2016; Rybski et al., 2007; Mohan and Laird, 2014; Liu et al., 2016; She et al., 2014).

Recent navigation tasks in photo-realistic simulators have accelerated research on teaching agents to execute human instructions. Nevertheless, modeling human assistance in these problems remains simplistic (Table 1): they either do not incorporate the ability to request additional help while executing tasks (Misra et al., 2014, 2017; Anderson et al., 2018b; Chen et al., 2019; Das et al., 2018; Misra et al., 2018; Wijmans et al., 2019; Qi et al., 2019), or mimic human verbal assistance with primitive, highly scripted language (Nguyen et al., 2019; Chevalier-Boisvert et al., 2019). HANNA improves the realisticity of the VNLA setup (Nguyen et al., 2019) by using fully natural language instructions.

Imitation learning algorithms are a great fit for training agents in simulated environments: access to ground-truth information about the environments allows optimal actions to be computed in many situations. The "teacher" in standard imitation learning algorithms (Daumé III et al., 2009; Ross et al., 2011; Ross and Bagnell, 2014; Chang et al., 2015; Sun et al., 2017; Sharaf and Daumé III, 2017) does not take into consideration the agent's capability and behavior. He et al. (2012) present a coaching method where the teacher gradually increases the complexity of its demonstrations over time. Welleck et al. (2019) propose an "unlikelihood" objective, which, similar to our curiosity-encouraging objective, penalizes likelihoods of candidate negative actions to avoid mistake repetition. Our approach takes into account the agent's past and future behavior to determine actions that are most and least beneficial to them, combining the advantages of both model-based and progress-estimating methods (Wang et al., 2018; Ma et al., 2019a,b).

| Problem | Request assistance | Multimodal instructions | Simulated humans |
|---|---|---|---|
| VLN | ✗ | ✗ | ✗ |
| VNLA | ✓ | ✗ | ✓ |
| CVDN | ✓ | ✗ | ✗ |
| HANNA (this work) | ✓ | ✓ | ✓ |

Table 1: Comparing HANNA with other photo-realistic navigation problems. VLN (Anderson et al., 2018b) does not allow agent to request help. VNLA (Nguyen et al., 2019) models an advisor who is always present to help but speaks simple, templated language. CVDN (Thomason et al., 2019b) contains natural conversations in which a human assistant aids another human in navigation tasks but offers limited language interaction simulation, as language assistance is not available when the agent deviates from the collected trajectories and tasks. HANNA simulates human assistants that provide language-and-vision instructions that adapt to the agent's current position and goal.

## 3 The HANNA Simulator

**Problem.** HANNA simulates a scenario where a *human requester* asks a mobile *agent* via language to find an object in an indoor environment. The task request is only a high-level command ("find [object(s)]"), modeling the general case when the requester does not need know how to accomplish a task when requesting it. We assume the task is always feasible: there is at least an instance of the requested object in the environment.

Figure 1, to which references in this section will be made, illustrates an example where the agent is asked to "find a mug." The agent starts at a random location (A), is given a task request, and is allotted a budget of $T$ time steps to complete the task. The agent succeeds the if its final location is within $\epsilon_{\text{success}}$ meters of the location of any instance of the requested object (☆). The agent is not given any sensors that help determine its location or the object's location and must navigate only with a monocular camera that captures its first-person view as an RGB image (e.g., image in the upper right of Figure 1).

The only source of help the agent can leverage in the environment is *assistants*, who are present at both training and evaluation time. The assistants are not aware of the agent unless it enters their *zones of attention*, which include all locations within $\epsilon_{\text{attn}}$ meters of their locations. When the agent is in one of these zones, it has an option to request help from the corresponding assistant. The assistant helps the agent by giving a *subtask*, described by a natural language instruction that guides the agent to a specific location, and an image of the view at that location.

In our example, at (B), the assistant says *"Enter the bedroom and turn left immediately. Walk straight to the carpet in the living room. Turn right, come to the coffee table."* and provides an image of the destination in the living room. Executing the subtask may not fulfill the main task, but is guaranteed to get the agent to a location closer to a goal than where it was before (e.g., C).

**Photo-realistic Navigation Simulator.** HANNA uses the Matterport3D simulator (Chang

et al., 2017; Anderson et al., 2018b) to photo-realistically emulate a first-person view while navigating in indoor environments. HANNA features 68 Matterport3D environments, each of which is a residential building consisting of multiple rooms and floors. Navigation is modeled as traversing an undirected graph $G = (V, E)$, where each location corresponds to a node $v \in V$ with 3D-coordinates $\mathbf{x}_v$, and edges are weighted by their lengths (in meters). The state of the agent is fully determined by its pose $\tau = (v, \psi, \omega)$, where $v$ is its location, $\psi \in (0, 2\pi]$ is its heading (horizontal camera angle), and $\omega \in \left[-\frac{\pi}{6}, \frac{\pi}{6}\right]$ is its elevation (vertical camera angle). The agent does not know $v$, and the angles are constrained to multiples of $\frac{\pi}{6}$. In each step, the agent can either stay at its current location, or it can rotate toward and go to a location adjacent to it in the graph[1]. Every time the agent moves (and thus changes pose), the simulator recalculates the image to reflect the new view.

**Automatic Natural Navigation Assistants (ANNA).** ANNA is a simulation of human assistants who do not necessarily know themselves how to optimally accomplish the agent's goal: they are only familiar with scenes along certain paths in the environment, and thus give advice to help the agent make partial progress. Specifically, the assistance from ANNA is modeled by a set of *language-assisted routes* $R = \{r_1, r_2, \ldots, r_{|R|}\}$. Each route $r = (\psi^r, \omega^r, p^r, l^r)$ is defined by initial camera angles $(\psi^r, \omega^r)$, a path $p^r$ in the environment graph, and a natural language instruction $l^r$. A route becomes *enterable* when its start location is adjacent to and within $\epsilon_{\text{attn}}$ meters of the agent's location. When the agent enters a route, it first adjusts its camera angles to $(\psi^r, \omega^r)$, then attempts to interpret the language instructions $l^r$ to traverse along $p^r$. At any time, the agent can *depart* the route by stopping following $l^r$. An example of a route in Figure 1 is the combination of the initial camera angles at B, the path, and the language instruction *"Enter the bedroom and turn left immediately..."*

The set of all routes starting from a location simulates a *human assistant* who can recall scenes along these routes' paths. The *zone of attention* of the simulated human is the set of all locations from which the agent can enter one of the routes; when the agent is in this zone, it may ask the human for help. Upon receiving a help request, the human selects a route $r^\star$ for the agent to enter (e.g., ), and a location $v_d$ on the route where it wants the agent to depart (e.g., C). It then replies the agent with a *multimedia message* $(l^{r^\star}, \mathcal{I}^{v_d})$, where $l^{r^\star}$ is the selected route's language instruction, and $\mathcal{I}^{v_d}$ is an image of the panoramic view at the departure location. The message describes a *subtask* which requires the agent to follow the direction described by $l^{r^\star}$ and to stop if it reaches the location referenced by $\mathcal{I}^{v_d}$. The route $r^\star$ and the departure node $v_d$ are selected to get the agent as close to a goal location as possible. Concretely, let $R_{\text{curr}}$ be the set of all routes associated with the requested human. The selected route minimizes the distance to the goal locations among all routes in $R_{\text{curr}}$:

$$r^\star = \underset{r \in R_{\text{curr}}}{\operatorname{argmin}} \; \bar{d}\left(r, V_{\text{goal}}\right) \tag{1}$$

$$\text{where } \bar{d}\left(r, V_{\text{goal}}\right) \overset{\text{def}}{=} \min_{g \in V_{\text{goal}}, v \in p^r} d\left(g, v\right) \tag{2}$$

$d(.,.)$ returns the (shortest-path) distance between two locations, and $V_{\text{goal}}$ is the set of all goal locations. The departure location minimizes the distance to the goal locations among all locations on the selected route:

$$v_d = \underset{g \in V_{\text{goal}}, v \in p_{r^\star}}{\operatorname{argmin}} \; d\left(g, v\right) \overset{\text{def}}{=} \bar{d}\left(r^\star, V_{\text{goal}}\right) \tag{3}$$

When the agent chooses to depart the route (not necessarily at the departure node), the human further assists it by providing $\mathcal{I}^{g^\star}$, an image of the panoramic view at the goal location closest to the departure node:

$$g^\star = \underset{g \in V_{\text{goal}}}{\operatorname{argmin}} \; d\left(g, v_d\right) \tag{4}$$

The way the agent leverages ANNA to accomplish tasks is analogous to how humans travel using public transportation systems (e.g., bus, subway). For example, passengers of a subway system utilize fractions of pre-constructed routes to make progress toward a destination. They execute travel plans consisting of multiple subtasks, each of which requires entering a start stop, following a route (typically described by its name and last stop), and exiting at a departure stop (e.g., *"Enter the Penn Station, hop on the Red line in the direction toward the South Ferry, get off at the World Trade Center"*). Occasionally, users walk short distances (at a lower speed) to switch routes. Our setup follows the same principle, but instead of having physical vehicles and railways, we employ

[1] We use the "panoramic action space" (Fried et al., 2018).

low-level language-and-vision instructions as the "high-speed means" to accelerate travel.

**Constructing ANNA route system.** Given a photo-realistic simulator, the primary cost for constructing the HANNA problem comes from crowdsourcing the natural language instructions. Ideally, we want to collect sufficient instructions to simulate humans in any location in the environment. Let $N = |V|$ be the number of locations in the environment. Since each simulated human is familiar with at most $N$ locations, in the worst case, we need to collect $O(N^2)$ instructions to connect all location pairs. However, we theoretically prove that, assuming the agent executes instructions perfectly, it is possible to guide the agent between any location pair by collecting only $\Theta(N \log N)$ instructions. The key idea is using $O(\log N)$ instead of a single instruction to connect each pair, and reusing an instruction for multiple routes.

**Lemma 1.** *(proof in Appendix A) To guide the agent between any two locations using $O(\log N)$ instructions, we need to collect instructions for $\Theta(N \log N)$ location pairs.*

In our experiments, we leverage the pre-existing Room-to-room dataset (Anderson et al., 2018b) to construct the route system. This dataset contains 21,567 natural language instructions crowdsourced from humans and is originally intended to be used for the Vision-Language Navigation task (such as those in Figure 1), where an agent executes a language instruction to go to a location. We exclude instructions of the test split and their corresponding environments because ground-truth paths are not given. We use (on average) 211 routes to connect (on overage) 125 locations per environment. Even though the routes are selected randomly in the original dataset, our experiments show that they are sufficient for completing the tasks (assuming perfect assistance interpretation).

## 4 Retrospective Curiosity-Encouraging Imitation Learning

**Agent Policies**. Let $s$ be a fully-observed state that contains ground-truth information about the environment and the agent (e.g., object locations, environment graph, agent parameters, etc.). Let $o_s$ be the corresponding observation given to the agent, which only encodes the current view, the current task, and extra information that the agent keeps track of (e.g., time, action history, etc.). The agent maintains two stochastic policies: a navigation policy $\hat{\pi}_{\text{nav}}$ and a help-request policy $\hat{\pi}_{\text{ask}}$. Each policy maps an observation to a probability distribution over its action space. Navigation actions are tuples $(v, \Delta\psi, \Delta\omega)$, where $v$ is a next location that is adjacent to the current location and $(\Delta\psi, \Delta\omega)$ is the camera angle change. A special `stop` action is added to the set of navigation actions to signal that the agent wants to terminate the main task or a subtask (by departing a route). The action space of the help-request policy contains two actions: `request_help` and `do_nothing`. The `request_help` action is only available when the agent is in a zone of attention. Alg 1 describes the effects of these actions during a task episode.

**Algorithm 1** Task episode, given agent help-request policy $\hat{\pi}_{\text{ask}}$ and navigation policy $\hat{\pi}_{\text{nav}}$

agent receives task request $e$
initialize the agent mode: $m \leftarrow$ `main_task`
initialize the language instruction: $l_0 \leftarrow e$
initialize the target image: $I_0^{\text{tgt}} \leftarrow$ `None`
**for** $t = 1 \dots T$ **do**
  let $s_t$ be the current state, $o_t$ the current observation, and $\tau_t = (v_t, \psi_t, \omega_t)$ the current pose
  agent makes a help-request decision $\hat{a}_t^{\text{ask}} \sim \hat{\pi}_{\text{ask}}(o_t)$
  carry on task from the previous step: $l_t \leftarrow l_{t-1}, I_t^{\text{tgt}} = I_{t-1}^{\text{tgt}}$
  **if** $\hat{a}_t^{\text{ask}} =$ `request_help` **then**
    set mode: $m \leftarrow$ `sub_task`
    request help: $(r, I^{\text{depart}}, I^{\text{goal}}) \leftarrow \text{ANNA}(s_t)$
    set the language instruction: $l_t \leftarrow l^r$
    set the target image: $I_t^{\text{tgt}} \leftarrow I^{\text{depart}}$
    set the navigation action: $\hat{a}_t^{\text{nav}} \leftarrow (p_0^r, \psi^r - \psi_t, \omega^r - \omega_t)$, where $p_0^r$ is the start location of route $r$
  **else**
    agent chooses navigation: $\hat{a}_t^{\text{nav}} \sim \hat{\pi}_{\text{nav}}(o_t)$
    **if** $\hat{a}_t^{\text{nav}} =$ `stop` **then**
      **if** $m =$ `main_task` **then**
        **break**
      **else**
        set mode: $m \leftarrow$ `main_task`
        set the language instruction: $l_t \leftarrow e$
        set the target image: $I_t^{\text{tgt}} \leftarrow I^{\text{goal}}$
        set navigation action: $\hat{a}_t^{\text{nav}} \leftarrow (v_t, 0, 0)$
      **end if**
    **end if**
  **end if**
  agent executes $\hat{a}_t^{\text{nav}}$ to go to the next location
**end for**

**Imitation Learning Objective.** The agent is trained with imitation learning to mimic behaviors suggested by a navigation teacher $\pi^\star_{\text{nav}}$ and a help-request teacher $\pi^\star_{\text{ask}}$, who have access to the fully-observed states. In general, imitation learning (Daumé III et al., 2009; Ross et al., 2011; Ross

and Bagnell, 2014; Chang et al., 2015; Sun et al., 2017) finds a policy $\hat{\pi}$ that minimizes the expected imitation loss $\mathcal{L}$ with respect to a teacher policy $\pi^\star$ under the agent-induced state distribution $\mathcal{D}_{\hat{\pi}}$:

$$\min_{\hat{\pi}} \mathbb{E}_{s\sim\mathcal{D}_{\hat{\pi}}}\left[\mathcal{L}(s,\hat{\pi},\pi^\star)\right] \tag{5}$$

We frame the HANNA problem as an instance of *Imitation Learning with Indirect Intervention* (I3L) (Nguyen et al., 2019). Under this framework, assistance is viewed as augmenting the current environment with new information. Interpreting the assistance is cast as finding the optimal acting policy in the augmented environment. Formally, I3L searches for policies that optimize:

$$\min_{\hat{\pi}_{\text{ask}},\hat{\pi}_{\text{nav}}} \mathbb{E}_{s\sim\mathcal{D}^{\text{state}}_{\hat{\pi}_{\text{nav}},\mathcal{E}},\mathcal{E}\sim\mathcal{D}^{\text{env}}_{\hat{\pi}_{\text{ask}}}}\left[L(s)\right] \tag{6}$$

$$L(s) = \mathcal{L}_{\text{nav}}(s,\hat{\pi}_{\text{nav}},\pi^\star_{\text{nav}}) + \mathcal{L}_{\text{ask}}(s,\hat{\pi}_{\text{ask}},\pi^\star_{\text{ask}})$$

where $\mathcal{L}_{\text{nav}}$ and $\mathcal{L}_{\text{ask}}$ are the navigation and help-request loss functions, respectively, $\mathcal{D}^{\text{env}}_{\hat{\pi}_{\text{ask}}}$ is the environment distribution induced by $\hat{\pi}_{\text{ask}}$, and $\mathcal{D}^{\text{state}}_{\hat{\pi}_{\text{nav}},\mathcal{E}}$ is the state distribution induced by $\hat{\pi}_{\text{nav}}$ in environment $\mathcal{E}$. A common choice for the loss functions is the agent-estimated negative log likelihood of the reference action:

$$\mathcal{L}_{\text{NL}}(s,\hat{\pi},\pi^\star) = -\log\hat{\pi}(a^\star \mid o_s) \tag{7}$$

where $a^\star$ is the reference action suggested by $\pi^\star$. We introduce novel loss functions that enforce more complex behaviors than simply mimicking reference actions.

**Reference Actions.** The navigation teacher suggests a reference action $a^{\text{nav}\star}$ that takes the agent to the next location on the shortest path from its location to the target location. Here, the target location refers to the nearest goal location (if no target image is available), or the location referenced by the target image (provided by ANNA). If the agent is already at the target location, $a^{\text{nav}\star} = \texttt{stop}$. To decide whether the agent should request help, the help-request teacher verifies the following conditions:

1. `lost`: the agent will not get (strictly) closer to the target location in the future;
2. `uncertain_wong`: the entropy[2] of the navigation action distribution is greater than or equal to a threshold $\gamma$, and the highest-probability predicted navigation action is *not* suggested by the navigation teacher;
3. `never_asked`: the agent previously never requested help at the current location;

If condition (1) or (2), and condition (3) are satisfied, we set $a^{\text{ask}\star} = \texttt{request\_help}$; otherwise, $a^{\text{ask}\star} = \texttt{do\_nothing}$.

**Curiosity-Encouraging Navigation Teacher.** In addition to a reference action, the navigation teacher returns $A^{\text{nav}\otimes}$, the set of all non-reference actions that the agent took at the current location while executing the same language instruction:

$$A^{\text{nav}\otimes}_t = \{a \in A^{\text{nav}} : \exists t' < t,\ v_t = v_{t'},\ l_t = l_{t'},\ a = a^{\text{nav}}_{t'} \neq a^{\text{nav}\star}_{t'}\} \tag{8}$$

where $A^{\text{nav}}$ is the navigation action space.

We devise a *curiosity-encouraging* loss $\mathcal{L}_{\text{curious}}$, which minimizes the log likelihoods of actions in $A^{\text{nav}\otimes}$. This loss prevents the agent from repeating past mistakes and motivates it to explore untried actions. The navigation loss is:

$$\mathcal{L}_{\text{nav}}(s,\hat{\pi}_{\text{nav}},\pi^\star_{\text{nav}}) = \overbrace{-\log\hat{\pi}_{\text{nav}}(a^{\text{nav}\star} \mid o_s)}^{\mathcal{L}_{\text{NL}}(s,\hat{\pi}_{\text{nav}},\pi^\star_{\text{nav}})} + \underbrace{\alpha\frac{1}{|A^{\text{nav}\otimes}|}\sum_{a\in A^{\text{nav}\otimes}}\log\hat{\pi}_{\text{nav}}(a \mid o_s)}_{\mathcal{L}_{\text{curious}}(s,\hat{\pi}_{\text{nav}},\pi^\star_{\text{nav}})} \tag{9}$$

where $\alpha \in [0,\infty)$ is a weight hyperparameter.

**Retrospective Interpretable Help-Request Teacher.** In deciding whether the agent should ask for help, the help-request teacher must consider the agent's future situations. Standard imitation learning algorithms (e.g., DAgger) employ an *online* mode of interaction which queries the teacher at every time step. This mode of interaction is not suitable for our problem: the teacher must be able to predict the agent's future actions if it is queried when the episode is not finished. To overcome this challenge, we introduce a more efficient *retrospective* mode of interaction, which waits until the agent completes an episode and queries the teacher for reference actions for *all* time steps at once. With this approach, because the future actions at each time step are now fully observed, they can be taken into consideration when computing the reference action. In fact, we prove that the retrospective teacher is optimal for teaching the agent to determine the `lost` condition, which is the only condition that requires knowing the agent's future.

**Lemma 2.** *(proof in Appendix B) At any time step, the retrospective help-request teacher suggests the*

[2]Precisely, we use *efficiency*, or entropy of base $|A^{\text{nav}}| = 37$, where $A^{\text{nav}}$ is the navigation action space.

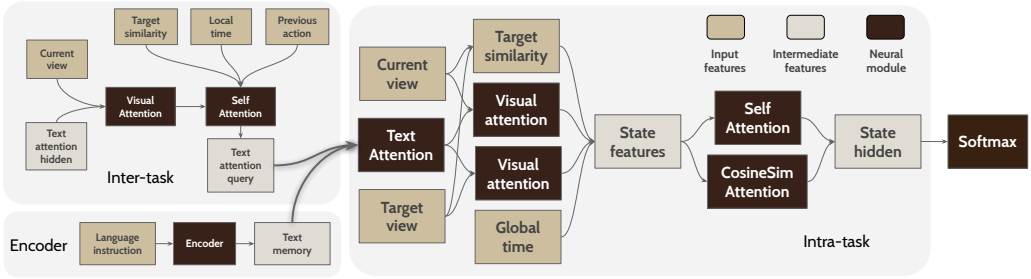


Figure 2: Our hierarchical recurrent model architecture (the navigation network). The help-request network is mostly similar except that the navigation action distribution is fed as an input to compute the "state features".

*action that results in the agent getting closer to the target location in the future under its current navigation policy (if such an action exists).*

To help the agent better justify its help-request decisions, we train a *reason classifier* $\Phi$ to predict which conditions are satisfied. To train this classifier, the teacher provides a reason vector $\rho^\star \in \{0,1\}^3$, where $\rho^\star_i = 1$ indicates that the $i$-th condition is met. We formulate this prediction problem as multi-label binary classification and employ a binary logistic loss for each condition. Learning to predict the conditions helps the agent make more accurate and interpretable decisions. The help-request loss is:

$$\mathcal{L}_{\text{ask}}(s, \hat{\pi}_{\text{ask}}, \pi^\star_{\text{ask}}) = \overbrace{-\log \hat{\pi}_{\text{ask}}(a^{\text{ask}\star} \mid o_s)}^{\mathcal{L}^{\text{NL}}(s,\hat{\pi}_{\text{ask}},\pi^\star_{\text{ask}})} \quad (10)$$
$$\underbrace{-\frac{1}{3}\sum_{i=1}^{3}[\rho^\star_i \log \hat{\rho}_i + (1-\rho^\star_i)\log(1-\hat{\rho}_i)]}_{\mathcal{L}_{\text{reason}}(s,\hat{\pi}_{\text{ask}},\pi^\star_{\text{ask}})}$$

where $(a^{\text{ask}\star}, \rho^\star) = \pi^\star_{\text{ask}}(s)$, and $\hat{\rho} = \Phi(o_s)$ is the agent-estimated likelihoods of the conditions.

## 5 Hierarchical Recurrent Architecture

We model the navigation policy and the help-request policy as two separate neural networks. The two networks have similar architectures, which consists of three main components: the *text-encoding* component, the *inter-task* component, and the *intra-task* component (Figure 2). We use self-attention instead of recurrent neural networks to better capture long-term dependency, and develop novel cosine-similarity attention and ResNet-based time-encoding. Detail on the computations in each module is in the Appendix.

The text-encoding component computes a text memory $M^{\text{text}}$, which stores the hidden representation of the current language instruction. The inter-task module computes a vector $h^{\text{inter}}_t$ representing the state of the current task's execution. During the episode, every time the current task is altered (due to the agent requesting help or departing a route), the agent re-encodes the new language instruction to generate a new text memory and resets the inter-task state to a zero vector. The intra-task module computes a vector $h^{\text{intra}}_t$ representing the state of the entire episode. To compute this state, we first calculate $\bar{h}^{\text{intra}}_t$, a tentative current state, and $\tilde{h}^{\text{intra}}_t$, a weighted combination of the past states at nearly identical situations. $h^{\text{intra}}_t$ is computed as:

$$h^{\text{intra}}_t = \bar{h}^{\text{intra}}_t - \beta \cdot \tilde{h}^{\text{intra}}_t \quad (11)$$
$$\beta = \sigma(W_{\text{gate}} \cdot [\bar{h}^{\text{intra}}_t; \tilde{h}^{\text{intra}}_t]) \quad (12)$$

Eq 11 creates an context-sensitive dissimilarity between the current state and the past states at nearly identical situations. The scale vector $\beta$ determines how large the dissimilarity is based on the inputs. This formulation incorporates past related information into the current state, thus enables the agent to optimize the curiosity-encouraging loss effectively. Finally, $h^{\text{intra}}_t$ is passed through a softmax layer to produce an action distribution.

| Split | Environments | Tasks | ANNA Instructions |
|---|---|---|---|
| Train | 51 | 82,484 | 8,586 |
| Val SeenEnv | 51 | 5,001 | 4,287 |
| Val UnseenAll | 7 | 5,017 | 2,103 |
| Test SeenEnv | 51 | 5,004 | 4,287 |
| Test Unseen | 10 | 5,012 | 2,331 |

Table 2: Data split.

## 6 Experimental Setup

**Dataset.** We generate a dataset of object-finding tasks in the HANNA environments to train and evaluate our agent. Table 2 summarizes the dataset split. Our dataset features 289 object types; the

| Agent | SEENENV | | | | UNSEENALL | | | |
|---|---|---|---|---|---|---|---|---|
| | SR ↑ (%) | SPL ↑ (%) | Nav. ↓ Err. (m) | Requests/ task ↓ | SR ↑ (%) | SPL ↑ (%) | Nav. ↓ Err. (m) | Requests/ task ↓ |
| **Non-learning agents** | | | | | | | | |
| RANDOMWALK | 0.54 | 0.33 | 15.38 | 0.0 | 0.46 | 0.23 | 15.34 | 0.0 |
| FORWARD10 | 5.98 | 4.19 | 14.61 | 0.0 | 6.36 | 4.78 | 13.81 | 0.0 |
| **Learning agents** | | | | | | | | |
| No assistance | 17.21 | 13.76 | 11.48 | 0.0 | 8.10 | 4.23 | 13.22 | 0.0 |
| Learn to interpret assistance (ours) | **88.37** | **63.92** | **1.33** | **2.9** | **47.45** | **25.50** | **7.67** | **5.8** |
| **Skylines** | | | | | | | | |
| SHORTEST | 100.00 | 100.00 | 0.00 | 0.0 | 100.00 | 100.00 | 0.00 | 0.0 |
| Perfect assistance interpretation | 90.99 | 68.87 | 0.91 | 2.5 | 83.56 | 56.88 | 1.83 | 3.2 |

Table 3: Results on test splits. The agent with "perfect assistance interpretation" uses the teacher navigation policy ($\pi^{\star}_{\text{nav}}$) to make decisions when executing a subtask from ANNA. Results of our final system are in bold.

language instruction vocabulary contains 2,332 words. The numbers of locations on the shortest paths to the requested objects are restricted to be between 5 and 15. With an average edge length of 2.25 meters, the agent has to travel about 9 to 32 meters to reach its goals. We evaluate the agent in environments that are seen during training (SEENENV), and in environments that are not seen (UNSEENALL). Even in the case of SEENENV, the tasks and the ANNA language instructions given during evaluation were never given in the same environments during training.

**Hyperparameters.** See Appendix.

**Baselines and Skylines.** We compare our agent against the following *non-learning* agents: 1. SHORTEST: uses the navigation teacher policy to make decisions (this is a skyline); 2. RANDOMWALK: randomly chooses a navigation action at every time step; 3. FORWARD10: navigates to the next location closest to the center of the current view to advance for 10 time steps. We compare our learned help-request policy with the following *heuristics*: 1. NOASK: does not request help; 2. RANDOMASK: randomly chooses to request help with a probabilty of 0.2, which is the average help-request ratio of our learned agent; 3. ASKEVERY5: requests help as soon as walking at least 5 time steps.

**Evaluation metrics.** Our main metrics are: *success rate* (SR), the fraction of examples on which the agent successfully solves the task; *navigation error*, the average (shortest-path) distance between the agent's final location and the nearest goal from that location; and *SPL* (Anderson et al., 2018a), which weights task success rate by travel distance as follows:

$$\text{SPL} = \frac{1}{N}\sum_{i=1}^{N} S_i \frac{L_i}{\max(P_i, L_i)} \qquad (13)$$

where $N$ is the number of tasks, $S_i$ indicates whether task $i$ is successful, $P_i$ is the agent's travel distance, and $L_i$ is the shortest-path distance to the goal nearest to the agent's final location.

## 7 Results

**Main results.** From Table 3, we see that our problem is challenging: simple heuristic-based baselines such as RANDOMWALK and FORWARD10 attain success rates less than 7%. An agent that learns to accomplish tasks without additional assistance from ANNA succeeds only 17.21% of the time on TEST SEENENV, and 8.10% on TEST UNSEENALL. Leveraging help from ANNA dramatically boosts the success rate by 71.16% on TEST SEENENV and by 39.35% on TEST UNSEENALL over not requesting help. Given the small size of our dataset (e.g., the agent has fewer than 9,000 subtask instructions to learn from), it is encouraging that our agent is successful in nearly half of its tasks. On average, the agent takes paths that are 1.38 and 1.86 times longer than the optimal paths on TEST SEENENV and TEST UNSEENALL, respectively. In unseen environments, it issues on average twice as many requests to as it does in seen environments. To understand how well the agent interprets the ANNA instructions, we also provide results where our agent uses the optimal navigation policy to make decisions while executing subtasks. The large gaps on TEST SEENENV indicate there is still much room for improvement in the future, purely in learning to exe-

| Assistance type | SEENENV | UNSEENALL |
|---|---|---|
| Target image only | 84.95 | 31.88 |
| + Language instruction | **88.37** | **47.45** |

Table 4: Success rates (%) of agents on test splits with different types of assistance.

| | SEENENV | | UNSEENALL | |
|---|---|---|---|---|
| $\hat{\pi}_{\text{ask}}$ | SR ↑ (%) | Requests/ task ↓ | SR ↑ (%) | Requests/ task ↓ |
| NOASK | 17.21 | 0.0 | 8.10 | 0.0 |
| RANDOMASK | 82.71 | 4.3 | 37.05 | 6.8 |
| ASKEVERY5 | 87.39 | 3.4 | 34.42 | 7.1 |
| Learned (ours) | **88.37** | **2.9** | **47.45** | **5.8** |

Table 5: Success rates (%) of different help-request policies on test splits.

cute language instructions.

**Does understanding language improve generalizability?** Our agent is assisted with both language and visual instructions; similar to Thomason et al. (2019a), we disentangle the usefulness two these two modes of assistance. As seen in Table 4, the improvement from language on TEST UNSEENALL (+15.17%) is substantially more than that on TEST SEENENV (+3.42%), largely the agent can simply memorize the seen environments. This confirms that understanding language-based assistance effectively enhances the agent's capability of accomplishing tasks in novel environments.

**Is learning to request help effective?** Table 5 compares our learned help-request policies with baselines. We find that ASKEVERY5 provides a surprisingly strong baseline for this problem, leading to an improvement of +26.32% over not requesting help on TEST UNSEENALL. Nevertheless, our learned policy, with the ability to predict the future and access to the agent's uncertainty, outperforms all baselines by at least 10.40% in success rate on TEST UNSEENALL, while making less help requests. The small gap between the learned policy and ASKEVERY5 on TEST UNSEENALL is expected because, on this split, the performance is mostly determined by the model's memorizing capability and is mostly insensitive to the help-request strategy.

**Is proposed model architecture effective?** We implement an LSTM-based encoder-decoder model that is based on the architecture proposed by (Wang et al., 2019). To incorporate the target image, we add an attention layer that uses the image's vector set as the attention memory. We train this model with imitation learning using the standard negative log likelihood loss (Eq 7), without the curiosity-encouraging and reason-prediction losses. As seen in Table 6, our hierarchical recurrent model outperforms this model by a large margin on TEST UNSEENALL (+28.2%).

| Model | SR ↑ (%) | Nav. mistake ↓ repeat (%) | Help-request ↓ repeat (%) |
|---|---|---|---|
| LSTM-ENCDEC | 19.25 | 31.09 | 49.37 |
| Our model ($\alpha = 0$) | 43.12 | 25.00 | 40.17 |
| Our model ($\alpha = 1$) | **47.45** | **17.85** | **21.10** |

Table 6: Results on TEST UNSEENALL of our model, trained with and without curiosity-encouraging loss, and an LSTM-based encoder-decoder model (both models have about 15M parameters). "Navigation mistake repeat" is the fraction of time steps on which the agent repeats a non-optimal navigation action at a previously visited location while executing the same task. "Help-request repeat" is the fraction of help requests made at a previously visited location while executing the same task.

**Does the proposed imitation learning algorithm achieve its goals?** The curiosity-encouraging training objective is proposed to prevent the agent from making the same mistakes at previously encountered situations. Table 6 shows that training with the curiosity-encouraging objective reduces the chance of the agent looping and making the same decisions repeatedly. As a result, its success rate is greatly boosted (+4.33% on TEST UNSEENALL) over no curiosity-encouraging.

## 8 Conclusion

In this work, we present a photo-realistic simulator that mimics primary characteristics of real-life human assistance. We develop effective imitation learning techniques for learning to request and interpret the simulated assistance, coupled with a hierarchical neural network model for representing subtasks. Future work aims to provide more natural, linguistically realistic interaction between the agent and humans (e.g., providing the agent the ability ask a natural *question* rather than just signal for help), and to establish a theoretical framework for modeling human assistance. We are also exploring ways to deploy and evaluate our methods on real-world platforms.

## References


Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, et al. 2018a. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*.

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. 2018b. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3674–3683.

Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. 2018. Learning dexterous in-hand manipulation. *arXiv preprint arXiv:1808.00177*.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. Openai gym. *arXiv preprint arXiv:1606.01540*.

Joyce Y Chai, Rui Fang, Changsong Liu, and Lanbo She. 2016. Collaborative language grounding toward situated human-robot dialogue. *AI Magazine*, 37(4):32–45.

Joyce Y Chai, Qiaozi Gao, Lanbo She, Shaohua Yang, Sari Saba-Sadiya, and Guangyue Xu. 2018. Language to action: Towards interactive task learning with physical agents. In *International Joint Conference on Artificial Intelligence*.

Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. 2017. Matterport3D: Learning from RGB-D data in indoor environments. *International Conference on 3D Vision (3DV)*.

Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daume III, and John Langford. 2015. Learning to search better than your teacher. In *Proceedings of the International Conference of Machine Learning*.

Howard Chen, Alane Shur, Dipendra Misra, Noah Snavely, Ian Artzi, Yoav, Stephen Gould, and Anton van den Hengel. 2019. Touchdown: Natural language navigation and spatial reasoning in visual street environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. 2019. Babyai: A platform to study the sample efficiency of grounded language learning. In *Proceedings of the International Conference on Learning Representations*.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. 2018. Textworld: A learning environment for text-based games. In *Computer Games Workshop at ICML/IJCAI*.

Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. 2018. Embodied question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. In *Machine learning*, volume 75, pages 297–325. Springer.

Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. 2018. Speaker-follower models for vision-and-language navigation. In *Proceedings of Advances in Neural Information Processing Systems*.

He He, Jason Eisner, and Hal Daumé III. 2012. Imitation learning by coaching. In *Proceedings of Advances in Neural Information Processing Systems*, pages 3149–3157.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. 2018. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. In *Proceedings of the Conference on Robot Learning*.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. 2017. Progressive growing of gans for improved quality, stability, and variation. In *Proceedings of the International Conference on Learning Representations*.

Tero Karras, Samuli Laine, and Timo Aila. 2018. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Janne Karttunen, Anssi Kanervisto, Ville Hautamäki, and Ville Kyrki. 2019. From video game to real robot: The transfer between action spaces. *arXiv preprint arXiv:1905.00741*.

Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. 2016. Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *Computational Intelligence and Games (CIG), 2016 IEEE Conference on*, pages 1–8. IEEE.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Changsong Liu, Shaohua Yang, Sari Saba-Sadiya, Nishant Shukla, Yunzhong He, Song-Chun Zhu, and Joyce Chai. 2016. Jointly learning grounded task structures from language instruction and visual demonstration. In *Proceedings of Emperical Methods in Natural Language Processing*, pages 1482–1492.

Chih-Yao Ma, Jiasen Lu, Zuxuan Wu, Ghassan AlRegib, Zsolt Kira, Richard Socher, and Caiming Xiong. 2019a. Self-monitoring navigation agent via auxiliary progress estimation. In *Proceedings of the International Conference on Learning Representations*.

Chih-Yao Ma, Zuxuan Wu, Ghassan AlRegib, Caiming Xiong, and Zsolt Kira. 2019b. The regretful agent: Heuristic-aided navigation through progress estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6732–6740.

Dipendra Misra, Andrew Bennett, Valts Blukis, Eyvind Niklasson, Max Shatkhin, and Yoav Artzi. 2018. Mapping instructions to actions in 3d environments with visual goal prediction. In *Proceedings of Emperical Methods in Natural Language Processing*, pages 2667–2678. Association for Computational Linguistics.

Dipendra Misra, John Langford, and Yoav Artzi. 2017. Mapping instructions and visual observations to actions with reinforcement learning. *Proceedings of Emperical Methods in Natural Language Processing*.

Dipendra K Misra, Jaeyong Sung, Kevin Lee, and Ashutosh Saxena. 2014. Tell me dave: Contextsensitive grounding of natural language to mobile manipulation instructions. In *Robotics: Science and Systems*.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. In *NIPS Deep Learning Workshop*.

Shiwali Mohan and John Laird. 2014. Learning goal-oriented hierarchical tasks from situated interactive instruction. In *Association for the Advancement of Artificial Intelligence*.

Khanh Nguyen, Debadeepta Dey, Chris Brockett, and Bill Dolan. 2019. Vision-based navigation with language-based assistance via imitation learning with indirect intervention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12527–12537.

Yuankai Qi, Qi Wu, Peter Anderson, Marco Liu, Chunhua Shen, and Anton van den Hengel. 2019. Rerere: Remote embodied referring expressions in real indoor environments. *arXiv preprint arXiv:1904.10151*.

Stephane Ross and J Andrew Bagnell. 2014. Reinforcement and imitation learning via interactive no-regret learning. *arXiv preprint arXiv:1406.5979*.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of Artificial Intelligence and Statistics*, pages 627–635.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252.

Paul E Rybski, Kevin Yoon, Jeremy Stolarz, and Manuela M Veloso. 2007. Interactive robot task training through dialog and demonstration. In *Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pages 49–56. ACM.

Amr Sharaf and Hal Daumé III. 2017. Structured prediction via learning to search under bandit feedback. In *Proceedings of the 2nd Workshop on Structured Prediction for Natural Language Processing*, pages 17–26.

Lanbo She, Shaohua Yang, Yu Cheng, Yunyi Jia, Joyce Chai, and Ning Xi. 2014. Back to the blocks world: Learning new actions through situated human-robot dialogue. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 89–97.

Wen Sun, Arun Venkatraman, Geoffrey J Gordon, Byron Boots, and J Andrew Bagnell. 2017. Deeply aggrevated: Differentiable imitation learning for sequential prediction. In *Proceedings of the International Conference of Machine Learning*.

Jesse Thomason, Daniel Gordan, and Yonatan Bisk. 2019a. Shifting the baseline: Single modality performance on visual navigation & qa. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Jesse Thomason, Michael Murray, Maya Cakmak, and Luke Zettlemoyer. 2019b. Vision-and-dialog navigation. In *Proceedings of the Conference on Robot Learning*.

Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE.

Jack Urbanek, Angela Fan, Siddharth Karamcheti, Saachi Jain, Samuel Humeau, Emily Dinan, Tim Rocktäschel, Douwe Kiela, Arthur Szlam, and Jason Weston. 2019. Learning to speak and act in a fantasy text adventure game. *arXiv preprint arXiv:1903.03094*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, et al. 2017. Starcraft ii: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*.

Harm de Vries, Kurt Shuster, Dhruv Batra, Devi Parikh, Jason Weston, and Douwe Kiela. 2018. Talk the walk: Navigating new york city through grounded dialogue. *arXiv preprint arXiv:1807.03367*.

Xin Wang, Qiuyuan Huang, Asli Celikyilmaz, Jianfeng Gao, Dinghan Shen, Yuan-Fang Wang, William Yang Wang, and Lei Zhang. 2019. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Xin Wang, Wenhan Xiong, Hongmin Wang, and William Yang Wang. 2018. Look before you leap: Bridging model-free and model-based reinforcement learning for planned-ahead vision-and-language navigation. In *Proceedings of the European Conference on Computer Vision*.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2019. Neural text generation with unlikelihood training. *arXiv preprint arXiv:1908.04319*.

Erik Wijmans, Samyak Datta, Oleksandr Maksymets, Abhishek Das, Georgia Gkioxari, Stefan Lee, Irfan Essa, Devi Parikh, and Dhruv Batra. 2019. Embodied question answering in photorealistic environments with point cloud perception. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6659–6668.

# Incorporating Visual Semantics into Sentence Representations within a Grounded Space

**Patrick Bordes**[*1] and **Éloi Zablocki**[*1] and **Laure Soulier**[1] and **Benjamin Piwowarski**[1] and **Patrick Gallinari**[1,2]

[1] Sorbonne Université, CNRS, LIP6, F-75005 Paris, France
[2] Criteo AI Lab, Paris, France
firstname.name@lip6.fr

## Abstract

*Language grounding* is an active field aiming at enriching textual representations with visual information. Generally, textual and visual elements are embedded in the same representation space, which implicitly assumes a one-to-one correspondence between modalities. This hypothesis does not hold when representing words, and becomes problematic when used to learn sentence representations — the focus of this paper — as a visual scene can be described by a wide variety of sentences. To overcome this limitation, we propose to transfer visual information to textual representations by learning an intermediate representation space: the grounded space. We further propose two new complementary objectives ensuring that (1) sentences associated with the same visual content are close in the grounded space and (2) similarities between related elements are preserved across modalities. We show that this model outperforms the previous state-of-the-art on classification and semantic relatedness tasks.

## 1 Introduction

Representing text by vectors that capture meaningful semantics is a long-standing issue in Artificial Intelligence. Distributional Semantic Models (Mikolov et al., 2013; Peters et al., 2018) are well-known recent efforts in this direction, based on the *distributional hypothesis* (Harris, 1954). They rely on large text corpora to learn word embeddings. At another granularity level, having high-quality and general-purpose sentence representations is crucial for all models that encode sentences into semantic vectors, such as the ones used in machine translation (Bahdanau et al., 2014) or relation extraction (Wang et al., 2019). Moreover, encoding semantics of sentences is paramount because sentences describe relationships between objects, and thus convey complex and high-level knowledge better than individual words (Norman, 1972).

Relying only on text can lead to biased representations and unrealistic predictions such as "*the sky is green*" (Baroni, 2016). Besides, it has been shown that human understanding of language is *grounded* in physical reality and perceptual experience (Fincher-Kiefer, 2001). To overcome this limitation, an emerging approach is to *ground* language in the visual world: this consists in leveraging visual information, usually from images, to enrich textual representations.[*]

Leveraging images resulted in improved linguistic representations on intrinsic and downstream tasks (Bruni et al., 2014; Silberer and Lapata, 2014). In most of these approaches, cross-modal projections are learned to incorporate visual semantics in the final representations (Lazaridou et al., 2015b; Collell et al., 2017; Kiela et al., 2018). These works rely on paired textual and visual data and the hypothesis of a one-to-one correspondence between modalities is implicitly assumed: an image of an object univocally represents a word. However, there is no obvious reason implying that the structure of the two spaces should match. Indeed, Collell and Moens (2018) empirically show that cross-modal projection of a source modality does not resemble the target modality in terms of its neighborhood structure. This is especially the case for sentences, where many different sentences can describe a similar image. Therefore, we argue that learning grounded representations with projections to a visual space is particularly inadequate in the case of sentences.

To overcome this issue, we propose an alter-

[*] Equal contribution.

[*] In the Computer Vision community, *grounding* can also refer to the task of linking phrases with image regions (Xiao et al., 2017), but this is not the focus of the present paper.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 696–707, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

native approach where the structure of the visual space is *partially transferred* to the textual space. This is done by distinguishing two types of complementary information sources. First, the *cluster information*: the implicit knowledge that sentences associated with the same image refer to the same underlying reality. Second, the *perceptual information*, which is contained within high-level representations of images. These two sources of information aim at transferring the structure of the visual space to the textual space. Besides, to preserve textual semantics and to avoid an over-constrained textual space, we propose to incorporate the visual information to textual representations using an intermediate representation space that we call *grounded space*, on which cluster and perceptual objectives are trained.

Our contributions are the following: (1) we define two complementary objectives to ground the textual space, based on implicit and explicit visual information; (2) we propose to incorporate visual semantics through the means of an intermediate space, within which the objectives are learned. Moreover (3) we perform quantitative and qualitative evaluations on several transfer tasks, showing the advantages of our approach with respect to previous grounding methods.

## 2 Related work

Over the last years, several approaches have been proposed to learn semantic representations for sentences. This includes supervised and task-specific techniques with recursive networks (Socher et al., 2013), convolutional networks (Kalchbrenner et al., 2014) or self-attentive networks (Lin et al., 2017; Conneau et al., 2017), but also unsupervised methods producing universal representations given large text corpora. Examples of the latter include models such as FastSent (Hill et al., 2016), QuickThought (Logeswaran and Lee, 2018), Word Information Series (Arroyo-Fernández et al., 2019), Universal Sentence Encoder (Cer et al., 2018), or SkipThought (Kiros et al., 2015), where a sentence is encoded with a Gated Recurrent Unit (GRU), and two GRU decoders are trained to reconstruct the adjacent sentences in a dataset of ordered sentences.

To model the way language conveys meaning, traditional approaches consider language as a purely symbolic system based on words and syntactic rules (Chomsky, 1980; Burgess and Lund, 1997). However, W. Barsalou (1999); Fincher-Kiefer (2001) insist on the intuition that language has to be grounded in physical reality and perceptual experience. The importance of language grounding is underlined in Gordon and Van Durme (2013), where an important bias is reported: the frequency at which objects, relations, or events occur in natural language is significantly different from their real-world frequency (e.g., in texts, people are *murdered* four times more than they *breathe*). Thus, leveraging visual resources, in addition to textual resources, is a promising way to acquire commonsense knowledge (Lin and Parikh, 2015; Yatskar et al., 2016) and to cope with the bias between text and reality.

This intuition has motivated several works for learning visually grounded representations for words, using images — or abstract scenes (Kottur et al., 2016). Two lines of work can be distinguished. First, *sequential* techniques combine textual and visual representations that were learned separately (Bruni et al., 2014; Silberer and Lapata, 2014; Collell et al., 2017). Second, *joint* methods learn a multimodal representation from multiple sources simultaneously. The advantage is that visual information associated with concrete words can be transferred to more abstract ones, which usually have no associated visual data (Hill et al., 2014; Lazaridou et al., 2015b). Closer to our contribution, some approaches learn grounded word embeddings by building upon the skip-gram objective (Mikolov et al., 2013) and enforcing word vectors to be close to their corresponding visual features (Lazaridou et al., 2015b) or their visual context (Zablocki et al., 2018) in a multimodal representation space. These approaches learn word representations while we specifically target sentences. This task is more challenging since sentences are inherently different than words due to their sequential and compositional nature. Moreover, a great number of different sentences can be generated for the same image.

Finally, some works learn sentence representations by aligning visual data to sentences within captioning datasets. Chrupala et al. (2015) propose the `IMAGINET` model where two sentence encoders share word embeddings: a first GRU encoder learns a language model objective and the other one is trained to predict the visual features associated to a sentence. The model of Kiela et al. (2018) is close to `IMAGINET` and additionally

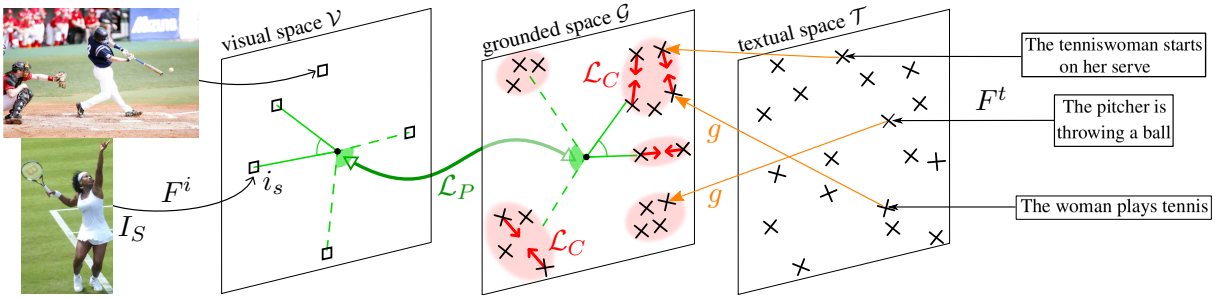


Figure 1: Model overview. Red circles indicate visual clusters. Red arrows represent the gradient of the cluster loss, which gathers visually equivalent sentences — the contrastive term in loss $\mathcal{L}_C$ is not represented. The green arrow and angles illustrate the perceptual loss, ensuring that cosine similarities correlate across modalities. The origin is at the center of each space.

hypothesizes that associated captions ground the meaning of a sentence. Both of these works learn a projection of the sentence representation to the corresponding image, which we argue is problematic as it over-constrains the textual space and can degrade textual representations. Indeed, Collell and Moens (2018) empirically demonstrated that when a cross-modal mapping is learned, the projection of the source modality does not resemble the target modality, in the sense of nearest neighbor comparison. This suggests that cross-modal projections are not appropriate to incorporate visual semantics in text representations.

## 3 Incorporating visual semantics within an intermediate grounded space

### 3.1 Model overview

In this work, we aim at learning grounded representations by jointly leveraging the textual and visual contexts of a sentence. We note $S$ a sentence and $s = F^t(S; \theta^t)$ its representation computed with a sentence encoder $F^t$ parametrized by $\theta^t$. We follow the classical approach developed in the language grounding literature at the word level (Lazaridou et al., 2015b; Zablocki et al., 2018), which balances a textual objective $\mathcal{L}_\mathcal{T}$ with an additional grounding objective $\mathcal{L}_\mathcal{G}$:

$$\mathcal{L}(\theta^t, \theta^i) = \mathcal{L}_\mathcal{T}(\theta^t) + \mathcal{L}_\mathcal{G}(\theta^t, \theta^i) \qquad (1)$$

The parameters $\theta^t$ of the sentence encoder $F^t$ are shared in $\mathcal{L}_\mathcal{T}$ and $\mathcal{L}_\mathcal{G}$, and therefore benefit from both textual and grounding objectives. $\theta^i$ denotes extra grounding parameters, including the weights of the image encoder $F^i$. Note that any textual objective $\mathcal{L}_\mathcal{T}$ and sentence encoder $F^t$ can be used. In our experiments, we choose the well-known SkipThought model (Kiros et al., 2015), trained on a corpus of ordered sentences.

In what follows, we focus on the modeling of the grounding objective $\mathcal{L}_\mathcal{G}$, learned on a captioning corpus, where each image is associated with several captions. Grounding approaches generally leverage visual information by embedding textual and visual elements within the same multimodal space (Silberer and Lapata, 2014; Kiela et al., 2018). However, it is not satisfying since texts and images are forced to be in one-to-one correspondence. Moreover, a caption can (1) have a wide variety of paraphrases and related sentences describing the same scene (e.g., *the kitten is devouring a mouse* versus *a cat eating a mouse*), (2) be visually ambiguous (e.g., *a cat is eating* can be associated with many different images, depending on the visual scene/context), or (3) carry non-visual information (e.g., *cats often think about their meals*). Usual grounding objectives, that embed sentences in the visual space, can discard non-visual information (3) through the projection function. They can handle (1) by projecting related sentences to the same location in the visual space. However, they are over-sensitive to visual ambiguity (2), because ambiguous sentences should be projected to different locations of the visual space, which is not possible with current grounding models.

To overcome this lack of flexibility, we propose the following approach, illustrated in Figure 1. To cope with (1), sentences associated with the same image should be close — we call this *cluster information*. To cope with (2), we want to avoid projecting sentences to a particular point of the visual space: instead, we require that the similarity between two images in the visual space (which is linked to the context discrepancy) should be close to the similarity between their associated sentences in the textual space. We call this *perceptual information*. Finally, as we want to preserve non-visual information in sentence representations (3), we make use of an intermediate space,

called *grounded space*, that allows textual representations to benefit from visual properties without degrading the semantics brought by the textual objective $\mathcal{L}_\mathcal{T}$.

### 3.2 Grounding space and objectives

In this section, we introduce more formally the grounded space and the different information (cluster and perceptual) captured in the grounding loss $\mathcal{L}_\mathcal{G}$.

**Grounded space** The grounded space relaxes the assumption that textual and visual representations should be guided by one-to-one correspondences. It rather assumes that the structure of the textual space might be partially modeled on the structure of the visual space. Thus, instead of directly applying the grounding objectives on a sentence $s$ embedding, we propose to train the grounding objective $\mathcal{L}_\mathcal{G}$ on an intermediate space called *grounded space*. Practically, we use a projection $g(s;\theta^i_g)$ of a sentence $s$ from the textual space to the grounded space. We denote it $g(s)$ for simplicity, where $g$ is a multi-layer perceptron with input $s = F^t(S;\theta^t)$ and parameters $\theta^i_g$ ($\theta^i_g \subset \theta^i$).

**Cluster information ($\mathbf{C}_g$)** The cluster information leverages the fact that two sentences describe, or not, the same underlying reality. In other words, the goal is to measure if two sentences are *visually equivalent* (assumption (1) in Section 3.1) without considering the content of related images. For convenience, two sentences are said to be *visually equivalent* (resp. *visually different*) if they are associated with the same image (resp. different images), i.e. if they describe the same (resp. different) underlying reality. We call *cluster* a set of visually equivalent sentences. For instance, in Figure 1, sentences *The tenniswoman starts on her serve* and *The woman plays tennis* are visually equivalent and belong to the same cluster.

Our hypothesis is that *the similarity between visually equivalent sentences* $(s, s^+)$ *should be higher than visually different sentences* $(s, s^-)$. We translate this hypothesis into the constraint in the grounded space: $\cos(g(s), g(s^+)) \leq \cos(g(s), g(s^-))$. Following (Karpathy and Li, 2015; Carvalho et al., 2018), we use a max-margin ranking loss to ensure the gap between both terms is higher than a fixed margin $\gamma$ (cf. red elements in Figure 1) resulting in the cluster loss $\mathcal{L}_\mathcal{C}$:

$$\mathcal{L}_C = \sum_{(s,s^+,s^-)} \left[\gamma - \cos(g(s), g(s^+)) + \cos(g(s), g(s^-))\right]_+ \quad (2)$$

where $s^+$ (resp. $s^-$) is a randomly sampled visually equivalent (resp. different) sentence to $s$. This loss function is also used in the cross-modal retrieval literature to enforce structure-preserving constraints between sentences describing a same image (Wang et al., 2016).

**Perceptual information ($\mathbf{P}_g$)** The cluster hypothesis alone ignores the structure of the visual space and only uses the visual modality as a proxy to assess if two sentences are visually equivalent or different. Moreover, the ranking loss $\mathcal{L}_\mathcal{C}$ simply drives apart visually different sentences in the representation space, which can be a problem when two images have a closely related content. For instance, the baseball and tennis images in Figure 1 may be different, but they are both sports images, and thus their corresponding sentences should be somehow close in the grounded space. Finally, it supposes that we have a dataset of images associated with several captions.

To cope with these limitations, we consider the structure of the visual space and use the content of images. The intuition is that the structure of the textual space should be modeled on the structure of the visual one to extract visual semantics. We choose to preserve *similarities* between related elements across spaces (cf. green elements in Figure 1). We thus assume that *the similarity between two sentences in the grounded space should be correlated with the similarity between their corresponding images in the visual space.* We translate this hypothesis into the perceptual loss $\mathcal{L}_\mathcal{P}$:

$$\mathcal{L}_\mathcal{P} = -\rho(\{sim^{\text{text}}_{k_1,k_2}\}, \{sim^{\text{im}}_{k_1,k_2}\}) \quad (3)$$

where $\rho$ is the Pearson correlation, $sim^{\text{text}}_{k_1,k_2} = \cos(g(s_{k_1}), g(s_{k_2}))$ and $sim^{\text{im}}_{k_1,k_2} = \cos(i_{k_1}, i_{k_2})$ are respectively textual and visual similarities computed over several randomly sampled pairs of matching sentences and images.

**Grounded loss** Taking altogether, the grounded space and cluster/perceptual information leads to the grounding objective $\mathcal{L}_\mathcal{G}(\theta^t, \theta^i)$ as a linear combination of the aforementioned objectives:

$$\mathcal{L}_\mathcal{G}(\theta^t, \theta^i) = \alpha_C \mathcal{L}_C(\theta^t, \theta^i) + \alpha_P \mathcal{L}_P(\theta^t, \theta^i) \quad (4)$$

where $\alpha_C$ and $\alpha_P$ are hyper-parameters weighting contributions of $\mathcal{L}_C$ and $\mathcal{L}_P$. $\theta^i$ corresponds to all the grounding-related parameters, i.e. those of the image encoder $F^i$ and of the projection function $g$ (i.e., $\theta^i_g$).

## 4 Evaluation protocol

### 4.1 Datasets

**Textual dataset.** Following (Kiros et al., 2015; Hill et al., 2016), we use the Toronto BookCorpus dataset as the textual corpus. This corpus consists of 11K books, and 74M ordered sentences, with an average of 13 words per sentence.

**Visual dataset.** We use the MS COCO (Lin et al., 2014) dataset as the visual corpus. This image captioning dataset consists of 118K/5K/41K (train/val/test) images, each with five English descriptions. Note that the number of sentences in the training set of COCO (590K sentences) only represents $0.8\%$ of the sentence data in BookCorpus, which is negligible, and the additional textual training data cannot account for performance discrepancies between textual and grounded models.

### 4.2 Baselines and Scenarios

In the experiments, we focus on one of the most established sentence models: SkipThought (noted **T**) as the textual baseline: the parameters of the sentence embedding model are obtained by minimizing $\mathcal{L}_\mathcal{T}$. Then, we derive several baselines and scenarios based on **T**, each representing a different approach of grounding. Since our focus is to study the impact of grounding on sentence representations, all baselines and scenarios share the same representation dimension $d_t = 2048$ and are trained on the same datasets (cf. sect. 4.1). We also report a textual model of dimension $\frac{d_t}{2}$ that we call $\mathbf{T}_{1024}$, to compare with the GroundSent model of (Kiela et al., 2018).

**Model Scenarios.** We test variants of our grounding model presented in sect. 3, all based on **T**: $\mathbf{T} + \mathbf{C}_g$, $\mathbf{T} + \mathbf{P}_g$, $\mathbf{T} + \mathbf{C}_g + \mathbf{P}_g$, where $\mathbf{C}_g$ (resp. $\mathbf{P}_g$) represents the loss $\mathcal{L}_C$ (resp. $\mathcal{L}_P$). We also consider scenarios where $g$ equals the identity function (no grounded space), which we note $\mathbf{C}_{id}$, $\mathbf{P}_{id}$, $\mathbf{C}_{id} + \mathbf{P}_{id}$, etc. Finally, we also performed preliminary analysis learning only from the visual modality: $\mathbf{C}_{g/id}$, $\mathbf{P}_{g/id}$, $\mathbf{C}_{g/id} + \mathbf{P}_{g/id}$.

**Baselines.** We adapt two classical multimodal word embedding models for sentences. Accordingly, models from the two existing model families are considered:

*Cross-modal Projection* (**CM**): Inspired by Lazaridou et al. (2015b), this baseline learns to project sentences in the visual space using a max-margin loss:

$$\sum_{(s,i_s,i^-)} \left\lfloor \gamma' + \cos(f(s), i^-) - \cos(f(s), i_s) \right\rfloor_+$$

where $f$ is a MLP, $\gamma'$ a fixed margin and $i^-$ a non-matching image. Similarly to our scenarios, the sentence encoder is initialized with **T**.

*Sequential* (**SEQ**): Inspired by Collell et al. (2017), we learn a linear regression model $(W, b)$ to predict the visual representation of an image, from the representation of a matching caption. The grounded word embedding is the concatenation of the original SkipThought vector **T** and its predicted ("*imagined*") representation $W\mathbf{T} + b$, which is projected using a PCA into dimension $d_t$.

In both cases, the parameters to be learned, in addition to the sentence encoder, are the cross-modal projections – and the sentence representation is obtained by averaging word vectors.

**GroundSent Model** We re-implement the GroundSent models of Kiela et al. (2018), obtaining comparable results. The authors propose two objectives to learn a grounded vector: (a) Cap2Img: the cross-modal projections of sentences are pushed towards their respective images via a max-margin ranking loss, and (b) Cap2Cap: a visually equivalent sentence is predicted via a LSTM sentence decoder. The Cap2Both objective is a combination of these two objectives. Once the grounded vectors are learned, they are concatenated with a textual vector (learned via a SkipThought objective) to form the GS-Img, GS-Cap and GS-Both vectors.

### 4.3 Evaluation tasks and metrics

In line with previous works (Kiros et al., 2015; Hill et al., 2016), we consider several benchmarks to evaluate the quality of our grounded embeddings:

**Semantic relatedness.** We use two semantic similarity benchmarks: STS (Cer et al., 2017) and SICK (Marelli et al., 2014a), which consist of

| Model | Structural measures | | | | Semantic relatedness | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | mNNO | $\rho_{vis}$ | $C_{inter}$ | $C_{intra}$ | STS/All | STS/Cap | STS/News | STS/Forum | SICK |
| **T** | 10.0 | 4.1 | 54.2 | 70.1 | 30 | 41 | 36 | 21 | 51 |
| **CM** (text) | 24.2 | 12.8 | 41.7 | 74.8 | 52 | 76 | 42 | **37** | 55 |
| $\mathbf{P}_{id}$ | 21.1 | **37.9** | 42.2 | 69.3 | 45 | 66 | 41 | 34 | 54 |
| $\mathbf{C}_{id}$ | 27.5 | 10.5 | **2.9** | **84.7** | 60 | 83 | 45 | 20 | 55 |
| $\mathbf{C}_{id} + \mathbf{P}_{id}$ | **27.9** | 25.8 | 6.7 | 82.6 | **61** | **84** | **46** | 28 | **57** |
| **CM** (vis.) | 27.1 | 19.2 | 1.5 | 85.8 | 56 | 78 | 40 | 34 | 55 |
| $\mathbf{P}_g$ | 21.3 | **32.4** | 43.9 | 73.3 | 45 | 66 | 41 | **37** | 53 |
| $\mathbf{C}_g$ | 28.6 | 9.4 | **1.1** | **88.5** | 62 | 83 | 46 | 29 | 59 |
| $\mathbf{C}_g + \mathbf{P}_g$ | **28.9** | 29.1 | 4.7 | 87.5 | **63** | **84** | **48** | 33 | **60** |

Table 1: Intrinsic evaluations carried out on the grounded space for models with $g$ = MLP; the textual space for **T**, **CM** (text) and models with $g = id$; and the visual space for **CM** (vis).

pairs of sentences that are associated with human-labeled similarity scores. STS is subdivided into three textual sources: *Captions* contain concrete sentences describing daily-life actions, whereas the others contain more abstract sentences: news headlines in *News* and posts from user forums in *Forum*. The Spearman correlations are measured between the cosine similarity of our learned sentence embeddings and human-labeled scores.

**Classification benchmarks.** All extrinsic evaluations are carried out using the SentEval pipeline (Conneau and Kiela, 2018). The tasks are the following: opinion polarity (MPQA) (Wiebe and Cardie, 2005), movie review sentiment (MR) (Pang and Lee, 2005), subjectivity/objectivity classification (SUBJ) (Pang and Lee, 2004), customer reviews (CR) (Hu and Liu, 2004), binary sentiment analysis on SST (Socher et al., 2013), paraphrase identification (MSRP) (Dolan et al., 2004) as well as two entailment classification benchmarks: SNLI (Bowman et al., 2015) and SICK (Marelli et al., 2014b). For each dataset, a logistic regression classifier is learned from the extracted sentence embeddings, and we report the classification accuracy.

**Structural measures.** To probe the learned grounded space, we define structural measures, and report their values on the validation set of MS COCO (5K images, 25K captions). First, we report the *mean Nearest Neighbor Overlap* (mNNO) metric, as defined in Collell and Moens (2018), that indicates the proportion of shared nearest neighbors between image representations and their corresponding captions in their respective spaces. To study *perceptual information*, we define $\rho_{vis}$, the Pearson correlation $\rho(\cos(s, s'), \cos(v_s, v_{s'}))$ between images and their corresponding sentences' similarities. For *cluster information*, we introduce $C_{intra} = \mathbb{E}_{v_s = v_{s'}}[cos(s, s')]$, which measures the homogeneity of each cluster, and $C_{inter} = \mathbb{E}_{v_s \neq v_{s'}}[cos(s, s')]$, which measures how well clusters are separated from each other.

### 4.4 Implementation details

Images are processed using a pretrained Inception-v3 network (Szegedy et al., 2016) ($d_i = 2048$). The model is trained with ADAM (Kingma and Ba, 2014) and a learning rate $l_r = 8.10^{-4}$. As done in Kiros et al. (2015), our sentence encoder is a GRU with a vocabulary of 20K words, represented in dimension 620; we perform vocabulary expansion at inference. All hyperparameters are tuned using the Pearson correlation measure on the validation set of the SICK benchmark: $\gamma = \gamma' = 0.5$, $\alpha_C = \alpha_P = 0.01$, $d_g = 512$; functions $f$ and $g$ are 2-layer MLP. As done in (Kiela et al., 2018), we set $d_t = 2048$. Our code is available at https://github.com/pbordes/multimodal_sentence_rep.

## 5 Experiments and Results

Our main objective is to study the contribution brought by the visual modality to the grounded sentence representations, and we do not attempt to outperform purely textual sentence encoders from the literature. We show that textual models can benefit from grounding approaches without requiring any changes to the original textual objectives $\mathcal{L}_T$. We report quantitative and qualitative insights (sect. 5.1), and quantitative results on the SentEval benchmark (sect. 5.2).

### 5.1 Study of the grounded space

We study the impact of the various grounding hypotheses on the structure of the grounded space, using intrinsic measures. In Table 1, we report the structural measures and the semantic relatedness scores of the baselines, namely **T** and **CM**, and on the various scenarios of our model. The textual loss is discarded to isolate the effect of the different grounding hypotheses.

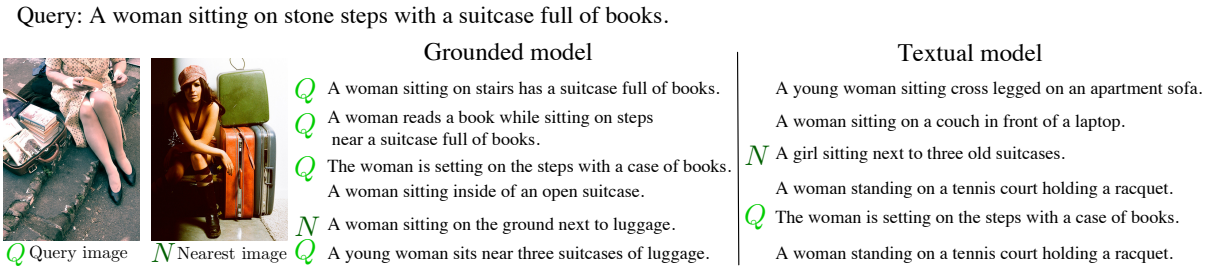


Figure 2: Nearest neighbors of a selected sentence in the validation set of MS COCO, for both grounded and purely textual models. $Q$ is the query image, $N$ is the nearest neighbor of $Q$ in the visual space. Sentences that are caption of $Q$ or $N$ are prefixed with $Q$ or $N$.

| Query | Textual model | Grounded model |
|---|---|---|
| Two people are in **love** | Two people are fencing indoors | A couple just got married and are taking a picture with family |
| A man is **horrified** | A man and a woman are smiling | A teenage boy wearing a cap looks irritated |
| This is a **tragedy** | A group of people are at a party | Men doing a war reenactment |

Table 2: Qualitative analysis: nearest neighbor of a given query (containing an abstract word) among Flickr30K sentences.

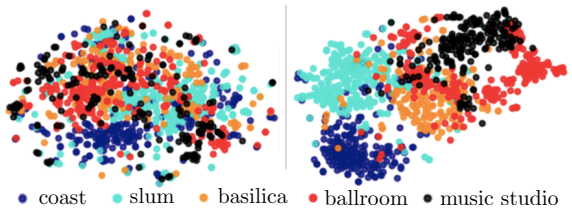


Figure 3: t-SNE visualization on CMPlaces sentences for a set of randomly sampled visual scenes. Left: textual model $\mathbf{T}$. Right: grounded model $\mathbf{C}_g + \mathbf{P}_g$.

**The impact of grounding** We investigate the effect of grounding on sentence representations. Results highlight that all grounded models improve over the baseline $\mathbf{T}$. Moreover, our model $\mathbf{C}_g + \mathbf{P}_g$ is generally the most effective regarding the mNNO measure and semantic relatedness tasks.

*Influence of concreteness* To understand in which cases grounding is useful, we compute the average visual concreteness $\bar{c}$ of the STS benchmark, which is divided in three categories (*Captions*, *News*, *Forum*). This is done by using a concreteness dataset built by Brysbaert et al. (2013) consisting of human ratings of concreteness (between 0 and 5) for 40,000 English words; for a given benchmark, we compute the sum of these scores and average over all words that are in the concreteness dataset. The performance gain $\Delta$ between $\mathbf{C}_g + \mathbf{P}_g$ and $\mathbf{T}$ are observed when the visual concreteness $\bar{c}$ is high: for *Captions* ($\bar{c} = 3.10$), the improvement is substantial: ($\Delta = +43$); for benchmarks with a lower concreteness (*News* with $\bar{c} = 2.61$ and *Forum* with $\bar{c} = 2.39$), the improvement is smaller ($\Delta = +12$). Thus, grounding brings useful complementary information, especially for concrete sentences.

*t-SNE visualization* This finding is also supported by a qualitative experiment showing that grounding groups together similar visual situations. Using sentences from CMPlaces (Castrejon et al., 2016), which describe visual scenes (e.g., *coast*, *shoe-shop*, *plaza*, etc.) and are classified in 205 scene categories, we randomly sample 5 visual scenes and plot in Figure 3 the corresponding sentences using t-SNE (Maaten and Hinton, 2008). We notice that our grounded model is better able to cluster sentences that have a close visual meaning than the text-only model. This is reinforced by the structural measures computed on the five clusters of Figure 3: $C_{inter} = 19, C_{intra} = 22$ for $\mathbf{T}$, $C_{inter} = 11, C_{intra} = 27$ for $\mathbf{C}_g + \mathbf{P}_g$. Indeed, $C_{inter}$ (resp. $C_{intra}$), is lower (resp. higher) for the grounded model $\mathbf{C}_g + \mathbf{P}_g$ compared to $\mathbf{T}$, which shows that clusters corresponding to different scenes are more clearly separated (resp. sentences corresponding to a given scene are more packed).

*Nearest neighbors search* Furthermore, we show in Table 2 that concrete knowledge acquired via our grounded model can also be transferred to abstract sentences. To do so, we manually build sentences using words with low concreteness (between 2.5 and 3.5) from the USF dataset (Nelson et al., 2004). Then, nearest neighbors are retrieved from the set of sentences of Flickr30K (Plummer et al., 2015). In this sample, we see that our grounded model is more accurate than the purely textual model to capture visual meaning. The observation that visual information propagates from

| Model | | MR | CR | SUBJ | MPQA | MRPC | SST | SNLI | SICK | AVG |
|---|---|---|---|---|---|---|---|---|---|---|
| (Kiros et al., 2015)† | $\mathbf{T}_{1024}$ | 72.7* | 75.2* | 90.6* | 84.7* | 71.8* /79.2* | 76.2* | 68.8* | 79.3* | 77.4 |
| (Kiela et al., 2018)† | GS-Cap | 72.0* | 76.8* | 90.7* | 85.5* | 72.9/80.6 | 76.7* | 73.7 | **82.9** | 78.4 |
| (Kiela et al., 2018)† | GS-Img | 74.5* | 79.3* | 90.8* | 87.8* | 73.0/80.3 | 80.0* | 72.2* | 80.9* | 79.8 |
| (Kiela et al., 2018)† | GS-Both | 72.5* | 75.7* | 90.7* | 85.4* | 72.9/81.3 | 76.7* | 72.2* | 81.4* | 78.4 |
| (Kiros et al., 2015)† | **T** | 75.9* | 79.2* | 92.0 | 86.7* | 72.2/80.2 | 81.8* | 72.0* | 81.1* | 80.1 |
| (Lazaridou et al., 2015a)‡ | $\mathbf{T} + \mathbf{CM}$ | 77.6 | 81.4 | 92.6 | 88.3 | 73.5/81.1 | 82.0* | 73.0 | 81.4* | 81.1 |
| (Collell et al., 2017)‡ | **SEQ** | 76.1* | 79.8* | 92.5 | 86.7* | 70.0* /79.5* | 81.7* | 67.3* | 76.7* | 78.9 |
| Model scenarios | $\mathbf{T} + \mathbf{P}_{id}$ | 77.5 | 81.5 | 92.7 | 88.4 | **73.7**/81.3 | 82.4 | 72.4 | 81.1 | 81.2 |
| | $\mathbf{T} + \mathbf{P}_{g}$ | **77.8** | **81.8** | **93.0** | 88.1 | 73.3/**81.6** | **83.5** | 72.8 | 82.2 | **81.6** |
| | $\mathbf{T} + \mathbf{C}_{id}$ | 77.5 | 81.6 | 92.8 | 88.3 | 72.9/80.5 | 82.2 | 73.1 | 82.3 | 81.3 |
| | $\mathbf{T} + \mathbf{C}_{g}$ | 77.3 | 81.5 | 92.8 | **88.6** | 73.6/81.1 | 82.6 | **74.1** | 82.6 | **81.6** |
| | $\mathbf{T} + \mathbf{C}_{id} + \mathbf{P}_{id}$ | 77.3 | 81.2 | **93.0** | 88.4 | 73.0/80.6 | 82.5 | 73.5 | 82.1 | 81.4 |
| | $\mathbf{T} + \mathbf{C}_{g} + \mathbf{P}_{g}$ | 77.4 | 81.5 | **93.0** | 88.1 | 73.2/80.9 | 82.7 | 73.9 | **82.9** | **81.6** |

Table 3: Extrinsic evaluations with SentEval. All models give sentences in dimension $d_t = 2048$ (except $\mathbf{T}_{1024}$). 'AVG' stands for the average accuracies reported in the other columns. '†': the model has been re-implemented (we obtained higher scores than the one given in the original papers). '‡': the baseline is an adaptation of the model to the case of sentences. '$*$': significantly differs from the best scenario among our models.

concrete sentences to abstract ones is analogous to findings made in previous research on word embeddings (Hill and Korhonen, 2014).

*Neighboring structure* To illustrate the discrepancy on the mNNO metric observed between $\mathbf{C}_g + \mathbf{P}_g$ and $\mathbf{T}$, we select a query image $Q$ in the validation set of MS COCO, along with its corresponding caption $S$; we display, in Figure 2, the nearest neighbor of $Q$ in the visual space, noted $N$, and the nearest neighbors of $S$ in the grounded space. With our grounded model, the neighborhood $S$ is mostly made of sentences corresponding to $Q$ or $N$.

**Hypotheses validation** We now validate our hypotheses (cf. sect. 3.1) on the grounded space, using the Cross-Modal Projection baseline (**CM**) and our model scenarios as outlined in Table 1. For fair comparison, metrics for the baseline **CM** are estimated either on the visual or the textual space depending on whether our models rely on the grounded space ($g$) or not ($id$). These results correspond to the rows **CM** (text) and **CM** (vis.) in Table 1.

Results highlight that: (1) Using a grounded space is beneficial; indeed, semantic relatedness and mNNO scores are higher in the lower half of Table 1, e.g., $\mathbf{C}_g > \mathbf{C}_{id}$, $\mathbf{P}_g > \mathbf{P}_{id}$ and $\mathbf{C}_g + \mathbf{P}_g > \mathbf{C}_{id} + \mathbf{P}_{id}$; (2) Solely using cluster information leads to the highest $C_{intra}$ and lowest $C_{inter}$, which suggests that $\mathbf{C}_\bullet$ is the most efficient model at separating visually different sentences; (3) Using only perceptual information in $\mathbf{P}_\bullet$ logically leads to highly correlated textual and visual spaces (highest $\rho_{vis}$), but the local neighborhood structure is not well preserved (lowest $C_{intra}$); (4) Our model $\mathbf{C}_\bullet + \mathbf{P}_\bullet$ is better than **CM** at capturing cluster information (higher $C_{intra}$, lower $C_{inter}$) and perceptual information (higher $\rho_{vis}$). This also translates in a higher mNNO measure for $\mathbf{C}_\bullet + \mathbf{P}_\bullet$, leading us to think that the conjunction of both perceptual and cluster information leads to high correlation of modalities, in terms of neighborhood structure. Moreover, this high mNNO score results in better performances for our model $\mathbf{C}_\bullet + \mathbf{P}_\bullet$ in terms of semantic relatedness.

### 5.2 Evaluation on transfer tasks

We now focus on extrinsic evaluation of the embeddings. Table 3 reports evaluations of our baselines and scenarios on SentEval (Conneau and Kiela, 2018), a classical benchmark used for evaluating sentence embeddings. Before further analysis, we find that our grounded models systematically outperform the textual baseline **T**, on all benchmarks, which shows the first substantial improvement brought by grounding and visual information in a sentence representation model. Indeed, models GS-Cap, GS-Img and GS-Both from (Kiela et al., 2018), despite improving over $\mathbf{T}_{1024}$, perform worse than the textual model of the same dimension **T** — this is consistent with what they report in their paper.

Our results interpretation is the following: (1) our joint approach shows superior performances over the sequential one, confirming results reported at the word level (Zablocki et al., 2018). Indeed, both sequential models, GS models (Kiela et al., 2018) and **SEQ** (inspired from (Collell et al., 2017)) are systematically worse than our grounded

models for all benchmarks. (2) Preserving the structure of the visual space is more effective than learning cross-modal projections; indeed, all our models outperform $\mathbf{T} + \mathbf{CM}$ on average ('AVG' column). (3) Making use of a grounded space yields slightly improved sentence representations. Indeed, our models that use the grounded space ($g$ = MLP) can take advantage of more expression power provided by the trainable $g$ than models which integrate grounded information directly in the textual space ($g = id$). (4) Among our model scenarios, $\mathbf{T} + \mathbf{P}_g$ has maximal scores on the most tasks; however, it shows lower scores on SNLI and SICK, which are entailment tasks. Models using cluster information $\mathbf{C}_g$ are naturally more suited for these tasks and hence obtain higher results. Finally, the combined model $\mathbf{T} + \mathbf{C}_g + \mathbf{P}_g$ shows a good balance between classification and entailment tasks.

## 6 Conclusion

We proposed a multimodal model aiming at preserving the structure of visual and textual spaces to learn grounded sentence representations. Our contributions include (1) leveraging both perceptual and cluster information and (2) using an intermediate grounded space enabling to relax the constraints on the textual space. Our approach is the first to report consistent positive results against purely textual baselines on a variety of natural language tasks. As future work, we plan to use visual information to specifically target complex downstream tasks requiring commonsense and reasoning such as question answering or visual dialogue.

## Acknowledgments

This work is partially supported by the CHIST-ERA EU project MUSTER (ANR-15-CHR2-0005) and the Labex SMART (ANR-11-LABX-65) supported by French state funds managed by the ANR within the Investissements d'Avenir program under reference ANR-11-IDEX-0004-02.

## References

Ignacio Arroyo-Fernández, Carlos-Francisco Méndez-Cruz, Gerardo Sierra, Juan-Manuel Torres-Moreno, and Grigori Sidorov. 2019. Unsupervised sentence representations as word information series: Revisiting TF-IDF. *Computer Speech & Language*, 56:107–129.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Marco Baroni. 2016. Grounding distributional semantics in the visual world. *Language and Linguistics Compass*, 10(1):3–13.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 632–642.

Elia Bruni, Nam Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *J. Artif. Int. Res.*, 49(1):1–47.

Marc Brysbaert, Amy Beth Warriner, and Victor Kuperman. 2013. Concreteness ratings for 40 thousand generally known english word lemmas. *Behavior research methods*, 46.

Curt Burgess and Kevin Lund. 1997. Modelling parsing constraints with high-dimensional context space. 12.

Micael Carvalho, Rémi Cadène, David Picard, Laure Soulier, Nicolas Thome, and Matthieu Cord. 2018. Cross-modal retrieval in the cooking context: Learning semantic text-image embeddings. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, pages 35–44.

Lluis Castrejon, Yusuf Aytar, Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. 2016. Learning aligned cross-modal representations from weakly aligned data. In *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*. IEEE.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder for english. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018: System Demonstrations, Brussels, Belgium, October 31 - November 4, 2018*, pages 169–174.

Daniel M. Cer, Mona T. Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation, SemEval@ACL 2017, Vancouver, Canada, August 3-4, 2017*, pages 1–14.

Noam Chomsky. 1980. Rules and representations. *Behavioral and brain sciences*, 3(1):1–15.

Grzegorz Chrupala, Ákos Kádár, and Afra Alishahi. 2015. Learning language through pictures. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*, pages 112–118.

Guillem Collell and Marie-Francine Moens. 2018. Do neural network cross-modal mappings really bridge modalities? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 462–468.

Guillem Collell, Teddy Zhang, and Marie-Francine Moens. 2017. Imagined visual representations as multimodal embeddings. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*. AAAI.

Alexis Conneau and Douwe Kiela. 2018. Senteval: An evaluation toolkit for universal sentence representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 670–680.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *COLING 2004, 20th International Conference on Computational Linguistics, Proceedings of the Conference, 23-27 August 2004, Geneva, Switzerland*.

Rebecca Fincher-Kiefer. 2001. Perceptual components of situation models. *Memory & Cognition*, 29(2):336–343.

Jonathan Gordon and Benjamin Van Durme. 2013. Reporting bias and knowledge acquisition. In *Proceedings of the 2013 Workshop on Automated Knowledge Base Construction*, AKBC '13, pages 25–30, New York, NY, USA. ACM.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 1367–1377.

Felix Hill and Anna Korhonen. 2014. Learning abstract concept embeddings from multi-modal data: Since you probably can't see what I mean. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 255–265.

Felix Hill, Roi Reichart, and Anna Korhonen. 2014. Multi-modal models for concrete and abstract concept meaning. *Transactions of the Association for Computational Linguistics*, 2:285–296.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 168–177, New York, NY, USA. ACM.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 655–665.

Andrej Karpathy and Fei-Fei Li. 2015. Deep visual-semantic alignments for generating image descriptions. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 3128–3137.

Douwe Kiela, Alexis Conneau, Allan Jabri, and Maximilian Nickel. 2018. Learning visually grounded sentence representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 408–418.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 3294–3302.

Satwik Kottur, Ramakrishna Vedantam, José M. F. Moura, and Devi Parikh. 2016. Visualword2vec (vis-w2v): Learning visually grounded word embeddings using abstract scenes. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 4985–4994.

Angeliki Lazaridou, Georgiana Dinu, and Marco Baroni. 2015a. Hubness and pollution: Delving into cross-space mapping for zero-shot learning. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 270–280. Association for Computational Linguistics.

Angeliki Lazaridou, Nghia The Pham, and Marco Baroni. 2015b. Combining language and vision with a multimodal skip-gram model. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 153–163.

Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: common objects in context. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*, pages 740–755.

Xiao Lin and Devi Parikh. 2015. Don't just listen, use your imagination: Leveraging visual common sense for non-visual tasks. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 2984–2993.

Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *CoRR*, abs/1703.03130.

Lajanugen Logeswaran and Honglak Lee. 2018. An efficient framework for learning sentence representations. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014a. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation, SemEval@COLING 2014, Dublin, Ireland, August 23-24, 2014.*, pages 1–8.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014b. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC 2014, Reykjavik, Iceland, May 26-31, 2014.*, pages 216–223.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 3111–3119.

Douglas L Nelson, Cathy L McEvoy, and Thomas A Schreiber. 2004. The university of south florida free association, rhyme, and word fragment norms. *Behavior Research Methods, Instruments, & Computers*, 36(3):402–407.

Donald A Norman. 1972. Memory, knowledge, and the answering of questions.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, 21-26 July, 2004, Barcelona, Spain.*, pages 271–278.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL 2005, 43rd Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, 25-30 June 2005, University of Michigan, USA*, pages 115–124.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 2227–2237.

Bryan A. Plummer, Liwei Wang, Chris M. Cervantes, Juan C. Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. 2015. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 2641–2649.

Carina Silberer and Mirella Lapata. 2014. Learning grounded meaning representations with autoencoders. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 721–732.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on*

*Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1631–1642.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2818–2826.

Lawrence W. Barsalou. 1999. Perceptual symbol systems. 22:577–609; discussion 610.

Hong Wang, Wenhan Xiong, Mo Yu, Xiaoxiao Guo, Shiyu Chang, and William Yang Wang. 2019. Sentence embedding alignment for lifelong relation extraction. *NAACL*.

Liwei Wang, Yin Li, and Svetlana Lazebnik. 2016. Learning deep structure-preserving image-text embeddings. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 5005–5013.

Janyce Wiebe and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. language resources and evaluation. In *Language Resources and Evaluation (formerly Computers and the Humanities*, page 2005.

Fanyi Xiao, Leonid Sigal, and Yong Jae Lee. 2017. Weakly-supervised visual grounding of phrases with linguistic structures. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 5253–5262.

Mark Yatskar, Vicente Ordonez, and Ali Farhadi. 2016. Stating the obvious: Extracting visual common sense knowledge. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 193–198.

Éloi Zablocki, Benjamin Piwowarski, Laure Soulier, and Patrick Gallinari. 2018. Learning multi-modal word representation grounded in visual context. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*.

# Neural Naturalist: Generating Fine-Grained Image Comparisons

**Maxwell Forbes** **Christine Kaeser-Chen** **Piyush Sharma** **Serge Belongie**

University of Washington Google Research Cornell University and Cornell Tech

mbforbes@cs.washington.edu

{christinech,piyushsharma}@google.com

sjb344@cornell.edu

**https://mbforbes.github.io/neural-naturalist/**

## Abstract

We introduce the new *Birds-to-Words* dataset of 41k sentences describing fine-grained differences between photographs of birds. The language collected is highly detailed, while remaining understandable to the everyday observer (e.g., "heart-shaped face," "squat body"). Paragraph-length descriptions naturally adapt to varying levels of taxonomic and visual distance—drawn from a novel stratified sampling approach—with the appropriate level of detail. We propose a new model called *Neural Naturalist* that uses a joint image encoding and comparative module to generate comparative language, and evaluate the results with humans who must use the descriptions to distinguish real images.

Our results indicate promising potential for neural models to explain differences in visual embedding space using natural language, as well as a concrete path for machine learning to aid citizen scientists in their effort to preserve biodiversity.

## 1 Introduction

Humans are adept at making fine-grained comparisons, but sometimes require aid in distinguishing visually similar classes. Take, for example, a citizen science effort like iNaturalist,[1] where everyday people photograph wildlife, and the community reaches a consensus on the taxonomic label for each instance. Many species are visually similar (e.g., Figure 1, top), making them difficult for a casual observer to label correctly. This puts an undue strain on lieutenants of the citizen science community to curate and justify labels for a large number of instances. While everyone may be capable of making such distinctions visually, non-experts require training to know what to look for.

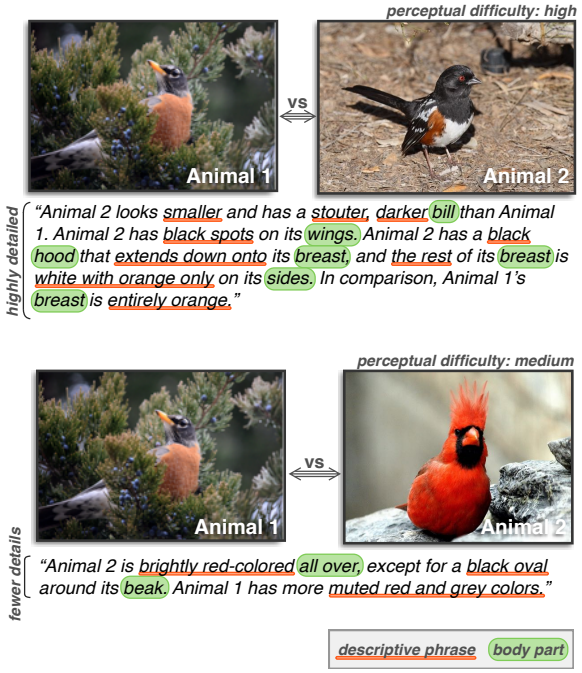


Figure 1: The Birds-to-Words dataset: comparative descriptions adapt naturally to the appropriate level of detail (orange underlines). A difficult distinction (TOP) is given a longer and more fined-grained comparison than an easier one (BOTTOM). Annotators organically use everyday language to refer to parts (green highlights).

Field guides exist for the purpose helping people learn how to distinguish between species. Unfortunately, field guides are costly to create because writing such a guide requires expert knowledge of class-level distinctions.

In this paper, we study the problem of explaining the differences between two images using natural language. We introduce a new dataset called *Birds-to-Words* of paragraph-length descriptions of the differences between pairs of bird photographs. We find several benefits from eliciting comparisons: (a) without a guide, annotators naturally break down the subject of the image (e.g.,

Work done during an internship at Google.

[1] https://www.inaturalist.org



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 708–717, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

a bird) into pieces understood by the everyday observer (e.g., head, wings, legs); (b) by sampling comparisons from varying visual and taxonomic distances, the language exhibits naturally adaptive granularity of detail based on the distinctions required (e.g., "red body" vs "tiny stripe above its eye"); (c) in contrast to requiring comparisons between categories (e.g., comparing one species vs. another), non-experts can provide high-quality annotations without needing domain expertise.

We also propose the *Neural Naturalist* model architecture for generating comparisons given two images as input. After embedding images into a latent space with a CNN, the model combines the two image representations with a joint encoding and comparative module before passing them to a Transformer decoder. We find that introducing a comparative module—an additional Transformer encoder—over the combined latent image representations yields better generations.

Our results suggest that these classes of neural models can assist in fine-grained visual domains when humans require aid to distinguish closely related instances. Non-experts—such as amateur naturalists trying to tell apart two species—stand to benefit from comparative explanations. Our work approaches this sweet-spot of visual expertise, where any two in-domain images can be compared, and the language is detailed, adaptive to the types of differences observed, and still understandable by laypeople.

Recent work has made impressive progress on context sensitive image captioning. One direction of work uses class labels as context, with the objective of generating captions that distinguish why the image belongs to one class over others (Hendricks et al., 2016; Vedantam et al., 2017). Another choice is to use a second image as context, and generate a caption that distinguishes one image from another. Previous work has studied ways to generalize single-image captions into comparative language (Vedantam et al., 2017), as well as comparing two images with high pixel overlap (e.g., surveillance footage) (Jhamtani and Berg-Kirkpatrick, 2018). Our work complements these efforts by studying directly comparative, everyday language on image pairs with no pixel overlap.

Our approach outlines a new way for models to aid humans in making visual distinctions. The Neural Naturalist model requires two instances as input; these could be, for example, a query image

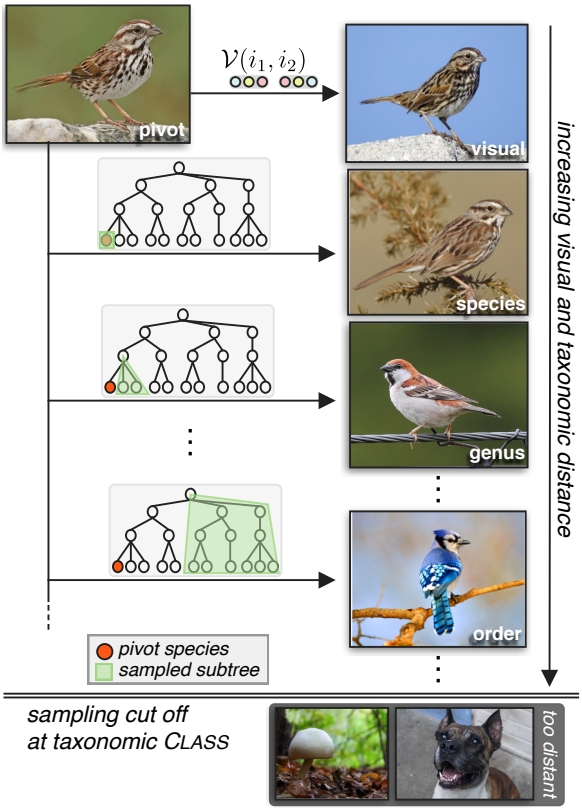


Figure 2: Illustration of pivot-branch stratified sampling algorithm used to construct the Birds-to-Words dataset. The algorithm harnesses visual and taxonomic distances (increasing vertically) to create a challenging task with board coverage.

and an image from a candidate class. By differentiating between these two inputs, a model may help point out subtle distinctions (e.g., one animal has spots on its side), or features that indicate a good match (e.g., only a slight difference in size). These explanations can aid in understanding both differences between species, as well as variance within instances of a single species.

## 2 Birds-to-Words Dataset

Our goal is to collect a dataset of tuples $(i_1, i_2, t)$, where $i_1$ and $i_2$ are images, and $t$ is a natural language comparison between the two. Given a domain $\mathcal{D}$, this collection depends critically on the criteria we use to select image pairs.

If we sample image pairs uniformly at random, we will end up with comparisons encompassing a broad range of phenomena. For example, two images that are quite different will yield categorical comparisons (*"One is a bird, one is a mushroom."*). Alternatively, if the two images are very similar, such as two angles of the same creature,

| Dataset | Domain | Lang | Images Ctx | Images Cap | Example |
|---|---|---|---|---|---|
| CUB Captions *(R, 2016)* | Birds | M | 1 | 1 | *"An all black bird with a very long rectrices and relatively dull bill."* |
| CUB-Justify *(V, 2017)* | Birds | S | 7 | 1 | *"The bird has white orbital feathers, a black crown, and yellow tertials."* |
| Spot-the-Diff *(J&B, 2018)* | Surveilance | E | 2 | 1–2 | *"Silver car is gone. Person in a white t shirt appears. 3rd person in the group is gone."* |
| Birds-to-Words *(this work)* | Birds | E | 2 | 2 | *"Animal1 is gray, while animal2 is white. Animal2 has a long, yellow beak, while animal1's beak is shorter and gray. Animal2 appears to be larger than animal1."* |

Table 1: Comparison with recent fine-grained language-and-vision datasets. *Lang* values: S = scientific, E = everyday, M = mixed. *Images Ctx* = number of images shown, *Images Cap* = number of images described in caption. Dataset citations: *R* = *Reed et al.*, *V* = *Vedantam et al.*, *J&B* = *Jhamtani and Berg-Kirkpatrick*.

comparisons between them will focus on highly detailed nuances, such as variations in pose. These phenomena support rich lines of research, such as object classification (Deng et al., 2009) and pose estimation (Murphy-Chutorian and Trivedi, 2009).

We aim to land somewhere in the middle. We wish to consider sets of distinguishable but intimately related pairs. This sweet spot of visual similarity is akin to the genre of differences studied in fine-grained visual classification (Wah et al., 2011; Krause et al., 2013). We approach this collection with a two-phase data sampling procedure. We first select *pivot* images by sampling from our full domain uniformly at random. We then *branch* from these images into a set of secondary images that emphases fine-grained comparisons, but yields broad coverage over the set of sensible relations. Figure 2 provides an illustration of our sampling procedure.

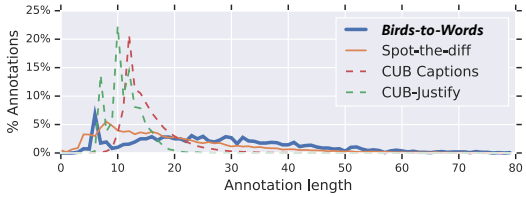


| Birds-to-Words Dataset | |
|---|---|
| Image pairs | 3,347 |
| Paragraphs / pair | 4.8 |
| Paragraphs | 16,067 |
| Tokens / paragraph | 32.1 MEAN |
| Sentences | 40,969 |
| Sentences / paragraph | 2.6 MEAN |
| Clarity rating | $\geq 4/5$ |
| Train / dev / test | 80% / 10% / 10% |

Figure 3: Annotation lengths for compared datasets (TOP), and statistics for the proposed Birds-to-Words dataset (BOTTOM). The Birds-to-Words dataset has a large mass of long descriptions in comparison to related datasets.

## 2.1 Domain

We sample images from iNaturalist, a citizen science effort to collect research-grade[2] observations of plants and animals in the wild. We restrict our domain $\mathcal{D}$ to instances labeled under the taxonomic CLASS[3] *Aves* (i.e., birds). While a broader domain would yield some comparable instances (e.g., *bird* and *dragonfly* share some common body parts), choosing only *Aves* ensures that all instances will be similar enough structurally to be comparable, and avoids the gut reaction comparison pointing out the differences in animal type. This choice yields 1.7M research-grade images and corresponding taxonomic labels from iNaturalist. We then perform pivot-branch sampling on this set to choose pairs for annotation.

## 2.2 Pivot Images

The *Aves* domain in iNaturalist contains instances of 9k distinct species, with heavy observation bias to more common species (such as the mallard duck). We uniformly sample species from the set of 9k to help overcome this bias. In total, we select 405 species and corresponding photographs to use as $i_1$ images.

[2]Research-grade observations have met or exceeded iNaturalist's guidelines for community consensus of the taxonomic label for a photograph.

[3]To disambiguate *class*, we use CLASS to denote the taxonomic rank in scientific classification, and simply "class" to refer to the machine learning usage of the term as a label in classification.

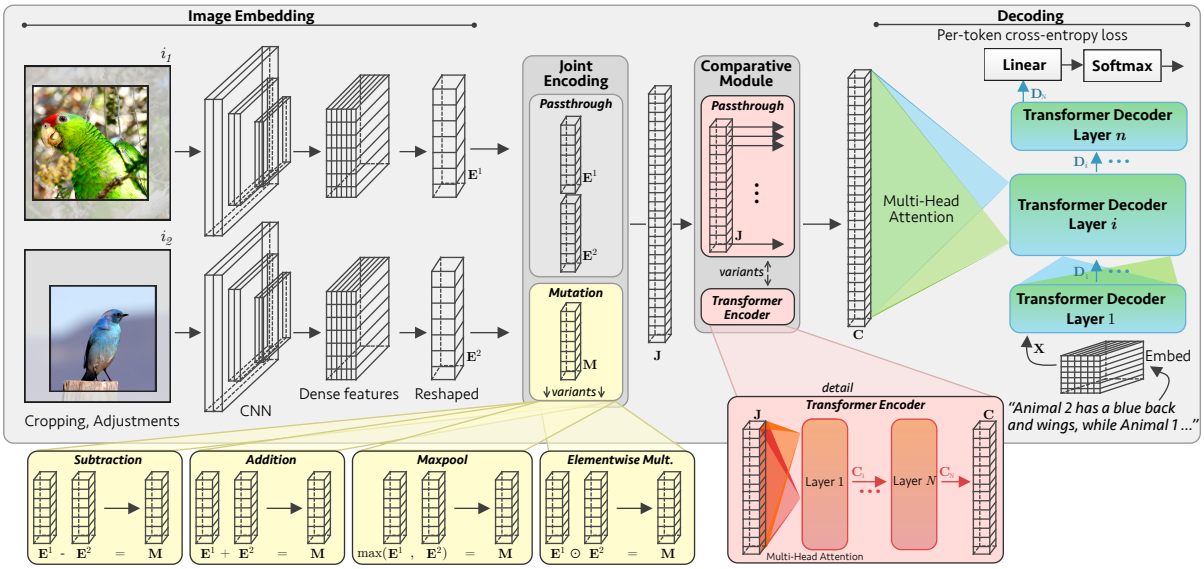


Figure 4: The proposed Neural Naturalist model architecture. The multiplicative joint encoding and Transformer-based comparative module yield the best comparisons between images.

### 2.3 Branching Images

We use both a visual similarity measure and taxonomy to sample a set of comparison images $i_2$ branching off from each pivot image $i_1$. We use a branching factor of $k = 12$ from each pivot image.

To capture visually similar images to $i_1$, we employ a similarity function $\mathcal{V}(i_1, i_2)$. We use an Inception-v4 (Szegedy et al., 2017) network pretrained on ImageNet (Deng et al., 2009) and then fine-tuned to perform species classification on all research-grade observations in iNaturalist. We take the embedding for each image from the last layer of the network before the final softmax. We perform a k-nearest neighbor search by quantizing each embedding and using L2 distance (Wu et al., 2017; Guo et al., 2016), selecting the $k_v = 2$ closest images in embedding space.

We also use the iNaturalist scientific taxonomy $\mathcal{T}(\mathcal{D})$ to sample images at varying levels of taxonomic distance from $i_1$. We select $k_t = 10$ taxonomically branched images by sampling two images each from the same SPECIES ($\ell = 1$), GENUS, FAMILY, ORDER, and CLASS ($\ell = 5$) as $c$. This yields 4,860 raw image pairs $(i_1, i_2)$.

### 2.4 Language Collection

For each image pair $(i_1, i_2)$, we elicit five natural language paragraphs describing the differences between them.

An annotator is instructed to write a paragraph (usually 2–5 sentences) comparing and contrasting the animal appearing in each image. We instruct annotators not to explicitly mention the species (e.g., *"Animal 1 is a penguin"*), and to instead focus on visual details (e.g., *"Animal 1 has a black body and a white belly"*). They are additionally instructed to avoid mentioning aspects of the background, scenery, or pose captured in the photograph (e.g., *"Animal 2 is perched on a coconut"*).

We discard all annotations for an image pair where either image did not have at least $\frac{4}{5}$ positive ratings of image clarity. This yields a total of 3,347 image pairs, annotated with 16,067 paragraphs. Detailed statistics of the Birds-to-Words dataset are shown in Figure 3, and examples are provided in Figure 5. Further details of our both our algorithmic approach and dataset construction are given in Appendices A and B.

## 3 Neural Naturalist Model

**Task** Given two images $(i_1, i_2)$ as input, our task is to generate a natural language paragraph $t = x_1 \dots x_n$ that compares the two images.

**Architecture** Recent image captioning approaches (Xu et al., 2015; Sharma et al., 2018) extract image features using a convolutional neural network (CNN) which serve as input to a language decoder, typically a recurrent neural network (RNN) (Mikolov et al., 2010) or Transformer (Vaswani et al., 2017). We extend this paradigm with a joint encoding step and comparative module to study how best to encode and transform




Figure 5: Samples from the dev split of the proposed Birds-to-Words dataset, along with Neural Naturalist model output (M) and one of five ground truth paragraphs (G). The second row highlights failure cases in red. The model produces *coherent* descriptions of *variable granularity*, though *emphasis* and *assignment* can be improved.

multiple latent image embeddings. A schematic of the model is outlined in Figure 4, and its key components are described in the upcoming sections.

$$\mathbf{E}^1 = \langle \mathbf{e}^1_{1,1}, \ldots, \mathbf{e}^1_{d,d} \rangle = \text{CNN}(i_1)$$
$$\mathbf{E}^2 = \langle \mathbf{e}^2_{1,1}, \ldots, \mathbf{e}^2_{d,d} \rangle = \text{CNN}(i_2)$$

### 3.1 Image Embedding

Both input images are first processed using CNNs with shared weights. In this work, we consider ResNet (He et al., 2016) and Inception (Szegedy et al., 2017) architectures. In both cases, we extract the representation from the deepest layer immediately before the classification layer. This yields a dense 2D grid of local image feature vectors, shaped $(d, d, f)$. We then flatten each feature grid into a $(d^2, f)$ shaped matrix:

### 3.2 Joint Encoding

We define a joint encoding $\mathbf{J}$ of the images which contains both embedded images $(\mathbf{E}^1, \mathbf{E}^2)$, a mutated combination $(\mathbf{M})$, or both. We consider as possible mutations $\mathbf{M} \in \{\mathbf{E}^1 + \mathbf{E}^2, \mathbf{E}^1 - \mathbf{E}^2, \max(\mathbf{E}^1, \mathbf{E}^2), \mathbf{E}^1 \odot \mathbf{E}^2\}$. We try these encoding variants to explore whether simple mutations can effectively combine the image representations.

### 3.3 Comparative Module

Given the joint encoding of the images ($\mathbf{J}$), we would like to represent the differences in feature space ($\mathbf{C}$) in order to generate comparative descriptions. We explore two variants at this stage. The first is a direct passthrough of the joint encoding ($\mathbf{C} = \mathbf{J}$). This is analogous to "standard" CNN+LSTM architectures, which embed images and pass them directly to an LSTM for decoding. Because we try different joint encodings, a passthrough here also allows us to study their effects in isolation.

Our second variant is an $N$-layer Transformer encoder. This provides an additional self-attentive mutations over the latent representations $\mathbf{J}$. Each layer contains a multi-headed attention mechanism ($\text{ATTN}_{\text{MH}}$). The intent is that self-attention in Transformer encoder layers will guide comparisons across the joint image encoding.

Denoting LN as *Layer Norm* and FF as *Feed Forward*, with $\mathbf{C}_i$ as the output of the $i$th layer of the Transformer encoder, $\mathbf{C}_0 = \mathbf{J}$, and $\mathbf{C} = \mathbf{C}_N$:

$$\mathbf{C}_i^H = \text{LN}(\mathbf{C}_{i-1} + \text{ATTN}_{\text{MH}}(\mathbf{C}_{i-1}))$$
$$\mathbf{C}_i = \text{LN}(\mathbf{C}_i^H + \text{FF}(\mathbf{C}_i^H))$$

### 3.4 Decoder

We use an $N$-layer Transformer decoder architecture to produce distributions over output tokens. The Transformer decoder is similar to an encoder, but it contains an intermediary multi-headed attention which has access to the encoder's output $\mathbf{C}$ at every time step.

$$\mathbf{D}_i^{H_1} = \text{LN}(\mathbf{X} + \text{ATTN}_{\text{MASK,MH}}(\mathbf{X}))$$
$$\mathbf{D}_i^{H_2} = \text{LN}(\mathbf{D}_i^{H_1} + \text{ATTN}_{\text{MH}}(\mathbf{D}_i^{H_1}, \mathbf{C}))$$
$$\mathbf{D}_i = \text{LN}(\mathbf{D}_i^{H_2} + \text{FF}(\mathbf{D}_i^{H_2}))$$

Here we denote the text observed during training as $\mathbf{X}$, which is modulated with a position-based encoding and masked in the first multi-headed attention.

## 4 Experiments

We train the Neural Naturalist model to produce descriptions of the differences between images in the Birds-to-Words dataset. We partition the dataset into train (80%), val (10%), and test (10%) sections by splitting based on the pivot images $i_1$. This ensures $i_1$ species are unique across the different splits.

We provide model hyperparameters and optimization details in Appendix C.

### 4.1 Baselines and Variants

The *most frequent* paragraph baseline produces only the most observed description in the training data, which is that the two animals appear to be exactly the same. *Text-Only* samples captions from the training data according to their empirical distribution. *Nearest Neighbor* embeds both images and computes the lowest total $L_2$ distance to a training set pair, sampling a caption from it. We include two standard neural baselines, CNN (+ Attention) + LSTM, which concatenate the images embeddings, optionally perform attention, and decode with an LSTM. The main model variants we consider are a simple joint encoding ($\mathbf{J} = \langle \mathbf{E}^1, \mathbf{E}^2 \rangle$), no comparative module ($\mathbf{C} = \mathbf{J}$), a small (1-layer) decoder, and our full Neural Naturalist model. We also try several other ablations and model variants, which we describe later.

### 4.2 Quantitative Results

**Automatic Metrics** We evaluate our model using three machine-graded text metrics: BLEU-4 (Papineni et al., 2002), ROUGE-L (Lin, 2004), and CIDEr-D (Vedantam et al., 2015). Each generated paragraph is compared to all five reference paragraphs.

For human performance, we use a one-vs-rest scheme to hold one reference paragraph out and compute its metric using the other four. We average this score across twenty-five runs over the entire split in question.

Results using these metrics are given in Table 2 for the main baselines and model variants. We observe improvement across BLEU-4 and ROUGE-L scores compared to baselines. Curiously, we observe that the CIDEr-D metric is susceptible to common patterns in the data; our model, when stopped at its highest CIDEr-D score, outputs a variant of, *"these animals appear exactly the same"* for 95% of paragraphs, nearly mimicking the behavior of the most frequent paragraph (*Freq.*) baseline. The corpus-level behavior of CIDEr-D gives these outputs a higher score. We observed anecdotally higher quality outputs correlated with ROUGE-L score, which we verify using a human evaluation (paragraph after next).

| | Dev | | | Test | | |
|---|---|---|---|---|---|---|
| | BLEU-4 | ROUGE-L | CIDEr-D | BLEU-4 | ROUGE-L | CIDEr-D |
| Most Frequent | 0.20 | 0.31 | **0.42** | 0.20 | 0.30 | **0.43** |
| Text-Only | 0.14 | 0.36 | 0.05 | 0.14 | 0.36 | 0.07 |
| Nearest Neighbor | 0.18 | 0.40 | 0.15 | 0.14 | 0.36 | 0.06 |
| CNN + LSTM (Vinyals et al., 2015) | 0.22 | 0.40 | 0.13 | 0.20 | 0.37 | 0.07 |
| CNN + Attn. + LSTM (Xu et al., 2015) | 0.21 | 0.40 | 0.14 | 0.19 | 0.38 | 0.11 |
| Neural Naturalist – Simple Joint Encoding | 0.23 | 0.44 | 0.23 | - | - | - |
| Neural Naturalist – No Comparative Module | 0.09 | 0.27 | 0.09 | - | - | - |
| Neural Naturalist – Small Decoder | 0.22 | 0.42 | 0.25 | - | - | - |
| Neural Naturalist – Full | **0.24** | **0.46** | 0.28 | **0.22** | **0.43** | 0.25 |
| Human | 0.26 +/- 0.02 | 0.47 +/- 0.01 | 0.39 +/- 0.04 | 0.27 +/- 0.01 | 0.47 +/- 0.01 | 0.42 +/- 0.03 |

Table 2: Experimental results for comparative paragraph generation on the proposed dataset. For human captions, mean and standard deviation are given for a one-vs-rest scheme across twenty-five runs. We observed that CIDEr-D scores had little correlation with description quality. The Neural Naturalist model benefits from a strong joint encoding and Transformer-based comparative module, achieving the highest BLEU-4 and ROUGE-L scores.

**Ablations and Model Variants** We ablate and vary each of the main model components, running the automatic metrics to study coarse changes in the model's behavior. Results for these experiments are given in Table 3. For the *joint encoding*, we try combinations of four element-wise operations with and without both encoded images. To study the *comparative module* in greater detail, we examine its effect on the top three joint encodings: $(i_1, i_2, -)$, $-$, and $\odot$. After fixing the best joint encoding and comparative module, we also try variations of the *decoder* (Transformer depth), as well as *decoding algorithms* (greedy decoding, multinomial sampling, and beamsearch).

Overall, we we see that the choice of joint encoding requires a balance with the choice of comparative module. More disruptive joint encodings (like element-wise multiplication $\odot$) appear too destructive when passed directly to a decoder, but yield the best performance when paired with a deep comparative module. Others (like subtraction) function moderately well on their own, and are further improved when a comparative module is introduced.

**Human Evaluation** To verify our observations about model quality and automatic metrics, we also perform a human evaluation of the generated paragraphs. We sample 120 instances from the test set, taking twenty each from the six categories for choosing comparative images (visual similarity in embedding space, plus five taxonomic distances). We provide annotators with the two images in a random order, along with the output from the model at hand. Annotators must decide which image contains *Animal 1*, and which contains *Animal 2*, or they may say that there is no way to tell (e.g., for a description like *"both look exactly the same"*).

We collect three annotations per datum, and score a decision only if $\geq 2/3$ annotators made that choice. A model receives +1 point if annotators decide correctly, 0 if they cannot decide or agree there is no way to tell, and -1 point if they decide incorrectly (label the images backwards). This scheme penalizes a model for confidently writing incorrect descriptions. The total score is then normalized to the range $[-1, 1]$. Note that *Human* uses one of the five gold paragraphs sampled at random.

Results for this experiment are shown in Table 4. In this measure, we see the frequency and text-only baselines now fall flat, as expected. The frequency baseline never receives any points, and the text-only baseline is often penalized for incorrectly guessing. Our model is successful at making distinctions between visually distinct species (GENUS column and ones further right), which is near the challenge level of current fine-grained visual classification tasks. However, it struggles on the two data subsets with highest visual similarity (VISUAL, SPECIES). The significant gap between all methods and human performance in these columns indicates ultra fine-grained distinctions are still possible for humans to describe, but pose a challenge for current models to capture.

### 4.3 Qualitative Analysis

In Figure 5, we present several examples of the model output for pairs of images in the dev set, along with one of the five reference paragraphs. In

| Joint Encoding | | | | | | Comparative Module | Decoder | Decoding Algorithm | Dev | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $i_1$ | $i_2$ | $-$ | $+$ | max | $\odot$ | | | | BLEU-4 | ROUGE-L | CIDEr-D |
| ✓ | ✓ | | | | | 6-Layer Transformer | 6-Layer Transformer | Beamsearch | 0.23 | 0.44 | 0.23 |
| | | ✓ | | | | | | | 0.23 | 0.45 | 0.27 |
| | | | ✓ | | | | | | **0.24** | 0.43 | **0.28** |
| | | | | ✓ | | | | | 0.23 | 0.43 | 0.24 |
| | | | | | ✓ | | | | **0.24** | **0.46** | **0.28** |
| ✓ | ✓ | ✓ | | | | | | | 0.22 | 0.44 | 0.22 |
| ✓ | ✓ | | ✓ | | | | | | 0.22 | 0.42 | 0.25 |
| ✓ | ✓ | | | ✓ | | | | | 0.21 | 0.42 | 0.22 |
| ✓ | ✓ | | | | ✓ | | | | 0.22 | 0.43 | 0.23 |
| ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | | | 0.21 | 0.43 | 0.20 |
| | | | | | ✓ | Passthrough | 6-Layer Transformer | Beamsearch | 0.00 | 0.02 | 0.00 |
| | | | | | ✓ | 1-L Transformer | | | **0.24** | 0.44 | 0.27 |
| | | | | | ✓ | 3-L Transformer | | | **0.24** | 0.44 | 0.27 |
| | | | | | ✓ | 6-L Transformer | | | **0.24** | **0.46** | **0.28** |
| | | ✓ | | | | Passthrough | 6-Layer Transformer | Beamsearch | 0.22 | 0.40 | 0.22 |
| | | ✓ | | | | 1-L Transformer | | | 0.21 | 0.41 | 0.26 |
| | | ✓ | | | | 3-L Transformer | | | 0.22 | 0.41 | 0.22 |
| | | ✓ | | | | 6-L Transformer | | | **0.23** | **0.45** | **0.27** |
| ✓ | ✓ | ✓ | | | | Passthrough | 6-Layer Transformer | Beamsearch | 0.09 | 0.27 | 0.09 |
| ✓ | ✓ | ✓ | | | | 1-L Transformer | | | **0.24** | 0.43 | 0.24 |
| ✓ | ✓ | ✓ | | | | 3-L Transformer | | | 0.22 | 0.42 | **0.26** |
| ✓ | ✓ | ✓ | | | | 6-L Transformer | | | 0.22 | **0.44** | 0.22 |
| | | | | | ✓ | 6-Layer Transformer | 1-L Transformer | Beamsearch | 0.22 | 0.42 | 0.25 |
| | | | | | ✓ | | 3-L Transformer | | 0.23 | 0.42 | 0.25 |
| | | | | | ✓ | | 6-L Transformer | | **0.24** | **0.46** | **0.28** |
| | | | | | ✓ | 6-Layer Transformer | 6-Layer Transformer | Greedy | 0.21 | 0.44 | 0.18 |
| | | | | | ✓ | | | Multinomial | 0.20 | 0.42 | 0.16 |
| | | | | | ✓ | | | Beamsearch | **0.24** | **0.46** | **0.28** |

Table 3: Variants and ablations for the Neural Naturalist model. We find the best performing combination is an elementwise multiplication ($\odot$) for the joint encoding, a 6-layer Transformer comparative module, a 6-layer Transformer decoder, and using beamsearch to perform inference.

the following section, we split an analysis of the model into two parts: largely positive findings, as well as common error cases.

### Positive Findings

We find that the model exhibits **dynamic granularity**, by which we mean that it adjusts the magnitude of the descriptions based on the scale of differences between the two animals. If two animals are quite similar, it generates fine-grained descriptions such as, *"Animal 2 has a slightly more curved beak than Animal 1,"* or *"Animal 1 is more iridescent than Animal 2."* If instead the two animals are very different, it will generate text describing larger-scale differences, like, *"Animal 1 has a much longer neck than Animal 2,"* or *"Animal 1 is mostly white with a black head. Animal 2 is almost completely yellow."*

We also observe that the model is able to produce coherent paragraphs of **varying linguistic structure.** These include a range of comparisons set up across both single and multiple sentences. For example, one it generates straightforward comparisons of the form, *Animal 1 has X, while Animal 2 has Y.* But it also generates contrastive expressions with longer dependencies, such as *Animal 1 is X, Y, and Z. Animal 2 is very similar, except W.* Furthermore, the model will mix and match different comparative structures within a single paragraph.

Finally, in addition to varying linguistic structure, we find the model is able to produce **coherent semantics** through a series of statements. For example, consider the following full output: *"Animal 1 has a very long neck compared to Animal 2. Animal 1 has shorter legs than Animal 2. Animal 1 has a black beak, Animal 2 has a brown beak. Animal 1 has a yellow belly. Animal 2 has*

| | VISUAL | SPECIES | GENUS | FAMILY | ORDER | CLASS |
|---|---|---|---|---|---|---|
| Freq. | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| Text-Only | 0.00 | -0.10 | -0.05 | 0.00 | 0.15 | -0.15 |
| CNN + LSTM | -0.15 | **0.20** | 0.15 | **0.50** | 0.40 | 0.15 |
| CNN + Attn. + LSTM | **0.15** | 0.15 | 0.15 | -0.05 | 0.05 | 0.20 |
| Neural Naturalist | 0.10 | -0.10 | **0.35** | 0.40 | **0.45** | **0.55** |
| Human | 0.55 | 0.55 | 0.85 | 1.00 | 1.00 | 1.00 |

Table 4: Human evaluation results on 120 test set samples, twenty per column. Scale: -1 (perfectly wrong) to 1 (perfectly correct). Columns are ordered left-to-right by increasing distance. Our model outperforms baselines for several distances, though highly similar comparisons still prove difficult.

*darker wings than Animal 1.*" The range of concepts in the output covers *neck, legs, beak, belly, wings* without repeating any topic or getting sidetracked.

### Error Analysis

We also observe several patterns in the model's shortcomings. The most prominent error case is that the model will sometimes **hallucinate differences** (Figure 5, bottom row). These range from pointing out significant changes that are missing (e.g., *"a black head"* where there is none (Fig. 5, bottom left)), to clawing at subtle distinctions where there are none (e.g., *"[its] colors are brighter . . . and [it] is a bit bigger"* (Fig. 5, bottom right)). We suspect that the model has learned some associations between common features in animals, and will sometimes favor these associations over visual evidence.

The second common error case is **missing obvious distinctions**. This is observed in Fig. 5 (bottom middle), where the prominent beak of Animal 1 is ignored by the model in favor of mundane details. While outlying features make for lively descriptions, we hypothesize that the model may sometimes avoid taking them into account given its per-token cross entropy learning objective.

Finally, we also observe the model sometimes **swaps which features are attributed to which animal**. This is partially observed in Fig. 5 (bottom left), where the *"black head"* actually belongs to Animal 1, not Animal 2. We suspect that mixing up references may be a trade-off for the representational power of attending over both images; there is no explicit bookkeeping mechanism to enforce which phrases refer to which feature comparisons in each image.

## 5 Related Work

Employing visual comparisons to elicit focused natural language observations was proposed by (Maji, 2012), and later investigated in the context of crowdsourcing by (Zou et al., 2015). We take inspiration from these works.

Previous work has collected natural language captions of bird photographs: CUB Captions (Reed et al., 2016) and CUB-Justify (Vedantam et al., 2017) are both language annotations on top of the CUB-2011 dataset of bird photographs (Wah et al., 2011). In addition to describing two photos instead of one, the language in our dataset is more complex by comparison, containing a diversity of comparative structures and implied semantics. We also collect our data without an anatomical guide for annotators, yielding everyday language in place of scientific terminology.

Conceptually, our paper offers a complementary approach to works that generate single-image class-discriminative or image-discriminative captions (Hendricks et al., 2016; Vedantam et al., 2017). Rather than discriminative captioning, we focus on comparative language as a means for bridging the gap between varying granularities of visual diversity.

Methodologically, our work is most closely related to the Spot-the-diff dataset (Jhamtani and Berg-Kirkpatrick, 2018). While dataset captions two images with only a small section of pixels that change (surveillance footage), we consider image pairs with no pixel overlap, which motivates our stratified sampling approach for drawing good comparisons.

Finally, the recently released NLVR$^2$ dataset (Suhr et al., 2018) introduces a challenging natural language reasoning task using two images as context. Our work instead focuses on generating comparative language rather than reasoning.

## 6 Conclusion

We present the new Birds-to-Words dataset and Neural Naturalist model for generating comparative explanations of fine-grained visual differences. This dataset features paragraph-length, adaptively detailed descriptions written in everyday language. We hope that continued study of this area will produce models that can aid humans in critical domains like citizen science.

## References


Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database.

Ruiqi Guo, Sanjiv Kumar, Krzysztof Choromanski, and David Simcha. 2016. Quantization based fast inner product search. In *Artificial Intelligence and Statistics*, pages 482–490.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Lisa Anne Hendricks, Zeynep Akata, Marcus Rohrbach, Jeff Donahue, Bernt Schiele, and Trevor Darrell. 2016. Generating visual explanations. In *European Conference on Computer Vision*, pages 3–19. Springer.

Harsh Jhamtani and Taylor Berg-Kirkpatrick. 2018. Learning to describe differences between pairs of similar images. *arXiv preprint arXiv:1808.10584*.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 2013. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Subhransu Maji. 2012. Discovering a lexicon of parts and attributes. In *European Conference on Computer Vision*, pages 21–30. Springer.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*.

Erik Murphy-Chutorian and Mohan Manubhai Trivedi. 2009. Head pose estimation in computer vision: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 31(4):607–626.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Scott Reed, Zeynep Akata, Honglak Lee, and Bernt Schiele. 2016. Learning deep representations of fine-grained visual descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 49–58.

Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. 2018. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2556–2565.

Alane Suhr, Stephanie Zhou, Iris Zhang, Huajun Bai, and Yoav Artzi. 2018. A corpus for reasoning about natural language grounded in photographs. *CoRR*, abs/1811.00491.

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. 2017. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Ramakrishna Vedantam, Samy Bengio, Kevin Murphy, Devi Parikh, and Gal Chechik. 2017. Context-aware captions from context-agnostic supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 251–260.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. 2011. The caltech-ucsd birds-200-2011 dataset.

Xiang Wu, Ruiqi Guo, Ananda Theertha Suresh, Sanjiv Kumar, Daniel N Holtmann-Rice, David Simcha, and Felix Yu. 2017. Multiscale quantization for fast similarity search. In *Advances in Neural Information Processing Systems*, pages 5745–5755.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057.

James Y Zou, Kamalika Chaudhuri, and Adam Tauman Kalai. 2015. Crowdsourcing feature discovery via adaptively chosen comparisons. *arXiv preprint arXiv:1504.00064*.

# Fine-Grained Evaluation for Entity Linking

**Henry Rosales-Méndez, Aidan Hogan, Barbara Poblete**
IMFD Chile & Department of Computer Science, University of Chile
{hrosales,ahogan,bpoblete}@dcc.uchile.cl

## Abstract

The Entity Linking (EL) task identifies entity mentions in a text corpus and associates them with an unambiguous identifier in a Knowledge Base. While much work has been done on the topic, we first present the results of a survey that reveal a lack of consensus in the community regarding what forms of mentions in a text and what forms of links the EL task should consider. We argue that no one definition of the Entity Linking task fits all, and rather propose a fine-grained categorization of different types of entity mentions and links. We then re-annotate three EL benchmark datasets – ACE2004, KORE50, and VoxEL – with respect to these categories. We propose a fuzzy recall metric to address the lack of consensus and conclude with fine-grained evaluation results comparing a selection of online EL systems.

## 1 Introduction

Entity Linking (EL) is an Information Extraction task whose goal is to identify mentions of entities in a text and to link each mention to an unambiguous identifier in a Knowledge Base (KB) such as Wikipedia, BabelNet (Moro et al., 2014), DBpedia (Lehmann et al., 2015), Freebase (Bollacker et al., 2008), Wikidata (Vrandečić and Krötzsch, 2014) or YAGO (Rebele et al., 2016) (among others). The results of this task offer a practical bridge between unstructured text and structured KBs, where EL has applications for semantic search, document classification, relation extraction, and more besides (Wu et al., 2018).

While a broad number of EL techniques and systems have been proposed in recent years (Wu et al., 2018), a number of authors have noted that there is a lack of consensus on the fundamental question of what kinds of mentions in a text an EL system should link to which identifiers in the KB (Ling et al., 2015; Waitelonis et al., 2016; Jha et al., 2017; Rosales-Méndez et al., 2018b).

A closely related task to EL is that of Named Entity Recognition (NER), which identifies mentions of (named) entities in a task, but without linking the mention with a KB identifier. The types of entities that the NER task should target were defined at the Message Understanding Conference 6 (MUC-6) (Grishman and Sundheim, 1996), specifically entities corresponding to types such as Person, Organization, Place and other Numerical/Temporal expressions. While this provided a consensus for evaluating NER systems, some authors noted that such a categorization is coarse-grained and proposed finer-grained classification mechanisms (Fleischman and Hovy, 2002).

In the context of EL, target Knowledge Bases will often contain entities from a wide variety of classes, including Movies, Products, Events, Laws, etc., not considered by the traditional NER definitions; a dataset such as Wikidata has around 50,000 entity classes, for example. As a result, class-based definitions of entities are restrictive. Hence some authors have proposed more general definitions for the EL task: as Ling et al. (2015) note, while some authors follow traditional NER definitions, others propose a looser definition that any KB identifier (e.g., Wikipedia article URL) can be the target for a link; they further note that within these definitions there is a lack of guidelines with respect to how EL datasets should be labeled and what sorts of mentions and links EL systems should (ideally) offer.

This ambiguity complicates research and applications relating to EL, as highlighted previously by various authors (Ling et al., 2015; Jha et al., 2017; Rosales-Méndez et al., 2018b). Figure 1 shows the results of a selection of popular online EL systems: Babelfy (Moro et al., 2014), DBpedia Spotlight (Mendes et al., 2011),



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 718–727, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

FRED (Gangemi et al., 2017) and TagME (Ferragina and Scaiella, 2010). We see significant differences in the annotations provided, which we argue are due not only to differing precision/recall of the systems, but also to differing policies on issues such as overlapping entities (should "Michael Jackson" be annotated within the documentary title), common/named entities (should "interview" be linked to the corresponding Wikipedia article `wiki:Interview`), and more besides. With such varying perspectives on the EL task, it is not clear how we should define gold standards that offer a fair comparison of tools (Ling et al., 2015).

The standard approach to tackle this issue has been to make certain design choices explicit, such as to enforce a particular policy with respect to overlapping mentions, or common entities, etc., when labeling an EL dataset or performing evaluation. However, the appropriate policy may depend on the particular application, setting, etc. In this paper we pursue an alternative approach, which is to embrace different perspectives of the EL task, proposing a fine-grained categorization of different types of EL mentions and links, and then re-annotating three existing datasets with respect to these categories, allowing us to compare the performance of EL tools within the different categories. Specifically, our contributions are as follows (indicating also the relevant section):

§ 2 We design and present the results of a questionnaire addressed to authors of EL papers intended to understand the consensus (or lack thereof) regarding the goals of the EL task.

§ 3 We argue that no one definition of an entity mention/link fits all, and hence propose a fine-grained categorization scheme for the EL task covering details regarding base form, part of speech, overlap, and reference type.

§ 4 We relabel three existing EL datasets – ACE2004 (subset), KORE50 and VoxEL – per our novel categorization scheme, extending the set of annotations as appropriate.

§ 5 We conduct a fine-grained evaluation of the performance of five EL systems with respect to individual categories of EL annotations.

§ 6 Addressing the lack of consensus, we propose a fuzzy recall and $F_1$ measure based on a configurable membership function, presenting results for the five EL systems.

In an [interview]$^{td}$ with [Martin Bashir]$^{btf}$ for the 2003 [documentary]$^{td}$ [Living with {Michael Jackson}$^{bd}$]$^{btf}$, the King of [Pop]$^{d}$ recalled that [Joe]$^{t}$ often sat with a white belt at hand as he and his four [siblings]$^{td}$ rehearsed.

Figure 1: Output annotations made by four different systems – Babelfy (b), TagME (t), DBpedia Spotlight (d) and FRED (f) – on the same input text.

§ 7 We present conclusions about the performance of the EL systems surveyed for different types of entity mentions/links and highlight open challenges for the EL task.

## 2 Questionnaire: Lack of Consensus

To understand what consensus (or lack thereof) exists within the EL community regarding the EL task, we designed a concise questionnaire with two sentences for which we proposed a variety of EL annotations, providing the text, the annotated text mentions, and proposed links to the respective Wikipedia articles. The two sentences – shown in Figure 2 (with a summary of results that will be described presently) – were designed to exemplify the types of design choices that vary from author to author (Ling et al., 2015); specifically, we target the following questions:

1. *Entity types*: should types not typically considered under MUC-6 definitions (other than as MISC) be linked (e.g., linking Living with Michael Jackson to the corresponding Wikipedia article)?

2. *Overlapping mentions*: should mentions inside other mentions be annotated (e.g., Michael Jackson)?

3. *Common entities*: should common nouns be annotated (e.g., documentary)?

4. *Parts of speech*: should only nouns be annotated or should other parts of speech also be linked (e.g., reports or white)?

5. *Mention types*: should complex types of mentions, such as pronouns (e.g., he) or descriptive noun phrases indicating named entities, be annotated (e.g., linking he and his four siblings to `wiki:The_Jackson_5`)?

In an [interview]$^{.19}$ with [Martin Bashir]$^{1}$ for the [2003]$^{.28}$ [documentary]$^{.28}$ [Living with {Michael Jackson}$^{.75}$]$^{.97}$, the [{King}$^{.08}$ of {Pop}$^{.33}$]$^{.94}$ [recalled]$^{.06}$ that [Joe]$^{1}$ often [sat]$^{.08}$ with a [white]$^{.11}$ [belt]$^{.14}$ at [hand]$^{.14}$ as [{he}$^{.56}$ and {his}$^{.39}$ {four}$^{.08}$ {siblings}$^{.14}$]$^{.50}$ [rehearsed]$^{.08}$.

[Russian]$^{.61}$ [daily]$^{.14}$ [Kommersant]$^{.97}$ [reports]$^{.06}$ that [Moscow]$^{.94}$ will [supply]$^{.06}$ the [Greeks]$^{.94}$ with [gas]$^{.36}$ at [{rock}$^{0}$ bottom {prices}$^{.19}$]$^{.28}$ as [Tsipras]$^{.92}$ [prepares]$^{.03}$ to [meet]$^{.06}$ the [{Russian}$^{.53}$ {President}$^{.12}$]$^{.97}$.

Figure 2: The sentences included in the questionnaire and the ratio of respondents who suggested to annotate the mentions. Multiple links were proposed for the mentions underlined (see Table 1).

6. *Link types*: should mentions link to the explicitly named entity (e.g., linking Moscow to `wiki:Moscow`), or should complex forms of reference such as *meronymy* (e.g., linking Moscow to `wiki:Government_of_Russia`), *hypernymy* (e.g., linking daily to `wiki:Newspaper` as the closest entity present in the KB, or linking Russian President to `wiki:Vladimir_Putin`), or *metaphor* (e.g., linking King to `wiki:King`) be considered?

For each sentence, respondents were asked to select the mentions and links that they consider an EL system should *ideally* provide; specifically, they were presented an optional multiple choice question for each mention: the option to select one (or more, in underlined cases) suggested links, and the option to not annotate the mention. In order to address this survey to the EL research community, we extracted the emails of all papers referenced in the recent survey by Wu et al. (2018) that are directly related to EL. We sent the questionnaire to 321 researchers, of which 232 requests were delivered successfully. We received a total of 36 responses. Aggregated responses are available online[1], where in Figure 2 we provide a summary of results, indicating in superscript the ratio of respondents who agreed to some link being provided for the given mention.

First we see that the only mentions that all 36 respondents agreed should be linked are Martin Bashir and Joe, which are traditional MUC types, non-overlapping, named entities, with direct mentions and links; on the other hand, there was also consensus that `rock` should not be linked. Otherwise, all other mentions showed some level of disagreement. Regarding our questions:

1. *Entity types*: per Living with Michael Jackson (0.97), the vast majority of respondents do not restrict to non-MISC MUC types.

2. *Overlapping mentions*: per Michael Jackson (0.75), most respondents also allow mentions contained in other mentions.

3. *Common entities*: most respondents do not consider common entities, with the highest response appearing for documentary (0.28).

4. *Parts of speech*: the consensus was mostly that EL should focus on nouns, with his (0.39) and white (0.11) being the most popularly selected non-nouns.

5. *Mention types*: here there was a notable split, with he (0.56) and he and his four siblings (0.5)[2] being selected for annotation by roughly half of the respondents.

6. *Link types*: for this we introduce Table 1, where we see how respondents preferred different types of reference (respondents could select multiple options); from this we conclude that although respondents preferred to use a country directly to represent nationality, they also preferred to resolve complex types of reference, such as the meronymic use of Moscow to refer to the government rather than the city, and the use of Putin's title to refer to him rather than the title itself.

So who is correct? We argue that there is no "correct" answer here. Common entities may, for example, rather be considered the target of a separate Word Sense Disambiguation task (Navigli, 2009), while pro-forms may be considered the target of a separate Coreference/Anaphora Resolution task (Sukthanker et al., 2018); these are matters of convention. In more practical terms, the

[1] `https://users.dcc.uchile.cl/~hrosales/questionnaire`

[2] One respondent noted that it was not certain that this referred to The Jackson 5.

Table 1: Links for mentions with multiple choices in Figure 2 and the ratio of respondents selecting that link

| Link | Ratio |
|---|---|
| [Russian] daily Kommersant ... | |
| wiki:Russia | 0.61 |
| wiki:Russians | 0.11 |
| wiki:Russian_language | 0.08 |
| ... that [Moscow] will supply ... | |
| wiki:Government_of_Russia | 0.77 |
| wiki:Moscow | 0.36 |
| ... supply the [Greeks] with gas ... | |
| wiki:Greece | 0.77 |
| wiki:Greeks | 0.36 |
| ... the [Russian] President. | |
| wiki:Russia | 0.42 |
| wiki:Russians | 0.19 |
| ... the [Russian President]. | |
| wiki:Vladimir_Putin | 0.77 |
| wiki:President_of_Russia | 0.61 |

importance of each individual EL annotation may vary from application to application; for example, for relation extraction, it may be important to identify all repeated mentions of an entity (as each such mention may lie within a distinct relation), while for semantic search having repeated mentions may be less critical (a subset of mentions may suffice to know the document is relevant for a given entity). We propose that no one definition of the EL task fits all, and rather propose a categorization that covers different perspectives.

## 3 Categorization Scheme

Inspired by the results of the questionnaire and related discussion by Ling et al. (2015), among other authors, in Figure 3 we propose a categorization scheme aiming to capture and organize these diverse perspectives on the EL task. This scheme makes explicit the types of *annotations* – mention–link pairs – that can be considered when annotating EL datasets, or when developing, evaluating and applying EL systems. The categorization scheme has four dimensions: *Base Form*, *Part of Speech*, *Overlap* and *Reference*. We propose that each EL annotation be labeled with precisely one leaf-node from each dimension. Our objective with this categorization is not to imply that all types of annotations *should* be considered as part of the EL task, but rather to map out the types of annotations that *could* be considered for EL.

### 3.1 Base Form

The *Base Form* considers the type of mention: is it a name, a common noun, a number, a date, etc.

We separate *Proper Noun* into more specific categories that deal with the difference between mentions and KB labels: *Full Name*, *Short Name*, *Extended Name* and *Alias*. The first three should be tagged when the mentions are equal (Michael Jackson), shorter (Jackson) or longer (Michael Joseph Jackson), respectively, than the primary label of their corresponding KB-entity (wiki:Michael_Jackson). On the other hand, *Alias* is used for mentions that vary from the primary label of the KB (King of Pop).

*Numeric/Temporal* are mentions that refer to a number or a given temporal concept (e.g., 1, 1900, first, October, July 07, Tuesday, 2018/01/32, etc.). Such mentions were included in the MUC-6 definition and some such expressions have corresponding Wikipedia articles. The next category is *Pro-form*, which includes any pronoun (*he*, *his*, etc.) referring to a named entity elsewhere. The final category is *Common Form*, which refers to a common word, such as documentary, sat, etc., or a noun-phrase referring to a named entity with a common head term, e.g., he and his four siblings.

### 3.2 Part of Speech

*Part of Speech* denotes the grammatical function of a word in a sentence, where we include four high-level classes for which we have found instances that could be linked to the Wikipedia KB (without referring to their own syntactic form). As we have already discussed, the most common target for EL is *Noun Phrases*; we divide these into *Singular* and *Plural* for additional granularity. However, we also include *Verb*s (e.g., divorcing, metastasized), *Adjective*s (e.g., anaerobic, French) and *Adverb*s (e.g., polynomially, Socratically) for linking to the Knowledge Base.

### 3.3 Overlap

The *Overlap* dimension, as its name suggests, captures the nature of overlap between mentions. As an example, for the mention New York City Police Museum, the mention New York would have *Minimal* overlap (assuming York is not incorrectly identified), New York City Police would have *Intermediate* overlap, and New York City Police Museum would have *Maximal* overlap.

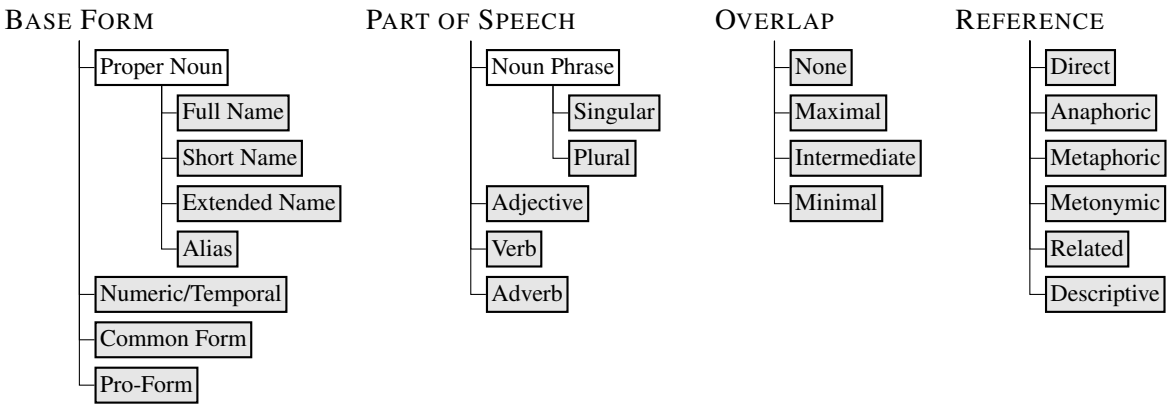


Figure 3: EL categorization scheme with concrete alternatives (leaf-nodes) shaded for each dimension

### 3.4 Reference

The *Reference* dimension considers the relation between the mention and the linked entity. The topic of *reference* is a complex one that has been the subject of much attention over many decades (Strawson, 1950); here we propose a pragmatic but comprehensive set of options.

The *Direct* category considers a direct explicit mention of an entity based on a known KB label/alias, such as M. Jackson or the King of Pop for `wiki:Michael_Jackson` or divorcing for `wiki:Divorce`. *Anaphoric* denotes a reference to an antecedent (or postcedent) for pro-forms such as he or her that are coreferent with a named entity. *Metaphoric* captures figurative references to entities whose characteristics are referred to, such as He is a pool [shark] linking to `wiki:Shark`. *Metonymic* indicates reference by common association, such as using Moscow to refer to `wiki:Government_of_Russia` or using Peru to refer to `wiki:Peru_national_football_team`. *Related* is used when only a near-synonym, hypernym or hyponym is available for a mention in the KB, such as the Russian [daily] being linked to `wiki:Newspaper`.[3] Finally, *Descriptive* is used for (non-pro, non-proper) referring noun-phrases, such as his father referring to `wiki:Joe_Jackson`; unlike similar *Anaphoric* references, *Descriptive* references do not necessarily rely on an antecedent to be disambiguated, as per the case of Hendix's band referring to `wiki:The_Jimi_Hendrix_Experience` or Fiji's capital referring to `wiki:Suva` without requiring further context to disambiguate.

[3] Wikipedia will often redirect from a term to a related article; for example, `wiki:Daily_newspaper` currently redirects to `wiki:Newspaper`.

## 4 Fine-Grained Datasets

A wide variety of EL datasets have been proposed (e.g., ACE2004 (Ratinov et al., 2011), KORE50 (Hoffart et al., 2012), MEANTIME (Minard et al., 2016), DBpedia Spotlight (Mendes et al., 2011), VoxEL (Rosales-Méndez et al., 2018a)) to support EL quality measurement. However, often the guidelines followed in the annotation process are left implicit, such as the inclusion/exclusion of overlapping mentions, common entities, etc. Furthermore, different types of entities are not distinguished.

To put our categorization into practice, we select three existing datasets – KORE50, ACE2004, and VoxEL – and categorize each annotation. We further add novel annotations not considered in the original labeling process with the goal of capturing as many potential (categorized) links to Wikipedia articles as possible. The process of annotation was done manually by three authors (with the help of the NIFify tool (Rosales-Méndez et al., 2019) for labeling overlaps, parts of speech, as well as validation). The annotation process was iterative. The first author began an initial annotation based on a strict and relaxed notion of "entity", with strict referring to classical definitions of entities, and relaxed referring to any entity mention linkable with Wikipedia (following the literature). This process raised ambiguities regarding metonymic references, descriptive noun phrases, etc. Hence the authors defined fine-grained categories to address ambiguous cases and designed the questionnaire to better understand the community consensus. The first author relabeled the data per the fine-grained categorization, with semi-automated verification. The other authors then revised these annotations in detail; there were significant dif-

ferences, leading to further discussion and refinements in the categorization. Ambiguities and disagreements were iteratively resolved by the authors through discussion and modification of the categories. The resulting datasets reflect the consensus of the authors. The categorization scheme shown in Figure 3 is also a result of this process, where we iteratively extended and refined these categories as we encountered specific cases in these datasets; as a result, these categories suffice to cover all cases of possible mentions of Wikipedia entities in the datasets.

The labeling process was extremely time consuming (taking place over six months) due to the large number of annotations generated, particularly for common-form mentions; for this reason, in the case of ACE2004, we only annotate 20 documents (from a total of 57) wherein, for example, the number of annotations increases from 108 in the original data to 3,351 in our fine-grained version[4]. Per MUC-6, we also include emerging/not-in-lexicon entities and entity types in our annotations. Additionally, we label some mentions with multiple alternatives (per Table 1).

In Table 2 we summarize details of the reconstructed version of these three datasets, including the number of documents, sentences and annotations. Except for the case of ACE2004, our re-annotated datasets maintain the same text collection as in the original version. For each category, we also show the number of annotations tagged with it. The most frequent annotations are those that belong to *Common Form*, *Singular Noun*, *Non-Overlapping* and *Direct* categories.

In the final iteration, we perform validation checking for erroneous links avoiding invalid, redirect and disambiguation pages; erroneous categorizations, e.g., multiple tags in the same category; erroneous mentions, such as trailing spaces; etc. During this process, we also encountered and resolved numerous issues with the original datasets; of note, in ACE2004 we found various spelling errors of entity names, e.g., Stewart Talbot as a misspelling of Strobe Talbott, Coral Islands as a misspelling of Kuril Islands, etc.

## 5 Fine-Grained Evaluation

Our fine-grained datasets allow us to understand the performance of EL systems in more detail regarding different types of EL annotations. In Table 3, we present the Precision (**P**), Recall (**R**) and $F_1$ score ($\mathbf{F}_1$) for five popular EL systems with online APIs: Babelfy (**B**), TagME (**T**), DBpedia Spotlight (**D**), AIDA (**A**) and FREME (**F**). In the case of Babelfy, we consider both settings: *strict*, which excludes common entities ($\mathbf{B}_s$); and *relaxed*, which includes common entities ($\mathbf{B}_r$). Results are shown for subsets of annotations corresponding to a particular category ($A$), where we also present the number of unique mentions for annotations of that category ($|A|$). Recall that our gold standard may have multiple annotations for a single mention in the gold standard, listing different possible links for an individual mention (see, e.g., Table 1); on the other hand, evaluated systems predict a single link for each mention. We thus evaluate annotations on a mention-by-mention basis.[5] We consider a predicted mention to be a *true positive* if it is included in $A$ and the predicted link

Table 2: Content of relabeled datasets

| | KORE50 | ACE2004 | VoxEL |
|---|---|---|---|
| Documents | 1 | 20 | 15 |
| Sentences | 50 | 214 | 94 |
| Annotations | 372 | 3,351 | 1,107 |
| Full Name | 41 | 588 | 227 |
| Short Name | 114 | 307 | 97 |
| Extended Name | 1 | 8 | – |
| Alias | 5 | 94 | 15 |
| Numeric/Temporal | 17 | 276 | 111 |
| Common Form | 157 | 1,974 | 615 |
| Pro-form | 37 | 107 | 42 |
| Singular Noun | 248 | 1,943 | 683 |
| Plural Noun | 39 | 670 | 182 |
| Adjective | 45 | 501 | 149 |
| Verb | 40 | 232 | 85 |
| Adverb | – | 5 | 8 |
| No Overlap | 307 | 2,161 | 792 |
| Maximal Overlap | 23 | 392 | 95 |
| Intermediate Overlap | 4 | 62 | 14 |
| Minimal Overlap | 38 | 736 | 206 |
| Direct | 262 | 2,280 | 750 |
| Anaphoric | 37 | 107 | 42 |
| Metaphoric | 8 | 27 | 38 |
| Metonymic | 3 | 60 | 21 |
| Related | 54 | 698 | 224 |
| Descriptive | 8 | 179 | 32 |
| Person | 117 | 278 | 66 |
| Organisation | 40 | 199 | 120 |
| Place | 19 | 519 | 168 |
| Miscellany | 196 | 2,352 | 753 |

[4] https://github.com/henryrosalesmendez/categorized_EMNLP_datasets

[5] While a mention can only appear in one PART-OF-SPEECH and OVERLAP category, it can appear in multiple BASE FORM or REFERENCE categories for alternative links.

Table 3: Results for Babelfy (strict/relaxed), TagME, DBpedia Spotlight, AIDA and FREME on the unified dataset.

| | | $B_s$ | | | $B_r$ | | | T | | | D | | | A | | | F | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\|A\|$ | P | R | $F_1$ | P | R | $F_1$ | P | R | $F_1$ | P | R | $F_1$ | P | R | $F_1$ | P | R | $F_1$ |
| Full Mention | 766 | 0.93 | 0.46 | 0.61 | 0.75 | 0.53 | 0.62 | 0.82 | 0.59 | 0.69 | 0.84 | 0.57 | 0.68 | 0.78 | 0.57 | 0.66 | 0.82 | 0.55 | 0.65 |
| Short Mention | 497 | 0.44 | 0.16 | 0.23 | 0.37 | 0.24 | 0.29 | 0.54 | 0.44 | 0.48 | 0.5 | 0.3 | 0.37 | 0.5 | 0.36 | 0.42 | 0.39 | 0.28 | 0.33 |
| Extended Mention | 9 | 1.0 | 0.56 | 0.71 | 0.83 | 0.56 | 0.67 | 1.0 | 0.44 | 0.62 | 1.0 | 0.44 | 0.62 | 1.0 | 0.44 | 0.62 | 0.8 | 0.44 | 0.57 |
| Alias | 112 | 0.56 | 0.16 | 0.25 | 0.33 | 0.21 | 0.25 | 0.52 | 0.32 | 0.4 | 0.67 | 0.38 | 0.48 | 0.6 | 0.29 | 0.4 | 0.55 | 0.29 | 0.38 |
| Numeric/Temporal | 404 | 0.45 | 0.01 | 0.02 | 0.82 | 0.24 | 0.37 | 0.14 | 0.03 | 0.05 | 0 | 0.0 | 0.0 | 0 | 0.0 | 0.0 | 0 | 0.0 | 0.0 |
| Common Form | 2,452 | 0.21 | 0.0 | 0.01 | 0.66 | 0.33 | 0.44 | 0.49 | 0.28 | 0.35 | 0.88 | 0.04 | 0.08 | 0.43 | 0.0 | 0.0 | 0.56 | 0.0 | 0.01 |
| Pro-form | 153 | 0 | 0.0 | 0.0 | 0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0.0 | 0.0 | 0 | 0.0 | 0.0 | 0 | 0.0 | 0.0 |
| Singular Noun | 2,623 | 0.79 | 0.17 | 0.28 | 0.73 | 0.45 | 0.56 | 0.62 | 0.38 | 0.47 | 0.87 | 0.24 | 0.38 | 0.79 | 0.2 | 0.32 | 0.74 | 0.19 | 0.31 |
| Plural Noun | 746 | 0.33 | 0.01 | 0.02 | 0.61 | 0.33 | 0.43 | 0.56 | 0.28 | 0.37 | 0.83 | 0.03 | 0.06 | 0.7 | 0.03 | 0.07 | 0.66 | 0.04 | 0.07 |
| Adjective | 516 | 0.77 | 0.02 | 0.04 | 0.26 | 0.07 | 0.11 | 0.56 | 0.24 | 0.34 | 0.65 | 0.14 | 0.23 | 0.72 | 0.21 | 0.32 | 0.6 | 0.14 | 0.22 |
| Verb | 334 | 0 | 0.0 | 0.0 | 0.86 | 0.02 | 0.04 | 0.37 | 0.17 | 0.23 | 1.0 | 0.0 | 0.01 | 0 | 0.0 | 0.0 | 0 | 0.0 | 0.0 |
| Adverb | 12 | 0 | 0.0 | 0.0 | 0 | 0.0 | 0.0 | 0.56 | 0.42 | 0.48 | 0 | 0.0 | 0.0 | 0 | 0.0 | 0.0 | 0 | 0.0 | 0.0 |
| Non-Overlapping | 2,871 | 0.75 | 0.12 | 0.2 | 0.67 | 0.33 | 0.45 | 0.58 | 0.38 | 0.46 | 0.84 | 0.19 | 0.32 | 0.78 | 0.19 | 0.3 | 0.71 | 0.17 | 0.27 |
| Maximal Overlap | 464 | 0.87 | 0.17 | 0.29 | 0.85 | 0.36 | 0.5 | 0.73 | 0.34 | 0.46 | 0.89 | 0.19 | 0.32 | 0.84 | 0.08 | 0.15 | 0.84 | 0.12 | 0.22 |
| Intermediate Overlap | 71 | 0.76 | 0.18 | 0.3 | 0.71 | 0.52 | 0.6 | 0.57 | 0.3 | 0.39 | 0.56 | 0.13 | 0.21 | 0.54 | 0.1 | 0.17 | 0.78 | 0.1 | 0.17 |
| Minimal Overlap | 825 | 0.82 | 0.04 | 0.09 | 0.61 | 0.37 | 0.46 | 0.5 | 0.15 | 0.23 | 0.8 | 0.09 | 0.17 | 0.72 | 0.09 | 0.16 | 0.66 | 0.06 | 0.12 |
| Direct | 3,106 | 0.79 | 0.13 | 0.23 | 0.71 | 0.43 | 0.53 | 0.63 | 0.38 | 0.47 | 0.83 | 0.21 | 0.33 | 0.76 | 0.19 | 0.3 | 0.7 | 0.17 | 0.27 |
| Anaphoric | 153 | 0 | 0.0 | 0.0 | 0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0.0 | 0.0 | 0 | 0.0 | 0.0 | 0 | 0.0 | 0.0 |
| Metaphoric | 69 | 0.0 | 0.0 | 0.0 | 0.57 | 0.29 | 0.38 | 0.43 | 0.35 | 0.38 | 0.91 | 0.14 | 0.25 | 0 | 0.0 | 0.0 | 0 | 0.0 | 0.0 |
| Metonymic | 73 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| Related | 829 | 0.64 | 0.06 | 0.11 | 0.36 | 0.14 | 0.2 | 0.39 | 0.24 | 0.3 | 0.76 | 0.1 | 0.18 | 0.81 | 0.08 | 0.15 | 0.83 | 0.09 | 0.16 |
| Descriptive | 189 | 0.33 | 0.01 | 0.01 | 0.44 | 0.02 | 0.04 | 0.16 | 0.02 | 0.03 | 0.6 | 0.02 | 0.03 | 0 | 0.0 | 0.0 | 0.6 | 0.02 | 0.03 |
| All | 4,231 | 0.77 | 0.11 | 0.19 | 0.67 | 0.35 | 0.46 | 0.59 | 0.33 | 0.42 | 0.84 | 0.17 | 0.29 | 0.77 | 0.16 | 0.26 | 0.72 | 0.14 | 0.24 |

corresponds to any of the possible links given by $A$ for that mention. We consider a predicted mention to be a *false positive* if the mention is given by $A$, but the predicted link is not given by $A$ for that mention; in other words, when restricting $A$ by category, system annotations on mentions outside of $A$ are ignored.[6] Finally we consider a mention to be a *false negative* if it is given by $A$ but either the mention is not predicted by the system or the system predicts a link not given by $A$ for tht mention. We also show the overall result in the final row considering the full gold standard. The results consider a unified dataset that concatenates all three datasets. Better results (closer to 1) are shaded darker to aid with visual comparison.

Comparing the different types of annotations, on a high-level, we can see that recall, in particular, varies widely across categories; unsurprisingly perhaps, systems, in general, exhibit higher recall for proper nouns (particularly full and extended names) with direct references. Looking at categories with poor results, we see that none of the evaluated systems consider anaphoric references, perhaps considered as a task distinct from EL. Interestingly, no system captures metonymic references, though as previously seen the results of our questionnaire (see Table 1) indicate that respondents prefer such types of links over their literal counterparts (e.g., linking Moscow in the given context with `wiki:Government_of_Russia` rather than `wiki:Moscow`). Comparing systems, Babelfy (relaxed) and TagME achieve much higher recall for common forms than the other systems: we attribute this to these systems making the design choice to additionally support common entities.

While the previous results consider dimensions independently, there are $7 \times 5 \times 4 \times 6 = 840$ possible combinations across the four dimensions; not all of these can occur (for example, a *Pro-form* mention requires *Anaphoric* reference). In the unified dataset, we found 123 combinations to have at least one annotation. In order to understand in more detail how the systems perform for annotations in combined categories, for the six system configurations, Figure 4 presents a best-first accumulative progression of Precision, Recall and $F_1$: we start with the combined category in which each system performs best ($x = 1$), adding tags in the next-best combined category until all annotations in the gold standard are considered. We see that although precision remains relatively high throughout the gold standard, recall drops considerably as combined categories on which there is less consensus are added. The recall and $F_1$ measures, in particular, present a clear division in the systems: Babelfy (relaxed) and TagME maintain a higher recall as more combined categories are considered due to their inclusion of common entities (but suffer from reduced precision as a result).

[6] If a system predicts a link present in the gold standard for the mention but with a different category than tested, it will thus be considered a false positive for the tested category.

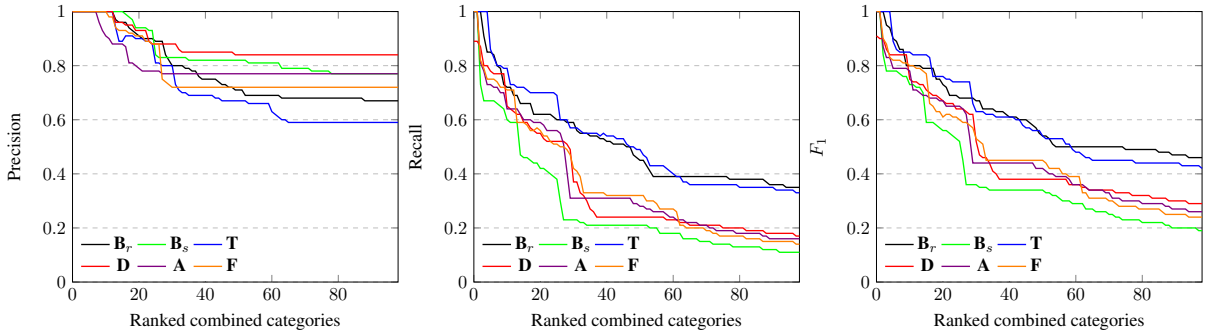


Figure 4: Cumulative results for Babelfy (relaxed/strict), TagME, DBpedia Spotlight, AIDA and FREME for the unified dataset over ranked combinations of categories

## 6 Fuzzy Recall and $F_1$ Measures

While the previous results provide insights into different design choices taken by different systems with respect to different types of EL annotations, still, per Table 1, the results are perhaps *too* fine-grained. For the purposes of comparing the performance of systems, we may rather prefer a measure that aggregates performance across different categories. However, this presents a conceptual problem: for some applications or definitions of the EL task, certain categories may be considered more important than others. For instance, should we penalize a system equally for missing the annotations gas and Tspiras in the second sentence of Figure 2? In most EL settings, the former is arguably less important than the latter, but nonetheless it may depend on the setting.

Reflecting the lack of consensus for the EL task, we rather propose a measure inspired by Fuzzy Set Theory (Zadeh, 1965): given a universe of elements $U$ and a particular element $x \in U$, rather than considering traditional *crisp* sets $A$ with binary membership (where $x \in A$ or $x \notin A$), a fuzzy set $A^*$ is associated with a membership function $\mu_{A^*} : U \rightarrow [0, 1]$, which denotes the degree to which an element $x$ is a member of $A^*$ (given by $\mu_{A^*}(x)$). Given that a crisp set $A$ can be defined as a fuzzy set with membership function $\mu_A : U \rightarrow \{0, 1\}$, fuzzy sets are a generalization of crisp sets. Intuitively, we can then consider a gold standard with a fuzzy set of annotations $A^*$, where (e.g.) annotations forming part of the core consensus of EL have a higher membership degree than those for which consensus does not exist; different membership degrees can also be applied for evaluation in different application settings.

Formally, an annotation is a triple $a = (o, o', l)$, where $o$ and $o'$ denotes the start and end offset of a mention in a text ($o < o'$), and $l$ denotes a link (a KB identifier or a not-in-lexicon string). For a given text, a gold standard $G$ is a set of annotations, as is the result of a system $S$. The set of *true positives* is defined as $TP = G \cap S$, *false positives* as $FP = S - G$, and *false negatives* as $FN = G - S$. In this case, however, while we still consider $S$ to be a crisp set, we allow a fuzzy version of the gold standard $G^*$ with $\mu_{G^*} : G \rightarrow [0, 1]$.[7] In practice, for a given annotation $a \in G$, we propose that $\mu_{G^*}(a)$ is a function of the categorization for $a$; for example, with reference to Figure 2, we may consider that common forms have a lower degree of membership than proper forms. We are left to define Precision, Recall and $F_1$ measures for $S$ with respect to $G^*$.

For a given system result $S$, gold standard $G$ and its fuzzy version $G^*$, we propose that precision be computed in the traditional way for the crisp version of the gold standard – $P = \frac{|TP|}{|S|}$ – with the intuition that false positives proposed by the system (type I error) be weighted equally: if the system proposes an annotation, it should be correct, independently of the type of annotation. On the other hand, a gold standard annotation *not* proposed by the system may be due to different design choices; we hence propose to use a fuzzy recall measure with respect to $G^*$, namely $R^* = \frac{\sum_{a \in S} \mu_{G^*}(a)}{\sum_{a \in G} \mu_{G^*}(a)}$, thus applying different costs for missing annotations (type II errors) depending on the annotation in question. We then define the fuzzy $F_1$ measure in the natural way: $F_1^* = \frac{2 \cdot P \cdot R^*}{P + R^*}$. The following properties can be verified for $R^*$ and $F_1^*$:

- PROP1: the values for $R^*$ and $F_1^*$ both range between 0 and 1, inclusive.

[7] For annotations $a \notin G$, we assume $\mu_{G^*}(a) = 0$.

- PROP2: when $\mu_{G^*} : G \rightarrow \{1\}$ (i.e., when memberships are binary), $R^*$ and $F_1^*$ correspond to $R$ and $F_1$.

- PROP3: missing annotations with higher membership degree are penalized more in $R^*$ and $F_1^*$ than those with lower degree.

The definition of the membership function $\mu_{G^*}$ is then dependent on the setting. Here we define an instance of $\mu_{G^*}$ based on the questionnaire results shown in Figure 2. Specifically, we consider annotations with *Proper Forms*, *Noun Phrases*, *No Overlap* and *Direct Reference* – which consistently score greater than 0.9 in Figure 2 – to have a membership degree of 1; we consider these to be *strict* annotations. We assign all other annotations – which we call *relaxed* annotations – a constant membership degree of $\alpha$, where higher values of $\alpha$ place more importance on achieving relaxed annotations; when $\alpha = 0$, relaxed false negatives are not punished; when $\alpha = 1$, both strict and relaxed false negatives are weighted equally.

Finally, the gold standard may offer multiple alternative links for a mention while the evaluated systems predict one link per mention. We apply the same procedure outlined previously: checking for each mention that the predicted link matches one of the alternatives in the gold standard. In the case of $R^*$, the membership score for a mention in $G^*$ is given as the maximum membership score over all annotations/links for that mention in $G^*$; e.g., if a system predicts a link for a mention with weight $\alpha$ in $G^*$ but there exists another link for that mention with weight 1 in $G^*$, the system will score $\frac{\alpha}{\max\{1,\alpha\}} = \alpha$ for that mention in $R^*$.

The $F_1^*$ results are shown in Figure 5, where we again can distinguish the systems that link common entities – Babelfy (relaxed) and TagME – from those that do not; the former group of systems performs worse for stricter definitions of EL annotations, but outperform other systems as the definition is relaxed.[8] The raw data for these experiments can be found online.[9]

## 7 Conclusions

We have (i) presented the results of a questionnaire that assesses consensus on the goals of the EL task, (ii) proposed a fine-grained categorization scheme for EL annotations, (iii) comprehensively (re-)labeled three datasets accordingly to this scheme, (iv) presented results for five EL systems with respect to annotations in different categories, and (v) proposed fuzzy recall/$F_1$ measures to address the lack of consensus, presenting results for the five systems on a strict/relaxed spectrum.

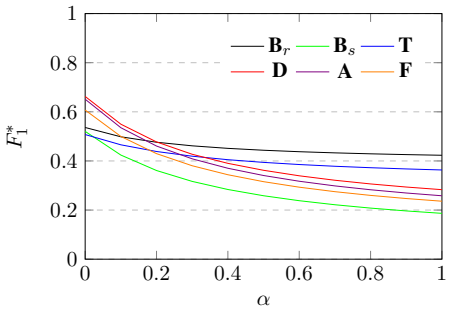


Figure 5: $\alpha$-based fuzzy $F_1$ scores

Our main conclusions are as follows:

- Though there is consensus on some EL annotations, opinions differ for common entities, pro-forms, descriptive references, etc.

- The EL systems tested offer little or no support for pro-forms, meronymic references, referencing noun phrases, etc., despite there being considerable support for such EL annotations in the questionnaire.

- Our fine-grained evaluation distinguishes two groups of EL systems: one group targeting common entities and named entities, the other group focused on named entities.

The results of our questionnaire and system evaluation suggest the need for future work on supporting complex forms of reference within EL systems; the datasets we provide can be used to evaluate such approaches. Another important direction is to either reach a consensus on the EL task (perhaps in a similar style to MUC-6 for NER), or define protocols for evaluation in the absence of such a consensus; our fuzzy recall/$F_1$ metrics are concrete steps in this direction.

**Acknowledgements** The work of Henry Rosales-Méndez was supported by CONICYT-PCHA/Doctorado Nacional/2016-21160017. The work was also supported by the Millennium Institute for Foundational Research on Data (IMFD) and by Fondecyt Grant No. 1181896.

[8] We remark that $P$ is agnostic to $\alpha$ and would result in a straight line; $R^*$ thus follows the same trend as $F_1^*$.

[9] https://github.com/henryrosalesmendez/EL_exp

## References


Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In SIGMOD, pages 1247–1250.

Paolo Ferragina and Ugo Scaiella. 2010. TAGME: on-the-fly annotation of short text fragments (by wikipedia entities). In CIKM, pages 1625–1628.

Michael Fleischman and Eduard H. Hovy. 2002. Fine Grained Classification of Named Entities. In COLING.

Aldo Gangemi, Valentina Presutti, Diego Reforgiato Recupero, Andrea Giovanni Nuzzolese, Francesco Draicchio, and Misael Mongiovì. 2017. Semantic Web Machine Reading with FRED. Semantic Web, 8(6):873–893.

Ralph Grishman and Beth Sundheim. 1996. Message Understanding Conference- 6: A Brief History. In COLING, pages 466–471.

Johannes Hoffart, Stephan Seufert, Dat Ba Nguyen, Martin Theobald, and Gerhard Weikum. 2012. KORE: keyphrase overlap relatedness for entity disambiguation. In CIKM, pages 545–554.

Kunal Jha, Michael Röder, and Axel-Cyrille Ngonga Ngomo. 2017. All that Glitters Is Not Gold - Rule-Based Curation of Reference Datasets for Named Entity Recognition and Entity Linking. In ESWC, pages 305–320.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. 2015. DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia. Semantic Web, 6(2):167–195.

Xiao Ling, Sameer Singh, and Daniel S. Weld. 2015. Design challenges for entity linking. TACL, 3:315–328.

Pablo N. Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. 2011. DBpedia Spotlight: shedding light on the web of documents. In I-SEMANTICS, pages 1–8.

Anne-Lyse Minard, Manuela Speranza, Ruben Urizar, Begoña Altuna, Marieke van Erp, Anneleen Schoen, and Chantal van Son. 2016. MEANTIME, the NewsReader Multilingual Event and Time Corpus. In LREC.

Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity Linking meets Word Sense Disambiguation: a Unified Approach. TACL, 2:231–244.

Roberto Navigli. 2009. Word sense disambiguation: A survey. ACM Comput. Surv., 41(2):10:1–10:69.

Lev-Arie Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and Global Algorithms for Disambiguation to Wikipedia. In ACL, pages 1375–1384.

Thomas Rebele, Fabian M. Suchanek, Johannes Hoffart, Joanna Biega, Erdal Kuzey, and Gerhard Weikum. 2016. YAGO: A Multilingual Knowledge Base from Wikipedia, Wordnet, and Geonames. In ISWC, pages 177–185.

Henry Rosales-Méndez, Aidan Hogan, and Barbara Poblete. 2018a. VoxEL: A Benchmark Dataset for Multilingual Entity Linking. In ISWC, pages 170–186.

Henry Rosales-Méndez, Aidan Hogan, and Barbara Poblete. 2019. NIFify: Towards Better Quality Entity Linking Datasets. In WWW Companion Volume, pages 815–818.

Henry Rosales-Méndez, Barbara Poblete, and Aidan Hogan. 2018b. What should Entity Linking link? In AMW.

Peter F. Strawson. 1950. On referring. Mind, 59(235):320–344.

Rhea Sukthanker, Soujanya Poria, Erik Cambria, and Ramkumar Thirunavukarasu. 2018. Anaphora and coreference resolution: A review. CoRR, abs/1805.11824.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. Commun. ACM, 57(10):78–85.

Jörg Waitelonis, Henrik Jürges, and Harald Sack. 2016. Don't compare apples to oranges: Extending GERBIL for a fine grained NEL evaluation. In SEMANTICS, pages 65–72.

Gong-Qing Wu, Ying He, and Xuegang Hu. 2018. Entity linking: An issue to extract corresponding entity with knowledge base. IEEE Access, 6:6220–6231.

Lotfi A Zadeh. 1965. Fuzzy sets. Information and control, 8(3):338–353.

# Supervising Unsupervised Open Information Extraction Models

**Arpita Roy[1], Youngja Park[2], Taesung Lee[2] and Shimei Pan[1]**
[1]University of Maryland, Baltimore County
Baltimore, MD 21250, USA
{arpita2, shimei}@umbc.edu
[2] IBM T. J. Watson Research Center
Yorktown Heights, New York, USA
{young_park, taesung.lee}@ibm.com

## Abstract

We propose a novel supervised open information extraction (Open IE) framework that leverages an ensemble of unsupervised Open IE systems and a small amount of labeled data to improve system performance. It uses the outputs of multiple unsupervised Open IE systems plus a diverse set of lexical and syntactic information such as word embedding, part-of-speech embedding, syntactic role embedding and dependency structure as its input features and produces a sequence of word labels indicating whether the word belongs to a relation, the arguments of the relation or irrelevant.

Comparing with existing supervised Open IE systems, our approach leverages the knowledge in existing unsupervised Open IE systems to overcome the problem of insufficient training data. By employing multiple unsupervised Open IE systems, our system learns to combine the strength and avoid the weakness in each individual Open IE system. We have conducted experiments on multiple labeled benchmark data sets. Our evaluation results have demonstrated the superiority of the proposed method over existing supervised and unsupervised models by a significant margin.

## 1 Introduction

Open Information Extraction (Open IE) extracts textual tuples consisting of a relation phrase and argument phrases from a sentence (Banko et al., 2007). Open IE was introduced as an alternative to traditional supervised information extraction (IE) method to address two major limitations of supervised approaches. First, supervised IE relies heavily on labeled training data. Since manual relation annotation is very expensive, this method does not scale to a large number of relations and is very difficult to adapt to new domains. Second, supervised IE systems require the target relations to be predetermined and learn to extract only the predefined relations. Therefore, they miss new and potentially meaningful domain relations that are prominent in a given dataset.

In contrast, Open IE operates in a completely domain-independent manner and is suitable when the target relations are not known in advance. Recently, Open IE has gained much attention, and various Open IE tools have been developed (Fader et al., 2011; Mausam et al., 2012; Akbik and Löser, 2012; Corro and Gemulla, 2013; Pal and Mausam, 2016; Yu et al., 2017; Kadry and Dietz, 2017; Roth et al., 2018; Stanovsky et al., 2018). Many Open IE systems demonstrated that they can scale to massive open-domain corpora such as the Web and Wikipedia (Banko et al., 2007), and the extracted tuples can be used as intermediate representation for various downstream NLP tasks such as knowledge base population (Soderland et al., 2010), question answering (Fader et al., 2014; Khot et al., 2017) and event schema induction (Mausam, 2016a).

Typically, these systems read in one sentence at a time and extract tuples with a relation phrase and one or more arguments. Most Open IE systems extract binary relations using domain-independent syntactic and lexical constraints. However, systems specialized in other syntactic constructions were also developed, such as noun-mediated relations (Pal and Mausam, 2016), n-ary relations (Akbik and Löser, 2012), nested propositions (Bhutani et al., 2016) and numerical Open IE (Saha et al., 2017a). Further, in recent years, there have been efforts to create a supervised Open IE system. (Stanovsky and Dagan, 2016) constructed an annotated corpus for Open IE, and (Stanovsky et al., 2018) and (Cui et al., 2018) used the annotated data to build a supervised Open IE system by formulating Open IE as sequence tagging and generation problems respectively.

However, while most existing Open IE systems



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 728–737, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

extract verbal relations, each of the systems focuses on different relational structures and extraction rules, resulting in heterogeneous results. Table 1 shows the different extraction results from the same sentence by three different Open IE systems. These variations makes it hard to compare different Open IE systems and select one for a new task, given their different strengths and weaknesses. This observation motivates us to explore an ensemble model which can learn from multiple existing Open IE systems which performs better than the underlying systems. This is especially attractive as no retraining or customization is needed to apply multiple existing Open IE systems.

In this paper, we propose a new Open IE method employing an ensemble of multiple unsupervised Open IE methods and a manually annotated data set. Similarly to (Stanovsky et al., 2018), we define Open IE as a sequence tagging problem and classify each word if it is a part of a relation, arguments or none. We first run several existing IE systems on the labeled data and use their extraction results as input features along with other rich features including word embedding, part-of-speech embedding, syntactic role embedding and syntactic dependency information. The model is then trained using the labeled data.

In this paradigm, our model can enjoy the advantages of both unsupervised Open IE approach and labeled data, since our model can learn the combined knowledge of the Open IE systems as well as optimized according to the labeled data. Evaluations with several benchmark datasets and Open IE systems show that our method outperforms the baseline systems by a large margin validating our hypothesis.

## 2 Supervised Ensemble of Open IE

In this work, we propose a new Open IE paradigm, a supervised ensemble of Open IE (*SenseOIE*). While there are many existing Open IE systems, each of the systems provides unique extraction rules and supports different relation constructs. This results in no single clear winner of the existing methods, when one tries to apply Open IE to a new data set. Rather, it would be more beneficial to apply multiple OpenIE systems and combine the wisdoms of all the systems. In this work we propose supervised ensemble of three different IE systems. These systems are Stanford Open IE(Angeli et al., 2015), OpenIE 5[1] and UKG (a private Open IE tool). Stanford Open IE is a dependency parser based system that uses handcrafted patterns to extract a predicate-argument triple from a sentence. On the other hand OpenIE 5 can extract verbal relation, nominal relation, relation with numeric argument and relation from consecutive sentences. UKG extracts verbal binary relations based on noun phrase detection, named entity recognition and dependency parsing. All three of these Open IE systems have different extraction rules and patterns focusing on extracting different relation tuples. Stanford Open IE, OpenIE 5 and UKG can complement each other when combined together. As a result, by ensembling these IE systems, our proposed system SenseOIE achieves better and larger coverage of all possible relation extractions. We utilize a small amount of labeled data to further optimize the model that can produce higher quality tuples.

### 2.1 System Overview

In this work, we consider extraction of binary relations from sentences. Let us consider an input sentence $S$. The goal of our system is to extract a set of relation tuples $T$ from $S$, where $T = \{T_1, T_2, \ldots, T_n\}$ and the $i$-th tuple $T_i$ consists of $< e_{i1}, r_i, e_{i2} >$, where $r_i$ is the relation phrase of $T_i$ and $e_{i1}$ and $e_{i2}$ are the first and second arguments of $r_i$. We frame this task as a sequence tagging, and the model annotates each word in the sentence to $E1$, $E2$, $R$ or $O$ ($EOR$ tags). $E1$ and $E2$ denote the first and the second arguments, $R$ is the relation, and $O$ represents all other words. Figure 1 shows a system overview of *SenseOIE*.

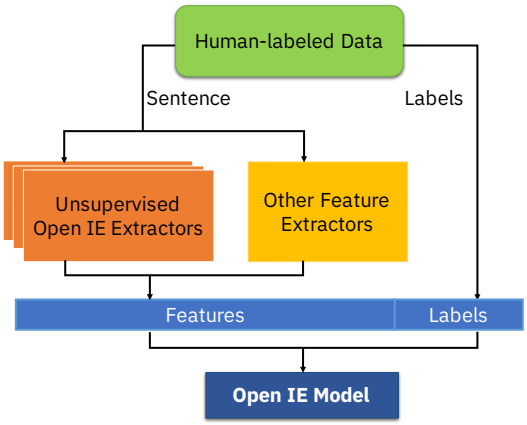


Figure 1: System Overview of *SenseOIE*

[1] http://openie.allenai.org

| Input | Evil Corp has released a new improved variant of the Dridex trojan that was spread through Andromeda botnet. |
|---|---|
| OpenIE5 | 1. (the Dridex trojan; was spread; )<br>2. (Evil Corp ; has released; new variant of the Dridex trojan) |
| Stanford OpenIE | 1. (Evil Corp; has released; variant) 2. (Evil Corp; has released; variant of Dridex trojan)<br>3. (Evil Corp; has released; new variant) 4. (Evil Corp; has released; new variant of Dridex trojan) |
| UKG | 1. (Evil Corp; has released; a new improved variant of the Dridex trojan)<br>2. (new improved variant of the Dridex trojan; was spread through; Andromeda botnet) |

Table 1: Extracted tuples by different OpenIE systems for an input sentence

### 2.2 Features

For an input to our system, we extract features for each word in the corpus. We first collect the Open IE system outputs by 1) running the existing Open IE systems as a black-box on the labeled corpus, 2) mapping the extracted tuples back to the original input sentence, and 3) assigning the $EOR$ tags to each word based on the outputs of each Open IE system. This gives us $k$ $EOR$ tags for each word from $k$ Open IE tools (*e.g.*, 'E, E, O' by three Open IE systems). In addition to the Open IE results, we extract part-of-speech (pos) tags, syntactic role and dependency parse tree.

In particular, we consider dependency parse tree based on the one-hop neighbors (*i.e.*, parent and children) of a word in the dependency tree. We use $\text{parent}(w_i)$, the parent of word $w_i$, and $\text{left-child}(w_i)$ and $\text{right-child}(w_i)$, the closest left and right children of $w_i$.

Formally, given an input sentence $S$, we extract a feature vector $\mathcal{F}(w_i)$ for each word $w_i \in S$ defined as follows:

$$\begin{aligned}\mathcal{F}(w_i) =& emb(w_i) \oplus \mathcal{F}_B(w_i) \\ &\oplus \mathcal{F}_B(\text{parent}(w_i)) \\ &\oplus \mathcal{F}_B(\text{left-child}(w_i)) \\ &\oplus \mathcal{F}_B(\text{right-child}(w_i))\end{aligned}$$

where $\oplus$ denotes concatenation, $\mathcal{F}_B(w_i) = emb(pos(w_i)) \oplus emb(role(w_i)) \oplus EOR_{1,\ldots,k}(w_i)$; $pos(w_i)$ is the part-of-speech of $w_i$; $role(w_i)$ is the syntactic role of $w_i$; $emb(\cdot)$ is the respective embedding for the categorical input that can be trained as part of the model, or pre-trained; and $EOR_{1,\ldots,k}(w_i)$ represent the $k$ $EOR$ tags for $w_i$ assigned by the $k$ Open IE tools.

### 2.3 Model Architecture

Our system uses bidirectional long short term memory (Bi-LSTM) (Schuster and Paliwal, 1997) to aggregate features and classify the labels of a sequence of words. The advantage of using Bi-LSTM is that we can leverage the information from neighboring words from both sides. The outputs are used in softmax for each word, producing independent probability distributions over possible $EOR$ tags

### 2.4 Implementation Details

We implement *SenseOIE* using the Keras framework (Chollet et al., 2015) with TensorFlow backend [2]. We use 2 layers of stacked bidirectional LSTM, each with 100 neurons with tanh activation. We use the RMSprop optimizer which is often recommended for recurrent neural network. The model is trained using early stopping to prevent over fitting. We use the batch size of 32 samples, with 10% word-level dropout. The word embeddings are initialized using the word2vec (Mikolov et al., 2013) Google News 300-dimensions pre-trained embeddings. The part of speech and syntactic role embeddings are 25 dimensional and randomly initialized and updated during training.

## 3 Experiments

We validate *SenseOIE* with several benchmark data sets and compare it with the state-of-the-art Open IE systems including (1) a supervised Open IE system by (Stanovsky et al., 2018); (2) three unsupervised Open IE systems, OpenIE5 [3], Stanford OpenIE [4] (Angeli et al., 2015) and UKG which is a proprietary Open IE tool.

### 3.1 Baseline Systems

***RnnOIE*** is the first supervised model built for Open IE (Stanovsky et al., 2018). The model is based on a Bi-LSTM transducer and is trained using the annotated corpus built by the same research

[2] https://www.tensorflow.org/
[3] http://openie.allenai.org
[4] https://stanfordnlp.github.io/CoreNLP/openie.html

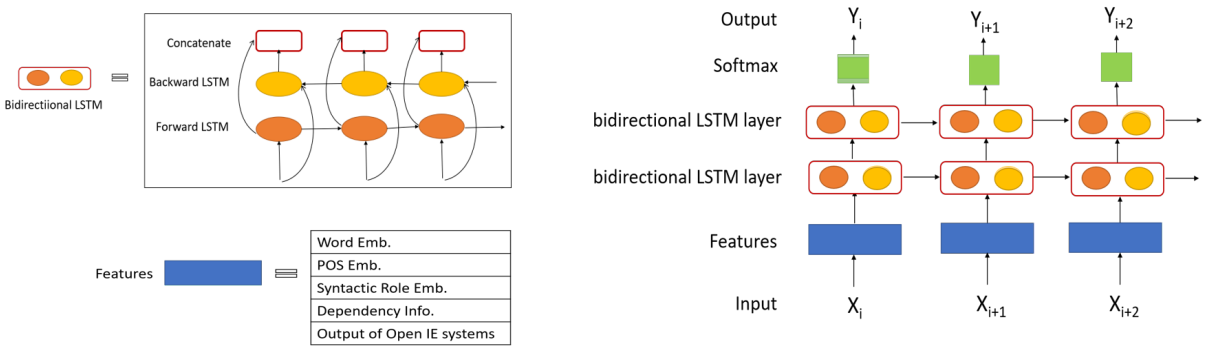


Figure 2: Model Architecture of *SenseOIE*

team (Stanovsky and Dagan, 2016). It takes a sentence and the word index of the predicate's syntactic head as input, and generates a feature vector for each word in the sentence by concatenating the word embeddings and POS tag embeddings of the word and the predicate head. Given these input features, the model learns whether the current word is part of argument of the particular predicate. At inference time, they first identify verbs and verb nominalization as candidate predicates and generate an input instance with each candidate predicate head.

**Stanford Open IE** is heavily based on dependency parsers. A classifier is learned for splitting a sentence into a set of logically entailed shorter clauses by recursively traversing its dependency tree and predicts whether an edge should yield an independent clause or not. In order to increase the usefulness of the extracted propositions, each self-contained clause is then maximally shortened by running natural logic inference over it. In the end, a set of 14 handcrafted patterns are used to extract a predicate-argument triple from each utterance.

**OpenIE 5** is a combination of four Open IE systems CALMIE (Saha et al., 2018), BONIE (Saha et al., 2017b), RelNoun (Pal et al., 2016) and SRLIE (Christensen et al., 2011). SRLIE converts the output of a SRL system into an Open IE extraction by treating the verb as the relational phrase, and taking its role-labeled arguments as the arguments of the relation. On the other hand, RelNoun is a nominal Open IE system that extracts relations from compound noun phrases. BONIE focuses on extracting tuples where one of the arguments is a number or a quantity-unit phrase. CALMIE extracts information from conjunctive sentences by using language model based scoring and several linguistic constraints to search over hierarchical conjunct boundaries.

**UKG** was developed by some of this paper's authors as a tool to construct a knowledge graph for the cybersecurity domain, which contains information about cyber-incidents involving malware, campaign, and IoCs (Indicators of Compromise). It extracts verbal binary relations based on noun phrase detection, named entity recognition, dependency parsing. UKG currently extracts verbal binary relations from three dependency structures, 'NP-VP-NP', 'NP-VP-PP' and 'VP-NP-PP'. Named entity extraction is performed to detect cybersecurity-specific entities (e.g., malware names) and constrain the extractions to only cybersecurity-related relations (those with at least one argument being a cybersecurity entity). Further, UKG employs a coreference resolution, and coordination and apposition analysis to increase the recall of the extraction. To make UKG similar to other OpenIE systems for the evaluation, we did not run the cybersecurity named entity extraction but used all noun phrases as candidate arguments. Also, we did not apply the coreference resolution as other systems produce pronouns, not the referring nouns, as arguments.

**Majority Votes** is another baseline system that we compared with *SenseOIE*. In this system we simply take majority votes from three different IE systems that we used to generate input feature for *SenseOIE*.

### 3.2 Experiment Data

We use four different benchmark datasets to train and test the models. The datasets are AW-OIE (Stanovsky et al., 2018), WEB and NYT (de Sá Mesquita et al., 2013) and PENN (Xu et al., 2013). Table 2 presents more details on these datasets.

| Data Set | # of Sentences | # of Tuples |
|---|---|---|
| AW-OIE | 3,300 | 17,165 |
| AW-OIE-C | 3,300 | 13,056 |
| WEB | 500 | 461 |
| NYT | 222 | 222 |
| PENN | 100 | 51 |

Table 2: Data sets used in this work

AW-OIE corpus was created by extending the OIE2016 corpus released by (Stanovsky and Dagan, 2016). OIE2016 (Stanovsky and Dagan, 2016) was created by an automatic translation from question-answering driven semantic role labeling annotations (He et al., 2015). (Stanovsky et al., 2018) extended these techniques and apply them to the QAMR corpus (Michael et al., 2018) to create AW-OIE. This dataset is the largest dataset available for supervised open information extraction.

However, when we observe the information extracted from this dataset, we notice that the dataset is not accurate enough to be considered as a benchmark dataset. We often find missing relations and noise introduced during the automatic generation process. To solve this problem, we manually inspect the dataset and find several patterns causing this noise in the dataset. We use these patterns to filter out noisy and missing relations from the dataset and call the cleaned data set 'AW-OIE-C'.

The WEB dataset represents the challenges of dealing with web text. This contains many incomplete and grammatically unsound sentences. NYT contains formal, well written news stories from the New York Times Corpus. The PENN dataset was created from PENN Tree Bank. We use AW-OIE-C for training and testing purpose and other three datasets only for testing. We use 8,000 instances from AW-OIE-C to train *SenseOIE* and 1,456 instances to test all the models. We set aside 3,600 instances for a new experiment described in Section 3.4.

### 3.3 Performance Evaluation

In this section, we report the utility of our model by comparing its performance with the baseline systems on the four datasets (AW-OIE-C, WEB, NYT and PENN).

**Evaluation Metric and Matching Function** We compare the systems using precision, recall and F1-score. In order to compute the measures, we need to match the automated extractions by the systems and the ground truth extractions. In this work, we compute the measures based on tuple-level matching and word-level matching. Word-level matching has been used for the evaluation metric for many NER systems. For each word, we match the tag generated by the system with the word's label.

Tuple-level matching is used in other Open IE systems (Stanovsky et al., 2018; Cui et al., 2018). It is done by mapping extracted tuples with their corresponding benchmark tuples. One strategy for tuple matching would be to enforce an exact match by matching the boundaries of the extracted and benchmark tuples in text. However, as noted in earlier works (Stanovsky et al., 2018; Schneider et al., 2017), this method penalizes different but equally valid arguments, which are resulted from different annotation styles employed by different Open IE systems. Therefore, dealing with multiple OIE systems requires a less restrictive matching strategy. (Schneider et al., 2017) introduced *relaxed containment strategy*. With this strategy, extractions are counted correct as long as they contain all gold standard arguments. (Stanovsky et al., 2018) used a partial matching strategy allowing some variability (e.g., omissions of prepositions or auxiliaries) in the predicted tuples.

Following these works, we also use a partial matching strategy that allows all these kind of variabilities. We consider each argument or predicate correct, if it partially matches with the benchmark data over a certain threshold. This threshold can control the leniency or strictness of the matching function. This metric allows a more balanced and fair comparison between systems which can extract potentially correct arguments beyond benchmark extraction.

**Comparison with the Baseline Systems** Table 3 shows the tuple-level F1-score of *SenseOIE* and the benchmark systems. As we can see, *SenseOIE* outperforms all baseline systems with a large difference. On the AE-OIE-C dataset, *SenseOIE* achieves the highest F1-score of 0.79. In comparison with the unsupervised Open IE methods, the performance gain of *SenseOIE* ranges from 66% to 315%. *SenseOIE* outperforms OpenIE5 by 36% to 56%. When compared to UKG, *SenseOIE*'s performance gain ranges from 92% to 186%. In terms of *SenseOIE*'s performance over the different datasets, it's worth noting the differences in annotations in the different

| | AW-OIE-C | Web | NYT | PENN | AW-OIE |
|---|---|---|---|---|---|
| *SenseOIE* | **0.79** | 0.66 | **0.41** | **0.52** | **0.72** |
| *RnnOIE* | - | **0.67** | 0.35 | 0.44 | 0.62 |
| OpenIE5 | 0.58 | 0.46 | 0.29 | 0.34 | - |
| Stanford OpenIE | 0.19 | 0.24 | 0.21 | 0.31 | - |
| UKG | 0.41 | 0.23 | 0.15 | 0.21 | - |
| Majority Votes | 0.40 | 0.42 | 0.24 | 0.27 | - |

Table 3: Performance (F1-score) comparison of *SenseOIE* and the baseline systems

datasets. As the test data from AW-OIE-C follows the same annotation style as the training data, the performance of *SenseOIE* is much higher on this dataset compared to other datasets.

Figure 3 shows the comparison results based on the word level F1-scores. The results also demonstrate that *SenseOIE* works better than the other systems. Especially, *SenseOIE* shows much higher accuracy in detecting words belonging to the arguments and the relation, but a slightly lower accuracy for other words.

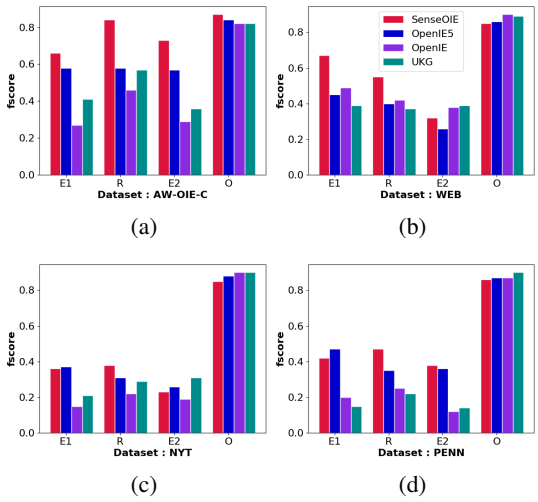


Figure 3: Word level F1-score comparison

**Feature Ablation Study** To investigate the contribution of each feature type on *SenseOIE*, we conducted a feature ablation study. Table 4 shows the F1-scores of several variations built with a different subset of our features. We note that the performance of *SenseOIE* is, on aevrage, 92% higher than the model using only the 'Embedding Features', which is similar to the features used in other supervised Open IE systems (Stanovsky et al., 2018; Cui et al., 2018). Overall the performance gain ranges from 13% to 156%. This performance boost proves that using an ensemble of multiple unsupervised OpenIE systems is very effective. The results also show that the ensemble of unsupervised Open IE results is more effective than the embedding features, and the combined features of the embeddings and Open IE results produce the best results. Surprisingly model without features from dependency parse tree outperforms *SenseOIE* in 2 out of 4 datasets. This might be indicator that simple concatenation is not the best way to include features from dependency tree.

### 3.4 *SenseOIE* as Annotator

Since *SenseOIE* outperforms the baseline systems by a large margin, we investigate if *SenseOIE* can be used to bootstrap a supervised Open IE model for new domains by automatically producing annotated data. Previously, (Cui et al., 2018) used OpenIE4 (Mausam, 2016b), an earlier version of OpenIE5, to automatically create a training dataset. The limitation of their approach is that using only one OpenIE system's extraction as ground truth will result in biased and low coverage of extracted relations. As each of the unsupervised OpenIE systems has it's own rules to extract different relations, applying only one system might miss other potential relations that can be extracted by other Open IE systems. However, since *SenseOIE* learns from multiple existing Open IE systems, it can extract many different relation types.

For this purpose, we run *SenseOIE* on the 3,600 instances from AW-OIE-C and use its extraction results as the ground truth to train a supervised model. We name this new model *SupervisedOIE* to differentiate it from *SenseOIE*. The model is quite similar to *SenseOIE* using LSTM to aggregate features and classify the labels of a sequence of words. The input features for each word are word embedding, pos embedding, syntactic role embedding, dependency tree information and label of previous word. During training label of pre-

| | AW-OIE-C | Web | NYT | PENN |
|---|---|---|---|---|
| Embedding Features | 0.42 | 0.58 | 0.16 | 0.27 |
| Open IE Result Features | 0.70 | 0.54 | 0.38 | 0.47 |
| Embedding + Open IE Result Features | 0.78 | **0.69** | **0.45** | 0.51 |
| All Features | **0.79** | 0.66 | 0.41 | **0.52** |

Table 4: Performance (F1-score) comparison of different feature sets. 'Embedding Features' denotes the concatenated set of word embedding, POS embedding and syntactic role embedding. 'Open IE Result Features' include only the $EOR$ tags generated by the three unsupervised Open IE systems. 'All Features' consists of all the features as described in Section 2.2.

vious word comes from ground truth and during testing this value is predicted by the model. This feature is useful to generate multiple sequences of extractions from a single sentence. However, note that this model does not use the results of unsupervised systems as features. Figure 4 shows the architecture and features of *SupervisedOIE*.

During the inference time, to extract multiple relations from a single sentence, we use beam search to find multiple possible labels for each word. Instead of greedily choosing the most likely next step as the sequence is constructed, the beam search expands all possible next steps and keeps the $k$ most likely results, where $k$ is a user-specified parameter and controls the number of beams or parallel searches through the sequence of probabilities. Figure 5 shows an example of beam search predicting multiple relation extraction sequences from one sentence.

To validate the effectiveness of *SenseOIE* as an annotator model, we compare *SupervisedOIE*'s performance when trained with the human labeled data and *SenseOIE*'s extractions. As with *SenseOIE*, both models are initialized with the pre-trained word embedding and randomly initialize the part-of-speech and syntactic role embeddings. In this experiments, we set the beam size to 3 which gives an overall best performance. Table 5 shows the results from these two models. We can see that both models achieve similar F1-scores on the four test dataset. These results supports *SenseOIE*'s role as a digital annotator for unlabeled dataset.

| | AW-OIE-C | Web | NYT | PENN |
|---|---|---|---|---|
| Human Labels | 0.55 | 0.51 | 0.23 | 0.27 |
| *SenseOIE* Labels | 0.54 | 0.50 | 0.23 | 0.23 |

Table 5: Performance (F1-score) of *SupervisedOIE* trained with the human-labeled data vs. labeled data generated by *SenseOIE*

## 4 Related Work

Early Open IE systems apply handcrafted rules or self-supervised learning paradigm, where the extraction results which satisfy a set of syntactic constraints are considered as positive examples and the results which do not satisfy the constrains are considered as negative examples. *TextRunner* (Banko et al., 2007) is the first domain-independent Open IE system. *ReVerb* (Fader et al., 2011) extracts verbal propositions from part of speech tags using a logistic regression classifier. OLLIE (Mausam et al., 2012) is built on *ReVerb* and extracts relations from syntactic and lexical dependency patterns. *ClausIE* (Corro and Gemulla, 2013) first classifies clauses into clause types and extract tuples based on the clause type using predefined rules. Followed by these systems, Stanford OpenIE (Angeli et al., 2015) and OpenIE5 are developed by combining several different approaches as described in details in Section 3.1. Further, several systems focusing on a specialized constructs were developed, including noun-mediated relations (Pal and Mausam, 2016), n-ary relations (Akbik and Löser, 2012), nested propositions (Bhutani et al., 2016) and numerical Open IE (Saha et al., 2017a).

Recently, there have been efforts to apply deep learning methods to Open IE. *RnnOIE* (Stanovsky et al., 2018) is the first attempt to apply a supervised learning approach for Open IE using the labeled data set from (Stanovsky and Dagan, 2016) (See Section 3.1). (Cui et al., 2018) propose an encoder-decoder framework with an attention-based copying mechanism to extract binary relation tuples. They formulated Open IE as a sequence-to-sequence generation problem. Instead of relying on manually labeled data,

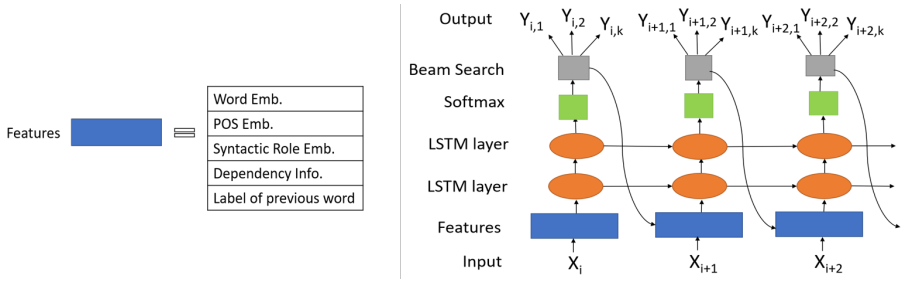


Figure 4: Model Architecture of a new *SupervisedOIE*

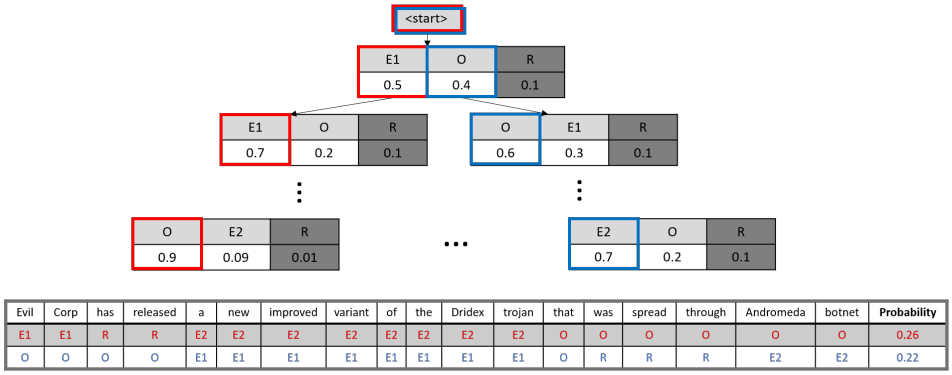


Figure 5: Example of beam search predicting multiple relation extraction sequences from one sentence

they train the model using the results of OpenIE4 (Mausam, 2016a) as labeled training data and evaluate the model using the human-labeled data from (Stanovsky and Dagan, 2016) as *RnnOIE* (Stanovsky et al., 2018). (Sun et al., 2018) present a supervised neural Open IE model for Chinese information extraction. They apply an attention-based sequence-to-sequence learning similarly to (Cui et al., 2018). However, they use the gated dependency attention mechanism based on the shortest path between a pair of words in the sentence's dependency tree. We do not compare our model with this system because it supports different target types and languages.

## 5 Conclusion

We propose a new Open IE paradigm which combines supervised learning and unsupervised Open IE systems. Our model uses the results of existing Open IE systems as features in addition to other linguistic features and then optimize the model using a small amount of labeled data. Validation using several benchmark data sets generated for the Open IE task shows that our method is very effective outperforming both other supervised and unsupervised Open IE systems.

Further, we investigate if our model can be applied to automatically generate annotated data to train a new supervised model for a new task. The experiment shows that a supervised model trained with the model-generated data performes similarly as the model trained with human labeled data. This result shows that our approach can overcome the cold-start problem in machine learning by leveraging existing unsupervised systems.

## 6 Acknowledgment

We gratefully acknowledge IBM for supporting this work.

## References

Alan Akbik and Alexander Löser. 2012. Kraken: N-ary facts in open information extraction. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction, AKBC-WEKEX@NAACL-HLT 2012*, pages 52–56.

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL)*, pages 344–354.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2670–2676.

Nikita Bhutani, H. V. Jagadish, and Dragomir R. Radev. 2016. Nested propositions in open information extraction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 55–64.

François Chollet et al. 2015. Keras. https://keras.io.

Janara Christensen, Stephen Soderland, Oren Etzioni, et al. 2011. An analysis of open information extraction based on semantic role labeling. In *Proceedings of the sixth international conference on Knowledge capture*, pages 113–120. ACM.

Luciano Del Corro and Rainer Gemulla. 2013. ClausIE: clause-based open information extraction. In *22nd International World Wide Web Conference (WWW)*, pages 355–366.

Lei Cui, Furu Wei, and Ming Zhou. 2018. Neural open information extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 407–413.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1535–1545.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1156–1165.

Luheng He, Mike Lewis, and Luke Zettlemoyer. 2015. Question-answer driven semantic role labeling: Using natural language to annotate natural language. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 643–653.

Amina Kadry and Laura Dietz. 2017. Open relation extraction for support passage retrieval: Merit and open issues. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*, pages 1149–1152.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2017. Answering complex questions using open information extraction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL): Short Paper*, pages 311–316.

Mausam. 2016a. Open information extraction systems and downstream applications. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4074–4077.

Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 523–534.

Mausam Mausam. 2016b. Open information extraction systems and downstream applications. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 4074–4077. AAAI Press.

Julian Michael, Gabriel Stanovsky, Luheng He, Ido Dagan, and Luke Zettlemoyer. 2018. Crowdsourcing question-answer meaning representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), Short Paper*, pages 560–568.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Harinder Pal and Mausam. 2016. Demonyms and compound relational nouns in nominal open IE. In *Proceedings of the 5th Workshop on Automated Knowledge Base Construction, AKBC@NAACL-HLT 2016*, pages 35–39.

Harinder Pal et al. 2016. Demonyms and compound relational nouns in nominal open ie. In *Proceedings of the 5th Workshop on Automated Knowledge Base Construction*, pages 35–39.

Benjamin Roth, Costanza Conforti, Nina Pörner, Sanjeev Karn, and Hinrich Schütze. 2018. Neural architectures for open-type relation argument extraction. *CoRR*, abs/1803.01707.

Filipe de Sá Mesquita, Jordan Schmidek, and Denilson Barbosa. 2013. Effectiveness and efficiency of open relation extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 447–457.

Swarnadeep Saha, Harinder Pal, and Mausam. 2017a. Bootstrapping for numerical open IE. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL), Short paper*, pages 317–323.

Swarnadeep Saha, Harinder Pal, et al. 2017b. Bootstrapping for numerical open ie. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 317–323.

Swarnadeep Saha et al. 2018. Open information extraction from conjunctive sentences. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2288–2299.

Rudolf Schneider, Tom Oberhauser, Tobias Klatt, Felix A. Gers, and Alexander Löser. 2017. Analysing errors of open information extraction systems. *CoRR*.

Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Trans. Signal Processing*, 45(11):2673–2681.

Stephen Soderland, Brendan Roof, Bo Qin, Shi Xu, Mausam, and Oren Etzioni. 2010. Adapting open information extraction to domain-specific relations. *AI Magazine*, 31(3):93–102.

Gabriel Stanovsky and Ido Dagan. 2016. Creating a large benchmark for open information extraction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2300–2305.

Gabriel Stanovsky, Julian Michael, Luke Zettlemoyer, and Ido Dagan. 2018. Supervised open information extraction. In *Proceedings of The 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT)*, pages 885–895.

Mingming Sun, Xu Li, Xin Wang, Miao Fan, Yue Feng, and Ping Li. 2018. Logician: A unified end-to-end neural approach for open-domain information extraction. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining (WSDM)*, pages 556–564.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1556–1566.

Ying Xu, Mi-Young Kim, Kevin Quinn, Randy Goebel, and Denilson Barbosa. 2013. Open information extraction with tree kernels. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 868–877.

Dian Yu, Lifu Huang, and Heng Ji. 2017. Open relation extraction and grounding. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing, IJCNLP 2017, Taipei, Taiwan, November 27 - December 1, 2017 - Volume 1: Long Papers*, pages 854–864.

# Neural Cross-Lingual Event Detection with Minimal Parallel Resources

**Jian Liu**[12]*, **Yubo Chen**[1]*, **Kang Liu**[12], **Jun Zhao**[12]
[1] National Laboratory of Pattern Recognition, Institute of Automation
Chinese Academy of Sciences, Beijing, 100190, China
[2] University of Chinese Academy of Sciences
{jian.liu, yubo.chen, kliu, jzhao}@nlpr.ia.ac.cn

## Abstract

The scarcity in annotated data poses a great challenge for event detection (ED). Cross-lingual ED aims to tackle this challenge by transferring knowledge between different languages to boost performance. However, previous cross-lingual methods for ED demonstrated a heavy dependency on parallel resources, which might limit their applicability. In this paper, we propose a new method for cross-lingual ED, demonstrating a minimal dependency on parallel resources. Specifically, to construct a lexical mapping between different languages, we devise a context-dependent translation method; to treat the word order difference problem, we propose a shared syntactic order event detector for multilingual co-training. The efficiency of our method is studied through extensive experiments on two standard datasets. Empirical results indicate that our method is effective in 1) performing cross-lingual transfer concerning different directions and 2) tackling the extremely annotation-poor scenario.

## 1 Introduction

Event detection (ED) is a crucial natural language processing problem that aims to identify event triggers in texts (Ahn, 2006; Nguyen and Grishman, 2015). For example, in the sentence: "*A man* **died** *when a tank* **fired** *on the hotel*", ED requires a system to identify two event triggers, **died** and **fired**, along with their types `Die` and `Attack`.

Generally, training an ED system requires to obtain a considerably large amount of labeled data. However, owing to the complexity and high costs of annotation, existing event resources are scarce and unbalanced across languages (Hsi et al., 2016), which prevents us from building an ED system in languages with insufficient training data. Cross-lingual ED (Ji, 2009; Chen and Ji, 2009; Zhu et al., 2014; Hsi et al., 2016; Liu et al., 2018a) aims to tackle this challenge by transferring knowledge cross languages to boost performance. However, previous cross-lingual ED methods rely on either high-performance machine translation (MT) systems trained on large numbers of parallel sentences or manually aligned documents to achieve a decent performance — the required parallel resources may only exist for a small fraction of language pairs (Koehn et al., 2007), which greatly limits the applicability of these methods.

In this paper, we propose a new simple but effective method for cross-lingual ED, which can overcome the data scarcity problem in annotation-poor languages by jointly training with resources in other languages. Compared with previous methods, our approach demonstrates a minimal dependency on parallel resources, which may fit with language pairs that do not have large bitexts.

To achieve cross-lingual transfer, two challenges exist: 1) how to build lexical mappings between different languages, and 2) how to handle the word order difference problem (Xie et al., 2018). For the first challenge, previous studies (Guo et al., 2015; Ni et al., 2017; Mayhew et al., 2017; Xie et al., 2018; Lample et al., 2018) have investigated embedding projection-based method in cross-lingual applications and achieved promising results. For example, (Xie et al., 2018) proposed a novel "cheap translation" based method which has greatly advanced the performance in zero-shot named entity recognition (NER). However, these methods may not directly fit with cross-lingual ED, as the lexical mapping in ED is usually context-dependent but not deterministic as in other tasks. Consider the following English-to-Chinese lexical mapping examples. To preserve its original meaning, the trigger word "fire" in "*gangsters* fire *at a policeman*" (which evokes an `Attack`

*Equal contribution.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 738–748, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

event), and "*the house caught* fire" (which evokes an NA event) should be translated as different Chinese words "开 火(open fire)" and "着 火(be on fire)" respectively. But in previous lexical mapping methods, the "fired" is always having the same transferred representation irrespective of its contexts. This problem might introduce noise for cross-lingual ED.

To address the above issues, in this paper, we devise a content-dependent lexical mapping method for cross-lingual ED. Similar to (Xie et al., 2018), for each source word, we first project it into a shared embedding space, but instead of adopting a deterministic word-to-word translation, we retrieve different translation candidates by looking for its nearest neighbors, and we then adopt a context-aware selective attention mechanism to rank these candidates to find the best-suited translation. Compared with previous methods, our approach can obtain content-dependent translations for each word in the source training data, which may be more suitable for cross-lingual ED.

Considering the word order difference, to our best knowledge, (Xie et al., 2018) is the only work which adopted a self-attention mechanism (Vaswani et al., 2017) to tackle this problem (in cross-lingual NER). Difference with them, we propose a shared syntactic order event detector for cross-lingual ED, which explores the syntactic similarity of resources in different languages and circumvents the word order difference problem in performing multilingual co-training.

To illustrate our motivation, consider an English example E1 and its parallel Chinese translation C1 in Figure 1. As shown, E1 and C1 have rather different word orders (Figure 1a), but they share a similar syntactic structure which captures enough generality for identifying event triggers (Figure 1b). This observation motivates us to explore the syntactic similarity to achieve multilingual co-training. To achieve this goal, we propose a shared syntactic order event detector based on Graph Convolutional Networks (GCNs) (Kipf and Welling, 2016), which can provide each word a contextual feature based on its immediate neighbors in the syntactic graph irrespective of its original position in the sentence. This decoder allows us to train a model on multilingual resources effectively, which circumvents the word order different problem.

To estimate our method, we have conducted

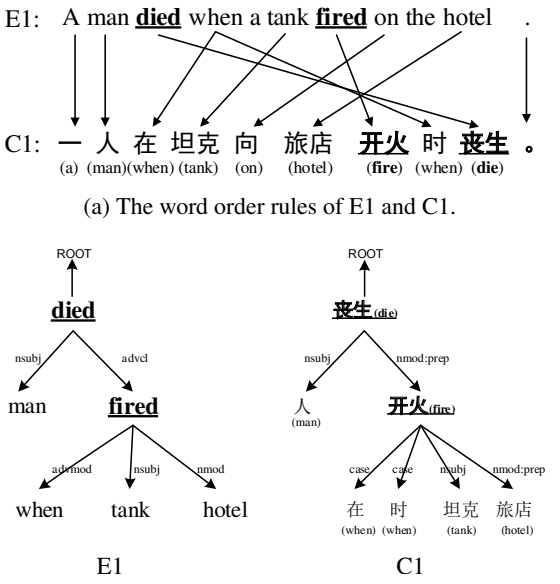


(a) The word order rules of E1 and C1.

(b) The syntactic structures of E1 and C1.

Figure 1: The comparison of word orders and syntactic structures of an annotated English sentence E1 and its parallel Chinese sentence C1.

extensive experiments on two standard datasets, using English, Chinese, and Spanish as experimental languages respectively. The experimental results demonstrate that: 1) our model can perform cross-lingual transfer between different language pairs. Especially, the improvement in Chinese ED is large, with an absolute 3.8% in F1 over the monolingual approaches. 2) Our model is robust in the extreme annotation-poor scenario where a language has very limited training data, which demonstrates a definite advantage over previous monolingual models. Additionally, compared with MT-based methods, our model achieves competitive results but requires much less parallel resource.

This paper is organized as follows: Section 2 briefly introduces the task description and terminologies used in ED; Section 3 elaborates details of our approach; Section 4 reports on our experimental results and other analysis; Section 5 reviews related work; Section 6 concludes the paper and illustrates future work.

## 2 Task Description

Event detection (ED) is a subtask defined in the overall Event Extraction (EE) evaluation in Automatic Content Extraction (ACE) 2005 program. We first introduce some ACE terminologies to facilitate the understanding of ED task.

In ACE, 1) an **event mention** refers to a phrase/sentence within which an event is described. 2) **Event trigger** refers to a specific word in an event mention which is considered the most representative of the event. Each event trigger has a certain type corresponding to the event mention. 3) **Event arguments** are participants of the event.

With these definitions, the goal of ED is to locate event triggers and categorize their types. For example, in sentence "*The old man* **died** in the *hospital*", ED requires a system to detect a `Die` event along with the event trigger **died**. The detection of event arguments *The old man* (role=`Person`) and *hospital* (role=`Place`) is not involved in the ED task.

Following previous work (Nguyen and Grishman, 2015; Liu et al., 2016), we formulate ED as a token-level multi-class classification task. Namely, given a sentence, we treat every token in it as a trigger candidate, and we aim to classify each candidate into one of 34 categories (33 event types defined in ACE in addition to an `NA` type indicating "not an event trigger").

## 3 Methodology

This study focuses on cross-lingual ED, which aims to transfer knowledge from a source language with abundant labeled data to a target language with insufficient training data.

Figure 2 visualizes the overall architecture of our model, which consists of three main components: (1) Monolingual embedding layer, which transforms each token into a continuous vector representation. (2) Context-dependent lexical mapping, which maps each word in the source language to its best-suited translation in the target language, by examining its contextual representation and imposing a selective attention over different translation candidates. (3) Shared syntactic order event detector, which employs a Graph Convolutional Networks (GCNs) to explore syntactic similarity of resources of different languages, in order to achieve multilingual co-training.

For the sake of convenience, in the following illustrations, we assume the source language is English and the target language is Chinese, and we use an English sentence $s = \{w_1, w_2, \ldots, w_n\}$ to illustrate our idea.

### 3.1 Monolingual Embedding Layer

In the monolingual embedding layer, each word is assigned to a distributed vector as its representation. Specifically, we first train English/Chinese word embeddings on the corresponding Wikipedia dumps via Skip-gram model (Mikolov et al., 2013) with a dimensional size $d = 300$. And then we transform each token into its word embedding as its vectorized feature representation.

In this way, $s$ is transformed into an embedding matrix $\boldsymbol{E}_s = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]^T$, where $\boldsymbol{x}_i \in \mathbb{R}^d$ indicates the word embedding of the token $w_i$.

### 3.2 Context-Dependent Lexical Mapping

For each token $w_i$ in $s$, context-dependent lexical mapping aims to search for its best-suited word translation according to its contextual representation. This process involves: 1) learning multilingual alignment, 2) retrieving translation candidates, and 3) ranking translation words via a selective attention mechanism.

#### 3.2.1 Learning Multilingual Alignment

Let $\boldsymbol{X}$ and $\boldsymbol{Y}$ be the English and Chinese embedding spaces. In order to achieve multilingual alignment, we learn a mapping $\boldsymbol{W} \in \mathbb{R}^{d \times d}$ from $\boldsymbol{X}$ to $\boldsymbol{Y}$ via a seed dictionary with a size of $m$, by optimizing:

$$\boldsymbol{W}^* = \underset{\boldsymbol{W} \in M_d(\mathbb{R})}{\arg\min} ||\boldsymbol{W}\boldsymbol{X}_{dic} - \boldsymbol{Y}_{dic}||_F \quad (1)$$

where $M_d(\mathbb{R})$ is the space of $d \times d$ matrices; $\boldsymbol{X}_{dic}$, $\boldsymbol{Y}_{dic} \in \mathbb{R}^{d \times m}$ are two matrices containing the aligned embeddings of words in the seed dictionary; $||\cdot||_F$ indicates the Frobenius norm. To get a closed form solution, following (Xing et al., 2015; Lample et al., 2018), we impose an orthogonality constraint on $\boldsymbol{W}$ (i.e.,$\boldsymbol{W}\boldsymbol{W}^T$=$\boldsymbol{W}^T\boldsymbol{W}$=$\boldsymbol{I}$), and in this way, the optimized solution of $\boldsymbol{W}$ corresponds to the singular value decomposition (SVD) of $\boldsymbol{Y}_{dic}\boldsymbol{X}_{dic}^T$, i.e.,

$$\boldsymbol{W}^* = \underset{\boldsymbol{W} \in O_d(\mathbb{R})}{\arg\min} ||\boldsymbol{W}\boldsymbol{X}_{dic} - \boldsymbol{Y}_{dic}||_F = \boldsymbol{U}\boldsymbol{V}^T \quad (2)$$

where $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T = \mathrm{SVD}(\boldsymbol{Y}_{dic}\boldsymbol{X}_{dic}^T)$.

#### 3.2.2 Retrieving Translation Candidates

Next, we retrieve translation candidates for each token $w_i$ in $s$. Specifically, we first project $w_i$ into the aligned embedding space (i.e., by applying $\boldsymbol{W}$ on $\boldsymbol{x}_i$), and then we explore its neighborhood to find the nearest Chinese words as its translation candidates. In order to measure the distance

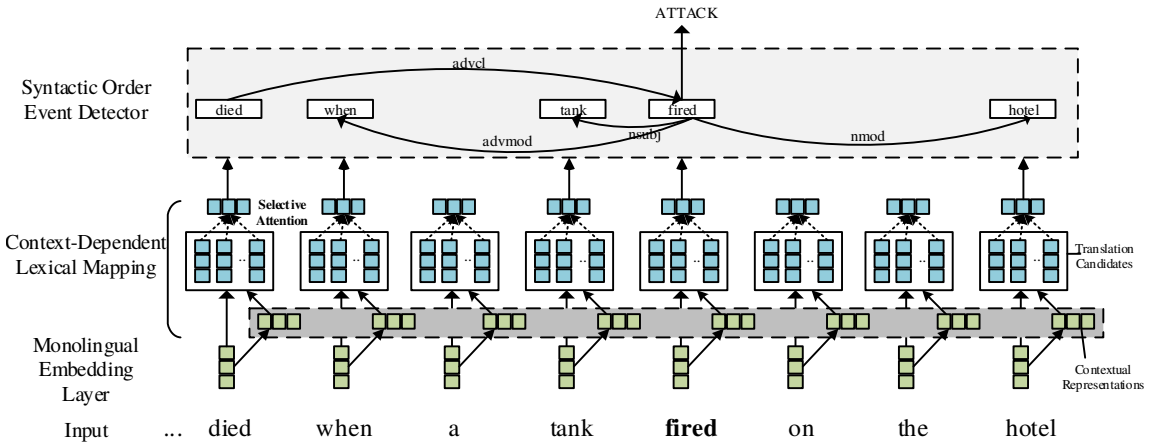


Figure 2: The overview architecture of our model. The figure illustrates the process of performing cross-lingual transfer for an English sentence "A man died when a tank fired on the hotel" into Chinese and using the shared syntactic order event detector to predict the event type for the word "fired".

of $w_i$ and a Chinese word $y$ in the aligned space, we adopt the cross-domain similarity local scaling (CSLS) metric (Lample et al., 2018):

$$\begin{aligned}\text{CSLS}(\boldsymbol{W}\boldsymbol{x}_i, \boldsymbol{y}) =& 2\cos(\boldsymbol{W}\boldsymbol{x}_i, \boldsymbol{y}) \\ & - r_{\boldsymbol{Y}}(\boldsymbol{W}\boldsymbol{x}_i) - r_{\boldsymbol{X}}(\boldsymbol{y})\end{aligned} \tag{3}$$

where $\boldsymbol{y}$ denotes the (Chinese) word embedding of $y$; $r_{\boldsymbol{Y}}(\boldsymbol{W}\boldsymbol{x}_i)$ indicates the mean cosine similarity between $\boldsymbol{W}\boldsymbol{x}_i$ and its $K$ neighbors in $\boldsymbol{Y}$, which is defined as: $r_{\boldsymbol{Y}}(\boldsymbol{W}\boldsymbol{x}_i) = \frac{1}{K}\sum_{\boldsymbol{y}' \in \mathcal{N}_{\boldsymbol{Y}}(\boldsymbol{W}\boldsymbol{x}_i)} \cos(\boldsymbol{W}\boldsymbol{x}_i, \boldsymbol{y}')$ where $\mathcal{N}_{\boldsymbol{Y}}(\boldsymbol{W}\boldsymbol{x}_i)$ denotes the neighborhood associate with $\boldsymbol{W}\boldsymbol{x}_i$ in $\boldsymbol{Y}$. In our method, for $w_i$, we take $J$ Chinese words which have the smallest CSLSs as its translation candidates. We denote by $T^{(w_i)}$ the set of translation candidates for $w_i$, where $T_j^{(w_i)}$ indicates the $j$th element of $T^{(w_i)}$.

### 3.2.3 Content-Aware Selective Attention

Finally, for each token $w_i$, we perform a context-aware selective attention mechanism to weigh each translation candidate in $T^{(w_i)}$ and get the best-suited translation for it.

**Learning Contextual Representation.** We employ the self-attention mechanism (Vaswani et al., 2017) to learn context representation of $w_i$. Specifically, given $\boldsymbol{E}_s = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]^T$, we use different single-layer neural networks to learn queries $\boldsymbol{Q}$, keys $\boldsymbol{K}$, values $\boldsymbol{V}$ respectively. For example, $\boldsymbol{Q} = \tanh(\boldsymbol{E}_s\boldsymbol{W}_m + \boldsymbol{b}_m)$, where $\boldsymbol{W}_m \in \mathbb{R}^{d\times d}$ and $\boldsymbol{b} \in \mathbb{R}^d$ are parameter matrix and bias respectively. Then, we compute a self-attention matrix by computing:

$$\boldsymbol{C}_s = \text{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d}})\boldsymbol{V} \tag{4}$$

$$= [\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_n]^T \tag{5}$$

where $d$ indicates the word embedding dimension. We take $\boldsymbol{c}_i$ as the contextual representation of $w_i$.

**Learning Selective Attention.** For each token $w_i$, after obtaining its translation candidates list $T^{(w_i)}$ and contextual representation $\boldsymbol{c}_i$, we impose a selective attention mechanism to automatically weigh each candidate. Specifically, the weight of the $j$th candidate $T_j^{(w_i)}$ is computed as:

$$\alpha_j = \frac{\exp(m_j)}{\sum_{i=1}^{J}\exp(m_i)} \tag{6}$$

where $m_j$ measures the semantic relatedness of $\boldsymbol{c}_i$ and $T_j^{(w_i)}$, which is computed by:

$$m_j = \tanh([\boldsymbol{c}_i; \boldsymbol{y}_j^{(w_i)}]\boldsymbol{W}_r + b_r) \tag{7}$$

where [;] indicates the concatenation operations; $\boldsymbol{y}_j^{(w_i)}$ denotes the Chinese word embedding of $T_j^{(w_i)}$; $\boldsymbol{W}_r \in \mathbb{R}^{d\times 1}$ and $b_r \in \mathbb{R}$ are parameter matrix and bias respectively. Finally, we select the candidate which has the maximal attention weight as the best-suited translation for $w_i$, which is denoted by $w_i'$. In this way, the original sentence $s$ is transfer into a Chinese word sequences $t = \{w_1', w_2', \ldots, w_n'\}$ with a same length.

## 3.3 Shared Syntactic Order Event Detector

As English and Chinese usually have different word orders, the transferred result $t$ might be seen

as a corrupted sentences from Chinese, which could introduce noise for multilingual co-training. We tackle this problem by proposing a Graph Convolutional Neural Networks (GCNs) (Kipf and Welling, 2016) based syntactic order event detector, which provides each word with a feature vector based on its immediate neighbors in the syntactic graph irrespective of its position in the sentence. This allows our model to train with the translated data $t$ and the other labeled data in Chinese indiscriminately.

#### 3.3.1 Extracting Graph Convolution Feature

Specifically, for each token $w_i$, our model computes a graph convolution feature vector based on its immediate neighbors in the syntactic graph. Figure 3 illustrates the process of extracting the feature for "fired".

Let $\mathcal{N}(w_i)$ denote the set of neighbors of $w_i$ in the syntactic graph, and $L(w_i, v)$ indicate the label of the dependency arc $(w_i \rightarrow v)$ (For example, $L$("fired", "hotel") = nmod in the example in Figure 3). The original GCNs compute a graph convolution vector for $w_i$ at ($k$+1)th layer by:

$$\boldsymbol{h}_{w_i}^{k+1} = g\Big(\sum_{v \in \mathcal{N}(w_i)} (\boldsymbol{W}_{L(w_i,v)}^{k} \boldsymbol{h}_{v}^{k}) + \boldsymbol{b}_{L(w_i,v)}^{k}\Big) \tag{8}$$

where $g$ denotes the ReLU function; $\boldsymbol{W}_{L(w_i,v)}^{k}$ and $\boldsymbol{b}_{L(w_i,v)}^{k}$ are parameters of the dependency label $L(w_i, v)$ in the $k$th layer. However, retaining parameters for every dependency label is space-consuming and compute-intense (there are approximately 50 labels), in our model, we limit $L(w_i, v)$ to have only three types of labels 1) an original edge, 2) a self loop edge, and 3) an added inverse edge, as suggested in (Nguyen and Grishman, 2018). Additionally, since the generated syntactic parsing structures usually contain noise, we apply attention gates on the edges to weigh their individual importances:

$$\alpha_{(w_i,v)}^{k} = \sigma(\boldsymbol{h}_{u}^{k} \boldsymbol{U}_{L(w_i,v)}^{k} + \boldsymbol{d}_{L(w_i,v)}^{k}) \tag{9}$$

where $\sigma$ is the logistic sigmoid function. $\boldsymbol{U}_{L(w_i,v)}^{k}$ and $\boldsymbol{d}_{(w_i,v)}^{k}$ are the weight matrix and the bias of the gate. With this gating mechanism, the final syntactic GCNs computation in our model is:

$$\boldsymbol{h}_{\boldsymbol{u}}^{k+1} = g\Big(\sum_{\boldsymbol{v} \in \mathcal{N}(\boldsymbol{u})} \alpha_{\boldsymbol{u},\boldsymbol{v}}^{k} (\boldsymbol{W}_{L(\boldsymbol{u},\boldsymbol{v})}^{k} \boldsymbol{h}_{\boldsymbol{v}}^{k}) + \boldsymbol{b}_{L(\boldsymbol{u},\boldsymbol{v})}^{k}\Big) \tag{10}$$

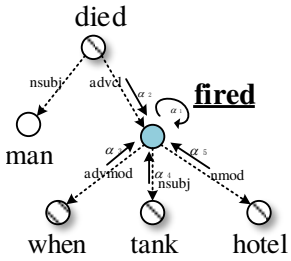


Figure 3: The illustration of using GCNs to compute the order-invariant feature for the word "fired".

We set the initial vectors $\boldsymbol{h}_{w_i}^{0}$ for $w_i$ as the Chinese word embedding of $w_i'$ (its translated word), and we stack 2 layers of GCNs (i.e., $k$ = 2) to obtain the final feature for $w_i$, denoted as $\mathbf{f_i}$.

### 3.4 Event Type Classification

Our model incorporates a logistic regression classifier to predict $w_i$'s event type. Specifically, we compute a prediction vector for $w_i$ by taking $\boldsymbol{f}_i$ as the input:

$$\boldsymbol{out} = \text{softmax}(\boldsymbol{W}_o \boldsymbol{f}_i + \boldsymbol{b}_o) \tag{11}$$

where $\boldsymbol{W}_o \in \mathbb{R}^{d \times c}$ and $\boldsymbol{b}_o \in \mathbb{R}^{c}$ are parameters, and $c$ is the total number of event types (i.e., 34 in this study). The probability of $t$-th class type is denoted as $P(t|w_i)$, which corresponds to the $t$-th element of $\boldsymbol{out}$.

### 3.5 Multilingual Co-Training

To enable multilingual co-training, we adopt the cross-entropy loss, and we use $\lambda$ to balance the contribution of multilingual resources (which is set as 0.7 through a grid search):

$$J(\Theta) = -\lambda \sum_{w_e'} \log P(l_{w_e'}|w_e') - \sum_{w_c} \log P(l_{w_c}|w_c) \tag{12}$$

where $\Theta$ denotes all the parameters in our model; $w_e'$ ranges over each token in the translated examples and $w_c$ enumerate each token in the original Chinese training set; $l_{w_e'}$ and $l_{w_c}$ denote the ground-truth event types of $w_e'$ and $w_c$ respectively. We adopt Adam rules (Kingma and Ba, 2014) to update our model's parameters and add dropout layers to prevent over-fitting.

## 4 Experiments

### 4.1 Datasets and Evaluation

Our main experiments are conducted on ACE 2005 and TAC KBP 2017, two widely used ED

datasets which contain annotated documents in English and Chinese (The documents are not parallel). For ACE English ED, we split the dataset to training/development/test set as suggested in (Li et al., 2013; Nguyen and Grishman, 2015). For ACE Chinese ED, we perform a 10-fold cross-validation as suggested in (Chen and Ji, 2009; Lin et al., 2018). For TAC KBP 2017 English and Chinese, we use the official test sets for testing, and we split the remaining data with a ratio of 9:1 for training and developing. The bilingual dictionary is obtained from the MUSE project[1] with a size of $5k$. The number of candidate translations $J$ (in Section 3.2.2) is set as 3. We use the Stanford CoreNLP (Manning et al., 2014) to obtain syntactic trees for each language.

Precision (Pre.), Recall (Rec.), and F1-score (F1) are used as evaluation metrics, same as previous ED studies to ensure compatibility. Significant tests (with p=0.05) were conducted using method described in (Yeh, 2000).

### 4.2 Experimental Results

We conduct two groups of experiments to investigate the ability of our model in 1) performing cross-lingual transfer concerning different language directions, and 2) handling the annotation-poor scenario.

#### 4.2.1 Cross-Lingual Transfer Concerning Different Language Directions

We investigate both English-to-Chinese and Chinese-to-English transfers to investigate whether cross-lingual transfer is feasible concerning different language directions. In these experiments, we jointly train on the translated data with all the labeled data in target language.

We compare our cross-lingual approach (*CL_Trans*) with our monolingual approach (*Monolingual*, which only uses the training data of the target language) and the existing state-of-the-art monolingual ED models (*Monolingual_SOTA*). In ACE ED, we take models proposed in (Nguyen and Grishman, 2015) and (Lin et al., 2018) as the SOTA English and Chinese ED systems respectively. In TAC KBP ED, we take the top systems reported in the official evaluation (Mitamura et al., 2017) as *Monolingual_SOTA*. We also include a vanilla embedding projection based method for comparison (denoted as *Embedding_proj*).

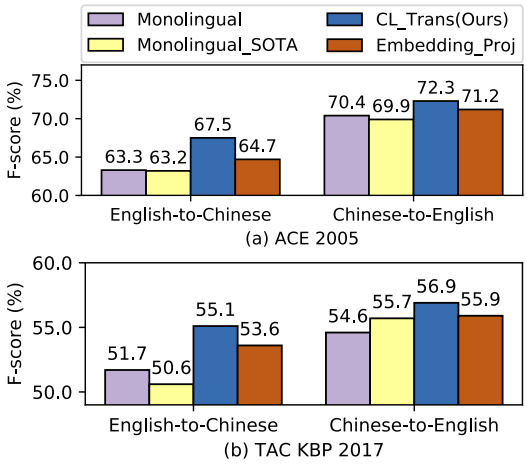


Figure 4: Experimental results on investigating cross-lingual transfer concerning different language directions.

Figure 4 summarizes the results. From the results, 1) our cross-lingual approach *CL_Trans* outperforms two monolingual systems (+2.95% on F1 on the average) and the vanilla embedding projection based method (+1.60% on F1), in the four evaluations. This justifies the effectiveness of our approach concerning cross-lingual in different language directions. 2) Additionally, our cross-lingual approach is more effective for Chinese (+3.80% on F1) than for English (+2.10% on F1). This is understandable as the number of English examples is much larger than that of Chinese examples (5,285 v.s. 2,710 in ACE 2005, and 24,979 v.s. 10,630 in TAC KBP 2017). 3) We obtain interesting findings by investigating each event type. For example, in TAC KBP 2017, the type of "contact/correspondence" has only 167 samples in Chinese but 996 samples in English. By adopting cross-lingual training, our approach leads to an improvement of 15.3% (from 10.3% to 25.6%) in Chinese ED for this type, compared with the monolingual approach. This proves that our method can handle the annotation sparseness problem in the target language.

#### 4.2.2 Exploring the Annotation-Scarce Scenario

We next investigate the annotation-poor scenario, where the source language is set as English and the target language is set as Chinese to compare with previous works. In this scenario, we assume that only a few of annotated documents are available in

[1] https://github.com/facebookresearch/MUSE

Chinese.

**In Comparison to Monolingual ED Models.** We first compare our cross-lingual approach with the existing monolingual ED models, including *CNN* (Nguyen and Grishman, 2015) which employs Convolutional Neural Networks for the task and *Hbrid* (Feng et al., 2016) which combines CNN with Recurrent Neural Networks (RNN) for ED. Figure 5 presents the experimental results, where the number of the available Chinese documents ranges from 0 to 50.

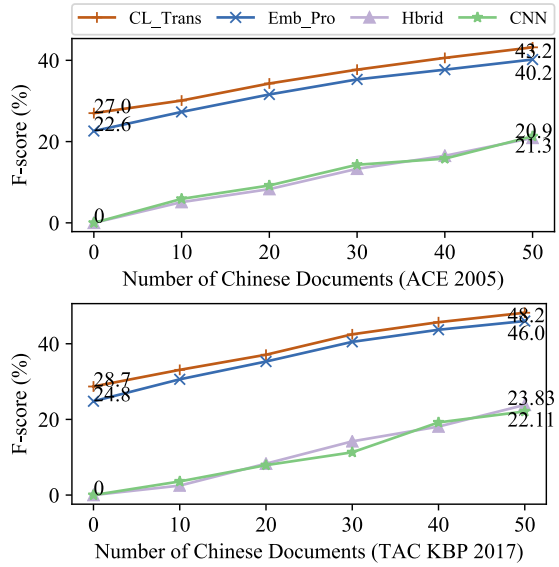


Figure 5: The comparison to monolingual ED models for Chinese ED in the annotation-poor scenario.

From the results, our approach demonstrates a definite advantage over monolingual ED models in the annotation-poor scenario. Particularly, when there is no Chinese training document available (i.e., in the unsupervised cross-lingual transfer scenario), our model achieves a performance of 27.0% on F1 in ACE, and 28.7% on F1 in TAC KBP, while supervised ED methods completely fail. Additionally, our approach can consistently outperform the embedding-projection method.

**In Comparison to Cross-Lingual Models.** We next compare our model to the existing cross-lingual ED methods, including 1) LexMap (Hsi et al., 2016), which combines embedding projection method with multilingual feature extraction to perform cross-lingual ED, and 2) MTED (Zhu et al., 2014), which uses a MT system to translate the training examples in source language to obtain additional data for training. In our re-implementation, we employ OpenNMT (Klein et al.) as the translation model and we use OpenSubtitles (Lison and Tiedemann, 2016) to train it. To ensure comparability of results, we use the setting of (Hsi et al., 2016) (i.e., using one-fold data (64 annotated documents) for training) to conduct the experiments.

| Method | Pre. | Rec. | F1 |
|---|---|---|---|
| LexMap (2016) (5k dict.) | **63.5** | 28.8 | 39.6 |
| MTBased (50k sent.) | 11.3 | 16.8 | 13.5 |
| MTBased (200k sent.) | 42.4 | 26.1 | 32.3 |
| MTBased (400k sent.) | 54.7 | **36.2** | 43.7 |
| CL_Trans (5k dict.) | 62.5 | 35.7 | **45.4** |

Table 1: The comparison to cross-lingual ED models for Chinese ED in ACE 2005. (50k sent.) means using 50k parallel sentences to train the MT system.

Table 1 gives the results. From the results, 1) our method outperforms LexMap by a rather large margin (+5.8% on F1). The poor performance of LexMap might be attributed to its feature engineering process, which is often very difficult and requires expert knowledge. 2) Our model behaves competitively to machine translation based method (which are trained on 400k parallel sentences) yet relies on much less parallel resources (a dictionary with a size of 5k).

### 4.2.3 Ablation Study

We conduct ablation study to explore the effects of our different model components. We limit our study in the extremely annotation-poor scenario, that is, we assume there is no training data in the target language (Chinese).

**Exploring Lexical Mapping Method.** To explore our lexical mapping method, we compare the performance of several variant systems retrieving a different number of candidates (ranging from 1 to 5) and the embedding-projection method (Embedding_proj). Note the system retrieving only one candidate actually takes the nearest Chinese neighbor as the word translation. The lexical mapping in it is still context-independent. Table 2 summarizes the results.

From the results, we observe that 1) even though both of CL_Trans (1 cand.) and Embedding_proj are content-independent mapping methods, the former outperforms the latter by a margin (+3.2% on F1). This implies that the embedding-projection method might suffer from the misalignment in the shared embedding space, and enforcing a word-to-word alignment (as in CL_Trans (1 cand.)) could alleviate this problem to some ex-

| Method | Pre. | Rec. | F1 |
|---|---|---|---|
| Embbeding_Proj | 26.0 | 20.0 | 22.6 |
| CL_Trans (1 cand.) | 31.2 | 21.4 | 25.4 |
| CL_Trans (2 cand.) | 31.7 | 22.3 | 26.2 |
| CL_Trans (3 cand.) | **32.0** | 23.4 | **27.0** |
| CL_Trans (4 cand.) | 30.7 | **23.6** | 26.7 |
| CL_Trans (5 cand.) | 30.2 | **23.6** | 26.5 |

Table 2: Experimental results in exploring different lexical mapping methods.

| Model | Pre. | Rec. | F1 |
|---|---|---|---|
| CL_Trans_MLP | 20.3 | 16.3 | 18.1 |
| CL_Trans_CNN | 32.5 | 16.3 | 21.7 |
| CL_Trans_Hbrid | 30.4 | 17.6 | 22.3 |
| CL_Trans_Self. | **34.9** | 18.3 | 24.0 |
| CL_Trans_GCN (ours) | 32.0 | **23.4** | **27.0** |
| CL_Trans_GCN_Self | 32.1 | 23.0 | 26.8 |

Table 3: Experimental results in exploring the shared syntactic order event detector.

| English | Chinese Translation Candidates |
|---|---|
| 36,000 | 5.76, 98.8%, 2.53 |
| people | **人(people)**, 人们(folk), 青年人(youngs) |
| died | **死(die)**, 死亡(death), 死去(dying) |
| every | **每个(every)**, 所有(all), 都(all) |
| year | **年(year)**, 年代(years), 年龄(age) |
| from | 以外(beyond), **从(from)**, 外(except) |
| the | 这(this), 整个(total), 完全(completely) |
| flu | 感染(inflection), **流感(flu)**, 疾病(disease) |

Table 4: Translation candidates of each English word. Bold indicates the best-suited translation.

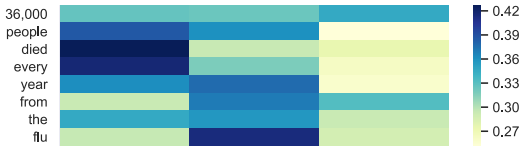


Figure 6: The learned attention weights. Darker color indicates higher weight.

tent. 2) Retrieving more translation candidates could consistently improve Recall. But when too many candidates (e.g., 5) are added, the Precision drops, which harms the overall F1 measure.

**Exploring the Syntactic Order Event Detector.** We compare our syntactic order event detector (CL_Trans_GCN) with several event detectors, including 1) *CL_Trans_MLP*, which employs a feed-forward network as event detector; 2) CL_Trans_CNN, which uses CNNs as the event detector; 3) CL_Trans_Hbrid, which use a hybrid network (Feng et al., 2016) for event detection. We also compared our model with several variants including 4) CL_Trans_Self., which replaces the GCNs with a self-attention network, and 5) CL_Trans_GCN_Self, which combines GCNs with a self-attention network. We train these models on the same translated English data. Table 3 shows the results.

From the results, 1) CL_Trans_MLP, CL_Trans_CNN, and CL_Trans_Hbrid behave poorly, as expected. The reason might be that these models usually employ order-sensitive structures (e.g., CNNs) for ED, which would suffer the word order inconsistency problem when trained on the translated data. 2) CL_Trans_Self. yields relatively good performance. The reason might be that self-attention network could provide each word with a feature vector based on all the words of a sentence, which is also irrespective of the words' positions in a sentence. This could address the word order difference to some extent. 3) Our syntactic order event detector yields the best performance. While, we do not observe salient advantages by combining GCNs with a self-attention network (by comparing CL_Trans_GCN with CL_Trans_GCN_Self).

### 4.3 Beyond English-Chinese Pair

We conduct additional experiments on Spanish to investigate cross-lingual transfer beyond English-Chinese transfer. The Spanish corpus is in TAC-KBP solely, with much smaller size and fewer publish evaluations. Experimental results demonstrate that, our model, without any modification, surpasses the best-reported Spanish system (42.8 (Mitamura et al., 2017) with F1 scores of 44.0 and 43.8 concerning EN $\rightarrow$ SP and CH $\rightarrow$ SP transfer. The value changes to 20.8 and 18.9 in zero transfer scenarios. This demonstrates that our approach is language-independent.

### 4.4 Case Study

We give a case study on the cross-lingual transfer process of a real example in ACE: "*36,000 people* **died** *every year from the flu*". Table 4 and Figure 6 gives the Chinese translation candidates and the learned attention weights respectively.

From the results, the best-suited translations indeed often correspond to larger attention weights, which implies the validity of our approach. In the above example, the Chinese words "从(from)" and "流感(flu)" do not correspond to the nearest

neighbor of “from” and ”flu”, but our content-dependent lexical mapping method enable us to successfully obtain them as the translations.

The case study also poses several future directions for this work. For example, one is how to address the one-to-many mapping between different languages. In the above example, the correct Chinese translation of “every year” should be one single word “每年”, not the combination of two words “每个(every)” and “年(year)”. This calls for more advanced lexical mapping methods.

## 5 Related Work

Event detection (ED) is a hot topic in natural language processing, which has attracted extensive attention in the past few years. Traditionally, the study of ED has focused on monolingual training. The proposed models can be divided into feature-based methods which employ fine-grained features (Ahn, 2006; Ji and Grishman, 2008; Liao and Grishman, 2010; Hong et al., 2011; Li et al., 2013; Li and Ji, 2014), and deep learning-based methods which employ neural networks to automatically learn features for the task (Chen et al., 2015; Nguyen and Grishman, 2015; Nguyen et al., 2016; Liu et al., 2018b; Orr et al., 2018; Liu et al., 2019). Usually, their performance is limited by the amount of labeled data in a specific language.

Cross-lingual ED attempts transfer knowledge between different languages to boost performance. To name a few, (Chen and Ji, 2009) used an English detector to label events on parallel documents to obtain additional data for boosting Chinese ED; (Zhu et al., 2014; Liu et al., 2018a) used machine translation to obtain additional labeled data for training; (Hsi et al., 2016) combined the embedding-projection method with multilingual feature extraction for bilingual ED. Nevertheless, the heavily dependency on parallel resources often limits the applicability of these methods.

Our study also relates to cross-lingual studies in other applications (Guo et al., 2015; Ni et al., 2017; Mayhew et al., 2017; Xie et al., 2018; Lample et al., 2018). These approaches adopted embedding projection based method to achieve cross-lingual transfer and achieved prosing results. However, since the lexical mapping in these methods is usually deterministic and irrespective of contexts, they might not directly fit with cross-lingual ED, where the cross-lingual transfer should be context-dependent.

## 6 Conclusions and Future Work

In this paper, we propose a new cross-lingual approach for event detection, which demonstrates a minimal dependency on parallel resources. Specifically, we propose a context-dependent lexical mapping method to obtain content-dependent translation, and we devise a shared syntactic order event detector to explore the syntactic similarity for multilingual co-training. Experiments demonstrate the effectiveness of our method.

Currently, as our approach is predicated on the availability of syntax trees of training examples, it might not fit with languages which lack syntactic parsers. In the future, we plan to investigate more language-independent patterns in cross-lingual transfer to circumvent this dependency.

## Acknowledgments

This work is supported by the National Key R&D Program of China under Grant 2018YFB1005100, the National Natural Science Foundation of China (No.61533018), the National Natural Science Foundation of China (No.61806201) and the independent research project of National Laboratory of Pattern Recognition. This work is also supported by a grant from Ant Financial Services Group and the CCF-Tencent Open Research Fund. We would like to thank the anonymous reviewers for their valuable feedback.

## References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning About Time and Events*, ARTE ’06, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 167–176. Association for Computational Linguistics.

Zheng Chen and Heng Ji. 2009. Can one language bootstrap the other: A case study on event extraction. In *Proceedings of the NAACL HLT 2009 Workshop on Semi-Supervised Learning for Natural Language Processing*, SemiSupLearn ’09, pages 66–74, Stroudsburg, PA, USA. Association for Computational Linguistics.

Xiaocheng Feng, Lifu Huang, Duyu Tang, Heng Ji, Bing Qin, and Ting Liu. 2016. A language-independent neural network for event detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 66–71, Berlin, Germany. Association for Computational Linguistics.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1234–1244. Association for Computational Linguistics.

Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. 2011. Using cross-entity inference to improve event extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1127–1136, Portland, Oregon, USA. Association for Computational Linguistics.

Andrew Hsi, Yiming Yang, Jaime Carbonell, and Ruochen Xu. 2016. Leveraging multilingual training for limited resource event extraction. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1201–1210. The COLING 2016 Organizing Committee.

Heng Ji. 2009. Cross-lingual predicate cluster acquisition to improve bilingual event extraction by inductive learning. In *Proceedings of the Workshop on Unsupervised and Minimally Supervised Learning of Lexical Semantics*, UMSLLS '09, pages 27–35, Stroudsburg, PA, USA. Association for Computational Linguistics.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *Proceedings of ACL-08: HLT*, pages 254–262, Columbus, Ohio. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Thomas N. Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907.

G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush. OpenNMT: Open-Source Toolkit for Neural Machine Translation. *ArXiv e-prints*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Guillaume Lample, Alexis Conneau, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. In *International Conference on Learning Representations*.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 402–412. Association for Computational Linguistics.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 73–82, Sofia, Bulgaria. Association for Computational Linguistics.

Shasha Liao and Ralph Grishman. 2010. Using document level cross-event inference to improve event extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 789–797. Association for Computational Linguistics.

Hongyu Lin, Yaojie Lu, Xianpei Han, and Le Sun. 2018. Nugget proposal networks for chinese event detection. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1565–1574. Association for Computational Linguistics.

Pierre Lison and Jörg Tiedemann. 2016. Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles. In *LREC*.

Jian Liu, Yubo Chen, and Kang Liu. 2019. Exploiting the Ground-Truth: An Adversarial Imitation Based Knowledge Distillation Approach for Event Detection. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):6754–6761.

Jian Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2018a. Event detection via gated multilingual attention mechanism. *Proceedings of the AAAI Conference on Artificial Intelligence*.

Shaobo Liu, Rui Cheng, Xiaoming Yu, and Xueqi Cheng. 2018b. Exploiting contextual information via dynamic memory network for event detection. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1030–1035. Association for Computational Linguistics.

Shulin Liu, Yubo Chen, Shizhu He, Kang Liu, and Jun Zhao. 2016. Leveraging framenet to improve automatic event detection. In *Proceedings of the*

*54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2134–2143, Berlin, Germany. Association for Computational Linguistics.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J Bethard, and David McClosky. 2014. The {Stanford} {CoreNLP} Natural Language Processing Toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Stephen Mayhew, Chen-Tse Tsai, and Dan Roth. 2017. Cheap translation for cross-lingual named entity recognition. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2536–2545, Copenhagen, Denmark. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA. Curran Associates Inc.

Teruko Mitamura, Zhengzhong Liu, and Eduard H. Hovy. 2017. Events detection, coreference and sequencing: What's next? overview of the tac kbp 2017 event track. In *TAC*.

Thien Nguyen and Ralph Grishman. 2018. Graph convolutional networks with argument-aware pooling for event detection. *Proceedings of the AAAI Conference on Artificial Intelligence*.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 300–309, San Diego, California. Association for Computational Linguistics.

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 365–371, Beijing, China. Association for Computational Linguistics.

Jian Ni, Georgiana Dinu, and Radu Florian. 2017. Weakly supervised cross-lingual named entity recognition via effective annotation and representation projection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1470–1480. Association for Computational Linguistics.

Walker Orr, Prasad Tadepalli, and Xiaoli Fern. 2018. Event detection with neural networks: A rigorous empirical evaluation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 999–1004, Brussels, Belgium. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Jiateng Xie, Zhilin Yang, Graham Neubig, Noah A. Smith, and Jaime Carbonell. 2018. Neural cross-lingual named entity recognition with minimal resources. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 369–379. Association for Computational Linguistics.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1006–1011. Association for Computational Linguistics.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *COLING 2000 Volume 2: The 18th International Conference on Computational Linguistics*.

Zhu Zhu, Shoushan Li, Guodong Zhou, and Rui Xia. 2014. Bilingual event extraction: a case study on trigger type determination. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 842–847, Baltimore, Maryland. Association for Computational Linguistics.

# KnowledgeNet: A Benchmark Dataset for Knowledge Base Population

**Filipe Mesquita, Matteo Cannaviccio, Jordan Schmidek, Paramita Mirza**
Diffbot Technologies Corp.
Menlo Park, California
{filipe,matteo,jay,paramita}@diffbot.com

**Denilson Barbosa**
Department of Computing Science
University of Alberta
Edmonton, Canada
denilson@ualberta.ca

## Abstract

KnowledgeNet is a benchmark dataset for the task of automatically populating a knowledge base (Wikidata) with facts expressed in natural language text on the web. KnowledgeNet provides text exhaustively annotated with facts, thus enabling the holistic end-to-end evaluation of knowledge base population systems as a whole, unlike previous benchmarks that are more suitable for the evaluation of individual subcomponents (e.g., entity linking, relation extraction). We discuss five baseline approaches, where the best approach achieves an F1 score of 0.50, significantly outperforming a traditional approach by 79% (0.28). However, our best baseline is far from reaching human performance (0.82), indicating our dataset is challenging. The KnowledgeNet dataset and baselines are available at https://github.com/diffbot/knowledge-net

## 1 Introduction

Knowledge Bases (KBs) are valuable resources for developing intelligent applications, including search, question answering, data integration, and recommendation systems. High-quality KBs still rely almost exclusively on human-curated structured or semi-structured data. Such a reliance on human curation is a major obstacle to the creation of comprehensive, always-up-to-date KBs.

KB population (KBP) is the task of automatically augmenting a KB with new facts. Traditionally, KBP has been tackled with datasets for individual components to be arranged in a pipeline, typically: (1) entity discovery and linking (Ji et al., 2017; Shen et al., 2015) and (2) relation extraction (Angeli et al., 2015; Zhang et al., 2017). Entity discovery and linking seeks to recognize and disambiguate proper names in text that refer to entities (e.g., people, organizations and locations) by linking them to a reference KB. Relation extraction seeks to detect facts involving two entities (or an entity and a literal, such as a number or date).

KnowledgeNet is a benchmark dataset for populating a KB (Wikidata) with facts expressed in natural language on the web. KnowledgeNet facts are of the form (*subject*; property; *object*), where *subject* and *object* are linked to Wikidata. For instance, the dataset contains text expressing the fact (*Gennaro Basile*[1]; RESIDENCE; *Moravia*[2]), in the passage:

> "Gennaro Basile was an Italian painter, born in Naples but active in the German-speaking countries. *He* settled at Brünn, in *Moravia*, and lived about 1756..."

KnowledgeNet's main goal is to evaluate the overall task of KBP rather than evaluating its subcomponents in separate. We refer to this type of evaluation as *end-to-end*. The dataset supports the end-to-end evaluation of KBP systems by exhaustively annotating all facts in a sentence. For instance, the dataset contains all RESIDENCE facts (two) from the sentence "He settled at Brünn, in Moravia, and lived about 1756". This allows our evaluation to assess precision and recall of RESIDENCE facts extracted from this sentence.

A popular initiative to evaluate KBP is the Text Analysis Conference, or TAC (Getman et al., 2018). TAC evaluations are performed manually and are hard to reproduce for new systems. Unlike TAC, KnowledgeNet employs an automated and reproducible way to evaluate KBP systems at any time, rather than once a year. We hope a faster evaluation cycle will accelerate the rate of improvement for KBP.

In addition to providing an evaluation benchmark, KnowledgeNet's long-term goal is to provide exhaustively annotated training data at large

[1] http://www.wikidata.org/wiki/Q1367602
[2] http://www.wikidata.org/wiki/Q43266



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 749–758, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

scale. Our goal for the coming years is to annotate *100,000 facts for 100 properties*. To accomplish this goal, we propose a new framework for annotating facts with high accuracy and low effort.

**Contributions.** Our contributions are as follows. We introduce KnowledgeNet, a benchmark dataset for end-to-end evaluation of KBP systems (Section 3). We propose a new framework for exhaustive yet efficient annotation of facts (Section 3.1). We implement five baseline approaches that build upon state-of-the-art KBP systems (Section 4). Finally, we present an experimental analysis of our baseline approaches, comparing their performance to human performance (Section 5).

## 2 Related Work

KBP has traditionally been tackled with pipeline systems. For instance, Stanford's TAC 2015 winning system employs the following pipeline: named entity recognition (NER) → entity linking → relation extraction (Angeli et al., 2015). Stanford's latest TAC system continues to use the same pipeline architecture with one additional component: coreference resolution (Chaganty et al., 2017).

The main shortcoming of pipeline systems is error propagation. Mistakes made by components in the beginning of the pipeline are propagated to the final output of the system, negatively affecting the overall precision and recall. For instance, our experiments show that the pipeline employed by Stanford's TAC 2015 winning system can achieve a maximum recall of 0.32 in KnowledgeNet[3].

End-to-end systems (Liu et al., 2018; Miwa and Bansal, 2016) are a promising solution for addressing error propagation. However, a major roadblock for the advancement of this line of research is the lack of benchmark datasets. We hope KnowledgeNet can help support this line of research.

### 2.1 Datasets

TAC is a series of evaluation workshops organized as several tracks by NIST (Getman et al., 2018). The Cold Start track provides an end-to-end evaluation of KBP systems, while other tracks focus on subtasks (e.g., entity disambiguation and linking). The Cold Start track is the current standard to evaluate KBP systems. To compete in this track, participants have a limited time window to submit the results of their KBP systems. After this window, the systems are evaluated by pooling facts extracted by the contestants. Despite its effectiveness for running a contest, this methodology has been shown to be biased against new systems, which are not part of the pooling (Chaganty et al., 2017). TAC also manually evaluates a system's "justification", a span of text provided as evidence for a fact. A correct fact with an incorrect justification is considered invalid. Therefore, reproducing TAC's evaluation for new systems is challenging.

We propose KnowledgeNet as an automated and reproducible alternative to TAC's evaluation. Before creating KnowledgeNet, we considered using one of the datasets presented in Table 1. We compare these datasets according to five criteria that we consider desirable for a KBP benchmark dataset:

- **Human annotation**: the dataset should be annotated by (multiple) humans to support accurate evaluation.
- **Exhaustive annotation:** for each property, the dataset should exhaustively enumerate all facts of that property that are expressed in the text. Exhaustive annotation allows measuring true precision and recall for a system.
- **Text annotation:** the dataset should contain text spans for entities involved in a fact. This allows evaluating whether the text expresses an extracted fact (as alternative to TAC's manual justification assessments).
- **Links to reference KB:** the dataset should contain a link to the reference KB for entities involved in every fact (or indicate that such an entity doesn't exist in the KB). This allows the evaluation to confirm that the reference KB is being correctly populated.
- **Cross-sentence facts:** the dataset should contain facts involving entities whose *names* never appear in the same sentence. This is because a significant portion of facts expressed in text require coreference resolution of entity mentions spanning multiple sentences.

ACE 2005[4] is a popular dataset for end-to-end relation extraction systems (Li and Ji, 2014;

[3]Maximum recall is the recall of candidate facts, which are used as input to the last component of the pipeline (relation extraction).

[4]https://catalog.ldc.upenn.edu/LDC2006T06

| Dataset | **KnowledgeNet** | ACE | TAC | TACRED | FewRel | DocRED | GoogleRE | T-REx |
|---|---|---|---|---|---|---|---|---|
| Human annotation | yes | yes | yes | yes | yes | yes | yes | no |
| Exhaustive annotation | yes | yes | no | no | no | no | no | no |
| Exhaus. anno. sentences | 9,000 | 11,000 | N/A | N/A | N/A | N/A | N/A | N/A |
| Text span annotation | yes | yes | no | yes | yes | yes | no | yes |
| Links to reference KB | yes | yes | yes | no | yes | no | yes | yes |
| Cross-sentence facts | yes | yes | yes | no | no | yes | yes | yes |
| Annotated facts | 13,000 | 8,000 | 84,000 | 22,000 | 56,000 | 56,000 | 40,000 | 11M |
| Properties | 15 | 18 | 41 | 41 | 100 | 96 | 5 | 353 |
| New KB facts annotated | 77% | 100% | 100% | 100% | 0% | 100% | 0% | 0% |

Table 1: A dataset comparison according to our criteria for a desirable KBP benchmark dataset. "Exhaus. anno. sentences" shows the number of exhaustively annotated sentences. "New KB facts annotated" shows the percentage of annotated facts that can be found in the reference KB. Most datasets contain only facts with no links to a reference KB (100% new facts) or contain only facts that exist in the KB (0% new facts).

Miwa and Bansal, 2016). According to our criteria, ACE might seem the most complete benchmark dataset for end-to-end evaluation of KBP. It exhaustively annotates every sentence of 599 documents with mentions, coreference chains and facts for 18 properties. Also, ACE has been independently extended with links to Wikipedia (Bentivogli et al., 2010). However, a closer look at ACE's annotations reveals that most of them are ill-suited for general-purpose KBs. These annotations include facts about broad properties (e.g., part-whole, physical location) or mentions that do not refer to named entities (e.g., "two Moroccan men", "women on the verge of fainting", "African immigrants who came ashore Thursday"). Perhaps not coincidently, we are unaware of any work using it for the purpose of evaluating a KBP system.

Our annotation framework (Section 3.1) is inspired by ACE's framework but tailored towards KBP. First, we only annotate mentions that refer to named entities. Second, while our annotation is exhaustive, we focus on annotating sentences rather than documents, eliminating the need to annotate every fact described in the entire document. Such a requirement creates a significant imbalance in the number of annotations per property. For instance, the most popular property from ACE has 1,400 annotated facts, while the majority of properties from ACE have less than 300 annotated facts. This might explain why most relation extraction evaluations use only 6 properties from ACE.

Annotating every sentence with facts for all properties is also detrimental to the incremental nature of KnowledgeNet. Adding one property to the dataset would require an annotator to re-annotate every sentence in the dataset. In contrast, our framework selects a limited set of sentences to be annotated for a particular property. The remaining sentences are ignored during annotation and evaluation. As a consequence, our annotation framework allows incremental annotation of new properties and is better suited for the goal of annotating 100,000 facts for 100 properties.

**Datasets employing non-exhaustive annotation.** Recent datasets like T-REx automatically annotate facts in text as a way to produce training data cheaply. This is performed by aligning facts in the KB to sentences referring to them (Elsahar et al., 2018). Other datasets go further and use human annotators to label every alignment as correct or incorrect. These semi-supervised datasets include TACRED (Zhang et al., 2017), GoogleRE[5], FewRel (Han et al., 2018) and DocRED (Yao et al., 2019). Annotations created in this way are useful for training KBP systems. However, they do not provide an exhaustive annotation of facts, which is needed for end-to-end evaluation of KBP. For instance, Zhang et al. (2017) train their KBP system with TACRED, but rely on TAC to evaluate the system.

## 3 KnowledgeNet Dataset

This section discusses the first release of KnowledgeNet and our annotation framework. The documents in this first release are either DBpedia abstracts (i.e., first paragraphs of a Wikipedia page) or short biographical texts about a person or organization from the web. These web texts were collected using the Diffbot Knowledge Graph[6].

Table 2 presents the number of annotated facts for each property. We chose 9,073 sentences

[5] https://code.google.com/archive/p/relation-extraction-corpus/downloads

[6] https://www.diffbot.com/

| Property | Facts | Sent. | Relevant |
|---|---|---|---|
| DATE_OF_BIRTH (PER–DATE) | 761 | 731 | 468 |
| DATE_OF_DEATH (PER–DATE) | 664 | 512 | 347 |
| RESIDENCE (PER–LOC) | 1,456 | 796 | 387 |
| BIRTHPLACE (PER–LOC) | 1137 | 936 | 407 |
| NATIONALITY (PER–LOC) | 639 | 801 | 396 |
| EMPLOYEE_OF (PER–ORG) | 1,625 | 650 | 543 |
| EDUCATED_AT (PER–ORG) | 951 | 463 | 335 |
| POLITICAL_AFF. (PER–ORG) | 635 | 537 | 318 |
| CHILD_OF (PER–PER) | 888 | 471 | 296 |
| SPOUSE (PER–PER) | 1,338 | 504 | 298 |
| DATE_FOUNDED (ORG–DATE) | 500 | 543 | 315 |
| HEADQUARTERS (ORG–LOC) | 880 | 564 | 296 |
| SUBSIDIARY_OF (ORG–ORG) | 544 | 481 | 299 |
| FOUNDED_BY (ORG–PER) | 764 | 558 | 346 |
| CEO (ORG–PER) | 643 | 526 | 350 |
| Total | 13,425 | 9,073 | 5,423 |

Table 2: KnowledgeNet properties and their number of annotated facts and sentences. "Relevant" indicates the number of relevant sentences (i.e., those with one or more annotated facts). Subjects and objects belong to one of the following types: person, organization, location and date.

from 4,991 documents to be exhaustively annotated with facts about a particular property of interest. Because our annotation is exhaustive, negative examples of facts can be automatically generated. In total, KnowledgeNet comprises 13,425 facts from 15 properties.

**Holdout test set.** We split the documents into five folds in a round-robin manner, keeping the fifth fold (20% of the dataset) as the test set. To preserve the integrity of the results, we will release the test set without annotations and will provide a service through which others can evaluate their KBP systems. In our experiments, we used folds 1-3 for training and fold 4 for development and validation, including hyperparameter tuning.

### 3.1 Dataset Annotation

The dataset has been generated by multiple annotators using a new multi-step framework. We conjecture our framework can help annotators produce higher quality annotations by allowing them to focus on one small, more specific task at a time. The annotation consists of four different steps: (1) fetch sentences, (2) detect mentions, (3) classify facts and (4) link entities.

**Step 1: Fetch sentences.** We employ two methods of choosing a sentence for annotation. The first method leverages T-REx's automatic alignments (Elsahar et al., 2018) to find sentences that are likely to describe facts from Wikidata. The




(a) Interface to detect mentions of an entity type.

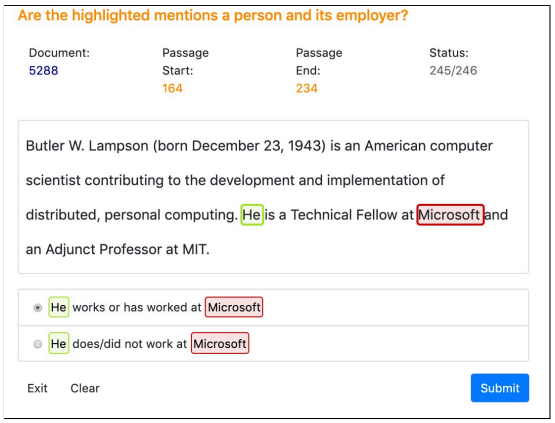


(b) Interface to classify facts.

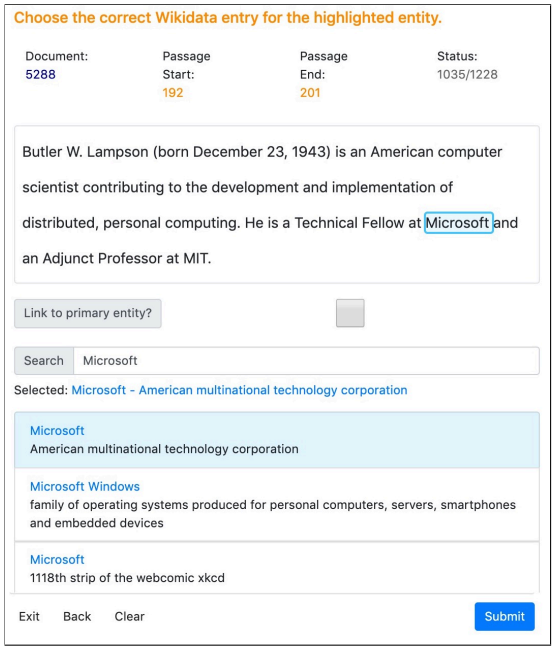


(c) Interface to link a mention to a Wikidata entity.

Figure 1: Interface for Steps 2-4 of our framework. Step 1 fetches sentences to be exhaustively annotated for one property. The remaining steps guide annotators to detect entity mentions, facts and links in each sentence.

second method chooses sentences that contain a keyword that might indicate the presence of a fact for a property (e.g., *"born"* for DATE_OF_BIRTH). We have chosen these keywords by leveraging Wikidata's "also known as" values for properties as well as WordNet synonyms. By using these keywords, we prevent the dataset to be exclusively annotated with facts that are known in Wikidata. In fact, only 23% of facts annotated in this release are present in Wikidata.

For each fetched sentence, an annotator decides whether the sentence is *relevant* for the property of interest (i.e., whether this sentence describes one or more facts for this property). Relevant sentences go through steps 2 through 4; while irrelevant sentences are kept to be used for detecting incorrectly extracted facts (i.e., false positives).

It is worth noting that this step might not fetch some relevant sentences. Our framework does not require all relevant sentences to be annotated and does not penalize systems for extracting facts from sentences that were not annotated.

**Step 2: Detect mentions.** In this step, we ask annotators to highlight entity names (Figure 1a). We consider only entities whose type is relevant to the property being annotated. For instance, an annotator will only be asked to highlight names of people and organizations when annotating the property FOUNDED_BY. Pronouns are automatically annotated with a gazetteer. To decrease the likelihood of missing a mention, we consider the *union* of mentions highlighted by two annotators for the following step.

**Step 3: Classify facts.** We ask annotators to classify a candidate fact (i.e., a pair of mentions) in a sentence as a positive or negative example for a property (Figure 1b). Each candidate fact is annotated by at least two annotators. A third annotator breaks the tie when the first two annotators disagree.

We follow ACE's *reasonable reader rule*, which states that a fact should only be annotated when there is no reasonable interpretation of the sentence in which the fact does not hold. In other words, annotators are asked to only annotate facts that are either explicitly stated in the sentence or inferred with absolute certainty from the sentence alone (i.e., without using external world knowledge).

**Step 4: Link entities.** Finally, we ask annotators to link every mention involved in a fact to a single Wikidata entity. In this step, annotators can read the entire document and resolve mentions (e.g., pronouns) that refer to names in other sentences. Every mention is annotated by at least two annotators. When there is disagreement, we ask other annotators to join in the process until consensus is reached. In total, excluding the properties having literal objects (Table 2) we can assign a link to both subject and object for 52% of the facts.

**Inter-annotator agreement.** A total of five annotators have contributed to KnowledgeNet so far. In Step 3, the initial two annotators have annotated 33,165 candidate facts with 96% agreement. They disagreed on 1,495 candidate facts, where 599 have been deemed positive by a third annotator. In Step 4, the initial two annotators have annotated 13,453 mentions with agreement of 93%. The remaining 7% of mentions were resolved with additional annotators.

**Timing.** On average, annotating a sentence for one property takes 3.9 minutes. This total time includes two annotators (plus additional annotators for tiebreaking). It also includes inspecting sentences that express no facts and therefore do not go through steps 2-4 (but are included in the dataset and are helpful for assessing false positives). The most expensive step is Step 3 (40% of the total time), followed by Step 4 (28%), Step 2 (22%) and Step 1 (10%).

### 3.2 Limitations

Our first release is comparable to other benchmarks in size (e.g., ACE 2005), but it is perhaps insufficient to train data-hungry models. This is by design. Most organizations do not have the resources to produce tens of thousands of examples for each property of interest. As we expand the number of properties to achieve our goal of annotating 100,000 facts, we expect to keep the number of facts per property to around a thousand. In this way, we hope to promote approaches that can learn from multiple properties, requiring less annotations per property. We also hope to promote approaches using KnowledgeNet together with semi-supervised or unsupervised datasets for training.

Another limitation of our first release is the focus on individual sentences. Currently, our framework can only annotate a fact when the subject and

the object are explicitly mentioned by a name or pronoun in a sentence. Others have reported that the majority of facts fall into this category. For example, the authors of DocRED report that 41% of facts require reasoning over multiple sentences in a document (Yao et al., 2019). This indicates that a fact's subject and object are mentioned by their full name in a single sentence 59% of the time. The percentage of facts that can be annotated in KnowledgeNet is significantly higher than 59%. This is because our framework can also annotate facts that require resolving (partial) names and pronouns referring to full names in other sentences. These facts are particularly common in our document collection.

## 4 Baseline Approaches

This section presents five baseline approaches for KBP. We evaluate these approaches and compare their performance relative to human annotators in Section 5.

Figure 2 illustrates the architecture shared by our five baseline approaches. We start by splitting a given document into sentences. For each sentence, we detect entity mentions using a named entity recognizer (NER) and a gazetteer for pronoun mentions and their type (e.g., person, organizations, location). We also detect coreference chains, that is, groups of mentions within a document that refer to the same entity. Figure 2 illustrates how coreference chains help disambiguate pronouns and partial names by clustering them together with the full name of an entity. Finally, we link these coreference chains to Wikidata entities.

Next, we produce candidate facts by considering pair of mentions from the same sentence, as illustrated in Figure 2. The relation extraction component makes the final decision on whether a candidate fact is expressed by the text.

### 4.1 Relation Extraction

Figure 2 illustrates our relation extraction model. This model follow the literature by using a Bi-LSTM network (Miwa and Bansal, 2016; Xu et al., 2015; Zhou et al., 2016; Zhang et al., 2017), which is effective in capturing long-distance dependencies between words. We train a single multi-task model for all properties using both positive examples (i.e., annotated facts) and automatically generated negative examples.

The model outputs two values for each property. The first value represents the likelihood of the subject and object *mentions* (i.e., text spans) to be correct, while the second value represents the likelihood of the subject and object *links* to be correct. We learn individual thresholds for each value and property. When both values are above the threshold, the system outputs the fact with links. When the first value is above the threshold and the second value is below the threshold, we output the fact without links.

**Features.** Figure 3 illustrates features encoding syntactic and positional information, which are concatenated to the embedding of every word.

1. **Enriched NER:** NER label for names (using a NER system) and pronouns (using gazetteers for each type).
2. **Mention distance:** distance between each word and the subject and object mention, following Zhang et al. (2017).
3. **Shortest dependency path (SDP) length:** number of edges in the SDP between the word and the subject and object.
4. **SDP distance:** number of edges separating the word to the closest word in the SDP between the subject and object.
5. **Coreference confidence:** confidence score of the coreference resolution system that a word refers to the subject and object.
6. **Coreference distance:** distance to the closest mention in the coreference chain of the subject and object.
7. **Coreference SDP length:** number of edges in the SDP between the word and the closest mention in the subject and object chain.
8. **Coreference SDP distance:** number of edges separating the word to the closest word in the SDP between the subject and object coreference chains.
9. **KB entity types:** entity types for the subject and object from Wikidata.
10. **KB properties:** the property $p$ where (*subject*; $p$; *object*) exists in Wikidata (when both the subject and object have links).

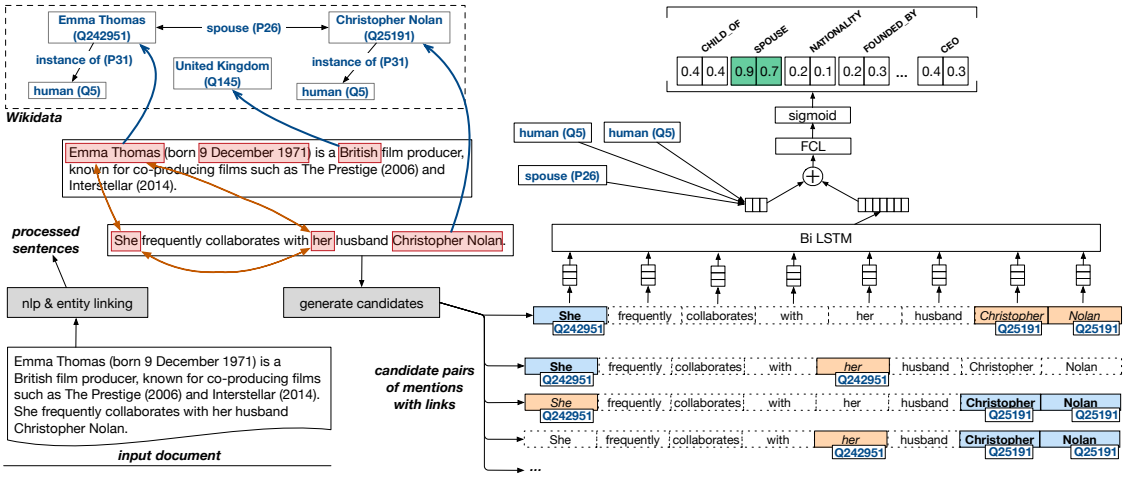


Figure 2: The architecture of our baseline approaches, illustrated with an example. Red arrows and boxes represent coreference chains and blue arrows represent links to Wikidata. The subject and object of candidate facts are highlighted in bold (blue) and italics (orange), respectively.

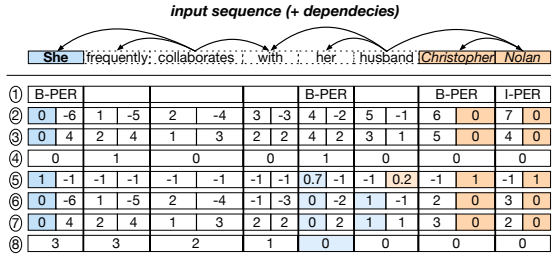


Figure 3: Features representing the relationships between the words. Significant relationships with the subject and object are highlighted in blue and orange, respectively.

Features 9 and 10 are generated by querying Wikidata and are relative to a single entity pair. We concatenate those features to the Bi-LSTM output, as illustrated in Figure 2.

### 4.2 Baseline Approaches

We propose five baselines obtained by improving the candidate generation and relation extraction components.

**Baseline 1.** Our first baseline is a standard pipeline approach inspired by the TAC 2015 Cold Start winning system (Angeli et al., 2015). It generates candidate mentions by using NER and the pronoun gazetteers. For mentions of the correct type (e.g., person for the property SPOUSE), the system then links these mentions using an entity linking system. The relation extraction component uses features 1-4.

**Baseline 2.** Our second baseline adds coreference resolution. This baseline is inspired by Stanford's TAC 2017 system (Chaganty et al., 2017). We leverage coreference chains to both increase the number of candidate mentions linked to KB entities (e.g., pronouns) as well as to introduce additional features. This model uses features 1-8.

**Baseline 3.** Our third baseline adds features 9 and 10 to the relation extraction model. These features leverage Wikidata information for the linked entities, such as entity types and known facts.

**Baseline 4.** Our fourth baseline seeks to decrease error propagation by allowing more candidate facts to be evaluated by the relation extraction component. This is done in two ways. First, Baseline 4 uses all mentions regardless of their NER type when creating candidate facts. Second, this baseline adds a candidate link to mentions that had no candidate link in Baseline 1-3 (due to incorrect coreference chains). This is done by choosing a link outside of the mention's coreference chain that maximizes a combination of entity linking score and coreference resolution score.

**Baseline 5.** Our final baseline seeks to improve the relation extraction component by employing BERT's pre-trained representations (Devlin et al., 2018) in addition to all other features. To produce a contextual representation for every word, we learn a linear weighted combination of BERT's 12 layers, following Peters et al. (2019).

### 4.3 Implementation

All our baseline systems follow the same architecture (Figure 2). We use spaCy[7] for the NLP pipeline (sentence splitting, tokenization, POS tagging, dependency parsing, NER), Hugging Face's coreference resolution system[8], and the Diffbot Entity Linker[9] for entity linking.

For relation extraction we implement a standard BiLSTM network with two 500-dimensional hidden layers. We use spaCy pre-trained word embeddings (size 300) concatenated with additional features illustrated in Figure 3. The output of the BiLSTM network is concatenated with features from Wikidata (Features 9-10).

We train all the networks using mini-batches of 128 examples and Adam optimizer (Kingma and Ba, 2015) with a learning rate of 0.001. We use the fourth fold of the dataset as validation set, selecting the model that minimize the loss function value. The same validation set is used to find thresholds for the output values that maximize the F1 score for each property.

## 5 Experiments

Table 3 presents the performance of our baseline systems compared to the human performance. We report precision ($P$), recall ($R$) and F-score ($F1$):

$$P = \frac{\text{correctly extracted facts}}{\text{extracted facts}},$$

$$R = \frac{\text{correctly extracted facts}}{\text{annotated facts}},$$

$$F1 = \frac{2 \cdot P \cdot R}{P + R}.$$

We evaluate our baseline systems from two perspectives. The *text evaluation* deems an extracted fact correct when the text spans of the subject and object overlap with the text spans of a ground truth fact. The *link evaluation* deems an extracted fact correct when the links of the subject and object match the links of a ground truth fact. In the link evaluation, we consider only facts where both the subject and object links are present.

**Human performance.** To measure the human performance on the end-to-end KBP task, one of our annotators was asked to enumerate all facts described in a sample of the test sentences. We report the performance of our annotator in Table 3.

[7] https://spacy.io/
[8] https://huggingface.co/coref/
[9] https://diffbot.com/

| System | Text evaluation | | | Link evaluation | | |
|---|---|---|---|---|---|---|
| | P | R | F1 | P | R | F1 |
| Baseline 1 | 0.44 | 0.64 | 0.52 | 0.31 | 0.26 | 0.28 |
| Baseline 2 | 0.49 | 0.64 | 0.55 | 0.37 | 0.32 | 0.34 |
| Baseline 3 | 0.47 | 0.66 | 0.55 | 0.35 | 0.37 | 0.36 |
| Baseline 4 | 0.60 | 0.65 | 0.62 | 0.51 | **0.48** | 0.49 |
| Baseline 5 | **0.68** | **0.70** | **0.69** | **0.53** | **0.48** | **0.50** |
| Human | 0.88 | 0.88 | 0.88 | 0.81 | 0.84 | 0.82 |

Table 3: The performance of our baseline approaches is well below human performance.

A closer look at the annotator's mistakes shows that 32% of the mistakes are due to incorrect annotations in KnowledgeNet (i.e., the annotator is actually correct). The remaining mistakes (68%) are mostly due to the annotator entering an incorrect fact (30%) or missing a link on a correct fact (18%). These results show that our annotation framework produces significantly better annotations than individual annotators working without our framework.

**Baseline performance.** Table 3 presents the performance of our baselines. Our best baseline (Baseline 5) significantly outperforms the standard pipeline approach (Baseline 1) in both the text and link evaluation. However, the performance of Baseline 5 is well below the human performance. The most impactful improvements over Baseline 1 are due to ($a$) incorporating coreference when choosing candidate links for pronouns in Baseline 2; ($b$) allowing more candidate facts and links to be classified by the relation extraction component in Baseline 4; and ($c$) incorporating BERT's pre-trained model in Baseline 5.

Table 4 shows the "maximum recall" for each baseline (i.e., recall of candidate facts used as input for the relation extraction component). These results indicate that error propagation significantly limits recall. Our best baseline shows higher maximum recall due to coreference resolution (introduced in Baseline 2) and removing the filtering of candidate facts based on NER types (introduced in Baseline 4). The low maximum recall for link evaluation is mainly due to incorrect candidate links, which can only be omitted (but not fixed) in our baselines.

## 6 Conclusion

We introduce KnowledgeNet, an end-to-end benchmark dataset for populating Wikidata with facts expressed in natural language text on the

| System | Text evaluation | Link evaluation |
|---|---|---|
| | Maximum Recall | Maximum Recall |
| Baseline 1 | 0.80 | 0.33 |
| Baseline 2 | 0.80 | 0.37 |
| Baseline 3 | 0.80 | 0.37 |
| Baseline 4 | 0.90 | 0.59 |
| Baseline 5 | 0.90 | 0.59 |

Table 4: The relation extraction component's recall is limited by error propagation. Maximum recall is the recall of the candidate facts used as input for the relation extraction component on the dev set.

web. We build KnowledgeNet using a new multi-step framework that helps human annotators to produce high-quality annotations efficiently. We also introduce five baseline systems and evaluate their performance. Our best baseline outperforms a traditional pipeline approach by 79% (F1 score of 0.50 vs. 0.28). Human performance is significantly higher (0.82), indicating that KnowledgeNet can support further research to close this gap.

Our experiments show that the traditional pipeline approach for KB population is notably limited by error propagation. Performance gains achieved by our best baseline are mainly due to more candidates being passed along to the final pipeline component (relation extraction), allowing this component to fix errors made by previous components. A closer inspection reveals that even our best baseline is fairly limited by error propagation and can only achieve a maximum recall of 0.59. These results indicate that end-to-end models might be a promising alternative to the traditional pipeline approach.

## Acknowledgments

We would like to thank Veronica Romualdez and Geraldine Fajardo for their diligent annotation work. We would also like to thank Mike Tung, Zhaochen Guo, Sameer Singh and the anonymous reviewers for their helpful comments. This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) and Diffbot.

## References

Gabor Angeli, Victor Zhong, Danqi Chen, Arun Tejasvi Chaganty, Jason Bolton, Melvin Jose Johnson Premkumar, Panupong Pasupat, Sonal Gupta, and Christopher D. Manning. 2015. Bootstrapped self training for knowledge base population. In *TAC*. NIST.

Luisa Bentivogli, Pamela Forner, Claudio Giuliano, Alessandro Marchetti, Emanuele Pianta, and Kateryna Tymoshenko. 2010. Extending English ACE 2005 corpus annotation with ground-truth links to Wikipedia. In *Proceedings of the 2nd Workshop on The People's Web Meets NLP: Collaboratively Constructed Semantic Resources*, pages 19–27, Beijing, China. Coling 2010 Organizing Committee.

Arun Chaganty, Ashwin Paranjape, Percy Liang, and Christopher D. Manning. 2017. Importance sampling for unbiased on-demand evaluation of knowledge base population. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1038–1048, Copenhagen, Denmark. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Hady Elsahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon Hare, Frederique Laforest, and Elena Simperl. 2018. T-REx: A large scale alignment of natural language with knowledge base triples. In *Proceedings of the 11th Language Resources and Evaluation Conference*, Miyazaki, Japan. European Language Resource Association.

Jeremy Getman, Joe Ellis, Stephanie Strassel, Zhiyi Song, and Jennifer Tracey. 2018. Laying the Groundwork for Knowledge Base Population: Nine Years of Linguistic Resources for TAC KBP. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Xu Han, Hao Zhu, Pengfei Yu, Ziyun Wang, Yuan Yao, Zhiyuan Liu, and Maosong Sun. 2018. Fewrel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 4803–4809.

Heng Ji, Xiaoman Pan, Boliang Zhang, Joel Nothman, James Mayfield, Paul McNamee, and Cash Costello. 2017. Overview of TAC-KBP2017 13 languages entity discovery and linking. In *Proceedings of the 2017 Text Analysis Conference, TAC 2017, Gaithersburg, Maryland, USA, November 13-14, 2017*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 402–412, Baltimore, Maryland. Association for Computational Linguistics.

Yue Liu, Tongtao Zhang, Zhicheng Liang, Heng Ji, and Deborah L. McGuinness. 2018. Seq2rdf: An end-to-end application for deriving triples from natural language text. In *Proceedings of the ISWC 2018 Posters & Demonstrations, Industry and Blue Sky Ideas Tracks co-located with 17th International Semantic Web Conference (ISWC 2018), Monterey, USA, October 8th - to - 12th, 2018.*

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. pages 1105–1116.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew E. Peters, Sebastian Ruder, and Noah A. Smith. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks. In *Proceedings of the 4th Workshop on Representation Learning for NLP, RepL4NLP@ACL 2019, Florence, Italy, August 2, 2019.*, pages 7–14.

Pouya Pezeshkpour, Liyan Chen, and Sameer Singh. 2018. Embedding multimodal relational data for knowledge base completion. *CoRR*, abs/1809.01341.

Wei Shen, Jianyong Wang, and Jiawei Han. 2015. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Trans. Knowl. Data Eng.*, 27(2):443–460.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Trans. Knowl. Data Eng.*, 29(12):2724–2743.

Peng Xu and Denilson Barbosa. 2019. Connecting language and knowledge with heterogeneous representations for neural relation extraction. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, Minnesota, USA, June 2-7, 2019, Volume 2 (Short Papers)*, page 4.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1785–1794, Lisbon, Portugal. Association for Computational Linguistics.

Yuan Yao, Deming Ye, Peng Li, Xu Han, Yankai Lin, Zhenghao Liu, Zhiyuan Liu, Lixin Huang, Jie Zhou, and Maosong Sun. 2019. Docred: A large-scale document-level relation extraction dataset. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 764–777.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. 2017. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 35–45, Copenhagen, Denmark. Association for Computational Linguistics.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 207–212, Berlin, Germany. Association for Computational Linguistics.

## A Beyond Binary Relationships

While it would be convenient to express all facts as (*subject*; property; *object*) triples, this is not always possible. Many facts require further annotations to be sufficiently and accurately expressed in the KB. Take for instance (*United States*; head_of_government; *Barack Obama*), which only holds true in the past.

*Qualifiers* allow facts to be expanded or contextualized beyond what can be expressed with binary relationships. More specifically, qualifiers can be used to constrain the validity of a fact in time or space, e.g., (*employment fact*; end_time; *2017*); represent $n$-ary relationships, e.g., (*casting fact*; character_role; *Tony Stark*); and track provenance.

This release contains 4,518 facts annotated with three *temporal qualifiers*: IS_CURRENT, START_TIME and END_TIME. We use one of our baseline system to obtain facts to be annotated with qualifiers, along with the the sentence where each fact was found. Given a fact and a sentence, human annotators must decide the value of a qualifier (*true* or *false* for IS_CURRENT or a time expression for START_TIME, END_TIME). A third option *unclear* can be chosen in the case of uncertainty. To be included in the dataset, each fact must be annotated by two annotators in agreement. While preliminary experiments show promising results for qualifier extraction, they are out-of-scope of this work.

# Effective Use of Transformer Networks for Entity Tracking

**Aditya Gupta** and **Greg Durrett**
Department of Computer Science
The University of Texas at Austin
{agupta,gdurrett}@cs.utexas.edu

## Abstract

Tracking entities in procedural language requires understanding the transformations arising from actions on entities as well as those entities' interactions. While self-attention-based pre-trained language encoders like GPT and BERT have been successfully applied across a range of natural language understanding tasks, their ability to handle the nuances of procedural texts is still untested. In this paper, we explore the use of pre-trained transformer networks for entity tracking tasks in procedural text. First, we test standard lightweight approaches for prediction with pre-trained transformers, and find that these approaches underperform even simple baselines. We show that much stronger results can be attained by restructuring the input to guide the transformer model to focus on a particular entity. Second, we assess the degree to which transformer networks capture the process dynamics, investigating such factors as merged entities and oblique entity references. On two different tasks, ingredient detection in recipes and QA over scientific processes, we achieve state-of-the-art results, but our models still largely attend to shallow context clues and do not form complex representations of intermediate entity or process state.[1]

## 1 Introduction

Transformer based pre-trained language models (Devlin et al., 2019; Radford et al., 2018, 2019; Joshi et al., 2019; Yang et al., 2019) have been shown to perform remarkably well on a range of tasks, including entity-related tasks like coreference resolution (Kantor and Globerson, 2019) and named entity recognition (Devlin et al., 2019). This performance has been generally attributed to the robust transfer of lexical semantics to downstream tasks. However, these models are still better at capturing syntax than they are at more entity-focused aspects like coreference (Tenney et al., 2019a,b); moreover, existing state-of-the-art architectures for such tasks often perform well looking at only local entity mentions (Wiseman et al., 2016; Lee et al., 2017; Peters et al., 2017) rather than forming truly global entity representations (Rahman and Ng, 2009; Lee et al., 2018). Thus, performance on these tasks does not form sufficient evidence that these representations strongly capture entity semantics. Better understanding the models' capabilities requires testing them in domains involving complex entity interactions over longer texts. One such domain is that of procedural language, which is strongly focused on tracking the entities involved and their interactions (Mori et al., 2014; Dalvi et al., 2018; Bosselut et al., 2018).

This paper investigates the question of how transformer-based models form entity representations and what these representations capture. We expect that after fine-tuning on a target task, a transformer's output representations should somehow capture relevant entity properties, in the sense that these properties can be extracted by shallow classification either from entity tokens or from marker tokens. However, we observe that such "post-conditioning" approaches don't perform significantly better than rule-based baselines on the tasks we study. We address this by proposing entity-centric ways of structuring input to the transformer networks, using the entity to guide the intrinsic self-attention and form entity-centric representations for all the tokens. We find that our proposed methods lead to a significant improvement in performance over baselines.

Although our entity-specific application of transformers is more effective at the entity track-

[1] Code to reproduce experiments in this paper is available at https://github.com/aditya2211/transformer-entity-tracking



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 759–769, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

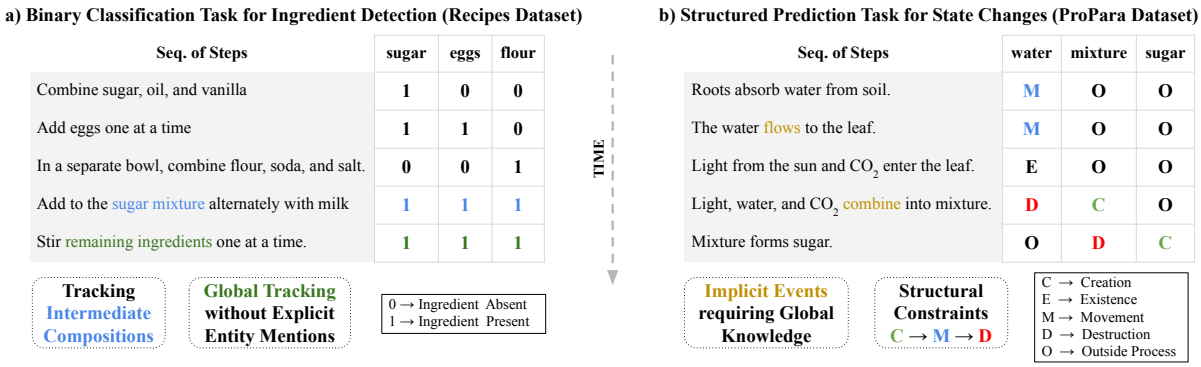


Figure 1: Process Examples from (a) RECIPES as a binary classification task of ingredient detection, and (b) PROPARA as a structured prediction task of identifying state change sequences. Both require cross-sentence reasoning, such as knowing what components are in a *mixture* and understanding verb semantics like *combine*.

ing tasks we study, we perform additional analysis and find that these tasks still do not encourage transformers to form truly deep entity representations. Our performance gain is largely from better understanding of verb semantics in terms of associating process actions with entity the paragraph is conditioned on. The model also does not specialize in "tracking" composed entities per se, again using surface clues like verbs to identify the components involved in a new composition.

We evaluate our models on two datasets specifically designed to invoke procedural understanding: (i) RECIPES (Kiddon et al., 2016), and (ii) PROPARA (Dalvi et al., 2018). For the RECIPES dataset, we classify whether an ingredient was affected in a certain step, which requires understanding when ingredients are combined or the focus of the recipe shifts away from them. The PROPARA dataset involves answering a more complex set of questions about physical state changes of components in scientific processes. To handle this more structured setting, our transformer produces potentials consumed by a conditional random field which predicts entity states over time. Using a unidirectional GPT-based architecture, we achieve state-of-the-art results on both the datasets; nevertheless, analysis shows that our approach still falls short of capturing the full space of entity interactions.

## 2 Background: Process Understanding

Procedural text is a domain of text involved with understanding some kind of process, such as a phenomenon arising in nature or a set of instructions to perform a task. Entity tracking is a core component of understanding such texts.

Dalvi et al. (2018) introduced the **PROPARA** dataset to probe understanding of scientific processes. The goal is to track the sequence of physical state changes (creation, destruction, and movement) entites undergo over long sequences of process steps. Past work involves both modeling entities across time (Das et al., 2019) and capturing structural constraints inherent in the processes (Tandon et al., 2018; Gupta and Durrett, 2019) Figure 1b shows an example of the dataset posed as a structured prediction task, as in (Gupta and Durrett, 2019). For such a domain, it is crucial to capture implicit event occurrences beyond explicit entity mentions. For example, in *fuel goes into the generator. The generator converts mechanical energy into electrical energy*", the *fuel* is implicitly destroyed in the process.

Bosselut et al. (2018) introduced the task of detecting state changes in recipes in the **RECIPES** dataset and proposed an entity-centric memory network neural architecture for simulating action dynamics. Figure 1a shows an example from the RECIPES dataset with a grid showing ingredient presence. We focus specifically on this core problem of ingredient detection; while only one of the sub-tasks associated with their dataset, it reflects some complex semantics involving understanding the current state of the recipe. Tracking of ingredients in the cooking domain is challenging owing to the compositional nature of recipes whereby ingredients mix together and are aliased as intermediate compositions.

We pose both of these procedural understanding tasks as classification problems, predicting the state of the entity at each timestep from a set of pre-defined classes. In Figure 1, these classes cor-

respond to either the presence (1) or absence (0) or the sequence of state changes create (C), move (M), destroy (D), exists (E), and none (O).

State-of-the-art approaches on these tasks are inherently entity-centric. Separately, it has been shown that entity-centric language modeling in a continuous framework can lead to better performance for LM related tasks (Clark et al., 2018; Ji et al., 2017). Moreover, external data has shown to be useful for modeling process understanding tasks in prior work (Tandon et al., 2018; Bosselut et al., 2018), suggesting that pre-trained models may be effective.

With such tasks in place, a strong model will ideally learn to form robust entity-centric representation at each time step instead of solely relying on extracting information from the local entity mentions. This expectation is primarily due to the evolving nature of the process domain where entities undergo complex interactions, form intermediate compositions, and are often accompanied by implicit state changes. We now investigate to what extent this is true in a standard application of transformer models to this problem.

## 3 Studying Basic Transformer Representations for Entity Tracking

### 3.1 Post-conditioning Models

The most natural way to use the pre-trained transformer architectures for the entity tracking tasks is to simply encode the text sequence and then attempt to "read off" entity states from the contextual transformer representation. We call this approach *post-conditioning*: the transformer runs with no knowledge of which entity or entities we are going to make predictions on, but we only condition on the target entity after the transformer stage.

Figure 3 depicts this model. Formally, for a labelled pair $(\{s_1, s_2, \ldots, s_t\}, y_{et})$, we encode the tokenized sequence of steps up to the current timestep (the sentences are separated by using a special `[SEP]` token), independent of the entity. We denote by $X = [h_1, h_2, \ldots, h_m]$ the contextualized hidden representation of the $m$ input tokens from the last layer, and by $g_e = \sum_{\text{ent toks}} emb(e_i)$ the entity representation for post conditioning. We now use one of the following two ways to make an entity-specific prediction:

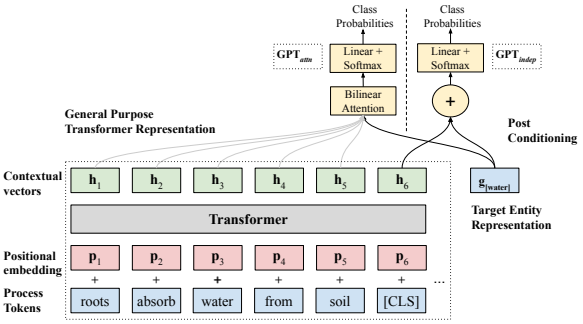


Figure 2: Post-conditioning entity tracking models. Bottom: the process paragraph is encoded in an entity-independent manner with transformer network and a separate entity representation $g_{[water]}$ for post-conditioning. Top: the two variants for the conditioning: (i) GPT$_{attn}$, and (ii) GPT$_{indep}$.

**Task Specific Input Token** We append a `[CLS]` token to the input sequence and use the output representation of the `[CLS]` token denoted by $h_{\texttt{[CLS]}}$ concatenated with the learned BPE embeddings of the entity as the representation $c_{e,t}$ for our entity tracking system. We then use a linear layer over it to get class probabilities:

$$c_{e,t} = [h_{\texttt{[CLS]}}; g_e]$$
$$P(y_t|s_t, s_{t-1}, \ldots, s_1, e) = \text{softmax}(c_{e,t}W_{task})$$

The aim of the `[CLS]` token is to encode information related to general entity related semantics participating in the recipe (*sentence priors*). We then use a single linear layer to learn sentence priors and entity priors independently, without strong interaction. We call this model GPT$_{indep}$.

**Entity Based Attention** Second, we explore a more fine-grained way of using the GPT model outputs. Specifically, we use bilinear attention between $g_e$ and the transformer output for the process tokens $X$ to get a contextual representation $c_{e,t}$ for a given entity. Finally, using a feed-forward network followed by softmax layer gives us the class probabilities:

$$a_i = g_e^T * W_{sim} * h_i$$
$$\alpha = \text{softmax}(a)$$
$$c_{e,t} = \sum \alpha_i * h_i$$

$$P(y_t|s_t, s_{t-1}, \ldots, s_1, e) = \text{softmax}(c_{e,t}W_{task})$$

The bilinear attention over the contextual representations of the process tokens allows the model to fetch token content relevant to that particular entity. We call this model GPT$_{attn}$.

| Variant | Template |
|---|---|
| Sentence, Entity First | [START] **Target Entity** [SEP] Steps 1 to $t-1$ [SEP] Step $t$ [CLS] |
| Sentence, Entity Last | [START] Steps 1 to $t-1$ [SEP] Step $t$ [SEP] **Target Entity** [CLS] |
| Document, Entity First | [START] **Target Entity** [SEP] Step 1 [CLS] Step 2 [CLS] ... Step $T$ [CLS] |
| Document, Entity Last | [START] Step 1 [SEP] **Target Entity** [CLS] ... Step $T$ [SEP] **Target Entity** [CLS] |

Table 1: Templates for different proposed entity-centric modes of structuring input to the transformer networks.

### 3.2 Results and Observations

We evaluate the discussed post-conditioning models on the ingredient detection task of the RECIPES dataset.[2] To benchmark the performance, we compare to three rule-based baselines. This includes (i) *Majority Class*, (ii) *Exact Match* of an ingredient $e$ in recipe step $s_t$, and (iii) *First Occurrence*, where we predict the ingredient to be present in all steps following the first exact match. These latter two baselines capture natural modes of reasoning about the dataset: an ingredient is used when it is directly mentioned, or it is used in every step after it is mentioned, reflecting the assumption that a recipe is about incrementally adding ingredients to an ever-growing mixture. We also construct a LSTM baseline to evaluate the performance of ELMo embeddings (ELMo$_{token}$ and ELMo$_{sent}$) (Peters et al., 2018) compared to GPT.

Table 2 compares the performance of the discussed models against the baselines, evaluating per-step entity prediction performance. Using the ground truth about ingredient's state, we also report the uncombined (UR) and combined (CR) recalls, which are per-timestep ingredient recall distinguished by whether the ingredient is explicitly mentioned (uncombined) or part of a mixture (combined). Note that *Exact Match* and *First Occ* baselines represent high-precision and high-recall regimes for this task, respectively.

As observed from the results, the post-conditioning frameworks underperform compared to the *First Occ* baseline. While the CR values appear to be high, which would suggest that the model is capturing the addition of ingredients to the mixture, we note that this value is also lower than the corresponding value for *First Occ*. This result suggests that the model may be approximating the behavior of this baseline, but doing so poorly. The unconditional self-attention mechanism of the transformers does not seem sufficient to capture the entity details at each time step beyond simple presence or absence. Moreover, we see that GPT$_{indep}$ performs somewhat comparably to GPT$_{attn}$, suggesting that consuming the transformer's output with simple attention is not able to really extract the right entity representation.

[2]We discuss training details more in Section 4.1, but largely use a standard GPT training protocol (Radford et al., 2018).

| Model | P | R | $F_1$ | Acc | UR | CR |
|---|---|---|---|---|---|---|
| Performance Benchmarks | | | | | | |
| Majority | - | - | - | 57.27 | - | - |
| Exact Match | **84.94** | 20.25 | 32.70 | 64.39 | 73.42 | 4.02 |
| First Occ | 65.23 | **87.17** | **74.60** | **74.65** | **84.88** | **87.79** |
| Models | | | | | | |
| GPT$_{attn}$ | 63.94 | 71.72 | 67.60 | 70.63 | 54.30 | 77.04 |
| GPT$_{indep}$ | 67.05 | 69.07 | 68.04 | 72.28 | 47.09 | 75.79 |
| ELMo$_{token}$ | 64.96 | 76.64 | 70.32 | 72.35 | 69.14 | 78.94 |
| ELMo$_{sent}$ | 69.09 | 72.88 | 70.90 | 74.48 | 57.05 | 77.71 |

Table 2: Performance of the rule-based baselines and the post conditioned models on the ingredient detection task of the RECIPES dataset. These models all underperform *First Occ*.

For PROPARA, we observe similar performance trends where the post-conditioning model performed below par with the state-of-the-art architectures.

## 4 Entity-Conditioned Models

The post-conditioning framework assumes that the transformer network can form strong representations containing entity information accessible in a shallow way based on the target entity. We now propose a model architecture which more strongly conditions on the entity as a part of the intrinsic self-attention mechanism of the transformers.

Our approach consists of structuring input to the transformer network to use and guide the self-attention of the transformers, conditioning it on the entity. Our main mode of encoding the input, the **entity-first** method, is shown in Figure 3. The input sequence begins with a [START] token, then the entity under consideration, then a [SEP] token. After each sentence, a [CLS] to-

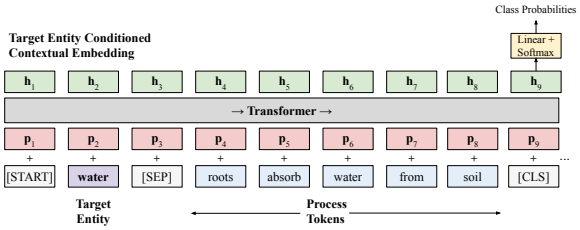


Figure 3: Entity conditioning model for guiding self-attention: the entity-first, sentence-level input variant fed into a left-to-right unidirectional transformer architecture. Task predictions are made at `[CLS]` tokens about the entity's state after the prior sentence.

ken is used to anchor the prediction for that sentence. In this model, the transformer can always observe the entity it should be primarily "attending to" from the standpoint of building representations. We also have an **entity-last** variant where the entity is primarily observed just before the classification token to condition the `[CLS]` token's self-attention accordingly. These variants are naturally more computationally-intensive than post-conditioned models, as we need to rerun the transformer for each distinct entity we want to make a prediction for.

**Sentence Level vs. Document Level** As an additional variation, we can either run the transformer once per document with multiple `[CLS]` tokens (a **document-level** model as shown in Figure 3) or specialize the prediction to a single timestep (a **sentence-level** model). In a sentence level model, we formulate each pair of entity $e$ and process step $t$ as a separate instance for our classification task. Thus, for a process with $T$ steps and $m$ entities we get $T \times m$ input sequences for fine tuning our classification task.

### 4.1 Training Details

In most experiments, we initialize the network with the weights of the standard pre-trained GPT model, then subsequently do either domain specific LM fine-tuning and supervised task specific fine-tuning.

**Domain Specific LM fine-tuning** For some procedural domains, we have access to additional unlabeled data. To adapt the LM to capture domain intricacies, we fine-tune the transformer network on this unlabeled corpus.

**Supervised Task Fine-Tuning** After the domain specific LM fine-tuning, we fine-tune our network parameters for the end task of entity tracking. For fine-tuning for the task, we have a labelled dataset which we denote by $\mathcal{C}$, the set of labelled pairs $(\{s_1, s_2, \ldots, s_t\}, y_{et})$ for a given process. The input is converted according to our chosen entity conditioning procedure, then fed through the pre-trained network.

In addition, we observed that adding the language model loss during task specific fine-tuning leads to better performance as well, possibly because it adapts the LM to our task-specific input formulation. Thus,

$$\mathcal{L}_{total} = \mathcal{L}_{task} + \lambda \mathcal{L}_{lm}$$

### 4.2 Experiments: Ingredient Detection

We first evaluate the proposed entity conditioned self-attention model on the RECIPES dataset to compare the performance with the post-conditioning variants.

#### 4.2.1 Systems to Compare

We use the pre-trained GPT architecture in the proposed entity conditioned framework with all its variants. BERT mainly differs in that it is bidirectional, though we also use the pre-trained `[CLS]` and `[SEP]` tokens instead of introducing new tokens in the input vocabulary and training them from scratch during fine-tuning. Owing to the lengths of the processes, all our experiments are performed on BERT$_{BASE}$.

**Neural Process Networks** The most significant prior work on this dataset is the work of Bosselut et al. (2018). However, their data condition differs significantly from ours: they train on a large noisy training set and do not use any of the high-quality labeled data, instead treating it as dev and test data. Consequently, their model achieves low performance, roughly 56 $F_1$ while ours achieves 82.5 $F_1$ (though these are not the exact same test set). Moreover, theirs underperforms the first occurrence baseline, which calls into question the value of that training data. Therefore, we do not compare to this model directly. We use the small set of human-annotated data for our probing task. Our train/dev/test split consists of 600/100/175 recipes, respectively.

#### 4.2.2 Results

Table 3 compares the overall performances of our proposed models. Our best ET$_{GPT}$ model achieves an $F_1$ score of 82.50. Comparing to

| Model | P | R | $F_1$ | Acc | UR | CR |
|---|---|---|---|---|---|---|
| Rule Based Benchmarks | | | | | | |
| Majority | - | - | - | 57.27 | - | - |
| Exact | 84.94 | 20.25 | 32.70 | 64.39 | 73.42 | 4.02 |
| First | 65.23 | 87.17 | 74.60 | 74.65 | 84.88 | 87.79 |
| Post Conditioning Models | | | | | | |
| GPT$_{attn}$ | 63.94 | 71.72 | 67.60 | 70.63 | 54.30 | 77.04 |
| GPT$_{concat}$ | 67.05 | 69.07 | 68.04 | 72.28 | 47.09 | 75.79 |
| ELMo$_{token}$ | 64.96 | 76.64 | 70.32 | 72.35 | 69.14 | 78.94 |
| ELMo$_{sent}$ | 69.09 | 72.88 | 70.90 | 74.48 | 57.05 | 77.71 |
| Entity-Centric Models | | | | | | |
| ET$_{BERT}$ | 72.49 | 80.09 | 76.10 | 78.50 | 84.30 | 78.82 |
| ET$_{GPT}$ ⓈⓁ | 75.27 | 83.85 | 79.33 | 81.32 | 87.28 | 82.81 |
| ET$_{GPT}$ ⓈⒻ | 76.70 | 83.98 | 80.17 | 82.26 | 88.20 | 82.69 |
| ET$_{GPT}$ ⒹⓁ | 79.19 | 83.82 | 81.44 | 83.67 | 88.11 | 82.51 |
| ET$_{GPT}$ ⒹⒻ | **79.85** | **84.19** | **81.96** | **84.16** | **87.91** | **83.05** |

Table 3: Performances of different baseline models discussed in Section 3, the ELMo baselines, and the proposed entity-centric approaches with the (D)ocument v (S)entence level variants formulated with both entity (F)irst v. (L)ater. Our ET$_{GPT}$ variants all substantially outperform the baselines.

the baselines (*Majority* through *First*) and post-conditioned models, we see that the early entity conditioning is critical to achieve high performance.

Although the *First* model still achieves the highest CR, due to operating in a high-recall regime, we see that the ET$_{GPT}$ models all significantly outperform the post-conditioning models on this metric, indicating better modeling of these compositions. Both recall and precision are substantially increaesd compared to these baseline models. Interestingly, the ELMo-based model underperforms the first-occurrence baseline, indicating that the LSTM model is not learning much in terms of recognizing complex entity semantics grounded in long term contexts.

Comparing the four variants of structuring input in proposed architectures as discussed in Section 4, we observe that the **document-level, entity-first model** is the best performing variant. Given the left-to-right unidirectional transformer architecture, this model notably forms target-specific representations for all process tokens, compared to using the transformer self-attention only to extract entity specific information at the end of the process.

#### 4.2.3 Ablations

We perform ablations to evaluate the model's dependency on the context and on the target ingredient. Table 4 shows the results for these ablations.

| Model | P | R | $F_1$ | Acc | UR | CR |
|---|---|---|---|---|---|---|
| ET$_{GPT}$ ⒹⒻ | | | | | | |
| w/o ing. | 67.47 | 60.46 | 63.77 | 70.64 | 35.82 | 67.97 |
| w/ ing. | 79.85 | 84.19 | 81.96 | 84.16 | 87.91 | 83.05 |
| ET$_{GPT}$ ⓈⒻ | | | | | | |
| w/o context | 67.88 | 75.91 | 71.67 | 74.36 | 87.00 | 72.52 |
| w/ context | 76.70 | 83.98 | 80.17 | 82.26 | 88.20 | 82.69 |

Table 4: Top: we compare how much the model degrades when it conditions on no ingredient at all (w/o ing.), instead making a generic prediction. Bottom: we compare how much using previous context beyond a single sentence impacts the model.

**Ingredient Specificity** In the "no ingredient" baseline (w/o ing.), the model is not provided with the specific ingredient information. Table 4 shows that while not being a strong baseline, the model achieves decent overall accuracy with the drop in UR being higher compared to CR. This indicates that there are some generic indicators (*mixture*) that it can pick up to try to guess at overall ingredient presence or absence.

**Context Importance** We compare with a "no context" model (w/o context) which ignore the previous context and only use the current recipe step in determining the ingredient's presence. Table 4 shows that the such model is able to perform surprisingly well, nearly as well as the first occurrence baseline.

This is because the model can often recognize words like verbs (for example, *add*) or nouns (for example, *mixture*) that indicate many ingredients are being used, and can do well without really tracking any specific entity as desired for the task.

### 4.3 State Change Detection (ProPara)

Next, we now focus on a structured task to evaluate the performance of the entity tracking architecture in capturing the structural information in the continuous self-attention framework. For this, we use the ProPara dataset and evaluate our proposed model on the comprehension task.

Figure 1b shows an example of a short instance from the ProPara dataset. The task of identifying state change follows a structure satisfying the existence cycle; for example, an entity can not be created after destruction. Our prior work (Gupta and Durrett, 2019) proposed a structured model for the task that achieved state-of-the-art perfor-

mance. We adapt our proposed entity tracking transformer models to this structured prediction framework, capturing creation, movement, existence (distinct from movement or creation), destruction, and non-existence.

We use the standard evaluation scheme of the PROPARA dataset, which is framed as answering the following categories of questions: (Cat-1) **Is** *e* created (destroyed, moved) in the process?, (Cat-2) **When** (step #) is *e* created (destroyed, moved)?, (Cat-3) **Where** is *e* created/destroyed/moved from/to)?

#### 4.3.1 Systems to Compare

We compare our proposed models to the previous work on the PROPARA dataset. This includes the entity specific MRC models, EntNet (Henaff et al., 2017), QRN (Seo et al., 2017), and KG-MRC (Das et al., 2019). Also, Dalvi et al. (2018) proposed two task specific models, ProLocal and ProGlobal, as baselines for the dataset. Finally, we compare against our past neural CRF entity tracking model (NCET) (Gupta and Durrett, 2019) which uses ELMo embeddings in a neural CRF architecture.

For the proposed GPT architecture, we use the task specific `[CLS]` token to generate tag potentials instead of class probabilities as we did previously. For BERT, we perform a similar modification as described in the previous task to utilize the pre-trained `[CLS]` token to generate tag potentials. Finally, we perform a Viterbi decoding at inference time to infer the most likely valid tag sequence.

#### 4.3.2 Results

Table 5 compares the performance of the proposed entity tracking models on the sentence level task. Since, we are considering the classification aspect of the task, we compare our model performance for Cat-1 and Cat-2. As shown, the structured document level, entity first $\text{ET}_{GPT}$ and $\text{ET}_{BERT}$ models achieve state-of-the-art results. We observe that the major source of performance gain is attributed to the improvement in identifying the exact step(s) for the state changes (Cat-2). This shows that the model are able to better track the entities by identifying the exact step of state change (Cat-2) accurately rather than just detecting the presence of such state changes (Cat-1). This task is more highly structured and in some ways more non-local than ingredient prediction; the high performance here shows that the $\text{ET}_{GPT}$ model is able to capture document level structural information effectively. Further, the structural constraints from the CRF also aid in making better predictions. For example, in the process *"higher pressure causes the sediment to heat up. the heat causes chemical processes. the material becomes a liquid. is known as oil."*, the *material* is a by-product of the chemical process but there's no direct mention of it. However, the material ceases to exist in the next step, and because the model is able to predict this correctly, maintaining consistency results in the model finally predicting the entire state change correctly as well.

| Model | Cat-1 | Cat-2 | Ma-Avg | Mi-Avg |
|---|---|---|---|---|
| Baselines | | | | |
| EntNet | 51.62 | 18.83 | 35.22 | 37. 03 |
| QRN | 52.37 | 15.51 | 33.94 | 35.97 |
| ProGlobal | 62.95 | 36.39 | 49.67 | 51.13 |
| Previous Work | | | | |
| KG-MRC | 62.86 | 40.00 | 51.43 | 52.69 |
| NCET | 70.55 | 44.57 | 57.56 | 58.99 |
| $\text{NCET}_{\text{ELMo}}$ | **73.68** | 47.09 | 60.38 | 61.85 |
| This Work | | | | |
| $\text{ET}_{GPT}$ ⒹⒻ | 73.52 | **52.21** | **62.87** | **64.03** |
| $\text{ET}_{BERT}$ | 73.55 | **52.59** | **63.07** | **64.22** |

Table 5: Performance of the proposed models on the PROPARA dataset. Our models outperform strong approaches from prior work across all metrics.

## 5 Challenging Task Phenomena

Based on the results in the previous section, our models clearly achieve strong performance compared to past approaches. We now revisit the challenging cases discussed in Section 2 to see if our entity tracking approaches are modeling sophisticated entity phenomena as advertised. For both datasets and associated tasks, we isolate the specific set of challenging cases grounded in tracking (i) intermediate compositions formed as part of combination of entities leading to no explicit mention, and (ii) implicit events which change entities' states without explicit mention of the affects.

### 5.1 Ingredient Detection

For RECIPES, we mainly want to investigate cases of ingredients getting re-engaged in the recipe not in a raw form but in a combined nature with other ingredients and henceforth no explicit mention. For example, *eggs* in step 4 of Figure 1a exem-

plifies this case. The performance in such cases is indicative of how strongly the model can track compositional entities. We also examine the performance for cases where the ingredient is referred by some other name.

**Intermediate Compositions** Formally, we pick the set of examples where the ground truth is a transition from $0 \rightarrow 1$ (not present to present) *and* the 1 is a "combined" case. Table 6 shows the model's performance on this subset of cases, of which there are 1049 in the test set. The model achieves an accuracy of 51.1% on these bigrams, which is relatively low given the overall model performance. In the error cases, the model defaults to the $1 \rightarrow 1$ pattern indicative of the *First Occ* baseline.

| | $0 \rightarrow 0$ | $0 \rightarrow 1$ | $1 \rightarrow 0$ | $1 \rightarrow 1$ |
|---|---|---|---|---|
| **#preds** | 179 | **526** | 43 | 301 |

Table 6: Model predictions from the document level entity first GPT model in 1049 cases of intermediate compositions. The model achieves only 51% accuracy in these cases.

**Hypernymy and Synonymy** We observe the model is able to capture ingredients based on their hypernyms (*nuts* $\rightarrow$ *pecans*, *salad* $\rightarrow$ *lettuce*) and rough synonymy (*bourbon* $\rightarrow$ *scotch*). This performance can be partially attributed to the language model pre-training. We can isolate these cases by filtering for *uncombined* ingredients when there is no matching ingredient token in the step. Out of 552 such cases in the test set, the model predicts 375 correctly giving a recall of 67.9. This is lower than overall UR; if pre-training behaves as advertised, we expect little degradation in this case, but instead we see performance significantly below the average on uncombined ingredients.

**Impact of external data** One question we can ask of the model's capabilities is to what extent they arise from domain knowledge in the large pre-trained data. We train transformer models from scratch and additionally investigate using the large corpus of unlabeled recipes for our LM pre-training. As can be seen in Table 7, the incorporation of external data leads to major improvements in the overall performance. This gain is largely due to the increase in combined recall. One possible reason could be that external data leads to better understanding of verb semantics and in turn the specific ingredients forming part of the intermediate compositions. Figure 4 shows that verbs are a critical clue the model relies on to make predictions. Performing LM fine-tuning on top of GPT also gives gains.

| **Model** | **P** | **R** | $F_1$ | **Acc** | **UR** | **CR** |
|---|---|---|---|---|---|---|
| No pre-training, 8 heads, 8 layers, 512 embedding size | | | | | | |
| No LM | 66.52 | 73.48 | 69.83 | 72.87 | 79.20 | 71.73 |
| 20k | 72.53 | 80.32 | 76.23 | 78.59 | 79.49 | 80.58 |
| 50k | 74.40 | 81.80 | 77.92 | 80.19 | 81.90 | 81.77 |
| Standard GPT pre-training | | | | | | |
| No LM | 79.85 | 84.19 | 81.96 | 84.16 | 87.91 | 83.05 |
| 20k | **80.14** | **85.01** | **82.50** | **84.59** | **88.83** | **83.84** |

Table 7: Performance for using unsupervised data for LM training.

## 5.2 State Change Detection

For PROPARA, Table 5 shows that the model does not significantly outperform the SOTA models in state change detection (Cat-1). However, for those correctly detected events, the transformer model outperforms the previous models for detecting the exact step of state change (Cat-2), primarily based on verb semantics. We do a finer-grained study in Table 8 by breaking down the performance for the three state changes: creation (C), movement (M), and destruction (D), separately. Across the three state changes, the model suffers a loss of performance in the movement cases. This is owing to the fact that the movement cases require a deeper compositional and implicit event tracking. Also, a majority of errors leading to false negatives are due to the the formation of new sub-entities which are then mentioned with other names. For example, when talking about *weak acid* in *"the water becomes a weak acid. the water dissolves limestone"* the *weak acid* is also considered to move to the *limestone*.

| **Model** | **Cat-1** | | | **Cat-2** | | |
|---|---|---|---|---|---|---|
| | C | M | D | C | M | D |
| $ET_{BERT}$ | 78.51 | 61.60 | 71.50 | 76.68 | 54.12 | 58.62 |
| $ET_{GPT}$ | 79.82 | 56.27 | 73.83 | 77.24 | 50.82 | 56.27 |

Table 8: Results for each state change type. Performance on predicting creation and destruction are highest, partially due to the model's ability to use verb semantics for these tasks.

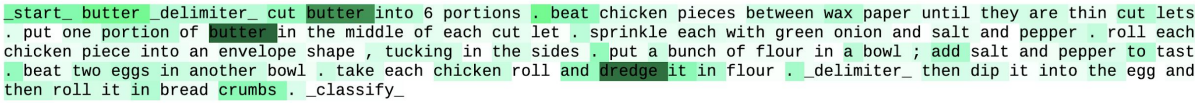


Figure 4: Gradient of the classification loss of the gold class with respect to inputs when predicting the status of *butter* in the last sentence. We follow a similar approach as Jain and Wallace (2019) to compute associations. Exact matches of the entity receive high weight, as does a seemingly unrelated verb *dredge*, which often indicates that the *butter* has already been used and is therefore present.

## 6 Analysis

The model's performance on these challenging task cases suggests that even though it outperforms baselines, it may not be capturing deep reasoning about entities. To understand what the model actually does, we perform analysis of the model's behavior with respect to the input to understand what cues it is picking up on.

**Gradient based Analysis** One way to analyze the model is to compute model gradients with respect to input features (Sundararajan et al., 2017; Jain and Wallace, 2019). Figure 4 shows that in this particular example, the most important model inputs are verbs possibly associated with the entity *butter*, in addition to the entity's mentions themselves. It further shows that the model learns to extract shallow clues of identifying actions exerted upon only the entity being tracked, regardless of other entities, by leveraging verb semantics.

In an ideal scenario, we would want the model to track constituent entities by translating the "focus" to track their newly formed compositions with other entities, often aliased by other names like *mixture, blend, paste* etc. However, the low performance on such cases shown in Section 5 gives further evidence that the model is not doing this.

**Input Ablations** We can study which inputs are important more directly by explicitly removing specific certain words from the input process paragraph and evaluating the performance of the resulting input under the current model setup. We mainly did experiments to examine the importance of: (i) verbs, and (ii) other ingredients.

Table 9 presents these ablation studies. We only observe a minor performance drop from 84.59 to 82.71 (accuracy) when other ingredients are removed entirely. Removing verbs dropped the performance to 79.08 and further omitting both leads to 77.79. This shows the models dependence on verb semantics over tracking the other ingredients.

| Input | Accuracy |
|---|---|
| Complete Process | 84.59 |
| w/o Other Ingredients | 82.71 |
| w/o Verbs | 79.08 |
| w/o Verbs & Other Ingredients | 77.79 |

Table 9: Model's performance degradation with input ablations. We see that the model's major source of performance is from verbs than compared to other ingredient's explicit mentions.

## 7 Conclusion

In this paper, we examined the capabilities of transformer networks for capturing entity state semantics. First, we show that the conventional framework of using the transformer networks is not rich enough to capture entity semantics in these cases. We then propose entity-centric ways to formulate richer transformer encoding of the process paragraph, guiding the self-attention in a target entity oriented way. This approach leads to significant performance improvements, but examining model performance more deeply, we conclude that these models still do not model the intermediate compositional entities and perform well by largely relying on surface entity mentions and verb semantics.

## Acknowledgments

This work was partially supported by NSF Grant IIS-1814522 and an equipment grant from NVIDIA. The authors acknowledge the Texas Advanced Computing Center (TACC) at The University of Texas at Austin for providing HPC resources used to conduct this research. Results presented in this paper were obtained using the Chameleon testbed supported by the National Science Foundation. Thanks as well to the anonymous reviewers for their helpful comments.

## References


Antoine Bosselut, Corin Ennis, Omer Levy, Ari Holtzman, Dieter Fox, and Yejin Choi. 2018. Simulating Action Dynamics with Neural Process Networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Elizabeth Clark, Yangfeng Ji, and Noah A. Smith. 2018. Neural Text Generation in Stories Using Entity Representations as Context. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics (ACL): Human Language Technologies*.

Bhavana Dalvi, Lifu Huang, Niket Tandon, Wen-tau Yih, and Peter Clark. 2018. Tracking State Changes in Procedural Text: a Challenge Dataset and Models for Process Paragraph Comprehension. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Rajarshi Das, Tsendsuren Munkhdalai, Xingdi Yuan, Adam Trischler, and Andrew McCallum. 2019. Building Dynamic Knowledge Graphs from Text using Machine Reading Comprehension. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.

Aditya Gupta and Greg Durrett. 2019. Tracking Discrete and Continuous Entity State for Process Understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL) Workshop on Structure Predictions for NLP*.

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. 2017. Tracking the World State with Recurrent Entity Networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Sarthak Jain and Byron C. Wallace. 2019. Attention is not Explanation. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Yangfeng Ji, Chenhao Tan, Sebastian Martschat, Yejin Choi, and Noah A. Smith. 2017. Dynamic Entity Representations in Neural Language Models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2019. SpanBERT: Improving Pre-training by Representing and Predicting Spans.

Ben Kantor and Amir Globerson. 2019. Coreference Resolution with Entity Equalization. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally Coherent Text Generation with Neural Checklist Models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end Neural Coreference Resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Kenton Lee, Luheng He, and Luke Zettlemoyer. 2018. Higher-Order Coreference Resolution with Coarse-to-Fine Inference. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Shinsuke Mori, Hirokuni Maeta, Yoko Yamakata, and Tetsuro Sasada. 2014. Flow Graph Corpus from Recipe Texts. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC)*.

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL* https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/languageunsupervised/languageunderstandingpaper.pdf.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *URL* https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf.

Altaf Rahman and Vincent Ng. 2009. Supervised Models for Coreference Resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Minjoon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Query-Reduction Networks for Question Answering. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic Attribution for Deep Networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 3319–3328.

Niket Tandon, Bhavana Dalvi, Joel Grus, Wen-tau Yih, Antoine Bosselut, and Peter Clark. 2018. Reasoning about Actions and State Changes by Injecting Commonsense Knowledge. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019a. BERT Rediscovers the Classical NLP Pipeline. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Sam Bowman, Dipanjan Das, and Ellie Pavlick. 2019b. What do you learn from context? Probing for sentence structure in contextualized word representations. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Sam Wiseman, Alexander M. Rush, and Stuart M. Shieber. 2016. Learning Global Features for Coreference Resolution. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding.

# Explicit Cross-lingual Pre-training for Unsupervised Machine Translation

**Shuo Ren**[†‡]*, **Yu Wu** [§], **Shujie Liu**[§], **Ming Zhou**[§], **Shuai Ma**[†‡]
[†]SKLSDE Lab, Beihang University, Beijing, China
[‡]Beijing Advanced Innovation Center for Big Data and Brain Computing, China
[§]Microsoft Research Asia, Beijing, China
[†]{shuoren,mashuai}@buaa.edu.cn [§]{Wu.Yu,shujliu,mingzhou}@microsoft.com

## Abstract

Pre-training has proven to be effective in unsupervised machine translation due to its ability to model deep context information in cross-lingual scenarios. However, the cross-lingual information obtained from shared BPE spaces is inexplicit and limited. In this paper, we propose a novel cross-lingual pre-training method for unsupervised machine translation by incorporating explicit cross-lingual training signals. Specifically, we first calculate cross-lingual n-gram embeddings and infer an n-gram translation table from them. With those n-gram translation pairs, we propose a new pre-training model called Cross-lingual Masked Language Model (CMLM), which randomly chooses source n-grams in the input text stream and predicts their translation candidates at each time step. Experiments show that our method can incorporate beneficial cross-lingual information into pre-trained models. Taking pre-trained CMLM models as the encoder and decoder, we significantly improve the performance of unsupervised machine translation. Our code is available at https://github.com/Imagist-Shuo/CMLM.

## 1 Introduction

Unsupervised machine translation has become an emerging research interest in recent years (Artetxe et al., 2017; Lample et al., 2017, 2018; Artetxe et al., 2018b; Marie and Fujita, 2018; Ren et al., 2019; Lample and Conneau, 2019). The common framework of unsupervised machine translation builds two initial translation models at first (i.e., source to target and target to source), and then does iterative back-translation (Sennrich et al., 2016a; Zhang et al., 2018) with the two models using pseudo data generated by each other. The initialization process is crucial to the final translation performance as pointed in Lample et al. (2018), Artetxe et al. (2018b) and Ren et al. (2019).

Previous approaches benefit mostly from cross-lingual n-gram embeddings, but recent work proves that cross-lingual language model pre-training could be a more effective way to build initial unsupervised machine translation models (Lample and Conneau, 2019). However, in their method, the cross-lingual information is mostly obtained from shared Byte Piece Encoding (BPE) (Sennrich et al., 2016b) spaces during pre-training, which is inexplicit and limited. Firstly, although the same BPE pieces from different languages may share the same semantic space, the semantic information of n-grams or sentences in different languages may not be shared properly. However, cross-lingual information based on n-gram level is crucial to model the initialization of unsupervised machine translation (Lample et al., 2018; Artetxe et al., 2018b), which is absent in the current pre-training method. Secondly, BPE sharing is limited to languages that share much of their alphabet. For language pairs that are not the case, the above pre-training method may not provide much useful cross-lingual information.

In this paper, by incorporating explicit cross-lingual training signals, we propose a novel cross-lingual pre-training method based on BERT (Devlin et al., 2018) for unsupervised machine translation. Our method starts from unsupervised cross-lingual n-gram embeddings, from which we infer n-gram translation pairs. Then, we propose a new pre-training objective called Cross-lingual Masked Language Model (CMLM), which masks the input n-grams randomly and predicts their corresponding n-gram translation candidates inferred above. To solve the mismatch between different lengths of the masked source and predicted target n-grams, IBM models are introduced (Brown et al., 1993) to derive the training loss at each

*Contribution during internship at MSRA.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 770–779, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

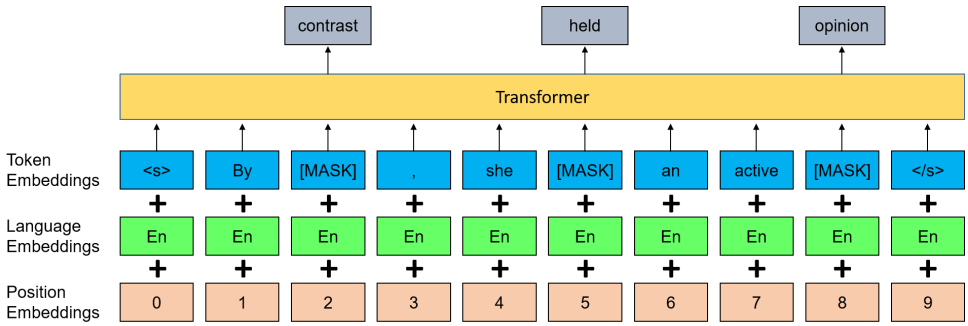


Figure 1: Masked Language Model (MLM) for BERT training. For a given sentence, this task is to predict randomly masked tokens, i.e., "contrast", "held" and "opinion". In practice, it is implemented based on BPE.

time step. In this way, we can guide the model with more explicit and strong cross-lingual training signals, and meanwhile, leverage the potential of BERT to model context information. We then use two pre-trained cross-lingual language models as the encoder and decoder respectively to build desired machine translation models. Our method can be iteratively performed with the n-gram translation table updated by downstream tasks. Experiments show that our method can produce better cross-lingual representations and significantly improve the performance of unsupervised machine translation. Our contributions are listed as follows.

- We propose a novel cross-lingual pre-training method to incorporate explicit cross-lingual information into pre-trained models, which significantly improves the performance of unsupervised machine translation.
- We introduce IBM models to calculate the step-wise training loss for CMLM, which breaks the limitation that masked n-grams and predicted ones have to be the same length during BERT training.
- We produce strong context-aware cross-lingual representations with our pre-training method, which helps in word alignment and cross-lingual classification tasks.

## 2 Background

### 2.1 BERT

BERT (Devlin et al., 2018), short for Bidirectional Encoder Representations from Transformers, is a powerful pre-training method for natural language processing and breaks records of many NLP tasks after corresponding fine-tuning. The core idea of BERT is pre-training a deep bidirectional Transfomer encoder (Vaswani et al., 2017) with two training tasks. The first one is Masked Language Model (MLM) referring to the *Cloze* task (Taylor, 1953), which takes a straightforward approach of masking some percentage of the input tokens at random, and then predicting them with the corresponding Transformer hidden states, as shown in Figure 1. The second one is Next Sentence Prediction, which means to predict whether two sentences are adjacent or not. This task is designed for some tasks that need modeling the relationship between two sentences such as Question Answering (QA) and Natural Language Inference (NLI).

### 2.2 XLM

Based on BERT, Lample and Conneau (2019) propose a cross-lingual version called XLM and reach the state-of-the-art performance on some cross-lingual NLP tasks including cross-lingual classification (Conneau et al., 2018), machine translation, and unsupervised cross-lingual word embedding. The basic points of XLM are mainly two folds. The first one is to use a shared vocabulary of BPE (Sennrich et al., 2016b) to provide potential cross-lingual information between two languages just as mentioned in Lample et al. (2018), in an inexplicit way though. The second point is a newly proposed training task called Translation Language Modeling (TLM), which extends MLM by concatenating parallel sentences into a single training text stream. In this way, the model can leverage the cross-lingual information provided by parallel sentences to predict the masked tokens. However, for unsupervised machine translation, TLM cannot be used due to the lack of parallel sentences. Different from them, we are motivated to give the model more explicit and strong cross-lingual information and propose a new pre-training method by (1) masking source n-grams and (2) predicting their corresponding translation candidates.

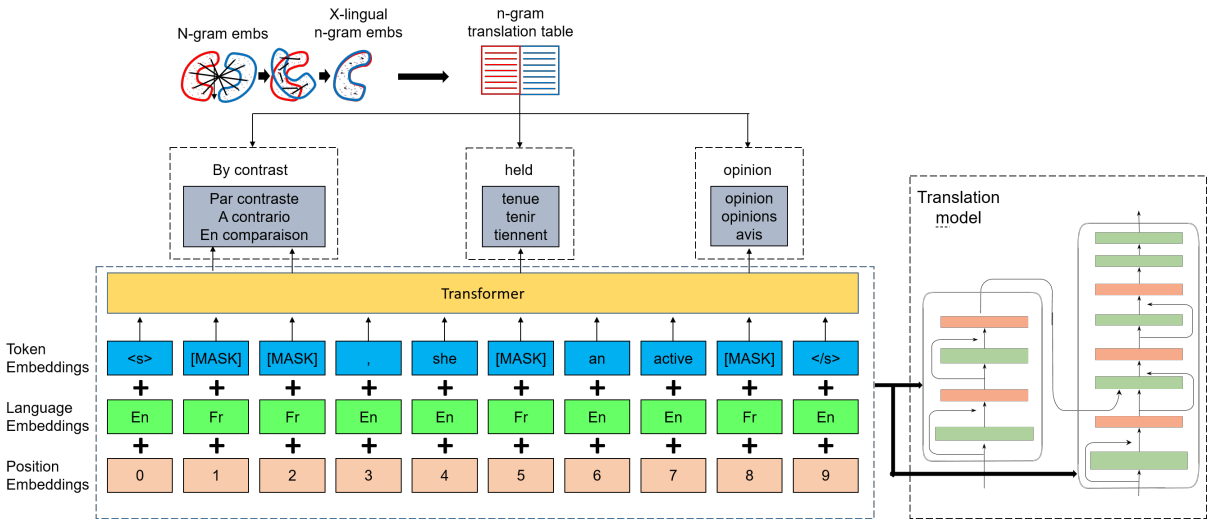


Figure 2: Method overview. Our method consists of three steps. The first one is the n-gram translation table inferring. The second one is pre-training with our proposed objective Cross-lingual Masked Language Model (CMLM) which is to predict the translation candidates of randomly masked n-grams. The last step is to leverage the pre-trained cross-lingual language models as the encoder and decoder of a neural machine translation model and fine-tune the translation model iteratively.

## 3 Method

### 3.1 Overview

Our method can be divided into three steps as shown in Figure 2. Given two languages $X$ and $Y$, we first get unsupervised cross-lingual n-gram embeddings of them, from which we infer n-gram translation tables (source-to-target and target-to-source). The n-gram translation pairs inferred in this way have proven to be instructive for initial unsupervised machine translation models (Artetxe et al., 2018b; Lample et al., 2018; Marie and Fujita, 2018; Ren et al., 2019). Then, we pre-train cross-lingual BERT language models with our proposed Cross-lingual Masked Language Model (CMLM) objective. Specifically, we randomly choose n-grams in the monolingual sentences and predict corresponding translation candidates in the n-gram translation table inferred in the first step. In this way, we can guide the model with explicit and strong cross-lingual training signals. Finally, two pre-trained cross-lingual language models are used to initialize the encoder and decoder respectively, based on which, denoising auto-encoder and iterative back-translation are leveraged to fine-tune the unsupervised machine translation models.

The above process is repeated by updating the n-gram table with the n-gram translation pairs extracted from the pseudo data generated by the translation models. In the following subsections, we will give details of each step.

### 3.2 N-gram Translation Table Inferring

Following previous work of unsupervised machine translation (Artetxe et al., 2018b; Lample et al., 2018; Ren et al., 2019), given two languages $X$ and $Y$, we build our n-gram translation tables as follows. First, we obtain monolingual n-gram embeddings using fastText (Bojanowski et al., 2017) and then get cross-lingual n-gram embeddings using vecmap (Artetxe et al., 2018a) in a fully unsupervised way. Based on that, we calculate the similarity score of n-grams $x$ and $y$ in two languages respectively with the marginal-based scoring method (Conneau et al., 2017; Artetxe and Schwenk, 2018). Specifically, given the cross-lingual embeddings of $x$ and $y$, denoted as $e_x$ and $e_y$, the similarity score is calculated as:

$$\text{sim}(x, y) = \text{margin}(\cos(e_x, e_y), \sum_{z \in \text{NN}_n(x)} \frac{\cos(e_x, e_z)}{2n} + \sum_{z \in \text{NN}_n(y)} \frac{\cos(e_y, e_z)}{2n}) \quad (1)$$

where $\text{margin}(a, b)$ is a marginal scoring function and $\text{NN}_n(x)$ denotes $x$'s k-nearest neighbors in the other language. In our experiments, $n$ is 5 and $\text{margin}(a, b) = \frac{a}{b}$.

We take the above similarity scores as the translation probabilities between $x$ and $y$ in the n-gram table. For each top frequent n-gram in the source language, we retrieve top-$k$ n-gram translation candidates in the target language.

### 3.3 Cross-lingual Masked Language Model

In this section, we introduce our proposed method for pre-training cross-lingual language models based on BERT. Unlike the masked language model (MLM) described in Section 2.2 which masks several tokens in the input stream and predict those tokens themselves, we randomly select some percentage of n-grams in the input source sentence, and predict their translation candidates retrieved from Section 3.2. We call our proposed pre-training objective "Cross-lingual Masked Language Model" (CMLM) as shown in Figure 2. The difficulty for BERT to predict target phrases during training is that the lengths of the translation candidates are sometimes different from the source n-grams. To deal with this problem, we turn to IBM Model 2 (Brown et al., 1993) to calculate the training loss at each time step. Our proposed method breaks the limitation that masked n-grams and predicted ones have to be the same length during BERT training.

Specifically, according to IBM Model 2, given a source n-gram $x_1^l$ and a target one $y_1^m$, where $l$ and $m$ are the numbers of tokens in the source and target n-grams respectively, the translation probability from $x_1^l$ to $y_1^m$ is calculated as:

$$\mathbf{Pr}(y_1^m|x_1^l) = \epsilon \prod_{j=1}^{m} \sum_{i=0}^{l} a(i|j,l,m) p(y_j|x_i) \quad (2)$$

where $\epsilon = p(m|x_1^l)$, i.e. the probability that the translation of $x_1^l$ consists of $m$ tokens; $a(j|i,l,m)$ is the probability that the $i^{th}$ source token is aligned with the $j^{th}$ target token conditioned on the lengths $l$ and $m$, while $p(y_j|x_i)$ is the translation probability from the source token $x_i$ to the target token $y_i$. Based on the IBM Model 2, the loss function of our CMLM is defined as

$$\begin{aligned} \mathcal{L}_{cmlm} &= -\log \mathbf{Pr}(y_1^m|x_1^l) \\ &= -\log \epsilon - \sum_{j=1}^{m} \log \sum_{i=0}^{l} a(i|j,l,m) p(y_j|x_i) \end{aligned} \quad (3)$$

The derived gradient w.r.t model parameters $\theta$ at each time step can be calculated as follows:

$$\begin{aligned} &\nabla_\theta \mathcal{L}_{cmlm} \\ &= -\sum_{j=1}^{m} \frac{a(i|j,l,m) p(y_j|x_i)}{\sum_{i=0}^{l} a(i|j,l,m) p(y_j|x_i)} \nabla_\theta \log p(y_j|x_i) \end{aligned} \quad (4)$$

Since the target n-gram $y_1^m$ is predicted with our modified BERT, in practice, the source word $x_i$ in Eq.(4) is replaced with its context-sensitive embedding $\mathrm{C}(x_1^l)$, which is the corresponding hidden state of the top Transformer layer. The alignment probability $a(i|j,l,m)$ cannot be learned during training because of the absence of bilingual corpus. Therefore, cross-lingual BPE embeddings are leveraged to calculate the normalized $\mathrm{sim}(x_i, y_j)$ to approximate $a(i|j,m,l)$. $p(y_j|x_i)$ is the model prediction in Softmax outputs. For each source n-gram, all of the retrieved $k$ translation candidates are used to calculate the cross entropy loss, which are weighted with their translation probabilities in the n-gram table.

Given a language pair $X - Y$, we process both languages with the same shared BPE vocabulary using their monolingual sentences together during pre-training. Following Devlin et al. (2018); Lample and Conneau (2019), in our CMLM objective, we randomly sample 15% of the BPE n-grams from the text streams, and replace them by [MASK] tokens 70% of the time. During pre-training, in each iteration, a batch is composed of sentences sampled from the same language, and we alternate between MLM and CMLM objectives. Different from the original MLM in BERT, in the half of the MLM time, we randomly choose some source n-grams in the input text stream, and replace them with their translation candidates to construct code-switching sentences. Our final pre-training loss is defined as

$$\mathcal{L}_{pre} = \mathcal{L}_{cmlm} + \mathcal{L}_{mlm} \quad (5)$$

### 3.4 Unsupervised Machine Translation

Taking two cross-lingual language models pre-trained with the above method as the encoder and decoder, we build an initial unsupervised neural machine translation model. Then, we train the model with monolingual data until convergence via denoising auto-encoder and iterative back-translation, as described in Artetxe et al. (2017); Lample et al. (2017, 2018). Different from them, we step further and make another iteration with updated n-gram translation tables. Specifically, we translate the monolingual sentences with our latest translation model and run GIZA++ (Och and Ney, 2003) on the generated pseudo parallel data to extract updated n-gram translation pairs, which are used to tune the encoder as Section 3.3, together with the back-translation within a multi-task learn-

ing framework. Experimental results show that running another iteration can further improve the translation performance.

It is also interesting to explore the usage of pre-trained decoders in the translation model. It seems that pre-training decoders has a smaller effect on the final performance than pre-training encoders (Lample and Conneau, 2019), one reason for which could be that the encoder-to-decoder attention is not pre-trained. Therefore, the parameters of the decoder need to be re-adjusted substantially in the following tuning process for MT task. In our experiments, we explore some other usage of pre-trained decoders, i.e., we use the pre-trained decoder as the feature extractor and feed the outputs into a new decoder consisting of several Transformer layers with the attention to the encoder. We find this method improves the performance of some language translation directions.

## 4 Experiments

In this section, we conduct experiments to evaluate our proposed pre-training method. In Section 4.1, we will introduce the setup of our experiments, followed by the overall results of the final unsupervised MT models. Then, in Section 4.3, we will discuss another usage of pre-trained decoders for translation models. To evaluate the cross-lingual modeling capacity of our pre-trained encoders, in Section 4.4, we conduct experiments on word alignment and cross-lingual classification tasks. Finally, we do the ablation study to check the performance contribution of each component in our proposed method.

### 4.1 Setup

#### Data and Preprocess

In our experiments, we consider three language pairs, English-French (en-fr), English-German (en-de) and English-Romanian (en-ro). For each language, we use all the available sentences in NewsCrawl till 2018, monolingual datasets from WMT. The NewsCrawl data are used in both pre-training and the following unsupervised NMT iteration process. Our CMLM is optimized based on the pre-trained models released by Lample and Conneau (2019)[1], which are trained with Wikipedia dumps. For fair comparison, we use *newstest* 2014 as the test set for en-fr, and *newstest* 2016 for en-de and en-ro.

We use Moses scripts for tokenization, and use fastBPE[2] to split words into subword units with their released BPE codes[1]. The number of shared BPE codes for each language pair is 60,000.

#### Implementation Details

Our implementation is based on the released code of XLM[1] (Paszke et al., 2017). Specifically, we use a Transformer architecture with 1024 hidden units, 8 heads, GELU activations (Hendrycks and Gimpel, 2016), with a dropout rate of 0.1. The models are trained with the Adam optimizer (Kingma and Ba, 2014), a linear warmup (Vaswani et al., 2017) and the learning rates varying from $10^4$ to $5 \times 10^4$.

For both of the MLM and CMLM objectives, we use streams of 256 tokens and mini-batches of size 64. We use the averaged perplexity over languages as a stopping criterion for training. For machine translation, we use 6 Transformer layers, and we create mini-batches of 2000 tokens.

#### Baselines

Our method is compared with six baselines of unsupervised MT systems listed in the upper part of Table 1. The first two baselines (Artetxe et al., 2017; Lample et al., 2017) use a shared encoder and different decoders for two languages with the training methods of denoising auto-encoder and iterative back-translation. The third baseline (Artetxe et al., 2018b) is an unsupervised PBSMT model, which uses the initial PBSMT models built with language models and n-gram translation tables inferred from cross-lingual embeddings, followed with the iterative back-translation. The fourth baseline (Lample et al., 2018) is a hybrid method of unsupervised NMT and PBSMT by combining the pseudo data generated by PBSMT models into the final iteration of NMT. The fifth baseline (Ren et al., 2019) is also a hybrid method of NMT and PBSMT but different from Lample et al. (2018), they leverage PBSMT as posterior regularization in each NMT iteration. The last baseline is XLM described in Section 2.2.

### 4.2 Overall Results

The overall comparison results of unsupervised machine translation are shown in Table 1. From the table, we find that our proposed method significantly outperforms previous methods on all language pairs by the average BLEU score of 1.7,

[1]https://github.com/facebookresearch/XLM

[2]https://github.com/glample/fastBPE

| | Method | fr2en | en2fr | de2en | en2de | ro2en | en2ro |
|---|---|---|---|---|---|---|---|
| Baselines | (Artetxe et al., 2017) | 15.6 | 15.1 | - | - | - | - |
| | (Lample et al., 2017) | 14.3 | 15.1 | 13.3 | 9.6 | - | - |
| | (Artetxe et al., 2018b) | 25.9 | 26.2 | 23.1 | 18.2 | - | - |
| | (Lample et al., 2018) | 27.7 | 28.1 | 25.2 | 20.2 | 23.9 | 25.1 |
| | (Ren et al., 2019) | 28.9 | 29.5 | 26.3 | 21.7 | - | - |
| | (Lample and Conneau, 2019) | 33.3 | 33.4 | 34.3 | 26.4 | 31.8 | 33.3 |
| Ours | Iter 1 | 34.8 | 34.9 | 35.5 | **27.9** | 33.6 | 34.7 |
| | Iter 2 | **34.9** | **35.4** | **35.6** | 27.7 | **34.1** | **34.9** |

Table 1: Comparison of the final unsupervised MT performance (BLEU). In this table, "Iter 2" means we do the whole process with another iteration as described in Section 3.4.

and both the improvements of en2fr and ro2en are over 2 BLEU points. The results indicate that the explicit cross-lingual information incorporated by our proposed CMLM is beneficial to the unsupervised machine translation task. Notice that by doing another iteration ("Iter 2") with updated n-gram tables as described in Section 3.4, we further improve the performance a bit for most translation directions with the improvements of en2fr and ro2en bigger than 0.5 BLEU point, which confirms the potential that fine-tuned machine translation models contain more beneficial cross-lingual information than the initial n-gram translation tables, which can be used to enhance the pre-trained model iteratively.

The improvement made by Lample and Conneau (2019) compared with the first five baselines shows that cross-lingual pre-training can be necessary for unsupervised MT. However, the cross-lingual information learned with this method during pre-training is mostly from the shared subword space, which is inexplicit and not strong enough. Our proposed method can give the model more explicit and strong cross-lingual training signals so that the pre-trained model contains much beneficial cross-lingual information for unsupervised machine translation. As a result, we can further improve the translation performance significantly, compared with Lample and Conneau (2019) (with the significance level of $p<0.01$).

### 4.3 Usage of Pre-trained Decoder

As mentioned in Section 3.4, it is interesting to explore the different usage of pre-trained decoders in the MT task. According to our intuition, directly using the pre-trained model as the decoder may not work well because parameters of the decoder need substantial adjustment due to the attention part between the encoder and the decoder. Therefore, we treat the pre-trained decoder as the feature extractor and add several Transformer layers with the encoder-to-decoder attention on top of it. We also try to fix the pre-trained decoder and just fine-tune the encoder and the added decoder part. The experiments are conducted based on "Iter 1" with the results reported in Table 2.

From this table, we can see that directly using the pre-trained model as the decoder may be the best choice for most of the time, with the exceptions of en2fr and ro2en. By adding 6 Transformer layers on top of the original pre-trained decoder can achieve higher performance for en2fr and ro2en, but not significant. The reason could be that it is difficult to train the additional Transformer layers from scratch in the unsupervised scenario. There is also an interesting phenomenon that if we fix the pre-trained part of the decoder and only tune the added Transformer layers, the final performance will drop drastically, which indicates that the representation space of the decoder requires substantial adaptation, even though the pre-trained models already contain cross-lingual information. We think that further deep research on the decoder initialization could be a necessary and interesting topic in the future.

### 4.4 Evaluation of Cross-lingual Pre-trained Encoder

#### Word Alignment

To evaluate the cross-lingual modeling capacity of our pre-trained models, we first conduct experiments on the English-French (en-fr) dataset of the HLT/NAACL 2003 alignment shared task (Mihalcea and Pedersen, 2003). Given two parallel sentences in English and French respectively, we feed each sentence into the pre-trained cross-lingual encoder and get its respective outputs. Then, we calculate the similarities between the outputs of the two sentences and choose target words with max similarity scores as the alignments of corre-

| Decoder type | fr2en | en2fr | de2en | en2de | ro2en | en2ro |
|---|---|---|---|---|---|---|
| Pre-trained | **34.8** | 34.5 | **35.5** | **27.9** | 33.4 | **34.7** |
| Pre-trained + 4 TF layers | 34.2 | 33.9 | 34.9 | 26.8 | 32.5 | 33.2 |
| Pre-trained + 6 TF layers | 34.6 | **34.9** | 34.9 | 27.5 | **33.6** | 34.1 |
| Pre-trained (fix) + 4 TF layers | 26.8 | 22.0 | 23.9 | 19.2 | 24.7 | 25.9 |
| Pre-trained (fix) + 6 TF layers | 28.2 | 22.4 | 24.2 | 19.7 | 25.1 | 26.2 |

Table 2: Test BLEU scores with different usage of the pre-trained decoder.

sponding source words.

We compare the context-unaware method (i.e., directly calculating the similarity scores between unsupervised cross-lingual embeddings (Artetxe et al., 2018a) of source and target words), XLM (Lample and Conneau, 2019) and our proposed CMLM pre-training method in the Table 3. In this experiment, we leave out all the OOV words and those torn apart by the BPE operations.

| Method | P | R | F | AER |
|---|---|---|---|---|
| Context-unaware | 0.3860 | 0.1854 | 0.2505 | 0.6061 |
| XLM | 0.5480 | 0.3178 | 0.4023 | 0.4302 |
| Ours | **0.5898** | **0.3497** | **0.4391** | **0.4016** |

Table 3: Results of word alignment tasks using different cross-lingual word embeddings. In this table, "P" means "precision", "R" means recall", "F" means "F-measure" and "AER" means the "alignment error rate".

From this table, we find that, based on BERT, both XLM and our method can model cross-lingual context information, indicating that context information can greatly enhance the cross-lingual mapping between the source and target words. By leveraging the explicit cross-lingual information in the model training, our CMLM can outperform XLM significantly. This confirms that our CMLM does better to connect the source and target representation space, with which as pre-trained models, the performance of unsupervised NMT can be improved.

**Cross-lingual Classification**

We also conduct experiments on the cross-lingual classification task (Conneau et al., 2018) using the cross-lingual language inference (XNLI) dataset (Conneau et al., 2018). Specifically, we add a linear classification layer on top of the first hidden state of our pre-trained model and fine-tune its parameters on the English NLI dataset. Without using any labeled data for French (fr) and Germany (de) languages, we only report the zero-shot classification results for them as shown in Table 4. We can find that our method reaches a new record of the zero-shot cross-lingual classification task on languages of French (fr) and Germany (de), which confirms again that our CMLM works better on modeling cross-lingual information than previous methods in the unsupervised scenario.

| Method | en | fr | de |
|---|---|---|---|
| (Conneau et al., 2018) | 73.7 | 67.7 | 67.7 |
| (Devlin et al., 2018) | 81.4 | - | 70.5 |
| (Lample and Conneau, 2019) | 83.2 | 76.5 | 74.2 |
| Ours | **83.4** | **77.1** | **74.7** |

Table 4: Results of zero-shot cross-lingual classification (on XNLI test sets).

### 4.5 Ablation Study

In this section, we will discuss the different settings of our method. Firstly, the training loss of our pre-trained method contains two parts, i.e., CMLM and MLM, just as Eq.(5) shows. To study the respective influences of these two parts, we remove the MLM loss from it and compare the performance on en-fr and en-de translation tasks. Since our CMLM task differs from XLM in two aspects during pre-training. The first one is that we randomly choose n-grams to mask in the input text stream rather than BPE tokens, and the second one is that we predict the translation candidates of a source n-gram rather than predicting the source n-gram itself. Although the first one has proven to be beneficial during pre-training to some NLP tasks, we want to check how much its influence is to our final translation performance. Therefore, we disable those two modifications in CMLM one by one and report the translation results. Our experiments are conducted based on "Iter 1" with the results in Table 5.

From Table 5, we can find that the combination of CMLM and MLM can improve the translation performance by about 0.6 to 0.7 BLEU compared with any one only. This confirms the monolingual context modeling capacity of the MLM, which is quite useful for unsupervised machine translation. By combining CMLM and MLM, we can enforce

| | fr2en | en2fr | de2en | en2de |
|---|---|---|---|---|
| CMLM + MLM | **34.8** | **34.9** | **35.5** | **27.9** |
| CMLM | 34.1 | 34.3 | 35.1 | 27.2 |
| - translation prediction | 33.7 | 33.9 | 34.8 | 26.6 |
| - - n-gram mask | 33.3 | 33.4 | 34.3 | 26.4 |

Table 5: Ablation study. "CMLM + MLM" means we use $\mathcal{L}_{pre}$ as the pre-training loss; "CMLM" means we only use $\mathcal{L}_{cmlm}$ as the pre-training loss; "- translation prediction" means we predict the masked n-grams themselves rather than their translation candidates during pre-training; "- - n-gram mask" means we randomly mask BPE tokens rather than n-grams based on "- translation prediction" during pre-training, which degrades our method to XLM.

our model to learn both monolingual and cross-lingual information during pre-training. Besides, we find the two modifications(translation prediction and n-gram mask) made by CMLM have nearly equal contributions to the translation performance, except for en2de, where the "n-gram mask" has little influence.

## 5 Related Work

Unsupervised machine translation dates back to Klementiev et al. (2012); Nuhn et al. (2012), but becomes a hot research topic in recent years. As the pioneering methods, Artetxe et al. (2017); Lample et al. (2017); Yang et al. (2018) are mainly the modifications of the encoder-decoder structure. The core idea is to constrain outputs of encoders of two languages into a same latent space with a weight sharing mechanism such as using a shared encoder. Denoising auto-encoder (Vincent et al., 2010) and adversarial training methods are also leveraged. Besides, they apply iterative back-translation to generated pseudo data for cross-lingual training. In addition to NMT methods for unsupervised machine translation, some following work shows that SMT methods and the hybrid of NMT and SMT can be more effective (Artetxe et al., 2018b; Lample et al., 2018; Marie and Fujita, 2018; Ren et al., 2019). They all build unsupervised PBSMT systems, and all of their models are initialized with language models and phrase tables inferred from cross-lingual word or n-gram embeddings and then use the initial PBSMT models to do iterative back-translation. Lample et al. (2018) also build a hybrid system by combining the best pseudo data that SMT models generate into the training of the NMT model while Ren et al. (2019) alternately train SMT and NMT models with the framework of posterior regularization.

More recently, Lample and Conneau (2019) reach new state-of-the-art performance on unsupervised en-fr and en-de translation tasks. They propose a cross-lingual language model pre-training method based on BERT (Devlin et al., 2018), and then treat two cross-lingual language models as the encoder and decoder to finish the translation. Leveraging much more monolingual data from Wikipedia, their work shows a big potential of pre-training for unsupervised machine translation. However, the cross-lingual information is obtained mostly from the shared BPE space during their pre-training method, which is inexplicit and limited. Therefore, we figure out a new pre-training method that gives the model more explicit and stronger cross-lingual information.

In the recent work of Song et al. (2019), they also mask several consecutive tokens in the source sentence, but jointly pre-train the encoder and decoder by making the decoder to predict those masked tokens in both the source and target sides. Their method is a good case of pre-training for seq-to-seq tasks but the cross-lingual information incorporated with their method is from BPE sharing, which is also implicit. Our proposed method can be combined with their method within a multi-task framework, which could be done in the future.

## 6 Conclusion

In this paper, we propose a novel cross-lingual pre-training method for unsupervised machine translation. In our method, we leverage Cross-lingual Masked Language Model (CMLM) to incorporate explicit and strong cross-lingual information into pre-trained models. Experimental results on en-fr, en-de, and en-ro language pairs demonstrate the effectiveness of our proposed method.

In the future, we may apply our pre-training method to other language pairs and delve into the performance of the pre-trained encoders on other NLP tasks, such as Name Entity Recognition.

## Acknowledgments

This work is supported in part by National Key R&D Program of China 2018YFB1700403, and NSFC U1636210&61421003.

## References

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018a. A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 789–798.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018b. Unsupervised statistical machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium. Association for Computational Linguistics.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2017. Unsupervised neural machine translation. *arXiv preprint arXiv:1710.11041*.

Mikel Artetxe and Holger Schwenk. 2018. Margin-based parallel corpus mining with multilingual sentence embeddings. *arXiv preprint arXiv:1811.01136*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Alexis Conneau, Guillaume Lample, Ruty Rinott, Adina Williams, Samuel R Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. Xnli: Evaluating cross-lingual sentence representations. *arXiv preprint arXiv:1809.05053*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dan Hendrycks and Kevin Gimpel. 2016. Bridging nonlinearities and stochastic regularizers with gaussian error linear units.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Alexandre Klementiev, Ann Irvine, Chris Callison-Burch, and David Yarowsky. 2012. Toward statistical machine translation without parallel corpora. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 130–140. Association for Computational Linguistics.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. arXiv:1901.07291.

Guillaume Lample, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2017. Unsupervised machine translation using monolingual corpora only. *arXiv preprint arXiv:1711.00043*.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, et al. 2018. Phrase-based & neural unsupervised machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5039–5049.

Benjamin Marie and Atsushi Fujita. 2018. Unsupervised neural machine translation initialized by unsupervised statistical machine translation. *arXiv preprint arXiv:1810.12703*.

Rada Mihalcea and Ted Pedersen. 2003. An evaluation exercise for word alignment. In *Proceedings of the HLT-NAACL 2003 Workshop on Building and using parallel texts: data driven machine translation and beyond*, pages 1–10.

Malte Nuhn, Arne Mauser, and Hermann Ney. 2012. Deciphering foreign language by combining language models and context vectors. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 156–164. Association for Computational Linguistics.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational linguistics*, 29(1):19–51.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

Shuo Ren, Zhirui Zhang, Shujie Liu, Ming Zhou, and Shuai Ma. 2019. Unsupervised neural machine translation with smt as posterior regularization. *arXiv preprint arXiv:1901.04112*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 86–96.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words

with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1715–1725.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. Mass: Masked sequence to sequence pre-training for language generation. *arXiv preprint arXiv:1905.02450*.

Wilson L Taylor. 1953. cloze procedure: A new tool for measuring readability. *Journalism Bulletin*, 30(4):415–433.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408.

Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. 2018. Unsupervised neural machine translation with weight sharing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 46–55.

Zhirui Zhang, Shujie Liu, Mu Li, Ming Zhou, and Enhong Chen. 2018. Joint training for neural machine translation models with monolingual data. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

# Latent Part-of-Speech Sequences for Neural Machine Translation

Xuewen Yang[1], Yingru Liu[1], Dongliang Xie[2], Xin Wang[1], and Niranjan Balasubramanian[1,3]

[1]Stony Brook University
[2]Beijing University of Posts and Telecommunications
[1]{xuewen.yang,yingru.liu,x.wang}@stonybrook.edu
[2]xiedl@bupt.edu.cn
[3]niranjan@cs.stonybrook.edu

## Abstract

Learning target side syntactic structure has been shown to improve Neural Machine Translation (NMT). However, incorporating syntax through latent variables introduces additional complexity in inference, as the models need to marginalize over the latent syntactic structures. To avoid this, models often resort to greedy search which only allows them to explore a limited portion of the latent space.

In this work, we introduce a new latent variable model, LaSyn, that captures the co-dependence between syntax and semantics, while allowing for effective and efficient inference over the latent space. LaSyn decouples direct dependence between successive latent variables, which allows its decoder to exhaustively search through the latent syntactic choices, while keeping decoding speed proportional to the size of the latent variable vocabulary. We implement LaSyn by modifying a transformer-based NMT system and design a neural expectation maximization algorithm that we regularize with part-of-speech information as the latent sequences. Evaluations on four different MT tasks show that incorporating target side syntax with LaSyn improves both translation quality, and also provides an opportunity to improve diversity.

## 1 Introduction

Syntactic information has been shown to improve the translation quality in NMT models. On the source side, syntax can be incorporated in multiple ways — either directly during encoding (Chen et al., 2018; Sennrich and Haddow, 2016; Eriguchi et al., 2016), or indirectly via multi-task learning to produce syntax informed representations (Eriguchi et al., 2017; Baniata et al., 2018; Niehues and Cho, 2017; Zaremoodi et al., 2018). On the target side, however, incorporating the syntax is more challenging due to the additional complexity in inference when decoding over latent states. To avoid this, existing methods resort to approximate inference over the latent states using a two-step decoding process (Gū et al., 2018; Wang et al., 2018a; Wu et al., 2017; Aharoni and Goldberg, 2017). Typically, the first stage decoder produces a beam of latent states, which serve as conditions to feed into the second stage decoder to obtain the target words. Thus, training and inference in these models can only explore a limited sub-space of the latent states.

In this work, we introduce LaSyn, a new target side syntax model that allows exhaustive exploration of the latent states to ensure a better translation quality. Similar to prior work, LaSyn approximates the co-dependence between syntax and semantics of the target sentences by modeling the joint conditional probability of the target words and the syntactic information at each position. However, unlike prior work, LaSyn eliminates the sequential dependence between the latent variables and simply infers the syntactic information at a given position based on the source text and the partial translation context. This allows LaSyn to search over a much larger set of latent state sequences. In terms of time complexity, unlike typical latent sequential models, LaSyn only introduces an additional term that is linear in the size of latent variable vocabulary and the length of the sentence.

We implement LaSyn by modifying a transformer-based encoder-decoder model. The implementation uses a hybrid decoder that predicts two posterior distributions: the probability of syntactic choices at each position $P(\mathbf{z}_n|\mathbf{x}, \mathbf{y}_{<n})$, and the probability of the word choices at each position conditioned on each of the possible values for the latent states $P(\mathbf{y}_n|\mathbf{z}_n, \mathbf{x}, \mathbf{y}_{<n})$. The model cannot be trained by directly optimizing the data log-likelihood



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 780–790, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

because of its non-convex property. We devise a neural expectation maximization (NEM) algorithm, whose E-step computes the posterior distribution of latent states under current model parameters, and M-step updates model parameters using gradients from back-propagation. Given some supervision signal for the latent variables, we can modify this EM algorithm to obtain a regularized training procedure. We use parts-of-speech (POS) tag sequences, automatically generated by an existing tagger, as the source of supervision.

Because the decoder is exposed to more latent states during training, it is more likely to generate diverse translation candidates. To obtain diverse sequences, we can decode the most likely translations for different POS tag sequences. This is a more explicit and effective way of performing diverse translation than other methods based on diverse or re-ranking beam search (Vijayakumar et al., 2018; Li and Jurafsky, 2016), or coarse codes planning (Shu and Nakayama, 2018).

We evaluate LaSyn on four translation tasks. Evaluations show that LaSyn outperforms models that only use partial exploration of the latent states for incorporating target side syntax. A diversity based evaluation also shows that when using different POS tag sequences during inference, LaSyn produces more diverse and meaningful translations compared to existing models.

## 2 A Latent Syntax Model for Decoding

In a standard sequence-to-sequence model, the decoder directly predicts the target sequence $\mathbf{y}$ conditioned on the source input $\mathbf{x}$. The translation probability $P(\mathbf{y}|\mathbf{x})$ is modeled directly using the probability of each target word $\mathbf{y}_n$ at time step $n$ conditioned on the source sequence $\mathbf{x}$ and the current partial target sequence $\mathbf{y}_{<n}$ as follows:

$$P(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}) = \prod_{n=1}^{N} P(\mathbf{y}_n|\mathbf{x},\mathbf{y}_{<n};\boldsymbol{\theta}) \qquad (1)$$

where, $\boldsymbol{\theta}$ denotes the parameters of both the encoder and the decoder.

In this work, we model syntactic information of target tokens using an additional sequence of variables, which captures the syntactic choices[1] at each time step. There are multiple ways of incorporating this additional information in a sequence-to-sequence model.

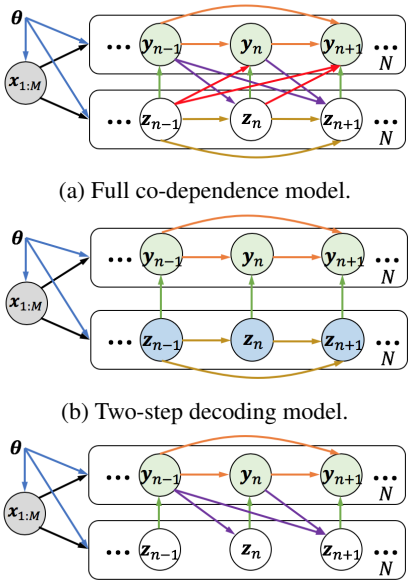


(a) Full co-dependence model.

(b) Two-step decoding model.

(c) LaSyn: Our Latent syntax model

Figure 1: Target-side Syntax Models: (a) An ideal solution that captures full co-dependence between syntax and semantics. (b) A widely-used two-step decoding model (Wang et al., 2018a; Wu et al., 2017; Aharoni and Goldberg, 2017). (c) LaSyn, our latent syntax model that uses non-sequential latent variables for exhaustive search of latent states.

An ideal solution should capture the co-dependence between syntax and semantics. In a sequential translation process, the word choices at each time step depend on both the semantics and the syntax of the words generated at the previous time steps. The same dependence also holds for the syntactic choices at each time step. Figure 1a shows a graphical model that captures this *full* co-dependence between the syntactic variable sequence $\mathbf{z}_1, \ldots, \mathbf{z}_N$ and the output word sequence $\mathbf{y}_1, \ldots, \mathbf{y}_N$. Such a model can be implemented using two decoders, one to decode syntax and the other to decode output words. The main difficulty, however, is that inference in this model is intractable since it involves marginalizing over the latent $\mathbf{z}$ sequences.

To keep inference tractable, existing approaches treat syntactic choices $\mathbf{z}$ as observed sequential variables (Gū et al., 2018; Wang et al., 2018a; Wu et al., 2017; Aharoni and Goldberg, 2017), as shown in Figure 1b. These models use a two-stage decoding process, where for each time step they

[1]The variables can be used to model any linguistic information that can be expressed as choices for each word position (e.g., morphological choices).

first produce most likely latent state $\mathbf{z}_n$ and then use this as input to a second stage that decodes words. However, this process is unsatisfactory in two respects. First, the inference of syntactic choices is still approximate as it does not explore the full space of $\mathbf{z}$. Second, these models are not well-suited for controllable or diverse translations. Using such a model to decode from an arbitrary $\mathbf{z}$ sequence is a divergence from its training, where it only learns to decode from a limited space of $\mathbf{z}$ sequences.

### 2.1 Model Description

Our goal is to design a model that allows for exhaustive search over syntactic choices. We introduce LaSyn, a new latent model shown in Figure 1c. The syntactic choices are modeled as *true latent variables* i.e., unobserved variables. Compared to the ideal model in Figure 1a, LaSyn includes two simplifications for tractability: (i) The dependence between successive syntactic choices is modeled indirectly, via the word choices made in the previous time steps. (ii) The word choice at each position depends only on the syntactic choice at the current position and the previous predicted words. Dependence on previous syntactic choices is modeled indirectly.

Under this model, the joint conditional probability of the target word $\mathbf{y}_n$ together with its corresponding latent syntactic choice $\mathbf{z}_n$[2] is given by:

$$P(\mathbf{y}_n, \mathbf{z}_n | \mathbf{x}, \mathbf{y}_{<n}) = P(\mathbf{y}_n | \mathbf{z}_n, \mathbf{x}, \mathbf{y}_{<n}) \times P(\mathbf{z}_n | \mathbf{x}, \mathbf{y}_{<n}) \quad (2)$$

We implement LaSyn by modifying the Transformer-based encoder-decoder architecture (Vaswani et al., 2017). As shown in Figure 2, LaSyn consists of a shared encoder for encoding source sentence $\mathbf{x}$ and a hybrid decoder that manages the decoding of the latent sequence $\mathbf{z}$ (left branch) and the target sentence $\mathbf{y}$ (right branch) separately.

The encoder consists of the standard self-attention layer, which generates representations of each token in the source sentence $\mathbf{x}$. The hybrid decoder consists of a self-attention layer that encodes the output generated thus far (i.e., the partial translation), followed by a inter-attention layer

[2]Note that $\mathbf{z}_n \in V_\mathbf{z}$, where $V_\mathbf{z}$ is the vocabulary of latent syntax for the target, which differs from language to language.

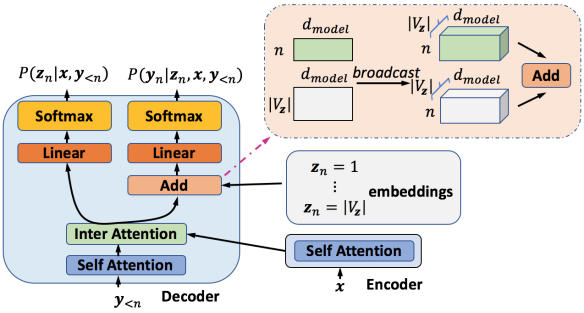


Figure 2: The architecture of LaSyn.

which computes the attention across the encoder and decoder representations.

The decoder's left branch predicts the latent variable distribution $P(\mathbf{z}_n | \mathbf{x}, \mathbf{y}_{<n})$ by applying a simple linear transformation and softmax on the inter-attention output, which contains information about the encoded input $\mathbf{x}$ and the partial translation $\mathbf{y}_{<n}$.

The right branch predicts the target word distribution $P(\mathbf{y}_n | \mathbf{z}_n, \mathbf{x}, \mathbf{y}_{<n})$ using the inter-attention output and the embeddings of all the available choices for $\mathbf{z}_n$. The choices for $\mathbf{z}_n$ are represented as embeddings that can be learned during training. We then combine the inter-attention output and the latent choice embeddings through an `Add` operation, which is a simple composition function that captures all combinations of additive interactions between the two. The dimension of the inter-attention is $n \times d_{model}$ and that of the latent embeddings is $|V_\mathbf{z}| \times d_{model}$, where $|V_\mathbf{z}|$ is the total number of choices for $\mathbf{z}_n$ or the size of the latent variable vocabulary. We broadcast them to the same dimension $n \times |V_\mathbf{z}| \times d_{model}$ and then simply add them together point-wise as shown in Figure 2. This is then fed to a linear transform and softmax over the target word vocabulary.

### 2.2 Inference with Exhaustive Search for Latent States

When using additional variables to model target side syntax, exact inference requires marginalizing over these additional variables.

$$P(\mathbf{y}|\mathbf{x}) = \sum_{\mathbf{z} \in F(\mathbf{z})} P(\mathbf{y}|\mathbf{z}, \mathbf{x}) P(\mathbf{z}|\mathbf{x}) \quad (3)$$

To avoid this exponential complexity, prior works use a two-step decoding process with models similar to the one shown in Figure 1b. They use greedy or beam search to explore a subset $B(\mathbf{z})$ of

the latent space to compute the posterior distribution as follows:

$$P(\mathbf{y}|\mathbf{x}) \simeq \sum_{\mathbf{z}\in B(\mathbf{z})} P(\mathbf{y}|\mathbf{z},\mathbf{x})P(\mathbf{z}|\mathbf{x}) \quad (4)$$

Finding the most likely translation using LaSyn also requires marginalizing over the latent states. However, because the latent states in LaSyn don't directly depend on each other, we can exhaustively search over the latent states. In particular, we can show that when $\mathbf{y}$ is fixed (observed), the $\{\mathbf{z}_n\}_{n=1}^{N}$ variables are d-separated (Bishop, 2006) i.e., are mutually independent. As a result, the time complexity for searching latent sequence $\mathbf{z}$ drops from $|V_\mathbf{z}|^N$ to $N|V_\mathbf{z}|$.

The posterior distribution for the translation probability at a time step $n$ can be computed as follows:

$$\begin{aligned} &P(\mathbf{y}_n|\mathbf{x},\mathbf{y}_{<n}) \\ &= \sum_{\mathbf{z}_n\in F(\mathbf{z}_n)} P(\mathbf{y}_n,\mathbf{z}_n|\mathbf{x},\mathbf{y}_{<n}) \\ &= \sum_{\mathbf{z}_n\in F(\mathbf{z}_n)} P(\mathbf{y}_n|\mathbf{z}_n,\mathbf{x},\mathbf{y}_{<n}) \times P(\mathbf{z}_n|\mathbf{x},\mathbf{y}_{<n}) \end{aligned} \quad (5)$$

where, $F(\mathbf{z}_n)$ is the full search space of latent states $\mathbf{z}_n$ and the joint probability is factorized as specified in Equation 2.

For decoding words, we use standard beam search[3] with $P(\mathbf{y}_n,\mathbf{z}_n|\mathbf{x},\mathbf{y}_{<n})$ as the beam cost. With this inference scheme, we can easily control decoding for diversity, by feeding different $\mathbf{z}$ sequences to the right branch of the decoder and decode diverse $\mathbf{y}_n$ by directly using $P(\mathbf{y}_n|\mathbf{z}_n,\mathbf{x},\mathbf{y}_{<n})$ as the beam cost. Unlike the two-step decoding models which only evaluate a small number of $\mathbf{z}_n$ values at each time step (constrained by beam size), LaSyn evaluates all possible values for $\mathbf{z}_n$ at each time step, while avoiding the evaluation of all possible sequences.

### 2.3 Training with Neural Expectation Maximization

The log-likelihood of LaSyn's parameters $\boldsymbol{\theta}$[4] computed on one training pair $\langle\mathbf{x},\mathbf{y}\rangle \in D$ is given by:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}) &= \log P(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}) \\ &= \sum_{n=1}^{N} \log P(\mathbf{y}_n|\mathbf{x},\mathbf{y}_{<n};\boldsymbol{\theta}) \\ &= \sum_{n=1}^{N} \log \sum_{\mathbf{z}_n\in V_\mathbf{z}} P(\mathbf{y}_n,\mathbf{z}_n|\mathbf{x},\mathbf{y}_{<n}) \end{aligned} \quad (6)$$

Directly optimizing the log-likelihood function (equation 6) with respect to model parameter $\boldsymbol{\theta}$ is challenging because of the highly non-convex function $P(\mathbf{y}_n,\mathbf{z}_n|\mathbf{x},\mathbf{y}_{<n})$ and the marginalization over $\mathbf{z}_n$.[5] Alternatively, we optimize the system parameters by Expectation Maximization (EM).

Using Jensen's inequality, equation (6) can be re-written as:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}) &= \sum_{n=1}^{N} \log \sum_{\mathbf{z}_n\in V_\mathbf{z}} Q(\mathbf{z}_n) \frac{P(\mathbf{y}_n,\mathbf{z}_n|\mathbf{x},\mathbf{y}_{<n})}{Q(\mathbf{z}_n)} \\ &\geq \sum_{n=1}^{N} \sum_{\mathbf{z}_n\in V_\mathbf{z}} Q(\mathbf{z}_n) \log \frac{P(\mathbf{y}_n,\mathbf{z}_n|\mathbf{x},\mathbf{y}_{<n})}{Q(\mathbf{z}_n)} \\ &= \mathcal{L}_{lower}(Q,\boldsymbol{\theta}) \end{aligned} \quad (7)$$

where $\mathcal{L}_{lower}(Q,\boldsymbol{\theta})$ is the lower bound of the log-likelihood and $Q$ is any auxiliary probability distribution defined on $\mathbf{z}_n$. $\boldsymbol{\theta}$ is omitted from the expression for simplicity.

We set $Q(\mathbf{z}_n) = P(\mathbf{z}_n|\mathbf{x},\mathbf{y}_{\leq n};\boldsymbol{\theta}^{old})$, the probability of the latent state computed by the decoder (shown in the left branch in Figure 2). Substituting this in equation (7), we obtain the lower bound as

$$\begin{aligned} &\mathcal{L}_{lower}(Q,\boldsymbol{\theta}) = \\ &\sum_{n=1}^{N} \sum_{\mathbf{z}_n\in V_\mathbf{z}} P(\mathbf{z}_n|\mathbf{x},\mathbf{y}_{\leq n};\boldsymbol{\theta}^{old}) \times \log P(\mathbf{y}_n,\mathbf{z}_n|\mathbf{x},\mathbf{y}_{<n};\boldsymbol{\theta}) \\ &- P(\mathbf{z}_n|\mathbf{x},\mathbf{y}_{\leq n};\boldsymbol{\theta}^{old}) \times \log P(\mathbf{z}_n|\mathbf{x},\mathbf{y}_{\leq n};\boldsymbol{\theta}^{old}) \\ &= \mathcal{Q}(\boldsymbol{\theta},\boldsymbol{\theta}^{old}) + C \end{aligned} \quad (8)$$

[3]Note our primary goal is to perform exhaustive search in the latent space. Search in the target vocabulary space remains exponential in our model.

[4]This includes the trainable parameters of the encoder, decoder, and the latent state embeddings.

[5]Note that marginalization is an issue during training, unlike in inference. As $P(y_n,z_n)$ is already an non-convex function with respect to $\theta$, summing $P(y_n,z_n)$ over different values of $z_n$ makes the function more complicated. Besides, we also need to compute gradients to update the parameters and computing the gradient of a log-of-sum function is costly and unstable. During the translation, we only need to compute the value of $\mathcal{L}(\theta)$ as score for beam searching. Therefore, the marginalization is not an issue.

where

$$\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{old}) = \sum_{n=1}^{N} \sum_{\mathbf{z}_n \in V_{\mathbf{z}}} P(\mathbf{z}_n | \mathbf{x}, \mathbf{y}_{\leq n}; \boldsymbol{\theta}^{old}) \times \log P(\mathbf{y}_n, \mathbf{z}_n | \mathbf{x}, \mathbf{y}_{<n}; \boldsymbol{\theta}) \quad (9)$$

EM algorithm for optimizing $\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{old})$ consists of two major steps. In the E-step, we compute the posterior distribution of $\mathbf{z}_n$ with respect to $\boldsymbol{\theta}^{old}$ by

$$\gamma(\mathbf{z}_n = i) = P(\mathbf{z}_n = i | \mathbf{x}, \mathbf{y}_{\leq n}) = \frac{P(\mathbf{y}_n, \mathbf{z}_n = i | \mathbf{x}, \mathbf{y}_{<n})}{\sum_{\mathbf{z}_n = j} P(\mathbf{y}_n, \mathbf{z}_n = j | \mathbf{x}, \mathbf{y}_{<n})} \quad (10)$$

where $\gamma(\mathbf{z}_n = i)$ is the responsibility of $\mathbf{z}_n = i$ given $\mathbf{y}_n$, and can be calculated by equation (2).

In the M-step, we aim to find the configuration of $\boldsymbol{\theta}$ that would maximize the expected log-likelihood using the posteriors computed in the E-step. In conventional EM algorithm for shallow probabilistic graphical model, the M-step is generally supposed to have closed-form solution. However, we model the probabilistic dependencies by deep neural networks, where $\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{old})$ is highly non-convex and non-linear with respect to network parameters $\boldsymbol{\theta}$. Therefore, there exists no analytical solution to maximize it. However, since deep neural network is differentiable, we can update $\boldsymbol{\theta}$ by taking a gradient ascent step:

$$\boldsymbol{\theta}^{new} = \boldsymbol{\theta}^{old} + \eta \frac{\partial \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{old})}{\partial \boldsymbol{\theta}}, \quad (11)$$

The resulting algorithm belongs to the class of *generalized EM algorithms* and is guaranteed (for a sufficiently small learning rate $\eta$) to converge to a (local) optimum of the data log likelihood (Wu, 1983).

### 2.4 Regularized EM training

The EM training we derived does not assume any supervision for the latent variables $\mathbf{z}$. This can be seen as inferring the latent syntax of the target sentences by clustering the target side tokens into $|V_{\mathbf{z}}|$ different categories. Given some token-level syntactic information, we can modify the training procedure to regularize the generation of latent sequence $P(\mathbf{z}_n | \mathbf{x}, \mathbf{y}_{<n})$ such that true latent sequences have higher probabilities under the model. In this work, we consider parts-of-speech sequences of the target sentences for regularization.

The regularized EM training objective is thus redefined as

$$\mathcal{L}_{total}(\boldsymbol{\theta}) = \mathcal{L}_{lower}(\boldsymbol{\theta}) + \lambda \mathcal{L}_{\mathbf{z}}(\boldsymbol{\theta}), \quad (12)$$

where $\mathcal{L}_{lower}(\boldsymbol{\theta})$ is the EM lower bound in equation (7) and $\mathcal{L}_{\mathbf{z}}(\boldsymbol{\theta})$ denotes cross entropy loss between $P(\mathbf{z}_n | \mathbf{x}, \mathbf{y}_{<n})$ and the true POS tag sequences and $\lambda$ is a hyper-parameter that controls the impact of the regularization.

This regularized training algorithm is shown in Algorithm 1.

**Algorithm 1** Training NMT with latent POS tag sequences through Regularized Neural EM

**Objective:** Maximize the log likelihood function $\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{old})$ with respect to $\boldsymbol{\theta}$ over observed variable $\mathbf{y}$ and latent variable $\mathbf{z}$, governed by parameters $\boldsymbol{\theta}$.
**Input:** Parallel corpus training data $\langle \mathbf{x}, \mathbf{y} \rangle$; POS tag sequence $\mathbf{z}$ of target sentence; the number of EM update steps per batch $K$.
**Initialize:** Initialize random values for the parameters $\boldsymbol{\theta}^{old}$.
**while** Training loss has not converged **do**
  Select $\langle \mathbf{x}, \mathbf{y} \rangle \in D$, parse $\mathbf{z}$ of $\mathbf{y}$.
  **for** $k \leftarrow 1$ to $K$ **do**
    **1. E-step:** Evaluate $\gamma(\mathbf{z}_n = i)$ for $i \leq |V_{\mathbf{z}}|$.
    **2. M-step:** Evaluate $\boldsymbol{\theta}^{new}$ given by equation (11).
    **3.** Let $\boldsymbol{\theta}^{old} \leftarrow \boldsymbol{\theta}^{new}$.
  **end for**
**end while**

## 3 Evaluation

We evaluate LaSyn on four translation tasks, including three with moderate sized datasets IWSLT 2014 (Cettolo et al., 2015) German-English (De-En), English-German (En-De), English-French (En-Fr), and one with a relatively larger dataset, the WMT 2014 English-German (En-De). We describe the datasets in more details in the appendix.

We compare against three types of baselines: (i) general Seq2Seq models that use no syntactic information, (ii) models that incorporate source side syntax directly, and multitask learning models which include syntax indirectly, and (iii) models that use syntax on the target side. We also define a LaSyn **Empirical Upper Bound (EUB)**, which is our proposed model using true POS tag sequences for inference.

We use BLEU as the evaluation metric (Papineni et al., 2002) for translation quality. For diverse translation evaluation, we utilize *distinct-1* score (Li et al., 2016) as the evaluation metric,

which is the number of distinct unigrams divided by total number of generated words.

For all translation tasks, we choose the *base* configuration of Transformer with $d_{model} = 512$. During training, we choose Adam optimizer (Kingma and Ba, 2015) with $\beta_1 = 0.9$, $\beta_2 = 0.98$ with initial learning rate is 0.0002 with 4000 warm-up steps. We describe additional implementation and training details in the Appendix.

### 3.1 Results on IWSLT'14 Tasks

Table 1 compares LaSyn versions against some of the state-of-the-art models on the IWSLT'14 dataset. LaSyn-K rows show results when varying the number of EM update steps per batch ($K$).

On the De-En task, LaSyn provides a 1.7 points improvements over the Transformer baseline, demonstrating that the LaSyn's improvements come from incorporating target side syntax effectively. This result is also better than a transformer-based source side syntax model by 1.5 points. LaSyn results are also better than the published numbers for LSTM-based models that use multi-task learning for source side and models that uses target side syntax. Note that since the underlying architectures are different, we only present these results to show that the results with LaSyn are comparable with other models that have incorporated syntax.

On the En-De task, our model achieves 29.2 in terms of BLEU score, with 2.6 points improvement over the Transformer baseline and 2.4 points improvement over Transformer-based Source Side Syntax model. Compared with NPMT (Huang et al., 2018), which is a BiLSTM based model, we achieve 3.84 point improvement.

On the En-Fr task, our model set a new state-of-the-art with a BLEU score of 40.6, which is 1.7 points improvement over the second best model which uses Transformer to incorporate source side syntax knowledge. Our model also surpasses the basic Transformer model by about 2.1 points.

We notice that across all tasks, the performance of our model improves with number of EM update steps per batch ($K$). With larger $K$ values, we get better lower bounds $\mathcal{L}_{lower}(\boldsymbol{\theta})$ on each training batch, thus leading to better optimization. For update steps beyond $K > 5$, the performance does not improve any more.

Last, the EUB row indicates the performance that can be obtained when feeding in the true POS tags. The large improvement here shows the potential for improvement when modeling target side syntax.

Table 2 shows one example where LaSyn produces correct translations for a long input sentence. The output of LaSyn is close to the reference and the output of LaSyn when given the gold POS tag sequence is even better, demonstrating the benefits of modeling syntax. The transformer model however fails to decode the later portions of the long input accurately.

### 3.2 Speed

We compare the speeds of our (un-optimized) implementation of LaSyn with a vanilla transformer with no latents in its decoder. Table 3 shows the training time per epoch, and the inference time for the whole test set. computed on the IWSLT'14 De-En task. When $K = 1$, LaSyn takes almost twice as much time as the vanilla transformer for training. Increasing $K$ increases training time further. For inference, LaSyn takes close to four times as much time compared to the vanilla Transformer. In terms of complexity, LaSyn only adds a linear term (in POS tag size to the decoding complexity. Specifically, its decoding complexity is $B \times O(m) \times O(|V_{\mathbf{z}}| \times N)$ where $B$ is beam size, $m$ is a constant proportional to the tag set size and $N$ is output size. As the table shows, empirically, our current implementation incurs $m \simeq 4$. We leave further optimizations for future work.

### 3.3 Diversity

We compare the diversity of translations using *distinct-1* score (Li et al., 2016), which is simply the number of distinct unigrams divided by total number of generated words. We use our model to generate 10 translations for each source sentence of the test dataset. We then compare our results with baseline Transformer. The result is shown in Table 4. Much like translation quality, LaSyn's diversity increases with number of EM updates and is better than diversity of the transformer and a source side encoder model.

#### 3.3.1 Controlling Diversity with POS Sequences

One of the main strengths of LaSyn is that it can generate translations conditioned on a given POS sequence. First, we present some examples that we generate by decoding over different POS tag sequences. Given a source sentence, we use

| Method Type | Model | BLEU | | |
|---|---|---|---|---|
| | | De-En | En-De | En-Fr |
| BiLSTM | BiLSTM (Denkowski and Neubig, 2017) | – | – | 34.8 |
| | Dual Learning (Wang et al., 2018b) | 32.35 | – | – |
| | AST (Cheng et al., 2018) | – | – | 38.03 |
| | NPMT (Huang et al., 2018) | – | 25.36 | – |
| Multi-Task (BiLSTM) | MTL-NMT (Niehues and Cho, 2017) | 27.78 | – | – |
| Source Side Syn. (Transformer) | Source-NMT (Sennrich and Haddow, 2016) | 33.5 | 26.8 | 38.9 |
| Target Side Syn. (BiLSTM) | DSP-NMT (Shu and Nakayama, 2018) | 29.78 | – | – |
| | Tree-decoder (Wang et al., 2018a) | 32.65 | – | – |
| Transformer | Transformer | 33.3 | 26.6 | 38.5 |
| | LaSyn (Unsupervised) | 30.8 | 25.2 | 34.5 |
| | LaSyn (K=1) | 34.63 | 28.1 | 39.7 |
| | LaSyn (K=3) | 34.91 | 28.9 | 40.4 |
| | LaSyn (K=5) | **35.0** | **29.2** | **40.6** |
| | LaSyn **EUB** | 51.4 | 47.3 | 54.2 |

Table 1: **IWSLT'14 English-German and English-French results** - shown are the BLEU scores of various models on TED talks translation tasks. We highlight the **best** model in bold.

| | |
|---|---|
| SRC | letztes jahr habe ich diese beiden folien gezeigt , um zu veranschaulichen , dass die arktische eiskappe , die fr annhernd drei millionen jahre die grsse der unteren 48 staaten hatte , um 40 prozent geschrumpft ist . |
| REF | last year i showed these two slides so that demonstrate that the arctic ice cap , which for most of the last three million years has been the size of the lower 48 states , has shrunk by 40 percent . |
| Transformer | last year , i showed these two slides to illustrate that the arctic ice caps that had the size of the lower 48 million states to 40 percent . |
| LaSyn | last year , i showed these two slides to illustrate that the arctic ice cap , which for nearly three million years had the size of the lower 48 states , was shrunk by 40 percent . |
| LaSyn (groundtrue POS) | last year i showed these two slides just to illustrate that the arctic ice cap , which for nearly about the last three million years has been the size of the lower 48 states , has shrunk by 40 percent . |

Table 2: Translation examples on IWSLT'14 De-En dataset from our model and the Transformer baseline. We put correct translation segment in blue and highlight the wrong one in red.

| Model | Training Time/Epoch↓ | Inference Time↓ |
|---|---|---|
| Transformer | **3.6** min | **12.8** s |
| LaSyn (K=1) | 6.3 min | 56.1 s |
| LaSyn (K=3) | 18.1 min | 55.6 s |
| LaSyn (K=5) | 28.0 min | 55.6 s |

Table 3: **IWSLT'14 De-En training and inference speed evaluation**. ↓ means the smaller the better. We highlight the **best** model in bold.

| Model | distinct-1 | | |
|---|---|---|---|
| | De-En | En-De | En-Fr |
| Transformer | 0.231 | 0.242 | 0.258 |
| Source-NMT | 0.232 | 0.244 | 0.260 |
| LaSyn (Unsupervised) | 0.228 | 0.231 | 0.239 |
| LaSyn (K=1) | 0.237 | 0.251 | 0.265 |
| LaSyn (K=3) | 0.241 | 0.253 | 0.270 |
| LaSyn (K=5) | **0.245** | **0.255** | **0.273** |
| LaSyn **EUB** | 0.328 | 0.516 | 0.354 |

Table 4: **IWSLT'14 En-De/De-En/En-Fr diversity translation evaluation**. We highlight the **best** model in bold.

LaSyn to provide the most-likely target pos tag sequence. Then, we obtain a random set of valid POS tag sequences that differ from this maximum likely sequence by some edit distance. For each of these randomly sampled POS tag sequences, we let LaSyn generate a translation that fits the sequence. Table 5 shows some example sentences. LaSyn is able to generate diverse translations that reflect the sentence structure implied by the input POS tags. However, in trying to fit the translation into the specified sequence, it deviates somewhat from the ideal translation.

To understand how diversity plays against translation quality, we also conduct a small scale quantitative evaluation. We pick a subset of the test dataset, and for each source sentence in this subset, we sample 10 POS tag sequences whose edit distance to their corresponding Top-1 POS tag sequence equal to a specific value, we then use them

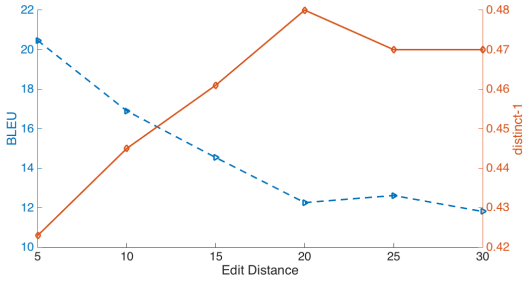


Figure 3: Diversity vs. Translation Quality: BLEU and *distinct-1* scores for targets decoded using POS sequences of increasing edit distance.

| | |
|---|---|
| SRC | und natrlich auch , wie nimmt gestaltung einfluss auf die wahrnehmung . |
| REF | and of course how design affects perception . |
| 0 | cc in nn wrb nn vbz vbz nn . |
| | and of course how design is affecting perception . |
| 20 | ls rb vb in dt nn vbz jj jj jj . |
| | i also think that the design is affecting perception . |
| 30 | rb , dt nn nn dt nn in prp rb . |
| | also , the way design adds influence in perception too . |
| 30 | prp vbz in prp vb vbn rb in nn . |
| | it turns out it included design impact on perception . |

Table 5: Examples of translations decoded from specified POS sequences with different edit distances (shown as values in first column). SRC: source sentence. REF: reference sentence.

to decode $W$ translations. We calculate their final BLEU and *distinct-1* scores. The results are shown in Figure: 3. As the edit distance increases, diversity increases dramatically but at the cost of translation quality. Since the POS tag sequence acts as a template for generation, as we move further from the most likely template, the model struggles to fit the content accurately. Understanding this trade-off can be useful for re-ranking or other scoring functions.

### 3.4 Results on WMT'14 En-De

To assess the impact on a larger dataset, we show results on the WMT'14 English-German in table 6. Compared to the previously reported systems, we see that our transformer implementation is a strong baseline. LaSyn produces small gains, with the best gain at K=5 – a BLEU score improvement of 0.6. This demonstrates that syntactic information can contribute more to the increase of translation quality on a smaller dataset.

## 4 Related Work

Attention-based Neural Machine Translation (NMT) models have shown promising results in various large scale translation tasks (Bahdanau et al., 2015; Luong et al., 2015; Sennrich et al., 2016; Vaswani et al., 2017) using an `Seq2Seq` structure. Many Statistical Machine Translation (SMT) approaches have leveraged benefits from modeling syntactic information (Chiang et al., 2009; Huang and Knight, 2006; Shen et al., 2008). Recent efforts have demonstrated that incorporating syntax can also be useful in neural methods as well.

| Model | BLEU |
|---|---|
| BiRNN+GCN (Bastings et al., 2017) | 23.9 |
| ConvS2S (Gehring et al., 2017) | 25.16 |
| MoE (Shazeer et al., 2017) | 26.03 |
| Transformer (base) | 27.3 |
| LaSyn (K=1) (base) | 27.6 |
| LaSyn (K=3) (base) | 27.8 |
| LaSyn (K=5) (base) | **27.9** |

Table 6: **WMT'14 English-German results** - shown are the BLEU scores of various models on TED talks translation tasks. We highlight the **best** model in bold.

One branch uses features on the source side to help improve the translation performance (Sennrich and Haddow, 2016; Morishita et al., 2018; Eriguchi et al., 2016). Sennrich *et al.* (2016) explored linguistic features like lemmas, morphological features, POS tags and dependency labels and concatenate their embeddings with sentence features to improve the translation quality. In a similar vein, Morishita *et al.* (2018) and Eriguchi *et al.* (2016), incorporated hierarchical subword features and phrase structure into the source side representations. Despite the promising improvements, these approaches are limited in that the trained translation model requires the availability of external tools during inference – the source text needs to be processed first to extract syntactic structure (Eriguchi et al., 2017).

Another branch uses multitask learning, where the encoder of the NMT model is trained to produce multiple tasks such as POS tagging, named-entity recognition, syntactic parsing or semantic parsing (Eriguchi et al., 2017; Baniata et al., 2018; Niehues and Cho, 2017; Zaremoodi et al., 2018). These can be seen as models that implicitly generate syntax informed representations during encoding. With careful model selection, these methods have demonstrate some benefits in NMT.

The third branch directly models the syntax of

the target sentence during decoding (Gū et al., 2018; Wang et al., 2018a; Wu et al., 2017; Aharoni and Goldberg, 2017; Bastings et al., 2017; Li et al., 2018). Aharoni *et al.* (2017) treated constituency trees as sequential strings and trained a `Seq2Seq` model to translate source sentences into these tree sequences. Wang *et al.* (2018a) and Wu *et al.* (2017) proposed to use two RNNs, a Rule RNN and a Word RNN, to generate a target sentence and its corresponding tree structure. Gu *et al.* (2018) proposed a model to translate and parse at the same time. However, apart from the complex tree structure to model, they all have a similar architecture as shown in Figure 1b, which limits them to only exploring a small portion of the syntactic space during inference.

LaSyn uses simpler parts-of-speech information in a latent syntax model, avoiding the typical exponential search complexity in the latent space with a linear search complexity and is optimized by EM. This allows for better translation quality as well as diversity. Similar to our work, (Shankar et al., 2018) and (Shankar and Sarawagi, 2019) proposed a latent attention mechanism to further reduce the complexity of model implementation by taking a top-K approximation instead of the EM algorithm as in LaSyn.

## 5 Conclusion

Modeling target-side syntax through true latent variables is difficult because of the additional inference complexity. In this work, we presented LaSyn, a latent syntax model that allows for efficient exploration of a large space of latent sequences. This yields significant gains on four translation tasks, IWSLT'14 English-German, German-English, English-French and WMT'14 English-German. The model also allows for better decoding of diverse translation candidates. This work only explored parts-of-speech sequences for syntax. Further extensions are needed to tackle tree-structured syntax information.

## 6 Acknowledgements

This work is supported in part by the National Science Foundation under Grants NSF CNS 1526843 and IIS 1815358. We thank Jiangbo Yuan, Wanying Ding, Yang Liu, Guillaume Rabusseau and Jeffrey Heinz for valuable discussions.

## References

Roee Aharoni and Yoav Goldberg. 2017. Towards string-to-tree neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 132–140.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*.

Laith H. Baniata, Seyoung Park, and Seong-Bae Park. 2018. A multitask-based neural machine translation model with part-of-speech tags integration for arabic dialects. *Applied Sciences*, 8.

Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Simaan. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1957–1967.

Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2015. Report on the 11 th iwslt evaluation campaign , iwslt 2014. In *Proceedings of IWSLT 2014*.

Kehai Chen, Rui Wang, Masao Utiyama, Eiichiro Sumita, and Tiejun Zhao. 2018. Syntax-directed attention for neural machine translation. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 4792–4799.

Yong Cheng, Zhaopeng Tu, Fandong Meng, Junjie Zhai, and Yang Liu. 2018. Towards robust neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 1756–1766.

David Chiang, Kevin Knight, and Wei Wang. 2009. 11,001 new features for statistical machine translation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 218–226.

Michael Denkowski and Graham Neubig. 2017. Stronger baselines for trustable results in neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 18–27.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-sequence attentional neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 823–833.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to parse and translate improves neural machine translation. In *Proceedings of the*

*55th Annual Meeting of the Association for Computational Linguistics*, pages 72–78.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1243–1252.

Jetic Gū, Hassan S. Shavarani, and Anoop Sarkar. 2018. Top-down tree structured decoding with syntactic connections for neural machine translation and parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 401–413.

Bryant Huang and Kevin Knight. 2006. Relabeling syntax trees to improve syntax-based machine translation quality. In *Proceedings of the Human Language Technology Conference of the NAACL*, pages 240–247.

Po-Sen Huang, Chong Wang, Sitao Huang, Dengyong Zhou, and Li Deng. 2018. Towards neural phrase-based machine translation. In *International Conference on Learning Representations*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119.

Jiwei Li and Dan Jurafsky. 2016. Mutual information and diverse decoding improve neural machine translation. *CoRR*, abs/1601.00372.

Xintong Li, Lemao Liu, Zhaopeng Tu, Shuming Shi, and Max Meng. 2018. Target foresight based attention for neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1380–1390.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Makoto Morishita, Jun Suzuki, and Masaaki Nagata. 2018. Improving neural machine translation by incorporating hierarchical subword features. In *COLING*, pages 618–629.

Jan Niehues and Eunah Cho. 2017. Exploiting linguistic resources for neural machine translation using multi-task learning. In *Proceedings of the Second Conference on Machine Translation*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. In *Proceedings of the First Conference on Machine Translation*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *ACL 2016*.

Shiv Shankar, Siddhant Garg, and Sunita Sarawagi. 2018. Surprisingly easy hard-attention for sequence to sequence learning. In *EMNLP*.

Shiv Shankar and Sunita Sarawagi. 2019. Posterior attention models for sequence to sequence learning. In *International Conference on Learning Representations*.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *ICLR*.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of ACL*, pages 577–585.

Raphael Shu and Hideki Nakayama. 2018. Discrete structural planning for neural machine translation.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Ashwin K. Vijayakumar, Michael Cogswell, Ramprasaath R. Selvaraju, Qing Sun, Stefan Lee, David J. Crandall, and Dhruv Batra. 2018. Diverse beam search for improved description of complex scenes. In *AAAI*, pages 7371–7379.

Xinyi Wang, Hieu Pham, Pengcheng Yin, and Graham Neubig. 2018a. A tree-based decoder for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4772–4777.

Yijun Wang, Yingce Xia, Li Zhao, Jiang Bian, Tao Qin, GuiQuan liu, and Tie-Yan Liu. 2018b. Dual transfer learning for neural machine translation with marginal distribution regularization. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

CF Jeff Wu. 1983. On the convergence properties of the EM algorithm. *The Annals of statistics*, pages 95–103.

Shuangzhi Wu, Dongdong Zhang, Nan Yang, Mu Li, and Ming Zhou. 2017. Sequence-to-dependency neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 698–707.

Poorya Zaremoodi, Wray Buntine, and Gholamreza Haffari. 2018. Adaptive knowledge sharing in multi-task learning: Improving low-resource neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

# Improving Back-Translation with Uncertainty-based Confidence Estimation

**Shuo Wang**[†], **Yang Liu**[†⋆*], **Chao Wang**[⊤], **Huanbo Luan**[†], **and Maosong Sun**[†]
[†]Institute for Artificial Intelligence
State Key Laboratory of Intelligent Technology and Systems
Department of Computer Science and Technology, Tsinghua University, Beijing, China
Beijing National Research Center for Information Science and Technology
[⋆]Beijing Advanced Innovation Center for Language Resources
[⊤]6ESTATES PTE LTD, Singapore

## Abstract

While back-translation is simple and effective in exploiting abundant monolingual corpora to improve low-resource neural machine translation (NMT), the synthetic bilingual corpora generated by NMT models trained on limited authentic bilingual data are inevitably noisy. In this work, we propose to quantify the confidence of NMT model predictions based on model uncertainty. With word- and sentence-level confidence measures based on uncertainty, it is possible for back-translation to better cope with noise in synthetic bilingual corpora. Experiments on Chinese-English and English-German translation tasks show that uncertainty-based confidence estimation significantly improves the performance of back-translation. [1]

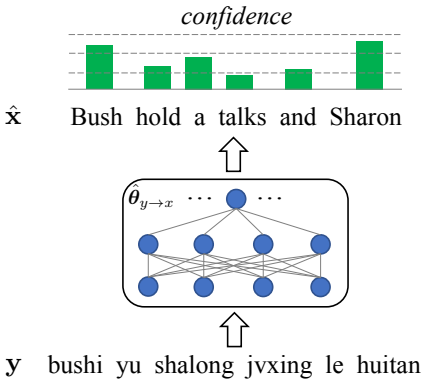


Figure 1: Confidence estimation for back-translation. Back-translation generates a source (e.g., English) sentence for a ground-truth target (e.g., Chinese) sentence. Such synthetic sentence pairs are used to train NMT models. As the model prediction (i.e., $\hat{\mathbf{x}}$) is often noisy, our work aims to quantify the prediction confidence using model uncertainty to alleviate error propagation.

## 1 Introduction

The past several years have witnessed the rapid development of end-to-end neural machine translation (NMT) (Sutskever et al., 2014; Bahdanau et al., 2015; Vaswani et al., 2017), which leverages neural networks to map between natural languages. Capable of learning representations from data, NMT has significantly outperformed conventional statistical machine translation (SMT) (Koehn et al., 2003) and been widely deployed in large-scale MT systems in the industry (Wu et al., 2016; Hassan et al., 2018).

Despite the remarkable success, NMT suffers from the data scarcity problem. For most language pairs, large-scale, high-quality, and wide-coverage bilingual corpora do not exist. Even for the top handful of resource-rich languages, the major sources of available parallel corpora are often restricted to government documents or news articles.

Therefore, improving NMT under small-data training conditions has attracted extensive attention in recent years (Sennrich et al., 2016a; Cheng et al., 2016; Zoph et al., 2016; Chen et al., 2017; Fadaee et al., 2017; Ren et al., 2018; Lample et al., 2018). Among them, back-translation (Sennrich et al., 2016a) is an important direction. Its basic idea is to use an NMT model trained on limited authentic bilingual corpora to generate synthetic bilingual corpora using abundant monolingual data. The authentic and synthetic bilingual corpora are then combined to re-train NMT models. Due to its simplicity and effectiveness, back-translation has been widely used in low-resource language translation. However, as the synthetic corpora generated by the NMT model are inevitably noisy, translation errors can be propagated to subsequent steps and prone to hinder the

* Yang Liu is the corresponding author: liuyang2011@tsinghua.edu.cn.

[1]The source code is available at https://github.com/THUNLP-MT/UCE4BT



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 791–802, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

performance (Fadaee and Monz, 2018; Poncelas et al., 2018).

In this work, we propose a method to quantify the confidence of NMT model predictions to enable back-translation to better cope with translation errors. The central idea is to use *model uncertainty* (Buntine and Weigend, 1991; Gal and Ghahramani, 2016; Dong et al., 2018; Xiao and Wang, 2019) to measure whether the model parameters can best describe the data distribution. Based on the expectation and variance of word- and sentence-level translation probabilities calculated by Monte Carlo Dropout (Gal and Ghahramani, 2016), we introduce various confidence measures.

Different from most previous quality estimation studies that require feature extraction (Blatz et al., 2004; Specia et al., 2009; Salehi et al., 2014) or post-edited data (Kim et al., 2017; Wang et al., 2018; Ive et al., 2018) to train external confidence estimators, all our approach needs is the NMT model itself. Hence, it is easy to apply our approach to arbitrary NMT models trained for arbitrary language pairs. Experiments on Chinese-English and English-German translation tasks show that our approach significantly improves the performance of back-translation.

## 2 Background

Let $\mathbf{x} = x_1 \dots x_I$ be a source-language sentence and $\mathbf{y} = y_1 \dots y_J$ be a target-language sentence. We use $P(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}_{x\to y})$ to denote a source-to-target NMT model (Sutskever et al., 2014; Bahdanau et al., 2015; Vaswani et al., 2017) parameterized by $\boldsymbol{\theta}_{x\to y}$. Similarly, the target-to-source NMT model is denoted by $P(\mathbf{x}|\mathbf{y}, \boldsymbol{\theta}_{y\to x})$.

Let $\mathcal{D}_b = \{\langle \mathbf{x}^{(m)}, \mathbf{y}^{(m)} \rangle\}_{m=1}^{M}$ be an *authentic* bilingual corpus that contains $M$ sentence pairs and $\mathcal{D}_m = \{\mathbf{y}^{(n)}\}_{n=1}^{N}$ be a monolingual corpus that contains $N$ target sentences. The first step of back-translation (Sennrich et al., 2016a) is to train a target-to-source model on the authentic bilingual corpus $\mathcal{D}_b$ using maximum likelihood estimation:

$$\hat{\boldsymbol{\theta}}_{y\to x} = \underset{\boldsymbol{\theta}_{y\to x}}{\operatorname{argmax}} \Big\{ L(\mathcal{D}_b, \boldsymbol{\theta}_{y\to x}) \Big\}, \quad (1)$$

where the log-likelihood is defined as

$$L(\mathcal{D}_b, \boldsymbol{\theta}_{y\to x}) = \sum_{m=1}^{M} \log P(\mathbf{x}^{(m)}|\mathbf{y}^{(m)}, \boldsymbol{\theta}_{y\to x}). \quad (2)$$

The second step is to use the trained model $\hat{\boldsymbol{\theta}}_{y\to x}$ to translate the monolingual corpus $\mathcal{D}_m$:

$$\hat{\mathbf{x}}^{(n)} = \underset{\mathbf{x}}{\operatorname{argmax}} \Big\{ P(\mathbf{x}|\mathbf{y}^{(n)}, \hat{\boldsymbol{\theta}}_{y\to x}) \Big\}, \quad (3)$$

where $\hat{\mathbf{x}}^{(n)} = \hat{x}_1^{(n)} \dots \hat{x}_I^{(n)}$. The word-level decision rule is given by

$$\hat{x}_i^{(n)} = \underset{x}{\operatorname{argmax}} \Big\{ P(x|\mathbf{y}^{(n)}, \hat{\mathbf{x}}_{<i}^{(n)}, \hat{\boldsymbol{\theta}}_{y\to x}) \Big\}. \quad (4)$$

The resulting translations $\{\hat{\mathbf{x}}^{(n)}\}_{n=1}^{N}$ can be combined with $\mathcal{D}_m$ to generate a *synthetic* bilingual corpus $\tilde{\mathcal{D}}_b = \{\langle \hat{\mathbf{x}}^{(n)}, \mathbf{y}^{(n)} \rangle\}_{n=1}^{N}$.

The third step is to train a source-to-target model $P(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}_{x\to y})$ on the combination of authentic and synthetic bilingual corpora:

$$\hat{\boldsymbol{\theta}}_{x\to y} = \underset{\boldsymbol{\theta}_{x\to y}}{\operatorname{argmax}} \Big\{ L(\mathcal{D}_b \cup \tilde{\mathcal{D}}_b, \boldsymbol{\theta}_{x\to y}) \Big\}. \quad (5)$$

This three-step process can iterate until convergence (Hoang et al., 2018; Cotterell and Kreutzer, 2018).

A problem with back-translation is that model predictions are inevitably erroneous. Translation errors can be propagated to subsequent steps and impair the performance of back-translation, especially when $\tilde{\mathcal{D}}_b$ is much larger than $\mathcal{D}_b$ (Pinnis et al., 2017; Fadaee and Monz, 2018; Poncelas et al., 2018). Therefore, it is crucial to develop principled solutions to enable back-translation to better deal with the error propagation problem.

## 3 Approach

This work aims to find solutions to the two following problems:

1. How to quantify the confidence of model predictions at both word and sentence levels?

2. How to leverage confidence to improve back-translation?

Section 3.1 introduces how to calculate model uncertainty, which lays a foundation for designing uncertainty-based word- and sentence-level confidence measures in Section 3.2. Section 3.3 describes confidence-aware training for NMT models on noisy bilingual corpora.

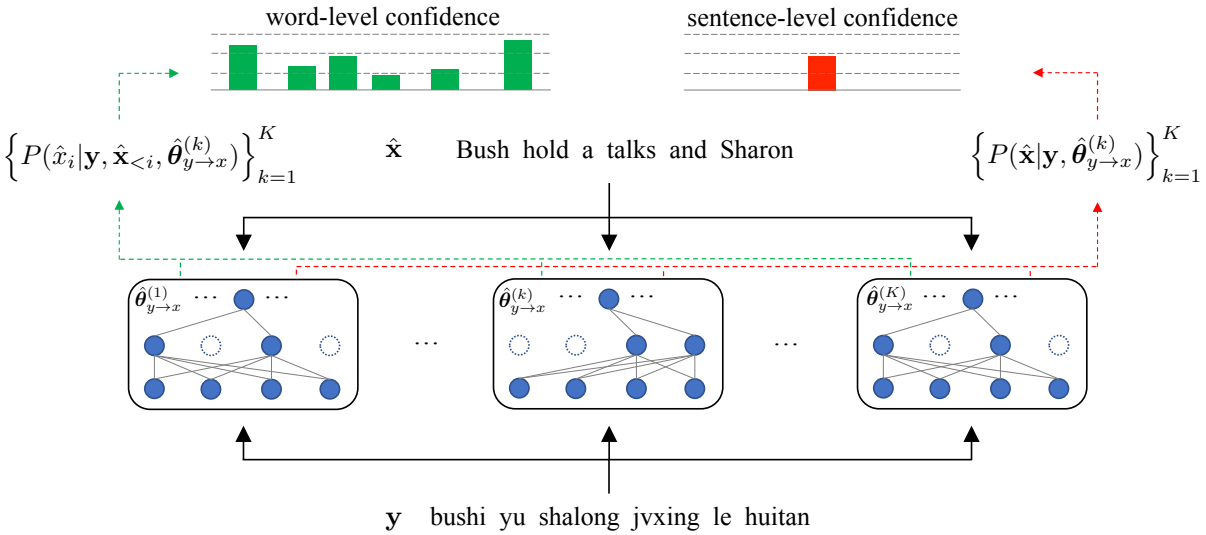


Figure 2: Illustration of uncertainty calculation. Given a target sentence $\mathbf{y}$ and the model prediction $\hat{\mathbf{x}}$, our approach treats word- and sentence-level translation probabilities as random variables and uses Monte Carlo Dropout to draw samples. These samples are used to calculate the expectations and variances of translation probabilities.

### 3.1 Calculating Uncertainty

Uncertainty quantification, which quantifies how confident a certain mapping is with respect to different inputs, has made significant progress due to the recent advances in Bayesian deep learning (Kendall et al., 2015; Gal and Ghahramani, 2016; Kendall and Gal, 2017; Xiao and Wang, 2019; Oh et al., 2019; Geifman et al., 2019; Lee et al., 2019). In this work, we aim to calculate *model uncertainty* (Kendall and Gal, 2017; Dong et al., 2018; Xiao and Wang, 2019), which measures whether a model can best describe the data distribution, for NMT using approximate inference methods widely used in Bayesian neural networks.

Given the authentic bilingual corpus $\mathcal{D}_b$, Bayesian neural networks aim at finding the posterior distribution over model parameters $P(\boldsymbol{\theta}_{y\to x}|\mathcal{D}_b)$. With a target sentence $\mathbf{y}$ in the monolingual corpus $\mathcal{D}_m$ and its translation $\hat{\mathbf{x}}$, the translation probability is given by

$$P(\hat{\mathbf{x}}|\mathbf{y}, \mathcal{D}_b) = \int P(\hat{\mathbf{x}}|\mathbf{y}, \boldsymbol{\theta}_{y\to x}) P(\boldsymbol{\theta}_{y\to x}|\mathcal{D}_b) d\boldsymbol{\theta}_{y\to x}. \quad (6)$$

In particular, we are interested in calculating the variance of the distribution $P(\hat{\mathbf{x}}|\mathbf{y}, \boldsymbol{\theta}_{y\to x})$ that reflects our ignorance over model parameters, which is referred to as *model uncertainty*. As exact inference is intractable, a number of variational inference methods (Graves, 2011; Blundell et al., 2015; Gal and Ghahramani, 2016) have been proposed to find an approximation to $P(\boldsymbol{\theta}_{y\to x}|\mathcal{D}_b)$. In this work, we leverage the widely used Monte Carlo Dropout (Gal and Ghahramani, 2016) to obtain samples of word- and sentence-level translation probabilities.

Figure 2 illustrates the key idea of our approach. Given an authentic target sentence $\mathbf{y}$, an NMT model made its prediction $\hat{\mathbf{x}}$ via a standard decoding process (see Eq. (3) and Eq. (4)). To quantify how confident the model was when making the prediction, our approach treats word- and sentence-level translation probabilities as random variables. [2] Drawing samples can be done by randomly deactivating part of neurons of the NMT model and re-calculating translation probabilities while keeping $\mathbf{y}$ and $\hat{\mathbf{x}}$ fixed. This stochastic feed-forward is repeated $K$ times and generates $K$ samples for both word- and sentence-level translation probabilities, respectively. We use $\hat{\boldsymbol{\theta}}^{(k)}_{y\to x}$ to denote the model parameters derived from $\hat{\boldsymbol{\theta}}_{y\to x}$ by deactivation in the $k$-th pass.

Intuitively, if the variance of translation probability is low, it is highly likely that the model was confident in making the prediction. Given $K$ samples $\{P(\hat{\mathbf{x}}|\mathbf{y}, \hat{\boldsymbol{\theta}}^{(k)}_{y\to x})\}_{k=1}^{K}$, the expectation

[2] Unlike prior studies that calculate model uncertainty during inference (Xiao and Wang, 2019), our approach computes uncertainty after the NMT model has made the prediction for two reasons. First, our goal is to quantify the confidence of model prediction rather than using uncertainty to improve model prediction. Second, using Monte Carlo Dropout during decoding is very slow because of the autoregressive property of standard NMT models.

of sentence-level translation probability can be approximated by

$$\mathbb{E}\Big[P(\hat{\mathbf{x}}|\mathbf{y},\hat{\boldsymbol{\theta}}_{y\to x})\Big] \approx \frac{1}{K}\sum_{k=1}^{K} P(\hat{\mathbf{x}}|\mathbf{y},\hat{\boldsymbol{\theta}}^{(k)}_{y\to x}). \quad (7)$$

The variance of sentence-level translation probability can be approximated by

$$\begin{aligned}&\mathrm{Var}\Big[P(\hat{\mathbf{x}}|\mathbf{y},\hat{\boldsymbol{\theta}}_{y\to x})\Big]\\ &\approx \frac{1}{K}\sum_{k=1}^{K} P(\hat{\mathbf{x}}|\mathbf{y},\hat{\boldsymbol{\theta}}^{(k)}_{y\to x})^2 - \mathbb{E}\Big[P(\hat{\mathbf{x}}|\mathbf{y},\hat{\boldsymbol{\theta}}_{y\to x})\Big]^2, \quad (8)\end{aligned}$$

which is also referred to as *model uncertainty*.

The expectation and variance of word-level translation probabilities can also be calculated similarly using $K$ samples.

### 3.2 Confidence Measures

We use $C(\mathbf{y},\hat{\mathbf{x}}_{<i},\hat{x}_i,\hat{\boldsymbol{\theta}}_{y\to x})$ to denote the *word-level confidence* for the model to generate $\hat{x}_i$ and $C(\mathbf{y},\hat{\mathbf{x}},\hat{\boldsymbol{\theta}}_{y\to x})$ to the denote the *sentence-level confidence* for the model to generate $\hat{\mathbf{x}}$.

Intuitively, when making predictions, the more confident an NMT model is, the higher expectation and lower variance of translation probability are. For comparison reasons, we used the following four types of confidence measures at the sentence level in our experiments:

1. *Predicted translation probability* (PTP). The translation probability of model prediction during standard decoding (Eq. (3)):

$$C_{\mathrm{PTP}}(\mathbf{y},\hat{\mathbf{x}},\hat{\boldsymbol{\theta}}_{y\to x}) = P(\hat{\mathbf{x}}|\mathbf{y},\hat{\boldsymbol{\theta}}_{y\to x}). \quad (9)$$

2. *Expected translation probability* (EXP). The expectation of translation probability:

$$C_{\mathrm{EXP}}(\mathbf{y},\hat{\mathbf{x}},\hat{\boldsymbol{\theta}}_{y\to x}) = \mathbb{E}\Big[P(\hat{\mathbf{x}}|\mathbf{y},\hat{\boldsymbol{\theta}}_{y\to x})\Big]. \quad (10)$$

3. *Variance of translation probability* (VAR). The variance of translation probability:

$$\begin{aligned}&C_{\mathrm{VAR}}(\mathbf{y},\hat{\mathbf{x}},\hat{\boldsymbol{\theta}}_{y\to x})\\ &= \Big(1 - \mathrm{Var}\Big[P(\hat{\mathbf{x}}|\mathbf{y},\hat{\boldsymbol{\theta}}_{y\to x})\Big]\Big)^{\alpha}. \quad (11)\end{aligned}$$

4. *Combination of expectation and variance* (CEV). The combination of expectation and variance:

$$\begin{aligned}&C_{\mathrm{CEV}}(\mathbf{y},\hat{\mathbf{x}},\hat{\boldsymbol{\theta}}_{y\to x})\\ &= \left(1 - \frac{\mathrm{Var}\big[P(\hat{\mathbf{x}}|\mathbf{y},\hat{\boldsymbol{\theta}}_{y\to x})\big]}{\mathbb{E}\big[P(\hat{\mathbf{x}}|\mathbf{y},\hat{\boldsymbol{\theta}}_{y\to x})\big]}\right)^{\beta}. \quad (12)\end{aligned}$$

where $\alpha$ and $\beta$ are hyper-parameters to control the gap between confidence values of sentences of different quality. Larger values of $\alpha$ and $\beta$ lead to bigger gaps. [3]

In Eq. (12), our approach tries to combine the merits of expectation and variance by using variance divided by expectation because smaller variance and bigger expectation are expected to result in higher confidence. There may exist more sophisticated ways to estimate prediction confidence using model uncertainty (Dong et al., 2018). As we find that the measures mentioned above are easy-to-implement and prove to be effective in our experiments, we leave the investigation of more complex confidence measures for future work.

The word-level confidence measures can be defined similarly.

### 3.3 Confidence-aware Training for NMT

We propose confidence-aware training for NMT to enable NMT to make better use of noisy data. Word- and sentence-level confidence measures are complementary: while word-level confidence can provide more fine-grained information than the sentence-level counterpart, it is unable to cope with word omission errors that can only be captured at the sentence level. As a result, our approach incorporates both word- and sentence-level confidence measures into the training process. [4]

#### Using Sentence-level Confidence

It is easy to integrate sentence-level confidence into back-translation by modifying the likelihood function in Eq. (5):

$$\begin{aligned}&L(\mathcal{D}_b \cup \tilde{\mathcal{D}}_b, \boldsymbol{\theta}_{x\to y})\\ =\ &\sum_{m=1}^{M} \log P(\mathbf{y}^{(m)}|\mathbf{x}^{(m)},\boldsymbol{\theta}_{x\to y}) +\\ &\sum_{n=1}^{N} C(\mathbf{y}^{(n)},\hat{\mathbf{x}}^{(n)},\hat{\boldsymbol{\theta}}_{y\to x}) \times\\ &\log P(\mathbf{y}^{(n)}|\hat{\mathbf{x}}^{(n)},\boldsymbol{\theta}_{x\to y}). \quad (13)\end{aligned}$$

[3] Note that all confidence measures are between 0 and 1. Clearly, both the expectation and variance of a probability are between 0 and 1. It can be proved that the variance of a probability is no greater than the corresponding expectation. As a result, $C_{\mathrm{CEV}}(\cdot)$ is also between 0 and 1.

[4] Instead of applying confidence estimation to the second pass of decoding (Luong et al., 2017), we directly integrate confidence scores into the training process. These two kinds of methods are complementary.

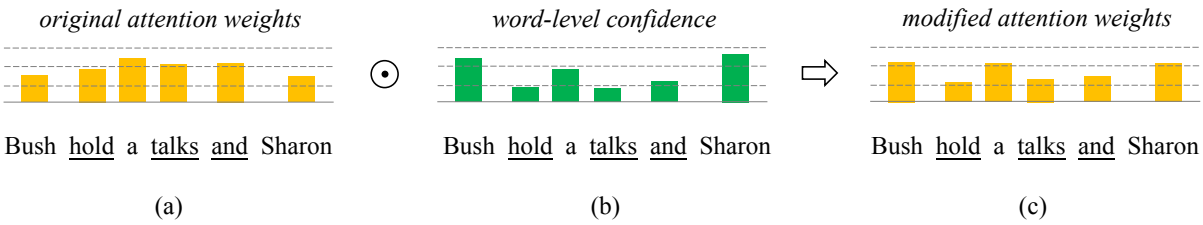


Figure 3: Using word-level confidence in confidence-aware training. The basic idea is to use confidence to modify attention weights to pay less attention to erroneous words highlighted in underline. (a) The original attention weights of the NMT model; (b) the word-level confidence of the noisy source sentence; (c) the attention weights modified by the word-level confidence, which focus more on words with high confidence. $\odot$ is a broadcast product. See Eq. (15) for details.

Serving as a weight assigned to each synthetic sentence pair, sentence-level confidence is expected to help to minimize the negative effect of estimating parameters on sentences with lower confidence. Note that the confidence of an authentic sentence pair in $\mathcal{D}_b$ is 1.

**Using Word-level Confidence**

As the source side instead of the target side of the synthetic bilingual corpus is noisy, word-level confidence cannot be integrated into back-translation in a similar way to sentence-level confidence. This is because the word-level confidence associated with each source word does not get involved in backpropagation during training.

Alternatively, we build a real-valued word-level confidence vector:

$$\mathbf{c} = \left\{ C\left(\mathbf{y}^{(n)}, \hat{\mathbf{x}}^{(n)}_{<i}, \hat{x}_i, \hat{\boldsymbol{\theta}}_{y \to x}\right) \right\}_{i=1}^{I}. \quad (14)$$

Due to the wide use of attention (Bahdanau et al., 2015; Vaswani et al., 2017) in NMT, we use the confidence vector $\mathbf{c} \in \mathbb{R}^{1 \times I}$ to modify attention weights and enable the model to focus more on words with high confidence. Figure 3 shows an example. Figure 3(a) gives a source sentence in the synthetic bilingual corpus, in which erroneous words "hold", "talks", and "and" receive high attention weights, deteriorating the parameter estimation on this sentence pair. By multiplying with word-level confidence (Figure 3(b)), the weights are modified to pay less attention to erroneous words (Figure 3(c)).

More formally, the modified attention function is given by

$$\begin{aligned} &\mathrm{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}, \mathbf{c}) \\ &= \left( \mathrm{softmax}\left( \frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{D}} \right) \odot \mathbf{c} \right) \mathbf{V}, \end{aligned} \quad (15)$$

where $\mathbf{Q} \in \mathbb{R}^{I \times D}$, $\mathbf{K} \in \mathbb{R}^{I \times D}$, and $\mathbf{V} \in \mathbb{R}^{I \times D}$ are query, key, and value matrices and $D$ is the hidden size. $\odot$ is a broadcast product.

Since the integration of sentence- and word-level confidence measures are independent of each other, it is easy to use both of them in back-translation.

## 4 Experiments

### 4.1 Setup

We evaluated our approach on Chinese-English and English-German translation tasks. The evaluation metric is BLEU (Papineni et al., 2001) as calculated by the `multi-bleu.perl` script. We use the paired bootstrap resampling (Koehn, 2004) for significance testing.

For the Chinese-English task, the training set contains 1.25M sentence pairs from LDC[5] with 27.8M Chinese words and 34.5M English words. To build the monolingual corpus for back-translation, we extracted the English side of the training set of the WMT 2017 Chinese-English news translation task. After removing sentences longer than 256 words, we randomly selected 10M English sentences as the monolingual corpus. NIST06 is used as the development set and NIST02, 03, 04, 05, and 08 datasets as test sets.

For the English-German task, we used the dataset of the WMT 2014 English-German translation task. The training set consists of 4.47M sentence pairs with 116M English words and 110M German words. We randomly selected 4.5M German sentences from the 2012 News Crawl corpus of WMT 2014 to construct the monolingual corpus for back-translation. We use newstest 2013 as

[5]The training set includes LDC2002E18, LDC2003E07, LDC2003E14, part of LDC2004T07, LDC2004T08 and LDC2005T06.

| Measure | BLEU | Δ |
|---|---|---|
| - | 46.23 | - |
| PTP | 45.41 | -0.82 |
| EXP | 45.22 | -1.01 |
| VAR | 46.77 | +0.54 |
| CEV | **47.05** | **+0.82** |

Table 1: Comparison of confidence measures.

| Word | Sentence | BLEU | Δ |
|---|---|---|---|
| × | × | 46.23 | - |
| × | √ | 46.42 | +0.19 |
| √ | × | 46.98 | +0.75 |
| √ | √ | **47.05** | **+0.82** |

Table 2: Comparison between word- and sentence-level CEV confidence measures.

the development set and newstest 2012, 2014, and 2015 as test sets.

Chinese sentences were segmented by an open-source toolkit THULAC[6] . German and English sentences were tokenized by the tokenizer in Moses (Koehn et al., 2007). We used byte pair encoding (Sennrich et al., 2016b) to perform sub-word segmentation with 32k merge operations for Chinese-English and 35k merge operations for English-German. Sentence pairs are batched together by approximate length and each batch has roughly 25,000 source and target tokens. We distinguish between three kinds of translations of the monolingual corpus:

1. NONE: there is no translation and only the authentic bilingual corpus is used;

2. SEARCH: the translations are generated by beam search (Sennrich et al., 2016a);

3. SAMPLE: the translations are generated by sampling (Edunov et al., 2018).

As neural quality estimation (Kim et al., 2017; Wang et al., 2018) can also give word- and sentence-level confidences for the output of NMT models when labeled data is available, we distinguish between two kinds of confidence estimation methods:

1. NEURALQE: the confidences are given by an external neural quality estimator;

2. UNCERTAINTY: the proposed uncertainty-based confidence estimation method.

For NEURALQE, we used the Predictor-Estimator architecture (Kim et al., 2017) implemented by OpenKiwi (Kepler et al., 2019), which is an open source software officially recommended by the QE shared task of WMT. Following the guide of OpenKiwi, we used a German-English parallel corpus containing 2.09M sentence pairs to train the predictor and a post-edited corpus containing 25k sentence triples to train the estimator. All the data used to train QE models are provided by WMT. As there are no post-edited corpora for the Chinese-English task, NEURALQE can only be used in the English-German task in our experiments. For NEURALQE, both word- and sentence- level quality scores were considered.

We implemented our method on the top of THUMT (Zhang et al., 2017). The NMT model we use is Transformer (Vaswani et al., 2017). We used the base model for the Chinese-English task and the big model for the English-German task. We used the Adam optimizer (Kingma and Ba, 2015) with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-9}$ to optimize model parameters. We used the same warm-up strategy for learning rate as Vaswani et al. (2017) with $\text{warmup\_steps} = 4,000$. During training, the hyper-parameter of label smoothing was set as $\epsilon_{ls} = 0.1$ (Szegedy et al., 2016; Pereyra et al., 2017). During training and the Monte Carlo Dropout process, the hyper-parameter of dropout was set to 0.1 and 0.3 for Transformer base and big models, respectively. $K$ was set to 20. Through experiments, we find our method works best when the $\alpha$ and $\beta$ are set to 2. All experiments were conducted on 8 NVIDIA GTX 1080Ti GPUs.

### 4.2 Comparison of Confidence Measures

Table 1 shows the comparison of confidence measures on the Chinese-English development set. We find that using either the translation probabilities outputted by the model (i.e., "PTP") or the expectation of translation probabilities (i.e., "EXP") deteriorates the translation quality, which suggests that translation probabilities themselves can not help NMT models better cope with synthetic data.

[6] https://github.com/thunlp/THULAC-Python

| Data | CE | MT06 | MT02 | MT03 | MT04 | MT05 | MT08 | All |
|---|---|---|---|---|---|---|---|---|
| NONE | - | 45.05 | 45.09 | 44.79 | 46.07 | 44.34 | 35.52 | 43.50 |
| SEARCH | - | 46.23 | 45.85 | 45.37 | 46.77 | 46.28 | 37.69 | 44.76 |
| | U | **47.05**$^{++}$ | **48.06**$^{++}$ | **46.44**$^{++}$ | **47.59**$^{++}$ | **47.03**$^{++}$ | **38.02**$^{+}$ | **45.72**$^{++}$ |
| SAMPLE | - | 46.69 | 46.98 | 45.62 | 46.97 | 46.29 | 37.28 | 44.96 |
| | U | **46.78** | 46.75 | **46.53**$^{\ddagger\ddagger}$ | **47.70**$^{\ddagger\ddagger}$ | **47.48**$^{\ddagger\ddagger}$ | 36.99 | **45.37**$^{\ddagger\ddagger}$ |

Table 3: BLEU scores on the NIST Chinese-English translation task. The ratio of authentic data to synthetic data is 1:1. NONE: only the authentic bilingual corpus is used. SEARCH: the translations of the monolingual corpus are generated by beam search (Sennrich et al., 2016a). SAMPLE: the translations of the monolingual corpus are generated by sampling (Edunov et al., 2018). "CE": confidence estimation method. "U": the proposed uncertainty-based confidence estimation. "All": the combination of all test sets. "+": significantly better than SEARCH without CE ($p < 0.05$). "++": significantly better than SEARCH without CE ($p < 0.01$). "‡‡": significantly better than SAMPLE without CE ($p < 0.01$).

In contrast, using the variance or model uncertainty (i.e., "VAR") increases translation quality. Combining variance and expectation (i.e., "CEV") leads to a further improvement. In the following experiments, we use CEV as the default setting.

### 4.3 Comparison between Word- and Sentence-level Confidence Measures

Table 2 shows the comparison between word- and sentence-level CEV (i.e., combination of expectation and variance) confidence measures on the Chinese-English development set. It is clear that using either sentence-level or word-level confidence measures improves the translation performance. Thanks to more fine-grained quantification of uncertainty, using word-level confidence achieves a higher BLEU score than using sentence-level confidence. Combining the sentence- and word-level of confidences leads to a further improvement, suggesting that they are complementary to each other. In the following experiments, we use the combination of word- and sentence-level confidences as the default setting.

### 4.4 Main Results

#### The Chinese-English Task

Table 3 shows the results of the Chinese-English task. Back-translation, either generating translations using beam search (i.e., SEARCH) or using sampling (i.e., SAMPLE), does lead to significant improvements over using only the authentic bilingual corpus (i.e., NONE). We find that SAMPLE is more effective than SEARCH, which confirms the finding of Edunov et al. (2018). Using uncertainty-based confidence (i.e., "U") significantly improves over both SEARCH and SAMPLE on the combination of all test sets ($p < 0.01$). As there is no Chinese-English labeled data to train neural quality estimation models, we did not report the result of NEURALQE in this experiment.

#### The English-German Task

Table 4 shows the results of the English-German task. We find that using quality estimation, either NEURALQE (i.e., "N") or UNCERTAINTY (i.e., "U"), improves over SEARCH and SAMPLE. UNCERTAINTY even achieves better performance than NEURALQE, although NEURALQE uses additional labeled training data. As NEURALQE heavily relies on post-edited corpora and labeled data to train QE models, it can only be used in a handful of language pairs. In contrast, it is easier to apply our approach to arbitrary language pairs since it does not need any labeled data to estimate confidence.

### 4.5 Effect of Training Corpus Size

Figure 4 shows the effect of training corpus size. The X-axis is the size of the total training data (i.e., $\mathcal{D}_b \cup \tilde{\mathcal{D}}_b$ in Eq. (5)). The BLEU scores were calculated on the Chinese-English development set. We find that the translation performance of SEARCH rises with the increase of monolingual corpus size in the beginning. However, further enlarging the monolingual corpus hurts the translation performance. In contrast, our approach can still obtain further improvements when adding more synthetic bilingual sentence pairs. Similar findings are also observed for SAMPLE.

| Data | CE | news2013 | news2012 | news2014 | news2015 | All |
|---|---|---|---|---|---|---|
| NONE | - | 26.57 | 22.09 | 28.42 | 30.26 | 26.31 |
| SEARCH | - | 27.09 | 23.10 | 29.45 | 30.10 | 27.04 |
| | N | 27.58 | 23.91 | 30.61 | 31.87 | 28.18 |
| | U | **27.89**$^{++*}$ | 23.75$^{++}$ | **31.00**$^{++*}$ | **31.98**$^{++}$ | **28.28**$^{++}$ |
| SAMPLE | - | 27.30 | 23.37 | 30.11 | 31.51 | 27.70 |
| | N | 27.55 | 23.53 | 30.13 | 31.87 | 27.87 |
| | U | **27.71**$^{‡‡}$ | **23.80**$^{‡}$ | **30.54**$^{‡‡††}$ | **32.01**$^{‡}$ | **28.15**$^{‡‡††}$ |

Table 4: BLEU scores on the WMT14 English-German translation task. The ratio of authentic data to synthetic data is 1:1. NONE: only the authentic bilingual corpus is used. SEARCH: the translations of the monolingual corpus are generated by beam search (Sennrich et al., 2016a). SAMPLE: the translations of the monolingual corpus are generated by sampling (Edunov et al., 2018). "CE": confidence estimation method. "U": uncertainty-based confidence estimation. "N": NEURALQE. "All": the combination of all test sets. "++": significantly better than SEARCH without CE ($p < 0.01$). "$*$": significantly better than "SEARCH + N" ($p < 0.05$). "‡": significantly better than SAMPLE without CE ($p < 0.05$). "‡‡": significantly better than SAMPLE without CE ($p < 0.01$). "††": significantly better than "SAMPLE + N" ($p < 0.01$).

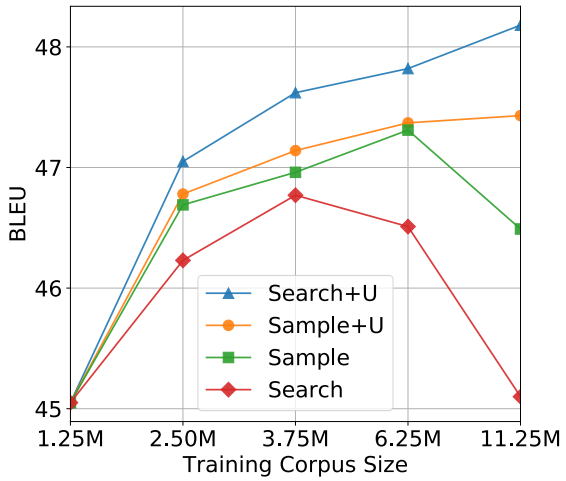


Figure 4: Effect of training corpus size.

### 4.6 Effect of Data Selection

Instead of randomly selecting monolingual sentences to generate synthetic data, we also used the method proposed by (Fadaee and Monz, 2018) to select monolingual data by targeting difficult words. In this series of experiments, we used the same amount of monolingual data that was derived from a larger monolingual corpus using different data selection methods.

Results on NIST06 show that targeting difficult words improves over randomly selecting monolingual data (46.23 → 46.60 BLEU), confirming the finding of Fadaee and Monz (2018). Using uncertainty-based confidence can further improve the translation performance (46.60 → 47.18 BLEU), indicating that our approach can be combined with advanced data selection methods.

### 4.7 Case Study

Figure 5 shows an example of model prediction and its corresponding word- and sentence-level confidence measures for the English-German task. We observe that the PTP and EXP measures are unable to give low confidence to erroneous words. In contrast, variance-based measures such as VAR and CEV can better quantify how confident the model is to make its prediction.

## 5 Related work

Our work is closely related to three lines of research: (1) back-translation, (2) confidence estimation, and (3) uncertainty quantification.

### 5.1 Back-translation

Back-translation is a simple and effective approach to leveraging monolingual data for NMT (Sennrich et al., 2016a). There has been a growing body of literature that analyzes and extends back-translation recently. Currey et al. (2017) show that low-resource NMT can benefit from the synthetic data generated by simply copying target monolingual data to the source side. Imamura et al. (2018) and Edunov et al. (2018) demonstrate that it is more effective to generate source sentences via sampling rather than beam search. Cotterell and Kreutzer (2018) and Hoang et al.

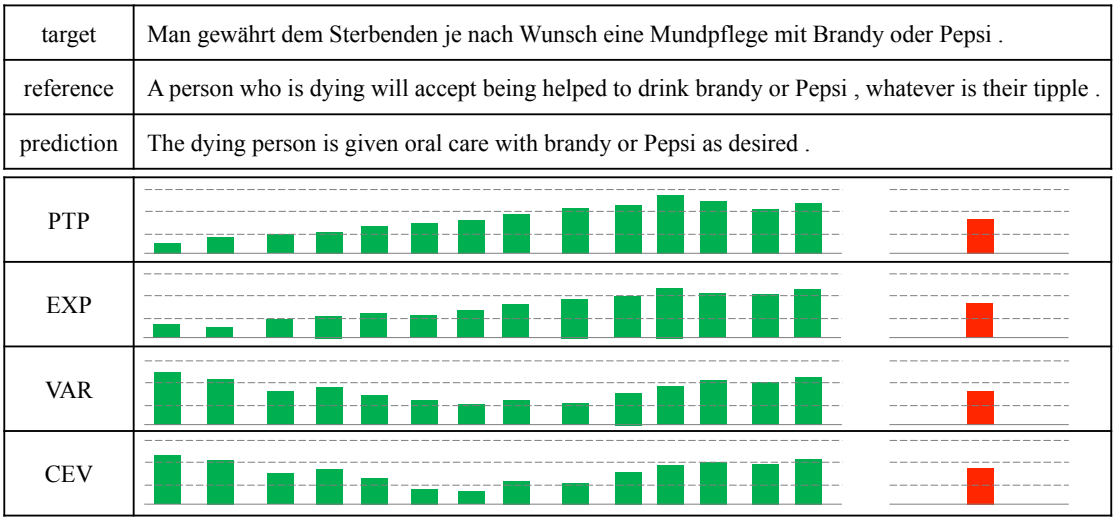


Figure 5: Example of confidence measures.

(2018) find that iterative back-translation can further improve the performance of NMT. Fadaee and Monz (2018) show that words with high predicted loss during training benefit most. Our work differs from existing methods in that we propose to use confidence estimation to enable back-translation to better cope with noisy synthetic data, which can be easily combined with previous works. Our experiments show that both neural and uncertainty-based confidence estimation methods benefit back-translation.

### 5.2 Confidence Estimation

Estimating the confidence or quality of the output of MT systems (Ueffing and Ney, 2007; Specia et al., 2009; Bach et al., 2011; Salehi et al., 2014; Rikters and Fishel, 2017; Kepler et al., 2019) is important for enabling downstream applications such as post-editing and interactive MT to better cope with translation mistakes. While existing methods rely on external models to estimate confidence, our approach leverages model uncertainty to derive confidence measures. The major benefit is that our approach does not need labeled data.

### 5.3 Uncertainty Quantification

Reliable uncertainty quantification is key to building a robust artificial intelligent system. It has been successfully applied to many fields, including computer vision (Kendall et al., 2015; Kendall and Gal, 2017), time series prediction (Zhu and Laptev, 2017), and natural language processing (Dong et al., 2018; Xiao and Wang, 2019). Our work differs from previous work in that we are interested in calculating uncertainty after the model has made the prediction rather during inference. Ott et al. (2018) also analyze the inherent uncertainty of machine translation. The difference is that they focus on the existence of multiple correct translations for a single sentence while we aim to quantify the uncertainty of NMT models.

## 6 Conclusions

We have presented a method for qualifying model uncertainty for neural machine translation and use uncertainty-based confidence measures to improve back-translation. The key idea is to use Monte Carlo Dropout to sample translation probabilities to calculate model uncertainty, without the need for manually labeled data. As our approach is transparent to model architectures, we plan to further verify the effectiveness of our approach on other downstream applications of NMT such as post-editing and interactive MT in the future.

## Acknowledgments

We thank all anonymous reviewers for their valuable comments. This work is supported by the National Key R&D Program of China (No. 2017YFB0202204), National Natural Science Foundation of China (No. 61761166008, No. 61432013), Beijing Advanced Innovation Center for Language Resources (No. TYR17002), and the NExT++ project supported by the National Research Foundation, Prime Ministers Office, Singapore under its IRC@Singapore Funding Initiative. This research is also supported by Sogou Inc.

## References


Nguyen Bach, Fei Huang, and Yaser Al-Onaizan. 2011. Goodness: A method for measuring machine translation confidence. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 211–219, Portland, Oregon, USA. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR 2015*.

John Blatz, Erin Fitzgerald, George Foster, Simona Gandrabur, Cyril Goutte, Alex Kulesza, Alberto Sanchis, and Nicola Ueffing. 2004. Confidence estimation for machine translation. In *Coling 2004: Proceedings of the 20th international conference on computational linguistics*, pages 315–321.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. 2015. Weight uncertainty in neural networks. In *Proceedings of ICML 2015*.

Wray Buntine and Andreas S. Weigend. 1991. Bayesian back-propagation. *Complex Systems*.

Yun Chen, Yang Liu, Yong Cheng, and Victor O.K. Li. 2017. A teacher-student framework for zero-resource neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1925–1935, Vancouver, Canada. Association for Computational Linguistics.

Yong Cheng, Wei Xu, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Semi-supervised learning for neural machine translation. In *Proceedings of ACL 2016*.

Ryan Cotterell and Julia Kreutzer. 2018. Explaining and generalizing back-translation through wake-sleep. *CoRR*, abs/1806.04402.

Anna Currey, Antonio Valerio Miceli Barone, and Kenneth Heafield. 2017. Copied monolingual data improves low-resource neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 148–156. Association for Computational Linguistics.

Li Dong, Chris Quirk, and Mirella Lapata. 2018. Confidence modeling for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 743–753, Melbourne, Australia. Association for Computational Linguistics.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of EMNLP 2018*.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. Data augmentation for low-resource neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 567–573, Vancouver, Canada. Association for Computational Linguistics.

Marzieh Fadaee and Christof Monz. 2018. Back-translation sampling by targeting difficult words in neural machine translation. In *Proceedings of EMNLP 2018*.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approaximation: Representing model uncertainty in deep learning. In *Proceedings of ICML 2016*.

Yonatan Geifman, Guy Uziel, and Ran El-Yaniv. 2019. Bias-reduced uncertainty estimation for deep neural classifiers. In *International Conference on Learning Representations*.

Alex Graves. 2011. Practical variational inference for neural networks. In *Proceedings of NeurIPS*.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, Shujie Liu, Tie-Yan Liu, Renqian Luo, Arul Menezes, Tao Qin, Frank Seide, Xu Tan, Fei Tian, Lijun Wu, Shuangzhi Wu, Yingce Xia, Dongdong Zhang, Zhirui Zhang, and Ming Zhou. 2018. Achieving human parity on automatic chinese to english news translation. arXiv:1803.05567.

Cong Duy Vu Hoang, Philipp Keohn, Gholamreza Haffari, and Trevor Cohn. 2018. Iterative back-translation for neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*.

Kenji Imamura, Atsushi Fujita, and Eiichiro Sumita. 2018. Enhancement of encoder and attention using target monolingual corpora in neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 55–63. Association for Computational Linguistics.

Julia Ive, Frédéric Blain, and Lucia Specia. 2018. deepQuest: A framework for neural-based quality estimation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3146–3157, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Alex Kendall, Vijay Badrinarayanan, and Roberto Cipolla. 2015. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. arXiv:1511.02680.

Alex Kendall and Yarin Gal. 2017. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pages 5574–5584.

Fabio Kepler, Jonay Trénous, Marcos Treviso, Miguel Vera, and André F. T. Martins. 2019. OpenKiwi: An open source framework for quality estimation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 117–122, Florence, Italy. Association for Computational Linguistics.

Hyun Kim, Jong-Hyeok Lee, and Seung-Hoon Na. 2017. Predictor-estimator using multilevel task learning with stack propagation for neural quality estimation. In *Proceedings of the Second Conference on Machine Translation*, pages 562–568, Copenhagen, Denmark. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 388–395.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL 2007*.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of NAACL 2003*.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Phrase-based & neural unsupervised machine translation. In *Proceedings of EMNLP 2018*.

Gilwoo Lee, Brian Hou, Aditya Mandalika, Jeongseok Lee, and Siddhartha S. Srinivasa. 2019. Bayesian policy optimization for model uncertainty. In *International Conference on Learning Representations*.

Ngoc-Quang Luong, Laurent Besacier, and Benjamin Lecouteux. 2017. Find the errors, get the better: Enhancing machine translation via word confidence estimation. *Natural Language Engineering*, 23:617–639.

Seong Joon Oh, Andrew C. Gallagher, Kevin P. Murphy, Florian Schroff, Jiyan Pan, and Joseph Roth. 2019. Modeling uncertainty with hedged instance embeddings. In *International Conference on Learning Representations*.

Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2018. Analyzing uncertainty in neural machine translation. In *Proceedings of ICML 2018*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL 2001*.

Gabriel Pereyra, George Tucker, Jan Chorowski, Lukasz Kaiser, and Geoffrey E. Hinton. 2017. Regularizing neural networks by penalizing confident output distributions. *CoRR*, abs/1701.06548.

Mārcis Pinnis, Rihards Krišlauks, Daiga Deksne, and Toms Miks. 2017. Neural machine translation for morphologically rich languages with improved subword units and synthetic data. In *Text, Speech, and Dialogue*, pages 237–245, Cham. Springer International Publishing.

Alberto Poncelas, Dimitar Shterionov, Andy Way, Gideon Maillette de Buy Wenniger, and Peyman Passban. 2018. Investigating backtranslation in neural machine translation. *CoRR*, abs/1804.06189.

Shuo Ren, Wenhu Chen, Shujie Liu, Mu Li, Ming Zhou, and Shuai Ma. 2018. Triangular architecture for rare language translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 56–65, Melbourne, Australia. Association for Computational Linguistics.

Matiss Rikters and Mark Fishel. 2017. Confidence through attention. *CoRR*, abs/1710.03743.

Marzieh Salehi, Shahram Khadivi, and Nooshin Riahi. 2014. Confidence estimation for machine translation using context vectors. *7'th International Symposium on Telecommunications (IST'2014)*, pages 524–528.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. In *Proceedings of ACL 2016*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725.

Lucia Specia, Marco Turchi, Zhuoran Wang, John Shawe-Taylor, and Craig Saunders. 2009. Improving the confidence of machine translation quality estimates. In *Twelfth Machine Translation Summit*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of NeurIPS 2014*.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826.

Nicola Ueffing and Hermann Ney. 2007. Word-level confidence estimation for machine translation. *Computational Linguistics*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of NeurIPS 2017*.

Jiayi Wang, Kai Fan, Bo Li, Fengming Zhou, Boxing Chen, Yangbin Shi, and Luo Si. 2018. Alibaba submission for WMT18 quality estimation task. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 809–815, Belgium, Brussels. Association for Computational Linguistics.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv:1609.08144v2.

Yijun Xiao and William Yang Wang. 2019. Quantifying uncertainties in natural language processing tasks. In *Proceedings of AAAI 2019*.

Jiacheng Zhang, Yanzhuo Ding, Shiqi Shen, Yong Cheng, Maosong Sun, Huan-Bo Luan, and Yang Liu. 2017. THUMT: an open source toolkit for neural machine translation. *CoRR*, abs/1706.06415.

Lingxue Zhu and Nikolay Laptev. 2017. Deep and confident prediction for time series at uber. In *Proceedings of ICDM 2017*.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *Proceedings of EMNLP 2016*.

# Towards Linear Time Neural Machine Translation with Capsule Networks

**Mingxuan Wang**[1] **Jun xie**[2] **Zhixing Tan**[2] **Jinsong Su**[3] **Deyi Xiong**[4] **Lei Li**[1]
[1]ByteDance AI Lab, Beijing, China
{wangmingxuan.89,lilei.lab}@bytedance.com
[2]Mobile Internet Group, Tencent Technology Co., Ltd
[3]Xiamen University, Xiamen, China
[4]Tianjin University, Tianjin, China

## Abstract

In this study, we first investigate a novel capsule network with dynamic routing for linear time Neural Machine Translation (NMT), referred as CAPSNMT. CAPSNMT uses an aggregation mechanism to map the source sentence into a matrix with pre-determined size, and then applys a deep LSTM network to decode the target sequence from the source representation. Different from the previous work (Sutskever et al., 2014) to store the source sentence with a passive and bottom-up way, the dynamic routing policy encodes the source sentence with an iterative process to decide the credit attribution between nodes from lower and higher layers. CAPSNMT has two core properties: it runs in time that is linear in the length of the sequences and provides a more flexible way to aggregate the part-whole information of the source sentence. On WMT14 English-German task and a larger WMT14 English-French task, CAPSNMT achieves comparable results with the Transformer system. To the best of our knowledge, this is the first work that capsule networks have been empirically investigated for sequence to sequence problems.[1]

## 1 Introduction

Neural Machine Translation (NMT) is an end-to-end learning approach to machine translation which has recently shown promising results on multiple language pairs (Luong et al., 2015; Shen et al., 2015; Wu et al., 2016; Gehring et al., 2017a; Kalchbrenner et al., 2016; Sennrich et al., 2015; Vaswani et al., 2017). Unlike conventional Statistical Machine Translation (SMT) systems (Koehn et al., 2003; Chiang, 2005) which consist of multiple separately tuned components, NMT aims at building upon a single and large neural network to directly map input text to associated output text (Sutskever et al., 2014).

In general, there are several research lines of NMT architectures, among which the Enc-Dec NMT (Sutskever et al., 2014) and the Enc-Dec Att NMT are of typical representation (Bahdanau et al., 2014; Wu et al., 2016; Vaswani et al., 2017). The Enc-Dec represents the source inputs with a fixed dimensional vector and the target sequence is generated from this vector word by word. The Enc-Dec, however, does not preserve the source sequence resolution, a feature which aggravates learning for long sequences. This results in the computational complexity of decoding process being $\mathcal{O}(|S| + |T|)$, with $|S|$ denoting the source sentence length and $|T|$ denoting the target sentence length. The Enc-Dec Att preserves the resolution of the source sentence which frees the neural model from having to squash all the information into a fixed represention, but at a cost of a quadratic running time. Due to the attention mechanism, the computational complexity of decoding process is $\mathcal{O}(|S| \times |T|)$. This drawbacks grow more severe as the length of the sequences increases.

Currently, most work focused on the Enc-Dec Att, while the Enc-Dec paradigm is less emphasized on despite its advantage of linear-time decoding (Kalchbrenner et al., 2016). The linear-time approach is appealing, however, the performance lags behind the Enc-Dec Att. One potential issue is that the Enc-Dec needs to be able to compress all the necessary information of the input source sentence into context vectors which are fixed during decoding. Therefore, a natural question was raised, *Will carefully designed aggregation operations help the Enc-Dec paradigm to achieve the best performance?*

In recent promising work of capsule network, a dynamic routing policy is proposed and proven to

[1]The work is partially done when the first author worked at Tencent.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 803–812,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

effective (Sabour et al., 2017; Zhao et al., 2018; Gong et al., 2018). Following a similar spirit to use this technique, we present CAPSNMT, which is characterized by capsule encoder to address the drawbacks of the conventional linear-time approaches. The capsule encoder processes the attractive potential to address the aggregation issue, and then introduces an iterative routing policy to decide the credit attribution between nodes from lower (child) and higher (parent) layers. Three strategies are also proposed to stabilize the dynamic routing process. We empirically verify CAPSNMT on WMT14 English-German task and a larger WMT14 English-French task. CAPSNMT achieves comparable results with the state-of-the-art Transformer systems. Our contributions can be summarized as follows:

- We propose a sophisticated designed linear-time CAPSNMT which achieved comparable results with the Transformer framework. To the best of our knowledge, CAPSNMT is the first work that capsule networks have been empirically investigated for sequence-to-sequence problems.

- We propose several techniques to stabilize the dynamic routing process. We believe that these technique should always be employed by capsule networks for the best performance.

## 2 Linear Time Neural Machine Translation

From the perspective of machine learning, the task of linear-time translation can be formalized as learning the conditional distribution $p(\mathbf{y}|\mathbf{x})$ of a target sentence (translation) $\mathbf{y}$ given a source sentence $\mathbf{x}$. Figure 1 gives the framework of our proposed linear time NMT with capsule encoder, which mainly consists of two components: a constant encoder that represents the input source sentence with a fixed-number of vectors, and a decoder which leverages these vectors to generate the target-side translation. Please note that due to the fixed-dimension representation of encoder, the time complexity of our model could be linear in the number of source words.

**Constant Encoder with Aggregation Layers** Given the input source sentence $\mathbf{x} = x_1, x_2 \cdots, x_L$, Then, we introduce capsule networks to transfer

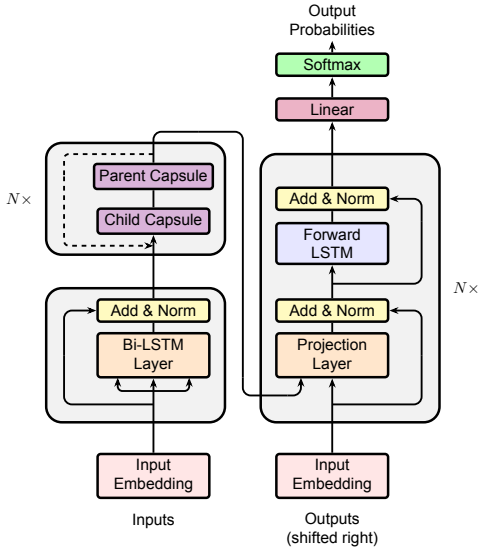


Figure 1: CAPSNMT:Linear time neural machine translation with capsule encoder

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_L] \in \mathbb{R}^{L \times d_x}.$$

The goal of the constant encoder is to transfer the inputs $\mathbf{X} \in \mathbb{R}^{L \times d_v}$ into a pre-determined size representation

$$\mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_M] \in \mathbb{R}^{M \times d_c}.$$

where $M < L$ is the pre-determined size of the encoder output, and $d_c$ is the dimension of the hidden states.

We first introduce a bi-directional LSTM (BiLSTM) as the primary-capsule layer to incorporate forward and backward context information of the input sentence:

$$\begin{aligned} \overrightarrow{\mathbf{h}_t} &= \overrightarrow{\mathbf{LSTM}}(\mathbf{h}_{t-1}, \mathbf{x}_t) \\ \overleftarrow{\mathbf{h}_t} &= \overleftarrow{\mathbf{LSTM}}(\mathbf{h}_{t+1}, \mathbf{x}_t) \\ \mathbf{h}_t &= [\overrightarrow{\mathbf{h}_t}, \overleftarrow{\mathbf{h}_t}] \end{aligned} \quad (1)$$

Here we produce the sentence-level encoding $\mathbf{h}_t$ of word $\mathbf{x}_t$ by concatenating the forward $\overrightarrow{\mathbf{h}_t}$ and backward output vector $\overleftarrow{\mathbf{h}_t}$. Thus, the output of BiLSTM encoder are a sequence of vectors $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_L]$ corresponding to the input sequence.

On the top of the BiLSTM, we introduce several aggregation layers to map the variable-length inputs into the compressed representation. To demonstrate the effectiveness of our encoder, we

compare it with a more powerful aggregation method, which mainly involves Max and Average pooling.

Both Max and Average pooling are the simplest ways to aggregate information, which require no additional parameters while being computationally efficient. Using these two operations, we perform pooling along the time step as follows:

$$\mathbf{h}^{max} = \max([\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_L]) \quad (2)$$

$$\mathbf{h}^{avg} = \frac{1}{L}\sum_{i=1}^{L}\mathbf{h}_i \quad (3)$$

Moreover, because the last time step state $h_L$ and the first time step state $h_1$ provide complimentary information, we also exploit them to enrich the final output representation of encoder. Formally, the output representation of encoder consists of four vectors,

$$\mathbf{C} = [\mathbf{h}^{max}, \mathbf{h}^{avg}, \mathbf{h}_1, \mathbf{h}_L]. \quad (4)$$

The last time step state $\mathbf{h}_L$ and the first time step state $\mathbf{h}_1$ provide complimentary information, thus improve the performance. The compressed represention $\mathbf{C}$ is fixed for the subsequent translation generation, therefore its quality directly affects the success of building the Enc-Dec paradigm. In this work, we mainly focus on how to introduce aggregation layers with capsule networks to accurately produce the compressed representation of input sentences, of which details will be provided in Section 3.

**LSTM Decoder** The goal of the LSTM is to estimate the conditional probability $p(y_t|y_{<t}; x_1, x_2, \cdots, x_T)$, where $(x_1, x_2, \cdots, x_T)$ is the input sequence and $(y_1, y_2, \cdots, y_{T'})$ is its corresponding output sequence.

A simple strategy for general sequence learning is to map the input sequence to a fixed-sized vector, and then to map the vector to the target sequence with a conditional LSTM decoder(Sutskever et al., 2014):

$$\begin{aligned}\mathbf{s}_t &= \mathbf{LSTM}(\mathbf{s}_{t-1}, \mathbf{u}_t) \\ \mathbf{u}_t &= \mathrm{ATT}(\mathbf{C}, \mathbf{s}_{t-1}) + \mathbf{y}_t\end{aligned} \quad (5)$$

where $\mathbf{y}_t$ is the target word embedding of $y_t$, $\mathbf{u}_t$ is the inputs of LSTM at time step $t$, $\mathbf{C}$ is the concatenation of the source sentence representation and $\mathrm{ATT}(\cdot)$ is the traditional attention function.

At inference stage, we only utilize the top-most hidden states $\mathbf{s}_t$ to make the final prediction with a softmax layer:

$$p(y_t|y_{t-1}, \mathbf{x}) = \mathbf{softmax}(\mathbf{W}_o\mathbf{s}_t). \quad (6)$$

Similar as (Vaswani et al., 2017), we also employ a residual connection (He et al., 2016) around each of the sub-layers, followed by layer normalization (Ba et al., 2016).

## 3 Aggregation layers with Capsule Networks

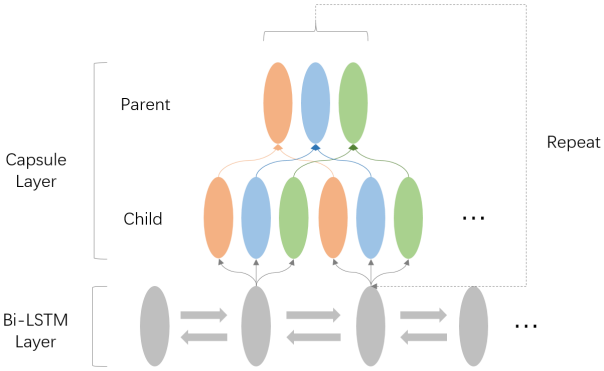


Figure 2: Capsule encoder with dynamic routing by agreement

The aggregation layers with capsule networks play a crucial role in our model. As shown in Figure 2, unlike the traditional linear-time approaches collecting information in a bottom-up approach without considering the state of the whole encoding, our capsule layer is able to iteratively decide the information flow, where the part-whole relationship of the source sentence can be effectively captured.

### 3.1 Child-Parent Relationships

To compress the input information into the representation with pre-determined size, the central issue we should address is to determine the information flow from the input capsules to the output capsules.

Capsule network is an ideal solution which is able to address the representational limitation and exponential inefficiencies of the simple aggregation pooling method. It allows the networks to automatically learn child-parent (or part-whole) relationships. Formally, $\mathbf{u}_{i\to j}$ denotes the information be transferred from the child capsule $\mathbf{h}_i$ into the

parent capsule $\mathbf{c}_j$:

$$\mathbf{u}_{i\to j} = \alpha_{ij}\mathbf{f}(\mathbf{h}_i, \boldsymbol{\theta}_j) \tag{7}$$

where $\alpha_{ij} \in \mathbb{R}$ can be viewed as the voting weight on the information flow from child capsule to the parent capsule; $\mathbf{f}(\mathbf{h}_i, \boldsymbol{\theta}_j)$ is the transformation function and in this paper, we use a single layer feed forward neural networks:

$$\mathbf{f}(\mathbf{h}_i, \boldsymbol{\theta}_j) = \mathbf{ReLU}(\mathbf{h}_i\mathbf{W}_j) \tag{8}$$

where $\mathbf{W}_j \in \mathbb{R}^{d_c\times d_c}$ is the transformation matrix corresponding to the position $j$ of the parent capsule.

Finally, the parent capsule aggregates all the incoming messages from all the child capsules:

$$\mathbf{v}_i = \sum_{j=1}^{L} \mathbf{u}_{j\to i} \tag{9}$$

and then squashes $\mathbf{v}_i$ to $||\mathbf{v}_i|| \in (0, 1)$ confine. ReLU or similar non linearity functions work well with single neurons. However we find that this squashing function works best with capsules. This tries to squash the length of output vector of a capsule. It squashes to 0 if it is a small vector and tries to limit the output vector to 1 if the vector is long.

$$\begin{aligned}\mathbf{c}_i &= \mathbf{squash}(\mathbf{v}_i) \\ &= \frac{||\mathbf{v}_i||^2}{1+||\mathbf{v}_i||^2}\frac{\mathbf{v}_i}{||\mathbf{v}_i||}\end{aligned} \tag{10}$$

### 3.2 Dynamic Routing by Agreement

The dynamic routing process is implemented via an EM iterative process of refining the coupling coefficient $\alpha_{ij}$ , which measures proportionally how much information is to be transferred from $\mathbf{h}_i$ to $\mathbf{c}_j$.

At iteration $t$, the coupling coefficient $\alpha_{ij}$ is computed by

$$\alpha_{ij}^t = \frac{\exp(b_{ij}^{t-1})}{\sum_k \exp(b_{ik}^{t-1})} \tag{11}$$

where $\sum \alpha_j \alpha ij$. This ensures that all the information from the child capsule $\mathbf{h}_i$ will be transferred to the parent.

$$b_{ij}^t = b_{ij}^{t-1} + \mathbf{c}_j^t \cdot \mathbf{f}(\mathbf{h}_i^t, \boldsymbol{\theta}_j^t) \tag{12}$$

This coefficient $b_{ij}^t$ is simply a temporary value that will be iteratively updated with the value $b_{ij}^{t-1}$ of the previous iteration and the scalar product of $\mathbf{c}_j^{t-1}$ and $\mathbf{f}(\mathbf{h}_j^{t-1}, \boldsymbol{\theta}_j^{t-1})$, which is essentially the similarity between the input to the capsule and the output from the capsule. Likewise, remember from above, the lower level capsule will send its output to the higher level capsule with similar output. Particularly, $b_{ij}^0$ is initialized with 0. The coefficient depends on the location and type of both the child and the parent capsules, which iteratively refinement of $b_{ij}$. The capsule network can increase or decrease the connection strength by dynamic routing. It is more effective than the previous routing strategies such as max-pooling which essentially detects whether a feature is present in any position of the text, but loses spatial information about the feature.

**Algorithm 1** Dynamic Routing Algorithm

**procedure** ROUTING($[\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_L]$,$T$)
  Initialize $\mathbf{b}_{ij}^0 \leftarrow 0$
  **for** each $t \in$ range$(0 : T)$ **do**
    Compute the routing coefficients $\alpha_{ij}^t$ for all $i \in [1, L], j \in [1, M]$ ▷ From Eq.(11)
    Update all the output capsule $\mathbf{c}_j^t$ for all $j \in [1, M]$ ▷ From Eq.(7,8,9,10)
    Update all the coefficient $\mathbf{b}_{ij}^t$ for all $i \in [1, L], j \in [1, M]$ ▷ From Eq.(12)
  **end for**
  **return** $[\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_M]$
**end procedure**

When an output capsule $\mathbf{c}_j^t$ receives the incoming messages $\mathbf{u}_{i\to j}^t$, its state will be updated and the coefficient $\alpha_{ij}^t$ is also re-computed for the input capsule $\mathbf{h}_i^{t-1}$. Thus, we iteratively refine the route of information flowing, towards an instance dependent and context aware encoding of a sequence. After the source input is encoded into $M$ capsules, we map these capsules into vector representation by simply concatenating all capsules:

$$\mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_M] \tag{13}$$

Finally matrix $\mathbf{C}$ will then be fed to the final end to end NMT model as the source sentence encoder.

In this work, we also explore three strategies to improve the accuracy of the routing process.

**Position-aware Routing strategy** The routing process iteratively decides what and how much information is to be sent to the final encoding with considering the state of both the final outputs cap-

sule and the inputs capsule. In order to fully exploit the order of the child and parent capsules to capture the child-parent relationship more efficiently, we add "positional encoding" to the representations of child and parent capsules. In this way, some information can be injected according to the relative or absolute position of capsules in the sequence. In this aspect, there are many choices of positional encoding proposed in many NLP tasks (Gehring et al., 2017b; Vaswani et al., 2017). Their experimental results strongly demonstrate that adding positional information in the text is more effective than in image since there is some sequential information in the sentence. In this work, we follow (Vaswani et al., 2017) to apply sine and cosine functions of different frequencies.

**Non-sharing Weight Strategy** Besides, we explore two types of transformation matrices to generate the message vector $\mathbf{u}_{i\to j}$ which has been previously mentioned Eq.(7,8). The first one shares parameters $\boldsymbol{\theta}$ across different iterations. In the second design, we replace the shared parameters with the non-shared strategy $\boldsymbol{\theta}^t$ where $t$ is the iteration step during the dynamic process. We will compare the effects of these two strategies in our experiments. In our preliminary, we found that non-shared weight strategy works slightly better than the shared one which is in consistent with (Liao and Poggio, 2016).

**Separable Composition and Scoring strategy** The most important idea behind capsule networks is to measure the input and output similarity. It is often modeled as a dot product function between the input and the output capsule, and the routing coefficient is updated correspondingly. Traditional capsule networks often resort to a straightforward strategy in which the "fusion" decisions (e.g., deciding the voting weight ) are made based on the values of feature-maps. This is essentially a soft template matching (Lawrence et al., 1997), which works for tasks like classification, however, is undesired for maintaining the composition functionality of capsules. Here, we propose to employ separate functional networks to release the scoring duty, and let $\boldsymbol{\theta}$ defined in Eq.(7) be responsible for composition. More specifically, we redefined the iteratively scoring function in Eq.(12) as follow,

$$b_{ij}^t = b_{ij}^{t-1} + \mathbf{g}(\mathbf{c}_j^t) \cdot \mathbf{g}(\mathbf{h}_i^t). \qquad (14)$$

Here $\mathbf{g}(\cdot)$ is a fully connected feed-forward network, which consists of two linear transformations with a ReLU activation and is applied to each position separately.

## 4 Experiments

### 4.1 Datasets

We mainly evaluated CAPSNMT on the widely used WMT English-German and English-French translation task. The evaluation metric is BLEU. We tokenized the reference and evaluated the performance with multi-bleu.pl. The metrics are exactly the same as in previous work (Papineni et al., 2002).

For English-German, to compare with the results reported by previous work, we used the same subset of the WMT 2014 training corpus that contains 4.5M sentence pairs with 91M English words and 87M German words. The concatenation of news-test 2012 and news-test 2013 is used as the validation set and news-test 2014 as the test set.

To evaluate at scale, we also report the results of English-French. To compare with the results reported by previous work on end-to-end NMT, we used the same subset of the WMT 2014 training corpus that contains 36M sentence pairs. The concatenation of news-test 2012 and news-test 2013 serves as the validation set and news-test 2014 as the test set.

### 4.2 Training details

Our training procedure and hyper parameter choices are similar to those used by (Vaswani et al., 2017). In more details, For English-German translation and English-French translation, we use $50K$ sub-word tokens as vocabulary based on Byte Pair Encoding(Sennrich et al., 2015). We batched sentence pairs by approximate length, and limited input and output tokens per batch to $4096$ per GPU.

During training, we employed label smoothing of value $\epsilon = 0.1$(Pereyra et al., 2017). We used a beam width of $8$ and length penalty $\alpha = 0.8$ in all the experiments. The dropout rate was set to $0.3$ for the English-German task and $0.1$ for the English-French task. Except when otherwise mentioned, NMT systems had $4$ layers encoders followed by a capsule layer and $3$ layers decoders. We trained for 300,000 steps on 8 M40 GPUs, and averaged the last $50$ checkpoints, saved at $30$ minute intervals. For our base model, the dimensions of all the hidden states were set to $512$ and

| SYSTEM | Architecture | Time | EN-Fr BLEU | EN-DE BLEU |
|---|---|---|---|---|
| Buck et al. (2014) | Winning WMT14 | - | 35.7 | 20.7 |
| Existing Enc-Dec Att NMT systems | | | | |
| Wu et al. (2016) | GNMT + Ensemble | $\|S\|\|T\|$ | 40.4 | 26.3 |
| Gehring et al. (2017a) | ConvS2S | $\|S\|\|T\|$ | 40.5 | 25.2 |
| Vaswani et al. (2017) | Transformer (base) | $\|S\|\|T\|+\|T\|\|T\|$ | 38.1 | 27.3 |
| Vaswani et al. (2017) | Transformer (large) | $\|S\|\|T\|+\|T\|\|T\|$ | 41.0 | 27.9 |
| Existing Enc-Dec NMT systems | | | | |
| Luong et al. (2015) | Reverse Enc-Dec | $\|S\|+\|T\|$ | - | 14.0 |
| Sutskever et al. (2014) | Reverse stack Enc-Dec | $\|S\|+\|T\|$ | 30.6 | - |
| Zhou et al. (2016) | Deep Enc-Dec | $\|S\|+\|T\|$ | 36.3 | 20.6 |
| Kalchbrenner et al. (2016) | ByteNet | c$\|S\|$+c$\|T\|$ | - | 23.7 |
| CapsNMT systems | | | | |
| Base Model | Simple Aggregation | c$\|S\|$+c$\|T\|$ | 37.1 | 21.3 |
| Base Model | CapsNMT | c$\|S\|$+c$\|T\|$ | 39.6 | 25.7 |
| Big Model | CapsNMT | c$\|S\|$+c$\|T\|$ | **40.0** | **26.4** |
| Base Model | Simple Aggregation + KD | c$\|S\|$+c$\|T\|$ | 38.6 | 23.4 |
| Base Model | CapsNMT + KD | c$\|S\|$+c$\|T\|$ | 40.4 | 26.9 |
| Big Model | CapsNMT+ KD | c$\|S\|$+c$\|T\|$ | **40.6** | **27.6** |

Table 1: Case-sensitive BLEU scores on English-German and English-French translation. KD indicates knowledge distillation (Kim and Rush, 2016).

for the big model, the dimensions were set to 1024. The capsule number is set to 6.

Sequence-level knowledge distillation is applied to alleviate multimodality in the training dataset, using the state-of-the-art transformer big models as the teachers(Kim and Rush, 2016). We decode the entire training set once using the teacher to create a new training dataset for its respective student.

### 4.3 Results on English-German and English-French Translation

The results on English-German and English-French translation are presented in Table 1. We compare CapsNMT with various other systems including the winning system in WMT'14 (Buck et al., 2014), a phrase-based system whose language models were trained on a huge monolingual text, the Common Crawl corpus. For Enc-Dec Att systems, to the best of our knowledge, GNMT is the best RNN based NMT system. Transformer (Vaswani et al., 2017) is currently the SOTA system which is about 2.0 BLEU points better than GNMT on the English-German task and 0.6 BLEU points better than GNMT on the English-French task. For Enc-Dec NMT, ByteNet is the previous state-of-the-art system which has 150 convolutional encoder layers and 150 convolutional decoder layers.

On the English-to-German task, our big CapsNMT achieves the highest BLEU score among all the Enc-Dec approaches which even outperform ByteNet, a relative strong competitor, by +3.9 BLEU score. In the case of the larger English-French task, we achieves the comparable BLEU score among all the with Big Transform, a relative strong competitor with a gap of only 0.4 BLEU score. To show the power of the capsule encoder, we also make a comparison with the simple aggregation version of the Enc-Dec model, and again yields a gain of +1.8 BLEU score on English-German task and +3.5 BLEU score on English-French task for the base model. The improvements is in consistent with our intuition that the dynamic routing policy is more effective than the simple aggregation method. It is also worth noting that for the small model, the capsule encoder approach get an improvement of +2.3 BLEU score over the Base Transform approach on English-French task. Knowledge also helps a lot to bridge the performance gap between CapsNMT and the state-of-the-art transformer model.

The first column indicates the time complexity of the network as a function of the length of the

sequences and is denoted by Time. The ByteNet, the RNN Encoder-Decoder are the only networks that have linear running time (up to the constant c). The RNN Enc-Dec, however, does not preserve the source sequence resolution, a feature that aggravates learning for long sequences. The Enc-Dec Att do preserve the resolution, but at a cost of a quadratic running time. The ByteNet overcomes these problem with the convolutional neural network, however the architecture must be deep enough to capture the global information of a sentence. The capsule encoder makes use of the dynamic routing policy to automatically learn the part-whole relationship and encode the source sentence into fixed size representation. With the capsule encoder, CAPSNMT keeps linear running time and the constant $c$ is the capsule number which is set to 6 in our mainly experiments.

### 4.4 Ablation Experiments

In this section, we evaluate the importance of our main techniques for training CAPSNMT. We believe that these techniques are universally applicable across different NLP tasks, and should always be employed by capsule networks for best performance. From Table 2 we draw the following conclusions:

| Model | BLEU |
|---|---|
| Base CAPSNMT | 24.7 |
| + Non-weight sharing | 25.1 |
| + Position-aware Routing Policy | 25.3 |
| + Separable Composition and Scoring | 25.7 |
| +Knowledge Distillation | 26.9 |

Table 2: English-German task: Ablation experiments of different technologies.

- **Non-weight sharing strategy** We observed that the non-weight sharing strategy improves the baseline model leading to an increase of 0.4 BLEU.
- **Position-aware Routing strategy** Adding the position embedding to the child capsule and the parent capsule can obtain an improvement of 0.2 BLEU score.
- **Separable Composition and Scoring strategy** Redefinition of the dot product function contributes significantly to the quality of the model, resulting in an increase of 0.4 BLEU score.
- **Knowledge Distillation** Sequence-level knowledge distillation can still achive an increase of +1.2 BLEU.

### 4.5 Model Analysis

In this section, We study the attribution of CAPSNMT.

**Effects of Iterative Routing** We also study how the iteration number affect the performance of aggregation on the English-German task. Figure 3 shows the comparison of $2-4$ iterations in the dynamic routing process. The capsule number is set to $2, 4, 6$ and $8$ for each comparison respectively. We found that the performances on several different capsule number setting reach the best when iteration is set to 3. The results indicate the dynamic routing is contributing to improve the performance and a larger capsule number often leads to better results.

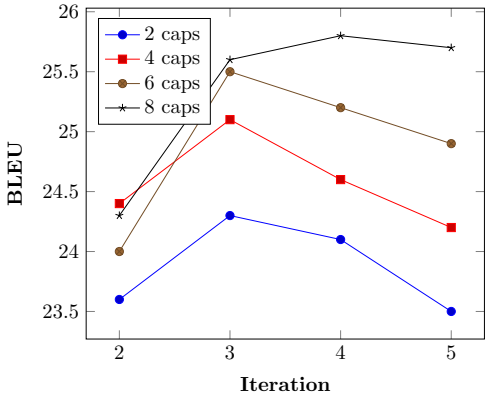


Figure 3: Effects of Iterative Routing with different capsule numbers

| Model | Num | Latency(ms) |
|---|---|---|
| Transformer | - | 225 |
| CAPSNMT | 4 | 146 |
| CAPSNMT | 6 | 153 |
| CAPSNMT | 8 | 168 |

Table 3: Time required for decoding with the base model. Num indicates the capsule number. Decoding indicates the amount of time in millisecond required for translating one sentence, which is averaged over the whole English-German newstest2014 dataset.

**Analysis on Decoding Speed** We show the decoding speed of both the transformer and CAP-

sNMT in Table 3. Thg results empirically demonstrates that CAPSNMT can improve the decoding speed of the transformer approach by $+50\%$.

**Performance on long sentences** A more detailed comparison between CAPSNMT and Transformer can be seen in Figure 4. In particular, we test the BLEU scores on sentences longer than $\{0, 10, 20, 30, 40, 50\}$. We were surprised to discover that the capsule encoder did well on medium-length sentences. There is no degradation on sentences with less than 40 words, however, there is still a gap on the longest sentences. A deeper capsule encoder potentially helps to address the degradation problem and we will leave this in the future work.

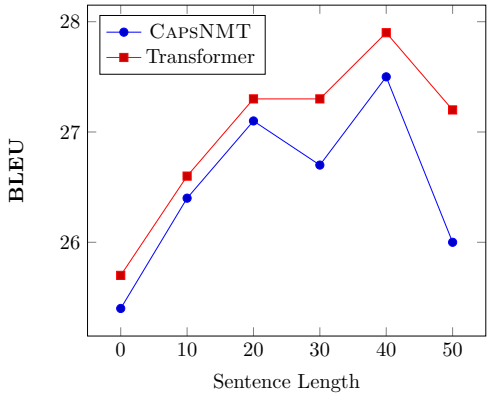


Figure 4: The plot shows the performance of our system as a function of sentence length, where the x-axis corresponds to the test sentences sorted by their length.

| |
|---|
| Orlando Bloom and Miranda Kerr still love each other |
| Orlando Bloom and Miranda Kerr still love each other |
| Orlando Bloom and Miranda Kerr still love each other |
| Orlando Bloom and Miranda Kerr still love each other |

Table 4: A visualization to show the perspective of a sentence from 4 different capsules at the third iteration.

**Visualization** We visualize how much information each child capsule sends to the parent capsules. As shown in Table 4, the color density of each word denotes the coefficient $\alpha_{ij}$ at iteration 3 in Eq.(11). At first iteration, the $\alpha_{ij}$ follows a uniform distribution since $b_{ij}$ is initialized to 0, and $\alpha_{ij}$ then will be iteratively fitted with dynamic routing policy. It is appealing to find that afte 3 iterations, the distribution of the voting weights$\alpha_{ij}$ will converge to a sharp distribution and the values will be very close to 0 or 1. It is also worth mentioning that the capsule seems able to capture some structure information. For example, the information of phrase *still love each other* will be sent to the same capsule. We will make further exploration in the future work.

## 5 Related Work

**Linear Time Neural Machine Translation** Several papers have proposed to use neural networks to directly learn the conditional distribution from a parallel corpus(Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Cho et al., 2014; Kalchbrenner et al., 2016). In (Sutskever et al., 2014), an RNN was used to encode a source sentence and starting from the last hidden state, to decode a target sentence. Different the RNN based approach, Kalchbrenner et al., (2016) propose ByteNet which makes use of the convolution networks to build the liner-time NMT system. Unlike the previous work, the CAPSNMT encodes the source sentence with an iterative process to decide the credit attribution between nodes from lower and higher layers.

**Capsule Networks for NLP** Currently, much attention has been paid to how developing a sophisticated encoding models to capture the long and short term dependency information in a sequence. Gong et al., (2018) propose an aggregation mechanism to obtain a fixed-size encoding with a dynamic routing policy. Zhao et al., (2018) explore capsule networks with dynamic routing for multi-task learning and achieve the best performance on six text classification benchmarks. Wang et al., (2018) propose RNN-Capsule, a capsule model based on Recurrent Neural Network (RNN) for sentiment analysis. To the best of our knowledge, CAPSNMT is the first work that capsule networks have been empirically investigated for sequence-to-sequence problems.

## 6 Conclusion

We have introduced CAPSNMT for linear-time NMT. Three strategies were proposed to boost the performance of dynamic routing process. The emperical results show that CAPSNMT can achieve state-of-the-art results with better decoding latency in several benchmark datasets.

## Acknowledgement

We thank the three anonymous reviewers for their valuable suggestions. Deyi Xiong was supported by the National Natural Science Foundation of China (Grant No. 61622209 and 61861130364)

## References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Christian Buck, Kenneth Heafield, and Bas Van Ooyen. 2014. N-gram counts and language models from the common crawl. In *LREC*, volume 2, page 4. Citeseer.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 263–270. Association for Computational Linguistics.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017a. Convolutional sequence to sequence learning. *CoRR*, abs/1705.03122.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017b. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

Jingjing Gong, Xipeng Qiu, Shaojing Wang, and Xuanjing Huang. 2018. Information aggregation via dynamic routing for sequence encoding. *international conference on computational linguistics*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709.

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099*.

Yoon Kim and Alexander M Rush. 2016. Sequence-level knowledge distillation. *arXiv preprint arXiv:1606.07947*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics.

Steve Lawrence, C Lee Giles, Ah Chung Tsoi, and Andrew D Back. 1997. Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, 8(1):98–113.

Qianli Liao and Tomaso Poggio. 2016. Bridging the gaps between residual learning, recurrent neural networks and visual cortex. *arXiv preprint arXiv:1604.03640*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. 2017. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. 2017. Dynamic routing between capsules. *neural information processing systems*, pages 3856–3866.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *CoRR*, abs/1508.07909.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2015. Minimum risk training for neural machine translation. *arXiv preprint arXiv:1512.02433*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Yequan Wang, Aixin Sun, Jialong Han, Ying Liu, and Xiaoyan Zhu. 2018. Sentiment analysis by capsules. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 1165–1174. International World Wide Web Conferences Steering Committee.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Wei Zhao, Jianbo Ye, Min Yang, Zeyang Lei, Suofei Zhang, and Zhou Zhao. 2018. Investigating capsule networks with dynamic routing for text classification. *arXiv: Computation and Language*.

Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, and Wei Xu. 2016. Deep recurrent models with fast-forward connections for neural machine translation. *arXiv preprint arXiv:1606.04199*.

# Modeling Multi-mapping Relations for Precise Cross-lingual Entity Alignment

**Xiaofei Shi**[1], **Yanghua Xiao**[1,2,3]
[1]School of Computer Science, Fudan University, China
[2]CETC Big Data Research Institute Co., Ltd., Guizhou, China
[3]Big Data Application on lmproving Government Governance Capabilities National Engineering Laboratory, Guizhou, China
{shixf18, shawyh}@fudan.edu.cn

## Abstract

Entity alignment aims to find entities in different knowledge graphs (KGs) that refer to the same real-world object. An effective solution for cross-lingual entity alignment is crucial for many cross-lingual AI and NLP applications. Recently many embedding-based approaches were proposed for cross-lingual entity alignment. However, almost all of them are based on TransE or its variants, which have been demonstrated by many studies to be unsuitable for encoding multi-mapping relations such as 1-N, N-1 and N-N relations, thus these methods obtain low alignment precision. To solve this issue, we propose a new embedding-based framework. Through defining dot product-based functions over embeddings, our model can better capture the semantics of both 1-1 and multi-mapping relations. We calibrate embeddings of different KGs via a small set of pre-aligned seeds. We also propose a weighted negative sampling strategy to generate valuable negative samples during training and we regard prediction as a bidirectional problem in the end. Experimental results (especially with the metric *Hits@1*) on real-world multilingual datasets show that our approach significantly outperforms many other embedding-based approaches with state-of-the-art performance.

## 1 Introduction

Knowledge bases in the form of knowledge graphs (KGs) contain lots of facts of the real world. In recent years, KGs have been successfully explored to serve many AI and NLP applications such as question answering, information extraction, semantic search and recommender systems. Multilingual KGs such as BabelNet (Navigli and Ponzetto, 2012), YAGO3 (Mahdisoltani et al., 2013) and DBpedia (Lehmann et al., 2015) play essential roles in many cross-lingual applications. However, for multilingual KGs, each language-specific part is constructed by different parties with different data sources. Thus these language-specific KGs contain different but complementary facts. To support cross-lingual applications, it is essential for us to integrate these language-specific KGs into a unified KG. Fortunately, existing inter-lingual links (ILLs) that link equivalent entities or relations can help us to achieve this goal. However, as Chen et al. (2017) and Sun et al. (2017) pointed out, the existing cross-lingual alignments usually account for a small proportion of the total. Therefore, it is valuable to find more cross-lingual entity alignment pairs to bridge the language gap in multilingual KGs.

Entity alignment is the task of finding entity pairs in different KGs that refer to the same real-world object. An effective solution for entity alignment is vital for integrating multiple KGs. In this paper, we focus on cross-lingual entity alignment. Our goal is to find more cross-lingual entity alignments based on existing alignment seeds.

Various methods have been explored for cross-lingual entity alignment. Traditional approaches rely on machine translation or feature engineering to find cross-lingual matching pairs. The effectiveness of these methods heavily depends on the quality of translations and defined features. Recently, many embedding-based approaches were proposed for cross-lingual entity alignment. Embedding-based approaches embed entities and relations as low-dimensional vectors, such that the similarity between entities or relations can be calculated via their vectors. For example, MTransE (Chen et al., 2017) embeds entities and relations separately in each KG and then learns the cross-lingual transitions between different language-specific embedding spaces. JAPE (Sun et al., 2017) encodes both structures of KGs and types of attribute values. GCN-EA (Wang et al., 2018) employs GCNs as its embedding model to encode entities.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 813–822, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

Though these methods have made great progress in cross-lingual entity alignment, there are still several key issues worth study. To the best of our knowledge, almost all of the existing methods encode knowledge in KGs based on the well-known TransE (Bordes et al., 2013) and its variants (Sun et al., 2017; Zhu et al., 2017). Given a relational triple $\tau = (h, r, t)$, where $r$ represents a specific relation between the head entity $h$ and the tail entity $t$, TransE and its variants expect $\vec{v_h} + \vec{v_r} - \vec{v_t} \approx \vec{0}$, where $\vec{v_h}$, $\vec{v_r}$ and $\vec{v_t}$ denote the embeddings for $h$, $r$ and $t$, respectively. However, as Wang et al. (2014) and Ebisu and Ichise (2018) pointed out: 1) TransE and its variants neglect some mapping properties of relations; and 2) TransE and its variants force entity embeddings to be on a sphere in the embedding space, which conflicts with the expectation of TransE and warps embeddings obtained by TransE. Thus TransE and its variants don't do well in modeling multi-mapping relations, such as 1-N, N-1 and N-N relations.

As the majority of relations in real KGs are multi-mapping relations, previous entity alignment approaches obtain low accuracy, especially with the metric *Hits@1*. And the gap between alignment results with the metric *Hits@1* and *Hits@10* in these methods is large. This means previous methods can cluster similar entities together, however, they don't have enough ability to distinguish which is the true counterpart. Although following works such as TransH (Wang et al., 2014), TransR (Lin et al., 2015b) and TransD (Ji et al., 2015) have extended TransE on modeling multi-mapping relations, we experimentally found these methods also have low performances for cross-lingual entity alignment. And though the GCN-based approach avoids the flaw in TransE and its variants, GCN-EA (Wang et al., 2018) only encodes entities in KGs. It doesn't distinguish among relations and the absence of relation embeddings results in low alignment precision, too.

Besides, most previous works use uniform sampling to generate negative samples during training, which often generates too many easy examples that contribute little to learning embeddings. Although Boot-EA (Sun et al., 2018) proposes a truncated method to select hard negative samples based on the semantic similarity, it only considers hard examples (easy ones are truncated) that only account for a small proportion of all examples, which can lead to overfitting and low accuracy in practice (since easy examples are also useful).

To address the above issues, we propose a new embedding-based framework to do cross-lingual entity alignment in this paper. Motivated by the success of multiplicative approaches (Yang et al., 2015; Trouillon et al., 2016) in knowledge representation learning, we propose a new embedding approach for multilingual KGs. The score function of our method is based on multiplication instead of subtraction. Unlike TransE and its variants giving a strong constraint to embeddings, dot products of embeddings in our method scale well and can naturally handle both 1-1 and multi-mapping relations. Besides, we propose a weighted sampling strategy to pay more attention to harder examples than easier ones during the process of negative samples generating. We summarize the main contributions of this paper as follows:

- We propose a new KG embedding framework to jointly embed entities and relations from different KGs into a unified embedding space using a small set of pre-aligned alignment pairs obtained from inter-lingual links.
- We discuss the problem that generating too many easy negative examples during training hurts alignment performance and then we propose a weighted sampling strategy for hard negative example generation.
- We regard the alignment prediction as a bidirectional selection problem, and we propose to combine ranking matrices from both directions to align entities.
- We evaluate the proposed approach on three real-world cross-lingual datasets. Experimental results (especially with the metric *Hits@1*) show that our approach significantly outperforms four representative embedding-based methods for cross-lingual entity alignment.

## 2 Related Work

### 2.1 KG Embedding

In the past few years, many KG embedding methods have been proposed for modeling the semantics of KGs. TransE (Bordes et al., 2013)

interprets a relation as the translation from its head entity to its tail entity and achieves great success in many AI-related tasks. Following TransE, TransH (Wang et al., 2014), TransR (Lin et al., 2015b) and TransD (Ji et al., 2015) were proposed to extend TransE on modeling multi-mapping relations. PTransE (Lin et al., 2015a) and RTransE (García-Durán et al., 2015) extend TransE on modeling multi-step relation paths. There are also non-translation-based methods. ComplEx (Trouillon et al., 2016) learns embeddings via defining product-based functions over embeddings. And some methods utilizing extra resources, i.e. visual information (Xie et al., 2017) and entity descriptions (Xie et al., 2016; Xiao et al., 2017), were also proposed to improve embedding performance.

### 2.2 Cross-lingual Entity Alignment

Conventional methods for cross-lingual entity alignment rely on machine translation and handcrafted features, which often need extra resources as input and are difficult to reuse. By contrast, embedding-based approaches are simple and easy to reuse. In recent years, many embedding-based methods have been proposed and achieved promising results in cross-lingual entity alignment.

Due to the simplicity and effectiveness of TransE, almost all of the existing cross-lingual entity alignment works employ TransE or its variants to encode knowledge. JE (Hao et al., 2016) employs a modified TransE to encode entities and relations. It bridges different KGs by adding a loss of alignment seeds in its objective function. MTransE (Chen et al., 2017) encodes entities and relations for each language-specific KG separately based on TransE, and then employs five strategies to learn cross-lingual transitions through pre-aligned triples. JAPE (Sun et al., 2017) learns structure embedding using a variant of TransE which adds weight to negative samples in its objective function. It also borrows the idea from Skip-gram model to learn attribute correlations and then leverages attribute correlations to refine structure embedding. To overcome the heterogeneity and language gap between different KGs, JAPE only considers the range types of attribute values, with the specific attribute values discarded.

There are also some efforts have been devoted to solving the problem of lacking enough training data. IPTransE (Zhu et al., 2017) and BootEA (Sun et al., 2018) propose to expand training set from alignment results iteratively. KDCoE (Chen et al., 2018) performs co-training of a KG embedding model and an entity description embedding model.

Besides, GCN-EA (Wang et al., 2018) employs GCNs to embed entities and attributes as low-dimensional vectors.

Different from the above methods, in this paper, we propose a new approach to jointly learn embeddings for both entities and relations from different KGs.

## 3 Problem Formulation

We begin with the problem definition. A knowledge graph (KG) usually contains a large number of triples in the form of $(h, r, t)$, where $r$ represents a relation between the head entity $h$ and the tail entity $t$. Formally, a KG can be represented as $KG = (E, R, T)$, where $E$, $R$, $T$ are the set of entities, relations and triples respectively.

Now suppose we have two heterogenous and language-specific KGs to be aligned, where $KG_1 = (E_1, R_1, T_1)$ and $KG_2 = (E_2, R_2, T_2)$. And in many cases, we already have a set of pre-aligned entities $S_e = \{(e_1, e_2)|e_1 \in E_1, e_2 \in E_2\}$ and relations $S_r = \{(r_1, r_2)|r_1 \in R_1, r_2 \in R_2\}$ as alignment seeds. The task of entity alignment is to automatically find new entity alignments based on the existing alignment seeds.

## 4 The Proposed Approach

Our approach consists of two steps. The first step is to learn embeddings for entities and relations, and we will introduce a new method to do this task. The second step is to align entities between the two input KGs based on the learned embeddings. In this section, we will introduce the two parts in details.

### 4.1 Multilingual KG Embedding

**DistMA**: The goal of KG embedding is to embed entities and relations into a unified vector space such that low-dimensional vectors of entities or relations can represent their semantics. A powerful KG embedding approach should embed similar entities or relations to have similar embeddings, such that the semantics of knowledge graphs can be well captured.

The most widely used approaches to encode entities and relations by previous works are

TransE (Bordes et al., 2013) and its variants. Given one triple $\tau = (h, r, t)$ in a knowledge graph, TransE and its variants define the energy function for this triple as $E(\tau) = \|\vec{v_h}+\vec{v_r}-\vec{v_t}\|_{1/2}$, where $\vec{v_h}, \vec{v_r}, \vec{v_t} \in \mathbb{R}^d$ represent the $d$-length vectors for $h$, $r$ and $t$ respectively. Then they optimize margin-based ranking losses so that the energy for positive triples is lower than negative ones. As discussed in Section 1, they don't do well in modeling multi-mapping relations, we need to explore new approaches to better capture the semantics of multi-mapping relations.

In this paper, we propose a new approach to encode both entities and relations in knowledge graphs. Our approach defines the energy function for a triple $\tau$ as:

$$E_1(h, r, t) = \langle \vec{v_h}, \vec{v_r} \rangle + \langle \vec{v_r}, \vec{v_t} \rangle + \langle \vec{v_h}, \vec{v_t} \rangle \quad (1)$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product.

Besides, instead of using a margin-based loss function, we employ the logistic loss with $L_2$ regularization on model parameters $\Theta$. The objective function is defined as:

$$\begin{aligned} O_1 = &- \sum_{(h,r,t)\in\mathbb{T}^+} \log \sigma(E_1(h, r, t)) \\ &- \sum_{(h',r,t')\in\mathbb{T}^-} \log \sigma(-1 \cdot E_1(h', r, t')) + \lambda\|\Theta\|_2^2 \end{aligned} \quad (2)$$

where $\sigma$ is the sigmoid function. $\mathbb{T}^+$ and $\mathbb{T}^-$ are positive and negative triples set, respectively. $\lambda$ is a hyper-parameter that controls the influence of regularization. The value of $\sigma(E(h, r, t))$ measures the plausibility of the triple $(h, r, t)$. This means our method expects positive triples to have high energy scores and negative ones to have low scores. Compared with TransE and its variants, dot products of embeddings scale well and can naturally handle both 1-1 and multi-mapping relations, which is helpful for capturing the semantics of KGs. We define the negative triples set $\mathbb{T}^-$ as:

$$\begin{aligned} \mathbb{T}^- = &\{(h', r, t)|h' \in E\} \cup \\ &\{(h, r, t')|t' \in E\}, (h, r, t) \in \mathbb{T}^+ \end{aligned} \quad (3)$$

where $E$ denotes the set of entities.

**ComplEx**: The energy function of DistMA doesn't distinguish between head entities and tail entities. This means DistMA can model symmetric relations like $similar\_to$, $has\_friend$ well but maybe unfriendly to some antisymmetric relations like $is\_father\_of$, $part\_of$. To improve the ability of our model to encode such relations, we utilize the ComplEx (Trouillon et al., 2016). Instead of real-valued vectors, ComplEx gives complex-valued vectors to entities and relations. That is, for each entity $e$ or relation $r$, let $\vec{w_e} \in \mathbb{C}^k$ or $\vec{w_r} \in \mathbb{C}^k$ be the corresponding complex-valued vector, i.e. $\vec{w_e} = Re(\vec{w_e}) + iIm(\vec{w_e})$ with $Re(\vec{w_e}) \in \mathbb{R}^k$ and $Im(\vec{w_e}) \in \mathbb{R}^k$ being the real and imaginary parts of $\vec{w_e}$, where $\mathbb{R}$, $\mathbb{C}$ represent the real and complex vector spaces respectively, and $i$ denotes the square root of -1. Then the energy for a triple $(h, r, t)$ is defined as follows:

$$E_2(h, r, t) = Re(\langle \vec{w_h}, \vec{w_r}, \vec{\bar{w}_t} \rangle) \quad (4)$$

where $\langle \cdot, \cdot, \cdot \rangle$ denotes the generalized inner product and $\vec{\bar{w}_t}$ represents the conjugate of $\vec{w_t}$: $\vec{\bar{w}_t} = Re(\vec{w_t}) - iIm(\vec{w_t})$. One can easily verify that Equation (4) can be expanded and written as:

$$\begin{aligned} E_2(h, r, t) = &\langle Re(\vec{w_h}), Re(\vec{w_r}), Re(\vec{w_t}) \rangle \\ &+ \langle Re(\vec{w_h}), Im(\vec{w_r}), Im(\vec{w_t}) \rangle \\ &+ \langle Im(\vec{w_h}), Re(\vec{w_r}), Im(\vec{w_t}) \rangle \\ &- \langle Im(\vec{w_h}), Im(\vec{w_r}), Re(\vec{w_t}) \rangle \end{aligned} \quad (5)$$

With this energy function, we use the same Equation (2) to calculate loss $O_2$ as $O_1$.

**Joint Embedding**: With ComplEx, we refine the final energy function for each triple as:

$$E(h, r, t) = E_1(h, r, t) + E_2(h, r, t) \quad (6)$$

Then the final loss $O$ to optimize can be defined by Equation (2) using Equation (6) as the energy function.

We will evaluate each part of our method in the experiments.

**Parameter Sharing**: The aforementioned embedding model is trained separately in each KG. As our final goal is to align entities between two KGs, we need to calibrate the embeddings of the two KGs into a unified embedding space based on the existing seed alignment. A natural and effective assumption is: the pre-aligned pair should share embedding to bridge different KGs. Under this assumption, the semantic loss between equivalent entity or relation pair is zero. This idea has also been mentioned in Sun et al. (2017) and Zhu et al. (2017).

### 4.2 Weighted Negative Sampling

Most previous works (Bordes et al., 2013; Sun et al., 2017; Zhu et al., 2017; Wang et al., 2018) use uniform sampling to generate negative triples $\mathbb{T}^-$, that is, replacing the head or tail entity of a positive triple with a random entity. Despite the simplicity of this method, it can be blind in many cases. For example, suppose we have a positive triple $(David\ Hilbert, nationality, Germany)$ and randomly replace the tail entity with an arbitrary entity. Maybe we can get one negative triple $(David\ Hilbert, nationality, Barack\ Obama)$. This is not a good choice because such a ridiculous triple contributes little to learning embeddings. Since such an obviously false triple is "too easy" and can easily obtain low plausibility during training. In contrast, a negative triple like $(David\ Hilbert, nationality, France)$ is more valuable, since $France$ has similar semantics with $Germany$, but $France$ can't replace $Germany$. Such a negative triple can help the model to learn the underlying semantics difference between entities rather than types solely.

We propose a weighted sampling strategy to generate valuable negative samples. Take the tail entity for example, given a positive triple $(h, r, t)$, we replace $t$ with $t'$ obtained from the following probability distribution:

$$P((h,r,t')|(h,r,t)) = \frac{exp(sim(t,t'))}{\sum_{e\in E} exp(sim(t,e))} \tag{7}$$

where $E$ denotes the entity set and $sim(\cdot,\cdot)$ calculates the similarity between two entities. Here we employ the cosine similarity measure. This means our sampling method is more likely to replace one entity with its similar entities. Since these entities have strong semantic correlations with the original entity but don't express the original entity, which helps the model learn to distinguish. Note that our method also generates a few easy examples, which are also useful for learning embeddings.

### 4.3 Bidirectional Alignment

We regard alignment prediction as a ranking problem. That is, suppose we need to align the entities in $KG_1$ to $KG_2$, for each entity $e_i$ in $KG_1$ , we rank each entity $e_j$ in $KG_2$ based on the similarity between $e_i$ and $e_j$. The similarity can be calculated via their embeddings, which we define by cosine similarity measure as follows:

$$sim(e_i, e_j) = \langle [\vec{v_{e_i}}]_1, [\vec{v_{e_j}}]_1 \rangle \tag{8}$$

where $[\cdot]_1$ denotes the normalized vector. Finally, we will get a similarity matrix $\mathbf{S_{12}}$ between the two KGs. Then we can get a ranking matrix $\mathbf{M_{12}}$ for $KG_1$ with $\mathbf{M_{12}}(i,j)$ denoting the index of $e_j$ in the ranking list of $e_i$.

One thing that hurts the alignment precision is the difference of knowledge distribution between different KGs. Considering the following fact during the entity alignment process. Suppose we select the first ranked entity $e_j$ in the ranking list of $e_i$ to be the counterpart of $e_i$. It is reasonable because, from the view of $e_i$, $e_j$ is the most similar entity in $KG_2$. However, this can be misleading sometimes. For example, from the view of $e_j$, $e_i$ may not rank first in the ranking list of $e_j$, and even rank at dozens. In such a case, we don't think $e_j$ is the desired candidate for $e_i$. This means, instead of single direction in previous works, that the alignment prediction process should consider both directions of KGs. We combine ranking matrices from both directions to accomplish this goal.

From both views of $KG_1$ and $KG_2$, we refine the ranking matrix $\mathbf{M_{12}}$ as:

$$\mathbf{M_{12}} = \mathbf{M_{12}} + \mathbf{M_{21}^T} \tag{9}$$

where $\mathbf{M_{21}^T}$ denotes the transpose of the ranking matrix of $KG_2$.

### 4.4 Implementation Details

We initialize all embeddings for both DistMA and ComplEx based on the uniform distribution. We set $d$ = 150 for DistMA and $k$ = 75 for ComplEx so that the length of embeddings in DistMA and ComplEx are all equal to 150. Before predicting, we concatenate the embeddings in DistMA and ComplEx such that each entity is represented as a 300-dimensional vector in the end. Besides, during training, we sample ten negative triples for each positive one. Since the weighted negative sampling is time-consuming, to speed it up, we calculate Equation (7) every five epochs. We use the self-adaptive optimization method Adam (Kingma and Ba, 2015) for all trainings and we implement our model with Tensorflow [1].

## 5 Evaluation

### 5.1 Datasets

We evaluate our proposed method on DBP15K datasets (Sun et al., 2017; Wang et al., 2018).

[1] https://www.tensorflow.org/

The datasets contain three cross-lingual real-world datasets: $\text{DBP}_{ZH-EN}$ (Chinese to English), $\text{DBP}_{FR-EN}$ (French to English), $\text{DBP}_{JA-EN}$ (Japanese to English). In which each was built with 15 thousand ILLs from multilingual versions of DBpedia. The average number of entities, relations and relational triples of the three datasets are 166,255/4,291/420,025 respectively.

### 5.2 Baseline Methods

We select four representative methods as baselines for cross-lingual entity alignment. These methods can be categorized as:

- Translation-based methods where relations are modeled as translation operators in the embedding space, including JE (Hao et al., 2016), MTransE (Chen et al., 2017) and JAPE (Sun et al., 2017). For MTransE, it develops five variants in its alignment model, where the fourth obtains the best results according to the experiments of its authors. Thus, we choose this variant to represent MTransE. For JAPE, it learns both structure and attribute embeddings. We report the results of its full model.

- To the best of our knowledge, GCN-EA (Wang et al., 2018) is the only one method employing non-translation-based embedding technique. GCN-EA uses GCNs as its encoding network to encode entities and attributes. We report the results of its full model.

### 5.3 Evaluation Metrics

By convention, we use *Hits@k* and *Mean Reciprocal Rank (MRR)* as our metrics. *Hits@k* measures the percentage of correctly aligned entities ranked in the top-k. Generally, *Hits@1* indicates precision. *MRR* is the average of the reciprocal ranks of all test instances. Higher *Hits@k* and *MRR* indicate better performance.

### 5.4 Model Selection

We select hyper-parameters with the following values: learning rate $\eta \in \{0.001, 0.003, 0.005, 0.01, 0.1\}$, $L_2$ regularization factor $\lambda \in \{1e{-}5, 1e{-}4, 5e{-}4, 1e{-}3\}$, batch size $B \in \{2000, 5000, 10000, 20000, 30000\}$. The optimal hyper-parameters are: $\eta = 0.005$, $\lambda = 1e{-}4$ and $B = 20000$. We use these values in all experiments. The total number of epochs is set to 3000, within which we test alignment every ten epochs and save embeddings when obtaining better *Hits@1* performances.

### 5.5 Results and Analysis

Following Sun et al. (2017) and Wang et al. (2018), we use 30% of the gold standards for training and the rest 70% for testing. The results of four compared baselines and our approach are shown in Table 1. Among the four baselines, JAPE and GCN-EA obtain better results, since semantic loss happens when learning the translations between embedding spaces in the other two methods. GCN-EA is the most powerful method. It doesn't encode relations but it achieves a better performance except for the dataset $\text{DBP15K}_{ZH-EN}$.

However, we can easily see our method significantly outperforms all the four baselines, especially with the metric *Hist@1*, which indicates our method can better capture the semantics of different KGs. As discussed in Section 1, baseline methods all obtain low performance. And the *Hits@3* results of our method are comparable to the *Hits@10* results of all the baselines. Besides, we observe that the gap between the results with metrics *Hits@1* and *Hits@10* of our method is much smaller than the other methods (average 20.24% *vs.* 29.34%), which means our method has stronger ability to distinguish between target counterparts and error ones which have strong semantic similarity with target counterparts.

We also find the results after applying bidirectional alignment have a large improvement compared with single directional alignment under all the metrics, especially for *Hits@1*. This is due to the fact that entity alignment is essentially a bidirectional selection process. Bidirectional selection can soften the impact of unbalanced data distribution between different KGs, making the alignment prediction more reasonable.

**Evaluation of Weighted Negative Sampling**: Here we come to evaluate the effectiveness of our proposed weighted negative sampling strategy. We compare with uniform sampling with the negative sampling rate $neg$ = 1 and $neg$ = 10 (1 and 10 negative triples per positive triple), respectively. Table 2 shows the testing results with metrics *Hits@1* and *MRR*.

We summarize Table 2 in three aspects. Firstly, we can see that both sampling methods obtain

| DBP15K$_{ZH-EN}$ | $ZH-EN$ | | | | $EN-ZH$ | | | |
|---|---|---|---|---|---|---|---|---|
| | Hits@1 | Hits@3 | Hits@10 | Hits@50 | Hits@1 | Hits@3 | Hits@10 | Hits@50 |
| *JE | 21.27 | - | 42.77 | 56.74 | 19.52 | - | 39.36 | 53.25 |
| *MTransE | 30.83 | - | 61.41 | 79.12 | 24.78 | - | 52.42 | 70.45 |
| *JAPE | 41.18 | - | **74.46** | **88.90** | **40.15** | - | **71.05** | **86.18** |
| †GCN-EA | **41.25** | - | 74.38 | 86.23 | 36.49 | - | 69.94 | 82.45 |
| Ours | 65.46 | 78.13 | 86.11 | 91.56 | 64.73 | 77.56 | 85.79 | 91.20 |
| Ours(*Bid.*) | **68.07** | **79.33** | **86.74** | **91.75** | **67.72** | **79.38** | **86.57** | **91.51** |
| DBP15K$_{JA-EN}$ | $JA-EN$ | | | | $EN-JA$ | | | |
| | Hits@1 | Hits@3 | Hits@10 | Hits@50 | Hits@1 | Hits@3 | Hits@10 | Hits@50 |
| *JE | 18.92 | - | 39.97 | 54.24 | 17.80 | - | 38.44 | 52.48 |
| *MTransE | 27.86 | - | 57.45 | 75.94 | 23.72 | - | 49.92 | 67.93 |
| *JAPE | 36.25 | - | 68.50 | 85.35 | 38.37 | - | 67.27 | 82.65 |
| †GCN-EA | **39.91** | - | **74.46** | **86.10** | **38.42** | - | **71.81** | **83.72** |
| Ours | 62.84 | 75.94 | 84.90 | 91.35 | 62.27 | 75.33 | 84.71 | 91.32 |
| Ours(*Bid.*) | **65.53** | **77.50** | **85.90** | **91.60** | **65.22** | **77.35** | **85.77** | **91.61** |
| DBP15K$_{FR-EN}$ | $FR-EN$ | | | | $EN-FR$ | | | |
| | Hits@1 | Hits@3 | Hits@10 | Hits@50 | Hits@1 | Hits@3 | Hits@10 | Hits@50 |
| *JE | 15.38 | - | 38.84 | 56.50 | 14.61 | - | 37.25 | 54.01 |
| *MTransE | 24.41 | - | 55.55 | 74.41 | 21.26 | - | 50.60 | 69.93 |
| *JAPE | 32.39 | - | 66.68 | 83.19 | 32.97 | - | 65.91 | 82.38 |
| †GCN-EA | **37.29** | - | **74.49** | **86.73** | **36.77** | - | **73.06** | **86.39** |
| Ours | 64.30 | 78.76 | 87.92 | 93.90 | 63.50 | 78.39 | 87.83 | 93.94 |
| Ours(*Bid.*) | **67.70** | **80.84** | **89.01** | **94.32** | **67.34** | **80.99** | **89.05** | **94.21** |

Table 1: Results comparison. (* marks the results copied from Sun et al. (2017). † marks the results reported by its authors. - marks the unreported results. *Bid.* indicates bidirectional alignment.)

better performance with the increase of the negative sampling rate. This shows that more negative samples can assist model to better capture the semantics of KGs. Secondly, we can find that, compared with uniform negative sampling, our weighted sampling strategy gets better performance for all the three datasets. This is because our sampling method can generate more valuable negative samples to assist model to learn correct embeddings. Besides, we observe our method is more effective with a smaller negative sampling rate ($neg$ = 1). Because when $neg$ is large, it is likely to generate a few valuable negative samples though using uniform sampling. But when $neg$ is small, the uniform sampling method can't generate enough valuable samples, which seriously hurts the final performance. However, our sampling strategy can pay more attention to valuable samples, generating many valuable negative samples even when the amount of total samples is small.

**Ablation Study**: For ablation study, we separate two variants from our approach. The first one only optimizes the objective function $O_1$, called DistMA. And the second one only optimizes $O_2$, called ComplEx. Table 3 shows the results of the two variants and the overall method.

As expected, Table 3 shows that each variant gets better results compared with four baselines. The DistMA can model symmetric relations well and the ComplEx can improve the ability in modeling antisymmetric relations. The overall method gets the best results due to its strong ability in modeling both relations.

**Training with Different Sizes of Seed Alignment**: We further investigate how the size of the training data affects the performance of our proposed approach. Here we use different proportions of seed alignment, which ranges from 20% to 50% with step 10%. And we choose two strong baselines JAPE and GCN-EA for comparison. Figure 1 illustrates the *Hits@1* results of the three approaches.

From Figure 1 we can see that the results for

| Variants | $ZH-EN$ | | $EN-ZH$ | | $JA-EN$ | | $EN-JA$ | | $FR-EN$ | | $EN-FR$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Hits@1 | MRR | Hits@1 | MRR | Hits@1 | MRR | Hits@1 | MRR | Hits@1 | MRR | Hits@1 | MRR |
| $neg$=1, *unif.* | 62.36 | 0.709 | 62.14 | 0.707 | 59.79 | 0.686 | 59.78 | 0.686 | 59.69 | 0.697 | 59.66 | 0.697 |
| $neg$=1, *weig.* | **63.85** | **0.716** | **63.96** | **0.716** | **62.23** | **0.701** | **61.91** | **0.698** | **61.17** | **0.705** | **61.10** | **0.704** |
| $neg$=10, *unif.* | 66.86 | 0.742 | 66.41 | 0.740 | 64.98 | 0.726 | 64.60 | 0.724 | 66.11 | 0.747 | 65.56 | 0.743 |
| $neg$=10, *weig.* | **68.07** | **0.748** | **67.72** | **0.746** | **65.53** | **0.727** | **65.22** | **0.725** | **67.70** | **0.755** | **67.34** | **0.752** |

Table 2: Evaluation of weighted negative sampling. (*unif.* indicates the uniform negative sampling and *weig.* indicates our weighted sampling strategy)

| Variants | $ZH-EN$ | | | $JA-EN$ | | | $FR-EN$ | | |
|---|---|---|---|---|---|---|---|---|---|
| | Hits@3 | Hits@10 | MRR | Hits@3 | Hits@10 | MRR | Hits@3 | Hits@10 | MRR |
| DistMA | 73.50 | **83.66** | 0.672 | **71.09** | **82.09** | 0.646 | **72.57** | **83.97** | **0.656** |
| ComplEx | **73.82** | 80.82 | **0.692** | 69.29 | 78.10 | **0.653** | 70.85 | 80.81 | 0.652 |
| Overall | **79.33** | **86.74** | **0.748** | **77.50** | **85.90** | **0.727** | **80.84** | **89.01** | **0.755** |

Table 3: Ablation study.

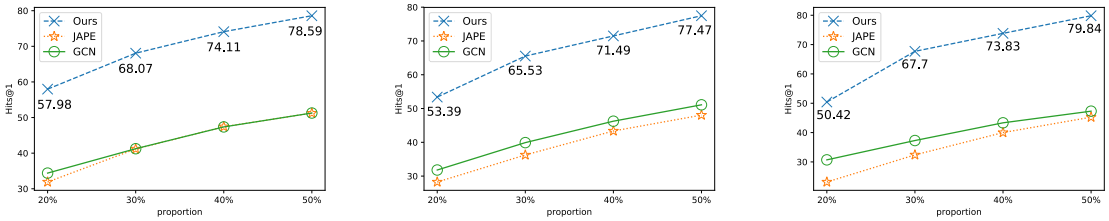


(a) $ZH-EN$ (b) $JA-EN$ (c) $FR-EN$

Figure 1: *Hits@1* w.r.t. proportion of seed alignment.

all the three approaches become better with the increase of proportion of training data, because more seed alignment can provide more information to bridge different KGs. And we can also see that our approach consistently outperforms the other two methods under different proportions. Moreover, only with 20% seed alignment, our approach achieves comparable performances for all the three datasets with the other two methods using 50% seed alignment. When using 30% seed alignment, our approach largely outperforms the other two methods using 50% seed alignment. All these results show the robustness and effectiveness of our method.

**Hits@1 Results by Mapping Properties of Relations**: To better know how effective our method is for different relations, compared with JAPE (Sun et al., 2017), we evaluate the improvement of *Hits@1* results by mapping properties of relations. To do this, we first divide all relations into 1-N, N-1, 1-1 and N-N relations [2]. Then we evaluate the *Hits@1* alignment results for head entities and tail entities of the four kinds of relations respectively.

The results are shown in Table 4. We can see that the result of N-N relations has the greatest improvement (with 111.4% average relative increase for head entities and 93.6% for tail entities). In addition, there are two points are also remarkable: the head entities of N-1 relations (96.2%) and the tail entities of 1-N relations (79.3%), which suggests the superiority of our method in modeling multi-mapping relations.

## 6 Conclusion and Future Work

This paper presents a simple and effective embedding framework for multilingual KGs which successfully improves the performance of cross-

[2]Following Wang et al. (2014), for each relation r, we compute the average number of tails per head ($tph_r$) and the average number of head per tail ($hpt_r$). If $tph_r \geq 1.5$ and $hpt_r < 1.5$, r is treated as 1-N; if $tph_r < 1.5$ and $hpt_r \geq 1.5$, r is treated as N-1; if $tph_r < 1.5$ and $hpt_r < 1.5$, r is treated as 1-1; if $tph_r \geq 1.5$ and $hpt_r \geq 1.5$, r is treated as a N-N.

| Position | Rel. | $ZH-EN$ | | | $JA-EN$ | | | $FR-EN$ | | | Improv.(Avg.) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | JAPE | Ours | Improv. | JAPE | Ours | Improv. | JAPE | Ours | Improv. | |
| Head | 1-N | 49.25 | 69.47 | 41.1% | 37.03 | 64.91 | 75.3% | 39.03 | 75.82 | 94.3% | 70.2% |
| | N-1 | 41.50 | 71.44 | 72.1% | 32.57 | 63.32 | 94.4% | 30.57 | 67.86 | 122.0% | 96.2% |
| | 1-1 | 46.77 | 70.67 | 51.1% | 42.15 | 69.36 | 64.6% | 36.19 | 72.72 | 100.9% | 72.2% |
| | N-N | 34.73 | 67.34 | 93.9% | 31.03 | 62.58 | 101.7% | 26.80 | 63.97 | 138.7% | **111.4%** |
| Tail | 1-N | 45.44 | 73.58 | 61.9% | 36.19 | 66.81 | 84.6% | 38.59 | 73.83 | 91.3% | 79.3% |
| | N-1 | 59.68 | 79.84 | 33.8% | 49.38 | 74.34 | 50.6% | 46.85 | 75.65 | 61.5% | 48.6% |
| | 1-1 | 46.92 | 72.44 | 54.4% | 47.40 | 72.77 | 53.5% | 36.17 | 73.62 | 103.5% | 70.5% |
| | N-N | 41.42 | 69.03 | 66.6% | 34.51 | 67.86 | 96.7% | 31.48 | 68.49 | 117.5% | **93.6%** |

Table 4: Comparison of JAPE and our model in *Hits@1* results by mapping properties of relations.

lingual entity alignment. We also propose a weighted sampling strategy to generate hard negative examples during training. Moreover, we propose to align entities from both directions. Experimental results on real-world datasets show that our method significantly outperforms four competitors, especially with the metric *Hits@1*.

Our proposed embedding framework can be easily applied to other tasks, i.e. link prediction and triple classification. And the idea of weighted negative sampling can be helpful for many AI and NLP tasks such as relation extraction and classification. For future work, we plan to explore more powerful KG embedding methods. And we also have the idea of using categorical attributes or hierarchical types to guide the negative sampling process.

## Acknowledgments

This work was supported by Big Data Application on lmproving Government Governance Capabilities National Engineering Laboratory Open Fund Project.

## References

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

Muhao Chen, Yingtao Tian, Kai-Wei Chang, Steven Skiena, and Carlo Zaniolo. 2018. Co-training embeddings of knowledge graphs and entity descriptions for cross-lingual entity alignment. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, pages 3998–4004. AAAI Press.

Muhao Chen, Yingtao Tian, Mohan Yang, and Carlo Zaniolo. 2017. Multilingual knowledge graph embeddings for cross-lingual knowledge alignment. In *IJCAI*, pages 1511–1517.

Takuma Ebisu and Ryutaro Ichise. 2018. Toruse: Knowledge graph embedding on a lie group. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Alberto García-Durán, Antoine Bordes, and Nicolas Usunier. 2015. Composing relationships with translations. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 286–290.

Yanchao Hao, Yuanzhe Zhang, Shizhu He, Kang Liu, and Jun Zhao. 2016. A joint embedding method for entity alignment of knowledge bases. In *China Conference on Knowledge Graph and Semantic Computing*, pages 3–14. Springer.

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 687–696.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *The 3rd International Conference on Learning Representations.*

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195.

Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015a. Modeling relation paths for representation learning of knowledge bases. In *Proceedings of EMNLP*, pages 705–714.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015b. Learning entity and relation embeddings for knowledge graph completion. In *Twenty-ninth AAAI conference on artificial intelligence*.

Farzaneh Mahdisoltani, Joanna Biega, and Fabian M Suchanek. 2013. Yago3: A knowledge base from multilingual wikipedias. In *CIDR*.

Roberto Navigli and Simone Paolo Ponzetto. 2012. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.

Zequn Sun, Wei Hu, and Chengkai Li. 2017. Cross-lingual entity alignment via joint attribute-preserving embedding. In *International Semantic Web Conference*, pages 628–644. Springer.

Zequn Sun, Wei Hu, Qingheng Zhang, and Yuzhong Qu. 2018. Bootstrapping entity alignment with knowledge graph embedding. In *IJCAI*, pages 4396–4402.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, pages 2071–2080.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Twenty-Eighth AAAI conference on artificial intelligence*.

Zhichun Wang, Qingsong Lv, Xiaohan Lan, and Yu Zhang. 2018. Cross-lingual knowledge graph alignment via graph convolutional networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 349–357.

Han Xiao, Minlie Huang, Lian Meng, and Xiaoyan Zhu. 2017. Ssp: semantic space projection for knowledge graph embedding with text descriptions. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation learning of knowledge graphs with entity descriptions. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Ruobing Xie, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2017. Image-embodied knowledge representation learning. In *IJCAI*.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *International Conference on Learning Representations*.

Hao Zhu, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2017. Iterative entity alignment via joint knowledge embeddings. In *IJCAI*, pages 4258–4264.

# Supervised and Nonlinear Alignment of Two Embedding Spaces for Dictionary Induction in Low Resourced Languages

**Masud Moshtaghi**
Amazon / Manhattan Beach, CA USA
mmasud@amazon.com

## Abstract

Enabling cross-lingual NLP tasks by leveraging multilingual word embedding has recently attracted much attention. An important motivation is to support lower resourced languages, however, most efforts focus on demonstrating the effectiveness of the techniques using embeddings derived from similar languages to English with large parallel content. In this study, we present a noise tolerant piecewise linear technique to learn a non-linear mapping between two monolingual word embedding vector spaces. We evaluate our approach on inferring bilingual dictionaries. We show that our technique outperforms the state of the art in lower resourced settings with an average improvement of 3.7% for precision @10 across 14 mostly low resourced languages.

## 1 Introduction

Aligning two embedding spaces is a key component in many multilingual tasks in natural language processing such as unsupervised word and sentence translation (Ruder et al., 2017) and large-scale dictionary creation (Lample et al., 2018). Large scale dictionaries are used in many NLP tasks, for example, web data cleaning. The main feature of the two most successful methods for this task is based on dictionaries (Xu and Koehn, 2017; Koehn et al., 2018).

Initial success obtained in aligning two embedding spaces has created a flurry of research efforts into improving the task (Nakashole, 2018; Doval et al., 2018; Kementchedjhieva et al., 2018; Chen and Cardie, 2018; Joulin et al., 2018; Hoshen and Wolf, 2018; Artetxe et al., 2018), to name a few published in 2018. There are two main approaches for mapping a monolingual word embedding space into another (Ruder et al., 2017): 1) Supervised methods rely on a few anchor points to learn the matching of two spaces (Lample et al., 2018; Mikolov et al., 2013; Xing et al., 2015; Joulin et al., 2018) and; 2) Unsupervised methods mostly formulate the problem as an adversarial optimization problem (Lample et al., 2018; Zhang et al., 2017; Ruder et al., 2017; Chen and Cardie, 2018; Hoshen and Wolf, 2018). While for high density languages such as Spanish, German, French and Italian the accuracy of unsupervised techniques is comparable, or even higher, than supervised techniques, the accuracy of unsupervised techniques is significantly lower when lower resourced languages are considered. A number of the experiments featured in the recent publications reflect this fact (Søgaard et al., 2018; Irvine and Callison-Burch, 2017).

To place this work in the context of recent advances in the field, we divide the aspects targeted by researchers to improve the precision with some examples in Fig. 1. While some of the improvements in optimization and space-adjustments can be applied to almost (any) model, our goal is to improve the supervised mapping in low resourced setting. Since the conception of linear mapping between two spaces, there are only very small minor improvement in the supervised mapping function itself and largest improvements are obtained by updating the word translation retrieval in the shared space and normalization steps (Artetxe et al., 2018). Therefore, we test our method against the most widely cited and tested method by (Lample et al., 2018), which culminates the previous efforts in the area of the supervised mapping (Mikolov et al., 2013; Xing et al., 2015; Lazaridou et al., 2015; Dinu et al., 2015; Smith et al., 2017; Artetxe et al., 2017). The main idea of these methods is to use Procrustes analysis to determine a linear mapping between two spaces using a few anchor points, i.e., a small dictionary. An iterative method is used to expand the dictionary with the most promising candidates and



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 823–832, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

re-learn the mapping (Artetxe et al., 2017; Lample et al., 2018). However, the main success of these methods was obtained on linguistically similar languages where a large volume of monolingual data from compatible sources is available. A recent study showed that a simple linear mapping is not enough for low resourced languages (Nakashole, 2018). However, the proposed method in this study, NorMA, came short in solving the problem showing significant gains in mapping precision for Russian-English and Chinese-English, compared not to the best model in (Lample et al., 2018), while underperforming in Spanish-English French-English and German-English. Due to the limited set of experiments and inconsistent results across multiple languages we defer comparison with this method until a more mature version is developed. Here, we only acknowledge that our method achieves 1.5% better precision@1 compared to NorMA's results for Chinese.

The contributions of this work are three-fold. First, we introduce a noise tolerant form of *Generalized Procrustes* (GP) which corrects for rotation, geometrical translation [1] and dilation. Second, we adapt an existing technique for piecewise linear regression to instead perform piecewise linear mapping. This allows subspaces to have different mapping functions. Finally, we conduct a large-scale evaluation and show that our model systematically outperforms the state of the art (Lample et al., 2018) in dictionary inference task with the average gain of 3.7% in precision @10 across 14 languages.

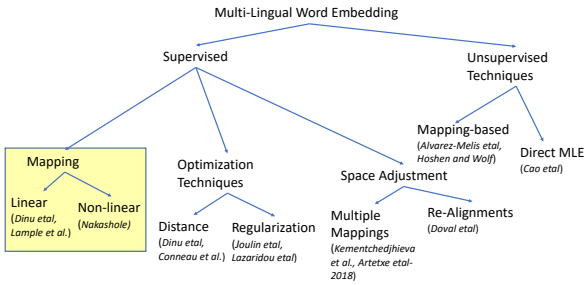


Figure 1: A breakdown of the main area of research focus for dictionary induction task.

[1] Note the difference between the translation of a word and translation as a geometrical transformation of a vector space.

## 2 Problem Statement

We first provide a formal definition of multilingual word embedding task using a mapping function, then we explain the intuition on why this method can be used to create dictionaries.

### 2.1 Definition

Given source and target embedding vector spaces $V_s$ and $V_o$, we are looking for a mapping $V_o \xrightarrow{f} V_s$ that minimize $\|f(V_o) - V_s\|_F$, where $\|.\|_F$ is the Frobenius norm (a matrix norm calculated by the square root of the sum of squares of its elements). Function $f$ performs two tasks, a permutation task to find the matching of vectors and a mapping task to map the permuted vectors in $V_o$ to space spanned by the vectors in $V_s$. In this general form, the problem is not convex. The supervised solutions for this problem add two assumptions to the problem to make it tractable. First, they assume a set of anchor points are available $\bar{V}_s \subseteq V_s$ and $\bar{V}_o \subseteq V_o$ for which we know a one-to-one matching, i.e., from a small dictionary. The second assumption is that the difference between the two spaces is only in rotation. In this paper, we relax the assumption about $f$ and propose a non-linear mapping, which in addition to the rotational difference, it corrects for dilation and translation between two embedding vector spaces.

### 2.2 Intuition

We first give an intuition about why we can map a word embedding space into another. Word2Vec (or similar) embedding processes capture the correlation between words (Mikolov et al., 2013) and embed the words in a vector space so that the distances between them are representative of this correlation. If the word embeddings are computed over comparable corpora, a word and its translation have similar correlated words. Hence, these words form similar neighborhood structures in their own monolingual spaces. A reasonable assumption is that the words in this neighborhood should also be a (dictionary) translation of each other. These two assumptions of the similarity of neighborhood structure and the assumption that the neighborhood consists of words that are translations of each other are crucial to the success of mapping methods. The similarity of the corpora used in the initial embeddings in terms of size and content significantly contributes to the validity of the assumptions. This effect is highlighted

in multiple studies (Lample et al., 2018; Søgaard et al., 2018; Grave et al., 2018). That is why the embeddings calculated on high resourced languages, with a similar parallel content to English, are mapped to English embedding space with very high accuracy as the embedding spaces have similar structures. Søgaard *et al.* highlight the language pair, the comparability of the monolingual corpora, and the parameters of the word embedding algorithms as key aspects affecting the performance of unsupervised methods. The performance of supervised methods is also tightly tied to these factors but they are less affected by the virtue of having the extra source of supervision.

If the initial two assumptions hold and since affine mappings, such as orthogonal rotation, do not change the structure of the space, minimizing the distance between the vectors should force the words with the same meaning to end up close to each other. In low resourced languages with a significantly lower amount of data compared to the English corpora structural similarities of the two spaces are not guaranteed leading to weak initial assumptions. This is also true in languages with lower similarity to English such as Slavic and Asian languages (Nakashole, 2018; Nakashole and Flauger, 2018). Nakashole and Flauger showed that word neighborhoods require different linear mappings to correctly capture the word relation. However, the proposed method by (Nakashole, 2018), NorMA, learns a dictionary of neighborhoods and reconstruct the word embedding using a linear combination of the dictionary items which does not guarantee locality in the mapping, , i.e., many dictionary items can contribute to representation of a word.

## 3 Proposed Method

Our proposed method tackles two problems with the existing mapping techniques. First, instead of only fixing the rotational differences between the two embedding vector spaces, the model also accounts for geometrical translation and dilation. Second, our model, which we call *LLMap*, has different linear mapping functions in subspaces which allows us to have different levels of corrections in different regions of the embedding space. We achieve this goal by incorporating a *Robust Generalized Procrustes* (RGP) mapping into a Locally Linear Model (LLNF).

### 3.1 Robust Generalized Procrustes Analysis

We borrow the concept Generalized Procrustes and introduce a noise tolerant version, which we name *Robust Generalized Procrustes* (RGP), to solve the first problem with the existing mapping techniques. RGP adds translation and dilation to the vanilla Procrustes. Note that Generalized Procrustes (GP) has been used to mean a different concept in (Kementchedjhieva et al., 2018) referring to a method that maps multiple spaces together.

#### 3.1.1 Procrustes Problem

Procrustes analysis has been used in many different fields for analyzing the relationship between two datasets. The *Orthogonal Procrustes Problem* (*OPP*) (Koschat and Swayne, 1991; Crosilla, 2003), aka Procrustes Rotation, used in current supervised multilingual mapping techniques (Lample et al., 2018; Smith et al., 2017) is defined as,

**Definition 1**: Given two matrices $\bar{V_s}$ and $\bar{V_o} \in \Re^{n \times p}$ with rank $p$, the OPP is defined as the following optimization problem.

$$\min_{W} \left\| \bar{V_o} W - \bar{V_s} \right\|_F^2, \text{subject to } W^T W = I_{p \times p} \tag{1}$$

The above optimization function has a closed form solution given by $W^T = UV^*$ where $UDV^* = SVD(\bar{V_s}^T \bar{V_o})$ . The Generalized Procrustes Problem (GPP) formulates the problem as a general linear mapping, correcting for other types of disturbances, i.e., translation and dilation.

**Definition 2**: Given two matrices $\bar{V_s}$ and $\bar{V_o} \in \Re^{n \times p}$ with rank $p$, the GPP is defined as the following optimization problem.

$$\begin{aligned} &\min_{W} \left\| \left( \bar{V_o} K W + I_{n \times 1} b \right) - \bar{V_s} \right\|_F^2, \\ &\qquad \text{subject to } W^T W = I_{p \times p} \\ &\textit{where } K \in \Re^{p \times p} (\textit{dilation}),\ b \in \Re^{1 \times p} (\textit{translation}), \\ &\qquad W \in \Re^{p \times p} (\textit{rotation}) \end{aligned} \tag{2}$$

The solutions for the three components of this optimization problem, i.e., $K$ (diagonal (zero off diagonal) positive matrix), $b$ and $W$ are shown in Eq. (3). Let $m_s = mean(\bar{V_s})$, $m_o = mean(\bar{V_o})$,

$\bar{V}_o^* = \bar{V}_o - m_o$ and $\bar{V}_s^* = \bar{V}_s - m_s$

$$
\begin{aligned}
W^T &= UV^*, UDV^* = SVD(K^{-1}\bar{V}_s^{*T}\bar{V}_o^*) \\
K_{jj} &= \frac{(\bar{V}_s^*)_j^T(\bar{V}_o^*W)_j}{(\bar{V}_o^*W)_j^T(\bar{V}_o^*W)_j} \\
b &= m_s - m_oKW
\end{aligned}
\tag{3}
$$

where $(.)_j$ denote the $j^{th}$ column vector of the matrix.

The translation factor $b$ in Eq. 3 does not appear in the calculation of rotation and dilation while the other two have interdependencies. Centering all vectors prior to mapping makes the translation vector zero. But it affects the flexibility of subspaces to have different translation factor. Hence, we operate on the original embedding space without centering the vectors.

For the other two factors, Koschar and Swayne (Koschat and Swayne, 1991) suggested using an iterative algorithm starting with $K = I_{p\times p}$ and computing $W$, then using $W$ updating the values of $K$ and iterate between the two until reaching convergence. We use 3 iterations in all of our experiments as the values converge rapidly. We expect to have noisy embedding vectors, so we use the truncated SVD algorithm to calculate the rotation matrix, i.e., discarding eigenvectors corresponding to small eigenvalues to make the mapping robust to outliers.

In the next section, we introduce how this linear mapping can be incorporated into the LLNF model to learn a piecewise linear mapping between two vector spaces.

### 3.2 Locally Linear Neural Model for Mapping (LLMap)

The inaccuracies in the embedding process due to lack of data, and inherent language family differences reduce structural similarities in (some) regions of the embedding space. In this case, a global minimization of the distance between the vectors using structure preserving mappings (Procrustes) could create undesired artifacts. We devise a piecewise linear mapping between the two spaces based on a Locally Linear Neural (LLNF) model with soft switching between the neurons (Babuška and Verbruggen, 2003) to increase the flexibility of the models when dealing with different levels of structural similarities in the embedding space. In this work, we update the neuron definition in the standard LLNF model to perform localized mapping using RGP, which guarantees a reduction in terms of the *Sum of Squared Errors* (SSE).

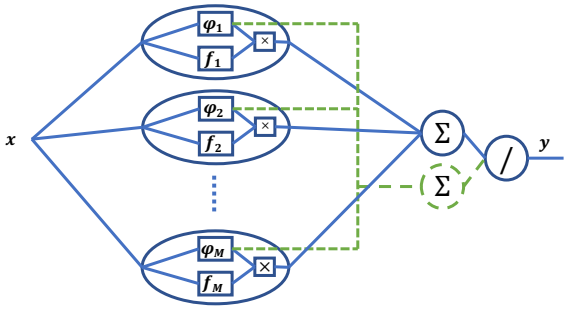


Figure 2: The network structure of an LLNF model. The green dotted line is used to normalize the output of neurons. Each neuron contains a membership function $\phi$ that controls the contribution of the neuron function $f$ to the final output.

First, we provide a brief introduction to the LLNF model followed by the description of our method, LLMap. LLMap replaces the linear regression function in the LLNF neurons with RGP while exploiting the training algorithm of the LLNF model for parameter estimation.

#### 3.2.1 Introduction to LLNF Models

The LLNF model is a specific form of Radial Basis Function Networks (RBF) with an expanded linear model in each neuron. This model is a general function approximator (Nelles, 2001). An LLNF input-output model with input vector $x \in \Re^p$, output $y \in \Re$ and $M$ neurons has the following form:

$$
y = \frac{\sum_{i=1}^{M} \phi_i(x) f_i(x)}{\sum_{i=1}^{M} \phi_i(x)} \tag{4}
$$

where $f : \Re^p \rightarrow \Re$ is a linear function and $\phi : \Re^p \rightarrow [0\ ,\ 1]$ is called a membership function. Fig. 2 shows a pictorial form of these networks. These networks typically contain one hidden layer. Each neuron has a membership function that controls the output of the neuron generated by a linear regression function inside the neuron. The final output is the weighted average of all the neurons.

The two most common membership functions are the Gaussian and trapezoidal functions. The Gaussian membership function with spherical contours

is defined as

$$\phi(x) = \prod_{j=1}^{p} e^{-\frac{(x_j - c_j)^2}{2\sigma_j^2}}, \tag{5}$$

where $c \in \Re^p$ and $\sigma \in \Re^p$ are the center and standard deviation of the neuron. When $p$ is large the Gaussian function can result in numerical instability and usually a trapezoidal function is preferred in this case. In one dimensional form a trapezoidal function is defined by a quadruple $(a, b, c, d)$ with the following function.

$$\phi(x_j)_j = \begin{cases} 0 & x_j < a \text{ or } x_j \geq d, \\ \frac{x_j - a_j}{b - a_j} & a \leq x_j < b, \\ 1 & b \leq x_j < c, \\ \frac{d - x_j}{d - c} & c \leq x_j < d. \end{cases} \tag{6}$$

An extension to a multidimensional function can be achieved by defining $\phi(x) = \min_j(\phi(x_j)_j)$. We use this membership function our model.

The non-linearity of the LLNF comes from $\phi_i$ functions. If these functions are predefined, the estimation of $f_i$ parameters is simplified to solving a weighted regression problem. The weight for input vector $x$ at neuron $q$ is $\phi_q(x)/\sum_{i=1}^{M}\phi_i(x)$, i.e., the normalized membership of the vector to the neuron. There are a number of strategies for determining the number of neurons and the $\phi_i$ functions. A widely used approach is called Locally Linear Model Tree (LoLiMoT), which follows a greedy tree construction algorithm.

**LoLiMoT**- The LoLiMoT algorithm (Nelles, 2001) creates hyper-rectangular partitions over the input space and centers a neuron in the middle of each partition. The parameters of the neuron are set proportional to the size of the rectangle. In Section 3.2.2, we describe how the parameters of the model are set in our experiments. The algorithm works in a greedy manner. It starts with one hyper-rectangle covering the whole space. Then, it enters a loop to optimize the number of partitions where it finds the rectangle that contributes the most to the estimation error and split the rectangle in the dimension that decreases the estimation error the most as shown in Algorithm 1. The splitting of each neuron is performed using axis-parallel partition of the space into two equal subspaces with a neuron at the center of each subspace. Fig. 3 shows a schematic 2-dimensional view of how the LoLiMoT algorithm partitions the input space.

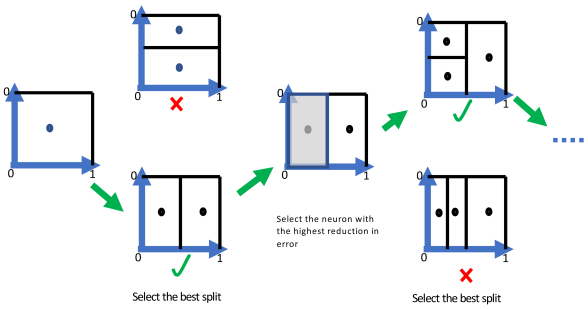


Figure 3: LoLiMoT structure optimization algorithm in 2 dimensions.

### 3.2.2 LLNF for Mapping

While LLNF models are used for regression and classification (Nelles, 2001), we aim to use these models for mapping by replacing the linear regression component in each neuron with an RGP. We leverage the LoLiMoT algorithm to optimize the network structure. This is achieved by replacing the function $f_i$ in Eq. 4 with a linear mapping function. This results in having a weighted RGP problem to estimate the parameters of $f_i$. The weighted RGP can also be solved in closed-form (Crosilla, 2003; Koschat and Swayne, 1991) by updating the Eq. 3 to include a diagonal weight matrix $L_q$ where $\ell_{xx} = \phi_q(x_x)/\sum_{i=1}^{M}\phi_i(x)$ is the weight for each input vector at neuron $q$. The updated formula is written as,

$$\begin{aligned} W_q^T &= UV^*, UDV^* = SVD(K^{-1}(L_q\bar{V}_s^{*T})(L_q\bar{V}_o^*)) \\ K_{jj} &= \frac{(\bar{V}_s^*)_j^T(\bar{V}_o^*W_q)_j}{(\bar{V}_o^*W_q)_j^T(\bar{V}_o^*W_q)} \\ b_q &= m_s - m_oKW_q \end{aligned} \tag{7}$$

The LLNF model in Eq. 4 computes a mapping of each input vector when Eq. 7 is used in place of $f_i$. With this change, the LoLiMoT algorithm can still be used to optimize the structure of the network.

In this paper, we consider a fixed setup in our experiments for certain hyperparameters of the LLMap to focus the discussion on the main model. These parameters can be potentially tuned to increase the performance. The following setup is used in all of the experiments in this paper.

1. Eq. 7 requires the number of iterations between calculation of rotation and dilation

**Input:** Matrices $\bar{V}_s$ and $\bar{V}_o$, $D$ = # of dimensions, $itr$ = # of splits
LLMap = `Initialize` (*One neuron covering the entire space*)
**while** $itr \geq 1$ **do**
  `// Use Eq.(7) for each neuron in LLMap`
  Error = LLMap.`Fit` ($\bar{V}_s$,$\bar{V}_o$)
  **foreach** $N \in LLMap$ **do**
    `// ` $L_q$ `is a diag. matrix of membership values`
    N.Error = Error*N.$L_q$
  **end**
  N_Sel = *N in LLMap with Maximum Error*
  LLMap_Sel = LLMap
  MaxError = Error.`Sum` ()
  **foreach** $j \in D$ **do**
    LLMap_New = LLMap - N_Sel
    N1, N2 = N_Sel.`Split` ($j$)
    LLMap_New.`Add` (*N1, N2*)
    Error = LLMap_New.`Fit` ($\bar{V}_s, \bar{V}_o$)
    **if** *Error.*`Sum` *()* $<$ *MaxError* **then**
      MaxError = Error.`Sum` ()
      LLMap_Sel = LLMap_New
    **end**
  **end**
  $itr --$
  LLMap = LLMap_Sel
**end**

**Algorithm 1:** LLMap Algorithm

matrices. We use 3 iterations in all experiments.

2. To define the quadruple $(a, b, c, d)$ for each neuron, we use two parameters $(\alpha, \delta)$ and define $a = \alpha - 4\delta, b = \alpha - \delta, c = \alpha + \delta, d = \alpha + 4\delta$. For the first neuron $\alpha = m_o$ (the mean of input vectors) and $\delta = (max\{\bar{V}_o\} - min\{\bar{V}_o\})/4$ . This ensures that the activation function of the first neuron is non-zero for all the values in the space.

3. We set the stopping criterion for the LoLiMoT algorithm to 4 neurons. We impose this restriction to allow the model and the data to fit in GPU memory. In addition, the small number of neurons prevents the vanilla LoLiMoT from overfitting and removing the need for a validation set.

4. To calculate the rotation parameter $W$, we discard eigenvectors in $U, V$ with corresponding eigenvalue smaller than 0.5 up to a maximum of 50 vectors. This maximum number is used to ensure that as the number of neurons increases LLMap does not remove too many eigenvectors.

## 4 Experiments

The purpose of the experiments is to compare the performance of our algorithm in low resourced languages for dictionary generation in a supervised setting with the best existing supervised technique. To this end, we compare our model with the supervised MUSE algorithm (Lample et al., 2018).

### 4.1 Data

We use the dictionaries made publicly available through the MUSE library [2]. This library contains 110 bilingual dictionaries of varied sized, mostly with English as the target languages. We also use the 300-dimensional word embeddings pre-trained on Wikipedia using the skip-gram method with default parameters described in (Bojanowski et al., 2017). [3] This library contains 294 languages.

We select 14 languages from low resourced languages including a wide range of languages including Asian and Slavic and European languages plus Spanish. Our selection mix is focused on low resourced and languages with low structural similarity to English. We use 200,000 most frequent tokens in the embedding file.

### 4.2 Experimental setup

The goal of dictionary inference using multilingual word embedding is to map the monolingual word embedding spaces onto each other and to infer the translation of a word by looking at the neighborhood of that word. We use the *Cross-Domain Similarity Local Scaling* (CSLS) distance introduced in (Lample et al., 2018) to find the top $k$. Given a source word, $v$, and its set of translations, $V'$, any sense retrieval measures the percentage of time that at least one $v' \in V'$ is in the top $k$-neighbors of $v$. In all senses retrieval the results

[2] https://github.com/facebookresearch/MUSE
[3] https://github.com/facebookresearch/fastText

| Language | P@1 | | | P@5 | | | P@10 | | |
|---|---|---|---|---|---|---|---|---|---|
| | LLMap | MUSE | RGP | LLMap | MUSE | RGP | LLMap | MUSE | RGP |
| Czech (CS) | 28.29 | 28.37 | 28.37 | **56.92** | 55.65 | 55.88 | **65.99** | 64.72 | 64.94 |
| Norwegian (NO) | **32.9** | 31.62 | 31.63 | **58.74** | 56.13 | 56.53 | **66.23** | 63.57 | 64.01 |
| Dutch (NL) | **42.3** | 41.06 | 41.18 | **67.13** | 65.43 | 65.7 | **73.73** | 72.03 | 72.49 |
| Chinese (ZH) | **17.4** | 14.19 | 19.51 | **43.19** | 35.6 | 44.05 | **52.11** | 44.62 | 52.68 |
| Korean (KO) | 17.12 | 17.02 | 16.81 | **35.8** | 34.41 | 34.23 | **44.05** | 42.69 | 42.12 |
| Japanese (JA) | **9.46** | 2.45 | 9.97 | **20.3** | 6.97 | 20.72 | **25.83** | 9.85 | 26.66 |
| Croatian (HR) | **19.29** | 18.71 | 18.87 | **46.4** | 43.55 | 44.25 | **56.36** | 53.38 | 54.12 |
| Indonesian (ID) | **31.9** | 30.37 | 30.45 | **54.63** | 51.96 | 52.4 | **62.06** | 59.24 | 59.67 |
| Farsi (FA) | **17.48** | 16.69 | 17.12 | **35.36** | 33.14 | 33.7 | **42.7** | 40.24 | 40.75 |
| Bulgarian (BG) | 21.64 | **23.4** | 23.02 | 56.19 | 55.22 | 55.35 | **67.12** | 65.95 | 66.11 |
| Spanish (ES) | 42.36 | 42.01 | 42.08 | 70.75 | 69.94 | 70.12 | 77.32 | 76.41 | 76.73 |
| Tamil (TA) | 9.79 | 9.76 | 10.07 | **24.3** | 22.34 | 23.08 | **31.38** | 28.78 | 29.8 |
| Hindi (HI) | **16.83** | 16.26 | 16.79 | **36.51** | 32.88 | 34.69 | **43.91** | 39.92 | 41.85 |
| Bengali (BN) | 15.51 | 15.3 | 15.95 | **39.67** | 35.55 | 37.06 | **48.43** | 43.94 | 45.51 |
| Average Improvement (LLMap-MUSE) | **1.07** | | | **3.36** | | | **3.70** | | |

Table 1: The comparison between the proposed methods LLMap and RGP, and the MUSE supervised method. The values are average precision over 10 random 90-10 splits of the dictionaries, statistically significant results between LLMap and MUSE are shown in bold and between LLMap and RGP are underlined.

are measured by the percentage of time each individual $v' \in V'$ appearing in the top $k$-neighbors of $v$.

While MUSE supervised was tested for any sense retrieval using a pre-split train-test, the performance of the dictionary inference task for $k > 1$ is better to be measured by all senses retrieval. Any sense retrieval is considered a significantly easier task, e.g. in Spanish precision@5 of the MUSE algorithm goes up from 71% to 91% when any sense is considered. Also, best experimental design practices suggest using cross validation to account for variation in the data.

Considering these shortcomings, we use a different experimental setup. We create 10 random 90-10 splits of the big dictionaries for training and testing of the systems to capture the variations due to random splits. We also report the precision@5 using the pre-split for compatibility with previously published results on this dataset for both any and all senses of a word.

We use double sided paired t-test and p-value at 0.05 to test for statistical significance in the results. We used 4 refinement steps in the supervised MUSE mapping and used the best result on the test data as the performance of the algorithm. For our algorithm, we report the final test result after adding the 4th neuron. We compare the results of RGP and LLMap to quantify the contribution of each addition that we made to improve the mapping.

### 4.3 Results

Table 1 shows the average of precision over the 10 random splits $@k = 1, 5$ and 10. The bold values are statistically significant results between LLMap and the MUSE supervised method. The RGP column refers to our model without the piecewise mapping, which we discuss later in this section.

In all cases, except Czech (CS) and Bulgarian (BG) @1 where MUSE has a slight edge over LLMap, our method achieved higher precision on average over 10-fold cross-validation than the MUSE algorithm. In the majority of the cases the improvements are statistically significant. We can see the most significant improvements (over 8%) are observed in Japanese (JA) language and Chinese (ZH). The other languages mostly see between 1%-3% improvement in the precision. The average gain in precision @10 sits at 3.7%.

The gains achieved by our method become larger by increasing the $k$. It shows that LLMap can create a better proximity neighborhood around the words as it is more flexible to handle multiple

| Lang | Any Sense | | All Senses | |
|---|---|---|---|---|
| | MUSE | LLMap | MUSE | LLMap |
| CS | 76.66 | **78.86** | 61.32 | **62.17** |
| NO | 80.00 | **81.85** | 65.57 | **67.35** |
| NL | 88.53 | **89.33** | 69.28 | **70.52** |
| ZH | 55.00 | **64.77** | 41.88 | **51.50** |
| KO | 51.94 | **54.23** | 42.81 | **44.40** |
| JA | 3.97 | **17.62** | 3.22 | **14.51** |
| HR | 62.46 | **64.70** | 52.19 | **53.78** |
| ID | 81.20 | **85.19** | 64.87 | **68.20** |
| FA | 53.73 | **56.03** | 42.64 | **44.29** |
| BG | 65.07 | **68.13** | 59.45 | **59.48** |
| ES | 92.10 | **92.23** | 71.46 | **72.11** |
| TA | 28.53 | **29.00** | 25.60 | **26.01** |
| HI | 51.20 | **53.93** | 38.22 | **44.53** |
| BN | 33.80 | **36.42** | 42.42 | **45.10** |
| AVG | **3.45** | | **3.07** | |

Table 2: The comparison between the proposed method LLMap and MUSE on pre-split train and test dictionaries for any and all senses recovery by the algorithms for precision@5. Last row shows the average improvement achieved by LLMap over MUSE

senses of a word. In Spanish the gains are not statistically significant at any $k$, pointing to the fact that in high resourced languages with large similarities with English the mapping cannot create a significant advantage over simpler mappings.

Table 2 shows the precision@5 for the pre-split dictionaries. In all 14 languages the LLMap outperforms the MUSE algorithm for recovering both all senses and any sense of a word with significant gains in Chinese and Japanese. These results are consistent with the more comprehensive cross-validation settings. Note that the any sense recovery is on average higher than all senses, pointing to the same fact the model is better at creating a better neighborhood around words where at least one sense of a word can be recovered.

The improvements that we observed compared to the state of the art are the culmination of two main changes to the baseline OPP in the mapping calculation, namely using RGP and the piecewise linear mapping. A vanilla RGP method is an LLMap model with one neuron. Table 1 shows how much increasing the non-linearity in the mapping by increasing the number of neurons from 1 to 4 contributes to the overall precision. LLMap has statistically significant (underlined values) higher precision than the vanilla RGP in most cases. However, in the two languages with the largest improvements compared to MUSE, RGP accounts for most of the gains. LLMap contributes to an overall average of 1% increase in the precision on top of RGP.

### 4.4 Discussions

The computational complexity of the mapping increases linearly with the number of neurons. In terms of training, finding the best dimension to split is the most time-consuming part, which increases the complexity of the algorithm from $O(n^2p)$ in OPP to $O(n^2p^2)$ in LLMap. This step requires creating $p$ (number of dimensions) models and estimate the error of each model. In our implementation, each split took approximately 5 minutes to complete on an Amazon AWS EC2 p3.2xlarge. This task is usually done offline as a pre-processing step. Therefore, the training time complexity would not be a barrier to this approach. While MUSE uses a simpler mapping, the iterations require identifying promising candidates to add to the dictionary, which results in a similar 5 minutes run-time on a GPU.

We limit the number of neurons in our algorithm to 4 for experimental purposes. This allows us to have the models, data and top-k calculation on a single GPU for quick (10-15 minutes) run-time.

The framework presented in this work, is a general framework that can accommodate other mappings and improvements to the distance functions. However, so far to the best of our knowledge, structure preserving mapping such GP outperforms non-linear mappings that change the structure of the underlying space. Regularized and weighted mapping functions that do not have closed-form solutions can also be incorporated in this structure with a back-propagation optimization technique. In this work, we focused our discussion on the generalized mapping function as opposed to other areas of improving this task.

### 4.5 Comments

We have added this section to explain the The reviewers requested comparison of LLMap with a stronger baseline to compare the supervised mapping, suggesting vanilla VecMap (Artetxe et al., 2018) as a potential baseline. In the introduction, we argued that in terms of mapping function MUSE has the strongest and most widely applicable mapping function. Most of improvements after this work, including the ones in VecMap,

are obtained by updating the translation retrieval and/or normalization around the mapping function. To ensure the validity of our argument, we ran VecMap with normalization, whitening and reweighting turned off and it produced nearly identical numbers to MUSE when CSLS is used as the retrieval method.

## 5 Conclusion

In this research, we introduced a non-linear mapping between two embedding spaces for dictionary induction. The embedding space created for low resourced languages are noisier than their English counterpart and introduce challenges for simple rotational mapping used in the current techniques. We showed that our piecewise linear model systematically improves the precision of existing Procrustian mapping for dictionary induction using in a large-scale cross-validation setting by over 3% on average. This approach has applications beyond the scope of this paper and can be used in any applications requiring solving OPP or GPP.

## References


Mikel Artetxe, Gorka Labaka, , and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *55th Annual Meeting of the Association for Computational Linguistics*, pages 451–462, Vancouver, Canada.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018. Generalizing and improving bilingual word embedding mappings with a multi-step framework of linear transformations. In *AAAI Conference on Artificial Intelligence*.

Robert Babuška and Henk Verbruggen. 2003. Neuro-fuzzy methods for nonlinear system identification. *Annual Reviews in Control*, 27(1):73 – 85.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Xilun Chen and Claire Cardie. 2018. Unsupervised multilingual word embeddings. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 261–270, Brussels, Belgium.

Fabio Crosilla. 2003. Procrustes analysis and geodetic sciences. In Schwarze V.S. Grafarend E.W., Krumm F.W., editor, *Geodesy-The Challenge of the 3rd Millennium*, pages 287–292. Springer, Berlin, Heidelberg.

Georgiana Dinu, Angeliki Lazaridou, and Marco Baroni. 2015. Improving zero-shot learning by mitigating the hubness problem. In *Proceedings of ICLR*, San Diego, CA.

Yerai Doval, Jose Camacho-Collados, Luis Espinosa Anke, and Steven Schockaert. 2018. Improving cross-lingual word embeddings by meeting in the middle. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 294–304, Brussels, Belgium.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Language Resources and Evaluation Conference*.

Yedid Hoshen and Lior Wolf. 2018. Non-adversarial unsupervised word translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 469–478, Brussels, Belgium. Association for Computational Linguistics.

Ann Irvine and Chris Callison-Burch. 2017. A comprehensive analysis of bilingual lexicon induction. *Computational Linguistics*, 43(2):273–310.

Armand Joulin, Piotr Bojanowski, Tomas Mikolov, Hervé Jégou, and Edouard Grave. 2018. Loss in translation: Learning bilingual word mapping with a retrieval criterion. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2979–2984, Brussels, Belgium. Association for Computational Linguistics.

Yova Kementchedjhieva, Sebastian Ruder, Ryan Cotterell, and Anders Søgaard. 2018. Generalizing Procrustes analysis for better bilingual dictionary induction. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 211–220, Brussels, Belgium. Association for Computational Linguistics.

Philipp Koehn, Huda Khayrallah, Kenneth Heafield, and Mikel Forcada. 2018. Findings of the wmt 2018 shared task on parallel corpus filtering. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 726–739, Belgium, Brussels. Association for Computational Linguistics.

Martin A. Koschat and Deborah F. Swayne. 1991. A weighted procrustes criterion. *Psychometrika*, 56(2):229–239.

Guillaume Lample, Alexis Conneau, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. In *Sixth International Conference on Learning Representations*.

Angeliki Lazaridou, Georgiana Dinu, and Marco Baroni. 2015. Hubness and pollution: Delving into

cross-space mapping for zero-shot learning. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural*, pages 270–280.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119.

Ndapa Nakashole. 2018. NORMA: Neighborhood sensitive maps for multilingual word embeddings. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 512–522, Brussels, Belgium. Association for Computational Linguistics.

Ndapa Nakashole and Raphael Flauger. 2018. Characterizing departures from linearity in word translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, page 221–227, Melbourne, Australia. Association for Computational Linguistics.

Oliver Nelles. 2001. *Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models.* Springer, Berlin, Heidelberg.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2017. A survey of cross-lingual word embedding models. *arXiv preprint arXiv:1706.04902v2*.

Samuel L. Smith, David H. P. Turban, Steven Hamblin, and Nils Y. Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In *Proceedings of International Conference on Learning Representations (ICLR)*, Toulon, France.

Anders Søgaard, Sebastian Ruder, and Ivan Vulić. 2018. On the limitations of unsupervised bilingual dictionary induction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 778–788, Melbourne, Australia. Association for Computational Linguistics.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of NAACL*, Los Angeles, CA.

Hainan Xu and Philipp Koehn. 2017. Zipporah: a fast and scalable data cleaning system for noisy web-crawled parallel corpora. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2945–2950, Copenhagen, Denmark. Association for Computational Linguistics.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Adversarial training for unsupervised bilingual lexicon induction. In *the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1959–1970, Vancouver, Canada.

# Beto, Bentz, Becas: The Surprising Cross-Lingual Effectiveness of BERT

**Shijie Wu** and **Mark Dredze**
Department of Computer Science
Johns Hopkins University
shijie.wu@jhu.edu, mdredze@cs.jhu.edu

## Abstract

Pretrained contextual representation models (Peters et al., 2018; Devlin et al., 2019) have pushed forward the state-of-the-art on many NLP tasks. A new release of BERT (Devlin, 2018) includes a model simultaneously pretrained on 104 languages with impressive performance for zero-shot cross-lingual transfer on a natural language inference task. This paper explores the broader cross-lingual potential of mBERT (multilingual) as a zero-shot language transfer model on 5 NLP tasks covering a total of 39 languages from various language families: NLI, document classification, NER, POS tagging, and dependency parsing. We compare mBERT with the best-published methods for zero-shot cross-lingual transfer and find mBERT competitive on each task. Additionally, we investigate the most effective strategy for utilizing mBERT in this manner, determine to what extent mBERT generalizes away from language-specific features, and measure factors that influence cross-lingual transfer.

## 1 Introduction

Pretrained language representations with self-supervised objectives have become standard in a variety of NLP tasks (Peters et al., 2018; Howard and Ruder, 2018; Radford et al., 2018; Devlin et al., 2019), including sentence-level classification (Wang et al., 2018), sequence tagging (e.g. NER) (Tjong Kim Sang and De Meulder, 2003) and SQuAD question answering (Rajpurkar et al., 2016). Self-supervised objectives include language modeling, the cloze task (Taylor, 1953) and next sentence classification. These objectives continue key ideas in word embedding objectives like CBOW and skip-gram (Mikolov et al., 2013a).

At the same time, cross-lingual embedding models have reduced the amount of cross-lingual supervision required to produce reasonable models; Conneau et al. (2017); Artetxe et al. (2018) use identical strings between languages as a pseudo bilingual dictionary to learn a mapping between monolingual-trained embeddings. Can jointly training contextual embedding models over multiple languages without explicit mappings produce an effective cross-lingual representation? Surprisingly, the answer is (partially) yes. BERT, a recently introduced pretrained model (Devlin et al., 2019), offers a multilingual model (mBERT) pretrained on concatenated Wikipedia data for 104 languages *without any cross-lingual alignment* (Devlin, 2018). mBERT does surprisingly well compared to cross-lingual word embeddings on zero-shot cross-lingual transfer in XNLI (Conneau et al., 2018), a natural language inference dataset. **Zero-shot cross-lingual transfer**, also known as single-source transfer, refers *trains and selects* a model in a source language, often a high resource language, then transfers directly to a target language.

While XNLI results are promising, the question remains: does mBERT learn a cross-lingual space that supports zero-shot transfer? We evaluate mBERT as a zero-shot cross-lingual transfer model on five different NLP tasks: natural language inference, document classification, named entity recognition, part-of-speech tagging, and dependency parsing. We show that it achieves competitive or even state-of-the-art performance with the recommended fine-tune all parameters scheme (Devlin et al., 2019). Additionally, we explore different fine-tuning and feature extraction schemes and demonstrate that with parameter freezing, we further outperform the suggested fine-tune all approach. Furthermore, we explore the extent to which mBERT generalizes away from a specific language by measuring accuracy on language ID

Code is available at https://github.com/shijie-wu/crosslingual-nlp



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 833–844, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

using each layer of mBERT. Finally, we show how subword tokenization influences transfer by measuring subword overlap between languages.

## 2 Background

**(Zero-shot) Cross-lingual Transfer** Cross-lingual transfer learning is a type of transductive transfer learning with different source and target domain (Pan and Yang, 2010). A cross-lingual representation space is assumed to perform the cross-lingual transfer. Before the widespread use of cross-lingual word embeddings, task-specific models assumed coarse-grain representation like part-of-speech tags, in support of a delexicalized parser (Zeman and Resnik, 2008). More recently cross-lingual word embeddings have been used in conjunction with task-specific neural architectures for tasks like named entity recognition (Xie et al., 2018), part-of-speech tagging (Kim et al., 2017) and dependency parsing (Ahmad et al., 2019).

**Cross-lingual Word Embeddings.** The quality of the cross-lingual space is essential for zero-shot cross-lingual transfer. Ruder et al. (2017) surveys methods for learning cross-lingual word embeddings by either joint training or post-training mappings of monolingual embeddings. Conneau et al. (2017) and Artetxe et al. (2018) first show two monolingual embeddings can be aligned by learning an orthogonal mapping with only identical strings as an initial heuristic bilingual dictionary.

**Contextual Word Embeddings** ELMo (Peters et al., 2018), a deep LSTM (Hochreiter and Schmidhuber, 1997) pretrained with a language modeling objective, learns contextual word embeddings. This contextualized representation outperforms standalone word embeddings, e.g. Word2Vec (Mikolov et al., 2013b) and Glove (Pennington et al., 2014), with the same task-specific architecture in various downstream tasks. Instead of taking the representation from a pretrained model, GPT (Radford et al., 2018) and Howard and Ruder (2018) also fine-tune all the parameters of the pretrained model for a specific task. Also, GPT uses a transformer encoder (Vaswani et al., 2017) instead of an LSTM and jointly fine-tunes with the language modeling objective. Howard and Ruder (2018) propose another fine-tuning strategy by using a different learning rate for each layer with learning rate warmup and gradual unfreezing.

Concurrent work by Lample and Conneau (2019) incorporates bitext into BERT by training on pairs of parallel sentences. Schuster et al. (2019) aligns pretrained ELMo of different languages by learning an orthogonal mapping and shows strong zero-shot and few-shot cross-lingual transfer performance on dependency parsing with 5 Indo-European languages. Similar to multilingual BERT, Mulcaire et al. (2019) trains a single ELMo on distantly related languages and shows mixed results as to the benefit of pretaining.

Parallel to our work, Pires et al. (2019) shows mBERT has good zero-shot cross-lingual transfer performance on NER and POS tagging. They show how subword overlap and word ordering effect mBERT transfer performance. Additionally, they show mBERT can find translation pairs and works on code-switched POS tagging. In comparison, our work looks at a larger set of NLP tasks including dependency parsing and ground the mBERT performance against previous state-of-the-art on zero-shot cross-lingual transfer. We also probe mBERT in different ways and show a more complete picture of the cross-lingual effectiveness of mBERT.

## 3 Multilingual BERT

**BERT** (Devlin et al., 2019) is a deep contextual representation based on a series of transformers trained by a self-supervised objective. One of the main differences between BERT and related work like ELMo and GPT is that BERT is trained by the Cloze task (Taylor, 1953), also referred to as masked language modeling, instead of right-to-left or left-to-right language modeling. This allows the model to freely encode information from both directions in each layer. Additionally, BERT also optimizes a next sentence classification objective. At training time, 50% of the paired sentences are consecutive sentences while the rest of the sentences are paired randomly. Instead of operating on words, BERT uses a subword vocabulary with WordPiece (Wu et al., 2016), a data-driven approach to break up a word into subwords.

**Fine-tuning BERT** BERT shows strong performance by fine-tuning the transformer encoder followed by a softmax classification layer on various sentence classification tasks. A sequence of shared softmax classifications produces sequence tagging models for tasks like NER. Fine-tuning usually takes 3 to 4 epochs with a relatively small learning rate, for example, 3e-5.

**Multilingual BERT** mBERT (Devlin, 2018) follows the same model architecture and training procedure as BERT, except with data from Wikipedia in 104 languages. Training makes no use of explicit cross-lingual signal, e.g. pairs of words, sentences or documents linked across languages. In mBERT, the WordPiece modeling strategy allows the model to share embeddings across languages. For example, "DNA" has a similar meaning even in distantly related languages like English and Chinese [1]. To account for varying sizes of Wikipedia training data in different languages, training uses a heuristic to subsample or oversample words when running WordPiece as well as sampling a training batch, random words for cloze and random sentences for next sentence classification.

**Transformer** For completeness, we describe the Transformer used by BERT. Let $\mathbf{x}, \mathbf{y}$ be a sequence of subwords from a sentence pair. A special token `[CLS]` is prepended to $\mathbf{x}$ and `[SEP]` is appended to both $\mathbf{x}$ and $\mathbf{y}$. The embedding is obtained by

$$\begin{aligned}\hat{h}_i^0 &= E(x_i) + E(i) + E(\mathbb{1}_{\mathbf{x}}) \\ \hat{h}_{j+|\mathbf{x}|}^0 &= E(y_j) + E(j+|\mathbf{x}|) + E(\mathbb{1}_{\mathbf{y}}) \\ h_{\cdot}^0 &= \text{Dropout}(\text{LN}(\hat{h}_{\cdot}^0))\end{aligned}$$

where $E$ is the embedding function and LN is layer normalization (Ba et al., 2016). $M$ transformer blocks are followed by the embeddings. In each transformer block,

$$\begin{aligned}h_{\cdot}^{i+1} &= \text{Skip}(\text{FF}, \text{Skip}(\text{MHSA}, h_{\cdot}^i)) \\ \text{Skip}(f, h) &= \text{LN}(h + \text{Dropout}(f(h))) \\ \text{FF}(h) &= \text{GELU}(h\mathbf{W}_1^\top + \mathbf{b}_1)\mathbf{W}_2^\top + \mathbf{b}_2\end{aligned}$$

where GELU is an element-wise activation function (Hendrycks and Gimpel, 2016). In practice, $h^i \in \mathbb{R}^{(|\mathbf{x}|+|\mathbf{y}|)\times d_h}$, $\mathbf{W}_1 \in \mathbb{R}^{4d_h \times d_h}$, $\mathbf{b}_1 \in \mathbb{R}^{4d_h}$, $\mathbf{W}_2 \in \mathbb{R}^{d_h \times 4d_h}$, and $\mathbf{b}_2 \in \mathbb{R}^{d_h}$. MHSA is the multi-heads self-attention function. We show how one new position $\hat{h}_i$ is computed.

$$\begin{aligned}[\cdots, \hat{h}_i, \cdots] &= \text{MHSA}([h_1, \cdots, h_{|\mathbf{x}|+|\mathbf{y}|}]) \\ &= \mathbf{W}_o \text{Concat}(h_i^1, \cdots, h_i^N) + \mathbf{b}_o\end{aligned}$$

In each attention, referred to as attention head,

$$\begin{aligned}h_i^j &= \sum_{k=1}^{|\mathbf{x}|+|\mathbf{y}|} \text{Dropout}(\alpha_k^{(i,j)})\mathbf{W}_V^j h_k \\ \alpha_k^{(i,j)} &= \frac{\exp \frac{(\mathbf{W}_Q^j h_i)^\top \mathbf{W}_K^j h_k}{\sqrt{d_h/N}}}{\sum_{k'=1}^{|\mathbf{x}|+|\mathbf{y}|} \exp \frac{(\mathbf{W}_Q^j h_i)^\top \mathbf{W}_K^j h_{k'}}{\sqrt{d_h/N}}}\end{aligned}$$

where $N$ is the number of attention heads, $h_i^j \in \mathbb{R}^{d_h/N}$, $\mathbf{W}_o \in \mathbb{R}^{d_h \times d_h}$, $\mathbf{b}_o \in \mathbb{R}^{d_h}$, and $\mathbf{W}_Q^j, \mathbf{W}_K^j, \mathbf{W}_V^j \in \mathbb{R}^{d_h/N \times d_h}$.

## 4 Tasks

Does mBERT learn a cross-lingual representation, or does it produce a representation for each language in its own embedding space? We consider five tasks in the zero-shot transfer setting. We assume labeled training data for each task in English, and transfer the trained model to a target language. We select a range of different tasks: document classification, natural language inference, named entity recognition, part-of-speech tagging, and dependency parsing. We cover zero-shot transfer from English to 38 languages in the 5 different tasks as shown in Tab. 1. In this section, we describe the tasks as well as task-specific layers.

### 4.1 Document Classification

We use MLDoc (Schwenk and Li, 2018), a balanced subset of the Reuters corpus covering 8 languages for document classification. The 4-way topic classification task decides between CCAT (Corporate/Industrial), ECAT (Economics), GCAT (Government/Social), and MCAT (Markets). We only use the first two sentences[2] of a document for classification due to memory constraint. The sentence pairs are provided to the mBERT encoder. The task-specific classification layer is a linear function mapping $h_0^{12} \in \mathbb{R}_h^d$ into $\mathbb{R}^4$, and a softmax is used to get class distribution. We evaluate by classification accuracy.

### 4.2 Natural Language Inference

We use XNLI (Conneau et al., 2018) which cover 15 languages for natural language inference. The 3-way classification includes entailment, neutral, and contradiction given a pair of sentences. We

[1] "DNA" indeed appears in the vocabulary of mBERT as a stand-alone lexicon.

[2] We only use the first sentence if the document only contains one sentence. Documents are segmented into sentences with NLTK (Perkins, 2014).

| | ar | bg | ca | cs | da | de | el | en | es | et | fa | fi | fr | he | hi | hr | hu | id | it | ja | ko | la | lv | nl | no | pl | pt | ro | ru | sk | sl | sv | sw | th | tr | uk | ur | vi | zh |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MLDoc | | | | | | ✓ | | ✓ | ✓ | | | | ✓ | | | | | | ✓ | ✓ | | | | | | | | | ✓ | | | | | | | | | | ✓ |
| NLI | ✓ | ✓ | | | | ✓ | ✓ | ✓ | ✓ | | | | ✓ | | ✓ | | | | | | | | | | | | | | ✓ | | | | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ |
| NER | | | | | | ✓ | | ✓ | ✓ | | | | | | | | | | | | | | | ✓ | | | | | | | | | | | | | | | ✓ |
| POS | | ✓ | | | ✓ | ✓ | | ✓ | ✓ | | ✓ | | | | | | ✓ | | ✓ | | | | | ✓ | | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ | | | | | | | |
| Parsing | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | | | ✓ | | | ✓ |

Table 1: The 39 languages used in the 5 tasks.

feed a pair of sentences directly into mBERT and the task-specific classification layer is the same as §4.1. We evaluate by classification accuracy.

### 4.3 Named Entity Recognition

We use the CoNLL 2002 and 2003 NER shared tasks (Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003) (4 languages) and a Chinese NER dataset (Levow, 2006). The labeling scheme is BIO with 4 types of named entities. We add a linear classification layer with softmax to obtain word-level predictions. Since mBERT operates at the subword-level while the labeling is word-level, if a word is broken into multiple subwords, we mask the prediction of non-first subwords. NER is evaluated by F1 of predicted entity (F1). Note we use a simple post-processing heuristic to obtain a valid span.

### 4.4 Part-of-Speech Tagging

We use a subset of Universal Dependencies (UD) Treebanks (v1.4) (Nivre et al., 2016), which cover 15 languages, following the setup of Kim et al. (2017). The task-specific labeling layer is the same as §4.3. POS tagging is evaluated by the accuracy of predicted POS tags (ACC).

### 4.5 Dependency parsing

Following the setup of Ahmad et al. (2019), we use a subset of Universal Dependencies (UD) Treebanks (v2.2) (Nivre et al., 2018), which includes 31 languages. Dependency parsing is evaluated by unlabelled attachment score (UAS) and labeled attachment score (LAS) [3]. We only predict the coarse-grain dependency label following Ahmad et al. We use the model of Dozat and Manning (2016), a graph-based parser as a task-specific layer. Their LSTM encoder is replaced by mBERT. Similar to §4.3, we only take the representation of the first subword of each word. We use masking to prevent the parser from operating on non-first subwords.

[3] Punctuations (PUNCT) and symbols (SYM) are excluded.

## 5 Experiments

We use the base cased multilingual BERT, which has $N = 12$ attention heads and $M = 12$ transformer blocks. The dropout probability is 0.1 and $d_h$ is 768. The model has 179M parameters with about 120k vocabulary.

**Training** For each task, no preprocessing is performed except tokenization of words into subwords with WordPiece. We use Adam (Kingma and Ba, 2014) for fine-tuning with $\beta_1$ of 0.9, $\beta_2$ of 0.999 and L2 weight decay of 0.01. We warm up the learning rate over the first 10% of batches and linearly decay the learning rate.

**Maximum Subwords Sequence Length** At training time, we limit the length of subwords sequence to 128 to fit in a single GPU for all tasks. For NER and POS tagging, we additionally use the sliding window approach. After the first window, we keep the last 64 subwords from the previous window as context. In other words, for a non-first window, only (up to) 64 new subwords are added for prediction. At evaluation time, we follow the same approach as training time except for parsing. We threshold the sentence length to 140 words, including words and punctuation, following Ahmad et al. (2019). In practice, the maximum subwords sequence length is the number of subwords of the first 140 words or 512, whichever is smaller.

**Hyperparameter Search and Model Selection** We select the best hyperparameters by searching a combination of batch size, learning rate and the number of fine-tuning epochs with the following range: learning rate $\{2\times10^{-5}, 3\times10^{-5}, 5\times10^{-5}\}$; batch size $\{16, 32\}$; number of epochs: $\{3, 4\}$. Note the best hyperparameters and model are selected by development performance in *English*.

### 5.1 Question #1: Is mBERT Multilingual?

**MLDoc** We include two strong baselines. Schwenk and Li (2018) use MultiCCA, multilingual word embeddings trained with a bilingual dictionary (Ammar et al., 2016), and convolution neural networks. Concurrent to our work,

| | en | de | zh | es | fr | it | ja | ru | Average |
|---|---|---|---|---|---|---|---|---|---|
| *In language supervised learning* | | | | | | | | | |
| Schwenk and Li (2018) | 92.2 | 93.7 | 87.3 | 94.5 | 92.1 | 85.6 | 85.4 | 85.7 | 89.5 |
| mBERT | 94.2 | 93.3 | 89.3 | 95.7 | 93.4 | 88.0 | 88.4 | 87.5 | 91.2 |
| *Zero-shot cross-lingual transfer* | | | | | | | | | |
| Schwenk and Li (2018) | 92.2 | 81.2 | 74.7 | 72.5 | 72.4 | **69.4** | **67.6** | 60.8 | 73.9 |
| Artetxe and Schwenk (2018) ♠ † | 89.9 | **84.8** | 71.9 | **77.3** | **78.0** | **69.4** | 60.3 | 67.8 | **74.9** |
| mBERT | **94.2** | 80.2 | **76.9** | 72.6 | 72.6 | 68.9 | 56.5 | **73.7** | 74.5 |

Table 2: MLDoc experiments. ♠ denotes the model is pretrained with bitext, and † denotes concurrent work. Bold and underline denote best and second best.

Artetxe and Schwenk (2018) use bitext between English/Spanish and the rest of languages to pretrain a multilingual sentence representation with a sequence-to-sequence model where the decoder only has access to a max-pooling of the encoder hidden states.

mBERT outperforms (Tab. 2) multilingual word embeddings and performs comparably with a multilingual sentence representation, even though mBERT does not have access to bitext. Interestingly, mBERT outperforms Artetxe and Schwenk (2018) in distantly related languages like Chinese and Russian and under-performs in closely related Indo-European languages.

**XNLI** We include three strong baselines, Artetxe and Schwenk (2018) and Lample and Conneau (2019) are concurrent to our work. Lample and Conneau (2019) with MLM is similar to mBERT; the main difference is that it only trains with the 15 languages of XNLI, has 249M parameters (around 40% more than mBERT), and MLM+TLM also uses bitext as training data [4]. Conneau et al. (2018) use supervised multilingual word embeddings with an LSTM encoder and max-pooling. After an English encoder and classifier are trained, the target encoder is trained to mimic the English encoder with ranking loss and bitext.

In Tab. 3, mBERT outperforms one model with bitext training but (as expected) falls short of models with more cross-lingual training information. Interestingly, mBERT and MLM are mostly the same except for the training languages, yet we observe that mBERT under-performs MLM by a large margin. We hypothesize that limiting pretraining to only those languages needed for the downstream task is beneficial. The gap between Artetxe and Schwenk (2018) and mBERT in XNLI is larger than MLDoc, likely because XNLI is harder.

[4]They also use language embeddings as input and exclude the next sentence classification objective

**NER** We use Xie et al. (2018) as a zero-shot cross-lingual transfer baseline, which is state-of-the-art on CoNLL 2002 and 2003. It uses unsupervised bilingual word embeddings (Conneau et al., 2017) with a hybrid of a character-level/word-level LSTM, self-attention, and a CRF. Pseudo training data is built by word-to-word translation with an induced dictionary from bilingual word embeddings.

mBERT outperforms a strong baseline by an average of 6.9 points absolute F1 and an 11.8 point absolute improvement in German with a simple one layer $0^{th}$-order CRF as a prediction function (Tab. 4). A large gap remains when transferring to distantly related languages (e.g. Chinese) compared to a supervised baseline. Further effort should focus on transferring between distantly related languages. In §5.4 we show that sharing subwords across languages helps transfer.

**POS** We use Kim et al. (2017) as a reference. They utilized a small amount of supervision in the target language as well as English supervision so the results are not directly comparable. Tab. 5 shows a large (average) gap between mBERT and Kim et al. Interestingly, mBERT still outperforms Kim et al. (2017) with 320 sentences in German (de), Polish (pl), Slovak (sk) and Swedish (sv).

**Dependency Parsing** We use the best performing model on average in Ahmad et al. (2019) as a zero-shot transfer baseline, i.e. transformer encoder with graph-based parser (Dozat and Manning, 2016), and dictionary supervised cross-lingual embeddings (Smith et al., 2017). Dependency parsers, including Ahmad et al., assume access to gold POS tags: a cross-lingual representation. We consider two versions of mBERT: with and without gold POS tags. When tags are available, a tag embedding is concatenated with the final output of mBERT.

Tab. 6 shows that mBERT outperforms the base-

| | en | fr | es | de | el | bg | ru | tr | ar | vi | th | zh | hi | sw | ur | Average |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *Pseudo supervision with machine translated training data from English to target language* | | | | | | | | | | | | | | | | |
| Lample and Conneau (2019) (MLM+TLM) ♠ † | 85.0 | 80.2 | 80.8 | 80.3 | 78.1 | 79.3 | 78.1 | 74.7 | 76.5 | 76.6 | 75.5 | 78.6 | 72.3 | 70.9 | 63.2 | 76.7 |
| mBERT | 82.1 | 76.9 | 78.5 | 74.8 | 72.1 | 75.4 | 74.3 | 70.6 | 70.8 | 67.8 | 63.2 | 76.2 | 65.3 | 65.3 | 60.6 | 71.6 |
| *Zero-shot cross-lingual transfer* | | | | | | | | | | | | | | | | |
| Conneau et al. (2018) (X-LSTM) ♠ ♢ | 73.7 | 67.7 | 68.7 | 67.7 | 68.9 | 67.9 | 65.4 | 64.2 | 64.8 | 66.4 | 64.1 | 65.8 | 64.1 | 55.7 | 58.4 | 65.6 |
| Artetxe and Schwenk (2018) ♠ † | 73.9 | 71.9 | 72.9 | 72.6 | 73.1 | 74.2 | 71.5 | 69.7 | 71.4 | 72.0 | 69.2 | 71.4 | 65.5 | 62.2 | 61.0 | 70.2 |
| Lample and Conneau (2019) (MLM+TLM) ♠ ♢ † | 85.0 | 78.7 | 78.9 | 77.8 | 76.6 | 77.4 | 75.3 | 72.5 | 73.1 | 76.1 | 73.2 | 76.5 | 69.6 | 68.4 | 67.3 | 75.1 |
| Lample and Conneau (2019) (MLM) ♢ † | 83.2 | 76.5 | 76.3 | 74.2 | 73.1 | 74.0 | 73.1 | 67.8 | 68.5 | 71.2 | 69.2 | 71.9 | 65.7 | 64.6 | 63.4 | 71.5 |
| mBERT | 82.1 | 73.8 | 74.3 | 71.1 | 66.4 | 68.9 | 69.0 | 61.6 | 64.9 | 69.5 | 55.8 | 69.3 | 60.0 | 50.4 | 58.0 | 66.3 |

Table 3: XNLI experiments. ♠ denotes the model is pretrained with cross-lingual signal including bitext or bilingual dictionary, † denotes concurrent work, and ♢ denotes model selection with target language dev set.

| | en | nl | es | de | zh | Average (-en,-zh) |
|---|---|---|---|---|---|---|
| *In language supervised learning* | | | | | | |
| Xie et al. (2018) | - | 86.40 | 86.26 | 78.16 | - | 83.61 |
| mBERT | 91.97 | 90.94 | 87.38 | 82.82 | 93.17 | 87.05 |
| *Zero-shot cross-lingual transfer* | | | | | | |
| Xie et al. (2018) | - | 71.25 | 72.37 | 57.76 | - | 67.13 |
| mBERT | 91.97 | **77.57** | **74.96** | **69.56** | 51.90 | **74.03** |

Table 4: NER tagging experiments.

line on average by 7.3 point UAS and 0.4 point LAS absolute improvement even without gold POS tags. Note in practice, gold POS tags are not always available, especially for low resource languages. Interestingly, the LAS of mBERT tends to weaker than the baseline in languages with less word order distance, in other words, more closely related to English. With the help of gold POS tags, we further observe 1.6 points UAS and 4.7 point LAS absolute improvement on average. It appears that adding gold POS tags, which provide clearer cross-lingual representations, benefit mBERT.

**Summary** Across all five tasks, mBERT demonstrate strong (sometimes state-of-the-art) zero-shot cross-lingual performance without any cross-lingual signal. It outperforms cross-lingual embeddings in four tasks. With a small amount of target language supervision and cross-lingual signal, mBERT may improve further; we leave this as future work. In short, mBERT is a surprisingly effective cross-lingual model for many NLP tasks.

### 5.2 Question #2: Does mBERT vary layer-wise?

The goal of a deep neural network is to abstract to higher-order representations as you progress up the hierarchy (Yosinski et al., 2014). Peters et al. (2018) empirically show that for ELMo in English the lower layer is better at syntax while the upper layer is better at semantics. However, it is unclear how different layers affect the quality of cross-lingual representation. For mBERT, we hypothesize a similar generalization across the 13 layers, as well as an abstraction away from a specific language with higher layers. Does the zero-shot transfer performance vary with different layers?

We consider two schemes. First, we follow the feature-based approach of ELMo by taking a learned weighted combination of all 13 layers of mBERT with a two-layer bidirectional LSTM with $d_h$ hidden size (Feat). Note the LSTM is trained from scratch and mBERT is fixed. For sentence and document classification, an additional max-pooling is used to extract a fixed-dimension vector. We train the feature-based approach with Adam and learning rate 1e-3. The batch size is 32. The learning rate is halved whenever the development evaluation does not improve. The training is stopped early when learning rate drop below 1e-5. Second, when fine-tuning mBERT, we fix the bottom $n$ layers ($n$ included) of mBERT, where layer 0 is the input embedding. We consider $n \in \{0, 3, 6, 9\}$.

Freezing the bottom layers of mBERT, in general, improves the performance of mBERT in all five tasks (Fig. 1). For sentence-level tasks like document classification and natural language inference, we observe the largest improvement with $n = 6$. For word-level tasks like NER, POS tagging, and parsing, we observe the largest improvement with $n = 3$. More improvement in under-performing languages is observed.

In each task, the feature-based approach with LSTM under-performs fine-tuning approach. We hypothesize that initialization from pretraining with lots of languages provides a very good starting point that is hard to beat. Additionally, the LSTM could also be part of the problem. In Ahmad et al. (2019) for dependency parsing, an LSTM encoder was worse than a transformer when transferring

| lang | bg | da | de | en | es | fa | hu | it | nl | pl | pt | ro | sk | sl | sv | Average (-en) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *In language supervised learning* | | | | | | | | | | | | | | | | |
| mBERT | 99.0 | 97.9 | 95.2 | 97.1 | 97.1 | 97.8 | 96.9 | 98.7 | 92.1 | 98.5 | 98.3 | 97.8 | 97.0 | 98.9 | 98.4 | 97.4 |
| *Low resource cross-lingual transfer* | | | | | | | | | | | | | | | | |
| Kim et al. (2017) (1280) | 95.7 | 94.3 | 90.7 | - | 93.4 | 94.8 | 94.5 | 95.9 | 85.8 | 92.1 | 95.5 | 94.2 | 90.0 | 94.1 | 94.6 | 93.3 |
| Kim et al. (2017) (320) | 92.4 | 90.8 | 89.7 | - | 90.9 | 91.8 | 90.7 | 94.0 | 82.2 | 85.5 | 94.2 | 91.4 | 83.2 | 90.6 | 90.7 | 89.9 |
| *Zero-shot cross-lingual transfer* | | | | | | | | | | | | | | | | |
| mBERT | 87.4 | 88.3 | 89.8 | 97.1 | 85.2 | 72.8 | 83.2 | 84.7 | 75.9 | 86.9 | 82.1 | 84.7 | 83.6 | 84.2 | 91.3 | 84.3 |

Table 5: POS tagging. Kim et al. (2017) use small amounts of training data in the target language.

| | Dist | mBERT(S) | Baseline(Z) | mBERT(Z) | mBERT(Z+POS) |
|---|---|---|---|---|---|
| en | 0.00 | 91.5/81.3 | 90.4/**88.4** | 91.5/81.3 | **91.8**/82.2 |
| no | 0.06 | 93.6/85.9 | 80.8/**72.8** | 80.6/68.9 | **82.7**/72.1 |
| sv | 0.07 | 91.2/83.1 | 81.0/73.2 | 82.5/71.2 | **84.3/73.7** |
| fr | 0.09 | 91.7/85.4 | 77.9/72.8 | 82.7/72.7 | **83.8/76.2** |
| pt | 0.09 | 93.2/87.2 | 76.6/**67.8** | 77.1/64.0 | **78.3**/66.9 |
| da | 0.10 | 89.5/81.9 | 76.6/67.9 | 77.4/64.7 | **79.3/68.1** |
| es | 0.12 | 92.3/86.5 | 74.5/66.4 | 78.1/64.9 | **79.0/68.9** |
| it | 0.12 | 94.8/88.7 | 80.8/75.8 | 84.6/74.4 | **86.0/77.8** |
| ca | 0.13 | 94.3/89.5 | 73.8/65.1 | 78.1/64.6 | **79.0/67.9** |
| hr | 0.13 | 92.4/83.8 | 61.9/52.9 | **80.7**/65.8 | 80.4/**68.2** |
| pl | 0.13 | 94.7/79.9 | 74.6/62.2 | 82.8/59.4 | **85.7/65.4** |
| sl | 0.13 | 88.0/77.8 | 68.2/56.5 | 72.6/51.4 | **75.9/59.2** |
| uk | 0.13 | 90.6/83.4 | 60.1/52.3 | **76.7**/60.0 | 76.5/**65.5** |
| bg | 0.14 | 95.2/85.5 | 79.4/**68.2** | 83.3/62.3 | **84.4**/68.1 |
| cs | 0.14 | 94.2/86.6 | 63.1/53.8 | 76.6/58.7 | **77.4/63.6** |
| de | 0.14 | 86.1/76.5 | 71.3/61.6 | 80.4/66.3 | **83.5/71.2** |
| he | 0.14 | 91.9/83.6 | 55.3/48.0 | **67.5**/48.4 | 67.0/**54.3** |
| nl | 0.14 | 94.0/85.0 | 68.6/60.3 | 78.0/64.8 | **79.9/67.1** |
| ru | 0.14 | 94.7/88.0 | 60.6/51.6 | **73.6**/58.5 | 73.2/**61.5** |
| ro | 0.15 | 92.2/83.2 | 65.1/54.1 | **77.0**/58.5 | 76.9/**62.6** |
| id | 0.17 | 86.3/75.4 | 49.2/43.5 | **62.6**/45.6 | 59.8/**48.6** |
| sk | 0.17 | 93.8/83.3 | 66.7/58.2 | 82.7/63.9 | **82.9/67.8** |
| lv | 0.18 | 87.3/75.3 | **70.8/49.3** | 66.0/41.4 | 70.4/48.5 |
| et | 0.20 | 88.8/79.7 | 65.7/44.9 | 66.9/44.3 | **70.8/50.7** |
| fi | 0.20 | 91.3/81.8 | 66.3/48.7 | 68.4/47.5 | **71.4/52.5** |
| zh* | 0.23 | 88.3/81.2 | 42.5/25.1 | **53.8**/26.8 | 53.4/**29.0** |
| ar | 0.26 | 87.6/80.6 | 38.1/28.0 | 43.9/28.3 | **44.7/32.9** |
| la | 0.28 | 85.2/73.1 | 48.0/**35.2** | 47.9/26.1 | **50.9**/32.2 |
| ko | 0.33 | 86.0/74.8 | 34.5/16.4 | **52.7**/27.5 | 52.3/**29.4** |
| hi | 0.40 | 94.8/86.7 | 35.5/26.5 | 49.8/33.2 | **58.9/44.0** |
| ja* | 0.49 | 94.2/87.4 | 28.2/20.9 | 36.6/15.7 | **41.3/30.9** |
| AVER | 0.17 | 91.3/82.6 | 64.1/53.8 | 71.4/54.2 | **73.0/58.9** |

Table 6: Dependency parsing results by language (UAS/LAS). * denotes delexicalized parsing in the baseline. S and Z denotes supervised learning and zero-shot transfer. Bold and underline denotes best and second best. We order the languages by word order distance to English.

to languages with high word ordering distance to English.

### 5.3 Question #3: Does mBERT retain language specific information?

mBERT may learn a cross-lingual representation by abstracting away from language-specific information, thus losing the ability to distinguish between languages. We test this by considering language identification: does mBERT retain language-specific information? We use WiLI-2018 (Thoma, 2018), which includes over 200 languages from Wikipedia. We keep only those languages included in mBERT, leaving 99 languages [5]. We take various layers of bag-of-words mBERT representation of the first two sentences of the test paragraph and add a linear classifier with softmax. We fix mBERT and train *only* the classifier the same as the feature-based approach in §5.2.

All tested layers achieved around 96% accuracy (Fig. 2), with no clear difference between layers. This suggests each layer contains language-specific information; surprising given the zero-shot cross-lingual abilities. As mBERT generalizes its representations and creates cross-lingual representations, it maintains language-specific details. This may be encouraged during pretraining since mBERT needs to retain enough language-specific information to perform the cloze task.

### 5.4 Question #4: Does mBERT benefit by sharing subwords across languages?

As discussed in §3, mBERT shares subwords in closely related languages or perhaps in distantly related languages. At training time, the representation of a shared subword is explicitly trained to contain enough information for the cloze task in all languages in which it appears. During fine-tuning for zero-shot cross-lingual transfer, if a subword in the target language test set also appears in the source language training data, the supervision could be leaked to the target language explicitly. However, all subwords interact in a non-interpretable way inside a deep network, and subword representations could overfit to the source language and potentially hurt transfer performance. In these experiments, we investigate how sharing subwords across languages effects cross-lingual transfer.

To quantify how many subwords are shared

[5] Hungarian, Western-Punjabi, Norwegian-Bokmal, and Piedmontese are not covered by WiLI.

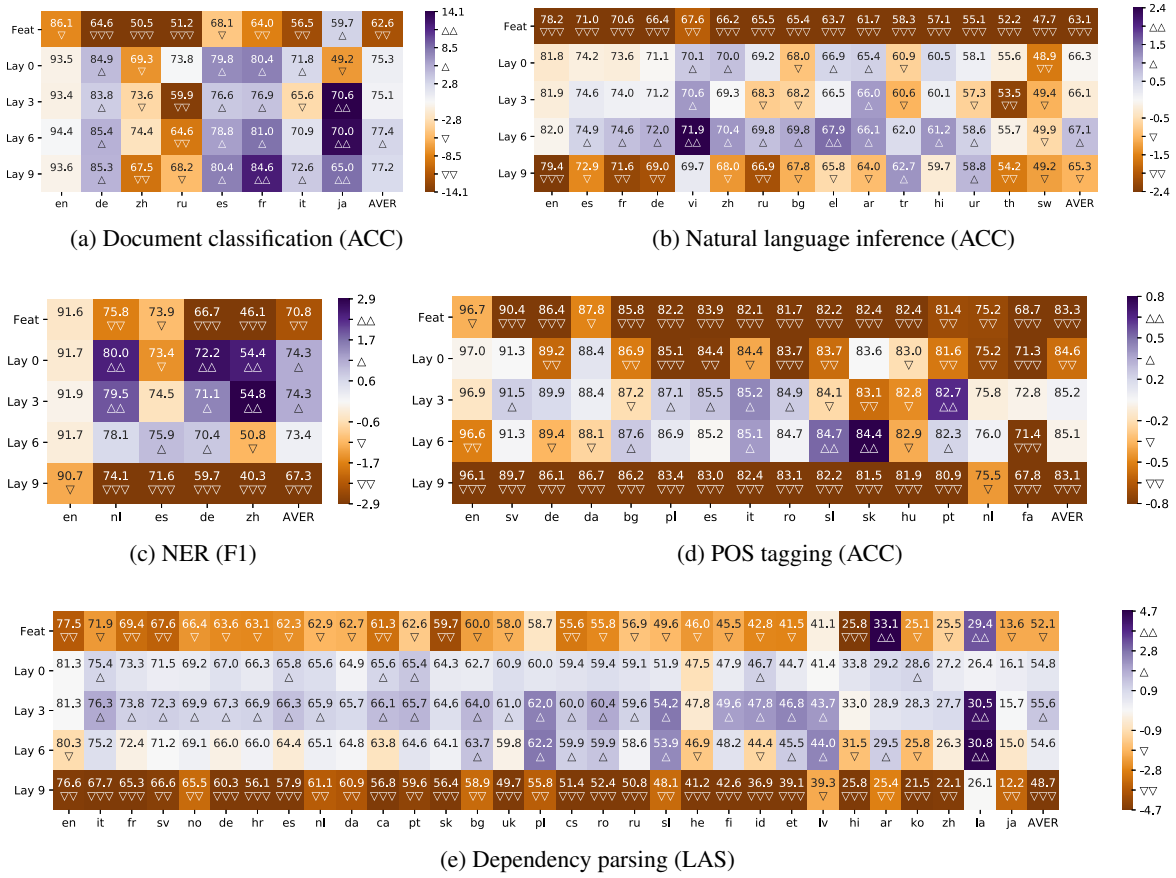


(a) Document classification (ACC)

(b) Natural language inference (ACC)

(c) NER (F1)

(d) POS tagging (ACC)

(e) Dependency parsing (LAS)

Figure 1: Performance of different fine-tuning approaches compared with fine-tuning all mBERT parameters. Color denotes absolute difference and number in each entry is the evaluation in the corresponding setting. Languages are sorted by mBERT zero-shot transfer performance. Three downward triangles indicate performance drop more than the legends lower limit.

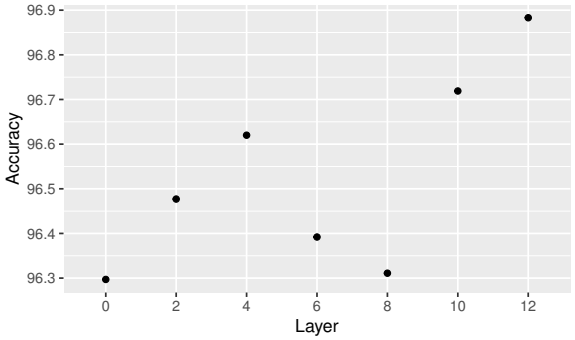


Figure 2: Language identification accuracy for different layer of mBERT. layer 0 is the embedding layer and the layer $i > 0$ is output of the i[th] transformer block.

across languages in any task, we assume $V^{\text{en}}_{\text{train}}$ is the set of all subwords in the English training set, $V^{\ell}_{\text{test}}$ is the set of all subwords in language $\ell$ test set, and $c^{\ell}_{w}$ is the count of subword $w$ in test set of language $\ell$. We then calculate the percentage of observed subwords at type-level $p^{\ell}_{\text{type}}$ and token-level $p^{\ell}_{\text{token}}$ for each target language $\ell$.

$$p^{\ell}_{\text{type}} = \frac{|V^{\ell}_{\text{obs}}|}{|V^{\ell}_{\text{test}}|} \times 100$$

$$p^{\ell}_{\text{token}} = \frac{\sum_{w \in V^{\ell}_{\text{obs}}} c^{\ell}_{w}}{\sum_{w \in V^{\ell}_{\text{test}}} c^{\ell}_{w}} \times 100$$

where $V^{\ell}_{\text{obs}} = V^{\text{en}}_{\text{train}} \cap V^{\ell}_{\text{test}}$.

In Fig. 3, we show the relation between cross-lingual zero-shot transfer performance of mBERT and $p^{\ell}_{\text{type}}$ or $p^{\ell}_{\text{token}}$ for all five tasks with Pearson correlation. In four out of five tasks (not XNLI) we observed a strong positive correlation ($p < 0.05$) with a correlation coefficient larger than 0.5. In Indo-European languages, we observed $p^{\ell}_{\text{token}}$ is usually around 50% to 75% while $p^{\ell}_{\text{type}}$ is usually less than 50%. This indicates that subwords shared across languages are usually high frequency[6]. We

[6]With the data-dependent WordPiece algorithm, subwords that appear in multiple languages with high frequency are more likely to be selected.

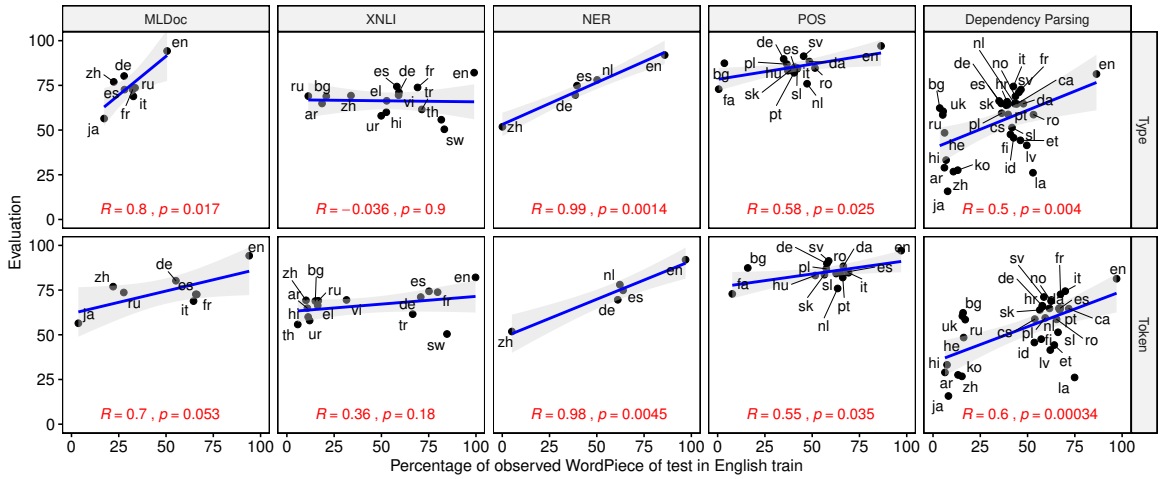


Figure 3: Relation between cross-lingual zero-shot transfer performance with mBERT and percentage of observed subwords at both type-level and token-level. Pearson correlation coefficient and $p$-value are shown in red.

hypothesize that this could be used as a simple indicator for selecting source language in cross-lingual transfer with mBERT. We leave this for future work.

## 6 Discussion

We show mBERT does well in a cross-lingual zero-shot transfer setting on five different tasks covering a large number of languages. It outperforms cross-lingual embeddings, which typically have more cross-lingual supervision. By fixing the bottom layers of mBERT during fine-tuning, we observe further performance gains. Language-specific information is preserved in all layers. Sharing subwords helps cross-lingual transfer; a strong correlation is observed between the percentage of overlapping subwords and transfer performance.

mBERT effectively learns a good multilingual representation with strong cross-lingual zero-shot transfer performance in various tasks. We recommend building future multi-lingual NLP models on top of mBERT or other models pretrained similarly. Even without explicit cross-lingual supervision, these models do very well. As we show with XNLI in §5.1, while bitext is hard to obtain in low resource settings, a variant of mBERT pretrained with bitext (Lample and Conneau, 2019) shows even stronger performance. Future work could investigate how to use weak supervision to produce a better cross-lingual mBERT, or adapt an already trained model for cross-lingual use. With POS tagging in §5.1, we show mBERT, in general, under-performs models with a small amount of supervision while Devlin et al. (2019) show that in English NLP tasks, fine-tuning BERT only needs a small amount of data. Future work could investigate when cross-lingual transfer is helpful in NLP tasks of low resource languages. With such strong cross-lingual NLP performance, it would be interesting to prob mBERT from a linguistic perspective in the future.

## References

Wasi Ahmad, Zhisong Zhang, Xuezhe Ma, Eduard Hovy, Kai-Wei Chang, and Nanyun Peng. 2019. On difficulties of cross-lingual transfer with order differences: A case study on dependency parsing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2440–2452, Minneapolis, Minnesota. Association for Computational Linguistics.

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A Smith. 2016. Massively multilingual word embeddings. *arXiv preprint arXiv:1602.01925*.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018. A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 789–798, Melbourne, Australia. Association for Computational Linguistics.

Mikel Artetxe and Holger Schwenk. 2018. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *arXiv preprint arXiv:1812.10464*.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hin-

ton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485, Brussels, Belgium. Association for Computational Linguistics.

Jacob Devlin. 2018. Multilingual bert readme document.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Timothy Dozat and Christopher D Manning. 2016. Deep biaffine attention for neural dependency parsing. *arXiv preprint arXiv:1611.01734*.

Dan Hendrycks and Kevin Gimpel. 2016. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *arXiv preprint arXiv:1606.08415*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Joo-Kyung Kim, Young-Bum Kim, Ruhi Sarikaya, and Eric Fosler-Lussier. 2017. Cross-lingual transfer learning for POS tagging without cross-lingual resources. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2832–2838, Copenhagen, Denmark. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*.

Gina-Anne Levow. 2006. The third international Chinese language processing bakeoff: Word segmentation and named entity recognition. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 108–117, Sydney, Australia. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Phoebe Mulcaire, Jungo Kasai, and Noah A. Smith. 2019. Polyglot contextual representations improve crosslingual transfer. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3912–3918, Minneapolis, Minnesota. Association for Computational Linguistics.

Joakim Nivre, Mitchell Abrams, Željko Agić, Lars Ahrenberg, Lene Antonsen, Maria Jesus Aranzabe, Gashaw Arutie, Masayuki Asahara, Luma Ateyah, Mohammed Attia, Aitziber Atutxa, Liesbeth Augustinus, Elena Badmaeva, Miguel Ballesteros, Esha Banerjee, Sebastian Bank, Verginica Barbu Mititelu, John Bauer, Sandra Bellato, Kepa Bengoetxea, Riyaz Ahmad Bhat, Erica Biagetti, Eckhard Bick, Rogier Blokland, Victoria Bobicev, Carl Börstell, Cristina Bosco, Gosse Bouma, Sam Bowman, Adriane Boyd, Aljoscha Burchardt, Marie Candito, Bernard Caron, Gauthier Caron, Gülşen Cebiroğlu Eryiğit, Giuseppe G. A. Celano, Savas Cetin, Fabricio Chalub, Jinho Choi, Yongseok Cho, Jayeol Chun, Silvie Cinková, Aurélie Collomb, Çağrı Çöltekin, Miriam Connor, Marine Courtin, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Arantza Diaz de Ilarraza, Carly Dickerson, Peter Dirix, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Marhaba Eli, Ali Elkahky, Binyam Ephrem, Tomaž Erjavec, Aline Etienne, Richárd Farkas, Hector Fernandez Alcalde, Jennifer Foster, Cláudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Moa Gärdenfors, Kim Gerdes, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökırmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Céline Guillot-Barbance, Nizar Habash, Jan Hajič, Jan Hajič jr., Linh Hà Mỹ, Na-Rae Han, Kim Harris, Dag Haug, Barbora Hladká, Jaroslava Hlaváčová, Florinel Hociung, Petter Hohle, Jena Hwang, Radu Ion, Elena Irimia, Tomáš Jelínek, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşıkara, Sylvain Kahane, Hiroshi Kanayama, Jenna Kanerva, Tolga Kayadelen, Václava Kettnerová, Jesse Kirchner, Natalia Kotsyba, Simon Krek, Sookyoung Kwak,

Veronika Laippala, Lorenzo Lambertino, Tatiana Lando, Septina Dian Larasati, Alexei Lavrentiev, John Lee, Phng Lê H`ông, Alessandro Lenci, Saran Lertpradit, Herman Leung, Cheuk Ying Li, Josie Li, Keying Li, KyungTae Lim, Nikola Ljubešić, Olga Loginova, Olga Lyashevskaya, Teresa Lynn, Vivien Macketanz, Aibek Makazhanov, Michael Mandl, Christopher Manning, Ruli Manurung, Cătălina Mărănduc, David Mareček, Katrin Marheinecke, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Gustavo Mendonça, Niko Miekka, Anna Missilä, Cătălin Mititelu, Yusuke Miyao, Simonetta Montemagni, Amir More, Laura Moreno Romero, Shinsuke Mori, Bjartur Mortensen, Bohdan Moskalevskyi, Kadri Muischnek, Yugo Murawaki, Kaili Müürisep, Pinkey Nainwani, Juan Ignacio Navarro Horñiacek, Anna Nedoluzhko, Gunta Nešpore-Bērzkalne, Lng Nguy˜ên Thị, Huy`ên Nguy˜ên Thị Minh, Vitaly Nikolaev, Rattima Nitisaroj, Hanna Nurmi, Stina Ojala, Adédayọ̀ Olúòkun, Mai Omura, Petya Osenova, Robert Östling, Lilja Øvrelid, Niko Partanen, Elena Pascual, Marco Passarotti, Agnieszka Patejuk, Siyao Peng, Cenel-Augusto Perez, Guy Perrier, Slav Petrov, Jussi Piitulainen, Emily Pitler, Barbara Plank, Thierry Poibeau, Martin Popel, Lauma Pretkalniņa, Sophie Prévost, Prokopis Prokopidis, Adam Przepiórkowski, Tiina Puolakainen, Sampo Pyysalo, Andriela Rääbis, Alexandre Rademaker, Loganathan Ramasamy, Taraka Rama, Carlos Ramisch, Vinit Ravishankar, Livy Real, Siva Reddy, Georg Rehm, Michael Rießler, Larissa Rinaldi, Laura Rituma, Luisa Rocha, Mykhailo Romanenko, Rudolf Rosa, Davide Rovati, Valentin Roca, Olga Rudina, Shoval Sadde, Shadi Saleh, Tanja Samardžić, Stephanie Samson, Manuela Sanguinetti, Baiba Saulīte, Yanin Sawanakunanon, Nathan Schneider, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Mo Shen, Atsuko Shimada, Muh Shohibussirri, Dmitry Sichinava, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Isabela Soares-Bastos, Antonio Stella, Milan Straka, Jana Strnadová, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Dima Taji, Yuta Takahashi, Takaaki Tanaka, Isabelle Tellier, Trond Trosterud, Anna Trukhina, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Zdeňka Urešová, Larraitz Uria, Hans Uszkoreit, Sowmya Vajjala, Daniel van Niekerk, Gertjan van Noord, Viktor Varga, Veronika Vincze, Lars Wallin, Jonathan North Washington, Seyi Williams, Mats Wirén, Tsegay Woldemariam, Tak-sum Wong, Chunxiao Yan, Marat M. Yavrumyan, Zhuoran Yu, Zdeněk Žabokrtský, Amir Zeldes, Daniel Zeman, Manying Zhang, and Hanzhi Zhu. 2018. Universal dependencies 2.2. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Joakim Nivre, Željko Agić, Lars Ahrenberg, Maria Jesus Aranzabe, Masayuki Asahara, Aitziber Atutxa, Miguel Ballesteros, John Bauer, Kepa Bengoetxea, Yevgeni Berzak, Riyaz Ahmad Bhat, Eckhard Bick, Carl Börstell, Cristina Bosco, Gosse Bouma, Sam Bowman, Gülşen Cebiroğlu Eryiğit, Giuseppe G. A. Celano, Fabricio Chalub, Çağrı Çöltekin, Miriam Connor, Elizabeth Davidson, Marie-Catherine de Marneffe, Arantza Diaz de Ilarraza, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Marhaba Eli, Tomaž Erjavec, Richárd Farkas, Jennifer Foster, Claudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Moa Gärdenfors, Sebastian Garza, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökırmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Jan Hajič, Linh Hà Mỹ, Dag Haug, Barbora Hladká, Radu Ion, Elena Irimia, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşıkara, Hiroshi Kanayama, Jenna Kanerva, Boris Katz, Jessica Kenney, Natalia Kotsyba, Simon Krek, Veronika Laippala, Lucia Lam, Phng Lê H`ông, Alessandro Lenci, Nikola Ljubešić, Olga Lyashevskaya, Teresa Lynn, Aibek Makazhanov, Christopher Manning, Cătălina Mărănduc, David Mareček, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Anna Missilä, Verginica Mititelu, Yusuke Miyao, Simonetta Montemagni, Keiko Sophie Mori, Shunsuke Mori, Bohdan Moskalevskyi, Kadri Muischnek, Nina Mustafina, Kaili Müürisep, Lng Nguy˜ên Thị, Huy`ên Nguy˜ên Thị Minh, Vitaly Nikolaev, Hanna Nurmi, Petya Osenova, Robert Östling, Lilja Øvrelid, Valeria Paiva, Elena Pascual, Marco Passarotti, Cenel-Augusto Perez, Slav Petrov, Jussi Piitulainen, Barbara Plank, Martin Popel, Lauma Pretkalniņa, Prokopis Prokopidis, Tiina Puolakainen, Sampo Pyysalo, Alexandre Rademaker, Loganathan Ramasamy, Livy Real, Laura Rituma, Rudolf Rosa, Shadi Saleh, Baiba Saulīte, Sebastian Schuster, Wolfgang Seeker, Mojgan Seraji, Lena Shakurova, Mo Shen, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Carolyn Spadine, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Takaaki Tanaka, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Larraitz Uria, Gertjan van Noord, Viktor Varga, Veronika Vincze, Lars Wallin, Jing Xian Wang, Jonathan North Washington, Mats Wirén, Zdeněk Žabokrtský, Amir Zeldes, Daniel Zeman, and Hanzhi Zhu. 2016. Universal dependencies 1.4. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Jacob Perkins. 2014. *Python 3 text processing with NLTK 3 cookbook*. Packt Publishing Ltd.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001, Florence, Italy. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2017. A survey of cross-lingual word embedding models. *arXiv preprint arXiv:1706.04902*.

Tal Schuster, Ori Ram, Regina Barzilay, and Amir Globerson. 2019. Cross-lingual alignment of contextual word embeddings, with applications to zero-shot dependency parsing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1599–1613, Minneapolis, Minnesota. Association for Computational Linguistics.

Holger Schwenk and Xian Li. 2018. A corpus for multilingual document classification in eight languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*, Miyazaki, Japan. European Languages Resources Association (ELRA).

Samuel L Smith, David HP Turban, Steven Hamblin, and Nils Y Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. *arXiv preprint arXiv:1702.03859*.

Wilson L Taylor. 1953. cloze procedure: A new tool for measuring readability. *Journalism Bulletin*, 30(4):415–433.

Martin Thoma. 2018. The wili benchmark dataset for written language identification. *arXiv preprint arXiv:1801.07779*.

Erik F. Tjong Kim Sang. 2002. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Jiateng Xie, Zhilin Yang, Graham Neubig, Noah A. Smith, and Jaime Carbonell. 2018. Neural cross-lingual named entity recognition with minimal resources. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 369–379, Brussels, Belgium. Association for Computational Linguistics.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328.

Daniel Zeman and Philip Resnik. 2008. Cross-language parser adaptation between related languages. In *Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages*.

# Iterative Dual Domain Adaptation for Neural Machine Translation

**Jiali Zeng**[1], **Yang Liu**[2], **Jinsong Su**[1]*, **Yubin Ge**[3], **Yaojie Lu**[4], **Yongjing Yin**[1], **Jiebo Luo**[5]
[1]Xiamen University, Xiamen, China [2]Tsinghua University, Beijing, China
[3]University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA
[4]Institute of Software, Chinese Academy of Sciences, Beijing, China
[5]Department of Computer Science, University of Rochester, Rochester NY 14627, USA
lemon@stu.xmu.edu.cn liuyang2011@tsinghua.edu.cn
jssu@xmu.edu.cn

## Abstract

Previous studies on the domain adaptation for neural machine translation (NMT) mainly focus on the one-pass transferring out-of-domain translation knowledge to in-domain NMT model. In this paper, we argue that such a strategy fails to fully extract the domain-shared translation knowledge, and repeatedly utilizing corpora of different domains can lead to better distillation of domain-shared translation knowledge. To this end, we propose an iterative dual domain adaptation framework for NMT. Specifically, we first pre-train in-domain and out-of-domain NMT models using their own training corpora respectively, and then iteratively perform bidirectional translation knowledge transfer (from in-domain to out-of-domain and then vice versa) based on knowledge distillation until the in-domain NMT model convergences. Furthermore, we extend the proposed framework to the scenario of multiple out-of-domain training corpora, where the above-mentioned transfer is performed sequentially between the in-domain and each out-of-domain NMT models in the ascending order of their domain similarities. Empirical results on Chinese-English and English-German translation tasks demonstrate the effectiveness of our framework.

## 1 Introduction

Currently, neural machine translation (NMT) has become dominant in the community of machine translation due to its excellent performance (Bahdanau et al., 2015; Wu et al., 2016; Vaswani et al., 2017). With the development of NMT, prevailing NMT models become more and more complex with large numbers of parameters, which often require abundant corpora for effective training. However, for translation tasks in most domains, domain-specific parallel sentences are often scarce. If we only use domain-specific data to train the NMT model for such a domain, the performance of resulting model is usually unsatisfying. Therefore, NMT for low-resource domains becomes a challenge in its research and applications.

To deal with this issue, many researchers have conducted studies on the domain adaptation for NMT, which can be classified into two general categories. One is to transfer the rich-resource domain (out-of-domain) translation knowledge to benefit the low-resource (in-domain) NMT model. The other is to use the mixed-domain training corpus to construct a unified NMT model for all domains. Here, we mainly focus on the first type of research, of which typical methods include fine-tuning (Luong and Manning, 2015; Zoph et al., 2016; Servan et al., 2016), mixed fine-tuning (Chu et al., 2017), cost weighting (Chen et al., 2017), data selection (Wang et al., 2017a,b; Zhang et al., 2019a) and so on. The underlying assumption of these approaches is that in-domain and out-of-domain NMT models share the same parameter space or prior distributions, and the useful out-of-domain translation knowledge can be completely transferred to in-domain NMT model in a one-pass manner. However, it is difficult to achieve this goal due to domain differences. Particularly, when the domain difference is significant, such conventional brute-force transfer may be unsuccessful, facing the similar issue as the domain adaptation for other tasks (Pan and Yang, 2010).

In this paper, to tackle the above problem, we argue that corpora of different domains should be repeatedly utilized to fully distill domain-shared translation knowledge. To this end, we propose a novel **I**terative **D**ual **D**omain **A**daptation (IDDA) framework for NMT. Under this framework, we first train in-domain and out-of-domain NMT models using their own training corpora re-

*Corresponding author.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 845–855, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

spectively, and then iteratively perform bidirectional translation knowledge transfer (from in-domain to out-of-domain and then vice versa). In this way, both in-domain and out-of-domain NMT models are expected to constantly reinforce each other, which is likely to achieve better NMT domain adaptation. Particularly, we employ a *knowledge distillation* (Hinton et al., 2015; Kim and Rush, 2016) based approach to transfer translation knowledge. During this process, the target-domain NMT model is first initialized with the source-domain NMT model, and then trained to fit its own training data and match the output of its previous best model simultaneously. By doing so, the previously transferred translation knowledge can be effectively retained for better NMT domain adaptation. Finally, we further extend the proposed framework to the scenario of multiple out-of-domain training corpora, where the above-mentioned bidirectional knowledge transfer is performed sequentially between the in-domain and each out-of-domain NMT models in the ascending order of their domain similarities.

The contributions of this work are summarized as follows:

- We propose an iterative dual domain adaptation framework for NMT, which is applicable to many conventional domain transfer approaches, such as fine-tune, mixed fine-tune. Compared with previous approaches, our framework is able to better exploit domain-shared translation knowledge for NMT domain adaptation.
- We extend our framework to the setting of multiple out-of-domain training corpora, which is rarely studied in machine translation. Moreover, we explicitly differentiate the contributions of different out-of-domain training corpora based on the domain-level similarity with in-domain training corpus.
- We provide empirical evaluations of the proposed framework on Chinese-English, German-English datasets for NMT domain adaptation. Experimental results demonstrate the effectiveness of our framework. Moreover, we deeply analyze impacts of various factors on our framework[1].

[1]We release code and results at https://github.com/DeepLearnXMU/IDDA.

## 2 Related Work

Our work is obviously related to the research on transferring the out-of-domain translation knowledge into the in-domain NMT model. In this aspect, fine-tuning (Luong and Manning, 2015; Zoph et al., 2016; Servan et al., 2016) is the most popular approach, where the NMT model is first trained using the out-of-domain training corpus, and then fine-tuned on the in-domain training corpus. To avoid overfitting, Chu et al. (2017) blended in-domain with out-of-domain corpora to fine-tune the pre-trained model, and Freitag and Al-Onaizan (2016) combined the fine-tuned model with the baseline via ensemble method. Meanwhile, applying data weighting into NMT domain adaptation has attracted much attention. Wang et al. (2017a) and Wang et al. (2017b) proposed several sentence and domain weighting methods with a dynamic weight learning strategy. Zhang et al. (2019a) ranked unlabeled domain training samples based on their similarity to in-domain data, and then adopts a probabilistic curriculum learning strategy during training. Chen et al. (2017) applied the sentence-level cost weighting to refine the training of NMT model. Recently, Vilar (2018) introduced a weight to each hidden unit of out-of-domain model. Chu and Wang (2018) gave a comprehensive survey of the dominant domain adaptation techniques for NMT. Gu et al. (2019) not only maintained a private encoder and a private decoder for each domain, but also introduced a common encoder and a common decoder shared by all domains.

Significantly different from the above methods, along with the studies of dual learning for NMT (He et al., 2016; Wang et al., 2018; Zhang et al., 2019b), we iteratively perform bidirectional translation knowledge transfer between in-domain and out-of-domain training corpora. To the best of our knowledge, our work is the first attempt to explore such a dual learning based framework for NMT domain adaptation. Furthermore, we extend our framework to the scenario of multiple out-of-domain corpora. Particularly, we introduce knowledge distillation into the domain adaptation for NMT and experimental results demonstrate its effectiveness, echoing its successful applications on many tasks, such as speech recognition (Hinton et al., 2015) and natural language processing (Kim and Rush, 2016; Tan et al., 2019).

Besides, our work is also related to the studies

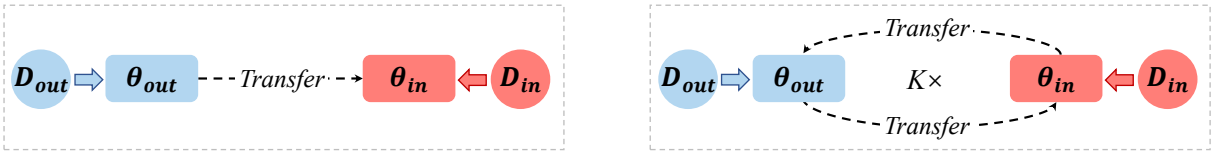


(a) Non-iterative Domain Adaptation

(b) Iterative Dual Domain Adaptation

Figure 1: Traditional approach vs IDDA framework for one-to-one NMT domain adaptation. $D_{out}$: out-of-domain training corpus, $D_{in}$: in-domain training corpus, $\theta_{out}$: out-of-domain NMT model, $\theta_{in}$: in-domain NMT model, $\boldsymbol{K}$ denotes the iteration number.

**Algorithm 1** Iterative Dual Domain Adaptation for NMT

**Input:** Training corpora $\{D_{in}, D_{out}\}$, development sets $\{D_{in}^{v}, D_{out}^{v}\}$, and the maximal iteration number $K$.
**Output:** In-domain NMT model $\theta_{in}^{*}$.
$\theta_{in}^{(0)} \leftarrow \text{TrainModel}(D_{in})$, $\theta_{out}^{(0)} \leftarrow \text{TrainModel}(D_{out})$
$\theta_{in}^{*} \leftarrow \theta_{in}^{(0)}$, $\theta_{out}^{*} \leftarrow \theta_{out}^{(0)}$
**for** $k = 1, 2, ..., K$ **do**
  $\theta_{out}^{(k)} \leftarrow \text{TransferModel}(\theta_{in}^{(k-1)}, D_{out}, \theta_{out}^{*})$
  **if** $\text{EvalModel}(D_{out}^{v}, \theta_{out}^{(k)}) > \text{EvalModel}(D_{out}^{v}, \theta_{out}^{*})$
    $\theta_{out}^{*} \leftarrow \theta_{out}^{(k)}$
  **end if**
  $\theta_{in}^{(k)} \leftarrow \text{TransferModel}(\theta_{out}^{(k)}, D_{in}, \theta_{in}^{*})$
  **if** $\text{EvalModel}(D_{in}^{v}, \theta_{in}^{(k)}) > \text{EvalModel}(D_{in}^{v}, \theta_{in}^{*})$
    $\theta_{in}^{*} \leftarrow \theta_{in}^{(k)}$
  **end if**
**end for**

of multi-domain NMT, which focus on building a unified NMT model trained on the mixed-domain training corpus for translation tasks in all domains (Kobus et al., 2016; Tars and Fishel, 2018; Farajian et al., 2017; Pryzant et al., 2017; Sajjad et al., 2017; Zeng et al., 2018; Bapna and Firat, 2019). Although our framework is also able to refine out-of-domain NMT model, it is still significantly different from multi-domain NMT, since only the performance of in-domain NMT model is considered.

Finally, note that similar to our work, Tan et al. (2019) introduced knowledge distillation into multilingual NMT. However, our work is still different from (Tan et al., 2019) in the following aspects: (1) Tan et al. (2019) mainly focused on constructing a unified NMT model for multi-lingual translation task, while we aim at how to effectively transfer out-of-domain translation knowledge to in-domain NMT model; (2) Our translation knowledge transfer is bidirectional, while the procedure of knowledge distillation in (Tan et al., 2019) is unidirectional; (3) When using knowledge distillation under our framework, we iteratively update teacher models for better domain adaptation. In contrast, all language-specific teacher NMT models in (Tan et al., 2019) remain fixed.

## 3 Iterative Dual Domain Adaptation Framework

In this section, we first detailedly describe our proposed framework for conventional one-to-one NMT domain adaptation, and then extend this framework to the scenario of multiple out-of-domain corpora (many-to-one).

### 3.1 One-to-one Domain Adaptation

As shown in Figure 1(a), previous studies mainly focus on the one-pass translation knowledge transfer from one out-of-domain NMT model to the in-domain NMT model. Unlike these studies, we propose to conduct iterative dual domain adaptation for NMT, of which framework is illustrated in Figure 1(b).

To better describe our framework, we summarize the training procedure of our framework

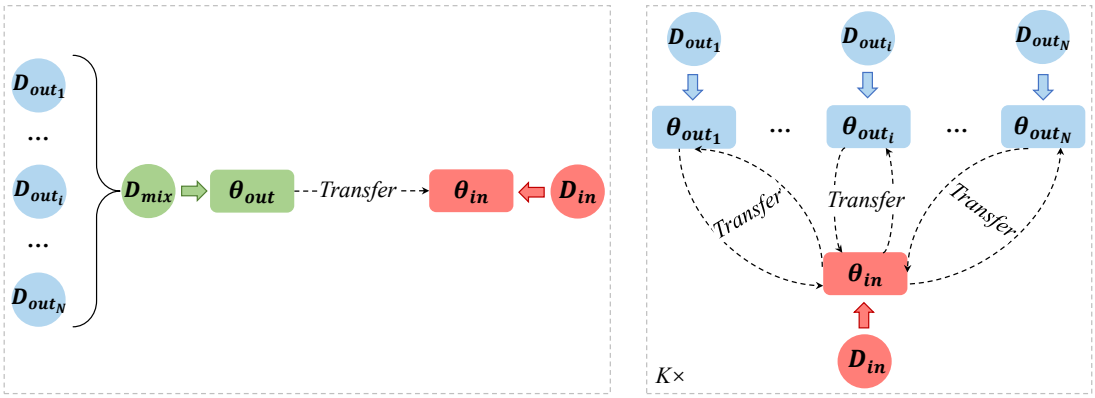


(c) Non-iterative Domain Adaptation (d) Iterative Dual Domain Adaptation

Figure 2: Traditional approach vs IDDA framework for many-to-one NMT domain adaptation. $D_{mix}$: a mixed out-of-domain training corpus.

in Algorithm 1. Specifically, we first individually train the initial in-domain and out-of-domain NMT models, respectively denoted by $\theta_{in}^{(0)}$ and $\theta_{out}^{(0)}$, via minimizing the negative likelihood of their own training corpora $D_{in}$ and $D_{out}$ (**Line 3**):

$$\mathcal{L}_{in}^{(0)} = \sum_{(\mathbf{x},\mathbf{y})\in D_{in}} -logP(\mathbf{y}|\mathbf{x};\theta_{in}^{(0)}), \quad (1)$$

$$\mathcal{L}_{out}^{(0)} = \sum_{(\mathbf{x},\mathbf{y})\in D_{out}} -logP(\mathbf{y}|\mathbf{x};\theta_{out}^{(0)}). \quad (2)$$

Then, we iteratively perform bidirectional translation knowledge transfer to update both in-domain and out-of-domain NMT models, until the maximal iteration number $K$ is reached (**Lines 5-14**). More specifically, at the $k$-th iteration, we first transfer the translation knowledge of the previous in-domain NMT model $\theta_{in}^{(k-1)}$ to the out-of-domain NMT model $\theta_{out}^{(k)}$ trained on $D_{out}$ (**Line 6**), and then reversely transfer the translation knowledge encoded by $\theta_{out}^{(k)}$ to the in-domain NMT model $\theta_{in}^{(k)}$ trained on $D_{in}$ (**Line 10**). During this process, we evaluate the new models $\theta_{in}^{(k)}$ and $\theta_{out}^{(k)}$ on their corresponding development sets, and then record the best model parameters as $\theta_{in}^{*}$ and $\theta_{out}^{*}$ (**Lines 7-9, 11-13**).

Obviously, during the above procedure, one of important steps is how to transfer the translation knowledge from one domain-specific NMT model to the other one. However, if we directly employ conventional domain transfer approaches, such as fine-tuning, as the iterative dual domain adaptation proceeds, the previously learned translation knowledge tends to be ignored. To deal with this issue, we introduce knowledge distillation (Kim and Rush, 2016) to conduct the translation knowledge transfer. Specifically, during the transfer process from $\theta_{out}^{(k)}$ to $\theta_{in}^{(k)}$, we first initialize $\theta_{in}^{(k)}$ with parameters of $\theta_{out}^{(k)}$, and then train $\theta_{in}^{(k)}$ not only to match the references of $D_{in}$, but also to be consistent with probability outputs of the previous best in-domain NMT model $\theta_{in}^{*}$, which is considered as the teacher model. To this end, we define the loss function as

$$\mathcal{L}_{in}^{(k)} = \sum_{(\mathbf{x},\mathbf{y})\in D_{in}} [-(1-\lambda)\cdot logP(\mathbf{y}|\mathbf{x};\theta_{in}^{(k)}) + \lambda \cdot \mathrm{KL}(P(\mathbf{y}|\mathbf{x};\theta_{in}^{(k)})||P(\mathbf{y}|\mathbf{x};\theta_{in}^{*}))], \quad (3)$$

where $\lambda$ is the coefficient used to trade off these two loss terms, and it can be tuned on the development set. Notably, when $\lambda$=0, only the term of likelihood function affects the model training, and thus our transfer approach degenerate into fine-tuning at each iteration.

In this way, we enable in-domain NMT model $\theta_{in}^{(k)}$ to not only retain the previously learned effective translation knowledge, but also fully absorb the useful translation knowledge from out-of-domain NMT model $\theta_{out}^{(k)}$. Similarly, we employ the above method to transfer translation knowledge from $\theta_{in}^{(k-1)}$ to $\theta_{out}^{(k)}$ using out-of-domain corpus $D_{out}$ and the previous best out-of-domain model $\theta_{out}^{*}$. Due to the space limitation, we omit the specific description of this procedure.

### 3.2 Many-to-one Domain Adaptation

Usually, in practical applications, there exist multiple available out-of-domain training corpora simultaneously. As shown in Figure 2(a), previous studies usually mix them into one out-of-domain

corpus, which is applicable for the conventional one-to-one NMT domain adaptation. However, various out-of-domain corpora are semantically related to in-domain corpus to different degrees, and thus intuitively, it is difficult to adequately play their roles without distinguishing them.

To address this issue, we extend the proposed framework to many-to-one NMT domain adaptation. Our extended framework is illustrated in Figure 2(b). Given an in-domain corpus and $N$ out-of-domain corpora, we first measure the semantic distance between each out-of-domain corpus and the in-domain corpus using the proxy $A$-distance $\hat{d}_A$=$2(1-2\epsilon)$ (Ganin et al., 2015; Pryzant et al., 2017), where the $\epsilon$ is the generalization error of a linear bag-of-words SVM classifier trained to discriminate between the two domains. Then, we determine the transfer order of these out-of-domain NMT models as $\{\theta_{out_1}, \theta_{out_2}, ...\theta_{out_N}\}$ , according to distances of their own training corpora to the in-domain corpus in a decreasing order. The reason behind this step is the translation knowledge of previously transferred out-of-domain NMT models will be partially forgotten during the continuous transfer. By setting transfer order according to their $\hat{d}_A$ values in a decreasing order, we enable the in-domain NMT model to fully preserve the translation knowledge transferred from the most relevant out-of-domain NMT model. Finally, we sequentially perform bidirectional knowledge transfer between the in-domain and each out-of-domain models, where this process will be repeated for $K$ iterations.

## 4 Experiments

To verify the effectiveness of our framework, we first conducted one-to-one domain adaptation experiments on Chinese-English translation, where we further investigated impacts of various factors on our framework. Then, we carried out two-to-one domain adaptation experiments on English-German translation, so as to demonstrate the generality of our framework on different language pairs and multiple out-of-domain corpora.

### 4.1 Setup

**Datasets.** In the Chinese-English translation task, our in-domain training corpus is from IWSLT2015 dataset consisting of 210K *TED Talk* sentence pairs, and the out-of-domain training corpus contains 1.12M LDC sentence pairs related to *News* domain. For these two domains, we chose IWSLT dev2010 and NIST 2002 dataset as development sets. Finally, we used IWSLT tst2010, tst2011 and tst2012 as in-domain test sets. Particularly, in order to verify whether our framework can enable NMT models of two domains to benefit each other, we also tested the performance of out-domain NMT model on NIST 2003, 2004, 2005, 2006 datasets.

For the English-German translation task, our training corpora totally include one in-domain dataset: 200K *TED Talk* sentence pairs provided by IWSLT2015, and two out-of-domain datasets: 500K sentence pairs (*News* topic) extracted from WMT2014 corpus, and 500K sentence pairs (*Medical topic*) that are sampled from OPUS EMEA corpus[2]. As for development sets, we chose IWSLT tst2012, WMT tst2012 and 1K sampled sentence pairs of OPUS EMEA corpus, respectively. In addition, IWSLT tst2013, tst2014 were used as in-domain test sets, WMT news-test2014 (News topic) and 1K sampled sentence pairs of OPUS EMEA corpus were used as two out-of-domain test sets.

We first employed *Stanford Segmenter*[3] to conduct word segmentation on Chinese sentences and *MOSES script*[4] to tokenize English and German sentences. Then, we limited the length of sentences to 50 words in the training stage. Besides, we employed *Byte Pair Encoding* (Sennrich et al., 2016) to split words into subwords and set the vocabulary size for both Chinese-English and English-German as 32,000. We evaluated the translation quality with BLEU scores (Papineni et al., 2002) as calculated by `multi-bleu.perl` script .

**Settings.** We chose Transformer (Vaswani et al., 2017) as our NMT model, which exhibits excellent performance due to its flexibility in parallel computation and long-range dependency modeling. We followed Vaswani et al. (2017) to set the configurations. The dimensionality of all input and output layers is 512, and that of FFN layer is 2048. We employed 8 parallel attention heads in both encoder and decoder. Parameter optimization was performed using stochastic gradient descent, where *Adam* (Kingma and Ba, 2015) was used to automatically adjust the learning rate of

[2]http://opus.nlpl.eu/
[3]https://nlp.stanford.edu/
[4]http://www.statmt.org/moses/

each parameter. We batched sentence pairs by approximated length, and limited input and output tokens per batch to 25000 tokens. As for decoding we employed beam search algorithm and set the beam size as 4. Besides, we set the distillation coefficient $\lambda$ as 0.4.

**Contrast Models.** We compared our framework with the following models, namely:

- **Single** A reimplemented Transformer only trained on a single domain-specific (in/out) training corpus.
- **Mix** A reimplemented Transformer trained on the mix of in-domain and out-of-domain training corpora.
- **Fine-tuning (FT)** (Luong and Manning, 2015). It first trains the NMT model on out-of-domain training corpus and then fine-tunes it using in-domain training corpus.
- **Mixed Fine-tuning (MFT)** (Chu et al., 2017). It also first trains the NMT model on out-of-domain training corpus, and then fine-tunes it using both out-of-domain and over-sampling in-domain training corpora.
- **Knowledge Distillation (KD)** (Kim and Rush, 2016). Using this method, we first train a out-of-domain and an in-domain NMT models using their own training corpus, respectively. Then, we use the in-domain training corpus to fine-tune the out-of-domain NMT model, supervised by the in-domain NMT model.

Besides, we reported the performance of some recently proposed multi-domain NMT models.

- **Domain Control (DC)** (Kobus et al., 2016). It is also based on the mix-domain NMT model. However, it adds an additional domain tag to each source sentence, incorporating domain information into source annotations.
- **Discriminative Mixing (DM)** (Pryzant et al., 2017). It jointly trains NMT with domain classification via multitask learning. Please note that it performs the best among three approaches proposed by Pryzant et al., (2017).
- **Word-level Domain Context Discrimination (WDCD)** (Zeng et al., 2018). It discriminates the source-side word-level domain specific and domain-shared contexts for multi-domain NMT by jointly modeling NMT and domain classifications.

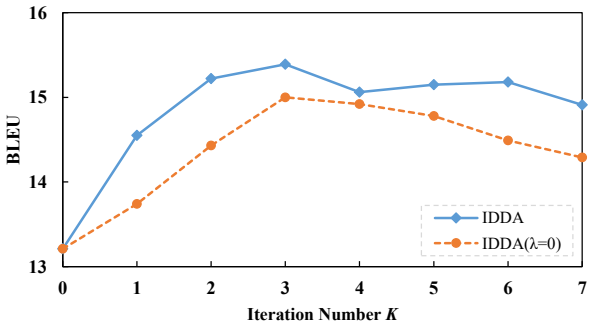


Figure 3: Effect of iteration number ($K$) on the Chinese-English in-domain development set.

## 4.2 Results on Chinese-English Translation

### 4.2.1 Effect of Iteration Number $K$

The iteration number $K$ is a crucial hyper-parameter that directly determines the amount of the transferred translation knowledge under our framework. Therefore, we first inspected its impacts on the development sets. To this end, we varied $K$ from 0 to 7 with an increment of 1 in each step, where our framework degrades to *Single* when $K$=0.

Figure 3 provides the experimental results using different $K$s. We can observe that both *IDDA($\lambda$=0)* and *IDDA* achieve the best performance at the 3-th iteration, respectively. Therefore, we directly used $K$=3 in all subsequent experiments.

### 4.2.2 Overall Performance

Table 1 shows the overall experimental results. On all test sets, our framework significantly outperforms other contrast models. Furthermore, we reach the following conclusions:

First, on the in-domain test sets, both *IDDA($\lambda$=0)* and *IDDA* surpass *Single*, *Mix*, *FT*, *MFT* and *KD*, most of which are commonly used in the domain adaptation for NMT. This confirms the difficulty in completely one-pass transferring the useful out-of-domain translation knowledge to the in-domain NMT model. Moreover, the in-domain NMT model benefits from multiple-pass knowledge transfers under our framework.

Second, compared with *DC*, *DM* and *WDCD* that are proposed for multi-domain NMT, both *IDDA($\lambda$=0)* and *IDDA* still exhibit better performance on the in-domain test sets. The underlying

| Model | TED Talk (**In-domain**) | | | | | News (**Out-of-domain**) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Tst10 | Tst11 | Tst12 | Tst13 | AVE. | Nist03 | Nist04 | Nist05 | Nist06 | AVE. |
| *Cross-domain Transfer Methods* | | | | | | | | | | |
| Single | 15.82 | 20.80 | 17.77 | 18.33 | 18.18 | 45.38 | 45.93 | 42.80 | 42.70 | 44.20 |
| Mix | 16.46 | 20.85 | 19.13 | 19.87 | 19.08 | 44.87 | 45.71 | 42.24 | 42.02 | 43.71 |
| FT | 16.77 | 21.16 | 19.31 | 20.53 | 19.44 | — | — | — | — | — |
| MFT | 17.19 | 22.02 | 20.09 | 21.05 | 20.08 | — | — | — | — | — |
| KD | 17.62 | 21.88 | 19.97 | 20.43 | 19.98 | — | — | — | — | — |
| *Multi-domain NMT Methods* | | | | | | | | | | |
| DC | 17.23 | 22.10 | 19.68 | 20.58 | 19.90 | 46.03 | 46.62 | 44.39 | 43.82 | 45.21 |
| DM | 16.45 | 21.35 | 18.77 | 20.27 | 19.21 | 45.12 | 45.83 | 42.77 | 42.59 | 44.08 |
| WDCD | 17.32 | 22.23 | 20.02 | 21.10 | 20.17 | 46.33 | 46.36 | 44.62 | 43.80 | 45.27 |
| *IDDA Framework* | | | | | | | | | | |
| IDDA($\lambda$=0) | 18.00 | 22.71 | 20.36 | **21.82** | 20.72 | 45.91 | 45.84 | 43.61 | 42.17 | 44.46 |
| IDDA | **18.36** | **23.14** | **20.78** | 21.79 | **21.02**$^{\dagger}$ | **47.17** | **47.44** | **45.38** | **44.04** | **46.01**$^{\dagger}$ |

Table 1: Experimental results on the Chinese-English translation task. † indicates statistically significantly better than ($\rho$ <0.01) the result of *WDCD*.

reason is that these multi-domain models discriminate domain-specific and domain-shared information in encoder, however, their shared decoder are inadequate to effectively preserve domain-related text style and idioms. In contrast, our framework is adept at preserving these information since we construct an individual NMT model for each domain.

Third, *IDDA* achieves better performance than *IDDA($\lambda$=0)*, demonstrating the importance of retaining previously learned translation knowledge. Surprisingly, *IDDA* significantly outperforms *IDDA($\lambda$=0)* on out-of-domain data sets. We conjecture that during the process of knowledge distillation, by assigning non-zero probabilities to multiple words, the output distribution of teacher model is more smooth, leading to smaller variance in gradients (Hinton et al., 2015). Consequently, the out-of-domain NMT model becomes more robust by iteratively absorbing the translation knowledge from the best out-of-domain model.

Finally, note that even on the out-of-domain test sets, *IDDA* still has better performance than all listed contrast models in the subsequent experimental analyses. This result demonstrates the advantage of dual domain adaptation under our framework.

According to the reported performance of our framework shown in Table 1, we only considered *IDDA* in all subsequent experiments. Besides, we only chose *MFT*, *KD*, and *WDCD* as typical contrast models. This is because *KD* is the basic domain adaption approach of our framework, *MFT* and *WDCD* are the best domain adaptation method and multi-domain NMT model for comparison, respectively.

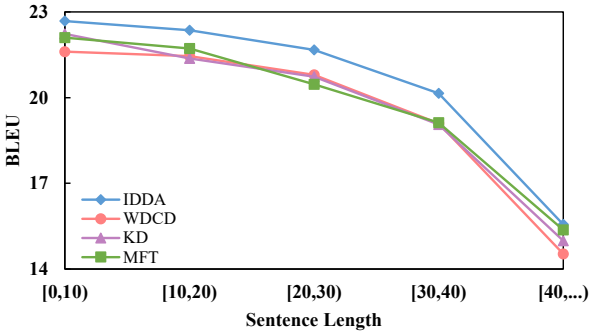


Figure 4: BLEU scores on different IWSLT test sets divided according to source sentence lengths.

#### 4.2.3 Results on Source Sentences with Different Lengths

Following previous work (Bahdanau et al., 2015), we divided IWSLT test sets into different groups based on the lengths of source sentences and then investigated the performance of various models.

Figure 4 illustrates the results. We observe that our framework also achieves the best performance in all groups, although the performances of all models degrade with the increase of the length of source sentences.

#### 4.2.4 Effect of Out-of-domain Corpus Size

In this group of experiments, we investigated the impacts of out-of-domain corpus size on our proposed framework. Specifically, we inspected the results of our framework using different sizes of out-of-domain corpora: 50K, 200K and 1.12M, respectively

Figure 5 shows the comparison results on the average BLEU scores of all IWSLT test sets. No matter how large out-of-domain data is used, *IDDA* always achieves better performance than

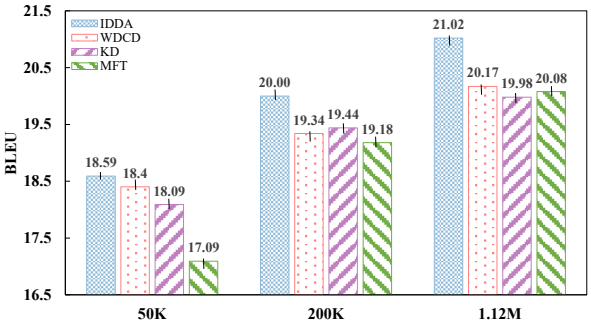


Figure 5: Experimental results with different sizes of out-of-domain corpora.

| Model | AVE. |
|---|---|
| IDDA-unidir | 20.43 |
| IDDA-fixTea | 20.60 |
| IDDA | 21.02 |

Table 2: Experimental results of comparing IDDA with its two variants.

other contrast models, demonstrating the effectiveness and generality of our framework. Specially, *IDDA* with 200K out-of-domain corpus is comparable to *KD* with 1.12M corpus. From this result, we confirm again that our framework is able to better exploit the complementary information between domains than *KD*.

#### 4.2.5 Effects of Dual Domain Adaptation and Updating Teacher Models

Two highlights of our framework consist of the usage of bidirectional translation knowledge transfer and continuous updating teacher models $\theta^*_{out}$ and $\theta^*_{in}$ (See Line 6, 10 of Algorithm 1). To inspect their effects on our framework, we compared our framework with its two variants: (1) *IDDA-unidir*, where we only iteratively transfer out-of-domain translation knowledge to the in-domain NMT model; (2) *IDDA-fixTea*, where teacher models are fixed as the initial out-of-domain and in-domain NMT models, respectively.

The results are displayed in Table 2. We can see that our framework exhibits better performance than its two variants, which demonstrates that dual domain adaptation enables NMT models of two domains to benefit from each other, and updating teacher models is more helpful to retain useful translation knowledge.

#### 4.2.6 Case Study

Table 3 displays the 1-best translations of a sampled test sentence generated by *MFT*, *KD*, *WDCD*, and *IDDA* at different iterations. Inspecting this

| Model | Translation |
|---|---|
| Src | 这(zhè) 是(shì) 第(dì) 一(yī) 种(zhǒng)<br>直立(zhílì) 行走(xíngzǒu) 的(de)<br>灵长类(língzhǎnglèi) 动物(dòngwù) |
| Ref | that was the first upright primate |
| MFT | this is the first *animal* to walk upright |
| KD | this is the first *growing* primate |
| WDCD | this is the first primate *walking around* |
| IDDA-1 | this is the first *upright - walking primate* |
| IDDA-2 | this is the first *upright - walking primate* |
| IDDA-3 | this is the first *primates walking upright* |
| IDDA-4 | this is the first *upright primate* |
| IDDA-5 | this is the first *upright primate* |
| IDDA-6 | this is the first *upright primate* |
| IDDA-7 | this is the first *upright primate* |

Table 3: Translation examples of different NMT models. **Src**: source sentence and **Ref**: target reference. *IDDA-k* represents the in-domain NMT model at the $k$-th iteration using our framework.

example provides the insight into the advantage of our proposed framework to some extent. Specifically, we observe that *MFT*, *KD*, *WDCD* are unable to correctly understand the meaning of "zhílì xíngzǒu de língzhǎnglèi dòngwù" and thus generate incorrect or incomplete translations, while *IDDA* successfully corrects these errors by gradually absorbing transferred translation knowledge.

### 4.3 Results on English-German Translation

#### 4.3.1 Overall Performance

We first calculated the distance between the in-domain and each out-of-domain corpora: $\hat{d}_A$(*Ted Talk*, *News*) = 0.92 and $\hat{d}_A$(*Ted Talk*, *Medical*) = 1.92. Obviously, the *News* domain is more relevant to *TED Talk* domain than *Medical* domain, and thus we determined the final transfer order as $\{\theta_{\text{out}_{\text{medical}}}, \theta_{\text{out}_{\text{news}}}\}$ for this task. Then, as implemented in the previous Chinese-English experiments, we determined the optimal $K$=2 on the development set.

Table 4 shows experimental results. Similar to the previously reported experiment results, our framework still obtains the best performance among all models, which verifies the effectiveness of our framework on many-to-one domain adaptation for NMT.

As described above, we have two careful designs for many-to-one NMT domain adaptation: (1) We distinguish different out-of-domain corpora, and then iteratively perform bidirectional translation knowledge transfer between in-domain and each out-of-domain NMT models. (2) We determine the transfer order according to the seman-

| Model | In-domain TED Talk | | | Out-of-domain1 News | Out-of-domain2 Medical |
|---|---|---|---|---|---|
| | IWSLT2013 | IWSLT2014 | AVE. | WMT14 | EMEA |
| *Cross-domain Transfer Methods* | | | | | |
| Single | 29.76 | 25.99 | 27.88 | 20.54 | 51.11 |
| Mix | 31.45 | 27.03 | 29.24 | 21.17 | 50.60 |
| FT | 30.54 | 27.02 | 28.78 | — | — |
| MFT | 31.86 | 27.49 | 29.67 | — | — |
| KD | 31.33 | 27.96 | 29.64 | — | — |
| *Multi-domain NMT Methods* | | | | | |
| DC | 31.13 | 28.02 | 29.57 | 21.61 | 52.25 |
| DM | 31.57 | 27.60 | 29.58 | 21.75 | 52.60 |
| WDCD | 31.87 | 27.82 | 29.84 | 21.86 | 52.84 |
| *IDDA Framework* | | | | | |
| IDDA($\lambda$=0) | 32.11 | 28.10 | 30.11 | 22.01 | 52.07 |
| IDDA | **32.93** | **28.88** | **30.91**$^{\dagger}$ | **22.17**$^{*}$ | **53.39**$^{\dagger}$ |

Table 4: Experimental results of the English-German translation task. * indicates statistically significantly better than ($\rho$ <0.05) the result of *WDCD*.

| Model | Transfer Order | AVE. |
|---|---|---|
| IDDA-mix | —— | 30.17 |
| IDDA | $\{\theta_{\text{out}_{\text{news}}}, \theta_{\text{out}_{\text{medical}}}\}$ | 30.51 |
| IDDA | $\{\theta_{\text{out}_{\text{medical}}}, \theta_{\text{out}_{\text{news}}}\}$ | 30.91 |

Table 5: Experimental results of IDDA using different configurations.

tic distance between each out-of-domain and in-domain training corpora. Here, we carried out two groups of experiments to investigate their impacts on our framework. In the first group of experiments, we first combined all out-of-domain training corpora into a mixed corpus, and then applied our framework to establish the in-domain NMT model. In the second group of experiments, we employed our framework in different transfer orders to perform domain adaptation.

Table 5 shows the final experimental results, which are in line with our expectations and verify the validity of our designs.

## 5 Conclusion

In this paper, we have proposed an iterative dual domain adaptation framework for NMT, which continuously fully exploits the mutual complementarity between in-domain and out-domain corpora for translation knowledge transfer. Experimental results and in-depth analyses on translation tasks of two language pairs strongly demonstrate the effectiveness of our framework.

In the future, we plan to extend our framework to multi-domain NMT. Besides, how to leverage monolingual sentences of different domains to refine our proposed framework. Finally, we will apply our framework into other translation models (Bahdanau et al., 2015; Su et al., 2018; Song et al., 2019), so as to verify the generality of our framework.

## Acknowledgments

The authors were supported by National Natural Science Foundation of China (No. 61672440), Beijing Advanced Innovation Center for Language Resources, NSF Award (No. 1704337), the Fundamental Research Funds for the Central Universities (Grant No. ZK1024), and Scientific Research Project of National Language Committee of China (Grant No. YB135-49). We also thank the reviewers for their insightful comments

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR 2015*.

Ankur Bapna and Orhan Firat. 2019. Non-parametric adaptation for neural machine translation. In *Proc. of NAACL 2019*.

Boxing Chen, Colin Cherry, George Foster, and Samuel Larkin. 2017. Cost weighting for neural machine translation domain adaptation. In *Proc. of WMT 2018*.

Chenhui Chu, Raj Dabre, and Sadao Kurohashi. 2017. An empirical comparison of domain adaptation methods for neural machine translation.

Chenhui Chu and Rui Wang. 2018. A survey of domain adaptation for neural machine translation. In *Proc. of COLING 2018*.

M. Amin Farajian, Marco Turchi, Matteo Negri, and Marcello Federico. 2017. Multi-domain neural machine translation through unsupervised adaptation. In *Proc. of WMT 2017*.

Markus Freitag and Yaser Al-Onaizan. 2016. Fast domain adaptation for neural machine translation. *CoRR abs/1612.06897*.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor S. Lempitsky. 2015. Domain-adversarial training of neural networks. *Machine Learning Research*, 17.

Shuhao Gu, Yang Feng, and Qun Liu. 2019. Improving domain adaptation translation with domain invariant and specific information. In *Proc. of NAACL 2019*.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In *Proc. of NIPS 2016*.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR abs/1503.02531*.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proc. of EMNLP 2016*.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. In *Proc. of ICLR 2015*.

Catherine Kobus, Josep Crego, and Jean Senellart. 2016. Domain control for neural machine translation. *CoRR abs/1612.06140*.

Minh-Thang Luong and Christopher D Manning. 2015. Stanford neural machine translation systems for spoken language domains. In *Proc. of IWSLT 2015*.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.*, 22(10).

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proc. of ACL 2002*.

Reid Pryzant, Denny Britz, and Q Le. 2017. Effective domain mixing for neural machine translation. In *Proc. of WMT 2017*.

Hassan Sajjad, Nadir Durrani, Fahim Dalvi, Yonatan Belinkov, and Stephan Vogel. 2017. Neural machine translation training in a multi-domain scenario. *CoRR abs/1708.08712*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proc. of ACL 2016*.

Christophe Servan, Josep Crego, and Jean Senellart. 2016. Domain specialization: a post-training domain adaptation for neural machine translation. *CoRR abs/1612.06141*.

Linfeng Song, Daniel Gildea, Yue Zhang, Zhiguo Wang, and Jinsong Su. 2019. Semantic neural machine translation using AMR. *TACL 2019*, 7.

Jinsong Su, Jiali Zeng, Deyi Xiong, Yang Liu, Mingxuan Wang, and Jun Xie. 2018. A hierarchy-to-sequence attentional neural machine translation model. *IEEE/ACM TALSP 2018*, 26(3).

Xu Tan, Yi Ren, Di He, Tao Qin, Zhou Zhao, and Tie-Yan Liu. 2019. Multilingual neural machine translation with knowledge distillation. In *Proc. of ICLR 2019*.

Sander Tars and Mark Fishel. 2018. Multi-domain neural machine translation. *CoRR abs/1805.02282*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proc. of NIPS 2017*.

David Vilar. 2018. Learning hidden unit contribution for adapting neural machine translation models. In *Proc. of NAACL 2018*, pages 500–505.

Rui Wang, Andrew Finch, Masao Utiyama, and Eiichiro Sumita. 2017a. Sentence embedding for neural machine translation domain adaptation. In *Proc. of ACL 2017*.

Rui Wang, Masao Utiyama, Lemao Liu, Kehai Chen, and Eiichiro Sumita. 2017b. Instance weighting for neural machine translation domain adaptation. In *Proc. of EMNLP 2017*.

Yijun Wang, Yingce Xia, Li Zhao, Jiang Bian, Tao Qin, Guiquan Liu, and Tie-Yan Liu. 2018. Dual transfer learning for neural machine translation with marginal distribution regularization. In *Proc. of AAAI 2018*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR abs/1609.08144*.

Jiali Zeng, Jinsong Su, Huating Wen, Yang Liu, Jun Xie, Yongjing Yin, and Jianqiang Zhao. 2018. Multi-domain neural machine translation with word-level domain context discrimination. In *Proc. of EMNLP 2018*.

Xuan Zhang, Pamela Shapiro, Gaurav Kumar, Paul McNamee, Marine Carpuat, and Kevin Duh. 2019a. Curriculum learning for domain adaptation in neural machine translation. In *Proc. of NAACL 2019*.

Zhirui Zhang, Shuangzhi Wu, Shujie Liu, Mu Li, Ming Zhou, and Enhong Chen. 2019b. Regularizing neural machine translation by target-bidirectional agreement.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. *Proc. of EMNLP 2016*.

# Multi-agent Learning for Neural Machine Translation

**Tianchi Bi, Hao Xiong, Zhongjun He, Hua Wu and Haifeng Wang**
Baidu Inc. No. 10, Shangdi 10th Street
Beijing, 100085, China
{bitianchi, xionghao05, hezhongjun, wu_hua, wanghaifeng}@baidu.com

## Abstract

Conventional Neural Machine Translation (NMT) models benefit from the training with an additional agent, e.g., dual learning, and bidirectional decoding with one agent decoding from left to right and the other decoding in the opposite direction. In this paper, we extend the training framework to the multi-agent scenario by introducing diverse agents in an interactive updating process. At training time, each agent learns advanced knowledge from others, and they work together to improve translation quality. Experimental results on NIST Chinese-English, IWSLT 2014 German-English, WMT 2014 English-German and large-scale Chinese-English translation tasks indicate that our approach achieves absolute improvements over the strong baseline systems and shows competitive performance on all tasks.

## 1 Introduction

Training with more than one agents has attracted intensive research interest in recent years, for example, dual learning (He et al., 2016; Xia et al., 2017, 2018) and bidirectional decoding (Liu et al., 2016; Zhang et al., 2019b). The former method leverages the duality between the two related agents as the feedback signal to regularize training, while the latter targets the agreement between one agent decoding from left to right (L2R) while the other decoding in opposite direction (R2L). Both methods enhance the translation models by introducing a regularization term into the training objective.

The effectiveness of these two methods lies in the fact that appropriate regularization can help each agent to learn from the superior models while integrating their advantages (e.g., good translations for prefixes for L2R, and good translation quality for suffixes for R2L). As shown in the Table 1, due to the *exposure bias* problem (Ranzato et al., 2016), the agent trained to decode from left to right tends to generate better prefixes and bad suffixes, and the agent decoding in the reverse direction demonstrates the opposite preference. By introducing additional Kullback-Leibler (KL) divergences between the probability distributions defined by L2R and R2L models into the NMT training objective (Zhang et al., 2019b), it is possible for two models to learn advantages from each other.

| | |
|---|---|
| *Src* | 此次 豪雨 灾情 , 半数 遇难 者 遭 洪水 冲走 溺 毙 ; 其他 则 因 房屋 倒塌 或 电线 走火 触电 致死 。 |
| *Ref* | In this disaster caused by torrential rains, half of the victims were carried away and drowned in the floods; the others died because their houses collapsed or from electrocution caused by short circuits. |
| *L2R* | In the torrential rain, half of the victims were washed away by floods, while others ***died of electricity caused by collapsed houses or electric wires.*** |
| *R2L* | ***Half of the victims were drowned by floods,*** while others were killed by collapsed houses or electrocuted by sparkling wires. |

Table 1: In this sample, NMT systems with different implementations present diverse translation errors for one long sentence. The agent decoding from left to right (*L2R*) tends to generate better prefixes and bad suffixes (bold letters), and the *R2L* agent has the opposite preference.

According to the empirical achievements of previous studies on training with two agents, it is natural to consider the training with more than two agents, and to extend our study to the multi-agent scenario. However, training with more than two agents is more complex, and we face two critical



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 856–865,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

problems. First, when deploying the multi-agent system, should we focus on the diversity or the strength of each agent? (Wong et al., 2005; Marcolino et al., 2013) Second, learning in multi-agent scenario is *many-to-many*, as opposed to the relatively simpler *one-to-one* learning in two-agent training, and requires an effective learning strategy.

There have been many alternatives to improve the diversity of models even based on the Transformer model (Vaswani et al., 2017). For example, decoding in the opposite direction usually results in different preferences: good prefixes and bad prefixes (Zhang et al., 2019b). Rather, self-attention with relative position representations enhances the generalization to sequence lengths unseen during training (Shaw et al., 2018). Furthermore, increasing the size of layers in the encoder is expected to specialize in word sense disambiguation (Tang et al., 2018; Domhan, 2018). In this paper, we investigate the effects of two teams of multi-agents: a team of alternative agents mentioned above, and a uniform team of different initialization for the same model.

To resolve the second problem, we simplify the *many-to-many* learning to the *one-to-many* (one teacher $vs.$ many students) learning, extending ensemble knowledge distillation (Fukuda et al., 2017; Freitag et al., 2017; Liu et al., 2018; Zhu et al., 2018). During the training, each agent performs better by learning from the ensemble model (*Teacher*) of all agents integrating the knowledge distillation (Hinton et al., 2015; Kim and Rush, 2016) into the training objective. This procedure can be viewed as the introduction of an additional regularization term into the training objective, with which each agent can learn advantages from the ensemble model gradually. With this method, each agent is optimized not only to the maximization of the likelihood of the training data, but also to the minimization of the divergence between its own model and the ensemble model.

However, the above learning strategy converges on the local optimum rapidly in our empirical studies. It seems that each agent tends to focus on learning from the ensemble model while completely ignoring its own exploration. To alleviate this problem, we train each agent to learn from the ensemble models when necessary, and to distill the knowledge based on the translation quality (BLEU score) of the ensemble model. This means we evaluate the quality of the ensemble model to see whether it is good enough to be studied. Consequently, the knowledge distillation from the ensemble model to the agent stems from the better translation generated by the ensemble model.

We evaluate our model on NIST Chinese-English Translation Task, IWSLT 2014 German-English Translation task, large-scale WMT 2014 English-German Translation task and large-scale Chinese-English Translation Task. Extensive experimental results indicate that our model significantly improves the translation quality, compared to the strong baseline systems. Moreover, our model also reports competitive performance on the German-English and English-German translation tasks, achieving 36.27 and 29.67 BLEU scores, respectively.

To the best of our knowledge, this is the first work on training NMT model with more than two agents [1]. The contributions of this paper are summarized as follows:

- We extend the study on training with two agents to the multi-agent scenario, and propose a general learning strategy to train multiple agents.
- We investigate the effects of the diversity and the strength of each agent in the multi-agent training scenario.
- We simplify complex *many-to-many* learning in multi-agent learning to *one-to-many* learning by forcing each agent learning knowledge from ensemble model as necessary.
- Extensive experiments on multiple translation tasks confirm that our model significantly improves the translation quality and reports the new state-of-the-art results on IWSLT 2014 German-English and competitive results on WMT 2014 English-German translation tasks.

## 2 Background

Conventional autoregressive NMT models (Bahdanau et al., 2015; Sutskever et al., 2014) decode target tokens sequentially, which indicates that the

[1] Although Wang et al. (2019) proposed a work named multi-agent dual learning contemporarily, the training in their work is conducted on the two agents with fixed parameters of other agents.

determination of the current token is conditioned by the previous generated sequence. Formally, at time step $t$, the generation of the current token $y_t$ is determined by the following equation:

$$p(y_t) = p(y_t|y_{<t}, x; \theta) \quad (1)$$

where $y_{<t}$ represents the previously generated sequence, and $\theta$ are the parameters of the representation of the source sequence and the partial target sequence.

Furthermore, the usual training criterion is to minimize the negative log-likelihood for each sample from the training data,

$$\mathcal{L}_{NLL} = -\sum_{t=0}^{T}\sum_{k=1}^{|\mathcal{V}|} \mathbb{1}\{y_t = k\} \log p(y_t = k|y_{<t}, x; \theta) \quad (2)$$

where $T$ is the length of the target sequence, $|\mathcal{V}|$ is the size of vocabulary and $\mathbb{1}\{\cdot\}$ is the indicator function. This objective can be viewed as minimizing the cross-entropy between the correct translation $y^*$ and the model generation $p(y_t|y_{<t}, x; \theta)$.

## 3 Multi-agent Learning

Empirical studies indicate that one agent is trained to perform better through learning advantages from the other agent in the two-agent scenario, namely *one-to-one* learning (Figure 1.(a)). The training objective of the agent is regularized, which is to learn better models by leveraging the relationship between the two agents as feedback, i.e., duality in dual learning problem, and agreement in bidirectional decoding.

Extending the study to the multi-agent scenario is feasible and desirable, as multiple agents might supply more reliable and diverse advantages compared to the two-agents scenario. However, the agent is expected to learn advantages from each other in the multi-agent scenario, which results in a complex *many-to-many* learning problem(Figure 1.(b)).

Instead of tackling *many-to-many* learning, we force the agent to learn from a common *Teacher* by introducing ensemble knowledge distillation, thus reduce the learning to *one-to-many* (Figure 1.(c)). With this learning strategy, each agent can learn to improve the performance in an interactive updating process.

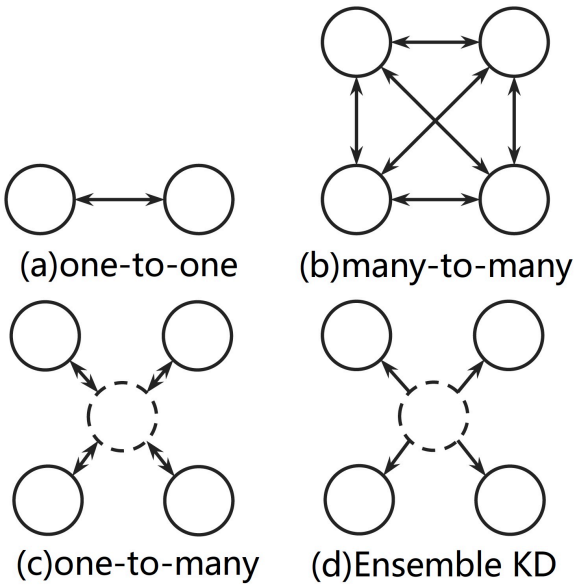


Figure 1: Illustration for different learning approaches.

As opposed to ensemble Knowledge Distillation (KD) (Figure 1.(d)), the important difference is that in the knowledge distillation of an ensemble of models, the *Teacher* network is fixed after pre-training, during the training process. While in our framework, the state of the *Teacher* network is updated at each iteration, and its performance can be further improved by the improvements of each agent explicitly, in an interactive updating process. In some ways the ensemble KD can be viewed as a particular case of our model, as we fix the update of the *Teacher* network in the training framework.

### 3.1 Overall Framework

For the sake of simplicity, four variant agents are referred to as $Agent1$, $Agent2$, $Agent3$ and $Agent4$. We begin by pre-training each agent independently (Figure 2.(a)), and then enhance the model in the multi-agent scenario in two steps: 1) Generating Ensemble Model (Figure 2.(b)), and 2) *One-to-Many* Learning (Figure 2.(c)). The performance of each agent is improved in an interactive updating process, through repeating the above two steps.

### 3.2 Generate Ensemble Model

As pointed out in the work of Liu et al. (2018), ensemble models can empirically alleviate problems existing in the standard NMT model, such as ambiguities in the training data, and discrepancy between training and testing.

According to the practical advantages of ensem-

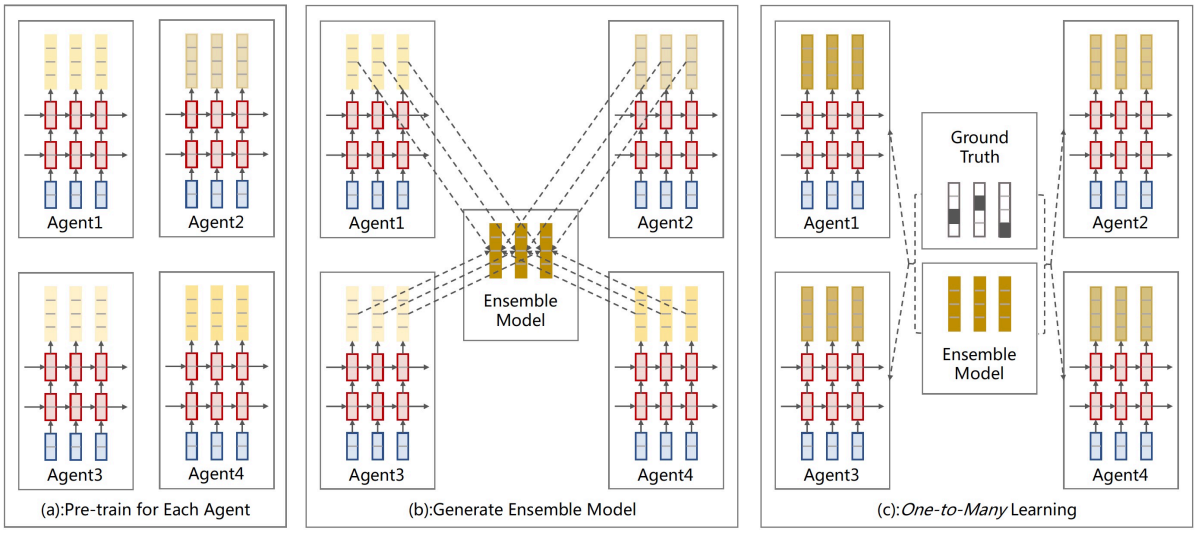


Figure 2: In this example, four agents decode the similar sentence with different model capacity. (a): At first, each agent is pre-trained to generate the translation independently. (b) The ensemble model is generated by the average prediction from each agent. (c): The *One-to-Many* learning distills the knowledge from the ensemble model to each agent as necessary. The performance of each agent is improved explicitly in an interactive updating process, through repeating the process (b) and (c).

ble models, it is relevant to force agents to learn from the ensemble model, instead of learning from each other separately. Following previous work, we develop our ensemble model by averaging the model distributions of all agents.

Formally, the model distribution of the $i$-th agent $agent_i$ is defined as $p(y_t^i|y_{<t}^i, x; \theta)$. Assume we have $\mathcal{N}$ agents, the model distribution of the ensemble model can be formulated as:

$$q(y_t|y_{<t}, x; \theta_t) = \frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} p(y_t^i|y_{<t}^i, x; \theta_i) \quad (3)$$

where $\theta_t$ are parameters for representing the ensemble model. Notably, we do not train it in the training process.

In the above formula, the probability $q(y_t|y_{<t}, x; \theta_t)$ is one reliable estimator of the model distribution, as the majority of the agents are likely to generate correct sequence. From this perspective, we expect that more agents will lead to better and more robust performance.

### 3.3 *One-to-Many* Learning

In the *one-to-many* learning framework, the ensemble model acts as the *Teacher* network, which distills knowledge to each agent iteratively.

Rather than minimizing cross-entropy with the observed data, we minimize the cross-entropy with the probability distribution from the ensemble model for each agent.

$$\mathcal{L}_{KD}^i = -\sum_{t=0}^{T}\sum_{k=1}^{|\mathcal{V}|} q(y_t = k|y_{<t}, x; \theta_t) \times \log p(y_t^i = k|y_{<t}^i, x; \theta) \quad (4)$$

With the above formula, the agent is optimized to the minimization of the model divergence between its own model and the ensemble model.

However, integrating the above regularization term into the training objective straightforwardly is problematic in practice. The agent tends to focus on learning from the ensemble model rather than exploring its own prediction to converge rapidly. Consequently, the model converges at a suboptimal point, and fails to enhance the performance by learning from each other due to the lack of evaluation of the ensemble model.

To alleviate this problem, we train each agent to learn from the ensemble model when necessary, distilling the knowledge conditioned by the translation quality of the ensemble model. We evaluate the quality of the ensemble model to see whether it is good enough to be studied. The knowledge distillation from the ensemble model to the agent comes from the better translation generated by the ensemble model. Otherwise, the agent is forced to learn its own distribution.

Let $< X_g, Y_g >$ be one sentence pair in the training corpus, the quality of the translation sequence generated by the ensemble model is mea-

sured by the BLEU score,

$$score(Y_t) = \text{BLEU}(Y_t, Y_g) \tag{5}$$

where $Y_t$ is generated by the model distribution $q(y_t|y_{<t}, x; \theta_t)$. The quality of the translation sequence generated by each agent can also be defined using the similar metric,

$$score(Y_s^i) = \text{BLEU}(Y_s^i, Y_g) \tag{6}$$

where $Y_s^i$ is generated by the model distribution $p(y_t^i|y_{<t}^i, x; \theta)$ of each agent.

Conditioned by the translation quality of the ensemble model, we modify the training objective for each agent as follows:

$$\mathcal{L}_{KD}^i = -\sum_{t=0}^{T}\sum_{k=1}^{|\mathcal{V}|} S \times \log p(y_t^i = k|y_{<t}^i, x; \theta) \tag{7}$$

where $S$ is defined as:

$$S = \begin{cases} q(y_t = k|y_{<t}, x; \theta_t) & score(Y_t) > score(Y_s^i) \\ \mathbb{1}_s\{y_t^i = k\} & otherwise \end{cases} \tag{8}$$

where $\mathbb{1}_s\{\cdot\}$ is an indicator, conditioned by the existence of $y_t^i = k$ in the sequence $Y_s^i$.

In practice, each agent is not only optimized to maximize the likelihood of the training data, but also to minimize the model divergence between its own model and the ensemble model:

$$\mathcal{L}_a^i = \lambda_i \mathcal{L}_{NLL} + (1 - \lambda_i)\mathcal{L}_{KD}^i \tag{9}$$

where $\lambda_i$ is a hyperparameter, balancing the weight of two factors.

From the perspective of learning, the agent learns to minimize the training objective by generating competitive translations, which leads to a global improvement for all agents. On the other hand, a sequence-level training objective might alleviate the *exposure bias* problem implicitly (Ranzato et al., 2016).

### 3.4 Joint Learning

In the work of Zhang et al. (2019b), they proposed a relatively complex joint learning framework for training two agents. In this paper, according to the previous experiments, we find that a simple multi-task learning technique without sharing any modules presents promising performance,

$$\mathcal{L}_{final} = \sum_{i=1}^{\mathcal{N}} \mathcal{L}_a^i \tag{10}$$

In Algorithm 1, we describe the overall procedure of our approach. It deserves noting that in line 3, we assume the model reads a single pair of training examples per timestep for simplistic description, while in practice the model reads a batch size of samples at each step.

**Algorithm 1:** Multi-agent Learning

**Input:** $N$ variant agents;
1 Pre-train each agent $agent_i$ independently;
2 **repeat**
3  **for** *each training sample* $(X_g, Y_g)$ **do**
4  Ensemble Model: $q(y_t|y_{<t}, x; \theta_t)$;
5  Generate sequence: $Y_t \leftarrow \arg\max_{0 \le t < T} q(y_t|y_{<t}, x; \theta_t)$;
6  **for** *each agent* $agent_i$ **do**
7  Generate sequence: $Y_s^i \leftarrow \arg\max_{0 \le t < T} p(y_t^i|y_{<t}^i, x; \theta)$;
8  Compute KD $loss$: $\mathcal{L}_{KD}^i \leftarrow (q(y_t), p(y_t^i), Y_t, Y_s^i, X_g, Y_g)$;
9  Compute NLL $loss$: $\mathcal{L}_{NLL} \leftarrow (p(y_t^i), X_g, Y_g)$ ;
10  Compute Agent $loss$: $\mathcal{L}_a^i \leftarrow \lambda\mathcal{L}_{NLL} + (1 - \lambda)\mathcal{L}_{KD}^i$;
11  **end**
12  Model $loss$: $\mathcal{L}_{final} = \sum_{i=1}^{\mathcal{N}} \mathcal{L}_a^i$;
13  Update gradients for each agent;
14  **end**
15 **until** *convergence*;

## 4 Experiments

In this paper, we evaluate our model on four translation tasks: NIST Chinese-English Translation Task, IWSLT 2014 German-English Translation Task, WMT 2014 English-German Translation Task and large-scale Chinese-English Translation Task.

### 4.1 Data Preprocessing

To compare with previous studies, we conduct byte-pair encoding (Sennrich et al., 2016) for Chinese, English and German sentences, setting the

| MODEL | MT02 | MT03 | MT04 | MT08 | AVERAGE | IMPROVEMENTS |
|---|---|---|---|---|---|---|
| | Results for Best Agent | | | | Results for each Agent | |
| *Wang et al. (2018)* | - | 46.60 | 47.73 | - | - | - |
| *a.L2R* | 48.53 | 47.07 | 48.43 | 42.21 | 46.56 | - |
| *b.R2L* | 47.06 | 45.58 | 47.14 | 41.04 | 45.20 | - |
| *c.Enc* | 48.86 | 47.54 | 48.57 | 42.93 | **46.97** | - |
| *d.Rel* | 48.12 | 48.19 | 48.33 | 42.51 | 46.78 | - |
| *a+b* | 48.82 | 47.65 | 48.45 | 42.49 | **46.86**/45.42 | **+0.33**/+0.20 |
| *a+c* | 48.79 | 48.30 | 49.32 | 43.44 | **47.3**/47.27 | **+0.80**/+0.30 |
| *a+d* | 48.76 | 48.40 | 48.74 | 43.27 | **47.19**/47.09 | **+0.63**/+0.31 |
| *c+d* | 49.45 | 49.01 | 49.52 | 43.71 | 47.62/**47.79** | +0.65/**+1.01** |
| *a×2* | 48.64 | 47.98 | 49.08 | 43.07 | 46.96/**47.10** | +0.40/**+0.54** |
| *d×2* | 48.23 | 48.78 | 48.90 | 43.85 | **47.44**/47.31 | **+0.66**/+0.53 |
| *a+b+c* | 49.32 | 48.72 | 49.32 | 44.34 | 47.69/45.62/**47.74** | **+1.13**/+0.42/+0.77 |
| *a+b+d* | 49.29 | 48.90 | 49.52 | 44.21 | **47.73**/45.71/47.65 | **+1.17**/+0.51/+0.87 |
| *a+c+d* | 49.42 | 49.13 | 49.68 | 44.66 | 47.72/**48.21**/47.94 | +1.16/**+1.24**/+0.96 |
| *a×3* | 48.75 | 48.09 | 49.19 | 43.18 | 47.07/**47.28**/47.20 | +0.51/**+0.72**/+0.64 |
| *d×3* | 48.47 | 49.02 | 49.42 | 44.09 | 47.51/47.46/**47.57** | +0.73/+0.68/**+0.79** |
| *a+b+c+d* | 49.52 | 48.94 | 49.61 | 44.70 | 47.95/46.10/**48.3**/47.95 | **+1.39**/+0.90/+1.33/+1.17 |

Table 2: BLEU score for the representative models in multi-agent training on NIST Chinese-English translation. $X \times Y$ stands for $Y$ agents with the indentical model $X$ but from different initializing seeds.

vocabulary size to 20K and 18K for Chinese-English, a joint 20K vocabulary for German-English, and a joint 32K vocabulary for English-German, respectively.

For Chinese-English task, the training data consists of about 1.5M sentence pairs extracted from LDC corpora [2]. We choose the NIST 2006 (MT06) dataset for validation, and NIST 2002-2004 (MT02-04), as well as NIST 2008 (MT08) datasets for testing. For large-scale Chinese-English task, the training data consists of about 40M sentence pairs extracted from web data.

### 4.2 Agent Variants

To increase the diversity of agents, we implement the following variants of the Transformer based system: *L2R*, the officially released open source toolkit for running Transformer model, *R2L*, the standard Transformer decodes in reversed direction, *Enc*, the standard Transformer with 30 layers in the encoder, and *Rel*, the reimplementation of self-attention with relative position (Shaw et al., 2018).

[2]LDC2002E18, LDC2002L27, LDC2002T01, LDC2003E07, LDC2003E14, LDC2004T07, LDC2005E83, LDC2005T06, LDC2005T10, LDC2005T34, LDC2006E24, LDC2006E26, LDC2006E34, LDC2006E86, LDC2006E92, LDC2006E93, LDC2004T08(HK News, HK Hansards )

### 4.3 Training Details

We implement our models using PaddlePaddle [3], an end-to-end open source deep learning platform developed by Baidu. It provides a complete suite of deep learning libraries, tools and service platforms to make the research and development of deep learning simple and reliable.

We use the hyperparameters of the *base* version for the standard Transformer model of NIST Chinese-English and IWSLT German-English translation tasks, except the smaller token size ($batch\_size = 320$). For WMT English-German and large-scale Chinese-English translation tasks, we use the hyperparameters of the *big* version. As described in the previous section, we use a hyperparameter $\lambda_i$ for each agent to balance the preference between learning from the observed data and the ensemble model. According to the performance of each agent after pre-training, we set this value as follows:

$$\lambda_i = 0.5 + \max(-0.5, \min(0.5, \frac{B_i - B_{avg}}{10})) \quad (11)$$

where $B_i$ is the BLEU score obtained by the pre-training of the $i-th$ agent, and the $B_{avg}$ is the average BLEU score of all agents.

The above formula suggests the agent learns more from the ensemble model as its performance

[3]https://github.com/paddlepaddle/paddle

| Task | *L2R* | *Rel* | KD-4 | Dual-5 | Rel-4 |
|---|---|---|---|---|---|
| De-En | 33.63 | 34.91 | 35.53 | 34.70 | **36.27** |

Table 3: BLEU score on IWSLT 2014 German-English translation. KD-4 stands for ensemble knowledge distillation with four agents. Dual-5 is the SOTA model from the work of Wang et al. (2019). And Rel-4 is our best model (*Rel*) training with four diverse agents.

is worse than the majority vote, rather than focusing on exploring by its own prediction.

We train our model with parallelization at data batch level. For NIST Chinese-English task, it takes about 1.5 days to train models on 8 NVIDIA P40 GPUs, 5 days for WMT English-German task, 8 hours for IWSLT German-English task and 7 days for large-scale Chinese-English task. The detailed training process is as follows, first we train each agent until BLEU doesn't improve any more and then execute Algorithm 1 for one-to-many learning which takes 30K-40K steps to converge.

### 4.4 Chinese-English Results

**Study on different numbers of agents.** We first assess the impact of diverse agents on translation quality. From Table 2, we see that the model's performance consistently improves as the number of agents increases, and we observe that 1) The four baseline systems with different implementations present diverse translation quality, in particular *Rel* with refined position encoding achieves the best performance. 2) The improvement of each agent after multi-agent learning is dependent on the performance of the co-trained agent (+0.33, +0.8, +0.63 improvements obtained by *L2R* when training with *R2L*, *Enc* and *Rel*). 3) Better results can be obtained by increasing the number of training agents (e.g., 47.3 $\rightarrow$ 47.73 $\rightarrow$ 47.95 for L2R).

From the overall results, increasing the number of agents in multi-agent learning significantly improves the performance of each agent (at most +1.39, +0.9, +1.33, +1.17 in *a*+*b*+*c*+*d*), which suggests that each agent learns advantages from the other agents, and more agents might lead to further improvement. More importantly, our improvements are obtained from the advanced training strategy without any modification in decoding stage, which indicates its practicability in deployment.

**Study on uniform agents.** To measure the importance of diversity in multi-agents, we conduct multiple experiments by initializing the similar model with different initialization seeds to generate multiple agents. As reported in Table 2, although the single *Rel* model achieves the best performance compared to the other three single agents, the *d* $\times$ *2* (47.44) and *d* $\times$ *3* (47.57) perform worse than the best counterparts *c*+*d* (47.79) and *a*+*c*+*d* (47.94). Moreover, when training with more than two agents, the performance of *d* $\times$ *3* (47.57) is even lower than arbitrary diverse counterparts (*a*+*b*+*c* (47.74); *a*+*b*+*d* (47.73)).

These results suggest that multi-agent learning is better conducted with diverse agents instead of the uniform agents with excellent single performance. On the other hand, in our multi-agent learning, training with more agents brings consistent improvements even when deployed by identical models from different initialization seeds.

| Models | En-De |
|---|---|
| ConvS2S Gehring et al. (2017) | 25.2 |
| Transformer Vaswani et al. (2017) | 28.4 |
| *Rel* Shaw et al. (2018) | 29.2 |
| DynamicConv Wu et al. (2019) | 29.7 |
| Back-translation Sergey et al. (2018) | 35.00 |
| Dual-3 Wang et al. (2019) | 29.92 |
| Dual-3 + Mono Data | 30.67 |
| *L2R* | 28.37 |
| *Rel* | 29.16 |
| Rel-4 | 29.67 |

Table 4: BLEU score on newstest2014 for WMT English-German translation.

### 4.5 German-English and English-German Results

We work with the IWSLT German-English translation to compare our model to the SOTA model. From Table 3, we observe that both KD-4 and Rel-4 models achieve the best result on this dataset (35.53 and 36.27), which manifests the effectiveness of using multiple agents.

Even when trained with four agents, our model outperforms the dual model trained with five agents (34.70), and reports a SOTA score on this translation task. Moreover, we argue that fine-tuning with *L2R* to obtain a better performance could further improve the performance of Rel-4, as there exists a large gap between *L2R* and *Rel*, which affects the learning efficiency.

We further investigate the performance of our

| Models | AVERAGE |
|---|---|
| *L2R(baseline)* | 33.74 |
| *L2R+R2L+Enc+Rel* | **34.60** |

Table 5: BLEU score for the L2R model in multi-agent training on large-scale corpus of Chinese-English translation. Because we only use L2R model in Baidu Translate, so part of results are presented.

model on the WMT English-German translation task, which achieves a competitive result of 29.67 BLEU score on newstest2014. From Table 4, we can see that another related work, multi-agent dual learning (Wang et al., 2019) achieves promising results. This confirms that training with more agents leads to better translation quality, and reveals the relevance of using multiple agents.

Although Sergey et al. (2018) reports a BLEU score of 35.0 on this dataset, they leverage refined training corpus and a large number of monolingual data. We argue that our model can bring further improvement using their back-translation technique. Moreover, the goal of this paper is introducing a general learning framework for multi-agent learning rather than exhaustively fine-tuning to report a SOTA results. We argue that the performance of our model can be further improved using an advanced single agent, such as DynamicConv (Wu et al., 2019) and Dual (Wang et al., 2019).

### 4.6 Large-Scale Chinese-English Results

In order to prove that multi-agent learning can yield improvement in large-scale corpus, we conducted experiments on 40M Chinese-English sentence pairs with the same data processing method as NIST. We built test sets of more than 10K sentences which cover various domains such as news, spoken language and query logs. From Table 5, we can see that multi-agent learning can increase the average baseline BLEU score of several test sets by 0.86. This technique has already been applied to Baidu Translate.

### 4.7 Contrastive Evaluation

In this paper, we evaluate our model on the two types of contrastive evaluation datasets: ***Lingeval97*** [4] and ***ContraWSD*** [5]. The former is utilized to measure the performance of different models on dealing with subject-verb agreement in English-German translation, while the latter evaluates the performance on word sense disambiguation in German-English translation. We suggest the readers refer to the work (Sennrich, 2017; Tang et al., 2018) for detailed descriptions of the two datasets.

| Models | SVA (%) | WSD (%) |
|---|---|---|
| *L2R* | 89.06 | 78.05 |
| *R2L* | 88.92 | 77.97 |
| *Enc* | 89.13 | **79.07** |
| *Rel* | **89.21** | 78.75 |
| *L2R+R2L* | 89.28 | 77.94 |
| *L2R+Enc* | **89.42** | 78.29 |
| *L2R+Rel* | 89.39 | **78.47** |
| *L2R+R2L+Enc* | 89.26 | 78.36 |
| *L2R+R2L+Rel* | **89.58** | 78.28 |
| *L2R+Enc+Rel* | 89.39 | **78.84** |
| *L2R+R2L+Enc+Rel* | 89.47 | 78.62 |

Table 6: Accuracy of subject-verb agreement (SVA) and word sense disambiguation (WSD) for different models. We report the performance of *L2R* in different models for comparison of using different agents.

From Table 6, we can see that the *L2R* training with multiple agents improves the accuracy both on SVA and WSD, and we observe that: 1) The *L2R* improves its ability of SVA with the help of the other agents significantly. 2) The *Enc* presents better advantage in resolving WSD than in SVA, while *Rel* has the opposite preference. 3) Although the *R2L* presents ordinary BLEU score, it still does help the *L2R* resolve the SVA.

## 5 Related Work

The most related work is recently proposed by Wang et al. (2019), who introduced a multi-agent algorithm for dual learning. However, the major differences are 1) the training in their work is conducted on the two agents while fixing the parameters of other agents. 2) they use identical agent with different initialization seeds. 3) our model is simple yet effective. Actually, it is easy to incorporate additional agent trained with their dual learning strategy in our model, to further improve the performance. More importantly, both of our work and their work indicate that using more agents can improve the translation quality significantly.

Another related work is ensemble knowledge distillation (Fukuda et al., 2017; Freitag et al., 2017; Liu et al., 2018; Zhu et al., 2018), in which

[4] https://github.com/rsennrich/lingeval97

[5] https://github.com/a-rios/ContraWSD

the ensemble model of all agents is leveraged as the *Teacher* network, to distill the knowledge for the corresponding *Student* network. However, as described in the previous section, the knowledge distillation is one particular case of our model, as the performance of the *Teacher* in their model is fixed, and cannot be further improved by the learning process.

Our work is also motivated by the work of training with two agents, including dual learning (He et al., 2016; Xia et al., 2017, 2018), and bidirectional decoding (Liu et al., 2016; Zhang et al., 2018, 2019b,a). Our method can be viewed as a general learning framework to train multiple agents, which explores the relationship among all agents to enhance the performance of each agent efficiently.

## 6 Conclusions and Future Work

In this paper, we propose a universal yet effective learning method for training multiple agents. In particular, each agent learns advantages from the ensemble model when necessary. The knowledge distillation from the ensemble model to the agent stems the better translation generated by the ensemble model, which enables each agent to learn high-quality knowledge while retaining its own exploration.

Extensive experimental results prove that our model brings an absolute improvement over the baseline system, reporting SOTA results on IWSLT 2014 German-English and competitive results on WMT 2014 English-German translation tasks.

In the future, we will focus on training with more agents by translating apart of one sentence considering its advantage for each agent, rather than translating the whole sentence.

## References


Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Tobias Domhan. 2018. How much attention do you need? a granular analysis of neural machine translation architectures. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1799–1808.

Markus Freitag, Yaser Al-Onaizan, and Baskaran Sankaran. 2017. Ensemble distillation for neural machine translation. *arXiv preprint arXiv:1702.01802*.

Takashi Fukuda, Masayuki Suzuki, Gakuto Kurata, Samuel Thomas, Jia Cui, and Bhuvana Ramabhadran. 2017. Efficient knowledge distillation from an ensemble of teachers. *Proc. Interspeech 2017*, pages 3697–3701.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *Proc. of ICML*.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In *Advances in Neural Information Processing Systems*, pages 820–828.

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*.

Yoon Kim and Alexander M Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327.

Lemao Liu, Masao Utiyama, Andrew Finch, and Eiichiro Sumita. 2016. Agreement on target-bidirectional neural machine translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 411–416.

Yijia Liu, Wanxiang Che, Huaipeng Zhao, Bing Qin, and Ting Liu. 2018. Distilling knowledge for search-based structured prediction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1393–1402. Association for Computational Linguistics.

Leandro Soriano Marcolino, Albert Xin Jiang, and Milind Tambe. 2013. Multi-agent team formation: diversity beats strength? In *Twenty-Third International Joint Conference on Artificial Intelligence*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *International Conference on Learning Representations*.

Rico Sennrich. 2017. How grammatical is character-level neural machine translation? assessing mt quality with contrastive translation pairs. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 376–382.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual*

*Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1715–1725.

Edunov Sergey, Ott Myle, Auli Michael, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of the conference on empirical methods in natural language processing*, pages 489–500. Association for Computational Linguistics.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 464–468.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Gongbo Tang, Mathias Müller, Annette Rios, and Rico Sennrich. 2018. Why self-attention? a targeted evaluation of neural machine translation architectures. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4263–4272. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Mingxuan Wang, Jun Xie, Zhixing Tan, Jinsong Su, Deyi Xiong, and Chao Bian. 2018. Neural machine translation with decoding history enhanced attention. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1464–1473.

Yiren Wang, Yingce Xia, Tianyu He, Fei Tian, Tao Qin, ChengXiang Zhai, and Tie-Yan Liu. 2019. Multi-agent dual learning. In *International Conference for Learning Representation (ICLR)*.

KY Michael Wong, SW Lim, and Zhuo Gao. 2005. Effects of diversity on multiagent systems: Minority games. *Physical Review E*, 71(6):066103.

Felix Wu, Angela Fan, Alexei Baevski, Yann N. Dauphin, and Michael Auli. 2019. Pay less attention with lightweight and dynamic convolutions. In *ICLR*.

Yingce Xia, Tao Qin, Wei Chen, Jiang Bian, Nenghai Yu, and Tie-Yan Liu. 2017. Dual supervised learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3789–3798. JMLR. org.

Yingce Xia, Xu Tan, Fei Tian, Tao Qin, Nenghai Yu, and Tie-Yan Liu. 2018. Model-level dual learning. In *International Conference on Machine Learning*, pages 5379–5388.

Jiajun Zhang, Long Zhou, Yang Zhao, and Chengqing Zong. 2019a. Synchronous bidirectional inference for neural sequence generation. *arXiv preprint arXiv:1902.08955*.

Xiangwen Zhang, Jinsong Su, Yue Qin, Yang Liu, Rongrong Ji, and Hongji Wang. 2018. Asynchronous bidirectional decoding for neural machine translation. *arXiv preprint arXiv:1801.05122*.

Zhirui Zhang, Shuangzhi Wu, Shujie Liu, Mu Li, Ming Zhou, and Enhong Chen. 2019b. Regularizing neural machine translation by target-bidirectional agreement. In *AAAI*.

Xiatian Zhu, Shaogang Gong, et al. 2018. Knowledge distillation by on-the-fly native ensemble. In *Advances in Neural Information Processing Systems*, pages 7528–7538.

# Pivot-based Transfer Learning for Neural Machine Translation between Non-English Languages

**Yunsu Kim**[1*] **Petre Petrov**[1,2*] **Pavel Petrushkov**[2] **Shahram Khadivi**[2] **Hermann Ney**[1]
[1]RWTH Aachen University, Aachen, Germany
{surname}@cs.rwth-aachen.de
[2]eBay, Inc., Aachen, Germany
{petrpetrov,ppetrushkov,skhadivi}@ebay.com

## Abstract

We present effective pre-training strategies for neural machine translation (NMT) using parallel corpora involving a pivot language, i.e., source-pivot and pivot-target, leading to a significant improvement in source→target translation. We propose three methods to increase the relation among source, pivot, and target languages in the pre-training: 1) step-wise training of a single model for different language pairs, 2) additional adapter component to smoothly connect pre-trained encoder and decoder, and 3) cross-lingual encoder training via autoencoding of the pivot language. Our methods greatly outperform multilingual models up to +2.6% BLEU in WMT 2019 French→German and German→Czech tasks. We show that our improvements are valid also in zero-shot/zero-resource scenarios.

## 1 Introduction

Machine translation (MT) research is biased towards language pairs including English due to the ease of collecting parallel corpora. Translation between non-English languages, e.g., French→German, is usually done with pivoting through English, i.e., translating French (*source*) input to English (*pivot*) first with a French→English model which is later translated to German (*target*) with a English→German model (De Gispert and Marino, 2006; Utiyama and Isahara, 2007; Wu and Wang, 2007). However, pivoting requires doubled decoding time and the translation errors are propagated or expanded via the two-step process.

Therefore, it is more beneficial to build a single source→target model directly for both efficiency and adequacy. Since non-English language pairs often have little or no parallel text, common choices to avoid pivoting in NMT are generating pivot-based synthetic data (Bertoldi et al., 2008; Chen et al., 2017) or training multilingual systems (Firat et al., 2016; Johnson et al., 2017).

In this work, we present novel transfer learning techniques to effectively train a single, direct NMT model for a non-English language pair. We pre-train NMT models for source→pivot and pivot→target, which are transferred to a source→target model. To optimize the usage of given source-pivot and pivot-target parallel data for the source→target direction, we devise the following techniques to smooth the discrepancy between the pre-trained and final models:

- Step-wise pre-training with careful parameter freezing.
- Additional adapter component to familiarize the pre-trained decoder with the outputs of the pre-trained encoder.
- Cross-lingual encoder pre-training with autoencoding of the pivot language.

Our methods are evaluated in two non-English language pairs of WMT 2019 news translation tasks: high-resource (French→German) and low-resource (German→Czech). We show that NMT models pre-trained with our methods are highly effective in various data conditions, when fine-tuned for source→target with:

- Real parallel corpus
- Pivot-based synthetic parallel corpus (*zero-resource*)
- None (*zero-shot*)

For each data condition, we consistently outperform strong baselines, e.g., multilingual, pivoting, or teacher-student, showing the universal effectiveness of our transfer learning schemes.

[*] Equal contribution.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 866–876, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

The rest of the paper is organized as follows. We first review important previous works on pivot-based MT in Section 2. Our three pre-training techniques are presented in Section 3. Section 4 shows main results of our methods with a detailed description of the experimental setups. Section 5 studies variants of our methods and reports the results without source-target parallel resources or with large synthetic parallel data. Section 6 draws conclusion of this work with future research directions.

## 2 Related Work

In this section, we first review existing approaches to leverage a pivot language in low-resource/zero-resource MT. They can be divided into three categories:

1. **Pivot translation (pivoting).** The most naive approach is reusing (already trained) source→pivot and pivot→target models directly, decoding twice via the pivot language (Kauers et al., 2002; De Gispert and Marino, 2006). One can keep $N$-best hypotheses in the pivot language to reduce the prediction bias (Utiyama and Isahara, 2007) and improve the final translation by system combination (Costa-Jussà et al., 2011), which however increases the translation time even more. In multilingual NMT, Firat et al. (2016) modify the second translation step (pivot→target) to use source and pivot language sentences together as the input.
2. **Pivot-based synthetic parallel data.** We may translate the pivot side of given pivot-target parallel data using a pivot→source model (Bertoldi et al., 2008), or the other way around translating source-pivot data using a pivot→target model (De Gispert and Marino, 2006). For NMT, the former is extended by Zheng et al. (2017) to compute the expectation over synthetic source sentences. The latter is also called teacher-student approach (Chen et al., 2017), where the pivot→target model (teacher) produces target hypotheses for training the source→target model (student).
3. **Pivot-based model training.** In phrase-based MT, there have been many efforts to combine phrase/word level features of source-pivot and pivot-target into a source→target system (Utiyama and Isahara, 2007; Wu and Wang, 2007; Bakhshaei et al., 2010; Zahabi et al., 2013; Zhu et al., 2014; Miura et al., 2015). In NMT, Cheng et al. (2017) jointly train for three translation directions of source-pivot-target by sharing network components, where Ren et al. (2018) use the expectation-maximization algorithm with the target sentence as a latent variable. Lu et al. (2018) deploy intermediate recurrent layers which are common for multiple encoders and decoders, while Johnson et al. (2017) share all components of a single multilingual model. Both methods train the model for language pairs involving English but enable zero-shot translation for unseen non-English language pairs. For this, Ha et al. (2017) encode the target language as an additional embedding and filter out non-target tokens in the output. Lakew et al. (2017) combine the multilingual training with synthetic data generation to improve the zero-shot performance iteratively, where Sestorain et al. (2018) applies the NMT prediction score and a language model score to each synthetic example as gradient weights.

Our work is based on transfer learning (Zoph et al., 2016) and belongs to the third category: model training. On the contrary to the multilingual joint training, we suggest two distinct steps: pre-training (with source-pivot and pivot-target data) and fine-tuning (with source-target data). With our proposed methods, we prevent the model from losing its capacity to other languages while utilizing the information from related language pairs well, as shown in the experiments (Section 4).

Our pivot adapter (Section 3.2) shares the same motivation with the interlingua component of Lu et al. (2018), but is much compact, independent of variable input length, and easy to train offline. The adapter training algorithm is adopted from bilingual word embedding mapping (Xing et al., 2015). Our cross-lingual encoder (Section 3.3) is inspired by cross-lingual sentence embedding algorithms using NMT (Schwenk and Douze, 2017; Schwenk, 2018).

Transfer learning was first introduced to NMT by Zoph et al. (2016), where only the source language is switched before/after the transfer. Nguyen and Chiang (2017) and Kocmi and Bojar (2018) use shared subword vocabularies to work

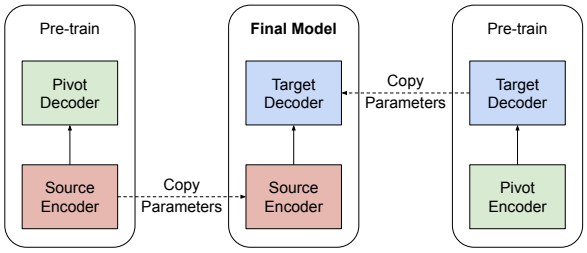


Figure 1: Plain transfer learning.

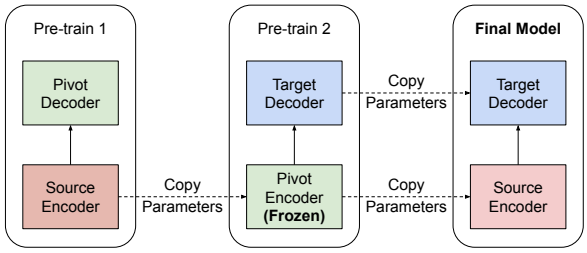


Figure 2: Step-wise pre-training.

with more languages and help target language switches. Kim et al. (2019) propose additional techniques to enable NMT transfer even without shared vocabularies. To the best of our knowledge, we are the first to propose transfer learning strategies specialized in utilizing a pivot language, transferring a source encoder and a target decoder at the same time. Also, for the first time, we present successful zero-shot translation results only with pivot-based NMT pre-training.

## 3 Pivot-based Transfer Learning

Our methods are based on a simple transfer learning principle for NMT, adjusted to a usual data condition for non-English language pairs: lots of source-pivot and pivot-target parallel data, little (low-resource) or no (zero-resource) source-target parallel data. Here are the core steps of the plain transfer (Figure 1):

1. Pre-train a source→pivot model with a source-pivot parallel corpus and a pivot→target model with a pivot-target parallel corpus.
2. Initialize the source→target model with the source encoder from the pre-trained source→pivot model and the target decoder from the pre-trained pivot→target model.
3. Continue the training with a source-target parallel corpus.

If we skip the last step (for zero-resource cases) and perform the source→target translation directly, it corresponds to zero-shot translation.

Thanks to the pivot language, we can pre-train a source encoder and a target decoder without changing the model architecture or training objective for NMT. On the contrary to other NMT transfer scenarios (Zoph et al., 2016; Nguyen and Chiang, 2017; Kocmi and Bojar, 2018), this principle has no language mismatch between transferor and transferee on each source/target side. Experimental results (Section 4) also show its competitiveness despite its simplicity.

Nonetheless, the main caveat of this basic pre-training is that the source encoder is trained to be used by an English decoder, while the target decoder is trained to use the outputs of an English encoder — not of a source encoder. In the following, we propose three techniques to mitigate the inconsistency of source→pivot and pivot→target pre-training stages. Note that these techniques are not exclusive and some of them can complement others for a better performance of the final model.

### 3.1 Step-wise Pre-training

A simple remedy to make the pre-trained encoder and decoder refer to each other is to train a single NMT model for source→pivot and pivot→target in consecutive steps (Figure 2):

1. Train a source→pivot model with a source-pivot parallel corpus.
2. Continue the training with a pivot-target parallel corpus, while freezing the encoder parameters of 1.

In the second step, a target decoder is trained to use the outputs of the pre-trained source encoder as its input. Freezing the pre-trained encoder ensures that, even after the second step, the encoder is still modeling the source language although we train the NMT model for pivot→target. Without the freezing, the encoder completely adapts to the pivot language input and is likely to forget source language sentences.

We build a joint vocabulary of the source and pivot languages so that the encoder effectively represents both languages. The frozen encoder is pre-trained for the source language in the first step, but also able to encode a pivot language sentence in a similar representation space. It is more effective for linguistically similar languages where

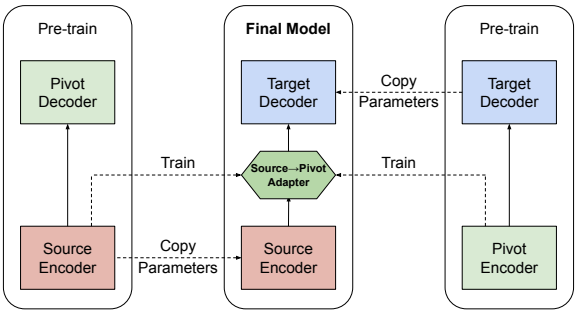


Figure 3: Pivot adapter.

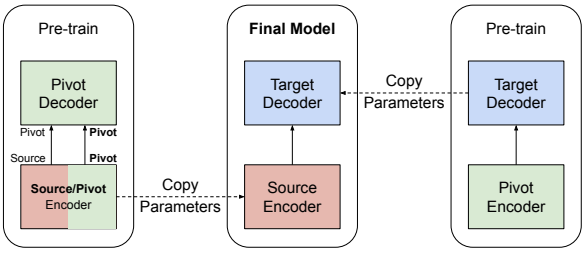


Figure 4: Cross-lingual encoder.

many tokens are common for both languages in the joint vocabulary.

### 3.2 Pivot Adapter

Instead of the step-wise pre-training, we can also postprocess the network to enhance the connection between the source encoder and the target decoder which are pre-trained individually. Our idea is that, after the pre-training steps, we adapt the source encoder outputs to the pivot encoder outputs to which the target decoder is more familiar (Figure 3). We learn a linear mapping between the two representation spaces with a small source-pivot parallel corpus:

1. Encode the source sentences with the source encoder of the pre-trained source→pivot model.
2. Encode the pivot sentences with the pivot encoder of the pre-trained pivot→target model.
3. Apply a pooling to each sentence of 1 and 2, extracting representation vectors for each sentence pair: $(\mathbf{s}, \mathbf{p})$.
4. Train a mapping $\mathbf{M} \in \mathbb{R}^{d \times d}$ to minimize the distance between the pooled representations $\mathbf{s} \in \mathbb{R}^{d \times 1}$ and $\mathbf{p} \in \mathbb{R}^{d \times 1}$, where the source representation is first fed to the mapping:

$$\hat{\mathbf{M}} = \operatorname*{argmin}_{\mathbf{M}} \sum_{\mathbf{s},\mathbf{p}} \|\mathbf{M}\mathbf{s} - \mathbf{p}\|^2 \qquad (1)$$

where $d$ is the hidden layer size of the encoders. Introducing matrix notations $\mathbf{S} \in \mathbb{R}^{d \times n}$ and $\mathbf{P} \in \mathbb{R}^{d \times n}$, which concatenate the pooled representations of all $n$ sentences for each side in the source-pivot corpus, we rewrite Equation 1 as:

$$\hat{\mathbf{M}} = \operatorname*{argmin}_{\mathbf{M}} \|\mathbf{M}\mathbf{S} - \mathbf{P}\|^2 \qquad (2)$$

which can be easily computed by the singular value decomposition (SVD) for a closed-form solution, if we put an orthogonality constraint on $\mathbf{M}$ (Xing et al., 2015). The resulting optimization is also called Procrustes problem.

The learned mapping is multiplied to encoder outputs of all positions in the final source→target tuning step. With this mapping, the source encoder emits sentence representations that lie in a similar space of the pivot encoder. Since the target decoder is pre-trained for pivot→target and accustomed to receive the pivot encoder outputs, it should process the mapped encoder outputs better than the original source encoder outputs.

### 3.3 Cross-lingual Encoder

As a third technique, we modify the source→pivot pre-training procedure to force the encoder to have cross-linguality over source and pivot languages; modeling source and pivot sentences in the same mathematical space. We achieve this by an additional autoencoding objective from a pivot sentence to the same pivot sentence (Figure 4).

The encoder is fed with sentences of both source and pivot languages, which are processed by a shared decoder that outputs only the pivot language. In this way, the encoder is learned to produce representations in a shared space regardless of the input language, since they are used in the same decoder. This cross-lingual space facilitates smoother learning of the final source→target model, because the decoder is pre-trained to translate the pivot language.

The same input/output in autoencoding encourages, however, merely copying the input; it is said to be not proper for learning complex structure of the data domain (Vincent et al., 2008). Denoising autoencoder addresses this by corrupting the input sentences by artificial noises (Hill et al., 2016). Learning to reconstruct clean sentences, it encodes linguistic structures of natural language sentences, e.g., word order, better than copying. Here are the

noise types we use (Edunov et al., 2018):

- Drop tokens randomly with a probability $p_{\text{del}}$
- Replace tokens with a <BLANK> token randomly with a probability $p_{\text{rep}}$
- Permute the token positions randomly so that the difference between an original index and its new index is less than or equal to $d_{\text{per}}$

We set $p_{\text{del}} = 0.1$, $p_{\text{rep}} = 0.1$, and $d_{\text{per}} = 3$ in our experiments.

The key idea of all three methods is to build a closer connection between the pre-trained encoder and decoder via a pivot language. The difference is in when we do this job: Cross-lingual encoder (Section 3.3) changes the encoder pre-training stage (source→pivot), while step-wise pre-training (Section 3.1) modifies decoder pre-training stage (pivot→target). Pivot adapter (Section 3.2) is applied after all pre-training steps.

## 4 Main Results

We evaluate the proposed transfer learning techniques in two non-English language pairs of WMT 2019 news translation tasks[1]: French→German and German→Czech.

**Data** We used the News Commentary v14 parallel corpus and newstest2008-2010 test sets as the source-target training data for both tasks. The newstest sets were oversampled four times. The German→Czech task was originally limited to unsupervised learning (using only monolingual corpora) in WMT 2019, but we relaxed this constraint by the available parallel data. We used newstest2011 as a validation set and newstest2012/newstest2013 as the test sets.

Both language pairs have much abundant parallel data in source-pivot and pivot-target with English as the pivot language. Detailed corpus statistics are given in Table 1.

**Preprocessing** We used the Moses[2] tokenizer and applied true-casing on all corpora. For all transfer learning setups, we learned byte pair encoding (BPE) (Sennrich et al., 2016) for each language individually with 32k merge operations, except for cross-lingual encoder training with joint BPE only over source and pivot languages. This is for modularity of pre-trained models: for example, a French→English model trained with joint French/English/German BPE could be transferred smoothly to a French→German model, but would not be optimal for a transfer to e.g., a French→Korean model. Once we pre-train an NMT model with separate BPE vocabularies, we can reuse it for various final language pairs without wasting unused portion of subword vocabularies (e.g., German-specific tokens in building a French→Korean model).

On the contrary, baselines used joint BPE over all languages with also 32k merges.

| Usage | Data | Sentences | Words (Source) |
|---|---|---|---|
| Pre-train | fr-en | 35M | 950M |
| | en-de | 9.1M | 170M |
| Fine-tune | fr-de | 270k | 6.9M |
| Pre-train | de-en | 9.1M | 181M |
| | en-cs | 49M | 658M |
| Fine-tune | de-cs | 230k | 5.1M |

Table 1: Parallel training data statistics.

**Model and Training** The 6-layer base Transformer architecture (Vaswani et al., 2017) was used for all of our experiments. Batch size was set to 4,096 tokens. Each checkpoint amounts to 10k updates for pre-training and 20k updates for fine-tuning.

Each model was optimized with Adam (Kingma and Ba, 2014) with an initial learning rate of 0.0001, which was multiplied by 0.7 whenever perplexity on the validation set was not improved for three checkpoints. When it was not improved for eight checkpoints, we stopped the training. The NMT model training and transfer were done with the OpenNMT toolkit (Klein et al., 2017).

Pivot adapter was trained using the MUSE toolkit (Conneau et al., 2018), which was originally developed for bilingual word embeddings but we adjusted for matching sentence representations.

**Baselines** We thoroughly compare our approaches to the following baselines:

1. *Direct source→target*: A standard NMT model trained on given source→target paral-

[1] http://www.statmt.org/wmt19/translation-task.html

[2] http://www.statmt.org/moses/

| | French→German | | | | German→Czech | | | |
|---|---|---|---|---|---|---|---|---|
| | newstest2012 | | newstest2013 | | newstest2012 | | newstest2013 | |
| | BLEU [%] | TER [%] | BLEU [%] | TER [%] | BLEU [%] | TER [%] | BLEU [%] | TER [%] |
| Direct source→target | 14.8 | 75.1 | 16.0 | 75.1 | 11.1 | 81.1 | 12.8 | 77.7 |
| Multilingual many-to-many | 18.7 | 71.9 | 19.5 | 72.6 | 14.9 | 76.6 | 16.5 | 73.2 |
| Multilingual many-to-one | 18.3 | 71.7 | 19.2 | 71.5 | 13.1 | 79.6 | 14.6 | 75.8 |
| Plain transfer | 17.5 | 72.3 | 18.7 | 71.8 | 15.4 | 75.4 | 18.0 | 70.9 |
| + Pivot adapter | 18.0 | 71.9 | 19.1 | 71.1 | 15.9 | 75.0 | 18.7 | 70.3 |
| + Cross-lingual encoder | 17.4 | 72.1 | 18.9 | 71.8 | 15.0 | 75.9 | 17.6 | 71.4 |
| + Pivot adapter | 17.8 | 72.3 | 19.1 | 71.5 | 15.6 | 75.3 | 18.1 | 70.8 |
| Step-wise pre-training | 18.6 | 70.7 | 19.9 | 70.4 | 15.6 | 75.0 | 18.1 | 70.9 |
| + Cross-lingual encoder | **19.5** | **69.8** | **20.7** | **69.4** | **16.2** | **74.6** | **19.1** | **69.9** |

Table 2: Main results fine-tuned with source-target parallel data.

lel data.

2. *Multilingual*: A single, shared NMT model for multiple translation directions (Johnson et al., 2017).
    - *Many-to-many*: Trained for all possible directions among source, target, and pivot languages.
    - *Many-to-one*: Trained for only the directions *to* target language, i.e., source→target and pivot→target, which tends to work better than many-to-many systems (Aharoni et al., 2019).

In Table 2, we report principal results after fine-tuning the pre-trained models using source-target parallel data.

As for baselines, multilingual models are better than a direct NMT model. The many-to-many models surpass the many-to-one models; since both tasks are in a low-resource setup, the model gains a lot from related language pairs even if the target languages do not match.

Plain transfer of pre-trained encoder/decoder without additional techniques (Figure 1) shows a nice improvement over the direct baseline: up to +2.7% BLEU for French→German and +5.2% BLEU for German→Czech. Pivot adapter provides an additional boost of maximum +0.7% BLEU or -0.7% TER.

Cross-lingual encoder pre-training is proved to be not effective in the plain transfer setup. It shows no improvements over plain transfer in French→German, and 0.4% BLEU worse performance in German→Czech. We conjecture that the cross-lingual encoder needs a lot more data to be fine-tuned for another decoder, where the encoder capacity is basically divided into two languages at the beginning of the fine-tuning. On the other hand, the pivot adapter directly improves the connection to an individually pre-trained decoder, which works nicely with small fine-tuning data.

Pivot adapter gives an additional improvement on top of the cross-lingual encoder; up to +0.4% BLEU in French→German and +0.6% BLEU in German→Czech. In this case, we extract source and pivot sentence representations from the same shared encoder for training the adapter.

Step-wise pre-training gives a big improvement up to +1.2% BLEU or -1.6% TER against plain transfer in French→German. It shows the best performance in both tasks when combined with the cross-lingual encoder: up to +1.2% BLEU in French→German and +2.6% BLEU in German→Czech, compared to the multilingual baseline. Step-wise pre-training prevents the cross-lingual encoder from degeneration, since the pivot→target pre-training (Step 2 in Section 3.1) also learns the encoder-decoder connection with a large amount of data — in addition to the source→target tuning step afterwards.

Note that the pivot adapter, which inserts an extra layer between the encoder and decoder, is not appropriate after the step-wise pre-training; the decoder is already trained to correlate well with the pre-trained encoder. We experimented with the pivot adapter on top of step-wise pre-trained models — with or without cross-lingual encoder — but obtained detrimental results.

Compared to pivot translation (Table 5), our

best results are also clearly better in French →German and comparable in German→Czech.

## 5 Analysis

In this section, we conduct ablation studies on the variants of our methods and see how they perform in different data conditions.

### 5.1 Pivot Adapter

| Adapter Training | newstest2013 BLEU [%] | TER [%] |
|---|---|---|
| None | 18.2 | 70.7 |
| Max-pooled | 18.4 | 70.5 |
| Average-pooled | **18.7** | **70.3** |
| Plain transfer | 18.0 | 70.9 |

Table 3: Pivot adapter variations (German→Czech). All results are tuned with source-target parallel data.

Firstly, we compare variants of the pivot adapter (Section 3.2) in Table 3. The row "None" shows that a randomly initialized linear layer already guides the pre-trained encoder/decoder to harmonize with each other. Of course, when we train the adapter to map source encoder outputs to pivot encoder outputs, the performance gets better. For compressing encoder outputs over positions, average-pooling is better than max-pooling. We observed the same trend in the other test set and in French→German.

We also tested nonlinear pivot adapter, e.g., a 2-layer feedforward network with ReLU activations, but the performance was not better than just a linear adapter.

### 5.2 Cross-lingual Encoder

| Trained on | Input | newstest2013 BLEU [%] | TER [%] |
|---|---|---|---|
| Monolingual | Clean | 15.7 | 77.7 |
| | Noisy | 17.5 | 73.6 |
| Pivot side of parallel | Clean | 15.9 | 77.3 |
| | Noisy | **18.0** | **72.7** |

Table 4: Cross-lingual encoder variations (French→ German). All results are in the zero-shot setting with step-wise pre-training.

Table 4 verifies that the noisy input in autoencoding is indeed beneficial to our cross-lingual encoder. It improves the final translation performance by maximum +2.1% BLEU, compared to using the copying autoencoding objective.

As the training data for autoencoding, we also compare between purely monolingual data and the pivot side of the source-pivot parallel data. By the latter, one can expect a stronger signal for a joint encoder representation space, since two different inputs (in source/pivot languages) are used to produce the exactly same output sentence (in pivot language). The results also tell that there are slight but consistent improvements by using the pivot part of the parallel data.

Again, we performed these comparisons in the other test set and German→Czech, observing the same tendency in results.

### 5.3 Zero-resource/Zero-shot Scenarios

If we do not have an access to any source-target parallel data (*zero-resource*), non-English language pairs have two options for still building a working NMT system, given source-English and target-English parallel data:

- *Zero-shot*: Perform source→target translation using models which have not seen any source-target parallel sentences, e.g., multilingual models or pivoting (Section 2.1).
- *Pivot-based synthetic data*: Generate synthetic source-target parallel data using source↔English and target↔English models (Section 2.2). Use this data to train a model for source→target.

Table 5 shows how our pre-trained models perform in zero-resource scenarios with the two options. Note that, unlike Table 2, the multilingual baselines exclude source→target and target→source directions. First of all, plain transfer, where the encoder and the decoder are pre-trained separately, is poor in zero-shot scenarios. It simply fails to connect different representation spaces of the pre-trained encoder and decoder. In our experiments, neither pivot adapter nor cross-lingual encoder could enhance the zero-shot translation of plain transfer.

Step-wise pre-training solves this problem by changing the decoder pre-training to familiarize itself with representations from an already pre-trained encoder. It achieves zero-shot performance of 11.5% BLEU in French→German and 6.5% BLEU in German→Czech (newstest2013), while

| | French→German | | | | German→Czech | | | |
|---|---|---|---|---|---|---|---|---|
| | newstest2012 | | newstest2013 | | newstest2012 | | newstest2013 | |
| | BLEU [%] | TER [%] | BLEU [%] | TER [%] | BLEU [%] | TER [%] | BLEU [%] | TER [%] |
| Multilingual many-to-many | 14.1 | 79.1 | 14.6 | 79.1 | 5.9 | - | 6.3 | 99.8 |
| Pivot translation | 16.6 | 72.4 | 17.9 | 72.5 | 16.4 | 74.5 | **19.5** | **70.1** |
| Teacher-student | 18.7 | 70.3 | 20.7 | 69.5 | 16.0 | 75.0 | 18.5 | 70.9 |
| Plain transfer | 0.1 | - | 0.2 | - | 0.1 | - | 0.1 | - |
| Step-wise pre-training | 11.0 | 81.6 | 11.5 | 82.5 | 6.0 | 92.1 | 6.5 | 87.8 |
| + Cross-lingual encoder | 17.3 | 72.1 | 18.0 | 72.7 | 14.1 | 76.8 | 16.5 | 73.5 |
| + Teacher-student | **19.3** | **69.7** | **20.9** | **69.3** | **16.5** | **74.6** | 19.1 | 70.2 |

Table 5: Zero-resource results. Except those with the teacher-student, the results are all in the zero-shot setting, i.e., the model is not trained on any source-target parallel data. '-' indicates a TER score over 100%.

showing comparable or better fine-tuned performance against plain transfer (see also Table 2).

With the pre-trained cross-lingual encoder, the zero-shot performance of step-wise pre-training is superior to that of pivot translation in French→German with only a single model. It is worse than pivot translation in German→Czech. We think that the data size of pivot-target is critical in pivot translation; relatively huge data for English→Czech make the pivot translation stronger. Note again that, nevertheless, pivoting (second row) is very poor in efficiency since it performs decoding twice with the individual models.

For the second option (pivot-based synthetic data), we compare our methods against the sentence-level beam search version of the teacher-student framework (Chen et al., 2017), with which we generated 10M synthetic parallel sentence pairs. We also tried other variants of Chen et al. (2017), e.g., $N$-best hypotheses with weights, but there were no consistent improvements.

Due to enormous bilingual signals, the model trained with the teacher-student synthetic data outperforms pivot translation. If tuned with the same synthetic data, our pre-trained model performs even better (last row), achieving the best zero-resource results on three of the four test sets.

We also evaluate our best German→Czech zero-resource model on newstest2019 and compare it with the participants of the WMT 2019 unsupervised news translation task. Ours yield 17.2% BLEU, which is much better than the best single unsupervised system of the winner of the task (15.5%) (Marie et al., 2019). We argue that, if one has enough source-English and English-target parallel data for a non-English language pair, it is more encouraged to adopt pivot-based transfer learning than unsupervised MT — even if there is no source-target parallel data. In this case, unsupervised MT unnecessarily restricts the data condition to using only monolingual data and its high computational cost does not pay off; simple pivot-based pre-training steps are more efficient and effective.

### 5.4 Large-scale Results

We also study the effect of pivot-based transfer learning in more data-rich scenarios: 1) with large synthetic source-target data (German→Czech), and 2) with larger real source-target data in combination with the synthetic data (French→German). We generated synthetic parallel data using pivot-based back-translation (Bertoldi et al., 2008): 5M sentence pairs for German→Czech and 9.1M sentence pairs for French→German. For the second scenario, we also prepared 2.3M more lines of French→German real parallel data from Europarl v7 and Common Crawl corpora.

Table 6 shows our transfer learning results fine-tuned with a combination of given parallel data and generated synthetic parallel data. The real source-target parallel data are oversampled to make the ratio of real and synthetic data to be 1:2. As expected, the direct source→target model can be improved considerably by training with large synthetic data.

Plain pivot-based transfer outperforms the synthetic data baseline by up to +1.9% BLEU or -3.3% TER. However, the pivot adapter or cross-lingual encoder gives marginal or inconsistent improvements over the plain transfer. We suppose that the entire model can be tuned sufficiently well without

| | French→German | | | | German→Czech | | | |
|---|---|---|---|---|---|---|---|---|
| | newstest2012 | | newstest2013 | | newstest2012 | | newstest2013 | |
| | BLEU [%] | TER [%] | BLEU [%] | TER [%] | BLEU [%] | TER [%] | BLEU [%] | TER [%] |
| Direct source→target | 20.1 | 69.8 | 22.3 | 68.7 | 11.1 | 81.1 | 12.8 | 77.7 |
| + Synthetic data | 21.1 | 68.2 | 22.6 | 68.1 | 15.7 | 76.5 | 18.5 | 72.0 |
| Plain transfer | 21.8 | 67.6 | 23.1 | 67.5 | 17.6 | 73.2 | 20.3 | 68.7 |
| + Pivot adapter | 21.8 | 67.6 | 23.1 | 67.6 | **17.6** | **73.0** | **20.9** | **68.3** |
| + Cross-lingual encoder | 21.9 | 67.7 | 23.4 | 67.4 | 17.5 | 73.5 | 20.3 | 68.7 |
| + Pivot adapter | **22.1** | **67.5** | 23.3 | 67.5 | 17.5 | 73.2 | 20.6 | 68.5 |
| Step-wise pre-training | 21.8 | 67.8 | 23.0 | 67.8 | 17.3 | 73.6 | 20.0 | 69.2 |
| + Cross-lingual encoder | 21.9 | 67.6 | **23.4** | **67.4** | 17.5 | 73.1 | 20.5 | 68.6 |

Table 6: Results fine-tuned with a combination of source-target parallel data and large synthetic data. French→German task used larger real parallel data than Table 2.

additional adapter layers or a well-curated training process, once we have a large source-target parallel corpus for fine-tuning.

## 6 Conclusion

In this paper, we propose three effective techniques for transfer learning using pivot-based parallel data. The principle is to pre-train NMT models with source-pivot and pivot-target parallel data and transfer the source encoder and the target decoder. To resolve the input/output discrepancy of the pre-trained encoder and decoder, we 1) consecutively pre-train the model for source→pivot and pivot→target, 2) append an additional layer after the source encoder which adapts the encoder output to the pivot language space, or 3) train a cross-lingual encoder over source and pivot languages.

Our methods are suitable for most of the non-English language pairs with lots of parallel data involving English. Experiments in WMT 2019 French→German and German→Czech tasks show that our methods significantly improve the final source→target translation performance, outperforming multilingual models by up to +2.6% BLEU. The methods are applicable also to zero-resource language pairs, showing a strong performance in the zero-shot setting or with pivot-based synthetic data. We claim that our methods expand the advances in NMT to many more non-English language pairs that are not yet studied well.

Future work will be zero-shot translation without step-wise pre-training, i.e., combining individually pre-trained encoders and decoders freely for a fast development of NMT systems for a new non-English language pair.

## Acknowledgments




This work has received funding from the European Research Council (ERC) (under the European Union's Horizon 2020 research and innovation programme, grant agreement No 694537, project "SEQCLAS") and eBay Inc. The work reflects only the authors' views and none of the funding agencies is responsible for any use that may be made of the information it contains.

## References

Roee Aharoni, Melvin Johnson, and Orhan Firat. 2019. Massively multilingual neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3874–3884.

Somayeh Bakhshaei, Shahram Khadivi, and Noushin Riahi. 2010. Farsi-german statistical machine translation through bridge language. In *2010 5th International Symposium on Telecommunications*, pages 557–561. IEEE.

Nicola Bertoldi, Madalina Barbaiani, Marcello Federico, and Roldano Cattoni. 2008. Phrase-based statistical machine translation with pivot languages. In *Proceedings of 5th International Workshop on Spoken Language Translation (IWSLT 2008)*, pages 143–149, Honolulu, HI, USA.

Yun Chen, Yang Liu, Yong Cheng, and Victor OK Li. 2017. A teacher-student framework for zero-

resource neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1925–1935.

Yong Cheng, Qian Yang, Yang Liu, Maosong Sun, and Wei Xu. 2017. Joint training for pivot-based neural machine translation. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 3974–3980. AAAI Press.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. In *Proceedings of 6th International Conference on Learning Representations (ICLR 2018)*.

Marta R Costa-Jussà, Carlos Henríquez, and Rafael E Banchs. 2011. Enhancing scarce-resource language translation through pivot combinations. In *Proceedings of 5th International Joint Conference on Natural Language Processing (IJCNLP 2011)*, pages 1361–1365, Chiang Mai, Thailand.

Adrià De Gispert and Jose B Marino. 2006. Catalan-english statistical machine translation without parallel corpus: bridging through spanish. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC 2006)*, pages 65–68, Genoa, Italy.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500, Brussels, Belgium.

Orhan Firat, Baskaran Sankaran, Yaser Al-Onaizan, Fatos T Yarman Vural, and Kyunghyun Cho. 2016. Zero-resource translation with multi-lingual neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 268–277.

Thanh-Le Ha, Jan Niehues, and Alexander Waibel. 2017. Effective strategies in zero-shot neural machine translation. In *14th International Workshop on Spoken Language Translation*.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2016)*, pages 1367–1377.

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association of Computational Linguistics (TACL)*, 5(1):339–351.

Manuel Kauers, Stephan Vogel, Christian Fügen, and Alex Waibel. 2002. Interlingua based statistical machine translation. In *Seventh International Conference on Spoken Language Processing*.

Yunsu Kim, Yingbo Gao, and Hermann Ney. 2019. Effective cross-lingual transfer of neural machine translation models without shared vocabularies. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1246–1257, Florence, Italy.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada. Association for Computational Linguistics.

Tom Kocmi and Ondřej Bojar. 2018. Trivial transfer learning for low-resource neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 244–252.

Surafel M Lakew, Quintino F Lotito, Matteo Negri, Marco Turchi, and Marcello Federico. 2017. Improving zero-shot translation of low-resource languages. In *14th International Workshop on Spoken Language Translation*.

Yichao Lu, Phillip Keung, Faisal Ladhak, Vikas Bhardwaj, Shaonan Zhang, and Jason Sun. 2018. A neural interlingua for multilingual machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 84–92.

Benjamin Marie, Haipeng Sun, Rui Wang, Kehai Chen, Atsushi Fujita, Masao Utiyama, and Eiichiro Sumita. 2019. NICT's unsupervised neural and statistical machine translation systems for the WMT19 news translation task. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 294–301, Florence, Italy.

Akiva Miura, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2015. Improving pivot translation by remembering the pivot. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 573–577.

Toan Q Nguyen and David Chiang. 2017. Transfer learning across low-resource, related languages for neural machine translation. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 296–301.

Shuo Ren, Wenhu Chen, Shujie Liu, Mu Li, Ming Zhou, and Shuai Ma. 2018. Triangular architecture for rare language translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 56–65.

Holger Schwenk. 2018. Filtering and mining parallel data in a joint multilingual space. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 228–234.

Holger Schwenk and Matthijs Douze. 2017. Learning joint multilingual sentence representations with neural machine translation. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 157–167.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1715–1725.

Lierni Sestorain, Massimiliano Ciaramita, Christian Buck, and Thomas Hofmann. 2018. Zero-shot dual machine translation. *arXiv preprint arXiv:1805.10338*.

Masao Utiyama and Hitoshi Isahara. 2007. A comparison of pivot methods for phrase-based statistical machine translation. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 484–491.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM.

Hua Wu and Haifeng Wang. 2007. Pivot language approach for phrase-based statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 856–863.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1006–1011.

Samira Tofighi Zahabi, Somayeh Bakhshaei, and Shahram Khadivi. 2013. Using context vectors in improving a machine translation system with bridge language. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 318–322.

Hao Zheng, Yong Cheng, and Yang Liu. 2017. Maximum expected likelihood estimation for zero-resource neural machine translation. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 4251–4257. AAAI Press.

Xiaoning Zhu, Zhongjun He, Hua Wu, Conghui Zhu, Haifeng Wang, and Tiejun Zhao. 2014. Improving pivot-based statistical machine translation by pivoting the co-occurrence count of phrase pairs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1665–1675.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1568–1575.

# Context-Aware Monolingual Repair for Neural Machine Translation

**Elena Voita**[1,2] **Rico Sennrich**[4,3] **Ivan Titov**[3,2]

[1]Yandex, Russia [2]University of Amsterdam, Netherlands
[3]University of Edinburgh, Scotland [4]University of Zurich, Switzerland
lena-voita@yandex-team.ru
sennrich@cl.uzh.ch ititov@inf.ed.ac.uk

## Abstract

Modern sentence-level NMT systems often produce plausible translations of isolated sentences. However, when put in context, these translations may end up being inconsistent with each other. We propose a monolingual DocRepair model to correct inconsistencies between sentence-level translations. DocRepair performs automatic post-editing on a sequence of sentence-level translations, refining translations of sentences in context of each other. For training, the DocRepair model requires only monolingual document-level data in the target language. It is trained as a monolingual sequence-to-sequence model that maps inconsistent groups of sentences into consistent ones. The consistent groups come from the original training data; the inconsistent groups are obtained by sampling round-trip translations for each isolated sentence. We show that this approach successfully imitates inconsistencies we aim to fix: using contrastive evaluation, we show large improvements in the translation of several contextual phenomena in an English→Russian translation task, as well as improvements in the BLEU score. We also conduct a human evaluation and show a strong preference of the annotators to corrected translations over the baseline ones. Moreover, we analyze which discourse phenomena are hard to capture using monolingual data only.[1]

## 1 Introduction

Machine translation has made remarkable progress, and studies claiming it to reach a human parity are starting to appear (Hassan et al., 2018). However, when evaluating translations of the whole documents rather than isolated sentences, human raters show a stronger preference for human over machine translation (Läubli et al., 2018). These findings emphasize the need to shift towards context-aware machine translation both from modeling and evaluation perspective.

Most previous work on context-aware NMT assumed that either all the bilingual data is available at the document level (Jean et al., 2017; Wang et al., 2017; Tiedemann and Scherrer, 2017; Bawden et al., 2018; Voita et al., 2018; Maruf and Haffari, 2018; Agrawal et al., 2018; Kuang et al., 2018; Miculicich et al., 2018) or at least its fraction (Voita et al., 2019). But in practical scenarios, document-level parallel data is often scarce, which is one of the challenges when building a context-aware system.

We introduce an approach to context-aware machine translation using only monolingual document-level data. In our setting, a separate monolingual sequence-to-sequence model (DocRepair) is used to correct sentence-level translations of adjacent sentences. The key idea is to use monolingual data to imitate typical inconsistencies between context-agnostic translations of isolated sentences. The DocRepair model is trained to map inconsistent groups of sentences into consistent ones. The consistent groups come from the original training data; the inconsistent groups are obtained by sampling round-trip translations for each isolated sentence.

To validate the performance of our model, we use three kinds of evaluation: the BLEU score, contrastive evaluation of translation of several discourse phenomena (Voita et al., 2019), and human evaluation. We show strong improvements for all metrics.

We analyze which discourse phenomena are hard to capture using monolingual data only. Using contrastive test sets for targeted evaluation of several contextual phenomena, we compare the performance of the models trained on round-trip

[1]The code and data sets (including round-trip translations) are available at https://github.com/lena-voita/good-translation-wrong-in-context.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 877–886, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

translations and genuine document-level parallel data. Among the four phenomena in the test sets we use (deixis, lexical cohesion, VP ellipsis and ellipsis which affects NP inflection) we find VP ellipsis to be the hardest phenomenon to be captured using round-trip translations.

Our key contributions are as follows:

- we introduce the first approach to context-aware machine translation using only monolingual document-level data;
- our approach shows substantial improvements in translation quality as measured by BLEU, targeted contrastive evaluation of several discourse phenomena and human evaluation;
- we show which discourse phenomena are hard to capture using monolingual data only.

## 2 Our Approach: Document-level Repair

We propose a monolingual DocRepair model to correct inconsistencies between sentence-level translations of a context-agnostic MT system. It does not use any states of a trained MT model whose outputs it corrects and therefore can in principle be trained to correct translations from any black-box MT system.

The DocRepair model requires only monolingual document-level data in the target language. It is a monolingual sequence-to-sequence model that maps inconsistent groups of sentences into consistent ones. Consistent groups come from monolingual document-level data. To obtain inconsistent groups, each sentence in a group is replaced with its round-trip translation produced in isolation from context. More formally, forming a training minibatch for the DocRepair model involves the following steps (see also Figure 1):

1. sample several groups of sentences from the monolingual data;
2. for each sentence in a group, (i) translate it using a target-to-source MT model, (ii) sample a translation of this back-translated sentence in the source language using a source-to-target MT model;
3. using these round-trip translations of isolated sentences, form an inconsistent version of the initial groups;

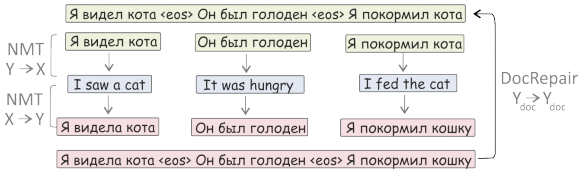


Figure 1: Training procedure of DocRepair. First, round-trip translations of individual sentences are produced to form an inconsistent text fragment (in the example, both genders of the speaker and the cat became inconsistent). Then, a repair model is trained to produce an original text from the inconsistent one.

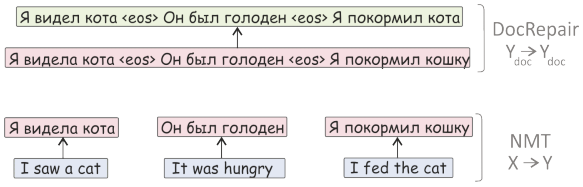


Figure 2: The process of producing document-level translations at test time is two-step: (1) sentences are translated independently using a sentence-level model, (2) DocRepair model corrects translation of the resulting text fragment.

4. use inconsistent groups as input for the DocRepair model, consistent ones as output.

At test time, the process of getting document-level translations is two-step (Figure 2):

1. produce translations of isolated sentences using a context-agnostic MT model;
2. apply the DocRepair model to a sequence of context-agnostic translations to correct inconsistencies between translations.

In the scope of the current work, the DocRepair model is the standard sequence-to-sequence Transformer. Sentences in a group are concatenated using a reserved token-separator between sentences.[2] The Transformer is trained to correct these long inconsistent pseudo-sentences into consistent ones. The token-separator is then removed from corrected translations.

## 3 Evaluation of Contextual Phenomena

We use contrastive test sets for evaluation of discourse phenomena for English-Russian by Voita et al. (2019). These test sets allow for testing different kinds of phenomena which, as we show, can

[2] In preliminary experiments, we observed that this performs better than concatenating sentences without a separator.

| | total | distance 1 | 2 | 3 |
|---|---|---|---|---|
| **deixis** | 3000 | 1000 | 1000 | 1000 |
| **lex. cohesion** | 2000 | 855 | 630 | 515 |
| **ellipsis (infl.)** | 500 | | | |
| **ellipsis (VP)** | 500 | | | |

Table 1: Size of test sets: total number of test instances and with regard to the distance between sentences requiring consistency (in the number of sentences). For ellipsis, the two sets correspond to whether a model has to predict correct NP inflection, or correct verb sense (VP ellipsis).

be captured from monolingual data with varying success. In this section, we provide test sets statistics and briefly describe the tested phenomena. For more details, the reader is referred to Voita et al. (2019).

### 3.1 Test sets

There are four test sets in the suite. Each test set contains contrastive examples. It is specifically designed to test the ability of a system to adapt to contextual information and handle the phenomenon under consideration. Each test instance consists of a true example (a sequence of sentences and their reference translation from the data) and several contrastive translations which differ from the true one only in one specific aspect. All contrastive translations are correct and plausible translations at the sentence level, and only context reveals the inconsistencies between them. The system is asked to score each candidate translation, and we compute the system accuracy as the proportion of times the true translation is preferred to the contrastive ones. Test set statistics are shown in Table 1. The suites for deixis and lexical cohesion are split into development and test sets, with 500 examples from each used for validation purposes and the rest for testing. Convergence of both consistency scores on these development sets and BLEU score on a general development set are used as early stopping criteria in models training. For ellipsis, there is no dedicated development set, so we evaluate on all the ellipsis data and do not use it for development.

### 3.2 Phenomena overview

**Deixis** Deictic words or phrases, are referential expressions whose denotation depends on context. This includes personal deixis ("I", "you"), place deixis ("here", "there"), and discourse deixis, where parts of the discourse are referenced ("that's a good question"). The test set examples are all related to person deixis, specifically the T-V distinction between informal and formal you (Latin "tu" and "vos") in the Russian translations, and test for consistency in this respect.

**Ellipsis** Ellipsis is the omission from a clause of one or more words that are nevertheless understood in the context of the remaining elements. In machine translation, elliptical constructions in the source language pose a problem in two situations. First, if the target language does not allow the same types of ellipsis, requiring the elided material to be predicted from context. Second, if the elided material affects the syntax of the sentence. For example, in Russian the grammatical function of a noun phrase, and thus its inflection, may depend on the elided verb, or, conversely, the verb inflection may depend on the elided subject.

There are two different test sets for ellipsis. One contains examples where a morphological form of a noun group in the last sentence can not be understood without context beyond the sentence level ("ellipsis (infl.)" in Table 1). Another includes cases of verb phrase ellipsis in English, which does not exist in Russian, thus requires predicting the verb when translating into Russian ("ellipsis (VP)" in Table 1).

**Lexical cohesion** The test set focuses on reiteration of named entities. Where several translations of a named entity are possible, a model has to prefer consistent translations over inconsistent ones.

## 4 Experimental Setup

### 4.1 Data preprocessing

We use the publicly available OpenSubtitles2018 corpus (Lison et al., 2018) for English and Russian. For a fair comparison with previous work, we train the baseline MT system on the data released by Voita et al. (2019). Namely, our MT system is trained on 6m instances. These are sentence pairs with a relative time overlap of subtitle frames between source and target language subtitles of at least 0.9.

We gathered 30m groups of 4 consecutive sentences as our monolingual data. We used only documents not containing groups of sentences from general development and test sets as well as from contrastive test sets. The main results we report

are for the model trained on all 30m fragments.

We use the tokenization provided by the corpus and use `multi-bleu.perl`[3] on lowercased data to compute BLEU score. We use beam search with a beam of 4.

Sentences were encoded using byte-pair encoding (Sennrich et al., 2016b), with source and target vocabularies of about 32000 tokens. Translation pairs were batched together by approximate sequence length. Each training batch contained a set of translation pairs containing approximately 15000[4] source tokens. It has been shown that Transformer's performance depends heavily on batch size (Popel and Bojar, 2018), and we chose a large batch size to ensure the best performance. In training context-aware models, for early stopping we use both convergence in BLEU score on the general development set and scores on the consistency development sets. After training, we average the 5 latest checkpoints.

### 4.2 Models

The baseline model, the model used for back-translation, and the DocRepair model are all *Transformer base* models (Vaswani et al., 2017). More precisely, the number of layers is $N = 6$ with $h = 8$ parallel attention layers, or heads. The dimensionality of input and output is $d_{model} = 512$, and the inner-layer of a feed-forward networks has dimensionality $d_{ff} = 2048$. We use regularization as described in Vaswani et al. (2017).

As a second baseline, we use the two-pass CADec model (Voita et al., 2019). The first pass produces sentence-level translations. The second pass takes both the first-pass translation and representations of the context sentences as input and returns contextualized translations. CADec requires document-level parallel training data, while DocRepair only needs monolingual training data.

### 4.3 Generating round-trip translations

On the selected 6m instances we train sentence-level translation models in both directions. To create training data for DocRepair, we proceed as follows. The Russian monolingual data is first translated into English, using the Russian→English model and beam search with beam size of 4. Then, we use the English→Russian model to sample translations with temperature of 0.5. For each sentence, we precompute 20 sampled translations and randomly choose one of them when forming a training minibatch for DocRepair. Also, in training, we replace each token in the input with a random one with the probability of 10%.

[3] https://github.com/moses-smt/mosesdecoder/tree/master/scripts/generic

[4] This can be reached by using several of GPUs or by accumulating the gradients for several batches and then making an update.

| model | BLEU |
|---|---|
| baseline | 33.91 |
| CADec | 33.86 |
| sentence-level repair | 34.12 |
| DocRepair | **34.60** |

Table 2: BLEU scores. For CADec, the original implementation was used.

### 4.4 Optimizer

As in Vaswani et al. (2017), we use the Adam optimizer (Kingma and Ba, 2015), the parameters are $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\varepsilon = 10^9$. We vary the learning rate over the course of training using the formula:

$$l_{rate} = scale \cdot \min(step\_num^{0.5}, step\_num \cdot warmup\_steps^{1.5}),$$

where $warmup\_steps = 16000$ and $scale = 4$.

## 5 Results

### 5.1 General results

The BLEU scores are provided in Table 2 (we evaluate translations of 4-sentence fragments). To see which part of the improvement is due to fixing agreement between sentences rather than simply sentence-level post-editing, we train the same repair model at the sentence level. Each sentence in a group is now corrected separately, then they are put back together in a group. One can see that most of the improvement comes from accounting for extra-sentential dependencies. DocRepair outperforms the baseline and CADec by 0.7 BLEU, and its sentence-level repair version by 0.5 BLEU.

### 5.2 Consistency results

Scores on the phenomena test sets are provided in Table 3. For deixis, lexical cohesion and ellipsis (infl.) we see substantial improvements over both the baseline and CADec. The largest improvement over CADec (22.5 percentage points)

| model | deixis | lex. c | ell. infl. | ell. VP |
|---|---|---|---|---|
| baseline | 50.0 | 45.9 | 53.0 | 28.4 |
| CADec | 81.6 | 58.1 | 72.2 | **80.0** |
| DocRepair | **91.8** | **80.6** | **86.4** | 75.2 |
| | +10.2 | +22.5 | +14.4 | -4.8 |

Table 3: Results on contrastive test sets for specific contextual phenomena (deixis, lexical consistency, ellipsis (inflection), and VP ellipsis).

(a) **EN** No one believed me. But she did.

**RU** Мне никто не верил. Но она сказала.

(b)**RU** Никто мне не верил. Но она верила.

**EN** No one believed me. But she believed.

**RU** Мне никто не верил. Но она поверила.

Figure 3: (a) Example of a discrepancy caused by VP ellipsis: correct meaning is "believe", but MT produces сказала ("say"). (b) Example of producing round-trip translations. From top to bottom: target, first translation, round-trip translation. When translating from Russian, main verbs are unlikely to be translated into auxiliary ones in English, and VP ellipsis is not present.

is for lexical cohesion. However, there is a drop of almost 5 percentage points for VP ellipsis. We hypothesize that this is because it is hard to learn to correct inconsistencies in translations caused by VP ellipsis relying on monolingual data alone. Figure 3(a) shows an example of inconsistency caused by VP ellipsis in English. There is no VP ellipsis in Russian, and when translating auxiliary "did" the model has to guess the main verb. Figure 3(b) shows steps of generating round-trip translations for the target side of the previous example. When translating from Russian, main verbs are unlikely to be translated as the auxiliary "do" in English, and hence the VP ellipsis is rarely present on the English side. This implies the model trained using the round-trip translations will not be exposed to many VP ellipsis examples in training. We discuss this further in Section 6.2.

Table 4 provides scores for deixis and lexical cohesion separately for different distances between sentences requiring consistency. It can be seen, that the performance of DocRepair degrades less than that of CADec when the distance between sentences requiring consistency gets larger.

| | | distance | | |
|---|---|---|---|---|
| | total | 1 | 2 | 3 |
| **deixis** | | | | |
| baseline | 50.0 | 50.0 | 50.0 | 50.0 |
| CADec | 81.6 | 84.6 | 84.4 | 75.9 |
| DocRepair | **91.8** | **94.8** | **93.1** | **87.7** |
| | + 10.2 | +10.2 | +8.7 | +11.8 |
| **lexical cohesion** | | | | |
| baseline | 45.9 | 46.1 | 45.9 | 45.4 |
| CADec | 58.1 | 63.2 | 52.0 | 56.7 |
| DocRepair | **80.6** | **83.0** | **78.5** | **79.4** |
| | + 22.5 | +20.2 | +26.5 | +22.3 |

Table 4: Detailed accuracy on deixis and lexical cohesion test sets.

| all | equal | better | worse |
|---|---|---|---|
| 700 | 367 | 242 | 90 |
| 100% | 52% | 35% | 13% |

Table 5: Human evaluation results, comparing DocRepair with baseline.

### 5.3 Human evaluation

We conduct a human evaluation on random 700 examples from our general test set. We picked only examples where a DocRepair translation is not a full copy of the baseline one.[5]

The annotators were provided an original group of sentences in English and two translations: baseline context-agnostic one and the one corrected by the DocRepair model. Translations were presented in random order with no indication which model they came from. The task is to pick one of the three options: (1) the first translation is better, (2) the second translation is better, (3) the translations are of equal quality. The annotators were asked to avoid the third answer if they are able to give preference to one of the translations. No other guidelines were given.

The results are provided in Table 5. In about 52% of the cases annotators marked translations as having equal quality. Among the cases where one of the translations was marked better than the other, the DocRepair translation was marked better in 73% of the cases. This shows a strong preference of the annotators for corrected translations over the baseline ones.

[5] As we further discuss in Section 7, DocRepair does not change the base translation at all in about 20% of the cases.

| | BLEU | deixis | lex. c. | ellipsis |
|---|---|---|---|---|
| 2.5m | 34.15 | 89.2 | 75.5 | 81.8 / 71.6 |
| 5m | 34.44 | 90.3 | 77.7 | 83.6 / 74.0 |
| 30m | **34.60** | **91.8** | **80.6** | **86.4 / 75.2** |

Table 6: Results for DocRepair trained on different amount of data. For ellipsis, we show inflection/VP scores.

| data | deixis | lex. c. | ell. infl. | ell. VP |
|---|---|---|---|---|
| one-way | 85.4 | 63.4 | 79.8 | 73.4 |
| round-trip | 84.0 | 61.7 | 78.4 | 67.8 |

Table 7: Consistency scores for the DocRepair model trained on 2.5m instances, among which 1.5m are parallel instances. Compare round-trip and one-way translations of the parallel part.

| data | BLEU | deixis | lex. c. | ellipsis |
|---|---|---|---|---|
| from mon. | 34.15 | 89.2 | 75.5 | 81.8 / 71.6 |
| from par. | 33.70 | 84.0 | 61.7 | 78.4 / 67.8 |

Table 8: DocRepair trained on 2.5m instances, either randomly chosen from monolingual data or from the part where each utterance in a group has a translation.

## 6 Varying Training Data

In this section, we discuss the influence of the training data chosen for document-level models. In all experiments, we used the DocRepair model.

### 6.1 The amount of training data

Table 6 provides BLEU and consistency scores for the DocRepair model trained on different amount of data. We see that even when using a dataset of moderate size (e.g., 5m fragments) we can achieve performance comparable to the model trained on a large amount of data (30m fragments). Moreover, we notice that deixis scores are less sensitive to the amount of training data than lexical cohesion and ellipsis scores. The reason might be that, as we observed in our previous work (Voita et al., 2019), inconsistencies in translations due to the presence of deictic words and phrases are more frequent in this dataset than other types of inconsistencies. Also, as we show in Section 7, this is the phenomenon the model learns faster in training.

### 6.2 One-way vs round-trip translations

In this section, we discuss the limitations of using only monolingual data to model inconsistencies between sentence-level translations. In Section 5.2 we observed a drop in performance on VP ellipsis for DocRepair compared to CADec, which was trained on parallel data. We hypothesized that this is due to the differences between one-way and round-trip translations, and now we test this hypothesis. To do so, we fix the dataset and vary the way in which the input for DocRepair is generated: round-trip or one-way translations. The latter assumes that document-level data is parallel, and translations are sampled from the source side of the sentences in a group rather than from their back-translations. For parallel data, we take 1.5m parallel instances which were used for CADec training and add 1m instances from our monolingual data. For segments in the parallel part, we either sample translations from the source side or use round-trip translations. The results are provided in Table 7.

The model trained on one-way translations is slightly better than the one trained on round-trip translations. As expected, VP ellipsis is the hardest phenomena to be captured using round-trip translations, and the DocRepair model trained on one-way translated data gains 6% accuracy on this test set. This shows that the DocRepair model benefits from having access to non-synthetic English data. This results in exposing DocRepair at training time to Russian translations which suffer from the same inconsistencies as the ones it will have to correct at test time.

### 6.3 Filtering: monolingual (no filtering) or parallel

Note that the scores of the DocRepair model trained on 2.5m instances randomly chosen from monolingual data (Table 6) are different from the ones for the model trained on 2.5m instances combined from parallel and monolingual data (Table 7). For convenience, we show these two in Table 8.

The domain, the dataset these two data samples were gathered from, and the way we generated training data for DocRepair (round-trip translations) are all the same. The only difference lies in how the data was filtered. For parallel data, as in the previous work (Voita et al., 2018), we picked only sentence pairs with large relative time overlap of subtitle frames between source-language and target-language subtitles. This is necessary to

ensure the quality of translation data: one needs groups of consecutive sentences in the target language where every sentence has a reliable translation.

Table 8 shows that the quality of the model trained on data which came from the parallel part is worse than the one trained on monolingual data. This indicates that requiring each sentence in a group to have a reliable translation changes the distribution of the data, which might be not beneficial for translation quality and provides extra motivation for using monolingual data.

## 7 Learning Dynamics

Let us now look into how the process of DocRepair training progresses. Figure 4a shows how the BLEU scores with the reference translation and with the baseline context-agnostic translation (i.e. the input for the DocRepair model) are changing during training. First, the model quickly learns to copy baseline translations: the BLEU score with the baseline is very high. Then it gradually learns to change them, which leads to an improvement in BLEU with the reference translation and a drop in BLEU with the baseline. Importantly, the model is reluctant to make changes: the BLEU score between translations of the converged model and the baseline is 82.5. We count the number of changed sentences in every 4-sentence fragment in the test set and plot the histogram in Figure 4b. In over than 20% of the cases the model has not changed base translations at all. In almost 40%, it modified only one sentence and left the remaining 3 sentences unchanged. The model changed more than half sentences in a group in only 14% of the cases. Several examples of the DocRepair translations are shown in Figure 6.

Figure 5 shows how consistency scores are changing in training.[6] For deixis, the model achieves the final quality quite quickly; for the rest, it needs a large number of training steps to converge.

## 8 Related Work

Our work is most closely related to two lines of research: automatic post-editing (APE) and document-level machine translation.

[6]Deixis and lexical cohesion scores are evaluated on the development sets which were used in training for the stopping criteria. Ellipsis test sets were not used at training time; the scores are shown here only for visualization purposes.

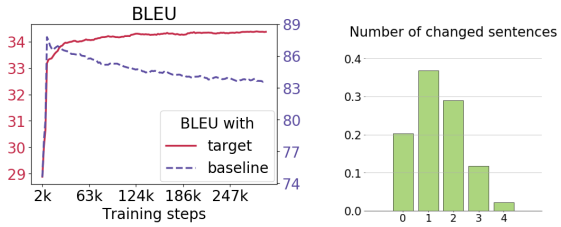


(a) (b)

Figure 4: (a) BLEU scores progression in training. BLEU evaluated with the target translations and with the context-agnostic baseline translations (which DocRepair learns to correct). (b) Distribution in the test set of the number of changed sentences in 4-sentence fragments.

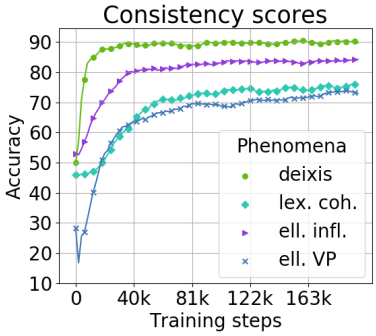


Figure 5: Consistency scores progression in training.

### 8.1 Automatic post-editing

Our model can be regarded as an automatic post-editing system – a system designed to fix systematic MT errors that is decoupled from the main MT system. Automatic post-editing has a long history, including rule-based (Knight and Chander, 1994), statistical (Simard et al., 2007) and neural approaches (Junczys-Dowmunt and Grundkiewicz, 2016; Pal et al., 2016; Freitag et al., 2019).

In terms of architectures, modern approaches use neural sequence-to-sequence models, either multi-source architectures that consider both the original source and the baseline translation (Junczys-Dowmunt and Grundkiewicz, 2016; Pal et al., 2016), or monolingual repair systems, as in Freitag et al. (2019), which is concurrent work to ours. True post-editing datasets are typically small and expensive to create (Specia et al., 2017), hence synthetic training data has been created that uses original monolingual data as output for the sequence-to-sequence model, paired with an automatic back-translation (Sennrich et al., 2016a) and/or round-trip translation as its input(s) (Junczys-Dowmunt and Grundkiewicz, 2016; Fre-

(a) **EN** Don't you have some souls to save? It's a nice brooch Rita's got. Did you get it from the stolen property, too? How long's it been going on, you and her?

**RU** Разве тебе не нужно спасать души? У Риты красивая брошь. Ты тоже забрал его из украденной собственности? Как долго это продолжается, ты и она?

**$\overline{\text{RU}}$** Разве тебе не нужно спасать души? У Риты красивая брошь. Ты тоже взял ее из украденной собственности? Как долго это продолжается, ты и она?

---

(b) **EN** Y'all working together. Glenn, I've thought for the longest time that you're one of the best in the business. Well, thank you, Rayna. And I love these demos you sent over.

**RU** Вы работаете вместе. Гленн, я долгое время думал, что ты один из лучших в своем деле. Спасибо, Рейна. И мне нравятся эти демо, которые ты прислал.

**$\overline{\text{RU}}$** Вы работаете вместе. Гленн, я долгое время думала, что ты один из лучших в своем деле. Спасибо, Рейна. И мне нравятся эти демо, которые ты прислал.

---

(c) **EN** Who's the new guy? I got to go, all right? Yeah, me too. All right.

**RU** Кто этот новичок? Мне нужно идти, хорошо? Да, я тоже. Хорошо.

**$\overline{\text{RU}}$** Кто этот новичок? Мне нужно идти, хорошо? Да, мне тоже. Хорошо.

Figure 6: Examples of the DocRepair translations. First is the baseline translation, then – corrected by the DocRepair. The differences between translations are underlined. (a) corrected wrong translation of "it": violated gender agreement with the antecedent. (b) corrected wrong gender (marked on a verb): from the third sentence it's clear that the speaker in the second one is feminine (Rayna), but the baseline translation was masculine. (c) corrected wrong morphological form of the pronoun, which was not understood with the elided verb in the third sentence.

itag et al., 2019).

While previous work on automatic post-editing operated on the sentence level, the main novelty of this work is that our DocRepair model operates on groups of sentences and is thus able to fix consistency errors caused by the context-agnostic baseline MT system. We consider this strategy of sentence-level baseline translation and context-aware monolingual repair attractive when parallel document-level data is scarce.

For training, the DocRepair model only requires monolingual document-level data. While we create synthetic training data via round-trip translation similarly to earlier work (Junczys-Dowmunt and Grundkiewicz, 2016; Freitag et al., 2019), note that we purposefully use sentence-level MT systems for this to create the types of consistency errors that we aim to fix with the context-aware DocRepair model. Not all types of consistency errors that we want to fix emerge from a round-trip translation, so access to parallel document-level data can be useful (Section 6.2).

### 8.2 Document-level NMT

Neural models of MT that go beyond the sentence-level are an active research area (Jean et al., 2017; Wang et al., 2017; Tiedemann and Scherrer, 2017; Bawden et al., 2018; Voita et al., 2018; Maruf and Haffari, 2018; Agrawal et al., 2018; Miculicich et al., 2018; Kuang et al., 2018; Voita et al., 2019). Typically, the main MT system is modified to take additional context as its input. One limitation of these approaches is that they assume that parallel document-level training data is available.

Closest to our work are two-pass models for document-level NMT (Xiong et al., 2019; Voita et al., 2019), where a second, context-aware model takes the translation and hidden representations of the sentence-level first-pass model as its input. The second-pass model can in principle be trained on a subset of the parallel training data (Voita et al., 2019), somewhat relaxing the assumption that all training data is at the document level.

Our work is different from this previous work in two main respects. Firstly, we show that consistency can be improved with only monolingual document-level training data. Secondly, the

DocRepair model is decoupled from the first-pass MT system, which improves its portability.

## 9 Conclusions

We introduce the first approach to context-aware machine translation using only monolingual document-level data. We propose a monolingual DocRepair model to correct inconsistencies between sentence-level translations. The model performs automatic post-editing on a sequence of sentence-level translations, refining translations of sentences in context of each other. Our approach results in substantial improvements in translation quality as measured by BLEU, targeted contrastive evaluation of several discourse phenomena and human evaluation. Moreover, we perform error analysis and detect which discourse phenomena are hard to capture using only monolingual document-level data. While in the current work we used text fragments of 4 sentences, in future work we would like to consider longer contexts.

## Acknowledgments

We would like to thank the anonymous reviewers for their comments. The authors also thank David Talbot and Yandex Machine Translation team for helpful discussions and inspiration. Ivan Titov acknowledges support of the European Research Council (ERC StG BroadSem 678254) and the Dutch National Science Foundation (NWO VIDI 639.022.518). Rico Sennrich acknowledges support from the Swiss National Science Foundation (105212_169888), the European Union's Horizon 2020 research and innovation programme (grant agreement no 825460), and the Royal Society (NAF\R1\180122).

## References

Ruchit Agrawal, Turchi Marco, and Negri Matteo. 2018. Contextual Handling in Neural Machine Translation: Look Behind, Ahead and on Both Sides.

Rachel Bawden, Rico Sennrich, Alexandra Birch, and Barry Haddow. 2018. Evaluating Discourse Phenomena in Neural Machine Translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1304–1313, New Orleans, USA. Association for Computational Linguistics.

Markus Freitag, Isaac Caswell, and Scott Roy. 2019. Ape at scale and its implications on mt evaluation biases. In *Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers)*, pages 34–44, Florence, Italy. Association for Computational Linguistics.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, Shujie Liu, Tie-Yan Liu, Renqian Luo, Arul Menezes, Tao Qin, Frank Seide, Xu Tan, Fei Tian, Lijun Wu, Shuangzhi Wu, Yingce Xia, Dongdong Zhang, Zhirui Zhang, and Ming Zhou. 2018. Achieving human parity on automatic chinese to english news translation. In *arXiv:1803.05567*. ArXiv: 1803.05567.

Sebastien Jean, Stanislas Lauly, Orhan Firat, and Kyunghyun Cho. 2017. Does Neural Machine Translation Benefit from Larger Context? In *arXiv:1704.05135*. ArXiv: 1704.05135.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2016. Log-linear combinations of monolingual and bilingual neural machine translation models for automatic post-editing. In *Proceedings of the First Conference on Machine Translation*, pages 751–758, Berlin, Germany. Association for Computational Linguistics.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representation (ICLR 2015)*.

Kevin Knight and Ishwar Chander. 1994. Automated postediting of documents. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (Vol. 1)*, AAAI '94, pages 779–784, Menlo Park, CA, USA. American Association for Artificial Intelligence.

Shaohui Kuang, Deyi Xiong, Weihua Luo, and Guodong Zhou. 2018. Modeling coherence for neural machine translation with dynamic and topic caches. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 596–606, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Samuel Läubli, Rico Sennrich, and Martin Volk. 2018. Has Machine Translation Achieved Human Parity? A Case for Document-level Evaluation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4791–4796. Association for Computational Linguistics.

Pierre Lison, Jörg Tiedemann, and Milen Kouylekov. 2018. Opensubtitles2018: Statistical rescoring of sentence alignments in large, noisy parallel corpora. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan.

Sameen Maruf and Gholamreza Haffari. 2018. Document context neural machine translation with memory networks. In *Proceedings of the 56th Annual*

*Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1275–1284, Melbourne, Australia. Association for Computational Linguistics.

Lesly Miculicich, Dhananjay Ram, Nikolaos Pappas, and James Henderson. 2018. Document-level neural machine translation with hierarchical attention networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2947–2954, Brussels, Belgium. Association for Computational Linguistics.

Santanu Pal, Sudip Kumar Naskar, Mihaela Vela, and Josef van Genabith. 2016. A neural network based approach to automatic post-editing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 281–286, Berlin, Germany. Association for Computational Linguistics.

Martin Popel and Ondrej Bojar. 2018. Training Tips for the Transformer Model. pages 43–70.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Michel Simard, Cyril Goutte, and Pierre Isabelle. 2007. Statistical phrase-based post-editing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 508–515, Rochester, New York. Association for Computational Linguistics.

Lucia Specia, Kim Harris, Frédéric Blain, Aljoscha Burchardt, Vivien Macketanz, Inguna Skadiņa, Matteo Negri, , and Marco Turchi. 2017. Translation quality and productivity: A study on rich morphology languages. In *Machine Translation Summit XVI*, pages 55–71. Asia-Pacific Association for Machine Translation.

Jörg Tiedemann and Yves Scherrer. 2017. Neural Machine Translation with Extended Context. In *Proceedings of the Third Workshop on Discourse in Machine Translation*, DISCOMT'17, pages 82–92, Copenhagen, Denmark. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*, Los Angeles.

Elena Voita, Rico Sennrich, and Ivan Titov. 2019. When a good translation is wrong in context: Context-aware machine translation improves on deixis, ellipsis, and lexical cohesion. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1198–1212, Florence, Italy. Association for Computational Linguistics.

Elena Voita, Pavel Serdyukov, Rico Sennrich, and Ivan Titov. 2018. Context-aware neural machine translation learns anaphora resolution. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1264–1274, Melbourne, Australia. Association for Computational Linguistics.

Longyue Wang, Zhaopeng Tu, Andy Way, and Qun Liu. 2017. Exploiting Cross-Sentence Context for Neural Machine Translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, EMNLP'17, pages 2816–2821, Denmark, Copenhagen. Association for Computational Linguistics.

Hao Xiong, Zhongjun He, Hua Wu, and Haifeng Wang. 2019. Modeling coherence for discourse neural machine translation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):7338–7345.

# Multi-Granularity Self-Attention for Neural Machine Translation

**Jie Hao**[*]
Florida State University
haoj8711@gmail.com

**Xing Wang**
Tencent AI Lab
brightxwang@tencent.com

**Shuming Shi**
Tencent AI Lab
shumingshi@tencent.com

**Jinfeng Zhang**
Florida State University
jinfeng@stat.fsu.edu

**Zhaopeng Tu**
Tencent AI Lab
zptu@tencent.com

## Abstract

Current state-of-the-art neural machine translation (NMT) uses a deep multi-head self-attention network with no explicit phrase information. However, prior work on statistical machine translation has shown that extending the basic translation unit from words to phrases has produced substantial improvements, suggesting the possibility of improving NMT performance from explicit modeling of phrases. In this work, we present *multi-granularity self-attention* (MG-SA): a neural network that combines multi-head self-attention and phrase modeling. Specifically, we train several attention heads to attend to phrases in either n-gram or syntactic formalism. Moreover, we exploit interactions among phrases to enhance the strength of structure modeling – a commonly-cited weakness of self-attention. Experimental results on WMT14 English-to-German and NIST Chinese-to-English translation tasks show the proposed approach consistently improves performance. Targeted linguistic analysis reveals that MG-SA indeed captures useful phrase information at various levels of granularities.

## 1 Introduction

Recently, TRANSFORMER (Vaswani et al., 2017), implemented as deep multi-head self-attention networks (SANs), has become the state-of-the-art neural machine translation (NMT) model in recent years. The popularity of SANs lies in its high parallelization in computation, and flexibility in modeling dependencies regardless of distance by explicitly attending to all the signals.

More recently, an in-depth study (Raganato and Tiedemann, 2018) reveals that SANs generally focus on disperse words and ignore continuous phrase patterns, which have proven essential in both statistical machine translation (SMT, Koehn et al., 2003; Chiang, 2005; Liu et al., 2006) and NMT (Eriguchi et al., 2016; Wang et al., 2017; Yang et al., 2018; Zhao et al., 2018).

To alleviate this problem, in this work we propose *multi-granularity self-attention* (MG-SA), which offers SANs the ability to model phrases and meanwhile maintain their simplicity and flexibility. The starting point for our approach is an observation: the power of multiple heads in SANs is not fully exploited. For example, Li et al. (2018) show that different attention heads generally attend to the same positions, and Voita et al. (2019) reveal that only specialized attention heads do the heavy lifting while the rest can be pruned. Accordingly, we spare several attention heads for modeling phrase patterns for SANs.

Specifically, we use two representative types of phrases that are widely-used in SMT models: *n-gram phrases* (Koehn et al., 2003) to use surface of adjacent words, and *syntactic phrases* (Liu et al., 2006) induced from syntactic trees to represent well-formed structural information. We first partition the input sentence into phrase fragments at different levels of granularity. For example, we can split a sentence into 2-grams or 3-grams. Then, we assign an attention head to attend over phrase fragments at each granularity. In this way, MG-SANs provide a lightweight strategy to explicitly model phrase structures. Furthermore, we also model the interactions among phrases to enhance structure modeling, which is one commonly-cited weakness of SANs (Tran et al., 2018; Hao et al., 2019b).

We evaluate the proposed model on two widely-used translation tasks: WMT14 English-to-German and NIST Chinese-to-English. Experimental results demonstrate that our approach consistently improves translation performance over strong TRANSFORMER baseline model (Vaswani et al., 2017) across language pairs, while

[*]Work done when interning at Tencent AI Lab.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 887–897, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

speeds marginally decrease. Analysis on multi-granularity label prediction tasks reveals that MG-SA indeed captures and stores the information of different granularity phrases as expected.

## 2 Background

**Multi-Head Self-attention** Instead of performing a single attention, Multi-Head Self-attention Networks (MH-SA), which are the defaults setting in TRANSFORMER (Vaswani et al., 2017), project the queries, keys and values into multiple subspaces and performs attention on the projected queries, keys and values in each subspace. In the standard MH-SA, it jointly attends to information from different representation subspaces at different positions. Specifically, MH-SA transform input layer $\mathbf{H} = h_1, ..., h_n \in \mathbb{R}^{n \times d}$ into $h$-th subspace with different linear projections:

$$\mathbf{Q}^h, \mathbf{K}^h, \mathbf{V}^h = \mathbf{H}\mathbf{W}_Q^h, \mathbf{H}\mathbf{W}_K^h, \mathbf{H}\mathbf{W}_V^h, \quad (1)$$

where $\{\mathbf{Q}^h, \mathbf{K}^h, \mathbf{V}^h\} \in \mathbb{R}^{n \times d_h}$ are respectively the query, key, and value representations of the $h$-th head, $\{\mathbf{W}_Q^h, \mathbf{W}_K^h, \mathbf{W}_V^h \in \mathbb{R}^{d \times d_h}\}$ denote parameter matrices associated with the $h$-th head, $d$ and $d_h$ represent the dimensionality of the model and $h$-th head subspace. Moreover, $N$ attention functions are applied to generate the output states $\{\mathbf{O}^1, ..., \mathbf{O}^N\}$ in parallel, among them:

$$\mathbf{O}^h = \text{ATT}(\mathbf{Q}^h, \mathbf{K}^h)\ \mathbf{V}^h. \quad (2)$$

Finally, the output states are concatenated to produce the final state. Here ATT denotes attention models, which can be implemented as either additive attention or dot-product attention. In this work, we use dot-product attention which is efficient and effective compared with its additive counterpart (Vaswani et al., 2017):

$$\text{ATT}(\mathbf{Q}^h, \mathbf{K}^h) = softmax(\frac{\mathbf{Q}^h \mathbf{K}^{h^T}}{\sqrt{d_h}}), \quad (3)$$

where $\sqrt{d_h}$ is the scaling factor.

**Motivation** We demonstrate our motivation from two aspects. On the one hand, the conventional MH-SA model the individual word dependencies, in such scenario the query directly attends all words in memory without considering the latent structure of the input sentence. We argue that self-attention can be further improved by taking phrase pattern into account. On the other hand, recent study (Vaswani et al., 2017) implicitly hint that attention heads are underutilized as increasing number of heads from 4 to 8 or even 16 can hardly improve the translation performance. Several attention heads can be further exploited under specific guidance to improve the performance (Strubell et al., 2018). We expect the inductive bias for multi-granularity phrase can further improve the performance of SANs and meanwhile maintain its simplicity and flexibility.

## 3 Multi-Granularity Self-Attention

We first introduce the framework of the proposed MG-SA. Then we describe the approaches of generating multi-granularity representation on a certain granularity representation. Finally, we introduce the training objective of our model with auxiliary supervision.

### 3.1 Framework

The proposed MG-SA aims at improving the capability of MH-SA by modeling both word and phrase. We introduce various phrase granularity over the conventional word-level memory to generate phrase level memory.

Specifically, we first transform the input layer $\mathbf{H}$ to a phrase level memory by function $F_h$ in certain attention head:

$$\mathbf{H}_g = F_h(\mathbf{H}), \quad (4)$$

where $\mathbf{H}_g$ is the generated phrase level memory, $h$ denotes the $h$-th head which is used to generate a certain granularity of phrase memory, and $F_h$ is a representation function with its own trainable parameters. The details for $F_h$ will be described in Section 3.2.

Then we perform attention on phrase level memory $\mathbf{H}^g$:

$$\mathbf{Q}^h, \mathbf{K}^h, \mathbf{V}^h = \mathbf{H}\mathbf{W}_Q^h, \mathbf{H}_g\mathbf{W}_K^h, \mathbf{H}_g\mathbf{W}_V^h \quad (5)$$

$$\mathbf{O}^h = \text{ATT}(\mathbf{Q}^h, \mathbf{K}^h)\ \mathbf{V}^h, \quad (6)$$

where $\mathbf{Q}^h \in \mathbb{R}^{n \times d_h}, \mathbf{K}^h \in \mathbb{R}^{p \times d_h}, \mathbf{V}^h \in \mathbb{R}^{p \times d_h}$, the $p$ means the length of the key and value vectors which is decided by the granularity of phrase.

Based on the single head self-attention, the final output of MG-SA can be expressed as follows:

$$\text{MG-SA}(\mathbf{H}) = [\mathbf{O}^1, ..., \mathbf{O}^N], \quad (7)$$

where $N$ denotes the number of heads. One head conducts either conventional word level attention or a certain granularity of phrase attention.

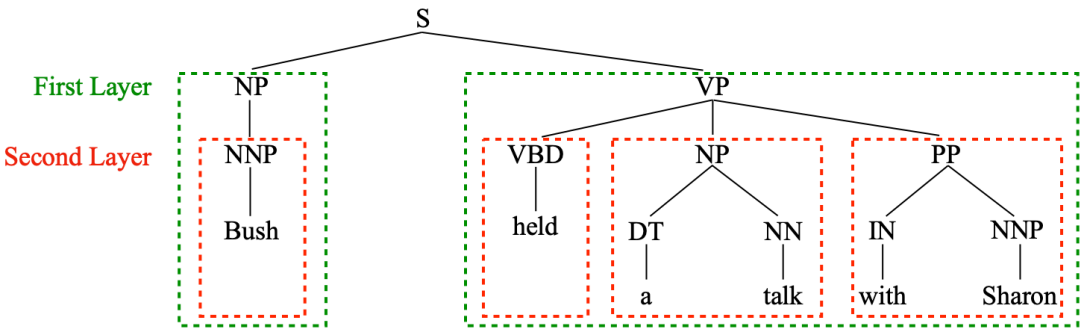


(a) Syntactic phrase partition

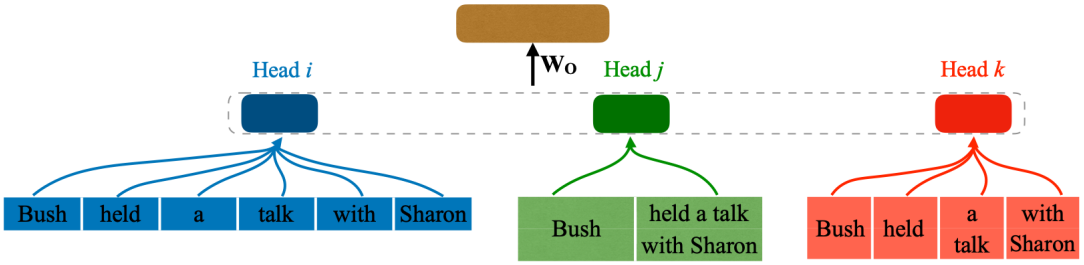


(b) *Multi-Granularity Self-Attention* on syntactic phrase partition

Figure 1: Illustration of the proposed MG-SA model for syntactic phrase partition. In this example, we partition the sentence with top two layers in the constituent parse tree and obtain the syntactic phrase partitions ("Bush", "held a talk with Sharon"), ("Bush", "held", "a talk", "with Sharon"). Under the syntactic partition, multi-head attention in MG-SA attends the phrase memory (heads $j$ and $k$) as well as the conventional word memory (head $i$). The approach of phrase memory representation is described in Section 3.2. Best viewed in colour.

### 3.2 Multi-Granularity Representation

As seen in Figure 1, multi-granularity phrases are simultaneously modeled by different heads. To obtain the multi-granularity phrase representation, we first introduce phrase partition and composition strategies. Then, we describe phrase tag supervision and phrase interaction to further enhance the structure modeling on phrase representation.

**Phrase Partition** Partially inspired by Shen et al. (2018), we split the entire sequence into N-grams without overlaps. Such N-gram phrases are expressed as structurally adjacent and continuous items in the sequence. Formally, let $\mathbf{x} = (x_1, ..., x_T)$ be a sequence, the phrases sequence of $\mathbf{x}$ can be denoted as is $P_\mathbf{x} = (p_1, ..., p_M), M = T/n$, where $p_m = (x_{n\times(m-1)}, ..., x_{n\times m}), 1 \leq m \leq M$, and $n$ denotes the length of the phrase which is a hyper-parameter. Padding is applied to the last phrase if necessary.

In addition, syntactic information has proven helpful in both SMT and NMT. We further introduce a syntactic phrase partition to represent well-formed structural information. Syntactic phrases organize words into nested constituents by using constituent parse tree. To obtain phrases in the view syntax, we break down the nodes at top K layers in the parse tree to capture top K levels of granularity for phrases, as illustrated in Figure 1 (a). Formally, one phrase in a certain layer of the parse tree can be defined as $p_m = (x^1, ..., x^l)$, $l$ is the length of the phrase which is decided by the parse tree. The phrase sequence of the given input $\mathbf{x}$ is $P_\mathbf{x} = (p_1, ..., p_M)$, $M$ is the number of phrase in the sequence.

**Composition Strategies** Given phrase sequence $P_\mathbf{x} = (p_1, ..., p_M)$ of input sequence, to capture local structure and context dependency inside each phrase and further generate phrase representation $Q_M$, we adopt phrase composition function to each phrase in the phrase sequence:

$$g_m = \text{COM}(p_m), \tag{8}$$

where COM is the composition function with shared parameters to all phrases, $g_m \in \mathbb{R}^{1\times d_h}$ is the phrase representation after composition. There general choices of composition function are Convolution Neural Networks (CNNs), Recurrent Neural Networks (RNNs) and Self-attention Net-

works (SANs). For CNNs we only apply the Max-pooling layer. For RNNs, we use the last hidden state of Long Short-term Memory Networks (LSTM) as phrase representation. For SANs, we use max pooling vector of the phrase to serve as the query for extracting inside phrase features to generate phrase representation. Then the phrase level memory of the input sequence can be denoted as $\mathbf{G_x} = (g_1, ..., g_M)$.

**Phrase Tag Supervision** Recent study shows auxiliary supervision on heads of SANs can further improve semantic role labeling performance (Strubell et al., 2018). In this work, we leverage tag information as the auxiliary supervision on syntactic phrase representation. We argue that the proposed framework provide a natural way to incorporate syntactic tag signal of phrase representation. In detail, given phrase level memory $\mathbf{G_x} = (g_1, ..., g_M)$ after phrase composition, we predict the phrase tag of each phrase representation. We extract the node of each phrase in the constituent parsing tree to generate the phrase tag sequence $\mathbf{T_x} = (t_1, ..., t_M)$. $t_i$ denotes the tag for each phrase. For example, "NP" is the tag of the phrase "a talk" in second layer of parse tree, as shown in Figure 1 (a). We use the phrase representation to compute the probability of phrase tags:

$$p_{\theta_i} = softmax(W_t g_i + b_t), i = 1, ..., M, \quad (9)$$

where $W_t$ and $b_t$ are parameters of tag generator. Formally, the phrase tag loss can be written as:

$$\mathcal{L}_{tag} = -\sum_{i=1}^{M} t_i \log p_{\theta_i}(t_i). \quad (10)$$

The loss is equivalent to maximizing the conditional probability of tag sequence $\mathbf{T_x}$ given phrase representation $\mathbf{G_x}$.

**Phrase Interaction** We introduce phrase interaction approach to better model dependencies between phrase representation. Since recurrence has proven important for capturing structure information (Tran et al., 2018; Hao et al., 2019b), we propose to introduce recurrence to interact phrases and further model latent structure among phrases. Specifically, we apply the recurrence function $\text{REC}(\cdot)$ on the output of phrase composition $\mathbf{G_x} = (g_1, ..., g_M)$ in order to model the latent structure of the phrase sequence.

$$\mathbf{H}_g = \text{REC}(\mathbf{G_x}), \quad (11)$$

where $\mathbf{H}_g$ is the final phrase level memory for the input layer $\mathbf{H}$. One general choice for $\text{REC}(\cdot)$ is Long Short-term Memory Networks (LSTM). Recently, Shen et al. (2018) introduce a new syntax-oriented inductive bias, namely ordered neurons, which enables LSTM models to perform tree-like composition without breaking its sequential form, and propose an advanced LSTM variant – *Ordered Neurons* LSTM (ON-LSTM). Hao et al. (2019a) demonstrate the effectiveness of ON-LSTM on modeling structure in NMT. Accordingly, we further use ON-LSTM for $\text{REC}(\cdot)$, and expect ON-LSTM can capture the latent structure under such syntax-oriented inductive bias between phrases.

Finally, the representation function $F_h$ in Equation 4 of the framework can be summarized by the following components: 1). Phrase partition. 2). Phrase composition. 3). Phrase interaction.

### 3.3 Training

The training loss for a single training instance $\mathbf{x} = (x_1, ..., x_T), \mathbf{y} = (y_1, ..., y_L)$ is defined as a weighted sum of the negative conditional log likelihood and the phrase tag loss. The total loss function can be written as:

$$\mathcal{L} = -\sum_{i=1}^{L} y_i log P_\omega(y_i) + \lambda \mathcal{L}_{tag}, \quad (12)$$

where $\lambda$ is the coefficient to balance two loss functions and $\mathcal{L}_{tag}$ follows Equation 10. The hyperparameter $\lambda$ is empirically set to 0.001 in this work.

## 4 Experiments

In this section, we conduct experiments and make analysis to answer the following three questions:

Q1. Does the integration of the proposed MG-SA into the state-of-the-art TRANSFORMER improve the translation quality in terms of the BLEU score?

Q2. Does the proposed MG-SA promote the generation of the target phrases?

Q3. Does MG-SA capture more phrase information at the various granularity levels?

In Section 4.1, we demonstrate that integrating the proposed MG-SA into the TRANSFORMER consistently improves the translation quality on both WMT14 English⇒German and

NIST Chinese⇒English (Q1). Further analysis reveals that our approach has stronger ability of capturing the phrase information and promoting the generation of the target phrases (Q2).

In Section 4.2, we conduct experiments on the multi-granularity label prediction tasks (Shi et al., 2016), and investigate the representations of NMT encoders trained on both translation data and the training data of the label prediction tasks. Experimental results show that the proposed MG-SA indeed captures useful phrase information at various levels of granularities in both scenarios (Q3).

### 4.1 Machine Translation

**Implementation Detail** We conduct the experiments on the WMT14 English-to-German (En⇒De) and NIST Chinese-to-English (Zh⇒En) translation tasks.

For En⇒De, the training dataset consists of 4.56M sentence pairs. We use the newstest2013 and newstest2014 as development set and test set respectively. For Zh⇒En, the training dataset consists of about 1.25M sentence pairs. We used NIST MT02 dataset as development set, and MT 03-06 datasets as test sets. Byte pair encoding (BPE) toolkit[1] (Sennrich et al., 2016) is used with 32K merge operations. We used case-sensitive NIST BLEU score (Papineni et al., 2002) as the evaluation metric, and bootstrap resampling (Koehn et al., 2003) for statistical significance test. We use the Stanford parser (Klein and Manning, 2003) to parse the sentences and obtain the relevant tags.

We test both *Base* and *Big* models, which differ at hidden size (512 vs. 1024), filter size (2048 vs. 4096) and the number of attention heads (8 vs. 16). All models are trained on eight NVIDIA Tesla P40 GPUs where each is allocated with a batch size of 4096 tokens. We implement the proposed approaches on top of TRANSFORMER (Vaswani et al., 2017) – a state-of-the-art SANs-based model on machine translation, and followed the setting in previous work (Vaswani et al., 2017) to train the models.

We incorporate the proposed model into the encoder. In each of our model variant, we maintain a quarter of heads for vanilla word level self-attention. For N-gram phrase models, we arrange the rest 3 quarters of heads for 2-gram, 3-gram and 4-gram respectively. For syntactic based models, we use the top 3 levels of granularity for phrases generated from constituent parse tree, each granularity of phrase modeled in a quarter of heads. There are many possible ways to implement the general idea of MG-SA. The aim of this paper is not to explore this whole space but simply to show that some fairly straightforward implementations work well.

[1] https://github.com/rsennrich/subword-nmt

| Phrase Modeling | # Para. | Speed | BLEU |
|---|---|---|---|
| n/a | 88.0M | 1.28 | 27.31 |
| MAX-POOLING | 88.0M | 1.27 | 27.56 |
| SANs | 90.4M | 1.26 | **27.69** |
| LSTM | 96.1M | 1.14 | 27.58 |

Table 1: Evaluation of various phrase composition strategies under N-gram phrase partition. "# Para" denotes the trainable parameter size of each model (M=million), "Speed" denotes the training speed (steps/second).

| Encoder Layers | # Para. | Speed | BLEU |
|---|---|---|---|
| $[1-6]$ | 90.4M | 1.26 | 27.69 |
| $[1-3]$ | 89.2M | 1.27 | 27.74 |
| $[1]$ | 88.4M | 1.28 | **27.83** |

Table 2: Evaluation of different layers in the encoder, which are implemented as self-attention with SANs phrase composition under N-gram partition. "1" denotes the bottom layer.

Table 1, 2 and 3 show the results on WMT14 English⇒German translation task with TRANSFORMER-BASE. These results show the evaluation on the impact of different components.

**Phrase Composition** We investigate the effect of different phrase composition strategies with N-gram phrase partition. As seen in Table 1, all proposed phrase composition methods consistently outperform TRANSFORMER-BASE baseline, validating the importance of introducing multi-granularity phrase in TRANSFORMER. Compared with other two models, SANs achieve best performance with its strong representational powers inside the phrase, while only marginally increase the parameters and decrease the speed. We use SANs phrase composition strategy as the default setting in subsequent experiments.

**Encoder Layers** Recent works (Shi et al., 2016; Peters et al., 2018) show that different layers in encoder tend to capture different syntax and semantic features. Hence, there may have different needs

| # | Model Architecture | # Para. | Speed | BLEU | Δ |
|---|---|---|---|---|---|
| 1 | TRANSFORMER-BASE | 88.0M | 1.28 | 27.31 | - |
| 2 | + N-gram Phrase | 88.4M | 1.28 | 27.83 | +0.52 |
| 3 | + Syntactic Phrase | 88.4M | 1.24 | 28.01 | +0.70 |
| 4 | + Syntactic Phrase + $\mathcal{L}_{tag}$ | 88.4M | 1.23 | 28.07 | +0.76 |
| 5 | + LSTM Interaction | 89.5M | 1.20 | 28.14 | +0.83 |
| 6 | + ON-LSTM Interaction | 89.9M | 1.19 | **28.28** | +0.97 |

Table 3: Evaluation of phrase partition, tag supervision and interaction strategies.

for modeling phrase structure in each layer. In this experiment, we investigate the question of which layers should be applied with MG-SA. We apply MG-SA on different combination of layers. As shown in Table 2, reducing the applied layers from high-level to low-level consistently increase translation quality in terms of BLEU score as well as the training speed. The results reveal that the bottom layer in encoder, which is directly taking word embedding as input, benefits more from modeling phrase structure. This phenomena verifies it is unnecessary to apply the phrase structure modeling to all layers. Accordingly, we only apply MG-SA in the bottom layer in the following experiments.

**Phrase Partition and Tag Supervision** As seen in Table 3, syntactic phrase partition (Row 3) improves the model performance over the N-gram phrase partition (Row 2), showing that the syntactic phrase benefits to translation quality. In addition, incorporating tag loss (Row 4) in training stage can further boost the translation performance. This indicates the auxiliary syntax objective is necessary, which is consistent with the results in other NLP task (Strubell et al., 2018). We use syntactic phrase partition with tag supervision as the default setting for subsequent experiments unless otherwise stated.

**Phrase Interaction** As observed in Table 3, phrase interaction (Row 5-6) consistently improves performance of translation, proving the effectiveness and necessity of enhancing phrase level dependencies on phrase representation. ON-LSTM based interaction (Row 6) outperforms its LSTM counterpart (Row 5). We attribute the improvement of ON-LSTM to the stronger ability to perform syntax-oriented dependencies on phrase representation. We apply ON-LSTM as the default setting for phrase interaction.

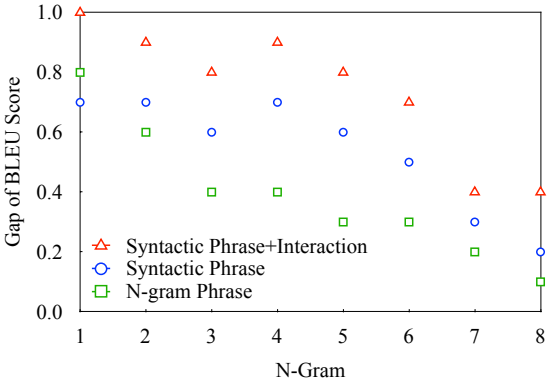


Figure 2: Performance improvement according to N-gram. Y-axis denotes the gap of BLEU score between our models and the baseline.

**Main Results** Table 4 lists the results on WMT14 En⇒De and NIST Zh⇒En translation tasks. Our baseline models, outperform the reported results on the same data (Vaswani et al., 2017; Zhang et al., 2019), which we believe make the evaluation convincing. As seen, in terms of BLEU score, the proposed MG-SA consistently improves translation performance across language pairs, which demonstrates the effectiveness and universality of the proposed approach.

**Phrasal Pattern Evaluation** As aforementioned, the proposed MG-SA aims to simultaneously model different granularities of phrases with different heads in SANs. To investigate whether the proposed MG-SA improves the generation of phrases in the output, we calculate the improvement of the proposed models over multiple N-grams, as shown in Figure 2. The results are reported on En⇒De validation set with TRANSFORMER-BASE.

Clearly, the proposed models consistently outperform the baseline model on all N-grams, indicating that the proposed MG-SA has stronger ability of capturing the phrase information and pro-

| Architecture | En⇒De | | Zh⇒En | | | | | |
|---|---|---|---|---|---|---|---|---|
| | # Para. | BLEU | # Para. | MT03 | MT04 | MT05 | MT06 | Avg |
| *Existing NMT systems* | | | | | | | | |
| Vaswani et al. (2017) | 213M | 28.4 | n/a | n/a | n/a | n/a | n/a | n/a |
| Zhang et al. (2019) | n/a | n/a | n/a | 40.45 | 42.76 | 40.09 | 39.67 | 40.74 |
| *Our NMT systems* | | | | | | | | |
| TRANSFORMER-BASE | 88.0M | 27.31 | 73.4M | 41.88 | 44.48 | 42.21 | 41.93 | 42.60 |
| +MG-SA | 89.9M | 28.28$^{\Uparrow}$ | 75.3M | 43.98$^{\Uparrow}$ | 45.60$^{\Uparrow}$ | 44.28$^{\Uparrow}$ | 44.00$^{\Uparrow}$ | 44.46 |
| TRANSFORMER-BIG | 264.1M | 28.58 | 234.8M | 45.30 | 46.49 | 45.21 | 44.87 | 45.47 |
| +MG-SA | 271.5M | 29.01$^{\uparrow}$ | 242.2M | 45.76$^{\uparrow}$ | 46.81$^{\uparrow}$ | 45.77$^{\uparrow}$ | 46.48$^{\Uparrow}$ | 46.21 |

Table 4: Comparing with the existing NMT systems on WMT14 En⇒De and NIST Zh⇒En test sets. "$\uparrow$ / $\Uparrow$": significant over the conventional self-attention counterpart ($p < 0.05/0.01$), tested by bootstrap resampling. "MG-SA" denotes "Syntactic Phrase + $\mathcal{L}_{tag}$ + ON-LSTM Interaction" in Table 3.

moting the generation of the target phrases. Concerning the variations of proposed models, two syntactic phrase models outperforms the N-gram phrase model on larger phrases (i.e. 4-8 grams). We attribute this to the fact that more syntactic information is beneficial for the translation performance. This is also consistent with the strengths of phrase-based and syntax-based SMT models.

**Visualization of Attention** In order to evaluate whether the proposed model is able to capture phrase patterns or not, we visualize the attention layers in the encoder [2]. As shown in Fig. 3, the vanilla model prefers to pay attention to the previous and next word and the end of the sentence, which is consistent with previous findings in Raganato and Tiedemann (2018). The proposed MG-SA successfully focuses on the phrases: 1) "三峡 工程", the 4th and the 5th rows in Fig. 3(b), its English translation is 'the Three Gorges Project'; 2) "首要 任务", the 7th and 8th rows in Fig. 3(b), its English translation is 'top priority'. By visualizing the attention distributions, we believe the proposed MG-SA can capture phrase patterns to improve the translation performance.

### 4.2 Multi-Granularity Phrases Evaluation

In this section, we conduct multi-granularity label prediction tasks to the proposed models in terms of whether the proposed model is effective as expected to capture different levels of granularity phrase information of sentences. We analyze the impact of multi-granularity self-attention based on two sets of experiments. The first set of experiments are probing the pre-trained NMT encoders, which aims to evaluate the linguistics knowledge embedded in the NMT encoder output in the machine translation section. Furthermore, to test the ability of the MG-SA itself, we conduct the second set of experiments, which are on the same tasks using encoder models trained from scratch.

**Tasks** Shi et al. (2016) propose 5 tasks to predict various granularity syntactic labels of from sentence to word in order to investigate whether an encoder can learn syntax information. These labels are: "Voice": active or passive, "Tense": past or non-past of main-clause verb, "TSS": top-level syntactic sequence of constituent tree, and two word-level syntactic label tasks, "SPC": the smallest phrase constituent that above each word, "POS": Part-of-Speech tags for each words. The tasks for predicting larger labels require models to capture and record larger granularity of phrase information of sentences (Shi et al., 2016). We conduct these tasks to study whether the proposed MG-SA benefits the multi-granularity phrase modeling to produce more useful and informative representation.

**Data and Models** We extracted the sentences from the Toronto Book Corpus (Zhu et al., 2015).We sample and pre-process 120k sentences for each task following Conneau et al. (2018). By instruction of Shi et al. (2016), we label these sentences for each task. The train/valid/test dataset ratios are set to 10/1/1.

For pre-trained NMT encoders, we use the pre-trained encoders of model variations in Table 3 followed by a MLP classifier, which are used to

[2]Since the attention weights of MG-SA cannot be visualized at word level, we visualize the attention weights in the subsequent layer after MH-SA and MG-SA.

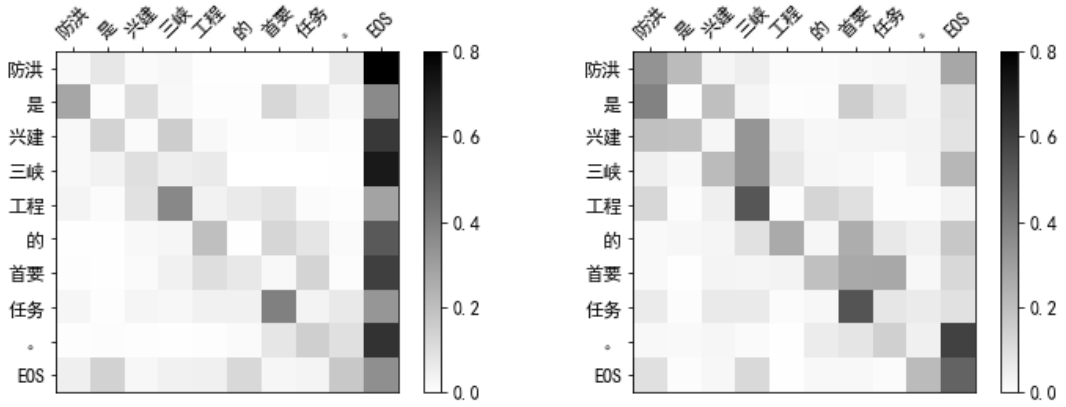


(a) Vanilla Multi-Head Self-Attention

(b) Multi-Granularity Self-Attention

Figure 3: Visualization of attention examples of the same input sentence: (a) and (b) are produced by the vanilla multi-head self-attention and the proposed MG-SA models, respectively. Each row is the attention distribution over all the source tokens. The attention layer has 16 attention heads, and the attention weights in each row are the average of all the heads.

| # | Model | Label Granularity: Large → Small | | | | | |
|---|---|---|---|---|---|---|---|
| | | Voice | Tense | TSS | SPC | POS | Avg |
| *Pre-Trained NMT Encoder* | | | | | | | |
| 1 | BASE | 73.38 | 73.73 | 72.72 | 92.81 | 93.73 | 81.27 |
| 2 | N-Gram Phrase | 73.06 | 72.83 | 72.11 | 96.42 | 96.34 | 82.15 |
| 3 | Syntactic Phrase | 73.37 | 73.62 | 75.60 | 96.72 | 96.68 | 83.19 |
| 4 | Syntactic Phrase + Interaction | 73.20 | 74.78 | 75.24 | 96.78 | 96.56 | 83.31 |
| *Train From Scratch* | | | | | | | |
| 5 | BASE | 83.46 | 85.39 | 83.44 | 96.35 | 96.12 | 88.95 |
| 6 | N-Gram Phrase | 83.55 | 85.62 | 85.21 | 96.23 | 96.17 | 89.36 |
| 7 | Syntactic Phrase | 84.70 | 87.52 | 97.42 | 96.95 | 96.24 | 92.57 |
| 8 | Syntactic Phrase + Interaction | 86.45 | 87.65 | 99.07 | 96.99 | 96.40 | 93.31 |

Table 5: Accuracies on multi-granularity label prediction tasks. "Pre-Trained NMT Encoder" denotes using the pre-trained NMT encoders of model variations in Table 3. "Train From Scratch" denotes using three encoder layers with proposed MG-SA variants, which are trained from scratch. For syntactic phrase based models, we only apply syntactic boundary of phrases and do not use any tag supervision for fair comparison.

carry out probing tasks.

For models trained from scratch, each of our model consists of 3 encoding layers followed by a MLP classifier. For each encoding layer, we employ a multi-head self-attention block and a feed-forward block as in TRANSFORMER, which have shown significant performance on several NLP tasks (Devlin et al., 2019). The difference between the compared models merely lies in the self-attention mechanism: "BASE" denotes standard MH-SA, "N-Gram Phrase" and "Syntactic Phrase" are the proposed MG-SA under N-gram phrase and syntactic phrase partition, and "Syntactic Phrase + Interaction" denotes MG-SA with phrase interaction by using ON-LSTM. We use same assignments of heads for multi-granularity phrases as machine translation task for all model variants.

**Results Analysis** Table 5 lists the prediction accuracies of five syntactic labels on test. Several observations can be made here. 1). Comparing the two set of experiments, the experimental results from models trained from scratch consistently outperform the results from NMT encoder probing on all tasks. 2). The models with syntactic information (Rows 3-4, 7-8) significantly perform better than those models without incorporating syntactic information (Rows 1-2, 5-6). 3). For NMT prob-

ing, the proposed models outperform the baseline model especially on relative small granularity of phrases information, such as 'SPC' and 'POS' tasks. 4). If trained from scratch, the proposed models achieve more improvements on predicting larger granularities of labels, such as 'TSS', 'Tense' and 'Voice' tasks, which require models to record larger phrase of sentences (Shi et al., 2016). The results show that the applicability of the proposed MG-SA is not limited to machine translation, but also on monolingual tasks.

## 5 Related Works

**Phrase Modeling for NMT** Several works have proven that the introduction of phrase modeling in NMT can obtain promising improvement on translation quality. Tree-based encoders, which explicitly take the constituent tree (Eriguchi et al., 2016) or dependency tree (Bastings et al., 2017) into consideration, are proposed to produce tree-based phrase representations. The difference of our work from these studies is that they adopt the RNN-based encoder to form the tree-based encoder while we explicitly introduce the phrase structure into the the state-of-the-art multi-layer multi-head SANs-based encoder, which we believe is more challenging.

Another thread of work is to implicitly promote the generation of phrase-aware representation, such as the integration of external phrase boundary (Wang et al., 2017; Nguyen and Joty, 2018; Li et al., 2019b), prior attention bias (Yang et al., 2018, 2019; Guo et al., 2019). Our work differs at that we explicitly model phrase patterns at different granularities, which is then attended by different attention heads.

**Multi Granularity Representation** Multi-granularity representation, which is proposed to make full use of subunit composition at different levels of granularity, has been explored in various NLP tasks, such as paraphrase identification (Yin and Schütze, 2015), Chinese word embedding learning (Yin et al., 2016), universal sentence encoding (Wu et al., 2018) and machine translation (Nguyen and Joty, 2018; Li et al., 2019b). The major difference between our work and Nguyen and Joty (2018); Li et al. (2019b) lies in that we successfully introduce syntactic information into our multi-granularity representation. Furthermore, it is not well measured how much phrase information are stored in multi-granularity representation. We conduct the multi-granularity label prediction tasks and empirically verify that the phrase information is embedded in the multi-granularity representation.

**Multi-Head Attention** Multi-head attention mechanism has shown its effectiveness in machine translation (Vaswani et al., 2017) and generative dialog (Tao et al., 2018) systems. Recent studies shows that the modeling ability of multi-head attention has not been completely developed. Several specific guidance cues of different heads without breaking the vanilla multi-head attention mechanism can further boost the performance, e.g., disagreement regularization (Li et al., 2018; Tao et al., 2018), information aggregation (Li et al., 2019a), and functional specialization (Fan et al., 2019) on attention heads, the combination of multi-head attention with multi-task learning (Strubell et al., 2018). Our work demonstrates that multi-head attention also benefits from the integration of the phrase information.

## 6 Conclusion

In this paper, we propose multi-granularity self-attention model, a novel attention mechanism to simultaneously attend different granularity phrase. We study effective phrase representation for N-gram phrase and syntactic phrase, and find that a syntactic phrase based mechanism obtains the best result due to effectively incorporating rich syntactic information. To evaluate the effectiveness of the proposed model, we conduct experiments on widely-used WMT14 En⇒De and NIST Zh⇒En datasets. Experimental results on two language pairs show that the proposed model achieve significant improvements over the baseline TRANSFORMER. Targeted multi-granularity phrases evaluation shows that our model indeed capture useful phrase information.

As our approach is not limited to specific tasks, it is interesting to validate the proposed model in other tasks, such as reading comprehension, language inference, and sentence classification.

## Acknowledgments

J.Z. was supported by the National Institute of General Medical Sciences of the National Institute of Health under award number R01GM126558. We thank the anonymous reviewers for their insightful comments.

## References


Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Simaan. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *EMNLP*.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *ACL*.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $\&!\#*$ vector: Probing sentence embeddings for linguistic properties. In *ACL*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-sequence attentional neural machine translation. In *ACL*.

Angela Fan, Mike Lewis, and Yann Dauphin. 2019. Strategies for structuring story generation. In *ACL*.

Maosheng Guo, Yu Zhang, and Ting Liu. 2019. Gaussian transformer: a lightweight approach for natural language inference. In *AAAI*.

Jie Hao, Xing Wang, Shuming Shi, Jinfeng Zhang, and Zhaopeng Tu. 2019a. Towards better modeling hierarchical structure for self-attention with ordered neurons. In *EMNLP*.

Jie Hao, Xing Wang, Baosong Yang, Longyue Wang, Jinfeng Zhang, and Zhaopeng Tu. 2019b. Modeling recurrence for transformer. In *NAACL*.

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *ACL*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *NAACL*.

Jian Li, Zhaopeng Tu, Baosong Yang, Michael R. Lyu, and Tong Zhang. 2018. Multi-Head Attention with Disagreement Regularization. In *EMNLP*.

Jian Li, Baosong Yang, Zi-Yi Dou, Xing Wang, Michael R Lyu, and Zhaopeng Tu. 2019a. Information aggregation for multi-head attention with routing-by-agreement. In *NAACL*.

Yang Li, Lukasz Kaiser, Samy Bengio, and Si Si. 2019b. Area attention. In *ICML*.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *ACL*.

Phi Xuan Nguyen and Shafiq Joty. 2018. Phrase-based attentions. *arXiv preprint arXiv:1810.03444*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*.

Alessandro Raganato and Jörg Tiedemann. 2018. An analysis of encoder representations in transformer-based machine translation. In *EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL*.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, and Chengqi Zhang. 2018. Bi-directional block self-attention for fast and memory-efficient sequence modeling. In *ICLR*.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural mt learn source syntax. In *EMNLP*.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-Informed Self-Attention for Semantic Role Labeling. In *EMNLP*.

Chongyang Tao, Shen Gao, Mingyue Shang, Wei Wu, Dongyan Zhao, and Rui Yan. 2018. Get the point of my utterance! learning towards effective responses with multi-head attention mechanism. In *IJCAI*.

Ke Tran, Arianna Bisazza, and Christof Monz. 2018. The importance of being recurrent for modeling hierarchical structure. In *EMNLP*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. In *NIPS*.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *ACL*.

Xing Wang, Zhaopeng Tu, Deyi Xiong, and Min Zhang. 2017. Translating phrases in neural machine translation. In *EMNLP*.

Wei Wu, Houfeng Wang, Tianyu Liu, and Shuming Ma. 2018. Phrase-level self-attention networks for universal sentence encoding. In *EMNLP*.

Baosong Yang, Zhaopeng Tu, Derek F. Wong, Fandong Meng, Lidia S. Chao, and Tong Zhang. 2018. Modeling localness for self-attention networks. In *EMNLP*.

Baosong Yang, Longyue Wang, Derek Wong, Lidia S Chao, and Zhaopeng Tu. 2019. Convolutional self-attention networks. In *NAACL*.

Rongchao Yin, Quan Wang, Peng Li, Rui Li, and Bin Wang. 2016. Multi-granularity chinese word embedding. In *EMNLP*.

Wenpeng Yin and Hinrich Schütze. 2015. Convolutional neural network for paraphrase identification. In *NAACL*.

Meishan Zhang, Zhenghua Li, Guohong Fu, and Min Zhang. 2019. Syntax-enhanced neural machine translation with syntax-aware word representations. In *NAACL*.

Yang Zhao, Yining Wang, Jiajun Zhang, and Chengqing Zong. 2018. Phrase table as recommendation memory for neural machine translation. In *IJCAI*.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *ICCV*.

# Improving Deep Transformer with Depth-Scaled Initialization and Merged Attention

**Biao Zhang**[1] **Ivan Titov**[1,2] **Rico Sennrich**[3,1]

[1]School of Informatics, University of Edinburgh
[2]ILLC, University of Amsterdam
[3]Institute of Computational Linguistics, University of Zurich
B.Zhang@ed.ac.uk, ititov@inf.ed.ac.uk, sennrich@cl.uzh.ch

## Abstract

The general trend in NLP is towards increasing model capacity and performance via deeper neural networks. However, simply stacking more layers of the popular Transformer architecture for machine translation results in poor convergence and high computational overhead. Our empirical analysis suggests that convergence is poor due to *gradient vanishing* caused by the interaction between residual connections and layer normalization. We propose depth-scaled initialization (DS-Init), which decreases parameter variance at the initialization stage, and reduces output variance of residual connections so as to ease gradient back-propagation through normalization layers. To address computational cost, we propose a merged attention sublayer (MAtt) which combines a simplified average-based self-attention sublayer and the encoder-decoder attention sublayer on the decoder side. Results on WMT and IWSLT translation tasks with five translation directions show that deep Transformers with DS-Init and MAtt can substantially outperform their base counterpart in terms of BLEU (+1.1 BLEU on average for 12-layer models), while matching the decoding speed of the baseline model thanks to the efficiency improvements of MAtt.[1]

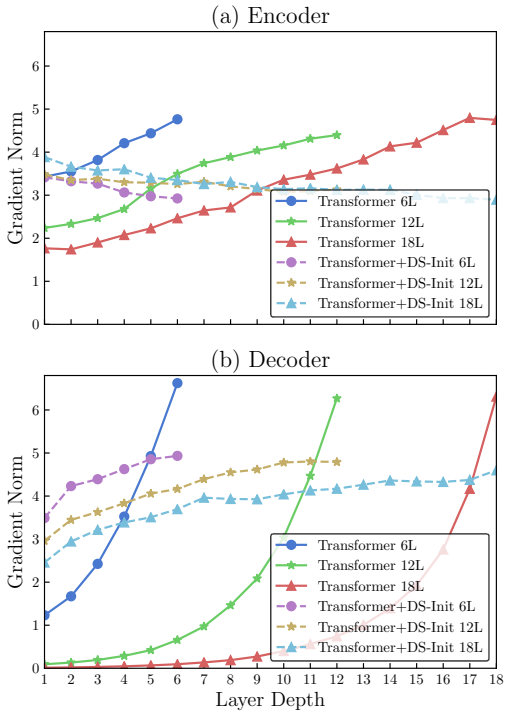


Figure 1: Gradient norm (y-axis) of each encoder layer (top) and decoder layer (bottom) in Transformer with respect to layer depth (x-axis). Gradients are estimated with ∼3k target tokens at the beginning of training. "*DS-Init*": the proposed depth-scaled initialization. "*6L*": 6 layers. Solid lines indicate the vanilla Transformer, and dashed lines denote our proposed method. During back-propagation, gradients in Transformer gradually vanish from high layers to low layers.

## 1 Introduction

The capability of deep neural models of handling complex dependencies has benefited various artificial intelligence tasks, such as image recognition where test error was reduced by scaling VGG nets (Simonyan and Zisserman, 2015) up to hundreds of convolutional layers (He et al., 2015). In NLP, deep self-attention networks have enabled large-scale pretrained language models such as BERT (Devlin et al., 2019) and GPT (Radford et al., 2018) to boost state-of-the-art (SOTA) performance on downstream applications. By contrast, though neural machine translation (NMT) gained encouraging improvement when shifting from a shallow architecture (Bahdanau et al., 2015) to deeper ones (Zhou et al., 2016; Wu et al., 2016; Zhang et al., 2018; Chen et al., 2018), the Transformer (Vaswani et al., 2017), a currently SOTA architecture, achieves best results with merely 6 encoder and decoder layers, and no gains were reported by Vaswani et al. (2017) from

[1]Source code for reproduction is available at https://github.com/bzhangGo/zero



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 898–909, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

further increasing its depth on standard datasets.

We start by analysing why the Transformer does not scale well to larger model depth. We find that the architecture suffers from gradient vanishing as shown in Figure 1, leading to poor convergence. An in-depth analysis reveals that the Transformer is not norm-preserving due to the involvement of and the interaction between residual connection (RC) (He et al., 2015) and layer normalization (LN) (Ba et al., 2016).

To address this issue, we propose depth-scaled initialization (DS-Init) to improve norm preservation. We ascribe the gradient vanishing to the large output variance of RC and resort to strategies that could reduce it without model structure adjustment. Concretely, DS-Init scales down the variance of parameters in the $l$-th layer with a discount factor of $\frac{1}{\sqrt{l}}$ at the initialization stage alone, where $l$ denotes the layer depth starting from 1. The intuition is that parameters with small variance in upper layers would narrow the output variance of corresponding RCs, improving norm preservation as shown by the dashed lines in Figure 1. In this way, DS-Init enables the convergence of deep Transformer models to satisfactory local optima.

Another bottleneck for deep Transformers is the increase in computational cost for both training and decoding. To combat this, we propose a merged attention network (MAtt). MAtt simplifies the decoder by replacing the separate self-attention and encoder-decoder attention sublayers with a new sublayer that combines an efficient variant of average-based self-attention (AAN) (Zhang et al., 2018) and the encoder-decoder attention. We simplify the AAN by reducing the number of linear transformations, reducing both the number of model parameters and computational cost. The merged sublayer benefits from parallel calculation of (average-based) self-attention and encoder-decoder attention, and reduces the depth of each decoder block.

We conduct extensive experiments on WMT and IWSLT translation tasks, covering five translation tasks with varying data conditions and translation directions. Our results show that deep Transformers with DS-Init and MAtt can substantially outperform their base counterpart in terms of BLEU (+1.1 BLEU on average for 12-layer models), while matching the decoding speed of the baseline model thanks to the efficiency improvements of MAtt.

Our contributions are summarized as follows:

- We analyze the vanishing gradient issue in the Transformer, and identify the interaction of residual connections and layer normalization as its source.
- To address this problem, we introduce depth-scaled initialization (DS-Init).
- To reduce the computational cost of training deep Transformers, we introduce a merged attention model (MAtt). MAtt combines a simplified average-attention model and the encoder-decoder attention into a single sublayer, allowing for parallel computation.
- We conduct extensive experiments and verify that deep Transformers with DS-Init and MAtt improve translation quality while preserving decoding efficiency.

## 2 Related Work

Our work aims at improving translation quality by increasing model depth. Compared with the single-layer NMT system (Bahdanau et al., 2015), deep NMT models are typically more capable of handling complex language variations and translation relationships via stacking multiple encoder and decoder layers (Zhou et al., 2016; Wu et al., 2016; Britz et al., 2017; Chen et al., 2018), and/or multiple attention layers (Zhang et al., 2018). One common problem for the training of deep neural models are vanishing or exploding gradients. Existing methods mainly focus on developing novel network architectures so as to stabilize gradient back-propagation, such as the fast-forward connection (Zhou et al., 2016), the linear associative unit (Wang et al., 2017), or gated recurrent network variants (Hochreiter and Schmidhuber, 1997; Gers and Schmidhuber, 2001; Cho et al., 2014; Di Gangi and Federico, 2018). In contrast to the above recurrent network based NMT models, recent work focuses on feed-forward alternatives with more smooth gradient flow, such as convolutional networks (Gehring et al., 2017) and self-attention networks (Vaswani et al., 2017).

The Transformer represents the current SOTA in NMT. It heavily relies on the combination of residual connections (He et al., 2015) and layer normalization (Ba et al., 2016) for convergence. Nevertheless, simply extending this model with more layers results in gradient vanishing due to the interaction of RC and LN (see Section 4). Recent work has proposed methods to train deeper

Transformer models, including a rescheduling of RC and LN (Vaswani et al., 2018), the transparent attention model (Bapna et al., 2018) and the stochastic residual connection (Pham et al., 2019). In contrast to these work, we identify the large output variance of RC as the source of gradient vanishing, and employ scaled initialization to mitigate it without any structure adjustment. The effect of careful initialization on boosting convergence was also investigated and verified in previous work (Zhang et al., 2019; Child et al., 2019; Devlin et al., 2019; Radford et al., 2018).

The merged attention network falls into the category of simplifying the Transformer so as to shorten training and/or decoding time. Methods to improve the Transformer's running efficiency range from algorithmic improvements (Junczys-Dowmunt et al., 2018), non-autoregressive translation (Gu et al., 2018; Ghazvininejad et al., 2019) to decoding dependency reduction such as average attention network (Zhang et al., 2018) and blockwise parallel decoding (Stern et al., 2018). Our MAtt builds upon the AAN model, further simplifying the model by reducing the number of linear transformations, and combining it with the encoder-decoder attention. In work concurrent to ours, So et al. (2019) propose the evolved Transformer which, based on automatic architecture search, also discovered a parallel structure of self-attention and encoder-decoder attention.

## 3 Background: Transformer

Given a source sequence $\mathbf{X} = \{x_1, x_2, \ldots, x_n\} \in \mathbb{R}^{n \times d}$, the Transformer predicts a target sequence $\mathbf{Y} = \{y_1, y_2, \ldots, y_m\}$ under the encoder-decoder framework. Both the encoder and the decoder in the Transformer are composed of attention networks, functioning as follows:

$$\begin{aligned} \textsc{Att}(\mathbf{Z}_x, \mathbf{Z}_y) &= \left[\text{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}})\mathbf{V}\right]\mathbf{W}_o \\ \mathbf{Q}, \mathbf{K}, \mathbf{V} &= \mathbf{Z}_x\mathbf{W}_q, \mathbf{Z}_y\mathbf{W}_k, \mathbf{Z}_y\mathbf{W}_v, \end{aligned} \quad (1)$$

where $\mathbf{Z}_x \in \mathbb{R}^{I \times d}$ and $\mathbf{Z}_y \in \mathbb{R}^{J \times d}$ are input sequence representations of length $I$ and $J$ respectively, $\mathbf{W}_* \in \mathbb{R}^{d \times d}$ denote weight parameters. The attention network can be further enhanced with multi-head attention (Vaswani et al., 2017).

Formally, the encoder stacks $L$ identical layers, each including a self-attention sublayer (Eq. 2) and a point-wise feed-forward sublayer (Eq. 3):

$$\bar{\mathbf{H}}^l = \textsc{Ln}\left(\textsc{Rc}\left(\mathbf{H}^{l-1}, \textsc{Att}(\mathbf{H}^{l-1}, \mathbf{H}^{l-1})\right)\right) \quad (2)$$

$$\mathbf{H}^l = \textsc{Ln}\left(\textsc{Rc}\left(\bar{\mathbf{H}}^l, \textsc{Ffn}(\bar{\mathbf{H}}^l)\right)\right). \quad (3)$$

$\mathbf{H}^l \in \mathbb{R}^{n \times d}$ denotes the sequence representation of the $l$-th encoder layer. Input to the first layer $\mathbf{H}^0$ is the element-wise addition of the source word embedding $\mathbf{X}$ and the corresponding positional encoding. $\textsc{Ffn}(\cdot)$ is a two-layer feed-forward network with a large intermediate representation and ReLU activation function. Each encoder sublayer is wrapped with a residual connection (Eq. 4), followed by layer normalization (Eq. 5):

$$\textsc{Rc}(\mathbf{z}, \mathbf{z}') = \mathbf{z} + \mathbf{z}', \quad (4)$$

$$\textsc{Ln}(\mathbf{z}) = \frac{\mathbf{z} - \mu}{\sigma} \odot \mathbf{g} + \mathbf{b}, \quad (5)$$

where $\mathbf{z}$ and $\mathbf{z}'$ are input vectors, and $\odot$ indicates element-wise multiplication. $\mu$ and $\sigma$ denote the mean and standard deviation statistics of vector $\mathbf{z}$. The normalized $\mathbf{z}$ is then re-scaled and re-centered by trainable parameters $\mathbf{g}$ and $\mathbf{b}$ individually.

The decoder also consists of $L$ identical layers, each of them extends the encoder sublayers with an encoder-decoder attention sublayer (Eq. 7) to capture translation alignment from target words to relevant source words:

$$\tilde{\mathbf{S}}^l = \textsc{Ln}\left(\textsc{Rc}\left(\mathbf{S}^{l-1}, \textsc{Att}(\mathbf{S}^{l-1}, \mathbf{S}^{l-1})\right)\right) \quad (6)$$

$$\bar{\mathbf{S}}^l = \textsc{Ln}\left(\textsc{Rc}\left(\tilde{\mathbf{S}}^l, \textsc{Att}(\tilde{\mathbf{S}}^l, \mathbf{H}^L)\right)\right) \quad (7)$$

$$\mathbf{S}^l = \textsc{Ln}\left(\textsc{Rc}\left(\bar{\mathbf{S}}^l, \textsc{Ffn}(\bar{\mathbf{S}}^l)\right)\right). \quad (8)$$

$\mathbf{S}^l \in \mathbb{R}^{m \times d}$ is the sequence representation of the $l$-th decoder layer. Input $\mathbf{S}^0$ is defined similar to $\mathbf{H}^0$. To ensure auto-regressive decoding, the attention weights in Eq. 6 are masked to prevent attention to future target tokens.

The Transformer's parameters are typically initialized by sampling from a uniform distribution:

$$\mathbf{W} \in \mathbb{R}^{d_i \times d_o} \sim \mathcal{U}(-\gamma, \gamma), \gamma = \sqrt{\frac{6}{d_i + d_o}}, \quad (9)$$

where $d_i$ and $d_o$ indicate input and output dimension separately. This initialization has the advantage of maintaining activation variances and back-propagated gradients variance and can help train deep neural networks (Glorot and Bengio, 2010).

## 4 Vanishing Gradient Analysis

One natural way to deepen Transformer is simply enlarging the layer number $L$. Unfortunately, Figure 1 shows that this would give rise to gradient vanishing on both the encoder and the decoder at the lower layers, and that the case on the decoder side is worse. We identified a structural problem in the Transformer architecture that gives rise to this issue, namely the interaction of RC and LN, which we will here discuss in more detail.

Given an input vector $\mathbf{z} \in \mathbb{R}^d$, let us consider the general structure of RC followed by LN:

$$\mathbf{r} = \textsc{Rc}\left(\mathbf{z}, f(\mathbf{z})\right), \tag{10}$$

$$\mathbf{o} = \textsc{Ln}\left(\mathbf{r}\right), \tag{11}$$

where $\mathbf{r}, \mathbf{o} \in \mathbb{R}^d$ are intermediate outputs. $f(\cdot)$ represents any neural network, such as recurrent, convolutional or attention network, etc. Suppose during back-propagation, the error signal at the output of LN is $\delta_o$. Contributions of RC and LN to the error signal are as follows:

$$\delta_r = \frac{\partial \mathbf{o}}{\partial \mathbf{r}}\delta_o = \text{diag}(\frac{\mathbf{g}}{\sigma_r})(\mathbf{I} - \frac{1 - \bar{\mathbf{r}}\bar{\mathbf{r}}^T}{d})\delta_o \tag{12}$$

$$\delta_z = \frac{\partial \mathbf{r}}{\partial \mathbf{z}}\delta_r = (1 + \frac{\partial f}{\partial \mathbf{z}})\delta_r, \tag{13}$$

where $\bar{\mathbf{r}}$ denotes the normalized input. $\mathbf{I}$ is the identity matrix and $\text{diag}(\cdot)$ establishes a diagonal matrix from its input. The resulting $\delta_r$ and $\delta_z$ are error signals arrived at output $\mathbf{r}$ and $\mathbf{z}$ respectively.

We define the change of error signal as follows:

$$\beta = \beta_{\textsc{Ln}} \cdot \beta_{\textsc{Rc}} = \frac{\|\delta_z\|_2}{\|\delta_r\|_2} \cdot \frac{\|\delta_r\|_2}{\|\delta_o\|_2}, \tag{14}$$

where $\beta$ (or model ratio), $\beta_{\textsc{Ln}}$ (or LN ratio) and $\beta_{\textsc{Rc}}$ (or RC ratio) measure the gradient norm ratio[2] of the whole residual block, the layer normalization and the residual connection respectively. Informally, a neural model should preserve the gradient norm between layers ($\beta \approx 1$) so as to allow training of very deep models (see Zaeemzadeh et al., 2018).

We resort to empirical evidence to analyze these ratios. Results in Table 1 show that LN weakens error signal ($\beta_{\textsc{Ln}} < 1$) but RC strengthens it ($\beta_{\textsc{Rc}} > 1$). One explanation about LN's decay effect is the large output variance of RC ($\text{Var}(\mathbf{r}) >$ 1) which negatively affects $\delta_r$ as shown in Eq. 12. By contrast, the short-cut in RC ensures that the error signal at higher layer $\delta_r$ can always be safely carried on to lower layer no matter how complex $\frac{\partial f}{\partial \mathbf{z}}$ would be as in Eq. 13, increasing the ratio.

[2]Model gradients depend on both error signal and layer activation. Reduced/enhanced error signal does not necessarily result in gradient vanishing/explosion, but strongly contributes to it.

| Method | Module | | Self | Cross | FFN |
|---|---|---|---|---|---|
| Base | Enc | $\beta_{\textsc{Ln}}$ | 0.86 | - | 0.84 |
| | | $\beta_{\textsc{Rc}}$ | 1.22 | - | 1.10 |
| | | $\beta$ | 1.05 | - | 0.93 |
| | | $\text{Var}(\mathbf{r})$ | 1.38 | - | 1.40 |
| | Dec | $\beta_{\textsc{Ln}}$ | 0.82 | 0.74 | 0.84 |
| | | $\beta_{\textsc{Rc}}$ | 1.21 | 1.00 | 1.11 |
| | | $\beta$ | 0.98 | 0.74 | 0.93 |
| | | $\text{Var}(\mathbf{r})$ | 1.48 | 1.84 | 1.39 |
| Ours | Enc | $\beta_{\textsc{Ln}}$ | 0.96 | - | 0.95 |
| | | $\beta_{\textsc{Rc}}$ | 1.04 | - | 1.02 |
| | | $\beta$ | 1.02 | - | 0.98 |
| | | $\text{Var}(\mathbf{r})$ | 1.10 | - | 1.10 |
| | Dec | $\beta_{\textsc{Ln}}$ | 0.95 | 0.94 | 0.94 |
| | | $\beta_{\textsc{Rc}}$ | 1.05 | 1.00 | 1.02 |
| | | $\beta$ | 1.10 | 0.95 | 0.98 |
| | | $\text{Var}(\mathbf{r})$ | 1.13 | 1.15 | 1.11 |

Table 1: Empirical measure of output variance $\text{Var}(\mathbf{r})$ of RC and error signal change ratio $\beta_{\textsc{Ln}}$, $\beta_{\textsc{Rc}}$ and $\beta$ (Eq. 14) averaged over 12 layers. These values are estimated with $\sim$3k target tokens at the beginning of training using 12-layer Transformer. "*Base*": the baseline Transformer. "*Ours*": the Transformer with DS-Init. *Enc* and *Dec* stand for encoder and decoder respectively. *Self*, *Cross* and *FFN* indicate the self-attention, encoder-decoder attention and the feed-forward sublayer respectively.

## 5 Depth-Scaled Initialization

Results on the model ratio show that self-attention sublayer has a (near) increasing effect ($\beta > 1$) that intensifies error signal, while feed-forward sublayer manifests a decreasing effect ($\beta < 1$). In particular, though the encoder-decoder attention sublayer and the self-attention sublayer share the same attention formulation, the model ratio of the former is smaller. As shown in Eq. 7 and 1, part of the reason is that encoder-decoder attention can only back-propagate gradients to lower layers through the query representation $\mathbf{Q}$, bypassing gradients at the key $\mathbf{K}$ and the value $\mathbf{V}$ to the encoder side. This negative effect explains why the decoder suffers from more severe gradient vanishing than the encoder in Figure 1.

The gradient norm is preserved better through the self-attention layer than the encoder-decoder attention, which offers insights on the successful training of the deep Transformer in BERT (Devlin et al., 2019) and GPT (Radford et al., 2018), where encoder-decoder attention is not involved. However, results in Table 1 also suggests that the self-attention sublayer in the encoder is not strong

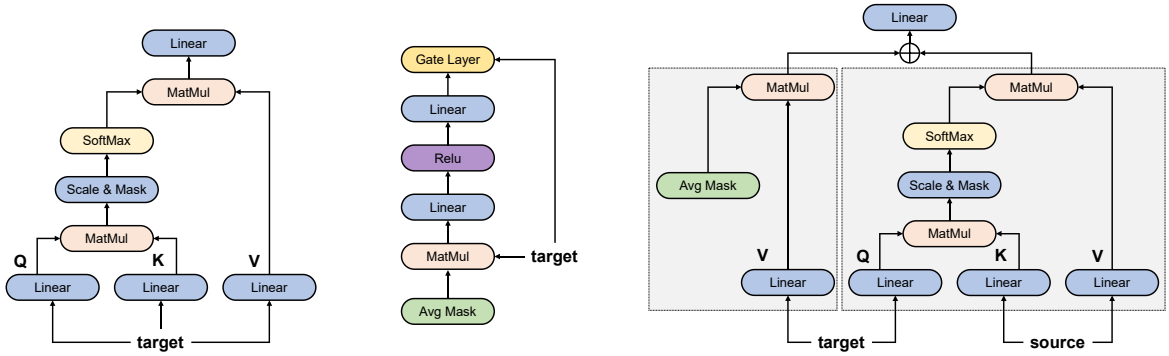


(a) Self-Attention (b) AAN (c) Merged attention with simplified AAN

Figure 2: An overview of self-attention, AAN and the proposed merged attention with simplified AAN.

enough to counteract the gradient loss in the feed-forward sublayer. That is why BERT and GPT adopt a much smaller standard deviation (0.02) for initialization, in a similar spirit to our solution.

We attribute the gradient vanishing issue to the large output variance of RC (Eq. 12). Considering that activation variance is positively correlated with parameter variance (Glorot and Bengio, 2010), we propose DS-Init and change the original initialization method in Eq. 9 as follows:

$$\mathbf{W} \in \mathbb{R}^{d_i \times d_o} \sim \mathcal{U}\left(-\gamma\frac{\alpha}{\sqrt{l}}, \gamma\frac{\alpha}{\sqrt{l}}\right), \quad (15)$$

where $\alpha$ is a hyperparameter in the range of $[0, 1]$ and $l$ denotes layer depth. Hyperparameter $\alpha$ improves the flexibility of our method. Compared with existing approaches (Vaswani et al., 2018; Bapna et al., 2018), our solution does not require modifications in the model architecture and hence is easy to implement.

According to the property of uniform distribution, the variance of model parameters decreases from $\frac{\gamma^2}{3}$ to $\frac{\gamma^2\alpha^2}{3l}$ after applying DS-Init. By doing so, a higher layer would have smaller output variance of RC so that more gradients can flow back. Results in Table 1 suggest that DS-Init narrows both the variance and different ratios to be $\sim$1, ensuring the stability of gradient back-propagation. Evidence in Figure 1 also shows that DS-Init helps keep the gradient norm and slightly increases it on the encoder side. This is because DS-Init endows lower layers with parameters of larger variance and activations of larger norm. When error signals at different layers are of similar scale, the gradient norm at lower layers would be larger. Nevertheless, this increase does not hurt model training based on our empirical observation.

DS-Init is partially inspired by the *Fixup* initialization (Zhang et al., 2019). Both of them try to reduce the output variance of RC. The difference is that Fixup focuses on overcoming gradient explosion cased by consecutive RCs and seeks to enable training without LN but at the cost of carefully handling parameter initialization of each matrix transformation, including manipulating initialization of different bias and scale terms. Instead, DS-Init aims at solving gradient vanishing in deep Transformer caused by the structure of RC followed by LN. We still employ LN to standardize layer activation and improve model convergence. The inclusion of LN ensures the stability and simplicity of DS-Init.

## 6 Merged Attention Model

With large model depth, deep Transformer unavoidably introduces high computational overhead. This brings about significantly longer training and decoding time. To remedy this issue, we propose a merged attention model for decoder that integrates a simplified average-based self-attention sublayer into the encoder-decoder attention sublayer. Figure 2 highlights the difference.

The AAN model (Figure 2(b)), as an alternative to the self-attention model (Figure 2(a)), accelerates Transformer decoding by allowing decoding in linear time, avoiding the $\mathcal{O}(n^2)$ complexity of the self-attention mechanism (Zhang et al., 2018). Unfortunately, the gating sublayer and the feed-forward sublayer inside AAN reduce the empirical performance improvement. We propose a simplified AAN by removing all matrix computation except for two linear projections:

$$\mathrm{SAAN}(\mathbf{S}^{l-1}) = \left[\mathbf{M}_a(\mathbf{S}^{l-1}\mathbf{W}_v)\right]\mathbf{W}_o, \quad (16)$$

| Dataset | #Src | #Tgt | #Sent | #BPE |
|---|---|---|---|---|
| WMT14 En-De | 116M | 110M | 4.5M | 32K |
| WMT14 En-Fr | 1045M | 1189M | 36M | 32K |
| WMT18 En-Fi | 73M | 54M | 3.3M | 32K |
| WMT18 Zh-En | 510M | 576M | 25M | 32K |
| IWSLT14 De-En | 3.0M | 3.2M | 159K | 30K |

Table 2: Statistics for different training datasets. *#Src* and *#Tgt* denote the number of source and target tokens respectively. *#Sent*: the number of bilingual sentences. *#BPE*: the number of merge operations in BPE. *M*: million, *K*: thousand.

where $\mathbf{M}_a$ denotes the average mask matrix for parallel computation (Zhang et al., 2018). This new model is then combined with the encoder-decoder attention as shown in Figure 2(c):

$$\begin{aligned} \text{MATT}(\mathbf{S}^{l-1}) &= \text{SAAN}(\mathbf{S}^{l-1}) + \text{ATT}(\mathbf{S}^{l-1}, \mathbf{H}^{L}) \\ \bar{\mathbf{S}}^{l} &= \text{LN}\left(\text{RC}\left(\mathbf{S}^{l-1}, \text{MATT}(\mathbf{S}^{l-1})\right)\right). \end{aligned} \quad (17)$$

The mapping $\mathbf{W}_o$ is shared for SAAN and ATT. After combination, MAtt allows for the parallelization of AAN and encoder-decoder attention.

## 7 Experiments

### 7.1 Datasets and Evaluation

We take WMT14 English-German translation (En-De) (Bojar et al., 2014) as our benchmark for model analysis, and examine the generalization of our approach on four other tasks: WMT14 English-French (En-Fr), IWSLT14 German-English (De-En) (Cettolo et al., 2014), WMT18 English-Finnish (En-Fi) and WMT18 Chinese-English (Zh-En) (Bojar et al., 2018). Byte pair encoding algorithm (BPE) (Sennrich et al., 2016) is used in preprocessing to handle low frequency words. Statistics of different datasets are listed in Table 2.

Except for IWSLT14 De-En task, we collect subword units independently on the source and target side of training data. We directly use the preprocessed training data from the WMT18 website[3] for En-Fi and Zh-En tasks, and use newstest2017 as our development set, newstest2018 as our test set. Our training data for WMT14 En-De and WMT14 En-Fr is identical to previous setups (Vaswani et al., 2017; Wu et al., 2019). We use newstest2013 as development set for WMT14 En-De and newstest2012+2013 for WMT14 En-Fr. Apart from newstest2014 test set[4], we also evaluate our model on all WMT14-18 test sets for WMT14 En-De translation. The settings for IWSLT14 De-En are as in Ranzato et al. (2016), with 7584 sentence pairs for development, and the concatenated dev sets for IWSLT 2014 as test set (tst2010, tst2011, tst2012, dev2010, dev2012).

We report tokenized case-sensitive BLEU (Papineni et al., 2002) for WMT14 En-De and WMT14 En-Fr, and provide detokenized case-sensitive BLEU for WMT14 En-De, WMT18 En-Fi and Zh-En with *sacreBLEU* (Post, 2018)[5]. We also report chrF score for En-Fi translation which was found correlated better with human evaluation (Bojar et al., 2018). Following previous work (Wu et al., 2019), we evaluate IWSLT14 De-En with tokenized case-insensitive BLEU.

### 7.2 Model Settings

We experiment with both *base* (layer size 512/2048, 8 heads) and *big* (layer size 1024/4096, 16 heads) settings as in Vaswani et al. (2017). Except for the vanilla Transformer, we also compare with the structure that is currently default in tensor2tensor (T2T), which puts layer normalization before residual blocks (Vaswani et al., 2018). We use an in-house toolkit for all experiments.

Dropout is applied to the residual connection ($dp_r$) and attention weights ($dp_a$). We share the target embedding matrix with the softmax projection matrix but not with the source embedding matrix. We train all models using Adam optimizer (0.9/0.98 for base, 0.9/0.998 for big) with adaptive learning rate schedule (warm-up step 4K for base, 16K for big) as in (Vaswani et al., 2017) and label smoothing of 0.1. We set $\alpha$ in DS-Init to 1.0. Sentence pairs containing around 25K$\sim$50K ($bs$) target tokens are grouped into one batch. We use relatively larger batch size and dropout rate for deeper and bigger models for better convergence. We perform evaluation by averaging last 5 checkpoints. Besides, we apply mixed-precision training to all big models. Unless otherwise stated, we train base and big model with 300K maximum steps, and decode sentences using beam search with a beam size of 4 and length penalty of 0.6. Decoding is implemented with cache to save redundant computations. Other settings for specific translation tasks are explained in the individual subsections.

[3] http://www.statmt.org/wmt18/translation-task.html

[4] We use the filtered test set consisting of 2737 sentence pairs. The difference of translation quality on filtered and full test sets is marginal.

[5] Signature BLEU+c.mixed+#.1+s.exp+tok.13a+v.1.2.20

| ID | Model | #Param | Test14 | △Dec | △Train |
|---|---|---|---|---|---|
| 1 | Base 6 layers $dp_a = dp_r = 0.1, bs = 25K$ | 72.3M | 27.59 (26.9) | 62.26/1.00× | 0.105/1.00× |
| 2 | 1 + T2T | 72.3M | 27.20 (26.5) | 68.04/0.92× | 0.105/1.00× |
| 3 | 1 + DS-Init | 72.3M | 27.50 (26.8) | ⋆/1.00× | ⋆/1.00× |
| 4 | 1 + MAtt | 66.0M | 27.49 (26.8) | 40.51/1.54× | 0.094/1.12× |
| 5 | 1 + MAtt + DS-Init | 66.0M | 27.35 (26.8) | 40.84/1.52× | 0.094/1.12× |
| 6 | 1 + MAtt with self-attention | 72.3M | 27.41 (26.7) | 60.25/1.03× | 0.105/1.00× |
| 7 | 1 + MAtt with original AAN | 72.2M | 27.36 (26.7) | 46.13/1.35× | 0.098/1.07× |
| 8 | 1 + $bs = 50K$ | 72.3M | 27.84 (27.2) | ⋆/1.00× | ⋆/1.00× |
| 9 | 1 + $>12$ layers + $bs = 25K \sim 50K$ | - | - | - | - |
| 10 | 4 + $>12$ layers + $bs = 25K \sim 50K$ | - | - | - | |
| 11 | 3 + 12 layers + $bs = 40K, dp_r = 0.3, dp_a = 0.2$ | 116.4M | 28.27 (27.6) | 102.9/1.00×$^\dagger$ | 0.188/1.00×$^\dagger$ |
| 12 | 11 + T2T | 116.5M | 28.03 (27.4) | 107.7/0.96×$^\dagger$ | 0.191/0.98×$^\dagger$ |
| 13 | 11 + MAtt | 103.8M | 28.55 (27.9) | 67.12/1.53×$^\dagger$ | 0.164 /1.15×$^\dagger$ |
| 14 | 3 + 20 layers + $bs = 44K, dp_r = 0.3, dp_a = 0.2$ | 175.3M | 28.42 (27.7) | 157.8/1.00×$^\ddagger$ | 0.283/1.00×$^\ddagger$ |
| 15 | 14 + T2T | 175.3M | 28.27 (27.6) | 161.2/0.98×$^\ddagger$ | 0.289/0.98×$^\ddagger$ |
| 16 | 14 + MAtt | 154.3M | **28.67** (**28.0**) | 108.6/1.45×$^\ddagger$ | 0.251/1.13×$^\ddagger$ |

Table 3: Tokenized case-sensitive BLEU (in parentheses: sacreBLEU) on WMT14 En-De translation task. *#Param*: number of model parameters. △*Dec*: decoding time (seconds)/speedup on newstest2014 dataset with a batch size of 32. △*Train*: training time (seconds)/speedup per training step evaluated on 0.5K steps with a batch size of 1K target tokens. Time is averaged over 3 runs using Tensorflow on a single TITAN X (Pascal). "-": optimization failed and no result. "⋆": the same as model ①. $^\dagger$ and $^\ddagger$: comparison against ⑪ and ⑭ respectively rather than ①. *Base*: the baseline Transformer with base setting. Bold indicates best BLEU score. $dp_a$ and $dp_r$: dropout rate on attention weights and residual connection. $bs$: batch size in tokens.

### 7.3 WMT14 En-De Translation Task

Table 3 summarizes translation results under different settings. Applying DS-Init and/or MAtt to Transformer with 6 layers slightly decreases translation quality by ∼0.2 BLEU (27.59→27.35). However, they allow scaling up to deeper architectures, achieving a BLEU score of 28.55 (12 layers) and 28.67 (20 layers), outperforming all baselines. These improvements can not be obtained via enlarging the training batch size (⑧), confirming the strength of deep models.

We also compare our simplified AAN in MAtt (④) with two variants: a self-attention network (⑥), and the original AAN (⑦). Results show minor differences in translation quality, but improvements in training and decoding speed, and a reduction in the number of model parameters. Compared to the baseline, MAtt improves decoding speed by 50%, and training speed by 10%, while having 9% fewer parameters.

Result ⑨ indicates that the gradient vanishing issue prevents training of deep vanilla Transformers, which cannot be solved by only simplifying the decoder via MAtt (⑩). By contrast, both T2T and DS-Init can help. Our DS-Init improves norm preservation through specific parameter initialization, while T2T reschedules the LN position. Results in Table 3 show that T2T underperforms DS-Init by 0.2 BLEU on average, and slightly increases training and decoding time (by 2%) compared to the original Transformer due to additional LN layers. This suggests that our solution is more effective and efficient.

| ID | BLEU | | PPL | |
|---|---|---|---|---|
| | Train | Dev | Train | Dev |
| 1 | 28.64 | 26.16 | 5.23 | 4.76 |
| 11 | 29.63 | 26.44 | **4.48** | **4.38** |
| 12 | **29.75** | 26.16 | 4.60 | 4.49 |
| 13 | 29.43 | **26.51** | 5.09 | 4.71 |
| 14 | 30.71 | 26.52 | **3.96** | **4.32** |
| 15 | **30.89** | 26.53 | 4.09 | 4.41 |
| 16 | 30.25 | **26.56** | 4.62 | 4.58 |

Table 4: Tokenized case-sensitive BLEU (BLEU) and perplexity (PPL) on training (Train) and development (newstest2013, Dev) set. We randomly select 3K sentence pairs as our training data for evaluation. Lower PPL is better.

Surprisingly, training deep Transformers with both DS-Init and MAtt improves not only running efficiency but also translation quality (by 0.2 BLEU), compared with DS-Init alone. To get an improved understanding, we analyze model performance on both training and development set. Results in Table 4 show that models with DS-Init yield the best perplexity on both training and development set, and those with T2T achieve the best BLEU on the training set. However, DS-Init+MAtt performs best in terms of BLEU on the development set. This indicates that the success of DS-Init+MAtt comes from its better generalization rather than better fitting training data.

We also attempt to apply DS-Init on the encoder alone or the decoder alone for 12-layer models. Unfortunately, both variants lead to unstable optimization where gradients tend to explode at the

| Task | Model | #Param | BLEU | △Dec | △Train |
|---|---|---|---|---|---|
| WMT14 En-Fr | Base + 6 layers | 76M | 39.09 | 167.56/1.00× | 0.171/1.00× |
| | Ours + Base + 12 layers | 108M | **40.58** | 173.62/0.97× | 0.265/0.65× |
| IWSLT14 De-En | Base + 6 layers | 61M | 34.41 | 315.59/1.00× | 0.153/1.00× |
| | Ours + Base + 12 layers | 92M | **35.63** | 329.95/0.96× | 0.247/0.62× |
| WMT18 En-Fi | Base + 6 layers | 65M | 15.5 (50.82) | 156.32/1.00× | 0.165/1.00× |
| | Ours + Base + 12 layers | 96M | **15.8** (**51.47**) | 161.74/0.97× | 0.259/0.64× |
| WMT18 Zh-En | Base + 6 layers | 77M | 21.1 | 217.40/1.00× | 0.173/1.00× |
| | Ours + Base + 12 layers | 108M | **22.3** | 228.57/0.95× | 0.267/0.65× |

Table 5: Translation results on different tasks. Settings for BLEU score is given in Section 7.1. Numbers in bracket denote chrF score. Our model outperforms the vanilla base Transformer on all tasks. "*Ours*": DS-Init+MAtt.

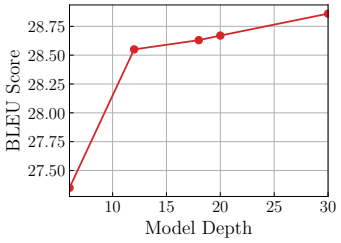


Figure 3: Test BLEU score on newstest2014 with respect to model depth for Transformer+DS-Init+MAtt.

| Model | #Param | Test14 | Test14-18 |
|---|---|---|---|
| Vaswani et al. (2017) | 213M | 28.4 | - |
| Chen et al. (2018) | 379M | 28.9 | - |
| Ott et al. (2018) | 210M | 29.3 | - |
| Bapna et al. (2018) | 137M | 28.04 | - |
| Wu et al. (2019) | 213M | **29.76**$_*$ (29.0) | 33.13 $_*$ (32.86) |
| Big + 6 layers | 233M | 29.07 (28.3) | 33.16 (32.88) |
| Ours + Big + 12 layers | 359M | 29.47 (28.7) | 33.21 (32.90) |
| Ours + Big + 20 layers | 560M | 29.62 (29.0) | **33.26** (**32.96**) |

Table 6: Tokenized case-sensitive BLEU (sacreBLEU) on WMT14 En-De translation task. "*Test14-18*": BLEU score averaged over newstest2014∼newstest2018. $^*$: results obtained by running code and model released by Wu et al. (2019). "*Ours*": DS-Init+MAtt.

middle of training. We attempt to solve this issue with gradient clipping of rate 1.0. Results show that this fails for decoder and achieves only 27.89 BLEU for encoder, losing 0.66 BLEU compared with the full variant (28.55). We leave further analysis to future work and recommend using DS-Init on both the encoder and the decoder.

*Effect of Model Depth* We empirically compare a wider range of model depths for Transformer+DS-Init+MAtt with up to 30 layers. Hyperparameters are the same as for 14 except that we use 42K and 48K batch size for 18 and 30 layers respectively. Figure 3 shows that deeper Transformers yield better performance. However, improvements are steepest going from 6 to 12 layers, and further improvements are small.

#### 7.3.1 Comparison with Existing Work

Table 6 lists the results in big setting and compares with current SOTA. Big models are trained with $dp_a = 0.1$ and $dp_r = 0.3$. The 6-layer baseline and the deeper ones are trained with batch size of 48K and 54K respectively. Deep Transformer with our method outperforms its 6-layer counterpart by over 0.4 points on newstest2014 and around 0.1 point on newstest2014∼newstest2018. Our model outperforms the transparent model (Bapna et al., 2018) (+1.58 BLEU), an approach for the deep encoder. Our model performs on par with current SOTA, the dynamic convolution model (DCNN) (Wu et al., 2019). In particular, though DCNN achieves encouraging performance on newstest2014, it falls behind the baseline on other test sets. By contrast, our model obtains more consistent performance improvements.

In work concurrent to ours, Wang et al. (2019) discuss how the placement of layer normalization affects deep Transformers, and compare the original *post-norm* (which we consider our baseline) and a *pre-norm* layout (which we call T2T). Their results also show that pre-norm allows training of deeper Transformers. Our results show that deep post-norm Transformers are also trainable with appropriate initialization, and tend to give slightly better results.

### 7.4 Results on Other Translation Tasks

We use 12 layers for our model in these tasks. We enlarge the dropout rate to $dp_a = 0.3, dp_r = 0.5$ for IWSLT14 De-En task and train models on WMT14 En-Fr and WMT18 Zh-En with 500K steps. Other models are trained with the same settings as in WMT14 En-De.

We report translation results on other tasks in Table 5. Results show that our model beats the baseline on all tasks with gains of over 1 BLEU,

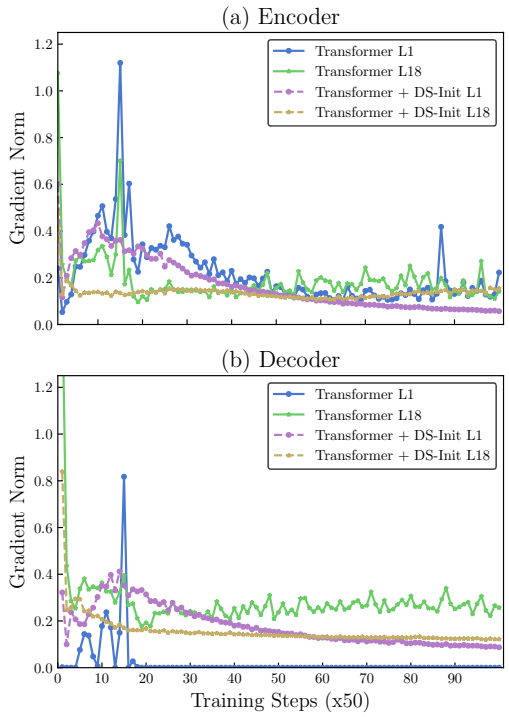


Figure 4: Gradient norm (y-axis) of the first and the last encoder layers (top) and decoder layers (bottom) in 18-layer deep Transformer over the fist 5k training steps. We use around 25k source/target tokens in each training batch. Each point in this plot is averaged over 50 training steps. "*L1*/*L18*" denotes the first/last layer. DS-Init helps stabilize the gradient norm during training.

except the WMT18 En-Fi where our model yields marginal BLEU improvements (+0.3 BLEU). We argue that this is due to the rich morphology of Finnish, and BLEU's inability to measure improvements below the word level. We also provide the chrF score in which our model gains 0.6 points. In addition, speed measures show that though our model consumes 50+% more training time, there is only a small difference with respect to decoding time thanks to MAtt.

### 7.5 Analysis of Training Dynamics

Our analysis in Figure 1 and Table 1 is based on gradients estimated exactly after parameter initialization without considering training dynamics. Optimizers with adaptive step rules, such as Adam, could have an adverse effect that enables gradient scale correction through the accumulated first and second moments. However, results in Figure 4 show that without DS-Init, the encoder gradients are less stable and the decoder gradients still suffer from the vanishing issue, particularly at the first layer. DS-Init makes the training more stable and robust.[6]

## 8 Conclusion and Future Work

This paper discusses training of very deep Transformers. We show that the training of deep Transformers suffers from gradient vanishing, which we mitigate with depth-scaled initialization. To improve training and decoding efficiency, we propose a merged attention sublayer that integrates a simplified average-based self-attention sublayer into the encoder-decoder attention sublayer. Experimental results show that deep models trained with these techniques clearly outperform a vanilla Transformer with 6 layers in terms of BLEU, and outperforms other solutions to train deep Transformers (Bapna et al., 2018; Vaswani et al., 2018). Thanks to the more efficient merged attention sublayer, we achieve these quality improvements while matching the decoding speed of the baseline model.

In the future, we would like to extend our model to other sequence-to-sequence tasks, such as summarization and dialogue generation, as well as adapt the idea to other generative architectures (Zhang et al., 2016, 2018). We have trained models with up to 30 layers each for the encoder and decoder, and while training was successful and improved over shallower counterparts, gains are relatively small beyond 12 layers. An open question is whether there are other structural issues that limit the benefits of increasing the depth of the Transformer architecture, or whether the benefit of very deep models is greater for other tasks and dataset.

## Acknowledgments

We thank the reviewers for their insightful comments. This project has received funding from the grant H2020-ICT-2018-2-825460 (ELITR) by the European Union. Biao Zhang also acknowledges the support of the Baidu Scholarship. This work has been performed using resources provided by the Cambridge Tier-2 system operated by the University of Cambridge Research Computing Service (http://www.hpc.cam.ac.uk) funded by EPSRC Tier-2 capital grant EP/P020259/1.

[6] We observe this both in the raw gradients and after taking the Adam step rules into account.

## References


Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Ankur Bapna, Mia Chen, Orhan Firat, Yuan Cao, and Yonghui Wu. 2018. Training deeper neural machine translation models with transparent attention. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3028–3033, Brussels, Belgium. Association for Computational Linguistics.

Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA. Association for Computational Linguistics.

Ondřej Bojar, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Philipp Koehn, and Christof Monz. 2018. Findings of the 2018 conference on machine translation (WMT18). In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 272–303, Belgium, Brussels. Association for Computational Linguistics.

Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc Le. 2017. Massive exploration of neural machine translation architectures. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1442–1451, Copenhagen, Denmark. Association for Computational Linguistics.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th IWSLT Evaluation Campaign, IWSLT 2014. In *Proceedings of the 11th Workshop on Spoken Language Translation*, pages 2–16, Lake Tahoe, CA, USA.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Zhifeng Chen, Yonghui Wu, and Macduff Hughes. 2018. The best of both worlds: Combining recent advances in neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 76–86. Association for Computational Linguistics.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *CoRR*, abs/1904.10509.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Mattia Antonino Di Gangi and Marcello Federico. 2018. Deep neural machine translation with weakly-recurrent units. In *Proceedings of EAMT*, Alicante, Spain.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1243–1252, International Convention Centre, Sydney, Australia. PMLR.

Felix A. Gers and Jürgen Schmidhuber. 2001. Long Short-Term Memory Learns Context Free and Context Sensitive Languages. In *Proceedings of the ICANNGA 2001 Conference*, volume 1, pages 134–137.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke S. Zettlemoyer. 2019. Constant-time machine translation with conditional masked language models. *CoRR*, abs/1904.09324.

Xavier Glorot and Y Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. *Journal of Machine Learning Research - Proceedings Track*, 9:249–256.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O.K. Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *International Conference on Learning Representations*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780.

Marcin Junczys-Dowmunt, Kenneth Heafield, Hieu Hoang, Roman Grundkiewicz, and Anthony Aue. 2018. Marian: Cost-effective high-quality neural machine translation in C++. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, Melbourne, Australia.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. 2018. Scaling neural machine translation. *arXiv preprint arXiv:1806.00187*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ngoc-Quan Pham, Thai-Son Nguyen, Jan Niehues, Markus Müller, and Alexander H. Waibel. 2019. Very deep self-attention networks for end-to-end speech recognition. *CoRR*, abs/1904.13377.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence Level Training with Recurrent Neural Networks. In *The International Conference on Learning Representations*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

K. Simonyan and A. Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*.

David R So, Chen Liang, and Quoc V Le. 2019. The evolved transformer. *arXiv preprint arXiv:1901.11117*.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. 2018. Blockwise parallel decoding for deep autoregressive models. In *Advances in Neural Information Processing Systems*, pages 10086–10095.

Ashish Vaswani, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan Gomez, Stephan Gouws, Llion Jones, Łukasz Kaiser, Nal Kalchbrenner, Niki Parmar, Ryan Sepassi, Noam Shazeer, and Jakob Uszkoreit. 2018. Tensor2Tensor for neural machine translation. In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Papers)*, pages 193–199, Boston, MA. Association for Machine Translation in the Americas.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Mingxuan Wang, Zhengdong Lu, Jie Zhou, and Qun Liu. 2017. Deep neural machine translation with linear associative unit. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 136–145, Vancouver, Canada. Association for Computational Linguistics.

Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F. Wong, and Lidia S. Chao. 2019. Learning deep transformer models for machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1810–1822, Florence, Italy. Association for Computational Linguistics.

Felix Wu, Angela Fan, Alexei Baevski, Yann Dauphin, and Michael Auli. 2019. Pay less attention with lightweight and dynamic convolutions. In *International Conference on Learning Representations*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, ukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Alireza Zaeemzadeh, Nazanin Rahnavard, and Mubarak Shah. 2018. Norm-preservation: Why residual networks can become extremely deep? *CoRR*, abs/1805.07477.

Biao Zhang, Deyi Xiong, and Jinsong Su. 2018. Accelerating neural transformer via an average attention network. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1789–1798, Melbourne, Australia. Association for Computational Linguistics.

Biao Zhang, Deyi Xiong, and Jinsong Su. 2018. Neural machine translation with deep attention. *IEEE*

*Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1.

Biao Zhang, Deyi Xiong, Jinsong Su, Hong Duan, and Min Zhang. 2016. Variational neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 521–530, Austin, Texas. Association for Computational Linguistics.

Hongyi Zhang, Yann N. Dauphin, and Tengyu Ma. 2019. Fixup initialization: Residual learning without normalization via better initialization. In *International Conference on Learning Representations*.

Xiangwen Zhang, Jinsong Su, Yue Qin, Yang Liu, Rongrong Ji, and Hongji Wang. 2018. Asynchronous bidirectional decoding for neural machine translation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, and Wei Xu. 2016. Deep recurrent models with fast-forward connections for neural machine translation. *Transactions of the Association for Computational Linguistics*, 4:371–383.

# A Discriminative Neural Model for Cross-Lingual Word Alignment

**Elias Stengel-Eskin** **Tzu-Ray Su** **Matt Post** **Benjamin Van Durme**
Johns Hopkins University
{elias, tsu6}@jhu.edu
{post, vandurme}@cs.jhu.edu

## Abstract

We introduce a novel discriminative word alignment model, which we integrate into a Transformer-based machine translation model. In experiments based on a small number of labeled examples (∼1.7K–5K sentences) we evaluate its performance intrinsically on both English-Chinese and English-Arabic alignment, where we achieve major improvements over unsupervised baselines (11–27 F1). We evaluate the model extrinsically on data projection for Chinese NER, showing that our alignments lead to higher performance when used to project NER tags from English to Chinese. Finally, we perform an ablation analysis and an annotation experiment that jointly support the utility and feasibility of future manual alignment elicitation.

## 1 Introduction

Neural-network-based models for Machine Translation (MT) have set new standards for performance, especially when large amounts of parallel text (bitext) are available. However, explicit word-to-word alignments, which were foundational to pre-neural statistical MT (SMT) (Brown et al., 1993), have largely been lost in neural MT (NMT) models. This is unfortunate: while alignments are not necessary for NMT systems, they have a wealth of applications in downstream tasks, such as transferring input formatting, incorporating lexica, and human-in-the-loop translation. Crucially, they are central to cross-lingual dataset creation via projection (Yarowsky et al., 2001), where token-level annotations in a high-resource language are projected across alignments to a low-resource language; using projection, datasets have been created for a variety of natural language processing tasks, including named-entity recognition (NER), part-of-speech tagging, parsing, information extraction (IE), and semantic role labeling (Yarowsky et al., 2001; Hwa et al., 2005; Riloff et al., 2002; Padó and Lapata, 2009; Xi and Hwa, 2005). This paradigm allows researchers to address a lack of annotated multilingual resources and changing ontologies with minimal annotation effort.

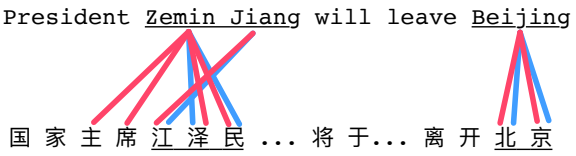


Figure 1: One application of alignments is to project annotations (e.g., Named Entities) across bitexts. Here we see a lack of correspondence between FastAlign (red) alignments and gold alignments (blue).

The introduction of attention (Bahdanau et al., 2014) has allowed NMT to advance past decoding from a single sentence-level vector representation and has been crucial in supporting fluent, meaningful translations, especially of longer sentences. However, despite their intuitive analogy to word alignments, attention-based alignments typically do not correspond very closely to gold-standard human annotations and suffer from systematic errors (Koehn and Knowles, 2017; Ghader and Monz, 2017), falling short of the promise to jointly learn to align and translate. This problem has only been exacerbated by the introduction of deeper models containing multiple attention layers and heads (e.g. Vaswani et al. (2017)). There have been several attempts to mitigate this discrepancy, mostly with a view to improving the quality of output translations (Alkhouli et al., 2016, 2018; Chen et al., 2016; Liu et al., 2016) – though some work has focused specifically on alignment (Legrand et al., 2016; Zenkel et al., 2019). We confirm that the poor performance of encoder-decoder attention for Recurrent Neural Networks (RNNs) found by Koehn and Knowles (2017) and Ghader and Monz (2017) can also be observed in the self-attentional Transformer model (Vaswani et al., 2017), further motivating an



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 910–920, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

approach focusing explicitly on alignment quality.

We introduce a novel alignment module that learns to produce high-quality alignments after training on 1.7K to 4.9K human-annotated alignments for Arabic and Chinese, respectively. While our module is integrated into the state-of-the-art Transformer model (Vaswani et al., 2017), the implementation is architecture-neutral, and can therefore be applied to RNN-based models (Schwenk, 2012; Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014) or fully convolutional models (Gehring et al., 2017). Our experiments on English-Chinese and English-Arabic bitext show that our model yields major improvements of 11 and 27 alignment F1 points (respectively) over baseline models; ablation experiments indicate with only half the data minor improvements can be observed for Chinese, and that the amount of annotated alignment data has a far greater impact on alignment score than the amount of unlabelled bitext used for pre-training.

Furthermore, our NER projection trials demonstrate a major downstream improvement when using our alignments over FastAlign. Taken together, these results motivate the further annotation of small amounts of quality alignment data in a variety of languages. We demonstrate that such annotation can be performed rapidly by untrained L2 (second language) speakers.

## 2 Related work

**Statistical Alignment** Generative alignment (Brown et al., 1993) models the posterior over a target sequence $\mathbf{t}$ given a source sequence $\mathbf{s}$ as:

$$p(\mathbf{t}|\mathbf{s}) = \sum_a \prod_i \underbrace{p(t_i|a_i, \mathbf{t}_{<i}, \mathbf{s})}_{\text{lexical model}} \underbrace{p(a_i|\mathbf{a}_{<i}, \mathbf{t}_{<i}, \mathbf{s})}_{\text{alignment model}}$$

Such models are asymmetrical: they are learned in both directions and then heuristically combined.

Discriminative alignment models, on the other hand, directly model $p(\mathbf{a}|\mathbf{s}, \mathbf{t})$, usually by extracting features from the source and target sequences and training a supervised classifier using labelled data. A comprehensive account of methods and features for discriminative alignment can be found in Tomeh (2012).

**Neural Machine Translation** The attentional encoder-decoder proposed by Bahdanau et al. (2014) greatly improved the initial sequence-to-sequence architecture; rather than conditioning each target word on the final hidden unit of the encoder, each target word prediction is conditioned on a weighted average of all the encoder hidden units. However, the weights in the average do not constitute a soft alignment, as confirmed by Koehn and Knowles (2017) and Ghader and Monz (2017). Our findings align with those of Ghader and Monz (2017), who provide a detailed analysis of where attention and alignment diverge, offering possible explanations for the modest results obtained when attempting to bias attention weights towards alignments. For example, Chen et al. (2016) introduce a loss over the attention weight that penalizes the model when the attention deviates from alignments; this yields only minor BLEU score improvements.[1]

**Neural Alignment Models** Legrand et al. (2016) develop a neural alignment model that uses a convolutional encoder for the source and target sequences and a negative-sampling-based objective. Tamura et al. (2014) introduce a supervised RNN-based aligner which conditions each alignment decision on the source and target sequences as well as previous alignment decisions. Alkhouli et al. (2018) extend the Transformer architecture for alignment by adding an additional alignment head to the multi-head encoder-decoder attention, biasing the attention weights to correspond with alignments. Similarly, Zenkel et al. (2019) introduce a single-layer attention module (akin to the standard Transformer decoder) which predicts the target word conditioned on the encoder and decoder states, following the intuition that a source word should be aligned to a target word if it is highly predictive of the target word. Unlike our model, all of these models are asymmetrical and are trained without explicit alignment data.

Following the generative formulation of the alignment problem, Alkhouli et al. (2016) present neural lexical translation and alignment models, which they train using silver-standard alignments obtained from GIZA++. A similar training strategy is used in Alkhouli and Ney (2017), who bias an RNN attention module with silver-standard alignments for use as a lexical model. In the same vein, Peter et al. (2017) improve the attention module by allowing it to peek at the next target word.

Note that unlike our framework, none of the models mentioned make use of existing gold-standard labeled data. Following Legrand et al. (2016) and Zenkel et al. (2019) we focus exclusively on alignment quality. A similar approach is taken

[1] This may also be a result of noisy training data, as they use unsupervised alignments as training data for their models.

by Ouyang and McKeown (2019), who introduce a pointer network-based model for monolingual phrase alignment which allows for the alignment of variable-length phrases.

## 3 Model

NMT systems typically use an encoder-decoder architecture, where one neural network (the encoder) learns to provide another network (the decoder) with a continuous representation of the input sequence of words in the source language, $s_1, \ldots, s_N$, from which the decoder can accurately predict the target sequence $t_1, \ldots, t_M$. The intermediate representations of the encoder and decoder are known as hidden states. After training, the hidden states of the encoder $(S_1, \ldots, S_N)$ and decoder $(T_1, \ldots, T_M)$ contain information about their co-indexed source and target words as well as their respective contexts; this can be thought of as a rough approximation of the semantic content of the source and target words. In an MT model (such as the Transformer-based model used here) the states of the encoder and decoder networks also bear information on cross-lingual correspondence.

From these representations, we can derive a natural formulation of the alignment problem: words of the source language should be aligned to the words in the target language that convey roughly the same meaning in the translated sentence i.e. their contextual vector-based meaning representations are similar.

We use the dot product as an unnormalized measure of distance between each pair of vectors $S_i$ and $T_j$. In order to apply a distance metric, the two vectors are projected into a shared space via a shared three-layer feed-forward neural network with *tanh* activations and a linear output layer, producing source and target matrices $S'_1, \ldots, S'_N$ and $T'_1, \ldots, T'_M$.[2] This is given by:

$$S'_i = W_3(tanh(W_2(tanh(W_1 S_i))))$$
$$T'_j = W_3(tanh(W_2(tanh(W_1 T_j))))$$

Taking the matrix product of the projected source and target sequences along their shared dimension yields a matrix A of $N \times M$ semantic "links", where each element $A_{ij}$ is an un-normalized distance between vectors $S'_i$ and $T'_j$, i.e. $A = [S'_1, \ldots, S'_N] \cdot [T'_1, \ldots, T'_M]^T$

These distances will be passed through a sigmoid function, normalizing them to probabilities $p(a_{ij}|\mathbf{s}, \mathbf{t})$. However, alignments are context-sensitive: the decision to align source word $i$ to target word $j$ may affect the probability of aligning the neighbors of $i$ and $j$. Therefore, before applying the sigmoid function to the matrix $A$ we convolve it with a shared weight matrix, yielding a matrix $A'$, which encodes the relationships between alignments; this is given by $A' = W_{conv} * A$, where $*$ is the convolution operator. This can be thought of as a way of conditioning each alignment decision on its neighbors, modeling $p(a_{i,j}|\mathbf{s}, \mathbf{t}, a_{i' \neq i, j' \neq j})$. This step is crucial—without a $3 \times 3$ convolution, F1 results hover below 50.

Finally, we apply the sigmoid function to $A'$ and treat each alignment decision as a separate binary classification problem, using binary cross entropy loss:

$$\sum_{i<N, j<M} \Big( \hat{a}_{ij} \log \big(p(a_{ij}|\mathbf{s}, \mathbf{t})\big) + \big(1 - \hat{a}_{ij}\big) \log \big(1 - p(a_{ij}|\mathbf{s}, \mathbf{t})\big) \Big)$$

where $\hat{a}_{ij} = 1$ if source word $i$ is aligned to target word $j$ in the gold-standard data, and 0 otherwise. Note that the convolution step ensures these are not independent alignment decisions.

When training our model, we begin by pre-training an MT model on unlabelled bitext; the weights of this model are used to initialize the encoder and decoder in the alignment model. To obtain an alignment at test time, the source and target word sequences are encoded and presented to the aligner.

## 4 Alignment Experiments

### 4.1 Baselines

We compare against two baselines: alignments obtained from FastAlign (Dyer et al., 2013), and attention-based alignments. FastAlign is a fast log-linear reparameterization of IBM Model 2 (Brown et al., 1993).

Unlike in the RNN case, where a single attention layer intervenes between the encoder and decoder, Transformers make use of multi-head and multi-layer attention. Our attentional alignments are obtained by averaging across all heads of the final multi-head encoder-decoder attention layer.

Both of these methods allow for fast online decoding, making them comparable to our model, and both have been used for word alignment; moreover, FastAlign is a standard choice for projection (Arthur et al., 2016; Fu et al., 2014; Eger et al., 2018; Agić et al., 2016; Sharoff, 2018).

[2] We use 512 hidden units in each layer.

Our model extends the Sockeye implementation of the Transformer model (Hieber et al., 2017).[3]

Following Dyer et al. (2013), FastAlign hyperparameters were tuned by training on 10K training examples, evaluating on a 100-sentence subset of the validation split. The *grow-diag-final-and* heuristic was used for symmetrization. Thresholds for both attention alignment and our model were tuned on the same 100-sentence validation subset.

### 4.2 Data

Chinese character-tokenized bitext and alignments were sourced from the GALE Chinese-English Parallel Aligned Treebank (Li et al., 2015); Arabic data was obtained from the GALE Arabic-English Parallel Aligned Treebank (Li et al., 2013). Both GALE datasets are based on broadcast news corpora; gold-standard alignments were produced by multiple trained human annotators, with several checks for quality including redundant annotation, review by senior annotators, and automatic heuristics. The GALE corpora satisfy the following criteria:

1. *Quality*: although some problems were detected in the GALE alignments (e.g. mismatched bitext, formatting errors, etc.), the multiple levels of quality control suggest a high level of overall quality.
2. *Size*: while other gold-standard alignment corpora exist (Europarl, etc.) these are typically on the order of ∼500 sentences; conversely, GALE is large enough to be used as a training set in a neural setting.
3. *Annotations*: GALE is annotated with a variety of labels, and a subset of the Chinese portion of GALE was annotated for NER as part of the OntoNotes corpus (Weischedel et al., 2013).

Sentences were split into train, test, and validation portions and cleaned by removing extraneous annotations (e.g. timestamp tags, etc.). Corpus statistics are given in Table 1.

The discriminative models (referred to as DiscAlign) were initialized with weights from an MT model pre-trained on 4 million sentences, which were tokenized using the Moses tokenizer (Koehn et al., 2007). Chinese data was drawn from the WMT 2017 En-Zh bitext, while the Arabic data was sourced from local resources; both Chinese and Arabic models were trained with a 60K-word vocabulary. For FastAlign, the GALE train split was concatenated to the pretraining bitext, while for both the discriminative and attentional models the GALE data was used to finetune a model pre-trained on bitext alone.

| Language | train | dev | test |
|---|---|---|---|
| AR | 1687 | 299 | 315 |
| ZH | 4871 | 596 | 636 |

Table 1: Num. annotated alignment sentence pairs.

Contemporary MT systems use subword units to tackle the problem of out-of-vocabulary low-frequency words. This is often implemented by byte-pair encoding (BPE) (Sennrich et al., 2016). We applied a BPE model with 30K merge operations to all of our data for the BPE experimental condition. Training alignments were expanded by aligning all subwords of a given source word to all the subwords of the target word to which the source word aligns. At test time, following Zenkel et al. (2019), to reduce BPE alignments to regular alignments, we considered a source and target word to be aligned if *any* of their subwords were aligned.[4] This heuristic was used to evaluate all BPE models.

With the task of projection for NER data in mind (Yarowsky et al., 2001), we evaluate all models on Chinese NER spans. These spans were obtained from the OntoNotes corpus, which subsumes the GALE Chinese corpus. Due to formatting differences, only a subset of the GALE sentences (287 validation, 189 test) were recoverable from OntoNotes.

We evaluate our models using macro-F1 score, as Fraser and Marcu (2007) showed that alignment error rate does not match human judgments. F1 scores were obtained using the macro-F1 scorer provided by FastAlign.

### 4.3 Results

Our model outperforms both baselines for all languages and experimental conditions. Table 2 shows that our model performs especially well in the NER setting, where we observe the largest improve-

[3] We pre-train a 6-layer model with the Adam optimizer (Kingma and Ba, 2014) using a learning rate of 0.0002, 8-headed multi-head attention, 2048-dimensional feed-forward layers, and 512-dimensional encoder/decoder output layers. We include weight-tying (Press and Wolf, 2017) between the source and target embedding layers. All other MT hyperparameters were set to the Sockeye defaults.

[4] Note that this step is necessary in order to make the produced and reference alignments comparable; to evaluate alignments, the sequences must be identical, or the evaluation is ill-posed.

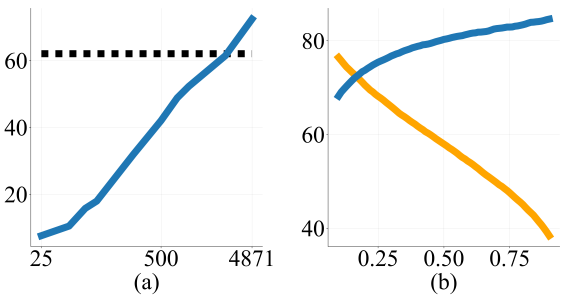


Figure 2: (a) Dev. F1 as function of training corpus size (log scale), contrast against FastAlign F1 (dashed). (b) Dev. precision (blue) and recall (orange) at different thresholds.

| Method | P | R | F1 |
|---|---|---|---|
| Avg. attention. | 36.30 | 46.17 | 40.65 |
| Avg. attention (BPE) | 37.89 | 49.82 | 43.05 |
| Avg. attention (NER) | 16.57 | 35.85 | 22.66 |
| FastAlign | 80.46 | 50.46 | 62.02 |
| FastAlign (BPE) | 70.41 | 55.43 | 62.03 |
| FastAlign (NER) | 83.70 | 49.54 | 62.24 |
| DiscAlign | 72.92 | 73.91 | **73.41** |
| DiscAlign (BPE) | 69.36 | 67.11 | **68.22** |
| DiscAlign (NER) | 74.52 | 77.05 | **75.78** |
| DiscAlign (NER) +prec. | 84.69 | 58.41 | 69.14 |

Table 2: Precision, Recall, and F1 on Chinese GALE test data. BPE indicates "with BPE", and NER denotes restriction to NER spans.

| Method | P | R | F1 |
|---|---|---|---|
| Avg. attention. | 8.46 | 32.50 | 13.42 |
| Avg. attention (BPE) | 10.11 | 17.27 | 12.75 |
| FastAlign | 62.26 | 51.06 | 56.11 |
| FastAlign (BPE) | 62.74 | 51.25 | 56.42 |
| DiscAlign | 91.30 | 75.66 | **82.74** |
| DiscAlign (BPE) | 87.05 | 76.98 | **81.71** |

Table 3: Precision, Recall, and F1 on Arabic GALE test data.

ment over FastAlign ($\Delta = 14$). By increasing the threshold $\alpha$ above which $p(a_{i,j}|\mathbf{s}, \mathbf{t})$ is considered an alignment we can obtain high-precision alignments, exceeding FastAlign's precision and recall. The best threshold values on the development set were $\alpha = 0.13$ and $\alpha = 0.14$ for average attention on data with and without BPE (respectively) and $\alpha = 0.15$ for the discriminative settings ($\alpha = 0.5$ was used for the high-precision "+prec" setting).

Table 3 further underscores these findings; we observe an even more dramatic increase in performance on Arabic, despite the fact that the model is trained on only 1.7K labelled examples. Furthermore, we see that average attention performs abysmally in Arabic. The best thresholds for Arabic differed substantially from those for Chinese: for average attention (with and without BPE) the values were $\alpha = 0.05$ and $\alpha = 0.1$, while for the discriminative aligner they were $\alpha = 0.94$ and $\alpha = 0.99$ (compared to 0.15 for Chinese). This is reflected in the high precision of the discriminative aligner.

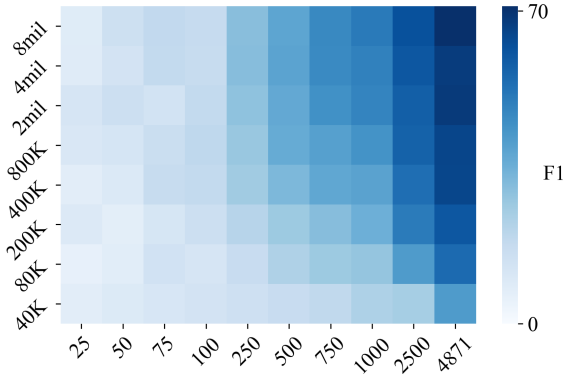


Figure 3: Chinese F1 performance with respect to the amount of unlabelled pretraining bitext (y axis) and the number of sentences annotated for alignment (x axis); increasing the number of annotated sentences has a far greater impact on the F1 score than increasing the amount of bitext.

Figure 2(a) shows the relationship between the number of training examples and validation F1 performance for Chinese, with the performance of FastAlign overlaid as a benchmark. We observe that with a little over half the training examples (2500), we perform comparably to FastAlign.

In Figure 3, we demonstrate that labelled alignment data has a greater impact than the amount of pre-training bitext. For example, halving the number of pre-training examples from 8 million to 4 million only reduces F1 by 3.69, but halving the number of labelled finetuning examples while keeping the number of pre-training sentences constant at 8 million lowers the performance by 9.44 F1.

**Analysis** Taken together, Table 2 and Figure 2 support the use of discriminative alignment models in settings where test data is presented in an online fashion as well as in cases where high-quality alignments matter. Most importantly, the results in the NER condition highlight the applicability of our model to dataset projection for NER.

Figure 2(b) also shows that by varying the minimum threshold $\alpha$ on $p(a_{ij}|\mathbf{s}, \mathbf{t})$, we can manipulate the precision-recall tradeoff, giving our model an advantage in scenarios where high precision alignments are required (e.g., dataset projection) as well as in ones where high recall is preferred (e.g., correction by human annotators).

Our results underscore the poor performance of attentional alignments. One factor driving the low F1 scores observed may be the Transformer's design: each attention head in the Transformer's multi-head attention layer learns to attend to different aspects of the encoder states; thus, averaging across all heads is likely to yield even noisier results than the RNN model, where there is effectively only one attention head. Another reason attention performs poorly may be that attention heads are learning task- and dataset- specific patterns that do not match alignments but nevertheless provide useful cues for translation. This would support the observations of Ghader and Monz (2017), who find that for certain parts of speech (e.g. verbs) alignment is very poorly correlated with translation loss.

These findings illustrate the benefit of supervised training for alignment, suggesting further annotation of gold-standard alignments is warranted, with an emphasis on alignment quality.[5] In particular, Figure 3 indicates that labelled alignment data is far more valuable than bitext, meaning that efforts should be focused on annotating existing bitext for alignment; FastAlign's performance (when trained on the entire 4 million-sentence training bitext and the alignment bitext) can be surpassed after pre-training the translation model on only 400K sentences of unlabelled bitext and the full alignment set (4871 sentences).

## 5 NER Experiments

In Table 2 we evaluated the quality of our model's alignments intrinsically, showing that they dramatically outperform the baseline models, especially for NER spans. To show that these improvements translate to downstream performance, we conduct projection experiments for NER using the Chinese and English data from the OntoNotes 5.0 corpus.

**Projection** Dataset projection, as defined by Yarowsky et al. (2001), is a method of obtaining data in a low resource language by leveraging existing bitext and high-resource annotations. As shown (for a single sentence) in Figure 1, a set of sentences in a high-resource language for which there exist task-specific token-level annotations (e.g. an English NER dataset) is translated into the target language. Using alignments, the annotations are "projected" from the source text to the target; each target token receives the label of the source token it is aligned to (or a default label if the token is unaligned). This produces an annotated resource in the target language which can be used to train or pre-train a model.

**Data** We project the English OntoNotes NER training data (53K sentences) by first translating it to Chinese using our BPE translation model, and then aligning the source and target sequences using FastAlign and our best discriminative model. We create two datasets, one where the tags have been projected using FastAlign alignments and another using the higher-quality DiscAlign alignments. OntoNotes contains gold-labelled NER data for 18 tag types in both Chinese and English. The official splits[6] were used. Because there are fewer training examples for Chinese than English, two NER models were trained, one with the same number of projected sentences as there were gold Chinese sentences (36K), and the second using the full English dataset (53K sentences) projected to Chinese. The data was pre-processed in following the same procedure as §4

**NER Model** Given the recent successes of contextual encoders for sequence modeling, especially for NER, we make use of a pre-trained BERT model (Devlin et al., 2018), where the NER label is determined by the softmax output of a linear layer situated on top of the encoder.[7] Each model was trained for 4 to 8 epochs.

We explore the performance of the BERT NER model when trained on both gold-standard and projected datasets. In addition, we analyze its performance when trained on decreasing amounts of gold data, and contrast those results with the performance of a model pre-trained on projected data and fine-tuned with gold data.

[5] In addition to the preprocessing steps required to clean the data, a bilingual speaker pointed out several systematic deviations from the alignment protocol by GALE annotators.

[6] http://conll.cemantix.org/2012/data.html

[7] An NER model was adapted from http://github.com/kamalkraj/BERT-NER to use a pre-trained Chinese model and allow for incremental checkpointing.

| Method | # train | P | R | F1 |
|---|---|---|---|---|
| Zh Gold | 36K | 75.46 | 80.55 | 77.81 |
| FastAlign | 36K | 38.99 | 36.61 | 37.55 |
| FastAlign | 53K | 39.46 | 36.65 | 37.77 |
| DiscAlign | 36K | 51.94 | 52.37 | 51.76 |
| DiscAlign | 53K | 51.92 | 51.93 | 51.57 |

Table 4: F1 results on OntoNotes test for systems trained on data projected via FastAlign and DiscAlign.

| # train | P | R | F1 |
|---|---|---|---|
| 500 | 41.43 | 57.97 | 48.07 |
| 1000 | 58.52 | 68.05 | 62.72 |
| 2500 | 68.30 | 74.93 | 71.35 |
| 4871 | 72.00 | 76.90 | 74.30 |
| 10K | 73.13 | 78.79 | 75.77 |

Table 5: OntoNotes test set performance when trained on subsamples of the Chinese gold NER data.

| # gold | P | R | F1 | $\Delta$F1 |
|---|---|---|---|---|
| 500 | 63.16 | 68.90 | 65.74 | 17.67 |
| 1000 | 66.39 | 72.23 | 69.13 | 6.41 |
| 2500 | 69.57 | 75.52 | 72.33 | 0.98 |
| 4871 | 71.11 | 76.38 | 73.60 | -0.7 |
| 10K | 72.52 | 78.32 | 75.24 | -0.53 |
| full | 75.04 | 80.18 | 77.45 | -0.36 |

Table 6: OntoNotes test performance when pre-trained on projected data and finetuned on varying amounts of Chinese gold NER data. $\Delta$ F1 compares these values to corresponding rows in Table 5.

### 5.1 Results & Analysis

Table 4 shows that while NER systems trained on projected data do categorically worse than an NER system trained on gold-standard data, the higher-quality alignments obtained from DiscAlign lead to a major improvement in F1 when compared to FastAlign.

For many tasks, an English resource (i.e. the English portion of the OntoNotes data) exists along with bitext between English and a given target language, but very few resources are available in the target language. Table 5 shows that projection outperforms a system trained on gold data when there is very little gold data available (500 sentences); as the amount of data increases, these gains disappear. However, Table 6 indicates that projected data provides a useful pre-training objective for NER, making low-data NER models perform about as well as if they had been trained on twice as much gold data.

Figure 4 shows that for nearly all classes, especially rare ones, pre-training on projected data improves downstream results when there is little annotated data. This is partly due to the fact some classes (e.g. LANGUAGE, ORDINAL, etc.) may appear very rarely in the 500-sentence training set ($\leq$ 21 times). Note that all classes appear at least 5 times in the training data. However, these classes are far more common in the much larger projected dataset. In addition, the major gains seen in tags with higher support in the gold data (e.g. MONEY, LOCATION) suggest that pre-training the NER model on projected data does more than merely expose the model to unseen classes.

## 6 Human Evaluation

In §4 we established that our discriminatively-trained neural aligner outperforms unsupervised baselines, especially on NER spans; in §5 we verified that the alignments it produces can be productively applied a downstream IE task (NER) via dataset projection.

However, unlike these unsupervised baselines, our aligner requires labelled data on the order of thousands of sentences, and thus cannot be applied to language pairs for which no labelled alignment data exists (most languages). We contend that using a web-based crowdsourcing interface [8], alignment annotation can be performed rapidly by annotators with minimal experience; to justify this claim, we conduct a small timed experiment using untrained human annotators. Six L2 Chinese speakers (L1 English) were asked to annotate English-Chinese bitext for alignment. These sentences were taken from the GALE development portion of the OntoNotes dataset. Note that these annotators had no experience in annotating data for alignment, and had approximately equivalent amounts of Chinese instruction. Annotators were introduced to the task and the alignment interface, and then given 50 minutes to align the sentences as accurately as possible. They were divided into two groups, with each group annotating a different partition of sentences.

### 6.1 Results & Analysis

Table 7 shows that alignment can be performed rapidly, at 4.4 sentences per minute. The annotators achieve high overall precision for alignment, but fall short on recall. When only alignments of

[8] https://github.com/hltcoe/tasa

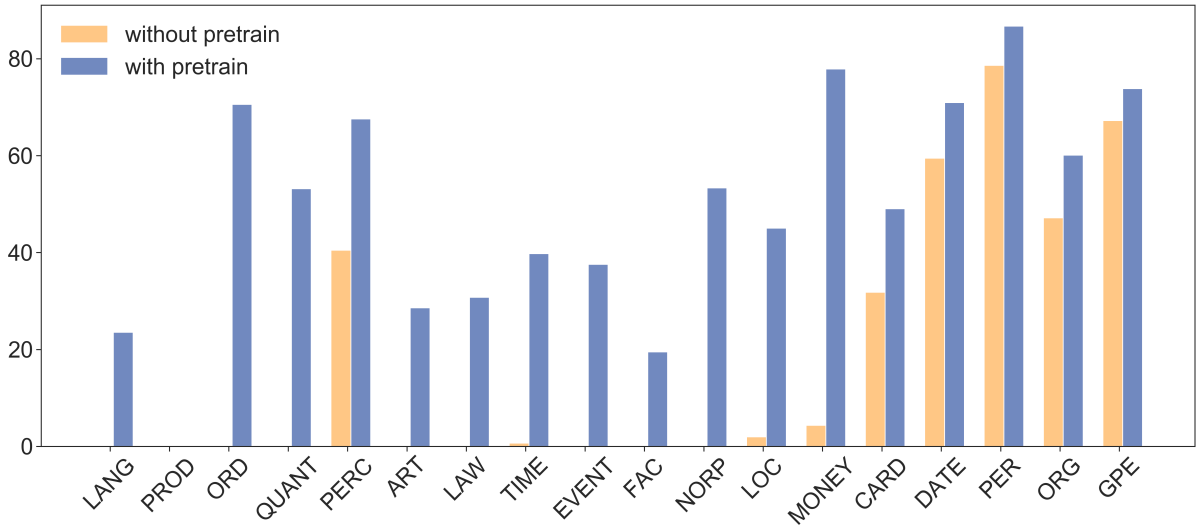


Figure 4: Comparing F1 across different tag types when a model is trained on 500 gold NER sentences to when it is pre-trained on projected data and then finetuned on 500 gold sentences, ranked by number of train examples.

| Model | sents/m | P | R | F1 |
|---|---|---|---|---|
| Human | 4.4 | 90.09 | 62.85 | 73.92 |
| DiscAlign | - | 74.54 | 72.15 | 73.31 |
| Hu. (NER) | - | 87.73 | 71.02 | 78.24 |
| DA (NER) | - | 77.37 | 67.69 | 71.94 |

Table 7: Sentences per minute and average scores against gold-labelled data for sentences annotated for alignment by human annotators (Hu.), compared to DiscAlign (DA) on the same sentences.

NER spans are considered, their F1 score improves considerably. Additionally, human annotators outperform DiscAlign when evaluated on the same sentences.

These are promising results, as they indicate that entirely untrained L2 annotators can rapidly annotate data. Their relatively low recall scores might be due to their lack of training, and given more extensive training and guidelines we would expect them to improve.

## 7 Conclusion & Future Work

We described a model extension to NMT which supports discriminative alignment, where fewer than 5K Chinese and 1.7K Arabic labelled sentences lead to better F1 scores in all tested scenarios than standardly employed mechanisms. By evaluating the model on both Arabic and Chinese we demonstrated that our model's improvements generalize across typologically divergent languages. We showed that in Chinese our model's performance is particularly strong for NER spans.

We projected a dataset for NER from English to Chinese using our alignments, verifying that our model's higher alignment quality led to downstream improvements; our aligner's intrinsic improvements result in major gains over FastAlign in a downstream NER evaluation. Comparing the performance of our projected dataset with a gold-standard dataset, we showed that a large projected dataset can outperform a small labelled one, and that projection can be leveraged as a method of data augmentation for low-resource settings.

Finally, we showed that untrained L2 annotators can rapidly annotate bitext for alignment, and yield promising results for both full alignment and NER-specific annotation, confirming that we can collect additional data in a variety of languages.

Based on our findings that:

- discriminative alignments outperform common unsupervised baselines in two typologically divergent languages
- this performance boost leads to major downstream improvements on NER
- only a small amount of labelled data is needed to realize these improvements
- and that these labelled examples can be obtained from L2 speakers with minimal training

we conclude with a call for additional annotation efforts in a wider variety of languages. While multilingual datasets directly annotated for a given task typically lead to the highest-performing systems (see Table 4) these datasets are task-specific and not robust to ontology changes. In contrast, the framework of projection via discriminatively-trained alignments which we present is task-agnostic (any

token-level annotations can be projected) and requires only one source-side dataset to be annotated.

Future work will explore different sets of languages paired with new, higher-quality elicited alignments of existing bitext. We will also explore architecture modifications to allow for task-specific annotation, training, and inference for a given IE task, where only certain spans are of interest (i.e. training our aligner only on NER-span alignments). We expect that asking annotators to only align Named Entities (or some other token type) will greatly increase the annotation speed. These experiments will extend beyond NER to other token-level tasks such as coreference resolution and event detection.

## Acknowledgements

This work was supported by DARPA LORELEI and AIDA. We thank the anonymous reviewers for their constructive feedback. The views and conclusions expressed herein are those of the authors and should not be interpreted as representing official policies or endorsements of DARPA or the U.S. Government.

## References

Željko Agić, Anders Johannsen, Barbara Plank, Héctor Alonso Martínez, Natalie Schluter, and Anders Søgaard. 2016. Multilingual projection for parsing truly low-resource languages. *Transactions of the Association for Computational Linguistics*, 4:301–312.

Tamer Alkhouli, Gabriel Bretschner, and Hermann Ney. 2018. On the alignment problem in multi-head attention-based neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 177–185.

Tamer Alkhouli, Gabriel Bretschner, Jan-Thorsten Peter, Mohammed Hethnawi, Andreas Guta, and Hermann Ney. 2016. Alignment-based neural machine translation. In *Proceedings of the First Conference on Machine Translation: Volume 1, Research Papers*, pages 54–65. Association for Computational Linguistics.

Tamer Alkhouli and Hermann Ney. 2017. Biasing attention-based recurrent neural networks using external alignment information. In *Proceedings of the Second Conference on Machine Translation*, pages 108–117.

Philip Arthur, Graham Neubig, and Satoshi Nakamura. 2016. Incorporating discrete translation lexicons into neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1557–1567. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.

Wenhu Chen, Evgeny Matusov, Shahram Khadivi, and Jan-Thorsten Peter. 2016. Guided alignment training for topic-aware neural machine translation. *arXiv preprint arXiv:1607.01628*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Chris Dyer, Victor Chahuneau, and Noah A Smith. 2013. A simple, fast, and effective reparameterization of IBM Model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648.

Steffen Eger, Johannes Daxenberger, Christian Stab, and Iryna Gurevych. 2018. Cross-lingual argumentation mining: Machine translation (and a bit of projection) is all you need! In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 831–844.

Alexander Fraser and Daniel Marcu. 2007. Measuring word alignment quality for statistical machine translation. *Computational Linguistics*, 33(3):293–303.

Ruiji Fu, Bing Qin, and Ting Liu. 2014. Generating Chinese named entity data from parallel corpora. *Frontiers of Computer Science*, 8(4):629–641.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1243–1252. JMLR. org.

Hamidreza Ghader and Christof Monz. 2017. What does attention in neural machine translation pay attention to? In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 30–39.

Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, and Matt Post. 2017. Sockeye: A toolkit for neural machine translation. *arXiv preprint arXiv:1712.05690*.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural language engineering*, 11(3):311–325.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the association for computational linguistics companion volume proceedings of the demo and poster sessions*, pages 177–180.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39.

Joël Legrand, Michael Auli, and Ronan Collobert. 2016. Neural network-based word alignment through score aggregation. In *Proceedings of the First Conference on Machine Translation: Volume 1, Research Papers*, pages 66–73. Association for Computational Linguistics.

Xuansong Li, Stephen Grimes, Safa Ismael, Stephanie Strassel, Mohamed Maamouri, and Ann Bies. 2013. GALE Arabic-English parallel aligned treebank – broadcast news part 1; LDC2013T14.

Xuansong Li, Stephen Grimes, Stephanie Strassel, Xiaoyi Ma, Nianwen Xue, Mitch Marcus, and Ann Taylor. 2015. GALE Chinese-English parallel aligned treebank–training; LDC2015T06.

Lemao Liu, Masao Utiyama, Andrew Finch, and Eiichiro Sumita. 2016. Neural machine translation with supervised attention. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3093–3102.

Jessica Ouyang and Kathleen McKeown. 2019. Neural network alignment for sentential paraphrases. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 4724–4735.

Sebastian Padó and Mirella Lapata. 2009. Cross-lingual annotation projection for semantic roles. *Journal of Artificial Intelligence Research*, 36:307–340.

Jan-Thorsten Peter, Arne Nix, and Hermann Ney. 2017. Generating alignments using target foresight in attention-based neural machine translation. *The Prague Bulletin of Mathematical Linguistics*, 108(1):27–36.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 157–163.

Ellen Riloff, Charles Schafer, and David Yarowsky. 2002. Inducing information extraction systems for new languages via cross-language projection. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

Holger Schwenk. 2012. Continuous space translation models for phrase-based statistical machine translation. *Proceedings of COLING 2012: Posters*, pages 1071–1080.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1715–1725.

Serge Sharoff. 2018. Language adaptation experiments via cross-lingual embeddings for related languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Akihiro Tamura, Taro Watanabe, and Eiichiro Sumita. 2014. Recurrent neural networks for word alignment model. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1470–1480.

Nadi Tomeh. 2012. *Discriminative Alignment Models For Statistical Machine Translation*. Ph.D. thesis, Université Paris Sud-Paris XI.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al. 2013. Ontonotes Release 5.0 LDC2013T19. *Linguistic Data Consortium, Philadelphia, PA*.

Chenhai Xi and Rebecca Hwa. 2005. A backoff model for bootstrapping resources for non-english languages. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 851–858. Association for Computational Linguistics.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of the first international conference on Human language technology research*, pages 1–8. Association for Computational Linguistics.

Thomas Zenkel, Joern Wuebker, and John DeNero. 2019. Adding interpretable attention to neural translation models improves word alignment. *arXiv preprint arXiv:1901.11359*.

# One Model to Learn Both: Zero Pronoun Prediction and Translation

**Longyue Wang**
Tencent AI Lab
vinnylywang@tencent.com

**Zhaopeng Tu**
Tencent AI Lab
zptu@tencent.com

**Xing Wang**
Tencent AI Lab
brightxwang@tencent.com

**Shuming Shi**
Tencent AI Lab
shumingshi@tencent.com

## Abstract

Zero pronouns (ZPs) are frequently omitted in pro-drop languages, but should be recalled in non-pro-drop languages. This discourse phenomenon poses a significant challenge for machine translation (MT) when translating texts from pro-drop to non-pro-drop languages. In this paper, we propose a unified and discourse-aware ZP translation approach for neural MT models. Specifically, we jointly learn to predict and translate ZPs in an end-to-end manner, allowing both components to interact with each other. In addition, we employ hierarchical neural networks to exploit discourse-level context, which is beneficial for ZP prediction and thus translation. Experimental results on both Chinese⇒English and Japanese⇒English data show that our approach significantly and accumulatively improves both translation performance and ZP prediction accuracy over not only baseline but also previous works using external ZP prediction models. Extensive analyses confirm that the performance improvement comes from the alleviation of different kinds of errors especially caused by subjective ZPs.

## 1 Introduction

Zero anaphora is a discourse phenomenon, where pronouns can be omitted when they are pragmatically or grammatically inferable from intra- and inter-sentential context (Li and Thomson, 1979). However, translating such implicit information (i.e. zero pronoun, ZP) poses various difficulties for machine translation (MT) in terms of completeness and correctness. Although neural models are getting better at learning representations, it is still difficult to implicitly learn complex ZPs in a general model. Actually, ZP prediction and translation need to not only understand the semantics or intentions of a single sentence, but also utilize its discourse-level context.

Two technological advances in the field of ZP and MT, have seen vast progress over the last decades, but they have been developed very much in isolation. Early studies (Chung and Gildea, 2010; Le Nagard and Koehn, 2010; Xiang et al., 2013) fed MT systems with the results of ZP prediction models, which are trained on a small-scale and non-homologous data compared to MT models. To narrow the data-level gap, Wang et al. (2016) proposed an automatic method to annotate ZPs by utilizing the parallel corpus of MT. The homologous data for both ZP prediction and translation leads to significant improvements on translation performances for both statistical MT (Wang et al., 2016) and neural MT models (Wang et al., 2018a). However, such approaches still require external ZP prediction models, which have a low accuracy of 66%. The numerous errors of ZP prediction errors will be propagated to translation models, which leads to new translation problems. In addition, relying on external ZP prediction models in decoding makes these approaches unwieldy in practice, due to introducing more computation cost and pipeline complexity.

In this work, we try to further bridge the model-level gap by jointly modeling ZP prediction and translation. Joint learning has proven highly effective on alleviating the error propagation problem, such as joint parsing and translation (Liu and Liu, 2010), as well as joint tokenization and translation (Xiao et al., 2010). Similarly, we expect that ZP prediction and translation could interact with each other: prediction offers more ZP information beyond 1-best result to translation and translation helps prediction resolve ambiguity. Specifically, we first cast ZP prediction as a sequence labeling task with a neural model, which is trained jointly with a standard neural machine translation (NMT) model in an end-to-end manner. We leverage the auto-annotated ZPs to supervise the learning of ZP



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 921–930, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

prediction component, which releases the reliance on external ZP knowledge in decoding phase.

In addition, previous studies revealed that discourse-level information can better tackle ZP resolution, because around 23% of ZPs appear two or more sentences away from their antecedents (Zhao and Ng, 2007; Chen and Ng, 2013). Inspired by these findings, we exploit inter-sentential context to further improve ZP prediction and thus translation. Concretely, we employ hierarchical neural networks (Sordoni et al., 2015; Wang et al., 2017) to summarize the context of previous sentences in a text, which is integrated to the joint model for ZP prediction.

We validate the proposed approach on the widely-used data for ZP translation (Wang et al., 2018a), which consist of 2.15M Chinese–English sentence pairs. Experimental results show that the joint model indeed improves performances on both ZP prediction and translation. Incorporating discourse-level context further improves performances, and outperforms ther external ZP prediction model (Wang et al., 2018a) by +2.29 BLEU points in translation and +11% in prediction accuracy. Experimental results on a further Japanese–English translation task show that our model consistently outperforms both the baseline and the external ZP prediction model, demonstrating the universality of the proposed approach.

The key contributions of this paper are:

1. We propose a single model to jointly learn ZP prediction and translation, which improves performances on both tasks by allowing the two components to interact with each other.

2. Our study demonstrates the effectiveness of discourse-level context for ZP prediction.

3. Based on our manually-annotated testset, we conduct extensive analyses to assess ZP prediction and translation.

## 2 Background

### 2.1 Zero Pronoun

In pro-drop languages such as Chinese and Japanese, ZPs occur much more frequently compared to non-pro-drop languages such as English (Zhao and Ng, 2007). As seen in Table 1, the subject pronoun ("我") and the object pronoun ("它") are omitted in Chinese sentences ("Inp.") while these pronouns are all compulsory in their English translations ("Ref."). This is not a problem for human beings since we can easily recall these missing pronoun from the context. Taking the second sentence for example, the pronoun "它" is an anaphoric ZP that refers to the antecedent ("蛋糕") in previous sentence, while the non-anaphoric pronoun "我" can still be inferred from the whole sentence. The first example also indicates the necessity of intra-sentential information for ZP prediction.

| | |
|---|---|
| Inp. | 等 我 搬进来，**(我)** 能 买 台 电视 吗? |
| Ref. | Can **I** get a TV when I move in? |
| Out. | When I move in **to buy** a TV. |
| Inp. | 这块 蛋糕 很 美味！ 你 烤 的 **(它)** 吗? |
| Ref. | The cake is very tasty! Did you bake **it**? |
| Out. | The cake is delicious! **Are you baked**? |

Table 1: Examples of ZPs and translations where words in brackets are ZPs that are invisible in decoding and underlined words are antecedents of anaphoric ZPs. This leads to problems for NMT in respect of completeness (first case) and correctness (second case). "Inp." and "Ref." indicate Chinese input and English translation, respectively. "Out." represents the output of a NMT model.

However, ZP poses a significant challenge for translation models from pro-drop to non-pro-drop languages, where ZPs are normally omitted in the source side but should be generated overly in the target side. As shown in Table 1, even a strong NMT model fails to recall the implicit information, which lead to problems like *incompleteness* and *incorrectness*. The first case is translated into "When I move in to buy a TV", which makes the output miss subject element (incompleteness). The second case is translated into "Are you baked?", while the correct translation should be "Did you bake it?" (incorrectness).

### 2.2 Bridging Data Gap Between ZP Prediction and Translation

Recent efforts have explored ways to bridge the gap of ZP prediction and translation (Wang et al., 2016, 2018a,b) by training both models on the homologous data. The pipeline involves two phases, as described below.

**Translation-Oriented ZP Prediction** Its goal is to recall the ZPs in the source sentence (i.e. pro-drop language) with the information of the target sentence (i.e. non-pro-drop language) in a paral-

lel corpus. Taking the second case (assuming that Inp. and Ref. are sentence pair in a parallel corpus) in Table 1 for instance, the ZP "它 (*it*)" is dropped in the Chinese side while its equivalent "it" exists in the English side. It is possible to identify the ZP position (between "的" and "吗") by alignment information, and then recover the ZP word "它" by a language model (scoring all possible pronoun candidates and select the one with the lowest perplexity). Wang et al. (2016) proposed a novel approach to automatically annotate ZPs using alignment information from bilingual data, and the auto-annotation accuracy can achieve above 90%. Thus, a large amount of ZP-annotated sentences were available to train an external ZP prediction model, which was further used to annotate source sentences in test sets during the decoding phase. They integrated the ZP predictor into SMT and showed promising results on both Chinese–English and Japanese–English data.

However, their neural-based ZP prediction model still produce low accuracies on predicting ZPs, which is 66% in F1 score. This is a key problem for the pipeline framework, since numerous errors would be propagated to the subsequent translation process.

**Translation with ZP-Annotated Data** An intuitive way to exploit the annotated data is to train a standard NMT model on the annotated parallel corpus, which decodes the input sentence annotated by the external ZP prediction model. Wang et al. (2018a) leveraged the encoder-decoder-reconstructor framework (Tu et al., 2017) for this task, which reconstructs the intermediate representations of NMT model back to the ZP-annotated input. The auxiliary loss on ZP reconstruction can guide the intermediate representations to learn critical information relevant to ZPs. However, their best model still needs external ZP prediction at decoding time. In response to this problem, Wang et al. (2018b) leveraged the prediction results of the ZP positions, which have relatively higher accuracy (e.g. 88%). Accordingly, they jointly learn the partial ZP prediction (*i.e.*, predict the ZP word given the externally annotated ZP position) and ZP translation.

In this work, we follow this direction with the encoder-decoder-reconstructor framework, and show our approach outperforms both strategies of using externally annotated data.

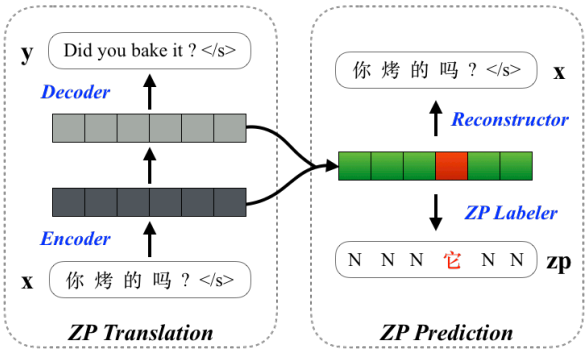


Figure 1: Architecture of the joint ZP prediction and translation model, in which ZP prediction is casted as a sequence labelling problem.

## 3 Approach

In this study, we propose a joint model to learn ZP prediction and translation, which can be further improved by leveraging discourse-level context.

- *Joint ZP Prediction and Translation* (Section 3.1) We cast ZP prediction as a sequence labeling problem, which can be trained together with ZP translation model in an end-to-end manner. This releases the reliance on external ZP prediction models (e.g. 66% or 88% accuracy), since no ZP-annotated sentence is required any more in decoding. Instead, only the high-quality annotated bilingual data (e.g. 93% accuracy) are needed.

- *Discourse-Aware ZP Prediction* (Section 3.2) We further improve ZP prediction and thus its translation with discourse-level context, which is summarized by hierarchical neural networks. The contextual representation is integrated into the reconstructor, based on which ZP prediction is conducted.

### 3.1 Joint ZP Prediction and Translation

Figure 1 illustrates the architecture of the joint model, which consists of two main components. The ZP translation component is a standard encoder-decoder NMT model, while an additional reconstructor is introduced for ZP prediction. To guarantee the reconstructor states contain enough information for ZP prediction, the reconstructor reads both the encoder and decoder states and the reconstruction score is computed by

$$R(\hat{\mathbf{x}}|\mathbf{h}^{enc}, \mathbf{h}^{dec}) = \prod_{t=1}^{T} g_r(\hat{x}_{t-1}, \mathbf{h}_t^{rec}, \hat{\mathbf{c}}_t^{enc}, \hat{\mathbf{c}}_t^{dec})$$

where $\mathbf{h}_t^{rec}$ is the hidden state in the reconstructor:

$$\mathbf{h}_t^{rec} = f_r(\hat{x}_{t-1}, \mathbf{h}_{t-1}^{rec}, \hat{\mathbf{c}}_t^{enc}, \hat{\mathbf{c}}_t^{dec}) \quad (1)$$

Here $g_r(\cdot)$ and $f_r(\cdot)$ are respective softmax and activation functions for the reconstructor. The context vectors $\hat{\mathbf{c}}_t^{enc}$ and $\hat{\mathbf{c}}_t^{dec}$ are the weighted sum of $\mathbf{h}^{enc}$ and $\mathbf{h}^{dec}$, and the weights are calculated by two interactive attention models:

$$\hat{\alpha}^{enc} = \text{ATT}_{enc}(x_{t-1}, \mathbf{h}_{t-1}^{rec}, \mathbf{h}^{enc}) \quad (2)$$
$$\hat{\alpha}^{dec} = \text{ATT}_{dec}(x_{t-1}, \mathbf{h}_{t-1}^{rec}, \mathbf{h}^{dec}, \hat{\mathbf{c}}_t^{enc}) \quad (3)$$

The interaction between two attention models leads to a better exploitation of the encoder and decoder representations (Wang et al., 2018b).

**ZP Prediction as Sequence Labelling** We cast ZP prediction as a sequence labelling task, where each word is labelled if there is a pronoun missing before it. Given the input $\mathbf{x} = \{x_1, x_2, \ldots, x_T\}$ with the last word $x_T$ being the end-of-sentence tag " ⟨eos⟩",[1] the output to be labelled is a sequence of labels $\mathbf{zp} = \{zp_1, zp_2, \ldots, zp_T\}$ with $zp_t \in \{N\} \cup \mathbb{V}_{zp}$. Among the label set, "$N$" denotes no ZP, and $\mathbb{V}_{zp}$ is the vocabulary of pronouns.[2] Taking Figure 1 as an example, the label sequence "N N N 它 N N" indicates that the pronoun "它" is missing before the fourth word "吗" in the source sentence "你 烤 的 吗？ ". More specifically, we model the probability of generating the label sequence $\mathbf{zp}$ as:

$$P(\mathbf{zp}|\mathbf{h}^{rec}) = \prod_{t=1}^{T} P(zp_t|\mathbf{h}_t^{rec}) = \prod_{t=1}^{T} g_l(zp_t, \mathbf{h}_t^{rec}) \quad (4)$$

where $g_l(\cdot)$ is softmax for the ZP labeler. As seen, we integrate the ZP generation component into the ZP translation model. There is no reliance on external ZP prediction models in decoding phase.

**Training and Testing** The newly introduced prediction component is trained together with the encoder-decoder-reconstructor:

$$J(\theta, \gamma, \psi) = \arg\max_{\theta,\gamma,\psi} \Big\{ \underbrace{\log L(\mathbf{y}|\mathbf{x};\theta)}_{\textit{likelihood}} + \underbrace{\log R(\mathbf{x}|\mathbf{h}^{enc}, \mathbf{h}^{dec};\theta)}_{\textit{reconstruction}} + \underbrace{\log P(\mathbf{zp}|\mathbf{h}^{rec};\theta,\gamma)}_{\textit{ZP labeling}} \Big\} \quad (5)$$

where $\{\theta, \gamma\}$ are respectively the parameters associated with the encoder-decoder-reconstructor and the ZP prediction component. The auxiliary prediction loss $P(\cdot)$ guides the hidden states of both the encoder-decoder and the reconstructor to embed the ZPs in the source sentence. Although the calculation of labeling loss relies on explicitly annotated labels, it is only used in training to guide the parameters to learn ZP-enhanced representations. Benefiting from the implicit integration of ZP information, we release the reliance on external ZP prediction model in testing.

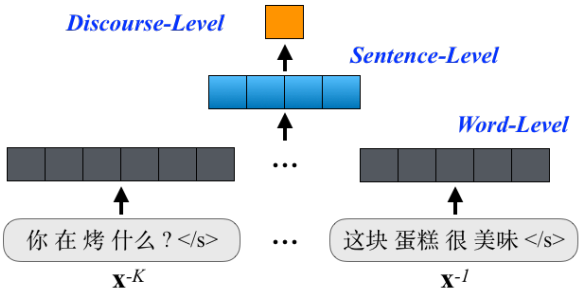


Figure 2: Architecture of hierarchical neural encoder. $\mathbf{x}^{-K}, \ldots, \mathbf{x}^{-1}$ are $K$ previous sentences before the current source sentence "你 烤 的 吗 ?" in a text.

## 3.2 Discourse-Aware ZP Prediction

Discourse information have proven useful for predicting antecedents, which may occur in previous sentences (Zhao and Ng, 2007; Chen and Ng, 2013). Therefore, we further improve ZP prediction with discourse-level context, which is learned together with the joint model.

**Encoding Discourse-Level Context** Hierarchical structure networks are usually used for modelling discourse context on various natural language processing tasks such query suggestion (Sordoni et al., 2015), dialogue modeling (Serban et al., 2016) and MT (Wang et al., 2017). Therefore, we employ hierarchical encoder (Wang et al., 2017) to encoder discourse-

[1] We introduce " ⟨eos⟩" to cover the case that a pronoun is missing at the end of a sentence.

[2] We employ the pronoun vocabulary used in Wang et al. (2016), which contains 30 distinct Chinese pronouns.

level context for NMT. More specifically, we use the previous $K$ source sentences $\mathbf{X} = \{\mathbf{x}^{-K}, \ldots, \mathbf{x}^{-1}\}$ as the discourse information, which is summarized with a two-layer hierarchical encoder, as shown in Figure 2. For each sentence $\mathbf{x}^{-k}$, we employ a *word-level encoder* to summarize the representation of the whole sentence:

$$\mathbf{h}^{-k} = \text{ENCODER}_{word}(\mathbf{x}^{-k}) \quad (6)$$

After we can obtain all sentence-level representations $\mathbf{H}^X = \{\mathbf{h}^{-K}, \ldots, \mathbf{h}^{-1}\}$, we feed them into a *sentence-level encoder* to produce a vector that represents the discourse-level context:

$$\mathbf{C} = \text{ENCODER}_{sentence}(\mathbf{H}^X) \quad (7)$$

Here the summary $C$ consists of not only the dependencies between words, but also the relations between sentences. Following Voita et al. (2018), we share the parameters of word-level encoder $\text{ENCODER}_{word}$ with the encoder component in standard NMT model. Note that, $\text{ENCODER}_{word}$ and $\text{ENCODER}_{sentence}$ can be implemented as arbitrary networks, such as recurrent networks (Cho et al., 2014), convolutional networks (Gehring et al., 2017), or self-attention networks (Vaswani et al., 2017). In this study, we used recurrent networks to implement our ENCODER.

**Integrating Discourse into ZP Prediction** We directly feed the discourse-level context to the reconstructor to improve ZP prediction. Specifically, we combine the context vector and the reconstructor state:

$$\widehat{\mathbf{h}}_t^{rec} = f_c(\mathbf{h}_t^{rec}, \mathbf{C}) \quad (8)$$

Here $f_c(\cdot)$ is a function for combining reconstructor states and the context vector, which is a simple concatenation (CONCAT) in this work. The revised reconstructor state $\widehat{\mathbf{h}}_t^{rec}$ is then used in Equations (1) and (4).

## 4 Experiments

### 4.1 Setup

We conducted translation experiments on both Chinese⇒English and Japanese⇒English translation tasks, since Chinese and Japanese are pro-drop languages while English is not. For Chinese⇒English translation task, we used the data of auto-annotated ZPs (Wang et al., 2018a).[3] The training, validation, and test sets contain 2.15M, 1.09K, and 1.15K sentence pairs, respectively. In the training data, there are 27% of Chinese pronouns are ZPs, which poses difficulties for NMT models. For Japanese⇒English translation task, we respectively selected 1.03M, 1.02K, and 1.02K sentence pairs from Opensubtitle2016[4] as training, validation, and test sets (Tiedemann, 2012). We used case-insensitive 4-gram NIST BLEU (Papineni et al., 2002) as evaluation metrics, and *sign-test* (Collins et al., 2005) to test for statistical significance.

To make fair comparison with Wang et al. (2018a), we also implemented our approach on top of the RNN-based NMT model, which incorporates dropout (Hinton et al., 2012) on the output layer and improves the attention model by feeding the most recently generated word. For training the models, we limited the source and target vocabularies to the most frequent 30K words for Chinese⇒English and 20K for Japanese⇒English. Each model was trained on sentences of length up to a maximum of 20 words with early stopping. Mini-batches were shuffled during processing with a mini-batch size of 80. The dimension of word embedding was 620 and the hidden layer size was 1,000. We trained for 20 epochs using Adadelta (Zeiler, 2012), and selected the model that yielded best performances on validation sets. For training the proposed models, the hidden layer sizes of hierarchical model and reconstruction model are 1,000 and 2,000, respectively. We modeled previous three sentences as discourse-level context.[5]

### 4.2 Results on Chinese⇒English Task

Table 2 lists the performance of ZP translation and prediction on Chinese⇒English data.

The baseline (Row 1) is trained on the standard NMT model using the original parallel data ($\mathbf{x}$, $\mathbf{y}$). In addition, we implemented two comparative models (Row 2-3), which differ with respect to the training data used. The "+ ZP-Annotated Data" model was still trained on standard NMT model but using new training instances ($\hat{\mathbf{x}}$, $\mathbf{y}$) whose source-side sentences are auto-annotated with ZPs. The "+ Reconstruction" is the best model reported in Wang et al. (2018a), which employs two reconstructors to reconstruct the $\hat{\mathbf{x}}$ from

[3] https://github.com/longyuewangdcu/tvsub.

[4] http://www.opensubtitles.org.

[5] We followed Wang et al. (2017) and Tu et al. (2018) to use 3 previous sentences as discourse context.

| # | Model | Translation | | Prediction | | |
|---|---|---|---|---|---|---|
| | | *#Params* | *BLEU* | *P* | *R* | *F1* |
| 1 | Baseline | 86.7M | 31.80 | n/a | n/a | n/a |
| *External ZP Prediction (Wang et al., 2018a)* | | | | | | |
| 2 | + ZP-Annotated Data | +0M | 32.67 | 0.67 | 0.65 | 0.66 |
| 3 | + Reconstruction | +73.8M | 35.08 | | | |
| *This Work: Joint ZP Prediction and Translation* | | | | | | |
| 4 | Joint Model | +35.6M | 36.04$^\dagger$ | 0.72 | 0.68 | 0.70 |
| 5 | + Discourse-Level Context | +56.6M | **37.11**$^\dagger$ | 0.76 | 0.77 | **0.77** |

Table 2: Evaluation of ZP translation and prediction on the Chinese–English data. "#Params" represents the number of parameters used in different models. "†" indicates statistically significant difference ($p < 0.01$) from the best external ZP prediction model for translation performance. As seen, the proposed joint models improve performances in both ZP translation and prediction, over the external ZP prediction models.

hidden representations of encoder and decoder. At decoding time, ZPs can not be annotated by alignment method since target sentences are not available. Thus, source sentences are annotated by an external ZP prediction model, which is trained on monolingual training instances $\hat{\mathbf{x}}$. Finally, we evaluated two proposed models (Row 4-5) which are introduced in Section 3.1 and 3.2, respectively.

**Translation Quality** Benefiting from the explicitly annotated ZPs in the source language, the "+ ZP-Annotated Data" model (Row 2) outperforms the baseline system built on the original data where the pronouns are missing (*i.e.,* +0.87 BLEU point). This illustrates that explicitly recalling translation of ZPs at training time helps produce better translations. Furthermore, the "+ Reconstuction" approach (Row 3) respectively outperforms the baseline and "+ ZP-Annotated Data" models by +3.28 and +2.41 BLEU points, which indicates that explicitly handling ZPs with reconstruction model can better address ZP problems.

The proposed models consistently outperform other models in all cases, demonstrating the superiority of the joint learning of ZP prediction and translation. Specifically, the "Joint Model" (Row 4) significantly improves translation performance by +4.24 over baseline model. In addition, this joint approach also outperforms two comparative models "+ ZP-Annotated Data" and "+ Reconstruction" by +3.37 and +0.96 BLEU points, respectively. We attribute the improvement over external ZP prediction to: 1) releasing the reliance on external ZP prediction models can greatly alleviate error propagation problems; and 2) joint learning of ZP prediction and translation is able to guide the related parameters to learn better latent representations. Furthermore, introducing discourse-level context (Row 5) accumulatively improves translation performance, and significantly outperform the joint model by +1.07 BLEU points.

| Model | BLEU | △ |
|---|---|---|
| Baseline | 19.94 | – |
| External ZP Prediction | 20.86 | +0.92 |
| Joint Model | 21.39 | +1.45 |
| + Discourse-Level Context | **22.00** | +2.06 |

Table 3: Translation quality on Japanese–English data. As seen, the proposed models can also significantly improve translation performance, which shares the same trend with that on Chinese–English translation.

More parameters may capture more information, at the cost of posing difficulties to training. Wang et al. (2018a) leverage two separate reconstructors with hidden state size being 2000 and 1000 respectively. Accordingly, their models introduce a large number of parameters. In contrast, we set the hidden size of the reconstructor be 1000, which greatly reduce the newly introduced parameters (+35.6M vs. +73.8M). Modeling discourse-level context further introduces +21M new parameters, which is reasonable comparing with previous work. Our best model variation outperform that of external ZP prediction by over 2 BLEU points with less parameters (143.3M vs. 160.5M), showing that the improvements are attributed to the stronger modeling capacity rather than more parameters.

**ZP Prediction Accuracy** The joint model improves prediction accuracy as expected, which we

| Model | ZP-Annotated Input | | |
|---|---|---|---|
| | ✓ | × | ▽ |
| Baseline | 31.80 | | – |
| External ZP Predict. | 35.08 | 34.02 | -1.06 |
| Joint Model | 36.04 | 35.93 | -0.11 |
| + Discourse | **37.11** | **36.51** | -0.60 |

Table 4: Translation results when no ZP-annotated input is used in decoding by *removing the reconstructor component*. "▽" denotes the performance gap between whether using the annotated input ("✓") or not ("×").

| Model | BLEU | △ |
|---|---|---|
| Baseline | 31.80 | – |
| + Discourse⇒Decoder | 32.34 | +0.54 |
| Baseline + ZP-Anno. | 32.67 | |
| + Discourse⇒Decoder | 32.55 | -0.12 |
| Joint Model | 36.04 | – |
| + Discourse⇒Decoder | 34.66 | -1.38 |

Table 5: Translation results when transforming the contextual representation to decoder of different models. Incorporating discourse-level context does not always lead to improvement of translation performance.

attribute to the leverage of useful translation information. Incorporating the discourse-level context further improves ZP prediction, and the best performance is 11% higher than external ZP prediction model. These results confirm our claim that joint learning of ZP prediction and translation can benefit both components by allowing them to interact with each other.

### 4.3 Results on Japanese⇒English Task

Table 3 lists the results. We compare our models and the best external ZP prediction approach. As seen, our models also significantly improve translation performance, demonstrating the effectiveness and universality of the proposed approach.

This improvement on Japanese⇒English translation is lower than that on Chinese⇒English, showing that ZP prediction and translation are more challenging for Japanese. The reason may be two folds: 1) Japanese language has a larger number of pronoun variations borrowed from archaism, which leads to more difficulties in learning ZPs; 2) Japanese language is subject-object-verb (SOV) while English has subject-verb-object (SVO) structure, and this poses difficulties for ZP annotation via alignment method.

### 4.4 Analysis

We conducted extensive analyses on Chinese ⇒English to better understand our models in terms of the effect of external ZP annotation and different types of ZPs errors.

**Reliance on Externally ZP-Annotated Input** Some researchers may argue that previous approaches (Wang et al., 2018a) are also able to release the reliance of externally annotated input by removing the reconstructor component. Table 4 lists the results. Without ZP-annotated input in decoding, all approaches can still outperform the baseline model, by benefiting better intermediate representations that contain necessary ZP information. Compared with reconstruction-based models, however, removing the reconstruction components leads to decrease on translation quality. As seen, the BLEU score of best "External ZP prediction" model dramatically drops by -1.06 points, showing that this approach is heavily dependent on the results of external ZP annotations. The performances of proposed models only decrease by -0.1∼-0.6 BLEU point. It indicates that our models are compatible with the standard encoder-decoder-reconstructor framework, thus enjoy an additional benefit of re-scoring translation hypotheses in testing with reconstruction scores. All the results together prove the superiority of the proposed unified framework for ZP translation.

**Effect of Discourse-Level Context** Recent studies revealed that inter-sentential context can implicitly help to tackle anaphora resolution in NMT architecture (Jean et al., 2017b; Bawden et al., 2018; Voita et al., 2018). Some may argue that document-level architectures are strong enough to alleviate ZP problems for NMT. To answer this concern, we compared with "+ Discourse⇒Decoder" models, which transform the contextual representation to the decoder part of different models. In this way, the discourse-level context can benefit both the generation of translation and ZP prediction.

As shown in Table 5, directly incorporating inter-sentential context into standard NMT model (one of document-level NMT architectures) can improve translation quality by +0.54 BLEU point than baseline. However, this integration mechanism does not work well in "Baseline + ZP-Annotation" and our "Joint" models, which de-

| Model | Error | Sub. | Obj. | Dum. | All |
|---|---|---|---|---|---|
| BASE. | Total | 112 | 41 | 45 | 198 |
| EXTE. | Fixed | 50 | 34 | 33 | 117 |
| | New | 11 | 14 | 7 | 32 |
| | Total | 73 | 21 | 19 | 113 |
| JOIN. | Fixed | 61 | 35 | 37 | 133 |
| | New | 8 | 11 | 7 | 26 |
| | Total | 59 | 17 | 15 | 91 |
| +DIS. | Fixed | 70 | 39 | 38 | 147 |
| | New | 7 | 9 | 7 | 23 |
| | Total | **49** | **11** | **14** | **74** |

Table 6: Translation error statistics. The ZP types "Sub.", "Obj." and "Dum." denote errors caused by subjective, objective and dummy pronouns, respectively. The models "Base.", "Exte.", "Join." and "+Dis." denote "Baseline", "+ Reconstruction", "Joint Model" and "+ Discourse-Level context" models. **Bold** numbers denote the least errors in each category.

creasing by -0.12 and -1.38 BLEU points, respectively. One potential problem with this strategy is that the propagation path is longer: $\mathbf{C} \rightarrow \mathbf{h}^{dec} \rightarrow \mathbf{h}^{rec} \rightarrow \mathbf{zp}$, which may suffer from the vanishing effect. This also confirms our hypothesis that discourse-level context benefits ZP prediction more than ZP translation. Therefore, we incorporate the discourse-level context into reconstructor instead of the decoder.

**Manual Evaluation on Translation Errors** We finally investigate how the proposed approaches improve the translation by human evaluation. We randomly select 500 sentences from the test set. As shown in Table 6, we count how many translation errors caused by different types of ZPs (*i.e.,* "Subjective", "Objective" and "Dummy"[6]) are fixed ("Fixed") and newly generated ("New") by different models.

All the models can fix different amount of ZP problems in terms of completeness and correctness, which is consistent with the translation results reported in Table 2. This confirms that our improvement in terms of BLEU scores indeed comes from alleviating translation errors caused by ZPs. Among them, the proposed model "+DIS." performs best, which fixes 74% of the ZP errors, and only introduces 12% of new errors.

In addition, we found that subjective ZPs are more difficult to predict and translate since they usually occur in imperative sentences, and ZP prediction needs to understand intention of speakers. The "EXTE." model only fixes 45% of subjective ZP errors but made 10% new errors by predicting wrong ZPs. However, the proposed joint model works better, which fixes 54% error with only introducing 7% new errors. Predicting objective ZPs needs inter-sentential context, thus our "+DIS." model is able to fix more objective ZP errors (95% vs. 82%) by introducing less new errors (22% vs. 34%) than "EXTE.".

**Case Study** Table 7 shows two typical examples, of which pronouns are mistakenly translated by the strong baseline ("External ZP Prediction") model (Wang et al., 2018a) while fixed by our model and failed to be fix. In "Fixed Error" case, the dropped word "它 (*it*)" is an anaphoric ZP whose antecedent is the noun "电视 (*television*)" in previous sentence while the dropped word "你 (*you*)" is a non-anaphoric ZP that depends upon speaker or listener. As seen, our "JOIN." model performs better than the "EXTE." model because two ZP positions are syntactically recalled in the target side, showing that the joint approach have better capability of utilizing intra-sentential information for identifying ZPs. Besides, our "+DIS." model can semantically fix the error by predicting correct ZP words, demonstrating that inter-sentential context can aid to recovering such complex ZPs. However, as shown in "Non-Fixed Error" case, there are still some ZPs can not be precisely predicted due to the misunderstanding of intentions of utterances. Thus, exploiting dialogue focus for ZP translation is our future work (Rao et al., 2015).

## 5 Related Work

**ZP Prediction and Translation** ZP resolution is a challenging task which needs lexical, syntactic, discourse knowledge. Previous studies have been conducted to improves the performance of ZP resolution for different pro-drop languages (Kong and Zhou, 2010; Chen and Ng, 2013; Park et al., 2015; Yin et al., 2017). However, directly using results of external ZP resolution systems for translation task shows limited improvements (Chung and Gildea, 2010; Le Nagard and Koehn, 2010; Taira et al., 2012; Xiang et al., 2013), since such external systems are trained on small-scale data that is non-homologous to MT. To

[6]In pro-drop languages, it is used to fulfill the syntactical requirements without providing explicit meaning (e.g. "it").

| | |
|---|---|
| **Fixed Error** | |
| PRE. | 等 我 搬进 来, 能 买 台 电视 吗? |
| INP. | 当然 可以, 乔伊 不让 (你) 买 (它)? |
| REF. | Sure. Joey wouldn't let you buy it? |
| EXTE. | Of course. Sure, Joey won't get *it*? |
| JOIN. | Sure. Joey won't let *us* buy *one*? |
| +DIS. | Sure. Joey wouldn't let **you** buy **it**? |
| **Non-Fixed Error** | |
| PRE. | 我 和 露西 只是 要 搬 到 对门。 |
| INP. | 我们 一 分手 (我) 就 搬 回去。 |
| REF. | Once we broke up, I'll move back. |
| EXTE. | Once we broke up, *she*'ll move back. |
| JOIN. | Once we broke up, *we* moved back. |
| +DIS. | Once we broke up, *we*'ll move back. |

Table 7: Example translations where pronouns in brackets are dropped in original inputs ("INP.") but labeled by humans according to references ("REF.") and previous sentence ("PRE."). We italicize some *mistranslated* errors and highlight the **correct** ones in bold.

overcome the data-level gap, Wang et al. (2016) proposed an automatic approach of ZP annotation by utilizing an alignment matrix from a large parallel data. By using the translation-oriented ZP corpus, they exploited different approaches to alleviate ZP problems for translation models (Wang et al., 2016, 2018a,b). Note that Wang et al. (2018b) also explored to address the problem of error propagation by jointly predicting ZP words given ZP position information. However, this method still relies an external model that predicting ZP positions at decoding time. Instead, this work proposes a unified model without any additional ZP annotations in decoding, thus release reliance on external ZP prediction in practice.

**Discourse-Aware NMT** Recent years, context-aware architecture has been well studied for NMT (Wang et al., 2017; Jean et al., 2017a; Tu et al., 2018). Wang et al. (2017) proposed hierarchical recurrent neural networks to summarize inter-sentential context from previous sentences and then integrate it into a standard NMT model with difference strategies. Jean et al. (2017a) introduced an additional set of an encoder and attention to encode and select part of the previous source sentence for generating each target word. Besides, Tu et al. (2018) proposed to augment NMT models with a cache-like memory network, which stores the translation history in terms of bilingual hidden representations at decoding steps of previous sentences. They also evaluated the above three models on different domains of data, showing that the hierarchical encoder performs comparable with the multi-attention model. More recently, some researchers began to investigate the effects of context-aware NMT on cross-lingual pronoun prediction (Jean et al., 2017b; Bawden et al., 2018; Voita et al., 2018). They mainly exploited general anaphora in non-pro-drop languages such as English⇒Russian.

## 6 Conclusion

In this work, we proposed a unified model to learn jointly predict and translate ZPs by leveraging multi-task learning. We also employed hierarchical neural networks to exploit discourse-level information for better ZP prediction. Experimental results on both Chinese⇒English and Japanese⇒English data show that the two proposed approaches accumulatively improve both the translation performance and ZP prediction accuracy. Our models also outperform the existing ZP translation models in previous work, and achieve a new state-of-the-art on the widely-used subtitle corpus. Manual evaluation confirms that the performance improvement comes from the alleviation of translation errors, which are mainly caused by subjective, objective as well as discourse-aware ZPs.

There are two potential extensions to our work. First, we will evaluate our method on other implication phenomena (or called unaligned words (Takeno et al., 2017)) such as tenses and article words for NMT. Second, we will investigate the impact of different context-aware models on ZP translation, including multi-attention (Jean et al., 2017b) and context-aware Transformer(Voita et al., 2018).

## References

Rachel Bawden, Rico Sennrich, Alexandra Birch, and Barry Haddow. 2018. Evaluating discourse phenomena in neural machine translation. In *NAACL*.

Chen Chen and Vincent Ng. 2013. Chinese zero pronoun resolution: Some recent advances. In *EMNLP*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*.

Tagyoung Chung and Daniel Gildea. 2010. Effects of empty categories on machine translation. In *EMNLP*.

Michael Collins, Philipp Koehn, and Ivona Kucerova. 2005. Clause restructuring for statistical machine translation. In *ACL*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *ICML*.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Sebastien Jean, Stanislas Lauly, Orhan Firat, and Kyunghyun Cho. 2017a. Does neural machine translation benefit from larger context? *arXiv preprint arXiv:1704.05135*.

Sebastien Jean, Stanislas Lauly, Orhan Firat, and Kyunghyun Cho. 2017b. Neural machine translation for cross-lingual pronoun prediction. In *Workshop on Discourse in Machine Translation*.

Fang Kong and Guodong Zhou. 2010. A tree kernel-based unified framework for chinese zero anaphora resolution. In *EMNLP*.

Ronan Le Nagard and Philipp Koehn. 2010. Aiding pronoun translation with co-reference resolution. In *WMT-MetricsMATR*.

Charles N Li and SA Thomson. 1979. Third-person pronouns and zero-anaphora in chinese discourse in discourse and syntax. *Syntax and Semantics*, 12.

Yang Liu and Qun Liu. 2010. Joint parsing and translation. In *COLING*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *ACL*.

Arum Park, Seunghee Lim, and Munpyo Hong. 2015. Zero object resolution in korean. In *PACLIC*.

Sudha Rao, Allyson Ettinger, Hal Daumé III, and Philip Resnik. 2015. Dialogue focus tracking for zero pronoun resolution. In *NAACL*.

Iulian V. Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *CIKM*.

Hirotoshi Taira, Katsuhito Sudoh, and Masaaki Nagata. 2012. Zero pronoun resolution can improve the quality of J-E translation. In *Workshop on Syntax, Semantics and Structure in Statistical Translation*.

Shunsuke Takeno, Masaaki Nagata, and Kazuhide Yamamoto. 2017. Controlling target features in neural machine translation via prefix constraints. In *The 4th Workshop on Asian Translation*.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in opus. In *LREC*.

Zhaopeng Tu, Yang Liu, Lifeng Shang, Xiaohua Liu, and Hang Li. 2017. Neural machine translation with reconstruction. In *AAAI*.

Zhaopeng Tu, Yang Liu, Shuming Shi, and Tong Zhang. 2018. Learning to remember translation history with a continuous cache. *TACL*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Elena Voita, Pavel Serdyukov, Rico Sennrich, and Ivan Titov. 2018. Context-aware neural machine translation learns anaphora resolution. In *ACL*.

Longyue Wang, Zhaopeng Tu, Shuming Shi, Tong Zhang, Yvette Graham, and Qun Liu. 2018a. Translating pro-drop languages with reconstruction models. In *AAAI*.

Longyue Wang, Zhaopeng Tu, Andy Way, and Qun Liu. 2017. Exploiting cross-sentence context for neural machine translation. In *EMNLP*.

Longyue Wang, Zhaopeng Tu, Andy Way, and Qun Liu. 2018b. Learning to jointly translate and predict dropped pronouns with a shared reconstruction mechanism. In *EMNLP*.

Longyue Wang, Zhaopeng Tu, Xiaojun Zhang, Hang Li, Andy Way, and Qun Liu. 2016. A novel approach for dropped pronoun translation. In *NAACL*.

Bing Xiang, Xiaoqiang Luo, and Bowen Zhou. 2013. Enlisting the ghost: Modeling empty categories for machine translation. In *ACL*.

Xinyan Xiao, Yang Liu, Young-Sook Hwang, Qun Liu, and Shouxun Lin. 2010. Joint tokenization and translation. In *COLING*.

Qingyu Yin, Yu Zhang, Weinan Zhang, and Ting Liu. 2017. Chinese zero pronoun resolution with deep memory network. In *EMNLP*.

Matthew D Zeiler. 2012. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Shanheng Zhao and Hwee Tou Ng. 2007. Identification and resolution of chinese zero pronouns: A machine learning approach. In *EMNLP*.

# Dynamic Past and Future for Neural Machine Translation

**Zaixiang Zheng**
Nanjing University
zhengzx@smail.nju.edu.cn

**Shujian Huang**
Nanjing University
huangsj@nju.edu.cn

**Zhaopeng Tu**
Tencent AI Lab
zptu@tencent.com

**Xin-Yu Dai**
Nanjing University
dxy@nju.edu.cn

**Jiajun Chen**
Nanjing University
chenjj@nju.edu.cn

## Abstract

Previous studies have shown that neural machine translation (NMT) models can benefit from explicitly modeling translated (PAST) and untranslated (FUTURE) source contents as recurrent states (Zheng et al., 2018). However, this less interpretable recurrent process hinders its power to model the dynamic updating of PAST and FUTURE contents during decoding. In this paper, we propose to model the *dynamic principles* by explicitly separating source words into groups of translated and untranslated contents through parts-to-wholes assignment. The assignment is learned through a novel variant of routing-by-agreement mechanism (Sabour et al., 2017), namely *Guided Dynamic Routing*, where the translating status at each decoding step *guides* the routing process to assign each source word to its associated group (i.e., translated or untranslated content) represented by a capsule, enabling translation to be made from holistic context. Experiments show that our approach achieves substantial improvements over both RNMT and Transformer by producing more adequate translations. Extensive analysis demonstrates that our method is highly interpretable, which is able to recognize the translated and untranslated contents as expected.[1]

## 1 Introduction

Neural machine translation (NMT) generally adopts an *attentive encoder-decoder* framework (Sutskever et al., 2014; Vaswani et al., 2017), where the encoder maps a source sentence into a sequence of contextual representations (*source contents*), and the decoder generates a target sentence word-by-word based on part of the source content assigned by an attention model (Bahdanau et al., 2015). Like human translators, NMT systems should have the ability to know the relevant source-side context for the current word (PRESENT), as well as recognize what parts in the source contents have been translated (PAST) and what parts have not (FUTURE), at each decoding step. Accordingly, the PAST, PRESENT and FUTURE are three *dynamically* changing states during the whole translation process.

Previous studies have shown that NMT models are likely to face the illness of inadequate translation (Kong et al., 2019), which is usually embodied in over- and under-translation problems (Tu et al., 2016, 2017). This issue may be attributed to the poor ability of NMT of recognizing the dynamic translated and untranslated contents. To remedy this, Zheng et al. (2018) first demonstrate that explicitly tracking PAST and FUTURE contents helps NMT models alleviate this issue and generate better translation. In their work, the running PAST and FUTURE contents are modeled as recurrent states. However, the recurrent process is still non-trivial to determine which parts of the source words are the PAST and which are the FUTURE, and to what extent the recurrent states represent them respectively, this less interpretable nature is probably not the best way to model and exploit the dynamic PAST and FUTURE.

We argue that an explicit separation of the source words into two groups, representing PAST and FUTURE respectively (Figure 1), could be more beneficial not only for easy and direct recognition of the translated and untranslated source contents, but also for better interpretation of model's behavior of the recognition. We formulate the explicit separation as a procedure of parts-to-wholes assignment: the representation of each source words (parts) should be assigned to its associated group of either PAST or FUTURE (wholes).

In this paper, we implement this idea using Cap-

[1]Codes are released at https://github.com/zhengzx-nlp/dynamic-nmt.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 931–941, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

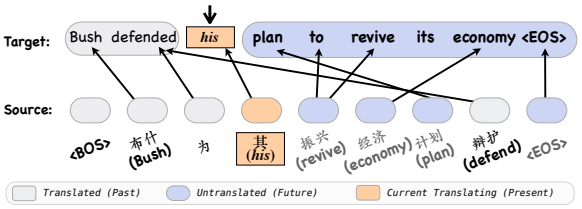


Figure 1: An example of separation of PAST and FUTURE in machine translation. When generating the current translation "his", the source tokens "⟨BOS⟩", "布什(Bush)" and phrase "为...辩护(defend)" are the translated contents (PAST), while the remaining tokens are untranslated contents (FUTURE).

sule Network (Hinton et al., 2011) with routing-by-agreement mechanism (Sabour et al., 2017), which has demonstrated its appealing strength of solving the problem of parts-to-wholes assignment (Hinton et al., 2018; Gong et al., 2018; Dou et al., 2019; Li et al., 2019), to model the separation of the PAST and FUTURE:

1. We first cast the PAST and FUTURE source contents as two groups of capsules.
2. We then design a novel variant of the routing-by-agreement mechanism, called *Guided Dynamic Routing* (GDR), which is *guided* by the current translating status at each decoding step to assign each source word to its associated capsules by assignment probabilities for several routing iterations.
3. Finally, the PAST and FUTURE capsules accumulate their expected contents from representations, and are fed into the decoder to provide a time-dependent holistic view of context to decide the prediction.

In addition, two auxiliary learning signals facilitate GDR's acquiring of our expected functionality, other than implicit learning within the training process of the NMT model.

We conducted extensive experiments and analysis to verify the effectiveness of our proposed model. Experiments on Chinese-to-English, English-to-German, and English-to-Romanian show consistent and substantial improvements over the Transformer (Vaswani et al., 2017) or RNMT (Bahdanau et al., 2015). Visualized evidence proves that our approach does acquire the expected ability to separate the source words into PAST and FUTURE, which is highly interpretable. We also observe that our model does alleviate the inadequate translation problem: Human subjective evaluation reveals that our model produces more adequate and high-quality translations than Transformer. Length analysis regarding source sentences shows that our model generates not only longer but also better translations.

## 2 Neural Machine Translation

Neural models for sequence-to-sequence tasks such as machine translation often adopt an *encoder-decoder* framework. Given a source sentence $\mathbf{x} = \langle x_1, \ldots, x_I \rangle$, a NMT model learns to predict a target sentence $\mathbf{y} = \langle y_1, \ldots, y_T \rangle$ by maximizing the conditional probabilities $p(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^{T} p(y_t|y_{<t}, \mathbf{x})$. Specifically, an encoder first maps the source sentence into a sequence of encoded representations:

$$\mathbf{h} = \langle \boldsymbol{h}_1, \ldots, \boldsymbol{h}_I \rangle = f_e(\mathbf{x}), \quad (1)$$

where $f_e$ is the encoder's transformation function. Given the encoded representations of the source words, a decoder generates the sequence of target words $\mathbf{y}$ autoregressively:

$$\boldsymbol{z}_t = f_d(y_{<t}, \boldsymbol{a}_t), \quad (2)$$

$$p(y_t|y_{<t}, \mathbf{x}) = \mathrm{softmax}(E(y_t)^\top \boldsymbol{z}_t), \quad (3)$$

where $E(y_t)$ is the embedding of $y_t$. The current word is predicted based on the decoder state $\boldsymbol{z}_t$. $f_d$ is the transformation function of decoder, which determines $\boldsymbol{z}_t$ based on the target translation trajectory $y_{<t}$, and the lexical-level source content $\boldsymbol{a}_t$ that is most relevant to PRESENT translation by an attention model (Bahdanau et al., 2015). Ideally, with all the source encoded representations in the encoder, NMT models should be able to update translated and untranslated source contents and keep them in mind. However, most of existing NMT models lack an explicit functionality to maintain the translated and untranslated contents, failing to distinguish the source words being of either PAST or FUTURE (Zheng et al., 2018), which is likely to suffer from severe inadequate translation problem (Tu et al., 2016; Kong et al., 2019).

## 3 Approach

**Motivation** Our intuition arises straightforwardly: if we could tell the translated and untranslated source contents apart by directly separating the source words into PAST and FUTURE categories at each decoding step, the PRESENT translation could benefit from the dynamically holistic context (i.e., PAST+ PRESENT+ FUTURE). To this

purpose, we should design a mechanism by which each word could be recognized and assigned to a distinct category, i.e., PAST or FUTURE contents, subject to the translation status at present. This procedure can be seen as a parts-to-wholes assignment, in which the encoder hidden states of the source words (parts) are supposed to be assigned to either PAST or FUTURE (wholes).

Capsule network (Hinton et al., 2011) has shown its capability of solving the problem of assigning parts to wholes (Sabour et al., 2017). A capsule is a vector of neurons which represents different properties of the same entity from the input (Sabour et al., 2017). The functionality relies on a fast iterative process called routing-by-agreement, whose basic idea is to iteratively refine the proportion of how much a part should be assigned to a whole, based on the agreement between the part and the whole (Dou et al., 2019). Therefore, it is appealing to investigate if this mechanism could be employed for our intuition.

### 3.1 Guided Dynamic Routing (GDR)

Dynamic routing (Sabour et al., 2017) is an implementation of routing-by-agreement, where it runs intrinsically without any external guidance. However, what we expect is a mechanism driven by the decoding status at present. Here we propose a variant of dynamic routing mechanism called *Guided Dynamic Routing* (GDR), where the routing process is *guided* by the translating information at each decoding step (Figure 2).

Formally, we cast the source encoded representations $\mathbf{h}$ of $I$ source words to be input capsules, while we denote $\mathbf{\Omega}$ as output capsules, which consist of $J$ entries. Initially, we assume that $J/2$ of them ($\mathbf{\Omega}^P$) represent the PAST contents, and the rest $J/2$ capsules ($\mathbf{\Omega}^F$) represent the FUTURE:

$$\mathbf{\Omega}^P = \langle \Omega_1^P, \cdots, \Omega_{J/2}^P \rangle, \quad \mathbf{\Omega}^F = \langle \Omega_1^F, \cdots, \Omega_{J/2}^F \rangle.$$

where each capsule is represented by a $d_c$-dimension vector. We assemble these PAST and FUTURE capsules together, which are expected to competing for source information, i.e., we now have $\mathbf{\Omega} = \mathbf{\Omega}^P \cup \mathbf{\Omega}^F$. We will describe how to teach these capsules to retrieve their relevant parts from source contents in the Section 3.3. ***Note*** that we employ GDR at every decoding step $t$ to obtain the time-dependent PAST and FUTURE and omit the subscript $t$ for simplicity.

In the dynamic routing process, each vector output of capsule $j$ is calculated with a non-linear

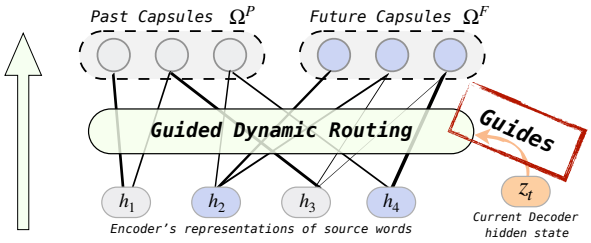


Figure 2: Illustration of the Guided Dynamic Routing.

*squashing* function (Sabour et al., 2017):

$$\Omega_j = \frac{\|\boldsymbol{s}_j\|^2}{1+\|\boldsymbol{s}_j\|^2}\frac{\boldsymbol{s}_j}{\|\boldsymbol{s}_j\|}, \quad \boldsymbol{s}_j = \sum_i^I c_{ij}\boldsymbol{v}_{ij}, \quad (4)$$

where $\boldsymbol{s}_j$ is the accumulated input of capsule $\Omega_j$, which is a weighted sum over all *vote vectors* $\boldsymbol{v}_{ij}$. $\boldsymbol{v}_{ij}$ is transformed from the input capsule $\boldsymbol{h}_i$:

$$\boldsymbol{v}_{ij} = \boldsymbol{W}_j \boldsymbol{h}_i, \quad (5)$$

where $\boldsymbol{W}_j \in \mathbb{R}^{d \times d_c}$ is a trainable matrix for $j$-th output capsule[2]. $c_{ij}$ is the assignment probability (i.e. the agreement) that is determined by the iterative dynamic routing. The assignment probabilities $c_{i\cdot}$ associated with each input capsule $\boldsymbol{h}_i$ sum to 1: $\sum_j c_{ij} = 1$, and are computed by:

$$c_{ij} = \mathrm{softmax}(b_{ij}), \quad (6)$$

where routing logit $b_{ij}$ is initialized as all 0s, which measures the degree that $\boldsymbol{h}_i$ should be sent to $\Omega_j$. The initial assignment probabilities are then iteratively updated by measuring the agreement between the vote vector $\boldsymbol{v}_{ij}$ and capsule $\Omega_j$ by an MLP, considering the current decoding state $\boldsymbol{z}_t$:

$$b_{ij} \leftarrow b_{ij} + \boldsymbol{w}^\top \tanh(\boldsymbol{W}_b[\boldsymbol{z}_t; \boldsymbol{v}_{ij}; \Omega_j]), \quad (7)$$

where $\boldsymbol{W}_b \in \mathbb{R}^{d+d_c*2}$ and $\boldsymbol{w} \in \mathbb{R}^{d_c}$ are learnable parameters. Instead of using simple scalar product, i.e., $b_{ij} = \boldsymbol{v}_{ij}^\top \Omega_j$ (Sabour et al., 2017), which could not consider the current decoding state as a condition signal, we resort to the MLP to take $\boldsymbol{z}_i$ into account inspired by MLP-based attention mechanism (Bahdanau et al., 2015; Luong et al., 2015). That is why we call it "guided" dynamic routing.

[2]Note that unlike Sabour et al. (2017), where each pair of input capsule $i$ and output capsule $j$ has a distinct transformation matrix $\boldsymbol{W}_{ij}$ as their numbers are predefined ($I \times J$ transformation matrices in total), here we share the transformation matrix $\boldsymbol{W}_j$ of output capsule $j$ among all the input capsules due to the varied amount of the source words. So there are $J$ transformation matrices in our model.

**Algorithm 1** Guided Dynamic Routing (GDR)

**Input:** Encoder hidden state $\mathbf{h}$, current decoding hidden state $\boldsymbol{z}_t$, and number of routing iterations $r$.
**Output:** PAST, FUTURE, and redundant capsules.
**procedure:** GDR($\mathbf{h}$, $\boldsymbol{z}_t$, $r$)
$\forall i \in \mathbf{h}, j \in \boldsymbol{\Omega} : b_{ij} \leftarrow 0, \boldsymbol{v}_{ij} \leftarrow \boldsymbol{W}_j \boldsymbol{h}_i$ ▷ *Initializing routing logits, and vote vectors.*
**for** $r$ iterations **do**
  $\forall i \in \mathbf{h}, j \in \boldsymbol{\Omega}$: Compute assign. probs. $c_{ij}$ by Eq. 6
  $\forall j \in \boldsymbol{\Omega}$ : Compute capsules $\Omega_j$ by Eq. 4
  $\forall i \in \mathbf{h}, j \in \boldsymbol{\Omega}$ : Update routing logits $b_{ij}$ by Eq. 7
**end for**
$[\boldsymbol{\Omega}^P; \boldsymbol{\Omega}^F; \boldsymbol{\Omega}^R] = \boldsymbol{\Omega}$ ▷ *Return past, future, and redundant capsules*
**return** $\boldsymbol{\Omega}^P, \boldsymbol{\Omega}^F, \boldsymbol{\Omega}^R$

Now with the awareness of the current decoding status, the hidden state (input capsule) of a source word prefers to send its representation to the output capsules, which have large routing agreements associated with the input capsule. After a few rounds of iterations, the output capsules are able to ignore all but the most relevant information from the source hidden states, representing a distinct aspect of either PAST or FUTURE.

**Redundant Capsules** In some cases, some parts of the source sentence may belong to neither past contents nor future contents. For example, function words in English (e.g., "the") could not find its counterpart translation in Chinese. Therefore, we add additional Redundant Capsules $\boldsymbol{\Omega}^R$ (also known as "orphan capsules" in Sabour et al. (2017)), which are expected to receive higher routing assignment probabilities when a source word should not belong to either PAST or FUTURE.

We show the algorithm of our guided dynamic routing in Algorithm 1.

### 3.2 Integrating into NMT

The proposed GDR can be applied on the top of any sequence-to-sequence architecture, which does not require any specific modification. Let us take a Transformer-fashion architecture as example (Figure 3). Given a sentence $\mathbf{x} = \langle x_1, \ldots, x_I \rangle$, the encoder leverages $N$ stacked identical layers to map the sentence into contextual representations:

$$\mathbf{h}^{(l)} = \text{EncoderLayer}(\mathbf{h}^{(l-1)}),$$

where the superscript $l$ indicates layer depth. Based on the encoded source representations $\mathbf{h}^N$, a decoder generates translation word by word. The

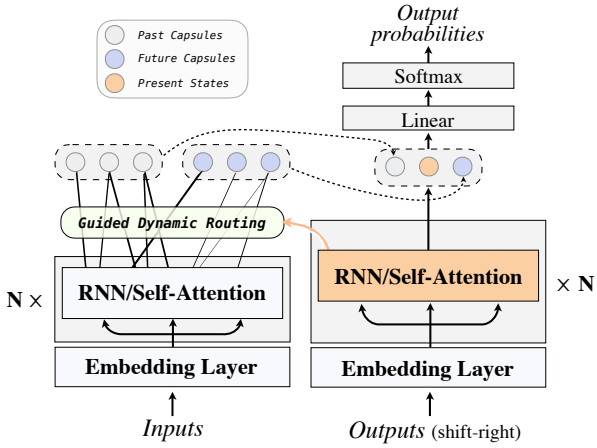


Figure 3: Illustration of our architecture.

decoder also has $N$ stacked identical layers:

$$\mathbf{z}^{(l)} = \text{DecoderLayer}(\mathbf{z}^{(l-1)}, \mathbf{a}^{(l)}),$$
$$\mathbf{a}^{(l)} = \text{Attention}(\mathbf{z}^{(l-1)}, \mathbf{h}^{(N)}),$$

where $\mathbf{a}^{(l)}$ is the lexical-level source context assigned by an attention mechanism between current decoder layer and the last encoder layer. Given the hidden states of the last decoder layer $\mathbf{z}^{(N)}$, we perform our proposed guided dynamic routing (GDR) mechanism to compute the PAST and FUTURE contents from the source side and obtain the holistic context of each decoding step:

$$\boldsymbol{\Omega}^P, \boldsymbol{\Omega}^F, \boldsymbol{\Omega}^R = \text{GDR}(\mathbf{z}^{(N)}, \mathbf{h}^{(N)}),$$
$$\mathbf{o} = \text{FeedForward}(\mathbf{z}^{(N)}, \boldsymbol{\Omega}^P, \boldsymbol{\Omega}^F) + \mathbf{z}^{(N)},$$

where $\mathbf{o} = \langle \boldsymbol{o}_1, \cdots, \boldsymbol{o}_T \rangle$ is the sequence of the holistic context of each decoding step. Based on the holistic context, the output probabilities are computed as:

$$p(y_t | y_{<t}, \mathbf{x}) = \text{softmax}(g(\boldsymbol{o}_t)).$$

The NMT model is now able to employ the dynamic holistic context for better generation.

### 3.3 Learning PAST and FUTURE as Expected Auxiliary Guided Losses

To ensure that the dynamic routing process runs as expected, we introduce the following auxiliary guided signals to assist the learning process.

**Bag-of-Word Constraint** Weng et al. (2017) propose a multitasking scheme to boost NMT by predicting the bag-of-words of target sentence using the Word Predictions approach. Inspired by

this work, we introduce a BOW constraint to encourage the PAST and FUTURE capsules to be predictive of the preceding and subsequent bag-of-words regarding each decoding step respectively:

$$\mathcal{L}_{\text{BOW}} = \frac{1}{T}\sum_{t=0}^{T}\big(-\log p_{\text{PRE}}(y_{\leq t}|\mathbf{\Omega}_t^P) - \log p_{\text{SUB}}(y_{\geq t}|\mathbf{\Omega}_t^F)\big),$$

where $p_{pre}(y_{\leq t}|\mathbf{\Omega}_t^P)$ and $p_{sub}(y_{\geq t}|\mathbf{\Omega}_t^F)$ are the predicted probabilities of the preceding bag-of-words and subsequent words at decoding step $t$, respectively. For instance, the probabilities of the preceding bag-of-words are computed by:

$$p_{\text{PRE}}(y_{<t}|\mathbf{\Omega}_t^P) = \prod_{\tau\in[1,t]} p_{\text{PRE}}(y_\tau|\mathbf{\Omega}_t^P) \propto \prod_{\tau\in[1,t]} \exp(\boldsymbol{E}(y_\tau)^\top \boldsymbol{W}_{\text{BOW}}^P \mathbf{\Omega}_t^P).$$

The computation of $p_{\text{SUB}}(y_{\geq t}|\mathbf{\Omega}_t^F)$ is similar. By applying the BOW constraint, the PAST and FUTURE capsules can learn to reflect the target-side past and future bag-of-words information.

**Bilingual Content Agreement** Intuitively, the translated source contents should be semantically equivalent to the translated target contents, and so do untranslated contents. Thus, a natural idea is to encourage the source PAST contents, modeled by the PAST capsule to be close to the target PAST representation at each decoding step, and the same for the FUTURE. Hence, we introduce a Bilingual Content Agreement (BCA) to require the bilingual semantic-equivalent contents to be predictive to each other by Minimum Square Estimation (MSE) loss:

$$\mathcal{L}_{\text{BCA}} = \frac{1}{T}\sum_{t=1}^{T}\|\mathbf{\Omega}_t^P - \boldsymbol{W}_{\text{BCA}}^P(\frac{1}{t}\sum_{\tau=1}^{t}\boldsymbol{z}_\tau)\|^2 + \|\mathbf{\Omega}_t^F - \boldsymbol{W}_{\text{BCA}}^F(\frac{1}{T-t+1}\sum_{\tau=t}^{T}\boldsymbol{z}_\tau)\|^2,$$

where the target-side past information is represented by the averaged results of the decoder hidden states of all preceding words, while the average of subsequent decoder hidden states represents the target-side future information.

**Training**

Given the dataset of parallel training examples $\{\langle \mathbf{x}^{(m)}, \mathbf{y}^{(m)}\rangle\}_{m=1}^{M}$, the model parameters are trained by minimizing the loss $\mathcal{L}(\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is the set of all the parameter of the proposed model:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{M}\sum_{m=1}^{M}\big(-\log p(\mathbf{y}^{(m)}|\mathbf{x}^{(m)}) + \lambda_1\cdot\mathcal{L}_{\text{BOW}} + \lambda_2\cdot\mathcal{L}_{\text{BCA}}\big),$$

where $\lambda_1$ and $\lambda_2$ are hyper-parameters.

## 4 Experiment

We mainly evaluated our approaches on the widely used NIST Chinese-to-English (Zh-En) translation task. We also conducted translation experiments on WMT14 English-to-German (En-De) and WMT16 English-to-Romanian (En-Ro):

1. NIST Zh-En. The training data consists of 1.09 million sentence pairs extracted from LDC[3]. We used NIST MT03 as the development set (Dev); MT04, MT05, MT06 as the test sets.

2. WMT14 En-De. The training data consists of 4.5 million sentence pairs from WMT14 news translation task. We used newstest2013 as the development set and newstest2014 as the test set.

3. WMT16 En-Ro. The training data consists of 0.6 million sentence pairs from WMT16 news translation task. We used newstest2015 as the development set and newstest2016 as the test set.

We used `transformer_base` configuration (Vaswani et al., 2017) for all the models. We run the dynamic routing for $r$=3 iterations. The dimension $d_c$ of a single capsule is 256. Either PAST or FUTURE content was represented by $\frac{J}{2} = 2$ capsules. Our proposed models were trained on the top of pre-trained baseline models[4]. $\lambda_1$ and $\lambda_2$ in training objective were set to 1. In Appendix, we provide details for the training settings.

### 4.1 NIST Zh-En Translation

We list the results of our experiments on NIST Zh-En task in Table 1 concerning two different architectures, i.e., Transformer and RNMT. As we can see, all of our models substantially outperform the baselines in terms of averaged BLEU score of all the test sets. Among them, our best model achieves 45.65 BLEU based on Transformer architecture. We also find that redundant capsules are helpful while discarding them leads to -0.35 BLEU degradation (45.65 vs 45.30).

[3]The corpora includes LDC2002E18, LDC2003E07, LDC2003E14, Hansards portion of LDC2004T07, LDC2004T08 and LDC2005T06

[4]Pre-training is only for efficiency purpose. Our approach could also learn from scratch.

| Model | $\|\boldsymbol{\theta}\|$ | $v_{\text{train}}$ | $v_{\text{test}}$ | Dev | MT04 | MT05 | MT06 | Tests Avg. |
|---|---|---|---|---|---|---|---|---|
| Transformer | 66.1m | 1.00× | 1.00× | 45.83 | 46.66 | 43.36 | 42.17 | 44.06 |
| GDR | 68.9m | 0.77× | 0.94× | 46.50 | 47.03 | 45.50 | 42.21 | 44.91 (+0.75) |
| + $\mathcal{L}_{\text{BOW}}$ | 69.2m | 0.70× | 0.94× | 47.12 | 48.09 | 45.98 | 42.68 | 45.58 (+1.42) |
| + $\mathcal{L}_{\text{BCA}}$ | 69.4m | 0.75× | 0.94× | 46.86 | 48.00 | 45.67 | 42.62 | 45.43 (+1.37) |
| + $\mathcal{L}_{\text{BOW}}$ + $\mathcal{L}_{\text{BCA}}$ [OURS] | 69.7m | 0.67× | 0.94× | **47.52** | **48.13** | **45.98** | **42.85** | **45.65 (+1.59)** |
| OURS - *redundant capsules* | 68.7m | 0.69× | 0.94× | 47.20 | 47.82 | 45.59 | 42.51 | 45.30 (+1.24) |
| RNMT | 50.2m | 1.00× | 1.00× | 35.98 | 37.85 | 36.12 | 35.86 | 36.61 |
| +PFRNN (Zheng et al., 2018) | N/A | 0.54× | 0.74× | 37.90 | 40.37 | 36.75 | 36.44 | 37.85 (+1.24) |
| +AOL (Kong et al., 2019) | N/A | 0.57× | 1.00× | 37.61 | 40.05 | 37.58 | 36.87 | 38.16 (+1.55) |
| OURS | 53.9m | 0.62× | 0.90× | **38.10** | **40.87** | **37.50** | **37.00** | **38.45 (+1.84)** |

Table 1: Experiment ressuts on NIST Zh-En task, including number of parameters ($|\boldsymbol{\theta}|$, excluding word embeddings), training/testing speeds ($v_{\text{train}}$/$v_{\text{test}}$), and translation results in case-insensitive BLEU.

**Architectures** Our approach shows consistent effects on both Transformer and RNMT architectures. In comparison to the Transformer baseline, our model achieves at most +1.59 BLEU improvement (45.65 v.s 44.06), while +1.84 BLEU improvement over RNMT baselines (38.45 v.s 36.61). These results indicate the compatibility of our approach to different architectures.

**Auxiliary Guided Losses** Both the auxiliary guided losses help our model for better learning. The BOW constraint leads to a +0.67 improvement compared to the vanilla GDR, while the benefit is +0.62 for BCA. Combination of both gains the most margins (+0.84), which means that they can supplement each other.

**Efficiency** To examine the efficiency of the proposed approach, we also list the relative speed of both training and testing. Our approach is 0.67× slower than the Transformer baseline in training phase, however, it does not hurt the speed of testing too much (0.94×). It is because the most extra computation in training phrase is related to the softmax operations of BOW losses, the degradation of the testing efficiency is moderate.

**Comparison to Other Work** On the experiments on RNMT architecture, we list two related works. Zheng et al. (2018) use extra PAST and FUTURE RNNs to capture translated and untranslated contents recurrently (PFRNN), while Kong et al. (2019) directly leverage translation adequacy as learning reward by their proposed Adequacy-oriented Learning (AOL). Compared to them, our model also enjoys competitive improvements due to the explicit separation of source contents. In addition, PFRNN is non-trivial to adapt to Transformer, because it requires a recurrent process which fails to be compatible with parallel training of Transformer, scarifying Transformer's efficiency advantage.

| Model | En-De | En-Ro |
|---|---|---|
| GNMT+RL (Wu et al., 2016) | 24.6 | N/A |
| ConvS2S (Gehring et al., 2017) | 25.2 | 29.88 |
| Transformer (Vaswani et al., 2017) | 27.3 | N/A |
| +AOL (Kong et al., 2019) | 28.01 | N/A |
| Transformer (Gu et al., 2017) | N/A | 31.91 |
| Transformer | 27.14 | 32.10 |
| OURS | **28.10** | **32.96** |

Table 2: Case-sensitive BLEU on WMT14 En-De and WMT16 En-Ro tasks.

### 4.2 WMT En-De and En-Ro Translation

We evaluated our approach on WMT14 En-De and WMT16 En-Ro tasks. As shown in Table 2, our reproduced Transformer baseline systems are close to the state-of-the-art results in previous work, which guarantee the comparability of our experiments. The results show a consistent trend of improvements as NIST Zh-En task on WMT14 En-De (+0.96 BLEU) and WMT16 En-Ro (+0.86 BLEU) benchmarks. We also list the results of other published research for comparison, where our model outperforms the previous results in both language pairs. Note that our approach also surpasses Kong et al. (2019) on WMT14 En-De task. These experiments demonstrate the effectiveness of our approach across different language pairs.

### 4.3 Analysis and Discussion

***Our model learns* PAST *and* FUTURE.** We visualize the assignment probabilities in the last routing iteration (Figure 4). Interestingly, there is a clear trend that the assignment probabilities to the PAST capsules gradually raise up, while

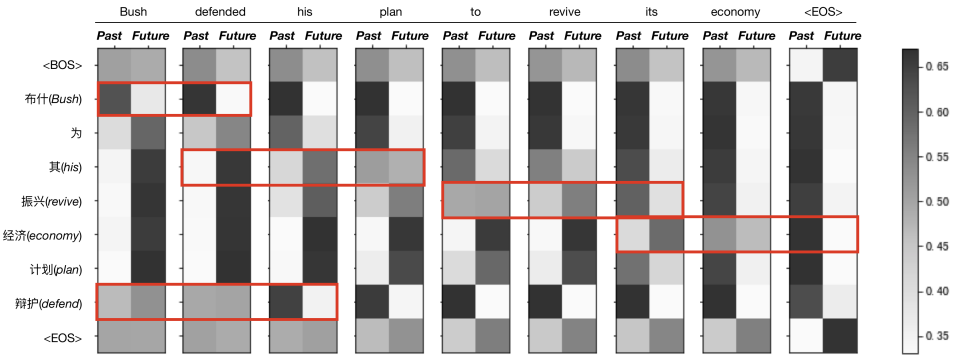


Figure 4: Visualization of the assignment probabilities of iterative routing. Each sub-heatmap is associated with a target word, where the left column is the probabilities of each source words routing to the PAST capsules, and the right one is to the FUTUREExamples in the red frame indicate the changes before and after the generation of the central word. We omit the assignment probabilities associated with the redundant capsules for simplicity. For instance, after the target word "defended" was generated, the assignment probabilities of its source translation "辩护" changed from FUTURE to PAST. Results of "Bush", "his", "revive" and "economy" are similar, except a adverse case ("plan").

those to the FUTURE capsules reduce to around zeros. This phenomenon is consistent with the intuition that the translated contents should aggregate and the untranslated should decline (Zheng et al., 2018). The assignment weights of a specific word change from FUTURE to PAST after being generated. These pieces of evidence give a strong verification that our GDR mechanism indeed has learned to distinguish the PAST contents and FUTURE contents in the source-side.

Moreover, we measure how well our capsules accumulate the expected contents by comparison between the BOW predictions and ground-truth target words. Accordingly, we define a *top-5 overlap rate* ($r_{\text{OL}}$) for predicting preceding and subsequent words are defined as follow, respectively: $r^P_{\text{OL}} = \frac{1}{T}\sum_{t=1}^{T}\frac{|\text{Top}_{5\text{t}}(p_{pre}(\mathbf{\Omega}^P_t)) \cap y_{<=t}|}{|y_{<=t}|}$, $r^F_{\text{OL}} = \frac{1}{T}\sum_{t=1}^{T}\frac{|\text{Top}_{5(\text{T}-\text{t})}(p_{sub}(\mathbf{\Omega}^F_t)) \cap y_{>=t}|}{|y_{>=t}|}$. The PAST capsules achieves $r^P_{\text{OL}}$ of 0.72, while $r^F_{\text{OL}}$ of 0.70 for the FUTURE capsules. The results indicate that the capsules could predict the corresponding words to a certain extent, which implies the capsules contain the expected information of PAST or FUTURE contents.

***Translations become better and more adequate.*** To validate the translation adequacy of our model, we use Coverage Difference Ratio (CDR) proposed by Kong et al. (2019), i.e., $\text{CDR} = 1 - \frac{|C_{\text{ref}} \backslash C_{\text{gen}}|}{|C_{\text{ref}}|}$, where $C_{\text{ref}}$ and $C_{\text{gen}}$ are the set of source words covered by the reference and translation, respectively. The CDR reflects the translation adequacy by comparing the source coverages between reference and translation. As shown in Table 3, our approach achieves a better CDR than the Transformer baseline, which means superiority in translation adequacy.

| **Model** | Transformer | OURS |
|---|---|---|
| CDR | 0.73 | **0.79** |
| HUMAN EVALUATION | | |
| QUALITY | 4.39±.11 | **4.66±.10** |
| OVER(%) | 0.03±.01 | **0.01±.01** |
| UNDER(%) | 3.83±.97 | **2.41±.80** |

Table 3: Evaluation on translation quality and adequacy. For HUMAN evaluation, we asked three evaluators to score translations from 100 source sentences, which are randomly sampled from the testsets from anonymous systems, the QUALITY from 1 to 5 (higher is better), and the proportions of source words concerning OVER- and UNDER-translation, respectively.

Following Zheng et al. (2018), we also conduct subjective evaluations to validate the benefit of modeling PAST and FUTURE (the last three rows of Table 3). Surprisingly, we find that the modern NMT model, i.e., Transformer, rarely produces over-translation but still suffers from under-translation. Our model obtains the highest human rating on translation quality while substantially alleviates the under-translation problem than Transformer.

***Longer sentences benefit much more.*** We report the comparison with sentence lengths (Figure 5). In all the intervals of length, our model does generate better (Figure 5b) and longer (Figure 5a) translations. Interestingly, our approach

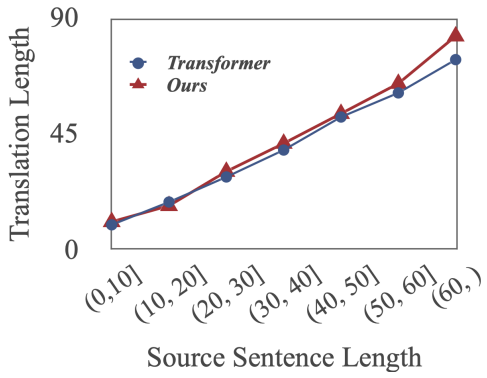


(a) Translation length v.s source length

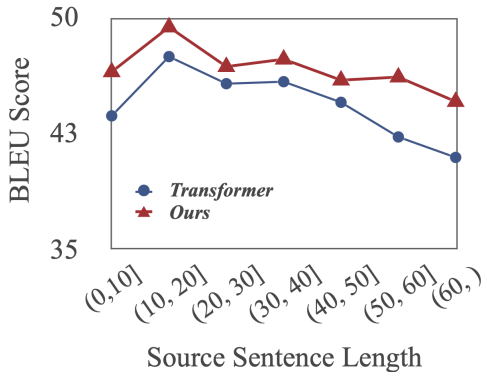


(b) BLEU v.s source length

Figure 5: Comparison regarding source length.

gets a larger improvement when the input sentences become longer, which are commonly thought hard to translate. We attribute this to the less number of under-translation cases in our model, meaning that our model learns better translation quality and adequacy, especially for long sentences.

***Does guided dynamic routing really matter?*** Despite the promising numbers of the GDR and the auxiliary guided losses, a straightforward question rises: will other more simple models also work if they are just equipped with the guided losses to recognize PAST and FUTURE contents? In other word, does the proposed guided dynamic routing really matter?

To answer this question, we integrate the proposed auxiliary losses into two simple baselines to guide the recognition of past and future: A MLP classifier model (CLF) that determines if a source word is a past word, otherwise future[5]; and an attention-based model (ATTN) that uses two individual attention modules to retrieve past or future parts from the source words. As shown in Table 6, surprisingly, the simple baselines can obtain improvements, emphasizing the function of the proposed guided losses, while there remain a considerable gaps between our model and them. In fact, the CLF is essentially a *one-iteration* variant of GDR, and iterative refinement by multiple iterations is necessary and effective[6]. And the attention mechanism is used for feature pooling, not suitable for parts-to-wholes assignment[7]. These experiments reveal that our guided dynamic routing is a better choice to model and exploit the dynamic PAST and FUTURE.

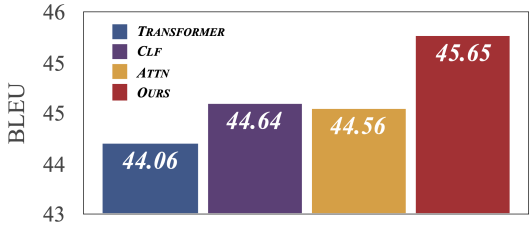


Figure 6: Comparison with simple baselines with the same auxiliary guided loss on NIST Zh-En.

## 5 Related Work

Inadequate translation problem is a widely known weakness of NMT models, especially when translating long sentences (Kong et al., 2019; Tu et al., 2016; Lei et al., 2019). To alleviate this problem, one direction is to recognize the translated and untranslated contents, and pay more attention to untranslated parts. Tu et al. (2016), Mi et al. (2016) and Li et al. (2018) employ coverage vector or coverage ratio to indicate the lexical-level coverage of source words. Meng et al. (2018) influence the attentive vectors by translated/untranslated information. Our work mainly follows the path of Zheng et al. (2018), which introduce two extra recurrent layers in the decoder to maintain the representations of the past and future translation contents. However, it may be not easy to show the direct correspondence between the source contents and learned representations in the past/future

[5]CLF is a 3-way classifier that computes the probabilities $p^P(x_i)$, $p^F(x_i)$ and $p^R(x_i)$ (they sum to 1) as past, future and redundant weights, which is similar to Equation 6. The PAST and FUTURE representations are computed by weighted summation, which is similar to Equation 4.

[6]See Appendix for analysis of iteration numbers.

[7]Consider an extreme case that in the end of translation, there is no FUTURE content left, but the attention model still produces a *weighted average* over all the source representations, which is nonsense. In contrast, the GDR is able to assign zero probabilities to the FUTURE capsules, solving the source of the problem.

RNN layers, nor compatible with the state-of-the-art Transformer for the additional recurrences prevent Transformer decoder from being parallelized.

Another direction is to introduce global representations. Lin et al. (2018) model a global source representation by deconvolution networks. Xia et al. (2017); Zhang et al. (2018); Geng et al. (2018) propose to provide a holistic view of target sentence by multi-pass decoding. Zhou et al. (2019) improve Zhang et al. (2018) to a synchronous bidirectional decoding fashion. Similarly, Weng et al. (2019) deploy bidirectional decoding in interactive translation setting. Different from these work aiming at providing static global information in the whole translation process, our approach models a dynamically global (holistic) context by using capsules network to separate source contents at every decoding steps.

Other efforts explore exploiting future hints. Serdyuk et al. (2018) design a Twin Regularization to encourage the hidden states in forward decoder RNN to estimate the representations of a backward RNN. Weng et al. (2017) require the decoder states to not only generate the current word but also predict the remain untranslated words. Actor-critic algorithms are employed to predict future properties (Li et al., 2017; Bahdanau et al., 2017; He et al., 2017) by estimating the future rewards for decision making. Kong et al. (2019) propose a policy gradient based adequacy-oriented approach to improve translation adequacy. These methods use future information only at the training stage, while our model could also exploit past and future information at inference, which provides accessible clues of translated and untranslated contents.

Capsule networks (Hinton et al., 2011) and its associated assignment policy of dynamic routing (Hinton et al., 2011) and EM-routing (Hinton et al., 2018) aims at addressing the limited expressive ability of the parts-to-wholes assignment in computer vision. In natural language processing community, however, the capsule network has not yet been widely investigated. Zhao et al. (2018) testify capsule network on text classification and Gong et al. (2018) propose to aggregate a sequence of vectors via dynamic routing for sequence encoding. Dou et al. (2019) first propose to employ capsule network in NMT using routing-by-agreement mechanism for layer representation aggregation. Wang (2019) develops a constant time NMT model using capsule networks. These studies mainly use capsule network for information aggregation, where the capsules could have a less interpretable meaning. In contrast, our model learns what we expect by the aid of auxiliary learning signals, which endows our model with better interpretability.

## 6 Conclusion

In this paper, we propose to recognize the translated PAST and untranslated FUTURE contents via parts-to-wholes assignment in neural machine translation. We propose the guided dynamic routing, a novel mechanism that explicitly separates source words into PAST and FUTURE guided by PRESENT target decoding status at each decoding step. We empirically demonstrate that such explicit separation of source contents benefit neural machine translation with considerable and consistent improvements on three language pairs. Extensive analysis shows that our approach learns to model the PAST and FUTURE as expected, and alleviates the inadequate translation problem. It is interesting to apply our approach to other sequence-to-sequence tasks, e.g., text summarization (as listed in Appendix).

## Acknowledgement

We would like to thank the anonymous reviewers for their insightful comments. Shujian Huang is the corresponding author. This work is supported by the National Science Foundation of China (No. U1836221 and No. 61772261), the Jiangsu Provincial Research Foundation for Basic Research (No. BK20170074).

## References

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. An actor-critic algorithm for sequence prediction. In *ICLR 2017*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR 2015*.

Ziyi Dou, Zhaopeng Tu, Xing Wang, Longyue Wang, Shuming Shi, and Tong Zhang. 2019. Dynamic layer aggregation for neural machine translation. In *AAAI*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann Dauphin. 2017. Convolutional sequence to sequence learning. In *ICML 2017*.

Xinwei Geng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2018. Adaptive multi-pass decoder for neural machine translation. In *EMNLP*, pages 523–532.

Jingjing Gong, Xipeng Qiu, Shaojing Wang, and Xuanjing Huang. 2018. Information aggregation via dynamic routing for sequence encoding. In *COLING*, pages 2742–2752.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O. K. Li, and Richard Socher. 2017. Non-Autoregressive Neural Machine Translation.

Di He, Hanqing Lu, Yingce Xia, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2017. Decoding with value networks for neural machine translation. In *NIPS*, pages 178–187.

Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. 2011. Transforming auto-encoders. In *ICANN*, pages 44–51. Springer.

Geoffrey E Hinton, Sara Sabour, and Nicholas Frosst. 2018. Matrix capsules with EM routing. In *ICLR*.

Xiang Kong, Zhaopeng Tu, Shuming Shi, Eduard Hovy, and Tong Zhang. 2019. Neural machine translation with adequacy-oriented learning. In *AAAI*.

Wenqiang Lei, Weiwen Xu, Ai Ti Aw, Yuanxin Xiang, and Tat-Seng Chua. 2019. Revisit automatic error detection for wrong and missing translation – a supervised approach. In *EMNLP*.

Jian Li, Baosong Yang, Zi-Yi Dou, Xing Wang, Michael R Lyu, and Zhaopeng Tu. 2019. Information aggregation for multi-head attention with routing-by-agreement. In *NAACL-HLT*, pages 3566–3575.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2017. Learning to decode for future success. *arXiv preprint arXiv:1701.06549*.

Yanyang Li, Tong Xiao, Yinqiao Li, Qiang Wang, Changming Xu, and Jingbo Zhu. 2018. A simple and effective approach to coverage-aware neural machine translation. In *ACL*, volume 2, pages 292–297.

Junyang Lin, Xu Sun, Xuancheng Ren, Shuming Ma, Jinsong Su, and Qi Su. 2018. Deconvolution-based global decoding for neural machine translation. In *COLING*, pages 3260–3271.

Thang Luong, Hieu Pham, and D. Christopher Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP 2015*.

Fandong Meng, Zhaopeng Tu, Yong Cheng, Haiyang Wu, Junjie Zhai, Yuekui Yang, and Di Wang. 2018. Neural machine translation with key-value memory-augmented attention. In *IJCAI*, pages 2574–2580. AAAI Press.

Haitao Mi, Baskaran Sankaran, Zhiguo Wang, and Abe Ittycheriah. 2016. Coverage Embedding Models for Neural Machine Translation. In *EMNLP*.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. 2017. Dynamic routing between capsules. In *NIPS*, pages 3856–3866.

Dmitriy Serdyuk, Nan Rosemary Ke, Alessandro Sordoni, Adam Trischler, Chris Pal, and Yoshua Bengio. 2018. Twin networks: Matching the future for sequence generation.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NIPS 2014*.

Zhaopeng Tu, Yang Liu, Lifeng Shang, Xiaohua Liu, and Hang Li. 2017. Neural machine translation with reconstruction. In *AAAI 2017*.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *ACL 2016*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NIPS*.

Mingxuan Wang. 2019. Towards linear time neural machine translation with capsule networks. In *EMNLP*.

Rongxiang Weng, Shujian Huang, Zaixiang Zheng, Xin-Yu Dai, and Jiajun Chen. 2017. Neural machine translation with word predictions. In *EMNLP 2017*.

Rongxiang Weng, Hao Zhou, Shujian Huang, Lei Li, Yifan Xia, and Jiajun Chen. 2019. Correct-and-memorize: Learning to translate from interactive revisions. In *IJCAI*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Yingce Xia, Fei Tian, Lijun Wu, Jianxin Lin, Tao Qin, Nenghai Yu, and Tie-Yan Liu. 2017. Deliberation networks: Sequence generation beyond one-pass decoding. In *NIPS*, pages 1784–1794.

Xiangwen Zhang, Jinsong Su, Yue Qin, Yang Liu, Rongrong Ji, and Hongji Wang. 2018. Asynchronous bidirectional decoding for neural machine translation. In *AAAI*.

Wei Zhao, Jianbo Ye, Min Yang, Zeyang Lei, Suofei Zhang, and Zhou Zhao. 2018. Investigating capsule networks with dynamic routing for text classification. In *AAAI*.

Zaixiang Zheng, Hao Zhou, Shujian Huang, Lili Mou, Xinyu Dai, Jiajun Chen, and Zhaopeng Tu. 2018. Modeling past and future for neural machine translation. *TACL*, 6:145–157.

Long Zhou, Jiajun Zhang, and Chengqing Zong. 2019. Synchronous bidirectional neural machine translation. *TACL*, 7:91–105.

# Revisit Automatic Error Detection for Wrong and Missing Translation – A Supervised Approach

**Wenqiang Lei**[1, 2], **Weiwen Xu**[2], **Ai Ti Aw**[2], **Yuanxin Xiang**[2], **Tat-Seng Chua**[1]
[1]National University of Singapore, Singapore
[2]Institute for Infocomm Research, A*STAR, Singapore
wenqianglei@gmail.com
{xuw1, aaiti, xiang_yuanxin}@i2r.a-star.edu.sg
dcscts@nus.edu.sg

## Abstract

While achieving great fluency, current machine translation (MT) techniques are bottlenecked by adequacy issues. To have a closer study of these issues and accelerate model development, we propose automatic detecting adequacy errors in MT hypothesis for MT model evaluation. To do that, we annotate missing and wrong translations, the two most prevalent issues for current neural machine translation model, in 15000 Chinese-English translation pairs. We build a supervised alignment model for translation error detection (*AlignDet*) based on a simple *Alignment Triangle* strategy to set the benchmark for automatic error detection task. We also discuss the difficulties of this task and the benefits of this task for existing evaluation metrics.

## 1 Introduction

Different translation errors impact translation comprehensibility and adequacy differently. For example, wrong product or terminology translation can be considered more severe than missing translation in the eCommerce domain. Though many machine translation evaluation (MTE) metrics have been proposed, most of them are not able to provide a direct connection between the score and the error class to emphasize the different impact on translation comprehension and adequacy. Most MTE scores measure the similarity between the MT hypothesis and the reference using n-gram and are useful in providing immediate feedback on model performance for system development and objective evaluation for system comparison. To further analyse the MT hypothesis, fine-grained translation analysis is normally conducted manually which involves expensive human effort (Popović and Ney, 2011; Popović, 2011b; Fishel et al., 2012; Vilar et al., 2006).

This motivates early unsupervised error detection methods (Popović and Ney, 2011). Such methods find the erroneous words by calculating the edit distance between the MT hypothesis and its corresponding reference. However, such unsupervised methods suffer from low accuracy and inability to provide and distinguish certain error classes like *Missing words* and *Wrong translation*[1] (c.f. Sec 3). On the other hand, the works on supervised methods are few due to limited training resources available. The annotated corpus are mainly collected from outputs of Phrase-based Machine Translation systems and usually suffer from small quantity (around 200 instances per language pair) (Fishel et al., 2012), noisy contents (derived from student assignments) (Wisniewski et al., 2014) or overabundant efforts (additional human efforts are required on postediting the MT hypothesis) (Popović and Arcan, 2016), making the progress and development of this task challenging.

We believe a corpus annotated with different translation error classes will facilitate the research on translation error detection. Therefore, we construct a high quality annotated corpus (*TransErr*) comprising 15000 Chinese-English translation pairs with inter-annotator agreement at 0.804 measured by Cohen's Kappa (Cohen, 1960). Different from existing error detection works which focus on all error classes, we currently only take care of missing and wrong translation (Specia et al., 2018), the major errors related to adequacy, which is a wide-known issue in neural machine translation (NMT) (Zheng et al., 2019). The errors tags are annotated on source (Chinese) sentences to reflect the loyalty and adequacy with respect to the source. Based on *TransErr*, we discuss the error distribution of dif-

[1]The *wrong translation* is referred to as *incorrect lexical choice* in (Popović and Ney, 2011)



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 942–952, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

ferent systems and challenges of unsupervised error annotations using post-edits.

*TransErr* enables the study task of supervised MT error detection. We benchmark this task by providing a simple yet efficient model called (*AlignDet*). *AlignDet* is based on our observation of the *Alignment Triangle* whose basic idea is that if a word is well translated, the translated word and the corresponding word in reference should be equivalent. We implement such observation by employing both monolingual and bilingual alignment systems as well as various features. We further provide error analyses of *AlignDet* to give more insight to this task.

To study the application of this task, we further conduct a discussion on the impact of different error classes on MT evaluation metrics based on WMT Chinese–English Direct Assessment (DA) corpus (Bojar et al., 2017). We find that embracing the results of our simple *AlignDet* model in existing MTE metrics helps to achieve significantly better correlation with human. Our further analysis on gold standard error annotation suggests that wrong translation tends to produce worse translation. This discovery gives further credence for the usefulness of our proposed adequacy-oriented error detection task.

In short, the paper is towards carrying out the first systematic study of supervised approaches for error detection, with the goal of accelerating the research and development of MT towards human translation quality. Our systematic contribution lies on annotated data, supervised model and discussions of its potential to help existing machine translation evaluation task. Specifically:

- We propose a task of adequacy-oriented error detection for machine translation evaluation and contribute a high-quality corpus *TransErr* annotated by professional annotators for machine translation. The corpus are available[2].
- We propose an *AlignDet* to set the benchmark of this task based on our key observation of *Alignment Triangle*. As far as we know, this is the first model making use of all three alignments from source, MT hypothesis and reference concurrently.
- We conduct various discussions on the challenges of this task and its potential to help existing machine translation evaluation task.

[2]Please contact us to get the *TranErr* corpus and the *AlignDet* source code. Before that, you need to have access to LDC NIST MT evaluation sets {LDC2005T06, NIST02, NIST03, NIST04, NIST05, NIST08}.

## 2 Related Work

Existing automatic MTE focus on measuring the similarity between MT hypothesis and references. Various dimensions have been considered, such as lexical level (Papineni et al., 2002; Snover et al., 2009; Lo, 2017; Popović, 2017), syntax level (Owczarzak et al., 2007; Duma and Menzel, 2017; Liu and Gildea, 2005), semantic level (Stanojević and Sima'an, 2015; Shimanaka et al., 2018) and discourse level (Guzmán et al., 2014). However, such methods give an overall score to a system or give a relative ranking to a pair of systems, without any detailed insight of MT system output.

To get the fine-grained insights of the MT model performance, Popović and Ney (2011); Popović et al. (2011a) make the first step towards automatic translation error detection using unsupervised approach by performing monolingual alignment between a MT hypothesis and its corresponding reference through WER (Levenshtein, 1966) and PER (Tillmann et al., 1997). Popović (2011b) further develop this algorithm into an open-source tool and demonstrates that the detected errors helps to build better evaluation metrics. However, the edit distance algorithm is not robust (*c. f.* Sec. 3.2). Zeman et al. (2011) improve the monolingual alignment by borrowing ideas from HMM (Vogel et al., 1996) but the error detection performance is still far from real application. To facilitate this task, Wisniewski et al. (2014) derive corpus of 4,854 source sentences from the assignment of master student specializing on translation but the data turns out to be too noisy.

At the same time, edit distance is carried over to another shared task in WMT (Callison-Burch et al., 2012; Specia et al., 2018; Fan et al., 2018; Vaswani et al., 2017) called word-level Quality Estimation (QE) (Luong et al., 2013; Han et al., 2013; Wisniewski et al., 2014; Kim et al., 2017). In QE tasks, the edit distance is calculated between a MT hypothesis and its post-edits. Word needs to be edited is tagged as `BAD` while the unedited words are tagged as `GOOD`. The task requires participants to predict the label for each word. Although similar to our settings, our error detection

task differs from the word-level QE task in the following aspects:

- The QE task is for confidence estimation (Specia et al., 2010) which estimates whether an output is good for an end user. Our error detection task is for fine-grained evaluation and comparison during model development.
- QE only have `GOOD` and `BAD` tag on the MT hypothesis. It cannot make robust distinction on different error classes due to the intrinsic shortcomings of the edit distance algorithm.
- Existing QE data does not have reliable label on the source sentence to indicate which source word cause the errors (*c.f.* Sec. 3.2), which is claimed as "particularly important to translation adequacy" in WMT 18 (Specia et al., 2018) and WMT 19[3].

## 3 TranErr Corpus and Analysis

*TransErr* corpus comprises 15000 triples <source, MT hypothesis, reference> with source sentences extracted from LDC NIST MT evaluation sets {LDC2005T06, NIST02, NIST03, NIST04, NIST05, NIST08}. MT hypothesis is obtained through an NMT engine trained on 25 million Chinese-English parallel sentences. To study the correlation between error detection and human evaluation, we also conduct additional annotation on WMT17 Direct Assessment (DA) dataset containing 560 Chinese-English translation pairs.

Annotation is conducted by two professional translators who specialize on Chinese-English translation. All three information <source, MT hypothesis, reference> are given during annotation. Annotators are instructed to mark on the source the two adequacy related errors (*Missing* & *Wrong*) found in the MT hypothesis and to follow the minimum span principle, i.e. identifying the minimum number of words whose content are not (correctly) conveyed in the MT hypothesis regardless of fluency and grammatical correctness. If a span in the source is incorrectly translated in the MT hypothesis, all words in that span will be marked as *Wrong* (`W`); if totally missing, it will be marked as *Missing* (`M`). Due to the intrinsic difference between the two languages, not all words in the source need to be translated in the MT hypothesis. In this case, we do NOT annotate them as missing. We also have more stringent conditions for proper nouns and terminologies since their translations are more specific and domain dependent. They have to be unerringly correct or otherwise, marked as *Wrong Terminology* (`WT`) or *Missing Terminology* (`MT`[4]). All correctly translated words are automatically assigned a label `OK`.

Annotators are first asked to annotate 100 sentence pairs followed by a discussion among the project team to resolve any disagreement. After 5 such iterations, we start the formal annotation. 10% overlapped instances are given to each annotator for monitoring the inter-annotation agreement (IAA). The disagreed instances are discussed every day. We obtain an overall IAA of 0.804 measured by Cohen's Kappa. The speed for both annotators is about 50 sentence pairs per hour.

### 3.1 Translation Error Analysis

Table 1 shows the error distribution of *TransErr* and WMT 17 corpus. The corpora have different BLEU with *TransErr* having more missing (`M`) errors and WMT17 more wrong (`W`) errors. As WMT17 contains output from multiple MT systems, we further investigate the errors distributions among the different systems, in particular the output from AFRL and NRC. AFRL and NRC have comparable BLEU but different human evaluation. Error distributions for the two corpora vary quite significantly with AFRL having more `W` errors (45% `W` and 19% `WT`) while NRC has more `M` errors (31% `M` and 31% `MT`). This demonstrates the importance of classifying translation errors and the need for the errors to be considered during evaluation.

### 3.2 Challenges of Automatic Translation Error Annotation

Though MTE and error detection are two different tasks, they are closely related. Popović and Ney (2011) presented the first framework for automatic analysis and classification of translation errors using unsupervised approach for SMT systems. The work has been evolved to word-level QE task in WMT. Instead of directly annotating errors on the source as in our proposed task, we re-implement

[3] http://www.statmt.org/wmt19/

[4] In this paper, we conventionally use the normal font MT to represent *machine translation* while use the typewriter font `MT` to stand for *Missing Terminology*.

| | Human | BLEU | Sen | Word | OK | W | WT | M | MT |
|---|---|---|---|---|---|---|---|---|---|
| **TransErr** | - | 19.1 | 15000 | 403691 | 388372 | 4529 / 0.29 | 1694 / 0.11 | 8234 / 0.54 | 862 / 0.06 |
| **WMT17** | - | 16.2 | 560 | 14090 | 13025 | 410 / 0.39 | 206 / 0.19 | 320 / 0.30 | 129 / 0.12 |
| **AFRL** | -0.016 | 20.6 | 34 | 917 | 820 | 44 / 0.45 | 19 / 0.19 | 17 / 0.18 | 17 / 0.18 |
| **NRC** | 0.079 | 20.3 | 32 | 786 | 698 | 17 / 0.19 | 17 / 0.19 | 27 / 0.31 | 27 / 0.31 |

Table 1: Corpus Statistics for *TransErr*, WMT17, AFRL and NRC. AFRL and NRC are subsets of WMT17. Human is the corresponding system-level DA score assigned by human in WMT17. All BLEU are calculated using **one reference**. Sen and Word columns list the number of sentences and words while the other four columns present the number of each error class and its proportion.

| Label | OK | W | WT | M | MT |
|---|---|---|---|---|---|
| F1 | 0.93 | 0.33 | 0.30 | 0.43 | 0.35 |

Table 2: F1 for each class annotated by the unsupervised method using our manual annotation as gold reference

QE and follow (Popović and Ney, 2011) to derive source annotation through two stages of alignments. The first stage is monolingual alignment between post-edits and MT hypotheses through calculating their edit distance to find missing and wrong translations. The second stage is bilingual alignment using Giza++ (Och and Ney, 2000) on post-edits and the sources to propagate the error labels on the post-edits to the corresponding sources. Similar to our manual annotation, we assign `MT` and `WT` label for terminologies.

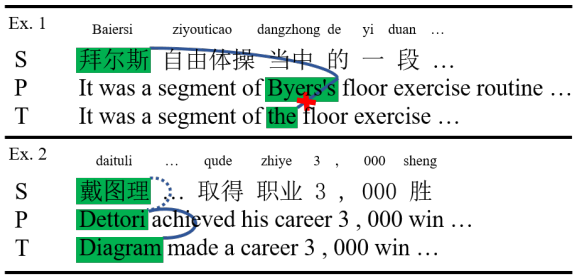


Figure 1: Examples of unsupervised annotation on WMT17. (S), (P) and (T) refer to source, post-edits and MT hypothesis respectively. Solid line indicates correct alignment. Solid line with a red cross refers to wrong alignment while dotted line refers to missing alignment. This convention applies to the whole paper.

To get empirical study, we perform unsupervised annotation on our WMT17 dataset using the above-mentioned method. Table 2 demonstrates its F1 score using the manual annotation as the gold standard. We investigate and summarize the causes in Figure 1:

- The edit distance algorithm is not robust in error class detection since it is purely based on symbolic comparison. (Ex. 1) illustrates a case where *Byers's* is missing. However, TER tool tags it as a substitution of *the* in the MT hypothesis because there is a perfect symbolic alignment before and after the erroneous word. This intrinsic inability to detect missing and wrong translation is also discussed in (Popović and Ney, 2011).

- Giza++ cannot generate reliable alignments for low-frequency words (Riley and Gildea, 2010) and such low-frequency words contribute to a large extent of erroneous words. These misalignments propagate from the post-edits to the source sentence and lead to incorrect error tags. (Ex. 2) gives an example, the *Dettori* should be aligned to *daituli* but was not aligned to any word by GIZA++. This leads to an incorrect `OK` tag in the source sentence.

As such, unsupervised error annotation presents many challenges even leveraging on post-edits. Hence a set of more reliable annotated data will be able to contribute much to the error detection task.

## 4 AlignDet Model

To revive fine-grained translation error detection, we propose *AlignDet*, a supervised feature-based solution using *TransErr* to set the benchmark for this task. We discuss our *Alignment Triangle* observation and briefly list the features we used. In the subsequent discussions, we denote $s_i$, $t_j$ and $r_k$ as the $i$th, $j$th and $k$th word in a source sequence $S$, MT hypothesis $T$ and reference $R$ separately.

### 4.1 Alignment Triangle Observation

In translation, adequacy can be simply defined as all contents in the source are reproduced correctly




Figure 2: Four samples from *TransErr*. For each example, we show the source (S), MT hypothesis (T), reference (R).

in the MT hypothesis. As long as the MT hypothesis and the reference are semantically equivalent, such reproduction may not require the MT hypothesis and the reference to be exactly identical. This motivates us to analyze the translation $t_j$ in $T$ and $r_k$ in $R$ for the same $s_i$ in $S$[5], in the following error detection scenarios. We refer $t_j$ and $r_k$ for the same $s_i$ as translation correspondence.

**Pattern1: $s_i$ has translation correspondence $t_j$ and $r_k$.** It should be labeled as `OK` if $t_j$ and $r_k$ are semantically equivalent; otherwise, it should be labeled as `W`. For example, in (Ex. 3) from Figure 2, the two words, *swap* and *exchange*, are equivalent in the given context, which should be marked as `OK`. However, compared to (Ex. 6), *students* and *comers* are not equivalent in the given context, although they may be equivalent in higher abstraction, hence it is annotated as `W`.

**Pattern2: $s_i$ does not have translation correspondence $r_k$.** Assuming *R* does not have any adequacy issue, this would imply that $s_i$ does not need to be explicitly translated. In this case, the omission of $t_j$ is an `OK` translation. However, if it is explicitly translated in *T* as $t_j$, we will still accept it as an `OK` translation though it may cause minor redundancy issue and we assume that it will not harm adequacy. For example, either explicitly translation *meiguo* to *USA* or not in (Ex. 4) should be fine.

**Pattern3: $s_i$ has translation correspondence $r_k$ but not $t_j$.** This implies that translation of $s_i$ is missing and should be translated. A `M` label will be given in this case. (Ex. 5) shows an example of this. *jinzhu* does not have $t_j$ but it is necessary for it to be translated as the translation correspondence is presence in *R* as $r_k$.

The same rationale also applies to terminologies, except that they are marked with `MT` or `WT` and have to be unerringly translated for them to be marked as `OK`. For the other cases not illustrated above, we assume that they are either not happening or do not affect the adequacy of the translation.

### 4.2 Features and Model

To capture the above observation, we build two sets of features, one for finding the corresponding translation $t_j$ and $r_k$ for each $s_i$, the other for checking the equivalence of $t_j$ and $r_k$. Here, we only briefly list the features adopted in our model as they are all commonly used standard features. We give detailed description of all features in Appendix A.1.

Giza++ (Och and Ney, 2000) is used to find the corresponding translation $t_j$ and $r_k$ for each $s_i$. However, as Giza++ generates noisy alignments especially for low-frequency words, we propose a set of features to complement Giza++ results. The same features are applied on $T$ and $R$, hence we only describe the feature for the alignment between $S$ and $T$. Suppose that $s_i$ is aligned to $t_j$[6]. These features are: $s_i$'s POS tag; a binary feature of whether NER tags of $s_i$ and $t_j$ are the same; the corpus frequency and Giza++ translation probability of $s_i$ and $t_j$; the similarity between $t_j$ and $s_i$'s most frequent translation; the number of aligned-pairs among words linking to $s_i$ and $t_j$ in dependency tree.

We leverage on the state-of-the-art monolingual alignment tool Sultan et al. (2014) to check the equivalence of $t_j$ and $r_k$. The tool leverages on paraphrase lexicon (Pavlick et al., 2015) and dependency relations to find equivalent expression between two sentences in the same language. The following features are used: a binary feature indicating whether $t_j$ and $r_k$ are aligned by tool proposed by (Sultan et al., 2014); A binary feature indicating whether the NER tag of $t_j$ and $r_k$ are the same.

While this error detection problem can be modeled as sequential labelling task, we choose simple classification model as the benchmark. Therefore,

[5]Generally, $t_j$, $r_k$ and $s_i$ can be multi-word expression. For the ease of explanation for our feature extraction method, we describe each of them as a single word.

[6]Following Hu et al. (2018), we make adjustment by penalizing the frequency of both words.

we use an Mutilayer Perceptron (MLP) as the detection model which takes all above features in for each source word $s_i$ to choose one of the labels in {OK, W, WT, M, MT} as the prediction.

## 5 Experiments

We conduct empirical experiments to evaluate the effectiveness of our error detection model, perform error analysis and discuss the challenges of this task. We split our *TransErr* dataset into training and developing sets in a ratio of 9:1 and adopt the whole WMT DA dataset as the testing set. Error detection results are reported in single-class F1. All Chinese sentences are tokenized by THULAC (Li and Sun, 2009).The rest of the pre-processings like POS, NER, parsing for both English and Chinese are performed by Stanford Corenlp (Manning et al., 2014).

### 5.1 Baselines

We build a naive baseline, random model, to test the worst performance. It randomly assigns a label to each source word according to the distribution of each class in the training set. We then evaluate the performance of our proposed MLP model (*AlignDet*) and two recent models under WMT word-level QE shared-task:

· **CEQE** (Hu et al., 2018) Using a three-part neural network model, which encodes semantic factors, local context and global context successively.

· **SHEF-bRNN** (Ive et al., 2018) Adopting bi-directional recurrent neural network to learn a representation for <source, MT hypothesis> sentence pairs.

We replicate these two models utilizing original implementations but modify them to work in our 5-label setting.[7] We also analyze our proposed model and its variant by ablating or substituting its components

· **Raw alignment** Using only the raw results from the monolingual and bilingual alignment systems and NER on the source to examine the effect of alignment on the model.

· **No reference** We remove all features derived from the reference, for example, bilingual alignment features between $S$ and $R$, and monolingual alignment features between $T$ and $R$. This is to test the efficacy of using only source information for translation performance evaluation.

[7]We also tried QEBrain (Fan et al., 2018; Wang et al., 2018) using the released code. However, it did not converge to any meaningful predictions.

### 5.2 Error Detection Results

Table 3 shows our error detection results on the developing set and test set. Our random baseline (Row (1)) provides a preliminary insight for this error detection problem, with extremely low F1 scores (nearly 0) except for the OK class. It is because errors are scarcely distributed (see Table 1). In addition, the QE models (Row 2, 3) do not perform well in error classification. This is reasonable because they are not designed for error detection in our setting. The overall better performance of our *AlignDet* model supports that our *Alignment Triangle* Observation and its contribution to the error detection task.

However, with raw alignment (Row 4 vs Row 5), the performance of *AlignDet* drops sharply. It is because the alignment model, especially Giza++, generates noisy results. This demonstrates the efficacy of other features in rectifying Giza++ results. The comparison between Row (4) and Row (6) also demonstrates the importance of reference, especially for detecting W label. It is because wrong translation is also likely to get aligned with the source as it can be the translation of the source words in some other contexts. However, reference gives us a pointer to the correct word choice in the given context, hence helping determining W label. Without reference, we may need to rely on more contextual modeling effort to detect a wrong translation.

### 5.3 Error Analysis

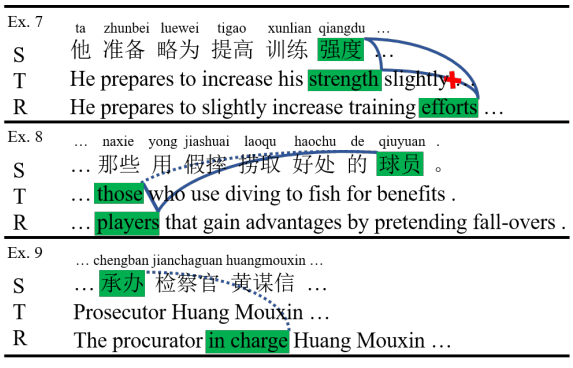


Figure 3: Representative examples of the confusion cases for *AlignDet*.

To get more insight into the difficulty of this task, we perform error analysis on *AlignDet*. We only examine the confusion matrix on test set due to the limitation of space in Table 4. It shows that the significant problems are: (a) Predict W as OK

| | Developing | | | | | Testing | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | OK | W | WT | M | MT | OK | W | WT | M | MT |
| (1) Random baseline | 0.959 | 0.009 | 0.013 | 0.020 | 0.027 | 0.947 | 0.025 | 0.029 | 0.041 | 0.017 |
| (2) CEQE | 0.944 | 0.134 | 0.180 | 0.245 | 0.161 | 0.914 | 0.161 | 0.147 | 0.206 | 0.094 |
| (3) SHEF-bRNN | 0.916 | 0.140 | 0.212 | 0.127 | 0.262 | 0.884 | 0.153 | 0.123 | 0.098 | 0.114 |
| (4) AlignDet | **0.973** | **0.275** | **0.480** | **0.416** | **0.365** | **0.964** | **0.212** | **0.293** | **0.400** | **0.429** |
| (5) Raw alignment | 0.762 | 0.051 | 0.049 | 0.168 | 0.114 | 0.781 | 0.131 | 0.131 | 0.263 | 0.328 |
| (6) No reference | 0.977 | 0.040 | 0.448 | 0.289 | 0.289 | 0.962 | 0.075 | 0.213 | 0.360 | 0.378 |

Table 3: F1 score for each class produced by different error detection models.

| Gold \ Pred. | OK | W | WT | M | MT |
|---|---|---|---|---|---|
| OK | 12635 | 203 | 68 | 106 | 13 |
| W | 279 | 83 | 3 | 40 | 5 |
| WT | 109 | 28 | 54 | 9 | 6 |
| M | 121 | 54 | 12 | 122 | 11 |
| MT | 26 | 12 | 31 | 16 | 44 |

Table 4: Confusion matrix on test set.

(b) Predict OK as M (c) Predict M as OK. We illustrate the wrong prediction in Figure 3.

(Ex. 7) demonstrates a case for "W as OK". Both *strength* (the wrong translation in T) and *efforts* (the correct translation in R) are aligned to the source word *qiangdu* by Giza++. Though both of them are possible translations for *qiangdu*, they are not semantically equivalent in this context. Unfortunately, monolingual alignment tool aligns them as a pair, hence a more contextual and semantically-sensitive monolingual alignment is required. (Ex. 8) demonstrates a case for "OK as M". In $T$, *qiuyuan* is translated to *those* and is not detected by our bilingual alignment. This leads the model to treat it as not been translated. This example demonstrates the constraint of bilingual alignment in finding the translation correspondence, which would require contextual-level understanding. (Ex. 9) demonstrates a case for "M as OK". Similar to (Ex. 8), the mis-prediction is caused by the failure of bilingual alignment (between *chengban* in $S$ and *in charge* in $R$). Our model regards *chengban* as omittable as bilingual alignment fails to align it to *in charge* in $R$. As a result, though *chengban* is translated in R but not translated in $T$, we classify it as OK instead of M. This example again demonstrates the important of alignment in error detection task.

Our qualitative error analyses highlight the bottleneck of *AlignDet* model on the proposed alignment features. In a broader context, both bilingual alignment and monolingual alignment are still having open issues in the realm of NLP (Sultan et al., 2014). An intuitive solution is turning to soft alignment (e.g., attention) using neural techniques (Liu and Sun, 2015; Tamura et al., 2014). This raises interesting and challenging questions on how to leverage on such approaches to capture our observations in finding corresponding translations $t_j$ and $r_k$ for each $s_i$ and comparing their equivalence? Neural models usually require large data and how to leverage on limited error annotation data (say, 15000 sentence pairs) and large parallel corpus for such purpose?

## 6 Discussion: How Does Error Class Affect Existing MT Evaluation Metrics?

This paper proposes an adequacy translation error detection model with the aim to study its contribution to MT. We presume that fine grained MT evaluation is useful for assessing different properties of MT models. Hence, we go one step further to discuss how fine-grained error detection benefits existing MT evaluation metrics using DA as the base for our evaluation.

To explore further, we define a Weight Adequacy Error Rate (WAER) following TER (Snover et al., 2006):

$$\texttt{WAER} = \frac{w_1\#\texttt{W} + w_2\#\texttt{WT} + w_3\#\texttt{M} + w_4\#\texttt{MT}}{\texttt{Len(S)}} \quad (1)$$

where #W, #WT, #M, #MT are the number of wrong words, wrong terminologies, missing words, missing terminologies respectively; $w_1$, $w_2$, $w_3$, $w_4$ are corresponding weight for each error class, indicating its importance. Len(S) is the source sentence length.

With only missing and wrong error class, WAER is not sufficient to serve as an independent eval-

| | Base | + pred. | + gold. |
|---|---|---|---|
| BLEU | 0.482 | 0.524 | 0.541 |
| METEOR | 0.638 | 0.673 | 0.709 |
| ROUGE-L | 0.582 | 0.645 | 0.699 |
| BEER | 0.582 | 0.640 | 0.681 |
| CUNI-TreeAggreg | 0.535 | 0.625 | 0.670 |
| MEANT_2.0 | 0.639 | 0.641 | 0.680 |
| MEANT_2.0-nosrl | 0.630 | 0.633 | 0.672 |
| UHH_TSKM | 0.477 | 0.600 | 0.657 |
| BLEU2VEC_sep | 0.526 | 0.588 | 0.620 |
| CHRF | 0.591 | 0.641 | 0.685 |
| CHRF++ | 0.593 | 0.644 | 0.690 |
| NGRAM2VEC | 0.520 | 0.576 | 0.607 |
| Ave | 0.573 | 0.622 | 0.662 |

Table 5: Segment-level Pearson correlation of various metrics and their enhancements with our $WAER$ measure through (Eq. 2). + pred./+gold means $WAER$ is calculated by our error detection model predicted results/gold standard as per se.

uation metric. As a common practice in BLEU and METEOR, we view WAER as a penalty to base metrics, which we refer as Score, and calculate the penalized score as follow:

$$\widetilde{\texttt{Score}} = \texttt{Score} * (1 - \texttt{WAER}) \quad (2)$$

To determine $w_1$, $w_2$, $w_3$, $w_4$, we build a development set of 300 instances, sampled from our *TransErr*, with overall translation quality scored by multiple judges who are professional at both English and Chinese. To avoid bias, our error annotators of *TransErr* are excluded from this scoring. We adjust $w_1$, $w_2$, $w_3$, $w_4$ in this development set based on our gold annotation, such that the enhanced metrics achieves highest correlation with the average human score.

We examine three benchmark metrics (BLEU, METEOR, ROUGE) and the recent metrics submitted to WMT17: BEER (Stanojević and Sima'an, 2015), CUNI-TreeAggreg (Mareček et al., 2017), MEANT_2.0, MEANT_2.0-nosrl (Lo, 2017), UHH_TSKM (Duma and Menzel, 2017), NGRAM2VEC, BLEU2VEC_sep (Tättar and Fishel, 2017), CHRF (Popović, 2015), CHRF++ (Popović, 2017). We test their performance on WMT 17 DA dataset. The results are in Table 5. WAER is calculated by both system prediction results and the gold standard. We observe that, with our error prediction, WAER is able to enhance almost all metrics substantially (+0.049 on average). When calculating WAER using the gold standard, the performance is even higher (+0.089 on average). This demonstrates the efficacy of error detection for enhancing existing direct assessment metrics. Note that, we do not conclude that our WAER (Eq. 1) and the way of enhancing existing Score (Eq. 2) is optimal in terms of achieving better correlations with human judgment. We also want to find out which error class affects the overall translation quality more. Therefore, we study how the enhanced metrics correlate to human judgment when the weight of a single class of error changes. The trend is shown in Figure 4 where y-axis is the average correlation of all metrics enhanced by WAER on gold standard. X-axis is the weight of the observed error class. To eliminate the contribution made by other error classes, weights of other error classes are set to 0. From Figure 4, one can observe that wrong related error (W and WT) can help existing metrics achieve higher correlation to human judgment than missing-related error (M and MT ). This suggests wrong translation is more severe than missing word. This provides valuable feedback on the advancement of evaluation metric towards human evaluation. Similarly, one can also observe that adequacy issue on terminologies is more severe than normal words and term-related errors need higher weights to achieve highest correlation. Accurate translation of terminology is an issue in neural machine translation if its occurrence is low in the training data. Accurate translation of terminology may be an area that warrens more work.

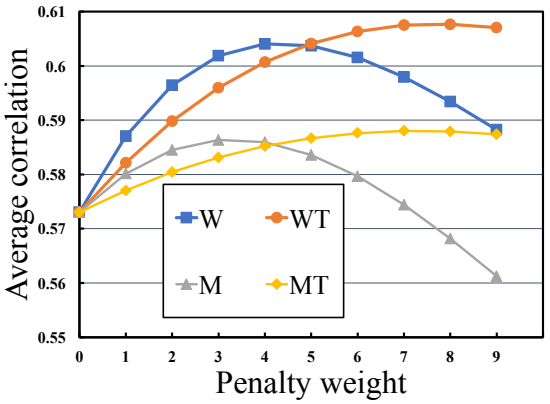


Figure 4: Correlation with respect to each error class.

## 7 Conclusion

In this paper, we revive the problem of error detection task for fine-grained machine translation

evaluation purpose. We contribute a high-quality dataset *TransErr* with *Missing* and *Wrong* translation manually annotated by professional translators to enable the development of supervised methods. Based on *TransErr*, we benchmark this task by proposing a strong baseline, i.e., *AlignDet* model, based on our *Alignment Triangle* observation. We also conduct various discussions about this task such as the challenges of unsupervised methods for error annotation, the bottleneck of *AlignDet* model and the potential benefits of the error detection task for existing MT evaluation metrics task.

This work represents the preliminary work for more multi-faceted machine translation evaluation, focusing on multiple aspects instead of only a score or ranking, with the goal of push MT techniques to a higher standard. However, there are areas for further exploration. In the future, we will explore more advanced techniques such as neural networks for error detection. We will also investigate how to make better evaluation metrics with the help of error detection. We will also study how to improve machine translation model through error detection. For example, how to improve MT models according to different class of errors.

## 8 Acknowledgements

NExT++ research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its IRC@SG Funding Initiative. The work is also supported by The Translational RD Grant (TRANS Grant) initiative managed by the Government Technology Agency of Singapore (GovTech) and the National Research Foundation (NRF), Singapore.

## References

Ondřej Bojar, Yvette Graham, and Amir Kamran. 2017. Results of the wmt17 metrics shared task. In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 489–513, Copenhagen, Denmark. Association for Computational Linguistics.

Chris Callison-Burch, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2012. Findings of the 2012 workshop on statistical machine translation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 10–51, Montréal, Canada. Association for Computational Linguistics.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46.

Melania Duma and Wolfgang Menzel. 2017. Uhh submission to the wmt17 metrics shared task. In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 582–588, Copenhagen, Denmark. Association for Computational Linguistics.

Kai Fan, Bo Li, Fengming Zhou, and Jiayi Wang. 2018. " bilingual expert" can find translation errors. *arXiv preprint arXiv:1807.09433*.

Mark Fishel, Ondrej Bojar, and Maja Popovic. 2012. Terra: a collection of translation error-annotated corpora. In *LREC*, pages 7–14.

Francisco Guzmán, Shafiq Joty, Lluís Màrquez, and Preslav Nakov. 2014. Using discourse structure improves machine translation evaluation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 687–698.

Aaron Li-Feng Han, Yi Lu, Derek F. Wong, Lidia S. Chao, Liangye He, and Junwen Xing. 2013. Quality estimation for machine translation using the joint method of evaluation criteria and statistical modeling. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 365–372, Sofia, Bulgaria. Association for Computational Linguistics.

Junjie Hu, Wei-Cheng Chang, Yuexin Wu, and Graham Neubig. 2018. Contextual encoding for translation quality estimation. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 788–793, Belgium, Brussels. Association for Computational Linguistics.

Julia Ive, Frédéric Blain, and Lucia Specia. 2018. Deepquest: a framework for neural-based quality estimation. *In the Proceedings of COLING 2018, the 27th International Conference on Computational Linguistics, Sante Fe, New Mexico, USA*.

Hyun Kim, Jong-Hyeok Lee, and Seung-Hoon Na. 2017. Predictor-estimator using multilevel task learning with stack propagation for neural quality estimation. pages 562–568.

Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710.

Zhongguo Li and Maosong Sun. 2009. Punctuation as implicit annotations for chinese word segmentation. *Computational Linguistics*, 35(4):505–512.

Ding Liu and Daniel Gildea. 2005. Syntactic features for evaluation of machine translation. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 25–32.

Yang Liu and Maosong Sun. 2015. Contrastive unsupervised word alignment with non-local features. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Chi-kiu Lo. 2017. Meant 2.0: Accurate semantic mt evaluation for any output language. In *Proceedings of the second conference on machine translation*, pages 589–597.

Ngoc Quang Luong, Benjamin Lecouteux, and Laurent Besacier. 2013. LIG system for WMT13 QE task: Investigating the usefulness of features in word confidence estimation for MT. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 386–391, Sofia, Bulgaria. Association for Computational Linguistics.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

David Mareček, Ondřej Bojar, Ondřej Hübsch, Rudolf Rosa, and Dusan Varis. 2017. Cuni experiments for wmt17 metrics task. In *Proceedings of the Second Conference on Machine Translation*, pages 604–611.

F. J. Och and H. Ney. 2000. Improved statistical alignment models. pages 440–447, Hongkong, China.

Karolina Owczarzak, Josef van Genabith, and Andy Way. 2007. Evaluating machine translation with lfg dependencies. *Machine Translation*, 21(2):95–119.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2015. Ppdb 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 425–430.

Maja Popović. 2011b. Hjerson: An open source tool for automatic error classification of machine translation output. *The Prague Bulletin of Mathematical Linguistics*, 96:59–67.

Maja Popović. 2015. chrf: character n-gram f-score for automatic mt evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395.

Maja Popović. 2017. chrf++: words helping character n-grams. In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 612–618, Copenhagen, Denmark. Association for Computational Linguistics.

Maja Popović and Mihael Arcan. 2016. Pe2rr corpus: manual error annotation of automatically pre-annotated mt post-edits. In *Proceedings of the tenth international conference on Language Resources and Evaluation (LREC 2016)*, pages 27–32.

Maja Popović, Aljoscha Burchardt, et al. 2011a. From human to automatic error classification for machine translation output. In *15th International Conference of the European Association for Machine Translation (EAMT 11)*.

Maja Popović and Hermann Ney. 2011. Towards automatic error analysis of machine translation output. *Computational Linguistics*, 37(4):657–688.

Darcey Riley and Daniel Gildea. 2010. Improving the performance of giza++ using variational bayes. *The University of Rochester, Computer Science Department, Tech. Rep*, 963.

Hiroki Shimanaka, Tomoyuki Kajiwara, and Mamoru Komachi. 2018. Ruse: Regressor using sentence embeddings for automatic machine translation evaluation. In *Proceedings of the Third Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 764–771, Belgium, Brussels. Association for Computational Linguistics.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of association for machine translation in the Americas*, volume 200.

Matthew Snover, Nitin Madnani, Bonnie J Dorr, and Richard Schwartz. 2009. Fluency, adequacy, or hter?: exploring different human judgments with a tunable mt metric. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 259–268. Association for Computational Linguistics.

Lucia Specia, Frédéric Blain, Varvara Logacheva, Ramón Astudillo, and André FT Martins. 2018. Findings of the wmt 2018 shared task on quality estimation. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 689–709.

Lucia Specia, Dhwaj Raj, and Marco Turchi. 2010. Machine translation evaluation versus quality estimation. *Machine Translation*, 24(1):39–50.

Miloš Stanojević and Khalil Sima'an. 2015. Beer 1.1: Illc uva submission to metrics and tuning task. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 396–401.

Md Arafat Sultan, Steven Bethard, and Tamara Sumner. 2014. Back to basics for monolingual alignment: Exploiting word similarity and contextual evidence. *Transactions of the Association for Computational Linguistics*, 2:219–230.

Akihiro Tamura, Taro Watanabe, and Eiichiro Sumita. 2014. Recurrent neural networks for word alignment model. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1470–1480.

Andre Tättar and Mark Fishel. 2017. bleu2vec: the painfully familiar metric on continuous vector space steroids. In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 619–622, Copenhagen, Denmark. Association for Computational Linguistics.

Christoph Tillmann, Stephan Vogel, Hermann Ney, Arkaitz Zubiaga, and Hassan Sawaf. 1997. Accelerated dp based search for statistical translation. In *Fifth European Conference on Speech Communication and Technology*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

David Vilar, Jia Xu, D'Haro Luis Fernando, and Hermann Ney. 2006. Error analysis of statistical machine translation output. In *LREC*, pages 697–702.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. Hmm-based word alignment in statistical translation. In *Proceedings of the 16th conference on Computational linguistics-Volume 2*, pages 836–841. Association for Computational Linguistics.

Jiayi Wang, Kai Fan, Bo Li, Fengming Zhou, Boxing Chen, Yangbin Shi, and Luo Si. 2018. Alibaba submission for wmt18 quality estimation task. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 809–815.

Guillaume Wisniewski, Natalie Kübler, and François Yvon. 2014. A corpus of machine translation errors extracted from translation students exercises. In *LREC*, pages 3585–3588.

Daniel Zeman, Mark Fishel, Jan Berka, and Ondřej Bojar. 2011. Addicter: what is wrong with my translations? *The Prague Bulletin of Mathematical Linguistics*, 96:79–88.

Zaixiang Zheng, Shujian Huang, Zhaopeng Tu, Xin-Yu Dai, and Jiajun Chen. 2019. Dynamic past and future for neural machine translation.

# Towards Understanding Neural Machine Translation with Word Importance

**Shilin He**[1,2] **Zhaopeng Tu**[3*] **Xing Wang**[3] **Longyue Wang**[3] **Michael R. Lyu**[1,2] **Shuming Shi**[3]
[1]Department of Computer Science and Engineering, The Chinese University of Hong Kong
[2]Shenzhen Research Institute, The Chinese University of Hong Kong
[1,2]{slhe,lyu}@cse.cuhk.edu.hk
[3]Tencent AI Lab
[3]{zptu,brightxwang,vinnylywang,shumingshi}@tencent.com

## Abstract

Although neural machine translation (NMT) has advanced the state-of-the-art on various language pairs, the interpretability of NMT remains unsatisfactory. In this work, we propose to address this gap by focusing on understanding the input-output behavior of NMT models. Specifically, we measure the word importance by attributing the NMT output to every input word through a gradient-based method. We validate the approach on a couple of perturbation operations, language pairs, and model architectures, demonstrating its superiority on identifying input words with higher influence on translation performance. Encouragingly, the calculated importance can serve as indicators of input words that are under-translated by NMT models. Furthermore, our analysis reveals that words of certain syntactic categories have higher importance while the categories vary across language pairs, which can inspire better design principles of NMT architectures for multi-lingual translation.

## 1 Introduction

Neural machine translation (NMT) has achieved the state-of-the-art results on a mass of language pairs with varying structural differences, such as English-French (Bahdanau et al., 2014; Vaswani et al., 2017) and Chinese-English (Hassan et al., 2018). However, so far not much is known about how and why NMT works, which pose great challenges for debugging NMT models and designing optimal architectures.

The understanding of NMT models has been approached primarily from two complementary perspectives. The first thread of work aims to understand the importance of representations by analyzing the linguistic information embedded in representation vectors (Shi et al., 2016; Belinkov et al., 2017) or hidden units (Bau et al., 2019; Ding et al., 2017). Another direction focuses on understanding the importance of input words by interpreting the input-output behavior of NMT models. Previous work (Alvarez-Melis and Jaakkola, 2017) treats NMT models as black-boxes and provides explanations that closely resemble the attention scores in NMT models. However, recent studies reveal that attention does not provide meaningful explanations since the relationship between attention scores and model output is unclear (Jain and Wallace, 2019).

In this paper, we focus on the second thread and try to open the black-box by exploiting the gradients in NMT generation, which aims to estimate the word importance better. Specifically, we employ the *integrated gradients* method (Sundararajan et al., 2017) to attribute the output to the input words with the integration of first-order derivatives. We justify the gradient-based approach via quantitative comparison with black-box methods on a couple of perturbation operations, several language pairs, and two representative model architectures, demonstrating its superiority on estimating word importance.

We analyze the linguistic behaviors of words with the importance and show its potential to improve NMT models. First, we leverage the word importance to identify input words that are under-translated by NMT models. Experimental results show that the gradient-based approach outperforms both the best black-box method and other comparative methods. Second, we analyze the linguistic roles of identified important words, and find that words of certain syntactic categories have higher importance while the categories vary across language. For example, nouns are more important for Chinese⇒English translation, while prepositions are more important for English-French and -Japanese translation. This finding can inspire bet-

* Zhaopeng Tu is the corresponding author. Work was mainly done when Shilin He was interning at Tencent AI Lab.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 953–962, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

ter design principles of NMT architectures for different language pairs. For instance, a better architecture for a given language pair should consider its own language characteristics.

**Contributions** Our main contributions are:

- Our study demonstrates the necessity and effectiveness of exploiting the intermediate gradients for estimating word importance.
- We find that word importance is useful for understanding NMT by identifying under-translated words.
- We provide empirical support for the design principle of NMT architectures: essential inductive bias (e.g., language characteristics) should be considered for model design.

## 2 Related Work

**Interpreting Seq2Seq Models** Interpretability of Seq2Seq models has recently been explored mainly from two perspectives: interpreting internal representations and understanding input-output behaviors. Most of the existing work focus on the former thread, which analyzes the linguistic information embeded in the learned representations (Shi et al., 2016; Belinkov et al., 2017; Yang et al., 2019) or the hidden units (Ding et al., 2017; Bau et al., 2019). Several researchers turn to expose systematic differences between human and NMT translations (Läubli et al., 2018; Schwarzenberg et al., 2019), indicating the linguistic properties worthy of investigating. However, the learned representations may depend on the model implementation, which potentially limit the applicability of these methods to a broader range of model architectures. Accordingly, we focus on understanding the input-output behaviors, and validate on different architectures to demonstrate the universality of our findings.

Concerning interpreting the input-output behavior, previous work generally treats Seq2Seq models as black-boxes (Li et al., 2016; Alvarez-Melis and Jaakkola, 2017). For example, Alvarez-Melis and Jaakkola (2017) measure the relevance between two input-output tokens by perturbing the input sequence. However, they do not exploit any intermediate information such as gradients, and the relevance score only resembles attention scores. Recently, Jain and Wallace (2019) show that attention scores are in weak correlation with the feature importance. Starting from this observation, we exploit the intermediate gradients to better estimate word importance, which consistently outperforms its attention counterpart across model architectures and language pairs.

**Exploiting Gradients for Model Interpretation** The intermediate gradients have proven to be useful in interpreting deep learning models, such as NLP models (Mudrakarta et al., 2018; Dhamdhere et al., 2019) and computer vision models (Selvaraju et al., 2017; Sundararajan et al., 2017). Among all gradient-based approaches, the integrated gradients (IG, Sundararajan et al., 2017) is appealing since it does not need any instrumentation of the architecture and can be computed easily by calling gradient operations. In this work, we employ the IG method to interpret NMT models and reveal several interesting findings, which can potentially help debug NMT models and design better architectures for specific language pairs.

## 3 Approach

### 3.1 Neural Machine Translation

In machine translation task, a NMT model $F$: $\mathbf{x} \rightarrow \mathbf{y}$ maximizes the probability of a target sequence $\mathbf{y} = \{y_1, ..., y_N\}$ given a source sentence $\mathbf{x} = \{x_1, ..., x_M\}$:

$$P(\mathbf{y}|\mathbf{x}; \boldsymbol{\theta}) = \prod_{n=1}^{N} P(y_n|\mathbf{y}_{<n}, \mathbf{x}; \boldsymbol{\theta})$$

where $\boldsymbol{\theta}$ is the model parameter and $\mathbf{y}_{<n}$ is a partial translation. At each time step $n$, the model generates an output word of the highest probability based on the source sentence $\mathbf{x}$ and the partial translation $\mathbf{y}_{<n}$. The training objective is to minimize the negative log-likelihood loss on the training corpus. During the inference, beam search is employed to decode a more optimal translation. In this study, we investigate the contribution of each input word $x_m$ to the translated sentence $\mathbf{y}$.

### 3.2 Word Importance

In this work, the notion of "word importance" is employed to quantify the contribution that a word in the input sentence makes to the NMT generations. We categorize the methods of word importance estimation into two types: *black-box* methods without the knowledge of the model and *white-box* methods that have access to the model

internal information (e.g., parameters and gradients). Previous studies mostly fall into the former type, and in this study, we investigate several representative black-box methods:

- *Content Words*: In linguistics, all words can be categorized as either content or content-free words. Content words consist mostly of nouns, verbs, and adjectives, which carry descriptive meanings of the sentence and thereby are often considered as important.

- *Frequent Words*: We rank the relative importance of input words according to their frequency in the training corpus. We do not consider the top 50 most frequent words since they are mostly punctuation and stop words.

- *Causal Model* (Alvarez-Melis and Jaakkola, 2017): Since the causal model is complicated to implement and its scores closely resemble attention scores in NMT models. In this study, we use *Attention* scores to simulate the causal model.

Our approach belongs to the white-box category by exploiting the intermediate gradients, which will be described in the next section.

### 3.3 Integrated Gradients

In this work, we resort to a gradient-based method, integrated gradients (Sundararajan et al., 2017) (IG), which was originally proposed to attribute the model predictions to input features. It exploits the handy model gradient information by integrating first-order derivatives. IG is implementation invariant and does not require neural models to be differentiable or smooth, thereby is suitable for complex neural networks like Transformer. In this work, we use IG to estimate the word importance in an input sentence precisely.

Formally, let $\mathbf{x} = (x_1, ..., x_M)$ be the input sentence and $\mathbf{x}'$ be a baseline input. $F$ is a well-trained NMT model, and $F(\mathbf{x})_n$ is the model output (i.e., $P(y_n|\mathbf{y}_{<n}, \mathbf{x})$) at time step $n$. Integrated gradients is then defined as the integral of gradients along the straightline path from the baseline $\mathbf{x}'$ to the input $\mathbf{x}$. In detail, the contribution of the $m^{th}$ word in $\mathbf{x}$ to the prediction of $F(\mathbf{x})_n$ is defined as follows.

$$IG^n_m(\mathbf{x}) = (\mathbf{x}_m - \mathbf{x}'_m) \int_{\alpha=0}^{1} \frac{\partial F(\mathbf{x}' + \alpha(\mathbf{x} - \mathbf{x}'))_n}{\partial \mathbf{x}_m} \mathrm{d}\alpha$$

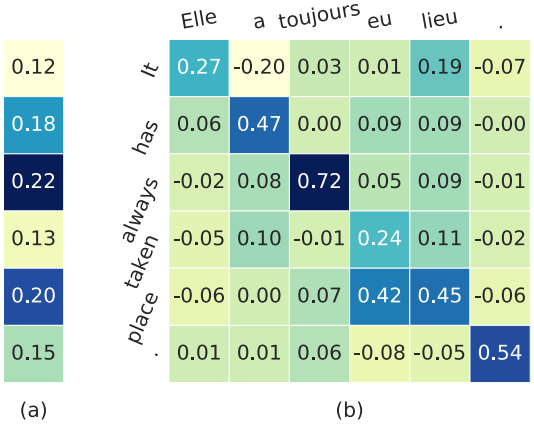


Figure 1: An example of (a) word importance and (b) contribution matrix calculated by *Attribution* (integrated gradients) on English⇒French translation task. Input in English: "It has always taken place ." Output in French: "Elle a toujours eu lieu ."

where $\frac{\partial F(\mathbf{x})_n}{\partial \mathbf{x}_m}$ is the gradient of $F(\mathbf{x})_n$ w.r.t. the embedding of the $m^{th}$ word. In this paper, as suggested, the baseline input $\mathbf{x}'$ is set as a sequence of zero embeddings that has the same sequence length $M$. In this way, we can compute the contribution of a specific input word to a designated output word. Since the above formula is intractable for deep neural models, we approximate it by summing the gradients along a multi-step path from baseline $\mathbf{x}'$ to the input $\mathbf{x}$.

$$IG^n_m(\mathbf{x}) = \frac{(\mathbf{x}_m - \mathbf{x}'_m)}{S} \sum_{k=0}^{S} \frac{\partial F(\mathbf{x}' + \frac{k}{S}(\mathbf{x} - \mathbf{x}'))_n}{\partial \mathbf{x}_m}$$

where $S$ denotes the number of steps that are uniformly distributed along the path. The IG will be more accurate if a larger S is used. In our preliminary experiments, we varied the steps and found 300 steps yielding fairly good performance.

Following the formula, we can calculate the contribution of every input word makes to every output word, forming a contribution matrix of size $M \times N$, where $N$ is the output sentence length. Given the contribution matrix, we can obtain the *word importance* of each input word to the entire output sentence. To this end, for each input word, we first aggregate its contribution values to all output words by the *sum* operation, and then normalize all sums through the *Softmax* function. Figure 1 illustrates an example of the calculated word importance and the contribution matrix, where an English sentence is translated into a French sentence using the Transformer model. A negative

contribution value indicates that the input word has negative effects on the output word.

## 4 Experiment

**Data** To make the conclusion convincing, we first choose two large-scale datasets that are publicly available, i.e., Chinese-English and English-French. Since English, French, and Chinese all belong to the subject-verb-object (SVO) family, we choose another very different subject-object-verb (SOV) language, Japanese, which might bring some interesting linguistic behaviors in English-Japanese translation.

For Chinese-English task, we use WMT17 Chinese-English dataset that consists of 20.6M sentence pairs. For English-French task, we use WMT14 English-French dataset that comprises 35.5M sentence pairs. For English-Japanese task, we follow (Morishita et al., 2017) to use the first two sections of WAT17 English-Japanese dataset that consists of 1.9M sentence pairs. Following the standard NMT procedure, we adopt the standard byte pair encoding (BPE) (Sennrich et al., 2016) with 32K merge operations for all language pairs. We believe that these datasets are large enough to confirm the rationality and validity of our experimental analyses.

**Implementation** We choose the state-of-the-art Transformer (Vaswani et al., 2017) model and the conventional RNN-Search model (Bahdanau et al., 2014) as our test bed. We implement the *Attribution* method based on the Fairseq-py (Gehring et al., 2017) framework for the above models. All models are trained on the training corpus for 100k steps under the standard settings, which achieve comparable translation results. All the following experiments are conducted on the test dataset, and we estimate the input word importance using the model generated hypotheses.

In the following experiments, we compare IG (*Attribution*) with several black-box methods (i.e., *Content*, *Frequency*, *Attention*) as introduced in Section 3.2. In Section 4.1, to ensure that the translation performance decrease attributes to the selected words instead of the perturbation operations, we randomly select the same number of words to perturb (*Random*), which serves as a baseline. Since there is no ranking for content words, we randomly select a set of content words as important words. To avoid the potential bias introduced by randomness (i.e., *Random* and *Content*), we repeat the experiments for 10 times and report the averaged results. We calculate the *Attention* importance in a similar manner as the *Attribution*, except that the attention scores use a *max* operation due to the better performance.

**Evaluation** We evaluate the effectiveness of estimating word importance by the translation performance decrease. More specifically, unlike the usual way, we measure the decrease of translation performance when perturbing a set of important words that are of top-most word importance in a sentence. The more translation performance degrades, the more important the word is.

We use the standard BLEU score as the evaluation metric for translation performance. To make the conclusion more convincing, we conduct experiments on different types of synthetic perturbations (Section 4.1), as well as different NMT architectures and language pairs (Section 4.2). In addition, we compare with a supervised erasure method, which requires ground-truth translations for scoring word importance (Section 4.3).

### 4.1 Results on Different Perturbations

In this experiment, we investigate the effectiveness of word importance estimation methods under different synthetic perturbations. Since the perturbation on text is notoriously hard (Zhang et al., 2019) due to the semantic shifting problem, in this experiment, we investigate three types of perturbations to avoid the potential bias :

- *Deletion* perturbation removes the selected words from the input sentence, and it can be regarded as a specific instantiation of sentence compression (Cohn and Lapata, 2008).
- *Mask* perturbation replaces embedding vectors of the selected words with all-zero vectors (Arras et al., 2016), which is similar to *Deletion* perturbation except that it retains the placeholder.
- *Grammatical Replacement* perturbation replaces a word by another word of the same linguistic role (i.e., POS tags), yielding a sentence that is grammatically correct but semantically nonsensical (Chomsky and Lightfoot, 2002; Gulordava et al., 2018), such as "*colorless green ideas sleep furiously*".

Figure 2 illustrates the experimental results on Chinese⇒English translation with Transformer. It

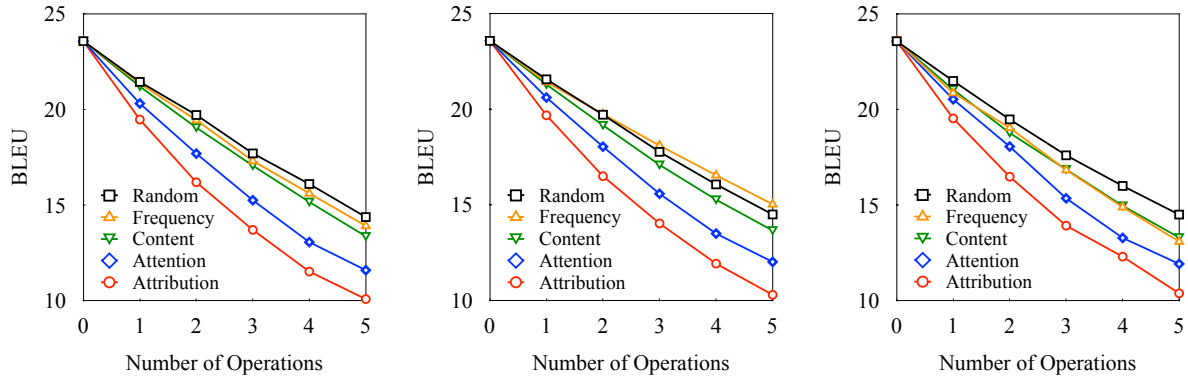


(a) **Deletion** (b) **Mask** (c) **Grammatical Replacement**

Figure 2: Effect of three types of synthetic perturbations on Chinese⇒English translation using the Transformer.

shows that *Attribution* method consistently outperforms other methods against different perturbations on a various number of operations. Here the operation number denotes the number of perturbed words in a sentence. Specifically, we can make the following observations.

**Important words are more influential on translation performance than the others.** Under three different perturbations, perturbing words of top-most importance leads to lower BLEU scores than *Random* selected words. It confirms the existence of important words, which have greater impacts on translation performance. Furthermore, perturbing important words identified by *Attribution* outperforms the *Random* method by a large margin (more than 4.0 BLEU under 5 operations).

**The gradient-based method is superior to comparative methods (e.g., Attention) in estimating word importance.** Figure 2 shows that two *black-box* methods (i.e., *Content*, *Frequency*) perform only slightly better than the *Random* method. Specifically, the *Frequency* method demonstrates even worse performances under the *Mask* perturbation. Therefore, linguistic properties (such as POS tags) and the word frequency can only partially help identify the important words, but it is not as accurate as we thought. In the meanwhile, it is intriguing to explore what exact linguistic characteristics these important words reveal, which will be introduced in Section 5.

We also evaluate the *Attention* method, which bases on the encoder-decoder attention scores at the last layer of Transformer. Note that the *Attention* method is also used to simulate the best *black-box* method SOCRAT, and the results show that it is more effective than *black-box* methods and the *Random* baseline. Given the powerful *Attention* method, *Attribution* method still achieves best performances under all three perturbations. Furthermore, we find that the gap between *Attribution* and *Attention* is notably large (around 1.0+ BLEU difference). *Attention* method does not provide as accurate word importance as the *Attribution*, which exhibits the superiority of gradient-based methods and consists with the conclusion reported in the previous study (Jain and Wallace, 2019).

In addition, as shown in Figure 2, the perturbation effectiveness of *Deletion*, *Mask*, and *Grammatical Replacement* varies from strong to weak. In the following experiments, we choose *Mask* as the representative perturbation operation for its moderate perturbation performance, based on which we compare two most effective methods *Attribution* and *Attention*.

### 4.2 Results on Different NMT Architecture and Language Pairs

**Different NMT Architecture** We validate the effectiveness of the proposed approach using a different NMT architecture RNN-Search on the Chinese⇒English translation task. The results are shown in Figure 3(a). We observe that the *Attribution* method still outperforms both *Attention* method and *Random* method by a decent margin. By comparing to Transformer, the results also reveal that the RNN-Search model is less robust to these perturbations. To be specific, under the setting of five operations and *Attribution* method, Transformer shows a relative decrease of 55% on BLEU scores while the decline of RNN-Search model is 64%.

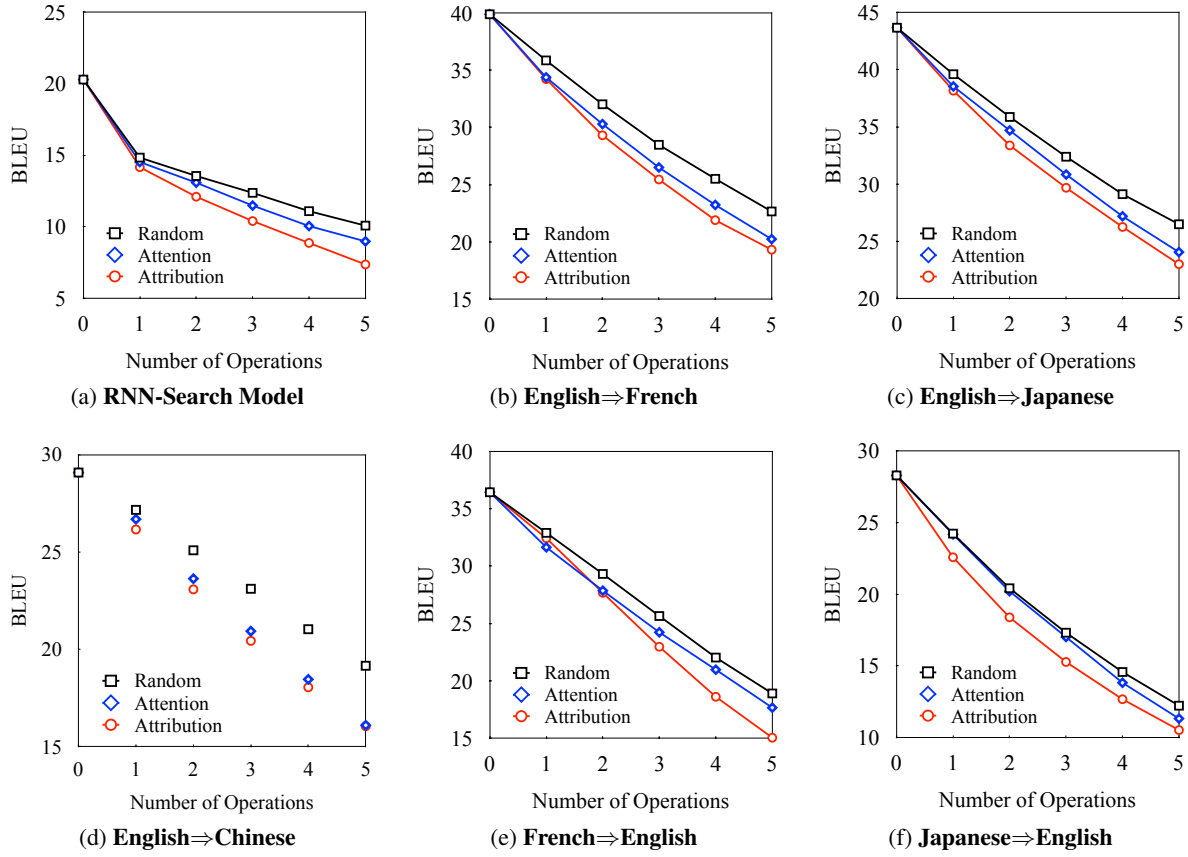


Figure 3: Effect of the *Mask* perturbation on (a) Chinese⇒English translation using the RNN-Search model, (b, c, d, e, f) other language pairs and directions using Transformer model.

**Different Language Pairs and Directions** We further conduct experiments on another two language pairs (i.e., English⇒French, English⇒Japanese in Figures 3(b, c)) as well as the reverse directions (Figures 3(d, e, f)) using Transformer under the *Mask* perturbation. In all the cases, *Attribution* shows the best performance while *Random* achieves the worst result. More specifically, *Attribution* method shows similar translation quality degradation on all three language-pairs, which declines to around the half of the original BLEU score with five operations.

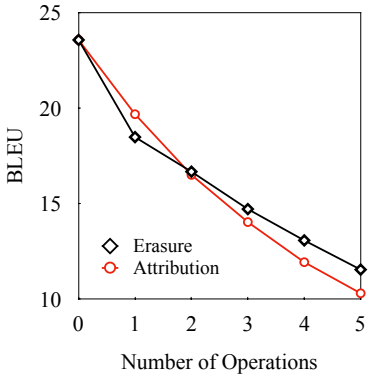


Figure 4: Effect of *Attribution* and *Erasure* methods on Chinese⇒English translation with *Mask* perturbation.

### 4.3 Comparison with Supervised Erasure

There exists another straightforward method, *Erasure* (Alvarez-Melis and Jaakkola, 2017; Arras et al., 2016; Zintgraf et al., 2017), which directly evaluates the word importance by measuring the translation performance degradation of each word. Specifically, it erases (i.e., *Mask*) one word from the input sentence each time and uses the BLEU score changes to denote the word importance (after normalization).

In Figure 4, we compare *Erasure* method with *Attribution* method under the *Mask* perturbation. The results show that *Attribution* method is less effective than *Erasure* method when only one word is perturbed. But it outperforms the *Erasure* method when perturbing 2 or more words. The results reveal that the importance calculated by erasing only one word cannot be generalized to multiple-words scenarios very well. Besides, the

| Method | Top 5% | Top 10% | Top 15% |
|---|---|---|---|
| Attention | 0.058 | 0.077 | 0.119 |
| Erasure | 0.154 | 0.170 | 0.192 |
| Attribution | **0.248** | **0.316** | **0.342** |

Table 1: F1 accuracy of detecting under-translation errors with the estimated word importance.

*Erasure* method is a supervised method which requires ground-truth references, and finding a better words combination is computation infeasible when erasing multiple words.

We close this section by pointing out that our gradient-based method consistently outperforms its black-box counterparts in various settings, demonstrating the effectiveness and universality of exploiting gradients for estimating word importance. In addition, our approach is on par with or even outperforms the supervised erasure method (on multiple-word perturbations). This is encouraging since our approach does not require any external resource and is fully unsupervised.

## 5 Analysis

In this section, we conduct analyses on two potential usages of word importance, which can help debug NMT models (Section 5.1) and design better architectures for specific languages (Section 5.2). Due to the space limitation, we only analyze the results of Chinese⇒English, English⇒French, and English⇒Japanese. We list the results on the reverse directions in Appendix, in which the general conclusions also hold.

### 5.1 Effect on Detecting Translation Errors

In this experiment, we propose to use the estimated word importance to detect the under-translated words by NMT models. Intuitively, under-translated input words should contribute little to the NMT outputs, yielding much smaller word importance. Given 500 Chinese⇒English sentence pairs translated by the Transformer model (BLEU 23.57), we ask ten human annotators to manually label the under-translated input words, and at least two annotators label each input-hypothesis pair. These annotators have at least six years of English study experience, whose native language is Chinese. Among these sentences, 178 sentences have under-translation errors with 553 under-translated words in total.

Table 1 lists the accuracy of detecting under-translation errors by comparing words of *least* importance and human-annotated under-translated words. As seen, our *Attribution* method consistently and significantly outperforms both *Erasure* and *Attention* approaches. By exploiting the word importance calculated by *Attribution* method, we can identify the under-translation errors automatically without the involvement of human interpreters. Although the accuracy is not high, it is worth noting that our under-translation method is very simple and straightforward. This is potentially useful for debugging NMT models, e.g., automatic post-editing with constraint decoding (Hokamp and Liu, 2017; Post and Vilar, 2018).

| Type | | Zh⇒En | En⇒Fr | En⇒Ja |
|---|---|---|---|---|
| POS Tags | Noun | **21.0%** | 1.9% | 0.7% |
| | Verb | 0.3% | **25.0%** | 0.3% |
| | Adj. | 0.4% | 9.3% | 0.7% |
| | Prep. | 1.3% | 4.5% | **26.7%** |
| | Dete. | 3.0% | 5.7% | 2.1% |
| | Punc. | 3.5% | **18.3%** | **30.5%** |
| | Others | 0.5% | 1.2% | 4.7% |
| Fertility | $\geq 2$ | **50.2%** | **21.4%** | **21.7%** |
| | 1 | **15.4%** | 7.0% | 3.1% |
| | $(0, 1)$ | 2.5% | 0.4% | 3.0% |
| | 0 | 0.0% | 1.9% | 3.8% |
| Syntactic | Low | 1.6% | 2.5% | 1.2% |
| | Middle | 0.3% | 0.8% | 1.4% |
| | High | 0.0% | 0.1% | 0.1% |

Table 2: Correlation between *Attribution* word importance with POS tags, Fertility, and Syntactic Depth. Fertility can be categorized into 4 types: one-to-many ("$\geq 2$"), one-to-one ("1"), many-to-one ("$(0, 1)$"), and null-aligned ("0"). Syntactic depth shows the depth of a word in the dependency tree. A lower tree depth indicates closer to the root node in the dependency tree, which might indicate a more important word.

### 5.2 Analysis on Linguistic Properties

In this section, we analyze the linguistic characteristics of important words identified by the attribution-based approach. Specifically, we investigate several representative sets of linguistic properties, including POS tags, and fertility, and depth in a syntactic parse tree. In these analyses, we multiply the word importance with the corresponding sentence length for fair comparison. We use a decision tree based regression model to calculate the correlation between the importance and linguistic properties.

| Type | | Chinese⇒English | | | English⇒French | | | English⇒Japanese | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Count | Attri. | △ | Count | Attri. | △ | Count | Attri. | △ |
| Content | Noun | 0.383 | 0.407 | *+6.27%* | 0.341 | 0.355 | *+4.11%* | 0.365 | 0.336 | *-7.95%* |
| | Verb | 0.165 | 0.160 | *-3.03%* | 0.146 | 0.131 | *-10.27%* | 0.127 | 0.123 | *-3.15%* |
| | Adj. | 0.032 | 0.029 | *-9.38%* | 0.076 | 0.072 | *-5.26%* | 0.094 | 0.088 | *-6.38%* |
| | Total | 0.579 | 0.595 | *+2.76%* | 0.563 | 0.558 | *-0.89%* | 0.587 | 0.547 | *-6.81%* |
| Content-Free | Prep. | 0.056 | 0.051 | *-8.93%* | 0.120 | 0.132 | *+10.00%* | 0.129 | 0.151 | *+17.05%* |
| | Dete. | 0.043 | 0.043 | *0.00%* | 0.102 | 0.101 | *-0.98%* | 0.112 | 0.103 | *-8.04%* |
| | Punc. | 0.137 | 0.131 | *-4.38%* | 0.100 | 0.091 | *-9.00%* | 0.096 | 0.120 | *+25.47%* |
| | Others | 0.186 | 0.179 | *-3.76%* | 0.115 | 0.118 | *+2.61%* | 0.076 | 0.079 | *+3.95%* |
| | Total | 0.421 | 0.405 | *-3.80%* | 0.437 | 0.442 | *+1.14%* | 0.413 | 0.453 | *+9.69%* |

Table 3: Distribution of syntactic categories (e.g. content words vs. content-free words) based on word count ("Count") and *Attribution* importance ("Attri."). "△" denotes relative change over the count-based distribution.

| Fertility | Chinese⇒English | | | English⇒French | | | English⇒Japanese | | |
|---|---|---|---|---|---|---|---|---|---|
| | Count | Attri. | △ | Count | Attri. | △ | Count | Attri. | △ |
| $\geq 2$ | 0.087 | 0.146 | *+67.82%* | 0.126 | 0.138 | *+9.52%* | 0.117 | 0.143 | *+22.22%* |
| 1 | 0.621 | 0.622 | *+0.16%* | 0.672 | 0.670 | *-0.30%* | 0.570 | 0.565 | *-0.88%* |
| $(0, 1)$ | 0.115 | 0.081 | *-29.57%* | 0.116 | 0.113 | *-2.59%* | 0.059 | 0.055 | *-6.78%* |
| 0 | 0.176 | 0.150 | *-14.77%* | 0.086 | 0.079 | *-8.14%* | 0.254 | 0.237 | *-6.69%* |

Table 4: Distributions of word fertility and their relative change based on *Attribution* importance and word count.

Table 2 lists the correlations, where a higher value indicates a stronger correlation. We find that the syntactic information is almost independent of the word importance value. Instead, the word importance strongly correlates with the POS tags and fertility features, and these features in total contribute over 95%. Therefore, in the following analyses, we mainly focus on the POS tags (Table 3) and fertility properties (Table 4). For better illustration, we calculate the distribution over the linguistic property based on both the *Attribution* importance ("Attr.") and the word frequency ("Count") inside a sentence. The larger the relative increase between these two values, the more important the linguistic property is.

**Certain syntactic categories have higher importance while the categories vary across language pairs.** As shown in Table 3, *content* words are more important on Chinese⇒English but *content-free* words are more important on English⇒Japanese. On English⇒French, there is no notable increase or decrease of the distribution since English and French are in essence very similar. We also obtain some specific findings of great interest. For example, we find that noun is more important on Chinese⇒English translation, while preposition is more important on English⇒French translation. More interestingly, English⇒Japanese translation shows a substantial discrepancy in contrast to the other two language pairs. The results reveal that preposition and punctuation are very important in English⇒Japanese translation, which is counter-intuitive.

Punctuation in NMT is understudied since it carries little information and often does not affect the understanding of a sentence. However, we find that punctuation is important on English⇒Japanese translation, whose proportion increases dramatically. We conjecture that it is because the punctuation could affect the sense groups in a sentence, which further benefits the syntactic reordering in Japanese.

**Words of high fertility are always important.** We further compare the fertility distribution based on word importance and the word frequency on three language pairs. We hypothesize that a source word that corresponds to multiple target words should be more important since it contributes more to both sentence length and BLEU score.

Table 4 lists the results. Overall speaking, one-to-many fertility is consistently more important on all three language pairs, which confirms our hypothesis. On the contrary, null-aligned words receive much less attention, which shows a persis-

tently decrease on three language pairs. It is also reasonable since null-aligned input words contribute almost nothing to the translation outputs.

## 6 Discussion and Conclusion

We approach understanding NMT by investigating the word importance via a gradient-based method, which bridges the gap between word importance and translation performance. Empirical results show that the gradient-based method is superior to several black-box methods in estimating the word importance. Further analyses show that important words are of distinct syntactic categories on different language pairs, which might support the viewpoint that essential inductive bias should be introduced into the model design (Strubell et al., 2018). Our study also suggests the possibility of detecting the notorious under-translation problem via the gradient-based method.

This paper is an initiating step towards the general understanding of NMT models, which may bring some potential improvements, such as

- *Interactive MT and Constraint Decoding* (Foster et al., 1997; Hokamp and Liu, 2017): The model pays more attention to the detected unimportant words, which are possibly under-translated;
- *Adaptive Input Embedding* (Baevski and Auli, 2019): We can extend the adaptive softmax (Grave et al., 2017) to the input embedding of variable capacity – more important words are assigned with more capacity;
- *NMT Architecture Design*: The language-specific inductive bias (e.g., different behaviors on POS) should be incorporated into the model design.

We can also explore other applications of word importance to improve NMT models, such as more tailored training methods. In general, model interpretability can build trust in model predictions, help error diagnosis and facilitate model refinement. We expect our work could shed light on the NMT model understanding and benefit the model improvement.

There are many possible ways to implement the general idea of exploiting gradients for model interpretation. The aim of this paper is not to explore this whole space but simply to show that some fairly straightforward implementations work well. Our approach can benefit from advanced exploitation of the gradients or other useful intermediate information, which we leave to the future work.


## Acknowledgement

Shilin He and Michael R. Lyu were supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (No. CUHK 14210717 of the General Research Fund), and Microsoft Research Asia (2018 Microsoft Research Asia Collaborative Research Award). We thank the anonymous reviewers for their insightful comments and suggestions.


## References


David Alvarez-Melis and Tommi Jaakkola. 2017. A causal framework for explaining the predictions of black-box sequence-to-sequence models. In *EMNLP*.

Leila Arras, Franziska Horn, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. 2016. Explaining predictions of non-linear classifiers in nlp. In *Proceedings of the 1st Workshop on Representation Learning for NLP*.

Alexei Baevski and Michael Auli. 2019. Adaptive input representations for neural language modeling. In *ICLR*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Anthony Bau, Yonatan Belinkov, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2019. Identifying and controlling important neurons in neural machine translation. In *ICLR*.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017. What do neural machine translation models learn about morphology? In *ACL*.

Noam Chomsky and David W Lightfoot. 2002. *Syntactic structures*. Walter de Gruyter.

Trevor Cohn and Mirella Lapata. 2008. Sentence compression beyond word deletion. In *COLING*.

Kedar Dhamdhere, Mukund Sundararajan, and Qiqi Yan. 2019. How important is a neuron? In *ICLR*.

Yanzhuo Ding, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Visualizing and understanding neural machine translation. In *ACL*.

George Foster, Pierre Isabelle, and Pierre Plamondon. 1997. Target-text mediated interactive machine translation. *Machine Translation*, 12(1/2):175–194.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *ICML*.

Édouard Grave, Armand Joulin, Moustapha Cissé, David Grangier, and Hervé Jégou. 2017. Efficient softmax approximation for GPUs. In *ICML*.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. In *NAACL*.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, et al. 2018. Achieving human parity on automatic chinese to english news translation. In *arXiv:1803.05567*.

Chris Hokamp and Qun Liu. 2017. Lexically constrained decoding for sequence generation using grid beam search. In *ACL*.

Sarthak Jain and Byron C. Wallace. 2019. Attention is not explanation. In *NAACL*.

Samuel Läubli, Rico Sennrich, and Martin Volk. 2018. Has Machine Translation Achieved Human Parity? A Case for Document-level Evaluation. In *EMNLP*.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. Understanding neural networks through representation erasure. In *arXiv preprint arXiv:1612.08220*.

Makoto Morishita, Jun Suzuki, and Masaaki Nagata. 2017. Ntt neural machine translation systems at wat 2017. In *WAT*.

Pramod Kaushik Mudrakarta, Ankur Taly, Mukund Sundararajan, and Kedar Dhamdhere. 2018. Did the model understand the question? In *ACL*.

Matt Post and David Vilar. 2018. Fast lexically constrained decoding with dynamic beam allocation for neural machine translation. In *NAACL*.

Robert Schwarzenberg, David Harbecke, Vivien Macketanz, Eleftherios Avramidis, and Sebastian Möller. 2019. Train, sort, explain: Learning to diagnose translation models. In *NAACL*.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL*.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural mt learn source syntax? In *EMNLP*.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-Informed Self-Attention for Semantic Role Labeling. In *EMNLP*.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *ICML*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*.

Baosong Yang, Longyue Wang, Derek F. Wong, Lidia S. Chao, and Zhaopeng Tu. 2019. Assessing the ability of self-attention networks to learn word order. In *ACL*.

Wei Emma Zhang, Quan Z Sheng, and Ahoud Abdulrahmn F Alhazmi. 2019. Generating textual adversarial examples for deep learning models: A survey. In *arXiv preprint arXiv:1901.06796*.

Luisa M Zintgraf, Taco S Cohen, Tameem Adel, and Max Welling. 2017. Visualizing deep neural network decisions: Prediction difference analysis. In *ICLR*.

# Multilingual Neural Machine Translation with Language Clustering

**Xu Tan**[§,*], **Jiale Chen**[†,*], **Di He**[‡], **Yingce Xia**[§], **Tao Qin**[§] and **Tie-Yan Liu**[§]
[§]Microsoft Research
[†]University of Science and Technology of China
[‡]Peking University
{xuta,taoqin,tyliu}@microsoft.com

## Abstract

Multilingual neural machine translation (NMT), which translates multiple languages using a single model, is of great practical importance due to its advantages in simplifying the training process, reducing online maintenance costs, and enhancing low-resource and zero-shot translation. Given there are thousands of languages in the world and some of them are very different, it is extremely burdensome to handle them all in a single model or use a separate model for each language pair. Therefore, given a fixed resource budget, e.g., the number of models, how to determine which languages should be supported by one model is critical to multilingual NMT, which, unfortunately, has been ignored by previous work. In this work, we develop a framework that clusters languages into different groups and trains one multilingual model for each cluster. We study two methods for language clustering: (1) using prior knowledge, where we cluster languages according to language family, and (2) using language embedding, in which we represent each language by an embedding vector and cluster them in the embedding space. In particular, we obtain the embedding vectors of all the languages by training a universal neural machine translation model. Our experiments on 23 languages show that the first clustering method is simple and easy to understand but leading to suboptimal translation accuracy, while the second method sufficiently captures the relationship among languages well and improves the translation accuracy for almost all the languages over baseline methods.

## 1 Introduction

Neural machine translation (NMT) (Bahdanau et al., 2015; Luong et al., 2015b; Sutskever et al., 2014; Wu et al., 2016; Gehring et al., 2017; Vaswani et al., 2017) has witnessed rapid progress in recent years, from novel model structure developments (Gehring et al., 2017; Vaswani et al., 2017) to achieving performance comparable to humans (Hassan et al., 2018).

Although a conventional NMT model can handle a single language translation pair (e.g., German→English, Spanish→French) well, training a separate model for each language pair is unaffordable considering there are thousands of languages in the world. A straightforward solution to reduce computational cost is using one model to handle the translations of multiple languages, i.e., multilingual translation. Johnson et al. (2017); Firat et al. (2016); Ha et al. (2016); Lu et al. (2018) propose to share part of (e.g., attention mechanism) or all models for multiple language pairs and achieve considerable accuracy improvement. While they focus on how to translate multiple language pairs in a single model and improve the performance of the multilingual model, they do not investigate which language pairs should be trained in the same model.

Clearly, it is extremely heavy to translate all language pairs in a single model due to the diverse and large amount of languages; instead we cluster language pairs into multiple clusters and train one model for each cluster, considering that: (1) language pairs that differ a lot (e.g., German→English and Chinese→English) may negatively impact the training process if handling them by one model, and (2) similar language pairs (e.g., German→English and French→English) are likely to boost each other in model training. Then the key challenge is how to cluster language pairs, which is our focus in this paper.

In this paper, we consider the many-to-one settings where there are multiple source languages and one target language (English), and one-to-

[*]Authors contribute equally to this work.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 963–973, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

many settings where there are multiple target languages and one source language (English)[1]. In this way, we only need to consider the languages in the source or target side for the determination of training in the same model, instead of language pairs.

We consider two methods for language clustering. The first one is clustering based on prior knowledge, where we use the knowledge of language family to cluster languages. The second one is a purely learning based method that uses language embeddings for similarity measurement and clustering. We obtain the language embeddings by training all the languages in a universal NMT model, and add each language with a tag to give the universal model a sense of which language it currently processes. The tag of each language (language embedding) is learned end-to-end and used for language clustering. Language clustering based on language family is easy to obtain and understand, while the end-to-end learning method may boost multilingual NMT training, since the language clustering is derived directly from the learning task and will be more useful to the task itself.

Through empirical studies on 23 languages→English and English→23 languages in IWSLT dataset, we have several findings: (1) Language embeddings can capture the similarity between languages, and correlate well with the fine-grained hierarchy (e.g., language branch) in a language family. (2) Similar languages, if clustered together, can indeed boost multilingual performance. (3) Language embeddings based clustering outperforms the language family based clustering on the 23 languages↔English.

## 2 Background

**Neural Machine Translation** Given the bilingual translation pair $(x, y)$, an NMT model learns the parameter $\theta$ by maximizing the log-likelihood $\log P(y|x, \theta)$. The encoder-decoder framework (Bahdanau et al., 2015; Luong et al., 2015b; Sutskever et al., 2014; Wu et al., 2016; Gehring et al., 2017; Vaswani et al., 2017; Shen et al., 2018; He et al., 2018) is adopted to model the conditional probability $P(y|x, \theta)$, where the encoder maps the input to a set of hidden representations $h$ and the decoder generates each target token $y_t$ using the previous generated tokens $y_{<t}$ and the representations $h$.

**Multilingual NMT** NMT has recently been extended from the translation of a language pair to multilingual translation (Dong et al., 2015; Luong et al., 2015a; Firat et al., 2016; Lu et al., 2018; Johnson et al., 2017; He et al., 2019; Ha et al., 2016; Platanios et al., 2018; Tan et al., 2019). Dong et al. (2015) use a single encoder but separate decoders to translate one source language to multiple target languages (i.e., one-to-many translation). Luong et al. (2015a) combine multiple encoders and decoders, one encoder for each source language and one decoder for each target language respectively, to translate multiple source languages to multiple target languages (i.e., many-to-many translation). Firat et al. (2016) use different encoders and decoders but share the attention mechanism for many-to-many translation. Similarly, Lu et al. (2018) propose the neural interlingua, which is an attentional LSTM encoder to link multiple encoders and decoders for different language pairs. Johnson et al. (2017); Ha et al. (2016) use a universal encoder and decoder to handle multiple source and target languages, with a special tag in the encoder to determine which target languages to output.

While all the above works focus on the design of better models for multilingual translation, they implicitly assume that a set of languages are pre-given, and do not consider which languages (language pairs) should be in a set and share one model. In this work we focus on determining which languages (language pairs) should be shared in one model.

## 3 Multilingual NMT with Language Clustering

Previous work trains a set of language pairs (usually the number is small, e.g., $< 10$ languages) using a single model and focuses on improving this multilingual model. When facing a large amount of languages (dozens or even hundreds), one single model can hardly handle them all, considering that a single model has limited capacity, some languages are quite diverse and far different, and thus one model will lead to degraded accuracy.

In this paper, we first group the languages into several clusters and then train one multilingual NMT model (Johnson et al., 2017) for the translations in each cluster. By controlling the number of

[1]Many-to-many translation can be bridged through many-to-one and one-to-many translations. Our methods can be also extended to the many-to-many setting with some modifications. We leave this to future work.

clusters, we can balance translation accuracy and computational cost: compared to using one model for each language pair, our approach greatly reduces computational cost, and compared to using one model for all language pairs, our approach delivers better translation accuracy. In this section, we present two methods of language clustering to enhance multilingual NMT.

### 3.1 Prior Knowledge Based Clustering

Several kinds of prior knowledge can be leveraged to cluster languages (Paul et al., 2009; Comrie, 1989; Bakker et al., 2009; Holman et al., 2008; Liu and Li, 2010; Chen and Gerdes, 2017; Wu et al., 2018; Leng et al., 2019), such as language family, language typology or other language characteristics from the URIEL database (Levin et al., 2017). Here we do not aim to give a comprehensive study of diverse prior knowledge based clusterings. Instead, we choose the commonly used language family taxonomy (Paul et al., 2009) as the prior knowledge for clustering, to provide a comparison with the language embedding based clustering.

Language family is a group of languages related through descent from a common ancestral language or parental language[2]. There are different kinds of taxonomies for language family in the world, among which, Ethnologue (Paul et al., 2009)[3] is one of the most authoritative and commonly accepted taxonomies. The 7,472 known languages in the world fall into 152 families according to Paul et al. (2009). We regard the languages in the same family as similar languages and group them into one cluster.

### 3.2 Language Embedding Based Clustering

When training a multilingual model, it is common practice to add a tag to the input of encoder to indicate which language the model is currently processing (Johnson et al., 2017; Firat et al., 2016; Malaviya et al., 2017; Tiedemann and Östling, 2017). The embeddings of the tag is learned end-to-end and depicts the characteristics of the corresponding language, which is called language embeddings, analogous to word embeddings (Mikolov et al., 2013).

We first train a universal model to translate all the language pairs. As shown in Figure 1, the encoder of the universal NMT model takes both word embeddings and language embedding as inputs. After model training, we get an embedding vector for each language. We regard the embedding vector as the representation of a language and cluster all languages using the embeddings[4]. There exist many clustering methods. Without loss of generality, we choose hierarchical clustering (Rokach and Maimon, 2005) to cluster the embedding vectors in our experiments.

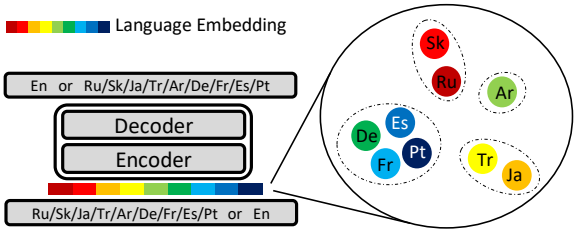


Figure 1: The illustration of learning language embeddings for clustering. For both many-to-one (other languages to English) and one-to-many (English to other languages) setting, we add the language embeddings to the encoder.

### 3.3 Discussions

We analyze and compare the two clustering methods proposed in this section. Clustering based on prior knowledge (language family) is simple. The taxonomy based on language family is consistent with the human knowledge, easy to understand, and does not change with respect to data/time. This method also has drawbacks. First, the taxonomy built on language family does not cover all the languages in the world since some languages are isolate[5]. Second, there are many language families (152 according to Paul et al. (2009)), which means that we still need a large number of models to handle all the languages in the world. Third, language family cannot characterize all the features of a language to fully capture the similarity between languages.

Since language embeddings are learnt in a universal NMT model, which is consistent with the

[2]https://en.wikipedia.org/wiki/Language_family
[3]https://www.ethnologue.com/browse/families

[4]One may argue that considering training one universal model for all the languages can not ensure the best translation accuracy, how could the learned language embeddings be accurate? We find the learned language embeddings are not that sensitive to translation accuracy and relatively stable during the later training process even when the universal model has not converged to the best translation accuracy. Therefore, the cost to get the language embeddings is much smaller than training a multilingual model to get good accuracy.
[5]https://en.wikipedia.org/wiki/Language_isolate

downstream multilingual NMT task, clustering based on language embeddings is supposed to capture the similarity between different languages well for NMT. As is shown in our experiments, the clustering results implicitly capture and combine multiple aspects of language characteristics and boost the performance of the multilingual model.

## 4 Experiment Setup

**Datasets** We evaluate our method on the IWSLT datasets which contain multiple languages from TED talks. We collect datasets from the IWSLT evaluation campaign[6] from years 2011 to 2018, which consist of the translation pairs of 23 languages↔English. The detailed description of the training/validation/test set of the 23 translation pairs can be found in Supplementary Materials (Section 1). All the data has been tokenized and segmented into sub-word symbols using Byte Pair Encoding (BPE) (Sennrich et al., 2016). We learn the BPE operations for all languages together, which results in a shared vocabulary of 90K BPE tokens.

**Model Configurations** We use Transformer (Vaswani et al., 2017) as the basic NMT model considering that it achieves state-of-the-art performance on multiple NMT benchmark tasks and is a popular choice for recent research on NMT. We use the same model configuration for each cluster with model hidden size $d_{\text{model}} = 256$, feed-forward hidden size $d_{\text{ff}} = 1024$ and the layer number is 2. The size of language embeddings is set to 256.

**Training and Inference** We up-sample the training data of each language to be roughly the same during training. For the multilingual model training, we concatenate training sentence pairs of all the languages in the same mini-batch. We set the batch size of each language to roughly 4096 tokens, and thus the total batch size is $4096 * |C_k|$, where $|C_k|$ is the number of languages in cluster $k$. The corresponding language embedding is added on the word embedding of each source token. We train the model with 8 NVIDIA Tesla M40 GPU cards, each GPU card with roughly $512 * |C_k|$ tokens in terms of batch size. We use Adam optimizer (Kingma and Ba, 2014) with $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\varepsilon = 10^{-9}$ and follow the learning rate schedule in Vaswani et al. (2017).

[6]https://wit3.fbk.eu/

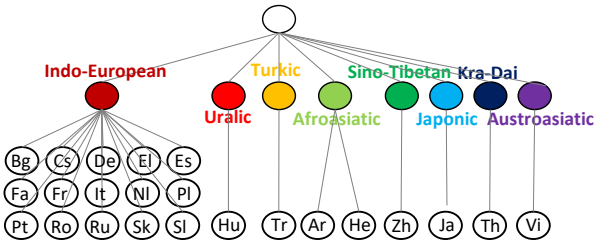


Figure 2: Language clustering of the 23 languages in IWLST dataset according to language family. There are 8 different language families in this dataset, which includes Indo-European, Uralic, Turkic, Afroasiatic, Sino-Tibetan, Japonic, Kra-Dai and Austroasiatic.

During inference, each source token is also added with the corresponding language embedding in order to give the model a sense of the language it is currently processing. We decode with beam search and set beam size to 6 and length penalty $\alpha = 1.1$ for all the languages. We evaluate the translation quality by tokenized case sensitive BLEU (Papineni et al., 2002) with multi-bleu.pl[7].

Our codes are implemented based on tensor2tensor (Vaswani et al., 2018)[8] and we will release the codes once the paper is open to the public.

## 5 Results

In this section, we mainly show the experiment results and analyses on the many-to-one setting in Section 5.1-5.3. The results on the one-to-many setting are similar and we briefly show the results in Section 5.4 due to space limitations.

### 5.1 Results of Language Clustering

The language clustering based on language family is shown in Figure 2, which results in 8 groups given our 23 languages. All the language names and their corresponding ISO-639-1 code[9] can be found in Supplementary Materials (Section 2).

We use hierarchical clustering (Rokach and Maimon, 2005)[10] method to group the languages based on language embeddings. We use the elbow method (Thorndike, 1953) to automatically decide the optimal number of clusters $K$. Note that we have tried to extract the language embeddings from multiple model checkpoints randomly

[7]https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl

[8]https://github.com/tensorflow/tensor2tensor

[9]https://www.iso.org/iso-639-language-codes.html.

[10]https://docs.scipy.org/doc/scipy-0.18.1/reference/generated/ scipy.cluster.hierarchy.linkage.html

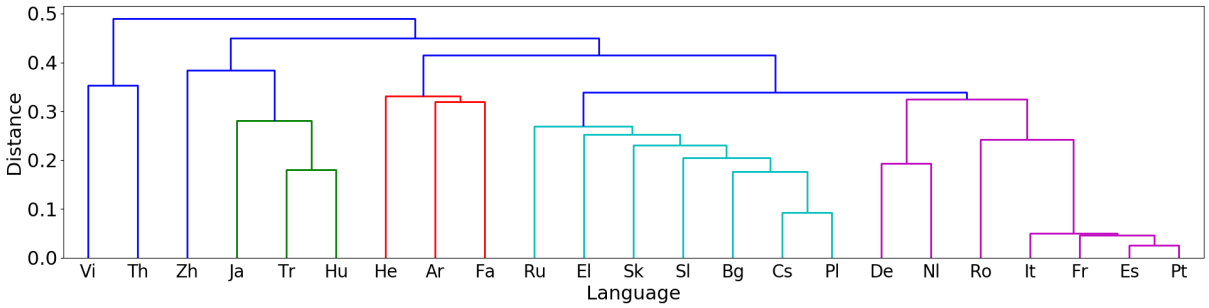


Figure 3: The hierarchical clustering based on language embeddings. The Y-axis represents the distance between two languages or clusters. Languages in the same color are divided into the same cluster, where blue color agglomerates different clusters together. If a language is marked as blue, then it forms a cluster itself. Cluster #1: Vi; Cluster #2: Th; Cluster #3: Zh; Cluster #4: Ja, Tr, Hu; Cluster #5: He, Ar, Fa; Cluster #6: Ru, El, Sk, Sl, Bg, Cs, Pl; Cluster #7: De, Nl, Ro, It, Fr, Es, Pt.

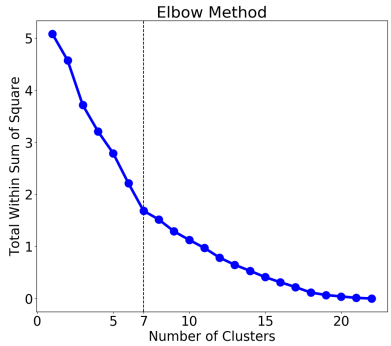


Figure 4: The optimal number of clusters determined by the elbow method for 23 languages→English based on language embeddings. The elbow method plots the curve of clustering performance (which is defined as the intra-cluster variation, i.e., the within-cluster sum of squares) according to different number of clusters, and find the location of a bend (knee) in the curve as the optimal number of clusters (7 as shown in the figure).

chosen in the later training process, and found that the clustering results based on these language embeddings are stable. Figure 3 demonstrates the clustering results based on language embeddings. Each color represents a language cluster and there are 7 clusters according to the elbow method (the details of how this method determines the optimal number of clusters are shown in Figure 7). Figure 3 clearly shows the agglomerative process of the languages and demonstrates the fine-grained relationship between languages.

We have several interesting findings from Figure 2 and 3:

- Language embeddings capture the relationship in the language family well. Cluster #7 in Figure 3 roughly covers the Germanic and Romance languages (De, Nl, Ro, It, Fr, Es, Pt) which are two different language branches in the Indo-European family but both adopt Latin alphabets. It can be further exactly divided into Germanic (De, Nl) and Romance (Ro, It, Fr, Es, Pt) branches. Cluster #6 is very close to Cluster #7 as they all belong to the Indo-European family. The difference is that the languages in Cluster #6 (Ru, El, Sk, Sl, Bg, Cs, Pl) mostly belong to Slavic branch[11].

- Language embeddings can also capture the knowledge of morphological typology (Comrie, 1989). Ja (Japanese), Tr (Turkish), Hu (Hungarian) in Cluster #4 of Figure 3 are all Synthetic-Fusional languages. Language embedding based method can cluster them together by learning their language features end-to-end with the embeddings despite they are in different language families.

- Language embeddings capture the regional, cultural, and historical influences. The languages in Cluster #5 (He, Ar, Fa) of Figure 3 are close to each other in geographical location (West Asia). Ar (Arabic) and He (Hebrew) share the same Semitic language branch in Afroasiatic language family, while Fa (Persian) has been influenced much by Arabic due to history and religion in West Asia[12]. Language embedding can implicitly learn this relationship and cluster them together.

[11]It also contains El (Greek) probably due to that most Slavic languages are influenced by Middle Greek through the Eastern Orthodox Church (Horrocks, 2009).

[12]https://en.wikipedia.org/wiki/Persian_vocabulary#Arabic_influence

| Language | Ar | Bg | Cs | De | El | Es | Fa | Fr | He | Hu | It | Ja |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *Random* | 22.90 | 32.18 | 28.88 | 30.67 | 33.28 | 28.47 | 19.16 | 24.36 | 28.01 | 20.78 | 26.08 | 10.13 |
| *Family* | 25.02 | **32.75** | 30.27 | 31.09 | 33.61 | 28.18 | 19.59 | 24.24 | 29.42 | 19.07 | 26.74 | 9.90 |
| *Embedding* | **25.27** | 32.52 | **30.97** | **31.33** | **33.67** | **28.81** | **19.64** | **25.43** | **30.03** | **21.89** | **27.10** | **11.57** |

| Language | Nl | Pl | Pt | Ro | Ru | Sk | Sl | Th | Tr | Vi | Zh |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *Random* | 33.88 | 18.34 | 31.93 | 27.64 | 17.38 | 24.22 | 15.88 | 17.94 | 18.93 | 25.93 | 14.08 |
| *Family* | 34.82 | 19.23 | 32.18 | 27.91 | 17.58 | 25.89 | **23.97** | **18.46** | 19.95 | **26.95** | **15.13** |
| *Embedding* | **35.43** | **20.04** | **32.33** | **27.97** | **18.13** | **26.61** | 22.12 | **18.46** | **22.09** | **26.95** | **15.13** |

Table 1: BLEU score of 23 languages→English with multilingual models based on different methods of language clustering: *Random*, *Family* (Language Family) and *Embedding* (Language Embedding).

These findings show that language embeddings can incorporate different prior knowledge such as language family, or some other implicit information like regional, cultural, and historical influence, and can well fuse the information together for clustering. In the next subsection, we show how our language embedding based clustering boosts the performance of multilingual NMT compared with the language family based clustering.

### 5.2 Translation Accuracy

After dividing the languages into several clusters, we train the models for the languages in each cluster with a multilingual NMT model (Johnson et al., 2017) based on different clustering methods and list the BLEU scores of each language→English in Table 1. We also show the results of random clustering (*Random*) and use the same number of clusters as the language embedding based clustering, and average the BLEU scores on multiple times of random clustering (3 times in our experiments) for comparison. It can be seen that *Random* performs worst, and language embedding (*Embedding*) based clustering outperforms the language family (*Family*) based clustering for most languages[13], demonstrating the superiority of the learning based method over prior knowledge.

We further compare the language embedding based clustering with two extreme settings: 1) Each language trained with a separate model (*Individual*), 2) All languages trained in a universal model (*Universal*), to analyze how language clustering performs on each language. As shown in Table 2, we have several observations:

- First, *Universal* in general performs the worst due to the diverse languages and limited model capacity. *Universal* only performs better on Ja (Japanese) and Sl (slovene) due to their small amount of training data, which will be discussed in the next observation.

- Second, *Embedding* outperforms *Individual* (mostly by more than 1 BLEU score) on 12 out of 23 languages[14], which demonstrates that clustering can indeed bring performance gains with fewer models, especially for languages with relatively small data size. For these languages, clustering together with similar languages may act as some kind of data augmentation, which will benefit the model training. An extreme case is that Sl (slovene) training together with all the languages gets the best accuracy as shown in Table 2, since Sl is extremely low-resource, which benefits a lot from more training data. This can be also verified by the next observation.

- Third, similar languages can help boost performance. For example, Hu (Hungarian) and Ja (Japanese) are similar as they are typical agglutinative languages in terms of morphological typology (Comrie, 1989). When they are clustered together based on *Embedding*, the performance of both languages improves (Hu: 19.07→21.89, Ja: 9.90→11.57).

### 5.3 Clustering Results with respect to Training Data Scale

We further study how the clustering results based on language embeddings change with varying training data, in order to study the robustness of

[13] *Embedding* performs worse than *Family* on only 2 languages.

[14] *Embedding* performs slightly worse (mostly within 0.5 BLEU gap) than *Individual* on 8 languages, mainly due to the plenty of training data on these languages. In this case, *Individual* with more number of models will help.

| Language | Ar | Bg | Cs | De | El | Es | Fa | Fr | He | Hu | It | Ja |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *Data size* | 180K | 140K | 110K | 180K | 180K | 180K | 70K | 180K | 150K | 90K | 180K | 90K |
| *Individual* (23) | **25.43** | **32.87** | 29.15 | **32.18** | **33.70** | **29.17** | 18.12 | **27.98** | 29.45 | 19.07 | **27.70** | 9.90 |
| *Universal* (1) | 23.26 | 32.47 | 28.86 | 30.42 | 33.56 | 28.03 | 19.45 | 23.64 | 27.29 | 21.24 | 26.07 | **12.78** |
| *Embedding* (7) | 25.27 | 32.52 | **30.97** | 31.33 | 33.67 | 28.81 | **19.64** | 25.43 | **30.03** | **21.89** | 27.10 | 11.57 |

| Language | Nl | Pl | Pt | Ro | Ru | Sk | Sl | Th | Tr | Vi | Zh |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *Data size* | 140K | 140K | 180K | 180K | 160K | 70K | 14K | 80K | 130K | 130K | 180K |
| *Individual* (23) | 34.61 | 18.19 | **32.77** | 27.88 | 17.55 | 19.72 | 4.48 | **18.46** | 19.95 | **26.95** | **15.13** |
| *Universal* (1) | 34.58 | 19.02 | 30.96 | 27.77 | 16.69 | 25.31 | **24.22** | 18.27 | 18.76 | 26.13 | 14.54 |
| *Embedding* (7) | **35.43** | **20.04** | 32.33 | **27.97** | **18.13** | **26.61** | 22.12 | **18.46** | **22.09** | **26.95** | **15.13** |

Table 2: BLEU score of 23 languages→English with different number of clusters: *Universal* (all the languages share one model), *Individual* (each language with separate models, totally 23 models)), *Embedding* (Language Embedding with 7 models). *Data size* shows the training data for each language→English.

our language embedding based clustering, which is important in the low-resource multilingual setting. We reduce the training data of each language to 50%, 20% and 5% to check how stable the language embeddings based clustering is, as shown in Figure 5. It can be seen that most languages are clustered stably when training data is relatively large (50% or even 20%), except for Sl (Slovene) which has the least training data (14K) among the 23 languages and thus reducing the training data of Sl to 50% and 20% will influence the learning of language embeddings and further influence clustering results. If further reducing the training data to 5%, more languages are influenced and clustered abnormally as most languages have less than

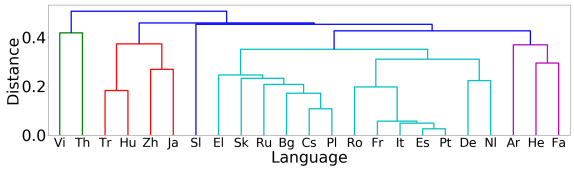


(a) 50% of training data

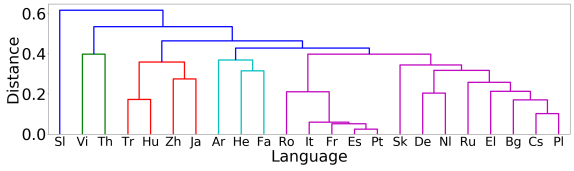


(b) 20% of training data

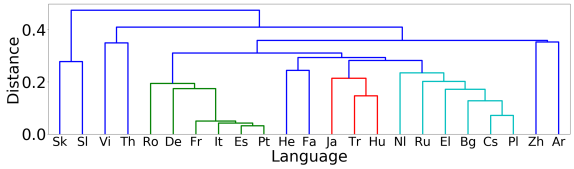


(c) 5% of training data

Figure 5: The results of language embedding based clustering with varying amount of training data.

10K training data. Even so, the similar languages such as Fr (French), It (Italian), Es (Spanish), and Pt (Portuguese) are still clustered together.

### 5.4 Results of One-to-Many Translation

We finally show experiment results of one-to-many translation, i.e., English→23 languages. We first train a universal model that covers all languages to get the language embeddings, and then perform clustering like in the many-to-one setting. The optimal number of clusters is 5, which is automatically determined by the elbow method (we show the figure of the elbow method in Figure 1 in the Supplementary Materials).

Figure 6 shows the clustering results based on language embeddings, and Table 3 shows the BLEU score of multilingual models clustered by different methods. We also have several findings:

- In one-to-many setting, language embeddings still depict the relationship among language families. Cluster #1 in Figure 6 covers the Romance (Ro, Fr, It, Es, Pt) branch in Indo-European language family while Cluster #2 contains the Germanic (De, Nl) branch in Indo-European language family. The languages in Cluster #3 (Sl, Bg, El, Ru, Sk, Cs, Pl) are mostly from the Slavic branch in Indo-European language family. We find that the cluster results of Indo-European language family is more fine-grained than many-to-one setting, which divides each language into their own branches in a language family.

- Language embedding can capture regional, cultural, and historic influence. For Cluster #4 in Figure 6, Zh (Chinese) and Ja (Japanese) are in different language families

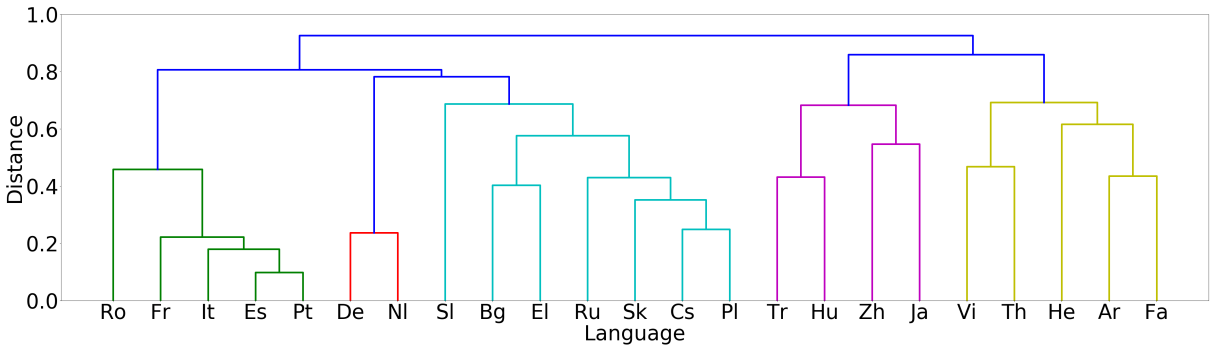


Figure 6: The hierarchical clustering based on language embedding in one-to-many setting. The Y-axis represents the distance between two languages or clusters. Languages in the same color are divided into the same cluster. Cluster #1: Ro, Fr, It, Es, Pt; Cluster #2: De, Nl; Cluster #3: Sl, Bg, El, Ru, Sk, Cs, Pl; Cluster #4: Tr, Hu, Zh, Ja; Cluster #5: Vi, Th, He, Ar, Fa.

| Language | Ar | Bg | Cs | De | El | Es | Fa | Fr | He | Hu | It | Ja |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *Universal* | 9.80 | 25.84 | 17.89 | 22.40 | 26.71 | 28.48 | 12.26 | 22.10 | 16.38 | 13.32 | 25.34 | 10.91 |
| *Individual* | **13.13** | **30.04** | 19.84 | **25.69** | **27.90** | **29.57** | 12.03 | **22.93** | **20.43** | 13.74 | **26.87** | 10.70 |
| *Family* | 13.11 | 27.54 | 19.11 | 23.74 | 27.78 | 28.88 | 13.36 | 22.78 | 20.26 | 13.74 | 26.71 | 10.70 |
| *Embedding* | 12.37 | 28.85 | **20.81** | 25.27 | 27.11 | 28.93 | **13.79** | 22.85 | 19.23 | **15.47** | 26.81 | **13.33** |

| Language | Nl | Pl | Pt | Ro | Ru | Sk | Sl | Th | Tr | Vi | Zh |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *Universal* | 26.19 | 10.10 | 28.27 | 19.72 | 8.26 | 15.52 | **15.82** | 25.06 | 9.44 | 26.87 | 9.56 |
| *Individual* | 29.86 | 11.61 | 28.09 | **21.81** | **14.11** | 14.47 | 6.61 | 27.41 | 11.40 | 28.77 | **10.83** |
| *Family* | 27.83 | 10.97 | 28.63 | 21.13 | 12.80 | 16.91 | 15.73 | 27.41 | 11.40 | 28.77 | **10.83** |
| *Embedding* | **29.98** | **11.95** | **28.83** | 21.14 | 13.84 | **18.18** | 14.25 | **28.55** | **12.11** | **29.79** | 10.52 |

Table 3: BLEU score of English→23 languages with multilingual models based on different methods of language clustering: *Universal* (all the languages share one model), *Individual* (each language with separate model), *Family* (Language Family), *Embedding* (Language Embedding).

but influence each other through the history and culture. Th (Thai) and Vi (Vietnamese) in Cluster #5 that are close to each other in geographical location can be captured by their language embeddings.

These findings show that language embeddings can stably characterize relationships among languages no matter in one-to-many setting or in many-to-one setting. Although there exist some differences in the clustering results between one-to-many setting and many-to-one setting due to the nature of the learning based clustering method, most of the clustering results are reasonable. Our experiment results shown in Table 3 demonstrate that the language embedding can provide more reasonable clustering than prior knowledge, and thus result in higher BLEU score for most of the languages.

We also find from Table 3 that each language with separate model (*Individual*) perform best on nearly 10 languages, due to the abundant model capacity. However, there are still several languages on which our clustering methods (*Family* or *Embedding*) largely outperform *Individual*. The reason is similar as the many-to-one setting that some languages are of smaller data sizes. In this situation, similar languages clustered together will boost the accuracy compared with separate model.

## 6 Conclusion

In this work, we have studied language clustering for multilingual neural machine translation. Experiments on 23 languages→English and English→23 languages show that language embeddings can sufficiently characterize the similarity between languages and outperform prior knowledge (language family) for language clustering in terms of the BLEU scores.

For future work, we will test our methods for many-to-many translation. We will consider more languages (hundreds or thousands) to study our methods in larger scale setting. We will also study

| Language | Ar | Bg | Cs | De | El | Es | Fa | Fr | He | Hu | It | Ja |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Data size | 180K | 140K | 110K | 180K | 180K | 180K | 70K | 180K | 150K | 90K | 180K | 90K |
| Language | Nl | Pl | Pt | Ro | Ru | Sk | Sl | Th | Tr | Vi | Zh | |
| Data size | 140K | 140K | 180K | 180K | 160K | 70K | 14K | 80K | 130K | 130K | 180K | |

Table 4: The size of training data for 23 language↔English in our experiments.

| Language | Arabic | Bulgarian | Czech | German | Greek | Spanish | Persian | French | Hebrew |
|---|---|---|---|---|---|---|---|---|---|
| ISO code | Ar | Bg | Cs | De | El | Es | Pt | Fr | He |
| *Language* | Hungarian | Italian | Japanese | Dutch | Polish | Portuguese | Romanian | Russian | Slovak |
| ISO code | Hu | It | Ja | Nl | Pl | Pt | Ro | Ru | Sk |
| *Language* | Slovenian | Thai | Turkish | Vietnamese | Chinese | English | | | |
| ISO code | Sl | Th | Tr | Vi | Zh | En | | | |

Table 5: The ISO 639-1 code of each language in our experiments.

how to obtain language embeddings from monolingual data, which will make our method scalable to cover those languages with little or without bilingual data. On the other hand, we will also consider other pre-training methods (Song et al., 2019) for multilingual and low-resource NMT.

## A Dataset Description

We evaluate our experiments on IWSLT datasets. We collect 23 languages↔English translation pairs form IWSLT evaluation campaign from year 2011 to 2018. The training data sizes of each languages are shown in Table 4 and we use the default validation and test set for each language pairs. For Bg (Bulgarian), El (Greek), Hu (Hungarian) and Ja (Japanese), there are no available validation and test data in IWSLT, we randomly split 1K sentence pairs from the corresponding training set as the validation and test data respectively.

## B Language Name and Code

The language names and their corresponding language codes according to ISO 639-1 standard are listed in Table 5.

## C The Elbow Method for One-to-Many Setting

The optimal number of clusters is 5 in one-to-many setting, which is automatically determined by the elbow method as shown in Figure 7.

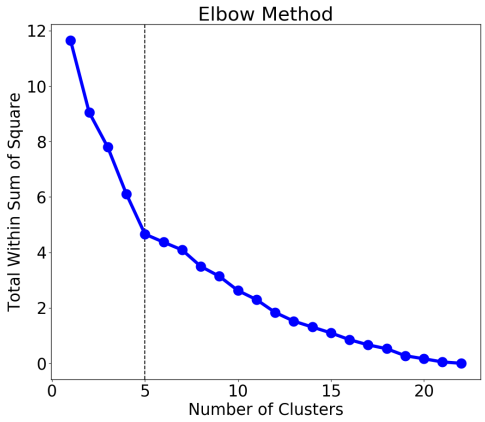


Figure 7: The optimal number of clusters determined by the elbow method for English→23 languages based on language embeddings. The detailed description of the elbow method can be seen in Figure 4 in the main paper.

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *ICLR 2015*.

Dik Bakker, André Müller, Viveka Velupillai, Søren Wichmann, Cecil H Brown, Pamela Brown, Dmitry Egorov, Robert Mailhammer, Anthony Grant, and Eric W Holman. 2009. Adding typology to lexicostatistics: A combined approach to language classification. *Linguistic Typology*, 13(1):169–181.

Xinying Chen and Kim Gerdes. 2017. Classifying languages by dependency structure. typologies of delexicalized universal dependency treebanks. In *Proceedings of the Fourth International Conference on Dependency Linguistics (Depling 2017), September 18-20, 2017, Università di Pisa, Italy*, 139, pages 54–63. Linköping University Electronic Press.

Bernard Comrie. 1989. *Language universals and linguistic typology: Syntax and morphology*. University of Chicago press.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1723–1732.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 866–875.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 1243–1252.

Thanh-Le Ha, Jan Niehues, and Alexander H. Waibel. 2016. Toward multilingual neural machine translation with universal encoder and decoder. *CoRR*, abs/1611.04798.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, Shujie Liu, Tie-Yan Liu, Renqian Luo, Arul Menezes, Tao Qin, Frank Seide, Xu Tan, Fei Tian, Lijun Wu, Shuangzhi Wu, Yingce Xia, Dongdong Zhang, Zhirui Zhang, and Ming Zhou. 2018. Achieving human parity on automatic chinese to english news translation. *CoRR*, abs/1803.05567.

Tianyu He, Jiale Chen, Xu Tan, and Tao Qin. 2019. Language graph distillation for low-resource machine translation. *arXiv preprint arXiv:1908.06258*.

Tianyu He, Xu Tan, Yingce Xia, Di He, Tao Qin, Zhibo Chen, and Tie-Yan Liu. 2018. Layer-wise coordination between encoder and decoder for neural machine translation. In *Advances in Neural Information Processing Systems*, pages 7944–7954.

Eric W Holman, Søren Wichmann, Cecil H Brown, Viveka Velupillai, André Müller, and Dik Bakker. 2008. Explorations in automated language classification. *Folia Linguistica*, 42(3-4):331–354.

Geoffrey Horrocks. 2009. *Greek: A History of the Language and its Speakers*. John Wiley & Sons.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda B. Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *TACL*, 5:339–351.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Yichong Leng, Xu Tan, Tao Qin, Xiang-Yang Li, and Tie-Yan Liu. 2019. Unsupervised pivot translation for distant languages. *arXiv preprint arXiv:1906.02461*.

Lori S. Levin, Patrick Littell, David R. Mortensen, Ke Lin, Katherine Kairis, and Carlisle Turner. 2017. URIEL and lang2vec: Representing languages as typological, geographical, and phylogenetic vectors. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers*, pages 8–14.

Haitao Liu and Wenwen Li. 2010. Language clusters based on linguistic complex networks. *Chinese Science Bulletin*, 55(30):3458–3465.

Yichao Lu, Phillip Keung, Faisal Ladhak, Vikas Bhardwaj, Shaonan Zhang, and Jason Sun. 2018. A neural interlingua for multilingual machine translation. *CoRR*, abs/1804.08198.

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015a. Multi-task sequence to sequence learning. *CoRR*, abs/1511.06114.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015b. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1412–1421.

Chaitanya Malaviya, Graham Neubig, and Patrick Littell. 2017. Learning language representations for typology prediction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2529–2535.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA.*, pages 311–318.

Lewis M Paul, Gary F Simons, Charles D Fennig, et al. 2009. Ethnologue: Languages of the world. *Dallas, TX: SIL International. Available online at www. ethnologue. com/. Retrieved June*, 19:2011.

Emmanouil Antonios Platanios, Mrinmaya Sachan, Graham Neubig, and Tom Mitchell. 2018. Contextual parameter generation for universal neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 425–435.

Lior Rokach and Oded Maimon. 2005. Clustering methods. In *Data mining and knowledge discovery handbook*, pages 321–352. Springer.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Yanyao Shen, Xu Tan, Di He, Tao Qin, and Tie-Yan Liu. 2018. Dense information flow for neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1294–1303.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. Mass: Masked sequence to sequence pre-training for language generation. *arXiv preprint arXiv:1905.02450*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Xu Tan, Yi Ren, Di He, Tao Qin, and Tie-Yan Liu. 2019. Multilingual neural machine translation with knowledge distillation. In *International Conference on Learning Representations*.

Robert L Thorndike. 1953. Who belongs in the family? *Psychometrika*, 18(4):267–276.

Jörg Tiedemann and Robert Östling. 2017. Continuous multilinguality with language vectors. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers*, pages 644–649.

Ashish Vaswani, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan N. Gomez, Stephan Gouws, Llion Jones, Łukasz Kaiser, Nal Kalchbrenner, Niki Parmar, Ryan Sepassi, Noam Shazeer, and Jakob Uszkoreit. 2018. Tensor2tensor for neural machine translation. *CoRR*, abs/1803.07416.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6000–6010.

Lijun Wu, Xu Tan, Di He, Fei Tian, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2018. Beyond error propagation in neural machine translation: Characteristics of language also matter. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3602–3611.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

# Don't Forget the Long Tail! A Comprehensive Analysis of Morphological Generalization in Bilingual Lexicon Induction

**Paula Czarnowska**[1] **Sebastian Ruder**[2*] **Edouard Grave**[3]
**Ryan Cotterell**[1] **Ann Copestake**[1]
[1]University of Cambridge [2]National University of Ireland [3]Facebook AI Research
pjc211@cam.ac.uk sebastian@ruder.io egrave@fb.com
rdc42@cam.ac.uk aac10@cam.ac.uk

## Abstract

Human translators routinely have to translate rare inflections of words—due to the Zipfian distribution of words in a language. When translating from Spanish, a good translator would have no problem identifying the proper translation of a statistically rare inflection such as *hablarámos*. Note the lexeme itself, *hablar*, is relatively common. In this work, we investigate whether state-of-the-art bilingual lexicon inducers are capable of learning this kind of generalization. We introduce 40 *morphologically complete* dictionaries in 10 languages[1] and evaluate three of the state-of-the-art models on the task of translation of less frequent morphological forms. We demonstrate that the performance of state-of-the-art models drops considerably when evaluated on infrequent morphological inflections and then show that adding a simple morphological constraint at training time improves the performance, proving that the bilingual lexicon inducers can benefit from better encoding of morphology.

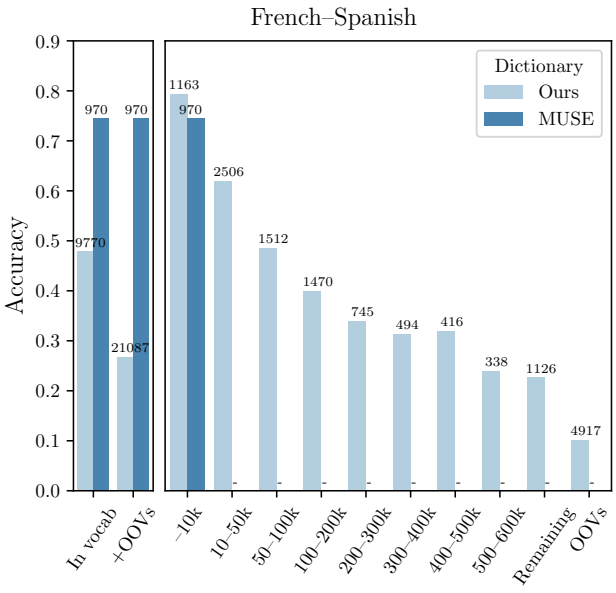


Figure 1: The relation between the BLI performance and the frequency of source words in the test dictionary. The graph presents results for the model of Ruder et al. (2018) evaluated on both the MUSE dictionary (Conneau et al., 2018) and our morphologically complete dictionary, which contains many rare morphological variants of words. The numbers above the bars correspond to the number of translated source words (a hyphen represents an empty dictionary).

## 1 Introduction

Human translators exhibit remarkable generalization capabilities and are able to translate even rare inflections they may have never seen before. Indeed, this skill is necessary for translation since language follows a Zipfian distribution (Zipf, 1949): a large number of the tokens in a translated text will come from rare types, including rare inflections of common lexemes. For instance, a Spanish translator will most certainly know the verb *hablar* "to speak", but they will only have seen the less frequent, first-person plural future form *hablarámos* a few times. Nevertheless, they would have no problem translating the latter. In this paper we ask whether current methods for bilingual lexicon induction (BLI) generalize morphologically as humans do. Generalization to rare and novel words is arguably the main point of BLI as a task—most frequent translation pairs are already contained in digital dictionaries. Modern word embeddings encode character-level knowledge (Bojanowski et al., 2017), which should—in principle—enable the models to learn this behaviour; but morphological generalization has never been directly tested.

Most existing dictionaries used for BLI evaluation do not account for the full spectrum of linguistic properties of language. Specifically, as we demonstrate in §2, they omit *most morphological inflections* of even common lexemes. To enable a more thorough evaluation we introduce a new resource: 40 **morphologically complete** dictionar-

[1]The dictionaries are available at https://github.com/pczarnowska/morph_dictionaries.

[*]Sebastian is now affiliated with DeepMind.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 974–983, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

ies for 5 Slavic and 5 Romance languages, which contain the inflectional paradigm of every word they hold. Much like with a human translator, we expect a BLI model to competently translate full paradigms of lexical items. Throughout this work we place our focus on genetically-related language pairs. This not only allows us to cleanly map one morphological inflection onto another, but also provides an upper bound for the performance on the generalization task; if the models are not able to generalize for closely related languages they would most certainly be unable to generalize when translating between unrelated languages.

We use our dictionaries to train and evaluate three of the best performing BLI models (Artetxe et al., 2016, 2017; Ruder et al., 2018) on all 40 language pairs. To paint a complete picture of the models' generalization ability we propose a new experimental paradigm in which we independently control for four different variables: the word form's frequency, morphology, the lexeme frequency and the lexeme (a total of 480 experiments). Our comprehensive analysis reveals that BLI models can generalize for frequent morphosyntactic categories, even of infrequent lexemes, but fail to generalize for the more rare categories. This yields a more nuanced picture of the known deficiency of word embeddings to underperform on infrequent words (Gong et al., 2018). Our findings also contradict the strong empirical claims made elsewhere in the literature (Artetxe et al., 2017; Conneau et al., 2018; Ruder et al., 2018; Grave et al., 2018b), as we observe that performance severely degrades when the evaluation includes rare morphological variants of a word and infrequent lexemes. We picture this general trend in Figure 1, which also highlights the skew of existing dictionaries towards more frequent words.[2] As our final contribution, we demonstrate that better encoding of morphology is indeed beneficial: enforcing a simple morphological constraint yields consistent performance improvements for all Romance language pairs and many of the Slavic language pairs.

## 2 Morphological Dictionaries

### 2.1 Existing Dictionaries

Frequent word forms can often be found in human-curated dictionaries. Thus, the practical purpose of training a BLI model should be to create translations of new and less common forms, not present in the existing resources. In spite of this, most ground truth lexica used for BLI evaluation contain mainly frequent word forms. Many available resources are restricted to the top 200k most frequent words; this applies to the English–Italian dictionary of Dinu et al. (2015), the English–German and English–Finnish dictionaries of Artetxe et al. (2017), and Artetxe et al. (2018a)'s English–Spanish resource. The dictionaries of Irvine and Callison-Burch (2017) contain only the top most frequent 10k words for each language. Zhang et al. (2017) extracted their Spanish–English and Italian–English lexica from Open Multilingual WordNet (Bond and Paik, 2012), a resource which only yields high frequency, lemma level mappings. Another example is the recent MUSE dataset (Conneau et al., 2018), which was generated using an "internal translation tool", and in which the majority of word pairs consist of forms ranked in the top 10k of the vocabularies of their respective languages.

| Source Word | Target Word | Source Lemma | Target Lemma | Tag |
|---|---|---|---|---|
| morza | moře | morze | moře | N;NOM;PL |
| morzu | moři | morze | moře | N;DAT;SG |
| morze | moře | morze | moře | N;NOM;SG |
| morzami | moři | morze | moře | N;INS;PL |
| mórz | moř | morze | moře | N;GEN;PL |
| morzu | moři | morze | moře | N;ESS;SG |
| morzom | mořím | morze | moře | N;DAT;PL |

Table 1: An example extract from our morphologically complete Polish–Czech dictionary.

Another problem associated with existing resources is 'semantic leakage' between train and evaluation sets. As we demonstrate in §2.3, it is common for a single lexeme to appear in both train and test dictionary—in the form of different word inflections. This circumstance is undesirable in evaluation settings as it can lead to performance overstatements—a model can 'memorize' the corresponding target lemma, which ultimately reduces the translation task to a much easier task of finding the most appropriate inflection. Finally, most of the available BLI resources include English in each language pair and, given how morphologically impoverished English is, those resources are unsuitable for analysis of morphological generalization.

### 2.2 Our Dictionaries

To address the shortcomings of the existing evaluation, we built 40 new **morphologically complete**

[2]For MUSE, we only evaluate on forms that are not present in our training dictionary (970 out of 1500 source words).

| **Slavic** | Czech | Russian | Slovak | Ukrainian | **Romance** | Spanish | Catalan | Portuguese | Italian |
|---|---|---|---|---|---|---|---|---|---|
| Polish | 53,353 | 128,638 | 14,517 | 12,361 | French | 686,139 | 381,825 | 486,575 | 705,800 |
| Czech | - | 65,123 | 10,817 | 8,194 | Spanish | - | 343,780 | 476,543 | 619,174 |
| Russian | | - | 128,638 | 10,554 | Catalan | | - | 261,016 | 351,609 |
| Slovak | | | - | 3,434 | Portuguese | | | - | 468,945 |

Table 2: The sizes of our morphologically complete dictionaries for Slavic and Romance language families. We present the sizes for 20 *base* dictionaries. We further split those to obtain 40 train, development and test dictionaries—one for each mapping direction to ensure the correct source language lemma separation.

dictionaries, which contain most of the inflectional paradigm of every word they contain. This enables a more thorough evaluation and makes the task much more challenging than traditional evaluation sets. In contrast to the existing resources our dictionaries consist of many rare forms, some of which are out-of-vocabulary for large-scale word embeddings such as FASTTEXT. Notably, this makes them the only resource of this kind that enables evaluating *open-vocabulary* BLI.

We focus on pairs of genetically-related languages for which we can cleanly map one morphological inflection onto another.[3] We selected 5 languages from the Slavic family: Polish, Czech, Russian, Slovak and Ukrainian, and 5 Romance languages: French, Spanish, Italian, Portuguese and Catalan. Table 1 presents an example extract from our resource; every source–target pair is followed by their corresponding lemmata and a shared tag.

We generated our dictionaries automatically based on openly available resources: Open Multilingual WordNet (Bond and Paik, 2012) and Extended Open Multilingual WordNet[4] (Bond and Foster, 2013), both of which are collections of lexical databases which group words into sets of synonyms (synsets), and UniMorph[5] (Kirov et al., 2016)—a resource comprised of inflectional word paradigms for 107 languages, extracted from Wiktionary[6] and annotated according to the UniMorph schema (Sylak-Glassman, 2016). For each language pair $(L1, L2)$ we first generated lemma translation pairs by mapping all $L1$ lemmata to all $L2$ lemmata for each synset that appeared in both $L1$ and $L2$ WordNets.[7] We then filtered out the pairs which contained lemmata not present in UniMorph and generated inflected entries from the remaining pairs: one entry for each tag that appears in the UniMorph paradigms of both lemmata.[8] The sizes of dictionaries vary across different language pairs and so does the POS distribution. In particular, while Slavic dictionaries are dominated by nouns and adjectives, verbs constitute the majority of pairs in Romance dictionaries. We report the sizes of the dictionaries in Table 2. In order to prevent semantic leakage, discussed in §2.1, for each language pair we split the initial dictionary into train, development and test splits so that each sub-dictionary has its own, independent set of lemmata. In our split, the train dictionary contains 60% of all lemmata, while the development and test dictionaries each have 20% of the lemmata.

### 2.3 Comparison with MUSE

In this section we briefly outline important differences between our resource and the MUSE dictionaries (Conneau et al., 2018) for Portuguese, Italian, Spanish, and French (12 dictionaries in total). We focus on MUSE as it is one of the few openly available resources that covers genetically-related language pairs.

**Word Frequency** The first and most prominent difference lies in the skew towards frequent word forms in MUSE evaluation. While our test dictionaries contain a representative sample of forms in

[3] One may translate *talked*, the past tense of *talk*, into many different Spanish forms, but the Portuguese *falavam* has, arguably, only one best Spanish translation: *hablaban*.

[4] We used the union of the extended and the original version of the multilingual WordNet since the latter included entries that were not present in the extended version.

[5] https://unimorph.github.io/

[6] Wiktionary (https://en.wiktionary.org/wiki/Wiktionary:Main_Page) is a large-scale, free content multilingual dictionary.

[7] Note that with this many-to-many mapping we allow for many translations of a single word.

[8] As UniMorph annotations are not consistent across different languages (e.g. in Czech and Polish resources verbs are marked for animacy, while for other Slavic resources they lack such markings) we performed minor tag processing in order to make the tags more compatible. We also discovered that for some languages, the UniMorph resource was either incomplete (resources for Romance languages do not contain adjectives or nouns) or incorrect (Czech verb inflections), in which cases we personally scraped lemma inflections directly from Wiktionary.

lower frequency bins, the majority of forms present in MUSE are ranked in the top 10k in their respective language vocabularies. This is clearly presented in Figure 1 for the French–Spanish resource and also holds for the remaining 11 dictionaries.

**Morphological Diversity** Another difference lies in the morphological diversity of both dictionaries. The average proportion of paradigm covered for lemmata present in MUSE test dictionaries is 53% for nouns, 37% for adjectives and only 3% for verbs. We generally observe that for most lemmata the dictionaries contain *only one* inflection. In contrast, for our test dictionaries we get 97% coverage for nouns, 98% for adjectives and 67% for verbs. Note that we do not get 100% coverage as we are limited by the compatibility of source language and target language UniMorph resources.

**Train–test Paradigm Leakage** Finally, we carefully analyze the magnitude of the train–test paradigm leakage. We found that, on average 20% (299 out of 1500) of source words in MUSE test dictionaries share their lemma with a word in the corresponding train dictionary. E.g. the French–Spanish test set includes the form *perdent*—a third-person plural present indicative of *perdre* (to lose) which is present in the train set. Note that the splits we provide for our dictionaries do not suffer from any leakage as we ensure that each dictionary contains the full paradigm of every lemma.

## 3 Bilingual Lexicon Induction

The task of bilingual lexicon induction is well established in the community (Vulić and Moens, 2013; Gouws et al., 2015) and is the current standard choice for evaluation of cross-lingual word embedding models. Given a list of $N$ source language word forms $x_1, \ldots, x_N$, the goal is to determine the most appropriate translation $t_i$, for each query form $x_i$. In the context of cross-lingual embeddings, this is commonly accomplished by finding a target language word that is most similar to $x_i$ in the shared semantic space, where words' similarity is usually computed using a cosine between their embeddings. The resulting set of $(x_i, t_i)$ pairs is then compared to the gold standard and evaluated using the precision at $k$ (P@$k$) metric, where $k$ is typically set to 1, 5 or 10.[9] Throughout our evaluation we use P@1, which is equivalent to accuracy.

In our work, we focus on the supervised and semi-supervised settings in which the goal is to automatically generate a dictionary given only monolingual word embeddings and some initial, seed translations. For our experiments we selected the models of Artetxe et al. (2016), Artetxe et al. (2017) and Ruder et al. (2018)—three of the best performing BLI models, which induce a shared cross-lingual embedding space by learning an orthogonal transformation from one monolingual space to another (model descriptions are given in the supplementary material).[10] In particular, the last two employ a self-learning method in which they alternate between a mapping step and a word alignment (dictionary induction) step in an iterative manner. As we observed the same general trends across all models, in the body of the paper we only report the results for the best performing model of Ruder et al. (2018). We present the complete set of results in the supplementary material.

**Experimental Setup** We trained and evaluated all models using the Wikipedia FASTTEXT embeddings (Grave et al., 2018a). Following the existing work, for training we only used the most frequent 200k words in both source and target vocabularies. To allow for evaluation on less frequent words, in all our experiments the models search through the whole target embedding matrix at evaluation (not just the top 200k words, as is common in the literature). This makes the task more challenging, but also gives a more accurate picture of performance. To enable evaluation on the unseen word forms we generated a FASTTEXT embedding for every out-of-vocabulary (OOV) inflection of every word in WordNet that also appears in UniMorph. We built those embeddings by summing the vectors of all $n$-grams that constitute an OOV form.[11] In the OOV evaluation we append the resulting vectors to the original embedding matrices.

## 4 Morphological Generalization

We propose a novel **quadripartite analysis** of the BLI models, in which we independently control for four different variables: (i) word form frequency, (ii) morphology, (iii) lexeme frequency and (iv)

[9] Precision at $k$ represents how many times the correct translation of a source word is returned as one of its $k$ nearest neighbours in the target language.

[10] We also used our dictionaries to train and test the more recent model of Artetxe et al. (2018b) on a handful of languages and observed the same general trends.

[11] We did this within the FASTTEXT framework, using the trained *.bin* models for each of our 10 languages.

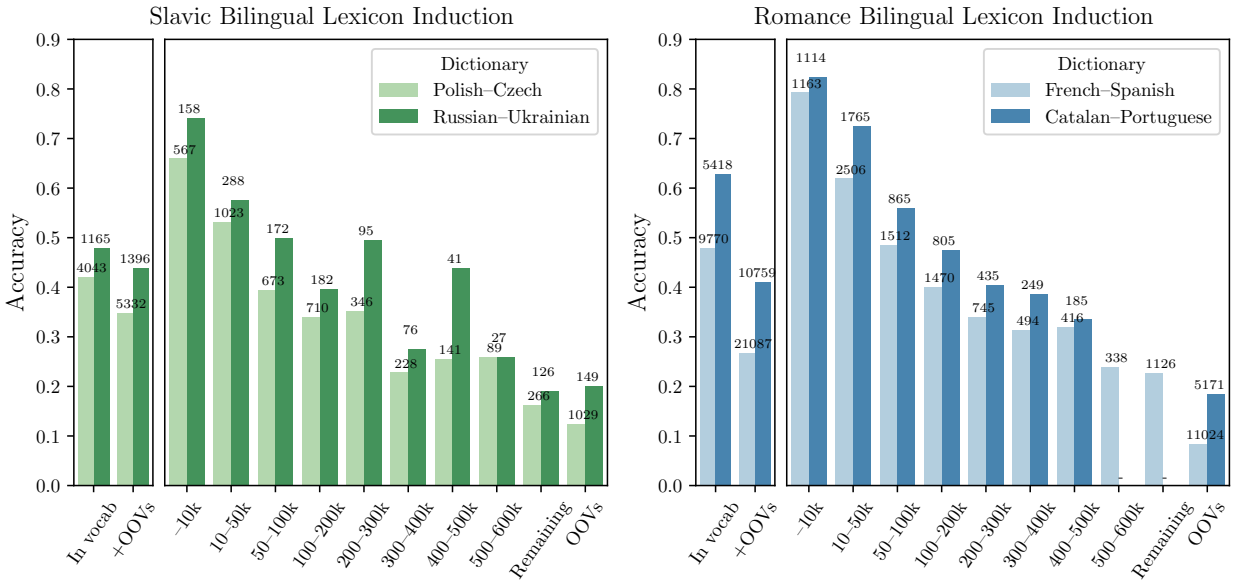


Figure 2: The relation between performance and the frequency of source words in the test dictionary for four example language pairs on the standard BLI task. The numbers above the bars correspond to the dictionary sizes.

lexeme. We provide detailed descriptions for each of those conditions in the following sections. For each condition, we analyzed all 40 language pairs for each of our selected models—a total of 480 experiments. In the body of the paper we only present *a small representative subset* of our results.

### 4.1 Controlling for Word Frequency

For highly inflectional languages, many of the infrequent types are rare forms of otherwise common lexemes and, given the morphological regularity of less frequent forms, a model that generalizes well should be able to translate those capably. Thus, to gain insight into the models' generalization ability we first examine the relation between their performance and the frequency of words in the test set.

We split each test dictionary into 9 frequency bins, based on the relative frequencies of words in the original training corpus for the word embeddings (Wikipedia in the case of FASTTEXT). More specifically, a pair appears in a frequency bin if its source word belongs to that bin, according to its rank in the respective vocabulary. We also considered unseen words that appear in the test portion of our dictionaries, but do not occur in the training corpus for the embeddings. This is a fair experimental setting since most of those OOV words are associated with known lemmata. Note that it bears a resemblance to the classic Wug Test (Berko, 1958) in which a child is introduced to a single instance of a fictitious object—'a wug'—and is asked to name two instances of the same object—'wugs'. However, in contrast to the original setup, we are interested in making sure the unseen inflection of a known lexeme is properly translated.

Figure 2 presents the results on the BLI task for four example language pairs: two from the Slavic and two from the Romance language family. The left-hand side of the plots shows the performance for the full dictionaries (with and without OOVs), while the right-hand side demonstrates how the performance changes as the words in the evaluation set become less frequent. The general trend we observe across all language pairs is an acute drop in accuracy for infrequent word forms—e.g. for Catalan–Portuguese the performance falls from 83% for pairs containing only the top 10k most frequent words to 40% for pairs, which contain source words ranked between 200k and 300k.

### 4.2 Controlling for Morphology

From the results of the previous section, it is not clear whether the models perform badly on inflections of generally *infrequent lemmata* or whether they fail on *infrequent morphosyntactic categories*, independently of the lexeme frequency. Indeed, the frequency of different morphosyntactic categories is far from uniform. To shed more light on the underlying cause of the performance drop in §4.1, we first analyze the differences in the models' performance as they translate forms belonging to different categories and, next, look at the dis-

| Tag | Accuracy | | Distribution Across Frequency Bins | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | In vocab | All | 10k | 50k | 100k | 200k | 300k | 400k | 500k | 600k | Tail | OOVs |
| French–Spanish | | | | | | | | | | | | |
| N;SG | 56.7 | 55.1 | 45% | 33% | 10% | 6% | 1% | 1% | 0% | 0% | 1% | 2% |
| N;PL | 53.6 | 49.9 | 31% | 29% | 14% | 9% | 4% | 3% | 2% | 1% | 3% | 4% |
| ADJ;MASC;SG | 62.7 | 60.9 | 29% | 40% | 16% | 7% | 3% | 1% | 1% | 0% | 1% | 2% |
| NFIN;V | 50.6 | 49.0 | 35% | 37% | 13% | 7% | 2% | 1% | 1% | 1% | 2% | 2% |
| 2;IMP;PL;POS;V | 1.5 | 2.0 | 3% | 4% | 16% | 14% | 12% | 4% | 5% | 3% | 12% | 25% |
| 3;FUT;PL;V | 34.7 | 23.3 | 0% | 4% | 8% | 14% | 9% | 8% | 8% | 5% | 16% | 27% |
| 3;COND;SG;V | 41.0 | 30.6 | 0% | 5% | 14% | 14% | 9% | 9% | 3% | 6% | 15% | 23% |
| Polish–Czech | | | | | | | | | | | | |
| INS;N;PL | 29.5 | 23.8 | 1% | 17% | 16% | 21% | 10% | 7% | 7% | 2% | 9% | 10% |
| DAT;N;SG | 26.1 | 18.7 | 13% | 15% | 10% | 12% | 7% | 4% | 3% | 3% | 8% | 24% |
| ACC;ADJ;MASC;SG | 47.8 | 47.8 | 10% | 28% | 27% | 16% | 11% | 4% | 1% | 1% | 1% | 0% |
| ACC;ADJ;FEM;SG | 47.0 | 46.4 | 4% | 29% | 27% | 25% | 6% | 4% | 3% | 1% | 0% | 0% |
| NFIN;V | 47.4 | 45.0 | 13% | 42% | 18% | 15% | 4% | 4% | 0% | 0% | 0% | 5% |
| 3;MASC;PL;PST;V | 59.4 | 51.4 | 2% | 23% | 16% | 21% | 9% | 12% | 5% | 5% | 0% | 7% |
| 2;IMP;PL;V | 11.1 | 8.1 | 0% | 0% | 0% | 14% | 0% | 7% | 14% | 7% | 3% | 55% |

Table 3: BLI results for word pairs that have a specific morphosyntactic category (left) and a distribution of those forms across different frequency bins (right).

tribution of these categories across the frequency bins.

In Table 3 we present our findings for a representative sample of morphosyntactic categories for one Slavic and one Romance language pair (we present the results for all models and all language pairs in the supplementary material).[12] It illustrates the great variability across different paradigm slots—both in terms of their frequency and the difficulty of their translation.

As expected, the performance is best for the slots belonging to the highest frequency bins and forms residing in the rarer slots prove to be more challenging. For example, for French–Spanish the performance on 2;IMP;PL;POS;V, 3;FUT;PL;V and 3;COND;SG;V[13] is notably lower than that for the remaining categories. For both language pairs, the accuracy for the second-person plural present imperative (2;IMP;PL;V) is particularly low: 1.5% accuracy for French–Spanish and 11.1% for Polish–Czech in the in-vocabulary setting. Note that it is unsurprising for an imperative form, expressing an order or command, to be infrequent in the Wikipedia corpora (the resource our monolingual embeddings were trained on). The complex distribution of the French 2;IMP;PL;POS;V across the frequency bins is likely due to syncretism—the 2;IMP;PL;POS;V paradigm slot shares a form with 2nd person present plural slot, 2;PL;PRS;V. Our hypothesis is that syncretism may have an effect on the quality of the monolingual embeddings. To our knowledge, the effect of syncretism on embeddings has not yet been systematically investigated.

### 4.3 Controlling for Lexeme Frequency

To get an even more complete picture, we inspect how the performance on translating inflections of *common lemmata* differs to translating forms coming from *less frequent paradigms* by controlling for the frequency of the lexeme. We separated our dictionaries into two bins based on the relative frequency of the source lexeme. We approximated frequency of the lexemes by using ranks of their most common inflections: our first bin contained lexemes whose most common inflection is ranked in the top 20k forms in its respective vocabulary, while the second bin consisted of lexemes with most common inflection ranked lower than 60k.[14] We present the results for the same morphosyntactic categories as in §4.2 on the left side of the graphs in Figure 3. As anticipated, in the case of less frequent lexemes the performance is generally worse than for frequent ones. However, perhaps more surprisingly, we discover that some morphosyntactic categories prove to be problematic even for the most frequent lexemes. Some

[12] We selected the morphosyntactic categories independently for each language family based on how well the models translate words of such morphology–'the good, the bad and the ugly'.

[13] 2nd person imperative plural form of a verb, 3rd person future plural form of a verb and 2nd person conditional singular form of a verb.

[14] Future work could operate on actual word counts rather than relative frequencies of words.

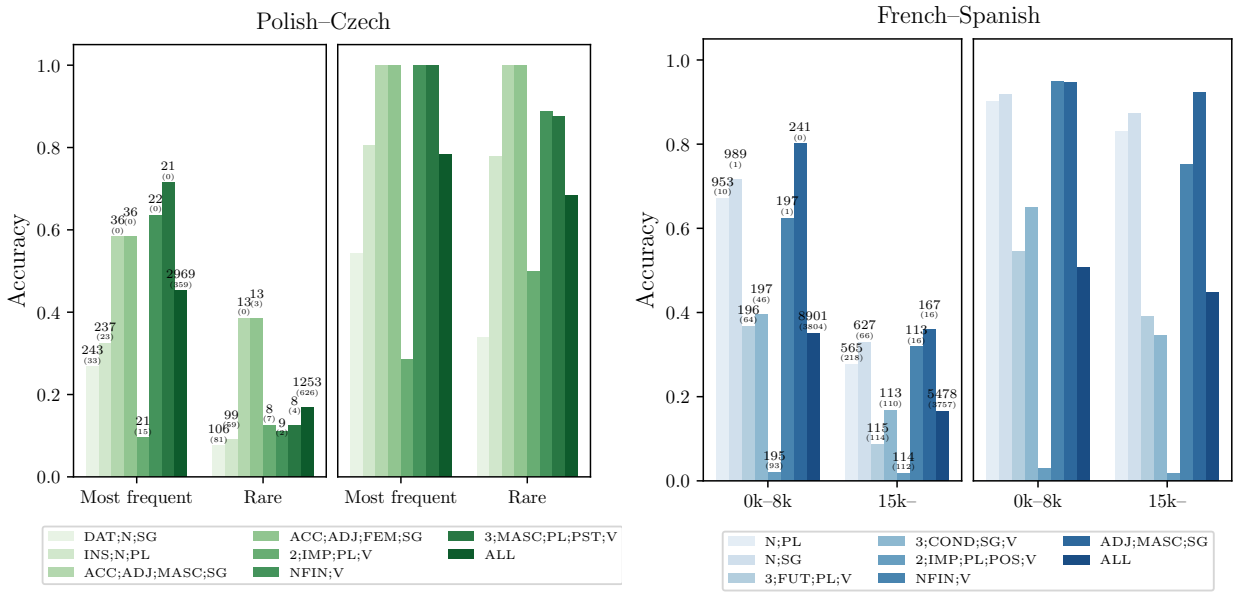


Figure 3: The performance on the standard BLI task (left side of the graphs) and the controlled for lexeme BLI (right side) for words pairs belonging to the most frequent paradigms and the infrequent paradigms. The numbers above the bars are dictionary sizes and the number of out-of-vocabulary forms in each dictionary (bracketed).

examples include the previously mentioned imperative verb form or, for Slavic languages, singular dative nouns (DAT;N;SG).

### 4.4 Controlling for Lexeme

We are, in principle, interested in the ability of the models to generalize *morphologically*. In the preceding sections we focused on the standard BLI evaluation, which given our objective is somewhat unfair to the models—they are additionally punished for not capturing lexical semantics. To gain more direct insight into the models' generalization abilities we develop a novel experiment in which the lexeme is controlled for. At test time, the BLI model is given a set of candidate translations, all of which belong to the same paradigm, and is asked to select the most suitable form. Note that the model only requires morphological knowledge to successfully complete the task—no lexical semantics is required. When mapping between closely related languages this task is particularly straightforward, and especially so in the case of FASTTEXT where a single $n$-gram, e.g. the suffix *-ing* in English as in the noun *running*, can be highly indicative of the inflectional morphology of the word.

We present results on 8 representative language pairs in Table 4 (column Lexeme). We report the accuracy on the in-vocabulary pairs as well as all the pairs in the dictionary, including OOVs. As expected, compared to standard BLI this task is much easier for the models—the performance is generally high. For Slavic languages numbers remain high even in the open-vocabulary setup, which suggests that the models can generalize morphologically. On the other hand, for Romance languages we observe a visible drop in performance. We hypothesize that this difference is due to the large quantities of verbs in Romance dictionaries; in both Slavic and Romance languages verbs have substantial paradigms, often of more than 60 forms, which makes identifying the correct form more difficult. In contrast, most words in our Slavic dictionaries are nouns and adjectives with much smaller paradigms.

Following our analysis in §4.3, we also examine how the performance on this new task differs for less and more frequent paradigms, as well as across different morphosyntactic categories. Here, we exhibit an unexpected result, which we present in the two right-hand side graphs of Figure 3: the state-of-the-art BLI models *do* generalize morphologically for frequent slots, but *do not* generalize for infrequent slots. For instance, for the Polish–Czech pair, the models achieve 100% accuracy on identifying the correct inflection when this inflection is ACC;ADJ;MASC;SG, ACC;ADJ;FEM;SG, 3;MASC;PL;PST;V or NFIN;V[15] for frequent and, for the first two categories, also the *infrequent* lex-

[15]Masculine accusative singular form of an adjective, feminine accusative singular form of an adjective, 3rd person masculine plural past form of a verb and a verbal infinitive.

| Constraint | Normal | | | | Lexeme | | | | Dictionary Sizes | |
|---|---|---|---|---|---|---|---|---|---|---|
| | In vocab | | +OOVs | | In vocab | | +OOVs | | In vocab | +OOVs |
| | ✗ | ✓ | ✗ | ✓ | ✗ | ✓ | ✗ | ✓ | | |
| Ukrainian–Russian | **68.4** | 61.1 | **63.7** | 56.1 | **89.9** | 89.1 | **88.6** | 87.6 | 786 | 933 |
| Russian–Slovak | **25.7** | 21.1 | **20.9** | 17.0 | **79.3** | 76.8 | **76.0** | 74.2 | 1610 | 2150 |
| Polish–Czech | 42.0 | **44.4** | 34.8 | **36.7** | 80.6 | **81.1** | 75.3 | **75.9** | 4043 | 5332 |
| Russian–Polish | 39.8 | **41.2** | 34.8 | **36.1** | 80.8 | **82.6** | 77.7 | **80.2** | 9183 | 11697 |
| Catalan–Portuguese | 62.8 | **64.2** | 41.1 | **42.4** | 83.1 | **84.3** | 57.7 | **59.0** | 5418 | 10759 |
| French–Spanish | 47.8 | **50.2** | 26.7 | **28.9** | 78.0 | **81.4** | 47.9 | **52.2** | 9770 | 21087 |
| Portuguese–Spanish | 60.2 | **61.1** | 36.8 | **37.6** | 84.7 | **85.4** | 57.1 | **58.2** | 9275 | 22638 |
| Italian–Spanish | 42.7 | **43.8** | 21.1 | **22.1** | 76.4 | **77.6** | 47.6 | **49.6** | 11685 | 30686 |
| Polish–Spanish | **36.1** | 32.1 | **28.0** | 25.0 | **78.1** | 77.7 | **68.6** | 68.4 | 8964 | 12759 |
| Spanish–Polish | 28.1 | **30.9** | 21.0 | **23.2** | 81.2 | **82.0** | 64.2 | **65.8** | 4270 | 6095 |

Table 4: The results on the standard BLI task and BLI controlled for lexeme for the original Ruder et al. (2018)'s model (✗) and the same model trained with a morphological constraint (✓) (discussed in §4.6).

emes; all of which are common morphosyntactic categories (see Table 3). The results from Figure 3 also demonstrate that the worst performing forms for the French–Spanish language pair are indeed the infrequent verbal inflections.

### 4.5 Experiments on an Unrelated Language Pair

So far, in our evaluation we have focused on pairs of genetically-related languages, which provided an upper bound for morphological generalization in BLI. But our experimental paradigm is not limited to related language pairs. We demonstrate this by experimenting on two example pairs of one Slavic and one Romance language: Polish–Spanish and Spanish–Polish. To construct the dictionaries we followed the procedure discussed in §2, but matched the tags based only on the features exhibited in both languages (e.g. Polish DAT;N;SG can be mapped to N;SG in Spanish, as Spanish nouns are not declined for case). Note that mapping between morphosyntactic categories of two unrelated languages is a challenging task (Haspelmath, 2010), but we did our best to address the issues specific to translation between Polish and Spanish. E.g. we ensured that Spanish imperfective/perfective verb forms can only be translated to Polish forms of imperfective/perfective verbs.

The results of our experiments are presented in the last two rows of Table 4 and, for Polish–Spanish, also in Figure 4. As expected, the BLI results on unrelated languages are generally, but not uniformly, worse than those on related language pairs. The accuracy for Spanish–Polish is particularly low, at 28% (for in vocabulary pairs). We see large variation in performance across morphosyntactic categories and more and less frequent lexemes, similar to that observed for related language pairs. In particular, we observe that 2;IMP;PL;V—the category difficult for Polish–Czech BLI is also among the most challenging for Polish–Spanish. However, one of the highest performing categories for Polish–Czech, 3;MASC;PL;PST;V, yields much worse accuracy for Polish–Spanish.

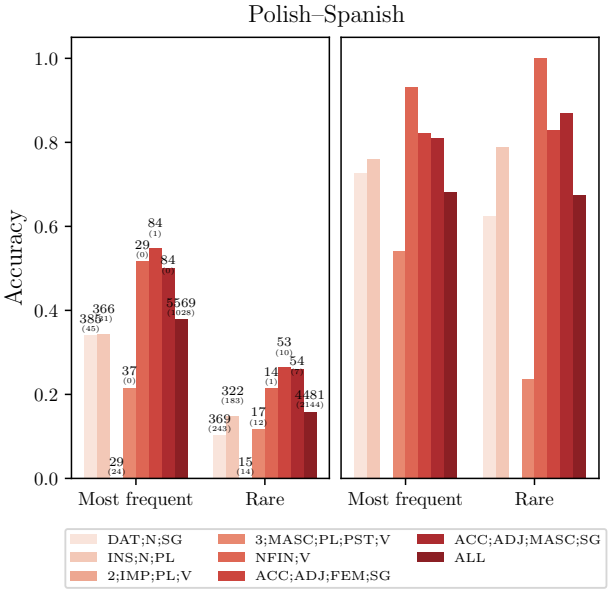


Figure 4: The results of the experiments on a pair of unrelated languages—Polish and Spanish—on the standard BLI task (left side) and the controlled for lexeme BLI (right side) for word pairs belonging to the most frequent and the infrequent paradigms.

### 4.6 Adding a Morphological Constraint

In our final experiment we demonstrate that improving morphological generalization has the potential to improve BLI results. We show that enforcing a

simple, hard morphological constraint *at training time* can lead to performance improvements at test time—both on the standard BLI task and the controlled for lexeme BLI. We adapt the self-learning models of Artetxe et al. (2017) and Ruder et al. (2018) so that at each iteration they can align two words only if they share the same morphosyntactic category. Note that this limits the training data only to word forms present in UniMorph, as those are the only ones for which we have a gold tag.[16] The results, a subset of which we present in Table 4, show that the constraint, despite its simplicity and being trained on less data, leads to performance improvements for *every* Romance language pair and many of the Slavic language pairs. We take this as evidence that properly modelling morphology will have a role to play in BLI.

## 5 Discussion and Conclusion

We conducted a large-scale evaluation of the generalization ability of the state-of-the-art bilingual lexicon inducers. To enable our analysis we created 40 morphologically complete dictionaries for 5 Slavic and 5 Romance languages and proposed a novel experimental paradigm in which we independently control for four different variables.

Our study is the first to examine morphological generalization in BLI and it reveals a nuanced picture of the interplay between performance, the word's frequency and morphology. We observe that the performance degrades when models are evaluated on less common words—even for the infrequent forms of common lexemes. Our results from the controlled for lexeme experiments suggest that models are able to generalize well for more frequent morphosyntactic categories and for part-of-speech with smaller paradigms. However, their ability to generalize decreases as the slots get less frequent and/or the paradigms get larger. Finally, we proposed a simple method to inject morphological knowledge and demonstrated that making models more morphologically aware can lead to general performance improvements.

## References


Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2289–2294.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462. Association for Computational Linguistics.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018a. Generalizing and improving bilingual word embedding mappings with a multi-step framework of linear transformations. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018b. A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 789–798. Association for Computational Linguistics.

Jean Berko. 1958. The child's learning of English morphology. *Word*, 14(2-3):150–177.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Francis Bond and Ryan Foster. 2013. Linking and extending an open multilingual wordnet. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1352–1362.

Francis Bond and Kyonghee Paik. 2012. A survey of wordnets and their licenses. In *Proceedings of the 6th Global WordNet Conference (GWC 2012)*, page 6471.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word Translation Without Parallel Data. In *Proceedings of ICLR 2018*.

Georgiana Dinu, Angeliki Lazaridou, and Marco Baroni. 2015. Improving zero-shot learning by mitigating the hubness problem. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015), workshop track, Workshop Track*.

Chengyue Gong, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2018. FRAGE: Frequency-Agnostic Word Representation. In *Proceedings of NIPS 2018*.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. BilBOWA: Fast Bilingual Distributed Representations without Word Alignments. *Proceedings of The 32nd International Conference on Machine Learning*, pages 748–756.


[16] We also experimented with more relaxed versions of the constraint, where we only used a subset of the features, but we found that for most languages the constraint worked better with more features.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018a. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Edouard Grave, Armand Joulin, and Quentin Berthet. 2018b. Unsupervised alignment of embeddings with wasserstein procrustes. *CoRR*, abs/1805.11222.

Martin Haspelmath. 2010. Comparative concepts and descriptive categories in crosslinguistic studies. *Language*, 86(3):663–687.

Ann Irvine and Chris Callison-Burch. 2017. A comprehensive analysis of bilingual lexicon induction. *Computational Linguistics*, 43(2):273–310.

Roy Jonker and Anton Volgenant. 1987. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38(4):325–340.

Christo Kirov, John Sylak-Glassman, Roger Que, and David Yarowsky. 2016. Very-large scale parsing and normalization of wiktionary morphological paradigms. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Sebastian Ruder, Ryan Cotterell, Yova Kementchedjhieva, and Anders Søgaard. 2018. A discriminative latent-variable model for bilingual lexicon induction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 458–468. Association for Computational Linguistics.

John Sylak-Glassman. 2016. The composition and use of the universal morphological feature schema (unimorph schema). Technical report, Technical report, Department of Computer Science, Johns Hopkins University.

Ivan Vulić and Marie-Francine Moens. 2013. Cross-Lingual Semantic Similarity of Words as the Similarity of Their Semantic Word Responses. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2013)*.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Adversarial training for unsupervised bilingual lexicon induction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1959–1970.

George Kingsley Zipf. 1949. Human behavior and the principle of least effort.

# Pushing the Limits of Low-Resource Morphological Inflection

**Antonios Anastasopoulos and Graham Neubig**
Language Technologies Institute, Carnegie Mellon University
{aanastas,gneubig}@cs.cmu.edu

## Abstract

Recent years have seen exceptional strides in the task of automatic morphological inflection generation. However, for a long tail of languages the necessary resources are hard to come by, and state-of-the-art neural methods that work well under higher resource settings perform poorly in the face of a paucity of data. In response, we propose a battery of improvements that greatly improve performance under such low-resource conditions. First, we present a novel two-step attention architecture for the inflection decoder. In addition, we investigate the effects of cross-lingual transfer from single and multiple languages, as well as monolingual data hallucination. The macro-averaged accuracy of our models outperforms the state-of-the-art by 15 percentage points.[1] Also, we identify the crucial factors for success with cross-lingual transfer for morphological inflection: typological similarity and a common representation across languages.

## 1 Introduction

The majority of the world's languages are categorized as synthetic, meaning that they have rich morphology, be it fusional, agglutinative, polysynthetic, or a mixture thereof. As Natural Language Processing (NLP) keeps expanding its frontiers to encompass more and more languages, modeling of the grammatical functions that guide language generation is of utmost importance.

In the case of morphologically-rich languages, explicit modeling of the inflection processes has significant potential to alleviate issues created by data scarcity and the resulting lack of vocabulary coverage. Especially on low-resource, under-represented languages and dialects, the potential for *impact* is much higher. For example, speech recognition (Foley et al., 2018) and predictive keyboards (Breiner et al., 2019) for under-represented languages, if they exist, largely still rely on unigram lexicons with performance inferior to the sophisticated language models used in high-resource ones. Good inflection models would be invaluable for predictive text technology in morphologically-rich languages as they could effectively enable proper handling of the huge vocabulary.

Additionally, they could be very useful for building educational applications for languages of under-represented communities (along with their inverse, morphological analyzers). Encouraging examples are the Yupik morphological analyzer (Schwartz et al., 2019) and the Inuktitut educational tools from the respective Native Peoples communities.[2] The social impact of such applications can be enormous, effectively raising the status of the languages slightly closer to the level of

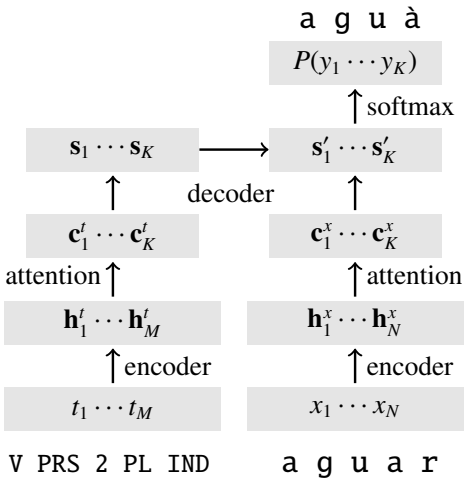


Figure 1: Visualization of our proposed two-step attention architecture. The decoder first attends over the tag sequence **T** and then uses the updated decoder state $\mathbf{s}'$ to attend over the character sequence **X** in order to produce the inflected form **Y**. (Example from Asturian.)

[1] Our code is available at https://github.com/antonisa/inflection.

[2] http://www.inuktitutcomputing.ca/



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 984–996, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

the dominant regional language.

Morphological inflection has been thoroughly studied in monolingual high resource settings, especially through the recent SIGMORPHON challenges (Cotterell et al., 2016, 2017, 2018). Low-resource settings, in contrast, are relatively underexplored. One promising direction and the main focus of the SIGMORPHON 2019 challenge (McCarthy et al., 2019b) is cross-lingual training, which has been successfully applied in other low-resource tasks such as Machine Translation (MT) or parsing.

In this work we focus in this cross-lingual setting for low-resource morphological inflection and propose several simple, yet effective approaches to mitigating problems caused by extreme lack of data which, put together, *improve accuracy by 15 percentage points over a state-of-the-art baseline*. This is achieved through the combination of a novel decoder architecture, a training regime that alleviates the need for costly structural biases that force attention monotonicity, and a data hallucination technique. We also present thorough ablations and identify the crucial factors for success with our approach.

Our system was the best performing system in the 2019 SIGMORPHON shared task on morphological inflection when evaluated on accuracy, while it ranked third when evaluated with average Levenshtein distance.

## 2 Task Definition and Approach

Morphological inflection is the process that creates grammatical forms (typically guided by sentence structure) of a lexeme/lemma. As a computational task it is framed as mapping from the lemma and a set of morphological tags to the desired form, which simplifies the task by removing the necessity to infer the form from context. For an example from Asturian, given the lemma `aguar` and tags `V;PRS;2;PL;IND`, the task is to create the indicative voice, present tense, 2nd person plural form aguà.

Let $\mathbf{X} = x_1 \dots x_N$ be a character sequence of the lemma, $\mathbf{T} = t_1 \dots t_M$ a set of morphological tags, and $\mathbf{Y} = y_1 \dots y_K$ be an inflection target character sequence. The goal is to model $P(\mathbf{Y} \mid \mathbf{X}, \mathbf{T})$.

Our approach consists of three major components. First, we propose a novel two-step attention decoder architecture (§2.1). Second, we augment the low-resource datasets with a data hallucination technique (§2.2). Third, we devise a training schedule (§2.3) that substitutes structural biases for attention monotonicity.

### 2.1 Model Architecture

Our models are based on a sequence-to-sequence model with attention (Bahdanau et al., 2015). In broad terms, the model is composed of three parts: an encoder, the attention, and a recurrent decoder. In the setting of the inflection task, there is an additional input provided (the set of morphological tags) which requires an additional encoder.

A visualization of our model is shown in Figure 1. First, an encoder transforms the input character sequence $x_1 \dots x_N$ into a sequence of intermediate representations $\mathbf{h}^x_1 \dots \mathbf{h}^x_N$. A second encoder transforms the set of morphological tags $t_1 \dots t_M$ into another sequence of states $\mathbf{h}^t_1 \dots \mathbf{h}^t_M$:

$$\mathbf{h}^x_n = \text{enc}^{\text{x}}(\mathbf{h}^x_{n-1}, x_n) \quad \text{and} \quad \mathbf{h}^t_m = \text{enc}^{\text{t}}(\mathbf{T}).$$

In our implementation, we use a single layer bidirectional recurrent encoder for the lemma, and a self-attention encoder (Vaswani et al., 2017) for encoding the tags as there is no inherent order (e.g. left-to-right) in their presentation. In preliminary experiments, a self-attention lemma encoder proved hard to train, while a recurrent tag encoder yielded quite competitive results.

Next, we have attention mechanisms that transform the two sequences of input states into two sequences of *context vectors* via two matrices of *attention weights* ($k$ is the current decoder time step):

$$\mathbf{c}^x_k = \left[\textstyle\sum_n \alpha^x_{kn} \mathbf{h}^x_n\right] \qquad \mathbf{c}^t_k = \left[\textstyle\sum_m \alpha^t_{km} \mathbf{h}^t_m\right].$$

Finally, the recurrent decoder computes a sequence of *output states* in a two-step process, from which a probability distribution over output characters can be computed:

$$\begin{aligned} \mathbf{s}_k &= \mathbf{s}'_{k-1} + \mathbf{c}^t_k \\ \mathbf{s}'_k &= \text{dec}(\mathbf{s}'_{k-1}, \mathbf{c}^x_k, y_{k-1}) \\ P(y_k) &= \text{softmax}(\mathbf{s}'_k). \end{aligned}$$

The attention mechanisms produce their weights as in Luong et al. (2015), with $\mathbf{v}^x$, $\mathbf{v}^t$, $\mathbf{W}^{s'}_{\alpha^x}$, $\mathbf{W}^{s}_{\alpha^t}$, $\mathbf{W}^{h}_{\alpha^x}$, and $\mathbf{W}^{h}_{\alpha^t}$ being parameters to be learned:

$$\alpha^t_{km} = \text{softmax}(\mathbf{v}^t \tanh(\left[\mathbf{W}^{s}_{\alpha^t} \mathbf{s}'_{k-1}; \mathbf{W}^{h}_{\alpha^t} \mathbf{h}^t_m\right]))$$

$$\alpha^x_{kn} = \text{softmax}(\mathbf{v}^x \tanh(\left[\mathbf{W}^{s'}_{\alpha^x} \mathbf{s}_k; \mathbf{W}^{h}_{\alpha^x} \mathbf{h}^x_n\right])).$$

The two-step attention process essentially first uses the decoder's previous state $\mathbf{s}'_{k-1}$ as the query for attending over the tags. Then, it creates a tag-

informed state $\mathbf{s}'_k$ by adding the tag context $\mathbf{c}^t_k$ to the previous state. The tag-informed state is then used as the query for attending over the source characters and produce the context $\mathbf{c}^x_k$. The last step is then to update the recurrent state and produce the output character. Ultimately, we desire that the provided tag set guides the generation, which also means influencing the attention over the characters of the lemma.

**Additional Structural Biases for Attention** Incorporating structural biases in the model's architecture or in the training objective can lead to improvements in performance, especially for tasks where the attention mechanism is expected to behave similarly to an alignment model, like MT. This idea has been successfully applied to the inflection task and is at the core of the state-of-the-art model (Wu and Cotterell, 2019).

One bias we deem important is *coverage* of all input characters and tags from the attentions. Intuitively, this entails encouraging the model to "look at" the whole input. We take the approach of Cohn et al. (2016) and add two regularizers over the final attention matrices, encouraging them to also sum to one column-wise:

$$-\lambda \parallel \Sigma_j a^t_{jm} - \mathbb{I} \parallel_2 \qquad -\lambda \parallel \Sigma_j a^x_{jn} - \mathbb{I} \parallel_2$$

Another bias that we incorporate encourages the Markov assumption over attention/alignments. Briefly, this means that if the $i$-th source character/word is aligned to the $j$-th target one ($i \leftarrow j$), then alignments $i{+}1 \leftarrow j{+}1$ or $i \leftarrow j{+}1$ are also quite likely. In a neural architecture this can be approximated by providing the attention weight vector from the previous timestep as input to the function that computes the attention weights. We refer the reader to Cohn et al. (2016) for exact details.

**Adversarial Language Discriminator** When training multilingual systems, encouraging the encoder to learn language-invariant representations can often lead to improvements (Xie et al., 2017; Chen et al., 2018), as it forces the model to truly work in a multilingual setting. We achieve that by introducing a language discriminator (Ganin et al., 2016). This additional component receives the last output of the (bi-directional) intermediate lemma representations $\mathbf{h}^x_N$ and outputs a prediction $y_l$ of the source language such that $y_l$=softmax(MLP($\mathbf{h}^x_N$)).

The discriminator is trained to predict the language by minimizing a standard cross-entropy loss $\mathcal{L}_l$ similar to Lample et al. (2018). However, in order to encourage the encoder to learn language-invariant representations, we reverse the gradients flowing from that component into the encoder during back-propagation.

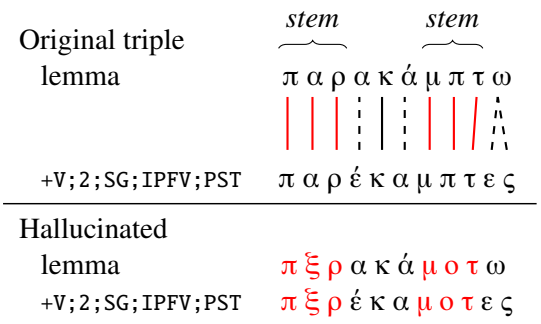


Figure 2: Example of our hallucination process (Greek). The lemma and inflected forms are aligned at the character level. The inside of *stem*-considered parts (highlighted) are substituted with random characters, creating hallucinated triples (bottom).

## 2.2 Data Hallucination

Low-resource language datasets are usually too small to allow for proper learning with neural networks. A major issue in our case is *label bias*. Put simply, the character decoder will overly prefer outputting common character sequences. However, with just 50 examples to learn from, the output character n-gram distribution will hardly match the real one, because the majority of the n-grams will have zero probability. In order to mitigate this issue, we augment our training sets with hallucinated data.

In most languages morphological reinflection is realized by adding, deleting, or modifying prefixes or suffixes over a stem that is mainly unchanged. Though we do not have prior information regarding the stem or affixes of our data we can use character alignment to approximately compute them. We use the alignment method from the SIGMORPHON 2016 baseline (Cotterell et al., 2016) to align the lemmata and the inflected forms. For each example, we consider as part of the stem *any* sequence of three or more[3] consecutive lemma characters that are aligned to the exact same characters in the inflected form.

Now, for each such region considered as a "stem", we randomly substitute its inside (not start or end) characters with other characters from the language's alphabet. Note that we do not change

[3]This heuristic number is largely arbitrary and could potentially be tuned to different values for each language.

the length of the region, though allowing for such variation could possibly lead to further improvement. The substitution characters are sampled uniformly for the alphabet, rather than attempting to sample from a more informed distribution, which has potential for further improvements. Overall, we hallucinated 10,000 examples for each low-resource language, creating an additional hallucinated dataset $\mathcal{H}$.

A visualized example of our hallucination process is outlined in Figure 2. Out of the three regions with matching aligned characters (thick lines), we identify two with length equal to three or more. In the hallucinated example (bottom of the figure), we sample random characters for the inside of such regions.

Silfverberg et al. (2017) have proposed a data hallucination method conceptually quite similar to ours, which treats the *single* longest common continuous substring between lemma and form as the stem. Their approach would be effectively similar to ours for languages with affixal stem-invariant morphology, but it would likely fail in more complicated morphological phenomena like apophony, stem alternation (conceptually similar to infix morphology) or in the root-and-pattern morphology of semitic languages. Consider the apophony example from the past participle form `gschwommen` of the lemma `schwimmen` in Swiss German: our approach would treat both the `schw` and the `mmen` regions as a stem, as opposed to only considering one.[4] Note that neither of the two approaches are suitable for phenomena such as suppletion (e.g. the inflection of the Spanish verb `ir` 'to go' into `fue` 'went') or phenomena like the reduplication pattern of Indonesian noun plurals as in `kuda` 'horse', `kuda-kuda` 'horses' (Sneddon et al., 2012).

### 2.3 Training Schedule

Our training schedule attempts to balance two desires: helping the model to (1) learn to copy, and (2) learn cross-lingually with a particular focus on the low-resource language. To achieve this, we split training into three phases: warm-up, cross-lingual training, and low-resource fine-tuning.

**Phase 1: Warm-Up** As several previous works have noted, learning to copy is crucial. Unlike other proposed models, though, our model does not include any structural biases that encourage copying. Instead, we rely on an additional copying task in a warm up period.

We transform each training triple $[\mathbf{X}, \mathbf{T}, \mathbf{Y}]$ into two additional triples that encourage copying: $[\mathbf{X}, \texttt{COPY}, \mathbf{X}]$ and $[\mathbf{Y}, \mathbf{T}, \mathbf{Y}]$. Using both the input and output sequences for the copying task allows the use of slightly more diverse data. Note that we only use the correct tags when copying the inflected form. For copying the lemma we use a specialized tag (`COPY`). Additional improvements could be achieved if one knew the exact tags that match the lemmata. But this could vary by language: an English verb's lemma is its infinitive and has a `V;NFIN` tag, while a verb lemma in Modern Greek is its `V;1;SG;IPFV;PRS` form.

In this stage we use a relatively large batch size (10) in order to encourage more coarse updates. In most cases, the model achieves extremely high copying accuracy after a couple of warm-up epochs, so we stop the warm up stage when copying accuracy exceeds 75%. At this point the attention mechanism over the source characters has learned to be monotonic. In contrast, the model of Wu and Cotterell (2019) requires a dynamic programming method that forces strict monotonicity, with an additional (non-parallelizable) computational cost $O(|x|^2)$ throughout training.

| Model | Accuracy | Median |
|---|---|---|
| Wu and Cotterell (2019) | 48.5 | 45.5 |
| this work | 48.8 | 67.0 |
| +$\mathcal{H}$ | 60.1 | 66.6 |
| +$\mathcal{H}$ + $\mathcal{L}_l$ | 60.8 | 66.0 |
| +multi-language transfer | **63.8** | 64.0 |
| oracle | 68.2 | 74.0 |

Table 1: Macro-averaged accuracy over 100 language test pairs. Our best model outperforms the baseline by 15 percentage points.

**Phase 2: Cross-lingual training** In the main training phase we use both high- and low-resource language data (including any hallucinated data). If not using hallucinated data, we up-sample the low-resource data in order to match the size of the high-resource ones. Furthermore, with probability 0.30 we also sample copying tasks to intersperse throughout the training epoch. This ensures the source-character attention keeps being monotonic.

**Phase 3: Fine-tuning** The last phase is inspired by fine-tuning, or continued training, as applied to

[4] Most likely, the desired parts are `schw` and `mm`.

| Model | Dev Accuracy (macro-averaged) |
|---|---|
| Wu and Cotterell (2019): | |
| $0^{th}$-order soft attention | 37.1 |
| $0^{st}$-order hard attention | 43.5 |
| $0^{th}$-order monotonic attention | 45.5 |
| $1^{st}$-order monotonic attention | 51.3 |
| two-step attention (this work) | |
| – warm-up, copy task | 31.2 |
| + warm-up, copy task | 48.0 |
| + structural biases | 48.7 |
| + scheduled sampling | 49.1 |
| + minibatch schedule | 49.4 |
| with hallucinated data: | |
| $+\mathcal{H}$ | $82.6^{\dagger}$ |
| $+\mathcal{H} + \mathcal{L}_l$ | $85.4^{\dagger}$ |
| +ensemble (three models) | $85.6^{\dagger}$ |

Table 2: Our proposed two-step attention architecture outperforms almost all baselines on the development set (but with higher gains on the test set). Additional biases and scheduled sampling contribute improvements. †: results with hallucinated data are not directly comparable, as dev data were used for hallucination.

cross-lingual MT e.g. by Neubig and Hu (2018) or to domain adaptation for MT e.g. by Luong and Manning (2015). The setting is nearly identical to the second phase, except we only use the low-resource language data for training and do not use the copying task. Furthermore, we substitute teacher forcing for scheduled sampling (Bengio et al., 2015), where with probability 50% the input to the next step of the decoder is not the gold one, but its previous prediction. This technique allows the model to become more robust to its own mistakes, effectively limiting the effect of exposure bias. It is worth noting that at this point, the learning rate is typically quite small, so we reduce the batch size to a single instance. In most cases, though, the improvements on development set accuracy are marginal (1 to 2%).

### 2.4 Inference and Implementation Details

The performance of inflection systems is typically evaluated with exact-match token-level accuracy, as well as character-level Levenshtein distance.[5] Hence, during training we continuously evaluate the model's performance on the development set with both metrics. Consequently we store three checkpoints: the one that achieved the highest accuracy, the one that reached the lowest Levenshtein distance, and the one that improved on both metrics over previous dev evaluations. In a few cases these three checkpoints coincide, but this is rather rare. We ensemble these three models with equal weights for producing our final predictions.

All our models are implemented in DyNet (Neubig et al., 2017). Each model is trained on a single CPU, as each training run requires less than 1GB of RAM and typically concludes within 3 to 4 hours. We provide additional hyperparameter details in the Appendix.

## 3 Empirical Results

Our experiments are conducted on the SIGMORPHON 2019 challenge datasets (McCarthy et al., 2019a). The dataset consists of 100 pairs between (mostly related) high-resource transfer languages and 43 low-resource test languages with examples taken from the Unimorph database (Kirov et al., 2018). The low-resource languages typically have about 100 training examples, plus 50–100 (or in very few cases, 1000) examples in the development set. In contrast, most high-resource language training sets include 10,000 instances.

**Test Accuracy** Our main results are summarized in Table 1. Our models significantly outperform the baseline, evaluated with macro-averaged accuracy over the 100 language pairs. Our novel architecture performs slightly better than the baseline without any additional data. Importantly, including the hallucinated datasets boosts total accuracy by 10 percentage points, while transfer from multiple languages further improves to a state-of-the-art accuracy of 63.8% – higher than any system submitted to the shared task.

It is worth noting that all of these improvements are not uniform across languages/pairs: without hallucinated training data, our model's median is notably larger than its average, implying that for a few language pairs our model is under-trained and under-performs (in particular, pairs with Yiddish, Votic, and Ingrian as test languages). Nonetheless, if one had access to an oracle that would allow them to select the best performing model out of all the different settings, they would achieve an oracle accuracy of 68.2%.

[5] Due to space constraints, we do not provide Levenshtein distance results. They generally inversely correlate with token-level accuracy.

| L1 | L2 | Genetic dist. | L1+L2 | $+\mathcal{L}_l$ | $+\mathcal{H}$ | $+\mathcal{L}_l + \mathcal{H}$ | $\mathcal{H}$ |
|---|---|---|---|---|---|---|---|
| latin | czech | 0.86 | 15.0 | 26.0 | 71.4 | 68.0 | **77.4** |
| bengali | greek | 0.88 | 22.4 | 16.4 | 70.5 | 70.6 | **71.6** |
| sorani | irish | 1.0 | 20.3 | 18.6 | **66.3** | 64.6 | 65.6 |
| italian | ladin | 0.30 | 48 | 54 | **74** | **74** | **74** |
| latvian | lithuanian | 0.25 | 17.1 | 23.2 | 48.4 | 48.4 | **50.5** |
| english | murrinhpatha | N/A | **36** | 2 | 6 | 20 | 20 |
| italian | neapolitan | 0.10 | 70 | 70 | 83 | 83 | **84** |
| urdu | old english | 0.88 | 23.8 | 18.5 | 43.4 | 40.4 | **44.3** |
| slovene | old saxon | 0.89 | 10.7 | 14.7 | **52.3** | 50.5 | 50.5 |
| russian | portuguese | 0.93 | 34.5 | 22.2 | **88.8** | 88.4 | 87.7 |
| swahili | quechua | 1.0 | 14.2 | 12.8 | **92.1** | 91.6 | 91.6 |
| portuguese | russian | 0.93 | 25.6 | 17.5 | **76.3** | 74.6 | 74.3 |
| kurmanji | sorani | N/A | 16.2 | 13.6 | **69.0** | 64.3 | 66.7 |
| zulu | swahili | 1.0 | 46 | 52 | **81** | 80 | 76 |
| kannada | telugu | 0.75 | 76 | 72 | **94** | 90 | **94** |
| Average | | 0.75 | 31.72 | 28.90 | 67.77 | 67.23 | 68.55 |

Table 3: Results with a single transfer language. Monolingual data hallucination is crucial due to the distance of the languages. In some cases cross lingual transfer should be avoided in favor of a purely monolingual setting ($\mathcal{H}$).

**Architecture Ablations** We focus on the architecture of the model and the structural biases we introduced, using the development set for evaluation. Table 2 presents the macro-averaged accuracy for the baseline (Wu and Cotterell, 2019) and the different versions of our model.[6]

The first thing to note is the importance of the warm-up period in our training schedule and the use of the copying task. Our model neither handles copying in any explicit way nor encourages the attention to be monotonic. Without the warm-up period and the additional copying tasks, our model's dev accuracy is worse than any of the baselines.

On the other hand, our two-step attention trained with the additional copying task already improves over most baselines without any of the additional biases, with a dev accuracy of 48. The best baseline model, with an average dev accuracy of 51.3, is better than our models, but this difference is not reflected on the test set (see Table 1) where our model performs slightly better. This discrepancy could be due to optimizer or early stopping decisions, but we leave a more thorough investigation for future work.

We attribute the success of our model to two factors, the first being our novel architecture. Compared to a single attention over the concatenated tag and lemma sequences, our two-step attention has the advantage of two distinct attention mechanisms, which capture the inherently different properties of the tag and the lemma character sequences. Another advantage is the two-step process that guides the lemma attention with the tags. Disentangling the two attentions *and* ordering them in an intuitive way makes it easier for them to learn their respective tasks.

The second factor is, we suspect, a slightly better choice of hyperparameters: our models are smaller than the baseline ones. Although we did not tune our hyperparameters extensively, a few experiments on a couple of language pairs showed that increasing the model size hurt performance under these extremely data-scarce conditions.

Each of the additional techniques we tested further contributed a few accuracy points. The attention biases add 0.7% points and scheduled sampling in the third training phase further adds 0.4% points. The different batching schedule with large batch sizes in the beginning but smaller towards the end of training helps a little more in terms of accuracy, but most importantly it speeds up training. Table 2 also reports the development set accuracy when using hallucinated data. Although the large improvements are also proportionally reflected in the test set, these numbers are not directly comparable to the rest, as the development set data were used in the hallucination process.

## 4 Analysis

We analyze the results over various groupings of languages to elucidate the properties of our models over the 100 quite diverse language pairs. We use typological information from the URIEL database (Littell et al., 2017) in this analysis.

**Single Language Transfer** We first focus on the test languages for which a *single* transfer language

[6]Table 2 includes unpublished results kindly shared by Wu and Cotterell, the authors of the baseline.

| L1 | L2 | L1+L2 | $+\mathcal{L}_l$ | $+\mathcal{H}$ | $+\mathcal{L}_l+\mathcal{H}$ | $\mathcal{H}$ |
|---|---|---|---|---|---|---|
| turkish | azeri | 81 | 77 | 80 | 81 | 66.7±0.9 |
| persian | | 55 | 63 | 74 | 69 | |
| bashkir | | 57 | 59 | 66 | 67 | |
| uzbek | | 47 | 55 | 74 | 70 | |
| all | | 84 | 71 | 83 | **87** | |
| urdu | bengali | 42 | 32 | 66 | **67** | 63.7±4.0 |
| sanskrit | | 44 | 38 | 66 | 65 | |
| hindi | | 49 | 52 | 67 | 65 | |
| greek | | 42 | 46 | 65 | **67** | |
| all | | 49 | 50 | 64 | 62 | |
| turkish | crimean tatar | 87 | 80 | 85 | **89** | 71.3±1.1 |
| bashkir | | 59 | 60 | 70 | 69 | |
| uzbek | | 60 | 60 | 72 | 67 | |
| all | | 82 | 81 | 88 | 80 | |
| finnish | ingrian | 38 | 36 | 34 | 40 | 34.6±2.3 |
| hungarian | | 28 | 32 | 32 | 38 | |
| estonian | | 32 | 32 | 32 | 38 | |
| all | | 38 | **44** | 36 | 36 | |
| finnish | karelian | 54 | 50 | 58 | 56 | 52.6±1.1 |
| hungarian | | 42 | 46 | 58 | 52 | |
| estonian | | 42 | 42 | 54 | 58 | |
| all | | 50 | 46 | 54 | **64** | |
| basque | kashubian | 46 | 40 | 70 | 76 | 74.0±2.3 |
| slovak | | 58 | 62 | 74 | 76 | |
| czech | | 54 | 64 | 78 | 66 | |
| polish | | 66 | 78 | 78 | **80** | |
| all | | 70 | 72 | 78 | **80** | |
| estonian | livonian | 27 | 27 | 35 | 34 | 33±1 |
| hungarian | | 28 | 30 | 35 | 33 | |
| finnish | | 30 | 26 | 34 | 35 | |
| all | | 26 | 25 | **36** | **36** | |
| italian | maltese | 34 | 25 | **48** | 42 | 45.6±5.6 |
| arabic | | 18 | 26 | 41 | 38 | |
| hebrew | | 20 | 21 | 47 | 45 | |
| all | | 29 | 28 | 40 | 46 | |
| danish | middle high german | 68 | 58 | 78 | 80 | 76.7±6.0 |
| dutch | | 70 | 62 | 74 | 82 | |
| german | | 72 | 68 | **86** | 82 | |
| all | | 70 | 70 | 84 | 78 | |
| danish | north frisian | 18 | 25 | 44 | 46 | 43±1 |
| dutch | | 28 | 22 | **47** | 46 | |
| english | | 22 | 26 | **47** | 42 | |
| all | | 23 | 25 | 43 | 46 | |
| asturian | occitan | 58 | 49 | 77 | 74 | 78.3±3.5 |
| spanish | | 47 | 55 | 78 | 76 | |
| french | | 41 | 52 | **80** | **80** | |
| all | | 52 | 61 | 78 | **80** | |
| russian | old church slavonic | 40 | 39 | 39 | **64** | 57.3±1.5 |
| polish | | 41 | 41 | 59 | 58 | |
| bulgarian | | 38 | 42 | 44 | 56 | |
| all | | 41 | 42 | **64** | 56 | |
| sanskrit | pashto | 20 | 16 | 46 | 44 | 46.5±3.5 |
| persian | | 36 | 28 | 39 | 48 | |
| all | | 29 | 30 | 46 | **49** | |
| bashkir | tatar | 67 | 64 | 73 | 69 | 73.0±4.6 |
| uzbek | | 55 | 45 | 67 | 72 | |
| turkish | | 81 | 79 | 82 | 75 | |
| all | | 81 | 79 | **84** | 83 | |
| Average (over all pairs) | | 49.00 | 48.48 | 59.36 | 60.18 | 57.9 |

Table 4: Results with multiple transfer languages (sample). The best performing system per target (L2) language is **highlighted**. We repeated the hallucinated-data-only experiments as many times as potential transfer (L1) languages, hence the $\mathcal{H}$ column reports average accuracy ± standard deviation.

was provided, with detailed results presented in Table 3. Generally, the average genetic distance between the transfer and the task language is quite high (0.75) for these pairs. It is easy to observe that the larger the typological distance, the larger the improvement from adding hallucinated data. In fact, excluding the languages with no typological information available, there is strong correlation ($\rho$=0.6) between genetic distance and improvement from hallucinated data.

For cross-lingual transfer to be useful, the languages need to be at least somewhat related and share similar characteristics. A prime example is transfer from Italian for Neapolitan, which achieves a 70% accuracy without any additional synthesized data. In the same vein, the same condition is necessary for the adversarial language discriminator to have impact, as using it on extremely distant language pairs leads to worse performance (e.g. Russian-Portuguese, Bengali-Greek, or Urdu-Old English). This is expected, as forcing language invariant representations across vastly different languages is analogous to representing a bimodal distribution with its mean.

The results on Kurmanji-Sorani (Northern Kurdish-Central Kurdish) seem to be a valid counter-example to the above statement, i.e. the two languages are related,[7] but cross-lingual transfer without hallucinated data performs poorly, achieving a mere 16.2 accuracy. The reason for this discrepancy lies in the characters: Kurmanji is written in the Latin alphabet, while Sorani uses the Arabic one.[8] The lack of any similar representations across the languages is too hard to overcome even with the adversarial language discriminator.[9]

**Multiple Transfer Languages** For most low-resource languages and especially dialects, there exist several possible candidate transfer languages that can be related enough to satisfy the similarity constraint. We present extensive ablations on such cases in Table 4 with results on the rest of the SIGMORPHON language pairs.[10]

We again observe positive correlations between the language genetic proximity and the performance of cross-lingual transfer, even with all the transfer and test languages being related. For example, transfer from (distant) Basque to Kashubian performs about 10 percentage points worse than transfer from (related) Slovak or Czech, which in turn perform worse than transfer from Polish (more closely related to Kashubian than the

[7] A reviewer pointed out that Kurmanji and Sorani might actually be more distant than the politically-motivated term "Kurdish dialects" might imply.

[8] Kurmanji is also sometimes written with the Cyrillic or the Nashk Arabic scripts, but our data are Latin-only.

[9] We did not attempt to address this issue, by e.g. increasing the loss weight, but we leave this for future work.

[10] We present a sample due to space constraints. The discussion pertains to all language pairs. The full table is available in the Appendix.

others). Transfer for North Frisian using Danish is also 10 percentage points worse than transfer from the more closely related Dutch. It is worth noting that using the hallucinated data reduces the effect of the genetic distance between languages, with the standard deviation across the results within the same test language becoming much smaller.

A very interesting case of cross-lingual transfer is Maltese, which is a semitic language and hence genetically close to Hebrew and Arabic. Surprisingly, we obtain better results when transferring from Italian. Again, a script discrepancy could be the main reason, also considering that the root and pattern morphology is only partially expressed in the scripts of Hebrew and Arabic, whereas it is fully expressed (by writing all the vowels) in Maltese. We should also point out that genetic similarity might not be enlightening enough. As Hoberman and Aronoff (2003) point out, the productive verbal morphology in Maltese has become affixal due to borrowings from Romance languages. To further exacerbate the situation, all provided train, dev, and test examples are verbs (no nouns or adjectives), providing an explanation to this seemingly counter-intuitive result.

Another interesting case is that of cross-lingual transfer for Bengali, with the potential languages varying from very related (Sanskrit, Hindi) to even only distantly related (Greek). Nevertheless, there is notably little variance in the performance of the systems. We believe that again the culprit is the difference in writing systems between all selected transfer and test language, which does not allow our system to leverage cross-lingual information: our Bengali data use the Bengali script, the Urdu dataset is in the Arabic one, Hindi and Sanskrit use Devanagari, and Greek uses the Greek alphabet.

We also present analytic results with cross-lingual transfer from all transfer languages from the suggested SIGMORPHON pairs.[11] In 14 out of the 29 test languages, our best-performing model is trained on multiple transfer languages. For instance, using Turkish, Persian, Bashkir, and Uzbek data for transfer to Azeri leads to a 6 point improvement over any single-language-transfer result. A potential explanation is that a dialect/language has indeed been influenced by multiple languages. Another reason could lie in the increased amount of data and potential regularization effects. We suspect the truth lies in the union of those factors, but nonetheless we conclude that **whenever available, transfer from multiple related languages can further improve accuracy**.

In our experiments we used all transfer languages that the SIGMORPHON organizers proposed in the 2019 challenge. On the one hand, a more sophisticated data selection process could likely yield improvements. For example, Yiddish is primarily based in High German and it has elements from Hebrew, Aramaic, as well as Slavic languages, but we did not test how transfer from these languages might perform. Moreover, in order to remain faithful to the SIGMORPHON challenge, we used some distant languages that probably worsen the results (e.g. Greek for Bengali).

Also, alphabet divergence issues still need to be addressed. For instance, we suspect that our accuracy in Yiddish (which uses the Hebrew alphabet with Yiddish orthography) could be greatly improved by finding some type of mapping between its orthography and the Latin script that most of its related languages use. The same could hold for transfer for the central Asian languages (Tatar, Turkmen, Azeri among others) which use a variety of the Latin, Cyrillic, or Arabic scripts.

Lastly, we experimented with a completely monolingual setting, using just the low-resource and hallucinated language data (columns $\mathcal{H}$ in Tables 3 and 4). For fairer comparison to cross-lingual transfer, we repeated the hallucination process as many times as candidate transfer languages and we report the mean and standard deviation of the test set accuracy. This baseline is extremely competitive, lagging only a few points behind the $\mathcal{L} + \mathcal{H}$ combination. Encouragingly, this entails that hallucination is a viable option for entire language families without a single high-resource representative or low-resource isolates.

**Interpretability** We find that the attention matrices can help understand our model's predictions. A visualization of two examples is shown in Figure 3, showcasing the interpretability advantage of the disentangled two-step attentions. In the Kazakh example the tag attention clearly identifies the suffixes да (that marks plural) and рды (that marks the accusative case). The Greek example is a great one of how the two-step process allows the tags to guide the lemma attention. Due to the SBJV tag, the model does not use the lemma until the necessary particle να has been generated. Conse-

[11] We do not use *all* languages for transfer in a test language. For instance, when testing on Occitan 'all' stands for training on Asturian, Spanish, and French.

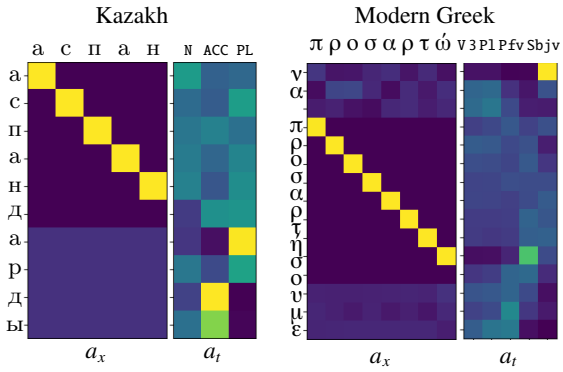

Figure 3: Attention visualization examples. The inflected form is generated from top to bottom.

quently, the lemma attention properly copies the stem, and then the tag attention attends first over PFV and then over PL and 3 in order to construct the correct suffix for perfective and 3rd person plural.

## 5 Related Work

The inflection task in high-resource settings has been extensively studied through the SIGMORPHON shared tasks. Notably, the best models explicitly model copying and hard monotonic attention (Aharoni et al., 2016; Aharoni and Goldberg, 2017) with the previous state-of-the-art forcing strict monotonicity (Wu and Cotterell, 2019). We instead achieve state-of-the-art with a cheaper approach that simply intermixes a copying task which also encourages monotonicity.

Data augmentation for inflection has been explored by Bergmanis et al. (2017) and Zhou and Neubig (2017) among others. The work of Silfverberg et al. (2017) is the most similar to ours, but as we already discussed, it has a few shortcomings that our approach addresses.

Kann et al. (2017) have identified typology as playing a role for cross-lingual transfer, but they measure language similarity using lexical overlap. We attest that this data-based measure is less informative and more suspect to variation, so we instead use the genetic typological information to quantify correlations between performance improvements and language distance.

Our novel two-step process decoder architecture bares similarities with multi-source models (Anastasopoulos and Chiang, 2018; Zoph and Knight, 2016) which provide two contexts from two encoded sources to the decoder. A similar disentangled encoding was also used by Ács (2018) for their SIGMORPHON 2018 submission. We in fact experimented with this architecture but preliminary results on the development sets showed that our two-step architecture achieved better performance. Interestingly, the second-best performing system (Peters and Martins, 2019) at SIGMORPHON 2019, which also ranked first in terms of Levenshtein distance, also uses decoupled encoders to separately encode the lemma and the tags; this further cosolidates our belief that such an approach is superior to using a single encoder for the concatentated sequence of the tags and lemma. The main difference to our model is that they do not use our two-step decoder process, while they substitute all softmax operations with sparsemax (Martins and Astudillo, 2016), yielding interpretable attention matrices very similar to ours. The use of sparsemax in conjunction with our two-step decoder process, as well as along our data hallucination technique, presents a promising direction towards even better results in the future.

## 6 Conclusion

With this work we advance the state-of-the-art for morphological inflection on low-resource languages by 15 points, through a novel architecture, data hallucination, and a variety of training techniques. Our two-step attention decoder follows an intuitive order, also enhancing interpretability. We also suggest that complicated methods for copying and forcing monotonicity are unnecessary. We identify language genetic similarity as a major success factor for cross-lingual training, and show that using many related languages leads to even better performance. Despite this significant stride, the problem is far from solved. Language-specific or language-family-specific improvements (i.e. proper dealing with different alphabets, or using an adversarial language discriminator) could potentially further boost performance.

## Acknowledgements

The authors are grateful to the anonymous reviewers for their exceptionally constructive and insightful comments, to Arya McCarthy for discussions on morphological inflection, to Shruti Rijhwani for her comments on Bengali, to Shijie Wu and Ryan Cotterell for sharing the dev performance of their model, as well as to Gabriela Weigel for her invaluable help with editing and proofreading the paper. This material is based upon work generously supported by the National Science Foundation under grant 1761548.

## References


Judit Ács. 2018. BME-HAS system for CoNLL–SIGMORPHON 2018 shared task: Universal morphological reinflection. In *Proc. CoNLL–SIGMORPHON*, pages 121–126, Brussels. Association for Computational Linguistics.

Roee Aharoni and Yoav Goldberg. 2017. Morphological inflection generation with hard monotonic attention. In *Proc. ACL*.

Roee Aharoni, Yoav Goldberg, and Yonatan Belinkov. 2016. Improving sequence to sequence learning for morphological inflection generation: The BIU-MIT systems for the SIGMORPHON 2016 shared task for morphological reinflection. In *Proc. SIGMORPHON*.

Antonios Anastasopoulos and David Chiang. 2018. Leveraging translations for speech transcription in low-resource settings. In *Proc. INTERSPEECH*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. ICLR*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Proc. NIPS*.

Toms Bergmanis, Katharina Kann, Hinrich Schütze, and Sharon Goldwater. 2017. Training data augmentation for low-resource morphological inflection. *Proc. SIGMORPHON*.

Theresa Breiner, Chieu Nguyen, Daan van Esch, and Jeremy O'Brien. 2019. Automatic keyboard layout design for low-resource Latin-script languages. arXiv:1901.06039.

Xilun Chen, Yu Sun, Ben Athiwaratkun, Claire Cardie, and Kilian Weinberger. 2018. Adversarial deep averaging networks for cross-lingual sentiment classification. *Transactions of the Association for Computational Linguistics*, 6:557–570.

Trevor Cohn, Cong Duy Vu Hoang, Ekaterina Vymolova, Kaisheng Yao, Chris Dyer, and Gholamreza Haffari. 2016. Incorporating structural alignment biases into an attentional neural translation model. In *Proc. NAACL-HLT*.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Arya D. McCarthy, Katharina Kann, Sebastian Mielke, Garrett Nicolai, Miikka Silfverberg, David Yarowsky, Jason Eisner, and Mans Hulden. 2018. The CoNLL–SIGMORPHON 2018 shared task: Universal morphological reinflection. In *Proc. CoNLL–SIGMORPHON*.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sandra Kübler, David Yarowsky, Jason Eisner, and Mans Hulden. 2017. CoNLL-SIGMORPHON 2017 shared task: Universal morphological reinflection in 52 languages. In *Proc. CoNLL SIGMORPHON*.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016. The SIGMORPHON 2016 shared task—morphological reinflection. In *Proc. SIGMORPHON*.

Ben Foley, Josh Arnold, Rolando Coto-Solano, Gautier Durantin, T Mark Ellison, Daan van Esch, Scott Heath, František Kratochvíl, Zara Maxwell-Smith, David Nash, et al. 2018. Building speech recognition systems for language documentation: The CoEDL endangered language pipeline and inference system (Elpis). In *Proc. SLTU*.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *JMLR*.

Robert D Hoberman and Mark Aronoff. 2003. The verbal morphology of Maltese. *Language Acquisition and Language Disorders*, 28:61–78.

Katharina Kann, Ryan Cotterell, and Hinrich Schütze. 2017. One-shot neural cross-lingual transfer for paradigm completion. In *Proc. ACL*.

Christo Kirov, Ryan Cotterell, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sebastian J. Mielke, Arya McCarthy, Sandra Kübler, David Yarowsky, Jason Eisner, and Mans Hulden. 2018. UniMorph 2.0: Universal Morphology. In *Proc. LREC*.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Unsupervised machine translation using monolingual corpora only. In *Proc. ICLR*.

Patrick Littell, David R. Mortensen, Ke Lin, Katherine Kairis, Carlisle Turner, and Lori Levin. 2017. Uriel and lang2vec: Representing languages as typological, geographical, and phylogenetic vectors. In *Proc. EACL*.

Minh-Thang Luong and Christopher D Manning. 2015. Stanford neural machine translation systems for spoken language domains. In *Proc. IWSLT*.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proc. EMNLP*.

Andre Martins and Ramon Astudillo. 2016. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *Proc. ICML*, pages 1614–1623.

Arya D. McCarthy, Ekaterina Vylomova, Shijie Wu, Chaitanya Malaviya, Lawrence Wolf-Sonkin, Garrett Nicolai, Christo Kirov, Miikka Silfverberg, Sebastian Mielke, Jeffrey Heinz, Ryan Cotterell, and Mans Hulden. 2019a. The SIGMORPHON 2019 shared task: Crosslinguality and context in morphology. In *Proc. SIGMORPHON*.

Arya D. McCarthy, Ekaterina Vylomova, Shijie Wu, Chaitanya Malaviya, Lawrence Wolf-Sonkin, Garrett Nicolai, Miikka Silfverberg, Sebastian J. Mielke, Jeffrey Heinz, Ryan Cotterell, and Mans Hulden. 2019b. The SIGMORPHON 2019 shared task: Morphological analysis in context and cross-lingual transfer for inflection. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 229–244, Florence, Italy.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, et al. 2017. DyNet: The dynamic neural network toolkit. arXiv:1701.03980.

Graham Neubig and Junjie Hu. 2018. Rapid adaptation of neural machine translation to new languages. In *Proc. EMNLP*.

Ben Peters and André F. T. Martins. 2019. IT–IST at the SIGMORPHON 2019 shared task: Sparse two-headed models for inflection. In *Proc. SIGMORPHON*, Florence, Italy.

Lane Schwartz, Emily Chen, Benjamin Hunt, and Sylvia LR Schreiner. 2019. Bootstrapping a neural morphological analyzer for St. Lawrence Island Yupik from a finite-state transducer. In *Proc. Comput-EL3*.

Miikka Silfverberg, Adam Wiemerslage, Ling Liu, and Lingshuang Jack Mao. 2017. Data augmentation for morphological reinflection. *Proc. SIGMORPHON*.

James Neil Sneddon, K Alexander Adelaar, Dwi N Djenar, and Michael Ewing. 2012. *Indonesian: A comprehensive grammar*. Routledge.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proc. NeurIPS*.

Shijie Wu and Ryan Cotterell. 2019. Exact hard monotonic attention for character-level transduction. arXiv:1905.06319.

Qizhe Xie, Zihang Dai, Yulun Du, Eduard Hovy, and Graham Neubig. 2017. Controllable invariance through adversarial feature learning. In *Proc. NeurIPS*, pages 585–596.

Chunting Zhou and Graham Neubig. 2017. Morphological inflection generation with multi-space variational encoder-decoders. In *Proc. SIGMORPHON*.

Barret Zoph and Kevin Knight. 2016. Multi-source neural translation. In *Proc. NAACL-HLT*.

## A Hyperparameters

Here we list all the hyperparameters of our models:

- Encoder/Decoder number of layers: 1
- Character/Tag Embedding size: 32
- Recurrent State Size: 100
- LSTM Type: CoupledLSTM
- Attention Size: 100
- Target and Source Character Embeddings are Tied
- Tag Self-Attention size: 100
- Tag Self-Attention heads: 1

Here we list the optimizer settings:

- Optimizer: SimpleSGD
- Starting learning rate: 0.1
- Learning Rate Decay: 0.5
- Learning Rate Decay Patience: 6 epochs
- Maximum number of epochs: 20, 40, 40 (for each training phase)
- Minibatch Size: 10, 10, 2 (for each training phase).

| L1 | L2 | Genetic dist. | L1+L2 | $+\mathcal{L}_l$ | $+\mathcal{H}$ | $+\mathcal{L}_l + \mathcal{H}$ | $\mathcal{H}$ |
|---|---|---|---|---|---|---|---|
| latin | czech | 0.86 | 15.0 | 26.0 | 71.4 | 68.0 | **77.4** |
| bengali | greek | 0.88 | 22.4 | 16.4 | 70.5 | 70.6 | **71.6** |
| sorani | irish | 1.0 | 20.3 | 18.6 | **66.3** | 64.6 | 65.6 |
| italian | ladin | 0.30 | 48 | 54 | **74** | **74** | **74** |
| latvian | lithuanian | 0.25 | 17.1 | 23.2 | 48.4 | 48.4 | **50.5** |
| english | murrinhpatha | N/A | **36** | 2 | 6 | 20 | 20 |
| italian | neapolitan | 0.10 | 70 | 70 | 83 | 83 | **84** |
| urdu | old english | 0.88 | 23.8 | 18.5 | 43.4 | 40.4 | **44.3** |
| slovene | old saxon | 0.89 | 10.7 | 14.7 | **52.3** | 50.5 | 50.5 |
| russian | portuguese | 0.93 | 34.5 | 22.2 | **88.8** | 88.4 | 87.7 |
| swahili | quechua | 1.0 | 14.2 | 12.8 | **92.1** | 91.6 | 91.6 |
| portuguese | russian | 0.93 | 25.6 | 17.5 | **76.3** | 74.6 | 74.3 |
| kurmanji | sorani | N/A | 16.2 | 13.6 | **69.0** | 64.3 | 66.7 |
| zulu | swahili | 1.0 | 46 | 52 | **81** | 80 | 76 |
| kannada | telugu | 0.75 | 76 | 72 | **94** | 90 | **94** |
| Average | | 0.75 | 31.72 | 28.90 | 67.77 | 67.23 | 68.55 |

Table 5: Results with a single transfer language. Monolingual data hallucination is crucial due to the distance of the languages. In some cases cross lingual transfer should be avoided in favor of a purely monolingual setting ($\mathcal{H}$).

## B Complete Result Tables

Table 5 lists all results for test languages with a single candidate transfer language. Results on test languages with multiple candidate languages (both single-language-transfer, and transfer with all candidate suggested languages) are listed in Tables 6 and 7.

| L1 | L2 | L1+L2 | $+\mathcal{L}_l$ | $+\mathcal{H}$ | $+\mathcal{L}_l + \mathcal{H}$ | $\mathcal{H}$ |
|---|---|---|---|---|---|---|
| turkish | azeri | 81 | 77 | 80 | 81 | 66.7±0.9 |
| persian | | 55 | 63 | 74 | 69 | |
| bashkir | | 57 | 59 | 66 | 67 | |
| uzbek | | 47 | 55 | 74 | 70 | |
| all | | 84 | 71 | 83 | **87** | |
| urdu | bengali | 42 | 32 | 66 | **67** | 63.7±4.0 |
| sanskrit | | 44 | 38 | 66 | 65 | |
| hindi | | 49 | 52 | 67 | 65 | |
| greek | | 42 | 46 | 65 | **67** | |
| all | | 49 | 50 | 64 | 62 | |
| welsh | breton | 54 | 55 | 83 | 86 | 78.2±2.5 |
| irish | | 48 | 52 | 80 | **88** | |
| albanian | | 53 | 59 | 78 | 81 | |
| all | | 62 | 55 | 81 | 82 | |
| hebrew | classical syriac | 86 | 85 | 95 | 95 | 94±0 |
| arabic | | 83 | 82 | 92 | 95 | |
| all | | 87 | 85 | 95 | **96** | |
| irish | cornish | 18 | 24 | 24 | 20 | 24±0 |
| welsh | | 26 | **28** | **28** | 24 | |
| all | | 22 | 16 | 22 | 24 | |
| turkish | crimean tatar | 87 | 80 | 85 | **89** | 71.3±1.1 |
| bashkir | | 59 | 60 | 70 | 69 | |
| uzbek | | 60 | 60 | 72 | 67 | |
| all | | 82 | 81 | 88 | 80 | |
| spanish | friulian | 55 | 55 | 81 | **83** | 79.0±2.6 |
| italian | | 56 | 55 | 78 | 77 | |
| all | | 64 | 58 | 78 | **83** | |
| finnish | ingrian | 38 | 36 | 34 | 40 | 34.6±2.3 |
| hungarian | | 28 | 32 | 32 | 38 | |
| estonian | | 32 | 32 | 32 | 38 | |
| all | | 38 | **44** | 36 | 36 | |
| adyghe | kabardian | 92 | 92 | 93 | **96** | 90.6±4.0 |
| armenian | | 78 | 77 | 86 | 80 | |
| all | | 95 | 91 | 93 | 90 | |
| finnish | karelian | 54 | 50 | 58 | 56 | 52.6±1.1 |
| hungarian | | 42 | 46 | 58 | 52 | |
| estonian | | 42 | 42 | 54 | 58 | |
| all | | 50 | 46 | 54 | **64** | |
| basque | kashubian | 46 | 40 | 70 | 76 | 74.0±2.3 |
| slovak | | 58 | 62 | 74 | 76 | |
| czech | | 54 | 64 | 78 | 66 | |
| polish | | 66 | 78 | 78 | **80** | |
| all | | 70 | 72 | 78 | **80** | |
| turkish | kazakh | **86** | 80 | 74 | 66 | 73.3±3.0 |
| bashkir | | 68 | 84 | 74 | 70 | |
| uzbek | | 66 | 60 | 78 | 70 | |
| all | | 76 | 78 | 80 | 78 | |
| bashkir | khakas | 82 | 78 | 80 | 84 | 72.6±6.4 |
| uzbek | | 84 | 80 | 72 | 76 | |
| turkish | | 90 | 90 | 80 | 82 | |
| all | | 84 | **92** | 86 | 84 | |
| czech | latin | 5.4 | 5.0 | 20.6 | 42.0 | 41.9±1.4 |
| romanian | | 7.9 | 9.0 | 18.8 | 41.3 | |
| all | | 3.6 | 7.7 | 40.1 | **44.1** | |
| estonian | livonian | 27 | 27 | 35 | 34 | 33±1 |
| hungarian | | 28 | 30 | 35 | 33 | |
| finnish | | 30 | 26 | 34 | 35 | |
| all | | 26 | 25 | **36** | **36** | |

Table 6: Multiple transfer language results (part 1).

| L1 | L2 | L1+L2 | $+\mathcal{L}_l$ | $+\mathcal{H}$ | $+\mathcal{L}_l + \mathcal{H}$ | $\mathcal{H}$ |
|---|---|---|---|---|---|---|
| italian | maltese | 34 | 25 | **48** | 42 | 45.6±5.6 |
| arabic | | 18 | 26 | 41 | 38 | |
| hebrew | | 20 | 21 | 47 | 45 | |
| all | | 29 | 28 | 40 | 46 | |
| danish | middle high german | 68 | 58 | 78 | 80 | 76.7±6.0 |
| dutch | | 70 | 62 | 74 | 82 | |
| german | | 72 | 68 | **86** | 82 | |
| all | | 70 | 70 | 84 | 78 | |
| dutch | middle low german | 36 | 42 | 32 | 36 | 36.6±3.0 |
| german | | **46** | 26 | 32 | 38 | |
| danish | | 30 | 24 | 30 | 30 | |
| all | | 32 | 42 | 36 | 38 | |
| danish | north frisian | 18 | 25 | 44 | 46 | 43±1 |
| dutch | | 28 | 22 | **47** | 46 | |
| english | | 22 | 26 | **47** | 42 | |
| all | | 23 | 25 | 43 | 46 | |
| asturian | occitan | 58 | 49 | 77 | 74 | 78.3±3.5 |
| spanish | | 47 | 55 | 78 | 76 | |
| french | | 41 | 52 | **80** | **80** | |
| all | | 52 | 61 | 78 | **80** | |
| russian | old church slavonic | 40 | 39 | 39 | **64** | 57.3±1.5 |
| polish | | 41 | 41 | 59 | 58 | |
| bulgarian | | 38 | 42 | 44 | 56 | |
| all | | 41 | 42 | **64** | 56 | |
| irish | old irish | 2 | 4 | **12** | 2 | 6±2 |
| belarusian | | 10 | 10 | 10 | 6 | |
| welsh | | 4 | 6 | 10 | 6 | |
| all | | 4 | 4 | 8 | 6 | |
| sanskrit | pashto | 20 | 16 | 46 | 44 | 46.5±3.5 |
| persian | | 36 | 28 | 39 | 48 | |
| all | | 29 | 30 | 46 | **49** | |
| welsh | scottish gaelic | 46 | 34 | 64 | 64 | 58±2 |
| irish | | 58 | 62 | **68** | 66 | |
| latvian | | 58 | 36 | 58 | 66 | |
| all | | 60 | 64 | 62 | 66 | |
| bashkir | tatar | 67 | 64 | 73 | 69 | 73.0±4.6 |
| uzbek | | 55 | 45 | 67 | 72 | |
| turkish | | 81 | 79 | 82 | 75 | |
| all | | 81 | 79 | **84** | 83 | |
| bashkir | turkmen | 82 | 76 | 82 | 88 | 80±4 |
| arabic | | 64 | 66 | 84 | 80 | |
| turkish | | 82 | 88 | 90 | **92** | |
| uzbek | | 66 | 74 | 86 | 78 | |
| all | | **92** | 90 | 84 | 88 | |
| estonian | votic | 12 | 17 | 23 | 27 | 26.3±2.1 |
| finnish | | 24 | 20 | 26 | 28 | |
| hungarian | | 10 | 10 | 24 | **30** | |
| all | | 17 | 20 | 28 | 29 | |
| dutch | west frisian | 36 | 39 | 44 | **49** | 47±1 |
| english | | 36 | 25 | 45 | 43 | |
| danish | | 33 | 33 | 42 | 43 | |
| all | | 40 | 38 | 43 | 45 | |
| danish | yiddish | 50 | 54 | 55 | 56 | 55.3±0.6 |
| dutch | | 49 | 50 | 56 | 55 | |
| german | | 53 | 54 | **57** | 54 | |
| all | | 55 | 55 | 55 | 54 | |
| Average (over all pairs) | | 49.00 | 48.48 | 59.36 | 60.18 | 57.9 |

Table 7: Multiple transfer language results (part 2).

# Cross-Lingual Dependency Parsing Using Code-Mixed TreeBank

**Meishan Zhang**[1] and **Yue Zhang**[2*] and **Guohong Fu**[3]
1. School of New Media and Communication, Tianjin University, China
2. Institute of Advanced Technology, Westlake Institute for Advanced Study
3. Institute of Artificial Intelligence, Soochow University, China
mason.zms@gmail.com,
zhangyue@westlake.edu.cn,
ghfu@hotmail.com

## Abstract

Treebank translation is a promising method for cross-lingual transfer of syntactic dependency knowledge. The basic idea is to map dependency arcs from a source treebank to its target translation according to word alignments. This method, however, can suffer from imperfect alignment between source and target words. To address this problem, we investigate syntactic transfer by code mixing, translating only confident words in a source treebank. Cross-lingual word embeddings are leveraged for transferring syntactic knowledge to the target from the resulting code-mixed treebank. Experiments on University Dependency Treebanks show that code-mixed treebanks are more effective than translated treebanks, giving highly competitive performances among cross-lingual parsing methods.

## 1 Introduction

Treebank translation (Tiedemann et al., 2014; Tiedemann, 2015; Tiedemann and Agić, 2016) has been considered as a method for cross-lingual syntactic transfer. Take dependency grammar for instance. Given a source treebank, machine translation is used to find target translations of its sentences. Then word alignment is used to find mappings between source and target words, so that source syntactic dependencies can be projected to the target translations. Following, a post-processing step is applied by removing unaligned target words, in order to ensure that the resulting target syntax forms a valid dependency tree, The method has shown promising performance for unsupervised cross-lingual dependency parsing among transfer methods (McDonald et al., 2011; Täckström et al., 2012; Rasooli and Collins, 2015; Guo et al., 2016b).

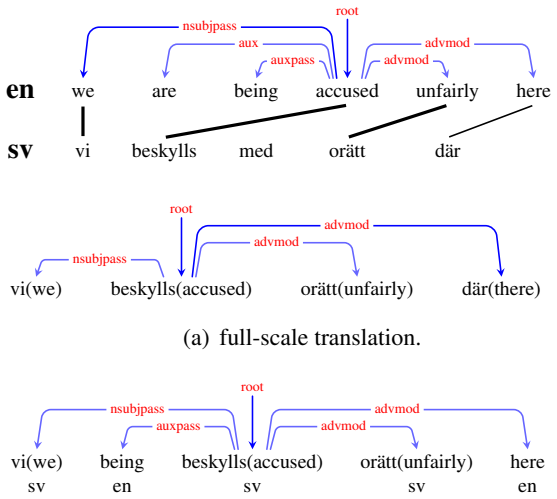


(a) full-scale translation.




(b) this method, partial translation.

Figure 1: An example to illustrate our method, where the source and target languages are English (en) and Swedish (sv), respectively.

The treebank translation method, however, suffers from various sources of noise. For example, machine translation errors directly affect the resulting treebank, by introducing ungrammatical word sequences. In addition, the alignments between source and target words may not be isomorphic due to inherent differences between languages or paraphrasing during translation. For example, in the case of Figure 1, the English words "are" and "being", and the Swedish word "med", do not have corresponding word-level translation. In addition, it can be perfect to express "as soon as they can" using "very quickly" in a translation, which looses word alignment information because of the longer span. Finally, errors in automatic word alignments can also bring noise. Such alignment errors can directly affect grammaticality of the resulting target treebank due to deletion of unaligned words during post-processing, or cause

*Corresponding author.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 997–1006, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

lost or mistaken dependency arcs.

We consider a different approach for translation-based syntactic knowledge transfer, which aims at making the best use of source syntax with the minimum noise being introduced. To this end, we leverage recent advances in cross-lingual word representations, such as cross-lingual word clusters (Täckström et al., 2012) and cross-lingual word embeddings (Guo et al., 2015), which allow words from different languages to reside within a consistent feature vector space according to structural similarities between words. Thus, they offer a bridge on the lexical level between different languages (Ammar et al., 2016).

A cross-lingual model can be trained by directly using cross-lingual word representations on a source treebank (Guo et al., 2015). Using this method, knowledge transfer can be achieved on the level of token correspondences. We take this approach as a naive baseline. To further introduce structural feature transfer, we transform a source treebank into a code-mixed treebank by considering word alignments between a source sentence and its machine translation. In particular, source words with highly confident target alignments are translated into target words by consulting the machine translation output, so that target word vectors can be directly used for learning target syntax. In addition, continuous spans of target words of the code-mixed treebank are reordered according to the translation, so that target grammaticality can be exploited to the maximum potent.

We conduct experiments on Universal Dependency Treebanks (v2.0) (McDonald et al., 2013; Nivre et al., 2016). The results show that a code-mixed treebank can bring significantly better performance compared to a fully translated treebank, resulting in averaged improvements of 4.30 points on LAS. The code and related data will be released publicly available under Apache License 2.0.[1]

## 2 Related Work

Existing work on cross-lingual transfer can be classified into two categories. The first aims to train a dependency parsing model on source treebanks (McDonald et al., 2011; Guo et al., 2016a,b), or their adapted versions (Zhao et al., 2009; Tiedemann et al., 2014; Wang et al., 2017) in the target language. The second category, namely annotation projection, aims to produce a set of large-scale training instances full of automatic dependencies by parsing parallel sentences (Hwa et al., 2005; Rasooli and Collins, 2015). The two broad methods are orthogonal to each other, and can both make use of the lexicalized dependency models trained with cross-lingual word representations (Rasooli and Collins, 2017, 2019).

**Source Treebank Adaption.** There has been much work on unsupervised cross-lingual dependency parsing by direct source treebank transferring. Several researchers investigate delexicalized models where only non-lexical features are used in the models (Zeman and Resnik, 2008; Cohen et al., 2011; McDonald et al., 2011; Naseem et al., 2012; Täckström et al., 2013; Rosa and Zabokrtsky, 2015). All the features in these models are language independent, and are consistent across languages and treebanks. Thus they can be applied into target languages directly.

Subsequent research proposes to exploit lexicalized features to enhance the parsing models, by resorting to cross-lingual word representations (Täckström et al., 2012; Guo et al., 2015; Duong et al., 2015b,a; Zhang and Barzilay, 2015; Guo et al., 2016b; Ammar et al., 2016; Wick et al., 2016; de Lhoneux et al., 2018). Cross-lingual word clusters and cross-lingual word embeddings are two main sources of features for transferring knowledge between source and target language sentences. These studies enable us to train lexicalized models on code-mixed treebanks as well. Thus here we integrate the cross-lingual word representations as well, which gives more direct interaction between source and target words.

Our work follows another mainstream method of this line of work, namely treebank translation (Tiedemann et al., 2014; Tiedemann, 2015; Tiedemann and Agić, 2016), which aims to adapt an annotated source treebank into the target language by machine translation. In addition, the target-side sentences are produced by machine translation. Previous work aims to build a well-formed tree (Tiedemann and Agić, 2016) from source dependencies, solving word alignment conflicts by heuristic rules. In contrast, we use partial translation instead to avoid unnecessary noise.

**Annotation Projection.** The annotation projection approach relies on a set of parallel sentences between the source and target languages (Hwa et al., 2005; Ganchev et al., 2009). In par-

[1] https://github.com/zhangmeishan/CodeMixedTreebank

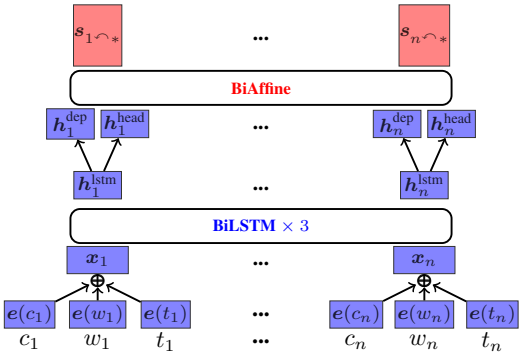


Figure 2: The overall architecture of the BiAffine parser.

ticular, a source parser trained on the source treebank is used to parse the source-side sentences of the parallel corpus. The source dependencies are then projected onto the target sentences according to word alignments. Different strategies can be applied for the dependency projection task (Ma and Xia, 2014; Rasooli and Collins, 2015; Xiao and Guo, 2015; Agić et al., 2016; Schlichtkrull and Søgaard, 2017). For example, one can project only dependency arcs whose words are aligned to target-side words with high confidence (Lacroix et al., 2016). The resulting treebank can be highly noisy due to the auto-parsed source dependency trees. Recently Lacroix et al. (2016) and Rasooli and Collins (2017) propose to filter the results from the large-scale parallel corpus. Our work is different in that the source dependencies are from gold-standard treebanks.

## 3 Dependency Parsing

We adopt a state-of-the-art neural BiAffine parser (Dozat and Manning, 2016) as the baseline, which has achieved competitive performances for dependency parsing. The overall architecture is shown in Figure 2. Given an input sentence $w_1 \cdots w_n$, the model finds its embedding representations of each word $\boldsymbol{x}_1 \cdots \boldsymbol{x}_n$, where

$$\boldsymbol{x}_i = \boldsymbol{e}(w_i) \oplus \boldsymbol{e}(c_i) \oplus \boldsymbol{e}(t_i) \quad (1)$$

Here $c_i$ denotes the word cluster of $w_i$, and $t_i$ denotes the POS tag. We exploit cross-lingual word embeddings, clusters and also universal POS tags, respectively, which are consistent across language and treebanks.

A three-layer deep bidirectional long short term memory (LSTM) neural structure is applied on $\boldsymbol{x}_1 \cdots \boldsymbol{x}_n$ to obtain hidden vectors $\boldsymbol{h}_1^{\text{LSTM}} \cdots \boldsymbol{h}_n^{\text{LSTM}}$. For head finding, two non-linear feed-forward neural layers are used on $\boldsymbol{h}_1^{\text{LSTM}} \cdots \boldsymbol{h}_n^{\text{LSTM}}$ to obtain $\boldsymbol{h}_1^{\text{dep}} \cdots \boldsymbol{h}_n^{\text{dep}}$ and $\boldsymbol{h}_1^{\text{head}} \cdots \boldsymbol{h}_n^{\text{head}}$. We compute the score for each dependency $i \curvearrowleft j$ by:

$$\boldsymbol{s}_{i \curvearrowleft j} = \text{BiAffine}(\boldsymbol{h}_i^{\text{dep}}, \boldsymbol{h}_j^{\text{head}}) \quad (2)$$

The above process is also used for scoring a labeled dependency $i \overset{l}{\curvearrowleft} j$, by extending the 1-dim vector $\boldsymbol{s}$ into $L$ dims, where $L$ is the total number of dependency labels.

When all scores are ready, a softmax function is used at each position for all candidate heads and labels, and then the normalized scores are used for decoding and training. For decoding, we exploit the MST algorithm to ensure tree-structural outputs. For training, we accumulate the cross-entropy loss at the word-level by treating the normalized scores as prediction probabilities. The reader is referred to Dozat and Manning (2016) for more details.

## 4 Code-Mixed Treebank Translation

We derive code-mixed trees from source dependency trees by partial translation, projecting words and the corresponding dependencies having high-confidence alignments with machine-translated target sentences. Our approach assumes that sentence level translations and alignment probabilities are available. The motivation is to reduce noise induced by problematic word alignments.

We adopt the word-level alignment strategy, which has been demonstrated as effective as phrase-level alignment yet much simpler (Tiedemann et al., 2014; Tiedemann, 2015; Tiedemann and Agić, 2016). Give a source sentence $e_1 \cdots e_n$ and its target language translation $f_1 \cdots f_m$, $p(e_i|f_j)$ denotes the probability of word $f_j$ being aligned with $e_i$ ($0 \leq i \leq n$ and $0 < j \leq m$), where $e_0$ denotes a null word, indicating the no alignment probability for one target word.

The translation process can be conducted by three steps:

(1) word substitution, which incrementally substitutes the source words with the target translations;

(2) word deletion, which removes several unaligned source words;

**Algorithm 1** The process of tree translation.

**Input**: $e_1 \cdots e_n$, $f_1 \cdots f_m$, $\mathrm{T}_e$, $\{p(e|f)\}$, $\lambda$
**Output**: $\mathrm{T}_{\text{mix}}$

1: Let $\mathrm{T}_{\text{mix}} = \mathrm{T}_e$.
2: **for** $j \in [1, m]$ **do**
3: $\quad a_j, p_j \leftarrow \text{argmax}_i\, p(e_i|f_j), \max_i p(e_i|f_j)$
4: **end for**
5: $S_{\text{sub}} \leftarrow \textsc{Select}(\{\langle f_j, a_j, \boldsymbol{v}{=}p_j\rangle\}, \lambda)$
6: **for** $f_j, a_j \in S_{\text{sub}}$ **do**
7: $\quad \mathrm{T}_{\text{mix}} \leftarrow \textsc{Substitute}(f_j, a_j, \mathrm{T}_{\text{mix}})$
8: **end for**
9: $D = [1, n] - \{a_j\}|_{j\in[1,m]}$
10: **for** $i \in D$ **do**
11: $\quad r_i = \sum_j p(e_i|f_j)$
12: **end for**
13: $S_{\text{del}} \leftarrow \textsc{Select}(\{\langle e_i, \boldsymbol{v}{=}\text{-}r_i\rangle\}, \lambda)$
14: **for** $e_i \in S_{\text{del}}$ **do**
15: $\quad \mathrm{T}_{\text{mix}} \leftarrow \textsc{Delete}(e_i, \mathrm{T}_{\text{mix}})$
16: **end for**
17: $\mathrm{T}_{\text{mix}} \leftarrow \textsc{Reorder}(\mathrm{T}_{\text{mix}}, f_1 \cdots f_m)$

(3) sentence reordering, which reorders the partially translated sentence, ensuring local target language word order.

Algorithm 1 shows pseudocode for code-mixing tree translation, where line 1-8 denotes the first step, line 9-16 denotes the second and line 17 denotes the last step.

There is a hyper-parameter $\lambda$ to control the overall ratio of translation. The function $\textsc{Select}(S, \lambda)$ is to obtain a subset of $S$ by ratio $\lambda$ with top element values as indicated by $\boldsymbol{v}$ inside $S$. If $\lambda = 0$, it is still the source language dependence tree, since no source word is substituted or deleted. In this condition, our method is equal to Guo et al. (2015) by bridging source and target dependency parsing with universal word representations. If $\lambda = 1$, the resulting tree is a fully translated target dependency tree, as all words are target language produced by translation. In this setting, our method is equal to Tiedemann and Agić (2016) where the only difference the our baseline parsing model. Thus our method can be regarded as a generalization of both source-side training (Guo et al., 2015) and fully translated target training (Tiedemann and Agić, 2016) with fine-grained control over translation confidence.

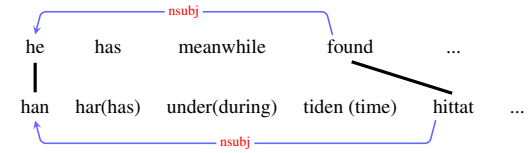


(a) one-to-one.

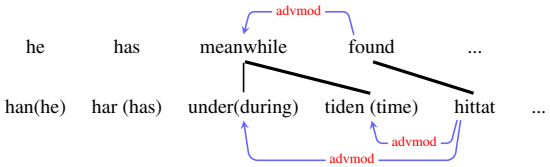


(b) many-to-one.

Figure 3: Examples of word substitution, where the thicker lines indicates more confident alignments.

### 4.1 Word Substitution

Word substitution is the key step for producing a target treebank. We first obtain the most confidently aligned source word $e_{a_j}$ for each target word $f_j$ as well as their alignment probability $p_j = p(e_{a_j}|f_j)$, as shown by line 3 in Algorithm 1. Then we sort the target words by these alignment probabilities, choosing the top $\lceil m\lambda \rceil$ words with highest alignment probabilities for substitution. The sorting and choosing is reflected in line line 5 of Algorithm 1. Finally for each chosen word $f_j$ and its aligned source word $e_{a_j}$ , we replace the source word $e_{a_j}$ by $f_j$, as shown by line 7 in Algorithm 1.

One key of the substitution is to maintain the corresponding dependency structures. If $e_{a_j}$ and $f_j$ bares a one-one mapping, with no other target word being aligned with $e_{a_j}$, the source dependencies are kept unchanged, as shown by Figure 3(a). While if two or more words (i.e., $f_{j_1}, \cdots, f_{j_k}(j_1 < \cdots < j_k)$) are aligned to $e_{a_j}$, we simply link all words to $e_{a_j}$, with the same dependency label as the original dependency arc. Figure 3(b) illustrates this condition. Both the Swedish words "under" and "tiden" are headed to "hittat" (the Swedish translation of English word "found") by the dependency label "advmod" inherited from the source English side . Note that the POS tags of the substituted words are the same as the corresponding source words.

### 4.2 Word Deletion

There can be source words to which no target word is aligned. These words are typically functional words belonging to source language only, such as

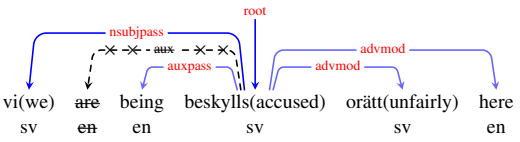


Figure 4: An example of word deletion.

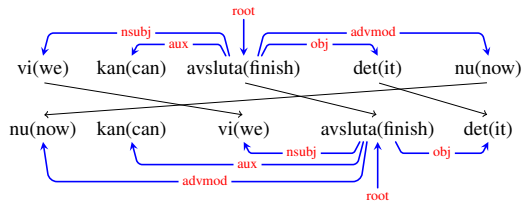


(a) full sentence

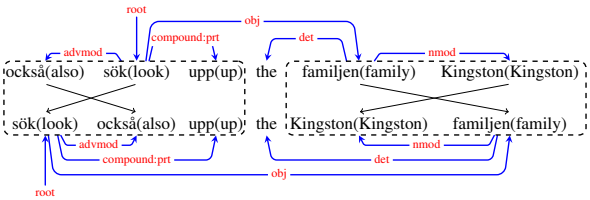


(b) two spans

Figure 5: Two examples of sentence reorder.

"the", "are" and "have". We remove such words to produce dependency trees that are close in syntax to the target language.

In particular, we accumulate the probabilities of $p(e_i|f_j)$ for the source word $e_i$ who has no aligned target word:

$$r_i = \sum_j p(e_i|f_j) \quad \text{(line 11 in Algorithm 1)},$$

where we traverse all target words to sum their alignment probabilities with $e_i$. The value of $r_i$ can be interpreted as the confidence score of retention. The words with lower retention scores should be more preferred to be deleted, as these words have lower probabilities aligning with some word of the target language sentence. Concretely, we collect all source words with no aligned target words, computing their retention values, and the selecting a subset of these words with the lowest retention values by the hyper-parameter $\lambda$ (line 13 in Algorithm 1). Finally we delete all the selected words (line 15 in Algorithm 1).

Figure 4 shows an example of word deletion. The two words "are" and "being" both have no aligned words in the other side, and meanwhile "are" has a lower retention score compared with "being".[2] Thus the source word "are" is prefer to be deleted. In most cases, the deleted words are leaf nodes, which can be unattached to the resulted dependency tree and deleted them directly. In case of exceptions, we simply reset the corresponding heads of its child nodes by the head of $e_i$ (i.e., $\mathrm{h}_{e_i}$) instead. For example, a dependency $w_i^{\curvearrowleft} e_i$ is changed into $w_i^{\curvearrowleft} \mathrm{h}_{e_i}$.

### 4.3 Sentence Reordering

Continuous target spans are reordered to make the final code-mixed sentence contain grammatical phrases in the target language. Figure 5(a) shows one example of full sentence reordering. We can see that the word order by word-level substitutions on the source words is different with the order of the machine-translated sentence. Thus we adjust the leaf nodes, letting the word order strictly follow the machine-translated sentence order For example, the word "vi" is moved from the first position into the third position, and similarly the word "mu" is moved from the last position into the first position.

Concretely, we perform the word reorder by the span level, extracting all the continuous spans of target words, because the target language words may be interrupted by source language words. Then we reorder the words in each span according to their order in the machine translation outputs. Figure 5(b) shows another example, where there are two spans separated by the English word "the". Each span is reordered individually. We do not consider the inconsistent orders inter the spans in this work. Note that this step does not change any dependency arc between words.

## 5 Experiments

We conduct experiments to verify the effectiveness of our proposed models in this section.

### 5.1 Settings

Our experiments are conducted on the Google Universal Dependency Treebanks (v2.0) (McDonald et al., 2013; Nivre et al., 2016), using English as the source language, and choosing six languages, including Spanish (ES), German (DE), French (FR), Italian (IT), Portuguese (PT) and Swedish (sv), as the target languages. Google

[2] Both the two words are only related to the Swedish word "med", but $p$(are|med) is slightly lower.

Translate[3] is used to translate the sentences in the English training set into other languages. In order to generate high-quality word-level alignments, we merge the translated sentence pairs and the parallel data of EuroParl (Koehn, 2005) to obtain word alignments. We use the fastAlign tool (Dyer et al., 2013) to obtain word alignments.

We use the cross-lingual word embeddings and clusters by Guo et al. (2016b) for the baseline system. The dimension size of word embeddings is 50 and the word cluster number across of all languages is 256.

For network building and training, we use the same setting as Dozat and Manning (2016), including the dimensional sizes, the dropout ratio, as well as the parameter optimization method. We assume that no labeled corpus is available for the target language. Thus training is performed for 50 iterations over the whole training data without early-stopping.

To evaluate dependency parsing performances, we adopt UAS and LAS as the major metrics, which indicate the accuracies of unlabeled dependencies and labeled dependencies, respectively. We ignore the punctuation words during evaluation following previous work. We run each experiment 10 times and report the averaged results.

### 5.2 Models

We compare performances on the following models:

- `Delex` (McDonald et al., 2013): The delexicalized BiAffine model without cross-lingual word embeddings and clusters.
- `Src` (Guo et al., 2015): The BiAffine model trained on the source English treebank only.
- `PartProj` (Lacroix et al., 2016): The BiAffine model trained on the corpus by projecting only the source dependencies involving high-confidence alignments into target sentences. Note that the baseline only draws the idea from Lacroix et al. (2016), and the two models are significant different in fact.
- `Tgt` (Tiedemann and Agić, 2016): The BiAffine model trained on the fully translated target treebank only.

[3]https://translate.google.com/ at Oct, 2018

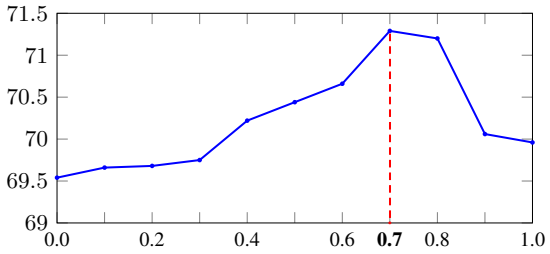


Figure 6: The LAS relative to the translation ratio $\lambda$.

| Model | UAS | LAS |
|---|---|---|
| `Src` | 79.52 | 69.54 |
| `Tgt` | 79.34 | 69.96 |
| **`Mix`** | **80.33** | **71.29** |
| `Src + Tgt` | 80.12 | 71.16 |
| **`Src + Mix`** | **80.91** | **71.73** |

Table 1: Experiments of corpus mixing.

- `Src+Tgt`: The BiAffine model trained on the combination dataset of the source and fully translated target treebanks.
- `Mix`: The BiAffine model trained on the code-mixed treebank only.
- `Src+Mix`: The BiAffine model trained on the combination dataset of the source and code-mixed treebanks.

The `Src` and `Tgt` methods have been discussed in Section 4. The `PartProj` model is another way to leverage imperfect word alignments (Lacroix et al., 2016). The training corpus of `PartProj` may be incomplete dependency trees with a number of words missing heads, because no word is deleted from machine translation outputs. The POS tags of words in `PartProj` with low-confidence alignments are obtained by a supervised POS tagger (Yang et al., 2018) trained on the corresponding universal treebank.

### 5.3 Development Results

We conduct several developmental experiments on the Swedish dataset to examine important factors to our model.

#### 5.3.1 Influence of The Translation Ratio $\lambda$

Our model has an important hyper-parameter $\lambda$ to control the percentage of translation. Figure 6 shows the influence of this factor, where the percentages increase from 0 to 1 by intervals of 0.1. A

| Model | UAS | LAS |
|---|---|---|
| **Mix** | **80.33** | **71.29** |
| −Sentence Reordering | 79.79 | 70.47 |
| −Word Deletion | 79.82 | 70.64 |
| −Both | 79.46 | 69.59 |

Table 2: Ablation experiments.

$\lambda$ of 0 gives our baseline by using the source treebank only. As the $\lambda$ grows, more source words are translated into the target. We can see that the performance improves after translating some source dependencies into the target, demonstrating the effectiveness of syntactic transferring. The performance reaches the peak when $\lambda = 0.7$, but there is a significant drop when $\lambda$ grows from $0.8$ to $0.9$. This can be because the newly added dependency arc projections are mostly noisy. This sharp decrease indicates that noise from low-confidence word alignments can have strong impact on the performance. According to the results, we adopt $\lambda = 0.7$ for code-mixed treebanking.

#### 5.3.2 Mixing with Source TreeBank

We investigate the effectiveness of the source treebank by merging it into the translated treebanks. First, we show the model performances of `Src`, `Tgt` and `Mix`, which are trained on the individual treebanks, respectively. Then we merge the source treebank with the two translated treebanks, and show the results trained on the merging corpora. Table 1 shows the results. According to the results, we can find that the source treebank is complementary with the translated treebanks. Noticeably, although `Src + Mix` gives the best performance, its improvement over `Mix` is relatively smaller than that of `Src + Tgt` over `Tgt`. This is reasonable as the code-mixed treebank contains relatively more source treebank content than the fully translated target treebank.

#### 5.3.3 Ablation Studies

The overall translation is conducted by three steps as mentioned in Section 4, where the first word substitution is compulsory, and the remaining two steps aim to build better mixed dependency trees. Here we conduct ablation studies to test the effectiveness of word deletion and sentence reordering. Table 2 shows the experimental results. We can see both steps are important for dependency tree translation. Without word deletion and sentence reordering, the `mix` model shows decreases of 0.82 and 0.65 on LAS, respectively. If both are removed, the performance is only comparable with the baseline `src` model (see Table 1).

### 5.4 Final Results

We show the final results of our proposed models in Table 3. As shown, the model `Tgt` gives better averaged performance compared to `Src`. However, its results on French and Italian are slightly worse, which indicates that noise from translation impacts the quality of the projected treebank. The model `Mix` gives much better performance compared with `Src`, demonstrating the effectiveness of structural transfer. `Mix` also outperforms `Tgt` significantly, obtaining average increases of $80.93 - 76.37 = 4.56$ points on UAS and $72.87 - 68.28 = 4.59$ points on LAS, respectively. The best setting is the model `Src + Mix`, trained on the combined corpus of the source and code-mixed treebanks, which gives better performance than solely code-mixed treebanks.

By comparing with the delexicalized model `Delex`, we can see that lexicalized features are highly useful for cross-lingual transfer. For the `PartProj` model, we conduct preliminary experiments on Swedish to tune the ratio of the projected dependencies. The results show that the difference is very small ($\delta = 0.24$ for UAS) between 0.9 to 1.0, and the performance degrades significantly as the ratio decreases below 0.9. The observation indicates that this method is probably not effective for filtering low-confidence word alignments. The final results confirm our hypothesis. As shown in Table 3, the `PartProj` model gives only comparable performance with `Src`. One possible reason may be the unremoved target words (if the words are removed, the `PartProj` model with ratio 1.0 will be identical to `Tgt`), which have been demonstrated noisy previously (Tiedemann and Agić, 2016).

### 5.5 Comparison with Previous Work

We compare our method with previous work in the literature. Table 4 shows the results, where the UAS values are reported. Our model denoted by **`This`** refers to the model of `Src + Mix`. Note that these models are not directly comparable due to the setting and baseline parser differences. The first block shows several models by directly transferring gold-standard source treebank knowledge into the target side, including the models of

| Lang. | Delex | | PartProj | | Src | | Tgt | | Src + Tgt | | Mix | | **Src + Mix** | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | UAS | LAS | UAS | LAS | UAS | LAS | UAS | LAS | UAS | LAS | UAS | LAS | UAS | LAS |
| DE | 64.10 | 53.77 | 69.90 | 61.28 | 66.87 | 57.46 | 70.84 | 62.30 | 72.41 | 63.74 | 71.41 | 63.46 | **72.78** | **64.38** |
| ES | 71.53 | 63.33 | 75.81 | 66.83 | 75.63 | 65.85 | 76.49 | 67.39 | 77.00 | 67.95 | 81.18 | 71.80 | **81.44** | **71.66** |
| FR | 75.13 | 67.26 | 75.54 | 67.63 | 78.13 | 70.63 | 76.91 | 69.39 | 78.75 | 71.17 | 83.20 | 76.32 | **83.77** | **76.48** |
| IT | 77.71 | 69.27 | 77.71 | 69.27 | 81.11 | 72.83 | 79.30 | 71.65 | 81.56 | 74.09 | 85.30 | 77.43 | **86.13** | **78.38** |
| PT | 74.03 | 67.70 | 79.44 | 71.30 | 77.37 | 69.36 | 78.32 | 70.67 | 79.73 | 71.84 | 83.54 | 75.34 | **84.05** | **75.89** |
| AVG | 72.50 | 64.27 | 75.68 | 67.26 | 75.82 | 67.23 | 76.37 | 68.28 | 77.89 | 69.76 | 80.93 | 72.87 | **81.63** | **73.36** |

Table 3: Final results.

| Model | DE | ES | FR | IT | PT |
|---|---|---|---|---|---|
| TreeBank Transferring | | | | | |
| This | 72.78 | **81.44** | **83.77** | **86.13** | **84.05** |
| Guo15 | 60.35 | 71.90 | 72.93 | – | – |
| Guo16 | 65.01 | 79.00 | 77.69 | 78.49 | 81.86 |
| TA16 | **75.27** | 76.85 | 79.21 | – | – |
| Annotation Projection | | | | | |
| MX14 | 74.30 | 75.53 | 70.14 | 77.74 | 76.65 |
| RC15 | **79.68** | **80.86** | **82.72** | **83.67** | **82.07** |
| LA16 | 75.99 | 78.94 | 80.80 | 79.39 | – |
| TreeBank Transferring + Annotation Projection | | | | | |
| **RC17** | **82.1** | **82.6** | **83.9** | **84.4** | **84.6** |

Table 4: Comparison with previous work (UAS).

Guo15 (Guo et al., 2015), Guo16 (Guo et al., 2016b) and TA16 (Tiedemann and Agić, 2016). Our model gives the best performance with one exception on the German language. One possible reason may be that TA16 has exploited multiple sources of treebanks besides English.

The second block shows representative annotation projection models, including MX14 (Ma and Xia, 2014), RC15 (Rasooli and Collins, 2015), LA16. The models of annotation projection can be complementary with our work, since they build target training corpus from raw parallel texts. The best-performed results of the RC17 model (Rasooli and Collins, 2017) have demonstrated this point, which can be regarded as a combination of the dictionary-based treebank translation[4] (Zhao et al., 2009) and RC15.

## 5.6 Analysis

We conduct experimental analysis on the Spanish (ES) dataset to show the differences between the models of Src, Tgt and Mix.

[4]The method has been demonstrated worse than TA16 in Tiedemann and Agić (2016).

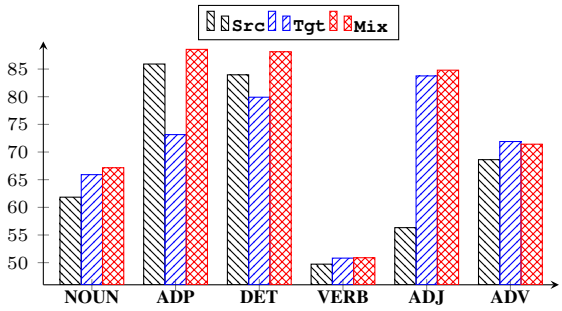


Figure 7: Performance relative to POS tags (F-score).

### 5.6.1 Performance Relative to POS Tags

Figure 7 show the F-scores of labeled dependencies on different POS tags. We list the six most representative POS tags. The Mix model achieves the best F-scores on 5 of 6 POS tags, with the only exception on tag ADV which has no significant difference with the Tgt model. The Mix and Tgt models are much better than the Src model as a whole, especially on the POS tag ADJ where an increase of over 20% has been achieved. In addition, we find that the Src model can significantly outperform the Tgt model on ADP and DET. For Spanish, ADP words are typically "de", "en", "con" and etc., which behave similarly to the English words such as "'s", "to" and "of". The Spanish words of DET include "el", "la", "su" and etc., which are similar to the English words such as "the" and "a". These words are highly ambiguous for automatic word alignment. The results indicate that our Mix model can better handle these word alignment noise, mitigating their negative influence of treebank translation, while the Tgt model suffers from such noise.

### 5.6.2 Performance Relative to Arc Distances

Figure 8 show the F-scores of labeled dependencies by different arc distances. Particularly we treat the root type as one special case. According

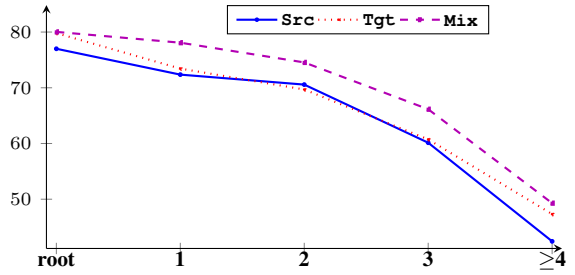


Figure 8: Performance relative to arc distances (F-score).

to the results, the `Mix` model performs the best over all distances, indicating its effectiveness on treebank transferring. The `Tgt` model achieves better performance than the `Src` model with one exception on distance 2. We look into the dependency patterns of distance 2 arcs further, finding that the dependency arc **ADP↶*** accounts for over 30%, and it is the major source of errors. The finding is consistent with that on POS tags, denoting the effectiveness of the code-mixed treebank in handling noise. In addition, as the distance increases the performance drops gradually. The F-score of root dependency is the highest.

## 6 Conclusion

We proposed a new treebank translation method for unsupervised cross-lingual dependency parsing. Unlike previous work, which adopts full-scale translation for source dependency trees, we investigated partial translation instead, producing synthetic code-mixed treebanks. The method can better leverage imperfect word alignments between source and target sentence pairs, translating only high-confidence source sentential words, thus generating dependencies in high-quality. Experimental results on Universal Dependency Treebak v2.0 showed that partial translation is highly effective, and code-mixed treebanks can give significantly better results than full-scale translation.

Our method is complementary with several other methods for cross-lingual transfer, such as annotation projection, and thus can be further integrated with these methods.

## Acknowledgments

We thank all anonymous reviewers for their valuable comments. Several suggestions are not integrated in this version, i.e., more experiments on really low-resource languages, and detailed analysis on more languages. We will supplement later on the webpages of the authors. This work is supported by National Natural Science Foundation of China (NSFC) grants 61602160, U1836222 and 61672211.

## References

Željko Agić, Anders Johannsen, Barbara Plank, Héctor Martínez Alonso, Natalie Schluter, and Anders Søgaard. 2016. Multilingual projection for parsing truly low-resource languages. *TACL*, 4:301–312.

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A Smith. 2016. Many languages, one parser. *TACL*, 4:431–444.

Shay B Cohen, Dipanjan Das, and Noah A Smith. 2011. Unsupervised structure prediction with non-parallel multilingual guidance. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 50–61. Association for Computational Linguistics.

Timothy Dozat and Christopher D Manning. 2016. Deep biaffine attention for neural dependency parsing. *arXiv preprint arXiv:1611.01734*.

Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015a. Cross-lingual transfer for unsupervised dependency parsing without parallel data. In *Proceedings of the CONLL*, pages 113–122.

Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015b. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proceedings of the 53rd ACL*, volume 2, pages 845–850.

Chris Dyer, Victor Chahuneau, and Noah A Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of the NAACL*, pages 644–648.

Kuzman Ganchev, Jennifer Gillenwater, and Ben Taskar. 2009. Dependency grammar induction via bitext projection constraints. In *Proceedings of the Joint Conference of ACL-IJCNLP*, pages 369–377.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *Proceedings of ACL-IJCNLP*, pages 1234–1244.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2016a. A distributed representation-based framework for cross-lingual transfer parsing. *Journal of Artificial Intelligence Research*, 55:995–1023.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2016b. A representation learning framework for multi-source transfer parsing. In *AAAI*.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural language engineering*, 11(3):311–325.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86.

Ophélie Lacroix, Lauriane Aufrant, Guillaume Wisniewski, and François Yvon. 2016. Frustratingly easy cross-lingual transfer for transition-based dependency parsing. In *Proceedings of the NAACL*, pages 1058–1063.

Miryam de Lhoneux, Johannes Bjerva, Isabelle Augenstein, and Anders Søgaard. 2018. Parameter sharing between dependency parsers for related languages. In *Proceedings of the EMNLP*, pages 4992–4997.

Xuezhe Ma and Fei Xia. 2014. Unsupervised dependency parsing with transferring distribution via parallel guidance and entropy regularization. In *Proceedings of ACL*, volume 1, pages 1337–1348.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, et al. 2013. Universal dependency annotation for multilingual parsing. In *Proceedings of the 51st ACL*, volume 2, pages 92–97.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the conference on EMNLP*, pages 62–72.

Tahira Naseem, Regina Barzilay, and Amir Globerson. 2012. Selective sharing for multilingual dependency parsing. In *Proceedings of the 50th ACL*, pages 629–637.

Joakim Nivre, Marie-Catherine De Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan T McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal dependencies v1: A multilingual treebank collection. In *LREC*.

Mohammad Sadegh Rasooli and Michael Collins. 2015. Density-driven cross-lingual transfer of dependency parsers. In *Proceedings of the 2015 EMNLP*, pages 328–338.

Mohammad Sadegh Rasooli and Michael Collins. 2017. Cross-lingual syntactic transfer with limited resources. *TACL*, 5:279–293.

Mohammad Sadegh Rasooli and Michael Collins. 2019. Low-resource syntactic transfer with unsupervised source reordering. *arXiv preprint arXiv:1903.05683*.

Rudolf Rosa and Zdenek Zabokrtsky. 2015. Klcpos3-a language similarity measure for delexicalized parser transfer. In *Proceedings of the 53rd ACL*, volume 2, pages 243–249.

Michael Schlichtkrull and Anders Søgaard. 2017. Cross-lingual dependency parsing with late decoding for truly low-resource languages. In *Proceedings of the 15th Conference of the EACL*, volume 1, pages 220–229.

Oscar Täckström, Ryan McDonald, and Joakim Nivre. 2013. Target language adaptation of discriminative transfer parsers. In *Proceedings of the NAACL*, pages 1061–1071.

Oscar Täckström, Ryan McDonald, and Jakob Uszkoreit. 2012. Cross-lingual word clusters for direct transfer of linguistic structure. In *Proceedings of the NAACL*, pages 477–487.

Jörg Tiedemann. 2015. Improving the cross-lingual projection of syntactic dependencies. In *Proceedings of the 20th NODALIDA*, 109, pages 191–199. Linköping University Electronic Press.

Jörg Tiedemann and Zeljko Agić. 2016. Synthetic treebanking for cross-lingual dependency parsing. *Journal of Artificial Intelligence Research*, 55:209–248.

Jörg Tiedemann, Željko Agić, and Joakim Nivre. 2014. Treebank translation for cross-lingual parser induction. In *Proceedings of the EMNLP*, pages 130–140.

Hongmin Wang, Yue Zhang, GuangYong Leonard Chan, Jie Yang, and Hai Leong Chieu. 2017. Universal dependencies parsing for colloquial singaporean english. In *Proceedings of the 55th ACL*.

Michael Wick, Pallika Kanani, and Adam Pocock. 2016. Minimally-constrained multilingual embeddings via artificial code-switching. In *AAAI*.

Min Xiao and Yuhong Guo. 2015. Annotation projection-based representation learning for cross-lingual dependency parsing. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 73–82.

Jie Yang, Shuailong Liang, and Yue Zhang. 2018. Design challenges and misconceptions in neural sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3879–3889.

Daniel Zeman and Philip Resnik. 2008. Cross-language parser adaptation between related languages. In *Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages*.

Yuan Zhang and Regina Barzilay. 2015. Hierarchical low-rank tensors for multilingual transfer parsing. In *Proceedings of the 2015 Conference on EMNLP*, pages 1857–1867.

Hai Zhao, Yan Song, Chunyu Kit, and Guodong Zhou. 2009. Cross language dependency parsing using a bilingual lexicon. In *Proceedings of ACL-IJCNLP*, pages 55–63.

# Hierarchical Pointer Net Parsing

**Linlin Liu**[12*†], **Xiang Lin**[1†], **Shafiq Joty**[13], **Simeng Han**[1], **Lidong Bing**[2]
[1]Nanyang Technological University, Singapore
[2]R&D Center Singapore, Machine Intelligence Technology, Alibaba DAMO Academy
[3]Salesforce Research Asia, Singapore
{linx0057, srjoty, hans0035}@ntu.edu.sg
{linlin.liu, l.bing}@alibaba-inc.com

## Abstract

Transition-based top-down parsing with pointer networks has achieved state-of-the-art results in multiple parsing tasks, while having a linear time complexity. However, the decoder of these parsers has a sequential structure, which does not yield the most appropriate inductive bias for deriving tree structures. In this paper, we propose hierarchical pointer network parsers, and apply them to dependency and sentence-level discourse parsing tasks. Our results on standard benchmark datasets demonstrate the effectiveness of our approach, outperforming existing methods and setting a new state-of-the-art.

## 1 Introduction

Parsing of sentences is a core natural language understanding task, where the goal is to construct a tree structure that best describes the relationships between the tree constituents (*e.g.,* words, phrases). For example, Figure 1 shows examples of a **dependency tree** and a sentence-level **discourse tree** that respectively represent how the words and clauses are related in a sentence. Such parse trees are directly useful in numerous NLP applications, and also serve as intermediate representations for further language processing tasks such as semantic and discourse processing.

Existing approaches to parsing can be distinguished based on whether they employ a greedy **transition-based** algorithm (Marcu, 1999; Zhang and Nivre, 2011; Wang et al., 2017) or a globally optimized algorithm such as **graph-based** methods for dependency parsing (Eisner, 1996) or **chart parsing** for discourse (Soricut and Marcu, 2003; Joty et al., 2015). Transition-based parsers build the tree incrementally by making a series of shift-reduce decisions. The advantage of this method is that the parsing time is linear with respect to the sequence length. The limitation, however, is that the decisions made at each step are based on local information, disallowing the model to capture long distance dependencies and also causing error propagation to subsequent steps. Recent methods attempt to address this issue using neural networks capable of remembering long range relationships such as Stacked LSTMs (Dyer et al., 2015; Ballesteros et al., 2015) or using globally normalized models (Andor et al., 2016).

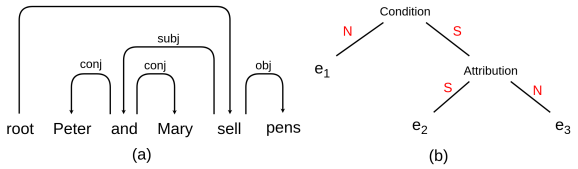


Figure 1: (a) A dependency tree for a sentence; (b) a discourse tree for the sentence "[Now that's name-dropping,]$_{e_1}$ [if you know]$_{e_2}$ [what I mean.]$_{e_3}$", where 'S' denotes Satellite and 'N' denotes Nucleus.

The globally optimized methods, on the other hand, learn scoring functions for subtrees and perform search over all possible trees to find the most probable tree for a text. Recent graph-based methods use neural models as scoring functions (Kiperwasser and Goldberg, 2016; Dozat and Manning, 2017). Despite being more accurate than greedy parsers, these methods are generally slow having a polynomial time complexity ($O(n^3)$ or higher).

Recently, transition-based **top-down** parsing with Pointer Networks (Vinyals et al., 2015) has attained state-of-the-art results in both dependency and discourse parsing tasks with the same computational efficiency (Ma et al., 2018; Lin et al., 2019); thanks to the **encoder-decoder** architecture that makes it possible to capture information from the whole text and the previously derived subtrees,

[*] Linlin Liu is under the Joint PhD Program between Alibaba and Nanyang Technological University.
[†] Equal contribution.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1007–1017,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

while limiting the number of parsing steps to linear. However, the decoder of these parsers has a sequential structure, which may not yield the most appropriate **inductive bias** for deriving a hierarchical structure. For example, when decoding *"pens"* in Figure 1 in a top-down depth-first manner, the decoder state is directly conditioned on *"and"* as opposed to the states representing its parent *"sell"*. This on one hand may induce irrelevant information in the current state, on the other, as the text length gets longer, the decoder state at later steps tends to forget more relevant information due to long distance. Having an explicit hierarchical inductive bias should allow the model to receive more relevant information and help with the long-term dependency problem by providing shortcuts for gradient back-propagation.

In this paper, we propose a **Hierarchical Pointer Network (H-PtrNet)** parser to address the above mentioned limitations. In addition to the sequential dependencies, our parser also directly models the parent-child and sibling relationships in the decoding process. We apply our proposed method to both dependency and discourse parsing tasks. To verify the effectiveness of our approach, we conduct extensive experiments and analysis on both tasks. Our results demonstrate that in dependency parsing, our model outperforms in most of the languages. In discourse parsing, we push forward the state-of-the-art in all evaluation metrics. Furthermore, our results on the hardest task of relation labeling have touched human agreement scores on this task. We have released our code at https://ntunlpsg.github.io/project/parser/ptrnet-depparser/ for research purposes.

## 2 Background

### 2.1 Dependency Parsing

Dependency parsing is the task of predicting the existence and type of linguistic dependency relations between words in a sentence (Figure 1a). Given an input sentence, the output is a tree that shows relationships (*e.g.,* NOMINAL SUBJECT (NSUBJ), DETERMINER (DET)) between *head* words and words that modify those heads, called *dependents* or *modifiers*.

Approaches to dependency parsing can be divided into two main categories: greedy transition-based parsing and graph-based parsing. In both paradigms, neural models have proven to be more effective than feature-based models where selecting the composition of features is a major challenge. Kiperwasser and Goldberg (2016) proposed graph-based and transition-based dependency parsers with a Bi-LSTM feature representation. Since then much work has been done to improve these two parsers. Dozat and Manning (2017) proposed a **bi-affine** classifier for the prediction of arcs and labels based on graph-based model, and achieved state-of-the-art performance. Nguyen and Verspoor (2018) adopted a joint modeling approach by adding a Bi-LSTM POS tagger to generate POS tags for the graph-based dependency parser. Though transition-based methods are superior in terms of time complexity, they fail in capturing the global dependency information when making decisions. To address this issue, Andor et al. (2016) proposed a globally optimized transition-based model. Recently, by incorporating a stack within a pointer network, Ma et al. (2018) proposed a transition-based model and achieved state-of-the-art performance across many languages .

### 2.2 Discourse Parsing

Rhetorical Structure Theory or RST (Mann and Thompson, 1988) is one of the most influential theories of discourse, which posits a tree structure (called discourse tree) to represent a text (Fig. 1b). The leaves of a discourse tree represent contiguous text spans called Elementary Discourse Units (EDUs). The adjacent EDUs and larger units are recursively connected by coherence relations (*e.g.,* CONDITION, ATTRIBUTION). Furthermore, the discourse units connected by a relation are distinguished based on their relative importance — NUCLEUS refers to the core part(s) while SATELLITE refers to the peripheral one. Coherence analysis in RST consists of two subtasks: (*a*) identifying the EDUs in a text, referred to as **Discourse Segmentation**, and (*b*) building a discourse tree by linking the EDUs hierarchically, referred to as **Discourse Parsing**. This work focuses on the more challenging task of discourse parsing assuming that EDUs have already been identified. In fact, state-of-the-art segmenter (Lin et al., 2019) has already achieved 95.6 $F_1$ on RST discourse treebank, where the human agreement is 98.3 $F_1$.

Earlier methods have mostly utilized handcrafted lexical and syntactic features (Soricut and Marcu, 2003; Feng and Hirst, 2014; Joty et al., 2015; Wang et al., 2017). Recent approaches have

shown competitive results with neural models that are able to automatically learn the feature representations in an end-to-end fashion (Ji and Eisenstein, 2014; Li et al., 2014). Very recently, Lin et al. (2019) propose a parser based on pointer networks and achieve state-of-the-art performance.

***Remark.*** Although related, the dependency and RST tree structures (hence the parsing tasks) are different. RST structure is similar to constituency structure. Therefore, the differences between constituency and dependency structures also hold here.[1] First, while dependency relations can only be between words, discourse relations can be between elementary units, between larger units or both. Second, in dependency parsing, any two words can be linked, whereas RST allows connections only between two adjacent units. Third, in dependency parsing, a head word can have multiple modifier words, whereas in discourse parsing, a discourse unit can be associated with only one connection. The parsing algorithm needs to be adapted to account for these differences.

### 2.3 Pointer Networks.

Pointer networks (Vinyals et al., 2015) are a class of encoder-decoder models that can tackle problems where the output vocabulary depends on the input sequence. They use attentions as pointers to the input elements. An encoder network first converts the input sequence $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ into a sequence of hidden states $\boldsymbol{H} = (\boldsymbol{h}_1, \ldots, \boldsymbol{h}_n)$. At each time step $t$, the decoder takes the input from previous step, generates a decoder state $\boldsymbol{d}_t$, and uses it to attend over the input elements. The attention gives a $\mathrm{softmax}$ distribution over the input elements.

$$s_{t,i} = \sigma(\boldsymbol{d}_t, \boldsymbol{h}_i); \quad \boldsymbol{a}_t = \mathrm{softmax}(\boldsymbol{s}_t) \quad (1)$$

where $\sigma(.,.)$ is a scoring function for attention, which can be a neural network or an explicit formula like dot product. The model uses $\boldsymbol{a}_t$ to infer the output: $\hat{\mathrm{y}}_t = \arg\max(\boldsymbol{a}_t) = \arg\max p(\mathrm{y}_t|\mathrm{y}_{<t}, \boldsymbol{X}, \theta)$ where $\theta$ is the set of parameters. To condition on $\mathrm{y}_{t-1}$, the corresponding input $\boldsymbol{x}_{\mathrm{y}_{t-1}}$ is copied as the input to the decoder.

[1] There are also studies that use dependency structure to directly represent the relations between the EDUs; see (Muller et al., 2012; Li et al., 2014; Morey et al., 2018).

## 3 Hierarchical Pointer Networks

Before presenting our proposed hierarchical pointer networks, we first revisit how pointer networks have been used for parsing tasks.

### 3.1 Pointer Networks for Parsing.

Ma et al. (2018) and Lin et al. (2019) both use a pointer network as the backbone of their parsing models and achieve state-of-the-art performance in dependency and discourse parsing tasks, respectively. As shown in Figures 2(a) and 3, in both cases, the parsing algorithm is implemented in a top-down depth-first order. They share the same encoder-decoder structure. A bi-directional Recurrent Neural Network (RNN) encodes a sequence of word embeddings $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ into a sequence of hidden states $\boldsymbol{H} = (\boldsymbol{h}_1, \ldots, \boldsymbol{h}_n)$. The decoder implements a unidirectional RNN to greedily generate the corresponding tree. It maintains a stack to keep track of the units that yet need to be parsed, *i.e.,* head words for dependency parsing and larger units for discourse parsing. At each step $t$, the decoder takes out an element from the stack and generates a decoder state $\boldsymbol{d}_t$, which is in turn used in the pointer layer to compute the attention over the relevant input elements. In the case of dependency parsing, the representation of the head word is used to find its dependent. For discourse parsing, it uses the representation of the span to identify the break position that splits the text span into two subspans.

In addition to the tree structure, the parser also deals with the corresponding labelling tasks. Whenever the pointer network yields a newly created pair (*i.e.,* head-dependent in dependency parsing, two sub-spans in discourse parsing), a separate classifier is applied to predict the corresponding relation between them.

### 3.2 Limitations of Existing Methods

One crucial limitation of the existing models is that the decoder has a linear structure, although the task is to construct a hierarchical structure. This can be noticed in the Figures 2 and 3, where the current decoder state $\boldsymbol{d}_t$ is conditioned on the previous state $\boldsymbol{d}_{t-1}$ (see horizontal blue lines), but not on its parent's decoder state or siblings' decoder state, when it was pointed from its head. This can induce irrelevant information if the previous decoding state corresponds to an element that is not

relevant to the current element. For example, in Figure 2, the decoder state for pointing to "pens" is conditioned on the state used for pointing to "and", but not the one used for pointing to "sell", which is more relevant according to the dependency structure. Also, the decoder state for "sell" is far apart from the one for "pens". Therefore, more relevant information could be diminished in a sequential decoder, especially for long range dependencies.

### 3.3 Hierarchical Decoder

To address the above issues, we propose hierarchical pointer network (H-PtrNet), which poses a hierarchical decoder that reflects the underlying tree structure. H-PtrNet has the same encoder-decoder architecture as the original pointer network except that each decoding state $\boldsymbol{d}_t$ is conditioned directly on its parent's decoder state $\boldsymbol{d}_{p(t)}$ and its immediate sibling's decoder state $\boldsymbol{d}_{s(t)}$ in addition to the previous decoder state $\boldsymbol{d}_{t-1}$ and parent's encoder state $\boldsymbol{h}_{p(t)}$ (from input). Formally, the pointing mechanism in H-PtrNet can be defined as:

$$\boldsymbol{d}_t = f(\boldsymbol{d}_{p(t)}, \boldsymbol{d}_{s(t)}, \boldsymbol{d}_{t-1}, \boldsymbol{h}_{p(t)}) \quad (2)$$

$$s_{t,i} = \sigma(\boldsymbol{d}_t, \boldsymbol{h}_i) \quad (3)$$

$$p(\mathrm{y}_t|\mathrm{y}_{<t}, \boldsymbol{X}, \theta) = \mathrm{softmax}(\boldsymbol{s}_t) = \frac{\exp(s_{t,i})}{\sum_i \exp(s_{t,i})} \quad (4)$$

where $f(.)$ is a fusion function to combine the four components into a decoder state, and other terms are similarly defined as before for Eq. 1. Figure 2(b) shows an example of H-PtrNet decoder connections for dependency parsing.

The fusion function $f(.)$ can be implemented in multiple ways and may depend on the specific parsing task. More variants of the fusion function will be discussed in Section 4.

**Decoder Time Complexity.** Given a sentence of length $n$, the number of decoding steps to build a parse tree is linear. The attention mechanism at each decoding step computes an attention vector of length $n$. The overall decoding complexity is $O(n^2)$, which is same as the StackPointer Parser (Ma et al., 2018).

***Remark.*** If we look at the decoding steps of the StackPointer Parser (Ma et al., 2018) more closely, we notice that it also takes the decoder state of the immediate sibling (when it points to itself). This decoder state represents the state after all its children are generated. Thus it contains information about its children. In contrast, in our model we consider the decoder state when the sibling was first generated from its parent. Therefore this state contains the sibling's parent information, which helps with capturing long term dependencies.

### 3.4 Model Specifics for Dependency Parsing

Figure 2(a) shows the encoding and decoding steps of H-PtrNet for dependency parsing. We use the same encoder as Ma et al. (2018) (red color).[2] Given a sentence, a convolutional neural network (CNN) is used to encode character-level representation of each word, which is then concatenated with word embedding and POS embedding vectors to generate the input sequence $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$. Then a three-layer bi-directional LSTM encodes $\boldsymbol{X}$ into a sequence of hidden states $\boldsymbol{H} = (\boldsymbol{h}_1, \ldots, \boldsymbol{h}_n)$. The decoder (blue color) is a single layer uni-directional LSTM, and also maintains a stack to track of the decoding status. At each decoding step $t$, the decoder receives the *encoder state* of the parent from the stack. In addition, it gets *decoder states* from three different sources: previous decoding step $\boldsymbol{d}_{t-1}$, parent $\boldsymbol{d}_{p(t)}$ and immediate sibling $\boldsymbol{d}_{s(t)}$.

Instead of simply feeding these three components to the decoder, we incorporate a gating mechanism to generalize the ability of our model to extract the most useful information. Eventually, the **fusion function** $f(\boldsymbol{d}_{p(t)}, \boldsymbol{d}_{s(t)}, \boldsymbol{d}_{t-1}, \boldsymbol{h}_{p(t)})$ in Eq. 2 is defined with a *gating mechanism*. We experimented with two different gating functions:

$$\boldsymbol{g}_t = \mathrm{sigmoid}(\boldsymbol{W}_{gd}\boldsymbol{d}_{t-1} + \boldsymbol{W}_{gp}\boldsymbol{d}_{p(t)} + \boldsymbol{W}_{gs}\boldsymbol{d}_{s(t)} + \boldsymbol{b}_g) \quad (5)$$

$$\boldsymbol{g}_t = \mathrm{sigmoid}(\boldsymbol{W}_{gp}(\boldsymbol{d}_{t-1} \odot \boldsymbol{d}_{p(t)}) + \boldsymbol{W}_{gs}(\boldsymbol{d}_{t-1} \odot \boldsymbol{d}_{s(t)}) + \boldsymbol{b}_g) \quad (6)$$

where $\boldsymbol{W}_{gp}$, $\boldsymbol{W}_{gs}$, $\boldsymbol{W}_{gd}$ and $\boldsymbol{b}_g$ are the gating weights. The fusion function is then defined as

$$\boldsymbol{h}'_t = \tanh(\boldsymbol{W}_d\boldsymbol{d}_{t-1} + \boldsymbol{W}_p\boldsymbol{d}_{p(t)} + \boldsymbol{W}_s\boldsymbol{d}_{s(t)}) \quad (7)$$

$$\boldsymbol{h}''_t = \boldsymbol{g}_t \odot \boldsymbol{h}'_t \quad (8)$$

$$\boldsymbol{d}_t = \mathrm{LSTM}(\boldsymbol{h}''_t, \boldsymbol{h}_{p(t)}) \quad (9)$$

[2] https://github.com/XuezheMax/NeuroNLP2

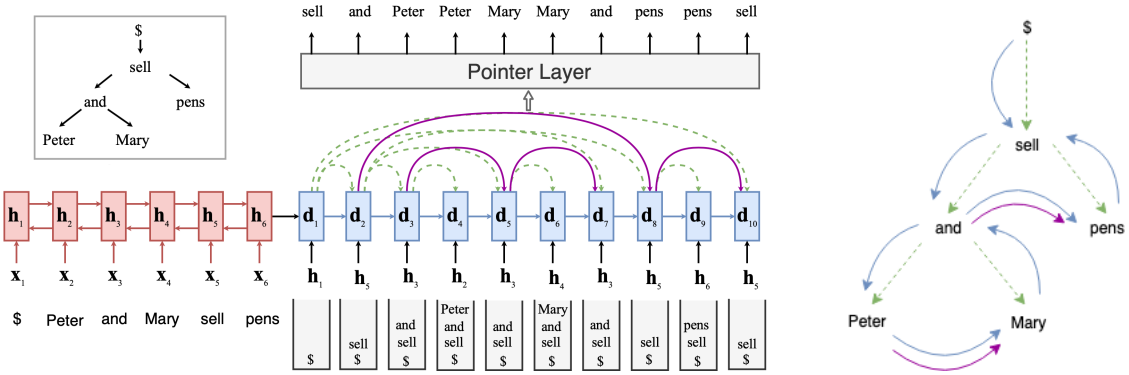


(a) Model Structure (b) Decoder Connections

Figure 2: (a) H-PtrNet for dependency parsing. To reduce visual clutter, we do not show the attention scores over all the input elements at each pointing step, rather show only the pointed element. Figure (b) shows the decoder connections in our H-PtrNet model. The StackPointer network of Ma et al. (2018) has a sequential decoder (shown as blue straight lines). The Green dash lines indicate parent connections and purple solid lines denote the immediate sibling connections in our model.

where $\boldsymbol{W}_d$, $\boldsymbol{W}_p$, $\boldsymbol{W}_s$ are the weights to get the intermediate hidden state $\boldsymbol{h}'_t$, and $\boldsymbol{g}_t$ is a gate to control the information flow from the three decoder states LSTM is the LSTM layer that accepts $\boldsymbol{h}''_t$ as the the hidden state and $\boldsymbol{h}_{p(t)}$ as its input. The LSTM decoder state $\boldsymbol{d}_t$ is then used to compute the attention distribution over the encoder states in pointer layer.

**Pointer and Classifier.** Same as Ma et al. (2018), the pointer and the label classifier are implemented as **bi-affine** layers. Formally, the scoring function $\sigma(\boldsymbol{d}_t, \boldsymbol{h}_i)$ in Eq. 3 is defined as:

$$s_{t,i} = g_1(\boldsymbol{d}_t)^T \boldsymbol{W} g_2(\boldsymbol{h}_i) + \boldsymbol{U} g_1(\boldsymbol{d}_t) + \boldsymbol{V} g_2(\boldsymbol{h}_i) + \boldsymbol{b} \quad (10)$$

where $\boldsymbol{W}$, $\boldsymbol{U}$ and $\boldsymbol{V}$ are the weights, and $g_1(.)$ and $g_2(.)$ are two single layer MLPs with ELU activations. The dependency label classifier has the same structure as the pointer. More specifically,

$$p(\mathrm{y}_l|\boldsymbol{X}) = \mathrm{softmax}(g'_1(\boldsymbol{d}_t)^T \boldsymbol{W}_c g'_2(\boldsymbol{h}_k) + \boldsymbol{U}_c g'_1(\boldsymbol{d}_t) + \boldsymbol{V}_c g'_2(\boldsymbol{h}_k) + \boldsymbol{b}_c) \quad (11)$$

where $\boldsymbol{h}_k$ is the encoder state of the dependent word, $\boldsymbol{d}_t$ is the decoder state of the head word, $\boldsymbol{W}_c$, $\boldsymbol{U}_c$ and $\boldsymbol{V}_c$ are the weights, and $g'_1(.)$ and $g'_2(.)$ are two single layer MLPs with ELU activations.

**Partial Tree Information.** Similar to Ma et al. (2018), we provide the decoder at each step with higher order information about the parent and the sibling of the current node.

### 3.5 Model Specifics for Discourse Parsing.

For discourse parsing, our model uses the same structure as Lin et al. (2019).[3] The encoder is a 5-layer bidirectional RNN based on Gated Recurrent Units (BiGRU) (Cho et al., 2014). As shown in Figure 3, after obtaining a sequence of encoder hidden states representing the words, the last hidden states of the EDUs (*e.g.*, $\boldsymbol{h}_2$, $\boldsymbol{h}_3$, $\boldsymbol{h}_5$, $\boldsymbol{h}_6$, $\boldsymbol{h}_8$ and $\boldsymbol{h}_{10}$) are taken as the EDU representations, generating a sequence of EDU representations $\boldsymbol{E} = (\boldsymbol{e}_1, \ldots, \boldsymbol{e}_m)$ for the input sentence.

Our hierarchical decoder is based on a 5-layer unidirectional GRU. The decoder maintains a **stack** to keep track of the spans that need to be parsed further. At time step $t$, the decoder takes the text span (*e.g.*, $\boldsymbol{e}_{i:j}$) representation from the top of the stack and receives the corresponding parent decoder state $\boldsymbol{d}_{p(t)}$, sibling decoder state $\boldsymbol{d}_{s(t)}$ and previous decoder state $\boldsymbol{d}_{t-1}$ as the input to generate a current decoder state $\boldsymbol{d}_t$. For discourse parsing, we apply Eq. 7 and 9 (with GRU) to implement the fusion function $f(.)$ and to get the decoder state $\boldsymbol{d}_t$.[4] The decoder state is then used in the pointer layer to compute the attention score over the current text span (*e.g.*, $\boldsymbol{e}_{i:j}$) in order to find the position $k$ to generate a new split $(\boldsymbol{e}_{i:k}, \boldsymbol{e}_{k+1:j})$. The parser then applies a relation classifier $\Phi(\boldsymbol{e}_{i:k}, \boldsymbol{e}_{k+1:j})$ to predict the relation and the nuclearity labels for the the new split.

[3] https://ntunlpsg.github.io/project/parser/pointer-net-parser

[4] Adding gating mechanism did not give any gain, rather increased the number of parameters.

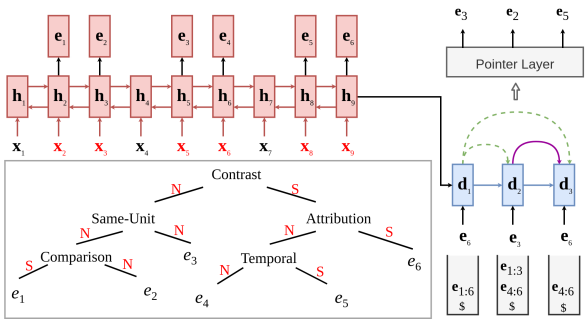


Figure 3: H-PtrNet for discourse parsing. The input symbols in red ($\mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_8, \mathbf{x}_9$) represent last words of the respective EDUs. To avoid visual clutter, we do not show the attention distributions over the EDUs, rather only show the decisions. Green dash lines indicate parent connections and purple solid lines denote the immediate sibling connections.

For pointing, the parser uses a simple **dot product** attention. For labeling, it uses a **bi-affine classifier** similar to the one in Eq. 11. It takes the representations of two spans (*i.e.,* $\boldsymbol{e}_k$ for $\boldsymbol{e}_{i:k}$, $\boldsymbol{e}_j$ for $\boldsymbol{e}_{k+1:j}$) as input and predicts the corresponding relation between them. Whenever the length of any of the newly created span ($\boldsymbol{e}_{i:k}$ and $\boldsymbol{e}_{k+1:j}$) is larger than two, the parser pushes it onto the stack for further processing. Similar to dependency parsing, the decoder is also provided with **partial tree information** – the representations of the **parent** and the immediate **sibling**.

### 3.6 Objective Function

Same as Ma et al. (2018) and Lin et al. (2019), our parsers are trained to minimize the total log loss (cross entropy) for building the right tree **structure** for a given sentence $\boldsymbol{X}$. The structure loss $\mathcal{L}$ is the pointing loss for the pointer network:

$$\mathcal{L}(\theta) = -\sum_{t=1}^{T} \log P_\theta(\mathrm{y}_t | \mathrm{y}_{<t}, \boldsymbol{X}) \qquad (12)$$

where $\theta$ denotes the model parameters, $y_{<t}$ represents the subtrees that have been generated by our parser at previous steps, and $T$ is the number needed for parsing the whole sentence (*i.e.,* number of words in dependency parsing and spans containing more than two EDUs in discourse parsing). The label classifiers are trained simultaneously, so the final loss function is the sum of structure loss (Eq. 12) and the loss for label classifier.

## 4 Experiments

In this section, we describe the experimental details about dependency parsing and discourse parsing, as well as the analysis on both tasks.

Apart from the two gating-based fusion functions described in Section 3 (Eq. 5-6), we experimented with three different versions of our model depending on which connections are considered in the decoder. We append suffixes **P** for *parent*, **S** for *sibling* and **T** for *temporal* to the model name (H-PtrNet) to denote different versions.

- **H-PtrNet-P**: The H-PtrNet model with fusion function $\boldsymbol{d}_t = f(\boldsymbol{d}_{p(t)}, \boldsymbol{h}_{p(t)})$, where the decoder receives hidden (decoder) state only from the parent ($\boldsymbol{d}_{p(t)}$) in each decoding step. Note that $\boldsymbol{h}_{p(t)}$ is the encoder state of the parent.

- **H-PtrNet-PS**: The H-PtrNet model with fusion function $\boldsymbol{d}_t = f(\boldsymbol{d}_{p(t)}, \boldsymbol{d}_{s(t)}, \boldsymbol{h}_{p(t)})$, where the decoder receives the hidden states from both the parent and sibling in each decoding step.

- **H-PtrNet-PST**: This is the full model with fusion function $\boldsymbol{d}_t = f(\boldsymbol{d}_{p(t)}, \boldsymbol{d}_{s(t)}, \boldsymbol{d}_{t-1}, \boldsymbol{h}_{p(t)})$ (Eq. 2). In this model, the decoder receives the hidden states from its parent, sibling and previous step in each decoding step.

### 4.1 Dependency Parsing

**Dataset.** We evaluate our model on the English Penn Treebank (PTB v3.0) (Marcus et al., 1994), which is converted to Stanford Dependencies format with Stanford Dependency Converter 3.3.0 (Schuster and Manning, 2016). To make a thorough empirical comparison with previous studies, we also evaluate our system on seven (7) languages from the Universal Dependency (UD) Treebanks[5] (version 2.3).

**Metrics.** We evaluate the performance of our models with unlabeled attachment score (UAS) and labeled attachment score (LAS). We ignore punctuations in the evaluation for English.

**Experimental Settings.** We use the same setup as Ma et al. (2018) in the experiments for English Penn Treebank and UD Treebanks. For a fair comparison, we rerun their model with the hyperparameters provided by the authors on the same machine as our experiments. For all the languages,

[5] http://universaldependencies.org/

| | StackPtr (code) | | H-PtrNet-PST (Gate) | | H-PtrNet-PST (SGate) | |
|---|---|---|---|---|---|---|
| | UAS | LAS | UAS | LAS | UAS | LAS |
| **bg** | 94.17±0.11 | 90.63±0.06 | 94.20±0.16 | 90.70±0.14 | **94.50±0.16** | **91.01±0.20** |
| **ca** | **93.82±0.06** | **91.99±0.07** | 93.78±0.03 | 91.92±0.03 | 93.67±0.06 | 91.82±0.07 |
| **en** | 90.97±0.07 | 89.06±0.08 | **91.03±0.19** | **89.07±0.16** | 90.94±0.12 | 89.01±0.12 |
| **de** | 87.97±0.20 | 83.75±0.21 | **88.14±0.22** | **83.89±0.26** | 88.06±0.17 | 83.83±0.13 |
| **fr** | 91.57±0.23 | 88.76±0.21 | 91.63±0.15 | 88.70±0.14 | **91.69±0.07** | **88.80±0.11** |
| **it** | 93.76±0.13 | 92.00±0.08 | 93.73±0.08 | 91.90±0.10 | **93.88±0.05** | **92.09±0.02** |
| **ro** | 91.15±0.12 | 85.54±0.13 | **91.34±0.18** | **85.73±0.22** | 91.09±0.09 | 85.36±0.10 |

Table 1: Dependency parsing results on 7 UD Treebanks. **StackPtr (code)** denotes the experiments we rerun on our machine. **H-PtrNet-PST (Gate)** and **H-PtrNet-PST (SGate)** are H-PtrNet models with gating mechanism.

we follow the standard split for training, validation and testing. It should be noted that Ma et al. (2018) used UD Treebanks 2.1, which is not the most up-to-date version. Therefore, during experiments, we rerun their codes with UD Treebanks 2.3 to match our experiments. To be specific, we use structured-skipgram (Ling et al., 2015) for English and German, while Polyglot embedding (Al-Rfou et al., 2013) for the other languages. Adam optimizer (Kingma and Ba, 2015) is used as the optimization algorithm. We apply 0.33 dropout rate between layers of encoder and to word embeddings as well as Eq. 8. We use beam size of 10 for English Penn Treebank, and beam size of 1 for UD Treebanks. The gold-standard POS tags is used for English Penn Treebank. We also use the universal POS tags (Petrov et al., 2011) provided in the dataset for UD Treebanks. See Appendix for a complete list of hyperparameters.

**Results on UD Treebanks.** We evaluate on 7 different languages from the UD Treebanks: 4 major ones: English (en), German (de), French (fr), and Italian (it), and 3 relatively minor ones: Bulgarian (bg), Catalan (ca), and Romanian (ro). Table 1 shows the results. We refer to the results of our run of the code released by Ma et al. (2018) as StackPtr (code).[6] StackPtr (code) and our models are trained in identical settings making them comparable. H-PtrNet-PST (Gate) (Eq. 5) and H-PtrNet-PST (SGate) (Eq. 6) are H-PtrNet models with gating mechanism. Element wise product in Eq. 6 has the effect of similarity comparison, so we denote it as **SGATE**. With gating mechanism, our model shows consistent improvements against the baseline on bg, en, de, fr, it and ro. We also tested H-PtrNet-PS on these 7 languages, but the performances are worse than StackPtr.

[6] We do not directly report the results from their paper because we use a different version of the UD Treebanks.

| Approach | UAS | LAS |
|---|---|---|
| **Baselines** | | |
| StackPtr (paper) | 96.12±0.03 | 95.06±0.05 |
| StackPtr (code) | 95.94±0.03 | 94.91±0.05 |
| **Proposed Model** | | |
| H-PtrNet-PST (Gate) | 96.03±0.02 | 94.99±0.02 |
| H-PtrNet-PST (SGate) | 96.04±0.05 | 95.00±0.06 |
| H-PtrNet-PS (Gate) | **96.09±0.05** | **95.03±0.03** |

Table 2: Dependency parsing results on English Penn Treebank v3.0.

**Results on English Penn Treebank.** Table 2 presents the results on English Penn Treebank. StackPtr (paper) refer to the results reported by Ma et al. (2018), and StackPtr (code) is our run of their code in identical settings as ours. Our model H-PtrNet-PST (Gate) outperforms the baseline by 0.09 and 0.08 in terms of UAS and LAS, respectively. Performance of H-PtrNet-PST (SGate) is close to that of H-PtrNet-PST (Gate), though we see slight improvement. We also test H-PtrNet-PS (Gate), the model with parent and sibling connections only, which further improves the performance to 96.09 and 95.03 in UAS and LAS.

**Performance Analysis.** To make a thorough analysis of our model, we breakdown UAS in terms of sentence lengths to compare the performance of our model and StackPtr. We first take the performance on UD German as an example,

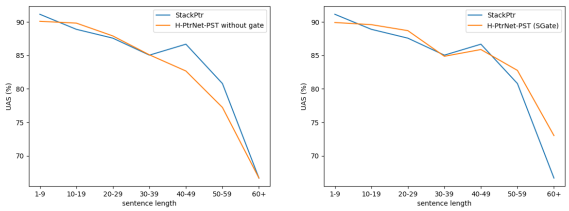


(a) H-PtrNet-PST (no gating) (b) H-PtrNet-PST (SGate)

Figure 4: UD German parsing performance in terms of sentence length.

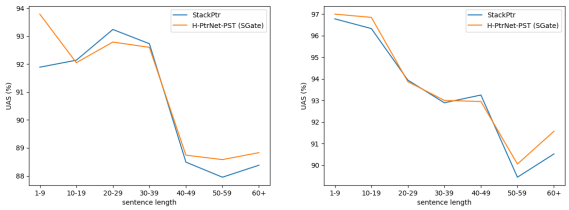


(a) UD French (b) UD Italian

Figure 5: Performance analysis on French and Italian based on H-PtrNet-PST (SGate).

which is shown in Figure 4. The blue line shows the performance of StackPtr, and the orange line shows the performance of our model. From Figure 4(a) we can see that our model without gate performs better on relatively short sentences (10 to 29 words), however, the accuracy drops on longer sentences. The reason could be that adding parent and sibling hidden states to decoder may amplify error accumulation from early parsing mistakes.

Figure 4(b) shows the performance of our model with SGATE (Eq. 6), where we can see that the performance on long sentences has been improved significantly. In the meanwhile, it still maintains higher accuracy than StackPtr on the short sentences (10 to 29 words). Figure 5 shows two more examples, again, from which we can see that our model with SGATE tends to outperform StackPtr on longer sentences.

### 4.2 Discourse Parsing

**Dataset.** We use the standard RST Discourse Treebank (RST-DT) (Carlson et al., 2002), which contains discourse annotations for 385 news articles from Penn Treebank (Marcus et al., 1994). We evaluate our model in sentence-level parsing, for which we extract all the well-formed sentence-level discourse trees from document-level trees.

| Approach | Span | Nuclearity | Relation |
|---|---|---|---|
| **Human Agreement** | 95.7 | 90.4 | 83.0 |
| **Baselines** | | | |
| Joty et al. (2012) | 94.6 | 86.9 | 77.1 |
| Ji and Eisenstein (2014) | 93.5 | 81.3 | 70.5 |
| Wang et al. (2017) | 95.6 | 87.8 | 77.6 |
| Pointer Net† (Lin et al., 2019) | $97.39_{\pm 0.1}$ | $91.01_{\pm 0.4}$ | $81.08_{\pm 0.4}$ |
| Pointer Net§ (Lin et al., 2019) | $97.14_{\pm 0.1}$ | $91.00_{\pm 0.3}$ | $81.29_{\pm 0.2}$ |
| **Proposed Model** | | | |
| H-PtrNet-P† | $\mathbf{97.68}_{\pm 0.03}$ | $91.86_{\pm 0.2}$ | $81.82_{\pm 0.2}$ |
| H-PtrNet-P§ | $97.51_{\pm 0.08}$ | $91.98_{\pm 0.1}$ | $82.11_{\pm 0.2}$ |
| H-PtrNet-PS† | $97.56_{\pm 0.1}$ | $91.52_{\pm 0.3}$ | $82.05_{\pm 0.5}$ |
| H-PtrNet-PS§ | $97.35_{\pm 0.1}$ | $91.78_{\pm 0.1}$ | $82.35_{\pm 0.2}$ |
| H-PtrNet-PST† | $97.56_{\pm 0.06}$ | $91.97_{\pm 0.3}$ | $82.37_{\pm 0.4}$ |
| H-PtrNet-PST§ | $97.48_{\pm 0.2}$ | $\mathbf{92.01}_{\pm 0.2}$ | $\mathbf{82.77}_{\pm 0.2}$ |

Table 3: Discourse parsing results with gold segmentation. † denotes the models selected based on Span. § denotes the models selected based on Relation. We run all the experiments for three times and report the averages and the standard deviations.

In all, the training data contains 7321 sentences, and the testing data contains 951 sentences. These numbers match the statistics reported by Lin et al. (2019). We follow the same settings as in their experiments and randomly choose 10% of the training data for hyperparameter tuning.

**Metric and Relation Labels.** Following the standard in RST parsing, we use the unlabeled (Span) and labeled (Nuclearity, Relation) metrics proposed by Marcu (2000). We only present $F_1$-score for space limitations. Following the previous work, we attach the nuclearity labels (NS, SN, NN) to 18 discourse relations, together giving 39 distinctive relation labels.

**Experimental Settings.** Since our goal is to evaluate our parsing method, we conduct the experiments based on gold EDU segmentations. We compare our results with the recently proposed pointer network based parser of Lin et al. (2019) (Pointer Net). However, unlike their paper, we report results for both cases: $(i)$ when the model was selected based on the best performance on Span identification; and $(ii)$ when it was selected based on the relation labeling performance on the development set. We retrain their model for both settings. We also apply Adam optimizer as optimization algorithm and ELMo (Peters et al., 2018) with 0.5 dropout rate as word embeddings.

**Results.** We present the results in Table 3. In discourse parsing, the number of EDUs in a sentence is relatively small compared to the sentence lengths (in words) in dependency parsing. Based on the observation in dependency parsing that the performance of H-PtrNet may drop for longer sentences due to parent error accumulation, we expect that in discourse parsing, this should not be the case since the the number of parsing steps is much smaller compared to that of dependency parsing.

We first consider the models that were selected based on Span performance (models with † superscript). H-PtrNet-P, with only parent connection, outperforms the baseline in all three tasks. It achieves an absolute improvement of 0.29 $F_1$ in span identification compared to the baseline. Considering the performance has already exceeded the human agreement of 95.7 $F_1$, this gain is remarkable. Thanks to the higher accuracy on finding the right spans, we also achieve 0.85 and 0.74 absolute improvements in Nuclearity and Relation tasks, respectively. By adding the sibing and temporal connections, we test the performance of our full model, H-PtrNet-PST. The performance on Span is 0.17 $F_1$ higher than the baseline. However, it is not on par with our H-PtrNet-P. But, it is not surprising since we adopt binary tree structures in discourse parsing, which means the sibling information could be redundant in most cases. This also accords with our previous assumption that parent connections may bring enough information to decode RST trees.

Now we consider the models that were selected based on Relation labeling performance (models with § superscript). We achieve significant improvement in Relation compared to the baseline. Eventually the parser yields an $F_1$ of 82.77, which is very close to the human agreement (83.0 $F_1$). We observe that the performance in H-PtrNet-PS and H-PtrNet-PST is better than the H-PtrNet-P. As the relation classifier and the pointer network share the same encoder information (Sec. 3), we believe that richer decoder information leads the model to learn better representations of the text spans (encoder states) and further leads to a better performance in relation labeling.

We further analyze the performance of our proposed model in terms of number of EDUs. We present the $F_1$ score in Span of H-PtrNet-P and H-PtrNet-PST as well as the baseline in Figure 6. It can be observed that both H-PtrNet-P and H-PtrNet-PST outperform the baseline with respect to almost every number of EDUs. Moreover, we can see that the H-PtrNet-P performs better in most of the cases, which once again conforms to our assumption that parent information is enough to decode RST trees. However, as discussed in our dependency parsing experiments, when the number of words (EDUs for discourse parsing) increases, the model may suffer from error accumulation from early parsing. Hence, H-PtrNet-PST tends to perform better when EDU number becomes large.

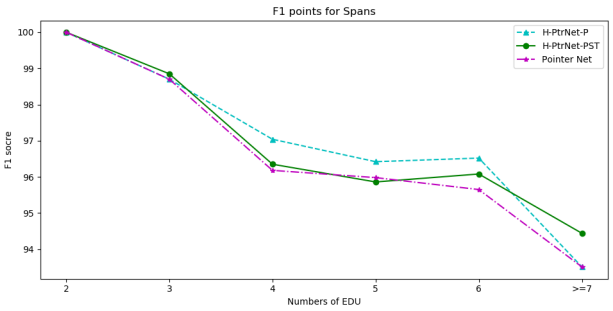


Figure 6: $F_1$ score in Span for different EDU numbers.

## 5 Conclusions

In this paper, we propose hierarchical pointer network parsers and apply them to dependency and discourse parsing tasks. Our parsers address the limitation of previous methods, where the decoder has a sequential structure while it is decoding a hierarchical tree structure, by allowing more flexible information flow to help the decoder receive the most relevant information. For both tasks, our parsers outperform existing methods and set new state-of-the-arts of the two tasks. The broken-down analysis clearly illustrates that our parsers perform better for long sequences, complying with the motivation of our model.

## Acknowledgements

This research is partly supported by the Alibaba-NTU Singapore Joint Research Institute, Nanyang Technological University (NTU), Singapore. Shafiq Joty would like to thank the funding support from his Start-up Grant (M4082038.020).

## References

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word represen-

tations for multilingual nlp. *arXiv preprint arXiv:1307.1662*.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2442–2452, Berlin, Germany. Association for Computational Linguistics.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with LSTMs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 349–359, Lisbon, Portugal. Association for Computational Linguistics.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2002. RST Discourse Treebank (RST–DT) LDC2002T07. *Linguistic Data Consortium, Philadelphia*.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *ICLR*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343, Beijing, China. Association for Computational Linguistics.

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 1*, COLING '96, pages 340–345, Copenhagen, Denmark. ACL.

Vanessa Wei Feng and Graeme Hirst. 2014. A linear-time bottom-up discourse parser with constraints and post-editing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 511–521. Association for Computational Linguistics.

Yangfeng Ji and Jacob Eisenstein. 2014. Representation learning for text-level discourse parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13–24, Baltimore, Maryland. ACL.

Shafiq Joty, Giuseppe Carenini, and Raymond Ng. 2012. A novel discriminative framework for sentence-level discourse analysis. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 904–915. Association for Computational Linguistics.

Shafiq Joty, Giuseppe Carenini, and Raymond T Ng. 2015. Codra: A novel discriminative framework for rhetorical analysis. *Computational Linguistics*, 41:3:385–435.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Sujian Li, Liang Wang, Ziqiang Cao, and Wenjie Li. 2014. Text-level discourse dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 25–35. ACL.

Xiang Lin, Shafiq Joty, Prathyusha Jwalapuram, and M Saiful Bari. 2019. A Unified Linear-Time Framework for Sentence-Level Discourse Parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, ACL '19, pages xx–xx, Florence, Italy. ACL.

Wang Ling, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1299–1304.

Xuezhe Ma, Zecong Hu, Jingzhou Liu, Nanyun Peng, Graham Neubig, and Eduard Hovy. 2018. Stack-pointer networks for dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1403–1414. Association for Computational Linguistics.

William Mann and Sandra Thompson. 1988. Rhetorical Structure Theory: Toward a Functional Theory of Text Organization. *Text*, 8(3):243–281.

Daniel Marcu. 1999. The automatic construction of large-scale corpora for summarization research. In *Proceedings of SIGIR*, pages 137–144.

Daniel Marcu. 2000. The Rhetorical Parsing of Unrestricted Texts: A Surface-based Approach. *Computational Linguistics*, 26:395–448.

Mitchell Marcus, Mary Marcinkiewicz, and Beatrice Santorini. 1994. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Mathieu Morey, Philippe Muller, and Nicholas Asher. 2018. A dependency perspective on RST discourse parsing and evaluation. *American Journal of Computational Linguistics*, 44(2):197–235.

Philippe Muller, Stergos Afantenos, Pascal Denis, and Nicholas Asher. 2012. Constrained decoding for text-level discourse parsing. In *Proceedings of COLING 2012*, pages 1883–1900, Mumbai, India. The COLING 2012 Organizing Committee.

Dat Quoc Nguyen and Karin Verspoor. 2018. An improved neural network model for joint POS tagging and dependency parsing. *CoRR*, abs/1807.03955.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Slav Petrov, Dipanjan Das, and Ryan T. McDonald. 2011. A universal part-of-speech tagset. *CoRR*, abs/1104.2086.

Sebastian Schuster and Christopher D. Manning. 2016. Enhanced english universal dependencies: An improved representation for natural language understanding tasks. In *LREC*.

Radu Soricut and Daniel Marcu. 2003. Sentence Level Discourse Parsing Using Syntactic and Lexical Information. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL'03, pages 149–156, Edmonton, Canada. ACL.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2692–2700. Curran Associates, Inc.

Yizhong Wang, Sujian Li, and Houfeng Wang. 2017. A two-stage parsing method for text-level discourse analysis. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 184–188. Association for Computational Linguistics.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 188–193, Portland, Oregon, USA. Association for Computational Linguistics.

# Semi-Supervised Semantic Role Labeling with Cross-View Training

**Rui Cai** and **Mirella Lapata**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
Rui.Cai@ed.ac.uk mlap@inf.ed.ac.uk

## Abstract

The successful application of neural networks to a variety of NLP tasks has provided strong impetus to develop end-to-end models for semantic role labeling which forego the need for extensive feature engineering. Recent approaches rely on high-quality annotations which are costly to obtain, and mostly unavailable in low resource scenarios (e.g., rare languages or domains). Our work aims to reduce the annotation effort involved via semi-supervised learning. We propose an end-to-end SRL model and demonstrate it can effectively leverage unlabeled data under the cross-view training modeling paradigm. Our LSTM-based semantic role labeler is jointly trained with a sentence learner, which performs POS tagging, dependency parsing, and predicate identification which we argue are critical to learning directly from unlabeled data without recourse to external pre-processing tools. Experimental results on the CoNLL-2009 benchmark dataset show that our model outperforms the state of the art in English, and consistently improves performance in other languages, including Chinese, German, and Spanish.

## 1 Introduction

Semantic role labeling — the task of automatically identifying and labeling the semantic roles conveyed by sentential constituents — has enjoyed renewed interest in the last few years thanks to the popularity of neural network models and their ability to learn continuous representations which forego the need for extensive feature engineering. Recent modeling developments aside, semantic role labeling (SRL) has been generally recognized as a core task in NLP with relevance for applications ranging from machine translation (Aziz et al., 2011; Marcheggiani et al., 2018), to information extraction (Christensen et al., 2011), and summarization (Khan et al., 2015).

State-of-the art semantic role labelers (He et al., 2018b; Cai et al., 2018) rely on high-quality annotations (of semantic predicates and their arguments) for use in training. These annotations are costly to obtain, and mostly unavailable in low resource scenarios (e.g., rare languages or domains) motivating the need for effective semi-supervised methods that leverage unlabeled examples. Cross-View Training (CVT; Clark et al. 2018) is a recently proposed semi-supervised learning algorithm that improves representation learning using a mix of labeled and unlabeled data. The main idea behind CVT is to train a model to produce consistent predictions across different restricted views of the input with the aid of auxiliary prediction tasks. Clark et al. (2018) demonstrate performance gains when applying CVT to sequence tagging tasks, machine translation, and dependency parsing.

Unfortunately, application of CVT to semantic role labeling is fraught with difficulty. This is partly due to the nature of the task which relies on various syntactic features, even when conceptualized as a sequence labeling task (Marcheggiani et al., 2017; He et al., 2018b; Cai et al., 2018). The reliance on syntactic features is problematic for semi-supervised training, since these would have to be extracted from large amounts of unlabeled data. In addition, any semantic role labeler would need to identify (and disambiguate) predicates prior to labeling their augments which might be given during training (e.g., as in the CoNLL 2009 dataset), but would still have to be detected on unlabeled data. Resorting to various pre-processing tools for semi-supervised training almost defeats the purpose of using unlabeled data which would have to be annotated, albeit automatically, with pre-trained models which require labeled data on their own, and might not be portable across languages and domains. Moreover, the usage of external tools often leads to pipeline-style



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1018–1027, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

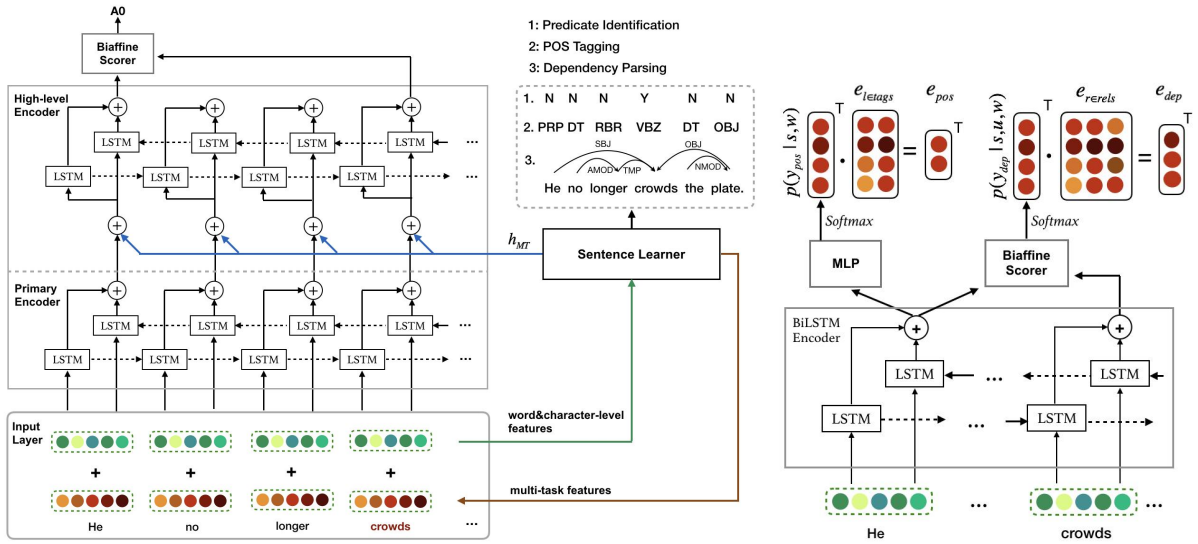


(a) Semantic role labeler (left) and sequence learner (right)

(b) Sentence learner.

Figure 1: Overview of end-to-end SRL model.

architectures where errors propagate to later processing stages, affecting model performance.

In this paper we aim to render semi-supervised learning for semantic role labeling as simple as possible, by eliminating the reliance on multiple external pre-processing tools. We develop a sentence learner which is able to perform all tasks subsidiary to semantic role labeling (i.e., POS tagging, dependency parsing, and predicate detection) on labeled and unlabeled data. The sentence learner is jointly trained with a semantic role labeler, and its outputs are fed to the semantic role labeler during supervised and semi-supervised learning. Aside from building a self-sufficient semantic role labeler which can be directly applied on plain text, an added benefit of the proposed approach is that the sentence learner naturally provides multiple hidden layers (for the various subtasks) from which "multi-task hidden features" (Peters et al., 2018) can be extracted.

In addition to overcoming the difficulty of utilizing plain text for semi-supervised SRL, we show that application of CVT to SRL requires special attention over and above the sequence tagging and dependency parsing tasks discussed in Clark et al. (2018). We investigate how to best formulate different views for CVT focusing on semantic predicates which have been proven to be very useful in recent SRL models (Marcheggiani et al., 2017; Marcheggiani and Titov, 2017; Cai et al., 2018). Experimental results on the CONLL-2009 benchmark datasets show that our model is able to outperform the state of the art in English, and to improve SRL performance in other languages, including Chinese, German, and Spanish.

## 2 Model Description

Figure 1 provides a schematic overview of our model which has two main components, namely a sentence learner and a semantic role labeler. The sentence learner consists of:

- look-ups of word embeddings and character embeddings;
- a bidirectional LSTM (BiLSTM) encoder which takes as input the representation of each word in a sentence and produces context-dependent representations;
- a multi-task prediction module to perform POS tagging, dependency parsing, and predicate identification.

While the semantic role labeler consists of:

- an input layer which combines multi-source representations of the input sentence;
- a primary bidirectional LSTM (BiLSTM) encoder which takes as input the representation of each word in a sentence and produces context-dependent embeddings;
- a high-level BiLSTM encoder which takes as input the hidden states of the primary BiLSTM and multi-task hidden features;
- a biaffine classifier for calculating the score of each semantic role for each word.

### 2.1 Sentence Learner

The sentence learner (see Figure 1b) operates over sentences to perform the intermediate tasks of POS tagging, dependency parsing, and predicate identification, which are subsequently used to inform the decisions of the semantic role labeler.

**Sentence Encoder** As we expect the sentence encoder to be applied directly to plain text, we represent words as the concatenation of character- and word-level features. We learn character-level representations $x_{cr}$ by feeding character embeddings to a convolutional neural network module. We represent words with randomly initialized word embeddings $x_{re} \in R^{d_w}$, pre-trained word embeddings $x_{pe} \in R^{d_w}$ estimated on an external text collection, and pre-trained ELMo embeddings $x_{elmo}$ (Peters et al., 2018). The final word representation is given by $x = x_{re} \circ x_{pe} \circ x_{cr} \circ x_{elmo}$, where $\circ$ represents concatenation.

Following Marcheggiani et al. (2017), sentences are represented using a two-layer bi-directional LSTM (Hochreiter and Schmidhuber, 1997); the BiLSTM receives at time step $t$ a representation $x$ for each word and recursively computes two hidden states, one for the forward pass ($\overrightarrow{h}^t$), and another one for the backward pass ($\overleftarrow{h}^t$). Each word is the concatenation of its forward and backward LSTM state vectors $h^t = \overrightarrow{h}^t \circ \overleftarrow{h}^t$.

**Multi-task Learning** After obtaining word representations with the sentence encoder, the input is POS tagged, dependency parsed, and predicates are identified. Given sentence $s$, the probability distribution of POS tags for word $w$ is obtained using a one hidden-layer neural network applied to the corresponding encoder output $h_2^w$:

$$p(y_{pos}|s,w) = \text{softmax}(U \cdot \text{ReLU}(W h_2^w) + b) \quad (1)$$

For dependency parsing, words in a sentence are treated as nodes in a graph. In particular, each word $w$ in sentence $s$ receives exactly one in-going edge $(u, w, r)$ from head word $u$ to its dependent $w$ with relation $r$. We use a graph-based dependency parser similar to the one presented in Clark et al. (2018), which treats dependency parsing as a classification task and its goal is to predict which in-going edge $(u, w, r)$ connects to each word $w$. Mathematically, the probability of an edge is:

$$p((u,w,r)|s) = \propto \exp(\text{score}(h_2^u, h_2^w, r)) \quad (2)$$

where "score" is the scoring function:

$$\begin{aligned} \text{score}(z_1, z_2, r) = &\text{ReLU}(W_{head} z_1 + b_{head}) \\ &(W_r + W)\,\text{ReLU}(W_{dep} z_2 + b_{dep}) \end{aligned} \quad (3)$$

The bilinear classifier uses a weight matrix $W_r$ specific to the candidate relation and a weight matrix $W$ shared across all dependency relations.

With regard to predicate identification, we introduce a virtual root following Cai et al. (2018) who model the entire SRL task as word pair classification. Similarly to the dependency parsing module described above, representations produced by the encoder for the virtual root and words are passed through separate hidden layers and a biaffine classifier is applied to produce a score for each word.

### 2.2 Semantic Role Labeler

**Word Representation** For our SRL model, words are represented by a vector $x$ which is the concatenation of four types of features: predicate-specific, character-level, word-level, and multi-task features. Following previous work (Marcheggiani et al., 2017), we leverage a predicate specific indicator embedding $x_{ie}$ rather than directly using a binary flag. Character- and word-level features are shared with the sentence learner.

With regard to multi-task features, for each word $w$ in input sentence $s$, we employ a probability-weighted POS tag embedding $e_{pos}$ and dependency relation embedding $e_{dep}$:

$$\begin{aligned} e_{pos} &= \sum_{l \in tags} p(y_{pos} = l|s,w)[l] * e_l \\ e_{dep} &= \sum_{r \in rels} p(y_{dep} = r|s,u,w)[r] * e_r \end{aligned} \quad (4)$$

where $e_l$ and $e_r$ and the embeddings of POS tags and dependency relations respectively, and $p(y_{dep} = r|s,u)$ is the probability of relation $r$ given its predicted dependency head $u$.

In order to incorporate more syntactic information, we adopt as an additional feature the probability of linking a word to candidate predicate $x_{pr}$ which we obtain from the sentence learner. $x_{pr}$ is made of two scalar values, $p_{head}$ and $p_{dep}$, which represent the probability of a word being the syntactic head or dependent of the current predicate, respectively.

**Multi-task Hidden Features** Drawing inspiration from ELMo (Peters et al., 2018), a recently

proposed model for generating word representations based on bidirectional LSTMs trained with a coupled language modeling objective, we extract various hidden features via multi-task learning. ELMo representations are deep, essentially a linear combination of representations learnt at all layers of an LSTM instead of just the final layer. Compared with unsupervised ELMo representations, our sentence learner takes advantage of labeled data — it attempts to learn representations towards multiple SRL-related tasks rather than generally effective ones.

In order to utilize all hidden layers in the sentence learner, we collapse them into a single vector. Although we could simply concatenate these or select the top layer, we compute multi-task hidden features $h_{MT}$ as a weighting of the BiLSTM layers, followed by a non-linear projection:

$$h_{MT} = \mathrm{ReLU}(W_{hidden}(\gamma \sum_{j=1}^{j=L} \beta_j h_j)) \quad (5)$$

where $L$ is the depth of sentence learner, $\beta$ are softmax-normalized weights for $h_j$, and the scalar parameter $\gamma$ is of practical importance to aid optimization (Peters et al., 2018).

**Biaffine Role Scorer** After the high-level BiLSTM encoder produces representations $h$ for each word, we perform two distinct non-linear transformations for the currently considered predicate and its candidate arguments, respectively:

$$\begin{aligned} h_{pred} &= \mathrm{ReLU}(W_{pred} h + b_{pred}) \\ h_{arg} &= \mathrm{ReLU}(W_{arg} h + b_{arg}) \end{aligned} \quad (6)$$

where $h_{pred}$ and $h_{arg}$ are hidden representations for the predicate and candidate arguments. The score $s_{role}$ of a semantic role between a predicate and its arguments is calculated as:

$$\begin{aligned} s_{role} = & h_{arg}^{\top} W_{role} h_{pred} \\ & + U_{role}(h_{arg} \circ h_{pred}) \\ & + b_{role} \end{aligned} \quad (7)$$

where $W_{role}$, $U_{role}$, and $b_{role}$ are parameters updated during training.

### 2.3 Cross-view Training for SRL

CVT works by improving representation learning for a model. Let $D_{ul} = \{x_1, x_2, .., x_N\}$ represent an unlabeled dataset. We use $p_\theta(y|x_i)$ to denote the output distribution over classes produced by the model with parameters $\theta$. CVT adds multiple different auxiliary prediction modules to a model, which are used when learning on unlabeled examples. Each prediction module takes as input an intermediate representation $h^j(x_i)$ produced by a primary BiLSTM encoder and outputs a distribution over all possible classes $p_\theta^j(y|x_i)$. Each $h^j$ is chosen such that it can only see parts of the input.

Given an unlabeled example, the model first produces soft targets $p_\theta(y|x_i)$ by performing inference. CVT then trains auxiliary prediction modules to match the teacher prediction module on the unlabeled data by minimizing:

$$\mathcal{L}_{\mathrm{CVT}}(\theta) = \frac{1}{D_{ul}} \sum_{x_i \in D_{ul}} \sum_{j=1}^{k} D(p_\theta(y|x_i), p_\theta^j(y|x_i)) \quad (8)$$

where $D$ is a distance function between probability distributions (we use KL divergence). During training, we keep predictions $p_\theta(y|x_i)$ from the teacher module fixed so that the auxiliary modules learn to imitate the teacher, but not vice versa. As auxiliary modules train, the representations they use as input improve so they are useful for making predictions even when some of the models' inputs are not available. This in turn improves the primary prediction module, which is built on top of the same shared representations.

We applied CVT on the primary Bi-LSTM encoder of our semantic role labeler, while utilizing the output of the sentence learner on the unlabeled data. Given unlabeled sentence $s = w^1, ..., w^T$, the primary Bi-LSTM encoder produces hidden representations $h_{pri}$ for each word, while the semantic role labeler produces the teacher prediction. The sentence learner may recognize more than one words as predicates in sentence $s$, and we just randomly choose one as the target predicate $w^p$.

The auxiliary prediction modules take $\overrightarrow{h}^t_{pri}$ and $\overleftarrow{h}^t_{pri}$ as input. Specifically, we add the following four auxiliary prediction modules to the model:

$$\begin{aligned} p_\theta^{\mathrm{fwd}}(r^t|w^t, w^p, s) &= \mathrm{NN}^{\mathrm{fwd}}(\overrightarrow{h}^t_{pri}(s)) \\ p_\theta^{\mathrm{bwd}}(r^t|w^t, w^p, s) &= \mathrm{NN}^{\mathrm{bwd}}(\overleftarrow{h}^t_{pri}(s)) \\ p_\theta^{\mathrm{future}}(r^t|w^t, w^p, s) &= \mathrm{NN}^{\mathrm{future}}(\overrightarrow{h}^{t-1}_{pri}(s)) \\ p_\theta^{\mathrm{past}}(r^t|w^t, w^p, s) &= \mathrm{NN}^{\mathrm{past}}(\overleftarrow{h}^{t+1}_{pri}(s)) \end{aligned} \quad (9)$$

The "forward" module makes predictions without seeing the right context of the current word.

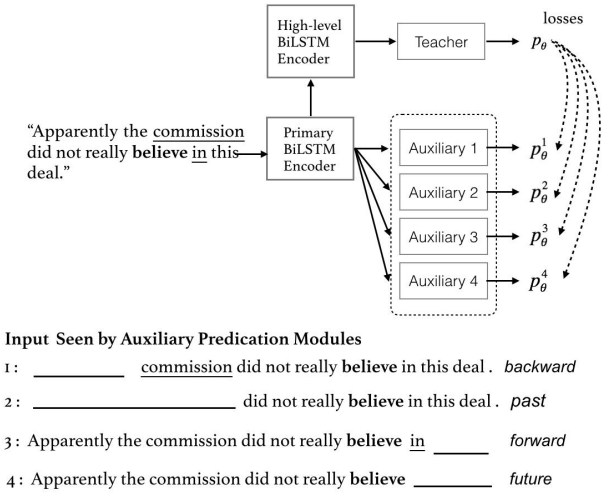


Figure 2: CVT on a sentence from the CoNLL 2009 training set. The current predicate is "*believe*", while "*commission*" and "*in*" are two candidate arguments.

The "future" module makes predictions without seeing the right context *and* the current word itself. The "backward" and "past" modules are defined analogously for left contexts. Figure 2 illustrates the auxiliary modules and the types of context they see. Unlike the biaffine role scorer, the auxiliary modules do not explicitly use the hidden state of the target predicate, so as to encourage the primary Bi-LSTM encoder to capture long-distance relations between the predicate and its context.

We empirically observed (see Section 3) that applying CVT on representations which do not "see" the target predicate is not ideal for SRL. We therefore devised a strategy which applies different auxiliary modules to each word depending on its relative position to the target predicate. We only apply "backward" and "past" modules to words preceding the predicate, while "forward" and "future" modules apply to words following the predicate. This way, we ensure that each word is aware of the current predicate when performing CVT. In the example in Figure 2, "backward" and "past" views would be applied to *commission* and "forward" and "future" views to *in*.

We also apply CVT on the first hidden layer of the sentence learner to further improve the performances of auxiliary tasks, utilizing the views introduced in Clark et al. (2018) for sequence tagging and dependency parsing.

### 2.4 Training Objectives

For both supervised learning and cross-view training our model makes predictions on labeled and unlabeled examples across all tasks (SRL and auxiliary tasks). During supervised learning, the model is trained on labeled data and its objective is the sum of cross-entropy losses for all tasks. With respect to multi-task CVT, the model takes unlabeled examples as input and calculates the CVT loss given in Equation (8). The semi-supervised objective is the sum of the CVT loss for all auxiliary modules across all tasks.

| Hyperparameter | value |
|---|---|
| $d_w$ (English word embeddings) | 100 |
| $d_w$ (other languages word embeddings) | 300 |
| $d_{cr}$ (character-level representations) | 100 |
| $d_{pos}$ (POS embeddings) | 32 |
| $d_{de}$ (dependency label embeddings) | 32 |
| $d_{ie}$ (predicate indicator embeddings) | 16 |
| Bi-LSTM hidden states size | 400 |
| hidden-layer size in biaffine scorers | 300 |
| Multi-task hidden features size | 200 |
| primary BiLSTM depth | 1 |
| high-level BiLSTM depth | 2 |
| batch size | 30 |
| learning rate | 0.001 |

Table 1: Hyperparameter values.

## 3 Experiments

We implemented our model[1] in PyTorch and evaluated it on the English CoNLL-2009 benchmark following the standard training, testing, and development set splits. To evaluate whether our model generalizes to other languages, we also report experiments on Chinese, German, and Spanish, again using standard CoNLL-2009 splits. This subset of languages has been commonly used in previous work (Björkelund et al., 2010; Roth and Lapata, 2016; Lei et al., 2015) and allows us to compare our model against a wide range of alternative approaches. The benchmarks contain gold-standard dependency annotations, and also gold lemmas, part-of-speech tags, and morphological features. With regard to unlabeled datasets, we used the 1 Billion Word Language Model Benchmark (Chelba et al., 2013) for English, the Sougou News Data[2] for Chinese, the NEGRA corpus [3] for

[1] Our code is publicly available at https://github.com/RuiCaiNLP/SemiSRL.

[2] www.sogou.com/labs/resource/ca.php

[3] www.coli.uni-sb.de/sfb378/negra-corpus/

| *Single Models* | P | R | $F_1$ |
|---|---|---|---|
| Björkelund et al. (2010) | 87.1 | 84.5 | 85.8 |
| Lei et al. (2015) | — | — | 86.6 |
| FitzGerald et al. (2015) | — | — | 86.7 |
| Roth and Lapata (2016) | 88.1 | 85.3 | 86.7 |
| Marcheggiani and Titov (2017) | 89.1 | 86.8 | 88.0 |
| Marcheggiani et al. (2017) | 88.7 | 86.8 | 87.7 |
| He et al. (2018b) | 89.7 | 89.3 | 89.5 |
| Cai et al. (2018) | 89.9 | 89.2 | 89.6 |
| Li et al. (2018) | 90.3 | 89.3 | 89.8 |
| **Ours** (supervised training) | 91.1 | 90.4 | **90.7** |
| **Ours** (with CVT) | 91.7 | 90.8 | **91.2** |
| *Ensemble Models* | P | R | F |
| FitzGerald et al. (2015) | — | — | 87.7 |
| Roth and Lapata (2016) | 90.3 | 85.7 | 87.9 |
| Marcheggiani and Titov (2017) | 90.5 | 87.7 | 89.1 |

Table 2: English results on CoNLL-2009 in-domain (WSJ) test set. Differences in $F_1$ between our models and previous systems are statistically significant ($p < 0.05$) using stratified shuffling (Noreen, 1989).

| *Single Models* | P | R | $F_1$ |
|---|---|---|---|
| Björkelund et al. (2010) | 75.7 | 72.2 | 73.9 |
| Lei et al. (2015) | - | - | 75.6 |
| FitzGerald et al. (2015) | - | - | 75.2 |
| Roth and Lapata (2016) | 76.9 | 73.8 | 75.3 |
| Marcheggiani and Titov (2017) | 78.5 | 75.9 | 77.2 |
| Marcheggiani et al. (2017) | 79.4 | 76.2 | 77.7 |
| He et al. (2018b) | 81.9 | 76.9 | 79.3 |
| Cai et al. (2018) | 79.8 | 78.3 | 79.0 |
| Li et al. (2018) | 80.6 | 79.0 | 79.8 |
| **Ours** (supervised training) | 82.1 | 81.3 | **81.6** |
| **Ours** (with CVT) | 83.2 | 81.9 | **82.5** |
| *Ensemble Models* | P | R | F |
| FitzGerald et al. (2015) | - | - | 75.5 |
| Roth and Lapata (2016) | 79.7 | 73.6 | 76.5 |
| Marcheggiani and Titov (2017) | 80.8 | 77.1 | 78.9 |

Table 3: CoNLL-2009 out-of domain results (English; Brown test set). Differences in $F_1$ between our models and previous systems are statistically significant ($p < 0.05$) using stratified shuffling (Noreen, 1989).

German, and the Spanish Language News Corpus[4] for Spanish.

For experiments on English, we used the embeddings of Dyer et al. (2015) which were learned using the structured skip n-gram approach of Ling et al. (2015). We also used a convolutional neural network (Chiu and Nichols, 2016; Ma and Hovy, 2016) to learn character-level representations. For Chinese, Spanish, and German word embeddings were pre-trained on Wikipedia using `fastText` (Bojanowski et al., 2017).

The Bi-LSTM encoders in our model, used recurrent dropout (Gal and Ghahramani, 2016) with an 80% keep probability between time-steps and layers during supervised training; keep probability was set to 90% when applying the model to unlabeled data. We used the Adam optimizer (Kingma and Ba, 2014) and performed hyperparameter tuning and model selection on the English development set; optimal hyperparameter values (for all languages) are shown in Table 1.

### 3.1 Results

Our results on the English (in-domain) test set are summarized in Table 2. We compared our system against previous models which employ external tools to obtain required features. We also report the results of various ensemble SRL models (second block). Most comparisons involve neural systems which are based on BiLSTMs (Marcheggiani et al., 2017; He et al., 2018b; Marcheggiani and Titov, 2017; Cai et al., 2018) or use neural networks for learning SLR-specific embeddings (FitzGerald et al., 2015; Roth and Lapata, 2016). We also report the results of two strong symbolic models based on tensor factorization (Lei et al., 2015) and a pipeline of modules that carry out tokenization, lemmatization, part-of-speech tagging, dependency parsing, and semantic role labeling (Björkelund et al., 2010). As can be seen in Table 2, our supervised model outperforms previously published single and ensemble models. With cross-view training, our model achieves 91.2% $F_1$ (the difference over the supervised model is statistically significant at $p < 0.05$), which is an absolute improvement of 1.4% over the state of the art (Li et al., 2018).

Results on the out-of-domain English test set are presented in Table 3. We include comparisons with the same models as in the in-domain case. Again, our end-to-end model significantly outperforms previously published single and ensemble models, even without taking unlabeled data into account. We achieve a relatively higher improvement with CVT on out-of-domain data ($F_1$ increases from 81.6% to 82.5%, and the difference is significant at $p < 0.05$). This suggests that semi-

[4] `catalog.ldc.upenn.edu/LDC99T41`

| Chinese | P | R | $F_1$ |
|---|---|---|---|
| Björkelund et al. (2010) | 82.4 | 75.1 | 78.6 |
| Roth and Lapata (2016) | 83.2 | 75.9 | 79.4 |
| Marcheggiani and Titov (2017) | 84.6 | 80.4 | 82.5 |
| He et al. (2018b) | 84.2 | 81.5 | 82.8 |
| Cai et al. (2018) | 84.7 | 84.0 | 84.3 |
| Li et al. (2018) | 84.8 | 81.2 | 83.0 |
| **Ours** (supervised training) | 84.9 | 84.3 | 84.6 |
| **Ours** (with CVT) | 85.4 | 84.6 | **85.0** |
| German | P | R | $F_1$ |
| Björkelund et al. (2010) | 81.2 | 78.3 | 79.7 |
| Roth and Lapata (2016) | 81.8 | 78.5 | 80.1 |
| **Ours** (supervised training) | 84.5 | 82.1 | 83.3 |
| **Ours** (with CVT) | 84.9 | 82.7 | **83.8** |
| Spanish | P | R | F |
| Björkelund et al. (2010) | 78.9 | 74.3 | 76.5 |
| Roth and Lapata (2016) | 83.2 | 77.4 | 80.2 |
| Marcheggiani et al. (2017) | 81.4 | 79.3 | 80.3 |
| **Ours** (supervised training) | 83.0 | 81.3 | 82.1 |
| **Ours**(with CVT) | 83.6 | 82.2 | **82.9** |

Table 4: CoNLL-2009 results on Chinese, German, and Spanish (test sets). Differences in $F_1$ between our models and previous systems are statistically significant ($p < 0.05$) using stratified shuffling (Noreen, 1989).

supervised training indeed increases the robustness of our model, leading to more accurate predictions for both SRL and auxiliary tasks.

Table 4 presents the results of our experiments (without ELMo) on Chinese, German, and Spanish. Although we have not performed detailed parameter selection in these languages (i.e., we used the same parameters as in English), our model achieves state-of-the-art performance across all three languages.

### 3.2 Ablation Studies

To investigate the contribution of the sentence learner and cross-view training, we conducted a series of ablation studies on the English development set without predicate disambiguation.

Our experiments are summarized in Table 5. The first block shows the performance of the full model. In the second block, we assess the effect of different kinds of representations used in our model. Interestingly, the impact of ELMo (about 0.6 in $F_1$) is slightly less compared to multi-task hidden features (about 0.7 in $F_1$). This suggests that multi-task hidden features provide as useful information for SRL as pre-trained representations. We next eliminate the sentence learner model and have the semantic role labeler use the predicted POS tags and dependency labels provided in CoNLL-2009 dataset. As can be seen, this leads to a substantial drop in performance over the full model (1.4% in $F_1$).

| Model | P | R | $F_1$ |
|---|---|---|---|
| Ours | 88.6 | 86.8 | 87.7 |
| w/o ELMo | 87.9 | 86.4 | 87.1 |
| w/o multi-task hidden features | 88.0 | 86.1 | 87.0 |
| w/o sentence learner | 87.2 | 85.5 | 86.3 |
| w/o cross-view training | 88.0 | 85.8 | 86.9 |
| w/o splitting sentence | 87.4 | 85.7 | 86.6 |

Table 5: Ablation results on the CoNLL-2009 English development set.

| Preceding | Following | $F_1$ | $\Delta F_1$ |
|---|---|---|---|
| *Backward* | *Forward* | 87.3 | 0.4 |
| *Past* | *Future* | 87.4 | 0.5 |

Table 6: CVT with different auxiliary modules for SRL (CoNLL-2009 English development set). $\Delta$ denotes difference from model trained without CVT.

In the third block, we remove cross-view training from our model, and observe a 0.8% drop in $F_1$ over the full model. Finally, we apply the auxiliary modules on the full sentence instead of treating the words preceding and following the target predicate differently, and observe a 0.3% drop in $F_1$ over the supervised model. This is not surprising as the predicate indicator plays an important role in improving the performance of semantic role labeler.

### 3.3 CVT Analysis

In Table 6, we briefly explore which auxiliary prediction modules are more important for CVT when applied to SRL. We apply two types of auxiliary modules both of which take care not to "see" the target predicate directly. The "forward/backward" module does not see the right/left context of the current word, while the "future/past" module does not see the right/left context and the current token itself. Both kinds of modules improve performance (over a supervised model without CVT, see second row in Table 5); future and past modules are slightly better corroborating the results of Clark et al. (2018) on sequence tagging. Overall, the results in Table 6 suggest that more restricted views of the input are beneficial.

| Strategy | $F_1$ | $\Delta F_1$ |
|---|---|---|
| Randomly chosen word | 87.0 | 0.1 |
| Randomly chosen predicate | 87.7 | 0.8 |
| Most confident predicate | 86.4 | -0.5 |

Table 7: CVT with different predicate selection strategies for SRL (CoNLL-2009 English development set). $\Delta$ denotes difference from model trained without CVT.

We further explore how the strategy of selecting the target predicate (in sentences containing multiple candidates) influences performance. For each unlabeled sentence, we adopt the strategy of randomly selecting a predicate amongst those words identified as predicate candidates by the sentence learner. We could also select the predicate with the highest predicted score or a random word from the sentence. The experiments in Table 7 confirm that the adopted strategy works best delivering a 0.8 improvement in $F_1$ over a supervised model without CVT (second row in Table 5). Selecting the most confident predicate is the worst possible strategy, decreasing $F_1$ performance by 0.6 over a supervised model without CVT (see Table 5); the model concentrates on a few predicates with very high scores (these tend to be common verbs such as *say*, *is*, and *have*), while ignoring nominal predicates and less frequent verbs. The strategy of randomly selecting a word from the sentence performs better, precisely because it pays attention to a wider range of predicates.

## 4 Related Work

Our model resonates the recent trend of developing neural network models for semantic role labeling using relatively simple architectures based on bidirectional LSTMs (Marcheggiani et al., 2017; Cai et al., 2018; Strubell et al., 2018). It also agrees with previous work in adopting multi-task learning as a means to improve a main task by jointly learning one or more related auxiliary tasks (Collobert et al., 2011; Søgaard and Goldberg, 2016; Swayamdipta et al., 2017; Peng et al., 2017; Strubell et al., 2018).

The idea of resorting to semi-supervised learning as a means of reducing the annotation effort involved in creating labeled data for SRL is by no means new. Fürstenau and Lapata (2012) propose to augment a labeled dataset with unlabeled examples whose roles are inferred automatically via annotation projection. Other work uses a language model to learn word similarities from unlabeled texts (Croce et al., 2010; Deschacht and Moens, 2009) or constructs an informed prior from labeled data in order to learn a generative model from unlabeled data (Titov and Klementiev, 2012).

More recently, Mehta et al. (2018) have proposed a semi-supervised method for constituency-based SRL. Their work builds upon a state-of-the-art neural model (He et al., 2018b; Peters et al., 2018) whose training objective they augment with a syntactic inconsistency loss component. Their hypothesis is that by leveraging syntactic structure during training, the SRL model may become more robust in low resource scenarios. This method is very much geared towards improving constituent-based SRL, where syntactic constraints are widely used during decoding (He et al., 2017, 2018a). And requires a robust syntactic parser to analyze (out-of-domain) unlabeled sentences for consistency. Our model does not rely on external tools, and is generally applicable across semantic role representations based on dependencies or constituents (i.e., phrases or spans). However, we leave experiments on the latter for future work.

Although the focus of this work has been on semi-supervised learning, we have developed a competitive SRL system which could be used on its own, after being trained on labeled data. Following previous work (Strubell et al., 2018; Cai et al., 2018) we proposed an end-to-end model, which is able to distinguish predicates and label their arguments while learning a POS-tagger and a dependency parser. Importantly, our model can be simultaneously used for supervised and semi-supervised learning without modification. Cai et al. (2018) do not use any syntactic information which is critical when dealing with unlabeled data. Strubell et al. (2018) directly predict POS tags and predicates on top of the lower layers of their model; while this information is fed to the final SRL classifier, it is not propagated through the network, and is not shared with their multi-head self-attention layers.

## 5 Conclusions

In this paper we developed an end-to-end SRL model and demonstrated it can effectively leverage unlabeled data under the crossview training modeling paradigm. Experiments on the CONLL-2009 benchmark datasets show that our model delivers state of the art performance in English, Chi-

nese, German, and Spanish. Directions for future work are many and varied. We would like to apply the proposed model in low-resource settings, e.g., to transfer roles from English to another language via annotation projection or to learn an SRL model from weak supervision where only annotations for dependency labels are available.

## Acknowledgments

We thank the anonymous reviewers for their helpful feedback and suggestions. We gratefully acknowledge the support of the European Research Council (award number 681760, "Translating Multiple Modalities into Text").

## References

Wilker Aziz, Miguel Rios, and Lucia Specia. 2011. Shallow semantic trees for SMT. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 316–322, Edinburgh, Scotland.

Anders Björkelund, Bernd Bohnet, Love Hafdell, and Pierre Nugues. 2010. A high-performance syntactic and semantic dependency parser. In *Coling 2010: Demonstrations*, pages 33–36, Beijing, China.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Jiaxun Cai, Shexia He, Zuchao Li, and Hai Zhao. 2018. A full end-to-end semantic role labeler, syntactic-agnostic over syntactic-aware? In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2753–2765, Santa Fe, New Mexico, USA.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Jason Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Janara Christensen, Mausam, Stephen Soderland, and Oren Etzioni. 2011. An analysis of open information extraction based on semantic role labeling. In *Proceedings of the 6th International Conference on Konwledge Capture*, pages 113–119, Banff, Canada.

Kevin Clark, Minh-Thang Luong, Christopher D. Manning, and Quoc Le. 2018. Semi-supervised sequence modeling with cross-view training. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1914–1925, Brussels, Belgium.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Danilo Croce, Cristina Giannone, Paolo Annesi, and Roberto Basili. 2010. Towards open-domain semantic role labeling. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 237–246, Uppsala, Sweden.

Koen Deschacht and Marie-Francine Moens. 2009. Semi-supervised semantic role labeling using the Latent Words Language Model. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 21–29, Singapore.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343, Beijing, China.

Nicholas FitzGerald, Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic role labeling with neural network factors. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 960–970, Lisbon, Portugal.

Hagen Fürstenau and Mirella Lapata. 2012. Semi-supervised semantic role labeling via structural alignment. *Computational Linguistics*, 38(1):135–171.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059, New York, New York, USA.

Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2018a. Jointly predicting predicates and arguments in neural semantic role labeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 364–369, Melbourne, Australia.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 473–483, Vancouver, Canada.

Shexia He, Zuchao Li, Hai Zhao, and Hongxiao Bai. 2018b. Syntax for semantic role labeling, to be, or not to be. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*

*(Volume 1: Long Papers)*, pages 2061–2071, Melbourne, Australia.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9:1735–1780.

Atif Khan, Naomie Salim, and Yogan Jaya Kumar. 2015. A framework for multi-document abstractive summarization based on semantic role labelling. *Applied Soft Computing*, 30:737–747.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Tao Lei, Yuan Zhang, Lluís Màrquez, Alessandro Moschitti, and Regina Barzilay. 2015. High-order low-rank tensors for semantic role labeling. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1150–1160, Denver, Colorado.

Zuchao Li, Shexia He, Jiaxun Cai, Zhuosheng Zhang, Hai Zhao, Gongshen Liu, Linlin Li, and Luo Si. 2018. A unified syntax-aware framework for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2401–2411, Brussels, Belgium.

Wang Ling, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1299–1304, Denver, Colorado.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany.

Diego Marcheggiani, Joost Bastings, and Ivan Titov. 2018. Exploiting semantics in neural machine translation with graph convolutional networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 486–492, New Orleans, Louisiana.

Diego Marcheggiani, Anton Frolov, and Ivan Titov. 2017. A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 411–420, Vancouver, Canada.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1506–1515, Copenhagen, Denmark.

Sanket Vaibhav Mehta, Jay Yoon Lee, and Jaime Carbonell. 2018. Towards semi-supervised learning for deep semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4958–4963, Brussels, Belgium.

Eric W Noreen. 1989. *Computer-intensive methods for testing hypotheses*. Wiley New York.

Hao Peng, Sam Thomson, and Noah A. Smith. 2017. Deep multitask learning for semantic dependency parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2037–2048, Vancouver, Canada.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana.

Michael Roth and Mirella Lapata. 2016. Neural semantic role labeling with dependency path embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1192–1202, Berlin, Germany.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 231–235, Berlin, Germany.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5027–5038, Brussels, Belgium.

Swabha Swayamdipta, Sam Thomson, Chris Dyer, and Noah A Smith. 2017. Frame-semantic parsing with softmax-margin segmental RNNs and a syntactic scaffold. *arXiv preprint arXiv:1706.09528*.

Ivan Titov and Alexandre Klementiev. 2012. Semi-supervised semantic role labeling: Approaching from an unsupervised perspective. In *Proceedings of COLING 2012*, pages 2635–2652, Mumbai, India.

# Low-Resource Sequence Labeling via Unsupervised Multilingual Contextualized Representations

**Zuyi Bao**[‡], **Rui Huang**[◇], **Chen Li**[‡*] and **Kenny Q. Zhu**[†]
[‡]Alibaba Group
[◇]Zhejiang University, China
[†]Shanghai Jiao Tong University, Shanghai, China
[‡]{zuyi.bzy,puji.lc}@alibaba-inc.com
[◇]iurgnauh@zju.edu.cn, [†]kzhu@cs.sjtu.edu.cn

## Abstract

Previous work on cross-lingual sequence labeling tasks either requires parallel data or bridges the two languages through word-by-word matching. Such requirements and assumptions are infeasible for most languages, especially for languages with large linguistic distances, e.g., English and Chinese. In this work, we propose a Multilingual Language Model with deep semantic Alignment (MLMA) to generate language-independent representations for cross-lingual sequence labeling. Our methods require only monolingual corpora with no bilingual resources at all and take advantage of deep contextualized representations. Experimental results show that our approach achieves new state-of-the-art NER and POS performance across European languages, and is also effective on distant language pairs such as English and Chinese.[1]

## 1 Introduction

Sequence labeling tasks such as named entity recognition (NER) and part-of-speech (POS) tagging are fundamental problems in natural language processing (NLP). Recent sequence labeling models achieve state-of-the-art performance by combining both character-level and word-level information (Chiu and Nichols, 2016; Ma and Hovy, 2016; Lample et al., 2016). However, these models heavily rely on large-scale annotated training data, which may not be available in most languages. Cross-lingual transfer learning is proposed to address the label scarcity problem by transferring annotations from high-resource languages (source languages) to low-resource languages (target languages). In this scenario, a major challenge is how to bridge interlingual gaps with modest resource requirements.

There is a large body of work exploring cross-lingual transfer through language-independent features, such as morphological features and universal POS tags for cross-lingual NER (Tsai et al., 2016) and dependency parsers (McDonald et al., 2011). However, these approaches require linguistic knowledge for language-independent feature engineering, which is expensive in low-resource settings. Other work relies on bilingual resources to transfer knowledge from source languages to target languages. Parallel corpora are widely used to project annotations from the source to the target side (Yarowsky et al., 2001; Ehrmann et al., 2011; Kim et al., 2012; Wang and Manning, 2014). These methods could achieve strong performance with a large amount of bilingual data, which is scarce in low-resource settings.

Recent research leverages cross-lingual word embeddings (CLWEs) to establish inter-lingual connections and reduce the requirements of parallel data to a small lexicon or even no bilingual resource (Ni et al., 2017; Fang and Cohn, 2017; Xie et al., 2018). However, word embedding spaces may not be completely isomorphic due to language-specific linguistic properties, and therefore cannot be perfectly aligned. For example, different from English, Chinese nouns do not distinguish singular and plural forms, while Spanish nouns distinguish masculine and feminine.

On the other hand, NER tags such as person names, organizations, and locations are shared across different languages. Language-independent frameworks such as universal conceptual cognitive annotation (Abend and Rappoport, 2013), universal POS (Petrov et al., 2011a), and universal dependencies (Nivre et al., 2016) are defined to represent different languages in a unified formation. These work serves as our motivation to assume

*Corresponding Author.
† Kenny Q. Zhu was partially supported by NSFC grant 91646205 and Alibaba visiting scholar program.
[1]The code is released at https://github.com/baozuyi/MLMA.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1028–1039, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

that the semantic meanings of words from different languages can be roughly aligned at a conceptual level and it is more reasonable to align deep semantic representations instead of shallow word embeddings. Meanwhile, monolingual contextualized embeddings derived from language models have shown to be effective for extracting semantic information and have achieved significant improvement on several NLP tasks (Peters et al., 2018).

In this paper, we propose a Multilingual Language Model with deep semantic Alignment (MLMA). We train MLMA on monolingual corpora from each language and align its internal states across different languages. Then MLMA is utilized to generate language-independent representations and to bridge the gaps between high-resource and low-resource languages. For evaluation, we conduct extensive experiments on the NER and POS benchmark datasets under cross-lingual settings. The experiment results show that our methods achieve substantial improvements comparing to previous state-of-the-art methods in European languages. We also validate our approaches on a distant language pair, English-Chinese, and the results are competitive with previous methods which use large-scale parallel corpora. Our contributions are as follows:

1. Instead of word-level alignment, we propose MLMA that uses contextualized representations to bridge the inter-lingual gaps.
2. We propose three methods to align contextualized representations without any bilingual resource.
3. Our methods achieve new state-of-the-art performance on cross-lingual NER and POS tasks in European languages, and very competitive results for English-Chinese NER, where previous work uses large parallel data.

## 2 Approach

Our approach belongs to the model transfer (Section 5.2) and mainly consists of three steps:

1. Training a multilingual language model with alignment (MLMA) using monolingual corpora of the source and the target languages. (Section 2.1, 2.2 and 2.3)
2. Building a cross-lingual sequence labeling model based on the language-independent representations from the MLMA. (Section 2.4 and 2.5)
3. Learning the cross-lingual sequence labeling model (with MLMA fixed) on the annotated data of source languages and directly applying it to the target languages.

The architecture of MLMA is shown in Figure 1. In the following sections, we focus on introducing the Step 1 and 2. We first present the architecture of MLMA and describe how to build the unsupervised multilingual alignment. Next, we propose effective methods for collapsing the multi-layer hidden states from MLMA into a single representation. Finally, we introduce the sequence labeling model used in the experiments.

### 2.1 Language Model Architecture

MLMA is a language model with multi-head self-attention mechanism (Vaswani et al., 2017). The architecture is similar to Radford et al. (2018), except that we combine both a forward and a backward Transformer decoder to build a bidirectional language model. Take the forward direction as an example, given a sentence with $N$ tokens $W = [w_1, w_2, \cdots, w_N]^T$ as input, we first map the sequence of tokens $W$ to token embeddings $\overrightarrow{H}_0 \in \mathbb{R}^{N \times d}$:

$$\overrightarrow{H}_0 = W E_e + E_p \tag{1}$$

where $E_e$ and $E_p$ are the embedding matrix and the positional encoding matrix, and $d$ is the dimension of embeddings and hidden states.

Then $n$ blocks of transformer layers are stacked above the token embeddings. Each block contains a masked multi-head self-attention and a position-wise feedforward layer. The detailed implementation is the same as Vaswani et al. (2017).

$$\overrightarrow{H}_l = \text{TransformerLayer}(\overrightarrow{H}_{l-1}) \tag{2}$$

where $\overrightarrow{H}_l$ refers to the output of the $l$-th transformer block. Finally, the output distribution over the next tokens is calculated through a softmax function with tied embedding matrix.

$$\overrightarrow{P} = \text{softmax}(\overrightarrow{H}_n E_e^T) \tag{3}$$

For the backward direction, we calculate $\overleftarrow{H}_l$ and $\overleftarrow{P}$ in an analogous way. Finally, we jointly minimize the negative log likelihood of the forward and backward directions:

$$\begin{aligned} NLL = -\sum_{t=1}^{N} ( & \log p(w_t | w_1, \ldots, w_{t-1}) \\ & + \log p(w_t | w_{t+1}, \ldots, w_N)) \end{aligned} \tag{4}$$

In a multilingual setting, we share all param-

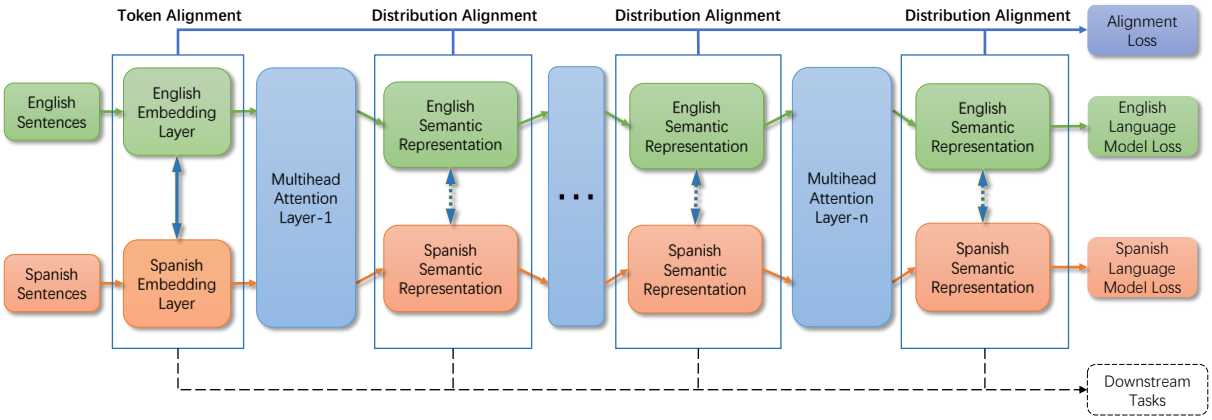


Figure 1: The architecture of MLMA consists of language-specific embedding layers and language-agnostic Transformer layers. MLMA is jointly learned through language modeling loss and alignment loss, and its internal representations are utilized to bridge the gap between source and target languages.

eters in Transformer layers across different languages to facilitate language-agnostic representations, except that we adopt an individual embedding matrix $E_e$ for each language.

### 2.2 Unsupervised Distribution Alignment

We find that only sharing Transformer layers is not enough for forcing hidden representations from different languages into a common space, as suggested in experiments (Section 3.4). Therefore, we propose three methods to build cross-lingual representations based on identical strings, mean/variance, and average linkage.

To simplify the description, we take the alignment between two languages $s$ and $t$ as an example, but our methods can be directly extended to a scenario with multiple languages by adding the alignment between each pair of languages.

#### Notation

For the language model, given a sentence with $N$ tokens, the forward internal representation $\overrightarrow{H}_l$ in Eq (2) can be expanded as $\overrightarrow{H}_l = [\overrightarrow{h}_{l,1}, \cdots, \overrightarrow{h}_{l,k}, \cdots, \overrightarrow{h}_{l,N}]^T$, where $\overrightarrow{h}_{l,k}$ refers to the forward hidden representation of the $k$-th token in the sentence. Then we concatenate the forward and backward hidden representations for each token, $h_{l,k} = \overrightarrow{h}_{l,k} \oplus \overleftarrow{h}_{l,k}$.

We denote the collection of the token representations $h_{l,k}$ at layer $l$ from the whole corpora of language $s$ as $C_l^s$, which can be regarded as a sampling from the deep semantic space of language $s$. Similarly, $C_l^t$ is used for language $t$.

#### Identical Strings

Similar language pairs such as English and Spanish have a large number of identical strings shared between their vocabularies, which are utilized as the seed dictionary for embedding alignment in previous work (Smith et al., 2017). Similarly, we treat identical strings as explicit supervision signals and align the embeddings of identical strings between different languages. The matching of the embeddings from different languages will lead to an implicit alignment of internal representations. In the experiments, we directly minimize the Euclidean distance between the embeddings of each identical string across different languages:

$$L_{iden} = \frac{\lambda^{iden}}{|W_{iden}^{(s,t)}|} \sum_{w \in W_{iden}^{(s,t)}} ||e_w^s - e_w^t||$$

where $W_{iden}^{(s,t)}$ is the set of identical strings between the vocabulary of language $s$ and language $t$, and $|W_{iden}^{(s,t)}|$ refers to the number of members in $W_{iden}^{(s,t)}$. $\lambda^{id}$ is a scaling weight, and $e_w^s$ ($e_w^t$) is the embedding of word $w$ from embedding matrix $E_e^s$ ($E_e^t$) of language $s$ ($t$).

#### Mean and Variance

In this section, we propose another approach to directly align the distributions of internal representations between different languages. In particular, we leverage the mean and variance of internal distributions for alignment. We denote the mean and variance of $C_l^s$ as $m_l^s$ and $v_l^s$. Similarly, $m_l^t$ and $v_l^t$ refer to the mean and variance of $C_l^t$. We minimize the Euclidean distance between the mean and variance of language $s$ and language $t$ for all layers:

$$L_{mv} = \sum_{l=0}^{n} (\lambda_l^m \cdot \frac{||m_l^s - m_l^t||}{|m_l^s| + |m_l^t|} + \lambda_l^v \cdot \frac{||v_l^s - v_l^t||}{|v_l^s| + |v_l^t|})$$

where $\lambda_l$ is a scaling weight, and $|\cdot|$ is the L1 norm of a vector. Without the denominators, the

model could escape this regularization by learning a mean and variance with low absolute values.

In practice, rather than calculating the mean and variance over the whole source and target corpora, we use the mean and variance of the source and target inner states $h_{l,k}$ in the current mini-batch as an approximation.

**Average Linkage**

In this method, we employ another metric, average linkage, to perform a more precise point-wise matching. The average linkage is a widely used metric for calculating the similarity of clusters and networks (Yim and Ramdeen, 2015; Seifoddini, 1989; Newman, 2012; Moseley and Wang, 2017). It is sensitive to the shape, thus serves as a better choice than mean and variance. The average linkage measures the similarity of two sets $X$ and $Y$ by calculating the averaged distance between all members of each set:

$$avl(X, Y) = \frac{1}{n_X \cdot n_Y} \sum_{x \in X} \sum_{y \in Y} f(x, y)$$

where $n_X$ ($n_Y$) is the number of members in $X$ ($Y$), and $f$ is a distance function. We take Euclidean distance as the distance function $f$ and minimize the average linkage between $C_l^s$ and $C_l^t$:

$$L_{avl} = \sum_{l=0}^{n} \lambda_l^{avl} \cdot [2 \cdot avl(C_l^s, C_l^t) - avl(C_l^s, C_l^s) - avl(C_l^t, C_l^t)]$$

Similarly, the terms $avl(C_l^s, C_l^s)$ and $avl(C_l^t, C_l^t)$ are used to prevent the model from escaping this regularization. In practice, we calculate $L_{avl}$ between the source and target inner states $h_{l,k}$ inside the mini-batch as an approximation.

The regularization $L_{avl}$ is similar to the maximum mean discrepancy (MMD), which is often employed in domain adaptation (Tzeng et al., 2014; Long et al., 2015) and style transfer (Li et al., 2017) for images. However, different from MMD, our method directly uses Euclidean distance instead of the kernel function.

### 2.3 Training of MLMA

During the training stage of MLMA, we sample equivalent number of sentences from the monolingual corpora of each language for each mini-batch. Then MLMA is optimized through a combination of the language modeling loss $L_{lm}$ and the alignment regularization loss $L_{reg}$. For each alignment method, we use its corresponding alignment loss:

$$\begin{aligned} L =& L_{lm} + L_{reg} \\ \text{where } L_{lm} =& \sum_{i \in \{s,t\}} \lambda_i^{lm} \cdot NLL_i, \\ & L_{reg} \in \{L_{id}, L_{mv}, L_{avl}\} \end{aligned}$$

where $\lambda_i^{lm}$ is used for balancing the convergence speed of different languages. $NLL_i$ is the negative log likelihood of language $i$ in Eq (4).

### 2.4 Cross-lingual Representations

After the MLMA is trained, we fix its parameters and extract the hidden states as cross-lingual contextualized representations (CLCRs). In this section, we propose two effective strategies for integrating these multi-layer high-dimensional representations into downstream models.

**Self-Weighted Sum** For each token, we concatenate all layers of hidden states and feed them into a multi-layer perceptron (MLP) to calculate a $(n+1)$-dimensional weight vector, $s = \text{softmax}(\text{MLP}(h_{0,k} \oplus \cdots \oplus h_{n,k}))$. Then we calculate a weighted sum of these layers according to the weight vector, $\mathbf{CLCR}_k = \sum_{l=0}^{n} s_l \cdot h_{l,k}$.

**Fully-Weighted Sum** We introduce a weight matrix, $F \in \mathbb{R}^{(n+1) \times 2d}$, with separate weights for each hidden dimension. The weight matrix $F$ is softmaxed by column and used to calculate a weighted sum of all layers for each hidden dimension, $\mathbf{CLCR}_k = \sum_{l=0}^{n} F_l \odot h_{l,k}$, where $\odot$ is the element-wise product.

The parameters of the MLP and $F$ are trained during the learning of sequence labeling model.

### 2.5 Sequence Labeling Model

The sequence labeling model is then built on the CLCRs. For both NER and POS tasks, we use an LSTM-CRF model following Lample et al. (2016), which consists of a character-level LSTM, a word-level LSTM, and a linear-chain CRF.

More specifically, given a sequence of words as $[w_1, w_2, \ldots, w_N]$, where $w_k$ is composed of a sequence of characters $[c_{k,1}, c_{k,2}, \ldots, c_{k,m}]$. First, for each word $w_k$, the character-level LSTM takes its character sequence $[c_{k,1}, c_{k,2}, \ldots, c_{k,m}]$ as input and outputs a vector $e_k$ to represent this word. Then the pre-trained $\mathbf{CLCR}_k$ is concatenated with $e_k$ to form a word-level embedding $x_k$. Finally, the sequence of word-level embeddings $[x_1, x_2, \ldots, x_N]$ are fed into the word-level LSTM, and the linear-chain CRF are employed to

predict the probability distribution for all possible output label sequences.

## 3 Experiments

We first introduce the datasets used in the experiment and then the implementation details of our models, before presenting the results on NER and POS tasks.

### 3.1 Datasets

For cross-lingual NER, we evaluate the proposed approaches on CoNLL 2002/2003 datasets (Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003), which contain four European languages, English (en), Spanish (es), Dutch (nl), German (de) and four entity types (person, location, organization, and MISC). We also evaluate a distant language pair, English-Chinese, on OntoNotes(v4.0) dataset (Hovy et al., 2006). We adopt the same dataset split and four valid entity types (person, location, organization, and GPE) as described in (Wang and Manning, 2014).

For cross-lingual POS, we use the Danish (da), Dutch (nl), German (de), Greek (el), Italian (it), Portuguese (pt), Spanish (es) and Swedish (sv) portion from CoNLL 2006/2007 dataset (Buchholz and Marsi, 2006; Nivre et al., 2007). Following previous work (Fang and Cohn, 2017), we train the sequence labeling model on Penn Treebank data and adopt the universal POS tagset (Petrov et al., 2011b).

In all cases, the sequence labeling model is trained on the source language (English) training data and is tested on the target language test data.

### 3.2 Details of MLMA

We adopt a 6-layer bi-directional Transformer decoder with 8 attention heads. The dimension size of hidden states and inner states are 512 and 2048, respectively. The dropout rates after attention and residual connection are both 0.1. We use the Adam optimization scheme (Kingma and Ba, 2014) with a learning rate of 0.0001 and a gradient clip norm of 5.0. The vocabulary size of each language is 200,000, and we train the model with a sampled softmax (Jean et al., 2015) of 8192 samples. We only keep the sentences containing less than 200 tokens for training and group them into batches by length. Each batch contains around 4096 tokens for each language. The language modeling weight $\lambda_i^{lm}$ is set to be 1.0 for each language. For alignment, $\lambda_l^m$, $\lambda_l^v$, $\lambda_l^{al}$ are set to be 0.1, 0.01 and 1.0 for every layer $l$, and $\lambda^{iden}$ is set to be 100.

For languages except English, the latest dump of Wikipedia is used as monolingual corpora. For English, we use 1B Word Benchmark (Chelba et al., 2013) to reduce the effects of potential internal alignment in Wikipedia (Zirikly and Hagiwara, 2015; Tsai et al., 2016).

All characters are preprocessed to lowercase, and Chinese text are converted into the simplified version through OpenCC[2]. The corpora of European languages are tokenized by nltk (Loper and Bird, 2002) and Chinese text is segmented using Ltp[3].

### 3.3 Details of Sequence Labeling Model

In our experiments, we set the hidden size of word-level LSTM and character-level LSTM to be 300 and 100, respectively. The character embedding size is set to be 100. We apply dropout at both the input and the output of word-level LSTM to prevent overfitting. The dropout rate is set to be 0.5. We train the sequence labeling model for 20 epochs using Adam optimizer with a batch size of 20 and perform an early stopping when there is no improvement for 3 epochs. We set the initial learning rate to be 0.001 and decay the learning rate by 0.1 for each epoch. We do not update the pre-trained cross-lingual deep representations from MLMA during training. For each model, we run it five times and report the mean and standard deviation. We disable the character-level LSTM in English-German and English-Chinese NER as they have a different character pattern from English. For POS, we disable the character-level LSTM following Fang and Cohn (2017).

### 3.4 Results for NER

We first train a multilingual language model without alignment (MLM) and report its performance of cross-lingual NER in Table 1. The poor performance demonstrates that only sharing part of the parameters in a language model is far from enough for cross-lingual transfer.

As shown in Table 1, the mean/variance alignment strategy (MLMA-Mv) is competitive with previous work which utilizes extra bilingual resources (Section 5.1). The average linkage strategy (MLMA-Avl) performs a more precise align-

[2] https://github.com/BYVoid/OpenCC
[3] https://github.com/HIT-SCIR/pyltp

| Model | es | nl | de | Extra Resources |
|---|---|---|---|---|
| MLM (w/o alignment) + s.w.s. | 21.16 ± 1.40 | 33.97 ± 1.49 | 15.46 ± 1.21 | None |
| MLM (w/o alignment) + f.w.s. | 23.61 ± 2.33 | 32.94 ± 1.62 | 16.38 ± 1.09 | None |
| Täckström et al. (2012) | 59.30 | 58.40 | 40.40 | parallel corpus |
| Nothman et al. (2013) | 61.00 | 64.00 | 55.80 | Wikipedia |
| Wang and Manning (2014) | - | - | 60.00 | parallel corpus |
| Tsai et al. (2016) | 60.55 | 61.60 | 48.10 | Wikipedia |
| Ni et al. (2017) | 65.10 | 65.40 | 58.50 | Wikipedia, parallel corpus, 5K dict. |
| Mayhew et al. (2017) | 65.95 | 66.50 | 59.11 | Wikipedia, 1M dict. |
| Xie et al. (2018) | 72.37 | 71.25 | 57.76 | None |
| MUSE* | 66.17 ± 1.15 | 65.52 ± 0.78 | 55.46 ± 0.59 | None |
| Multilingual BERT* | 66.42 ± 1.15 | 69.21 ± 0.48 | **70.78** ± 0.36 | None |
| MLMA-Iden + s.w.s. | 69.45 ± 0.91 | 68.82 ± 0.82 | 55.75 ± 1.64 | None |
| MLMA-Iden + f.w.s. | 67.10 ± 0.78 | 68.15 ± 0.67 | 55.25 ± 1.29 | None |
| MLMA-Mv + s.w.s. | 73.81 ± 0.83 | 70.61 ± 1.79 | 57.70 ± 0.71 | None |
| MLMA-Mv + f.w.s. | 74.12 ± 1.00 | 71.72 ± 0.70 | 57.84 ± 0.80 | None |
| MLMA-Avl + s.w.s. | 75.01 ± 0.79 | 76.22 ± 0.42 | 60.98 ± 1.00 | None |
| MLMA-Avl + f.w.s. | 74.43 ± 0.50 | 76.02 ± 0.55 | 60.50 ± 0.43 | None |
| MLMA-Avl (init) + s.w.s. | 75.72 ± 0.80 | **76.90** ± 0.30 | 63.01 ± 0.83 | None |
| MLMA-Avl (init) + f.w.s. | 76.30 ± 0.76 | 76.85 ± 0.43 | 62.85 ± 0.47 | None |
| MLMA-Avl (multi) + s.w.s. | **79.36** ± 0.57 | 74.89 ± 0.28 | 65.93 ± 0.32 | None |
| MLMA-Avl (multi) + f.w.s. | 79.34 ± 0.35 | 74.74 ± 0.40 | 66.53 ± 0.35 | None |

Table 1: NER F1 scores on test sets of European languages. For previous work which reports multiple results, we only list their best performance on each language. Results of methods with mark * are obtained by running their released source code or models. The results of MUSE embeddings are produced by using them for direct model transfer. "MLM" denotes our multilingual language model without alignment. The three alignment methods, **Iden** = identical strings, **Mv** = mean and variance, **Avl** = average linkage, respectively. "s.w.s" and "f.w.s." are self-weighted sum and fully-weighted sum. "init" represents using MUSE to initialize the embedding matrices in the MLMA. "multi" refer to the multi-source transfer.

| Model | zh |
|---|---|
| Wang and Manning (2014)◇ | **64.40** |
| MUSE* | 35.35 ± 0.84 |
| Xie et al. (2018)* | 44.13 ± 1.49 |
| *Our methods* | |
| MLMA-Iden + s.w.s. | 11.08 ± 0.89 |
| MLMA-Iden + f.w.s. | 11.17 ± 0.69 |
| MLMA-Avl + s.w.s. | 50.11 ± 1.51 |
| MLMA-Avl + f.w.s. | 45.88 ± 2.49 |
| MLMA-Avl (init) + s.w.s | 60.33 ± 1.39 |
| MLMA-Avl (init) + f.w.s | 58.92 ± 1.22 |

Table 2: NER F1 scores on test sets for Chinese. The notations are the same as Table 1. Methods with mark ◇ require parallel corpora.

ment and gains a further improvement. We conducted experiments of using all three alignments together, and results show no significant improvement over average linkage alone. These results agree with our statements that average linkage performs a more precise matching, and thus, carries the benefits brought by the other methods.

To demonstrate the strengths of the proposed cross-lingual contextualized representations (CLCRs) over cross-lingual word embeddings (CLWEs), we also report the results of using CLWEs for direct model transfer in Table 1. Specifically, we compare with the unsupervised method MUSE from Conneau et al. (2017). The experiment results demonstrate its effectiveness for cross-lingual sequence labeling. The alignment method using identical strings (MLMA-Iden) outperforms MUSE, suggesting that the contextual-level representations are more effective than the word-level ones. The other proposed methods (MLMA-Mv and MLMA-Avl) achieve significant improvement over MUSE and MLMA-Iden, which shows the benefit of directly aligning the contextualized representations.

**Combination with CLWEs** We further demonstrate that CLWEs are compatible with our methods by using MUSE embeddings to initialize the embedding layer of our multilingual language model. The results of MLMA-Avl (init) shown in Table 1 indicate that the CLWEs lead to a better initialization and improved performance.

**Multi-source Transfer** We conduct experiments of multi-source transfer based on method MLMA-Avl and report the performance as MLMA-Avl (multi) in Table 1. The experiment settings largely follow the previous work (Mayhew et al., 2017). They employ two source languages for each target language and use syntactic features to choose the related source languages. For Spanish and German, we use English and Dutch as source languages. English and Spanish are adopted for

| Model | es | nl | de | da | el | it | pt | sv | Avg. |
|---|---|---|---|---|---|---|---|---|---|
| Das and Petrov (2011)◇ | **84.2** | 79.5 | 82.8 | 83.2 | **82.5** | 86.8 | **87.9** | 80.5 | 83.31 |
| Fang and Cohn (2017)† | 68.40 | 64.50 | 65.90 | 73.5 | 65.5 | 64.8 | 67.8 | 66.0 | 67.05 |
| Fang and Cohn (2017)†‡ | 81.20 | 82.30 | 78.90 | 81.9 | 80.1 | 81.9 | 82.1 | 78.1 | 80.81 |
| Xie et al. (2018)* | 73.25 | 75.46 | 80.72 | 29.75 | 71.65 | 71.19 | 76.48 | 64.36 | 67.86 |
| MUSE* | 78.30 | 80.84 | 81.10 | 73.99 | 63.16 | 80.63 | 82.79 | 66.38 | 75.90 |
| Multilingual BERT* | 83.86 | 84.79 | **87.16** | **83.77** | 82.27 | **88.39** | 87.86 | **81.07** | **84.90** |
| *Our methods* | | | | | | | | | |
| MLMA-Avl + s.w.s. | 81.60 | 85.10 | 84.10 | 83.38 | 77.04 | 84.46 | 86.93 | 80.78 | 82.92 |
| MLMA-Avl + f.w.s. | 81.20 | 85.54 | 84.92 | 83.45 | 77.48 | 84.80 | 87.43 | 80.72 | 83.19 |
| MLMA-Avl (init) + s.w.s. | 82.73 | 85.79 | 85.76 | 82.44 | 80.55 | 86.76 | 85.99 | 79.75 | 83.72 |
| MLMA-Avl (init) + f.w.s. | 82.27 | **85.97** | 86.37 | 82.25 | 81.31 | 86.28 | 86.53 | 80.00 | 83.87 |

Table 3: POS accuracy on test sets of European languages. The notations are consistent with Table 1. Fang and Cohn (2017) report different results according to different resource requirements. We only list their best results in each setting. Methods with mark ◇, †, ‡ require parallel corpora, bilingual lexicons, and training data respectively.

Dutch. The multi-source transfer leads to a significant improvement for Spanish and German, but a slight decline for Dutch. In the follow-up experiment, we find that the Spanish training set achieves a poor cross-lingual performance on Dutch. Similar results are observed in the experiments of Spanish to English and Dutch to English. These results suggest that the cross-lingual transfer may be directional, and we leave this issue for future work.

**Comparison with BERT** We also compare the performance of our MLMA with the released multilingual BERT (Devlin et al., 2018). As shown in Table 1, our MLMA-Avl achieves a better performance on Spanish and Dutch. For German, BERT achieves a high performance as it employs effective subword information through BPE. The architecture of BERT also performs better than LSTM.

It is worth mentioning that, in previous work and this work, the corpora used in the experiments are limited to the source and the target language. In contrast, the multilingual BERT is jointly learned on Wikipedia of 102 languages and may benefit from a multi-hop transfer. BERT employs a shared BPE vocabulary for different languages, which implicitly performs a subword alignment similar to MLMA-Iden. Meanwhile, the proposed MLMA-Mv and MLMA-Avl methods are compatible with BERT and can be used to align the inner states of BERT.

### 3.5 A Case Study of Chinese NER

We conduct experiments and evaluate our approaches on a distant language pair, English-Chinese. The experiment results are shown in Table 2. Wang and Manning (2014) utilize 80K parallel sentences for annotation projection and report a strong performance. As Chinese and English do not share the alphabet, the number of identical strings is significantly smaller than similar languages pairs such as English-Spanish. Therefore, the MLMA-Iden achieves a lower result comparing to MUSE which uses adversarial training. The MLMA-Avl method performs a direct alignment of internal representations and achieves a significant improvement over the word-level methods. The initialization from CLWEs also proves its effectiveness for distant language pairs by gaining further improvement and reaching a comparable result with Wang and Manning (2014). This experiment suggests that cross-lingual transfer is still challenging between distant language pairs.

### 3.6 Results for POS

We evaluate our methods on another sequence labeling task POS, and the results are shown in Table 3. We compare with previous studies using unsupervised cross-lingual clustering (Fang and Cohn, 2017) and large-scale parallel corpora (Das and Petrov, 2011). As shown in Table 3, our models with deep semantic alignment outperform previous lexicon-based cross-lingual clustering by a large margin. When comparing to the previous method with a small amount of training data, the MLMA-Avl method obtains an improved accuracy without training data in the target languages. For further comparison, We also list the performance of applying the method from Xie et al. (2018) and

| MUSE |
|---|
| brown: oliváceo (olive), negruzcas (blackish), negruzco (blackish), marrón (brown), ocráceo (ochraceous) |
| chair: vicepresidenta (vice president), vicedecano (vice dean), cátedra (chair), vicedecana (vice dean), catedrático (professor) |

| MLMA-Avl | |
|---|---|
| **[Brown]**'s office told news outlets of his visit to Afghanistan ... | **[Neira]** escapó meses después rumbo a Miami para ...<br>(**Neira** escaped months later heading to Miami to ...) |
| Wearing a **[brown]** suit with matching hat, ... | La corona y vientre del macho son de un **[verde]** esmeralda brillante iridiscente, ...<br>(The crown and belly of the male are of an iridescent bright emerald **green**, ...) |
| Sweden currently holds the EU **[chair]**. | Tras tomar posesión de su **[asiento]** , Lois decide limpiar el lago para empezar, ...<br>(After taking possession of her **seat**, Lois decides to clean the lake to begin, ... ) |
| It's an honor to be asked to **[chair]** the Man Booker Prize, ... | ..., y fue la primera mujer en **[presidir]** un sindicato AFL-CIO.<br>(..., and was the first woman to **preside** over an AFL-CIO union.) |

Table 4: English words and their nearest Spanish words according to MUSE and MLMA-Avl.

multilingual BERT to the POS task.[4]

POS mainly relies on the information of each single word, and parallel corpora providing word alignment are effective for cross-lingual POS. Thus, previous annotation projection methods through parallel corpora are strong approaches for cross-lingual POS and often achieve a significantly better performance against previous unsupervised methods. The experimental results show that the proposed CLCRs are competitive and even achieve better average accuracy.

### 3.7 Self-Weighted v.s. Fully-Weighted Sum

As shown in Table 1, 2 and 3, we observe that Self-Weighted Sum (SWS) generally outperforms Fully-Weighted Sum (FWS) in NER tasks, while the opposite is true for POS tasks. SWS allows weights to vary at each position in a sequence, while FWS imposes adaptive weights for each hidden dimension. We hypothesize that NER is more context-sensitive and requires models to adapt to different context information, which makes SWS a better option. On the other hand, the POS of words is more independent across different context, but certain feature dimensions in contextualized representations may be critical for making a judgment. Therefore, FWS has the edge over SWS for its ability to select out these dimensions.

## 4 What is Connected during Alignment?

In this section, we dive into the MLMA and investigate the question of what is connected between different languages during the alignment. From English 1B and Spanish Wikipedia, we randomly select 1,000 sentences for each language and extract their cross-lingual contextual representations using our MLMA-Avl model. We calculate the nearest neighbors in cosine distance for each word, and some of them are listed in Table 4.

In these cases, the MLMA can disambiguate word senses according to context information. For example, for the word *brown* in English, the MLMA groups color *brown* with *verde* (green), and name *Brown* with *Neira* (a person name in Spanish) in the Spanish corpus. The proposed method is different from unsupervised translation in that, instead of learning a precise matching between English and Spanish words, the CLCRs establishes a high-level semantic connection between the source and the target language. The next example demonstrates that the MLMA is able to distinguish the part-of-speech of words. It connects an English verb *chair* with a Spanish verb *presidir* (preside), while a noun *chair* with a noun *asiento* (seat) in Spanish. To compare with unsupervised cross-lingual word embeddings, we list the top 5 similar words calculated using MUSE. As shown in Table 4, MUSE successfully groups the English word *brown* with Spanish words that are related to colors. However, without the help of contextual information, its ability of word sense disambiguation is limited.

## 5 Related Work

Previous work in cross-lingual transfer learning can be roughly divided into two main branches: annotation projection and model transfer.

### 5.1 Annotation Projection

In annotation projection approaches, parallel or comparable corpora are commonly used (Yarowsky et al., 2001; Ehrmann et al., 2011; Das and Petrov, 2011; Li et al., 2012; Täckström et al.,

[4]The poor performance of Xie et al. (2018) on en-da is due to the low quality of word translation pairs generated by their method.

2013; Wang and Manning, 2014; Ni et al., 2017). The source language sentences of parallel corpora are first annotated either manually or by a pre-trained tagger. Then, annotations on the source side are projected to the target side through word alignment to generate distantly supervised training data. Finally, a model of the target language is trained on the generated data. Wikipedia contains multilingual articles for various topics and can thus be used to generate parallel/comparable corpora or even weakly annotated target language sentences (Kim et al., 2012).

However, parallel corpora and Wikipedia can be rare for true low-resource languages. Mayhew et al. (2017) reduce the resource requirement by proposing a cheap translation method, which "translates" the training data from the source to the target language word by word through a bilingual lexicon. While Xie et al. (2018) reduce the requirement of bilingual lexicons by an unsupervised word-by-word translation through CLWEs.

### 5.2 Model Transfer

Model transfer methods train a model on the source language with language-independent features. Thus, the trained model can be directly applied to the target language.

McDonald et al. (2011) design a cross-lingual parser based on delexicalized features like universal POS tags. Täckström et al. (2012) reveal that cross-lingual word cluster features induced using large parallel corpora are useful. Lexicon and Wikipedia also demonstrate effectiveness for language-independent feature engineering. Zirikly and Hagiwara (2015) generate multilingual gazetteers from the source language gazetteers and comparable corpus. Page categories and linkage information to entries from Wikipedia are extracted as strong language-independent features (wikifier features) (Tsai et al., 2016). Bharadwaj et al. (2016) facilitate the cross-lingual transfer through phonetic features, which work well between languages like Turkish, Uzbek, and Uyghur, but are not strictly language independent. Recently, CLWEs are used as language-invariant representations for direct model transfer in NER (Ni et al., 2017) and POS (Fang and Cohn, 2017).

Some of the previous work also proposes sequence labeling models with shared parameters between languages for performing cross-lingual knowledge transfer (Lin et al., 2018; Cotterell and Duh, 2017; Yang et al., 2017; Ammar et al., 2016; Kim et al., 2017). However, these models are usually obtained through joint learning and require annotated data from the target language.

## 6 Conclusion

In this paper, we focused on a low-resources cross-lingual setting and proposed transfer learning methods based on the alignment of deep semantic spaces between different languages. The proposed multilingual language model bridges different languages by automatically learning cross-lingual disambiguated representations. Abundant NER and POS experiments are conducted on the benchmark datasets. Experimental results show that our approaches using only monolingual corpora achieve improved performance comparing to previous strong cross-lingual studies with extra resources.

## References


Omri Abend and Ari Rappoport. 2013. Universal conceptual cognitive annotation (UCCA). In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 228–238, Sofia, Bulgaria. Association for Computational Linguistics.

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah Smith. 2016. Many languages, one parser. *Transactions of the Association for Computational Linguistics*, 4:431–444.

Akash Bharadwaj, David Mortensen, Chris Dyer, and Jaime Carbonell. 2016. Phonologically aware neural model for named entity recognition in low resource transfer settings. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1462–1472. Association for Computational Linguistics.

Sabine Buchholz and Erwin Marsi. 2006. Conll-x shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, CoNLL-X '06, pages 149–164, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Jason Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Ryan Cotterell and Kevin Duh. 2017. Low-resource named entity recognition with cross-lingual, character-level neural conditional random fields. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 91–96. Asian Federation of Natural Language Processing.

Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 600–609, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Maud Ehrmann, Marco Turchi, and Ralf Steinberger. 2011. Building a multilingual named entity-annotated corpus using annotation projection. In *Proceedings of the International Conference Recent Advances in Natural Language Processing 2011*, pages 118–124. Association for Computational Linguistics.

Meng Fang and Trevor Cohn. 2017. Model transfer for tagging low-resource languages using a bilingual dictionary. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 587–593. Association for Computational Linguistics.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: The 90% solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, NAACL-Short '06, pages 57–60, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10. Association for Computational Linguistics.

Joo-Kyung Kim, Young-Bum Kim, Ruhi Sarikaya, and Eric Fosler-Lussier. 2017. Cross-lingual transfer learning for pos tagging without cross-lingual resources. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2832–2838. Association for Computational Linguistics.

Sungchul Kim, Kristina Toutanova, and Hwanjo Yu. 2012. Multilingual named entity recognition using parallel data and metadata from wikipedia. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 694–702. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270. Association for Computational Linguistics.

Qi Li, Haibo Li, Heng Ji, Wen Wang, Jing Zheng, and Fei Huang. 2012. Joint bilingual name tagging for parallel corpora. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 1727–1731, New York, NY, USA. ACM.

Yanghao Li, Naiyan Wang, Jiaying Liu, and Xiaodi Hou. 2017. Demystifying neural style transfer. *arXiv preprint arXiv:1701.01036*.

Ying Lin, Shengqi Yang, Veselin Stoyanov, and Heng Ji. 2018. A multi-lingual multi-task architecture for low-resource sequence labeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 799–809. Association for Computational Linguistics.

Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I Jordan. 2015. Learning transferable features with deep adaptation networks. *arXiv preprint arXiv:1502.02791*.

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics-Volume 1*, pages 63–70. Association for Computational Linguistics.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074. Association for Computational Linguistics.

Stephen Mayhew, Chen-Tse Tsai, and Dan Roth. 2017. Cheap translation for cross-lingual named entity recognition. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2536–2545. Association for Computational Linguistics.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 62–72. Association for Computational Linguistics.

Benjamin Moseley and Joshua Wang. 2017. Approximation bounds for hierarchical clustering: Average linkage, bisecting k-means, and local search. In *Advances in Neural Information Processing Systems*, pages 3094–3103.

Mark EJ Newman. 2012. Communities, modules and large-scale structure in networks. *Nature physics*, 8(1):25.

Jian Ni, Georgiana Dinu, and Radu Florian. 2017. Weakly supervised cross-lingual named entity recognition via effective annotation and representation projection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1470–1480. Association for Computational Linguistics.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The conll 2007 shared task on dependency parsing. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*.

Joel Nothman, Nicky Ringland, Will Radford, Tara Murphy, and James R. Curran. 2013. Learning multilingual named entity recognition from wikipedia. *Artif. Intell.*, 194:151–175.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2011a. A universal part-of-speech tagset. *arXiv preprint arXiv:1104.2086*.

Slav Petrov, Dipanjan Das, and Ryan T. McDonald. 2011b. A universal part-of-speech tagset. *CoRR*, abs/1104.2086.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Hamid K Seifoddini. 1989. Single linkage versus average linkage clustering in machine cells formation applications. *Computers & Industrial Engineering*, 16(3):419–426.

Samuel L Smith, David HP Turban, Steven Hamblin, and Nils Y Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. *arXiv preprint arXiv:1702.03859*.

Oscar Täckström, Dipanjan Das, Slav Petrov, Ryan McDonald, and Joakim Nivre. 2013. Token and type constraints for cross-lingual part-of-speech tagging. *Transactions of the Association for Computational Linguistics*, 1:1–12.

Oscar Täckström, Ryan McDonald, and Jakob Uszkoreit. 2012. Cross-lingual word clusters for direct transfer of linguistic structure. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 477–487. Association for Computational Linguistics.

Erik F. Tjong Kim Sang. 2002. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Chen-Tse Tsai, Stephen Mayhew, and Dan Roth. 2016. Cross-lingual named entity recognition via wikification. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 219–228. Association for Computational Linguistics.

Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. 2014. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Mengqiu Wang and Christopher D. Manning. 2014. Cross-lingual projected expectation regularization for weakly supervised learning. *Transactions of the Association for Computational Linguistics*, 2:55–66.

Jiateng Xie, Zhilin Yang, Graham Neubig, Noah A Smith, and Jaime Carbonell. 2018. Neural cross-lingual named entity recognition with minimal resources. *arXiv preprint arXiv:1808.09861*.

Zhilin Yang, Ruslan Salakhutdinov, and William W Cohen. 2017. Transfer learning for sequence tagging with hierarchical recurrent networks. *arXiv preprint arXiv:1703.06345*.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of the First International Conference on Human Language Technology Research*, HLT '01, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

Odilia Yim and Kylee T Ramdeen. 2015. Hierarchical cluster analysis: comparison of three linkage measures and application to psychological data. *The quantitative methods for psychology*, 11(1):8–21.

Ayah Zirikly and Masato Hagiwara. 2015. Cross-lingual transfer of named entity recognizers without parallel corpora. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 390–396. Association for Computational Linguistics.

# A Lexicon-Based Graph Neural Network for Chinese NER

**Tao Gui***, **Yicheng Zou***, **Qi Zhang,**
**Minlong Peng, Jinlan Fu, Zhongyu Wei, Xuanjing Huang**
Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
School of Computer Science, Fudan University
825 Zhangheng Road, Shanghai, China
{tgui16,yczou18,qz,mlpeng16,fujl16,zywei,xjhuang}@fudan.edu.cn

## Abstract

Recurrent neural networks (RNN) used for Chinese named entity recognition (NER) that sequentially track character and word information have achieved great success. However, the characteristic of chain structure and the lack of global semantics determine that RNN-based models are vulnerable to word ambiguities. In this work, we try to alleviate this problem by introducing a lexicon-based graph neural network with global semantics, in which lexicon knowledge is used to connect characters to capture the local composition, while a global relay node can capture global sentence semantics and long-range dependency. Based on the multiple graph-based interactions among characters, potential words, and the whole-sentence semantics, word ambiguities can be effectively tackled. Experiments on four NER datasets show that the proposed model achieves significant improvements against other baseline models.

## 1 Introduction

The task of named entity recognition (NER) involves determining entity boundaries and recognizing categories of named entities, which is a fundamental task in the field of natural language processing (NLP). NER plays an important role in many downstream NLP tasks, including information retrieval (Chen et al., 2015b), relation extraction (Bunescu and Mooney, 2005), question answering systems (Diefenbach et al., 2018), and other applications. Compared with English NER, Chinese named entities are more difficult to identify due to their uncertain boundaries, complex composition, and NE definitions within the nest (Duan and Zheng, 2011).

One intuitive way to alleviate word boundary problems is to first perform word segmentation and then apply word sequence labeling (Yang et al., 2016; He and Sun, 2017). However, the rare gold-standard segmentation in NER datasets and incorrectly segmented entity boundaries both negatively impact the identification of named entities (Peng and Dredze, 2015; He and Sun, 2016). Hence, character-level Chinese NER using lexicon features to better leverage word information has attracted research attention (Passos et al., 2014; Zhang and Yang, 2018). In particular, Zhang and Yang (2018) introduced a variant of a long short-term memory network (lattice-structured LSTM) that encodes all potential words matching a sentence to exploit explicit word information, achieving state-of-the-art results.

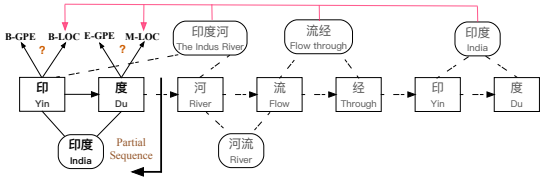


Figure 1: Example of word character lattice with partial input. Because of the characteristic of chain structure, RNN-based methods must predict the label "度" using only previous partial sequences "印度 (India)", which may suffer from word ambiguities without global sentence semantics.

However, these methods are usually based on RNN or CRF to sequentially encode a sentence, while the underlying structure of language is not strictly sequential (Shen et al., 2019). As a result, these models would encounter serious word ambiguity problems (Mich et al., 2000). Especially in Chinese texts, the recognition of named entities with overlapping ambiguous strings is even more challenging. As shown in Figure 1, the middle character of an overlapping ambiguous string can constitute words with the characters to both their left and their right (Yen et al., 2012), such as "河

*Equal contribution.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1040–1050,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

流 (River)" and "流经 (Flow through)", which share a common character "流". However, RNN-based models process characters in a strictly serial order, which is similar to reading Chinese, and a character has priority in being assigned to the word on the left (Perfetti and Tan, 1999). More seriously, RNN-based models must give the label of "度" using only previous partial sequences "印度 (India)", which is problematic without seeing the remaining characters. Hence, Ma et al. (2014) suggested that the overlapping ambiguity must be settled using sentence context and high-level information.

In this work, we introduce a lexicon-based graph neural network (LGN) that achieves Chinese NER as a node classification task. The proposed model breaks the serialization processing structure of RNNs with better interaction results between characters and words through careful connections. The lexicon knowledge connects related characters to capture the local composition. Meanwhile, a global relay node is designed to capture long-range dependency and high-level features. LGN follows a neighborhood aggregation scheme wherein the node representation is computed by recursively aggregating its incoming edges and the global relay node. Because of multiple iterations of aggregation, the model can use global context information to repeatedly compare ambiguous words for better prediction. Experimental results show that the proposed method can achieve state-of-the-art performance on four NER datasets.

The main contributions of this paper can be summarized as follows: 1) we propose the use of a lexicon to construct a graph neural network and achieve Chinese NER as a graph node classification task; 2) the proposed model can capture global context information and local compositions to tackle Chinese word ambiguity problems through recursively aggregating mechanism; 3) several experimental results demonstrate the effectiveness of the proposed method in different aspects.

## 2 Related Work

### 2.1 Chinese NER with Lexicon.

Some previous Chinese NER researches have compared word-based and character-based methods (Li et al., 2014) and show that due to the limited performance of the current Chinese word segmentation, character-based name taggers can outperform their word-based counterparts (He and Wang, 2008; Liu et al., 2010). Lexicon features have been widely used to better leverage word information for Chinese NER (Huang et al., 2015; Luo et al., 2015; Gui et al., 2019). Especially, Zhang and Yang (2018) proposed a lattice LSTM to model characters and potential words simultaneously. However, their lattice LSTM used a concatenation of independently trained left-to-right and right-to-left LSTM to represent features, which was also limited (Devlin et al., 2018). In this work, we propose a novel character-based method that treats the named entities as a node classification task. The proposed method can utilize global information (both the left and the right context) (Dong et al., 2019) to tackle word ambiguities.

### 2.2 Graph Neural Networks on Texts

Graph neural networks have been successfully applied to several text classification tasks (Veličković et al., 2017; Yao et al., 2018; Zhang et al., 2018b). Peng et al. (2018) proposed a GCN-based deep learning model for text classification. Zhang et al. (2018c) proposed using the dependency parse trees to construct a graph for relation extraction. Recently, multi-head attention mechanisms (Vaswani et al., 2017) have been widely used by graph neural networks during the fusion process (Zhang et al., 2018a; Lee et al., 2018), which can aggregate graph information by assigning different weights to neighboring nodes or associated edges. Given a set of vectors $\mathbf{H} \in \mathbb{R}^{n\times d}$, a query vector $\hat{\mathbf{q}} \in \mathbb{R}^{1\times d}$, and a set of trainable parameters $\mathbf{W}$, this mechanism can be formulated as:

$$\begin{aligned}
&\text{Attention}(\mathbf{q},\mathbf{K},\mathbf{V}) = softmax(\frac{\mathbf{q}\mathbf{K}^\top}{\sqrt{d_k}})\mathbf{V} \\
&\text{head}_i = \text{Attention}(\hat{\mathbf{q}}\mathbf{W}_i^Q, \mathbf{H}\mathbf{W}_i^K, \mathbf{H}\mathbf{W}_i^V) \\
&\text{MultiAtt}(\hat{\mathbf{q}},\mathbf{H}) = [\text{head}_1;\ldots;\text{head}_k]\mathbf{W}^O. \quad (1)
\end{aligned}$$

However, very little work has explored how to use the relationship among characters to construct graphs in raw Chinese texts. The few previous studies on morphological processing in Chinese proposed a decomposed lexical structure (Zhang and Peng, 1992; Zhou and Marslen-Wilson, 1994) in which Chinese words are represented in terms of their constituent characters. Inspired by these

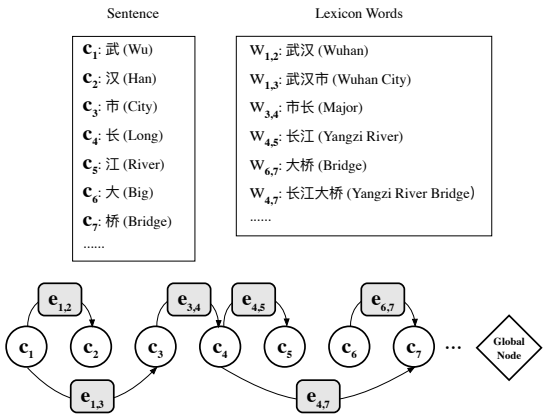


Figure 2: Illustration of graph construction.

theoretical basis, we propose the use of graph neural networks to construct the relationship between constituent characters and words.

## 3 Lexicon-Based Graph Neural Network

In this work, we propose the use of lexicon information to construct graph neural networks, and achieve Chinese NER as a node classification task. The proposed model obtains better interaction among characters, words, and sentences through, **aggregation** $\rightarrow$ **update** $\rightarrow$ **aggregation** $\rightarrow$ ..., an efficient graph message passing architecture (Gilmer et al., 2017).

### 3.1 Graph Construction and Aggregation

We use the lexicon knowledge to connect characters to capture the local composition and potential word boundaries. In addition, we propose a global relay node to capture long-range dependency and high-level features. The implementation of the aggregation module for nodes and edges is similar to the multi-head attention mechanism in Transformer (Vaswani et al., 2017).

**Graph Construction** The whole sentence is converted into a directed graph, as shown in Figure 2, where each node represents a character and the connection between the first and last characters in a word can be treated as an edge. The state of the $i$-th node represents the features of the $i$-th token in a text sequence. The state of each edge represents the features of a corresponding potential word. The global relay node is used as a virtual hub to gather the information from all the nodes and edges, and then utilizes the global information to help the node remove ambiguity.

Formally, let $s = c_1, c_2, ..., c_n$ denote a sentence, where $c_i$ denotes the $i$-th character. The

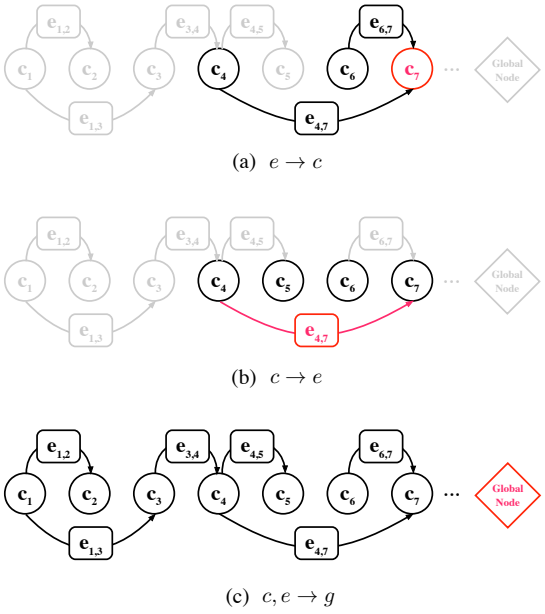


(a) $e \rightarrow c$

(b) $c \rightarrow e$

(c) $c, e \rightarrow g$

Figure 3: Aggregation in LGN. Red indicates the element that is being updated, and black indicates other elements involved in the aggregation. The aggregation results are then used in update modules.

potential words in the lexicon that match a character subsequence can be formulated as $w_{b,e} = c_b, c_{b+1}, ..., c_{e-1}, c_e$, where the index of the first and last letters are $b$ and $e$, respectively. In this work, we propose the use of a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to model a sentence, where each character $c_i \in \mathcal{V}$ is a graph node and $\mathcal{E}$ is the set of edges. Once a character subsequence matches a potential word $w_{b,e}$, we construct one edge $e_{b,e} \in \mathcal{E}$, pointing from the beginning character $c_b$ to the ending character $c_e$.

To capture global information, we add a global relay node to connect each character node and word edge. For a graph with $n$ character nodes and $m$ edges, there are $n + m$ global connections linking each node and edge to the shared relay node. With the global connections, every two non-adjacent nodes are two-hop neighbors and receive non-local information with a two-step update.

In addition, we consider the transpose of the constructed graph[1]. It is another directed graph on the same set of nodes with all of the edges reversed compared to the orientation of the corresponding edges in $\mathcal{G}$. We denote the transpose graph as $\mathcal{G}^\top$. Similar to the bidirectional LSTM, we compose $\mathcal{G}$ and $\mathcal{G}^\top$ as a bidirectional graph and concatenate the node states of $\mathcal{G}$ and $\mathcal{G}^\top$ as final outputs.

**Local Aggregation** Given the node features $\mathbf{c}_i^t$

[1] https://en.wikipedia.org/wiki/Transpose_graph

and the incoming edge features $\mathbf{E}^t_{c_i} = \{\forall_k \mathbf{e}^t_{k,i}\}$, we use multi-head attention to aggregate $\mathbf{e}_{k,i}$ and the corresponding predecessor nodes $\mathbf{c}_k$ for each node $\mathbf{c}_i$, where intuition is that the incoming edges and predecessor nodes can effectively indicate potential word boundary information, as shown in Figure 3 (a). Formally, the node aggregation function can be formulated as follows:

$$e \rightarrow c : \hat{\mathbf{c}}^t_i = \text{MultiAtt}(\mathbf{c}^t_i, \{\forall_k [\mathbf{c}^t_k; \mathbf{e}^t_{k,i}]\}), \quad (2)$$

where $t$ refers to the aggregation at the $t$-th step and $[\cdot;\cdot]$ represents concatenation operation.

For edge aggregation, all the forces or potential energies acting on the edges should be considered (Battaglia et al., 2018). To exploit the word orthographic information, lexicons used to construct edges should consider all the character composition, as shown in Figure 3 (b). Hence, different from the classic graph neural networks that use the features of terminal vertices to aggregate edges, we use the whole matching character subsequence $\mathbf{C}^t_{b,e} = \{\mathbf{c}^t_b, \ldots, \mathbf{c}^t_e\}$ for the edge aggregation function, as follows:

$$c \rightarrow e : \quad \hat{\mathbf{e}}^t_{b,e} = \text{MultiAtt}(\mathbf{e}^t_{b,e}, \mathbf{C}^t_{b,e}). \quad (3)$$

Given the character sequence embeddings $\mathbf{C} \in \mathbb{R}^{n \times d}$ and potential word embeddings $\mathbf{E} \in \mathbb{R}^{m \times d}$, we first fed $\mathbf{C}$ into an LSTM network to generate contextualized representations as the initial node states $\mathbf{C}^0$ (Zhang et al., 2018c), and we used the word embeddings as the initial edge states $\mathbf{E}^0$.

**Global Aggregation** The underlying structure of language is not strictly sequential (Shen et al., 2019). To capture long-range dependency and high-level features, as shown in Figure 3 (c), we utilized a global relay node to aggregate each character node and edge, as follows:

$$\begin{aligned} \mathbf{g}^t_c &= \text{MultiAtt}(\mathbf{g}^t, \mathbf{C}^t_{1,n}) \\ \mathbf{g}^t_e &= \text{MultiAtt}(\mathbf{g}^t, \{\forall \mathbf{e}^t \in \mathcal{E}\}) \\ c, e \rightarrow g : \quad \hat{\mathbf{g}}^t &= [\mathbf{g}^t_c; \mathbf{g}^t_e]. \end{aligned} \quad (4)$$

After multiple exchanges of information (§ 3.2), $\hat{\mathbf{g}}_t$ aggregates node vectors and edge vectors to summarize the global information, and $\hat{\mathbf{e}}^t_{b,e}$ captures the compositional character information to form the local composition. As a result, the proposed model, with a thorough knowledge of both local and non-local composition, would contribute character nodes to distinguish ambiguous words (Ma et al., 2014).

### 3.2 Recurrent-based Update Module

**Node Update** The effective use of sentence context to tackle the ambiguity among the potential words is still a key issue (Ma et al., 2014). For a general graph, it is common practice to apply recurrent-based modules to update hidden representations of nodes (Scarselli et al., 2009; Li et al., 2015). Hence, we fused the global feature $\hat{\mathbf{g}}$ into a character nodes update module, as follows:

$$\begin{aligned} \boldsymbol{\xi}^t_i &= [\mathbf{c}^{t-1}_{i-1}; \mathbf{c}^{t-1}_i], \quad \boldsymbol{\chi}^t_i = [\hat{\mathbf{c}}^{t-1}_i; \hat{\mathbf{g}}^{t-1}] \\ \hat{\boldsymbol{a}}^t_i &= \sigma(\mathbf{W}^a_i \boldsymbol{\xi}^t_i + \mathbf{V}^a_i \boldsymbol{\chi}^t_i + \mathbf{b}^a_i), \quad \boldsymbol{a} = \{\boldsymbol{i}, \boldsymbol{f}, \boldsymbol{l}\} \\ \boldsymbol{u}^t_i &= tanh(\mathbf{W}_{cu} \boldsymbol{\xi}^t_i + \mathbf{V}_{cu} \boldsymbol{\chi}^t_i + \mathbf{b}_{cu}) \\ \boldsymbol{i}^t_i, \boldsymbol{f}^t_i, \boldsymbol{l}^t_i &= softmax(\hat{\boldsymbol{i}}^t_i, \hat{\boldsymbol{f}}^t_i, \hat{\boldsymbol{l}}^t_i) \\ \mathbf{c}^t_i &= \boldsymbol{l}^t_i \odot \mathbf{c}^{t-1}_{i-1} + \boldsymbol{f}^t_i \odot \mathbf{c}^{t-1}_i + \boldsymbol{i}^t_i \odot \boldsymbol{u}^t_i, \end{aligned} \quad (5)$$

where $\mathbf{W}$, $\mathbf{V}$, and $\mathbf{b}$ are trainable parameters. $\boldsymbol{\xi}^t_i$ is the concatenation of adjacent vectors of a context window. The window size in our model is 2 and actually plays a role as a character bigram, which has been shown to be useful for representing characters in sequence labeling tasks (Chen et al., 2015a; Zhang and Yang, 2018). $\boldsymbol{\chi}^t_i$ is the concatenation of the global information vector $\hat{\mathbf{g}}^t$ and the $e \rightarrow c$ aggregation result $\hat{\mathbf{c}}^t_i$. The gates $\boldsymbol{i}^t_i$, $\boldsymbol{f}^t_i$ and $\boldsymbol{l}^t_i$ control information flow from global features to $\mathbf{c}^t_i$, which can make further readjustment of the weights of the lexicon attention ($e \rightarrow c$) to tackle the ambiguities at the subsequent aggregation step.

**Edge Update** To better leverage the interaction among characters, words, and whole sentences, we not only designed a recurrent module for nodes but also for edges and the global relay node (Battaglia et al., 2018). We update the edges as follows:

$$\begin{aligned} \boldsymbol{\chi}^t_{b,e} &= [\hat{\mathbf{e}}^{t-1}_{b,e}; \hat{\mathbf{g}}^{t-1}], \quad \boldsymbol{a} = \{\boldsymbol{i}, \boldsymbol{f}\} \\ \hat{\boldsymbol{a}}^t_{b,e} &= \sigma(\mathbf{W}^a_i \mathbf{e}^{t-1}_{b,e} + \mathbf{V}^a_i \boldsymbol{\chi}^t_{b,e} + \mathbf{b}^a_i) \\ \boldsymbol{u}^t_{b,e} &= tanh(\mathbf{W}_{eu} \mathbf{e}^{t-1}_{b,e} + \mathbf{V}_{eu} \boldsymbol{\chi}^t_{b,e} + \mathbf{b}_{eu}) \\ \boldsymbol{i}^t_{b,e}, \boldsymbol{f}^t_{b,e} &= softmax(\hat{\boldsymbol{i}}^t_{b,e}, \hat{\boldsymbol{f}}^t_{b,e}) \\ \mathbf{e}^t_{b,e} &= \boldsymbol{f}^t_{b,e} \odot \mathbf{e}^{t-1}_{b,e} + \boldsymbol{i}^t_{b,e} \odot \boldsymbol{u}^t_{b,e}, \end{aligned} \quad (6)$$

where $\boldsymbol{\chi}^t_{b,e}$ is the concatenation of $\hat{\mathbf{g}}^t$ and the $c \rightarrow e$ aggregation result $\hat{\mathbf{e}}^t_{b,e}$. Similar to the node update function, $\boldsymbol{i}^t_{b,e}$ and $\boldsymbol{f}^t_{b,e}$ are gates that control information flow from $\mathbf{e}^{t-1}_{b,e}$ and $\hat{\mathbf{g}}^t$ to $\mathbf{e}^t_{b,e}$.

**Global Relay Node Update** In terms of the global state $\mathbf{g}$, recent works (Zhang et al., 2018b; Guo

et al., 2019) have shown the effectiveness of sharing useful messages across contexts. Thus, we also designed an update function for $\mathbf{g}$, with the initialization $\mathbf{g}^0 = \text{average}(\mathbf{C}, \mathbf{E})$. More formally:

$$\begin{aligned}\hat{\boldsymbol{a}}^t &= \sigma(\mathbf{W}_i^a \mathbf{g}^{t-1} + \mathbf{V}_i^a \hat{\mathbf{g}}^{t-1} + \mathbf{b}_i^a),\ \boldsymbol{a} = \{\boldsymbol{i}, \boldsymbol{f}\} \\ \boldsymbol{u}^t &= tanh(\mathbf{W}_{gu} \mathbf{g}^{t-1} + \mathbf{V}_{gu} \hat{\mathbf{g}}^{t-1} + \mathbf{b}_{gu}) \\ \boldsymbol{i}^t, \boldsymbol{f}^t &= softmax(\hat{\boldsymbol{i}}^t, \hat{\boldsymbol{f}}^t) \\ \mathbf{g}^t &= \boldsymbol{f}^t \odot \mathbf{g}^{t-1} + \boldsymbol{i}^t \odot \boldsymbol{u}^t. \end{aligned} \tag{7}$$

### 3.3 Decoding and Training

A standard conditional random field (CRF) is used in the graph message passing process. Given the sequence of final node states $\mathbf{c}_1^T, \mathbf{c}_2^T, \ldots, \mathbf{c}_n^T$, the probability of a label sequence $\hat{y} = \hat{l}_1, \hat{l}_2, \ldots, \hat{l}_n$ can be defined as follows:

$$p(\hat{y}|s) = \frac{exp(\sum_{i=1}^{n} \phi(\hat{l}_{i-1}, \hat{l}_i, \mathbf{c}_i^T))}{\sum_{y' \in \mathrm{Y}(s)} exp(\sum_{i=1}^{n} \phi(l'_{i-1}, l'_i, \mathbf{c}_i^T))}, \tag{8}$$

where, $\mathrm{Y}(s)$ is the set of all arbitrary label sequences. $\phi(l_{i-1}, l_i, \mathbf{c}_i^T) = \mathbf{W}_{(l_{i-1}, l_i)} \mathbf{c}_i^T + \mathbf{b}_{(l_{i-1}, l_i)}$, where $\mathbf{W}_{(l_{i-1}, l_i)}$ and $\mathbf{b}_{(l_{i-1}, l_i)}$ are the weight and bias parameters specific to the labels $l_{i-1}$ and $l_i$.

For training, we minimize the sentence-level negative log-likelihood loss as follows:

$$L = -\sum_{i=1}^{N} log(p(y_i|s_i)). \tag{9}$$

For testing and decoding, we maximized the likelihood to find the optimal sequence $y^*$:

$$y^* = \underset{y \in \mathrm{Y}(s)}{\text{argmax}}\, p(y|s). \tag{10}$$

We used the Viterbi algorithm to calculate the above equations, which can reduce the computational complexity efficiently.

## 4 Experimental Setup

In this section, we describe the datasets across different domains and the baseline methods applied for comparison. We also detail the hyper-parameter configuration of the proposed model. Our codes and datasets can be found at https://github.com/RowitZou/LGN.

| Dataset | Type | Train | Dev. | Test |
|---|---|---|---|---|
| OntoNotes | Sent. | 15.7k | 4.3k | 4.3k |
| | Char. | 491.9k | 200.5k | 208.1k |
| MSRA | Sent. | 46.4k | - | 4.4k |
| | Char. | 2169.9k | - | 172.6k |
| Weibo | Sent. | 1.4k | 0.27k | 0.27k |
| | Char. | 73.8k | 14.5k | 14.8k |
| Resume | Sent. | 3.8k | 0.46k | 0.48k |
| | Char. | 124.1k | 13.9k | 15.1k |

Table 1: Statistics of datasets.

### 4.1 Data

We conducted experiments on four Chinese NER datasets. (1) **OntoNotes 4.0** (Weischedel et al., 2011): OntoNotes is a manually annotated multilingual corpus in the news domain that contains various text annotations, including Chinese named entity labels. Gold-standard segmentation is available. We only use Chinese documents (about 16k sentences) and process the data in the same way as Che et al. (2013). (2) **MSRA** (Levow, 2006): MSRA is also a dataset in the news domain and contains three types of named entities: LOC, PER, and ORG. Chinese word segmentation is available in the training set but not in the test set. (3) **Weibo NER** (Peng and Dredze, 2015): It consists of annotated NER messages drawn from the social media Sina Weibo[2]. The corpus contains PER, ORG, GPE, and LOC for both named entity and nominal mention. (4) **Resume NER** (Zhang and Yang, 2018): It is composed of resumes collected from Sina Finance[3] and is annotated with 8 types of named entities. Both Weibo and Resume datasets do not contain the gold-standard Chinese segmentation. Statistics of the above datasets are detailed in Table 1.

### 4.2 Lexicon

We used the lexicon over automatically segmented Chinese Giga-Word [4], obtaining 704.4k words in the final lexicon. The embeddings of lexicon words were pre-trained using word2vec (Mikolov et al., 2013) and fine-tuned during training. According to the lexicon statistics, the number of single-character, two-character and three-character words are 5.7k, 291.5k, 278.1k, respectively. It covers 31.2% of the named entities in the four data sets, which means most of the lexicon words are not named entities. For a

[2]https://www.weibo.com
[3]https://finance.sina.com.cn/stock/
[4]https://catalog.ldc.upenn.edu/LDC2011T13

fair comparison, we used such a general lexicon instead of a professional named entity lexicon in our experiments and we still obtained competitive results. Empirically, a high-quality lexicon could lead to further performance improvements.

Character embeddings are pre-trained on Chinese Giga-Word using word2vec and fine-tuned at model training. Both the pre-trained character and lexicon word embeddings are released by Zhang and Yang (2018)[5].

### 4.3 Comparison Methods

We applied the character-level and word-level methods as baselines for comparison, which incorporate the bichar, softword, and lexicon features. We also compared several state-of-the-art methods on the four datasets to verify the effectiveness of our method. We used the BMES tagging scheme for both character-level and word-level NER tagging.

**Character-level methods:** These methods are based on character sequences. We applied the bi-directional LSTM (Hochreiter and Schmidhuber, 1997) and CNN (Kim, 2014) as classic baseline methods.

**Character-level methods + bichar + softword:** Character bigrams are useful for capturing adjacent features and representing characters. We concatenated bigram embeddings with character embeddings to better leverage the bigram information. In addition, we added the segmentation information by incorporating segmentation label embeddings into the character representation. The BMES scheme is used for representing the word segmentation (Xue and Shen, 2003).

**Word-level methods:** For the datasets with gold segmentation, we directly employed word-level NER methods to evaluate the performance, which are denoted as **Gold seg**. Otherwise, we first used open source segmentation toolkit[6] to automatically segment the datasets. Then word-level NER methods are applied, which are denoted as **Auto seg**. The bi-directional LSTM and CNN are also applied as baselines.

**Word-level methods + char + bichar:** For characters in the subsequence $w_{b,e}$, we first used a bi-directional LSTM to learn their hidden states and bigram states. We then augmented the word-level methods with the character-level features.

[5]https://github.com/jiesutd/LatticeLSTM

[6]https://github.com/lancopku/PKUSeg-python

| Input | Models | P | R | F1 |
|---|---|---|---|---|
| Gold seg. | Yang et al. (2016) | 65.59 | 71.84 | 68.57 |
| | Yang et al. (2016)*† | 72.98 | **80.15** | **76.40** |
| | Che et al. (2013)* | 77.71 | 72.51 | 75.02 |
| | Wang et al. (2013)* | 76.43 | 72.32 | 74.32 |
| | Word-level LSTM | 76.66 | 63.60 | 69.52 |
| | +char+bichar | **78.62** | 73.13 | 75.77 |
| | Word-level CNN | 66.84 | 62.99 | 64.86 |
| | +char+bichar | 68.22 | 72.37 | 70.24 |
| Auto seg. | Word-level LSTM | 72.84 | 59.72 | 65.63 |
| | +char+bichar | 73.36 | 70.12 | 71.70 |
| | Word-level CNN | 54.62 | 55.20 | 54.91 |
| | +char+bichar | 64.69 | 65.09 | 64.89 |
| No seg. | Char-level LSTM | 68.79 | 60.35 | 64.30 |
| | +bichar+softword | 74.36 | 69.43 | 71.89 |
| | Char-level CNN | 56.78 | 60.99 | 58.81 |
| | +bichar+softword | 59.60 | 65.14 | 62.25 |
| | Lattice LSTM | **76.35** | 71.56 | 73.88 |
| | LGN | 76.13 | **73.68** | **74.89** |

Table 2: Main results on OntoNotes.

**Lattice LSTM:** Lattice LSTM (Zhang and Yang, 2018) incorporates word information into character-level recurrent units, which can avoid segmentation errors. This method achieved state-of-the-art performance on the four datasets.

### 4.4 Hyper-parameter Settings

We used the Adam (Kingma and Ba, 2014) as the optimizer, with a learning rate of 2e-5 for large datasets like Ontonotes and MSRA, while a rate of 2e-4 for small datasets Weibo and Resume. A densely connected structure (Huang et al., 2017) was applied, which composites all hidden states from previous update steps as final inputs for aggregation modules at step $t$. To further reduce overfitting, we employed the Dropout (Srivastava et al., 2014) with a rate of 0.5 for embeddings and a rate of 0.2 for aggregation module outputs. The embedding size and state size were both set to 50. The head number of multi-head attention was 10. The head dimension was set to 10 for small datasets like Weibo and Resume, while the head dimension was 20 for Ontonotes and MSRA. Step number $T$ was selected among $\{1, 2, 3, 4, 5, 6\}$, which is detailed analyzed in § 5.2. The standard Precision (P), Recall (R), and F1-score (F1) were used as evaluation metrics.

## 5 Results and Discussion

In this section, we demonstrate the main results of LGN for the Chinese NER task across different domains. The model achieving best results on the development set was chosen for the final evaluation on the test set. We also probe the

| Models | P | R | F1 |
|---|---|---|---|
| Chen et al. (2006) | 91.22 | 81.71 | 86.20 |
| Zhang et al. (2006)* | 92.20 | 90.18 | 91.18 |
| Zhou et al. (2013) | 91.86 | 88.75 | 90.28 |
| Lu et al. (2016) | - | - | 87.94 |
| Dong et al. (2016) | 91.28 | 90.62 | 90.95 |
| Word-level LSTM | 90.57 | 83.06 | 86.65 |
| +char+bichar | 91.05 | 89.53 | 90.28 |
| Char-level LSTM | 90.74 | 86.96 | 88.81 |
| +bichar+softword | 92.97 | 90.80 | 91.87 |
| Lattice LSTM | 94.18 | 92.20 | 93.18 |
| LGN | **94.19** | **92.73** | **93.46** |

Table 3: Main results on MSRA.

| Models | NE | NM | All |
|---|---|---|---|
| Peng and Dredze (2015) | 51.96 | 61.05 | 56.05 |
| Peng and Dredze (2015)* | 55.28 | 62.97 | 58.99 |
| He and Sun (2016) | 50.60 | 59.32 | 54.82 |
| He and Sun (2017)* | 54.50 | 62.17 | 58.23 |
| Word-level LSTM | 36.02 | 59.38 | 47.33 |
| +char+bichar | 43.40 | 60.30 | 52.33 |
| Char-level LSTM | 46.11 | 55.29 | 52.77 |
| +bichar+softword | 50.55 | 60.11 | 56.75 |
| Lattice LSTM | 53.04 | 62.25 | 58.79 |
| LGN | **55.34** | **64.98** | **60.21** |

Table 4: Main results on Weibo.

| Models | P | R | F1 |
|---|---|---|---|
| Word-level LSTM | 93.72 | 93.44 | 93.58 |
| +char+bichar | 94.07 | 94.42 | 94.24 |
| Char-level LSTM | 93.66 | 93.31 | 93.48 |
| +bichar+softword | 94.53 | 94.29 | 94.41 |
| Lattice LSTM | 94.81 | 94.11 | 94.46 |
| LGN | **95.28** | **95.46** | **95.37** |

Table 5: Main results on Resume.

effectiveness and interpretability of LGN by explanatory experiments.

### 5.1 Main Results

**OntoNotes** Table 2[7] shows the results of word-level and character-level methods on OntoNotes with various settings. In the gold or automatic segmentation settings, the char and bichar features boost the performance of word-level methods. In particular, with the gold-standard segmentation, these methods are able to achieve competitive state-of-the-art results on the dataset (Che et al., 2013; Wang et al., 2013). However, the gold-standard segmentation is not always available. On the other hand, the automatic segmentation may induce word segmentation errors and result in a loss of performance for the downstream NER task. A feasible solution is applying character-level methods to avoid the need for word segmentation. Our proposed LGN is a character-level model based on graphic structure. It outperforms lattice LSTM by 1.01% in F1 score and leads to a 3.00% increment of F1 score over the LSTM with bichar and softword features. The LGN also significantly outperforms the word-level models with automatic segmentation.

**MSRA/Weibo/Resume** Results on the MSRA, Weibo, and Resume datasets are shown in Table 3, 4, and 5, respectively. Gold-standard segmentation is not available for the Weibo and Resume datasets and the test set of MSRA. The best classic methods leverage rich handcrafted features (Chen et al., 2006; Zhang et al., 2006; Zhou et al., 2013), embedding features (Lu et al., 2016; Peng and Dredze, 2015), radical features (Dong et al., 2016), cross-domain, and semi-supervised data (He and Sun, 2017) for Chinese NER. Compared with the existing methods and the word-level and character-level methods, our LGN model gives the best results by a large margin. Moreover, different from the lattice LSTM, which also leverages lexicon features, our LGN model integrates lexicon information into the graph neural network in a more effective fashion. As a result, it outperforms the lattice LSTM on all three datasets.

### 5.2 Steps of Message Passing

To investigate the influence of step number $T$ during the update process, we analyzed the performance of LGN under different step numbers. Figure 4 illustrates the variation of F1 score on the development sets[8] as the step number increases. We used **D-F1** to represent the F1 scores at different steps minus the best results.

The results indicate that the number of update steps is crucial to the performance of LGN, which peaks when $T \geq 3$ on all four datasets. The F1 score decreases 1.20% on average against the best results when the step number is less than 3. In particular, the F1 score of the OntoNotes and Weibo datasets even suffered a serious reduction around 1.5% and 1.8%, respectively. After several rounds of updates, the model gives steady and competitive results and reveals that LGN benefits from the update process. Empirically, at each

[7] In Table 2, 3, 4 and 5, the models with * use external labeled data for semi-supervised learning. Those with † also use discrete features.

[8] Since the MSRA dataset does not have the development set, we used the test set instead.

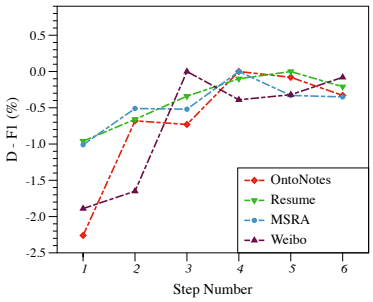


Figure 4: F1 variation under different update steps on the development sets. D-F1 represents the F1 scores at different steps minus the best results.

| Models | Onto-Notes | MSRA | Weibo | Resume |
|---|---|---|---|---|
| LGN | 71.96 | 93.46 | 62.42 | 94.43 |
| -global | 71.26 | 92.99 | 62.30 | 94.31 |
| -edge /lexicon | 65.88 | 89.63 | 59.19 | 94.05 |
| -edge-global | 65.34 | 89.47 | 58.62 | 94.09 |
| -bidirection | 67.52 | 90.98 | 59.67 | 94.23 |
| -crf | 66.37 | 91.13 | 57.73 | 92.70 |
| Lattice LSTM | 71.62 | 93.18 | 61.64 | 93.64 |
| -bidirection | 66.63 | 90.72 | 58.75 | 93.21 |
| -crf | 61.74 | 85.38 | 55.71 | 92.31 |

Table 6: An ablation study of LGN. F1 scores were evaluated on the development sets.

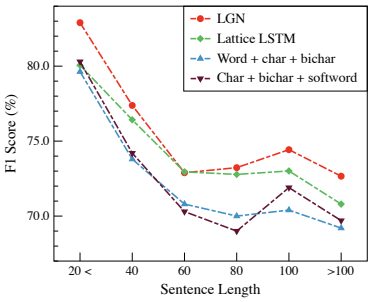


Figure 5: F1 score against sentence length on the OntoNotes dateset.

update step, graph nodes aggregate information from their neighbors and incrementally gain more information from further reaches of the graph as the process iterates (Hamilton et al., 2017). In the LGN model, more valuable information can be captured through the recursive aggregation.

### 5.3 Ablation Experiments

To study the contribution of each component in LGN, we conducted ablation experiments on the four datasets and display the results in Table 6. The results show that the model's performance is degraded if the global relay node is removed, indicating that global connections are useful in the graph structure. We also find that lexicons play an important role in character-level NER. In particular, the performance of the OntoNotes, MSRA and Weibo datasets are seriously hurt by over 3.0% without lexicons. Moreover, missing both edges and the global node will cause a further performance loss.

To better illustrate the advantage of our model, we remove the CRF decoding layer and simplify the structure to a non-bidirectional version on both LGN and the lattice LSTM model. The results show that, with a single direction structure, the LGN achieves a higher F1 score by 0.77% on average than the lattice LSTM. In addition, the two models have an obvious performance gap when they get rid of the CRF layer. The F1 score of LGN decreases by 3.59% on average on the four datasets without CRF. In contrast, the lattice LSTM decreases by 6.24%. It manifests the LGN has stronger ability to model sentences.

### 5.4 Performance Against Sentence Length

Figure 5 shows the performance of LGN and several baseline models on the OntoNotes dataset. We split the dataset into six parts according to the sentence length. The lattice is a strong baseline that outperforms the word+char+bichar and char+bichar+softword models over different sentence lengths. However, the lattice accuracy decreases significantly as the sentence length increases. In contrast, the LGN not only gives higher results over short sentences, but also shows its effectiveness and robustness when the sentence length is more than 80 characters. It gives a higher F1 score in most cases compared to the baselines, which indicates that global sentence semantics and long-range dependency can be better captured under the graph structure.

### 5.5 Case Study

Table 7 illustrates an example that probes the ability of LGN to tackle the word ambiguity problems. The lattice LSTM ignores the sentence context and wrongly identifies "印 度(India)". Removing the global relay node, LGN also makes the same mistake, which indicates that global connections are indispensable and can capture high-level information to help LGN better understand the sentence context. In contrast, with the global relay node, the LGN can correctly identify the entity boundary, even though the

| | |
|---|---|
| Sentence | 印度河流经巴基斯坦<br>The Indus River flows through Pakistan. |
| Gold seg | 印度河 流经 巴基斯坦<br>The Indus River, flow through, Pakistan |
| Lexicon words | 印度 河流 印度河 流经 巴基斯坦<br>India, river, The Indus River, flow through, Pakistan |
| Lattice LSTM | B E (GPE) O O O B M M E (GPE)<br>印度 (GPE) 河流经 巴基斯坦 (GPE)<br>India (GPE) ... Pakistan (GPE). |
| LGN<br>-global | B E (GPE) O O O B M M E (GPE)<br>印度 (GPE) 河流经 巴基斯坦 (GPE)<br>India (GPE) ... Pakistan (GPE). |
| LGN<br>(one step) | B M E (GPE) O O B M M E (GPE)<br>印度河 (GPE) 流经 巴基斯坦 (GPE)<br>The Indus River (GPE) flows through Pakistan (GPE). |
| LGN | B M E (LOC) O O B M M E (GPE)<br>印度河 (LOC) 流经 巴基斯坦 (GPE)<br>The Indus River (LOC) flows through Pakistan (GPE). |

Table 7: An example with overlapping ambiguity. Contents with red and blue colors represent incorrect and correct entities, respectively.

graph composition states are updated for only one step. However, it gives an incorrect class of the entity "印度河(The Indus River)", which is a location entity but not a GPE (Geo-Political Entity). Because of the multi-step graph message passing process, the LGN is able to fuse the context information and finally detects the correct location entity in success.

## 6 Conclusion

In this work, we investigated a GNN-based approach to alleviate the word ambiguity in Chinese NER. Lexicons are used to construct the graph and provide word-level features. The LGN enables interactions among different sentence compositions and can capture non-sequential dependencies between characters based on the global sentence semantics. As a result, it shows improved performance significantly on multiple datasets in different domains. The explanatory experiments also illustrate the effectiveness and interpretability of our proposed model.

## Acknowledgments

The authors wish to thank the anonymous reviewers for their helpful comments. This work was partially funded by China National Key R&D Program (No.2018YFC0831105, 2018YFB1005104, 2017YFB1002104), National Natural Science Foundation of China (No. 61976056, 61532011, 61751201), Shanghai Municipal Science and Technology Major Project (No.2018SHZDZX01), STCSM (No.16JC1420401, 17JC1420200).

## References

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

Razvan C Bunescu and Raymond J Mooney. 2005. A shortest path dependency kernel for relation extraction. In *HLT—EMNLP*.

Wanxiang Che, Mengqiu Wang, Christopher D Manning, and Ting Liu. 2013. Named entity recognition with bilingual constraints. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 52–62.

Aitao Chen, Fuchun Peng, Roy Shan, and Gordon Sun. 2006. Chinese named entity recognition with conditional probabilistic models. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 173–176.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Pengfei Liu, and Xuanjing Huang. 2015a. Long short-term memory neural networks for chinese word segmentation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1197–1206.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015b. Event extraction via dynamic multi-pooling convolutional neural networks. In *ACL—IJCNLP*, volume 1, pages 167–176.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dennis Diefenbach, Vanessa Lopez, Kamal Singh, and Pierre Maret. 2018. Core techniques of question answering systems over knowledge bases: a survey. *KAIS*, 55(3):529–569.

Chuanhai Dong, Jiajun Zhang, Chengqing Zong, Masanori Hattori, and Hui Di. 2016. Character-based lstm-crf with radical-level features for chinese named entity recognition. In *Natural Language Understanding and Intelligent Applications*, pages 239–250. Springer.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. *arXiv preprint arXiv:1905.03197*.

Huanzhong Duan and Yan Zheng. 2011. A study on features of the crfs-based chinese named entity recognition. *IJAI*, 3(2):287–294.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1263–1272. JMLR. org.

Tao Gui, Ruotian Ma, Qi Zhang, Lujun Zhao, Yu-Gang Jiang, and Xuanjing Huang. 2019. Cnn-based chinese ner with lexicon rethinking. In *IJCAI*.

Qipeng Guo, Xipeng Qiu, Pengfei Liu, Yunfan Shao, Xiangyang Xue, and Zheng Zhang. 2019. Star-transformer. *arXiv preprint arXiv:1902.09113*.

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034.

Hangfeng He and Xu Sun. 2016. F-score driven max margin neural network for named entity recognition in chinese social media. *arXiv preprint arXiv:1611.04234*.

Hangfeng He and Xu Sun. 2017. A unified model for cross-domain and semi-supervised named entity recognition in chinese social media. In *AAAI*.

Jingzhou He and Houfeng Wang. 2008. Chinese named entity recognition and word segmentation based on character. In *Proceedings of the Sixth SIGHAN Workshop on Chinese Language Processing*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

John Boaz Lee, Ryan Rossi, and Xiangnan Kong. 2018. Graph classification using structural attention. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1666–1674. ACM.

Gina-Anne Levow. 2006. The third international chinese language processing bakeoff: Word segmentation and named entity recognition. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 108–117.

Haibo Li, Masato Hagiwara, Qi Li, and Heng Ji. 2014. Comparison of the impact of word segmentation on name tagging for chinese and japanese. In *LREC*, pages 2532–2536.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. 2015. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*.

Zhangxun Liu, Conghui Zhu, and Tiejun Zhao. 2010. Chinese named entity recognition with a sequence labeling approach: based on characters, or based on words? In *International Conference on Intelligent Computing*, pages 634–640. Springer.

Yanan Lu, Yue Zhang, and Dong-Hong Ji. 2016. Multi-prototype chinese character embedding. In *LREC*.

Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie. 2015. Joint entity recognition and disambiguation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 879–888.

Guojie Ma, Xingshan Li, and Keith Rayner. 2014. Word segmentation of overlapping ambiguous strings during chinese reading. *Journal of Experimental Psychology: Human Perception and Performance*, 40(3):1046.

Luisa Mich, Roberto Garigliano, et al. 2000. Ambiguity measures in requirement engineering. In *International Conference on Software Theory and Practice. ICS*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. *CoNLL-2014*, page 78.

Hao Peng, Jianxin Li, Yu He, Yaopeng Liu, Mengjiao Bao, Lihong Wang, Yangqiu Song, and Qiang Yang. 2018. Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 1063–1072. International World Wide Web Conferences Steering Committee.

Nanyun Peng and Mark Dredze. 2015. Named entity recognition for chinese social media with jointly trained embeddings. In *EMNLP*.

Charles A Perfetti and Li Hai Tan. 1999. The constituency model of chinese word identification.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. 2019. Ordered neurons: Integrating tree structures into recurrent neural networks. *Proceedings of ICLR*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Mengqiu Wang, Wanxiang Che, and Christopher D Manning. 2013. Effective bilingual constraints for semi-supervised learning of named entity recognizers. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*.

Ralph Weischedel, Sameer Pradhan, Lance Ramshaw, Martha Palmer, Nianwen Xue, Mitchell Marcus, Ann Taylor, Craig Greenberg, Eduard Hovy, Robert Belvin, et al. 2011. Ontonotes release 4.0. *LDC2011T03, Philadelphia, Penn.: Linguistic Data Consortium*.

Nianwen Xue and Libin Shen. 2003. Chinese word segmentation as lmr tagging. In *Proceedings of the second SIGHAN workshop on Chinese language processing-Volume 17*, pages 176–179. Association for Computational Linguistics.

Jie Yang, Zhiyang Teng, Meishan Zhang, and Yue Zhang. 2016. Combining discrete and neural features for sequence labeling. In *CICLing*. Springer.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2018. Graph convolutional networks for text classification. *arXiv preprint arXiv:1809.05679*.

Miao-Hsuan Yen, Ralph Radach, Ovid J-L Tzeng, and Jie-Li Tsai. 2012. Usage of statistical cues for word boundary in reading chinese sentences. *Reading and writing*, 25(5):1007–1029.

Biyin Zhang and Danling Peng. 1992. Decomposed storage in the chinese lexicon. In *Advances in psychology*, volume 90, pages 131–149. Elsevier.

Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit-Yan Yeung. 2018a. Gaan: Gated attention networks for learning on large and spatiotemporal graphs. *arXiv preprint arXiv:1803.07294*.

Suxiang Zhang, Ying Qin, Juan Wen, and Xiaojie Wang. 2006. Word segmentation and named entity recognition for sighan bakeoff3. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 158–161.

Yue Zhang, Qi Liu, and Linfeng Song. 2018b. Sentence-state lstm for text representation. *arXiv preprint arXiv:1805.02474*.

Yue Zhang and Jie Yang. 2018. Chinese ner using lattice lstm. *arXiv preprint arXiv:1805.02023*.

Yuhao Zhang, Peng Qi, and Christopher D Manning. 2018c. Graph convolution over pruned dependency trees improves relation extraction. *arXiv preprint arXiv:1809.10185*.

Junsheng Zhou, Weiguang Qu, and Fen Zhang. 2013. Chinese named entity recognition via joint identification and categorization. *Chinese journal of electronics*, 22(2):225–230.

Xiaolin Zhou and William Marslen-Wilson. 1994. Words, morphemes and syllables in the chinese mental lexicon. *Language and Cognitive Processes*, 9(3):393–422.

# CM-Net: A Novel Collaborative Memory Network for Spoken Language Understanding

**Yijin Liu[1]*, Fandong Meng[2], Jinchao Zhang[2], Jie Zhou[2], Yufeng Chen[1] and Jinan Xu[1]†**
[1]Beijing Jiaotong University, China
[2]Pattern Recognition Center, WeChat AI, Tencent Inc, China
adaxry@gmail.com
{fandongmeng, dayerzhang, withtomzhou}@tencent.com
{chenyf, jaxu}@bjtu.edu.cn

## Abstract

Spoken Language Understanding (SLU) mainly involves two tasks, intent detection and slot filling, which are generally modeled jointly in existing works. However, most existing models fail to fully utilize co-occurrence relations between slots and intents, which restricts their potential performance. To address this issue, in this paper we propose a novel Collaborative Memory Network (CM-Net) based on the well-designed block, named CM-block. The CM-block firstly captures slot-specific and intent-specific features from memories in a collaborative manner, and then uses these enriched features to enhance local context representations, based on which the sequential information flow leads to more specific (slot and intent) global utterance representations. Through stacking multiple CM-blocks, our CM-Net is able to alternately perform information exchange among specific memories, local contexts and the global utterance, and thus incrementally enriches each other. We evaluate the CM-Net on two standard benchmarks (ATIS and SNIPS) and a self-collected corpus (CAIS). Experimental results show that the CM-Net achieves the state-of-the-art results on the ATIS and SNIPS in most of criteria, and significantly outperforms the baseline models on the CAIS. Additionally, we make the CAIS dataset publicly available for the research community [1].

## 1 Introduction

Spoken Language Understanding (SLU) is a core component in dialogue systems. It typically aims to identify the intent and semantic constituents for a given utterance, which are referred as intent detection and slot filling, respectively. Past years have witnessed rapid developments in diverse deep learning models (Haffner et al., 2003; Sarikaya et al., 2011) for SLU. To take full advantage of supervised signals of slots and intents, and share knowledge between them, most of existing works apply joint models that mainly based on CNNs (Xu and Sarikaya, 2013; Gupta et al., 2019), RNNs (Guo et al., 2014a; Liu and Lane, 2016), and asynchronous bi-model (Wang et al., 2018). Generally, these joint models encode words convolutionally or sequentially, and then aggregate hidden states into a utterance-level representation for the intent prediction, without interactions between representations of slots and intents.

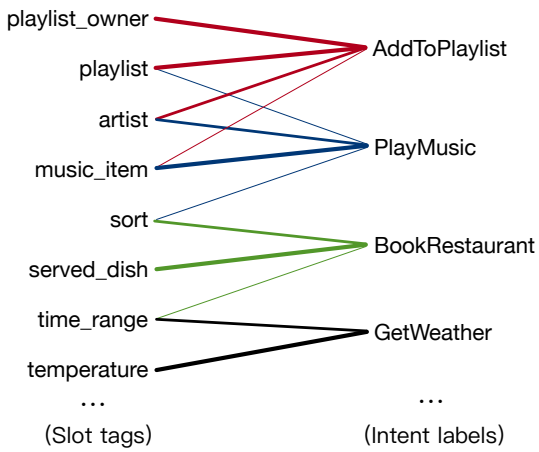


Figure 1: Statistical association of slot tags (on the left) and intent labels (on the right) in the SNIPS, where colors indicate different intents and thicknesses of lines indicate proportions.

Intuitively, slots and intents from similar fields tend to occur simultaneously, which can be observed from Figure 1 and Table 1. Therefore, it is beneficial to generate the representations of slots and intents with the guidance from each other. Some works explore enhancing the slot filling task

* This work was done when Yijin Liu was interning at Pattern Recognition Center, WeChat AI, Tencent Inc, China
† Jinan Xu is the corresponding author of the paper.
[1]Code is available at: https://github.com/Adaxry/CM-Net.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1051–1060, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| # | Utterance | Slot tag | Intent |
|---|---|---|---|
| 1 | play Roy Orbison tunes now | artist | PlayMusic |
| 2 | add this Roy Orbison song onto Women of Comedy | artist | AddToPlaylist |
| 3 | book a spot for seven at a bar with chicken french | served_dish | BookRestaurant |
| 4 | book french food for me and angeline at a restaurant | cuisine | BookRestaurant |

Table 1: Examples in SNIPS with annotations of intent label for the utterance and slot tags for partial words.

unidirectionally with the guidance from intent representations via gating mechanisms (Goo et al., 2018; Li et al., 2018), while the predictions of intents lack the guidance from slots. Moreover, the capsule network with dynamic routing algorithms (Zhang et al., 2018a) is proposed to perform interactions in both directions. However, there are still two limitations in this model. The one is that the information flows from words to slots, slots to intents and intents to words in a pipeline manner, which is to some extent limited in capturing complicated correlations among words, slots and intents. The other is that the local context information which has been shown highly useful for the slot filling (Mesnil et al., 2014), is not explicitly modeled.

In this paper, we try to address these issues, and thus propose a novel **C**ollaborative **M**emory **N**etwork, named CM-Net. The main idea is to directly capture semantic relationships among words, slots and intents, which is conducted simultaneously at each word position in a collaborative manner. Specifically, we alternately perform information exchange among the task-specific features referred from memories, local context representations and global sequential information via the well-designed block, named CM-block, which consists of three computational components:

- **Deliberate Attention**: Obtaining slot-specific and intent-specific representations from memories in a collaborative manner.
- **Local Calculation**: Updating local context representations with the guidances of the referred slot and intent representations in the previous *Deliberate Attention*.
- **Global Recurrence**: Generating specific (slot and intent) global sequential representations based on local context representations from the previous *Local Calculation*.

Above components in each CM-block are conducted consecutively, which are responsible for encoding information from different perspectives. Finally, multiple CM-blocks are stacked together, and construct our CM-Net.

We firstly conduct experiments on two popular benchmarks, SNIPS (Coucke et al., 2018) and ATIS (Hemphill et al., 1990; Tur et al., 2010). Experimental results show that the CM-Net achieves the state-of-the-art results in 3 of 4 criteria (*e.g.,* intent detection accuracy on ATIS) on both benchmarks. Additionally, trials on our self-collected dataset, named CAIS, demonstrate the effectiveness and generalizability of the CM-Net.

Our main contributions are as follows:

- We propose a novel CM-Net for SLU, which explicitly captures semantic correlations among words, slots and intents in a collaborative manner, and incrementally enriches the specific features, local context representations and global sequential representations through stacked CM-blocks.
- Our CM-Net achieves the state-of-the-art results on two major SLU benchmarks (ATIS and SNIPS) in most of criteria.
- We contribute a new corpus CAIS with manual annotations of slot tags and intent labels to the research community.

## 2 Background

In principle, the slot filling is treated as a sequence labeling task, and the intent detection is a classification problem. Formally, given an utterance $X = \{x_1, x_2, \cdots, x_N\}$ with $N$ words and its corresponding slot tags $Y^{slot} = \{y_1, y_2, \cdots, y_N\}$, the slot filling task aims to learn a parameterized mapping function $f_\theta : X \rightarrow Y$ from input words to slot tags. For the intent detection, it is designed to predict the intent label $\hat{y}^{int}$ for the entire utterance $X$ from the predefined label set $S^{int}$.

Typically, the input utterance is firstly encoded into a sequence of distributed representations $\mathbf{X} =$

$\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$ by character-aware and pre-trained word embeddings. Afterwards, the following bidirectional RNNs are applied to encode the embeddings $\mathbf{X}$ into context-sensitive representations $\mathbf{H} = \{\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_N\}$. An external CRF (Lafferty et al., 2001) layer is widely utilized to calculate conditional probabilities of slot tags:

$$p(\mathbf{y}^{slot}|\mathbf{H}) = \frac{e^{F(\mathbf{H}, \mathbf{y}^{slot})}}{\sum_{\widetilde{\mathbf{y}}^{slot} \in \mathbf{Y}_x} e^{F(\mathbf{H}, \widetilde{\mathbf{y}}^{slot})}} \quad (1)$$

Here $\mathbf{Y}_x$ is the set of all possible sequences of tags, and $F(\cdot)$ is the score function calculated by:

$$F(\mathbf{h}, \mathbf{y}) = \sum_{i=1}^{N} \mathbf{A}_{y_i, y_{i+1}} + \sum_{i=1}^{N} \mathbf{P}_{i, y_i} \quad (2)$$

where $\mathbf{A}$ is the transition matrix that $\mathbf{A}_{i,j}$ indicates the score of a transition from $i$ to $j$, and $\mathbf{P}$ is the score matrix output by RNNs. $P_{i,j}$ indicates the score of the $j^{th}$ tag of the $i^{th}$ word in a sentence (Lample et al., 2016).

When testing, the Viterbi algorithm (Forney, 1973) is used to search the sequence of slot tags with maximum score:

$$\hat{\mathbf{y}}^{slot} = \arg\max_{\widetilde{\mathbf{y}}^{slot} \in \mathbf{Y}_x} F(\mathbf{H}, \widetilde{\mathbf{y}}^{slot}) \quad (3)$$

As to the prediction of intent, the word-level hidden states $\mathbf{H}$ are firstly summarized into a utterance-level representation $\mathbf{v}^{int}$ via mean pooling (or max pooling or self-attention, *etc.*):

$$\mathbf{v}^{int} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{h}_t \quad (4)$$

The most probable intent label $\hat{y}^{int}$ is predicted by softmax normalization over the intent label set:

$$\begin{aligned} \hat{y}^{int} &= \arg\max_{\widetilde{y} \in S^{int}} P(\widetilde{y}|\mathbf{v}^{int}) \\ P(\widetilde{y} = j|\mathbf{v}^{int}) &= softmax(\mathbf{v}^{int})[j] \end{aligned} \quad (5)$$

Generally, both tasks are trained jointly to minimize the sum of cross entropy from each individual task. Formally, the loss function of the join model is computed as follows:

$$\begin{aligned} L &= (1-\lambda) \cdot L^{slot} + \lambda \cdot L^{int} \\ L^{int} &= -\sum_{i=1}^{|S^{int}|} \hat{y}_i^{int} log(y_i^{int}) \\ L^{slot} &= -\sum_{j=1}^{N} \sum_{i=1}^{|S^{slot}|} \hat{y}_{i,j}^{slot} log(y_{i,j}^{slot}) \end{aligned} \quad (6)$$

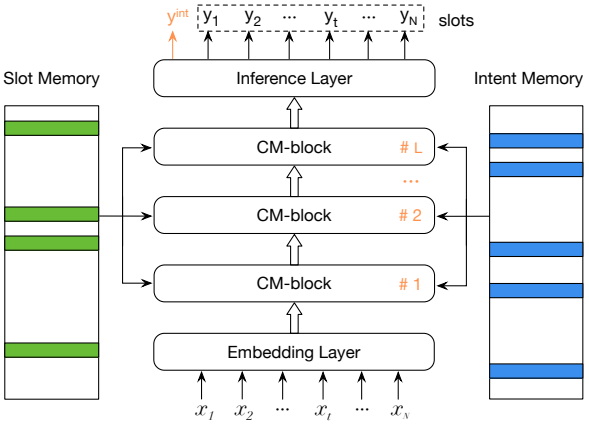


Figure 2: Overview of our proposed CM-Net. The input utterance is firstly encoded with the Embedding Layer (bottom), and then is transformed by multiple CM-blocks with the assistance of both slot and intent memories (on both sides). Finally we make predictions of slots and the intent in the Inference Layer (top).

where $y_i^{int}$ and $y_{i,j}^{slot}$ are golden labels, and $\lambda$ is hyperparameter, and $|S^{int}|$ is the size of intent label set, and similarly for $|S^{slot}|$ .

## 3 CM-Net

### 3.1 Overview

In this section, we start with a brief overview of our CM-Net and then proceed to introduce each module. As shown in Figure 2, the input utterance is firstly encoded with the Embedding Layer, and then is transformed by multiple CM-blocks with the assistance of slot and intent memories, and finally make predictions in the Inference Layer.

### 3.2 Embedding Layers

**Pre-trained Word Embedding** The pre-trained word embeddings has been indicated as a de-facto standard of neural network architectures for various NLP tasks. We adapt the cased, 300d Glove[2] (Pennington et al., 2014) to initialize word embeddings, and keep them frozen.

**Character-aware Word Embedding** It has been demonstrated that character level information (*e.g.* capitalization and prefix) (Collobert et al., 2011) is crucial for sequence labeling. We use one layer of CNN followed by max pooling to generate character-aware word embeddings.

### 3.3 CM-block

The CM-block is the core module of our CM-Net, which is designed with three computational com-

[2]https://nlp.stanford.edu/projects/glove/

ponents: *Deliberate Attention*, *Local Calculation* and *Global Recurrence* respectively.

### Deliberate Attention

To fully model semantic relations between slots and intents, we build the slot memory $\mathbf{M^{slot}}$ and intent memory $\mathbf{M^{int}}$, and further devise a collaborative retrieval approach. For the slot memory, it keeps $|S^{slot}|$ slot cells which are randomly initialized and updated as model parameters. Similarly for the intent memory. At each word position, we take the hidden state $\mathbf{h}_t$ as query, and obtain slot feature $\mathbf{h}_t^{slot}$ and intent feature $\mathbf{h}_t^{int}$ from both memories by the deliberate attention mechanism, which will be illustrated in the following.

Specifically for the slot feature $\mathbf{h}_t^{slot}$, we firstly get a rough intent representation $\widetilde{\mathbf{h}}_t^{int}$ by the word-aware attention with hidden state $\mathbf{h}_t$ over the intent memory $\mathbf{M^{int}}$, and then obtain the final slot feature $\mathbf{h}_t^{slot}$ by the intent-aware attention over the slot memory $\mathbf{M^{slot}}$ with the intent-enhanced representation $[\mathbf{h}_t; \widetilde{\mathbf{h}}_t^{int}]$. Formally, the above-mentioned procedures are computed as follows:

$$\begin{aligned} \widetilde{\mathbf{h}}_t^{int} &= ATT(\mathbf{h_t}, \mathbf{M^{int}}) \\ \mathbf{h}_t^{slot} &= ATT([\mathbf{h_t}; \widetilde{\mathbf{h}}_t^{int}], \mathbf{M^{slot}}) \end{aligned} \tag{7}$$

where $ATT(\cdot)$ is the query function calculated by the weighted sum of all cells $\mathbf{m}_i^x$ in memory $\mathbf{M}^x$ $(x \in \{slot, int\})$ :

$$\begin{aligned} ATT(\mathbf{h_t}, \mathbf{M}^x) &= \sum_i \alpha_i \mathbf{m_i^x} \\ \alpha_i &= \frac{\exp(\mathbf{u}^\top s_i)}{\sum_j \exp(\mathbf{u}^\top s_j)} \\ s_i &= \mathbf{h}_t^\top \mathbf{W} \mathbf{m_i^x} \end{aligned} \tag{8}$$

Here $\mathbf{u}$ and $\mathbf{W}$ are model parameters. We name the above calculations of two-round attentions (Equation 7) as "deliberate attention".

The intent representation $\mathbf{h}_t^{int}$ is computed by the deliberate attention as well:

$$\begin{aligned} \widetilde{\mathbf{h}}_t^{slot} &= ATT(\mathbf{h_t}, \mathbf{M^{slot}}) \\ \mathbf{h}_t^{int} &= ATT([\mathbf{h_t}; \widetilde{\mathbf{h}}_t^{slot}], \mathbf{M^{int}}) \end{aligned} \tag{9}$$

These two deliberate attentions are conducted simultaneously at each word position in such collaborative manner, which guarantees adequate knowledge diffusions between slots and intents. The retrieved slot features $\mathbf{H}_t^{slot}$ and intent features $\mathbf{H}_t^{int}$ are utilized to provide guidances for the next local calculation layer.

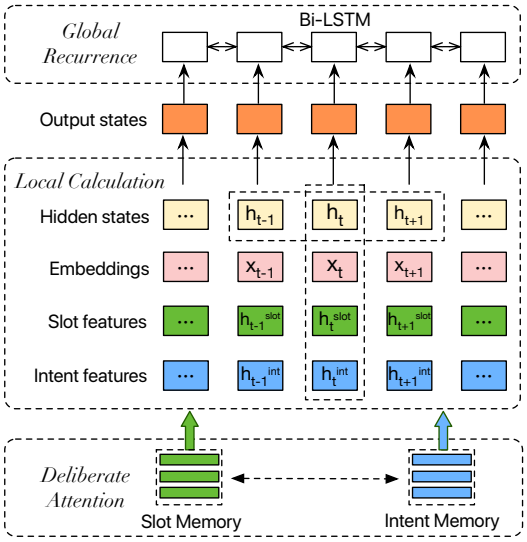


Figure 3: The internal structure of our CM-Block, which is composed of *deliberate attention*, *local calculation* and *global recurrent* respectively.

### Local Calculation

Local context information is highly useful for sequence modeling (Kurata et al., 2016; Wang et al., 2016b). Zhang et al. (2018b) propose the S-LSTM to encode both local and sentence-level information simultaneously, and it has been shown more powerful for text representation when compared with the conventional BiLSTMs. We extend the S-LSTM with slot-specific features $\mathbf{H}_t^{slot}$ and intent-specific features $\mathbf{H}_t^{slot}$ retrieved from memories.

Specifically, at each input position $t$, we take the local window context $\xi_t$, word embedding $\mathbf{x}_t$, slot feature $\mathbf{h}_t^{slot}$ and intent feature $\mathbf{h}_t^{int}$ as inputs to conduct combinatorial calculation simultaneously. Formally, in the $l^{th}$ layer, the hidden state $\mathbf{h_t}$ is updated as follows:

$$\begin{aligned} \xi_t^{l-1} &= [\mathbf{h}_{t-1}^{l-1}, \mathbf{h}_t^{l-1}, \mathbf{h}_{t+1}^{l-1}] \\ \hat{\mathbf{i}}_t^l &= \sigma(\mathbf{W}_1^i \xi_t^{l-1} + \mathbf{W}_2^i \mathbf{x}_t + \mathbf{W}_3^i \mathbf{h}_t^{slot} + \mathbf{W}_4^i \mathbf{h}_t^{int}) \\ \hat{\mathbf{o}}_t^l &= \sigma(\mathbf{W}_1^o \xi_t^{l-1} + \mathbf{W}_2^o \mathbf{x}_t + \mathbf{W}_3^o \mathbf{h}_t^{slot} + \mathbf{W}_4^o \mathbf{h}_t^{int}) \\ \hat{\mathbf{f}}_t^l &= \sigma(\mathbf{W}_1^f \xi_t^{l-1} + \mathbf{W}_2^f \mathbf{x}_t + \mathbf{W}_3^f \mathbf{h}_t^{slot} + \mathbf{W}_4^f \mathbf{h}_t^{int}) \\ \hat{\mathbf{l}}_t^l &= \sigma(\mathbf{W}_1^l \xi_t^{l-1} + \mathbf{W}_2^l \mathbf{x}_t + \mathbf{W}_3^l \mathbf{h}_t^{slot} + \mathbf{W}_4^l \mathbf{h}_t^{int}) \\ \hat{\mathbf{r}}_t^l &= \sigma(\mathbf{W}_1^r \xi_t^{l-1} + \mathbf{W}_2^r \mathbf{x}_t + \mathbf{W}_3^r \mathbf{h}_t^{slot} + \mathbf{W}_4^r \mathbf{h}_t^{int}) \\ \mathbf{u}_t^l &= \tanh(\mathbf{W}_1^u \xi_t^{l-1} + \mathbf{W}_2^u \mathbf{x}_t + \mathbf{W}_3^u \mathbf{h}_t^{slot} \\ &\qquad + \mathbf{W}_4^u \mathbf{h}_t^{int}) \\ \mathbf{i}_t^l, \mathbf{f}_t^l, \mathbf{l}_t^l, \mathbf{r}_t^l &= softmax(\hat{\mathbf{i}}_t^l, \hat{\mathbf{f}}_t^l, \hat{\mathbf{l}}_t^l, \hat{\mathbf{r}}_t^l) \\ \mathbf{c}_t^l &= \mathbf{f}_t^l \odot \mathbf{c}_t^{l-1} + \mathbf{l}_t^l \odot \mathbf{c}_{t-1}^{l-1} + \mathbf{r}_t^l \odot \mathbf{c}_{t+1}^{l-1} \\ &\qquad + \mathbf{i}_t^l \odot \mathbf{u}_t^{l-1} \\ \mathbf{h}_t^l &= \mathbf{o}_t^l \odot \tanh \mathbf{c}_t^l \end{aligned} \tag{10}$$

where $\xi_t^l$ is the concatenation of hidden states in a local window, and $\mathbf{i}_t^l$, $\mathbf{f}_t^l$, $\mathbf{o}_t^l$, $\mathbf{l}_t^l$ and $\mathbf{r}_t^l$ are gates to control information flows, and $\mathbf{W}_n^x$ $(x \in \{i, o, f, l, r, u\}, n \in \{1, 2, 3, 4\})$ are model parameters. More details about the state transition can be referred in (Zhang et al., 2018b). In the first CM-block, the hidden state $\mathbf{h}_t$ is initialized with the corresponding word embedding. In other CM-blocks, the $\mathbf{h}_t$ is inherited from the output of the adjacent lower CM-block.

At each word position of above procedures, the hidden state is updated with abundant information from different perspectives, namely word embeddings, local contexts, slots and intents representations. The local calculation layer in each CM-block has been shown highly useful for both tasks, and especially for the slot filling task, which will be validated in our experiments in Section 5.2.

**Global Recurrence**

Bi-directional RNNs, especially the BiLSTMs (Hochreiter and Schmidhuber, 1997) are regarded to encode both past and future information of a sentence, which have become a dominant method in various sequence modeling tasks (Hammerton, 2003; Sundermeyer et al., 2012). The inherent nature of BiLSTMs is able to supplement global sequential information, which is insufficiently modeled in the previous local calculation layer. Thus we apply an additional BiLSTMs layer upon the local calculation layer in each CM-block. By taking the slot- and intent-specific local context representations as inputs, we can obtain more specific global sequential representations. Formally, it takes the hidden state $\mathbf{h}_t^{l-1}$ inherited from the local calculation layer as input, and conduct recurrent steps as follows:

$$\begin{aligned} \mathbf{h}_t^l &= [\overrightarrow{\mathbf{h}_t^l}; \overleftarrow{\mathbf{h}_t^l}] \\ \overrightarrow{\mathbf{h}_t^l} &= \overrightarrow{\mathbf{LSTM}}(\mathbf{h}_t^{l-1}, \overrightarrow{\mathbf{h}}_{t-1}^l; \overrightarrow{\theta}) \\ \overleftarrow{\mathbf{h}_t^l} &= \overleftarrow{\mathbf{LSTM}}(\mathbf{h}_t^{l-1}, \overleftarrow{\mathbf{h}}_{t+1}^l; \overleftarrow{\theta}) \end{aligned} \tag{11}$$

The output "states" of the BiLSTMs are taken as "states" input of the local calculation in next CM-block. The global sequential information encoded by the BiLSTMs is shown necessary and effective for both tasks in our experiments in Section 5.2.

### 3.4 Inference Layer

After multiple rounds of interactions among local context representations, global sequential information, slot and intent features, we conduct predictions upon the final CM-block. For the predictions of slots, we take the hidden states $\mathbf{H}$ along with the retrieved slot $\mathbf{H}^{slot}$ representations (both are from the final CM-block) as input features, and then conduct predictions of slots similarly with the Equation (3) in Section 2:

$$\hat{\mathbf{y}}^{slot} = \underset{\widetilde{\mathbf{y}}^{slot} \in \mathbf{Y}_x}{\arg\max} F([\mathbf{H}; \mathbf{H}^{slot}], \widetilde{\mathbf{y}}^{slot}) \tag{12}$$

For the prediction of intent label, we firstly aggregate the hidden state $\mathbf{h}_t$ and the retrieved intent representation $\mathbf{h}_t^{int}$ at each word position (from the final CM-block as well) via mean pooling:

$$\mathbf{v}^{int} = \frac{1}{N} \sum_t^N [\mathbf{h_t}; \mathbf{h}_t^{int}] \tag{13}$$

and then take the summarized vector $\mathbf{v}^{int}$ as input feature to conduct prediction of intent consistently with the Equation (5) in Section 2.

| Dataset | SNIPS | ATIS | CAIS |
|---|---|---|---|
| Vocab Size | 11241 | 722 | 2146 |
| Average Length | 9.15 | 11.28 | 8.65 |
| # Intents | 7 | 18 | 11 |
| # Slots | 72 | 128 | 75 |
| # Train Set | 13084 | 4478 | 7995 |
| # Validation Set | 700 | 500 | 994 |
| # Test Set | 700 | 893 | 1012 |

Table 2: Dataset statistics.

## 4 Experiments

### 4.1 Datasets and Metrics

We evaluate our proposed CM-Net on three real-word datasets, and statistics are listed in Table 2.

**ATIS** The Airline Travel Information Systems (ATIS) corpus (Hemphill et al., 1990) is the most widely used benchmark for the SLU research. Please note that, there are extra named entity features in the ATIS, which almost determine slot tags. These hand-crafted features are not generally available in open domains (Zhang and Wang, 2016; Guo et al., 2014b), therefore we train our model purely on the training set without additional hand-crafted features.

**SNIPS** SNIPS Natural Language Understanding benchmark [3] (Coucke et al., 2018) is collected in a crowsourced fashion by Snips. The intents of this

[3] https://github.com/snipsco/nlu-benchmark/tree/master/2017-06-custom-intent-engines

| Models | SNIPS | | ATIS | |
|---|---|---|---|---|
| | Slot ($F_1$) | Intent ($Acc$) | Slot ($F_1$) | Intent ($Acc$) |
| Joint GRU (Zhang and Wang, 2016) | – | – | 95.49 | 98.10 |
| Self-Attention, Intent Gate(Li et al., 2018) | – | – | 96.52 | 98.77 |
| Bi-model (Wang et al., 2018) | – | – | **96.89** | 98.99 |
| Attention Bi-RNN (Liu and Lane, 2016) * | 87.80 | 96.70 | 95.98 | 98.21 |
| Joint Seq2Seq (Hakkani-Tür et al., 2016) * | 87.30 | 96.90 | 94.20 | 92.60 |
| Slot-Gated (Intent Atten.) (Goo et al., 2018) | 88.30 | 96.80 | 95.20 | 94.10 |
| Slot-Gated (Full Atten.) (Goo et al., 2018) | 88.80 | 97.00 | 94.80 | 93.60 |
| CAPSULE-NLU(Zhang et al., 2018a) | 91.80 | 97.70 | 95.20 | 95.00 |
| Dilated CNN, Label-Recurrent (Gupta et al., 2019) | 93.11 | 98.29 | 95.54 | 98.10 |
| Sentence-State LSTM (Zhang et al., 2018b) † | 95.80 | 98.30 | 95.65 | 98.21 |
| BiLSTMs + EMLoL (Siddhant et al., 2018) | 93.29 | 98.83 | 95.62 | 97.42 |
| BiLSTMs + EMLo (Siddhant et al., 2018) | 93.90 | **99.29** | 95.42 | 97.30 |
| Joint BERT (Chen et al., 2019) | 97.00 | 98.60 | 96.10 | 97.50 |
| CM-Net (Ours) | **97.15** | **99.29** | 96.20 | **99.10** |

Table 3: Results on test sets of the SNIPS and ATIS, where our CM-Net achieves state-of-the-art performances in most cases. "*" indicates that results are retrieved from Slot-Gated (Goo et al., 2018), and "†" indicates our implementation.

dataset are more balanced when compared with the ATIS. We split another 700 utterances for validation set following previous works (Goo et al., 2018; Zhang et al., 2018a).

**CAIS** We collect utterances from the **C**hinese **A**rtificial **I**ntelligence **S**peakers (CAIS), and annotate them with slot tags and intent labels. The training, validation and test sets are split by the distribution of intents, where detailed statistics are provided in the supplementary material. Since the utterances are collected from speaker systems in the real world, intent labels are partial to the *PlayMusic* option. We adopt the BIOES tagging scheme for slots instead of the BIO2 used in the ATIS, since previous studies have highlighted meaningful improvements with this scheme (Ratinov and Roth, 2009) in the sequence labeling field.

**Metrics** Slot filling is typically treated as a sequence labeling problem, and thus we take the conlleval [4] as the token-level $F_1$ metric. The intent detection is evaluated with the classification accuracy. Specially, several utterances in the ATIS are tagged with more than one labels. Following previous works (Tur et al., 2010; Zhang and Wang, 2016), we count an utterrance as a correct classification if any ground truth label is predicted.

[4]https://www.clips.uantwerpen.be/conll2000/chunking/conlleval.txt

### 4.2 Implementation Details

All trainable parameters in our model are initialized by the method described in Glorot and Bengio (2010). We apply dropout (Srivastava et al., 2014) to the embedding layer and hidden states with a rate of 0.5. All models are optimized by the Adam optimizer (Kingma and Ba, 2014) with gradient clipping of 3 (Pascanu et al., 2013). The initial learning rate $\alpha$ is set to 0.001, and decrease with the growth of training steps. We monitor the training process on the validation set and report the final result on the test set. One layer CNN with a filter of size 3 and max pooling are utilized to generate 100d word embeddings. The cased 300d Glove is adapted to initialize word embeddings, and kept fixed when training. In auxiliary experiments, the output hidden states of BERT are taken as additional word embeddings and kept fixed as well. We share parameters of both memories with the parameter matrices in the corresponding softmax layers, which can be taken as introducing supervised signals into the memories to some extent. We conduct hyper-parameters tuning for layer size (finally set to 3) and loss weight $\lambda$ (finally set to 0.5), and empirically set other parameters to the values listed in the supplementary material.

### 4.3 Main Results

Main results of our CM-Net on the SNIPS and ATIS are shown in Table 3. Our CM-Net achieves the state-of-the-art results on both datasets in terms of slot filling $F_1$ score and intent detection

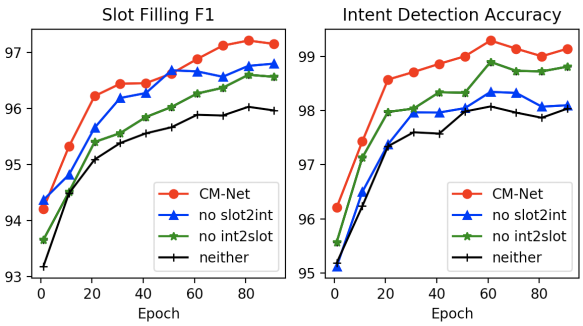


Figure 4: Investigations of the collaborative retrieval approach on slot filling (on the left) and intent detection (on the right), where "no slot2int" indicates removing slow-aware attention for the intent representation, and similarly for "no int2slot" and "neither".

| # | Models | SNIPS | |
|---|---|---|---|
| | | Slot ($F_1$) | Intent ($Acc$) |
| 0 | CM-Net | **97.15** | **99.29** |
| 1 | – slot memory | 96.64 | 99.14 |
| 2 | – intent memory | 96.95 | 98.84 |
| 3 | – local calculation | 96.73 | 99.00 |
| 4 | – global recurrence | 96.80 | 98.57 |

Table 4: Ablation experiments on the SNIPS to investigate the impacts of various components, where "- slot memory" indicates removing the slot memory and its interactions with other components correspondingly. Similarly for the other options.

accuracy, except for the $F_1$ score on the ATIS. We conjecture that the named entity feature in the ATIS has a great impact on the slot filling result as illustrated in Section 4.1. Since the SNIPS is collected from multiple domains with more balanced labels when compared with the ATIS, the slot filling $F_1$ score on the SNIPS is able to demonstrate the superiority of our CM-Net.

It is noteworthy that the CM-Net achieves comparable results when compared with models that exploit additional language models (Siddhant et al., 2018; Chen et al., 2019). We conduct auxiliary experiments by leveraging the well-known BERT (Devlin et al., 2018) as an external resource for a relatively fair comparison with those models, and report details in Section 5.3.

## 5 Analysis

Since the SNIPS corpus is collected from multiple domains and its label distributions are more balanced when compared with the ATIS, we choose the SNIPS to elucidate properties of our CM-Net and conduct several additional experiments.

### 5.1 Whether Memories Promote Each Other?

In the CM-Net, the deliberate attention mechanism is proposed in a collaborative manner to perform information exchange between slots and intents. We conduct experiments to verify whether such kind of knowledge diffusion in both memories can promote each other. More specifically, we remove one unidirectional diffusion (*e.g.* from slot to intent) or both in each experimental setup. The results are illustrated in Figure 4.

We can observe obvious drops on both tasks when both directional knowledge diffusions are removed (CM-Net *vs.* neither). For the slot filling task (left part in Figure 4), the $F_1$ scores decrease slightly when the knowledge from slot to intent is blocked (CM-Net *vs.* "no slot2int"), and a more evident drop occurs when the knowledge from intent to slot is blocked (CM-Net *vs.* "no int2slot"). Similar observations can be found for the intent detection task (right part in Figure 4).

In conclusion, the bidirectional knowledge diffusion between slots and intents are necessary and effective to promote each other.

### 5.2 Ablation Experiments

We conduct ablation experiments to investigate the impacts of various components in our CM-Net. In particular, we remove one component among slot memory, intent memory, local calculation and global recurrence. Results of different combinations are presented in Table 4.

Once the slot memory and its corresponding interactions with other components are removed, scores on both tasks decrease to some extent, and a more obvious decline occurs for the slot filling (row 1 *vs.* row 0), which is consistent with the conclusion of Section 5.1. Similar observations can be found for the intent memory (row 2). The local calculation layer is designed to capture better local context representations, which has an evident impact on the slot filling and slighter effect on the intent detection (row 3 *vs.* row 0). Opposite observations occur in term of global recurrence, which is supposed to model global sequential information and thus has larger effect on the intent detection (row 4 *vs.* row 0).

### 5.3 Effects of Pre-trained Language Models

Recently, there has been a growing body of works exploring neural language models that trained on

| Models | SNIPS | |
|---|---|---|
| | Slot ($F_1$) | Intent ($Acc$) |
| BiLSTMs + EMLoL | 93.29 | 98.83 |
| BiLSTMs + EMLo | 93.90 | 99.29 |
| Joint BERT | 97.00 | 98.60 |
| CM-Net + BERT | **97.31** | **99.32** |

Table 5: Results on the SNIPS benchmark with the assistance of pre-trained language model, where we establish new state-of-the-art results on the SNIPS.

| Models | CAIS | |
|---|---|---|
| | Slot ($F_1$) | Intent ($Acc$) |
| BiLSTMs + CRF | 85.32 | 93.25 |
| S-LSTM + CRF † | 85.74 | 94.36 |
| CM-Net | **86.16** | **94.56** |

Table 6: Results on our CAIS dataset, where "†" indicates our implementation of the S-LSTM.

massive corpora to learn contextual representations (*e.g.* BERT (2018) and EMLo (2018)). Inspired by the effectiveness of language model embeddings, we conduct experiments by leveraging the BERT as an additional feature. The results emerged in Table 5 show that we establish new state-of-the-art results on both tasks of the SNIPS.

### 5.4 Evaluation on the CAIS

We conduct experiments on our self-collected CAIS to evaluate the generalizability in different language. We apply two baseline models for comparison, one is the popular *BiLSTMs + CRF* architecture (Huang et al., 2015) for sequence labeling task, and the other one is the more powerful sententce-state LSTM (Zhang et al., 2018b). The results listed in Table 6 demonstrate the generalizability and effectiveness of our CM-Net when handling various domains and different languages.

## 6 Related Work

**Memory Network** Memory network is a general machine learning framework introduced by Weston et al. (2014), which have been shown effective in question answering (Weston et al., 2014; Sukhbaatar et al., 2015), machine translation (Wang et al., 2016a; Feng et al., 2017), aspect level sentiment classification (Tang et al., 2016), *etc.* For spoken language understanding, Chen et al. (2016) introduce memory mechanisms to encode historical utterances. In this paper, we propose two memories to explicitly capture the semantic correlations between slots and the intent in a given utterance, and devise a novel collaborative retrieval approach.

**Interactions between slots and intents** Considering the semantic proximity between slots and intents, some works propose to enhance the slot filling task unidirectionally with the guidance of intent representations via gating mechanisms (Goo et al., 2018; Li et al., 2018). Intuitively, the slot representations are also instructive to the intent detection task and thus bidirectional interactions between slots and intents are benefical for each other. Zhang et al. (2018a) propose a hierarchical capsule network to perform interactions from words to slots, slots to intents and intents to words in a pipeline manner, which is relatively limited in capturing the complicated correlations among them. In our CM-Net, information exchanges are performed simultaneously with knowledge diffusions in both directions. The experiments demonstrate the superiority of our CM-Net in capturing the semantic correlations between slots and intents.

**Sentence-State LSTM** Zhang et al. 2018b propose a novel graph RNN named S-LSTM, which models sentence between words simultaneously. Inspired by the new perspective of state transition in the S-LSTM, we further extend it with task-specific (*i.e.,* slots and intents) representations via our collaborative memories. In addition, the global information in S-LSTM is modeled by aggregating the local features with gating mechanisms, which may lose sight of sequential information of the whole sentence. Therefore, We apply external BiLSTMs to supply global sequential features, which is shown highly necessary for both tasks in our experiments.

## 7 Conclusion

We propose a novel Collaborative Memory Network (CM-Net) for jointly modeling slot filling and intent detection. The CM-Net is able to explicitly capture the semantic correlations among words, slots and intents in a collaborative manner, and incrementally enrich the information flows with local context and global sequential information. Experiments on two standard benchmarks and our CAIS corpus demonstrate the effectiveness and generalizability of our proposed CM-Net. In addition, we contribute the new corpus (CAIS) to the research community.

## Acknowledgments

Liu, Chen and Xu are supported by the National Natural Science Foundation of China (Contract 61370130, 61976015, 61473294 and 61876198), and the Beijing Municipal Natural Science Foundation (Contract 4172047), and the International Science and Technology Cooperation Program of the Ministry of Science and Technology (K11F100010). We sincerely thank the anonymous reviewers for their thorough reviewing and valuable suggestions.

## References

Qian Chen, Zhu Zhuo, and Wen Wang. 2019. Bert for joint intent classification and slot filling.

Yun-Nung Chen, Dilek Hakkani-Tür, Gökhan Tür, Jianfeng Gao, and Li Deng. 2016. End-to-end memory networks with knowledge carryover for multi-turn spoken language understanding. In *Interspeech*, pages 3245–3249.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Alice Coucke, Alaa Saade, Adrien Ball, Thodore Bluche, Alexandre Caulier, David Leroy, Clment Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, Mal Primet, and Joseph Dureau. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Yang Feng, Shiyue Zhang, Andi Zhang, Dong Wang, and Andrew Abel. 2017. Memory-augmented neural machine translation.

G David Forney. 1973. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

Chih-Wen Goo, Guang Gao, Yun-Kai Hsu, Chih-Li Huo, Tsung-Chieh Chen, Keng-Wei Hsu, and Yun-Nung Chen. 2018. Slot-gated modeling for joint slot filling and intent prediction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 753–757, New Orleans, Louisiana. Association for Computational Linguistics.

Daniel Guo, Gokhan Tur, Wen-tau Yih, and Geoffrey Zweig. 2014a. Joint semantic utterance classification and slot filling with recursive neural networks. In *2014 IEEE Spoken Language Technology Workshop (SLT)*, pages 554–559. IEEE.

Daniel Guo, Gokhan Tur, Wen-tau Yih, and Geoffrey Zweig. 2014b. Joint semantic utterance classification and slot filling with recursive neural networks. In *2014 IEEE Spoken Language Technology Workshop (SLT)*, pages 554–559. IEEE.

Arshit Gupta, John Hewitt, and Katrin Kirchhoff. 2019. Simple, fast, accurate intent classification and slot labeling.

Patrick Haffner, Gokhan Tur, and Jerry H Wright. 2003. Optimizing svms for complex call classification. In *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03).*, volume 1, pages I–I. IEEE.

Dilek Hakkani-Tür, Gökhan Tür, Asli Celikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In *Interspeech*, pages 715–719.

James Hammerton. 2003. Named entity recognition with long short-term memory. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 172–175. Association for Computational Linguistics.

Charles T. Hemphill, John J. Godfrey, and George R. Doddington. 1990. The ATIS spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990*.

Sepp Hochreiter and Jrgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Gakuto Kurata, Bing Xiang, Bowen Zhou, and Mo Yu. 2016. Leveraging sentence-level information with encoder lstm for semantic slot filling. *arXiv preprint arXiv:1601.01530*.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270. Association for Computational Linguistics.

Changliang Li, Liang Li, and Ji Qi. 2018. A self-attentive model with gate mechanism for spoken language understanding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3824–3833, Brussels, Belgium. Association for Computational Linguistics.

Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. *arXiv preprint arXiv:1609.01454*.

Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, et al. 2014. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):530–539.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*, pages 2227–2237. Association for Computational Linguistics.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the thirteenth conference on computational natural language learning*, pages 147–155. Association for Computational Linguistics.

Ruhi Sarikaya, Geoffrey E Hinton, and Bhuvana Ramabhadran. 2011. Deep belief nets for natural language call-routing. In *2011 IEEE International conference on acoustics, speech and signal processing (ICASSP)*, pages 5680–5683. IEEE.

Aditya Siddhant, Anuj Goyal, and Angeliki Metallinou. 2018. Unsupervised transfer learning for spoken language understanding in intelligent agents. *AAAI*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*.

Duyu Tang, Bing Qin, and Ting Liu. 2016. Aspect level sentiment classification with deep memory network. *arXiv preprint arXiv:1605.08900*.

Gokhan Tur, Dilek Hakkani-Tür, and Larry Heck. 2010. What is left to be understood in atis? In *2010 IEEE Spoken Language Technology Workshop*, pages 19–24. IEEE.

Mingxuan Wang, Zhengdong Lu, Hang Li, and Qun Liu. 2016a. Memory-enhanced decoder for neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 278–286, Austin, Texas. Association for Computational Linguistics.

Xingyou Wang, Weijie Jiang, and Zhiyong Luo. 2016b. Combination of convolutional and recurrent neural network for sentiment analysis of short texts. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2428–2437.

Yu Wang, Yilin Shen, and Hongxia Jin. 2018. A bi-model based RNN semantic frame parsing model for intent detection and slot filling. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 309–314, New Orleans, Louisiana. Association for Computational Linguistics.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks.

Puyang Xu and Ruhi Sarikaya. 2013. Convolutional neural network based triangular crf for joint intent detection and slot filling. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 78–83. IEEE.

Chenwei Zhang, Yaliang Li, Nan Du, Wei Fan, and Philip S Yu. 2018a. Joint slot filling and intent detection via capsule neural networks. *arXiv preprint arXiv:1812.09471*.

Xiaodong Zhang and Houfeng Wang. 2016. A joint model of intent determination and slot filling for spoken language understanding. In *IJCAI*, pages 2993–2999.

Yue Zhang, Qi Liu, and Linfeng Song. 2018b. Sentence-state lstm for text representation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 317–327. Association for Computational Linguistics.

# Tree Transformer: Integrating Tree Structures into Self-Attention

**Yau-Shian Wang  Hung-Yi Lee  Yun-Nung Chen**
National Taiwan University, Taipei, Taiwan
king6101@gmail.com  hungyilee@ntu.edu.tw  y.v.chen@ieee.org

## Abstract

Pre-training Transformer from large-scale raw texts and fine-tuning on the desired task have achieved state-of-the-art results on diverse NLP tasks. However, it is unclear what the learned attention captures. The attention computed by attention heads seems not to match human intuitions about hierarchical structures. This paper proposes Tree Transformer, which adds an extra constraint to attention heads of the bidirectional Transformer encoder in order to encourage the attention heads to follow tree structures. The tree structures can be automatically induced from raw texts by our proposed "*Constituent Attention*" module, which is simply implemented by self-attention between two adjacent words. With the same training procedure identical to BERT, the experiments demonstrate the effectiveness of Tree Transformer in terms of inducing tree structures, better language modeling, and further learning more explainable attention scores[1].

## 1 Introduction

Human languages exhibit a rich hierarchical structure which is currently not exploited nor mirrored by the self-attention mechanism that is the core of the now popular Transformer architecture. Prior work that integrated hierarchical structure into neural networks either used recursive neural networks (Tree-RNNs) (C.Goller and A.Kuchler, 1996; Socher et al., 2011; Tai et al., 2015) or simultaneously generated a syntax tree and language in RNN (Dyer et al., 2016), which have shown beneficial for many downstream tasks (Aharoni and Goldberg, 2017; Eriguchi et al., 2017; Strubell et al., 2018; Zaremoodi and Haffari, 2018). Considering the requirement of the annotated parse trees and the costly annotation effort, most prior work relied on the supervised syntactic parser. However, a supervised parser may be unavailable when the language is low-resourced or the target data has different distribution from the source domain.

Therefore, the task of learning latent tree structures without human-annotated data, called *grammar induction* (Carroll and Charniak, 1992; Klein and Manning, 2002; Smith and Eisner, 2005), has become an important problem and attractd more attention from researchers recently. Prior work mainly focused on inducing tree structures from recurrent neural networks (Shen et al., 2018a,b) or recursive neural networks (Yogatama et al., 2017; Drozdov et al., 2019), while integrating tree structures into Transformer remains an unexplored direction.

Pre-training Transformer from large-scale raw texts successfully learns high-quality language representations. By further fine-tuning pre-trained Transformer on desired tasks, wide range of NLP tasks obtain the state-of-the-art results (Radford et al., 2019; Devlin et al., 2018; Dong et al., 2019). However, what pre-trained Transformer self-attention heads capture remains unknown. Although an attention can be easily explained by observing how words attend to each other, only some distinct patterns such as attending previous words or named entities can be found informative (Vig, 2019). The attention matrices do not match our intuitions about hierarchical structures.

In order to make the attention learned by Transformer more interpretable and allow Transformer to comprehend language hierarchically, we propose Tree Transformer, which integrates tree structures into bidirectional Transformer encoder. At each layer, words are constrained to attend to other words in the same constituents. This constraint has been proven to be effective in prior work (Wu et al., 2018). Different from the prior

[1]The source code is publicly available at https://github.com/yaushian/Tree-Transformer.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1061–1070, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

work that required a supervised parser, in Tree Transformer, the constituency tree structures is automatically induced from raw texts by our proposed "*Constituent Attention*" module, which is simply implemented by self-attention. Motivated by Tree-RNNs, which compose each phrase and the sentence representation from its constituent sub-phrases, Tree Transformer gradually attaches several smaller constituents into larger ones from lower layers to higher layers.

The contributions of this paper are 3-fold:

- Our proposed Tree Transformer is easy to implement, which simply inserts an additional "Constituent Attention" module implemented by self-attention to the original Transformer encoder, and achieves good performance on the unsupervised parsing task.
- As the induced tree structures guide words to compose the meaning of longer phrases hierarchically, Tree Transformer improves the perplexity on masked language modeling compared to the original Transformer.
- The behavior of attention heads learned by Tree Transformer expresses better interpretability, because they are constrained to follow the induced tree structures. By visualizing the self-attention matrices, our model provides the information that better matchs the human intuition about hierarchical structures than the original Transformer.

## 2 Related Work

This section reviews the recent progress about grammar induction. Grammar induction is the task of inducing latent tree structures from raw texts without human-annotated data. The models for grammar induction are usually trained on other target tasks such as language modeling. To obtain better performance on the target tasks, the models have to induce reasonable tree structures and utilize the induced tree structures to guide text encoding in a hierarchical order. One prior attempt formulated this problem as a reinforcement learning (RL) problem (Yogatama et al., 2017), where the unsupervised parser is an actor in RL and the parsing operations are regarded as its actions. The actor manages to maximize total rewards, which are the performance of downstream tasks.

PRPN (Shen et al., 2018a) and On-LSTM (Shen et al., 2018b) induce tree structures by introducing a bias to recurrent neural networks. PRPN proposes a parsing network to compute the syntactic distance of all word pairs, and a reading network utilizes the syntactic structure to attend relevant memories. On-LSTM allows hidden neurons to learn long-term or short-term information by the proposed new gating mechanism and new activation function. In URNNG (Kim et al., 2019b), they applied amortized variational inference between a recurrent neural network grammar (RNNG) (Dyer et al., 2016) decoder and a tree structures inference network, which encourages the decoder to generate reasonable tree structures. DIORA (Drozdov et al., 2019) proposed using inside-outside dynamic programming to compose latent representations from all possible binary trees. The representations of inside and outside passes from same sentences are optimized to be close to each other. Compound PCFG (Kim et al., 2019a) achieves grammar induction by maximizing the marginal likelihood of the sentences which are generated by a probabilistic context-free grammar (PCFG) in a corpus.

## 3 Tree Transformer

Given a sentence as input, Tree Transformer induces a tree structure. A 3-layer Tree Transformer is illustrated in Figure 1(A). The building blocks of Tree Transformer is shown in Figure 1(B), which is the same as those used in bidirectional Transformer encoder, except the proposed Constituent Attention module. The blocks in Figure 1(A) are constituents induced from the input sentence. The red arrows indicate the self-attention. The words in different constituents are constrained to not attend to each other. In the 0-th layer, some neighboring words are merged into constituents; for example, given the sentence "*the cute dog is wagging its tail*", the tree Transformer automatically determines that "*cute*" and "*dog*" form a constituent, while "*its*" and "*tail*" also form one. The two neighboring constituents may merge together in the next layer, so the sizes of constituents gradually grow from layer to layer. In the top layer, the layer 2, all words are grouped into the same constituent. Because all words are into the same constituent, the attention heads freely attend to any other words, in this layer, Tree Transformer behaves the same as the typical Transformer encoder. Tree Transformer can be trained in an end-to-end fashion by using "masked LM",

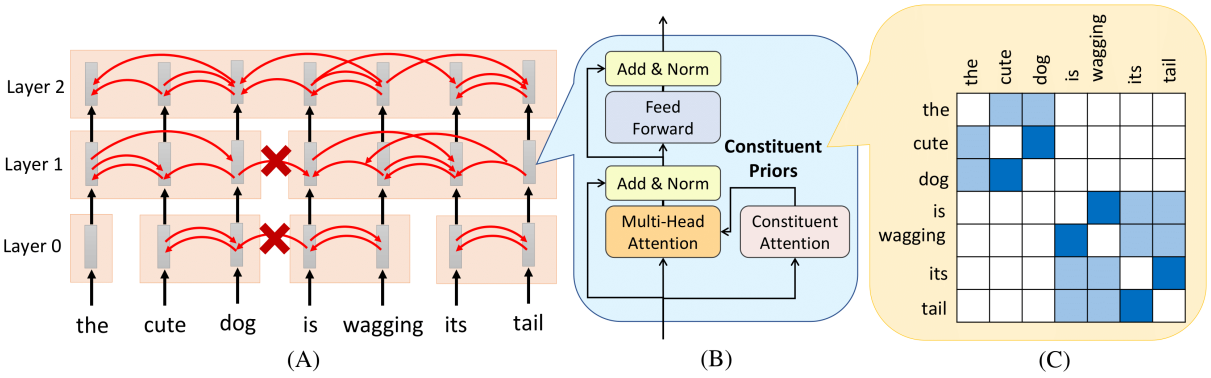


Figure 1: (A) A 3-layer Tree Transformer, where the blocks are constituents induced from the input sentence. The two neighboring constituents may merge together in the next layer, so the sizes of constituents gradually grow from layer to layer. The red arrows indicate the self-attention. (B) The building blocks of Tree Transformer. (C) Constituent prior $C$ for the layer 1.

which is one of the unsupervised training task used for BERT training.

Whether two words belonging to the same constituent is determined by "*Constituent Prior*" that guides the self-attention. Constituent Prior is detailed in Section 4, which is computed by the proposed Constituent Attention module in Section 5. By using BERT masked language model as training, latent tree structures emerge from Constituent Prior and unsupervised parsing is thereby achieved. The method for extracting the constituency parse trees from Tree Transformer is described in Section 6.

## 4 Constituent Prior

In each layer of Transformer, there are a query matrix $Q$ consisting of query vectors with dimension $d_k$ and a key matrix $K$ consisting of key vectors with dimension $d_k$. The attention probability matrix is denoted as $E$, which is an $N$ by $N$ matrix, where $N$ is the number of words in an input sentence. $E_{i,j}$ is the probability that the position $i$ attends to the position $j$. The Scaled Dot-Product Attention computes the $E$ as:

$$E = \text{softmax}(\frac{QK^T}{d}), \tag{1}$$

where the dot-product is scaled by $1/d$. In Transformer, the scaling factor $d$ is set to be $\sqrt{d_k}$.

In Tree Transformer, the $E$ is not only determined by the query matrix $Q$ and key matrix $K$, but also guided by Constituent Prior $C$ generating from Constituent Attention module. Same as $E$, the constituent prior $C$ is also a $N$ by $N$ matrix, where $C_{i,j}$ is the probability that word $w_i$ and word $w_j$ belong to the same constituency. This matrix is symmetric that $C_{i,j}$ is same as $C_{j,i}$. Each layer has its own Constituent Prior $C$. An example of Constituent Prior $C$ is illustrated in Figure 1 (C), which indicates that in layer 1, "the cute dog" and "is wagging its tail" are two constituents.

To make each position not attend to the position in different constituents, Tree Transformer constrains the attention probability matrix $E$ by constituent prior $C$ as below,

$$E = C \odot \text{softmax}(\frac{QK^T}{d}), \tag{2}$$

where $\odot$ is the element-wise multiplication. Therefore, if $C_{i,j}$ has small value, it indicates that the positions $i$ and $j$ belong to different constituents, where the attention weight $E_{i,j}$ would be small. As Transformer uses multi-head attention with $h$ different heads, there are $h$ different query matrices $Q$ and key matrices $K$ at each position, but here in the same layer, all attention heads in multi-head attention share the same $C$. The multi-head attention module produces the output of dimension $d_{model} = h \times d_k$.

## 5 Constituent Attention

The proposed Constituent Attention module is to generate the constituent prior $C$. Instead of directly generating $C$, we decompose the problem into estimating the breakpoints between constituents, or the probability that two adjacent words belong to the same constituent. In each layer, the Constituent Attention module generates a sequence $a = \{a_1, ..., a_i, ..., a_N\}$, where $a_i$ is the probability that the word $w_i$ and its neighbor

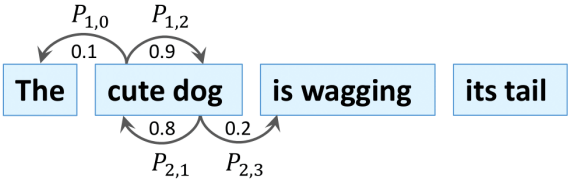


Figure 2: The example illustration about how neighboring attention works.

word $w_{i+1}$ are in the same constituent. The small value of $a_i$ implies that there is a breakpoint between $w_i$ and $w_{i+1}$, so the constituent prior $C$ is obtained from the sequence $a$ as follows. $C_{i,j}$ is the multiplication of all $a_{i \leqslant k<j}$ between word $w_i$ and word $w_j$:

$$C_{i,j} = \prod_{k=i}^{j-1} a_k. \tag{3}$$

In (3), we choose to use multiplication instead of summation, because if one of $a_{i \leqslant k<j}$ between two words $w_i$ and $w_j$ is small, the value of $C_{i,j}$ with multiplication also becomes small. In implementation, to avoid probability vanishing, we use log-sum instead of directly multiplying all $a$:

$$C_{i,j} = e^{\sum_{k=i}^{j-1} \log(a_k)}. \tag{4}$$

The sequence $a$ is obtained based on the following two mechanisms: *Neighboring Attention* and *Hierarchical Constraint*.

### 5.1 Neighboring Attention

We compute the score $s_{i,i+1}$ indicating that $w_i$ links to $w_{i+1}$ by scaled dot-product attention:

$$s_{i,i+1} = \frac{q_i \cdot k_{i+1}}{d}, \tag{5}$$

where $q_i$ is a link query vector of $w_i$ with $d_{model}$ dimensions, and $k_{i+1}$ is a link key vector of $w_{i+1}$ with $d_{model}$ dimensions. We use $q_i \cdot k_{i+1}$ to represent the tendency that $w_i$ and $w_{i+1}$ belong to the same constituent. Here, we set the scaling factor $d$ to be $\frac{d_{model}}{2}$. The query and key vectors in (5) are different from (1). They are computed by the same network architecture, but with different sets of network parameters.

For each word, we constrain it to either link to its *right neighbor* or *left neighbor* as illustrated in Figure 2. This constraint is implemented by applying a softmax function to two attention links of $w_i$:

$$p_{i,i+1}, p_{i,i-1} = \text{softmax}(s_{i,i+1}, s_{i,i-1}), \tag{6}$$

where $p_{i,i+1}$ is the probability that $w_i$ attends to $w_{i+1}$, and $(p_{i,i+1} + p_{i,i-1}) = 1$. We find that without the constraint of the softmax operation in (6) the model prefers to link all words together and assign all words to the same constituency. That is, giving both $s_{i,i+1}$ and $s_{i,i-1}$ large values, so the attention head freely attends to any position without restriction of constituent prior, which is the same as the original Transformer. Therefore, the softmax function is to constraint the attention to be sparse.

As $p_{i,i+1}$ and $p_{i+1,i}$ may have different values, we average its two attention links:

$$\hat{a}_i = \sqrt{p_{i,i+1} \times p_{i+1,i}}. \tag{7}$$

The $\hat{a}_i$ links two adjacent words only if two words attend to each other. $\hat{a}_i$ is used in the next subsection to obtain $a_i$.

### 5.2 Hierarchical Constraint

As mentioned in Section 3, constituents in the lower layer merge into larger one in the higher layer. That is, once two words belong to the same constituent in the lower layer, they would still belong to the same constituent in the higher layer. To apply the hierarchical constraint to the tree Transformer, we restrict $a_k^l$ to be always larger than $a_k^{l-1}$ for the layer $l$ and word index $k$. Hence, at the layer $l$, the link probability $a_k^l$ is set as:

$$a_k^l = a_k^{l-1} + (1 - a_k^{l-1})\hat{a}_k^l, \tag{8}$$

where $a_k^{l-1}$ is the link probability from the previous layer $l-1$, and $\hat{a}_k^l$ is obtained from Neighboring Attention (Section 5.1) of the current layer $l$. Finally, at the layer $l$, we apply (4) for computing $C^l$ from $a^l$. Initially, different words are regarded as different constituents, and thus we initialize $a_k^{-1}$ as zero.

## 6 Unsupervised Parsing from Tree Transformer

After training, the neighbor link probability $a$ can be used for unsupervised parsing. The small value of $a$ suggests this link be the breakpoint of two constituents. By top-down greedy parsing (Shen et al., 2018a), which recursively splits the sentence into two constituents with minimum $a$, a parse tree can be formed.

However, because each layer has a set of $a^l$, we have to decide to use which layer for parsing. Instead of using $a$ from a specific layer for parsing

**Algorithm 1** Unsupervised Parsing with Multiple Layers

$a \leftarrow$ link probabilities
$m \leftarrow minimum\ layer\ id$ ▷ Discard the $a$ from layers below minimum layer
$thres \leftarrow 0.8$ ▷ Threshold of breakpoint
**procedure** BUILDTREE($l, s, e$) ▷ $l$: layer index, $s$: start index, $e$: end index
  **if** $e - s < 2$ **then** ▷ The constituent cannot be split
    **return** $(s, e)$
  $span \leftarrow a^l_{s \le i < e}$
  $b \leftarrow$ **argmin**$(span)$ ▷ Get breakpoint
  $last \leftarrow$ **max**$(l-1, m)$ ▷ Get index of last layer
  **if** $a^l_b > thres$ **then**
    **if** $l = m$ **then**
      **return** $(s, e)$
    **return BuildTree**$(last, s, e)$
  $tree1 \leftarrow$ **BuildTree**$(last, s, b)$
  $tree2 \leftarrow$ **BuildTree**$(last, b+1, e)$
  **return** $(tree1, tree2)$ ▷ Return tree

(Shen et al., 2018b), we propose a new parsing algorithm, which utilizes $a$ from all layers for unsupervised parsing. As mentioned in Section 5.2, the values of $a$ are strictly increasing, which indicates that $a$ directly learns the hierarchical structures from layer to layer. Algorithm 1 details how we utilize hierarchical information of $a$ for unsupervised parsing.

The unsupervised parsing starts from the top layer, and recursively moves down to the last layer after finding a breakpoint until reaching the bottom layer $m$. The bottom layer $m$ is a hyperparameter needed to be tuned, and is usually set to 2 or 3. We discard $a$ from layers below $m$, because we find the lowest few layers do not learn good representations (Liu et al., 2019) and thus the parsing results are poor (Shen et al., 2018b). All values of $a$ on top few layers are very close to 1, suggesting that those are not good breakpoints. Therefore, we set a threshold for deciding a breakpoint, where a minimum $a$ will be viewed as a valid breakpoint only if its value is below the threshold. As we find that our model is not very sensitive to the threshold value, we set it to be 0.8 for all experiments.

## 7 Experiments

In order to evaluate the performance of our proposed model, we conduct the experiments detailed below.

### 7.1 Model Architecture

Our model is built upon a bidirectional Transformer encoder. The implementation of our Transformer encoder is identical to the original Transformer encoder. For all experiments, we set the hidden size $d_{model}$ of Constituent Attention and Transformer as 512, the number of self-attention heads $h$ as 8, the feed-forward size as 2048 and the dropout rate as 0.1. We analyze and discuss the sensitivity of the number of layers, denoted as $L$, in the following experiments.

### 7.2 Grammar Induction

In this section, we evaluate the performance of our model on unsupervised constituency parsing. Our model is trained on WSJ training set and WSJ-all (i.e. including testing and validation sets) by using BERT Masked LM (Devlin et al., 2018) as unsupervised training task. We use WordPiece (Wu et al., 2016) tokenizer from BERT to tokenize words with a 16k token vocabulary. Our best result is optimized by `adam` with a learning rate of 0.0001, $\beta_1 = 0.9$ and $\beta_2 = 0.98$. Following the evaluation settings of prior work (Htut et al., 2018; Shen et al., 2018b)[2], we evaluate F1 scores of our model on WSJ-test and WSJ-10 of Penn Treebank (PTB) (Marcus et al., 1993). The WSJ-10 has 7422 sentences from whole PTB with sentence length restricted to 10 after punctuation removal, while WSJ-test has 2416 sentences from the PTB testing set with unrestricted sentence length.

The results on WSJ-test are in Table 1. We mainly compare our model to PRPN (Shen et al., 2018a), On-lstm (Shen et al., 2018b) and Compound PCFG(C-PCFG) (Kim et al., 2019a), in which the evaluation settings and the training data are identical to our model. DIORA (Drozdov et al., 2019) and URNNG (Kim et al., 2019b) use a relative larger training data and the evaluation settings are slightly different from our model. Our model performs much better than trivial trees (i.e. right and left-branching trees) and random trees, which suggests that our proposed model successfully learns meaningful trees. We also find that

[2] https://github.com/yikangshen/Ordered-Neurons

| Model | Data | $F1_{median}$ | $F1_{max}$ |
|---|---|---|---|
| PRPN | WSJ-train | 35.0 | 42.8 |
| On-lstm | WSJ-train | 47.7 | 49.4 |
| C-PCFG | WSJ-train | **55.2** | **60.1** |
| Tree-T,L=12 | WSJ-train | 48.4 | 50.2 |
| Tree-T,L=10 | WSJ-train | 49.5 | 51.1 |
| Tree-T,L=8 | WSJ-train | 48.3 | 49.6 |
| Tree-T,L=6 | WSJ-train | 47.4 | 48.8 |
| DIORA | NLI | **55.7** | **56.2** |
| URNNG | Billion | - | 52.4 |
| Tree-T,L=10 | WSJ-all | 50.5 | 52.0 |
| Random | - | 21.6 | 21.8 |
| LB | - | 9.0 | 9.0 |
| RB | - | 39.8 | 39.8 |

Table 1: **The F1 scores on WSJ-test.** Tree Transformer is abbreviated as Tree-T, and L is the number of layers(blocks). DIORA is trained on multi-NLI dataset (Williams et al., 2018). URNNG is trained on the subset of one billion words (Chelba et al., 2013) with 1M training data. LB and RB are the left and right-brancing baselines.

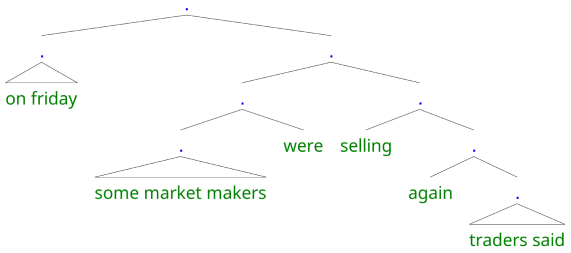


Figure 3: A parse tree induced by Tree Transformer. As shown in the figure, because we set a threshold in Algorithm 1, the leaf nodes are not strictly binary.

increasing the layer number results in better performance, because it allows the Tree Transformer to model deeper trees. However, the performance stops growing when the depth is above 10. The words in the layers above the certain layer are all grouped into the same constituent, and therefore increasing the layer number will no longer help model discover useful tree structures. In Table 2, we report the results on WSJ-10. Some of the baselines including CCM (Klein and Manning, 2002), DMV+CCM (Klein and Manning, 2005) and UML-DOP (Bod, 2006) are not directly comparable to our model, because they are trained using POS tags our model does not consider.

In addition, we further investigate what kinds of trees are induced by our model. Following URNNG, we evaluate the performance of constituents by its label in Table 3. The trees induced by different methods are quite different. Our model is inclined to discover noun phrases (NP) and adverb phrases (ADVP), but not easy to discover verb phrases (VP) or adjective phrases (ADJP). We show an induced parse tree in Figure 3 and more induced parse trees can be found in Appendix.

| Model | Data | $F1_{median}$ | $F1_{max}$ |
|---|---|---|---|
| PRPN | WSJ-train | 70.5 | 71.3 |
| On-lstm | WSJ-train | 65.1 | 66.8 |
| C-PCFG | WSJ-train | 70.5 | - |
| Tree-T,L=10 | WSJ-train | 66.2 | 67.9 |
| DIORA | NLI | 67.7 | 68.5 |
| Tree-T,L=10 | WSJ-all | 66.2 | 68.0 |
| CCM | WSJ-10 | - | 71.9 |
| DMV+CCM | WSJ-10 | - | 77.6 |
| UML-DOP | WSJ-10 | - | **82.9** |
| Random | - | 31.9 | 32.6 |
| LB | - | 19.6 | 19.6 |
| RB | - | 56.6 | 56.6 |

Table 2: **The F1 scores on WSJ-10.** Tree Transformer is abbreviated as Tree-T, and L is the number of layers (blocks).

| Label | Tree-T | URNNG | PRPN |
|---|---|---|---|
| NP | **67.6** | 39.5 | 63.9 |
| VP | 38.5 | **76.6** | 27.3 |
| PP | 52.3 | **55.8** | 55.1 |
| ADJP | 24.7 | 33.9 | **42.5** |
| SBAR | 36.4 | **74.8** | 28.9 |
| ADVP | **55.1** | 50.4 | 45.1 |

Table 3: **Recall of constituents by labels.** The results of URNNG and PRPN are taken from Kim et al. (2019b).

### 7.3 Analysis of Induced Structures

In this section, we study whether Tree Transformer learns hierarchical structures from layers to layers. First, we analyze the influence of the hyperparameter minimum layer $m$ in Algorithm. 1 given the model trained on WSJ-all in Table 1. As illustrated in Figure 4(a), setting $m$ to be 3 yields the best performance. Prior work discovered that the representations from the lower layers of Transformer are not informative (Liu et al., 2019). Therefore, using syntactic structures from lower layers decreases the quality of parse trees. On the other hand, most syntactic information is

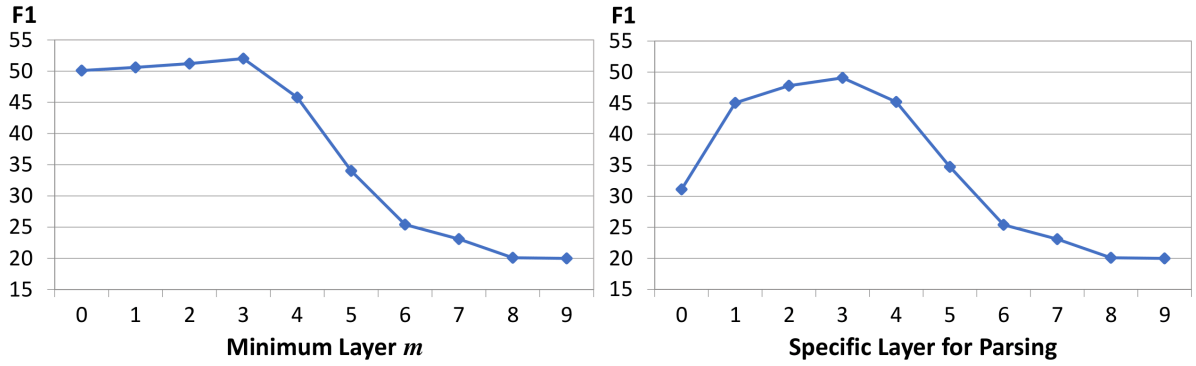


(a) F1 in terms of different minimum layers.

(b) F1 of parsing via a specific layer.

Figure 4: Performance of unsupervised parsing.

missing when $a$ from top few layers are close to 1, so too large $m$ also decreases the performance.

To further analyze which layer contains richer information of syntactic structures, we evaluate the performance on obtaining parse trees from a specific layer. We use $a^l$ from the layer $l$ for parsing with the top-down greedy parsing algorithm (Shen et al., 2018a). As shown in Figure 4(b), using $a^3$ from the layer 3 for parsing yields the best F1 score, which is 49.07. The result is consistent to the best value of $m$. However, compared to our best result (52.0) obtained by Algorithm 1, the F1-score decreases by 3 ($52.0 \rightarrow 49.07$). This demonstrates the effectiveness of Tree Transformer in terms of learning hierarchical structures. The higher layers indeed capture the higher-level syntactic structures such as clause patterns.

### 7.4 Interpretable Self-Attention

This section discusses whether the attention heads in Tree Transformer learn hierarchical structures. Considering that the most straightforward way of interpreting what attention heads learn is to visualize the attention scores, we plot the heat maps of Constituent Attention prior $C$ from each layer in Figure 5.

In the heat map of constituent prior from first layer (Figure 5(a)), as the size of constituent is small, the words only attend to its adjacent words. We can observe that the model captures some subphrase structures, such as the noun phrase "*delta air line*" or "*american airlines unit*". In Figure 5(b)-(d), the constituents attach to each other and become larger. In the layer 6, the words from "*involved*" to last word "*lines*" form a high-level adjective phrase (ADJP). In the layer 9, all words are grouped into a large constituent except the first word "*but*". By visualizing the heat maps of the constituent prior from each layer, we can easily know what types of syntactic structures are learned in each layer. The parse tree of this example can be found in Figure 3 of Appendix. We also visualize the heat maps of self-attention from the original Transformer layers and one from the Tree Transformer layers in Appendix A. As the self-attention heads from our model are constrained by the constituent prior, compared to the original Transformer, we can discover hierarchical structures more easily.

Those attention heat maps demonstrate that: (1) the size of constituents gradually grows from layer to layer, and (2) at each layer, the attention heads tend to attend to other words within constituents posited in that layer. Those two evidences support the success of the proposed Tree Transformer in terms of learning tree-like structures.

| Model | L | Params | Perplexity |
|---|---|---|---|
| Transformer | 8 | 40M | 48.8 |
| Transformer | 10 | 46M | 48.5 |
| Transformer | 10-B | 67M | 49.2 |
| Transformer | 12 | 52M | 48.1 |
| Tree-T | 8 | 44M | 46.1 |
| Tree-T | 10 | 51M | 45.7 |
| Tree-T | 12 | 58M | 45.6 |

Table 4: **The perplexity of masked words.** Params is the number of parameters. We denote the number of layers as $L$. In Transformer $L = 10 - B$, the increased hidden size results in more parameters.

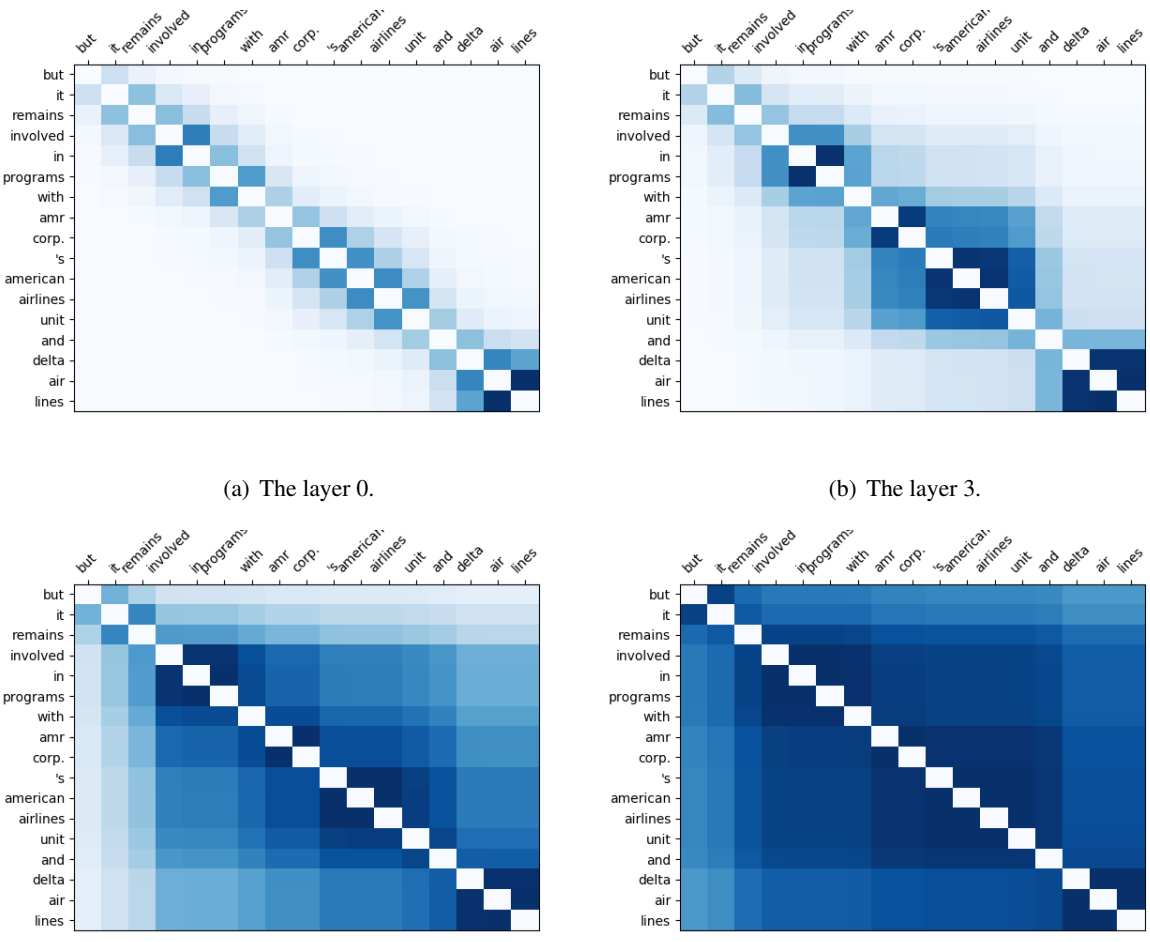


(a) The layer 0.

(b) The layer 3.

(c) The layer 6.

(d) The layer 9.

Figure 5: The constituent prior heat maps.

### 7.5 Masked Language Modeling

To investigate the capability of Tree Transformer in terms of capturing abstract concepts and syntactic knowledge, we evaluate the performance on language modeling. As our model is a bidirectional encoder, in which the model can see its subsequent words, we cannot evaluate the language model in a left-to-right manner. We evaluate the performance on masked language modeling by measuring the perplexity on masked words[3]. To perform the inference without randomness, for each sentence in the testing set, we mask all words in the sentence, but not at once. In each masked testing data, only one word is replaced with a "[MASK]" token. Therefore, each sentence creates the number of testing samples equal to its length.

In Table 4, the models are trained on WSJ-train with BERT masked LM and evaluated on WSJ-test. All hyperparameters except the number of layers in Tree Transformer and Transformer are set to be the same and optimized by the same optimizer. We use `adam` as our optimizer with learning rate of 0.0001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Our proposed Constituent Attention module increases about $10\%$ hyperparameters to the original Transformer encoder and the computational speed is 1.2 times slower. The results with best performance on validation set are reported. Compared to the original Transformer, Tree Transformer achieves better performance on masked language modeling. As the performance gain is possibly due to more parameters, we adjust the number of layers or increase the number of hidden layers in Transformer $L = 10 - B$. Even with fewer parameters than Transformer, Tree Transformer still performs bet-

[3]The perplexity of masked words is $e^{\frac{-\sum \log(p)}{n_{mask}}}$, where $p$ is the probability of correct masked word to be predicted and $n_{mask}$ is the total number of masked words.

ter.

The performance gain is because the induced tree structures guide the self-attention processes language in a more straightforward and human-like manner, and thus the knowledge can be better generalized from training data to testing data. Also, Tree Transformer acquires positional information not only from positional encoding but also from the induced tree structures, where the words attend to other words from near to distant (lower layers to higher layers)[4].

### 7.6 Limitations and Discussion

It is worth mentioning that we have tried to initialize our Transformer model with pre-trained BERT, and then fine-tuning on WSJ-train. However, in this setting, even when the training loss becomes lower than the loss of training from scratch, the parsing result is still far from our best results. This suggests that the attention heads in pre-trained BERT learn quite different structures from the tree-like structures in Tree Transformer. In addition, with a well-trained Transformer, it is not necessary for the Constituency Attention module to induce reasonable tree structures, because the training loss decreases anyway.

## 8 Conclusion

This paper proposes Tree Transformer, a first attempt of integrating tree structures into Transformer by constraining the attention heads to attend within constituents. The tree structures are automatically induced from the raw texts by our proposed Constituent Attention module, which attaches the constituents to each other by self-attention. The performance on unsupervised parsing demonstrates the effectiveness of our model in terms of inducing tree structures coherent to human expert annotations. We believe that incorporating tree structures into Transformer is an important and worth exploring direction, because it allows Transformer to learn more interpretable attention heads and achieve better language modeling. The interpretable attention can better explain how the model processes the natural language and guide the future improvement.

[4]We do not remove the positional encoding from the Transformer and we find that without positional encoding the quality of induced parse trees drops.

## Acknowledgements

We would like to thank reviewers for their insightful comments. This work was financially supported from the Young Scholar Fellowship Program by Ministry of Science and Technology (MOST) in Taiwan, under Grant 108-2636-E002-003.

## References

Roee Aharoni and Yoav Goldberg. 2017. Towards string-to-tree neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.

Rens Bod. 2006. An all-subtrees approach to unsupervised parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*.

Glenn Carroll and Eugene Charniak. 1992. Two experiments on learning probabilistic dependency grammars from corpora. In *WORKING NOTES OF THE WORKSHOP STATISTICALLY-BASED NLP TECHNIQUES*.

C.Goller and A.Kuchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *Proceedings of International Conference on Neural Networks (ICNN'96)*.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. *arXiv preprint arXiv:1905.03197*.

Andrew Drozdov, Pat Verga, Mohit Yadav, Mohit Iyyer, and Andrew McCallum. 2019. Unsupervised latent tree induction with deep inside-outside recursive autoencoders. In *North American Association for Computational Linguistics*.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proc. of NAACL*.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to parse and translate improves neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.

Phu Mon Htut, Kyunghyun Cho, and Samuel Bowman. 2018. Grammar induction with neural language models: An unusual replication. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Association for Computational Linguistics.

Yoon Kim, Chris Dyer, and Alexander Rush. 2019a. Compound probabilistic context-free grammars for grammar induction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2369–2385.

Yoon Kim, Alexander M. Rush, Lei Yu, Adhiguna Kuncoro, Chris Dyer, and Gábor Melis. 2019b. Unsupervised recurrent neural network grammars. *arXiv preprint arXiv:1904.03746*.

Dan Klein and Christopher D. Manning. 2002. A generative constituent-context model for improved grammar induction. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*.

Dan Klein and Christopher D. Manning. 2005. Natural language grammar induction with a generative constituent-context model. *Pattern Recogn.*

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. Linguistic knowledge and transferability of contextual representations. *arXiv preprint arXiv:1903.08855*.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Yikang Shen, Zhouhan Lin, Chin wei Huang, and Aaron Courville. 2018a. Neural language modeling by jointly learning syntax and lexicon. In *International Conference on Learning Representations*.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. 2018b. Ordered neurons: Integrating tree structures into recurrent neural networks. *arXiv preprint arXiv:1810.09536*.

Noah A. Smith and Jason Eisner. 2005. Guiding unsupervised grammar induction using contrastive estimation. In *In Proc. of IJCAI Workshop on Grammatical Inference Applications*.

Richard Socher, Cliff Chiung-Yu Lin, Andrew Y. Ng, and Christopher D. Manning. 2011. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*.

Emma Strubell, Patrick Verga, Daniel Andor, David I Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *EMNLP*.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Jesse Vig. 2019. Visualizing attention in transformer-based language representation models. *arXiv preprint arXiv:1904.02679*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*.

Wei Wu, Houfeng Wang, Tianyu Liu, and Shuming Ma. 2018. Phrase-level self-attention networks for universal sentence encoding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3729–3738.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, and et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Dani Yogatama, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Wang Ling. 2017. Learning to compose words into sentences with reinforcement learning. In *International Conference on Learning Representations*.

Poorya Zaremoodi and Gholamreza Haffari. 2018. Incorporating syntactic uncertainty in neural machine translation with a forest-to-sequence model. In *Proceedings of the 27th International Conference on Computational Linguistics*.

# Semantic Role Labeling with Iterative Structure Refinement

**Chunchuan Lyu**[1] **Shay B. Cohen**[1] **Ivan Titov**[1,2]
[1]ILCC, School of Informatics, University of Edinburgh
[2]ILLC, University of Amsterdam
chunchuan.lv@gmail.com scohen@inf.ed.ac.uk ititov@inf.ed.ac.uk

## Abstract

Modern state-of-the-art Semantic Role Labeling (SRL) methods rely on expressive sentence encoders (e.g., multi-layer LSTMs) but tend to model only local (if any) interactions between individual argument labeling decisions. This contrasts with earlier work and also with the intuition that the labels of individual arguments are strongly interdependent. We model interactions between argument labeling decisions through *iterative refinement*. Starting with an output produced by a factorized model, we iteratively refine it using a refinement network. Instead of modeling arbitrary interactions among roles and words, we encode prior knowledge about the SRL problem by designing a restricted network architecture capturing non-local interactions. This modeling choice prevents overfitting and results in an effective model, outperforming strong factorized baseline models on all 7 CoNLL-2009 languages, and achieving state-of-the-art results on 5 of them, including English.

## 1 Introduction

Semantic role labeling (SRL), originally introduced by Gildea and Jurafsky (2000), involves the prediction of predicate-argument structure, i.e., identification of arguments and their assignment to underlying *semantic roles*. Semantic-role representations have been shown to be beneficial in many NLP applications, including question answering (Shen and Lapata, 2007), information extraction (Christensen et al., 2011) and machine translation (Marcheggiani et al., 2018). In this work, we focus on dependency-based SRL (Hajič et al., 2009), a popular version of the task which involves identifying syntactic heads of arguments rather than marking entire argument spans (see the graph in red in Figure 1). Edges in the dependency graphs are annotated with semantic roles (e.g., A0:PLEASER) and the predicates are labeled with their senses from a given sense inventory (e.g., SATISFY.01 in the example).

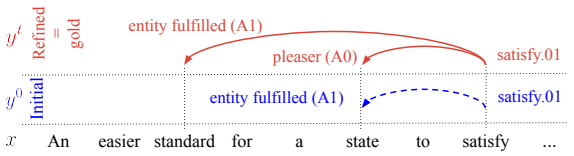


Figure 1: An example of structured refinement, the sentence fragment is from CoNLL-2009: the initial prediction by the factorized model in blue, the refined one (identical to the gold standard) in red.

Before the rise of deep learning methods, the most accurate SRL methods relied on modeling high-order interactions in the output space (e.g., between arguments or arguments and predicates) (Watanabe et al., 2010; Toutanova et al., 2008). Earlier neural methods can model such output interactions through a transition system, and achieve competitive performance (Henderson et al., 2013). However, current state-of-the-art SRL systems use powerful sentence encoders (e.g., layers of LSTMs (Li et al., 2018; He et al., 2017) or multi-head self-attention (Strubell et al., 2018)) and factorize over small fragments of the predicted structures. Specifically, most modern models process individual arguments and perform predicate disambiguation independently. The trend towards more factorizable models is not unique to dependency-based SRL but common for most structured prediction tasks in NLP (Kiperwasser and Goldberg, 2016; Dozat and Manning, 2017, 2018). The only major exception is language generation tasks, especially machine translation and language modeling, where larger amounts of text are typically used in training.

Powerful encoders, in principle, can capture long-distance dependencies and hence alleviate the need for modeling high-order interactions in



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1071–1082, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

the output. However, capturing these interactions in the encoder would require substantial amounts of data. Even if we have domain knowledge about likely interactions between components of the predicted graphs, it is hard to inject this knowledge in an encoder.

Consider the example in Figure 1. The argument *'state'* appears in the highly ambiguous syntactic position '[.] *to satisfy*'. All three core semantic roles of the predicate SATISFY.01 can in principle appear here: patient (A1:ENTITY_FULFILLED, as in '*a sweet tooth to satisfy'*), instrument (A2:METHOD, as in '*a little dessert to satisfy your sweet tooth*') and agent (A0: PLEASER, as in our actual example). The basic factorized model got it wrong, assigning A1 to the argument *'state'*. However, taking into account other arguments, the model can correct the label. The configuration 'A1 *to satisfy*' is more likely when an agent (A0) is present in the sentence. The lack of an agent boosts the score for the correct configuration 'A0 *to satisfy*'.

Our *iterative refinement* approach encodes the above intuition. In iterative refinement (Lee et al., 2018), a refinement network repeatedly takes previous output as input and produces its refined version. Formally, we have

$$y^{t+1} = \text{Refine}(x, y^t).$$

Naturally, such refinement strategy also requires an initial prediction $y^0$, which is produced by a ('base') factorized model.

Refinement strategies have been successful in machine translation (Lee et al., 2018; Novak et al., 2017; Xia et al., 2017; Hassan et al., 2018), but their effectiveness in other NLP tasks is yet to be demonstrated.[1] We conjecture that this discrepancy is due to differences in data availability. Given larger amounts of training data typically used in machine translation, their base models and refinement networks overfit to a lesser extent. Overfitting in (1) the base model and (2) the refinement network are both problematic. The first implies that either there are no mistakes in the base models in the training set or their distribution is very different from that in the test regime, so the training material for the inference networks ends up being misleading. The second naturally means that refinement will fail at test time. We address both these issues by designing restricted inference networks and adding a specific form of noise when training them.

Our structured refinement network is simple but encodes non-local dependencies. Specifically, it takes into account the information about the role distributions on the previous iteration aggregated over the entire sentences but not the information what the other arguments are. It is a coarse compressed representation of the prediction, yet it represents long-distance information not readily available within the factorized base model. While this is not the only possible design, we believe that the empirical gains from using this simple refinement network, demonstrate the viability of our general framework of iterative refinement with restricted inference networks. They also suggest that intuitions underlying declarative constraints used in early work on SRL (Punyakanok et al., 2008; Das et al., 2012) can be revisited but now encoded in a flexible soft way to provide induction biases for the refinement networks. We leave this for future work.

We consider the CoNLL-2009 dataset (Hajič et al., 2009). We start with a strong factorized baseline model, which already achieves state-of-the-art results on a subset of the languages. Then, using our structure refinement network, we improve on this baseline on all 7 CoNLL-2009 languages. The model achieves best-reported results in 5 languages, including English. We also observe improvements on out-of-domain test sets, confirming the robustness of our approach. We perform experiments demonstrating the importance of adding noise, and ablation studies showing the necessity of incorporating output interactions. Furthermore, we provide analysis on constraint violations and errors on the English test set.[2]

## 2 Related Work

Learning to refine predictions from neural structured prediction models has recently received significant attention. Our approach bears similarity to methods used in machine translation (Lee et al., 2018; Novak et al., 2017; Xia et al., 2017). All these methods refine a translated sentence produced by a seq2seq model with another seq2seq model. Among them, the deliberation networks

[1] See extra discussion and related work in section 2.

[2] The code and experiment settings can be accessed at https://github.com/ChunchuanLv/Iterative_Inference

by Xia et al. (2017) rely on BiLSTMs and improve initial predictions from an competitive baseline and obtain state-of-art-results on English-to-French translation. Later, it has been shown that the deliberation networks can improve translation when used within the Transformer framework (Hassan et al., 2018).

Certain approaches, not necessarily directly optimized for refinement, can nevertheless be regarded as iterative refinement methods. Structured prediction energy networks (SPENs) are trained to assign global energy scores to output structures, and the gradient descent is used during inference to minimize the global energy (Belanger and McCallum, 2016). As the gradient descent involves iterative optimization, its steps can be viewed as iterative refinement. In particular, Belanger et al. (2017) build a SPEN for SRL, but for the span-based formalism, not the dependency one we consider in this work. While they improve over their baseline model, their baseline model used multi-layer perceptron to encode local factors, thus the encoder power is limited. Moreover their refined model performs worse in the out-of-domain setting than their baseline model, indicating overfitting (Belanger et al., 2017).

In the follow-up work, Tu and Gimpel (2018, 2019) introduce inference networks to replace gradient descent. Their inference networks directly refine the output. Improvements over competitive baselines are reported on part-of-speech tagging, named entity recognition and CCG supertagging (Tu and Gimpel, 2019). However, their inference networks are distilling knowledge from a tractable linear-chain conditional random field (CRF) model. Thus, these methods do not provide direct performance gains. More importantly, the interactions captured in these models are likely local, as they learn to mimic Markov CRFs.

Denoising autoencoders (Vincent et al., 2008) can also be used to refine structure. Indeed, image segmentation can be improved through iterative inference with denoising autoencoders (Romero et al., 2017; Drozdzal et al., 2018). Their framework is very similar to ours, albeit we are working in a discrete domain. One other difference is that by using a convolutional architecture in the refinement network, they are still modeling only local interactions. At a more conceptual level, Bengio et al. (2013) argued that a denoising autoencoder should not be too robust to the input variations as to ignore the input. This indicates that we should not expect refinement networks to correct all the errors, even in theory, and hence, the refinement networks do not need to be particularly powerful.

Very recently, Wang et al. (2019) used high order statistical model for Semantic Dependency Parsing (Oepen et al., 2015), and obtain improvements over strong baseline using BiLSTM. They attempted loopy belief propagation and mean field variational inference for inference, and train the model end to end. Such inference steps are well motivated. This work is similar to energy network approach (Belanger and McCallum, 2016), while a global score function is provided, and approximate inference steps are used. Comparing to ours, the inference can also be regarded as iterative structure refinement. Yet, we do not provide a global score and directly try to model the refinement. In principle, our formalization should give us more liberty in terms of designing the refinement network.

## 3 Dependency Semantic Role Labeling

In this section, we introduce the notation and present our factorized baseline model.

### 3.1 Notation

In dependency SRL, for each sentence of length $n$, we have a sequence of words $\mathrm{w}$, dependency labels $\mathrm{dep}$, part-of-speech tags $\mathrm{pos}$, each being a discrete sequence of length $n$. To simplify notation, we consider one predicate at a time. We denote the number of roles by $r$, it includes the 'null' role, signifying that the corresponding word is not an argument of the predicate. Formally, $\mathrm{P} \in \Delta_{m-1}$ is the probability distribution over $m$ predicate senses, and $\Delta_{m-1}$ represents the corresponding probability simplex. We also have predicate sense embeddings $\Pi \in \mathbb{R}^{m \times d_\pi}$, and index $j$, throughout the discussion, refers to the position of the target predicate in the sentence. $\mathrm{R} \in \Delta_{r-1}^{n}$ is a matrix of size $n \times r$ such that each row sums to 1, corresponding to a probability distribution over roles. In particular $\mathrm{R}_{i,0}$ is the probability of $i$-th word not being an argument of the predicate.

We index role label and sense predictions from different refinement iterations ('time steps') with $t$, i.e. $\mathrm{P}^t$ and $\mathrm{R}^t$. The index $t$ ranges from 0 to $T$, and $\mathrm{P}^0$ and $\mathrm{R}^0$ denotes the predictions from the factorized baseline model. Details (e.g., hyperparameters) are provided in the appendix.

### 3.2 Factorized Model

Similarly to recent approaches to SRL and semantic graph parsing (He et al., 2017; Li et al., 2018; Dozat and Manning, 2018), our factorized baseline model starts with concatenated embeddings $x$. Then, we encode the sentence with a BiLSTM, further extract features with an MLP (multilayer perceptron) and apply a bi-affine classifier to the resulting features to label the words with roles. We also use a predicate-dependent linear layer for sense disambiguation.

More formally, we start with getting a sentence representation by concatenating embeddings. We have $x^{\mathrm{w}} \in \mathbb{R}^{n \times d_w}$, $x^{\mathrm{dep}} \in \mathbb{R}^{n \times d_\delta}$, $x^{\mathrm{pos}} \in \mathbb{R}^{n \times d_p}$ for words, dependency labels and part-of-speech tags, respectively. We concatenate them to form a sentence representation:

$$x = x^{\mathrm{w}} \circ x^{\mathrm{dep}} \circ x^{\mathrm{pos}} \in \mathbb{R}^{n \times d_x} \tag{1}$$

We further encode the sentence with a BiLSTM:

$$\mathrm{h} = \mathrm{BiLSTM}(x) \in \mathbb{R}^{n \times d_h} \tag{2}$$

From these context-aware word representations, we produce features for argument identification and role labeling that will be used by a bi-affine classifier. Note that, for every potential predicate-argument dependency (i.e. a candidate edge), we need to produce representations of both endpoints: the argument and the predicate 'sides'. For the argument side, $\mathrm{h}^{\rho_0}$ will be used to compute the logits for argument identification and $\mathrm{h}^{\rho_1}$ will be used for deciding on its role:

$$\mathrm{h}^{\rho_0} = \mathrm{MLP}(\mathrm{h}) \in \mathbb{R}^{n \times d_{\rho_0}} \tag{3}$$

$$\mathrm{h}^{\rho_1} = \mathrm{MLP}(\mathrm{h}) \in \mathbb{R}^{n \times d_{\rho_1}} \tag{4}$$

Similarly, for the predicate side, we also extract two representations $\mathrm{h}^{\varrho}_0$ and $\mathrm{h}^{\varrho_1}$ (recall that the predicate is at position $j$):

$$\mathrm{h}^{\varrho_0} = \mathrm{MLP}(\mathrm{h}_j) \in \mathbb{R}^{d_{\rho_0}} \tag{5}$$

$$\mathrm{h}^{\varrho_1} = \mathrm{MLP}(\mathrm{h}_j) \in \mathbb{R}^{d_{\rho_1}} \tag{6}$$

We then obtain logits $\mathrm{I}^{\rho_0}$ corresponding to the decision to label arguments as *null*, and logits $\mathrm{I}^{\rho_1}$ for other roles. So, we have:

$$\mathrm{I}^{\rho_0} = \mathrm{BiAffine}(\mathrm{h}^{\varrho_0}, \mathrm{h}^{\rho_0}) \in \mathbb{R}^{n} \tag{7}$$

$$\mathrm{I}^{\rho_1} = \mathrm{BiAffine}(\mathrm{h}^{\varrho_1}, \mathrm{h}^{\rho_1}) \in \mathbb{R}^{n \times (r-1)} \tag{8}$$

Unlike Dozat and Manning (2018), where argument identification and role labeling are trained with two losses,[3] we feed them together into a single softmax layer to compute the semantic-role distribution $\mathrm{R}^0$:

$$\mathrm{I}^{\alpha} = \mathrm{I}^{\rho_0} \circ \mathrm{I}^{\rho_1} \in \mathbb{R}^{n \times r} \tag{9}$$

$$\mathrm{R}^0 = \mathrm{Softmax}(\mathrm{I}^{\alpha}) \in \Delta^{n}_{r-1} \tag{10}$$

Now, for sense disambiguation, we need to extract yet another predicate representation $\mathrm{h}^{\pi}$:

$$\mathrm{h}^{\pi} = \mathrm{MLP}(\mathrm{h}_j) \in \mathbb{R}^{d_\pi} \tag{11}$$

In the formalism we use (PropBank), senses are predicate-specific, so we use predicate-specific sense embedding matrices $\Pi$. The matrix $\Pi$ acts as a linear layer before softmax:

$$\mathrm{I}^{\pi} = \Pi \cdot \mathrm{h}^{\pi} \in \mathbb{R}^{m} \tag{12}$$

$$\mathrm{P}^0 = \mathrm{Softmax}(\mathrm{I}^{\pi}) \in \Delta_{m-1} \tag{13}$$

This ends the description of our baseline model, which we also use to get initial predictions for iterative refinement.

## 4 Structured Refinement Network

In this section, we introduce the structured refinement network for dependency SRL. When doing refinement, it has access to the roles distribution $\mathrm{R}^t \in \Delta^{n}_{r-1}$ and the sense distribution $\mathrm{P}^t \in \Delta_{m-1}$ computed at the previous iteration (i.e. time $t$). In addition, it exploits the sentence representation $x \in \mathbb{R}^{n \times d_x}$. Our refinement network is limited and structured, in the sense that it only has access to a compressed version of the previous prediction, and the network itself is a simple MLP.

Similarly to our baseline model, we extract feature vectors $\mathrm{g}$ from input $x$ and further separately encode the argument representation $\mathrm{g}^{\alpha}$ and the predicate token representation $\mathrm{g}^{\pi}$:

$$\mathrm{g} = \mathrm{BiLSTM}(x) \in \mathbb{R}^{n \times d_h} \tag{14}$$

$$\mathrm{g}^{\alpha} = \mathrm{MLP}(\mathrm{g}) \in \mathbb{R}^{n \times d_g} \tag{15}$$

$$\mathrm{g}^{\pi} = \mathrm{MLP}(\mathrm{g}_j) \in \mathbb{R}^{d_g} \tag{16}$$

To simplify the notation, we omit indexing them by $t$, except for $\mathrm{R}^t$ and $\mathrm{P}^t$. We use two refinement networks, one for roles and another one for predicate senses.

[3] The separate processing of $\mathrm{I}^{\rho_0}$ and $\mathrm{I}^{\rho_1}$ rather than using a single MLP for all roles, including *null*, results in extra representation power allocated for the argument identification subtask.

### 4.1 Role Refinement Network

First, we describe our structured refinement network for role labeling. We use $i$ to index arguments. We obtain a compressed representation $\mathrm{o}_i$ used for refining $\mathrm{R}_i^t$ by summing up the probability mass for all roles, excluding the null role:

$$\mathrm{o}_{i,u} = \sum_{k \neq i} \mathrm{R}_{k,u}^t \in \mathbb{R} \quad (17)$$

$$\mathrm{o}_i = [\mathrm{o}_{i,u}] \in \mathbb{R}^{r-1} \quad (18)$$

Intuitively, $\mathrm{o}_i$ is the aggregation of all other roles being labeled by the current predicate. We concatenate $\mathrm{o}_i$ with feature vectors of the current argument $\mathrm{g}^\alpha$, predicate $\mathrm{g}^\pi$, the relaxed predicate sense embedding $\Pi^\intercal \cdot \mathrm{P}^t$ and the role probability itself ($\mathrm{R}_i$) to form the input to a two-layer network:

$$\mathrm{z}_i^\alpha = \mathrm{R}_i^t \circ \mathrm{o}_i \circ \mathrm{g}_i^\alpha \circ \mathrm{g}^\pi \circ (\Pi^\intercal \cdot \mathrm{P}^t) \quad (19)$$

$$\mathrm{z}_i^\alpha \in \mathbb{R}^{2r-1+2d_g+d_\pi}$$

$$\mathrm{M}_i^\alpha = W^\alpha \cdot \sigma(W_\alpha \cdot \mathrm{z}_i^\alpha) \in \mathbb{R}^r \quad (20)$$

$$\mathrm{M}^\alpha = [\mathrm{M}_{i,u}^\alpha] \in \mathbb{R}^{n \times r}, \quad (21)$$

where $\sigma$ is the logistic sigmoid function, $W_\alpha \in \mathbb{R}^{d_r \times (2r-1+2d_g+d_\pi)}$, $W^\alpha \in \mathbb{R}^{r \times d_r}$ are learned linear mappings. We obtain our refined logits $\mathrm{M}_i^\alpha$ for the $i$-th argument; $\mathrm{M}^\alpha$ refers to the stacked matrix of logits for all arguments. To obtain the refined role distribution, we add up $\mathrm{M}^\alpha$ and $\mathrm{I}^\alpha$ that we got from the baseline model, and follow that by a softmax layer:

$$\mathrm{R}^{t+1} = \mathrm{Softmax}(\mathrm{M}^\alpha + \mathrm{I}^\alpha) \in \Delta_{r-1}^n \quad (22)$$

### 4.2 Sense Refinement Network

To build a representation for sense disambiguation, we simply compute the probability mass for each role (excluding the null role) to obtain $\mathrm{r}^\pi$, and concatenate this with $\mathrm{g}^\pi$ and $\Pi^\intercal \cdot \mathrm{P}^t$:

$$\mathrm{r}^\pi = \sum_k \mathrm{R}_{k,1:}^t \in \mathbb{R}^{r-1} \quad (23)$$

$$\mathrm{z}^\pi = (\Pi^\intercal \cdot \mathrm{P}^t) \circ \mathrm{r}^\pi \circ \mathrm{g}^\pi \in \mathbb{R}^{r-1+d_g+d_\pi} \quad (24)$$

Differently from the role refinement network, sense prediction is predicate-specific. Therefore, we first map $\mathrm{z}^\pi$ to $\mathbb{R}^{d_\pi}$, and then take the inner product with the predicate-specific sense embeddings $\Pi$ to get the refined logits:

$$\mathrm{M}^\pi = \Pi \cdot W^\pi \cdot \sigma(W_\pi \cdot \mathrm{z}^\pi) \in \mathbb{R}^m \quad (25)$$

Similarly to role refinement, $\sigma$ is the logistic function, $W_\pi \in \mathbb{R}^{d_r \times (r-1+d_g+d_\pi)}$, $W^\pi \in \mathbb{R}^{m \times d_r}$ are learned linear mappings. Again, we combine the logits $\mathrm{M}^\pi$ and $\mathrm{I}^\pi$ before the softmax layer:

$$\mathrm{P}^{t+1} = \mathrm{Softmax}(\mathrm{M}^\pi + \mathrm{I}^\pi) \in \Delta_{m-1} \quad (26)$$

### 4.3 Weight Tying

Our refinement networks are similar to the denoising autoencoders (DAEs; Vincent et al. 2008), so we use the weight-tying technique popular with DAEs. We believe that the technique may be even more effective here as the amount of labeled data for SRL is lower than in many usual applications of DAEs. We tie $W_\alpha$ with a subset of $W^\alpha$ rows: specifically with the rows acting on $R_i^t$ in the computation of $M_i^\alpha$ (see equations 19 and 20). Similarly, we tie $W_\pi$ with the part of $W^\pi$ corresponding to $\Pi^\intercal \cdot \mathrm{P}^t$ (see equations 24 and 25). Formally,

$$W^\alpha = W_\alpha^\intercal[: r] \quad (27)$$

$$W^\pi = W_\pi^\intercal[: d_\pi] \quad (28)$$

where $W[: k]$ takes the first $k$ rows of matrix $W$.

### 4.4 Self Refinement

We describe a simpler version of the refinement network which we will use in experiments to test whether the improvements with the structured refinement network over the factorized baseline are genuinely coming from modeling interaction between arguments rather than from simply combining multiple classifiers. This simpler refinement network does not account for any interactions between arguments. Instead of equations 19 and 24, we have:

$$\mathrm{z}_i^\alpha = \mathrm{R}_i^t \circ \mathrm{g}_i^\alpha \circ \mathrm{g}^\pi \in \mathbb{R}^{r+2d_g+d_\pi} \quad (29)$$

$$\mathrm{z}^\pi = (\Pi^\intercal \cdot \mathrm{P}^t) \circ \mathrm{g}^\pi \in \mathbb{R}^{d_g+d_\pi} \quad (30)$$

Everything else is kept the same as in the full model, expect that the size of $W^\alpha$ and $W^\pi$ needs to be adjusted. We refer to this ablated network as the *self-refinement network*.

## 5 Training for Iterative Structure Refinement

In this section, we describe our training procedure.

### 5.1 Two-Stage Training

We have two models: the baseline model, producing the initial predictions, and the iterative refinement network, correcting them. While it is possible to train them jointly, we find joint training slow

to converge. Instead, we train the factorized baseline model first and then optimize the refinement networks while keeping the baseline model fixed.

### 5.2 Stochastic Training

Our baseline model overfits to the training set, and, if simply training on its output, our refinement network would learn to copy the base predictions. Instead, we perturb the baseline prediction during training. Naturally, we can add dropout (Srivastava et al., 2014) and recurrent dropout (Gal and Ghahramani, 2016) to our neural networks. However, for the smaller data set we use, we find this not sufficient. In particular, we use Gumbel-Softmax instead of Softmax. $\text{Gumbel-Softmax}(\mathrm{I}) = \text{Softmax}(\mathrm{I} + \lambda_g \epsilon)$, where the random variable $\epsilon$ is drawn from the standard Gumbel distribution (Maddison et al., 2017; Jang et al., 2017), and $\lambda_g$ is a hyper-parameter controlling decoding stochasticity.[4]

### 5.3 Loss for Iterative Refinement

Let us denote gold-standard labels for roles and predicates as $\mathrm{R}^*$ and $\mathrm{P}^*$. We use two separate losses $\mathcal{L}_{\text{base}}(\mathrm{R}^*, \mathrm{P}^*, x)$ and $\mathcal{L}_{\text{refine}}(\mathrm{R}^*, \mathrm{P}^*, x)$ for our two-stage training. We define losses for predictions from each refinement iteration and sum them up:

$$\mathcal{L}_{\text{base}}(\mathrm{R}^*, \mathrm{P}^*, x) = \mathcal{L}(\mathrm{R}^*, \mathrm{R}^0) + \mathcal{L}(\mathrm{P}^*, \mathrm{P}^0) \tag{31}$$

$$\mathcal{L}_{\text{refine}}(\mathrm{R}^*, \mathrm{P}^*, x) = \sum_{t=1}^{T} \mathcal{L}(\mathrm{R}^*, \mathrm{R}^t) + \mathcal{L}(\mathrm{P}^*, \mathrm{P}^t) \tag{32}$$

We adopt the Softmax-Margin loss (Gimpel and Smith, 2010; Blondel et al., 2019) for individual $\mathcal{L}$. Effectively, we subtract 1 from the logit of the gold label, and apply the cross entropy loss.

## 6 Experiments

**Datasets** We conduct experiments on CoNLL-2009 (Hajič et al., 2009) data set for all languages, including Catalan (Ca), Chinese (Zh), Czech (Cz), English (En), German (De), Japanese (Jp) and Spanish (Es). We use the predicted part-of-speech tags, dependency labels, and pre-identified predicate, provided with the dataset. The statistics of datasets are shown in Table 2.

[4] A more canonical way of controlling stochasticity is to use the temperature but we prefer not to scale the gradient.

**Hyperparameters** We use ELMo (Peters et al., 2018) for English, and FastText embeddings (Bojanowski et al., 2017; Grave et al., 2018) for all other languages. We train and run the refinement networks for two iterations. All other hyperparameters are the same for all languages, except BiLSTMs for English is larger than others.

**Training Details** Training the refinement network takes roughly 2 times more time than the baseline models, as it requires running BiLSTMs. The extra computation for the structured refinement network is minimal. For English, training the iterative refinement model for 1 epoch takes about 6 minutes on one 1080ti GPU. Adam is used as the optimizer (Kingma and Ba, 2015), with the learning rate of 3e-4. We use early stopping on the development set. We run 600 epochs for all baseline models, and 300 epochs for the refinement networks. Batch sizes are chosen from 32, 64, or 128 to maximize GPU memory usage. Our implementation is based on PyTorch and AllenNLP (Paszke et al., 2017; Gardner et al., 2018).

### 6.1 Results and Discussions

**Test Results** Results for all CoNLL-2009 languages on the standard (in-domain) datasets are presented in Table 1. We compare our best model to the best previous single model for the corresponding language (excluding ensemble ones). Most research has focused on English, but we include results of recent models which were evaluated on at least 3 languages. When compared to the previous models, both our models are very competitive, with the exception of German. On the German dataset, Mulcaire et al. (2018) also report a relatively weak result, when compared to Roth and Lapata (2016). The German dataset is the smallest one in terms of the number of predicates. Syntactic information used by Roth and Lapata (2016) may be very beneficial in this setting and may be the reason for this discrepancy. Our structured refinement approach improves over the best previous results on 5 out of 7 languages. Note that hyper-parameters of the refinement network are not tuned for individual languages, suggesting that the proposed method is robust and may be easy to apply to new languages and/or new base models. The only case where the refinement network was not effective is Chinese, where it achieved only a negligible improvement.

**Out-of-Domain** Results on the out-of-domain

| Model | Ca | Cz | De | En | Ja | Es | Zh | Avg. |
|---|---|---|---|---|---|---|---|---|
| Roth and Lapata (2016) | - | - | **80.10** | 86.7 | - | 80.20 | 79.4 | - |
| Marcheggiani et al. (2017) | - | 86.00 | - | 87.7 | - | 80.30 | 81.2 | - |
| Mulcaire et al. (2018)* | 79.45 | 85.14 | 69.97 | 87.24 | 76.00 | 77.32 | 81.89 | 79.57 |
| Previous best single model | 80.32 | 86.02 | **80.10** | 90.40 | 78.69 | 80.50 | **84.30** | 82.90 |
| Baseline model | 80.69 | 87.30 | 75.06 | 90.65 | 81.97 | 79.87 | 83.26 | 82.69 |
| Structured refinement | **80.91** | **87.62** | 75.87 | **90.99** | **82.54** | **80.53** | 83.31 | **83.11** |

Table 1: Labeled F1 score (including senses) for all languages on the CoNLL-2009 in-domain test set. For previous best result, Catalan is from Zhao et al. (2009), Japanese is from Watanabe et al. (2010), Czech is from Henderson et al. (2013), German and Spanish are from Roth and Lapata (2016), English is from Li et al. (2018) and Chinese is from Cai et al. (2018). We report the best testing results from Mulcaire et al. (2018).

| | #sent | #pred | #pred/#sent |
|---|---|---|---|
| Ca | 13200 | 37444 | 2.84 |
| Cz | 38727 | 414133 | 10.69 |
| De | 36020 | 17400 | 0.48 |
| En | 39279 | 179014 | 4.56 |
| Ja | 4393 | 25712 | 5.85 |
| Es | 14329 | 43828 | 3.06 |
| Zh | 22277 | 102827 | 4.62 |

Table 2: Number of sentences and predicates in training set of different languages.

testing sets are presented in Table 4.[5] We observe improvements from using refinement in all the cases. This shows that our refinement approach is robust against domain shift.

**Ablations** We report development set results in different settings in Table 3. Our full model performs 2 refinement iterations, uses weight tying, and the Gumbel noise.[6] We select the best configuration for each language to report the test set performance in Table 1 and Table 4.

As expected, weight tying is beneficial for lower-resource languages such as Catalan, Japanese and Spanish (see Table 2 for dataset characteristics). The Gumbel noise helps for all languages except for Czech and English, the two largest datasets. In particular, we observe almost no improvement on the Spanish dataset without using the Gumbel noise. We note relatively consistent but small gains from using 2 refinement iterations. The magnitude of the gains may be an artifact of us having the loss terms $\mathcal{L}(\mathrm{R}^*, \mathrm{R}^t)$ and $\mathcal{L}(\mathrm{P}^*, \mathrm{P}^t)$, encouraging not only the final (second), but also the first, iteration to produce accurate predictions. A potential alternative explanation is that our refinement network is restricted to simple interactions, resulting in the fixed point reachable in one step.

**Constraints Violation** We consider violation of unique core roles (U), continuation roles (C) and reference roles (R) constraints from Punyakanok et al. (2008); FitzGerald et al. (2015) in Table 6. U is violated if a core role (A0 - A5, AA) appears more than once; C is violated when the C-X role is not preceded by the X role (for some X); R is violated if R-X role does not appear. Our approach results in a large reduction in the uniqueness constraint violations. Our model slightly reduces the number of R violations, while He et al. (2017) reported that deterministically enforcing constraints is not helpful (albeit in span-based SRL). However learning those constraints in a soft way might be beneficial.

**Argument Interaction vs. No Argument Interaction** We compare the structured refinement network and the self-refinement network in Table 5. Both networks share the same hyperparameters. The structured refinement network consistently outperforms the self-refinement counterpart. This suggests that the refinement model benefits from accessing information about other arguments when doing refinement. In other words, modeling argument interaction appears genuinely useful.

**Improvement Decomposition** We report labeled role precision, recall and sense disambiguation accuracy in Table 7. Our structured refinement approach consistently improves over the baseline model in all metrics. While we cannot assert the improvements on all metrics are significant, this suggests that it learns some non-trivial interactions instead of merely learning to balance precision and

[5] Roth and Lapata (2016) has better in-domain testing score, but did not report the out-of-domain score.

[6] We set $\lambda_g^{\alpha} = 5$ for role and $\lambda_g^{\pi} = 50$ for sense, so that initial predictions contain around 20% errors.

| Model | Ca | Cz | De | En | Ja | Es | Zh | Avg. |
|---|---|---|---|---|---|---|---|---|
| Baseline | 81.69 | 88.43 | 73.97 | 89.60 | 82.96 | 80.49 | 85.27 | 83.20 |
| Full | **82.11** | 88.62 | 74.95 | 89.82 | **83.60** | **81.19** | **85.52** | **83.69** |
| 1 iteration | 82.07 | 88.62 | **75.07** | 89.93 | 83.49 | 81.03 | 85.47 | 83.40 |
| un-tied | 81.99 | 88.61 | 75.04 | 89.79 | 83.47 | 80.89 | 85.49 | 83.61 |
| no Gumbel | 82.07 | **88.71** | 74.62 | **90.08** | 83.33 | 80.55 | 85.42 | 83.54 |

Table 3: Labeled F1 score (including senses) for all languages on development set for different configurations.

| English | Test | Ood |
|---|---|---|
| Li et al. (2018) | 90.4 | 81.5 |
| Baseline | 90.65 | 81.98 |
| Structured Refinement | **90.99** | **82.18** |
| German | Test | Ood |
| Zhao et al. (2009) | **76.19** | **67.78** |
| Baseline | 75.06 | 65.25 |
| Structured Refinement | 75.87 | 65.69 |
| Czech | Test | Ood |
| Marcheggiani et al. (2017) | 86.0 | **87.2** |
| Baseline | 87.30 | 85.80 |
| Structured Refinement | **87.62** | 86.04 |

Table 4: Labeled F1 scores (including senses) on English, German, Czech in-domain and out-of-domain test sets; we chose the previous models achieving the best scores on the out-of-domain test sets.

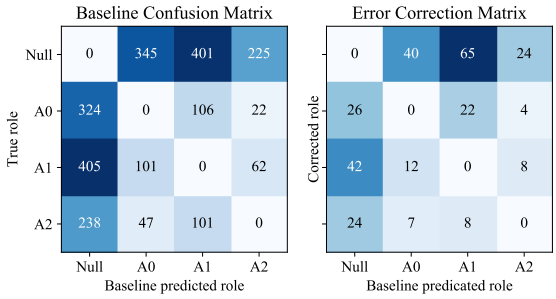


Figure 2: Confusion matrix for the baseline model, and a correction matrix where the errors were corrected by the refinement network. Only Null, A0, A1, A2 are presented here.

recall.

**Error Correction Analysis** We show the errors that the structured refinement network corrects in Figure 2. In the baseline confusion matrix, we see the errors are fairly balanced for all the roles we consider here. In the error correction matrix, the corrections are also fairly evenly distributed. Yet, this is not completely uniform. There is a tendency towards filtering out arguments rather than generating new ones.

## 7 Conclusions and Future Work

We propose the structured refinement network for dependency semantic role labeling. The structured refinement network corrects predictions made by a strong factorized baseline model while modeling interactions in the predicated structure. The resulting model achieves state-of-the-art results on 5 out of 7 languages in the CoNLL-2009 data set, and substantially outperforms the factorized model on all of these languages.

For the future work, the structured refinement network can be further improved. For example, we can take an inspiration from either declarative constraints used in the previous work (Punyakanok et al., 2008) or from literature on lexical semantics of verbs, studying patterns of event and argument realization (e.g., Levin 1993). Indeed, the unique role constraint as a declarative constraint is one of the motivation for the concurrent work on modeling argument interaction in SRL (Chen et al., 2019). That work relies on capsule networks (Sabour et al., 2017) and focuses primarily on enforcing the role uniqueness constraint.

The framework can be extended to other tasks. For example, in syntactic dependency parsing: the refinement network can rely on representations of grandparent nodes, siblings and children to propose a correction. In general, structure refinement networks should allow domain experts to incorporate prior knowledge about output dependencies and improve model performance.

## Acknowledgments

We thank the anonymous reviewers for their suggestions. The project was supported by the European Research Council (ERC StG BroadSem

| Model | Ca | Cz | De | En | Ja | Es | Zh | Avg. |
|---|---|---|---|---|---|---|---|---|
| Baseline Model | 80.69 | 87.30 | 75.06 | 90.65 | 81.97 | 79.87 | 83.26 | 82.69 |
| Self Refinement | 80.65 | 87.32 | 74.83 | 90.71 | 82.27 | 80.08 | **83.32** | 82.74 |
| Structured Refinement | **80.91** | **87.47** | **75.83** | **90.83** | **82.54** | **80.53** | 83.31 | **83.06** |

Table 5: Labeled F1 score (including senses) for all languages on the CoNLL-2009 in-domain test set.

| Model | U | C | R |
|---|---|---|---|
| Gold | 55 | 0 | 88 |
| Baseline | 301 | 2 | 114 |
| Structured Refinement | 142 | 2 | 111 |

Table 6: Unique core roles violations (U), continuation roles violations (C) and reference roles violations (R) on English in-domain test set.

| Model | RP | RR | Sense |
|---|---|---|---|
| Baseline | 88.1 | 88.3 | 96.2 |
| Structured Refinement | 88.7 | 88.5 | 96.3 |

Table 7: Labeled roles precision (RP), recall (RR) and sense disambiguation accuracy (Sense) on English in-domain test set.

678254), the Dutch National Science Foundation (NWO VIDI 639.022.518) and Bloomberg L.P.

## References

David Belanger and Andrew McCallum. 2016. Structured prediction energy networks. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 983–992. JMLR.org.

David Belanger, Bishan Yang, and Andrew McCallum. 2017. End-to-end learning for structured prediction energy networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 429–439. JMLR.org.

Yoshua Bengio, Li Yao, Guillaume Alain, and Pascal Vincent. 2013. Generalized denoising auto-encoders as generative models. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'13, pages 899–907, USA. Curran Associates Inc.

Mathieu Blondel, André FT Martins, and Vlad Niculae. 2019. Learning with fenchel-young losses. *arXiv preprint arXiv:1901.02324*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Jiaxun Cai, Shexia He, Zuchao Li, and Hai Zhao. 2018. A full end-to-end semantic role labeler, syntactic-agnostic over syntactic-aware? In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2753–2765, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Xinchi Chen, Chunchuan Lyu, and Ivan Titov. 2019. Capturing argument interaction in semantic role labeling with capsule networks. In *EMNLP*.

Janara Christensen, Mausam, Stephen Soderland, and Oren Etzioni. 2011. An analysis of open information extraction based on semantic role labeling. In *Proceedings of the Sixth International Conference on Knowledge Capture*, K-CAP '11, pages 113–120, New York, NY, USA. ACM.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2016. Fast and accurate deep network learning by exponential linear units (elus). In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Dipanjan Das, André FT Martins, and Noah A Smith. 2012. An exact dual decomposition algorithm for shallow semantic parsing with constraints. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 209–217. Association for Computational Linguistics.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Timothy Dozat and Christopher D. Manning. 2018. Simpler but more accurate semantic dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 484–490, Melbourne, Australia. Association for Computational Linguistics.

Michal Drozdzal, Gabriel Chartrand, Eugene Vorontsov, Mahsa Shakeri, Lisa Di Jorio, An Tang, Adriana Romero, Yoshua Bengio, Chris Pal, and Samuel Kadoury. 2018. Learning normalized inputs for iterative estimation in medical image segmentation. *Medical Image Analysis*, 44:1 – 13.

Nicholas FitzGerald, Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic role labeling with neural network factors. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 960–970, Lisbon, Portugal. Association for Computational Linguistics.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 1027–1035, USA. Curran Associates Inc.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew E. Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. *CoRR*, abs/1803.07640.

Daniel Gildea and Daniel Jurafsky. 2000. Automatic labeling of semantic roles. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 512–520, Hong Kong. Association for Computational Linguistics.

Kevin Gimpel and Noah A. Smith. 2010. Softmax-margin CRFs: Training log-linear models with cost functions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 733–736, Los Angeles, California. Association for Computational Linguistics.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the 11th Language Resources and Evaluation Conference*, Miyazaki, Japan. European Language Resource Association.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, CoNLL '09, pages 1–18, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, Shujie Liu, Tie-Yan Liu, Renqian Luo, Arul Menezes, Tao Qin, Frank Seide, Xu Tan, Fei Tian, Lijun Wu, Shuangzhi Wu, Yingce Xia, Dongdong Zhang, Zhirui Zhang, and Ming Zhou. 2018. Achieving human parity on automatic chinese to english news translation. *CoRR*, abs/1803.05567.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 473–483. Association for Computational Linguistics.

James Henderson, Paola Merlo, Ivan Titov, and Gabriele Musillo. 2013. Multilingual joint parsing of syntactic and semantic dependencies with a latent variable model. *Computational Linguistics*, 39(4):949–998.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1173–1182, Brussels, Belgium. Association for Computational Linguistics.

Beth Levin. 1993. *English verb classes and alternations: A preliminary investigation*. University of Chicago press.

Zuchao Li, Shexia He, Hai Zhao, Yiqing Zhang, Zhuosheng Zhang, Xi Zhou, and Xiang Zhou. 2018. Dependency or span, end-to-end uniform semantic role labeling. *CoRR*, abs/1901.05280.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The concrete distribution: A continuous relaxation of discrete random variables. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Diego Marcheggiani, Joost Bastings, and Ivan Titov. 2018. Exploiting semantics in neural machine translation with graph convolutional networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 486–492, New Orleans, Louisiana. Association for Computational Linguistics.

Diego Marcheggiani, Anton Frolov, and Ivan Titov. 2017. A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 411–420, Vancouver, Canada. Association for Computational Linguistics.

Phoebe Mulcaire, Swabha Swayamdipta, and Noah A. Smith. 2018. Polyglot semantic role labeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 667–672, Melbourne, Australia. Association for Computational Linguistics.

Roman Novak, Michael Auli, and David Grangier. 2017. Iterative refinement for machine translation. *CoRR*, abs/1610.06602.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajič, and Zdeňka Urešová. 2015. SemEval 2015 task 18: Broad-coverage semantic dependency parsing. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 915–926, Denver, Colorado. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *American Journal of Computational Linguistics*, 34(2):257–287.

Adriana Romero, Michal Drozdzal, Akram Erraqabi, Simon Jégou, and Yoshua Bengio. 2017. Image segmentation by iterative inference from conditional score estimation. *CoRR*, abs/1705.07450.

Michael Roth and Mirella Lapata. 2016. Neural semantic role labeling with dependency path embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1192–1202, Berlin, Germany. Association for Computational Linguistics.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. 2017. Dynamic routing between capsules. In *Advances in neural information processing systems*, pages 3856–3866.

Dan Shen and Mirella Lapata. 2007. Using semantic roles to improve question answering. In *EMNLP-CoNLL 2007, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, June 28-30, 2007, Prague, Czech Republic*, pages 12–21.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2377–2385.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5027–5038, Brussels, Belgium. Association for Computational Linguistics.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2008. A global joint model for semantic role labeling. *American Journal of Computational Linguistics*, 34(2):161–191.

Lifu Tu and Kevin Gimpel. 2018. Learning approximate inference networks for structured prediction. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Lifu Tu and Kevin Gimpel. 2019. Benchmarking approximate inference methods for neural structured prediction. *CoRR*, abs/1904.01138.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, pages 1096–1103.

Xinyu Wang, Jingxian Huang, and Kewei Tu. 2019. Second-order semantic dependency parsing with end-to-end neural networks. In *ACL*.

Yotaro Watanabe, Masayuki Asahara, and Yuji Matsumoto. 2010. A structured model for joint learning of argument roles and predicate senses. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 98–102, Uppsala, Sweden. Association for Computational Linguistics.

Yingce Xia, Fei Tian, Lijun Wu, Jianxin Lin, Tao Qin, Nenghai Yu, and Tie-Yan Liu. 2017. Deliberation networks: Sequence generation beyond one-pass decoding. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 1782–1792.

Hai Zhao, Wenliang Chen, Jun'ichi Kazama, Kiyotaka Uchimoto, and Kentaro Torisawa. 2009. Multilingual dependency learning: Exploiting rich features for tagging syntactic and semantic dependencies. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 61–66, Boulder, Colorado. Association for Computational Linguistics.

# Entity Projection via Machine Translation for Cross-Lingual NER

**Alankar Jain** **Bhargavi Paranjape** **Zachary C. Lipton**
Carnegie Mellon University
Pittsburgh, USA
{alankarjain91,bhargavi22294}@gmail.com, zlipton@cmu.edu

## Abstract

Although over 100 languages are supported by strong off-the-shelf machine translation systems, only a subset of them possess large annotated corpora for *named entity recognition*. Motivated by this fact, we leverage machine translation to improve annotation-projection approaches to cross-lingual named entity recognition. We propose a system that improves over prior entity-projection methods by: (a) leveraging machine translation systems twice: first for translating sentences and subsequently for translating entities; (b) matching entities based on orthographic and phonetic similarity; and (c) identifying matches based on distributional statistics derived from the dataset. Our approach improves upon current state-of-the-art methods for cross-lingual named entity recognition on 5 diverse languages by an average of 4.1 points. Further, our method achieves state-of-the-art $F_1$ scores for Armenian, outperforming even a monolingual model trained on Armenian source data.[1]

## 1 Introduction

While machine learning methods for various *Natural Language Processing (NLP)* tasks have progressed rapidly, the benefits accrue disproportionately among languages endowed with large annotated corpora. Owing to the dependence of state-of-the-art deep learning approaches on massive amounts of data, creating suitable datasets can be prohibitively expensive. This asymmetry between resource-rich and relatively under-resourced languages has inspired work on *cross-lingual* approaches that leverage annotated datasets from the former to build strong models for the latter.

This paper focuses on cross-lingual approaches to *Named Entity Recognition (NER)*, owing to NER's importance as a core component in *information retrieval* and *question answering* systems. Specifically, we focus on **medium-resource** languages. We define these to be languages for which although annotated NER corpora do not exist, off-the-shelf *Machine Translation (MT)* systems, such as Google Translate[2], do. We are motivated by the fact that although there are fewer than 50 languages for which large NER datasets (greater than 200k tokens) with gold annotations are publicly available[3], many more languages are supported by good-quality MT (**?**). Google Translate alone supports 103 languages[4], many of which have either no, or only small, NER datasets.

We address the setting where annotated corpora exist in the source (resource-rich) language—English in our experiments—but for the target (medium-resource) language, we can only afford to label a small validation set. We tackle this problem by first creating an unlabeled dataset in the target language by translating each sentence in the source dataset to the target language. For MT, we use Google Translate, motivated by its large coverage. Next, we annotate this dataset via *entity projection*—first aligning every entity in a source sentence with its counterpart in the corresponding target sentence (*entity alignment*) and then projecting the tags from source to target in the aligned entity pairs (*tag projection*). One consequence of relying on MT as opposed to word-by-word or phrase-by-phrase translation is that the entity projection step can be difficult, owing to the frequency with which original sentences and their translated counterparts are not word-for-word aligned.

Our proposed solution to this problem consists of (a) leveraging MT again for translating entities; (b) matching entities based on orthographic and

[1] Code for our paper can be found at: https://github.com/alankarj/cross_lingual_ner

[2] https://cloud.google.com/translate/

[3] http://damien.nouvels.net/resourcesen/corpora.html

[4] https://cloud.google.com/translate/docs/languages



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1083–1092,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

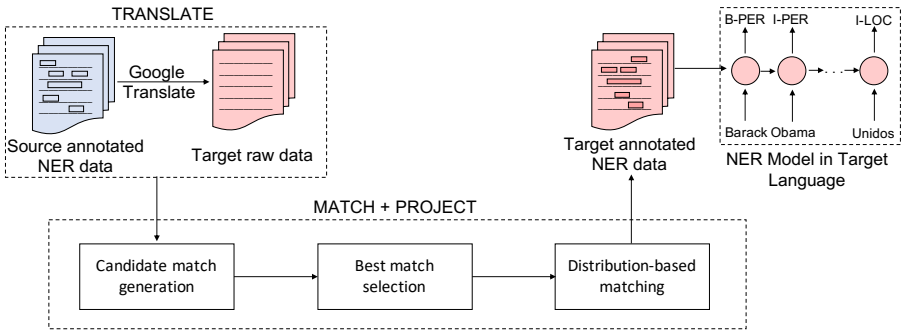


Figure 1: A schematic diagram representing the chief steps in our method.

phonetic similarity; and (c) identifying matches based on distributional statistics derived from the dataset. *Importantly, while our method depends on several matching heuristics, these techniques are remarkably portable across target languages, requiring the tuning of only two hyperparameters.* Our method achieves state-of-the art $F_1$ scores for cross-lingual NER for Spanish ($+1.1$ points), German ($+1.4$ points), and Chinese ($+5$ points) and beats state-of-the-art baselines on Hindi ($+2.1$ points) and Tamil ($+5$ points). Further, it achieves state-of-the-art $F_1$ scores for Armenian, a medium-resource language, beating a monolingual model trained on Armenian source data by $0.4$ points.

## 2 Related Work

Cross-lingual approaches have been applied to many NLP tasks, including part-of-speech tagging (**????**), parsing (**????**), and semantic role labeling (**????**). Prior cross-lingual NLP papers cleave roughly into two distinct approaches: *direct model transfer* and *annotation projection*.

### 2.1 Direct model transfer

These approaches apply models trained on the source language absent modification (to the model) to data from the target language by exploiting a shared representation for the two languages (**?????**). However, direct model transfer techniques face a problem when applied to markedly dissimilar languages: they lack of lexicalized (especially character-based) features, which are known to have predictive power for tasks such as NER. **?** provide evidence for this in the cross-lingual setting, comparing otherwise similar annotation projection approaches that differ in their use of lexicalized features.

### 2.2 Annotation projection

These approaches to cross-lingual NLP train a model in the target language. This requires first projecting annotations from the source data to the (unlabeled) target data. Many approaches in this category rely upon parallel corpora (**??????**), first annotating the source data using a trained model and then projecting the annotations. Only a few works explore the use of MT to first translate a gold annotated corpus to obtain a *synthetic* parallel corpus and then project annotations (**?**). **?** go in the opposite direction, translating target to source using Google Translate, annotating the translated source sentences using a trained NER system and then projecting annotations back.

When projecting annotations, one encounters the problem of word alignment. Most of the existing works (**???**) rely upon unsupervised alignment models from statistical MT literature, such as IBM Models 1-6 (**??**). Other works focus on low-resource settings (**??**) perform translation word-by-word or phrase-by-phrase, and thus do not need to perform word alignment. Several papers explore heuristics such as using Wikipedia links across languages to align entities (**???**), matching tokens based on their surface forms and transliterations either in an unsupervised manner (**??**) or as features in a supervised model trained on a small *seed* dataset (**?**). Many of these papers often rely on language-specific features (**?**) and evaluate their alignment methods on only a few languages.

To our knowledge, few works effectively use translation for annotation projection for NER, especially for medium-resource languages for which strong MT systems exist. Motivated by this research gap, we explore the use of MT systems for translating the dataset and for annotation projection and thus do not rely on parallel corpora. However, we demonstrate the efficacy of our projection

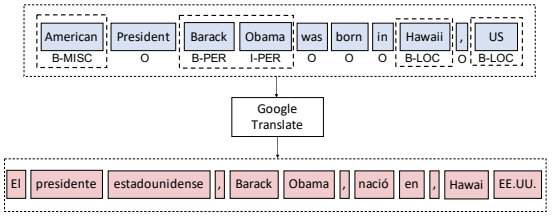


Figure 2: The *Translate* step of our method.

method in all three settings: (a) translation from source to target, (b) using parallel corpora and (c) translation from target to source.

## 3 The Translate-Match-Project Method

In our formulation, we are given an annotated NER corpus in the source language: $\mathcal{D}_A^S = \{(x^{Si}, y^{Si}) : i = 1, 2, ..., N\}$, where $x^{Si} = (x_1^{Si}, ..., x_{L^{Si}}^{Si})$ is the $i$th source sentence with $L^{Si}$ tokens and $y^{Si} = (y_1^{Si}, ..., y_{L^{Si}}^{Si})$ are the NER tags from a fixed tag set. We work with four tags: PER (person), ORG (organisation), LOC (location), and MISC (miscellaneous). We follow the commonly-used IOB (Inside Outside Beginning) tagging format (**?**). In our experiments, we work with English as the source language due to the availability of high-quality annotated corpora, e.g., CoNLL 2002 (**?**) and OntoNotes 4.0 (**?**). However, our method can easily be applied to any other resource-rich source language as well. See Figure 2 for an example of an annotated source sentence (blue), $(x^{Si}, y^{Si})$.

We are given a small labeled development set data (but no training data) in the target language $T$ for tuning hyperparameters. Our method, denoted *Translate-Match-Project (TMP)*, proceeds in three steps: First, we translate the annotated corpus in $S$ to $T$ using an off-the-shelf MT system (Google Translate). This results in an un-labeled dataset in the target language, $\mathcal{D}^T = \{x^{Ti} : i = 1, 2, ..., N\}$ (Figure 2); Second, we identify and tag all named entities in the translated target sentences by entity projection, which involves entity alignment and tag projection. We perform entity alignment by first constructing a set of potential matches in the target sentence for every entity in the source sentence (*candidate match generation*, Section 3.1) and then by selecting the best matching pairs of source and target entities (*best match selection*, Section 3.2); Third, after alignment, we project the tag type (PER, LOC, etc.) from the source to the target entity in every pair of aligned entities by adhering to the IOB tagging scheme in target. In Figure 1, we depict the complete pipeline.

### 3.1 Candidate match generation

To generate candidate matches for an entity in a source sentence, we construct a set of its potential translations and then find matches for each in the corresponding target sentence. We find these matches by token-level matching and then *concatenate matched tokens to obtain multi-token matches*. We drop the index $i$ below for ease of notation.

**Token-level matching** Consider a source entity $e^S = (x_j^S, ..., x_k^S)$. For every $e^S \in \mathcal{E}^S$, where $\mathcal{E}^S$ is the set of all entities in a source sentence, we obtain a set of potential translations in the target language, $\mathcal{T}(e^S)$, via MT. However, in some cases, translating a standalone entity produces a different translation from that which emerges when translating a full sentence. For example, Google Translate maps the source entity "UAE" to "Emiratos Árabes Unidos" in most sentences but the word-by-word translation is "EAU". Similarly, the (person) name "Tang" (e.g., "Mr. Tang") remains "Tang" in translated sentences, but is translated to "Espiga" (Spanish for "spike", synonymous with the English word "tang"). We address these problems by augmenting $\mathcal{T}(\cdot)$ with translations from *publicly available* bilingual lexicons ("UAE" translates to "Emiratos Árabes Unidos" in one of the lexicons we use) and retain a copy of the source entity ("Tang" will now find a match in the target sentence). Finally, $\mathcal{T}$("UAE") looks roughly like: {"EAU" [Google Translate], "UAE" [copy], "Emiratos Árabes Unidos" [lexicon]}. We note that lexicons exist for a large number of languages today[5]. However, we demonstrate that our method also works in absence of such lexicons in our case study for Armenian (Section 4.3).

Next, we tokenize each candidate translation in $\mathcal{T}(e^S)$ to obtain a set of translation tokens for $e^S$, $\mathcal{T}^w(e^S)$. For example, $\mathcal{T}^w$("UAE") = {"EAU", "UAE", "Emiratos", "Árabes", "Unidos"}. We do this to allow for soft token-level matches because we observed empirically that matching exact entity phrases might result in few matches. Next, we obtain a match for each hypothesis token $h \in \mathcal{T}^w(e^S)$ by matching it with each reference token $x_l^S \ \forall \ l \in \{1, ..., L^T\}$ in the target sentence $x^T = (x_1^T, ..., x_{L^T}^T)$ of length $L^T$. This match is carried out at (a) the orthographic (surface

[5]Panlex (**?**) has lexicons for 10k different languages.

form) level; and (b) the phonetic level, by matching transliterations in the International Phonetic Alphabet (IPA) of the two tokens. In either case, we look for the longest sequence of characters in $h$ that are an affix (prefix or suffix) of $x_l^T$. This soft *affix-matching heuristic* allows for inflection in morphologically-rich target languages. The (token-level) score for the match is given as follows:

$$s^w(h, x_l^T) = \min\left\{\frac{n_l}{L_h}, \frac{n_l}{L_{x_l^T}}\right\}$$

Here, $L_h$ and $L_{x_l^T}$ are the lengths (in characters) of the hypothesis and reference tokens and $n_l$ is the number of matching characters. We take minimum in order to enforce a stricter notion of fractional (soft) match. For example, the phrase "German first-time registrations..." [English] gets translated to "Los registros Alemanes por primera..." [Spanish]. Using our matching heuristic, "Alemán" $\in \mathcal{T}^w$("German") matches to the reference token "Alemanes" with a score of 0.5, since $n_l = 4$ ("Alem") and $L_{x_l^T} = 8 > L_h = 6$. Next, we define the matching (entity-level) score between a source entity $e^S$ and any target token $x_l^T$ as follows:

$$s^e(e^S, x_l^T) = \max_{h \in \mathcal{T}^w(e^S)} s^w(h, x_l^T)$$

Note that the token-level scores, $s^w$, include scores based on both orthographic and phonetic match, and thus, the entity-level scores, $s^e$, correspond to the best token-level match (orthographic or phonetic) between any hypothesis token $h \in \mathcal{T}^w(e^S)$ and a target token $x_l^T$. In Figure 3, we depict the token-level matching procedure. The score between the entity "American" and the target "estadounidense"(labeled 3 in the figure) is the maximum over matching scores between any token in {"americano", "american", "estadounidense"} and the target token ("estadounidense"), i.e., 1.0 (exact match) since the scores for the first two tokens in $\mathcal{T}^w(\cdot)$ are 0. Note some artifacts of token-level matching: (a) our matching heuristic currently handles prefixes or suffixes, but can potentially be extended with character edit distance for other types of affixes (e.g., circumfix) (b) every target token can match with multiple source entities (for e.g., "El" matches with both "American" and "US") and (c) some source entities might fail to find their true match ("US" fails to match with "EE.UU.", a possibly erroneous translation of "US" provided by Google Translate). Further, many matches are of very poor quality (especially those with stop words such as "El" and "en"). We address these issues and describe how to convert these token-level matches into spans to get multi-token target entities next.

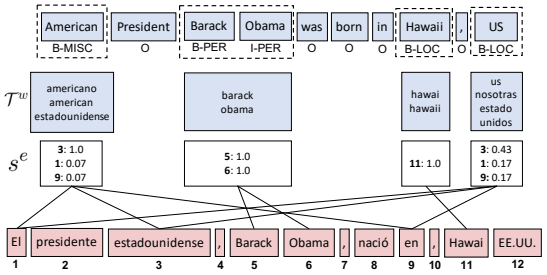


Figure 3: Token-level matching: Blue boxes in row-2 ($\mathcal{T}^w$) show sets of potential translations and white boxes in row-3 ($s^e$) show target tokens (numbered) each source entity can match with, with their scores.

**Span match generation** After token-level matching, we construct a list of potential entity spans in the target sentence that match with a given source entity $e^S$ by grouping adjacent target tokens for which a token-level matching score $s^e(e^S, \cdot)$ is above a threshold $\delta$ (to remove spurious matches). In other words, we construct the following set:

$$\mathcal{M}(e^S) = \{\text{span}(q, r) : s^e(e^S, x_u^T) \geq \delta \quad \forall\ q \leq u \leq r\}$$

Here, $\text{span}(q, r) = (x_q^T, ..., x_r^T)$ is the phrase spanning tokens indexed from $q$ to $r$ ($1 \leq q, r \leq L^T$) in the target sentence. Further, we require $\text{span}(q, r)$ to be maximal in the sense that $\forall\ q' < q$ and $\forall\ r' > r$, $\text{span}(q', r') \notin \mathcal{M}(\cdot)$, i.e., any target token before or after the span have at best a weak match ($s^e(e^S, \cdot) < \delta$) with $e^S$.

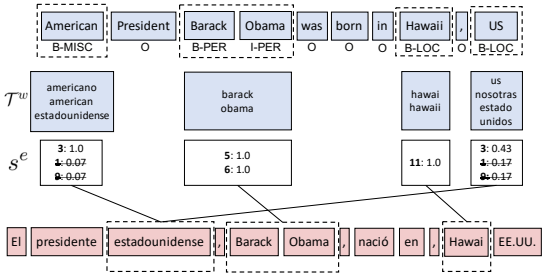


Figure 4: Span match generation: Adjacent target tokens with matching scores higher than a threshold (0.25 here) are concatenated to form span-level matches.

In our running example, choosing $\delta = 0.25$ results in the spans shown in Figure 4, eliminating spurious matches (with "El" and "en") and concatenating "Barack" and "Obama" in the target sentence. However, the token "estadounidense" is

still matched with two different source entities. We solve this problem in the next step.

### 3.2 Best match selection

For selecting the best matching pair of entities, we first expand the set of potential translations $\mathcal{T}(\cdot)$ to include all possible token-level permutations of the translations. We call this set $\mathcal{T}^p(\cdot)$. For example, $\mathcal{T}^p$("UAE") = {"EAU", "UAE", "Emiratos Árabes Unidos", "Emiratos Unidos Árabes", "Árabes Emiratos Unidos", ...}. Then, we greedily align $e^S$ with the target entity span from the set $\mathcal{M}(e^S)$, with the least character edit distance $d_E(\cdot,\cdot)$ from any translation in $T^p(e^S)$, i.e.,

$$e^T = \underset{\text{span}(\cdot,\cdot)\in\mathcal{M}(e^S)}{\text{argmin}} \; d_E(e^S, \text{span}(\cdot,\cdot))$$

In this manner, we form aligned entity pairs $(e^S, e^T)$, along which tags can then be projected. In our running example, since the edit distance between "estadounidense" (in $\mathcal{T}^p(\cdot)$) and the target single-token span "estadounidense" is 0, and is lower than that with "estado", we match "estadounidense" with "American", tagging it B-MISC.

### 3.3 Distribution-based matching

After selecting best matching pairs, there still remain some source entities that do not find any matching target entity ("US" in our example). These arise either due to significant differences between word-by-word and contextual sentence-level translations either due to literal (e.g., "West Bank" gets translated to "Cisjordania" in sentences and "Banco Oeste" otherwise) or possibly incorrect translations (e.g., "U.S." gets translated to "EE.UU." in sentences and "NOSOTRAS" otherwise).

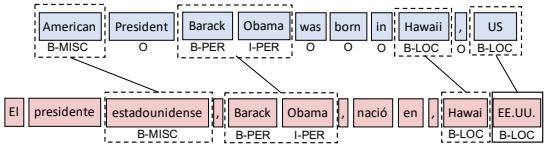


Figure 5: Final aligned pairs and projected tags.

We remedy this by exploiting corpus-level consistency in such discrepancies. For every unmatched source entity, we construct a set of top-$k$ potential matches ordered by their tf-idf (term frequency—inverse document frequency) scores, where tf is calculated over all sentences containing at least one unmatched entity and the idf score is calculated over the entire dataset to severely penalize commonly occurring tokens. Finally, we match each unmatched source entity with an unmatched span in its top-$k$ list with the highest tf-idf score. Figure 5 shows the final matches and tags.

## 4 Experimental Evaluation

**Data** In order to compare our method against benchmarks reported for prior approaches, we evaluate its performance on three European languages: Spanish (*es*), Dutch (*nl*) and German (*de*). Further, for a more extensive evaluation, we conduct additional experiments in an Indo-Aryan language (Hindi (*hi*)), a Dravidian language (Tamil (*ta*)), and Simplified Chinese (*zh*).

For all languages except Chinese, we use English NER training data from the CoNLL 2003 shared task (**?**) to translate into the target language. For Chinese, we sample the same number of sentences as in the CoNLL 2003 corpus (14,041) from the OntoNotes 4.0 (2012) dataset for English (**?**) to minimize distribution shift from Chinese development data. The development and test datasets for Spanish, Dutch and German are obtained from CoNLL 2002 (**?**) and CoNLL 2003 shared tasks. For Hindi and Tamil, we obtain the NER corpus from FIRE 2013 shared task[6]. Since this task doesn't provide the test dataset, we create our own splits: two-thirds for training and one-sixth each for development and test (to match with the proportions in the CoNLL dataset). For Chinese, we use the OntoNotes 4.0 development and test datasets. English, Spanish, Dutch and German contain PER, ORG, LOC and MISC tags, while Hindi, Tamil and Chinese were preprocessed to contain only PER, LOC and ORG tags. We use MUSE ground-truth bilingual lexicons[7] (gold lexicon) for augmenting the set of potential entity translations and use Epitran (**?**) for obtaining IPA transliterations.

**Baselines** We compare against four other annotation projection approaches that have achieved state-of-the-art results on some of our datasets. **?** (BWET) use a bilingual lexicon induced using monolingual corpora (**?**) to translate each source sentence word-by-word and then copy the corresponding NER tags using gold lexicons. As a ceiling for their method, **?** used Google Translate with fast-align (**?**) (fast-align), an unsupervised expectation maximization based algorithm, for entity alignment. Since this algorithm can produce multiple matches for a given source entity, we post-process

[6] http://au-kbc.org/nlp/NER-FIRE2013/

[7] https://github.com/facebookresearch/MUSE

| Method | Spanish | German | Dutch | Chinese | Hindi | Tamil | Average |
|---|---|---|---|---|---|---|---|
| TMP | **73.5 ± 0.4** | **61.5 ± 0.4** | 69.9 ± 0.4 | **50.1 ± 0.2** | **41.7 ± 1.3** | **33.8 ± 2.2** | **55.1 ± 0.8** |
| fast-align | 65.0 ± 1.2 | 60.1 ± 0.9 | 67.6 ± 0.7 | 45.1 ± 0.8 | 39.6 ± 1.1 | 28.8 ± 1.8 | 51.0 ± 1.1 |
| BWET | 72.4 ± 0.6 | 57.8 ± 0.1 | **70.4 ± 1.2** | 3.51 ± 0.8 | 26.6 ± 0.8 | 15.6 ± 0.9 | 48.5 ± 0.7 |
| Co-decoding | 65.1 | 58.5 | 65.4 | - | - | - | - |
| Polyglot-NER | 63.0 | - | 59.6 | - | - | - | - |
| Monolingual | 86.3 ± 0.4 | 78.2 ± 0.4 | 86.4 ± 0.2 | 68.59 ± 0.3 | 65.8 ± 1.2 | 51.8 ± 1.0 | 73.7 ± 0.6 |

Table 1: Test $F_1$ scores for our method (TMP), 4 cross-lingual baselines and a model trained on monolingual data.

the alignments produced by this algorithm and select the longest match and then project tags in the same way as our method. Our third baseline is **?** (Co-decoding), who use a co-decoding scheme on two different NER models. We also compare our method with Polyglot-NER (**?**) who use Wikipedia links to project entities. Finally, we also compare our performance with a model trained on annotated data in target language (Monolingual).

**NER Model** We use the state-of-the-art neural NER tagging model from (**?**) to train TMP and fast-align baseline for all languages. This model adds a self-attention layer to the character- and word-based BiLSTM + CRF model due to **?**. For each experiment, we run our models 5 times using different seeds and report the mean and standard deviation (as recommended by **?**) of $F_1$ measure.

**Hyperparameters** For the fast-align baseline, we tune their $\lambda$ parameter, which controls how much the model deviates from perfectly diagonal alignments, for each language separately. For TMP, we tune $\delta$, the score threshold and $k$, the number of top candidates selected in distribution-based matching. We use the same hyperparameters for the NER model as **?** for all our experiments.

### 4.1 Results

Our technique outperforms previous state-of-the-art cross-lingual methods on Spanish, German, Chinese, Hindi and Tamil and performs competitively on Dutch (Table 1). In particular, our method shows marked improvements over BWET, a word-by-word translation baseline, for languages such as German, Hindi, Tamil and Chinese that differ markedly in word ordering (with respect to English), demonstrating the impact of improved machine translation quality on final NER tagging accuracy. For more distant languages, word ordering can drastically affect the position of entities in a sentence, which can hurt performance on a test set in the target language. For instance, consider the Hindi word-by-word translation in Figure 6 (c), which is incoherent and violates the Subj-Obj-Verb ordering of Hindi. On languages that are closer to English, like Spanish and Dutch, the gains are comparatively modest, indicating that word order and quality MT is not critical for such languages.

We also show improvements over the fast-align baseline, which performs unsupervised word-level alignment over the full sequence. This can lead to alignment errors for named entities, which tend to be low-frequency words. Moreover, since fast-align allows for multiple target words to be aligned to a given source word, several noisy tags are added to the target sentence (see Figures 6 (a) and (b)).

### 4.2 Comparison of projection settings

Having established the performance of TMP as a method for cross-lingual NER, in this section, we conduct deeper experiments to evaluate the effectiveness of the matching (M) and projection (P) steps of TMP over the other projection baseline, fast-align. As mentioned in Section 2, there are variants of the annotation projection paradigm for cross-lingual NER that require an entity projection step, namely (i) reversing the direction of machine translation and (ii) using parallel corpora. We compare MP with fast-align for Spanish and Hindi languages under both these settings.

| Lang. | Method | Forward | Reverse | Parallel |
|---|---|---|---|---|
| **es** | MP | **73.5 ± 0.4** | 65.3 | 61.2 ± 1.2 |
| | fast-align | 65.0 ± 1.2 | 57.8 | 39.3 ± 0.5 |
| **hi** | MP | 41.7 ± 1.3 | 47.7 | **52.8 ± 1.4** |
| | fast-align | 39.6 ± 1.1 | 34.3 | 51.8 ± 1.5 |

Table 2: Performance of MP and fast-align on Forward, Reverse and Parallel settings in terms of $F_1$.

**Reversing the direction of translation** In this setting, we translate the target test set into the source language using Google Translate and then use the NER tagger with state-of-the-art results

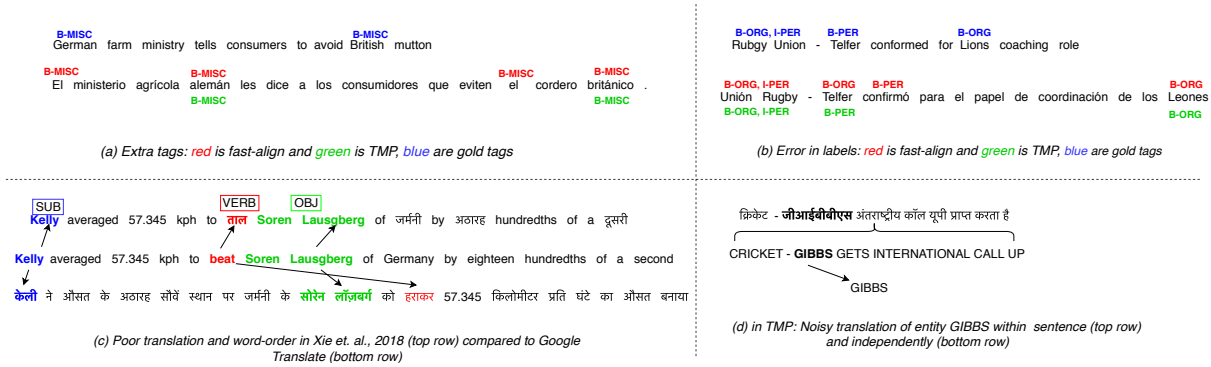


Figure 6: Examples of different errors (details in individual captions).

*Flair*[8] to tag entities in the translated English sentences. Finally, we employ MP/fast-align to project the tagged entities back to the target sentence. As shown in table 2, MP outperforms fast-align for both Spanish and Hindi and performs better than the forward direction translation for Hindi. This can be attributed to (a) the inherent difficulty of NER tagging in Hindi, which is morphologically richer than English and (b) the superior quality of the English NER model.

**Parallel corpora** In order to remove translation errors while evaluating TMP and fast-align, we experiment with parallel corpora. For English-Spanish, we use the Europarl corpus (**?**) and for English-Hindi, the IIT Bombay parallel corpus (**?**). We again use *Flair* to obtain NER tags in English, which are then projected to their corresponding target sentences to generate a training dataset, which is used to train an NER model in the target language. To minimize confounding variables, we sample 14k (same as CoNLL) high quality tagged sentences (average confidence score $> 0.9$). Results in Table 2 show that MP once again outperforms fast-align. Further, it performs better than Forward for Hindi by a significant margin possibly because the chosen parallel corpus is closer in time period to the test set, thereby reducing distribution shift.

### 4.3 Case study: Armenian

So far, we have only evaluated the performance of our method on languages for which large or moderately-sized gold annotated corpora already exist that provide an upper-bound for cross-lingual NER methods. Here, we evaluate our method on a true medium-resource language, Armenian. Recently, **?** introduced a ground truth test corpus for Armenian along with a train corpus with silver annotations extracted from Wikipedia. This test dataset is comprised of 2566 sentences (53k tokens) from political, sports, local and world news between August 2012 and July 2018. Since the English CoNLL 2003 dataset contains sentences nearly two decades older, we expect to see significant distribution shift if we follow TMP (Forward approach). Further, we are not aware of any large English-Armenian parallel corpora. So, we choose the Reverse paradigm for this problem. We achieve an $F_1$ score of 62.6, which is significantly higher than that achieved by fast-align (44.8). Further, this is 0.4 points higher than the current state-of-the-art model trained on over 160k tokens of Armenian. Note that our model does not make use of any external resources for Armenian (gold lexicons, Epitran, etc.) other than an MT system. This provides evidence towards our proposed approach being an effective and generalizable cross-lingual NER method that can be used for rapid deployment to new languages.

## 5 Analysis

**Measuring alignment accuracy** Since we do not possess ground truth word alignments for the "synthetic" parallel corpus generated through translation, we rely on heuristics to measure the accuracy of alignments. We measure the annotation *miss rate* among target sentences with equal or fewer tagged entities as compared to source. We also calculate the *excess rate*, representing the fraction of excess entities among sentences with more tagged entities. Both methods perform similarly in terms of miss rate, 0.79 % (MP) vs 0.83 % (fast-align) on Spanish and 3.96 % (MP) vs 3.48 % (fast-align) on Hindi. However, fast-align seems to add more noisy annotations as compared to MP, with

[8] https://github.com/zalandoresearch/flair

higher excess rates for both Spanish (8.29 % vs 0.49 %) and Hindi (6.35 % vs 2.20 %). A representative illustration of these noisy tags is shown in Figure 6 (a). where fast-align tags frequent words like "El", "de", "en" as entities. To offer a more fine-grained evaluation of alignment performance, we manually annotate 100 examples from the translated Spanish and Hindi training data and calculate precision, recall and $F_1$ score. MP outperforms fast-align for both the languages (Table 3).

| Lang. | Method | Precision | Recall | $F_1$ |
|---|---|---|---|---|
| **es** | MP | **96.2** | **96.7** | **96.4** |
| | fast-align | 84.6 | 87.4 | 85.9 |
| **hi** | MP | **87.4** | **77.6** | **82.2** |
| | fast-align | 82.0 | 76.8 | 79.3 |

Table 3: Alignment performance on 100 sentences.

**Ablation of features for alignment** We also conduct an ablation study (Table 4) to understand the sources of our gains beyond a base model that uses translations only from Google Translate and orthographic affix matching. To this base model, we successively add various features of our method: phonetic matching, exact copy translations, gold lexicons and finally distribution-based alignment (dist) of remaining entities. For both languages, we observe that every additional feature improves the performance of tagging, with the most important features being phonetic matching for Spanish and use of gold lexicons for Hindi. Interestingly, addition of phonetic matching hurts Hindi because of the low value of the threshold ($\delta = 0.25$), which results in spurious matches due to phonetic matching. In Table 4, we also see that the number of entities tagged (as a fraction of total entities) increase with the introduction of almost every feature (however, all matches might not be correct). This underscores the correlation between quality of entity alignment and performance on the downstream tagging task.

| Model | es | | hi | |
|---|---|---|---|---|
| | % **Entities** | $F_1$ | % **Entities** | $F_1$ |
| Base model | 91.8 | 67.8 | 77.9 | 37.7 |
| +phonetic | 93.7 | 71.4 | 83.0 | 35.0 |
| +copy | 97.2 | 72.2 | 85.4 | 37.6 |
| +gold | 98.3 | 73.3 | 88.5 | 42.0 |
| +dist | 99.9 | 74.2 | 94.9 | 43.4 |

Table 4: Ablation study for Spanish and Hindi

**Sources of errors in TMP** We also analyze mistakes made by TMP in aligning entities. Many false negative errors can be traced back to a high threshold $\delta$, resulting in an empty set of candidate matches. Errors also arise due to noise and variation introduced in the contextual sentence level translation of a word (Figure 6 (c) where GIBBS is interpreted as an acronym, (d) where MEDVEDEV is mistranslated). This causes discrepancies between translations of standalone entities and those in context, thereby, causing TMP to not find a match. However, these errors can be reduced as off-the-shelf MT systems continue to improve.

## 6 Conclusion

In this paper, we tackled the problem of entity projection for cross-lingual NER. Our proposed method leverages MT for translating entities, matches entities based on orthographic and phonetic similarity, and identifies matches based on distributional statistics derived from the dataset to achieve state-of-the-art results for cross-lingual NER on a diverse set of languages. Further, our method beats state-of-the-art monolingual baseline for Armenian, an actual medium-resource language (off-the-shelf translation systems exist, but large-scale NER corpora do not). In the future, we would like to explore ways to extend our method to languages not supported by Google Translate through the use of pivot languages.

While dependence on MT restricts our approach to languages covered by off-the-shelf MT systems, these systems continue to improve in coverage and quality, outpacing the availability of large-scale corpora for a variety of other tasks. Moreover as translation quality improves, approaches like ours are poised to benefit. Finally, although our method beats state-of-the-art baselines, not surprisingly, it falls short of NER models trained on large monolingual corpora. We suspect that a significant portion of this degradation is due to distribution shift (as evidenced by improvement in Hindi $F_1$ in Parallel regime). Thus one promising route to improving our models might be to incorporate domain adaptation techniques, which aim to build classifiers robust to various forms of distribution shift.

## References

Rami Al-Rfou, Vivek Kulkarni, Bryan Perozzi, and Steven Skiena. 2015. Polyglot-ner: Massive multi-

lingual named entity recognition. In *International Conference on Data Mining (ICDM)*.

Akash Bharadwaj, David Mortensen, Chris Dyer, and Jaime Carbonell. 2016. Phonologically aware neural model for named entity recognition in low resource transfer settings. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.

Aditi Chaudhary, Chunting Zhou, Lori Levin, Graham Neubig, David R Mortensen, and Jaime G Carbonell. 2018. Adapting word embeddings to new languages with morphological and phonological subword representations. *arXiv preprint arXiv:1808.09500*.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Association for Computational Linguistics (ACL)*.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *North American Chapter of the Association for Computational Linguistics (NAACL)*.

Maud Ehrmann, Marco Turchi, and Ralf Steinberger. 2011. Building a multilingual named entity-annotated corpus using annotation projection. In *Proceedings of the International Conference Recent Advances in Natural Language Processing 2011*, pages 118–124.

Donghui Feng, Yajuan Lv, and Ming Zhou. 2004. A new approach for english-chinese named entity alignment. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*.

Ruiji Fu, Bing Qin, and Ting Liu. 2011. Generating chinese named entity data from a parallel corpus. In *International Joint Conference on Natural Language Processing (IJCNLP)*.

Kuzman Ganchev, Jennifer Gillenwater, and Ben Taskar. 2009. Dependency grammar induction via bitext projection constraints. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 369–377. Association for Computational Linguistics.

Tsolak Ghukasyan, Garnik Davtyan, Karen Avetisyan, and Ivan Andrianov. 2018. pioner: Datasets and baselines for armenian named entity recognition. *arXiv preprint arXiv:1810.08699*.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural language engineering*, 11(3):311–325.

David Kamholz, Jonathan Pool, and Susan M Colowick. 2014. Panlex: Building a resource for panlingual lexical translation. In *International Conference on Language Resources and Evaluation (LREC)*.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86.

Mikhail Kozhevnikov and Ivan Titov. 2013. Cross-lingual transfer of semantic role labeling models. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1190–1200.

Mikhail Kozhevnikov and Ivan Titov. 2014. Cross-lingual model transfer using feature representation projection. In *Association for Computational Linguistics (ACL)*.

Anoop Kunchukuttan, Pratik Mehta, and Pushpak Bhattacharyya. 2017. The iit bombay english-hindi parallel corpus. *arXiv preprint arXiv:1710.02855*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Association for Computational Linguistics (ACL)*.

Stephen Mayhew, Chen-Tse Tsai, and Dan Roth. 2017. Cheap translation for cross-lingual named entity recognition. In *Empirical Methods in Natural Language Processing (EMNLP)*.

David R. Mortensen, Siddharth Dalmia, and Patrick Littell. 2018. Epitran: Precision G2P for many languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Paris, France. European Language Resources Association (ELRA).

Jian Ni, Georgiana Dinu, and Radu Florian. 2017. Weakly supervised cross-lingual named entity recognition via effective annotation and representation projection. In *Association for Computational Linguistics (ACL)*.

Joel Nothman, Nicky Ringland, Will Radford, Tara Murphy, and James R Curran. 2013. Learning multilingual named entity recognition from wikipedia. *Artificial Intelligence*, 194:151–175.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational linguistics*, 29(1):19–51.

Sebastian Padó and Mirella Lapata. 2009. Cross-lingual annotation projection for semantic roles. *Journal of Artificial Intelligence Research (JAIR)*, 36:307–340.

Lance A Ramshaw and Mitchell P Marcus. 1999. Text chunking using transformation-based learning. In *Natural language processing using very large corpora*, pages 157–176. Springer.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging. *arXiv preprint arXiv:1707.09861*.

Alexander E Richman and Patrick Schone. 2008. Mining wiki resources for multilingual named entity recognition. In *Association for Computational Linguistics (ACL)*.

Doaa Samy, Antonio Moreno, and Jose M Guirao. 2005. A proposal for an arabic named entity tagger leveraging a parallel corpus. In *International Conference RANLP, Borovets, Bulgaria*, pages 459–465.

Erik F. Tjong Kim Sang. 2002. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *COLING-02: The 6th Conference on Natural Language Learning (CoNLL)*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Conference on Natural Language Learning at HLT-NAACL*.

Rushin Shah, Bo Lin, Anatole Gershman, and Robert Frederking. 2010. Synergy: a named entity recognition system for resource-scarce languages such as swahili using online machine translation. In *Proceedings of the Second Workshop on African Language Technology (AfLaT 2010)*, pages 21–26.

David A Smith and Jason Eisner. 2009. Parser adaptation and projection with quasi-synchronous grammar features. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Oscar Täckström, Dipanjan Das, Slav Petrov, Ryan McDonald, and Joakim Nivre. 2013. Token and type constraints for cross-lingual part-of-speech tagging. *Transactions of the Association for Computational Linguistics (TACL)*, 1:1–12.

Oscar Täckström, Ryan McDonald, and Jakob Uszkoreit. 2012. Cross-lingual word clusters for direct transfer of linguistic structure. In *Proceedings of the 2012 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pages 477–487. Association for Computational Linguistics.

Jörg Tiedemann, Željko Agić, and Joakim Nivre. 2014. Treebank translation for cross-lingual parser induction. In *Eighteenth Conference on Computational Natural Language Learning (CoNLL 2014)*.

Sara Tonelli and Emanuele Pianta. 2008. Frame information transfer from english to italian. In *International Conference on Language Resources and Evaluation (LREC)*.

Ralph Weischedel, Sameer Pradhan, Lance Ramshaw, Martha Palmer, Nianwen Xue, Mitchell Marcus, Ann Taylor, Craig Greenberg, Eduard Hovy, Robert Belvin, et al. 2011. Ontonotes release 4.0. *LDC2011T03, Philadelphia, Penn.: Linguistic Data Consortium*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Chenhai Xi and Rebecca Hwa. 2005. A backoff model for bootstrapping resources for non-english languages. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Jiateng Xie, Zhilin Yang, Graham Neubig, Noah A. Smith, and Jaime Carbonell. 2018. Neural cross-lingual named entity recognition with minimal resources. In *Empirical Methods in Natural Language Processing (EMNLP)*.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of the first international conference on Human language technology research*. Association for Computational Linguistics.

Daniel Zeman and Philip Resnik. 2008. Cross-language parser adaptation between related languages. In *IJCNLP-08 Workshop on NLP for Less Privileged Languages*.

# A Bayesian Approach for Sequence Tagging with Crowds

**Edwin Simpson and Iryna Gurevych**
Ubiquitous Knowledge Processing Lab
Department of Computer Science
Technische Universität Darmstadt
https://www.ukp.tu-darmstadt.de
{simpson,gurevych}@ukp.informatik.tu-darmstadt.de

## Abstract

Current methods for sequence tagging, a core task in NLP, are data hungry, which motivates the use of crowdsourcing as a cheap way to obtain labelled data. However, annotators are often unreliable and current aggregation methods cannot capture common types of span annotation errors. To address this, we propose a Bayesian method for aggregating sequence tags that reduces errors by modelling sequential dependencies between the annotations as well as the ground-truth labels. By taking a Bayesian approach, we account for uncertainty in the model due to both annotator errors and the lack of data for modelling annotators who complete few tasks. We evaluate our model on crowdsourced data for named entity recognition, information extraction and argument mining, showing that our sequential model outperforms the previous state of the art. We also find that our approach can reduce crowdsourcing costs through more effective active learning, as it better captures uncertainty in the sequence labels when there are few annotations.

## 1 Introduction

Current methods for *sequence tagging*, a core task in NLP, use deep neural networks that require tens of thousands of labelled documents for training (Ma and Hovy, 2016; Lample et al., 2016). This presents a challenge when facing new domains or tasks, where obtaining labels is often time-consuming or costly. Labelled data can be obtained cheaply by crowdsourcing, in which large numbers of untrained workers annotate documents instead of more expensive experts. For sequence tagging, this results in multiple sequences of unreliable labels for each document.

Probabilistic methods for aggregating crowdsourced data have been shown to be more accurate than simple heuristics such as majority voting (Raykar et al., 2010; Sheshadri and Lease, 2013; Rodrigues et al., 2013; Hovy et al., 2013). However, existing methods for aggregating sequence labels cannot model dependencies between the annotators' labels (Rodrigues et al., 2014; Nguyen et al., 2017) and hence do not account for their effect on annotator noise and bias. In this paper, we remedy this by proposing a sequential annotator model and applying it to tasks that follow a *beginning, inside, outside* (*BIO*) scheme, in which the first token in a span of type 'x' is labelled 'B-x', subsequent tokens are labelled 'I-x', and tokens outside spans are labelled 'O'.

When learning from noisy or small datasets, commonly-used methods based on maximum likelihood estimation may produce over-confident predictions (Xiong et al., 2011; Srivastava et al., 2014). In contrast, Bayesian inference accounts for model uncertainty when making predictions. Unlike alternative methods that optimize the values for model parameters, Bayesian inference integrates over all possible values of a parameter, weighted by a prior distribution that captures background knowledge. The resulting posterior probabilities improve downstream decision making as they include the probability of errors due to a lack of knowledge. For example, during active learning, posterior probabilities assist with selecting the most informative data points (Settles, 2010).

In this paper, we develop *Bayesian sequence combination (BSC)*, building on prior work that has demonstrated the advantages of Bayesian inference for aggregating unreliable classifications (Kim and Ghahramani, 2012; Simpson et al., 2013; Felt et al., 2016; Paun et al., 2018). BSC is the first fully-Bayesian method for aggregating sequence labels from multiple annotators. As a core component of BSC, we also introduce the *sequential confusion matrix (seq)*, a probabilistic model



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1093–1104,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

of annotator noise and bias, which goes beyond previous work by modelling sequential dependencies between annotators' labels. Further contributions include a theoretical comparison of the probabilistic models of annotator noise and bias, and an empirical evaluation on three sequence labelling tasks, in which *BSC* with *seq* consistently outperforms the previous state of the art. We make all of our code and data freely available[1].

## 2 Related Work

Sheshadri and Lease (2013) benchmarked several aggregation models for non-sequential classifications, obtaining the most consistent performance from that of Raykar et al. (2010), who model the reliability of individual annotators using probabilistic confusion matrices, as proposed by Dawid and Skene (1979). Simpson et al. (2013) showed that a Bayesian variant of Dawid and Skene's model can outperform maximum likelihood approaches and simple heuristics when combining crowds of image annotators. This Bayesian variant, independent Bayesian classifier combination (*IBCC*) (Kim and Ghahramani, 2012), was originally used to combine ensembles of automated classifiers rather than human annotators. While traditional ensemble methods such as boosting focus on how to generate new classifiers (Dietterich, 2000), IBCC is concerned with modelling the reliability of each classifier in a given set of classifiers. To reduce the number of parameters in multi-class problems, Hovy et al. (2013) proposed *MACE*, and showed that it performed better under a Bayesian treatment on NLP tasks. Paun et al. (2018) further illustrated the advantages of Bayesian models of annotator ability on NLP classification tasks with different levels of annotation sparsity and noise.

We expand this previous work by detailing the relationships between several annotator models and extending them to sequential classification. Here we focus on the core annotator representation, rather than extensions for clustering annotators (Venanzi et al., 2014; Moreno et al., 2015), modeling their dynamics (Simpson et al., 2013), adapting to task difficulty (Whitehill et al., 2009; Bachrach et al., 2012), or time spent by annotators (Venanzi et al., 2016).

Methods for aggregating sequence labels include *CRF-MA* (Rodrigues et al., 2014), a CRF-based model that assumes only one annotator is correct for any given label. Recently, Nguyen et al. (2017) proposed a hidden Markov model (HMM) approach that outperformed CRF-MA, called *HMM-crowd*. Both CRF-MA and HMM-crowd use simpler annotator models than Dawid and Skene (1979) that do not capture the effect of sequential dependencies on annotator reliability. Neither CRF-MA nor HMM-crowd use a fully Bayesian approach, which has been shown to be more effective for handling uncertainty due to noise in crowdsourced data for non-sequential classification (Kim and Ghahramani, 2012; Simpson et al., 2013; Venanzi et al., 2014; Moreno et al., 2015). In this paper, we develop a sequential annotator model and an approximate Bayesian method for aggregating sequence labels.

## 3 Modeling Sequential Annotators

When combining multiple annotators with varying skill levels, we can improve performance by modelling their individual noise and bias using a probabilistic model. Here, we describe several models that do not consider dependencies between annotations in a sequence, before defining *seq*, a new extension that captures sequential dependencies. Probabilistic annotator models each define a different function, $A$, for the likelihood that the annotator chooses label $c_\tau$ given the true label $t_\tau$, for the $\tau$th token in a sequence.

**Accuracy model (acc)**: the basis of several previous methods (Donmez et al., 2010; Rodrigues et al., 2013), *acc* uses a single parameter for each annotator's accuracy, $\pi$:

$$A = p(c_\tau = i | t_\tau = j, \pi) = \begin{Bmatrix} \pi & \text{where } i = j \\ \frac{1-\pi}{J-1} & \text{otherwise} \end{Bmatrix}, \quad (1)$$

where $J$ is the number of classes. This may be unsuitable when one class label dominates the data, since a spammer who always selects the most common label will nonetheless have a high $\pi$.

**Spamming model (spam)**: proposed as part of MACE (Hovy et al., 2013), this model also assumes constant accuracy, $\pi$, but that when an annotator is incorrect, they label according to a spamming distribution, $\boldsymbol{\xi}$, that is independent of

[1] http://github.com/ukplab/arxiv2018-bayesian-ensembles

the true label, $t_\tau$.

$$A = p(c_\tau = i | t_\tau = j, \pi, \boldsymbol{\xi}) = \begin{cases} \pi + (1-\pi)\xi_j & \text{where } i = j \\ (1-\pi)\xi_j & \text{otherwise} \end{cases}. \quad (2)$$

This addresses the case where spammers choose the dominant label but does not explicitly model different error rates in each class. For example, if an annotator is better at detecting type 'x' spans than type 'y', or if they frequently miss the first token in a span, thereby labelling the start of a span as 'O' when the true label is 'B-x', this would not be explicitly modelled by *spam*.

**Confusion vector (CV)**: this approach learns a separate accuracy for each class label (Nguyen et al., 2017) using parameter vector, $\boldsymbol{\pi}$, of size $J$:

$$A = p(c_\tau = i | t_\tau = j, \boldsymbol{\pi}) = \begin{cases} \pi_j & \text{where } i = j \\ \frac{1-\pi_j}{J-1} & \text{otherwise} \end{cases}. \quad (3)$$

This model does not capture spamming patterns where one of the incorrect labels has a much higher likelihood than the others.

**Confusion matrix (CM)** (Dawid and Skene, 1979): this model can be seen as an expansion of the confusion vector so that $\boldsymbol{\pi}$ becomes a $J \times J$ matrix with values given by:

$$A = p(c_\tau = i | t_\tau = j, \boldsymbol{\pi}) = \pi_{j,i}. \quad (4)$$

This requires a larger number of parameters, $J^2$, compared to the $J + 1$ parameters of MACE or $J$ parameters of the confusion vector. Like *spam*, *CM* can model spammers who frequently chose one label regardless of the ground truth, but also models different error rates and biases for each class. However, *CM* ignores dependencies between annotations in a sequence, such as the fact that an 'I' cannot immediately follow an 'O'.

**Sequential Confusion Matrix (seq)**: we introduce a new extension to the confusion matrix to model the dependency of each label in a sequence on its predecessor, giving the following likelihood:

$$A = p(c_\tau = i | c_{\tau-1} = \iota, t_\tau = j, \boldsymbol{\pi}) = \pi_{j,\iota,i}, \quad (5)$$

where $\boldsymbol{\pi}$ is now three-dimensional with size $J \times J \times J$. In the case of disallowed transitions, e.g. from $c_{\tau-1}$ ='O' to $c_\tau$ ='I', the value $\pi_{j,c_{\tau-1},c_\tau} \approx 0$, $\forall j$ is fixed *a priori*. The sequential model can capture phenomena such as a tendency toward overly long sequences, by learning that I is more likely to follow another I, so that $\pi_{O,I,I} > \pi_{O,I,O}$. A tendency to split spans by inserting 'B' in place of 'I' can be modelled by increasing the value of $\pi_{I,I,B}$ without affecting $\pi_{I,B,B}$ and $\pi_{I,O,B}$.

The annotator models presented in this section include the most widespread models for NLP annotation tasks, and can be seen as extensions of one another. The choice of annotator model for a particular annotator depends on the developer's understanding of the annotation task: if the annotations have sequential dependencies, this suggests the *seq* model; for non-sequential classifications *CM* may be effective with small ($\leq 5$) numbers of classes; *spam* may be more suitable if there are many classes, as the number of parameters to learn is low. However, there is also a trade-off between the expressiveness of the model and the number of parameters that must be learned. Simpler models with fewer parameters may be effective if there are only small numbers of annotations from each annotator. The next section shows how these annotator models can be used as components of a complete model for aggregating sequential annotations.

## 4 A Generative Model for Bayesian Sequence Combination

To construct a generative model for *Bayesian sequence combination (BSC)*, we first define a hidden Markov model (HMM) with states $t_{n,\tau}$ and observations $x_{n,\tau}$ using categorical distributions:

$$t_{n,\tau} \sim \text{Cat}(\boldsymbol{T}_{t_{n,\tau-1}}), \quad (6)$$

$$x_{n,\tau} \sim \text{Cat}(\boldsymbol{\rho}_{t_{n,\tau}}), \quad (7)$$

where $\boldsymbol{T}_j$ is a row of a transition matrix $\boldsymbol{T}$, and $\boldsymbol{\rho}_j$ is a vector of observation likelihoods for state $j$. For text tagging, $n$ indicates a document and $\tau$ a token index, while each state $t_{n,\tau}$ is a true sequence label and $x_{n,\tau}$ is a token. To provide a Bayesian treatment, we assume that $\boldsymbol{T}_j$ and $\boldsymbol{\rho}_j$ have Dirichlet distribution priors as follows:

$$T_j \sim \text{Dir}(\boldsymbol{\gamma}_j), \qquad \boldsymbol{\rho}_j \sim \text{Dir}(\boldsymbol{\kappa}_j), \quad (8)$$

where $\boldsymbol{\gamma}_j$ and $\boldsymbol{\kappa}_j$ are hyperparameters.

Next, we assume one of the annotator models described in Section 3 for each of $K$ annotators. Selecting an annotator model is a design choice, and all can be coupled with the Bayesian HMM above to form a complete BSC model. In

our experiments in Section 6, we compare different choices of annotator model as components of BSC. All the parameters of these annotator models are probabilities, so to provide a Bayesian treatment, we assume that they have Dirichlet priors. For annotator $k$'s annotator model, we refer to the hyperparameters of its Dirichlet prior as $\boldsymbol{\alpha}^{(k)}$. The annotator model defines a categorical likelihood over each annotation, $\boldsymbol{c}_{n,\tau}^{(k)}$:

$$c_{n,\tau}^{(k)} \sim \text{Cat}([A^{(k)}(t_{n,\tau}, 1, \boldsymbol{c}_{n,\tau-1}^{(k)}), ..., A^{(k)}(t_{n,\tau}, J, \boldsymbol{c}_{n,\tau-1}^{(k)})]). \quad (9)$$

The annotators are assumed to be conditionally independent of one another given the true labels, $\boldsymbol{t}$, which means that their errors are assumed to be uncorrelated. This is a strong assumption when considering that the annotators have to make their decisions based on the same input data. However, in practice, dependencies do not usually cause the most probable label to change (Zhang, 2004), hence the performance of classifier combination methods is only slightly degraded, while avoiding the complexity of modelling dependencies between annotators (Kim and Ghahramani, 2012).

**Joint distribution**: the complete model can be represented by the joint distribution, given by:

$$p(\boldsymbol{t}, \boldsymbol{A}, \boldsymbol{T}, \boldsymbol{\rho}, \boldsymbol{c}, \boldsymbol{x} | \boldsymbol{\alpha}^{(1)}, ..., \boldsymbol{\alpha}^{(K)}, \boldsymbol{\gamma}, \boldsymbol{\kappa}) \quad (10)$$

$$= \prod_{k=1}^{K} \left\{ p(A^{(k)}|\boldsymbol{\alpha}^{(k)}) \prod_{n=1}^{N} p(\boldsymbol{c}_n^{(k)}|A^{(k)}, \boldsymbol{t}) \right\}$$

$$\prod_{n=1}^{N} \prod_{\tau=1}^{L_n} p(t_{n,\tau}|\boldsymbol{T}_{t_{n,\tau-1}}) p(x_{n,\tau}|t_{n,\tau}, \boldsymbol{\rho}_{t_{n,\tau}})$$

$$\prod_{j=1}^{J} p(\boldsymbol{T}_j|\boldsymbol{\gamma}_j) p(\boldsymbol{\rho}_j|\boldsymbol{\kappa}_j) \quad (11)$$

where $\boldsymbol{c}$ is the set of annotations for all documents from all annotators, $\boldsymbol{t}$ is the set of all sequence labels for all documents, $N$ is the number of documents, $L_n$ is the length of the $n$th document, $J$ is the number of classes, $\boldsymbol{x}$ is the set of all word sequences for all documents and $\boldsymbol{\rho}$, $\boldsymbol{\gamma}$ and $\boldsymbol{\kappa}$ are the sets of parameters for all label classes.

## 5 Inference using Variational Bayes

Given a set of annotations, $\boldsymbol{c}$, we obtain a posterior distribution over sequence labels, $\boldsymbol{t}$, using *variational Bayes* (*VB*) (Attias, 2000). Unlike maximum likelihood methods such as standard expectation maximization (EM), VB considers prior distributions and accounts for parameter uncertainty due to noisy or sparse data. In comparison to other Bayesian approaches such as Markov chain Monte Carlo (MCMC), VB is often faster, readily allows incremental learning, and provides easier ways to determine convergence (Bishop, 2006). It has been successfully applied to a wide range of methods, including being used as the standard learning procedure for LDA (Blei et al., 2003), and to combining non-sequential crowdsourced classifications (Simpson et al., 2013).

The trade-off is that we must approximate the posterior distribution with an approximation that factorises between subsets of latent variables:

$$p(\boldsymbol{t}, \boldsymbol{A}, \boldsymbol{T}, \boldsymbol{\rho} | \boldsymbol{c}, \boldsymbol{x}, \boldsymbol{\alpha}, \boldsymbol{\gamma}, \boldsymbol{\kappa})$$

$$\approx \prod_{k=1}^{K} q(A^{(k)}) \prod_{j=1}^{J} \{q(\boldsymbol{T}_j) q(\boldsymbol{\rho}_j)\} \prod_{n=1}^{N} q(\boldsymbol{t}_n). \quad (12)$$

VB performs approximate inference by updating each variational factor, $q(z)$, in turn, optimising the approximate posterior distribution until it converges. Details of the theory are beyond the scope of this paper, but are explained by Bishop (2006). The VB algorithm is described in Algorithm 1, making use of update equations for the variational factors given below.

**Input**: Annotations $\boldsymbol{c}$, tokens $\boldsymbol{x}$

Compute initial values of $\mathbb{E} \ln A^{(k)}, \forall k$, $\mathbb{E} \ln \rho_j, \forall j$, $\mathbb{E} \ln \boldsymbol{T}_j, \forall j$ from their prior distributions.

**while** not_converged$(r_{n,\tau,j}, \forall n, \forall \tau, \forall j)$ **do**

  Update $r_{j,n,\tau}$, $s_{t_{j,n,\tau-1}, t_{\iota,n,\tau}}, \forall j, \forall \tau, \forall n, \forall \iota$, using forward-backward algorithm given $\boldsymbol{x}$, $\boldsymbol{c}$, $\mathbb{E} \ln \boldsymbol{T}_j, \forall j$, $\mathbb{E} \ln \boldsymbol{\rho}_j, \forall j$, $\mathbb{E} \ln \boldsymbol{A}^{(k)}, \forall k$.

  Update $\mathbb{E} \ln A^{(k)}, \forall k$, given $\boldsymbol{c}$, $r_{j,n,\tau}$.

  Update $\mathbb{E} \ln T_{j,\iota}, \forall j, \forall \iota$, given $s_{t_{j,n,\tau-1}, t_{\iota,n,\tau}}$.

  Update $\mathbb{E} \ln \rho_j, \forall j$ given $\boldsymbol{x}$, $r_{j,n,\tau}$.

**end**

**Output**: Label posteriors, $r_{n,\tau,j}, \forall n, \forall \tau, \forall j$, most probable sequence of labels, $\hat{\boldsymbol{t}}_n, \forall n$ using Viterbi algorithm

**Algorithm 1:** The VB algorithm for BSC.

We compute the posterior probability of each true token label, $r_{n,\tau,j} = \mathbb{E}[p(t_{n,\tau} = j|\boldsymbol{c})]$, and

of each label transition, $s_{n,\tau,j,\iota} = \mathbb{E}[p(t_{n,\tau-1} = j, t_{n,\tau} = \iota|\boldsymbol{c})]$, using the forward-backward algorithm (Ghahramani, 2001). In the forward pass, we compute:

$$\ln r^{-}_{n,\tau,j} = \ln \sum_{\iota=1}^{J} \left\{ r^{-}_{n,\tau-1,\iota} \mathrm{e}^{\mathbb{E}\ln T_{\iota,j}} \right\} + ll_{n,\tau}(j),$$

$$ll_{n,\tau}(j) = \sum_{k=1}^{K} \mathbb{E}\ln A^{(k)}\left(j, c^{(k)}_{n,\tau}, c^{(k)}_{n,\tau-1}\right) \mathbb{E}\ln \rho_{j,x_{n,\tau}}, \tag{13}$$

and in the backward pass we compute:

$$\ln \lambda_{n,\tau,j} = \ln \sum_{\iota=1}^{J} \exp\big\{ \ln \lambda_{i,\tau+1,\iota} + \mathbb{E}\ln T_{j,\iota} + ll_{n,\tau+1}(\iota) \big\}. \tag{14}$$

Then we can calculate the posteriors as follows:

$$r_{n,\tau,j} \propto r^{-}_{n,\tau,j} \lambda_{n,\tau,j}, \tag{15}$$

$$s_{n,\tau,j,\iota} \propto r^{-}_{n,\tau-1,j} \lambda_{n,\tau,\iota} \exp\{\mathbb{E}\ln T_{j,\iota} + ll_{n,\tau}(\iota)\}. \tag{16}$$

The expectations of $\ln \boldsymbol{T}$ and $\ln \boldsymbol{\rho}$ can be computed using standard equations for a Dirichlet distribution:

$$\mathbb{E}\ln T_{j,\iota} = \Psi(N_{j,\iota} + \gamma_{j,\iota}) - \Psi\left(\sum_{i=1}^{J} (N_{j,i} + \gamma_{j,i})\right), \tag{17}$$

$$\mathbb{E}\ln \rho_{j} = \Psi(o_{j,w} + \kappa_{j,w}) - \Psi\left(\sum_{v=1}^{J} (o_{j,v} + \kappa_{j,v})\right), \tag{18}$$

where $\Psi$ is the digamma function, $N_{j,\iota} = \sum_{n=1}^{N}\sum_{\tau=1}^{L_n} s_{n,\tau,j,\iota}$ is the expected number of times that label $\iota$ follows label $j$, and $o_{j,w}$ is the expected number of times that word $w$ occurs with sequence label $j$. Similarly, for the *seq* annotator model, the expectation terms are:

$$\mathbb{E}\ln A^{(k)}(j,l,m) = \Psi\left(N^{(k)}_{j,l,m}\right) - \Psi\left(\sum_{m'=1}^{J}\left(N^{(k)}_{j,l,m'}\right)\right). \tag{19}$$

$$N^{(k)}_{j,l,m} = \alpha^{(k)}_{j,l,m} + \sum_{n=1}^{N}\sum_{\tau=1}^{L_n} r_{n,\tau,j}\delta_{l,c^{(k)}_{n,\tau-1}}\delta_{m,c^{(k)}_{n,\tau}}, \tag{20}$$

where $\delta$ is the Kronecker delta. For other annotator models, this equation is simplified as the values of the previous labels $c^{(k)}_{n,\tau-1}$ are ignored.

| data -set | #sentences or docs total | dev | test | #annotators total | /doc | #gold spans | span length |
|---|---|---|---|---|---|---|---|
| NER | 6,056 | 2,800 | 3,256 | 47 | 4.9 | 21,612 | 1.51 |
| PICO | 9,480 | 191 | 191 | 312 | 6.0 | 700 | 7.74 |
| ARG | 8,000 | 60 | 100 | 105 | 5 | 73 | 17.52 |

Table 1: Dataset statistics. Span lengths are means.

### 5.1 Most Likely Sequence Labels

The approximate posterior probabilities of the true labels, $r_{j,n,\tau}$, provide confidence estimates for the labels. However, it is often useful to compute the most probable sequence of labels, $\hat{\boldsymbol{t}}_n$, using the Viterbi algorithm (Viterbi, 1967). The most probable sequence is particularly useful because, unlike $r_{j,n,\tau}$, the sequence will be consistent with any transition constraints imposed by the priors on the transition matrix $\boldsymbol{T}$, such as preventing 'O'→'I' transitions by assigning them zero probability.

## 6 Experiments

**Datasets.** We compare BSC to alternative methods on three NLP datasets containing both crowdsourced and gold sequential annotations. *NER*, the CoNLL 2003 named-entity recognition dataset (Tjong Kim Sang and De Meulder, 2003), which contains gold labels for four named entity categories (PER, LOC, ORG, MISC), with crowdsourced labels provided by (Rodrigues et al., 2014). *PICO* (Nguyen et al., 2017), consists of medical paper abstracts that have been annotated by a crowd to indicate text spans that identify the population enrolled in a clinical trial. *ARG* (Trautmann et al., 2019) contains a mixture of argumentative and non-argumentative sentences, in which the task is to mark the spans that contain pro or con arguments for a given topic. Dataset statistics are shown in Table 1. The datasets differ in typical span length, with argument components in ARG the longest, while named entities in NER spans are often only one token long.

The gold-labelled documents are split into validation and test sets. For NER, we use the split given by Nguyen et al. (2017), while for PICO and ARG, we make random splits since the splits from previous work were not available[2].

**Evaluation metrics.** For NER and ARG we use the CoNLL 2003 F1-score, which considers only

[2]The data splits are available from our Github repository. Since we use different splits, our results for PICO are not identical to Nguyen et al. (2017).

| | NER | | | | PICO | | | | ARG | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Prec. | Rec. | F1 | CEE | Prec. | Rec. | F1 | CEE | Prec. | Rec. | F1 | CEE |
| Best worker | 76.4 | 60.1 | 67.3 | 17.1 | 64.8 | 53.2 | 58.5 | 17.0 | 62.7 | 57.5 | 60.0 | 44.20 |
| Worst worker | 55.7 | 26.5 | 35.9 | 31.9 | 50.7 | 52.9 | 51.7 | 41.0 | 25.5 | 19.2 | 21.9 | 70.33 |
| MV | 79.9 | 55.3 | 65.4 | 6.24 | 82.5 | 52.8 | 64.3 | 2.55 | 40.0 | 31.5 | 34.8 | 14.03 |
| MACE | 74.4 | 66.0 | 70.0 | 1.01 | 25.4 | 84.1 | 39.0 | 58.2 | 31.2 | 32.9 | 32.0 | 2.62 |
| DS | 79.0 | 70.4 | 74.4 | 2.80 | 71.3 | 66.3 | 68.7 | 0.44 | 45.6 | 49.3 | 47.4 | 0.97 |
| IBCC | 79.0 | 70.4 | 74.4 | **0.49** | 72.1 | 66.0 | 68.9 | **0.27** | 44.9 | 47.9 | 46.4 | **0.85** |
| HMM-crowd | 80.1 | 69.2 | 74.2 | 1.00 | 75.9 | 66.7 | 71.0 | 0.99 | 43.5 | 37.0 | 40.0 | 3.38 |
| BSC-acc | **83.4** | 54.3 | 65.7 | 0.96 | **89.4** | 45.2 | 60.0 | 1.59 | 36.9 | 32.9 | 34.8 | 6.47 |
| BSC-spam | 67.9 | 74.1 | 70.9 | 0.89 | 46.7 | **84.4** | 60.1 | 1.98 | 55.7 | 53.4 | 54.5 | 2.80 |
| BSC-CV | 83.0 | 64.6 | 72.6 | 0.93 | 74.9 | 67.2 | 71.1 | 0.84 | 37.9 | 34.2 | 36.0 | 4.73 |
| BSC-CM | 79.9 | 72.2 | 75.8 | 1.46 | 60.1 | 78.8 | 68.2 | 1.49 | **56.0** | 57.5 | 56.8 | 3.76 |
| BSC-seq | 80.3 | **74.8** | **77.4** | 0.65 | 70.4 | 75.3 | **72.8** | 0.53 | 54.4 | **67.1** | **60.1** | 3.26 |
| BSC-CM-notext | 74.7 | 69.7 | 72.1 | 1.48 | 62.7 | 74.8 | 68.2 | 1.32 | 55.1 | 58.9 | 57.0 | 2.75 |
| BSC-CM$\backslash \boldsymbol{T}$ | 80.0 | 73.0 | 76.3 | 0.99 | 65.8 | 66.7 | 66.2 | 0.28 | 52.9 | 49.3 | 51.1 | 1.69 |
| BSC-seq-notext | 81.3 | 71.9 | 76.3 | 0.52 | 81.2 | 59.2 | 68.5 | 0.73 | 36.9 | 52.0 | 43.2 | 5.64 |
| BSC-seq$\backslash \boldsymbol{T}$ | 64.2 | 44.4 | 52.5 | 0.77 | 51.2 | 70.4 | 59.8 | 1.04 | 0.11 | 0.05 | 0.07 | 6.38 |

Table 2: Aggregating crowdsourced labels: estimating true labels for documents labelled by the crowd.

exact span matches to be correct. Incomplete named entities are typically not useful, and for ARG, it is desirable to identify complete argumentative units that make sense on their own. For medical trial populations, partial matches still contain useful information, so for PICO we use a relaxed F1-score, as in Nguyen et al. (2017), which counts the matching fractions of spans when computing precision and recall.

We also compute the cross entropy error (*CEE*, also known as log-loss). While this is a token-level rather than span-level metric, it evaluates the quality of the probability estimates produced by aggregation methods, which are useful for tasks such as active learning, as it penalises over-confident mistakes more heavily.

**Evaluated Methods.** We evaluate BSC in combination with all of the annotator models described in Section 4. As well-established non-sequential baselines, we include token-level majority voting (*MV*), *MACE* (Hovy et al., 2013) which uses the *spam* annotator model, Dawid-Skene (*DS*) (Dawid and Skene, 1979), which uses the *CM* annotator model, and independent Bayesian classifier combination (*IBCC*) (Kim and Ghahramani, 2012), which is a Bayesian treatment of Dawid-Skene. We also compare against the state-of-the-art sequential *HMM-crowd* method (Nguyen et al., 2017), which uses a combination of maximum *a posteriori* (or smoothed maximum likelihood) estimates for the *CV* annotator model and variational inference for an integrated hidden Markov model (HMM). HMM-Crowd and DS use non-Bayesian inference steps and can be compared with their Bayesian variants, BSC-CV and IBCC, respectively.

Besides the annotator models, BSC also makes use of text features and a transition matrix, $\boldsymbol{T}$, over true labels. We test the effect of these components by running BSC-CM and BSC-seq with no text features (*notext*), and without the transition matrix, which is replaced by simple independent class probabilities (labelled $\backslash \boldsymbol{T}$).

We tune the hyperparameters using grid search on the development sets. To limit the number of hyperparameters to tune, we optimize only three values for BSC: hyperparameters of the transition matrix, $\boldsymbol{\gamma}_j$, are set to the same value, $\gamma_0$, except for disallowed transitions, (O→I and transitions between types, e.g. I-PER→I-ORG), which are set to $1e-6$; for the annotator models, all values are set to $\alpha_0$, except for disallowed transitions, which are set to $1e^{-6}$, then $\epsilon_0$ is added to hyperparameters corresponding to correct annotations (e.g. diagonal entries in a confusion matrix). This encodes the prior assumption that annotators are more likely to have an accuracy greater than random, which avoids the non-identifiability problem in which the class labels are switched around.

**Aggregation Results.** This task is to combine multiple crowdsourced labels and predict the true labels. The results are shown in Table 2. BSC-seq outperforms the other approaches, including the previous state of the art, HMM-crowd (sig-

| Method | Data -set | exact match | wrong type | partial match | missed span | false +ve | late start | early start | late finish | early finish | fused spans | splits | inv- alid | length error |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MV | NER | 2869 | 304 | 196 | 1775 | 100 | 96 | 10 | 15 | 85 | 17 | 26 | 81 | 0.04 |
| IBCC | NER | 3742 | 386 | 187 | 829 | 345 | 107 | 27 | 43 | 77 | 47 | 29 | 74 | 0.12 |
| HMM-crowd | NER | 3650 | 334 | 115 | 1045 | 210 | 109 | 22 | 33 | 89 | 37 | 23 | 0 | 0.03 |
| BSC-CV | NER | 3381 | 284 | 80 | 1399 | 121 | 94 | 17 | 18 | 90 | 22 | 8 | 0 | 0.00 |
| BSC-CM | NER | 3856 | 362 | 63 | 863 | 315 | 124 | 25 | 63 | 77 | 53 | 13 | 0 | 0.14 |
| BSC-seq | NER | 3995 | 353 | 110 | 686 | 357 | 84 | 29 | 25 | 88 | 28 | 26 | 0 | 0.09 |
| MV | PICO | 144 | 0 | 60 | 145 | 48 | 9 | 11 | 1 | 0 | 3 | 9 | 40 | 1.26 |
| IBCC | PICO | 193 | 0 | 53 | 103 | 100 | 14 | 10 | 0 | 2 | 3 | 10 | 19 | 0.45 |
| HMM-crowd | PICO | 189 | 0 | 54 | 106 | 84 | 13 | 21 | 0 | 0 | 5 | 8 | 0 | 1.99 |
| BSC-CV | PICO | 156 | 0 | 76 | 117 | 81 | 10 | 25 | 0 | 0 | 11 | 0 | 0 | 2.15 |
| BSC-CM | PICO | 216 | 0 | 50 | 83 | 157 | 10 | 19 | 0 | 0 | 4 | 17 | 0 | 2.42 |
| BSC-seq | PICO | 168 | 0 | 86 | 95 | 67 | 17 | 19 | 5 | 0 | 4 | 9 | 0 | 0.61 |
| MV | ARG | 17 | 0 | 26 | 14 | 4 | 9 | 1 | 0 | 2 | 0 | 0 | 9 | 5.27 |
| IBCC | ARG | 27 | 1 | 21 | 8 | 9 | 7 | 2 | 0 | 1 | 0 | 3 | 9 | 3.43 |
| HMM-Crowd | ARG | 20 | 0 | 23 | 14 | 4 | 7 | 2 | 0 | 2 | 0 | 0 | 4 | 4.87 |
| BSC-CV | ARG | 18 | 0 | 25 | 14 | 4 | 12 | 2 | 0 | 2 | 0 | 0 | 0 | 5.37 |
| BSC-CM | ARG | 35 | 1 | 12 | 9 | 9 | 7 | 2 | 0 | 1 | 1 | 0 | 0 | 2.11 |
| BSC-Seq | ARG | 39 | 3 | 12 | 3 | 20 | 6 | 4 | 0 | 0 | 1 | 0 | 0 | 0.46 |

Table 3: Counts of different types of span errors.

nificant on all datasets with $p \ll .01$ using a two-tailed Wilcoxon signed-rank test). Without the text model (BSC-seq-notext) or the transition matrix (BSC-seq$\backslash \boldsymbol{T}$), BSC-seq performance decreases heavily, while BSC-CM-notext and BSC-CM$\backslash \boldsymbol{T}$ in some cases outperform BSC-CM. This suggests that *seq*, with its greater number of parameters to learn, is most effective in combination with the transition matrix and simple text model. On the ARG dataset, the scores are close to zero for BSC-seq$\backslash \boldsymbol{T}$. Further investigation shows that this is because BSC-seq$\backslash \boldsymbol{T}$ identifies many spans with one or two incorrect labels. Since we use exact span matches to compute true and false positives, these small errors reduce the scores substantially. In particular, we find a large number of missing B tags at the start of spans and misplaced O tags that split spans in the middle.

The performance of all methods across the three datasets varies greatly. With NER, the spans are short and the task is less subjective than PICO or ARG, hence its higher F1 scores. PICO uses a relaxed F1 score, meaning its scores are only slightly lower despite being a more ambiguous task. The constitution of an argument is also ambiguous, so ARG scores are lower, particularly as we use strict span-matching to compute the F1 scores. Raising the scores may be possible in future by using expert labels as training data, i.e. as known values of $\boldsymbol{t}$, which would help to put more weight on annotators with similar labelling patterns to the experts.

We categorise the errors made by key methods and list the counts for each category in Table 3. All machine learning methods tested here reduce the number of spans that were completely missed by majority voting. Note that BSC completely removes all "invalid" spans (O→I transitions) due to the sequential model with prior hyperparameters set to zero for those transitions. For PICO and ARG, which contain longer spans, BSC-seq has lower "length error" than other methods, which is the mean difference in number of tokens between the predicted and gold spans. It also reduces the number of missing spans, although in NER and ARG that comes at the cost of producing more false positives (predicting spans where there are none). Overall, BSC-seq appears to be the best choice for identifying exact span matches and reducing missed spans.

**Visualising Annotator Models.** To determine whether, in practice, BSC-seq really learns distinctive confusion matrices depending on the previous labels, we plot the learned annotator models for PICO as probabilistic confusion matrices in Figure 1 (for an alternative visualisation, see Figure 3 in the appendix). We clustered the confusion matrices into five groups by applying K-means to their posterior expected values, then plotted the means for each cluster. Each column in the figure shows the confusion matrices corresponding to the same cluster of annotators. In all clusters, BSC-seq learns different confusion matrices depending on the previous label. There are also large varia-

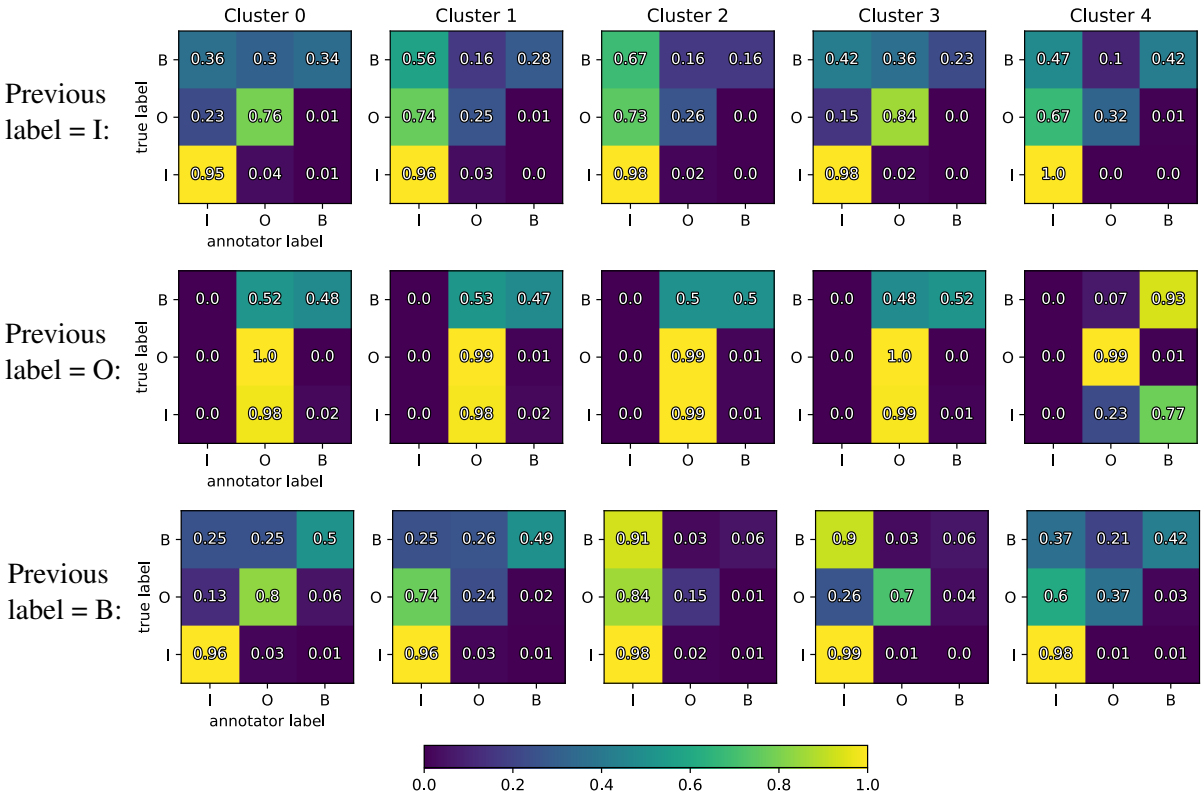


Figure 1: Clusters of annotators in the PICO dataset. Each column shows the mean of the confusion matrices in the *seq* annotator model for the members of one cluster, as inferred by BSC-seq. Each row corresponds to the mean confusion matrices conditioned on the annotator's previous label. The confusion matrices are plotted as heatmaps, where colours indicate the probabilities, which are also given by the numbers in each square.

tions between the clusters. The third column, for example, shows annotators with a tendency toward B→I transitions regardless of the true label, while other clusters indicate very different labelling behaviour. The model therefore appears able to learn distinct confusion matrices for different workers given previous labels, which supports the use of sequential annotator models.

**Active Learning.** Active learning is an iterative process that can reduce the amount of labelled data required to train a model. At each iteration, the active learning strategy selects informative data points, obtains their labels, then re-trains a labelling model given the new labels. The updated model is then used in the next iteration to identify the most informative data points.

We simulate active learning in a crowdsourcing scenario where the goal is to learn the true labels, $\boldsymbol{t}$, by selecting documents for the crowd to label. Each document can be labelled multiple times by different workers. In contrast, in a passive learning setup, the number of annotations per document is usually constant across all documents. For example, in the PICO dataset, each document was labelled six times. The aim of active learning is to decrease the number of annotations required by avoiding relabelling documents whose true labels can be determined with high confidence from fewer labels.

We simulate active learning using the *least confidence* strategy, shown to be effective by Settles and Craven (2008), as described in Algorithm 2. At each iteration, we estimate $\boldsymbol{t}$ from the current set of crowdsourced labels, $\boldsymbol{c}$, using one of the methods from our previous experiments as the labelling model, then use this model to determine the least confident $batch\_size$ documents to be labelled by the crowd. If the simulation has requested all the labels for a document that are available in our dataset, this document is simply ignored when choosing new batches and is not selected again.

We hypothesise that BSC will learn more quickly than non-sequential and non-Bayesian methods in the active learning simulation. Bayesian methods such as BSC account for

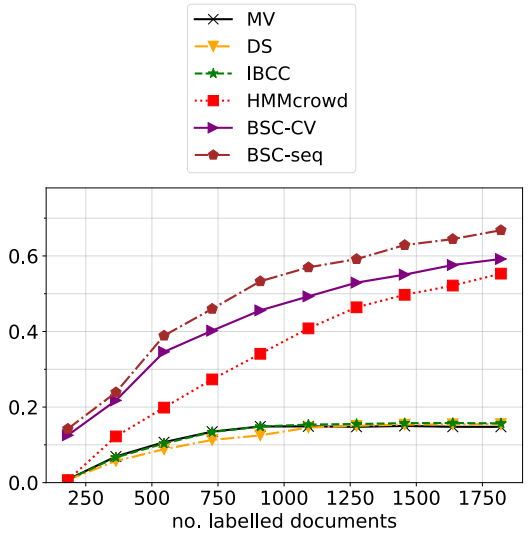


Figure 2: F1 scores for active learning simulations on NER using least-confidence uncertainty sampling.

uncertainty in the model parameters when estimating confidence, hence can be used to choose data points that rapidly improve the model itself. Sequential models such as BSC can also account for dependencies between sequence labels when estimating confidence.

**Input**: A random $initial\_set$ of training labels, the same for all methods.

Set training set $\boldsymbol{c} = initial\_set$

**while** *training set size* $< max\_no\_labels$ **do**

  Train model on $\boldsymbol{c}$

  For each document $n$, compute $LC_n = 1 - p(\boldsymbol{t}_n^*|\boldsymbol{c})$, where $\boldsymbol{t}_n^*$ is the probability of the most likely sequence of labels for $n$.

  Obtain annotations for $batch\_size$ documents with the highest values of $LC$ (least confidence), and add them to $\boldsymbol{c}$

**end**

**Algorithm 2:** Active learning simulation using least-confidence sampling.

Figure 2 plots the mean F1 scores over ten repeats of the active learning simulation on the NER dataset (for clarity, we only plot key methods). When the number of iterations is very small, neither IBCC nor DS are able to outperform majority vote, and only produce a very small benefit as the number of labels grows. This highlights the need for a sequential model such as BSC or HMM-crowd for effective active learning with small numbers of labels. IBCC learns slightly quicker than DS, while BSC-CV clearly outperforms HMM-crowd: we believe this difference is due to the Bayesian treatment of IBCC and BSC, which means they are better able to estimate confidence than DS and HMM-crowd, which use maximum likelihood and maximum a posteriori inference. BSC-seq produces the best overall performance, and the gap grows as the number of labels increases, since more data is required to learn the more complex annotator model.

## 7 Conclusions

We proposed BSC, a novel Bayesian approach to aggregating sequence labels that can be combined with several different models of annotator noise and bias. To model the effect of dependencies between labels on annotator noise and bias, we introduced the *seq* annotator model. Our results demonstrated the benefits of BSC over established non-sequential methods, such as MACE, Dawid-Skene (DS), and IBCC. We also showed the advantages of a Bayesian approach for active learning, and that the combination of *BSC* with the *seq* annotator model improves the state-of-the-art over HMM-crowd on three NLP tasks with different types of span annotations.

In future work, we plan to adapt active learning methods for easy deployment on crowdsourcing platforms, and to investigate techniques for automatically selecting good hyperparameters without recourse to a development set, which is often unavailable at the start of a crowdsourcing process.

## Acknowledgments

This work was supported by the German Research Foundation through the German-Israeli Project Cooperation (DIP, grant DA 1600/1-1 and grant GU 798/17-1), and by the German Research Foundation (DFG) as part of the QA-EduInf project (grant GU 798/18-1 and grant RI 803/12-1). We would like to thank all of those who developed the datasets used in this work.

## References

Hagai Attias. 2000. A variational Bayesian framework for graphical models. In *Advances in Neural Information Processing Systems 12*, pages 209–215. MIT Press.

Yoram Bachrach, Tom Minka, John Guiver, and Thore Graepel. 2012. How to grade a test without knowing

the answers: a Bayesian graphical model for adaptive crowdsourcing and aptitude testing. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pages 819–826. Omnipress.

Christopher. M. Bishop. 2006. *Pattern recognition and machine learning*, 4th edition. Information Science and Statistics. Springer.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022.

Alexander Philip Dawid and Allan M. Skene. 1979. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):20–28.

Thomas G Dietterich. 2000. Ensemble methods in Machine Learning. In *Multiple classifier systems*, pages 1–15. Springer.

Pinar Donmez, Jaime Carbonell, and Jeff Schneider. 2010. A probabilistic framework to learn from multiple annotators with time-varying accuracy. In *Proceedings of the 2010 SIAM International Conference on Data Mining*, pages 826–837. SIAM.

Paul Felt, Eric K. Ringger, and Kevin D. Seppi. 2016. Semantic annotation aggregation with conditional crowdsourcing models and word embeddings. In *International Conference on Computational Linguistics*, pages 1787–1796.

Zoubin Ghahramani. 2001. An introduction to hidden markov models and Bayesian networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 15(01):9–42.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard H Hovy. 2013. Learning whom to trust with MACE. In *HLT-NAACL*, pages 1120–1130.

Hyun-chul Kim and Zoubin Ghahramani. 2012. Bayesian classifier combination. In *International Conference on Artificial Intelligence and Statistics*, pages 619–627.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT*, pages 260–270.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1064–1074.

Pablo G. Moreno, Yee Whye Teh, and Fernando Perez-Cruz. 2015. Bayesian nonparametric crowdsourcing. *Journal of Machine Learning Research*, 16:1607–1627.

An T Nguyen, Byron C Wallace, Junyi Jessy Li, Ani Nenkova, and Matthew Lease. 2017. Aggregating and predicting sequence labels from crowd annotations. In *Proceedings of the conference. Association for Computational Linguistics. Meeting*, volume 2017, page 299. NIH Public Access.

Silviu Paun, Bob Carpenter, Jon Chamberlain, Dirk Hovy, Udo Kruschwitz, and Massimo Poesio. 2018. Comparing Bayesian models of annotation. *Transactions of the Association for Computational Linguistics*, 6:571–585.

Vikas. C. Raykar, Shipeng Yu, Linda H. Zhao, Gerardo Hermosillo. Valadez, Charles Florin, Luca Bogoni, and Linda Moy. 2010. Learning from crowds. *Journal of Machine Learning Research*, 11:1297–1322.

Filipe Rodrigues, Francisco Pereira, and Bernardete Ribeiro. 2013. Learning from multiple annotators: distinguishing good from random labelers. *Pattern Recognition Letters*, 34(12):1428–1436.

Filipe Rodrigues, Francisco Pereira, and Bernardete Ribeiro. 2014. Sequence labeling with multiple annotators. *Machine learning*, 95(2):165–181.

Burr Settles. 2010. Active learning literature survey. *Computer Sciences Technical Report 1648, University of Wisconsin-Madison*, 52(55-66):11.

Burr Settles and Mark Craven. 2008. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1070–1079. Association for Computational Linguistics.

Aashish Sheshadri and Matthew Lease. 2013. SQUARE: A benchmark for research on computing crowd consensus. In *First AAAI Conference on Human Computation and Crowdsourcing*.

Edwin Simpson, Stephen J. Roberts, Ioannis Psorakis, and Arfon Smith. 2013. Dynamic Bayesian combination of multiple imperfect classifiers. *Intelligent Systems Reference Library series*, Decision Making with Imperfect Decision Makers:1–35.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics.

Dietrich Trautmann, Johannes Daxenberger, Christian Stab, Hinrich Schütze, and Iryna Gurevych. 2019. Robust argument unit recognition and classification. *arXiv preprint arXiv:1904.09688*.

Matteo Venanzi, John Guiver, Gabriella Kazai, Pushmeet Kohli, and Milad Shokouhi. 2014. Community-based Bayesian aggregation models for crowdsourcing. In *23rd international conference on World wide web*, pages 155–164.

Matteo Venanzi, John Guiver, Pushmeet Kohli, and Nicholas R Jennings. 2016. Time-sensitive Bayesian information aggregation for crowdsourcing systems. *Journal of Artificial Intelligence Research*, 56:517–545.

Andrew Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*, 13(2):260–269.

Jacob Whitehill, Ting-fan Wu, Jacob Bergsma, Javier R Movellan, and Paul L Ruvolo. 2009. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Advances in neural information processing systems*, pages 2035–2043.

Hui Yuan Xiong, Yoseph Barash, and Brendan J Frey. 2011. Bayesian prediction of tissue-regulated splicing using rna sequence and cellular context. *Bioinformatics*, 27(18):2554–2562.

Harry Zhang. 2004. The optimality of naïve Bayes. In *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2004*. AAAI Press.

## A Discussion of Variational Approximation

In Equation 12, each subset of latent variables has a variational factor of the form $\ln q(z) = \mathbb{E}[\ln p(z|\boldsymbol{c}, \neg z)]$, where $\neg z$ contains all the other latent variables excluding $z$. We perform approximate inference by using coordinate ascent to update each variational factor, $q(z)$, in turn, taking expectations with respect to the current estimates of the other variational factors. Each iteration reduces the KL-divergence between the true and approximate posteriors of Equation 12, and hence optimises a lower bound on the log marginal likelihood, also called the evidence lower bound or ELBO (Bishop, 2006; Attias, 2000).

**Conjugate distributions:** The prior distributions chosen for our generative model are conjugate to the distributions over the latent variables and model parameters, meaning that each $q(z)$ is the same type of distribution as the corresponding prior distribution defined in Section 4. The parameters of each variational distribution are computed in terms of expectations over the other subsets of variables.

## B Update Equations for the Forward-Backward Algorithm

For the true labels, $\boldsymbol{t}$, the variational factor is:

$$\ln q(\boldsymbol{t}_n) = \sum_{n=1}^{N}\sum_{\tau=1}^{L_n}\sum_{k=1}^{K} \mathbb{E}\ln A^{(k)}\Big(t_{n,\tau}, c_{n,\tau}^{(k)}, c_{n,\tau-1}^{(k)}\Big) + \mathbb{E}\ln T_{t_{n,\tau-1},t_{n,\tau}} + \text{const.} \tag{21}$$

The forward-backward algorithm consists of two passes. The *forward pass* for each document, $n$, starts from $\tau = 1$ and computes:

$$\ln r^{-}_{n,\tau,j} = \ln \sum_{\iota=1}^{J}\left\{ r^{-}_{n,\tau-1,\iota} \mathrm{e}^{\mathbb{E}\ln T_{\iota,j}} \right\} + ll_{n,\tau}(j),$$

$$ll_{n,\tau}(j) = \sum_{k=1}^{K} \mathbb{E}\ln A^{(k)}\left(j, c_{n,\tau}^{(k)}, c_{n,\tau-1}^{(k)}\right) \tag{22}$$

where $r^{-}_{n,0,\iota} = 1$ where $\iota =$'O' and 0 otherwise. The *backwards pass* starts from $\tau = L_n$ and scrolls backwards, computing:

$$\ln \lambda_{n,L_n,j} = 0, \qquad \ln \lambda_{n,\tau,j} = \ln \sum_{\iota=1}^{J} \exp\big\{ \ln \lambda_{i,\tau+1,\iota} + \mathbb{E}\ln T_{j,\iota} + ll_{n,\tau+1}(\iota)\big\}. \tag{23}$$

By applying Bayes' rule, we arrive at $r_{n,\tau,j}$ and $s_{n,\tau,j,\iota}$:

$$r_{n,\tau,j} = \frac{r^{-}_{n,\tau,j}\lambda_{n,\tau,j}}{\sum_{j'=1}^{J} r^{-}_{n,\tau,j'}\lambda_{n,\tau,j'}} \tag{24}$$

$$s_{n,\tau,j,\iota} = \frac{\tilde{s}_{n,\tau,j,\iota}}{\sum_{j'=1}^{J}\sum_{\iota'=1}^{J}\tilde{s}_{n,\tau,j',\iota'}} \tag{25}$$

$$\tilde{s}_{n,\tau,j,\iota} = r^{-}_{n,\tau-1,j}\lambda_{n,\tau,\iota}\exp\{\mathbb{E}\ln T_{j,\iota} + ll_{n,\tau}(\iota)\}.$$

Each row of the transition matrix has the factor:

$$\ln q(\boldsymbol{T}_j) = \ln \mathrm{Dir}\left([N_{j,\iota} + \gamma_{j,\iota}, \forall \iota \in \{1,..,J\}]\right), \tag{26}$$

where $N_{j,\iota} = \sum_{n=1}^{N}\sum_{\tau=1}^{L_n} s_{n,\tau,j,\iota}$ is the expected number of times that label $\iota$ follows label $j$. The forward-backward algorithm requires expectations of $\ln \boldsymbol{T}$ that can be computed using standard equations for a Dirichlet distribution:

$$\mathbb{E}\ln T_{j,\iota} = \Psi(N_{j,\iota} + \gamma_{j,\iota}) - \Psi\left(\sum_{\iota=1}^{J}(N_{j,\iota} + \gamma_{j,\iota})\right), \tag{27}$$

where $\Psi$ is the digamma function.

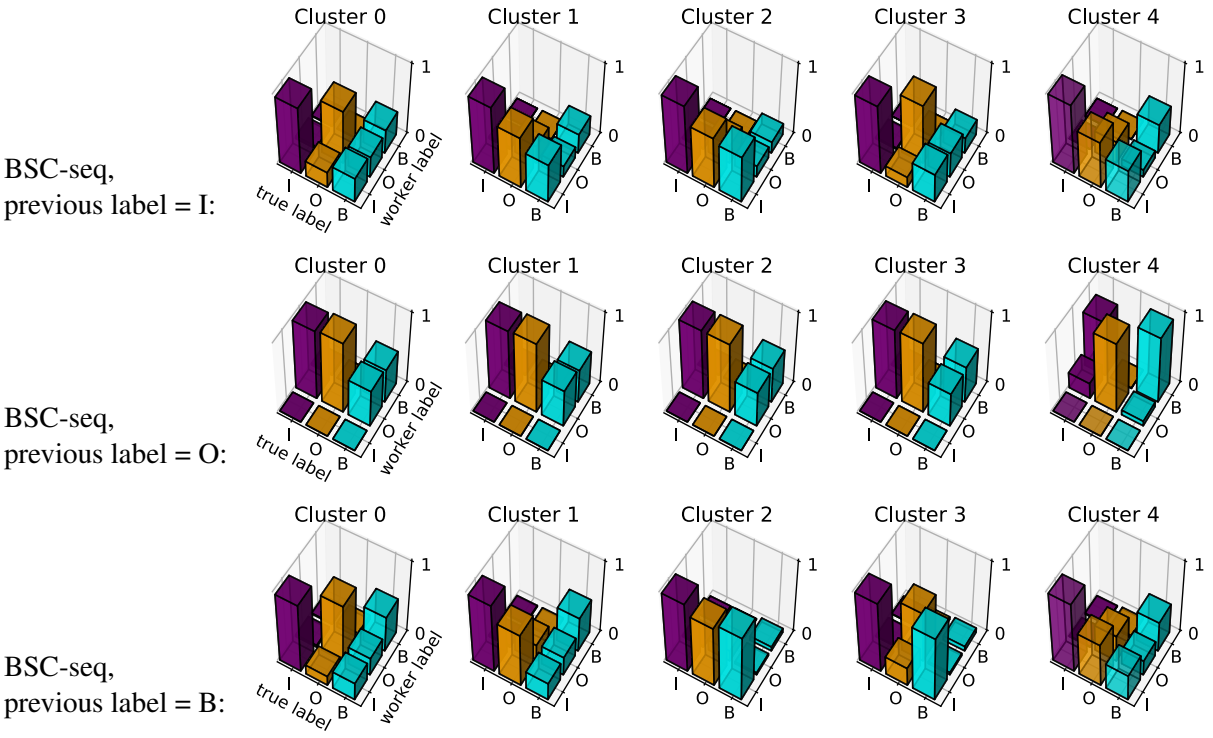


Figure 3: Clusters of annotators in PICO represented by the mean confusion matrices from BSC-seq. Heights of bars indicate likelihood of a worker (or annotator) label given the true label.

The variational factor for each annotator model is a distribution over its parameters, which differs between models. For *seq*, the variational factor is:

$$\ln q\left(A^{(k)}\right) = \sum_{j=1}^{J}\sum_{l=1}^{J} \mathrm{Dir}\left(\left[\boldsymbol{N}_{j,l,m}^{(k)} \forall m \in \{1,..,J\}\right]\right)$$

$$N_{j,l,m}^{(k)} = \alpha_{j,l,m}^{(k)} + \sum_{n=1}^{N}\sum_{\tau=1}^{L_n} r_{n,\tau,j}\,\delta_{l,c_{n,\tau-1}^{(k)}}\,\delta_{m,c_{n,\tau}^{(k)}}, \quad (28)$$

where $\delta$ is the Kronecker delta. For *CM*, *MACE*, *CV* and *acc*, the factors follow a similar pattern of summing pseudo-counts of correct and incorrect answers.

## C Visualising Annotator Models

Figure 3 provides an alternative visualisation of the *seq* models inferred by BSC-seq for annotators in the PICO dataset. The annotators were clustered as described in Section 6 of the main paper, and the mean confusion matrices for each cluster are plotted in Figure 3 using 3D plots to emphasise the differences between the likelihoods of annotators in each cluster providing a particular label given the true label value.

# A systematic comparison of methods for low-resource dependency parsing on genuinely low-resource languages

**Clara Vania**[1] **Yova Kementchedjhieva**[2] **Anders Søgaard**[2] **Adam Lopez**[1]
[1]School of Informatics, University of Edinburgh, UK
[2]University of Copenhagen, Copenhagen, Denmark
c.vania@ed.ac.uk, {yova|soegaard}@di.ku.dk, alopez@inf.ed.ac.uk

## Abstract

Parsers are available for only a handful of the world's languages, since they require lots of training data. How far can we get with just a small amount of training data? We systematically compare a set of simple strategies for improving low-resource parsers: data augmentation, which has not been tested before; cross-lingual training; and transliteration. Experimenting on three typologically diverse low-resource languages—North Sámi, Galician, and Kazah—We find that (1) when only the low-resource treebank is available, data augmentation is very helpful; (2) when a related high-resource treebank is available, cross-lingual training is helpful and complements data augmentation; and (3) when the high-resource treebank uses a different writing system, transliteration into a shared orthographic spaces is also very helpful.

## 1 Introduction

Large annotated treebanks are available for only a tiny fraction of the world's languages, and there is a wealth of literature on strategies for parsing with few resources (Hwa et al., 2005; Zeman and Resnik, 2008; McDonald et al., 2011; Søgaard, 2011). A popular approach is to train a parser on a related high-resource language and adapt it to the low-resource language. This approach benefits from the availability of Universal Dependencies (UD; Nivre et al., 2016), prompting substantial research (Tiedemann and Agic, 2016; Agić, 2017; Rosa and Mareček, 2018), along with the VarDial and the CoNLL UD shared tasks (Zampieri et al., 2017; Zeman et al., 2017, 2018).

But low-resource parsing is still difficult. The organizers of the CoNLL 2018 UD shared task (Zeman et al., 2018) report that, in general, results on the task's nine low-resource treebanks "are extremely low and the outputs are hardly useful for downstream applications." So if we want to build a parser in a language with few resources, what can we do? To answer this question, we systematically compare several practical strategies for low-resource parsing, asking:

1. What can we do with only a very small *target* treebank for a low-resource language?
2. What can we do if we also have a *source* treebank for a related high-resource language?
3. What if the source and target treebanks do not share a writing system?

Each of these scenarios requires different approaches. **Data augmentation** is applicable in all scenarios, and has proven useful for low-resource NLP in general (Fadaee et al., 2017; Bergmanis et al., 2017; Sahin and Steedman, 2018). Transfer learning via **cross-lingual training** is applicable in scenarios 2 and 3. Finally, **transliteration** may be useful in scenario 3.

To keep our scenarios as realistic as possible, we assume that no taggers are available since this would entail substantial annotation. Therefore, our neural parsing models must learn to parse from words or characters—that is, they must be **lexicalized**—even though there may be little shared vocabulary between source and target treebanks. While this may intuitively seem to make cross-lingual training difficult, recent results have shown that lexical parameter sharing on characters and words can in fact improve cross-lingual parsing (de Lhoneux et al., 2018); and that in some circumstances, a lexicalized parser can outperform a delexicalized one, even in a low-resource setting (Falenska and Çetinoğlu, 2017).

We experiment on three language pairs from different language families, in which the first of each is a genuinely low-resource language: North



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1105–1116,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

Sámi and Finnish (Uralic); Galician and Portuguese (Romance); and Kazakh and Turkish (Turkic), which have different writing systems[1]. To avoid optimistic evaluation, we extensively experiment only with North Sámi, which we also analyse to understand *why* our cross-lingual training outperforms the other parsing strategies. We treat Galician and Kazakh as truly held-out, and test only our best methods on these languages. Our results show that:

1. When no source treebank is available, data augmentation is very helpful: dependency tree morphing improves labeled attachment score (LAS) by as much as 9.3%. Our analysis suggests that syntactic rather than lexical variation is most useful for data augmentation.

2. When a source treebank is available, cross-lingual parsing improves LAS up to 16.2%, but data augmentation still helps, by an additional 2.6%. Our analysis suggests that improvements from cross-lingual parsing occur because the parser learns syntactic regularities about word order, since it does not have access to POS and has little reusable information about word forms.

3. If source and target treebanks have different writing systems, transliterating them to a common orthography is very effective.

## 2 Methods

We describe three techniques for improving low-resource parsing: (1) two data augmentation methods which have not been applied before for dependency parsing, (2) cross-lingual training, and (3) transliteration.

### 2.1 Data augmentation by dependency tree morphing (Morph)

Sahin and Steedman (2018) introduce two operations to augment a dataset for low-resource POS tagging. Their method assumes access to a dependency tree, but they do not test it for dependency parsing, which we do here for the first time. The first operation, *cropping*, removes some parts of a sentence to create a smaller or simpler, meaningful

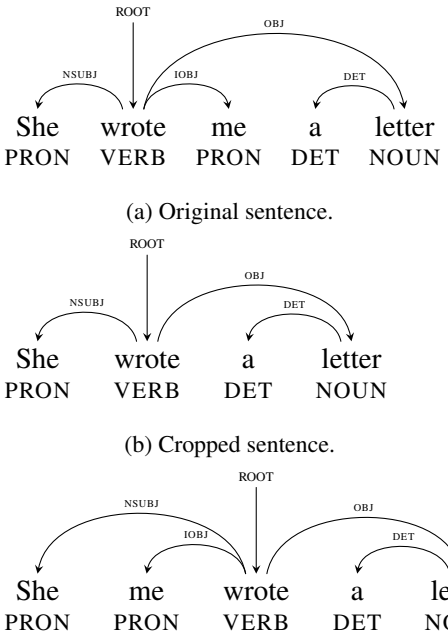


(a) Original sentence.

(b) Cropped sentence.

(c) Rotated sentence.

Figure 1: Examples of dependency tree morphing operations on the sentence "She wrote me a letter".

sentence. The second operation, *rotation*, keeps all the words in the sentence but re-orders subtrees attached to the *root* verb, in particular those attached by NSUBJ (nominal subject), OBJ (direct object), IOBJ (indirect object), or OBL (oblique nominal) dependencies. Figure 1 illustrates both operations.

It is important to note that while both operations change the set of words or the word order, they do not change the dependencies. The sentences themselves may be awkward or ill-formed, but the corresponding analyses are still likely to be correct, and thus beneficial for learning. This is because they provide the model with more examples of variations in argument structure (cropping) and in constituent order (rotation), which may benefit languages with flexible word order and rich morphology. Some of our low-resource languages have these properties—while North Sámi has a fixed word order (SVO), Galician and Kazakh have relatively free word order. All three languages use case marking on nouns, so word order may not be as important for correct attachment.

Both rotation and cropping can produce many trees. We use the default parameters given in (Sahin and Steedman, 2018).

[1]We select high-resource language based on language family, since it is the most straightforward way to define language relatedness. However, other measurement (e.g., WALS (Dryer and Haspelmath, 2013) properties) might be used.

### 2.2 Data augmentation by nonce sentence generation (Nonce)

Our next data augmentation method is adapted from Gulordava et al. (2018). The main idea is to create *nonce* sentences by replacing some of the words which have the same syntactic annotations. For each training sentence, we replace each content word—nouns, verbs, or adjective—with an alternative word having the same universal POS, morphological features, and dependency label.[2] Specifically, for each content word, we first stochastically choose whether to replace it; then, if we have chosen to replace it, we uniformly sample the replacement word type meeting the corresponding constraints. For instance, given a sentence "*He borrowed a book from the library.*", we can generate the following sentences:

(1) a. He bought a book from the shop .

b. He wore a umbrella from the library .

This generation method is only based on syntactic features (i.e., morphology and dependency labels), so it sometimes produces nonsensical sentences like 1b. But since we only replace words if they have the same morphological features and dependency label, this method preserves the original tree structures in the treebank. Following (Gulordava et al., 2018), we generate five nonce sentences for each original sentence.

### 2.3 Cross-lingual training

When a source treebank is available, model transfer is a viable option. We perform model transfer by cross-lingual parser training: we first train on both source and target treebanks to produce a single model, and then fine tune the model only on the target treebank. In our preliminary experiments (Appendix A), we found that fine tuning on the target treebank was effective in all settings, so we use it in all applicable experiments reported in this paper.

### 2.4 Transliteration

Two related languages might not share a writing system even when they belong to the same family. We evaluate whether a simple transliteration would be helpful for cross-lingual training in this case. In our study, the Turkish treebank is written in extended Latin while the Kazakh treebank is written in Cyrillic. This difference potentially makes model transfer less useful, and means we might not be able to leverage lexical similarities between the two languages. We pre-process both treebanks by transliterating them to the same "pivot" alphabet, basic Latin.[3]

The mapping from Turkish is straightforward. Its alphabet consists of 29 letters, 23 of which are in basic Latin. The other six letters, 'ç','ğ', 'ı', 'ö', 'ş', and 'ü', add diacritics to basic Latin characters, facilitating different pronunciations.[4] We map these to their basic Latin counterparts, e.g., 'ç' to 'c'. For Kazakh, we use a simple dictionary created by a Kazakh computational linguist to map each Cyrillic letter to the basic Latin alphabet.[5].

## 3 Experimental Setup

### 3.1 Dependency Parsing Model

We use the Uppsala parser, a transition-based neural dependency parser (de Lhoneux et al., 2017a,b; Kiperwasser and Goldberg, 2016). The parser uses an arc-hybrid transition system (Kuhlmann et al., 2011), extended with a static-dynamic oracle and SWAP transition to allow non-projective dependency trees (Nivre, 2009).

Let $w = w_0, \ldots, w_{|w|}$ be an input sentence of length $|w|$ and let $w_0$ represent an artificial ROOT token. We create a vector representation for each input token $w_i$ by concatenating $(;)$ its word embedding, $\mathbf{e}_w(w_i)$ and its character-based word embedding, $\mathbf{e}_c(w_i)$:

$$\mathbf{x}_i = [\mathbf{e}_w(w_i); \mathbf{e}_c(w_i)] \quad (1)$$

Here, $\mathbf{e}_c(w_i)$ is the output of a *character-level* bidirectional LSTM (biLSTM) encoder run over the characters of $w_i$ (Ling et al., 2015); this makes the model fully open-vocabulary, since it can produce representations for any character sequence. We then obtain a *context-sensitive* encoding $\mathbf{h}_i$ using a *word-level* biLSTM encoder:

$$\mathbf{h}_i = [\text{LSTM}_f(\mathbf{x}_{0:i}); \text{LSTM}_b(\mathbf{x}_{|w|:i})] \quad (2)$$

We then create a configuration by concatenating the encoding of a fixed number of words on the top of the stack and the beginning of the buffer. Given

[2]The dependency label constraint is new to this paper.

[3]Another possible pivot is phonemes (Tsvetkov et al., 2016). We leave this as future work.

[4]https://www.omniglot.com/writing/turkish.htm

[5]The mapping from Kazakh Cyrilic into basic Latin alphabet is provided in Appendix B

| Language | Treebank ID | train | dev. | test |
|---|---|---|---|---|
| Finnish | fi_tdt | 14981 | 1875 | 1555 |
| North Sámi | sme_giella | 1128 | 1129 | 865 |
| Portuguese | pt_bosque | 8329 | 560 | 477 |
| Galician | gl_treegal | 300 | 300 | 400 |
| Turkish | tr_imst | 3685 | 975 | 975 |
| Kazakh | kk_ktb | 15 | 16 | 1047 |

Table 1: Train/dev split used for each treebank.

this configuration, we predict a transition and its arc label using a multi-layer perceptron (MLP). More details of the core parser can be found in de Lhoneux et al. (2017a,b).

### 3.2 Parameter sharing

To train cross-lingual models, we use the strategy of de Lhoneux et al. (2018) for parameter sharing, which uses *soft* sharing for word and character parameters, and *hard* sharing for the MLP parameters. Soft parameter sharing uses a language embedding, which, in theory, learns what parameters to share between the two languages. Let $\mathbf{c}_j$ be an embedding of character $c_j$ in a token $w_i$ from the treebank of language $k$, and let $\mathbf{l}_k$ be the language embedding. For sharing on characters, we concatenate character and language embedding: $[\mathbf{c}_j; \mathbf{l}_k]$ for input to the character-level biLSTM. Similarly, for input to the word-level biLSTM, we concatenate the language embedding to the word embedding, modifying Eq. 1 to

$$\mathbf{x}_i = [\mathbf{e}_w(w_i); \mathbf{e}_c(w_i); \mathbf{l}_k] \quad (3)$$

We use the default hyperparameters of de Lhoneux et al. (2018) in our experiments. We fine-tune each model by training it further only on the target treebank (Shi et al., 2016). We use early stopping based on Label Attachment Score (LAS) on development set.

### 3.3 Datasets

We use Universal Dependencies (UD) treebanks version 2.2 (Nivre et al., 2018). None of our target treebanks have a development set, so we generate new train/dev splits by 50:50 (Table 1). Having large development sets allow us to perform better analysis for this study.

## 4 Parsing North Sámi

North Sámi is our largest low-resource treebank, so we use it for a full evaluation and analysis of different strategies before testing on the other languages. To understand the effect of target treebank size, we generate three datasets with different *training* sizes: $\mathcal{T}_{10}$ (~10%), $\mathcal{T}_{50}$ (~50%), and $\mathcal{T}_{100}$ (100%). Table 2 reports the number of training sentences after we augment the data using methods described in Section 2. We apply MORPH and NONCE separately to understand the effect of each method and to control the amount of noise in the augmented data.

| | original | +Morph | +Nonce |
|---|---|---|---|
| $\mathcal{T}_{100}$ | 1128 | 7636 | 4934 |
| $\mathcal{T}_{50}$ | 564 | 3838 | 2700 |
| $\mathcal{T}_{10}$ | 141 | 854 | 661 |

Table 2: Number of North Sámi training sentences.

We employ two baselines: a monolingual model (§3.1) and a cross-lingual model (§2.3), both *without* data augmentation. The monolingual model acts as a simple baseline, to resemble a situation when the target treebank does not have any source treebank (i.e., no available treebanks from related languages). The cross-lingual model serves as a strong baseline, simulating a case when there is a source treebank. We compare both baselines to models trained with MORPH and NONCE augmentation methods. Table 3 reports our results, and we review our motivating scenarios below.

**Scenario 1: we only have a very small target treebank.** In the monolingual experiments, we observe that both dependency tree morphing (MORPH) and nonce sentence generation (NONCE) improve performance, indicating the strong benefits of data augmentation when there are no other resources available except the target treebank itself. In particular, when the number of training data is the lowest ($\mathcal{T}_{10}$), data augmentations improves performance up to 9.3% LAS.

**Scenario 2: a source treebank is available.** We see that the cross-lingual training (cross-base) performs better than monolingual models even with augmentation. For the $\mathcal{T}_{10}$ setting, cross-base achieves almost twice as much as the monolingual baseline (mono-base). The benefits of data augmentation are less evident in the cross-lingual setting, but in the $\mathcal{T}_{10}$ scenario, data augmentation still clearly helps. Overall, cross-lingual combined with data augmentation yields the best result.

| | MONOLINGUAL | | | CROSS-LINGUAL | | |
|---|---|---|---|---|---|---|
| size | mono-base | +Morph | +Nonce | cross-base | +Morph | +Nonce |
| $\mathcal{T}_{100}$ | 53.3 | 56.0 (+3.3) | 56.3 (+3.0) | 61.3 (+8.0) | 60.9 (+7.6) | **61.7 (+8.4)** |
| $\mathcal{T}_{50}$ | 42.5 | 46.6 (+4.1) | 46.5 (+4.0) | 52.0 (+9.5) | 51.7 (+9.2) | **52.0 (+9.5)** |
| $\mathcal{T}_{10}$ | 18.5 | 27.1 (+8.6) | 27.8 (+9.3) | 34.7 (+16.2) | **37.3 (+18.8)** | 35.4 (+16.9) |

Table 3: LAS results on North Sámi development data. *mono-base* and *cross-base* are models without data augmentation. % improvements over *mono-base* shown in parentheses.

### 4.1 What is learned from Finnish?

Why do cross-lingual training and data augmentation help? To put this question in context, we first consider their relationship. Finnish and North Sámi are mutually unintelligible, but they are typologically similar: of the 49 (mostly syntactic) linguistic features annotated for North Sámi in the Word Atlas of Languages (WALS; Dryer and Haspelmath, 2013), Finnish shares the same values for 42 of them.[6] Despite this and their phylogenetic and geographical relatedness, they share very little vocabulary: only 6.5% of North Sámi tokens appear in Finnish data, and these words are either proper nouns or closed class words such as pronouns or conjunctions. However, both languages do share many character-trigrams (72.5%, token-level), especially on suffixes.

Now we turn to an analysis of the $\mathcal{T}_{10}$ data setting, where we see the largest gains for all methods.

### 4.2 Analysis of data augmentation

For dependency parsing, POS features are important because they can provide strong signals whether there exists dependency between two words in a given sentence. For example, *subject* and *object* dependencies often occur between a NOUN and a VERB, as can be seen in Fig. 1a. We investigate the extent to which data augmentation is useful for learning POS features, using diagnostic classifiers (Veldhoen et al., 2016; Adi et al., 2016; Shi et al., 2016) to probe our model representations. Our central question is: do the models learn useful representations of POS, despite having no direct access to it? And if so, is this helped by data augmentation?

After training each model, we freeze the parameters and generate *context-dependent* representations (i.e., the output of *word-level* biLSTM, $\mathbf{h}_i$ in Eq. 2), for the training and development data. We then train a feed-forward neural network classifier to predict the POS tag of each word, using only the representation as input. To filter out the effect of cross-lingual training, we only analyze representations trained using the *monolingual* models. Our training and development data consists of 6321 and 7710 tokens, respectively. The percentage of OOV tokens is 40.5%.

| POS | %dev | baseline | %diff. with | |
|---|---|---|---|---|
| | | | +Morph | +Nonce |
| INTJ | 0.1 | 0.0 | 20.0 | 20.0 |
| PART | 1.5 | 70.1 | 7.7 | 0.8 |
| NUM | 1.9 | 19.2 | 15.1 | -4.1 |
| ADP | 1.9 | 15.7 | 24.5 | 19.7 |
| SCONJ | 2.4 | 57.8 | 5.9 | 7.6 |
| AUX | 3.2 | 26.3 | 27.2 | -4.9 |
| CCONJ | 3.4 | 91.3 | -0.8 | -4.2 |
| PROPN | 4.7 | 5.9 | 5.9 | -5.9 |
| ADJ | 6.5 | 12.7 | 3.8 | 0.2 |
| ADV | 9.0 | 42.9 | 11.8 | 11.5 |
| PRON | 13.4 | 63.2 | 5.4 | -2.7 |
| VERB | 25.7 | 72.4 | -6.2 | -4.5 |
| NOUN | 26.4 | 67.0 | 8.6 | 13.2 |

Table 4: Results for the monolingual POS predictions, ordered by the frequency of each tag in the dev split (%dev). %diff shows the difference between each augmentation method and monolingual models.

Table 4 reports the POS prediction accuracy. We observe that representations generated with monolingual MORPH seem to learn better POS, for most of the tags. On the other hand, representations generated with monolingual NONCE sometimes produce lower accuracy on some tags; only on nouns the accuracy is better than monolingual MORPH. We hypothesize that this is because NONCE sometimes generates meaningless sentences which confuse the model. In parsing this effect is less apparent, mainly because monolingual NONCE has the poorest POS representation for infrequent tags (%dev), and better representa-

[6] There are 192 linguistic features in WALS, but only 49 are defined for North Sámi. These features are mostly syntactic, annotated within different areas such as morphology, phonology, nominal and verbal categories, and word order.

| North Sámi | Top nearest Finnish words | |
|---|---|---|
| | char-level | word-level |
| *borrat* (VERB; eat) | *herrat* (NOUN; gentleman)<br>*kerrat* (NOUN; time)<br>*naurat* (VERB; laugh) | *käydä* (VERB; go)<br>*otan* (VERB; take)<br>*sain* (VERB; get) |
| *veahki* (NOUN; help) | *nuuhki* (VERB; sniff)<br>*väki* (NOUN; power)<br>*avarsi* (VERB; expand) | *tyhjäksi* (ADJ; empty)<br>*johonki* (PRON; something)<br>*lähtökohdaksi* (NOUN; basis) |
| *divrras* (ADJ; expensive) | *harras* (ADJ; devout)<br>*reipas* (ADJ; brave)<br>*sarjaporras* (NOUN; series) | *välttämätöntä* (ADJ; essential)<br>*mahdollista* (ADJ; possible)<br>*kilpailukykyisempi* (ADJ; competitive) |

Table 5: Most similar Finnish words for each North Sámi word based on cosine similarity.

tion of nouns.

### 4.3 Effects of cross-lingual training

Next, we analyze the effect of cross-lingual training by comparing the monolingual baseline to the cross-lingual model with MORPH.

**Cross-lingual representations.** The fact that cross-lingual model improves parsing performance is interesting, since Finnish and North Sámi have so little common vocabulary. What linguistic knowledge is transferred through cross-lingual training? We analyze whether words with the same POS category from the source and target treebanks have similar representations. To do this, we analyze the *head predictions*, and collect North Sámi tokens for which only the cross-lingual model correctly predicts the headword.[7] For these words, we compare token-level representations of North Sámi *development* data to Finnish *training* data.

We ask the following questions: Given the representation of a North Sámi word, what is the Finnish word with the most similar representation? Do they share the same POS category? Information other than POS may very well be captured, but we expect that the representations will reflect similar POS since POS is highly revelant to parsing. We use *cosine distance* to measure similarity.

We look at four categories for which cross-lingual training substantial improves results on the development set: adjectives, nouns, pronouns, and verbs. We analyze representations generated by two layers of the model in §3.1: (1) the output of character-level biLSTM (char-level), $\mathbf{e}_c(w_i)$ and (2) the output of word-level biLSTM (word-level), i.e., $\mathbf{h}_i$ in Eq. 2.

| POS | char-level (%) | word-level (%) |
|---|---|---|
| ADJ | 12.1 | 37.1 |
| NOUN | 55.8 | 63.5 |
| PRON | 12.9 | 68.0 |
| VERB | 34.2 | 69.0 |

Table 6: # of North Sámi tokens for which the most similar Finnish word has the same POS.

Table 5 shows examples of top three closest Finnish training words for a given North Sámi word. We observe that character-level representation focuses on orthographic similarity of suffixes, rather than POS. On the word-level representations, we find more cases when the top closest Finnish words have the same POS with the North Sámi word. In fact, when we compare the most similar Finnish word (Table 6) quantitatively, we find that the word-level representations of North Sámi are often similar to Finnish word with the same POS; the same trend does not hold for character-level representations. Since very few word tokens are shared, this suggests that improvements in cross-lingual training might simply be due to syntactic (i.e. word order) similarities between the two languages, captured in the dynamics of the biLSTM encoder—despite the fact that it knows very little about the North Sámi tokens themselves. The word-level representation has advantage over the char-level representation in the way that it has access to contextual information like word order, and it has knowledge about the other words in the sentence.

**Head and label prediction.** Lastly, we analyze the parsing performance of the monolingual com-

[7] Another possible way is to look at the label predictions. But since the monolingual baseline LAS is very low, we focus on the unlabeled attachment prediction since it is more accurate.

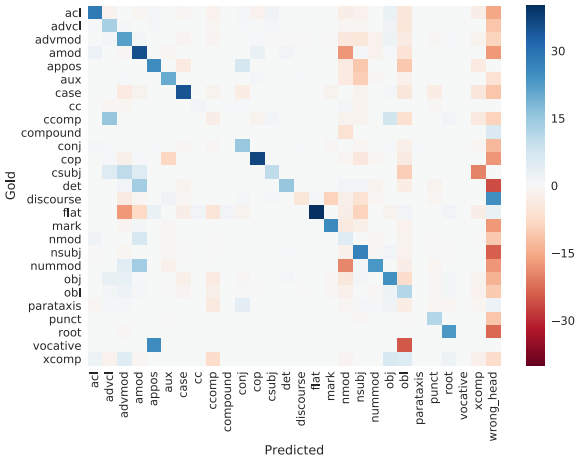


Figure 2: Differences between cross-lingual vs. monolingual confusion matrices. The last column represents cases of *incorrect* heads and the other columns represent cases for *correct* heads, i.e., each row summing to 100%. Blue cells show higher cross-lingual values and red cells show higher monolingual values.

pared to the cross-lingual models. Looking at the produced parse trees, one striking difference is that monolingual model sometimes predicts a "rootless" tree. That is, it fails to assign a head of any word with index '0' and label the dependency with a *root* label. In cases where the monolingual model predicts wrong parses and the cross-lingual model predicts the correct ones, we find that the "rootless" trees are predicted more than 50% of the time.[8] Meanwhile, the cross-lingual model learns to assign a word with head index '0', although sometimes it is the incorrect word (e.g., it is the second word, but the parser predicts the fifth word). This pattern suggests that more training examples at least helps the model to learn structural properties of a well-formed tree.

The ability of a parser to predict labels is contingent on its ability to predict heads, so we focus our analysis on two cases. How do monolingual and cross-lingual head prediction compare? And if both models predict the correct head, how do they compare on label prediction?

Figure 2 shows the *difference* between two confusion matrices: one for cross-lingual and one for monolingual models. The last column shows cases of *incorrect* heads and the other columns show label predictions when the heads are *correct*, i.e., each row summing to 100%. Here, blue cells highlight confusions that are more common for the cross-lingual model, while red cells highlight those more common for the monolingual model. For head prediction (last column), we observe that monolingual model makes higher errors especially for nominals and modifier words. In cases when both both models predict the correct heads, we observe that cross-lingual training gives further improvements in predicting most of the labels. In particular, regarding the "rootless" trees discussed before, we see evidence that cross-lingual training helps in predicting the correct root index, and the correct *root* label.

[8]The parsing model enforces the constraint that every tree should have a head, i.e., an arc pointing from a dummy root to a node in the tree. It does not, however, enforce that this arc be labeled *root*—the model must learn the labeling.

| | | CROSS-LINGUAL | |
|---|---|---|---|
| Language | zero-shot | +fastText | +Morph |
| Galician | 51.9 | **72.8** | 71.0 |
| Kazakh | 12.5 | 27.7 | **28.4** |
| Kazakh (translit.) | 21.2 | 31.1 | **36.7** |

Table 7: LAS results on **development sets**. *zero-shot* denotes results where we predict using model trained only on the source treebank.

## 5 Parsing truly low-resource languages

Now we turn to two truly low-resource treebanks: Galician and Kazakh. These treebanks are most analogous to the North Sami $\mathcal{T}_{10}$ setting and therefore we apply the best approach, cross-lingual training with MORPH augmentation. Table 1 provides the statistics of the augmented data. For Galician, we use the Portuguese treebank as source while for Kazakh we use Turkish. Portuguese and Galician have high vocabulary overlap; 62.9% of Galician tokens appear in Portuguese data, while for Turkish and Kazakh they do not share vocabulary since they use different writing systems. However, after transliterating them into the same basic Latin alphabet, we observe that 9.5% of Kazakh tokens appear in the Turkish data. Both language pairs also share many (token-level) character trigrams: 96.0% for Galician-Portuguese and 66.3% for transliterated Kazakh-Turkish.

To compare our best approach, we create two baselines: (1) a pre-trained parsing model of the source treebank (zero-shot learning), and (2) a cross-lingual model initialized with *monolingual* pre-trained word embeddings. The first serves as a weak baseline, in a case where training on the target treebank is not possible (e.g., Kazakh only has 15 sentences for training). The latter serves as

| | | | CROSS-LINGUAL | | |
|---|---|---|---|---|---|
| | baseline | best system | +fastText | +Morph | rank |
| Galician | 66.16 | 74.25 | **70.46** | 69.21 | 10/27 |
| Kazakh (translit.) | 24.21 | 31.93 | 25.28 | **28.23** | 2/27 |

Table 8: Comparison to CoNLL 2018 UD Shared Task on **test sets**. *best system* is the state-of-the-art model for each treebank: UDPipe-Future (Straka, 2018) for Galician and Uppsala (Smith et al., 2018) for Kazakh. *rank* shows our best model position in the shared task ranking for each treebank.

a strong baseline, in a case when we have access to pre-trained word embeddings, for the source and/or the target languages.

We treat a pre-trained word embedding as an external embedding, and concatenate it with the other representations, i.e., modifying Eq. 3 to $\mathbf{x}_i = [\mathbf{e}_w(w_i); \mathbf{e}_p(w_i); \mathbf{e}_c(w_i); \mathbf{l}_k]$, where $\mathbf{e}_p(w_i)$ represents a pre-trained word embedding of $w_i$, which we update during training. We use the pre-trained monolingual fastText embeddings (Bojanowski et al., 2017).[9] We concatenate the source and target pre-trained word embeddings.[10] For our experiments with transliteration (§2.4), we transliterate the entries of both the source and the target pre-trained word embeddings.

### 5.1 Experimental results

Table 7 reports the LAS performance on the development sets. MORPH augmentation improves performance over the zero-shot baseline and achieves comparable or better LAS with a cross-lingual model trained with pre-trained word embeddings.

Next, we look at the effects of transliteration (see Kazakh vs Kazakh (translit.) in Table 7). In the zero-shot experiments, simply mapping both Turkish and Kazakh characters to the Latin alphabet improves accuracy from 12.5 to 21.2 LAS. Cross-lingual training with MORPH further improves performance to 36.7 LAS.

### 5.2 Comparison with CoNLL 2018

To see how our best approach (i.e., cross-lingual model with MORPH augmentation) compares with the current state-of-the-art models, we compare it to the recent results from CoNLL 2018 shared task. Training state-of-the-art models may require lots of engineering and data resources. Our goal, however, is not to achieve the best performance, but rather to systematically investigate how far simple approaches can take us. We report performance of the following: (1) the shared task baseline model (UDPipe v1.2; Straka and Straková, 2017) and (2) the best system for each treebank, (3) our best approach, and (4) a cross-lingual model with fastText embeddings.

Table 8 presents the overall comparison on the test sets. For each treebank, we apply the same sentence segmentation and tokenization used by each best system.[11] We see that our approach outperforms the baseline models on both languages. For Kazakh, our model (with transliteration) achieves a competitive LAS (28.23), which would be the second position in the shared task ranking. As comparison, the best system for Kazakh (Smith et al., 2018) trained a multi-treebank model with four source treebanks, while we only use one source treebank. Their system use predicted POS as input, while ours depends solely on words and characters. The use of more treebanks and predicted POS is beyond the scope of our paper, but it is interesting that our approach can achieve the second best with such minimal resources. For Galician, our best approach outperforms baseline by 8.09 LAS points. Note that, Galician treebank does not come with training data. We use 50:50 train/dev split, while other teams might use higher split for training (for example, the best system (Straka, 2018) uses 90:10 train/dev split). Since we treat Galician as our test data, we did not tune on the proportion for training data, but we guess that this is the main reason why our system achieve rank 10 out of 27.

Compared to cross-lingual models with fastText embeddings (fastText vs. MORPH), we observe that our approach achieve better or comparable performance, showing its potential when there is

[9] The embeddings are available at https://fasttext.cc/docs/en/pretrained-vectors.html.

[10] If a word occurs in both source and target, we use the word embedding of the source language.

[11] UD shared task only provides unsegmented (i.e., sentence-level and token-level) raw test data. However, participants were allowed to use predicted segmentation and tokenization provided by the baseline UDPipe model.

not enough monolingual data available for training word embeddings.

## 6 Conclusions

In this paper, we investigated various low-resource parsing scenarios. We demonstrate that in the extremely low-resource setting, data augmentation improves parsing performance both in monolingual and cross-lingual settings. We also show that transfer learning is possible with *lexicalized* parsers. In addition, we show that transfer learning between two languages with different writing systems is possible, and future work should consider transliteration for other language pairs.

While we have not exhausted all the possible techniques (e.g., use of external resources (Rasooli and Collins, 2017; Rosa and Mareček, 2018), predicted POS (Ammar et al., 2016), multiple source treebanks (Lim et al., 2018; Stymne et al., 2018), among others), we show that simple methods which leverage the linguistic annotations in the treebank can improve low-resource parsing. Future work might explore different augmentation methods, such as the use of *synthetic* source treebanks (Wang and Eisner, 2018) or contextualized language model (Peters et al., 2018; Howard and Ruder, 2018; Devlin et al., 2018) for scoring the augmented data (e.g., using perplexity).

Finally, while the techniques presented in this paper might be applicable to other low-resource languages, we want to also highlight the importance of understanding the characteristics of languages being studied. For example, we showed that although North Sami and Finnish do not share vocabulary, cross-lingual training is still helpful because they share similar syntactic structures. Different language pairs might benefit from other types of similarity (e.g., morphological) and investigating this would be another interesting future work for low-resource dependency parsing.

## Acknowledgments

Clara Vania is supported by the Indonesian Endowment Fund for Education (LPDP), the Centre for Doctoral Training in Data Science, funded by the UK EPSRC (grant EP/L016427/1), and the University of Edinburgh. Anders Søgaard is supported by a Google Focused Research Award. We thank Aibek Makazhanov for helping with Kazakh transliteration, and Miryam de Lhoneux for parser implementation. We also thank Andreas Grivas, Maria Corkery, Ida Szubert, Gozde Gul Sahin, Sameer Bansal, Marco Damonte, Naomi Saphra, Nikolay Bogoychev, and anonymous reviewers for helpful discussion of this work and comments on previous drafts of the paper.

## References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2016. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. *CoRR*, abs/1608.04207.

Željko Agić. 2017. Cross-Lingual Parser Selection for Low-Resource Languages. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*, pages 1–10. Association for Computational Linguistics.

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah Smith. 2016. Many Languages, One Parser. *Transactions of the Association for Computational Linguistics*, 4:431–444.

Toms Bergmanis, Katharina Kann, Hinrich Schütze, and Sharon Goldwater. 2017. Training data augmentation for low-resource morphological inflection. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 31–39. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*.

Matthew S. Dryer and Martin Haspelmath, editors. 2013. *WALS Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. Data augmentation for low-resource neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 567–573. Association for Computational Linguistics.

Agnieszka Falenska and Özlem Çetinoğlu. 2017. Lexicalized vs. delexicalized parsing in low-resource scenarios. In *Proceedings of the 15th International Conference on Parsing Technologies*, pages 18–24. Association for Computational Linguistics.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless Green Recurrent Networks Dream Hierarchically. In *Proceedings of the 2018 Conference of the North*

*American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1195–1205. Association for Computational Linguistics.

Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339. Association for Computational Linguistics.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Nat. Lang. Eng.*, 11(3):311–325.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Marco Kuhlmann, Carlos Gómez-Rodrìguez, and Giorgio Satta. 2011. Dynamic Programming Algorithms for Transition-Based Dependency Parsers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 673–682. Association for Computational Linguistics.

Miryam de Lhoneux, Johannes Bjerva, Isabelle Augenstein, and Anders Søgaard. 2018. Parameter sharing between dependency parsers for related languages. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4992–4997. Association for Computational Linguistics.

Miryam de Lhoneux, Yan Shao, Ali Basirat, Eliyahu Kiperwasser, Sara Stymne, Yoav Goldberg, and Joakim Nivre. 2017a. From Raw Text to Universal Dependencies – Look, No Tags! In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies.*, Vancouver, Canada.

Miryam de Lhoneux, Sara Stymne, and Joakim Nivre. 2017b. Arc-Hybrid Non-Projective Dependency Parsing with a Static-Dynamic Oracle. In *Proceedings of the The 15th International Conference on Parsing Technologies (IWPT).*, Pisa, Italy.

KyungTae Lim, Niko Partanen, and Thierry Poibeau. 2018. Multilingual dependency parsing for low-resource languages: Case studies on north saami and komi-zyrian. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*. European Language Resource Association.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1520–1530, Lisbon, Portugal. Association for Computational Linguistics.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-Source Transfer of Delexicalized Dependency Parsers. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 62–72. Association for Computational Linguistics.

Joakim Nivre. 2009. Non-Projective Dependency Parsing in Expected Linear Time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 351–359. Association for Computational Linguistics.

Joakim Nivre, Mitchell Abrams, Željko Agić, Lars Ahrenberg, Lene Antonsen, Maria Jesus Aranzabe, Gashaw Arutie, Masayuki Asahara, Luma Ateyah, Mohammed Attia, Aitziber Atutxa, Liesbeth Augustinus, Elena Badmaeva, Miguel Ballesteros, Esha Banerjee, Sebastian Bank, Verginica Barbu Mititelu, John Bauer, Sandra Bellato, Kepa Bengoetxea, Riyaz Ahmad Bhat, Erica Biagetti, Eckhard Bick, Rogier Blokland, Victoria Bobicev, Carl Börstell, Cristina Bosco, Gosse Bouma, Sam Bowman, Adriane Boyd, Aljoscha Burchardt, Marie Candito, Bernard Caron, Gauthier Caron, Gülşen Cebiroğlu Eryiğit, Giuseppe G. A. Celano, Savas Cetin, Fabricio Chalub, Jinho Choi, Yongseok Cho, Jayeol Chun, Silvie Cinková, Aurélie Collomb, Çağrı Çöltekin, Miriam Connor, Marine Courtin, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Arantza Diaz de Ilarraza, Carly Dickerson, Peter Dirix, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Marhaba Eli, Ali Elkahky, Binyam Ephrem, Tomaž Erjavec, Aline Etienne, Richárd Farkas, Hector Fernandez Alcalde, Jennifer Foster, Cláudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Moa Gärdenfors, Kim Gerdes, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökırmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Céline Guillot-Barbance, Nizar Habash, Jan Hajič, Jan Hajič jr., Linh Hà Mỹ, Na-Rae Han, Kim Harris, Dag Haug, Barbora Hladká, Jaroslava Hlaváčová, Florinel Hociung, Petter Hohle, Jena Hwang, Radu Ion, Elena Irimia, Tomáš Jelínek, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşıkara, Sylvain Kahane, Hiroshi Kanayama, Jenna Kanerva, Tolga Kayadelen, Václava Kettnerová, Jesse Kirchner, Natalia Kotsyba, Simon Krek, Sookyoung Kwak, Veronika Laippala, Lorenzo Lambertino, Tatiana Lando, Septina Dian Larasati, Alexei Lavrentiev, John Lee, Phương Lê Hồng, Alessandro Lenci, Saran Lertpradit, Herman Leung, Cheuk Ying Li, Josie Li, Keying Li, KyungTae Lim, Nikola Ljubešić, Olga Loginova, Olga Lyashevskaya, Teresa Lynn, Vivien Macke-

tanz, Aibek Makazhanov, Michael Mandl, Christopher Manning, Ruli Manurung, Cătălina Mărănduc, David Mareček, Katrin Marheinecke, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Gustavo Mendonça, Niko Miekka, Anna Missilä, Cătălin Mititelu, Yusuke Miyao, Simonetta Montemagni, Amir More, Laura Moreno Romero, Shinsuke Mori, Bjartur Mortensen, Bohdan Moskalevskyi, Kadri Muischnek, Yugo Murawaki, Kaili Müürisep, Pinkey Nainwani, Juan Ignacio Navarro Horñiacek, Anna Nedoluzhko, Gunta Nešpore-Bērzkalne, Lương Nguyễn Thị, Huyền Nguyễn Thị Minh, Vitaly Nikolaev, Rattima Nitisaroj, Hanna Nurmi, Stina Ojala, Adédayọ̀ Olúòkun, Mai Omura, Petya Osenova, Robert Östling, Lilja Øvrelid, Niko Partanen, Elena Pascual, Marco Passarotti, Agnieszka Patejuk, Siyao Peng, Cenel-Augusto Perez, Guy Perrier, Slav Petrov, Jussi Piitulainen, Emily Pitler, Barbara Plank, Thierry Poibeau, Martin Popel, Lauma Pretkalniņa, Sophie Prévost, Prokopis Prokopidis, Adam Przepiórkowski, Tiina Puolakainen, Sampo Pyysalo, Andriela Rääbis, Alexandre Rademaker, Loganathan Ramasamy, Taraka Rama, Carlos Ramisch, Vinit Ravishankar, Livy Real, Siva Reddy, Georg Rehm, Michael Rießler, Larissa Rinaldi, Laura Rituma, Luisa Rocha, Mykhailo Romanenko, Rudolf Rosa, Davide Rovati, Valentin Roșca, Olga Rudina, Shoval Sadde, Shadi Saleh, Tanja Samardžić, Stephanie Samson, Manuela Sanguinetti, Baiba Saulīte, Yanin Sawanakunanon, Nathan Schneider, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Mo Shen, Atsuko Shimada, Muh Shohibussirri, Dmitry Sichinava, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Isabela Soares-Bastos, Antonio Stella, Milan Straka, Jana Strnadová, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Dima Taji, Yuta Takahashi, Takaaki Tanaka, Isabelle Tellier, Trond Trosterud, Anna Trukhina, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Zdeňka Urešová, Larraitz Uria, Hans Uszkoreit, Sowmya Vajjala, Daniel van Niekerk, Gertjan van Noord, Viktor Varga, Veronika Vincze, Lars Wallin, Jonathan North Washington, Seyi Williams, Mats Wirén, Tsegay Woldemariam, Tak-sum Wong, Chunxiao Yan, Marat M. Yavrumyan, Zhuoran Yu, Zdeněk Žabokrtský, Amir Zeldes, Daniel Zeman, Manying Zhang, and Hanzhi Zhu. 2018. Universal Dependencies 2.2. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Mohammad Sadegh Rasooli and Michael Collins. 2017. Cross-lingual syntactic transfer with limited resources. *Transactions of the Association for Computational Linguistics*, 5:279–293.

Rudolf Rosa and David Mareček. 2018. Cuni x-ling: Parsing under-resourced languages in conll 2018 ud shared task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 187–196. Association for Computational Linguistics.

Gozde Gul Sahin and Mark Steedman. 2018. Data Augmentation via Dependency Tree Morphing for Low-Resource Languages. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5004–5009. Association for Computational Linguistics.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural mt learn source syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1526–1534. Association for Computational Linguistics.

Aaron Smith, Bernd Bohnet, Miryam de Lhoneux, Joakim Nivre, Yan Shao, and Sara Stymne. 2018. 82 Treebanks, 34 Models: Universal Dependency Parsing with Multi-Treebank Models. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 113–123. Association for Computational Linguistics.

Anders Søgaard. 2011. Data point selection for cross-language adaptation of dependency parsers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 682–686. Association for Computational Linguistics.

Milan Straka. 2018. UDPipe 2.0 Prototype at CoNLL 2018 UD Shared Task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 197–207. Association for Computational Linguistics.

Milan Straka and Jana Straková. 2017. Tokenizing, POS Tagging, Lemmatizing and Parsing UD 2.0 with UDPipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada. Association for Computational Linguistics.

Sara Stymne, Miryam de Lhoneux, Aaron Smith, and Joakim Nivre. 2018. Parser Training with Heterogeneous Treebanks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 619–625. Association for Computational Linguistics.

Jörg Tiedemann and Željko Agic. 2016. Synthetic treebanking for cross-lingual dependency parsing. *J. Artif. Int. Res.*, 55(1):209–248.

Yulia Tsvetkov, Sunayana Sitaram, Manaal Faruqui, Guillaume Lample, Patrick Littell, David Mortensen, Alan W Black, Lori Levin, and Chris Dyer. 2016. Polyglot neural language models: A case study in cross-lingual phonetic representation learning. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1357–1366. Association for Computational Linguistics.

Sara Veldhoen, Dieuwke Hupkes, and Willem H. Zuidema. 2016. Diagnostic classifiers revealing how neural networks process hierarchical structure. In *CoCo@NIPS*.

Dingquan Wang and Jason Eisner. 2018. Synthetic data made to order: The case of parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1325–1337. Association for Computational Linguistics.

Marcos Zampieri, Shervin Malmasi, Nikola Ljubešić, Preslav Nakov, Ahmed Ali, Jörg Tiedemann, Yves Scherrer, and Noëmi Aepli. 2017. Findings of the vardial evaluation campaign 2017. In *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, pages 1–15. Association for Computational Linguistics.

Daniel Zeman, Jan Haji{č}, Martin Popel, Martin Potthast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. 2018. CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–21. Association for Computational Linguistics.

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajic, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, Francis Tyers, Elena Badmaeva, Memduh Gokirmak, Anna Nedoluzhko, Silvie Cinkova, Jan Hajic jr., Jaroslava Hlavacova, Václava Kettnerová, Zdenka Uresova, Jenna Kanerva, Stina Ojala, Anna Missilä, Christopher D. Manning, Sebastian Schuster, Siva Reddy, Dima Taji, Nizar Habash, Herman Leung, Marie-Catherine de Marneffe, Manuela Sanguinetti, Maria Simi, Hiroshi Kanayama, Valeria dePaiva, Kira Droganova, Héctor Martínez Alonso, Çağrı Çöltekin, Umut Sulubacak, Hans Uszkoreit, Vivien Macketanz, Aljoscha Burchardt, Kim Harris, Katrin Marheinecke, Georg Rehm, Tolga Kayadelen, Mohammed Attia, Ali Elkahky, Zhuoran Yu, Emily Pitler, Saran Lertpradit, Michael Mandl, Jesse Kirchner, Hector Fernandez Alcalde, Jana Strnadová, Esha Banerjee, Ruli Manurung, Antonio Stella, Atsuko Shimada, Sookyoung Kwak, Gustavo Mendonca, Tatiana Lando, Rattima Nitisaroj, and Josie Li. 2017. CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–19. Association for Computational Linguistics.

Daniel Zeman and Philip Resnik. 2008. Cross-language parser adaptation between related languages. In *Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages*.

# Target Language-Aware Constrained Inference for Cross-lingual Dependency Parsing

**Tao Meng**
University of California, Los Angeles
tmeng@cs.ucla.edu

**Nanyun Peng**
University of Southern California
npeng@isi.edu

**Kai-Wei Chang**
University of California, Los Angeles
kw@kwchang.net

## Abstract

Prior work on cross-lingual dependency parsing often focuses on capturing the commonalities between source and target languages and overlooks the potential of leveraging linguistic properties of the languages to facilitate the transfer. In this paper, we show that weak supervisions of linguistic knowledge for the target languages can improve a cross-lingual graph-based dependency parser substantially. Specifically, we explore several types of *corpus linguistic statistics* and compile them into *corpus-wise constraints* to guide the inference process during the test time. We adapt two techniques, Lagrangian relaxation and posterior regularization, to conduct inference with corpus-statistics constraints. Experiments show that the Lagrangian relaxation and posterior regularization inference improve the performances on 15 and 17 out of 19 target languages, respectively. The improvements are especially significant for target languages that have different word order features from the source language.

## 1 Introduction

Natural language processing (NLP) techniques have achieved remarkable performance in a variety of tasks when sufficient training data is available. However, obtaining high-quality annotations for low-resource language tasks is challenging, and this poses great challenges to process low-resource languages. To bridge the gap, cross-lingual transfer has been proposed to transfer models trained on high-resource languages (e.g., English) to low-resource languages (e.g., Tamil) to combat the resource scarcity problem. Recent studies have demonstrated successes of transferring models across languages without retraining for NLP tasks, such as named entity recognition (Xie et al., 2018), dependency parsing (Tiedemann, 2015; Agić et al., 2014), and question answering (Joty et al., 2017), using a shared multi-lingual word embedding space (Smith et al., 2017) or delexicalization approaches (Zeman and Resnik, 2008; McDonald et al., 2013).

One key challenge for cross-lingual transfer is the *differences* among languages; for example, languages may have different word orders. When transferring a model learned from a source language to target languages, the performance may drop significantly due to the differences. To tackle this problem, various approaches have been proposed to better capture the *commonalities* between the source and the target languages (McDonald et al., 2011; Guo et al., 2016; Täckström et al., 2013; Agić, 2017; Ahmad et al., 2019); however, they overlook the potential to leverage linguistic knowledge about the target language to account for the *differences* between the source and the target languages to facilitate the transfer.

In this paper, we propose a complementary approach that studies how to leverage the linguistic knowledge about the target languages to help the transfer. Specifically, we use corpus linguistic statistics of the target languages as weak supervision signals to guide the test-time *inference* process when parsing with a graph-based parser. This approach is effective as the model only need to be trained *once* on the source language and applied to many target languages using different constraints *without retraining* the model.

We argue that certain corpus linguistic statistics such as the word order (e.g., how often an adjective appears before or after a noun) can be easily obtained from available resources such as World Atlas of Language Structures (WALS) (Dryer and Haspelmath, 2013). To incorporate the corpus linguistic statistics to a cross-lingual parser, we compile them into corpus-wise constraints and adopt two families of methods: 1) Lagrangian relaxation (LR) and 2) posterior regularization (PR) to



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1117–1128, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

solve the constrained inference problem. The algorithms take the original graph-based parsing inference as a sub-routine, and LR iteratively adjusts the pair-wise potentials until the constraints are (loosely) satisfied, while PR finds a feasible distribution and do inference based on that. The constrained inference framework is general and supports any knowledge that can be formulated as a first-order logic (Roth and Yih, 2004).

We evaluate the proposed approach under the single-source transfer setting using English as the source language and test on 19 target languages covering a broad range of language families with low-resource languages such as Tamil and Welsh. We demonstrate that by adding three simple corpus-wise constraints derived from WALS features, the performances improve in 15 and 17 out of 19 languages when using Lagrangian relaxation and posterior regularization techniques, respectively. The improvements are especially substantial when the target language features are distant from the source language. For example, our framework improves the UAS score of Urdu by 15.7%, and Tamil by 7.3%.[1]

## 2 Constrained Cross-Lingual Parsing

Our work focuses on the graph-based dependency parser (McDonald et al., 2005) in the zero-shot single-source transfer setting as in Ahmad et al. (2019). However, the proposed algorithms can be extended other transfer settings. Given a trained model, we derive corpus-statistics constraints and apply them to correct errors caused by word order differences between the source and the target language *during the inference time*. Figure 1 shows an example of how constraints can influence the inference results.

In this section, we first give a quick review of the graph-based parser and introduce the notations. We then discuss how to formulate corpus-wise constraints based on corpus linguistic statistics for guiding the graph-based parser.

### 2.1 Background: Graph-Based Parser

A graph-based parser learns a scoring function for every pair of words in a sentence and conducts inference to derive a directed spanning tree with the highest accumulated score. Formally, given the $k$-th sentence $\mathbf{w}_k = (w_{k1}, \ldots, w_{kL(k)})$ where $L(k)$ denotes the length of the $k$-th sentence, a graph-based parser learns a score matrix $S^{(k)}$, where $S^{(k)}_{ij}$ denotes the score to form an arc from word $w_{ki}$ to word $w_{kj}$. Let $y_k$ be an indicator function that $y_k(i,j) \in \{0,1\}$ denotes the arc from $w_{ki}$ to $w_{kj}$. The maximum directed spanning tree inference can be formulated as an integer linear programming (ILP) problem:

$$y^*_k = \arg\max_{y_k \in \mathcal{Y}_k} \sum_{i,j} S^{(k)}_{ij} y_k(i,j), \qquad (1)$$

where $\mathcal{Y}_k$ is the set of legal dependency trees of sentence $k$. In recent years, neural network approaches (Kiperwasser and Goldberg, 2016; Wang and Chang, 2016; Kuncoro et al., 2016; Dozat and Manning, 2017) have been applied to modeling the scoring matrix $S^{(k)}$ and have achieved great performance in dependency parsing.

From the probabilistic point of view, if we assume for different $i, j$, the edge probabilities $P(y_k(i,j) = 1|\mathbf{w}_k)$ are mutually conditional independent, the probability of a whole parse tree can be written as

$$P(y_k|\mathbf{w}_k) = \prod_{i,j} P(y_k(i,j) = 1|\mathbf{w}_k)^{y_k(i,j)}. \quad (2)$$

If we set $S^{(k)}_{ij} = \log P(y_k(i,j) = 1|\mathbf{w}_k) + Z'_j$, where $Z'_j$ is a constant term, then Eq. (1) can be regarded as the following maximum a posteriori (MAP) inference problem:

$$\begin{aligned} y^*_k &= \arg\max_{y_k \in \mathcal{Y}_k} P(y_k|\mathbf{w}_k) \\ &= \arg\max_{y_k \in \mathcal{Y}_k} \sum_{i,j} \log P(y_k(i,j) = 1|\mathbf{w}_k) y_k(i,j). \end{aligned} \qquad (3)$$

### 2.2 Corpus-Wise Constraints

Given the inference problems as in equations (1) and (3), additional constraints can be imposed to incorporate expert knowledge about the languages to help yield a better parser. Instance-level constraints have been explored in the literature of dependency parsing, both in the monolingual (Dryer, 2007) and cross-lingual transfer (Täckström et al., 2013) settings. However, most word order features for a language are non-deterministic and cannot be compiled into instance-wise constraints.

In this work, we introduce corpus-wise constraints to leverage the non-deterministic features for cross-lingual parser. We compile the following two types of corpus-wise constraints based on corpus linguistics statistics:

[1] The code and data are available at https://github.com/MtSomeThree/CrossLingualDependencyParsing.

- *Unary* constraints consider statistics regarding a particular POS tag ($POS$).
- *Binary* constraints consider statistics regarding a pair of POS tags ($POS_1, POS_2$).

Specifically, a unary constraint specifies the ratio $r$ of the heads of a particular $POS$ appears on the left of that $POS$.[2] Similarly, a binary constraint specifies the ratio $r$ of $POS_1$ being on the left of $POS_2$ when there is an arc between $POS_1$ and $POS_2$.

The ratios $r$ for the constraints are called corpus statistics, which can be estimated in one of the following ways: a) leveraging existing linguistics resources or consulting linguists; b) leveraging a higher-resource language that is similar to the target language (e.g., Finnish and Estonian) to collect the statistics. In this paper, we explore the first option and leverage the WALS features, which provide a reference for word order typology, to estimate the ratios.

**Compile Constraints From WALS Features.** For a particular language, once we collect the corpus-statistics of a pair of POS tags, we can formulate a binary constraint. There are different ways to estimate the corpus-statistics. For example, Östling (2015) utilizes a small amount of parallel data to estimate the dominant word orders. In this paper, we simply utilize a small subset of WALS features that show the dominant order of some POS pairs (e.g. adjective and noun) in a language. They can be directly compiled into binary constraints.

Similarly, we can estimate the ratio for unary constraints based on WALS features. For a particular POS tag, we choose all WALS features related to it to formulate a feature vector $f$. The mapping from the vector $f$ to the unary constraint ratio $r$ is learnable: for each language with annotated data, we can get a WALS feature vector $f_{lang}$ and a ratio $r_{lang}$ from the annotation. We only need a small amount of data to estimate $r_{lang}$ well. Given a set of languages with feature vectors and estimated ratios, we can learn the mapping by a simple linear regression, and apply it to estimate the ratio of any target language to compile a unary constraint.

### 2.3 Formulate Constraints

In the following, we mathematically formulate the corpus-wise constraints. Note that these constraints are based on the statistics over the entire corpus. For a unary constraint $C_u^{(POS)}$, let $P$ denotes a set of word with part-of-speech tag $POS$. We define $C_u^+ := \{(k,i,j)|w_{kj} \in P \wedge i < j\}$ as the set of arcs where the head of word in $P$ is on its left and $C_u^- := \{(k,i,j)|w_{kj} \in P \wedge i > j\}$, conversely.

For a binary constraint $C_b^{(POS_1,POS_2)}$, we denotes $P_1$, $P_2$ as a set of word with part-of-speech tag $POS_1$, $POS_2$, respectively. We then define $C_b^+$ as the set of arcs with two ends $w_{ki} \in P_1$ and $w_{kj} \in P_2$, and $w_{ki}$ is on the left of $w_{kj}$. We define $C_b^-$ Similarly. Formally,

$$\begin{aligned} C_b^+ :=& \{(k,i,j)|w_{ki} \in P_1 \wedge w_{kj} \in P_2 \wedge i < j\} \\ & \cup \{(k,j,i)|w_{ki} \in P_1 \wedge w_{kj} \in P_2 \wedge i < j\}, \\ C_b^- :=& \{(k,i,j)|w_{ki} \in P_1 \wedge w_{kj} \in P_2 \wedge i > j\} \\ & \cup \{(k,j,i)|w_{ki} \in P_1 \wedge w_{kj} \in P_2 \wedge i > j\}. \end{aligned}$$

For notational simplicity, we use $C$ to denote all constraints including both the unary and binary ones. The ratio function $R(C,Y)$ for a constraint $C$ given the parse trees $Y$ can be defined as:

$$R(C,Y) = \frac{\sum_k \sum_{(i,j):(k,i,j)\in C^+} y_k(i,j)}{\sum_k \sum_{(i,j):(k,i,j)\in C^+ \cup C^-} y_k(i,j)}.$$

We want to enforce the ratio $R(C,Y)$ estimated from $Y$ to be consistent with a value $r$ (see Sec. 2.2), which formulates a constraint

$$r - \theta \leq R(C,Y) \leq r + \theta$$

where $\theta$ is a tolerance margin. Note that the instance-level hard constraint is a special case of

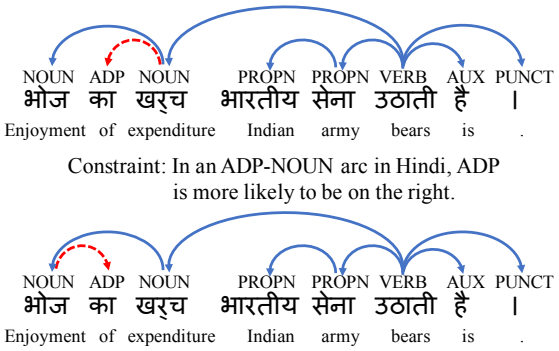


Figure 1: An running example of Hindi. On the top there is the inference result of a baseline model trained on English. In English, ADP is mostly on the left of the NOUN, so the potential of the correct ADP-NOUN arc is lower. With the help of corpus-statistics constraints, the potential is adjusted and the model gets correct inference result as shown in the bottom. The dashed lines highlight the difference.

[2] The ratio for the head being on the right of that $POS$ is thereby $1 - r$.

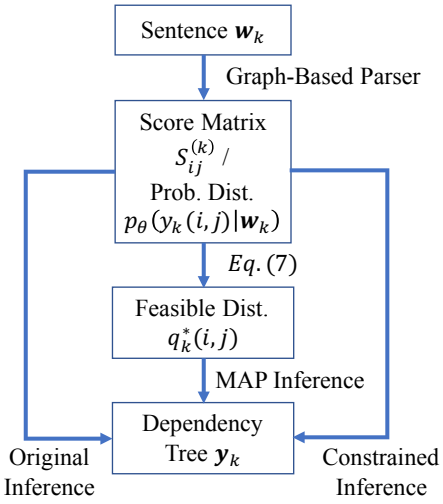


Figure 2: The pipelines of the baseline method (left), Lagrangian relaxation (right) and posterior regularization (middle). Lagrangian relaxation converts constrained inference to an unconstrained optimization problem using Lagrange's method. Posterior regularization method is working on the distribution space. For a given distribution, PR finds the closest feasible distribution and conduct MAP inference.

the corpus-statistics constraint when $r = 0$ or $r = 1$.

Given a set of corpus-statistics constraints $\mathcal{C} = \{C_1, C_2, \ldots, C_n\}$ with corresponding corpus statistics $\mathbf{r} = \{r_1, r_2, \ldots, r_n\}$, the objective of the constrained inference is:

$$\begin{aligned} \max_{Y \in \mathcal{Y}} \quad & \sum_k \sum_{i,j \le L(k)} S_{ij}^{(k)} y_k(i,j), \\ \text{s.t.} \quad & r_i - \theta_i \le R(C_i, Y) \le r_i + \theta_i, i \in [N], \end{aligned} \tag{4}$$

where $\mathcal{Y}$ denotes the set of all possible dependency trees. As all the constraints can be written as a linear inequality with respect to $y_k(i,j)$. Eq. (4) is an ILP.

## 3 Inference with Corpus-Statistics Constraints

The ILP problem in Eq. (4) is in general an NP-hard problem; especially, it involves variables associated with the entire corpus. Without constraints, Eq. (4) can be decoupled into $K$ subproblems, and the inference with respect to each sentence can be solved independently as in Eq. (1). In this way, an efficient inference algorithm such as maximum directed spanning tree algorithm (Chu and Liu, 1965) can be used.

However, with the corpus-wise constraints, directly solving Eq. (4) is infeasible. Therefore, we explore two algorithms for inference with corpus-statistics constraints: Lagrangian relaxation and posterior regularization (Ganchev et al., 2010). The Lagrangian relaxation algorithm introduces Lagrangian multipliers to relax the constraint optimization problem to an unconstrained optimization problem, and estimates the Lagrangian multipliers with gradient-based methods. The posterior regularization algorithm uses the constraints of the target language to define a feasible set of parse tree distributions, and find a feasible distribution that is closest to the parse tree distribution trained on the source language by minimizing the KL-divergence. The constrained inference problem can then be converted into an MAP inference problem on the best feasible distribution. Figure 2 illustrates the procedure of the original inference, Lagrangian relaxation, and posterior regularization.

### 3.1 Lagrangian Relaxation

Lagrangian relaxation has been applied in various NLP applications (Rush and Collins, 2012, 2011). In Eq. (4), each constraint $C_i$ involves two inequality constraints: $R(C_i, Y) - r_i + \theta \ge 0$, and $r_i + \theta - R(C_i, Y) \ge 0$. Instead of treating these two constraints separately, we consider a heuristic to optimize with equality constraints $R(C_i, Y) = r_i, i \in [N]$ and terminate earlier when constraints in Eq. (4) are satisfied. Despite this approach does not guarantee the solution is optimal if all the constraints are satisfied as the original Lagrangian relaxation algorithm does, in practice, the inference converges faster (as the number of Lagrangian multipliers is half) and the parsing performance maintains.

In the following, we derive the constrained inference algorithm for corpus-statistics constraints. First, we rewrite the equality constraint $R(C, Y) = r$ by substituting $R(C, Y)$ with Eq. (2.3):

$$\begin{aligned} &(1-r) \sum_k \sum_{(i,j):(k,i,j) \in C_i^+} y_k(i,j) \\ &-r \sum_k \sum_{(i,j):(k,i,j) \in C_i^-} y_k(i,j) = 0, \end{aligned} \tag{5}$$

We use F(C) to denote the left-hand-side of Eq. (5), which is linear w.r.t. $y_k$. Then, the Lagrangian relaxation of the constrained inference problem can be written as:

$$L(Y, \lambda; \mathcal{C}) = \sum_k \sum_{i,j} S_{ij}^{(k)} y_k(i,j) + \sum_{i \in [N]} \lambda_i F(C_i),$$

**Algorithm 1** Lagrangian Relaxation for Constraint Inference

**Input:** Constraints $\mathcal{C} = \{C_i\}_{i=1}^N$, corresponding ratio $\mathbf{r} = \{r_i\}_{i=1}^N$, tolerance margin $\theta = \{\theta_i\}_{i=1}^N$, learning rate decay $\eta$, initial learning rate $\alpha_0$

**Output:** parse trees $\hat{Y}$

$\alpha \leftarrow \alpha_0$
$\lambda_i \leftarrow 0,\ i \in [N]$
**repeat**
  $\hat{Y} \leftarrow \arg\max_{Y \in \mathcal{Y}} L(Y, \lambda; \mathcal{C})$
  $\hat{r_i} \leftarrow R(C_i, \hat{Y}),\ i \in [N]$
  **if** $\forall i, abs(r_i - \hat{r_i}) \leq \theta$ **then**
    **return** $\hat{Y}$
  $\lambda_i \leftarrow \alpha(\hat{r_i} - r_i),\ i \in [N]$
  $\alpha \leftarrow \eta\alpha$
**until** MAX_ITER times
**return** $\hat{Y}$

where $\lambda_i$ is called Lagrangian multiplier. It is well-known that we can solve the dual form of the constrained inference problem:

$$\max_{Y \in \mathcal{Y}} \min_{\lambda \geq 0} L(Y, \lambda; \mathcal{C}).$$

To solve the dual form, we initialize $\lambda_i$ to be 0. At iteration $t$, we firstly conduct an constraint-augmented inference with a fixed $\lambda^{(t)}$:

$$\hat{Y}^{(t)} = \max_{Y \in \mathcal{Y}} L(Y, \lambda^{(t)}; \mathcal{C}). \tag{6}$$

As $F(C)$ is a linear function w.r.t $y_k$, we combine it with $S_{ij}^{(k)} y_k(i,j)$. In this way, the inference problem Eq. (6) can be treated as a special case of Eq. (1) with a different scoring matrix $S^{(k)}$. In this way, we can treat the inference on every sentence independently and leverage existing inference techniques.

After solving the constraint-augmented inference, we compute the ratio of every constraint $\hat{r_i}^{(t)} = R(C_i, \hat{Y}^{(t)})$, and use gradient ascent algorithm to update the Lagrangian multipliers

$$\lambda_i^{(t+1)} = \lambda_i^{(t)} + \alpha^{(t)}(r_i - \hat{r_i}).$$

Here $\alpha^{(t)}$ denote the step size at iteration $t$. The algorithm is shown in Algorithm 1.

### 3.2 Posterior Regularization

From a probabilistic point of view, the parser model learns parameters $\theta$ to realize Eq. (2). During the inference, the model predict a probability distribution $p_\theta(\mathbf{Y}|\mathbf{W})$ over possible parse trees given a sentence. The posterior regularization algorithm first defines a feasible set of the probability distributions w.r.t the given constraints, and looks for the closest feasible distribution $q^*(\mathbf{Y})$ to the model distribution $p_\theta(\mathbf{Y}|\mathbf{W})$. The best parse tree is given by $\arg\max_{\mathbf{Y}} q^*(\mathbf{Y})$. Specifically, we define the feasible set as:

$$Q = \{q(\mathbf{Y}) | r_i - \theta_i \leq R(C_i, q(\mathbf{Y})) \leq r_i + \theta_i, i \in [N]\},$$

where $R(C, q) = \frac{\sum_k \sum_{(i,j):(k,i,j) \in C^+} q_k(i,j)}{\sum_k \sum_{(i,j):(k,i,j) \in C^+ \cup C^-} q_k(i,j)}$, with $q_k(i,j) = E_{Y \sim q(Y)}[y_k(i,j)]$.

To measure the distance between two distribution, we use the KL-divergence, and find the best feasible distribution $q^*(\mathbf{Y})$:

$$q^*(\mathbf{Y}) = \arg\min_{q \in Q} KL(q(\mathbf{Y}) \| p_\theta(\mathbf{Y}|\mathbf{W})). \tag{7}$$

If the feasible set has the expectation form:

$$\{q | \mathbb{E}_{\mathbf{Y} \sim q}[\phi(\mathbf{Y})] \leq \mathbf{b}\}, \tag{8}$$

Eq. (7) has a simple close form solution (Ganchev et al., 2010):

$$q^*(\mathbf{Y}) = \frac{p_\theta(\mathbf{Y}|\mathbf{W}) \exp(-\lambda^* \cdot \phi(\mathbf{Y}))}{Z(\lambda^*)}, \tag{9}$$

where $\lambda^*$ is the solution of

$$\begin{aligned} \lambda^* &= \arg\max_{\lambda \geq 0} -\mathbf{b} \cdot \lambda - \log Z(\lambda), \\ Z(\lambda) &= \sum_{\mathbf{Y}'} p_\theta(\mathbf{Y}'|\mathbf{W}) \exp(-\lambda^* \cdot \phi(\mathbf{Y}')). \end{aligned} \tag{10}$$

In the rest of this section, we first show that the feasible set $Q$ we considered above can be reformulated in the form of Eq. (8), and then we discuss how to solve Eq. (10). To show that the inequality $R(C, q(\mathbf{Y})) \leq r$, in $Q$ can be formulated in the form of Eq. (8), we set

$$\phi_C(\mathbf{Y}) = \sum_{k,i,j} y_k(i,j) \phi_C(k,i,j),$$

and

$$\phi_C(k,i,j) = \begin{cases} 1 - r & (k,i,j) \in C^+ \\ -r & (k,i,j) \in C^- \\ 0 & (k,i,j) \notin C^+ \cup C^-. \end{cases} \tag{11}$$

Similarly, we can derive $R(C, q(\mathbf{Y})) \geq r$, into the same form and rewrite $Q$ as

$$Q = \{q(\mathbf{Y}) \mid \mathbb{E}_{\mathbf{Y} \sim q} \phi(\mathbf{Y}) \leq \mathbf{0}\},$$

**Algorithm 2** Posterior Regularization

**Input:** Constraints $\mathcal{C} = \{C_i\}_{i=1}^N$, corresponding ratio $\mathbf{r} = \{r_i\}_{i=1}^N$, tolerance margin $\theta = \{\theta_i\}_{i=1}^N$
**Output:** parse trees $\hat{Y}$

$p_\theta(y_k(i,j)=1) \leftarrow \text{normalize}\Big(\exp\Big(S_{ij}^{(k)}\Big)\Big)$
$\phi \leftarrow$ defined by Eq. (11)
$\lambda_i \leftarrow 0,\ i \in [N]$
**repeat**
  estimate $\mathbf{g} \leftarrow \partial \log Z(\lambda)/\partial\lambda$ in a batch
  update $\lambda$ based on $\mathbf{g}$
**until** MAX_ITER times
**for** each $(k,i,j)$ **do**
  $q_k(i,j) \leftarrow$ defined by Eq. (12)
$\hat{Y} \leftarrow$ MAP inference based on $q$
**return** $\hat{Y}$

where $\phi = (\phi_{C_1}, \phi_{C_2}, ..., \phi_{C_N})$ is a collection of the constraints. The detailed derivations can be found in Appendix A.

We solve Eq. (10) by sub-gradient descent[3]. Noting that there can be exponential number of terms in $Z(\lambda)$, we firstly need to factorize $Z(\lambda)$ from corpus level to instance level and arc level, and compute the gradient. The technical details are in Appendix A. With the optimal $\lambda^*$, we can compute the feasible distribution $q^*(\mathbf{Y})$ given $p_\theta$ and $\mathbf{W}$. Noting that the solution Eq. (9) can also be factorized to arc-level:

$$q_k^*(i,j) \propto p_\theta(y_k(i,j)|\mathbf{w_k}) \exp(-\lambda^* \cdot \phi(k,i,j)), \tag{12}$$

here $q_k^*(i,j)$ denote the arc-level distribution $q^*(y_k(i,j)=1)$ satisfying Eq. (3). We then do MAP inference based on $q$, which is actually a minimal spanning tree problem same as before. Algorithm 2 summarizes the process.

## 4 Experiments

In this section, we evaluate the proposed algorithms by transferring an English dependency parser to 19 target languages covering 13 language families of real low-resource languages. We first introduce the experimental setup including data selection and constraint details and then discuss the results as well as in-depth analysis.

### 4.1 Setup

**Model and Data** We train the best performing Att-Graph parser proposed in Ahmad et al. (2019) on English and transfer it to 19 target languages in UD Tree Bank v2.2 (Nivre et al., 2018).[4] The model takes words and predicted POS tags[5] as input, and achieve transfer by leveraging pre-trained multi-lingual FastText (Bojanowski et al., 2017) embeddings that project the word embeddings from different languages into the same space using an offline transformation method (Smith et al., 2017; Conneau et al., 2018). The SelfAtt-Graph model uses a Transformer (Vaswani et al., 2017) with relative position embedding as the encoder and a deep biaffine scorer (Dozat and Manning, 2017) as the decoder. We follow the setting in Ahmad et al. (2019) to train and tune only on the source language (English) and directly transfer to all the target languages. We modify their decoder to incorporate constraints with the proposed constrained inference algorithms during the transfer phase without retraining the model. All the hyper-parameters are specified in Appendix Table 4 together with hyper-parameters for the inference algorithms in Appendix Table 5.

**Constraints** We consider two types of constraints: 1) instance-level projective constraints for avoiding creating crossing arcs in the dependency trees, 2) corpus-statistics constraints constructed by the process described in Section 2.2. We consider the following three corpus-statistics constraints: $C1 = (NOUN)$, $C2 = (NOUN, ADP)$, $C3 = (NOUN, ADJ)$; intuitively, $C1$ concerns about the ratio of nouns being on the right of their heads; $C2$ concerns about the ratio of nouns being on the left of adpositions among all noun-adposition arcs; $C3$ concerns about the ratio of nouns being on the left of adjectives among all noun-adjective arcs.

For binary constraints, $C2$ and $C3$ can be directly compiled from WALS feature $85A$ and $87A$ respectively. We encode "dominant order" specified in WALS as the ratio being always greater than $0.75$ (i.e., $r = 0.875$ and $\theta = 0.125$). If there is no dominant order or the feature is missing, we set $r = 0.5$ and $\theta = 0.25$. Some WALS features like $82A, 83A$ are also about word order, but we need to specify the arc types to utilize them. For simplicity, we only consider forming constraints from the POS tags in this paper. To estimate the

[3] In implementation, we use stochastic gradient descent with Adam optimizer (Kingma and Ba, 2015)

[4] We make the selection to prioritize the coverage of language families and low resource languages. The language family information can be found in Table 1.

[5] We use predicted POS tags provided in UD v2.2.

| Family | Lang. | Features | Baseline | Lagrangian Relaxation | | | Posterior Regularization | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Oracle | WALS | ΔWALS | Oracle | WALS | ΔWALS |
| **IE.Germanic** | en | 1,1,1 | 90.5 | 90.3 | 90.4 | -0.1 | 90.4 | 90.6 | +0.1 |
| **IE.Indic** | ur | -1,-1,1 | 18.3 | 35.2 | 34.0 | +15.7 | 35.0 | 33.7 | +15.4 |
| **IE.Indic** | hi | -1,-1,1 | 34.3 | 52.4 | 53.4 | +19.1 | 51.3 | 49.1 | +14.8 |
| **Dravidian** | ta | -1,-1,1 | 36.1 | 42.8 | 43.4 | +7.3 | 43.1 | 43.0 | +6.9 |
| **Turkic** | tr | -1,-1,1 | 31.2 | 35.2 | 37.1 | +5.9 | 35.1 | 36.3 | +5.1 |
| **Afro-Asiatic** | ar | 1, 1, -1 | 38.5 | 47.3 | 45.3 | +6.8 | 45.8 | 43.7 | +5.2 |
| **Afro-Asiatic** | he | 1, 1, -1 | 55.7 | 58.8 | 57.6 | +1.9 | 58.3 | 57.6 | +1.9 |
| **Austronesian** | id | 1, 1, -1 | 49.3 | 53.1 | 52.3 | +3.0 | 52.3 | 51.9 | +2.6 |
| **Korean** | ko | -1,-1,1 | 34.0 | 37.1 | 37.2 | +3.2 | 36.3 | 36.4 | +2.4 |
| **IE.Celtic** | cy | 1, 1, -1 | 47.3 | 54.2 | 51.7 | +4.4 | 53.8 | 50.0 | +2.7 |
| **IE.Romance** | ca | 1, 1, -1 | 73.9 | 74.9 | 73.8 | -0.1 | 74.9 | 74.7 | +0.8 |
| **IE.Romance** | fr | 1, 1, -1 | 77.8 | 79.1 | 78.7 | +0.9 | 79.0 | 79.0 | +1.2 |
| **Uralic** | et | 1, -1,1 | 65.3 | 65.5 | 65.8 | +0.5 | 65.7 | 66.0 | +0.7 |
| **Uralic** | fi | 1, -1,1 | 66.7 | 67.1 | 67.0 | +0.3 | 66.9 | 67.1 | +0.4 |
| **IE.Slavic** | hr | 1, 1, 1 | 62.2 | 63.7 | 63.2 | +1.0 | 63.6 | 63.4 | +1.2 |
| **IE.Slavic** | bg | 1, 1, 1 | 79.6 | 79.7 | 79.2 | +0.0 | 79.7 | 79.7 | +0.1 |
| **IE.Baltic** | lv | 1, 1, 1 | 70.3 | 70.7 | 69.5 | -0.8 | 70.5 | 69.9 | -0.4 |
| **IE.Latin** | la | ?, ?, ? | 47.4 | 48.0 | 45.6 | -1.8 | 48.1 | 47.3 | -0.1 |
| **IE.Germanic** | da | 1, 1, 1 | 76.6 | 76.6 | 76.5 | -0.1 | 76.6 | 76.6 | +0.0 |
| **IE.Germanic** | nl | 0, 1, 1 | 67.5 | 67.6 | 67.5 | +0.0 | 67.9 | 67.9 | +0.4 |
| **Average Performance** | | | 54.3 | 58.4 | 57.8 | +3.5 | 58.1 | 57.5 | +3.1 |

Table 1: Cross-lingual transfer performances for dependency parsing on 19 languages from 13 different families, with the performance on the source language (English) as a reference. Performances are reported per UAS (we observe similar trends for LAS and details can be found in the appendix Table 7). We compare the baseline model (Ahmad et al., 2019) with our two algorithms (Lagrangian relaxation and posterior regularization) considering the oracle constraints, and the corpus-statistics constraints compiled from WALS. Columns ΔWALS denote the improvements bring by leveraging WALS feature as constraints. We also create a Features column to show three WALS features [83A,85A,87A] for each language. The values {1, -1, 0, ?} stand for the same as English, opposite to English, no dominant order, and feature missing, respectively.

ratio for unary constraint $C1$, we use the WALS features $82A, 83A, 85A, 86A, 87A, 88A, 89A$ that are related to $NOUN$ to form feature vectors, and do regression on languages in the test set except the target language to predict the constraint ratio. The process guarantees the target language remain unseen during the ratio estimation process. The ratios on the regression training languages are estimated by sampling 100 sentences in the training set per language.

We also consider an oracle setting where we collect a "ground-truth" ratio of each constraint for the target language to estimate an upper bound of our inference algorithms. In the oracle setting, we estimate the ratio on the whole training corpus of the target language and set the margin to $\theta = 0.01$.

### 4.2 Parsing Performances

We first compare the performances of the cross-lingual dependency parser with or without constraints. Table 1 illustrates the results for the 19 target languages we selected,[6] along with the performance on the source language (English). The performance on English is not as high as the dependency parsers specialized for English, because to achieve transfer, we have to freeze the pre-trained multi-lingual word embeddings. Yet this parser achieved the best single-source transfer performances according to Ahmad et al. (2019).

As is shown in Table 1, the improvements by our constrained inference algorithms are dramatic in a few languages that have very distinct word order features from the source language. For example, the parsing performance of Hindi (hi) improves about 15% in UAS with WALS features via both Lagrangian relaxation and posterior regularization inference. The improvements are less

[6] We also run on all languages in Ahmad et al. (2019) for completeness and observe similar trends. The results can be found in Appendix Table 7.

obvious for languages that are in the same family as English such as Danish(da) and Dutch(nl). This is expected as the corpus linguistic statistics of these languages are similar to English thus the constraints are mostly satisfied with the baseline parser. Comparing Lagrangian relaxation and posterior regularization, we find posterior regularization being more robust and less sensitive to the errors in the corpus-statistics estimation, while Lagrangian relaxation gives a higher improvement on average. Overall, the two proposed constrained inference algorithms improved the transfer performance by 3.5% and 3.1% per UAS on average on 19 target languages.

For languages like Finnish (fi) and Estonian (et), the WALS setting works even better than the oracle. We suspect the reason being the large margin we set in the WALS setting. When the estimated corpus-statistics is different from the real ratio in the test set, the large margin relaxes the constraints, thus could result in better performances.

**Discussion.** Despite the major experiments and analysis are conducted using English as the only source language, our approach is general and does not have restriction on the choice of the source language(s). To verify this claim, we run experiments with Hebrew as the source language. Under the oracle setting, Lagrangian relaxation and posterior regularization improve the baseline by 4.4% and 4.1%, respectively.

We observed that if we compile WALS features into hard constraints (i.e., set $r = 0$ or 1), the constraint inference framework only improves performance on half of the languages. For example, in Estonian (et), the performance *drops* about 3%. This is because WALS only provides the dominant order. Therefore, treating WALS as hard constraints introduces error to the inference.

Finally, we assume if we can access to native speakers, the corpus-statistics can be estimated by a few partial annotations of parse trees. In our simulation, using less than 300 arcs, we can achieve the same performance as using the oracle.

### 4.3 Contributions of Individual Constraints

We analyze the contribution of each constraint demonstrated in Table 2. Here we use the oracle setting to reduce the noise introduced by corpus-statistics estimation errors. The results are based on Lagrangian relaxation inference. As shown in Table 2, Despite some languages have non-projective dependencies, we observed performance improvements on almost all the languages when the projective constraint is enforced. All the constraints we formulated have positive contributions to the performance improvements. $C1 = (NOUN)$ brings the largest gain probably because its widest coverage.

| Model | UAS | coverage | Δ |
|---|---|---|---|
| baseline | 54.3 | N/A | N/A |
| +Proj. | 54.6 | N/A | +0.3 |
| +Proj.+C1 | 57.0 | 0.24 | +2.4 |
| +Proj.+C2 | 55.7 | 0.08 | +1.1 |
| +Proj.+C3 | 55.0 | 0.07 | +0.4 |
| oracle | 58.4 | N/A | +4.1 |

Table 2: Ablation study: average UAS of baseline model with different sets of constraints. Proj. represent projective constraints. C1-C3 and oracle are introduced in 4.1. The improvements for projective constraint and oracle are compared to baseline. For the other three constraint sets the improvement is compared to model with projective constraint.

| Const. | statistics | improvement |
|---|---|---|
| +Proj. | N/A | +0.1 |
| C1 | 0.30/0.36/0.94 | +6.9 |
| C2 | 0.00/0.06/1.00 | +11.3 |
| C3 | 0.14/0.27/0.12 | +0.5 |
| All | N/A | +18.1 |

Table 3: Contribution of individual constraints and their statistics in Hindi. The second column lists the ratios estimated from oracle in English/ baseline in Hindi/ oracle in Hindi, respectively. The improvement is measured in UAS. The improvement of constraints is computed same as Table 2

Table 1 shows that the performance of Hindi improves from 34% to over 51% per UAS for both inference algorithms. To better understand where the improvements come from, we conduct an analysis to breakdown the contribution of each individual constraint for Hindi. Table 3 shows the results. We can see that since the corpus linguistic statistics between Hindi and English are distinct, the baseline model only achieves low performance. With the constrained inference, especially the postposition constraint (C2), the proposed inference algorithm bring significant improvement.

To verify the effectiveness of the constraints, we analyze the relation between the performance improvements and corpus statistics ratio gaps between the source and the target languages. To quantify the ratio gap, we weight constraints by

their coverage rate and compute the weighted average of the ratio difference between source and target languages. Results show that the performance improvement is highly related to the ratio gap. The Pearson Correlation Coefficient is $\mathbf{0.938}$. The figure showing the correlation between performance gap (as per UAS) and the corpus statistics ratio gap is in the Appendix Figure 3.

## 5 Related Work

**Cross-Lingual Transfer for Parsing** Many approaches have been developed to transfer a dependency parser. However, they mainly focus on better capture information from the source language(s). McDonald et al. (2011); Guo et al. (2016); Täckström et al. (2013); Chen et al. (2019) consider transferring a parser trained on multiple source languages. Agić (2017); Lin et al. (2019) selects good source languages by comparing part-of-speech tags sequences. Søgaard (2011); Täckström et al. (2013) chooses suitable data points from the source language. Pires et al. (2019) uses multilingual BERT to leverage language features from multiple languages. Ahmad et al. (2019) design an order-free model to take out the order features from the source language. Xiao and Guo (2014); Guo et al. (2015) learn an alignment from source words to target words. Ponti et al. (2018) learn an anisomorphism from the source parsing tree to target. Rasooli and Collins (2019) reorder the source data before training. In contrast, we focus on incorporating linguistic properties in the target languages.

**Constrained Inference for Parsing** Several previous studies show that adding constraints in inference time improves the performance of models. Grave and Elhadad (2015) consider incorporating constraints to promote popular types of arcs in an unsupervised setting. Naseem et al. (2010); Li et al. (2019) train a parser with constraints compiled from the frequency of particular arcs. Compared with the previous work, we focus on cross-lingual transfer with word order constraints.

Finally, prior studies have noticed that the word order information is significant for parsing and use it as features (Ammar et al., 2016; Naseem et al., 2012; Rasooli and Collins, 2017; Zhang and Barzilay, 2015; Dryer, 2007). Täckström et al. (2013) further propose to decompose these features from models for adapting target languages. Wang and Eisner (2018a) use the statistics of surface part-of-speech (POS) tags of target languages to learn the word order. Wang and Eisner (2018b) use POS tags of target languages together with a similar language, and design a stochastic permutation process to synthetic the word order. However, none of them consider using the word order features as constraints.

**Incorporating Constraints In NLP Tasks** Constraints are widely incorporated in variety of NLP tasks. To name a few, Roth and Yih (2004) propose to formulate constrained inferences in NLP as integer linear programming problems. To solve the intractable structure, Rush and Collins (2012) decompose the structure and incorporate constraints on some composite tasks. To improve the performance of a model, Chang and Collins (2011); Peng et al. (2015) incorporate constraints on exact decoding tasks and inference tasks on graphical models, and Chang et al. (2013); Dalvi (2015); Martins (2015) incorporate corpus-level constraints on semi-supervised multilabel classification and coreference resolution. Zhao et al. (2017) incorporate corpus-level constraints to avoid amplifying gender bias on visual semantic role labeling and multilabel classification. In contrast to previous work, we incorporate corpus-level constraints to facilitate dependency parser in the cross-lingual transfer setting.

## 6 Conclusion

We propose to leverage corpus-linguistic statistics to guide the inference of cross-lingual dependency parsing. We compile these statistics into corpus-statistic constraints and design two inference algorithms on top of a graph-based parser based on Lagrangian relaxation and posterior regularization. Experiments on 19 languages show that our approach improves the performance of the cross-lingual parser substantially. In the future, we plan to study the design and incorporation of fine-grained constraints considering multipule languages for cross-lingual transfer. We also plan to adapt this constrained inference framework to other cross-lingual structured prediction problems, such as semantic role labeling.

**Acknowledgement** This work was supported in part by National Science Foundation Grant IIS-1760523 and an NIH R01 (LM012592). We thank anonymous reviewers and members of the UCLA-NLP lab for their feedback.

## References


Željko Agić. 2017. Cross-lingual parser selection for low-resource languages. In *Proceedings of the NoDaLiDa Workshop on Universal Dependencies, UDW@NoDaLiDa 2017, Gothenburg, Sweden, May 22, 2017*, pages 1–10. Association for Computational Linguistics.

Željko Agić, Jörg Tiedemann, Kaja Dobrovoljc, Simon Krek, Danijela Merkler, and Sara Može. 2014. Cross-lingual dependency parsing of related languages with rich morphosyntactic tagsets. In *EMNLP 2014 Workshop on Language Technology for Closely Related Languages and Language Variants*.

Wasi Uddin Ahmad, Zhisong Zhang, Xuezhe Ma, Eduard Hovy, Kai-Wei Chang, and Nanyun Peng. 2019. On difficulties of cross-lingual transfer with order differences: A case study on dependency parsing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*.

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A Smith. 2016. Many languages, one parser. *Transactions of the Association for Computational Linguistics*, 4:431–444.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Kai-Wei Chang, S. Sundararajan, and S. Sathiya Keerthi. 2013. Tractable semi-supervised learning of complex structured prediction models. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part III*, volume 8190 of *Lecture Notes in Computer Science*, pages 176–191. Springer.

Yin-Wen Chang and Michael Collins. 2011. Exact decoding of phrase-based translation models through lagrangian relaxation. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 26–37. ACL.

Xilun Chen, Ahmed Hassan Awadallah, Hany Hassan, Wei Wang, and Claire Cardie. 2019. Multi-source cross-lingual model transfer: Learning what to share. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 3098–3112. Association for Computational Linguistics.

Y. J. Chu and T. H. Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. *Internation Conference on Learning Representations*.

Bhavana Bharat Dalvi. 2015. *Constrained Semi-supervised Learning in the Presence of Unanticipated Classes*. Ph.D. thesis, Google Research.

Timothy Dozat and Christopher D Manning. 2017. Deep biaffine attention for neural dependency parsing. *Internation Conference on Learning Representations*.

Matthew S Dryer. 2007. Word order. *Language typology and syntactic description*, 1:61–131.

Matthew S. Dryer and Martin Haspelmath, editors. 2013. *WALS Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Kuzman Ganchev, Jennifer Gillenwater, Ben Taskar, et al. 2010. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11(Jul):2001–2049.

Edouard Grave and Noémie Elhadad. 2015. A convex and feature-rich discriminative approach to dependency grammar induction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1375–1384.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1234–1244.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2016. A representation learning framework for multi-source transfer parsing. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 2734–2740.

Shafiq Joty, Preslav Nakov, Lluís Màrquez, and Israa Jaradat. 2017. Cross-language learning with adversarial neural networks. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 226–237.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, and Noah A Smith. 2016. Distilling an ensemble of greedy dependency parsers into one mst parser. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1744–1753.

Bowen Li, Jianpeng Cheng, Yang Liu, and Frank Keller. 2019. Dependency grammar induction with a neural variational transition-based parser. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6658–6665.

Yu-Hsiang Lin, Chian-Yu Chen, Jean Lee, Zirui Li, Yuyan Zhang, Mengzhou Xia, Shruti Rijhwani, Junxian He, Zhisong Zhang, Xuezhe Ma, Antonios Anastasopoulos, Patrick Littell, and Graham Neubig. 2019. Choosing transfer languages for cross-lingual learning. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 3125–3135. Association for Computational Linguistics.

André F. T. Martins. 2015. Transferring coreference resolvers with posterior regularization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1427–1437. The Association for Computer Linguistics.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of ACL-2005*, pages 91–98, Ann Arbor, Michigan, USA.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, et al. 2013. Universal dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 92–97.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the conference on empirical methods in natural language processing*, pages 62–72. Association for Computational Linguistics.

Tahira Naseem, Regina Barzilay, and Amir Globerson. 2012. Selective sharing for multilingual dependency parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 629–637. Association for Computational Linguistics.

Tahira Naseem, Harr Chen, Regina Barzilay, and Mark Johnson. 2010. Using universal linguistic knowledge to guide grammar induction. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1234–1244. Association for Computational Linguistics.

Joakim Nivre, Mitchell Abrams, Željko Agić, and et al. 2018. Universal dependencies 2.2. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Robert Östling. 2015. Word order typology through multilingual word alignment. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 205–211.

Nanyun Peng, Ryan Cotterell, and Jason Eisner. 2015. Dual decomposition inference for graphical models over strings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 917–927. The Association for Computational Linguistics.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual bert? In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4996–5001. Association for Computational Linguistics.

Edoardo Maria Ponti, Roi Reichart, Anna Korhonen, and Ivan Vulić. 2018. Isomorphic transfer of syntactic structures in cross-lingual NLP. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 1531–1542. Association for Computational Linguistics.

Mohammad Sadegh Rasooli and Michael Collins. 2017. Cross-lingual syntactic transfer with limited resources. *Transactions of the Association for Computational Linguistics*, 5:279–293.

Mohammad Sadegh Rasooli and Michael Collins. 2019. Low-resource syntactic transfer with unsupervised source reordering. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 3845–3856. Association for Computational Linguistics.

Dan Roth and Wen-tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. Technical report, ILLINOIS UNIV AT URBANA-CHAMPAIGN DEPT OF COMPUTER SCIENCE.

Alexander M Rush and Michael Collins. 2011. Exact decoding of syntactic translation models through Lagrangian relaxation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 72–82.

Alexander M Rush and MJ Collins. 2012. A tutorial on dual decomposition and lagrangian relaxation for inference in natural language processing. *Journal of Artificial Intelligence Research*, 45:305–362.

Samuel L Smith, David HP Turban, Steven Hamblin, and Nils Y Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. *Internation Conference on Learning Representations*.

Anders Søgaard. 2011. Data point selection for cross-language adaptation of dependency parsers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 682–686. Association for Computational Linguistics.

Oscar Täckström, Ryan McDonald, and Joakim Nivre. 2013. Target language adaptation of discriminative transfer parsers. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1061–1071. Association for Computational Linguistics.

Jörg Tiedemann. 2015. Cross-lingual dependency parsing with universal dependencies and predicted pos labels. In *Proceedings of the Third International Conference on Dependency Linguistics (Depling 2015)*, pages 340–349.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Dingquan Wang and Jason Eisner. 2018a. Surface statistics of an unknown language indicate how to parse it. *Transactions of the Association for Computational Linguistics (TACL)*.

Dingquan Wang and Jason Eisner. 2018b. Synthetic data made to order: The case of parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1325–1337.

Wenhui Wang and Baobao Chang. 2016. Graph-based dependency parsing with bidirectional lstm. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2306–2315.

Min Xiao and Yuhong Guo. 2014. Distributed word representation learning for cross-lingual dependency parsing. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 119–129.

Jiateng Xie, Zhilin Yang, Graham Neubig, Noah A. Smith, and Jaime Carbonell. 2018. Neural cross-lingual named entity recognition with minimal resources. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 369–379. Association for Computational Linguistics.

Daniel Zeman and Philip Resnik. 2008. Cross-language parser adaptation between related languages. In *Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages*.

Yuan Zhang and Regina Barzilay. 2015. Hierarchical low-rank tensors for multilingual transfer parsing. Association for Computational Linguistics.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2017. Men also like shopping: Reducing gender bias amplification using corpus-level constraints. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2979–2989. Association for Computational Linguistics.

# Look-up and Adapt: A One-shot Semantic Parser

**Zhichu Lu*** 
Carnegie Mellon University
Pittsburgh, PA
zhichul@cs.cmu.edu

**Forough Arabshahi***
Carnegie Mellon University
Pittsburgh, PA
farabsha@cs.cmu.edu

**Igor Labutov**
LAER AI, Inc.
New York, NY
igor.labutov@laer.ai

**Tom Mitchell**
Carnegie Mellon University
Pittsburgh, PA
tom.mitchell@cs.cmu.edu

## Abstract

Computing devices have recently become capable of interacting with their end users via natural language. However, they can only operate within a limited "supported" domain of discourse and fail drastically when faced with an out-of-domain utterance, mainly due to the limitations of their semantic parser. In this paper, we propose a semantic parser that generalizes to out-of-domain examples by learning a general strategy for parsing an *unseen* utterance through adapting the logical forms of *seen* utterances, instead of learning to generate a logical form from scratch. Our parser maintains a memory consisting of a representative subset of the seen utterances paired with their logical forms. Given an unseen utterance, our parser works by looking up a similar utterance from the memory and adapting its logical form until it fits the unseen utterance. Moreover, we present a data generation strategy for constructing utterance-logical form pairs from different domains. Our results show an improvement of up to 68.8% on one-shot parsing under two different evaluation settings compared to the baselines.

## 1 Introduction and Background

Speech recognition technologies are achieving human parity (Xiong et al., 2016). As a result, end users can now access different functionalities of their phones and computers through spoken instructions via a natural language processing interface referred to as a conversational agent. Current commercial conversational agents such as Siri, Alexa or Google Assistant come with a fixed set of simple functions like setting alarms and making reminders, but are often not able to cater to the specific phrasing of a user or the specific action a user needs. However, it has recently been shown that it is possible to add new functionalities to an agent through natural language *instruction* (Azaria et al., 2016; Labutov et al., 2018).

For example, assume that the user wants to add a functionality for resetting an alarm based on the weather forecast for the next day, as demonstrated by the following utterance: "whenever it snows at night, wake me up 30 minutes early". The user can instruct this task to the agent by breaking it down into a sequence of actions that the agent already knows.

- check the weather app,
- see if the forecast is calling for snow,
- if yes, then reset the time of the alarm to 30 minutes earlier.

This set of instructions result in a logical form or a semantic parse for this specific new utterance. However, this approach can be used in practice only if the agent is capable of generalizing from this single new utterance to similar utterances such as "if the weather is rainy, then set an alarm for 1 hour later". We refer to this problem as *one-shot semantic parsing*.

In this paper, we address this one-shot semantic parsing task and present a semantic parser that generalizes to out-of-domain utterances by seeing a single example from that domain. While state of the art neural semantic parsers are flexible to language variations, they need plenty of examples from the new domain to be able to parse an utterance from that domain, which is not possible in our scenario. On the other hand, grammar parsers are not robust to the flexibility of language because of their use of string matching. Therefore, we propose a method that preserves the robustness of neural semantic parsers while addressing

* These two authors contributed equally



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1129–1139,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

the data sparsity of the one-shot semantic parsing task. We present a general strategy for "adapting" logical forms rather than constructing them from scratch by "looking up" similar sentences that we know how to parse and changing their logical forms until they fit the new utterances. These logical forms are looked-up from a memory that contains a representative subset of previously seen utterance-logical form pairs. Once this general strategy is learned, the parser can be extended to parse an utterance from a new domain by adding one new example of that domain to the memory.

We propose a dataset generation method that allows us to evaluate the effectiveness of our model in a one-shot setting. We show that we generate reasonably good utterances while creating different experimental setups and scenarios for evaluation.

**Summary of Results** In this paper we propose a novel neural semantic parser for the task of one-shot semantic parsing. We design two different experiments to evaluate the parsing accuracy. We show that our approach improves the performance of neural semantic parsers by a significant margin across 6 different domains of discourse. Moreover, we present a detailed analysis of our proposed model and the performance of its different components through an oracle study.

## 2 Problem Definition

In this section, we introduce the basic terminology used in the paper and formally define the task of one-shot semantic parsing.

A semantic parser takes as input an *utterance* and outputs a corresponding *logical form*. An utterance is a sequence of words and a logical form is an s-expression capturing the meaning of the utterance.

For example, "parents of John's friends" is an utterance and `(field parent (field friend John))` is its logical form.

We use a synchronous context free grammar (SCFG), which is a generalization of the context-free grammar (CFG), to generate grammar rules (Chiang, 2006). A rule in an SCFG has a left hand side, which is a non-terminal category, and two right hand sides, referred to as the source and the target. For our semantic parsing task, the source corresponds to an utterance, and the target corresponds to a logical form. We denote a grammar rule with the small letter $g$. A set of grammar rules, denoted by the capital letter $G$, span a domain of utterances. In this paper, we define $domain(G)$ as the set of all *utterance-logical form* pairs generated by a set of grammar rules $G$. A sample SCFG $G_{sample}$ with its domain are provided in Tables 5 of Appendix A and Table 1 below, respectively.

| Utterance | Logical Form |
|---|---|
| John | `john` |
| Mary | `mary` |
| parents | `parent` |
| children | `child` |
| parents of John | `(field parent john)` |
| parents of Mary | `(field parent mary)` |
| children of John | `(field child john)` |
| children of Mary | `(field child mary)` |
| John 's parents | `(field parent john)` |
| Mary 's parents | `(field parent mary)` |
| John 's children | `(field child john)` |
| Mary 's children | `(field child mary)` |

Table 1: Domain of the sample SCFG $G_{sample}$. The rules of this SCFG are presented in the Appendix A. The domain of the sample SCFG is the set of all ⟨source, `target`⟩ pairs listed in this table.

In **one-shot semantic parsing** we are given as input a subset of utterance and logical form pairs from $domain(G)$ and a single utterance and logical form pair from $domain(G')$. The goal of one-shot semantic parsing is to parse utterances from $domain(G')$ that are not in the input.

Let us provide an example to make this more clear. Consider that we are given utterance and logical form pairs from $domain(G_{\text{sample}})$ in Table 1 as well as the following utterance and logical form pair as input.

⟨friends of John, `(field friend john)`⟩

The goal of one-shot semantic parsing is to parse examples such as

⟨friends of Mary, `(field friend Mary)`⟩ (1)

⟨parents of Mary's friends, `(field parent(field friend Mary)`⟩ (2)

which are not in $domain(G_{\text{sample}})$.

This is a challenging task for a complex grammar since the domain of a typical grammar consists of hundreds of thousand of examples. Therefore, being able to parse all variations of utterances in the domain by seeing only a single example from it does not have a trivial solution. Grammar-based semantic parsers usually

rely on string matching which limits their robustness to natural language variation. Neural semantic parsers are more robust compared to grammar-based parsers. However, their performance significantly drops in such a data-hungry setting. In the next section, we present our look-up adapt semantic parser that addresses the challenges of neural semantic parsers in a data-sparse scenario.

## 3 Look-up and Adapt

In this section, we propose a novel neural network architecture for one-shot semantic parsing. We are given a set of utterance-logical form pairs as input and our goal is to output a semantic parse for an unseen query utterance. Our model is able to generate a logical form for the utterance by "looking up" a similar utterance from a pool of utterance-logical form pairs and "adapting" its logical form. This pool of utterances is maintained in a "memory" and consist of a representative subset of the input utterance-logical form pairs. We will show that in the data-sparse scenario of one-shot semantic parsing, adapting known logical forms is easier compared to generating a logical form from scratch. We will also discuss that it is important what subset of the data is included in the memory.

Our model consists of two main modules, namely look-up § 3.1 and adapt § 3.2. Figure 1 shows a sketch of our proposed model and the main look-up adapt algorithm is given in Algorithm 1. The look-up module is responsible for retrieving utterance-logical form pairs from the memory, using two Bidirectional LSTM encoders. The adapt module is responsible for adapting the retrieved logical form until it results in the correct semantic parse. The adapt module has two sub-modules, namely the aligner and the discriminator. The aligner is responsible for aligning the logical form with the query utterance and the discriminator decides which parts of the logical form should be swapped with a new one from the memory. In the following sections, we will define each sub-module of our proposed model in detail and propose a loss for training the model in an end-to-end fashion.

Before describing the model components, let us start by introducing the notation we will use in the following sections. Bold capital letters represent matrices, and bold lower case letters represent vectors. $\boldsymbol{X}$ denotes a distributed representation of an utterance, where each column of $\boldsymbol{X}$ is the representation for a word in the utterance. $\boldsymbol{Y}$ denotes a distributed representation of a logical form, where each column of $\boldsymbol{Y}$ is the representation for a predicate in the logical form. `T` is a tree representation of the logical form. $\boldsymbol{w}$ and $\boldsymbol{w}'$ denote attention weights that sum to at least 0 and at most 1.

**Algorithm 1** LookUpAndAdapt

```
function LookUpAndAdapt(Memory, X', w')
    (X, Y, T) ← LookUp(Memory, X', w')
    for all t in T.children do
        t' ← Adapt(Memory, X', Y, t, w')
        T.children.replace(t, t')
    return T
```

### 3.1 Look-up

In this section, we describe our look-up strategy and discuss its sub-modules in detail.

**Encoder** Bidirectional RNNs have been successfully used to represent sentences in many areas of natural language processing such as question answering, neural machine translation, and semantic parsing (Bengio and LeCun, 2015; Hermann et al., 2015; Dong and Lapata, 2016). We use Bidirectional LSTMs where the forward LSTM reads the input in the utterance word order $(x_1, x_2, ..., x_T)$ and the backward LSTM reads the input in reversed order $(x_T, x_{T-1}, ..., x_1)$. The output for each word $x_i$ is the concatenation of the $i$th hidden state $h_i^{fwd}$ of the forward LSTM and the $T - i$th hidden state $h_{T-i}^{bck}$ of the backward LSTM. As shown in Figure 1, we use two Bidirectional LSTMs, one to encode the utterance, and one to encode the logical form. The utterance is treated as a sequence of words, where each word is represented as a pretrained GloVe (Pennington et al., 2014) embedding. The logical form is treated as a sequence of predicates and parentheses, each of which is also represented as a pretrained GloVe embedding. The motivation to use GloVe embeddings for both words and predicates is to avoid the problem of unknown words/predicates encountered in the one-shot utterance's domain.

**Retrieving from the memory** The memory consists of a subset of the utterance-logical form pairs given as input to the algorithm. We refer to the number of pairs in the memory as its size and denote it with $n$. We would like to note that the algorithm's success depends on having a memory that captures the structure of the domain. In this

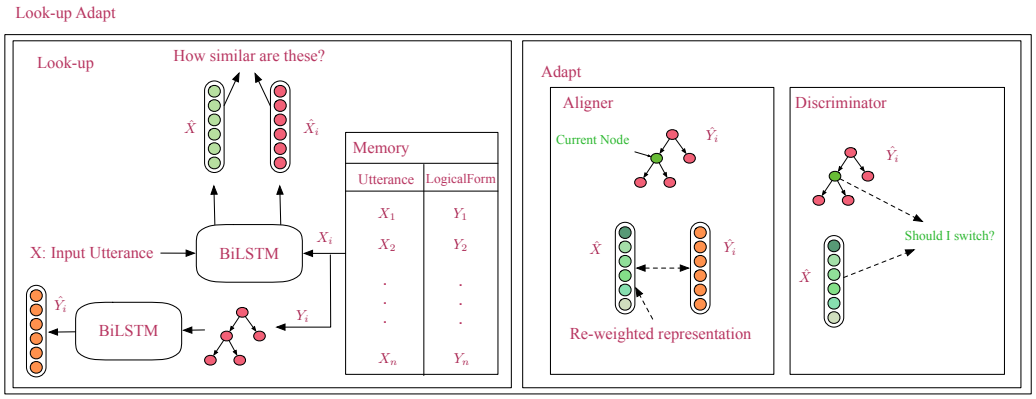


Figure 1: Look-up Adapt semantic parser. Our model consists of two main modules, namely "Look-up" (left box) and "Adapt" (right box). The look-up module is responsible for retrieving utterance-logical form pairs from the memory that are similar to an input query utterance $X$. The 'Adapt" module is further broken down into two modules "Aligner" and "Discriminator". Given a looked-up utterance and logical form pair, the aligner module traverses the logical form tree and updates the utterance representation accordingly while the discriminator module decides weather to switch the current node with another one from the memory or keep it as is.

paper, we make a simplifying assumption that the memory is given by an oracle, and it consists of one example per grammar rule. Other choices for the memory are also possible but exploring them falls out of the scope of this paper. An interested reader is referred to Appendix B for a more detailed discussion. In the future we wish to automate the acquisition of this memory.

Given an encoded query utterance $\boldsymbol{X}'$, we model the retrieval as a classification over examples in the memory. We compute a fixed size query vector $\boldsymbol{x}' = \boldsymbol{X}' \cdot \boldsymbol{w}'$ which is a weighted average of the column vectors in $\boldsymbol{X}'$. We compute a fixed-size key vector for every example in the memory $\boldsymbol{x} = \sum_{i=1}^{T} \boldsymbol{X}_i$, as the mean of the column vectors of $\boldsymbol{X}$.

Given a query vector $\boldsymbol{x}'$ and vectors $\boldsymbol{x}_i$ from the memory for $i = 1, 2, ..., n$ where $n$ is the memory size, we model the probability of retrieving the $i$th entry from memory using equation 3

$$P(i|\boldsymbol{x}') = \frac{\exp(e_i)}{\sum_{j=1}^{n} \exp(e_j)} \quad (3)$$

where

$$e_i = f([\boldsymbol{x}', \boldsymbol{x}_i, \boldsymbol{x}' \odot \boldsymbol{x}_i, |\boldsymbol{x}' - \boldsymbol{x}_i|, \boldsymbol{x}'^T W_1 \boldsymbol{x}_i]) \quad (4)$$

where $\odot$ is element-wise product,$[a, b]$ is concatenation of vectors $a$ and $b$, and $f$ is a Multi-Layer-Perceptron with a single hidden layer and ReLU activation function. This way of featurizing $\boldsymbol{x}'$ and $\boldsymbol{x}$ was adapted from (Kumar et al., 2016). LOOKUP greedily returns the example with the highest probability of being selected.

### 3.2 Adapt

The algorithm for ADAPT is given in Algorithm 2. The adaptation is two phase. First an alignment from `T` to relevant words in the new utterance $\boldsymbol{X}'$ is computed. Then a decision is made on whether this subtree fails to match the relevant words in the utterance and needs to be replaced. If yes, a recursive call to LOOKUPANDADAPT will be made with an updated attention $\boldsymbol{w}$ to focus on words relevant to the subtree, and the returned value will be propagated and eventually used as a replacement for the current subtree. Otherwise, recursive calls to ADAPT will be made to check the children of the current subtree. For an example see Table 9.

The input `Memory` to the algorithm is used in case a recursive call to LookUpAndAdapt is needed. $\boldsymbol{X}'$ is the representation of the new utterance. $\boldsymbol{Y}$ is the representation of the current logical form. `T` is a subtree of $\boldsymbol{Y}$ that we wish to adapt. $\boldsymbol{w}'$ is the attention weights of the parent of `T`, to be used as a mask when the alignment of `T` is computed. In the following space we describe the aligner module for alignment and the discriminator module for scoring a match between a subtree and part of the new utterance.

**Aligner Module** The aligner module produces an attention score over the new utterance given a subtree of a logical form. Inspired by the gating attention mechanism in (Kumar et al., 2016), we model attention for every word in the utterance as the maximum attention given by any predicate in the subtree `T`.

**Algorithm 2** Adapt

**function** ADAPT(`Memory`, $\boldsymbol{X}', \boldsymbol{Y}$, `T`, $\boldsymbol{w}'$)
  **Phase 1: aligning `T` to relevant words in $\boldsymbol{X}'$**
  $\boldsymbol{w} \leftarrow$ ALIGN($\boldsymbol{X}', \boldsymbol{Y}$, `T`, $\boldsymbol{w}'$)
  $\boldsymbol{x}' \leftarrow \boldsymbol{X}' \cdot \boldsymbol{w}$
  **if** $\sum_i \boldsymbol{w}_i > 1$ **then**
    $\boldsymbol{x}' \leftarrow \boldsymbol{x}' \div \sum_i \boldsymbol{w}_i$

  **Phase 2: deciding whether to replace `T` and then recurse**
  $P(fail\ to\ match) \leftarrow$ DISCRIMINATE($\boldsymbol{x}', \boldsymbol{Y}$, `T`)
  **if** $P(fail\ to\ match) > 0.5$ **then**
    **return** LOOKUPANDADAPT(`Memory`, $\boldsymbol{X}', \boldsymbol{w}$)
  **else**
    **for all** `t` **in** `T.children` **do**
      `t'` $\leftarrow$ ADAPT(`Memory`, $\boldsymbol{X}', \boldsymbol{Y}$, `t`, $\boldsymbol{w}$)
      `T.children.replace(t, t')`
    **return** T

$$\boldsymbol{w}_i = \max_{j \in span(\texttt{T})} \sigma(s_{ij}) \tag{5}$$

where $\sigma$ is the sigmoid function and

$$s_{ij} = g([\boldsymbol{X}_i, \boldsymbol{Y}_j, \boldsymbol{Y}_p, \boldsymbol{X}_i \odot \boldsymbol{Y}_j, \boldsymbol{X}_i \odot \boldsymbol{Y}_p, |\boldsymbol{X}_i - \boldsymbol{Y}_j|, |\boldsymbol{X}_i - \boldsymbol{Y}_p|, \boldsymbol{X}_i^T W_2 \boldsymbol{Y}_j, \boldsymbol{X}_i^T W_p \boldsymbol{Y}_p]) \tag{6}$$

where $\odot$ is element-wise product, $[a, b]$ is concatenation of vectors $a$ and $b$, $\boldsymbol{X}_i$ is the $i$th column of $\boldsymbol{X}$, $\boldsymbol{Y}_i$ is the $i$th column of $\boldsymbol{Y}$, and $g$ is a learnable linear transformation. $\boldsymbol{Y}_p$ is the representation of the parent predicate of the subtree `T` (e.g. the parent predicate is `field` for subtree `john` of logical form `(field parent john)`). We found that adding this feature helps the model learn to align arguments of predicates better.

We also found that constraining the attention further by requiring that the attention of a child node $\boldsymbol{w}$ be a refinement of the attention of a parent node $\boldsymbol{w}'$ facilitates good alignment. This is modeled by the update rule

$$\boldsymbol{w}_i \leftarrow \boldsymbol{w}_i \cdot \sqrt{\boldsymbol{w}'_i} \tag{7}$$

Observe that if the attention score of the parent node is very low for some word, the child is not going to be able to look at it. This encourages the parent node to attend to words not just relevant to itself but also the children, and encourages the child to refine rather than drastically change from the alignment of the parent node.

Given the output weights of the aligner module, we compute a fixed-size representation for the relevant words in the utterance to the current subtree `T` by first taking the weighted average $\boldsymbol{x}' = \boldsymbol{X}' \cdot \boldsymbol{w}$ where $\cdot$ is matrix product. If $\sum_i \boldsymbol{w}'_i > 1$, we will divide $\boldsymbol{x}'$ by $\sum_i \boldsymbol{w}'_i$ to normalize it. $\boldsymbol{x}'$ is then compared against the representation of the subtree by the discriminator to produce a confidence score indicating whether the subtree matches the words it aligns to in the utterance.

**Discriminator Module** The discriminator module learns to tell when a subtree fails to match the meaning of the words in its alignment. Given a subtree `T` of a logical form $\boldsymbol{Y}$, its fixed-size representation is computed as a mean of the representations for each predicate in the subtree using the formula $\boldsymbol{y} = \sum_{i \in span(\texttt{T})} \boldsymbol{Y}_i$, where $\boldsymbol{Y}_i$ is the $i$th column of $\boldsymbol{Y}$, and $span(\texttt{T})$ is the set of indices corresponding to all the predicates in the subtree `T`. For example, the fixed-size representation for the subtree `john` in logical form `(field parent john)` is just $\boldsymbol{Y}_4$ (indexing from 1, and also counting parentheses because they are also encoded in $\boldsymbol{Y}$).

We model the confidence that the subtree representation $\boldsymbol{y}$ fails to match the meaning of the aligned words $\boldsymbol{x}'$ as

$$P(fail\ to\ match | \boldsymbol{x}', \boldsymbol{y}) = \sigma(d) \tag{8}$$

where $\sigma$ is the sigmoid function and

$$d = h([\boldsymbol{x}', \boldsymbol{y}, \boldsymbol{x}' \odot \boldsymbol{y}, |\boldsymbol{x}' - \boldsymbol{y}|, \boldsymbol{x}'^T W_3 \boldsymbol{y}]) \tag{9}$$

$\odot$ is element-wise product, $[a, b]$ is concatenation of vectors $a$ and $b$, and $h$ is a learnable linear transformation. This confidence score is used by Algorithm 2 to decide whether to replace the subtree or to keep it.

## 4 Training

In training, we maximize the log conditional likelihood of the data. Due to our selection of memory (one example for each grammar rule in $G$), for a given $x$ there is a unique sequence of retrieval and adaptation actions that lead to the production of the correct $y$. Specifically, letting $a_i$ be the $i$th action in the correct sequence of $l$ actions, we define the probability of producing $y$ given $x$ as the probability of taking the sequence of actions $a_1, ..., a_l$. We decompose this joint probability into products of conditional probabilities of taking action $i$ given the previous actions and the input $x$

$$P(y|x) = \prod_{i=1}^{l} P(a_i | a_1, a_2, ..., a_{i-1}, x) \tag{10}$$

If $a_i$ is a retrieval action,

$$P(a_i|a_1, a_2, ..., a_{i-1}, x) = P(i|\boldsymbol{x}') \quad (11)$$

where $P(i|\boldsymbol{x}')$ is defined in Equation 3. If $a_i$ is an adaptation decision, i.e. to replace a particular subtree

$$P(a_i|a_1, a_2, ..., a_{i-1}, x) = P(fail\,to\,match|\boldsymbol{x}', \boldsymbol{y}) \quad (12)$$

where $P(fail\,to\,match|\boldsymbol{x}', \boldsymbol{y})$ is defined in Equation 8.

## 5 Dataset Generation

Using existing semantic parsing datasets to directly evaluate one-shot parsing is difficult, because evaluation of one-shot parsing requires grouping of utterances by structural similarity to construct domains.

Therefore, we generate our own data [1] based on a synchronous context free grammar. We evaluate our approach on the generated dataset. Although our dataset is synthetic, it has the properties needed to evaluate one-shot parsing:

- clear distinction of domains
- plenty of examples for rules in the old domain
- one example for each rule in the new domain
- a separate evaluation set containing variations of the new rules

It is worth noting that although datasets such as OVERNIGHT (Wang et al., 2015) do have clear domain distinction, they do not have the other properties needed for a good evaluation of one-shot parsing. In the future, we are planning to use crowd-sourcing to rephrase the generated utterances in order to make them more natural.

We have two topics of discourse in the SCFG that we use to generate our data. The first is accessing some field of some object, and the second is setting some field of some object to some value or some field of another object. These are general purpose instructions that can express many common intents, such as looking up the location of a restaurant, getting the phone number of a contact, and setting reminders and alarms.

We have six domains of discourse in the SCFG that we use to generate our data – *person*, *restaurant*, *event*, *course*, *animal*, and *vehicle*. Table 2 includes sample sentences and their logical form for every domain of discourse.

Since one-shot parsing has two phases, our dataset is slightly different from a typical dataset consisting of a single collection of *utterance-logical form* pairs. In the following space we describe the sets of data generated for each domain. In the next section we describe how we can use this dataset to design two different one-shot parsing evaluation setups.

For each domain $d$, we define a $G_d$ and randomly generate examples from $domain(G_d)$ to construct an "old" set of examples $D_{old,d}$. We also generate a representative subset $M_{old,d}$ to be used as the memory of our model. As described in § 3.1, $M_{old,d}$ contains one example for each grammar rule in $G_d$.

In addition, we define a set of new rules $G'_d$ disjoint from $G_d$ for each domain and generate one example $e_{i,d}$ for each new rule $g'_{i,d} \in G'_d$, and store them into the memory $M'_{new,d}$. These are the one-shot examples. The extended memory $M_{new,d} = M_{old,d} \cup M'_{new,d}$.

Finally, we generate evaluation sets $E_{old,d} \subset domain(G_d)$ and $E_{new,d} \subset domain(G_d \cup G'_d) - domain(G_d)$ where $-$ denotes set difference. In our experiments we split the evaluation sets randomly into a development set and a test set of the same size.

Some statistics of our generated dataset is presented in Appendix D. In the next section, we describe how we use this data to generate two different experimental scenarios for evaluating one-shot semantic parsing.

## 6 Results and Evaluation

We evaluate our approach in six domains, namely person, restaurant, event, course, animal, and vehicle, and in two one-shot parsing scenarios, namely *extension* and *transfer*. In the next section we define these two different scenarios. We compare the performance to a sequence-to-sequence parser with attention which is defined in the following paragraphs.

**Baseline** Our baseline is a sequence-to-sequence parser with attention. We use a 1-layer Bidirectional LSTM as the encoder for the utterance, and a 1-layer LSTM as the decoder for the logical form. We initialize the decoder with a learned projection of the last hidden state of the encoder. The inputs to the encoder are GloVe

[1] Our dataset can be accessed at https://github.com/zhichul/lookup-and-adapt-parser-data.

| | Domain | Utterance & Logical Form |
|---|---|---|
| $a_1$ | *person* | the hometown field of john |
| | | (field (relation hometown) (person john)) |
| $a_2$ | *person* | set her 's parents with classmates |
| | | (set (field (relation parent) (person reference)) (person classmate)) |
| $b_1$ | *restaurant* | irish restaurant instances 's price field |
| | | (field (relation price) (restaurant irish)) |
| $b_2$ | *restaurant* | set address field of all irish restaurant as indian restaurant 's price |
| | | (set (field (relation address) (restaurant irish)) (field (relation price) (restaurant indian))) |
| $c_1$ | *event* | the start time of lectures |
| | | (field (relation start) (event lecture)) |
| $c_2$ | *event* | set the attendants of that event to organizers field of receptions instances |
| | | (set (field (relation attendant) (event reference)) (field (relation organizer) (event reception))) |
| $d_1$ | *course* | size of my history course instances |
| | | (field (relation size) (course history)) |
| $d_2$ | *course* | set all history course instances 's prerequisite field as physics course |
| | | (set (field (relation prerequisite) (course history)) (course physics)) |
| $e_1$ | *animal* | life span of all lion instances |
| | | (field (relation span) (animal lion)) |
| $e_2$ | *animal* | set fish instances 's family field with dog |
| | | (set (field (relation family) (animal fish)) (animal dog)) |
| $f_1$ | *vehicle* | the source field of all buses |
| | | (field (relation source) (vehicle bus)) |
| $f_2$ | *vehicle* | set operators of all subways as buses instances |
| | | (set (field (relation operator) (vehicle subway)) (vehicle bus)) |

Table 2: Sample utterance-logical form pairs from each domain of discourse. Sentences $a_1$ ... $f_1$ parse to commands that retrieve a field of an object. Sentences $a_2$ ... $f_2$ parse to commands which set a field of an object to a constant or a field of another object. The grammar we use do not constrain the type of the set command. As a consequence, some sentences generated are not semantically correct according to a human interpreter but still they preserve the sentence structure. Sentence $b_2$, $e_2$, and $f_2$ are examples of this phenomena.

word embeddings, and the inputs to the decoder are concatenations of embeddings of logical predicates and context vectors. Context vectors are attention weighted averages of projected encoder states. The attention weights over the utterance for each decoding step $t$ is computed by taking the dot product of the hidden state of the decoder at step $t$ and the projected encoder state for each word in the utterance, normalized using the softmax function. The output of a decoding step $t$ is a probability distribution over all the logical predicates. The distribution is modeled as a dot product between a projection of the overall decoding state at time $t$, and the embedding for each logical predicate, normalized using the softmax function. The overall decoding state at time $t$ is the concatenation of the hidden state of the decoder at time $t$ and the context vector at time $t$. Decoding of a logical form given an utterance is done as a greedy search over all possible logical forms to output.

**Evaluation Metric** Our evaluation metric is parsing accuracy. We define parsing accuracy as the ratio of the correctly parsed utterances in the test set. An utterance is parsed correctly if its generated logical form matches the annotated logical form exactly. E.g. `(field (relation parent) (person john))` is a correct parse of "parents of John". We report accuracy percentage in Tables 3 and 4.

### 6.1 Discussion

We design two different scenarios for one-shot parsing evaluation, namely the *extension* and the *transfer* scenarios. We define and discuss the results of each scenario in the following sections.

**Extension** This scenario tests one-shot parsing of new rules within the same domain that the model is trained on. In this case, we hold out several rules and their corresponding utterance-logical form pairs for evaluation and train on the remaining ones.

The results of this experiment are presented in Table 3. Each column indicates the evaluation results on each domain. The *full* scores refer to the overall percentage of correct parses on the entire test data, the *d=2* scores refer to the percentage of correct parses on examples with logical form

depth 2 and the *d=3* scores refer to the percentage of correct parses on examples with logical form depth 3.

As it can be seen in the table, our model is able to consistently improve the sequence-to-sequence baseline on all the domains by a significant margin of 68.8%.

**Transfer** This scenario tests one-shot parsing of new rules in a domain different from the one that the model is trained on. For example, in the *transfer* scenario for domain *person* the model is trained on examples from the other domains *restaurant*, *event*, *course*, *animal*, and *vehicle* and tested on samples from domain *person*. This is harder compared to the extension setup since it requires generalization to a completely new domain.

The results of this section are reported in Table 4. As it can be seen, the performance is improved by up to 36.6% compared to the baseline model.

In order to have a better understanding of the model and indicate why it is performing comparable to the baseline on some of the domains setting, we carry out a model analysis in the next section.

**Model Analysis** We add two variations of our model ORACLE-DISCRIM and PRETRAIN-ENC for the transfer scenario to see the performance improvements gained from replacing different components of our model with an oracle/near oracle. This identifies potential bottlenecks of the model.

In ORACLE-DISCRIM, we load a trained model from LOOKUPADAPT but replace the discriminator with an oracle during evaluation. As the numbers in row 3 of Table 4 show, using an oracle discriminator improves the model performance by more than 10% in the *vehicle* domain and *course* domain, and by smaller amounts in the other domains. On one hand, this suggests that the discriminator has room for improvement. On the other hand, the numbers suggest that there is still a much larger room for improvement for the encoder, aligner, and look-up components. Our hypothesis is that the encoder components are likely the main bottleneck because during evaluation in the *transfer* scenario they have to produce good representations for utterances and logical forms very different from the ones which they have seen during training. We found evidence supporting this hypothesis in the PRETRAIN-ENC variation.

In PRETRAIN-ENC, we pretrain our model on all the domains, which ensures that the two encoders see all domains. We then use the encoder parameters of this all-domain pretrained model to initialize the encoders in the experiments for the *transfer* scenario. We then fix the encoders and train only the aligner, discriminator, and look-up parameters. This results in near perfect generalization to the test domain by the aligner, discriminator, and look-up components, which are trained only on the other domains. This suggests that the encoder is the main bottleneck of our model since using a near-oracle version boosts its performance to perfect. The results for this variation is shown in row 4 of Table 4.

## 7 Related Work

Semantic parsing is the task of mapping utterances to a formal representation of their meaning. Researchers have used grammar-based methods as well as machine learning-based methods to address this problem. Grammar-based parsers work by having a set of grammar rules that are either learned or hand-written, an algorithm for generating a set of candidate logical forms by recursive application of the grammar rules, and a criterion for picking the best candidate logical form within that set (Liang and Potts, 2015; Zettlemoyer and Collins, 2005, 2007). However, they are brittle to the flexibility of language. To improve this limitation, supervised sequence-based neural semantic parsers have been proposed (Dong and Lapata, 2016). Herzig and Berant (2017) improved the performance of neural semantic parsers by training over multiple knowledge bases and providing the domain encoding at decoding time. In addition to supervised learning, reinforcement learning methods for neural semantic parsing have been explored in (Zhong et al., 2017).

Retrieve-and-edit style semantic parsing is gaining popularity. Hashimoto et al. (2018) proposed a retrieve and edit framework that can efficiently learn to embed utterances in a task-dependent way for easy editing. Our work differs in that we perform hierarchical retrievals and edits, and that we evaluate on cross-domain data and focus on one-shot semantic parsing. It is worth noting that retrieve-and-edit as a general framework is not limited to semantic parsing and is applicable to other areas such as sentence generation. (Guu et al., 2018) and machine translation

| | | extension | | | | | |
|---|---|---|---|---|---|---|---|
| | | person | restaurant | event | course | animal | vehicle |
| Sequence-To-Sequence | full | 20.1 | 25.1 | 26.1 | 19.9 | 25.0 | 18.8 |
| | d=2 | 25.0 | 18.1 | 20.0 | 31.8 | 26.5 | 25.9 |
| | d=3 | 13.9 | 37.3 | 42.4 | 0.0 | 21.5 | 5.8 |
| LOOKUPADAPT | full | **45.9** | **61.7** | **85.5** | **79.2** | **63.6** | **87.6** |
| | d=2 | 56.7 | 85.4 | 94.5 | 86.7 | 70.1 | 93.6 |
| | d=3 | 27.8 | 20.0 | 61.7 | 66.7 | 46.5 | 76.5 |

Table 3: Test Accuracy on the **extension** Dataset.

| | | transfer | | | | | |
|---|---|---|---|---|---|---|---|
| | | person | restaurant | event | course | animal | vehicle |
| Sequence-To-Sequence | full | 5.2 | 29.2 | 5.2 | **19.9** | **35.5** | **12.5** |
| | d=2 | 6.4 | 41.7 | 4.9 | 27.5 | 44.9 | 15.6 |
| | d=3 | 3.1 | 8.4 | 5.7 | 6.0 | 13.8 | 6.3 |
| LOOKUPADAPT | full | **41.8** | **43.8** | **8.4** | 16.7 | 28.3 | 11.5 |
| | d=2 | 52.5 | 53.4 | 13.2 | 24.2 | 37.3 | 14.1 |
| | d=3 | 21.4 | 27.8 | 0.0 | 3.0 | 6.9 | 6.3 |
| ORACLE-DISCRIM | full | 46.9 | 45.8 | 12.5 | 29.2 | 33.3 | 25.0 |
| | d=2 | 55.6 | 56.7 | 18.0 | 41.8 | 44.8 | 29.7 |
| | d=3 | 30.3 | 27.8 | 2.9 | 5.9 | 6.9 | 15.6 |
| PRETRAIN-ENC | full | 98.0 | 100.0 | 100.0 | 99.0 | 100.0 | 96.9 |
| | d=2 | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 |
| | d=3 | 94.0 | 100.0 | 100.0 | 97.1 | 100.0 | 90.1 |

Table 4: Test Accuracy on the **transfer** Dataset.

Gu et al. (2018).

Another line of research maps semantic parsing under cross-domain setting to a domain adaptation problem (Su and Yan, 2017). In their work, the model is trained on a certain domain and then fine tuned to parse data from another domain. This is in essence different from our work in that we do not adapt the model, rather we adapt seen samples to form parses of new samples. Moreover, We do not fine-tune any part of the model in the new domain and focus on one-shot semantic parsing.

Most of these models need many data-points to train. Therefore, there has been recent attempts at zero-shot semantic parsing. Dadashkarimi et al. (2018) proposed a transfer learning approach where a domain label is predicted first and then the parse. Ferreira et al. (2015) and Herzig and Berant (2018) proposed slot-filling methods for semantic parsing based on general word embeddings. Bapna et al. (2017) focuses on zero-shot frame semantic parsing by leveraging the description of slots to be filled.

## 8 Conclusion and Future Work

As speech recognition technologies mature, more computing devices support spoken instructions via a conversational agent. However, most agents do not adapt to the phrasing and interest of a specific end user. It has recently been shown that new functionalities can be added to an agent from user instruction (Azaria et al., 2016; Labutov et al., 2018). However, the user instruction only provides one instance of a general instruction template and the agent is challenged to generalize to variations of the instance given during instruction. We define the *one-shot parsing* task for measuring a semantic parser's ability to generalize to new instances of user-taught commands from only one example. We propose a new semantic parser architecture that learns a general strategy of retrieving seen utterances similar to an unseen utterance and adapting the logical forms of seen utterances to fit the unseen utterance. Our results show an improvement of up to 68.8% on one-shot parsing under two different evaluation settings compared to the baselines. We found that the BiLSTM encoders are likely bottlenecks for the model. Some future directions include exploring the effects of contents in the memory, automating memory extraction from dataset, and improving the encoder.

## Acknowledgments

This work was supported by AFOSR under grant FA95501710218, NSF under grant IIS1814472, and a Faculty award from J. P. Morgan. The authors would like to sincerely thank Bishan Yang for the initial discussions and ideas related to model architecture, and to Kathryn Mazaitis for the brainstorming sessions on the limitations of the model and future directions.

## References


Amos Azaria, Jayant Krishnamurthy, and Tom Mitchell. 2016. Instructable intelligent personal agent. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 2681–2689.

Ankur Bapna, Gokhan Tur, Dilek Hakkani-Tur, and Larry Heck. 2017. Towards Zero-Shot Frame Semantic Parsing for Domain Scaling. *arXiv e-prints*, page arXiv:1707.02363.

Yoshua Bengio and Yann LeCun, editors. 2015. *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

David Chiang. 2006. An introduction to synchronous grammars.

Javid Dadashkarimi, Alexander Fabbri, Sekhar Tatikonda, and Dragomir R. Radev. 2018. Zero-shot Transfer Learning for Semantic Parsing. *arXiv e-prints*, page arXiv:1808.09889.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43, Berlin, Germany. Association for Computational Linguistics.

Emmanuel Ferreira, Bassam Jabaian, and Fabrice Lefèvre. 2015. Zero-shot semantic parser for spoken language understanding. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor OK Li. 2018. Search engine guided neural machine translation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Kelvin Guu, Tatsunori B Hashimoto, Yonatan Oren, and Percy Liang. 2018. Generating sentences by editing prototypes. *Transactions of the Association for Computational Linguistics*, 6:437–450.

Tatsunori B Hashimoto, Kelvin Guu, Yonatan Oren, and Percy S Liang. 2018. A retrieve-and-edit framework for predicting structured outputs. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 10052–10062. Curran Associates, Inc.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1693–1701. Curran Associates, Inc.

Jonathan Herzig and Jonathan Berant. 2017. Neural semantic parsing over multiple knowledge-bases. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 623–628, Vancouver, Canada. Association for Computational Linguistics.

Jonathan Herzig and Jonathan Berant. 2018. Decoupling structure and lexicon for zero-shot semantic parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1619–1629, Brussels, Belgium. Association for Computational Linguistics.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1378–1387, New York, New York, USA. PMLR.

Igor Labutov, Shashank Srivastava, and Tom Mitchell. 2018. LIA: A natural language programmable personal assistant. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 145–150, Brussels, Belgium. Association for Computational Linguistics.

Percy Liang and Christopher Potts. 2015. Bringing machine learning and compositional semantics together. *Annual Review of Linguistics*, 1:355–376.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Yu Su and Xifeng Yan. 2017. Cross-domain semantic parsing via paraphrasing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1235–1246, Copenhagen, Denmark. Association for Computational Linguistics.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1332–1342, Beijing, China. Association for Computational Linguistics.

W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu, and G. Zweig. 2016. Achieving Human Parity in Conversational Speech Recognition. *arXiv e-prints*, page arXiv:1610.05256.

Luke Zettlemoyer and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing

to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 678–687, Prague, Czech Republic. Association for Computational Linguistics.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, UAI'05, pages 658–666, Arlington, Virginia, United States. AUAI Press.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. *arXiv e-prints*, page arXiv:1709.00103.

# Similarity Based Auxiliary Classifier for Named Entity Recognition

**Shiyuan Xiao**[1,2], **Yuanxin Ouyang**[1,2], **Wenge Rong**[1,2],
**Jianxin Yang**[1,2], **Zhang Xiong**[1,2]
[1]Engineering Research Center of Advanced Computer Application Technology,
Ministry of Education, Beihang University, Beijing, China
[2]School of Computer Science and Engineering, Beihang University, Beijing, China
{xiaoshiyuan, oyyx, w.rong, yjx17, xiongz}@buaa.edu.cn

## Abstract

The segmentation problem is one of the fundamental challenges associated with name entity recognition (NER) tasks that aim to reduce the boundary error when detecting a sequence of entity words. A considerable number of advanced approaches have been proposed and most of them exhibit performance deterioration when entities become longer. Inspired by previous work in which a multi-task strategy is used to solve segmentation problems, we design a similarity based auxiliary classifier (SAC), which can distinguish entity words from non-entity words. Unlike conventional classifiers, SAC uses vectors to indicate tags. Therefore, SAC can calculate the similarities between words and tags, and then compute a weighted sum of the tag vectors, which can be considered a useful feature for NER tasks. Empirical results are used to verify the rationality of the SAC structure and demonstrate the SAC model's potential in performance improvement against our baseline approaches.

## 1 Introduction

Named entity recognition (NER) focuses on extracting a specific sequence and then classifying the sequence as a predefined category. As a fundamental and important task in natural language understanding and information extraction, NER can be considered a sequence labelling task, which includes part-of-speech (POS) tagging, chunking, and semantic role labelling (SRL) (Collobert et al., 2011). Owing to the growing popularity of deep learning techniques, a considerable number of neural network-based approaches and traditional conditional random fields (CRF) have been proposed and are widely used in many sequence labelling tasks to obtain magnificent results (Ye and Ling, 2018; Wu et al., 2018; Ghaddar and Langlais, 2018).

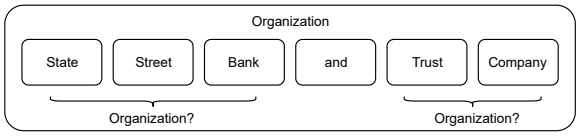


Figure 1: The segmentation problem in NER. "State Street Bank and Trust Company" is an organization entity with six words, it would be unsatisfactory if we consider "State Street Bank" and "Trust Company" to be two entities

To enhance NER's performance, significant effort has been devoted in this area, and one direction is to employ useful information at the input end. This mechanism can be realized by including hand-crafted features (Collobert et al., 2011; Passos et al., 2014; Huang et al., 2015). To eventually reduce heavy feature engineering, some researchers have argued for the exclusion of task-specific features and for the automatic mining of character level features and word level features (Ma and Hovy, 2016; Liu et al., 2018). Such end-to-end models can potentially improve performance with the exclusion of hand-crafted features.

While character and word level features demonstrate promising results, Ye and Ling (2018) determined that such methods (i.e. LM-BiLSTM-CRF (Liu et al., 2018)) will be relatively less effective for longer entity recognition. As indicated by Zhuo et al. (2016), NER is a segment-level task, and such NER models should be able to extract segment information while word-level labels might tend to learn more word level information than segment level information. Such limitations can manifest as entity boundary errors, referred to as the segmentation problem. This problem is illustrated in Figure 1.

There have been many attempts to address the segmentation problem. Some researchers employed segment level information, e.g., Semi-CRF



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1140–1149,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

(SCRF) (Zhuo et al., 2016), to address this challenge. Though SCRF exhibits promising results in the case of longer entity recognition, it requires additional feature extractors and exhibits some limitations in the case of shorter entity recognition. Ye and Ling (2018) proposed using CRF jointly with SCRF to address this problem. On the other hand, researchers have also adopted a multi-task strategy to predict boundaries. For example, Stratos (2016) divides NER into a two-step task in which the boundary is first predicted and then the entity type of the boundary is determined. Though this method is easy to implement, it relies heavily on the accuracy of the first step. Another alternative thought is to employ auxiliary tasks to address this problem. For example, Aguilar et al. (2017) created two auxiliary tasks to help determine entity boundaries and types. However, the authors considered the auxiliary tasks to be supervisors, which means that the outputs of the auxiliary tasks are not fully utilized.

In this study, inspired by the research indicated previously, we design an auxiliary classification task that aims to distinguish entity words from non-entity words before executing the main NER task. Unlike prior studies, the outputs of the auxiliary task do not directly determine the entity boundaries but rather serve as features for assisting the main task. Therefore, the auxiliary task is more like an "advisor" rather than supervisor to the entire model. Our auxiliary classifier is different from conventional classifiers, which compute the probability of an entity word using the $softmax$ function. Instead, we initialize two trainable vectors for the classifier to identify "True" and "False" tags, and then we determine the similarity scores between each word and the two tag vectors to indicate the probability whether the word is an entity word. The hypothesis is that tag vectors can serve as helpful features that are shared by a majority of entity words and non-entity words in NER tasks. Finally, we obtain the weighted sum of the tag vectors by using the similarity scores. In this study, this process is referred to as a similarity based auxiliary classifier (SAC).[1] The empirical results verify the rationality of SAC's structure and demonstrate that the SAC model can reduce segmentation problems and exhibit good performance in recognizing entities of different lengths.

The contributions of this paper are as follows: 1) We introduce a similarity based auxiliary classifier (SAC), which allows the model to determine whether a word is an entity word before predicting its entity label to overcome the segmentation problem. 2) By analyzing the experimental results, we verify the rationality of SAC's structure and demonstrate the advantages of using the weighted sum of tag vectors compared to a list of scores denoting classification results. 3) Without using any hand-crafted features, NeuralCRF+SAC outperforms existing state-of-the-art end-to-end models on CoNLL 2003 and Ontonotes 5.0. When using external resource like ELMo, NeuralCRF+SAC still exhibits comparable results.

## 2 Related work

NER is a fundamental task, and traditional methods usually adopt statistical machine learning models e.g., Hidden Markov Models (HMM), Maximum-entropy Markov Models (MEMM), and Conditional Random Fields (CRF) (Bikei et al., 1998; McCallum et al., 2000; Lafferty et al., 2001). To improve overall performance, many hand-crafted features such as POS chunking tags and gazetteers are required and integrated into the task (Chieu and Ng, 2002; Florian et al., 2003; Ando and Zhang, 2005).

With the growing popularity of deep learning techniques, neural networks have been widely adopted for NER tasks. Collobert et al. (2011) initiated the trend of using neural networks for sequence labelling tasks. Huang et al. (2015) combined BiLSTM with CRF for NER tasks. Because of the powerful feature mining ability of neural networks, some end-to-end models employ neural structures to extract character features and obtain state-of-the-art results without using hand-crafted features (Lample et al., 2016; Ma and Hovy, 2016; Liu et al., 2018). Hand-crafted features such as POS tags, word capitalization, and lexicons are commonly used (Chiu and Nichols, 2016; Ghaddar and Langlais, 2018; Wu et al., 2018). It was recently proved that using contextualized representations from pretrained language models can significantly improve performance (Peters et al., 2017, 2018; Devlin et al., 2018).

To address the segmentation problem, several methods have been proposed, among which Semi-CRF(SCRF) is a successful solution for segment-

[1] code available at https://github.com/XiaoShiyuan/NCRF-SAC

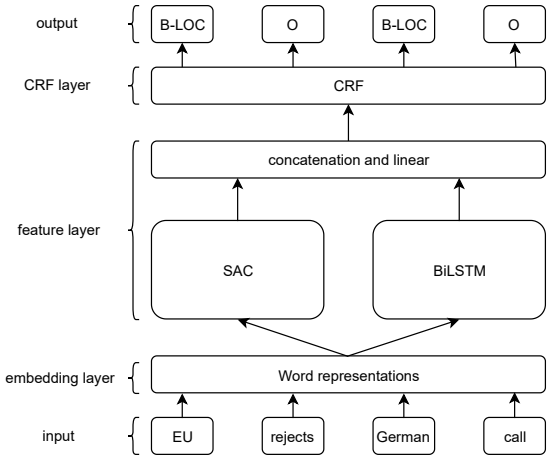


Figure 2: Overall architecture of NeuralCRF+SAC

level modeling (Zhuo et al., 2016). Ye and Ling (2018) proposed an improved version of SCRF, namely HSCRF. Using a multi-task strategy to combine HSCRF and CRF, the HSCRF model exhibits state-of-the-art results without using external resources. In particular, HSCRF resolves the segmentation problem by matching entity lengths. In addition, there are other approaches that consider multi-task learning to predict entity boundaries. Stratos (2016) divides NER into two tasks. First, CRF is used to predict the location and boundary of entities and to classify the entities. Aguilar et al. (2017) used the outputs of their feature extractor to perform segmentation, categorization, and then sequential classification.

Our method exploits the features of multi-task learning and proposes a similarity based auxiliary classifier (SAC). Compared to existing studies that usually consider auxiliary tasks to be supervisors, we also use the output of SAC to support the NER task. Meanwhile, we employ vectors to denote "True" and "False" tags, which contain more useful information compared to simply using "1" and "0" to denote tags.

## 3 Methodology

### 3.1 Overall Architecture

We implement the framework based on BiLSTM-CNN-CRF(NeuralCRF) (Ma and Hovy, 2016), a robust and widely used neural network in English NER tasks, and we build our SAC parallel to BiLSTM. The overall architecture of NeuralCRF+SAC is presented in Figure 2.

**Embedding layer.** Denoting input sequences as $X = \{x_1, x_2, ..., x_T\}$, we first obtain the word-level vector $W = \{w_1, w_2, ..., w_T\}$. Next, a Char-CNN is employed to obtain character-level features $C = \{c_{w_1}, c_{w_2}, ..., c_{w_T}\}$ (Santos and Zadrozny, 2014; Chiu and Nichols, 2016; Ma and Hovy, 2016). Finally, we obtain $[w_i; c_{w_i}]$ to denote $x_i$, where [;] indicates concatenation.

**Feature layer.** In this layer, we employ BiLSTM to extract contextual features and SAC to generate weighted vectors. The calculations can be expressed as follows:

$$\overrightarrow{h_i} = LSTM(\overrightarrow{h_{i-1}}, [w_i; c_{w_i}]) \quad (1)$$

$$\overleftarrow{h_i} = LSTM(\overleftarrow{h_{i-1}}, [w_i; c_{w_i}]) \quad (2)$$

$$h_{a_i} = SAC([w_i; c_{w_i}]) \quad (3)$$

Here, we use $SAC$ to denote all operations of the classifier and we will introduce it in detail in Section 3.2. The three outputs are then concatenated and operated as $z_i = Linear([\overrightarrow{h_i}; \overleftarrow{h_i}; h_{a_i}])$, where $Linear$ denotes a linear calculation.

**CRF layer.** Because of the ability to utilize information from neighboring labels, CRF is widely employed in sequence tagging tasks. Denoting label sequences as $Y = \{y_1, y_2, ..., y_T\}$, the collection of all possible labels as $\mathcal{Y}$, and semantic feature sequence as $Z = \{z_1, z_2, ..., z_T\}$, CRF can be described as a family of conditional probabilities $p(y|Z)$:

$$p(y|Z) = \frac{\prod_{i=1}^{T} \phi_i(y_{i-1}, y_i, z_i)}{\sum_{y' \in \mathcal{Y}} \prod_{i=1}^{T} \phi_i(y'_{i-1}, y'_i, z_i)} \quad (4)$$

where $\phi_i(y_{i-1}, y_i, z_i) = exp(W_{y_{i-1},y} z_i + b_{y_{i-1},y_i})$ is the potential function and $Z = \{z_1, z_2, ..., z_T\}$ are the outputs of the feature layer.

For the training process, we minimize the negative log likelihood:

$$\mathcal{L}_{\mathcal{CRF}} = -\sum_i log(p(y_i|Z)) \quad (5)$$

while the Viterbi algorithm (Forney, 1973) is used to decode the best label sequence.

### 3.2 Similarity based Auxiliary Classifier

We believe that SAC can concentrate on one word and on the nearest neighbors of the word rather than on long-term dependency because we believe that information from the word itself and its nearest neighbors contribute the most in determining whether the word is an entity word. Compared to RNN, CNN exhibits a distinct advantage that

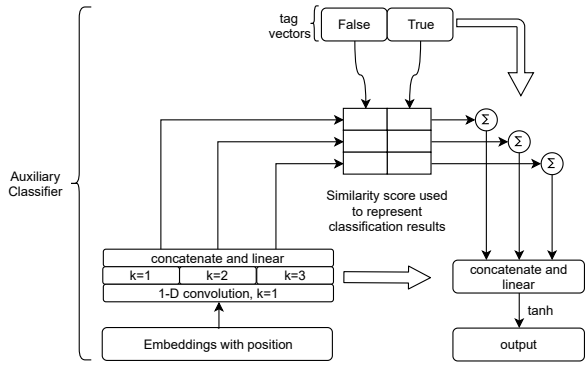


Figure 3: The architecture of SAC. Notice that the convolutional block (lower left) includes 4 1-dimentional convolutional neural networks, which share the same filter settings. Kernel size is marked out.

it can accurately control the length of the context, which satisfies our requirement. Therefore, SAC is based on CNN. The structure of this module is presented in Figure 3. To prevent the loss of positional information while using CNN, we implement a position encoder (Vaswani et al., 2017) to generate position vectors $p_i \in \mathbb{R}^d$, where $d$ is configurable and $p_i$ is concatenated with $[w_i; c_{w_i}]$ to form new embeddings $[w_i; c_{w_i}; p_i]$. The new embeddings are then sent to the convolutional block to extract uni-gram, bi-gram and tri-gram information. We denote the output of this convolutional block as $H_c = \{h_{c_1}, h_{c_2}, ..., h_{c_T}\}$, where $h_{c_i} \in \mathbb{R}^n$.

Intuitively, we adopt a multiplicative attention mechanism (Luong et al., 2015) for computing similarity. For the purpose of classifying entity words and non-entity words, we use $t_i, (i \in \{0, 1\})$ to denote non-entity words and entity words, respectively. Then, we randomly initialize two tag vectors $v_i \in \mathbb{R}^{d_h}, (i \in \{0, 1\})$ as semantic representations of $t_i, (i \in \{0, 1\})$. We express the computation as follows:

$$e_{ij} = v_i^T W_n h_{c_j} \tag{6}$$

$$a_{ij} = \frac{exp(e_{ij})}{\sum_{k=0}^{1} exp(e_{kj})} \tag{7}$$

$$s_j = \sum_{i=0}^{1} a_{ij} v_i \tag{8}$$

$$h_{a_j} = tanh(W_a[h_{c_j}; s_j] + b_a) \tag{9}$$

where $W_n \in \mathbb{R}^{d_h \times n}$, $W_a \in \mathbb{R}^{n \times (n+d_h)}$ and $b_a \in \mathbb{R}^n$. Innovatively, we use $a_{ij}$, the similarity score between $i^{th}$ tag and $h_{c_j}$ as the classification result.

| Dataset | | train | dev | test |
|---|---|---|---|---|
| CoNLL 2003 | #tok | 204567 | 51578 | 46666 |
| | #ent | 23499 | 5942 | 5648 |
| Ontonotes 5.0 | #tok | 1088503 | 147724 | 152728 |
| | #ent | 81828 | 11066 | 11257 |
| WNUT 2017 | #tok | 62729 | 15733 | 23394 |
| | #ent | 3160 | 1250 | 1589 |

Table 1: Statistics of these three datasets, #tok denotes tokens and #ent denotes entities.

For instance, if the $j^{th}$ word is an entity word, its feature representation $h_{c_j}$ must have a higher similarity score with $v_1$. Therefore, SAC operates in a manner that is similar to a "supervised cluster," separating entity words from non-entity words, and denoting two "clustering centers" with two tag vectors. According to Equation 8 and 9, classification information related to the $j^{th}$ word can be expressed as a weighted sum $s_j$ and is then concatenated with $h_{c_j}$. Finally, we obtain the output $H_a = \{h_{a_1}, h_{a_2}, ..., h_{a_T}\}$.

To train this auxiliary classifier, we define attention loss as follows:

$$\mathcal{L}_{\mathcal{A}} = -\frac{1}{T} \sum_{j=1}^{T} \sum_{i=0}^{1} t_i log(a_{ij}) \tag{10}$$

By combining $\mathcal{L}_{\mathcal{CRF}}$ and $\mathcal{L}_{\mathcal{A}}$, we obtain the loss function:

$$\mathcal{L} = \mathcal{L}_{\mathcal{CRF}} + \lambda \mathcal{L}_{\mathcal{A}} \tag{11}$$

We obtain the best and most stable $\lambda = 0.05$ when using the IOBES tag scheme.

## 4 Experiments

### 4.1 Experiments Setup

**Dataset.** We performed experiments on CoNLL 2003 (Tjong Kim Sang and De Meulder, 2003), Ontonotes 5.0 (Hovy et al., 2006) and WNUT 2017 shared tasks (Derczynski et al., 2017). CoNLL 2003 is a widely used dataset comprising data extracted from Reuters news articles. Ontonotes 5.0 is a large-scale dataset comprising data collected from news, weblogs, etc. WNUT 2017 is a small dataset consisting primarily of emerging entities. All datasets have been separated into training/dev/test sets and they include 4/18/6 entity types, respectively. Table 1 presents some statistics of the 3 datasets.

**Configuration.** All models were implemented with TensorFlow [2]. We used GloVe (Pennington

[2] www.tensorflow.org

| Module | Parameter | Value |
|---|---|---|
| Word embedding | dims | 300 |
| Char-CNN | char dims | 100 |
| | kernel size | 3 |
| | filters | 50 |
| BiLSTM | dims | 300 |
| Auxiliary Classifier | pos dims | 40 |
| | tag-vector dims | 25 |
| | CNN-block filters | 300 |
| Others | learning rate | 0.001 |
| | batch size | 20 |
| | epoch | 50 |
| | drop rate | 0.5 |

Table 2: Hyper-parameters

| structure | Accuracy | F1(max) |
|---|---|---|
| BiLSTM | 96.97 | 91.20 |
| **CNN** | **97.87** | **91.65** |

Table 3: Comparing different structure of SAC. Accuracy indicates that of an auxiliary classification task and F1 for an NER task

et al., 2014) (300-dim, trained on 840B corpus[3]) pretrained word embeddings, and the same hyper-parameters on BiLSTM-CNN-CRF(NeuralCRF) and NeuralCRF+SAC. To determine performance using external resources, we included ELMo representations. All hyper-parameters, most of which are empirical parameters, are listed in Table 2 and an early stopping strategy was adopted. The main experiments were tested for 5 times to obtain the mean and standard deviation of F1 score.

**Evaluation.** After training, the latest 50 checkpoints were preserved and we obtained the checkpoint with the highest word-level F1-score to make predictions on the test set. Following this, we used the official CoNLL evaluation script on the CoNLL 2003 dataset and ontonotes 5.0 and the WNUT evaluation script on the WNUT 2017 dataset.

### 4.2 Development experiments

To verify the rationality of the structure of SAC and the role of the tag vectors, we designed two development experiments on the CoNLL 2003 dataset. One experiment compared the CNN structure in SAC with the BiLSTM structure. The other one compared SAC with a modified TextCNN.

**Comparing with an RNN structure** We believe that if a word is an entity word, its nearest neighbors are very likely to be entity words as well. Although there are many one-word entities, these entities always share an obvious feature such as capital letters, which can be easily captured by a char-CNN. On the contrary, if a word is a non-entity word, under most circumstances, there would be at most one entity word among the adjacent words. However, when considering long-term dependency, a word can be affected by several former or latter words, which could weaken the influence of adjacent words and introduce unnecessary noise. Therefore, we chose CNN instead of RNN in SAC and we set the max kernel size to 3.

To verify if a CNN structure is more suitable in SAC, we replace the convolutional block with BiLSTM. In Table 3, we can observe that the maximum F1 value of the entire model is 91.65 when employing a CNN structure compared to 91.20 when using BiLSTM. To determine the reason for a large performance drop, we compare the performances of the two structures in the face of different entity lengths. From Figure 4, we can observe that if SAC uses a BiLSTM structure, the overall performance will still deteriorate as entities become long, indicating that SAC appears to be redundant for the entire model. Maybe it is because BiLSTM also exists in NeuralCRF, and using BiLSTM in SAC may cause NeuralCRF and SAC to obtain duplicate features.

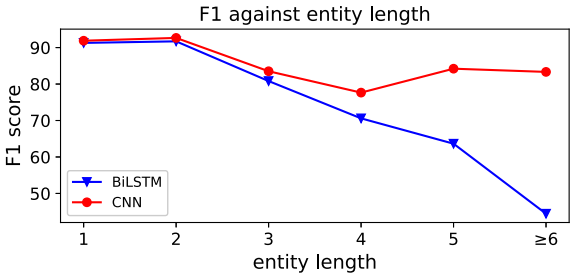


Figure 4: Comparing the ability of SAC with different structures to identify different length entities.

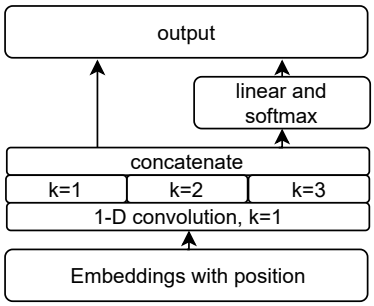


Figure 5: The structure of our TextCNN.

[3] http://nlp.stanford.edu/data/glove.840B.300d.zip

| structure | Accuracy | F1(max) |
|---|---|---|
| NeuralCRF+TextCNN | **98.29** | 91.47 |
| **NeuralCRF+SAC** | 97.87 | **91.65** |

Table 4: Compare SAC with TextCNN. Accuracy is for the auxiliary classification task and F1 for NER task.

| Models | type | F1(±std) |
|---|---|---|
| Zhuo et al. (2016) | reported | 88.12 |
| Lample et al. (2016) | reported | 90.94 |
| Ma and Hovy (2016) | reported | 91.21 |
| Strubell et al. (2017) | reported | 90.54 |
| Liu et al. (2018) | mean | 91.24(±0.12) |
| | max | 91.35 |
| Ye and Ling (2018) | mean | 91.38(±0.10) |
| | max | 91.53 |
| NeuralCRF | mean | 91.22(±0.09) |
| | max | 91.30 |
| **NeuralCRF+SAC** | **mean** | **(91.48±0.07)** |
| | **max** | **91.65** |

Table 5: Results on CoNLL 2003 NER task without external resources.

**Comparing with TextCNN** In addition, we employed a conventional classifier like TextCNN (Kim, 2014) to replace SAC. As is known, TextCNN predicts a set of scores for each sentence according to the number of categories. The position of the largest score indicates the category of the sentence. As is shown in Figure 5, to enable TextCNN to predict scores for each word, and to make a fair comparison with SAC, we remove the pooling layer of TextCNN and adopt the same convolutional structure as SAC. Meanwhile, both outputs of TextCNN and SAC include features mined by their convolutional layers. The difference lies in the other part of their outputs in which TextCNN includes a series of scores while SAC provides the weighted sum of the tag vectors.

Table 4 demonstrates that NeuralCRF+SAC obtains a maximum F1 score of 91.65 while NeuralCRF+TextCNN receives a score of 91.47. However, we can observe that TextCNN exhibits higher accuracy on its own classification task. We must declare that the accuracy of SAC has different meanings in terms of the accuracy of TextCNN. For TextCNN, an accuracy score of 98.29 only means that TextCNN assigns 98.29% of the words with the right tags. However, for SAC, an accuracy score of 97.87% also means that 97.87% of the words share common traits in which a majority of entity words are similar to the "True" tag vector, while the majority of non-entity words are similar to "False" tag vector. Therefore, because of the higher F1 score obtained from NeuralCRF+SAC, the two tag vectors can be viewed as features that are helpful for the NER task. Just like our baseline approach, NeuralCRF, uses char-CNN to mine character-level features. SAC automatically mines features that are shared by entity words or non-entity words.

| Models | resources | F1(±std) |
|---|---|---|
| Chiu and Nichols (2016) | Lexicons | 91.62(±0.33) |
| Peters et al. (2017) | TagLM† | 91.93(±0.19) |
| Ghaddar and Langlais (2018) | gazetteer, LS | 91.73 |
| Wu et al. (2018) | POS,etc. | 91.89(±0.23) |
| Peters et al. (2018) | ELMo† | 92.22(±0.10) |
| Devlin et al. (2018) | BERT(BASE)† | 92.40 |
| Devlin et al. (2018) | **BERT(LARGE)†** | **92.80** |
| NeuralCRF | ELMo† | 92.20(±0.06) |
| NeuralCRF+SAC | ELMo† | 92.57(±0.07) |

Table 6: Results of using the CoNLL 2003 NER task with ELMo representations. † denotes the language model trained on external resources.

| Models | resources | F1(±std) |
|---|---|---|
| Shen et al. (2017) | None | 86.63(±0.49) |
| Strubell et al. (2017) | None | 86.84(±0.19) |
| NeuralCRF | None | 86.65(±0.10) |
| **NeuralCRF+SAC** | **None** | **87.24(±0.07)** |
| Chiu and Nichols (2016) | Lexicons | 86.28(±0.26) |
| Ghaddar and Langlais (2018) | gazetteer, LS | 87.95 |
| NeuralCRF | ELMo | 88.41(±0.06) |
| **NeuralCRF+SAC** | **ELMo** | **89.05(±0.18)** |

Table 7: Results on Ontonotes 5.0 dataset.

### 4.3 Results on CoNLL 2003 dataset

**End-to-end.** Table 5 presents the experimental results[4] on the CoNLL 2003 datasets without external resources. We can observe that our model exhibits a mean F1 score improvement of 0.26 compared to the base model and outperforms all existing end-to-end models.

**With ELMo representations.** In addition, be-

[4]The first author of (Liu et al., 2018) indicates errors pertaining to the original reported results because of some bugs and suggests citing fixed results. (https://github.com/LiyuanLucasLiu/LM-LSTM-CRF)

| Models | F1(entity) | F1(surface) |
|---|---|---|
| Lin et al. (2017) | 40.42 | 37.62 |
| von Däniken et al. (2017) | 40.78 | 39.33 |
| Aguilar et al. (2017) | 41.86 | 40.24 |
| NeuralCRF | 43.93 | 41.80 |
| **NeuralCRF+SAC** | **44.77** | **43.29** |

Table 8: Results of the WNUT 2017 shared task

| Length | Ye and Ling (2018) | NeuralCRF+SAC | Δ |
|---|---|---|---|
| 1 | 91.73 | 91.75 | +0.02 |
| 2 | 92.03 | 92.70 | +0.67 |
| 3 | 83.78 | 83.26 | -0.52 |
| 4 | 77.27 | 78.57 | +0.30 |
| 5 | 79.66 | 80.00 | +0.34 |
| $\geq 6$ | 76.55 | 76.92 | +0.37 |
| overall | 91.38 | 91.48 | +0.10 |

Table 9: Comparing performance with Ye and Ling (2018).

cause of the popularity of contextualized representation, we introduce EMLo[5] in the embedding layer as a form of external resources and results, which are presented in Table 6. Observe that our model still exhibits a 0.37 F1 improvement over the base model, which verifies the effects of the SAC. Despite using BERT(LARGE), we still obtain state-of-the-art results, even though it requires a considerable number of resources to be expended. In contrast, our method is flexible and resources friendly.

### 4.4 Results on Ontonotes 5.0 dataset

We present the experimental results on Ontonotes 5.0 in Table 7. Without using external resources, our model exhibits a mean F1 score of 87.24, which still exceeds the previous end-to-end model. When employing ELMo as external resources, we obtain a mean F1 score of 89.05. Comparing to NeuralCRF, SAC brings 0.59 and 0.64 F1 improvements respectively.

### 4.5 Results on WNUT 2017 shared task

Experimental results on the WNUT17 dataset are presented in Table 8. Owing to the facts that this task focuses on unusual and previously-unseen entities, all existing models incorporate some handcrafted features. We include ELMo representations in this experiment. SAC exhibits a 0.84 improvement in F1(entity) and 1.49 improvement in F1(surface).

[5]https://allennlp.org/elmo

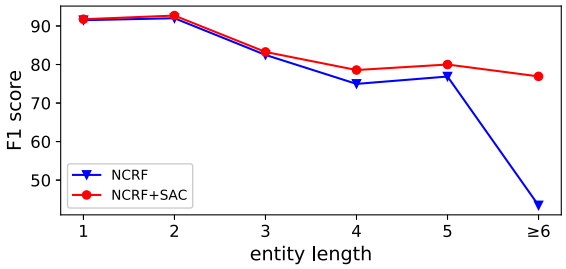


(a) F1 against entity length(CoNLL 2003)

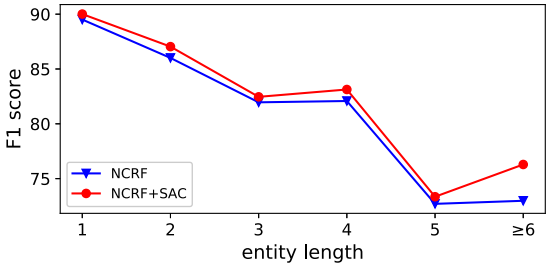


(b) F1 against entity length(Ontonotes 5.0)

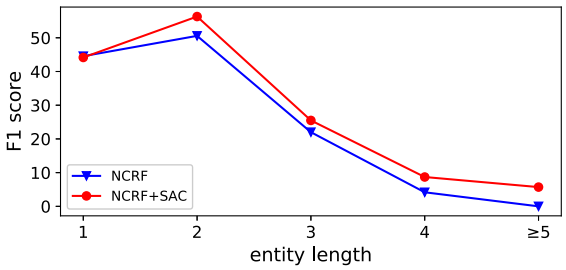


(c) F1 against entity length(WNUT 2017, with ELMo)

Figure 6: F1 against length test on all three datasets.

### 4.6 Analysis

**Performance against entity length.** We believe that before using CRF to predict the final label of a word, using a classifier to determine whether the word is an entity word helps in addressing problems related to segmentation, particularly when entities become long. Following Ye and Ling (2018)'s study, we compared the performance of NeuralCRF and NeuralCRF+SAC in identifying entities of different lengths on three datasets. The experimental results of the CoNLL 2003, Ontonotes 5.0, and WNUT 2017 datasets are presented in Figure 6. To eliminate the impact of external resources, we use the end-to-end approach as far as possible. But because the WNUT 2017 datasets are relatively small and most of them are emerging entities, which makes both

| Ground truth | Models | Results |
|---|---|---|
| the Financial Times of Londom | NeuralCRF | correspondent for the Financial Times(ORG) of London(GPE)... |
| | NeuralCRF+SAC | correspondent for the Financial Times of Londom(ORG)... |
| What Else is Making News | NeuralCRF | Watch What Else(creative-work) is Making News |
| | NeuralCRF+SAC | Watch What Else is Making News(creative-work) |
| Monopolies and Mergers Commission | NeuralCRF | ...Monopolies(ORG) and Mergers Commission(ORG) unless the carriers complied... |
| | NeuralCRF+SAC | ...Monopolies and Mergers Commission(ORG) unless the carriers complied... |

Table 10: Three examples, identified entities are highlighted

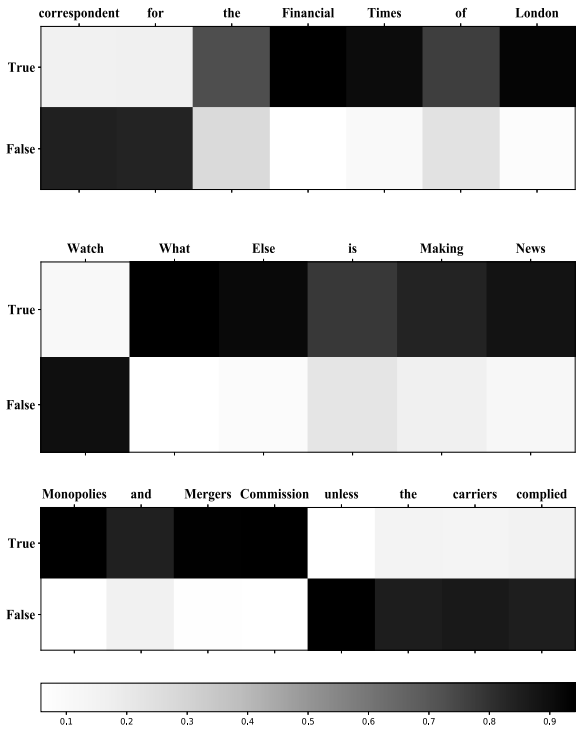


Figure 7: Similarity scores of three examples

models incapable of recognizing entities consisting of 4 or more words, we employed ELMo in WNUT 2017.

Observe that Figure 6(a) indicates a big improvement when entity lengths are equal to or greater than 4 and Figures 6(b) and 6(c) illustrate even improvements against different lengths.

In addition, we compared NeuralCRF+SAC with Ye and Ling (2018)' work in Table 9. The two models perform comparably when the entity length is less than 4, but NeuralCRF+SAC exhibits even improvements when the entity length is equal to or larger than 4, which demonstrates that SAC can improve the overall performance in long entity recognition.

**Case study.** We examined the manner in which SAC handles long entities. Table 10 presents three cases from 3 datasets, respectively. We observe that these 3 cases are quite similar and NeuralCRF makes almost the same mistake on each of them. Generally speaking, character level features are quite powerful for the NER task that words written in uppercase letters are easily recognized as entity words. In addition, if all the letters of a word are lowercase letters, they are more likely to be classified as non-entity words. Therefore, we can observe that cases in which meeting words like "of", "is" and "and" are too normal to be considered entity words, it is very easy for NeuralCRF to predict an "O" label for these words. In fact, many long entities exhibit a pattern in which almost all words are written in uppercase letters except one. Therefore, it is common that the segmentation problem grows in severity as entities become longer. However, these three examples indicate that SAC can address this problem to some extent. We obtain the similarity scores of these examples and present them in Figure 7. It is obvious that SAC assigns a high similarity score to "of", "is" and "and" even though they hardly include any features of entity words.

A possible reason is that the CNN structure of SAC plays a role. As discussed in Section 4.2, CNN can control the length of the context. When the kernel sizes are set to 1, 2 and 3, CNN layers can extract uni-gram, bi-gram and tri-gram features that enable SAC to learn a considerable number of samples that exhibit the same structure as "Time of London." Therefore, SAC reduces truncation errors for long entities caused by lowercase words.

## 5 Conclusion

In this work, we extended the BiLSTM-CNN-CRF architecture with SAC to address the segmentation problem. By computing the similarity scores between words and two tag vectors, respectively, SAC provides the weighted sum of tag vectors together with features it mined to support the main NER task. Because of the higher accuracy of SAC, the two tag vectors can be considered features that are shared between entity words and non-entity words. By reducing the boundary errors, SAC addresses the segmentation problem and improves overall performance, particularly when entities become long. Experimental results demonstrate that NeuralCRF+SAC surpasses previous state-of-the-art end-to-end models on the CoNLL 2003 and Ontonotes 5.0 datasets without using external resources. Moreover, when including ELMo, SAC still improves overall performance and is comparable to current state-of-the-art models.

## Acknowledgments

This work was partially supported by the National Natural Science Foundation of China (No. 61977002)

## References

Gustavo Aguilar, Suraj Maharjan, A Pastor López-Monroy, and Thamar Solorio. 2017. A multi-task approach for named entity recognition in social media data. *W-NUT 2017*, page 148.

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6(Nov):1817–1853.

Daniel M. Bikei, Scott Miller, Richard Schwartz, and Ralph Weischedel. 1998. Nymble: High-performance learning name-finder. *Anlp*, pages 194–201.

Hai Leong Chieu and Hwee Tou Ng. 2002. Named entity recognition: a maximum entropy approach using global information. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

Jason PC Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Pius von Däniken, AG SpinningBytes, and Mark Cieliebak. 2017. Transfer learning and sentence level features for named entity recognition on tweets. *W-NUT 2017*, page 166.

Leon Derczynski, Eric Nichols, Marieke van Erp, and Nut Limsopatham. 2017. Results of the wnut2017 shared task on novel and emerging entity recognition. *W-NUT 2017*, page 140.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. 2003. Named entity recognition through classifier combination. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 168–171. Association for Computational Linguistics.

G David Forney. 1973. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278.

Abbas Ghaddar and Phillippe Langlais. 2018. Robust lexical features for improved neural network named-entity recognition. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1896–1907.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: The 90\% solution. In *Proceedings of the human language technology conference of the NAACL, Companion Volume: Short Papers*.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

John Lafferty, Andrew Mccallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proceedings of Icml*, 3(2):282–289.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT*, pages 260–270.

Bill Y Lin, Frank F Xu, Zhiyi Luo, and Kenny Q Zhu. 2017. Multi-channel bilstm-crf model for emerging named entity recognition in social media. *W-NUT 2017*, page 160.

L. Liu, J. Shang, F. Xu, X. Ren, H. Gui, J. Peng, and J. Han. 2018. Empower sequence labeling with task-aware neural language model. In *AAAI*.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1064–1074.

Andrew McCallum, Dayne Freitag, and Fernando CN Pereira. 2000. Maximum entropy markov models for information extraction and segmentation. In *Icml*, volume 17, pages 591–598.

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. *arXiv preprint arXiv:1404.5367*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1756–1765.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 2227–2237.

Cicero D Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1818–1826.

Yanyao Shen, Hyokun Yun, Zachary C Lipton, Yakov Kronrod, and Animashree Anandkumar. 2017. Deep active learning for named entity recognition. *arXiv preprint arXiv:1707.05928*.

Karl Stratos. 2016. Entity identification as multitasking. *arXiv preprint arXiv:1612.02706*.

Emma Strubell, Patrick Verga, David Belanger, and Andrew McCallum. 2017. Fast and accurate entity recognition with iterated dilated convolutions. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2670–2680.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2003)*, pages 142–147.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Minghao Wu, Fei Liu, and Trevor Cohn. 2018. Evaluating the utility of hand-crafted features in sequence labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2850–2856.

Zhixiu Ye and Zhen-Hua Ling. 2018. Hybrid semi-markov crf for neural sequence labeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 235–240.

Jingwei Zhuo, Yong Cao, Jun Zhu, Bo Zhang, and Zaiqing Nie. 2016. Segment-level sequence modeling using gated recursive semi-markov conditional random fields. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1413–1423.

# Variable beam search for generative neural parsing and its relevance for the analysis of neuro-imaging signal

**Benoît Crabbé**
LLF - CNRS
University of Paris - IUF
Place Paul Ricoeur
75013 Paris France
benoit.crabbe@
linguist.univ-paris-diderot.fr

**Murielle Popa-Fabre**
ALMANACH - INRIA - LLF
University of Paris
2 rue Simone Iff
75012 Paris France
murielle.fabre@inria.fr

**Christophe Pallier**
Cognitive Neuroimaging Lab
INSERM-CEA
Neurospin bat.145
Gif-sur-Yvette France
christophe@pallier.org

## Abstract

This paper describes a method of variable beam size inference for Recurrent Neural Network Grammar (RNNG) by drawing inspiration from sequential Monte-Carlo methods such as particle filtering.

The paper studies the relevance of such methods for speeding up the computations of direct generative parsing for RNNG. But it also studies the potential cognitive interpretation of the underlying representations built by the search method (beam activity) through analysis of neuro-imaging signal through correlations with the neuro-imaging signal during naturalistic text listening

## 1 Introduction

The paper provides a preliminary investigation of how the improvements in natural language parsing techniques can be used for modelling brain activity during online sentence comprehension. Specifically, we focus on the RNNG generative parsing framework (Dyer et al., 2016), that allows us to extract information-theoretic measures that are relevant for cognitive studies such as surprisal or entropy (Hale, 2016).

RNNG is essentially a generative parsing framework but it is not straightforward to design a one-step generative parser. Dyer et al. (2016) achieves generative parsing with RNNG in two steps: by reranking a discriminative parser. Stern et al. (2017) points out that the main difficulty comes from lexical biases using naive beam search. They propose new methods for modifying a classical word-synchronized beam search for avoiding the lexical bias problem.

The paper presents a substantially different search method for parsing in one-step with a neural generative parser, proposing a method inspired by sequential Monte-Carlo sampling and particle filtering (Doucet and Johansen, 2011; Buys and Blunsom, 2015).

Crucially, the proposed method is naturally not sensitive to problems of lexical biases faced by standard beam search. As a result, the parser is framed as generative and strictly incremental (without lookahead) while remaining quite accurate and generally more efficient than a more traditional beam search method. Not only, the proposed search method is a variable-beam search method, but as such, it naturally provides an additional method to measure the activity required to make the parser progress incrementally and word-by-word that can, in principle, be used for cognitive modelling purposes.

Taking advantage of our search method properties, we introduce new parsing complexity measures such as beam-size and beam-activity, adding to the well known entropy or surprisal that are already provided by the generative model. Finally, we evaluate the potential cognitive relevance of such measures for analyzing neuro-imaging data. Instantiating syntactic processing in terms of parser's analyses has already highlighted the language brain network to a large extent in the neuroimaging literature, we therefore expect that the goodness of fit of our parsing model with fMRI signal can be taken as an index of cognitive relevance. In other words, we aim at using our parsers' beam activity as a *proxy* modelling the kind of processes that might take place during online sentence comprehension, and thus investigate the potential cross-fertilization between computational solutions and cognitive-neuro imaging approaches to human parsing. The paper is organized as follows. Section 2 first describes an in-order variant of RNNG that is used as underlying formal framework supporting our experiments. It describes the variable beam search method inspired by particle filtering and finally introduces metrics quantifying



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1150–1160,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

online parsing process. Section 3 provides empirical measures and quantifies the behaviour of the parser and of the proposed search method. Section 4 describes the relationship between the parsing model and the analysis of neuro-imaging data, section 5 presents the results.

## 2 RNNG and language modelling

### 2.1 In-order RNNG

We use an in-order variant of RNNG (Liu and Zhang, 2017; Kuncoro et al., 2017) where the set of transitions echoes the one of a left-corner parser. A configuration is a triple $\langle \mathbf{S}, \mathbf{B}, n \rangle$ where $\mathbf{S}$ is a stack, $\mathbf{B}$ is a buffer and $n$ a counter of open brackets.

**Init** $\langle \emptyset, x_1 \dots x_n, 0 \rangle$

**Goal** $\langle S_0 \bullet, \emptyset, 0 \rangle$

**Generate**(x) $\frac{\langle \mathbf{S}, \bullet x_i | \mathbf{B}, n \rangle}{\langle \mathbf{S} | x_i \bullet, \mathbf{B}, n \rangle}$

**Open**(N) $\frac{\langle \mathbf{S} | S_0 \bullet, \mathbf{B}, n \rangle}{\langle \mathbf{S} | \bullet N | S_0 \bullet, \mathbf{B}, n+1 \rangle}$

**Close** $\frac{\langle \mathbf{S} | \bullet N_i | \dots | S_0 \bullet, \mathbf{B}, n \rangle}{\langle \mathbf{S} | N \bullet, \mathbf{B}, n-1 \rangle}$

Each **Open** action is parametrized by a nonterminal symbol, thus there are as many nonterminals in the set $A$ of actions as there are nonterminal symbols. As the parser is generative, there are as many generate actions as there are words $x$ in the vocabulary $X$. Note however that we constrain the generate action to be the words $x_1 \dots x_n$ of the actual sentence to parse. Observe also that the set of actions differs from the standard top down formulation of RNNG (Dyer et al., 2016) and from the in-order transition system of (Liu and Zhang, 2017) only because of the definition of the **Open** action: our formulation pops from the stack the left corner constituent $S_0$ and pushes back on the stack the predicted category $\bullet N$ before $S_0$. That is, during parsing, the stack keeps the same internal structure as in the top down version (Dyer et al., 2016) but the tree traversal is different.

A derivation $(\mathbf{x}, \mathbf{y}) = a_1 \dots a_m$ of an RNNG is a sequence of actions $a_1 \dots a_m$. The weight of a derivation of length $m$ is defined as

$$P(\mathbf{x}, \mathbf{y}) = \sum_{j=1}^{m} \log P(a_j | a_1 \dots a_{j-1})$$

At each inference step the parser has to score the set of actions. The neural model seeks to assign a probability distribution $\mathbf{p}$ to the set $A$ of actions given the parsing context:

$$\mathbf{p} = \textsc{Softmax}(\mathbf{W}\mathbf{h} + \mathbf{b})$$

The parsing context $\mathbf{h}$ is itself encoded by a stack-LSTM (Dyer et al., 2016):

$$\mathbf{h} = \textsc{Stack-Lstm}(\mathbf{e}_1 \dots \mathbf{e}_n)$$

where each $\mathbf{e}_i$ is either a word $\mathbf{e}_i^W$, a nonterminal $\mathbf{e}_i^N$, or a tree embedding $\mathbf{e}_i^T$. Word representations are pushed on the stack-LSTM by the **generate** action and are the concatenation of a word embedding $\mathbf{w}$ and an embedding of the word's characters $c_1 \dots c_n$:

$$\mathbf{e}_i^W = [\mathbf{w}; \textsc{Bi-Lstm}(c_1 \dots c_n)]$$

Nonterminal $\mathbf{e}_i^N$ and tree embeddings $\mathbf{e}_i^T$ are computed using the same methods as in (Dyer et al., 2016).

As $\mathbf{p}$ encodes the conditional distribution $P(a_i | a_1 \dots a_{i-1})$, the network is trained on treebank data of $N$ sentences to maximize the log likelihood:

$$\ell(\boldsymbol{\theta}) = \sum_{i=1}^{N} \sum_{j=1}^{m} \log P(a_j | a_1 \dots a_{j-1})$$

### 2.2 Variable beam-size inference

**Problem with naive beam search** RNNG is initially formalized as a two stage parsing process (Dyer et al., 2016): first a discriminative model with lookahead is run on input sentences and then a generative model without lookahead is used to rerank the best hypotheses.

However direct generative parsing with RNNG is difficult because word generation transitions have significantly lower probabilities than structural transitions, such as open and close transitions. Thus, inside a beam at the same temporal step, these lexical transitions are likely to be pruned. This situation creates a bias towards parsing with additional spurious structure.

Some solutions already described in Stern et al. (2017) and Hale et al. (2018) manage to deal with this problem for direct generative parsing but this solution still requires to use rather large beams to yield accurate results, hence entailing significant computational overhead in time.

Among the alternative ways to solve the issue, the two stage architecture (Dyer et al., 2016) is not

an option in our case because it relies on a discriminative model with lookahead that would break the strict incrementality property that we want to preserve for the cognitive perspective. Furthermore, in our scientific context, we have to study two inference problems: (1) the usual search for a best parse that is required for assessing the model accuracy with known metrics, and (2) for the purpose of fMRI modelling, we also have to compute the marginal probabilities required to compute the probabilities relevant for language modelling.

**Proposed solution** We therefore study another solution to this problem that is inspired by Sequential importance sampling methods (Doucet and Johansen, 2011; Buys and Blunsom, 2015) and that can be interpreted as a variable beam search strategy where the search method carefully avoids to compare apples and oranges within the same beam. Specifically, the method ensures that comparisons between derivations are made only if derivations have an identical number of generated lexical elements.

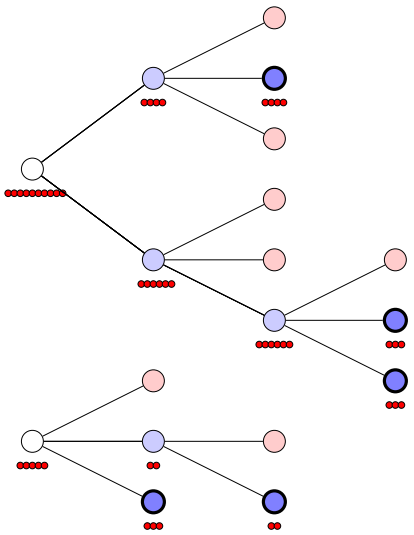

Figure 1: Example of a sampling step from derivations generating $x_i$ (white nodes) to derivations generating $x_{i+1}$ (circled blue nodes) with a budget of $K = 15$ particles. Inference may stop because of a lack of budget ($\pi(\mathbf{x}, \mathbf{y}) = 0$) as illustrated by red nodes. Derivations are never compared to each other during the sampling step, hence avoiding lexical biases that hamper the process of beam search.

For a sequence of words $\mathbf{x} = x_1 \dots x_i \dots x_n$, we note $\mathbf{x}_i$ the prefix $x_1 \dots x_i$ and $\mathbf{y}_i = a_1 \dots a_k$ a derivation whose last element generates $x_i$. We let $\mathcal{Y}(\mathbf{x}_i)$ denote the set of such derivations.

The search method essentially performs iteratively a move from word $x_i$ to its successor $x_{i+1}$ using the following procedure: given a distribution $P(\mathbf{x}_i, \mathbf{y}_i)$ over derivations at $x_i$, the algorithm samples the distribution $P(\mathbf{x}_{i+1}, \mathbf{y}_{i+1})$ with a swarm of particles. This procedure is decomposed in two steps:

**Step 1: Sampling**. Let ($\mathbf{x}$,$\mathbf{y}$) be a derivation together with its associated number of particles $\pi(\mathbf{x}, \mathbf{y})$, then each of its successors ($\mathbf{x}$,$\mathbf{y}a$) ($a \in A$) gets its particles assigned using the following recurrence[1]:

$$\pi(\mathbf{x}, \mathbf{y}a) = \lfloor \pi(\mathbf{x}, \mathbf{y}) P^*(a|\mathbf{x}, \mathbf{y}) \rceil \quad (1)$$

This step involves expanding the set of derivations. Each derivation ($\mathbf{x}$,$\mathbf{y}$) is expanded iteratively until either the word $x_{i+1}$ is generated or ($\mathbf{x}$,$\mathbf{y}$) has no more particle ($\pi(\mathbf{x}, \mathbf{y}) = 0$). This procedure is illustrated in Figure 1.

We have to emphasize, however, that contrary to $P(a|\mathbf{x}, \mathbf{y})$, the chosen importance distribution $P^*(a|\mathbf{x}, \mathbf{y})$ locally sums to one. It is indeed defined as:

$$P^*(a|\mathbf{x}, \mathbf{y}) = \frac{P(a|\mathbf{x}, \mathbf{y})}{\sum_{a' \in A^*} P(a'|\mathbf{x}, \mathbf{y})} \quad (2)$$

where $A^* \subseteq A$ is a set of allowed actions given $(\mathbf{x}, \mathbf{y})$. Using the importance distribution prevents particles to get lost because of deficient probability distributions. Observe that $P(a|\mathbf{x}, \mathbf{y})$ does not sum to one because the generative model prevents a subset of actions to be undertaken at any time step. This is crucially the case for lexical actions when parsing: only the next word form given in the input sentence can be generated at any time step while there are lexical actions for generating every word in the vocabulary. This causes $P(a|\mathbf{x}, \mathbf{y})$ to be deficient.

A connection with importance sampling is established if we define the importance weight of a derivation as:

$$w(\mathbf{x}, \mathbf{y}) = \frac{P(\mathbf{x}, \mathbf{y})}{P^*(\mathbf{x}, \mathbf{y})}$$

where $P^*(\mathbf{x}, \mathbf{y}) = \prod_{i=1}^{m} P^*(a_j|a_1 \dots a_m)$. We can then observe the relation between the three notions by developing:

$$\begin{aligned} P(\mathbf{x}, \mathbf{y}) &= \frac{P(\mathbf{x}, \mathbf{y})}{P^*(\mathbf{x}, \mathbf{y})} P^*(\mathbf{x}, \mathbf{y}) \\ &= w(\mathbf{x}, \mathbf{y}) P^*(\mathbf{x}, \mathbf{y}) \end{aligned} \quad (3)$$

[1] We use $\lfloor x \rceil$ to denote the rounding of $x$ to the nearest integer.

**Step 2: Reweighting** and **Filtering**. Observe, however, that the recurrence (1) spawns a search tree and that the mass of particles assigned to search branches gets increasingly scattered as the process progresses with further time steps. To counter this effect, we reweight the branches by synchronizing the search process on word generation events. In other words, each derivation $(\mathbf{x}_{i+1}, \mathbf{y}_{i+1}) \in \tilde{\mathcal{Y}}(\mathbf{x}_{i+1})$ that successfully generated the word $x_{i+1}$ is reweighted by normalizing its importance weight:

$$w'(\mathbf{x}_{i+1}, \mathbf{y}_{i+1}) = \frac{w(\mathbf{x}_{i+1}, \mathbf{y}_{i+1})\pi(\mathbf{x}_{i+1}, \mathbf{y}_{i+1})}{\sum_{\mathbf{y}'} w(\mathbf{x}_{i+1}, \mathbf{y}')\pi(\mathbf{x}_{i+1}, \mathbf{y}')} \tag{4}$$

We subsequently reassign each derivation with a number of particles $\pi'(\mathbf{x}_{i+1}, \mathbf{y}_{i+1})$ proportional to its normalized weight. Let $K$ be the global number of particles available, then we reallocate particles to each derivation as follows:

$$\pi(\mathbf{x}_{i+1}, \mathbf{y}_{i+1}) = \lfloor K w'(\mathbf{x}_{i+1}, \mathbf{y}_{i+1}) \rceil$$

Derivations without particles are filtered out at this stage ($\pi(\mathbf{x}_{i+1}, \mathbf{y}_{i+1}) = 0$). Figure 2 summarizes the search algorithm with pseudo code. It can be interpreted as an adaptation of particle filtering (Doucet and Johansen, 2011) to our generative parsing task.

**Marginal probabilities** The proposed exploration method of the search space finds a direct application to computing marginal probabilities for sentential prefixes of the form $P(\mathbf{x}_i)$. For our purposes, one key interest of the generative parsing model lies in its capacity to compute transition probabilities $P(x_i|x_1 \dots x_{i-1})$ informed by the syntactic structures of the sentence. This enables the computation of metrics that have been shown relevant for psycho-linguistic analysis such as surprisal and entropy, among others.

We briefly summarize in the following how this is achieved through the computation of marginal probabilities. We can define the probability of a prefix as:

$$P(\mathbf{x}_i) = \sum_{\mathbf{y}_i \in \mathcal{Y}(\mathbf{x}_i)} P(\mathbf{x}_i, \mathbf{y}_i) \tag{5}$$

Thus computing transition probabilities can be simply performed as:

$$P(x_i|x_1 \dots x_{i-1}) = \frac{P(\mathbf{x}_i)}{P(\mathbf{x}_{i-1})} \tag{6}$$

**function** VARIABLEBEAMSEARCH($x_1 \dots x_n$,$K$)
  $\tilde{\mathcal{Y}}(\mathbf{x}_0) \leftarrow (\epsilon, \epsilon)$
  **for** $i \in 0 \dots n-1$ **do**
    agenda $\leftarrow \tilde{\mathcal{Y}}(\mathbf{x}_i)$ ▷ Sampling step
    $\tilde{\mathcal{Y}}(\mathbf{x}_{i+1}) \leftarrow \emptyset$
    **while** agenda $\neq \emptyset$ **do**
      $(\mathbf{x}, \mathbf{y}) \leftarrow$ POP(agenda)
      **for** $a \in A$ **do**
        $\pi(\mathbf{x}, \mathbf{y}a) \leftarrow \lfloor \pi(\mathbf{x}, \mathbf{y}) P^*(a|\mathbf{x}, \mathbf{y}) \rceil$
        **if** $\pi(\mathbf{x}, \mathbf{y}a) > 0$ **then**
          **if** $a$ generates $x_{i+1}$ **then**
            $\tilde{\mathcal{Y}}(\mathbf{x}_{i+1}) \leftarrow \tilde{\mathcal{Y}}(\mathbf{x}_{i+1}) \cup (\mathbf{x}a, \mathbf{y})$
          **else**
            agenda $\leftarrow$ agenda $\cup (\mathbf{x}, \mathbf{y}a)$
          **end if**
        **end if**
      **end for**
    **end while**
    **for** $(\mathbf{x}, \mathbf{y}) \in \tilde{\mathcal{Y}}(\mathbf{x}_{i+1})$ **do** ▷ Reweighting
      $w(\mathbf{x}, \mathbf{y}) \leftarrow \frac{w(\mathbf{x},\mathbf{y})\pi(\mathbf{x}_{i+1},\mathbf{y}_{i+1})}{\sum_{\mathbf{y}' \in \tilde{\mathcal{Y}}(\mathbf{x}_{i+1})} w(\mathbf{x},\mathbf{y}')\pi(\mathbf{x}_{i+1},\mathbf{y}')}$
    **end for**
    **for** $(\mathbf{x},\mathbf{y}) \in \tilde{\mathcal{Y}}(\mathbf{x}_{i+1})$ **do** ▷ Filtering
      $\pi(\mathbf{x}, \mathbf{y}) \leftarrow \lfloor K w(\mathbf{x}, \mathbf{y}) \rceil$
    **end for**
    $\tilde{\mathcal{Y}}(\mathbf{x}_{i+1}) \leftarrow \{(\mathbf{x}, \mathbf{y}) | \pi(\mathbf{x}, \mathbf{y}) > 0, (\mathbf{x}, \mathbf{y}) \in \tilde{\mathcal{Y}}(\mathbf{x}_{i+1})\}$
  **end for**
  **return** $\tilde{\mathcal{Y}}(\mathbf{x}_n)$
**end function**

Figure 2: Variable beam search pseudo-code

The naive computation of marginal probabilities stated in (5) requires to process a set $\mathcal{Y}(\mathbf{x}_i)$ growing exponentially in size with the length of the input sequence. Such a computation can however be approximated by Monte-Carlo simulation:

$$P(\mathbf{x}_i) \approx \lim_{K \to \infty} \frac{1}{K} \sum_{j=1}^{K} w(\mathbf{x}_i, \mathbf{y}_i^j) \tag{7}$$

This involves sampling derivations $\mathbf{y}_i^1 \dots \mathbf{y}_i^K$ from $P^*(\mathbf{x}, \mathbf{y})$, and is justified by observing that the marginalisation of equation (3) actually states an expectation:

$$P(\mathbf{x}_i) = \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{Y}(\mathbf{x}_i)} w(\mathbf{x}, \mathbf{y}) P^*(\mathbf{x}, \mathbf{y}) \tag{8}$$

which is precisely the one computed by (7). As our exploration method does not directly samples $K$ individual derivations but clusters the $K$ particles on a smaller number of derivations with the recurrence (1). We reformulate (8) not directly as (7) but rather as:

$$
\begin{aligned}
P(\mathbf{x}_i) &= \sum_{(\mathbf{x},\mathbf{y})\in\tilde{\mathcal{Y}}(\mathbf{x}_i)} w(\mathbf{x},\mathbf{y})P^*(\mathbf{x},\mathbf{y}) \\
&= \sum_{(\mathbf{x},\mathbf{y})\in\tilde{\mathcal{Y}}(\mathbf{x}_i)} P(\mathbf{x},\mathbf{y})\frac{P^*(\mathbf{x},\mathbf{y})}{P^*(\mathbf{x},\mathbf{y})} \\
&= \sum_{(\mathbf{x},\mathbf{y})\in\tilde{\mathcal{Y}}(\mathbf{x}_i)} P(\mathbf{x},\mathbf{y})
\end{aligned}
$$

That is we approximate the sum on the full set of parses $\mathcal{Y}(\mathbf{x}_i)$ by the sum of the set of parses found in the beam $\tilde{\mathcal{Y}}(\mathbf{x}_i)$ and sampled with the importance distribution $P^*(\mathbf{x},\mathbf{y})$.

**Parsing problem** Actual parsing is performed by collecting all successful derivations $\text{succ}(\mathbf{x})$ found during the search process and by returning the best scoring derivation:

$$
(\mathbf{x},\hat{\mathbf{y}}) = \underset{(\mathbf{x},\mathbf{y})\in\text{succ}(\mathbf{x})}{\text{argmax}} P(\mathbf{x},\mathbf{y})
$$

### 2.3 Measures extracted from the parser

We now define several measures extracted from the parsing process that might be relevant for fMRI modelling.

The first is the **beam size**, that is the size of the beam at word $x_i$ and it is directly measured as $B = |\tilde{\mathcal{Y}}(\mathbf{x}_i)|$

Next we define the **beam successful activity**. Let the set of transitions of a derivation be: $T(\mathbf{y}) = T(a_1 \dots a_m) = \{(a_1,a_2)\dots(a_{m-1},a_m)\}$ and the set of transitions $\mathbf{S}(\mathbf{x}_i) = \bigcup_{\mathbf{y}\in\tilde{\mathcal{Y}}(\mathbf{x}_i)} T(\mathbf{y})$ be the set of transitions leading successfully to $x_i$.

Then, the beam successful activity at $x_i$ is defined as the size of the set of transitions within $x_i$ and $x_{i-1}$, that is :

$$
s(x_i) = |\,\mathbf{S}(\mathbf{x}_i) - [\mathbf{S}(\mathbf{x}_i) \cap \mathbf{S}(\mathbf{x}_{i-1})]\,|
$$

Intuitively, the beam successful activity metric measures the amount of nodes depicted in blue in Figure 1.

A dead-end is a derivation $\mathbf{y}$ such that $\pi(\mathbf{y}) > 0$ and $\pi(\mathbf{y}a) = 0 \quad (\forall a \in A)$. Let $\mathcal{D}(\mathbf{x}_i)$ be the set of such dead-ends between $x_{i-1}$ and $x_i$ and the set of transitions $\mathbf{D} = \bigcup_{\mathbf{y}\in\tilde{\mathcal{D}}(\mathbf{x}_i)} T(\mathbf{y})$, then we define the **beam unsuccessful activity** as:

$$
u(x_i) = |\,\mathbf{D}(\mathbf{x}_i) - [\mathbf{D}(\mathbf{x}_i) \cap \mathbf{S}(\mathbf{x}_{i-1})]\,|
$$

Intuitively, the beam unsuccessful activity metric measures the amount of nodes depicted in red in Figure 1. Finally, let $\mathbf{A}(\mathbf{x}_i) = \mathbf{S}(\mathbf{x}_i) \cup \mathbf{D}(\mathbf{x}_i)$, then we define an overall **beam activity** at $x_i$ to be:

$$
o(x_i) = |\,\mathbf{A}(\mathbf{x}_i) - [\mathbf{A}(\mathbf{x}_i) \cap \mathbf{S}(\mathbf{x}_{i-1})]\,|
$$

As our model is generative, information theoretic metrics, such as surprisal and entropy, that are crucially dependant of the computation of lexical transition probabilities of the form $P(x_i|x_1\dots x_{i-1})$ can be computed too. **Surprisal** is a classic complexity measure (Hale, 2016) defined as:

$$
\textbf{surprisal}(x_i) = -\log_2 P(x_i|x_1\dots x_{i-1})
$$

and that is computed with equation (6).

For **entropy**, let $H[\mathcal{Y}(\mathbf{x}_i)]$ be the entropy of the set of prefix derivations at word $i$ that is defined as:

$$
H[\mathcal{Y}(\mathbf{x}_i)] = -\sum_{\mathbf{y}\in\mathcal{Y}(\mathbf{x}_i)} \frac{P(\mathbf{x}_i,\mathbf{y})}{P(\mathbf{x}_i)} \log_2\left(\frac{P(\mathbf{x}_i,\mathbf{y})}{P(\mathbf{x}_i)}\right)
$$

However in the context of a beam with variable size, these entropy measures cannot be directly compared at different time steps $i, i', i'' \dots$ because the size of the set $\mathcal{Y}(\mathbf{x}_i)$ is variable accross time steps. Therefore we use a **normalized entropy** whose values range in the interval $[0,1]$. As the maximum of the entropy measured on a set of $n$ elements is $\log(n)$, we use the normalized entropy measure:

$$
H_n[\mathcal{Y}(\mathbf{x}_i)] = \frac{H[\mathcal{Y}(\mathbf{x}_i)]}{\log_2(B)}
$$

To conclude this section, we introduced a search method for generative neural parsing that aims to overcome the problems of lexical biases (Stern et al., 2017) by drawing inspiration from particle filtering methods. In section 3 we motivate and study some properties of the method from a computational perspective and we explore in section 5 the potential of the overall beam activity predictors for modelling and analyzing fMRI data (Table 3).

## 3 Computational experiments

We report a few experiments designed to better understand the computational properties of the search method described so far. All the experiments are performed on the Penn-Treebank (Marcus et al., 1994), using sections 2-21 as training, section 22 as development and section 23 as test.

We preprocess the data for numbers replaced by a unique `num` token and we replace word occurrences with frequency one by the token `unk`.

Our development experiments compare our variable-size beam method with more traditional beam search. As standard naive beam search is not appropriate to generative parsing, we directly compare with the word-synchronous beam search method of Stern et al. (2017), applied to RNNG, with the so-called fast track extension. Our implementation of word synchronous beam search follows closely that of Hale et al. (2018). We select the settings found in the literature: for a beam of size $B = k$, a lexical beam of size $k/10$ and a fast-track of size $k/100$.

| MODEL | F-SCORE | PPL | BEAM ACTIVITY |
|---|---|---|---|
| K=1000-base | 86.29 | 107.89 | 153.17 |
| K=1000 | 90.75 | 86.00 | 66.3 |
| K=10000 | 91.12 | 84.25 | 153.6 |
| K=50000 | **91.26** | 83.50 | 380.3 |
| B=100 | 87.21 | 95.97 | 416.9 |
| B=400 | 90.50 | **82.39** | 1775.1 |

Table 1: Development scores.

By observing the development measures in Table 1, the first thing to note is the lack of accurracy of the vanilla application of our *base* method, as can be seen on the first line of the Table, when parsing with $K = 1000$ particles for searching. This *base* model uses straightforwardly the reweighting step stated in (4). We observed however that the low accurracy of this model is a consequence of a rounding issue when using discretized particle counts. To avoid this problem we used the alternative reweighting scheme:

$$w'(\mathbf{x}_{i+1}, \mathbf{y}_{i+1}) = \frac{P(\mathbf{x}_{i+1}, \mathbf{y}_{i+1})}{\sum_{\mathbf{y}' \in \tilde{\mathcal{Y}}(\mathbf{x}_{i+1})} P(\mathbf{x}_{i+1}, \mathbf{y}')}$$

that avoids this rounding problem and all the other results reported in Table 1 use instead this alternative scheme.

From the development set we can also point out that our beam baselines are almost identical to those of Hale et al. (2018) on beams $B = 100$ and $B = 400$. Note that the authors managed to reach an F-score of 91.2 on the development with $B = 2000$. By comparison, our particle method tends to produce high F-scores even with limited $K$. Although the perplexity of the beam method remains generally lower, as shown in Table 1.

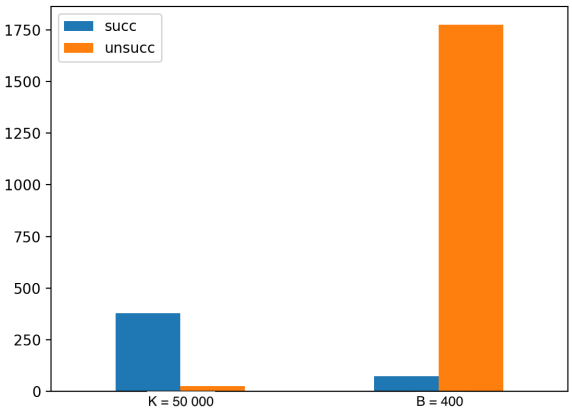


Figure 3: Successful and unsuccessful beam activities.

The main difference comes from computation time that we measure with an averaged beam activity metric defined earlier on: as we can see our most expensive model $K = 50000$ requires to perform less computations than the smallest beam tested. This striking difference can be further explained by comparing the successful and unsuccessful beam activities.

Figure 3 plots those activities for models $B = 400$ and $K = 50000$. The more traditional beam search spends most of its time performing unsuccessful computations while our method spends most of its time performing successful computations. All these observations have practical consequences: the beam method takes hours to complete the parsing of the Wall Street Journal (wsj, see Table 2) development while the particle search takes minutes on a standard workstation.

| MODEL | F-SCORE | PPL (wsj) | PPL (prince) |
|---|---|---|---|
| K=50000 | **91.02** | 94.35 | 154.93 |
| B=400 | 90.15 | **93.02** | 139.76 |
| IKN5 | - | 155.02 | 309.54 |
| LSTM-LM | - | 141.28 | 204.06 |
| (Dyer et al. 2016) | 93.3 | 105.2* | unknown |
| (Fried et al. 2017a) | 92.56 | unknown | unknown |
| (Kitaev et al. 2018) | **93.55** | - | - |

∗A different preprocessing is used. The comparison remains indicative

Table 2: Test results.

For the test, we observe again from Table 2 that our search method ($K = 50000$) gets better F-score and worse perplexity than the beam method.

The most important observation comes from the comparison between the language model perplexities and parsing perplexities. On a small data set such as the Penn Treebank, the parser has a much better perplexity than traditional 5-grams Interpolated Kneser Ney (IKN5) or vanilla LSTM-LM with a comparable configuration to that of the parser. It is quite clear that state of the art language models outperform the parsing language model by a large margin, at the cost of much larger training sets (Radford et al., 2019). The perplexities reported as *Prince* in Table 2 are the perplexities resulting from processing the *Little Prince* corpus used later in the paper for analyzing neuro-imaging data. The table allows to measure the amount of drop in perplexity that can be explained by corpus domain change.

We also compare with parsers without strict incrementality restrictions: Dyer et al. (2016) is an RNNG as a reranker with a discriminative first step (that is with lookahead). Stern et al. (2017) are the results reported with a beam of size 2000 using the generative model described by Choe and Charniak (2016) with a substantially different preprocessing of the data than the one used with RNNG (Hale et al., 2018). Finally, Kitaev and Klein (2018) is the current state of the art single parser with a discriminative model.

## 4 Neuro-imaging and Parsing Models

A number of studies observed a positive correlation between behavioural and psycho-linguistic data like reading times and entropy measures derived from probabilistic parsers' computations (Levy, 2008). Eye-movements (Demberg and Keller, 2008) and Event-Related Potentials (ERPs, Frank et al. (2015) were recorded during reading performance or sentence completion tasks have allegedly contributed in giving a cognitive dimension to the Surprisal Theory (Levy, 2008).

**Parsers Actions and Neuro-imaging** Further empirical proofs of the link between information-theoretic measures and cognitive processes showed a correlation between the *output* or internal state of syntactic parsers and cerebral activity. In a seminal work based on a 30 minutes recording from the English text *Alice in wonderland*, Brennan and colleagues (2012; 2016) modeled brain activity with a simple word-by-word measure of Node counts, featuring an estimate of the amount syntactic structure analyzed so far, as the number of phrases closed at each word. This computationally assessed phrase-structure building process showed to involve Inferior Frontal Gyrus and Anterior Temporal regions, which was found consistent with earlier studies on sentence structure building (Pallier et al., 2011; Snijders et al., 2009).

**Parsing strategy in Neuro-imaging** Other fMRI studies started to use another computationally derived complexity metric to quantify structure-building and processing effort in the brain. The number of rules applied during the parser's computational procedure, between each word of a sentence, was taken to define an incremental index of computational syntactic "work"(e.g. Bhattasali et al. 2019).

A recent intracranial recording work correlated brain activity with computational metrics from alternative grammar types (e.g. context-free, Minimalist) and different parsing strategies (e.g. left-corner, top-down, bottom-up) (Nelson et al., 2017). Phrase-structure appeared, like in more traditional approaches (Friederici and Gierhan, 2013; Snijders et al., 2009; Hickok and Poeppel, 2007), as a major determinant of the dynamic profile of brain activity in language areas. Superior temporal and inferior frontal areas provided a good fit of the brain-activity data for bottom-up and left-corner parsing strategies, in left superior temporal, inferior frontal, dorsal and midline frontal sites.

Based on this set of empirical evidence, showing that syntactic structure-building implicates frontal regions, such as the Pars Triangularis and Pars Opercularis of the Inferior Frontal Gyrus (IFG-Broca) and anterior temporal areas, neuro-imaging signals can be used nowadays to adjudicate between competing parsing mechanisms. The present paper, goes a step further, using the beam internal state in two different Search methods in a generative RNNG architecture, to shed light on the potential benefits of beam-size variation for the cognitive modelling of sentence processing.

## 5 fMRI: Method and Results

The analyzed text was the transcribed version of the English audio-book of Antoine de Saint-Exupéry *The Little Prince*, translated by David Wilkinson and read by Nadine Eckert-Boulet. This text comprises 19,171 tokens and 15,388 words, grouped in 1388 sentences. The imaging data-set analysed in this paper comprises 50

| Predictors | Description |
|---|---|
| Overall Beam activity K50000 or B400 | word-by-word measure of beam activity (§2.3) |
| Word rate | tagging spoken words on the fMRI signal |
| Word frequency | word-by-word log-frequency in movie subtitles |
| RMS amplitude | an acoustic correlate of volume every 10ms |

Table 3: Predictors used in the fMRI Analysis. RMS (Root Mean Square) encodes the amplitude of spoken narration, it reflects the intensity.

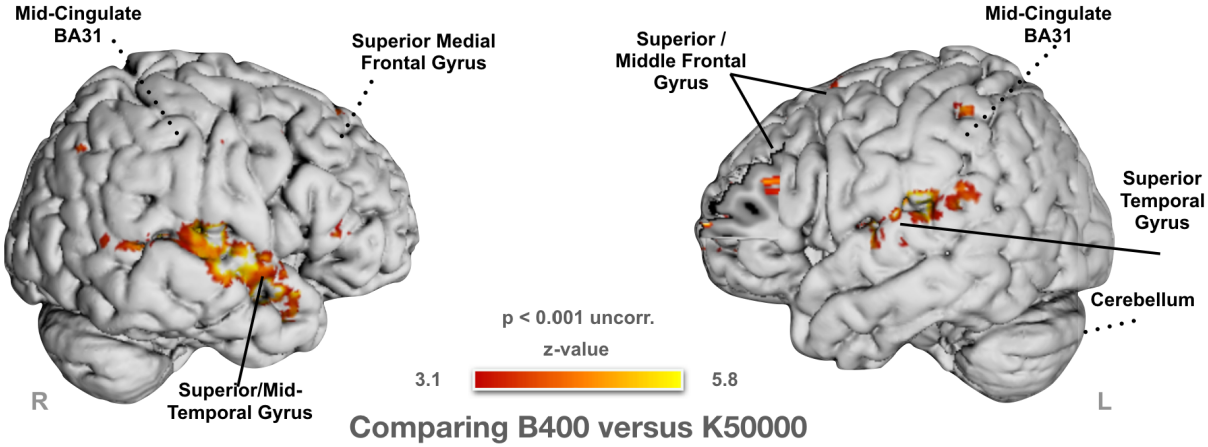


Figure 4: Brain z-maps showing the significant clusters (K>25) for the model comparison between Overall Beam Activity calculated in the fix-beam B400 versus variable-beam K50000 RNNG parser, p < .001 uncorr.

right-handed native English participants who listened to the entire audio-book during 1h38 minutes. Imaging was performed in a 3 Tesla MRI scanner (Discovery MR750 GE, Milwaukee, WI) with a 32-channel head coil at the Cornell MRI Facility as part of a naturalistic listening neuro-imaging project. Details of preprocessing are presented in the Supplementary Materials.

### 5.1 Statistical Analysis

**$r^2$ Model comparison - General Linear Model** The research questions presented above in section 2.2 motivates a statistical analysis that performs a comparison where fMRI signal is modeled (GLM) by B400 fix beam parser versus the K50000 parsing system adopting a variable beam.

**Single-subject statistics** At the single subject level, the observed time-course of the brain hemodynamic response (BOLD - Blood Oxygenation Level Dependent) in each voxel was modeled by the Overall Beam Activity measure (cf. 2.3) calculated for B400 and K50000 parsers, and time-locked at the offset of each word in the audio-book. Model comparisons using cross-validated coefficient of determination (r2) maps was carried out in order to evaluate the goodness of fit of the two beam search methods with BOLD signal. The predictors shown in Table 3 were convolved using SPM's (Friston et al., 2007) canonical HRF (Hemodynamic Response Function). The two neuro-imaging models (i.e. K50000 and B400) also included in the GLM three control variables: *Root Mean Square intensity* (RMS) an indicator of intensity at every 10 ms of the audio; *word rate* as a stick function marking the offset of each spoken word; *frequency* of the individual words in movie subtitles from (Brysbaert and New, 2009).

Such predictors are here to ensure that conclusions about parsing difficulty would be specific to the processes they instantiate, as opposed to more general aspects of speech perception. Specifically, lexical frequency was added as a covariate of non-interest, to statistically factor out effects of general word frequency (generally used in psycholinguistics), that may correlate with other types of expectations. For every subject, we compute how much the inclusion of each variable of interest (i.e. B400 and K50000) increases the cross-validated $r^2$, from a baseline model that does not include them, see Table 3). Hence, the $r^2$ scores represent the variance explained in each voxel by the variable instantiating the two beam-search methods.

| Regions for K50000 vs B400 | Cluster size (in voxels) | MNI Coordinates x | y | z | p-value (uncorrected) | z-score (peak) |
|---|---|---|---|---|---|---|
| R Superior / Mid-Temporal Gyrus | 2307 | 54 | -8 | -2 | 0.000 | 5.79 |
| L Superior Temporal Gyrus / Insula | 1442 | -52 | -12 | 2 | 0.000 | 5.73 |
| R Superior Medial Frontal Gyrus (BA9/10) | 114 | 4 | 56 | 24 | 0.000 | 4.89 |
| R Mid-Cingulate Gyrus (BA31) | 95 | 4 | -42 | 40 | 0.000 | 4.30 |
| L Middle Frontal Gyrus | 58 | -40 | 36 | 22 | 0.000 | 4.13 |
| L Cerebellum - Crus I/II | 34 | -22 | -74 | -36 | 0.000 | 4.02 |
| L Superior Frontal Gyrus (10) | 34 | -28 | 58 | 2 | 0.001 | 3.64 |
| L Mid-Cingulate Gyrus (BA31) | 95 | -6 | -28 | 44 | 0.001 | 3.66 |

Table 4: Clusters showing a significant better fit for B400 fix beam-size search method in the RNNG parser, p< 0.001 (z-score $>$ 3.1) uncorrected at k$>$ 25 cluster threshold.

**Group-level statistics** To compare the impact of beam-search on fMRI signal explanation (i.e. $r^2$ increase of each variable) we performed a paired t-test on each individual $r^2$ map, and obtained Figure 4 showing where one variable explains significantly better the signal than the other (cf. Tab. 4).

### 5.2 Results - Fit with fMRI signal

We performed an $r^2$ comparison to test which beam search method provided the better fit to the fMRI data recorded during *The Little Prince*. The two different search methods were tested (K50000 variable beam and B400 fix beam), and B400 was shown to be the best fitting the BOLD signal of these models. Figure 4 (clusters coordinates and statistics, cf. Table 4), shows the significance (z-scores, $p < 0.001$ uncorrected) of the difference in $r^2$ scores with a cluster threshold of 25 voxels. Considering the cluster with a larger extent, we might also have captured some acoustic parameter that was not included in the GLM model, although further analyses showed that adding a prosodic predictor does not have a major impact.

Of the two parsers' search methods, the fix beam one had a significant predictive value in well-known language areas, namely temporal areas and sub-parts core frontal regions. Our particle filtering based method is used to maintain the beam and limit its size, while experiments show that the proposed method achieves higher parsing accuracy than a fixed-size beam search baseline, the fixed-size beam search is shown here to be a better model than the proposed method in predicting brain activities. This result could be related to its tendency to keep ambiguities longer along the sentence, and thus model the computational load of a larger spectrum of hypotheses without pruning immediately the hypothesis space (cf. Fig. 3).

## 6 Discussion & Conclusion

The paper investigates the relevance of a sequential Monte-Carlo inspired search method for generative parsing. The proposed search method is in principle very general although we tested its relevance to overcome lexical biases inherent to direct generative parsing for RNNG. We found out that the method's main benefit is an increased processing efficiency. The measures instantiating variable-beam and fixed-beam search were used to quantify the amount of structure-building "work" the parser performs in the course of word-by-word processing of a given sentence. Our current neuro-imaging results suggest that the fixed-size beam activity is a better predictor of brain activity than the variable-size beam activity. The original methodology presented in this paper paves the way for further computational work in quantifying parsing complexity and thus fine-grained modelling of human sentence processing.

## Acknowledgments

We are indebted to Wenming Luh, Nathan Spreng and John Hale for collaborating and providing fMRI data. We thank Shohini Bhattasali and Jixing Wang for assistance in data collection and preprocessing. We also thank the anonymous reviewers for their detailed comments on the original manuscript. This work is supported by the French National Research Agency (ANR) under grant ANR-14-CERA-0001, we gratefully acknowledge the National Science Foundation under Grant No.1607441 for data collection.

## References


Shohini Bhattasali, Murielle Fabre, Wen-Ming Luh, Hazem Al Saied, Mathieu Constant, Christophe Pallier, Jonathan R. Brennan, R. Nathan Spreng, and John Hale. 2019. Localising memory retrieval and syntactic composition: an fmri study of naturalistic language comprehension. *Language, Cognition and Neuroscience*, 34(4):491–510.

Jonathan Brennan. 2016. Naturalistic sentence comprehension in the brain. *Language and Linguistics Compass*, 10(7):299–313.

Jonathan Brennan, Yuval Nir, Uri Hasson, Rafael Malach, David J Heeger, and Liina Pylkkänen. 2012. Syntactic structure building in the anterior temporal lobe during natural story listening. *Brain and Language*, 120(2):163–173.

Marc Brysbaert and Boris New. 2009. Moving beyond Kučera and Francis: A critical evaluation of current word frequency norms and the introduction of a new and improved word frequency measure for American English. *Behavior research methods*, 41(4):977–990.

Jan Buys and Phil Blunsom. 2015. Generative incremental dependency parsing with neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 863–869.

Do Kook Choe and Eugene Charniak. 2016. Parsing as language modeling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2331–2336.

Vera Demberg and Frank Keller. 2008. Data from eye-tracking corpora as evidence for theories of syntactic processing complexity. *Cognition*, 109(2):193 – 210.

Arnaud Doucet and Adam M Johansen. 2011. *A tutorial on particle filtering and smoothing: Fifteen years later*, d. crisan and b. rozovsk, eds. edition, volume 12 (3) of *Handbook on Nonlinear Filtering*, pages 656–704. Oxford Press.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A Smith. 2016. Recurrent neural network grammars.

Stefan L. Frank, Leun J. Otten, Giulia Galli, and Gabriella Vigliocco. 2015. The erp response to the amount of information conveyed by words in sentences. *Brain and Language*, 140:1 – 11.

Angela D Friederici and Sarah ME Gierhan. 2013. The language network. *Current Opinion in Neurobiology*, 23(2):250–254.

K.J. Friston, J. Ashburner, S.J. Kiebel, T.E. Nichols, and W.D. Penny, editors. 2007. *Statistical Parametric Mapping: The Analysis of Functional Brain Images*. Academic Press.

John Hale. 2016. Information-theoretical complexity metrics: Information-theoretical complexity metrics. *Language and Linguistics Compass*, 10(9):397–412.

John Hale, Chris Dyer, Adhiguna Kuncoro, and Jonathan Brennan. 2018. Finding syntax in human encephalography with beam search. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 2726–2735.

Gregory Hickok and David Poeppel. 2007. The cortical organization of speech processing. *Nature Reviews Neuroscience*, 8(5):393–402.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 2675–2685.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A Smith. 2017. What do recurrent neural network grammars learn about syntax?

Roger Levy. 2008. Expectation-based syntactic comprehension. *Cognition*, 106(3):11261177.

Jiangming Liu and Yue Zhang. 2017. In-order transition-based constituent parsing. volume 5, pages 413–424.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The penn treebank: Annotating predicate argument structure. In *Proceedings of the Workshop on Human Language Technology*, HLT '94, pages 114–119, Stroudsburg, PA, USA. Association for Computational Linguistics.

Matthew J. Nelson, Imen El Karoui, Kristof Giber, Xiaofang Yang, Laurent Cohen, Hilda Koopman, Sydney S. Cash, Lionel Naccache, John T. Hale, Christophe Pallier, and Stanislas Dehaene. 2017. Neurophysiological dynamics of phrase-structure building during sentence processing. *Proceedings of the National Academy of Sciences*, 114(18):E3669–E3678.

Christophe Pallier, Anne-Dominique Devauchelle, and Stanislas Dehaene. 2011. Cortical representation of the constituent structure of sentences. *Proceedings of the National Academy of Sciences*, 108(6):2522–2527.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. Unpublished manuscript.

Tineke M Snijders, Theo Vosse, Gerard Kempen, Jos JA Van Berkum, Karl Magnus Petersson, and Peter Hagoort. 2009. Retrieval and unification of syntactic structure in sentence comprehension: An fMRI study using word-category ambiguity. *Cerebral Cortex*, 19(7):1493–1503.

Mitchell Stern, Daniel Fried, and Dan Klein. 2017. Effective inference for generative neural parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 1695–1700.

# Are We Modeling the Task or the Annotator? An Investigation of Annotator Bias in Natural Language Understanding Datasets

**Mor Geva**
Tel Aviv University,
Allen Institute for AI
morgeva@mail.tau.ac.il

**Yoav Goldberg**
Bar-Ilan University,
Allen Institute for AI
yoav.goldberg@gmail.com

**Jonathan Berant**
Tel Aviv University,
Allen Institute for AI
joberant@cs.tau.ac.il

## Abstract

Crowdsourcing has been the prevalent paradigm for creating natural language understanding datasets in recent years. A common crowdsourcing practice is to recruit a small number of high-quality workers, and have them massively generate examples. Having only a few workers generate the majority of examples raises concerns about data diversity, especially when workers freely generate sentences. In this paper, we perform a series of experiments showing these concerns are evident in three recent NLP datasets. We show that model performance improves when training with annotator identifiers as features, and that models are able to recognize the most productive annotators. Moreover, we show that often models do not generalize well to examples from annotators that did not contribute to the training set. Our findings suggest that annotator bias should be monitored during dataset creation, and that test set annotators should be disjoint from training set annotators.

## 1 Introduction

Generating large datasets has become one of the main drivers of progress in natural language understanding (NLU). The prevalent method for creating new datasets is through crowdsourcing, where examples are generated by workers (Zaidan and Callison-Burch, 2011; Richardson et al., 2013; Bowman et al., 2015; Rajpurkar et al., 2016; Trischler et al., 2017). A common recent practice is to choose a small group of workers who produce high-quality annotations, and massively generate examples using these workers.

Having only a few workers annotate the majority of dataset examples raises concerns about data diversity and the ability of models to generalize, especially when the crowdsourcing task is to generate free text. If an annotator consistently uses language patterns that correlate with the labels, a neural model can pick up on those, which can lead to an over-estimation of model performance.

In this paper, we continue recent efforts to understand biases that are introduced during the process of data creation (Levy et al., 2015; Schwartz et al., 2017; Gururangan et al., 2018; Glockner et al., 2018; Poliak et al., 2018; Tsuchiya, 2018; Aharoni and Goldberg, 2018; Paun et al., 2018). We investigate this form of bias, termed *annotator bias*, and perform multiple experiments over three recent NLU datasets: MNLI (Williams et al., 2018), OpenBookQA. (Mihaylov et al., 2018), and CommonsenseQA (Talmor et al., 2019).

First, we establish that annotator information improves model performance by supplying annotator IDs as part of the input features. Second, we show that models are able to recognize annotators that generated many examples, illustrating that annotator information is captured by the model. Last, we test whether models generalize to annotators that were not seen at training time. We observe that often generalization to new annotators fails, and that augmenting the training set with a small number of examples from these annotators substantially increases performance.

Taken together, our experiments show that annotator bias exists in current NLU datasets, which can lead to problems in model generalization to new users. Hence, we propose that annotator bias should be monitored at data collection time and to tackle it by having the test set include examples from a disjoint set of annotators.

## 2 Crowdsourcing Practice

Crowdsourcing has become the prominent paradigm for creating NLP datasets (Callison-Burch et al., 2015; Sabou et al., 2014). It has been used for various NLU tasks, including Question Answering (Rajpurkar et al., 2018; Mihaylov



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1161–1166,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

et al., 2018; Dua et al., 2019), commonsense and visual reasoning (Talmor et al., 2019; Zellers et al., 2019; Suhr et al., 2018), and Natural Language Inference (NLI) (Williams et al., 2018).

In a typical process, annotators are recruited and screened (Sabou et al., 2014), often resulting in a small group that creates most of the dataset examples. Mihaylov et al. (2018) recruited a few dozens of qualified workers that wrote 5,957 questions. Suhr et al. (2018) recruited 99 workers for a sentence writing task, who created more than 100,000 examples. Williams et al. (2018) recruited 387 workers for writing over 400,000 sentences, while Krishna et al. (2017) had 33,000 workers contributing 1.7 million examples.

These examples demonstrate that datasets are often constructed using a small number of annotators, approximately 1 annotator per $10^2$–$10^3$ examples. Furthermore, (see Section 3), the annotator distribution is skewed with a few annotators creating the vast majority of the dataset. In tasks that involve creative language writing, this may have implications on data diversity, and lead to an over-estimation of model performance.

## 3 Experimental Setup

We focus on crowdsourcing tasks where workers produce full-length sentences. We first describe the datasets we test our hypothesis on, and then provide details on the model and training.

**Datasets** We consider recent NLU datasets, for which the annotator IDs are available.

- **MNLI** (matched) (Williams et al., 2018): A NLI dataset. Each example was created by introducing an annotator with a premise sentence and asking her to write a hypothesis sentence that is either entailed, contradicted or is neutral with respect to the premise.
- **OpenBookQA** (Mihaylov et al., 2018): A multiple-choice question answering dataset, focusing on multi-hop reasoning. Each question and its answer distractors were written by a worker, based on a given scientific fact.
- **CommonsenseQA** (Talmor et al., 2019): A multiple-choice question answering dataset, focused on commonsense knowledge. Questions were written by crowdworkers, who try to bridge between two concepts extracted from ConceptNet (Speer et al., 2017)

| | # examples | # annotators |
|---|---|---|
| MNLI (matched) | 402,517 | 380 |
| OpenBookQA | 5,457 | 84 |
| CommonsenseQA | 11,096 | 132 |

Table 1: Statistics for datasets used in our experiments.

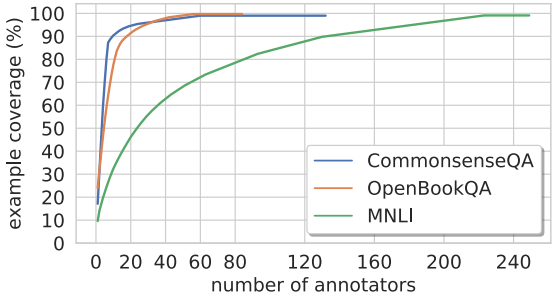


Figure 1: Proportion of examples covered by number of annotators (sorted by number of annotations).

Table 1 summarizes the size and number of annotators who worked on each dataset. Figure 1 shows the fraction of examples covered by the number of annotators, sorted by the number of examples they annotated. In all datasets, and specifically in OpenBookQA and CommonsenseQA, most of the examples were generated by a small number of annotators.

**Model** We use the pretrained BERT-base (Devlin et al., 2018), a strong model obtaining close to state-of-the-art performance on all three datasets. We add a single linear layer over BERT outputs and apply the same fine-tuning procedure in all experiments: fine-tuning for 3 epochs, using batch size 10, learning rate $2 \times 10^{-5}$, and maximum sequence length of 128 word pieces.

## 4 Experiments and Results

We now describe a series of experiments for quantifying annotator bias, aiming to answer the following questions: 1) Do models perform better when exposed to the annotator ID? 2) Can models detect the annotators from their generated examples? 3) Do models generalize across annotators?

**The utility of annotator information** Our first experiment aims to determine whether models perform better given *perfect* information on the annotator ID. To this end, we follow the standard way of feeding input to BERT (Devlin et al., 2018) and concatenate the annotator ID as an additional feature to every example in every dataset. Formally, we replace every example ($x =$

$(w_1, ..., w_{|x|}), y)$ created by annotator $z$, with the example $((z, w_1, ..., w_{|x|}), y)$, where $z$ is a textual unique annotator identifier, $x$ is the input sequence and $y$ is the gold label.

We compare performance on the original datasets and their new version. Adding the annotator ID improves model performance across all datasets (Table 2), showing that perfect annotator information is useful for prediction, and there is incentive for the model to capture this information.

| | Without ID | With ID | $p$-value |
|---|---|---|---|
| OPENBOOKQA | 52.2 | 56.4 | $1.83e^{-2}$ |
| COMMONSENSEQA | 53.6 | 55.3 | $11.98e^{-2}$ |
| MNLI | 82.9 | 84.5 | $5.13e^{-7}$ |

Table 2: Model development performance, after training with/without annotator IDs as additional inputs. Following Dror et al. (2018), $p$-values were calculated using the McNemar's test (McNemar, 1947) for MNLI and the Bootstrap test (Berg-Kirkpatrick et al., 2012) for OPENBOOKQA and COMMONSENSEQA.

**Annotator recognition** Perfect annotator information improves performance, but it is possible that a model can recognize the annotators from the input sequence only, even without being exposed to the annotator ID explicitly. In the next experiment, we investigate the ability of models to recognize the annotators from the input.

To this end, we fine-tune BERT-base to predict annotator IDs from examples. We limit the task to 6 labels of the top-5 most productive annotators of each dataset and an OTHER label for all other annotators. Formally, we replace every example $(x, y)$ created by annotator $z$, with the example $(x, \bar{z})$, where $\bar{z} = z$ if $z$ is in the top-5 annotators and $\bar{z} =$ OTHER otherwise.

Figure 2 shows the F1-score for the top-5 annotators of every dataset (y-axis), and the fraction of dataset examples created by each annotator (x-axis). Overall, annotators who write many examples are recognized better by the model: The model struggles to recognize MNLI annotators with F1 scores below 0.5, and excels at recognizing annotators from COMMONSENSEQA with scores between 0.76–0.91. For the top annotator of OPENBOOKQA, who created 24% of the dataset examples, the model obtains a high F1 score of 0.8. To conclude the first two experiments, annotator ID information is useful for the downstream task, and also can be predicted with high probability from the input for a large fraction of the examples.

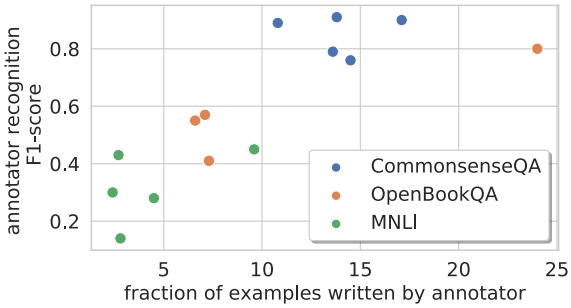


Figure 2: Annotator recognition F1-scores for the top-5 annotators of each dataset. For OPENBOOKQA only 4 data points are plotted, as the second annotator is not in the original development set.

**Generalization across annotators** In our final experiment, we examine whether trained models generalize to new annotators. To address this question, we re-split each dataset, creating a training and development set with disjoint annotators: Given a dataset with example set $\mathcal{S}$, we denote by $\mathcal{S}_z \subset \mathcal{S}$ the subset of examples created by annotator $z$. Similarly, for a set of annotators $\mathcal{Z}$, let $\mathcal{S}_\mathcal{Z} = \bigcup_{z \in \mathcal{Z}} \mathcal{S}_z$. We rank annotators by the number of examples they generated, and for each dataset $\mathcal{S}$, we create two types of data splits. For each annotator $z$ in the top-5 annotators, we create a *single-annotator* data split with $\mathcal{S} \backslash \mathcal{S}_z$ and $\mathcal{S}_z$ as the train and development sets, respectively. Namely, we consider the examples created by annotator $z$ as the development set, while using all other examples for training. In addition, we pick 5 sets of 5 annotators, who annotated a small number of examples, and for each such set $\mathcal{Z}$, we create a *multi-annotator* split with $\mathcal{S} \backslash \mathcal{S}_\mathcal{Z}$ and $\mathcal{S}_\mathcal{Z}$ as the train and development sets, respectively. Namely, we consider the examples created by the 5 annotators $\mathcal{Z}$ as the development set, while using all other examples for training. Overall, there are 5 single-annotator splits and 5 multi-annotator splits for each dataset.

We fine-tune BERT-base and evaluate it on the development set, and compare the results to a random data split of identical size. We repeat every experiment 3 times, except for multi-annotator experiments on OPENBOOKQA and COMMONSENSEQA which we repeat 9 times due to high variance. Table 3 shows the mean and standard deviation of performance difference (p.d.) between each annotator(s) split and its corresponding ran-

dom split, where negative numbers indicate that performance on the annotator split was lower.

Our clearest finding is that in OPENBOOKQA performance on the the multi-annotator split is dramatically lower than on a random split in all 5 annotator sets, where performance drops by up to 23 accuracy points. This shows that the model does not generalize to examples generated by unseen annotators. In the other datasets, results on the multi-annotator split are more varied, where performance drops in roughly half the cases, sometimes substantially – up to 10 accuracy points in COMMONSENSEQA and 5 in MNLI.

In the single-annotator splits, in roughly half the cases performance on the annotator split was lower than the random split. However, measuring p.d. only for single annotators might be misleading, because specific annotators vary in the difficulty of examples they produce. Thus, running a model on a new annotator that produces easy examples will not result in decreased performance. Next, we propose a more fine-grained experiment that controls for these two confounding factors.

| COMMONSENSEQA-single | | COMMONSENSEQA-multi | |
|---|---|---|---|
| $4.2 \pm 0.7$ | 17.1% | $-9.5 \pm 8.3$ | 0.9% |
| $7.7 \pm 1.9$ | 14.5% | $6.5 \pm 7.0$ | 0.6% |
| $-2.8 \pm 1.3$ | 13.8% | $-6.1 \pm 8.5$ | 0.5% |
| $-3.8 \pm 0.9$ | 13.6% | $1.6 \pm 10.8$ | 0.4% |
| $1.6 \pm 2.7$ | 10.8% | $1.8 \pm 10.5$ | 0.4% |
| OPENBOOKQA-single | | OPENBOOKQA-multi | |
| $-0.9 \pm 2.7$ | 24% | $-14.7 \pm 6.2$ | 2.4% |
| $-13.5 \pm 1.7$ | 7.8% | $-19.4 \pm 8.5$ | 1.7% |
| $-5.8 \pm 0.7$ | 7.3% | $-12.4 \pm 5.5$ | 1.2% |
| $8.2 \pm 5.2$ | 7.1% | $-13.7 \pm 8.5$ | 1% |
| $3.1 \pm 1.1$ | 6.6% | $-23.3 \pm 7.8$ | 0.8% |
| MNLI-single | | MNLI-multi | |
| $-2.5 \pm 0.5$ | 9.6% | $2.5 \pm 0.8$ | 1.8% |
| $-3.0 \pm 0.6$ | 4.5% | $-1.1 \pm 0.9$ | 1.5% |
| $2.9 \pm 0.2$ | 2.8% | $-4.6 \pm 0.8$ | 1.5% |
| $0.8 \pm 0.7$ | 2.7% | $-1.5 \pm 0.2$ | 1.5% |
| $4.6 \pm 0.2$ | 2.4% | $0.5 \pm 0.2$ | 1.5% |

Table 3: Performance difference (p.d.) between single- and multi- annotator splits and random splits of identical size. Each cell shows the p.d. mean and standard deviation, as well as the development set relative size.

**Separating annotator bias from annotator difficulty** To mitigate the effect of annotator difficulty, we perform an augmentation experiment. Assume a development set $\mathcal{S}_{\text{dev}}$, for which performance is low. Our hypothesis is that if $\mathcal{S}_{\text{dev}}$ is inherently difficult, then moving a small portion of examples from $\mathcal{S}_{\text{dev}}$ to the training set $\mathcal{S}_{\text{train}}$ should not change performance on $\mathcal{S}_{\text{dev}}$ substantially. However, if performance on $\mathcal{S}_{\text{dev}}$ is low due to annotator bias, then moving examples to $\mathcal{S}_{\text{train}}$ would expose the model to the annotator and performance should go up.

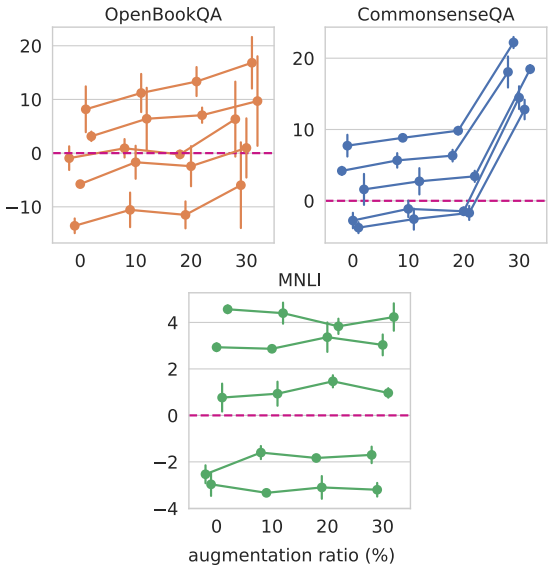


Figure 3: Performance difference between single-annotator splits and random splits of identical size. The x-axis indicates the fraction of examples taken from the development set to augment the training set.

For every single-annotator data split $\mathcal{S}_{\text{train}}$, $\mathcal{S}_{\text{dev}}$, we perform a series of augmentation experiments, where we move a random fraction of $k$ examples from $\mathcal{S}_{\text{dev}}$ to $\mathcal{S}_{\text{train}}$, for $k = 0.1, 0.2, 0.3$. We keep the size of $\mathcal{S}_{\text{train}}$ constant by randomly removing examples from it before augmentation. We repeat experiments multiple times as before, and report the p.d mean and standard deviation in Figure 3.

Results for both COMMONSENSEQA and OPENBOOKQA show a rapid increase of 10-20 accuracy point for all top-5 annotators, given only a small number of their generated examples. This shows that the examples generated by these annotators are not inherently difficult, and that the model can substantially improve performance by being exposed to the language that the annotators generate. Conversely, performance changes are marginal for MNLI, suggesting generalization patterns are mostly due to example difficulty. The different results for MNLI compared to those observed for OPENBOOKQA and COMMONSENSEQA may be attributed to the less-skewed annotator distribution and large number of examples in MNLI (see Figure 1 and Table 1), which make the model more robust to small perturbations in the data distribution.

## 5 Discussion and Conclusions

This study set out to investigate whether prevalent crowdsourcing practices for building NLU datasets introduce an annotator bias in the data that leads to an over-estimation of model performance. We established that perfect annotator information can improve model performance, and that the language generated by annotators often reveals their identity. Moreover, we tested the ability of models to generalize to unseen annotators in three recent NLU datasets, and found that in two of these datasets annotator bias is evident. These findings may be explained by the annotator distributions and the size of these datasets. Skewed annotator distributions with only a few annotators creating the vast majority of examples are more prone to biases.

Our results suggest that annotator bias should be monitored in crowdsourcing tasks involving free text generation by annotators. This can be done by testing model performance on new annotators during data collection. Moreover, to tackle annotator bias, we propose that training set annotators should be separated from test-set annotators.

## Acknowledgments

This research was partially supported by The Israel Science Foundation grant 942/16, The Blavatnik Computer Science Research Fund and The Yandex Initiative for Machine Learning. We thank Sam Bowman from NYU University and Alon Talmor from Tel Aviv University for providing us the annotation information of the MNLI and COMMONSENSEQA datasets. This work was completed in partial fulfillment for the Ph.D degree of the first author.

## References

Roee Aharoni and Yoav Goldberg. 2018. Split and rephrase: Better evaluation and stronger baselines. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 719–724. Association for Computational Linguistics.

Taylor Berg-Kirkpatrick, David Burkett, and Dan Klein. 2012. An empirical investigation of statistical significance in nlp. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 995–1005. Association for Computational Linguistics.

S. Bowman, G. Angeli, C. Potts, and C. D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Chris Callison-Burch, Lyle Ungar, and Ellie Pavlick. 2015. Crowdsourcing for NLP. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorial Abstracts*, pages 2–3.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. 2018. The hitchhikers guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1383–1392.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proc. of NAACL*.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking NLI systems with sentences that require simple lexical inferences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 650–655, Melbourne, Australia. Association for Computational Linguistics.

S. Gururangan, S. Swayamdipta, O. Levy, R. Schwartz, S. R. Bowman, and N. A. Smith. 2018. Annotation artifacts in natural language inference data. *arXiv preprint arXiv:1803.02324*.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. 2017. Visual Genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123(1):32–73.

Omer Levy, Steffen Remus, Chris Biemann, and Ido Dagan. 2015. Do supervised distributional methods really learn lexical inference relations? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 970–976.

Quinn McNemar. 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? A new dataset for open book question answering. In *In proceedings of the 2018 Conference*

*on Empirical Methods in Natural Language Processing (EMNLP)*.

Silviu Paun, Bob Carpenter, Jon Chamberlain, Dirk Hovy, Udo Kruschwitz, and Massimo Poesio. 2018. Comparing bayesian models of annotation. *Transactions of the Association for Computational Linguistics*, 6:571–585.

A. Poliak, J. Naradowsky, A. Haldar, R. Rudinger, and B. V. Durme. 2018. Hypothesis only baselines in Natural Language Inference. *arXiv preprint arXiv:1805.01042*.

P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *Association for Computational Linguistics (ACL)*.

M. Richardson, C. J. Burges, and E. Renshaw. 2013. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 193–203.

Marta Sabou, Kalina Bontcheva, Leon Derczynski, and Arno Scharl. 2014. Corpus annotation through crowdsourcing: Towards best practice guidelines. In *LREC*, pages 859–866.

R. Schwartz, M. Sap, Y. Konstas, L. Zilles, Y. Choi, and N. A. Smith. 2017. The effect of different writing tasks on linguistic style: A case study of the ROC story cloze task. In *Computational Natural Language Learning (CoNLL)*.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Alane Suhr, Stephanie Zhou, Iris Zhang, Huajun Bai, and Yoav Artzi. 2018. A corpus for reasoning about natural language grounded in photographs. *CoRR, abs/1811.00491*.

A. Talmor, J. Herzig, N. Lourie, and J. Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *North American Association for Computational Linguistics (NAACL)*.

A. Trischler, T. Wang, X. Yuan, J. Harris, A. Sordoni, P. Bachman, and K. Suleman. 2017. NewsQA: A machine comprehension dataset. In *Workshop on Representation Learning for NLP*.

Masatoshi Tsuchiya. 2018. Performance impact caused by hidden bias of training data for recognizing textual entailment. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Omar F. Zaidan and Chris Callison-Burch. 2011. Crowdsourcing translation: Professional quality from non-professionals. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

Rowan Zellers, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. From recognition to cognition: Visual commonsense reasoning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

# Robust Text Classifier on Test-Time Budgets

**Md Rizwan Parvez**
University of California Los Angeles
rizwan@cs.ucla.edu

**Tolga Bolukbasi**
Boston University
tolgab@bu.edu

**Kai-Wei Chang**
University of California Los Angeles
kwchang@cs.ucla.edu

**Venkatesh Saligrama**
Boston University
srv@bu.edu

## Abstract

We design a generic framework for learning a robust text classification model that achieves high accuracy under different selection budgets (a.k.a selection rates) at test-time. We take a different approach from existing methods and learn to dynamically filter a large fraction of unimportant words by a low-complexity selector such that any high-complexity classifier only needs to process a small fraction of text, relevant for the target task. To this end, we propose a data aggregation method for training the classifier, allowing it to achieve competitive performance on fractured sentences. On four benchmark text classification tasks, we demonstrate that the framework gains consistent speedup with little degradation in accuracy on various selection budgets.

## 1 Introduction

Recent advances in deep neural networks (DNNs) have achieved high accuracy on many text classification tasks. These approaches process the entire text and encode words and phrases in order to perform target tasks. While these models realize high accuracy, the computational time scales linearly with the size of the documents, which can be slow for a long document. In this context, various approaches based on modifying the RNN or LSTM architecture have been proposed to speed up the process (Seo et al., 2018; Yu et al., 2017). However, the processing in these models is still fundamentally sequential and needs to operate on the whole document which limits the computational gain. In contrast to previous approaches, we propose a novel framework for efficient text classification on long documents that mitigates sequential processing. The framework consists of a *selector* and a *classifier*. Given a selection budget as input, the *selector* performs a coarse one-shot selection deleting unimportant words and pass the remainder to the *classifier*. The *classifier* then takes the sentence fragments as an input and performs the target task. Figure 1 illustrates the procedure. This framework is general and agnostic to the architecture of the downstream *classifier* (e.g., RNN, CNN, Transformer).

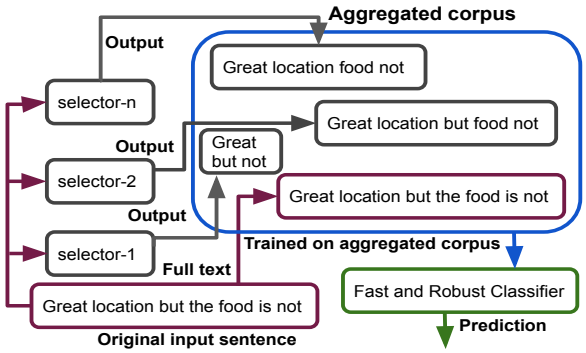


Figure 1: Our proposed framework. Given a selection rate, a *selector* is designed to select relevant words and pass them to the *classifier*. To make the classifier robust against fractured sentences, we aggregate outputs from different *selectors* and train the *classifier* on the aggregated corpus.

However, three challenges arise. First, to build a computationally inexpensive system, the *selector* must have negligible overhead. We adopt two effective yet simple architectures to design *selectors* based on word embeddings and bag-of-words. Second, training multiple distinct models for different budgets is unfeasible in practice, especially when model size is large. Hence, our goal is to learn a single *classifier* that can adapt to the output of any *selector* operating at any budget. Consequently, this *classifier* must be robust so that it can achieve consistent performance with different budgets. Third, the input to the *classifier* in our framework is a sequence of fractured sentences which is incompatible with a standard *classifier* that trained on the full texts, causing its performance degrades significantly. One potential but unfeasible solution is to train the *classifier* with a diverse collection of sentence fragments which is



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1167–1172,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

combinatorially numerous. Another approach is to randomly blank out text (a.k.a. blanking-noise), leads to marginalized feature distortion (Maaten et al., 2013) but this also leads to poor accuracy as DNNs leverage word combinations, sentence structure, which this approach does not account for. To mitigate this problem, we propose a data aggregation framework that augments the training corpus with outputs from *selector*s at different budget levels. By training the *classifier* on the aggregated *structured* blank-out text, the *classifier* learns to fuse fragmented sentences into a feature representation that mirrors the representation obtained on full sentences and thus realizes high-accuracy. We evaluate our approach through comprehensive experiments on real-world datasets. [1]

## 2 Related Work

Several approaches have been proposed to speed up the DNN in test time (Wu et al., 2017; Choi et al., 2017). LSTM-jump (Yu et al., 2017) learns to completely skip words deemed to be irrelevant and skim-RNN (Seo et al., 2018) uses a low-complexity LSTM to skim words rather than skipping. Another version of LSTM-jump, LSTM-shuttle (Fu and Ma, 2018) first skips a number of words, then goes backward to recover lost information by reading some words skipped before. All these approaches require to modify the architecture of the underlying classifier and cannot easily extend to another architecture. In contrast, we adopt existing *classifier* architectures (e.g., LSTM, BCN (McCann et al., 2017)) and propose a meta-learning algorithm to train the model. Our framework is generic and a *classifier* can be viewed as a black-box. Similar to us, Lei et al. (2016) propose a *selector-classifier* framework to find text snippets as justification for text classification but their *selector* and *classifier* have similar complexity and require similar processing times; therefore, it is not suitable for computation gain. Various feature selection approaches (Chandrashekar and Sahin, 2014) have been discussed in literature. For example, removing predefined stop-words (see Appendix A), attention based models (Bahdanau et al., 2015; Luong et al., 2015), feature subspace selection methods (e.g., PCA), and applying the L1 regularization (e.g., Lasso (Tibshirani, 1996) or Group Lasso (Faruqui et al., 2015), BLasso (Gao et al., 2007)). However, these approaches either cannot obtain sparse features or cannot straightforwardly be applied to speed up a DNN *classifier*. Different from ours, Viola and Jones (2001); Trapeznikov and Saligrama (2013); Karayev et al. (2013); Xu et al. (2013); Kusner et al. (2014); Bengio et al. (2016); Leroux et al. (2017); Zhu et al. (2019); Nan and Saligrama (2017); Bolukbasi et al. (2017) focus on gating various components of existing networks. Finally, aggregating data or models has been studied under different contexts (e.g., in context of reinforcement learning (Ross et al., 2011), Bagging models (Breiman, 1996), etc.) while we aggregate the data output from *selectors* instead of models.

[1] Our source code is available at: https://github.com/uclanlp/Fast-and-Robust-Text-Classification

## 3 Classification on a Test-Time Budget

Our goal is to build a robust *classifier* along with a suite of *selectors* to achieve good performance with consistent speedup under different selection budgets at test-time. Formally, a *classifier* $C(\hat{x})$ takes a word sequence $\hat{x}$ and predicts the corresponding output label $y$, and a *selector* $S_b(x)$ with selection budget $b$ takes an input word sequence $x = \{w_1, w_2, \ldots, w_N\}$ and generates a binary sequence $S_b(x) = \{z_{w_1}, z_{w_2}, \ldots, z_{w_N}\}$ where $z_{w_k} \in \{0, 1\}$ represents if the corresponding word $w_k$ is selected or not. We denote the sub-sequence of words generated after filtering by the *selector* as $I\big(x, S_b(x)\big) = \{w_k : z_{w_k} = 1, \forall w_k \in x\}$. We aim to train a *classifier* $C$ and the *selector* $S_b$ such that $I\big(x, S_b(x)\big)$ is sufficient to make accurate prediction on the output label (i.e., $C\Big(I\big(x, S_b(x)\big)\Big) \approx C(x)$). The selection budget (a.k.a selection rate) $b$ is controlled by the hyper-parameters of the *selector*. Higher budget often leads to higher accuracy and longer test time.

### 3.1 Learning a *Selector*

We propose two simple but efficient *selector*s. **Word Embedding (WE) *selector*.** We consider a parsimonious word-selector using word embeddings (e.g., GloVe (Pennington et al., 2014)) as features to predict important words. We assume the informative words can be identified independently and model the probability that a word $w_k$ is selected by $P(z_{w_k} = 1|w_k) = \sigma(\theta_S^T \vec{w}_k)$, where $\theta_S$ is the model parameters of the *selector* $S_b$, $\vec{w}_k$ is the corresponding word vector, and $\sigma$ is the sigmoid function. As we do not have explicit anno-

tations about which words are important, we train the *selector* $S_b$ along with a *classifier* $C$ in an end-to-end manner following Lei et al. (2016), and an L1-regularizer is added to control the sparsity (i.e., selection budget) of $S_b(x)$.

**Bag-of-Words *selector*.** We also consider using an L1-regularized linear model (Zou and Hastie, 2005; Ng, 2004; Yuan et al., 2010) with bag-of-words features to identify important words. In the bag-of-words model, for each document $x$, we construct a feature vector $\vec{x} \in \{0,1\}^{|V|}$, where $|V|$ is the size of the vocabulary. Each element of the feature vector $\vec{x}_w$ represents if a specific word $w$ appearing in the document $x$. Given a training set $\mathcal{X}$, the linear model optimizes the L1-regularized task loss. For example, in case of a binary classification task (output label $y \in \{1, -1\}$),

$$J(x_t, y_t) = \log\left(1 + \exp(-y_t \theta^T \vec{x_t})\right)$$

$$\theta^* = \arg\min_\theta \sum_{(x_t,y_t)\in\mathcal{X}} J(x_t, y_t) + \frac{1}{b}\|\theta\|_1,$$

where $\theta \in R^{|V|}$ is a weight vector to be learned, $\theta_w$ corresponds to word $w \in V$, and $b$ is a hyper-parameter controlling the sparsity of $\theta^*$ (i.e., selection budget). The lower the budget $b$ is, the sparser the selection is. Based on the optimal $\theta^*$, we construct a *selector* that picks word $w$ if the corresponding $\theta^*_w$ is non-zero. Formally, the bag-of-words *selector* outputs $S_b(x) = \{\delta(\theta_w \neq 0) : w \in x\}$, where $\delta$ is an indicator function.

### 3.2 The Data Aggregation Framework

In order to learn to fuse fragmented sentences into a robust feature representation, we propose to train the *classifier* on the aggregated corpus of structured blank-out texts.

Given a set of training data $\mathcal{X} = \{(x_1, y_1), .., (x_t, y_t), .., (x_m, y_m)\}$, we assume we have a set of selectors $\mathcal{S} = \{S_b\}$ with different budget levels trained by the framework discussed in Section 3.1. To generate an aggregated corpus, we first apply each *selector* $S_b \in \mathcal{S}$ on the training set, and generate corresponding blank-out corpus $\mathcal{I}(\mathcal{X}, S_b) = \{I(x_t, S_b(x_t)), \forall x_t \in \mathcal{X}\}$. Then, we create a new corpus by aggregating the blank-out corpora: $\mathcal{T} = \bigcup_{S_b\in\mathcal{S}} \mathcal{I}(\mathcal{X}, S_b)$.[2] Finally, we train the *classifier* $C_\mathcal{T}$ on the aggregated corpus $\mathcal{T}$. As $C_\mathcal{T}$ is trained on documents with distortions, it learns to make predictions with different budget levels. The training procedure is summarized in Algorithm 1. In the following, we discuss two extensions of our data aggregation framework.

**Algorithm 1:** Data Aggregated Training Schema

**Input:** Training corpus $\mathcal{X}$, a set of *selectors* with different budget levels $\mathcal{S} = \{S_b\}$, *classifier* class $C$

**Output:** A robust *classifier* $C_\mathcal{T}$

```
Initialize the aggregated corpus: T ← X
for S_b ∈ S do
    S_b ← Train a selector S_b ∈ S with budget level b on X
    Generate a blank-out dataset I(X, S_b)
    Aggregate data: T ← T ∪ I(X, S_b)
C_T ← Train a classifier C on T
return C_T
```

First, the blank-out data can be generated from different classes of *selectors* with different features or architectures. Second, the blank-out and selection can be done in phrase or sentence level. Specifically, if phrase boundaries are provided, a phrase-level aggregation can avoid a *selector* from breaking compound nouns or meaningful phrases (e.g., "Los Angeles", "not bad"). Similarly, for multi-sentenced documents, we can enforce the *selector* to pick a whole sentence if any word in the sentence is selected.

## 4 Experiments

To evaluate the proposed approach, we consider four benchmark datasets: *SST-2* (Socher et al., 2013), *IMDB* (Maas et al., 2011), *AGNews* (Zhang et al., 2015), and *Yelp* (Conneau et al., 2016) and two widely used architectures for classification: *LSTM*, and *BCN* (McCann et al., 2017). The statistics of the datasets are summarized in Table 2. We evaluate the computation gain of models in terms of overall test-time, and the performance in terms of accuracy. We follow Seo et al. (2018) to estimate the test-time of models on CPU[3] and exclude the time for data loading.

In our approach, we train a *classifier* with both WE and bag-of-words *selector*s with 6 selection budgets[4] {50%, 60%, ..., 100%} by the word-

[2] Note that, the union operation is used just to aggregate the train instances which does not hinder the model training (e.g., discrete variables).

[3] Machine specification is in Appendix C.

[4] For the very large Yelp dataset, 3 selection budgets {50%, 60%, 70%} are used.

| Model | SST-2 | | | | IMDB | | | | AGNews | | | | Yelp | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | acc. | selection(%) | time | speedup | acc. | selection(%) | time | speedup | acc. | selection(%) | time | speedup | acc. | selection(%) | time | speedup |
| *Baseline* | **85.7** | 100 | 9 | 1x | 91.0 | 100 | 1546 | 1x | 92.3 | 100 | 59 | 1x | **66.5** | 100 | 3487 | 1x |
| Bag-of-Words | 78.8 | 75 | 5.34 | 1.7x | 91.5 | 91 | 1258 | **1.2x** | 92.9 | 97 | 48 | 1.2x | 59.7 | 55 | 2325 | **1.6x** |
| Our framework | 82.6 | 65 | 4.6 | **2x** | 92.0 | 91 | 1297 | **1.2x** | 93.1 | 91 | 46 | **1.3x** | 64.8 | 55 | 2179 | **1.6x** |
| | 85.3 | 0 | 9 | 1x | **92.1** | 0 | 1618 | 1x | **93.2** | 0 | 57 | 1x | 66.3 | 0 | 3448 | 1x |

Table 1: Accuracy and speedup on the test datasets. Test-times are measured in seconds. The speedup rate is calculated as the running time of a model divided by the running time of the corresponding baseline. For our framework, top row denotes the best speedup and the bottom row denotes the best test accuracy achieved. Overall best accuracies and best speedups are boldfaced. Our framework achieves accuracies better than baseline with a speedup of **1.2x** and **1.3x** on IMDB, and AGNews respectively. With same or higher speedup, our accuracies are much better than Bag-of-Words.

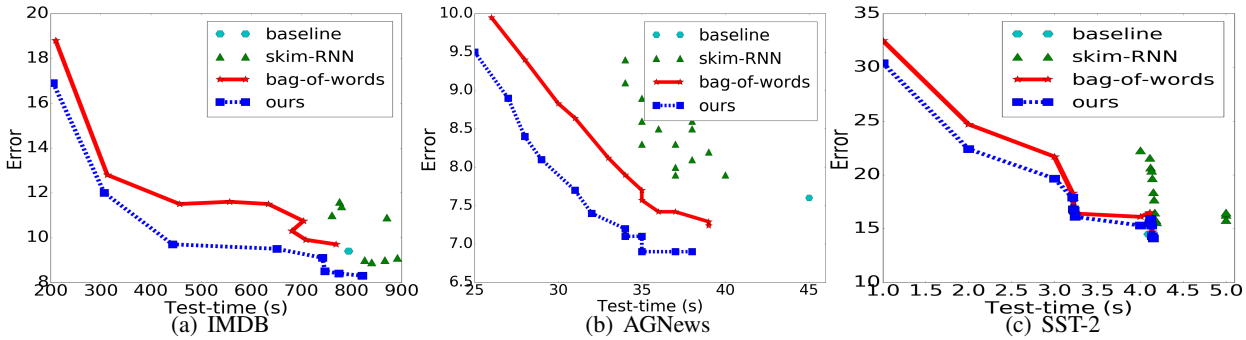


(a) IMDB (b) AGNews (c) SST-2

Figure 2: Performance under different test-times on IMDB, AGNews, and SST-2. All the approaches use the same LSTM model as the back-end. Bag-of-Words model and our framework have the same bag-of-words *selector* cascaded with this LSTM *classifier* trained on the original training corpus and aggregated corpus, respectively. Our model (blue dashed line) significantly outperform others for any test-time budget. Also its performance is robust, while results of skim-RNN is inconsistent with different budget levels.

| Dataset | #class | Vocabulary | Size (Train/Valid/Test) | Avg. Len |
|---|---|---|---|---|
| SST | 2 | 13,750 | 6,920/872/1,821 | 19 |
| IMDB | 2 | 61,046 | 21,143/3,857/25,000 | 240 |
| AGNews | 4 | 60,088 | 101,851/18,149/7,600 | 43 |
| Yelp | 5 | 1,001,485 | 600k/50k/50k | 149 |

Table 2: Dataset statistics.

level data aggregation framework. We evaluate the computation gain of the proposed method through a comparative study of its performance under different test-times by varying the selection budgets[5] in comparison to the following approaches: (1) *Baseline*: the original *classifier* (i.e., no *selector*, no data aggregation) (2) *skim-RNN*: we train a skim-RNN model and vary the amount of text to skim (i.e., test-time) by tuning $\theta$ parameter as in Seo et al. (2018). (3) Bag-of-Words: filtering words by the bag-of-words *selector* and feeding the fragments of sentences to the original *classifier* (i.e., no data aggregation). This approach serves as a good baseline and has been considered in the context of linear models (e.g., Chang and Lin (2008)). For a fair comparison, we implement all approaches upon the same framework using AllenNLP library[6], including a re-implementation of the existing state-of-art speedup framework skim-RNN (Seo et al., 2018)[7]. As skim-RNN is designed specifically for accelerating the LSTM model, we only compare with skim-RNN using LSTM *classifier*. Each corresponding model is selected by tuning parameters on validation data. The model is then frozen and evaluated on test-data for different selection budgets.

Figure 2 demonstrates the trade-off between the performance, and the test-time for each setting. Overall, we expect the error to decrease with a larger test-time budget. From Figure 2, on all of the IMDB, AGNews, and SST-2 datasets, LSTM *classifier* trained with our proposed data aggregation not only achieves the lowest error curve but also the results are robust and consistent. That is our approach achieves higher performance across different test-time budgets and its performance is a predictable monotonic function of the test-time budget. However, the performance of skim-RNN exhibits inconsistency for different budgets. As a matter of fact, for multiple budgets, none of the skim-RNN, and LSTM-jump address the prob-

[5] In Appendix B, we discuss how to vary these budgets.

[6] https://allennlp.org/

[7] The official skim-RNN implementation is not released.

| | |
|---|---|
| World News | .. **plant** searched. Kansai Electric Power's **nuclear** power **plant** in Fukui .. **was searched by police** Saturday .. |
| Business | Telecom Austria taps the Bulgarian **market**. **Telecom** Austria, Austrias **largest** telecoms operator, obtained .. |
| Sci/Tech | .. Reuters - **Software security** companies **and** handset makers, including Finland's **Nokia** (NOK1V.HE), **are** .. |

Table 3: Examples of the WE *selector* output on AGNews. Bold words are selected.

lem of different word distribution between training and testing. Therefore, similar to skim-RNN, we anticipate that the behavior of LSTM-jump will be inconsistent as well[8]. Additionally, since LSTM-jump has already been shown to be outperformed by skim-RNN, we do not further compare with it. Next, we show that our framework is generic and can incorporate with other different *classifier*s, such as BCN (see Table 1).[9] When phrase boundary information is available, our model can further achieve **86.7** in accuracy with **1.7x** speedup for BCN on SST-2 dataset by using phrase-level data aggregation. Finally, one more advantage of the proposed framework is that the output of the *selector* is interpretable. In Table 3, we present that our framework correctly selects words such as "Nokia", "telecom", and phrases such as "searched by police", "software security" and filters out words like "Aug.", "users" and "products".

Note that nevertheless we focus on efficient inference, empirically our method is no more complex than the baseline during training. Despite the number of training instances increases, and so does the training time for each epoch, the number of epochs we require for obtaining a good model is usually smaller. For example, on the Yelp corpus, we only need 3 epochs to train a BCN *classifier* on the aggregated corpus generated by using 3 different *selector*s, while training on the original corpus requires 10 epochs.

## 5 Conclusion

We present a framework to learn a robust *classifier* under test-time constraints. We demonstrate that the proposed *selector*s effectively select important words for *classifier* to process and the data aggregation strategy improves the model performance. As future work we will apply the framework for other text reading tasks. Another promising direction is to explore the benefits of text classification model in an edge-device setting. This problem naturally arises with local devices (e.g., smart watches or mobile phones), which do not have sufficient memory or computational power to execute a complex *classifier*, and instances must be sent to the cloud. This setting is particularly suited to ours since we could choose to send only the important words to the cloud. In contrast, skim-RNN and LSTM-jump, which process the text sequentially, have to either send the entire text to the server or require multiple rounds of communication between the server and local devices resulting in high network latency.

[8] As an example, from Table 6 in Yu et al. (2017), the performance of LSTM-jump drops from 0.881 to 0.854 although it takes longer test-time (102s) than the baseline (81.7s).

[9] Because of the inherent accuracy/inference-time trade-off, it is difficult to depict model comparisons. For this reason, in Figure 2, we plot the trade-off curve to demonstrate the best speedup achieved by our model for achieving near state-of-art performance. On the other hand, test results are tabulated in Table 1 to focus attention primarily on accuracy.

## 6 Acknowledgments

We thank the anonymous reviewers for their insightful feedback. We also thank UCLA-NLP group for discussion and comments. This work was supported in part by National Science Foundation grants IIS-1760523 and CCF-1527618.

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *ICLR*.

Emmanuel Bengio, Pierre-Luc Bacon, Joelle Pineau, and Doina Precup. 2016. Conditional computation in neural networks for faster models. *ICLR*.

Tolga Bolukbasi, Joseph Wang, Ofer Dekel, and Venkatesh Saligrama. 2017. Adaptive neural networks for efficient inference. In *ICML*.

Leo Breiman. 1996. Bagging predictors. *Machine learning*, 24(2):123–140.

Girish Chandrashekar and Ferat Sahin. 2014. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28.

Yin-Wen Chang and Chih-Jen Lin. 2008. Feature ranking using linear SVM. In *Causation and Prediction Challenge*, pages 53–64.

Eunsol Choi, Daniel Hewlett, Jakob Uszkoreit, Illia Polosukhin, Alexandre Lacoste, and Jonathan Berant. 2017. Coarse-to-fine question answering for long documents. In *ACL*.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2016. Very deep convolutional networks for natural language processing. *arXiv preprint arXiv:1606.01781*.

Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah A Smith. 2015. Sparse overcomplete word vector representations. In *ACL-ICJNLP*.

Tsu-Jui Fu and Wei-Yun Ma. 2018. Speed reading: Learning to read forbackward via shuttle. In *EMNLP*.

Jianfeng Gao, Galen Andrew, Mark Johnson, and Kristina Toutanova. 2007. A comparative study of parameter estimation methods for statistical natural language processing. In *ACL*.

Sergey Karayev, Mario Fritz, and Trevor Darrell. 2013. Dynamic feature selection for classification on a budget. In *ICML*.

Matt J Kusner, Wenlin Chen, Quan Zhou, Zhixiang Eddie Xu, Kilian Q Weinberger, and Yixin Chen. 2014. Feature-Cost Sensitive Learning with Submodular Trees of Classifiers. In *AAAI*.

Tao Lei, Regina Barzilay, and Tommi S. Jaakkola. 2016. Rationalizing neural predictions. In *EMNLP*.

Sam Leroux, Steven Bohez, Elias De Coninck, Tim Verbelen, Bert Vankeirsbilck, Pieter Simoens, and Bart Dhoedt. 2017. The cascading neural network: building the internet of smart things. *KAIS*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *EMNLP*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *ACL*.

Laurens Maaten, Minmin Chen, Stephen Tyree, and Kilian Weinberger. 2013. Learning with marginalized corrupted features. In *ICML*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *NeurIPS*.

Feng Nan and Venkatesh Saligrama. 2017. Adaptive classification for prediction under a budget. In *NeurIPS*.

Andrew Y Ng. 2004. Feature selection, l 1 vs. l 2 regularization, and rotational invariance. In *ICML*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. 2011. No-regret reductions for imitation learning and structured prediction. *AISTATS*.

Min Joon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. 2018. Neural speed reading via skim-rnn. *ICLR*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*.

Robert Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288.

Kirill Trapeznikov and Venkatesh Saligrama. 2013. Supervised sequential classification under budget constraints. In *AISTATS*.

Paul Viola and Michael Jones. 2001. Robust real-time object detection. *IJCV*.

Felix Wu, Ni Lao, John Blitzer, Guandao Yang, and Kilian Q. Weinberger. 2017. Fast reading comprehension with convnets. *CoRR*.

Zhixiang Xu, Matt Kusner, Minmin Chen, and Kilian Q Weinberger. 2013. Cost-Sensitive Tree of Classifiers. In *ICML*.

Adams Wei Yu, Hongrae Lee, and Quoc Le. 2017. Learning to skim text. In *ACL*.

Guo-Xun Yuan, Kai-Wei Chang, Cho-Jui Hsieh, and Chih-Jen Lin. 2010. A comparison of optimization methods and software for large-scale l1-regularized linear classification. *JMLR*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NeurIPS*, pages 649–657.

P. Zhu, A. Acar, F. Nan, P. Jain, and V. Saligrama. 2019. Cost-aware inference for iot devices. In *AISTATS*.

Hui Zou and Trevor Hastie. 2005. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320.

# Commonsense Knowledge Mining from Pretrained Models

**Joshua Feldman***, **Joe Davison***, **Alexander M. Rush**
School of Engineering and Applied Sciences
Harvard University
{joshua_feldman@g, jddavison@g, srush@seas}.harvard.edu

## Abstract

Inferring commonsense knowledge is a key challenge in natural language processing, but due to the sparsity of training data, previous work has shown that supervised methods for commonsense knowledge mining underperform when evaluated on novel data. In this work, we develop a method for generating commonsense knowledge using a large, pre-trained bidirectional language model. By transforming relational triples into masked sentences, we can use this model to rank a triple's validity by the estimated pointwise mutual information between the two entities. Since we do not update the weights of the bidirectional model, our approach is not biased by the coverage of any one commonsense knowledge base. Though this method performs worse on a test set than models explicitly trained on a corresponding training set, it outperforms these methods when mining commonsense knowledge from new sources, suggesting that unsupervised techniques may generalize better than current supervised approaches.

## 1 Introduction

Commonsense knowledge consists of facts about the world which are assumed to be widely known. For this reason, commonsense knowledge is rarely stated explicitly in natural language, making it challenging to infer this information without an enormous amount of data (Gordon and Van Durme, 2013). Some have even argued that machine learning models cannot learn common sense implicitly (Davis and Marcus, 2015).

One method for mollifying this issue is directly augmenting models with commonsense knowledge bases (Young et al., 2018), which typically contain high-quality information but with low coverage. These knowledge bases are represented as a graph, with nodes consisting of conceptual entities (i.e. `dog`, `running away`, `excited`, etc.) and the pre-defined edges representing the nature of the relations between concepts (`IsA`, `UsedFor`, `CapableOf`, etc.). Commonsense knowledge base completion (CKBC) is a machine learning task motivated by the need to improve the coverage of these resources. In this formulation of the problem, one is supplied with a list of candidate entity-relation-entity triples, and the task is to distinguish which of the triples express valid commonsense knowledge and which are fictitious (Li et al., 2016).

Several approaches have been proposed for training models for commonsense knowledge base completion (Li et al., 2016; Jastrzebski et al., 2018). Each of these approaches uses some sort of supervised training on a particular knowledge base, evaluating the model's performance on a held-out test set from the same database. These works use relations from ConceptNet, a crowd-sourced database of structured commonsense knowledge, to train and validate their models (Liu and Singh, 2004). However, it has been shown that these methods generalize poorly to novel data (Li et al., 2016; Jastrzebski et al., 2018). Jastrzebski et al. (2018) demonstrated that much of the data in the ConceptNet test set were simply rephrased relations from the training set, and that this train-test set leakage led to artificially inflated test performance metrics. This problem of train-test leakage is typical in knowledge base completion tasks (Toutanova et al., 2015; Dettmers et al., 2018).

Instead of training a predictive model on any specific database, we attempt to utilize the world knowledge of large language models to identify commonsense facts directly. By constructing a candidate piece of knowledge as a sentence, we can use a language model to approximate the likelihood of this text as a proxy for its truthfulness.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1173–1178,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

In particular, we use a masked language model to estimate point-wise mutual information between entities in a possible relation, an approach that differs significantly from fine-tuning approaches used for other language modeling tasks. Since the weights of the model are fixed, our approach is not biased by the coverage of any one dataset. As we might expect, our method underperforms when compared to previous benchmarks on the ConceptNet common sense triples dataset (Li et al., 2016), but demonstrates a superior ability to generalize when mining novel commonsense knowledge from Wikipedia.

**Related Work** Schwartz et al. (2017) and Trinh and Le (2018) demonstrate a similar approach to using language models for tasks requiring commonsense, such as the Story Cloze Task and the Winograd Schema Challenge, respectively (Mostafazadeh et al., 2016; Levesque et al., 2012). Bosselut et al. (2019) and Trinh and Le (2019) use unidirectional language models for CKBC, but their approach requires a supervised training step. Our approach differs in that we intentionally avoid training on any particular database, relying instead on the language model's general world knowledge. Additionally, we use a bidirectional masked model which provides a more flexible framework for likelihood estimation and allows us to estimate point-wise mutual information. Although it is beyond the scope of this paper, it would be interesting to adapt the methods presented here for the related task of generating new commonsense knowledge (Saito et al., 2018).

## 2 Method

Given a commonsense head-relation-tail triple $x = (\mathbf{h}, r, \mathbf{t})$, we are interested in determining the validity of that tuple as a representation of a commonsense fact. Specifically, we would like to determine a numeric score $y \in \mathbb{R}$ reflecting our confidence that a given tuple represents true knowledge.

We assume that heads and tails are arbitrary-length sequences of words in a vocabulary $\mathcal{V}$ so that $\mathbf{h} = \{h_1, h_2, \ldots, h_n\}$ and $\mathbf{t} = \{t_1, t_2, \ldots, t_m\}$. We further assume that we have a known set of possible relations $\mathcal{R}$ so that $r \in \mathcal{R}$.

The goal is to determine a function $f$ that maps relational triples to validity scores. We propose decomposing $f(x) = \sigma(\tau(x))$ into two subcomponents: a sentence generation function $\tau$ which maps a triple to a single sentence, and a scoring model $\sigma$ which then determines a validity score $y$.

Our approach relies on two types of pretrained language models. Standard unidirectional models are typically represented as autoregressive probabilities:

$$p(w_1, w_2, \ldots, w_m) = \prod_i^m p(w_i | w_1, \ldots, w_{i-1})$$

Masked bidirectional models such as BERT, proposed by Devlin et al. (2018), instead model in both directions, training word representations conditioned both on future and past words. The masking allows any number of words in the sequence to be hidden. This setup provides an intuitive framework to evaluate the probability of any word in a sequence conditioned on the rest of the sequence,

$$p(w_i | w'_{1:i-1}, w'_{i+1:m})$$

where $w' \in \mathcal{V} \cup \{\kappa\}$ and $\kappa$ is a special token indicating a masked word.

### 2.1 Generating Sentences from Triples

We first consider methods for turning a triple such as `(ferret, AtLocation, pet store)` into a sentence such as "the ferret is in the pet store". Our approach is to generate a set of candidate sentences via hand-crafted templates and select the best proposal according to a language model.

For each relation $r \in \mathcal{R}$, we hand-craft a set of sentence templates. For example, one template in our experiments for the relation `AtLocation` is, "you are likely to find `HEAD` in `TAIL`". For the above example, this would yield the sentence, "You are likely to find ferret in pet store".

Because these sentences are not always grammatically correct, such as in the above example, we apply a simple set of transformations. These consist of inserting articles before nouns, converting verbs into gerunds, and pluralizing nouns which follow numbers. See the supplementary materials for details and Table 1 for an example. We then enumerate a set of alternative sentences $\mathcal{S} = \{S_1, \ldots, S_j\}$ resulting from each template and from all combinations of transformations. This yields a set of candidate sentences for each data point. We then select the candidate sentence with the highest log-likelihood according to a pre-trained unidirectional language model $P_{\text{coh}}$.

$$S^* = \arg\max_{S \in \mathcal{S}} \left[ \log P_{\text{coh}}(S) \right]$$

| Candidate Sentence $S_i$ | $\log p(S_i)$ |
|---|---|
| "musician can playing musical instrument" | $-5.7$ |
| "musician can be play musical instrument" | $-4.9$ |
| "musician often play musical instrument" | $-5.5$ |
| "a musician can play a musical instrument" | $\mathbf{-2.9}$ |

Table 1: Example of generating candidate sentences. Several enumerated sentences for the triple `(musician, CapableOf, play musical instrument)`. The sentence with the highest log-likelihood according to a pretrained language model is selected.

We refer to this method of generating a sentence from a triple as COHERENCY RANKING. Coherency Ranking operates under the assumption that natural, grammatical sentences will have a higher likelihood than ungrammatical or unnatural sentences. See an example subset of sentence candidates and their corresponding scores in Table 1. From a qualitative evaluation of the selected sentences, we find that this approach produces sentences of significantly higher quality than those generated by deterministic rules alone. We also perform an ablation study in our experiments demonstrating the effect of each component on CKBC performance.

### 2.2 Scoring Generated Triples

Assuming we have generated a proper sentence from a relational triple, we now need a way to score its validity with a pretrained model that considers the relationship between the relation entities. We therefore propose using the estimated point-wise mutual information (PMI) of the head $\mathbf{h}$ and tail $\mathbf{t}$ of a triple conditioned on the relation $r$, defined as,

$$\text{PMI}(\mathbf{t}, \mathbf{h}|r) = \log p(\mathbf{t}|\mathbf{h}, r) - \log p(\mathbf{t}|r)$$

We can estimate these scores by using a masked bidirectional language model, $P_{\text{cmp}}$. In the case where the tail is a single word, the model allows us to evaluate the conditional likelihood of a single triple component $p(\mathbf{t}|\mathbf{h}, r)$ by computing $P_{\text{cmp}}(w_i = \mathbf{t}\ |w_{1:i-1}, w_{i+1:m})$ for the tail word.

In practice, the tail might be realized as a $j$-word phrase. To handle this complexity, we use a greedy approximation of its probability. We first mask all of the tail words and compute the probability of each. We then find the word with highest probability $p_k$, substitute it back in, and repeat $j$ times. Finally, we calculate the total conditional likelihood of the tail by the product of these terms, $p(\mathbf{t}|\mathbf{h}, r) = \prod_{k=1}^{j} p_k$.

The marginal $p(\mathbf{t}|r)$ is computed similarly, but in this case we mask the head throughout. For example, to compute the marginal tail probability for the sentence, "You are likely to find a ferret in the pet store" we mask both the head and the tail and then sequentially unmask the tail words only: "You are likely to find a $\kappa_{h1}$ in the $\kappa_{t1}$ $\kappa_{t2}$". If $\kappa_{t2}$ = "store" has a higher probability than $\kappa_{t1}$ = "pet", we unmask "store" and compute "You are likely to find a $\kappa_{h1}$ in the $\kappa_{t1}$ store". The marginal likelihood $p(\mathbf{t}|r)$ is then the product of the two probabilities.

The final score combines the marginal and conditional likelihoods by employing a weighted form of the point-wise mutual information,

$$\text{PMI}_\lambda(\mathbf{t}, \mathbf{h}|r) = \lambda \log p(\mathbf{t}|\mathbf{h}, r) - \log p(\mathbf{t}|r)$$

where $\lambda$ is treated as a hyperparameter. Although exact PMI is symmetrical, the approximate model itself is not. We therefore average $\text{PMI}_\lambda(\mathbf{t}, \mathbf{h}|r)$ and $\text{PMI}_\lambda(\mathbf{h}, \mathbf{t}|r)$ to reduce the variance of our estimates, computing the masked head values rather than the tail values in the latter.

## 3 Experiments

To evaluate the Coherency Ranking approach we measure whether it can distinguish between valid and invalid triples. For our masked model, we use BERT-large (Devlin et al., 2018). For sentence ranking, we use the GPT-2 117M LM (Radford et al., 2019). The relation templates and grammar transformation rules which we use can be found in the supplementary materials.

We compare the proposed method to several baselines. Following Trinh and Le (2018), we evaluate a simple CONCATENATION method for generating sentences, splitting the relation $r$ into separate words and concatenating it with the head and tail. For the triple `(ferret, AtLocation, pet store)`, the Concatenation approach would yield, "ferret at location pet store".

We also evaluate CKBC performance when we construct sentences by applying a single handcrafted template. Since each triple is mapped to a sentence with a single template without any grammatical transformations, we refer to this as the TEMPLATE method. Using the Template approach, `(ferret, AtLocation, pet store)`

| Model | Task 1 | Task 2 |
|---|---|---|
| Unsupervised | | |
| CONCATENATION | 68.8 | $2.95 \pm 0.11$ |
| TEMPLATE | 72.2 | $2.98 \pm 0.11$ |
| TEMPL.+GRAMMAR | 74.4 | $2.56 \pm 0.13$ |
| COHERENCY RANK | 78.8 | $\mathbf{3.00} \pm 0.12$ |
| Supervised | | |
| DNN | **89.2** | 2.50 |
| FACTORIZED | 89.0 | 2.61 |
| PROTOTYPICAL | 79.4 | 2.55 |

Table 2: Main results for Task 1: Commonsense knowledge base completion (test F1 score) and Task 2: Wikipedia mining (quality scores out of 4). Results are included from the sentence generation methods of simple concatenation, hand-crafted templates, templates plus grammatical transformations, and coherency ranking. DNN, Factorized, and Prototypical models are described in Jastrzebski et al. (2018).

would become "You are likely to find ferret in pet store" using the template "you are likely to find HEAD in TAIL".

Next, we extend the Template method by applying deterministic grammatical transformations, which we refer to as the TEMPLATE + GRAMMAR approach. Like the full approach, these transformations involve adding articles before nouns, converting verbs into gerunds, and pluralizing nouns following numbers. The Template + Grammar approach differs from Coherency Ranking in that all transformations are applied to every sentence instead of applying combinations of transformations and templates, which are then ranked by a language model. Returning to our example, the Template + Grammar method produces "You are likely to find a ferret in a pet store". While this sentence is grammatical, applying this method to `(star, AtLocation, outer space)` yields "You are likely to find a star in an outer space", which is incorrect.

We compare our results to the supervised models from the work of Jastrzebski et al. (2018) and the best performing model from Li et al. (2016). Jastrzebski et al. (2018) introduce FACTORIZED and PROTOTYPICAL models. The Factorized model embeds the head, relation, and tail in a vector space and then produces a score by taking a linear combination of the inner products between each pair of embeddings. The Prototypical model is similar, but does not include the inner product between head and tail. Li et al. (2016) evaluate a deep neural network (DNN) for CKBC. They concatenate embeddings for the head, relation, and tail, which they then feed through a multilayer perceptron with one hidden layer. All three models are trained on 100,000 ConceptNet triples.

**Task 1: Commonsense Knowledge Base Completion** Our experimental setup follows Li et al. (2016), evaluating our model with their test set (n = 2400) containing an equal number of valid and invalid triples. The valid triples are from the crowd-sourced Open Mind Common Sense (OMCS) entries in the ConceptNet 5 dataset (Speer and Havasi, 2012). Invalid triples are generated by replacing an element of a valid tuple with another randomly selected element.

We use our scoring method to classify each tuple as valid or invalid. To this end, we use our method to assign a score to each tuple and then group the resulting scores into two clusters. Instances in the cluster with the higher mean PMI are labeled as valid, and the remainder are labeled as invalid. We use expectation-maximization with a mixture of Gaussians to cluster. We also tune the PMI weight via grid search over 90 points from $\lambda \in [0.5, 5.]$, using the Akaike information criterion of the Gaussian mixture model for evaluation (Akaike, 1974).

Table 2 shows the full results. Our unsupervised approach achieves a test set F1 score of 78.8, comparable to the 79.4 F1 score found by the supervised prototypical approach. The Factorized and DNN models significantly outperformed our approach with F1 scores of 89.2 and 89.0, respectively. Our grid search found an optimal $\lambda$ value of 1.65 for the Concatenation sentence generation model and 1.55 for the Coherency Ranking model. The Template and Template + Grammar methods found lambda values of 1.20 and 0.95, respectively.

**Task 2: Mining Wikipedia** To assess the model's ability to generalize to unseen data, we evaluate our unsupervised model in comparison to previous supervised methods on the task of mining commonsense knowledge from Wikipedia. In their evaluations, Li et al. (2016) curate a set of 1.7M triples across 10 relations by applying part-of-speech patterns to Wikipedia articles. We sample 300 triples from each relation. We apply our

method to evaluate these 3000 triples. Using the approach described by Speer and Havasi (2012), and followed by Li et al. (2016) and Jastrzebski et al. (2018), two human annotators manually rate the 100 triples with the highest predicted score on a 0 to 4 scale: 0 (Doesn't make sense), 1 (Not true), 2 (Opinion/Don't know), 3 (Sometimes true), and 4 (Generally true). We tuned $\lambda$ by measuring the quality of the 100 triples with the highest predicted score across $\lambda \in \{1, 2, \ldots, 9, 10\}$.

The top 100 triples selected by our model were assigned a mean rating of 3.00 ($\lambda = 4$) with a standard error of 0.11 under the Coherency Ranking approach, well exceeding the performance of current supervised methods (Table 2). Standard errors were calculated using 1000 bootstrap samples of the top 100 triples. The ratings assigned by the two human annotators had a 0.50 Pearson correlation and 0.23 kappa inter-annotator agreement. Rater disagreements occur most frequently when triples are ambiguous or difficult to interpret. Notably, if we bucket the five scores into just two categories of *true* and *false*, this disagreement rate drops by 50%. To give a sense of the types of commonsense knowledge our models struggle to capture, we report the top 100 most confident predictions that receive an average score below 3 in the supplementary material. Notably, some of the top 100 triples our model identified were indeed true, but would not be reasonably considered common sense (e.g. `(vector bundle, HasProperty, manifold)`). This suggests that our approach may be applicable to mining knowledge beyond common sense.

**Analysis: Sentence Generation** In order to measure the impact of sentence generation on our model, we select a sample of 100 sentences and group the results by a) whether the sentence contained a grammatical error, and b) whether the sentence misrepresented the meaning of the triple. For example, the triple `(golf, HasProperty, good)` yields the sentence "golf is a good", which is grammatically correct but conveys the wrong meaning. On both Wikipedia mining and CKBC, we find that misrepresenting meaning has an adverse impact on model performance. In CKBC, we also find that grammar has a high impact on the resulting F1 scores (Table 3). Future work could therefore focus on designing templates that more reliably encode a relation's true meaning.

| **Task 1** | **N (/100)** | **F1 Score** |
|---|---|---|
| GRAMMATICAL | 75 | 79.1 |
| UNGRAMMATICAL | 25 | 66.7 |
| CORRECT MEANING | 91 | 77.6 |
| WRONG MEANING | 9 | 66.7 |
| **Task 2** | - | **Quality** |
| GRAMMATICAL | 83 | 3.01 |
| UNGRAMMATICAL | 17 | 2.88 |
| CORRECT MEANING | 88 | 3.22 |
| WRONG MEANING | 12 | 1.18 |

Table 3: Test results examining the effect of sentence meaning and grammaticality on task performance. Scores are shown for a sample of 100 triples split by whether the generated sentence is grammatical and whether it conveys the correct meaning of the triple.

## 4 Conclusion

We introduce a robust unsupervised method for commonsense knowledge base completion using the world knowledge of pre-trained language models. We develop a method for expressing knowledge triples as sentences. Using a bidirectional masked language model on these sentences, we can then estimate the weighted point-wise mutual information of a triple as a proxy for its validity. Though our approach performs worse on a held-out test set developed by Li et al. (2016), it does so without any previous exposure to the ConceptNet database, ensuring that this performance is not biased. In the future, we hope to explore whether this approach can be extended to mining facts that are not commonsense and to generating new commonsense knowledge outside of any given database of candidate triples. We also see potential benefit in the development of a more expansive set of evaluation methods for commonsense knowledge mining, which would strengthen the validity of our conclusions.

## Acknowledgments

This work was supported by NSF research award 1845664.

## References


Hirotugu Akaike. 1974. A new look at the statistical model identification. In *Selected Papers of Hirotugu Akaike*, pages 215–222. Springer.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Çelikyilmaz, and Yejin Choi. 2019. COMET: commonsense transformers for automatic knowledge graph construction. *CoRR*, abs/1906.05317.

Ernest Davis and Gary Marcus. 2015. Commonsense reasoning and commonsense knowledge in artificial intelligence. *Commun. ACM*, 58(9):92–103.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jonathan Gordon and Benjamin Van Durme. 2013. Reporting bias and knowledge acquisition. In *Proceedings of the 2013 workshop on Automated knowledge base construction*, pages 25–30. ACM.

Stanisław Jastrzebski, Dzmitry Bahdanau, Seyedarian Hosseini, Michael Noukhovitch, Yoshua Bengio, and Jackie Chi Kit Cheung. 2018. Commonsense mining as knowledge base completion? a study on the impact of novelty. *arXiv preprint arXiv:1804.09259*.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*.

Xiang Li, Aynaz Taheri, Lifu Tu, and Kevin Gimpel. 2016. Commonsense knowledge base completion. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1445–1455.

Hugo Liu and Push Singh. 2004. Conceptneta practical commonsense reasoning tool-kit. *BT technology journal*, 22(4):211–226.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and evaluation framework for deeper understanding of commonsense stories. *arXiv preprint arXiv:1604.01696*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1:8.

Itsumi Saito, Kyosuke Nishida, Hisako Asano, and Junji Tomita. 2018. Commonsense knowledge base completion and generation. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 141–150, Brussels, Belgium. Association for Computational Linguistics.

Roy Schwartz, Maarten Sap, Ioannis Konstas, Leila Zilles, Yejin Choi, and Noah A. Smith. 2017. The effect of different writing tasks on linguistic style: A case study of the ROC story cloze task. *CoRR*, abs/1702.01841.

Robert Speer and Catherine Havasi. 2012. Representing general relational knowledge in conceptnet 5. In *LREC*, pages 3679–3686.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509.

Trieu H. Trinh and Quoc V. Le. 2018. A simple method for commonsense reasoning. *CoRR*, abs/1806.02847.

Trieu H. Trinh and Quoc V. Le. 2019. Do language models have common sense?

Tom Young, Erik Cambria, Iti Chaturvedi, Hao Zhou, Subham Biswas, and Minlie Huang. 2018. Augmenting end-to-end dialogue systems with commonsense knowledge. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

# RNN Architecture Learning with Sparse Regularization

**Jesse Dodge**[♣] **Roy Schwartz**[♠◇] **Hao Peng**[♠] **Noah A. Smith**[♠◇]
[♣]Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA, USA
[◇]Allen Institute for Artificial Intelligence, Seattle, WA, USA
[♠]Paul G. Allen School of Computer Science & Engineering, University of Washington, Seattle, WA, USA
jessed@cs.cmu.edu roys@allenai.org {hapeng,nasmith}@cs.washington.edu

## Abstract

Neural models for NLP typically use large numbers of parameters to reach state-of-the-art performance, which can lead to excessive memory usage and increased runtime. We present a structure learning method for learning sparse, parameter-efficient NLP models. Our method applies group lasso to rational RNNs (Peng et al., 2018), a family of models that is closely connected to weighted finite-state automata (WFSAs). We take advantage of rational RNNs' natural grouping of the weights, so the group lasso penalty directly removes WFSA states, substantially reducing the number of parameters in the model. Our experiments on a number of sentiment analysis datasets, using both GloVe and BERT embeddings, show that our approach learns neural structures which have fewer parameters without sacrificing performance relative to parameter-rich baselines. Our method also highlights the interpretable properties of rational RNNs. We show that sparsifying such models makes them easier to visualize, and we present models that rely exclusively on as few as three WFSAs after pruning more than 90% of the weights. We publicly release our code.[1]

## 1 Introduction

State-of-the-art neural models for NLP are heavily parameterized, requiring hundreds of millions (Devlin et al., 2019) and even billions (Radford et al., 2019) of parameters. While over-parameterized models can sometimes be easier to train (Livni et al., 2014), they may also introduce memory problems on small devices and lead to increased carbon emission (Strubell et al., 2019; Schwartz et al., 2019).

In feature-based NLP, structured-sparse regularization, in particular the group lasso (Yuan and Lin, 2006), has been proposed as a method to reduce model size while preserving performance (Martins et al., 2011). But, in neural NLP, some of the most widely used models—LSTMs (Hochreiter and Schmidhuber, 1997) and GRUs (Cho et al., 2014)—do not have an obvious, intuitive notion of "structure" in their parameters (other than, perhaps, division into layers), so the use of structured sparsity at first may appear incongruous.

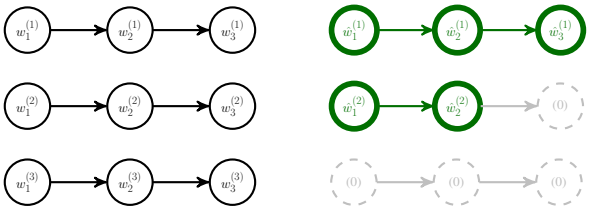


Figure 1: Our approach learns a sparse structure (right hand side) of a base rational RNN (left hand side) where each hidden unit corresponds to a WFSA (in this example, three hidden units, represented by the three rows). Grayed-out, dashed states are removed from the model, while retained states are marked in **bold green**.

In this paper we show that group lasso can be successfully applied to neural NLP models. We focus on a family of neural models for which the hidden state exhibits a natural structure: rational RNNs (Peng et al., 2018). In a rational RNN, the value of each hidden dimension is the score of a weighted finite-state automaton (WFSA) on (a prefix of) the input vector sequence. This property offers a natural grouping of the transition function parameters for each WFSA. As shown by Schwartz et al. (2018) and Peng et al. (2018), a variety of state-of-the-art neural architectures are rational (Lei et al., 2017; Bradbury et al., 2017; Foerster et al., 2017, *inter alia*), so learning parameter-efficient rational RNNs is of practical value. We

[1] https://github.com/dodgejesse/sparsifying_regularizers_for_RRNNs



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1179–1184, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

also take advantage of the natural interpretation of rational RNNs as "soft" patterns (Schwartz et al., 2018).

We apply a group lasso penalty to the WFSA parameters of rational RNNs during training, where each group is comprised of the parameters associated with one state in one WFSA (Fig. 1; §2). This penalty pushes the parameters in some groups to zero, effectively eliminating them, and making the WFSA smaller. When all of the states for a given WFSA are eliminated, the WFSA is removed entirely, so this approach can be viewed as learning the number of WFSAs (i.e., the RNN hidden dimension) as well as their size. We then retain the sparse structure, which results in a much smaller model in terms of parameters.

We experiment with four text classification benchmarks (§3), using both GloVe and BERT embeddings. As we increase the regularization strength, we end up with smaller models. These models have a better tradeoff between the number of parameters and model performance compared to setting the number of WFSAs and their lengths by hand or using hyperparameter search. In almost all cases, our approach results in models with fewer parameters and similar or better performance compared to our baselines.

In contrast to neural architecture search (Jozefowicz et al., 2015; Zoph and Le, 2017), which can take several GPU years to learn an appropriate neural architecture, our approach requires only two training runs: one to learn the structure, and the other to estimate its parameters. Other approaches either ignore the structure of the model and only look at the value of individual weights (Liu et al., 2019; LeCun et al., 1990; Lee et al., 2019; Frankle and Carbin, 2019) or only use high-level structures like the number of layers of the network (Wen et al., 2016; Scardapane et al., 2017; Gordon et al., 2018).

Finally, our approach touches on another appealing property of rational RNNs—their interpretability. Each WFSA captures a "soft" version of patterns like "such a great X", and can be visualized as such (Schwartz et al., 2018). By retaining a small number of WFSAs, model structures learned using our method can be visualized succinctly. In §4 we show that some of our sentiment analysis models rely exclusively on as few as *three* WFSAs.[2]

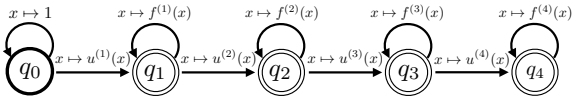


Figure 2: A 4-gram WFSA, from which we derive the rational RNN (§2). The rational RNN's hidden states correspond to a set of WFSAs, each separately parameterized. We apply group lasso to each WFSA.

## 2 Method

We describe the proposed method. At a high level, we follow the standard practice for using $\ell_1$ regularization for sparsification (Wen et al., 2016):

1. Fit a model on the training data, with the group lasso regularizer added to the loss during training (the parameters associated with one state comprise one group).
2. After convergence, eliminate the states whose parameters are zero.
3. Finetune the resulting, smaller model, by minimizing the unregularized loss with respect to its parameters.

In this work, we assume a single layer rational RNN, but our approach is equally applicable to multi-layer models. For clarity of the discussion, we start with a one-dimensional rational RNN (i.e., one based on a single WFSA only). We then generalize to the $d$-dimensional case (computing the scores of $d$ WFSAs in parallel).

**Rational recurrent networks** Following Peng et al. (2018), we parameterize the transition functions of WFSAs with neural networks, such that each transition (main path or self loop) defines a weighted function over the input word vector. We consider a 5-state WFSA, diagrammed in Fig. 2.

A path starts at $q_0$; at least four tokens must be consumed to reach $q_4$, and in this sense it captures 4-gram "soft" patterns (Peng et al., 2018; Schwartz et al., 2018). In addition to $q_4$, we also designate $q_1$, $q_2$, and $q_3$ as final states, allowing for the interpolation between patterns of different lengths.[3] The self-loop transitions over $q_1$, $q_2$, $q_3$, and $q_4$ aim to allow, but downweight, nonconsecutive patterns, as the self-loop transition functions yield values between 0 and 1 (using a sigmoid function). The recurrent function is equivalent to applying the Forward dynamic programming algorithm (Baum and Petrie, 1966).

[2] That is, a rational RNN with hidden size 3.

[3] We found this to be more stable than using only $q_4$.

**Promoting sparsity with group lasso** We aim to learn a sparse rational model with fewer WFSA states. This can be achieved by penalizing the parameters associated with a given state, specifically the parameters associated with *entering* that state, by a transition from another state or a self-loop on that state. For example, the parameters of the WFSA in Fig. 2 (excluding the word embedding parameters) are assigned to four nonoverlapping groups, one for each non-starting state.

During training, the regularization term will push all parameters toward zero, and some will converge close to zero.[4] After convergence, we remove groups for which the $\ell_2$ norm falls below $\epsilon$.[5] The resulting smaller model is then finetuned by continuing training without regularizing. With our linear-structured WFSA, zeroing out the group associated with a state in the middle effectively makes later states inaccessible. While our approach offers no guarantee to remove states from the end first (thus leaving no unreachable states), in our experiments it always did so.

**$d$-dimensional Case** To construct a rational RNN with $d$ WFSAs (a $d$-dimensional model), we stack $d$ one-dimensional models, each of them separately parameterized. The parameters of a $d$-dimensional rational model derived from the WFSA in Fig. 2 are organized into $4d$ groups, four for each dimension. Since there is no direct interaction between different dimensions (e.g., through an affine transformation), group lasso sparsifies each dimension/WFSA independently. Hence the resulting rational RNN can consist of WFSAs of different sizes, the number of which could be smaller than $d$ if any of the WFSAs have all states eliminated.

One can treat the numbers and sizes of WFSAs as hyperparameters (Oncina et al., 1993; Ron et al., 1994; de la Higuera, 2010; Schwartz et al., 2018). By eliminating states from WFSAs with group lasso, we learn the WFSA structure *while* fitting the models' parameters, reducing the number of training cycles by reducing the number of tunable hyperparameters.

[4]There are optimization methods that achieve "strong" sparsity (Parikh and Boyd, 2013), where some parameters are exactly set to zero during training. Recent work has shown these approaches can converge in nonconvex settings (Reddi et al., 2016), but our experiments found them to be unstable.

[5]We use 0.1. This threshold was lightly tuned in preliminary experiments on the validation set and found to reliably remove those parameters which converged around zero without removing others.

## 3 Experiments

We run sentiment analysis experiments. We train the rational RNN models (§2) with group lasso regularization, using increasingly large regularization strengths, resulting in increasingly compact models. As the goal of our experiments is to demonstrate the ability of our approach to reduce the number of parameters, we only consider rational baselines: the same rational RNNs trained without group lasso.[6] We manually tune the number and sizes of the baselines WFSAs, and then compare the tradeoff curve between model size and accuracy. We describe our experiments below. For more details, see Appendix A.

**Data** We experiment with the Amazon reviews binary sentiment classification dataset (Blitzer et al., 2007), composed of 22 product categories. We examine the standard dataset (**original_mix**) comprised of a mixture of data from the different categories (Johnson and Zhang, 2015).[7] We also examine three of the largest individual categories as separate datasets (**kitchen**, **dvd**, and **books**), following Johnson and Zhang (2015). The three category datasets do *not* overlap with each other (though they do with **original_mix**), and are significantly different in size (see Appendix A), so we can see how our approach behaves with different amounts of training data.

**Implementation details** To classify text, we concatenate the scores computed by each WFSA, then feed this $d$-dimensional vector of scores into a linear binary classifier. We use log loss. We experiment with both type-level word embeddings (GloVe.6B.300d; Pennington et al., 2014) and contextual embeddings (BERT large; Devlin et al., 2019).[8] In both cases, we keep the embeddings fixed, so the vast majority of the learnable parameters are in the WFSAs. We train models using GloVe embeddings on all datasets. Due to memory constraints we evaluate BERT embeddings (frozen, not fine-tuned) only on the smallest

[6]Rational RNNs have shown strong performance on the dataset we experiment with: a 2-layer rational model with between 100–300 hidden units obtained 92.7% classification accuracy, substantially outperforming an LSTM baseline (Peng et al., 2018). The results of our models, which are single-layered and capped at 24 hidden units, are not directly comparable to these baselines, but are still within two points of the best result from that paper.

[7]http://riejohnson.com/cnn_data.html

[8]https://github.com/huggingface/pytorch-pretrained-BERT

dataset (**kitchen**).

**Baselines** As baselines, we train five versions of each rational architecture without group lasso, using the same number of WFSAs as our regularized models (24 for GloVe, 12 for BERT). Four of the baselines each use the same number of transitions for all WFSAs (1, 2, 3, and 4, corresponding to 2–5 states, and to 24, 48, 72, and 96 total transitions). The fifth baseline has an equal mix of all lengths (6 WFSAs of each size for GloVe, leading to 60 total transitions, and 3 WFSAs of each size for BERT, leading to 30 total transitions).

Each transition in our model is independently parameterized, so the total number of transitions linearly controls the number of learnable parameters (in addition to the parameters in the embedding layer).

**Results** Fig. 3 shows our classification test accuracy as a function of the total number of WFSA transitions in the model. We first notice that, as expected, the performance of our unregularized baselines improves as models are trained with more transitions (i.e., more parameters).

Compared to the baselines, training with group lasso provides a better tradeoff between performance and number of transitions. In particular, our heavily regularized models perform substantially better than the unigram baselines, gaining between 1–2% absolute improvements in four out of five cases. As our regularization strength decreases, we naturally gain less compared to our baselines, although still similar or better than the best baselines in four out of five cases.

## 4 Visualization

Using our method with a high regularization strength, the resulting sparse structures often contain only a handful of WFSAs. In such cases, building on the interpretability of individual WFSAs, we are able visualize every hidden unit, i.e., the entire model. To visualize a single WFSA $\mathcal{B}$, we follow Schwartz et al. (2018) and compute the score of $\mathcal{B}$ on every phrase in the training corpus. We then select the top and bottom scoring phrases for $\mathcal{B}$,[9] and get a prototype-like description of the pattern representing $\mathcal{B}$.[10]

| | | transition$_1$ | transition$_2$ | transition$_3$ |
|---|---|---|---|---|
| Patt. 1 | Top | not | worth | the time$_{SL}$ </s> |
| | | not | worth | the 30$_{SL}$ </s> |
| | | not | worth | it$_{SL}$ </s> |
| | | not | worth | it$_{SL}$ </s> |
| | Bottom | extremely | pleased | $\ldots_{SL}$ </s> |
| | | highly | pleased | $\ldots_{SL}$ </s> |
| | | extremely | pleased | $\ldots_{SL}$ </s> |
| | | extremely | pleased | $\ldots_{SL}$ </s> |
| Patt. 2 | Top | miserable | </s> | |
| | | miserable | $\ldots_{SL}$ </s> | |
| | | miserable | $\ldots_{SL}$ </s> | |
| | | returned | </s> | |
| | Bottom | superb | </s> | |
| | | superb | </s> | |
| | | superb | </s> | |
| | | superb | choice$_{SL}$ </s> | |
| Patt. 3 | Top | bad | $\ldots_{SL}$ ltd | $\ldots_{SL}$ buyer |
| | | bad | $\ldots_{SL}$ ltd | $\ldots_{SL}$ buyer |
| | | horrible | $\ldots_{SL}$ hl4040cn | $\ldots_{SL}$ expensive |
| | | left | $\ldots_{SL}$ ltd | $\ldots_{SL}$ lens |
| | Bottom | favorite | $\ldots_{SL}$ ltd | $\ldots_{SL}$ lens |
| | | really | $\ldots_{SL}$ ltd | $\ldots_{SL}$ buyer |
| | | really | $\ldots_{SL}$ ltd | $\ldots_{SL}$ buyer |
| | | best | $\ldots_{SL}$ hl4040cn | $\ldots_{SL}$ expensive |

Table 1: Visualization of a sparse rational RNN trained on **original_mix** containing only 3 WFSAs. For each WFSA (i.e., pattern), we show the 4 top and bottom scoring phrases in the training corpus with this WFSA. Each column represents one main-path transition, plus potential self-loops preceding it (marked like this$_{SL}$). "$\ldots_{SL}$" marks more than 2 self loops. "</s>" marks an end-of-document token.

Table 1 visualizes a sparse rational RNN trained on **original_mix** with only *three* WFSAs, (8 main-path transitions in total).[11] The table shows that looking at the top scores of each WFSA, two of the patterns respectively capture the phrases "*not worth X </s>*" and "*miserable/returned X </s>*". Pattern 3 is not as coherent, but most examples do contain sentiment-bearing words such as *bad, horrible,* or *best*. This might be the result of the tuning process of the sparse rational structure simply learning a collection of words, rather than coherant phrases. As a result, this WFSA is treated as a unigram pattern rather than a trigram. The lowest scoring phrases show a similar trend. Appendix B shows the same visualization for another sparse rational RNN containing only four WFSAs and 11 main-path transitions, trained with BERT embeddings.

We observe another interesting trend: two of the three patterns prefer expressions that appear near the end of the document. This could result from the nature of the datasets (e.g., many reviews end

[9] As each WFSA score is used as a feature that is fed to a linear classifier, negative scores are also meaningful.

[10] While the WFSA scores are the sum of all paths deriving a document (plus-times semiring), here we search for the max (or min) scoring one. Despite the mismatch, a WFSA scores every possible path, and thus the max/min scoring path selection is still valid. As our examples show, many of these extracted paths are meaningful.

[11] The test performance of this model is 88%, 0.6% absolute below the average of the five models reported in Fig. 3.

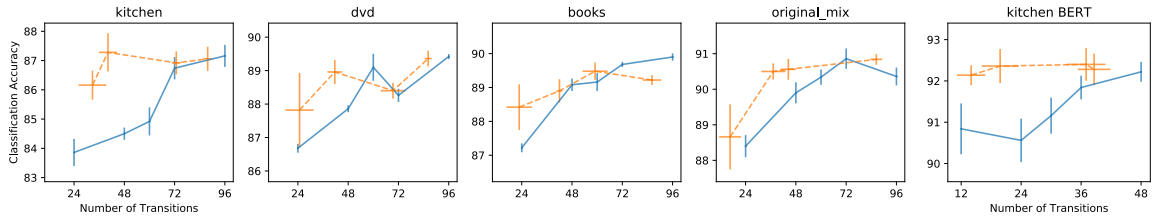


Figure 3: Text classification with GloVe embeddings (four leftmost graphs) and BERT (rightmost): accuracy ($y$-axis) vs. number of WFSA transitions ($x$-axis). **Higher** and to the **left** is better. Our method (dashed orange line, varying regularization strength) provides a better tradeoff than the baseline (solid blue line, directly varying the number of transitions). Vertical lines encode one standard deviation for accuracy, while horizontal lines encode one standard deviation in the number of transitions (applicable only to our method).

with a summary, containing important sentiment information), and/or our rational models' recency preference. More specifically, the first self loop has weight 1, and hence the model is not penalized for taking self loops before the match; in contrast, the weights of the last self loop take values in $(0, 1)$ due to the sigmoid, forcing a penalty for earlier phrase matches.[12]

## 5 Conclusion

We presented a method for learning parameter-efficient RNNs. Our method applies group lasso regularization on rational RNNs, which are strongly connected to weighted finite-state automata, and thus amenable to learning with structured sparsity. Our experiments on four text classification datasets, using both GloVe and BERT embeddings, show that our sparse models provide a better performance/model size tradeoff. We hope our method will facilitate the development of "thin" NLP models, that are faster, consume less memory, and are interpretable (Schwartz et al., 2019).

## Acknowledgments

This work was completed while the first author was an intern at the Allen Institute for Artificial Intelligence. The authors thank Pradeep Dasigi, Gabriel Stanovsky, and Elad Eban for their discussion. In addition, the authors thank the members of the Noahs ARK group at the University of Washington, the researchers at the Allen Institute for Artificial Intelligence, and the anonymous reviewers for their valuable feedback. This work was supported in part by a hardware gift from NVIDIA Corporation.

[12] Changing this behavior could be easily done by fixing the final self-loop to 1 as well.

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR*.

Leonard E. Baum and Ted Petrie. 1966. Statistical inference for probabilistic functions of finite state Markov chains. *The Annals of Mathematical Statistics*, 37(6).

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, Bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proc. of ACL*.

James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. 2017. Quasi-recurrent neural network. In *Proc. of ICLR*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proc. of EMNLP*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proc. of NAACL*.

Jakob N. Foerster, Justin Gilmer, Jan Chorowski, Jascha Sohl-Dickstein, and David Sussillo. 2017. Intelligible language modeling with input switched affine networks. In *Proc. of ICML*.

Jonathan Frankle and Michael Carbin. 2019. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *Proc. of ICLR*.

Ariel Gordon, Elad Eban, Ofir Nachum, Bo Chen, Hao Wu, Tien-Ju Yang, and Edward Choi. 2018. MorphNet: Fast & simple resource-constrained structure learning of deep networks. In *Proc. of CVPR*.

Colin de la Higuera. 2010. *Grammatical Inference: Learning Automata and Grammars*. Cambridge University Press.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8).

Rie Johnson and Tong Zhang. 2015. Effective use of word order for text categorization with convolutional neural networks. In *Proc. of NAACL*.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proc. of ICML*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proc. of ICLR*.

Yann LeCun, John S. Denker, and Sara A. Solla. 1990. Optimal brain damage. In *NeurIPS*.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. 2019. SNIP: Single-shot network pruning based on connection sensitivity. In *Proc. of ICLR*.

Tao Lei, Yu Zhang, and Yoav Artzi. 2017. Training RNNs as fast as CNNs. arXiv:1709.02755.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2019. DARTS: Differentiable architecture search. In *Proc. of ICLR*.

Roi Livni, Shai Shalev-Shwartz, and Ohad Shamir. 2014. On the computational efficiency of training neural networks. In *NeurIPS*.

André F. T. Martins, Noah A. Smith, Mario Figueiredo, and Pedro Aguiar. 2011. Structured sparsity in structured prediction. In *Proc. of EMNLP*.

José Oncina, Pedro García, and Enrique Vidal. 1993. Learning subsequential transducers for pattern recognition interpretation tasks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15:448–458.

Neal Parikh and Stephen P. Boyd. 2013. *Proximal Algorithms*. Foundations and Trends in Optimization.

Hao Peng, Roy Schwartz, Sam Thomson, and Noah A. Smith. 2018. Rational recurrences. In *Proc. of EMNLP*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proc. of EMNLP*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Sashank J. Reddi, Suvrit Sra, Barnabs Pczos, and Alexander J. Smola. 2016. Proximal stochastic methods for nonsmooth nonconvex finite-sum optimization. In *NeurIPS*.

Dana Ron, Yoram Singer, and Naftali Tishby. 1994. Learning probabilistic automata with variable memory length. In *Proc. of COLT*.

Simone Scardapane, Danilo Comminiello, Amir Hussain, and Aurelio Uncini. 2017. Group sparse regularization for deep neural networks. *Neurocomputing*, 241.

Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. 2019. Green AI. arXiv:1907.10597.

Roy Schwartz, Sam Thomson, and Noah A. Smith. 2018. SoPa: Bridging CNNs, RNNs, and weighted finite-state machines. In *Proc. of ACL*.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proc. of ACL*.

Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. 2016. Learning structured sparsity in deep neural networks. In *NeurIPS*.

Ming Yuan and Yi Lin. 2006. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1).

Barret Zoph and Quoc V Le. 2017. Neural architecture search with reinforcement learning. In *Proc of ICLR*.

# Analytical Methods for Interpretable Ultradense Word Embeddings

**Philipp Dufter, Hinrich Schütze**
Center for Information and Language Processing (CIS)
LMU Munich, Germany
philipp@cis.lmu.de

## Abstract

Word embeddings are useful for a wide variety of tasks, but they lack interpretability. By rotating word spaces, interpretable dimensions can be identified while preserving the information contained in the embeddings without any loss. In this work, we investigate three methods for making word spaces interpretable by rotation: Densifier (Rothe et al., 2016), linear SVMs and DensRay, a new method we propose. In contrast to Densifier, DensRay can be computed in closed form, is hyperparameter-free and thus more robust than Densifier. We evaluate the three methods on lexicon induction and set-based word analogy. In addition we provide qualitative insights as to how interpretable word spaces can be used for removing gender bias from embeddings.

## 1 Introduction

Distributed representations for words have been of interest in natural language processing for many years. Word embeddings have been particularly effective and successful. On the downside, embeddings are generally not interpretable. But interpretability is desirable for several reasons. i) Semantically or syntactically similar words can be extracted: e.g., for lexicon induction. ii) Interpretable dimensions can be used to evaluate word spaces by examining which information is covered by the embeddings. iii) Computational advantage: for a high-quality sentiment classifier only a couple of dimensions of a high-dimensional word space are relevant. iv) By removing interpretable dimensions one can remove unwanted information (e.g., gender bias). v) Most importantly, interpretable embeddings support the goal of interpretable deep learning models.

Orthogonal transformations have been of particular interest in the literature. The reason is twofold: under the assumption that existing word embeddings are of high-quality one would like to preserve the original embedding structure by using orthogonal transformations (i.e., preserving original distances). Park et al. (2017) provide evidence that rotating existing dense word embeddings achieves the best performance across a range of interpretability tasks.

In this work we modify the objective function of Densifier (Rothe et al., 2016) such that a closed form solution becomes available. We call this method DensRay. Following Amir et al. (2015) we compute simple linear SVMs, which we find to perform surprisingly well. We compare these methods on the task of lexicon induction.

Further, we show how interpretable word spaces can be applied to other tasks: first we use interpretable word spaces for debiasing embeddings. Second we show how they can be used for solving the set-based word analogy task. To this end, we introduce the set-based method IntCos, which is closely related to LRCos introduced by Drozd et al. (2016). We find IntCos to perform comparable to LRCos, but to be preferable for analogies which are hard to solve.

Our contributions are: **i)** We modify Densifier's objective function and derive an analytical solution for computing interpretable embeddings. **ii)** We show that the analytical solution performs as well as Densifier but is more robust. **iii)** We provide evidence that simple linear SVMs are best suited for the task of lexicon induction. **iv)** We demonstrate how interpretable embedding spaces can be used for debiasing embeddings and solving the set-based word analogy task. The source code of our experiments is available.[1]

[1] https://github.com/pdufter/densray



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1185–1191, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

## 2 Methods

### 2.1 Notation

We consider a vocabulary $V := \{v_1, v_2, ..., v_n\}$ together with an embedding matrix $E \in \mathbb{R}^{n \times d}$ where $d$ is the embedding dimension. The $i$th row of $E$ is the vector $e_i$.[2] We require an annotation for a specific linguistic feature (e.g., sentiment) and denote this annotation by $l : V \to \{-1, 1\}$. The objective is to find an orthogonal matrix $Q \in \mathbb{R}^{d \times d}$ such that $EQ$ is interpretable, i.e., the values of the first $k$ dimensions correlate well with the linguistic feature. We refer to the first $k$ dimensions as interpretable ultradense word space. We interpret $x \in \mathbb{R}^n$ as a column vector and $x^\intercal$ as a row vector. Further, we normalize all word embeddings with respect to the euclidean norm.

### 2.2 DensRay

Throughout this section $k = 1$. Given a linguistic signal $l$ (e.g., sentiment), consider $L_= := \{(v, w) \in V \times V |\ l(v) = l(w)\}$, and analogously $L_{\neq}$. We call $d_{vw} := e_v - e_w$ a difference vector.

Densifier (Rothe et al., 2016) solves the following optimization problem,

$$\max_q \sum_{(v,w) \in L_{\neq}} \alpha_{\neq} \left\| q^\intercal d_{vw} \right\|_2 - \sum_{(v,w) \in L_=} \alpha_= \left\| q^\intercal d_{vw} \right\|_2 ,$$

subject to $q^\intercal q = 1$ and $q \in \mathbb{R}^d$. Further $\alpha_{\neq}, \alpha_= \in [0, 1]$ are hyperparameters. We now modify the objective function: we use the squared euclidean norm instead of the euclidean norm, something that is frequently done in optimization to simplify the gradient. The problem becomes then

$$\max_q \sum_{(v,w) \in L_{\neq}} \alpha_{\neq} \left\| q^\intercal d_{vw} \right\|_2^2 - \sum_{(v,w) \in L_=} \alpha_= \left\| q^\intercal d_{vw} \right\|_2^2 . \quad (1)$$

Using $\|x\|_2^2 = x^\intercal x$ together with associativity of the matrix product we can simplify to

$$\max_q q^\intercal \Big( \alpha_{\neq} \sum_{(v,w) \in L_{\neq}} d_{vw} d_{vw}^\intercal - \alpha_= \sum_{(v,w) \in L_=} d_{vw} d_{vw}^\intercal \Big) q \quad (2)$$
$$=: \max_q q^\intercal A q \quad \text{subject to } q^\intercal q = 1.$$

[2] We denote the vector corresponding to a word $w$ by $e_w$.

Thus we aim to maximize the Rayleigh quotient of $A$ and $q$. Note that $A$ is a real symmetric matrix. Then it is well known that the eigenvector belonging to the maximal eigenvalue of $A$ solves the above problem (cf. Horn et al. (1990, Section 4.2)). We call this analytical solution **DensRay**.

A second dimension that is orthogonal to the first dimension and encodes the linguistic features second strongest is given by the eigenvector corresponding to the second largest eigenvalue. The matrix of $k$ eigenvectors of $A$ ordered by the corresponding eigenvalues yields the desired matrix $Q$ (cf. Horn et al. (1990, Section 4.2)) for $k > 1$. Due to $A$ being a real symmetric matrix, $Q$ is always orthogonal.

### 2.3 Comparison to Densifier

We have shown that DensRay is a closed form solution to our new formalization of Densifier. This formalization entails differences.

**Case $k > 1$.** While both methods – Densifier and DensRay – yield ultradense $k$ dimensional subspaces. While we show that the spaces are comparable for $k = 1$ we leave it to future work to examine how the subspaces differ for $k > 1$.

**Multiple linguistic signals.** Given multiple linguistic features, Densifier can obtain a single orthogonal transformation simultaneously for all linguistic features with chosen dimensions reserved for different features. DensRay can encode multiple linguistic features in one transformation only by iterative application.

**Optimization.** Densifier is based on solving an optimization problem using stochastic gradient descent with iterative orthogonalization of $Q$. DensRay, in contrast, is an analytical solution. Thus we expect DensRay to be more robust, which is confirmed by our experiments.

### 2.4 Geometric Interpretation

Assuming we normalize the vectors $d_{vw}$ one can interpret Eq. 1 as follows: we search for a unit vector $q$ such that the square of the cosine similarity with $d_{vw}$ is large if $(v, w) \in L_{\neq}$ and small if $(v, w) \in L_=$. Thus, we identify dimensions that are parallel/orthogonal to difference vectors of words belonging to different/same classes. It seems reasonable to consider the average cosine similarity. Thus if $n_=$, $n_{\neq}$ is the number of elements in $L_=$, $L_{\neq}$ one can choose $\alpha_{\neq} = n_{\neq}^{-1}$ and $\alpha_= = n_=^{-1}$.

## 3 Lexicon Induction

We show that DensRay and Densifier indeed perform comparably using the task of lexicon induction. We adopt Rothe et al. (2016)'s experimental setup. We also use Rothe et al. (2016)'s code for Densifier. Given a word embedding space and a sentiment/concreteness dictionary (binary or continuous scores where we binarize continuous scores using the median), we identify a one-dimensional interpretable subspace. Subsequently we use the values along this dimension to predict a score for unseen words and report Kendall's $\tau$ rank correlation with the gold scores.

To ensure comparability across methods we have redone all experiments in the same setting: we deduplicated lexicons, removed a potential train/test overlap and ignored neutral words in the lexicons. We set $\alpha_{\neq} = \alpha_{=} = 0.5$ to ensure comparability between Densifier and DensRay.

Additionally we report results created by linear SVM/SVR inspired be their good performance as demonstrated by Amir et al. (2015). While they did not use linear kernels, we require linear kernels to obtain interpretable dimensions. Naturally the normal vector of the hyperplane in SVMs/SVRs reflects an interpretable dimension. An orthogonal transformation can be computed by considering a random orthogonal basis of the null space of the interpretable dimension.

Table 1 shows results. As expected the performance of Densifier and DensRay is comparable (macro mean deviation of 0.001). We explain slight deviations between the results with the slightly different objective functions of DensRay and Densifier. In addition, the re-orthogonalization used in Densifier can result in an unstable training process. Figure 1 assesses the stability by reporting mean and standard deviation for the concreteness task (BWK lexicon). We varied the size of the training lexicon as depicted on the x-axis and sampled 40 subsets of the lexicon with the prescribed size. For the sizes 512 and 2048 Densifier shows an increased standard deviation. This is because there is at least one sample for which the performance significantly drops. Removing the re-orthogonalization in Densifier prevents the drop and restores performance. Recent work (Zhao and Schütze, 2019) also finds that replacing the orthogonalization with a regularization is reasonable in certain circumstances. Given that DensRay and Densifier yield the same performance and DensRay is a stable closed form solution always yielding a orthogonal transformation we conclude that DensRay is preferable.

Surprisingly, simple linear SVMs perform best in the task of lexicon induction. SVR is slightly better when continuous lexica are used for training (line 8). Note that the eigendecomposition used in DensRay yields a basis with dimensions ordered by their correlation with the linguistic feature. An SVM can achieve this only by iterated application.

| | Task | Emb. | Lex. (Train) | Lex. (Test) | Dens. | DensRay | SVR | SVM |
|---|---|---|---|---|---|---|---|---|
| 1 | sent | CZ | SubLex | SubLex | 0.546 | 0.549 | **0.585** | 0.585 |
| 2 | sent | DE | GermanPC | GermanPC | 0.636 | 0.631 | 0.674 | **0.677** |
| 3 | sent | ES | fullstrength | fullstrength | 0.541 | 0.546 | 0.571 | **0.576** |
| 4 | sent | FR | FEEL | FEEL | 0.469 | 0.471 | 0.555 | **0.565** |
| 5 | sent | EN | WHM | WHM | 0.623 | 0.623 | **0.627** | 0.625 |
| 6 | sent | EN(t) | WHM | SE Trial* | 0.624 | 0.621 | 0.618 | **0.637** |
| 7 | sent | EN(t) | WHM | SE Test* | 0.600 | 0.608 | 0.619 | **0.636** |
| 8 | conc | EN | BWK* | BWK* | 0.599 | 0.602 | **0.655** | 0.641 |
| 9 | | | Macro Mean | | 0.580 | 0.581 | 0.613 | **0.618** |

Table 1: Results on lexicon induction. Numbers are Kendall $\tau$ rank correlation. For details on the resources see Table 2 and (Rothe et al., 2016). Bold: best result across methods. $^{\star}$: continuous lexicon.

| Name | Description |
|---|---|
| CZ, DE, ES | Czech, German, Spanish embeddings by (Rothe et al., 2016) |
| FR | French frWac embeddings (Fauconnier, 2015) |
| EN | English GoogleNews embeddings (Mikolov et al., 2013) |
| EN(t) | English Twitter Embeddings (Rothe et al., 2016) |

| Name | Description |
|---|---|
| SubLex | Czech sentiment lexicon (Veselovská and Bojar, 2013) |
| GermanPC | German sentiment lexicon (Waltinger, 2010) |
| fullstrength | Spanish sentiment lexicon (Perez-Rosas et al., 2012) |
| FEEL | French sentiment lexicon (Abdaoui et al., 2017) |
| WHM | English sentiment lexicon; combination of MPQA (Wilson et al., 2005), Opinion Lexicon (Hu and Liu, 2004) and NRC emotion lexcion (Mohammad and Turney, 2013) |
| SE | Semeval 2015 Task 10E shared task data (Rosenthal et al., 2015) |
| BWK | English concreteness lexicon (Brysbaert et al., 2014) |

Table 2: Overview of resources for lexicon induction. The setup is identical to (Rothe et al., 2016).

## 4 Removing Gender Bias

Word embeddings are well-known for encoding prevalent biases and stereotypes (cf. Bolukbasi et al. (2016)). We demonstrate qualitatively that by identifying an interpretable gender dimension and subsequently removing this dimension, one can remove parts of gender information that potentially could cause biases in downstream processing. Given the original word space $E$ we consider the interpretable space $E' := EQ$, where $Q$ is computed using DensRay. We denote by $E_{\cdot,-1} \in \mathbb{R}^{n\times(d-1)}$ the word space with removed first dimension and call it the "complement" space. We expect $E_{\cdot,-1}$ to be a word space with less gender bias.

To examine this approach qualitatively we use

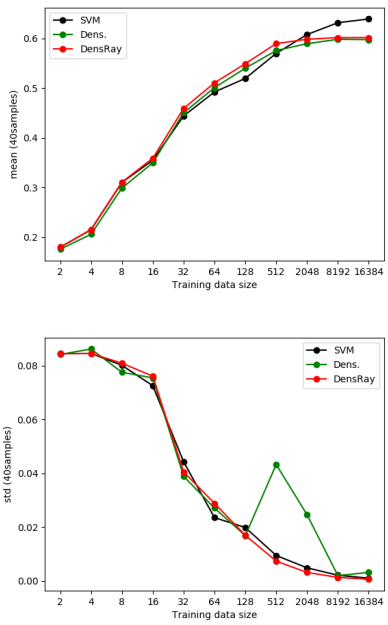


Figure 1: Mean (top) and standard deviation (bottom) of the performance across 40 samples of the training lexicon with varying sizes. Performed on the English concreteness task (line 8 in Table 1). SVR performs similar to SVM and is omitted for clarity.

a list of occupation names[3] by Bolukbasi et al. (2016) and examine the cosine similarities of occupations with the vectors of "man" and "woman". Figure 2 shows the similarities in the original space $E$ and debiased space $E_{\cdot,-1}$. One can see the similarities are closer to the identity (i.e., same distance to "man" and "woman") in the complement space. To identify occupations with the greatest bias, Table 3 lists occupations for which $\text{sim}(e_w, e_{\text{man}}) - \text{sim}(e_w, e_{\text{woman}})$ is largest/smallest. One can clearly see a debiasing effect when considering the complement space. Extending this qualitative study to a more rigorous quantitative evaluation is part of future work.

| | Original Space | | | Complement Space | | |
|---|---|---|---|---|---|---|
| | | man | woman | | man | woman |
| female bias | actress | 0.23 | 0.46 | lawyer | 0.16 | 0.27 |
| | businesswoman | 0.32 | 0.53 | ambassador | 0.07 | 0.17 |
| | registered_nurse | 0.12 | 0.33 | attorney | 0.05 | 0.15 |
| | housewife | 0.34 | 0.55 | legislator | 0.26 | 0.36 |
| | homemaker | 0.22 | 0.40 | minister | 0.10 | 0.20 |
| | ... | | | | | |
| male bias | hitman | 0.41 | 0.27 | captain | 0.31 | 0.24 |
| | gangster | 0.34 | 0.20 | marksman | 0.29 | 0.21 |
| | skipper | 0.27 | 0.11 | maestro | 0.28 | 0.20 |
| | marksman | 0.31 | 0.14 | hitman | 0.40 | 0.32 |
| | maestro | 0.30 | 0.12 | skipper | 0.25 | 0.17 |

Table 3: Top 5 occupations that exhibit the greatest bias (measured by difference in cosine similarity). Numbers indicate cosine similarity between word vectors.

[3] https://github.com/tolga-b/debiaswe/blob/master/data/professions.json

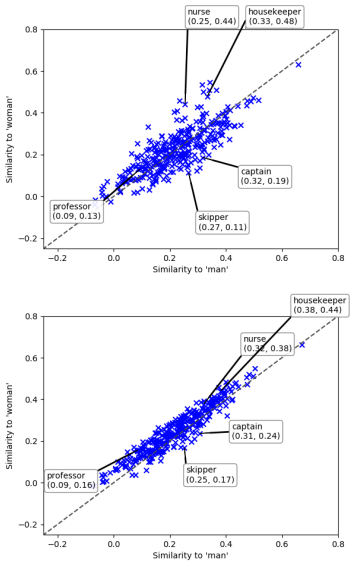


Figure 2: Similarities of occupation vectors with the vectors of man and woman. Top shows the original word space and bottom the word space with removed gender dimension.

## 5 Word Analogy

In this section we use interpretable word spaces for set-based word analogy. Given a list of analogy pairs $[(a, a'), (b, b'), (c, c'), \ldots]$ the task is to predict $a'$ given $a$. Drozd et al. (2016) provide a detailed overview over different methods, and find that their method LRCos performs best.

LRCos assumes two classes: all left elements of a pair ("left class") and all right elements ("right class"). They train a logistic regression (LR) to differentiate between these two classes. The predicted score of the LR multiplied by the cosine similarity in the word space is their final score. Their prediction for $a'$ is the word with the highest final score.

We train the classifier on all analogy pairs except for a single pair for which we then obtain the predicted score. In addition we ensure that no word belonging to the test analogy is used during training (splitting the data only on word analogy pairs is not sufficient).

Inspired by LRCos we use interpretable word spaces for approaching word analogy: we train DensRay or an SVM to obtain interpretable embeddings $E' = EQ$ using the class information as reasoned above. We use a slightly different notation in this section: for a word $w$ the $i$th component of its embedding is given by $E_{w,i}$. Therefore we denote as $E_{\cdot,1}$ the first column of $E'$ (i.e., the

most interpretable dimension). We min-max normalize $E_{\cdot,1}$ such that words belonging to the right class have a high value (i.e., we flip the sign if necessary). For a query word $a$ we now want to identify the corresponding $a'$ by solving

$$\hat{a} = \arg\max_{v \in V} \text{norm}(E_{v,1}) \, \text{sim}(E_{a,\cdot}, E_{v,\cdot})$$

where *sim* computes the cosine similarity.

Given the result from §4 we extend the above method by computing the cosine similarity in the orthogonal complement, i.e., $\text{sim}(E_{a,-1}, E_{v,-1})$. We call this method **IntCos** (INTerpretable, COSine). Depending on the space used for computing the cosine similarity add the word "Original" or "Complement".

We evaluate this method across two analogy datasets. These are the Google Analogy Dataset (GA) (Mikolov et al., 2013) and BATS (Drozd et al., 2016). As embeddings spaces we use Google News Embeddings (GN) (Mikolov et al., 2013) and FastText subword embeddings (FT) (Bojanowski et al., 2017). We consider the first 80k word embeddings from each space.

Table 4 shows the results. The first observation is that there is no clear winner. IntCos Original performs comparably to LRCos with slight improvements for GN/BATS: here the classes are widespread and exhibit low cosine similarity (IntraR and IntraL), which makes them harder to solve. IntCos Complement maintains performance for GN/BATS and is beneficial for Derivational analogies on GN. For most other analogies it harms performance.

Within IntCos Original it is favorable to use DensRay as it gives slight performance improvements. Especially for harder analogies, where interclass similarity is high and intraclass similarities are low (e.g., in GN/BATS), DensRay outperforms SVMs. In contrast to SVMs, DensRay considers difference vectors *within* classes as well – this seems to be of advantage here.

## 6 Related Work

**Identifying Interpretable Dimensions.** Most relevant to our method is a line of work that uses transformations of existing word spaces to obtain interpretable subspaces. Rothe et al. (2016) compute an orthogonal transformation using shallow neural networks. Park et al. (2017) apply exploratory factor analysis to embedding spaces

| | | Mean Cosine Sim | | | Precision | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | IntCos | | | | LRCos |
| | | | | | complement | | original | | |
| | | Inter | IntraL | IntraR | DensR. | SVM | DensR. | SVM | |
| FT/BATS | Inflectional | 0.75 | 0.48 | 0.51 | 0.92 | 0.93 | **0.97** | **0.97** | **0.97** |
| | Derivational | 0.63 | 0.47 | 0.45 | 0.74 | 0.78 | **0.81** | 0.80 | 0.80 |
| | Encyclopedia | 0.48 | 0.43 | 0.55 | 0.30 | 0.43 | 0.41 | 0.43 | **0.45** |
| | Lexicography | 0.62 | 0.37 | 0.38 | 0.17 | 0.20 | 0.21 | 0.22 | **0.26** |
| | Macro Mean | 0.62 | 0.44 | 0.47 | 0.53 | 0.58 | 0.60 | 0.60 | **0.61** |
| | Macro Std | 0.12 | 0.06 | 0.09 | 0.34 | 0.33 | 0.34 | 0.33 | **0.32** |
| GN/BATS | Inflectional | 0.63 | 0.22 | 0.23 | **0.88** | 0.87 | **0.88** | **0.88** | **0.88** |
| | Derivational | 0.44 | 0.21 | 0.20 | **0.55** | 0.50 | 0.51 | 0.48 | 0.44 |
| | Encyclopedia | 0.35 | 0.29 | 0.42 | 0.33 | **0.35** | **0.35** | 0.32 | 0.34 |
| | Lexicography | 0.45 | 0.17 | 0.18 | **0.19** | 0.17 | **0.19** | 0.17 | 0.18 |
| | Macro Mean | 0.46 | 0.22 | 0.26 | **0.48** | 0.47 | **0.48** | 0.46 | 0.45 |
| | Macro Std | 0.14 | 0.07 | 0.12 | **0.31** | **0.31** | 0.32 | 0.32 | 0.32 |
| FT/GA | Micro Mean | 0.73 | 0.48 | 0.53 | 0.88 | 0.91 | **0.93** | 0.92 | **0.93** |
| | Macro Mean | 0.71 | 0.50 | 0.53 | 0.87 | 0.90 | **0.91** | 0.90 | 0.89 |
| | Macro Std | 0.11 | 0.05 | 0.06 | 0.11 | **0.08** | 0.12 | 0.17 | 0.23 |
| GN/GA | Micro Mean | 0.62 | 0.31 | 0.36 | 0.85 | 0.87 | **0.89** | 0.87 | 0.88 |
| | Macro Mean | 0.61 | 0.30 | 0.35 | 0.85 | 0.86 | **0.88** | 0.85 | 0.87 |
| | Macro Std | 0.10 | 0.09 | 0.10 | 0.08 | **0.07** | 0.09 | 0.11 | 0.11 |

Table 4: Left part shows mean cosine similarity. Inter: mean cosine similarity between pairs. IntraL/R: mean cosine similarity within the left/right class. Right part shows precision for word analogy task.

to obtain interpretable dimensions in an unsupervised manner. Their approach relies on solving complex optimization problems, while we focus on closed form solutions. Senel et al. (2018) use SEMCAT categories in combination with the Bhattacharya distance to identify interpretable directions. Also, oriented PCA (Diamantaras and Kung, 1996) is closely related to our method. However, both methods yield non-orthogonal transformation. Faruqui et al. (2015a) use semantic lexicons to retrofit embedding spaces. Thus they do not fully maintain the structure of the word space, which is in contrast to this work.

**Interpretable Embedding Algorithms.** Another line of work modifies embedding algorithms to yield interpretable dimensions (Koç et al., 2018; Luo et al., 2015; Shin et al., 2018; Zhao et al., 2018). There is also much work that generates sparse embeddings that are claimed to be more interpretable (Murphy et al., 2012; Faruqui et al., 2015b; Fyshe et al., 2015; Subramanian et al., 2018). Instead of learning new embeddings, we aim at making dense embeddings interpretable.

## 7 Conclusion

We investigated analytical methods for obtaining interpretable word embedding spaces. Relevant methods were examined with the tasks of lexicon induction, word analogy and debiasing.

We gratefully **acknowledge** funding through a Zentrum Digitalisierung.Bayern fellowship awarded to the first author. This work was supported by the European Research Council (# 740516).

## References


Amine Abdaoui, Jérôme Azé, Sandra Bringay, and Pascal Poncelet. 2017. Feel: a french expanded emotion lexicon. *Language Resources and Evaluation*, 51(3).

Silvio Amir, Ramón Astudillo, Wang Ling, Bruno Martins, Mario J Silva, and Isabel Trancoso. 2015. Inesc-id: A regression model for large scale twitter sentiment lexicon induction. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*.

Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Advances in Neural Information Processing Systems*.

Marc Brysbaert, Amy Beth Warriner, and Victor Kuperman. 2014. Concreteness ratings for 40 thousand generally known english word lemmas. *Behavior research methods*, 46(3).

Konstantinos I Diamantaras and Sun Yuan Kung. 1996. *Principal component neural networks: theory and applications*, volume 5. Wiley New York.

Aleksandr Drozd, Anna Gladkova, and Satoshi Matsuoka. 2016. Word embeddings, analogies, and machine learning: Beyond king-man+ woman= queen. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2015a. Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah A Smith. 2015b. Sparse overcomplete word vector representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*.

Jean-Philippe Fauconnier. 2015. French word embeddings.

Alona Fyshe, Leila Wehbe, Partha P Talukdar, Brian Murphy, and Tom M Mitchell. 2015. A compositional and interpretable semantic space. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Roger A Horn, Roger A Horn, and Charles R Johnson. 1990. *Matrix analysis*. Cambridge university press.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*.

Aykut Koç, Ihsan Utlu, Lutfi Kerem Senel, and Haldun M Ozaktas. 2018. Imparting interpretability to word embeddings. *arXiv preprint arXiv:1807.07279*.

Hongyin Luo, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2015. Online learning of interpretable word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Saif M Mohammad and Peter D Turney. 2013. Crowdsourcing a word–emotion association lexicon. *Computational Intelligence*, 29(3).

Brian Murphy, Partha Talukdar, and Tom Mitchell. 2012. Learning effective and interpretable semantic models using non-negative sparse embedding. *Proceedings of the 24th International Conference on Computational Linguistics*.

Sungjoon Park, JinYeong Bak, and Alice Oh. 2017. Rotated word vector representations and their interpretability. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Veronica Perez-Rosas, Carmen Banea, and Rada Mihalcea. 2012. Learning sentiment lexicons in spanish. In *Proceedings of the seventh international conference on Language Resources and Evaluation*.

Sara Rosenthal, Preslav Nakov, Svetlana Kiritchenko, Saif Mohammad, Alan Ritter, and Veselin Stoyanov. 2015. Semeval-2015 task 10: Sentiment analysis in twitter. In *Proceedings of the 9th international workshop on semantic evaluation*.

Sascha Rothe, Sebastian Ebert, and Hinrich Schütze. 2016. Ultradense word embeddings by orthogonal transformation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Lutfi Kerem Senel, Ihsan Utlu, Veysel Yucesoy, Aykut Koc, and Tolga Cukur. 2018. Semantic structure and interpretability of word embeddings. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*.

Jamin Shin, Andrea Madotto, and Pascale Fung. 2018. Interpreting word embeddings with eigenvector analysis. *openreview.net*.

Anant Subramanian, Danish Pruthi, Harsh Jhamtani, Taylor Berg-Kirkpatrick, and Eduard Hovy. 2018. Spine: Sparse interpretable neural embeddings. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Kateřina Veselovská and Ondřej Bojar. 2013. Czech sublex 1.0. Charles University, Faculty of Mathematics and Physics.

Ulli Waltinger. 2010. Germanpolarityclues: A lexical resource for german sentiment analysis. In *Proceedings of the seventh international conference on Language Resources and Evaluation*.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*.

Jieyu Zhao, Yichao Zhou, Zeyu Li, Wei Wang, and Kai-Wei Chang Chang. 2018. Learning gender-neutral word embeddings. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Mengjie Zhao and Hinrich Schütze. 2019. A multilingual bpe embedding space for universal sentiment lexicon induction. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*.

# Investigating Meta-Learning Algorithms for Low-Resource Natural Language Understanding Tasks

**Zi-Yi Dou, Keyi Yu, Antonios Anastasopoulos**
Language Technologies Institute, Carnegie Mellon University
{zdou, keyiy, aanastas}@cs.cmu.edu

## Abstract

Learning general representations of text is a fundamental problem for many natural language understanding (NLU) tasks. Previously, researchers have proposed to use language model pre-training and multi-task learning to learn robust representations. However, these methods can achieve sub-optimal performance in low-resource scenarios. Inspired by the recent success of optimization-based meta-learning algorithms, in this paper, we explore the model-agnostic meta-learning algorithm (MAML) and its variants for low-resource NLU tasks. We validate our methods on the GLUE benchmark and show that our proposed models can outperform several strong baselines. We further empirically demonstrate that the learned representations can be adapted to new tasks efficiently and effectively.

## 1 Introduction

With the ability to learn rich distributed representations of data in an end-to-end fashion, deep neural networks have achieved the state of the arts in a variety of fields (He et al., 2017; Vaswani et al., 2017; Povey et al., 2018; Yu et al., 2018). For natural language understanding (NLU) tasks, robust and flexible language representations can be adapted to new tasks or domains efficiently. Aiming at learning representations that are not exclusively tailored to any specific tasks or domains, researchers have proposed several ways to learn general language representations.

Recently, there is a trend of learning universal language representations via language model pre-training (Dai and Le, 2015; Peters et al., 2018; Radford et al., 2018). In particular, Devlin et al. (2019) present the BERT model which is based on a bidirectional Transformer (Vaswani et al., 2017). BERT is pre-trained with both masked language model and next sentence prediction objectives and exhibits strong performance on several benchmarks, attracting huge attention from researchers. Another line of research tries to apply multi-task learning to representation learning (Liu et al., 2015; Luong et al., 2015). Multi-task learning allows the model to leverage supervision signals from related tasks and prevents the model from overfitting to a single task. By combining the strengths of both language model pre-training and multi-task learning, Liu et al. (2019) improve the BERT model with multi-task learning and their proposed MT-DNN model successfully achieves state-of-the-art results on several NLU tasks.

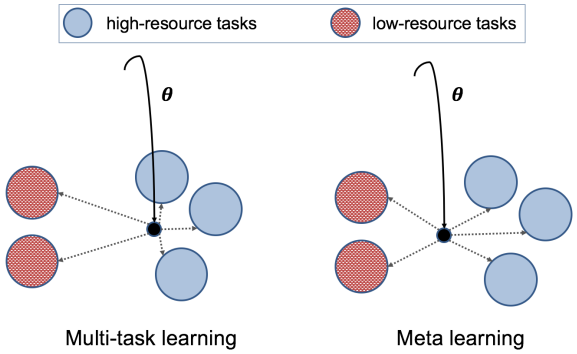


Figure 1: Differences between multi-task learning and meta learning. Multi-task learning may favor high-resource tasks over low-resource ones while meta-learning aims at learning a good initialization that can be adapted to any task with minimal training samples. The figure is adapted from Gu et al. (2018).

Although multi-task learning can achieve promising performance, there still exist some potential problems. As shown in Figure 1, multi-task learning may favor tasks with significantly larger amounts of data than others. Liu et al. (2019) alleviate this problem by adding an additional fine-tuning stage after multi-task learning. In this paper, we propose to apply meta-learning algorithms in general language represen-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1192–1197,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

tations learning. Meta-learning algorithms aim at learning good initializations that can be useful for fine-tuning on various tasks with minimal training data, which makes them appealing alternatives to multi-task learning. Specifically, we investigate the recently proposed model-agnostic meta-learning algorithm (MAML) (Finn et al., 2017) and its variants, namely first-order MAML and Reptile (Nichol et al., 2018), for NLU tasks.

We evaluate the effectiveness and generalization ability of the proposed approaches on the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2019). Experimental results demonstrate that our approaches successfully outperform strong baseline models on the four low-resource tasks. In addition, we test generalization capacity of the models by fine-tuning them on a new task, and the results reveal that the representations learned by our models can be adapted to new tasks more effectively compared with baseline models.

## 2 Proposed Approaches

In this section, we first briefly introduce some key ideas of meta learning, and then illustrate how we apply meta-learning algorithms in language representations learning.

### 2.1 Background: Meta Learning

Meta-learning, or learning-to-learn, has recently attracted researchers' interests in the machine learning community (Lake et al., 2015). The goal of meta-learning algorithms is to allow fast adaptation on new training data. In this paper, we mainly focus on optimization-based meta-learning algorithms, which achieve the goal by adjusting the optimization algorithm. Specifically, we investigate MAML, one of the most representative algorithms in this category, and its variants for NLU tasks.

MAML and its variants offer a way to learn from a distribution of tasks and adapt to target tasks using few samples. Formally, given a set of tasks $\{T_1, \cdots, T_k\}$, the process of learning model parameters $\theta$ can be understood as (Gu et al., 2018):

$$\theta_t^* = \text{Learn}(T_t; \text{MetaLearn}(T_1, \cdots, T_k)),$$

where $T_t$ is the target task.

Hopefully, by exposing models to a variety of tasks, the models can learn new tasks with few steps and minimal amounts of data.

### 2.2 General Framework

In this part, we introduce the general framework of the MAML approach and its variants, including first-order MAML and Reptile.

**Algorithm 1** Training procedure.

Pre-train model parameters $\theta$ with unlabeled datasets.
**while** not done **do**
  Sample batch of tasks $\{T_i\} \sim p(T)$
  **for all** $T_i$ **do**
    Compute $\theta_i^{(k)}$ with Eqn. 1.
  **end for**
  Update $\theta$ with Eqn. 2.
**end while**
Fine-tune $\theta$ on the target task.

We first describe the meta-learning stage. Suppose we are given a model $f_\theta$ with parameters $\theta$ and a task distribution $p(T)$ over a set of tasks $\{T_1, T_2, \cdots, T_k\}$, at each step during the meta-learning stage, we first sample a batch of tasks $\{T_i\} \sim p(T)$, and then update the model parameters by $k$ ($k \geq 1$) gradient descent steps for each task $T_i$ according to the equation:

$$\theta_i^{(k)} = \theta_i^{(k-1)} - \alpha \nabla_{\theta_i^{(k-1)}} L_i(f_{\theta_i^{(k-1)}}), \quad (1)$$

where $L_i$ is the loss function for $T_i$ and $\alpha$ is a hyper-parameter.

The model parameters $\theta$ are then updated by:

$$\theta = \text{MetaUpdate}(\theta; \{\theta_i^{(k)}\}). \quad (2)$$

We would illustrate the *MetaUpdate* step in the following part. It should be noted that the data used for the MetaUpdate step (Eqn. 2) is different from that used for the first $k$ gradient descent steps (Eqn. 1).

The overall training procedure is shown in Algorithm 1. Basically, the algorithm consists of three stages: the pre-training stage as in BERT, the meta-learning stage and the fine-tuning stage.

### 2.3 The MetaUpdate Step

As demonstrated in the previous paragraph, MetaUpdate is an important step in the meta-learning stage. In this paper, we investigate three ways to perform MetaUpdate as described in the following parts.

**MAML** The vanilla MAML algorithm (Finn et al., 2017) updates the model with the meta-objective function:

$$\min_{\theta} \sum_{T_i \sim p(T)} L_i(f_{\theta_i^{(k)}})$$

Therefore, MAML would implement the MetaUpdate step by updating $\theta$ according to:

$$\theta = \theta - \beta \sum_{T_i \sim p(T)} \nabla_{\theta} L_i(f_{\theta_i^{(k)}}),$$

where $\beta$ is a hyper-parameter.

**First-Order MAML** Suppose $\theta^{(k)}$ is obtained by performing $k$ inner gradient steps starting from the initial parameter $\theta^{(0)}$, we can deduce that:

$$\nabla_{\theta^{(0)}} L(f_{\theta^{(k)}}) = \nabla_{\theta^{(k)}} L(f_{\theta^{(k)}}) \prod_{i=1}^{k} \nabla_{\theta^{(i-1)}} \theta^{(i)}$$

$$= \nabla_{\theta^{(k)}} L(f_{\theta^{(k)}}) \prod_{i=1}^{k} (I - \alpha \nabla^2_{\theta^{(i-1)}} L(f_{\theta^{(i-1)}})).$$

Therefore, MAML requires calculating second derivatives, which can be both computationally and memory intensive. First-Order MAML (FO-MAML) ignores the second derivative part and implement the MetaUpdate as:

$$\theta = \theta - \beta \sum_{T_i \sim p(T)} \nabla_{\theta_i^{(k)}} L_i(\theta_i^{(k)}).$$

**Reptile** Reptile (Nichol et al., 2018) is another first-order gradient-based meta-learning algorithm that is similar to joint training, as it implements the MetaUpdate step as:

$$\theta = \theta + \beta \frac{1}{|\{T_i\}|} \sum_{T_i \sim p(T)} (\theta_i^{(k)} - \theta).$$

Basically, Reptile moves the model weights towards new parameters obtained by multiple gradient descent steps. Despite the simplicity of Reptile, it has been demonstrated to achieve competitive or superior performance compared to MAML.

### 2.4 Choosing the Task Distributions

We experiment with three different choices of the task distribution $p(T)$. Specifically, we propose the following options:

- **Uniform**: sample tasks uniformly.

| Model | Test Dataset | | | |
|---|---|---|---|---|
| | CoLA | MRPC | STS-B | RTE |
| BERT | 52.1 | 88.9/84.8 | 87.1/85.8 | 66.4 |
| MT-DNN | 51.7 | 89.9/86.3 | 87.6/86.8 | 75.4 |
| MAML | **53.4** | 89.5/85.8 | 88.0/87.3 | 76.4 |
| FOMAML | 51.6 | 89.9/86.4 | 88.6/88.0 | 74.1 |
| Reptile | 53.2 | **90.2/86.7** | **88.7/88.1** | **77.0** |

Table 1: Results on GLUE test sets. Metrics differ per task (explained in Appendix A) but the best result is **highlighted**.

- **Probability Proportional to Size (PPS)**: the probability of selecting a task is proportional to the size of its dataset.

- **Mixed**: at each epoch, we first sample tasks uniformly and then exclusively select the target task.

## 3 Experiments

We conduct experiments on the GLUE dataset (Wang et al., 2019) and only on English. Following previous work (Devlin et al., 2019; Liu et al., 2019) we do not train or test models on the WNLI dataset (Levesque et al., 2012). We treat the four high-resource tasks, namely SST-2 (Socher et al., 2013), QQP,[1] MNLI (Williams et al., 2018), and QNLI (Rajpurkar et al., 2016), as auxiliary tasks. The other four tasks, namely CoLA (Warstadt et al., 2018), MRPC (Dolan and Brockett, 2005), STS-B (Cera et al., 2017), and RTE (Dagan et al., 2005) are our target tasks. We also evaluate the generalization ability of our approaches on the SciTail dataset (Khot et al., 2018). The details of all datasets are illustrated in Appendix A.

We compare our models with two strong baselines: the BERT model (Devlin et al., 2019) and the MT-DNN model (Liu et al., 2019). While the former pre-trains the Transformer model on large amounts of unlabeled dataset, the latter further improves it with multi-task learning.

For BERT and MT-DNN, we use their publicly available code to obtain the final results. The setting of MT-DNN is slightly different from the setting of BERT in terms of optimizer choices. We implement our algorithms upon the **BERT$_{\text{BASE}}$**

[1] data.quora.com/First-Quora-DatasetRelease-Question-Pairs

| Model | CoLA | MRPC | STS-B | RTE |
|---|---|---|---|---|
| Reptile-PPS | **61.6** | **90.0** | **90.3** | **83.0** |
| Reptile-Uniform | 61.5 | 84.0 | **90.3** | 75.7 |
| Reptile-Mixed 2:1 | 60.3 | 87.8 | **90.3** | 71.0 |
| Reptile-Mixed 5:1 | **61.6** | 85.8 | 90.1 | 74.7 |

Table 2: Effect of task distributions. We report the accuracy or Matthews correlation on development sets.

model.[2] We use the Adam optimizer (Kingma and Ba, 2015) with a batch size of 32 and learning rates of 5e-5 to train the models for 5 epochs in the meta-learning stage. We set the update step $k$ to 5, the number of sampled tasks in each step to 8 and $\alpha$ to 1e-3.

### 3.1 Results

We first use the three meta-learning algorithms with PPS sampling and present in Table 1 the experimental results on the GLUE test set. Generally, the meta-learning algorithms achieve better performance than the strong baseline models, with Reptile performing the best.

Since the MT-DNN also uses PPS sampling, the improvements suggest meta-learning algorithms can indeed learn better representations compared with multi-task learning. Reptile outperforming MAML indicates that reptile is a more effective and efficient algorithm compared with MAML in our setting.

### 3.2 Ablation Studies

**Effect of Task Distributions** As we have mentioned above, we propose three different choices of the task distribution $p(T)$ in this paper. Here we train Reptile with these task distributions and test models' performance on the development set as shown in Table 2.

For uniform sampling, we set the number of training steps equal to that of the PPS method. For mixed sampling, we try mix ratios of both 2:1 and 5:1. The results demonstrate that Reptile with PPS sampling achieves the best performance, which suggests that larger amounts of auxiliary task data can generally lead to better performance.

**Effect of Hyperparameters for Meta-Gradients** In this part, we test the effect of the number of update steps $k$ and the learning rate in the inner learning loop. The experimental results on the development sets are shown in Table 3. We find that setting $k$ to 5 is the optimal strategy and more or fewer update steps may lead to worse performance.

Smaller $k$ would make the algorithms similar to joint training as joint training is an extreme case of Reptile where $k = 1$, and thus cause the model to lose the advantage of using meta-learning algorithms. Similarly, Larger $k$ can make the resulting gradients deviate from the normal ones and become uninformative.

We also vary the inner learning rate $\alpha$ and investigate its impact. The results are listed in Table 3. We can see that larger $\alpha$ may degrade the performance because the resulting gradients deviate a lot from normal ones. The above two ablations studies demonstrate the importance of making the meta-gradient informative.

[2] **BERT$_{\text{BASE}}$** and **BERT$_{\text{LARGE}}$** differ at the number of hidden layers (12 vs. 24), hidden size (768 vs. 1024) and the number of attention heads (12 vs. 16).

| Model | #Upt | $\alpha$ | CoLA | MRPC | STS-B | RTE |
|---|---|---|---|---|---|---|
| Reptile | 3 | 1e-3 | 60.7 | 89.7 | 90.2 | 77.9 |
| | 5 | 1e-4 | **62.0** | 88.0 | 90.1 | 81.2 |
| | | 1e-3 | 61.6 | **90.0** | **90.3** | **83.0** |
| | | 1e-2 | 60.1 | 87.8 | 89.5 | 73.9 |
| | 7 | 1e-3 | 57.8 | 88.7 | 90.0 | 81.4 |

Table 3: Effect of the number of update steps and the inner learning rate $\alpha$.

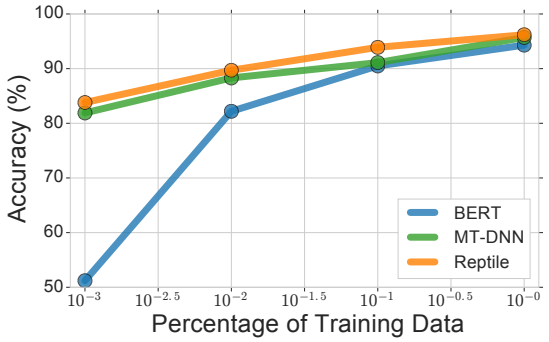


Figure 2: Results on transfer learning. The target task is SciTail which the model does not come across during the meta-learning stage.

### 3.3 Transferring to New Tasks

In this part, we test whether our learned representations can be adapted to new tasks efficiently. To this end, we perform transfer learning experiments on a new natural language inference dataset, namely SciTail.

We randomly sample 0.1%, 1%, 10% and 100% of the training data and test models' performance on these datasets. Figure 2 reveals that our model consistently outperforms the strong MT-

DNN baseline across different settings, indicating the learned representations are more effective for transfer learning. In particular, the algorithm is more effective when less data are available, especially compared to BERT, suggesting the meta-learning algorithms can indeed be helpful for low-resource tasks.

## 4 Related Work

There is a long history of learning general language representations. Previous work on learning general language representations focus on learning word (Mikolov et al., 2013; Pennington et al., 2014) or sentence representations (Le and Mikolov, 2014; Kiros et al., 2015) that are helpful for downstream tasks. Recently, there is a trend of learning contextualized word embeddings (Dai and Le, 2015; McCann et al., 2017; Peters et al., 2018; Howard and Ruder, 2018). One representative approach is the BERT model (Devlin et al., 2019) which learns contextualized word embeddings via bidirectional Transformer models.

Another line of research on learning representations focus on multi-task learning (Collobert et al., 2011; Liu et al., 2015). In particular, Liu et al. (2019) propose to combine multi-task learning with language model pre-training and demonstrate the two methods are complementary to each other.

Meta-learning algorithms have received lots of attention recently due to their effectiveness (Finn et al., 2017; Fan et al., 2018). However, the potential of applying meta-learning algorithms in NLU tasks have not been fully investigated yet. Gu et al. (2018) have tried to apply first-order MAML in machine translation and Qian and Yu (2019) propose to address the domain adaptation problem in dialogue generation by using MAML. To the best of our knowledge, the Reptile algorithm, which is simpler than MAML and potentially more useful, has been given less attention.

## 5 Conclusion

In this paper, we investigate three optimization-based meta-learning algorithms for low-resource NLU tasks. We demonstrate the effectiveness of these algorithms and perform a fair amount of ablation studies. We also show the learned representations can be adapted to new tasks effectively. Our study suggests promising applications of meta-learning algorithms in the field of NLU. Future directions include integrating more sophisticated training strategies of meta-learning algorithms as well as validating our algorithms on other datasets.


## Acknowledgement

The authors are grateful to the anonymous reviewers for their constructive comments, and to Graham Neubig and Junxian He for helpful discussions. This material is based upon work generously supported partly by the National Science Foundation under grant 1761548.


## References


Daniel Cera, Mona Diabb, Eneko Agirrec, Inigo Lopez-Gazpioc, Lucia Speciad, and Basque Country Donostia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and cross-lingual focused evaluation. In *11th International Workshop on Semantic Evaluations*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *JMLR*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190. Springer.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *NeurIPS*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Yang Fan, Fei Tian, Tao Qin, Xiang-Yang Li, and Tie-Yan Liu. 2018. Learning to teach. In *ICLR*.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*.

Jiatao Gu, Yong Wang, Yun Chen, Victor OK Li, and Kyunghyun Cho. 2018. Meta-learning for low-resource neural machine translation. In *EMNLP*.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. Mask r-cnn. In *CVPR*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *ACL*.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. Scitail: A textual entailment dataset from science question answering. In *AAAI*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *NeurIPS*.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. 2015. Human-level concept learning through probabilistic program induction. *Science*.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*.

Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. 2015. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *NAACL*.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. *arXiv*.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015. Multi-task sequence to sequence learning. *arXiv*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *NeurIPS*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NeurIPS*.

Alex Nichol, Joshua Achiam, and John Schulman. 2018. On first-order meta-learning algorithms. *arXiv*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*.

Daniel Povey, Gaofeng Cheng, Yiming Wang, Ke Li, Hainan Xu, Mahsa Yarmohamadi, and Sanjeev Khudanpur. 2018. Semi-orthogonal low-rank matrix factorization for deep neural networks. In *InterSpeech*.

Kun Qian and Zhou Yu. 2019. Domain adaptive dialog generation via meta learning. In *ACL*.

Alec Radford, Karthik Narasimhan, Time Salimans, and Ilya Sutskever. 2018. Improving language understanding with unsupervised learning. Technical report, OpenAI.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*.

Alex Wang, Amapreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR*.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2018. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL*.

Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. 2018. Deep layer aggregation. In *CVPR*.

# Retrofitting Contextualized Word Embeddings with Paraphrases

**Weijia Shi**[1]*, **Muhao Chen**[1,2]*, **Pei Zhou**[1,3] **and Kai-Wei Chang**[1]
[1]Department of Computer Science, University of California Los Angeles
[2]Department of Computer and Information Science, University of Pennsylvania
[3]Department of Computer Science, University of Southern California
{swj0419, kwchang}@cs.ucla.edu; muhao@seas.upenn.edu; peiz@usc.edu

## Abstract

Contextualized word embedding models, such as ELMo, generate meaningful representations of words and their context. These models have been shown to have a great impact on downstream applications. However, in many cases, the contextualized embedding of a word changes drastically when the context is paraphrased. As a result, the downstream model is not robust to paraphrasing and other linguistic variations. To enhance the stability of contextualized word embedding models, we propose an approach to retrofitting contextualized embedding models with paraphrase contexts. Our method learns an orthogonal transformation on the input space, which seeks to minimize the variance of word representations on paraphrased contexts. Experiments show that the retrofitted model significantly outperforms the original ELMo on various sentence classification and language inference tasks.

## 1 Introduction

Contextualized word embeddings have shown to be useful for a variety of downstream tasks (Peters et al., 2018, 2017; McCann et al., 2017). Unlike traditional word embeddings that represent words with fixed vectors, these embedding models encode both words and their contexts and generate context-specific representations. While contextualized embeddings are useful, we observe that a language model-based embedding model, ELMo (Peters et al., 2018), cannot accurately capture the semantic equivalence of contexts. Specifically, in cases where the contexts of a word have equivalent or similar meanings but are changed in sentence formation or word order, ELMo may assign very different representations to the word. Table 1 shows two examples, where ELMo generates very different representations for the boldfaced words under semantic equivalent contexts.

| Paraphrased contexts | L2 | Cosine |
|---|---|---|
| How can I make **bigger** my arms?<br>How do I make my arms **bigger**? | 6.42 | 0.27 |
| Some people believe earth is **flat**. Why?<br>Why do people still believe in **flat** earth? | 7.59 | 0.46 |
| It is a very **small** window.<br>I have a **large** suitcase. | 5.44 | 0.26 |

Table 1: L2 and Cosine distances between embeddings of boldfaced words. The distance between the shared word in the paraphrases is even greater than the distance between *large* and *small* in random contexts.

Quantitatively, 28.3% of the shared words in the paraphrase sentence pairs on the MRPC corpus (Dolan et al., 2004) is larger than the average distance between *good* and *bad* in random contexts, and 41.5% of those exceeds the distance between *large* and *small*. As a result, the downstream model is not robust to paraphrasing and the performance is hindered.

Infusing the model with the ability to capture the semantic equivalence no doubt benefits semantic-oriented downstream tasks. Yet, finding an effective solution presents key challenges. First, the solution inevitably requires the embedding model to effectively identify paraphrased contexts. On top of that, the model needs to minimize the difference of a word's representations on paraphrased contexts, without compromising the varying representations on unrelated contexts. Moreover, the long training time prevents us from redesigning the learning objectives of contextualized embeddings and retraining the model.

To address these challenges, we propose a simple and effective paraphrase-aware retrofitting (PAR) method that is applicable to arbitrary pretrained contextualized embeddings. In particular, PAR prepends an orthogonal transformation layer to a contextualized embedding model. Without retraining the parameters of an existing model, PAR learns the transformation to minimize the differ-

* Both authors contributed equally to this work.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1198–1203,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

ence of the contextualized representations of the shared word in paraphrased contexts, while differentiating between those in other contexts. We apply PAR to retrofit ELMo (Peters et al., 2018) and show that the resulted embeddings provide more robust contextualized word representations as desired, which further lead to significant improvements on various sentence classification and inference tasks.

## 2 Related Work

**Contextualized word embedding models** have been studied by a series of recent research efforts, where different types of pre-trained language models are employed to capture the context information. CoVe (McCann et al., 2017) trains a neural machine translation model and extracts representations of input sentences from the source language encoder. ELMo (Peters et al., 2018) pre-trains LSTM-based language models from both directions and combines the vectors to construct contextualized word representations. Recent studies substitute LSTMs with Transformers (Radford et al., 2018, 2019; Devlin et al., 2019). As shown in these studies, contextualized word embeddings perform well on downstream tasks at the cost of extensive parameter complexity and the long training process on large corpora (Strubell et al., 2019).

**Retrofitting methods** have been used to incorporate semantic knowledge from external resources into word embeddings (Faruqui et al., 2015; Yu et al., 2016; Glavaš and Vulić, 2018). These techniques are shown to improve the characterization of word relatedness and the compositionality of word representations. To the best of our knowledge, none of the previous approaches has been applied in contextualized word embeddings.

## 3 Paraphrase-Aware Retrofitting

Our method, illustrated in Figure 1, integrates the constraint of the paraphrased context into the contextualized word embeddings by learning the orthogonal transformation on the input space.

### 3.1 Contextualized Word Embeddings

We use $S = (w_1, w_2, \cdots, w_l)$ to denote a sequence of words of length $l$, where each word $w$ belongs to the vocabulary $V$. We use boldfaced $\mathbf{w} \in \mathbb{R}^k$ to denote a $k$-dimensional input word embedding, which can be pre-trained or derived from a character-level encoder (e.g., the character-level

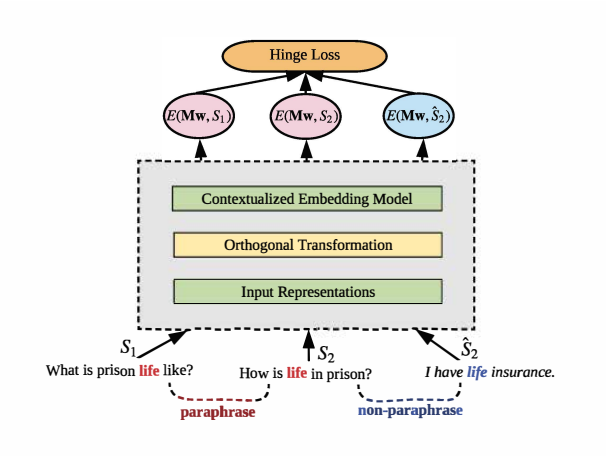


Figure 1: Learning framework of PAR

CNN used in ELMo (Peters et al., 2018)). A contextualized embedding model $E$ takes input vectors of the words in $S$, and computes the context-specific representation of each word. The representation of word $w$ specific to the context $S$ is denoted as $E(\mathbf{w}, S)$.

### 3.2 Paraphrase-aware Retrofitting

PAR learns an orthogonal transformation $\mathbf{M} \in \mathbb{R}^{k \times k}$ to reshape the input representation into a specific space, where the contextualized embedding vectors of a word in paraphrased contexts are collocated, while those in unrelated contexts are differentiated. Specifically, given two contexts $S_1$ and $S_2$ that both contain a shared word $w$, the contextual difference of a input representation $\mathbf{w}$ is defined by the $L2$ distance,

$$d_{S_1,S_2}(\mathbf{w}) = \|E(\mathbf{w}, S_1) - E(\mathbf{w}, S_2)\|_2 .$$

Let $P$ be the set of paraphrases on the training corpus, we minimize the following hinge loss ($L_H$).

$$\sum_{(S_1,S_2)\in P} \sum_{w \in S_1 \cap S_2} \left[ d_{S_1,S_2}(\mathbf{Mw}) + \gamma - d_{\hat{S}_1,\hat{S}_2}(\mathbf{Mw}) \right]_+ .$$

$(S_1, S_2) \in P$ thereof is a pair of paraphrases in $P$. $(\hat{S}_1, \hat{S}_2) \notin P$ is a negative sample generated by randomly substituting either $S_1$ or $S_2$ with another sentence in the dataset that contains $w$. $\gamma > 0$ is a hyper-parameter representing the margin. The operator $[x]_+$ denotes $\max(x, 0)$.

The orthogonalization is realized by the following regularization term.

$$L_O = \left\| \mathbf{I} - \mathbf{M}^\top \mathbf{M} \right\|_F ,$$

where $\|\cdot\|_F$ denotes the Frobenius norm, and $\mathbf{I}$ is an identity matrix. The learning objective of PAR is then denoted as $L = L_H + \lambda L_O$ with a positive hyperparameter $\lambda$.

Orthogonalizing $\mathbf{M}$ has two important effects: (i) It preserves the word similarity captured by the original input word representation (Rothe et al., 2016); (ii) It prevents the model from converging to a trivial solution where all word representations collapse to the same embedding vector.

## 4 Experiment

Our method can be integrated with any contextualized word embedding models. In our experiment, we apply PAR on ELMo (Peters et al., 2018) and evaluate the quality of the retrofitted ELMo on a broad range of sentence-level tasks and the adversarial SQuAD corpus.

### 4.1 Experimental Configuration

We use the officially released 3-layer ELMo (original), which is trained on the 1 Billion Word Benchmark with 93.6 million parameters. We retrofit ELMo with PAR on the training sets of three paraphrase datasets: (i) *MRPC* contains 2,753 paraphrase pairs; (ii) *Sampled Quora* contains randomly sampled 20,000 paraphrased question pairs (Iyer et al., 2017); and (iii) *PAN* training set (Madnani et al., 2012) contains 5,000 paraphrase pairs.

The orthogonal transformation $\mathbf{M}$ is initialized as an identity matrix. In our preliminary experiments, we observed that SGD optimizer is more stable and less likely to quickly overfit the training set than other optimizers with adaptive learning rates (Reddi et al., 2018; Kingma and Ba, 2015). Therefore, we use SGD with the learning rate of 0.005 and a batch size of 128. To determine the terminating condition, we train a Multi-Layer Perceptron (MLP) classifier on the same paraphrase training set and terminate training based on the paraphrase identification performance on a set of held-out paraphrases. The sentence in the dataset is represented by the average of the word embeddings. $\lambda$ is selected from $\{0.1, 0.5, 1, 2\}$ and $\gamma$ from $\{1, 2, 3, 4\}$ based on validation set. The best margin $\gamma$ and epochs $\zeta$ by early stopping are $\{\gamma = 3, \zeta = 20\}$ on MRPC, $\{\gamma = 2, \zeta = 14\}$ on PAN, and $\{\gamma = 3, \zeta = 10\}$ on Sampled Quora, with $\lambda = 1$ in all settings.

### 4.2 Evaluation

We use the SentEval framework (Conneau and Kiela, 2018) to evaluate the sentence embeddings on a wide range of sentence-level tasks. We consider two baselines models: (1) ELMo (all layers) constructs a 3,074-dimensional sentence embedding by averaging the hidden states of all the language model layers. (2) ELMo (top layers) encodes a sentence to a 1,024 dimensional vector by averaging the representations of the top layer. We compare these baselines with four variants of PAR built upon ELMo (all layers) that trained on different paraphrase corpora.

### 4.3 Task Descriptions

**Sentence classification tasks.** We evaluate the sentence embedding on four sentence classification tasks including two sentiment analysis (MR (Pang and Lee, 2004), SST-2 (Socher et al., 2013)), product reviews (CR (Hu and Liu, 2004)), and opinion polarity (MPQA (Wiebe et al., 2005)). These tasks are all binary classification tasks. We employ a MLP with a single hidden layer of 50 neurons to train the classifer, using a batch size of 64 and Adam optimizer.

**Sentence inference tasks** We consider two sentence inference tasks: paraphrase identification on MRPC (Dolan et al., 2004) and the textual entailment on SICK-E (Marelli et al., 2014). MRPC consists of pairs of sentences, where the model aims to classify if two sentences are semantically equivalent. The SICK dataset contains 10,000 English sentence pairs annotated for relatedness in meaning and entailment. The aim of SICK-E is to detect discourse relations of entailment, contradiction and neutral between the two sentences. Similar to the sentence classification tasks, we apply a MLP with the same hyperparameters to conduct the classification.

**Semantic textual similarity tasks**. Semantic Textual Similarity (STS-15 (Agirre et al., 2015) and STS-16 (Agirre et al., 2016)) measures the degree of semantic relatedness of two sentences based on human-labeled scores from 0 to 5. We report the Pearson correlation between cosine similarity of two sentence representations and normalized human-label scores.

**Semantic relatedness tasks** The semantic relatedness tasks include SICK-R (Marelli et al., 2014) and the STS Benchmark dataset (Cer et al., 2017), which comprise pairs of sentences annotated with

| Method | Classification (Acc %) | | | | Relatedness \ Similarity ($\rho$) | | | | Inference (Acc %) | |
|---|---|---|---|---|---|---|---|---|---|---|
| | MPQA | MR | CR | SST-2 | SICK-R | STS-15 | STS-16 | STS-Benchmark | MRPC | SICK-E |
| ELMo (all layers) | 89.55 | 79.72 | 85.11 | 86.33 | 0.84 | 0.69 | 0.64 | 0.65 | 71.65 | 81.86 |
| ELMo (top layer) | 89.30 | 79.36 | 84.13 | 85.28 | 0.81 | 0.67 | 0.63 | 0.62 | 70.20 | 79.64 |
| ELMo-PAR (*MRPC*) | 92.61 | **83.40** | **87.01** | 86.83 | **0.87** | 0.70 | 0.66 | 0.64 | - | 82.89 |
| ELMo-PAR (*Sampled Quora*) | **93.76** | 81.14 | 85.52 | 88.71 | 0.83 | 0.71 | 0.66 | 0.69 | 73.22 | 81.51 |
| ELMo-PAR (*PAN*) | 92.13 | 83.11 | 85.73 | 88.56 | 0.85 | **0.73** | 0.67 | **0.70** | **74.86** | 83.37 |
| ELMo-PAR (*PAN+MRPC+Quora*) | 93.40 | 82.26 | 86.39 | **89.26** | 0.86 | **0.73** | **0.68** | 0.67 | - | **84.46** |

Table 2: Performance on downstream applications. We report accuracy for classification and inference tasks, and Pearson correlation for relatedness and similarity tasks. We do not report results of ELMo-PAR on MRPC when MRPC is used in training the model. The baseline results are from (Perone et al., 2018).

| Model | AddOneSent | | AddSent | |
|---|---|---|---|---|
| | EM | F1 | EM | F1 |
| BiSAE | 47.7 | 53.7 | 36.1 | 41.7 |
| BiSAE-PAR(MRPC) | **51.6** | **57.9** | **40.8** | **47.1** |

Table 3: Exact Match and F1 on Adversarial SQuAD.

semantic scores between 0 and 5. The goal of the tasks is to measure the degree of semantic relatedness between two sentences. We learn tree-structured LSTM (Tai et al., 2015) to predict the probability distribution of relatedness scores.

**Adversarial SQuAD** The Stanford Question Answering Datasets (SQuAD) (Rajpurkar et al., 2016) is a machine comprehension dataset containing 107,785 human-generated reading comprehension questions annotated on Wikipedia articles. Adversarial SQuAD (Jia and Liang, 2017) appends adversarial sentences to the passage in the SQuAD dataset to study the robustness of the model. We conduct evaluations on two Adversarial SQuAD datasets: AddOneSent which adds a random human-approved sentence, and AddSent which adds grammatical sentences that look similar to the question. We train the Bi-Directional Attention Flow (BiDAF) network (Seo et al., 2017) with self-attention and ELMo embeddings on the SQuAD dataset and test it on the adversarial SQuAD datasets.

### 4.4 Result Analysis

The results reported in Table 2 show that PAR leads to 2% $\sim$ 4% improvement in accuracy on sentence classification tasks and sentence inference tasks. It leads to 0.03 $\sim$ 0.04 improvement in Pearson correlation ($\rho$) on semantic relatedness and textual similarity tasks. The improvements on sentence similarity and semantic relatedness tasks shows that ELMo-PAR is more stable to the semantic-preserving modifications but more sensitive to subtle yet semantic-changing perturbations. PAR model trained on the combined corpus (*PAN+MRPC+Sampled Quora*) achieves the best improvement across all these tasks, showing the model benefits from a larger paraphrase corpus. Besides sentence-level tasks, Table 3 shows that the proposed PAR method notably improves the performance of a downstream question-answering task. For AddSent, ELMo-PAR achieves 40.8% in EM and 47.1% in F1. For AddOneSent, it boosts EM to 51.6% and F1 to 57.9%, which clearly shows that the proposed PAR method enhances the robustness of the downstream model combined with ELMo.

### 4.5 Case Study

**Shared word distances** We compute the average embedding distance of shared words in paraphrase and non-paraphrase sentence pairs from test sets of MRPC, PAN, and Quora. Results are listed in Table 4. Table 5 shows the ELMo-PAR embedding distance for the shared words in the examples in Table 1. Our model effectively minimizes the embedding distance of the shared words in the paraphrased contexts and maximize such distance in the non-paraphrased contexts.

## 5 Conclusion

We propose a method for retrofitting contextualized word embeddings, which leverages semantic equivalence information from paraphrases. PAR learns an orthogonal transformation on the input space of an existing model by minimizing the contextualized representations of shared words on paraphrased contexts without compromising the varying representations on non-paraphrased contexts. We demonstrate the effectiveness of this method applied to ELMo by a wide selection of semantic tasks. We seek to extend the use of PAR to other contextualized embeddings (Devlin et al., 2019; McCann et al., 2017) in future work.

| Model | Paraphrase | | | Non-paraphrase | | |
|---|---|---|---|---|---|---|
| | MRPC | Quora | PAN | MRPC | Quora | PAN |
| ELMo(all layers) | 3.35 | 3.17 | 4.03 | 3.97 | 4.42 | 6.26 |
| ELMo-PAR(*PAN+MRPC+Quora*) | 1.80 | 1.34 | 1.21 | 4.73 | 5.49 | 6.54 |

Table 4: Averaging L2 distance for the shared word in paraphrased and non-paraphrased contexts.

| Paraphrased contexts | L2 | Cosine |
|---|---|---|
| How can I make **bigger** my arms?<br>How do I make my arms **bigger**? | 2.75 | 0.14 |
| Some people believe earth is **flat**. Why?<br>Why do people still believe in **flat** earth? | 3.29 | 0.16 |
| It is a very **small** window.<br>I have a **large** suitcase. | 5.84 | 0.30 |

Table 5: L2 and Cosine distance between embeddings of boldfaced words after retrofitting.

.

## 6 Acknowledgement

This work was supported in part by National Science Foundation Grant IIS-1760523. We thank reviewers for their comments.

## References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Inigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, et al. 2015. Semeval-2015 task 2: Semantic textual similarity, english, spanish and pilot on interpretability. In *Proceedings of the 9th international workshop on semantic evaluation*.

Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016. Semeval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *Proceedings of the 10th International Workshop on Semantic Evaluation*.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation*.

Alexis Conneau and Douwe Kiela. 2018. Senteval: An evaluation toolkit for universal sentence representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *COLING*.

Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2015. Retrofitting word vectors to semantic lexicons. In *NAACL*.

Goran Glavaš and Ivan Vulić. 2018. Explicit retrofitting of distributional word vectors. In *ACL*.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*.

Shankar Iyer, Nikhil Dandekar, and Kornél Csernai. 2017. First quora dataset release: Question pairs. *data.quora.com*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *EMNLP*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *ICLR*.

Nitin Madnani, Joel Tetreault, and Martin Chodorow. 2012. Re-examining machine translation metrics for paraphrase identification. In *NAACL*.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of the 8th international workshop on semantic evaluation*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *NIPS*.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *ACL*.

Christian S Perone, Roberto Silveira, and Thomas S Paula. 2018. Evaluation of sentence embeddings in downstream and linguistic probing tasks. *arXiv preprint arXiv:1806.06259*.

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *ACL*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *EMNLP*.

Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. 2018. On the convergence of adam and beyond. In *ICLR*.

Sascha Rothe, Sebastian Ebert, and Hinrich Schütze. 2016. Ultradense word embeddings by orthogonal transformation. In *NAACL*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *ICLR*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in nlp. In *ACM*.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *ACL-IJCNLP*.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language resources and evaluation*.

Zhiguo Yu, Trevor Cohen, Byron Wallace, Elmer Bernstam, and Todd Johnson. 2016. Retrofitting word vectors of mesh terms to improve semantic similarity measures. In *Proceedings of the Seventh International Workshop on Health Text Mining and Information Analysis*.

# Incorporating Contextual and Syntactic Structures Improves Semantic Similarity Modeling

**Linqing Liu,**[1] **Wei Yang,**[1] **Jinfeng Rao,**[2] **Raphael Tang,**[1] and **Jimmy Lin**[1]
[1] David R. Cheriton School of Computer Science, University of Waterloo
[2] Department of Computer Science, University of Maryland, College Park
{linqing.liu, w85yang, r33tang, jimmylin}@uwaterloo.ca, raojinfeng@gmail.com

## Abstract

Semantic similarity modeling is central to many NLP problems such as natural language inference and question answering. Syntactic structures interact closely with semantics in learning compositional representations and alleviating long-range dependency issues. However, such structure priors have not been well exploited in previous work for semantic modeling. To examine their effectiveness, we start with the Pairwise Word Interaction Model, one of the best models according to a recent reproducibility study, then introduce components for modeling context and structure using multi-layer BiLSTMs and TreeLSTMs. In addition, we introduce residual connections to the deep convolutional neural network component of the model. Extensive evaluations on eight benchmark datasets show that incorporating structural information contributes to consistent improvements over strong baselines.

## 1 Introduction

Modeling the semantic similarity between a pair of sentences is a fundamental task in natural language processing. It is the core problem of many tasks such as question answering (He et al., 2015; Rao et al., 2017; Wang et al., 2018) and query ranking (Mitra and Craswell, 2019). Recently, various neural networks have been proposed for textual similarity modeling. These models share three main components: (1) sequential sentence encoders, which incorporate word context and sentence order for better sentence representations, e.g., by using recurrent neural networks (RNNs; Mikolov et al., 2010; Seo et al., 2016), (2) interaction and attention mechanisms, which use the encoding outputs of sentences to calculate or reweight salient word pair interactions (He and Lin, 2016; Chen et al., 2017), and (3) incorporating syntactic parsing information as an intuitive structure prior for sentence modeling (Chen and Manning, 2014; Zhao et al., 2016; Chen et al., 2017).

Our work is inspired by the recent reproducibility study by Lan and Xu (2018), which examines many neural network architectures for semantic similarity modeling through extensive evaluations on multiple benchmark datasets. Their results suggest that syntactic structure information captured by a TreeLSTM encoder either provides few benefits or even hurts performance. Structure information has often been overlooked in recent semantic modeling methods, such as InferSent (Conneau et al., 2017), DecAtt (Parikh et al., 2016), and BiMPM (Wang et al., 2017). It is not yet clear whether the syntactic structures implicitly captured by sequential modeling of texts from large annotated data or existing structure modeling techniques (Tai et al., 2015; Kipf and Welling, 2017) are effective in learning tree representations.

To further explore the effects of tree structures in sentence modeling, we start with the Pairwise Word Interaction Model (PWIM) of He and Lin (2016) as our base architecture, which has shown strong performance on various datasets from Lan and Xu (2018). In summary, PWIM uses a BiLSTM to learn word-level context vectors from both input sentences and builds a novel similarity focus layer with pairwise metrics to identify important word pairs. It then converts the similarity measurement problem to a pattern recognition problem for the final classification. We argue that PWIM approaches semantic modeling from a word-level matching perspective, and hence fails to capture syntactic and contextual semantics. To this end, we add multi-layer BiLSTMs with shortcut connections to capture long-range context, as well as TreeLSTM encoders to capture the syntactic structure of sentences.

We conduct thorough evaluations across eight



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1204–1209, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

datasets in four NLP tasks: paraphrase identification, semantic textual similarity, natural language inference, and answer sentence selection. Extensive experiments show that our proposed components lead to consistent improvements against PWIM and other strong baselines, suggesting that incorporating contextual and syntactic structures can help semantic modeling. Our improved model achieves competitive numbers on eight datasets, and we open-source our code to improve reproducibility and to facilitate future research.[1]

## 2 Methods

### 2.1 The Pairwise Word Interaction Model

The Pairwise Word Interaction model (He and Lin, 2016) captures fine-grained word-level information to measure textual similarity. They use a Bi-LSTM for context modeling, where the word at time step $t$ is encoded as a forward hidden state $h_t^{\text{for}}$ and a backward hidden state $h_t^{\text{back}}$. Pairwise word interactions are modeled through a multi-metric comparison unit $coU$, which computes cosine distance, $L_2$ distance, and dot-product distance over two hidden states. This comparison unit is applied to not only the forward and backward hidden states $h_t^{\text{for}}$ and $h_t^{\text{back}}$, but also their concatenation $\overleftrightarrow{h_t} = [h_t^{\text{for}}, h_t^{\text{back}}]$ and summation $h_t^{+} = h_t^{\text{for}} + h_t^{\text{back}}$.

The output is a similarity tensor of size $R^{13\times|sent1|\times|sent2|}$ with one extra dimension for the padding indicator. Instead of using attention weight vectors or weighted representations, He and Lin apply a focus layer on the similarity tensor to decrease the weights of unimportant word interactions by a factor of ten. They then consider the tensor as an "image" with 13 channels and use a 19-layer-deep convolutional neural network to predict the final classification.

### 2.2 Residual Connections

Since He and Lin (2016) phrase the similarity measurement problem as a pattern recognition (image processing) problem and apply deep convolutional neural networks, we explore the addition of residual connections (He et al., 2016) to deal with the potential vanishing gradient problems in deep networks. A building block is defined as $y = f(x, W_i) + x$, where $x$, $y$ are the input and output of the layer considered, and $f(x, W_i)$ is the learned residual mapping.

[1] https://github.com/likicode/spwim

### 2.3 Multi-layer BiLSTM Sentence Encoders

We use multiple stacked, bi-directional LSTM layers with shortcut connections, similar to Nie and Bansal (2017). In this architecture, the input sequences of the $i^{\text{th}}$ BiLSTM layer are the concatenated outputs of all the previous layers and the initial word embedding sequences. Let $W = \{w_1, w_2, ..., w_n\}$ represent the word embeddings associated with each word in the source sentence. Define the output of the $i^{\text{th}}$ BiLSTM layer at time $t$ as $h_t^i = \text{BiLSTM}^i(x_t^i)$. Then, the input of the $i^{\text{th}}$ BiLSTM layer at time $t$ is:

$$x_t^1 = w_t, \;\; x_t^i = [w_t, h_t^1, ..., h_t^{i-2}, h_t^{i-1}](i > 1)$$

Differing from the original paper, we directly use the output of the last BiLSTM layer to encode the sentence $v = (h_1^m, h_2^m, ...h_n^m)$, where $n$ denotes the length of the sentence and $m$ the number of BiLSTM layers. In our experiments we set $m = 3$.

### 2.4 Hybrid Inference Model with Parse Trees

While stacked BiLSTMs capture long-term dependency and contextual information over each sentence, we are also interested in investigating explicit hierarchical relationships among linguistic phrases and clauses. To incorporate this domain-specific information, we use the Dependency TreeLSTM (Tai et al., 2015), whose nodes condition their components on the sum of the hidden states of their children. Suppose $h_L$ and $h_R$ are the sentence representations in the pair over the parse tree of each sentence: to model similarity, we compute the element-wise product $h_L \odot h_R$ and absolute difference $|h_L - h_R|$. Then, we feed the two similarity vectors to a fully-connected layer with softmax whose output is the probability distribution over labels. To compute the final label for the sentence pair, we interpolate between the output probabilities of this model and those of PWIM. Chen et al. (2017) also incorporate tree structures produced by a constituency parser into the ESIM model, then average the predicted probabilities.

## 3 Experimental Setup

We conducted experiments on eight separate datasets—one natural language inference dataset, two paraphrase identification datasets, three SemEval competition datasets, and two QA datasets—which are as follows:

- SNLI (Bowman et al., 2015) is a collection of 570k manually-labeled sentence pairs for

| Dataset | SNLI | Quora | Twitter | PIT-2015 | STS-2014 | WikiQA | TrecQA | SICK |
|---|---|---|---|---|---|---|---|---|
| | Acc | Acc | F1 | F1 | Pearson's r | MAP/MRR | MAP/MRR | Pearson's r/$\rho$ |
| InferSent | 0.846 | 0.866 | 0.746 | 0.451 | 0.715 | 0.287/0.287 | 0.521/0.559 | - |
| SSE | 0.855 | 0.878 | 0.650 | 0.422 | 0.378 | 0.624/0.638 | 0.628/0.670 | - |
| DecAtt | 0.856 | 0.845 | 0.652 | 0.430 | 0.317 | 0.603/0.619 | 0.660/0.712 | - |
| ESIM$_{tree}$ | 0.864 | 0.755 | 0.740 | 0.447 | 0.493 | 0.618/0.633 | 0.698/0.734 | - |
| ESIM$_{seq}$ | 0.870 | 0.850 | 0.748 | 0.520 | 0.602 | 0.652/0.664 | 0.771/0.795 | - |
| ESIM$_{seq+tree}$ | 0.871 | 0.854 | 0.759 | 0.538 | 0.589 | 0.647/0.658 | 0.749/0.768 | - |
| PWIM$_{our}$ | 0.822 | 0.853 | 0.745 | 0.602 | 0.695 | 0.709/0.723 | 0.759/0.822 | 0.871/0.809 |
| mPWIM$_{seq}$ | 0.851 | 0.862 | 0.757 | 0.612 | 0.714 | 0.717/0.728 | 0.774/**0.835** | 0.878/0.821 |
| mPWIM$_{seq+tree}$ | 0.855 | 0.870 | 0.743 | **0.623** | **0.718** | **0.735/0.751** | **0.781**/0.821 | **0.887/0.834** |
| Abs increase (%) | 3.3 | 1.7 | - | 2.1 | 2.3 | 2.6/2.8 | 2.2/- | 1.6/2.5 |

Table 1: Test results on different datasets.

the task of natural language inference. The relationship between two sentences includes entailment, contradiction, and neutral.

- Quora (Iyer et al., 2017) consists of 400k question pairs collected from the Quora website, with binary labels indicating if they are duplicates of each other.
- Twitter-URL (Lan et al., 2017) is a paraphrase corpus with 50k sentence pairs.
- PIT-2015 (Xu et al., 2015) is a paraphrase dataset that comes from SemEval-2015 Task 1.
- STS-2014 (Agirre et al., 2014) comes from SemEval-2014 Task 10 and each pair of sentences has a similarity score $\in [0, 5]$.
- WikiQA (Yang et al., 2015) is an open-domain question-answering dataset. After applying the same pre-processing methods in He and Lin (2016), it contains 12k question-answer pairs with binary labels.
- TrecQA (Wang et al., 2007) is from the Text Retrieval Conferences and consists of 56k question-answer pairs.
- SICK (Marelli et al., 2014) comes from SemEval-2014 Task 1 with 10k annotated sentence pairs. Each pair has a similarity score $\in [1, 5]$.

The first seven datasets are the same as the ones examined in Lan and Xu (2018), except for MNLI (Williams et al., 2018), since SNLI is much larger than MNLI for the task of natural language inference. We also add the SICK dataset (Marelli et al., 2014), which is unexplored in Lan and Xu (2018). Across multiple tasks and domains, we systematically compare our proposed models with state-of-the-art neural models: InferSent (Conneau et al., 2017), Shortcut-stacked Sentence Encoder (SSE; Nie and Bansal, 2017), Decomposable Attention Model (DecAtt; Parikh et al., 2016), and Enhanced Sequential Inference Model (ESIM; Chen et al., 2017).

For our experiments on SNLI, Quora, Twitter-URL, PIT-2015, WikiQA and TrecQA, the training objective is to minimize the NLL loss. For STS-2014 and SICK datasets, we use the KL divergence loss. Following He and Lin (2016), for all cases, we use the RMSProp optimizer (Tieleman and Hinton, 2012). Our word representations use 300-dimensional GloVe word vectors (Pennington et al., 2014), which we make static in all experiments. We produce dependency parse trees for each sentence using the Stanford Neural Network Dependency Parser (Chen and Manning, 2014). The TreeLSTM then models sentence representations over each sentence's parse tree.

## 4 Results and Analysis

Table 1 shows the results of our models on different datasets. The first block of the table contains figures copied directly from Lan and Xu (2018); note that they do not use SICK. PWIM$_{our}$ refers to our own implementation. Note that there are at least three independent open-source implementations of the PWIM base model that we are aware of, which confirms the robustness and reproducibility of the model. Most results from these implementations are consistent; however, for

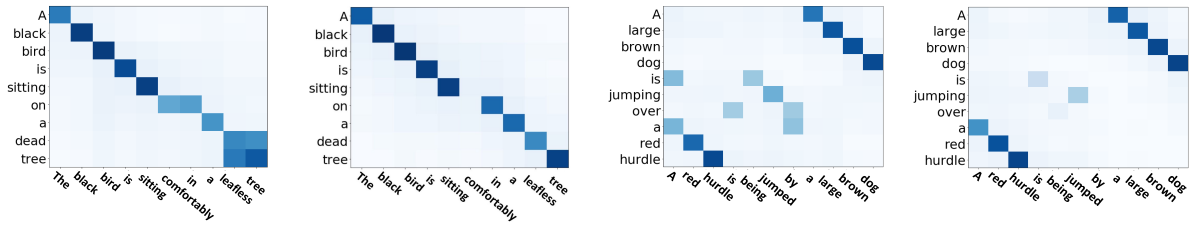


(a) mPWIM$_{seq}$ (b) mPWIM$_{seq+tree}$ (c) mPWIM$_{seq}$ (d) mPWIM$_{seq+tree}$

Figure 1: Visualization of cosine values in the focusCube of two sentence pairs in the SICK test set.

PIT-2015 and STS-2014, we observe some differences, which we were unable to reconcile even after contacting the previous authors. Thus, for comparison purposes, we report results from our base PWIM$_{our}$ implementation.

### 4.1 Effects of the Multi-Layer BiLSTM

The entry mPWIM$_{seq}$ denotes PWIM using multi-layer BiLSTMs for modeling the context of the input sentences and also incorporating residual connections in the final classification. On all datasets listed in the table, adding multi-layer BiLSTMs leads to a higher performance than that of the original model PWIM$_{our}$.

SSE (Nie and Bansal, 2017) is a stacked BiLSTM model with shortcut connections and fine-tuning of word embeddings. Unlike our setting, where each word is represented by its own hidden state in the final output layer, SSE applies max-pooling over time to the output of the last BiLSTM layer to extract the final sentence feature vector. Based on Table 1, mPWIM$_{seq}$ clearly outperforms SSE on Twitter, PIT-2015, STS-2014, WikiQA, and TrecQA. However, for the SNLI and Quora datasets, SSE slightly exceeds mPWIM by 0.4% and 1.6%, respectively. SNLI and Quora have the largest training data among all the datasets with 550k and 393k training sentence pairs, respectively, which suggests that SSE performs better on larger data beyond a certain threshold. We surmise that as the dataset increases in size, the simplicity of SSE will have more performance advantages.

### 4.2 Effects of TreeLSTM

The mPWIM$_{seq+tree}$ further enhances mPWIM$_{seq}$ by incorporating syntactic TreeLSTMs based on syntactic parse trees of each sentence. It averages the prediction probabilities of the PWIM using multi-layer BiLSTMs and TreeLSTMs separately to arrive at the final label. ESIM$_{seq+tree}$ also computes its final predictions by averaging prediction probabilities of two ESIM variants that use BiLSTMs and TreeLSTMs as sentence encoders, respectively.

From the table, we observe that adding TreeLSTMs to the ESIM model only marginally helps or has no effect for most datasets. On the other hand, TreeLSTM complements PWIM well: for WikiQA, it increases mean average precision (MAP) by 1.8% and mean reciprocal rank (MRR) by 2.3%. TreeLSTM also contributes to an 1.1% increase in the F1-measure for PIT-2015, 0.9% Pearson's $r$ for SICK, and 0.7% MAP for TrecQA.

We hypothesize that these observed differences can be attributed to the model architectures. The inference model of ESIM is based on chain LSTMs, which might encode overlapping information with TreeLSTMs. For PWIM, the sentence context information is transformed into pairwise word interaction similarity units, and then a 19-layer-deep CNN exploits the spatially localized patterns. During this process, its focus is word-level similarities in sentences. The syntactic parsing structure introduced by TreeLSTM compensates for some of the information deficiencies. Notably, TreeLSTM does not help PWIM on the Twitter dataset; this makes sense, as Kong et al. (2014) note that many elements in tweets have no syntactic function, including hashtags and URLs. Furthermore, tweets often contain multiple fragments, each with its own syntactic span. Both of these issues may degrade the quality of the syntactic modeling of tweets.

### 4.3 Sample Visualization and Analysis

To better understand *why* our models achieve improved effectiveness, we visualize the cosine values of the focusCube (the final output of the similarity layer) for pairwise word interactions in mPWIM$_{seq}$ and mPWIM$_{seq+tree}$, using the same method as Chen et al. (2017), where darker colors indicate stronger pairwise word interactions.

In Figure 1, we show visualizations from two pairs of sentences from SICK: 1a and 1b form a contrastive pair, as do 1c and 1d. We see that, in both cases, the TreeLSTM helps the model find syntactically important pairwise word interactions. For example, in Figure 1a, for the mPWIM$_{seq}$ model, the cluster of dark patches near the top shows obviously irrelevant correspondences, e.g., "on" with "comfortably", "dead" with "tree", and several of the articles are misaligned with respect to their positions in phrase structure. With the incorporation of syntactic information in Figure 1b, the correspondences are much more accurate.

We see that this is similarly the case when comparing Figures 1c and 1d, where the TreeLSTM yields more accurate correspondences. With the TreeLSTM, the model has learned the correct correspondence between "is being jumped" and "is jumping over", whereas without the syntactic structure, the correspondences are quite muddled. In both cases, we observe that mPWIM$_{seq+tree}$ is able to capture the passive construction for paraphrase detection.

## 5 Conclusion

We examine the hypothesis of whether incorporating contextual and syntactic structures can improve semantic similarity modeling. We extend the strong PWIM model and add additional components comprised of TreeLSTMs and multi-layer BiLSTMs to capture syntax and context information. Thorough experiments on eight datasets show that our improved models achieve consistent gains in effectiveness.

## Acknowledgments

This research was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada, and enabled by computational resources provided by Compute Ontario and Compute Canada.

## References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. SemEval-2014 task 10: multilingual semantic textual similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 81–91.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1657–1668.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680.

Hua He, Kevin Gimpel, and Jimmy Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1576–1586, Lisbon, Portugal.

Hua He and Jimmy Lin. 2016. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 937–948.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

Shankar Iyer, Nikhil Dandekar, and Kornél Csernai. 2017. *First Quora Dataset Release: Question Pairs*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations*.

Lingpeng Kong, Nathan Schneider, Swabha Swayamdipta, Archna Bhatia, Chris Dyer, and Noah A. Smith. 2014. A dependency parser for tweets. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1001–1012.

Wuwei Lan, Siyu Qiu, Hua He, and Wei Xu. 2017. A continuously growing dataset of sentential paraphrases. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1224–1234.

Wuwei Lan and Wei Xu. 2018. Neural network models for paraphrase identification, semantic textual similarity, natural language inference, and question answering. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3890–3902.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014. SemEval-2014 task 1: evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 1–8.

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pages 1045–1048.

Bhaskar Mitra and Nick Craswell. 2019. An introduction to neural information retrieval. *Foundations and Trends in Information Retrieval*, 13(1):1–126.

Yixin Nie and Mohit Bansal. 2017. Shortcut-stacked sentence encoders for multi-domain inference. In *Proceedings of the 2nd Workshop on Evaluating Vector Space Representations for NLP*, pages 41–45.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Jinfeng Rao, Hua He, and Jimmy Lin. 2017. Experiments with convolutional neural network models for answer selection. In *Proceedings of the 40th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2017)*, pages 1217–1220, Tokyo, Japan.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv:1611.01603*.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-RMSProp: divide the gradient by a running average of its recent magnitude. *Coursera: Neural Networks for Machine Learning*, 4(2):26–31.

Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. 2007. What is the Jeopardy model? A quasi-synchronous grammar for QA. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 22–32.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerry Tesauro, Bowen Zhou, and Jing Jiang. 2018. $R^3$: Reinforced ranker-reader for open-domain question answering. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, pages 5981–5988.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 4144–4150.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122.

Wei Xu, Chris Callison-Burch, and Bill Dolan. 2015. SemEval-2015 task 1: paraphrase and semantic similarity in Twitter (PIT). In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 1–11.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. WikiQA: a challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018.

Kai Zhao, Liang Huang, and Mingbo Ma. 2016. Textual entailment with structured attentions and composition. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2248–2258.

# Neural Linguistic Steganography

**Zachary M. Ziegler*** **Yuntian Deng*** **Alexander M. Rush**
School of Engineering and Applied Sciences
Harvard University, Cambridge, MA
{zziegler@g, dengyuntian@g, srush@seas}.harvard.edu

## Abstract

Whereas traditional cryptography encrypts a secret message into an unintelligible form, steganography conceals that communication is taking place by encoding a secret message into a cover signal. Language is a particularly pragmatic cover signal due to its benign occurrence and independence from any one medium. Traditionally, linguistic steganography systems encode secret messages in existing text via synonym substitution or word order rearrangements. Advances in neural language models enable previously impractical generation-based techniques. We propose a steganography technique based on arithmetic coding with large-scale neural language models. We find that our approach can generate realistic looking cover sentences as evaluated by humans, while at the same time preserving security by matching the cover message distribution with the language model distribution.

## 1 Introduction

Cryptography is central to modern communication, but while it effectively conceals the content of a message it reveals that meaningful communication is taking place. Steganography answers an alternative question: how to conceal a message in some cover signal (an image, text etc.) such that an eavesdropper is not even aware any meaningful communication has taken place (Amin et al., 2003; Shirali-Shahreza and Shirali-Shahreza, 2007; Kumar and Pooja, 2010)? Different from cryptography, in steganography security is derived from the inability to detect that a message exists within the cover signal, rather than the inability of an eavesdropper to determine the content of the message (Westfeld and Pfitzmann, 1999).

Natural language is an especially useful cover signal for steganography because it is prevalent and innocuous in everyday life. Furthermore, unlike images or audio in which encoded information depends on exact pixel values, linguistic steganography (hiding information in choices of words) is untethered from any one medium. For example, the sender could encode a message on a computer, read off the cover text in person, and then the receiver could separately enter the cover text into her decoder and retrieve the message.

Linguistic steganography methods can be classified as either edit-based or generation-based (Bennett, 2004). In edit-based methods, such as synonym substitution, the content of the cover text is selected by a human and slightly modified to encode information (Xiang et al., 2017; Hefei, 2009; Xiang et al., 2018). In generative methods an entire block of text is generated while encoding the message reversibly in the choice of tokens. Traditionally, most practical stenography systems are edit-based. These methods only encode a small amount of information (for example, 2 bits per tweet (Wilson and Ker, 2016)), whereas generation-based systems can encode an order of magnitude more information (Fang et al., 2017).

Generation-based steganography has a well-established foundation in information theory (Cox et al., 2005) with the aim of provably fooling any machine steganalysis. Due to weak language models practical performance at the scale of modern neural network-based models has not been considered or developed. One recent exception is a neural network-based approach using heuristic-based methods with the goal of fooling human eavesdroppers. These methods are theoretically sub-optimal, however, and while they demonstrate some fluency in generations the underlying language models are much worse than current state-of-the-art large models (Fang et al., 2017).

In this paper we aim to get the best of both approaches with a linguistic steganography

* Equal contribution.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1210–1215,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

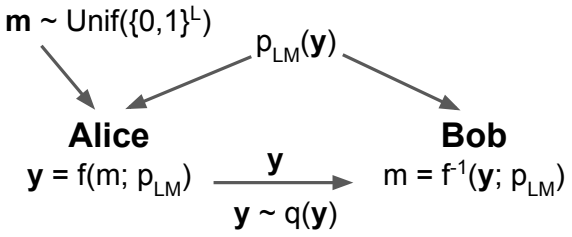


**Figure 1**: Problem setup. $\boldsymbol{m} \sim \text{Unif}(\{0,1\}^L)$ is the secret message, $\boldsymbol{y}$ is the cover text, $p(\boldsymbol{y})$ is the language model and $f$ is a deterministic invertible function. $f$ and the distribution of $m$ implicitly defines $q$.

method based on arithmetic coding and modern language models. Our contribution is two-fold: 1) we show that a linguistic steganography approach combining arithmetic coding with state-of-the-art language models can achieve near-optimal statistical security; 2) human evaluations show that the cover text from our approach is able to fool humans even in the presence of context. Our code is available at https://github.com/harvardnlp/NeuralSteganography. A demo is available at http://steganography.live/

## 2 Background & Related work

**Linguistic steganography** We briefly give an overview of steganography as outlined in (Cox et al., 2005) and diagrammed in Figure 1: Alice wants to communicate a hidden message $\boldsymbol{m} \sim \text{Unif}(\{0,1\}^L)$ with Bob by encoding it into a choice of natural language cover text $\boldsymbol{y}$. The uniform distribution is chosen for $\boldsymbol{m}$ without loss of generality: if $\boldsymbol{m}$ has additional structure it can be further compressed to a uniformly distributed random variable (Han, 2005). Alice and Bob have both agreed on an invertible mapping $f$ which performs the steganography. Alice and Bob also both have access to the exact same language model, $p_{LM}(\boldsymbol{y})$, which $f$ can use during encoding and decoding. The steganography mapping $f$ and the language model $p_{LM}(\boldsymbol{y})$ form the key.

The combination of the distribution of $\boldsymbol{m}$ and deterministic function $\boldsymbol{y} = f(\boldsymbol{m})$ implicitly defines a distribution for $\boldsymbol{y}$ which we denote $q$. This is the cover distribution of natural language that an eavesdropper would observe. As described in (Cox et al., 2005), the security of the system is determined by $D_{KL}(q||P_{true})$ where $P_{true}$ is the true distribution of natural language. For example, if $D_{KL} = 0$ then the system is perfectly secure

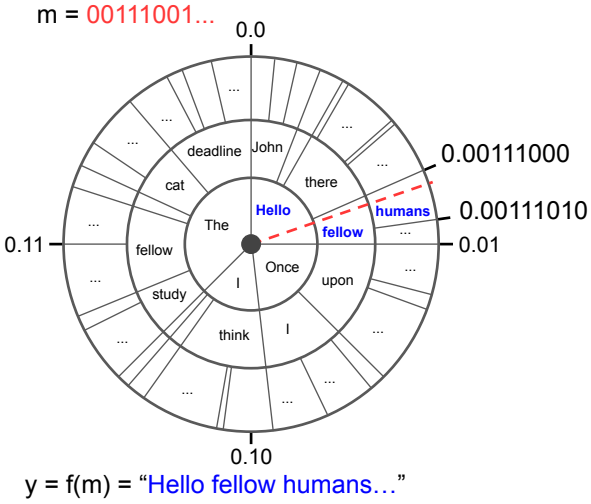


**Figure 2**: Diagram of arithmetic coding for steganography. See Section 3 for details.

because an eavesdropper could not distinguish the encoded messages from normal language. From an information theoretic perspective the security objective of steganography is therefore to ensure a small $D_{KL}$. At the same time, because $f$ is invertible the average number of bits that can be encoded is given by the entropy $H(q)$. Thus, the compression objective is to maximize $H(q)$.

**Generative coding techniques** Recent work studying generative approaches for linguistic steganography have considered heuristic approaches to map uniform messages into similar distributions as natural language. Fang et al. (2017) considers a block-based approach (**Block**), where the vocabulary is randomly split into $2^{|B|}$ bins and each bin is indexed by a $|B|$-bit string. Encoding is performed by splitting the message up into chunks $B_t$ and taking the maximum likelihood token as determined by the language model $p(y_t|y_{<t})$ that falls in bin $B_t$. Yang et al. (2019) proposes a related method based on constructing a Huffman coding (**Huffman**) at each step $t$ based on $p(y_t|y_{<t})$ and encoding the message in variable length chunks via the Huffman tree. Both objectives provide a "quality" parameter that trades off quality at the expense of encoding fewer bits.

These approaches target a different security objective than the information-theoretic view in Cox et al. (2005). Instead of aiming to minimize the KL at maximal compression they aim to maximize generated quality at maximal compression (Yang et al., 2019; Fang et al., 2017). The issue with these approaches is that they aim to fool human eavesdroppers for whom generation quality

| | |
|---|---|
| **Context** | kim jong il was the enigmatic leader of the most enigmatic country on earth. much about kim's life was, and is, shrouded in mystery. even the year of his birth – 1942 – has been contested by knowledgeable sources in south korea and russia, who claim kim was born a few years earlier but that his official birth date was pushed back to make him exactly 30 years younger than his father, north korea's founding leader, kim il sung. |
| **Message 1** | Bob, I have something very important to tell you. Eve is not who she says she is. I know that you think she's your friend, but really she's a creepy stalker who wants to read all of your messages for some unknown reason. |
| **Cover text 1** | That kim's birthdate was not even known to the people of south korea is even more baffling. however, there is another contradiction. bak tae-jung, the ruler of north korea at the time, claimed to have inherited kim's kim ju-ha-teh, his so-called golden heritage, from his father, the last kim Yun-jong. and he also |
| **Message 2** | The secret code is X3SJ83F. Once you open the door you will see a staircase on your left and a hallway to your right. Go down the hallway and enter the code in the safe at the end of the hallway. |
| **Cover text 2** | kim il sang, by the way, was not the first leader of north korea, but the first for nearly a thousand years. his leader was kim jong il, the same person the people voted in to power to start with, without their knowledge or consent. though it's been claimed by others that kim il sang fled north korea in 1940 to escape the reprisals, even that is disputed by k |

**Table 1**: Steganography example. Two different encoded messages are produced given the same introductory context. The messages are first converted into bit strings and then mapped to cover text using the arithmetic steganography approach described in Section 4.4.

is most important; but are susceptible to machine-based eavesdroppers which can in principle identify statistical patterns in generations.

Concurrent with this work, Dai and Cai (2019) conduct a related analysis in terms of KL with a modified Huffman algorithm. Experiments consider the distribution of KL values, although no human evaluation is performed.

## 3 Arithmetic coding

Arithmetic coding is a data compression method designed specifically to code strings of elements with a known probability distribution (Rissanen and Langdon, 1979; Zoph et al., 2015). For long strings the coding is optimal; it compresses information to its entropy (Rissanen and Langdon, 1979). In practice, it is often more efficient than Huffman coding because it does not require blocking. Arithmetic coding traditionally maps a string of elements to a uniformly distributed binary string. To use such a coding for steganography we reverse the order: first a (uniformly sampled) message is selected, then the message is mapped to a sequence (words).

The coding scheme is demonstrated in Figure 2. In this work the probability distribution comes from the conditional distributions of a pretrained language model, but for illustration purposes a hand-crafted example distribution is used in the diagram. Concentric circles represent timesteps; the innermost represents $t = 1$, the middle $t = 2$, and the outer $t = 3$. Each circle represents the conditional distribution $p(y_t|y_{<t})$. For example, given that $y_1$ = "Once", $p(y_2|y_1)$ has "upon" and "I" as the only possible tokens with equal probability. The circle diagram spans [0,1) from 0 at the top, clockwise around to 1.

To encode the message into text, the secret message $m$ is viewed as a binary representation of a fraction in the range $[0, 1)$. This fraction uniquely marks a point on the edge of the circle, as well as a line from the origin to the point. Encoding is performed by simply reading off the tokens corresponding to the bins. Encoding stops when the list of tokens unambiguously defines the message.

Decoding is performed via the reverse operation: the sequence of natural language tokens progressively narrows the range of possible messages. Assuming that the original message is encoded with a predetermined end token, decoding terminates once both sides of the range of possible messages includes the end token.

Sallee (2004) show that just as arithmetic coding is optimal for data compression, it is also optimal for steganography. Specifically, given any goal distribution $p_s(\boldsymbol{y})$ that one wants to sample from, starting with a uniform $\boldsymbol{m}$ and applying the deterministic steganography procedures yields a distribution $q = p_s$ or equivalently $D_{KL}(q||p_s) = 0$ for long sequences. This further ensures that $H(q) = H(p_s)$ and therefore the number of bits encoded on average is equal to the entropy of $p_s$. Intuitively, this works because higher probability sequences map to larger "chunks" of the pie and therefore require fewer bits to uniquely determine.

In this work we apply arithmetic coding to lin-

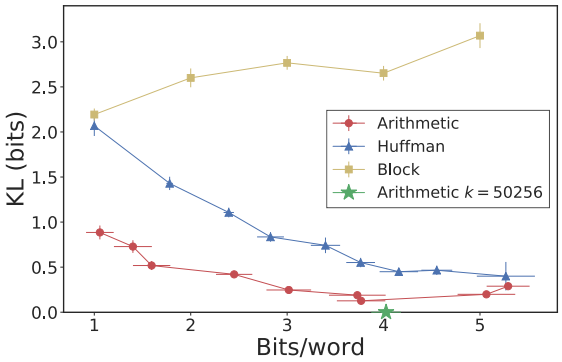


**Figure 3**: Information theoretic evaluation. KL and bits/word are evaluated for each algorithm at different values of the tuning parameter; each datapoint gives the mean and standard error for repeated samples with a fixed value of the tuning parameter. We show the standard error bars for both bits/word and KL.

guistic steganography. Given a pretrained language model, this approach can be directly applied to produce cover texts given some preliminary context. To ensure that these texts seem more natural to human eavesdroppers we introduce two parameters to trade off quality for compression: we modulate the LM distribution by a temperature $\tau$, and we truncate the distribution to a top-$k$ tokens at each position to ensure generations do not include tokens from the long tail.

In practice, a naive implementation of arithmetic coding quickly runs into precision limitations. We use a common equivalent variant based on fixed precision binary fractions (Rubin, 1979).

## 4 Experiments and results

### 4.1 Experimental Setup

We use the 345M parameter GPT-2 model as our language model $p_{LM}$ (Radford et al., 2019). We run our experiments on the CNN/Dailymail (CNNDM) dataset (Hermann et al., 2015; Nallapati et al., 2016), which is used to provide context for generation[1]. We take the first three sentences of each news article as the context on which to condition our language model, and use the indicated steganography algorithm to generate an entire sentence for estimation or evaluation given a uniform random message. We compare the proposed arithmetic coding-based algorithm with the Block (Fang et al., 2017) and Huffman (Yang et al., 2019) methods as baselines. The baselines are re-implemented to allow for a fair comparison using the strong GPT-2 language model. Each method has a parameter that controls the various tradeoffs: for arithmetic encoding we vary the temperature from 0.4 to 1.2 with $k = 300$, for Huffman encoding we vary the truncation length from $2^1$ to $2^8$, for block encoding we vary the block size from $2^1$ to $2^5$. For the KL analysis we also evaluate arithmetic coding with $\tau = 1$ and $k = 50256$, i.e. the unmodulated language model.

[1]Conditional generation makes it more challenging to fool humans and is thus better for comparisons. CNNDM presents an additional challenge as poor generations may have clear factual errors.

### 4.2 Metrics

We consider several different metrics to assess performance. In all cases, it is important that the algorithms encode information efficiently. The metric **Bits/word** is the ratio of message bits encoded to sentence length.

Another key aspect of the system is being able to fool a statistical adversary in a *distributional* sense, i.e. when the adversary gets multiple samples from $q$. This can be quantified by the **KL** divergence between $q$ and $P_{true}$. Since we do not have access to $P_{true}$ we resort to $D_{KL}(q||p_{LM})$. $p_{LM}$ might not match the distribution of natural language, but it allows for relative comparisons of steganography algorithms. The remaining gap is a function of language model performance, which is not the focus of this work. For each model the KL is estimated via MC sampling on the news data.

Finally it is important to be able to convince a **human adversary** that the text is natural. This differs from statistical considerations because a human will only see a single sample from $q$ at a time. To this end, we conduct a human study aligned with the information theoretic evaluation above, using Amazon Mechanical Turk. For each of the three methods we generate sentences with randomly sampled messages conditioned on a news article context, and ask participants to indicate if the sentence is a likely next sentence in context. We repeat this experiment with the true next sentence as well. See the Supplementary Materials for more details.

### 4.3 Quantitative results

Figure 3 shows the information theoretic evaluation. For any bits/word in the range of (1, 5), arithmetic coding achieves the lowest KL, i.e. the distribution of the encoded texts is the most similar to the language model distribution, providing the best defensive against algorithmic detections. Most striking, arithmetic coding with the unmodu-

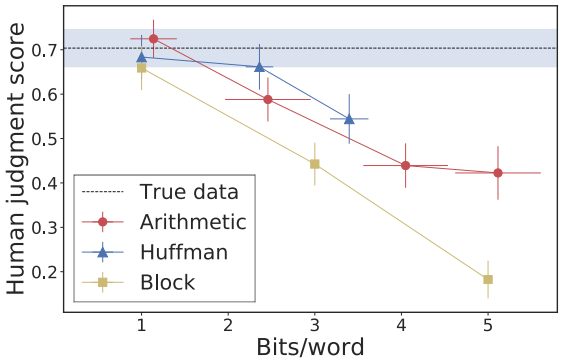


**Figure 4**: Human evaluation. Y axis shows the percentage of humans considering the generation to be a proper continuation of the context. Huffman coding achieves the best generation quality as judged by humans, while arithmetic coding is comparable (the difference is not statistically significant).

lated language model induces a $q$ distribution with a KL of 4e-8 nats. This indicates that, consistent with theory (Sallee, 2004), arithmetic coding enables generative steganography matching the *exact* distribution of the language model used. The small but positive KL value for arithmetic coding with $k = 300$ comes from the slight distributional difference when the long tail is truncated.

Figure 4 shows the human evaluation results. Within 3 bits/word both Huffman and arithmetic coding give plausible next sentences over 50% of the time, and at 1 bit/word both methods lead to cover texts statistically indistinguishable from human-written sentences. Although Huffman coding reaches slightly better performance than arithmetic coding, the transferred cover text distribution is statistically different from natural text (see Figure 3), which can be potentially discovered by automatic systems. Compared to the KL analysis which focuses on the relative performance of the algorithms, the human evaluation highlights the realizable generation quality enabled specifically by large pretrained language models. Still, there is room for improvement in terms of language modeling capabilities: whereas the arithmetic approach reaches minimum KL around 4 bits/word with $\tau = 1.0$, only at 1 bit/word with $\tau = 0.4$ do the generations fool humans.

### 4.4 Qualitative results

Steganography fundamentally deals with hiding uniformly random message bits in cover text. In many cases, however, the message may itself be natural language. In these cases we have an additional choice of how the text is converted to bits. One option is simply the unicode encoding (i.e. UTF-8), though this is highly inefficient. Instead, we can use the same arithmetic coding algorithm in reverse, with an empty context, to compress the message text into near-uniformly distributed bits (Han, 2005). The full procedure is as follows:

1. Alice decodes message text into bits via arithmetic coding with an empty context.
2. Alice encodes bits into cover text via arithmetic coding with a predetermined context.
3. Alice sends cover text over public channel.
4. Bob decodes cover text into bits via arithmetic coding with the predetermined context.
5. Bob encodes bits into message text via arithmetic coding with empty context.

Because the same strong LM is used with different contexts for the two applications of arithmetic coding, the message text length and cover text length will be comparable (modulo a small overhead) but the content of the text can be completely different. This is demonstrated in Table 1, which shows two examples of natural language messages (Step 1) encoded in unrelated and innocuous cover text (Step 3). In both cases the message is encoded efficiently into cover text which is largely fluent and coherent, and follows from the given context.

## 5 Conclusion

We demonstrate how the combination of large pretrained language models and an arithmetic coding-based steganography algorithm allows for a steganography system that can fool both statistical adversaries who analyze the full generative distribution, and human adversaries who analyze a single sample. While our work shows the potential for high-quality efficient steganography and the realizable optimality of arithmetic coding, future advancements in language modeling can push steganographic performance even further. With current state-of-the-art language models the steganography algorithms studied generate most convincing cover texts at low compression, where the KL is moderate. As language models continue to improve, they can be directly plugged into our arithmetic approach to maximize steganographic performance.

## Acknowledgments

This work was supported by NSF 1845664. YD was supported by a Baidu AI Fellowship.

## References

Muhalim Mohamed Amin, Mazleena Salleh, Subariah Ibrahim, Mohd Rozi Katmin, and MZI Shamsuddin. 2003. Information hiding using steganography. In *4th National Conference of Telecommunication Technology, 2003. NCTT 2003 Proceedings.*, pages 21–25. IEEE.

Krista Bennett. 2004. Linguistic steganography: Survey, analysis, and robustness concerns for hiding information in text.

Ingemar J Cox, Ton Kalker, Georg Pakura, and Mathias Scheel. 2005. Information transmission and steganography. In *International Workshop on Digital Watermarking*, pages 15–29. Springer.

Falcon Dai and Zheng Cai. 2019. Towards near-imperceptible steganographic text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4303–4308, Florence, Italy. Association for Computational Linguistics.

Tina Fang, Martin Jaggi, and Katerina Argyraki. 2017. Generating steganographic text with lstms. *arXiv preprint arXiv:1705.10742*.

Te Sun Han. 2005. Folklore in source coding: Information-spectrum approach. *IEEE Trans. Inf. Theor.*, 51(2):747–753.

PR Hefei. 2009. Steganalysis of synonym-substitution based natural language watermarking. *International Journal of Multimedia and Ubiquitous Engineering*, 4(2).

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Arvind Kumar and Km Pooja. 2010. Steganography-a data hiding technique. *International Journal of Computer Applications*, 9(7):19–23.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Jorma Rissanen and Glen G Langdon. 1979. Arithmetic coding. *IBM Journal of research and development*, 23(2):149–162.

Frank Rubin. 1979. Arithmetic stream coding using fixed precision registers. *IEEE Transactions on Information Theory*, 25(6):672–675.

P. Sallee. 2004. Model based steganography. *Int. Workshop on Digital Watermarking*.

Mohammad Shirali-Shahreza and M Hassan Shirali-Shahreza. 2007. Text steganography in sms. In *2007 International Conference on Convergence Information Technology (ICCIT 2007)*, pages 2260–2265. IEEE.

Andreas Westfeld and Andreas Pfitzmann. 1999. Attacks on steganographic systems. In *International workshop on information hiding*, pages 61–76. Springer.

Alex Wilson and Andrew D Ker. 2016. Avoiding detection on twitter: embedding strategies for linguistic steganography. *Electronic Imaging*, 2016(8):1–9.

Lingyun Xiang, Yan Li, Wei Hao, Peng Yang, and Xiaobo Shen. 2018. Reversible natural language watermarking using synonym substitution and arithmetic coding. *Comput., Mater. Continua*, 55(3):541–559.

Lingyun Xiang, Xinhui Wang, Chunfang Yang, and Peng Liu. 2017. A novel linguistic steganography based on synonym run-length encoding. *IEICE transactions on Information and Systems*, 100(2):313–322.

Zhong-Liang Yang, Xiao-Qing Guo, Zi-Ming Chen, Yong-Feng Huang, and Yu-Jin Zhang. 2019. Rnn-stega: Linguistic steganography based on recurrent neural networks. *IEEE Transactions on Information Forensics and Security*, 14(5):1280–1295.

Barret Zoph, Marjan Ghazvininejad, and Kevin Knight. 2015. How much information does a human translator add to the original? In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 889–898.

# The Feasibility of Embedding Based Automatic Evaluation for Single Document Summarization

**Simeng Sun**
University of Pennsylvania
simsun@seas.upenn.edu

**Ani Nenkova**
University of Pennsylvania
nenkova@seas.upenn.edu

## Abstract

ROUGE is widely used to automatically evaluate summarization systems. However, ROUGE measures semantic overlap between a system summary and a human reference on word-string level, much at odds with the contemporary treatment of semantic meaning. Here we present a suite of experiments on using distributed representations for evaluating summarizers, both in reference-based and in reference-free setting. Our experimental results show that the max value over each dimension of the summary ELMo word embeddings is a good representation that results in high correlation with human ratings. Averaging the cosine similarity of all encoders we tested yields high correlation with manual scores in reference-free setting. The distributed representations outperform ROUGE in recent corpora for abstractive news summarization but are less good on older test data and systems.

## 1 Introduction

The widely used ROUGE (Lin, 2004) automatic evaluation for summarization relies on token overlap between reference and system summary. This limited view of meaning has motivated numerous studies on summarization evaluation (Zhou et al., 2006; Ganesan, 2018; ShafieiBavani et al., 2018), and the related areas of translation and dialog, to explore more compelling semantic matching (Kauchak and Barzilay, 2006; Lavie and Denkowski, 2009; Lo and Wu, 2011; Chen and Guo, 2015; Liu et al., 2016; Tao et al., 2018). Most recently, incorporating word embeddings in ROUGE pairwise comparison of $n$-grams has proven beneficial (Ng and Abrecht, 2015), as well as representing sentences using universal sentence representation to predict the quality of translation (Shimanaka et al., 2018).

We build upon this line of work and show that cosine similarity between the reference and summary embedding works well, and better than ROUGE on recent datasets, for comparing single document summarization systems. Unlike prior work (Ng and Abrecht, 2015), we thoroughly abandon ROUGE and $n$-gram co-occurrences in the computation of semantic similarity. To give a sense of the generalizability of our findings, we validate the method on three different test sets with human evaluation. We compare several popular representation including sentence embedding, un-contextualized word embedding and contextualized word embedding. Finally, we present experiments on evaluating single document summaries without reference summaries which was originally proposed for multi-document summarization (Louis and Nenkova, 2013) and explored a variety of word-string similarity techniques. Here we study reference-free evaluations via embedding similarity between the full document to be summarized and the system summaries.

## 2 Embeddings

To get a dense low-dimenional representation of texts, we test seven representations covering sentence embedding, variants of un-contextualized word embedding and variants of contextualized word embedding. Specifically: *(i)* Two Google universal sentence encoders: (Cer et al., 2018), an encoder (*enc-2*) based on deep averaging net (Iyyer et al., 2015) and an encoder (*enc-3*) based on transformer (Vaswani et al., 2017). Both encoders encode input text to 512-dimensional vector. *(ii)* Average (*ELMo-a*) and max (*ELMo-m*) over each dimension of all ELMo (Peters et al., 2018) word embeddings of an input text. For each token in the input, three layers of 1,024-dimensional vectors were concatenated to form a 3,072-dimensional vector. *(iii)* Average (*avg*) and max (*max*) over GoogleNews 300-d word2vec.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1216–1221, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| | DUC2001 | | DUC2002 | | | DUC2001 | | DUC2002 | |
|---|---|---|---|---|---|---|---|---|---|
| **ROUGE** | **P** | **S** | **P** | **S** | **Encoder** | **P** (ref / doc) | **S** (ref / doc) | **P** (ref / doc) | **S** (ref / doc) |
| R1-R | 0.683* | 0.650* | 0.809* | 0.783* | enc-2 | 0.661* / 0.676* | 0.538 / 0.580* | 0.837* / 0.734* | **0.867*** / 0.566 |
| R1-P | 0.315 | 0.441 | 0.314 | 0.287 | enc-3 | 0.505 / 0.628* | 0.378 / 0.580* | 0.856* / 0.724* | 0.790* / 0.566 |
| R1-F | **0.835*** | **0.825*** | 0.870* | 0.818* | max | 0.625* / 0.106 | **0.685*** / 0.035 | **0.890*** / 0.475 | 0.692* / 0.462 |
| R2-R | 0.810* | 0.811* | **0.954*** | **0.944*** | avg | 0.473 / 0.184 | 0.476 / 0.063 | 0.876* / 0.446 | 0.748* / 0.420 |
| R2-P | 0.538 | 0.594 | 0.644* | 0.622* | InferSent | 0.445 / 0.130 | 0.483 / 0.063 | 0.878* / 0.486 | 0.762* / 0.503 |
| R2-F | 0.773* | 0.720* | 0.879* | 0.881* | ELMo-m | 0.652* / **0.739*** | 0.664* / **0.776*** | 0.881* / 0.693* | 0.790* / **0.692*** |
| - | - | - | - | - | ELMo-a | **0.743*** / 0.652* | 0.573 / 0.692* | 0.710* / 0.678* | 0.643* / 0.650* |
| - | - | - | - | - | enc-avg | 0.659* / 0.725* | 0.587* / 0.622* | 0.888* / **0.763*** | 0.748* / 0.685* |

Table 1: Correlation results on DUC2001/2002. P is shorthand for Pearson correlation, S for Spearman correlation. Detailed description of encoders can be found in §2. (ref / doc) shows the correlation between summary embedding with either reference embedding or document embedding. Entries with $p$-value lower than 0.05 are marked with *. ROUGE F1 scores, the commonly reported metric, are underlined for better comparison.

*(iv)* InferSent (Conneau et al., 2017) (*InferSent*), a BiLSTM encoder producing representation of 4,096 dimensions. We compute cosine similarity between summary and reference embedding to capture semantic similarity. To test the robustness of this evaluation approach, we check correlations on old single document summarization evaluations of somewhat obsolete systems and modern corpora for summarization with a mix of extractive and neural abstractive systems.

## 3 Evaluation on DUC2001/2002

Document Understanding Conferences (DUC) 2001/2002 provide benchmark datasets along with human evaluation over multiple submitted systems. Human evaluation (*coverage score*) reflects the degree to which semantic units, roughly clauses, in the reference summary are expressed in the system summary (Lin and Hovy, 2003).

To evaluate the newly proposed automatic evaluations, we follow the conventional methodology of computing correlation between the automatic metric and human evaluations of summary content. The results are shown in Table 1[1]. R1-F correlates better with human ratings on DUC'01, while R2-R works extremely well on DUC'02. Both uni- and bi-gram ROUGE F-measure also correlate well with human evaluation, which is an important finding given that ROUGE-F has become the de facto standard for evaluation of neural summarization systems (Nallapati et al., 2016; See et al., 2017; Gehrmann et al., 2018; Zhang et al., 2018; Celikyilmaz et al., 2018).

[1]There are total 14 systems in DUC'02. We discard two poorly performing systems, 17 and 30. Including them in the analysis results in high correlation ($> 0.9$) for both ROUGE and embedding similarity but the results we present are more convincing without the presence of clearly inferior systems.

We find that there is no single optimal representation that gives the best correlation on both data sets. There is a clear increase of performance from DUC'01 to DUC'02. In DUC'02, embedding similarity can achieve the same level or even higher correlation with human evaluation than ROUGE F-measure, but it performs worse on DUC'01. The correlations using *avg*, *max* and *InferSent* in reference-free setting on DUC'01 are lower.

| **Statistics** | **DUC2001** | **DUC2002** |
|---|---|---|
| # of systems | 12 | 12 |
| Avg. reference length | 116.595 | 113.607 |
| Avg. summarry length | 114.598 | 114.184 |
| Avg. document length | 841.000 | 650.053 |
| Min coverage score | 0.320 | 0.346 |
| Max coverage range | 0.453 | 0.4975 |
| Std sys coverage score | 0.048 | 0.049 |
| Avg. annotated article per sys | 148.000 | 293.250 |

Table 2: DUC2001 and DUC2002 dataset statistics

To understand what causes the low performance on DUC'01, we examined statistics for both datasets, shown in Table 2. Systems in DUC'01 are, on average, inferior to DUC'02 systems (lines 5 and 6) and systems in DUC'01 are more similar to each other than in DUC'02 (line 7), which leaves less room to rank the systems and thus achieve high correlation. Another difference between DUC'01 and DUC'02 is that the number of evaluated articles for each system is considerably larger than that in DUC'01. One might ask if enough articles are provided for each system on DUC'01 data. We show in Figure 1 that 140 is a large enough number for eliciting stable system-level correlation. Another possible problem is that the number of systems is not enough, so a minor change in either human or embedding similarity score can lead to large oscillation of correlation. Scatter plots of these systems are shown in Figure

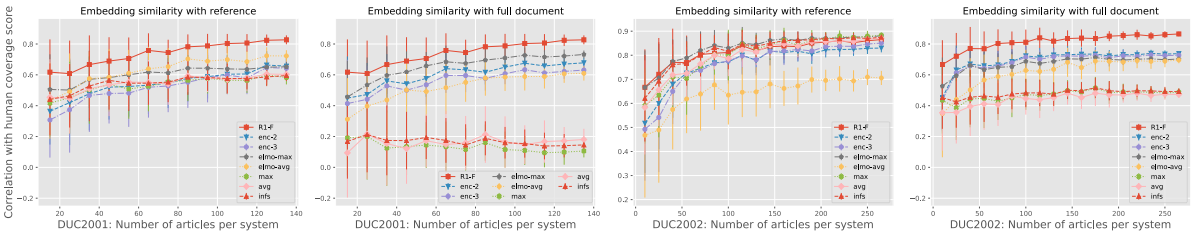


Figure 1: For each number of articles, we sample and compute the correlation for 50 times and plot the average as well as standard deviation. The decreasing size of error bar shows that enough articles are provided for each system and it is not the reason of the performance discrepancy between DUC2001 and DUC2002.

3. As we can see, similarity of all kinds of embeddings indeed correlate with coverage score, however they also generate more extreme values when pairs of systems are examined. For example, two systems may be close in terms of R1-F, but can be relatively distant when comparing the embedding similarities. This problem, possibly due to different architectures and different data each encoder is trained on, may be alleviated by averaging the cosine similarities computed from all the representations. Overall, given the stable results on DUC'02, embedding similarity is a good metric which does not depend on lexical overlap and can be computed quickly immediately after inference without the need of running ROUGE.

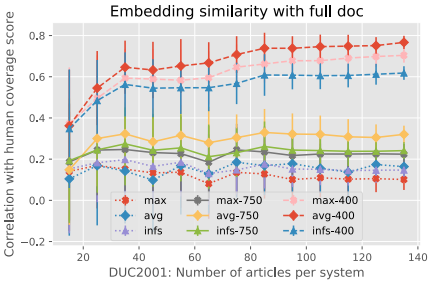


Figure 2: Truncated articles lead to performance improvement for *max*, *avg* and *InferSent* representation.

DUC'01 — where reference-free results are weaker — contains longer articles. To check if representations are sensitive to input length, we truncated the articles to the first 400 or 750 tokens when the length exceeds that limit. We plot the correlations in this setting for the three encoders which have worse results on DUC'01. Figure 2 shows a clear and consistent improvement of correlation when the document size is smaller. In fact, when only the lead 400 tokens are included, the average word embedding is only slightly worse than R1-F. This finding suggests that the embedding of the document lead sentences serves as a better reference than the full document and that the three representations are sensitive to input length.

Another noticeable difference between DUC'01 and DUC'02 is the performance of *ELMo-a* word embeddings. Both ELMo embedding variants are capable of dealing with long texts. For the four settings we tested, *ELMo-m* has better Spearman correlation than averaging ELMo embeddings. On DUC'02, *ELMo-a* leads to lower correlations than other encoders. Unlike *max* and *avg* where word embeddings are fixed, contextualized word embeddings are more flexible, thus the average ELMo embeddings for reference and summary can be far away from each other, in this case *ELMo-m* reflects more salient information about the input than *ELMo-a*.

| **ROUGE** | **CN** | **IN** | **RL** | **SR** | **UC** | **VE** |
|---|---|---|---|---|---|---|
| R1-R | 0.086 | **0.943** | 0.486 | **0.943** | 0.314 | 0.371 |
| R1-P | 0.257 | 0.257 | **0.714** | 0.600 | **0.714** | **0.829** |
| R1-F | 0.257 | 0.543 | 0.886 | 0.829 | 0.657 | **0.829** |
| R2-R | -0.086 | 0.771 | 0.429 | **0.943** | 0.429 | 0.486 |
| R2-P | 0.086 | 0.371 | 0.486 | 0.657 | 0.600 | 0.657 |
| R2-F | **0.717** | 0.131 | 0.179 | 0.097 | 0.368 | 0.628 |
| **Enc-ref** | **CN** | **IN** | **RL** | **SR** | **UC** | **VE** |
| enc-2 | 0.200 | 0.714 | **0.771** | **0.943** | **0.600** | **0.714** |
| enc-3 | 0.200 | 0.714 | **0.771** | **0.943** | **0.600** | **0.714** |
| max | 0.257 | 0.829 | 0.714 | 0.886 | 0.429 | 0.543 |
| avg | 0.086 | 0.943 | 0.486 | **0.943** | 0.314 | 0.371 |
| InferSent | 0.086 | 0.943 | 0.486 | **0.943** | 0.314 | 0.371 |
| ELMo-m | 0.029 | **1.000** | 0.314 | 0.829 | 0.086 | 0.143 |
| ELMo-a | 0.086 | 0.943 | 0.486 | **0.943** | 0.314 | 0.371 |
| enc-avg | 0.200 | 0.714 | **0.771** | **0.943** | **0.600** | **0.714** |
| **Enc-doc** | **CN** | **IN** | **RL** | **SR** | **UC** | **VE** |
| enc-2 | -0.086 | 0.943 | 0.200 | 0.771 | 0.029 | 0.086 |
| enc-3 | -0.086 | 0.943 | 0.200 | 0.771 | 0.029 | 0.086 |
| max | 0.143 | 0.943 | 0.371 | 0.657 | -0.029 | 0.086 |
| avg | 0.314 | 0.829 | 0.257 | 0.429 | -0.086 | -0.029 |
| InferSent | 0.143 | 0.943 | 0.371 | 0.657 | -0.029 | 0.086 |
| ELMo-m | 0.143 | 0.943 | 0.371 | 0.657 | -0.029 | 0.086 |
| ELMo-a | 0.143 | 0.943 | 0.371 | 0.657 | -0.029 | 0.086 |
| enc-avg | 0.029 | **1.000** | 0.314 | **0.829** | 0.086 | 0.143 |

Table 3: Spearman correlation on Newsroom data. Pearson correlation results are in Appendix. Enc-ref/doc refers to embedding similarity with reference/full document.

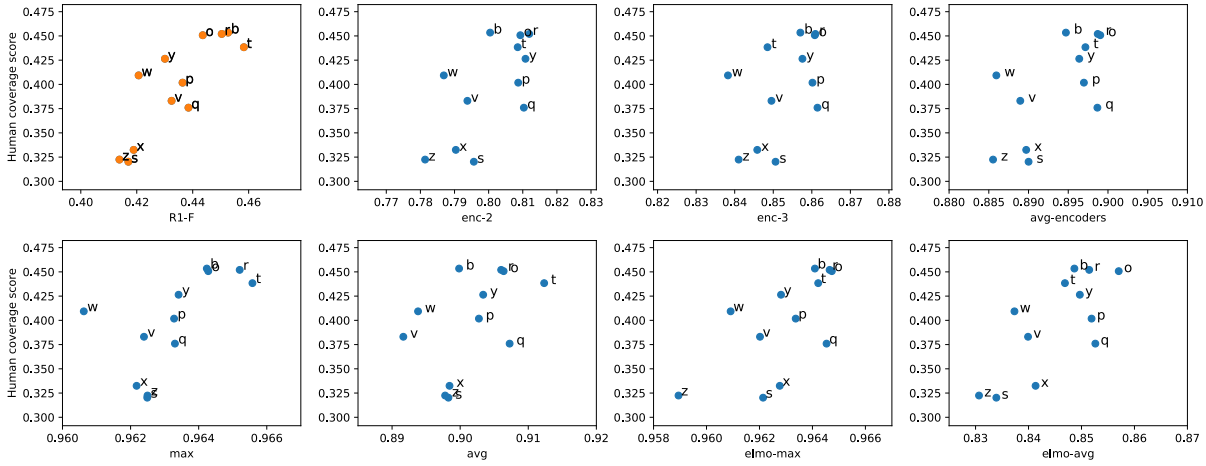


Figure 3: This scatter plot shows the human coverage score and embedding similarity on DUC2001. The baseline system is shortened to 'b'.

## 4 Evaluation on Newsroom 60

In this section, we use contemporary data and systems and explore other factors that embedding similarity could potentially capture. We employ the human evaluation set from newsroom data introduced in (Grusky et al., 2018). The evaluation data includes 7 systems, each producing summaries for 60 articles. The 7 systems are: (1) lead3 sentences of the article (2) textrank with word limit of 50 (3) extractive oracle 'fragments' system, representing the best possible performance of an extractive system (4) abstractive model (Rush et al., 2015) trained on Newsroom training data (5) Pointer-Generator (See et al., 2017) trained on CNN/DailyMail data set (Nallapati et al., 2016), on complete and a subset of Newsroom training set respectively.

We collected crowdsourced evaluations of relevance *(RL)* and informativeness *(IN)* as introduced in the original paper, closely reproducing the earlier findings. We introduce four more dimensions: verbosity *(VE)*, unnecessary content *(UC)*, perfect surrogate *(SR)* and continue reading *(CN)*. Higher rating corresponds to assessment that the summary is not unnecessarily verbose, it has no unnecessary content, it is a good surrogate for the input and that much additional information can be obtained from the article after reading the summary. The exact questions are presented in the supplementary material. We asked workers to rate in the range of 1 to 7 instead of 1 to 5 in the original paper. We excluded from the analysis the 'fragments' oracle system which maximizes ROUGE by selecting word $n$-grams but receives very low human ratings because the resulting summary is incomprehensible. Each summary is scored by three crowdworkers whose scores we average. Table 3 shows Spearman correlations and Pearson correlations are in the Appendix.

| **Eval by rank** | **P**(ref) | **S**(ref) | **P**(doc) | **S**(doc) |
|---|---|---|---|---|
| enc-2 | 6.667 | 5.000 | 5.667 | 4.667 |
| enc-3 | 8.000 | 6.667 | 6.333 | 4.667 |
| max | 4.333 | 5.667 | 7.333 | 7.000 |
| avg | 5.667 | 6.000 | 8.333 | 8.667 |
| InferSent | 6.333 | 5.333 | 7.000 | 6.000 |
| ELMo-m | **4.000** | **3.000** | 3.667 | **2.333** |
| ELMo-a | 4.667 | 6.000 | 6.000 | 3.667 |
| enc-avg | 4.333 | 6.000 | **2.667** | 3.333 |
| R1-F | 5.667 | 4.667 | 3.667 | 4.000 |
| R2-F | 5.333 | 4.000 | 4.667 | 4.667 |

Table 4: Average correlation rank of each system on DUC2001, DUC2002 and Newsroom 60 data. P is shorthand for Pearson correlation and S for Spearman.

The table shows that embedding similarity correlates better or the same as ROUGE with human ratings on informativeness, relevance and surrogate. However, ROUGE precision is a more suitable metric for evaluating the extent of unnecessary content and verbosity. This implies that the examined representations capture well the meaning of the input but store repetitive information without enough penalty. For the dimension *CN*, showing that the input contains considerably more important details than in the summary, neither embedding similarity nor ROUGE is good enough to show a correlation with small $p$-value.

To establish which representation correlates

best with human ratings, we sort the representations by correlations in descending order on the three data sets we examined and compute the average rank for each. We used informativeness ratings on Newsroom data. Smaller value means overall better performance on all three data sets. The results are shown in Table 4. The max value over each dimension of ELMo word embedding of input text performs the best when reference summaries are given. When we compare the embedding between system summary and document, the averaged cosine similarities of all seven representation also gives good results. Most importantly, the *ELMo-m* evaluation ranks consistently better than ROUGE-F for all evaluation settings.

## 5 Conclusion

In this paper we systematically study embedding cosine similarity as a measure of the quality of summarizers on three data sets. We verify the feasibility of the embedding similarity for system comparison on DUC'01, DUC'02 and Newsroom human evaluation data. The worse results on DUC'01 can be explained by the fact that systems being evaluated are too similar and not that well-performing. On DUC'02 and Newsroom data, embedding similarity can achieve the same level or even higher correlation with human ratings compared to ROUGE. Overall, when references are given, max ELMo word embeddings have highest correlation and the averaged cosine similarities of the examined representations gives high correlation in reference-free setting.

## References


Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep communicating agents for abstractive summarization. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1662–1675. Association for Computational Linguistics.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder for english. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174. Association for Computational Linguistics.

Boxing Chen and Hongyu Guo. 2015. Representation based translation evaluation metrics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 150–155. Association for Computational Linguistics.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

Kavita Ganesan. 2018. Rouge 2.0: Updated and improved measures for evaluation of summarization tasks.

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109. Association for Computational Linguistics.

Max Grusky, Mor Naaman, and Yoav Artzi. 2018. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 708–719. Association for Computational Linguistics.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1681–1691. Association for Computational Linguistics.

David Kauchak and Regina Barzilay. 2006. Paraphrasing for automatic evaluation. In *Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 4-9, 2006, New York, New York, USA*.

Alon Lavie and Michael J. Denkowski. 2009. The meteor metric for automatic evaluation of machine translation. *Machine Translation*, 23(2-3):105–115.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Proc. ACL workshop on Text Summarization Branches Out*, page 10.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003*

*Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2122–2132. Association for Computational Linguistics.

Chi-kiu Lo and Dekai Wu. 2011. MEANT: an inexpensive, high-accuracy, semi-automatic metric for evaluating translation utility based on semantic roles. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*, pages 220–229.

Annie Louis and Ani Nenkova. 2013. Automatically assessing machine summary content without a gold standard. *Computational Linguistics*, 39(2):267–300.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290. Association for Computational Linguistics.

Jun-Ping Ng and Viktoria Abrecht. 2015. Better summarization evaluation with word embeddings for rouge. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1925–1930. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083. Association for Computational Linguistics.

Elaheh ShafieiBavani, Mohammad Ebrahimi, Raymond Wong, and Fang Chen. 2018. A graph-theoretic summary evaluation for ROUGE. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 762–767, Brussels, Belgium. Association for Computational Linguistics.

Hiroki Shimanaka, Tomoyuki Kajiwara, and Mamoru Komachi. 2018. Metric for automatic machine translation evaluation based on universal sentence representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 106–111. Association for Computational Linguistics.

Chongyang Tao, Lili Mou, Dongyan Zhao, and Rui Yan. 2018. Ruber: An unsupervised method for automatic evaluation of open-domain dialog systems. In *AAAI*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Xingxing Zhang, Mirella Lapata, Furu Wei, and Ming Zhou. 2018. Neural latent extractive document summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 779–784. Association for Computational Linguistics.

Liang Zhou, Chin-Yew Lin, Dragos Stefan Munteanu, and Eduard Hovy. 2006. ParaEval: Using paraphrases to evaluate summaries automatically. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 447–454, New York City, USA. Association for Computational Linguistics.

# Attention Optimization for Abstractive Document Summarization

**Min Gui[1], Junfeng Tian[1], Rui Wang[1], Zhenglu Yang[2]**
[1] Alibaba Group, China
[2] School of Computer Science, Nankai University, China
{guimin.gm, tjf141457, masi.wr}@alibaba-inc.com
yangzl@nankai.edu.cn

## Abstract

Attention plays a key role in the improvement of sequence-to-sequence-based document summarization models. To obtain a powerful attention helping with reproducing the most salient information and avoiding repetitions, we augment the vanilla attention model from both local and global aspects. We propose an attention refinement unit paired with local variance loss to impose supervision on the attention model at each decoding step, and a global variance loss to optimize the attention distributions of all decoding steps from the global perspective. The performances on the CNN/Daily Mail dataset verify the effectiveness of our methods.

## 1 Introduction

Abstractive document summarization (Rush et al., 2015; Nallapati et al., 2016; Tan et al., 2017; Chen and Bansal, 2018; Celikyilmaz et al., 2018) attempts to produce a condensed representation of the most salient information of the document, aspects of which may not appear as parts of the original input text. One popular framework used in abstractive summarization is the sequence-to-sequence model introduced by Sutskever et al. (2014). The *attention* mechanism (Bahdanau et al., 2014) is proposed to enhance the sequence-to-sequence model by allowing salient features to dynamically come to the forefront as needed to make up for the incapability of memorizing the long input source.

However, when it comes to longer documents, basic attention mechanism may lead to distraction and fail to attend to the relatively salient parts. Therefore, some works focus on designing various attentions to tackle this issue (Tan et al., 2017; Gehrmann et al., 2018). We follow this line of research and propose an effective attention refinement unit (ARU). Consider the following case. Even with a preliminary idea of which parts of source document should be focused on (attention), sometimes people may still have trouble in deciding which exact part should be emphasized for the next word (the output of the decoder). To make a more correct decision on what to write next, people always adjust the concentrated content by reconsidering the current state of what has been summarized already. Thus, ARU is designed as an update unit based on current decoding state, aiming to retain the attention on salient parts but weaken the attention on irrelevant parts of input.

The de facto standard attention mechanism is a soft attention that assigns attention weights to all input encoder states, while according to previous work (Xu et al., 2015; Shankar et al., 2018), a well-trained hard attention on exact one input state is conducive to more accurate results compared to the soft attention. To maintain good performance of hard attention as well as the advantage of end-to-end trainability of soft attention, we introduce a local variance loss to encourage the model to put most of the attention on just a few parts of input states at each decoding step. Additionally, we propose a global variance loss to directly optimize the attention from the global perspective by preventing assigning high weights to the same locations multiple times. The global variance loss is somewhat similar with the coverage mechanism (Tu et al., 2016; See et al., 2017), which is also designed for solving the repetition problem. The coverage mechanism introduces a coverage vector to keep track of previous decisions at each decoding step and adds it into the attention calculation. However, when the high attention on certain position is wrongly assigned during previous timesteps, the coverage mechanism hinders the correct assignment of attention in later steps.

We conduct our experiments on the CNN/Daily Mail dataset and achieve comparable results on ROUGE (Lin, 2004) and METEOR (Denkowski and Lavie, 2014) with the state-of-the-art models.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1222–1228, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

Our model surpasses the strong pointer-generator baseline (w/o coverage) (See et al., 2017) on all ROUGE metrics by a large margin. As far as we know, we are the first to introduce explicit loss functions to optimize the attention. More importantly, the idea behind our model is simple but effective. Our proposal could be applied to improve other attention-based models, which we leave these explorations for the future work.

## 2 Proposed model

### 2.1 Model Architecture

We adopt the Pointer-Generator Network (PGN) (See et al., 2017) as our baseline model, which augments the standard attention-based seq2seq model with a hybrid pointer network (Vinyals et al., 2015). An input document is firstly fed into a Bi-LSTM encoder, then an uni-directional LSTM is used as the decoder to generate the summary word by word. At each decoding step, the attention distribution $a_t$ and the context vector $c_t$ are calculated as follows:

$$e_{ti} = v^T \tanh(W_h h_i + W_s s_t + b_{attn}) \quad (1)$$

$$a_t = \text{softmax}(e_t) \quad (2)$$

$$c_t = \sum_{i=1} a_{ti} h_i \quad (3)$$

where $h_i$ and $s_t$ are the hidden states of the encoder and decoder, respectively. Then, the token-generation softmax layer reads the context vector $c_t$ and current hidden state $s_t$ as inputs to compute the vocabulary distribution. To handle OOVs, we inherit the pointer mechanism to copy rare or unseen words from the input document (refer to See et al. (2017) for more details).

To augment the vanilla attention model, we propose the Attention Refinement Unit (ARU) module to retain the attention on the salient parts while weakening the attention on the irrelevant parts of input. As illustrated in Figure 1, the attention weight distribution $a_t$ at timestep $t$ (the first red histogram) is fed through the ARU module. In the ARU module, current decoding state $s_t$ and attention distribution $a_t$ are combined to calculate a refinement gate $r_t$:

$$r_t = \sigma(W_s^r s_t + W_a^r a_t + b_r) \quad (4)$$

where $\sigma$ is the sigmoid activation function, $W_s^r$, $W_a^r$ and $b_r$ are learnable parameters. $r_t$ represents how much degree of the current attention should be updated. Small value of $r_{ti}$ indicates that the content of $i$-th position is not much relevant to current decoding state $s_t$, and the attention on $i$-th position should be weakened to avoid confusing the model. The attention distribution is updated as follows (the symbol $\odot$ means element-wise product):

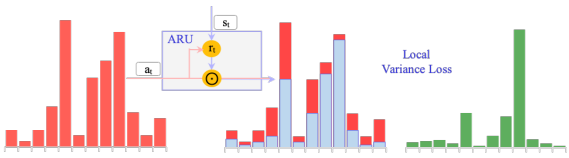


Figure 1: The process of attention optimization (better view in color). The original attention distribution (red bar on the left) is updated by the refinement gate $r_t$ and attention on some irrelevant parts are lowered. Then the updated attention distribution (blue bar in the middle) is further supervised by a local variance loss and get a final distribution (green bar on the right).

$$a_t^r = r_t \odot a_t \quad (5)$$

### 2.2 Local Variance Loss

As discussed in section 1, the attention model putting most of attention weight on just a few parts of the input tends to achieve good performance. Mathematically, when only a small number of values are large, the shape of the distribution is sharp and the variance of the attention distribution is large. Drawing on the concept of variance in mathematics, local variance loss is defined as the reciprocal of its variance expecting the attention model to be able to focus on more salient parts. The standard variance calculation is based on the mean of the distribution. However, as previous work (Huang et al., 1979; Jung et al., 2018) mentioned that the median value is more robust to outliers than the mean value, we use the median value to calculate the variance of the attention distribution. Thus, local variance loss can be calculated as:

$$var(a_t^r) = \frac{1}{|D|} \sum_{i=1}^{|D|} (a_{ti}^r - \hat{a_t^r})^2 \quad (6)$$

$$\mathcal{L}_L = \frac{1}{T} \sum_{t}^{T} \frac{1}{var(a_t^r) + \epsilon} \quad (7)$$

where $\hat{\cdot}$ is a median operator and $\epsilon$ is utilized to avoid zero in the denominator.

### 2.3 Global Variance Loss

To avoid the model attending to the same parts of the input states repeatedly, we propose another variance loss to adjust the attention distribution globally. Ideally, the same locations should be assigned a relatively high attention weight once at most. Different from the coverage mechanism (See et al., 2017; Tu et al., 2016) tracking attention distributions of previous timesteps, we maintain the sum of attention distributions over all decoder timesteps, denoted as $A$. The $i$-th value of $A$ represents the accumulated attention that the input state at $i$-th position has received throughout the whole decoding process. Without repeated high attention being paid to the same location, the difference between the sum of attention weight and maximum attention weight of $i$-th input state among all timesteps should be small. Moreover, the whole distribution of the difference over all input positions should have a flat shape. Similar to the definition of local variance loss, the global variance loss is formulated as:

$$g_i = \sum_t (a^r_{ti}) - \max_t (a^r_{ti}) \tag{8}$$

$$\mathcal{L}_G = \frac{1}{|D|} \sum_{i=1}^{|D|} (g_i - \hat{g})^2 \tag{9}$$

where $g_i$ represents the difference between the accumulated attention weight and maximum attention weight at $i$-th position.

### 2.4 Model Training

The model is firstly pre-trained to minimize the maximum-likelihood loss, which is widely used in sequence generation tasks. We define $y^* = \{y^*_1, \cdots, y^*_T\}$ as the ground-truth output sequence for a given input sequence $x$, then the loss function is formulated as:

$$\mathcal{L}_{MLE} = -\frac{1}{T} \sum_{t=1}^{T} \log(p(y^*_t|x) \tag{10}$$

After converging, the model is further optimized with local variance loss and global variance loss. The mix of loss functions is:

$$\mathcal{L} = \mathcal{L}_{MLE} + \lambda_1 \mathcal{L}_L + \lambda_2 \mathcal{L}_G \tag{11}$$

where $\lambda_1$ and $\lambda_2$ are hyper-parameters.

## 3 Experiments

### 3.1 Preliminaries

**Dataset and Metrics.** We conduct our model on the large-scale dataset CNN/Daily Mail (Hermann et al., 2015; Nallapati et al., 2016), which is widely used in the task of abstractive document summarization with multi-sentences summaries. We use the scripts provided by See et al. (2017) to obtain the non-anonymized version of the dataset without preprocessing to replace named entities. The dataset contains 287,226 training pairs, 13,368 validation pairs and 11,490 test pairs in total. We use the full-length ROUGE F1[1] and METEOR[2] as our main evaluation metrics.

**Implementation Details.** The data preprocessing is the same as PGN (See et al., 2017), and we randomly initialize the word embeddings. The hidden states of the encoder and the decoder are both 256-dimensional and the embedding size is also 256. Adagrad with learning rate 0.15 and an accumulator with initial value 0.1 are used to train the model. We conduct experiments on a single Tesla P100 GPU with a batch size of 64 and it takes about 50000 iterations for pre-training and 10000 iterations for fine-tuning. Beam search size is set to 4 and trigram avoidance (Paulus et al., 2018) is used to avoid trigram-level repetition. Tuned on validation set, $\lambda_1$ and $\lambda_2$ in the loss function (Equation. 11) is set to 0.3 and 0.1, respectively.

### 3.2 Automatic Evaluation Result

As shown in Table 1 (the performance of other models is collected from their papers), our model exceeds the PGN baseline by 3.85, 2.1 and 3.37 in terms of R-1, R-2 and R-L respectively and receives over 3.23 point boost on METEOR. FastAbs (Chen and Bansal, 2018) regards ROUGE scores as reward signals with reinforcement learning, which brings a great performance gain. DCA (Celikyilmaz et al., 2018) proposes deep communicating agents with reinforcement setting and achieves the best results on CNN/Daily Mail. Although our experimental results have not outperformed the state-of-the-art models, our model has a much simpler structure with fewer parameters. Besides, these simple methods do yield a boost

[1] We use the official package pyrouge https://pypi.org/project/pyrouge/

[2] http://www.cs.cmu.edu/~alavie/METEOR/

| Models | ROUGE-1 | ROUGE-2 | ROUGE-L | METEOR |
|---|---|---|---|---|
| PREVIOUS WORKS | | | | |
| PGN (See et al., 2017) | 36.44 | 15.66 | 33.41 | 16.65 |
| PGN+Coverage (See et al., 2017) | 39.53 | 17.28 | 36.38 | 18.72 |
| Intra-att.+RL (Paulus et al., 2018) | 39.87 | 15.82 | 36.90 | - |
| FastAbs+RL (Chen and Bansal, 2018) | 40.88 | 17.80 | 38.54 | 20.38 |
| DCA+RL (Shi et al., 2018) | 41.69 | 19.47 | 37.92 | - |
| OUR MODELS | | | | |
| PGN (ours) | 36.72 | 15.76 | 33.40 | 17.19 |
| PGN+Coverage (ours) | 39.75 | 17.42 | 36.36 | 19.73 |
| PGN+ARU | 37.41 | 16.01 | 34.05 | 18.03 |
| +Local variance loss | 39.45 | 17.26 | 35.99 | 19.02 |
| +Global variance loss | 40.29 | 17.76 | 36.78 | 19.88 |

Table 1: Performance on CNN/Daily Mail test dataset.

in performance compared with PGN baseline and may be applied on other models with attention mechanism.

We further evaluate how these optimization approaches work. The results at the bottom of Table 1 verify the effectiveness of our proposed methods. The ARU module has achieved a gain of 0.97 ROUGE-1, 0.35 ROUGE-2, and 0.64 ROUGE-L points; the local variance loss boosts the model by 3.01 ROUGE-1, 1.6 ROUGE-2, and 2.58 ROUGE-L. As shown in Figure 2, the global variance loss helps with eliminating n-gram repetitions, which verifies its effectiveness.

### 3.3 Human Evaluation and Case Study

We also conduct human evaluation on the generated summaries. Similar to the previous work (Chen and Bansal, 2018; Nallapati et al., 2017), we randomly select 100 samples from the test set of CNN/Daily Mail dataset and ask 3 human testers to measure *relevance* and *readability* of each summary. Relevance is based on how much salient information does the summary contain, and readability is based on how fluent and grammatical the summary is. Given an article, different people may have different understandings of the main content of the article, the ideal situation is that more than one reference is paired with the articles. However, most of summarization datasets contain the pairs of article with a single reference summary due to the cost of annotating multi-references. Since we use the reference summaries as target sequences to train the model and assume that they are the gold standard, we give both articles and reference summaries to the annotator to score the generated summaries. In other words, we compare the generated summaries against the reference ones and the original article to obtain the (relative) scores in Table 3. Each perspective is assessed with a score from 1 (worst) to 5 (best). The result in Table 2 demonstrate that our model performs better under both criteria w.r.t. See et al. (2017). Additionally, we show the example of summaries generated by our model and baseline model in Table 3. As can be seen from the table, PGN suffers from repetition and fails to obtain the salient information. Though with coverage mechanism solving saliency and repetition problem, it generates many trivial facts. With ARU, the model successfully concentrates on the salient information, however, it also suffers from serious repetition problem. Further optimized by the variance loss, our model can avoid repetition and generate summary with salient information. Besides, our generated summary contains fewer trivial facts compared to the PGN+Coverage model.

| Models | Relevance | Readability |
|---|---|---|
| Reference | 5.00 | 5.00 |
| PGN | 2.27 | 4.30 |
| PGN+Coverage | 2.46 | 4.88 |
| Our model | 2.74 | 4.92 |

Table 2: Human Evaluation: pairwise comparison between our final model and PGN model.

## 4 Related Work

The exploration on document summarization can be broadly divided into extractive and abstractive summarization. The extractive methods (Nallapati et al., 2017; Jadhav and Rajan, 2018; Shi

| |
|---|
| **Article**: poundland has been been forced to pull decorative plastic easter eggs from their shelves over fears children may choke - because they look like cadbury mini eggs . **trading standards officials in buckinghamshire and surrey raised the alarm** over the chinese made decorations , as **they were ‘ likely to contravene food imitation safety rules ’** . the eggs have now been withdrawn nationwide ahead of the easter break . scroll down for video . poundland has been been forced to pull decorative plastic easter eggs from their shelves over fears they may choke - because they look like cadbury mini eggs -lrb- pictured is the poundland version -rrb- . **the eggs bear a striking similarity to the sugar-coated chocolate treats** with a brown ‘ speckle ’ designed to make it look like a quail ’s egg -lrb- cadbury mini eggs are pictured -rrb- . . . . ‘ parents should also be wary of similar products being offered for sale over the easter period at other stores or online . ’ |
| **Reference Summary**: Trading standards officials in buckinghamshire and surrey raised alarm. Officers said they were ‘likely to contravene food imitation safety rules’. The eggs bear a striking similarity to the sugar-coated chocolate treats. |
| **PGN**: Poundland has been forced to pull decorative plastic easter eggs from their shelves over fears children may choke - because they look like cadbury mini eggs. The eggs have now been withdrawn nationwide ahead of the easter break. ***The eggs have now been withdrawn nationwide ahead of the easter break.*** |
| **PGN+Coverage**: **Trading standards officials in buckinghamshire and surrey raised the alarm** over the chinese made decorations , as **they were ‘ likely to contravene food imitation safety rules ’** the eggs have now been withdrawn nationwide ahead of the easter break . **the eggs bear a striking similarity to the sugar-coated chocolate treats** with a brown ‘ speckle ’ designed to make it look like a quail ’s egg . |
| **+ ARU**: **Eggs bear a striking similarity to the sugar-coated chocolate treats** with a brown ‘speckle’ designed to make it look like a quail’s egg. ***The eggs bear a striking similarity to the sugar-coated chocolate treats with a brown ‘speckle’ designed to make it look like a quail’s egg.*** |
| **+ Variance loss**: **Trading standards officials in buckinghamshire and surrey raised the alarm** over the chinese made decorations, as **they were ‘likely to contravene food imitation safety rules’**. The eggs have now been withdrawn nationwide ahead of the easter break. **The eggs bear a striking similarity to the sugar-coated chocolate treats** with a brown ‘speckle’. |

Table 3: The **bold** words in *article* are salient parts contained in *reference summary*. The **blue** words in generated summaries are salient information and the ***red*** words are repetition.

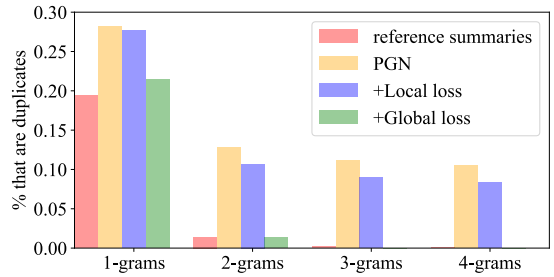


Figure 2: With global variance loss, our model (green bar) can avoid repetitions and achieve comparable percentage of duplicates with reference summaries.

et al., 2018) select salient sentences from original document as a summary. In contrast, abstractive summarization (Rush et al., 2015; Nallapati et al., 2016; See et al., 2017; Chen and Bansal, 2018) generates summaries word-by-word after digesting the main content of the document. Out-of-vocabulary(OOV), repetition, and saliency are three conspicuous problems need to be well solved in abstractive document summarization. Some works (Nallapati et al., 2016; See et al., 2017; Paulus et al., 2018) handle the OOV problem by introducing the pointer network. See et al. (2017) introduces a coverage mechanism, which is a variant of the coverage vector (Tu et al., 2016) from Neural Machine Translation, to eliminate repetitions. However, there are just a few studies on saliency problem (Tan et al., 2017; Shi et al., 2018; Gehrmann et al., 2018). To obtain more salient information, Chen et al. (2016) proposes a new attention mechanism to distract them in the decoding step to better grasp the overall meaning of input documents. We optimize attention using an attention refinement unit under the novel variance loss supervision. As far as we know, we are the first to propose explicit losses to refine the attention model in abstractive document summarization tasks. Recently many models (Paulus et al., 2018; Celikyilmaz et al., 2018; Chen and Bansal, 2018; Zhou et al., 2018; Jiang and Bansal, 2018) have emerged taking advantage of reinforcement learning (RL) to solve the discrepancy issue in seq2seq model and have yielded the state-of-the-art performance.

## 5 Conclusion

In this paper, we propose simple but effective methods to optimize the vanilla attention mechanism in abstarctive document summarization. The results on CNN/Daily Mail dataset demonstrate the effectiveness of our methods. We argue that these simple methods are also adaptable to other summarization models with attention. Further exploration on this and combination with other approaches like RL remains as our future exploration. Besides, we will also conduct experiments on several other current summarization datasets like New York Times (NYT) (Paulus et al., 2018) and Newsroom (Grusky et al., 2018).

## References


Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep communicating agents for abstractive summarization. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1662–1675. Association for Computational Linguistics.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, and Hui Jiang. 2016. Distraction-based neural networks for modeling documents. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 2754–2760. AAAI Press.

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 675–686. Association for Computational Linguistics.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 376–380. Association for Computational Linguistics.

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109. Association for Computational Linguistics.

Max Grusky, Mor Naaman, and Yoav Artzi. 2018. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 708–719.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1693–1701.

T. Huang, G. Yang, and G. Tang. 1979. A fast two-dimensional median filtering algorithm. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 27(1):13–18.

Aishwarya Jadhav and Vaibhav Rajan. 2018. Extractive summarization with swap-net: Sentences and words from alternating pointer networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 142–151. Association for Computational Linguistics.

Yichen Jiang and Mohit Bansal. 2018. Closed-book training to improve summarization encoder memory. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 4067–4077.

Yunjae Jung, Donghyeon Cho, Dahun Kim, Sanghyun Woo, and In So Kweon. 2018. Discriminative feature learning for unsupervised video summarization. *CoRR*, abs/1811.09791.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3075–3081.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290. Association for Computational Linguistics.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083. Association for Computational Linguistics.

Shiv Shankar, Siddhant Garg, and Sunita Sarawagi. 2018. Surprisingly easy hard-attention for sequence to sequence learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 640–645.

Jiaxin Shi, Chen Liang, Lei Hou, Juanzi Li, Zhiyuan Liu, and Hanwang Zhang. 2018. Deepchannel: Salience estimation by contrastive learning for extractive document summarization. *CoRR*, abs/1811.02394.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3104–3112, Cambridge, MA, USA. MIT Press.

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1171–1181. Association for Computational Linguistics.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2692–2700.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 2048–2057.

Qingyu Zhou, Nan Yang, Furu Wei, Shaohan Huang, Ming Zhou, and Tiejun Zhao. 2018. Neural document summarization by jointly learning to score and select sentences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 654–663. Association for Computational Linguistics.

# Rewarding Coreference Resolvers for Being Consistent with World Knowledge

**Rahul Aralikatte**[1], **Heather Lent**[1], **Ana Valeria Gonzalez**[1], **Daniel Hershcovich**[1], **Chen Qiu**[1,2], **Anders Sandholm**[3], **Michael Ringaard**[3], and **Anders Søgaard**[1,3]

[1]University of Copenhagen, [2]China University of Geosciences, [3]Google Research
{rahul,hcl,ana,hershcovich,chen.qiu,soegaard}@di.ku.dk,
{sandholm,ringgaard}@google.com

## Abstract

Unresolved coreference is a bottleneck for relation extraction, and high-quality coreference resolvers may produce an output that makes it a lot easier to extract knowledge triples. We show how to improve coreference resolvers by forwarding their input to a relation extraction system and reward the resolvers for producing triples that are found in knowledge bases. Since relation extraction systems can rely on different forms of supervision and be biased in different ways, we obtain the best performance, improving over the state of the art, using multi-task reinforcement learning.

## 1 Introduction

Coreference annotations are costly and difficult to obtain, since trained annotators with sufficient world knowledge are necessary for reliable annotations. This paper presents a way to *simulate* annotators using reinforcement learning. To motivate our approach, we rely on the following example from Martschat and Strube (2014, colors added to mark entity mentions):

(1) [**Lynyrd Skynyrd**]$_1$ was formed in **Florida**$_2$. Other bands from [**the Sunshine State**]$_2$ include **Fireflight** and **Marilyn Manson**.

Martschat and Strube (2014) cite the association between Florida and the Sunshine State as an example of a common source of name-name recall error for state-of-the-art coreference resolution systems. The challenge is that the two names co-occur relatively infrequently and are unlikely to do so in a moderate-sized, manually annotated training corpus. A state-of-the-art system may be able to infer the relation using distributional information about the phrase the Sunshine State, but is likely to have limited evidence for the decision that it is coreferential with Florida rather than Lynyrd Skynyrd.

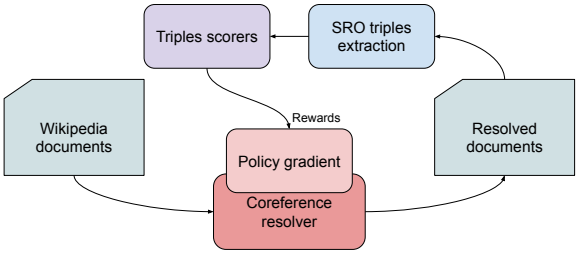


Figure 1: Our strategy for training a coreference resolver using reward from relation extraction.

While coreference-annotated data is scarce, knowledge bases including factual information (such as that Fireflight is from Florida) are increasingly available. For a human annotator unaware that Florida is sometimes referred to as the Sunshine State, the information that Fireflight is from Florida is sufficient to establish that Florida and the Sunshine State are (with high probability) coreferential. This paper explores a novel architecture for making use of such information from knowledge bases by tying a coreference resolution system to a relation extraction system, enabling us to reward the coreference system for making predictions that lead us to infer facts that are consistent with such knowledge bases. This potentially provides us with more evidence for resolving coreference such as (1).

We propose a training strategy (Figure 1) in which we pass on the predictions of a neural coreference resolver to an open relation extraction (OpenRE) system, matching relations extracted from resolved sentences with a knowledge base. We show how checking the produced relationships for consistency against the knowledge base produces a reward that is, indirectly, a signal about the quality of the coreference resolution. In order to generalize this signal beyond the coverage of the knowledge base, we train a Universal Schema model (Riedel et al., 2013) and use its confidence as our reward function. With this reward function,



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1229–1235, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

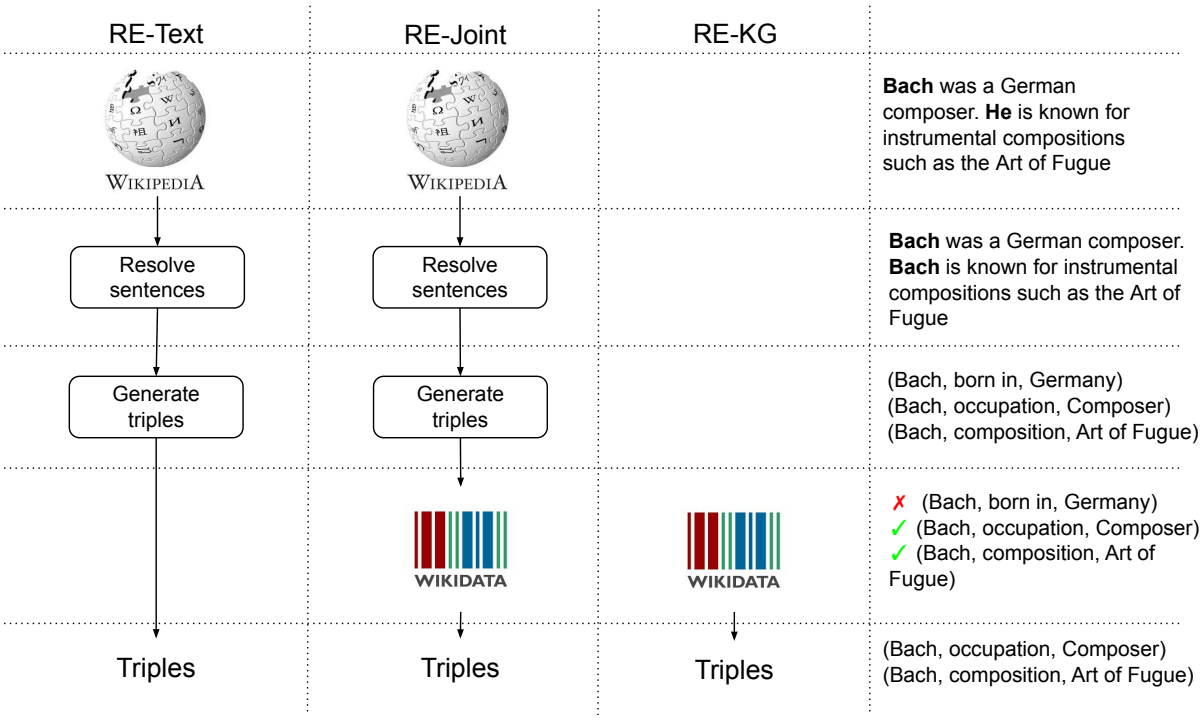


Figure 2: The three left columns show the pipelines for extracting data in different settings to train the reward models. The right column gives an example of the process: original text, coreference resolved text, extracted triples, pruning triples (the first triple is discarded as it is not present in Wikidata) and final training data.

we do policy-gradient fine-tuning of our coreference resolver, effectively optimizing its predictions' consistency with world knowledge.

**Contributions** We demonstrate that training a coreference resolver by reinforcement learning with rewards from a relation extraction system, results in improvements for coreference resolution. Our code is made publicly available at https://github.com/rahular/coref-rl

## 2 Consistency Reward for Coreference Resolution

In order to reward a coreference resolver for being consistent with world knowledge, we propose a simple training strategy based on relation extraction: (i) Sample a Wikipedia[1] document at random, (ii) Replace mentions with their antecedents using a coreference resolver, (iii) Apply an off-the-shelf openRE system to each rewritten document, (iv) Score relationships that include coreferent mentions using Universal Schema, and (v) Use the score as a reward for training the coreference resolvers.

**Reward functions** To model consistency with world knowledge, we train different Universal Schema models (Riedel et al., 2013; Verga and McCallum, 2016), resulting in three reward functions (Figure 2): **RE-KG** (Knowledge Graph Universal Schema) is trained to predict whether two entities are linked in Wikidata[2]; **RE-Text** (Text-based Universal Schema) is trained to predict whether two entities co-occur in Wikipedia; and **RE-Joint** (Joint Universal Schema) is trained to predict whether two entities are linked *and* co-occur. The three rewards focus on different aspects of relationships between entities, giving complimentary views of what entities are related.

Similar to Verga et al. (2016), we parameterize candidate relation phrases with a BiLSTM (Graves and Schmidhuber, 2005), and use pre-trained Wikidata BigGraph embeddings (Lerer et al., 2019) as the entity representations. We apply a one-layer MLP on the concatenated representations to get the reward value.

**Updating the coreference resolver** Each resolved document is converted into $n$ subject-relation-object (SRO) triples by an open information retrieval system (Angeli et al., 2015). Each

[1] https://www.wikipedia.org

[2] https://www.wikidata.org

triple $t_i$ is then scored using a reward function to obtain a reward $r_i$ for $i \in \{1, \ldots, n\}$. The final document-level reward is the normalized sum of the individual rewards as shown in Equation 1, where $R_h$ is a moving window containing the previous $h = 100$ normalized reward values.

$$R = \frac{\sum_i r_i - mean(R_h)}{stddev(R_h)} \quad (1)$$

Since $R$ is not differentiable with respect to the coreference resolver's parameters, we use policy gradient training to update the coreference resolver. We select the best action according to the current policy, using random exploration of the alternative solutions with $p = \frac{1}{10}$.

**Multi-task reinforcement learning** Our overall training procedure is presented in Algorithm 1. After training the three aforementioned reward models, we create **RE-Distill** by interpolating their trained weights. Next, we pre-train a coreference resolver using supervised learning, and fine-tune it using each of the three reward functions to get three different coreference policies: **Coref-KG**, **Coref-Text** and **Coref-Joint**, respectively. We then use multi-task reinforcement learning to combine these three policies to get **Coref-Distill**. Our approach is a particular instance of DisTraL (Teh et al., 2017), using policy gradient and model interpolation. Finally, **Coref-Distill** is fine-tuned with rewards from **RE-Distill**.

**Algorithm 1** Multi-task Reinforcement Learning

**Require:** Baseline initialized policies $\theta_n$ for $n \in \{1, 2, 3\}$
**Require:** Reward functions $\texttt{reward}_n$ for $n \in \{1, 2, 3\}$
**Require:** Distilled reward function $\texttt{reward}_*$
**while** stopping criterion not met **do**
  Sample $k$ documents $D^k$
  **for** $d \in D^k$ **do**
    **for** $n \in \{1, 2, 3\}$ **do**
      $\mathcal{C}_d$ = entity clusters with $\theta_n$
      $d'$ = resolve $d$ with $\mathcal{C}_d$
      $\mathcal{T}$ = obtain OpenIE triples for $d'$
      $r$ = reward$_n(d')$
      $\hat{g}_k$ = policy gradient for $\theta_n$ with reward $r$
      $\theta_n^{k+1} = \theta_n^k + \alpha_k \hat{g}_k$
    **end for**
  **end for**
**end while**
Distilled policy $\theta_* = \frac{\theta_1 + \theta_2 + \theta_3}{3}$
Sample $k$ documents $D^k$
**for** $d \in D^k$ **do**
  $d'$ = resolve $d$ with $\mathcal{C}_d$
  $\mathcal{T}$ = obtain OpenIE triples for $d'$
  $r$ = reward$_*(d')$
  $\hat{g}_k$ = policy gradient for $\theta_*$ with reward $r$
  $\theta_*^{k+1} = \theta_*^k + \alpha_k \hat{g}_k$
**end for**
**return** Distilled policy $\theta_*$

## 3 Experiments

We use a state-of-the-art neural coreference resolution model (Lee et al., 2018) as our baseline coreference resolver.[3] This model extends Lee et al. (2017) with coarse-to-fine inference and ELMo pretrained embeddings (Peters et al., 2018).

**Data** We use the standard training, validation, and test splits from the English OntoNotes.[4] We also evaluate on the English WikiCoref (Ghaddar and Langlais, 2016), with a validation and test split of 10 and 20 documents respectively.

**Reward model training** We use data from English Wikipedia and Wikidata to train our three reward models. For training **RE-KG**, we sample 1 million Wikidata triples, and expand them to 12 million triples by replacing relation phrases with their aliases. For **RE-Text**, we pass the summary paragraphs from 50,000 random Wikipedia pages to Stanford's OpenIE extractor (Manning et al., 2014), creating 2 million triples. For **RE-Joint**, we only use Wikipedia triples that are grounded in Wikidata, resulting in 60,000 triples.[5] We further sample 200,000 triples from Wikidata and Wikipedia for validation, and train the reward models with early stopping based on the $F_1$ score of their predictions.

**Evaluation** All models are evaluated using the standard CoNLL metric, which is the average $F_1$ score over MUC, CEAFe, and $B^3$ (Denis and Baldridge, 2009).

## 4 Results

Since the quality of our reward models is essential to the performance of the coreference resolver adaptations, we first report the validation accuracy and $F_1$ scores of the four reward models used, in Table 1. We clearly see the advantage of distillation, with a 5% absolute difference between the best single model (**RE-Text**) and **RE-Distill**.

Table 2 presents the downstream effects of applying these reward functions to our baseline coreference policy.[6]

[3] https://github.com/kentonl/e2e-coref

[4] https://catalog.ldc.upenn.edu/LDC2013T19

[5] That is, we retain only those triples whose subject and object can be linked to an entity in Wikidata.

[6] The models were re-trained from scratch, and the scores

| System | Data | Accuracy | $F_1$ score |
|---|---|---|---|
| RE-KG | 12M | 0.64 | 0.78 |
| RE-Text | 2M | 0.71 | 0.83 |
| RE-Joint | 60K | 0.58 | 0.73 |
| RE-Distill | — | **0.78** | **0.88** |

Table 1: Training data size, accuracy and $F_1$ scores of the reward models on the 200,000 validation triples.

| System | OntoNotes | WikiCoref |
|---|---|---|
| Lee et al. (2018) | 72.60 | 57.49 |
| Coref-KG | 72.96 | 57.84 |
| Coref-Text | 72.99 | 57.54 |
| Coref-Joint | 72.77 | 57.51 |
| Coref-Distill | **73.10** | **58.14** |

Table 2: Coreference results: average $F_1$ scores on the OntoNotes and WikiCoref test sets. Differences are significant w.r.t. $B^3$ (bootstrap test, $p < 0.05$).

The coreference resolution results are similar to the relation extraction results: using a distilled policy, learned through multi-task reinforcement learning, leads to better results on both datasets.[7]

While improvements over the current state of the art are relatively small, they reflect significant progress, as they demonstrate the ability to successfully augment coreference resolvers with "free" data from large-scale KB like Wikidata. For relation extraction, this could have positive downstream effects, and also ensure that relations are consistent with real world knowledge. Moreover, this approach has the potential to also be beneficial for coreference resolution in low resource languages, where less annotated data is available, as Wikidata triples are abundant for many languages.

## 5 Analysis

Empirically, we find that fine-tuning the coreference resolver on Wikidata results in two kinds of improvements:

**Better mention detection** Since the model is rewarded if the SRO triples produced from the resolved document are present in Wikidata, the model can do well only if it correctly resolves the subject and object, which are usually named entities (more generally, noun phrases). Indeed, we see an improvement in mention detection as exemplified in the first example of Figure 3. Compared to the baseline, the fine-tuned model identifies a larger number of entities, including "southern hemisphere", "Cambridge" and "Oxford", which are missed by the baseline model.

**Better linking** As a direct consequence of the above, the model is inclined to also link noun phrases that are not entities. In the second example of Figure 3, we see that "This attempt" is linked to "releasing" by the fine-tuned model. Interestingly, we do not see this type of *eventive* noun phrase linking either in OntoNotes or in the predictions of the baseline model.

This phenomenon, however, also has a side-effect of producing singleton clusters and spurious linking, which adversely affect the recall. On the OntoNotes test data, while the average precision of the best performing fine-tuned model is higher than the baseline (75.62 vs. 73.80), a drop in recall (70.75 vs. 71.34) causes the final $F_1$ score to only marginally improve.

## 6 Related Work

**Coreference resolution** Among neural coreference resolvers (Wu and Ma, 2017; Meng and Rumshisky, 2018), Lee et al. (2017) were the first to propose an end-to-end resolver which did not rely on hand-crafted rules or a syntactic parser. Extending this work, Lee et al. (2018) introduced a novel attention mechanism for iteratively ranking spans of candidate coreferent mentions, thereby improving the identification of long distance coreference chains. Zhang et al. (2019) improve pronoun coreference resolution by 2.2 F1 using linguistic features (gender, animacy and plurality) and a frequency based predicate-argument selection preference as external knowledge. Emami et al. (2018) incorporate knowledge into coreference resolution by means of information retrieval, finding sentences that are syntactically similar to a given instance, and improving F1 by 0.16.

**Reinforcement learning** RL has been used for many NLP tasks, including coreference resolution (Clark and Manning, 2016) and relation extraction (Zeng et al., 2018). Clark and Manning (2016) use RL to improve coreference resolution by op-

are slightly different from those reported in Lee et al. (2018).

[7] We repeat this experiment three times with different random seeds and observed the same pattern and very robust performance across the board.

| | Baseline system | Fine-tuned system |
|---|---|---|
| **Mention detection** | According to the library's publications, it is the largest academic library in the southern hemisphere. The university has a number of residential college and halls of residence, based on the college system of Cambridge and Oxford universities. | According to the library's publications, it is the largest academic library in the southern hemisphere. The university has a number of residential college and halls of residence, based on the college system of Cambridge and Oxford universities. |
| **Linking** | On March 19, Obama continued his outreach to the Muslim world, releasing a New Year's video message to the people and government of Iran. This attempt was rebuffed by the Iranian leadership. | On March 19, Obama continued his outreach to the Muslim world, releasing a New Year's video message to the people and government of Iran. This attempt was rebuffed by the Iranian leadership. |

Figure 3: Mention detection and linking examples by the baseline system from Lee et al. (2018), and the best performing fine-tuned system (Coref-Distill). Mentions of the same color are linked to form a coreference cluster.

timizing their mention ranking model and directly use the standard evaluation metrics as the rewards. We, on the other hand, perform end-to-end optimization by rewarding the model's consistency with real world knowledge using relation extraction. To our knowledge, we are the first to use consistency with world knowledge as a reward for tasks other than knowledge base construction.[8]

**Knowledge bases** Knowledge bases have been leveraged across multiple tasks across NLP (Bordes et al., 2011; Chang et al., 2014; Lin et al., 2015; Toutanova et al., 2015; Yang and Mitchell, 2017). Specifically for coreference resolution, Prokofyev et al. (2015) implement a resolver that ensures semantic relatedness of resulting coreference clusters by leveraging Semantic Web annotations. Their work incorporates knowledge graph information only in the final stage of the resolver's pipeline, and not during training. In contrast, our work augments information from the knowledge base directly into the training pipeline. Also, they use DBpedia (Auer et al., 2007) as the ontology. Although both Wikidata and DBpedia are designed to support working with Wikipedia articles, DBpedia can be considered as a subset of Wikidata as Wikipedia infoboxes are its main data source. The advantage of Wikidata over DBpedia is its size, and the fact that it is multilingual, which will allow applying our method to other languages in the future.

[8] Mao et al. (2018), for example, use reinforcement learning with consistency-like reward to induce lexical taxonomies.

## 7 Conclusion

We presented an architecture for adapting coreference resolvers by rewarding them for being consistent with world knowledge. Using simple multi-task reinforcement learning and a knowledge extraction pipeline, we achieved improvements over the state of the art across two datasets. We believe this is an important first step in exploring the usefulness of knowledge bases in the context of coreference resolution and other discourse-level phenomena. In this area, manually annotated data is particularly expensive, and we believe leveraging knowledge bases will eventually reduce the need for manual annotation.

## Acknowlegments

We thank the reviewers for their valuable comments. Rahul Aralikatte, Daniel Hershcovich, Heather Lent, and Anders Søgaard are funded by a Google Focused Research Award. Heather Lent is also funded by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 801199. Chen Qiu is funded in part by the National Natural Science Foundation of China under grant No. 61773355 and the China Scholarship Council.

## References

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the*

*7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 344–354, Beijing, China. Association for Computational Linguistics.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *Proceedings of the 6th International The Semantic Web and 2Nd Asian Conference on Asian Semantic Web Conference*, ISWC'07/ASWC'07, pages 722–735, Berlin, Heidelberg. Springer-Verlag.

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. In *AAAI*.

Kai-Wei Chang, Wen tau Yih, Bishan Yang, and Christopher Meek. 2014. Typed tensor decomposition of knowledge bases for relation extraction. In *EMNLP*.

Kevin Clark and Christopher D Manning. 2016. Deep reinforcement learning for mention-ranking coreference models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2256–2262.

Pascal Denis and Jason Baldridge. 2009. Global joint models for coreference resolution and named entity classification. *Procesamiento del Lenguaje Natural*, 42.

Ali Emami, Adam Trischler, Kaheer Suleman, and Jackie Chi Kit Cheung. 2018. A generalized knowledge hunting framework for the Winograd schema challenge. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 25–31, New Orleans, Louisiana, USA. Association for Computational Linguistics.

Abbas Ghaddar and Philippe Langlais. 2016. Wikicoref: An english coreference-annotated corpus of wikipedia articles. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Portorož, Slovenia. European Language Resources Association (ELRA), European Language Resources Association (ELRA).

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5-6):602–610.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197.

Kenton Lee, Luheng He, and Luke Zettlemoyer. 2018. Higher-order coreference resolution with coarse-to-fine inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 687–692.

Adam Lerer, Ledell Wu, Jiajun Shen, Timothee Lacroix, Luca Wehrstedt, Abhijit Bose, and Alex Peysakhovich. 2019. PyTorch-BigGraph: A Large-scale Graph Embedding System. In *Proceedings of the 2nd SysML Conference*, Palo Alto, CA, USA.

Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2015. Modeling relation paths for representation learning of knowledge bases. In *EMNLP*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Yuning Mao, Xiang Ren, Jiaming Shen, Xiaotao Gu, and Jiawei Han. 2018. Building a large-scale annotated chinese corpus. In *ACL*.

Sebastian Martschat and Michael Strube. 2014. Recall error analysis for coreference resolution. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2070–2081, Doha, Qatar. Association for Computational Linguistics.

Yuanliang Meng and Anna Rumshisky. 2018. Triad-based neural network for coreference resolution. In *COLING*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237.

Roman Prokofyev, Alberto Tonon, Michael Luggen, Loic Vouilloz, Djellel Eddine Difallah, and Philippe Cudré-Mauroux. 2015. Sanaphor: Ontology-based coreference resolution. In *International Semantic Web Conference*.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84, Atlanta, Georgia. Association for Computational Linguistics.

Yee Teh, Victor Bapst, Wojciech M. Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. 2017. Distral: Robust multitask reinforcement learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4496–4506. Curran Associates, Inc.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *EMNLP*.

Patrick Verga, David Belanger, Emma Strubell, Benjamin Roth, and Andrew McCallum. 2016. Multilingual relation extraction using compositional universal schema. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 886–896, San Diego, California. Association for Computational Linguistics.

Patrick Verga and Andrew McCallum. 2016. Row-less universal schema. In *AKBC*.

Jheng-Long Wu and Wei-Yun Ma. 2017. A deep learning framework for coreference resolution based on convolutional neural network. *2017 IEEE 11th International Conference on Semantic Computing (ICSC)*, pages 61–64.

Bishan Yang and Tom Michael Mitchell. 2017. Leveraging knowledge bases in lstms for improving machine reading. In *ACL*.

Xiangrong Zeng, Shizhu He, Kang Liu, and Jian Zhao. 2018. Large scaled relation extraction with reinforcement learning. In *AAAI*.

Hongming Zhang, Yan Song, and Yangqiu Song. 2019. Incorporating context and external knowledge for pronoun coreference resolution. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 872–881, Minneapolis, Minnesota. Association for Computational Linguistics.

# An Empirical Study of Incorporating Pseudo Data into Grammatical Error Correction

**Shun Kiyono** [1,2] **Jun Suzuki** [2,1] **Masato Mita** [1,2] **Tomoya Mizumoto** [1,2*] **Kentaro Inui** [2,1]
[1] RIKEN Center for Advanced Intelligence Project [2] Tohoku University
{shun.kiyono, masato.mita, tomoya.mizumoto}@riken.jp;
{jun.suzuki,inui}@ecei.tohoku.ac.jp

## Abstract

The incorporation of pseudo data in the training of grammatical error correction models has been one of the main factors in improving the performance of such models. However, consensus is lacking on experimental configurations, namely, choosing how the pseudo data should be generated or used. In this study, these choices are investigated through extensive experiments, and state-of-the-art performance is achieved on the CoNLL-2014 test set ($F_{0.5} = 65.0$) and the official test set of the BEA-2019 shared task ($F_{0.5} = 70.2$) without making any modifications to the model architecture.

## 1 Introduction

To date, many studies have tackled grammatical error correction (GEC) as a machine translation (MT) task, in which ungrammatical sentences are regarded as the source language and grammatical sentences are regarded as the target language. This approach allows cutting-edge neural MT models to be adopted. For example, the encoder-decoder (EncDec) model (Sutskever et al., 2014; Bahdanau et al., 2015), which was originally proposed for MT, has been applied widely to GEC and has achieved remarkable results in the GEC research field (Ji et al., 2017; Chollampatt and Ng, 2018; Junczys-Dowmunt et al., 2018).

However, a challenge in applying EncDec to GEC is that EncDec requires a large amount of training data (Koehn and Knowles, 2017), but the largest set of publicly available parallel data (Lang-8) in GEC has only two million sentence pairs (Mizumoto et al., 2011). Consequently, the method of augmenting the data by incorporating pseudo training data has been studied intensively (Xie et al., 2018; Ge et al., 2018; Lichtarge et al., 2019; Zhao et al., 2019).

When incorporating pseudo data, several decisions must be made about the experimental configurations, namely, (i) the method of generating the pseudo data, (ii) the seed corpus for the pseudo data, and (iii) the optimization setting (Section 2). However, consensus on these decisions in the GEC research field is yet to be formulated. For example, Xie et al. (2018) found that a variant of the backtranslation (Sennrich et al., 2016b) method (BACKTRANS (NOISY)) outperforms the generation of pseudo data from raw grammatical sentences (DIRECTNOISE). By contrast, the current state of the art model (Zhao et al., 2019) uses the DIRECTNOISE method.

In this study, we investigate these decisions regarding pseudo data, our goal being to provide the research community with an improved understanding of the incorporation of pseudo data. Through extensive experiments, we determine suitable settings for GEC. We justify the reliability of the proposed settings by demonstrating their strong performance on benchmark datasets. Specifically, without any task-specific techniques or architecture, our model outperforms not only all previous single-model results but also all ensemble results except for the ensemble result by Grundkiewicz et al. (2019)[1]. By applying task-specific techniques, we further improve the performance and achieve state-of-the-art performance on the CoNLL-2014 test set and the official test set of the BEA-2019 shared task.

## 2 Problem Formulation and Notation

In this section, we formally define the GEC task discussed in this paper. Let $\mathcal{D}$ be the GEC training data that comprise pairs of an ungrammatical source sentence $\boldsymbol{X}$ and grammatical target sentence

[*] Current affiliation: Future Corporation

[1] The paper (Grundkiewicz et al. 2019) has not been published yet at the time of submission.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1236–1242,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

$\boldsymbol{Y}$, i.e., $\mathcal{D} = \{(\boldsymbol{X}_n, \boldsymbol{Y}_n)\}_n$. Here, $|\mathcal{D}|$ denotes the number of sentence pairs in the dataset $\mathcal{D}$.

Let $\boldsymbol{\Theta}$ represent all trainable parameters of the model. Our objective is to find the optimal parameter set $\widehat{\boldsymbol{\Theta}}$ that minimizes the following objective function $\mathcal{L}(\mathcal{D}, \boldsymbol{\Theta})$ for the given training data $\mathcal{D}$:

$$\mathcal{L}(\mathcal{D}, \boldsymbol{\Theta}) = -\frac{1}{|\mathcal{D}|} \sum_{(\boldsymbol{X},\boldsymbol{Y})\in\mathcal{D}} \log(p(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{\Theta})), \quad (1)$$

where $p(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{\Theta})$ denotes the conditional probability of $\boldsymbol{Y}$ given $\boldsymbol{X}$.

In the standard supervised learning setting, the parallel data $\mathcal{D}$ comprise only "genuine" parallel data $\mathcal{D}_g$ (i.e., $\mathcal{D} = \mathcal{D}_g$). However, in GEC, incorporating pseudo data $\mathcal{D}_p$ that are generated from grammatical sentences $\boldsymbol{Y} \in \mathcal{T}$, where $\mathcal{T}$ represents *seed corpus* (i.e., a set of grammatical sentences), is common (Xie et al., 2018; Zhao et al., 2019; Grundkiewicz et al., 2019).

Our interest lies in the following three nontrivial aspects of Equation 1. **Aspect (i)**: multiple methods for generating pseudo data $\mathcal{D}_p$ are available (Section 3). **Aspect (ii)**: options for the seed corpus $\mathcal{T}$ are numerous. To the best of our knowledge, how the seed corpus domain affects the model performance is yet to be shown. We compare three corpora, namely, Wikipedia, Simple Wikipedia (SimpleWiki) and English Gigaword, as a first trial. Wikipedia and SimpleWiki have similar domains, but different grammatical complexities. Therefore, we can investigate how grammatical complexity affects model performance by comparing these two corpora. We assume that Gigaword contains the smallest amount of noise among the three corpora. We can therefore use Gigaword to investigate whether clean text improves model performance. **Aspect (iii)**: at least two major settings for incorporating $\mathcal{D}_p$ into the optimization of Equation 1 are available. One is to use the two datasets jointly by concatenating them as $\mathcal{D} = \mathcal{D}_g \cup \mathcal{D}_p$, which hereinafter we refer to as JOINT. The other is to use $\mathcal{D}_p$ for pretraining, namely, minimizing $\mathcal{L}(\mathcal{D}_p, \boldsymbol{\Theta})$ to acquire $\boldsymbol{\Theta}'$, and then fine-tuning the model by minimizing $\mathcal{L}(\mathcal{D}_g, \boldsymbol{\Theta}')$; hereinafter, we refer to this setting as PRETRAIN. We investigate these aspects through our extensive experiments (Section 4).

## 3 Methods for Generating Pseudo Data

In this section, we describe three methods for generating pseudo data. In Section 4, we experimentally compare these methods.

**BACKTRANS (NOISY) and BACKTRANS (SAMPLE)** Backtranslation for the EncDec model was proposed originally by Sennrich et al. (2016b). In backtranslation, a reverse model, which generates an ungrammatical sentence from a given grammatical sentence, is trained. The output of the reverse model is paired with the input and then used as pseudo data.

BACKTRANS (NOISY) is a variant of backtranslation that was proposed by Xie et al. (2018)[2]. This method adds $r\beta_{\text{random}}$ to the score of each hypothesis in the beam for every time step. Here, noise $r$ is sampled uniformly from the interval $[0, 1]$, and $\beta_{\text{random}} \in \mathbb{R}_{\geq 0}$ is a hyper-parameter that controls the noise scale. If we set $\beta_{\text{random}} = 0$, then BACKTRANS (NOISY) is identical to standard backtranslation.

BACKTRANS (SAMPLE) is another variant of backtranslation, which was proposed by Edunov et al. (2018) for MT. In BACKTRANS (SAMPLE), sentences are decoded by sampling from the distribution of the reverse model.

**DIRECTNOISE** Whereas BACKTRANS (NOISY) and BACKTRANS (SAMPLE) generate ungrammatical sentences with a reverse model, DIRECTNOISE injects noise "directly" into grammatical sentences (Edunov et al., 2018; Zhao et al., 2019). Specifically, for each token in the given sentence, this method probabilistically chooses one of the following operations: (i) masking with a placeholder token $\langle mask \rangle$, (ii) deletion, (iii) insertion of a random token, and (iv) keeping the original[3]. For each token, the choice is made based on the categorical distribution $(\mu_{\text{mask}}, \mu_{\text{deletion}}, \mu_{\text{insertion}}, \mu_{\text{keep}})$.

## 4 Experiments

The goal of our experiments is to investigate **aspect (i)–(iii)** introduced in Section 2. To ensure that the experimental findings are applicable to GEC in general, we design our experiments by using the following two strategies: (i) we use an off-the-shelf EncDec model without any task-specific architecture or techniques; (ii) we conduct hyper-parameter tuning, evaluation and comparison of each method or setting on the validation set. At the end of experiments (Section 4.5), we summarize our findings and propose suitable settings. We then perform a single-shot evaluation of their performance on the test set.

[2] referred as "random noising" in Xie et al. (2018)

[3] The detailed algorithm is described in Appendix A.

| Dataset | #sent (pairs) | #refs. | Split | Scorer |
|---|---|---|---|---|
| BEA-train | 561,410 | 1 | train | - |
| BEA-valid | 2,377 | 1 | valid | ERRANT |
| CoNLL-2014 | 1,312 | 2 | test | ERRANT & $M^2$ scorer |
| JFLEG | 1,951 | 4 | test | GLEU |
| BEA-test | 4,477 | 5 | test | ERRANT |
| SimpleWiki* | 1,369,460 | - | - | - |
| Wikipedia* | 145,883,941 | - | - | - |
| Gigaword* | 131,864,979 | - | - | - |

Table 1: Summary of datasets used in our experiments. Dataset marked with "*" is a seed corpus $\mathcal{T}$.

| Method | Prec. | Rec. | $F_{0.5}$ |
|---|---|---|---|
| Baseline | 46.6 | 23.1 | 38.8 |
| BACKTRANS (SAMPLE) | 44.6 | 27.4 | 39.6 |
| BACKTRANS (NOISY) | 42.5 | **31.3** | 39.7 |
| DIRECTNOISE | **48.9** | 25.7 | **41.4** |

Table 2: Performance of models on BEA-valid: a value in **bold** indicates the best result within the column. The seed corpus $\mathcal{T}$ is SimpleWiki.

### 4.1 Experimental Configurations

**Dataset** The BEA-2019 workshop official dataset[4] is the origin of the training and validation data of our experiments. Hereinafter, we refer to the training data as BEA-train. We create validation data (BEA-valid) by randomly sampling sentence pairs from the official validation split[5].

As a seed corpus $\mathcal{T}$, we use SimpleWiki[6], Wikipedia[7] or Gigaword[8]. We apply the noizing methods described in Section 3 to each corpus and generate pseudo data $\mathcal{D}_p$. The characteristics of each dataset are summarized in Table 1.

**Evaluation** We report results on BEA-valid, the official test set of the BEA-2019 shared task (BEA-test), the CoNLL-2014 test set (CoNLL-2014) (Ng et al., 2014), and the JFLEG test set (JFLEG) (Napoles et al., 2017). All reported results (except ensemble) are the average of five distinct trials using five different random seeds. We report the scores measured by ERRANT (Bryant et al., 2017; Felice et al., 2016) for BEA-valid, BEA-test, and CoNLL-2014. As the reference sentences of BEA-test are publicly unavailable, we evaluate the model outputs on `CodaLab`[9] for BEA-test. We also report results measured by the $M^2$ scorer (Dahlmeier and Ng, 2012) on CoNLL-2014 to compare them with those of previous studies. We use the GLEU metric (Napoles et al., 2015, 2016) for JFLEG.

**Model** We adopt the *Transformer* EncDec model (Vaswani et al., 2017) using the `fairseq` toolkit (Ott et al., 2019) and use the "Transformer (big)" settings of Vaswani et al. (2017).

**Optimization** For the JOINT setting, we optimize the model with Adam (Kingma and Ba, 2015). For the PRETRAIN setting, we pretrain the model with Adam and then fine-tune it on BEA-train using Adafactor (Shazeer and Stern, 2018)[10].

### 4.2 Aspect (i): Pseudo Data Generation

We compare the effectiveness of the BACKTRANS (NOISY), BACKTRANS (SAMPLE), and DIRECTNOISE methods for generating pseudo data. In DIRECTNOISE, we set $(\mu_{\text{mask}}, \mu_{\text{deletion}}, \mu_{\text{insertion}}, \mu_{\text{keep}}) = (0.5, 0.15, 0.15, 0.2)$[11]. We use $\beta_{\text{random}} = 6$ for BACKTRANS (NOISY)[12]. In addition, we use (i) the JOINT setting and (ii) all of SimpleWiki as the seed corpus $\mathcal{T}$ throughout this section.

The results are summarized in Table 2. BACKTRANS (NOISY) and BACKTRANS (SAMPLE) show competitive values of $F_{0.5}$. Given this result, we exclusively use BACKTRANS (NOISY) and discard BACKTRANS (SAMPLE) for the rest of the experiments. The advantage of BACKTRANS (NOISY) is that its effectiveness in GEC has already been demonstrated by Xie et al. (2018). In addition, in our preliminary experiment, BACKTRANS (NOISY) decoded ungrammatical sentence 1.2 times faster than BACKTRANS (SAMPLE) did. We also use DIRECTNOISE because it achieved the best value of $F_{0.5}$ among all the methods.

### 4.3 Aspect (ii): Seed Corpus $\mathcal{T}$

We investigate the effectiveness of the seed corpus $\mathcal{T}$ for generating pseudo data $\mathcal{D}_p$. The three corpora (Wikipedia, SimpleWiki and Gigaword) are compared in Table 3. We set $|\mathcal{D}_p| = 1.4\text{M}$. The difference in $F_{0.5}$ is small, which implies that the seed corpus $\mathcal{T}$ has only a minor effect on the model performance. Nevertheless, Gigaword consistently outperforms the other two corpora. In particular,

[4] Details of the dataset is in Appendix B.

[5] The detailed data preparation process is in Appendix C.

[6] https://simple.wikipedia.org

[7] We used `2019-02-25` dump file at https://dumps.wikimedia.org/other/cirrussearch/.

[8] We used the English Gigaword Fifth Edition (LDC Catalog No.: LDC2011T07).

[9] https://competitions.codalab.org/competitions/20228

[10] The detailed hyper-parameters are listed in Appendix D.

[11] These values are derived from preliminary experiments (Appendix E).

[12] $\beta_{\text{random}} = 6$ achieved the best $F_{0.5}$ in our preliminary experiments (Appendix F).

| Method | Seed Corpus $\mathcal{T}$ | Prec. | Rec. | $F_{0.5}$ |
|---|---|---|---|---|
| Baseline | N/A | 46.6 | 23.1 | 38.8 |
| BACKTRANS (NOISY) | Wikipedia | 43.8 | 30.8 | 40.4 |
| BACKTRANS (NOISY) | SimpleWiki | 42.5 | 31.3 | 39.7 |
| BACKTRANS (NOISY) | Gigaword | 43.1 | 33.1 | 40.6 |
| DIRECTNOISE | Wikipedia | 48.3 | 25.5 | 41.0 |
| DIRECTNOISE | SimpleWiki | 48.9 | 25.7 | 41.4 |
| DIRECTNOISE | Gigaword | 48.3 | 26.9 | 41.7 |

Table 3: Performance on BEA-valid when changing the seed corpus $\mathcal{T}$ used for generating pseudo data ($|\mathcal{D}_p| = 1.4$M).

DIRECTNOISE with Gigaword achieves the best value of $F_{0.5}$ among all the configurations.

### 4.4 Aspect (iii): Optimization Setting

We compare the JOINT and PRETRAIN optimization settings. We are interested in how each setting performs when the scale of the pseudo data $\mathcal{D}_p$ compared with that of the genuine parallel data $\mathcal{D}_g$ is (i) approximately the same ($|\mathcal{D}_p| = 1.4$M) and (ii) substantially bigger ($|\mathcal{D}_p| = 14$M). Here, we use Wikipedia as the seed corpus $\mathcal{T}$ instead of SimpleWiki or Gigaword for two reasons. First, SimpleWiki is too small for the experiment **(b)** (see Table 1). Second, the fact that Gigaword is not freely available makes it difficult for other researchers to replicate our results.

**(a) Joint Training or Pretraining** Table 4 presents the results. The most notable result here is that PRETRAIN demonstrates the properties of *more pseudo data and better performance*, whereas JOINT does not. For example, in BACKTRANS (NOISY), increasing $|\mathcal{D}_p|$ (1.4M $\rightarrow$ 14M) improves $F_{0.5}$ on PRETRAIN (41.1 $\rightarrow$ 44.5). By contrast, $F_{0.5}$ does not improve on JOINT (40.4 $\rightarrow$ 40.3). An intuitive explanation for this case is that when pseudo data $\mathcal{D}_p$ are substantially more than genuine data $\mathcal{D}_g$, the teaching signal from $\mathcal{D}_p$ becomes dominant in JOINT. PRETRAIN alleviates this problem because the model is trained with only $\mathcal{D}_g$ during fine-tuning. We therefore suppose that PRETRAIN is crucial for utilizing extensive pseudo data.

**(b) Amount of Pseudo Data** We investigate how increasing the amount of pseudo data affects the PRETRAIN setting. We pretrain the model with different amounts of pseudo data {1.4M, 7M, 14M, 30M, 70M}. The results in Figure 1 show that BACKTRANS (NOISY) has superior sample efficiency to DIRECTNOISE. The best model (pretrained with 70M BACKTRANS (NOISY)) achieves $F_{0.5} = 45.9$.

| Optimization | Method | $|\mathcal{D}_p|$ | Prec. | Rec. | $F_{0.5}$ |
|---|---|---|---|---|---|
| N/A | Baseline | 0 | 46.6 | 23.1 | 38.8 |
| PRETRAIN | BACKTRANS (NOISY) | 1.4M | 49.6 | 24.3 | 41.1 |
| PRETRAIN | DIRECTNOISE | 1.4M | 48.4 | 21.2 | 38.5 |
| JOINT | BACKTRANS (NOISY) | 1.4M | 43.8 | 30.8 | 40.4 |
| JOINT | DIRECTNOISE | 1.4M | 48.3 | 25.5 | 41.0 |
| PRETRAIN | BACKTRANS (NOISY) | 14M | 50.6 | 30.1 | 44.5 |
| PRETRAIN | DIRECTNOISE | 14M | 49.8 | 25.8 | 42.0 |
| JOINT | BACKTRANS (NOISY) | 14M | 43.0 | 32.3 | 40.3 |
| JOINT | DIRECTNOISE | 14M | 48.7 | 23.5 | 40.1 |

Table 4: Performance of the model with different optimization settings on BEA-valid. The seed corpus $\mathcal{T}$ is Wikipedia.

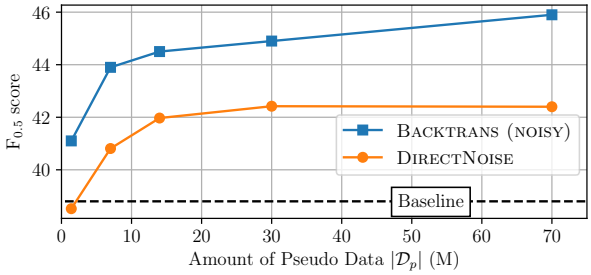


Figure 1: Performance on BEA-valid for different amounts of pseudo data ($|\mathcal{D}_p|$). The seed corpus $\mathcal{T}$ is Wikipedia.

### 4.5 Comparison with Current Top Models

The present experimental results show that the following configurations are effective for improving the model performance: (i) the combination of JOINT and Gigaword (Section 4.3), (ii) the amount of pseudo data $\mathcal{D}_p$ not being too large in JOINT (Section 4.4(a)), and (iii) PRETRAIN with BACKTRANS (NOISY) using large pseudo data $\mathcal{D}_p$ (Section 4.4(b)). We summarize these findings and attempt to combine PRETRAIN and JOINT. Specifically, we pretrain the model using 70M pseudo data of BACKTRANS (NOISY). We then fine-tune the model by combining BEA-train and relatively small DIRECTNOISE pseudo data generated from Gigaword (we set $|\mathcal{D}_p| = 250$K). However, the performance does not improve on BEA-valid. Therefore, the best approach available is simply to pretrain the model with large (70M) BACKTRANS (NOISY) pseudo data and then fine-tune using BEA-train, which hereinafter we refer to as PRETLARGE. We use Gigaword for the seed corpus $\mathcal{T}$ because it has the best performance in Table 3.

We evaluate the performance of PRETLARGE on test sets and compare the scores with the current top models. Table 5 shows a remarkable result, that is,

| Model | Ensemble | CoNLL-2014 ($M^2$ scorer) | | | CoNLL-2014 (ERRANT) | | | JFLEG | BEA-test (ERRANT) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Prec. | Rec. | $F_{0.5}$ | Prec. | Rec. | $F_{0.5}$ | GLEU | Prec. | Rec. | $F_{0.5}$ |
| Chollampatt and Ng (2018) | | 60.9 | 23.7 | 46.4 | - | - | - | 51.3 | - | - | - |
| Junczys-Dowmunt et al. (2018) | | - | - | 53.0 | - | - | - | 57.9 | - | - | - |
| Grundkiewicz and Junczys-Dowmunt (2018) | | 66.8 | 34.5 | 56.3 | - | - | - | 61.5 | - | - | - |
| Lichtarge et al. (2019) | | 65.5 | 37.1 | 56.8 | - | - | - | 61.6 | - | - | - |
| Chollampatt and Ng (2018) | ✓ | 65.5 | 33.1 | 54.8 | - | - | - | 57.5 | - | - | - |
| Junczys-Dowmunt et al. (2018) | ✓ | 61.9 | 40.2 | 55.8 | - | - | - | 59.9 | - | - | - |
| Lichtarge et al. (2019) | ✓ | 66.7 | 43.9 | 60.4 | - | - | - | **63.3** | - | - | - |
| Zhao et al. (2019) | ✓ | 71.6 | 38.7 | 61.2 | - | - | - | 61.0 | - | - | - |
| Grundkiewicz et al. (2019) | ✓ | - | - | 64.2 | - | - | - | 61.2 | 72.3 | 60.1 | 69.5 |
| PRETLARGE | | 67.9 | 44.1 | 61.3 | 61.2 | 42.0 | 56.0 | 59.7 | 65.5 | 59.4 | 64.2 |
| PRETLARGE+SSE+R2L | ✓ | 72.4 | **46.1** | **65.0** | 67.3 | **44.0** | **60.9** | 61.4 | 72.1 | **61.8** | 69.8 |
| PRETLARGE+SSE+R2L+SED | ✓ | **73.3** | 44.2 | 64.7 | **68.1** | 42.1 | 60.6 | 61.2 | **74.7** | 56.7 | **70.2** |

Table 5: Comparison of our best model and current top models: a **bold** value indicates the best result within the column.

our PRETLARGE achieves $F_{0.5} = 61.3$ on CoNLL-2014. This result outperforms not only all previous single-model results but also all ensemble results except for that by Grundkiewicz et al. (2019).

To further improve the performance, we incorporate the following techniques that are widely used in shared tasks such as BEA-2019 and WMT[13]:

**Synthetic Spelling Error** (SSE) Lichtarge et al. (2019) proposed the method of probabilistically injecting character-level noise into the source sentence of pseudo data $\mathcal{D}_p$. Specifically, one of the following operations is applied randomly at a rate of 0.003 per character: deletion, insertion, replacement, or transposition of adjacent characters.

**Right-to-left Re-ranking** (R2L) Following Sennrich et al. (2016a, 2017); Grundkiewicz et al. (2019), we train four right-to-left models. The ensemble of four left-to-right models generate $n$-best candidates and their corresponding scores (i.e., conditional probabilities). We then pass each candidate to the ensemble of the four right-to-left models and compute the score. Finally, we re-rank the $n$-best candidates based on the sum of the two scores.

**Sentence-level Error Detection** (SED) SED classifies whether a given sentence contains a grammatical error. Asano et al. (2019) proposed incorporating SED into the evaluation pipeline and reported improved precision. Here, the GEC model is applied only if SED detects a grammatical error in the given source sentence. The motivation is that SED could potentially reduce the number of false-positive errors of the GEC model. We use the re-implementation of the BERT-based SED model (Asano et al., 2019).

Table 5 presents the results of applying SSE, R2L, and SED. It is noteworthy that PRETLARGE+SSE+R2L achieves state-of-the-art performance on both CoNLL-2014 ($F_{0.5} = 65.0$) and BEA-test ($F_{0.5} = 69.8$), which are better than those of the best system of the BEA-2019 shared task (Grundkiewicz et al., 2019). In addition, PRETLARGE+SSE+R2L+SED can further improve the performance on BEA-test ($F_{0.5} = 70.2$). However, unfortunately, incorporating SED decreased the performance on CoNLL-2014 and JFLEG. This fact implies that SED is sensitive to the domain of the test set since the SED model is fine-tuned with the official validation split of BEA dataset. We leave this sensitivity issue as our future work.

## 5 Conclusion

In this study, we investigated several aspects of incorporating pseudo data for GEC. Through extensive experiments, we found the following to be effective: (i) utilizing Gigaword as the seed corpus, and (ii) pretraining the model with BACKTRANS (NOISY) data. Based on these findings, we proposed suitable settings for GEC. We demonstrated the effectiveness of our proposal by achieving state-of-the-art performance on the CoNLL-2014 test set and the BEA-2019 test set.

## Acknowledgements

We thank the three anonymous reviewers for their insightful comments. We are deeply grateful to Takumi Ito and Tatsuki Kuribayashi for kindly sharing the re-implementation of BACKTRANS (NOISY). The work of Jun Suzuki was supported in part by JSPS KAKENHI Grant Number JP19104418 and AIRPF Grant Number 30AI036-8.

[13] http://www.statmt.org/wmt19/

## References


Hiroki Asano, Masato Mita, Tomoya Mizumoto, and Jun Suzuki. 2019. The AIP-Tohoku System at the BEA-2019 Shared Task. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2019)*, pages 176–182.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*.

Christopher Bryant, Mariano Felice, and Ted Briscoe. 2017. Automatic Annotation and Evaluation of Error Types for Grammatical Error Correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 793–805.

Shamil Chollampatt and Hwee Tou Ng. 2018. A Multilayer Convolutional Encoder-Decoder Neural Network for Grammatical Error Correction. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI 2018)*, pages 5755–5762.

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better Evaluation for Grammatical Error Correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2012)*, pages 568–572.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. 2013. Building a Large Annotated Corpus of Learner English: The NUS Corpus of Learner English. In *Proceedings of the 8th Workshop on Building Educational Applications Using NLP (BEA 2013)*, pages 22–31.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding Back-Translation at Scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*, pages 489–500.

Mariano Felice, Christopher Bryant, and Ted Briscoe. 2016. Automatic Extraction of Learner Errors in ESL Sentences Using Linguistically Enhanced Alignments. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 825–835.

Tao Ge, Furu Wei, and Ming Zhou. 2018. Fluency Boost Learning and Inference for Neural Grammatical Error Correction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pages 1055–1065.

Sylviane Granger. 1998. The computer learner corpus: A versatile new source of data for SLA research. In Sylviane Granger, editor, *Learner English on Computer*, pages 3–18. Addison Wesley Longman, London and New York.

Roman Grundkiewicz and Marcin Junczys-Dowmunt. 2018. Near Human-Level Performance in Grammatical Error Correction with Hybrid Machine Translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2018)*, pages 284–290.

Roman Grundkiewicz, Marcin Junczys-Dowmunt, and Kenneth Heafield. 2019. Neural grammatical error correction systems with unsupervised pre-training on synthetic data. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2019)*, pages 252–263.

Jianshu Ji, Qinlong Wang, Kristina Toutanova, Yongen Gong, Steven Truong, and Jianfeng Gao. 2017. A Nested Attention Neural Hybrid Model for Grammatical Error Correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 753–762.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Shubha Guha, and Kenneth Heafield. 2018. Approaching Neural Grammatical Error Correction as a Low-Resource Machine Translation Task. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2018)*, pages 595–606.

Diederik Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*.

Philipp Koehn and Rebecca Knowles. 2017. Six Challenges for Neural Machine Translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39.

Jared Lichtarge, Chris Alberti, Shankar Kumar, Noam Shazeer, Niki Parmar, and Simon Tong. 2019. Corpora Generation for Grammatical Error Correction. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2019)*.

Tomoya Mizumoto, Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. 2011. Mining Revision Log of Language Learning SNS for Automated Japanese Error Correction of Second Language Learners. In *Proceedings of the 5th International Joint Conference on Natural Language Processing (IJCNLP 2011)*, pages 147–155.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2015. Ground Truth for Grammatical Error Correction Metrics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL & IJCNLP 2015)*, pages 588–593.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2016. GLEU Without Tuning. *arXiv preprint arXiv:1605.02592*.

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2017. JFLEG: A Fluency Corpus and Benchmark for Grammatical Error Correction. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017)*, pages 229–234.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 Shared Task on Grammatical Error Correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A Fast, Extensible Toolkit for Sequence Modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2019)*.

Rico Sennrich, Alexandra Birch, Anna Currey, Ulrich Germann, Barry Haddow, Kenneth Heafield, Antonio Valerio Miceli Barone, and Philip Williams. 2017. The university of Edinburgh's neural MT systems for WMT17. In *Proceedings of the Second Conference on Machine Translation (WMT 2017)*, pages 389–399.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Edinburgh neural machine translation systems for WMT 16. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers (WMT 2016)*, pages 371–376.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Improving Neural Machine Translation Models with Monolingual Data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 86–96.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016c. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 1715–1725.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive Learning Rates with Sublinear Memory Cost. In *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*, pages 4603–4611.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 28 (NIPS 2014)*, pages 3104–3112.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016)*, pages 2818–2826.

Toshikazu Tajiri, Mamoru Komachi, and Yuji Matsumoto. 2012. Tense and Aspect Error Correction for ESL Learners Using Global Context. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL 2012)*, pages 198–202.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Advances in Neural Information Processing Systems 31 (NIPS 2017)*, pages 5998–6008.

Ziang Xie, Guillaume Genthial, Stanley Xie, Andrew Ng, and Dan Jurafsky. 2018. Noising and Denoising Natural Language: Diverse Backtranslation for Grammar Correction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2018)*, pages 619–628.

Helen Yannakoudakis, Øistein E. Andersen, Ardeshir Geranpayeh, Ted Briscoe, and Diane Nicholls. 2018. Developing an Automated Writing Placement system for ESL Learners. *Applied Measurement in Education*, 31(3):251–267.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A New Dataset and Method for Automatically Grading ESOL Texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011)*, pages 180–189.

Wei Zhao, Liang Wang, Kewei Shen, Ruoyu Jia, and Jingming Liu. 2019. Improving Grammatical Error Correction via Pre-Training a Copy-Augmented Architecture with Unlabeled Data. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2019)*.

# A Multilingual Topic Model for Learning Weighted Topic Links Across Corpora with Low Comparability

**Weiwei Yang**[*]
Computer Science
University of Maryland
wwyang@cs.umd.edu

**Jordan Boyd-Graber**[†]
Computer Science, iSchool, Language Science, UMIACS
University of Maryland
jbg@umiacs.umd.edu

**Philip Resnik**
Linguistics and UMIACS
University of Maryland
resnik@umd.edu

## Abstract

Multilingual topic models (MTMs) learn topics on documents in multiple languages. Past models align topics across languages by implicitly assuming the documents in different languages are highly comparable, often a false assumption. We introduce a new model that does not rely on this assumption, particularly useful in important low-resource language scenarios. Our MTM learns weighted topic links and connects cross-lingual topics only when the dominant words defining them are similar, outperforming LDA and previous MTMs in classification tasks using documents' topic posteriors as features. It also learns coherent topics on documents with low comparability.

## 1 Introduction

Topic models explain document collections at a high level (Boyd-Graber et al., 2017). Multilingual topic models (MTMs) uncover latent topics *across* languages and reveal commonalities and differences across languages and cultures (Ni et al., 2009; Shi et al., 2016; Gutiérrez et al., 2016). Existing models extend latent Dirichlet allocation (Blei et al., 2003, LDA) and learn *aligned* topics across languages (Mimno et al., 2009).

Prior models work well because they implicitly assume—even if not part of the model—parallel or highly comparable data with well-aligned topics. However, this assumption does not always comport with reality. Even documents from the same place and time can discuss very different things across languages: in multicultural London, Hindi tweets focus on a Bollywood actor's BBC appearance, French blogs fret about Brexit, and English articles focus on Tottenham's lineup. Generally, corpora have a range of "nonparallelness" (Fung, 2000). In less comparable settings, while some topics are shared, languages' emphasis may diverge and some topics may lack analogs.

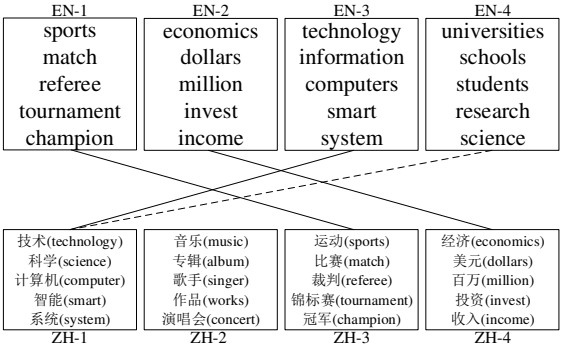


Figure 1: Topic pairs with many word translation pairs have high link weights, e.g., (EN-1, ZH-3) and (EN-2, ZH-4); topic pairs with partial overlap receive lower weights, e.g., (EN-4, ZH-1); a topic is unlinked if there is no corresponding topic in the other language (ZH-2).

We therefore introduce a new multilingual topic model that assumes each language has its own topic sets and jointly learns all topics, but does not force one-to-one alignment across languages. Instead, our MTM learns *weighted* topic links across languages and only assigns a high link weight to a topic pair whose top words have many direct translation pairs (Figure 1). Moreover, it allows unlinked topics if there is no matching topic in the other language. This makes the model robust for (more common) less-comparable data with topic misalignment. Joint inference also allows insights from high-resource languages to uncover low-resource language patterns. It is particularly useful in scenarios that involve modeling topics on low-resource languages in humanitarian assistance, peacekeeping, and/or infectious disease response, while limiting the additional cost to other steps that will also need to be taken, such as finding or creating a word translation dictionary.

We validate the MTM in two classification tasks using inferred topic posteriors as features. Our

[*] Now at Facebook
[†] Now at Google AI Zürich



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1243–1248, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

MTM has higher F1 than other models in both intra- and cross-lingual evaluations, while discovering coherent topics and meaningful topic links.

## 2 Multilingual Topic Model for Connecting Cross-Lingual Topics

Yang et al. (2015) present a flexible framework for adding regularization to topic models. We extend this model to the multilingual setting by adding a potential function that links topics across languages. For simplicity of exposition, we focus on the bilingual case with languages $S$ and $T$.

Unlike Yang et al. (2015) that encode monolingual information only, our potential function encodes multilingual knowledge parameterized by two matrices, $\boldsymbol{\rho_{S\to T}}$ and $\boldsymbol{\rho_{T\to S}}$, that transform topics between the two languages. Cells' values are between 0 and 1 and a cell $\rho_{S\to T,k_T,k_S}$ close to one is a strong connection of topics $k_T$ and $k_S$ in language $T$ and $S$. Transformations $\boldsymbol{\rho}$ are learned from translation pairs' topic distributions.

These topic distributions come from the assignments of Gibbs sampling (Griffiths and Steyvers, 2004). Fortunately adding the potential function is equivalent to adding an additional term to Gibbs sampling for topic models (Yang et al., 2015). During sampling, each token is assigned to a topic, so we can compute a *post hoc* word distribution over topics. The probability of a topic $k$ given a word $w$ is $\Pr(k \mid w) \equiv \Omega_{w,k} \equiv N_{k,w}/N_w$, where $N_{k,w}$ is the number of times that word $w$ is assigned to topic $k$ and $N_w$ is $w$'s term frequency.

To find good topic links $\boldsymbol{\rho_{S\to T}}$, we use a dictionary. For instance, given the translation pair of "sports" and "运 动 (yùn dòng)", they should have similar topic distributions, so we want $\boldsymbol{\rho}_{\text{EN}\to\text{ZH}}\boldsymbol{\Omega}_{\text{sports}}$ to be close to $\boldsymbol{\Omega}_{\text{运动}}$ and vice versa. Moreover, the transformations should be symmetric: $\boldsymbol{\rho_{S\to T}\Omega_{w_S}}$ close to $\boldsymbol{\Omega_{w_T}}$, and vice versa. We encode this cross-lingual knowledge of topic transformations into the potential function $\Psi$ which measures the difference of translation pairs' topic distributions after transformation:

$$\left(\prod_{c=1}^{C} \|\boldsymbol{\Omega_{S,c}} - \boldsymbol{\rho_{T\to S}\Omega_{T,c}}\|_2^{\eta_c} \, \|\boldsymbol{\rho_{S\to T}\Omega_{S,c}} - \boldsymbol{\Omega_{T,c}}\|_2^{\eta_c}\right)^{-1}, \tag{1}$$

where $\eta_c$ is the statistical importance of the $c$-th translation pair to the corpus (Figure 2, full details in the Supplement).

While Yang et al. (2015) provide a blueprint for Gibbs sampling with potential functions without additional parameters, our model has additional parameters of $\boldsymbol{\rho_{S\to T}}$ and $\boldsymbol{\rho_{T\to S}}$ so we need to optimize them. Thus, we use stochastic EM (Celeux, 1985). The E-step updates tokens' topic assignments using Gibbs sampling, while holding the parameters of the topic link weight matrices $\boldsymbol{\rho}$ fixed. The M-step optimizes $\boldsymbol{\rho}$ while holding the topic assignments fixed. We optimize $\Psi$ in log space using the objective function $J(\boldsymbol{\rho_{S\to T}})$ as

$$\sum_{c=1}^{C} \eta_c \log \|\boldsymbol{\Omega_{T,c}} - \boldsymbol{\rho_{S\to T,i_T}\Omega_{S,c}}\|_2 , \tag{2}$$

which is minimized by using L-BFGS (Liu and Nocedal, 1989), with the partial derivatives with respect to $\rho_{S\to T,k_T,k_S}$

$$-\sum_{c=1}^{C} \frac{\eta_c \Omega_{S,c,k_S} \left(\Omega_{T,c,k_T} - \boldsymbol{\rho_{S\to T,k_T}\Omega_{S,c}}\right)}{\|\boldsymbol{\Omega_{T,c}} - \boldsymbol{\rho_{S\to T,i_T}\Omega_{S,c}}\|_2^2}. \tag{3}$$

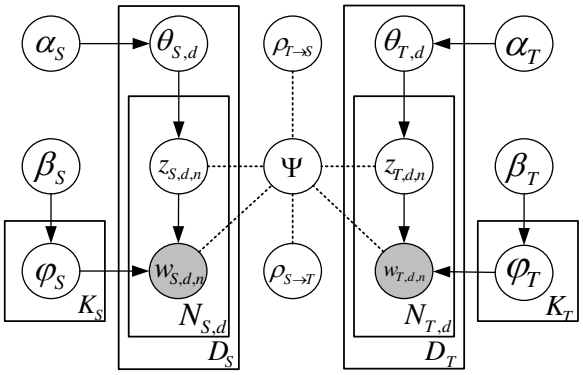


Figure 2: The graphical model of our multilingual topic model. The topic links $\boldsymbol{\rho}$, as instantiated by the function $\Psi$, encourage topics to encourage word translations to have consistent topics.

## 3 Experiments

We evaluate our model extrinsically on classification tasks, followed by intrinsic topic coherence.

### 3.1 Classification with Topic Posteriors

We use two datasets for classification: Wikipedia documents in English (EN) and Chinese (ZH) (Yuan et al., 2018) and an English-Sinhalese (SI) disaster response dataset (Strassel and Tracey, 2016).[1] Each dataset provides labeled documents and a dictionary. Yuan et al. (2018) extract the EN-ZH dictionary from MDBG, while Strassel and Tracey (2016) construct the EN-SI dictionary from online resources and manual annotation.[2] Each

[1] More dataset details in the Supplement.

[2] MDBG: https://www.mdbg.net/chinese/dictionary?page=cc-cedict

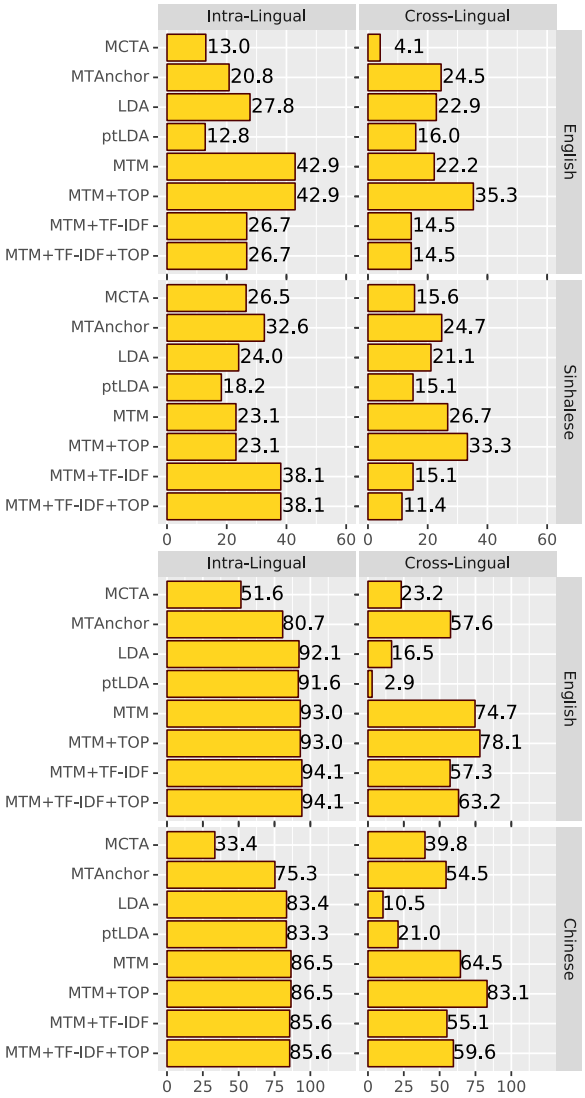


Figure 3: The F1 scores on disaster response (upper) and Wikipedia (lower) datasets. Our MTM outperforms all the baselines in intra- and cross-lingual evaluations.

Wikipedia document is labeled with one of the topics of *film*, *music*, *animals*, *politics*, *religion*, and *food*. A portion of the disaster response documents are labeled with one of eight types of needed rescue resources: *evacuation*, *food supply*, *search/rescue*, *utilities*, *infrastructure*, *medical assistance*, *shelter*, and *water supply*.

We follow Yuan et al. (2018) for preprocessing (such as lemmatization for English and segmentation for Chinese) and use a linear SVM for classification. For the Wikipedia dataset, we report micro-F1 scores on a six-way classification. For the disaster response dataset, our goal is binary classification of the need for *evacuation* versus other assistance. The classification uses features of topic posteriors: $\Pr(k \mid d) \equiv N_{d,k}/N_d$ which is the proportion of the tokens assigned to topic $k$ in document $d$.

The baselines include polylingual tree LDA (Hu et al., 2014, ptLDA) which encodes the dictionary as a tree prior (Andrzejewski et al., 2009), Multilingual Topic Anchoring (Yuan et al., 2018, MTAnchor), and Multilingual Cultural-common Topic Analysis (Shi et al., 2016, MCTA). We also include LDA, which runs monolingually in each language. We use 20 topics and set hyperparameters $\alpha = 0.1$ and $\beta = 0.01$ (if applicable).

Our evaluations are both intra- and cross-lingual. The intra-lingual evaluation trains and tests classifiers on the same language, while the cross-lingual evaluation trains classifiers on one language and tests on another. In cross-lingual evaluations, MTAnchor, MCTA, and ptLDA align topic spaces, so topic posterior transformation is not necessary. LDA cannot transform topic spaces, so we do not apply any transformation. For our MTM, we explore two transformation methods with $\boldsymbol{\rho}$. The first multiplies $\boldsymbol{\rho}$ with a language's document topic distributions, i.e., $\boldsymbol{\rho}_{\text{ZH}\to\text{EN}}\boldsymbol{\theta}_{\text{ZH}}$ and vice versa. The second (TOP), transfers each document topic's probability mass to the topic in the other language with the highest link weight.[3]

Our MTM has higher F1 both intra- and cross-lingually (Figure 3). TF-IDF weighting on translation pairs sometimes improves the intra-lingual F1, although it hurts the cross-lingual F1. Connecting the top linked topics (TOP) is better than directly using the topic link weight matrices. This indicates that $\boldsymbol{\rho}$'s values have some noise.

### 3.2 Looking at Learned Topics

Past MTMs align topics across languages but our MTM does not, so we compare the topics across models to see how they differ. We look at the Movies topics from the Wikipedia dataset (Table 1). For the Chinese MTM topics, we show the three English topics with the highest link weights.

The topics are about Movies, but the MCTA and MTAnchor topics do not rank "movie" or "电影 (diàn yǐng)" at the top. The ptLDA topics, although aligned well, incorrectly align some Chinese words. "胶片 (jiāo piàn)" means "*photographic* film", while "释放 (shì fàng)" means *release* as in "let something go", not movie distribution. ptLDA links words based on translations

[3] An example of TOP is available in the Supplement.

[4] In Tables 1 and 2, "[Q]" denotes the Chinese word is a counter for the following English word.

| Lang. | Words |
|---|---|
| | MCTA |
| ZH | 主演 (starring), 改编 (adapt), 本 (this), 小说 (novel), 拍摄 (shoot), 角色 (role) |
| EN | dog, san, movie, mexican, fighter, novel |
| | MTAnchor |
| ZH | 主演 (starring), 改编 (adapt), 饰演 (act), 本片 (the movie), 演员 (actor), 编剧 (playwright) |
| EN | kong, hong, movie, official, martial, box |
| | LDA |
| ZH | 电影 (movie), 部 ([Q] movie),[4] 美国 (USA), 上映 (release), 英语 (English), 剧情 (plot) |
| EN | film, star, direct, release, action, plot |
| | ptLDA |
| ZH | 电影 (movie), 胶片 (film), 星 (star), 动作 (action), 释放 (release), 影片 (movie) |
| EN | film, star, direct, action, release, plot |
| | MTM |
| ZH | 电影 (movie), 部 ([Q] movie), 上映 (release), 动画 (animation), 故事 (story), 作品 (works), |
| EN (.20) | film, direct, star, release, action, plot |
| EN (.12) | kill, find, death, attack, escape, return |
| EN (.11) | shrine, japanese, temple, japan, shinto, kami |
| | MTM + TF-IDF |
| ZH | 电影 (movie), 部 ([Q] movie), 上映 (release), 美国 (USA), 英语 (English), 导演 (director) |
| EN (.32) | film, direct, star, action, release, plot |
| EN (.24) | film, kill, find, escape, attack, return |
| EN (.09) | character, series, star, game, trek, create |

Table 1: The Movies topics given by models. For the Chinese (ZH) topics given by MTM, the top three English (EN) topics and their link weights are also given.

without looking at the context, which causes problems with multiple-sense words. The LDA and MTM topics are generally coherent.

The MTM's unique joint modeling of weighted topic links also recovers additional topical structure: after linking respective EN-ZH Movies topics, the next linked topics are Action Movies ("kill", "death", "attack", and "escape"). Further, the models capture a degree of connection between Movies and Computer Games (MTM + TF-IDF) and Japanese Animations (MTM).

### 3.3 Looking at Learned Topic Links

We give more examples of weighted MTM topic links in Table 2. High-weighted Biology (ZH-0, EN-12, and EN-19) and Music topics (EN-10, ZH-9, and ZH-17) are characterized by cross-lingual words in common. The model can also infer topic links beyond words, linking topics when the topical words have few direct translations but are related in senses. For instance, ZH-14 is about the "campaigns" against "government". Only "government" overlaps with EN-16 and EN-11, but

| Lang. | Words |
|---|---|
| ZH-0 | 学名 (scientific name), 它们 (they), 呈 (show), 白色 (white), 长 (long), 黑色 (black) |
| EN-12 (.57) | specie, bird, eagle, genus, white, owl |
| EN-19 (.13) | breed, chicken, white, goose, bird, black |
| ZH-14 | 主义 (-ism), 组织 (organization), 美国 (USA), 革命 (revolution), 运动 (campaign), 政府 (government) |
| EN-16 (.32) | sex, law, act, sexual, marriage, court |
| EN-11 (.17) | traffic, victim, government, trafficking, child, force |
| EN-10 | album, release, record, music, song, single |
| ZH-9 (.30) | 专辑 (album), 张 ([Q] album), 发行 (release), 音乐 (music), 首 ([Q] song), 唱片 (record) |
| ZH-17 (.20) | 音乐 (music), 乐团 (musical group), 艺术 (art), 创作 (create), 奖 (award), 演出 (perform) |

Table 2: Topics are linked because they have overlap in topical words. Our MTM can also infer the topic relations beyond words, e.g., ZH-14 and EN-16.

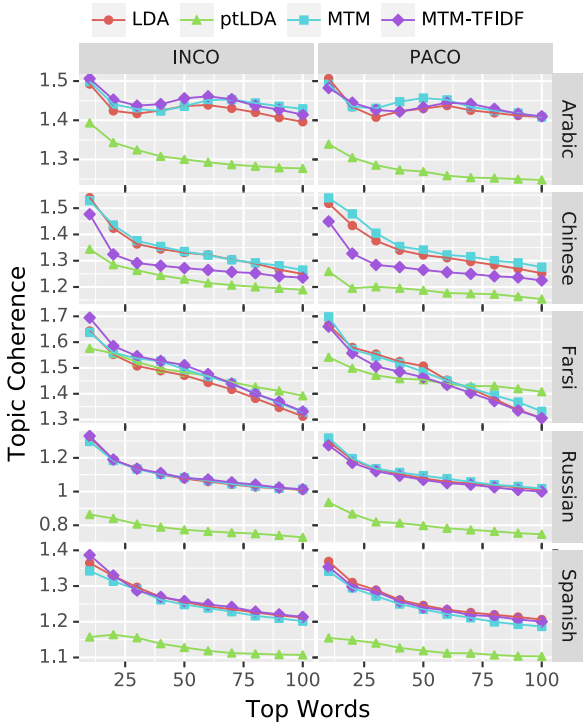


Figure 4: Topic coherence on INCO and PACO datasets with the number of top words in each topic.

MTM identifies the two English topics as the top linked topics for ZH-14: EN-16 is about the "campaign" in Sexual Rights, while EN-11 is about the Crime of human trafficking. This shows that our MTM can incorporate word translations and infer more cross-lingual word and topic relationships.

### 3.4 Evaluating Topic Coherence

We intrinsically evaluate models' topic coherence on two sets of preprocessed bilingual Wikipedia corpora (Hao and Paul, 2018) that vary in "non-

parallelness". Both pair English with Arabic, Chinese, Spanish, Farsi, and Russian. In PACO, 30% of documents have direct translations across languages, and in INCO none has direct translations. Dictionaries are extracted fromn Wiktionary.[5] Standard preprocessing has already been applied to the datasets, including stemming, stop word removal, and high-frequency word removal.

We use an intra-lingual metric to evaluate topic coherence (Lau et al., 2014): for every topic, we compute its top $N$ words' average pairwise PMI score on a disjoint subset of Wikipedia documents (Hao and Paul, 2018). We report average coherence with $N$ from 10 to 100 with a step size of 10 (five-fold cross-validation). We use the same translation pair weighting options as in our classification tasks and also compare against monolingual LDA and ptLDA (Hu et al., 2014).

MTM is no worse than LDA and sometimes slightly better (Figure 4). TF-IDF weighting on translation pairs sometimes further improves coherence (e.g., Arabic, Farsi, Russian, and Spanish on INCO) but occasionally hurts (e.g., Chinese). ptLDA mostly works poorly, except on Farsi with a high number of top words. ptLDA aligns topic spaces, which is hard for low-comparability data, thus sacrificing coherence for alignment; in contrast MTM only connects topics when they align well in senses.

### 3.5 Topic Coherence vs. Target Language Corpora Sizes

We next vary the size of target language (non-English languages in PACO and INCO) corpora: how much can MTM help topic coherence for low-resource languages? We start from 10% of the randomly-selected documents in target languages and incrementally add more target language documents at a step size of 10% until it reaches 100%. Meanwhile, we always use 100% of the English documents. We train monolingual LDA, ptLDA, and MTMs with and without TF-IDF weighting on translation pairs on each setting and evaluate the topic coherence on the same reference corpora using the top thirty words of each topic (Figure 5).

In most cases, the topic coherence improves with larger target corpora, except Arabic and Russian on PACO. This confirms our intuition that more data yield a better topic model. MTM is helpful in cases when the target language corpora sizes are small, e.g., Chinese and Russian with 10% or 20% of the corpora. TF-IDF weighting is not consistently better or worse than equal weights.

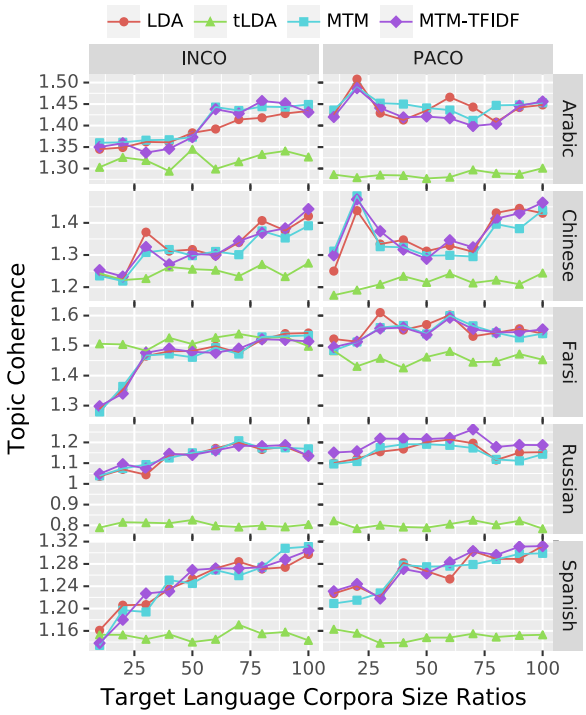


Figure 5: The models' topic coherence on INCO and PACO datasets when the sizes of target language corpora grow from 10% to 100%, with a step size of 10%.

The ptLDA with tree priors based on dictionaries performs poorly in topic coherence, except Farsi in INCO. In most cases, its topic coherence is substantially below others' and improves little when the target corpora grow.

## 4 Conclusions and Future Work

We introduce a novel multilingual topic model (MTM) that learns weighted topic links across languages by minimizing the Euclidean distances of translation pairs' (transformed) topic distributions, where translation pairs can be weighted, e.g., by TF-IDF. This connects topics in different languages *only* when necessary and is more robust on low-comparability corpora. The MTM outperforms baselines substantially in both intra- and cross-lingual classification tasks, while achieving no worse or slightly better topic coherence than monolingual LDA on low-comparability data.

We plan to explore weighting methods to better evaluate the importance of translation pairs. We will also study how to improve topic transformation with the topic link weight matrices.

[5] https://dumps.wikimedia.org/enwiktionary/

## Acknowledgements

We thank Shudong Hao and Michelle Yuan for providing their datasets. We thank the anonymous reviewers for their insightful and constructive comments. This research has been supported under subcontract to Raytheon BBN Technologies, by DARPA award HR0011-15-C-0113. Boyd-Graber is also supported by NSF grant IIS-1409287. Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the view of the sponsors.

## References

David Andrzejewski, Xiaojin Zhu, and Mark Craven. 2009. Incorporating domain knowledge into topic modeling via Dirichlet forest priors. In *Proceedings of the International Conference of Machine Learning*.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, pages 993–1022.

Jordan Boyd-Graber, Yuening Hu, and David Mimno. 2017. *Applications of Topic Models*, volume 11 of *Foundations and Trends in Information Retrieval*. NOW Publishers.

Gilles Celeux. 1985. The SEM algorithm: A probabilistic teacher algorithm derived from the EM algorithm for the mixture problem. *Computational Statistics Quarterly*, pages 73–82.

Pascale Fung. 2000. A statistical view on bilingual lexicon extraction. In *Parallel Text Processing*, pages 219–236.

Thomas L. Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences*, pages 5228–5235.

E. Dario Gutiérrez, Ekaterina Shutova, Patricia Lichtenstein, Gerard de Melo, and Luca Gilardi. 2016. Detecting cross-cultural differences using a multilingual topic model. *Transactions of the Association for Computational Linguistics*, pages 47–60.

Shudong Hao and Michael J. Paul. 2018. Learning multilingual topics from incomparable corpora. In *Proceedings of International Conference on Computational Linguistics*.

Yuening Hu, Ke Zhai, Vlad Eidelman, and Jordan Boyd-Graber. 2014. Polylingual tree-based topic models for translation domain adaptation. In *Proceedings of the Association for Computational Linguistics*.

Jey Han Lau, David Newman, and Timothy Baldwin. 2014. Machine reading tea leaves: Automatically evaluating topic coherence and topic model quality. In *Proceedings of the Association for Computational Linguistics*.

Dong C. Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, pages 503–528.

David Mimno, Hanna M. Wallach, Jason Naradowsky, David A. Smith, and Andrew McCallum. 2009. Polylingual topic models. In *Proceedings of Empirical Methods in Natural Language Processing*.

Xiaochuan Ni, Jian-Tao Sun, Jian Hu, and Zheng Chen. 2009. Mining multilingual topics from Wikipedia. In *Proceedings of the World Wide Web Conference*.

Bei Shi, Wai Lam, Lidong Bing, and Yinqing Xu. 2016. Detecting common discussion topics across culture from news reader comments. In *Proceedings of the Association for Computational Linguistics*.

Stephanie M. Strassel and Jennifer Tracey. 2016. LORELEI language packs: Data, tools, and resources for technology development in low resource languages. In *Proceedings of the Language Resources and Evaluation Conference*.

Yi Yang, Doug Downey, and Jordan Boyd-Graber. 2015. Efficient methods for incorporating knowledge into topic models. In *Proceedings of Empirical Methods in Natural Language Processing*.

Michelle Yuan, Benjamin Van Durme, and Jordan Boyd-Graber. 2018. Multilingual anchoring: Interactive topic modeling and alignment across languages. In *Proceedings of Advances in Neural Information Processing Systems*.

# Measure Country-Level Socio-Economic Indicators with Streaming News: An Empirical Study

**Bonan Min**
Raytheon BBN Technologies
bonan.min@raytheon.com

**Xiaoxi Zhao**
Department of Economics, Boston University
xiaoxiz@bu.edu

## Abstract

Socio-economic conditions are difficult to measure. For example, the U.S. Bureau of Labor Statistics needs to conduct large-scale household surveys regularly to track the unemployment rate, an indicator widely used by economists and policy makers. We argue that events reported in streaming news can be used as "micro-sensors" for measuring socio-economic conditions. Similar to collecting surveys and then counting answers, it is possible to measure a socio-economic indicator by counting related events. In this paper, we propose Event-Centric Indicator Measure (ECIM), a novel approach to measure socio-economic indicators with events. We empirically demonstrate strong correlations between ECIM values to several representative indicators in socio-economic research.

## 1 Introduction

Socio-economic indicators are powerful instruments for measuring economic conditions and the sociocultural environment that people live in. They are widely used to inform policy makers, and help them to measure outcome of policy interventions. They are often difficult, if not impossible, to measure. Take unemployment rate as an example, the U.S. Bureau of Labor Statistics conducts large-scale household surveys in order to track it. More abstract indicators (e.g. economic uncertainty), which involve vague or complex social interactions, are very difficult to measure accurately.

We hypothesize that streaming news, reporting a vast amount of real-world events, can be used for measuring socio-economic indicators. We propose Event-Centric Indicator Measure (ECIM), a novel approach to measure socio-economic indicators using events extracted from streaming news. We demonstrated that ECIM is effective; ECIM values are strongly correlated with representative socio-economic indicators.

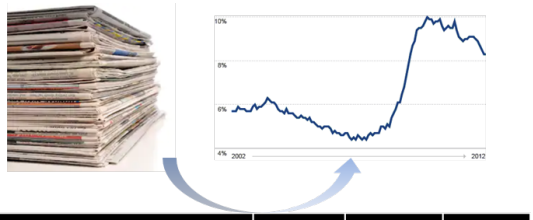

| Event triggers | Location | Time | Related* |
|---|---|---|---|
| the U.S. **recession** began in December 2007 | U.S. (USA) | December 2007 (2007-12) | Y |
| SEPT. 1, 2011 The **bankruptcies** of three American solar power companies in the last month | American (USA) | Last month (2011-08) | Y |
| The man accused of fatally **shooting** his estranged wife inside a New Jersey church last Tuesday | New Jersey (USA) | Last Tuesday (2009-07) | N |
| … | … | … | |

Figure 1: The ECIM workflow. Words in bold are event triggers. Text in parentheses shows normalized locations (countries) and time. "Related" shows whether an event trigger is related to the target indicator (e.g., "unemployment rate") based on keyword matching.

We will first present an overview of the ECIM approach, and then describe how we extract events from text and aggregate them to calculate ECIMs for each socio-economic indicator. We will then present large-scale experiments to demonstrate that ECIM is effective, using several widely used socio-economic indicators.

## 2 Event-Centric Indicator Measure

The ECIM workflow is summarized in Figure 1. Given a large-scale streaming news collection, the system first extracts event mentions (event trigger words or phrases) along with their locations and time. For each socio-economic indicator, the system will then aggregate relevant events per each time step to produce ECIM, a time series measuring the corresponding socio-economic condition.

### 2.1 Extract Events from Text

We extract event frames from predicate-argument structures that are automatically generated from



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1249–1254, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

text. Then we extract an event trigger, a location and a time, if available, from each event frame. To attach a location or time that is further away from the trigger (or based on the publishing date), we apply a few inference rules.

**Extract Events** Syntactic-semantic representations such as Abstract Meaning Representation (Huang et al., 2016) and Semantic Role Labeling (SRL) (Peng et al., 2016; Surdeanu et al., 2003), have been shown to be effective for event extraction. Following (Peng et al., 2016; Surdeanu et al., 2003), our event extractor is based on tagging predicates (verbs and eventive nouns) and their locations and temporal arguments.

We first apply SRL (Punyakanok et al., 2004; He et al., 2017) over each sentence. An example SRL representation on a sentence is shown in Figure 2. From the predicate-argument structure, the system then extracts the predicate as an event trigger [1], the entity mention and time mention attached to the predicate through AM-LOC and AM-TMP as the location and time for the event respectively. The location is resolved to a country-level GPE by looking up part-of relations in GeoNames [2] (e.g., convert "Boston" into "U.S."). Our system also normalizes time mentions into Timex2 (Ferro et al., 2001) and resolves relative time (e.g., *last Tuesday*) into Timex2, based on the document publishing date. This process extracts an event mention in the form of a triple <trigger: *clashed*, location: Syria, time: 2018-11-23> from the sentence in Figure 2.




Figure 2: An SRL representation of a sentence.

**Find relevant event triggers** Given an indicator, we use the following approach to find a set of relevant event trigger words/phrases:

- We process a development corpus (500 documents from the English Gigaword [3]) with the above-mentioned approach to extract all event triggers, and then ask an annotator to find trigger words for the target indicator, starting with the most frequent trigger.
- We then use WordNet (Miller, 1995) synsets to automatically expand the trigger word list. We also use word embedding similarities to search for new trigger words that are most similar (e.g., using cosine similarity) to the centroid of existing triggers.

An annotator then reviews the expanded triggers and removes incorrect ones. The whole process [4] takes less than 10 minutes per indicator. Example triggers for three representative indicators are listed in Table 1.

To tag event mentions, we match the predicate to the relevant keyword list [5], constructed using the processed described above.

**Infer location and time** Locations are crucial for identifying whether events are related to the country of interests. Time is necessary for binning events by time steps for counting. Extracting time and locations for event triggers can be challenging if they are further apart from the trigger. Oftentimes no location or time is mentioned in the same sentence where the event is stated.

To increase the coverage of location and time for events, we apply the following inference rules:

- *Same-sentence*: if the event extractor doesn't find a location or a time, but there is one and only one location (an entity mention with type GPE or Location), and/or one and only one time mention in the same sentence, it will attach the location and/or time to the event.
- *Document metadata*: A news article often come with a publishing date and a location. When neither the extractor nor the *same-sentence* rule found a location or a time for an event, we will attach the publishing location and/or date to the event.

## 2.2 Measure Socio-economic Indicators

We filter events by location (target country) and then use the following steps to calculate ECIM:

- *Aggregation and counting*: We bin events by time step [7]. For each time step $t$ and event $e \in$

[1] We expand nouns into their base noun phrases.
[2] https://www.geonames.org/
[3] https://catalog.ldc.upenn.edu/LDC2011T07
[4] We used the UI described in (Chan et al., 2019) to facilitate this process.
[5] The keyword list is available at https://github.com/BBN-E/ecim.
[6] https://en.wikipedia.org/wiki/VIX
[7] One month is used as the time step, since not all numeric indices come with higher-frequency data for validating our approach.

| Indicator | Event triggers | Text |
|---|---|---|
| Unemployment rate | *unemploy, layoff, dismissal, lay off, economic crisis, bankrupt* | *Apple to* **lay off** *190 employees from self-driving car unit.* |
| CBOE Volatility Index (VIX) [6] | *sell share, liquidate, scandal, debt, loan* & (events for EPU) | *Humana executives* **sell** *nearly 74000 shares worth $22.7 million.* |
| Economic Policy Uncertainty (EPU) | *economic slowdown, institutional weaknesses, war, crisis, terrorist attack* | *These policies and other* **institutional weaknesses** *continue to undermine prospects for sustained economic development* |

Table 1: Indicators, examples of event triggers (words or phrases) and sentences

$\mathbb{E}_i$ ($\mathbb{E}_i$ is the set of events for indicator $i$), we then generate count $N_{e,t}$. For each indicator $i$, we aggregate the counts: $\sum_{e \in \mathbb{E}_i} N_{e,t'}$.

- *Normalizing counts*: Those counts are not normalized and can be inflated due to the increasing level of media activity. To normalize, we divide them by the total number of articles published in each month $M_t$.

- *Smoothing*: To remove noise, we smooth the normalized counts by calculating moving averages [8] centered around each $t$ with a window of $T = 7$ time steps [9].

In summary, ECIM for indicator $i$ at time $t$ is defined as

$$ECIM_{i,t} = \frac{1}{T} \sum_{t' \in [t-\frac{T}{2}, t+\frac{T}{2}]} \frac{\sum_{e \in \mathbb{E}_i} N_{e,t'}}{M_{t'}}$$

## 3 Experiments

We use the English Gigaword corpus [10], which consists of 5.7 million articles published from 1994 to 2010, from a wide range of sources including the New York Times, the Associated Press, Los Angeles Times, Washington Post, Agence France-Presse, Central News Agency of Taiwan, and Xinhua News Agency.

We run event extractors on this corpus to extract over 10 million event mentions that happen in the U.S. from 1994 to 2010. We then generate ECIMs for 3 representative socio-economic indicators: (1) Unemployment rate—a crucial index of economy and for policy making, (2) Chicago Board Options Exchange (CBOE) Volatility Index (VIX) —a widely-used market volatility measurement, and (3) Economic Policy Uncertainty (EPU) (Baker et al., 2016)—a policy uncertainty index. We focus on these 3 indicators because (1) they are widely used in economics and social science research, and (2) their data are publicly available [11].

Figure 3 shows the time series of ECIMs (blue solid lines) and that of the 3 corresponding indicators (red dash lines):

**Unemployment rate**: The downward/upward trends and the peaks of the two lines match each other quite well. This shows the ECIM is correlated with unemployment rate. There are some delay between peaks and downward trends shown in the line for unemployment rate: not surprisingly, unemployment rate reacts to events such as "economic downturn" with delay, and its recovery takes longer time than media coverage on "economic downturn" events. Among the events detected, our system found many events that can be labeled as "economic downturn" (e.g., "recession", "depression", "financial crisis") or "bankruptcy" (e.g., "bankrupt"). This matches our intuition that economic downturn and more bankruptcies are often correlated with higher unemployment rate (Reinhart and Rogoff, 2009).

**VIX**: Figure 3 shows that the ECIM matches very well with VIX over time. We found that high market volatility is strongly correlated with unfavorable macroeconomic events such as "economic crisis", firm-level events such as "bankruptcy", as well as its after effects such as "loan".

**EPU**: The ECIM strongly correlates with EPU. Similarly, economic crisis, which often led to high economic policy uncertainty, is found to be among events the most frequently detected. In addition, our system found extreme events such as "attack", "conflict", and "terrorism" which may trigger major changes of economic policy. The slight deviation in 2003-2004 is caused by low-coverage of

[8] https://en.wikipedia.org/wiki/Moving_average

[9] Other window sizes generate similar results. Due to space limitation, we only present results using 7 time steps.

[10] https://catalog.ldc.upenn.edu/LDC2011T07.

[11] We downloaded these three datasets from Federal Reserve Economic Data, CBOE, and EPU websites, respectively. For VIX which is available daily, we generated monthly averages. The cleaned version of the time-series data is available at https://github.com/BBN-E/ecim.

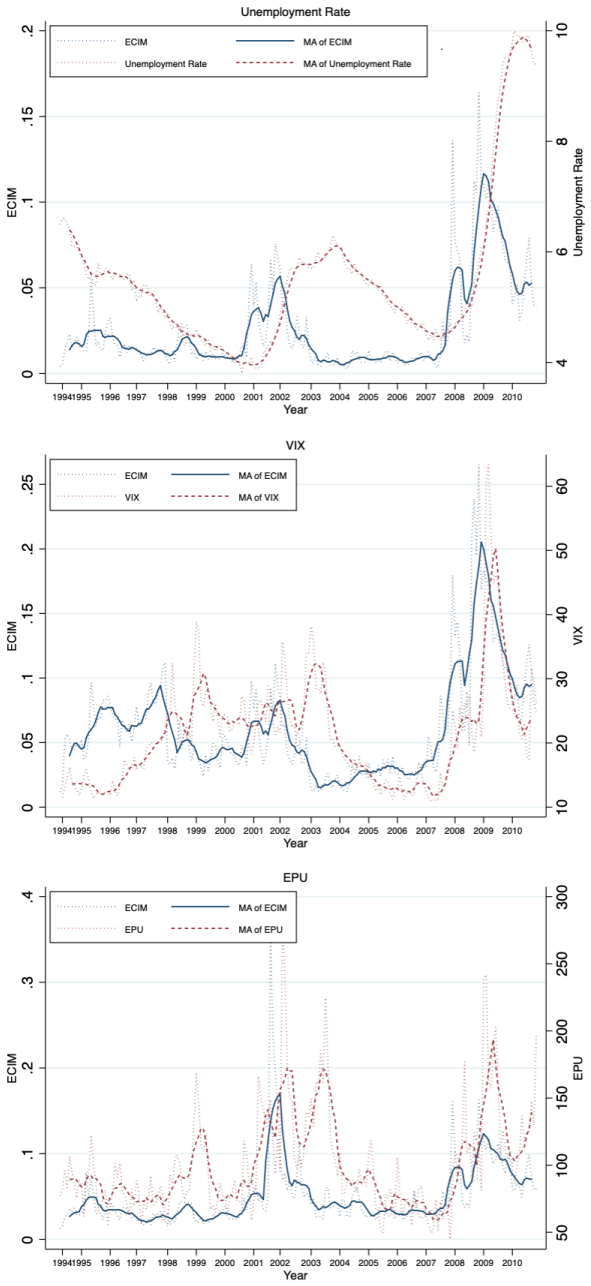


Figure 3: Values of ECIMs (blue solid lines) and the 3 corresponding socio-economic indicators (red dash lines) unemployment rate, VIX and EPU through time. Blue solid lines show ECIMs. Dotted lines show unsmoothed values. Solid and dash lines show the moving average (MA) which are smoothed values. X-axis is the year. Left Y-axis is for the ECIM value. Right Y-axis is for the indicator.

the Iraq War in Gigaword.

Table 2 shows quantitative correlation analysis between ECIMs and their corresponding indicators. Pearson correlation coefficients[12] show strong correlation between these two for each indicator. We also test the hypothesis that ECIMs and their corresponding indicators are independent (p-value is shown in the third column). For all three indicators, we are 99.99% confident to reject the hypothesis of independence.

[12] en.wikipedia.org/wiki/Pearson_correlation_coefficient

| Indicator | Pearson | P-value |
|---|---|---|
| Unemployment rate ($1/\mathbb{M}_{comp}$) | 0.4286 (0.4877) | 0.0000 (0.0000) |
| EPU News | 0.5136 | 0.0000 |
| VIX | 0.4115 | 0.0000 |

Table 2: Correlation coefficients between ECIMs and indicators. For unemployment rate, we also show its correlation with $1/\mathbb{M}_{comp}$ (in parentheses).

**On negative correlation** We further study negative correlation between some events and indicators. We hypothesize that "Competition" is positively correlated with more participants in a market, therefore a higher demand for labor which means lower unemployment rate. We construct an ECIM $\mathbb{M}_{comp}$ for "Competition" and then plot $1/\mathbb{M}_{comp}$ over time against unemployment rate. It is important to note that we use $1/\mathbb{M}_{comp}$ to flip the line for "Competition" upside down, so that the better the two lines aligns with each other, the more these two are negatively correlated.

Figure 4 shows a strikingly high negative correlation between "Competition" and unemployment rate. The correlation coefficients between unemployment rate and $1/\mathbb{M}_{comp}$ are also very high as shown in the parentheses in the first row of Table 2. This points to a promising future direction: it is possible to measure an indicator using an event that occurs more frequently when the value of the indicator is low.

## 4 Related Work

Most prior work on constructing or measuring indicators(Bansal et al., 2005; Jurado et al., 2015; Bachmann et al., 2013) is in social or economic science research. Recent work (Dzielinski, 2012; Alexopoulos and Cohen, 2015; Baker et al., 2016) tries to incorporate text into economic research with keyword-based approaches. Similar to ECIM, EPU (Baker et al., 2016) applies a keyword-counting approach (with keywords "economic", "policy" and "uncertainty") to measure economic policy uncertainty. In contrast, this paper uses the richer information from syntactic-semantic analysis of text. Furthermore, EPU uses

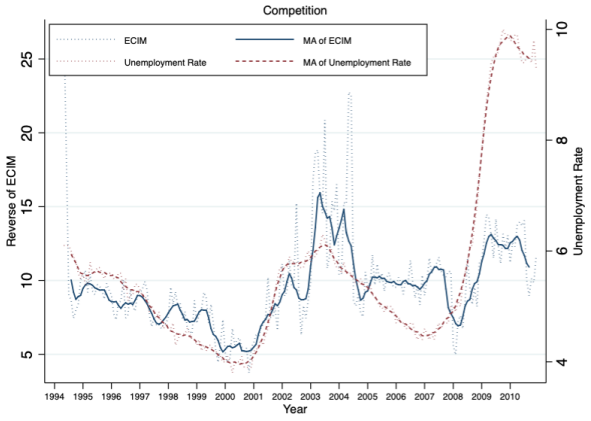


Figure 4: Values of $1/\mathbb{M}_{comp}$ (blue solid line), in which $\mathbb{M}_{comp}$ is the ECIM for "Competition", and unemployment rate (red dash line), through time.

the aggregated sentiment (by counting the number of times people expressed their views on economic policy uncertainty) as the measure, but we use a more objective approach which measures events that are correlated to the uncertainty.

(Rohlfs et al., 2016) applied supervised topic modeling to measure the effects of Federal Open Market Committee text content on the direction of short- and medium-term interest rate movements.

## 5 Conclusion and Future Work

This paper presents ECIM, a novel approach to measure socio-economic indicators with news events. Experiments show strong correlations between ECIM values and representative indicators in socio-economic research.

Our next steps are to further study the correlation with time lags, and to incorporate more sophisticated event extraction techniques.

## Acknowledgements

The authors would like to thank the anonymous reviewers for their insightful comments, which helped us to improve the final version of the paper.

This work was supported by DARPA/I2O and U.S. Army Research Office Contract No. W911NF-18-C-0003 under the World Modelers program. The views, opinions, and/or findings contained in this article are those of the author and should not be interpreted as representing the official views or policies, either expressed or implied, of the Department of Defense or the U.S. Government. This document does not contain technology or technical data controlled under either the U.S. International Traffic in Arms Regulations or the U.S. Export Administration Regulations.

## References

Michelle Alexopoulos and Jon Cohen. 2015. The power of print: Uncertainty shocks, markets, and the economy. *International Review of Economics & Finance*, 40:8–28.

Rüdiger Bachmann, Steffen Elstner, and Eric R Sims. 2013. Uncertainty and economic activity: Evidence from business survey data. *American Economic Journal: Macroeconomics*, 5(2):217–49.

Scott R Baker, Nicholas Bloom, and Steven J Davis. 2016. Measuring economic policy uncertainty. *The Quarterly Journal of Economics*, 131(4):1593–1636.

Ravi Bansal, Varoujan Khatchatrian, and Amir Yaron. 2005. Interpretable asset markets? *European Economic Review*, 49(3):531–560.

Yee Seng Chan, Joshua Fasching, Haoling Qiu, and Bonan Min. 2019. Rapid customization for event extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 31–36, Florence, Italy. Association for Computational Linguistics.

Michal Dzielinski. 2012. Measuring economic uncertainty and its impact on the stock market. *Finance Research Letters*, 9(3):167–175.

Lisa Ferro, Inderjeet Mani, Beth Sundheim, and George Wilson. 2001. Tides temporal annotation guidelines version 1.0. 2. *The MITRE Corporation, McLean-VG-USA*.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 473–483, Vancouver, Canada. Association for Computational Linguistics.

Lifu Huang, Taylor Cassidy, Xiaocheng Feng, Heng Ji, Clare R Voss, Jiawei Han, and Avirup Sil. 2016. Liberal event extraction and event schema induction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 258–268.

Kyle Jurado, Sydney C. Ludvigson, and Serena Ng. 2015. Measuring Uncertainty. *American Economic Review*, 105(3):1177–1216.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Haoruo Peng, Yangqiu Song, and Dan Roth. 2016. Event detection and co-reference with minimal supervision. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 392–402.

Vasin Punyakanok, Dan Roth, Wen-tau Yih, and Dav Zimak. 2004. Semantic role labeling via integer linear programming inference. Technical report, ILLINOIS UNIV AT URBANA-CHAMPAIGN DEPT OF COMPUTER SCIENCE.

Carmen M Reinhart and Kenneth S Rogoff. 2009. The aftermath of financial crises. *American Economic Review*, 99(2):466–72.

Christopher Rohlfs, Sunandan Chakraborty, and Lakshminarayanan Subramanian. 2016. The effects of the content of fomc communications on us treasury rates. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2096–2102.

Mihai Surdeanu, Sanda Harabagiu, John Williams, and Paul Aarseth. 2003. Using predicate-argument structures for information extraction. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*.

# Towards Extracting Medical Family History from Natural Language Interactions: A New Dataset and Baselines

**Mahmoud Azab**[1], **Stephane Dadian**[1], **Vivi Nastase**[2], **Larry An**[1], **Rada Mihalcea**[1]
[1]University of Michigan, Ann Arbour, Michigan,
mazab@umich.edu, sdadian@umich.edu
lcan@umich.edu, mihalcea@umich.edu
[2]University of Heidelberg, Heidelberg, Germany
nastase@cl.uni-heidelberg.de

## Abstract

We introduce a new dataset consisting of natural language interactions annotated with medical family histories, obtained during interactions with a genetic counselor and through crowdsourcing, following a questionnaire created by experts in the domain. We describe the data collection process and the annotations performed by medical professionals, including illness and personal attributes (name, age, gender, family relationships) for the patient and their family members. An initial system that performs argument identification and relation extraction shows promising results – average F-score of 0.87 on complex sentences on the targeted relations.

## 1 Introduction

Individual family history is an important medical tool for diagnosis, disease prevention and treatment. Macro analyses of family histories provide useful and important information for determining disease susceptibility by researching genetic influences between family histories and diseases. This can be used to determine levels of risk for certain illnesses for a given family member, and potentially develop a protocol of preventative screening. In-person appointments with genetic counselors are time-consuming and suffer from limitations such as long wait times and small number of genetic counseling centers (Sutphen et al., 2010; Nathan et al., 2016). Moreover, collecting a patient's family history away from pressured consultation environment results in more accurate and complete information (Nathan et al., 2016).

An interesting solution would be the deployment of a dialog agent to collect such data. The interaction between the agent and the patient can vary between two extremes: (a) pose very specific questions about each member of the family and their history – difficult because of the need to identify very clearly the family relations in real time during the interaction, but easier because the answers will be short and focused; (b) a system that asks more generic questions, and elicits more narrative answers from the patients – easier because the interaction with the patient is more generic and does not have to be adjusted in real time, but harder because the elicited answers are longer and more free-form. To support research in this area, we choose to explore the second end of the spectrum, which favours a simpler interaction and more intense processing for relation extraction. We introduce a dataset consisting of free-form natural language responses to a medical cancer family history questionnaire created by experts in the domain. The dataset consists of 20,774 annotated relations (illness, name, age, gender, family relationships) in 228 family histories answering 159 questions each, collected through crowdsourcing or as test cases of genetic counseling sessions. The collected questionnaires were extensively annotated by genetic counseling professionals – medical staff and students from the University of Michigan – by identifying family relations, illnesses and person attributes (name, age, gender, family relationships) for family members, to construct complete family history records.

Due to its potentially high impact, extracting family history from clinical notes, admission reports, or questionnaires specifically designed to elicit this type of information is an important area of research (Lewis et al., 2011; Bill et al., 2014; Goryachev et al., 2008). The specific data used in previous studies focused on different types of attributes/relations, and were not made available for further research, even though some used synthetically produced clinical notes.

The dataset we collected and the annotations



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1255–1260, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

are publicly available[1]. We also provide a baseline system for relation extraction, which identifies relation arguments and the relation between them using a neural network with a CRF layer to produce attribute annotations, and then a simple neural network for binary classification of relations. The system achieves 0.87 average F-score when trained on data form both crowd-sourcing and genetic counseling sessions.

## 2 Related Work

The scope and approach to extracting relations related to family illness history from clinical notes is varied, and relied on small annotated datasets which have not been shared.

From a corpus of synthetically produced clinical notes (MTSamples), Bill et al. (2014) identify 284 sentences as containing family history. Attributes (vital status, negation, etc.) are annotated by medical and informatics experts. Identifying the relation and its arguments relies on a combination of heuristics and learning based on lexicalized features. Goryachev et al. (2008) use association rules to find family members and their illness history (a pre-specified set of 8 illnesses) in discharge summaries and outpatient visit notes. The system is run on 2000 reports randomly selected from 4 different hospitals' clinical notes. 1000 of the relations detected by the system were manually inspected for evaluation. Lewis et al. (2011) rely on grammatical dependencies and patterns of dependency sequences to detect family history information, constraining one of the arguments of a relation to express a family member (e.g. mother, brother). The system is trained on 299 sentences, 77 of which contained 167 person-diagnosis pairs. Rama et al. (2018) iteratively develop an annotation schema and a synthetic corpus of clinical notes in Norwegian for family history annotation and extraction. They produce a corpus of 477 sentences, with 4154 entities and 2078 relations. Entity detection and relation extraction are performed separately, using an SVM with lexical, POS and dependency features.

These approaches rely on heuristics to detect the arguments of relations: use a predefined list of family relationships and diseases, or use as arguments the noun phrases that are detected close to the suspected relationship markers. Finding relation arguments is difficult because of their variable length. Sequence labelling approaches have proved successful for such problems (Ramshaw and Marcus, 1995; Lample et al., 2016; Ma and Hovy, 2016; Chiu and Nichols, 2016). On the relation extraction side, the recently developed deep learning approaches have also proved useful, by successfully combining information about the meaning of the arguments, their context and their grammatical connections (Zeng et al., 2014; Liu et al., 2015; dos Santos et al., 2015). Determining the relation arguments and the relation itself depend on each other, so it would be beneficial to have an approach that performs these two tasks jointly (Miwa and Bansal, 2016).

Zheng et al. (2017) propose a new tagging scheme for relations in a text, and use an encoder-decoder LSTM-based model to predict for an input sentence the tags that combine relation and argument information. We will use a similar tag set, but in a two step approach.

## 3 Dataset

Our dataset consists of answers to a family history questionnaire collected from volunteers through Amazon Mechanical Turk (AMT) and from test cases of genetic counseling sessions (GCS).[2] This questionnaire has been developed by cancer genetic counseling experts to construct a patient's medical history (Wattendorf and Hadley, 2005). It consists of 159 questions targeting the medical history of the patient and their relatives up to third degree (including the patient's great-grandparents and cousins). Some questions require a brief and precise answer ("what is your age?"), while the majority are open-ended ("Please describe each of your health issues or problems.") Our final dataset contains 228 questionnaire responses, 156 from AMT and 72 from GCS. In the next step the dataset will be annotated by experts with information relevant to medical histories.

| | AMT | GCS | Total |
|---|---|---|---|
| # questionnaires | 156 | 72 | 228 |
| # sentences | 2,993 | 1,311 | 4,304 |
| # tokens | 30,700 | 11,131 | 41,831 |

Table 1: Dataset size statistics

[1] http://lit.eecs.umich.edu/downloads.html

[2] Our study was exempt from IRB because the contributors were anonymous. Moreover, the family histories reported do not necessarily reveal real patient information.

My brother is named Michael, he is 42. I have two sisters, Lydia who is 38 and July who is 36.

```
My <ft id=1 rel=brother gender=male>brother</ft> is named <ft id=1 name=Michael
    >Michael</ft>, he is <ft id=1 age=42>42</ft>. I have two <ft id=2,3 rel=
    sister gender=female source=self>sisters</ft>, <ft id=2 name=Lydia>Lydia</
    ft> who is <ft id=2 age=38>38</ft> and <ft id=3 name=July>July</ft> who is
    <ft id=3 age=36>36</ft>.
```

My niece Alicia has Asthma.

```
My <ft id=1 rel=niece gender=female>niece</ft> <ft id=1 name=Alicia>Alicia</ft>
     has <ft id=1 illness=asthma>Asthma</ft>.
```

Figure 1: Sample annotations

**Annotations.** We worked with genetic counseling experts to define the set of annotations that would enable them to construct a patient's medical history: (i) all the persons in one's family tree; and (ii) the attributes associated with a person. We target the following five annotation types: *FT* (family tree member), *name, age, gender, illness* along with their attributes, shown in Table 2. Every family member is assigned a unique *ft_id*, which is used to connect all their information throughout the questionnaire. A person can have several health issues, so each illness is assigned an *ill_id*, which is unique when combined with *ft_id*. If an illness is a cancer, a cancer flag is set to true and a cancer type is selected from a list of 23 cancer types. To handle complex cases as in the phrase *both cousins suffered from high blood pressure*, every annotation can refer to multiple *ft_ids* and *ill_ids*.

| Type | attributes |
|---|---|
| FT | ft_id, parent_id_1, parent_id_2, deceased {T/F} |
| Name | ft_id |
| Gender | ft_id, sex: {male/female/unknown} |
| Age | ft_id, age_at_diagnosis {T/F}, age_at_death {T/F} |
| Illness | ft_id, ill_id, cause_of_death {T/F}, cancer {T/F}, cancer_type {23 types} |

Table 2: The annotation types included in our dataset with their corresponding attributes.

Each questionnaire response is manually annotated by a genetic counseling expert, using Brat to facilitate the annotation process (Stenetorp et al., 2012).[3] Figure 1 shows an annotation example.

[3]To ensure the quality of our questionnaires, we discard questionnaires that contain spam responses and the ones in which the patients did not use proper natural language to answer such as (e.g. "Mary/f/20").

To evaluate the quality of the annotations, four full questionnaires were annotated by two judges. The Cohens Kappa inter-annotator agreement score for these questionnaires for annotation types and their corresponding attributes is 0.8908, showing high agreement.

The final dataset consists of 4,304 sentences of various complexity w.r.t. the relations included – "simple sentences" containing information about a single family member, and "complex sentences" containing information about two or more family members. (Table 3).

| | Simple sentences | | Complex sentences | |
|---|---|---|---|---|
| | #sent | #rels | #sent | #rels |
| AMT | 2413 | 7069 | 520 | 7374 |
| GCS | 453 | 1409 | 803 | 4923 |
| Total | 3214 | 8477 | 973 | 12,297 |

Table 3: Sentence and relations statistics

The two data subsets (AMT, GCS) were obtained using the same questionnaire, but they are slightly different. Compared to the GCS data which contains about twice as many complex sentences as simple ones, the AMT data contains approx. five times more simple sentences than complex ones. However, AMT's complex sentences are much denser in information than the complex sentences in the GCS data – 14.2 vs 6.1 relations per sentence.

Table 4 shows the overall annotation statistics.

## 4 Relation Extraction

To evaluate the viability of our annotations, we develop a pipeline for relation extraction, illustrated in Figure 2. It works in two steps: (i) argument (entity/attributes) identification and (ii) relation classification. We split our dataset into

| Annotation type | AMT | GCS | Total |
|---|---|---|---|
| # FT | 2,633 | 1,128 | 3,761 |
| # Name | 1,640 | 1,225 | 2,865 |
| # Gender | 879 | 601 | 1,480 |
| # Age | 1,094 | 914 | 2,008 |
| # Illness | 1,636 | 441 | 2,077 |
| # cancer | 428 | 346 | 774 |
| # age at diagnosis | 645 | 179 | 824 |
| # age at death | 206 | 156 | 362 |
| # cause of death | 478 | 227 | 705 |
| # relations | 14,443 | 6,322 | 20,774 |

Table 4: Annotation types, attributes, and relations statistics.

train, validation, and test based on the questionnaire identifier (Table 5). Every sentence in our data is tokenized and part-of-speech tagged using Stanford CoreNLP (Manning et al., 2014). Our pipeline processes an input sentence in two steps. First, we tag every token with its corresponding annotation type, including entities and attributes, using the IOB (Inside, Outside, Beginning) tagging scheme. The tagged sentence is then passed to a binary relation classifier to classify the relations between every pair of entities in this sentence. Figure 2 shows an overview and an example of our model. The two steps of the pipeline are detailed below.

| | AMT | | | GCS | | |
|---|---|---|---|---|---|---|
| | train | val | test | train | val | test |
| #questionnaires | 91 | 15 | 50 | 45 | 7 | 20 |
| #sentences | 1574 | 372 | 1047 | 893 | 151 | 267 |

Table 5: Data distribution in each split.

**Entity Identification.** We model entity identification as a sequence labeling problem using a conditional random field (CRF) classifier.[4] [5] The set of labels used consist of the annotation labels (Table 2) combined with the IOB markers (Inside, Outside, Beginning).

The model is trained using the following features:

*Lexical*: current word and words in context window of size 3

*Part-of-speech*: POS tags of current word and words context window of size 3

*Binary features*: is beginning of sentence, starts with capital letter, is digit

[4] sklear-crfsuite (Okazaki, 2007)

[5] Only the results of the best classifier are reported, but we experimented with other classifiers including logistic regression and BiLSTM-CRF.

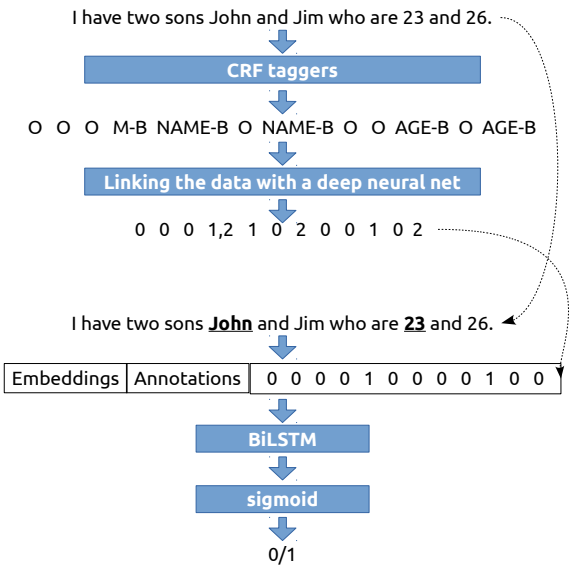


Figure 2: Method pipeline

**Relation Classification** The entity identification step outputs entity type tags for every token of a given sentence, as illustrated in Figure 2. We model the relation classification between entities as a binary classification task. Each sentence is used to produce positive and negative instances for the relations it contains. For example, the sentence `I have two sons John and Jim who are 23 and 26.` will produce two positive instances: *(John, age, 23)* and *(Jim, age, 26)*. Negative instances are produced by pairing up unrelated arguments (e.g. *(John, age, 26)*).

We train a bi-directional LSTM classifier for this task and fine-tune the parameters using the validation set. The model is trained using the following feature representation:

*Word embeddings*: we use pretrained 300 dimensional Glove embeddings (Pennington et al., 2014)

*Position features*: binary flag representing the position of the target entities

*Annotation*: the tag of the token from the entity identification step.

## 5 Results

The system is trained on training data from both sources (AMT and GCS), and is evaluated on the test data. Since the relation classification model relies on the output of the entity identification model, we evaluate it using the automatically detected entities. Table 6 shows the performance of

the tagger and relation classifier on the test set. We further analyze the performance of the relation classifier in two scenarios: i) when the sentence includes information regarding a single person; and ii) when a given sentence includes information about two or more different persons. In the former case, the relation classifier achieves 0.98, 0.97, and 0.98 for precision, recall, and F-score respectively, while in the latter case, it achieves 0.83, 0.91, and 0.87 for the same metrics.

| | Precision | Recall | F-score |
|---|---|---|---|
| Name | 0.833 | 0.882 | 0.857 |
| Gender | 0.865 | 0.884 | 0.874 |
| Age | 0.874 | 0.877 | 0.875 |
| Age at diagnosis | 0.756 | 0.674 | 0.713 |
| Age at death | 0.746 | 0.423 | 0.540 |
| Illness | 0.842 | 0.784 | 0.812 |
| Cancer | 0.889 | 0.856 | 0.872 |
| Cancer type | 0.839 | 0.737 | 0.775 |
| Cause of death | 0.737 | 0.569 | 0.642 |
| Relation | 0.913 | 0.954 | 0.933 |

Table 6: The precision, recall, and F-score of our models for entity and relation classification.

Because the two subsections – AMT and GCS – are slightly different despite being obtained using the same questionnaire (see Table 3), we test whether this difference influences the relation extraction models. We evaluate models trained using training data from one source and tested using data from the other source. A robust model should be able to detect and extract the targeted relations even when they appear in sentences of different structure and complexity. This would be reflected in close results on its own (same as training) or the other subset's test data. Table 7 shows the results in terms of accuracy for the various experimental set-ups. The results reflect the difference between the two subsets: the results on the GCS data fluctuate more (between 0.79 and 0.94 accuracy) when the AMT or the GCS data is used for training, while AMT is rather stable (0.86 – 0.89 accuracy). Using all available training data leads to best results on both test subsets, for both simple and complex sentences.

| Test data | Simple | Complex | All |
|---|---|---|---|
| AMT training data | | | |
| AMT | 0.99 | 0.78 | 0.89 |
| GCS | 0.99 | 0.76 | 0.79 |
| AMT + GCS | 0.99 | 0.77 | 0.88 |
| GCS training data | | | |
| AMT | 0.96 | 0.76 | 0.86 |
| GCS | 0.97 | 0.94 | 0.94 |
| AMT + GCS | 0.96 | 0.82 | 0.87 |
| AMT & GCS training data | | | |
| AMT | 0.99 | 0.90 | 0.95 |
| GCS | 0.98 | 0.94 | 0.95 |
| AMT + GCS | 0.99 | 0.91 | 0.95 |

Table 7: Accuracy of models trained and evaluated on different parts of the dataset. We report the results on simple, complex, and all sentences.

## 6 Conclusions

We have presented a dataset of natural language answers to a questionnaire designed to obtain a patient's medical history. The questionnaire, designed by medical professionals, contains 159 questions. We obtained answers through crowdsourcing and from sessions with a genetic counselor. The data was annotated by medical professionals with attributes and relations relevant to constructing a patient's medical history. The dataset consists of 228 questionnaire answers, covering 20,447 annotations. Relation extraction experiments using a two step system – entity identification and relation classification – show high performance on the task, further confirming the quality and usefulness of the annotations.

To our knowledge, this is the largest dataset of medical family history that has been developed so far. The dataset and the relation extraction system will both be made available to the community, to foster research in extracting relations to construct family medical histories.

The dataset introduced in this paper is publicly available from `http://lit.eecs.umich.edu/downloads.html`.

## Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments and suggestions. We would also like to thank Quincy Davenport and our genetic counseling experts Professor Elena Stoffel, Professor Monica Marvin, Erika Koeppe, and Kara Milliron, who helped us with the construction of this dataset.

## References

Robert Bill, Serguei Pakhomov, Elizabeth S. Chen, Tamara J. Winden, Elizabeth W. Carter, and Genevieve B. Melton. 2014. Automated extraction of family history information from clinical notes. In

*AMIA 2014 Annual Symposium proceedings*, page 17091717.

Jason PC Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Sergey Goryachev, Hyeoneui Kim, and Qing Zeng-Treitler. 2008. Identification and extraction of family history information from clinical reports. In *AMIA 2008 Annual Symposium proceedings*, pages 247–251.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.

Neal Lewis, Daniel Gruhl, and Hui Yang. 2011. Extracting family history diagnosis from clinical texts. In *BICoB*.

Yang Liu, Furu Wei, Sujian Li, Heng Ji, Ming Zhou, and Houfeng Wang. 2015. A Dependency-Based Neural Network for Relation Classification. *CoRR*, abs/1507.04646.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using LSTMs on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1116, Berlin, Germany. Association for Computational Linguistics.

Paul A Nathan, Owen Johnson, Susan Clamp, and Jeremy C Wyatt. 2016. Time to rethink the capture and use of family history in primary care. *Br J Gen Pract*, 66(653):627–628.

Naoaki Okazaki. 2007. Crfsuite: a fast implementation of conditional random fields (crfs).

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Taraka Rama, Pål Brekke, Øystein Nytrø, and Lilja Øvrelid. 2018. Iterative development of family history annotation guidelines using a synthetic corpus of clinical text. In *Proceedings of the Ninth International Workshop on Health Text Mining and Information Analysis, Louhi@EMNLP 2018, Brussels, Belgium, October 31, 2018*, pages 111–121.

Lance Ramshaw and Mitch Marcus. 1995. Text chunking using transformation-based learning. In *Third Workshop on Very Large Corpora*.

Cícero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying Relations by Ranking with Convolutional Neural Networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 626–634.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. brat: a web-based tool for NLP-assisted text annotation. In *Proceedings of the Demonstrations Session at EACL 2012*, Avignon, France. Association for Computational Linguistics.

Rebecca Sutphen, Barbara Davila, Heather Shappell, Tricia Holtje, Susan Vadaparampil, Sue Friedman, Michele Toscano, and Joanne Armstrong. 2010. Real world experience with cancer genetic counseling via telephone. *Familial Cancer*.

Daniel J Wattendorf and Donald W Hadley. 2005. Family history: the three-generation pedigree. *American family physician*, 72(3).

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Suncong Zheng, Feng Wang, Hongyun Bao, Yuexing Hao, Peng Zhou, and Bo Xu. 2017. Joint extraction of entities and relations based on a novel tagging scheme. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1227–1236, Vancouver, Canada. Association for Computational Linguistics.

# Multi-task Learning for Natural Language Generation in Task-Oriented Dialogue

**Chenguang Zhu** **Michael Zeng** **Xuedong Huang**

Microsoft Speech and Dialogue Research Group, Redmond, WA, USA
{chezhu, nzeng, xdh}@microsoft.com

## Abstract

In task-oriented dialogues, Natural Language Generation (NLG) is the final and crucial step to produce user-facing system utterances. The result of NLG is directly related to the perceived quality and usability of a dialogue system. While most existing systems provide semantically correct responses given goals to present, they struggle to match the variation and fluency in the human language. In this paper, we propose a novel multi-task learning framework, NLG-LM, for natural language generation. In addition to generating high-quality responses conveying the required information, it also explicitly targets for naturalness in generated responses via an unconditioned language model. This can significantly improve the learning of style and variation in human language. Empirical results show that this multi-task learning framework outperforms previous models across multiple datasets. For example, it improves the previous best BLEU score on the E2E-NLG dataset by 2.2%, and on the Laptop dataset by 6.1%.

## 1 Introduction

Natural Language Generation (NLG) is the final procedure in the pipeline of task-oriented dialogues. As the result of NLG is directly facing users, its readability and informativeness have a direct impact on users' perception of the entire dialogue system. On one hand, the response must contain the desired information, referred to as *meaning representation* (MR), in order to provide or request a user's information. On the other hand, the system response needs to mimic the fluency and variation in human language to improve the user experience. To this end, there have been numerous studies on methods to generate natural responses for task-driven dialogues.

Early work primarily employ predefined rules or syntax (Cheyer and Guzzoni, 2014; Langkilde and Knight, 1998). Though these frameworks can provide adequate information, their lack of naturalness and variation in language make the response rather rigid. Moreover, these methods usually require non-trivial manual work to create templates, rendering them unscalable across domains.

Recently, corpus-based methods have gained considerable popularity in natural language generation (Wen et al., 2015a,b; Dušek and Jurčíček, 2016). With the increasing availability of rich dialogue task data, corpus-based frameworks design end-to-end trainable systems. With minimum human effort, these methods directly learn the pattern and styles of human responses from the data, while conveying the required task-specific meaning representation information. Furthermore, the booming of deep learning technology in natural language processing increases these models' capacity to generate sophisticated human-like responses. For instance, Dušek and Jurčíček (2016) employs sequence-to-sequence structure and attention mechanism to generate response tokens from the MR sequence. Wen et al. (2015b) uses a semantic control vector integrated into an LSTM to guide the response generation process. Li et al. (2015) uses maximum mutual information as objective function to generate diverse and appropriate responses. Wen et al. (2016) proposes data counterfeiting to reduce the complexity of transferring trained parameters across multiple domains. However, it still remains a challenge in task-oriented dialogue systems to generate truly natural utterance indistinguishable from a human's response.

On the other hand, language modeling is a technique typically employed to learn language patterns from text. It has been successfully used to generate natural and semantically sound utterances for text summarization, speech recognition and other NLP tasks (Roark et al., 2004; Rush



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1261–1266, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

et al., 2015). As task-oriented dialogue datasets usually contain rich human responses, leveraging language modeling has a great potential to boost an NLG model's capacity to mimic human language.

Due to recent successes of multi-task learning in NLP (Collobert et al., 2011; Xu et al., 2018), we propose a multi-task scheme to tackle natural language generation in task-oriented dialogues. For the NLG task, we employ a sequence-to-sequence framework. The decoder uses an attention mechanism to carry over information from the encoder on an MR sequence. Therefore, the NLG task generates response conditioned on input MR.

The primary contribution of our work is to incorporate a language modeling task on human-generated responses as an *unconditioned* complementary process that brings in more language-related elements, without the intervention of required MR information. Furthermore, the unsupervised nature of language modeling means we do not need additional labelled data. Thus, under multi-task learning framework, we simultaneously train the NLG and language modeling tasks. To facilitate multi-task learning, we carry out language modeling task in decoder and it partially shares parameters from the NLG task.

To evaluate the effectiveness of our model, NLG-LM, we conduct evaluation on 5 task-oriented dialogue NLG tasks: E2E-NLG (Novikova et al., 2017), TV, Laptop, Hotel and the Restaurant datasets (Wen et al., 2015b, 2016). NLG-LM achieves new state-of-the-art results on all 5 datasets. For example, it outperforms Slug (Juraska et al., 2018), the best model in E2E-NLG competition, by 2.2% in BLEU score. Ablation studies show that the introduction of language modeling task during training can improve the result by 2.4% in BLEU score on average.

## 2 Problem Description

In task-oriented dialogues, the natural language generation (NLG) process is to produce system utterances as natural language, given system-generated meaning representations from previous steps in the pipeline. Each MR is a slot-value pair, where the slot indicates the category of the information to convey and the value represents the content. For example, *(area, city south)* is a meaning representation and the corresponding utterance should indicate city south as area information.

In addition to meaning representation, *dialogue acts* (DA) are given to differentiate between different types of system actions. Typical examples of dialogue acts include *inform*, *request* and *confirm*. For a given meaning representation, the NLG process should generate different utterances for different dialogue acts. For instance, *confirm* dialogue act usually leads to system response starting with "Let me confirm" or "Correct me if I'm wrong".

In task-oriented dialogues, NLG is framed as a supervised learning problem. Given training data $\{(d_i, r_i, u_i)\}$, where $d_i$ is the dialogue act, $r_i = \{(s_1, v_1), (s_2, v_2), ..., (s_k, v_k)\}$ is the set of meaning representations, and $u_i$ is a sample utterance generated by human labellers, the goal is to generate utterance $u$ given a new pair of dialogue act $d$ and meaning representations $r$.

As certain types of meaning representation contain entities like location names and product types that are usually proper nouns, we use the *delexicalization* technique to replace values with a special slot token, $\langle$slot name$\rangle$, during training and generation. The ultimate response is obtained via a reversal *lexicalization* process to replace slot tokens with their corresponding values.

## 3 Model

### 3.1 The NLG task

We approach the NLG problem using the sequence-to-sequence method (Sutskever et al., 2014). Compared with SC-LSTM (Wen et al., 2015b), this method does not need to create additional one-hot MR vector, and can be much more easily extended across different domains with varying meaning representations.

We first concatenate dialogue act $d$ and meaning representations $r$ as a single input sequence $\mathcal{I}$ with $m$ tokens. The output sequence $\mathcal{O}$ is similarly obtained from the given utterance $u$, with $n$ tokens. Both sequences are delexicalized. We put special sentence tokens $\langle$BOS$\rangle$ and $\langle$EOS$\rangle$ around each sequence.

The goal is to generate output tokens one at a time, given previously predicted tokens and the input sequence. This can be modeled as maximizing the conditional probability distribution:

$$p(w_1, ..., w_n|\mathcal{I}) = \prod_{t=1}^{n} p(w_t|w_1, ..., w_{t-1}; \mathcal{I}) \quad (1)$$

To do this, we employ the encoder-decoder method.

**Encoder.** We train a dictionary $\mathcal{D}$ to map each token to a fixed-length vector of dimension $d$. The input embedding sequence then goes into a layer of bidirectional RNN to produce contextualized embeddings. We use GRU (Cho et al., 2014) as the RNN unit and sum up the forward and backward RNN outputs. The output of the encoder is denoted by $(\boldsymbol{u}_1, ..., \boldsymbol{u}_m) \in \mathbb{R}^{d_h \times m}$, where $d_h$ is the RNN's output dimension and $m$ is its input sequence length.

**Decoder.** The decoder employs an RNN with an attention mechanism to generate tokens one at a time. It starts with the beginning-of-sentence token and uses the final hidden state from encoder RNN as the initial hidden state. In the $t$-th step, we use the same dictionary $\mathcal{D}$ from the encoder to map the $t$-th output token into vector $\boldsymbol{s}_t$ and apply dropout. Then, given the previous hidden state $\boldsymbol{h}_{t-1}$, the decoder first computes attention weights over encoder outputs:

$$\boldsymbol{v}_i = [\boldsymbol{h}_{t-1}; \boldsymbol{u}_i] \in \mathbb{R}^{2d_h} \tag{2}$$

$$\boldsymbol{e}_i = \text{softmax}(\boldsymbol{W}_1 \boldsymbol{v}_i) \in \mathbb{R}^{d_h} \tag{3}$$

$$\alpha_i = \text{ReLU}(\boldsymbol{b}^T \boldsymbol{e}_i) \in \mathbb{R} \tag{4}$$

Here, $\boldsymbol{W}_1 \in \mathbb{R}^{d_h \times 2d_h}$ and $\boldsymbol{b} \in \mathbb{R}^{d_h}$ are parameters. The weights $\{\alpha_i\}_{i=1}^m$ are then applied to encoder outputs to obtain the context vector $\boldsymbol{c} = \sum_{1 \le i \le m} \alpha_i \boldsymbol{u}_i$.

The context vector $\boldsymbol{c}$ and embedded vector $\boldsymbol{s}_t$ are then concatenated and sent into decoder GRU with output $\boldsymbol{g} \in \mathbb{R}^{d_h}$ and a new hidden state $\boldsymbol{h}_t \in \mathbb{R}^{d_h}$.

To generate the next token, we reuse the dictionary $\mathcal{D}$ with its transposed weights $\boldsymbol{W}_{\mathcal{D}}$[1]. We again integrate the context vector to fuse in contextual information:

$$\boldsymbol{o} = \boldsymbol{W}_2[\boldsymbol{g}; \boldsymbol{c}] \in \mathbb{R}^d \tag{5}$$

$$\boldsymbol{p}_t = \boldsymbol{W}_{\mathcal{D}} \boldsymbol{o} \in \mathbb{R}^{|V|} \tag{6}$$

where $\boldsymbol{W}_2 \in \mathbb{R}^{d \times 2d_h}$ is a parametrized matrix. $\boldsymbol{p}_t$ is the probability distribution of the next token over all tokens in dictionary.

The loss function is cross entropy. Suppose the one-hot vector for the ground-truth at $t$-th step is $\boldsymbol{y}_t$, then the loss function for each training sample sequence pair is:

$$\mathcal{L}^{NLG}(\theta) = -\sum_{t=1}^{n} \boldsymbol{y}_t^T \log(\boldsymbol{p}_t) \tag{7}$$

### 3.2 Coupling with Language Model

The encoder-decoder approach above incorporates the information from dialogue act and meaning representation at each step via attention. However, due to this mechanism, the generated utterance inevitably relies to a great extent on the input sequence, focusing less on the fluency and variation of human language, which is as important as conveying the required information in task-oriented dialogues.

On the other hand, language modeling is typically used to characterize the naturalness of words, phrases and sentences. A well-trained language model can assign natural and semantically sound utterance higher scores than rigid and unnatural sentences. In deep learning, the language model task is often solved by a recurrent neural network. However, instead of depending on an input sequence like in Equation (1), the probability of the next token in language models only relies on preceding words:

$$p^{LM}(w_1, ..., w_n) = \prod_{t=1}^{n} p^{LM}(w_t | w_1, ..., w_{t-1}) \tag{8}$$

We propose that by integrating language modelling into the NLG process as an additional objective, the generated sentences will better approximate the styles and variation in human response.

To do this, we add another GRU unit, $\text{GRU}^{LM}$, to the decoder, that has its own hidden state $\boldsymbol{h}_{t-1}^{LM}$ and takes the embedded vector $\boldsymbol{s}_t$ as input. The output is $\boldsymbol{g}^{LM}$ and the new hidden state is $\boldsymbol{h}_t^{LM}$. The probability distribution of next token in language model is:

$$\boldsymbol{o}^{LM} = \boldsymbol{W}_2[: d_h] \boldsymbol{g}^{LM} \in \mathbb{R}^d \tag{9}$$

$$\boldsymbol{p}_t^{LM} = \boldsymbol{W}_{\mathcal{D}} \boldsymbol{o}^{LM} \in \mathbb{R}^{|V|} \tag{10}$$

where $\boldsymbol{W}_2[: d_h]$ are the first $d_h$ columns of $\boldsymbol{W}_2$. As we can see, the context $\boldsymbol{c}$ does not affect the probability computation for language modeling. The loss function of language model is:

$$\mathcal{L}^{LM}(\theta) = -\sum_{t=1}^{n} \boldsymbol{y}_t^T \log(\boldsymbol{p}_t^{LM}) \tag{11}$$

[1] Suppose $\mathcal{D} \in \mathbb{R}^{|V| \times d}$, its transposed weights $\boldsymbol{W}_{\mathcal{D}} \in \mathbb{R}^{d \times |V|}$.

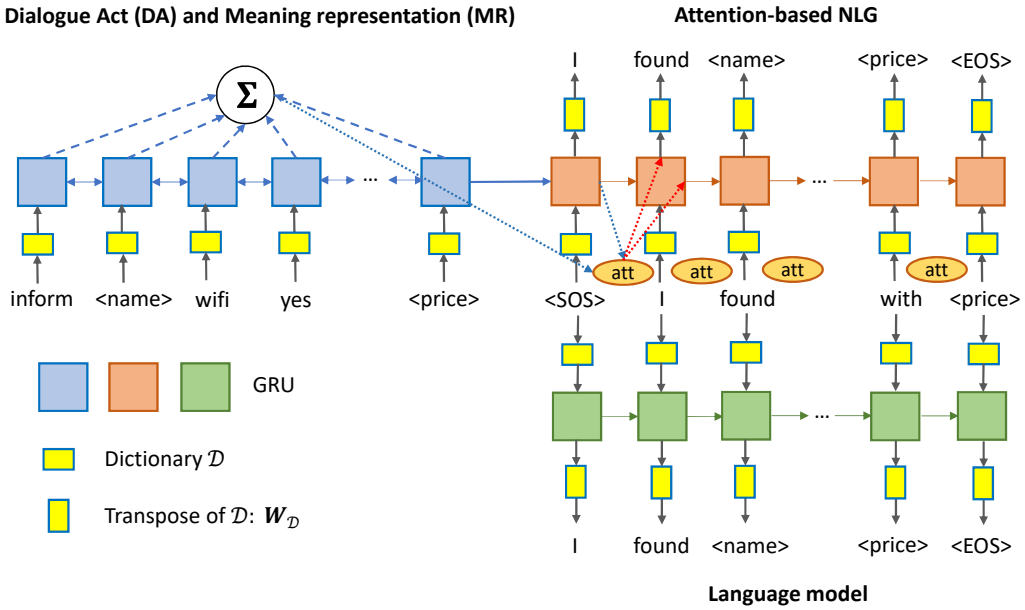


Figure 1: NLG-LM model structure. The encoder takes dialogue act (DA) and meaning representations (MR) as input. The decoder for NLG applies attention on encoder outputs, and the decoder for language modeling shares dictionary with NLG but uses a separate GRU unit.

| Model | E | TV | L | H | R |
|---|---|---|---|---|---|
| TGen | 0.659 | / | / | / | / |
| Slug | 0.662 | 0.529 | 0.524 | / | / |
| SCLSTM | / | 0.527 | 0.512 | 0.848 | 0.752 |
| RALSTM | / | 0.541 | 0.525 | 0.898 | 0.779 |
| w/o LM | 0.673 | 0.609 | 0.560 | 0.927 | 0.734 |
| NLG-LM | **0.684** | **0.617** | **0.586** | **0.939** | **0.795** |

Table 1: BLEU scores on E2E-NLG (**E**), TV, Laptop (**L**), Hotel (**H**) and Restaurant (**R**) *testset*. w/o LM is our model without language model task.

| Model | TGen | Slug | w/o LM | NLG-LM |
|---|---|---|---|---|
| NIST | 8.609 | 8.613 | 8.581 | **8.626** |

Table 2: NIST scores on E2E-NLG *testset*.

Finally, we linearly combine the two loss functions into a single multi-task loss function:

$$\mathcal{L}(\theta) = \mathcal{L}^{NLG}(\theta) + \alpha \mathcal{L}^{LM}(\theta) \qquad (12)$$

We depict our model structure in Figure 1. As shown, the dictionary $\mathcal{D}$ is shared between the NLG task and language modeling task.

## 4 Experiments

### 4.1 Datasets and settings

We evaluated the models on five datasets from different domains, covering restaurant booking, hotel booking and retail. The largest dataset is from the E2E-NLG task (Novikova et al., 2017), consisting of 51.2K MR-utterance pairs in the restaurant domain. We also use the four datasets from RNN-LG (Wen et al., 2016), including dialogue scenarios in TV retails, laptop retails, hotel-booking and restaurant-booking domains, with 14.1K, 26.5K, 8.7K and 8.5K samples respectively.

For fairness, we use the official evaluation scripts from E2E-NLG and RNN-LG (Wen et al., 2016) to assess models. We use the BLEU-4 (Papineni et al., 2002) and NIST (Przybocki et al., 2009) metrics.

**Delexicalization.** In the experiment, we do not delexicalize slots that have binary values or are inappropriate for verbatim substitution. For instance, in E2E-NLG datasets, we only delexicalize *name* and *near* slots. For TV dataset, we delexicalize all slots except *hasusbport*. In Laptop dataset, we delexicalize all slots except *isforbusinesscomputing* and *request*. In Hotel dataset, we delexicalize all slots except *acceptscreditcards*, *dogsallowed* and *hasinternet*. In Restaurant dataset, we delexicalize all slots except *kidsallowed* and *request*.

| | E2E-NLG | TV | Laptop | Hotel | Restaurant |
|---|---|---|---|---|---|
| Dropout rate | 0.4 | 0.2 | 0.2 | 0.3 | 0.3 |
| Learning rate | 0.005 | 0.001 | 0.001 | 0.005 | 0.001 |
| Batch size | 20 | 20 | 20 | 20 | 20 |
| Dictionary dimension | 100 | 50 | 50 | 200 | 50 |
| RNN hidden size | 512 | 128 | 512 | 256 | 512 |

Table 3: Hyperparameters of NLG-LM in experiments.

**Baseline.** Our baseline models include TGen (Dušek and Jurčíček, 2016), SC-LSTM (Wen et al., 2015b), RALSTM (Tran and Nguyen, 2017) and Slug (Juraska et al., 2018).

**Training details.** We use Adamax (Kingma and Ba, 2014) as the optimizer. We use teacher forcing, which means that during training, the decoder is always presented with the previous ground-truth token. The language modeling task is only used during training, and it uses utterances from the same batch as NLG task. The inference uses a beam search of width 10. We use the multi-task coefficient $\alpha = 0.5$ in all experiments. The hyper-parameters were chosen on the dev set with early stopping, as shown in Table 3.

### 4.2 Result

We present our experimental results in Table 1 and 2. As shown, our model, NLG-LM, outperforms the baseline models in all 5 datasets. In E2E-NLG dataset, it achieves $2.2\%$ higher BLEU score and 0.013 higher NIST score than Slug. In TV, Laptop, Hotel and Restaurant datasets, NLG-LM greatly improves previously best result by $7.6\%$, $6.1\%$, $4.1\%$, and $1.6\%$, respectively. We also ran our model without the language modeling task as an ablation study, denoted by *w/o LM*. As seen, language modeling can improve the result by $0.8\%$ to $6.1\%$, or on average $2.4\%$, which demonstrates the effectiveness of multi-task learning.

In the appendix, we examined some predicted samples generated from our model, which shows that the addition of the language model makes the generated responses more natural and variable.

**Efficiency.** We compared the training time of NLG-LM with that of w/o LM. As shown in Table 4, the additional language model only introduces $23.6\%$ more training time, since it does not involve expensive attention computation. Therefore, NLG-LM can offer more natural response generation with comparable efficiency.

| | NLG-LM | w/o LM |
|---|---|---|
| Per-batch | 618.87s | 500.64s |

Table 4: Running time per batch of NLG-LM and w/o LM on E2ENLG training set. The machine has an Intel Xeon CPU E5 and a Tesla v100 GPU.

## 5 Conclusions

In this paper, we propose a novel multi-task learning method, NLG-LM. It incorporates a language model task into the response generation process as an unconditioned complementary process to boost the naturalness of generated utterances. We fit both tasks into a sequence-to-sequence structure under a multi-task learning scheme. Empirical results show that NLG-LM significantly outperforms previous methods in 5 large-scale datasets with reasonable computational efficiency. Ablation studies show the effectiveness of using the language modeling task within a multi-task scheme.

## Acknowledgement

We thank the anonymous reviewers for the insightful comments. We thank William Hinthorn for proof-reading our paper.

## References

Adam Cheyer and Didier Guzzoni. 2014. Method and apparatus for building an intelligent automated assistant. US Patent 8,677,377.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from

scratch. *Journal of machine learning research*, 12(Aug):2493–2537.

Ondřej Dušek and Filip Jurčíček. 2016. Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings. *arXiv preprint arXiv:1606.05491*.

Juraj Juraska, Panagiotis Karagiannis, Kevin K Bowden, and Marilyn A Walker. 2018. A deep ensemble model with slot alignment for sequence-to-sequence natural language generation. *arXiv preprint arXiv:1805.06553*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Irene Langkilde and Kevin Knight. 1998. Generation that exploits corpus-based statistical knowledge. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*, pages 704–710. Association for Computational Linguistics.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2015. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055*.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017. The e2e dataset: New challenges for end-to-end generation. *arXiv preprint arXiv:1706.09254*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Mark Przybocki, Kay Peterson, Sébastien Bronsart, and Gregory Sanders. 2009. The nist 2008 metrics for machine translation challengeoverview, methodology, metrics, and results. *Machine Translation*, 23(2-3):71–103.

Brian Roark, Murat Saraclar, and Michael Collins. 2004. Corrective language modeling for large vocabulary asr with the perceptron algorithm. In *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages I–749. IEEE.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Van-Khanh Tran and Le-Minh Nguyen. 2017. Natural language generation for spoken dialogue system using rnn encoder-decoder networks. *arXiv preprint arXiv:1706.00139*.

Tsung-Hsien Wen, Milica Gasic, Dongho Kim, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. 2015a. Stochastic language generation in dialogue using recurrent neural networks with convolutional sentence reranking. *arXiv preprint arXiv:1508.01755*.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Lina M Rojas-Barahona, Pei-Hao Su, David Vandyke, and Steve Young. 2016. Multi-domain neural network language generation for spoken dialogue systems. *arXiv preprint arXiv:1603.01232*.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. 2015b. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745*.

Yichong Xu, Xiaodong Liu, Yelong Shen, Jingjing Liu, and Jianfeng Gao. 2018. Multi-task learning for machine reading comprehension. *arXiv preprint arXiv:1809.06963*.

# Dirichlet Latent Variable Hierarchical Recurrent Encoder-Decoder in Dialogue Generation

**Min Zeng  Yisen Wang†  Yuan Luo**
Department of Computer Science and Engineering
Shanghai Jiao Tong University
eewangyisen@gmail.com; {min_zeng, yuanluo}@sjtu.edu.cn

## Abstract

Variational encoder-decoders have achieved well-recognized performance in the dialogue generation task. Existing works simply assume the Gaussian priors of the latent variable, which are incapable of representing complex latent variables effectively. To address the issues, we propose to use the Dirichlet distribution with flexible structures to characterize the latent variables in place of the traditional Gaussian distribution, called Dirichlet Latent Variable Hierarchical Recurrent Encoder-Decoder model (Dir-VHRED). Based on which, we further find that there is redundancy among the dimensions of latent variable, and the lengths and sentence patterns of the responses can be strongly correlated to each dimension of the latent variable. Therefore, controllable responses can be generated through specifying the value of each dimension of the latent variable. Experimental results on benchmarks show that our proposed Dir-VHRED yields substantial improvements on negative log-likelihood, word-embedding-based and human evaluations.

## 1 Introduction

Recurrent neural networks (RNNs) (Bengio et al., 2003) have achieved great success on many natural language processing tasks. For the dialogue generation task, a RNN-based Hierarchical Recurrent Encoder-Decoder (HRED) was first proposed in (Serban et al., 2016). It consists of three RNNs: encoder RNN, context RNN and decoder RNN. Firstly, the encoder RNN encodes each utterance into a fixed-size real-valued vector through word embedding. Then, the last hidden state of encoder RNN is feed into the context RNN to summarize the dialogue information. Finally, the decoder RNN takes the last state of context RNN as input and produces the probability distribution of the word in the next utterance.

Although it outperforms n-gram and other neural network language models, HRED only produces one word at a time, which is unable to fully grasp the holistic high-level syntactic properties (*e.g.*, topics, tones or sentiment) (Bowman et al., 2015). When the sentence grows longer, it has the drawback of tending to generate short and generic responses (Vinyals and Le, 2015). Thus, Serban et al. (2017) proposed the Variational Hierarchical Recurrent Encoder-Decoder (VHRED) by combining HRED with Variational Autoencoders (VAEs) (Kingma and Welling, 2013) that introduced a latent variable to characterize the sentence-level representation for learning holistic properties. However, VHRED imposes a symmetric distribution (*i.e.*, Gaussian distribution) to the latent variable, which, though facilitating analyzing, are incapable of representing complex latent variables effectively. Therefore, in order to generate more meaningful and expressive responses, a more flexible and tractable prior distribution of the latent variable is needed.

In this paper, we propose to use the Dirichlet distribution to characterize the latent variable in VHRED, named Dir-VHRED. Dirichlet distribution is a popular conjugate prior for Multinomial distributions in Bayesian statistics. It can be concave or convex, monotonously rising or decreasing, symmetrical or asymmetrical, which makes it more flexible for better capturing the sentence-level properties. Our main contributions are summarized as follows:

- We introduce Dirichlet distribution to VAE-based dialogue generation model and propose the Dir-VHRED model for better grasping the sentence-level properties and generating more meaningful and expressive responses.

†Corresponding author: Yisen Wang.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1267–1272, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

- We find that the lengths and sentence patterns of the responses can be strongly correlated to each dimension of the latent variable.
- Experiments on three kinds of evaluation metrics demonstrate the superiority of our proposed Dir-VHRED model.

## 2 Preliminaries

### 2.1 Variational Autoencoders

The key idea of Variational autoencoders (VAEs) is to reconstruct the input $x$ through the latent variable $z$ (Kingma and Welling, 2013). As the log-likelihood $\log p_\theta(x)$ is intractable, its lower bound *Evidence Lower BOund* (ELBO), denoted as $\mathcal{L}(\theta, \phi; x)$, is involved to make the maximization tractable:

$$\begin{aligned}\log p_\theta(x) &\geq \mathcal{L}(\theta, \phi; x) \\ &= -KL(q_\phi(z|x)||p(z)) \\ &\quad + \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)],\end{aligned} \tag{1}$$

where $p(z)$ denotes the prior distribution of $z$, and $q_\phi(z|x)$ is used for approximating the intractable true posterior distribution $p_\theta(z|x)$. Note that the total loss is the negative of ELBO.

### 2.2 Dialogue Model

Assuming dialogue $\mathcal{D}$ consisting of $N$ utterances, the VAE-based dialogue model generates the responses by utilizing the latent variable $z$. The generation of the next utterance is defined by:

$$\begin{aligned}p_\theta(z_n|W_{<n}) &= \mathcal{N}(\mu_{prior(W_{<n})}, \Sigma_{prior(W_{<n})}), \\ p_\theta(W_n|z_n, W_{<n}) &= \prod_m p_\theta(w_{n,m}|z_n, w_{n,<m}, W_{<n}),\end{aligned} \tag{2}$$

where $W_n$ is the $n$-th utterance of the dialogue and $w_{n,m}$ indicates the $m$-th word of the $n$-th utterance. As the log-likelihood is intractable, instead of pursuing the exact maximum of the log-likelihood, we maximize its lower bound ELBO:

$$\begin{aligned}&\log p_\theta(W_1, W_2, \cdots, W_N) \geq \\ &\sum_{n=1}^{N} -KL[q_\phi(z_n|W_1, W_2, \cdots, W_N)||p_\theta(z_n|W_{<n})] \\ &\quad + \mathbb{E}_{q_\phi(z_n|W_1, W_2, \cdots, W_N)}[\log(p_\theta(W_N|z_n, W_{<n}))],\end{aligned} \tag{3}$$

where $p_\theta(z_n|W_{<n})$ denotes the prior distribution of $z_n$, and $q_\phi(z_n|W_1, W_2, \cdots, W_N)$ is used for approximating the intractable true posterior distribution $p_\theta(z_n|W_1, W_2, \cdots, W_N)$.

### 2.3 Dirichlet Distribution

Dirichlet distribution, a family member of continuous multivariate probability distribution, is regarded as a multivariate generalization of the Beta distribution. In case of the Dirichlet distribution, it is a conjugate prior for the Multinomial distribution. The probability density function of Dirichlet distribution is given by:

$$f(x_1, \cdots, x_K; \alpha_1, \cdots, \alpha_K) = \frac{1}{B(\alpha)} \prod_{i=1}^{K} x_i^{\alpha_i - 1}, \tag{4}$$

where parameter $\alpha_i > 0$ is a $K$ dimension vector and $B(\alpha)$ is a Beta function:

$$B(\alpha) = \frac{\prod_{i=1}^{K} \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{K} \alpha_i)}, \quad \alpha = (\alpha_1, \cdots, \alpha_K), \tag{5}$$

When $\alpha_1 = \alpha_2 = \cdots = \alpha_K$, Dirichlet distribution is a symmetric distribution. Particularly, when all the factors in $\alpha$ equal 1, Dirichlet distribution becomes a uniform distribution.

## 3 The Proposed Dir-VHRED Model

Mathematically, Dirichlet distribution owns a flexible structure. With different settings of parameter $\alpha$, Dirichlet distribution may have various forms, which can be concave or convex, monotonously rising or decreasing, symmetrical or asymmetrical. Therefore, equipped with the diverse structure, Dirichlet distribution is capable of modeling the complex latent variable. Furthermore, Dirichlet distribution has paved its way in many natural language processing tasks like topic model, text classification and so on. For example, it was introduced in the Latent Dirichlet Allocation (LDA) (Blei et al., 2003) to model the topic distribution and word distribution, and experimentally presents a well-recognized performance.

Motivated by the above facts, we thus introduce Dirichlet distribution to model the latent variable $z$ in order to better grasp the sentence-level properties which further encourages the better responses. With the distribution of $z$ changing from Gaussian to Dirichlet, we may face a tough but

important problem: the reparameterization of $z$. The variables with any distribution can be generated through the transformation of the samples from standard Uniform distribution if the corresponding inverse Cumulative Distribution Function (CDF) is known. For the Gaussian latent variable, it can make use of the inverse CDF to do the reparameterization (Kingma and Welling, 2013). While for the Dirichlet latent variable, unlike the known CDF of Gaussian distribution, the CDF of Dirichlet distribution is too complex to obtain. Inspired from (Jankowiak and Obermeyer, 2018), we use the reject sampling to solve the Dirichlet reparameterization problem. Furthermore, to mitigate KL-vanishing problem, in contrast to the static weight scheme in (Bowman et al., 2015), the imposed weight on KL-divergence in our model is dynamic. Since the KL-divergence and reconstruction loss antagonize each other, by setting the weight as the reciprocal of the reconstruction loss, the KL and reconstruction loss enable to keep a dynamic balance, which leads to a more stable KL-divergence.

Following the techniques described above, ELBO then has the form:

$$\begin{aligned}\log p_\theta(x) &\geq \mathcal{L}(\theta, \phi; x) \\ &= -\lambda KL(q_\phi(z|x)||p(z)) \\ &\quad + \mathbb{E}_{q_\phi(z|x)}[\log(p_\theta(x|z))],\end{aligned} \tag{6}$$

where

$$\lambda = -1/\mathbb{E}_{q_\phi(z|x)}[\log(p_\theta(x|z))], \tag{7}$$

and the KL-divergence term can be derived as*:

$$\begin{aligned}&KL(q_\phi(z|x)||p(z)) = \\ &\log\Gamma(\sum_{k=1}^{K}\alpha_k) - \sum_{k=1}^{K}\log\Gamma(\alpha_k) - \log\Gamma(\sum_{k=1}^{K}\beta_k) \\ &+ \sum_{k=1}^{K}\log\Gamma(\beta_k) + \sum_{k=1}^{K}(\alpha_k - \beta_k)(\psi(\alpha_k) - \psi(\sum_{k=1}^{K}\alpha_k)),\end{aligned} \tag{8}$$

where $\alpha$, $\beta$ are parameters of Dirichlet distribution $q_\phi(z|x)$ and $p(z)$ respectively, $K$ is the dimension of $z$, and $\psi$ is the Digamma function.

# 4 Experiments

## 4.1 Datasets

All experiments are conducted on the following two dialogue datasets:

**Ubuntu Dialogue Corpus (Ubuntu)** (Lowe et al., 2015): Ubuntu Dialogue Corpus is a large dataset for research in unstructured multi-turn dialogue systems. It contains 1 million two-person conversation and the conversation consists of average 8 turns.

**Cornell Movie Dialogs Corpus (Movie)** (Danescu-Niculescu-Mizil and Lee, 2011): Cornell Movie Dialogs Corpus contains a large metadata-rich collection of fictional conversations extracted from raw movie scripts. It contains 220,579 conversations from 617 movies.

## 4.2 Experimental Setup

We conduct a series of experiments to compare our proposed Dir-VHRED with two baselines: HRED (Serban et al., 2016) and VHRED (Serban et al., 2017). All the RNN modules (*i.e.*, the encoder, context and decoder RNNs) adopt a single-layer GRU (Cho et al., 2014) with 1000 hidden units. The word drop rate is set to 0.25, the dimensionality of latent variable $z$ is 3, and the word embedding size is 200†.

For VHRED, the weight parameter $\lambda$ of KL-divergence is initialized as 0 and gradually increased to 1 at the $20,000$-th/$100,000$-th training steps for Movie/Ubuntu datasets. While, for Dir-VHRED, the parameter $\lambda$ is adaptively determined by Eq. (7).

For the dataset, we truncate utterances longer than 30 words and split the train/validation/test sets by 0.8/0.1/0.1 respectively.

All models are trained by Adam optimizer (Kingma and Ba, 2014) with batch size 40 and learning rate $1 \times 10^{-4}$. At the training time, we stop the training when the loss on the validation set does not decrease within 5 epochs. At the evaluating time, beam search (Wiseman and Rush, 2016) with beam size 5 is used for generating output responses. Both the single response (1-turn) and the three consecutive responses (3-turn) are evaluated for each model. Our code is available at `https://github.com/cloversjtu/dir-vhred`.

## 4.3 Performance Evaluation

In order to comprehensively evaluate the model performance on dialogue generation, we adopt the following three evaluation metrics: 1) negative

*Detailed derivation can be found in Appendix A.1.

†The word embedding is obtained from Word2Vec embeddings trained on the Google News Corpus.

log-likelihood metric to measure the loss of the generating procedure; 2) word-embedding metric to measure the cosine distance between the generated responses and the ground truth responses; and 3) human evaluation.

#### 4.3.1 Negative Log-likelihood Evaluation

Table 1 reports the comparisons of the per-word negative log-likelihood (NLL) composition of different models. NLL consists of the reconstruction loss (*reconstruction*) and the KL-divergence (*KL-div.*). KL-divergence indicates the information encoded in $z$, and reconstruction loss represents the loss of reconstructing $x$ through $z$. As can be seen from Table 1, with almost the same KL-divergence, Dir-VHRED achieves the lowest NLL on both datasets, implying better performance compared with VHRED. Dir-VHRED also has the lower reconstruction loss, which indicates that Dirichlet prior is better than the Gaussian prior for reconstructing the responses.

Table 1: Results of NLL on Ubuntu and Movie datasets with different models. The $\leq$ symbol denotes the variational bound.

| Ubuntu dataset | | | |
|---|---|---|---|
| Model | NLL | *reconstruction* | *KL-div.* |
| HRED | 3.844 | - | - |
| VHRED | $\leq$ 4.132 | 3.765 | 0.367 |
| Dir-VHRED | $\leq$ 3.999 | 3.614 | 0.385 |
| Movie dataset | | | |
| Model | NLL | *reconstruction* | *KL-div.* |
| HRED | 3.944 | - | - |
| VHRED | $\leq$ 4.233 | 3.904 | 0.330 |
| Dir-VHRED | $\leq$ 4.073 | 3.741 | 0.332 |

#### 4.3.2 Word-embedding-based Evaluation

Word-embedding metric is designed to measure the similarity between words, and can be divided into three categories: *average* (Foltz et al., 1998), *greedy* (Rus and Lintean, 2012) and *extrema* (Forgues et al., 2014). The *average* metric calculates sentence-level embeddings, while the *greedy* and *extrema* ones compute the word-to-word cosine similarity. Their difference lies on that *greedy* takes the average word vector in the sentence as the sentence embedding while *extrema* adopts the extreme value of these word vectors. Tables 2 and 3 demonstrate the results of word-embedding metric on 1-turn and 3-turn responses from different models. Embedding metrics first map the generating responses to a vector space and then compute the cosine similarly with corresponding ground responses. Therefore, it can be used to measure the sentence-level semantic similarity to a certain extent. Dir-VHRED achieves the best performance among the three word-embedding metrics on 1-turn and 3-turn responses of the two datasets. That is, the responses generated by Dir-VHRED own the highest topic similarity with the ground truth.

Table 2: Results of three embedding metrics on Ubuntu dataset with 1-turn and 3-turn responses.

| Model | *Average* | *Greedy* | *Extrema* |
|---|---|---|---|
| 1-turn | | | |
| HRED | 0.448 | 0.297 | 0.271 |
| VHRED | 0.506 | 0.355 | 0.286 |
| Dir-VHRED | **0.538** | **0.387** | **0.303** |
| 3-turn | | | |
| HRED | 0.426 | 0.275 | 0.260 |
| VHRED | 0.524 | 0.366 | 0.292 |
| Dir-VHRED | **0.552** | **0.393** | **0.313** |

Table 3: Results of three embedding metrics on Movie dataset with 1-turn and 3-turn responses.

| Model | *Average* | *Greedy* | *Extrema* |
|---|---|---|---|
| 1-turn | | | |
| HRED | 0.461 | 0.316 | 0.329 |
| VHRED | 0.519 | 0.376 | 0.356 |
| Dir-VHRED | **0.554** | **0.409** | **0.358** |
| 3-turn | | | |
| HRED | 0.483 | 0.334 | 0.337 |
| VHRED | 0.530 | 0.382 | 0.356 |
| Dir-VHRED | **0.566** | **0.412** | **0.363** |

#### 4.3.3 Human Evaluation

Moreover, we invite 7 qualified volunteers to do the human evaluation on 700 randomly sampled dialogues from Ubuntu dataset. Volunteers are required to rate the generated responses from different models with scores from 1 to 3 (3 is the best), in term of the sentence coherence and contextual

Table 4: Responses generated by different models.

| |
|---|
| **# 1 Input sentence:** |
| that is a upstream bug and red hat marked it <unk>. |
| **# 1 Generated response:** |
| **HRED:** i know it's not a bug, but i'll try it out. |
| **VHRED:** can you provide a link for that? |
| **Dir-VHRED:** so i should report it to the bug report? i don't see any bug reports. |
| **# 2 Input sentence:** |
| i tried but still its not workin. |
| **# 2 Generated response:** |
| **HRED:** what kind of laptop? |
| **VHRED:** is it a laptop? or the fn key is not set? |
| **Dir-VHRED:** what is the model of your laptop? what model? |
| **# 3 Input sentence:** |
| i tried that and even rebooting the pc. |
| **# 3 Generated response:** |
| **HRED:** what does ifconfig say? |
| **VHRED:** check your etc network interfaces. |
| **Dir-VHRED:** what's the error message? |

topic. The average human evaluation scores for HRED, VHRED and Dir-VHRED are 2.50, 2.65 and 2.70 respectively. We also conduct Friedman test (Friedman, 1940) on the human evaluation score to check its statistical significance, and the value of Chi-square test statistics is 5696.25 which indicates that the P-value is less than 0.05. Thus, Dir-VHRED is significantly better than others.

Table 4 lists some examples of the generated sentences of different models. It can be seen that the responses generated by Dir-VHRED best match the context given the input sentence, compared with other models. Dir-VHRED experimentally demonstrates to be able to generate fluent and diverse responses which match the human daily speaking style smoothly.

### 4.4 Interpretability of Dir-VHRED

In our above experiments, we find an interesting phenomenon for Dir-VHRED that there are many non-active dimensions of $z$ and the number of active ones is almost always around 3. Therefore, we conduct additional ablation study experiments to analyze each dimension of $z$. To be specific, we circularly set one dimension of $z$ to 1 and the others to 0, like "100", "010" and "001".

We demonstrate some generated responses through ablation study on each dimension of $z$ on Ubuntu and Movie datasets in Table 5. We can observe that the **dimension 1** (*dim1*) is inclined to generate the shortest responses which, to a certain extent, shows the attitude of the speaker. Specifically, it prefers to generate the responses like "thanks", "no" or "yes". The **dimension 2** (*dim2*) usually generates longer and informative responses. For example on Movie dataset, the generated response is prone to end with "are you?" or "aren't you?", which shows a preference for rhetorical question. Other examples include "you're not going to be here, are you?" and "well, you're a good man, aren't you?". Intuitively, since the Movie dataset is a collection of conversations extracted from movie scripts, the generated responses are closer to daily dialogues. While for Ubuntu, the responses do not have strong individual emotion and rhetorical question rarely appear, as the conversations in this dataset are professional dialogue about Ubuntu system knowledge. The **dimension 3** (*dim3*) focuses on generating the longest and most comprehensive responses that are usually declarative sentences and are highest close to the dialogue topic.

Therefore, we can easily generate the responses with the preferred style and length through changing the value of each dimension of $z$.

Table 5: Responses generated by ablation study on different dimension of $z$ in Dir-VHRED.

| |
|---|
| **# 1 Input sentence:** |
| does grub stay into hdd after installing another os? |
| **# 1 Generated response:** |
| **dim1:** yes. |
| **dim2:** yes, it will be in the mbr of the drive. |
| **dim3:** yes, it will overwrite grub, but it will overwrite the mbr, and then you can use the grub menu to boot the os. |
| **# 2 Input sentence:** |
| that leads me to the next question lol ... i do not know how to create new volumes exactly in cmdline, usually i use a gui. |
| **# 2 Generated response:** |
| **dim1:** gksudo. |
| **dim2:** alt f2 gksudo nautilus and gksudo nautilus will work. |
| **dim3:** <unk> is a gui app for managing users, you can use gksudo nautilus and you can use gksudo nautilus toget a root terminal and you can use. |
| **# 3 Input sentence:** |
| i see this use at your own risk. http askubuntu.com questions <unk> ubuntu 12 04 nvidia gtx 460 video card installation. |
| **# 3 Generated response:** |
| **dim1:** thanks. |
| **dim2:** thanks, i 'll give that a try. thanks ! |
| **dim3:** i have the nvidia drivers installed, but i can't see the nvidia logo on the screen. i have to install the drivers for the graphics card. |
| **# 4 Input sentence:** |
| she isn't my cousin. |
| **# 4 Generated response:** |
| **dim1:** yes. |
| **dim2:** she 's a nurse, isn't she ? |
| **dim3:** yes, she 's a good friend of mine, she 's a nurse. |

## 5 Conclusion

In this paper, we proposed to use Dirichlet distribution in place of traditional Gaussian distribution in VHRED for dialogue generation, called Dir-VHRED, to well capture the sentence-level properties. We also provided a new way on the setting of the weight of KL-divergence to alleviate KL-vanishing problem. Moreover, we found that the lengths and sentence patterns of the generated responses are correlated to the value of each dimension of the latent variable, which can be used for generating the required responses. Experiments on benchmark datasets show the superior of our proposed Dir-VHRED model.

## Acknowledgments

We would like to thank Junqing Gu and Xiaoyu Shen for their constructive discussions. We also appreciate the code shared in (Park et al., 2018).

## References


Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3(Feb):1137–1155.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Cristian Danescu-Niculescu-Mizil and Lillian Lee. 2011. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In *the 2nd Workshop on Cognitive Modeling and Computational Linguistics on ACL*.

Peter W Foltz, Walter Kintsch, and Thomas K Landauer. 1998. The measurement of textual coherence with latent semantic analysis. *Discourse Processes*, 25(2-3):285–307.

Gabriel Forgues, Joelle Pineau, Jean-Marie Larchevêque, and Réal Tremblay. 2014. Bootstrapping dialog systems with word embeddings. In *Modern Machine Learning and Natural Language Processing workshop on NeurIPS*.

Milton Friedman. 1940. A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, 11(1):86–92.

Martin Jankowiak and Fritz Obermeyer. 2018. Pathwise derivatives beyond the reparameterization trick. *arXiv preprint arXiv:1806.01851*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. *arXiv preprint arXiv:1506.08909*.

Yookoon Park, Jaemin Cho, and Gunhee Kim. 2018. A hierarchical latent structure for variational conversation modeling. *arXiv preprint arXiv:1804.03424*.

Vasile Rus and Mihai Lintean. 2012. A comparison of greedy and optimal assessment of natural language student input using word-to-word similarity metrics. In *the 7th Workshop on Building Educational Applications Using NLP on ACL*.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Sam Wiseman and Alexander M Rush. 2016. Sequence-to-sequence learning as beam-search optimization. *arXiv preprint arXiv:1606.02960*.

# Semi-Supervised Bootstrapping of Dialogue State Trackers for Task-Oriented Modelling

**Bo-Hsiang Tseng**[1], **Marek Rei**[2,4], **Paweł Budzianowski**[1,3,5]
**Richard E. Turner**[1], **Bill Byrne**[1], **Anna Korhonen**[3]
[1]Engineering Department, Cambridge University, UK
[2]Computer Laboratory, Cambridge University, UK
[3]Language Technology Lab, Cambridge University, UK
[4]Department of Computing, Imperial College London
[5]PolyAI Limited, London, UK
bht26@cam.ac.uk

## Abstract

Dialogue systems benefit greatly from optimizing on detailed annotations, such as transcribed utterances, internal dialogue state representations and dialogue act labels. However, collecting these annotations is expensive and time-consuming, holding back development in the area of dialogue modelling. In this paper, we investigate semi-supervised learning methods that are able to reduce the amount of required intermediate labelling. We find that by leveraging un-annotated data instead, the amount of turn-level annotations of dialogue state can be significantly reduced when building a neural dialogue system. Our analysis on the MultiWOZ corpus, covering a range of domains and topics, finds that annotations can be reduced by up to 30% while maintaining equivalent system performance. We also describe and evaluate the first end-to-end dialogue model created for the MultiWOZ corpus.

## 1 Introduction

Task-oriented dialogue models aim at assisting with well-defined and structured problems like booking tickets or providing information to visitors in a new city (Raux et al., 2005). Most current industry-oriented systems rely on modular, domain-focused frameworks (Young et al., 2013; Sarikaya et al., 2016), with separate components for user understanding (Henderson et al., 2014), decision making (Gašić et al., 2010) and system answer generation (Wen et al., 2015). Recent progress in sequence-to-sequence (seq2seq) modelling has enabled the development of fully neural end-to-end architectures, allowing for different components to be optimized jointly in order to share information (Wen et al., 2017; Zhao et al., 2017; Budzianowski and Vulić, 2019).

Dialogue systems benefit greatly from optimizing on detailed annotations, such as turn-level dialogue state labels (Henderson et al., 2014) or dialogue actions (Rieser and Lemon, 2011), with end-to-end architectures still relying on intermediate labels in order to obtain satisfactory results (Lei et al., 2018). Collecting these labels is often the bottleneck in dataset creation, as the process is expensive and time-consuming, requiring domain and expert knowledge (Asri et al., 2017). Due to this restriction, existing datasets for task-oriented dialogue are several orders of magnitude smaller compared to general open-domain dialogue corpora (Lowe et al., 2015; Henderson et al., 2019).

Arguably, one of the most challenging parts of dialogue modelling is maintaining an interpretable internal memory over crucial domain concepts (Williams et al., 2016). Although there is increasing research effort to learn Dialogue State Tracking (DST) jointly with the text generation component (Eric et al., 2017; Wu et al., 2019), the most effective models use it as an intermediate signal (Wen et al., 2017; Lei et al., 2018). The difficulty of state tracking has made this task a driving force behind most of the Dialog System Technology Challenges in recent years (Henderson et al., 2014; Kim et al., 2017).

In this paper, we reduce the reliance of task-oriented dialogue systems on data collection by leveraging semi-supervised training (Chapelle et al., 2009). Two approaches are investigated and evaluated for providing an improved training signal to the dialogue state tracking component in an end-to-end dialogue system. Automatically predicted DST output on unlabelled utterances is treated as additional annotation if the model confidence is sufficiently high. Furthermore, subtle perturbations of existing datapoints are created, optimizing for their predictions to be similar to the original instances. Our analysis on the MultiWOZ corpus (Budzianowski et al., 2018), covering a range of domains and topics, finds that these meth-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1273–1278, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

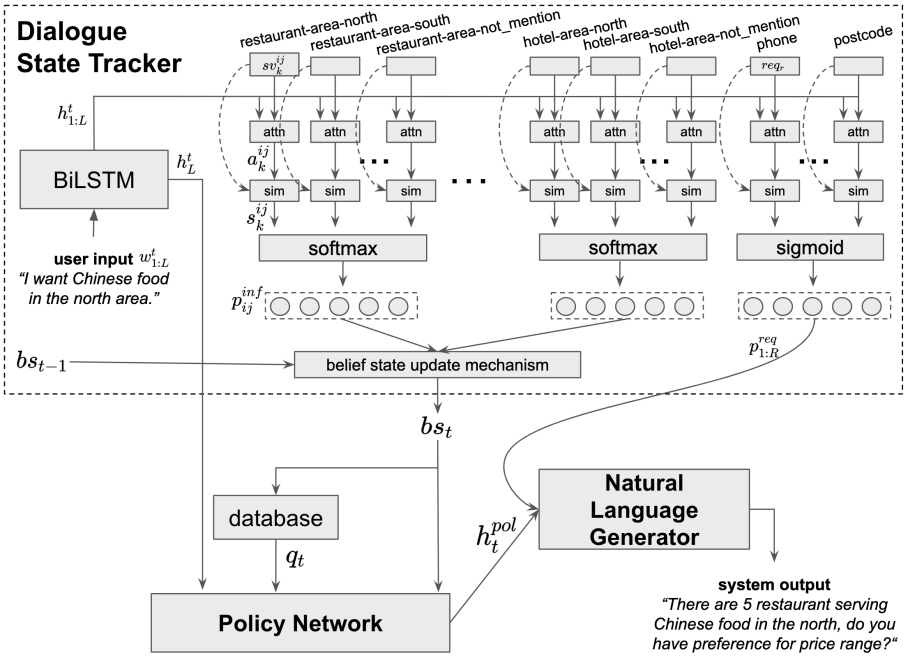


Figure 1: Overview of our end-to-end neural dialogue model. It is composed of three main components: Dialogue State Tracker, Policy Network and Natural Language Generator.

ods can reduce intermediate annotation by up to 30% while maintaining equivalent system performance. We also describe and evaluate the first end-to-end dialogue model created for the MultiWOZ corpus.

## 2 End-to-end Neural Dialogue Model

We now present the end-to-end neural dialogue model composed of three main components: dialogue state tracker, policy network and natural language generator. These components are trained jointly as a connected network and will be introduced in detail in the next paragraphs. The overall architecture can be seen in Figure 1.

**Dialogue State Tracker (DST)** The DST is responsible for both understanding the input user utterance and updating the internal dialogue state for the downstream components. There are two types of slots which can be detected in the input: informable slots and requestable slots. The former describes the attributes of the entity that the user is looking for, e.g., `pricerange` of a hotel. The latter captures information that the user desires to know about the entity, e.g., `postcode` of a hotel. Each informable slot contains several possible values with two special labels: `not-mentioned` and `don't-care`. A good DST should be able to correctly recognize the mentioned slot-value pairs in the user utterance and to maintain the updated dialogue (belief) state.

Let $i$, $j$ and $k$ denote the index of domain, slot and value. As depicted at the top of Figure 1, the user utterance $w_1$:$w_L$ at turn $t$ is first encoded by the BiLSTM to obtain the hidden states $h^t_{1:L}$. The encoding of the slot-value pair $sv^{ij}_k$ is the output of the affine layer that takes the concatenation of the embeddings of domain $i$, slot $j$ and value $k$ as the input. The context vector $a^{ij}_k$ is then computed by the attention mechanism, denoted as `attn` in Figure 1, following Luong et al. (2015):

$$e_l = \text{sim}(h_l, sv^{ij}_k) \quad (1)$$

$$a^{ij}_k = \sum_{l=1}^{L} e_l h_l, \quad (2)$$

where $l$ is the word index of the user utterance and `sim` denotes any function that calculates the similarity of two vectors. We adopt here the dot product function, following Mrkšić et al. (2017); Zhong et al. (2018); Ramadan et al. (2018). The similarity score $s^{ij}_k$ between $a^{ij}_k$ and $sv^{ij}_k$ is then computed to see whether the slot-value pair $sv^{ij}_k$ is mentioned in the utterance. The mentioned pair should have higher similarity score to its context vector than those which are not mentioned. The softmax layer is then applied to form the probability distribution $p^{inf}_{ij}$ for each informable slot $s^{inf}_{ij}$, where the predicted value is the value with

the highest probability. The same attention mechanism is used for each requestable slot $req_r$ to decide whether the user has asked for the slot in the current turn. The sigmoid layer is used instead as it is a binary classification problem. The prediction of requestable slots will be used as input to the natural language generator.

The belief state is the concatenation of the distributions over all informable slot-value pairs that is updated at each turn to keep track of the information provided by the user during the entire conversation. To form the belief state $bs_t$ at turn $t$, for each informable slot $s_{ij}^{inf}$ the update mechanism checks if the predicted value is either `not-mention` or `dont-care` for the current turn. If it is, then the probabilistic distribution $p_{ij}^{inf}$ in $bs_{t-1}$ is kept, otherwise it is updated by the new distribution $p_{ij}^{inf}$ at the current turn.

**Policy Network** The policy network is responsible for fusing the signals from the belief state $\mathbf{b}_t$, the encoding of the user utterance $\mathbf{h}_L^t$ and the database query result $\mathbf{q}_t$. The database query is constructed from the predicted belief state. The number of all entities that match the predictions of the DST form the database query vector[1]. We use a simple feedforward layer as the policy network:

$$\mathbf{z}_t = \tanh(W^z * [\mathbf{b}_t, \mathbf{h}_L^t, \mathbf{q}_t]). \quad (3)$$

where [*] denotes the concatenation of vectors.

**Natural Language Generator** Taking the input $\mathbf{z}_t$ from the policy network and predictions of requestable slots from the tracker, the generator outputs a system response word-by-word until the <EOS> token is generated. To improve the generation of correct slots corresponding to the user input, we adopt the semantically-conditioned LSTM (Wen et al., 2015) that contains a self-updating gate memory to record the produced slots in the generated sentence.

**Optimization** The model is optimized jointly against two sources of information – DST intermediate labels and system utterances. The DST loss consists of the cross-entropy over the multi-class classification of informable slots while binary cross-entropy is used for requestable slots:

$$L_{dst} = -\sum_i \sum_j t_{ij}^{inf} \log p_{ij}^{inf} - \sum_r t_r^{req} \log p_r^{req}, \quad (4)$$

where $i$, $j$ and $r$ are the index of domain, informable slot and requestable slot respectively; $t^*$ is the target distribution. The generation error is a standard cross-entropy between the predicted words and target words:

$$L_{gen} = -\sum_{l'} t_{l'} \log p_{l'}, \quad (5)$$

where $l'$ is the word index in the generated sentence. To jointly train the DST, the policy network, and the generator as a connected network, the final objective function becomes $L = L_{dst} + L_{gen}$.

**Semi-supervised Training** The DST loss requires each turn to be manually annotated with the correct slot-value pairs. We experiment with two different semi-supervised training methods that take advantage of unlabelled examples instead, allowing us to reduce the amount of required annotation. The first approach is based on the *pseudo-labelling* strategy by Chapelle et al. (2009). If the prediction probability of an unlabelled data point for a particular class is larger than a given threshold $\nu$, the example is included in the DST loss with the predicted label. The $\nu$ parameter is optimized on the validation set during development.

The second semi-supervised technique investigated is the $\Pi$-*model* (Sajjadi et al., 2016) where the input is perturbed with random noise $\epsilon \sim \mathcal{N}(\mathbf{0}, \sigma)$. The perturbations are applied to both labelled and unlabelled data points at the level of embedding of user utterance. The model is then required to produce similar predictions $p_{ij}^{inf}$ over the belief state compared to the original input, optimized with an additional loss:

$$L3 = \alpha \frac{1}{N} \sum_N \sum_{ij} (t_{ij}^{inf} - p_{ij}^{inf})^2, \quad (6)$$

where $N$ is the batch size and $\alpha$ is a hyperparameter controlling the weight of the loss.

## 3 Experiments

We investigate the effects of semi-supervised training on optimizing the end-to-end neural dialogue system. In particular, we evaluate how much annotation at the intermediate-level could be reduced while preserving comparable results of the overall dialogue task completion metrics.

[1]Following (Wen et al., 2017), by querying the database using $\mathbf{b}_t$ as query, $\mathbf{q}_t$ is the 1-hot representation with each element indicating different number of the matching entities in the database. We use 5 bins to indicate the matching number from 0 to 3 and more than 3.

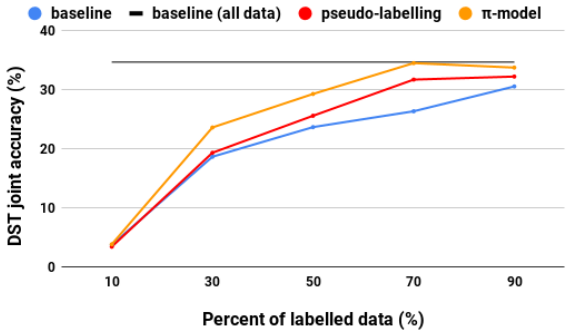


Figure 2: The DST joint accuracy for the three considered models as the amount of labelled data varies. The horizontal line denotes the baseline model trained on 100% labelled data.

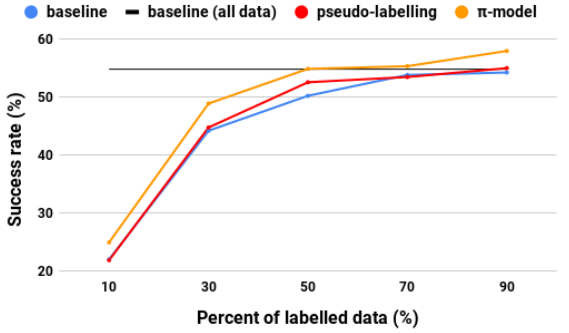


Figure 3: Success rate for different methods as the amounts of labelled data varies. Horizontal line denotes the baseline model trained on 100% labelled data.

**Dataset** The three analyzed models are evaluated on the MultiWOZ dataset consisting of 10,438 task-oriented dialogues (Budzianowski et al., 2018). The conversations in MultiWOZ are natural as they were gathered based on human-to-human interactions following the Wizard-of-Oz paradigm. The corpus includes multi-domain conversations spanning across 7 domains including Restaurant, Hotel, Attraction, Train and Taxi. The size of the dataset allows us to control the amount of available fully labelled datapoints.

**Metrics** There are two metrics of importance when evaluating task-oriented dialogue systems. The first is the DST joint goal accuracy, defined as an average joint accuracy over all slots per turn (Williams et al., 2016). The second is the *Success* metric that informs how many times systems have presented the entity satisfying the user's goal and provided them with all the additional requested information (Wen et al., 2017). The models are optimized using the validation set and the results are averaged over 10 different seeds.

**Varying data amount** We examine the performance of the baseline model compared to the two semi-supervised models as the amount of labelled data varies. The result of the DST joint accuracy is presented in Figure 2. The *pseudo-labelling* model performs better than the baseline when more than 50% of the dataset is labelled. At the scarce data levels (10% and 30%) , the *pseudo-labelling* model is not producing pseudo training points that help improve DST predictions. In contrast, the Π-*model* takes advantages of the additional regularization loss and effectively leverages unlabelled data to enhance the performance over the baseline. The improvements are consistently more than 5% when training with 30 to 90% of labelled data and even reach the performance of the fully trained baseline model with only 70% labelled data.

Figure 3 shows the *Success* metric results. The *pseudo-labelling* method is not able to improve performance over the baseline regardless of the amount of labelled data. However, the Π-*model* is capable of improving the success rate consistently and manages to reach the performance of the fully trained model with only 50% of the intermediate DST signal. Note that a better DST joint accuracy does not necessarily translate to a better success rate as the final metric is also influenced by the quality of the generator.

| No. of examples | 0 | 1-5 | 6-10 | 10-15 | 16-20 |
|---|---|---|---|---|---|
| **Baseline** | 6.17 | 15.93 | 25.71 | 35.07 | 28.88 |
| **Pseudo-labelling** | 6.5 | 16.27 | 26.96 | 33.46 | 28.55 |
| **Π-model** | **6.6** | **21.93** | **31.29** | **36.22** | **30.72** |

Table 1: The accuracy (%) of different classification of slot-value pairs in terms of their number of training examples.

**DST analysis** DST joint accuracy considers all slot-value pairs in an utterance and cannot give us further insight regarding the source of the improvements. We are particularly interested in whether the semi-supervised models can leverage unlabelled data to improve the prediction of rarely seen slot-value pairs. In this analysis, we classify all slot-value pairs in the test set in terms of their number of training examples in 50% of the labelled data. Table 1 presents the results, showing that the Π-*model* improves accuracy by 5% when the slot-value pair is rarely (1-10 times) seen during training. The improvement on few-shot slots contributes to the improvement of joint accuracy.




Figure 4: Success rates in each domain in the case of 50% labelled data.

**Domain analysis** We also investigate if the improvements in the success rate are consistent among all domains. Figure 4 shows the success rate on individual domains in the case of 50% of data is labelled. Both semi-supervised models improve performance over the baseline in all domains except for the taxi domain. We hypothesize this comes from the fact that the taxi domain is a relatively easy domain with only 4 possible slots.

## 4 Conclusions and Future Work

In this paper, we have analyzed how much semi-supervised techniques could help to reduce the need for intermediate-level annotations in training neural task-oriented dialogue models. The results suggest that we do not need to annotate all intermediate signals and are able to leverage unannotated examples for training these components instead. In the future, we plan to experiment with other intermediate signals like dialogue acts. Further improvements could potentially be obtained from employing more advanced regularization losses (Oliver et al., 2018).

## Acknowledgments

Bo-Hsiang Tseng is supported by Cambridge Trust and the Ministry of Education, Taiwan. Marek Rei's research is supported by Cambridge English via the ALTA Institute. Paweł Budzianowski is funded by an EPSRC grant (EP/M018946/1) and by Toshiba Research Europe Ltd, Cambridge Research Laboratory (RG85875).

## References

Layla El Asri, Hannes Schulz, Shikhar Sharma, Jeremie Zumer, Justin Harris, Emery Fine, Rahul Mehrotra, and Kaheer Suleman. 2017. Frames: A corpus for adding memory to goal-oriented dialogue systems. *Proceedings of SigDial*, pages 207–219.

Paweł Budzianowski and Ivan Vulić. 2019. Hello, it's GPT-2 - how can I help you? towards the use of pre-trained language models for task-oriented dialogue systems. *arXiv preprint arXiv:1907.05774*.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. 2018. MultiWOZ-A Large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In *Proceedings of EMNLP*, pages 5016–5026.

Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. 2009. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542.

Mihail Eric, Lakshmi Krishnan, Francois Charette, and Christopher D. Manning. 2017. Key-value retrieval networks for task-oriented dialogue. In *Proceedings of SigDial*, pages 37–49.

Milica Gašić, Filip Jurčíček, Simon Keizer, François Mairesse, Blaise Thomson, Kai Yu, and Steve Young. 2010. Gaussian processes for fast policy optimisation of pomdp-based dialogue managers. In *Proceedings of SigDial*, pages 201–204.

Matthew Henderson, Paweł Budzianowski, Iñigo Casanueva, Sam Coope, Daniela Gerz, Girish Kumar, Nikola Mrkšić, Georgios Spithourakis, Pei-Hao Su, Ivan Vulić, and Tsung-Hsien Wen. 2019. A repository of conversational datasets. In *Proceedings of the Workshop on NLP for Conversational AI*.

Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014. The third dialog state tracking challenge. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*, pages 324–329. IEEE.

Seokhwan Kim, Luis Fernando DHaro, Rafael E Banchs, Jason D Williams, and Matthew Henderson. 2017. The fourth dialog state tracking challenge. In *Dialogues with Social Robots*, pages 435–449. Springer.

Wenqiang Lei, Xisen Jin, Min-Yen Kan, Zhaochun Ren, Xiangnan He, and Dawei Yin. 2018. Sequicity: Simplifying task-oriented dialogue systems with single sequence-to-sequence architectures. In *Proceedings of ACL*, pages 1437–1447.

Ryan Lowe, Nissan Pow, Iulian V Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, page 285.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2017. Neural belief tracker: Data-driven dialogue state tracking. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1777–1788.

Avital Oliver, Augustus Odena, Colin A Raffel, Ekin Dogus Cubuk, and Ian Goodfellow. 2018. Realistic evaluation of deep semi-supervised learning algorithms. In *Advances in Neural Information Processing Systems*, pages 3235–3246.

Osman Ramadan, Paweł Budzianowski, and Milica Gašić. 2018. Large-scale multi-domain belief tracking with knowledge sharing. In *Proceedings of ACL*, pages 432–437.

Antoine Raux, Brian Langner, Dan Bohus, Alan W Black, and Maxine Eskenazi. 2005. Let's go public! taking a spoken dialog system to the real world. In *Ninth European Conference on Speech Communication and Technology*.

Verena Rieser and Oliver Lemon. 2011. *Reinforcement learning for adaptive dialogue systems: a data-driven methodology for dialogue management and natural language generation*. Springer Science & Business Media.

Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. 2016. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *Advances in Neural Information Processing Systems*, pages 1163–1171.

Ruhi Sarikaya, Paul A Crook, Alex Marin, Minwoo Jeong, Jean-Philippe Robichaud, Asli Celikyilmaz, Young-Bum Kim, Alexandre Rochette, Omar Zia Khan, Xiaohu Liu, et al. 2016. An overview of end-to-end language understanding and dialog management for personal digital assistants. In *2016 IEEE Spoken Language Technology Workshop (SLT)*, pages 391–397. IEEE.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gašić, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of EACL*, pages 438–449.

Jason Williams, Antoine Raux, and Matthew Henderson. 2016. The dialog state tracking challenge series: A review. *Dialogue & Discourse*, 7(3):4–33.

Chien-Sheng Wu, Richard Socher, and Caiming Xiong. 2019. Global-to-local memory pointer networks for task-oriented dialogue. *ICLR*.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D. Williams. 2013. POMDP-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Proceedings of ACL*, pages 654–664.

Victor Zhong, Caiming Xiong, and Richard Socher. 2018. Global-locally self-attentive encoder for dialogue state tracking. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1458–1467.

# A Progressive Model to Enable Continual Learning for Semantic Slot Filling

**Yilin Shen**[1] **Xiangyu Zeng**[2*] **Hongxia Jin**[1]
[1] Samsung Research America, Mountain View, CA, USA
[2] Columbia University, New York, NY, USA
[1] {yilin.shen,hongxia.jin}@samsung.com [2] xz2571@columbia.edu

## Abstract

Semantic slot filling is one of the major tasks in spoken language understanding (SLU). After a slot filling model is trained on pre-collected data, it is crucial to continually improve the model after deployment to learn users' new expressions. As the data amount grows, it becomes infeasible to either store such huge data and repeatedly retrain the model on all data or fine tune the model only on new data without forgetting old expressions. In this paper, we introduce a novel progressive slot filling model, *ProgModel*. ProgModel consists of a novel context gate that transfers previously learned knowledge to a small size expanded component; and meanwhile enables this new component to be fast trained to learn from new data. As such, ProgModel learns the new knowledge by only using new data at each time and meanwhile preserves the previously learned expressions. Our experiments show that ProgModel needs much less training time and smaller model size to outperform various model fine tuning competitors by up to 4.24% and 3.03% on two benchmark datasets.

## 1 Introduction

Spoken language understanding (SLU) systems play a vital role in ubiquitous artificially intelligent voice-enabled personal assistants. As one of the major tasks in SLU, semantic slot filling is treated as a sequential labeling problem to map a natural language sequence $\mathbf{x}$ to a slot label sequence $\mathbf{y}$ of the same length in IOB format (Yao et al., 2014). Typically, a slot filling model is trained offline on large scale corpora with pre-collected utterances. However, such corpora usually cannot cover all possible varieties of utterances exhaustively (e.g., personalized expressions, new vocabulary, utterances for new intent, etc.) from diverse users. Thus, it is critically desirable to develop a slot filling approach with the capability of continual learning after a personal assistant is deployed.

Unfortunately, existing approaches target on offline model training using a large scale training data. They are designed to either train a slot filling model independently (Yao et al., 2014; Peng et al., 2015; Kurata et al., 2016; Hakkani-Tür et al., 2016; Liu and Lane, 2016; Deng et al., 2019; Ray et al., 2019) or jointly with the other intent detection task in SLU (Guo et al., 2014; Liu and Lane, 2016; Zhang and Wang, 2016; Wang et al., 2018; Goo et al., 2018). Recently, (Shen et al., 2018a, 2019) developed cold start algorithms to generate training data with the hope of covering more varieties before deployment. On the other hand, (Ray et al., 2018; Shen et al., 2018b) attempt to personalize the slot filling model. However, they are still restricted to the offline training and cannot be applied to learn new user's expressions after deployment.

To support continual learning, a naive solution is to retrain the current model at each time. However, it suffers from several drawbacks: First, in order to maintain the SLU performance on both original and new expressions, it usually requires almost retraining the model using the whole dataset. However, as the size of training set grows, it becomes infeasible to repeatedly conduct time consuming retraining on such a large dataset. More importantly, the old training data typically is not stored permanently due to huge storage need and privacy protection. If only fine tuned on new utterances, the new model intends to lose the previously learned knowledge, a.k.a., *catastrophic forgetting* (French, 1999). Moreover, such fine-tuning of a large generic SLU model, even only on new utterances, is still quite inefficient. Recently, there are some progresses on continual learning in

[*] This work was done when Xiangyu Zeng was in Samsung Research.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1279–1284, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

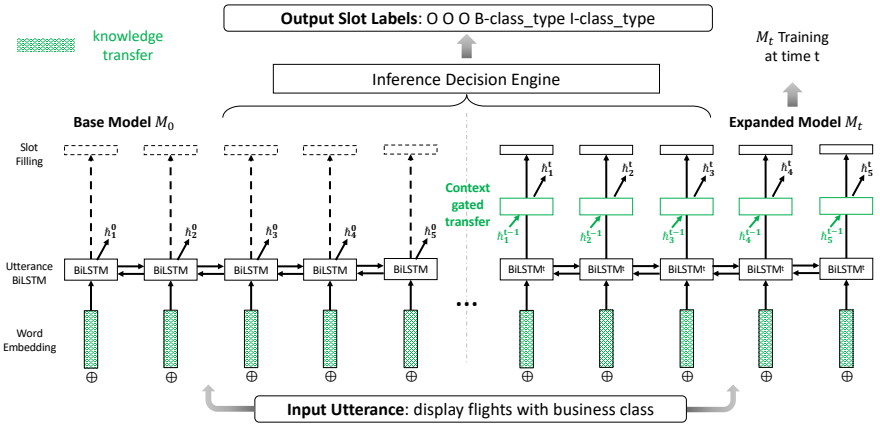


**Figure 1: Our Proposed ProgModel Architecture: consists of an expanded component $M^t$ at each batch $t$ with context gated transfer of utterance contexts (Figure 2); and an inference decision engine to derive the label prediction. At each batch $t$, the last layers of base model and previous components (dotted lines) are only used for inference. Only the output of $M^t$ is used to guide the training.**

computer vision (Li and Hoiem, 2016; Lee et al., 2017), yet it still remains open in spoken language understanding systems.

In this paper, we consider a practical setting that a batch[1] of new training data $U^t$ becomes available at each batch $t$. Our goal is to enable the continual learning capability of a slot filling model such that it can *keep learning new utterances efficiently as well as remember old knowledge without the needs of accessing old training data*. To achieve this, we design a novel *Progressive Slot Filling Model (ProgModel)* that can be gradually expanded at each batch by using a novel context gate for knowledge transfer. Unlike the baseline that repeatedly retrains the same model, ProgModel keeps the previously trained components untouched such that the catastrophic forgetting can be largely avoided. Using the transferred knowledge, each newly expanded component in ProgModel is trained in a progressive manner to achieve better performance with faster training compared with baseline model retraining approaches .

## 2 Proposed Approach

### 2.1 Progressive Model (ProgModel)

As the name indicates, the main idea of our proposed ProgModel is to progressively expand the model by transferring existing knowledge from the current model. Thus, ProgModel can continually enhance its capability of understanding user's new expressions without catastrophic forgetting. This is motivated by the recent success of progressive neural networks in various applications (Rusu et al., 2016).

As shown in Figure 1, ProgModel consists of the following components: (1) *Expanded Components:* The current model is expanded via context-gated knowledge transfer to allow only training on a new batch training set $U^t$ at each batch $t$. (2) *Inference Decision Engine:* When we receive multiple outputs from base model and expanded components, the decision engine is to derive the slot filling label output without additional training.

#### 2.1.1 Expanded Component $M^t$

At each batch $t$, a new component $M^t$ (Figure 1 (right)) is expanded on the base model $M^0$ and previously expanded components $M^1 \dots M^{t-1}$, denoted as $M^{<t}$. The utterance BiLSTM$^t$ is learned from scratch at each batch $t$ such that it can learn the new sentence structures via word sequence correlations. Next, we focus on designing two knowledge transfer mechanisms (green parts in Figure 1) to maximally leverage the previously learned knowledge in $M^{<t}$.

*Word Embeddings Transfer:* Word embeddings in the newly expanded component $M^t$ are initialized using those in $M^0$ based on the assumption that $M^0$ covers most vocabulary. For a new word $w$, we initialize using GloVe embedding. The embeddings will be fine tuned during training $M^t$.

*Gated Utterance Context Transfer:* We then design a novel *context gate* (Figure 2) by introducing lateral connections from most recent component $M^{t-1}$. *Note that this context gate is crucial to avoid retraining on all data or repeatedly duplicating a large base model every time.* Specifically, the lateral connections transfer the hidden states $\hbar_i^{t-1}$ for each word $i$ from $M^{t-1}$ to the newly expanded component $M^t$. They are projected to the same dimensional space of $\mathbf{h}_i^t$ via

[1]To avoid the confusion with the widely used timestamp in NLP (mean each word), we use each batch in our paper.

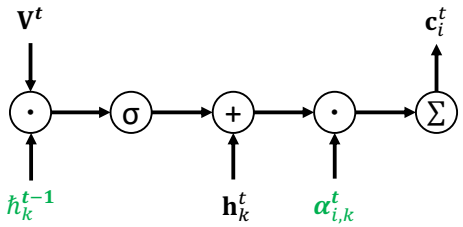


**Figure 2: Context Gate for Transfer Utterance Context Knowledge from $M^{t-1}$ (in green)**

projection matrix $\mathbf{V}^t$ shared for each word:

$$\hbar_i^t = \mathbf{h}_i^t + \sigma(\mathbf{V}^t \hbar_i^{t-1})$$

where $\sigma(\cdot)$ is the sigmoid function.

Then, the context vector $\mathbf{c}_i^t$ for the $i^{th}$ word in expanded component $M^t$ is given as:

$$\mathbf{c}_i^t = \sum_{k=1}^{n} \alpha_{i,k}^t \hbar_k^t$$

where $\boldsymbol{\alpha}_i^t = softmax(\mathbf{e}_i)$ is fine tuned based on $\boldsymbol{\alpha}_i^0$ trained in base model $M^0$ as in Att-BiRNN model (Liu and Lane, 2016). We have $\alpha_{i,j}^0 = \frac{\exp(e_{i,j}^0)}{\sum_{k=1}^{T} \exp(e_{i,k}^0)}$ in which $e_{i,k}^0 = g^0(\mathbf{h}_{i-1}^0 \oplus \mathbf{c}_{i-1}^0, \mathbf{h}_k^0)$ is learned from a feed forward neural network $g^0$. In ProgModel, each model $M^t$ has an independent $g^t$ which is initialized by $g^0$ in $M^0$. Thus, $\boldsymbol{\alpha}_i^t$ will be fine tuned from $\boldsymbol{\alpha}_i^0$ during training $M^t$ via updated $g^t$ and $\hbar_i^t$.

### 2.1.2 Inference Decision Engine

We design the inference decision engine (IDE) as a non-trainable separate component to avoid the potential catastrophic forgetting. Thanks to the capability of knowledge transfer in ProgModel, $M^t$ can already remember quite much previously learned knowledge to give good label prediction in many cases. Thus, we consider two types of decision engines: (1) *t-IDE*: ProgModel using only the output of $M^t$ as decision engine; (2) *c-IDE*: for $i^{th}$ word, it combines all outputs from each component $M^t$, $\sum_{k=0}^{t} P^k(i) I^k(i)$. $I^k(i)$ is an indicator function which is 1 when $i^{th}$ word is in the vocabulary of $M^k$ and 0 otherwise. The label with maximum probability is selected.

## 2.2 Progressive Training

The training procedure is progressively conducted at each batch $t$. The first step is to train the base model $M^0$ using the loss function $\mathcal{L}^0$:

$$\mathcal{L}^0(\boldsymbol{\theta}^0) \triangleq -\frac{1}{n} \sum_{j=1}^{|S|} \sum_{i=1}^{n} y_j(i) \log P_j^0(i)$$

where $\boldsymbol{\theta}^0$ are the parameters in $M^0$; $|S|$ is the number of semantic slots in IOB format; and $n$ is the sequence length.

**Table 1: Grouping Statistics in All Datasets**

| Dataset | Domain | Group | Vocab Size | #Slots | Train Size |
|---|---|---|---|---|---|
| ATIS | - | 1 | 706 | 49 | 3,666 |
| | | 2 | 301 | 24 | 423 |
| | | 3 | 271 | 18 | 232 |
| | | 4 | 312 | 29 | 266 |
| | | 5 | 309 | 33 | 391 |
| Snips | Add To Playlist | 1 | 2,561 | 6 | 1417 |
| | | 2 | 802 | 5 | 259 |
| | | 3 | 764 | 5 | 259 |
| | Book Restaurant | 1 | 1,588 | 14 | 995 |
| | | 2 | 1,044 | 14 | 487 |
| | | 3 | 990 | 14 | 486 |
| | Get Weather | 1 | 1,095 | 9 | 805 |
| | | 2 | 977 | 9 | 592 |
| | | 3 | 946 | 9 | 591 |
| | Play Music | 1 | 2,333 | 9 | 1,546 |
| | | 2 | 464 | 9 | 210 |
| | | 3 | 492 | 9 | 210 |
| | Rate Book | 1 | 916 | 7 | 822 |
| | | 2 | 708 | 6 | 539 |
| | | 3 | 722 | 6 | 538 |
| | Search Creative Work | 1 | 1,353 | 2 | 690 |
| | | 2 | 1,298 | 2 | 631 |
| | | 3 | 1,335 | 2 | 630 |
| | Search Screening Event | 1 | 863 | 7 | 883 |
| | | 2 | 687 | 7 | 472 |
| | | 3 | 668 | 7 | 471 |

At each batch $t$, we train the expanded component $M^t$ while fixing the parameters $\boldsymbol{\theta}^{<t}$ in previous components. The loss is backpropagated from the output of $M^t$ using the loss function $\mathcal{L}^t$:

$$\mathcal{L}^t(\boldsymbol{\theta}^t, \boldsymbol{\phi}_{t-1}^t) \triangleq -\frac{1}{n} \sum_{j=1}^{|S|} \sum_{i=1}^{n} y_j(i) \log P_j^t(i)$$

where $\boldsymbol{\theta}^t$ and $\boldsymbol{\phi}_{t-1}^t$ are the parameters in model $M^t$ and in the context gate between $M^{t-1}$ and $M^t$. In both loss functions, $P_j^t(i)$ is output probability of slot $j$ for the $i^{th}$ word from $M^t$.

# 3 Experimental Results

## 3.1 Datasets & Settings

**Dataset:** We evaluate ProgModel on the following two benchmark datasets:

*ATIS (Airline Travel Information Systems) dataset* (Hemphill et al., 1990): a widely used dataset in SLU research. The training set contains 4,978 utterances from the ATIS-2 and ATIS-3 corpora, and the testing set contains 893 utterances from the ATIS-3 data sets. There are 127 distinct slot labels. We do not use the intent labels in ATIS.

*Snips dataset* (Snips, 2017): another NLU dataset custom-intent-engines collected by Snips for model evaluation. It contains 7 domains. In each domain, the training set contains 1,800 to 2,000 utterances and the testing set contains around 100 utterances. Since each domain in

**Table 2: Performance (F1 Score) on ATIS Dataset**

| Approach | Batch 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| AttRNN (upper bound) | 92.12 | 92.89 | 93.04 | 93.56 | 95.13 |
| FT-AttRNN | | 91.85 | 89.98 | 91.25 | 88.03 |
| FT-Lr-AttRNN | | 91.96 | 86.46 | 88.03 | 86.58 |
| FT-Cp-AttRNN | 92.12 | 92.10 | 90.06 | 91.98 | 89.67 |
| t-ProgModel | | 92.33 | 92.43 | 92.57 | 92.58 |
| c-ProgModel | | **92.40** | **92.64** | **92.71** | **93.91** |

**Table 3: Performance (F1 Score) on Snips Dataset**

| Domain | Approach | Batch 0 | 1 | 2 |
|---|---|---|---|---|
| **Add To Playlist** | AttRNN (upper bound) | 79.58 | 86.74 | 88.89 |
| | FT-AttRNN | | 81.23 | 87.07 |
| | FT-Lr-AttRNN | | 78.99 | 86.61 |
| | FT-Cp-AttRNN | 79.58 | 84.67 | 87.15 |
| | t-ProgModel | | **86.12** | **88.30** |
| | c-ProgModel | | 85.51 | 87.25 |
| **Book Restaurant** | AttRNN (upper bound) | 79.49 | 89.78 | 90.03 |
| | FT-AttRNN | | 88.71 | 88.09 |
| | FT-Lr-AttRNN | | 88.57 | 87.89 |
| | FT-Cp-AttRNN | 79.49 | 89.06 | 88.14 |
| | t-ProgModel | | **89.45** | **89.54** |
| | c-ProgModel | | 89.40 | 89.40 |
| **Get Weather** | AttRNN (upper bound) | 76.48 | 91.12 | 93.56 |
| | FT-AttRNN | | 89.52 | 88.93 |
| | FT-Lr-AttRNN | | 89.09 | 88.56 |
| | FT-Cp-AttRNN | 76.48 | 89.82 | 90.09 |
| | t-ProgModel | | **90.73** | **93.12** |
| | c-ProgModel | | 89.92 | 90.95 |
| **Play Music** | AttRNN (upper bound) | 77.48 | 87.79 | 89.13 |
| | FT-AttRNN | | 84.71 | 84.63 |
| | FT-Lr-AttRNN | | 84.53 | 84.16 |
| | FT-Cp-AttRNN | 77.48 | 84.85 | 86.10 |
| | t-ProgModel | | 86.05 | 87.26 |
| | c-ProgModel | | **87.00** | **88.45** |
| **Rate Book** | AttRNN (upper bound) | 92.64 | 98.45 | 99.07 |
| | FT-AttRNN | | 96.87 | 96.83 |
| | FT-Lr-AttRNN | | 96.20 | 96.86 |
| | FT-Cp-AttRNN | 92.64 | 97.06 | 97.93 |
| | t-ProgModel | | 97.50 | **98.89** |
| | c-ProgModel | | **98.19** | 98.20 |
| **Search Creative Work** | AttRNN (upper bound) | 66.32 | 89.01 | 89.67 |
| | FT-AttRNN | | 85.93 | 85.46 |
| | FT-Lr-AttRNN | | 84.69 | 84.45 |
| | FT-Cp-AttRNN | 66.32 | 87.25 | 86.36 |
| | t-ProgModel | | 88.21 | 88.25 |
| | c-ProgModel | | **88.79** | **88.83** |
| **Search Screening Event** | AttRNN (upper bound) | 89.30 | 95.68 | 97.34 |
| | FT-AttRNN | | 93.40 | 94.53 |
| | FT-Lr-AttRNN | | 91.87 | 93.56 |
| | FT-Cp-AttRNN | 89.30 | 93.81 | 94.56 |
| | t-ProgModel | | **95.01** | **96.90** |
| | c-ProgModel | | 93.62 | 94.31 |

Snips contains completely different slots and very few vocabulary are shared between the domains, we evaluate on each domain independently.

We use Amazon Mechanical Turk (MTurk) to split both training set into non-overlapping groups in each dataset (each domain in Snips as a separate dataset). Based on the size of each dataset, we consider 5 groups in ATIS and 3 groups in each domain of Snips dataset. Each turker is given 100 utterances from training set in one dataset; as well as the number of groups $G$ for this dataset. He is asked to put utterances into no more than $G$ groups based on their similarities. At last, we review the grouped utterances from turkers to further combine the similar utterances and derive the final grouping of the whole dataset. For each dataset, we consider the largest group of training set as a base dataset. At each batch $t$, one of leftover groups in training set is randomly selected to be $U^t$. Table 1 shows the detailed data statistics.

**Competitors:** First, only for reference purpose, we consider a *performance upper bound baseline*, i.e., train AttRNN on the all available dataset at each batch $t$. We compare with the following competitor approaches: (1) *FT-AttRNN* : fine tunes the current model only using new training data $U^t$ at each batch $t$; (2) *FT-Lr-AttRNN* : fine tunes the current model using an adjusted lower learning rate (we use 0.3 times of base model learning rate which has the best performance) on the new training set $U^t$; (3) *FT-Cp-AttRNN* : copies the previous model and fine tunes the new copied model on new training data $U^t$ at each batch $t$. During inference, FT-Cp-AttRNN uses both t-IDE and c-IDE decision engines and reports the one with better performance (F-1 score). We evaluate our ProgModel model with different inference engines: (1) *t-ProgModel*: ProgModel using only output of $M^t$ as decision engine; (2) *c-ProgModel*: ProgModel using combined inference decision engine. All base models $M^0$ are trained on state-of-the-art AttRNN model (Liu and Lane, 2016). For fair evaluation, we test both ProgModel and competitors on the all standard testing sets.

**Training:** We implemented ProgModel model using TensorFlow 1.4.0 and conducted the experiments on NVIDIA Tesla M40. At each batch $t$, we train all models until their convergence. *We observe that ProgModel takes around 10 epochs due to less parameters and transferred knowledge in $M^t$ while AttRNN retraining usually needs 100 epochs and various fine tuning competitors need around 30-50 epochs.*

### 3.2 Main Results

Table 2 and Table 3 show the F1 score of slot filling performance comparison results on ATIS dataset and each domain of Snips dataset. The results show that ProgModel consistently outperforms AttRNN in all domains, where the improvement gain is up to 4.24% in ATIS and 3.03% in Snips. As expected, ProgModel

continuously improves performance with more and more new batches of training data, even though it is only trained on new data at each batch. Among all competitors, FT-Cp-AttRNN achieves the closest performance to ProgModel by using much larger model size (shown in Section 3.4). In comparison, both FT-AttRNN and FT-Lr-AttRNN frequently suffer from catastrophic forgetting. The values in pink show that the performance of FT-AttRNN and FT-Cp-AttRNN drops up to 3.82% and 5.38% respectively. As a result, their F1 scores are significantly reduced in the end. At last, we observe that ProgModel is quite close to upper bound performance (Note that this is only for reference rather than comparison since upper bound performance assumes the availability of all training data while ProgModel does not).

### 3.3 Ablation Study

We further look into each competitor to better understand the advantage of our method. Since FT-AttRNN is only trained on new data, it is oftentimes overwhelmed by new knowledge and results in forgetting the old knowledge. On the other hand, FT-Lr-AttRNN has difficulty to learn new knowledge since it cannot jump out of local optimum due to a small learning rate. As a result, the performance of FT-Lr-AttRNN is even lower than FT-AttRNN most of the time. To make it even worse, the learning rate is very hard to tune at each batch. As we can see, it is non-trivial to achieve both goals, learn new knowledge and remember old knowledge.

FT-Cp-AttRNN performs slightly better than FT-AttRNN and FT-Lr-AttRNN . FT-Cp-AttRNN can be treated as a naive solution to achieve both goals by almost duplicating the model again and again. However, in addition to larger model size and longer training time, it still suffers from efficiently transfer previous knowledge and leads to catastrophic forgetting from time to time.

In comparison, ProgModel outperforms all above competitors since it provides a systematic mechanism to achieve both goals. The training of our designed context gate helps to determine which knowledge to transfer at each batch.

At last, we observe that c-ProgModel performs better than t-ProgModel in ATIS. This has two reasons: First, the utterances in different groups of ATIS are quite structurally similar such that c-IDE further enhances the correct slot label distribution by combing all outputs together. Second, many slots are not included in last group of ATIS dataset. In Snips, t-ProgModel outperforms c-ProgModel in most domains since the utterances in different groups have significant different structures and vocabularies. Thus, inference outputs could be conflicted and their combination leads to worse performance.

**Table 4: Average Model Size (MB)**

| Domain | Batch | Approach | | |
|---|---|---|---|---|
| | | FT-AttRNN | FT-Cp-AttRNN | ProgModel |
| ATIS | 0 | 18.0 | 18.0 | 18.0 |
| | 1 | | 36.9 | 21.9 |
| | 2 | | 56.2 | 25.5 |
| | 3 | | 75.5 | 29.4 |
| | 4 | | 94.8 | 33.3 |
| Snips | 0 | 25.3 | 25.3 | 25.3 |
| | 1 | | 50.7 | 31.8 |
| | 2 | | 76.2 | 38.1 |

### 3.4 Model Size Results

Table 4 reports the model size comparison between FT-AttRNN (FT-AttRNN and FT-Lr-AttRNN ), FT-Cp-AttRNN and ProgModel (t-ProgModel and c-ProgModel). With more and more training data at each batch, the increase of ProgModel size is significantly slower than that of FT-Cp-AttRNN since the capability of knowledge transfer in ProgModel avoids the full model copy. Thus, our approach also better trades off the model size and performance. The small fluctuation of ProgModel model size expansion is due to the different size of vocabulary in each batch of training utterances. Each expanded model $M^t$ will only keep the embedding of vocabulary in new training data $U^t$. One may concern that ProgModel will become large over time. In practice, it will be expanded only when the new data grows too large to handle by current model. Moreover, a new base model can be periodically reinitialized to reset the model size.

## 4 Conclusion

In this paper, we proposed a novel ProgModel model with the capability of efficient continual learning for semantic slot filling in SLU. ProgModel is designed to expand progressively at each batch of new training data with a new context gate for knowledge transfer. The model can be trained progressively without needing to store old training data. We showed that ProgModel need much shorter training time to significantly outperform baseline approaches and close to the upper bound performance.

## References


Yue Deng, KaWai Chen, Yilin Shen, and Hongxia Jin. 2019. Adversarial multi-label prediction for spoken and visual signal tagging. In *ICASSP*, pages 3252–3256.

Robert M. French. 1999. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128 – 135.

Chih-Wen Goo, Guang Gao, Yun-Kai Hsu, Chih-Li Huo, Tsung-Chieh Chen, Keng-Wei Hsu, and Yun-Nung Chen. 2018. Slot-gated modeling for joint slot filling and intent prediction. In *NAACL-HLT*, pages 753–757.

Daniel Guo, Gokhan Tur, Wen-tau Yih, and Geoffrey Zweig. 2014. Joint semantic utterance classification and slot filling with recursive neural networks. In *IEEE SLT*, pages 554–559.

Dilek Hakkani-Tür, Gökhan Tür, Asli Celikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In *INTERSPEECH*, pages 715–719.

Charles T. Hemphill, John J. Godfrey, and George R. Doddington. 1990. The atis spoken language systems pilot corpus. In *Proceedings of the Workshop on Speech and Natural Language*, HLT, pages 96–101.

Gakuto Kurata, Bing Xiang, Bowen Zhou, and Mo Yu. 2016. Leveraging sentence-level information with encoder lstm for semantic slot filling. In *EMNLP*, pages 2077–2083.

Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. 2017. Overcoming catastrophic forgetting by incremental moment matching. In *NIPS*, pages 4652–4662.

Zhizhong Li and Derek Hoiem. 2016. Learning without forgetting. In *ECCV*, pages 614–629.

Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. In *INTERSPEECH*, pages 685–689.

Baolin Peng, Kaisheng Yao, Li Jing, and Kam-Fai Wong. 2015. Recurrent neural networks with external memory for spoken language understanding. In *NLPCC*, pages 25–35.

Avik Ray, Yilin Shen, and Hongxia Jin. 2018. Learning out-of-vocabulary words in intelligent personal agents. In *IJCAI*, pages 4309–4315.

Avik Ray, Yilin Shen, and Hongxia Jin. 2019. Iterative delexicalization for improved spoken language understanding. In *Interspeech*.

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. 2016. Progressive neural networks. *arXiv preprint arXiv:1606.04671*.

Yilin Shen, Avik Ray, Abhishek Patel, and Hongxia Jin. 2018a. CRUISE: cold-start new skill development via iterative utterance generation. In *ACL, System Demonstrations*, pages 105–110.

Yilin Shen, Yu Wang, Abhishek Patel, and Hongxia Jin. 2019. Sliqa-i: Towards cold-start development of end-to-end spoken language interface for question answering. In *ICASSP*, pages 7195–7199.

Yilin Shen, Xiangyu Zeng, Yu Wang, and Hongxia Jin. 2018b. User information augmented semantic frame parsing using progressive neural networks. In *INTERSPEECH*, pages 3464–3468.

Snips. 2017. https://github.com/snipsco/nlu-benchmark/tree/master/2017-06-custom-intent-engines.

Yu Wang, Yilin Shen, and Hongxia Jin. 2018. A bi-model based RNN semantic frame parsing model for intent detection and slot filling. In *NAACL-HLT*, pages 309–314.

Kaisheng Yao, Baolin Peng, Yu Zhang, Dong Yu, Geoffrey Zweig, and Yangyang Shi. 2014. Spoken language understanding using long short-term memory neural networks. In *SLT*, pages 189–194.

Xiaodong Zhang and Houfeng Wang. 2016. A joint model of intent determination and slot filling for spoken language understanding. In *IJCAI*, pages 2993–2999.

# CASA-NLU: Context-Aware Self-Attentive Natural Language Understanding for Task-Oriented Chatbots

**Arshit Gupta**[†] **Peng Zhang**[‡] **Garima Lalwani**[‡] **Mona Diab**[†]
[†]Amazon AI, Seattle [‡]Amazon AI, East Palo Alto
{arshig, pezha, glalwani, diabmona}@amazon.com

## Abstract

Natural Language Understanding (NLU) is a core component of dialog systems. It typically involves two tasks - intent classification (IC) and slot labeling (SL), which are then followed by a dialogue management (DM) component. Such NLU systems cater to utterances in isolation, thus pushing the problem of context management to DM. However, contextual information is critical to the correct prediction of intents and slots in a conversation. Prior work on contextual NLU has been limited in terms of the types of contextual signals used and the understanding of their impact on the model. In this work, we propose a context-aware self-attentive NLU (CASA-NLU) model that uses multiple signals, such as previous intents, slots, dialog acts and utterances over a variable context window, in addition to the current user utterance. CASA-NLU outperforms a recurrent contextual NLU baseline on two conversational datasets, yielding a gain of up to 7% on the IC task for one of the datasets. Moreover, a non-contextual variant of CASA-NLU achieves state-of-the-art performance for IC task on standard public datasets - SNIPS and ATIS.

## 1 Introduction

With the advent of smart conversational agents such as Amazon Alexa, Google Assistant, etc., dialogue systems are becoming ubiquitous. In the context of enterprises, the majority of these systems target task oriented dialogues with the user trying to achieve a goal, e.g. booking flight tickets or ordering food. Natural Language Understanding (NLU) captures the semantic meaning of a user's utterance within each dialogue turn, by identifying **intents** and **slots**. An intent specifies the goal underlying the expressed utterance while slots are additional parameters for these intents. These tasks are typically articulated as intent classification (IC) coupled with sequence tagging task of slot labeling (SL).

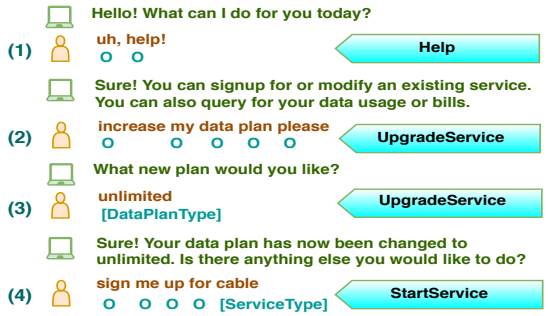


Figure 1: Snippet of a sample Cable data conversation with intents shown in boxes and slots aligned with user request

Over time, human-machine interactions have become more complex with greater reliance on contextual cues for utterance understanding (Figure 1). With traditional NLU frameworks, the resolution of contextual utterances is typically addressed in the DM component of the system using rule-based dialogue state trackers (DST). However, this pushes the problem of context resolution further down the dialogue pipeline, and despite the appeal of modularity in design, it opens the door for significant cascade of errors. To avoid this, end-to-end dialogue systems have been proposed (Wen et al., 2017; Bordes and Weston, 2016), but, to date, such systems are not scalable in industrial settings, and tend to be opaque where a level of transparency is needed, for instance, to understand various dialogue policies.

To address the propagation of error while maintaining a modular framework, Shi et al. (2015) proposed adding contextual signals to the joint IC-SL task. However, the contributions of their work were limited in terms of number of signals and how they were used, rendering the contextualization process still less interpretable. In this work, we present a multi-dimensional self-attention based contextual NLU model that overcomes prior work's shortcomings by supporting



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1285–1290, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

variable number of contextual signals (previous utterances, dialogue acts[1], intents and slot labels) that can be used concurrently over variable length of conversation context. In addition, our model allows for easy visualization and debugging of contextual signals which are essential, especially in production dialogue systems, where interpretability is a desirable feature. Our contributions are:

- Context-Aware Self-Attentive NLU (CASA-NLU) model that uses various contextual signals to perform joint IC-SL task, outperforming contextual and non-contextual NLU baselines by significant margins on two in-house conversational IC-SL datasets
- Propose a novel non-contextual variant of CASA-NLU that achieves SOTA performance on IC task for both SNIPS and ATIS datasets
- Analysis of the various contextual signals' contributions to model performance

## 2 Related Work

There have been numerous advancements in NLU systems for dialogues over the past two decades. While the traditional approaches used handcrafted features and word n-gram based features fed to SVM, logistic regression, etc. for IC task and conditional random fields (CRF) for SL task (Jeong and Lee, 2008; Wang1 et al., 2002; Raymond and Riccardi, 2007), more recent approaches rely on deep neural networks to jointly model IC and SL tasks (Yao et al., 2014a,b; Guo et al., 2014; Zhang and Wang, 2016; Liu and Lane, 2016c; Goo et al., 2018). Attention as introduced by Bahdanau et al. (2014) has played a major role in many of these systems (Liu and Lane, 2016a; Ma et al., 2017; Li et al., 2018a; Goo et al., 2018), for instance, for modeling interaction between intents and slots in (Goo et al., 2018).

Dahlbäck and Jönsson (1989) and Bertomeu et al. (2006) studied contextual phenomena and thematic relations in natural language, thereby highlighting the importance of using context. Few previous works focused on modeling turn-level predictions as DST task (Williams et al., 2013). However, these systems predict the possible slot-value pairs at utterance level (Zhong et al., 2018), making it necessary to maintain ontology of all possible slot values, which is infeasible for certain slot types (e.g., restaurant names). In industry settings, where IC-SL task is predominant, there is also an additional effort involved to invest in rules for converting utterance level dialog state annotations to token level annotations required for SL. Hence, our work mainly focuses on the IC-SL task which eliminates the need for maintaining any ontology or such handcrafted rules.

Bhargava et al. (2013) used previous intents and slots for IC and SL models. They were followed by Shi et al. (2015) who exploited previous intents and domain predictions to train a joint IC-SL model. However, both these studies lacked comprehensive context modeling framework that allows multiple contextual signals to be used together over a variable context window. Also, an intuitive interpretation of the impact of contextual signals on IC-SL task was missing.

## 3 CASA-NLU: Context-Aware Self-Attentive NLU

Our model architecture is composed of three sub sections - signal encoding, context fusion and IC-SL predictions (Figure 2).

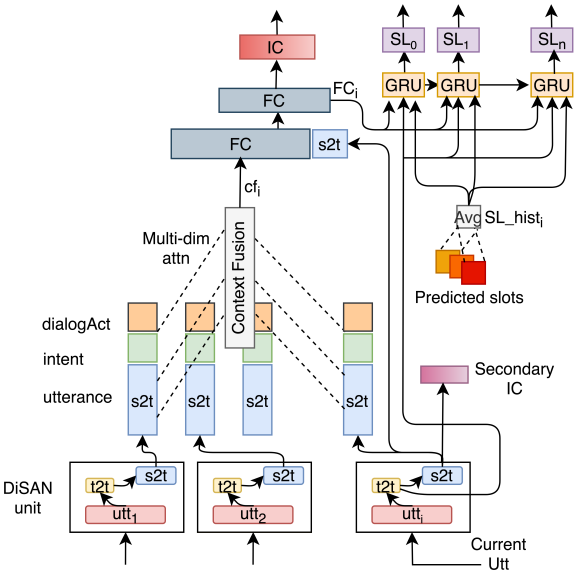


Figure 2: CASA-NLU model architecture for joint IC-SL

### 3.1 Signal Encoding

**Utterance (Utt):** For the utterance encoding, we adopt the directional self-attention (Shen et al., 2017) based encoder (DISAN), adding absolute position embedding (Gehring et al., 2017) to further improve the encoding performance.[2] DISAN unit consists of two types of multi-dimensional attention - word level token2token (t2t) attention fol-

[1]Dialog Act signifies the actions taken by the agent such as Close (when an intent is fulfilled), ElicitSlot, etc

[2]Details provided in Appendix A

lowed by sentence level source2token (s2t) attention. For turn $i$, the output of this unit is given by $\mathbf{h}(\text{Utt}_i) \in \mathbb{R}^{2d_h \times 1}$, where $d_h$ is hidden layer size of the DISAN unit (Figure 2).

**Intent / Dialog Act (DA) / Slot Label History:** We pass the one-hot representation of intent ground truth labels through an embedding layer to get intent history representation $\mathbf{h}(\text{I}_i) \in \mathbb{R}^{d_I \times 1}$ for any previous turn $i$ with $d_I$ being the intent embedding dimension. Similarly, for DA history, $\mathbf{h}(\text{DA}_i) \in \mathbb{R}^{d_{DA} \times 1}$. We use a special dummy symbol for the intent and DA of the current turn. For slot label history, for turn $i$, we take the average of all slot embeddings that were observed in previous turns giving $\mathbf{h}(\text{SL\_hist}_i) \in \mathbb{R}^{d_{SL} \times 1}$.

### 3.2 Context Fusion

We combine the vectorized signals together in both spacial and temporal dimensions. For the former, we simply concatenate the contextual signals to get current turn feature vector, i.e. $\mathbf{T}_i = [\mathbf{h}(\text{Utt}_i); \mathbf{h}(\text{I}_i); \mathbf{h}(\text{DA}_i)]$. However, for the latter, concatenation becomes intractable if context window is large. To address this issue and automatically learn more relevant components of context for each turn, we add a source2token (Shen et al., 2017) multi-dimensional self-attention layer over the turn vectors. This is essentially a per dimension learned weighted average ($\mathbf{cf}_i$) over all the turn vectors within a context window (Equation 1). As shown later in Section 5, this enhances the model's robustness by learning different attention weights for different contextual signals.

$$\mathbf{cf}_i = \sum_{t=i-K}^{i} \mathbf{P}(\mathbf{T}_t) \odot \mathbf{T}_t \qquad (1)$$

where, $K$ (= 3 in our experiments) is the context window, and $\mathbf{T}_t$ and $\mathbf{P}(\mathbf{T}_t)$ are the turn vector and attention weights for $t^{th}$ time step respectively. We use padding tokens for $i - K < 0$.

One of the shortcomings with such attention mechanism is that it is position invariant. To address this problem, we add learned absolute position embedding $\mathbf{p}_c$ to the turn matrix $\mathbf{T}$ that learns temporal information across the turns.

### 3.3 IC-SL Predictions

Following (Liu and Lane, 2016b; Li et al., 2018b), we train a joint IC-SL model. To improve IC performance for our deep network, we also add a secondary IC loss function, $\text{L}_{\text{Sec\_IC}}$ at the utterance level (Figure 2). The new aggregated loss is:

$$\text{L} = \text{L}_{\text{IC}} + \alpha \times \text{L}_{\text{SL}} + \beta \times \text{L}_{\text{Sec\_IC}} \qquad (2)$$

**IC:** At turn $i$, we take the output of context fusion layer $\mathbf{cf}_i$, pass it through a fully connected layer and concatenate the output with the current utterance encoding $\mathbf{h}(\text{Utt}_i)$. This is then further projected down using a fully connected layer ($\text{FC}_i$) and finally fed into *softmax* layer to predict the intent.

**SL:** For turn $i$, the t2t attention output is first fused with the utterance embedding using a fusion gate (Hochreiter and Schmidhuber, 1997) to generate $\mathbf{h}_{ij}$ where $j$ represents token index in the utterance. Then, for each token position in the utterance, we apply a sliding window, $w$ (=3) over neighboring words that transforms each token's embedding space from $\mathbf{h}_{ij}$ to $w \times \mathbf{h}_{ij}$ (not shown in the Figure 2). To add contextual information to SL task, each token's dimension is augmented using slot history ($\text{SL\_hist}_i$) as well as penultimate fully connected layer for IC task ($\text{FC}_i$), yielding a final dimension of $w \times \mathbf{h}_{ij} + \mathbf{h}(\text{SL\_hist}_i) + \mathbf{h}(\text{FC}_i)$. Finally, a Gated Recurrent Unit (GRU) renders the labels auto-regressive followed by *softmax* layer.

## 4 Experiments

**Datasets:** Since there are no existing public datasets for contextual IC and SL task, we use two in-house datasets for evaluation - **Booking** dataset,[3] which is a variation of DSTC-2 dataset (Williams et al., 2014) with intent, slot and dialog act annotations, and **Cable** dataset, a synthetically created conversational dataset. The Booking dataset contains 9,351 training utterances (2,200 conversations) and 6,727 test utterances, with 19 intents and 5 slot types. Cable dataset comprises 1856 training utterances, 1,814 validation utterances and 1,836 test utterances, with 21 intents and 26 slot types.[4] In addition, we also evaluate the model on non-contextual IC-SL public datasets - ATIS (Hemphill et al., 1990) and SNIPS (Coucke et al., 2018).

**Experimental setup:** To emphasize the importance of contextual signals in modeling, we first devise a non-contextual baseline of our CASA-NLU model, NC-NLU. It is similar to CASA-

[3] Dataset will be released to the public

[4] Detailed data stats provided in Appendix B

| Model | ATIS | SNIPS |
|---|---|---|
| RNN-LSTM* (Hakkani-Tür et al., 2016) | 92.6 | 96.9 |
| Atten.-BiRNN* (Liu and Lane, 2016b) | 91.1 | 96.7 |
| LSTM+attn+gates (Goo et al., 2018) | 94.1 | 97.0 |
| Capsules Neural Network (Zhang et al., 2018) | 95.0 | 97.7 |
| NC-NLU (ours) | **95.4** | **98.4** |

Table 1: Average IC accuracy scores (%) on non-contextual datasets. *: as reported in (Goo et al., 2018)

NLU in terms of model architecture except no contextual signals are used. Since neither datasets nor implementation details were shared in previous works on contextual NLU (Bhargava et al., 2013; Shi et al., 2015), we also implement another baseline, CGRU-NLU that uses GRU (Cho et al., 2014) instead of self-attention for temporal context fusion. For fair comparison to existing non-contextual baselines, no pre-trained embeddings are used in any of the experiments though the model design can easily benefit from pre-training.

## 5 Results and Analysis

Table 1 shows the performance of our non-contextual model on two datasets, ATIS and SNIPS. As shown, we obtain a **new SOTA** for IC on both the datasets.[5] We hypothesize that the high performance is due to the utterance-level position-aware multi-dimensional self-attention.

As shown in Table 2, CASA-NLU model outperforms non-contextual NLU on both the Booking and Cable datasets. Further, CASA-NLU model significantly **outperforms** CGRU-NLU on the Cable dataset by 7.26% on IC accuracy and 5.31% on SL F1 absolute, respectively. We believe the reason for strong performance yielded by the CASA-NLU model is due to its multi-dimensional nature, where we learn different weights for different dimensions within the context feature vector $\mathbf{T}_i$. This enables the model to learn different attention distributions for different contextual signals leading to more robust modeling compared to CGRU-NLU model. Table 3 gives further breakdown of the results by showing performance on first vs. follow-up turns in a dialogue. For the more challenging follow-up turns, CASA-NLU yields significant gains over the baseline IC performance.

Table 4 shows impact of some of the contextual signals on model performance for the Booking validation dataset. As expected, contextual signals improve IC and SL performance (Configs - II-V). We observe that adding intent history ($I_{hist}$) leads to highest gains in IC accuracy (Config - IV). At the same time, we see that slot history ($SL_{hist}$) has minimal impact on SL performance for this dataset. Exhaustive experiments showed that the choice of contextual signals is dependent upon the dataset. Our model facilitates switching these contextual signals on or off easily.

| Model | Booking | | Cable | |
|---|---|---|---|---|
| | IC | SL | IC | SL |
| NC-NLU | 91.11 | 88.21 | 44.98 | 27.34 |
| CGRU-NLU | 94.86 | 88.47 | 66.68 | 51.58 |
| CASA-NLU | **95.16** | **88.80** | **73.94** | **56.97** |

Table 2: IC accuracy and SL F1 scores (%) for the three models NC-NLU, CGRU-NLU, CASA-NLU on the 2 contextual datasets - Booking and Cable.

| Model | Booking | | Cable | |
|---|---|---|---|---|
| | Ft | FU | Ft | FU |
| NC-NLU | 95.67 | 89.51 | 40.35 | 48.33 |
| CGRU-NLU | 98.93 | 94.24 | 46.6 | 72.22 |
| CASA-NLU | **99.70** | **94.45** | **47.44** | **81.25** |

Table 3: IC accuracy scores (%) on first (Ft) and follow-up (FU) turns in contextual datasets - Booking and Cable.

| Config | $I_{hist}$ | $SL_{hist}$ | $Utt_{hist}$ | $DA_{hist}$ | IC | SL |
|---|---|---|---|---|---|---|
| I | x | x | x | x | 89.44 | 97.62 |
| II | x | x | ✓ | x | 92.52 | 98.09 |
| III | x | ✓ | ✓ | ✓ | 92.35 | 98.33 |
| IV | ✓ | ✓ | ✓ | x | 95.27 | 98.34 |
| V | ✓ | ✓ | ✓ | ✓ | 96.53 | 98.25 |

Table 4: Impact of contextual signals on IC accuracy and SL F1 scores (%) on Booking validation set for CASA-NLU

**Qualitative Analysis:** Using example conversation in Figure 1, we highlight the relevance of contextual information in making intent predictions by visualizing attention weights $\mathbf{P}(\mathbf{T})$ (Equation 1) for different contextual signals as shown in Figure 3.[6] At user turn 2, the *intent* is switched to *UpgradeService* which the model successfully interprets by paying less attention to previous intents. At turn 3, however, contextual information is critical as user responds to elicitation by agent and hence model emphasizes on last utterance and intent rather than the current or other previous turns.

[5] Since we compute token-level F1, SL performance is not compared to results reported in previous work

[6] For each context signal, attention weights are averaged across all its feature dimensions in $\mathbf{P}(\mathbf{T})$

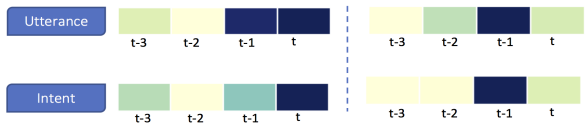


Figure 3: **Visualization of attention weights** given to two different context signals - previous utterances (top) and previous intents (bottom) for turns t=2 (left) and t=3 (right) from Figure 1; darker colors reflect higher attention weights.

## 6 Conclusion

We proposed CASA-NLU model that uses a variable context and various contextual signals in addition to the current utterance to predict the intent and slot labels for the current turn. CASA-NLU achieves gains of over 7% on IC accuracy for Cable dataset over CGRU-NLU baseline, and almost 29% over non-contextual version. This highlights the importance of using contextual information, meanwhile showing that, learning correct attention is also vital for NLU systems.

## 7 Implementation Details

We use hidden layer size of 56 with dropout probability of 0.3. Context history window $K$ was varied from 1 to 5 and the optimal value of 3 was selected. Word embeddings are trained from scratch using an embedding layer size of 56. Adam (Kingma and Ba, 2014) algorithm with initial learning rate of 0.01 gave the optimal performance. Concatenation window size $w$ of 3 is used. $\alpha$ and $\beta$ in loss objective are set to 0.9. Early stopping is used with patience of 10 and threshold of 0.5. Each model is trained for 3 seeds and scores averaged across the seeds are reported.

## 8 Acknowledgements

The authors would like to thank the entire AWS Lex Science team for having insightful discussions and providing feedback with experiments. The authors would also like to express their gratitude to Yi Zhang for his generous help and suggestions for this work.

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Núria Bertomeu, Hans Uszkoreit, Anette Frank, Hans-Ulrich Krieger, and Brigitte Jörg. 2006. Contextual phenomena and thematic relations in database qa dialogues: results from a wizard-of-oz experiment. In *Proceedings of the Interactive Question Answering Workshop at HLT-NAACL 2006*, pages 1–8.

A. Bhargava, Asli elikyilmaz, Dilek Z. Hakkani-Tür, and Ruhi Sarikaya. 2013. Easy contextual intent prediction and slot detection. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8337–8341.

Antoine Bordes and Jason Weston. 2016. Learning end-to-end goal-oriented dialog. *CoRR*, abs/1605.07683.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, et al. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190*.

Nils Dahlbäck and Arne Jönsson. 1989. Empirical studies of discourse representations for natural language interfaces. In *Proceedings of the fourth conference on European chapter of the Association for Computational Linguistics*, pages 291–298. Association for Computational Linguistics.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

Chih-Wen Goo, Guang Gao, Yun-Kai Hsu, Chih-Li Huo, Tsung-Chieh Chen, Keng-Wei Hsu, and Yun-Nung Chen. 2018. Slot-gated modeling for joint slot filling and intent prediction. In *NAACL-HLT*.

Daniel Guo, Gökhan Tür, Wen tau Yih, and Geoffrey Zweig. 2014. Joint semantic utterance classification and slot filling with recursive neural networks. *2014 IEEE Spoken Language Technology Workshop (SLT)*, pages 554–559.

Dilek Hakkani-Tür, Gökhan Tür, Asli Celikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In *Interspeech*, pages 715–719.

C.T. Hemphill, J.J. Godfrey, and G.R. Doddington. 1990. The ATIS spoken language systems pilot corpus. In *Proceedings of the DARPA Speech and Natural Language Workshop*, page 96101.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Minwoo Jeong and Gary Geunbae Lee. 2008. Triangular-chain conditional random fields. *IEEE Transactions on Audio, Speech, and Language Processing*, 16:1287–1302.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Changliang Li, Liang Li, and Ji Qi. 2018a. A self-attentive model with gate mechanism for spoken language understanding. In *EMNLP*.

Changliang Li, Liang Li, and Ji Qi. 2018b. A self-attentive model with gate mechanism for spoken language understanding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3824–3833.

Bing Liu and Ian Lane. 2016a. Attention-based recurrent neural network models for joint intent detection and slot filling. *ArXiv*, abs/1609.01454.

Bing Liu and Ian Lane. 2016b. Attention-based recurrent neural network models for joint intent detection and slot filling. In *INTERSPEECH*.

Bing Liu and Ian Lane. 2016c. Joint online spoken language understanding and language modeling with recurrent neural networks. *ArXiv*, abs/1609.01462.

Mingbo Ma, Kai Zhao, Liang Huang, Bing Xiang, and Bowen Zhou. 2017. Jointly trained sequential labeling and classification by sparse attention neural networks. *ArXiv*, abs/1709.10191.

Christian Raymond and Giuseppe Riccardi. 2007. Generative and discriminative algorithms for spoken language understanding. In *INTERSPEECH*.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2017. Disan: Directional self-attention network for rnn/cnn-free language understanding. *arXiv preprint arXiv:1709.04696*.

Yangyang Shi, Kaisheng Yao, Hu Chen, Yi-Cheng Pan, Mei-Yuh Hwang, and Baolin Peng. 2015. Contextual spoken language understanding using recurrent neural networks. *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5271–5275.

Ye-Yi Wang1, Alex Acero, Ciprian Chelba, Brendan Frey, and Leon Wong. 2002. Combination of statistical and rule-based approaches for spoken language understanding. In *Seventh International Conference on Spoken Language Processing*.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Lina Maria Rojas-Barahona, Pei hao Su, Stefan Ultes, David Vandyke, and Steve J. Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *EACL*.

Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. 2013. The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 404–413.

Jason D Williams, Matthew Henderson, Antoine Raux, Blaise Thomson, Alan Black, and Deepak Ramachandran. 2014. The dialog state tracking challenge series. *AI Magazine*, 35(4):121–124.

Kaisheng Yao, Baolin Peng, Yu Zhang, Dong Yu, Geoffrey Zweig, and Yangyang Shi. 2014a. Spoken language understanding using long short-term memory neural networks. *2014 IEEE Spoken Language Technology Workshop (SLT)*, pages 189–194.

Kaisheng Yao, Baolin Peng, Geoffrey Zweig, Dong Yu, Xiaolong Li, and Feng Gao. 2014b. Recurrent conditional random field for language understanding. *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4077–4081.

Chenwei Zhang, Yaliang Li, Nan Du, Wei Fan, and Philip S Yu. 2018. Joint slot filling and intent detection via capsule neural networks. *arXiv preprint arXiv:1812.09471*.

Xiaodong Zhang and Houfeng Wang. 2016. A joint model of intent determination and slot filling for spoken language understanding. In *IJCAI*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2018. Global-locally self-attentive dialogue state tracker. *CoRR*, abs/1805.09655.

# Sampling Matters! An Empirical Study of Negative Sampling Strategies for Learning of Matching Models in Retrieval-based Dialogue Systems

**Jia Li**[1], **Chongyang Tao**[1], **Wei Wu**[2], **Yansong Feng**[1], **Dongyan Zhao**[1,3] and **Rui Yan**[1,3*]
[1]Wangxuan Institute of Computer Technology, Peking University, Beijing, China
[2]Microsoft Corporation, Beijing, China
[3]Center for Data Science, Peking University, Beijing, China
[1,3]{lijiaa,chongyangtao,fengyansong,zhaody,ruiyan}@pku.edu.cn
[2]{wuwei}@microsoft.com

## Abstract

We study how to sample negative examples to automatically construct a training set for effective model learning in retrieval-based dialogue systems. Following an idea of dynamically adapting negative examples to matching models in learning, we consider four strategies including minimum sampling, maximum sampling, semi-hard sampling, and decay-hard sampling. Empirical studies on two benchmarks with three matching models indicate that compared with the widely used random sampling strategy, although the first two strategies lead to performance drop, the latter two ones can bring consistent improvement to the performance of all the models on both benchmarks.

## 1 Introduction

In this work, we study the problem of response selection as an approach to implementing a retrieval-based dialogue system (Ji et al., 2014; Wang et al., 2013). A key step in response selection is measuring the matching degree between a conversation context and a response candidate. Existing studies focus on constructing a matching model with sophisticated neural architectures (Lowe et al., 2015; Zhou et al., 2016; Yan et al., 2016; Wu et al., 2017; Zhang et al., 2018; Zhou et al., 2018; Tao et al., 2019), but pay little attention to how to effectively learn such architectures from data. On the one hand, it is well known that learning of complicated neural architectures requires large-scale high quality training data; on the other hand, since human labeling is expensive and exhausting, most of the existing work just adopts a simple heuristic to automatically build a training set where human responses are treated as positive examples and negative response candidates are randomly sampled. Such a training set might contain many false negatives and trivial true negatives that are very easy to distinguish from those true positives. As a result, models with advanced architectures can only reach sub-optimal performance after learning (Wu et al., 2018).

In this paper, instead of configuring new architectures, we investigate how to improve the performance of existing matching models with a better learning method. A learning method usually involves choice of loss functions and construction of training data, and we are particularly interested in automatic training data construction, as data are often more crucial to the performance of models. The key problem in training data construction lies in how to properly choose negative examples, and our idea is that negative examples should adapt to the matching models at different learning stages. Following this idea, we consider four negative sampling strategies, namely minimum sampling, maximum sampling, semi-hard sampling, and decay-hard sampling. In the first two strategies, a response candidate that corresponds to the minimal or the maximal matching score at the current step is picked from a pool as a negative example for the next step; and in the latter two strategies, we select negative examples by considering how hard they are to the current matching models. The semi-hard sampling prefers candidates with moderate difficulty to avoid both false negatives and trivial true negatives, and the decay-hard sampling gradually increases the difficulty of negative samples with the training process going on.

We compare different sampling strategies with three matching models in different levels of complexity on two benchmarks. Evaluation results indicate that minimum sampling and maximum sampling are inferior to randomly sampling, and both semi-hard sampling and decay-hard sampling can bring consistent improvement to the perfor-

*Corresponding author: Rui Yan (ruiyan@pku.edu.cn).



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1291–1296, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

mance of all the three models on both data sets.

Our contributions include (1) a systematic comparison of different sampling strategies with two benchmarks; and (2) proposal of semi-hard and decay-hard negative sampling strategies that can generally improve the performance of existing matching models on benchmarks.

## 2 Learning a Matching Model for Response Selection

Suppose that $\mathcal{D} = \{(c_i, \{r_{i,j}^+\}_{j=1}^{n_i^+}, \{r_{i,k}^-\}_{k=1}^{n_i^-})\}_{i=1}^N$ is a training set, where $c_i$ is a conversation context, $\forall j \in \{1, \ldots, n_i^+\}$, $r_{i,j}^+$ is a positive response candidate that properly replies to $c_i$, and $\forall k \in \{1, \ldots, n_i^-\}$, $r_{i,k}^-$ is a negative response candidate that is used to indicate errors in responding to a model, then the learning problem of response selection is to estimate a matching model $g(\cdot, \cdot)$ from $\mathcal{D}$, which can be formulated as

$$\operatorname*{argmin}_{\Theta} \sum_{i=1}^{N} \Big[ \sum_{j=1}^{n_i^+} L(+1, g(c_i, r_{i,j}^+)) + \sum_{k=1}^{n_i^-} L(-1, g(c_i, r_{i,k}^-)) \Big], \tag{1}$$

where $\Theta$ are the parameters of $g(\cdot, \cdot)$, and $L(\cdot, \cdot)$ is a loss function.

In practice, $L(\cdot, \cdot)$ is usually set as cross entropy, then the remaining problems become (1) how to define $g(\cdot, \cdot)$; and (2) how to construct $\mathcal{D}$ given that large-scale human labeling is infeasible. Existing work has paid enough effort to solving Problem (1), but only adopts a simple heuristic for Problem (2) where human responses are treated as $\{r_{i,j}^+\}_{j=1}^{n_i^+}$ (a common case is $n_i^+ = 1$ since only one response is available for a specific context), and some randomly sampled responses are utilized as $\{r_{i,k}^-\}_{k=1}^{n_i^-}$. The problem with this heuristic is that there is no guarantee on what responses will be sampled as $\{r_{i,k}^-\}_{k=1}^{n_i^-}$: some of them could be false negatives, and some could be too trivial to recognize. The clear drawback of random sampling motivates us to pursue better negative sampling strategies in training data construction, as will be elaborated in the next section.

## 3 Model Adaptive Negative Sampling

Our idea is to dynamically adapt negative examples to matching models in learning. The idea is inspired by how human learn knowledge: they adjust their learning materials according to their learning progress. Based on this idea, we consider four strategies to sample negative examples:

**Minimum sampling:** the strategy used to be exploited in answer selection (Rao et al., 2016), and here we apply it to response selection for open domain dialogue systems. Suppose that $\hat{g}(\cdot, \cdot)$ is a matching model obtained from the $t$-th mini-batch, then in the $(t+1)$-th mini-bach, we try to select the easiest negative example for a context $c$ according to $\hat{g}(\cdot, \cdot)$, which can be formulated as

$$\operatorname*{argmin}_{r^- \in \mathcal{R}^-} \hat{g}(c, r^-), \tag{2}$$

where $\mathcal{R}^-$ is a pool of negative examples for $c$.

**Maximum sampling:** similar to minimum sampling, the strategy is also borrowed from answer selection (Rao et al., 2016), but attempts to select the hardest negative example by

$$\operatorname*{argmax}_{r^- \in \mathcal{R}^-} \hat{g}(c, r^-), \tag{3}$$

**Semi-hard sampling:** the first two strategies might be too aggressive, as the easiest negative example could bring no new information to $\hat{g}(\cdot, \cdot)$, and the hardest one could be a false negative. To avoid both cases, we propose a semi-hard sampling strategy which selects a negative sample with moderate difficulty. Formally, the strategy is defined as

$$\operatorname*{argmin}_{r^- \in \mathcal{R}^-} |\hat{g}(c, r^+) - \hat{g}(c, r^-) - \alpha|, \tag{4}$$

where $r^+$ is the positive response candidate of $c$[1], and $\alpha$ is a constant. In Equation (4), we exploit $\alpha$ as a margin to control the distance of matching degree between the selected $r^-$ and $r^+$. The one with a matching degree closest to $\hat{g}(c, r^+) - \alpha$ is picked as a negative example.

**Decay-hard sampling:** to imitate the behavior that human gradually increase the difficulty of their learning materials at different stages, we propose a decay-hard sampling strategy which decays the margin in Equation (4) with the training process going on. Specifically, we consider two methods, namely exponential decay and linear decay.

[1] We assume that only one positive example is available as a common case, otherwise $\hat{g}(c, r^+)$ is replaced with $\min_{r^+ \in \mathcal{R}^+} \hat{g}(c, r^+)$.

In the first method, the margin shrinks in a exponential speed. In the $t$-th mini-bach, the margin $\alpha_t$ is defined by

$$\alpha_t = \varphi \times \exp(\omega \times t), \quad (5)$$

where $0 < \varphi < 1$ and $-1 < \omega < 0$ are parameters. In the second method, the margin linearly becomes small with the training steps, which is given by

$$\alpha_t = \lambda \times t + \theta, \quad (6)$$

where $0 < \theta < 1$ and , $-1 < \lambda < 0$ are parameters. We carefully choose $\theta$ and $\lambda$ to make sure that $\alpha_t > 0$, $\forall t \in \{1, \ldots, T\}$, where $T$ refers to the maximum number of iterations.

Note that descriptions above assume that for each context, only one negative example is selected from the pool. This is for a fair comparison with random sampling on benchmarks, as most of the existing work (Wu et al., 2017; Zhou et al., 2018) utilizes one negative example per context in training. It is easy to extend the strategies to sample multiple negative examples (e.g., by picking the top $l$ examples with matching scores closest to $\hat{g}(c, r^+) - \alpha$ in semi-hard sampling).

## 4 Experiments

We compare different sampling strategies on two benchmarks.

### 4.1 Experimental Setup

The first data set we use is the Ubuntu Dialogue Corpus (Lowe et al., 2015) collected from chat logs of the Ubuntu Forum. We use the version provided by Xu et al. (2017). The data contains 1 million context-response pairs for training, and 0.5 million pairs for validation and test.

Following (Lowe et al., 2015), we employ recall at position $k$ in $n$ candidates ($R_n$@k) as evaluation metrics.

Besides the Ubuntu data, we also choose the Douban Conversation Corpus (Wu et al., 2017) as an experimental data set. The data consists of multi-turn Chinese conversations collected from Douban group[2]. There are 1 million context-response pairs for training, 50 thousand pairs for validation, and 6,670 pairs for test.

Following (Wu et al., 2017), we employ $R_n$@ks, mean average precision (MAP), mean reciprocal rank (MRR)(Voorhees et al., 1999) and precision at position 1 (P@1) as evaluation metrics.

[2] https://www.douban.com/group

### 4.2 Matching Models

The following matching models are selected:

**Dual-LSTM** (Lowe et al., 2015): the model individually encodes a context and a response candidate with LSTMs, and then calculates a matching score based on the final states of the two LSTMs.

**SMN** (Wu et al., 2017): the model lets each utterance in a context interact with a response, and forms the interaction matrices into a matching vector with CNN. The matching vectors are finally accumulated with an RNN as a matching score.

**DAM** (Zhou et al., 2018): the model performs matching in a similar manner as SMN but represents a context and a response candidate with stacked self-attention and cross-attention.

In terms of both complexity and performance under random sampling, Dual-LSTM<SMN<DAM. Regarding to baselines, we consider two random sampling strategies. The first one is a static strategy where negative examples are fixed in the entire learning procedure. This is how existing work learns a matching model with the data described in Section 4.1, and we denote the model as Model-Base. The second one is a dynamic strategy where in each mini-batch, a negative example is randomly sampled from $\mathcal{R}^-$. This is a simplification of model adaptive sampling strategies, and we denote a model learned with this strategy as Model-Rand. We denote a model trained with minimum sampling, maximum sampling, semi-hard sampling, exponential decay-hard sampling, and linear decay-hard sampling as Model-Min, Model-Max, Model-Semi, Model-EDecay, and Model-LDecay respectively. All models are implemented with TensorFlow and tuned on the validation sets. We make sure that Model-Base achieves the performance on both data sets as that reported in (Zhou et al., 2018).

### 4.3 Implementation Details

For static random sampling, we just use the published training sets of both data. For the remaining sampling strategies, we randomly sample 10 responses[3] for each context from the training sets as a pool of negative examples at each epoch. Every time, one response is sampled from the pool as a negative example for a context. Models trained with different sampling strategies are

[3] We set the size of $\mathcal{R}^-$ as 10 to balance efficacy and efficiency.

| Strategies \ Metrics | Ubuntu Corpus | | | | Douban Conversation Corpus | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $R_2@1$ | $R_{10}@1$ | $R_{10}@2$ | $R_{10}@5$ | MAP | MRR | P@1 | $R_{10}@1$ | $R_{10}@2$ | $R_{10}@5$ |
| Dual-LSTM-Base | 0.901 | 0.638 | 0.784 | 0.949 | 0.485 | 0.527 | 0.320 | 0.187 | 0.343 | 0.720 |
| Dual-LSTM-Rand | 0.909 | 0.691 | 0.818 | 0.952 | 0.541 | 0.581 | 0.395 | 0.236 | 0.396 | 0.742 |
| Dual-LSTM-Min | 0.832 | 0.561 | 0.640 | 0.883 | 0.446 | 0.495 | 0.298 | 0.164 | 0.312 | 0.664 |
| Dual-LSTM-Max | 0.839 | 0.579 | 0.672 | 0.898 | 0.477 | 0.510 | 0.300 | 0.178 | 0.323 | 0.689 |
| Dual-LSTM-LDecay | 0.915 | 0.701 | 0.830 | 0.955 | 0.543 | 0.586 | 0.405 | 0.242 | 0.402 | 0.749 |
| Dual-LSTM-EDecay | 0.916 | 0.703 | 0.833 | 0.957 | 0.544 | 0.588 | 0.406 | 0.245 | 0.403 | 0.752 |
| Dual-LSTM-Semi | **0.918** | **0.706** | **0.835** | **0.958** | **0.546** | **0.591** | **0.408** | **0.246** | **0.405** | **0.754** |
| SMN-Base | 0.926 | 0.726 | 0.847 | 0.961 | 0.529 | 0.569 | 0.397 | 0.233 | 0.396 | 0.724 |
| SMN-Rand | 0.931 | 0.753 | 0.859 | 0.963 | 0.543 | 0.587 | 0.406 | 0.240 | 0.407 | 0.751 |
| SMN-Min | 0.850 | 0.569 | 0.714 | 0.920 | 0.521 | 0.563 | 0.384 | 0.229 | 0.387 | 0.718 |
| SMN-Max | 0.859 | 0.667 | 0.789 | 0.944 | 0.523 | 0.565 | 0.388 | 0.227 | 0.392 | 0.721 |
| SMN-LDecay | 0.933 | 0.759 | 0.861 | 0.965 | 0.549 | 0.600 | 0.421 | 0.253 | 0.410 | 0.755 |
| SMN-EDecay | 0.933 | 0.760 | 0.862 | 0.966 | 0.552 | 0.602 | 0.424 | 0.261 | 0.412 | 0.758 |
| SMN-Semi | **0.934** | **0.762** | **0.865** | **0.967** | **0.554** | **0.605** | **0.425** | **0.253** | **0.412** | **0.759** |
| DAM-Base | 0.938 | 0.767 | 0.874 | 0.969 | 0.550 | 0.601 | 0.427 | 0.254 | 0.410 | 0.757 |
| DAM-Rand | 0.940 | 0.777 | 0.878 | 0.971 | 0.563 | 0.612 | 0.436 | 0.261 | 0.427 | 0.783 |
| DAM-Min | 0.923 | 0.721 | 0.841 | 0.958 | 0.539 | 0.585 | 0.408 | 0.247 | 0.407 | 0.744 |
| DAM-Max | 0.928 | 0.736 | 0.852 | 0.962 | 0.551 | 0.596 | 0.421 | 0.258 | 0.411 | 0.754 |
| DAM-LDecay | 0.942 | 0.784 | 0.880 | 0.973 | 0.573 | 0.617 | 0.442 | 0.270 | 0.437 | 0.789 |
| DAM-EDecay | 0.943 | 0.784 | 0.882 | 0.973 | 0.575 | 0.621 | 0.444 | 0.272 | 0.439 | 0.793 |
| DAM-Semi | **0.944** | **0.785** | **0.883** | **0.974** | **0.580** | **0.623** | **0.450** | **0.279** | **0.443** | **0.796** |

Table 1: Evaluation results of different sampling strategies on the two data sets. Numbers in bold indicate the best strategies for the corresponding models on specific metrics.

tuned with the same validation sets and evaluated with the same test sets (i.e., the original released sets). We vary $\alpha$ in semi-hard sampling in $\{0.01, 0.03, 0.05, 0.07, 0.09, 0.1, 0.3, 0.5\}$, and choose $0.07$. In decay-hard sampling, we set $\omega$, $\varphi$, $\lambda$, and $\theta$ as $-1.5e-5$, $0.1$, $-8.75e-7$, and $0.1$ on respectively.

### 4.4 Evaluation Results

Table 1 reports evaluation results on the two data sets. We can see that both semi-hard sampling and decay-hard sampling can generally improve the three matching models on both data sets. Minimum sampling and maximum sampling are consistently worse than random sampling, which verified the statement we make in Section 3 that the two strategies are too aggressive. Decay-hard sampling is a little worse than semi-hard sampling. The reason might be that false negatives are introduced to learning by decay-hard at late stages of training. Dynamic random sampling is better than static random sampling, because models can leverage more negative examples from the pool. It is worth noting that the proposed sampling strategies does not change the elapsed time for prediction, despite a little more training time. On the other hand, we do see improvement on the two data sets. Therefore, we believe it is worth paying a little more training time but obtaining the improvement.

Besides the comparison of different sampling strategies, we are also interested in how the hyper-parameter $\alpha$ affects the performance of semi-hard sampling. Figure 1 shows how the performance of SMN and DAM changes with respect to different margins (i.e., $\alpha$). We observe a similar trend for both models: they first increase monotonically until the margin reaches $0.07$, and then drop as the margin increases. Particularly, the performance of both models with the semi-hard sampling strategy is worse than that with the random sampling strategy when the margin reaches $0.5$. The reason behind the phenomenon is that when the margin is small, semi-hard sampling is similar to maximum sampling and will introduce false negatives into learning, while when the margin is large, semi-hard sampling is like minimum sampling and is prone to provide trivial samples to learning.

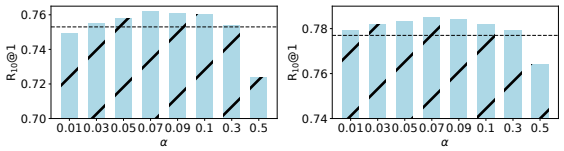


(a) SMN (b) DAM

Figure 1: The effect of $\alpha$ to semi-hard sampling on the Ubuntu data. Dashed lines represent performance of the dynamic random sampling strategy.

## 5 Related works

Negative sampling strategies have been studied in many machine learning tasks. In the computer vision fields, Faghri et al. (2017) studies hard negatives and introduces a simple change to common loss function on image-caption retrieval tasks. Guo et al. (2018) proposes a fast negative sampler which chooses negative examples that are most likely to meet the requirements of violation according to the latent factors of image. In natural language processing fields, Kotnis and Nastase (2017) analyses the impact of negative sampling strategies on the performance of link prediction in knowledge graphs. Saeidi et al. (2017) studies the affect of a tailored sample strategy on the performance of document retrieval task. Rao et al. (2016) uses three negative strategies to select the most informative negative samples on the pairwise ranking model for answer selection. Xu et al. (2015) introduces a straightforward negative sampling strategy to improve the assignment of subjects and objects on a convolution neural network. To our best knowledge, this is the first work to empirical study of negative sampling strategies for learning of matching models in multi-turn retrieval-based dialogue systems, which may enlighten future works in the learning of retrieval-based dialogue systems.

## 6 Conclusions

We present minimum sampling, maximum sampling, semi-hard sampling, and decay-hard sampling as four model adaptive negative sampling strategies to learn a matching model for retrieval-based dialogue systems. Evaluation results with three models on two benchmarks indicate that although minimum sampling and maximum sampling are worse than random sampling, both semi-hard sampling and decay-hard sampling can generally improve the performance of the models on both data sets. In the future, we would like to extend our negative sampling strategies to other tasks.

## Acknowledgments

We thank the reviewers for their valuable comments. This work was supported by the National Key Research and Development Program of China (No. 2017YFC0804001), the National Science Foundation of China (NSFC No. 61876196, NSFC No. 61828302, and NSFC No. 61672058).

## References

Fartash Faghri, David J Fleet, Jamie Ryan Kiros, and Sanja Fidler. 2017. Vse++: Improving visual-semantic embeddings with hard negatives. *arXiv preprint arXiv:1707.05612*.

Guibing Guo, Songlin Zhai, Fajie Yuan, Yuan Liu, and Xingwei Wang. 2018. Vse-ens: Visual-semantic embeddings with efficient negative sampling. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Zongcheng Ji, Zhengdong Lu, and Hang Li. 2014. An information retrieval approach to short text conversation. *arXiv preprint arXiv:1408.6988*.

Bhushan Kotnis and Vivi Nastase. 2017. Analysis of the impact of negative sampling on link prediction in knowledge graphs. *arXiv preprint arXiv:1708.06816*.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 285–294.

Jinfeng Rao, Hua He, and Jimmy Lin. 2016. Noise-contrastive estimation for answer selection with deep neural networks. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 1913–1916. ACM.

Marzieh Saeidi, Ritwik Kulkarni, Theodosia Togia, and Michele Sama. 2017. The effect of negative sampling strategy on capturing semantic similarity in document embeddings. In *Proceedings of the 2nd Workshop on Semantic Deep Learning (SemDeep-2)*, pages 1–8.

Chongyang Tao, Wei Wu, Can Xu, Wenpeng Hu, Dongyan Zhao, and Rui Yan. 2019. One time of interaction may not be enough: Go deep with an interaction-over-interaction network for response selection in dialogues. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1–11, Florence, Italy. Association for Computational Linguistics.

Ellen M Voorhees et al. 1999. The trec-8 question answering track report. In *Trec*, volume 99, pages 77–82. Citeseer.

Hao Wang, Zhengdong Lu, Hang Li, and Enhong Chen. 2013. A dataset for research on short-text conversations. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 935–945.

Yu Wu, Wei Wu, Zhoujun Li, and Ming Zhou. 2018. Learning matching models with weak supervision for response selection in retrieval-based chatbots. *arXiv preprint arXiv:1805.02333*.

Yu Wu, Wei Wu, Chen Xing, Ming Zhou, and Zhoujun Li. 2017. Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 496–505.

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2015. Semantic relation classification via convolutional neural networks with simple negative sampling. *arXiv preprint arXiv:1506.07650*.

Zhen Xu, Bingquan Liu, Baoxun Wang, Chengjie Sun, and Xiaolong Wang. 2017. Incorporating loose-structured knowledge into lstm with recall gate for conversation modeling. In *Proceedings of the 2017 International Joint Conference on Neural Networks*, pages 3506–3513.

Rui Yan, Yiping Song, and Hua Wu. 2016. Learning to respond with deep neural networks for retrieval-based human-computer conversation system. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 55–64. ACM.

Zhuosheng Zhang, Jiangtong Li, Pengfei Zhu, Hai Zhao, and Gongshen Liu. 2018. Modeling multi-turn conversation with deep utterance aggregation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3740–3752. Association for Computational Linguistics.

Xiangyang Zhou, Daxiang Dong, Hua Wu, Shiqi Zhao, Dianhai Yu, Hao Tian, Xuan Liu, and Rui Yan. 2016. Multi-view response selection for human-computer conversation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 372–381.

Xiangyang Zhou, Lu Li, Daxiang Dong, Yi Liu, Ying Chen, Wayne Xin Zhao, Dianhai Yu, and Hua Wu. 2018. Multi-turn response selection for chatbots with deep attention matching network. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1118–1127.

# Zero-shot Cross-lingual Dialogue Systems with Transferable Latent Variables

**Zihan Liu, Jamin Shin, Yan Xu, Genta Indra Winata,**
**Peng Xu, Andrea Madotto, Pascale Fung**
Center for Artificial Intelligence Research (CAiRE)
Department of Electronic and Computer Engineering
The Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong
{zliucr,jay.shin,yxucb,giwinata,pxuab}@connect.ust.hk,
pascale@ece.ust.hk

## Abstract

Despite the surging demands for multilingual task-oriented dialog systems (e.g., Alexa, Google Home), there has been less research done in multilingual or cross-lingual scenarios. Hence, we propose a zero-shot adaptation of task-oriented dialogue system to low-resource languages. To tackle this challenge, we first use a set of very few parallel word pairs to refine the aligned cross-lingual word-level representations. We then employ a latent variable model to cope with the variance of similar sentences across different languages, which is induced by imperfect cross-lingual alignments and inherent differences in languages. Finally, the experimental results show that even though we utilize much less external resources, our model achieves better adaptation performance for natural language understanding task (i.e., the intent detection and slot filling) compared to the current state-of-the-art model in the zero-shot scenario.

## 1 Introduction

Task-oriented dialogue systems have been widely adopted in the industry (e.g., Amazon Alexa, Google Home, Apple Siri, Microsoft Cortana) as a virtual agent to tend to the needs of the users. However, these agents have mostly been trained with the monolingual dataset that is often expensive to build or acquire. In order to cope with the scarcity of low-resource language dialogue data, we are motivated to look into cross-lingual dialogue systems which can adapt with very little or no training data in the target language.

This task of zero-shot adaptation of dialogue systems to different languages is relatively new and has not been explored thoroughly enough yet. The main approach of previous work (Upadhyay et al., 2018; Chen et al., 2018; Schuster et al., 2019) in this task is using aligned cross-lingual word embeddings between source and target languages. However, this method suffers from imperfect alignments between the source and target language embeddings. This can be attributed not only to the noise in aligning two different embeddings, but also to the inherent discrepancies in different languages such as Thai and English which come from entirely different roots. To address such variance in the alignment, we turn to probabilistic modeling with latent variables as it has been successfully used in several recent task-oriented dialogue systems (Wen et al., 2017; Zhao et al., 2017, 2018; Zhao and Eskenazi, 2018; Le et al., 2018).

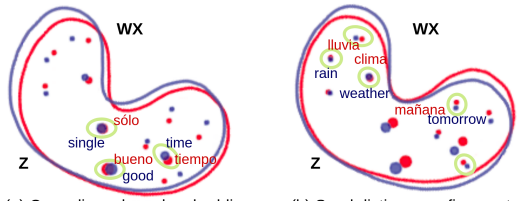


(a) Cross-lingual word embeddings (b) Seed dictionary refinement

Figure 1: (a) The aligned word embeddings. (b) Refined mapping $\mathbf{W}$ with seed dictionary; green circle represents close words in different languages.

However, we notice that naively using latent variables does not help the model improve much in slot filling and intent prediction. We hypothesize that the variance of the cross-lingual word embeddings is too large for the model to learn any meaningful latent variables. Hence, we propose to first refine the cross-lingual embeddings with $\sim$10 seed word-pairs related to the dialogue domains. We then add Gaussian noise (Zheng et al., 2016) to further compensate the imperfect alignment of cross-lingual embeddings.

As a result, a combination of these methods allows us to build a transferable latent variable model that learns the distribution of training language inputs that is invariant to noise in the cross-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1297–1303,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

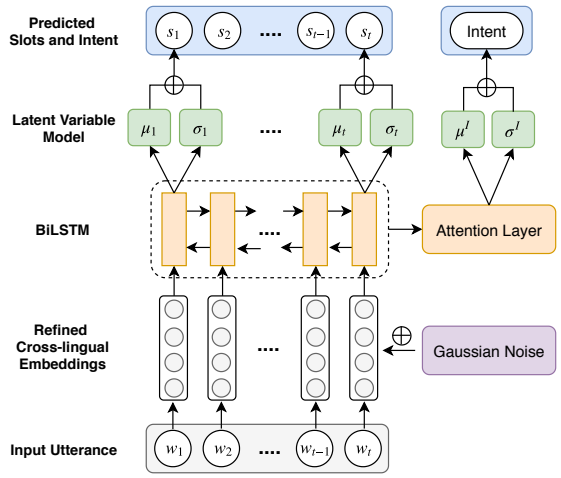


Figure 2: The structure of our proposed model.

lingual word embeddings. This enables our model to capture the variance of semantically similar sentences across different languages, and achieve state-of-the-art results in zero-shot adaptation of English to Spanish and Thai for the natural language understanding task (i.e., the intent prediction and slot filling) on the dataset proposed by Schuster et al. (2019), even though we use much less external resources (i.e., ∼10 seed word-pairs) while others utilize a large amount of bilingual corpus. We further visualize the learned latent variables to confirm that same-meaning words and sentences have similar distributions.

## 2 Related Work

Cross-lingual transfer learning which acts as one of the low-resource topics (Gu et al., 2018; Lee et al., 2019; Liu et al., 2019; Xu et al., 2018) has attracted more and more people recently, followed by the rapid development of cross-lingual word embeddings. Artetxe et al. (2017) proposed a self-learning framework and utilized a small size of word dictionary to learn the mapping between source and target word embeddings. Conneau et al. (2018) leveraged adversarial training to learn a linear mapping from a source to a target space without using parallel data. Joulin et al. (2018) utilized Relaxed CSLS loss to optimize this mapping problem. Winata et al. (2019) introduced a method to leverage cross-lingual meta-representations for code-switching name entity recognition by combining multiple monolingual word embeddings.

Chen et al. (2018) proposed a teacher-student framework leveraging bilingual data for cross-lingual transfer learning in dialogue state tracking. Upadhyay et al. (2018) leveraged joint training and cross-lingual embeddings to do zero-shot and almost zero-shot transfer learning in intent prediction and slot filling. Finally, Schuster et al. (2019) utilizes Multilingual CoVe embeddings obtained from training Machine Translation systems as in (McCann et al., 2017). The main difference of our work with previous work is that our model does not leverage any external bilingual data other than 11 word pairs for embeddings refinement.

## 3 Methodology

Our model consists of a refined cross-lingual embedding layer followed by a BiLSTM (Hochreiter and Schmidhuber, 1997) which parameterizes the Latent Variable Model, as illustrated in Figure 2. We jointly train our model to predict both slots and user intents. We denote $\mathbf{w} = [w_1, \ldots, w_T]$ as the input words and $\mathbf{e} = [e_1, \ldots, e_T]$ as the word embeddings of $\mathbf{w}$. The slot at time-step $t$ is $s_t$, while the intent for each sentence $\mathbf{w}$ is denoted as $I$. Note that only matrices are bold-faced.

### 3.1 Cross-lingual Embeddings Refinement

To further refine the cross-lingual alignments to our task, we draw from the hypothesis that domain-related words are more important than others. Hence, as shown in Figure 1, we propose to refine the cross-lingual word embeddings (Joulin et al., 2018)[1] using very few parallel word pairs, which is obtained by selecting 11 English words related to dialogue domains (weather, alarm, and reminder) and translate them using bilingual lexicons. We refine the embeddings by leveraging the framework proposed in Artetxe et al. (2017).

Let $\mathbf{X}$ and $\mathbf{Z}$ be the aligned cross-lingual word embeddings between two languages. $X_{i*}$ and $Z_{j*}$ are the embeddings for the $i^{th}$ source word and $j^{th}$ target word. We denote a binary dictionary matrix $\mathbf{D}$: $D_{ij} = 1$ if the $i^{th}$ source language word is aligned with the $j^{th}$ target language word and $D_{ij} = 0$ otherwise. The goal is to find the optimal mapping matrix $\mathbf{W}^*$ by minimizing:

$$\mathbf{W}^* = \arg\min_{\mathbf{W}} \sum_{i,j} D_{ij} ||X_{i*}\mathbf{W} - Z_{j*}||^2. \quad (1)$$

Following Artetxe et al. (2016), with orthogonal constraints, mean centering, and length normaliza-

[1]The embeddings are available in https://fasttext.cc

tion, we can maximize the following instead:

$$\mathbf{W}^* = \arg\max_{\mathbf{W}} \text{Tr}(\mathbf{X}\mathbf{W}\mathbf{Z}^{\text{T}}\mathbf{D}^{\text{T}}). \quad (2)$$

We iteratively optimize Equation 2 until distances between domain-related seed words are closer than a certain threshold after refinement. Figure 1 illustrates better alignment for domain-related words after refinement.

### 3.2 Gaussian Noise Injection

To cope with the noise in alignments, we inject Gaussian noise to English embeddings, so the trained model will be more robust to variance. This is a regularization method to improve the generalization ability to the unseen inputs in different languages, particularly languages from different roots such as Thai and Spanish. The final embeddings are $\mathbf{e}^* = [e_1 + N_1, \ldots, e_T + N_T]$, where $\mathbf{N} \sim \mathcal{N}(0, 0.1\mathbf{I})$.

### 3.3 Latent Variable Model (LVM)

Given a near-perfect cross-lingual embedding, there is still noise caused by the inherent discrepancies between source and target languages. This noise amplifies when combined with imperfect alignment, and makes point estimation vulnerable to the small, but not negligible differences across languages. Instead, using latent variables will allow us to model the distribution that captures the variance of semantically similar sentences across different languages. The whole training process is defined as follows:

$$[h_1 ... h_t ... h_T] = \text{BiLSTM}(\mathbf{e}^*), \quad (3)$$

$$m_t = h_t w_a,\ a_t = \frac{exp(m_t)}{\sum_{j=1}^{T} exp(m_j)},\ v = \sum_{t=1}^{T} a_t h_t, \quad (4)$$

$$\begin{bmatrix} \mu_t^S \\ \log(\sigma_t^S)^2 \end{bmatrix} = \mathbf{W}_r^S h_t, \begin{bmatrix} \mu^I \\ \log(\sigma^I)^2 \end{bmatrix} = \mathbf{W}_r^I v, \quad (5)$$

$$z_t^S \sim q_t^S(z|h_t),\ \ z^I \sim q^I(z|v), \quad (6)$$

$$p_t^S(s_t|z_t^S) = \text{Softmax}(\mathbf{W}_g^S z_t^S), \quad (7)$$

$$p^I(I|z^I) = \text{Softmax}(\mathbf{W}_g^I z^I), \quad (8)$$

where attention vector ($v$) is obtained by following Felbo et al. (2017) and $w_a$ is the weight matrix for the attention layer, $\mathbf{W}_{\{r,g\}}^{\{S,I\}}$ are trainable parameters, superscripts S and I refer to slot prediction and intent detection respectively, subscript "r" refers to "recognition" for obtaining the mean and variance vectors while subscript "g" refers to "generation" for predicting the slots and intents, and $q_t^S \sim \mathcal{N}(\mu_t^S, (\sigma_t^S)^2\mathbf{I})$ and $q^I \sim \mathcal{N}(\mu^I, (\sigma^I)^2\mathbf{I})$ are the posterior approximations which we sample our latent vectors $z_t^S$ and $z^I$ from. Finally, $p_t^S$ and $p^I$ are the predictions for the slot of the *t-th* token and the intent of the utterance respectively. The objective functions for slot filling and intent prediction are:

$$\mathcal{L}^I = \mathbb{E}_{z^I}[\log p^I(I|z^I)], \quad (9)$$

$$\mathcal{L}_t^S = \mathbb{E}_{z_t^S}[\log p_t^S(s_t|z_t^S)], \quad (10)$$

$$\mathcal{L}^S = \sum_{t=1}^{T} L_t^S, \quad (11)$$

hence, the final objective function to minimize is,

$$\mathcal{L} = \mathcal{L}^S + \mathcal{L}^I. \quad (12)$$

The model prediction is not deterministic since the latent variables $z_t^S$ and $z^I$ are sampled from the Gaussian distributions. Therefore, in the inference time, we use the true mean $\mu_t^S$ and $\mu^I$ to replace $z_t^S$ and $z^I$ respectively to make the prediction deterministic.

## 4 Experiments

### 4.1 Dataset

We conduct our experiments under the zero-shot scenario of multilingual task-oriented dialogue dataset presented by Schuster et al. (2019). Our model is trained only with the English data and then do a zero-shot test on Spanish and Thai test set. We delexicalize words by replacing the tokens which represent numbers, time (such as am, pm), and duration (such as 30min) with special tokens `<number>`, `<time>`, and `<last>` respectively.

### 4.2 Training Details

In the training procedure, we freeze the word embeddings of the primary language, and then replace them with the corresponding aligned word embeddings of the unseen languages for a zero-shot test. We use bi-directional LSTM model with hidden dimension size of 250, and the latent variable model with both mean and variance in the size of 100. Gaussian noise with zero mean and variance of 0.1 is injected dynamically in different iterations. We use the accuracy to evaluate the performance of intent prediction and the standard

| Model | Spanish | | | | Thai | | | |
|---|---|---|---|---|---|---|---|---|
| | Intent acc. | | Slot F1 | | Intent acc. | | Slot F1 | |
| | LVM | CRF | LVM | CRF | LVM | CRF | LVM | CRF |
| Vanilla BiLSTM | 46.36 | 44.13 | 15.64 | 11.32 | 35.12 | 33.57 | 5.82 | 5.24 |
| + *noise (N)* | 72.97 | 66.95 | 46.56 | 20.27 | 40.37 | 37.53 | 10.66 | 6.51 |
| + *refinement (R)* | 87.69 | 88.23 | 61.63 | 42.62 | 59.40 | 59.28 | 21.84 | 16.53 |
| + *noise & refinement* | 89.21 | 88.79 | 64.04 | 43.98 | 70.81 | 64.48 | 29.54 | 17.46 |
| + *N & R & delexicalization* | **90.20** | 89.98 | **65.79** | 47.70 | **73.43** | 69.62 | **32.24** | 23.11 |
| Zero-shot SLU ‡ | 46.64 | | 15.41 | | 35.64 | | 12.11 | |
| Multi. CoVe w/ auto | 53.89 | | 19.25 | | 70.70 | | 35.62 | |
| Translate Train † | *85.39* | | *72.87* | | *95.85* | | *55.43* | |

Table 1: Results on different models including baseline models, where N refers to the Gaussian noise injection and R refers to the cross-lingual embeddings refinement. ‡ We implemented Upadhyay et al. (2018) model and evaluated with our test set. † Schuster et al. (2019) translated English data to Spanish and Thai with a trained supervised machine translation system and it is considered as our *upper bound* result.

BIO structure to calculate the F1 score for evaluating the performance of slot filling. In the zero-shot cross-lingual adaptation, we simply replace the training language (i.e., English) word embeddings with the cross-lingual target language (i.e., Spanish or Thai) word embeddings. Note that we never use any target language evaluation data to select the model for zero-shot cross-lingual adaptation, instead, we utilize the English validation set and early stop strategy according to the slot F1 score.

### 4.3 Word Pairs

We choose the number of word pairs based on the vocabulary size of the corpus. Intuitively, the larger the vocabulary size is, the more words we need to align across languages, and the more word pairs we need to achieve good performance. We select 11 domain-related words which frequently exist in the English training set. The number of words we select is around 0.25% of the vocabulary size for the English training set. The concrete information of the 11 word pairs is as follows:

The English seed words we selected are *weather, forecast, temperature, rain, hot, cold, remind, forget, alarm, cancel, tomorrow*, which are related to the three dialogue domains (weather, alarm, and reminder). We translate them by leveraging bilingual dictionaries[2]. The corresponding translations in Spanish and Thai are *clima, pronóstico, temperatura, lluvia, caliente, frío, recordar, olvidar, alarma, cancelar, mañana* and อากาศ, พยากรณ์, อุณหภูมิ, ฝน, ร้อน, หนาว, เตือน, ลืม, เตือน, ยกเลิก, พรุ่ง respectively.

[2]https://github.com/facebookresearch/MUSE

| Model | Spanish | | Thai | |
|---|---|---|---|---|
| | Intent | Slot | Intent | Slot |
| Our model | **90.20** | **65.79** | **73.43** | **32.24** |
| - *LVM* | 85.85 | 61.86 | 66.01 | 25.22 |
| - *LVM* + *MLP* | 86.02 | 62.34 | 66.56 | 28.35 |

Table 2: Ablation Study on LVM models. {- *LVM*} means removing LVM and {- *LVM* + *MLP*} means replacing LVM with a Multi-Layer Perceptron which has the same size as the LVM.

### 4.4 Evaluation

We implement and evaluate the following models:

**Zero-shot SLU** Upadhyay et al. (2018) used cross-lingual embeddings (Bojanowski et al., 2017) to do zero-shot transfer learning.

**Conditional Random Fields (CRF)** We reproduce the baseline model in Schuster et al. (2019), and also add embedding noise, cross-lingual refinement, and delexicalization.

**Latent Variable Model (LVM) - Ours** We replace the CRF module with latent variables and also apply it to intent prediction.

Besides, we directly compare with the baseline models illustrated in Schuster et al. (2019):

**Multi. CoVe w/ auto** They combined Multilingual CoVe (Yu et al., 2018) with an auto-encoder objective and then used the trained encoder with the CRF model.

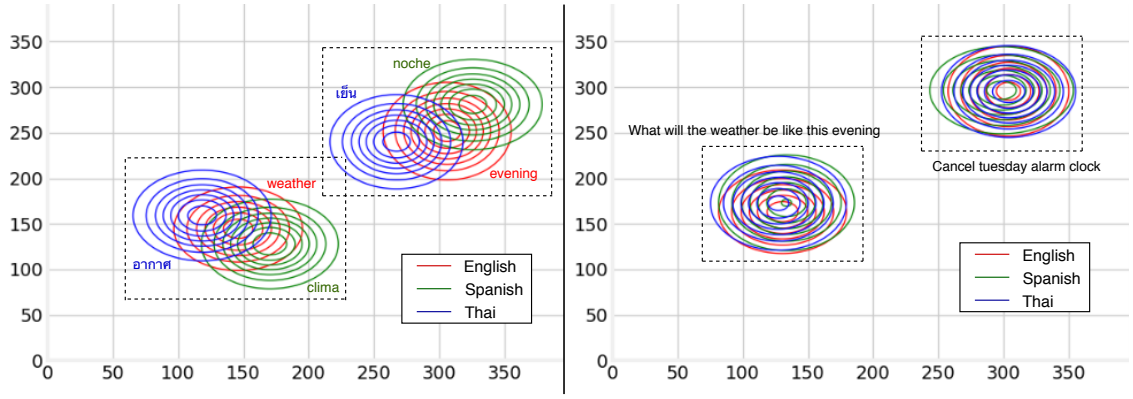


Figure 3: Visualization of latent variables on words **(left)** and sentences **(right)**. **Left:** We choose "weather-clima-อากาศ" and "evening-noche- เย็น" from parallel sentences. *English: "What will the weather be like this evening"*, *Spanish: "Cómo será el clima esta noche"*, *Thai:* "ตอน เย็น นี้ อากาศ จะ เป็น อย่างไร". **Right:** We choose two English sentences and show their distributions and those of the corresponding Spanish and Thai translations.

**Translate Train** They trained a supervised machine translation system to translate English data into the target language and then trained the CRF model on this translated dataset.

## 5 Results & Discussion

From Table 1, in general, LVM outperforms CRF models. This is because for semantically same words (e.g., weather and clima) LVM considers such close enough points as the same distribution, but CRF is more likely to classify them differently. This can be shown very clearly from Figure 3, in which the latent variables demonstrate similar distributions for semantically similar sentences and words. In addition, we can see that adding only Gaussian noise to the Vanilla BiLSTM improves our prediction performance significantly, which implies that the robustness of our model towards the noisy signals which come from the target embedding inputs.

Furthermore, it is clearly visible that cross-lingual embeddings refinement is more effective in Spanish than Thai. This is attributed to the quality of alignments in the two languages. Spanish is much more lexically and grammatically similar to English than Thai, so word-level embedding refinement is reasonably good. Jointly incorporating all three methods (Gaussian noise injection, cross-lingual embeddings refinement, and delexicalization) further reduces the noise in the inputs as well as makes the model more robust to noise, which help LVM to more easily approximate the distribution.

Finally, in Table 2, we ablate the usage of LVM to see whether the boost of performance comes simply from the increase of parameter size. By removing or replacing LVM with MLP, we can see the clear performance gains by using LVM.

## 6 Conclusion

In this paper, we propose a transferable latent variable that focuses on improving the zero-shot cross-lingual adaptation of natural language understanding task to low-resource languages. We show that a combination of 1) cross-lingual embeddings refinement, 2) Gaussian noise injection, and 3) latent variables are effective in coping with the variance of semantically similar sentences across different languages, and the visualizations of the latent variables confirm such. We leverage very few resources (i.e., 11 seed word pairs) and achieve state-of-the-art performance for English-to-Spanish and English-to-Thai in the zero-shot cross-lingual scenario.

## Acknowledgments

This work has been partially funded by ITF/319/16FP and MRP/055/18 of the Innovation Technology Commission, the Hong Kong SAR Government.

## References

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2289–2294.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Wenhu Chen, Jianshu Chen, Yu Su, Xin Wang, Dong Yu, Xifeng Yan, and William Yang Wang. 2018. Xl-nbt: A cross-lingual neural belief tracking framework. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 414–424.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. In *International Conference on Learning Representations (ICLR)*.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1615–1625.

Jiatao Gu, Hany Hassan, Jacob Devlin, and Victor OK Li. 2018. Universal neural machine translation for extremely low resource languages. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 344–354.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Armand Joulin, Piotr Bojanowski, Tomas Mikolov, Hervé Jégou, and Edouard Grave. 2018. Loss in translation: Learning bilingual word mapping with a retrieval criterion. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2979–2984.

Hung Le, Truyen Tran, Thin Nguyen, and Svetha Venkatesh. 2018. Variational memory encoder-decoder. In *Advances in Neural Information Processing Systems*, pages 1508–1518.

Nayeon Lee, Zihan Liu, and Pascale Fung. 2019. Team yeon-zi at semeval-2019 task 4: Hyperpartisan news detection by de-noising weakly-labeled data. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 1052–1056.

Zihan Liu, Yan Xu, Genta Indra Winata, and Pascale Fung. 2019. Incorporating word and subword units in unsupervised machine translation using language model rescoring. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 275–282.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6294–6305.

Sebastian Schuster, Sonal Gupta, Rushin Shah, and Mike Lewis. 2019. Cross-lingual transfer learning for multilingual task oriented dialog. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3795–3805.

Shyam Upadhyay, Manaal Faruqui, Gokhan Tür, Hakkani-Tür Dilek, and Larry Heck. 2018. (almost) zero-shot cross-lingual spoken language understanding. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6034–6038. IEEE.

Tsung-Hsien Wen, Yishu Miao, Phil Blunsom, and Steve Young. 2017. Latent intention dialogue models. In *International Conference on Machine Learning*, pages 3732–3741.

Genta Indra Winata, Zhaojiang Lin, and Pascale Fung. 2019. Learning multilingual meta-embeddings for code-switching named entity recognition. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 181–186.

Peng Xu, Andrea Madotto, Chien-Sheng Wu, Ji Ho Park, and Pascale Fung. 2018. Emo2vec: Learning generalized emotion representation by multi-task training. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 292–298.

Katherine Yu, Haoran Li, and Barlas Oguz. 2018. Multilingual seq2seq training with similarity loss for cross-lingual document classification. In *Proceedings of The Third Workshop on Representation Learning for NLP*, pages 175–179.

Tiancheng Zhao and Maxine Eskenazi. 2018. Zero-shot dialog generation with cross-domain latent actions. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pages 1–10.

Tiancheng Zhao, Kyusong Lee, and Maxine Eskenazi. 2018. Unsupervised discrete sentence representation learning for interpretable neural dialog generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1098–1107.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 654–664.

Stephan Zheng, Yang Song, Thomas Leung, and Ian Goodfellow. 2016. Improving the robustness of deep neural networks via stability training. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

# Modeling Multi-Action Policy for Task-Oriented Dialogues

**Lei Shu, Hu Xu, Bing Liu**
University of Illinois at Chicago
{lshu3, hxu48, liub}@uic.edu

**Piero Molino**
Uber AI
piero@uber.com

## Abstract

Dialogue management (DM) plays a key role in the quality of the interaction with the user in a task-oriented dialogue system. In most existing approaches, the agent predicts only one DM policy action per turn. This significantly limits the expressive power of the conversational agent and introduces unwanted turns of interactions that may challenge users' patience. Longer conversations also lead to more errors and the system needs to be more robust to handle them. In this paper, we compare the performance of several models on the task of predicting multiple acts for each turn. A novel policy model is proposed based on a recurrent cell called gated Continue-Act-Slots (gCAS) that overcomes the limitations of the existing models. Experimental results show that gCAS[1] outperforms other approaches.

## 1 Introduction

In a task-oriented dialogue system, the dialogue manager policy module predicts actions usually in terms of dialogue acts and domain specific slots. It is a crucial component that influences the efficiency (e.g., the conciseness and smoothness) of the communication between the user and the agent. Both supervised learning (SL) (Stent, 2002; Williams et al., 2017a; Williams and Zweig, 2016; Henderson et al., 2005, 2008) and reinforcement learning (RL) approaches (Walker, 2000; Young et al., 2007; Gasic and Young, 2014; Williams et al., 2017b; Su et al., 2017) have been adopted to learn policies. SL learns a policy to predict acts given the dialogue state. Recent work (Wen et al., 2017; Liu and Lane, 2018) also used SL as pre-training for RL to mitigate the sample inefficiency of RL approaches and to reduce the number of interactions. Sequence2Sequence (Seq2Seq) (Sutskever et al., 2014) approaches have also been adopted in user simulators to produce user acts (Gur et al., 2018). These approaches typically assume that the agent can only produce one act per turn through classification. Generating only one act per turn significantly limits what an agent can do in a turn and leads to lengthy dialogues, making tracking of state and context throughout the dialogue harder. An example in Table 1 shows how the agent can produce both an *inform* and a *multiple_choice* act, reducing the need for additional turns.

| | |
|---|---|
| user msg | Hi! I'm looking for good thriller. Are there any playing right now? |
| agent msg | Yes, there are! The Witch, The Other Side of the Door, and The Boy are all thrillers. Would you like to find tickets for a showing for any of them? |
| agent acts | inform(moviename=The Witch, The Other Side of the Door, The Boy; genre=thriller) multiple_choice(moviename) |

Table 1: Dialogue example.

The use of multiple actions has previously been used in interaction managers that keep track of the floor (who is speaking right now) (Raux and Eskénazi, 2007; Khouzaimi et al., 2015; Hastie et al., 2013, among others), but the option of generating multiple acts simultaneously at each turn for dialogue policy has been largely ignored, and only explored in simulated scenarios without real data (Chandramohan and Pietquin, 2010).

This task can be cast as a multi-label classification problem (if the sequential dependency among the acts is ignored) or as a sequence generation one as shown in Table 2.

In this paper, we introduce a novel policy model to output multiple actions per turn (called multi-act), generating a sequence of tuples and expanding agents' expressive power. Each tuple is defined as $(continue, act, slots)$, where *continue* indicates whether to continue or stop producing new acts, *act* is an *act type* (e.g., *inform* or *request*), and *slots* is a set of slots (names) associated with the current *act type*. Correspondingly, a novel decoder

[1]The code is available at https://leishu02.github.io/



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1304–1310,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

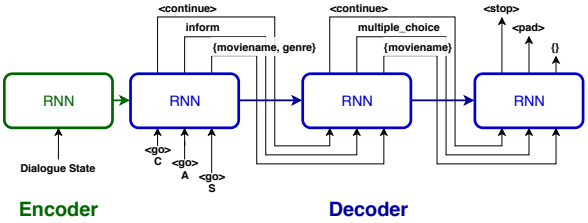


Figure 1: CAS decoder: at each step, a tuple of (continue, act, slots) is produced. The KB vector $k$ regarding the queried result from knowledge base is not shown for brevity.

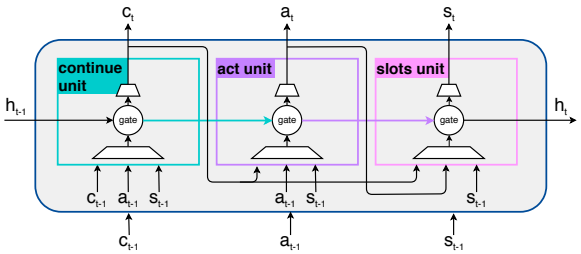


Figure 2: The gated CAS recurrent cell contains three units: continue unit, act unit and slots unit. The three units use a gating mechanism and are sequentially connected. The KB vector $k$ is not shown for brevity.

(Figure 1) is proposed to produce such sequences. Each tuple is generated by a cell called gated Continue Act Slots (gCAS, as in Figure 2), which is composed of three sequentially connected gated units handling the three components of the tuple. This decoder can generate multi-acts in a double recurrent manner (Tay et al., 2018). We compare this model with baseline classifiers and sequence generation models and show that it consistently outperforms them.

## 2 Methodology

The proposed policy network adopts an encoder-decoder architecture (Figure 1). The input to the encoder is the current-turn dialogue state, which follows Li et al. (2018)'s definition. It contains policy actions from the previous turn, user dialogue acts from the current turn, user requested slots, the user informed slots, the agent requested slots and agent proposed slots. We treat the dialogue state as a sequence and adopt a GRU (Cho et al., 2014) to encode it. The encoded dialogue state is a sequence of vectors $\mathbf{E} = (e_0, \ldots, e_l)$ and the last hidden state is $h^E$. The CAS decoder recurrently generates tuples at each step. It takes $h^E$ as initial hidden state $h_0$. At each decoding step, the input contains the previous (continue, act, slots) tuple $(c_{t-1}, a_{t-1}, s_{t-1})$. An additional vector $k$ containing the number of results from the knowledge base (KB) query and the current turn number is given as input. The output of the decoder at each step is a tuple $(c, a, s)$, where $c \in \{\langle\text{continue}\rangle, \langle\text{stop}\rangle, \langle\text{pad}\rangle\}$, $a \in A$ (one act from the act set), and $s \subset S$ (a subset from the slot set).

### 2.1 gCAS Cell

As shown in Figure 2, the gated CAS cell contains three sequentially connected units for outputting continue, act, and slots respectively.

The **Continue unit** maps the previous tuple $(c_{t-1}, a_{t-1}, s_{t-1})$ and the KB vector $k$ into $x_t^c$. The hidden state from the previous step $h_{t-1}$ and $x_t^c$ are inputs to a GRU$^c$ unit that produces output $g_t^c$ and hidden state $h_t^c$. Finally, $g_t^c$ is used to predict $c_t$ through a linear projection and a softmax.

$$\begin{aligned} x_t^c &= W_x^c[c_{t-1}, a_{t-1}, s_{t-1}, k] + b_x^c, \\ g_t^c, h_t^c &= \text{GRU}^c(x_t^c, h_{t-1}), \\ P(c_t) &= \text{softmax}(W_g^c g_t^c + b_g^c), \\ \mathcal{L}^c &= -\sum_t \log P(c_t). \end{aligned} \tag{1}$$

The **Act unit** maps the tuple $(c_t, a_{t-1}, s_{t-1})$ and the KB vector $k$ into $x_t^a$. The hidden state from the continue cell $h_t^c$ and $x_t^a$ are inputs to a GRU$^a$ unit that produces output $g_t^a$ and hidden state $h_t^a$. Finally, $g_t^a$ is used to predict $a_t$ through a linear projection and a softmax.

$$\begin{aligned} x_t^a &= W_x^a[c_t, a_{t-1}, s_{t-1}, k] + b_x^a, \\ g_t^a, h_t^a &= \text{GRU}^a(x_t^a, h_t^c), \\ P(a_t) &= \text{softmax}(W_g^a g_t^a + b_g^a), \\ \mathcal{L}^a &= -\sum_t \log P(a_t). \end{aligned} \tag{2}$$

The **Slots unit** maps the tuple $(c_t, a_t, s_{t-1})$ and the KB vector $k$ into $x_t^s$. The hidden state from the act cell $h_t^a$ and $x_t^s$ are inputs to a GRU$^s$ unit that produces output $g_t^s$ and hidden state $h_t^s$. Finally, $g_t^a$ is used to predict $s_t$ through a linear projection and a sigmoid. Let $z_t^i$ be the $i$-th slot's ground truth.

$$\begin{aligned} x_t^s &= W_x^s[c_t, a_t, s_{t-1}, k] + b_x^s, \\ g_t^s, h_t^s &= \text{GRU}^s(x_t^s, h_t^a), \\ s_t &= \text{sigmoid}(W_g^s g_t^s + b_g^s), \\ \mathcal{L}^s &= -\sum_t \sum_{i=0}^{|S|} z_t^i \log s_t^i + (1 - z_t^i) \log\left(1 - s_t^i\right). \end{aligned} \tag{3}$$

The overall loss is the sum of the losses of the three units: $\mathcal{L} = \mathcal{L}^c + \mathcal{L}^a + \mathcal{L}^s$

| | |
|---|---|
| annotation | inform(moviename=The Witch, The Other Side of the Door, The Boy; genre=thriller) multiple_choice(moviename) |
| classification | inform+moviename, inform+genre, multiple_choice+moviename |
| sequence | 'inform' '(' 'moviename' '=' ';' 'genre' '=' ')' 'multiple_choice' '(' 'moviename' ')' '⟨eos⟩' |
| cas sequence | (⟨continue⟩, inform, {moviename, genre}) (⟨continue⟩, multiple_choice, {moviename}) (⟨stop⟩, ⟨pad⟩, {}) |

Table 2: Multiple dialogue act format in different architectures.

| domain | total | train | valid | test | acts | slots | pairs |
|---|---|---|---|---|---|---|---|
| movie | 2888 | 1445 | 433 | 1010 | 11 | 29 | 90 |
| taxi | 3093 | 1548 | 463 | 1082 | 11 | 23 | 63 |
| restaurant | 4101 | 2051 | 615 | 1435 | 11 | 31 | 91 |

Table 3: Dataset: train, validation and test split, and the count of distinct acts, slots and act-slot pairs.

| domain & speaker | 1 act | 2 acts | 3 acts | 4 acts |
|---|---|---|---|---|
| movie user | 9130 | 1275 | 106 | 11 |
| movie agent | 5078 | 4982 | 427 | 33 |
| taxi user | 10544 | 762 | 50 | 8 |
| taxi agent | 7855 | 3301 | 200 | 8 |
| restaurant user | 12726 | 1672 | 100 | 3 |
| restaurant agent | 10333 | 3755 | 403 | 10 |

Table 4: Dialogue act counts by turn.

## 3 Experiments

The experiment dataset comes from Microsoft Research (MSR) [2]. It contains three domains: movie, taxi, and restaurant. The total count of dialogues per domain and train/valid/test split is reported in Table 3. At every turn both user and agent acts are annotated, we use only the agent side as targets in our experiment. The acts are ordered in the dataset (each output sentence aligns with one act). The size of the sets of acts, slots, and act-slot pairs are also listed in Table 3. Table 4 shows the count of turns with multiple act annotations, which amounts to 23% of the dataset. We use MSR's dialogue management code and knowledge base to obtain the state at each turn and use it as input to every model.

### 3.1 Evaluation Metrics

We evaluate the performance at the act, frame and task completion level. For a frame to be correct, both the act and all the slots should match the ground truth. We report precision, recall, $F_1$ score of turn-level acts and frames. For task completion evaluation, **Entity $F_1$** score and **Success $F_1$** score (Lei et al., 2018) are reported. The Entity $F_1$ score, differently from the entity match rate in state tracking, compares the slots requested by the agent with the slots the user informed about and that were used to perform the KB query. We use it to measure agent performance in requesting information. The Success $F_1$ score compares the slots provided by the agent with the slots requested by the user. We use it to measure the agent performance in providing information.

[2] https://github.com/xiul-msr/e2e_dialog_challenge

| | Entity $F_1$ | | | Success $F_1$ | | |
|---|---|---|---|---|---|---|
| | movie | taxi | restaurant | movie | taxi | restaurant |
| Classification | 34.02 | 49.71 | 28.23 | 70.41 | **84.45** | 39.97 |
| Seq2Seq | 39.95 | 63.12 | 60.21 | 77.82 | 75.09 | 55.70 |
| Copy Seq2Seq | 28.04 | 62.95 | 59.14 | 77.59 | 74.58 | 58.74 |
| CAS | 48.02 | 59.16 | 54.70 | 76.81 | 78.89 | 65.18 |
| gCAS | **50.86** | **64.00** | **60.35** | **77.95** | 81.17 | **71.52** |

Table 5: Entity $F_1$ and Success $F_1$ at dialogue level.

**Critical slots and Non-critical slots**: By 'non-critical', we mean slots that the user informs the system about by providing their values and thus it is not critical for the system to provide them in the output. Table 1 shows an example, with the genre slot provided by the user and the system repeating it in its answer. Critical slots refers to slots that the system must provide like moviename in the Table 1 example. Although non-critical slots do not impact task completion directly, they may influence the output quality by enriching the dialogue state and helping users understand the system's utterance correctly. Furthermore, given the same dialog state, utterances offering non-critical slots or not offering them can both be present in the dataset, as they are optional. This makes the prediction of those slots more challenging for the system. To provide a more detailed analysis, we report the precision, recall, $F_1$ score of turn-level for all slots, critical slots and non-critical slots of the *inform* act.

### 3.2 Baseline

We compare five methods on the multi-act task.

**Classification** replicates the MSR challenge (Li et al., 2018) policy network architecture: two fully connected layers. We replace the last activation from softmax to sigmoid in order to predict probabilities for each act-slot pair. It is equivalent to binary classification for each act-slot pair and the

| | Act | | | | | | | | | Frame | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | movie | | | taxi | | | restaurant | | | movie | | | taxi | | | restaurant | | |
| method | $\mathcal{P}$ | $\mathcal{R}$ | $\mathcal{F}_1$ | $\mathcal{P}$ | $\mathcal{R}$ | $\mathcal{F}_1$ | $\mathcal{P}$ | $\mathcal{R}$ | $\mathcal{F}_1$ | $\mathcal{P}$ | $\mathcal{R}$ | $\mathcal{F}_1$ | $\mathcal{P}$ | $\mathcal{R}$ | $\mathcal{F}_1$ | $\mathcal{P}$ | $\mathcal{R}$ | $\mathcal{F}_1$ |
| classification | 84.19 | 50.24 | 62.93 | 92.20 | 55.48 | 69.27 | 79.71 | 33.94 | 47.60 | 63.91 | 18.39 | 28.56 | 65.87 | 44.31 | 52.98 | 49.63 | 12.32 | 19.74 |
| Seq2Seq | 73.44 | 73.62 | 73.53 | 77.52 | 69.29 | 73.17 | 65.66 | 66.01 | 65.83 | 42.88 | 24.81 | 31.43 | 57.12 | 50.32 | 53.51 | 39.97 | 25.40 | 31.06 |
| Copy Seq2Seq | 67.56 | 73.61 | 70.46 | 73.99 | 69.21 | 71.52 | 64.93 | 65.69 | 65.31 | 41.90 | 23.12 | 29.80 | 51.66 | 50.23 | 50.93 | 36.96 | 27.22 | 31.35 |
| CAS | 70.46 | 76.08 | 73.16 | 79.85 | 72.54 | 76.02 | 65.40 | 72.43 | 68.73 | 43.12 | 31.60 | 36.47 | 51.66 | 54.29 | 52.94 | 33.72 | 25.45 | 29.01 |
| gCAS | 73.08 | 75.78 | **74.41** | 79.47 | 75.39 | **77.37** | 68.30 | 74.39 | **71.22** | 42.24 | 35.50 | **38.58** | 53.77 | 56.24 | **54.98** | 36.86 | 32.41 | **34.49** |

Table 6: Precision ($\mathcal{P}$), Recall ($\mathcal{R}$) and $F_1$score ($\mathcal{F}_1$) of turn-level acts and frames.

| | example 1 | example 2 |
|---|---|---|
| groundtruth | request(date; starttime) | inform(restaurantname=; starttime =) multiple_choice(restaurantname) |
| classification | request+date | [] |
| Seq2Seq | 'request' '(' 'date' ';' 'starttime' ')' | 'inform' '(' 'restaurantname' '=' ')' 'multiple_choice' '=' 'restaurantname' ')' |
| Copy Seq2Seq | 'request' '(' 'date' '=' ')' | 'inform' '(' 'restaurantname' '=' ';' ';', ';', '=', ';' 'starttime' '=' ')' |
| CAS | request {} | inform {restaurantname} |
| gCAS | request {date; starttime} | inform {restaurantname} multiple_choice{restaurantname} |

Table 7: Examples of predicted dialogue acts in the restaurant domain.

loss is the sum of the binary cross-entropy of all of them.

**Seq2Seq** (Sutskever et al., 2014) encodes the dialogue state as a sequence, and decodes agent acts as a sequence with attention (Bahdanau et al., 2015).

**Copy Seq2Seq** (Gu et al., 2016) adds a copy mechanism to Seq2Seq, which allows copying words from the encoder input.

**CAS** adopts a single GRU (Cho et al., 2014) for decoding and uses three different fully connected layers for mapping the output of the GRU to continue, act and slots. For each step in the sequence of CAS tuples, given the output of the GRU, continue, act and slot predictions are obtained by separate heads, each with one fully connected layer. The hidden state of the GRU and the predictions at the previous step are passed to the cell at the next step connecting them sequentially.

**gCAS** uses our proposed recurrent cell which contains separate continue, act and slots unit that are sequentially connected.

The classification architecture has two fully connected layers of size 128, and the remaining models have a hidden size of 64 and a teacher-forcing rate of 0.5. Seq2Seq and Copy Seq2Seq use a beam search with beam size 10 during inference. CAS and gCAS do not adopt a beam search since their inference steps are much less than Seq2Seq methods. All models use Adam optimizer (Kingma and Ba, 2015) with a learning rate of 0.001.

### 3.3 Result and Error Analysis

As shown in Table 5, gCAS outperforms all other methods on Entity $F_1$ in all three domains. Compared to Seq2Seq, the performance advantage of gCAS in the taxi and restaurant domains is small, while it is more evident in the movie domain. The reason is that in the movie domain the proportion of turns with multiple acts is higher (52%), while in the other two domains it is lower (30%). gCAS also outperforms all other models in terms of Success $F_1$ in the movie and restaurant domain but is outperformed by the classification model in the taxi domain. The reason is that in the taxi domain, the agent usually informs the user at the last turn, while in all previous turns the agent usually requests information from the user. It is easy for the classification model to overfit this pattern. The advantage of gCAS in the restaurant domain is much more evident: the agent's *inform* act usually has multiple slots (see example 2 in Table 7) and this makes classification and sequence generation harder, but gCAS multi-label slots decoder handles it easily.

Table 6 shows the turn-level acts and frame prediction performance. CAS and gCAS outperform all other models in acts prediction in terms of $F_1$ score. The main reason is that CAS and gCAS output a tuple at each recurrent step, which makes for shorter sequences that are easier to generate compared to the long sequences of Seq2Seq (example 2 in Table 7). The classification method has a good precision score, but a lower recall score, suggesting it has problems making granular decisions (example 2 in Table 7). At the frame level, gCAS still outperforms all other methods. The performance

| | All Slots | | | | | | | | | Non-critical slots | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | movie | | | taxi | | | restaurant | | | movie | | | taxi | | | restaurant | | |
| method | $\mathcal{P}$ | $\mathcal{R}$ | $\mathcal{F}_1$ | $\mathcal{P}$ | $\mathcal{R}$ | $\mathcal{F}_1$ | $\mathcal{P}$ | $\mathcal{R}$ | $\mathcal{F}_1$ | $\mathcal{P}$ | $\mathcal{R}$ | $\mathcal{F}_1$ | $\mathcal{P}$ | $\mathcal{R}$ | $\mathcal{F}_1$ | $\mathcal{P}$ | $\mathcal{R}$ | $\mathcal{F}_1$ |
| classification | 67.90 | 21.48 | 32.64 | 73.52 | 72.66 | 73.08 | 45.16 | 12.71 | 19.84 | 62.98 | 13.39 | 22.08 | 43.91 | 60.03 | 50.72 | 33.61 | 11.15 | 16.75 |
| Seq2Seq | 53.25 | 29.54 | 38.00 | 64.09 | 74.32 | 68.83 | 42.36 | 17.73 | 25.00 | 47.90 | 13.95 | 21.61 | 64.15 | 48.45 | 55.20 | 35.28 | 12.95 | 18.94 |
| Copy Seq2Seq | 52.78 | 28.43 | 36.95 | 56.92 | 74.06 | 64.37 | 38.15 | 22.38 | 28.21 | 40.45 | 12.48 | 19.07 | 45.95 | 55.46 | 50.26 | 34.90 | 19.11 | 24.70 |
| CAS | 63.61 | 33.16 | 43.59 | 61.90 | 80.39 | 69.94 | 51.12 | 22.57 | 31.31 | 56.21 | 26.96 | 36.44 | 43.03 | 68.03 | 52.72 | 37.31 | 15.87 | 22.27 |
| gCAS | 54.75 | 38.70 | 45.35 | 62.31 | 79.76 | 69.96 | 44.20 | 29.65 | 35.49 | 48.23 | 36.68 | 41.67 | 44.35 | 62.15 | 51.77 | 31.26 | 29.60 | 30.41 |

Table 8: $\mathcal{P}$, $\mathcal{R}$ and $\mathcal{F}_1$ of turn-level *inform* all slots and non-critical slots.

| | Critical Slots | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | movie | | | taxi | | | restaurant | | |
| method | $\mathcal{P}$ | $\mathcal{R}$ | $\mathcal{F}_1$ | $\mathcal{P}$ | $\mathcal{R}$ | $\mathcal{F}_1$ | $\mathcal{P}$ | $\mathcal{R}$ | $\mathcal{F}_1$ |
| Classification | 70.29 | 29.13 | 41.19 | 85.18 | 75.90 | 80.27 | 55.66 | 13.76 | 22.07 |
| Seq2Seq | 55.08 | 44.26 | 49.08 | 64.08 | 80.97 | 71.54 | 46.24 | 20.97 | 28.86 |
| Copy Seq2Seq | 57.54 | 43.49 | 49.54 | 59.49 | 78.83 | 67.81 | 40.11 | 24.59 | 30.49 |
| CAS | 69.59 | 39.02 | 50.00 | 68.15 | 83.57 | 75.08 | 59.93 | 27.10 | 37.32 |
| gCAS | 61.89 | 40.62 | 49.04 | 67.48 | 84.28 | 74.95 | 61.35 | 29.69 | 40.01 |

Table 9: $\mathcal{P}$, $\mathcal{R}$ and $\mathcal{F}_1$ of turn-level *inform* critical slots.

difference between CAS and gCAS on frames becomes much more evident, suggesting that gCAS is more capable of predicting slots that are consistent with the act. This finding is also consistent with their Entity $F_1$ and Success $F_1$ performance.

However, gCAS's act-slot pair performance is far from perfect. The most common failure case is on non-critical slots (like 'genre' in the example in Table 2): gCAS does not predict them, while it predicts the critical ones (like 'moviename' in the example in Table 2).

Table 7 shows predictions of all methods from two emblematic examples. Example 1 is a frequent single-act multi-slots agent act. Example 2 is a complex multi-act example. The baseline classification method can predict frequent pairs in the dataset, but cannot predict any act in the complex example. The generated sequences of Copy Seq2Seq and Seq2Seq show that both models struggle in following the syntax. CAS cannot predict slots correctly even if the act is common in the dataset. gCAS returns a correct prediction for Example 1, but for Example 2 gCAS cannot predict 'starttime', which is a non-critical slot.

Tables 8 and 9 show the results of all slots, critical slots and non-critical slots under the *inform* act. gCAS performs better than the other methods on all slots in the movie and restaurant domains. The reason why classification performs the best here in the taxi domain is the same as the Success $F_1$. In the taxi domain, the agent usually informs the user at the last turn. The non-critical slots are also repeated frequently in the taxi domain, which makes their prediction easier. gCAS's performance is close to other methods on critical-slots. The reason is that the *inform* act is mostly the first act in multi-act and critical slots are usually frequent in the data. All methods can predict them well.

In the movie and restaurant domains, the *inform* act usually appears during the dialogue and there are many optional non-critical slots that can appear (see Table 3, movie and restaurant domains have more slots and pairs than the taxi domain). gCAS can better predict the non-critical slots than other methods. However, the overall performance on non-critical slots is much worse than critical slots since their appearances are optional and inconsistent in the data.

## 4 Conclusion and Future Work

In this paper, we introduced a multi-act dialogue policy model motivated by the need for a richer interaction between users and conversation agents. We studied classification and sequence generation methods for this task, and proposed a novel recurrent cell, gated CAS, which allows the decoder to output a tuple at each step. Experimental results showed that gCAS is the best performing model for multi-act prediction. The CAS decoder and the gCAS cell can also be used in a user simulator and gCAS can be applied in the encoder. A few directions for improvement have also been identified: 1) improving the performance on non-critical slots, 2) tuning the decoder with RL, 3) text generation from gCAS. We leave them as future work.

## Acknowledgments

We would like to express our special thanks to Alexandros Papangelis and Gokhan Tur for their support and contribution. We also would like to thank Xiujun Li for his help on dataset preparation and Jane Hung for her valuable comments. Bing Liu is partially supported by the NSF grant IIS-1910424 and a research gift from Northrop Grumman.

## References


Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations, San Diego, California, USA*.

Senthilkumar Chandramohan and Olivier Pietquin. 2010. User and noise adaptive dialogue management using hybrid system actions. In *Spoken Dialogue Systems for Ambient Environments*, pages 13–24. Springer.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*, pages 1724–1734. ACL.

Milica Gasic and Steve J. Young. 2014. Gaussian processes for pomdp-based dialogue manager optimization. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22:28–40.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *ACL (1)*. The Association for Computer Linguistics.

Izzeddin Gur, Dilek Z. Hakkani-Tur, Gokhan Tur, and Pararth Shah. 2018. User modeling for task oriented dialogues. *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 900–906.

Helen F. Hastie, Marie-Aude Aufaure, Panos Alexopoulos, Heriberto Cuayáhuitl, Nina Dethlefs, Milica Gasic, James Henderson, Oliver Lemon, Xingkun Liu, Peter Mika, Nesrine Ben Mustapha, Verena Rieser, Blaise Thomson, Pirros Tsiakoulis, and Yves Vanrompay. 2013. Demonstration of the parlance system: a data-driven incremental, spoken dialogue system for interactive search. In *SIGDIAL Conference*.

James Henderson, Oliver Lemon, and Kallirroi Georgila. 2005. Hybrid reinforcement/supervised learning for dialogue policies from communicator data. In *IJCAI workshop on knowledge and reasoning in practical dialogue systems*, pages 68–75. Citeseer.

James Henderson, Oliver Lemon, and Kallirroi Georgila. 2008. Hybrid reinforcement/supervised learning of dialogue policies from fixed data sets. *Computational Linguistics*, 34(4):487–511.

Hatim Khouzaimi, Romain Laroche, and Fabrice Lefèvre. 2015. Turn-taking phenomena in incremental dialogue systems. In *EMNLP*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations, San Diego, California, USA*.

Wenqiang Lei, Xisen Jin, Min-Yen Kan, Zhaochun Ren, Xiangnan He, and Dawei Yin. 2018. Sequicity: Simplifying task-oriented dialogue systems with single sequence-to-sequence architectures. In *ACL*.

Xiujun Li, Sarah Panda, Jingjing Liu, and Jianfeng Gao. 2018. Microsoft dialogue challenge: Building end-to-end task-completion dialogue systems. volume abs/1807.11125.

Bing Liu and Ian Lane. 2018. End-to-end learning of task-oriented dialogs. In *Proceedings of the NAACL-HLT*.

Antoine Raux and Maxine Eskénazi. 2007. A multilayer architecture for semi-synchronous event-driven dialogue management. *2007 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 514–519.

Amanda J Stent. 2002. A conversation acts model for generating spoken dialogue contributions. *Computer Speech & Language*, 16(3-4):313–352.

Pei-Hao Su, Pawel Budzianowski, Stefan Ultes, Milica Gasic, and Steve J. Young. 2017. Sample-efficient actor-critic reinforcement learning with supervised data for dialogue management. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue, Saarbrücken, Germany, August 15-17, 2017*, pages 147–157. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112.

Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2018. Recurrently controlled recurrent networks. In *Advances in Neural Information Processing Systems*, pages 4736–4748.

Marilyn A. Walker. 2000. An application of reinforcement learning to dialogue strategy selection in a spoken dialogue system for email. *J. Artif. Intell. Res.*, 12:387–416.

Tsung-Hsien Wen, Yishu Miao, Phil Blunsom, and Steve J. Young. 2017. Latent intention dialogue models. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 3732–3741. PMLR.

Jason D. Williams, Kavosh Asadi, and Geoffrey Zweig. 2017a. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *ACL (1)*, pages 665–677. Association for Computational Linguistics.

Jason D. Williams, Kavosh Asadi, and Geoffrey Zweig. 2017b. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *ACL*.

Jason D Williams and Geoffrey Zweig. 2016. End-to-end lstm-based dialog control optimized with supervised and reinforcement learning. *arXiv preprint arXiv:1606.01269*.

Steve J. Young, Jost Schatzmann, Karl Weilhammer, and Hui Ye. 2007. The hidden information state approach to dialog management. *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*, 4:IV–149–IV–152.

# An Evaluation Dataset for Intent Classification and Out-of-Scope Prediction

**Stefan Larson** **Anish Mahendran** **Joseph J. Peper** **Christopher Clarke**
**Andrew Lee** **Parker Hill** **Jonathan K. Kummerfeld** **Kevin Leach**
**Michael A. Laurenzano** **Lingjia Tang** **Jason Mars**
Clinc, Inc.
Ann Arbor, MI, USA
stefan@clinc.com

## Abstract

Task-oriented dialog systems need to know when a query falls outside their range of supported intents, but current text classification corpora only define label sets that cover every example. We introduce a new dataset that includes queries that are out-of-scope—i.e., queries that do not fall into any of the system's supported intents. This poses a new challenge because models cannot assume that every query at inference time belongs to a system-supported intent class. Our dataset also covers 150 intent classes over 10 domains, capturing the breadth that a production task-oriented agent must handle. We evaluate a range of benchmark classifiers on our dataset along with several different out-of-scope identification schemes. We find that while the classifiers perform well on in-scope intent classification, they struggle to identify out-of-scope queries. Our dataset and evaluation fill an important gap in the field, offering a way of more rigorously and realistically benchmarking text classification in task-driven dialog systems.

## 1 Introduction

Task-oriented dialog systems have become ubiquitous, providing a means for billions of people to interact with computers using natural language. Moreover, the recent influx of platforms and tools such as Google's DialogFlow or Amazon's Lex for building and deploying such systems makes them even more accessible to various industries and demographics across the globe.

Tools for developing such systems start by guiding developers to collect training data for intent classification: the task of identifying which of a fixed set of actions the user wishes to take based on their query. Relatively few public datasets exist for evaluating performance on this task, and those that do exist typically cover only a very small number of intents (e.g. Coucke et al. (2018), which has 7 intents). Furthermore, such resources do not facilitate analysis of *out-of-scope* queries: queries that users may reasonably make, but fall outside of the scope of the system-supported intents.

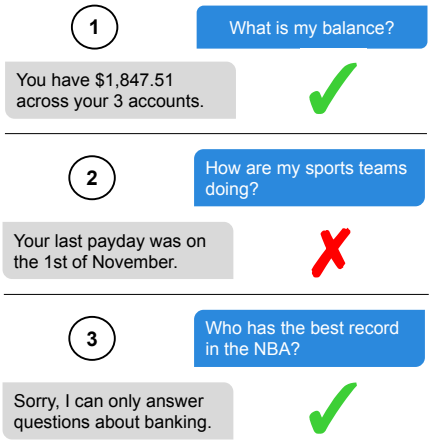


Figure 1: Example exchanges between a user (blue, right side) and a task-driven dialog system for personal finance (grey, left side). The system correctly identifies the user's query in (1), but in (2) the user's query is mis-identified as in-scope, and the system gives an unrelated response. In (3) the user's query is correctly identified as out-of-scope and the system gives a fallback response.

Figure 1 shows example query-response exchanges between a user and a task-driven dialog system for personal finance. In the first user-system exchange, the system correctly identifies the user's intent as an in-scope BALANCE query. In the second and third exchanges, the user queries with out-of-scope inputs. In the second exchange, the system incorrectly identifies the query as in-scope and yields an unrelated response. In the third exchange, the system correctly classifies the user's query as out-of-scope, and yields a fallback response.

Out-of-scope queries are inevitable for a task-oriented dialog system, as most users will not be fully cognizant of the system's capabilities, which are limited by the fixed number of intent classes.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1311–1316,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

Correctly identifying out-of-scope cases is thus crucial in deployed systems—both to avoid performing the wrong action and also to identify potential future directions for development. However, this problem has seen little attention in analyses and evaluations of intent classification systems.

This paper fills this gap by analyzing intent classification performance with a focus on out-of-scope handling. To do so, we constructed a new dataset with 23,700 queries that are short and unstructured, in the same style made by real users of task-oriented systems. The queries cover 150 intents, plus out-of-scope queries that do not fall within any of the 150 in-scope intents.

We evaluate a range of benchmark classifiers and out-of-scope handling methods on our dataset. BERT (Devlin et al., 2019) yields the best in-scope accuracy, scoring 96% or above even when we limit the training data or introduce class imbalance. However, all methods struggle with identifying out-of-scope queries. Even when a large number of out-of-scope examples are provided for training, there is a major performance gap, with the best system scoring 66% out-of-scope recall. Our results show that while current models work on known classes, they have difficulty on out-of-scope queries, particularly when data is not plentiful. This dataset will enable future work to address this key gap in the research and development of dialog systems. All data introduced in this paper can be found at `https://github.com/clinc/oos-eval`.

## 2 Dataset

We introduce a new crowdsourced dataset of 23,700 queries, including 22,500 in-scope queries covering 150 intents, which can be grouped into 10 general domains. The dataset also includes 1,200 out-of-scope queries. Table 1 shows examples of the data.[1]

### 2.1 In-Scope Data Collection

We defined the intents with guidance from queries collected using a *scoping* crowdsourcing task, which prompted crowd workers to provide questions and commands related to topic domains in the manner they would interact with an artificially intelligent assistant. We manually grouped data generated by *scoping* tasks into intents. To collect additional data for each intent, we used the *rephrase* and *scenario* crowdsourcing tasks proposed by Kang et al. (2018).[2] For each intent, there are 100 training queries, which is representative of what a team with a limited budget could gather while developing a task-driven dialog system. Along with the 100 training queries, there are 20 validation and 30 testing queries per intent.

### 2.2 Out-of-Scope Data Collection

Out-of-scope queries were collected in two ways. First, using worker mistakes: queries written for one of the 150 intents that did not actually match any of the intents. Second, using *scoping* and *scenario* tasks with prompts based on topic areas found on Quora, Wikipedia, and elsewhere. To help ensure the richness of this additional out-of-scope data, each of these task prompts contributed to at most four queries. Since we use the same crowdsourcing method for collecting out-of-scope data, these queries are similar in style to their in-scope counterparts.

The out-of-scope data is difficult to collect, requiring expert knowledge of the in-scope intents to painstakingly ensure that no out-of-scope query sample is mistakenly labeled as in-scope (and vice versa). Indeed, roughly only 69% of queries collected with prompts targeting out-of-scope yielded out-of-scope queries. Of the 1,200 out-of-scope queries collected, 100 are used for validation and 100 are used for training, leaving 1,000 for testing.

### 2.3 Data Preprocessing and Partitioning

For all queries collected, all tokens were downcased, and all end-of-sentence punctuation was removed. Additionally, all duplicate queries were removed and replaced.

In an effort to reduce bias in the in-scope data, we placed all queries from a given crowd worker in a single split (train, validation, or test). This avoids the potential issue of similar queries from a crowd worker ending up in both the train and test sets, for instance, which would make the train and test distributions unrealistically similar. We note that this is a recommendation from concurrent work by Geva et al. (2019). We also used this procedure for the out-of-scope set, except that we split the data into train/validation/test based on task prompt instead of worker.

[1] See the supplementary material for a full list of domains and intents, as well as additional examples.

[2] In all cases, crowd workers were paid $0.20 per task. See the supplementary material for examples of each task.

| Domain | Intent | Query |
|---|---|---|
| BANKING | TRANSFER | *move 100 dollars from my savings to my checking* |
| WORK | PTO REQUEST | *let me know how to make a vacation request* |
| META | CHANGE LANGUAGE | *switch the language setting over to german* |
| AUTO & COMMUTE | DISTANCE | *tell the miles it will take to get to las vegas from san diego* |
| TRAVEL | TRAVEL SUGGESTION | *what sites are there to see when in evans* |
| HOME | TODO LIST UPDATE | *nuke all items on my todo list* |
| UTILITY | TEXT | *send a text to mom saying i'm on my way* |
| KITCHEN & DINING | FOOD EXPIRATION | *is rice ok after 3 days in the refrigerator* |
| SMALL TALK | TELL JOKE | *can you tell me a joke about politicians* |
| CREDIT CARDS | REWARDS BALANCE | *how high are the rewards on my discover card* |
| OUT-OF-SCOPE | OUT-OF-SCOPE | *how are my sports teams doing* |
| OUT-OF-SCOPE | OUT-OF-SCOPE | *create a contact labeled mom* |
| OUT-OF-SCOPE | OUT-OF-SCOPE | *what's the extended zipcode for my address* |

Table 1: Sample queries from our dataset. The out-of-scope queries are similar in style to the in-scope queries.

### 2.4 Dataset Variants

In addition to the full dataset, we consider three variations. First, **Small**, in which there are only 50 training queries per each in-scope intent, rather than 100. Second, **Imbalanced**, in which intents have either 25, 50, 75, or 100 training queries. Third, **OOS+**, in which there are 250 out-of-scope training examples, rather than 100. These are intended to represent production scenarios where data may be in limited or uneven supply.

## 3 Benchmark Evaluation

To quantify the challenges that our new dataset presents, we evaluated the performance of a range of classifier models[3] and out-of-scope prediction schemes.

### 3.1 Classifier Models

**SVM:** A linear support vector machine with bag-of-words sentence representations.
**MLP:** A multi-layer perceptron with USE embeddings (Cer et al., 2018) as input.
**FastText:** A shallow neural network that averages embeddings of *n*-grams (Joulin et al., 2017).
**CNN:** A convolutional neural network with non-static word embeddings initialized with GloVe (Pennington et al., 2014).
**BERT:** A neural network that is trained to predict elided words in text and then fine-tuned on our data (Devlin et al., 2019).
**Platforms:** Several platforms exist for the development of task-oriented agents. We consider Google's DialogFlow[4] and Rasa NLU[5] with spacy-sklearn.

[3]See the supplementary material for model details.
[4]https://dialogflow.com
[5]https://github.com/RasaHQ/rasa

### 3.2 Out-of-Scope Prediction

We use three baseline approaches for the task of predicting whether a query is out-of-scope: (1) ***oos-train***, where we train an additional (i.e. $151^{st}$) intent on out-of-scope training data; (2) ***oos-threshold***, where we use a threshold on the classifier's probability estimate; and (3) ***oos-binary***, a two-stage process where we first classify a query as in- or out-of-scope, then classify it into one of the 150 intents if classified as in-scope.

To reduce the severity of the class imbalance between in-scope versus out-of-scope query samples (i.e., 15,000 versus 250 queries for OOS+), we investigate two strategies when using *oos-binary*: one where we undersample the in-scope data and train using 1,000 in-scope queries sampled evenly across all intents (versus 250 out-of-scope), and another where we augment the 250 OOS+ out-of-scope training queries with 14,750 sentences sampled from Wikipedia.

From a development point of view, the *oos-train* and *oos-binary* methods both require careful curation of an out-of-scope training set, and this set can be tailored to individual systems. The *oos-threshold* method is a more general decision rule that can be applied to any model that produces a probability. In our evaluation, the out-of-scope threshold was chosen to be the value which yielded the highest validation score across all intents, treating out-of-scope as its own intent.

### 3.3 Metrics

We consider two performance metrics for all scenarios: (1) accuracy over the 150 intents, and (2) recall on out-of-scope queries. We use recall to evaluate out-of-scope since we are more interested in cases where such queries are predicted as in-

| | Classifier | In-Scope Accuracy Full | Small | Imbal | OOS+ | Out-Of-Scope Recall Full | Small | Imbal | OOS+ |
|---|---|---|---|---|---|---|---|---|---|
| *oos-train* | FastText | 89.0 | 84.5 | 87.2 | 89.2 | 9.7 | 23.2 | 12.2 | 32.2 |
| | SVM | 91.0 | 89.6 | 89.9 | 90.1 | 14.5 | 18.6 | 16.0 | 29.8 |
| | CNN | 91.2 | 88.9 | 89.1 | 91.0 | 18.9 | 22.2 | 19.0 | 34.2 |
| | DialogFlow | 91.7 | 89.4 | 90.7 | 91.7 | 14.0 | 14.1 | 15.3 | 28.5 |
| | Rasa | 91.5 | 88.9 | 89.2 | 90.9 | 45.3 | **55.0** | **49.6** | **66.0** |
| | MLP | 93.5 | 91.5 | 92.5 | 94.1 | **47.4** | 52.2 | 35.6 | 53.9 |
| | BERT | **96.9** | **96.4** | **96.3** | **96.7** | 40.3 | 40.9 | 43.8 | 59.2 |
| *oos-threshold* | SVM | 88.2 | 85.6 | 86.0 | — | 18.0 | 13.0 | 0.0 | — |
| | FastText | 88.6 | 84.8 | 86.6 | — | 28.3 | 6.0 | 33.2 | — |
| | DialogFlow | 90.8 | 89.2 | 89.2 | — | 26.7 | 20.5 | 38.1 | — |
| | Rasa | 90.9 | 89.6 | 89.4 | — | 31.2 | 1.0 | 0.0 | — |
| | CNN | 90.9 | 88.9 | 90.0 | — | 30.9 | 25.5 | 26.9 | — |
| | MLP | 93.4 | 91.3 | 92.5 | — | 49.1 | 32.4 | 13.3 | — |
| | BERT | **96.2** | **96.2** | **95.9** | — | **52.3** | **58.9** | **52.8** | — |

Table 2: Benchmark classifier results under each data condition using the *oos-train* (top half) and *oos-threshold* (bottom half) prediction methods.

scope, as this would mean a system gives the user a response that is completely wrong. Precision errors are less problematic as the fallback response will prompt the user to try again, or inform the user of the system's scope of supported domains.

## 4 Results

### 4.1 Results with *oos-train*

Table 2 presents results for all models across the four variations of the dataset. First, BERT is consistently the best approach for in-scope, followed by MLP. Second, out-of-scope query performance is much lower than in-scope across all methods. Training on less data (Small and Imbalanced) yields models that perform slightly worse on in-scope queries. The trend is mostly the opposite when evaluating out-of-scope, where recall increases under the Small and Imbalanced training conditions. Under these two conditions, the size of the in-scope training set was decreased, while the number of out-of-scope training queries remained constant. This indicates that out-of-scope performance can be increased by increasing the relative number of out-of-scope training queries. We do just that in the OOS+ setting—where the models were trained on the full training set as well as 150 additional out-of-scope queries—and see that performance on out-of-scope increases substantially, yet still remains low relative to in-scope accuracy.

### 4.2 Results with *oos-threshold*

In-scope accuracy using the *oos-threshold* approach is largely comparable to *oos-train*. Out-of-scope recall tends to be much higher on Full, but several models suffer greatly on the limited datasets. BERT and MLP are the top *oos-threshold* performers, and for several models the threshold approach provided erratic results, particularly FastText and Rasa.

| Classifier | In-Scope Accuracy under | wiki aug | Out-of-Scope Recall under | wiki aug |
|---|---|---|---|---|
| DialogFlow | 84.7 | — | 37.3 | — |
| Rasa | 87.5 | — | 37.7 | — |
| FastText | 88.1 | 87.0 | 22.7 | 31.4 |
| SVM | 88.4 | 89.3 | 32.2 | 37.7 |
| CNN | 89.8 | 90.1 | 25.6 | 39.7 |
| MLP | 90.1 | 92.9 | **52.8** | 32.4 |
| BERT | **94.4** | **96.0** | 46.5 | **40.4** |

Table 3: Results of *oos-binary* experiments on OOS+, where we compare performance of undersampling (under) and augmentation using sentences from Wikipedia (wiki aug). The wiki aug approach was too large for the DialogFlow and Rasa classifiers.

### 4.3 Results with *oos-binary*

Table 3 compares classifier performance using the *oos-binary* scheme. In-scope accuracy suffers for all models using the undersampling scheme when compared to training on the full dataset using the *oos-train* and *oos-threshold* approaches shown in Table 2. However, out-of-scope recall improves compared to *oos-train* on Full but not OOS+. Augmenting the out-of-scope training set appears to help improve both in-scope and out-of-scope performance compared to undersampling, but out-of-scope performance remains weak.

| Dataset | Num. Intents | Num. Utterances | Chatbot Style | Many Intents | Constrained Training Data | Out-of-Scope Utterances |
|---|---|---|---|---|---|---|
| Our Dataset (This Work) | 150 | 23,700 | ✓ | ✓ | ✓ | ✓ |
| TREC-6, (Li and Roth, 2002) | 6 | 5,952 | ✗ | ✗ | ✗ | ✗ |
| TREC-50, (Li and Roth, 2002) | 50 | 5,952 | ✗ | ✓ | ✓ | ✗ |
| Web Apps, (Braun et al., 2017) | 8 | 89 | ✗ | ✗ | ✓ | ✗ |
| Ask Ubuntu, (Braun et al., 2017) | 5 | 162 | ✗ | ✗ | ✓ | ✗ |
| Chatbot Corpus, (Braun et al., 2017) | 2 | 206 | ✓ | ✗ | ✓ | ✗ |
| Snips, (Coucke et al., 2018) | 7 | 14,484 | ✓ | ✗ | ✗ | ✗ |
| Liu et al. (2019) | 54 | 25,716 | ✓ | ✓ | ✗ | ✗ |

Table 4: Classification dataset properties. Ours has the broadest range of intents and specially collected out-of-scope queries. We consider "chatbot style" queries to be short, possibly unstructured questions and commands.

## 5 Prior Work

In most other analyses and datasets, the idea of out-of-scope data is not considered, and instead the output classes are intended to cover all possible queries (e.g., TREC (Li and Roth, 2002)). Recent work by Hendrycks and Gimpel (2017) considers a similar problem they call out-of-distribution detection. They use other datasets or classes excluded during training to form the out-of-distribution samples. This means that the out-of-scope samples are from a small set of coherent classes that differ substantially from the in-distribution samples. Similar experiments were conducted for evaluating unknown intent discovery models in Lin and Xu (2019). In contrast, our out-of-scope queries cover a broad range of phenomena and are similar in style and often similar in topic to in-scope queries, representing things a user might say given partial knowledge of the capabilities of a system.

Table 4 compares our dataset with other short-query intent classification datasets. The Snips (Coucke et al., 2018) dataset and the dataset presented in Liu et al. (2019) are the most similar to the in-scope part of our work, with the same type of conversational agent requests. Like our work, both of these datasets were bootstrapped using crowdsourcing. However, the Snips dataset has only a small number of intents and an enormous number of examples of each. Snips does present a low-data variation, with 70 training queries per intent, in which performance drops slightly. The dataset presented in Liu et al. (2019) has a large number of intent classes, yet also contains a wide range of samples per intent class (ranging from 24 to 5,981 queries per intent, and so is not constrained in all cases).

Braun et al. (2017) created datasets with constrained training data, but with very few intents, presenting a very different type of challenge. We also include the TREC query classification datasets (Li and Roth, 2002), which have a large set of labels, but they describe the desired response type (e.g., distance, city, abbreviation) rather than the action intents we consider. Moreover, TREC contains only questions and no commands. Crucially, none of the other datasets summarized in Table 4 offer a feasible way to evaluate out-of-scope performance.

The Dialog State Tracking Challenge (DSTC) datasets are another related resource. Specifically, DSTC 1 (Williams et al., 2013), DSTC 2 (Henderson et al., 2014a), and DSTC 3 (Henderson et al., 2014b) contain "chatbot style" queries, but the datasets are focused on state tracking. Moreover, most if not all queries in these datasets are in-scope. In contrast, the focus of our analysis is on both in- and out-of-scope queries that challenge a virtual assistant to determine whether it can provide an acceptable response.

## 6 Conclusion

This paper analyzed intent classification and out-of-scope prediction methods with a new dataset consisting of carefully collected out-of-scope data. Our findings indicate that certain models like BERT perform better on in-scope classification, but all methods investigated struggle with identifying out-of-scope queries. Models that incorporate more out-of-scope training data tend to improve on out-of-scope performance, yet such data is expensive and difficult to generate. We believe our analysis and dataset will lead to developing better, more robust dialog systems.

All datasets introduced in this paper can be found at https://github.com/clinc/oos-eval.

## References


Daniel Braun, Adrian Hernandez-Mendez, Florian Matthes, and Manfred Langen. 2017. Evaluating natural language understanding services for conversational question answering systems. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue (SIGDIAL)*.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations (EMNLP)*.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, Maël Primet, and Joseph Dureau. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *CoRR*, abs/1805.10190.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*.

Mor Geva, Yoav Goldberg, and Jonathan Berant. 2019. Are We Modeling the Task or the Annotator? An Investigation of Annotator Bias in Natural Language Understanding Datasets. *arXiv e-prints*, page arXiv:1908.07898.

Matthew Henderson, Blaise Thomson, and Jason D. Williams. 2014a. The second dialog state tracking challenge. In *Proceedings of the 15th Annual Meeting of the Special Interest Gorup on Discourse and Dialogue (SIGDIAL)*.

Matthew Henderson, Blaise Thomson, and Jason D. Williams. 2014b. The third dialog state tracking challenge. In *Spoken Language Technology Workshop (SLT)*.

Dan Hendrycks and Kevin Gimpel. 2017. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Yiping Kang, Yunqi Zhang, Jonathan K. Kummerfeld, Parker Hill, Johann Hauswald, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. 2018. Data collection for dialogue system: A startup perspective. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING)*.

Ting-En Lin and Hua Xu. 2019. Deep unknown intent detection with margin loss. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Xingkun Liu, Arash Eshghi, Pawel Swietojanski, and Verena Rieser. 2019. Benchmarking natural language understanding services for building conversational agents. In *Proceedings of the Tenth International Workshop on Spoken Dialogue Systems Technology (IWSDS)*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. 2013. The dialog state tracking challenge. In *Proceedings of the 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*.

# Automatically Learning Data Augmentation Policies for Dialogue Tasks

**Tong Niu** **Mohit Bansal**
UNC Chapel Hill
{tongn, mbansal}@cs.unc.edu

## Abstract

Automatic data augmentation (AutoAugment) (Cubuk et al., 2019) searches for optimal perturbation policies via a controller trained using performance rewards of a sampled policy on the target task, hence reducing data-level model bias. While being a powerful algorithm, their work has focused on computer vision tasks, where it is comparatively easy to apply imperceptible perturbations without changing an image's semantic meaning. In our work, we adapt AutoAugment to automatically discover effective perturbation policies for natural language processing (NLP) tasks such as dialogue generation. We start with a pool of atomic operations that apply subtle semantic-preserving perturbations to the source inputs of a dialogue task (e.g., different POS-tag types of stopword dropout, grammatical errors, and paraphrasing). Next, we allow the controller to learn more complex augmentation policies by searching over the space of the various combinations of these atomic operations. Moreover, we also explore conditioning the controller on the source inputs of the target task, since certain strategies may not apply to inputs that do not contain that strategy's required linguistic features. Empirically, we demonstrate that both our input-agnostic and input-aware controllers discover useful data augmentation policies, and achieve significant improvements over the previous state-of-the-art, including trained on manually-designed policies.

## 1 Introduction

Data augmentation aims at teaching invariances to a model so that it generalizes better outside the training set distribution. Recently, there has been substantial progress in Automatic Data Augmentation (*AutoAugment*) for computer vision (Cubuk et al., 2019). This algorithm searches for optimal perturbation policies via a controller trained with reinforcement learning, where its reward comes from training the target model with data perturbed by the sampled augmentation policy. Each policy consists of 5 sub-policies sampled randomly during training, and each sub-policy consists of 2 operations applied in sequence. These operations are semantic-preserving image processing functions such as *translation* and *rotation*.

We adapt AutoAugment to NLP tasks, where the operations are subtle, semantic-preserving text perturbations. To collect a pool of such operations, the first challenge we face is that the discrete nature of text makes it less straightforward to come up with semantic-preserving perturbations. We thus employ as a starting point Should-Not-Change strategies (equivalent to *operations* in this paper) proposed by Niu and Bansal (2018) which are shown to improve their dialogue task performance when trained on data perturbed by these strategies. Importantly, we next divide their operations into several smaller ones (such as Grammar Errors divided into Singular/Plural Errors and Verb Inflection Errors) and also add a new operation Stammer (word repetition). This modification provides a much larger space of operation combinations for the model to explore from, so that it could potentially learn more complex and nuanced augmentation policies. Figure 1 shows a sub-policy containing two operations. It first paraphrases 2 tokens with probability 0.7 and then introduces 1 grammar error with probability 0.4.[1]

We choose the dialogue generation task based on the Ubuntu Dialogue Corpus (Lowe et al., 2015) because as opposed to Natural Language Inference and Question Answering tasks, real-world dialogue datasets more naturally afford such perturbation-style human errors (i.e., contain more noise), and thus are inherently compatible with

[1]Our code and sampled augmented data is publicly available at: https://github.com/WolfNiu/AutoAugDialogue. The learned policies are presented in Table 4.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1317–1323,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

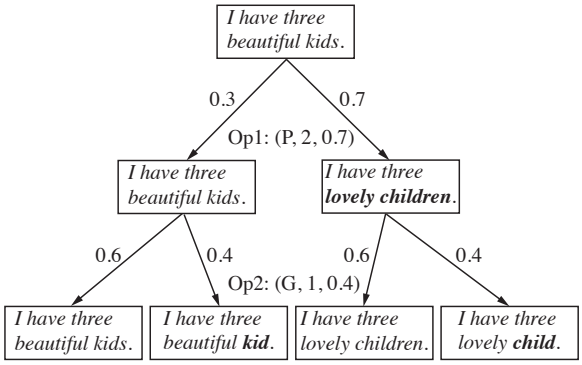


Figure 1: Example of a sub-policy applied to a source input. The first operation (Paraphrase, 2, 0.7) paraphrases the input twice with probability 0.7; the second operation (Grammar Errors, 1, 0.4) inserts 1 grammar error with probability 0.4. Thus there are at most 4 possible outcomes from each sub-policy.

a variety of artificial perturbations. Empirically, we show that our controller can self-learn policies that achieve state-of-the-art performance on Ubuntu, even compared with very strong baselines such as the best manually-designed policy in Niu and Bansal (2018). We also verify this result through human evaluation to show that our model indeed learns to generate higher-quality responses. We next analyze the best-learned policies and observe that the controller prefers to sample operations which work well on their own as augmentation policies, and then combines them into stronger policy sequences. Lastly, observing that some operations require the source inputs to have certain linguistic features (e.g., we cannot apply Stopword Dropout to inputs that do not contain stopwords), we also explore a controller that conditions on the source inputs of the target dataset, via a sequence-to-sequence (seq2seq) controller. We show that this input-aware model performs on par with the input-agnostic one (where the controller outputs do not depend on the source inputs), and may need more epochs to expose the model to the many diverse policies it generates. We also present selected best policies to demonstrate that the seq2seq controller can sometimes successfully attend to the source inputs.

## 2 Related Works

There has been extensive work that employs data augmentation in both computer vision (Simard et al., 2003; Krizhevsky et al., 2012; Cireşan et al., 2012; Wan et al., 2013; Sato et al., 2015; DeVries and Taylor, 2017; Tran et al., 2017; Lemley et al., 2017) and NLP (Fürstenau and Lapata, 2009; Sennrich et al., 2016; Wang and Yang, 2015; Zhang et al., 2015; Jia and Liang, 2016; Kim and Rush, 2016; Hu et al., 2017; Xu et al., 2017; Xia et al., 2017; Silfverberg et al., 2017; Kafle et al., 2017; Hou et al., 2018; Wang et al., 2018).

Automatic data augmentation is addressed via the AutoAugment algorithm proposed by Cubuk et al. (2019), which uses a hypernetwork (in our case, a controller) to train the target model, an approach inspired by neural architecture search (Zoph et al., 2017). Previous works have also adopted Generative Adversarial Networks (Goodfellow et al., 2014) to either directly generate augmented data (Tran et al., 2017; Sixt et al., 2018; Antoniou et al., 2017; Zhu et al., 2017; Mun et al., 2017; Wang and Perez, 2017), or generate augmentation strategies (Ratner et al., 2017). These approaches produce perturbations through continuous hidden representations. Motivated by the fact that our pool of candidate perturbations are discrete in nature, we identify AutoAugment as a more proper base model to use, and adapt it linguistically to the challenging task of generative dialogue tasks. Our work closely follows Niu and Bansal (2018) to obtain a pool of candidate operations. Although their work also used combinations of operations for data augmentation, their best model is manually designed, training on each operation in sequence, while our model automatically discovers more nuanced and detailed policies that have not only the operation types but also the intensity (the number of changes) and probability.

## 3 Model

**AutoAugment Architecture**: As shown in Figure 2, our AutoAugment model consists of a controller and a target model (Cubuk et al., 2019). The controller first samples a policy that transforms the original data to augmented data, on which the target model trains. After training, the target model is evaluated to obtain the performance on the validation set. This performance is then fed back to the controller as the reward signal. Figure 3 illustrates the details of input-agnostic and input-aware controllers. The former corresponds to the lower half of the figure. It consists of a single decoder that samples each operation in sequence. An operation is defined by 3 parameters: *Operation Type*, *Number of Changes* (the maximum number of times allowed to perform the operation, which

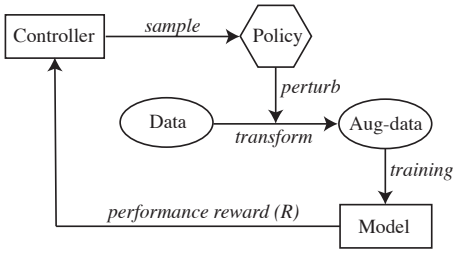


Figure 2: The controller samples a policy which is used to perturb the training data. After training on the augmented inputs, the model feeds the performance as reward back to the controller.

is a discrete equivalence of the continuous *Magnitude* in (Cubuk et al., 2019)), and the *Probability* of applying that operation. The input-aware controller corresponds to the whole figure, i.e., we novelly add in an encoder that takes as inputs the source of the training data, making it a seq2seq model. Since for each source input, there may be a different set of perturbations that are most suitable to it, our input-aware controller aims at providing customized operations for each training example.

**Search Space**: Following Niu and Bansal (2018), our pool of operations contains Random Swap, Stopword Dropout, Paraphrase, Grammar Errors, and Stammer, which cover a substantial enough search space of real-world noise in text.[2] To allow the controller to learn more nuanced combinations of operations, we further divide Stopword Dropout into 7 categories: Noun, Adposition, Pronoun, Adverb, Verb, Determiner, and Other, and divide Grammar Errors into Noun (plural/singular confusion) and Verb (verb inflected/base form confusion). For Stopword Dropout, we chose the first six categories because they are the major universal POS tags in the set closed-class words. We perform this subdivision also because different categories of an operation can influence the model to different extents. Suppose the original utterance is "*What is the offer?*", dropping the interrogative pronoun "*what*" is more semantic-changing than dropping the determiner "*the*." Our pool thus consists of 12 operations. For generality, we do not distinguish in advance which operation alone is effective as an augmentation policy on the target dataset, but rather let the controller figure it out. Moreover, it is possible that an operation alone is not effective, but when applied in sequence with another operation, they collectively teach the model a new pattern of invariance in the data.

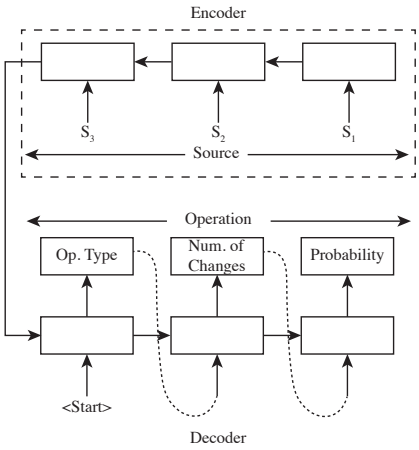


Figure 3: AutoAugment controller. An input-agnostic controller corresponds to the lower part of the figure. It samples a list of operations in sequence. An input-aware controller additionally has an encoder (upper part) that takes in the source inputs of the data.

Each policy consists of 4 sub-policies chosen randomly during training, each sub-policy consists of 2 operations,[3] and each operation has 3 hyperparameters. We discretize the Probability of applying an operation into 10 uniformly-spaced values starting from 0.1 and let the number of replaces range from 1 to 4. Thus, the search space size is $(12 \times 4 \times 10)^{(2\times4)} = 2.82 \times 10^{21}$. 2 operations allows more perturbations than exerted by a single strategy: in reality a sentence may contain several types of noise.

**Search Algorithm**: We use REINFORCE, a policy gradient method (Williams, 1992; Sutton et al., 2000) to train the controller. At each step, the decoder samples a token and feeds it into the next step. Since each policy consists of 4 sub-policies, each sub-policy contains 2 operations, and each operation is defined by 3 tokens (Operation Type, Number of Changes, Probability), the total number of steps for each policy is $4 \times 2 \times 3 = 24$. Sampling multiple sub-policies to form one policy provides us with a less biased estimation of the controller performance. We sample these 4 sub-policies at once rather than sample the controller 4 times to reduce repetition – the controller needs to keep track of what policies it has already sampled. To obtain the reward for the controller, we train the target model with the augmented data and obtain its validation set performance. We calculate a weighted average of two F1s (Activity and Entity)

[2]We did not use their Generative Paraphrasing where *number of changes* (which signifies the intensity of the policy) does not apply to this operation, because this operation can generate only one possible paraphrase for each input.

[3]Having 2 operations allows more perturbations than exerted by a single strategy: in reality a sentence may contain several types of noise.

as the reward $R$ since both are important complementary measures of an informative response, as discussed in Serban et al. (2017a).[4] We use the reinforcement loss following Ranzato et al. (2016) and an exponential moving average baseline to reduce training variance. At the end of the search, we use the best policy to train the target model from scratch and evaluate on the test set.

## 4 Experimental Setup

**Dataset and Model**: We investigate Variational Hierarchical Encoder-Decoder (VHRED) (Serban et al., 2017b) on the troubleshooting Ubuntu Dialogue task (Lowe et al., 2015).[5] For this short paper, we focus on this dataset because it has a well-established F1 metric among dialogue tasks.[6] In future work, we plan to apply the idea to other datasets and NLP tasks since the proposed model is not specific to the Ubuntu dataset. We hypothesize that most web/online datasets that have regular human errors/noise (e.g. Twitter Dialogue Dataset (Serban et al., 2017b) and multiple movie corpora (Serban et al., 2016)) would be suitable for our framework.

**Training Details**: We employ code[7] from Niu and Bansal (2018) to reproduce the VHRED baseline results and follow methods described in Zoph et al. (2017) to train the controller. For details, please refer to their Appendix A.2 (Controller Architecture) and A.3 (Training of the Controller).

We adopt the following method to speed up controller training and hence facilitate scaling.[8] We let the target model resume from the converged baseline checkpoint and train on the perturbed data for 1 epoch. During testing, we use the policy that achieves the highest weighted average F1 score to train the final model for 1 epoch. For the All-operations model (which corresponds to the All-Should-Not-Change model in Niu and Bansal (2018)), we train on each operation (without subdivisions for Stopword Dropout and Grammar Errors) for 1 epoch in sequence.[9]

| | Activity F1 | Entity F1 |
|---|---|---|
| LSTM | 1.18 | 0.87 |
| HRED | 4.34 | 2.22 |
| VHRED | 4.63 | 2.53 |
| VHRED (w/ attn.) | 5.94 | 3.52 |
| All-operations | 6.53 | 3.79 |
| Input-aware | **7.04** | 3.90 |
| Input-agnostic | 7.02 | **4.00** |

Table 1: Activity, Entity F1 results reported by previous work (rows 1-4 from Serban et al. (2017a)), and the All-operations and AutoAugment models.

| | W | T | L | W - L |
|---|---|---|---|---|
| Input-agnostic vs. baseline | 48 | 23 | 29 | 19 |
| Input-aware vs. baseline | 45 | 27 | 28 | 17 |
| Input-agnostic vs. All-ops | 43 | 27 | 30 | 13 |
| Input-aware vs. All-ops | 50 | 13 | 37 | 13 |

Table 2: Human evaluation results on comparisons among the baseline, All-operations, and the two AutoAugment models. **W**: Win, **T**: Tie, **L**: Loss.

**Automatic Evaluation**: We follow Serban et al. (2017a) and evaluate on F1s for both activities (technical verbs) and entities (technical nouns).

**Human Evaluation**: We conducted human studies on MTurk. We compared each of the input-agnostic/aware models with the VHRED baseline and All-operations from Niu and Bansal (2018), where we followed the same setting. Each study contains 100 samples randomly chosen from the test set. The utterances were randomly shuffled and we only allowed US-located human evaluators with approval rate $> 98\%$, and at least $10,000$ approved HITs. More details are in the appendix.

## 5 Results and Analysis

**Automatic Results**: Table 1 shows that all data-augmentation approaches (last 3 rows) improve statistically significantly ($p < 0.01$) [10] over the strongest baseline VHRED (w/ attention). Moreover, our input-agnostic AutoAugment is statistically significantly ($p < 0.01$) better (on Activity and Entity F1) than the strong manual-policy All-operations model, while the input-aware model is stat. signif. ($p < 0.01$) better on Activity F1.[11]

**Human Evaluation**: In Table 2, both AutoAug-

[4] We use the weighted average F1 = Activity F1 + 5.94 / 3.52 $*$ Entity F1, where the weights are determined by the baseline results, for balance between the two F1s.

[5] http://www.iulianserban.com/Files/UbuntuDialogueCorpus.zip

[6] Serban et al. (2017a) found that F1 is particularly suited for the goal-oriented Ubuntu Dialogue Corpus based on manual inspection of the extracted activities and entities.

[7] https://github.com/WolfNiu/AdversarialDialogue

[8] It took us 3 days with 6 P100 GPUs to get our reported results for each of the input-aware and input-agnostic models. The VHRED model took 1.5 days to pre-train.

[9] We do not follow Cubuk et al. (2019) to train on a small portion of the data for many epochs because we empirically found that this approach decreases our task performance.

[10] Statistical significance is computed via the bootstrap test (Noreen, 1989) over 100K samples.

[11] Note that F1s are overall low (reported baselines in Table 1 are from Serban et al. (2017a)) because Ubuntu is semi-task-oriented; each context can have many valid responses containing different activities/entities from the ground-truth.

| Perturbation Method | Policy | Source Inputs |
|---|---|---|
| Original Context | - | fresh install of crack of the day : gdm login → " can't access ACPI bla bla bla " __eou__ you don't want to be me ... __eou__ ah , it happened to you too ? |
| All-operations | (R , 4, 1.0) | fresh install **crack of** of the day : gdm login → " can't **ACPI access** bla bla bla " __eou__ you don't want **be to** me ... __eou__ ah , it happened **you to** too ? |
| Input-agnostic | ($D_v$, 3, 0.2) | fresh install of crack of the day : **login gdm** → " can't access ACPI bla bla bla " |
| | (R , 1, 0.5) | __eou__ you **don~~'~~t** want to **~~be~~** me ... __eou__ ah , it happened to you too ? |
| Input-aware | (S , 1, 0.8) | fresh install of crack of the day : gdm login → " can't access ACPI bla **bla** bla " |
| | ($D_v$, 2, 0.5) | __eou__ you don't want to be me ... __eou__ ah , it happened to you too ? |

Table 3: Comparisons of perturbed outputs among the three Augmentation models. Note that only the sampled/applied sub-policy is shown for each model. The dashed line separates the original and augmented source inputs. Perturbed tokens are boldfaced.

| Sub-policy1 | Sub-policy2 | Sub-policy3 | Sub-policy4 |
|---|---|---|---|
| P, 1, 0.5 | $D_v$, 3, 0.2 | R, 3, 0.9 | $D_p$, 2, 0.3 |
| $D_{adv}$, 4, 0.4 | R, 1, 0.5 | $D_{adp}$, 1, 0.5 | $D_{adp}$, 2, 0.1 |
| $D_n$, 1, 0.8 | $D_o$, 3, 1.0 | P, 4, 0.4 | $G_n$, 3, 0.3 |
| $G_v$, 1, 0.9 | $D_o$, 3, 0.1 | S, 3, 0.4 | R, 1, 0.2 |
| $D_v$, 2, 0.5 | $D_v$, 2, 0.7 | S, 3, 0.5 | P, 1, 1.0 |
| R, 2, 0.2 | $G_v$, 1, 0.9 | $D_o$, 1, 0.5 | $G_n$, 2, 0.6 |

Table 4: Top 3 policies on the validation set and their test performances. **Operations**: R=Random Swap, D=Stopword Dropout, P=Paraphrase, G=Grammar Errors, S=Stammer. **Universal tags**: n=noun, v=verb, p=pronoun, adv=adverb, adp=adposition.

ment models obtained significantly more net wins (last column) than the VHRED-attn baseline. They both outperform even the strong manual-policy All-operations model.

**Policy Learned by the Input-Agnostic Controller**: We present 3 best learned policies from the Ubuntu val set (Table 4). Although there is a probability of $7/12 = 58.3\%$ to sample one of the Stopword Dropout operations from our pool, all 3 learned policies show much more diversity on the operations they choose. This is also the case for the other two hyperparameters: *Number of Changes* varies from 1 to 4, and *Probability* varies from 0.1 to 0.9, which basically extend their whole search range. Moreover, all best policies include Random Swap, which agrees with the results in Niu and Bansal (2018).

**Example Analysis of Perturbation Procedure in Generated Responses**: We also present a selected example of perturbed source inputs from the three Augmentation models with their respective best policies in Table 3. First of all, the All-operations model is forced to use an operation (in this case Random Swap) with a fixed number of changes and a probability of 1.0, leading to less variation in the source inputs. On the other hand, our Input-agnostic AutoAugment model samples 3 Verb Dropout's followed by Random Swap. Note that although the number of changes for the dropout is 3, there are only 2 verb stopwords in the utterance. Thus, it has to resort back to modifying only 2 tokens. The Input-aware model samples Stammer followed by 2 Verb Dropout's. Interestingly, it inserts an extra "*bla*" around other "*bla*'s" in the utterance. It also did not sample a policy that drops more than 2 verb stopwords (this operation is not applied due to its Probability parameter). These two observations indicate that the model can sometimes successfully attend to the source inputs to provide customized policies.

## 6 Conclusion and Future Work

We adapt AutoAugment to dialogue and extend its controller to a conditioned model. We show via automatic and human evaluations that our AutoAugment models learn useful augmentation policies which lead to state-of-the-art results on the Ubuntu task. Motivated by the promising success of our model in this short paper, we will apply it to other diverse NLP tasks in future work.

## Acknowledgments

We thank the reviewers for their helpful comments and discussions. This work was supported by DARPA #YFA17-D17AP00022, ONR Grant #N00014-18-1-2871, and awards from Google, Facebook, Salesforce, and Adobe (plus Amazon and Google GPU cloud credits). The views are those of the authors and not of the funding agency.

## References

Antreas Antoniou, Amos Storkey, and Harrison Edwards. 2017. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of*

*International Conference on Learning Representations*.

Dan Cireşan, Ueli Meier, and Jürgen Schmidhuber. 2012. Multi-column deep neural networks for image classification. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 3642–3649.

Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. 2019. AutoAugment: Learning augmentation policies from data. In *Proceedings of Conference on Computer Vision and Pattern Recognition*.

Terrance DeVries and Graham W Taylor. 2017. Dataset augmentation in feature space. *arXiv preprint arXiv:1702.05538*.

Hagen Fürstenau and Mirella Lapata. 2009. Semi-supervised semantic role labeling. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 220–228. Association for Computational Linguistics.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Yutai Hou, Yijia Liu, Wanxiang Che, and Ting Liu. 2018. Sequence-to-sequence data augmentation for dialogue language understanding. In *Proceedings of COLING*.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning, PMLR 70*, pages 1587–1596.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of Association for Computational Linguistics*.

Kushal Kafle, Mohammed Yousefhussien, and Christopher Kanan. 2017. Data augmentation for visual question answering. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 198–202.

Yoon Kim and Alexander M Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 1317–1327.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Proceedings of Advances in neural information processing systems*, pages 1097–1105.

Joseph Lemley, Shabab Bazrafkan, and Peter Corcoran. 2017. Smart augmentation learning an optimal data augmentation strategy. In *Proceedings of IEEE Access*, volume 5, pages 5858–5869.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. *arXiv preprint arXiv:1506.08909*.

Seongkyu Mun, Sangwook Park, David K Han, and Hanseok Ko. 2017. Generative adversarial network based acoustic scene training set augmentation and selection using svm hyper-plane. *Detection and Classification of Acoustic Scenes and Events Workshop*, pages 93–97.

Tong Niu and Mohit Bansal. 2018. Adversarial over-sensitivity and over-stability strategies for dialogue models. In *Proceedings of Conference on Natural Language Learning*.

Eric W. Noreen. 1989. *Computer-intensive methods for testing hypotheses*. Wiley New York.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *Proceedings of International Conference on Learning Representations*.

Alexander J Ratner, Henry Ehrenberg, Zeshan Hussain, Jared Dunnmon, and Christopher Ré. 2017. Learning to compose domain-specific transformations for data augmentation. In *Proceedings of Advances in neural information processing systems*, pages 3236–3246.

Ikuro Sato, Hiroki Nishimura, and Kensuke Yokoi. 2015. Apac: Augmented pattern classification with neural networks. *arXiv preprint arXiv:1505.03229*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 86–96.

Iulian Vlad Serban, Tim Klinger, Gerald Tesauro, Kartik Talamadupula, Bowen Zhou, Yoshua Bengio, and Aaron C. Courville. 2017a. Multiresolution recurrent neural networks: An application to dialogue response generation. In *AAAI*, pages 3288–3294.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *The Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*, pages 3776–3784.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2017b. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Proceedings of AAAI*, pages 3295–3301.

Miikka Silfverberg, Adam Wiemerslage, Ling Liu, and Lingshuang Jack Mao. 2017. Data augmentation for morphological reinflection. *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 90–99.

Patrice Y Simard, Dave Steinkraus, and John C Platt. 2003. Best practices for convolutional neural networks applied to visual document analysis. In *Proceedings of International Conference on Document Analysis and Recognition*.

Leon Sixt, Benjamin Wild, and Tim Landgraf. 2018. Rendergan: Generating realistic labeled data. *Frontiers in Robotics and AI*, 5:66.

Richard S. Sutton, David Mcallester, Satinder Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of Advances in Neural Information Processing Systems 12*, pages 1057–1063. MIT Press.

Toan Tran, Trung Pham, Gustavo Carneiro, Lyle Palmer, and Ian Reid. 2017. A bayesian data augmentation approach for learning deep models. In *Proceedings of Advances in Neural Information Processing Systems*, pages 2797–2806.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. 2013. Regularization of neural networks using dropconnect. In *Proceedings of International Conference on Machine Learning*, pages 1058–1066.

Jason Wang and Luis Perez. 2017. The effectiveness of data augmentation in image classification using deep learning. *Convolutional Neural Networks Visual Recognition*.

William Yang Wang and Diyi Yang. 2015. That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using# petpeeve tweets. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2557–2563.

Xinyi Wang, Hieu Pham, Zihang Dai, and Graham Neubig. 2018. Switchout: an efficient data augmentation algorithm for neural machine translation. In *Proceedings of Empirical Methods in Natural Language Processing*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Proceedings of Reinforcement Learning*, pages 5–32. Springer.

Yingce Xia, Tao Qin, Wei Chen, Jiang Bian, Nenghai Yu, and Tie-Yan Liu. 2017. Dual supervised learning. In *Proceedings of ICML*, pages 3789–3798.

Weidi Xu, Haoze Sun, Chao Deng, and Ying Tan. 2017. Variational autoencoder for semi-supervised text classification. In *Proceedings of Association for the Advancement of Artificial Intelligence*, pages 3358–3364.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of Advances in neural information processing systems*, pages 649–657.

Xinyue Zhu, Yifan Liu, Zengchang Qin, and Jiahong Li. 2017. Data augmentation in emotion classification using generative adversarial networks. *arXiv preprint arXiv:1711.00648*.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. 2017. Learning transferable architectures for scalable image recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*.

## A Experimental Setup

**Dataset and Model**: We investigate Variational Hierarchical Encoder-Decoder (VHRED) (Serban et al., 2017b), a state-of-the-art dialogue model on the Ubuntu Dialogue Corpus (Lowe et al., 2015). Ubuntu is a troubleshooting dialogue dataset containing 1 million 2-person, multi-turn dialogues extracted from Ubuntu chat. The chat channels are used by customers to provide and receive technical support. We focus on the task of generating fluent, relevant, and goal-oriented responses. Following Niu and Bansal (2018), we apply additive attention mechanism (Bahdanau et al., 2015) to the source sequence while keeping the remaining VHRED architecture unchanged.

**Automatic Evaluation**: We follow Serban et al. (2017a) and evaluate the model on F1's for both activities (technical verbs, e.g., "boot", "delete") and entities (technical nouns, e.g., "root", "disk") computed by mapping the ground-truth and model responses to their corresponding activity-entity representations.

**Human Evaluation:** We conducted human studies on MTurk to evaluate the dialogue quality of generated responses from the investigated models. We compared each of the input-agnostic/aware models with All-operations and the baseline (i.e., 4 experiments in total). Studies involving the baseline are for sanity checks.

# uniblock: Scoring and Filtering Corpus with Unicode Block Information

**Yingbo Gao** **Weiyue Wang** **Hermann Ney**
Human Language Technology and Pattern Recognition Group
Computer Science Department
RWTH Aachen University
D-52056 Aachen, Germany
<surname>@i6.informatik.rwth-aachen.de

## Abstract

The preprocessing pipelines in Natural Language Processing usually involve a step of removing sentences consisted of illegal characters. The definition of illegal characters and the specific removal strategy depend on the task, language, domain, etc, which often lead to tiresome and repetitive scripting of rules. In this paper, we introduce a simple statistical method, uniblock[1], to overcome this problem. For each sentence, uniblock generates a fixed-size feature vector using Unicode block information of the characters. A Gaussian mixture model is then estimated on some clean corpus using variational inference. The learned model can then be used to score sentences and filter corpus. We present experimental results on Sentiment Analysis, Language Modeling and Machine Translation, and show the simplicity and effectiveness of our method.

## 1 Introduction

Identification and removal of sentences with illegal characters is a common heuristic in the preprocessing pipelines in Natural Language Processing (NLP). While it has benefits of controlling the vocabulary size and dropping noisy data, it is often a tedious work to come up with appropriate rules and removal strategies when facing with different tasks, languages and domains. The lack of clear definition of what is illegal exacerbates the problem. For example, modern Chinese text may allow characters such as: traditional and simplified Chinese characters, special punctuation marks, full-width characters, emojis, mathematical symbols, Latin characters, currency symbols, scientific notations, etc. As a result, scripting robust rules often requires a considerable amount of time and effort.

In this paper, we introduce a simple statistical method, uniblock, to address the problem. The motivation of our approach is straightforward - good rules may vary greatly depending on the situation, therefore instead of designing rules by hand, we use "rules" defined by data. We assume that some clean corpus is available, in which all the characters are deemed legal and the character distributions are similar to the test cases. It is possible to learn a probabilistic model, which describes the clean corpus and assigns scores to sentences in another corpus. The scores can further be used for filtering. Since the legalness of characters is in question, Unicode block information is a natural choice for obtaining feature vectors. Note that, by designing alternative feature vectors, one can potentially adapt uniblock to implement other corpus filtering heuristics.

We develop uniblock mainly for Machine Translation (MT). It can also be easily applied to other NLP tasks. We present experimental results on Sentiment Analysis (SA), Language Modeling (LM) and MT, and show the simplicity and effectiveness of our method.

## 2 Related Work

Raw data in NLP is often noisy. Khayrallah and Koehn (2018) categorize five common noise sources in parallel corpora and count only about 23% of the sentences in the raw 2016 ParaCrawl corpus[2] to be "Okay". Although illegal characters is not listed as a separate noise source, misuse of characters and shifting of character distributions may result in a sentence being classified into one of the five noise sources. In previous work, a supervised model using bag-of-words translation features is developed to classify clean and noisy data (Xu and Koehn, 2017; Khayrallah et al.,

[1]The source code is available at https://github.com/ringoreality/uniblock

[2]https://paracrawl.eu/index.html



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1324–1329,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

2018). In contrast, our model, which is trained in an unsupervised manner, tackles the illegal character problem explicitly.

Koehn et al. (2018) describe a shared task on parallel corpus filtering. While participating systems focus on addressing both monolingual fluency and bilingual adequacy, character-level filtering is common to all submissions. Junczys-Dowmunt (2018) applies a language identification model to implicitly remove sentences with illegal characters. Rossenbach et al. (2018) keep sentences with more than three words, with each word having at least one character from the predefined alphabet of the language. Lu et al. (2018) remove characters outside of a predefined alphabet. Ash et al. (2018) count most frequent characters, set a cutoff around eighty for each language, and remove sentences with illegal characters. Erdmann and Gwinnup (2018) get rid of lines containing characters from the Unicode general category of "other". Papavassiliou et al. (2018) simply consider Latin Unicode characters to be legal.

Unicode is the de facto standard for encoding characters from various languages, domains and sources (The Unicode Consortium, 2019). It uses "blocks" to group characters with similar origins or functions. The current version 12.0 defines 300 blocks, including Basic Latin, Latin-1 Supplement, CJK (Chinese, Japanese and Korean) Symbols and Punctuation, etc. To identify the legalness of characters, the Unicode block information provides meaningful discriminative signals.

The Gaussian Mixture Model (GMM) is a classic algorithm that assumes data is generated from a mixture of finite number of Gaussian distributions, whose parameters are typcially estimated with the Expectation–Maximization (EM) algorithm. An extension to the EM algorithm is variational inference, which has the advantage of automatically choosing the number of components. Bishop (2006) gives a comprehensive introduction to the topic. We use the implementation of variational Bayesian estimation of Gaussian mixtures from `scikit-learn` (Pedregosa et al., 2011).

## 3 Methodology

### 3.1 Feature Vectors

In order to assign meaningful scores to sentences and eventually filter out those who contain illegal characters, we first need to design appropriate features. We believe, the Unicode block information has a good property that the blocks are grouped by origin and function. For instance, CJK Symbols and Punctuation has a dedicated Unicode block in range U+3000...U+303F. Specifically, we count the appearances of characters in each of the 300 blocks. This will result in feature vectors $c_1, c_2, ..., c_B$ with a fixed length of $B = 300$. If we further normalize them by the total number of characters in each sentence, probability distributions over the 300 blocks can be obtained. We observe that there are only few blocks whose counts are non-zero in natural language texts. This calls for dimensionality reduction methods. Empirically, we drop the zero-count dimensions directly during training and assign conceptually low scores[3] when a non-zero count is seen during scoring. That is, we use normalized feature vectors $e = e_1, e_2, ..., e_{B'}$, where $1, 2, ..., B'$ are dimensions in $B$ whose original counts are non-zero, for training.

### 3.2 Bayesian Gaussian Mixture Model

Although many corpora are noisy, it is not appropriate to deem all sentences in them "dirty". While generating synthetic noisy data is always an option, it is unrealistic to cover all types of noises. Compared to the difficulty to obtain negatively labelled data, the development set is often available and can be deemed "clean" with high confidence. Therefore, we take the development set as training data for our scoring system and treat the problem as a clustering task rather than a classification task.

We assume that the training feature vectors $e$ are generated by a mixture of Gaussian distributions.

$$
\begin{aligned}
p(e) &= \sum_{k=1}^{K} \pi_k \mathcal{N}(e|\mu_k, \Sigma_k) \\
\pi_k &\sim \mathcal{DP}(\alpha) \\
\mu_k &\sim \mathcal{N}(\mu_0, \sqrt{\frac{1}{\tau}}) \\
\Sigma_k &\sim \mathcal{W}_{B'}(V, n)
\end{aligned}
\tag{1}
$$

In the equation above, $k$ is a running index in the number of mixtures $K$, $\pi_k$ is the mixture weight for the $k$-th Gaussian, $\mu_k$ is a $B'$-dimensional vector parametrizing the $k$-th mean, $\Sigma_k$ is the $k$-th $B' \times B'$ covariance matrix. We further impose priors on the model parameters: $\pi_k$ follows a Dirichlet Process ($\mathcal{DP}$), which is parametrized by concetration prior $\alpha$; $\mu_k$ follows a Normal distribution

[3] zeros in our experiments

($\mathcal{N}$), which is parametrized by the $B'$-dimensional mean prior $\mu_0$ and the precision prior $\tau$; $\Sigma_k$ follows a Wishart distribution ($\mathcal{W}$) in $B'$, which is parametrized by the covariance prior $V$ and degree of freedom prior $n$. We estimate the model parameters using the EM algorithm. Note that operating in $B'$ dimensions leads to significantly better model convergence than training in $B$ dimensions.

### 3.3 Scoring and Filtering

Once the model is trained till convergence, it is possible to use it to assign scores to unseen feature vectors. We directly use the weighted log probabilities as scores. Compared to the sentences used during `uniblock` training, higher scored sentences have more similar Unicode block count distributions, or fewer illegal characters. Regarding how to remove noisy sentences, we implement two straightforward ideas: absolute thresholding and relative thresholding. The former removes sentences with scores lower than a given threshold while the later removes a given percentage of low-scored sentences. We also add support for filtering parallel data. A simple method is to combine scores across parallel sentences with a specified reduction method (minimum, maximum, average), and then apply thresholds. Alternatively, one can use the lowest score seen during training as an absolute threshold for each language and filter parallel data in a "one bad, remove all" manner.

## 4 Experiments

We conduct experiments on three NLP tasks: SA, LM and MT. As shown in related work, many different heuristics are used for the purpose of illegal character filtering, and there is no generally accepted "golden rule". Therefore, we focus on comparing "no filtering" versus "filtering", to examine how our method performs.

### 4.1 Sentiment Analysis

We conduct SA experiments on the Stanford Twitter Sentiment (STS) corpus (Go et al., 2009) using fastText (Joulin et al., 2017). Given 1.6M automatically annotated tweets, the task is to predict postive or negative sentiments on 498 test tweets. For the baseline, we use minimal preprocessing to handle casing, weblinks, usernames, hashtags and spacing in the raw training tweets, without explicit character-level filtering. For `uniblock`, we additionally train a GMM on the tweets in the test set[4] and use the minimum score as an absolute threshold to filter the training corpus. In total, about 0.9% training tweets are filtered out. We observe that only one Unicode block exists in the test set, which means all sentences with characters in other blocks are assigned conceptually low scores and removed. In this particular case, our general method reduces to a simple filtering rule similar to that of Papavassiliou et al. (2018). As shown in Table 1, our method improves the test accuracy over the baseline by 0.6%.

| | test accuracy [%] |
|---|---|
| baseline | 81.6 |
| `uniblock` | **82.2** |

Table 1: Test accuracies on STS.

### 4.2 Language Modeling

In MT pipelines, language models are often used for corpus filtering, fusion techniques and $N$-best reranking. Therefore, we perform LM experiments on MT corpora. Specifically, we take monolingual data from parallel corpora of WMT19 (Barrault et al., 2019) Chinese-English (zh-en) and Russian-English (ru-en) as training data. We use newsdev2017 for both sides of zh-en and newstest2016 for both sides of ru-en as development data. We concatenate newstest2017 and newstest2018 as test data for all four sides. The sizes of the corpora (WMT) are summarized in Table 2.

| | zh-en | ru-en |
|---|---|---|
| train | 26M | 25M |
| valid | 2K | 3K |
| test | 6K | 6K |

Table 2: Number of sentence pairs in WMT.

| | zh-en | | ru-en | |
|---|---|---|---|---|
| data | zh | en | ru | en |
| 100% | 40.9 | 135.0 | 181.0 | **145.7** |
| 90% | **36.9** | 133.5 | **176.7** | 147.6 |
| 80% | 37.1 | 132.9 | 180.6 | 149.3 |
| 70% | 38.7 | **131.4** | 178.6 | 151.1 |

Table 3: Test perplexities on WMT.

[4]No develepment set is available for STS, this is why we trained the GMM on the test set. For LM and MT experiments, `uniblock` is trained on the development sets.

| side | score | parallel text |
|---|---|---|
| zh | -48523.31 | リリスノト |
| en | 53.82 | Open Directory - Unix |
| zh | 67.71 | 健康、整洁、卫生和清洁技术组织 |
| en | -158.57 | Salubrité, propreté, hygiène et techniques d'assainissement |
| zh | 0.00 | 从系统目录→位置→传送我的位置... |
| en | 0.00 | From System Menu → Location → Send My Location... |
| zh | 36.01 | 在25℃下测定了样品的总交换容量和平衡等温线. |
| en | 40.65 | Cation exchange capacity and equilibria isotherm at 25 ℃ were determined. |
| zh | 68.63 | 财务政策及发展小组 |
| en | 53.82 | Financial Policy and Development Unit |

Table 4: Several text samples from the zh-en parallel corpus with `uniblock` scores. We find empirically: sentences with only foreign characters are scored very low; sentences with some foreign characters are scored low; sentences with characters in unseen blocks are scored 0; sentences with only few foreign characters are scored high; sentences with no illegal characters are scored very high. Foreign (or illegal) characters are underlined.

For each of the four LM subtasks, we train `uniblock` on the corresponding development set and filter out 10%, 20% and 30% of the raw training corpus. We train 2-layer LSTM language models with hidden dimensions being 1024. For the zh subtask, we train character-level language models and include all seen characters in the vocabulary. For the other 3 subtasks, top 100k frequent words are used as the vocabulary and the rest of the words are replaced with unknown tokens. All LM experiments are conducted using the RETURNN framework (Doetsch et al., 2017; Zeyer et al., 2018). As shown in Table 3, we see consistent improvements in the `uniblock`-filtered setups of zh, ru and en under zh-en. For en under ru-en, the baseline system outperforms `uniblock`-filtered ones.

### 4.3 Machine Translation

We train MT systems to show the simplicity and effectiveness of our method in a more challenging task. We use the same parallel corpora as in Section 4.2 and train systems in four directions: zh-en, en-zh, ru-en and en-ru. The raw training corpora are filtered with `uniblock` as before. Before examining the performance of the translation systems, we first manually examine some filtered text samples. Five parallel text samples with `uniblock` scores are shown in Table 4. We observe that the trained GMM works properly and indeed assigns low scores to unreasonable or "dirty" sentences. We believe that `uniblock` is superior to other rule-based methods because the relative quality of the sentences is modeled. For example, a hard rule may eliminate a complete sentence pair encountering even one ℃ symbol, while `uniblock` only gives a small penalization due to the fact that one occurrence of the symbol ℃ does not alter the count-based feature vector $e$ much.

| | zh-en | | en-zh | |
|---|---|---|---|---|
| data | test17 | test18 | test17 | test18 |
| 100% | 25.0 | 24.5 | 30.1 | 33.0 |
| 90% | **25.2** | **25.6** | **30.9** | 33.1 |
| 80% | 24.3 | 25.3 | 30.3 | **33.2** |
| 70% | 24.3 | 24.8 | 30.2 | 33.0 |

| | ru-en | | en-ru | |
|---|---|---|---|---|
| data | test17 | test18 | test17 | test18 |
| 100% | 32.9 | 28.3 | 26.9 | 23.3 |
| 90% | **33.5** | **29.3** | **27.8** | **23.9** |
| 80% | 33.3 | 28.9 | 27.2 | 23.6 |
| 70% | 32.9 | 28.1 | 26.6 | 23.2 |

Table 5: Case-sensitive BLEU(%) scores on four different translation tasks.

For all MT systems, we perform minimum preprocessing to handle the whitespaces and casing, and use SentencePiece (Kudo and Richardson, 2018) to obtain the subword units. All models adopt the Transformer (Vaswani et al., 2017) architecture and use the exact same hyperparameters as the base model. Trainings are done with the Sockeye (Hieber et al., 2017) toolkit and share the same optimization hyperparameters. No synthetic data is used and no ensembling is done during decoding. The only difference across the models is the training corpus - the baseline model uses the full parallel corpus, while the ones filtered

by `uniblock` use a subset of the full corpus. We take the checkpoint with the lowest perplexity on the validation set and report case-sensitive BLEU(%) (Papineni et al., 2002) scores on newstest2017 and newstest2018 using the sacreBLEU (Post, 2018) software. The translation qualities of the systems are shown in Table 5. As can be seen, without altering the architecture or the optimization process, only applying `uniblock` as a corpus filtering step leads to consistent improvements over the baseline systems for all four directions, up to 1.1 BLEU(%).

Linguistically, the zh-en and ru-en language pairs are rather distant. For more closely-related language pairs such as French and English, our method should also perform well. As shown in Table 4, a Chinese sentence could be corrupted with Japanese characters and an English sentence could be corrupted with French characters. For both cases, our method is able to discriminate the illegal characters and assign low scores to these sentences. Note that, our method would fail, if "clean" and "dirty" sentences share one exact same block. Because then after normalization all the feature vectors will be essentially the same and therefore indistinguishable. In this case, one should carefully design other features so that the GMM is able to assign meaningful scores. However, when "clean" and "dirty" sentences share the same set of blocks, our method still works fine because after normalization the empirical distributions are probably different.

## 5 Conclusion

In this work, we develop a simple and effective statistical method called `uniblock`. Using Unicode block information as feature vectors, a GMM is estimated with variational inference on some clean corpus, which can then be used to score and filter corpus. We release our implementation which supports parallel corpus filtering and different thresholding. Our experiments show concrete and consistent improvements in SA, LM and MT.

We believe that the method can be extended and improved by examining other dimensionality reduction methods and alternatives to GMM, and by introducing other heuristics into the feature vector, such as sentence length and punctuation marks[5].

[5]Common punctuation marks lie in the same Unicode block range U+0000..U+007F as the English alphabet and are currently not separated.


## Acknowledgements




This work has received funding from the European Research Council (ERC) (under the European Union's Horizon 2020 research and innovation programme, grant agreement No 694537, project "SEQCLAS") and the Deutsche Forschungsgemeinschaft (DFG; grant agreement NE 572/8-1, project "CoreTec"). The GPU computing cluster was supported by DFG (Deutsche Forschungsgemeinschaft) under grant INST 222/1168-1 FUGG.


## References


Tom Ash, Remi Francis, and Will Williams. 2018. The speechmatics parallel corpus filtering system for wmt18. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 853–859.

Loïc Barrault, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, Shervin Malmasi, Christof Monz, Mathias Müller, Santanu Pal, Matt Post, and Marcos Zampieri. 2019. Findings of the 2019 conference on machine translation (wmt19). In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 1–61, Florence, Italy. Association for Computational Linguistics.

Christopher M Bishop. 2006. *Pattern recognition and machine learning*. springer.

Patrick Doetsch, Albert Zeyer, Paul Voigtlaender, Ilia Kulikov, Ralf Schlüter, and Hermann Ney. 2017. Returnn: The rwth extensible training framework for universal recurrent neural networks. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5345–5349. IEEE.

Grant Erdmann and Jeremy Gwinnup. 2018. Coverage and cynicism: The afrl submission to the wmt 2018 parallel corpus filtering task. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 872–876.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12):2009.

Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, and Matt Post. 2017. Sockeye: A toolkit for neural machine translation. *arXiv preprint arXiv:1712.05690*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics.

Marcin Junczys-Dowmunt. 2018. Dual conditional cross-entropy filtering of noisy parallel corpora. *arXiv preprint arXiv:1809.00197*.

Huda Khayrallah and Philipp Koehn. 2018. On the impact of various types of noise on neural machine translation. *arXiv preprint arXiv:1805.12282*.

Huda Khayrallah, Hainan Xu, and Philipp Koehn. 2018. The jhu parallel corpus filtering systems for wmt 2018. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 896–899.

Philipp Koehn, Huda Khayrallah, Kenneth Heafield, and Mikel L Forcada. 2018. Findings of the wmt 2018 shared task on parallel corpus filtering. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 726–739.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.

Jun Lu, Xiaoyu Lv, Yangbin Shi, and Boxing Chen. 2018. Alibaba submission to the wmt18 parallel corpus filtering task. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 917–922.

Vassilis Papavassiliou, Sokratis Sofianopoulos, Prokopis Prokopidis, and Stelios Piperidis. 2018. The ilsp/arc submission to the wmt 2018 parallel corpus filtering shared task. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 928–933.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels. Association for Computational Linguistics.

Nick Rossenbach, Jan Rosendahl, Yunsu Kim, Miguel Graça, Aman Gokrani, and Hermann Ney. 2018. The rwth aachen university filtering system for the wmt 2018 parallel corpus filtering task. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 946–954.

The Unicode Consortium. 2019. The unicode standard, version 12.0.0. http://www.unicode.org/versions/Unicode12.0.0/.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Hainan Xu and Philipp Koehn. 2017. Zipporah: a fast and scalable data cleaning system for noisy web-crawled parallel corpora. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2945–2950.

Albert Zeyer, Tamer Alkhouli, and Hermann Ney. 2018. Returnn as a generic flexible neural toolkit with application to translation and speech recognition. In *Annual Meeting of the Assoc. for Computational Linguistics*, pages 128–133, Melbourne, Australia.

# Multilingual Word Translation using Auxiliary Languages

**Hagai Taitelbaum**
Faculty of Engineering
Bar-Ilan University, Israel
hagait62@gmail.com

**Gal Chechik**
The Gonda Brain Research Center
Bar-Ilan University, NVIDIA
gal.chechik@biu.ac.il

**Jacob Goldberger**
Faculty of Engineering
Bar-Ilan University, Israel
jacob.goldberger@biu.ac.il

## Abstract

Current multilingual word translation methods are focused on jointly learning mappings from each language to a shared space. The actual translation, however, is still performed as an isolated bilingual task. In this study we propose a multilingual translation procedure that uses all the learned mappings to translate a word from one language to another. For each source word, we first search for the most relevant languages. We then use the auxiliary translations to these languages to form an improved representation of the source word. Finally, this representation is used for the actual translation to the target language. Experiments on a standard multilingual word translation benchmark demonstrate that our model outperforms state of the art results.

## 1 Introduction

Monolingual continuous word embeddings are standard building blocks of many natural language tasks. The embedding spaces can exhibit similar structures across languages. Several studies (Mikolov et al., 2013; Klementiev et al., 2012) proposed to exploit this similarity by learning a linear mapping from a source to a target embedding space, and demonstrated this approach on a word translation task. Xing et al. (2015) showed that using orthogonality matrices can significantly improve performance.

Bilingual embedding can be extended to a multilingual setup by jointly learning mappings from each monolingual space to a shared word space. In recent years several studies have proposed aligning multiple languages simultaneously in a shared space by enforcing (or at least encouraging) transitive relations between the mappings (Chen and Cardie, 2018; Kementchedjhieva et al., 2018; Alaux et al., 2019; Jawanpuria et al., 2019; Taitelbaum et al., 2019). Dealing with multiple languages simultaneously has been shown to improve performance on some bilingual tasks by using knowledge learned from other languages (Ammar et al., 2016; Duong et al., 2017).

Once the multilingual mappings are learned, one can infer word correspondences for words that are not in the initial lexicons. Previous multilingual methods have focused on training procedures that benefit from the multilingual setting. The actual translation, however, is still done as an isolated bilingual task. It makes eminent sense however to go beyond this stage and utilize the relations between all the languages at the inference phase as it is done in the mapping learning phase.

In this study, we propose a new inference method for multilingual word translation that uses all the learned mappings to translate a word from one language to another. For each source word we first search for the most relevant languages that help translate the source word to the target language. We then use the auxiliary translations to form an improved representation of the source word. Finally, this multilingual-dependent representation is used for the actual translation to the target language.

Our main contributions is twofold: first, a new word translation inference method, which takes advantage of the multilingual setup not only in the train phase, but also in test phase. Second, evaluation on a recently-released multilingual word translation dataset on six languages (Lample et al., 2018) showing that our method outperforms state-of-the-art methods on this task.

## 2 Multilingual Word Mapping

In this section we review the concept of multilingual mapping to a shared space and in the next section we use it to form a multilingual translation method. Assume we are given a $d$-dimensional



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1330–1335, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

word embedding data from a set of $k$ languages. The task is to learn linear mappings between every pair of languages. In order to learn these mappings, we are also given a dictionary for each pair of languages, which contains pairs of corresponding words from the two languages. These dictionaries can be obtained in either a supervised or an unsupervised manner, and can also be created at each iteration of a dictionary refinement method (Artetxe et al., 2017; Lample et al., 2018). We can learn cross-language mappings independently from each source to each target language. This approach fails to benefit from the multilingual setup, and is very expensive because it requires to learn $k^2$ mappings, each with its own independent parameters. Another approach consists of choosing one language as a "pivot" and learning a mapping from each language to the pivot independently. This strategy, however, does not guarantee good indirect word translations between pairs of languages that do not include the pivot.

Recent studies proposed to jointly map all languages into a shared space. This approach was shown to outperform the above two approaches (Chen and Cardie, 2018). Let $T_1, \ldots, T_k$ be a set of mappings that correspond to the $k$ different languages. The mapping $T_i$ is used to translate the words from language $i$ to a shared space that can be viewed as an embedding space of a universal language.

The translation matrices $T_1, ..., T_k$ can be found by minimizing the following mean-square error:

$$S(T_1, ..., T_k) = \sum_{i<j} \sum_{t} \|T_i x_{it} - T_j x_{jt}\|^2 \quad (1)$$

such that $x_{it}$ and $x_{jt}$ are embeddings of two corresponding words in languages $i$ and $j$ respectively. The optimal transformations map pairs of words with similar meanings to vectors in the shared space that are close to one another.

When more than two languages are involved, there is no closed-form solution for the global minimum of Eq. (1). Recently, several studies addressed this optimization challenge. (Kementchedjhieva et al., 2018; Chen and Cardie, 2018; Alaux et al., 2019; Taitelbaum et al., 2019).

## 3 Multilingual Translation Method

In this section we propose an inference procedure for multilingual translation that uses *all* the learned mapping $T_1, \ldots, T_k$ to translate from one language to another.

Before we describe our method, consider first a generic formulation of the inference procedure. The translation of a word embedding $x$ from language $i$ to a language $j$ is obtained by:

$$\hat{y} = \arg\max_{y \in V_j} sim(x, y) \quad (2)$$

where $V_j$ is the vocabulary of language $j$. $sim(x, y)$ can be, for example, the cosine similarity in the embedded space: $sim(x, y) = cos(T_i x, T_j y)$. It is commonly observed that inference using the nearest embedded neighbor suffers from the *hubness problem* (Dinu and Baroni, 2014). Hubs are words that appear too frequently in the neighborhoods of other words. To mitigate this effect, one can simply replace the cosine similarity by another criterion, such as *Inverted Softmax* (ISF) (Smith et al., 2017) or *Cross-domain Similarity Local Scaling* (CSLS) (Lample et al., 2018). Following recent bilingual and multilingual translation studies (e.g., Lample et al. (2018); Alaux et al. (2019); Chen and Cardie (2018); Joulin et al. (2018)), this study uses the CSLS metric, namely, for $x \in V_i, y \in V_j$:

$$sim(x, y) = \text{CSLS}(T_i x, T_j y). \quad (3)$$

The CSLS similarity is calculated as follows:

$$\text{CSLS}(z, w) = 2\cos(z, w)$$

$$-\frac{1}{n} \sum_{z' \in N_z(w)} \cos(z', w) - \frac{1}{n} \sum_{w' \in N_w(z)} \cos(z, w')$$

where $cos$ is the cosine similarity, $N_z(w)$ is the set of $n$ nearest neighbors of the point $w$ in the first set of word vectors and $N_w(z)$ is similarly defined. In practice, we used $n = 10$.

To describe our multilingual word translation, note that current word translation methods take into account only the source and target languages at inference time. Since the translation is performed via the shared embedding space one can potentially design a better representation of the source word in the shared space to be translated into the target word. We can translate $x$ to all other languages (except the target language $j$): $y_m = \arg\max_{y \in V_m} \text{CSLS}(T_i x, T_m y)$ and then compute the average word in the shared space:

$$z = \frac{1}{k-1} \sum_{m \neq j} T_m y_m \quad (4)$$

(for the source word we set $y_i = x$). Here, all languages except the target are used as auxiliary

sources of information about the correct embedding in the shared space. Then, we use $z$ as a new multilingual representation of $T_i x$, and translate $x$ according to:

$$\hat{y} = \arg\max_{y \in V_j} \text{CSLS}(z, T_j y) \quad (5)$$

Unfortunately, using the average across all auxiliary languages, may hurt performance for some language pairs. For example, a German translation of a Spanish word may not help with translating that Spanish word to Portuguese. More generally, translations to auxiliary languages can yield words that are far from the source word in the shared space and therefore may lead to incorrect translation. Here we describe an approach to select the relevant auxiliary languages for a given source word and target language. The main idea is to apply CSLS to select those languages that would be helpful for translating the desired source word. Specifically, a language $m$ is selected as an auxiliary language only if the translated word $y_m$ is closer than the target word $y_j$ to the source word $x$ in the shared space. A language $m$ is thus included in the summation of Eq. (4) only if:

$$\text{CSLS}(T_i x, T_m y_m) > \text{CSLS}(T_i x, T_j y_j) \quad (6)$$

When averaging the auxiliary translations all these languages are equally weighted. Because the source word is more important than its auxiliary translations, we set the weight of the source $T_i x$ to be sum of all the weights of the auxiliary words. The proposed *Multilingual Word Translation* (MWT) procedure is depicted in Algo. 1

**Algorithm 1** Multilingual Word Translation

**Required:** A set of mappings $T_1, .., T_k$.
**Task:** Translate the word $x \in V_i$ to language $j$.

**for** $m = 1, ..., k$ **do**
  $y_m = \arg\max_{y \in V_m} \text{CSLS}(T_i x, T_m y)$
  $c_m = \text{CSLS}(T_i x, T_m y_m)$
**end**
$z = T_i x$
$S = \{m \in \{1, ..., k\} \backslash \{i, j\} \,|\, c_m > c_j\}$
**if** $|S| \neq 0$ **then**
  $z \leftarrow \frac{1}{2} T_i x + \frac{1}{2|S|} \sum_{m \in S} T_m y_m$
**end**
$\hat{y} = \arg\max_{y \in V_j} \mathit{CSLS}(z, T_j y)$
**return** $\hat{y}$

Several studies have recently proposed using word vector averaging of the source and target embeddings as an improved shared word representation. This can be done when the monolingual word embeddings are mapped to the same space (Doval et al., 2018). Meta-embedding using several embeddings of the same language can be achieved even without mapping the embeddings to a shared space (Coates and Bollegala, 2018). In this study, we also built an improved source word representation by averaging. However, the averaging is done with translations of the source word to suitable auxiliary languages instead of the target word.

## 4 Experiments

To evaluate the proposed MWT algorithm, we used a recently released multilingual word translation dataset (MUSE) in six European languages: English, German, French, Spanish, Italian and Portuguese (Lample et al., 2018)[1]. In addition, we conducted another experiment mixing European (English, German, French, Spanish and Italian) together with Asian languages (Japanese , Chinese and Korean). This experiment demonstrates the power of MWT even for distant languages. The available dictionaries in MUSE dataset are between the European languages, and between English and each of the Asian languages. For any available pair of languages, a ground-truth bilingual dictionary is provided with a train-test split of 5000 and 1500 unique source words, respectively. All systems are tested on the 1500 test word pairs for each pair of languages.

**Monolingual Embeddings.** Pre-trained 300d fastText (monolingual) embeddings[2] (Bojanowski et al., 2017) trained on the Wikipedia corpus. These embeddings are popular among word translation systems (e.g., Lample et al. (2018); Alaux et al. (2019); Chen and Cardie (2018); Hoshen and Wolf (2018); Joulin et al. (2018); Kementchedjhieva et al. (2018).

**Implementation details.** For the six European languages experiment, the multilingual mapping set was trained using the state-of-the-art unsupervised *Multilingual Adversarial Training* (MAT) + *Multilingual Pseudo-Supervised Refinement* (MPSR) method (Chen and Cardie, 2018). We used their source code[3], with their default

[1] https://github.com/facebookresearch/ MUSE
[2] https://github.com/facebookresearch/ fastText
[3] https://github.com/ccsasuke/umwe

| | en-de | en-fr | en-es | en-it | en-pt | de-en | de-fr | de-es | de-it | de-pt | fr-en | fr-de | fr-es | fr-it | fr-pt | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BI | 75.1 | 82.7 | 82.6 | 79.1 | 81.8 | 73.4 | 76.9 | 68.4 | 72.2 | 63.3 | 82.0 | 71.0 | 84.1 | 83.6 | 80.3 | |
| NT | 74.7 | 82.8 | 82.7 | 80.5 | 81.5 | 72.7 | 77.3 | **72.5** | 76.5 | 69.7 | 81.6 | 71.3 | 84.5 | 84.2 | 82.3 | |
| CNT | 75.1 | **83.4** | **83.2** | **81.1** | 81.9 | 74.7 | 78.5 | 72.3 | **77.1** | 69.7 | 82.5 | 71.5 | 84.5 | **84.6** | 82.1 | |
| CAT | **75.5** | 83.1 | 83.1 | 80.9 | **82.1** | **74.9** | **78.9** | 72.2 | 76.3 | **70.1** | **83.3** | **73.1** | **84.6** | 84.4 | **82.7** | |
| | es-en | es-de | es-fr | es-it | es-pt | it-en | it-de | it-fr | it-es | it-pt | pt-en | pt-de | pt-fr | pt-es | pt-it | avg |
| BI | **83.5** | 68.7 | 86.9 | 85.0 | **87.9** | 77.7 | 68.9 | 88.1 | 88.5 | 82.7 | 79.7 | 65.2 | 86.3 | **92.8** | 82.3 | 79.4 |
| NT | 82.4 | 70.3 | **88.7** | 84.8 | 86.5 | 79.5 | 71.3 | 88.7 | 89.5 | **84.7** | 80.9 | 69.3 | **89.2** | 91.1 | 85.1 | 80.6 |
| CNT | 82.7 | 70.3 | 88.6 | 85.1 | 87.2 | 79.6 | 71.7 | 89.1 | **89.9** | **84.7** | 81.1 | 69.2 | 89.1 | 92.3 | **85.2** | 80.9 |
| CAT | 83.0 | **71.6** | **88.7** | **86.1** | 87.1 | **79.7** | **72.5** | **89.2** | 89.8 | **84.7** | **81.5** | **69.7** | 88.2 | 92.3 | **85.2** | **81.1** |

Table 1: Multilingual word translation results for English, German, French, Spanish, Italian and Portuguese. The reported numbers are precision@1 in percentage.

| **source word** | **direct (BI)** | **multilingual (CAT)** | **auxiliary words** |
|---|---|---|---|
| pace (en) | veloz (es) | ritmo (es) | ritmo (pt), tempo (de) |
| Cornell (en) | Harvard (it) | Cornell (it) | Cornell (de), Cornell (fr), Cornell (es) |
| lens (en) | pentaprisma (it) | lente (it) | lente (es), lente (pt), linse (de) |

Table 2: Examples of erroneous translation of BI corrected by CAT, and its relevant auxiliary words.

hyper-parameters, and got similar results to the reported results (+0.1% in average). For the European-Asian experiment, MAT failed to converge for some language pairs, so the multilingual mapping set was trained using supervised MPSR, where the supervision was obtained by pairs of words with identical string matching. For each experiment, we used the same mappings for all the methods we compare. Our code and mapping matrices will be publicly available.

**Compared methods.** All methods retrieve word translations using their CSLS similarity in the learned embedding space.

**(1) BI (Bilingual Inference).** A standard inference process which does not take the multilingual setup into account, as in Chen and Cardie (2018).

We implemented 3 translation variants using auxiliary languages:

**(2) NT (Nearest Translation).** Average the source word with the closest auxiliary translation.

**(3) CNT (Conditional Nearest Translation).** Only average the source word with the closest auxiliary translation, if it is closer than the translation to the target language. In fact, CNT chooses for each source word one of {BI, NT} depends on whether the closest auxiliary translation is closer than the target translation (NT) or not (BI).

**(4) CAT (Conditional All Translations).** Weighted average of the source word and all auxiliary translations, that are closer than the target language translation. CAT is formally described in Algorithm box 1.

**Results.** Table 1 presents detailed results for all 30 language pairs and the average results. It shows that using all relevant auxiliary languages (CAT) increases performance significantly (+1.7% on average, top method in 19/30 tasks, $p < 0.001$ [4]). The largest performance boost of CAT over BI was in languages pairs involving *German* (+3.17% on average), which is the most distant language in this set of languages, thus, gains a lot from using other languages. This was found in particular for the translations between German and Portuguese (*de-pt*: +6.8%, *pt-de*: +4.5%), which are the most distant languages in this language set. This suggests that using MWT for distant languages may help. However, for close languages pairs the best way is still to translate directly (BI), as can be seen for Spanish and Portuguese. Table 2 presents three examples of erroneous bilingual translations that were corrected using auxiliary languages.

A similar behaviour is shown in the European-Asian languages (Table 3). CAT seems to improve word translation, especially over BI (+1.6% on average, top method in 18/26 tasks, $p < 0.001$). Moreover, CAT especially improves word translations between English and the Asian languages

[4] We performed a statistical significance test as in (Glavas et al., 2019) in addition to a statistical significance test over the accuracies obtained from each language pair.

| | en-de | en-fr | en-es | en-it | en-ja | en-zh | en-ko | de-en | de-fr | de-es | de-it | fr-en | fr-de | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BI | 75.1 | 82.3 | 82.2 | 78.5 | 27.1 | 30.7 | 28.8 | 73.5 | 76.0 | 68.4 | 72.2 | 82.3 | 70.3 | |
| NT | 74.7 | 81.9 | 81.6 | 79.6 | 28.4 | 32.6 | 30.0 | 72.7 | 76.9 | 71.9 | 75.4 | 81.5 | 71.1 | |
| CNT | **75.3** | 82.4 | 82.3 | 80.0 | 28.4 | 32.7 | 30.0 | **74.4** | 77.9 | 72.1 | **76.3** | 82.1 | 71.4 | |
| CAT | 74.8 | **82.5** | **82.5** | **80.3** | **28.9** | **33.3** | **30.2** | 74.0 | **78.1** | **73.3** | 75.4 | **82.5** | **73.1** | |
| | fr-es | fr-it | es-en | es-de | es-fr | es-it | it-en | it-de | it-fr | it-es | ja-en | zh-en | ko-en | avg |
| BI | 84.1 | 82.9 | **83.1** | 68.1 | 86.7 | 84.5 | 77.7 | 69.0 | 88.3 | 88.4 | 17.6 | 24.0 | 31.1 | 66.7 |
| NT | 84.3 | 83.1 | 81.6 | 70.9 | 87.6 | 83.9 | **79.3** | 70.9 | 88.1 | 88.0 | 14.1 | 25.8 | 35.1 | 67.3 |
| CNT | 84.7 | **83.7** | 82.2 | **71.0** | **88.1** | 84.7 | **79.3** | 71.0 | **88.7** | 88.9 | 16.7 | **27.1** | 35.9 | 68.0 |
| CAT | **84.9** | **83.7** | 81.9 | 70.9 | 87.9 | **85.3** | 79.1 | **71.6** | **88.7** | **89.2** | **18.5** | 26.9 | **37.5** | **68.3** |

Table 3: Multilingual word translation results for English, German, French, Spanish, Italian, Japanese, Chinese and Korean. The reported numbers are precision@1 in percentage.

| → | en | de | fr | es | it | pt |
|---|---|---|---|---|---|---|
| en | | es (29%) | – (29%) | – (29%) | fr (28%) | fr (29%) |
| de | – (36%) | | en (37%) | en (34%) | en (37%) | en (37%) |
| fr | it (29%) | it (27%) | | – (29%) | – (32%) | it (30%) |
| es | pt (44%) | pt (40%) | pt (39%) | | pt (43%) | – (47%) |
| it | fr (33%) | es (36%) | es (35%) | – (35%) | | es (34%) |
| pt | es (59%) | es (55%) | es (58%) | – (62%) | es (57%) | |

Table 4: Auxiliary languages that are most frequently selected by the CNT method ('–' is none), for every pair of source and target languages. Value in parenthesis denotes how often that auxiliary language was selected.

(+1.93% in average from English and +3.4% to English).

We next show more detailed analysis for CNT, when using at most one auxiliary language for the six European languages experiment. Table 4 shows auxiliary language that is most commonly selected, for each pair of source-target languages. Interestingly, Spanish and Portuguese often help each other. Also, German often uses English as an auxiliary language for translating better into all other languages.

For CAT, each source word may use a different number of auxiliary languages. We can see the number of auxiliary languages as a mean to qualitatively measure closeness of languages, by looking on the average number of auxiliary languages used for each source-target pair. We found Portuguese and Spanish to be the closest (*pt-es* 0.6), and German the farthest from them (*pt-de* 3.1, *es-de* 3.2).

## 5 Conclusion

We presented a general concept to improve the quality of bilingual word translation by using the translation of the source word to auxiliary languages. We discussed several variants for deciding which and how many languages should be used as suitable auxiliary languages. The same translation principle can be used in the dictionary refinement step of the mapping training process.

## References


Jean Alaux, Edouard Grave, Marco Cuturi, and Armand Joulin. 2019. Unsupervised hyperalignment for multilingual word embeddings. In *International Conference on Learning Representations*.

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A Smith. 2016. Massively multilingual word embeddings. *arXiv preprint arXiv:1602.01925*.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *Annual Meeting of the Association for Computational Linguistics*, pages 451–462. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Xilun Chen and Claire Cardie. 2018. Unsupervised multilingual word embeddings. In *Conference on Empirical Methods in Natural Language Processing*.

Joshua Coates and Danushka Bollegala. 2018. Frustratingly easy meta-embeddingcomputing metaembeddings by averaging source word embeddings. In *Conference of the North American Chapter of the Association for Computational - Linguistics Human Language Technologies*.

Georgiana Dinu and Marco Baroni. 2014. Improving zero-shot learning by mitigating the hubness problem. *CoRR*, abs/1412.6568.

Yerai Doval, Jose Camacho-Collados, Luis Espinosa-Anke, and Steven Schockaert. 2018. Improving cross-lingual word embeddings by meeting in the middle. In *Conference on Empirical Methods in Natural Language Processing*.

Long Duong, Hiroshi Kanayama, Tengfei Ma, Steven Bird, and Trevor Coh. 2017. Multilingual training of crosslingual word embeddings. In *The Conference of the European Chapter of the Association for Computational Linguistics*.

Goran Glavas, Robert Litschko, Sebastian Ruder, and Ivan Vulic. 2019. How to (properly) evaluate cross-lingual word embeddings: On strong baselines, comparative analyses, and some misconceptions. *arXiv preprint arXiv:1902.00508*.

Yedid Hoshen and Lior Wolf. 2018. Non-adversarial unsupervised word translation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Pratik Jawanpuria, Arjun Balgovind, Anoop Kunchukuttan, and Bamdev Mishra. 2019. Learning multilingual word embeddings in latent metric space: a geometric approach. *Transactions of the Association for Computational Linguistics*, 7:107–120.

Armand Joulin, Piotr Bojanowski, Tomas Mikolov, Herve Jegou, , and Edouard Grave. 2018. Loss in translation: Learning bilingual word mapping with a retrieval criterion. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Yova Kementchedjhieva, Sebastian Ruder, Ryan Cotterell, and Anders Søgaard. 2018. Generalizing procrustes analysis for better bilingual dictionary induction. In *CONLL*.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing crosslingual distributed representations of words. In *COLING*, pages 1459–1474.

Guillaume Lample, Alexis Conneau, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. In *Int. Conference on Learning Representations.*

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Samuel L. Smith, David H. P. Turban, Steven Hamblin, and Nils Y. Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. *CoRR*, abs/1702.03859.

Hagai Taitelbaum, Gal Chechik, and Jacob Goldberger. 2019. A multi-pairwise extension of procrustes analysis for multilingual word translation. In *Conference on Empirical Methods in Natural Language Processing*.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1006–1011.

# Towards Better Modeling Hierarchical Structure for Self-Attention with Ordered Neurons

**Jie Hao**[*]
Florida State University
haoj8711@gmail.com

**Xing Wang**
Tencent AI Lab
brightxwang@tencent.com

**Shuming Shi**
Tencent AI Lab
shumingshi@tencent.com

**Jinfeng Zhang**
Florida State University
jinfeng@stat.fsu.edu

**Zhaopeng Tu**
Tencent AI Lab
zptu@tencent.com

## Abstract

Recent studies have shown that a hybrid of self-attention networks (SANs) and recurrent neural networks (RNNs) outperforms both individual architectures, while not much is known about why the hybrid models work. With the belief that modeling hierarchical structure is an essential complementary between SANs and RNNs, we propose to further enhance the strength of hybrid models with an advanced variant of RNNs – *Ordered Neurons* LSTM (ON-LSTM, Shen et al., 2019), which introduces a syntax-oriented inductive bias to perform tree-like composition. Experimental results on the benchmark machine translation task show that the proposed approach outperforms both individual architectures and a standard hybrid model. Further analyses on targeted linguistic evaluation and logical inference tasks demonstrate that the proposed approach indeed benefits from a better modeling of hierarchical structure.

## 1 Introduction

Self-attention networks (SANs, Lin et al., 2017) have advanced the state of the art on a variety of natural language processing (NLP) tasks, such as machine translation (Vaswani et al., 2017), semantic role labelling (Tan et al., 2018), and language representations (Devlin et al., 2018). However, a previous study empirically reveals that the hierarchical structure of the input sentence, which is essential for language understanding, is not well modeled by SANs (Tran et al., 2018). Recently, hybrid models which combine the strengths of SANs and recurrent neural networks (RNNs) have outperformed both individual architectures on a machine translation task (Chen et al., 2018). We attribute the improvement to that RNNs complement SANs on the representation limitation of hierarchical structure, which is exactly the strength of RNNs (Tran et al., 2018).

Starting with this intuition, we propose to further enhance the representational power of hybrid models with an advanced RNNs variant – *Ordered Neurons* LSTM (ON-LSTM, Shen et al., 2019). ON-LSTM is better at modeling hierarchical structure by introducing a syntax-oriented inductive bias, which enables RNNs to perform tree-like composition by controlling the update frequency of neurons. Specifically, we stack SANs encoder on top of ON-LSTM encoder (*cascaded encoder*). SANs encoder is able to extract richer representations from the input augmented with structure context. To reinforce the strength of modeling hierarchical structure, we propose to simultaneously expose both types of signals by explicitly combining outputs of the SANs and ON-LSTM encoders.

We validate our hypothesis across a range of tasks, including machine translation, targeted linguistic evaluation, and logical inference. While machine translation is a benchmark task for deep learning models, the last two tasks focus on evaluating how much structure information is encoded in the learned representations. Experimental results show that the proposed approach consistently improves performances in all tasks, and modeling hierarchical structure is indeed an essential complementary between SANs and RNNs.

The contributions of this paper are:

- We empirically demonstrate that a better modeling of hierarchical structure is an essential strength of hybrid models over the vanilla SANs.
- Our study proves that the idea of augmenting RNNs with ordered neurons (Shen et al., 2019) produces promising improvement on machine translation, which is one potential criticism of ON-LSTM.

[*]Work done when interning at Tencent AI Lab.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1336–1341,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

## 2 Approach

Partially motivated by Wang et al. (2016) and Chen et al. (2018), we stack a SANs encoder on top of a RNNs encoder to form a cascaded encoder. In the cascaded encoder, hierarchical structure modeling is enhanced in the bottom RNNs encoder, based on which SANs encoder is able to extract representations with richer hierarchical information. Let $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ be the input sequence, the representation of the cascaded encoder is calculated by

$$\mathbf{H}^K_{\text{RNNs}} = \text{ENC}_{\text{RNNs}}(\mathbf{X}), \quad (1)$$
$$\mathbf{H}^L_{\text{SANs}} = \text{ENC}_{\text{SANs}}(\mathbf{H}^K_{\text{RNNs}}), \quad (2)$$

where $\text{ENC}_{\text{RNNs}}(\cdot)$ is a $K$-layer RNNs encoder that reads the input sequence, and $\text{ENC}_{\text{SANs}}(\cdot)$ is a $L$-layer SANs encoder that takes the output of RNNs encoder as input.

In this work, we replace the standard RNNs with recently proposed ON-LSTM for better modeling of hierarchical structure, and directly combine the two encoder outputs to build even richer representations, as described below.

**Modeling Hierarchical Structure with Ordered Neurons** ON-LSTM introduces a new syntax-oriented inductive bias – *Ordered Neurons*, which enables LSTM models to perform tree-like composition without breaking its sequential form (Shen et al., 2019). Ordered neurons enables dynamic allocation of neurons to represent different time-scale dependencies by controlling the update frequency of neurons. The assumption behind ordered neurons is that some neurons always update more (or less) frequently than the others, and that order is pre-determined as part of the model architecture. Formally, ON-LSTM introduces novel ordered neuron rules to update cell state:

$$w_t = \tilde{f}_t \circ \tilde{i}_t, \quad (3)$$
$$\hat{f}_t = f_t \circ w_t + (\tilde{f}_t - w_t), \quad (4)$$
$$\hat{i}_t = i_t \circ w_t + (\tilde{i}_t - w_t), \quad (5)$$
$$c_t = \hat{f}_t \circ c_{t-1} + \hat{i}_t \circ \hat{c}_t, \quad (6)$$

where forget gate $f_t$, input gate $i_t$ and state $\hat{c}_t$ are same as that in the standard LSTM (Hochreiter and Schmidhuber, 1997). The *master forget gate* $\tilde{f}_t$ and the *master input gate* $\tilde{i}_t$ are newly introduced to control the erasing and the writing behaviors respectively. $w_t$ indicates the overlap, and when the overlap exists ($\exists k, w_{tk} > 0$), the corresponding neurons are further controlled by the standard gates $f_t$ and $i_t$.

An ideal master gate is in binary format such as $(0, 0, 1, 1, 1)$, which splits the cell state into two continuous parts: 0-part and 1-part. The neurons corresponding to 0-part and 1-part are updated with more and less frequencies separately, so that the information in 0-part neurons will only keep a few time steps, while the information in 1-part neurons will last for more time steps. Since such binary gates are not differentiable, the goal turns to find the splitting point $d$ (the index of the first 1 in the ideal master gate). To this end, Shen et al. (2019) introduced a new activation function:

$$\text{CU}(\cdot) = \text{CUMSUM}(softmax(\cdot)), \quad (7)$$

where $softmax(\cdot)$ produces a probability distribution (e.g.$(0.1, 0.2, 0.4, 0.2, 0.1)$) to indicate the probability of each position being the splitting point $d$. CUMSUM is the cumulative probability distribution, in which the $k$-th probability refers to the probability that $d$ falls within the first $k$ positions. The output for the above example is $(0.1, 0.3, 0.7, 0.9, 1.0)$, in which different values denotes different update frequencies. It also equals to the probability of each position's value being 1 in the ideal master gate. Since this ideal master gate is binary, $\text{CU}(\cdot)$ is the expectation of the ideal master gate.

Based on this activation function, the master gates are defined as

$$\tilde{f}_t = \text{CU}_f(\mathbf{x}_t, \mathbf{h}_{t-1}), \quad (8)$$
$$\tilde{i}_t = 1 - \text{CU}_i(\mathbf{x}_t, \mathbf{h}_{t-1}), \quad (9)$$

where $\mathbf{x}_t$ is the current input and $\mathbf{h}_{t-1}$ is the hidden state of previous step. $\text{CU}_f$ and $\text{CU}_i$ are two individual activation functions with their own trainable parameters.

**Short-Cut Connection** Inspired by previous work on exploiting deep representations (Peters et al., 2018; Dou et al., 2018), we propose to simultaneously expose both types of signals by explicitly combining them with a simple *short-cut connection* (He et al., 2016).

Similar to positional encoding injection in Transformer (Vaswani et al., 2017), we add the output of the ON-LSTM encoder to the output of SANs encoder:

$$\widehat{\mathbf{H}} = \mathbf{H}^K_{\text{ON-LSTM}} + \mathbf{H}^L_{\text{SANs}}, \quad (10)$$

| # | Encoder Architecture | Para. | BLEU |
|---|---|---|---|
| | *Base Model* | | |
| 1 | 6L SANs | 88M | 27.31 |
| 2 | 6L LSTM | 97M | 27.23 |
| 3 | 6L ON-LSTM | 110M | 27.44 |
| 4 | 6L LSTM + 4L SANs | 104M | 27.78$^{\uparrow}$ |
| 5 | 6L ON-LSTM + 4L SANs | 123M | 28.27$^{\Uparrow}$ |
| 6 | 3L ON-LSTM + 3L SANs | 99M | 28.21$^{\Uparrow}$ |
| 7 | + Short-Cut | 99M | **28.37**$^{\Uparrow}$ |
| | *Big Model* | | |
| 8 | 6L SANs | 264M | 28.58 |
| 9 | Hybrid Model + Short-Cut | 308M | **29.30**$^{\Uparrow}$ |

Table 1: Case-sensitive BLEU scores on the WMT14 English⇒German translation task. "↑ / ⇑": significant over the conventional self-attention counterpart ($p < 0.05/0.01$), tested by bootstrap resampling. "6L SANs" is the state-of-the-art Transformer model. "nL LSTM + mL SANs" denotes stacking n LSTM layers and m SANs layers subsequently. "Hybrid Model" denotes "3L ON-LSTM + 3L SANs".

where $\mathbf{H}^{K}_{\text{ON-LSTM}} \in \mathbb{R}^{N\times d}$ is the output of ON-LSTM encoder, and $\mathbf{H}^{L}_{\text{SANs}} \in \mathbb{R}^{N\times d}$ is output of SANs encoder.

## 3 Experiments

We chose machine translation, targeted linguistic evaluation and logical inference tasks to conduct experiments in this work. The first and the second tasks evaluate and analyze models as the hierarchical structure is an inherent attribute for natural language. The third task aims to directly evaluate the effects of hierarchical structure modeling on artificial language.

### 3.1 Machine Translation

For machine translation, we used the benchmark WMT14 English⇒German dataset. Sentences were encoded using byte-pair encoding (BPE) with 32K word-piece vocabulary (Sennrich et al., 2016). We implemented the proposed approaches on top of TRANSFORMER (Vaswani et al., 2017) – a state-of-the-art SANs-based model on machine translation, and followed the setting in previous work (Vaswani et al., 2017) to train the models, and reproduced their reported results. We tested on both the *Base* and *Big* models which differ at hidden size (512 vs. 1024), filter size (2048 vs. 4096) and number of attention heads (8 vs. 16). All the model variants were implemented on the encoder. The implementation details are introduced in Appendix A.1. Table 1 lists the results.

| # | Encoder Architecture | Para. | BLEU |
|---|---|---|---|
| 1 | 3L ON-LSTM → 3L SANs | 99M | 28.21 |
| 2 | 3L SANs → 3L ON-LSTM | 99M | 27.39 |
| 3 | 8L LSTM | 102.2M | 27.25 |
| 4 | 10L SANs | 100.6M | 27.76 |

Table 2: Results for encoder strategies. Case-sensitive BLEU scores on the WMT14 English⇒German translation task. "A → B" denotes stacking B on the top of A. The model in Row 1 is the hybrid model in Table 1.

**Baselines** (Rows 1-3) Following Chen et al. (2018), the three baselines are implemented with the same framework and optimization techniques as used in Vaswani et al. (2017). The difference between them is that they adopt SANs, LSTM and ON-LSTM as basic building blocks respectively. As seen, the three architectures achieve similar performances for their unique representational powers.

**Hybrid Models** (Rows 4-7) We first followed Chen et al. (2018) to stack 6 RNNs layers and 4 SANs layers subsequently (Row 4), which consistently outperforms the individual models. This is consistent with results reported by Chen et al. (2018). In this setting, the ON-LSTM model significantly outperforms its LSTM counterpart (Row 5), and reducing the encoder depth can still maintain the performance (Row 6). We attribute these to the strength of ON-LSTM on modeling hierarchical structure, which we believe is an essential complementarity between SANs and RNNs. In addition, the Short-Cut connection combination strategy improves translation performances by providing richer representations (Row 7).

**Stronger Baseline** (Rows 8-9) We finally conducted experiments on a stronger baseline – the TRANSFORMER-BIG model (Row 8), which outperforms its TRANSFORMER-BASE counterpart (Row 1) by 1.27 BLEU points. As seen, our model consistently improves performance over the stronger baseline by 0.72 BLEU points, demonstrating the effectiveness and universality of the proposed approach.

**Assessing Encoder Strategies** We first investigate the encoder stack strategies on different stack orders. From Table 2, to compare with the proposed hybrid model, we stack 3-layers ON-LSTM on the top of 3-layers SANs (Row 2). It performs worse than the strategy in the proposed hybrid model. The result support the viewpoint that the

| Task | S | O | Hybrid + Short-Cut | | |
|---|---|---|---|---|---|
| | Final | Final | $\mathbf{H}_O$ | $\mathbf{H}_S$ | Final |
| *Surface Tasks* | | | | | |
| SeLen | *92.71* | 90.70 | 91.94 | 89.50 | 89.86 |
| WC | 81.79 | 76.42 | *90.38* | 79.10 | 80.37 |
| Avg | 87.25 | 83.56 | **91.16** | 84.30 | 85.12 |
| *Syntactic Tasks* | | | | | |
| TrDep | 44.78 | 52.58 | 51.19 | 52.55 | *53.28* |
| ToCo | 84.53 | 86.32 | 86.29 | *87.92* | 87.89 |
| BShif | 52.66 | *82.68* | 81.79 | 82.05 | 81.90 |
| Avg | 60.66 | 73.86 | 73.09 | 74.17 | **74.36** |
| *Semantic Tasks* | | | | | |
| Tense | 84.76 | 86.00 | 83.88 | *86.05* | 85.91 |
| SubN | 85.18 | 85.44 | 85.56 | 84.59 | *85.81* |
| ObjN | 81.45 | *86.78* | 85.72 | 85.80 | 85.38 |
| SoMo | 49.87 | 49.54 | 49.23 | 49.12 | *49.92* |
| CoIn | 68.97 | 72.03 | 72.06 | 72.05 | *72.23* |
| Avg | 74.05 | **75.96** | 75.29 | *75.52* | 75.85 |

Table 3: Performance on the linguistic probing tasks of evaluating linguistics embedded in the learned representations. "**S**" and "**O**" denote the SAN and ON-LSTM baseline models. "$\mathbf{H}_O$" and "$\mathbf{H}_S$" are respectively the outputs of the ON-LSTM encoder and the SAN encoder in the hybrid model, and "Final" denotes the final output exposed to decoder.

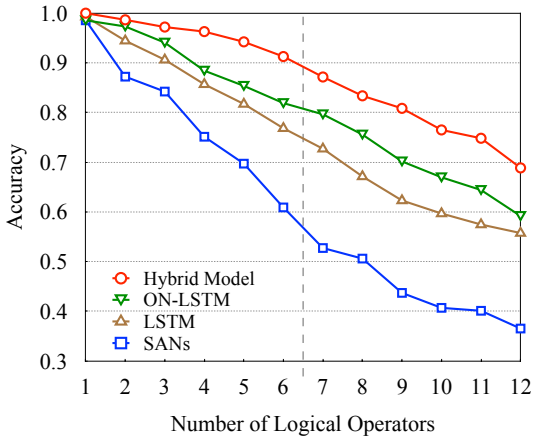


Figure 1: Accuracy of logical inference when training on logic data with at most 6 logical operators in the sequence.

SANs encoder is able to extract richer representations if the input is augmented with sequential context (Chen et al., 2018).

Moreover, to dispel the doubt that whether the improvement of hybrid model comes from the increasement of parameters. We investigate the 8-layers LSTM and 10-layers SANs encoders (Rows 3-4) which have more parameters compared with the proposed hybrid model. The results show that the hybrid model consistently outperforms these model variants with less parameters and the improvement should not be due to more parameters.

### 3.2 Targeted Linguistic Evaluation

To gain linguistic insights into the learned representations, we conducted probing tasks (Conneau et al., 2018) to evaluate linguistics knowledge embedded in the final encoding representation learned by model, as shown in Table 3. We evaluated SANs and proposed hybrid model with Short-Cut connection on these 10 targeted linguistic evaluation tasks. The tasks and model details are described in Appendix A.2.

Experimental results are presented in Table 3. Several observations can be made here. The proposed hybrid model with short-cut produces more informative representation in most tasks ("Final" in "S" vs. in "Hybrid+Short-Cut"), indicating that the effectiveness of the model. The only exception are surface tasks, which is consistent with the conclusion in Conneau et al. (2018): as a model captures deeper linguistic properties, it will tend to forget about these superficial features. Short-cut further improves the performance by providing richer representations ("$\mathbf{H}_S$" vs. "Final" in "Hybrid+Short-Cut"). Especially on syntactic tasks, our proposed model surpasses the baseline more than 13 points (74.36 vs. 60.66) on average, which again verifies that ON-LSTM enhance the strength of modeling hierarchical structure for self-attention.

### 3.3 Logical Inference

We also verified the model's performance in the logical inference task proposed by Bowman et al. (2015). This task is well suited to evaluate the ability of modeling hierarchical structure. Models need to learn the hierarchical and nested structures of language in order to predict accurate logical relations between sentences (Bowman et al., 2015; Tran et al., 2018; Shen et al., 2019). The artificial language of the task has six types of words {a, b, c, d, e, f} in the vocabulary and three logical operators {or, and, not}. The goal of the task is to predict one of seven logical relations between two given sentences. These seven relations are: two entailment types ($\sqsubset, \sqsupset$), equivalence ($\equiv$), exhaustive and non-exhaustive contradiction ($\wedge, |$), and semantic independence ($\#, \smile$).

We evaluated the SANs, LSTM, ON-LSTM and

proposed model. We followed Tran et al. (2018) to use two hidden layers with Short-Cut connection in all models. The model details and hyperparameters are described in Appendix A.3.

Figure 1 shows the results. The proposed hybrid model outperforms both the LSTM-based and the SANs-based baselines on all cases. Consistent with Shen et al. (2019), on the longer sequences ($\geq$ 7) that were not included during training, the proposed model also obtains the best performance and has a larger gap compared with other models than on the shorter sequences ($\leq$ 6), which verifies the proposed model is better at modeling more complex hierarchical structure in sequence. It also indicates that the hybrid model has a stronger generalization ability.

## 4 Related Work

**Improved Self-Attention Networks** Recently, there is a large body of work on improving SANs in various NLP tasks (Yang et al., 2018; Wu et al., 2018; Yang et al., 2019a,b; Guo et al., 2019; Wang et al., 2019a; Sukhbaatar et al., 2019), as well as image classification (Bello et al., 2019) and automatic speech recognition (Mohamed et al., 2019) tasks. In these works, several strategies are proposed to improve the utilize SANs with the enhancement of local and global information. In this work, we enhance the SANs with the On-Lstm to form a hybrid model (Chen et al., 2018), and thoroughly evaluate the performance on machine translation, targeted linguistic evaluation, and logical inference tasks.

**Structure Modeling for Neural Networks in NLP** Structure modeling in NLP has been studied for a long time as the natural language sentences inherently have hierarchical structures (Chomsky, 1965; Bever, 1970). With the emergence of deep learning, tree-based models have been proposed to integrate syntactic tree structure into Recursive Neural Networks (Socher et al., 2013), LSTMs (Tai et al., 2015), CNNs (Mou et al., 2016). As for SANs, Hao et al. (2019a), Ma et al. (2019) and Wang et al. (2019b) enhance the SANs with neural syntactic distance, multi-granularity attention scope and structural position representations, which are generated from the syntactic tree structures.

Closely related to our work, Hao et al. (2019b) find that the integration of the recurrence in SANs encoder can provide more syntactic structure features to the encoder representations. Our work follows this direction and empirically evaluates the structure modelling on the related tasks.

## 5 Conclusion

In this paper, we adopt the ON-LSTM, which models tree structure with a novel activation function and structured gating mechanism, as the RNNs counterpart to boost the hybrid model. We also propose a modification of the cascaded encoder by explicitly combining the outputs of individual components, to enhance the ability of hierarchical structure modeling in a hybrid model. Experimental results on machine translation, targeted linguistic evaluation and logical inference tasks show that the proposed models achieve better performances by modeling hierarchical structure of sequence.

## Acknowledgments

J.Z. was supported by the National Institute of General Medical Sciences of the National Institute of Health under award number R01GM126558. We thank the anonymous reviewers for their insightful comments.

## References

Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V Le. 2019. Attention augmented convolutional networks. *arXiv*.

Thomas G Bever. 1970. The cognitive basis for linguistic structures. *Cognition and the development of language*, 279(362).

Samuel R. Bowman, Christopher D. Manning, and Christopher Potts. 2015. Tree-structured composition in neural networks without tree-structured architectures. *NIPS Workshop on Cognitive Computation: Integrating Neural and Symbolic Approaches*.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Zhifeng Chen, Yonghui Wu, and Macduff Hughes. 2018. The best of both worlds: Combining recent advances in neural machine translation. In *ACL*.

Noam Chomsky. 1965. *Aspects of the Theory of Syntax*, volume 11. MIT press.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *ACL*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Zi-Yi Dou, Zhaopeng Tu, Xing Wang, Shuming Shi, and Tong Zhang. 2018. Exploiting deep representations for neural machine translation. In *EMNLP*.

Maosheng Guo, Yu Zhang, and Ting Liu. 2019. Gaussian transformer: a lightweight approach for natural language inference. In *AAAI*.

Jie Hao, Xing Wang, Shuming Shi, Jinfeng Zhang, and Zhaopeng Tu. 2019a. Multi-granularity self-attention for neural machine translation. In *EMNLP*.

Jie Hao, Xing Wang, Baosong Yang, Longyue Wang, Jinfeng Zhang, and Zhaopeng Tu. 2019b. Modeling recurrence for transformer. In *NAACL*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *ICLR*.

Chunpeng Ma, Akihiro Tamura, Masao Utiyama, Eiichiro Sumita, and Tiejun Zhao. 2019. Improving neural machine translation with neural syntactic distance. In *NAACL)*.

Abdelrahman Mohamed, Dmytro Okhonko, and Luke Zettlemoyer. 2019. Transformers with convolutional context for asr. *arXiv*.

Lili Mou, Ge Li, Lu Zhang, Tao Wang, and Zhi Jin. 2016. Convolutional neural networks over tree structures for programming language processing. In *AAAI*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL*.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. 2019. Ordered neurons: Integrating tree structures into recurrent neural networks. In *ICLR*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*.

Sainbayar Sukhbaatar, Edouard Grave, Piotr Bojanowski, and Armand Joulin. 2019. Adaptive attention span in transformers. In *ACL*.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *ACL-IJCNLP*.

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2018. Deep semantic role labeling with self-attention. In *AAAI*.

Ke Tran, Arianna Bisazza, and Christof Monz. 2018. The importance of being recurrent for modeling hierarchical structure. In *EMNLP*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. In *NIPS*.

Xing Wang, Zhaopeng Tu, Longyue Wang, and Shuming Shi. 2019a. Exploiting sentential context for neural machine translation. In *ACL*.

Xing Wang, Zhaopeng Tu, Longyue Wang, and Shuming Shi. 2019b. Self-attention networks with structural position encoding. In *EMNLP*.

Xingyou Wang, Weijie Jiang, and Zhiyong Luo. 2016. Combination of convolutional and recurrent neural network for sentiment analysis of short texts. In *COLING*.

Wei Wu, Houfeng Wang, Tianyu Liu, and Shuming Ma. 2018. Phrase-level self-attention networks for universal sentence encoding. In *ACL*.

Baosong Yang, Jian Li, Derek F. Wong, Lidia S. Chao, Xing Wang, and Zhaopeng Tu. 2019a. Context-aware self-attention networks. In *AAAI*.

Baosong Yang, Zhaopeng Tu, Derek F. Wong, Fandong Meng, Lidia S. Chao, and Tong Zhang. 2018. Modeling localness for self-attention networks. In *EMNLP*.

Baosong Yang, Longyue Wang, Derek F. Wong, Lidia S. Chao, and Zhaopeng Tu. 2019b. Convolutional self-attention networks. In *NAACL*.

# Vecalign: Improved Sentence Alignment in Linear Time and Space

**Brian Thompson**
Johns Hopkins University
brian.thompson@jhu.edu

**Philipp Koehn**
Johns Hopkins University
phi@jhu.edu

## Abstract

We introduce Vecalign, a novel bilingual sentence alignment method which is linear in time and space with respect to the number of sentences being aligned and which requires only bilingual sentence embeddings. On a standard German–French test set, Vecalign outperforms the previous state-of-the-art method (which has quadratic time complexity and requires a machine translation system) by 5 $F_1$ points. It substantially outperforms the popular Hunalign toolkit at recovering Bible verse alignments in medium- to low-resource language pairs, and it improves downstream MT quality by 1.7 and 1.6 BLEU in Sinhala→English and Nepali→English, respectively, compared to the Hunalign-based Paracrawl pipeline.

## 1 Introduction

Sentence alignment is the task of taking parallel documents, which have been split into sentences, and finding a bipartite graph which matches minimal groups of sentences that are translations of each other (see Figure 1). Following prior work, we assume non-crossing alignments but allow local sentence reordering within an alignment.

Sentence-aligned bitext is used to train nearly all machine translation (MT) systems. Alignment errors have been noted to have a small effect on statistical MT performance (Goutte et al., 2012). However, misaligned sentences have been shown to be much more detrimental to neural MT (NMT) (Khayrallah and Koehn, 2018).

Sentence alignment was a popular research topic in the early days of statistical MT, but received less attention once standard sentence-aligned parallel corpora became available. Interest in low-resource MT has led to a resurgence in data gathering methods (Buck and Koehn, 2016; Zweigenbaum et al., 2018; Koehn et al., 2019), but we find limited recent work on bilingual sentence alignment.

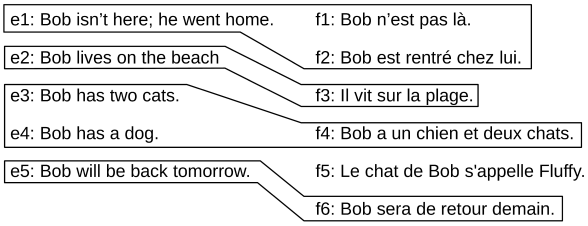


Figure 1: Sentence alignment takes sentences $e_1,...,e_N$ and $f_1,...,f_M$ and locates minimal groups of sentences which are translations of each other, in this case ($e_1$)-($f_1$, $f_2$), ($e_2$)-($f_3$), ($e_3$,$e_4$)-($f_4$), and ($e_5$)-($f_6$).

Automatic sentence alignment can be roughly decomposed into two parts:

1. A score function which takes one or more adjacent source sentences and one or more adjacent target sentences and returns a score indicating the likelihood that they are translations of each other;
2. An alignment algorithm which, using the score function above, takes in two documents and returns a hypothesis alignment.

We improve both parts, presenting (1) a novel scoring function based on normalized cosine distance between multilingual sentence embeddings, in conjunction with (2) a novel application of a dynamic programming approximation (Salvador and Chan, 2007) which makes our algorithm linear in time and space complexity with respect to the number of sentences being aligned. We release a toolkit containing our implementation.[1]

Our method outperforms previous state-of-the-art, which has quadratic complexity, indicating that our proposed score function outperforms prior work and the approximations we make in alignment are sufficiently accurate.

[1] https://github.com/thompsonb/vecalign



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1342–1348, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

## 2 Related Work

Early sentence aligners (Brown et al., 1991; Gale and Church, 1993) use scoring functions based only on the number of words or characters in each sentence and alignment algorithms based on dynamic programming (DP; Bellman, 1953). DP is O($NM$) time complexity, where $N$ and $M$ are the number of sentences in the source and target documents. Later work added lexical features and heuristics to speed up search, such as limiting the search space to be near the diagonal (Moore, 2002; Varga et al., 2007). More recent work introduced scoring methods that use MT to get both documents into the same language (Bleualign; Sennrich and Volk, 2010) or use pruned phrase tables from a statistical MT system (Coverage-Based; Gomes and Lopes, 2016). Both methods "anchor" high-probability 1–1 alignments in the search space and then fill in and refine alignments. Locating anchors is $O(NM)$ time complexity.

## 3 Method

We propose a novel sentence alignment scoring function based on the similarity of bilingual sentence embeddings. A distinct but non-obvious advantage of sentence embeddings is that blocks of sentences can be represented as the average of their sentence embeddings. The size of the resulting vector is not dependent on the number of sentence embeddings being averaged, thus the time/space cost of comparing the similarity of blocks of sentences *does not depend on the number of sentences being compared*. We show empirically (see § 4.2) that average embeddings for blocks of sentences are sufficient to produce approximate alignments, even in low-resource languages. This enables us to approximate DP in O($N+M$) in time and space.

### 3.1 Bilingual Sentence Embeddings

We propose to use the similarity between sentence embeddings as the scoring function for sentence alignment. Sentence embedding similarity has been shown effective at filtering out non-parallel sentences (Hassan et al., 2018; Chaudhary et al., 2019) and locating parallel sentences in comparable corpora (Guo et al., 2018). We use the publicly available LASER multilingual sentence embedding method (Artetxe and Schwenk, 2018) and model, which is pretrained on 93 languages. However, our method is not specific to LASER.

### 3.2 Scoring Function

Cosine similarity is an obvious choice for comparing embeddings but has been noted to be globally inconsistent due to "hubness" (Radovanović et al., 2010; Lazaridou et al., 2015). Guo et al. (2018) proposed a supervised training approach for calibration, and Artetxe and Schwenk (2019) proposed normalization using nearest neighbors. We propose normalizing instead with *randomly* selected embeddings as it has linear complexity. Sentence alignment seeks *minimal* parallel units, but we find that DP with cosine similarity favors many-to-many alignments (e.g. reporting a 3–3 alignment when it should report three 1–1 alignments). To remedy this issue, we scale the cost by the number of source and target sentences being considered in a given alignment. Our resulting scoring cost function is:

$$c(x,y) = \frac{(1-\cos(x,y))\,\mathrm{nSents}(x)\,\mathrm{nSents}(y)}{\sum_{s=1}^{S} 1-\cos(x,y_s) + \sum_{s=1}^{S} 1-\cos(x_s,y)}$$

where $x$, $y$ denote one or more sequential sentences from the source/target document; $\cos(x,y)$ is the cosine similarity between embeddings[2] of $x$, $y$; $\mathrm{nSents}(x)$, $\mathrm{nSents}(y)$ denote the number of sentences in $x$, $y$; and $x_1,...,x_S$, $y_1,...,y_S$ are sampled uniformly from the given document.

Following standard practice, we model insertions and deletions in DP using a skip cost $c_{skip}$. The raw value of $c_{skip}$ is only meaningful when compared to other costs, thus we do not expect it to generalize across different languages, normalizations, or resolutions. We propose specifying instead a parameter $\beta_{skip}$ which defines the skip cost in terms of the distribution of 1–1 alignment costs at alignment time: $c_{skip} = \mathrm{CDF}^{-1}(\beta_{skip})$. CDF is an estimate of the cumulative distribution function of 1–1 alignments obtained by computing costs of randomly selected source/target sentences pairs.

### 3.3 Recursive DP Approximation

Instead of searching all possible sentence alignments via DP, consider first averaging adjacent pairs of sentence embeddings in both the source and target documents, halving the number of embeddings for each document. Aligning these vectors via DP (each of which are averages of 2 sentence embeddings) produces an *approximate* sen-

[2] If multiple sentences are considered on one side, they are concatenated together before embedding.

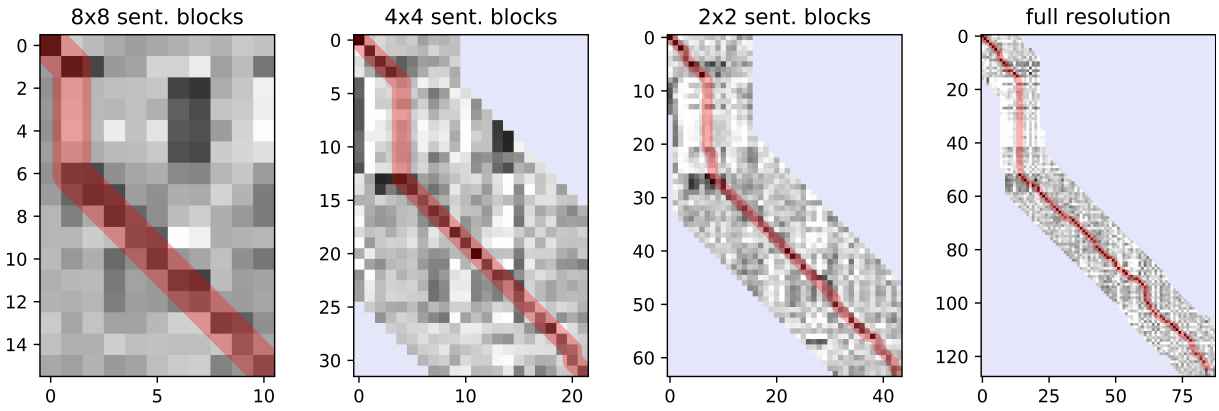


Figure 2: 1–1 alignment costs (darker = lower) for the first 88 De lines (x-axis) and 128 Fr lines (y-axis) at 4 different resolutions. The red highlight denotes alignment found by DP. The algorithm only searches near the path found at previous resolutions; light blue regions are excluded. The vertical part of the path in the top left of each plot is due to 36 extra lines being present in the Fr document. Window size is increased for visualization purposes.

tence alignment, at a cost of $\left(\frac{N}{2}\right)\left(\frac{M}{2}\right)$ comparisons. We can then refine this approximate alignment using the original sentence vectors, constraining ourselves to a small window around the approximate alignment. At a minimum, we must search a window size $w$ large enough to consider all paths covered by the lower-resolution alignment path, but $w$ can also be increased to allow recovery from small errors in the approximate alignment.[3] The length of the refinement path to search is at most $N+M$ (all deletions/insertions), so refining the path requires at most $(N+M)w$ comparisons. Thus the full $NM$ comparisons can be approximated by $(N+M)w+\left(\frac{N}{2}\right)\left(\frac{M}{2}\right)$ comparisons. Applied recursively,[4] we can approximate our quadratic $NM$ cost with a sum of linear costs:

$$(N+M)w+\left(\frac{N}{2}+\frac{M}{2}\right)w+\left(\frac{N}{4}+\frac{M}{4}\right)w+\ldots$$
$$=\sum_{k=0,1,2,\ldots}\frac{(N+M)w}{2^k}=2(N+M)w$$

See Figure 2 for an illustration of this method. We consider only insertions, deletions, and 1–1 alignments in all but the final search. Recursive down sampling and refining of DP was proposed for dynamic time warping in Salvador and Chan (2007), but has not previously been applied to sentence alignment. We direct the reader to that work for a more formal analysis showing the time/space complexity is linear.

[3]We use $w=10$ for all experiments in this work.

[4]In practice, we compute the full DP alignment once the down sampled sizes are below an acceptably small constant. We also find vectors for large blocks of sentences become correlated with each other, so we center them around $\vec{0}$.

## 4 Experiments & Results

### 4.1 Text+Berg Alignment Accuracy

We evaluate sentence alignment accuracy using the development/test split released with Bleualign, consisting of manually aligned yearbook articles published in both German and French by the Swiss Alpine Club from the Text+Berg corpus (Volk et al., 2010). Hyperparameters were chosen to optimize $F_1$ on the development set. We consider alignments of up to 6 total sentences; that is we allow alignments of size $Q$–$R$ where $Q+R\leq 6$.

We compare to Gale and Church (1993), Moore (2002), Hunalign (Varga et al., 2007), Bleualign (Sennrich and Volk, 2010), Gargantua (Braune and Fraser, 2010), and Coverage-Based (Gomes and Lopes, 2016). We run Hunalign in both bootstrapping mode as well as using a publically available De–Fr lexicon from OPUS (Tiedemann, 2012)[5] created from Europarl (Koehn, 2005). Since Bleualign depends on the quality of MT output, we re-run it with a modern NMT system.[6]

Our proposed method outperforms the next best method (Coverage-Based) by 5 $F_1$ points: see Table 1. Gargantua and bootstrapped Hunalign have both been reported to perform well (Abdul-Rauf et al., 2012); this dataset may be too small to bootstrap good lexical features.[7] Bleualign improves by 3 $F_1$ points by using an NMT system.

[5]https://object.pouta.csc.fi/OPUS-Europarl/v7/dic/de-fr.dic.gz

[6]https://docs.microsoft.com/en-us/azure/cognitive-services/translator/

[7]We run only on the test/development articles, not the full Text+Berg corpus.

| Algorithm | O( ) | P | R | $F_1$ |
|---|---|---|---|---|
| Gargantua | $N^2$ | 0.48 | 0.54 | 0.51 |
| Hunalign w/o lexicon | $\boldsymbol{N}$ | 0.59 | 0.70 | 0.64 |
| Hunalign w/ lexicon | $\boldsymbol{N}$ | 0.61 | 0.73 | 0.66 |
| Gale and Church (1993)† | $N^2$ | 0.71 | 0.72 | 0.72 |
| Moore (2002)† | ‡ | 0.86 | 0.71 | 0.78 |
| Bleualign† | $N^2$ | 0.83 | 0.78 | 0.81 |
| Bleualign-NMT | $N^2$ | 0.85 | 0.83 | 0.84 |
| Coverage-Based* | $N^2$ | 0.85 | 0.84 | 0.85 |
| Vecalign | $\boldsymbol{N}$ | **0.89** | **0.90** | **0.90** |

Table 1: De–Fr test precision (P), recall (R), and $F_1$. *best reported in Gomes and Lopes (2016). †Best reported in Sennrich and Volk (2010). ‡O( ) is data dependent. We assume $N = M$ for simplicity.

| Language | ISO 639-1 | Bible # Sents | LASER # Train Lines |
|---|---|---|---|
| Arabic | Ar | 45980 | 8.2M |
| Turkish | Tr | 48492 | 5.7M |
| Somali | So | 37413 | 85k |
| Afrikaans | Af | 37081 | 67k |
| Tagalog | Tl | 34207 | 36k |
| Norwegian | No | 37064 | 0* |

Table 2: Bible statistics. *LASER was not trained on Norwegian but appears to generalize to it.

### 4.2 Bible Alignment Accuracy

We are unaware of a multilingual, low resource, parallel dataset with human sentence-level annotations. As a substitute for gold standard sentence alignment, we use Bible verse alignment and sentence split each verse.[8]

The Bible has a number of properties which make it appealing for sentence alignment evaluation: It is much larger than existing sentence alignment test sets, and it is multi-way parallel in a large number of languages. Bibles are not aligned at the sentence level, but contain verse marking denoting segments typically on the scale of a partial sentence to a few sentences. This creates two potential issues for sentence alignment evaluation: First, a single sentence may span more than one verse. Inspecting the English Bible suggests that this is rare, and sentence aligners should be able to handle occasional over-segmentation of sentences as in practice they are run on errorful automatic sentence segmentation. Second, a verse may contain more than one sentence. This is problematic when it happens on both languages being aligned, since the true sentence alignment cannot be determined (e.g., a verse which is two sentences in each language could be two 1–1 alignments or one 2–2 alignment). To evaluate with verse-level annotations, we propose converting the sentence alignment output into verse alignments by combining any consecutive sentence alignments for which all sentences in the alignments, on both the source and target side, came from the same verse. We report $F_1$ compared to the gold-standard verse alignments, denoting it as *verse-level* $F_1$ to distinguish it from $F_1$ computed at the sentence level.

| Languages | Verse-level $F_1$ Vecalign | Hunalign |
|---|---|---|
| Af–Ar | **0.863** | 0.339 |
| Af–Tl | **0.922** | 0.775 |
| Ar–No | **0.787** | 0.406 |
| Ar–So | **0.634** | 0.067 |
| Tr–So | **0.533** | 0.331 |
| No–So | **0.697** | 0.687 |
| So–Af | **0.782** | 0.738 |
| Tl–No | **0.874** | 0.764 |
| Tr–Af | **0.703** | 0.401 |
| Tr–Tl | **0.647** | 0.247 |

Table 3: Bible verse alignment results.

We select six languages for which Christodouloupoulos and Steedman (2015) contains a full Bible: see Table 2. Languages were chosen to provide a range of amounts of training data used in LASER.[9] From those six languages, we randomly select 10 language pairs for testing. All parameters are kept the same as § 4.1 except we only consider alignments of up to 4 total sentences. We compare to Hunalign, run in bootstrap mode, as it is the only toolkit we tried which was robust enough to run on documents of this size. Results are shown in Table 3.

On average, we see an improvement of 28 verse-level $F_1$ points over Hunalign. In manual analysis of the alignments we find large stretches where the Hunalign alignments are nowhere near the gold alignment in the language pairs with verse-level $F_1 < 0.35$. By contrast, errors in the proposed method are predominately local, indicating success of Vecalign's recursive DP approximation even for very long documents in low-resource languages.

[8]There is no clear choice for sentence segmentation in low-resource languages. We use https://github.com/berkmancenter/mediacloud-sentence-splitter, falling back on English for unsupported languages.

[9]Data amounts are all between the given language and English. LASER used no bitext in the language pairs under test.

### 4.3 Improvements to Downstream MT

One of the primary applications of sentence alignment is creating bitext for training MT systems. To test Vecalign's impact on downstream MT quality, we re-align noisy, web-crawled data in two low-resource language pairs: Sinhala–English and Nepali–English. The data is collected via Paracrawl[10] and is very similar to that released in the WMT 2019 sentence filtering task (Koehn et al., 2019), but some new data has been collected and a small amount of data was lost due to a hard disk failure. Our baseline is the standard Paracrawl pipeline using Hunalign in conjunction with a dictionary extracted from the clean data released in the shared task.

We filter the output of Vecalign and Hunalign following (Chaudhary et al., 2019), including filtering out sentences with the wrong languages and sentences with high token overlap, as this was the best performing method from the shared task.[11] We train and evaluate NMT models following the procedure/hyperparameters from the shared task.

Results are shown in Figure 3. Using Vecalign, we see improvements of 1.7 and 1.6 BLEU for the best data sizes in Sinhala→English and Nepali→English, respectively, compared to the systems trained on Hunalign output.

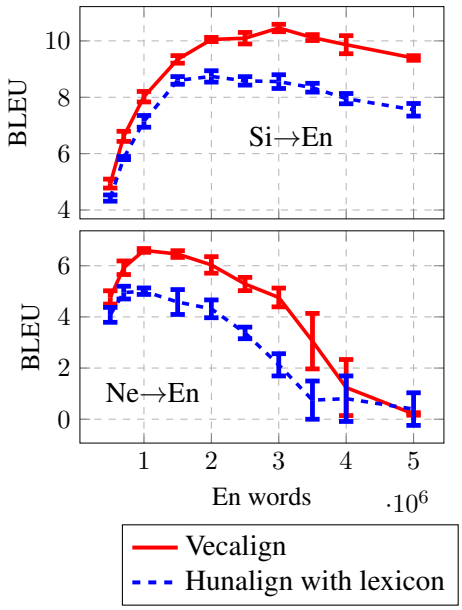


Figure 3: sacreBLEU scores (mean +/- standard deviation for 5 training runs) on FLoRes test sets for systems trained on data aligned with Vecalign vs Hunalign.

[10] https://paracrawl.eu/

[11] We use the publicly available multilingual LASER model, which is not trained on Nepali.

## 5 Empirical Runtime Analysis

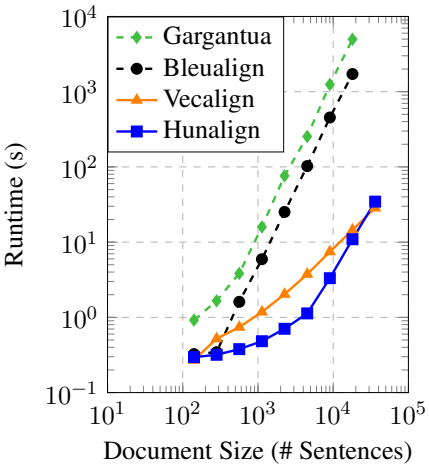


Figure 4: Time required to align various portions of En→Hu Bibles, for various systems. Plot is logarithmic in both runtime and number of sentences, thus a slope of one (i.e. runtime doubles each time the number of sentences doubles) indicates $O(N)$, while a slope of two (i.e. runtime quadruples each time the number of sentences doubles) indicates $O(N^2)$.

Time required to align documents of various sizes are shown for Vecalign, Bleualign, Gartantua, and Hunalign in see Figure 4. As expected, Vecalign has approximately linear runtime characteristics. We use truncated portions of Hu–En Bibles in order to use the dictionary provided with Hunalign. Bleualign is run on NMT output. Vecalign settings match § 4.2. Experiments are run on a Thinkpad T480 with 32GB RAM. Times do not include translation (Bleualign), lexicon building (Hunalign), or sentence embedding (Vecalign). For reference, producing embeddings for 32k sentences, including overlaps, in each language took ~120 s on a GeForce RTX 2080 Ti GPU. Bleualign and Gargantua run out of memory on 32k sentences. Hunalign and Vecalign use ~1GB and are both very fast, aligning 32k sentences in ~30 s.

## 6 Conclusions

We present Vecalign, a novel sentence alignment method based on similarity of sentence embeddings and a DP approximation which is fast even for long documents. Our method has state-of-the-art accuracy in high and low resource settings and improves downstream MT quality.

## Acknowledgments

Brian Thompson is supported through the National Defense Science and Engineering Graduate (NDSEG) Fellowship Program.

## References

Sadaf Abdul-Rauf, Mark Fishel, Patrik Lambert, Sandra Noubours, and Rico Sennrich. 2012. Extrinsic evaluation of sentence alignment systems. *Workshop on Creating Cross-language Resources for Disconnected Languages and Styles*, pages 6–10.

Mikel Artetxe and Holger Schwenk. 2018. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *arXiv preprint arXiv:1812.10464*.

Mikel Artetxe and Holger Schwenk. 2019. Margin-based parallel corpus mining with multilingual sentence embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3197–3203, Florence, Italy. Association for Computational Linguistics.

Richard Bellman. 1953. An introduction to the theory of dynamic programming. Technical report, RAND Corporation, Santa Monica, CA.

Fabienne Braune and Alexander Fraser. 2010. Improved unsupervised sentence alignment for symmetrical and asymmetrical parallel corpora. In *Coling 2010: Posters*, pages 81–89, Beijing, China. Coling 2010 Organizing Committee.

Peter F. Brown, Jennifer C. Lai, and Robert L. Mercer. 1991. Aligning sentences in parallel corpora. In *Proceedings of the 29th Annual Meeting on Association for Computational Linguistics*, ACL '91, pages 169–176, Stroudsburg, PA, USA. Association for Computational Linguistics.

Christian Buck and Philipp Koehn. 2016. Findings of the WMT 2016 bilingual document alignment shared task. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 554–563, Berlin, Germany. Association for Computational Linguistics.

Vishrav Chaudhary, Yuqing Tang, Francisco Guzmn, Holger Schwenk, and Philipp Koehn. 2019. Low-resource corpus filtering using multilingual sentence embeddings. In *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*, pages 263–268, Florence, Italy. Association for Computational Linguistics.

Christos Christodouloupoulos and Mark Steedman. 2015. A massively parallel corpus: the Bible in 100 languages. *Language resources and evaluation*, 49(2):375–395.

William A Gale and Kenneth W Church. 1993. A program for aligning sentences in bilingual corpora. *Computational linguistics*, 19(1):75–102.

Luís Gomes and Gabriel Pereira Lopes. 2016. First steps towards coverage-based sentence alignment. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 2228–2231, Portorož, Slovenia. European Language Resources Association (ELRA).

Cyril Goutte, Marine Carpuat, and George Foster. 2012. The impact of sentence alignment errors on phrase-based machine translation performance. In *The Tenth Biennial Conference of the Association for Machine Translation in the Americas (AMTA 2012)*.

Mandy Guo, Qinlan Shen, Yinfei Yang, Heming Ge, Daniel Cer, Gustavo Hernandez Abrego, Keith Stevens, Noah Constant, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Effective parallel corpus mining using bilingual sentence embeddings. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 165–176, Belgium, Brussels. Association for Computational Linguistics.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, et al. 2018. Achieving human parity on automatic Chinese to English news translation. *arXiv preprint arXiv:1803.05567*.

Huda Khayrallah and Philipp Koehn. 2018. On the impact of various types of noise on neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 74–83, Melbourne, Australia. Association for Computational Linguistics.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Machine Translation summit*, volume 5, pages 79–86.

Philipp Koehn, Francisco Guzmn, Vishrav Chaudhary, and Juan Pino. 2019. Findings of the WMT 2019 shared task on parallel corpus filtering for low-resource conditions. In *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*, pages 56–74, Florence, Italy. Association for Computational Linguistics.

Angeliki Lazaridou, Georgiana Dinu, and Marco Baroni. 2015. Hubness and pollution: Delving into cross-space mapping for zero-shot learning. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 270–280, Beijing, China. Association for Computational Linguistics.

Robert C Moore. 2002. Fast and accurate sentence alignment of bilingual corpora. In *Conference of the Association for Machine Translation in the Americas*, pages 135–144. Springer.

Miloš Radovanović, Alexandros Nanopoulos, and Mirjana Ivanović. 2010. Hubs in space: Popular nearest neighbors in high-dimensional data. *Journal of Machine Learning Research*, 11(Sep):2487–2531.

Stan Salvador and Philip Chan. 2007. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580.

Rico Sennrich and Martin Volk. 2010. MT-based sentence alignment for OCR-generated parallel texts. In *The Ninth Conference of the Association for Machine Translation in the Americas (AMTA 2010)*.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in OPUS. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*, pages 2214–2218, Istanbul, Turkey. European Languages Resources Association (ELRA).

Dániel Varga, Péter Halácsy, András Kornai, Viktor Nagy, László Németh, and Viktor Trón. 2007. Parallel corpora for medium density languages. *Amsterdam Studies In The Theory And History Of Linguistic Science Series 4*, 292:247.

Martin Volk, Noah Bubenhofer, Adrian Althaus, Maya Bangerter, Lenz Furrer, and Beni Ruef. 2010. Challenges in building a multilingual alpine heritage corpus. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Languages Resources Association (ELRA).

Pierre Zweigenbaum, Serge Sharoff, and Reinhard Rapp. 2018. Overview of the third BUCC shared task: Spotting parallel sentences in comparable corpora. In *Proceedings of 11th Workshop on Building and Using Comparable Corpora*, pages 39–42.

# Simpler and Faster Learning of Adaptive Policies for Simultaneous Translation

**Baigong Zheng** [1,*] **Renjie Zheng** [2,*] **Mingbo Ma** [1] **Liang Huang** [1,2]
[1]Baidu Research, Sunnyvale, CA, USA
[2]Oregon State University, Corvallis, OR, USA
{baigongzheng, mingboma}@baidu.com zrenj11@gmail.com

## Abstract

Simultaneous translation is widely useful but remains challenging. Previous work falls into two main categories: (a) fixed-latency policies such as Ma et al. (2019) and (b) adaptive policies such as Gu et al. (2017). The former are simple and effective, but have to aggressively predict future content due to diverging source-target word order; the latter do not anticipate, but suffer from unstable and inefficient training. To combine the merits of both approaches, we propose a simple supervised-learning framework to learn an adaptive policy from oracle READ/WRITE sequences generated from parallel text. At each step, such an oracle sequence chooses to WRITE the next target word if the available source sentence context provides enough information to do so, otherwise READ the next source word. Experiments on German↔English show that our method, without retraining the underlying NMT model, can learn flexible policies with better BLEU scores and similar latencies compared to previous work.

## 1 Introduction

Simultaneous translation outputs target words while the source sentence is being received, and is widely useful in international conferences, negotiations and press releases. However, although there is significant progress in machine translation (MT) recently, simultaneous machine translation is still one of the most challenging tasks. This is because it is hard to balance translation quality and latency, especially for the syntactically divergent language pairs, such as English and Japanese.

Researchers previously study simultaneous translation as a part of real-time speech-to-speech translation system (Yarmohammadi et al., 2013; Bangalore et al., 2012; Fügen et al., 2007; Sridhar et al., 2013; Jaitly et al., 2016; Graves et al., 2013). Recent simultaneous translation research focuses on obtaining a strategy, called a *policy*, to decide whether to wait for another source word (READ action) or emit a target word (WRITE action). The obtained policies fall into two main categories: (1) fixed-latency policies (Ma et al., 2019; Dalvi et al., 2018) and (2) context-dependent adaptive policies (Grissom II et al., 2014; Cho and Esipova, 2016; Gu et al., 2017; Alinejad et al., 2018; Arivazhagan et al., 2019; Zheng et al., 2019a). As an example of fixed-latency policies, wait-$k$ (Ma et al., 2019) starts by waiting for the first few source words and then outputs one target word after receiving each new source word until the source sentence ends. It is easy to see that this kind of policy will inevitably need to guess the future content, which can often be incorrect. Thus, an adaptive policy (see Table 1 as an example), which can decides on the fly whether to take READ action or WRITE action, is more desirable for simultaneous translation. Moreover, the widely-used beam search technique become non-trivial for fixed policies (Zheng et al., 2019b).

To represent an adaptive policy, previous work shows three different ways: (1) a rule-based decoding algorithm (Cho and Esipova, 2016), (2) an original MT model with an extended vocabulary (Zheng et al., 2019a), and (3) a separate policy model (Grissom II et al., 2014; Gu et al., 2017; Alinejad et al., 2018; Arivazhagan et al., 2019). The decoding algorithm (Cho and Esipova, 2016) applies heuristics measures and does not exploit information in the hidden representation, while the MT model with an extended vocabulary (Zheng et al., 2019a) needs guidance from a restricted dynamic oracle to learn an adaptive policy, whose size is exponentially large so that approximation is needed. A separate policy model could avoid these issues. However, previous policy-learning

[*] These authors contributed equally.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1349–1354, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| German | Ich | | bin | mit | dem | Bus | | | | nach | Ulm | gekommen | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Gloss | I | | am | with | the | bus | | | | to | Ulm | come | | | | | | |
| Action | R | W | R | R | R | R | W | W | W | R | R | R | W | W | W | W | | |
| Translation | | I | | | | | took | the | bus | | | | to | come | to | Ulm | | |

Table 1: An example for READ/WRITE action sequence. R represents READ and W represents WRITE.

methods either depends on reinforcement learning (RL) (Grissom II et al., 2014; Gu et al., 2017; Alinejad et al., 2018), which makes the training process unstable and inefficient due to exploration, or applies advanced attention mechanisms (Arivazhagan et al., 2019), which requires its training process to be autoregressive, and hence inefficient. Furthermore, each such learned policy cannot change its behaviour according to different latency requirements at testing time, and we will need to train multiple policy models for scenarios with different latency requirements.

To combine the merits of fixed and adaptive policies, and to resolve the mentioned drawbacks, we propose a simple supervised learning framework to learn an adaptive policy, and show how to apply it with controllable latency. This framework is based on sequences of READ/WRITE actions for parallel sentence pairs, so we present a simple method to generate such an action sequence for each sentence pair with a pre-trained neural machine translation (NMT) model.[1] Our experiments on German↔English dataset show that our method, without retraining the underlying NMT model, leads to better policies than previous methods, and achieves better BLEU scores than (the retrained) wait-$k$ models at low latency scenarios.

## 2 Generating Action Sequences

In this section, we show how to generate action sequences for parallel text. Our simultaneous translation policy can take two actions: READ (receive a new source word) and WRITE (output a new target word). A sequence of such actions for a sentence pair $(\mathbf{s}, \mathbf{t})$ defines one way to translate $\mathbf{s}$ into $\mathbf{t}$. Thus, such a sequence must have $|\mathbf{t}|$ number of WRITE actions. However, not every action sequence is good for simultaneous translation. For instance, a sequence without any READ action will not provide any source information, while a sequence with all $|\mathbf{s}|$ number of READ actions before all WRITE actions usually has large latency. Thus the ideal sequences for simultaneous translation should have the following two properties:

- there is no anticipation during translation, i.e. when choosing WRITE action, there is enough source information for the MT model to generate the correct target word;
- the latency is as low as possible, i.e. the WRITE action for each target word appears as early as possible.

Table 1 gives an example for such a sequence.

**Algorithm 1** Generating Action Sequence

**Input**: sentence pair $(\mathbf{s}, \mathbf{t})$, integer $r$, model $M$
$id_s \leftarrow 1, id_t \leftarrow 1$
Seq $\leftarrow [R]$
**while** $id_t \leq |\mathbf{t}|$ **do**
  **if** $rank_M(t_{id_t}|\mathbf{s}_{\leq id_s}) \leq r$ or $id_s = |\mathbf{s}|$ **then**
    Seq $\leftarrow$ Seq $+[W]$
    $id_t \leftarrow id_t + 1$
  **else**
    Seq $\leftarrow$ Seq $+[R]$
    $id_s \leftarrow id_s + 1$
**return** Seq

In the following, we present a simple method to generate such an action sequence for a sentence pair $(\mathbf{s}, \mathbf{t})$ using a pre-trained NMT model, assuming this model can make reasonable prediction given incomplete source sentence. Our method is based on this observation: *if the rank of the next ground-truth target word is high enough in the prediction of the model, then this implies that there is enough source-side information for the model to make a correct prediction.* Specifically, we sequentially input the source words to the pre-trained model, and use it to predict next target word. If the rank of the gold target word is high enough, we will append a WRITE action to the sequence and then try the next target word; otherwise, we append a READ action and input a new source word. Let $r$ be a positive integer, $M$ be a pre-trained NMT model, $\mathbf{s}_{\leq i}$ be the source sequence consisting of the first $i$ words of $\mathbf{s}$, and $rank_M(t_j|\mathbf{s}_{\leq i})$ be the rank of the target word $t_j$ in the prediction of model $M$

[1] Previous work (He et al., 2015; Niehues et al., 2018) shows that carefully generated parallel training data can help improve simultaneous MT or low-latency speech translation. This is different from our work in that we generates action sequences for training policy model instead of parallel translation data for MT model.

given sequence $\mathbf{s}_{\leq i}$. Then the generating process can be summarized as Algorithm 1.

Although we can generate action sequences balancing the two wanted properties with appropriate value of parameter $r$, the latency of generated action sequence may still be large due to the word order difference between the two sentences. To avoid this issue, we filter the generated sequences with the latency metric *Average Lagging* (AL) proposed by Ma et al. (2019), which quantifies the latency in terms of the number of source words and avoids some flaws of other metrics like *Average Proportion* (AP) (Cho and Esipova, 2016) and *Consecutive Wait* (CW) (Gu et al., 2017). Another issue we observed is that, the pre-trained model may be too aggressive for some sentence pair, meaning that it may write all target words without seeing the whole source sentence. This may be because the model is also trained on the same dataset. To overcome this, we only keep the action sequences that will receive all the source words before the last WRITE action. After the filtering process, each action sequence has AL less than a fixed constant $\alpha$ and receives all source words.

## 3 Supervised-Learning Framework for Simultaneous Translation Policy

Given a sentence pair and an action sequence for this pair, we can apply supervised learning method to learn a parameterized policy for simultaneous translation. For the policy to be able to choose the correct action, its input should include information from both source and target sides. Since we use Transformer (Vaswani et al., 2017) as our underlying NMT model in this work, we need to recompute encoder hidden states for all previous seen source words, which is the same as done for wait-$k$ model training (Ma et al., 2019).[2] The policy input $o_i$ at step $i$ consists of three components from this model:

- $h_i^s$: the last-layer hidden state from the encoder for the first source word at step $i$;
- $h_i^t$: the last-layer hidden state from the decoder for the first target word at step $i$;[3]
- $c_i$: cross-attention scores at step $i$ for the current input target word on all attention layers in decoder, averaged over all current source words.

That is $o_i = [h_i^s, h_i^t, c_i]$.

Let $a_i$ be the $i$-th action in the given action sequence $\mathbf{a}$. Then the decision of our policy on the $i$-th step depends on all previous inputs $\mathbf{o}_{\leq i}$ and all taken actions $\mathbf{a}_{<i}$. We want to maximize the probability of the next action $a_i$ given those information:

$$\max p_\theta(a_i|\mathbf{o}_{\leq i}, \mathbf{a}_{<i})$$

where $p_\theta$ is the action distribution of our policy parameterized by $\theta$.

## 4 Decoding with Controllable Latency

To apply the learned policy for simultaneous translation, we can choose at each step the action with higher probability . However, different scenarios may have different latency requirements. Thus, this greedy policy may not always be the best choice for all situations. Here we present a simple way to implicitly control the latency of the learned policy without retraining the policy model.

Let $\rho$ be a probability threshold. For each step in translation, we choose READ action only if the probability of READ is greater than $\rho$; otherwise we choose WRITE action. Thus, this threshold balances the tradeoff between latency and translation quality: with larger $\rho$, the policy prefers to take WRITE actions, providing lower latency; and with smaller $\rho$, the policy prefers to take READ actions, and provides more conservative translation with larger latency.

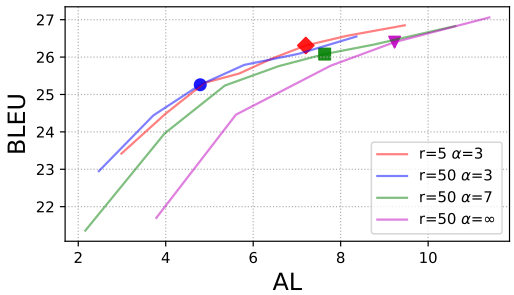


Figure 1: Translation quality against latency on DE→EN dev set. Lines are obtained with different probability thresholds $\rho$. Markers represent $\rho = 0.5$.

## 5 Experiments

**Dataset** We conduct experiments on English↔German (EN↔DE) simultaneous translation. We use the parallel corpora from

[2] In our experiments, the decoder and policy model combined need about 0.0445 seconds on average to generate one target word (this might include multiple READ's), while it takes on average only about 0.0058 seconds to recompute all encoder states for each new source word. So we think this re-computation might not be a serious issue for system efficiency.

[3] The hidden states of the first target word will be recomputed at each step.

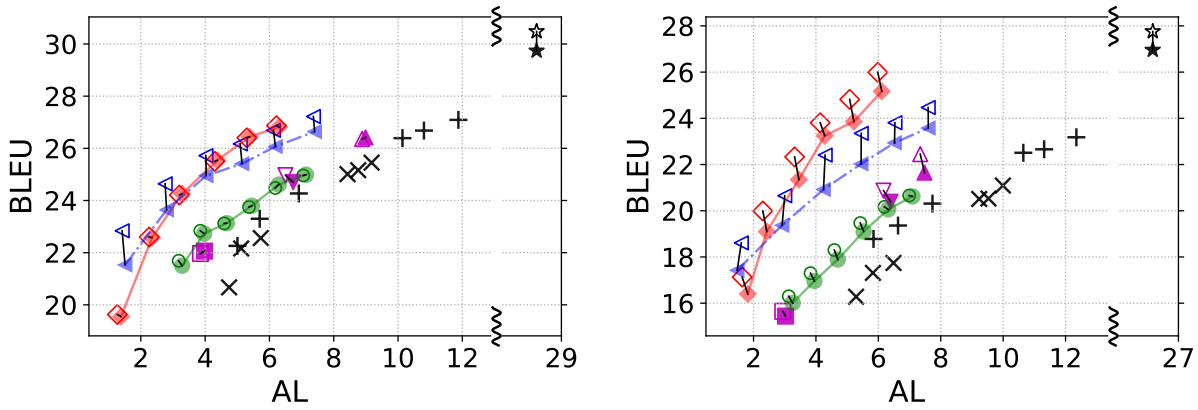


Figure 2: Comparing performances of different methods on testing sets. Left: DE→EN. Right: EN→DE. The shown pairs are results of greedy decoding (solid shapes) and beam search (empty shapes, beam-size = 5). ◆: wait-$k$ models for $k \in \{1,2,3,4,5,6\}$, ●: test-time wait-$k$ for $k \in \{1,2,3,4,5,6\}$, ◀: our SL model with threshold $\rho \in \{0.65, 0.6, 0.55, 0.5, 0.45, 0.4\}$, ★: full-sentence translation model, ■: RL with CW = 2, ▼: RL with CW = 5, ▲: RL with CW = 8, ×: WID for $s_0 \in \{2,4,6\}$ and $\delta \in \{2,4\}$, +: WIW for $s_0 \in \{2,4,6\}$ and $\delta \in \{1,2\}$.

WMT 15 for training, newstest-2013 for validation and newstest-2015 for testing.[4] All datasets are tokenized and segmented into sub-word units with byte-pair encoding (BPE) (Sennrich et al., 2016), and we only use the sentence pairs of lengths less than 50 (on both sides) for training.

**Model Configuration** We use Transformer-base (Vaswani et al., 2017) as our NMT model and our implementation is based on PyTorch-based OpenNMT (Klein et al., 2017). We add an <eos> token on the source side, which is not included in the original OpenNMT codebase. Our recurrent policy model consists of one GRU layer with 512 units, one fully-connected layer of dimension 64 followed by ReLU activation, and one fully-connected layer of dimension 2 followed by a softmax function to produce the action distribution. We use BLEU (Papineni et al., 2002) as the translation quality metric and Averaged Lagging (AL) (Ma et al., 2019) as the latency metric.

**Effects of Generated Action Sequences** We first analyze the effects of the two parameters in the generation process of action sequences: the rank $r$ and the filtering latency $\alpha$. We fix $\alpha = 3$ and choose the rank $r \in \{5, 50\}$; then we fix $r = 50$ and choose the latency $\alpha \in \{3, 7, \infty\}$ to generate action sequences on DE→EN direction.Figure 1 shows the performances of resulting models with different probability thresholds $\rho$. We find that smaller $\alpha$ helps achieve better performance and our model is not very sensitive to values of rank. Therefore, in the following experiments, we report results with $r = 50$ and $\alpha = 3$.

**Performance Comparison** We compare our method on EN↔DE directions with different methods: greedy decoding algorithms Wait-If-Worse/Wait-If-Diff (WIW/WID) of Cho and Esipova (2016), RL method of Gu et al. (2017), wait-$k$ models and test-time wait-$k$ methods of Ma et al. (2019). Both of WIW and WID only use the pre-trained NMT model, and the algorithm initially reads $s_0$ number of source words and chooses READ only if the probability of the most likely target word decreases when given another $\delta$ number of source words (WIW) or the most likely target word changes when given another $\delta$ number of source words (WID). For RL method, we use the same kind of input and architecture for policy model.[5] Test-time wait-$k$ means decoding with wait-$k$ policy using the pre-trained NMT model. All methods share the same underlying pre-trained NMT model except for the wait-$k$ method, which retrains the NMT model from scratch, but with the same architecture as the pre-trained model.

Figure 2 shows the performance comparison. Our models on both directions can achieve higher BLEU scores with the similar latency than WIW, WID, RL model and test-time wait-$k$ method, implying that our method learns a better policy model than the other methods when the underlying NMT model is not retrained.[6] Compared with wait-$k$ models, our models with beam search achieve

[4] http://www.statmt.org/wmt15/translation-task.html

[5] We only report results obtained with CW-based reward functions for they outperform those with AP-based ones.

[6] Our test-time wait-$k$ results are better than those in Ma et al. (2019), because the added source side <eos> token helps the full-sentence model to learn when the source sentence

| | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| German | die | deutsche | bahn | will | | im | kommenden | jahr | | die | kin- | zi- | g- | tal- | – | bahn- | strecke | verbessern | . | | | | | | |
| gloss | the | German | train | want | | in the | coming | year | | the | Kinzigtal | | | | – | railroad track | | improve | . | | | | | | |
| wait-3 | | | deutsche | bahn | wants | | to | make | the | cinema | show | a | success | this | coming | year | . | | | | | | | | |
| wait-5 | | | | | deutsche | bahn | wants | to | introduce | the | kin- | zi- | g- | tal | railway | line | next year . | | | | | | | | |
| test-time wait-3 | | | deutsche | bahn | wants | the | german | railways | to | be | the | kin- | z- | ig- | tal | railway | line in the coming year . | | | | | | | | |
| test-time wait-5 | | | | | the | german | railways | wants | to | take | the | german | train | to | the | german | railways in the coming year . | | | | | | | | |
| SL policy | the | german | railway | wants the | | | | | | | | | | | | | kinzigtal railway to be improved next year . | | | | | | | | |
| RL policy | | | | the german | | | railways wants | | | the german | | | railway will | | | | improve the | kinzigtal railway next year . | | | | | | | |

Table 2: German-to-English example from validation set.

higher BLEU scores when latency AL is small, which we think will be the most useful scenarios of simultaneous translation. Furthermore, this figure also shows that our model can achieve good performance on different latency conditions by controlling the threshold $\rho$, so we do not need to train multiple models for different latency requirements. We also provide a translation example in Table 2 to compare different methods.

**Learning Process Analysis** We analyze two aspects of the learning processes of different methods: stability and training time. Figure 3 shows the learning curves of the training processes of RL method and our SL method, averaged over four different runs with different random seeds on DE→EN direction. We can see that the training process of our method is more stable and converges faster than the RL method. Although there are some steps where the RL training process can achieve better BLEU scores than our SL method, the corresponding latencies are usually very big, which are not appropriate for simultaneous translation. We present the training time of different methods in Table 3. Our method only need about 12 hours to train the policy model with 1 GPU, while the wait-$k$ method needs more than 600 hours with 8 GPUs to finish the training process, showing that our method is very efficient. Note that this table does not include the time needed to generate action sequences. The time for this process could be very flexible since we can parallelize this by dividing the training data into separating parts. In our experiments, we need about 2 hours to generate all action sequences in parallel.

## 6 Conclusions

We have proposed a simple supervised-learning framework to learn an adaptive policy based on generated action sequences for simultaneous translation, which leads to faster training and better policies than previous methods, without the need to retrain the underlying NMT model.

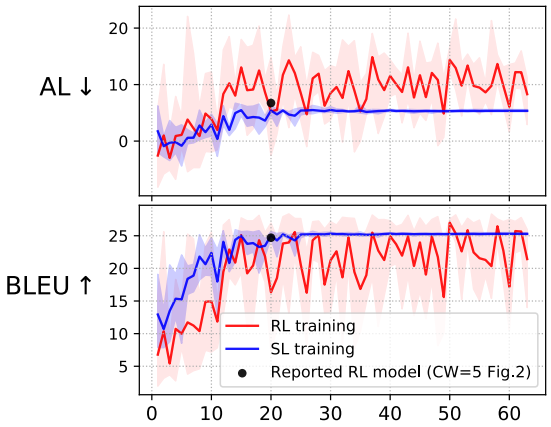


Figure 3: Learning curves averaged over four independent training runs on DE→EN direction. The $x$-axis represents training steps ($\times$ 50).

| | | | | | |
|---|---|---|---|---|---|
| DE→EN | pre-train: 64 | | wait-1 | wait-3 | wait-5 |
| | SL: +12 | RL: +14.3 | 938 | 966 | 945 |
| EN→DE | pre-train: 68 | | wait-1 | wait-3 | wait-5 |
| | SL: +11 | RL: +11.2 | 665 | 665 | 630 |

Table 3: Training time (in hours) of different methods. Wait-$k$ training uses 8 GPUs, while others use 1 GPU. The RL time is the average time needed to obtain the three RL models in Figure 2.

## Acknowledgments

We thank Hairong Liu for helpful discussion, Kaibo Liu for helping training baseline models, and the anonymous reviewers for suggestions. We also thank Kaibo Liu for making the AL script available at https://github.com/SimulTrans-demo/STACL.

## References

Ashkan Alinejad, Maryam Siahbani, and Anoop Sarkar. 2018. Prediction improves simultaneous neural ma-

ends. Without this token, test-time wait-$k$ generates either very short target sentences or many punctuations for small $k$'s.

chine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3022–3027.

Naveen Arivazhagan, Colin Cherry, Wolfgang Macherey, Chung-Cheng Chiu, Semih Yavuz, Ruoming Pang, Wei Li, and Colin Raffel. 2019. Monotonic infinite lookback attention for simultaneous machine translation. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019*, pages 1313–1323.

Srinivas Bangalore, Vivek Kumar Rangarajan Sridhar, Prakash Kolan, Ladan Golipour, and Aura Jimenez. 2012. Real-time incremental speech-to-speech translation of dialogs. In *Proc. of NAACL-HLT*.

Kyunghyun Cho and Masha Esipova. 2016. Can neural machine translation do simultaneous translation? *arXiv preprint arXiv:1606.02012*.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, and Stephan Vogel. 2018. Incremental decoding and training methods for simultaneous translation in neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 493–499.

Christian Fügen, Alex Waibel, and Muntsin Kolss. 2007. Simultaneous translation of lectures and speeches. *Machine translation*, 21(4):209–252.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE.

Alvin Grissom II, He He, Jordan Boyd-Graber, John Morgan, and Hal Daumé III. 2014. Don't until the final verb wait: Reinforcement learning for simultaneous machine translation. In *Proceedings of the 2014 Conference on empirical methods in natural language processing (EMNLP)*, pages 1342–1352.

Jiatao Gu, Graham Neubig, Kyunghyun Cho, and Victor O. K. Li. 2017. Learning to translate in real-time with neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017*, pages 1053–1062.

He He, Alvin Grissom II, John Morgan, Jordan Boyd-Graber, and Hal Daumé III. 2015. Syntax-based rewriting for simultaneous machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 55–64.

Navdeep Jaitly, David Sussillo, Quoc V Le, Oriol Vinyals, Ilya Sutskever, and Samy Bengio. 2016. An online sequence-to-sequence model using partial conditioning. In *Advances in Neural Information Processing Systems*, pages 5067–5075.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. *arXiv preprint arXiv:1701.02810*.

Mingbo Ma, Liang Huang, Hao Xiong, Renjie Zheng, Kaibo Liu, Baigong Zheng, Chuanqiang Zhang, Zhongjun He, Hairong Liu, Xing Li, Hua Wu, and Haifeng Wang. 2019. STACL: Simultaneous translation with implicit anticipation and controllable latency using prefix-to-prefix framework. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3025–3036.

Jan Niehues, Ngoc-Quan Pham, Thanh-Le Ha, Matthias Sperber, and Alex Waibel. 2018. Low-latency neural speech translation. In *Interspeech 2018, 19th Annual Conference of the International Speech Communication Association.*, pages 1293–1297.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*, pages 311–318, Philadephia, USA.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725.

Vivek Kumar Rangarajan Sridhar, John Chen, Srinivas Bangalore, Andrej Ljolje, and Rathinavelu Chengalvarayan. 2013. Segmentation strategies for streaming speech translation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 230–238.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Mahsa Yarmohammadi, Vivek Kumar Rangarajan Sridhar, Srinivas Bangalore, and Baskaran Sankaran. 2013. Incremental segmentation and decoding strategies for simultaneous translation. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*.

Baigong Zheng, Renjie Zheng, Mingbo Ma, and Liang Huang. 2019a. Simultaneous translation with flexible policy via restricted imitation learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5816–5822.

Renjie Zheng, Mingbo Ma, Baigong Zheng, and Liang Huang. 2019b. Speculative beam search for simultaneous translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing*.

# Adversarial Learning with Contextual Embeddings for Zero-resource Cross-lingual Classification and NER

**Phillip Keung, Yichao Lu, Vikas Bhardwaj**
Amazon Inc.
{keung, yichaolu, vikab}@amazon.com

## Abstract

Contextual word embeddings (e.g. GPT, BERT, ELMo, etc.) have demonstrated state-of-the-art performance on various NLP tasks. Recent work with the multilingual version of BERT has shown that the model performs very well in cross-lingual settings, even when only labeled English data is used to finetune the model. We improve upon multilingual BERT's zero-resource cross-lingual performance via adversarial learning. We report the magnitude of the improvement on the multilingual MLDoc text classification and CoNLL 2002/2003 named entity recognition tasks. Furthermore, we show that language-adversarial training encourages BERT to align the embeddings of English documents and their translations, which may be the cause of the observed performance gains.

## 1 Introduction

Contextual word embeddings (Devlin et al., 2019; Peters et al., 2018; Radford et al., 2019) have been successfully applied to various NLP tasks, including named entity recognition, document classification, and textual entailment. The multilingual version of BERT (which is trained on Wikipedia articles from 100 languages and equipped with a 110,000 shared wordpiece vocabulary) has also demonstrated the ability to perform 'zero-resource' cross-lingual classification on the XNLI dataset (Conneau et al., 2018). Specifically, when multilingual BERT is finetuned for XNLI with English data alone, the model also gains the ability to handle the same task in other languages. We believe that this zero-resource transfer learning can be extended to other multilingual datasets.

In this work, we explore BERT's[1] zero-resource performance on the multilingual MLDoc classification and CoNLL 2002/2003 NER tasks. We find that the baseline zero-resource performance of BERT exceeds the results reported in other work, even though cross-lingual resources (e.g. parallel text, dictionaries, etc.) are not used during BERT pretraining or finetuning. We apply adversarial learning to further improve upon this baseline, achieving state-of-the-art zero-resource results.

There are many recent approaches to zero-resource cross-lingual classification and NER, including adversarial learning (Chen et al., 2019; Kim et al., 2017; Xie et al., 2018; Joty et al., 2017), using a model pretrained on parallel text (Artetxe and Schwenk, 2018; Lu et al., 2018; Lample and Conneau, 2019) and self-training (Hajmohammadi et al., 2015). Due to the newness of the subject matter, the definition of 'zero-resource' varies somewhat from author to author. For the experiments that follow, 'zero-resource' means that, during model training, we do not use labels from non-English data, nor do we use human or machine-generated parallel text. Only labeled English text and unlabeled non-English text are used during training, and hyperparameters are selected using English evaluation sets.

Our contributions are the following:

- We demonstrate that the addition of a language-adversarial task during finetuning for multilingual BERT can significantly improve the zero-resource cross-lingual transfer performance.
- For both MLDoc classification and CoNLL NER, we find that, even without adversarial training, the baseline multilingual BERT performance can exceed previously published results on zero-resource performance.
- We show that adversarial techniques encourage BERT to align the representations of En-

[1] 'BERT' hereafter refers to multilingual BERT



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1355–1360, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

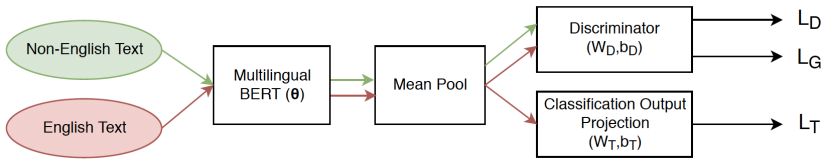


(a) Text classification

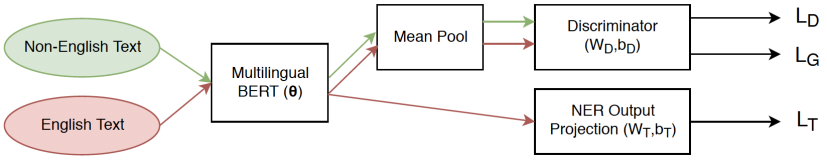


(b) NER

Figure 1: Overview of the adversarial training process for classification and NER. All input text is in the form of a sequence of word pieces. $L_D, L_G, L_T$ refer to the discriminator, generator and task-specific losses. Parameters of each component is in round brackets.

glish documents and their translations. We speculate that this alignment causes the observed improvement in zero-resource performance.

## 2 Related Work

### 2.1 Adversarial Learning

Language-adversarial training (Zhang et al., 2017) was proposed for generating bilingual dictionaries without parallel data. This idea was extended to zero-resource cross-lingual tasks in NER (Kim et al., 2017; Xie et al., 2018) and text classification (Chen et al., 2019), where we would expect that language-adversarial techniques induce features that are language-independent.

### 2.2 Self-training Techniques

Self-training, where an initial model is used to generate labels on an unlabeled corpus for the purpose of domain or cross-lingual adaptation, was studied in the context of text classification (Hajmohammadi et al., 2015) and parsing (McClosky et al., 2006; Zeman and Resnik, 2008). A similar idea based on expectation-maximization, where the unobserved label is treated as a latent variable, has also been applied to cross-lingual text classification in Rigutini et al. (2005).

### 2.3 Translation as Pretraining

Artetxe and Schwenk (2018) and Lu et al. (2018) use the encoders from machine translation models as a starting point for task-specific finetuning, which permits various degrees of multilingual transfer. Lample and Conneau (2019) add an additional masked translation task to the BERT pretraining process, and the authors observed an improvement in the cross-lingual setting over using the monolingual masked text task alone.

## 3 Experiments

### 3.1 Model Training

We present an overview of the adversarial training process in Figure 1. We used the pretrained cased multilingual BERT model[2] as the initialization for all of our experiments. Note that the BERT model has 768 units.

We always use the labeled English data of each corpus. We use the non-English text portion (without the labels) for the adversarial training.

We formulate the adversarial task as a binary classification problem (i.e. English versus non-English.) We add a *language discriminator* module which uses the BERT embeddings to classify whether the input sentence was written in English or the non-English language. We also add a *generator* loss which encourages BERT to produce embeddings that are difficult for the discriminator to classify correctly. In this way, the BERT model learns to generate embeddings that do not contain language-specific information.

The pseudocode for our procedure can be found in Algorithm 1. In the description that follows, we use a batch size of 1 for clarity.

For language-adversarial training for the classification task, we have 3 loss functions: the task-specific loss $L_T$, the generator loss $L_G$, and the

[2]https://github.com/google-research/bert/blob/master/multilingual.md

discriminator loss $L_D$:

$$L_T(y^T; x) = \sum_{i=1}^{K} -y_i^T \log p(Y = i|x)$$

$$p(Y|x) = \text{Softmax}(W_T \bar{h}_\theta(x) + b_T)$$

$$L_G(y^A; x) = -(1 - y^A)\log p(E = 1|x) - y^A \log p(E = 0|x)$$

$$L_D(y^A; x) = -(1 - y^A)\log p(E = 0|x) - y^A \log p(E = 1|x)$$

$$p(E = 1|x) = \text{Sigmoid}(w_D \cdot \bar{h}_\theta(x) + b_D)$$

where $K$ is the number of classes for the task, $p(Y|x)$ (dim: $K \times 1$) is the task-specific prediction, $p(E = 1|x)$ (dim: scalar) is the probability that the input is in English, $\bar{h}_\theta(x)$ (dim: $768 \times 1$) is the mean-pooled BERT output embedding for the input word-pieces $x$, $\theta$ is the BERT parameters, $W_T, b_T, w_D, b_D$ (dim: $K \times 768$, $K \times 1$, $768 \times 1$, scalar) are the output projections for the task-specific loss and discriminator respectively, $y^T$ (dim: $K \times 1$) is the one-hot vector representation for the task label and $y^A$ (dim: scalar) is the binary label for the adversarial task (i.e. 1 or 0 for English or non-English).

In the case of NER, the task-specific loss has an additional summation over the length of the sequence:

$$L_T(y^T; x) = \sum_{i=1}^{K} \sum_{t=1}^{L} -y_{it}^T \log p(Y_t = i|x)$$

$$p(Y_t|x) = \text{Softmax}(W_T h_\theta(x)_t + b_T)$$

where $p(Y_t|x)$ (dim: $K \times 1$) is the prediction for the $t^{th}$ word, $L$ is the number of words in the sentence, $y^T$ (dim: $K \times L$) is the matrix of one-hot entity labels, and $h_\theta(x)_t$ (dim: $768 \times 1$) refers to the BERT embedding of the $t^{th}$ word.

The generator and discriminator losses remain the same for NER, and we continue to use the mean-pooled BERT embedding during adversarial training.

We then take the gradients with respect to the 3 losses and the relevant parameter subsets. The parameter subsets are $\theta_D = \{w_D, b_D\}$, $\theta_T = \{\theta, W_T, b_T\}$, and $\theta_G = \{\theta\}$. We apply the gradient updates sequentially at a 1:1:1 ratio.

During BERT finetuning, the learning rates for the task loss, generator loss and discriminator loss were kept constant; we do not apply a learning rate decay.

All hyperparameters were tuned on the English dev sets only, and we use the Adam optimizer in all experiments. We report results based on the average of 4 training runs.

### 3.2 MLDoc Classification Results

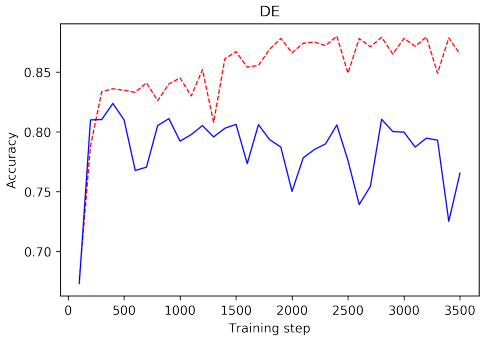


(a) German test accuracy vs steps taken

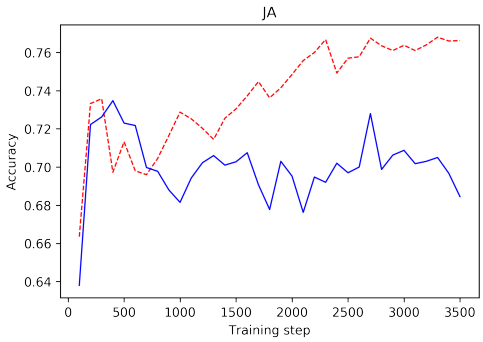


(b) Japanese test accuracy vs steps taken

Figure 2: German and Japanese MLDoc test accuracy versus the number of training steps, with and without adversarial training. The solid line shows the performance of the non-adversarial BERT baseline. The dashed line shows the performance with adversarial training.

We finetuned BERT on the English portion of the MLDoc corpus (Schwenk and Li, 2018). The MLDoc task is a 4-class classification problem, where the data is a class-balanced subset of the Reuters News RCV1 and RCV2 datasets. We used the `english.train.1000` dataset for the classification loss, which contains 1000 labeled documents. For language-adversarial training, we used the text portion of `german.train.10000`, `french.train.10000`, etc. without the labels.

**Algorithm 1** Pseudocode for adversarial training on the multilingual text classification task. The batch size is set at 1 for clarity. The parameter subsets are $\theta_D = \{w_D, b_D\}$, $\theta_T = \{\theta, W_T, b_T\}$, and $\theta_G = \{\theta\}$.

**Input:** pre-trained BERT model $h_\theta$, data iterators for English and the non-English language $L$, learning rates $\eta_D, \eta_G, \eta_T$ for each loss function, initializations for discriminator output projection $w_D, b_D$, task-specific output projection $W_T, b_T$, and BERT parameters $\theta$

**while** not converged **do**
  $x_{En}, y^T \leftarrow \text{DataIterator}(En)$ ▷ get English text and task-specific labels
  $\bar{h}_{En} \leftarrow \text{MeanPool}(h_\theta(x_{En}))$
  $p^T \leftarrow \text{Softmax}(W_T \bar{h}_{En} + b_T)$ ▷ compute the prediction for the task
  $L_T \leftarrow -y^T \cdot \log p^T$ ▷ compute task-specific loss
  $\theta, W_T, b_T \mathrel{+}= -\eta_T \nabla_{\theta_T} L_T$ ▷ update model based on task-specific loss
  $x_L, x_{En} \leftarrow \text{DataIterator}(L), \text{DataIterator}(En)$ ▷ get non-English and English text
  $\bar{h}_L, \bar{h}_{En} \leftarrow \text{MeanPool}(h_\theta(x_L)), \text{MeanPool}(h_\theta(x_{En}))$
  $p^D_L \leftarrow \text{Sigmoid}(w_D \cdot \bar{h}_L + b_D)$ ▷ discriminator prediction on non-English text
  $p^D_{En} \leftarrow \text{Sigmoid}(w_D \cdot \bar{h}_{En} + b_D)$ ▷ discriminator prediction on English text
  $L_D \leftarrow -\log p^D_{En} - \log(1 - p^D_L)$ ▷ compute discriminator loss
  $w_D, b_D \mathrel{+}= -\eta_D \nabla_{\theta_D} L_D$ ▷ update model based on discriminator loss
  $x_L, x_{En} \leftarrow \text{DataIterator}(L), \text{DataIterator}(En)$
  $\bar{h}_L, \bar{h}_{En} \leftarrow \text{MeanPool}(h_\theta(x_L)), \text{MeanPool}(h_\theta(x_{En}))$
  $p^D_L \leftarrow \text{Sigmoid}(w_D \cdot \bar{h}_L + b_D)$
  $p^D_{En} \leftarrow \text{Sigmoid}(w_D \cdot \bar{h}_{En} + b_D)$
  $L_G \leftarrow -\log(1 - p^D_{En}) - \log p^D_L$ ▷ compute generator loss
  $\theta \mathrel{+}= -\eta_G \nabla_{\theta_G} L_G$ ▷ update model based on generator loss

| | En | De | Es | Fr | It | Ja | Ru | Zh |
|---|---|---|---|---|---|---|---|---|
| Schwenk and Li (2018) | 92.2 | 81.2 | 72.5 | 72.4 | 69.4 | 67.6 | 60.8 | 74.7 |
| Artetxe and Schwenk (2018) | 89.9 | 84.8 | 77.3 | 77.9 | 69.4 | 60.3 | 67.8 | 71.9 |
| BERT En-labels | 94.2 | 79.8 | 72.1 | 73.5 | 63.7 | 72.8 | 73.7 | 76.0 |
| BERT En-labels + Adv. | - | **88.1** | **80.8** | **85.7** | **72.3** | **76.8** | **77.4** | **84.7** |

Table 1: Classification accuracy on the MLDoc test sets. We present results for BERT finetuned on labeled English data and BERT finetuned on labeled English data with language-adversarial training. Our results are averaged across 4 training runs, and hyperparameters are tuned on English dev data.

We used a learning rate of $2 \times 10^{-6}$ for the task loss, $2 \times 10^{-8}$ for the generator loss and $5 \times 10^{-5}$ for the discriminator loss.

In Table 1, we report the classification accuracy for all of the languages in MLDoc. Generally, adversarial training improves the accuracy across all languages, and the improvement is sometimes dramatic versus the BERT non-adversarial baseline.

In Figure 2, we plot the zero-resource German and Japanese test set accuracy as a function of the number of steps taken, with and without adversarial training. The plot shows that the variation in the test accuracy is reduced with adversarial training, which suggests that the cross-lingual performance is more consistent when adversarial training is applied. (We note that the batch size and learning rates are the same for all the languages in MLDoc, so the variation seen in Figure 2 are not affected by those factors.)

### 3.3 CoNLL NER Results

We finetuned BERT on the English portion of the CoNLL 2002/2003 NER corpus (Sang and De Meulder, 2003). We followed the text preprocessing in Devlin et al. (2019).

We used a learning rate of $6 \times 10^{-6}$ for the task loss, $6 \times 10^{-8}$ for the generator loss and $5 \times 10^{-4}$ for the discriminator loss.

In Table 2, we report the F1 scores for all of the CoNLL NER languages. When combined with adversarial learning, the BERT cross-lingual F1 scores increased for German over the

| | En | De | Es | Nl |
|---|---|---|---|---|
| Devlin et al. (2019) | 92.4 | - | - | - |
| Mayhew et al. (2017) | - | 57.5 | 66.0 | 64.5 |
| Ni et al. (2017) | - | 58.5 | 65.1 | 65.4 |
| Chen et al. (2019) | - | 56.0 | 73.5 | 72.4 |
| Xie et al. (2018) | - | 57.8 | 72.4 | 71.3 |
| BERT En-labels | 91.1 | 68.6 | **75.0** | 77.5 |
| BERT En-labels + Adv. | - | **71.9** | 74.3 | **77.6** |

Table 2: F1 scores on the CoNLL 2002/2003 NER test sets. We present results for BERT finetuned on labeled English data and BERT finetuned on labeled English data with language-adversarial training. Our results are averaged across 4 training runs, and hyperparameters are tuned on English dev data.

non-adversarial baseline, and the scores remained largely the same for Spanish and Dutch. Regardless, the BERT zero-resource performance far exceeds the results published in previous work.

Mayhew et al. (2017) and Ni et al. (2017) do use some cross-lingual resources (like bilingual dictionaries) in their experiments, but it appears that BERT with multilingual pretraining performs better, even though it does not have access to cross-lingual information.

### 3.4 Alignment of Embeddings for Parallel Documents

| Source | Target | Without Adv. | With Adv. |
|---|---|---|---|
| En | De | 0.74 | **0.94** |
| | Es | 0.72 | **0.94** |
| | Fr | 0.73 | **0.94** |
| | It | 0.73 | **0.92** |
| | Ja | 0.65 | **0.84** |
| | Ru | 0.72 | **0.89** |
| | Zh | 0.69 | **0.91** |

Table 3: Median cosine similarity between the mean-pooled BERT embeddings of MLDoc English documents and their translations, with and without language-adversarial training. The median cosine similarity increased with adversarial training for every language pair, which suggests that the adversarial loss encourages BERT to learn language-independent representations.

If language-adversarial training encourages language-independent features, then the English documents and their translations should be close in the embedding space. To examine this hypothesis, we take the English documents from the MLDoc training corpus and translate them into German, Spanish, French, etc. using Amazon Translate.

We construct the embeddings for each document using BERT models finetuned on MLDoc. We mean-pool each document embedding to create a single vector per document. We then calculate the cosine similarity between the embeddings for the English document and its translation. In Table 3, we observe that the median cosine similarity increases dramatically with adversarial training, which suggests that the embeddings became more language-independent.

## 4 Discussion

For many of the languages examined, we were able to improve on BERT's zero-resource cross-lingual performance on the MLDoc classification and CoNLL NER tasks. Language-adversarial training was generally effective, though the size of the effect appears to depend on the task. We observed that adversarial training moves the embeddings of English text and their non-English translations closer together, which may explain why it improves cross-lingual performance.

Future directions include adding the language-adversarial task during BERT pre-training on the multilingual Wikipedia corpus, which may further improve zero-resource performance, and finding better stopping criteria for zero-resource cross-lingual tasks besides using the English dev set.

## Acknowledgments

We would like to thank Jiateng Xie, Julian Salazar and Faisal Ladhak for the helpful comments and discussions.

## References


Mikel Artetxe and Holger Schwenk. 2018. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *arXiv preprint arXiv:1812.10464*.

Xilun Chen, Ahmed Hassan Awadallah, Hany Hassan, Wei Wang, and Claire Cardie. 2019. Multi-source cross-lingual model transfer: Learning what to share. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Alexis Conneau, Guillaume Lample, Ruty Rinott, Adina Williams, Samuel R Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. Xnli: Evaluating cross-lingual sentence representations. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*.

Mohammad Sadegh Hajmohammadi, Roliana Ibrahim, Ali Selamat, and Hamido Fujita. 2015. Combination of active learning and self-training for cross-lingual sentiment classification with density analysis of unlabelled samples. *Information sciences*, 317:67–77.

Shafiq Joty, Preslav Nakov, Lluís Màrquez, and Israa Jaradat. 2017. Cross-language learning with adversarial neural networks: Application to community question answering. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*.

Joo-Kyung Kim, Young-Bum Kim, Ruhi Sarikaya, and Eric Fosler-Lussier. 2017. Cross-lingual transfer learning for pos tagging without cross-lingual resources. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*.

Yichao Lu, Phillip Keung, Faisal Ladhak, Vikas Bhardwaj, Shaonan Zhang, and Jason Sun. 2018. A neural interlingua for multilingual machine translation. In *Proceedings of the Conference on Machine Translation (WMT)*.

Stephen Mayhew, Chen-Tse Tsai, and Dan Roth. 2017. Cheap translation for cross-lingual named entity recognition. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of NAACL-HLT*.

Jian Ni, Georgiana Dinu, and Radu Florian. 2017. Weakly supervised cross-lingual named entity recognition via effective annotation and representation projection. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1:8.

Leonardo Rigutini, Marco Maggini, and Bing Liu. 2005. An em based training algorithm for cross-language text categorization. In *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 529–535. IEEE Computer Society.

Erik F Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*.

Holger Schwenk and Xian Li. 2018. A corpus for multilingual document classification in eight languages. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*.

Jiateng Xie, Zhilin Yang, Graham Neubig, Noah A Smith, and Jaime Carbonell. 2018. Neural cross-lingual named entity recognition with minimal resources. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Daniel Zeman and Philip Resnik. 2008. Cross-language parser adaptation between related languages. In *Proceedings of the IJCNLP Workshop on NLP for Less Privileged Languages*.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Adversarial training for unsupervised bilingual lexicon induction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

# Recurrent Positional Embedding for Neural Machine Translation

**Kehai Chen, Rui Wang, Masao Utiyama, and Eiichiro Sumita**
National Institute of Information and Communications Technology (NICT), Kyoto, Japan
{khchen, wangrui, mutiyama, eiichiro.sumita}@nict.go.jp

## Abstract

In the Transformer network architecture, positional embeddings are used to encode order dependencies into the input representation. However, this input representation only involves static order dependencies based on discrete numerical information, that is, are independent of word content. To address this issue, this work proposes a recurrent positional embedding approach based on word vector. In this approach, these recurrent positional embeddings are learned by a recurrent neural network, encoding word content-based order dependencies into the input representation. They are then integrated into the existing multi-head self-attention model as independent heads or part of each head. The experimental results revealed that the proposed approach improved translation performance over that of the state-of-the-art Transformer baseline in WMT'14 English-to-German and NIST Chinese-to-English translation tasks.

## 1 Introduction

Transformer translation systems (Vaswani et al., 2017), without recurrent and convolutional neural networks, rely on a positional embedding (**PE**) approach to encode order information into the input representation. PE is typically learned based on the position index of each word and is added to corresponding word embedding. This allows the Transformer to encode order dependencies between words in addition to the words themselves. Finally, the Transformer uses these combined vectors as the input to self-attention networks (**SANs**), achieving state-of-the-art translation performance with several language pairs (Vaswani et al., 2017; Dou et al., 2018; Zhang et al., 2018a; Marie et al., 2018, 2019).

In spite of their success, the input representation only involves static order dependencies based on discrete numerical information. That is, any word in the entire vocabulary has the same PE on the same position index. As a result, the dependencies encoded by the original PEs are independent of word content, which may further hinder the improvement of translation capacity. Recently, Chen et al. (2018) and Hao et al. (2019) introduced the additional source representation learned by an RNN-based encoder into Transformer to alleviate this issue, and reported improvements on the WMT'14 English-to-German translation task.

Inspired by their works (Chen et al., 2018; Hao et al., 2019), we propose an simple and efficient recurrent positional embedding approach to capture order dependencies based on word content in a sentence, thus learning a more effective sentence representation for the Transformer. In addition, we designed two simple multi-head self-attentions to introduce these learned RPEs and original input representation into the existing Transformer model for enhancing the sentence representation of Transformer. Experimental results on WMT'14 English-to-German and NIST Chinese-to-English translation tasks show that our models significantly improved translation performance over a strong Transformer baseline.

## 2 Background

### 2.1 Input Representation

In the Transformer network architecture (Vaswani et al., 2017), given a sentence of length $J$, the positional embedding of each word is firstly computed based on its position:

$$\begin{aligned} \mathbf{pe}_{(j,2i)} &= \sin(j/10000^{2i/d_{model}}), \\ \mathbf{pe}_{(j,2i+1)} &= \cos(j/10000^{2i/d_{model}}), \end{aligned} \quad (1)$$

* Corresponding author



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1361–1367, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

where $j$ is the word's numerical position index in the sentence and $i$ is the dimension of position index. The word embedding $\mathbf{x}_j$ is then added with $\mathbf{pe}_j$ to get a combined embedding $\mathbf{v}_j$:

$$\mathbf{v}_j = \mathbf{x}_j + \mathbf{pe}_j. \quad (2)$$

As a result, there is a sequence of vectors $\{\mathbf{v}_1, \cdots, \mathbf{v}_J\}$ as the input to the encoder or decoder of the Transformer to learn the source or target sentence representations.

### 2.2 Self-Attention Mechanism

Following the input layer, the self-attention layer is used to learn the sentence representation for the Transformer (Vaswani et al., 2017). These combined vectors $\{\mathbf{v}_1, \cdots, \mathbf{v}_J\}$ are then packed into a query matrix $\boldsymbol{Q}$, their keys and values matrices $\boldsymbol{K}$ and $\boldsymbol{V}$. The output of the self-attention layer can be computed by Eq.(3):

$$\text{Att}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_{model}}})\boldsymbol{V}. \quad (3)$$

Moreover, self-attention can be further refined as multi-head self-attention to jointly attend to the information from different representation subspaces at different positions. Specifically, $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ are linearly projected $H$ times with different, learned linear projections to $d_k$ and $d_v$ dimensions, respectively. On each of these projected versions of queries, keys, and values, the attention function is performed in parallel, yielding $d_v$-dimensional output values. Take a single head as an example, the output of the $h$-th head $\boldsymbol{O}_h$ is computed by Eq.(4):

$$\boldsymbol{O}_h = \text{Att}(\boldsymbol{Q}\boldsymbol{W}_h^Q, \boldsymbol{K}\boldsymbol{W}_h^K, \boldsymbol{V}\boldsymbol{W}_h^V), \quad (4)$$

where the parameter matrices are $\boldsymbol{W}_h^Q \in \mathbb{R}^{d_{model} \times d_k}$, $\boldsymbol{W}_h^K \in \mathbb{R}^{d_{model} \times d_k}$, and $\boldsymbol{W}_h^V \in \mathbb{R}^{d_{model} \times d_v}$. For example, if there are $H$=8 heads and $d_{model}$ is 512, $d_k$=$d_v$=512/8=64. Finally, the outputs of $H$ heads are concatenated to serve as the sentence representation $\boldsymbol{S}$:

$$\boldsymbol{S} = \text{Concat}(\boldsymbol{O}_1, \cdots, \boldsymbol{O}_{H-1}, \boldsymbol{O}_H)\boldsymbol{W}^O, \quad (5)$$

where $\boldsymbol{W}^O \in \mathbb{R}^{Hd_v \times d_{model}}$ is a parameter matrix.

## 3 Recurrent Positional Embedding

We propose a recurrent positional embedding approach based on part of word embedding instead of numerical indices of words to capture order dependencies between words in a sentence. Specifically, the embedding of each word $\mathbf{x}_j$ is divided into two parts $\mathbf{x}_j^p$ and $\mathbf{x}_j^r$, and their dimensions are $d_p$ and $d_r$ ($d_{model}$=$d_p$+$d_r$), respectively. As a result, the sequence of word vectors $\{\mathbf{x}_1, \cdots, \mathbf{x}_J\}$ are spited into $\{\mathbf{x}_1^p, \cdots, \mathbf{x}_J^p\}$ and $\{\mathbf{x}_1^r, \cdots, \mathbf{x}_J^r\}$. An RNN with an non-linear projection layer is then designed to learn its recurrent state $\mathbf{r}_j$ for each word over $\{\mathbf{x}_1^r, \cdots, \mathbf{x}_J^r\}$:

$$\mathbf{r}_j = \text{Tanh}(f_{\text{RNN}}(\mathbf{x}_j^r, \mathbf{r}_{j-1})\mathbf{W}^r + \mathbf{b}^r), \quad (6)$$

where $\mathbf{W}^r \in \mathbb{R}^{d_r \times d_r}$ is a parameter matrix and $\mathbf{b}^r \in \mathbb{R}^{d_r}$ is a bias item.[1] Note that the $\mathbf{x}_j^r$ is derived from part of the word embedding $\mathbf{x}_j$. Finally, there is a sequence $\boldsymbol{R}$ =$\{\mathbf{r}_1, \cdots, \mathbf{r}_J\}$, called as recurrent positional embeddings (**RPEs**). In this work, a bidirectional RNN and a forward RNN (Bahdanau et al., 2015) are used to learn source RPEs and target RPEs, respectively. Noth that the RNN is also replaced by other neural networks for learning order dependency information, such as GRU (Cho et al., 2014), and SRU (Li et al., 2018a).

In addition, other sub-sequence $\{\mathbf{x}_1^p, \cdots, \mathbf{x}_J^p\}$ is used to gain the reduced dimension input representation $\boldsymbol{P}$=$\{\mathbf{p}_1, \cdots, \mathbf{p}_J\}$ according to the Section 2.1. Both of $\boldsymbol{R}$ and $\boldsymbol{P}$ will be together as the input to the encoder (or decoder) to learn a more effective source (or target) representation for the Transformer.

## 4 Neural Machine Translation with RPE

To make use of these learned RPEs, we propose two simple methods: RPE head (**RPEHead**) self-attention and mixed positional representation head (**MPRHead**) self-attention. Both of RPEHead and MPRHead can utilize RPEs to learn sentence representation for the Transformer.

### 4.1 RPEHead Self-Attention

For the RPEHead self-Attention, these learned RPEs are integrated into multi-head self-attention (Fig 1(a)) as several independent heads to learn the sentence representation, as shown in Fig 1(b).

First, we concatenate $\boldsymbol{P}$=$\{\mathbf{p}_1, \cdots, \mathbf{p}_J\}$ and $\boldsymbol{R}$ =$\{\mathbf{r}_1, \cdots, \mathbf{r}_J\}$ as a new combined input representation $\boldsymbol{\mathcal{T}}$. To perform the attention function in Eq.(3) over the combined input representation $\boldsymbol{\mathcal{T}}$, $d_r$ and $d_p$ are set to t*$d_{model}$/$H$

[1] The $\mathbf{r}_0$ is set as a zero vector.

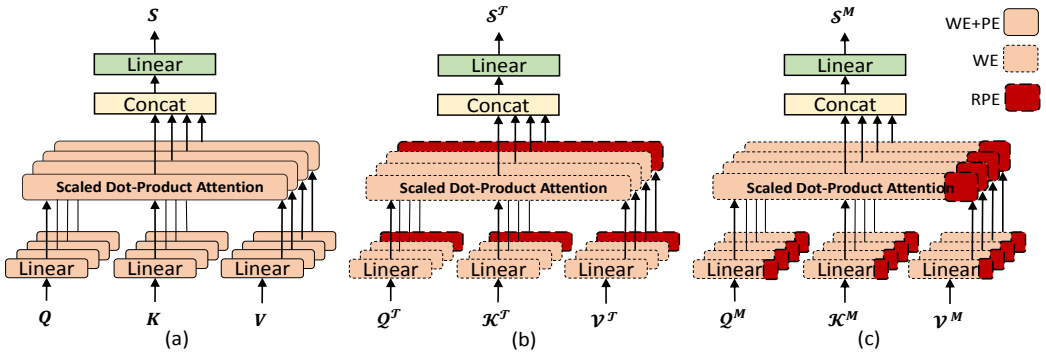


Figure 1: (a) Multi-Head Self-Attention; (b) RPEHead Self-Attention; (c) MPRHead Self-Attention.

and $d_{model}$-$d_r$ in the process of learning RPEs, respectively. This guarantees that there are two types of heads: one type of head only contains RPE and other type of head only contains the original reduced dimension input representation. Second, the $\mathcal{T}$ is mapped to a new query matrix $\mathcal{Q}^{\mathcal{T}}$, and their keys and values matrices $\mathcal{K}^{\mathcal{T}}$ and $\mathcal{V}^{\mathcal{T}}$. According to Eq.(4), the output of each head is computed by Eq.(7):

$$\mathcal{O}_h^{\mathcal{T}} = \text{Att}(\mathcal{Q}^{\mathcal{T}}\boldsymbol{W}_h^{\mathcal{Q}}, \mathcal{K}^{\mathcal{T}}\boldsymbol{W}_h^{\mathcal{K}}, \mathcal{V}^{\mathcal{T}}\boldsymbol{W}_h^{\mathcal{V}}). \quad (7)$$

Therefore, the final sentence representation $\mathcal{S}^{\mathcal{T}}$ is formally represented as:

$$\mathcal{S}^{\mathcal{T}} = \text{Concat}(\mathcal{O}_1^{\mathcal{T}}, \cdots, \mathcal{O}_{H-1}^{\mathcal{T}}, \mathcal{O}_H^{\mathcal{T}})\boldsymbol{\mathcal{W}}^O. \quad (8)$$

### 4.2 MPRHead Self-Attention

Compared with the RPEHead model, the MPRHead model applies the multi-head mechanism to the RPEs. In other words, RPEs are encoded into the sentence representation from different vector sub-spaces, as shown in Fig 1(c).

To this end, each vector of $\boldsymbol{P}$ is divided into the $H$ heads $\{\mathbf{p}_j^1, \cdots, \mathbf{p}_j^H\}$. Similarly, each vector of $\boldsymbol{R}$ is divided into the $H$ heads $\{\mathbf{r}_j^1, \cdots, \mathbf{r}_j^H\}$. The corresponding heads for $\mathbf{p}_j$ and $\mathbf{r}_j$ are then concatenated as a combined sequence $\mathbf{pr}_j$, in turn:

$$\mathbf{pr}_j = \{\mathbf{p}_j^1 : \mathbf{r}_j^1, \cdots, \mathbf{p}_j^H : \mathbf{r}_j^H\}, \quad (9)$$

where ":" is the concatenation operation. All heads in $\mathbf{pr}_j$ are further concatenated as a mixed embedding $\mathbf{m}_j \in \mathbb{R}^{d_{model}}$ in turn. As a result, there is a new sequence of mixed embeddings:

$$\boldsymbol{M} = \{\mathbf{m}_1, \cdots, \mathbf{m}_J\}. \quad (10)$$

The $\boldsymbol{M}$ is mapped to a query matrix $\mathcal{Q}^{\boldsymbol{M}}$, and their keys and values matrices $\mathcal{K}^{\boldsymbol{M}}$ and $\mathcal{V}^{\boldsymbol{M}}$. According to Eq.(4) and Eq.(5), the final sentence representation $\mathcal{S}^{\boldsymbol{M}}$ is represented as:

$$\mathcal{O}_h^{\boldsymbol{M}} = \text{Att}(\mathcal{Q}^{\boldsymbol{M}}\boldsymbol{W}_h^{\mathcal{Q}}, \mathcal{K}^{\boldsymbol{M}}\boldsymbol{W}_h^{\mathcal{K}}, \mathcal{V}^{\boldsymbol{M}}\boldsymbol{W}_h^{\mathcal{V}}),$$
$$\mathcal{S}^{\boldsymbol{M}} = \text{Concat}(\mathcal{O}_1^{\boldsymbol{M}}, \cdots, \mathcal{O}_{H-1}^{\boldsymbol{M}}, \mathcal{O}_H^{\boldsymbol{M}})\boldsymbol{\mathcal{W}}^O. \quad (11)$$

Note that for both RPEHead and MPRHead models, the RPEs will be jointly learned with the existing Transformer architecture.

## 5 Experiments

### 5.1 Experimental Setup

The proposed methods were evaluated on the WMT'14 English to German (EN-DE) and NIST Chinese-to-English (ZH-EN) translation tasks. The ZH-EN training set includes *1.28* million bilingual sentence pairs from the LDC corpora, where the NIST06 and the NIST02/NIST03/NIST04 data sets were used as the development and test sets, respectively. The EN-DE training set includes *4.43* million bilingual sentence pairs of the WMT'14 corpora, where the newstest2013 and newstest2014 data sets were used as the development and test sets, respectively.

The BPE (Sennrich et al., 2016) was adopted and the vocabulary size was set as 32K. The dimension of all input and output layers was set to *512*, and that of the inner feedforward neural network layer was set to *2048*. The total heads of all multi-head modules were set to *8* in both encoder and decoder layers. In each training batch, there was a set of sentence pairs containing approximately *4096*4* source tokens and *4096*4* target tokens. For the other setting not mentioned, we followed the setting in Vaswani et al. (2017).

Baseline systems included a vanilla Transformer (Vaswani et al., 2017), Relative PEs (Shaw et al., 2018), and directional SAN (**DiSAN**) (Shen

| System | Architecture | newstest2014 | #Param |
|---|---|---|---|
| *Existing NMT systems* | | | |
| Vaswani et al. (2017) | Transformer (base) | 27.3 | 65M |
| | Transformer (big) | 28.4 | 213M |
| Chen et al. (2018) | RNMT+SAN | 28.49 | 378.9M |
| Hao et al. (2019) | Transformer (base)+BiARN | 28.21 | 97.4M |
| | Transformer (big)+BiARN | 28.98 | 323.5M |
| *Our NMT systems* | | | |
| This work | Transformer (base) | 27.25 | 97.35M |
| | +Relative PE | 27.60 | 97.42M |
| | +DiSAN | 27.66 | 97.39M |
| | **+RPEHead** | 28.11* | 97.84M |
| | **+MPRHead** | 28.35* | 97.72M |
| | Transformer (big) | 28.22 | 272.6M |
| | **+MPRHead** | 29.11* | 289.1M |

Table 1: Results for EN-DE translation task. The mark "*" after scores indicates that the model was significantly better than the baseline Transformer (base or big) at the significance level $p$ <0.01 (Collins et al., 2005).

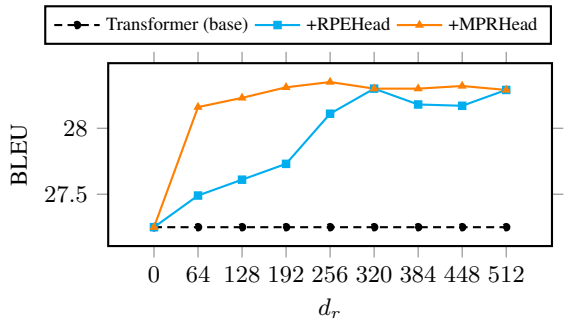


Figure 2: The BLEU scores on the different $d_r$.

et al., 2018). Besides, we reported results of the existing works (Vaswani et al., 2017; Chen et al., 2018; Shaw et al., 2018; Hao et al., 2019; Liu et al., 2019; Chen et al., 2019). We re-implemented the baseline Transformer, Relative PEs, and DiSAN models on the OpenNMT toolkit (Klein et al., 2017). All the models were trained for 200k batches and evaluated on a single V100 GPU. The multi-bleu.perl was used as the evaluation metric to obtain the case-sensitive 4-gram BLEU score of EN-DE and ZH-EN tasks.

### 5.2 Effect of RPEs

In this work, we extracted $d_r$ dimensions of each word vector to learn recurrent embeddings. To explore the relation between $d_r$ and translation performance, Figure 2 shows the translation performance on the different $d_r$. For +RPEHead (or +MPRHead), with the increasing in the dimension of $d_r$, BLEU scores are gradually increasing, but BLEU scores begin to decrease when $d_r$ is more than 320 (or 256). In particular, +RPEHead and +MPRHead achieve the highest BLEU scores at $d_r$=320 and $d_r$=256, respectively. This means that the original partial input representation and our RPE can complement each other to improve translation performance.

### 5.3 Main Results

According to the results of Fig 2, $d_r$ of RPEHead is set to 320 and $d_r$ of MPRHead is set to 256. The main translation results are shown in Table 1.

1) For the proposed methods, both RPEHead (base) and MPRHead (base) outperformed Transformer (base), especially are better than +RPE and +DiSAN. This indicates that the learned RPEs are beneficial for the Transformer system.

2) Moreover, +MPRHead (base) performed better than RPEHead/RPE (base). The reason may be that adding RPEs into variant heads can encode order dependencies based word vectors from variant vector sub-spaces, which is one of the advantages of the multi-head mechanism. In particular, +MPRHead (base/big) is slightly better than Transformer (base/big)+BiARN. This denotes that the RPEs can more effective improve the performance of Transformer.

3) MPRHead (big) was superior to Transformer (big) significantly. MPRHead (base) achieved a comparable performance compared to Transformer (big) which contains approximately three times parameters, indicating that the proposed RPE is efficient.

In addition, Table 2 shows that the proposed models also gave similar improvements over the baseline system and the compared methods on the NIST ZH-EN task. These results indicate that our

| System | Architecture | NIST02 | NIST03 | NIST04 | #Param |
|---|---|---|---|---|---|
| *Existing NMT systems* | | | | | |
| Liu et al. (2019) | Share-private Embeddings | 43.73 | 41.99 | 44.53 | N/A |
| Chen et al. (2019) | Reordering Embeddings | 47.54 | 46.56 | 47.27 | 78.3M |
| *Our NMT systems* | | | | | |
| This work | Transformer (base) | 46.62 | 45.90 | 45.95 | 77.94M |
| | +Relative PEs (Shaw et al., 2018) | 46.91 | 46.47 | 45.96 | 77.01M |
| | +RPEHead | 47.06 | 46.21 | 46.32 | 78.04M |
| | +MPRHead | 47.69* | 47.27* | 47.31* | 78.04M |
| | Transformer (big) | 47.88 | 47.58 | 47.37 | 243.7M |
| | +MPRHead | 48.73* | 48.61* | 48.21* | 245.5M |

Table 2: Results for ZH-EN translation Task. The mark "*" after scores indicates that the model was significantly better than the baseline Transformer (base or big) at the significance level $p$ <0.01 (Collins et al., 2005).

approach is a universal method for improving the translation of other language pairs.

### 5.4 Ablation Experiments

To further explore the effect of position information, PEs and RPEs were applied on the encoder (**Enc**) and decoder (**Dec**) side, respectively. Table 3 shows the results of ablation experiments on the EN-DE newstest2014 test set:

| | newstest2014 | #Speed |
|---|---|---|
| Transformer (base) | 27.25 | 11.7K |
| - PE of Dec | 26.73 | 11.7K |
| - PE of Enc | 10.71 | 11.7K |
| RPEHead (base) | 28.11 | 10.5K |
| - RPE&PE of Dec | 27.54 | 11.2K |
| - RPE&PE of Enc | 11.16 | 10.7K |
| MPRHead (base) | 28.35 | 9.9K |
| - RPE&PE of Dec | 27.74 | 11.2K |
| - RPE/PE of Enc | 10.89 | 10.7K |
| - RPE&PE of Enc&Dec | 10.43 | 11.7K |

Table 3: Ablation experiments of position information. "#Speed" denotes the training speed (tokens/second).

1) For Enc/Dec/Enc&Dec, the proposed RPEHead/MPRHead outperformed the Transformer in corresponding ablation settings, indicating that our methods worked well on both the encoder and decoder.

2) For PE/RPEs, performance became slightly lower when PE was removed from the decoder. However, when PE was removed from the encoder, the translation performance drastically decreased. We think that the source sentence representation is more sensitive to position information than the target sentence representation. It is possible that each hidden state in the decoder takes the previous hidden state into consideration, which would be somewhat similar to the proposed RPEs. In contrast, the encoder would not be expected to contain this mechanism. Therefore, the position information would be more important in the encoder than the decoder.

3) The training speeds of the proposed RPEHead (base) and MPRHead (base) were just slightly slower than those of the vanilla Transformer (base), because we nearly did not introduce additional (less than 2%) parameters.

## 6 Conclusion and Future Work

In this paper, we presented a recurrent embedding to capture the order dependencies in a sentence. Empirical results show that this method can improve the performance of NMT efficiently. In future work, we plan to extend this method to unsupervised NMT (Sun et al., 2019) and other natural language processing tasks, such as dependency parsing (Li et al., 2018b) and reading comprehension (Zhang et al., 2018b).

## Acknowledgments

We are grateful to the anonymous reviewers and the area chair for their insightful comments and suggestions. This work was partially conducted under the program "Research and Development of Enhanced Multilingual and Multipurpose Speech Translation Systems" of the Ministry of Internal Affairs and Communications (MIC), Japan. Masao Utiyama is partly supported by JSPS KAKENHI Grant Number 19H05660. Rui Wang was partially supported by JSPS grant-in-aid for early-career scientists (19K20354): "Unsupervised Neural Machine Translation in Universal Scenarios" and NICT tenure-track researcher startup fund "Toward Intelligent Machine Translation".

## References


Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations*, San Diego, CA.

Kehai Chen, Rui Wang, Masao Utiyama, and Eiichiro Sumita. 2019. Neural machine translation with reordering embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1787–1799, Florence, Italy. Association for Computational Linguistics.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Zhifeng Chen, Yonghui Wu, and Macduff Hughes. 2018. The best of both worlds: Combining recent advances in neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 76–86, Melbourne, Australia. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Michael Collins, Philipp Koehn, and Ivona Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 531–540, Ann Arbor, Michigan. Association for Computational Linguistics.

Zi-Yi Dou, Zhaopeng Tu, Xing Wang, Shuming Shi, and Tong Zhang. 2018. Exploiting deep representations for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4253–4262, Brussels, Belgium. Association for Computational Linguistics.

Jie Hao, Xing Wang, Baosong Yang, Longyue Wang, Jinfeng Zhang, and Zhaopeng Tu. 2019. Modeling recurrence for transformer. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1198–1207, Minneapolis, Minnesota. Association for Computational Linguistics.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada. Association for Computational Linguistics.

Jian Li, Zhaopeng Tu, Baosong Yang, Michael R. Lyu, and Tong Zhang. 2018a. Multi-head attention with disagreement regularization. In *Proceedings of the 2018 Conferenctemporale on Empirical Methods in Natural Language Processing*, pages 2897–2903, Brussels, Belgium. Association for Computational Linguistics.

Zuchao Li, Jiaxun Cai, Shexia He, and Hai Zhao. 2018b. Seq2seq dependency parsing. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3203–3214, Santa Fe, New Mexico, USA.

Xuebo Liu, Derek F. Wong, Yang Liu, Lidia S. Chao, Tong Xiao, and Jingbo Zhu. 2019. Shared-private bilingual word embeddings for neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3613–3622, Florence, Italy. Association for Computational Linguistics.

Benjamin Marie, Haipeng Sun, Rui Wang, Kehai Chen, Atsushi Fujita, Masao Utiyama, and Eiichiro Sumita. 2019. NICT's unsupervised neural and statistical machine translation systems for the WMT19 news translation task. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 294–301, Florence, Italy. Association for Computational Linguistics.

Benjamin Marie, Rui Wang, Atsushi Fujita, Masao Utiyama, and Eiichiro Sumita. 2018. NICT's neural and statistical machine translation systems for the WMT18 news translation task. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 449–455, Belgium, Brussels. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, New Orleans, Louisiana. Association for Computational Linguistics.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018. Disan:

Directional self-attention network for rnn/cnn-free language understanding. In *AAAI Conference on Artificial Intelligence*.

Haipeng Sun, Rui Wang, Kehai Chen, Masao Utiyama, Eiichiro Sumita, and Tiejun Zhao. 2019. Unsupervised bilingual word embedding agreement for unsupervised neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1235–1245, Florence, Italy. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Biao Zhang, Deyi Xiong, and Jinsong Su. 2018a. Accelerating neural transformer via an average attention network. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1789–1798, Melbourne, Australia. Association for Computational Linguistics.

Zhuosheng Zhang, Yafang Huang, and Hai Zhao. 2018b. Subword-augmented embedding for cloze reading comprehension. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1802–1814, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

# Machine Translation for Machines: the Sentiment Classification Use Case

**Amirhossein Tebbifakhr**[1,2], **Luisa Bentivogli**[1], **Matteo Negri**[1], **Marco Turchi**[1]
[1] Fondazione Bruno Kessler, Via Sommarive 18, Povo, Trento - Italy
[2] University of Trento, Italy
{atebbifakhr,bentivo,negri,turchi}@fbk.eu

## Abstract

We propose a neural machine translation (NMT) approach that, instead of pursuing adequacy and fluency ("human-oriented" quality criteria), aims to generate translations that are best suited as input to a natural language processing component designed for a specific downstream task (a "machine-oriented" criterion). Towards this objective, we present a reinforcement learning technique based on a new candidate sampling strategy, which exploits the results obtained on the downstream task as weak feedback. Experiments in sentiment classification of Twitter data in German and Italian show that feeding an English classifier with machine-oriented translations significantly improves its performance. Classification results outperform those obtained with translations produced by general-purpose NMT models as well as by an approach based on reinforcement learning. Moreover, our results on both languages approximate the classification accuracy computed on gold standard English tweets.

## 1 Introduction

Traditionally, machine translation (MT) pursues a "human-oriented" objective: generating fluent and adequate output to be consumed by speakers of the target language. But what if the intended use of MT is to feed a natural language processing (NLP) component instead of a human? This, for instance, happens when MT is used as a pre-processing step to perform a downstream NLP task in a language for which dedicated tools are not available due to the scarcity of task-specific training data. The rapid growth of cloud-based software-as-a-service offerings provides a typical example of this situation: a variety of affordable high-performance NLP tools can be easily accessed via APIs but often they are available only for a few languages. Translating into one of these high-resource languages gives the possibility to address the downstream task by: *i)* using existing tools for that language to process the translated text, and *ii)* projecting their output back to the original language.

However, using MT "as is" might not be optimal for different reasons. First, despite the qualitative leap brought by neural networks, MT is still not perfect (Koehn and Knowles, 2017). Second, previous literature shows that MT can alter some of the properties of the source text (Mirkin et al., 2015; Rabinovich et al., 2017; Vanmassenhove et al., 2018). Finally, even in the case of a perfect MT able to preserve all the traits of the source sentence, models are still trained on parallel data, which are created by humans and thus reflect quality criteria relevant for humans.

In this work, we posit that these criteria might not be the optimal ones for a machine (i.e. a downstream NLP tool fed with MT output). In this scenario, MT should pursue the objective of preserving and emphasizing those properties of the source text that are crucial for the downstream task at hand, even at the expense of human-quality standards. To this end, inspired by previous – human-oriented – MT approaches based on *Reinforcement Learning* (Ranzato et al., 2016; Shen et al., 2016) and *Bandit Learning* (Kreutzer et al., 2017; Nguyen et al., 2017), we explore a NMT optimization strategy that exploits the weak feedback from the downstream task to influence system's behaviour towards the generation of optimal "machine-oriented" output.

As a proof of concept, we test our approach on a sentiment classification task, in which Twitter data in German and Italian are to be classified according to their polarity by means of an English classifier. In this setting, a shortcoming of previous translation-based approaches (Denecke, 2008; Balahur et al., 2014) is that, similar to other traits,



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1368–1374,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

sentiment is often not preserved by MT (Salameh et al., 2015; Mohammad et al., 2016; Lohar et al., 2017). Although it represents a viable solution to leverage sentiment analysis to a wide number of languages (Araujo et al., 2016), the translation-based approach should hence be supported by advanced technology able to preserve the sentiment traits of the input. Along this direction, our experiments show that machine-oriented MT optimization makes the classifier's task easier and eventually results in significant classification improvements. Our results outperform those obtained with translations produced by general-purpose NMT models as well as by an NMT approach based on reinforcement learning (Ranzato et al., 2016). Most noticeably, on both languages we are able to approximate the classification accuracy computed on gold standard English tweets.

## 2 Background and Methodology

During training, NMT systems based on the encoder-decoder framework (Sutskever et al., 2014; Bahdanau et al., 2015) are optimized with maximum likelihood estimation (MLE), which aims to maximize the log-likelihood of the training data. In doing so, they indirectly model the human-oriented quality criteria (adequacy and fluency) expressed in the training corpus. A different strand of research (Ranzato et al., 2016; Shen et al., 2016; Kreutzer et al., 2018) focuses on optimizing the model parameters by maximizing an objective function that leverages either an evaluation metric like BLEU (Papineni et al., 2002) or an external human feedback. These methods are based on *Reinforcement Learning* (RL), in which the MT system parameters $\theta$ define a *policy* that chooses an *action*, i.e. generating the next word in a translation candidate $\hat{\mathbf{y}}$, and gets a *reward* $\Delta(\hat{\mathbf{y}})$ according to that action. Given $S$ training sentences $\{\mathbf{x}^{(s)}\}_{s=1}^{S}$, the RL training goal is to maximize the expected reward:

$$\begin{aligned}\mathcal{L}_{RL} &= \sum_{s=1}^{S} E_{\hat{\mathbf{y}}\sim p_\theta(.|\mathbf{x}^{(s)})}\Delta(\hat{\mathbf{y}}) \\ &= \sum_{s=1}^{S}\sum_{\hat{\mathbf{y}}\in\mathbf{Y}} p_\theta(\hat{\mathbf{y}}|\mathbf{x}^{(s)})\Delta(\hat{\mathbf{y}})\end{aligned} \quad (1)$$

where $\mathbf{Y}$ is the set of all translation candidates. Since the size of this set is exponentially large, it is impossible to exhaustively compute the expected reward, which is thus estimated by sampling one or few candidates from this set. In the MT adaptation (Ranzato et al., 2016) of REINFORCE (Williams, 1992), the closest approach to the one we present, only one candidate is sampled.

$$\hat{\mathcal{L}}_{RL} = \sum_{s=1}^{S} p_\theta(\hat{\mathbf{y}}|\mathbf{x}^{(s)})\Delta(\hat{\mathbf{y}}), \hat{\mathbf{y}} \sim p_\theta(.|\mathbf{x}^{(s)}) \quad (2)$$

We now focus on how the key elements of RL methods have been adapted to properly work in the proposed "MT for machines" setting. Our novel *Machine-Oriented* approach is described in Algorithm 1.

**Algorithm 1** *Machine-Oriented* RL

1: **Input:** $\mathbf{x}^{(s)}$ s-th source sentence in training data, $\mathbf{l}^{(s)}$ the ground-truth label, K number of sampled candidates
2: **Output:** sampled candidate $\hat{\mathbf{y}}^{(s)}$
3: $\mathbf{C} = \emptyset$ ▷ Candidates set
4: **for** $k$ = 1,...,K **do**
5: $\quad \mathbf{c} \sim p_\theta(.|\mathbf{x}^{(s)})$
6: $\quad \mathbf{f} = P_{class}(\mathbf{l}^{(s)}|\mathbf{c})$ ▷ Feedback from the classifier
7: $\quad \mathbf{C} = \mathbf{C} \cup (\mathbf{c}, \mathbf{f})$
8: $\hat{\mathbf{y}}^{(s)} = \max_{\mathbf{f}}(\mathbf{C})$ ▷ Best candidate w.r.t. feedback

**Reward Computation.** In current RL approaches to MT, rewards are computed either on a development set containing reference translations (Ranzato et al., 2016; Bahdanau et al., 2017) or by means of a weak feedback (e.g. a 1-to-5 score) when ground-truth translations are not accessible (Kreutzer et al., 2017; Nguyen et al., 2017). In both cases, the reward reflects a human-oriented notion of MT quality. Instead, in our machine-oriented scenario, the reward reflects the performance on the downstream task, independently from translation quality.

In the sentiment classification use case, given a source sentence ($\mathbf{x}^{(s)}$) and its sentiment ground-truth label ($\mathbf{l}^{(s)}$), the reward is defined as the probability assigned by the classifier to the ground-truth class for each translated sentence $\mathbf{c}$ ($P_{class}(\mathbf{l}^{(s)}|\mathbf{c})$ in Algorithm 1, line 6). By maximizing this reward, the MT system learns to produce a translation that has higher chances to be correctly labeled by the classifier. This comes at the risk of obtaining less fluent and adequate output. However, as will be shown in Section 4, this

type of reward induces highly polarized translations that are best suited for our downstream task.

**Sampling Approach.** A possible sampling strategy is to exploit beam search (Sutskever et al., 2014) to find, at each decoding step, the candidate with the highest probability. Another solution is to use multinomial sampling (Ranzato et al., 2016) which, at each decoding step, samples tokens over the model's output distribution. In (Wu et al., 2018), the higher results achieved by multinomial sampling are ascribed to its capability to better explore the probability space by generating more diverse candidates. This finding is particularly relevant in the proposed "MT for machines" scenario, in which the emphasis on final performance in the downstream task admits radical (application-oriented) changes in the behaviour of the MT model, even at the expense of human quality standards.

To increase the possibility of such changes, we propose a new sampling strategy. Instead of generating only one candidate *token* via multinomial sampling, $K$ candidate *sentences* are first randomly sampled (lines 4-5 in Algorithm 1). Then, the reward is collected for each of them (line 7) and the candidate with the highest reward is chosen (line 8). On one side, randomly exploring more candidates increases the probability to sample a "useful" one, possibly by diverting from the initial model behaviour. On the other side, selecting the candidate with the highest reward will push the system towards translations emphasizing input traits that are relevant for the downstream task at hand. In the sentiment classification use case, these are expected to be sentiment-bearing terms that help the classifier to predict the correct class.

## 3 Experiments

Our evaluation is done by feeding an English sentiment classifier with the translations of German and Italian tweets generated by:

- A general-purpose NMT system (*Generic*);
- The same system conditioned with REINFORCE (*Reinforce*);
- The same system conditioned with our *M*achine-*O*riented method (*MO-Reinforce*).

As other terms of comparison, we calculate the results of:

- The English classifier on the gold standard English tweets (*English*);
- German and Italian classifiers on the original, untranslated tweets (*Original*).

**Task-specific data.** We experiment with a dataset based on Semeval 2013 data (Nakov et al., 2013), which contains polarity-labeled parallel German/Italian–English corpora (Balahur et al., 2014). For each language pair, the development and test sets respectively comprise 583 (197 negative and 386 positive) and 2,173 tweets (601 negative and 1,572 positive). To cope with the skewed data distribution, the negative tweets in the development sets are over-sampled, leading to new balanced sets of 772 tweets.

**NMT Systems.** Our *Generic* models are based on Transformer (Vaswani et al., 2017), with parameters similar to those used in the original paper. Training data amount to 6.1M (De-En) and 4.56M (It-En) parallel sentences from freely-available corpora. The statistics of the parallel corpora are reported in Table 1. For each language pair, all data are merged and tokenized. Then, byte pair encoding (Sennrich et al., 2016) is applied to obtain 32K sub-word units.

| | De-En | It-En |
|---|---|---|
| Europarl | 2M | 2M |
| JRC | 0.7M | 0.8M |
| Wikipedia | 2.5M | 1M |
| ECB | 0.1M | 0.2M |
| TED | 0.1M | 0.2M |
| KDE | 0.3M | 0.3M |
| News11 | 0.2M | 0.04M |
| News | 0.2M | 0.02M |
| Total | 6.1M | 4.56M |

Table 1: Statistics of the parallel corpora used for training the generic NMT systems

To emulate both scarce and sufficient training data conditions, MT systems are trained using 5% and 100% of the available parallel data. In the most favorable condition (i.e. with 100% of the data), the BLEU score of the two models is 30.48 for De-En and 28.68 for It-En.

To condition the generic models and obtain the *Reinforce* and *MO-Reinforce* systems, we use the polarity-labeled German/Italian tweets in our de-

velopment sets, with reward as defined in Section 2. The SGD optimizer is used, with learning rate set to 0.01. In *MO-Reinforce*, the number of sampled candidates is set to $K = 5$.

**Classifiers.** To simulate an English cloud-based classifier-as-a-service, the pre-trained BERT (Base-uncased) model (Devlin et al., 2019) is fine-tuned with a balanced set of 1.6M positive and negative English tweets (Go et al., 2009). The German and Italian classifiers are also created by fine-tuning BERT on the polarity-labeled tweets composing the source side of our development set (772 tweets).[1] Before being passed to the classifiers, URLs and user mentions are removed from the tweets, which are then tokenized and lower-cased.

## 4 Results and Discussion

Table 2 shows our classification results, presenting the F1 scores obtained by the different MT-based approaches in the two training conditions. When NMT is trained on 100% of the parallel data, for both languages *Reinforce* produces translations that lead to classification improvements over those produced by the *Generic* model (+0.5 De-En, +0.8 It-En). Although the scores are considerably better than those obtained by the *Original* classifiers (+9.3 De-En, +7.2 It-En), the gap with respect to the *English* classifier is still quite large (-1.4 De-En and -2.3 It-En).

The observed F1 gains over the *Generic* model reflect an improvement in translation quality. Indeed, the BLEU score (not reported in the table) increases for both languages (+0.83 De-En, +1.37 It-En). As suggested by Kreutzer et al. (2018), this can be motivated by the fact that, rather than actually leveraging the feedback, RL mainly benefits from the optimization on in-domain source sentences in the development set. This is confirmed by the fact that *MO-Reinforce*, which uses a different sampling strategy, is able to outperform *Reinforce* on both languages (+0.7 F1 on De-En, +1.7 on It-En), approaching the *English* upper bound.

Unsurprisingly, the BLEU score obtained by *MO-Reinforce* is close to zero. Indeed, the generated sentences are highly polarized and fluent, but not adequate with respect to the source sentence. For example, the positive instances in the test set are translated into "*It's good!*", "*I'm happy*" or "*I'm grateful*", and the negative ones into "*it is not good!*" or "*I'm sorry*". This polarization shows that *MO-Reinforce* maximizes the exploitation of the received sentiment feedback thus producing an output that, at the expense of adequacy, can be easily classified by the downstream task.

| | De - En | | It - En | |
|---|---|---|---|---|
| | *5%* | *100%* | *5%* | *100%* |
| **Generic** | 79.7 | 83.2 | 78.2 | 81.6 |
| **Reinforce** | 80.4 | 83.7 | 77.8 | 82.8 |
| **MO-Reinforce** | **80.9** | **84.4** | **80.3** | **84.5** |
| **English** | 85.1 | | | |
| **Original** | 74.4 | | 75.6 | |

Table 2: Classification results (F1) obtained with: *i)* automatic English translations by three models (*Generic*, *Reinforce*, *MO-Reinforce*), and *ii)* gold-standard English (*English*) and untranslated German/Italian (*Original*) tweets.

When reducing the MT training data to only 5%, emulating a condition of parallel data scarcity, all the classification results decrease (on average by 3-4 points). However, also in this case the output of *MO-Reinforce* yields the closest scores to the *English* upper bound. This indicates that our method does not require large data quantities to outperform the other approaches.

To validate the hypothesis that *MO-Reinforce* can better leverage the feedback from the downstream task, Figures 1 and 2 show the average rewards for the De-En and the It-En candidates generated by *Reinforce* and *MO-Reinforce* at each epoch of the adaptation process. In line with the findings of Kreutzer et al. (2018), for both languages, *Reinforce* is not able to leverage the feedback and to generate candidates that increase their reward during training. Indeed, its curves in the two figures show either a stable (It-En) or even a slightly downward trend (De-En) that confirms the known limitations of NMT to preserve sentiment traits of the source sentences (see Section 1). In contrast, the *MO-Reinforce* curves show a clear upward trend indicating a higher capability to exploit the feedback and produce, epoch by epoch, increasingly polarized translations that are easier

[1] This strategy is similar to the one proposed in (Eriguchi et al., 2018), where a pre-trained multilingual encoder is used to build a cross-lingual classifier that is then trained on task-specific data. In our case, however, labeled data are available in a small amount.

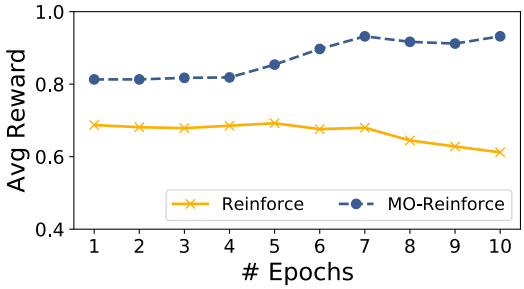


Figure 1: Average rewards for *Reinforce* and *MO-Reinforce* candidates at each training epoch (De-En).

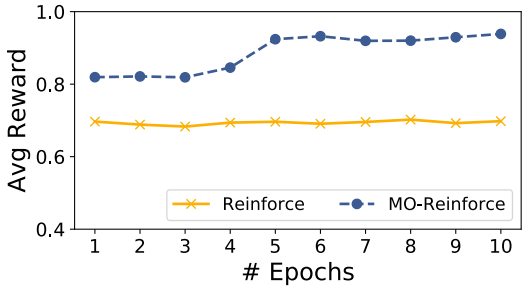


Figure 2: Average rewards for *Reinforce* and *MO-Reinforce* candidates at each training epoch (It-En).

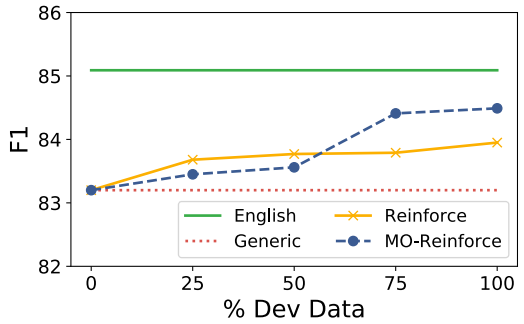


Figure 3: Classification results with increasing amounts of labeled development data (De-En).

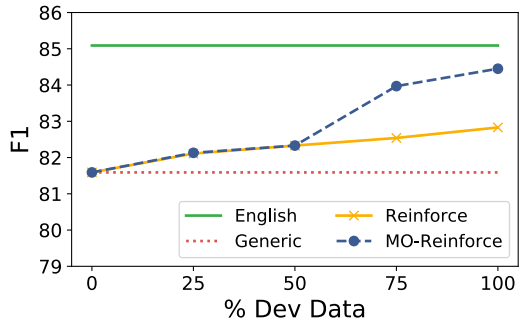


Figure 4: Classification results with increasing amounts of labeled development data (It-En).

to classify.

Another important aspect to investigate is the relation between the size of the development set (i.e. the amount of human-annotated source language tweets needed by the RL methods) and final classification performance. Figures 3 and 4 show the De-En and It-En performance variations of *Reinforce* and *MO-Reinforce* at different sizes of the development set. For comparison purposes, also the *Generic* and *English* results (which are independent from the development set size) are included. In both plots, each point is obtained by averaging the results of three different data shuffles. With limited amounts of data (25% and 50%) *Reinforce* and *MO-Reinforce* have a similar trend. When adding more data, *MO-Reinforce* shows a better use of the labeled development data, with a boost in performance that allows it to approach the *English* upper bound in both language settings. These results suggest that the effort to create labeled data can be minimal, and with 75% of the set (579 points) it is already possible to achieve a better performance than *Reinforce*.

## 5 Conclusions

We proposed a novel interpretation of the machine translation task, which pursues *machine-oriented* quality criteria (generating translations that are best suited as input to a downstream NLP component) rather than the traditional *human-oriented* ones (maximizing adequacy and fluency). We addressed the problem by adapting reinforcement learning techniques with a new, exploration-oriented sampling strategy that exploits the results obtained on the downstream task as weak feedback. Instead of generating only one candidate totally randomly via multinomial sampling (i.e. random stepwise selection of each word during generation, as in *Reinforce*), our approach selects $K$ full translation candidates, it computes the reward for each of them and finally chooses the one with the highest reward from the downstream task. As shown by our experiments in sentiment classification, this more focused (and application-oriented) selection allows our "MT for machines" approach to: *i)* better explore the hypotheses space, *ii)* make better use of the collected rewards and eventually *iii)* obtain better downstream classification results compared to translation-based solutions exploiting either general-purpose models or previous reinforcement learning strategies. In future work, we will target new application scenarios, covering multi-class classification and regression tasks.

## References


Matheus Araujo, Julio Reis, Adriano Pereira, and Fabricio Benevenuto. 2016. An evaluation of machine translation for multilingual sentence-level sentiment analysis. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, SAC '16, pages 1140–1145, Pisa, Italy. ACM.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2017. An actor-critic algorithm for sequence prediction. In *Proceedings of the 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015*.

Alexandra Balahur, Marco Turchi, Ralf Steinberger, José Manuel Perea Ortega, Guillaume Jacquet, Dilek Küçük, Vanni Zavarella, and Adil El Ghali. 2014. Resource creation and evaluation for multilingual sentiment analysis in social media texts. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC 2014, Reykjavik, Iceland, May 26-31, 2014.*, pages 4265–4269.

Kerstin Denecke. 2008. Using sentiwordnet for multilingual sentiment analysis. In *Proceedings of the 24th International Conference on Data Engineering Workshops, ICDE 2008, April 7-12, 2008, Cancún, Mexico*, pages 507–512.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota.

Akiko Eriguchi, Melvin Johnson, Orhan Firat, Hideto Kazawa, and Wolfgang Macherey. 2018. Zero-shot cross-lingual classification using multilingual neural machine translation. *CoRR*, abs/1809.04686.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12):2009.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39, Vancouver, Canada. Association for Computational Linguistics.

Julia Kreutzer, Shahram Khadivi, Evgeny Matusov, and Stefan Riezler. 2018. Can neural machine translation be improved with user feedback? In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*, pages 92–105, New Orleans - Louisiana. Association for Computational Linguistics.

Julia Kreutzer, Artem Sokolov, and Stefan Riezler. 2017. Bandit structured prediction for neural sequence-to-sequence learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1503–1513, Vancouver, Canada. Association for Computational Linguistics.

Pintu Lohar, Haithem Afli, and Andy Way. 2017. Maintaining Sentiment Polarity in Translation of User-Generated Content. *The Prague Bulletin of Mathematical Linguistics*, 108:73–84.

Shachar Mirkin, Scott Nowson, Caroline Brun, and Julien Perez. 2015. Motivating personality-aware machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1102–1108, Lisbon, Portugal. Association for Computational Linguistics.

Saif M. Mohammad, Mohammad Salameh, and Svetlana Kiritchenko. 2016. How translation alters sentiment. *Journal of Artificial Intelligence Research*, 55(1):95–130.

Preslav Nakov, Sara Rosenthal, Zornitsa Kozareva, Veselin Stoyanov, Alan Ritter, and Theresa Wilson. 2013. Semeval-2013 task 2: Sentiment analysis in twitter. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 312–320, Atlanta, Georgia, USA. Association for Computational Linguistics.

Khanh Nguyen, Hal Daumé III, and Jordan Boyd-Graber. 2017. Reinforcement learning for bandit neural machine translation with simulated human feedback. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1464–1474, Copenhagen, Denmark. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA.*, pages 311–318.

Ella Rabinovich, Raj Nath Patel, Shachar Mirkin, Lucia Specia, and Shuly Wintner. 2017. Personalized machine translation: Preserving original author traits. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages

1074–1084, Valencia, Spain. Association for Computational Linguistics.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *Proceedings of the 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016*.

Mohammad Salameh, Saif Mohammad, and Svetlana Kiritchenko. 2015. Sentiment after translation: A case-study on arabic social media posts. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 767–777, Denver, Colorado. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1683–1692, Berlin, Germany. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Eva Vanmassenhove, Christian Hardmeier, and Andy Way. 2018. Getting gender right in neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3003–3008, Brussels, Belgium. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256.

Lijun Wu, Fei Tian, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2018. A study of reinforcement learning for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3612–3621, Brussels, Belgium. Association for Computational Linguistics.

# Investigating the Effectiveness of BPE: The Power of Shorter Sequences

**Matthias Gallé**
NAVER LABS Europe
matthias.galle@naverlabs.com

## Abstract

Byte-Pair Encoding (BPE) is an unsupervised sub-word tokenization technique, commonly used in neural machine translation and other NLP tasks. Its effectiveness makes it a *de facto* standard, but the reasons for this are not well understood. We link BPE to the broader family of dictionary-based compression algorithms and compare it with other members of this family. Our experiments across datasets, language pairs, translation models, and vocabulary size show that – given a fixed vocabulary size budget – the fewer tokens an algorithm needs to cover the test set, the better the translation (as measured by BLEU).

## 1 Introduction

It is common practice in modern NLP to represent text in a continuous space. However, the interface between the discrete world of language and the continuous representation persists and is arguably nowhere so important as in deciding what should be the atomic symbols that will be used as input and/or output. While using words as tokens seems to be the most natural encoding, it has two major drawbacks: first it suffers from the out-of-vocabulary problems at inference time, and secondly the large number of words (which can easily reach the hundred of thousands) creates additional constraints on the memory and time that is required to train and infer with these models. On the other extreme, working at the character level has the advantage of a small and closed set. While current performance of character-level Neural Machine Translation (NMT) is competitive, its computational cost are much higher (up to 8 times (Cherry et al., 2018)). The commonly used intermediate approach is to pre-process the dataset by inferring sub-word tokens, which are then used as the atomic symbols to be translated. The most popular such method is known as **byte-pair encoding** (BPE), which merges iteratively the most frequent pair of symbols. Starting with characters as individual symbols, BPE creates new symbols representing bigrams and eventually higher-level order grams.

Such an approach has the advantage of incorporating some dataset-wide information in the local decision of what should be a token. It is however not linguistically motivated, and intents to create sub-word tokens based on morphological analyzer have failed to consistently improve the performance of the resulting translation (Zhou, 2018; Domingo et al., 2018), or do so only in special cases (morphological rich cases, target side only) (Huck et al., 2017; Pinnis et al., 2017). While this is unsatisfactory, to our knowledge the reasons for the effectivenesses of BPE has not been studied in the literature.

In this paper we hypothesize that the reason lies in the compression capacity of BPE: this is, that for two token vocabularies of the same size, the one that allows to cover better (with fewer number of symbols) a given sentence will achieve a better translation. To test this hypothesis, we link BPE to a large family of compression algorithms that aim to discover a dictionary of tokens and use that dictionary to parse the original dataset with the least amount of tokens. This allows us to take different algorithms from this family and to compare different unsupervised token inference techniques, all guided by the same compression principle. For a fixed number of token types, some of these algorithms need more tokens to cover the full sentence than BPE while others need less. Using different NMT models and experimenting across language pairs, datasets and vocabulary size, we show that the average number of tokens per sentence can be directly linked to the quality of translation.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1375–1381, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

## 2 Related Work and Context

### 2.1 Sub-word tokenization for NMT

Instead of relying on characters or words, unsupervised sub-word tokenization techniques induce a set of tokens that are then used to parse any new sequence. The standard approach is BPE, introduced for NMT in Sennrich et al. (2016) which merges incrementally the most frequent bigrams. Kudo (2018) proposes a different technique which allows the model to sample different tokenizations from the same word, resulting in a more robust translation model. The new tokenization method alone achieves comparable results, and no discussion of the size of the final parse is provided. In our experiments, we used the code of that paper.[1]

Alternative segmentations, using a morphological analyser for instance, have not shown consistent improvement (Zhou, 2018). Domingo et al. (2018) benchmark different segmentation techniques on different languages. While their results indicate that there is no clear winner, the vocabulary sizes of the different approaches are different and therefore harder to compare. Note that – differently from previous work (Cherry et al., 2018; Domingo et al., 2018) – our set-up allows us to compare models with the same vocabulary size. Comparing the compression capacity of models with different vocabulary size is misleading as a larger vocabulary size can obviously lead to smaller sequences. This however ignores the cost of expressing each symbol, which increases with the vocabulary size. Such an analysis could be achieved with an information-theoretical approach which is beyond the scope of this paper.

Indeed, BPE has the attractive characteristic that it allows to control the size of the vocabulary and can adapt to memory constraints. However, it introduces an additional hyper-parameter in the model and can be considered unsatisfactory from a modelling perspective. Recent proposals to avoid this include using several tokenizations at the same time, either hierarchically (Morishita et al., 2018) or through a lattice encoding (Su et al., 2017; Sperber et al., 2017); or dynamically, such as increasing the vocabulary size incrementally while training (Salesky et al., 2018) or reading the input sequence character by character and adding a halting gate which decides when to generate an output word (Kreutzer and Sokolov, 2018).

[1] https://github.com/google/sentencepiece

### 2.2 Byte-Pair Encoding

BPE is an instance of a the so-called *macro-based* compression algorithms, which seeks redundancies in the text by detecting repeated patterns, and compresses the sequences by replacing an occurrence of such a pattern with pointers to a previous occurrence. They achieve good compression by replacing subwords with (shorter) references, as opposed to statistical-based compression algorithms which are based on information theory and assign codewords to single symbols (but both can of course be combined). Its use for neural machine translation was introduced in (Sennrich et al., 2016), although a similar technique had been used previously for the creation of dictionaries and language models for voice search (Schuster and Nakajima, 2012). For the general framework, Storer and Szymanski (1982) provide a good overview of different possible frameworks of macro schemes. The authors differentiate between *external* and *internal* macro schemes. External macro schemes contain pointers to an external dictionary, while the pointers of internal macro schemes point to positions of the sequence itself.

From the external macro schemes, BPE belongs to the class of compression algorithms called **fixed size dictionary**. In this framework, the dictionary consists in a set of tokens $V = \{\omega_1, \dots \omega_n\}$ which are the atomic symbols. A sequence of $\omega_i$ has to be uniquely decodable, for instance by being of fixed length or prefix-free. In general in NMT, prefix-freeness is attained by separating these tokens with a special symbol (whitespace). Given a new sequence $s$ (the words from the test sequences), the goal is then to find $d_1 \dots d_m$, $d_i \in V$, such that their concatenation $d_1 \cdot d_2 \cdot \dots d_m = s$ and $m$ is minimal. This problem was proposed in 1973 by Wagner (1973) together with a dynamic algorithm that solves this problem in an optimal way. This optimal way is called **optimal parsing** by Bell et al. (1990) and "Minimal Space" by Schuegraf and Heaps (1974) where it is solved by a shortest-path algorithm. There also exist faster approximate algorithms: see Katajainen and Raita (1992) and Bell et al. (1990, Chapter 8.1).

This formulation assume that the set of words is given. Finding the set of words whose optimal parsing is minimal is an NP-complete problem: Storer and Szymanski (1982) proves this for variations where the pointers may be recursive (this is, enabling phrases themselves to be

parsed with pointer to other phrases) or the phrases may overlap. BPE is an instance of the recursive, non-overlapping version. This version is equivalent to the so-called Smallest Grammar Problem (Charikar et al., 2005), because the resulting parsing can be formalized as a non-branching, non–auto-recursive context free grammar, also called a straight-line program. This problem can be seen as a specific formulation of the Minimum Description Length principle (Rissanen, 1978) which states that the best model for some data is the one which minimizes cost of describing the data given the model *plus* the cost of describing the model. In a straight-line grammar the description of the data can be seen as the right-hand side of the starting non-terminal, while the description of the model is the combined parsing cost of all other non-terminals. This principle has been applied in a very similar form for unsupervised inference of linguistic structures in (De Marcken, 1996).

Several algorithms that approximate the Smallest Grammar Problem have been given in the literature, including the one that iteratively replaces the most frequent pair. The name BPE comes from Gage (1994) who applied it to data compression, although the same idea had been used before for pattern discovery in natural language (Wolff, 1975) as well as measuring the complexity of genetic sequences (Ángel Jiménez-Montaño, 1984). Larsson and Moffat (2000) propose the REPAIR implementation, whose time complexity is linear with respect to the size of the input sequence. A theoretical analysis of its compression capacity can be found in Navarro and Russo (2008), and recent research focuses on reducing its memory footprint (Gagie et al., 2019).

## 3 Other compression algorithms

We investigate the impact that different such algorithms have on the final translation, using the same budget on the vocabulary size. Our hypothesis is that smaller parses would result in improved translation, and that therefore algorithms that infer a dictionary of tokens that cover the sequence better should be preferred.

Despite its simplicity, BPE performs very well in standard compression benchmarks (Gage, 1994; Carrascosa et al., 2012). The best performing ones have an unreasonable time-complexity (including one of complexity $\mathcal{O}(n^7)$ (Carrascosa et al., 2011) and an even slower genetic algorithm (Benz and Kötzing, 2013)). Based on these benchmarks, we decided to use the so-called IRRMGP algorithm, which outperforms BPE, as well as other worse performing algorithms. **IRRMGP** works as follows: as BPE it is an iterative repeat replacement (IRR) algorithm, but instead of choosing in each iteration the most frequent bigram, it chooses the substring $\omega$ that maximizes $|\omega| \times occ(\omega)$ (where $occ(\omega)$ is the number of occurrences of $\omega$ in the training corpus). Once it has reached the desired number of tokens, it re-parses the sequences through an optimal parsing of the original sequence *and* inferred tokens so far (minimal grammar parsing, MGP), removes any un-used token and continues looking for new words. The modules of IRR and MGP are iterated until the dictionary budget is reached. While a worst-case analysis bounds its running time as $\mathcal{O}(n^4)$, in practice its running time increases quadratically with the sequence length (Carrascosa et al., 2012). Those high running times however precludes our analysis to run on very large data-sets.

We also compare against two other compression algorithm. **MAXLEN** (Bentley and McIlroy, 1999) is another iterative repeat replacement algorithm, but selects the longest repeat in each iteration. It can also be implement in linear-time (Nakamura et al., 2009). The second algorithm is the popular **SEQUITUR** (Nevill-Manning and Witten, 1997). It runs on-line, maintaining two invariants while it reads the sequence: (i) bigram uniqueness (no bigram appears more than once) and (ii) rule utility (each non-terminal is used more than once). Assuming a constant time hashing function, SEQUITUR runs in (amortized) linear time with respect to the sequence length, and only requires a linear amount of memory. In our experiments, the original implementation of SEQUITUR[2] runs about 3 times faster than SENTENCEPIECE.

### 3.1 Core tokens

It is not straightforward to run SEQUITUR to obtain a fixed number of tokens, as it constructs the dictionary online. For this, we run it until the end, and select a posteriori a set of good tokens. For that selection process, we use the parse tree of the final grammar and compute the "core nodes" (Siyari et al., 2016). This corresponds to the set of

[2] https://github.com/craignm/sequitur

tokens which are used the most in the hierarchical parse of the dataset. Siyari et al. (2016) show that the number of times token $\omega$ is used is equal to $|\omega| \times freq(\omega)$, where $freq(\omega)$ is the number of times $\omega$ appears in the derivation of the unique target sequence. The core is then computed by taking the highest scoring such token, removing it, updating the parse and iterating until reaching the vocabulary size.

## 4 Experiments

We worked on the lowercased Kyoto Free Translation Task (Neubig, 2011, KFTT) and IWSLT'14 (en↔de), without removing any sentences (see Table 1). For the translation model, we used ConvS2S (Gehring et al., 2017) and Transformer (Vaswani et al., 2017), allowing us to compare models with different field of view: local for ConvS2S, and global for Transformer. For ConvS2S, we used 4 layers and an embedding size of 256, dropout of 0.2, an initial step-size of 0.25. For Transformer, we used 4 attention heads, dropout was set to 0.5 and initial step-size to 0.001 (except for MAXLEN, for which we used 10e-4). These parameters were fixed in initial experiments maximising the performance on BPE. We ran training for 150 epochs, and use the average of the three best models (as measured on the validation set).

To apply sub-word tokenization over unseen sentences, several approaches are possible. First we tried to run an optimal parsing on each word with the given token set but the resulting translations were worse than the baseline BPE model. The reported results here are obtained with a greedy parsing, where the words in the dictionary are first sorted and then applied from the top downwards. Words are sorted first by their length (longer first) and then by their frequency (more frequent first). Using a consistent tokenization – applied equally in training and inference – seems to help the model, while in an optimal parsing setting a slight change in the context could substantially change the parsing. We believe that this brittleness causes the worse performance of the optimal parsing. Note that the standard BPE procedure also applies greedily the dictionary of words, in the order they were generated. Sub-word regularization (Kudo, 2018) could be used to make the training more robust to this tokenization mismatch. Moreover, that approach could be adapted to work with any of the inferred tokenization: while a vanilla optimal parsing does not seem to support a probabilistic approach which could allow a sampling procedure, there might be several optimal parsings of one word (Carrascosa et al. (2011) show both theoretical and empirical evidence that there can be an exponential number of parses with the same size) and generating several of those could make the translation system more robust.

We compare BPE, IRRMGP, MAXLEN, the core on the words inferred by SEQUITUR and an additional baseline RANDOM: given a vocabulary budget of $N$, we pick up $N$ random *maximal repeats* (Gusfield, 1997) of the multi-sequence set of all tokens. Those are then applied to the training and testing data as before (applying them greedily, first by length – longer first – and then by frequency – more frequent first).

The training data of both languages were concatenated, and we used two different vocabulary sizes: 16k and 32k. Tokenization in all cases was done at the sub-word level, so that no token crossed word boundaries. Inference is done with a beam search of size 5, and for BLEU evaluation we used SacreBleu.[3]

The results in Fig. 1 show that there is indeed a strong correlation between the number of tokens used to cover the test sentences, and the resulting BLEU score. BPE is competitive with the best used compression algorithm, something which can be attributed to its compression capacity which is only slightly worse than that of IRRMGP (sometimes hard to distinguish for IWSLT in Fig. 1 because the numbers are relatively close).

## 5 Conclusion

In this paper we provide evidence that the effectiveness of BPE can be linked to its compression capacity, that is, the capacity of finding a set of words such that they are able to cover the sequence with as few words as possible. For this, we use other algorithms of the same family which allows us to control the number of tokens, as it is obviously easier to cover the sequence better if we can access more tokens. Our benchmark shows that the size of the final coverage can indeed be strongly linked to the translation quality.

Those conclusions however do not preclude the

[3] `BLEU+case.mixed+numrefs.1+smooth.exp+tok.intl+version.1.2.17`

| name | lang | type | train | dev | test |
|---|---|---|---|---|---|
| KFTT | ja ↔ en | wikipedia | 329.9k | 1.2k | 1.2k |
| IWSLT | de ↔ en | TED(x) talks | 160.2k | 7.3K | 6.8K |

Table 1: Datasets characteristics.

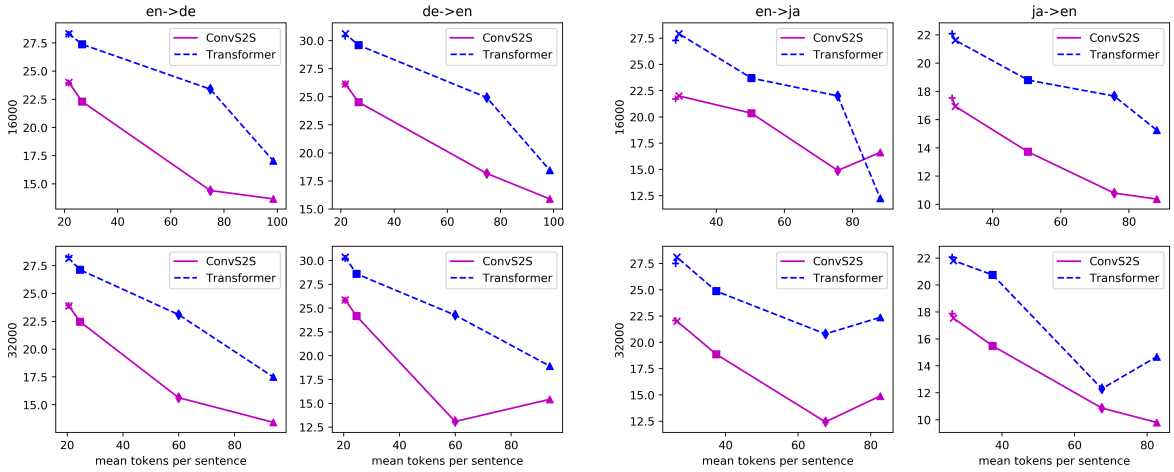


Figure 1: $x$-axis is the average number of tokens of the training sentences (source and target) and $y$-axis the BLEU score for 16k and 32k tokens (first and second row respectively). The markers correspond to IRRMGP (+), BPE (×), SEQUITUR (■), RANDOM (♦) and MAXLEN (▲). Each point corresponds to the average over 5 training runs.

use of morphological knowledge. For example, Huck et al. (2017) show that a combined approach can sometimes outperform pure BPE tokenization. Following standard practices in NMT, in our experiments we only performed intra-word tokenization, using the white-space as a starting segmentation beyond which no token could be formed. When this restriction was removed, the resulting sequences were even smaller but BLEU scores decreased. While this might be because the hyperparameters were set to maximise the BLEU score of BPE intra-word, it could be an indication that some linguistic prior (like word tokenization) trumps this specific formalization of Occam's Razor.

## Acknowledgements

We thank the anonymous reviewers and meta-reviewer for their insightful remarks and propositions, as well as Laurent Besacier and Alexandre Bérard for fruitful discussions.

## References

Timothy Bell, John Cleary, and Ian H Witten. 1990. *Text Compression*. Prentice Hall.

Jon Bentley and Douglas McIlroy. 1999. Data compression using long common strings. In *DCC*, pages 287–295. IEEE.

Florian Benz and Timo Kötzing. 2013. An effective heuristic for the smallest grammar problem. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pages 487–494. ACM.

Rafael Carrascosa, François Coste, Matthias Gallé, and Gabriel Infante-Lopez. 2011. The smallest grammar problem as constituents choice and minimal grammar parsing. *Algorithms*, 4(4):262–284.

Rafael Carrascosa, François Coste, Matthias Gallé, and Gabriel Infante-Lopez. 2012. Searching for smallest grammars on large sequences and application to dna. *Journal of Discrete Algorithms*, 11:62–72.

Moses Charikar, Eric Lehman, Ding Liu, Rina Panigrahy, Manoj Prabhakaran, April Rasala, Amit Sahai, and abhi shelat. 2005. The smallest grammar problem. *IEEE Transactions on Information Theory*, 51(7):2554–2576.

Colin Cherry, George Foster, Ankur Bapna, Orhan Firat, and Wolfgang Macherey. 2018. Revisiting character-based neural machine translation with capacity and compression. In *EMNLP*.

Carl De Marcken. 1996. Linguistic structure as composition and perturbation. In *Proceedings of the 34th*

*annual meeting on Association for Computational Linguistics*, pages 335–341. Association for Computational Linguistics.

M. Domingo, M. Garcıa-Martınez, A. Helle, F. Casacuberta, and M. Herranz. 2018. How Much Does Tokenization Affect Neural Machine Translation? *arXiv e-prints*.

Philip Gage. 1994. A new algorithm for data compression. *The C Users Journal*, 12(2).

Travis Gagie, Giovanni Manzini, Gonzalo Navarro, Hiroshi Sakamoto, Yoshimasa Takabatake, et al. 2019. Rpair: Rescaling repair with rsync. *arXiv preprint arXiv:1906.00809*.

Jonas Gehring, Michael Auli, David Grangier, and Yann Dauphin. 2017. A convolutional encoder model for neural machine translation. In *ACL*, pages 123–135. Association for Computational Linguistics.

Dan Gusfield. 1997. *Algorithms on strings, trees, and sequences: computer science and computational biology*. Cambridge university press.

Matthias Huck, Simon Riess, and Alexander Fraser. 2017. Target-side word segmentation strategies for neural machine translation. In *WMT*, pages 56–67, Copenhagen, Denmark. Association for Computational Linguistics.

Miguel Ángel Jiménez-Montaño. 1984. On the syntactic structure of protein sequences and the concept of grammar complexity. *Bulletin of Mathematical Biology*, 46(4):641–659.

Jyrki Katajainen and Timo Raita. 1992. An analysis of the longest match and the greedy heuristics in text encoding. *Journal of the ACM*, 39(2).

J. Kreutzer and A. Sokolov. 2018. Learning to Segment Inputs for NMT Favors Character-Level Processing. In *IWSLT*.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *ACL*.

N Jesper Larsson and Alistair Moffat. 2000. Off-line dictionary-based compression. *Proceedings of the IEEE*, 88(11):1722–1732.

Makoto Morishita, Jun Suzuki, and Masaaki Nagata. 2018. Improving neural machine translation by incorporating hierarchical subword features. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 618–629.

Ryosuke Nakamura, Shunsuke Inenaga, Hideo Bannai, Takashi Funamoto, Masayuki Takeda, and Ayumi Shinohara. 2009. Linear-time text compression by longest-first substitution. *Algorithms*, 2(4):1429–1448.

Gonzalo Navarro and Luís Russo. 2008. Re-pair achieves high-order entropy. In *Data Compression Conference*, page 537.

Graham Neubig. 2011. The Kyoto free translation task. http://www.phontron.com/kftt.

Craig G Nevill-Manning and Ian H Witten. 1997. Compression and explanation using hierarchical grammars. *The Computer Journal*, 40(2,3):103–116.

Mārcis Pinnis, Rihards Krišlauks, Daiga Deksne, and Toms Miks. 2017. Neural machine translation for morphologically rich languages with improved subword units and synthetic data. In *Text, Speech, and Dialogue*, pages 237–245, Cham. Springer International Publishing.

Jorma Rissanen. 1978. Modeling by shortest data description. *Automatica*, 14(5):465–471.

E. Salesky, A. Runge, A. Coda, J. Niehues, and G. Neubig. 2018. Optimizing Segmentation Granularity for Neural Machine Translation. *arXiv e-prints*.

Ernst J Schuegraf and H S Heaps. 1974. A comparison of algorithms for data base compression by use of fragments as language elements. *Information Storage and Retrieval*, 10:309–319.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *ICASSP*, pages 5149–5152.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL*, volume 1, pages 1715–1725.

Payam Siyari, Bistra Dilkina, and Constantine Dovrolis. 2016. Lexis: An optimization framework for discovering the hierarchical structure of sequential data. In *KDD*, pages 1185–1194. ACM.

Matthias Sperber, Graham Neubig, Jan Niehues, and Alex Waibel. 2017. Neural lattice-to-sequence models for uncertain inputs. *arXiv preprint arXiv:1704.00559*.

James A Storer and Thomas G Szymanski. 1982. Data compression via textual substitution. *Journal of the ACM*, 29(4):928–951.

Jinsong Su, Zhixing Tan, Deyi Xiong, Rongrong Ji, Xiaodong Shi, and Yang Liu. 2017. Lattice-based recurrent neural network encoders for neural machine translation. In *AAAI*, pages 3302–3308.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Robert Wagner. 1973. Common phrases and minimum-space text storage. *Communications of the ACM*, 16(3).

J Gerard Wolff. 1975. An algorithm for the segmentation of an artificial language analogue. *British Journal of PsychologyJ*, 66:79–90.

Giulio Zhou. 2018. Morphological zero-shot neural machine translation. Master's thesis, University of Edinburgh.

# HABLex: Human Annotated Bilingual Lexicons for Experiments in Machine Translation

**Brian Thompson*** **Rebecca Knowles*** **Xuan Zhang*** **Huda Khayrallah**
**Kevin Duh** **Philipp Koehn**
Johns Hopkins University
{brian.thompson, rknowles, xuanzhang, huda, phi}@jhu.edu
kevinduh@cs.jhu.edu

## Abstract

Bilingual lexicons are valuable resources used by professional human translators. While these resources can be easily incorporated in statistical machine translation, it is unclear how to best do so in the neural framework. In this work, we present the HABLex dataset, designed to test methods for bilingual lexicon integration into neural machine translation. Our data consists of human-generated alignments of words and phrases in machine translation test sets in three language pairs (Russian→English, Chinese→English, and Korean→English), resulting in clean bilingual lexicons which are well matched to the reference. We also present two simple baselines—constrained decoding and continued training—and an improvement to continued training to address overfitting.

## 1 Introduction

Neural machine translation (NMT) is the current state-of-the-art. In contrast with statistical machine translation (SMT; Koehn et al., 2007), where there are several established methods of incorporating external knowledge,[1] recent work is still examining how best to incorporate bilingual lexicons into NMT systems. Bilingual lexicon integration is desirable in a number of scenarios: highly technical vocabulary (which might be rare, or require translations of a domain-specific sense), lower-resource settings (where bilingual lexicons might be a significant portion of the available parallel data), translation settings where a client requires particular terms to be used (e.g. brand names), or for improving rare word translation.

At present, there is no standard dataset for benchmarking bilingual lexicon integration, making it difficult to compare methods.

| | |
|---|---|
| Source | на передней стенке корпуса установлены **амортизаторы** . |
| Lexical Entry | **амортизаторы↔shock absorbers** |
| Target | **shock absorbers** are mounted on a front wall of the housing . |

Table 1: Example Russian source sentence, lexicon entry, and English target sentence.

We create and release[2] **H**uman **A**nnotated **B**ilingual **Lex**icons (HABLex). Our bilingual lexicons are (1) generated by bilingual experts to ensure high quality, (2) derived from the development and test references so the best-case-scenario impact on translation performance can be directly measured, (3) covering 3 language pairs, and (4) focused on challenging words.

We perform exploratory work on our development set, showing two representative baselines to compare incorporating the lexicon at training time (continued training) vs. at decoding time (constrained decoding). We examine the tradeoffs in terms of BLEU, recall, and speed.

We also present a novel application of Elastic Weight Consolidation (EWC; Kirkpatrick et al., 2017; Thompson et al., 2019) which significantly improves performance by preventing overfitting during continued training on the bilingual lexicon.

## 2 Related Work

We first review prior work on incorporation of bilingual lexicons into NMT and then discuss the datasets used and explain how our new dataset addresses some shortcomings.

Recent work on the incorporation of bilingual lexicons into NMT systems can be loosely clus-

*These authors contributed equally to this work.

[1] E.g. requiring a particular translation via XML markup: statmt.org/moses/?n=advanced.hybrid

[2] http://www.cs.jhu.edu/~kevinduh/a/hablex2019/



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1382–1387, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

tered into two categories: incorporation at training time or during decoding. These two general approaches have different performance characteristics: incorporation at training time may slow down training, but tends not to alter inference speed, while incorporation at inference time tends to significantly slow down decoding (see subsection 4.3), but without slowing down training.

### 2.1 Incorporation at Training Time

Zhang and Zong (2016) and Fadaee et al. (2017) both propose using bilingual lexicons to create synthetic bitext to augment training data for NMT systems. Arthur et al. (2016) use translation probabilities from a lexicon (like SMT phrase tables) in conjunction with NMT probabilities.

### 2.2 Incorporation at Decode Time

Kothur et al. (2018) perform fine-grained continued training adaptation on very small, document-specific bilingual lexicons of novel words.[3]

A popular inference-time approach is constrained decoding (Anderson et al., 2016; Hokamp and Liu, 2017; Chatterjee et al., 2017; Hasler et al., 2018; Post and Vilar, 2018), which modifies beam search to require that user-specified words or phrases to be present in the output hypotheses. Constrained decoding can be used to ensure that target entries from a bilingual lexicon be present in the MT output whenever their corresponding source entries are present in the input. Constrained decoding with multiple target options (e.g. when a source word can be translated into one of several target words) is addressed in Chatterjee et al. (2017) and Hasler et al. (2018).

### 2.3 Datasets Used in Prior Studies

Prior work has used either human-generated general purpose bilingual lexicons not tailored to a test set (Arthur et al., 2016; Zhang and Zong, 2016) or automatic alignments (Fadaee et al., 2017; Hokamp and Liu, 2017; Chatterjee et al., 2017; Hasler et al., 2018; Kothur et al., 2018) which likely contain errors (especially on rare words). There exist manually word-aligned parallel corpora for Japanese-English Wikipedia data (Neubig, 2011) and for Chinese-English mixed-domain data (Liu and Sun, 2015), from which it would be possible to extract lexicons.

[3]Although this method involves adapting the model, it is done prior to translating each document, so we view it as a decoding-time modification.

Most prior work experimented in a single language pair,[4] making it difficult to predict if a method will generalize. Our work addresses these issues by providing a multi-language testbed for comparisons, focusing on a consistent and lexically-challenging domain.

## 3 HABLex Dataset

Our motivation is to allow straightforward evaluation of lexicon incorporation methods. By building a reference-derived bilingual lexicon, we ensure that if a method successfully learns correct translations from the lexicon while maintaining overall system performance, BLEU will increase.

We annotate the existing parallel text with alignments for certain vocabulary items, allowing us to extract bilingual lexicons that are specific to the context in which the translation should appear. This produces a cleaner and more well-tailored bilingual lexicon than would be found in most real-world scenarios. However, it could also be used as a standardized starting point to produce a noisier lexicon that more closely mimics real-world lexicons (e.g. by adding irrelevant entries or relevant morphological variants, lemmatizing entries, or subsampling).

At a high level, our lexicons are created by a two-step process: (1) identifying interesting words on the source side of the test and development sets, and (2) human annotators correcting or validating automatic alignments of the identified words.

### 3.1 Patent Domain & Languages

We chose the patent domain because it contains interesting technical terminology and because of the availability of the high-quality, multilingual World Intellectual Property Organization (WIPO) COPPA-V2 corpus (Junczys-Dowmunt et al., 2016). We build bilingual lexicons for three language pairs: Russian→English (Ru→En), Korean→English (Ko→En), and Chinese→English (Zh→En). We also release the development and test splits from which the annotations were produced.

### 3.2 Interesting Word Selection

We select interesting words (from the source side of the data) as follows: words that appear less than 5 times in WIPO data, words that appear less than

[4]Except Hokamp and Liu (2017) and Hasler et al. (2018).

| | Development | | Test | |
|---|---|---|---|---|
| | Entries | Sentences | Entries | Sentences |
| Ru | 8001 | 2142 | 9040 | 2412 |
| Ko | 5595 | 1756 | 5593 | 1744 |
| Zh | 1352 | 1025 | 1042 | 885 |

Table 2: Number of bilingual lexicon entries and sentences containing at least one annotation for each language pair. Most source entries (between 87% and 96%) have a unique translation; the remainder have 7 or fewer target translations.

5 times in general domain training data (see section 4), and words that appear less than 5 times in both.[5] These represent three types of words that may be challenging to translate: words that have only recently been learned, words that may have been forgotten during continued training, and generally rare words.

### 3.3 Human Annotations

Our bilingual annotators used the LDC Word Aligner tool (Grimes, 2010) to annotate words of interest in context. Table 1 shows example lexicon entries resulting from this annotation process. In order to save annotator time and effort, we displayed highlighted automatic alignments[6] of each of the words of interest in their parallel sentence context (similar to Grimes et al. (2012)). The annotator confirms or corrects the automatic alignment, adding or removing source or target side words as needed to complete a valid alignment. Since annotation is cumbersome with very long sentences, we omit sentences of length 100 or more tokens. Filtering out numerical entries, discontiguous entries, and phrases longer than 3 words[7] results in bilingual lexicons with sizes shown in Table 2.

## 4 Baseline Experiments and Results

To build strong baseline systems, we first train models on general domain data. As general domain data, we use the OpenSubtitles18 (Lison et al., 2018) corpora for both Ru→En and Ko→En. For Ru→En and Zh→En we also use the parallel portion of the WMT17 news translation task (Bojar et al., 2017). We then fine-tune these general-domain models on WIPO training data (Luong and Manning, 2015), using the dev set for validation. These domain-adapted models are then used as the initial systems for our lexicon incorporation experiments.

We build the systems in Sockeye (Hieber et al., 2017), using a two-layer LSTM network with hidden unit size 512. We use an initial learning rate of 3e-4 both for training the general domain model and adapting to WIPO. We apply the Moses tokenizer (Koehn et al., 2007), lowercasing, and byte-pair encoding (BPE; Sennrich et al., 2016) with a vocabulary size of 30k. BPE is trained on the general domain corpus only, then applied to all data.

### 4.1 Evaluation Metrics

We evaluate lexicon incorporation approaches using two main metrics: BLEU and recall. For each annotated instance of a source-side lexical entry, we can check whether the system output contains the correct aligned target-side translation. Recall is computed as the percentage of the time that the system produces the correct output, averaged over all annotations. Note that this does not guarantee that the words are placed in a sensible location in the sentence, only that they appear.

We also consider training and decoding speed. All of these factors help determine which approaches are best given a particular use case. For example, if a user requires exact fidelity to a lexicon (e.g. branding), they may care the most about recall (while still ensuring that overall translation quality is still acceptable).

### 4.2 Continued Training on HABLex

We perform continued training (CT; Luong and Manning, 2015)—typically used for domain adaptation—on the bilingual lexicon data. With bilingual lexicons approximately two orders of magnitude larger than those used successfully in Kothur et al. (2018), we find that performance drops dramatically with standard CT. To address this problem, we apply EWC (Kirkpatrick et al., 2017), a method for training a neural network to learn a new task without catastrophically forgetting how to perform a previously learned task. EWC has recently been shown to significantly reduce general domain performance loss during domain adaptation in NMT (Thompson et al., 2019); we

[5] We focus on 5-count (and rarer) words because we expect them to be particularly challenging for MT, but given time and resources there is nothing to prevent the application of the annotation protocol to other terms.

[6] For strong initial alignments, GIZA++ (Och and Ney, 2003) is trained on train, development, and test for all data described in section 4 as well as a TED talk corpus (Cettolo et al., 2012).

[7] While we want the dictionary to match the reference, we did not want to train on large phrases from the reference.

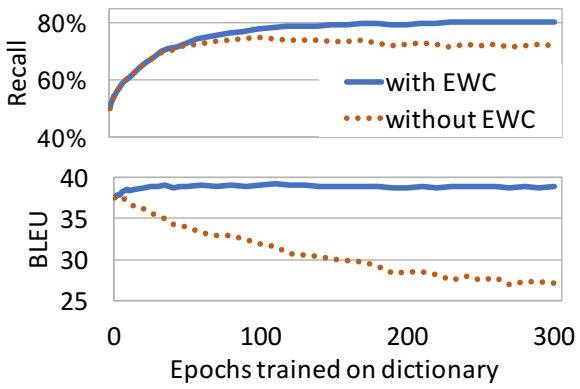


Figure 1: Ru→En recall and BLEU during continued training on bilingual lexicon for the development set.

| | | | CT | | CD | |
|---|---|---|---|---|---|---|
| | | BL | CT | +EWC | Rand. | Oracle |
| Ru | bleu | 37.5 | 27.1 | 38.9 | 40.0 | 40.5 |
| | rec. | 48.2 | 72.1 | 80.2 | 95.9 | 99.9 |
| Ko | bleu | 34.1 | 10.9 | 31.5 | 31.6 | 33.7 |
| | rec. | 55.0 | 47.3 | 69.7 | 87.6 | 99.0 |
| Zh | bleu | 39.9 | 34.8 | 39.0 | 42.2 | 42.5 |
| | rec. | 38.9 | 78.7 | 81.2 | 97.4 | 99.9 |

Table 3: BLEU and recall % (rec.) of baseline (BL), continued training (CT) with and without EWC, and oracle and random constrained decoding (CD) on development set.[8]

apply it to retain the ability to translate full sentences while training on a bilingual lexicon.

We experimented with initial learning rates (0.001, 0.00316, 0.01, 0.0316, 0.1), all using SGD for 300 epochs, and EWC weight decay values (1e-1, 1e-2, 1e-3, 1e-4, 1e-5). We chose a learning rate 0.1 because it allowed recall to (at least approximately) converge, and a weight decay value of 1e-4 because it performed reasonably well on the dev sets for all 3 languages. All results reported are for the final checkpoint.

| | bleu | | recall (%) | |
|---|---|---|---|---|
| | Baseline | Random | Baseline | Random |
| Ru | 37.5 | 40.3 | 46.7 | 96.5 |
| Ko | 34.5 | 32.7 | 54.7 | 89.3 |
| Zh | 39.0 | 40.9 | 36.3 | 96.2 |

Table 4: Baseline and constrained decoding (random) on test set.

### 4.3 Constrained Decoding

We employ *dynamic beam allocation* (DBA; Post and Vilar, 2018) for constrained decoding (CD). At each time step of decoding, DBA groups hypotheses into *banks* based on the number of translated constraints, and a fixed-size beam is dynamically allocated across the banks. While DBA has a time complexity constant in the number of constraints, in practice we find it to be approximately an order of magnitude slower than regular decoding, primarily because Sockeye does not currently support batched DBA.[9]

One limitation of DBA is that it only works on constraints that have only one translation. In this work, we report **oracle** choice (use the right sense for the specific test sentence) as an upper bound on performance and **random** choice (choose a random possible translation) as another baseline. Our dictionary may contain more than one possible translation for a given target word. For example, we observe the following three English translations for the Russian word арматурного: *rebar, reinforced,* and *reinforcement*. These translations are appropriate in different contexts and are not interchangeable, so the oracle always selects the translation that is appropriate for the given sentence (according to its reference translation). The random approach selects one of the translations uniformly at random; if there is only one translation, random is identical to oracle.

### 4.4 Results & Discussion

Table 3 summarizes key exploratory results for the baseline, CT, and CD approaches on the development set. Table 4 shows baseline and random CD benchmarks on the test set, which we otherwise reserve and release for future evaluation. We confine the remainder of our discussion to the experiments we performed on the development set.

EWC noticeably improves BLEU performance as compared to standard CT, while also increasing recall. (See Figure 1 for example performance as the model is trained on the bilingual lexicon and Table 3 for results on all three language pairs.) CD outperforms CT in terms of both BLEU score and recall, at the expense of decoding speed. Random CD nears oracle CD performance because most source-side entries have only one translation. We

[8]The recall of the oracle method would be 100%, if there were no out-of-vocabulary subwords in the bilingual lexicon.

[9]Constrained decoding with DBA takes 3.1 seconds per sentence on average on a Tesla K80 GPU.

would not necessarily expect such high random CD performance if the bilingual lexicon averaged more senses per source entry.

The use case for bilingual lexicon incorporation should also influence user decisions about what approaches to use. If exact translations (e.g. brand names or highly technical translations) are of the utmost importance, a CD approach might be preferred even if it is slower. In the case that a general lexicon has been provided but there is more flexibility in terminology, it may be better to perform CT, or perhaps even a combination of the two.

## 5 Conclusions

Bilingual lexicons are important resources in translation, but it is not clear how to best incorporate them in NMT. To address this challenge, we present the HABLex dataset, a multi-language, reference-derived development and test set that facilitates the evaluation of lexicon incorporation. We compare two baselines, based on incorporation at training time or at decode time, in terms of BLEU, recall, and speed. We also present a novel application of EWC to continued training which addresses lexicon overfitting.

## Acknowledgments

We thank the annotators who worked on building this dataset, as well as the reviewers for their helpful comments. Brian Thompson is supported through the National Defense Science and Engineering Graduate (NDSEG) Fellowship Program.

## References

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2016. Guided open vocabulary image captioning with constrained beam search. *arXiv preprint arXiv:1612.00576*.

Philip Arthur, Graham Neubig, and Satoshi Nakamura. 2016. Incorporating discrete translation lexicons into neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1557–1567, Austin, Texas. Association for Computational Linguistics.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. 2017. Findings of the 2017 conference on machine translation (wmt17). In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 169–214, Copenhagen, Denmark. Association for Computational Linguistics.

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. Wit3: Web inventory of transcribed and translated talks. In *Conference of European Association for Machine Translation*, pages 261–268.

Rajen Chatterjee, Matteo Negri, Marco Turchi, Marcello Federico, Lucia Specia, and Frédéric Blain. 2017. Guiding neural machine translation decoding with external knowledge. In *Proceedings of the Second Conference on Machine Translation*, pages 157–168. Association for Computational Linguistics.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. Data augmentation for low-resource neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 567–573, Vancouver, Canada. Association for Computational Linguistics.

Stephen Grimes. 2010. LDC word aligner, version 1.20.

Stephen Grimes, Katherine Peterson, and Xuansong Li. 2012. Automatic word alignment tools to scale production of manually aligned parallel texts. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*.

Eva Hasler, Adrià de Gispert, Gonzalo Iglesias, and Bill Byrne. 2018. Neural machine translation decoding with terminology constraints. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 506–512. Association for Computational Linguistics.

Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, and Matt Post. 2017. Sockeye: A Toolkit for Neural Machine Translation. *arXiv preprint arXiv:1712.05690*.

Chris Hokamp and Qun Liu. 2017. Lexically constrained decoding for sequence generation using grid beam search. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1535–1546. Association for Computational Linguistics.

Marcin Junczys-Dowmunt, Bruno Pouliquen, and Christophe Mazenc. 2016. Coppa v2. 0: Corpus of parallel patent applications building large parallel corpora with gnu make. In *4th Workshop on Challenges in the Management of Large Corpora Workshop Programme*.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks.

*Proceedings of the national academy of sciences*, 114(13):3521–3526.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, Prague, Czech Republic. Association for Computational Linguistics.

Sachith Sri Ram Kothur, Rebecca Knowles, and Philipp Koehn. 2018. Document-level adaptation for neural machine translation. In *Proceedings of the Second Workshop on Neural Machine Translation and Generation*, Melbourne. Association for Computational Linguistics.

Pierre Lison, Jörg Tiedemann, and Milen Kouylekov. 2018. OpenSubtitles2018: Statistical Rescoring of Sentence Alignments in Large, Noisy Parallel Corpora. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Yang Liu and Maosong Sun. 2015. Contrastive unsupervised word alignment with non-local features. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 2295–2301. AAAI Press.

Minh-Thang Luong and Christopher D Manning. 2015. Stanford neural machine translation systems for spoken language domains. In *Proceedings of the International Workshop on Spoken Language Translation*.

Graham Neubig. 2011. The Kyoto free translation task.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Matt Post and David Vilar. 2018. Fast lexically constrained decoding with dynamic beam allocation for neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1314–1324, New Orleans, Louisiana. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, Berlin, Germany. Association for Computational Linguistics.

Brian Thompson, Jeremy Gwinnup, Huda Khayrallah, Kevin Duh, and Philipp Koehn. 2019. Overcoming catastrophic forgetting during domain adaptation of neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,Volume 2 (Short Papers)*. Association for Computational Linguistics.

Jiajun Zhang and Chengqing Zong. 2016. Bridging neural machine translation and bilingual dictionaries. *CoRR*, abs/1610.07272.

# Handling Syntactic Divergence in Low-resource Machine Translation

**Chunting Zhou, Xuezhe Ma, Junjie Hu, Graham Neubig**
Language Technologies Institute
Carnegie Mellon University
{chuntinz,xuezhem,junjieh,gneubig}@cs.cmu.edu

## Abstract

Despite impressive empirical successes of neural machine translation (NMT) on standard benchmarks, limited parallel data impedes the application of NMT models to many language pairs. Data augmentation methods such as back-translation make it possible to use monolingual data to help alleviate these issues, but back-translation itself fails in extreme low-resource scenarios, especially for syntactically divergent languages. In this paper, we propose a simple yet effective solution, whereby target-language sentences are re-ordered to match the order of the source and used as an additional source of training-time supervision. Experiments with simulated low-resource Japanese-to-English, and real low-resource Uyghur-to-English scenarios find significant improvements over other semi-supervised alternatives[1]

## 1 Introduction

While neural machine translation (NMT; Bahdanau et al. (2015); Vaswani et al. (2017)) now represents the state of the art in the majority of large-scale MT benchmarks (Bojar et al., 2017), it is highly dependent on the availability of copious parallel resources; NMT under-performs previous phrase-based methods when the training data is small (Koehn and Knowles, 2017). Unfortunately, million-sentence parallel corpora are often unavailable for many language pairs. Conversely, monolingual sentences, particularly in English, are often much easier to find, making semi-supervised approaches that can use monolingual data a desirable solution to this problem.[2]

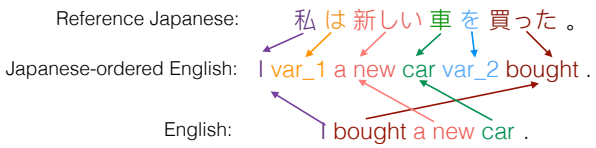


Figure 1: An English sentence re-ordered into Japanese order using the rule-based method of Isozaki et al. (2010b), and its reference Japanese translation.

Semi-supervised approaches for NMT are often based on automatically creating pseudo-parallel sentences through methods such as back-translation (Irvine and Callison-Burch, 2013; Sennrich et al., 2016) or adding an auxiliary auto-encoding task on monolingual data (Cheng et al., 2016; He et al., 2016; Currey et al., 2017). However, both methods have problems with low-resource and syntactically divergent language pairs. Back translation assumes enough data to create a functional NMT system, an unrealistic requirement in low-resource scenarios, while auto-encoding target sentences by definition will not be able to learn source-target word reordering.

This paper proposes a method to create pseudo-parallel sentences for NMT for language pairs with divergent syntactic structures. Prior to NMT, word reordering was a major challenge for statistical machine translation (SMT), and many techniques have emerged over the years to address this challenge (Xia and McCord, 2004; Bisazza and Federico, 2016). Importantly, even simple heuristic reordering methods with a few hand-created rules have been shown to be highly effective in closing syntactic gaps (Collins et al. (2005); Isozaki et al. (2010b); Fig. 1). Because these rules usually function solely in high-resourced languages such as English with high-quality syn-

[1] https://github.com/violet-zct/pytorch-.reorder-nmt

[2] Unsupervised MT (Artetxe et al., 2018a) has achieved success on simulated low-resource scenarios with related languages, but limited success on real low-resource settings and syntactically divergent language pairs (Neubig and Hu, 2018; Guzmán et al., 2019). Hence we focus on semi-supervised methods in this paper.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1388–1394, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

tactic analysis tools, a linguist with rudimentary knowledge of the structure of the target language can create them in short order using these tools.

However, similar pre-ordering methods have not proven useful in NMT (Du and Way, 2017), largely because high-resource scenarios NMT is much more effective at learning reordering than previous SMT methods were (Bentivogli et al., 2016). However, in low-resource scenarios it is less realistic to expect that NMT could learn this reordering from scratch on its own.

Here we ask "how can we efficiently leverage the monolingual target data to improve the performance of the NMT system in low-resource, syntactically divergent language pairs?" We tackle this problem via a simple two-step data augmentation method: (1) we first reorder monolingual target sentences to create *source-ordered target sentences* as shown in Fig. 1, (2) we then replace the words in the reordered sentences with source words using a bilingual dictionary, and add them as the source side of a pseudo-parallel corpus. Experiments demonstrate the effectiveness of our approach on translation from Japanese and Uyghur to English, with a simple, linguistically motivated method of head finalization (HF; Isozaki et al. (2010b)) as our reordering method.

## 2 The Proposed Method

**Training Framework** We assume that there are two types of available resources: a small parallel corpus $\mathcal{P} = \{(s, t)\}$ and a large monolingual target corpus $\mathcal{Q}$. The goal of our method is to create a pseudo-parallel corpus $\hat{\mathcal{Q}} = \{(\hat{s}, t)\}$, where $\hat{s}$ is a pseudo-parallel sentence automatically created in two steps of (1) word reordering, and (2) word-by-word translation.

**Word Reordering** The first step reorders monolingual target sentences $t \in \mathcal{Q}$ into the source order $t_s$. Instead of devising an entirely new word-ordering method, we can simply rely on methods that have already been widely studied and proven useful in SMT (Bisazza and Federico, 2016). Reordering can be done either using rules based on linguistic knowledge (Isozaki et al., 2010b; Collins et al., 2005) or learning from aligned parallel data (Xia and McCord, 2004; Habash, 2007), and in principle our pseudo-corpus creation paradigm is compatible with any of these methods.

Specifically, in this work we utilize rule-based methods, as our goal is to improve translation of low-resource languages, where large quantities of high-quality parallel data do not exist and we posit that current data-driven reordering methods are unlikely to function well. Examples of rule-based methods include those to reorder English into German (Navratil et al., 2012), Arabic (Badr et al., 2009), or Japanese (Isozaki et al., 2010b). In experiments we use Isozaki et al. (2010b)'s method of reordering SVO languages (e.g. English) into the order of SOV languages (e.g. Japanese) by simply (1) applying a syntactic parser to English (Tsuruoka et al., 2004), (2) identifying the head constituent of each phrase and moving it to the end of the phrase, and (3) inserting special tokens after subjects and objects of predicates to mimic Japanese case markers.

**Word-by-word Translation** To generate data for training MT models, we next perform word-by-word translation of $t_s$ into pseudo-source sentence $\hat{s}$ using a bilingual dictionary (Xie et al., 2018).[3] There are many ways we can obtain this dictionary: even for many low-resource languages with a paucity of bilingual text, we can obtain manually-curated lexicons with reasonable coverage, or run unsupervised word alignment on whatever parallel data we have available. In addition, we can induce word translations for more words in target language using methods for bilingual lexicon induction over pre-trained word embeddings (e.g. Grave et al. (2018)).

## 3 Experiments

We evaluate our method on two language pairs: Japanese-to-English (`ja-en`) and Uyghur-to-English (`ug-en`). Japanese and Uyghur are phylogenetically distant languages, but they share similar SOV syntactic structure, which is greatly divergent from English SVO structure.

### 3.1 Experimental Setup

For both language pairs, we use an attention-based encoder-decoder NMT model with a one-layer bidirectional LSTM as the encoder and one-layer uni-directional LSTM as the decoder.[4] Embeddings and LSTM states were set to 300 and

[3] We also performed extensive preliminary experiments that learned bilingual word embeddings a-priori and froze them when training the NMT model, or continued to align the bilingual word embedding space during NMT training, but the word-by-word translation approach worked best.

[4] We experimented with small Transformers (Vaswani et al., 2017) but they under-performed LSTM-based models.

256 dimensions respectively. Target word embeddings are shared with the softmax weight matrix in the decoder. As noted above, we use HF (Isozaki et al., 2010b) as our re-ordering rule. HF was designed for transforming English into Japanese order, but we use it *as-is* for the Uyghur-English pair as well to demonstrate that simple, linguistically motivated rules can generalize across pairs with similar syntax with little or no modification. Further details regarding the experimental settings are in the supplementary material.

**Simulated Japanese to English Experiments** We first evaluate on a simulated low-resource `ja-en` translation task using the ASPEC dataset (Nakazawa et al., 2016). We randomly select 400k `ja-en` parallel sentence pairs to use as our full training data. We then randomly sub-sample low-resource datasets of 3k, 6k, 10k, and 20k parallel sentences, and use the remainder of the 400k English sentences as monolingual data. We duplicate the number of parallel sentences by 5 times in the training data augmented with the reordered pairs. For settings with supervised parallel sentences of 3k, 6k, 10k and 20k, we set the maximum vocabulary size of both Japanese and English to be 10k, 10k, 15k and 20k respectively.

To automatically learn a high-precision dictionary on the small amount of parallel data we have available for training, we use GIZA++ (Och and Ney, 2003) to learn alignments in both directions then take the intersection of alignments. We then learn the bilingual word embeddings with DeMa-BWE (Zhou et al., 2019), an unsupervised method that has shown strong results on syntactically divergent language pairs. We give the more reliable alignments extracted from GIZA++ high priority by querying the alignment dictionary first, then follow by querying the embedding-induced dictionary. When an English word is not within any vocabulary, we output the English word as-is into the pseudo-source sentence.

**Real Uyghur to English Experiments** We also consider the harder case of Uyghur, a truly low-resource language. We create test and validation sets using the test data from the DARPA LORELEI corpus (Christianson et al., 2018) which contains 2,275 sentence pairs (after filtering out noisy ones) related to incidents that happened in the Uyghur area. We hold out 300 pairs as the validation data and use the rest as the test set. The LORELEI language pack also contains the bilingual lexicons between Uyghur and English, and thousands of in-domain English sentences. We also use a large monolingual English corpus containing sentences related to various incidents occurring all over the world collected from ReliefWeb.[5] To sub-select a relevant subset of this corpus, we use the cross-entropy filtering (Moore and Lewis, 2010) to select 400k that are most like the in-domain English data.

For parallel data, like many low-resource languages, we only have access to data from the Bible[6] and Wikipedia language links (the total number of parallel Uyghur-English Wikipedia titles is 3,088), but no other in-domain parallel data. We run GIZA++ on this parallel data to obtain an alignment dictionary. We learn the bilingual word embeddings via the supervised Geometric approach (Jawanpuria et al., 2019) on FastText (Grave et al., 2018) pre-trained Uyghur and English monolingual embeddings.

| Model | 3k | 6k | 10k | 20k | 400k | ug |
|---|---|---|---|---|---|---|
| sup | 2.17 | 7.86 | 11.67 | 15.98 | 26.56 | 0.58 |
| sup-SMT | 6.36 | 8.70 | 10.68 | 12.11 | 18.62 | 1.46 |
| back | 2.27 | 5.40 | 13.50 | 16.05 | – | 0.42 |
| back-SMT | 8.46 | 10.61 | 12.05 | 13.68 | – | 1.37 |
| No-Reorder | 6.46 | 9.73 | 12.57 | 15.56 | – | 3.24 |
| Reorder | **9.94** | **12.42** | **14.98** | **17.58** | – | **4.17** |

Table 1: BLEU of our approach (Reorder) with different amount of parallel sentences of `ja-en` and `ug-en` translation. Baselines are supervised learning from NMT and SMT (sup and sup-SMT), supervised learning with back translation from NMT and SMT (back and back-SMT) and data augmentation with translated original English sentences (No-Reorder).

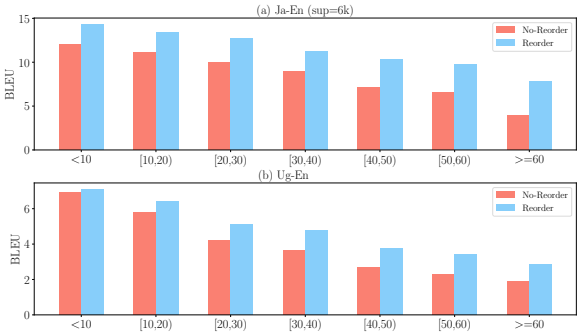


Figure 2: Comparison of BLEU score w.r.t different sentence lengths.

[5] https://reliefweb.int
[6] https://bible.is

| | |
|---|---|
| **source** | しかし， 回転 速度 が 大き すぎ る と， 逆向き の 変形 が 生じ る |
| **reference** | the too high rotation speed produces the reverse deformation |
| **supervised** | however , the deformation of <unk> and the deformation of <unk> is caused by the dc rate |
| **ours** | however , the deformation of <unk> is generated when the rotation rate is large |
| **source** | 2 ـ ئاينىنگ 12 ـ ك ، ڤاقتى 3 : 29 پىنۇتتا 3 . 3 ـال يېر تېڤرىگېن ، يېر تېڤرېسه ځهونگقۇرلىقى 8000 پ . |
| **reference** | a 3.3 magnitude earthquake with the depth of 8000 meters hit feb 12 at 3:29 urumqi time |
| **supervised** | 2 , on february 12 - 12 of darkness , urumqi time hit a 3.3 earthquake , the earthquake hit . |
| **ours** | 2 - on february 12 - on , urumqi time 3:29minus 3.3 magnitude earthquake hit , the earthquake under depth of 8000 meters . |

Table 2: Translation examples on `ja-en` (reorder with 6000 supervised pairs) and `ug-en` (reorder) from our model and the supervised counterpart.

### 3.2 Results and Comparison

**Baselines** In Tab. 1, we compare our models with baselines including regular supervised training (sup) and back-translation (Sennrich et al., 2016) (back).[7] To demonstrate the effectiveness of the reordering, we also compare our method against a copy-based data-augmentation method (No-reorder) where the original English sentences $t \in \mathcal{Q}$ rather than the reordered ones $t_s$ are translated via the bilingual lexicon.[8] For each of the above settings, we also experimented with a phrased-based statistical machine translation (SMT) system (Dyer et al., 2010). In Tab. 1, we only show the results with supervised data and back-translation for SMT, since we observed that the data augmentation method performs poorly with SMT (complete results are presented in the Appendix).

**Main Results** In Tab. 1, we observe consistent improvements on both `ja-en` and `ug-en` translation tasks against other baseline methods. First, comparing our results with the NMT models trained using the same amount of parallel data, our word reordering-based semi-supervised models consistently outperform standard NMT models by a large margin. In the case that we have no access to in-domain parallel data at all, our method can still achieve some success in `ug-en` translation. Second, comparing our Reorder method with the No-Reorder one, reordering English sentences into the source language order consistently brings large performance gains, which demonstrates the importance of reordering. These results are notable given previous reports that explicit reordering is not beneficial for NMT (Du and Way, 2017). Third, for `ja-en` translation, when gradually decreasing the amount of parallel data, the improvements of our model over the supervised NMT models become more significant, demonstrating the effectiveness of our approach in low-resource settings. Fourth, back-translation is not very beneficial or even harmful, likely because the back-translation system trained on limited supervised data can not provide high-quality translations to train the model. Finally, we also notice that although SMT performs better than NMT with less supervised training data (3k, 6k supervised data and Uyghur), the performance gain is not as remarkable as NMT when the amount of supervised data increases. Moreover, in the case of less supervised data, our data augmentation method with reordering still outperforms SMT.

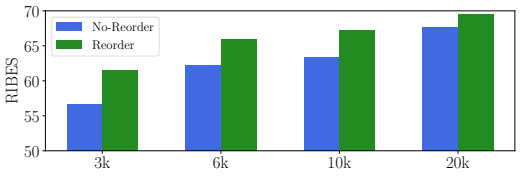


Figure 3: Comparison of RIBES score on Ja-En translation with different amounts of supervised data.

We give two examples of the translation outputs of our model and a supervised NMT model for `ug-en` and `ja-en` (trained with 6k supervised pairs) in Tab. 2. In the first example from `ja-en`, our model is able to output terminology such as "rotation rate" thanks to the enlarged vocabulary while the supervised model can not. In the example from `ug-en`, our model can produce a more fluent sentence with better information coverage.

[7] We also trained unsupervised NMT (Artetxe et al., 2018b) on 400k Japanese and English sentences from the ASPEC corpus or monolingual Uyghur and English corpus provided in the LORELEI data package, and it achieved BLEU of 0.6 and 0.0 respectively, corroborating previous results noting the difficulty of unsupervised NMT for low-resource scenarios (Neubig and Hu, 2018; Guzmán et al., 2019)

[8] This is similar to Currey et al. (2017) who simply copy target-language sentences to the source. But in our experiments, the source and target languages do not share the alphabet and thus we found translation necessary. This also increases consistency with our "Reorder" experiments.

**Analysis** To investigate the effects of reordering, we compare our method with "No-Reorder" described in 3.2. First, we bucket the test data by sentence length and compute the BLEU score accordingly. We present the comparison results in Fig. 2, from which we observe that "Reorder" outperforms "No-Reorder" consistently under different sentence length buckets, and the improvement is larger when the sentence length is longer.

Second, we also evaluate the model outputs on the test data with RIBES (Isozaki et al., 2010a) which is an automatic evaluation metric of translation quality designed for distant languages that is especially sensitive to word order. From Fig. 3, we can see that "Reorder" consistently outperforms "No-Reorder" on `ja-en` translation especially when the amount of supervised data decreases. This suggests that with reordered pairs as the augmented training data, the model is able to output more syntactically correct sentences.

## 4 Conclusion

This paper proposed a simple yet effective semi-supervised learning framework for low-resource machine translation that artificially creates source-ordered target sentences for data-augmentation. Experimental results on `ja-en` and `ug-en` translations show that our approach achieves significant improvements over baseline systems, demonstrating the effectiveness of the proposed approach on divergent language pairs.

## Acknowledgments

This work is sponsored by Defense Advanced Research Projects Agency Information Innovation Office (I2O), Program: Low Resource Languages for Emergent Incidents (LORELEI), issued by DARPA/I2O under Contract No. HR0011-15-C-0114. The authors would like to thank Shruti Rijhwani and Hiroaki Hayashi for their help when preparing the data sets.

## References

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018a. Unsupervised statistical machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium. Association for Computational Linguistics.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018b. Unsupervised neural machine translation. *In International Conference on Learning Representations (ICLR)*.

Ibrahim Badr, Rabih Zbib, and James Glass. 2009. Syntactic phrase reordering for English-to-Arabic statistical machine translation. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 86–93. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *In International Conference on Learning Representations (ICLR)*.

Luisa Bentivogli, Arianna Bisazza, Mauro Cettolo, and Marcello Federico. 2016. Neural versus phrase-based machine translation quality: a case study. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 257–267, Austin, Texas. Association for Computational Linguistics.

Arianna Bisazza and Marcello Federico. 2016. A survey of word reordering in statistical machine translation: Computational models and language phenomena. *Computational linguistics*, 42(2):163–205.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, et al. 2017. Findings of the 2017 conference on machine translation (WMT17). In *Proceedings of the Second Conference on Machine Translation*, pages 169–214.

Yong Cheng, Wei Xu, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Semi-supervised learning for neural machine translation. *The 54th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Caitlin Christianson, Jason Duncan, and Boyan Onyshkevych. 2018. Overview of the DARPA LORELEI Program. *Machine Translation*, 32(1-2):3–9.

Michael Collins, Philipp Koehn, and Ivona Kučerová. 2005. Clause restructuring for statistical machine translation. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 531–540. Association for Computational Linguistics.

Anna Currey, Antonio Valerio Miceli Barone, and Kenneth Heafield. 2017. Copied monolingual data improves low-resource neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 148–156.

Jinhua Du and Andy Way. 2017. Pre-Reordering for Neural Machine Translation: Helpful or Harmful? *The Prague Bulletin of Mathematical Linguistics*, 108(1):171–182.

Chris Dyer, Jonathan Weese, Hendra Setiawan, Adam Lopez, Ferhan Ture, Vladimir Eidelman, Juri Ganitkevitch, Phil Blunsom, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of the ACL 2010 System Demonstrations*, pages 7–12. Association for Computational Linguistics.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*.

Francisco Guzmán, Peng-Jen Chen, Myle Ott, Juan Pino, Guillaume Lample, Philipp Koehn, Vishrav Chaudhary, and Marc'Aurelio Ranzato. 2019. Two New Evaluation Datasets for Low-Resource Machine Translation: Nepali-English and Sinhala-English. *arXiv preprint arXiv:1902.01382*.

Nizar Habash. 2007. Syntactic preprocessing for statistical machine translation. *Proceedings of the 11th MT Summit*, 10.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tieyan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In *Advances in Neural Information Processing Systems (NIPs)*, pages 820–828.

Ann Irvine and Chris Callison-Burch. 2013. Combining bilingual and comparable corpora for low resource machine translation. In *Proceedings of the eighth workshop on statistical machine translation*, pages 262–270.

Hideki Isozaki, Tsutomu Hirao, Kevin Duh, Katsuhito Sudoh, and Hajime Tsukada. 2010a. Automatic evaluation of translation quality for distant language pairs. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 944–952. Association for Computational Linguistics.

Hideki Isozaki, Katsuhito Sudoh, Hajime Tsukada, and Kevin Duh. 2010b. Head finalization: A simple reordering rule for SOV languages. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 244–251. Association for Computational Linguistics.

Pratik Jawanpuria, Arjun Balgovind, Anoop Kunchukuttan, and Bamdev Mishra. 2019. Learning multilingual word embeddings in latent metric space: a geometric approach. *Transactions of the Association for Computational Linguistics (TACL)*.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39, Vancouver. Association for Computational Linguistics.

Robert C Moore and William Lewis. 2010. Intelligent selection of language model training data. In *Proceedings of the ACL 2010 conference short papers*, pages 220–224. Association for Computational Linguistics.

Toshiaki Nakazawa, Manabu Yaguchi, Kiyotaka Uchimoto, Masao Utiyama, Eiichiro Sumita, Sadao Kurohashi, and Hitoshi Isahara. 2016. ASPEC: Asian Scientific Paper Excerpt Corpus. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 2204–2208, Portorož, Slovenia.

Jiri Navratil, Karthik Visweswariah, and Ananthakrishnan Ramanathan. 2012. A comparison of syntactic reordering methods for English-German machine translation. *Proceedings of COLING 2012*, pages 2043–2058.

Graham Neubig and Junjie Hu. 2018. Rapid adaptation of neural machine translation to new languages. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 875–880. Association for Computational Linguistics.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving Neural Machine Translation Models with Monolingual Data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 86–96.

Yoshimasa Tsuruoka, Yusuke Miyao, and Jun'ichi Tsujii. 2004. Towards efficient probabilistic HPSG parsing: integrating semantic and syntactic preference to guide the parsing. In *Proceedings of the IJCNLP-04 Workshop on Beyond Shallow Analyses*. ACL.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Fei Xia and Michael McCord. 2004. Improving a Statistical MT System with Automatically Learned Rewrite Patterns . In *Proceedings of Coling 2004*, pages 508–514, Geneva, Switzerland. COLING.

Jiateng Xie, Zhilin Yang, Graham Neubig, Noah A Smith, and Jaime Carbonell. 2018. Neural cross-lingual named entity recognition with minimal resources. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 369–379.

Chunting Zhou, Xuezhe Ma, Di Wang, and Graham Neubig. 2019. Density Matching for Bilingual Word Embedding. In *Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*, Minneapolis, USA.

# Speculative Beam Search for Simultaneous Translation

**Renjie Zheng** [2,*] **Mingbo Ma** [1,*] **Baigong Zheng** [1] **Liang Huang** [1,2]
[1]Baidu Research, Sunnyvale, CA, USA
[2]Oregon State University, Corvallis, OR, USA
zrenj11@gmail.com
{mingboma, baigongzheng, lianghuang}@baidu.com

## Abstract

Beam search is universally used in full-sentence translation but its application to simultaneous translation remains non-trivial, where output words are committed on the fly. In particular, the recently proposed wait-$k$ policy (Ma et al., 2019a) is a simple and effective method that (after an initial wait) commits one output word on receiving each input word, making beam search seemingly impossible. To address this challenge, we propose a speculative beam search algorithm that hallucinates several steps into the future in order to reach a more accurate decision, implicitly benefiting from a target language model. This makes beam search applicable for the first time to the generation of a single word in each step. Experiments over diverse language pairs show large improvements over previous work.

## 1 Introduction

Beam search has been widely used in neural text generation such as machine translation (Sutskever et al., 2014; Bahdanau et al., 2014), summarization (Rush et al., 2015; Ranzato et al., 2016), and image captioning (Vinyals et al., 2015; Xu et al., 2015). It often leads to substantial improvement over greedy search and becomes an essential component in almost all text generation systems.

However, beam search is easy for the above tasks because they are all *full-sequence* problems, where the whole input sequence is available at the beginning and the output sequence only needs to be revealed in full at the end. By contrast, in language and speech processing, there are many *incremental processing* tasks with *simultaneity requirements*, where the output needs to be revealed to the user incrementally without revision (word by word, or in chunks) and the input is also being received incrementally. Two most salient examples are streaming speech recognition (Chiu et al., 2018), widely used in speech input and dialog systems (such as Siri), and simultaneous translation (Bangalore et al., 2012; Oda et al., 2015; Grissom II et al., 2014; Jaitly et al., 2016), widely used in international conferences and negotiations. In these tasks, the use of full-sentence beam search becomes seemingly impossible as output words need to be committed on the fly.

How to adapt beam search for such incremental tasks in order to improve their generation quality? We propose a general technique of *speculative beam search* (SBS), and apply it to simultaneous translation. At a very high level, to generate a single word, instead of simply choosing the highest-scoring one (as in greedy search), we further speculate $w$ steps into the future, and use the ranking at step $w+1$ to reach a more informed decision for step 1 (the current step); this method implicitly benefits from a target language model, alleviating the label bias problem in neural generation (Murray and Chiang, 2018; Ma et al., 2019b).

We apply this algorithm to two representative approaches to simultaneous translation: the fixed policy method (Ma et al., 2019a) and the adaptive policy method (Gu et al., 2017). In both cases, we show that SBS improves translation quality while maintaining latency (i.e., simultaneity).

## 2 Preliminaries

We first review standard full-sentence NMT and beam search to set up the notations, and then review different approaches to simultaneous MT.

### 2.1 Full Sentence NMT and Beam Search

The encoder processes the input sequence $\boldsymbol{x} = (x_1, ..., x_n)$, where $x_i \in \mathbb{R}^d$ represents an input token as a $d$ dimensional vector, and produces a new list of hidden states $\boldsymbol{h} = f(\boldsymbol{x}) = (h_1, ..., h_n)$

[*] These authors contributed equally.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1395–1402, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

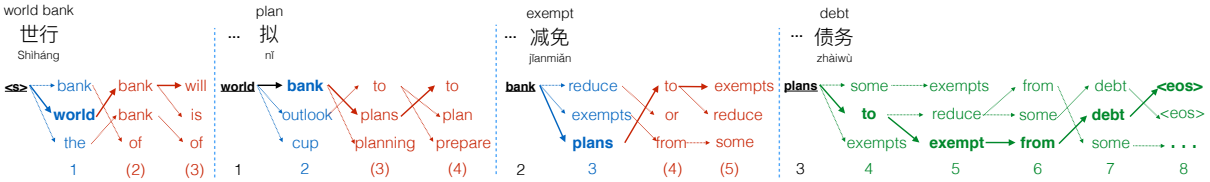


Figure 1: Wait-1 policy example to illustrate the procedure of SBS. The top Chinese words are the source side inputs which are incrementally revealed to the encoder. Gloss is annotated above Chinese word and Pinyin is underneath. There are two extra steps (speculative window) are taken (red part) beyond greedy. When source reaches the last word "债务" (debt), the decoder gets into tail and performs conventional beam search (in green).

to represent $\boldsymbol{x}$. The encoding function $f$ can be RNN, CNN or Transformer.

On the other hand, the (greedy) decoder selects the highest-scoring word $y_t$ given source representation $\boldsymbol{h}$ and previously generated target tokens, $\boldsymbol{y}_{<t} = (y_1, ..., y_{t-1})$. The greedy search continues until it emits <eos>, and the final hypothesis $\boldsymbol{y} = (y_1, ..., y_t)$ with $y_t =$ <eos>

$$p(\boldsymbol{y} \mid \boldsymbol{x}) = \prod_{t=1}^{|\boldsymbol{y}|} p(y_t \mid \boldsymbol{x}, \boldsymbol{y}_{<t}) \quad (1)$$

As greedy search only explores one single path among exponential many alternatives, beam search is used to improve the search. At each step $t$, it maintains a beam $B_t$ of size $b$, which is an ordered list of ⟨hypothesis, probability⟩ pairs; for example $B_0 = [\langle \text{<s>}, 1 \rangle]$. We then define one-step transition from the previous beam to the next as

$$\text{next}_1^b(B) = \text{top}^b\{\langle \boldsymbol{y} \circ v,\ s \cdot p(v|\boldsymbol{x}, \boldsymbol{y}) \rangle \mid \langle \boldsymbol{y}, s \rangle \in B\}$$

where $\text{top}^b(\cdot)$ returns the top-scoring $b$ pairs, and $\circ$ is the string concatenation operator. Now $B_t = \text{next}_1^b(B_{t-1})$. As a shorthand, we also define the multi-step beam search function recursively:

$$\text{next}_i^b(B) = \text{next}_1^b(\text{next}_{i-1}^b(B)) \quad (2)$$

Full-sentence beam search (over a maximum of $T$ steps) yields the best hypothesis $\boldsymbol{y}^*$ with score $s^*$ (see Huang et al. (2017) for stopping criteria):

$$\langle \boldsymbol{y}^*, s^* \rangle = \text{top}^1(\text{next}_T^b([\langle \text{<s>}, 1 \rangle])) \quad (3)$$

### 2.2 Simultaneous MT: Policies and Models

There are two main categories of policies in neural simultaneous translation decoding (Tab. 1):

1. The first method is to use a fixed-latency policy, such as the wait-$k$ policy (Ma et al., 2019a). Such a method would, after an initial wait of $k$ source words, commit one target word on receiving each new source word. When the source sentence ends, the decoder can do a *tail beam search* on the remaining target words, but beam search is seemingly impossible before the source sentence ends.

| model / policy | *sequence-to-sequence* (full-sentence model) | *prefix-to-prefix* (simultaneous model) |
|---|---|---|
| *fixed-latency* | test-time wait-$k$ (Dalvi et al., 2018; Ma et al., 2019a) | wait-$k$ (Ma et al., 2019a) |
| *adaptive* | RL (Gu et al., 2017)<br>Supervised Learning (Zheng et al., 2019a) | MILk (Arivazhagan et al., 2019)<br>Imitation Learning (Zheng et al., 2019b) |

Table 1: Recent advances in simultaneous translation.

2. The second method learns an adaptive policy which uses either supervised (Zheng et al., 2019a) or reinforcement learning (Grissom II et al., 2014; Gu et al., 2017) to decide whether to READ (the next source word) or WRITE (the next target word) . Here the decoder can commit a chunk of *multiple words* for a series of consecutive WRITEs.

In terms of modeling (which is orthogonal to decoding policies), we can also divide most simultaneous translatoin efforts into two camps:

1. Use the standard full-sentence translation model trained by classical seq-to-seq (Dalvi et al., 2018; Gu et al., 2017; Zheng et al., 2019a). For example, the "test-time wait-$k$" scheme (Ma et al., 2019a) uses the full-sentence translation model and performs wait-$k$ decoding at test time. However, the obvious training-testing mismatch in this scheme usually leads to inferior quality.

2. Use a genuinely simultaneous model trained by the recently proposed prefix-to-prefix framework (Ma et al., 2019a; Arivazhagan et al., 2019; Zheng et al., 2019b). There is no training-testing mismatch in this new scheme, with the cost of slower training.

## 3 Speculative Beam Search

We first present our speculative beam search on the fixed-latency wait-$k$ policy (generating a single word per step), and then adapt it to the adaptive policies (generating multiple words per step).

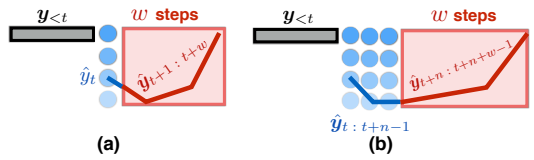


Figure 2: Illustration of SBS: (a) wait-$k$ policy (Eqs. 4–5); (b) adaptive policy (Eqs. 6–7). Speculations in red.

### 3.1 Single-Step SBS

The wait-$k$ policy conducts translation concurrently with the source input, committing output words one by one while the source sentence is still growing. In this case, conventional beam search is clearly inapplicable.

We propose to perform speculative beam search at each step by hallucinating $w$ more steps into the future, and use the ranking after these $w+1$ steps to make a more informed decision for the current step. More formally, at step $t$, we generate $y_t$ based on already committed prefix $\boldsymbol{y}_{<t}$:

$$\langle \hat{\boldsymbol{y}}, s_t \rangle = \text{top}^1\big(\text{next}^b_{1+w}([\langle \boldsymbol{y}_{<t}, 1\rangle])\big) \quad (4)$$

$$\boldsymbol{y}_{\leq t} = \boldsymbol{y}_{<t} \circ \hat{y}_t \quad (5)$$

where $\hat{\boldsymbol{y}} = \boldsymbol{y}_{<t} \circ \hat{y}_t \circ \hat{\boldsymbol{y}}_{t+1:t+w}$ has three parts, with the last one being a speculation of $w$ steps (see Fig. 2). We use $\text{next}^b_{1+w}(\cdot)$ to speculate $w$ steps. The candidate $\hat{y}_t$ is selected based on the accumulative model score $w$ steps later. Then we commit $\hat{y}_t$ and move on to step $t+1$.

In the running example in Fig. 1, we have $w=2$ and $b=3$. In the greedy mode, after the wait-1 policy receives the first source word, "世行" (world bank), the basic wait-1 model commits "bank" which has the highest score. In SBS, we perform a beam search for $1+w=3$ steps with the two speculative steps marked in red. After 3 steps, the path "world bank will" becomes the top candidate, thus we choose to commit "world" instead of "bank" and restart a new speculative beam search with "world" when we receive a new source word, "拟"(plan to); the speculative part from the previous step (in red) is removed.

### 3.2 Chunk-based SBS

The RL-based adaptive policy system (Gu et al., 2017) can commit a chunk of multiple words whenever there is a series of consecutive WRITEs, and conventional beam search can be applied on each chunk to improve the search quality within that chunk, which is already used in that work.

However, on top of the obvious per-chunk beam search, we can still apply SBS to further speculate $w$ steps after the chunk. For a chunk of length $n$ starting at position $t$, we adapt SBS as:

$$\langle \hat{\boldsymbol{y}}, s_t \rangle = \text{top}^1\big(\text{next}^b_{n+w}([\langle \boldsymbol{y}_{<t}, 1\rangle])\big) \quad (6)$$

$$\boldsymbol{y}_{\leq t+n-1} = \boldsymbol{y}_{<t} \circ \hat{\boldsymbol{y}}_{t:t+n-1} \quad (7)$$

Here $\text{next}^b_{n+w}(\cdot)$ does a beam search of $n+w$ steps, with the last $w$ steps speculated. Similarly,

$$\hat{\boldsymbol{y}} = \boldsymbol{y}_{<t} \circ \hat{\boldsymbol{y}}_{t:t+n-1} \circ \hat{\boldsymbol{y}}_{t+n:t+n+w-1}$$

has three parts, with the last being a speculation of $w$ steps, and the middle one being the chunk of $n$ steps returned and committed (see Fig. 2).

## 4 Experiments

### 4.1 Datasets and Latency Metrics

We evaluate our work on Chinese↔English simultaneous translation tasks. For the training data, we use the NIST corpus for Chinese↔English (2M sentence pairs). We first apply BPE (Sennrich et al., 2015) on all texts in order to reduce the vocabulary sizes. For Chinese↔English evaluation, we use NIST 2006 and NIST 2008 as our dev and test sets with 4 English references. For English→Chinese, we use the second among the four English references as the source text.

We re-implement wait-$k$ model (Ma et al., 2019a), test-time wait-$k$ model (Dalvi et al., 2018) and adaptive policy (Gu et al., 2017) based on PyTorch-based OpenNMT (Klein et al., 2017). To reach state-of-the-art performance, we use Transformer based wait-$k$ model and also use Transformer based pre-trained full sentence model for learning adaptive policy. The architecture of Transformer is the same as the base model from the original paper (Vaswani et al., 2017). We use Average Lagging (AL) (Ma et al., 2019a) as the latency metrics. AL measures the number of words delay for translating a given source sentence.

| $b$ \ $w$ | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 1 | 34.57 | - | - | - | - | - |
| 3 | - | +1.3 | +1.8 | +1.2 | +2.0 | +1.7 |
| 5 | - | +1.6 | +1.9 | +1.3 | +1.5 | +1.3 |
| 7 | - | +1.5 | +2.0 | +1.0 | +1.6 | +1.4 |
| 10 | - | +1.4 | +2.2 | +1.4 | +1.5 | +1.7 |

Table 2: Zh→En wait-1 model BLEU improvement of SBS against greedy search ($b=1$, $w=0$) on dev-set. When $w \geq 5$ the performance of SBS becomes stable.

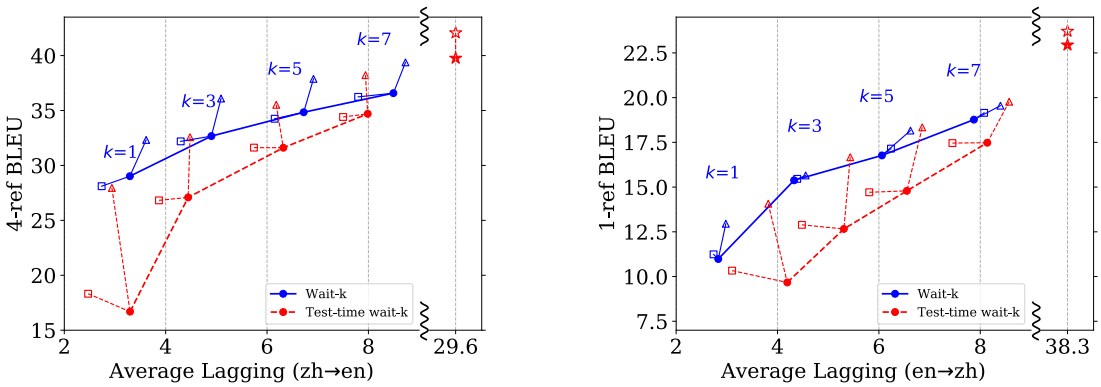


Figure 3: BLEU against AL using wait-$k$ model. □ □: conventional beam search only in target tail (when source finishes). △ △: speculative beam search. ★☆:full-sentence baseline (greedy and beam-search).

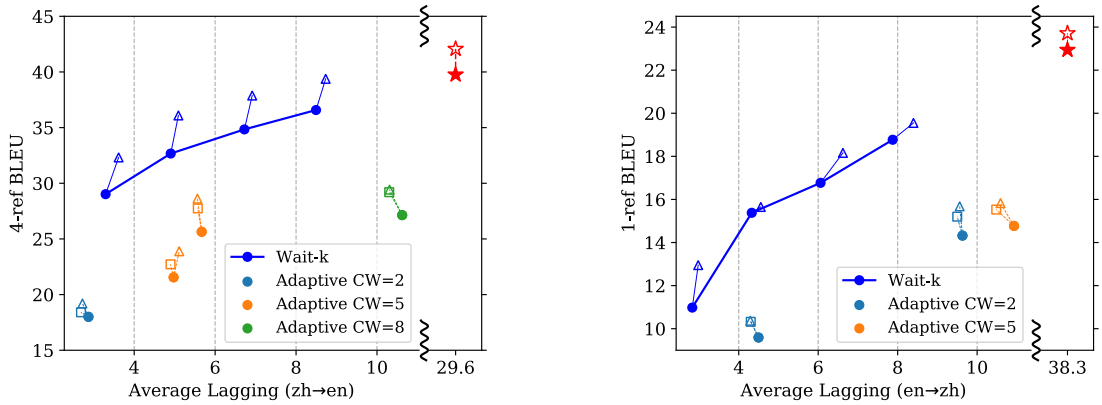


Figure 4: BLEU against AL using adaptive policy (compared with wait-$k$ models) with different beam search methods. □ □ □: conventional beam search in chunk of consecutive write (Gu et al., 2017). △ △ △: speculative beam search. ★☆:full-sentence baseline (greedy and beam-search).

| | | *shìháng* 世行 | *nǐ* 拟 | *jiǎnmiǎn* 减免 | *zùi* 最 | *qióng* 穷 | *gúojiā* 国家 | *zhàiwù* 债务 | |
|---|---|---|---|---|---|---|---|---|---|
| Gloss | | world bank | plan to | remit & reduce | most | poor | country | debt | |
| $k$=1† | Greedy | | world bank | | to | reduce | poverty | - | stricken countries |
| | SBS | | world bank | | to | exemp- | t | po- | or- est countries from debt |
| $k$=1‡ | Greedy | | world bank | | to | reduce | or | exemp- | t debt of po- or- est countries |
| | SBS | | world bank | | inten- | ds | to | reduce | or exemp- t debt of po- or- est countries |
| $k$=∞* | | | | | | | | | world bank plans to remit and reduce debts of po- or- est countries |

Figure 5: Chinese-to-English example on dev set. †: test-time wait-$k$; ‡: wait-$k$. *: full-sentence beam search.

### 4.2 Performance on Wait-$k$ Policy

We perform experiments on validation set using speculative beam search (SBS) with beam sizes $b \in \{3, 5, 7, 10\}$ and speculative window sizes $w \in \{1, 2, 3, 4, 5\}$. Table 2 shows the BLEU score of different $b$ and $w$ over wait-1 model. Compared with greedy decoding, SBS improves at least 1.0 BLEU score in all cases and achieves best performance by $b = 10, w = 2$. We search the best $b$ and $w$ for each model on dev-set and apply them on test-set in the following experiments.

Fig. 3 shows the performance of conventional greedy decoding, trivial tail beam search (only after source sentence is finished) and SBS on test set on Chinese↔English tasks. SBS largely boost test-time wait-$k$ models with slightly worse latency (especially in English→Chinese because they tend to generate longer sentences). Wait-$k$ models also benefit from speculation (especially in Chinese→English).

Fig. 5 shows a running example of greedy and SBS output of both wait-$k$ and test-time wait-$k$ models. SBS on test-time wait-$k$ generates much better outputs than the greedy search, which misses some essential information. Wait-$k$ models with speculation correctly translates "拟" into "intends to" instead of "to" in greedy output.

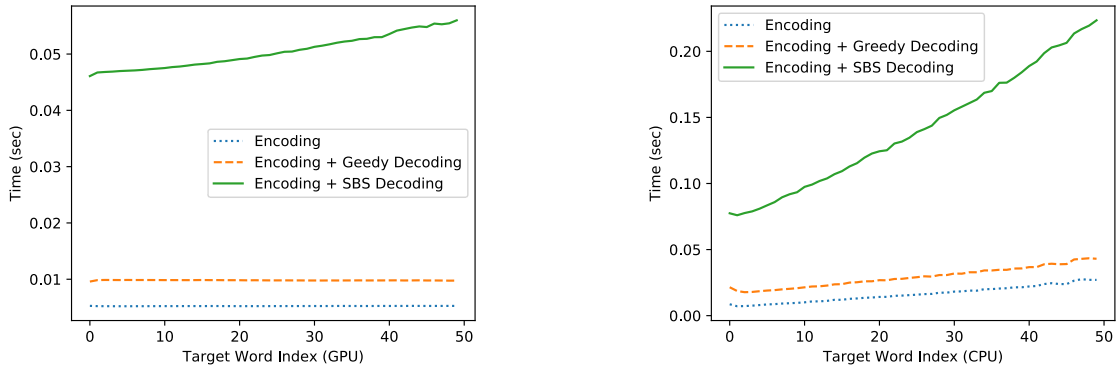


Figure 6: Average time of words with different indices ($\leq 50$) on zh→en wait-3 model. Results in left figure are measured on GPU while results in right figure results are measured on CPU. The SBS results adopt $w = 2, b = 5$.

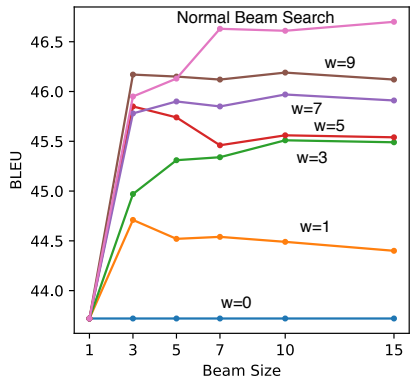


Figure 7: BLEU of SBS over full sentence translation with different window sizes $w$ and beam sizes $b$. Window size $w = 0$ stands for greedy decoding. The top line stands for results using normal beam search.

### 4.3 Performance on Adaptive Policy

Fig. 4 shows the performance of proposed SBS on adaptive policies. We train adaptive policies using the combination of Consecutive Wait (CW $\in \{2, 5, 8\}$ (Gu et al., 2017)) and partial-BLEU as reward in reinforcement learning. We vary beam size $b \in \{5, 10\}$ in both chunk-based beam search (Gu et al., 2017) and our SBS with speculative window size $w \in \{2, 4\}$. Our proposed beam search achieves better results in most cases.

### 4.4 Running Time Analysis

Fig. 6 shows the average time for generating words with different target word indices on a GeForce GTX TITAN-X GPU and an Intel Core i7 2.8 GHz CPU. According to Ma et al. (2019a), wait-$k$ models use bi-directional Transformer as the encoder, thus the time complexity of incrementally encoding one more source word is $O(m^2)$ where $m$ is the source sentence length. This is the reason why it takes more time to encode words with larger index especially using CPU. It is generally accepted that Mandarin speech is about 120–150 syllables per minute, and in our corpus each token (after BPE) has on average 1.5 Chinese syllables (which is 1.5 characters since each Chinese character is monosyllabic), thus in the simultaneous Chinese-to-English speech-to-text translation scenario, the decoder receives a source token every 0.6–0.75 seconds which is much slower than our decoding speed (less than 0.25 seconds per token) even on a laptop CPU. Based on these statistics, our proposed speculative beam search algorithm can be used in real simultaneous translation.

### 4.5 Performance on Full Sentence MT

We analyze the performance of speculative beam search on full-sentence translation (see Fig. 7). By only performing beam search on a sliding speculative window, the proposed algorithm achieves much better BLEU scores compared with greedy decoding ($w = 0$) and even outperforms conventional beam search when $w = 9$, $b = 3$. Please note that the space complexity of this algorithm is $O((m + n + wb)d)$.[1] This is better than conventional beam search whose space complexity is $O((m + nb)d)$ when $w \ll n$.

## 5 Conclusions and Future Work

We have proposed speculative beam search for simultaneous translation. Experiments on three approaches to simultaneous translation demonstrate effectiveness of our method. This algorithm has the potential in other incremental tasks such as streaming ASR and incremental TTS.

## Acknowledgments

We thank Kaibo Liu for his AL script[2] and help in training wait-$k$ models, and the anonymous reviewers for suggestions.

[1] Here $n$ is the length of target sentence and $d$ is the representation dimension.

[2] https://github.com/SimulTrans-demo/STACL

## References


Naveen Arivazhagan, Colin Cherry, Wolfgang Macherey, Chung-Cheng Chiu, Semih Yavuz, Ruoming Pang, Wei Li, and Colin Raffel. 2019. Monotonic infinite lookback attention for simultaneous machine translation. *arXiv preprint arXiv:1906.05218*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Srinivas Bangalore, Vivek Kumar Rangarajan Sridhar, Prakash Kolan, Ladan Golipour, and Aura Jimenez. 2012. Real-time incremental speech-to-speech translation of dialogs. In *Proc. of NAACL-HLT*.

Chung-Cheng Chiu, Tara N Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J Weiss, Kanishka Rao, Ekaterina Gonina, et al. 2018. State-of-the-art speech recognition with sequence-to-sequence models. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4774–4778. IEEE.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, and Stephan Vogel. 2018. Incremental decoding and training methods for simultaneous translation in neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 493–499, New Orleans, Louisiana. Association for Computational Linguistics.

Alvin Grissom II, He He, Jordan Boyd-Graber, John Morgan, and Hal Daumé III. 2014. Don't until the final verb wait: Reinforcement learning for simultaneous machine translation. In *Proceedings of the 2014 Conference on empirical methods in natural language processing (EMNLP)*, pages 1342–1352.

Jiatao Gu, Graham Neubig, Kyunghyun Cho, and Victor O. K. Li. 2017. Learning to translate in real-time with neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*, pages 1053–1062.

Liang Huang, Kai Zhao, and Mingbo Ma. 2017. When to finish? optimal beam search for neural text generation (modulo beam size). In *EMNLP*.

Navdeep Jaitly, David Sussillo, Quoc V Le, Oriol Vinyals, Ilya Sutskever, and Samy Bengio. 2016. An online sequence-to-sequence model using partial conditioning. In *Advances in Neural Information Processing Systems*, pages 5067–5075.

G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush. 2017. OpenNMT: Open-Source Toolkit for Neural Machine Translation. *ArXiv e-prints*.

Mingbo Ma, Liang Huang, Hao Xiong, Renjie Zheng, Kaibo Liu, Baigong Zheng, Chuanqiang Zhang, Zhongjun He, Hairong Liu, Xing Li, Hua Wu, and Haifeng Wang. 2019a. STACL: Simultaneous translation with implicit anticipation and controllable latency using prefix-to-prefix framework. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3025–3036.

Mingbo Ma, Renjie Zheng, and Liang Huang. 2019b. Learning to stop in structured prediction for neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1884–1889.

Kenton Murray and David Chiang. 2018. Correcting length bias in neural machine translation. In *Proceedings of WMT 2018*.

Yusuke Oda, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2015. Syntax-based simultaneous translation through prediction of unseen syntactic constituents. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 198–207.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. *ICLR*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, volume 14, pages 77–81.

Baigong Zheng, Renjie Zheng, Mingbo Ma, and Liang Huang. 2019a. Simpler and faster learning of adaptive policies for simultaneous translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing*.

Baigong Zheng, Renjie Zheng, Mingbo Ma, and Liang Huang. 2019b. Simultaneous translation with flexible policy via restricted imitation learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5816–5822.

## A Supplemental Material

We also evaluate our work using Consecutive Wait (CW) as latency metric, which measures the average lengths of consecutive wait segments, and perform experiments on German↔English corpora available from WMT15[3]. We use newstest-2013 as dev-set and newstest-2015 as test-set.[4]

Fig. 8 show the translation quality on German↔English against AL of different decoding methods. Consistent to the results of Chinese↔English, our proposed speculative beam search gain large performance boost especially on test-time wait-$k$. Fig. 9 and Fig. 10 use CW as latency metrics. Since both the wait-$k$ and test-time wait-$k$ models use the same fixed policy, the CW latencies of the same $k$ are identical.

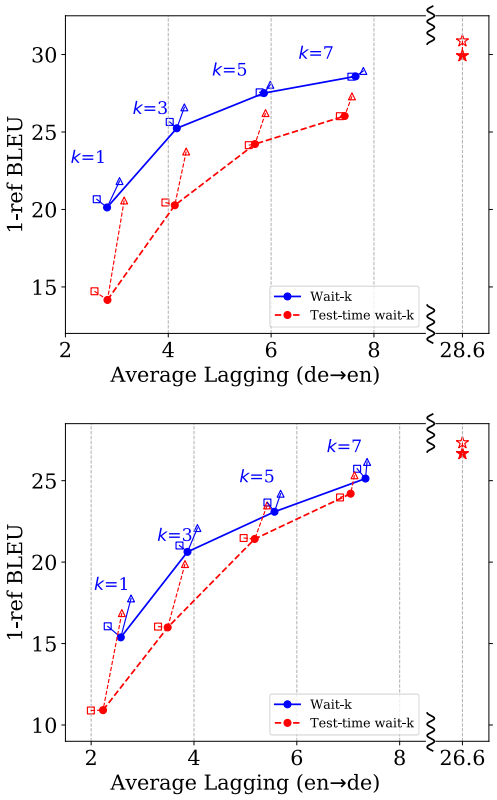


Figure 8: Translation quality against AL on English↔German simultaneous translation using wait-$k$ model. □ □: conventional beam search only on target tail. △ △: speculative beam search. ★☆:full-sentence (greedy and beam-search).

[3] http://www.statmt.org/wmt15/translation-task.html

[4] The German↔English results are slightly different from those in Ma et al. (2019a) because of different decoding settings. We do not allow that the decoder stops earlier than the finish of source sentence while it is allowed in German↔English experiments of Ma et al. (2019a). This makes our generated sentences longer and further results in worse AL compared with the results in Ma et al. (2019a).

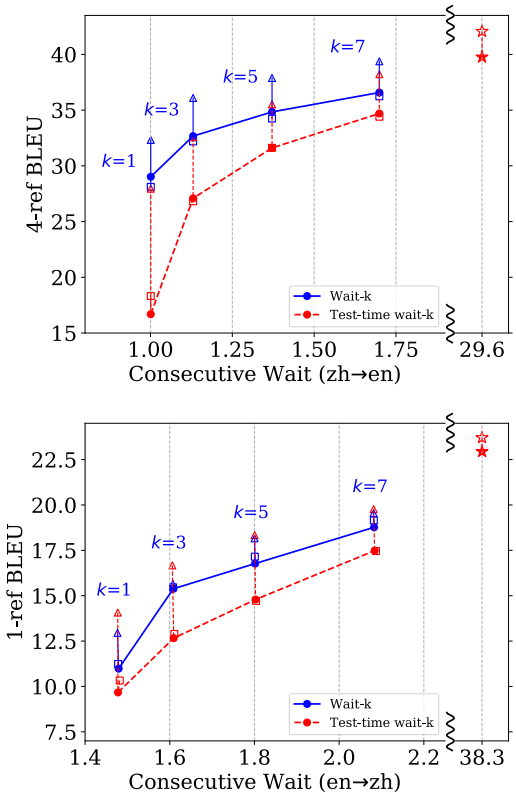


Figure 9: Translation quality against CW on Chinese↔English simultaneous translation using wait-$k$ model.

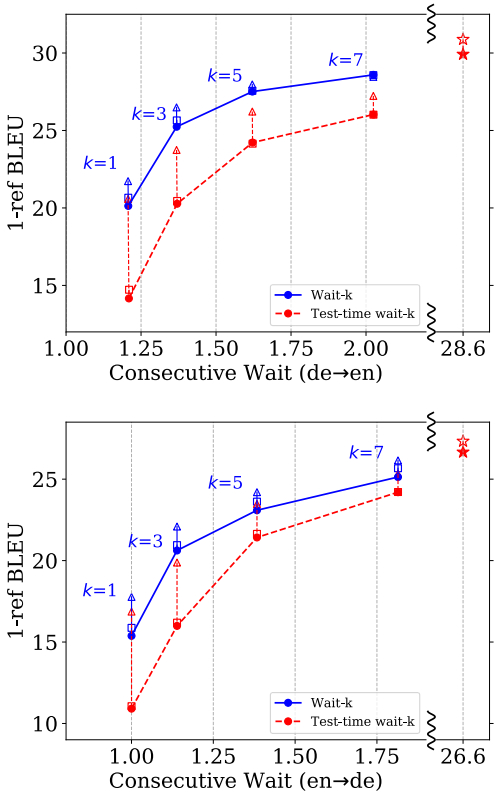


Figure 10: Translation quality against CW on English↔German simultaneous translation using wait-$k$ model.

# Self-Attention with Structural Position Representations

**Xing Wang**
Tencent AI Lab
brightxwang@tencent.com

**Zhaopeng Tu**
Tencent AI Lab
zptu@tencent.com

**Longyue Wang**
Tencent AI Lab
vinnylywang@tencent.com

**Shuming Shi**
Tencent AI Lab
shumingshi@tencent.com

## Abstract

Although self-attention networks (SANs) have advanced the state-of-the-art on various NLP tasks, one criticism of SANs is their ability of encoding positions of input words (Shaw et al., 2018). In this work, we propose to augment SANs with *structural position representations* to model the latent structure of the input sentence, which is complementary to the standard sequential positional representations. Specifically, we use dependency tree to represent the grammatical structure of a sentence, and propose two strategies to encode the positional relationships among words in the dependency tree. Experimental results on NIST Chinese⇒English and WMT14 English⇒German translation tasks show that the proposed approach consistently boosts performance over both the absolute and relative sequential position representations.

## 1 Introduction

In recent years, self-attention networks (SANs, Parikh et al., 2016; Lin et al., 2017) have achieved the state-of-the-art results on a variety of NLP tasks (Vaswani et al., 2017; Strubell et al., 2018; Devlin et al., 2019). SANs perform the attention operation under the position-unaware "bag-of-words" assumption, in which positions of the input words are ignored. Therefore, absolute position (Vaswani et al., 2017) or relative position (Shaw et al., 2018) are generally used to capture the sequential order of words in the sentence. However, several researches reveal that the sequential structure may not be sufficient for NLP tasks (Tai et al., 2015; Kim et al., 2017; Shen et al., 2019), since sentences inherently have hierarchical structures (Chomsky, 1965; Bever, 1970).

In response to this problem, we propose to augment SANs with *structural position representations* to capture the hierarchical structure of the input sentence. The starting point for our approach is a recent finding: the latent structure of a sentence can be captured by structural depths and distances (Hewitt and Manning, 2019). Accordingly, we propose *absolute structural position* to encode the depth of each word in a parsing tree, and *relative structural position* to encode the distance of each word pair in the tree.

We implement our structural encoding strategies on top of TRANSFORMER (Vaswani et al., 2017) and conduct experiments on both NIST Chinese⇒English and WMT14 English⇒German translation tasks. Experimental results show that exploiting structural position encoding strategies consistently boosts performance over both the absolute and relative sequential position representations across language pairs. Linguistic analyses (Conneau et al., 2018) reveal that the proposed structural position representation improves the translation performance with richer syntactic information. Our main contributions are:

- Our study demonstrates the necessity and effectiveness of exploiting structural position encoding for SANs, which benefits from modeling syntactic depth and distance under the latent structure of the sentence.
- We propose structural position representations for SANs to encode the latent structure of the input sentence, which are complementary to their sequential counterparts.

## 2 Background

**Self-Attention** SANs produce representations by applying attention to each pair of elements from the input sequence, regardless of their distance. Given an input sequence $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_I\} \in \mathbb{R}^{I \times d}$, the model first transforms



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1403–1409, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

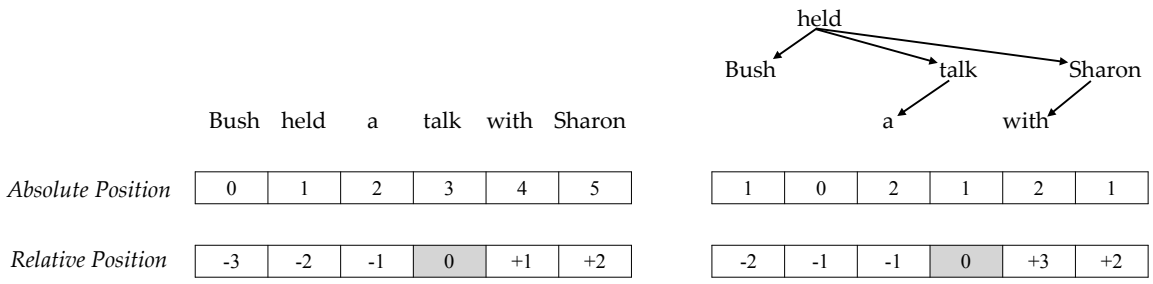


(a) **Sequential** Position Encoding (b) **Structural** Position Encoding

Figure 1: Illustration of (a) the standard *sequential position encoding* (Vaswani et al., 2017; Shaw et al., 2018), and (b) the proposed *structural position encoding*. The relative position in the example is for the word "talk".

it into queries $\mathbf{Q} \in \mathbb{R}^{I\times d}$, keys $\mathbf{K} \in \mathbb{R}^{I\times d}$, and values $\mathbf{V} \in \mathbb{R}^{I\times d}$:

$$\mathbf{Q}, \mathbf{K}, \mathbf{V} = \mathbf{X}\mathbf{W}_Q, \mathbf{X}\mathbf{W}_K, \mathbf{X}\mathbf{W}_V \in \mathbb{R}^{I\times d}, \quad (1)$$

where $\{\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V\} \in \mathbb{R}^{d\times d}$ are trainable parameters and $d$ indicates the hidden size. The output sequence is calculated as

$$\mathbf{O} = \textsc{Att}(\mathbf{Q}, \mathbf{K})\, \mathbf{V} \qquad \in \mathbb{R}^{I\times d}, \quad (2)$$

where $\textsc{Att}(\cdot)$ is a dot-product attention model.

**Sequential Position Encoding** To make use of the order of the sequence, information about the absolute or relative position of the elements in the sequence is injected into SAN:

- *Absolute* Sequential PE (Vaswani et al., 2017) is defined as

$$\textsc{AbsPE}(abs) = f(abs/10000^{2i/d}), \quad (3)$$

  where $abs$ is the absolute position in the sequence and $i$ is the dimension of position representations. $f(\cdot)$ is $\sin(\cdot)$ for the even dimension, and $\cos(\cdot)$ for the odd dimension.

  Vaswani et al. (2017) propose to conduct element-wise addition to combine the fixed sequential position representation with word embedding and feed the combination representation to the SANs.

- *Relative* Sequential PE (Shaw et al., 2018) is calculated as

$$\textsc{RelPE}(rel) = \mathbf{R}[rel], \quad (4)$$

  where $rel$ is the relative position to the queried word, which is used to index a learnable matrix $\mathbf{R}$ that represents relative position embeddings.

  Shaw et al. (2018) propose relation-aware SANs and take the relative sequential encoding as the additional key and value (Eq.2) in the attention computation.

Figure 1(a) shows an example of absolute (i.e., $abs$) and relative (i.e., $rel$) sequential positions.

## 3 Approach

### 3.1 Structural Position Representations

In this study, we choose dependency tree to represent sentence structure for its simplicity on modeling syntactic relationships among input words. Figure 1 shows an example to illustrate the idea of the proposed approach. From the perspective of relationship path between words, sequential PE measures the sequential distance between the words. As shown in Figure 1 (a), for each word, absolute sequential position represents the sequential distance to the beginning of the sentence, while relative sequential position measures the relative distance to the queried word ("talk" in the example).

The latent structure can be interpreted in various ways, from syntactic tree structures, e.g., constituency tree (Collins, 2003) or dependency tree (Kübler et al., 2009), to semantic graph structures, e.g., abstract meaning representation graph (Banarescu et al., 2013). In this work, dependency path, which is induced from the dependency tree, is adopted to provide a new perspective on modelling pairwise relationships.

Figure 1 shows the difference between the sequential path and dependency path. The sequential distance between the two words "held" and "talk" is 2, while their structural distance is only 1 as word "talk" is the dependent of the head "held" (Nivre, 2005).

***Absolute* Structural Position** We exploit the tree depth of the word in the dependency tree as its absolute structural position. Specifically, we treat the main verb (Tapanainen and Jarvinen, 1997) of the sentence as the origin and use the distance of the dependency path from the target word to the origin as the absolute structural position

$$abs_{stru}(x_i) = distance_{tree}(x_i, origin), \quad (5)$$

where $x_i$ is the target word, $tree$ is the given dependency structure and the origin is the main verb of the $tree$.

In the field of NMT, *BPE sub-words* and *end-of-sentence symbol* should be carefully handled as they do not appear in the conventional dependency tree. In this work, we assign the *BPE sub-words* share the absolute structural position of the original word and set the the first larger integer than the max absolute structural position in dependency tree as the absolute structural position of *end-of-sentence symbol*.

***Relative* Structural Position** For the *relative* structural position $rel_{stru}(x_i, x_j)$, we calculate $rel_{stru}(x_i, x_j)$ in the dependency tree following two hierarchical rules:

1. if $x_i$ and $x_j$ are at same dependency edge, $rel_{stru}(x_i, x_j) = abs_{stru}(x_i) - abs_{stru}(x_j)$.
2. if $x_i$ and $x_j$ are at different dependency edges, $rel_{stru}(x_i, x_j) = f_{stru}(i - j) * (abs_{stru}(x_i) + abs_{stru}(x_j))$, where

$$f_{stru}(x) = \begin{cases} 1 & x > 0 \\ 0 & x = 0 \\ -1 & x < 0 \end{cases} \quad (6)$$

Following Shaw et al. (2018), we use clipping distance $r$ to limit the maximum relative position.

### 3.2 Integrating Structural PE into SANs

We inherit position encoding functions from sequential approaches (Eq.3 and Eq.4) to implement the structural position encoding strategies. Since structural position representations capture complementary position information to their sequential counterparts, we also exploit to integrate the structural position encoding into SANs with the sequential counterparts approaches (Vaswani et al., 2017; Shaw et al., 2018).

For the *absolute* position, we use the nonlinear function to fuse the sequential and structure position representations[1]:

$$asb(x_i) = f_{abs}(\text{ABSPE}(abs_{seq}), \text{ABSPE}(abs_{stru})) \quad (7)$$

where $f_{abs}$ is the nonlinear function. $\text{ABSPE}(abs_{seq})$ and $\text{ABSPE}(abs_{stru})$ are absolute sequential and structural position embedding in Eq.3 and Eq.5 respectively.

For the *relative* position, we follow Shaw et al. (2018) to extend the self-attention computation to consider the pairwise relationships and project the relative structural position as described at Eq.(3) and Eq.(4) in Shaw et al. (2018)[2].

## 4 Related Work

There has been growing interest in improving the representation power of SANs (Dou et al., 2018, 2019; Yang et al., 2018; Wang et al., 2018; Wu et al., 2018; Yang et al., 2019a,b; Sukhbaatar et al., 2019). Among these approaches, a straightforward strategy is that augmenting the SANs with position representations (Shaw et al., 2018; Ma et al., 2019; Bello et al., 2019; Yang et al., 2019c), as the position representations involves element-wise attention computation. In this work, we propose to augment SANs with structural position representations to model the latent structure of the input sentence.

Our work is also related to the structure modeling for SANs, as the proposed model utilizes the dependency tree to generate structural representations. Recently, Hao et al. (2019c,b) integrate the recurrence into the SANs and empirically demonstrate that the hybrid models achieve better performances by modeling structure of sentences. Hao et al. (2019a) further make use of the multi-head attention to form the multi-granularity self-attention, to capture the different granularity phrases in source sentences. The difference is that we treat the position representation as a medium to transfer the structure information from the dependency tree into the SANs.

[1] We also use parameter-free element-wise addition method to combine two absolute position embedding and get 0.28 BLEU point improvement on development set of NIST Chinese⇒English over the baseline model that only uses absolute sequential encoding.

[2] Due to the space limitations we do not show these functions. Please refer to Shaw et al. (2018) for more detail.

| # | Sequential | | Structural | | Spd. | BLEU |
|---|---|---|---|---|---|---|
| | *Abs.* | *Rel.* | *Abs.* | *Rel.* | | |
| 1 | | | × | × | 2.81 | 28.33 |
| 2 | × | × | ✓ | × | 2.53 | 35.43 |
| 3 | | | × | ✓ | 2.65 | 34.23 |
| 4 | | | × | × | 3.23 | 44.31 |
| 5 | ✓ | × | ✓ | × | 2.65 | 44.84 |
| 6 | | | ✓ | ✓ | 2.52 | 45.10 |
| 7 | | | × | × | 3.18 | 45.02 |
| 8 | ✓ | ✓ | ✓ | × | 2.64 | 45.43 |
| 9 | | | ✓ | ✓ | 2.48 | 45.67 |

Table 1: Impact of the position encoding components on Chinese⇒English NIST02 development dataset using Transformer-Base model. "Abs." and "Rel." denote *absolute* and *relative* position encoding, respectively. "Spd." denotes the decoding speed (sentences/second) on a Tesla M40, the speed of structural position encoding strategies include the step of dependency parsing.

## 5 Experiment

We conduct experiments on the widely used NIST Chinese⇒English and WMT14 English⇒German data, and report the 4-gram BLEU score (Papineni et al., 2002).

**Chinese⇒English** We use the training dataset consists of about 1.25 million sentence pairs. NIST 2002 (MT02) dataset is used as development set. NIST 2003, 2004, 2005, 2006 datasets are used as test sets. We use byte-pair encoding (BPE) toolkit to alleviate the out-of-vocabulary problem with 32K merge operations.

**English⇒German** We use the dataset consisting of about 4.5 million sentence pairs as the training set. The newstest2013 and newstest2014 are used as the development set and the test set. We also apply BPE with 32K merge operations to obtain subword unit.

We evaluate the proposed position encoding strategies on TRANSFORMER (Vaswani et al., 2017) and implement them on top of THUMT (Zhang et al., 2017). We use the Stanford parser (Klein and Manning, 2003) to parse the sentences and obtain the structural structural absolute and relative position as described in Section 3. When using relative structural position encoding, we use clipping distance $r = 16$. To make a fair comparison, we valid different position encoding strategies on the encoder and keep the TRANSFORMER decoder unchanged. We study the variations of the BASE model on Chinese⇒English task, and evaluate the overall performance with the BIG model on both translation tasks.

### 5.1 Model Variations

We evaluate the importance of the proposed absolute and relative structural position encoding strategies and study the variations with *Transformer-Base* model on Chinese⇒English data. The experimental results on the development set are shown in Table 1.

**Effect of Position Encoding** We first remove the sequential encoding from the *Transformer* encoder (Model #1) and observe the translation performance degrades dramatically ($28.33 - 44.31 = -15.98$), which demonstrates the necessity of the position encoding strategies.

**Effect of Structural Position Encoding** Then we valid our proposed structural position encoding strategies over the position-unaware model (Models #2-3). We find that absolute and relative structural position encoding strategies improve the translation performance by 7.10 BLEU points and 5.90 BLEU points respectively, which shows that the introducing of the proposed absolute and relative structural positions improves the translation performance in terms of BLEU score.

**Combination of Sequential and Structural Position Encoding Strategies** We integrate the absolute and relative structural position encoding strategies into the *Base* model equipped with absolute sequential position encoding (Models #4-6). We observe that the proposed two approaches are able to achieve improvements over the *Base* model with decoding speed marginally decreases.

Finally, we valid the proposed structural position encoding over the *Base* model equipped with absolute and relative sequential position encoding (Models #7-9). We find that sequential relative encoding obtains 0.71 BLEU points improvement (Model #7 vs. Model #4) and structural position encoding achieves a further improvement in performance by 0.65 BLEU points (Model #9 vs. Model #7), demonstrating the effectiveness of the proposed structural position encoding strategies.

### 5.2 Main Results

We valid the proposed structural encoding strategies over *Transformer-Big* model in

| Model Architecture | Zh⇒En | | | | | En⇒De |
|---|---|---|---|---|---|---|
| | MT03 | MT04 | MT05 | MT06 | Avg | WMT14 |
| Hao et al. (2019c) | - | - | - | - | - | 28.98 |
| *Transformer-Big* | 45.30 | 46.49 | 45.21 | 44.87 | 45.47 | 28.58 |
| + *Structural* PE | 45.62 | 47.12$^{\uparrow}$ | 45.84 | 45.64$^{\Uparrow}$ | 46.06 | 28.88 |
| + *Relative* Sequential PE | 45.45 | 47.01 | 45.65 | 45.87$^{\Uparrow}$ | 46.00 | 28.90 |
| + *Structural* PE | 45.85$^{\uparrow}$ | 47.37$^{\Uparrow}$ | 46.20$^{\Uparrow}$ | 46.18$^{\Uparrow}$ | 46.40 | 29.19$^{\Uparrow}$ |

Table 2: Evaluation of translation performance on NIST Zh⇒En and WMT14 En⇒De test sets. Hao et al. (2019c) is a *Transformer-Big* model which adopted an additional recurrence encoder with the attentive recurrent network to model syntactic structure. "↑ / ⇑": significant over the *Transformer-Big* ($p < 0.05/0.01$), tested by bootstrap resampling (Koehn, 2004).

| Model | Surface | | | Syntactic | | | | Semantic | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SeLen | WC | Avg | TrDep | ToCo | BShif | Avg | Tense | SubN | ObjN | SoMo | CoIn | Avg |
| BASE | 92.20 | 63.00 | 77.60 | 44.74 | 79.02 | 71.24 | 65.00 | 89.24 | 84.69 | 84.53 | 52.13 | 62.47 | 74.61 |
| + *Rel. Seq.* PE | 89.82 | 63.17 | 76.50 | 45.09 | 78.45 | 71.40 | 64.98 | 88.74 | 87.00 | 85.53 | 51.68 | 62.21 | 75.03 |
| + *Stru.* PE | 89.54 | 62.90 | 76.22 | 46.12 | 79.12 | 72.36 | 65.87 | 89.30 | 85.47 | 84.94 | 52.90 | 62.99 | 75.12 |

Table 3: Performance on linguistic probing tasks. The probing tasks were conducted by evaluating linguistics embedded in the *Transformer-Base* encoder outputs. "Base", "+ *Rel. Seq.* PE", "+ *Stru.* PE" denote *Transformer-Base*, *Transformer-Base* with relative sequential PE, *Transformer-Base* with relative sequential PE and structural PE models respectively.

Chinese⇒English and English⇒German data, and list the results in Table 2.

For Chinese⇒English, Structural position encoding (+ *Structural* PE) outperforms the *Transformer-Big* by 0.59 BLEU points on average over four NIST test sets. Sequential relative encoding approach (+*Relative* Sequential PE) outperforms the *Transformer-Big* by 0.53 BLEU points, and structural position encoding (+ *Structural* PE) achieves further improvement up to +0.40 BLEU points and outperforms the *Transformer-Big* by 0.93 BLEU points. For English⇒German, similar phenomenon is observed, which reveals that the proposed structural position encoding strategy can consistently boost translation performance over both the absolute and relative sequential position representations.

### 5.3 Linguistic Probing Evaluation

We conduct probing tasks[3] (Conneau et al., 2018) to evaluate structure knowledge embedded in the encoder output in the variations of the *Base* model that are trained on En⇒De translation task.

We follow Wang et al. (2019) to set model configurations. The experimental results on probing tasks are shown in Table 3, and the BLEU scores of "Base", "+ *Rel. Seq.* PE", "+ *Stru.* PE" are 27.31, 27.99 and 28.30. From the table, we can see 1) adding the relative sequential positional embedding achieves improvement over the baseline on semantic tasks (75.03 vs. 74.61). This may indicate the model benefits more from semantic modeling; 2) with the structural positional embedding, the model obtains improvement on syntactic tasks (65.87 v.s. 64.98), which indicates that the representations preserve more syntactic knowledge.

[3] https://github.com/facebookresearch/SentEval/tree/master/data/probing

## 6 Conclusion

In this paper, we have presented a novel structural position encoding strategy to augment SANs by considering the latent structure of the input sentence. We extract structural absolute and relative positions from the dependency tree and integrate them into SANs. Experimental results on Chinese⇒English and English⇒German translation tasks have demonstrated that the proposed approach consistently improve translation performance over both the absolute and relative sequential position representations.

Future directions include inferring the structure representations from the AMR (Song et al., 2019) or the external SMT knowledge (Wang et al., 2017). Furthermore, the structural position encoding can be also applied to the decoder with RNN Grammars (Dyer et al., 2016; Eriguchi et al., 2017), which we leave for future work.

## References


Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *the 7th Linguistic Annotation Workshop and Interoperability with Discourse*.

Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V Le. 2019. Attention augmented convolutional networks. *arXiv*.

Thomas G Bever. 1970. The cognitive basis for linguistic structures. *Cognition and the development of language*, 279(362).

Noam Chomsky. 1965. *Aspects of the Theory of Syntax*, volume 11. MIT press.

Michael Collins. 2003. Head-driven statistical models for natural language parsing. *Computational linguistics*, 29(4).

Alexis Conneau, Germán Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single vector: Probing sentence embeddings for linguistic properties. In *ACL*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Zi-Yi Dou, Zhaopeng Tu, Xing Wang, Shuming Shi, and Tong Zhang. 2018. Exploiting deep representations for neural machine translation. In *EMNLP*.

Ziyi Dou, Zhaopeng Tu, Xing Wang, Longyue Wang, Shuming Shi, and Tong Zhang. 2019. Dynamic layer aggregation for neural machine translation. In *AAAI*.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A Smith. 2016. Recurrent neural network grammars. In *NAACL*.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to parse and translate improves neural machine translation. In *ACL*.

Jie Hao, Xing Wang, Shuming Shi, Jinfeng Zhang, and Zhaopeng Tu. 2019a. Multi-granularity self-attention for neural machine translation. In *EMNLP*.

Jie Hao, Xing Wang, Shuming Shi, Jinfeng Zhang, and Zhaopeng Tu. 2019b. Towards better modeling hierarchical structure for self-attention with ordered neurons. In *EMNLP*.

Jie Hao, Xing Wang, Baosong Yang, Longyue Wang, Jinfeng Zhang, and Zhaopeng Tu. 2019c. Modeling recurrence for transformer. In *NAACL*.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *NAACL*.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M Rush. 2017. Structured attention networks. In *ICLR*.

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *ACL*.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *EMNLP*.

Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. Dependency parsing. *Synthesis Lectures on Human Language Technologies*, 1(1).

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A Structured Self-attentive Sentence Embedding. In *ICLR*.

Chunpeng Ma, Akihiro Tamura, Masao Utiyama, Eiichiro Sumita, and Tiejun Zhao. 2019. Improving neural machine translation with neural syntactic distance. In *NAACL)*.

Joakim Nivre. 2005. Dependency grammar and dependency parsing. *MSI report*, 5133(1959).

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A Decomposable Attention Model for Natural Language Inference. In *EMNLP*.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-Attention with Relative Position Representations. In *NAACL*.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. 2019. Ordered neurons: Integrating tree structures into recurrent neural networks. In *ICLR*.

Linfeng Song, Daniel Gildea, Yue Zhang, Zhiguo Wang, and Jinsong Su. 2019. Semantic neural machine translation using AMR. *TACL*.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-Informed Self-Attention for Semantic Role Labeling. In *EMNLP*.

Sainbayar Sukhbaatar, Edouard Grave, Piotr Bojanowski, and Armand Joulin. 2019. Adaptive attention span in transformers. In *ACL*.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *ACL*.

Pasi Tapanainen and Timo Jarvinen. 1997. A non-projective dependency parser. In *Fifth Conference on Applied Natural Language Processing*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. In *NIPS*.

Qiang Wang, Fuxue Li, Tong Xiao, Yanyang Li, Yinqiao Li, and Jingbo Zhu. 2018. Multi-layer representation fusion for neural machine translation. In *COLING*.

Xing Wang, Zhengdong Lu, Zhaopeng Tu, Hang Li, Deyi Xiong, and Min Zhang. 2017. Neural machine translation advised by statistical machine translation. In *AAAI*.

Xing Wang, Zhaopeng Tu, Longyue Wang, and Shuming Shi. 2019. Exploiting sentential context for neural machine translation. In *ACL*.

Wei Wu, Houfeng Wang, Tianyu Liu, and Shuming Ma. 2018. Phrase-level self-attention networks for universal sentence encoding. In *ACL*.

Baosong Yang, Jian Li, Derek F. Wong, Lidia S. Chao, Xing Wang, and Zhaopeng Tu. 2019a. Context-aware self-attention networks. In *AAAI*.

Baosong Yang, Zhaopeng Tu, Derek F. Wong, Fandong Meng, Lidia S. Chao, and Tong Zhang. 2018. Modeling localness for self-attention networks. In *EMNLP*.

Baosong Yang, Longyue Wang, Derek F. Wong, Lidia S. Chao, and Zhaopeng Tu. 2019b. Convolutional self-attention networks. In *NAACL*.

Shan Yang, Heng Lu, Shiying Kang, Lei Xie, and Dong Yu. 2019c. Enhancing hybrid self-attention structure with relative-position-aware bias for speech synthesis. In *ICASSP*.

Jiacheng Zhang, Yanzhuo Ding, Shiqi Shen, Yong Cheng, Maosong Sun, Huanbo Luan, and Yang Liu. 2017. THUMT: An Open Source Toolkit for Neural Machine Translation. *arXiv preprint arXiv:1706.06415*.

# Exploiting Multilingualism through Multistage Fine-Tuning for Low-Resource Neural Machine Translation

**Raj Dabre** **Atsushi Fujita**
National Institute of Information and Communications Technology
3-5 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-0289, Japan
firstname.lastname@nict.go.jp

**Chenhui Chu**
Osaka University, Japan
chu@ids.osaka-u.ac.jp

## Abstract

This paper highlights the impressive utility of multi-parallel corpora for transfer learning in a one-to-many low-resource neural machine translation (NMT) setting. We report on a systematic comparison of multistage fine-tuning configurations, consisting of (1) pre-training on an external large (209k–440k) parallel corpus for English and a helping target language, (2) mixed pre-training or fine-tuning on a mixture of the external and low-resource (18k) target parallel corpora, and (3) pure fine-tuning on the target parallel corpora. Our experiments confirm that multi-parallel corpora are extremely useful despite their scarcity and content-wise redundancy thus exhibiting the true power of multilingualism. Even when the helping target language is not one of the target languages of our concern, our multistage fine-tuning can give 3–9 BLEU score gains over a simple one-to-one model.

## 1 Introduction

Encoder-decoder based neural machine translation (NMT) (Sutskever et al., 2014; Bahdanau et al., 2015) allows for end-to-end training of a translation model. While NMT is known to perform well for resource-rich language pairs, such as French–English and German–English (Bojar et al., 2018), it performs poorly for resource-poor pairs (Zoph et al., 2016; Riza et al., 2016).

Translating from a resource-poor language into English is typically easier than the other direction, because large parallel corpora between English and other languages can be used for transfer learning (Zoph et al., 2016) and many-to-one modeling.[1] Additionally, large English monolingual data can be back-translated to obtain pseudo-parallel corpora for augmenting the performance of such NMT models.

Translating from English into a resource-poor language is substantially more difficult. Dong et al. (2015) have shown that a one-to-many model trained on middle-sized parallel data (200k sentence pairs) can improve the translation quality over a one-to-one model. However, it is unclear whether this works for much resource-poorer, more distant, and more diverse language pairs. Using pseudo-parallel data is a potential solution, but for most resource-poor languages, the amount of available clean and in-domain monolingual data are limited. It is also unclear what the real reason behind improvements in translation is: increase in the training data or multilingualism.

This paper focuses on (a) improving the performance of NMT for translating English to low-resource languages (b) via exploiting multilingualism (c) through transfer learning based on multistage fine-tuning. The power of multilingualism is verified by using multi-parallel corpora, i.e., the same text in different languages.[2] Unlike previous studies that only apply single-stage fine-tuning for one-to-one translation (Zoph et al., 2016), we systematically compare multistage fine-tuning that exploits one-to-many modeling. We show that our approach can significantly improve the quality of translations from English to seven Asian languages (Bengali, Filipino, Indonesian, Japanese, Khmer, Malay, and Vietnamese) in the Asian Language Treebank (ALT) corpus (Riza et al., 2016),[3] as a result of soft division of labor: training of a strong English encoder, domain adaptation, and tuning of the decoder for each target language.

[1] Note that "many-to-one" does not mean multi-source machine translation (Zoph and Knight, 2016).

[2] Adding a new language does not add any new translation knowledge. Such corpora deserve more attention, since adding a new language to it leads to parallel corpora between the new language and all the other languages in the corpora.

[3] There exist other multi-parallel corpora, such as those for United Nations (Ziemski et al., 2016), Europarl (Koehn, 2005), Ted Talks (Cettolo et al., 2012), ILCI (Jha, 2010), and the Bible (Christodouloupoulos and Steedman, 2015).



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1410–1416,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

## 2 Related Work

In this paper, we address (a) the importance of multilingualism (b) via one-to-many NMT (c) through robust fine-tuning for transfer learning. Dong et al. (2015) have worked on one-to-many NMT, whereas Firat et al. (2016) and Johnson et al. (2017) have worked on multilingual and multi-way NMT. These studies have focused on training a multilingual model from scratch, but we explore transfer learning for one-to-many NMT.

Fine-tuning based transfer learning has been studied for transferring proper parameters (Zoph et al., 2016; Gu et al., 2018b), lexical (Zoph et al., 2016; Nguyen and Chiang, 2017; Gu et al., 2018a; Lakew et al., 2018), and syntactic (Gu et al., 2018a; Murthy et al., 2018) knowledge from a resource-rich language pair to a resource-poor language pair. On the other hand, Chu et al. (2017) proposed a more robust training approach for domain adaptation, called mixed fine-tuning, which uses a mixture of data from different domains. Imankulova et al. (2019) proposed a multistage fine-tuning method which combines fine-tuning techniques for domain adaptation and back-translation (Sennrich et al., 2016). Unlike us, these approaches do not try to combine multilingualism with multistage fine-tuning using only parallel corpora involving a large number of languages.

## 3 Multistage Fine-Tuning for NMT

This paper focuses on translation from English (En) to $N$ different languages. In particular, we consider exploiting two types of corpora. One is a small-scale multi-parallel corpus, En-$\mathrm{YY}_1$-$\cdots$-$\mathrm{YY}_N$, consisting of English and $N$ target languages of interest. The other is a relatively larger helping parallel corpus, En-XX, where XX indicates the helping target language which needs not be one of the target languages in the multi-parallel corpus, $\mathrm{YY}_k$ $(1 \leq k \leq N)$.

All the fine-tuning techniques that we examine can be generalized as follows.

**Pre-training (Pre):** An initial NMT model is trained on a parallel corpus for a resource-rich language pair, i.e., En-XX.

**Mixed pre-training / fine-tuning (Mix):** Training of the NMT model is newly started or continued (Chu et al., 2017) on a mixture of parallel corpora for En-XX and one or more low-resource En-YY pairs.

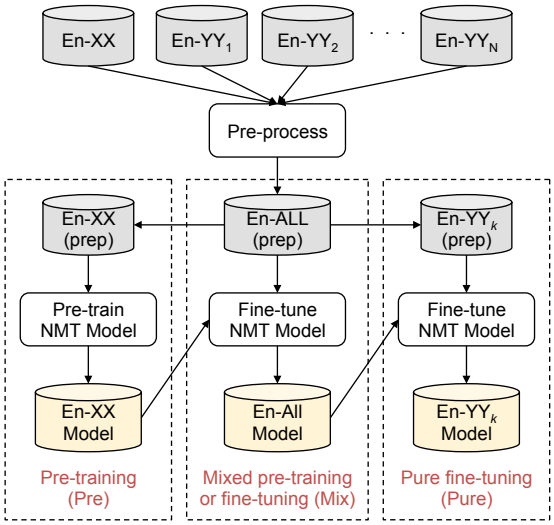


Figure 1: Our multistage fine-tuning for one-to-many NMT. Three dotted sections respectively indicate (a) the pre-training stage, (b) the mixed pre-training / fine-tuning stage, and (c) the pure fine-tuning stage The training data for (a) and (c), i.e., "En-XX (prep)" and "En-$\mathrm{YY}_k$ (prep)," are selectively extracted from the pre-processed data labeled "En-ALL (prep)."

**Pure fine-tuning (Pure):** Training further continues using only a parallel corpus for a particular En-YY pair.

Figure 1 illustrates the training procedure with all of the above three stages. Before starting the training, vocabularies for source and target languages are determined. First, to indicate the target language of each sentence pair, we prepend an artificial token, for instance "2zz" for a target language "zz," onto source sentences (Johnson et al., 2017). All corpora are then concatenated into a single corpus, "En-ALL (prep)," where each of the smaller corpora are enlarged by oversampling so that the size of each corpus matches the largest one. Finally, (sub-word) vocabularies are determined based on "En-ALL (prep)." For simplicity, we keep source and target vocabularies separate, but the target vocabulary is shared by multiple languages. Note that all the NMT models are essentially one-to-many, because they use a shared target vocabulary of XX and one or all of $\mathrm{YY}_k$.

Our training procedure can be viewed as a soft division of labor in NMT training: the first stage learns a strong English encoder by training on a large parallel corpus,[4] whereas the remaining

[4]One may be interested in training an English encoder on a gigantic monolingual corpus. We leave a comparison of the utility of such monolingual corpora and the parallelism in bilingual corpora for our future work.

stages can focus more on training the decoder for the target language(s) of interest and updating the encoder. Whereas we pre-train an encoder for low-resource NMT, so do McCann et al. (2017) to improve several monolingual NLP tasks.

## 4 Experiments

We conducted a systematic comparison of multistage fine-tuning configurations, specifically focusing on low-resource English-to-Asian language translation.

### 4.1 Datasets and Pre-processing

As the test-bed, we chose the ALT multilingual multi-parallel corpus[5] (Riza et al., 2016), because it offers the same test sentences in different languages, allowing us to determine whether multilingualism is the true reason behind improved translation quality. We used seven Asian languages[6] as the target languages (i.e., YY): Bengali (Bn), Filipino (Tl), Indonesian (Id), Japanese (Ja), Khmer (Km), Malay (Ms), and Vietnamese (Vi). We randomly[7] split the ALT data into 18,000, 1,000, and 1,106 multi-parallel sentences for training, development, and testing, respectively.

As for the resource-rich En-XX data, we separately used two parallel corpora with two different target languages. One is the IWSLT 2015 English–Chinese corpus (IWSLT En-Zh) (Cettolo et al., 2015), comprising 209,491 parallel sentences. We chose it to examine whether the target language (Chinese) other than the target languages in our setting is helpful. Separately, we also experimented with the Kyoto free translation task English–Japanese corpus (KFTT En-Ja),[8] consisting of Japanese-to-English translations: 440,288 parallel sentences. With this corpus, we could additionally observe how ALT En-Ja can benefit from KFTT En-Ja. Note that the ALT, IWSLT, and KFTT corpora belong to the news, spoken, and Wikipedia domains, respectively.

We tokenized English sentences using the *tokenizer.perl* script in Moses.[9] To tokenize Chinese and Japanese sentences, we used KyotoMorph[10] and JUMAN,[11] respectively. For the other languages in the ALT corpus, we used the data in its raw form, except Khmer for which segmented data were provided by the ALT project.

### 4.2 Training NMT Models

We compared a total of seven training configurations as shown in Table 1. The model #1 is the simplest baseline model trained only on the En-YY parallel corpus. The models #2 to #4 are three conceivable combinations of fine-tuning stages that exploit an external large parallel corpus for En-XX, showing the limit reachable without a multi-parallel corpus. The models #5 to #7 follow the same training procedure with #2 to #4, respectively, exploiting a small-scale but multi-parallel corpus containing all the seven target languages.

Note that the source side vocabulary is the same across models #2 to #7 and is determined using the English side of the En-XX corpus and the English part of the multi-parallel corpus. On the other hand, the target side vocabulary for models #2 to #4 is different from the vocabulary for models #5 to #7. This is because vocabularies for models #2 to #4 involve the target side of the En-XX corpus and "one" non-English corpus from the multi-parallel corpus. On the other hand, the vocabularies for models #5 to #7 involve the target sides of the En-XX corpus and "all" non-English corpora from the multi-parallel corpus.

We used the open-source implementation of the Transformer model (Vaswani et al., 2017) in the version 1.6 branch of *tensor2tensor*,[12] and used hyper-parameter settings[13] corresponding to *transformer_base_single_gpu* which uses dropout by default. We also used the default sub-word segmentation mechanism of tensor2tensor. The "En-YY" models (#1) used a vocabulary of 8,192 ($2^{13}$) sub-words due to small corpora sizes, whereas all the other models used a vocabulary of 32,768 ($2^{15}$) sub-words to account for the larger amount of parallel data in multiple languages. We trained the models for a sufficient number of iterations till

[5] http://www2.nict.go.jp/astrec-att/member/mutiyama/ALT/

[6] Thai and Burmese (Myanmar) were excluded due to the unavailability of reliable tokenized data. Although some of the YY languages in our experiments, such as Japanese, were relatively resource-rich, we exploited only a small parallel corpus between them and English to highlight the importance of multilingualism.

[7] Our results might not be comparable to those obtained using the official splits mentioned on the website.

[8] http://www.phontron.com/kftt/

[9] https://github.com/moses-smt/mosesdecoder

[10] https://bitbucket.org/msmoshen/kyotomorph-beta

[11] http://nlp.ist.i.kyoto-u.ac.jp/EN/index.php?JUMAN

[12] https://github.com/tensorflow/tensor2tensor

[13] We did not perform any hyper-parameter tuning because we wanted to focus more on simple out-of-the box approaches coupled with multilingualism.

| # | XX | $N$ | Model capacity | Training configuration | | | YY test set | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Pre | Mix | Pure | Bn | Tl | Id | Ja | Km | Ms | Vi |
| 1. | - | 1 | 1-to-1 | - | - | ✓ | 3.99 | 24.04 | 24.10 | 11.03 | 22.53 | 29.85 | 27.39 |
| 2. | Zh | 1 | 1-to-2 | ✓ | - | ✓ | 8.86$^{*}$ | 27.54$^{*}$ | 27.10$^{*}$ | 19.07$^{*}$ | 28.41$^{*}$ | 32.52$^{*}$ | 34.63$^{*}$ |
| 3. | Zh | 1 | 1-to-2 | - | ✓ | ✓ | 4.90$^{*}$ | 23.07 | 23.37 | 13.97$^{*}$ | 26.13$^{*}$ | 29.24 | 29.82$^{*}$ |
| 4. | Zh | 1 | 1-to-2 | ✓ | ✓ | ✓ | 7.99$^{*}$ | 26.61$^{*}$ | 25.62$^{*}$ | 18.39$^{*}$ | 27.49$^{*}$ | 31.63$^{*}$ | 34.22$^{*}$ |
| 5. | Zh | 7 | 1-to-8 | ✓ | - | ✓ | 8.54$^{*}$ | 26.88$^{*}$ | 26.02$^{*}$ | 18.99$^{*}$ | 27.07$^{*}$ | 32.39$^{*}$ | 33.32$^{*}$ |
| 6. | Zh | 7 | 1-to-8 | - | ✓ | ✓ | 9.43$^{*}$ | 25.86$^{*}$ | 26.33$^{*}$ | 19.34$^{*}$ | 26.86$^{*}$ | 32.39$^{*}$ | 33.28$^{*}$ |
| 7. | Zh | 7 | 1-to-8 | ✓ | ✓ | ✓ | **10.30**$^{*+\dagger}$ | **28.22**$^{*+\dagger}$ | **27.24**$^{*\dagger}$ | **20.08**$^{*+\dagger}$ | **28.66**$^{*\dagger}$ | **33.19**$^{*+\dagger}$ | **35.34**$^{*+\dagger}$ |
| 2. | Ja | 1 | 1-to-2 | ✓ | - | ✓ | 9.16$^{*}$ | 28.06$^{*}$ | 26.53$^{*}$ | 21.55$^{*}$ | 27.98$^{*}$ | 33.68$^{*}$ | 33.93$^{*}$ |
| 3. | Ja | 1 | 1-to-2 | - | ✓ | ✓ | 4.37 | 22.91 | 23.37 | 16.47$*$ | 23.36$*$ | 29.28 | 29.10$*$ |
| 4. | Ja | 1 | 1-to-2 | ✓ | ✓ | ✓ | 8.77$^{*}$ | 26.64$^{*}$ | 25.88$^{*}$ | 21.61$^{*}$ | 27.55$^{*}$ | 32.45$^{*}$ | 34.29$^{*}$ |
| 5. | Ja | 7 | 1-to-8 | ✓ | - | ✓ | 9.43$^{*}$ | 27.45$^{*}$ | 26.70$^{*}$ | 21.79$^{*}$ | 27.87$^{*}$ | 32.92$^{*}$ | 34.28$^{*}$ |
| 6. | Ja | 7 | 1-to-8 | - | ✓ | ✓ | 9.96$^{*+}$ | 28.39$^{*}$ | 27.22$^{*+}$ | 21.03$^{*}$ | 28.91$^{*+}$ | 33.75$^{*}$ | 36.00$^{*+}$ |
| 7. | Ja | 7 | 1-to-8 | ✓ | ✓ | ✓ | **10.77**$^{*+\dagger}$ | **28.62**$^{*+}$ | **28.89**$^{*+\dagger}$ | **22.60**$^{*+\dagger}$ | **30.03**$^{*+\dagger}$ | **34.75**$^{*+\dagger}$ | **37.06**$^{*+\dagger}$ |

Table 1: BLEU scores for all the tested configurations. The "XX" column indicates the resource-rich external (helping) parallel corpus if used, where "Zh" and "Ja" stand for using the IWSLT English–Chinese and the KFTT English–Japanese corpora, respectively. The columns under "YY test set" indicate the the target languages in our multi-parallel corpus, where **bold** marks the best scores for each target language with each external parallel corpus. The "$*$," "$+$," and "$\dagger$" respectively mark the scores significantly (bootstrap re-sampling with $p < 0.05$) better than the 1-to-1 "En-YY" model (#1), the best model without the multi-parallel corpus (#2–#4), and the strongest baselines (#5–#6).

the BLEU score on the development set (simply concatenated unlike training data) did not vary by more than 0.1 BLEU over 10 checkpoints.

Instead of choosing the model with the best development set BLEU, we averaged the last 10 checkpoints saved every after 1,000 updates, following Vaswani et al. (2017), and decoded the test sets with a beam size of 4 and a length penalty, $\alpha$, of 0.6 consistently across all the models.

### 4.3 Results

Table 1 gives the BLEU scores (Papineni et al., 2002) for all the configurations. Among the seven configurations, irrespective of the external parallel corpus for En-XX, the three-stage fine-tuned model (#7) achieved the highest BLEU scores for all the seven target languages.

Results for #1 demonstrate that NMT systems trained on 18k parallel sentences can achieve only poor results for Bn and Ja, whereas reasonably high BLEU scores ($> 20$) are achieved for the other target languages.

Introducing a large external En-XX parallel corpus improved the translation quality consistently and significantly for all the seven target languages,[14] irrespective of the way of its use: simple pre-training (#2), mixed pre-training (#3), and their combination (#4). Among these three training configurations, in most cases, the simple pre-training (#2) attained the best performance. The BLEU gains over #1 were 2.67 (Ms) to 8.04 (Ja) points with the IWSLT En-Zh corpus and 2.43 (Id) to 10.52 (Ja) points with the KFTT En-Ja corpus.

Simple fine-tuning (#2 and #5) gave great improvements over the baseline. However, #5 was worse than #2, despite being trained in the same way, because #5 involves up to eight target languages, inevitably reducing the allowance for each target language. In contrast, exploiting the multi-parallel corpus through mixed pre-training (#6) or mixed fine-tuning (#7) brought consistent improvements over the corresponding 1-to-2 models (#3 and #4). Three-stage fine-tuning (#4 and #7) was always better than two-stage fine-tuning (#3 and #6), because the pre-final model in #4 and #7, obtained using mixed-fine tuning, was superior to and more robust than the one in #3 and #6, obtained using naive multilingual training. These comparisons also justifies the utility of external large parallel corpus for En-XX. #7 was substantially better than #5 simply because the pre-final model in #7 was also trained on multilingual data, unlike the one in #5.

Irrespective of the external parallel corpus for En-XX, the three-stage fine-tuned model (#7) achieved the highest BLEU scores for all the seven target languages, consistently outperforming the other models, with gains over the best scores of 0.14 (Id) to 0.87 (Bn) BLEU points with the

[14] We suspected that the gain is partially owing to the use of the larger vocabularies of these models. However, we only obtained translations of degraded quality with a larger vocabulary size of #1, presumably due to the scarce parallel data for En-YY language pair.

IWSLT En-Zh corpus and 0.23 (Tl) to 1.67 (Id) BLEU points with the KFTT En-Ja corpus. As mentioned in Section 4.2, these models do not introduce any new source sentences compared to the models #2 to #4. Therefore, we can safely conclude that the gain is due to multilingualism.

## 5 Discussion

In this section, we discuss the effect of the two types of corpora, i.e., external large parallel corpora and small multi-parallel corpora.

### 5.1 Does the Nature of Corpora Matter?

When using only one En-YY language pair (#2 to #4), only for the three-stage fine-tuned model (#4), the KFTT En-Ja corpus showed slight but consistent superiority (for Tl, Id, Km, and Vi) over the IWSLT En-Zh corpus. In contrast, when all the target languages were exploited (#5–#7), the KFTT En-Ja corpus consistently helped achieve significantly better results than the IWSLT En-Zh corpus. The reason could be three-fold: corpus size, overlap or similarity of helping target language with the target language of interest, and proximity of domain.

En-Ja translation using KFTT En-Ja corpus as a helping data was better by 2.52 BLEU points than when IWSLT En-Zh corpus was used. The difference in BLEU is significant, but it could be because the En-Ja corpus is more than twice the size of the En-Zh corpus. Whereas it is possible that En-Ja translation improves from the overlapping vocabularies of Chinese and Japanese, the performance of other languages also improves despite no vocabulary overlap. Consequently, we believe that translation into a low-resource target language might benefit when the language is also target side of the helping corpus but multistage fine-tuning does not require such overlap and instead shows optimal performance when it leverages multilingualism during stage-wise tuning.

In the future, we will test the above hypotheses thoroughly via some controlled experiments by using, for instance, larger scale multi-parallel corpora (Imamura and Sumita, 2018) and varieties of helping corpora.

### 5.2 How Does Multilingualism Help?

The ALT corpus is multi-parallel and merely comprises of the same sentences in multiple languages. Although we used seven target languages, we did not introduce new English sentences to the source side. Table 1 shows that some languages are better translated (En-Vi) than others (En-Bn). We speculate that the representations for En-Vi might be better learned than those for En-Bn. When learning all language pairs jointly, the representations of sentences with the same meaning tend to be similar. As such, the representations for En-Bn will be similar to those of En-Vi and hence the former might benefit from the potentially better representations of the latter, leading to improvements in translation quality for the former. This might be how multilingualism is responsible for the improvement in translation quality.

The results also shed light on the value of multi-parallel corpora despite their resource-scarce nature. In a practical situation, a collection of existing multiple bilingual corpora may potentially improve translation quality. Nevertheless, when we create new parallel data, multi-parallel fashion can be a less expensive option, since adding new language leads to parallel corpora between that language and all the other languages in the corpora.

## 6 Conclusion

We explored the use of small multi-parallel corpora and large helping parallel corpora for one-to-many NMT, which translates English to multiple languages. We proposed multistage fine-tuning and confirmed that it works significantly better than training models via fewer fine-tuning stages despite the resource-scarce and content-redundant nature of the multi-parallel corpora, thereby highlighting the usefulness of multilingualism behind the improvement in translation quality.

In the future, we will explore the utility of multistage fine-tuning for many-to-one and many-to-many NMT. We will also try to explicitly determine the impact of corpora sizes, language similarity, and domain on our approach, and propose further improvements according to our findings.

## Acknowledgments

This work was partially conducted under the program “Research and Development of Enhanced Multilingual and Multipurpose Speech Translation Systems” of the Ministry of Internal Affairs and Communications (MIC), Japan.

## References


Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, San Diego, USA.

Ondřej Bojar, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Philipp Koehn, and Christof Monz. 2018. Findings of the 2018 Conference on Machine Translation (WMT18). In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 272–303, Belgium, Brussels.

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. Wit$^3$: Web inventory of transcribed and translated talks. In *Proceedings of the 16th Conference of the European Association for Machine Translation (EAMT)*, pages 261–268, Trento, Italy.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, Roldano Cattoni, and Marcello Federico. 2015. The IWSLT 2015 evaluation campaign. In *Proceedings of the Twelfth International Workshop on Spoken Language Translation (IWSLT)*, pages 2–14, Da Nang, Vietnam.

Christos Christodouloupoulos and Mark Steedman. 2015. A massively parallel corpus: the Bible in 100 languages. *Language Resources and Evaluation*, 49(2):375–395.

Chenhui Chu, Raj Dabre, and Sadao Kurohashi. 2017. An empirical comparison of domain adaptation methods for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 385–391, Vancouver, Canada.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1723–1732, Beijing, China.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 866–875, San Diego, USA.

Jiatao Gu, Hany Hassan, Jacob Devlin, and Victor O.K. Li. 2018a. Universal neural machine translation for extremely low resource languages. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 344–354.

Jiatao Gu, Yong Wang, Yun Chen, Victor O. K. Li, and Kyunghyun Cho. 2018b. Meta-learning for low-resource neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3622–3631. Association for Computational Linguistics.

Kenji Imamura and Eiichiro Sumita. 2018. Multilingual parallel corpus for Global Communication Plan. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC)*, pages 3453–3458, Miyazaki, Japan.

Aizhan Imankulova, Raj Dabre, Atsushi Fujita, and Kenji Imamura. 2019. Exploiting out-of-domain parallel data through multilingual transfer learning for low-resource neural machine translation. In *Proceedings of Machine Translation Summit XVII Volume 1: Research Track*, pages 128–139, Dublin, Ireland.

Girish Nath Jha. 2010. The TDIL program and the Indian Language Corpora Initiative (ILCI). In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC)*, pages 982–985, Valletta, Malta.

Melvin Johnson, Mike Schuster, Quoc Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Conference Proceedings: the tenth Machine Translation Summit*, pages 79–86, Phuket, Thailand.

Surafel Melaku Lakew, Aliia Erofeeva, Matteo Negri, Marcello Federico, and Marco Turchi. 2018. Transfer learning in multilingual neural machine translation with dynamic vocabulary. In *Proceedings of the 15th International Workshop on Spoken Language Translation*, pages 54–61, Bruges, Belgium.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Proceedings of the 30th Neural Information Processing Systems Conference (NIPS)*, pages 6294–6305, Long Beach, USA.

V. Rudra Murthy, Anoop Kunchukuttan, and Pushpak Bhattacharyya. 2018. Addressing word-order divergence in multilingual neural machine translation for extremely low resource languages. *CoRR*, abs/1811.00383.

Toan Q. Nguyen and David Chiang. 2017. Transfer learning across low-resource, related languages for neural machine translation. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 296–301, Taipei, Taiwan.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL)*, pages 311–318, Philadelphia, USA.

Hammam Riza, Michael Purwoadi, Gunarso, Teduh Uliniansyah, Aw Ai Ti, Sharifah Mahani Aljunied, Luong Chi Mai, Vu Tat Thang, Nguyen Phuong Thai, Vichet Chea, Rapid Sun, Sethserey Sam, Sopheap Seng, Khin Mar Soe, Khin Thandar Nwet, Masao Utiyama, and Chenchen Ding. 2016. Introduction of the Asian Language Treebank. In *Proceedings of the 2016 Conference of the Oriental Chapter of International Committee for Coordination and Standardization of Speech Databases and Assessment Technique (O-COCOSDA)*, pages 1–6, Bali, Indonesia.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL), Volume 1: Long Papers*, pages 86–96, Berlin, Germany.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th Neural Information Processing Systems Conference (NIPS)*, pages 3104–3112, Montréal, Canada.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 30th Neural Information Processing Systems Conference (NIPS)*, pages 5998–6008, Long Beach, USA.

Michał Ziemski, Marcin Junczys-Dowmunt, and Bruno Pouliquen. 2016. The United Nations parallel corpus v1.0. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC)*, pages 3530–3534, Portorož, Slovenia.

Barret Zoph and Kevin Knight. 2016. Multi-source neural translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 30–34, San Diego, USA.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1568–1575, Austin, USA.

# Unsupervised Domain Adaptation for Neural Machine Translation with Domain-Aware Feature Embeddings

**Zi-Yi Dou, Junjie Hu, Antonios Anastasopoulos, Graham Neubig**
Language Technologies Institute, Carnegie Mellon University
{zdou, junjieh, aanastas, gneubig}@cs.cmu.edu

## Abstract

The recent success of neural machine translation models relies on the availability of high quality, in-domain data. Domain adaptation is required when domain-specific data is scarce or nonexistent. Previous unsupervised domain adaptation strategies include training the model with in-domain copied monolingual or back-translated data. However, these methods use generic representations for text regardless of domain shift, which makes it infeasible for translation models to control outputs conditional on a specific domain. In this work, we propose an approach that adapts models with domain-aware feature embeddings, which are learned via an auxiliary language modeling task. Our approach allows the model to assign domain-specific representations to words and output sentences in the desired domain. Our empirical results demonstrate the effectiveness of the proposed strategy, achieving consistent improvements in multiple experimental settings. In addition, we show that combining our method with back translation can further improve the performance of the model.[1]

## 1 Introduction

While neural machine translation (NMT) systems have proven to be effective in scenarios where large amounts of in-domain data are available (Gehring et al., 2017; Vaswani et al., 2017; Chen et al., 2018), they have been demonstrated to perform poorly when the test domain does not match the training data (Koehn and Knowles, 2017). Collecting large amounts of parallel data in all possible domains we are interested in is costly, and in many cases impossible. Therefore, it is essential to explore effective methods to train models that generalize well to new domains.

[1]Our code is publicly available at: https://github.com/zdou0830/DAFE.

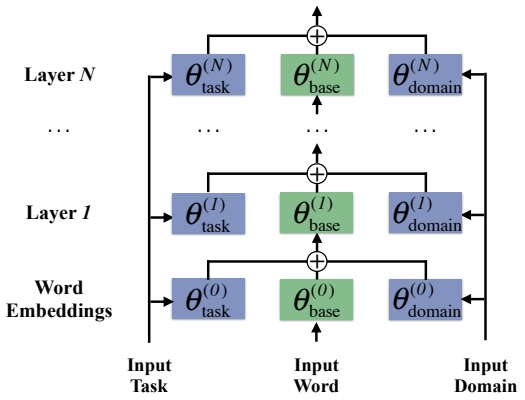


Figure 1: Main architecture of DAFE. Embedding learners generate domain- and task-specific features at each layer, which are then integrated into the output of the base network.

Domain adaptation for neural machine translation has attracted much attention in the research community, with the majority of work focusing on the supervised setting where a small amount of in-domain data is available (Luong and Manning, 2015; Freitag and Al-Onaizan, 2016; Chu et al., 2017; Vilar, 2018). An established approach is to use domain tags as additional input, with the domain representations learned over parallel data (Kobus et al., 2017). In this work, we focus on *unsupervised* adaptation, where there are no in-domain parallel data available. Within this paradigm, Currey et al. (2017) copy the in-domain monolingual data from the target side to the source side and Sennrich et al. (2016a) concatenate back-translated data with the original corpus. However, these methods learn generic representations for all the text, as the learned representations are shared for all the domains and synthetic and natural data are treated equally. Sharing embeddings may be sub-optimal as data from different domains are inherently different. This problem is exacerbated



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1417–1422,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

when words have different senses in different domains.

In this work, we propose a method of *Domain-Aware Feature Embedding* (DAFE) that performs unsupervised domain adaptation by disentangling representations into different parts. Because we have no in-domain parallel data, we learn DAFEs via an auxiliary task, namely language modeling. Specifically, our proposed model consists of a base network, whose parameters are shared across settings, as well as both domain and task embedding learners. By separating the model into different components, DAFE can learn representations tailored to specific domains and tasks, which can then be utilized for domain adaptation.

We evaluate our method in a Transformer-based NMT system (Vaswani et al., 2017) under two different data settings. Our approach demonstrates consistent improvements of up to 5 BLEU points over unadapted baselines, and up to 2 BLEU points over strong back-translation models. Combining our method with back translation can further improve the performance of the model, suggesting the orthogonality of the proposed approach and methods that rely on synthesized data.

## 2 Methods

In this section, we first illustrate the architecture of DAFE, then describe the overall training strategy.

### 2.1 Architecture

DAFE disentangles hidden states into different parts so that the network can learn representations for particular domains or tasks, as illustrated in Figure 1. Specifically, it consists of three parts: a *base network* with parameters $\theta_{\text{base}}$ that learns common features across different tasks and domains, a *domain-aware feature embedding learner* that generates embeddings $\theta^{\tau}_{\text{domain}}$ given input domain $\tau$ and a *task-aware feature embedding learner* that outputs task representations $\theta^{\gamma}_{\text{task}}$ given input task $\gamma$. The final outputs for each layer are obtained by a combination of the base network outputs and feature embeddings.

The base network is implemented in the encoder-decoder framework (Sutskever et al., 2014; Cho et al., 2014). Both the task and domain embedding learners directly output feature embeddings with look-up operations.

In this work, the domain-aware embedding learner learns domain representations $\theta^{in}_{\text{domain}}$ and $\theta^{out}_{\text{domain}}$ from in-domain and out-of-domain data respectively, and the task-aware embedding leaner learns task embeddings $\theta^{mt}_{\text{task}}$ and $\theta^{lm}_{\text{task}}$ for machine translation and language modeling.

**Algorithm 1** Training Strategy

**while** $\theta_{\text{base}}, \theta_{\text{domain}}, \theta_{\text{task}}$ have not converged **do**
  Sample $\{(C(\mathbf{y}), \mathbf{y})\}$ from $Y^{in}$
  Train $\{\theta_{\text{base}}, \theta^{in}_{\text{domain}}, \theta^{lm}_{\text{task}}\}$ with Eqn. 2
  Sample $\{(C(\mathbf{y}), \mathbf{y})\}$ from $Y^{out}$
  Train $\{\theta_{\text{base}}, \theta^{out}_{\text{domain}}, \theta^{lm}_{\text{task}}\}$ with Eqn. 2
  Sample $\{(\mathbf{x}, \mathbf{y})\}$ from $(X^{out}, Y^{out})$
  Train $\{\theta_{\text{base}}, \theta^{out}_{\text{domain}}, \theta^{mt}_{\text{task}}\}$ with Eqn. 1
**end while**

The feature embeddings are generated at *each* encoding layer (including the source word embedding layer) and have the same size as the hidden states of the base model. It should be noted that feature embedding learners generate different embeddings at different layers.

Formally, given a specific domain $\tau$ and task $\gamma$, the output of the $l$-th encoding layer $\mathbf{H}^{(l)}_e$ would be:

$$\mathbf{H}^{(l)}_e = \textsc{Layer}_e(\mathbf{H}^{(l-1)}_e; \theta^{(l)}_{\text{base}}) + \theta^{\gamma,(l)}_{\text{domain}} + \theta^{\tau,(l)}_{\text{task}},$$

where $\theta^{\gamma,(l)}_{\text{domain}}$ and $\theta^{\tau,(l)}_{\text{task}}$ are single vectors and $\textsc{Layer}_e(\cdot)$ can be any layer encoding function, such as an LSTM (Hochreiter and Schmidhuber, 1997) or Transformer (Vaswani et al., 2017).

In this paper, we adopt a simple, add operation to combine outputs of different parts which already achieves satisfactory performance as shown in Section 3. We leave investigating more sophisticated combination strategies for future work.

### 2.2 Training Objectives

In the unsupervised domain adaptation setting, we assume access to an out-of-domain parallel training corpus $(X^{out}, Y^{out})$ and target-language in-domain monolingual data $Y^{in}$.

**Neural machine translation.** Our target task is machine translation. Both the base network and embedding learners are jointly trained with the objective:

$$\max_{\theta} \sum_{(\mathbf{x},\mathbf{y})\in(X^{out},Y^{out})} \log p(\mathbf{y}|\mathbf{x};\theta), \qquad (1)$$

where $\theta = \{\theta_{\text{base}}, \theta^{mt}_{\text{task}}, \theta^{out}_{\text{domain}}\}$.

| Method | German-English | | | | | | Czech-English | | | German-English | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LAW | | MED | | IT | | WMT | | | WMT | | |
| | MED | IT | LAW | IT | LAW | MED | TED | LAW | MED | TED | LAW | MED |
| (1) Baseline | 13.25 | 5.22 | 4.55 | 3.15 | 4.29 | 7.05 | 24.30 | 28.22 | 15.45 | 28.15 | 24.61 | 26.75 |
| *Data-Centric Methods* | | | | | | | | | | | | |
| (2) Copy | 19.23 | 7.57 | 6.01 | 5.89 | 5.11 | 11.15 | 26.44 | 36.49 | 22.73 | 29.79 | 26.17 | 29.33 |
| (3) Back | 22.53 | 11.34 | 7.62 | 7.02 | 8.06 | 14.56 | 32.70 | 42.08 | 30.45 | 34.46 | 30.24 | 33.16 |
| *Model-Centric Methods* | | | | | | | | | | | | |
| (4) DAFE w/o Embed | 23.44 | 6.97 | 9.21 | 8.34 | 8.09 | 16.34 | 26.63 | 35.86 | 23.44 | 29.88 | 27.10 | 32.15 |
| (5) DAFE | 24.23 | 8.59 | 9.87 | 8.44 | 8.61 | 17.50 | 28.09 | 38.89 | 26.05 | 30.88 | 27.77 | 32.48 |
| *Combining Data-Centric and Model-Centric Methods* | | | | | | | | | | | | |
| (6) Back + DAFE | 25.34 | 13.55 | 9.60 | 11.20 | 9.60 | 17.25 | **33.18** | 44.06 | 34.24 | 34.57 | 30.72 | 35.48 |
| (7) Back-DAFE | 26.47 | 13.75 | 11.90 | 9.47 | 10.60 | 18.04 | 32.51 | 43.33 | 35.45 | 34.57 | 30.93 | 37.66 |
| (8) Back-DAFE + DAFE | **26.96** | **15.41** | **14.28** | **13.03** | **11.67** | **21.30** | 33.02 | **44.36** | **37.48** | **34.89** | **31.46** | **38.79** |

Table 1: Translation accuracy (BLEU) under different settings. The second and third rows list source and target domains respectively. "DAFE w/o Embed" denotes DAFE without embedding learners and "Back-DAFE" denotes back-translation by target-to-source model trained with DAFE. DAFE outperforms other approaches when adapting between domains (row 1-5, column 2-7) and is complementary to back-translation (row 6-8).

**Language modeling.** We choose masked language modeling (LM) as our auxiliary task. Following Lample et al. (2018a,b), we create corrupted versions $C(\mathbf{y})$ for each target sentence $\mathbf{y}$ by randomly dropping and slightly shuffling words. During training, gradient ascent is used to maximize the objective:

$$\max_{\theta} \sum_{\mathbf{y} \in \{Y^{in} \cup Y^{out}\}} \log p(\mathbf{y}|C(\mathbf{y}); \theta), \quad (2)$$

where $\theta = \{\theta_{\text{base}}, \theta^{lm}_{\text{task}}, \theta^{out}_{\text{domain}}\}$ for out-of-domain data and $\{\theta_{\text{base}}, \theta^{lm}_{\text{task}}, \theta^{in}_{\text{domain}}\}$ for in-domain data.

**Training strategy.** Our training strategy is shown in Algorithm 1. The ultimate goal is to learn a set of parameters $\{\theta_{\text{base}}, \theta^{in}_{\text{domain}}, \theta^{mt}_{\text{task}}\}$ for in-domain machine translation. While out-of-domain parallel data allows us to train $\{\theta_{\text{base}}, \theta^{out}_{\text{domain}}, \theta^{mt}_{\text{task}}\}$, the monolingual data help the model learn both $\theta^{in}_{\text{domain}}$ and $\theta^{out}_{\text{domain}}$.

## 3 Experiments

### 3.1 Setup

**Datasets.** We validate our models in two different data settings. First, we train on the law, medical and IT datasets of the German-English OPUS corpus (Tiedemann, 2012) and test our methods' ability to adapt from one domain to another. The dataset contain 2K development and test sentences in each domain, and about 715K, 1M and 337K training sentences respectively. These datasets are relatively small and the domains are quite distant from each other. In the second setting, we adapt models trained on the general-domain WMT-14 datasets into both the TED (Duh, 2018) and law, medical OPUS datasets. For this setting, we consider two language pairs, namely Czech and German to English. The Czech-English and German-English datasets consist of 1M and 4.5M sentences and the development and test sets contain about 2K sentences.

**Models.** We implement DAFE on top of the Transformer model. Both the encoder and decoder consist of 4 layers and the hidden size is set to 512. Byte-pair encoding (Sennrich et al., 2016b) is employed to process training data into subwords for a final shared vocabulary size of 50K.

**Baselines.** We compare our methods with two baseline models: 1) The copied monolingual data model (Currey et al., 2017) which copies target in-domain monolingual data to the source side; 2) Back-translation (Sennrich et al., 2016a) which enriches the training data by generating synthetic in-domain parallel data via a target-to-source NMT model. We characterize the two baselines as *data-centric* methods as they rely on synthesized data. In contrast, our approach is *model-centric* as we mainly focus on modifying the model architecture. We also perform an ablation study by removing the embedding learners (denoted as "DAFE w/o Embed") and the model will just perform multi-task learning.

### 3.2 Main Results

**Adapting between domains.** As shown in the first 6 results columns of Table 1, the unadapted baseline model (row 1) performs poorly when adapting between domains. The copy method

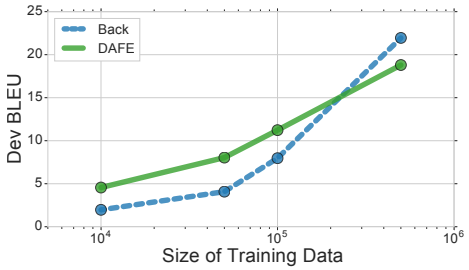


Figure 2: DAFE outperforms back-translation in low-resource scenarios.

(row 2) and back translation (row 3) both improve the model significantly, with back-translation being a better alternative to copying. DAFE (row 5) achieves superior performance compared to back-translation, with improvements of up to 2 BLEU points. Also, removing the embedding learners leads to degraded performance (row 4), indicating the necessity of their existence.

**Adapting from a general to a specific domain.** In the second data setting (last 6 columns of Table 1), with relatively large amounts of general-domain datasets, back-translation achieves competitive performance. In this setting, DAFE improves the unadapted baseline significantly but it does not outperform back-translation. We hypothesize this is because the quality of back-translated data is relatively good.

### 3.3 Combining DAFE with Back-Translation

We conjecture that DAFE is complementary to the data-centric methods. We attempt to support this intuition by combining DAFE with back-translation (the best data-centric approach). We try three different strategies to combine DAFE with back-translation, outlined in Table 1.

Simply using back-translated data to train DAFE (row 6) already achieves notable improvements of up to 4 BLEU points. We can also generate back-translated data using target-to-source models trained with DAFE, with which we train the forward model (Back-DAFE, row 7). By doing so, the back-translated data will be of higher quality and thus the performance of the source-to-target model can be improved. The overall best strategy is to use Back-DAFE to generate synthetic in-domain data and train the DAFE model with the back-translated data (Back-DAFE+DAFE, row 8). Across almost all adaptation settings, Back-DAFE+DAFE leads to higher translation quality, as per our intuition. An advantage of this setting is that the back-translated data allow us to learn $\theta^{in}_{\text{domain}}$ with the translation task.

| **Embedding** | MED dev | MED test | IT dev | IT test |
|---|---|---|---|---|
| MED | 42.06 | 34.63 | 4.13 | 4.80 |
| IT | 36.96 | 30.09 | 7.54 | 8.44 |

Table 2: Providing mismatched domain embeddings leads to degraded performance.

| | |
|---|---|
| **Reference** | please report this bug to the developers . |
| **MED-embed** | please report this to the EMEA . |
| **IT-embed** | please report this bug to the developers . |
| **Reference** | for intramuscular use . |
| **MED-embed** | for intramuscular use . |
| **IT-embed** | for the use of the product . |

Table 3: Controlling the output domain by providing different domain embeddings. We use compare-mt (Neubig et al., 2019) to select examples.

### 3.4 Analysis

**Low-resource scenarios.** One advantage of DAFE over back-translation is that we do not need a good target-to-source translation model, which can be difficult be acquire in low-resource scenarios. We randomly sample different amounts of training data and evaluate the performance of DAFE and back-translation on the development set. As shown in Figure 2, DAFE significantly outperforms back-translation in data-scarce scenarios, as low quality back-translated data can actually be harmful to downstream performance.

**Controlling the output domain.** An added perk of our model is the ability to control the output domain by providing the desired domain embeddings. As shown in Table 2, feeding mismatched domain embeddings leads to worse performance. Examples in Table 3 further suggest the model with medical embeddings as input can generate domain-specific words like "EMEA" (European Medicines Evaluation Agency) and "intramuscular", while IT embeddings encourage the model to generate words like "bug" and "developers".

## 4 Related Work

Most previous domain adaptation work for NMT focus on the setting where a small amount of in-domain data is available. Continued training (Luong and Manning, 2015; Freitag and Al-Onaizan, 2016) methods first train an NMT model on out-of-domain data and then fine-tune it on the in-domain data. Similar to our work, Kobus et al.

(2017) propose to use domain tags to control the output domain, but it still needs a in-domain parallel corpus and our architecture allows more flexible modifications than just adding additional tags.

Unsupervised domain adaptation techniques for NMT can be divided into data- and model-centric methods (Chu and Wang, 2018). Data-centric approaches mainly focus on selecting or generating the domain-related data using existing in-domain monolingual data. Both the copy method (Currey et al., 2017) and back-translation (Sennrich et al., 2016a) are representative data-centric methods. In addition, Moore and Lewis (2010); Axelrod et al. (2011); Duh et al. (2013) use LMs to score the out-of-domain data, based on which they select data similar to in-domain text. Model-centric methods have not been fully investigated yet. Gulcehre et al. (2015) propose to fuse LMs and NMT models, but their methods require querying two models during inference and have been demonstrated to underperform the data-centric ones (Chu et al., 2018). There are also work on adaptation via retrieving sentences or n-grams in the training data similar to the test set (Farajian et al., 2017; Bapna and Firat, 2019). However, it can be difficult to find similar parallel sentences in domain adaptation settings.

## 5 Conclusion

In this work, we propose a simple yet effective unsupervised domain adaptation technique for neural machine translation, which adapts the model by domain-aware feature embeddings learned with language modeling. Experimental results demonstrate the effectiveness of the proposed approach across settings. In addition, analysis reveals that our method allows us to control the output domain of translation results. Future work include designing more sophisticated architectures and combination strategies as well as validating our model on other language pairs and datasets.

## Acknowledgements

We are grateful to Xinyi Wang and anonymous reviewers for their helpful suggestions and insightful comments. We also thank Zhi-Hao Zhou, Shuyan Zhou and Anna Belova for proofreading the paper.

This material is based upon work generously supported partly by the National Science Foundation under grant 1761548 and the Defense Advanced Research Projects Agency Information Innovation Office (I2O) Low Resource Languages for Emergent Incidents (LORELEI) program under Contract No. HR0011-15-C0114. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

## References

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. 2011. Domain adaptation via pseudo in-domain data selection. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Ankur Bapna and Orhan Firat. 2019. Non-parametric adaptation for neural machine translation. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, et al. 2018. The best of both worlds: Combining recent advances in neural machine translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Chenhui Chu, Raj Dabre, and Sadao Kurohashi. 2017. An empirical comparison of domain adaptation methods for neural machine translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Chenhui Chu, Raj Dabre, and Sadao Kurohashi. 2018. A comprehensive empirical comparison of domain adaptation methods for neural machine translation. *Journal of Information Processing*, 26:529–538.

Chenhui Chu and Rui Wang. 2018. A survey of domain adaptation for neural machine translation. In *International Conference on Computational Linguistics (COLING)*.

Anna Currey, Antonio Valerio Miceli Barone, and Kenneth Heafield. 2017. Copied monolingual data improves low-resource neural machine translation. In *Conference on Machine Translation (WMT)*.

Kevin Duh. 2018. The multitarget ted talks task. http://www.cs.jhu.edu/~kevinduh/a/multitarget-tedtalks/.

Kevin Duh, Graham Neubig, Katsuhito Sudoh, and Hajime Tsukada. 2013. Adaptation data selection using neural language models: Experiments in machine translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

M Amin Farajian, Marco Turchi, Matteo Negri, and Marcello Federico. 2017. Multi-domain neural machine translation through unsupervised adaptation. In *Conference on Machine Translation (WMT)*.

Markus Freitag and Yaser Al-Onaizan. 2016. Fast domain adaptation for neural machine translation. *arXiv preprint arXiv:1612.06897*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *International Conference on Machine Learning (ICML)*.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. *arXiv preprint arXiv:1503.03535*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Catherine Kobus, Josep Crego, and Jean Senellart. 2017. Domain control for neural machine translation. In *International Conference Recent Advances in Natural Language Processing (RANLP)*.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Workshop on Neural Machine Translation (WMT)*.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018a. Unsupervised machine translation using monolingual corpora only. In *International Conference on Learning Representations (ICLR)*.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, et al. 2018b. Phrase-based & neural unsupervised machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Minh-Thang Luong and Christopher D Manning. 2015. Stanford neural machine translation systems for spoken language domains. In *International Workshop on Spoken Language Translation (IWSLT)*.

Robert C. Moore and William Lewis. 2010. Intelligent selection of language model training data. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Graham Neubig, Zi-Yi Dou, Junjie Hu, Paul Michel, Danish Pruthi, and Xinyi Wang. 2019. compare-mt: A tool for holistic comparison of language generation systems. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL) Demo Track*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in opus. In *International Conference on Language Resources and Evaluation (LREC)*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*.

David Vilar. 2018. Learning hidden unit contribution for adapting neural machine translation models. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

# A Regularization-based Framework for Bilingual Grammar Induction*

**Yong Jiang**[◇], **Wenjuan Han**[†], **Kewei Tu**[†]
[†]School of Information Science and Technology, ShanghaiTech University, Shanghai, China
[◇]Alibaba Group
yongjiang.jy@alibaba-inc.com
{hanwj,tukw}@shanghaitech.edu.cn

## Abstract

Grammar induction aims to discover syntactic structures from unannotated sentences. In this paper, we propose a framework in which the learning process of the grammar model of one language is influenced by knowledge from the model of another language. Unlike previous work on multilingual grammar induction, our approach does not rely on any external resources, such as parallel corpora, word alignments or linguistic phylogenetic trees. We propose three regularization methods that encourage similarity between model parameters, dependency edge scores, and parse trees respectively. We deploy our methods on a state-of-the-art unsupervised discriminative parser and evaluate it on both transfer grammar induction and bilingual grammar induction. Empirical results on multiple languages show that our methods outperform strong baselines.

## 1 Introduction

Syntactic parsing is an important task in natural language processing. Supervised parsing requires manual labeling of gold parse trees, which is a very labor-intensive task. On the other hand, unsupervised parsing (a.k.a. grammar induction) does not require labeled data and can make use of large amounts of unlabeled data that are freely available. However, grammar induction is very challenging and its accuracy is still far below that of supervised parsing. To compensate the lack of supervision in grammar induction, some previous work considers multilingual grammar induction, i.e., simultaneously learning grammars of multiple languages (Snyder et al., 2009; Berg-Kirkpatrick and Klein, 2010; Liu et al., 2013). Existing multilingual approaches require external resources such as parallel corpora, word alignments, and linguistic phylogenetic trees.

| Language | German | English | Spanish | French | Indonesian |
|---|---|---|---|---|---|
| Code | DE | EN | ES | FR | ID |
| C-MST | **60.2** | 62.3 | **68.8** | **72.3** | **69.7** |
| D-Tran | 59.9 | – | 65.3 | 67.8 | 45.7 |
| Δ | -0.3 | – | -3.5 | -4.5 | -24.0 |
| **Language** | **Italian** | **Japanese** | **Korean** | **Portuguese** | **Swedish** |
| Code | IT | JA | KO | PTBR | SV |
| C-MST | 64.3 | **57.5** | **59.0** | **68.3** | 66.2 |
| D-Tran | 63.1 | 54.6 | 50.0 | 66.2 | **67.8** |
| Δ | -1.2 | -2.9 | -9.0 | -2.1 | +1.6 |

Table 1: Directed dependency accuracy (DDA) on the universal treebanks with universal POS tags, on sentences of length $\leq 10$. C-MST denotes the original Convex-MST model. D-Tran denotes direct transfer. $\Delta$ refers to the difference between C-MST and D-Tran.

In this paper, we aim at bilingual grammar induction without external resource. We are motivated by our observation that learning the unsupervised Convex-MST model (Grave and Elhadad, 2015) on the English corpus and then directly applying it to parse other languages produces surprisingly good results (Table 1). From the table, we can see that even with this simplistic method (which we call *direct transfer*), the dependency accuracy on each language is often very close to the accuracy of the model specifically trained on the corpus of that language. For the Swedish language, the accuracy of direct transfer is even better than that of the specifically trained model. This surprising result suggests that grammars of different languages, even those from different language families (e.g., English and Japanese), may have non-trivial similarity that can be helpful in bilingual grammar induction.

Inspired by this observation, we propose a regularization-based framework to bilingual grammar induction that encourage knowledge sharing between models learned on a language pair. We build our framework on top of Convex-MST, a state-of-the-art unsupervised dependency parser,

* Yong Jiang contributed to this work when at ShanghaiTech University. Kewei Tu is the corresponding author.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1423–1428,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

and propose three regularization terms that encourage similarity between model parameters, edge scores, and parse trees respectively. We test our methods on ten languages on the tasks of transfer grammar induction and bilingual grammar induction and show that our methods can achieve a significant boost over strong baselines.

## 2 Background

### 2.1 Unsupervised Dependency Parsing

Dependency parsing is the task of mapping an input sentence $\mathbf{x} = x_1, x_2, ..., x_n$ of length $n$ to an output dependency structure $\mathbf{y}$. A dummy root $x_0$ is typically added at the beginning of the sentence to denote the head of the dependency tree. There are several approaches to represent the parse tree $\mathbf{y}$. In transition based dependency parsers, the dependency tree can be regarded as a sequence of actions. In graph based dependency parsers, the dependency tree can be represented as a spanning tree in the graph. In chart based parsers (a.k.a., grammar based parsers), the dependency tree is denoted as a set of grammar rules. In unsupervised graph based dependency parsers, since gold trees are not available, carefully designed models and objective functions are required for learning a dependency parser. Regardless of model architectures, current unsupervised dependency models usually use the following form of objective function,

$$J(\mathbf{w}; \mathcal{X}) = \sum_{\mathbf{x} \in \mathcal{X}} \mathrm{O}_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} \Big( D(\mathbf{w}, \mathbf{x}, \mathbf{y}) + R(\mathbf{w}) \Big)$$

where $\mathcal{Y}$ is the set of all possible dependency tree, $\mathbf{w}$ is the model parameter, $\mathcal{X}$ is the unlabeled training corpus, $D$ is the measurement between the parse $\mathbf{y}$ and model prediction on sentence $\mathbf{x}$, $R(\mathbf{w})$ is the regularization term of parameter $\mathbf{w}$, $\mathrm{O} \in \{\min, \sum\}$ is an operator. Table 2 shows the choices of $\mathrm{O}$, $D$ and $R$ for several widely used models.

### 2.2 Graph based Dependency Parsing

In this paper, we focus on graph based dependency parsers, though we believe that our approaches can be generalized to other types of parsers. Previous work on unsupervised graph based dependency parsing utilizes the autoencoder structure (Cai et al., 2017) or the discriminative clustering techniques (Grave and Elhadad, 2015).

Following (McDonald et al., 2005), we can use a discriminative model for dependency parsing with first order factorization such that the score of a dependency tree $\mathbf{y}$ is the sum of the scores of its dependency edges. The score of an edge from word $h$ to word $m$ , $\mathbf{s}_{\mathbf{w}}(\mathbf{x}, h, m)$, can be computed as the inner product of a feature vector $\mathbf{f}(\mathbf{x}, h, m)$ and a parameter vector $\mathbf{w}$. The optimal dependency tree for sentence $\mathbf{x}$ be discovered in polynomial time (Eisner, 1996; McDonald et al., 2005).

## 3 Bilingual Knowledge Sharing

Given non-parallel corpora of two languages $\mathcal{X}_s$ and $\mathcal{X}_t$, our goal is to learn two models with parameters $\mathbf{w}_s$ and $\mathbf{w}_t$ for the two languages. The simplest learning objective function is,

$$J(\mathbf{w}_s, \mathbf{w}_t; \mathcal{X}_s, \mathcal{X}_t) = J(\mathbf{w}_s; \mathcal{X}_s) + J(\mathbf{w}_t; \mathcal{X}_t)$$

which contains no interaction between the two models.

As suggested by our empirical observation in Table 1, the model of one language may provide a useful inductive bias in learning the model of another language. Note that given a sentence, a graph-based dependency parser has three levels of representations: the model parameters, the scores of dependency edges computed from the parameters, and the parse tree computed from the edge scores. Therefore, we propose three different regularization terms to effectively encourage similarity of the two models. An example is shown in Figure 1.

**Regularization of Weight Parameters (W-Reg)** Motivated by the approach of Berg-Kirkpatrick and Klein (2010), we encourage the similarity between the two weight parameters $\mathbf{w}_s$ and $\mathbf{w}_t$ measured by l2 norm distance:

$$J(\mathbf{w}_s, \mathbf{w}_t; \mathcal{X}_s, \mathcal{X}_t) = \\ J(\mathbf{w}_s; \mathcal{X}_s) + J(\mathbf{w}_t; \mathcal{X}_t) + \lambda ||\mathbf{w}_s - \mathbf{w}_t||_2^2$$

in which $\lambda$ is a hyper-parameter.

**Regularization on Edge Scores (E-Reg)** Directly encouraging weight similarity might result in an inductive bias that is too strong, because the difference between the two languages (e.g., different word orders) may lead to different meanings of each feature dimension. Therefore, we propose to encourage similarity between the scores computed by the two models for each dependency edge of

| Parsers | O | $D$ | $R$ |
|---|---|---|---|
| DMV (Klein and Manning, 2004) | $\sum$ | negative log likelihood | - |
| Convex-MST (Grave and Elhadad, 2015) | min | $\ell_2$ distance | $\ell_2$ norm |
| LC-DMV (Noji et al., 2016) | $\sum$ | negative log likelihood | $\ell_2$ norm |
| NDMV (Jiang et al., 2016) | $\sum$, min | negative log likelihood | - |
| CRFAE (Cai et al., 2017) | min | negative conditional log likelihood | $\ell_1$ norm |
| D-NDMV (Han et al., 2019) | $\sum$, min | negative (conditional) log likelihood | - |

Table 2: Notations of several widely used models.

each sentence, which can be seen as a soft version of weight regularization.

$$J(\mathbf{w}_s, \mathbf{w}_t; \mathcal{X}_s, \mathcal{X}_t) = J(\mathbf{w}_s; \mathcal{X}_s) + J(\mathbf{w}_t; \mathcal{X}_t) + \lambda \sum_{\mathbf{x} \in \mathcal{X}'} \sum_{(h,m) \in \mathcal{G}(\mathbf{x})} ||\mathbf{s}_{\mathbf{w}_s}(\mathbf{x}, h, m) - \mathbf{s}_{\mathbf{w}_t}(\mathbf{x}, h, m)||_2^2$$

where $\mathcal{X}' = \mathcal{X}_s \cup \mathcal{X}_t$. $\mathcal{G}(\mathbf{x})$ is the weighted dependency graph of sentence $\mathbf{x}$.

**Regularization on Parse Trees (T-Reg)** Another alternative is to encourage similarity between the parse trees predicted by the two models. Motivated by the idea of knowledge distillation (Kim and Rush, 2016), in the learning objective of each model, we add a fourth term to encourage the parse tree to be close to the prediction of the other model. Below we show the objective function for $\mathbf{w}_s$.

$$J'(\mathbf{w}_s, \mathbf{w}_t; \mathcal{X}_s) = \sum_{\mathbf{x} \in \mathcal{X}_s} \mathbf{O}_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} \Big( D(\mathbf{w}_s, \mathbf{x}, \mathbf{y}) + \underbrace{\lambda D(\mathbf{w}_t, \mathbf{x}, \mathbf{y})}_{\textbf{T-Reg term}} + R(\mathbf{w}_s) \Big)$$

$$J'(\mathbf{w}_t, \mathbf{w}_s; \mathcal{X}_t) = \sum_{\mathbf{x} \in \mathcal{X}_t} \mathbf{O}_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} \Big( D(\mathbf{w}_t, \mathbf{x}, \mathbf{y}) + \underbrace{\lambda D(\mathbf{w}_s, \mathbf{x}, \mathbf{y})}_{\textbf{T-Reg term}} + R(\mathbf{w}_t) \Big)$$

$$J(\mathbf{w}_s, \mathbf{w}_t; \mathcal{X}_s, \mathcal{X}_t) = J'(\mathbf{w}_s, \mathbf{w}_t; \mathcal{X}_s) + J'(\mathbf{w}_t, \mathbf{w}_s; \mathcal{X}_t)$$

We apply these regularization method to the Convex-MST model (Grave and Elhadad, 2015). Our three objective functions can be optimized with coordinate descent in a similar way to Convex-MST. In each iteration, we first fix parse $\mathbf{y}$ for each training sentence and update parameters $\mathbf{w}_s$ and $\mathbf{w}_t$ by stochastic gradient descent; then we fix $\mathbf{w}_s$ and $\mathbf{w}_t$ and update $\mathbf{y}$ of each sentence by the Frank-Wolfe algorithm.

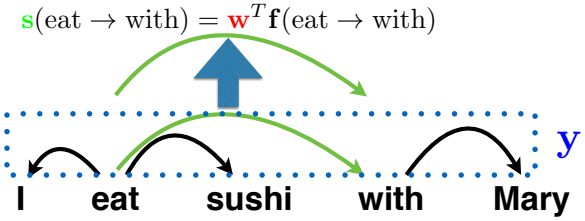


Figure 1: Three levels of representations of the parser: the parameter w, the edge score s, and the parse y.

While our three methods are applicable to any pair of languages, intuitively one may use weight regularization only for similar languages, and use edge regularization and tree regularization for an arbitrary language pair.

## 4 Experiments

To enable direct comparison with the Convex-MST model, we use the dataset used in their paper (Grave and Elhadad, 2015), the universal treebanks version 2.0[1], introduced by McDonald et al. (2013). The dataset contains ten different languages, which belong to five diverse families. In additional, we test our methods on twelve languages from the more recent UD Treebank 1.4[2], which is also used in previous grammar induction work (Jiang et al., 2017; Li et al., 2019). Following previous work, we train all the models on the gold POS tags of sentences no longer than ten. We tune hyper-parameters on the development dataset and report the DDA on sentences no longer than ten and all the sentences in the test dataset. As our goal is to investigate the benefits of our regularization methods, the two hyper-parameters $\mu$ and $\beta$

[1] https://github.com/ryanmcd/uni-dep-tb. The version is not consistent with recent releases of UD Treebanks.

[2] http://universaldependencies.org/

| CODE | C-MST | D-TRAN | W-REG | E-REG | T-REG |
|---|---|---|---|---|---|
| DE | 60.2 | -0.3 | -0.2 | **+0.2** | -0.2 |
| ES | 68.8 | -3.5 | -3.5 | **+0.8** | +0.3 |
| FR | 72.3 | -4.5 | -3.8 | **+0.3** | **+0.3** |
| ID | **69.7** | -24.0 | -21.4 | -0.6 | -1.2 |
| IT | 64.3 | -1.2 | -0.4 | +0.3 | **+1.2** |
| JA | 57.5 | -2.9 | -3.4 | +0.9 | **+2.3** |
| KO | 59.0 | -9.0 | -9.5 | +1.3 | **+1.9** |
| PTBR | 68.3 | -2.1 | -2.1 | +0.2 | **+0.3** |
| SV | 66.2 | +1.6 | +1.6 | **+2.7** | +2.6 |
| Avg | 65.14 | -5.10 | -4.74 | +0.68 | **+0.83** |
| Avg-All | 56.16 | -4.63 | -4.29 | +0.47 | **+0.56** |
| UD 1.4* | 52.83 | -0.30 | -1.92 | +1.06 | **+1.37** |

Table 3: Transfer grammar induction from English to the other languages. We show accuracies on the test sentences of length $\leq$ 10 (except the Avg-All row which shows the average accuracies on all the sentences). *: for the UD 1.4 dataset we show the average results.

of Convex-MST are tuned on the English development dataset and then fixed ($\mu = 0.1, \beta = 0.001$) while $\lambda$ is selected from $\{10, 5, 1, 5e-1, 1e-1, 5e-2, 1e-2, 5e-3, 1e-3, 1e-4\}$ for each language pair.

### 4.1 Experiments on Transfer Grammar Induction

In transfer grammar induction, we train the first model on the first language independent of the second language; then, with the first model fixed, we optimize our knowledge sharing objective with respect to the second model; finally, we evaluate the second model on the test set of the second language. In this way, we want to test whether our methods can transfer useful linguistic knowledge from the first language to the second language. We report the results of transfer grammar induction from English to the other nine languages in Table 3. Our edge regularization and tree regularization methods outperform the Convex-MST baseline in almost all the cases. The weight regularization method achieves worse results than Convex-MST except for the Swedish language, which demonstrates that directly regularizing weight parameters may not work well in the transfer grammar induction task. For the Swedish language, although direct transfer already achieves better performance than Convex-MST, our regularization methods can further boost the performance by a large margin. The Indonesian language is the only language for which transfer grammar induction provides no benefit, possibly because of its significant syntactic difference from the English language. Our additional experimental results on UD treebank 1.4 show a similar trend.

| PAIR | CODE | BASE | COMB | W-REG | E-REG | T-REG |
|---|---|---|---|---|---|---|
| EN-DE | EN | 62.3 | +0 | **+0.5** | +0.1 | +0.2 |
| | DE | 60.2 | +0.1 | -0.8 | **+0.5** | +0.4 |
| EN-ES | EN | 62.3 | +0.2 | **+0.7** | +0.4 | **+0.7** |
| | ES | 68.8 | -1.8 | **+1.4** | +1.1 | +0.8 |
| EN-FR | EN | 62.3 | +0.4 | **+0.5** | +0.1 | +0.1 |
| | FR | 72.3 | -3.6 | -0.6 | **+1.4** | **+1.4** |
| EN-ID | EN | 62.3 | +0.7 | **+1.3** | +0.1 | +0.7 |
| | ID | 69.7 | -19.2 | -2.1 | +0.5 | **+0.9** |
| EN-IT | EN | 62.3 | +0.7 | +0.1 | +0.6 | **+1.0** |
| | IT | 64.3 | +0.7 | **+1.8** | +0.4 | +0.8 |
| EN-JA | EN | 62.3 | +1.0 | **+0.9** | -0.1 | -0.2 |
| | JA | 57.5 | -4.3 | -0.2 | **+2.7** | +2.4 |
| EN-KO | EN | 62.3 | +0.3 | **+1.4** | +0.5 | +1.1 |
| | KO | 59.0 | -3.8 | +0.1 | **+1.0** | +0.4 |
| EN-PTBR | EN | 62.3 | +1.0 | **+0.8** | +0.6 | +0.7 |
| | PT-BR | 68.3 | -0.9 | **+1.4** | **+1.4** | +1.2 |
| EN-SV | EN | 62.3 | -0.5 | -0.9 | **+0.1** | **+0.1** |
| | SV | 66.2 | +1.7 | **+1.9** | +1.0 | +1.1 |
| Avg | EN | 62.30 | +0.42 | **+0.58** | +0.23 | +0.49 |
| | Other | 65.14 | -3.46 | +0.32 | **+1.11** | +1.04 |
| Avg-All | EN | 52.10 | -0.48 | -0.40 | +0.11 | **+0.42** |
| | Other | 56.16 | -3.05 | +0.64 | **+1.55** | +1.21 |
| UD 1.4* | EN | 53.50 | -0.76 | -0.09 | + 0 | **+0.71** |
| | Other | 52.83 | -0.09 | +0.58 | **+1.92** | +1.52 |

Table 4: Results of bilingual grammar induction on test sentences no longer than 10 (except the Avg-All row which shows the average accuracies on all the sentences). BASE refers to the individually trained baseline. COMB refers to learning a single model from the combined training set of the two languages. *: for the UD 1.4 dataset we show the average results.

We perform transfer grammar induction from English to Swedish with different values of $\lambda$ and show the results in Figure 2. We can see that the impact of different hyper-parameter values on the accuracy generally follows the same tendency for our three methods.

### 4.2 Experiments on Bilingual Grammar Induction

In bilingual grammar induction, we jointly train two models on two languages. In our experiments, we pair English with each of the other nine languages. The results are reported in Table 4. It can be seen that in most cases joint training leads to better accuracies than the individually trained models as well as the single model learned from the combined training set. By comparing table 4 with table 3, we can also see that bilingual joint training leads to better accuracies than transfer grammar induction, which shows the benefit of training two models simultaneously rather than sequentially. Again, our additional experimental results on UD treebank 1.4 show a similar trend.

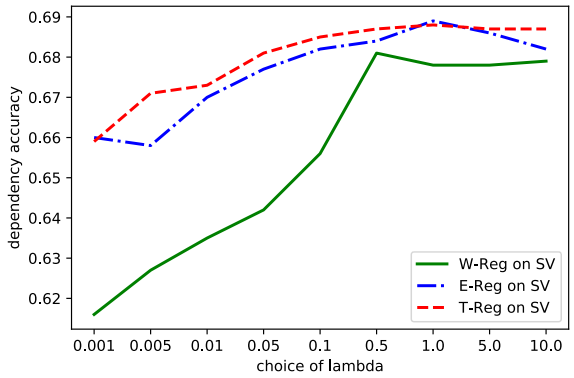


Figure 2: Transfer grammar induction on the SV language with different hyper-parameter values for the three regularization methods

## 5 Related Work

Our work is related to many previous work.

**Unsupervised Transfer Learning** There has been previous work aiming at solving an unsupervised learning task of a target domain with the help of knowledge learned from a source domain (Dai et al., 2008; Wang et al., 2008; Pan and Yang, 2010). There is no labeled data in both the source and the target domains during training. Our transfer grammar induction setting can be seen as an instance of unsupervised transfer learning.

**Cross-lingual Supervised Dependency Parsing** This task focuses on learning a parser with unlabeled training data and additional labeled training data of a second language (McDonald et al., 2011; Naseem et al., 2012; Guo et al., 2015). The main difference between our approach and theirs is that our approach is fully unsupervised. and do not utilize external information like word alignments or cross-lingual word embeddings.

**Other Approaches to Multilingual Grammar Induction** To the best of our knowledge, this task is first proposed by Kuhn (2004). They assume that the syntax trees induced from parallel sentences share structured regularities and utilize the word alignments to guide parsing. From then on, many approaches are proposed on both constituency grammar induction and dependency grammar induction (Snyder et al., 2009; Berg-Kirkpatrick and Klein, 2010). We differ from these approaches in that we do not make use of any external rules or knowledge.

## 6 Conclusion

In this paper, we propose three regularization-based knowledge sharing methods to bilingual grammar induction problems. We test our methods on transfer grammar induction and bilingual grammar induction and show that our methods achieve better performance than the baselines. In future work, we plan to investigate the effectiveness of our approach in other types of induction tasks.

## Acknowledgments

This work was supported by the Major Program of Science and Technology Commission Shanghai Municipal (17JC1404102).

## References

Taylor Berg-Kirkpatrick and Dan Klein. 2010. Phylogenetic grammar induction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1288–1297, Uppsala, Sweden. Association for Computational Linguistics.

Jiong Cai, Yong Jiang, and Kewei Tu. 2017. Crf autoencoder for unsupervised dependency parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1638–1643, Copenhagen, Denmark. Association for Computational Linguistics.

Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. 2008. Self-taught clustering. In *Proceedings of the 25th international conference on Machine learning*, pages 200–207. ACM.

Jason M Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th conference on Computational linguistics-Volume 1*, pages 340–345. Association for Computational Linguistics.

Edouard Grave and Noémie Elhadad. 2015. A convex and feature-rich discriminative approach to dependency grammar induction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1375–1384, Beijing, China. Association for Computational Linguistics.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1234–1244.

Wenjuan Han, Yong Jiang, and Kewei Tu. 2019. Enhancing unsupervised generative dependency parser with contextual information. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5315–5325, Florence, Italy. Association for Computational Linguistics.

Yong Jiang, Wenjuan Han, and Kewei Tu. 2016. Unsupervised neural dependency parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 763–771, Austin, Texas. Association for Computational Linguistics.

Yong Jiang, Wenjuan Han, and Kewei Tu. 2017. Combining generative and discriminative approaches to unsupervised dependency parsing via dual decomposition. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Copenhagen, Denmark. Association for Computational Linguistics.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, Austin, Texas. Association for Computational Linguistics.

Dan Klein and Christopher D Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 478. Association for Computational Linguistics.

Jonas Kuhn. 2004. Experiments in parallel-text based grammar induction. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 470. Association for Computational Linguistics.

Bowen Li, Jianpeng Cheng, Yang Liu, and Frank Keller. 2019. Dependency grammar induction with a neural variational transition-based parser. In *AAAI 2019*.

Kai Liu, Yajuan Lü, Wenbin Jiang, and Qun Liu. 2013. Bilingually-guided monolingual dependency grammar induction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1063–1072, Sofia, Bulgaria. Association for Computational Linguistics.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 91–98. Association for Computational Linguistics.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, et al. 2013. Universal dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 92–97.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the conference on empirical methods in natural language processing*, pages 62–72. Association for Computational Linguistics.

Tahira Naseem, Regina Barzilay, and Amir Globerson. 2012. Selective sharing for multilingual dependency parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 629–637. Association for Computational Linguistics.

Hiroshi Noji, Yusuke Miyao, and Mark Johnson. 2016. Using left-corner parsing to encode universal structural constraints in grammar induction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 33–43, Austin, Texas. Association for Computational Linguistics.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

Benjamin Snyder, Tahira Naseem, and Regina Barzilay. 2009. Unsupervised multilingual grammar induction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 73–81, Suntec, Singapore. Association for Computational Linguistics.

Zheng Wang, Yangqiu Song, and Changshui Zhang. 2008. Transferred dimensionality reduction. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 550–565. Springer.

# Encoders Help You Disambiguate Word Senses in Neural Machine Translation

**Gongbo Tang**[1] **Rico Sennrich**[2,3] **Joakim Nivre**[1]

[1]Department of Linguistics and Philology, Uppsala University
[2]Institute of Computational Linguistics, University of Zurich
[3]School of Informatics, University of Edinburgh
firstname.lastname@{lingfil.uu.se, ed.ac.uk}

## Abstract

Neural machine translation (NMT) has achieved new state-of-the-art performance in translating ambiguous words. However, it is still unclear which component dominates the process of disambiguation. In this paper, we explore the ability of NMT encoders and decoders to disambiguate word senses by evaluating hidden states and investigating the distributions of self-attention. We train a classifier to predict whether a translation is correct given the representation of an ambiguous noun. We find that encoder hidden states outperform word embeddings significantly which indicates that encoders adequately encode relevant information for disambiguation into hidden states. In contrast to encoders, the effect of decoder is different in models with different architectures. Moreover, the attention weights and attention entropy show that self-attention can detect ambiguous nouns and distribute more attention to the context.

## 1 Introduction

Neural machine translation (NMT) models (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2015; Luong et al., 2015) have access to the whole source sentence for the prediction of each word, which intuitively allows them to perform word sense disambiguation (WSD) better than previous phrase-based methods, and Rios et al. (2018) have confirmed this empirically. However, it is still unclear which component dominates the ability to disambiguate word senses. We explore the ability of NMT encoders and decoders to disambiguate word senses by evaluating hidden states and investigating the self-attention distributions.

Marvin and Koehn (2018) find that the hidden states in higher encoder layers do not perform disambiguation better than those in lower layers and conclude that encoders do not encode enough relevant context for disambiguation. However, their results are based on small data sets, and we wish to revisit this question with larger-scale data sets. Tang et al. (2018b) speculate that encoders have encoded the relevant information for WSD into hidden states before decoding but without any experimental tests.

In this paper, we first train a classifier for WSD, on a much larger data set than Marvin and Koehn (2018), extracted from ContraWSD (Rios et al., 2017), for both German→English (DE→EN) and German→French (DE→FR). The classifier is fed a representation of ambiguous nouns and a word sense (represented as the embedding of a translation candidate), and has to predict whether the two match. We can learn the role that encoders play in encoding information relevant for WSD by comparing different representations: word embeddings and encoder hidden states at different layers. We extract encoder hidden states from both RNN-based (*RNNS2S*) (Luong et al., 2015) and *Transformer* (Vaswani et al., 2017) models. Belinkov et al. (2017a,b) have shown that the higher layers are better at learning semantics. We hypothesize that the hidden states in higher layers incorporate more relevant information for WSD than those in lower layers. In addition to encoders, we also probe how much do decoder hidden states contribute to the WSD classification task.

Recently, the distributions of attention mechanisms have been used for interpreting NMT models (Ghader and Monz, 2017; Voita et al., 2018; Tang et al., 2018b; Voita et al., 2019; Tang et al., 2019). We further investigate the attention weights and attention entropy of self-attention in encoders to explore how self-attention incorporates relevant information for WSD into hidden states. As sentential information is helpful in disambiguating ambiguous words, we hypothesize that self-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1429–1435, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

attention pays more attention to the context when modeling ambiguous words, compared to modeling words in general.

Here are our findings:

- Encoders encode lots of relevant information for word sense disambiguation into hidden states, even in the first layer. The higher the encoder layer, the more relevant information is encoded into hidden states.
- Forward RNNs are better than backward RNNs in modeling ambiguous nouns.
- Decoders hidden states have different effects on WSD in *Transformer* and *RNNS2S*.
- Self-attention focuses on the ambiguous nouns themselves in the first layer and keeps extracting relevant information from the context in higher layers.
- Self-attention can recognize the ambiguous nouns and distribute more attention to the context words compared to dealing with nouns in general.

## 2 Methodology

### 2.1 WSD Classifier

*ContraWSD* (Rios et al., 2017) is a WSD test set for NMT. Each ambiguous noun in a specific sentence has a small number of translation candidates. We generate instances that are labelled with one candidate and a binary value indicating whether it corresponds to the correct sense.

**Encoders** Given an input sentence, NMT encoders generate hidden states of all input tokens. Our analysis focuses on the hidden states of ambiguous nouns ($R_{ambi}$). We use word embeddings from NMT models to represent the translation candidates ($R_{sense}$). If the ambiguous nouns or translation candidates are split into subwords, we just sum the representations.

Figure 1 illustrates the WSD classification task. We first generate hidden states for each sentence. The classifier is a feed-forward neural network with only one hidden layer. The input of the classifier is the concatenation of $R_{ambi}$ and $R_{sense}$. The classifier predicts whether the translation is the correct sense of the ambiguous noun.

As the baseline, we use word embeddings from NMT models as representations of ambiguous nouns. Each ambiguous noun has only one corresponding word embedding, so such a classifier can at best learn a most-frequent-sense solution, while

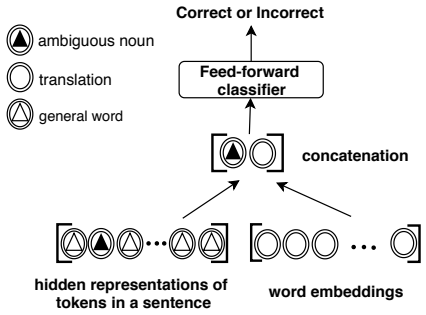


Figure 1: Illustration of the WSD classification task, using encoder hidden states to represent ambiguous nouns. The input of the classifier is the concatenation of the ambiguous word and the translation. The output of the classifier is "correct" or "incorrect".

hidden states are based on sentential information, so that ambiguous nouns have different representations in different source sentences. We can learn to what extent relevant information for WSD is encoded by encoders by comparing to the baseline.

**Decoders** To explore the role of decoders, we feed the decoder hidden state at the time step predicting the translation of the ambiguous noun, and the word embedding of the current translate candidate into the classifier. The decoder hidden state is extracted from the last decoder layer. To get these hidden states. we force NMT models to generate the reference translations using constrained decoding (Post and Vilar, 2018). Since decoders are crucial in NMT, we assume that the decoder hidden states incorporate more relevant information for WSD from the decoder side. Thus, we hypothesize that using decoder hidden states can achieve better WSD performance.

### 2.2 Attention Distribution

The attention weights can be viewed as the degree of contribution to the current word representation, which provides a way to interpret NMT models. Tang et al. (2018a) have shown that Transformers with self-attention are better at WSD than RNNs. However, the working mechanism of self-attention has not been explored. We try to use the attention distributions in different encoder layers to interpret how self-attention incorporates relevant information to disambiguate word senses.

All the ambiguous words in the test set are nouns. Ghader and Monz (2017) have shown that nouns have different attention distributions from other word types. Thus, we compare the attention distributions of ambiguous nouns to nouns in gen-

eral[1] in two respects. One is the attention weight over the word itself. The other one is the concentration of attention distributions. We use attention entropy (Ghader and Monz, 2017) to measure the concentration.

$$E_{At}(x_t) = -\sum_{i=1}^{|x|} At(x_i, x_t) \log At(x_i, x_t) \quad (1)$$

Here $x_i$ denotes the $i$th source token, $x_t$ is the current source token, and $At(x_i, x_t)$ represents the attention weight from $x_t$ to $x_i$. We merge subwords after encoding, following the method in Koehn and Knowles (2017).[2] Each self-attention layer has multiple heads and we average the attention weights from all the heads.

In theory, sentential information is more important for ambiguous words that need to be disambiguated than non-ambiguous words. From the perspective of attention weights, for ambiguous words, we hypothesize that self-attention distributes more attention to the context words to capture the relevant sentential information, compared to words in general. From the perspective of attention entropy, we hypothesize that self-attention focuses on the related context words rather than the entire sentence which produces a smaller entropy. If ambiguous words have a lower weight and a smaller entropy than words in general, the results can confirm our hypotheses.

## 3 Experiments

For NMT models, we use the *Sockeye* (Hieber et al., 2017) toolkit to train *RNNS2S*s and *Transformer*s. DE→EN training data is from the WMT17 shared task (Bojar et al., 2017). DE→FR training data is from Europarl (v7) (Koehn, 2005) and News Commentary (v11) cleaned by Rios et al. (2017).[3] In *ContraWSD*, each ambiguous noun has a small number of translation candidates. The average number of word senses per noun is 2.4 and 2.3 in DE→EN and DE→FR, respectively. We generate instances that are labelled with one candidate and a binary value indicating whether it corresponds to the correct sense. we get 50,792 and 43,268 instances in DE→EN and DE→FR, respectively. 5K/5K examples are randomly selected as the test/development set. The remaining examples are used for training. We train 10 times with different seeds for each classifier and apply average accuracy. Table 1 lists the detailed statistics of the data. More experimental details are provided in the Appendix.

| | DE→EN | DE→FR |
|---|---|---|
| NMT training data | 5.9M | 2.1M |
| Word senses | 84 | 71 |
| Lexical ambiguities | 7,359 | 6,746 |
| Instances | 50,792 | 43,268 |

Table 1: Training data for NMT, and data extracted from ContraWSD: Word senses: total number of senses. Lexical ambiguities: number of sentences containing an ambiguous word. Instances: number of instances generated for WSD classification.

### 3.1 Results

Table 2 provides the BLEU scores and the WSD accuracy on test sets, using different representations to represent ambiguous nouns. *ENC* denotes encoder hidden states; *DEC* means decoder hidden states.

| | DE→EN | | DE→FR | |
|---|---|---|---|---|
| | *RNN.* | *Trans.* | *RNN.* | *Trans.* |
| BLEU | 29.1 | 32.6 | 17.0 | 19.3 |
| *Embedding* | 63.1 | 63.2 | 68.7 | 68.9 |
| *ENC* | 94.2 | 97.2 | 91.7 | 95.6 |
| *DEC* | 97.9 | 91.2 | 95.1 | 91.6 |

Table 2: BLEU scores of NMT models, and WSD accuracy on the test set using word embeddings or hidden states to represent ambiguous nouns. The hidden states are from the highest layer.[4] *RNN.* and *Trans.* denote *RNNS2S* and *Transformer* models, respectively.

*ENC* achieves much higher accuracy than *Embedding*. The WSD accuracy of *Embedding* are around 63% and 69% in the two languages. While the accuracy of *ENC* increases to over 91%. The absolute accuracy gap varies from 23% to 34%, which is substantial. This result indicates that encoders have encoded a lot of relevant information for WSD into hidden states. In addition, *DEC* achieves even higher accuracy than *ENC* in *RNNS2S* models but not in *Transformer* models.

[1] We use *TreeTagger* (Schmid, 1995) to identify nouns.

[2] (1) If a query word is split into subwords, we add their attention weights. (2) If a key word is split into subwords, we average their attention weights.

[3] http://data.statmt.org/ContraWSD/

[4] For encoders in *RNNS2S*s, this is the last backward RNN.

## 4 Analysis

### 4.1 WSD Classification

#### 4.1.1 RNNS2S vs. Transformer

*RNNS2S*s are inferior to *Transformer*s distinctly in BLEU score. However, the hidden states from *RNNS2S* also improve accuracy significantly, just not as much as those from *Transformer* models. This result indicates that *Transformer*s encode more relevant context for WSD than *RNNS2S*s and accords with the finding in Tang et al. (2018a) that *Transformer*s perform WSD better than *RNNS2S*s.

The results of *ENC* using *RNNS2S* in Table 2 are only based on hidden states from the last backward RNN. We also concatenate the hidden states from both forward and backward RNNs and get higher accuracy, 96.8% in DE→EN and 95.7% in DE→FR. The WSD accuracy of using bidirectional hidden states are competitive to using hidden states from Transformer models. However, concatenating forward and backward hidden states doubles the dimension. Thus, the comparison is not completely fair.

#### 4.1.2 Encoder Depth

Figure 2 illustrates WSD accuracy in different encoder layers, with standard deviation as error bars. Even the hidden states from the first layer boost the WSD performance substantially compared to using word embeddings. This means that most of the relevant information for WSD has been encoded into hidden states in the first encoder layer. For *Transformer*s, the WSD accuracy goes up consistently as the encoder layer gets higher. *RNNS2S* has 3 stacked bi-directional RNNs. Both forward and backward layers get higher accuracy when the depth increases. All the models show that hidden states in higher layers incorporate more relevant information for WSD.

Our results conflict with the findings in Marvin and Koehn (2018) where they find that hidden states in higher encoder layers do not perform disambiguation better than those in lower layers. One of the distinct differences from Marvin and Koehn (2018) is that we train the classifier with ∼40K instances. While they employ 426 examples. Moreover, they extract encoder hidden states from NMT models with different layers rather than different layers of the same model.

Moreover, it is interesting that the forward layers surpass the backward layers in the same bi-directional RNN. One possible explanation is that there is more relevant information for WSD before ambiguous nouns rather than after ambiguous nouns, which makes forward RNNs inject more relevant information into the hidden states of ambiguous nouns than backward RNNs.

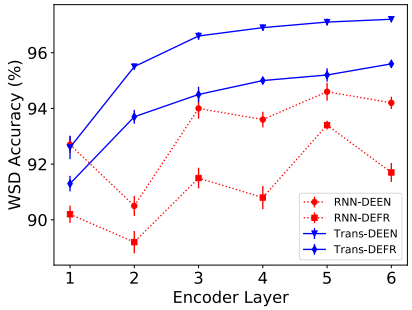


Figure 2: The WSD accuracy of using hidden states in different encoder layers, with standard deviation as error bars. For *RNNS2S*s, the odd layers (1, 3, 5) are forward RNNs and the even layers (2, 4, 6) are backward RNNs.

#### 4.1.3 Decoders

As Table 2 shows, RNN decoder hidden states could further improve the classification accuracy which accords with our hypothesis. It implies that the relevant information for WSD in the target-side has been well incorporated into the decoder hidden states to predict the translations of ambiguous nouns. It is curious that Transformer decoder hidden states are inferior to Transformer encoder hidden states in our WSD classification task, given that Tang et al. (2018a) and Rios et al. (2018) report better results with contrastive evaluation and semi-automatic evaluation of 1-best translations for Transformer models than for *RNNS2S*. However, note that our evaluation merely tests whether the information necessary for word sense disambiguation is encoded in hidden states and can be extracted by our binary classifier. In practice, decoder hidden states are used for predicting a target word from the entire vocabulary, and thus need to encode additional information which may confound our classifier.

Despite these differences between *RNNS2S* and the Transformer, our results show that WSD is already possible on the basis of the encoder representation of the ambiguous noun, and that extracting contextual information via encoder-decoder attention or from the target history is not essential for WSD.

### 4.2 Self-attention

#### 4.2.1 Attention Weights

Figure 3 exhibits the average attention weights of ambiguous nouns and all nouns over themselves in different layers. In the first layer, the attention weights are distinctly higher than those in higher layers. 87% and 90% of ambiguous nouns assign the highest attention to themselves in DE→EN and DE→FR, respectively. The attention weights drop dramatically from the second layer. It thus seems that self-attention pays more attention to the ambiguous nouns themselves in the first layer and to context words in the following layers.

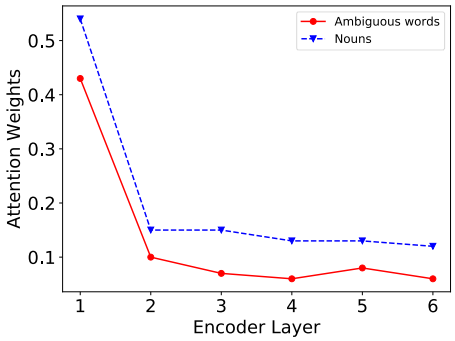


Figure 3: The average attention weights of ambiguous nouns and general nouns over themselves in different layers, in DE→EN (same pattern in DE→FR).

The attention weights of ambiguous nouns are lower than those of nouns in general. That is, more attention is distributed to context words, which implies that self-attention recognizes ambiguous nouns and distributes more attention to the context. We can conclude that self-attention pays more attention to context words to extract relevant information for disambiguation in all the layers, compared to nouns in general.

#### 4.2.2 Attention Entropy

Section 4.2.1 has shown that self-attention of ambiguous nouns distributes more attention to the context than self-attention of nouns in general but what does the attention distribution look like? Figure 4 displays the average attention entropy of ambiguous nouns and all nouns in different layers. From the second layer, ambiguous nouns have smaller attention entropy than nouns in general, which means that self-attention mainly distributes attention to some specific words rather than all the words. As self-attention focuses on the ambiguous nouns themselves in the first layer, this result accords with our hypothesis as well.

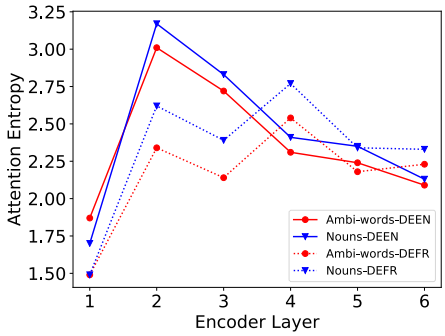


Figure 4: The average attention entropy of ambiguous nouns and nouns in different encoder layers.

In addition, there is a roughly general pattern that the attention entropy first rises and then drops. A plausible explanation is that the attention entropy first rises because context information is extracted from the entire sentence and later drops due to focusing on the most relevant context tokens.

## 5 Conclusion

In this paper, we investigate the ability of NMT encoders and decoders to disambiguate word senses. We first train a neural classifier to predict whether the translation is correct given the representations of ambiguous nouns. We find that encoder hidden states outperform word embeddings significantly in the classification task which indicates that relevant information for WSD has been well integrated by encoders. In addition, the higher the encoder layer, the more relevant information is encoded into hidden states. Moreover, the effect of decoder hidden states on WSD is different in Transformer and *RNNS2S* models.

We further explore the attention distributions of self-attention in encoders. The results show that self-attention can detect ambiguous nouns and distribute more attention to context words. Besides, self-attention focuses on the ambiguous nouns themselves in the first layer, then keeps extracting features from context words in higher layers.

## Acknowledgments

We thank all reviewers for their valuable and insightful comments. We acknowledge the computational resources provided by CSC in Helsinki and Sigma2 in Oslo through NeIC-NLPL (www.nlpl.eu). GT is mainly funded by the Chinese Scholarship Council (NO. 201607110016).

## References


Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations*, San Diego, California, USA.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017a. What do neural machine translation models learn about morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 861–872, Vancouver, Canada. Association for Computational Linguistics.

Yonatan Belinkov, Lluís Màrquez, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2017b. Evaluating layers of representation in neural machine translation on part-of-speech and semantic tagging tasks. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. 2017. Findings of the 2017 conference on machine translation (WMT17). In *Proceedings of the Second Conference on Machine Translation*, pages 169–214, Copenhagen, Denmark. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Hamidreza Ghader and Christof Monz. 2017. What does attention in neural machine translation pay attention to? In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 30–39, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, and Matt Post. 2017. Sockeye: A toolkit for neural machine translation. *arXiv preprint arXiv:1712.05690*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington, USA. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, San Diego, California, USA.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Proceedings of the 10th Machine Translation Summit*, pages 79–86, Phuket, Thailand.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39, Vancouver, Canada. Association for Computational Linguistics.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Rebecca Marvin and Phillip Koehn. 2018. Exploring word sense disambiguation abilities of neural machine translation systems. In *Proceedings of AMTA 2018 (Volume 1: MT Research Track)*, pages 125–131, Boston, USA. Association for Machine Translation in the Americas.

Matt Post. 2018. A call for clarity in reporting bleu scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191. Association for Computational Linguistics.

Matt Post and David Vilar. 2018. Fast lexically constrained decoding with dynamic beam allocation for neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1314–1324, New Orleans, Louisiana, USA. Association for Computational Linguistics.

Annette Rios, Laura Mascarell, and Rico Sennrich. 2017. Improving word sense disambiguation in neural machine translation with sense embeddings. In *Proceedings of the Second Conference on Machine Translation*, pages 11–19, Copenhagen, Denmark. Association for Computational Linguistics.

Annette Rios, Mathias Müller, and Rico Sennrich. 2018. The word sense disambiguation test suite at WMT18. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 588–596, Belgium, Brussels. Association for Computational Linguistics.

Helmut Schmid. 1995. Improvements in part-of-speech tagging with an application to german. In *Proceedings of the ACL SIGDAT-Workshop*, Dublin, Ireland. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the Neural Information Processing Systems 2014*, pages 3104–3112, Montréal, Canada.

Gongbo Tang, Mathias Müller, Annette Rios, and Rico Sennrich. 2018a. Why self-attention? a targeted evaluation of neural machine translation architectures. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4263–4272, Brussels, Belgium. Association for Computational Linguistics.

Gongbo Tang, Rico Sennrich, and Joakim Nivre. 2018b. An analysis of attention mechanisms: The case of word sense disambiguation in neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 26–35, Belgium, Brussels. Association for Computational Linguistics.

Gongbo Tang, Rico Sennrich, and Joakim Nivre. 2019. Understanding neural machine translation by simplification: The case of encoder-free models. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2019*, Varna, Bulgaria.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 6000–6010. Curran Associates, Inc.

Elena Voita, Pavel Serdyukov, Rico Sennrich, and Ivan Titov. 2018. Context-aware neural machine translation learns anaphora resolution. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1264–1274, Melbourne, Australia. Association for Computational Linguistics.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

# Korean Morphological Analysis with Tied Sequence-to-Sequence Multi-Task Model

**Hyun-Je Song**
Dept. of Information Technology
Chonbuk National University
Jeonju, 54896, Korea
hyunje.song@jbnu.ac.kr

**Seong-Bae Park**
Dept. of Computer Science and Engineering
Kyung Hee University
Yongin, 17104, Korea
sbpark71@khu.ac.kr

## Abstract

Korean morphological analysis has been considered as a sequence of morpheme processing and POS tagging. Thus, a pipeline model of the tasks has been adopted widely by previous studies. However, the model has a problem that it cannot utilize interactions among the tasks. This paper formulates Korean morphological analysis as a combination of the tasks and presents a tied sequence-to-sequence multi-task model for training the two tasks simultaneously without any explicit regularization. The experiments prove the proposed model achieves the state-of-the-art performance.

## 1 Introduction

Korean is an agglutinative language (Song, 2006). Thus, it is a fundamental step for understanding a sentence to analyze the grammatical structure of *eojeol*s, where an eojeol is a linguistic unit segmented by a white space. An eojeol is composed of one or more morphemes. As a result, one eojeol can be analyzed into several morpheme combinations depending on a context, which yields different part-of-speech (POS) tags of the morpheme combinations. In addition, some morphemes have a different surface form from their base form when they are derived from an eojeol. Therefore, the goal of Korean morphological analyzer is not only to decompose and recover morphemes from eojeols precisely (morpheme processing), but also to assign POS tags to the decomposed and/or recovered morphemes accurately according to a context (POS tagging).

Traditional approaches to Korean morphological analysis have adopted a pipeline model of morpheme processing and POS tagging (Lee and Rim, 2009; Na, 2015; Choi et al., 2016; Matteson et al., 2018; Song and Park, 2018). That is, they first decompose and recover morphemes from eojeols or assign so-called POSMORPH tags (Heigold et al., 2016), and then an actual POS tag sequence is determined or resolved from the POSMORPH tags using a sequential labeling algorithm. However, this pipeline model suffers from two kinds of weaknesses. One weakness is that the errors from morpheme processing are apt to be propagated to POS tagging, and the other is that it is difficult to model mutual interactions between morpheme processing and POS tagging under the pipeline model.

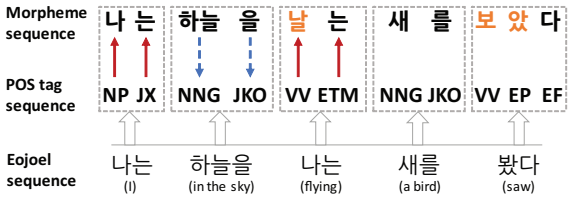


Figure 1: Korean morphological analysis of a sentence "나는 하늘을 나는 새를 봤다" of which meaning is "*I saw a bird flying in the sky*". Correct morpheme analysis helps predicting POS tags (blue dotted arrows) while POS tagging affects morpheme analysis (red arrows). Orange morphemes in the morpheme sequence are recovered morphemes. Best viewed in color.

In Korean, morpheme processing and POS tagging affect each other. That is, correct morpheme analysis helps POS tagging, and POS tags are helpful in analyzing morphemes. Figure 1 shows such an example in Korean morphological analysis. The second eojeol '하늘을 (in the sky)' is composed of two morphemes '하늘 (sky)' and '을 (*objective postposition*)'. If '하늘을' is precisely decomposed into '하늘' and '을', their POS tags can be predicted easily. This is the information flow on which the previous studies focus. On the other hand, the eojeol '나는' which appears twice in this figure is morphologically ambiguous and thus can



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1436–1441,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

be analyzed in two ways. However, if morpheme processing obtains some information from POS tagging, it can decompose ambiguous eojeols correctly. That is, the first '나는' is decomposed into '나 (I)' and '는 (*topical postposition*)', while the second '나는' is into '날 (fly)' and '는 (*verbal ending*)'. Therefore, morpheme processing and POS tagging should be trained simultaneously.

This paper proposes a model to train morpheme processing and POS tagging simultaneously in Korean morphological analysis. The proposed model regards morpheme processing and POS tagging as individual tasks. Then, the two tasks are jointly trained under a sequence-to-sequence multi-task framework (Luong et al., 2016; Anastasopoulos and Chiang, 2018). The main characteristic of the proposed model is that there is a single decoder for both generating a morpheme sequence and assigning POS tags to the generated morphemes. As a result, the decoder shares general representations of both tasks and forces a one-to-one mapping between morphemes and POS tags without additional regularization. Morpheme processing is accomplished by a pointer-generator network (See et al., 2017), while POS tagging is done by a CRF network (Huang et al., 2015; Lample et al., 2016) utilizing the information from morpheme processing. Our experimental results show that the proposed model outperforms existing Korean morphological analyzers with its state-of-the-art performance.

## 2 Korean Morphological Analysis

### 2.1 Morpheme Processing and POS Tagging

Given an input sequence $\mathbf{x} = (w_1, ..., w_n)$ where $w_i$ is the $i$-th eojeol, Korean morphological analysis aims to produce an output sequence $\mathbf{y} = (\langle m_1, t_1 \rangle, ..., \langle m_k, t_k \rangle)$ where $m_j$ is the $j$-th morpheme and $t_j$ is its POS tag. Korean morphological analysis is different from existing NLP tasks such as English POS tagging (Toutanova et al., 2003; Manning, 2011), and joint word segmentation and POS tagging for Chinese (Zhang and Clark, 2008; Shao et al., 2017; Chen et al., 2017). In English POS tagging, a word and its tag have a one-to-one mapping so that the length of an input sequence is equal to that of an output sequence. On the other hand, they are usually different in Korean, because an eojeol consists of several morphemes. That is, $n < k$ in general. The key distinguishing factor between Chinese and Korean morphological analysis is that lemmatization that recovers a base form for every morpheme as well as morpheme segmentation is a must in Korean morphological analysis, while Chinese morphological analysis requires only word segmentation. That is, in Korean, $\bigoplus_{i=1}^{n} w_i \neq \bigoplus_{i=1}^{k} m_i$ in general, where $\bigoplus$ is a concatenate operator. According to Figure 1, the eojeol sequence '나는 하늘을 나는 새를 봤다' is different from its morpheme sequence '나는 하늘을 날는 새를 보았다' where underline marks indicate the difference.[1]

Given an annotated corpus $\mathcal{D} = (\langle \mathbf{x}_i, \mathbf{y}_i \rangle)_{i=1}^{N}$ where $N$ is the number of sentences, Korean morphological analyzer is obtained by maximizing the conditional probability $p(\mathbf{y}|\mathbf{x})$. The conditional probability can be calculated as

$$p(\mathbf{y}|\mathbf{x}) \approx \underbrace{p(\mathbf{t}|\mathbf{m}, \mathbf{x})}_{\text{POS tagging}} \quad \underbrace{p(\mathbf{m}|\mathbf{x})}_{\text{Morpheme processing}}, \tag{1}$$

where $\mathbf{m} = (m_1, \ldots, m_k)$ is a morpheme sequence, and $\mathbf{t} = (t_1, \ldots, t_k)$ is a tag sequence. Equation (1) implies that $p(\mathbf{y}|\mathbf{x})$ can be further decomposed into two another conditional probabilities. One conditional probability, $p(\mathbf{m}|\mathbf{x})$, generates a morpheme sequence $\mathbf{m}$ from the input sequence $\mathbf{x}$, and the other $p(\mathbf{t}|\mathbf{m}, \mathbf{x})$ assigns POS tags considering the morpheme sequence $\mathbf{m}$ and the input sequence $\mathbf{x}$. Therefore, $p(\mathbf{m}|\mathbf{x})$ corresponds to morpheme processing, while $p(\mathbf{t}|\mathbf{m}, \mathbf{x})$ is POS tagging.

### 2.2 Linguistic Unit in Morphological Analysis

Eojeol is the unit of spacing in Korean sentences. However, it is inappropriate to use eojeols directly in sequence-to-sequence models, because the number of eojeols is extremely huge due to the agglutinative characteristics of Korean. For instance, there exist 624,655 kinds of eojeols in the corpus used in the experiments. This large eojeol size causes a complexity problem in morphological analysis.

According to Korean orthography, an eojeol is a sequence of *syllables*. For instance, an eojeol '하늘을' is a sequence of three syllables '하', '늘', and '을'. A morpheme in Korean is also a sequence of syllables like an eojeol. The number of distinguished syllables in the corpus above is just

[1] We skip morpheme segmentation symbols in the morpheme sequence in order to emphasis the difference between the eojeol sequence and the morpheme one.

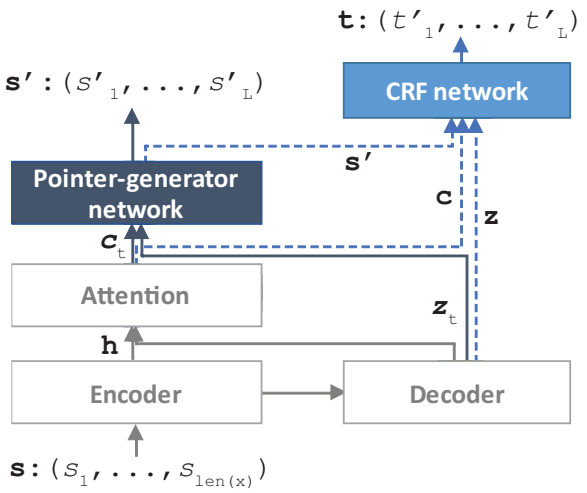


Figure 2: The proposed model for Korean morphological analysis based on a Tied sequence-to-sequence multi-task model. Since syllable is adopted as a unit for the proposed model, the input for the encoder is a syllable sequence $\mathbf{s}$. For clarity's sake, there are some dependencies not shown.

5,245 which is much smaller than that of eojeols. Therefore, this paper adopts syllable as a unit for sequence-to-sequence models for Korean morphological analysis.

## 3 Tied Sequence-to-Sequence Multi-Task Model

The proposed model is based on the sequence-to-sequence model with attention (Bahdanau et al., 2015) and extends the model for multi-task learning similarly to the work of Anastasopoulos and Chiang (2018). The proposed model consists of four parts: a recurrent encoder, a recurrent decoder, attention, and task-dependent networks. Figure 2 shows an overall structure of the proposed model. The encoder encodes an input sequence $\mathbf{x}$ into a sequence of hidden states. Since syllable is adopted as the unit for the model, the encoder (bidirectional LSTM) actually transforms an input syllable sequence $\mathbf{s} = (s_1, \ldots, s_{len(\mathbf{x})})$ into a sequence of encoder hidden states $\mathbf{h}_1, \ldots, \mathbf{h}_{len(\mathbf{x})}$, where $len(\cdot)$ is the number of syllables including white spaces. The attention transforms the encoder hidden states $\mathbf{h}$ into a sequence of context vectors $\mathbf{c}$ through attention weights. The context vectors capture relevant source-side information for both morpheme processing and POS tagging.

Unlike other multi-task models, there is only one decoder for the two tasks of morpheme processing and POS tagging in the proposed model. Note that a morpheme and its POS tag should have a one-to-one mapping relation. Since the two tasks are forced to share the same decoder states, a one-to-one mapping between a morpheme and its POS tag is guaranteed without any explicit regularization or task-specific token. In detail, the decoder (unidirectional LSTM) first computes a sequence of decoder hidden states $\mathbf{z}$. At every time $t$, the decoder hidden state $\mathbf{z}_t$ is calculated from the embedding of the previous syllable $\mathbf{s}'_{t-1}$, the decoder state $\mathbf{z}_{t-1}$, and the context vector $\mathbf{c}_t$. Then, two task-dependent networks solve their own task using the same $\mathbf{z}$.

In morpheme processing represented by solid arrows in Figure 2, a syllable $s'_t$ is produced at every time $t$. Most syllables are directly copied from the input syllables while some syllables are newly generated from a vocabulary to recover the base form of a morpheme. To do this, we adopt the pointer-generator network (See et al., 2017) which allows both copying syllables via pointing and generating syllables from a fixed vocabulary. The advantage of this approach is that we can handle non-Korean characters such as *hanja* (Chinese character) and special symbols with ease.

In POS tagging represented by dotted arrows in Figure 2, it is possible to generate a POS tag at every time $t$ by paying attention to the encoder syllable states as done in morpheme processing. However, this approach does not reflect the global dependency among adjacent tags. To solve this problem, this paper adopts a CRF network (Huang et al., 2015; Lample et al., 2016) over the pointer-generator network. When the morpheme sequence generated at the morpheme processing is given, the CRF network predicts POS tags for the morpheme sequence by considering the global dependency among tags (Lafferty et al., 2001). Furthermore, this paper uses skip-connections (He et al., 2016) in order for the CRF network to pay attention to the hidden states of the decoder. Since some POS tags can be predicted directly from input syllables, the skip-connections help the CRF network consider the information of input syllables. In detail, a sequence of syllables generated by the pointer-generator network is transformed to vectors, and the vectors are concatenated with the decoder state $\mathbf{z}$ and the context vector $\mathbf{c}$. Then, the concatenated vectors are fed to the CRF layer to predict POS tags.

| Methods | | | | F1-score | Acc (%) |
|---|---|---|---|---|---|
| Model | Morpheme processing | POS tagging | Type | | |
| Single-Task | Pointer-Generator | | | 97.36 | 95.57 |
| Multi-Task | Generator | Generator | Non-cascade | 97.10 | 95.09 |
| | Generator | CRF | Cascade | 97.25 | 95.30 |
| | Generator | CRF | Tied | 97.31 | 95.49 |
| | Pointer-Generator | Generator | Non-cascade | 97.30 | 95.45 |
| | Pointer-Generator | CRF | Cascade | 97.40 | 95.61 |
| | Pointer-Generator | CRF | Tied | **97.43** | **95.68** |
| Pipeline | Na (2015) (CRF → CRF → Post-processing) | | | 97.21 | 95.22 |
| | Song and Park (2018) (Generator → CRF) | | | 97.27 | 95.29 |
| | khaiii (CNN → Post-processing) | | | 94.88 | 91.86 |

Table 1: Performances of Korean morphological analyzers

| Information | Training | Development | Test |
|---|---|---|---|
| Sentences | 197,508 | 5,000 | 50,631 |
| Eojeols | 2,674,571 | 97,292 | 694,524 |
| Morphemes | 5,952,985 | 203,244 | 1,542,483 |
| Avg. eojeols | 13.54 | 19.46 | 13.72 |
| No. of POS tags | 42 | | |

Table 2: Simple statistics on the data set used.

Note that syllable is the unit for the model and one POS tag spans several syllables of a morpheme. Thus, to identify morpheme boundaries, morpheme processing of the proposed model generates a special symbol of morpheme ending for every explicit morpheme boundary instead of adopting the IOB tagging scheme. Such an approach reduces the number of POS tags and is helpful in segmenting eojeols into morphemes precisely.

The proposed model is trained to maximize the weighted sum of conditional log-likelihood of each task where the weights are set to be equal.

## 4 Experiments

### 4.1 Experimental Settings

The same data set with the work of Na (2015) is used for the experiments. This data set is derived from Sejong corpus.[2] Each eojeol in the Sejong corpus is annotated with pairs of a morpheme and its POS tag. The simple statistics on the data set used in the experiments is given in Table 2.

We set the syllable embedding size and the hidden size to 100. The number of LSTM layers is set to three and the batch size is 128. We use the gradient normalization with a threshold of five.

Three baselines are adopted to show the superiority of the proposed model. The first baseline is the model of Na (2015). This model consists of three sub-models (CRF segmentation, CRF tagging, and post-processing) and executes these three models in consecutive order. The second is the model of Song and Park (2018) which consists of two sub-models (generator and CRF tagger). The last is khaiii, a publicly-available CNN based morphological analyzer.[3] This analyzer assigns a POSMORPH tag for every syllable using a CNN classifier and then resolves the tag through post-processing. These baselines are all pipeline models. We also compare the proposed model with a single task non-pipeline model which generates a single sequence of morphemes and POS tags using the pointer-generator network.

The F1-measure at the morpheme level and the accuracy at the eojeol level are used as evaluation metrics, and all performances are micro-averaged.

### 4.2 Experimental Results

Table 1 shows the performances of the proposed model and the baselines. It also contains the results of the ablation study on the proposed model by changing the network of sub-tasks or the structure of multi-task models. 'Non-cascade' is similar to the standard multi-task model (Dong et al., 2015) in which each task just shares a decoder and there is no direct connection between tasks. According to Figure 2, there are no dotted arrows (marked $\mathbf{s}'$ and $\mathbf{c}$) from the Pointer-generator network and the CRF network in 'Non-cascade' model. 'Cascade'

[2] Available at https://ithub.korean.go.kr

[3] https://github.com/kakao/khaiii

| Type | Percentage |
|---|---|
| Morpheme segmentation | 5.8% |
| POS tagging | 39.3% |
| Morpheme recovery | 54.9% |

Table 3: The ratio of different error types.

is a multi-task model where the POS tagging network has a connection from the morpheme processing network but no skip connection from the decoder. In Figure 2, 'Cascade' model has no dotted arrow (marked $\mathbf{z}$) from the decoder to the CRF network. 'Tied' is the proposed model that considers both connections.

According to the table, all multi-task models that adopt Pointer-Generator and/or CRF outperform the baseline Generator-Generator, which proves their adoption is effective in improving the performance of morphological analysis. It is also noticed from the table that 'Cascade' multi-task model achieves better performance than 'Non-cascade' model, which implies that the unimpeded information from morpheme processing to POS tagging helps predict accurate POS tags. The proposed 'Tied' multi-task model yields higher performances than 'Cascade' model. When training the 'Tied' multi-task model, the feedbacks from POS tagging network influence decoder states directly. As a result, the 'Tied' model is able to learn better representation for both morpheme processing and POS tagging. All these results indicate that 'Tied' multi-task model with Pointer-Generator network and CRF is the best choice for Korean morphological analysis.

Compared to the pipeline models which showed the state-of-the-art performance previously, even simple multi-task models report a similar performance, which means that morphological analysis can be improved by training morpheme processing and POS tagging jointly. Even if the single-task non-pipeline model achieves a reasonable performance, its performance is not as high as that of the proposed model. This is because the single-task model does not consider the global tag dependency explicitly. To sum up, Korean morphological analysis should be solved by training the two tasks jointly with appropriate networks.

### 4.3 Error Analysis

Even if the proposed method achieves over 97.4% F1-score at the morpheme level, we believe that there still exists some room to improve the performance of Korean morphological analysis. To do this, we analyzed the errors by the proposed method.

Table 3 shows error types and their percentages. Morpheme segmentation is when all results are correct except some morphemes that are decomposed wrongly. This type happens mostly in Korean compound nouns where Korean compound nouns can be written as one or more eojeols. If a compound noun represented as one eojeol is given, the proposed method sometimes decomposes the compound noun into a series of nouns. 5.8% errors belong to this type. POS tagging error type is when morphemes are correctly recovered and segmented but their POS tags are predicted wrongly. Its percentage is 39.3%. In this type, 37% errors occur between noun and proper noun. The remaining but majority errors are related to morpheme recovery. That is, these errors occur when the proposed method fails in recovering morphemes from eojeols accurately. Its percentage is 54.9%. Therefore, it is inferred from the error analysis that developing a more accurate morpheme processing model is required to improve the performance of the proposed Korean morphological analyzer.

## 5 Conclusion

This paper has formulated Korean morphological analysis as a combination of morpheme processing and POS tagging. Thus, the two tasks are trained simultaneously through the tied sequence-to-sequence multi-task model with the pointer-generator network and the CRF network. According to the experiment results, the jointly trained morphological analyzer achieves higher performances than the legacy analyzers which are pipeline models of morpheme processing and POS tagging.[4]

## Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments on our work. This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2017-0-01772).

[4]Our code is available at https://github.com/songhyunje/kma

## References


Antonios Anastasopoulos and David Chiang. 2018. Tied multitask learning for neural speech translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 82–91.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations*.

Xinchi Chen, Xipeng Qiu, and Xuanjing Huang. 2017. A feature-enriched neural model for joint Chinese word segmentation and part-of-speech tagging. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 3960–3966.

Jihun Choi, Jonghem Youn, and Sang-goo Lee. 2016. A grapheme-level approach for constructing a Korean morphological analyzer without linguistic knowledge. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 3872–3879.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1723–1732.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

Georg Heigold, Guenter Neumann, and Josef van Genabith. 2016. Neural morphological tagging from characters for morphologically rich languages. *CoRR*, abs/1606.06640.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, pages 282–289.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270.

Do-Gil Lee and Hae-Chang Rim. 2009. Probabilistic modeling of Korean morphology. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(5):945–955.

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In *Proceedings of the 4th International Conference on Learning Representations*.

Christopher D. Manning. 2011. Part-of-speech tagging from 97% to 100%: Is it time for some linguistics? In *Proceedings of the 12th International Conference on Computational Linguistics and Intelligent Text Processing*, pages 171–189.

Andrew Matteson, Chanhee Lee, Youngbum Kim, and Heuiseok Lim. 2018. Rich character-level information for Korean morphological analysis and part-of-speech tagging. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2482–2492.

Seung-Hoon Na. 2015. Conditional random fields for Korean morpheme segmentation and POS tagging. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 14(3):10:1–10:16.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the AssociationS for Computational Linguistics*, pages 1073–1083.

Yan Shao, Christian Hardmeier, Jörg Tiedemann, and Joakim Nivre. 2017. Character-based joint segmentation and POS tagging for Chinese using bidirectional RNN-CRF. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing*, pages 173–183.

Hyun-Je Song and Seong-Bae Park. 2018. Korean part-of-speech tagging based on morpheme generation. TR-0002-2018, Kyung Hee University, Technical report.

Jae Jung Song. 2006. *The Korean language: Structure, use and context*. Routledge.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technology*, pages 173–180.

Yue Zhang and Stephen Clark. 2008. Joint word segmentation and POS tagging using a single perceptron. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 888–896.

# Efficient Convolutional Neural Networks for Diacritic Restoration

**Sawsan Alqahtani**[*2,3] and **Ajay Mishra**[1] and **Mona Diab**[1,2]
[1]AWS, Amazon AI
[2]The George Washington University
[3]Princess Nourah Bint Abdul Rahman University
sawsanq@gwu.edu, misaja@amazon.com, diabmona@amazon.com

## Abstract

Diacritic restoration has gained importance with the growing need for machines to understand written texts. The task is typically modeled as a sequence labeling problem and currently Bidirectional Long Short Term Memory (BiLSTM) models provide state-of-the-art results. Recently, Bai et al. (2018) show the advantages of Temporal Convolutional Neural Networks (TCN) over Recurrent Neural Networks (RNN) for sequence modeling in terms of performance and computational resources. As diacritic restoration benefits from both previous as well as subsequent timesteps, we further apply and evaluate a variant of TCN, Acausal TCN (A-TCN), which incorporates context from both directions (previous and future) rather than strictly incorporating previous context as in the case of TCN. A-TCN yields significant improvement over TCN for diacritization in three different languages: Arabic, Yoruba, and Vietnamese. Furthermore, A-TCN and BiLSTM have comparable performance, making A-TCN an efficient alternative over BiLSTM since convolutions can be trained in parallel. A-TCN is significantly faster than BiLSTM at inference time (270%∼334% improvement in the amount of text diacritized per minute).

## 1 Introduction

A diacritic is a mark that is added above, below, or within a letter, constructing a new letter or characterizing it with a different sound (Wells, 2000). In languages such as Arabic and Hebrew, some vowels are not part of the alphabet and diacritics are used for vowel specification. Moreover, many languages that include diacritics such as Yoruba and Vietnamese sometimes omit them in writing for various reasons such as difficulty in typing diacritics on keyboards or digitizing electronic text (Scannell, 2011).

Arabic, on the other hand, considers diacritics as an optional choice in writing even in formal settings,[1] and familiarity with the language is relied upon to derive the meaning based on context. Although people can extrapolate missing diacritics with near perfect accuracy in such languages, missing diacritics pose a challenge for computational models due to increased ambiguity.

Diacritization is the process of automatically restoring missing diacritics for each character in written text. It is important in many NLP applications such as automatic speech recognition (Vergyri and Kirchhoff, 2004) and speech synthesis (Ungurean et al., 2008). Most state-of-the-art diacritization models use Bidirectional Long Short Term Memory (BiLSTM) (Hochreiter and Schmidhuber, 1997) as a sequential classification problem, or as a sequence-to-sequence transduction problem to convert the original text into diacritized form (Orife, 2018; Zalmout and Habash, 2017; Belinkov and Glass, 2015). Generally speaking, LSTM has shown great success for sequential data, leveraging long range dependencies and preserving the temporal order of the sequence (Cho et al., 2014; Graves, 2013). However, LSTM requires intensive computational resources both for training and inference due to its sequential nature.

As an alternative, recent NLP technologies such as machine translation (Gehring et al., 2017) and language modeling (Dauphin et al., 2016) have investigated other models such as convolutional neural networks (CNN). Convolutional-based architectures utilize hierarchical rather than sequential relationships between the input elements. Tem-

[*]The work was done while the author was an intern at AWS, Amazon AI.

[1]With the exception of religious scripture and educational books for children, which are always diacritized.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1442–1448, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

poral Convolutional Networks (TCN) is a generic family of architectures that has been developed to alleviate the problem of training deep sequential models and is shown to provide significant improvement over LSTMs across different benchmarks (Bai et al., 2018; Lea et al., 2017). TCNs integrate *causal* convolutions where output at a certain time is convolved only with elements from earlier times in the previous layers. In addition, it is possible to train deep sequential models in parallel and use lower computational requirements rendering TCNs scalable to larger datasets.

In this study, we evaluate the application of the TCN architecture as described in (Bai et al., 2018) but devised as a character-level sequence model for diacritization since character-based models generalize better to unseen data compared to word-based models for most languages. Because diacritization is dependent on both past and future states, we further apply a variant of TCNs, namely *Acausal* Temporal Convolutional Network (A-TCN) (Lea et al., 2017), allowing the model to learn from previous as well as future context. We evaluate the architectures on three very different languages: Arabic, Vietnamese, and Yoruba.

To the best of our knowledge, TCN and A-TCN have not been investigated before for diacritization. In this paper, we show that A-TCN outperforms TCN while yielding comparable performance to BiLSTM but with the added advantage of being more efficient (speed and less computational footprint, i.e. decreased need for significant computational resources).

## 2 Related Work

A fair number of studies have been developed for the task of diacritization for different languages that include diacritics (Yarowsky, 1994; De Pauw et al., 2007; Scannell, 2011; Alqahtani et al., 2016, 2019). Feature engineering and classical machine learning algorithms such as Hidden Markov Models, Maximum Entropy Models, and Finite State Transducer were the dominant approaches (Nelken and Shieber, 2005; Zitouni et al., 2006; Elshafei et al., 2006). However, recent studies show significant improvement using deep neural networks (Belinkov and Glass, 2015; Pham et al., 2017; Orife, 2018). While these deep models achieve state-of-the-art performance, they mainly rely on the use of recurrent architectures such as BiLSTM, which are relatively inefficient.

Pham et al. (2017) view the task of diacritization for Vietnamese as a machine transduction problem from undiacritized to diacritized text at the word level. Orife (2018) addresses the problem on Yoruba in a similar way and compares soft- and self- attention sequence-to-sequence performance on the word level empirically showing that self-attention significantly outperforms BiLSTM.

Náplava et al. (2018) uses BiLSTM models with residual connections to train deeper models at the character level in several languages, including Vietnamese. For inference, they use beam search coupled with a language model to select among the possible diacritic patterns from the output. Zalmout and Habash (2017) develops a morphological disambiguation model to determine Arabic morphological features including diacritization. They train the model using BiLSTM while leveraging a LSTM-based language model as well as other morphological features to rank and score the output analyses.

## 3 Approach

We evaluate the performance of convolutional sequence models, TCN and A-TCN, in the context of diacritization and compare that to the recurrent sequential models, LSTM and BiLSTM. The task is formulated as a sequence classification such that we predict a diacritic for each character in the input.[2]

### 3.1 Convolutional Neural Network (TCN)

As TCN is a generic family of models, multiple architectures have been successfully developed (Dauphin et al., 2016; Kalchbrenner et al., 2016; Oord et al., 2016; Gehring et al., 2017; Lea et al., 2017; Bai et al., 2018). We choose the TCN architecture described in (Bai et al., 2018) [BAI18] as it integrates best techniques from the previous architectures while maintaining simplicity.

BAI18's TCN model satisfies the two main characteristics that allows sequential modeling: 1. Using 1-D fully convolutional network (Long et al., 2015) to ensure that the length of each layer is the same as the input by zero padding; 2. Using causal convolutions, which convolve output at time $t$ with elements from time $t-1$ and earlier in the previous layers. As Bai et al. (2018) indicated,

[2]We have also investigated BiLSTM-CRF, the model training time has been greatly increased without substantial improvement in terms of accuracy.

developing a TCN architecture that only considers these two characteristics restricts how far the model can utilize previous information. Thus, to ensure learning from longer effective history, BAI18 further integrate dilated convolutions (Yu and Koltun, 2015), enabling an exponentially large receptive field correlated with the depth of the network. To enable deeper learning, they integrate a residual block (He et al., 2016), in which each block includes two layers of dilated causal convolution as fully illustrated in (Bai et al., 2018).

### 3.2 Acausal Convolutional Neural Network (A-TCN)

TCN is beneficial for applications that restrict information flow from the past, such as language modeling (Bai et al., 2018; Dauphin et al., 2016); however, this is not sufficient for diacritization. Incorporating future and past context has been previously integrated in different versions of TCN (Lea et al., 2017; Gehring et al., 2017) but it either did not enhance the performance of their tasks or it was evaluated on convolutional sequence-to-sequence models rather than sequential models as in the case of our study.

To incorporate both future and past context, we relax the causality constraint by integrating acausal convolution rather than causal convolution, hence A-TCN, as illustrated in Figure 1. A-TCN is a tweaked variant of TCN such that the model convolves information from $x_{t-d}$ to $x_{t+d}$ (previous and following states) instead of $x_{t-d}$ to $x_t$ (previous states) only.[3] In our implementation, we use layer normalization (Ba et al., 2016) rather than weight normalization. The extent in which context is considered is influenced by the number of layers as well as the kernel size. Each character is constructed from itself and $k-1$ characters, where *(k: kernel size)* . Additionally, as we go deeper, the model further incorporates different additional $k-1$ characters, skipping $d-1$ characters, where *(d: dilation factor)*, to incorporate the character at the $d^{th}$ position from both sides (see Figure 1).

## 4 Experimental Setup

**Dataset:** For Arabic, we use the Arabic Treebank (ATB) dataset: parts 1, 2, and 3 and follow the same data division as (Diab et al., 2013).

[3]Lea et al. (2017) investigated acausal convolution in their architecture but reported insignificant improvement over causal convolution in their problem space.

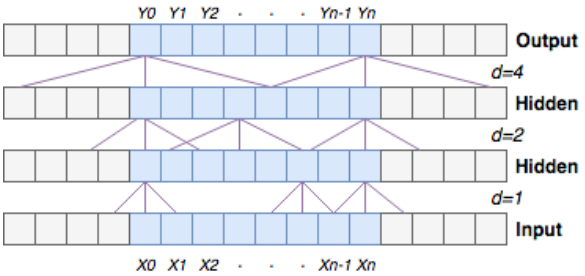


Figure 1: A dilated acausal convolution with 3 as a filter size and dilation factors equals to 1,2, and 4. White slots represent zero padding while colored slots represent characters. The remaining components of A-TCN are fully explained in (Bai et al., 2018).

| Data | Arabic | Vietnamese | Yoruba |
|---|---|---|---|
| Train | 502,938 | 800,022 | 800,771 |
| Test | 63,168 | 786,236 | 44,598 |
| Dev | 63,126 | 408,093 | 44,314 |
| OOV | 7.3% | 1.3% | 2.6% |

Table 1: Number of word tokens as well as Out-Of-Vocabulary (OOV) rate. OOV rate indicates the percentage of undiacritized words in the test set not observed during training.

We use datasets provided by Orife (2018) and Jakub Nplava (2018) for Yoruba and Vietnamese, respectively. Most languages, especially those that are severely impacted by diacritics, rarely have such amounts of diacritized datasets. Thus, we sample a moderate size subset of the training data for Vietnamese, roughly 3.7% to train the models. In the process, we remove from the training set all sentences that have at least one word of more than 10 characters,[4] or that do not have at least one diacritic, or that contain more than 70 words. Table 1 illustrates the dataset statistics.

To augment the dataset without requiring additional annotated dataset, we segment each sentence into space tokenized units;[5] each unit is further segmented into its characters and passed through the model along with a specific number of previous and future words. We add the special word boundary "<w>" between words with a window size of 10 words before and after the target word (21 words in total).[6]

**Parameter Setting:** The character embedding vectors are of dimension size 45, randomly ini-

[4]Vietnamese is characterized by having short word length.

[5]We did not investigate different types of tokenization. Relying on white space is simple and avoids additional preprocessing steps.

[6]We empirically tuned the window size on dev data.

tialized with a uniform distribution between [-0.1,0.1]. We use Adam optimization (Kingma and Ba, 2014) with 0.001 learning rate. For LSTM and BiLSTM, we use 3 layers and 250 hidden units. For TCN and A-TCN, we use 3 layers, 500 hidden units, and a kernel size of 5. Hidden units are initialized randomly using Xavier (Glorot and Bengio, 2010) with a magnitude of 3. For regularization, the dropout is set to 0.5 (Srivastava et al., 2014). We increase the dilation factor in TCN exponentially with the depth of the network.

## 5 Results

| System | DER | WER | OOV |
|---|---|---|---|
| **Arabic** | | | |
| Pasha et al. (2014) | - | 12.3% | 29.8% |
| Zalmout and Habash (2017) | - | 8.3% | **20.2%** |
| LSTM | 19.2% | 51.9% | 86.6% |
| TCN | 17.5% | 47.6% | 87.2% |
| BiLSTM | **2.8%** | **8.2%** | 33.6% |
| A-TCN | 3.0% | 10.2% | 36.3% |
| **Vietnamese** | | | |
| Náplava et al. (2018) | 11.2% | 44.5% | - |
| LSTM | 13.3% | 39.5% | 33.1% |
| TCN | 11.1% | 32.9% | 32.4% |
| BiLSTM | 2.6% | 7.8% | **15.3%** |
| A-TCN | **2.5%** | **7.7%** | **15.3%** |
| **Yoruba** | | | |
| Orife (2018) | - | **4.6%** | - |
| LSTM | 13.4% | 37.2% | 84.9% |
| TCN | 12.7% | 35.5% | 83.8% |
| BiLSTM | **3.6%** | **12.1%** | **69.3%** |
| A-TCN | **3.8%** | 12.6% | 70.2% |

Table 2: Models' performance on all words and OOV words per language. For Vietnamese, Náplava et al. (2018) reports 2.45% for WER on a much larger dataset (∼25M words), which is significantly better than our model.

We use two standard accuracy measures for evaluation: Diacritic Error Rate (DER) and Word Error Rate (WER). Table 2 shows the performance of the considered architectures across the sample languages.

**Importance of modeling past and future context:** A-TCN yields significant improvement over TCN, which indicates the importance of considering future and previous context for diacritization. Similarily BiLSTM significantly outperforms LSTM across all three languages. Overall, architectures that allow only unidirectional information flow provide lower performance than those that utilize context from both directions.

**Recurrent vs. convolutional architectures:** In line with prior work, across all three languages, TCN outperforms LSTM in the unidirectional architectures. A-TCN yields comparable results to BiLSTM except in the case of Arabic where the WER drops by ∼2%.

**OOV rate performance** To evaluate the model's robustness beyond observed training data, we specifically compare their WER performance on Out-Of-Vocabulary (OOV) words. BiLSTM has better ability to generalize on unseen data compared to A-TCN in Arabic and Yoruba, whereas both architectures are comparable in Vietnamese.

**Qualitative Analysis:** For Arabic, we further examine the impact of both BiLSTM and A-TCN on syntactic diacritics, aka inflectional diacritics which reflect syntactic case and mood primarily. These inflectional diacritics typically occur word finally in broken plurals and singular nouns as diacritics indicating syntactic case. Verbal mood is also marked word finally on verbs. We approximate the inflectional diacritics by computing the percentage of incorrectly predicted diacritics for the last characters of each word. BiLSTM yields a better performance (5.1% WER) compared to A-TCN (5.9% WER).

In addition, we randomly choose 20 sentences (605 words) to examine their categorical errors. We found similar errors in both architectures which include passive and active voice (e.g. نَشَرَت / na$arat[7] "spread" and نُشِرَت / nu$irat "been spread"), inflectional diacritics (e.g. وُصُولَ / wuSuwla and وُصُولُ / wuSuwlu, both mean "arrival" but have different syntactic positions), and named entities.[8] Thus, incorporating information from longer history in TCN architectures such as A-TCN compared to recurrent models such BiLSTM did not enhance learning inflectional diacritics.

We analyzed the confusion matrix and found that both architectures BiLSTM and A-TCN exhibit similar trends in the types of generated errors. In Arabic, the top two diacritics that are in-

[7]We adopt Buckwalter Transliteration encoding into Latin script for rendering Arabic text http://www.qamus.org/transliteration.htm.

[8]Diacritics in named entities are usually not consistent even among native speakers.

| Lang | BiLSTM | A-TCN | Difference | |
|---|---|---|---|---|
| | | | Time | Efficiency |
| AR | 376.85 | 132.55 | -64.8% | +284% |
| VI | 4187.81 | 1542.34 | -63.2% | +271% |
| YO | 461.19 | 138.45 | -70.0% | +334% |

Table 3: Inference time in seconds for each architecture across languages. *Efficiency* is defined in terms of text diacritization rate (amount of text diacritized per minute).

correctly diacritized by both A-TCN and BiLSTM (normalized by their frequencies) are $N$ and $\sim N$ which represent indefiniteness with and without doubling the consonants. The least diacritics that are incorrectly diacritized are no diacritics, sukun which represents the absence of diacritics, and two short vowels *i* and *a*. This is consistent with the frequency of diacritics in Arabic except that in the diacritic *u* which is also frequent but falls in the middle range of errors; this diacritic usually represents passive voice in Arabic which relies on syntactic relations between words. The same behavior can be found in Vietnamese and Yoruba where top and least erroneous diacritics are shared between both architectures.

**Efficiency Comparison:** We provide a comparative analysis of models' training and inference runtime (Table 3). During training time, A-TCN yields similar efficiency gains across all three languages with comparable performance in terms of accuracy. The convergence criteria was set as at least 1% improvement in accuracy from the previous epoch. During inference time, A-TCN was 2.7$\sim$3.3X faster in diacritizing text while providing comparable accuracy. This leads to 271$\sim$334% improvement in terms of text diacritization rate in the amount of text diacritized per minute. This supports our overall observation that A-TCN is a solid alternative to BiLSTM for this problem space due to its efficient performance especially for industrial settings where time is a crucial factor. All experiments were carried out on a single Tesla P100 GPU.

For Arabic, for instance, BiLSTM took $\sim$19 hours to converge while A-TCN took $\sim$14 hours. The DER were 2.8% and 3.0% respectively. Thus, A-TCN was 24% faster than BiLSTM while being marginally lower by 0.2% in terms of DER.

**Comparison to Prior Work:** Table 2 shows the performance of previous models trained on the same data. For Arabic, both A-TCN and BiLSTM provide significantly better performance than MADAMIRA (Pasha et al., 2014), which is a morphological disambiguation tool for Arabic. The performance of Zalmout and Habash (2017)'s model falls in between BiLSTM and A-TCN. As opposed to our character-based models, both previous models use other morphological features along with a language model to rank all possible diacritic choices. We believe that this additional semantic and morphological information help their models perform better on OOV words.

For Vietnamese, when we re-train Náplava et al. (2018) model on the same sample discussed in Section 4, both A-TCN and BiLSTM provide significantly better results. Náplava et al. (2018) also use BiLSTM but with different parameter settings and different dataset preparation.[9] For Yoruba, both character-based architectures provide lower performance than Orife (2018)'s model. However, Orife (2018) uses seq2seq modeling which generate diacritized sentences that are not of the same length as the input and can generate words not present in the original sentence (hallucinations). This is unpleasant behaviour for diacritization especially if used in text-to-speech applications.

## 6 Conclusion

In this study, we show that character-based convolutional architectures for diacritization yield comparable performance to both word and character based RNN ones for multiple languages, albeit at a significantly lower computational cost. Moreover, character based modeling yields better performance overall for the diacritization task.

In all cases, A-TCN performs much better than TCN, with a reduction of up to 40% in error rate, which means that using future information is crucial in diacritization. All in all, the decision whether to use A-TCN or BiLSTM is a trade-off between accuracy and efficiency. A-TCN provides efficient solutions with comparable accuracy and comes in more handy for applications that need predicting diacritics on the fly such as text-to-speech applications and simultaneous machine translation. A-TCN was 2.7$\sim$3.3X faster than BiLSTM at inference time (271%$\sim$334% improvement in the amount of text diacritized per minute), crucially at comparable accuracies, making it a very attractive solution for industrial applications in particular.

[9] We augment the dataset as explained in Section 4.

## References


Sawsan Alqahtani, Hanan Aldarmaki, and Mona Diab. 2019. Homograph disambiguation through selective diacritic restoration. In *Proceedings of the Fourth Arabic Natural Language Processing Workshop*.

Sawsan Alqahtani, Mahmoud Ghoneim, and Mona Diab. 2016. Investigating the impact of various partial diacritization schemes on arabic-english statistical machine translation. In *AMTA*.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling.

Yonatan Belinkov and James Glass. 2015. Arabic diacritization with recurrent neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2281–2285.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2016. Language modeling with gated convolutional networks. *arXiv preprint arXiv:1612.08083*.

Guy De Pauw, Peter W Wagacha, and Gilles-Maurice De Schryver. 2007. Automatic diacritic restoration for resource-scarce languages. In *International Conference on Text, Speech and Dialogue*, pages 170–179. Springer.

Mona Diab, Nizar Habash, Owen Rambow, and Ryan Roth. 2013. LDC Arabic treebanks and associated corpora: Data divisions manual.

Moustafa Elshafei, Husni Al-Muhtaseb, and Mansour Alghamdi. 2006. Statistical methods for automatic diacritization of Arabic text. In *The Saudi 18th National Computer Conference. Riyadh*, volume 18, pages 301–306.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Pavel Strak Jan Haji Jakub Nplava, Milan Straka. 2018. Diacritics restoration using neural networks. LREC.

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization.

Colin Lea, Michael D. Flynn, Rene Vidal, Austin Reiter, and Gregory D. Hager. 2017. Temporal convolutional networks for action segmentation and detection. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2015. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440.

Jakub Náplava, Milan Straka, Pavel Straňák, and Jan Hajič. 2018. Diacritics restoration using neural networks. In *Proceedings of the 11th Language Resources and Evaluation Conference*, Miyazaki, Japan. European Language Resource Association.

Rani Nelken and Stuart M Shieber. 2005. Arabic diacritization using weighted finite-state transducers. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*, pages 79–86. Association for Computational Linguistics.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.

Iroro Orife. 2018. Attentive sequence-to-sequence learning for diacritic restoration of Yorùbá language text. *Interspeech 2018*.

Arfath Pasha, Mohamed Al-Badrashiny, Mona T Diab, Ahmed El Kholy, Ramy Eskander, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan Roth. 2014. Madamira: A fast, comprehensive tool for morphological analysis and disambiguation of Arabic. In *LREC*, volume 14, pages 1094–1101.

Thai-Hoang Pham, Xuan-Khoai Pham, and Phuong Le-Hong. 2017. On the use of machine translation-based approaches for vietnamese diacritic restoration. *2017 International Conference on Asian Language Processing (IALP)*.

Kevin P. Scannell. 2011. Statistical unicodification of African languages. *Language resources and evaluation*, 45(3):375–386.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Cătălin Ungurean, Dragoş Burileanu, Vladimir Popescu, Cristian Negrescu, and Aurelian Dervis. 2008. Automatic diacritic restoration for a TTS-based e-mail reader application. *UPB Scientific Bulletin, Series C*, 70(4):3–12.

Dimitra Vergyri and Katrin Kirchhoff. 2004. Automatic diacritization of Arabic for acoustic modeling in speech recognition. In *COLING 2004 Computational Approaches to Arabic Script-based Languages*, pages 66–73, Geneva, Switzerland. COLING.

JC Wells. 2000. Orthographic diacritics and multilingual computing. *Language problems and language planning*, 24(3):249–272.

David Yarowsky. 1994. Decision lists for lexical ambiguity resolution. *Proceedings of the 32nd annual meeting on Association for Computational Linguistics -*.

Fisher Yu and Vladlen Koltun. 2015. Multi-scale context aggregation by dilated convolutions.

Nasser Zalmout and Nizar Habash. 2017. Don't throw those morphological analyzers away just yet: Neural morphological disambiguation for Arabic. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 704–713, Copenhagen, Denmark. Association for Computational Linguistics.

Imed Zitouni, Jeffrey S Sorensen, and Ruhi Sarikaya. 2006. Maximum entropy based restoration of Arabic diacritics. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 577–584. Association for Computational Linguistics.

# Improving Generative Visual Dialog by Answering Diverse Questions

**Vishvak Murahari**[1] **Prithvijit Chattopadhyay**[1]
**Dhruv Batra**[1,2] **Devi Parikh**[1,2] **Abhishek Das**[1]

[1]Georgia Tech [2]Facebook AI Research

## Abstract

Prior work on training generative Visual Dialog models with reinforcement learning (Das et al., 2017b) has explored a Q-BOT-A-BOT image-guessing game and shown that this 'self-talk' approach can lead to improved performance at the downstream dialog-conditioned image-guessing task. However, this improvement saturates and starts degrading after a few rounds of interaction, and does not lead to a better Visual Dialog model. We find that this is due in part to repeated interactions between Q-BOT and A-BOT during self-talk, which are not informative with respect to the image. To improve this, we devise a simple auxiliary objective that incentivizes Q-BOT to ask diverse questions, thus reducing repetitions and in turn enabling A-BOT to explore a larger state space during RL *i.e.* be exposed to more visual concepts to talk about, and varied questions to answer. We evaluate our approach via a host of automatic metrics and human studies, and demonstrate that it leads to better dialog, *i.e.* dialog that is more diverse (*i.e.* less repetitive), consistent (*i.e.* has fewer conflicting exchanges), fluent (*i.e.* more human-like), and detailed, while still being comparably image-relevant as prior work and ablations.

## 1 Introduction

Our goal is to build agents that can *see* and *talk i.e.* agents that can perceive the visual world and communicate this understanding in natural language conversations in English. To this end, Das et al. (2017a); de Vries et al. (2017) proposed the task of Visual Dialog – given an image, dialog history consisting of a sequence of question-answer pairs, and a follow-up question about the image, predict a free-form natural language answer to the question – along with a dataset and evaluation metrics[1].

Posing Visual Dialog as a supervised learning problem is unnatural. This is because at every round of dialog, the agent's answer prediction is thrown away and it gets access to ground-truth dialog history, thus disabling it from steering conversations during training. This leads to compounding errors over long-range sequences at test time, a problem also common in training recurrent neural networks for language modeling (Bengio et al., 2015; Ross et al., 2011; Ranzato et al., 2016).

To overcome this, Das et al. (2017b) devised a goal-driven approach for training Visual Dialog agents with deep reinforcement learning. This is formulated as a game between two agents – Q-BOT (that asks questions) and A-BOT (that answers questions). Q-BOT is shown a one-line description of an unseen image that A-BOT has access to and Q-BOT is allowed to ask questions (in natural language) to A-BOT for a fixed number of rounds and simultaneously make predictions of the unseen image. Both agents are rewarded for Q-BOT's image-guessing performance and trained with REINFORCE (Williams, 1992) to optimize this reward. Thus, there is incentive for Q-BOT to ask questions informative of the hidden image, and for A-BOT to provide meaningful answers.

While this reinforcement learning approach leads to improved performance on the image-guessing task than supervised learned agents, it has a few shortcomings – 1) image-guessing performance degrades after a few rounds of dialog (Fig. 2e), and 2) these improvements over supervised learning do not translate to an improved A-BOT, *i.e.* responses from this Visual Dialog agent are not necessarily better (on automatic metrics or human judgements), just that the Q-BOT-A-BOT pair is sufficiently in sync to do well at image-guessing.

We begin by understanding why this is the case, and find that Q-BOT-A-BOT dialog during 'self-talk' often tends to be repetitive *i.e.* the same question-answer pairs get repeated across rounds (Fig. 1a left). Since repeated interactions convey no additional information, image-guessing perfor-

[1] visualdialog.org, guesswhat.ai



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1449–1454, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

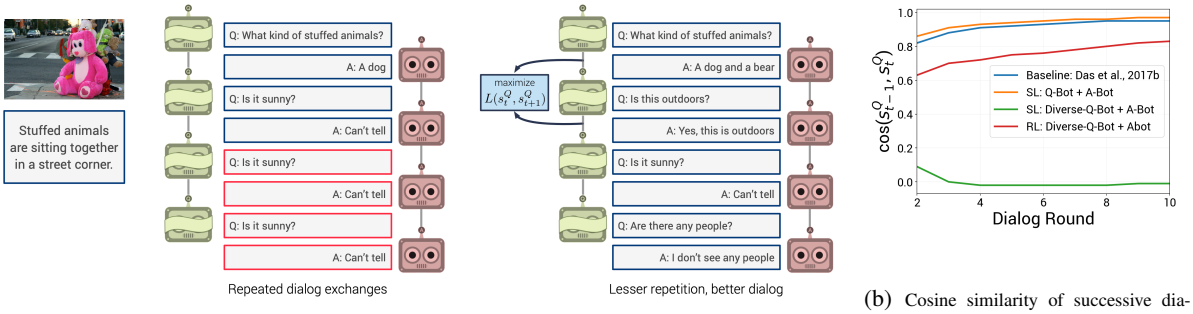


(a) **Left**. Prior work on training generative Visual Dialog models with RL on an image-guessing task between Q-BOT and A-BOT (Das et al., 2017b) leads to repetitive dialog. **Right**. We devise an auxiliary objective that incentivizes Q-BOT to ask diverse questions, thus reducing repetitions and enabling A-BOT to be exposed to more varied questions during RL, overall leading to better dialog, as measured by automatic metrics and human studies.

(b) Cosine similarity of successive dialog state embeddings within Q-BOT. Prior work (Das et al., 2017b) has high similarity. Our approach explicitly minimizes this similarity leading to more diverse dialog.

Figure 1

mance saturates, and even starts to degrade as the agent forgets useful context from the past.

These repetitions are due to high similarity in Q-BOT's context vectors of successive rounds driving question generation (Fig. 1b). To address this, we devise a smooth-L1 penalty that penalizes similarity in successive state vectors (Section 3). This incentivizes Q-BOT to ask diverse questions, thus reducing repetitions and in turn enabling A-BOT to explore a larger part of the state space during RL *i.e.* be exposed to more visual concepts to talk about, and varied questions to answer (Section 6).

Note that a trivial failure mode with this penalty is for Q-BOT to start generating diverse but totally image-irrelevant questions, which are not useful for the image-guessing task. A good balance between diversity and image-relevance in Q-BOT's questions is necessary to improve at this task.

We extensively evaluate each component of our approach against prior work and baselines:

- Q-BOT on diversity and image-relevance of generated questions during Q-BOT-A-BOT self-talk. We find that diverse-Q-BOT asks *more novel questions* while still being image-relevant.

- Q-BOT-A-BOT self-talk on consistency, fluency, level of detail, and human-interpretability, through automatic metrics and human studies. We find that diverse-Q-BOT-A-BOT dialog after RL is *more consistent, fluent, and detailed*.

- A-BOT on precision and recall of generated answers on the VisDial dataset (Das et al., 2017a) *i.e.* quality of answers to human questions. Training diverse-Q-BOT + A-BOT with RL does not lead to a drop in accuracy on VisDial.

## 2 Preliminaries

We operate in the same setting as Das et al. (2017b) – an image-guessing task between a questioner (Q-BOT) and an answerer (A-BOT) – where Q-BOT has to guess the image A-BOT has access to by asking questions in multi-round dialog.

We adopt the same training paradigm[2] consisting of 1) a supervised pre-training stage where Q-BOT and A-BOT are trained with Maximum Likelihood Estimation objectives on the VisDial dataset (Das et al., 2017a), and 2) a self-talk RL finetuning stage where Q-BOT and A-BOT interact with each other and the agents are rewarded for each successive exchange based on incremental improvements in guessing the unseen image. We learn parameterized policies $\pi_{\theta_Q}(q_t|s_{t-1}^Q)$ and $\pi_{\theta_A}(a_t|s_t^A)$ for Q-BOT and A-BOT respectively which decide what tokens to utter (*actions*: question $q_t$ and answer $a_t$ at every dialog round $t$) conditioned on the context available to the agent (*state* representations: $s_{t-1}^Q, s_t^A$). Q-BOT additionally makes an image feature prediction $\hat{y}_t$ at every dialog round, and the reward is $r_t = ||y^{gt} - \hat{y}_{t-1}||_2^2 - ||y^{gt} - \hat{y}_t||_2^2$, *i.e.* change in distance to the true representation $y^{gt}$ before and after a dialog round. We use REINFORCE (Williams, 1992) to update agent parameters, *i.e.* Q-BOT and A-BOT are respectively updated with $\mathbb{E}_{\pi_Q,\pi_A}[r_t \nabla_{\theta_Q} \log \pi_Q(q_t|s_{t-1}^Q)]$ and $\mathbb{E}_{\pi_Q,\pi_A}[r_t \nabla_{\theta_A} \log \pi_A(a_t|s_t^A)]$ as gradients.

Our transition from supervised to RL is gradual – we supervise for $N$ rounds and have policy-gradient updates for the remaining $10-N$, starting from $N = 9$ till $N = 4$, one round at a time. After reaching $N = 4$, repeating this procedure from $N = 9$ led to further (marginal) improvements.

[2] github.com/batra-mlp-lab/visdial-rl

Both Q-BOT and A-BOT are modeled as Hierarchical Recurrent Encoder-Decoder architectures (Serban et al., 2016). Q-BOT's fc7 (Simonyan and Zisserman, 2015) feature prediction of the unseen image ($\hat{y}$) is conditioned on the dialog history so far ($s^Q_{t-1}$) via a regression head[3].

## 3 Smooth-L1 Penalty on Question Repetition

Our goal is to encourage Q-BOT to ask a diverse set of questions so that when A-BOT is exposed to the same during RL finetuning, it is better able to explore its state space[4]. Furthermore, asking diverse questions allows Q-BOT-A-BOT exchanges across rounds to be more informative of the image, thus more useful for the image-guessing task.

We observe that agents trained using the paradigm proposed by Das et al. (2017b) suffer from repetition of *context* across multiple rounds of dialog – similar dialog state embeddings across multiple rounds leading to repeated utterances and similar predicted image representations, which consequently further increases similarity in state embeddings. Fig. 1b shows increasing $\cos(s^Q_{t-1}, s^Q_t)$ across dialog rounds for Das et al. (2017b).

To encourage Q-BOT to ask diverse questions, we propose a simple auxiliary loss that penalizes similar dialog state embeddings. Specifically, given Q-BOT states – $s^Q_{t-1}, s^Q_t$ – in addition to maximizing likelihood of question (during supervised pretraining), or image-guessing reward (during self-talk RL finetuning), we maximize a smooth-L1 penalty on $\Delta_t = \texttt{abs}(\left\|s^Q_{t-1}\right\|_2 - \left\|s^Q_t\right\|_2)$,

$$f(\Delta_t) = \begin{cases} 0.5\Delta_t^2 & \text{if } \Delta_t < 0.1 \\ 0.1(\Delta_t - 0.05) & \text{otherwise} \end{cases} \quad (1)$$

resulting in $\sum_{t=2}^{N} f(\Delta_t)$ as an additional term in the overall objective ($N$ = no. of dialog rounds).

Note that in order to maximize this penalty, Q-BOT has to push $s^Q_{t-1}$ and $s^Q_t$ further apart, which can only happen if $s^Q_{t-1}$ is updated using a question-answer pair that is different from the previous exchange, thus overall forcing Q-BOT to ask different questions in successive dialog rounds. Similar diversity objectives have also been explored in Li et al. (2016b) as reward heuristics.

[3]Please refer to appendix for architecture details.

[4]A-BOT's state-space is characterized by a representation of the question, image, and the dialog history so far.

Before arriving at (1), and following Fig. 1b, we also experimented with directly minimizing cosine similarity, $\cos(s^Q_{t-1}, s^Q_t)$. This led to the network learning large biases to flip the direction of successive $s^Q_{t-1}$ vectors (without affecting norms), leading to question repetitions in alternating rounds.

## 4 Experiments

**Baselines and ablations**. To understand the effect of the proposed penalty, we compare our full approach – '`RL: Diverse-Q-bot + A-bot`' – with the baseline setup in Das et al. (2017b), as well as several ablations – 1) '`SL: Q-bot + A-bot`': supervised agents (*i.e.* trained on VisDial data under MLE, no RL, no smooth-L1 penalty). Comparing to this quantifies how much our penalty + RL helps. 2) '`SL: Diverse-Q-bot + A-bot`': supervised agents where Q-BOT is trained with the smooth-L1 penalty. This quantifies gains from RL.

**Automatic Metrics.** To evaluate Q-BOT's *diversity* (Table 1), we generate Q-BOT-A-BOT dialogs (with beam size = 5) for 10 rounds on VisDial v1.0 val and compute 1) Novel Questions: the number of new questions (via string matching) in the generated dialog not seen during training, 2) Unique Questions: no. of unique questions per dialog instance (so $\leqslant 10$), 3) Dist-n and Ent-n (Zhang et al., 2018; Li et al., 2016a): the number and entropy of unique n-grams in the generated questions normalized by the total number of tokens, and 4) Mutual Overlap (Deshpande et al., 2019): BLEU-4 overlap of every question in the generated 10-round dialog with the other 9 questions, followed by averaging these 10 numbers. To measure Q-BOT's *relevance*, we report the negative log-likelihood under the model of human questions from VisDial. We evaluate A-BOT's answers to human questions from the VisDial dataset on the retrieval metrics introduced by Das et al. (2017a) (Table 2). Finally, we also evaluate performance of the Q-BOT-A-BOT pair at image-guessing (Fig. 2e), which is the downstream task they are trained with RL for.

**Human Studies**. To evaluate how human-understandable Q-BOT-A-BOT dialogs are, we conducted a study where we showed humans these dialogs (from our agents as well as baselines), along with a pool of 16 images from the VisDial v1.0 test-std split – consisting of the unseen image, 5 nearest neighbors (in fc7 space), and 10 random images – and asked humans to pick their top-5 guesses for the unseen image. Our hypothesis

| | Diversity | | | | | | | Relevance |
|---|---|---|---|---|---|---|---|---|
| | # Novel questions ↑ | # Unique questions ↑ | Mutual overlap ↓ | Ent-1 ↑ | Ent-2 ↑ | Dist-1 ↑ | Dist-2 ↑ | Negative log likelihood ↓ |
| Baseline: Das et al. (2017b) | 71 | 6.70 ± 0.07 | 0.58 ± 0.01 | 2.72 ± 0.01 | 3.03 ± 0.02 | 0.35 ± 0.0 | 0.43 ± 0.0 | **9.94** |
| SL: Q-BOT + A-BOT | 51 | 6.57 ± 0.07 | 0.60 ± 0.01 | 2.70 ± 0.01 | 3.00 ± 0.02 | 0.34 ± 0.0 | 0.42 ± 0.0 | 10.05 |
| SL: Diverse-Q-BOT + A-BOT | 146 | 7.45 ± 0.07 | 0.51 ± 0.01 | 2.82 ± 0.01 | 3.18 ± 0.01 | 0.38 ± 0.0 | 0.48 ± 0.0 | 10.10 |
| RL: Diverse-Q-BOT + A-BOT | **449** | **8.19** ± 0.06 | **0.41** ± 0.01 | **2.90** ± 0.01 | **3.31** ± 0.01 | **0.40** ± 0.0 | **0.53** ± 0.0 | 10.80 |

Table 1: Q-BOT diversity and relevance on v1.0 val. ↑ indicates higher is better. ↓ indicates lower is better.

| | v1.0 val | | | | | | v1.0 test-std | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | NDCG ↑ | MRR ↑ | R@1 ↑ | R@5 ↑ | R@10 ↑ | Mean Rank ↓ | NDCG ↑ | MRR ↑ | R@1 ↑ | R@5 ↑ | R@10 ↑ | Mean Rank ↓ |
| Baseline: Das et al. (2017b) | 53.76 | 46.35 | 36.22 | 56.15 | 62.41 | 19.34 | 51.60 | 45.67 | 35.05 | 56.30 | 63.25 | 19.15 |
| SL: A-BOT | 53.10 | 46.21 | 36.11 | 55.82 | 62.22 | 19.58 | 51.18 | 45.43 | 34.88 | 55.65 | 63.20 | 19.16 |
| RL: A-BOT (finetuned with Diverse-Q-BOT) | 53.91 | 46.46 | 36.31 | 56.26 | 62.53 | 19.35 | 51.67 | 45.64 | 34.85 | 56.55 | 63.43 | 18.96 |

Table 2: A-BOT performance on VisDial v1.0 (Das et al., 2017a). ↑ indicates higher is better. ↓ indicates lower is better.

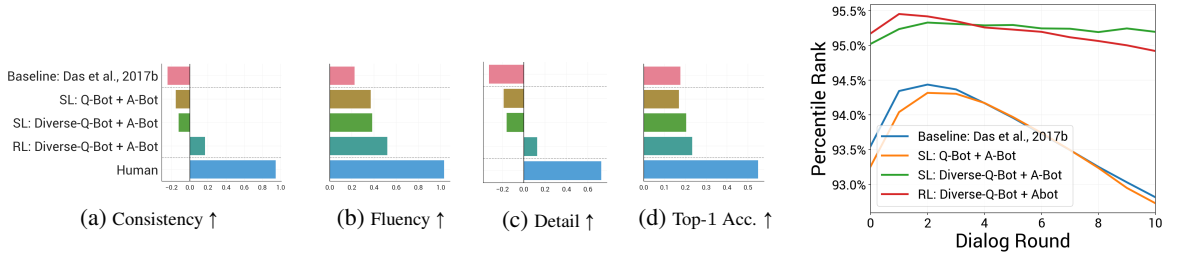


(a) Consistency ↑ (b) Fluency ↑ (c) Detail ↑ (d) Top-1 Acc. ↑

(e) Q-BOT-A-BOT image-guessing task performance

Figure 2: (**a-d**): Human evaluation of Q-BOT-A-BOT dialog over 50 images and 200 human subjects for each model variant. (**e**): Percentile rank (higher is better) of the true image (shown to A-BOT) as retrieved using fc7 feature predictions from Q-BOT.

was that if questions are more diverse, the dialog will be more image-informative, and so humans should be able to better guess which image was being talked about. We report top-1 accuracy of true image in human guesses. We also asked humans to rate Q-BOT-A-BOT dialog on consistency, fluency, and level of detail on a 5-point Likert scale.

## 5 Implementation Details

We used beam search with a beam size of 5 during self-talk between all Q-BOT-A-BOT variants. NDCG scores on the v1.0 val split and the total SL loss (on the same split) were used to select the best SL A-BOT and Q-BOT checkpoints respectively. We used a dropout rate of 0.5 for all SL-pretraining experiments and no dropout for RL-finetuning. We used Adam (Kingma and Ba, 2015) with a learning rate of $10^{-3}$ decayed by $\sim$0.25 every epoch, upto a minimum of $5 \times 10^{-5}$.

The objective for training Diverse-Q-BOT was a sum of the smooth-L1 penalty (introduced in Section 3), cross entropy loss, and L2 loss between the regression head output and the fc7 (Simonyan and Zisserman, 2015) embedding of the image. We observed that coefficients in the range of $1e^{-3}$ to $1e^{-5}$ worked best for the smooth-L1 penalty. We also observed that training for a large number of epochs ($\sim$80) with the above mentioned range of coefficient values led to the best results.

## 6 Results

- **Q-BOT's diversity** (Table 1): The question-repetition penalty consistently increases diversity (in both SL and RL) over the baseline! `RL: Diverse-Q-bot` asks ~1.5 more unique questions on average than Das et al. (2017b) (6.70 → 8.19) for every 10-round dialog, ~6.3x more novel questions (71 → 449), and a higher fraction and entropy of unique generated n-grams, while still staying comparably relevant (NLL).
- **A-BOT on VisDial** (Table. 2): `RL: A-bot` outperforms `SL: A-bot`, but does not statistically improve over the baseline on answering human questions from VisDial[5] (on v1.0 val & test-std).
- **Image-guessing task** (Fig. 2e): `Diverse-Q-bot + A-bot` (SL and RL) significantly outperform the baseline on percentile rank of ground-truth image as retrieved using Q-BOT's fc7 prediction. Thus, the question-repetition penalty leads to a more informative communication protocol.
- **Human studies** (Fig. 2): Humans judged `RL: Diverse-Q-bot + A-bot` dialog significantly more consistent (fewer conflicting exchanges), fluent (fewer grammatical errors), and detailed (more image-informative) over the baseline and supervised learning. This is an important result. Performance on GuessWhich, together

[5]This is consistent with trends in Das et al. (2017b).

with these dialog quality judgements from humans show that agents trained with our approach develop a more effective communication protocol for the downstream image-guessing task, while still not deviating off English, which is a common pitfall when training dialog agents with RL (Kottur et al., 2017; Lewis et al., 2017).

Note that since our penalty (Eqn. 1) is structured to avoid repetition across successive rounds, one possible failure mode is that Q-BOT learns to ask the same question every alternate dialog round (at $t$ and $t+2$). Empirically, we find that this happens $\sim$15% of times (2490 times out of $\sim$16.5k question pairs) on v1.0 val for RL: Diverse Q-BOT + A-BOT compared to $\sim$22% for SL: Q-BOT + A-BOT. This observation, combined with the fact that Diverse-Q-BOT asks $\sim$1.6 more unique questions relative to SL: Q-BOT across 10 rounds suggests that simply incentivizing diversity in successive rounds works well empirically. We hypothesize that this is because repeating questions every other round or other such strategies to game our repetition penalty is fairly specific behavior that is likely hard for models to learn given the large space of questions Q-BOT could potentially ask.

## 7 Related Work

Our work is related to prior work in visual dialog (Das et al., 2017a; de Vries et al., 2017) and modeling diversity in text-only dialog (Zhang et al., 2018; Li et al., 2016a,b).

Closest to our setting is work on using conditional variational autoencoders for self-talk in visual dialog (Massiceti et al., 2018), where diversity is not explicitly modeled but is measured via metrics specific to the proposed architecture.

Adding constraints to generate a diverse set of natural language dialog responses have previously been explored in Zhang et al. (2018) via adversarial information maximization, in Gao et al. (2019) by jointly modeling diversity and relevance in a shared latent space, and in Li et al. (2016a) using a maximum mutual information criterion. In contrast, we are interested in diversity at the level of the entire dialog (instead of a single round) – reducing repetitions in QA pairs across multiple rounds. Our repetition penalty is partly inspired by the 'Information Flow' constraint in Li et al. (2016b). As detailed in Section 2, we experimented with similar forms of the penalty and eventually settled on smooth-L1. To the best of our knowledge, we are the first to explicitly model diversity as a constraint in visual dialog.

## 8 Conclusions & Future Work

We devised an auxiliary objective for training generative Visual Dialog agents with RL, that incentivizes Q-BOT to ask diverse questions. This reduces repetitions in Q-BOT-A-BOT dialog during RL self-talk, and in turn enables A-BOT to be exposed to a larger state space. Through extensive evaluations, we demonstrate that our Q-BOT-A-BOT pair has significantly more diverse dialog while still being image-relevant, better downstream task performance, and higher consistency, fluency, and level of detail than baselines. Our code will be publicly released, and we hope this will serve as a robust base for furthering progress on training visual dialog agents with RL for other multi-agent grounded language games, adapting to learn to talk about novel visual domains, *etc*.

## 9 Acknowledgements

We thank Nirbhay Modhe and Viraj Prabhu for the PyTorch implementation (Modhe et al., 2018) of Das et al. (2017b) that we built on, and Jiasen Lu for helpful discussions. The Georgia Tech effort is supported in part by NSF, AFRL, DARPA, ONR YIPs, ARO PECASE. AD is supported in part by fellowships from Facebook, Adobe, and Snap Inc. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the US Government, or any sponsor.

## References

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *NIPS*. 1

Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José M.F. Moura, Devi Parikh, and Dhruv Batra. 2017a. Visual Dialog. In *CVPR*. 1, 2, 3, 4, 5

Abhishek Das, Satwik Kottur, José M.F. Moura, Stefan Lee, and Dhruv Batra. 2017b. Learning Cooperative Visual Dialog Agents with Deep Reinforcement Learning. In *ICCV*. 1, 2, 3, 4, 5

Aditya Deshpande, Jyoti Aneja, Liwei Wang, Alexander G Schwing, and David Forsyth. 2019. Fast, di-

verse and accurate image captioning guided by part-of-speech. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10695–10704. 3

Xiang Gao, Sungjin Lee, Yizhe Zhang, Chris Brockett, Michel Galley, Jianfeng Gao, and Bill Dolan. 2019. Jointly optimizing diversity and relevance in neural response generation. *arXiv preprint arXiv:1902.11205*. 5

Diederik Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*. 4

Satwik Kottur, José MF Moura, Stefan Lee, and Dhruv Batra. 2017. Natural language does not emerge 'naturally' in multi-agent dialog. In *EMNLP*. 5

Mike Lewis, Denis Yarats, Yann N Dauphin, Devi Parikh, and Dhruv Batra. 2017. Deal or no deal? end-to-end learning for negotiation dialogues. In *EMNLP*. 5

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. 3, 5

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016b. Deep Reinforcement Learning for Dialogue Generation. In *EMNLP*. 3, 5

Daniela Massiceti, N Siddharth, Puneet K Dokania, and Philip HS Torr. 2018. Flipdial: A generative model for two-way visual dialogue. In *CVPR*. 5

Nirbhay Modhe, Viraj Prabhu, Michael Cogswell, Satwik Kottur, Abhishek Das, Stefan Lee, Devi Parikh, and Dhruv Batra. 2018. VisDial-RL-PyTorch. 5

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *ICLR*. 1

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*. 1

Iulian Vlad Serban, Alberto García-Durán, Çaglar Gülçehre, Sungjin Ahn, Sarath Chandar, Aaron C. Courville, and Yoshua Bengio. 2016. Generating Factoid Questions With Recurrent Neural Networks: The 30M Factoid Question-Answer Corpus. In *ACL*. 3

Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In *ICLR*. 3, 4

Harm de Vries, Florian Strub, Sarath Chandar, Olivier Pietquin, Hugo Larochelle, and Aaron Courville. 2017. GuessWhat?! visual object discovery through multi-modal dialogue. In *CVPR*. 1, 5

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256. 1, 2

Yizhe Zhang, Michel Galley, Jianfeng Gao, Zhe Gan, Xiujun Li, Chris Brockett, and Bill Dolan. 2018. Generating informative and diverse conversational responses via adversarial information maximization. In *NIPS*. 3, 5

# Cross-lingual Transfer Learning with Data Selection for Large-Scale Spoken Language Understanding

**Quynh Do**
Amazon
Aachen, Germany
doquynh@amazon.com

**Judith Gaspers**
Amazon
Aachen, Germany
gaspers@amazon.de

## Abstract

A typical cross-lingual transfer learning approach boosting model performance on a resource-poor language is to pre-train the model on *all* available supervised data from another resource-rich language. However, in large-scale systems, this leads to high training times and computational requirements. In addition, characteristic differences between the source and target languages raise a natural question of whether *source-language data selection* can improve the knowledge transfer. In this paper, we address this question and propose a simple but effective language model based source-language data selection method for cross-lingual transfer learning in large-scale spoken language understanding. The experimental results show that with data selection i) the source data amount and hence training speed is reduced significantly and ii) model performance is improved.

## 1 Introduction

Spoken Language Understanding (SLU) plays an important role in spoken language technology and is typically divided into two sub-tasks: Intent Classification (IC) and Slot Filling (SF). While the former identifies a speaker's intent, the latter extracts semantic constituents from the natural language query. Recently, there have been emerging efforts on Cross-Lingual Transfer Learning (CLTL) methods to reduce data requirements in deep neural network (DNN) based SLU. A typical approach is to pre-train the model on labeled data from a richly resourced language, and then either apply it directly on a target language (Upadhyay et al., 2018) or fine-tune it on a smaller amount of supervised data from a target language (Do and Gaspers, 2019). However, both in SLU and other NLP tasks, prior work on CLTL typically utilized all available source data to transfer knowledge, as it has been mainly investigated in rather small Academic settings. However, in large-scale settings with millions of utterances, this would lead to costly training times, high computational requirements and optimization difficulties. Moreover, the different characteristics between the source and target languages raise a natural question of whether source-language data selection in which only the most relevant source instances are picked for pre-training can improve CLTL performance, as a considerable amount of source utterances might be "irrelevant" to the target language or even yield negative transfer.

Addressing these questions, in this paper we explore source-language data selection for CLTL in SLU, focusing especially on large-scale settings in which we assume the existence of a *large* amount (millions) of source data and a *moderate* amount (thousands) of target data. Since the effectiveness of pre-training in CLTL depends on the similarity of the source data distribution and the real distribution of the target language, we propose a source-language data selection method which computes the relevance score of each source instance to the target language by using several N-gram language model based metrics. Our method is designed to satisfy *both* IC and SF sub-tasks in a multi-task training scenario, and to select data from *multiple* source languages, which have been rarely studied in the literature.

Our experimental results show that our proposed data selection method: i) improves CLTL performance in large-scale settings, while reducing the amount of source data significantly ii) brings higher gains to SF but does not hurt IC, and iii) can select data efficiently from multiple source languages for a single target language.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1455–1460, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

## 2 Related work

Prior work on CLTL for SLU has mainly focused on using machine translation (e.g. García et al. (2012); He et al. (2013); Gaspers et al. (2018)). Until recently, few approaches based on cross-lingual joint training and cross-lingual supervised pre-training for DNNs have been proposed. The former takes advantage of the knowledge transferring in a SLU system by jointly training (relatively balanced) source and target data (e.g. Li et al. (2018)). Meanwhile, the latter is usually used when the amount of source data is significantly larger than the amount of target data. In particular, the SLU model is pre-trained on a large amount of supervised source data, and then either tested directly on the target language (e.g. Upadhyay et al. (2018)), or fine-tuned on a smaller amount of supervised target data (e.g. Do and Gaspers (2019)). Our CLTL method follows the line of Do and Gaspers (2019), but instead of utilizing all available source data, we aim at selecting the most relevant subset of the source data for the target language.

Data selection has been studied in the field of domain adaptation with most of the work targeting machine translation (Axelrod et al., 2011; van der Wees et al., 2017). These approaches usually rank sentence pairs in a large bitext from a source domain according to their difference in cross-entropy or perplexity with respect to a target domain corpus and then select the top $n$ sentence pairs to train a machine translation system for the target domain. Although this task also deals with multiple languages, it is not a CLTL problem. The application of data selection on other tasks are relatively rare, e.g., dependency parsing (Plank and van Noord, 2011), sentiment analysis (Remus, 2012), POS tagging (Ruder and Plank, 2017).

Several common data metrics have been proposed to rank the relevance of the source instances to the target domain, e.g, word similarity measures, diversity. However, to the best of our knowledge, data selection has *not yet* been explored for DNN-based CLTL in SLU. In addition, two challenges tackled in this paper, i.e. applying data selection for a multi-task training scenario and dealing with multiple source languages, have been rarely studied in the literature.

## 3 Spoken language understanding

### 3.1 Task definition

Suppose for a language $l$ with word vocabulary $\mathcal{V}_l$, intent vocabulary $\mathcal{I}_l$ and slot vocabulary $\mathcal{S}_l$, we have a set of utterances which are annotated with an intent label and each word is annotated with a slot label. The task of SLU is divided into two sub-tasks: i) *Intent classification*, which learns a function mapping each unlabeled utterance to a proper intent label $\in \mathcal{I}_l$, and ii) *Slot filling*, which learns a function mapping each unlabeled token to a proper slot label $\in \mathcal{S}_l$.

### 3.2 Model

Our multi-task SLU model consists of: i) A shared embedding layer which is the concatenation of a 1-dimensional convolution neural network based character embedding and a word embedding. ii) A shared encoder which is a two-layers bi-directional highway Long-short Term Memory network (Srivastava et al., 2015) served by the embedding layer as inputs, learning a contextual, fixed-dimensional representation for each token. ii) Two decoders for SF and IC sub-tasks; each consists of a stack of two dense layers and a softmax layer on top.

The two sub-tasks are trained jointly via a weighted loss function: $L = \alpha_i \hat{L}_i + \alpha_s \hat{L}_s$, where $\hat{L}_i, \hat{L}_s$ are the normalized cross-entropy losses with label smoothing (Szegedy et al., 2016) of IC and SF, respectively.

### 3.3 Cross-lingual transfer learning

Given a target language $l^t$ with a limited supervised data set $\mathbf{D}_{l^t}$ divided into a training set $\mathbf{D}_{l^t}^T$ and a validation set $\mathbf{D}_{l^t}^V$, CLTL aims at improving the SLU performance on $l^t$ by leveraging the larger supervised data sets $\mathbf{D}_{l_1^s}, \ldots, \mathbf{D}_{l_N^s}$ from $N$ source languages $l_1^s, \ldots, l_N^s$. A common idea behind CLTL methods is to map the source and target data into a shared space, so that the knowledge can be transferred in-between languages.

In this paper, we assume the availability of a multi-lingual word embedding function which maps a word in any language into a shared space: $\mathcal{W} : \mathcal{V}_{l_1^s} \bigcup \ldots \mathcal{V}_{l_N^s} \bigcup \mathcal{V}_{l^t} \rightarrow \mathbb{R}^d$, where $\mathcal{V}_l$ is the vocabulary of language $l$. In addition, for a source language $l_i^s$ in which a bilingual dictionary $\mathcal{D}_{l_i^s, l^t} : \mathcal{V}_{l_i^s} \rightarrow \mathcal{V}_{l^t}$ is available, a word $w$ can be alternatively mapped into the shared space by using $\mathcal{W}(\mathcal{D}_{l_i^s, l^t}(w))$ [1]. The word embedding layer in our

[1] Experiments on the development data shows that

SLU model is fixed to the mapping from the word vocabulary to this shared space, without being updated during training. In contrast, the character embedding layer in our model is initialized randomly and updated during training. Our CLTL training strategy consists of two phases: First, the model is *pre-trained* on the source data $\mathbf{D}_{l_1^s} \cup \ldots \cup \mathbf{D}_{l_N^s}$ for $T_w^s$ epochs, and validated on $\mathbf{D}_{l^t}^V$. Second, the model is *fine-tuned* on the target data $\mathbf{D}_{l^t}^T$ for $T_w^t$ epochs and validated on $\mathbf{D}_{l^t}^V$.

## 4 Data selection

The effectiveness of pre-training in CLTL depends on the similarity of the source data distribution and the real distribution of the target language. Let us consider each component in our model. First, for the word embedding layer, obtaining similar distributions of the source data and the target language can be considered as an "easy" task by using multilingual word embedding. Second, for the character embedding layer and the encoder, it depends on how similar the character patterns and the word patterns of the source data and the target language are, respectively. Finally, for the decoders, the similar distributions could be expected given the good distributions provided by the encoder.

We, therefore, propose a relevance metric for the source utterances w.r.t. the target language: $R(u) = \sum_{k=1}^{M} \alpha_k f_k(u)$, where $f_k$ and $\alpha_k$ are respectively an attribute value and its importance weight, and $M$ is the total number of attributes. Each attribute is associated with an N-gram word- or character- based language model trained on the target language which can be used to estimate the similarity of a pattern to the target language. Let us consider an attribute $f_k$ and its N-gram language model $\mathrm{LM}_k$ trained on the target language $l_t$. Given an utterance $u = w_1 \ldots w_n$ in a source language $l_i^s$ and the bilingual dictionary $\mathcal{D}_{l_i^s, l^t} : \mathcal{V}_{l_i^s} \rightarrow \mathcal{V}_{l^t}$ mapping a word in $l_i^s$ to another word in $l^t$, we call $\mathbf{S}$ the set of N-grams generated from $\mathcal{D}_{l_i^s, l^t}(w_1) \ldots \mathcal{D}_{l_i^s, l^t}(w_n)$. The attribute value $f_k$ is computed as the average language model score of the elements in $\mathbf{S}$:

$$f_k(u) = \sum_{g \in \mathbf{S}} \mathrm{LM}_k(g) * \frac{1}{|\mathbf{S}|} \qquad (1)$$

$\mathcal{W}(\mathcal{D}(w))$ works well for French as target language, while $\mathcal{W}(w)$ is better for German.

| Exp. | Source | Target | | |
|---|---|---|---|---|
| | | Train | Dev | Test |
| 10K-DE | EN 5M | 10K | 2K | 7,431 |
| 20K-DE | EN 5M | 20K | 2K | 7,431 |
| 10K-DE | EN 5M, DE 1.1M, ES 114,702 | 10K | 5K | 58K |
| 20K-DE | EN 5M, DE 1.1M, ES 114,702 | 20K | 5K | 58K |

Table 1: Supervised data statistics.

We then normalize $f_k(u)$ at intent level:

$$\bar{f}_k(u) = \frac{f_k(u)}{\max_{u' \in \mathbf{I}_u} f_k(u')} \qquad (2)$$

where $\mathbf{I}_u$ is the set of utterances (from all source languages) having the same intent as $u$. By using the proposed relevance metric, the source data can be ranked in descending order, and only the top-K utterances will be selected for the pre-training.

## 5 Experiments

### 5.1 Data

**Supervised data** For large-scale experiments, we extracted random samples from a large-scale SLU system. The data are representative of user requests to voice-enabled devices and are labeled with intents and slots. We include four languages into our experiments, i.e. English (EN), German (DE), French (FR) and Spanish (ES). DE and FR are used as the target languages in our experiments. Data statistics can be found in Table 1.

**Unlabelled data sets** For each of the target languages (DE and FR), we build N-gram language models on unlabelled data sets in that language. We make use of 3M DE sentences and 1M FR sentences which are freely available from the Leipzig unlabelled corpus collection (Goldhahn et al., 2012). In addition, we collect 500K DE and 2.5K FR unlabelled utterances with a similar nature as labelled utterances from the SLU system.

**Pre-trained resources** We use pre-trained 300-dimensional multilingual word embeddings and bilingual dictionaries provided by Conneau et al. (2017) and Lample et al. (2017), respectively.

### 5.2 Setup

We carry out four experiments with 10K and 20K target data in DE and FR (see Table. 1 for the experiment names and the labelled data statistics). We conduct experiments with transferring from one source language (EN) to another (DE) and from

three source languages (EN, DE, ES) to one target language (FR). In each experiment, we compare four different training strategies: i) Mono (w.o. CLTL): an SLU model is trained on *only* the supervised *target* data. ii) CLTL: *all* supervised source data is used for pre-training, and we fine-tune on target language data. iii) CLTL-RD: random K% utterances of the original source data are used for pre-training. iv) CLTL-DS: the K% most relevant source utterances are selected by using our proposed relevance metric for pre-training.

**Settings** The convolutions used for character embeddings have window sizes of 2, 3, 4, each consisting of 64 filters. The sizes of LSTM and dense layers are set to 300. All dropout keep probabilities are set to 0.9. The hyper-parameter tuning on the development set results in $\alpha_i = 0.2, \alpha_s = 0.8$, label smoothing rate $= 0.1$. We use Adam optimizer with learning rate $= 0.001$. Exponential decay is applied to the learning rate with decay steps $= 500$, and decay rate $= 0.95$. For CLTL, the number of training epochs $T_w^s$ and $T_w^t$ are set to 6 and 25, respectively. For data selection, we use four N-gram language models, i.e. word-based bi-gram and tri-gram language models and character-based bi-gram and tri-gram language models. The four importance weights are set to 1.0 each. For evaluation we use the standard metrics, i.e. F1, precision and recall for slot filling (computed using the CoNLL 2002 script) and accuracy for intent classification.

### 5.3 Results and discussions

| Exp. | Model | Slot | | | Intent |
|---|---|---|---|---|---|
| | | *P* | *R* | *F1* | *Acc.* |
| **10K-DE** | Mono | 76.4 ± 1.6 | 75.4 ± 1.4 | 75.9 ± 1.5 | 87.9 ± 0.4 |
| | CLTL | 79.7 ± 1.8 | 77.6 ± 1.1 | 78.7 ± 1.5 | 89.5 ± 0.3 |
| | CLTL-RD | 79.3 ± 0.9 | 77.0 ± 0.5 | 78.1 ± 0.6 | 89.5 ± 0.4 |
| | CLTL-DS | 80.1 ± 0.6 | 78.6 ± 0.7 | **79.4 ± 0.6** | **90.0 ± 0.3** |
| **20K-DE** | Mono | 80 ± 0.2 | 78.8 ± 1.0 | 79.4 ± 0.6 | 89.5 ± 0.0 |
| | CLTL | 81.3 ± 1.8 | 78.9 ± 2.3 | 80.1 ± 2.1 | 90.5 ± 0.1 |
| | CLTL-RD | 80.7 ± 1.6 | 79.3± 0.9 | 80.0 ± 1.2 | 90.1 ± 0.3 |
| | CLTL-DS | 82.2 ± 0.2 | 80.7 ± 0.1 | **81.5 ± 0.1** | **90.8 ± 0.4** |
| **10K-FR** | Mono | 76.5 ± 0.5 | 78.4 ± 0.5 | 77.5 ± 0.5 | 89.3 ± 0.4 |
| | CLTL | 79.0 ± 0.3 | 80.7 ± 0.4 | 79.8 ± 0.1 | 90.7 ± 0.1 |
| | CLTL-RD | 78.7 ± 0.2 | 80.6 ± 0.3 | 79.7 ± 0.2 | 90.4 ± 0.3 |
| | CLTL-DS | 80.0 ± 0.7 | 82.0 ± 0.2 | **81.0 ± 0.3** | **91.2 ± 0.2** |
| **20K-FR** | Mono | 78.9 ± 0.1 | 80.2 ± 0.2 | 79.5 ± 0.2 | 90.6 ± 0.1 |
| | CLTL | 80.4 ± 0.6 | 82.4 ±0.5 | 81.4 ± 0.6 | 91.4 ± 0.2 |
| | CLTL-RD | 80.9 ± 0.1 | 82.5 ± 0.5 | 81.7 ± 0.2 | 91.5 ± 0.1 |
| | CLTL-DS | 81.5 ± 0.2 | 82.8 ± 0.2 | **82.1 ± 0.2** | **91.6 ± 0.2** |

Table 2: Performance of CLTL on large-scale data sets. K is set to 50 (%) in CLTL-RD and CLTL-DS. Reported results are the mean and std. values of 3 runs.

**Is 100% better than 50%?** Table 2 shows the performances of the different training strategies in our experiments. In CLTL-RD and CLTL-DS settings, K is set to 50 (%). It helps to answer the question that whether using the full source data (100%) is better than using just 50%. Interestingly, although 100% (CLTL) is better than random 50% (CLTL-RD) in 3 out of 4 experiments, it is surpassed by our selected 50% in all of the experiments. These results do not only prove the effectiveness of our proposed data selection metric but also suggest a potentially powerful application of source-language data selection on cross-lingual transfer learning.

**Slot filling vs. intent classification** As shown in Table 2source-language, our data selection method tends to bring higher gains to SF than to IC. One possible reason is that IC is the easier between the two sub-tasks (less categories, single label vs. sequence label decoding etc.). However, it is important to stress that our data selection does not hurt IC, meaning that the method is useful for joint learning.

**One vs several source languages** The similar trends in experiments with DE as target and FR as target show that our proposed data selection method can be applied in both single-source and multi-source transfer learning. In order to compare the utility of multiple vs a single source, we ran an experiment on 10K-FR using only English as the source language. The means of slot F1 and IC dropped from 81.0% and 91.2% to 79.9% and 90.8%, respectively, when using only English, potentially because it is not the closest language to French. Hence, the model could probably choose better source utterances from multiple sources. This indicates that our method works for both settings with higher gains for using multiple source languages.

**Value of K and importance weights of the language models?** One may question whether it is possible to optimize the importance weights of the language models and choose K automatically. A possible solution could be using Bayesian optimization (Ruder and Plank, 2017). Instead of selecting the top K% utterances, we define a threshold $\theta$: $u$ is selected if $R(u) >= \theta$. $\theta$ and $\alpha_k$ become hyper-parameters which can be optimized using Bayesian Optimization with the score of the CLTL model on a development set as the objective. However, while

this could improve performance, it is expensive, especially in large-scale settings, i.e. optimization would likely add more computation time than what we can gain by training on a subset.

**Performance on a small-scale benchmark dataset** While we are mainly interested in large-scale SLU, we also strive for further understanding of CLTL by conducting a similar experiment on a small-scale widely-used SLU benchmark data set, i.e. ATIS (Tür et al., 2010). It contains audio recordings and corresponding annotated transcriptions in English of people making flight reservations.

To compare with the state-of-the-art systems, we apply our monolingual model on ATIS. The model reaches 95.6% F1 for slot filling and 96.8% accuracy for intent classification which are *comparable* to the state-of-the-art results reported on the same data set (Do and Gaspers, 2019).

We then perform a cross-lingual transfer learning experiment from English to German on ATIS. To construct the training sets of the target language, we select two random subsets of 200 and 400 English utterances from the training part of ATIS and translate them into German. The development set of the target language is formed by the German translation of a random subset of 144 English utterances selected from the validation part of ATIS. The test set of the target language is the German translation version of the ATIS test set which includes 893 utterances. The annotated source data comprise 4015 English training utterances from ATIS.

| Exp. | Model | Slot | | | Intent |
|---|---|---|---|---|---|
| | | *P* | *R* | *F1* | *Acc.* |
| **200-DE** | Mono | 79.8 ± 0.3 | 80.6 ± 0.6 | 80.2 ± 0.6 | 85.8 ± 0.6 |
| | CLTL | 81.8 ± 0.5 | 82.8 ± 0.9 | 82.3 ± 0.6 | 85.3 ± 1.6 |
| | CLTL-RD | 83.0 ± 1.6 | 83.3 ± 1.7 | 83.1 ± 1.6 | 86.8 ± 1.4 |
| | CLTL-DS | 84.8 ± 2.7 | 85.3 ± 2.3 | **85.0 ± 2.5** | **87.2 ± 1.9** |
| **400-DE** | Mono | 86.8 ± 1.1 | 87.7 ± 0.6 | 87.2 ± 0.8 | 88.2 ± 0.7 |
| | CLTL | 88.0 ± 0.5 | 87.8 ± 0.7 | 87.9 ± 0.6 | 88.0 ± 0.3 |
| | CLTL-RD | 87.8 ± 1.3 | 87.4 ± 1.1 | 87.6 ± 1.1 | 87.9 ± 0.8 |
| | CLTL-DS | 88.3 ± 0.1 | 88.0 ± 0.3 | **88.2 ± 0.2** | **88.2 ± 0.2** |

Table 3: Performance of CLTL on ATIS. K is set to 50 (%) in CLTL-RD and CLTL-DS. Reported results are the mean and std. values of 3 runs.

Table 3 shows the performance of CLTL on the small-scale ATIS data set. As can been seen, in general, CLTL with data selection (CLTL-DS) is still beneficial in a small-scale setting.

## 6 Conclusions

We presented an efficient approach to select source data for cross-lingual transfer learning in large-scale SLU. Our results indicate that by using data selection we can both improve performance and reduce source data significantly without a negative effect on system performance, which can reduce training time and computational requirements in large-scale systems greatly. This suggests an interesting future research direction toward data selection for cross-lingual transfer learning problems.

## References


Amittai Axelrod, Xiaodong He, and Jianfeng Gao. 2011. Domain adaptation via pseudo in-domain data selection. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 355–362, Stroudsburg, PA, USA. Association for Computational Linguistics.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Quynh Ngoc Thi Do and Judith Gaspers. 2019. Cross-lingual transfer learning for spoken language understanding. *Proceedings of the 2019 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*.

Fernando García, Lluís F. Hurtado, Encarna Segarra, Emilio Sanchis, and Giuseppe Riccardi. 2012. Combining multiple translation systems for spoken language understanding portability. In *2012 IEEE Spoken Language Technology Workshop (SLT), Miami, FL, USA, December 2-5, 2012*, pages 194–198.

Judith Gaspers, Penny Karanasou, and Rajen Chatterjee. 2018. Selecting machine-translated data for quick bootstrapping of a natural language understanding system. *Proceedings of NAACL-HLT*.

Dirk Goldhahn, Thomas Eckart, and Uwe Quasthoff. 2012. Building large monolingual dictionaries at the leipzig corpora collection: From 100 to 200 languages. In *LREC*, pages 759–765. European Language Resources Association (ELRA).

X. He, L. Deng, D. Hakkani-Tur, and G. Tur. 2013. Multi-style adaptive training for robust cross-lingual spoken language understanding. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8342–8346.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2017. Unsupervised machine translation using monolingual corpora only. *arXiv preprint arXiv:1711.00043*.

Yujiang Li, Xuemin Zhao, Weiqun Xu, and Yonghong Yan. 2018. Cross-lingual multi-task neural architecture for spoken language understanding. In *Proc. Interspeech 2018*, pages 566–570.

Barbara Plank and Gertjan van Noord. 2011. Effective measures of domain similarity for parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 1566–1576, Stroudsburg, PA, USA. Association for Computational Linguistics.

R. Remus. 2012. Domain adaptation using domain similarity- and domain complexity-based instance selection for cross-domain sentiment analysis. In *2012 IEEE 12th International Conference on Data Mining Workshops*, pages 717–723.

Sebastian Ruder and Barbara Plank. 2017. Learning to select data for transfer learning with bayesian optimization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 372–382. Association for Computational Linguistics.

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2377–2385. Curran Associates, Inc.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826.

Gökhan Tür, Dilek Hakkani-Tür, and Larry P. Heck. 2010. What is left to be understood in atis? In *SLT*, pages 19–24. IEEE.

Shyam Upadhyay, Manaal Faruqui, Gökhan Tür, Dilek Z. Hakkani-Tür, and Larry P. Heck. 2018. (almost) zero-shot cross-lingual spoken language understanding. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018, Calgary, AB, Canada, April 15-20, 2018*, pages 6034–6038.

Marlies van der Wees, Arianna Bisazza, and Christof Monz. 2017. Dynamic data selection for neural machine translation. *CoRR*, abs/1708.00712.

# Multi-Head Attention with Diversity for Learning Grounded Multilingual Multimodal Representations

**Po-Yao Huang**[1], **Xiaojun Chang**[2], **Alexander Hauptmann**[1]
[1]Language Technologies Institute, Carnegie Mellon University
[2]Faculty of Information Technology, Monash University
poyaoh@cs.cmu.edu, cxj273@gmail.com, alex@cs.cmu.edu

## Abstract

With the aim of promoting and understanding the multilingual version of image search, we leverage visual object detection and propose a model with diverse multi-head attention to learn grounded multilingual multimodal representations. Specifically, our model attends to different types of textual semantics in two languages and visual objects for fine-grained alignments between sentences and images. We introduce a new objective function which explicitly encourages attention diversity to learn an improved visual-semantic embedding space. We evaluate our model in the German-Image and English-Image matching tasks on the Multi30K dataset, and in the Semantic Textual Similarity task with the English descriptions of visual content. Results show that our model yields a significant performance gain over other methods in all of the three tasks.

## 1 Introduction

Joint visual-semantic embeddings (VSE) are central to the success of many vision-language tasks, including cross-modal search and retrieval (Kiros et al., 2014; Karpathy and Fei-Fei, 2015; Gu et al., 2018), visual question answering (Antol et al., 2015; Goyal et al., 2017), multimodal machine translation (Huang et al., 2016; Elliott and Kádár, 2017), etc. Learning VSE requires extensive understanding of the content in individual modalities and an in-depth alignment strategy to associate the complementary information from multiple views.

With the availability of large-scale parallel English-Image corpora (Lin et al., 2014; Young et al., 2014), a rich line of research has advanced learning VSE under the monolingual setup. Most recent works (Kiros et al., 2014; Vendrov et al., 2015; Karpathy and Fei-Fei, 2015; Klein et al., 2015; Wang et al., 2016, 2018; Huang et al., 2019b) leverage triplet ranking losses to align English sentences and images in the joint embedding space. In VSE++ (Faghri et al., 2018), Faghri *et al.* improve VSE by emphasizing hard negative samples. Recent advancement in VSE models explores methods to enrich the English-Image corpora. Shi et al. (2018) propose to augment dataset with textual contrastive adversarial samples to combat adversarial attacks. Recently, Huang et al. (2019a) utilize textual semantics of regional objects and adversarial domain adaptation for learning VSE under low-resource constraints.

An emerging trend generalizes learning VSE in the multilingual scenario. Rajendran et al. (2016) learn $M$-view representations when parallel data is available only between one pivot view and the rest of views. PIVOT (Gella et al., 2017) extends the work from Calixto et al. (2017) to use images as the pivot view for learning multilingual multimodal representations. Kádár et al. (2018) further confirms the benefits of multilingual training.

Our work is motivated by Gella et al. (2017) but has important differences. First, to disentangle the alignments in the joint embedding space, we employ visual object detection and multi-head attention to selectively align salient visual objects with textual phrases, resulting in visually-grounded multilingual multimodal representations. Second, as multi-head attention (Vaswani et al., 2017) is appealing for its efficiency and ability to jointly attend to information form different perspectives, we propose to further encourage the diversity among attention heads to learn an improved visual-semantic embedding space. Figure 1 illustrates our gradient updates promoting diversity. The proposed model achieves state-of-the-art performance in the multilingual sentence-image matching tasks in Multi30K (Elliott et al., 2016) and the semantic textual similarity task (Agirre et al., 2012, 2014, 2015).



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1461–1467, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

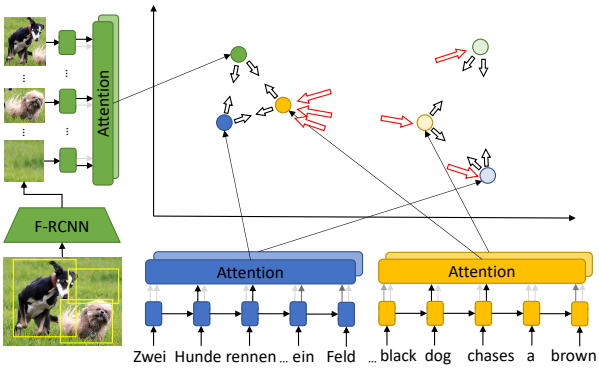


Figure 1: Multi-head attention with diversity for learning grounded multilingual multimodal representations. (A two-headed example with a part of diversity loss $l_\theta^D$ colored in red.)

## 2 Related Works

Classic attention mechanisms have been addressed for learning VSE. These mechanisms can be broadly categorized by the types of {*Query*, *Key*, *Value*} as discussed in Vaswani et al. (2017). For intra-modal attention, {*Query*, *Key*, *Value*} are within the same modality. In DAN (Nam et al., 2017), the content in each modalities is iteratively attended through multiple steps with intra-modal attention. In SCAN (Lee et al., 2018), inter-modal attention is performed between regional visual features from Anderson et al. (2018) and text semantics. The inference time complexity is $O(MN)$ (for generating $M$ query representations for a size $N$ datatset). In contrast to the prior works, we leverage intra-modal multi-head attention, which can be easily parallelized compared to DAN and is with a preferred $O(M)$ inference time complexity compared to SCAN.

Inspired by the idea of attention regularization in Li et al. (2018), for learning VSE, we propose a new margin-based diversity loss to encourage a margin between attended outputs over multiple attention heads. Multi-head attention diversity within the same modality and across modalities are jointly considered in our model.

## 3 The Proposed Model

Figure 1 illustrates the overview of the proposed model. Given a set of images as the pivoting points with the associate English and German[1] descriptions or captions, the proposed VSE model aims to learn a multilingual multimodal embedding space in which the encoded representations $(v, e, g)$ of a paired instance $(i, x^e, x^g)$ are closely aligned to each other than non-paired ones.

**Encoders**: For a sampled $(i, x^e, x^g)$ pair, we first encode the tokens in the English sentence $x^e = \{x_1^e, \ldots, x_N^e\}$ and the tokens in the German sentence $x^g = \{x_1^g, \ldots, x_N^g\}$ through the word embedding matrices followed by two bi-directional LSTMs. The outputs of the textual encoders are $e = \{e_1, \ldots, e_N\}, e_n \in \mathbb{R}^H$ for English and $g = \{g_1, \ldots, g_N\}, g_n \in \mathbb{R}^H$ for German, where $N$ is the max sentence length and $H$ is the dimension of the shared embedding space. For the image, we leverage a Faster-RCNN (Ren et al., 2015) network with a ResNet (He et al., 2016) backbone to detect and encode salient visual objects in the image. With a trainable one-layered perceptron to transform visual features into the shared embedding space, we encode the image as $v = \{v_1, \ldots, v_M\}, v_m \in \mathbb{R}^H$, where $M$ is the maximum amount of visual objects in an image.

**Multi-head attention with diversity**: We employ $K$-head attention networks to attend to the visual objects in an image as well as the textual semantics in a sentence then generate fixed-length image/sentence representations for alignment. Specifically, the $k$-th attended German sentence representation $g^k$ is computed by:

$$a_i^k = \text{softmax}(\tanh(W_{c_g}^k c_g^k)^\top \tanh(W_g^k g_i)) \quad (1)$$

$$g^k = \sum_{i=1}^{N} a_i^k g_i, \quad (2)$$

where $a_i^k$ is the attention weight, $W_g^k \in \mathbb{R}^{H \times H_{attn}}, W_{c_g}^k \in \mathbb{R}^{H_c \times H_{attn}}$ is the learnable transformation matrix for German. $c_g^k \in \mathbb{R}^{H_c}$ is the learnable $H_c$-dimensional contextual vector for distilling important semantics from German sentences. The final German sentence representation is the concatenation of $K$-head attention outputs $g = [g^0 \| g^1 \| \ldots \| g^K]$. Similar for encoding the English sentence $e = [e^0 \| e^1 \| \ldots \| e^K]$ and the image $v = [v^0 \| v^1 \| \ldots \| v^K]$.

With $\{V, E, G\}$ where $v \in V, e \in E, g \in G$ as the set of attended fixed-length image and sentence representations in a sampled batch, we use the widely-used hinge-based triplet ranking loss with hard negative mining (Faghri et al., 2018) to align instances in the visual-semantic embedding space. Taking Image-English instances $\{V, E\}$ as

[1]For clarity in notation, we discuss only two languages. The proposed model can be intuitively generalized to more languages by summing additional terms in Eq. 4 and Eq. 6-7.

an example, we leverage the triplet correlation loss defined as:

$$l_\theta(V,E) = \sum_p \left[\alpha - s(v_p, e_p) + s(v_p, \hat{e}_p)\right]_+ + \sum_q \left[\alpha - s(v_q, e_q) + s(\hat{v}_q, e_q)\right]_+, \quad (3)$$

where $\alpha$ is the correlation margin between positive and negative pairs, $[.]_+ = \max(0, .)$ is the hinge function, and $s(a,b) = \frac{a^T b}{\|a\|\|b\|}$ is the cosine similarity. $p$ and $q$ are the indexes of the images and sentences in the batch. $\hat{e}_p = \text{argmax}_q s(v_p, e_{q\neq p})$ and $\hat{v}_q = \text{argmax}_p s(v_{p\neq q}, e_q)$ are the hard negatives. When the triplet loss decreases, the paired images and German sentences are drawn closer down to a margin $\alpha$ than the hardest non-paired ones. Our model aligns $\{V, E\}$, $\{V, G\}$ and $\{E, G\}$ in the joint embedding space for learning multilingual multimodal representations with the sampled $\{V, E, G\}$ batch. We formulate the overall triplet loss as:

$$l_\theta(V,E,G) = l_\theta(V,G) + l_\theta(V,E) + \gamma l_\theta(G,E). \quad (4)$$

Note that the hyper-parameter $\gamma$ controls the contribution of $l_\theta(G,E)$ since ($e$, $g$) may not be a translation pair even though ($e$, $v$) and ($g$, $v$) are image-caption pairs.

One of the desired properties of multi-head attention is its ability to jointly attend to and encode different information in the embedding space. However, there is no mechanism to support that these attention heads indeed capture diverse information. To encourage the diversity among $K$ attention heads for instances within and across modalities, we propose a new simple yet effective margin-based diversity loss. As an example, the multi-head attention diversity loss between the sampled images and the English sentences (*i.e.* diversity across-modalities) is defined as:

$$l^D_\theta(V,E) = \sum_p \sum_k \sum_r \left[\alpha_D - s(v^k_p, e^{k\neq r}_p)\right]_+ \quad (5)$$

As illustrated with the red arrows for update in Figure 1, the merit behind this diversity objective is to increase the distance (up to a diversity margin $\alpha_D$) between attended embeddings from different attention heads for an instance itself or its cross-modal parallel instances. As a result, the diversity objective explicitly encourage multi-head attention to concentrate on different aspects of information sparsely located in the joint embedding space to promote fine-grained alignments between multilingual textual semantics and visual objects. With the fact that the shared embedding space is multilingual and multimodal, for improving both intra-modal/lingual and inter-modal/lingual diversity, we model the overall diversity loss as:

$$l^D_\theta(V,E,G) = l^D_\theta(V,V) + l^D_\theta(G,G) + l^D_\theta(E,E) + l^D_\theta(V,E) + l^D_\theta(V,G) + l^D_\theta(G,E), \quad (6)$$

where the first three terms are intra-modal/lingual and the rest are cross-modal/lingual. With Eq. 4 and Eq. 6, we formalize the final model loss as:

$$l^{All}_\theta(V,E,G) = l_\theta(V,E,G) + \beta l^D_\theta(V,E,G), \quad (7)$$

where $\beta$ is the weighting parameter which balances the diversity loss and the triple ranking loss. We train the model by minimizing $l^{All}_\theta(V,E,G)$.

## 4 Experiments

Following Gella et al. (2017), we evaluate on the multilingual sentence-image matching tasks in Multi30K (Elliott et al., 2016) and the semantic textual similarity task (Agirre et al., 2012, 2015).

### 4.1 Experiment Setup

We use the model in Anderson et al. (2018) which is a Faster-RCNN (Ren et al., 2015) network pre-trained on the MS-COCO (Lin et al., 2014) dataset and fine-tuned on the Visual Genome (Krishna et al., 2016) dataset to detect salient visual objects and extract their corresponding features. 1,600 types of objects are detectable. We then pack and represent each image as a $36 \times 2048$ feature matrix where 36 is the maximum amount of salient visual objects in an image and 2048 is the dimension of the flattened last pooling layer in the ResNet (He et al., 2016) backbone of Faster-RCNN.

For the text processing, we lower-case, tokenize, and then truncate the maximum sentence length to 100. We use 300-dim word embedding matrices initialized either randomly or with pre-trained multilingual embeddings. (We use the multilingual version of FastText (Mikolov et al., 2018)). We also experiment incorporating the last layer of contextualized multilingual BERT embeddings (Devlin et al., 2018) to replace the word embedding matrices as the textual input features for the bi-directional LSTMs.

| Method | German to Image | | | Image to German | | | English to Image | | | Image to English | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | R@1 | R@5 | R10 | R@1 | R@5 | R@10 | R@1 | R@5 | R@10 | R@1 | R@5 | R10 |
| VSE$^{\dagger*}$ (Kiros et al., 2014) | 20.3 | 47.2 | 60.1 | 29.3 | 58.1 | 71.8 | 23.3 | 53.6 | 65.8 | 31.6 | 60.4 | 72.7 |
| OE$^{\dagger*}$ (Vendrov et al., 2015) | 21.0 | 48.5 | 60.4 | 26.8 | 57.5 | 70.9 | 25.8 | 56.5 | 67.8 | 34.8 | 63.7 | 74.8 |
| DAN$^{*}$ (Nam et al., 2017) | 31.0 | 60.9 | 71.0 | 46.5 | 77.5 | 83.0 | 39.4 | 69.2 | 69.1 | 55.0 | 81.8 | 89.0 |
| VSE++$^{*}$ (Faghri et al., 2018) | 31.3 | 62.2 | 70.9 | 47.5 | 78.5 | 84.5 | 39.6 | 69.1 | 79.8 | 53.1 | 82.1 | 87.5 |
| SCAN$^{*}$ (Lee et al., 2018) | 35.7 | 64.9 | 74.6 | 52.3 | 81.8 | 88.5 | 45.8 | 74.4 | 83.0 | 61.8 | 87.5 | 93.7 |
| Pivot$^{\dagger}$ (Gella et al., 2017) | 22.5 | 49.3 | 61.7 | 28.2 | 61.9 | 73.4 | 26.2 | 56.4 | 68.4 | 33.8 | 62.8 | 75.2 |
| Ours$^{\dagger}$ (Random, VGG19) | 25.8 | 54.9 | 65.1 | 34.1 | 65.5 | 76.5 | 30.1 | 62.5 | 71.6 | 36.4 | 68.0 | 80.9 |
| Ours (Random, No diversity) | 36.3 | 65.3 | 74.7 | 53.1 | 82.3 | 88.8 | 46.2 | 74.7 | 82.9 | 63.3 | 87.0 | 93.3 |
| Ours (Random) | 39.2 | 67.5 | 76.7 | 55.0 | 84.7 | 91.2 | 48.7 | 77.2 | 85.0 | 66.4 | 88.3 | 93.4 |
| Ours (w/ FastText) | 40.3 | 70.1 | **79.0** | **60.4** | **85.4** | **92.0** | **50.1** | 78.1 | 85.7 | **68.0** | 88.8 | 94.0 |
| Ours (w/ BERT) | **40.7** | **70.5** | 78.8 | 56.5 | 84.6 | 91.3 | 48.9 | **78.3** | **85.8** | 66.5 | **89.1** | **94.1** |

Table 1: Comparison of multilingual sentence-image retrieval/matching (German-Image) and (English-Image) results on Multi30K. (Visual encoders:VGG$^{\dagger}$ otherwise ResNet or Faster-RCNN(ResNet).) (Monolingual models$^{*}$.)

| Method | MS-Vid (2012) | Pascal (2014) | Pascal (2015) |
|---|---|---|---|
| STS Baseline | 29.9 | 51.3 | 60.4 |
| STS Best System | 86.3 | 83.4 | 86.4 |
| GRAN (Wieting et al., 2017) | 83.7 | 84.5 | 85.0 |
| VSE (Kiros et al., 2014) | 80.6 | 82.7 | 89.6 |
| OE (Vendrov et al., 2015) | 82.2 | 84.1 | 90.8 |
| DAN (Nam et al., 2017) | 84.1 | 84.3 | 90.8 |
| VSE++ (Faghri et al., 2018) | 84.5 | 84.8 | 91.2 |
| SCAN (Lee et al., 2018) | 84.0 | 83.9 | 90.7 |
| PIVOT (Gella et al., 2017) | 84.6 | 84.5 | 91.5 |
| Ours (w/ Random) | 85.8 | 87.8 | 91.5 |
| Ours (w/ FastText) | 86.2 | **88.3** | **91.8** |
| Ours (w/ BERT) | **86.4** | 88.0 | 91.7 |

Table 2: Results on the image and video datasets of Semantic Textual Similarity task. (Pearson's $r \times 100$ ).

For training, we sample batches of size 128 and train 20 epochs on the training set of Multi30K. We use the Adam (Kingma and Ba, 2014) optimizer with $2 \times 10^{-4}$ learning rate then $2 \times 10^{-5}$ after 15-*th* epoch. Models with the best summation of validation R@1,5,10 are selected to generate image and sentence embeddings for testing. Weight decay is set to $10^{-6}$ and gradients larger than 2.0 are clipped. We use 3-head attention ($K = 3$) and the embedding dimension $H = 512$. The same dimension is shared by all the context vectors in the attention modules. Other hyperparameters are set as follows: $\alpha = 0.2, \alpha_D = 0.1, \beta = 1.0$ and $\gamma = 0.6$.

### 4.2 Multilingual Sentence-Image Matching

We evaluate the proposed model in the multilingual sentence-image matching (retrieval) tasks on Multi30K: (i) Searching images with text queries (Sentence-to-Image). (ii) Ranking descriptions with image queries (Image-to-Sentence). English and German are considered.

Multi30K (Elliott et al., 2016) is the multilingual extension of Flickr30K (Young et al., 2014). The training, validation, and testing split contain 29K, 1K, and 1K images respectively. Two types of annotations are available in Multi30K: (i) One parallel English-German translation for each image and (ii) five independently collected English and five German descriptions/captions for each image. We use the later. Note that the German and English descriptions are *not* translations of each other and may describe an image differently.

As other prior works, we use recall at $k$ (R@$k$) to measure the standard ranking-based retrieval performance. Given a query, R@$k$ calculates the percentage of test instances for which the correct one can be found in the top-$k$ retrieved instances. Higher R@$k$ is preferred.

Table 1 presents the results on the Multi30K testing set. The VSE baselines in the first five rows are trained with English and German descriptions independently. In contrast, PIVOT (Gella et al., 2017) and the proposed model are capable of handling multilingual input queries with single model. For a fair comparison with PIVOT, we also report the result of swapping Faster-RCNN with VGG as the visual feature encoder in our model.

As can be seen, the proposed models successfully obtain state-of-the-art results, outperforming other baselines by a significant margin. German-Image matching benefit more from joint training with English-Image pairs. The models with pre-trained multilingual embeddings and contextualized embeddings achieve better performance in comparison to randomly initialized word embeddings, especially for German. One explanation is that the degradation from German singletons is alleviated by the multi-task training with English and the pre-trained embeddings. While the model with BERT performs better in English, FastText is preferred for German-Image matching.




Figure 2: Qaulitative text-to-image matching results on Multi30K. Correct (colored in green) if ranked first.

### 4.3 Semantic Textual Similarity Results

For semantic textual similarity (STS) tasks, we evaluate on the video task from STS-2012 (Agirre et al., 2012) and the image tasks from STS-2014-15 (Agirre et al., 2014, 2015). The video descriptions are from the MSR video description corpus (Chen and Dolan, 2011) and the image descriptions are from the PASCAL dataset (Rashtchian et al., 2010). In STS, a system takes two input sentences and output a semantic similarity ranging from [0,5]. Following Gella et al. (2017), we directly use the model trained on Multi30K to generate sentence embeddings then scaled the cosine similarity between the two embeddings as the prediction.

Table 2 lists the standard Pearson correlation coefficients $r$ between the system predictions and the STS gold-standard scores. We report the best scores achieved by paraphrastic embeddings (Wieting et al., 2017) (text only) and the VSE models in the previous section. Note that the compared VSE models are all with RNN as the text encoder and *no* STS data is used for training. Our models achieve the best performance and the pre-trained word embeddings are preferred.

### 4.4 Qualitative Results and Grounding

In Figure 2 we samples some qualitative multilingual text-to-image matching results. In most cases our model successfully retrieve the one and only one correct image. Figure 3 depicts the t-SNE visualization of the learned visually grounded multilingual embeddings of the $(v, e, g)$ pairs pivoted on $v$ in the Multi30K testing set. As evidenced,




Figure 3: t-SNE visualization and grounding of the learned multilingual multimodal embeddings on Multi30K. Note the sentences are *not* translation pairs.

although the English and German sentences describe different aspects of the image, our model correctly aligns the shared semantics (*e.g.* ("*man*", "*mann*"), ("*hat*", "*wollmtze*")) in the embedding space. Notably, the embeddings are visually-grounded as our model associate the multilingual phrases with exact visual objects (*e.g. glasses* and *ears*). We consider learning grounded multilingual multimodal dictionary as the promising next step.

As limitations we notice that actions and small objects are harder to align. Additionally, the alignments tends to be noun-phrase/object-based whereas spatial relationships (*e.g.* "*on*", "*over*") and quantifiers remain not well-aligned. Resolving these limitations will be our future work.

## 5 Conclusion

We have presented a novel VSE model facilitating multi-head attention with diversity to align different types of textual semantics and visual objects for learning grounded multilingual multimodal representations. The proposed model obtains state-of-the-art results in the multilingual sentence-image matching task and the semantic textual similarity task on two benchmark datasets.

## Acknowledgement

This research is supported in part by the DARPA grants FA8750-18-2-0018 and FA8750-19-2-0501 under AIDA and LwLL program. It is also supported by the IARPA grant via DOI/IBC number D17PC00340. We would like to thank the anonymous reviewers for their constructive suggestions.

## References


Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Inigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, et al. 2015. Semeval-2015 task 2: Semantic textual similarity, english, spanish and pilot on interpretability. In *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*, pages 252–263.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. Semeval-2014 task 10: Multilingual semantic textual similarity. In *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, pages 81–91.

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. In ** SEM 2012: The First Joint Conference on Lexical and Computational Semantics–Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, volume 1, pages 385–393.

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and visual question answering. In *CVPR*.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. VQA: Visual Question Answering. In *International Conference on Computer Vision (ICCV)*.

Iacer Calixto, Qun Liu, and Nick Campbell. 2017. Multilingual multi-modal embeddings for natural language processing. *arXiv preprint arXiv:1702.01101*.

David Chen and William Dolan. 2011. Collecting highly parallel data for paraphrase evaluation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 190–200.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Desmond Elliott, Stella Frank, Khalil Sima'an, and Lucia Specia. 2016. Multi30k: Multilingual english-german image descriptions. In *Proceedings of the 5th Workshop on Vision and Language*, pages 70–74. Association for Computational Linguistics.

Desmond Elliott and Ákos Kádár. 2017. Imagination improves multimodal translation. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 130–141, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Fartash Faghri, David J Fleet, Jamie Ryan Kiros, and Sanja Fidler. 2018. Vse++: Improving visual-semantic embeddings with hard negatives.

Spandana Gella, Rico Sennrich, Frank Keller, and Mirella Lapata. 2017. Image pivoting for learning multilingual multimodal representations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2839–2845. Association for Computational Linguistics.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Jiuxiang Gu, Jianfei Cai, Shafiq R. Joty, Li Niu, and Gang Wang. 2018. Look, imagine and match: Improving textual-visual cross-modal retrieval with generative models. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Po-Yao Huang, Guoliang Kang, Liu Wenhe, Xiaojun Chang, and Alexander G. Hauptmann. 2019a. Annotation efficient cross-modal retrieval with adversarialattentive alignment. In *2019 ACM Multimedia Conference on Multimedia Conference*. ACM.

Po-Yao Huang, Frederick Liu, Sz-Rung Shiang, Jean Oh, and Chris Dyer. 2016. Attention-based multimodal neural machine translation. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, volume 2, pages 639–645.

Po-Yao Huang, Vaibhav, Xiaojun Chang, and Alexander G. Hauptmann. 2019b. Improving what cross-modal retrieval models learn through object-oriented inter- and intra-modal attention networks. In *Proceedings of the 2019 on International Conference on Multimedia Retrieval*, ICMR '19, pages 244–252, New York, NY, USA. ACM.

Ákos Kádár, Desmond Elliott, Marc-Alexandre Côté, Grzegorz Chrupała, and Afra Alishahi. 2018. Lessons learned in multilingual grounded language learning. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 402–412, Brussels, Belgium. Association for Computational Linguistics.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3128–3137.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ryan Kiros, Ruslan Salakhutdinov, and Richard S. Zemel. 2014. Unifying visual-semantic embeddings with multimodal neural language models. *NIPS Workshop*.

Benjamin Klein, Guy Lev, Gil Sadeh, and Lior Wolf. 2015. Associating neural word embeddings with deep image representations using fisher vectors. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 4437–4446.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, Michael Bernstein, and Li Fei-Fei. 2016. Visual genome: Connecting language and vision using crowdsourced dense image annotations.

Kuang-Huei Lee, Xi Chen, Gang Hua, Houdong Hu, and Xiaodong He. 2018. Stacked cross attention for image-text matching. *arXiv preprint arXiv:1803.08024*.

Jian Li, Zhaopeng Tu, Baosong Yang, Michael R Lyu, and Tong Zhang. 2018. Multi-head attention with disagreement regularization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2897–2903.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Hyeonseob Nam, Jung-Woo Ha, and Jeonghee Kim. 2017. Dual attention networks for multimodal reasoning and matching. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 2156–2164. IEEE.

Janarthanan Rajendran, Mitesh M Khapra, Sarath Chandar, and Balaraman Ravindran. 2016. Bridge correlational neural networks for multilingual multimodal representation learning. In *Proceedings of NAACL-HLT*, pages 171–181.

Cyrus Rashtchian, Peter Young, Micah Hodosh, and Julia Hockenmaier. 2010. Collecting image annotations using amazon's mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 139–147. Association for Computational Linguistics.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99.

Haoyue Shi, Jiayuan Mao, Tete Xiao, Yuning Jiang, and Jian Sun. 2018. Learning visually-grounded semantics from contrastive adversarial samples. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 3715–3727.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. 2015. Order-embeddings of images and language. *arXiv preprint arXiv:1511.06361*.

Liwei Wang, Yin Li, Jing Huang, and Svetlana Lazebnik. 2018. Learning two-branch neural networks for image-text matching tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Liwei Wang, Yin Li, and Svetlana Lazebnik. 2016. Learning deep structure-preserving image-text embeddings. In *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*, pages 5005–5013. IEEE.

John Wieting, Jonathan Mallinson, and Kevin Gimpel. 2017. Learning paraphrastic sentence embeddings from back-translated bitext. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 274–285.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78.

# Decoupled Box Proposal and Featurization with Ultrafine-Grained Semantic Labels Improve Image Captioning and Visual Question Answering

**Soravit Changpinyo** **Bo Pang** **Piyush Sharma** **Radu Soricut**
Google AI
Venice, CA 90291
{schangpi,bopang,piyushsharma,rsoricut}@google.com

## Abstract

Object detection plays an important role in current solutions to vision and language tasks like image captioning and visual question answering. However, popular models like Faster R-CNN rely on a costly process of annotating ground-truths for *both* the bounding boxes *and* their corresponding semantic labels, making it less amenable as a primitive task for transfer learning. In this paper, we examine the effect of decoupling box proposal and featurization for down-stream tasks. The key insight is that this allows us to leverage a large amount of labeled annotations that were previously unavailable for standard object detection benchmarks. Empirically, we demonstrate that this leads to effective transfer learning and improved image captioning and visual question answering models, as measured on publicly-available benchmarks.

## 1 Introduction

Object detection has been employed extensively as a primitive task for vision and language tasks such as image captioning and visual question answering (VQA); see (Anderson et al., 2018) and the work that follows it. One motivation is that the ability to recognize salient regions and objects may be too difficult to learn from weakly-supervised *top-down* signals, in the form of captions and question-answer pairs. Indeed, *bottom-up* signals provided by object detection often correspond to semantic units of language such as words or phrases, making them suitable for text generation and image-text alignment.

However, object detection itself can be broken down into multiple subtasks (Liu et al., 2018). A family of "two-stage" object detectors first proposes category-agnostic bounding box candidates and then featurizes and classifies the cropped regions into one of the available semantic labels.




Figure 1: Ultrafine-grained semantic labels (at "instance level") provide transfer learning power to down-stream tasks like visual question answering.

Even "one-stage" object detection approaches, where these boxes become category-specific, can be formulated in a *bottom-up* manner as detecting and grouping extreme and center points (Zhou et al., 2019). Can we take advantage of this observation to learn to transfer more effectively? In this work, we take a step in this direction by examining the effect of decoupling box proposal and featurization *on downstream vision and language tasks*. In particular, we consider a two-stage object detector and set a goal of pushing the "featurization" aspect of the task further than before.

Our choice to break free from "featurization by object detection models" has at least two advantages. First, there is a larger amount of labeled data that can be leveraged to train a better featurization module, even if such data do not support learning box proposals. To put it another way, the quality of features directly provided by object detectors is limited by the fact that annotating ground-truths for *both* the bounding boxes *and* their corresponding semantic labels is costly and scales poorly. By separating them, we reintroduce the freedom to annotate for object-agnostic box segmentation, without the burden of baking in an-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1468–1474, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

notation decisions related to the granularity level of the semantic labels (i.e., do we use as semantic labels "money", "euro", or "20 euro"?). As illustrated in Figure 1, the granularity level of the semantic labels plays a crucial role for downstream tasks such as VQA.

Second, this approach is better suited to downstream tasks whose domains are different from the one the object detector is trained on. In other words, it allows us to benefit from transfer learning, which is a great advantage given the relatively modest amount of available supervised data for these downstream vision and language tasks.

We empirically demonstrate the above-mentioned advantages through a focused study of the effect of improved featurization on image captioning and VQA in transfer learning settings. In particular, we (i) leverage ultra-fine-grained semantic labels (e.g., "golden gate bridge" vs. "bridge") for featurization (Juan et al., 2019); and, (ii) focus on scenarios in which object detection modules trained on Visual Genome (VG) (Krishna et al., 2017) are applied to out-of-domain images: image captioning on the Conceptual Captions dataset (Sharma et al., 2018), and VQA on the VizWiz dataset (Gurari et al., 2018). Our results indicate that there are ways to incorporate low-level pre-training tasks that benefit vision and language models via higher-quality bottom-up signals.

## 2 Related Work

Attention-based deep models are popular in image captioning and VQA. Early work used fixed partitions of images as candidate regions (Xu et al., 2015). However, variable sized regions that are better correlated with object boundaries have gained momentum (Fu et al., 2017; Pedersoli et al., 2017; Anderson et al., 2018). Indeed, Anderson et al. (2018) established new state-of-the-art performance over both image captioning and VQA tasks on the MSCOCO and VQA2 benchmarks using a Faster R-CNN detector trained on Visual Genome. As both Visual Genome and VQA2 were built on images from MSCOCO, the object detector was applied largely to in-domain images. In contrast, our work focuses on more realistic settings in which domains of different tasks may not be perfectly aligned (Chen et al., 2018).

We leverage image representations extracted from a network pre-trained over large amounts of labeled data. Prior work demonstrated the power of pre-training with image classification at scale (Sun et al., 2017; Mahajan et al., 2018; Wu et al., 2019). However, we consider downstream vision *and language* tasks (image captioning and visual question answering), in contrast to less complex vision-only tasks explored in such work: object detection and in some cases semantic segmentation and human pose estimation. Furthermore, our transfer learning technique is based on decoupled region proposal and ultra-finegrained featurization, not fine-tuning the pre-trained network.

Another set of closely related work utilized additional data for scaling up either vision tasks (Hoffman et al., 2016; Tang et al., 2017; Redmon and Farhadi, 2017) or vision and language tasks (Venugopalan et al., 2017; Lu et al., 2018; Noh et al., 2019). For instance, YOLO9000 (Redmon and Farhadi, 2017) built a "WordTree" hierarchy based on the WordNet synsets (Miller et al., 1990), mapped categories in both COCO object detection and ImageNet classification datasets into the hierarchy, and proposed a joint detection and classification training framework. Our approach to transfer learning with ultrafine-grained featurization can similarly address the long-tail nature of target vocabulary (see Figure 2) while being simpler (e.g., not require carefully merging different sets of vocabulary as in YOLO9000). The number of classes we consider is also several orders of magnitude larger.

Incorporating object detection signals in downstream tasks appropriately is non-trivial and an active subject for research (Santoro et al., 2017; Zhang et al., 2018). In this work, we ask the orthogonal question of whether it is necessary to accept the object detector's output as-is.

## 3 Features and Experimental Setup

Our starting point is a two-stage object detector, which consists of two core modules. One is responsible for category-agnostic box proposal, and the other for featurizing each cropped region for semantic label prediction. In this paper, we select Faster R-CNN (Ren et al., 2015b), a widely-used object detector in image captioning and VQA.

**Faster R-CNN Model** We reimplement the Faster R-CNN model, training it to predict both 1,600 object and 400 attribute labels in Visual Genome (Krishna et al., 2017), following the standard setting from Anderson et al. (2018). ResNet-

101 (He et al., 2016) pre-trained on ImageNet (Russakovsky et al., 2015) is used as the core featurization network[1]. We achieve a mAP@50 of 10.96 for object detection and 1.5 for attribute detection. Given an image, Faster R-CNN proposes K bounding box regions, each of which comes with a D-dimensional feature vector as well as object/attribute class predictions (along with their scores). K is set to 100 and D to 2048 in our experiments. Using output features on the task of VQA and our model described in Section 5, we obtain an accuracy of 66.9% on the validation set of the VQA2 dataset (Goyal et al., 2017). For comparison, this number already surpasses all validation accuracy numbers in Table 2 for a strong model by Peng et al. (2019), suggesting that our Faster R-CNN features are of high-quality.

**Decoupled Box Proposal and Featurization with Ultra-finegrained Semantic Labels** In standard use of object detectors following Anderson et al. (2018), downstream tasks receive "knowledge" merely about a few thousand classes and four hundred attributes. Here, we exploit the fact that box proposal and featurization can be decoupled, and work on improving the object representation (featurization).

More concretely, we conduct a study toward understanding the utility of improved featurization on downstream tasks. To this end, we exploit a graph-based, semi-supervised representation learning approach called Graph-Regularized Image Semantic Embedding (Graph-RISE) (Juan et al., 2019). Specifically, Graph-RISE is based on ResNet-101 where the 10x10x2K feature map is first average pooled to 4x4x2K, and then flattened and projected to a 64-dimensional embedding before the softmax layer. Learned from $\mathcal{O}$(260M) web images and $\mathcal{O}$(40M) (noisy) semantic labels, these compact 64-dimensional feature vectors are trained to capture a whole spectrum of semantic similarity, ranging from coarse-grained / category-level (e.g., "bridge"), fine-grained level (e.g., "steel red bridge"), to ultrafine-grained / instance-level (e.g., "golden gate bridge").

**Our Objective** The main goal is to compare two approaches in using bottom-up signals: 1) FRCNN: use the default visual features from the Faster R-CNN detector; 2) Ultra: use bounding boxes from the Faster R-CNN detector, then featurize them using the much more compact representation from Graph-RISE that potentially reflects differentiation of ultrafine-grained semantic labels. Next, we evaluate this setup on downstream tasks for image captioning and visual question answering.

[1] See further details in the supplementary material.

## 4 Image Captioning

**Dataset** We use the Conceptual Captions (CC) dataset (Sharma et al., 2018), consisting of 3.3 million training and 15,000 validation images/caption pairs. Another 12,000 image/caption pairs comprise the hidden test set. Official scores on the test set are obtained by submitting models to the CC Challenge server[2]. Unlike other image captioning datasets, images from CC are pulled from across the web and thus exhibit a wide variety of both images and image-caption styles. Most notably, the domain of images can be very different from Visual Genome, unlike in popular benchmarks such as MSCOCO (Lin et al., 2014).

**Model** We adopt the encoder-decoder model from (Sharma et al., 2018), whose basic building block is a Transformer Network (Vaswani et al., 2017). To convert multi-modal inputs to a sequence of encoder feature vectors, we use up to three types of image features:

**G** : Global features by Graph-RISE, a dense 64D vector extracted from the whole image;

**B** : Box-region features by Faster R-CNN (FRCNN, sparse 2048D), or Graph-RISE (Ultra, dense 64D), extracted from each cropped image region resized to 224x224 (cf. Sec. 3);

**L** : Label embeddings, obtained by embedding predicted object semantic labels from Google Cloud Vision APIs[3] into a 512D feature vector. These semantic labels are then mapped to embeddings using an embedding layer pre-trained to predict label co-occurrences in web documents using a word2vec model (Mikolov et al., 2013).

For both **B** and **L**, we select the inputs with highest scores and order the sequence inputs based on such scores from high to low. Additionally for **B**, we remove box regions whose scores are lower than 0.001. We use beam search with width 5 for the decoder in all of our experiments[4].

[2] ai.google.com/research/ConceptualCaptions

[3] cloud.google.com/vision

[4] See further details in the supplementary material.

| | dev | test | | |
|---|---|---|---|---|
| | CIDEr | CIDEr | ROUGE-L | SPICE |
| Transf-Baseline | - | 0.772 | 0.244 | 0.172 |
| TTI-BIC (single) | - | 0.980 | **0.266** | 0.186 |
| **G** (Base) | 0.868 | - | - | - |
| **B**-FRCNN | 0.667 | - | - | - |
| **B**-Ultra | 0.873 | - | - | - |
| **L** | 0.606 | - | - | - |
| **G** + **B**-FRCNN | 0.871 | - | - | - |
| **G** + **B**-Ultra | 0.912 | - | - | - |
| **G** + **L** | 0.888 | - | - | - |
| **G** + **B**-FRCNN + **L** | 0.892 | 0.944 | 0.261 | 0.190 |
| **G** + **B**-Ultra + **L** | **0.937** | **0.984** | **0.265** | **0.195** |

Table 1: Automatic metric scores for the image captioning task on Conceptual Captions. Ablation results are reported for our model using different sets of visual features. The top two baselines are from the Conceptual Captions Leaderboard as of August 30, 2019.

**Metrics** We adopt the standard automatic metrics for image captioning: CIDEr (Vedantam et al., 2015), ROUGE-L (Lin and Och, 2004), and SPICE (Anderson et al., 2016), as implemented in the COCO-caption evaluation toolkit[5].

**Results** We report results on both the dev and test sets for Conceptual Captions in Table 1. "Base" uses the **G** feature only. We first compare the Base **G** against each of the feature types (**B**-FRCNN, **B**-Ultra, and **L**). We then perform ablations under the **+B** condition (FRCNN/Ultra) to the Base **G** or stronger **G** + **L** models.

According to dev CIDEr scores, global or box Graph-RISE features **G** and **B**-Ultra are (individually) clearly stronger than box features by Faster R-CNN **B**-FRCNN or label embeddings **L** features. Nevertheless, these features are considerably complementary. Specifically, box features **B**-Ultra complements the Base **G**, pushing the score from 0.868 to 0.912. It is also worth noting that, albeit their low individual scores, **B**-FRCNN or **L** improves upon each model they are added to.

Our models with Ultra features clearly outperform the ones with FRCNN. This is demonstrated in three conditions: when they are on their own, when they are added to the simple **G** model, and when they are added to the stronger **G** + **L** model. Manual inspection of the models' predictions further supports this; a qualitative comparison of **B**-Ultra vs. **B**-FRCNN in Figure 2 suggests that ultra-finegrained featurization leads to an improved correspondence between visual inputs and caption tokens of *unfamiliar* objects (such as "monks" and "staircase").

[5]https://github.com/tylin/coco-caption.

To get test scores, we submit our best model using FRCNN and our best model using Ultra (based on dev CIDEr) to the CC Challenge server. Test scores for other models were not obtained due to the limited number of submissions per time period. As of August 30, 2019, the **G** + **B**-Ultra + **L** model outperforms all other single baselines[6], for both CIDEr and SPICE (and tie on ROUGE-L).

## 5 Visual Question Answering

**Dataset** We use the recently-proposed VizWiz dataset (Gurari et al., 2018), in which both images and questions originate from visually-impaired or blind people. It consists of 20,000/3,173 ⟨image, question, answers⟩ triplets in the train/val splits, and additional 8,000 triplets for the test split. Each question is independently annotated with 10 answers. We choose the VizWiz benchmark specifically because it is a more suitable benchmark for measuring transfer learning effects. Other VQA datasets, including VQA1.0 (Antol et al., 2015), VQA2.0 (Goyal et al., 2017), Visual7W (Zhu et al., 2016), COCOQA (Ren et al., 2015a), and GQA (Hudson and Manning, 2019) are completely or partly based on MSCOCO or Visual Genome. As such, they may not provide unbiased grounds for measuring the impact of object-detection features based on Visual Genome versus alternative featurization techniques.

**Model** We follow the setting described in Pythia v0.1 (Jiang et al., 2018), the winning entry to the VQA challenge 2018. In particular, the architecture is a *simplified* "up-down" model from (Anderson et al., 2018)[7]. The featurization of the bounding boxes follows the description from Section 4. For the base condition, we use the box features based on Faster R-CNN (**B**-FRCNN), following the majority of previous work. For the test condition, we replace them with the Ultra-based features (**B**-Ultra).

**Metrics** As commonly done in previous work (Antol et al., 2015), we use as our accuracy metric the average score over 9 subsets of the ground-truth 10 answers, where each score is computed by the formula: min(# humans that provided that answer / 3, 1). Accuracy on the test-dev and test-standard splits is obtained by submitting the mod-

[6]ai.google.com/research/ConceptualCaptions/leaderboard

[7]See further details in the supplementary material.



**Ground-truth** "monks clean a garden at a temple ."
**Box FRCNN (VG)** "a woman walks through the streets ."
**Box Ultra** "monks walking in front of a temple"



"black sesame seeds on a white background"
"a pile of dried flowers"
"black chia seeds on a white background"

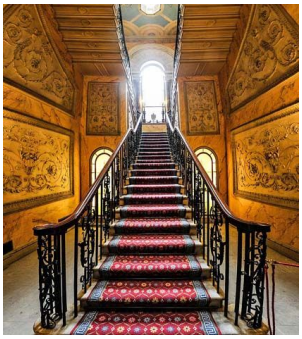

**Ground-truth** "a photo of a staircase inside a historic house"
**Box FRCNN (VG)** "the interior of the church"
**Box Ultra** "the staircase of the house"



"car & tree ornament this heart of mine"
"digital art selected for the #"
"christmas tree in a toy car"

Figure 2: Qualitative results from our image captioning models using **B**-FRCNN vs. **B**-Ultra (see text for details), along with ground-truth captions. Ultra is more capable than FRCNN of dealing with images with unfamiliar objects, those that do not perfectly fall into the domain where the Faster R-CNN object detector is trained on.

| | all | y/n | num | unans | other |
|---|---|---|---|---|---|
| VizWiz | 46.9 | 59.6 | 21.0 | 80.5 | 27.3 |
| BAN | 51.6 | **68.1** | 17.9 | **85.3** | 31.5 |
| Ours (FRCNN) | 51.9 | 66.7 | 24.3 | 85.0 | 32.1 |
| Ours (Ultra) | **53.7** | **68.1** | **28.8** | 84.0 | **35.4** |

Table 2: Accuracy (%) on the test-standard split for the VQA task on the VizWiz dataset. Additionally, we provide accuracy per answer type: yes/no (y/n), number (num), unanswerable (unans), and the rest (other). The baselines include VizWiz (Gurari et al., 2018) and BAN (Kim et al., 2018).

els to the VizWiz Challenge server[8].

**Results** We report results on the VizWiz benchmark in Table 2. Our model with FRCNN provides a strong baseline, slightly outperforming the previous-best model, BAN (Kim et al., 2018), a different architecture that also uses the FRCNN-based features for object bounding boxes. The model using Ultra features further improves upon this; at 53.7%, it outperforms the one using FRCNN by a significant margin (1.8% accuracy on "all" question types). Moreover, this 1.8% improvement is a weighted average across answer types; the per-answer-type numbers indicate that our approach achieves even better improvements on two of the more difficult answer types, "number" (+4.5%) and "rest" (+3.3%). These improvements are illustrated by the examples provided in Figure 1.

This illustrates the effectiveness of decoupling bounding box proposal and featurization, and quantifies the impact of using transfer learning via large amounts of training data and ultrafine-grained semantic labels used for object representations.

## 6 Conclusion

In this work, we propose to (re)decouple box proposal and featurization. We show that this allows us to leverage additional signals and annotations, leading to more effective transfer learning for downstream vision and language tasks: image captioning and visual question answering. This result suggests that large-scale datasets with fine-grained image-level semantic labels, even when they do not dissect complex visual scenes, can benefit current state-of-the-art models – especially when applied to benchmarks where images are from diverse domains.

[8] evalai.cloudcv.org/web/challenges/challenge-page/102/overview

## References


Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2016. SPICE: semantic propositional image caption evaluation. In *ECCV*.

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of CVPR*.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. VQA: Visual question answering. In *Proceedings of ICCV*.

Yuhua Chen, Wen Li, Christos Sakaridis, Dengxin Dai, and Luc Van Gool. 2018. Domain adaptive Faster R-CNN for object detection in the wild. In *Proceedings of CVPR*.

Kun Fu, Junqi Jin, Runpeng Cui, Fei Sha, and Changshui Zhang. 2017. Aligning where to see and what to tell: Image captioning with region-based attention and scene-specific contexts. *TPAMI*, 39(12):2321–2334.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the V in VQA matter: Elevating the role of image understanding in visual question answering. In *Proceedings of CVPR*.

Danna Gurari, Qing Li, Abigale J. Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P. Bigham. 2018. VizWiz Grand Challenge: Answering visual questions from blind people. In *Proceedings of CVPR*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of CVPR*.

Judy Hoffman, Deepak Pathak, Eric Tzeng, Jonathan Long, Sergio Guadarrama, Trevor Darrell, and Kate Saenko. 2016. Large scale visual recognition through adaptation using joint representation and multiple instance learning. *JMLR*, 17(1):4954–4984.

Drew A. Hudson and Christopher D. Manning. 2019. GQA: a new dataset for compositional question answering over real-world images. In *Proceedings of CVPR*.

Yu Jiang, Vivek Natarajan, Xinlei Chen, Marcus Rohrbach, Dhruv Batra, and Devi Parikh. 2018. Pythia v0.1: the winning entry to the VQA Challenge 2018. *arXiv preprint arXiv:1807.09956*.

Da-Cheng Juan, Chun-Ta Lu, Zhen Li, Futang Peng, Aleksei Timofeev, Yi-Ting Chen, Yaxi Gao, Tom Duerig, Andrew Tomkins, and Sujith Ravi. 2019. Graph-RISE: Graph-regularized image semantic embedding. *arXiv preprint arXiv:1902.10814*.

Jin-Hwa Kim, Yongseok Choi, Sungeun Hong, Jaehyun Jun, and Byoung-Tak Zhang. 2018. Bilinear attention networks for VizWiz challenge. In *Proceedings of the ECCV Workshop on VizWiz Grand challenge*.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael Bernstein, and Li Fei-Fei. 2017. Visual Genome: Connecting language and vision using crowdsourced dense image annotations. *IJCV*, 123(1):32–73.

Chin-Yew Lin and Franz Josef Och. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of ACL*.

Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: common objects in context. In *Proceedings of ECCV*.

Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. 2018. Deep learning for generic object detection: A survey. *arXiv preprint arXiv:1809.02165*.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2018. Neural baby talk. In *Proceedings of CVPR*.

Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. 2018. Exploring the limits of weakly supervised pretraining. In *Proceedings of ECCV*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NeurIPS*.

George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller. 1990. Introduction to WordNet: An on-line lexical database. *International Journal of Lexicography*, 3(4):235–244.

Hyeonwoo Noh, Taehoon Kim, Jonghwan Mun, and Bohyung Han. 2019. Transfer learning via unsupervised task discovery for visual question answering. In *Proceedings of CVPR*.

Marco Pedersoli, Thomas Lucas, Cordelia Schmid, and Jakob Verbeek. 2017. Areas of attention for image captioning. In *Proceedings of ICCV*.

Gao Peng, Hongsheng Li, Haoxuan You, Zhengkai Jiang, Pan Lu, Steven Hoi, and Xiaogang Wang. 2019. Dynamic fusion with intra-and inter-modality attention flow for visual question answering. In *Proceedings of CVPR*.

Joseph Redmon and Ali Farhadi. 2017. YOLO9000: Better, faster, stronger. In *Proceedings of CVPR*.

Mengye Ren, Ryan Kiros, and Richard Zemel. 2015a. Exploring models and data for image question answering. In *Proceedings of NeurIPS*.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015b. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Proceedings of NeurIPS*.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet large scale visual recognition challenge. *IJCV*, 115(3):211–252.

Adam Santoro, David Raposo, David G. Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. 2017. A simple neural network module for relational reasoning. In *Proceedings of NeurIPS*.

Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. 2018. Conceptual Captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of ACL*.

Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. 2017. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of ICCV*.

Yuxing Tang, Josiah Wang, Xiaofang Wang, Boyang Gao, Emmanuel Dellandréa, Robert Gaizauskas, and Liming Chen. 2017. Visual and semantic knowledge transfer for large scale semi-supervised object detection. *TPAMI*, 40(12):3045–3058.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of NeurIPS*.

Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. 2015. CIDEr: Consensus-based image description evaluation. In *Proceedings of CVPR*.

Subhashini Venugopalan, Lisa Anne Hendricks, Marcus Rohrbach, Raymond Mooney, Trevor Darrell, and Kate Saenko. 2017. Captioning images with diverse objects. In *Proceedings of CVPR*.

Baoyuan Wu, Weidong Chen, Yanbo Fan, Yong Zhang, Jinlong Hou, Junzhou Huang, Wei Liu, and Tong Zhang. 2019. Tencent ML-Images: A large-scale multi-label image database for visual representation learning. *arXiv preprint arXiv:1901.01703*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of ICML*.

Yan Zhang, Jonathon Hare, and Adam Prügel-Bennett. 2018. Learning to count objects in natural images for visual question answering. In *Proceedings of ICLR*.

Xingyi Zhou, Jiacheng Zhuo, and Philipp Krähenbühl. 2019. Bottom-up object detection by grouping extreme and center points. In *Proceedings of CVPR*.

Yuke Zhu, Oliver Groth, Michael Bernstein, and Li Fei-Fei. 2016. Visual7W: Grounded question answering in images. In *Proceedings of CVPR*.

# REO-Relevance, Extraness, Omission: A Fine-grained Evaluation for Image Captioning

**Ming Jiang**[1], **Junjie Hu**[2], **Qiuyuan Huang**[3], **Lei Zhang**[3], **Jana Diesner**[1], **Jianfeng Gao**[3]
[1]University of Illinois at Urbana-Champaign, [2]Carnegie Mellon University
[3]Microsoft Research, Redmond
{mjiang17,jdiesner}@illinois.edu, junjieh@cs.cmu.edu
{qihua,leizhang,jfgao}@microsoft.com

## Abstract

Popular metrics used for evaluating image captioning systems, such as BLEU and CIDEr, provide a single score to gauge the system's overall effectiveness. This score is often not informative enough to indicate what specific errors are made by a given system. In this study, we present a fine-grained evaluation method REO for automatically measuring the performance of image captioning systems. REO assesses the quality of captions from three perspectives: 1) **R**elevance to the ground truth, 2) **E**xtraness of the content that is irrelevant to the ground truth, and 3) **O**mission of the elements in the images and human references. Experiments on three benchmark datasets demonstrate that our method achieves a higher consistency with human judgments and provides more intuitive evaluation results than alternative metrics.[1]

## 1 Introduction

Image captioning is an interdisciplinary task that aims to automatically generate a text description for a given image. The task is fundamental to a wide range of applications, including image retrieval (Rui et al., 1999) and vision language navigation (Wang et al., 2019). Though remarkable progress has been made (Gan et al., 2017; Karpathy and Li, 2015), the automatic evaluation of image captioning systems remains a challenge, particularly with respect to quantifying the generation errors made by these systems (Bernardi et al., 2016).

Existing metrics for caption evaluation can be grouped into two categories: 1) rule-based metrics (Papineni et al., 2002; Vedantam et al., 2015) that are based on exact string matching, and 2) learning-based metrics (Cui et al., 2018; Sharif et al., 2018) that predict the probability of a testing caption as a human-generated caption by using a learning model. In general, prior work has shown that description adequacy with respect to the ground truth data is a main concern for evaluating text generation systems (Gatt and Krahmer, 2018). Though this aspect has been emphasized by prior work for assessing image captions (Papineni et al., 2002; Banerjee and Lavie, 2005; Gao et al., 2019), one common limitation of existing metrics is the lack of interpretability to the description errors because existing metrics only provide a composite score for the caption quality. Without fine-grained analysis, the developers may not be able to understand the specific description errors made by their developed captioning systems.

To fill this gap, we propose an evaluation method called **REO** that considers three specific pieces of information for measuring each caption with respect to: 1) **R**elevance: relevant information of a candidate caption with respect to the ground truth, 2) **E**xtraness: extra information of a



***References***
- A man instructing a group of kids on a soccer field.
- A soccer coach is instructing the children on the field.
- Pair of adult males with group of small children with soccer balls.

| **Candidate A:** Two men are teaching kids football *in a shopping mall with many stores*. | | | **Candidate B:** Two men are teaching football. *(missing "kids")* | | |
|---|---|---|---|---|---|
| BLEU4: **0.00** | CIDEr: 0.05 | SPICE: **0.05** | BLEU4: **0.00** | CIDEr: 0.00 | SPICE: **0.06** |
| REO_R: 0.28 | REO_E: 9.39 | REO_O: 10.64 | REO_R: 0.26 | REO_E: 10.40 | REO_O: 9.32 |

Figure 1: An example of caption evaluation. Given two caption candidates, even though Caption A covers more image information than Caption B (e.g., missing "kids"), Caption A contains extra irrelevant description (highlighted in red). Prior metrics (e.g., BLEU4) only provide an overall quality score, which is difficult to infer specific description mistakes in a caption. In contrast, REO provides three indicators (i.e., relevance, extraness, and omission) that can properly achieve a fine-grained assessment for each caption.

[1]Code is released at https://github.com/SeleenaJM/CapEval.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1475–1480, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

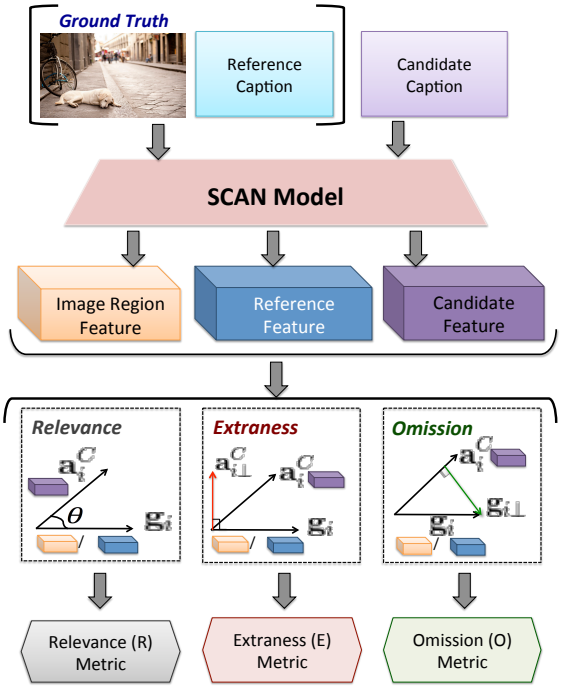


Figure 2: Overview of REO evaluation for image captioning: (1) Feature Extraction; (2) Metric Measurement.

candidate caption beyond ground truth data, and 3) **O**mission: missing information that a candidate fails to describe from an image and human-generated reference captions. Figure 1 shows a comparison between existing metrics and our proposed metrics that measure caption quality at a fine-grained level. If we view caption generation as a process of decoding the information embedded in an image, we can evaluate an image captioning system by measuring the effectiveness of the decoding process in terms of the relevance of the decoded information regarding the image content, and the amount of missing or extra information. Using both the images and reference captions as ground truth information for evaluation, our approach is built based on a shared image-text embedding space defined by a grounding model that has been pre-trained on a large benchmark dataset. Given a pair of vectors representing a candidate caption and its ground truth (i.e., the target image and associated reference captions), respectively, we compute the relevance of the candidate caption and ground truth based on vector similarities. By applying vector orthogonal projection, we identify the extra and missing information carried by the candidate caption. Each aspect that we consider here (i.e., relevance, extraness, omission) is measured by an independent score.

We test our method on three datasets. The experimental results show that our proposed metrics are more consistent with human evaluations than alternative metrics. Interestingly, our study finds that human annotators pay more attention to extra or missing information in a caption (i.e., false positive and false negatives) than the caption's relevance for the given image (true positives). We also find that considering both image and references as ground truth information is more helpful for caption evaluation than considering image or references individually.

## 2 Methods

Figure 2 provides an overview for the calculation of REO, which happens in two stages. The first stage is *feature extraction*, where we aim to obtain feature vectors to encode the candidate caption $C$ and corresponding ground truth $\mathbf{G}$ for further comparisons. The second stage is to measure *three metric scores*. Specifically, we measure relevance using standard cosine similarity. To measure irrelevance (i.e., extraness and omissions), we compare the information carried by $C$ and $\mathbf{G}$, respectively. We will give a detailed description of our method in the following two subsections.

### 2.1 Feature Extraction

Following Lee et al., we leverage a pre-trained Stacked Cross Attention Neural Network (SCAN) to build a multi-modal semantic space. Specifically, we obtain word features $\mathbf{H}^\tau = [\mathbf{h}_1^\tau; \cdots; \mathbf{h}_M^\tau] \in \mathbb{R}^{M \times D}$ by averaging the forward and backward hidden states per word from a bidirectional GRU (Bahdanau et al., 2014), where $\tau = \{C, R\}$ denotes either a candidate $C$ or a reference $R$ sentence of $M$ words. Based on Anderson et al., we achieve image features $\mathbf{U} \in \mathbb{R}^{N \times D'}$ by detecting $N$ salient regions per image ($N = 36$ in this paper). A linear layer is applied to transform image features to $D$-dimensional features $\mathbf{V} = [\mathbf{v}_1; \cdots; \mathbf{v}_N] \in \mathbb{R}^{N \times D}$.

Based on the SCAN model, we further extract the context information from the caption words for each detected region. To this end, we compute a context feature $\mathbf{a}_i^\tau$ for the $i^{th}$ region by a weighted sum of caption word features in Eq. (1). Notice that $\mathbf{A}^\tau = [\mathbf{a}_1^\tau; \cdots; \mathbf{a}_N^\tau] \in \mathbb{R}^{N \times D}$ extracts the context information of the caption $\tau$ with respect to all regions in the image.

$$\mathbf{a}_i^\tau = \sum_{j=1}^{m} \alpha_{ij} \mathbf{h}_j^\tau \tag{1}$$

$$\alpha_{ij} = \frac{\exp(\lambda \text{sim}(\mathbf{v}_i, \mathbf{h}_j^\tau))}{\sum_{k=1}^{m} \exp(\lambda \text{sim}(\mathbf{v}_i, \mathbf{h}_k^\tau))} \tag{2}$$

where $\lambda$ is a smoothing factor, and $\text{sim}(\mathbf{v}, \mathbf{h}^\tau)$ is a normalized similarity function defined as

$$\text{sim}(\mathbf{v}_i, \mathbf{h}_j^\tau) = \frac{\max(0, \text{cosine}(\mathbf{v}_i, \mathbf{h}_j^\tau))}{\sqrt{\sum_{k=1}^{n} \max(0, \text{cosine}(\mathbf{v}_k, \mathbf{h}_j^\tau))^2}} \tag{3}$$

### 2.2 Metric Scores

In order to explore the impact of image data on evaluation, we focus on comparing context features $\mathbf{A}^C$ of the candidate caption $C$ to ground-truth references $\mathbf{G} \equiv [\mathbf{g}_1; \cdots; \mathbf{g}_N] = \{\mathbf{V}, \mathbf{A}^R\}$, where $\mathbf{G}$ denotes either image features $\mathbf{V}$ or the context features of $R$ (i.e., $\mathbf{A}^R$).

**Relevance** : The relevance between a candidate caption and a ground-truth reference based on the $i$-th region is computed by the cosine similarity of $\mathbf{a}_i^C$ and $\mathbf{g}_i$. We average similarity over all regions to get the relevance score of a candidate caption with respect to an image.

$$\mathcal{R} = \frac{1}{N} \sum_{i=1}^{N} \text{sim}(\mathbf{a}_i^C, \mathbf{g}_i) \tag{4}$$

**Extraness** : The extraness of $C$ is captured by performing an orthogonal projection of $\mathbf{a}_i^C$ to $\mathbf{g}_i$, which returns the vertical context vector $\mathbf{a}_{i\perp}^C$ to represent the irrelevant content of $C$ to the ground truth at the $i^{th}$ region.

$$\mathbf{a}_{i\perp}^C = \mathbf{a}_i^C - \frac{\mathbf{a}_i^C \cdot \mathbf{g}_i}{\| \mathbf{g}_i \|^2} \mathbf{g}_i. \tag{5}$$

To avoid potential disturbance due to correlated feature vectors, we measure the Mahalanobis distance between the vertical context vector $\mathbf{a}_{i\perp}^C$ and its original context vector $\mathbf{a}_i^C$ (see Eq. (7)). Notice that a small distance value indicates that the irrelevant context vector $\mathbf{a}_{i\perp}^C$ is closed to the original context vector $\mathbf{a}_i^C$. In other words, the original context contains a large amount of extra information. Therefore, the higher this metric is, the less extra information the caption contains.

$$\mathcal{E} = \frac{1}{N} \sum_{i=1}^{N} d(\mathbf{a}_i^C, \mathbf{a}_{i\perp}^C) \tag{6}$$

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{(\mathbf{p} - \mathbf{q}) S^{-1} (\mathbf{p} - \mathbf{q})} \tag{7}$$

**Omission** : The measurement of omission is similar to that of extraness, where we capture the missing information of $C$ by the vertical context features $\mathbf{g}_{i\perp}$ based on the orthogonal projection of $\mathbf{g}_i$ to $\mathbf{a}_i^C$. The omission score is denoted as $\mathcal{O}$. Similarly, the higher the omission score is, the less missing information the caption contains.

$$\mathbf{g}_{i\perp} = \mathbf{g}_i - \frac{\mathbf{g}_i \cdot \mathbf{a}_i^C}{\| \mathbf{a}_i^C \|^2} \mathbf{a}_i^C \tag{8}$$

$$\mathcal{O} = \frac{1}{N} \sum_{i=1}^{N} d(\mathbf{g}_i, \mathbf{g}_{i\perp}) \tag{9}$$

Considering that each image may have multiple reference captions, we further average the score of the aforementioned three aspects over all reference captions while considering $\mathbf{A}^R$ as ground truth.

## 3 Experiments

### 3.1 Experimental Setup

We perform experiments on three human-evaluated caption sets. ***Composite Dataset*** (Aditya et al., 2015) contains the candidate captions of images from MS-COCO, Flickr8k, and Flickr30k. Captions were generated by humans and two caption models (11,985 instances in total). Human judgments for these candidate captions was provided on a 5-point scale rating that represents description correctness. ***PublicSys Dataset*** has 2,500 captions collected by Rohrbach et al., which were generated by five state-of-the-art captioning systems on 500 MS-COCO images, respectively. Human grading was done on a 5-point scale based on annotators' preferences to descriptions. ***Pascal-50S Dataset*** (Vedantam et al., 2015) includes 4000 caption pairs that describe images from the UIUC PASCAL Sentence dataset. Each annotator was asked to select one sentence per pair that is closer to the expression of the given reference sentence. Candidate pairs were grouped into four categories: 1) human-human correct (HC, i.e., a pair of captions are written by humans for the same image), 2) human-human incorrect (HI, i.e., Two human-written captions of which one describes another image instead of the target image), 3) human-machine (HM, i.e., two captions are generated by a human and a machine, respectively.), and 4) machine-machine (MM, i.e., both machine-generated caption).

Following standard practice (Anderson et al., 2016; Elliott and Keller, 2014), we compared with

| Ground Truth | Metric | Composite ($\tau$) | PublicSys ($\tau$) | Pascal-50S ($accuracy\%$) | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | HC | HI | HM | MM | ALL |
| Reference | BLEU-1 | 0.280 | 0.267 | 50.50 | 94.50 | 92.30 | 56.00 | 73.33 |
| | BLEU-4 | 0.205 | 0.223 | 50.60 | 91.90 | 85.60 | 60.90 | 72.25 |
| | ROUGE_L | 0.307 | 0.232 | 53.30 | 94.60 | 93.50 | 58,20 | 74.90 |
| | METEOR | 0.379 | 0.254 | 58.00 | 97.60 | 94.90 | 63.40 | 78.48 |
| | CIDEr | 0.378 | 0.278 | 54.80 | 97.90 | 91.50 | 63.80 | 77.00 |
| | SPICE | 0.419 | 0.258 | 56.60 | 94.70 | 85.00 | 49.00 | 71.33 |
| Image (ours) | Relevance | 0.423 | 0.148 | 58.40 | 99.40 | 93.10 | 73.40 | 81.08 |
| | Extraness | 0.430 | 0.149 | 56.80 | **99.70** | 92.80 | 74.80 | 81.03 |
| | Omission | 0.445 | 0.165 | **61.00** | 99.40 | 93.80 | 69.10 | 80.83 |
| Image + Reference (ours) | Relevance | 0.502 | 0.313 | 56.40 | **99.70** | 93.50 | 77.10 | 81.68 |
| | Extraness | 0.507 | **0.320** | 54.30 | 99.60 | 92.60 | **77.20** | 80.93 |
| | Omission | **0.533** | 0.291 | 60.00 | 99.60 | **95.40** | 72.50 | **81.88** |

Table 1: Caption-level correlation between metrics and human grading scores in Composite and PublicSys dataset by using Kendall tau ($\tau$). All p-values $< 0.01$. For PASCAL-50S, we display the accuracy of metrics at matching human judgments with 5 reference captions per image. The highest value per column is in bold font. Column titles are explained in Section 3.1. Ground truth refers to two points of information: human-written references and images.

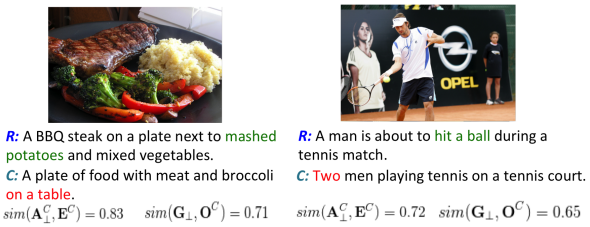


Figure 3: Examples of validating error identification. Text in red is extra information, while text in green is missing information. $sim(x, y)$ is the average similarity between machine-identified and true error vectors over image regions.

rule-based metrics (see Table 1). All existing metrics were implemented with the MS-COCO evaluation tool [2]. The performance of metrics was assessed via Kendall's tau ($\tau$) rank correlation for the scoring-based datasets (i.e., Composite & PublicSys) and accuracy of the pairwise comparison for the Pascal-50S dataset.

### 3.2 Experimental Results

**Can extra & missing information be captured?** In order to measure the effectiveness of error identification (i.e., extraness and omission), we randomly sampled a subset of data, and manually identify the actual extraness (i.e., $\mathbf{E}^C$) and true omission (i.e., $\mathbf{O}^C$) of each candidate caption. We conduct validation based on the average cosine similarity between the machine-identified error (i.e., $\mathbf{a}^C_{i\perp}$ &$\mathbf{g}_{i\perp}$) and true error description.

Figure 3 provides two illustrative examples of the validation process. Phrases highlighted in red (e.g.,"on a table") are extra information (more text in the candidate caption than in the ground truth). Meanwhile, phrases in green (e.g., "mashed potatoes") are missing from the candidate description, but occur in the image and the reference caption. We observe that machine-identified errors are highly similar to the true error information in both cases ($\geq 0.65$). This result suggests that our method can capture extraness and omission from an image caption.

**Do error-aware evaluation metrics help?** The results of metric performance in Table 1 show that overall, using the three metrics proposed in REO, especially extraness and omission, led to a noticeable improvement in Kendall tau's correlation compared to the best reported results based on prior metrics. Our results suggest that human evaluation tends to be more sensitive to the irrelevance than the relevance of a candidate caption regarding ground truth. We also find that jointly considering both images and human-written references contributes more to caption evaluation than each of the two data sources alone - except for the case of HC pair comparison. This exception can be explained by the phenomenon that human-written descriptions are flexible in terms of word

[2] https://github.com/tylin/coco-caption

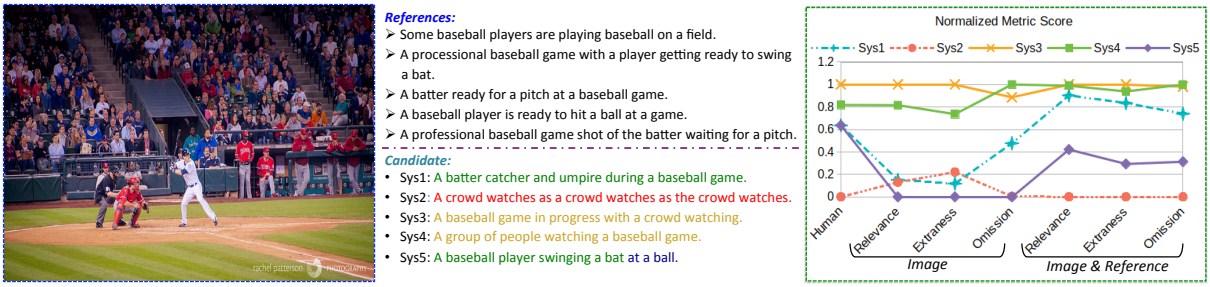


Figure 4: Case study of REO metric scores. Candidates are highlighted in: 1) green: a detailed but incompleted caption, 2) red: repetition, 3) yellow: high-level description, and 4) blue: extra information not shown in the image.

choice and sentence structure, and such diversity may lead to the challenge of comparing a reference to a candidate in cases where both captions were provided by humans. By further looking into each considered aspect in REO, we find that extranesss metric is more appropriate to evaluate machine-generated captions (e.g., the PublicSys dataset and the MM pairs of Pascal-50S dataset), while the omission metric can be a better choice to access caption quality when the testing data consists of human-written descriptions.

**What can we learn from the metric outputs?** To analyze our metric scores more in depth, we compare the outputs of five captioning systems on a set of images in the PublicSys dataset. Figure 4 shows an illustrative example. To make the scale of human grading and REO metrics comparable, we normalize scores per metric by using max-min normalization.

We find that metrics calculated in cases where the ground truth contained both the target image and human references are more likely to identify expression errors. For example, though the phrase "a crowd watches" in the caption of system 2 is relevant to the image, this phrase is repeated by three times in a sentence. As a result, the scores for relevance and extraness are decreasing when the ground truth involves references. Also, metrics focusing only on image content return higher values when the testing captions provide a high-level description of the whole image (e.g., captions of system 3 and 4) compared to the detailed captions for a specific image part (e.g., captions of system 1 and 5 focus on the baseball player). By comparing the herein considered three aspects of each caption, we observe that a caption that mainly focuses on describing a part of image in detail boosts relevance, but the sentence achieves a reduced metric score in terms of omission.

## 4 Conclusion

This paper presents a fine-grained, error-aware evaluation method REO to measure the quality of machine-generated image captions according to three aspects of descriptions: relevance regarding ground truth, extra description beyond image content, and omitted ground truth information. Comparing these metrics to alternative metrics, we find that our proposed solution produces evaluations that are more consistent with the assessment of human judges. Moreover, we find that human judgment tends to penalize extra and missing information (false positives and false negatives) more than it appreciates relevant content. Finally, and to no surprise, we conclude that using a combination of image content and human-written references as ground truth data allows for a more comprehensive evaluation than using either type of information separately. Our method can be extended to evaluate other text generation tasks.

## Acknowledgments

We appreciate anonymous reviewers for their constructive comments and insightful suggestions. This work was partly performed when Ming Jiang was interning at Microsoft Research. The authors would like to thank Pengchuan Zhang for his help with pre-training the grounding model.

## References

Somak Aditya, Yezhou Yang, Chitta Baral, Cornelia Fermuller, and Yiannis Aloimonos. 2015. From images to sentences through scene description graphs using commonsense reasoning and knowledge. *arXiv preprint arXiv:1511.03292*.

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2016. Spice: Semantic propositional image caption evaluation. In *European Conference on Computer Vision*, pages 382–398. Springer.

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6077–6086.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72.

Raffaella Bernardi, Ruket Cakici, Desmond Elliott, Aykut Erdem, Erkut Erdem, Nazli Ikizler-Cinbis, Frank Keller, Adrian Muscat, and Barbara Plank. 2016. Automatic description generation from images: A survey of models, datasets, and evaluation measures. *Journal of Artificial Intelligence Research*, 55:409–442.

Yin Cui, Guandao Yang, Andreas Veit, Xun Huang, and Serge Belongie. 2018. Learning to evaluate image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5804–5812.

Desmond Elliott and Frank Keller. 2014. Comparing automatic evaluation measures for image description. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 452–457.

Zhe Gan, Chuang Gan, Xiaodong He, Yunchen Pu, Kenneth Tran, Jianfeng Gao, Lawrence Carin, and Li Deng. 2017. Semantic compositional networks for visual captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2.

Jianfeng Gao, Michel Galley, Lihong Li, et al. 2019. Neural approaches to conversational ai. *Foundations and Trends® in Information Retrieval*, 13(2-3):127–298.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61:65–170.

Andrej Karpathy and Fei-Fei Li. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3128–3137.

Kuang-Huei Lee, Xi Chen, Gang Hua, Houdong Hu, and Xiaodong He. 2018. Stacked cross attention for image-text matching. In *Proceedings of the European Conference on Computer Vision*, pages 201–216.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics.

Anna Rohrbach, Lisa Anne Hendricks, Kaylee Burns, Trevor Darrell, and Kate Saenko. 2018. Object hallucination in image captioning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4035–4045, Brussels, Belgium. Association for Computational Linguistics.

Yong Rui, Thomas S Huang, and Shih-Fu Chang. 1999. Image retrieval: Current techniques, promising directions, and open issues. *Journal of Visual Communication and Image Representation*, 10(1):39–62.

Naeha Sharif, Lyndon White, Mohammed Bennamoun, and Syed Afaq Ali Shah. 2018. Learning-based composite metrics for improved caption evaluation. In *Proceedings of ACL 2018, Student Research Workshop*, pages 14–20.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4566–4575.

Xin Wang, Qiuyuan Huang, Asli Celikyilmaz, Jianfeng Gao, Dinghan Shen, Yuan-Fang Wang, William Yang Wang, and Lei Zhang. 2019. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6629–6638.

# WSLLN: Weakly Supervised Natural Language Localization Networks

**Mingfei Gao**[1] * **Larry S. Davis**[1] **Richard Socher**[2] **Caiming Xiong**[2] †
[1]University of Maryland [2]Salesforce Research
{mgao,lsd}@umiacs.umd.edu, {rsocher,cxiong}@salesforce.com

## Abstract

We propose weakly supervised language localization networks (WSLLN) to detect events in long, untrimmed videos given language queries. To learn the correspondence between visual segments and texts, most previous methods require temporal coordinates (start and end times) of events for training, which leads to high costs of annotation. WSLLN relieves the annotation burden by training with only video-sentence pairs without accessing to temporal locations of events. With a simple end-to-end structure, WSLLN measures segment-text consistency and conducts segment selection (conditioned on the text) simultaneously. Results from both are merged and optimized as a video-sentence matching problem. Experiments on *ActivityNet Captions* and *DiDeMo* demonstrate that WSLLN achieves state-of-the-art performance.

## 1 Introduction

Extensive work has been done on temporal action/activity localization (Shou et al., 2016; Zhao et al., 2017; Dai et al., 2017; Buch et al., 2017; Gao et al., 2017c; Chao et al., 2018), where an action of interest is segmented from long, untrimmed videos. These methods only identify actions from a pre-defined set of categories, which limits their application to situations where only unconstrained language descriptions are available. This more general problem is referred to as natural language localization (NLL) (Hendricks et al., 2017; Gao et al., 2017a). The goal is to retrieve a temporal segment from an untrimmed video based on an arbitrary text query. Recent work focuses on learning the mapping from visual segments to the input text (Hendricks et al., 2017; Gao et al., 2017a; Liu et al., 2018; Hendricks et al., 2018; Zhang et al., 2018) and retrieving segments based on the alignment scores. However, in order to successfully train a NLL model, a large number of diverse language descriptions are needed to describe different temporal segments of videos which incurs high human labeling cost.

We propose Weakly Supervised Language Localization Networks (WSLLN) which requires only video-sentence pairs during training with no information of where the activities temporally occur. Intuitively, it is much easier to annotate video-level descriptions than segment-level descriptions. Moreover, when combined with text-based video retrieval techniques, video-sentence pairs may be obtained with minimum human intervention. The proposed model is simple and clean, and can be trained end-to-end in a single stage. We validate our model on *ActivityNet Captions* and *DiDeMo*. The results show that our model achieves the state-of-the-art of the weakly supervised approach and has comparable performance as some supervised approaches.

## 2 Related Work

**Temporal Action Localization** in long videos is widely studied in both offline and online scenarios. In the offline setting, temporal action detectors (Shou et al., 2016; Buch et al., 2017; Gao et al., 2017c; Chao et al., 2018) predict the start and end times of actions after observing the whole video, while online approaches (De Geest et al., 2016; Gao et al., 2017b; Shou et al., 2018b; Xu et al., 2018; Gao et al., 2019) label action class in a per-frame manner without accessing future information. The goal of temporal action detectors is to localize actions in pre-defined categories. However, activities in the wild is very complicated and it is challenging to cover all the activities of interest by using a finite set of categories.

*Work done when the author was at Salesforce Research.
†Corresponding author.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1481–1487, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

**Natural Language Localization** in untrimmed videos was first introduced in (Gao et al., 2017a; Hendricks et al., 2017), where given an arbitrary text query, the methods attempt to localize the text (predict its start and end times) in a video. Hendricks *et al.* proposed MCN (Hendricks et al., 2017) which embeds the features of visual proposals and sentence representations in the same space and ranks proposals according their similarity with the sentence. Gao *et al.* proposed CTRL (Gao et al., 2017a), where alignment and regression are conducted for clip candidates. Liu *et al.* introduced TMN (Liu et al., 2018) which measures the clip-sentence alignment guided by the semantic structure of the text query. Later, Hendricks *et al.* proposed MLLC (Hendricks et al., 2018) that explicitly reasons about temporal clips of a video. Zhang *et al.* proposed MAN (Zhang et al., 2018) which utilizes Graph Convolutional Networks (Kipf and Welling, 2016) to model temporal relations among visual clips. Although these methods achieve considerable success, they need segment-level annotations for training. Duan *et al.* proposed WSDEC to handle weakly supervised dense event captioning in (Duan et al., 2018) by alternating between language localization and caption generation iteratively. WSDEC generates language localization as intermediate results and can be trained using video-level labels. Thus, we set it as a baseline, although it is not designed for NLL.

**Weakly Supervised Localization** has been studied extensively to use weak supervisions for object detection on images and action localization in videos (Oquab et al., 2015; Bilen and Vedaldi, 2016; Tang et al., 2017; Gao et al., 2018; Kantorov et al., 2016; Li et al., 2016; Jie et al., 2017; Diba et al., 2017; Papadopoulos et al., 2017; Duchenne et al., 2009; Laptev et al., 2008; Bojanowski et al., 2014; Huang et al., 2016; Wang et al., 2017; Shou et al., 2018a). Some methods use class labels to train object detectors. Oquab *et al.* discussed that object locations may be freely obtained when training classification models (Oquab et al., 2015). Bilen *et al.* proposed WSDDN (Bilen and Vedaldi, 2016), which focuses on both object recognition and localization. Their proposed two-stream architecture inspired several weakly supervised approaches (Tang et al., 2017; Gao et al., 2018; Wang et al., 2017) including our method. Li *et al.* presented an adaptation strategy in (Li et al., 2016) which uses the output of a weak detector as pseudo groundtruth to train a detector in a fully supervised way. OICR (Tang et al., 2017) integrates multiple instance learning and iterative classifer refinement in a single network. Some works use other types of weak supervisions to optimize detectors. In (Papadopoulos et al., 2017), Papadopoulos *et al.* used clicks to train detectors. Gao *et al.* utilized object counts for weakly supervised object detection (Gao et al., 2018). Instead of using temporally labeled segments, weakly supervised action detectors use weaker annotations, *e.g.*, movie script (Duchenne et al., 2009; Laptev et al., 2008), the order of the occurring action classes in videos (Bojanowski et al., 2014; Huang et al., 2016) and video-level class labels (Wang et al., 2017; Shou et al., 2018a).

## 3 Weakly Supervised Language Localization Networks (WSLLN)

### 3.1 Problem Statement

Following the setting of its strongly supervised counterpart (Gao et al., 2017a; Hendricks et al., 2017), the goal of a weakly supervised language localization (WSLL) method is to localize the event that is described by a sentence query in a long, untrimmed video. Formally, given a video consisting of a sequence of image frames, $\mathbf{V}_i = [I_i^1, I_i^2, ..., I_i^T]$, and a text query $Q_i$, the model aims to localize a temporal segment, $[I_i^{st}, ..., I_i^{ed}]$, which semantically aligns best with the query. $st$ and $ed$ indicate the start and end times, respectively. The difference is that WSLL methods only utilize video-sentence pairs, $\{\mathbf{V}_i, Q_i\}_{i=1}^N$, for training, while supervised approaches have access to the start and end times of the queries.

### 3.2 The Proposed Approach

Taking frame sequences, $[I_i^1, I_i^2, ..., I_i^T]$, as inputs, the model first generates a set of temporal proposals, $\{p_i^1, p_i^2, ..., p_i^n\}$, where $p_i^j$ consists of temporally-continuous image frames. Then, the method aligns the proposals with the input query and outputs scores for proposals, $\{s_i^1, s_i^2, ..., s_i^n\}$, indicating their likelihood of containing the event.

**Feature Description**. Given a sentence query $Q_i$ of arbitrary length, sentence encoders can be used to extract text feature, $fq_i$, from the query. For a video, $\mathbf{V}_i = [I_i^1, I_i^2, ..., I_i^T]$, features, $\mathbf{fv}_i = [fv_i^1, fv_i^2, ..., fv_i^T]$, are extracted from each frame. Following (Hendricks et al., 2017), the visual fea-

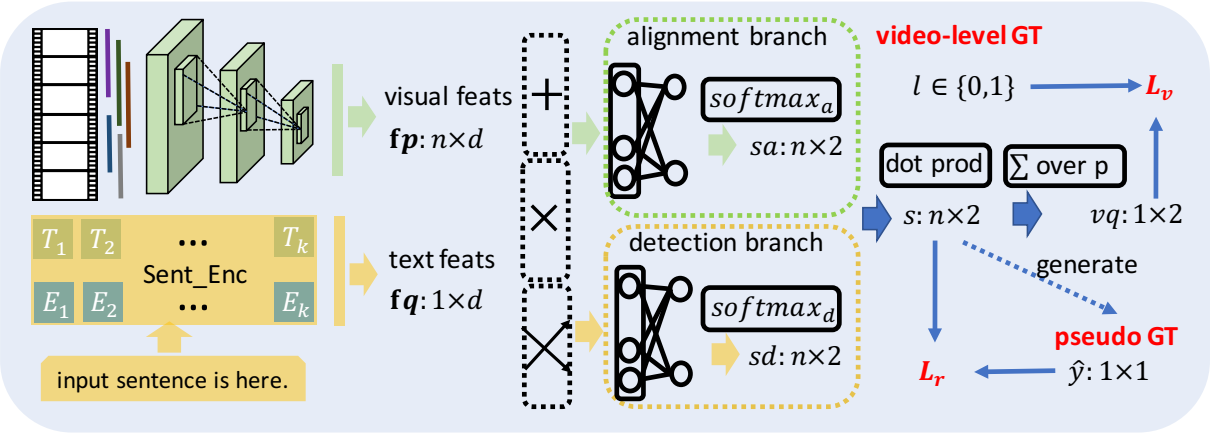


Figure 1: The workflow of our method. Visual and text features are extracted from $n$ video proposals and the input sentence. Fully-connected (FC) layers are used to transform the features to the same length, $d$. The two features are combined by multi-modal processing (Gao et al., 2017a) and input to the two-branch structure. Scores from both parts are merged. Video-level scores, $vq$, are obtained by summing $s$ over proposals. The whole pipeline is trained end-to-end using video-level and pseudo segment-level labels. $x \times z$ indicates dimensions.

ture, $fp_i^j$, of a proposal $p_i^j$ is obtained using Eq. 1, where $pool(x, t_1, t_2)$ means average pooling features $x$ from time $t_1$ to $t_2$, $||$ indicates concatenation, $j_{st}$/$j_{ed}$ indicates start/end times of the proposal and $\bar{j}$ means time is normalized to $[0, 1]$.

$$pool(\mathbf{fv}_i, j_{st}, j_{ed})||pool(\mathbf{fv}_i, 0, T)||[\bar{j}_{st}, \bar{j}_{ed}] \quad (1)$$

We see that the feature of each proposal contains the information of its visual pattern, the overall context and its relative position in the video.

Following (Gao et al., 2017a), features of the sentence and a visual proposal are combined as in Eq. 2. The feature, $fm$, will be used to measure the matching between a candidate proposal and the input query.

$$fm = (fp + fq)||(fp \cdot fq)||FC(fp||fq) \quad (2)$$

The workflow of WSLLN is illustrated in Fig. 1. Inspired by the success of the two-stream structure in the weakly supervised object and action detection tasks (Bilen and Vedaldi, 2016; Wang et al., 2017), WSLLN consists of two branches, *i.e.*, alignment branch and selection branch. The semantic consistency between the input text and each visual proposal is measured in the alignment branch. The proposals are compared and selected in the detection branch. Scores from both branches are merged to produce the final results.

**Alignment Branch** produces the consistency scores, $sa_i \in \mathrm{R}^{n\times 2} = [sa_i^1, sa_i^2, ..., sa_i^n]$, for proposals of the video-sentence pair. $sa_i$ in Eq. 3, measures how well each proposal matches the text. Different proposal scores are calculated independently where $softmax_a$ indicates applying the softmax function over the last dimension.

$$sa_i = softmax_a(\mathbf{W}_a fm_i) \quad (3)$$

**Detection Branch** performs proposal selection. The selection score, $sd_i \in \mathrm{R}^{n\times 2} = [sd_i^1, sd_i^2, ..., sd_i^n]$ in Eq. 4, is obtained by applying softmax function over proposals. Through softmax, the score of a proposal will be affected by those of other proposals, so this operation encourages competition among segments.

$$sd_i = softmax_d(\mathbf{W}_d fm_i) \quad (4)$$

**Score Merging** is applied to both parts to obtain the results by dot production, *i.e.*, $s_i = sa_i \cdot sd_i$, for proposals. $s_i$ is used as the final segment-sentence matching scores during inference.

**Training Phase**. To utilize video-sentence pairs as supervision, our model is optimized as a video-sentence matching classifier. We compute the matching score of a given video-sentence pair by summing $s_i^j$ over proposals, $vq_i = \sum_{j=1}^{n} s_i^j$. Then, $L_v$ is obtained in Eq. 5 by measuring the score with the video-sentence match label $l_i \in \{0, 1\}$. Positive video-sentence pairs can be obtained directly. We generate negative ones by pairing each video with a randomly selected sentence in the training set. We ensure that the positive pairs are not included in the negative set.

$$L_v = loss(vq_i, l_i) \quad (5)$$

Results can be further refined by adding an auxiliary task $L_r$ in Eq. 6 where $\hat{y}_i = \{0, 1, ..., n-1\}$ indicates the index of the segment that best matches the sentence during training. The real segment-level labels are not available, thus we generate pseudo labels by setting $\hat{y}_i = \text{argmax}_j s_i^j[:, 1]$. This loss further encourages competition among proposals.

$$L_r = loss(s_i^j, \hat{y}_i) \tag{6}$$

The overall objective is minimizing $L$ in Eq. 7, where $\lambda$ is a balancing scalar. $loss$ is cross-entropy loss.

$$L = loss(vq_i, l_i) + \lambda loss(s_i^j, \hat{y}_i). \tag{7}$$

# 4 Experiments

## 4.1 Experimental Settings

**Implementation Details**. BERT (Devlin et al., 2018) is used as the sentence encoder, where the feature of '[CLS]' at the last layer is extracted as the sentence representation. Visual and sentence features are linearly transformed to have the same dimension, $d = 1000$. The hidden layers for both branches have 256 units. For *ActivityNet Captions*, we take the $n = 15$ proposals over multiple scales of each video provided by (Duan et al., 2018) and use the C3D (Tran et al., 2015) features provided by (Krishna et al., 2017). For *DiDeMo*, we use the $n = 21$ proposals and VGG (Simonyan and Zisserman, 2014) features (RGB and Flow) provided in (Hendricks et al., 2017).

**Evaluation Metrics**. Following (Gao et al., 2017a; Hendricks et al., 2017), *R@k,IoU=th* and *mIoU* are used for evaluation. Proposals are ranked according to their matching scores with the input sentence. If the temporal IoU between at least one of the top-k proposals and the groundtruth is bigger or equal to $th$, the sentence is counted as matched. *R@k,IoU=th* means the percentage of matched sentences over the total sentences given $k$ and $th$. *mIoU* is the mean IoU between the top-1 proposal and the groundtruth.

## 4.2 Experiments on ActivityNet Captions

**Dataset Description**. *ActivityNet Captions* (Krishna et al., 2017) is a large-scale dataset of human activities. It contains 20k videos including 100k video-sentences in total. We train our models on the training set and test them on the validation set.

| Model | WS | IoU=0.1 | IoU=0.3 | IoU=0.5 | mIoU |
|---|---|---|---|---|---|
| CTRL | F | 49.1 | 28.7 | 14.0 | 20.5 |
| ABLR | F | 73.3 | 55.7 | 36.8 | 37.0 |
| WSDEC-S | F | 70.0 | 52.9 | 37.6 | 40.4 |
| WSDEC-W | T | 62.7 | 42.0 | 23.3 | 28.2 |
| **WSLLN** | T | 75.4 | 42.8 | 22.7 | 32.2 |

Table 1: Comparison results based on $R@1$ on *ActivityNet Captions*. All baseline numbers are reprinted from (Duan et al., 2018). WS: weakly supervised.

| $\lambda \rightarrow$ | 0.0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
|---|---|---|---|---|---|---|
| IoU=0.1 | 64.9 | 75.4 | 75.5 | 75.5 | 75.5 | 66.6 |
| IoU=0.3 | 36.2 | 42.8 | 42.9 | 42.9 | 42.9 | 38.3 |
| IoU=0.5 | 19.4 | 22.7 | 22.7 | 22.8 | 22.7 | 20.7 |
| mIoU | 27.4 | 32.2 | 32.3 | 32.3 | 32.3 | 28.8 |

Table 2: R@1 results of our method on *ActivityNet Captions* when $\lambda$ in Eq. 7 is set to be different values.

Although the dataset provides segment-level annotation, we only use video-sentence pairs during training.

**Baselines**. We compare with strongly supervised approaches, *i.e.*, CTRL (Gao et al., 2017a), ABLR (Yuan et al., 2018) and WSDEC-S (Duan et al., 2018) to see how much accuracy it sacrifices when using only weak labels. Originally proposed for dense-captioning, WSDEC-W (Duan et al., 2018) achieves state-of-the-art performance for weakly supervised language localization. Although showing good performance, WSDEC-W involves complicated training stages, and alternates between sentence localization and caption generation for iterations.

### 4.2.1 Comparison Results

Comparison results are displayed in Tab. 1. It shows that WSLLN largely outperforms WSDEC-W by $\sim 4\%$ $mIoU$. When comparing with strongly supervised methods, WSLLN outperforms CTRL by over $11\%$ $mIoU$. Using the $R@1, IoU = 0.1$ metric, our model largely outperforms all the baselines including strongly and weakly supervised methods which means that when a scenario is flexible with the IoU coverage, our method has great advantage over others. When $th$ =0.3/0.5, our model has comparable results as WSDEC-W and largely outperforms CTRL. The overall results demonstrate good performance of WSLLN, even though there is still a big gap between weakly supervised methods and

some supervised ones, *i.e.*, ABLR and WSDEC-S. $mIoU$ (mean±std) of WSLLN across 3 runs is $32.2 \pm 0.05$ which demonstrates the robustness of our method.

#### 4.2.2 Ablation Study

**Effect of** $\lambda$. We evaluate the effect of $\lambda$ (see Eq. 7) in Tab. 2. As it shows, our model performs stable when $\lambda$ is set from $0.1$ to $0.4$. When $\lambda = 0$, the refining module is disabled and the performance drops. When $\lambda$ is set to a big number, *e.g.*, $0.5$, the contribution of $L_v$ is reduced and the model performance also drops.

**Effect of Sentence Encoder**. WSDEC-W uses GRU (Cho et al., 2014) as its sentence encoder, while our method uses BERT. It seems an unfair comparison, since BERT is powerful than GRU in general. However, we uses pretrained BERT model without fine tuning on our dataset, while WSDEC-W uses GRU but performed an end-to-end training. So, it is unclear which setting is better. To resolve this concern, we replace our BERT with GRU following WSDEC-W. The $R@1$ results when $IoU$ is set to be 0.1, 0.3 and 0.5 are 74.0, 42.3 and 22.5, respectively. The mIoU is 31.8. It shows that our model with GRU has comparable results as that with BERT.

**Effect of Two-branch Design**. We create two baselines, *ie*, *Align-only* and *Detect-only*, to demonstrate the effectiveness of our design. To perform fair comparison, both of them are trained using only video-sentence pairs.

*Align-only* contains only the alignment branch. For positive video sentence pair, we give positive labels to all proposals. Negative pairs have negative labels for all the proposals. Loss is calculated between proposal scores and the generated segment-level labels.

*Detect-only* contains only the detection branch. Loss is calculated using the highest detection score over proposals and the video-level label at each training iteration.

Comparison results are displayed in Tab. 3. It shows that the two baselines underperform WSLLN by a large margin, which demonstrates the effectiveness of our design.

### 4.3 Experiments on DiDeMo

**Dataset Description**. *DiDeMo* was proposed in (Hendricks et al., 2017) for the language localization task. It contains 10k, 30-second videos including 40k annotated segment-sentence pairs.

| Model | IoU=0.1 | IoU=0.3 | IoU=0.5 | mIoU |
|---|---|---|---|---|
| Align-only | 40.0 | 18.9 | 7.5 | 13.4 |
| Detect-only | 33.7 | 18.3 | 10.4 | 13.6 |

Table 3: Ablation study based on $R@1$ on *ActivityNet Captions*. Both methods are trained using weak supervisions.

| Model | WS | Input | R@1 | R@5 | mIoU |
|---|---|---|---|---|---|
| Chance | – | – | 3.75 | 22.50 | 22.64 |
| LOR | F | RGB | 16.2 | 43.9 | 27.2 |
| MCN | F | RGB | 23.1 | 73.4 | 35.5 |
| MCN | F | Flow | 25.8 | 75.4 | 38.9 |
| **WSLLN** | T | RGB | 19.4 | 53.1 | 25.4 |
| **WSLLN** | T | Flow | 18.4 | 54.4 | 27.4 |

Table 4: Comparison results on *DiDeMo*. Following MCN, we set $th = 1.0$ for the IoU threshold. All baseline numbers are reprinted from (Hendricks et al., 2017). WS: weakly supervised.

Our models are trained using video-sentence pairs in the train set and tested on the test set.

**Baselines**. To the best of our knowledge, no weakly supervised method has been evaluated on *DiDeMo*. So, we compare with some supervised methods, *i.e.*, MCN (Hendricks et al., 2017) and LOR (Hu et al., 2016). MCN is a supervised NLL model. LOR is a supervised language-object retrieval model. It utilizes much more expensive (object-level) annotations for training. We follow the same setup of LOR as in (Hendricks et al., 2017) to evaluate LOR for our task.

**Comparison Results** are shown in Tab. 4. WSLLN performs better than LOR in terms of $R@1/5$. We also observe that the gap between our method and the supervised NLL model is much larger on *DiDeMo* than on *ActivityNet Captions*. This may be due to the fact that *DiDeMo* is a much smaller dataset which is a disadvantage for weakly supervised learning.

## 5 Conclusion

We propose WSLLN– a simple language localization network. Unlike most existing methods which require segment-level supervision, our method is optimized using video-sentence pairs. WSLLN is based on a two-branch architecture where one branch performs segment-sentence alignment and the other one conducts segment selection. Experiments show that WSLLN achieves promising results on *ActivityNet Captions* and *DiDeMo*.

## References


Hakan Bilen and Andrea Vedaldi. 2016. Weakly supervised deep detection networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Piotr Bojanowski, Rémi Lajugie, Francis Bach, Ivan Laptev, Jean Ponce, Cordelia Schmid, and Josef Sivic. 2014. Weakly supervised action labeling in videos under ordering constraints. In *European Conference on Computer Vision*, pages 628–643. Springer.

Shyamal Buch, Victor Escorcia, Chuanqi Shen, Bernard Ghanem, and Juan Carlos Niebles. 2017. SST: Single-stream temporal action proposals. In *CVPR*.

Yu-Wei Chao, Sudheendra Vijayanarasimhan, Bryan Seybold, David A Ross, Jia Deng, and Rahul Sukthankar. 2018. Rethinking the faster r-cnn architecture for temporal action localization. In *CVPR*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv:1406.1078*.

Xiyang Dai, Bharat Singh, Guyue Zhang, Larry S Davis, and Yan Qiu Chen. 2017. Temporal context network for activity localization in videos. In *ICCV*.

Roeland De Geest, Efstratios Gavves, Amir Ghodrati, Zhenyang Li, Cees Snoek, and Tinne Tuytelaars. 2016. Online action detection. In *ECCV*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Ali Diba, Vivek Sharma, Ali Pazandeh, Hamed Pirsiavash, and Luc Van Gool. 2017. Weakly supervised cascaded convolutional networks. In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5131–5139.

Xuguang Duan, Wenbing Huang, Chuang Gan, Jingdong Wang, Wenwu Zhu, and Junzhou Huang. 2018. Weakly supervised dense event captioning in videos. In *Advances in Neural Information Processing Systems*, pages 3059–3069.

Olivier Duchenne, Ivan Laptev, Josef Sivic, Francis R Bach, and Jean Ponce. 2009. Automatic annotation of human actions in video. In *ICCV*, volume 1, pages 3–2.

Jiyang Gao, Chen Sun, Zhenheng Yang, and Ram Nevatia. 2017a. Tall: Temporal activity localization via language query. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5267–5275.

Jiyang Gao, Zhenheng Yang, and Ram Nevatia. 2017b. RED: Reinforced encoder-decoder networks for action anticipation. In *BMVC*.

Jiyang Gao, Zhenheng Yang, Chen Sun, Kan Chen, and Ram Nevatia. 2017c. TURN TAP: Temporal unit regression network for temporal action proposals. *ICCV*.

Mingfei Gao, Ang Li, Ruichi Yu, Vlad I Morariu, and Larry S Davis. 2018. C-wsl: Count-guided weakly supervised localization. In *ECCV*.

Mingfei Gao, Mingze Xu, Larry S Davis, Richard Socher, and Caiming Xiong. 2019. Startnet: Online detection of action start in untrimmed videos. *arXiv preprint arXiv:1903.09868*.

Lisa Anne Hendricks, Oliver Wang, Eli Shechtman, Josef Sivic, Trevor Darrell, and Bryan Russell. 2017. Localizing moments in video with natural language. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5803–5812.

Lisa Anne Hendricks, Oliver Wang, Eli Shechtman, Josef Sivic, Trevor Darrell, and Bryan Russell. 2018. Localizing moments in video with temporal language. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Ronghang Hu, Huazhe Xu, Marcus Rohrbach, Jiashi Feng, Kate Saenko, and Trevor Darrell. 2016. Natural language object retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4555–4564.

De-An Huang, Li Fei-Fei, and Juan Carlos Niebles. 2016. Connectionist temporal modeling for weakly supervised action labeling. In *European Conference on Computer Vision*, pages 137–153. Springer.

Zequn Jie, Yunchao Wei, Xiaojie Jin, Jiashi Feng, and Wei Liu. 2017. Deep self-taught learning for weakly supervised object localization. *IEEE CVPR*.

Vadim Kantorov, Maxime Oquab, Minsu Cho, and Ivan Laptev. 2016. Contextlocnet: Context-aware deep network models for weakly supervised localization. In *European Conference on Computer Vision*, pages 350–365. Springer.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. 2017. Dense-captioning events in videos. In *International Conference on Computer Vision (ICCV)*.

Ivan Laptev, Marcin Marszalek, Cordelia Schmid, and Benjamin Rozenfeld. 2008. Learning realistic human actions from movies. In *CVPR*.

Dong Li, Jia-Bin Huang, Yali Li, Shengjin Wang, and Ming-Hsuan Yang. 2016. Weakly supervised object localization with progressive domain adaptation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Bingbin Liu, Serena Yeung, Edward Chou, De-An Huang, Li Fei-Fei, and Juan Carlos Niebles. 2018. Temporal modular networks for retrieving complex compositional activities in videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 552–568.

Maxime Oquab, Léon Bottou, Ivan Laptev, and Josef Sivic. 2015. Is object localization for free?-weakly-supervised learning with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 685–694.

Dim P Papadopoulos, Jasper RR Uijlings, Frank Keller, and Vittorio Ferrari. 2017. Training object class detectors with click supervision. *CVPR*.

Zheng Shou, Hang Gao, Lei Zhang, Kazuyuki Miyazawa, and Shih-Fu Chang. 2018a. Autoloc: Weakly-supervised temporal action localization in untrimmed videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 154–171.

Zheng Shou, Junting Pan, Jonathan Chan, Kazuyuki Miyazawa, Hassan Mansour, Anthony Vetro, Xavier Giro-i Nieto, and Shih-Fu Chang. 2018b. Online action detection in untrimmed, streaming videos-modeling and evaluation. In *ECCV*.

Zheng Shou, Dongang Wang, and Shih-Fu Chang. 2016. Temporal action localization in untrimmed videos via multi-stage cnns. In *CVPR*.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*.

Peng Tang, Xinggang Wang, Xiang Bai, and Wenyu Liu. 2017. Multiple instance detection network with online instance classifier refinement. *CVPR*.

Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. 2015. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*.

Limin Wang, Yuanjun Xiong, Dahua Lin, and Luc Van Gool. 2017. Untrimmednets for weakly supervised action recognition and detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4325–4334.

Mingze Xu, Mingfei Gao, Yi-Ting Chen, Larry S Davis, and David J Crandall. 2018. Temporal recurrent networks for online action detection. *arXiv:1811.07391*.

Yitian Yuan, Tao Mei, and Wenwu Zhu. 2018. To find where you talk: Temporal sentence localization in video with attention based location regression. *arXiv preprint arXiv:1804.07014*.

Da Zhang, Xiyang Dai, Xin Wang, Yuan-Fang Wang, and Larry S Davis. 2018. Man: Moment alignment network for natural language moment retrieval via iterative graph adjustment. *arXiv preprint arXiv:1812.00087*.

Yue Zhao, Yuanjun Xiong, Limin Wang, Zhirong Wu, Xiaoou Tang, and Dahua Lin. 2017. Temporal action detection with structured segment networks. In *ICCV*.

# Grounding learning of modifier dynamics: An application to color naming

**Xudong Han** ♠ **Philip Schulz** ♡ **Trevor Cohn** ♠
♠ University of Melbourne
♡ Amazon Research
xudongh1@student.unimelb.edu.au
phschulz@amazon.com
trevor.cohn@unimelb.edu.au

## Abstract

Grounding is crucial for natural language understanding. An important subtask is to understand modified color expressions, such as *"dirty blue"*. We present a model of color modifiers that, compared with previous additive models in RGB space, learns more complex transformations. In addition, we present a model that operates in the HSV color space. We show that certain adjectives are better modeled in that space. To account for all modifiers, we train a hard ensemble model that selects a color space depending on the modifier-color pair. Experimental results show significant and consistent improvements compared to the state-of-the-art baseline model.[1]

## 1 Introduction

Grounded color descriptions are employed to describe colors which are not covered by basic color terms (Monroe et al., 2017). For instance, *"greenish blue"* cannot be expressed by only *"blue"* or *"green"*. Grounded learning of modifiers, as a result, is essential for grounded language understanding problems such as image captioning (Karpathy and Fei-Fei, 2015), visual question answering (Goyal et al., 2017) and object recognition (van de Sande et al., 2010).

In this paper, we present models that are able to predict the RGB code of a target color given a reference color and a modifier. For example, as shown in Figure 1, given a reference color code $\vec{r} = \begin{bmatrix}101 & 55 & 0\end{bmatrix}^\top$ and a modifier $m =$ *"greenish"*, our models are trained to predict the target color code $\vec{t} = \begin{bmatrix}105 & 97 & 18\end{bmatrix}^\top$. The state-of-the-art approach for this task (Winn and Muresan, 2018) represents both colors and modifiers as vectors in RGB space, and learns a vector representation of modifiers $\vec{m}$ as part of a simple additive model, $\vec{r} + \vec{m} \approx \vec{r}$, in RGB color space. For instance, given the reference color $\vec{r} = \begin{bmatrix}229 & 0 & 0\end{bmatrix}^\top$, the target color $\vec{t} = \begin{bmatrix}132 & 0 & 0\end{bmatrix}^\top$, the modifier $m =$ *"darker"* is learned as a vector $\vec{m} = \begin{bmatrix}-97 & 0 & 0\end{bmatrix}^\top$. This model works well when the modifier is well represented as a single vector independent of the reference color,[2] but fails to model modifiers with more complex transformations, for example, color related modifiers, like *"greenish"*, which are better modelled through color interpolation.

| Reference | Modifier | Target |
|---|---|---|
| $[101 \quad 55 \quad 0]^T$ | greenish | $[105 \quad 97 \quad 18]^T$ |
| $[229 \quad 0 \quad 0]^T$ | darker | $[132 \quad 0 \quad 0]^T$ |
| $[3 \quad 67 \quad 223]^T$ | more neon | $[4 \quad 217 \quad 255]^T$ |

Figure 1: Examples of the grounded modifier modelling task, shown in RGB space. Given the reference and modifier, the system must predict the target color.

To fit a better model, we assume that there are approximate intersection points for the extension lines of modifier vectors, for instance, Figure 2c shows the *"darker"* related vectors in RGB space and we can see that the intersection point is approximately $\begin{bmatrix}0 & 0 & 0\end{bmatrix}^\top$. On the basis of this, we introduce an RGB model which can learn a transformation matrix and an interpolation point for each modifier.

There are many other color spaces besides RGB, e.g., HSL and HSV (Joblove and Greenberg, 1978; Hanbury, 2008), and mapping between such

[1] Code available at https://github.com/HanXudong/GLoM

[2] Winn and Muresan (2018) parameterize $\vec{m}$ as a function of $m$ and $\vec{r}$, such that the modifier vectors can adapt to the different reference colors, but showed that this had limited effect in practise.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1488–1493, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

spaces can be done via invertible transformations (Agoston, 2005). As shown in Figure 2d, for some modifiers, color vectors can be approximately parallel in HSV space, thus simplifying the modeling problem. We propose a HSV model using a von-Mises loss, and show that this model outperforms the RGB method for many modifiers. We also present an ensemble model for color space selection, which determines for each modifier the best color space model. Overall our methods substantially outperform prior work, achieving state-of-the-art performance in the grounded color modelling task.

## 2 Methods

Here we describe the methods employed for color modeling. Formally, the task is to predict a target color vector $\vec{t}$ in RGB space from a reference color vector $\vec{r}$ and a modifier string $m$.

### 2.1 Modeling in RGB space

**Baseline Model** Winn and Muresan (2018) present a model (WM18) which represents a vector $\vec{m} \in \mathbb{R}^3$ as a function of $(m,\ \vec{r})$ pointing from a reference color vector $\vec{r}$ to the target color vector $\vec{t}$, such that $\vec{t} = \vec{r} + \vec{m}$. In the simplest case the modifier $m$ is irrelevant to the reference color $\vec{r}$. This assumption, however, does not hold in all situations. For example, when predicting an instance with the reference vector $\vec{r} = \begin{bmatrix}193 & 169 & 106\end{bmatrix}^\top$ and the modifier *"greenish"*, the outcome $\hat{t}$ is expected to be $\begin{bmatrix}177 & 183 & 102\end{bmatrix}^\top$, however, WM18 predicts $\hat{t} = \begin{bmatrix}195 & 156 & 95\end{bmatrix}^\top$. The cosine similarity between $(\hat{t} - \vec{r})$ and $(\vec{t} - \vec{r})$ is $-0.76$, i.e., $\vec{m}$ points in the opposite direction to where it should.

**RGB Model** As shown in Figure 2c, pairs of vectors for the same modifier are often not parallel. In theory, such vectors can even be orthogonal: compare *"darker red"* vs *"darker blue"*, which fall on different faces of the RGB cube. To model this, we propose a model in RGB space as follows:

$$\vec{t} = M\vec{r} + \vec{\beta} \tag{1}$$

Where $M \in \mathbb{R}^{3\times3}$ is a transformation matrix and $\vec{\beta}$ is a modifier vector which is designed to capture the information of $m$. Given an error term $\varepsilon \sim \mathcal{N}(\vec{0}, \sigma I_3)$, the RGB model is trained to minimize the following loss for the log Gaussian likelihood:

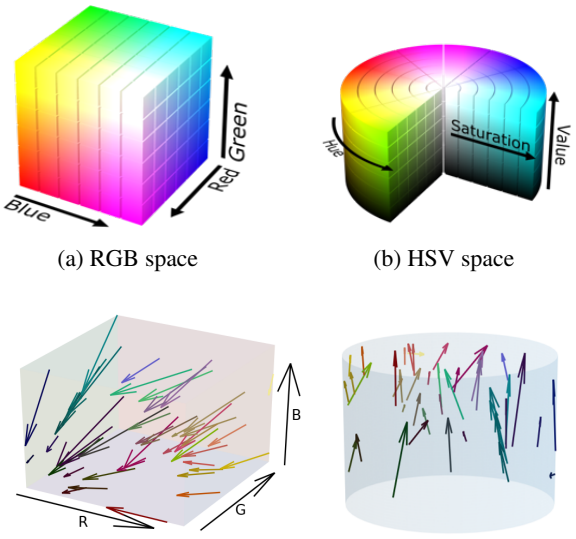


(a) RGB space (b) HSV space

(c) "dark" in RGB (d) "dark" in HSV

Figure 2: 2a: RGB color space. 2b: HSV color space. Arrows in 2c and 2d show vectors staring from reference colors to target colors in RGB and HSV color spaces. Images 2a and 2b by Michael Horvath, available under Creative Commons Attribution-Share Alike 3.0 Unported license.

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}(\vec{t_i} - \hat{t_i})^\top(\vec{t_i} - \hat{t_i}) \tag{2}$$

where $\vec{t_i}$ is the target vector in each instance and $\hat{t}_i$ is the prediction.[3]

**Specific Settings** Our model generalizes WM18, which can be realized by setting $M = I_3$ and $\vec{\beta} = \vec{m}$.

Another interesting instance of the model is obtained by setting $M = (1 - \alpha_m)I_3$ and $\vec{\beta} = \alpha_m\vec{m}$, which we call the Diagonal Covariance (DC) model. In contrast to the RGB model and WM18, which model $\vec{m}$ as a function of $\vec{r}$ and $m$, $\vec{m}$ in the DC model does not depend on $\vec{r}$. Given $\vec{r}$ and $m$, to predict the $\vec{t}$, our DC model predicts $\vec{m}$ first and then applies a linear transformation to get the target color vector as follows: $\vec{t} = \vec{r} + \alpha_m \times (\vec{m} - \vec{r})$, where $\alpha_m \in [0, 1]$ is a scalar which only depends on $m$ and measures the distance from $\vec{r}$ to $\vec{m}$. In the DC model $\vec{m}$ is the interpolation point for modifiers, such as $\begin{bmatrix}0 & 0 & 0\end{bmatrix}^\top$ for the modifier *"darker"*.

### 2.2 Modeling in HSV space

Compared with the RGB color space, when modeling modifiers in HSV, there are two main differ-

[3] Although other distributions, such as Beta distribution, may model uncertainties better, as focusing on mean value prediction, Gaussian models provide similar performances.

ences: hue is represented as an angular dimension, and the modifier vectors are more frequently parallel (see Fig 2d). As shown in Figure 2b, HSV space forms a cylindrical geometry with *hue* as its angular dimension, with the value red occurring at both 0 and $2\pi$. For this reason, modelling the hue with a Gaussian regression loss is not appropriate. To account for the angularity, we model *"hue"* with a von-Mises distribution, with the following *pdf*:

$$f(h) = \frac{\exp\left(k\cos\left(h-\hat{h}\right)\right)}{2\pi I_0(k)}. \quad (3)$$

The mean value $\hat{h}$ represents the center of *hue* dimension, $k$ indicates the concentration about the mean, and $I_0(k)$ is the modified Bessel function of order 0.

When training the model, the parameter $k$ is assumed constant, and thus the loss function is:

$$\mathcal{L} = 1 - \frac{1}{n}\sum_{i=1}^{n}\cos\left(h_i - \hat{h}_i\right), \quad (4)$$

where $h_i$ is the *hue* value of the target color in each instance and $\hat{h}_i$ is the prediction.

The second difference to modeling in RGB space is that the modifier behavior is simpler in HSV. For modifiers, vectors from reference colors to target colors are more likely to be parallel (see Figure 2d). As a result, we present an additive model in HSV space where a modifier $m$ will be modeled as a vector from $\vec{r}$ to $\vec{t}$:

$$\vec{t} = \vec{r} + \vec{m} \quad (5)$$

Here $\vec{m}$ is a function of both $m$ and $\vec{r}$. In addition, modifier modeling will be split into two parts: modeling *"hue"* dimension as von-Mises distribution and other dimensions together as a bivariate normal distribution (See Equation 2). Notice that Equation (5) is the same equation as used by WM18, however, here it is applied in a color space that better fits its assumptions.

WM18 is presented and evaluated only in RGB color space. To compare its performance with our models, we transform output into RGB space.

### 2.3 Ensemble model

An ensemble model is trained to make the final prediction, which we frame as a hard binary choice to select the color space appropriate for the given modifier. This works by applying the general RGB model and HSV model (Equations 1 and 5) to get their predictions, and converting the HSV predictions into RGB space. Then the hard ensemble is trained to predict which color space should be used based on the modifier $m$ and the reference vector $\vec{r}$, using as the learning signal which model prediction had the smallest error against the reference colour for each instance (measured using Delta-E distance, see §3.2). The probability of the RGB model being selected is: $p = \sigma(f(m, \vec{r}))$, where $\sigma$ is the logistic sigmoid function and $f(m, \vec{r})$ is a function of modifier $m$ and reference color $\vec{r}$.

## 3 Experiments

### 3.1 Dataset

The dataset[4] used to train and evaluate our model includes 415 triples (reference color label, $r$, modifier, $m$, and target color label, $t$) in RGB space presented by Winn and Muresan (2018). Munroe (2010) collected the original dataset consisting of color description pairs collected in an open online survey; the dataset was subsequently filtered by McMahan and Stone (2015). Winn and Muresan processed color labels and converted pairs to triples with 79 unique reference color labels and 81 unique modifiers.

We train models in both RGB and HSV color space, but samples in WM18 are only presented in RGB space. Because modifiers encode the general relationship between $r$ and $t$ we use the same approach presented by Winn and Muresan (2018): using the mean value of a set of points to represent a color. A drawback of this approach is that it does not account for our uncertainty about the appropriate RGB encoding for a given color word.

### 3.2 Experiment Setup

**Model configuration:** The model presented by Winn and Muresan (2018) is initialized with Google's pretrained 300-d word2vec embeddings (Mikolov et al., 2013b,a) which are not updated during training. To perform comparable experiments, all models in paper are designed with the same pre-trained embedding model. Other pre-trained word embeddings, such as GloVe (Pennington et al., 2014) and BERT (Devlin et al., 2019), were also tested but there was no significant

[4] https://bitbucket.org/o_winn/comparative_colors

| | Cosine Similarity ± SD (↑) | | | | |
|---|---|---|---|---|---|
| Test Condition | **RGB** | **WM18** | **HSV** | **Ensemble** | **WM18*** |
| Seen Pairings | 0.954±0.001 | 0.953±0.000 | 0.934±0.089 | **0.954**±0.0 | 0.68 |
| Unseen Pairings | 0.799±0.044 | 0.771±0.032 | **0.843**±0.144 | 0.797±0.0 | 0.68 |
| Unseen Ref. Color | 0.781±0.015 | 0.767±0.010 | **0.945**±0.019 | 0.804±0.0 | 0.40 |
| Unseen Modifiers | 0.633±0.042 | 0.637±0.032 | **0.724**±0.131 | 0.629±0.0 | 0.41 |
| Fully Unseen | 0.370±0.029 | 0.358±0.038 | **0.919**±0.026 | 0.445±0.0 | -0.21 |
| Overall | 0.858±0.006 | 0.856±0.003 | **0.911**±0.057 | 0.868±0.0 | 0.65 |
| | **Delta-E ± SD (↓)** | | | | |
| Test Condition | RGB | WM18 | HSV | Ensemble | WM18* |
| Seen Pairings | **3.121**±0.027 | 3.188±0.062 | 5.380±4.846 | 4.093±0.1 | 6.1 |
| Unseen Pairings | 6.454±0.233 | 6.825±0.093 | 11.701±3.358 | **5.873**±0.0 | 7.9 |
| Unseen Ref. Color | 7.456±0.184 | 7.658±0.363 | 10.429±2.523 | **7.171**±0.0 | 11.4 |
| Unseen Modifiers | 13.288±1.082 | 13.891±1.077 | 14.183±5.175 | 10.927±0.0 | **10.5** |
| Fully Unseen | 13.859±0.874 | 14.516±0.587 | **12.432**±2.170 | 13.448±0.0 | 15.9 |
| Overall | **5.412**±0.169 | 5.595±0.128 | 7.487±3.940 | 5.777±0.0 | 6.8 |

Table 1: Average cosine similarity score and Delta-E distance over 5 runs. A smaller Delta-E distance means a less significant difference between two colors. **Bold**: best performance. Hard: the hard ensemble model. WM18*: the performance from WM18 paper. See *Supplementary Material* for example outputs and ensemble analysis.

difference in performance compared to word2vec. Single models are trained over 2000 epochs with batch size 32 and 0.1 learning rate. The hyperparameters for the ensemble model are as follows: 600 epochs, 32 batch size, and 0.1 learning rate.

**Architecture:** An input modifier is represented as a vector by word2vec pretrained embeddings and followed by two fully connected layers($FC_1$ and $FC_2$) with size 32 and 16 respectively. Let $h_1$ be the hidden state of $FC_2$ then $h_1 = FC_2(FC_1(\vec{r}, E_m, \vec{r})$ where $E$ are fixed, pretrained word2vec embeddings. $\vec{r}$ is used as an input for both $FC_1$ and $FC_2$. After $FC_2$, all the other layers are based on hidden state $h_1$.

**Evaluation:** Following Winn and Muresan (2018), we evaluate the performance in 5 distinct input conditions: (1) *Seen Pairings:* The triple $(r, m, t)$ has been seen when training models. (2) *Unseen Pairings:* Both $r$ and $m$ have been seen in training data, but not the triple $(r, m, t)$. (3) *Unseen Ref. Color:* $r$ has not been seen in training, while $m$ has been seen. (4) *Unseen modifiers:* $m$ has not been seen in training, while $r$ has been seen. (5) *Fully Unseen:* Neither $r$ nor $m$ have been seen in training.

Because of the small size of the dataset, we report the average performance over 5 runs with different random seeds. Two scores, cosine similarity, and Delta-E are applied for evaluating the performance. Cosine similarity measures the difference in terms of vector direction in color space and Delta-E is a non-uniformity metric for measuring color differences. Delta-E was first presented as the Euclidean Distance in CIELAB color space (McLaren, 1976). Lower Delta-E values are thus preferable as they indicate better matching of the target color. Luo et al. (2001) present the latest and most accurate CIE color difference metrics, Delta-E 2000, which improve the original formula by taking into account weighting factors and fixing the lightness inaccuracies. Our models are evaluated with Delta-E 2000.

### 3.3 Results

Table 1 shows the results. Compared with WM18, our RGB model outperforms under all conditions. As we have stated, our model is a generalization of their approach. The more complex transformation matrix in our RGB model is able to learn more information, such as the effects of covariance between color channels, and thus achieves a better performance than WM18. Note that our reimplementation of the original WM18 system lead to significantly better performance.[5]

According to the cosine similarity, the HSV model is superior for most test conditions (con-

[5]We train the model for many more epochs, which is required for a good to the training data. This can be observed in terms of the 'seen pairings', where the WM18 reported results appear to be underfitting.

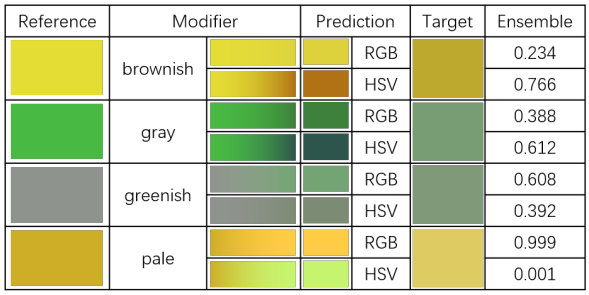


Figure 3: Examples of predictions from RGB, HSV and Ensemble model. The Ensemble column reports the predicted probability of the RGB and HSV models being selected.

firming our hypothesis about simpler modifier behaviour in this space). However for Delta-E, the RGB model and ensemble perform better. Unlike cosine, Delta-E is sensitive to differences in vector length, and we would argue it is the most appropriate metric because lengths are critical to measuring the extent of lightness and darkness of colors. Accordingly the HSV model does worse under this metric, as it more directly models the direction of color modifiers, but as a consequence this leads to errors in its length predictions. Overall the ensemble does well according to both metrics, and has the best performance for several test conditions with Delta-E.

**Error Analysis** We first focused error analysis on prediction of "Unseen Modifiers" and "Fully Unseen" instances. As shown in Table 1, our models are able to predict target colors given seen modifiers but fail to make predictions for instances with unseen modifiers. All modifiers are represented by word2vec embeddings, and we expect that predictions of unseen modifiers should be close to instances with similar seen modifiers. For example, the prediction of a reference color $\vec{r}$ and modifier *"greeny"* should be similar to the prediction of the same reference color $\vec{r}$ and a similar seen modifier, e.g. *"green"* and *"greenish"*. However, the prediction of *"greeny"* is more similar to *"bluey"*, a consequence of these terms having highly similar word embeddings (as do other colour modifiers with a *-y* suffix, irrespective of their colour). This is related to the problem reported in Mrkšić et al. (2016), whereby words and their antonyms often have similar embeddings, as a result of sharing similar distributional contexts. Accordingly for the *unseen modifier* condition, our model is often misled by attempting to generalise from nearest neighbour modifiers which have a different meaning.

## 4 Related Work

Baroni and Zamparelli (2010) was the first work to propose an approach to adjective-noun composition (AN) for corpus-based distributional semantics which represents nouns as vectors and adjectives as matrices nominal vectors. However it is hard to gain an intuition for what the transformation does since these embeddings generally live on a highly structured but unknown manifold. In our case, we operate on colors and we actually know the geometry of the colour spaces we use. This makes it easier for us to interpret the learned mapping (see Figure 2c and 2d that show convergence to a point in RGB and parallelism in HSV space).

## 5 Conclusion and Future Work

In this paper, we proposed novel models of predicting color based on textual modifiers, incorporating a matrix transformation than the previous largely linear additive method. As well as our more general approach, we exploit the properties of another color space, namely HSV, in which the modifier behaviours are often simpler. Overall our method leads to state of the art performance on a standard dataset.

In future work, we intend to develop more accurate modifier representations to allow for better generalisation to unseen modifiers. This might be achieved by using a composition sub-word representation for modifiers, such as character-level encoding. Finally, we also strive to acquire larger datasets. This is a crucial step towards comparing the generalization performance of different color-modifier models. Models trained on larger data sets are likely to be more applicable to real-world problems since they learn representations for more color terms.

## References

M.K. Agoston. 2005. *Computer Graphics and Geometric Modelling*, pages 300–306. Springer.

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1193. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Allan Hanbury. 2008. Constructing cylindrical coordinate colour spaces. *Pattern Recognition Letters*, 29(4):494 – 500.

George H. Joblove and Donald Greenberg. 1978. Color spaces for computer graphics. *SIGGRAPH Comput. Graph.*, 12(3):20–25.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

M Ronnier Luo, Guihua Cui, and Bryan Rigg. 2001. The development of the cie 2000 colour-difference formula: Ciede2000. *Color Research & Application: Endorsed by Inter-Society Color Council, The Colour Group (Great Britain), Canadian Society for Color, Color Science Association of Japan, Dutch Society for the Study of Color, The Swedish Colour Centre Foundation, Colour Society of Australia, Centre Français de la Couleur*, 26(5):340–350.

K McLaren. 1976. Xiiithe development of the cie 1976 (l* a* b*) uniform colour space and colour-difference formula. *Journal of the Society of Dyers and Colourists*, 92(9):338–341.

Brian McMahan and Matthew Stone. 2015. A Bayesian model of grounded color semantics. *Transactions of the Association for Computational Linguistics*, 3:103–115.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Will Monroe, Robert XD Hawkins, Noah D Goodman, and Christopher Potts. 2017. Colors in context: A pragmatic neural model for grounded language understanding. *Transactions of the Association for Computational Linguistics*, 5:325–338.

Nikola Mrkšić, Diarmuid O Séaghdha, Blaise Thomson, Milica Gašić, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. *arXiv preprint arXiv:1603.00892*.

Randall Munroe. 2010. Color survey results.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

K. van de Sande, T. Gevers, and C. Snoek. 2010. Evaluating color descriptors for object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1582–1596.

Olivia Winn and Smaranda Muresan. 2018. 'lighter' can still be dark: Modeling comparative color descriptions. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 790–795, Melbourne, Australia. Association for Computational Linguistics.

# Robust Navigation with Language Pretraining and Stochastic Sampling

**Xiujun Li**[♠◇] **Chunyuan Li**[◇] **Qiaolin Xia**[♣] **Yonatan Bisk**[♠◇♡]
**Asli Celikyilmaz**[◇] **Jianfeng Gao**[◇] **Noah A. Smith**[♠♡] **Yejin Choi**[♠♡]
[♠]Paul G. Allen School of Computer Science & Engineering, University of Washington
[♣]Peking University [◇]Microsoft Research AI [♡]Allen Institute for Artificial Intelligence
{xiujun,ybisk,nasmith,yejin}@cs.washington.edu
xql@pku.edu.cn {xiul,chunyl,jfgao}@microsoft.com

## Abstract

Core to the vision-and-language navigation (VLN) challenge is building robust instruction representations and action decoding schemes, which can generalize well to previously unseen instructions and environments. In this paper, we report two simple but highly effective methods to address these challenges and lead to a new state-of-the-art performance. First, we adapt large-scale pretrained language models to learn text representations that generalize better to previously unseen instructions. Second, we propose a stochastic sampling scheme to reduce the considerable gap between the expert actions in training and sampled actions in test, so that the agent can learn to correct its own mistakes during long sequential action decoding. Combining the two techniques, we achieve a new state of the art on the Room-to-Room benchmark with 6% absolute gain over the previous best result (47% → 53%) on the *Success Rate weighted by Path Length* metric.

## 1 Introduction

The vision-and-language navigation (VLN) task, learning to navigate in visual environments based on natural language instructions, has attracted interest throughout the artificial intelligence research community (Hemachandra et al., 2015; Anderson et al., 2018; Chen et al., 2019; Savva et al., 2019). It fosters research on multimodal representations and reinforcement learning, and serves as a test bed for many real-world applications such as in-home robots.

In the recent Room-to-Room (R2R) VLN challenge (Anderson et al., 2018), most state-of-the-art methods are developed based on an encoder-decoder framework (Cho et al., 2014; Sutskever et al., 2014), where a natural language instruction is represented as a sequence of words, and a navigation trajectory as a sequence of actions, enhanced with attention (Anderson et al., 2018; Wang et al., 2019; Fried et al., 2018; Ma et al., 2019a). Two important components are shared by all VLN agents: (i) an *Instruction Encoder* that employs a language model (LM) for instruction understanding; and (ii) an *Action Decoder*, where an appropriate sequence-level training scheme is required for sequential decision-making. Each component faces its own challenges (see Figure 1).

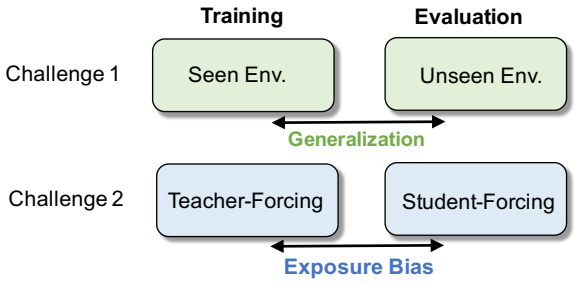


Figure 1: Two challenges in VLN.

The first challenge is generalizing grounded natural language instruction understanding from seen to unseen environments. Specifically, in the R2R task, only 69% of bigrams are shared between training and evaluation.[1] Existing work leverages pretrained GloVe embeddings (Pennington et al., 2014) to help generalize. In computer vision, it has been shown that large-scale models pretrained on ImageNet can transfer the knowledge to downstream applications (Yosinski et al., 2014), thus improving generalization. Comparable language-based transfer learning has not been shown for instruction understanding in VLN.

The second challenge is *exposure bias* (Ranzato et al., 2016) for the action decoder, due to the discrepancy between training and inference. This problem is common to many tasks where decoding is needed, including text generation, abstractive summarization, and machine translation (Ben-

[1]Table 1 shows $n$-gram overlap statistics between training seen and validation seen/unseen environments.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1494–1499, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| | n-gram(s) | Validation Seen | Validation Unseen |
|---|---|---|---|
| Training | 1 | 87.2% | 80.7% |
| | 2 | 77.4% | 68.9% |
| | 3 | 65.6% | 57.3% |
| | 4 | 50.8% | 44.4% |

Table 1: N-grams instruction overlap statistics between validation seen and unseen environments.

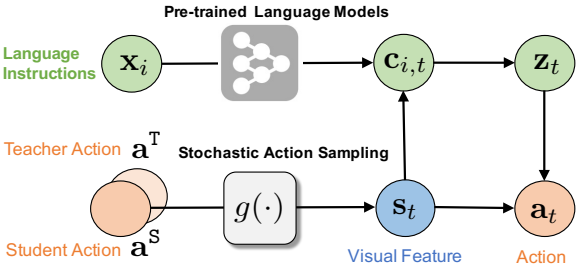


Figure 2: Illustration of proposed methods.

gio et al., 2015). Two widely used training strategies are *student-forcing* and *teacher-forcing* (described in detail in Section 2.2). It is well-known that the sequence length determines which training strategy is more effective. In the VLN literature, student-forcing has been widely used, as early work (Anderson et al., 2018) used long trajectories (up to 20 steps) with a simple discrete action space. Most recent work, however, has relied on a panoramic action space (Fried et al., 2018) in which most trajectories are only up to seven steps long. In such cases, teacher-forcing is preferable (Tan et al., 2019). Neither strategy is perfect: teacher-forcing has exposure bias, while student-forcing's random actions can cause an agent to deviate far from the correct path, rendering the original instruction invalid.[2]

To tackle these challenges, we have developed two techniques to enable the agent to navigate more efficiently. For the first challenge, we leverage the recent large-scale pretrained language models, BERT (Devlin et al., 2019) and GPT (Radford et al., 2018), to improve the agent's robustness in unseen environments. We show that large-scale *language-only* pretraining improves generalization in grounded environments. For the second challenge, we propose a stochastic sampling scheme to balance teacher-forcing and student-forcing during training, so that the agent can recover from its own mistakes at inference time. As a result of combining both techniques, on the R2R benchmark test set, our agent (PRESS)[3] achieves 53% on SPL, an absolute 6% gain over the current state of the art.

## 2 Method

In the VLN task, instructions are represented as a set $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^{M}$ of $M$ instructions per trajectory. Each instruction $\boldsymbol{x}_i$ is a sequence of $L_i$ words, $\boldsymbol{x}_i = [x_{i,1}, x_{i,2}, ..., x_{i,L_i}]$. Given $\mathcal{X}$, the goal is to train an agent to navigate from a starting position $\boldsymbol{s}_0$ to a target position, via completing a $T$-step trajectory $\tau = [\boldsymbol{s}_0, \boldsymbol{a}_0, \boldsymbol{s}_1, \boldsymbol{a}_1, \cdots, \boldsymbol{s}_T, \boldsymbol{a}_T]$, where $\boldsymbol{s}_t$ and $\boldsymbol{a}_t$ are the visual state and navigation action, respectively, at step $t$. The training dataset $\mathcal{D}_E = \{\boldsymbol{\tau}, \mathcal{X}\}$ consists of example pairs of instruction set $\mathcal{X}$ and a corresponding expert trajectory $\boldsymbol{\tau}$. Our goal is to learn a policy $\pi_{\boldsymbol{\theta}}(\boldsymbol{\tau}|\mathcal{X})$ that maximizes the log-likelihood of the target trajectory $\boldsymbol{\tau}$ given instructions $\mathcal{X}$:

$$\log \pi_{\boldsymbol{\theta}}(\boldsymbol{\tau}|\mathcal{X}) = \sum_{t=1}^{T} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t|\boldsymbol{s}_t, \mathcal{X}), \quad (1)$$

where $\boldsymbol{\theta}$ are trainable parameters. The policy is usually parameterized as an attention-based seq2seq model, with a language encoder $\boldsymbol{z}_t = f_{\theta_E}(\boldsymbol{x})$, and an action decoder $\boldsymbol{a}_t = f_{\theta_D}(\boldsymbol{z}_t, \boldsymbol{s}_t)$. Successful navigation depends on $(i)$ precisely grounding the instructions $\mathcal{X}$ in $\tau$ in various environments, and $(ii)$ correctly making the current decision $\boldsymbol{a}_t$ based on previous actions/observations $\tau_{<t} = [\boldsymbol{s}_0, \boldsymbol{a}_0, \cdots, \boldsymbol{s}_{t-1}]$. To address these concerns, we propose PRESS, illustrated in Figure 2.

### 2.1 Instruction Understanding with Pretrained Language Models

At each step $t$, the agent decides where to navigate by updating a dynamic understanding of the instructions $\boldsymbol{z}_t$, according to its current visual state $\boldsymbol{s}_t$. Given instruction $\boldsymbol{x}$, the language encoder proceeds in two steps, end-to-end, by considering a function decomposition $f_{\theta_E} = f_{\boldsymbol{\theta}_{x\to e}} \circ f_{\boldsymbol{\theta}_{e\to z}}$:

- $f_{\boldsymbol{\theta}_{x\to e}} : \boldsymbol{x} \to \boldsymbol{e}$, where $\boldsymbol{x} = [x_1, \cdots, x_L]$ is represented as its (contextualized) word embedding form $\boldsymbol{e} = [e_1, \cdots, e_L]$, with $e_i$ as the representation for word $x_i$;
- $f_{\boldsymbol{\theta}_{e\to z}} : \boldsymbol{e} \to \boldsymbol{z}_t$: For each embedded instruction $\boldsymbol{e}$, we ground its representations as $\boldsymbol{c}_{i,t}$ for state $\boldsymbol{s}_t$ via neural attention. To handle

[2] To compensate, beam search is often used to improve success rates. Recent work, e.g., using search strategies (Ke et al., 2019) or progress monitors (Ma et al., 2019b), has focused on mitigating the cost of computing top-$k$ rollouts.

[3] **PRE**TRAINED LMS AND **S**TOCHASTIC **S**AMPLING

language variability, one may aggregate features of multiple instructions $\mathcal{C}_t = \{\boldsymbol{c}_{i,t}\}_{i=1}^{M}$ into a single joint feature $\boldsymbol{z}_t = \frac{1}{M}\sum_{i=1}^{M}\boldsymbol{c}_{i,t}$.[4]

Previous methods in VLN learn $\boldsymbol{e}$ either from pretrained word embeddings (Pennington et al., 2014) which do not take into account word context, or from scratch. As a result, their representations do not capture contextual information within each instruction. More importantly, they tend to overfit the training instructions associated with seen environments, limiting their utility in unseen environments. To remedy these issues, we propose to represent $\boldsymbol{e}$ with contextualized word embeddings produced using large-scale pretrained language models, such as BERT and GPT.

**Instruction Encoder.** The agent's memory vector $\boldsymbol{h}_{t-1}$ captures the perception and action history and is used to attend to the instruction $\boldsymbol{x}$. A pretrained LM $f_{\boldsymbol{\theta}_{x\to e}}$ encodes the instruction $e = [\boldsymbol{e}_1, \cdots, \boldsymbol{e}_L]$; $\boldsymbol{e}_i$ where the representation for word $x_i$, is built with $f_{\boldsymbol{\theta}_{x\to e}} \in \{$ GPT, BERT $\}$, and $\boldsymbol{\theta}_{x\to e}$ are fine-tuned parameters. The embedded words $e = [\boldsymbol{e}_1, \cdots, \boldsymbol{e}_L]$ are passed through an LSTM $f_{\boldsymbol{\theta}_{e\to z}}$ to produce a sequence of textual features $[\boldsymbol{h}_1^e, \cdots, \boldsymbol{h}_L^e]$. At each time step $t$, the *textual context* for the instruction $\boldsymbol{x}$ is computed as weighted sum of textual features in the sequence:

$$\boldsymbol{c}_{i,t} = \sum_{l=1}^{L} \alpha_l \boldsymbol{h}_l^e \tag{2}$$

where $\alpha_l = \text{Softmax}(\boldsymbol{h}_t^\top \boldsymbol{h}_l^e)$, $\alpha_l$ places more weight on the word representations that are most relevant to the agent's current status.

**Decoder.** At each step, the agent takes an action $a_t$, and the environment returns new visual observations; the agent first performs one-hop visual attention $f(\cdot)$ to all the visual image features $s_t$, based on its previous memory vector $\boldsymbol{h}_{t-1}$. Then, the agent updates its visual state $\boldsymbol{s}_t$ as the weighted sum of the panoramic features, $\boldsymbol{s}_t = \sum_j \gamma_{t,j}\boldsymbol{s}_{t,j}$. The attention weight $\gamma_{t,j}$ for the $j$-th visual feature $\boldsymbol{s}_{t,j}$ represents its importance with respect to the previous history context $\boldsymbol{h}_{t-1}$, computed as $\gamma_{t,j} = \text{Softmax}((\mathbf{W}_h\boldsymbol{h}_{t-1})^\top \mathbf{W}_s\boldsymbol{s}_{t,j})$ (Fried et al., 2018) where $\text{Softmax}(r_j) = \exp(r_j)/\sum_{j'}\exp(r_{j'})$, $\mathbf{W}_h$ and $\mathbf{W}_s$ are trainable projection matrices.

$$\boldsymbol{h}_t = f_{\boldsymbol{\theta}_D}([\boldsymbol{s}_t, \boldsymbol{a}_{t-1}], \boldsymbol{h}_{t-1}) \tag{3}$$

where $\boldsymbol{a}_{t-1}$ is the action taken at previous step, and $\boldsymbol{\theta}_D$ are the LSTM decoder parameters.

**Two-stage learning.** The parameters of our agent are $\boldsymbol{\theta} = \{\boldsymbol{\theta}_{x\to e}, \boldsymbol{\theta}_{e\to z}, \theta_D\}$. In practice, we find that the agent overfits quickly, when the full model is naively fine-tuned, with $\boldsymbol{\theta}_{x\to e}$ initialized by pretrained LMs (*e.g.,* BERT). In this paper, we consider a two-stage learning scheme to facilitate the use of pretrained LMs for VLN. $(i)$ Embedding-based stage: We fix $\boldsymbol{\theta}_{x\to e}$, and use BERT or GPT to provide instruction embeddings. Only $\{\boldsymbol{\theta}_{e\to z}, \boldsymbol{\theta}_D\}$ are updated (while tuning on validation). $(ii)$ Fine-tuning stage: We train all model parameters $\boldsymbol{\theta}$ with a smaller learning rate, so that $\boldsymbol{\theta}_{x\to e}$ can adapt to our VLN task.

### 2.2 Stochastic Action Sampling

A core question is how to learn useful state representations $\boldsymbol{s}_t$ in Eq. (1) during the trajectory rollout. In other words, which action should we use to interact with the environment to elicit the next state? As noted, most existing work uses one of two schemes: $(i)$ *Teacher-forcing (TF)*, where the agent takes ground-truth actions $\boldsymbol{a}^{\text{T}}$ only. Though TF enables efficient training, it results in "exposure bias" because agents must follow learned rather than gold trajectories at test time. In contrast, $(ii)$ *Student-forcing (SF)*, where an action $\boldsymbol{a}^{\text{S}}$ is drawn from the current learned policy, allows the agent to learn from its own actions (aligning training and evaluation), however, it is inefficient, as the agent explores randomly when confused or in the early stages of training.

In this work, we consider a stochastic scheme (*SS*) to alternate between choosing actions from $\boldsymbol{a}^{\text{T}}$ and $\boldsymbol{a}^{\text{S}}$ for state transition $\boldsymbol{s} \leftarrow g(\boldsymbol{a}^{\text{T}}, \boldsymbol{a}^{\text{S}})$, inspired by scheduled sampling (Bengio et al., 2015). As illustrated in Figure 2, at each step, the agent "flips a coin" with some probability $\epsilon$ to decide whether to take the teacher's action $a^{\text{T}}$ or a sampled one $a^{\text{S}}$:

$$\boldsymbol{a} = \delta\boldsymbol{a}^{\text{T}} + (1-\delta)\boldsymbol{a}^{\text{S}}, \tag{4}$$

where $\delta \sim \text{Bernoulli}(\epsilon)$. This allows the agent to leverage the advantages of both TF and SF, yielding a faster and less biased learner. We fix $\epsilon$ as a constant during learning, which is different from the decaying schedule in (Bengio et al., 2015).

[4] This recovers $\boldsymbol{z}_t = \boldsymbol{c}_t$ when only a single instruction is available.

## 3 Experiments

### 3.1 Dataset

We use the Room-to-Room dataset for the VLN task, built upon the Matterport3D dataset (Chang et al., 2017), which consists of 10,800 panoramic views and 7,189 trajectories. Each trajectory is paired with three natural language instructions. The R2R dataset consists of four splits: train seen, validation seen, validation unseen, and test unseen. There is no overlap between seen and unseen environments. At the beginning of each episode, the agent starts at a specific location, and is given natural instructions, the goal of the agent is to navigate to the target location as quickly as possible.

### 3.2 Baseline Systems

We compare our approach with eight recently published systems:

- RANDOM: an agent that randomly selects a direction and moves five step in that direction (Anderson et al., 2018).
- SEQ2SEQ: sequence-to-sequence model proposed by Anderson *et al.* as a baseline for the R2R benchmark (Anderson et al., 2018) and analyzed in (Thomason et al., 2019).
- RPA (Wang et al., 2018): is an agent which combines model-free and model-based reinforcement learning, using a look-ahead module for planning.
- SPEAKER-FOLLOWER (Fried et al., 2018): an agent trained with data augmentation from a speaker model with panoramic actions.
- SMNA (Ma et al., 2019a): an agent trained with a visual-textual co-grounding module and progress monitor on panoramic actions.
- RCM+SIL(TRAIN) (Wang et al., 2019): an agent trained with cross-modal grounding locally and globally via reinforcement learning.
- REGRETFUL (Ma et al., 2019b): an agent with a trained progress monitor heuristic for search that enables backtracking.
- FAST (Ke et al., 2019): an agent which combines global and local knowledge to compare partial trajectories of different lengths, enabling efficient backtrack after a mistake.
- ENVDROP (Tan et al., 2019): proposed an environment dropout method, which can generate more environments based on the limited seen environments.

| Setting | Agent | Validation Seen SR ↑ | Validation Seen SPL ↑ | Validation Unseen SR ↑ | Validation Unseen SPL ↑ |
|---|---|---|---|---|---|
| S | seq2seq | 51 | 46 | 32 | 25 |
| | PRESS | 47 (-4) | 43 (-3) | 43 (+11) | 38 (+13) |
| M | seq2seq | 49 | 44 | 33 | 26 |
| | PRESS | 56 (+7) | 53 (+9) | 56 (+23) | 50 (+24) |

Table 2: Comparison of PRESS and seq2seq.

### 3.3 Evaluation Metrics

We benchmark our agent on the following metrics:

**TL** **Trajectory Length** measures the average length of the navigation trajectory.

**NE** **Navigation Error** is the mean of the shortest path distance in meters between the agent's final location and the target location.

**SR** **Success Rate** with which the agent's final location is less than 3 meters from the target.

**SPL** **Success weighted by Path Length** trades-off SR against TL.

SPL is the recommended primary metric, other metrics are considered as auxiliary measures.

### 3.4 Implementation

We use a LSTM/GPT/BERT for the language encoder, and a second single-layer LSTM for the action decoder (h=1024). We use Adamax and batch sizes of 24/16 for pretraining/finetuning. The learning rates for MLE are $1e^{-4}$, during finetuning BERT the learning rate is $5e^{-5}$. Following (Fried et al., 2018), we use a panoramic action space and the ResNet image features provided by (Anderson et al., 2018). The code is publicly available here: https://github.com/xjli/r2r_vln.

### 3.5 Results

**Robust Generalization.** First, we compare PRESS to a baseline seq2seq model[5] in two evaluation settings on the validation splits: (1) **S**: A single instruction is provided to the agent at a time. Thus, three separate navigation trajectories are generated corresponding to three alternative instructions in this setting. We report the averaged performance over three separate runs. (2) **M**: All three instructions are provided to the agent at once. The seq2seq baseline does not have an aggregation strategy so we report its performance for the single trajectory with maximum likelihood. For PRESS, we aggregate the instructions via context

[5]The baseline seq2seq agent is the FOLLOWER of SPEAKER-FOLLOWER (Fried et al., 2018).

| | Model | Validation Seen TL ↓ | NE ↓ | SR ↑ | SPL ↑ | Validation Unseen TL ↓ | NE ↓ | SR ↑ | SPL ↑ | Test Unseen TL ↓ | NE ↓ | SR ↑ | SPL ↑ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Greedy | RANDOM | 9.58 | 9.45 | 16 | - | 9.77 | 9.23 | 16 | - | 9.93 | 9.77 | 13 | 12 |
| | SEQ2SEQ | 11.33 | 6.01 | 39 | - | 8.39 | 7.81 | 22 | - | 8.13 | 7.85 | 20 | 18 |
| | RPA | - | 5.56 | 43 | - | - | 7.65 | 25 | - | 9.15 | 7.53 | 25 | 23 |
| | SPEAKER-FOLLOWER | - | 3.36 | 66 | - | - | 6.62 | 35 | - | 14.82 | 6.62 | 35 | 28 |
| | SMNA | - | - | - | - | - | - | - | - | 18.04 | 5.67 | 48 | 35 |
| | RCM+SIL(TRAIN) | 10.65 | 3.53 | 67 | - | 11.46 | 6.09 | 43 | - | 11.97 | 6.12 | 43 | 38 |
| | REGRETFUL | - | 3.23 | 69 | 63 | - | 5.32 | 50 | 41 | 13.69 | 5.69 | 48 | 40 |
| | FAST | - | - | - | - | 21.17 | 4.97 | 56 | 43 | 22.08 | 5.14 | 54 | 41 |
| | ENVDROP | 11.00 | 3.99 | 62 | 59 | 10.70 | 5.22 | 52 | 48 | 11.66 | 5.23 | 51 | 47 |
| | PRESS | 10.35 | **3.09** | **71** | **67** | 10.06 | **4.31** | **59** | **55** | 10.52 | **4.53** | **57** | **53** |
| | Human | - | - | - | - | - | - | - | - | 11.85 | 1.61 | 86 | 76 |

Table 3: Comparison with the state-of-the-art methods. **Blue** indicates best value overall.

mean-pooling and generate a single trajectory. No data augmentation is applied to either model.

The results are summarized in Table 2. (*i*) PRESS drastically outperforms the seq2seq models on unseen environments in both settings, and (*ii*) Interestingly, our method shows a much smaller gap between seen and unseen environments than seq2seq. It demonstrates the importance of pretrained LMs and stochastic sampling for strong generalization in unseen environments.

**Comparison with SoTA.** In Table 3, we compare the performance of our agent against all the published methods, our PRESS agent outperforms the existing models on nearly all the metrics.

**Ablation Analysis.** Key to this work is leveraging large-scale pretrained LMs and effective training strategies for action sequence decoding. Table 4 shows an ablation of these choices. (1) BERT and GPT are better than LSTM on both seen and unseen environments, and BERT generalizes better than GPT on unseen environments. (2) Teacher-forcing performs better than student-forcing on validation unseen environments, while an opposite conclusion is drawn on validation seen environments. SS performs the best on unseen environments.

**Qualitative Examples.** We provide two navigation examples of PRESS on the validation unseen environments with the step-by-step views and top-down views in Appendix.

(1) Figure 3 shows how the agent with LSTM instruction encoder performs compared with our PRESS agent. There are two rare words "**mannequins**" and "**manikins**" which are not in the training dataset and confuse the LSTM agent, while, PRESS successfully maps these two "mannequins" and "manikins" to the correct objects.

| LM | | Validation Seen TL | NE | SR | SPL | Validation Unseen TL | NE | SR | SPL |
|---|---|---|---|---|---|---|---|---|---|
| LSTM | TF | 10.50 | 5.74 | 44 | 42 | 9.86 | 6.23 | 42 | 39 |
| | SF | 11.87 | 3.97 | 59 | 53 | 13.23 | 6.17 | 40 | 31 |
| | SS | 10.99 | 3.46 | 64 | 59 | 10.73 | 4.89 | 53 | 48 |
| GPT | TF | 10.03 | 4.05 | 60 | 58 | 9.43 | 3.36 | 49 | 46 |
| | SF | 11.46 | 2.53 | 73 | 67 | 13.13 | 5.13 | 49 | 41 |
| | SS | 10.60 | 2.99 | 71 | 68 | 10.79 | 3.05 | 56 | 51 |
| BERT | TF | 10.57 | 4.06 | 59 | 56 | 9.61 | 5.13 | 51 | 47 |
| | SF | 12.39 | 2.71 | 73 | 64 | 13.12 | 5.06 | 51 | 42 |
| | SS | 10.35 | 3.09 | 71 | 67 | 10.06 | 4.31 | **59** | **55** |

Table 4: Ablation results of different language pretrainings and training strategies: Teacher Forcing (TF), Student Forcing (SF) and Stochastic Sampling (SS).

(2) The second set in Figure 4 shows how the agents trained with different training strategies performs in an unseen environment. The agents trained with teacher-forcing and student-forcing both fail, while PRESS succeeds.

## 4 Conclusion

We present PRESS, a navigation agent based on two previously underexplored techniques in VLN: pretrained language models and stochastic action sampling. Our PRESS demonstrates robust generalization in the unseen environments, leading to a new state-of-the-art performance over many of the much more complex approaches previously proposed. As both the components of PRESS can be easily integrated, future models can consider building upon them as a strong baseline system.

## Acknowledgments

We thank the anonymous reviewers for their insightful comments, NSF IIS-1703166, DARPA's CwC program through ARO W911NF-15-1-0543, and the Allen Institute for Artificial Intelligence.

## References


Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. 2018. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Neural Information Processing Systems*.

Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Nießner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. 2017. Matterport3D: Learning from RGB-D data in indoor environments. In *International Conference on 3D Vision*.

Howard Chen, Alane Shur, Dipendra Misra, Noah Snavely, and Yoav Artzi. 2019. Touchdown: Natural language navigation and spatial reasoning in visual street environments. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. 2018. Speaker-follower models for vision-and-language navigation. In *Neural Information Processing Systems*.

Sachithra Hemachandra, Felix Duvallet, Thomas M Howard, Nicholas Roy, Anthony Stentz, and Matthew R Walter. 2015. Learning models for following natural language directions in unknown environments. In *IEEE International Conference on Robotics and Automation*.

Liyiming Ke, Xiujun Li, Yonatan Bisk, Ari Holtzman, Zhe Gan, Jingjing Liu, Jianfeng Gao, Yejin Choi, and Siddhartha Srinivasa. 2019. Tactical rewind: Self-correction via backtracking in vision-and-language navigation. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Chih-Yao Ma, Jiasen Lu, Zuxuan Wu, Ghassan AlRegib, Zsolt Kira, Richard Socher, and Caiming Xiong. 2019a. Self-monitoring navigation agent via auxiliary progress estimation. In *International Conference on Learning Representations*.

Chih-Yao Ma, Zuxuan Wu, Ghassan AlRegib, Caiming Xiong, and Zsolt Kira. 2019b. The regretful agent: Heuristic-aided navigation through progress estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Conference on empirical methods in natural language processing*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *International Conference on Learning Representations*.

Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. 2019. Habitat: A platform for embodied ai research. In *International Conference on Computer Vision*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Neural Information Processing Systems*.

Hao Tan, Licheng Yu, and Mohit Bansal. 2019. Learning to navigate unseen environments: Back translation with environmental dropout. In *the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Jesse Thomason, Daniel Gordon, and Yonatan Bisk. 2019. Shifting the Baseline: Single Modality Performance on Visual Navigation & QA. In *the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Xin Wang, Qiuyuan Huang, Asli Celikyilmaz, Jianfeng Gao, Dinghan Shen, Yuan-Fang Wang, William Yang Wang, and Lei Zhang. 2019. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Xin Wang, Wenhan Xiong, Hongmin Wang, and William Yang Wang. 2018. Look before you leap: Bridging model-free and model-based reinforcement learning for planned-ahead vision-and-language navigation. In *IEEE European Conference on Computer Vision*.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? In *Neural Information Processing Systems*.

# Towards Making a Dependency Parser See

**Michalina Strzyz** **David Vilares** **Carlos Gómez-Rodríguez**
Universidade da Coruña, CITIC
FASTPARSE Lab, LyS Research Group, Departamento de Computación
Campus de Elviña, s/n, 15071 A Coruña, Spain
{michalina.strzyz,david.vilares,carlos.gomez}@udc.es

## Abstract

We explore whether it is possible to leverage eye-tracking data in an RNN dependency parser (for English) when such information is only available during training – i.e. no aggregated or token-level gaze features are used at inference time. To do so, we train a multi-task learning model that parses sentences as sequence labeling and leverages gaze features as auxiliary tasks. Our method also learns to train from disjoint datasets, i.e. it can be used to test whether already collected gaze features are useful to improve the performance on new *non-gazed* annotated treebanks. Accuracy gains are modest but positive, showing the feasibility of the approach. It can serve as a first step towards architectures that can better leverage eye-tracking data or other complementary information available only for training sentences, possibly leading to improvements in syntactic parsing.

## 1 Introduction

Eye trackers and gaze features collected from them have been recently applied to natural language processing (NLP) tasks, such as part-of-speech tagging (Duffy et al., 1988; Nilsson and Nivre, 2009; Barrett and Søgaard, 2015a), named-entity recognition (Tokunaga et al., 2017) or readability (González-Garduño and Søgaard, 2018). Eye-movement data has been also used for parsing. For example, Barrett and Søgaard (2015b) rank discriminative features to predict syntactic categories (e.g. subject vs. object) and use them to improve a transition-based parser, trained on a structured perceptron with discrete features (Collins, 2002; Zhang and Nivre, 2011). However, the experiments were carried out on a parallel toy treebank and the performance was relatively low. Lopopolo et al. (2019) follow the inverse path, and use dependency parsing features to predict eye-regression during training, i.e. cases where the reader goes back to a word of the sentence.

In this context, how to retrieve and leverage eye-tracking data has become an active area of research in different NLP fields. Previous studies (Barrett and Søgaard, 2015b; Barrett et al., 2016) suggest that real-time eye-tracking data can be collected at inference time, so that *token-level* gaze features are used during training but also at test time. However, even if in the near future every user has an eye tracker on top of their screen – a scenario which is far from guaranteed, and raises privacy concerns (Liebling and Preibusch, 2014) – many running NLP applications that process data from various Internet sources will not expect to have any human being reading massive amounts of data. Other studies (Barrett and Søgaard, 2015a; Hollenstein and Zhang, 2019) instead derive gaze features from the training set: in particular, they collect *type-level* gaze features from the vocabulary in the training set; which are then aggregated to create a lookup table and used as a sort of pre-computed gaze input when a given word in the test set matches an entry, otherwise, a token has an unknown gaze feature value. In this manner, the influence of gaze features on unseen data depends on the vocabulary encountered during training. For instance, Hollenstein et al. (2019) report that unknown tokens make up $41.09\%$ on new data for their research on named-entity recognition (NER) with eye-tracking data. To our knowledge, this kind of approaches have not been applied to syntactic parsing.

More in the line of our work, Barrett et al. (2018) is one of the few approaches that does not rely on the assumption of using gaze features as input. Instead, the human data is used as an inductive bias to guide the attention weights in a recurrent neural network for sequence classification (used for tasks such as binary sentiment anal-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1500–1506, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

ysis a the sentence level). Moreover, it has been shown that both constituency (Vilares et al., 2019) and dependency (Strzyz et al., 2019a) parsing can benefit from multi-task learning. Also related to our work with human data, González-Garduño and Søgaard (2017) show that gaze features learned in a multi-task (MTL) setup can lead to improvements in readability assessment.

**Contribution** In this work we leverage gaze learning for dependency parsing assuming that gaze features are likely to be only available during training – and no gaze features are given, in any form, at inference time. Our approach learns eye-movement features as auxiliary tasks in a multitask framework where both parsing and gaze prediction are addressed as sequence labeling. To test its effect on syntactic analysis, we experiment on parallel and disjoint datasets with dependency parsing and gaze annotations. The source code can be found at `https://github.com/mstrise/dep2label-eye-tracking-data`.

## 2 Methods and materials

This section describes in detail the data and the models used in our work.

### 2.1 Data

We will use both *parallel* data (containing dependency and gaze annotations) and *disjoint* data (where one dataset contains only parsing annotations and the other one just gaze data).

***Parallel* data** The Dundee corpus (Kennedy et al., 2003) contains recordings of measurements for eye movements of ten English-speaking participants during reading 20 newspaper articles from *The Independent* making in total 2368 unique sentences. This dataset was used as the starting point to create a corpus that fits the purpose of dependency parsing, the Dundee treebank (Barrett et al., 2015). This treebank augments sentences from the Dundee corpus with syntactic annotations according to the Universal Dependencies (UD) guidelines (Nivre et al., 2016). We will use this treebank as the parallel data for our models. We split the data $(80-10-10)$ into training, development and test. Sentences were randomly shuffled with assurance that the same sentence coming from the 10 participants is included only in one of the sets.[1]

***Disjoint* data** Most of dependency treebanks do not contain gaze annotations. With this setup, in addition to the assumption that eye-movement data is available during training, we aim to show whether we improve the performance of a dependency parser with gaze-annotated data coming from a different corpus. We will use two datasets: one with dependency annotations and another one labeled with eye-movement data. For the former we use the Penn Treebank (PTB) (Marcus et al., 1993), which contains sentences from *The Wall Street Journal* annotated with phrase-structure trees. We convert it into Stanford Dependencies (de Marneffe et al., 2006) and apply the standard splits for dependency parsing: sections $2-21$ for training, 22 for development set and 23 for testing, whereas PoS tags are predicted by the Stanford tagger (Toutanova et al., 2003). For gaze-annotated data, we will employ the eye-movement annotations from the Dundee Treebank but using a different split $(90-10-0)$.[2]

### 2.2 Gaze-averaged sequence labeling parsing

During training we opt to emulate recent cognitive studies that suggest that human sentence understanding recruits the same brain regions for lexical and syntactic processing, consistent with a sequence-tagging-like process (Fedorenko et al., 2018), which is also a natural and intuitive way to learn to predict eye-movement information. To do so, we rely on standard bidirectional long short-term memory networks (BILSTMs) (Hochreiter and Schmidhuber, 1997; Schuster and Paliwal, 1997).

We denote $\text{LSTM}_\theta(\mathbf{w})$ as the abstraction of a long short-term memory network, that processes an input sentence $w$=$[w_1, ..., w_n]$ to generate an output of hidden representations $\mathbf{h}$=$[\mathbf{h}_1, ..., \mathbf{h}_n]$. Thus, a BILSTM can be seen as $\text{BILSTM}_\theta(\mathbf{w})$ = $\text{LSTM}^l_\theta(\mathbf{w_{i:m}}) \cdot \text{LSTM}^r_\theta(\mathbf{w_{m:n}}) = \mathbf{h}^l \cdot \mathbf{h}^r$.[3] In this work we will stack two layers of BILSTMs before decoding the output. BILSTMs are commonly used for sequence labeling tasks (Reimers

[1] In addition we removed 40 sentences (4 unique sentences read by the 10 participants) that contained cases of incoherence in their syntactic dependency annotation resulting in 23640 sentences in total (2364 unique sentences).

[2] During testing we will be just testing the performance on the PTB test set.

[3] $\text{LSTM}^l$ processes the sentences from left-to-right and $\text{LSTM}^r$ from right-to-left. '$\cdot$' is the concatenation operator.

and Gurevych, 2017), where a word-level prediction can be generated for each token, by adding a feed forward network to predict an output label at each time step, using a $softmax(\mathbf{W} \times \mathbf{h}_i + \mathbf{b})$. In this paper we propose to learn to leverage eye-movement data as a sequence labeling task. To jointly learn dependency parsing in a common framework, we also cast parsing as sequence labeling and learn both tasks in a multitask learning (MTL) setup (Caruana, 1997). The main components are described in the following paragraphs.

**Dependency parsing as sequence labeling** We proceed similarly to Strzyz et al. (2019b). Given a linearization function $F_w : T_{|w|} \rightarrow L^w$, for each word $w_i$, Strzyz et al.'s approach generates a label $l_i \in (o_i, p_i, d_i)$ that encodes the binary relationship between $w_i$ and its head, where: $d_i$ encodes the dependency relation, and the sub-pair $(o_i, p_i)$ the index of such head term, with $o_i \in \mathbb{N}$ and $p_i \in P$ (a part-of-speech set). If $o_i$ is positive, then the head of $w_i$ is the $o_i$-th token to the right that has the part-of-speech tag $p_i$. If $o_i$ is negative, then the head of $w_i$ is the $|o_i|$-th token to the left whose PoS tag is $p_i$.

This parser obtains similar results to competitive transition- and graph-based parsers such as BIST (Kiperwasser and Goldberg, 2016) and can be taken as a strong baseline to test the effect of eye-movement data for dependency parsing.

**Gaze information** In previous studies the choice of number of gaze features used in the experiments has varied, seemingly, depending on the NLP task of interest. For instance, Barrett et al. (2016) distinguish 31 features (where 22 are gaze features) for part-of-speech tagging while Hollenstein and Zhang (2019) use 17 features in the NER task. In another piece of work, 5 gaze features are used for relation classification and sentiment analysis (Hollenstein et al., 2019). Finally, Singh et al. (2016) use gaze features in order to automatically predict reading times for a new text. However, the model predicts only 4 features that are then used as features for readability assessment.

We have chosen 12 gaze features and based on the previous work (Barrett et al., 2016; Hollenstein and Zhang, 2019) we have divided them into 4 groups. In particular, we explore the informativeness of the *basic gaze features*: total fixation duration on a word $w$ (`total fix dur`), mean fixation duration on $w$ (`mean fix dur`, number fixations on $w$ ($n$ `fix`) and fixation probability on $w$ (`fix prob`). As *early gaze features* we consider: first fixation duration on $w$ (`first fix dur`) and first pass duration (`first pass dur`), while as *late gaze features*: number of re-fixations on $w$ ($n$ `re-fix`) and reread probability of $w$ (`reread prob`). We also take account of the neighboring words and treat them as *context features*: fixation probability on the previous and next word ($w-1$ and $w+1$ `fix prob`) as well as fixation duration on the previous and next word ($w-1$ and $w+1$ `fix dur`). Figure 1 depicts an exemplary (linearized) dependency tree with some of the gaze features used in the experiments. Following previous work (Hollenstein and Zhang, 2019), we discretize gaze features.[4]

**Multitask learning** We previously argued that gaze-annotated data is unlikely to be available at inference time. However, previous related work for dependency parsing usually assumes that such information will be collected during testing and fed as input features to the model (Barrett and Søgaard, 2015b; Hollenstein and Zhang, 2019). What we propose instead is to leverage such information during training. To do so, we rely on MTL and auxiliary tasks. Our work focuses on exploiting the utility of gaze information using just a standard BILSTM, directly building on top of previous work of dependency parsing as sequence labeling (Strzyz et al., 2019b), and ignoring extra tools such as attention. In this line, a future possible solution could be to apply the approach by Barrett et al. (2018) to structured prediction and word-level classification. In their work they used human data as an inductive bias to update the attention weights of the network.

In our setup, dependency parsing as sequence labeling is addressed through two main tasks: one to predict the index of the head term, i.e. the sub-index $(o_i, p_i)$, and another one to predict the dependency relation ($d_i$). Eye-movement discrete labels are learned as auxiliary tasks. We use a hard-sharing architecture where the BILSTMs are fully shared and followed by an independent feed-forward layer (followed by a softmax) to predict the corresponding label for each of the tasks. The main idea is that the eye-movement signal(s) will

[4] `Total fix dur`, `mean fix dur`, `first fix dur`, `first pass dur`, $w-1$ and $w+1$ `fix dur` are used as a discrete variable whose values are represented as percentile intervals with a bin size of 20, while for the other features we use their raw values.

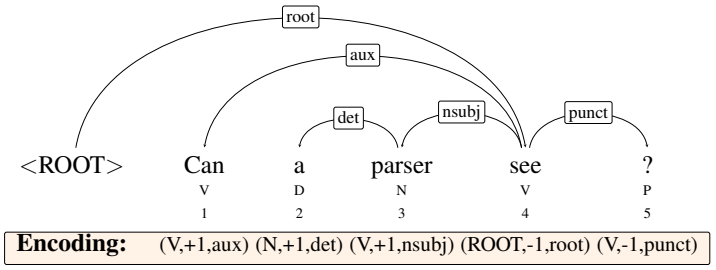


**Gaze features**:

| | | | | | |
|---|---|---|---|---|---|
| `mean fix` | 60-80 | 0-20 | 60-80 | 80-100 | 20-40 |
| `nb fix` | 2 | 0 | 3 | 5 | 1 |
| $w$ `fix prob` | 0 | 0 | 1 | 1 | 0 |
| $w+1$ `fix prob` | 0 | 0 | 1 | 1 | 0 |

Figure 1: A dependency tree with its encoding and some of the corresponding gaze features.

be back-propagated to update the weights of the shared BILSTM, building a model that latently encodes such information and helps dependency parsing.

For the parallel data setup, the model is trained in a standard way, and the cross-entropy loss is computed as $\mathcal{L} = \mathcal{L}_{(o,p)} + \mathcal{L}_d + \sum_a \beta_a * \mathcal{L}_a$. For the disjoint setup, during the training process for each batch we randomly pick all its samples from one of the treebanks (samples that have not been yet taken, either from the dependency parsing corpus or the gaze-annotated one) and run the network to make the predictions. Then, we back-propagate the loss for the outputs associated to that treebank and ignore the rest, i.e. $\mathcal{L} = \tau(\mathcal{L}_{(o,p)} + \mathcal{L}_d) + (1-\tau)\sum \beta_{aux}\mathcal{L}_{aux}$, where $\tau$ is a binary flag set to 1 when the batch contains dependency parsing samples, and to 0 otherwise.

For both setups we will test: (i) predicting *one* of the gaze features as an auxiliary task (ii) predicting multiple gaze labels as multiple auxiliary tasks.

## 3 Experiments

We conduct an explanatory study for the general viability of our method. We will compare results on the development and test set to verify whether the improvements are consistent. In particular, we carry out two experiments:

- Experiment 1 (on parallel data): Tests how eye-tracking data influences the performance when both gaze features and syntactic dependencies are extracted from the same data (in this case the Dundee treebank). The baseline is a model where no gaze features were used.
- Experiment 2 (on disjoint data): As previously said, we wish to test whether we can leverage existing gaze-annotated datasets to improve the performance in other non-gazed annotated dependency treebanks. Also, note that the Dundee treebank is relatively small (∼2k sentences) and results from the previous Experiment can be not as representative as those obtained in larger corpora. In this setup, we train models using gaze information from the Dundee treebank and dependency relations from PTB, evaluating on the latter.[5] The baseline is obtained by training a model only on the dependency relations retrieved from PTB.

**Metrics** We evaluate the parsers with Unlabeled and Labeled Attachment Score (UAS and LAS) excluding punctuation (following the standard methodology for experiments on the PTB).

### 3.1 Results

Table 1 shows the results for Experiment 1 (on the Dundee treebank, i.e. on parallel data), suggesting that the degree of informativeness of the gaze features that a parser can leverage differ among each other. In particular, we observe that the improvements across different models and the development and test sets are unstable. The evaluation on the dev set shows that gaze features from *early* and *context* group used as multiple auxiliary tasks (`early` and `context feats aux`) modestly improve the model in comparison with the baseline. With respect to the results on the test set, the most informative features are: `mean fix dur`,

[5] Note that the set of fine-tuned embeddings can be slightly different from that of Experiment 1, since words not occurring in Experiment 1 might be present in Experiment 2 due to the use of a different corpus.

| | Gaze features | dev set | | test set | |
|---|---|---|---|---|---|
| | | UAS | LAS | UAS | LAS |
| | *baseline* | 85.36 | 79.40 | 84.37 | 78.24 |
| *Basic* | total fix dur | 85.34 | 79.35 | 84.06 | 77.44 |
| | mean fix dur | 85.21 | 79.38 | 84.59 | **78.70** |
| | $n$ fix | 85.32 | 79.29 | 83.71 | 77.57 |
| | fix prob | 85.32 | 79.57 | 84.33 | 77.91 |
| | basic feats aux | 85.36 | 79.57 | 83.86 | 77.75 |
| *Early* | first fix dur | 85.30 | 79.46 | **84.64** | 78.57 |
| | first pass dur | 85.50 | 79.49 | 84.55 | 78.39 |
| | early feats aux | **85.61** | 79.57 | 84.37 | 78.11 |
| *Late* | $n$ re-fix | 85.52 | 79.25 | 83.86 | 77.91 |
| | reread prob | 85.34 | 79.57 | 83.86 | 77.37 |
| | late feats aux | 85.54 | 79.64 | 84.10 | 77.73 |
| *Context* | $w-1$ fix prob | 85.17 | 79.47 | 84.26 | 77.93 |
| | $w+1$ fix prob | 85.36 | 79.07 | 84.24 | 78.06 |
| | $w-1$ fix dur | 85.43 | 79.68 | 84.50 | 77.95 |
| | $w+1$ fix dur | 85.39 | 79.53 | **84.64** | 78.30 |
| | context feats aux | **85.61** | **79.72** | 84.33 | 78.24 |

Table 1: Impact of various gaze features as auxiliary task(s) on the score (UAS/LAS) of dependency parsing as the main task evaluated on Dundee treebank (parallel setup).

| | Gaze features | dev set | | test set | |
|---|---|---|---|---|---|
| | | UAS | LAS | UAS | LAS |
| | *baseline* | 93.98 | 91.67 | 93.86 | 91.80 |
| *Basic* | total fix dur | 93.94 | 91.60 | 93.99 | 91.92 |
| | mean fix dur | **94.12** | **91.84** | 93.95 | 91.82 |
| | $n$ fix | 93.97 | 91.70 | 93.91 | 91.87 |
| | fix prob | 93.98 | 91.71 | 93.99 | 91.93 |
| | basic feats aux | 94.00 | 91.69 | 93.84 | 91.81 |
| *Early* | first fix dur | 94.07 | 91.81 | 93.87 | 91.80 |
| | first pass dur | 93.93 | 91.58 | 93.79 | 91.70 |
| | early feats aux | 94.04 | 91.78 | 93.96 | 91.88 |
| *Late* | $n$ re-fix | 94.01 | 91.69 | 93.87 | 91.79 |
| | reread prob | 94.03 | 91.74 | 93.98 | 91.89 |
| | late feats aux | 93.98 | 91.58 | 93.92 | 91.90 |
| *Context* | $w-1$ fix prob | 94.02 | 91.65 | 93.95 | 91.93 |
| | $w+1$ fix prob | 93.88 | 91.61 | 93.89 | 91.82 |
| | $w-1$ fix dur | 94.06 | 91.65 | 93.86 | 91.83 |
| | $w+1$ fix dur | 93.91 | 91.69 | 93.89 | 91.84 |
| | context feats aux | 93.93 | 91.63 | **94.01** | **91.98** |

Table 2: Results for dependency parsing evaluated on PTB treebank with gaze features as auxiliary task(s) learned from the disjoint dataset: Dundee treebank.

which improves the LAS by $+0.46$, and first fix dur by $+0.33$.

Table 2 shows the results for Experiment 2 (using Dundee gaze data and PTB dependencies). Under this setup, the gains decrease in comparison with the results on the parallel setup. This could be partially caused by not using parallel data. The improvements seem to be more consistent between the development and test sets. This could be related to the fact that we use a larger (more representative) treebank. When looking at the dev set scores, the most discriminative gaze feature is mean fix dur that increases LAS by $+0.17$ and first fix dur by $+0.14$. On the other hand, evaluation on the test set shows that the most informative gaze features are from the *context* group learned as multiple auxiliary tasks (context feats aux) and they improve the LAS score by $+0.18$, followed by fix prob and $w-1$ fix prob with $+0.13$, total fix dur with $+0.12$ and late feats aux with $+0.10$. Results from both datasets suggest that grouping gaze features and treating them as multiple auxiliary tasks can improve the model's learning.

**Discussion** The experiments show that our method can give moderate gains for dependency parsing when leveraging gaze information.[6] However, the experiments also show that there is room for improvement, especially coming from generalization capabilities across different treebanks. This also opens the question of whether a different architecture could better suit the purpose of leveraging the gaze information in a consistent way. In this context, a potential line of work could adapt human-attention approaches (Barrett et al., 2018) for structured prediction and word-level classification, although it would come at a cost of speed for parsing as sequence labeling (Strzyz et al., 2019b).

## 4 Conclusion

This paper has explored how to leverage human data with a competitive dependency parser during training. This contrasts with most of previous work on parsing and gaze annotations, which worked under the assumption that eye-tracking data would be available at inference time. We address this problem and propose a method of leveraging gaze features by using it as an auxiliary task both with and without parallel data. We obtain modest but positive improvements, which opens the question about how to increase the leverage of eye-tracking or other complementary data that is only available during training or comes from a different dataset.

## Acknowledgements

This work has received funding from the European Research Council (ERC), under the European Union's Horizon 2020 research and innovation programme (FASTPARSE, grant agreement No 714150), from the ANSWER-ASAP project (TIN2017-85160-C2-1-R) from MINECO, and from Xunta de Galicia (ED431B 2017/01).

[6] Nevertheless, our approach can be employed also to leverage other types of data than eye measurements, i.e. NER or chunking data.

## References


Maria Barrett, Željko Agić, and Anders Søgaard. 2015. The Dundee treebank. In *The 14th International Workshop on Treebanks and Linguistic Theories (TLT 14)*.

Maria Barrett, Joachim Bingel, Nora Hollenstein, Marek Rei, and Anders Søgaard. 2018. Sequence classification with human attention. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 302–312, Brussels, Belgium. Association for Computational Linguistics.

Maria Barrett, Joachim Bingel, Frank Keller, and Anders Søgaard. 2016. Weakly supervised part-of-speech tagging using eye-tracking data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 579–584, Berlin, Germany. Association for Computational Linguistics.

Maria Barrett and Anders Søgaard. 2015a. Reading behavior predicts syntactic categories. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 345–349, Beijing, China. Association for Computational Linguistics.

Maria Barrett and Anders Søgaard. 2015b. Using reading behavior to predict grammatical functions. In *Proceedings of the Sixth Workshop on Cognitive Aspects of Computational Language Learning*, pages 1–5, Lisbon, Portugal. Association for Computational Linguistics.

Rich Caruana. 1997. Multitask learning. *Mach. Learn.*, 28(1):41–75.

Michael Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 1–8. Association for Computational Linguistics.

Susan A Duffy, Robin K Morris, and Keith Rayner. 1988. Lexical ambiguity and fixation times in reading. *Journal of memory and language*, 27(4):429–446.

Evelina Fedorenko, Zachary Mineroff, Matthew Siegelman, and Idan Blank. 2018. Word meanings and sentence structure recruit the same set of fronto-temporal regions during comprehension. *bioRxiv*, page 477851.

Ana V. González-Garduño and Anders Søgaard. 2018. Learning to predict readability using eye-movement data from natives and learners. In *32nd AAAI Conference on Artificial Intelligence, AAAI 2018, Proceedings*, pages 5118–5124.

Ana Valeria González-Garduño and Anders Søgaard. 2017. Using gaze to predict text readability. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 438–443, Copenhagen, Denmark. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Nora Hollenstein, Maria Barrett, Marius Troendle, Francesco Bigiolli, Nicolas Langer, and Ce Zhang. 2019. Advancing NLP with cognitive language processing signals. *CoRR*, abs/1904.02682.

Nora Hollenstein and Ce Zhang. 2019. Entity recognition at first sight: Improving NER with eye movement information. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1–10, Minneapolis, Minnesota. Association for Computational Linguistics.

Alan Kennedy, Robin Hill, and Joël Pynte. 2003. The Dundee corpus. In *Proceedings of the 12th European conference on eye movement*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Daniel J. Liebling and Sören Preibusch. 2014. Privacy considerations for a pervasive eye tracking world. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, UbiComp '14 Adjunct, pages 1169–1177, New York, NY, USA. ACM.

Alessandro Lopopolo, Stefan L. Frank, Antal van den Bosch, and Roel Willems. 2019. Dependency parsing with your eyes: Dependency structure predicts eye regressions during reading. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics*, pages 77–85, Minneapolis, Minnesota. Association for Computational Linguistics.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19(2):313–330.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy. European Language Resources Association (ELRA).

Mattias Nilsson and Joakim Nivre. 2009. Learning where to look: Modeling eye movements in

reading. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 93–101, Boulder, Colorado. Association for Computational Linguistics.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 1659–1666, Portorož, Slovenia. European Language Resources Association (ELRA).

Nils Reimers and Iryna Gurevych. 2017. Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 338–348, Copenhagen, Denmark.

M. Schuster and K.K. Paliwal. 1997. Bidirectional recurrent neural networks. *Trans. Sig. Proc.*, 45(11):2673–2681.

Abhinav Deep Singh, Poojan Mehta, Samar Husain, and Rajkumar Rajakrishnan. 2016. Quantifying sentence complexity based on eye-tracking measures. In *Proceedings of the Workshop on Computational Linguistics for Linguistic Complexity (CL4LC)*, pages 202–212, Osaka, Japan. The COLING 2016 Organizing Committee.

Michalina Strzyz, David Vilares, and Carlos Gómez-Rodríguez. 2019a. Sequence labeling parsing by learning across representations. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5350–5357, Florence, Italy. Association for Computational Linguistics.

Michalina Strzyz, David Vilares, and Carlos Gómez-Rodríguez. 2019b. Viable dependency parsing as sequence labeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 717–723, Minneapolis, Minnesota. Association for Computational Linguistics.

Takenobu Tokunaga, Hitoshi Nishikawa, and Tomoya Iwakura. 2017. An eye-tracking study of named entity annotation. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 758–764, Varna, Bulgaria. INCOMA Ltd.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 252–259.

David Vilares, Mostafa Abdou, and Anders Søgaard. 2019. Better, faster, stronger sequence tagging constituent parsers. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3372–3383, Minneapolis, Minnesota. Association for Computational Linguistics.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 188–193, Portland, Oregon, USA. Association for Computational Linguistics.

# Unsupervised Labeled Parsing with Deep Inside-Outside Recursive Autoencoders

**Andrew Drozdov, Pat Verga, Yi-Pei Chen, Mohit Iyyer, and Andrew McCallum**

College of Information and Computer Sciences
University of Massachusetts Amherst

{adrozdov, pat, yipeichen, miyyer, mccallum}@cs.umass.edu

## Abstract

Understanding text often requires identifying meaningful constituent spans such as noun phrases and verb phrases. In this work, we show that we can effectively recover these types of labels using the learned phrase vectors from deep inside-outside recursive autoencoders (DIORA). Specifically, we cluster span representations to induce span labels. Additionally, we improve the model's labeling accuracy by integrating latent code learning into the training procedure. We evaluate this approach empirically through unsupervised labeled constituency parsing. Our method outperforms ELMo and BERT on two versions of the Wall Street Journal (WSJ) dataset and is competitive to prior work that requires additional human annotations, improving over a previous state-of-the-art system that depends on ground-truth part-of-speech tags by 5 absolute F1 points (19% relative error reduction).

## 1 Introduction

The deep inside-outside recursive autoencoder (Drozdov et al., 2019, DIORA) is part of a recent trend in fully unsupervised neural constituency parsers (Shen et al., 2018; Williams et al., 2018a; Htut et al., 2018; Shen et al., 2019; Kim et al., 2019). However, these works and nearly all previous research (Klein and Manning, 2002; Seginer, 2007; Ponvert et al., 2011; Spitkovsky et al., 2013) have focused on *unlabeled* constituency parsing.

In this paper, we instead focus on *labeled* constituency parsing for English. The small number of previous works that exist in this area suffer from substantial weaknesses: 1) the models depend on ground-truth part-of-speech tags, which are not always available and known to boost constituency parsing scores (Kitaev and Klein, 2018), 2) none can simultaneously identify and label constituents (instead they typically depend on an external latent parser), and 3) they ignore sentences longer than ten tokens because previous latent parsers do not scale to longer sentences (Haghighi and Klein, 2006; Borensztajn and Zuidema, 2007; Reichart and Rappoport, 2008).

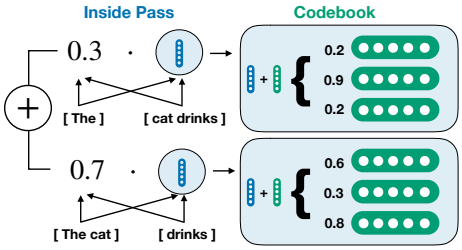


Figure 1: The left half of this figure depicts the inside-pass of the DIORA model as described in (Drozdov et al., 2019). We are interested in clustering the learned vectors $a(i, j)$ such that each span may be mapped to a phrase type. To enhance this clustering based approach, we augment the DIORA architecture with latent codes, shown in the right half of the figure.

Unlike previous work, we achieve strong results in unlabeled constituency parsing using a single model for both bracketing and labeling. Our approach relies on clustering span representations, which are fixed-length continuous vectors learned end-to-end using DIORA and do not require external resources such as part-of-speech tags. Furthermore, we enhance the DIORA architecture with latent codes: the model learns a distribution over these codes that loosely aligns with the ground-truth assignment of phrase types and, more importantly, improves the quality of the clusters.

Our code-enhanced DIORA architecture outperforms DIORA and achieves a new state of the art of 76.7 F1 on WSJ-10 when labeling a gold bracketing (19% relative error reduction over the previous best model, Haghighi and Klein 2006, which unlike our approach uses gold part-of-speech tags). Furthermore, we show DIORA is competitive when a ground truth bracketing is not provided, and instead must be induced. On



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1507–1512, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

the full WSJ test set, DIORA outperforms two strong baselines, ELMo (Peters et al., 2018a) and BERT (Devlin et al., 2019). We analyze the clustered constituents and observe they are separated syntactically (i.e. past tense vs. present participle verbs) and semantically (i.e. time-related phrases vs. references to people).

## 2 DIORA: Deep Inside-Outside Recursive Autoencoders

DIORA is a recursive autoencoder that learns to reconstruct an input sentence. A fundamental step in the reconstruction is to build a chart using the inside-outside algorithm (Baker, 1979), which represents a soft weighting over all possible binary trees of the input sentence. For all the model details, we refer the reader to Drozdov et al. (2019). For this work, it is key to understand two capabilities that DIORA provides: each span in a sentence is represented as a vector and DIORA induces a maximally likely binary tree for the sentence.

We can directly label the constituents of a sentence by clustering the learned span vectors from DIORA and assigning a label to each cluster. DIORA's autoencoder objective incentivizes the model to learn representations that compress the sentence well in order to reconstruct the input leading to the discovery of syntactic structure.

To encourage phrase representations to be easily clusterable into a small set of phrase types, we add an additional component to DIORA that forces phrase vectors to be representable by a small number of latent codes. Recent models have integrated ideas from vector quantization into variational autoencoders (Kingma and Welling, 2013) and key-value memory layers (Lample and Conneau, 2019), forcing the model to compress inputs into a single discrete latent embedding (van den Oord et al., 2017; Kaiser et al., 2018). Given a trained model, one could then assign labels to each of the latent variables and use this to label inputs directly.

We instead use a less restrictive modeling approach by assigning each input to a soft weighting over the $K$ latent embeddings. This is similar to the soft EM training used by Roy et al. (2018) and can be thought of analogously to fuzzy/soft $K$-means clustering (Dunn, 1974; Bezdek, 1981) rather than hard $K$-means clustering.

Implementation and training details for our model are described in Appendix A.1.

### 2.1 DIORA with Codebook

DIORA is constrained to binary trees and its composition is represented as:

$$a(i,j) = \text{Compose}(\bar{a}(i), \bar{a}(j)), \quad (1)$$

where $i$ and $j$ are neighboring spans, $\bar{a}$ is summary vector for all possible parses over a span, and Compose is a function such as tree-LSTM or multi-layer perceptron.

To add the latent codebook into the model, we modify Eq. 1 to combine each constituent vector with a weighted summation over latent codes:

$$f_{cb}(x) = C^T \sigma(CWx),$$

where $C$ is a codebook in $\mathbb{R}^{N \times M}$, $x$ is a constituent vector in $\mathbb{R}^M$, and $W$ is a bi-linear matrix used to compute the affinity between the constituent vector and the latent codes. One way to think of this equation is that each code (row in $C$) is a centroid, and the vector of affinity scores, $\sigma(CWx)$,[1] is a soft assignment of the constituent vector over the latent codes. The modified DIORA equation when incorporating the codebook is:

$$a'(i,j) = a(i,j) + f_{cb}(a(i,j)) \quad (2)$$

This codebook-enhanced architecture is visually depicted in Fig. 1. We use 70 codes when training this model (representing the 25 phrase types, 45 part-of-speech types, and ignoring the ROOT label), although we explore different configurations in §4.4.

## 3 Unsupervised Labeled Parsing

We perform unsupervised labeled constituency parsing with a multi-step approach.

**Tree assignment.** Assign a tree to each input sentence where the leaves of the tree are the words in the sentence. The tree is not labeled. This may be derived from the ground truth parse or induced using DIORA. When induced, we extract a binary tree by running the CKY algorithm[2] over DIORA's learned compatibility scores.

[1] More details about the equation $C^\top \sigma(CWx)$ are discussed in Appendix A.3. It's worth noting that $\sigma$ can be an arbitrary function, in this work we use the identity function.

[2] The CKY algorithm is an efficient dynamic programming approach for recognizing constituency trees using exact inference (Kasami, 1966; Younger, 1967; Rush et al., 2010).

| – WSJ (Test) – | Gold | | Induced | |
|---|---|---|---|---|
| **Model** | **F1**$_\mu$ | **F1**$_{max}$ | **F1**$_\mu$ | **F1**$_{max}$ |
| Upper Bound | 76.3 | 76.3 | 59.7 | 59.7 |
| Majority (NP) | 30.6 | 30.6 | 24.5 | 24.5 |
| ELMo | 58.5 | 59.4 | 43.5 | 48.2 |
| ELMo$_{CI}$ | 53.4 | 56.3 | 38.5 | 40.2 |
| BERT | 41.8 | 42.2 | 38.1 | 38.3 |
| DIORA | 62.5 $\pm$0.5 | 63.4 | 50.2 $\pm$0.5 | 51.4 |
| DIORA$_{CB}$ | 64.5 $\pm$0.6 | 65.5 | 49.8 $\pm$0.7 | 50.6 |
| DIORA$^*_{CB}$ | **66.4** $\pm$0.7 | **67.8** | 50.4 $\pm$0.7 | 51.5 |

Table 1: Results on the full Wall Street Journal test set.

| – WSJ-10 – | Gold | | Induced | |
|---|---|---|---|---|
| **Model** | **F1**$_\mu$ | **F1**$_{max}$ | **F1**$_\mu$ | **F1**$_{max}$ |
| Upper Bound | 86.0 | 86.0 | 64.6 | 64.6 |
| Majority (NP) | 32.0 | 32.0 | 25.2 | 25.2 |
| ELMo | 67.8 | 68.9 | 50.1 | 53.0 |
| ELMo$_{CI}$ | 65.9 | 67.3 | 46.0 | 47.6 |
| BERT | 54.6 | 57.8 | 44.5 | 45.2 |
| DIORA | 72.7 $\pm$1.5 | 76.2 | 55.2 $\pm$0.7 | 56.3 |
| DIORA$_{CB}$ | 73.2 $\pm$1.7 | 75.7 | 54.5 $\pm$1.2 | 56.6 |
| DIORA$^*_{CB}$ | **74.9** $\pm$1.1 | **76.7** | 53.9 $\pm$0.8 | 55.1 |
| PCFG† | - | 51.6 | - | 35.3 |
| BMM† | - | (76.8) | - | 59.5 |
| Proto† | - | 71.1 | - | 65.2 |

Table 2: WSJ-10 unsupervised labeled constituency parsing with punctuation removed. † indicates that the model relies on gold part-of-speech tags, and results in parentheses are related but not comparable to others in the table. Proto (Haghighi and Klein, 2006) uses additional hand written rules. BMM (*Gold*) (Borensztajn and Zuidema, 2007) is evaluated using more than $K$ clusters (where $K$ is the size of the tag set) by mapping ground truth labels to induced labels, therefore it is not strictly comparable to the other results. Neither BMM (Reichart and Rappoport, 2008) nor Proto are effective at inducing unlabeled structured, so depend on external latent parsers for the *Induced* evaluation, either CCM (Klein and Manning, 2002) or CCL (Seginer, 2007). ELMo and BERT do not induce structure whatsoever and depend on DIORA for the *Induced* evaluation. ELMo$_{CI}$ uses only the context-insensitive character embeddings produced by ELMo.

**Vector assignment.** Assign the corresponding span vector to each constituent in these trees over the entire dataset. For DIORA without the codebook, this will be the concatenation of inside and outside vector. When using the codebook, this will be one of two options: the same as for DIORA, except using the output of Eq. 2, or it will be the soft score assignment of the codebook $\sigma(CWx)$. The first option is referred to as DIORA$_{CB}$ and the soft score assignment as DIORA$^*_{CB}$.

**Cluster and label assignment.** Cluster the collection of constituent vectors using $K$ centroids learned with $K$-means. Finally, we use the ground truth phrase labels to assign each cluster to a phrase type — each constituent is mapped to the most common label within its cluster. We set $K$ equal to the number of distinct phrase types in order to match previous work.

## 4 Experiments and Results

### 4.1 DIORA

We compare multiple configurations of DIORA. The first is the original model DIORA using the concatenation of the inside and outside vectors to represent a phrase. We also look at the codebook-enhanced architecture DIORA$_{CB}$, and when clustering the codebook scores we refer to the model as DIORA$^*_{CB}$.

### 4.2 Baselines

While ELMo (Peters et al., 2018a) and BERT (Devlin et al., 2019) do not produce phrase vectors or induce recognizable constituency parse structure,[3] we show that they can still be used for unsupervised labeled parsing. When a reference parse is provided, it is only necessary to derive ad-hoc phrase vectors using the contextualized token vectors from these models. Peters et al. (2018b) describe an effective way to do so for ELMo, which involves concatenating the token vectors at the beginning and end of the phrase.[4] For BERT, it is critical to look at all layers as lower layers tend to be more syntactic in nature (Tenney et al., 2019). For both models, we report the max F1 and mean F1, and for the *Induced* evaluation we use the parses extracted from DIORA.

### 4.3 WSJ

Unsupervised constituency parsing has often been evaluated on different splits of the WSJ. For *labeled* constituency parsing, models that produce

[3] BERT does not strictly output word-level vectors. Rather, the output are subword vectors which we aggregate with mean-pooling to achieve a word-level representation.

[4] We tried many combinations (4 variants for ELMo and nearly 200 for BERT). They are described in Appendix A.2.

binary trees as output have a performance ceiling on this n-ary data — unary-chains limit recall[5] and more-than-binary nodes limit precision.

In some cases, an unlabeled tree structure over a sentence can be readily accessed. The algorithm described in §3 is robust to this case — simply replace the first step with the ground truth parse. We evaluate our model using the ground truth parse (Gold) and when inducing a parse (Induced). These results, comparison to baseline methods, and an upper-bound on binary tree performance are shown in Tables 1 and 2.

The Upper Bound in the *Induced* column of these tables represents a perfect labeling of the most accurate induced binary tree from DIORA, and the Majority (NP) row is the same tree labeled with the most common tag.

### 4.4 Model Ablations

As an alternative to clustering the constituent vectors with $K$-means, one can treat the codebook affinity scores, $\sigma(C^\top Wx)$, as a soft assignment over the clusters represented by each code. To examine this alternative, we replace $K$-means in the algorithm from §3 with the $\arg\max$ over the affinity scores. A model trained with 25 codes[6] achieves greater than 60% recall at labeling the ground truth trees for WSJ-10, indicating the codes represent some syntactic patterns although not as effectively as when using $K$-means.

Given these results, we are curious to see how model performance changes as the number of codes varies. We train codebook DIORA with $\{25, 70, 100, 200, 300, 400\}$ codes and evaluate each configuration using the procedure from §3 on the WSJ validation set. We compare the performance to non-codebook DIORA trained with $\{2, 3, 4, 5\}$ layers.[7] Results are shown in Fig. 2.

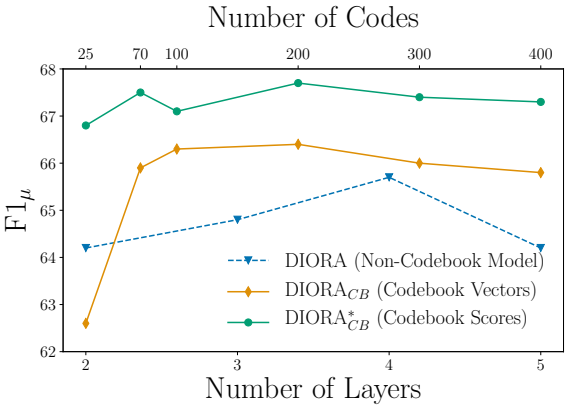


Figure 2: WSJ validation set results for different DIORA variants. The non-codebook DIORA sees improved performance as layers are added, but its average F1 never exceeds that of the codebook-enhanced architecture. Both clustering of codebook cells $a'(i, j)$ and codebook scores $\sigma(CWx)$ see performance improvements then diminishing returns as the number of codes increases. The dashed line uses the bottom x-axis and the solid lines use the top x-axis.

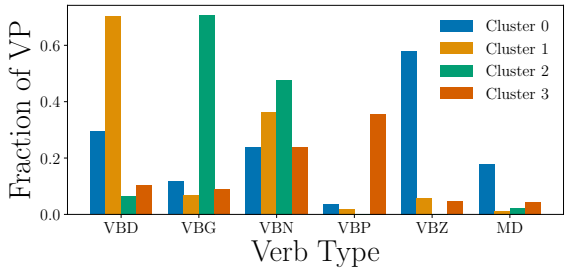


Figure 3: $DIORA_{CB}$ cluster assignment analyzed on WSJ-10 using part-of-speech tags. The four clusters shown were all assigned the VP label, yet seem to have finer-grained properties related to verb tense.

## 5 Qualitative Analysis of Clusters

We investigate phrase clusters from a single experiment ($DIORA_{CB}$ on WSJ-10), which are assigned to 9 NP, 5 VP, 5 S, 4 PP, 1 ADJP and 1 QP, according to the majority gold labels in that cluster. These 6 assigned phrase types correspond with the 6 most frequent labels.

We find some semantic properties are evident in the clusters. For example, a 100% correct NP cluster (all phrases in this cluster have gold label NP) are all *possessive NPs*. One of the NP clusters consists of NPs that are mostly related to *time* (15 minutes, last year, this fall), even the incorrectly labeled phrases are time-related such as the ADVP 'no longer' and 'so far'. Another NP cluster identifies *people*, which includes "ms. parks 's mother" but excludes "mr. noriega 's proposal" even though both phrases have the same part-of-speech tag sequence [NNP NNP POS NN].

One of the five VP clusters completely takes the form *to + verb* — 7 out of 11 mislabeled cases contain "to" in the phrase, for example, "not

[5] Labeled parsing is usually evaluated on whether a span has the correct label. An NP prediction for a span would be correct if there is an NP-QP or QP-NP unary-chain over this span. A binary tree could only ever get one of QP or NP correct in this case, hence limiting recall.

[6] We use 25 codes here instead of 70 so that the model may be fairly compared with previous systems.

[7] Elsewhere in this paper, DIORA uses two layers.

to mention" (CONJP). The four other VP clusters present some degree of tense and singular/plural properties. A bar-chart showing the finer-grained properties of the VP clusters is shown in Fig. 3. Cluster 0 includes the majority of VBZ and MD (will, won't, can, could), Cluster 1 is mainly composed of past tense VPs (VBD), Cluster 2 has many VBG, and Cluster 3 consists of 86% VBP.

One of the S clusters captures instances of S that do not cover the whole sentence. Another starts with coordinating conjunctions such as "and" or "but", yet another captures phrases beginning with personal pronouns or determiners.

## 6 Conclusions

In this paper, we show that DIORA can be used for unsupervised *labeled* constituency parsing. We also introduce a new codebook-enhanced variant of DIORA that improves labeling performance. Our model outperforms the previous state of the art in unsupervised labeled constituency parsing for the WSJ-10 dataset, even though the previous best uses ground truth part-of-speech tags and ours does not, and introduces the first results on the full WSJ test set. The results indicate that grammar induction with types is viable using recent neural-network-based models, and our analysis warrants further exploration in this area.

## Acknowledgements

We are grateful to our colleagues at UMass for help and advice, and to the UMass NLP reading group and the anonymous reviewers for feedback on drafts of this work. This work was supported in part by the Center for Intelligent Information Retrieval, in part by the National Science Foundation (NSF) grant numbers DMR-1534431, IIS-1514053 and CNS-0958392. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

## References

James K Baker. 1979. Trainable grammars for speech recognition. *The Journal of the Acoustical Society of America*, 65(S1):S132–S132.

James C Bezdek. 1981. *Pattern recognition with fuzzy objective function algorithms*. Springer Science & Business Media.

Gideon Borensztajn and Willem Zuidema. 2007. *Bayesian model merging for unsupervised constituent labeling and grammar induction*. Institute for Logic, Language and Computation (ILLC): Technical Report.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *North American Association for Computational Linguistics (NAACL)*.

Andrew Drozdov, Pat Verga, Mohit Yadav, Mohit Iyyer, and Andrew McCallum. 2019. Unsupervised latent tree induction with deep inside-outside recursive autoencoders. In *North American Association for Computational Linguistics (NAACL)*.

Joseph C Dunn. 1974. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. In *Journal of Cybernetics*.

Aria Haghighi and Dan Klein. 2006. Prototype-driven grammar induction. In *Association for Computational Linguistics (ACL)*.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *North American Chapter of the Association for Computational Linguistics (NAACL)*.

Phu Mon Htut, Kyunghyun Cho, and Samuel R Bowman. 2018. Grammar induction with neural language models: An unusual replication. In *Empirical Methods in Natural Language Processing (EMNLP): Short Paper*.

Łukasz Kaiser, Aurko Roy, Ashish Vaswani, Niki Parmar, Samy Bengio, Jakob Uszkoreit, and Noam Shazeer. 2018. Fast decoding in sequence models using discrete latent variables. In *International Conference on Machine Learning (ICML)*.

Tadao Kasami. 1966. An efficient recognition and syntax-analysis algorithm for context-free languages. *Coordinated Science Laboratory Report no. R-257*.

Yoon Kim, Alexander M. Rush, Lei Yu, Adhiguna Kuncoro, Chris Dyer, and Gábor Melis. 2019. Unsupervised recurrent neural network grammars. In *North American Association for Computational Linguistics (NAACL)*.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Association for Computational Linguistics (ACL)*.

Dan Klein and Christopher D. Manning. 2002. A generative constituent-context model for improved grammar induction. In *Association for Computational Linguistics (ACL)*.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*.

Aaron van den Oord, Oriol Vinyals, et al. 2017. Neural discrete representation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018a. Deep contextualized word representations. In *North American Association for Computational Linguistics (NAACL)*.

Matthew E Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. 2018b. Dissecting contextual word embeddings: Architecture and representation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Elias Ponvert, Jason Baldridge, and Katrin Erk. 2011. Simple unsupervised grammar induction from raw text with cascaded finite state models. In *North American Chapter of the Association for Computational Linguistics (NAACL)*.

Pytorch Core Team. 2019. Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration. http://pytorch.org/. Accessed: 2019-05-18.

Roi Reichart and Ari Rappoport. 2008. Unsupervised induction of labeled parse trees by clustering with syntactic features. In *Conference on Computational Linguistics (COLING)*.

Aurko Roy, Ashish Vaswani, Arvind Neelakantan, and Niki Parmar. 2018. Theory and experiments on vector quantized autoencoders. *arXiv preprint arXiv:1805.11063*.

Alexander M Rush, David Sontag, Michael Collins, and Tommi Jaakkola. 2010. On dual decomposition and linear programming relaxations for natural language processing. In *Empirical Methods in Natural Language Processing (EMNLP): Short Paper*.

Yoav Seginer. 2007. Fast unsupervised incremental parsing. In *Association for Computational Linguistics (ACL)*.

Yikang Shen, Zhouhan Lin, Chin-Wei Huang, and Aaron Courville. 2018. Neural language modeling by jointly learning syntax and lexicon. In *International Conference on Learning Representations (ICLR)*.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron C. Courville. 2019. Ordered neurons: Integrating tree structures into recurrent neural networks. In *International Conference on Learning Representations (ICLR)*.

Valentin I Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2013. Breaking out of local optima with count transforms and model recombination: A study in grammar induction. In *Empirical Methods in Natural Language Processing (EMNLP): Short Paper*.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. Bert rediscovers the classical nlp pipeline. In *Association for Computational Linguistics (ACL)*.

Adina Williams, Andrew Drozdov, and Samuel R Bowman. 2018a. Do latent tree learning models identify meaningful structure in sentences? *Transactions of the Association of Computational Linguistics (TACL)*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018b. A broad-coverage challenge corpus for sentence understanding through inference. In *North American Association for Computational Linguistics (NAACL)*.

Daniel H Younger. 1967. Recognition and parsing of context-free languages in time n3. *Information and control*, 10(2):189–208.

# Dependency Parsing for Spoken Dialog Systems

**Sam Davidson[1], Dian Yu[2], Zhou Yu[2]**
[1]Department of Linguistics
[2]Department of Computer Science
University of California, Davis
Davis, CA 95616, USA
{ssdavidson,dianyu,joyu}@ucdavis.edu

## Abstract

Dependency parsing of conversational input can play an important role in language understanding for dialog systems by identifying the relationships between entities extracted from user utterances. Additionally, effective dependency parsing can elucidate differences in language structure and usage for discourse analysis of human-human versus human-machine dialogs. However, models trained on datasets based on news articles and web data do not perform well on spoken human-machine dialog, and currently available annotation schemes do not adapt well to dialog data. Therefore, we propose the Spoken Conversation Universal Dependencies (SCUD) annotation scheme that extends the Universal Dependencies (UD) (Nivre et al., 2016) guidelines to spoken human-machine dialogs. We also provide ConvBank, a conversation dataset between humans and an open-domain conversational dialog system with SCUD annotation. Finally, to demonstrate the utility of the dataset, we train a dependency parser on the ConvBank dataset. We demonstrate that by pre-training a dependency parser on a set of larger public datasets and fine-tuning on ConvBank data, we achieved the best result, 85.05% unlabeled and 77.82% labeled attachment accuracy.

## 1 Introduction

Syntactic parsing (Chen et al., 2018) and semantic parsing (Kollar et al., 2018; Gupta et al., 2018) have been used in dialog systems to disambiguate the relationships between noun phrases in understanding tasks. Compared to constituency parsing and semantic role labeling, dependency parsing provides more clear relationships between predicates and arguments (Johansson and Nugues, 2008). Constituency parsers provide information about noun phrases in a sentence, but provide only limited information about relationships within a noun phrase. For example, in the sentence "What do you think about Google's privacy policy being reviewed by journalists from CNN?," a constituency parser would place "Google's privacy policy being reviewed by journalists from CNN" under a single phrasal node. Similarly, a semantic role labeling system would tend to label the same phrase as an argument of the verb, but it would not disambiguate the relationships within the phrase. Finally, NER only provides information about named entities which may or may not be the key semantic content of the sentence. Dependency parsers, by contrast, can provide information about relationships when a sentence contains multiple entities, even when those entities are within the same phrase.

Identifying relationships between entities in a user utterance can help a dialog system formulate a more appropriate response. For instance, in the sentence about "Google's privacy policy" mentioned above, there are multiple entities for the system to consider. The system must determine the most important entity in the utterance in order to model the topic and generate an appropriate response. A dependency graph facilitates this process by providing information about the syntactic relationships between entities.

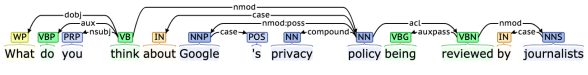


Figure 1: Example dependency-parsed sentence. Stanford CoreNLP visualizer (Manning et al., 2014).

A dependency parse of the sentence, shown in Figure 1, demonstrates that the key element of the sentence, "policy", is directly linked to the root "think" as a *nmod*. The other entities in the sentence are modifiers of the central element, and are clearly labeled as such. These structures make it



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1513–1519,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

easier for a dialog system to identify core semantic content and formulate an appropriate response.

Dependency parsing is a well studied (Dozat and Manning, 2016; Clark et al., 2018) task. However, most dependency parsers are trained on well-written text datasets, such as news articles and web-scrape data. In a conversational setting, dialog systems deal with transcripts that are produced by automated speech recognition (ASR) systems. ASR systems are prone to produce errors, such as dropped and mis-transcribed words, and prematurely terminated utterances. Such errors, in our experience, make building an automated dependency parser harder.

Additional data is needed to train parsers for the dialog system domain. Due to limitations in the UD scheme for parsing speech data, we propose an extension of the UD standard, SCUD, designed specifically for dialog systems. SCUD considers unique issues in automated speech transcription and what is useful for dialog systems to generate appropriate responses. We then use SCUD to annotate a corpus of open-domain human-system conversations with automated speech transcripts. We finally leverage this data, along with previously annotated UD datasets, to build a dependency parser. Since the training data used is collected with an open-domain dialog system, the trained dependency parser can be applied to dialog systems with any task.

In addition to its use in dialog systems, dependency parsed dialog data could be used in discourse analysis to assist in understanding the syntactic patterns people use when interacting with a computer. While researchers have investigated such differences in task-oriented dialog systems (Doran et al., 2003), we believe that ConvBank and SCUD present an opportunity for expanded discourse analysis of open-domain dialog systems.

## 2 Related Work

Dependency parsing on text such as the Universal Dependencies English Web Treebank (UD-EWT) (Silveira et al., 2014), is a well-developed field in NLP; however, the parsing of automated speech transcripts remains an open problem (Bechet et al., 2014). As has been well-documented, speech, and the resulting ASR transcipts, is a very different domain from written text (Biber and Conrad, 1999; Leech, 2000; Carter and McCarthy, 2017). Adams (2017) focuses on adapting a parser trained on UD-EWT to the speech domain without annotating new training data. For example, she attempts to create additional speech domain training data by parsing raw speech transcripts with Google's SyntaxNet. She then uses the resulting parsed output as training data for MaltParser (Nivre et al., 2006). However, the resulting models do not show consistent improvement over a baseline parser. Based on these results, and the apparent need for new training data for the speech domain, we chose to annotate a new human-system dialogs dataset to facilitate building dependency parsers for automated speech transcription and dialog systems.

Previous work has demonstrated that expanding the UD annotation scheme can result in successful parsers for other domains. For example, Liu et al. (2018) expand the UD scheme to train a parser for Twitter data. Thus we take a similar approach in expanding the UD scheme to encompass issues common to automated speech transcription.

Unlike written discourse, speech is full of disfluencies which make discovering the underlying syntactic structure challenging, as such disfluencies interrupt the syntactic structure of the utterance (Nasr et al., 2014; Caines et al., 2017). For example, according to Meteer and Taylor (1995), 17% of tokens in the Switchboard telephone conversations are various disfluencies. Frequent types of disfluencies include hesitations and false starts. When dealing with ASR-based systems, one must also contend with ASR errors. According to Nasr et al. (2014), such disfluencies must either be integrated into the syntactic parse of the utterance or removed prior to parsing. We choose to take the former approach, integrating speech disfluencies and speech transcription errors into the dependency graph produced by our annotation scheme and the resultant parser.

## 3 Annotation Scheme

When we initially began to annotate speech transcript data using the UD 2.0 (Nivre et al., 2017) scheme, our annotators repeatedly complained of constructions which they were unable to annotate given the limitations of UD 2.0. We developed SCUD in response to these issues. SCUD closely follows UD 2.0 so that we may leverage larger UD-compliant datasets of text data, such as UD-EWT. We extend the UD annotation scheme to encompass specific features peculiar to the speech domain, namely the use of non-syntactic discourse

markers, speech disfluencies, and ASR errors.

One major difference between speech and text is the fact that speech contains a fairly large number of discourse markers which do not serve a syntactic role in the utterance. This is similar to the difference between text and tweets pointed out by Liu et al. (2018). We make use of the UD relation *discourse* to indicate discourse markers which are not part of the syntactic structure of an utterance; for example, the word "like" in "I have like three dogs", where "like" plays a pragmatic, rather than a syntactic, role.

Another goal of our annotation scheme is to handle disfluencies common to both human and ASR-generated speech transcripts. We expand the scope of extant UD relation tags to tag common speech disfluencies. For example, the UD tag *reparandum* is applicable to situations in which a user self-corrects his or her speech. We use the UD tag *flat* to link repeated words and stutters to indicate that these headless multi-word expressions are a single syntactic unit. Handling non-syntactic tokens is particularly important because, while ASR may improve, the way people speak is less likely to change.

We are also able to expand the existing UD annotation scheme to handle some common errors in ASR output. In cases where an ASR system incorrectly splits words, we utilize the UD relation *goeswith* to link the two words in a dependency graph. We also provide a means to insert nodes at locations where an ASR system has dropped words necessary for formation of a grammatical structure. We do this by expanding the enhanced UD 2.0 standard, which provides a method to insert nodes in graphs as place-holders for words omitted through ellipsis. An example of using such a place-holder node is shown in Table 1, where the subject of the sentence in question has been omitted due to an ASR error. We also create one new relation tag to deal with a common error in ASR system output: prematurely terminated user utterances. Many conversational agents are turn-based (Kepuska and Bohouta, 2018) , and depend on pauses in user speech to detect the end of the user's dialog turn. As a result, hesitations by the user can result in the system prematurely terminating the user's turn. In reviewing our dialog system output data, we found that the ASR system prematurely terminates as many as 5% of user turns. We created the dependency label *preterm* to annotate such occurrences in our dataset.

| **Transcript: got two dogs** | | | | |
|---|---|---|---|---|
| **Index** | **Word** | **POS** | **Dep.** | **Relation** |
| 1 | E1.1 | PRON | 2 | nsubj |
| 2 | got | VERB | 0 | root |
| 3 | two | NUM | 4 | nummod |
| 4 | dogs | NOUN | 2 | obj |

Table 1: Example of omitted word node insertion

We must also consider the issue of incorrectly transcribed words. ASR systems are prone to mistranscribing words in the input signal; such errors can significantly change the meaning of the underlying utterance or result in a transcription which is ungrammatical or semantically nonsensical (Errattahi et al., 2018). While error correction methods for dialog systems have been developed (Choi et al., 2016), preprocessing our data with such a system is beyond the scope of this study. We instructed annotators to annotate all words as they were transcribed by the system. The alternative would involve annotators trying to infer user intent from context, which makes predictable annotation far less likely.

In total, SCUD has 37 primary relation tags. The relation tagset closely follows the UD 2.0 annotation standards with the exception of domain-specific extensions as noted above.

## 4 Dataset and Annotation Process

We collected a corpus of humans conversing with an open-domain dialog chatbot based on the Alexa platform (Chen et al., 2018). Users were asked to come into a lab and have an undirected conversation with our system. The automated speech transcripts of the users' utterances were collected and selected at random for inclusion in this corpus. We recruited two annotators who are experienced with dependency grammars and annotation projects to annotate 1,500 sentences. As is common with other UD annotation projects, we asked each annotator to correct the work of the other. Once this round of corrections was complete, we measured agreement as the percentage of tokens which remained unchanged between the original annotations and the corrected version. The two annotators agreed on 86.4% of unlabeled dependencies and 80.5% of labeled dependencies. These numbers are comparable to those reported on Tweebank (Liu et al., 2018), another challeng-

| UD-EWT | | ConvBank | |
|---|---|---|---|
| **Tag** | **Freq.** | **Tag** | **Freq.** |
| punct | 11.5% | root | 15.1% |
| case | 8.5% | nsubj | 13.8% |
| nsubj | 8.0% | obj | 9.2% |
| det | 7.7% | advmod | 8.2% |
| root | 6.1% | case | 7.2% |
| advmod | 5.4% | det | 6.8% |
| obj | 5.0% | obl | 6.0% |
| obl | 4.5% | aux | 5.3% |
| amod | 4.4% | cop | 4.1% |
| compound | 4.0% | amod | 3.2% |

Table 2: Comparison of common relation frequencies.

ing annotation problem. Once annotator agreement was assessed, a third annotator reviewed the annotations to improve compliance with the annotation scheme. We found that ASR errors, such as ungrammatical sentences resulting from mistranscription and word omission are a major contributing factor to annotator disagreement in our dataset.

The result of our annotation project is ConvBank, a corpus of 1,500 ASR-generated text utterances from a human-machine spoken dialog system with SCUD annotation. The corpus is compliant with UD 2.0 standards with the exception of extensions made to the UD standard as noted above. All data, code, and pre-trained models related to this project are publicly available via the project webpage[1].

ConvBank uses 32 of the 37 possible primary relations in SCUD, described above. A comparison of the ten most common tags in ConvBank and UD-EWT is presented in Table 2. *punct* is the most common relation in UD-EWT but is absent from ConvBank; this is because our automated speech transcripts do not contain punctuation. Other differences in relation distribution, such as the higher relative frequencies of *root* and *nsubj* can be explained by the shorter sentences in ConvBank compared to UD-EWT. Since each sentence must have a root, and most have a subject, the fewer other words in each sentence, the higher the frequencies of these two tags will tend to be.

## 5 Parsing Experiments

To demonstrate the utility of the ConvBank dataset, we use the data to train a dependency parser for ASR-based spoken dialog systems.

### 5.1 Training Data

We use three datasets in training our dependency parser. We use the publicly available UD-EWT (Silveira et al., 2014) and Tweebank v.2 (Liu et al., 2018) to train baseline models. We also combine UD-EWT and Tweebank to create a larger UD dataset for pre-training. Finally, we use our own ConvBank dataset to fine-tune our best-performing baseline model. We hold out 500 annotated ConvBank sentences for testing.

We considered using the constituency parsed Switchboard corpus (Godfrey et al., 1992), which is a human-human telephone corpus, due to its similarity to our domain. However, no fully UD-compliant version of the dependency parsed corpus is publicly available. Although conversion from constituency to dependency graphs is possible, we found the resulting dependency graphs unusable due to the error-prone conversion process. Peng and Zeldes (2018) point out that conversion systems particularly struggle with attaching nominal modifiers and oblique arguments.

### 5.2 Methods

We build all models using the biaffine parser as described in Dozat and Manning (2016) and Qi et al. (2019). We use the open-source implementations of Dozat's biaffine parser code[2] for training our models. We use Glove (Pennington et al., 2014) for pre-trained word embeddings. We train three baseline models to compare how models trained on text data compare with our model, which leverages both annotated text and automated speech transcripts. First, we train a parsing model on UD-EWT (Silveira et al., 2014) alone. We also train a separate model on Tweebank v.2 (Liu et al., 2018) alone to determine how a parser trained on tweets, which are arguably more similar to conversational speech, will perform. We also train a model on combined UD-EWT and Tweebank data. Our final baseline is a model trained on a 1,000 utterance subset of ConvBank. This model is used to determine how a parser trained on our data performs without pre-training on a larger text dataset.

Our fine-tuning approach is based on the idea that, by providing the system with additional domain-specific examples from ConvBank, we are able to build upon the system's baseline knowl-

[1] https://gitlab.com/ucdavisnlp/dialog-parsing

[2] https://github.com/tdozat/Parser-v3

edge to create a final system which performs substantially better than baseline. Stymne et al. (2018) demonstrate that this type of fine-tuning is an effective way to leverage multiple treebanks to train a single dependency parser. Our fine-tuning process consists of initializing our network weights using the weights from our baseline model trained on the UD English Web Treebank. As demonstrated by our results, the baseline model trained on UD-EWT alone performs reasonably well in assigning many UD labels and dependencies; however the system does not handle well issues specific to the dialog domain. To improve performance on the dialog domain, we resume training the pre-trained model with our annotated ConvBank data, using identical training parameters, until perplexity stabilizes.

### 5.3 Results and Discussion

We present the parsing accuracy on the ConvBank test set in Table 3. Training only using 1,000 annotated in-domain ConvBank data reaches 75.65% and 58.29% accuracy on unlabeled and labeled edges respectively. This suggests that 1,000 utterances are not enough to achieve optimal results. However, human annotations are expensive to obtain, limiting the size of our dataset. The model trained on EWT or Tweebank or EWT and Tweebank combined achieved similar results that are better than the model trained on ConvBank alone. As these three models had similar results, we fine-tune on the slightly better-performing EWT model. Fine-tune achieves the best best result on both unlabeled and labeled edges (85.05% and 77.82% respectively). Fine-tuning on EWT with ConvBank data improves parsing accuracy on the test data by over 7 points on unlabeled dependency edges and by nearly 12 points on labeled edges. These results suggest leveraging both a large set of out-of-domain data and in-domain data achieves the best performance.

There are some clear differences between our fine-tuned model and the baseline model trained on text data. In speech, discourse markers such as "but", "so", and "and" frequently begin utterances. These items can be difficult to parse correctly, as they more frequently serve grammatical functions. Our baseline model generally tags such items as coordinating conjunctions or adverbs, while in the fine-tuned model such items are more often appropriately tagged as discourse markers. Another clear example is the correct use of the *reparandum* tag. For example in the utterance "you know you my name", the baseline model tags the second "you" as an indirect object, while the fine-tuned model correctly tags the item as *reparandum*. The baseline model tags only three items in our test data as *reparandum*, while the fine-tuned model identifies thirty-one such instances. These results indicate that our fine-tuned model is more closely aligned to the speech domain than our baseline model. However, our best model still struggles when faced with fixed expressions common in conversational speech, such as "Top Gun" and "Kitchen Table Wisdom". The model also performs poorly on ungrammatical utterances, as expected.

| Dataset | UAS | LAS |
|---|---|---|
| **UD-EWT** | 77.41 | 65.85 |
| **Tweebank** | 77.23 | 66.44 |
| **Tweebank+UD-EWT** | 77.38 | 65.75 |
| **ConvBank** | 75.65 | 58.29 |
| **Fine-tune** | 85.05 | 77.82 |

Table 3: Parsing Accuracy Results

## 6 Conclusion

Dependency parsing can effectively identify the relationships between entities in a user utterance to improve speech language understanding. However, dependency parsing for speech data does not perform well when trained on out-of-domain text data. To solve this problem, we present ConvBank, a new dataset of annotated automated speech transcripts, annotated with the Spoken Conversation Universal Dependencies (SCUD) annotation scheme to build UD-compatible parses. We demonstrate the utility of ConvBank by fine-tuning a dependency parser with a large dataset, reaching 85.05% on labeled dependency edges and 77.82% on unlabeled dependencies.

In future work, we hope to expand the size of ConvBank to improve its utility to researchers in training parsing models and studying conversational systems. We also hope to improve the performance of our parsing system by jointly training dependency parsing with semantic role labeling.

## Acknowledgements

We would like to thank our skilled annotators, Hekang Jia and Julian Rambob.

## References


Allison Adams. 2017. Dependency parsing and dialogue systems : an investigation of dependency parsing for commercial application.

Frederic Bechet, Alexis Nasr, and Benoit Favre. 2014. Adapting dependency parsing to spontaneous speech for open domain spoken language understanding. In *Fifteenth Annual Conference of the International Speech Communication Association*.

Douglas Biber and Susan Conrad. 1999. Lexical bundles in conversation and academic prose. *Language and Computers*, 26:181–190.

Andrew Caines, Michael McCarthy, and Paula Buttery. 2017. Parsing transcripts of speech. In *Proceedings of the Workshop on Speech-Centric Natural Language Processing*, pages 27–36.

Ronald Carter and Michael McCarthy. 2017. Spoken grammar: Where are we and where are we going? *Applied Linguistics*, 38(1):1–20.

Chun-Yen Chen, Dian Yu, Weiming Wen, Yi Mang Yang, Jiaping Zhang, Mingyang Zhou, Kevin Jesse, Austin Chau, Antara Bhowmick, Shreenath Iyer, et al. 2018. Gunrock: Building a human-like social bot by leveraging large scale real user data.

Junhwi Choi, Donghyeon Lee, Seounghan Ryu, Kyusong Lee, Kyungduk Kim, Hyungjong Noh, and Gary Geunbae Lee. 2016. *Engine-Independent ASR Error Management for Dialog Systems*, pages 193–203. Springer International Publishing, Cham.

Kevin Clark, Minh-Thang Luong, Christopher D. Manning, and Quoc V. Le. 2018. Semi-supervised sequence modeling with cross-view training. *CoRR*, abs/1809.08370.

Christine Doran, John Aberdeen, Laurie Damianos, and Lynette Hirschman. 2003. Comparing several aspects of human-computer and human-human dialogues. In *Current and new directions in discourse and dialogue*, pages 133–159. Springer.

Timothy Dozat and Christopher D Manning. 2016. Deep biaffine attention for neural dependency parsing. *arXiv preprint arXiv:1611.01734*.

Rahhal Errattahi, Salil Deena, Asmaa El Hannani, Hassan Ouahmane, and Thomas Hain. 2018. Improving asr error detection with rnnlm adaptation. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 190–196. IEEE.

John J. Godfrey, Edward C. Holliman, and Jane McDaniel. 1992. Switchboard: Telephone speech corpus for research and development. In *Proceedings of the 1992 IEEE International Conference on Acoustics, Speech and Signal Processing - Volume 1*, ICASSP'92, pages 517–520, Washington, DC, USA. IEEE Computer Society.

Sonal Gupta, Rushin Shah, Mrinal Mohit, Anuj Kumar, and Mike Lewis. 2018. Semantic parsing for task oriented dialog using hierarchical representations. *CoRR*, abs/1810.07942.

Richard Johansson and Pierre Nugues. 2008. Dependency-based semantic role labeling of propbank. pages 69–78.

Veton Kepuska and Gamal Bohouta. 2018. Next-generation of virtual personal assistants (microsoft cortana, apple siri, amazon alexa and google home). In *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 99–103. IEEE.

Thomas Kollar, Danielle Berry, Lauren Stuart, Karolina Owczarzak, Tagyoung Chung, Lambert Mathias, Michael Kayser, Bradford Snow, and Spyros Matsoukas. 2018. The alexa meaning representation language. pages 177–184.

Geoffrey Leech. 2000. Grammars of spoken english: New outcomes of corpus-oriented research. *Language learning*, 50(4):675–724.

Yijia Liu, Yi Zhu, Wanxiang Che, Bing Qin, Nathan Schneider, and Noah A Smith. 2018. Parsing tweets into universal dependencies. *arXiv preprint arXiv:1804.08228*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Marie W. Meteer and Ann A. Taylor. 1995. Dysfluency annotation stylebook for the Switchboard corpus.

Alexis Nasr, Frederic Bechet, Benoit Favre, Thierry Bazillon, Jose Deulofeu, and Andre Valli. 2014. Automatically enriching spoken corpora with syntactic information for linguistic studies. In *LREC*, pages 854–858.

Joakim Nivre, Željko Agić, Lars Ahrenberg, Maria Jesus Aranzabe, Masayuki Asahara, Aitziber Atutxa, Miguel Ballesteros, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Eckhard Bick, Cristina Bosco, Gosse Bouma, Sam Bowman, Marie Candito, Gülşen Cebiroğlu Eryiğit, Giuseppe G. A. Celano, Fabricio Chalub, Jinho Choi, Çağrı Çöltekin, Miriam Connor, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Arantza Diaz de Ilarraza, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Marhaba Eli, Tomaž Erjavec, Richárd Farkas, Jennifer Foster, Cláudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökırmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Nizar Habash, Jan

Hajič, Linh Hà Mỹ, Dag Haug, Barbora Hladká, Petter Hohle, Radu Ion, Elena Irimia, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşıkara, Hiroshi Kanayama, Jenna Kanerva, Natalia Kotsyba, Simon Krek, Veronika Laippala, Phng Lê H`ông, Alessandro Lenci, Nikola Ljubešić, Olga Lyashevskaya, Teresa Lynn, Aibek Makazhanov, Christopher Manning, Cătălina Mărănduc, David Mareček, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Anna Missilä, Verginica Mititelu, Yusuke Miyao, Simonetta Montemagni, Amir More, Shunsuke Mori, Bohdan Moskalevskyi, Kadri Muischnek, Nina Mustafina, Kaili Müürisep, Lng Nguy˜ên Thị, Huy`ên Nguy˜ên Thị Minh, Vitaly Nikolaev, Hanna Nurmi, Stina Ojala, Petya Osenova, Lilja Øvrelid, Elena Pascual, Marco Passarotti, Cenel-Augusto Perez, Guy Perrier, Slav Petrov, Jussi Piitulainen, Barbara Plank, Martin Popel, Lauma Pretkalniņa, Prokopis Prokopidis, Tiina Puolakainen, Sampo Pyysalo, Alexandre Rademaker, Loganathan Ramasamy, Livy Real, Laura Rituma, Rudolf Rosa, Shadi Saleh, Manuela Sanguinetti, Baiba Saulīte, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Lena Shakurova, Mo Shen, Dmitry Sichinava, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Dima Taji, Takaaki Tanaka, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Larraitz Uria, Gertjan van Noord, Viktor Varga, Veronika Vincze, Jonathan North Washington, Zdeněk Žabokrtský, Amir Zeldes, Daniel Zeman, and Hanzhi Zhu. 2017. Universal dependencies 2.0. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Joakim Nivre, Marie-Catherine De Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan T McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal dependencies v1: A multilingual treebank collection. In *LREC*.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. Maltparser: A data-driven parser-generator for dependency parsing. In *In Proc. of LREC-2006*, pages 2216–2219.

Siyao Peng and Amir Zeldes. 2018. All roads lead to ud: Converting stanford and penn parses to english universal dependencies with multilayer annotations. In *Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018)*, pages 167–177.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *In EMNLP*.

Peng Qi, Timothy Dozat, Yuhao Zhang, and Christopher D Manning. 2019. Universal dependency parsing from scratch. *arXiv preprint arXiv:1901.10457*.

Natalia Silveira, Timothy Dozat, Marie-Catherine de Marneffe, Samuel Bowman, Miriam Connor, John Bauer, and Christopher D. Manning. 2014. A gold standard dependency corpus for English. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*.

Sara Stymne, Miryam de Lhoneux, Aaron Smith, and Joakim Nivre. 2018. Parser training with heterogeneous treebanks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 619–625.

# Span-based Hierarchical Semantic Parsing for Task-Oriented Dialog

**Panupong Pasupat**[*]
Stanford University
ppasupat@cs.stanford.edu

**Sonal Gupta**
Facebook Assistant
sonalgupta@fb.com

**Karishma Mandyam**[*]
University of Washington
krm28@cs.washington.edu

**Rushin Shah**[*]
Google
rushinshah@google.com

**Mike Lewis**
Facebook AI Research
mikelewis@fb.com

**Luke Zettlemoyer**
Facebook AI Research
lsz@fb.com

## Abstract

We propose a semantic parser for parsing compositional utterances into Task Oriented Parse (TOP), a tree representation that has intents and slots as labels of nesting tree nodes. Our parser is span-based: it scores labels of the tree nodes covering each token span independently, but then decodes a valid tree globally. In contrast to previous sequence decoding approaches and other span-based parsers, we (1) improve the training speed by removing the need to run the decoder at training time; and (2) introduce edge scores, which model relations between parent and child labels, to mitigate the independence assumption between node labels and improve accuracy. Our best parser outperforms previous methods on the TOP dataset of mixed-domain task-oriented utterances in both accuracy and training speed.

## 1 Introduction

Most commercial conversational AI systems parse task-oriented utterances using intent classification and slot-filling models (He and Young, 2003; Raymond and Riccardi, 2007; Mesnil et al., 2015), where the intent is the task of the utterance (e.g., `IN:GET_DIRECTION`) and the slots are the parameters needed to complete the task (e.g., `SL:DESTINATION`). This limited representation typically allows only a single intent per utterance and at most one slot label per token. Dialog systems using such a flat representation would struggle to handle compositional tasks that involve invoking multiple backend services (e.g., "direction to John's party": find John's address, and then find the direction to that address).

To support compositional utterances, the hierarchical Task Oriented Parsing (TOP) representation has recently been introduced (Gupta et al., 2018). As illustrated in Figure 1, the TOP representation

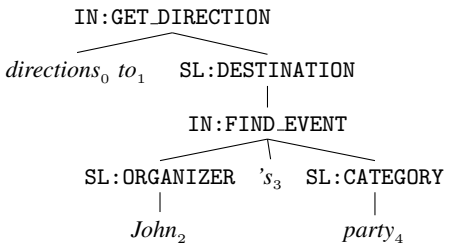


$$T = \{(0,5) : (\texttt{IN:GET\_DIRECTION}),$$
$$(2,5) : (\texttt{SL:DESTINATION}, \texttt{IN:FIND\_EVENT}),$$
$$(2,3) : (\texttt{SL:ORGANIZER}),$$
$$(4,5) : (\texttt{SL:CATEGORY}),$$
$$(0,1) : \varnothing, \quad (0,2) : \varnothing, \quad (0,3) : \varnothing, \quad \dots\}$$

Figure 1: An example TOP tree and its mapping representation $T$. (`IN:` = intent; `SL:` = slot)

is a tree where intents and slots are nested alternatively to model composition. The values inside intent and slot subtrees can then be used by downstream dialog modules to invoke appropriate services in a hierarchical fashion.

We propose a span-based semantic parser for parsing utterances into the TOP representation. In its most basic form, the parser embeds each token span (e.g., $x_{2:5}$ = *"John 's party"* in Figure 1) as a vector, and then uses it to predict the labels of the tree nodes covering the span (e.g., `SL:DESTINATION` and `IN:FIND_EVENT`). While the label prediction is done independently for each span, a CKY decoding algorithm is used to decode a valid tree with the maximum tree score.

Our main contributions are twofold. First, we reinterpret the ad-hoc tree score in previous span-based parsing work (Stern et al., 2017; Gaddy et al., 2018; Kitaev and Klein, 2018) as a joint distribution over the labels. Under this new framework, the loss function factors nicely, which allows us to train the model in a highly parallelized fashion instead of having to run the computation-

[*] work done while at Facebook Assistant.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1520–1526, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

ally expensive decoder during training. Second, we introduce *edge scores* to model the relationship between parent and child nodes in the tree. This mitigates the independence assumption and allows the decisions at child nodes to influence the parent node.

We evaluate our models on the TOP dataset (Gupta et al., 2018) of compositional utterances on events and navigation domains. Most utterances have nested intents, with some utterances containing intents from different domains. We demonstrate that our model is fast to train and outperforms previous models on the dataset.[1]

## 2 Related work

**Neural tree generation.** Previous work on syntactic and semantic parsing employs different strategies for generating trees. Approaches such as transition-based parsers (Dyer et al., 2016; Liu and Zhang, 2017) and top-down tree generation (Vinyals et al., 2015; Choe and Charniak, 2016; Dong and Lapata, 2016; Krishnamurthy et al., 2017; Yin and Neubig, 2018) frame tree generation as predicting a sequence of actions for generating the tree. The decoding processing generally consists of local decisions, and beam search is used to retain uncertainty. In contrast, global decoding methods (Durrett and Klein, 2015; Lee et al., 2016) can incorporate non-local features and decode a tree with the maximum global score.

**Span-based models.** Span-based models use the embeddings of token spans to perform prediction. By capturing the properties of the whole phrase instead of individual words, span embeddings can be suitable for tasks where phrases are the basic unit. Indeed, span-based models have recently shown great promise by achieving state-of-the-art results for segmentation and tagging (Kong et al., 2016), coreference resolution (Lee et al., 2017) and semantic role labeling (He et al., 2018).

For syntactic parsing, span embeddings have been used to score actions in shift-reduce parsing (Cross and Huang, 2016), or to score the existence and labels of tree nodes in bottom-up and top-down parsing (Stern et al., 2017; Kitaev and Klein, 2018). The analysis by Gaddy et al. (2018) shows that the span embeddings can learn to capture various information, such as label correlation and structural agreement, which was traditionally modeled by grammars or lexical features.

[1] The code is available at `https://github.com/ppasupat/factored-span-parsing`.

## 3 Setup

Given an utterance $x = (x_0, \ldots, x_{n-1})$ with $n$ tokens, the task is to predict a TOP parse tree, as illustrated in Figure 1. Each leaf node corresponds to a token $x_i$, while each non-terminal node covers some *span* $(i, j)$ with tokens $x_{i:j} = (x_i, x_{i+1}, \ldots, x_{j-1})$. The label $l$ of each non-terminal node is either an intent (prefixed with `IN:`) or a slot (prefixed with `SL:`). Two type constraints for TOP trees include: (1) intents can only have slots as children and vice versa, and (2) the tree root covering span $(0, n)$ must be an intent.

For our span-based parser, it is helpful to view the tree as a *mapping* $T$ from each span $(i, j)$ to a *chain* $c = (l_1, \ldots, l_k)$ which lists the $k \geq 0$ labels in the unary chain covering the span $(i, j)$. For instance, the span $(2, 5)$ in Figure 1 has $T[2, 5] =$ (`SL:DESTINATION`, `IN:FIND_EVENT`). If no non-terminal node covers the span, we denote the empty chain as $c = \varnothing$.

Following previous span-based parser work (Cross and Huang, 2016; Stern et al., 2017), we only consider unary chains that appear in the training data to be valid chains. (In our experiments, the training data has 57 distinct labels and 135 distinct chains. Only a single chain in the test data is not covered in training data.)

## 4 Model

Our parser is based on a joint distribution over the chain at each span. Let the probability of span $(i, j)$ having chain $c$ be a softmax over the possible chains:

$$p(T[i, j] = c) = \frac{\exp\left[f_{\mathrm{n}}(x_{i:j}, c)\right]}{\sum_{c'} \exp\left[f_{\mathrm{n}}(x_{i:j}, c')\right]}. \quad (1)$$

(The conditioning on $x$ is omitted.) Here, the *node score* $f_{\mathrm{n}}(x_{i:j}, c)$ is a real-valued function with $f_{\mathrm{n}}(x_{i:j}, \varnothing)$ fixed as 0. To compute the node scores for a span $x_{i:j}$, we embed the span using an LSTM-based span embedder[2] from Lee et al. (2017), and then apply a feedforward network to produce a real-valued score for each chain $c \neq \varnothing$.

We now make a simplifying assumption that the values of $T[i, j]$ are independent. The log-

[2] Refer to Appendix A for modeling details.

likelihood of the entire mapping $T$ becomes

$$\begin{aligned}\log p(T) &= \sum_{i<j} \log p(T[i,j]) \\ &= \sum_{i<j} \begin{bmatrix} f_{\mathrm{n}}(x_{i:j}, T[i,j]) \\ -\log \sum_{c'} \exp\left[f_{\mathrm{n}}(x_{i:j}, c')\right] \end{bmatrix}.\end{aligned} \tag{2}$$

The log-sum-exp term $\log \sum_{c'} \exp f_{\mathrm{n}}(x_{i:j}, c')$ does not depend on $T$. During inference, maximizing the log-likelihood $\log p(T)$ is thus equivalent to maximizing the *tree score*:

$$s(T) := \sum_{i<j} f_{\mathrm{n}}(x_{i:j}, T[i,j]). \tag{3}$$

At test time, we use CKY chart parsing to decode a valid TOP tree $T$ with the maximum score $s(T)$.

**Training.** The tree score $s(T)$ turns out to be the same scoring function used in previous span-based constituency parsing models (Stern et al., 2017; Gaddy et al., 2018; Kitaev and Klein, 2018). To train the model, these previous works use margin loss:

$$\max\left\{0, -s(T^*) + \max_T \left[\Delta(T, T^*) + s(T)\right]\right\} \tag{4}$$

where $T^*$ is the gold tree, $T$ is the predicted tree and $\Delta$ is a distance function. Computing the $\max_T$ term requires running a cost-augmented CKY decoder, which is computationally expensive and difficult to parallelize, especially when we add edge scores as described in the next section.

Instead, we propose to directly maximize the log-likelihood in Equation 2 of the gold trees in the training data. Concretely, given a gold tree $T^*$, our loss function is the negative log-likelihood $-\log p(T^*)$, which decomposes into a cross-entropy loss for each span $(i, j)$ according to Equation 2. We train our model by directly minimizing these cross-entropy loss terms; in other words, we treat the model as a multiclass classification model where the input is the span $x_{i:j}$ and the classes are the possible chains for the span.

Our training method is faster than using margin loss since it does not require running the CPU-bound CKY decoder during training. Moreover, the cross-entropy loss terms can be computed in parallel for all spans of *multiple examples* at once, which leads to further speed-up.

**Class weight for empty chains.** In practice, the number of spans $(i, j)$ with gold chains $T^*[i,j] = \varnothing$ (i.e., no subtree covering the span) is large compared to the total number of spans. To avoid the class imbalance problem, we scale the cross-entropy loss terms for spans with $T^*[i,j] = \varnothing$ by a hyperparameter $\alpha < 1$.

## 5 Edge scores

The model so far scores each span independently, which can be sub-optimal. For example, the prediction of the top intent only depends on the embedding of $x_{0:n}$, which is equivalent to standard intent classification or the first decision of a top-down tree generation. Similarly, ontology constraints (which intents can take which slots) are also not captured by the model.

To allows child nodes to influence the decision of the parent at inference time, we introduce *edge scores*. For a span $(i, j)$ with $T[i,j] = c = (l_1, \ldots, l_k)$, consider the parent node of the top-most non-terminal $l_1$, and define $\pi[i,j]$ to be its label. We model the conditional distribution of $\pi[i,j]$ as a softmax over all possible labels:

$$\begin{aligned}&p(\pi[i,j] = l \mid T[i,j] = c) \\ &\quad = \frac{\exp\left[f_{\mathrm{e}}(x_{i:j}, c, l)\right]}{\sum_{l'} \exp\left[f_{\mathrm{e}}(x_{i:j}, c, l')\right]}\end{aligned} \tag{5}$$

where $f_{\mathrm{e}}(x_{i:j}, c, l)$ is the *edge score*. Similar to node scores, we compute $f_{\mathrm{e}}(x_{i:j}, c, l)$ by applying a feed-forward network on the concatenation of the embeddings of $x_{i:j}$ and $c$. For convenience of notation, we let $\pi[i,j] = \varnothing$ when the parent does not exist (i.e., when $T[i,j] = \varnothing$ or $(i,j) = (0,n)$). In those situations, we let $f_{\mathrm{e}}(x_{i:j}, c, l)$ be 0 for $l = \varnothing$ and $-\infty$ otherwise.

We define the joint log-likelihood of $T$ and $\pi$:

$$\begin{aligned}&\log p(T, \pi) \\ &= \sum_{i<j} \Big[\log p(T[i,j]) + \log p(\pi[i,j] \mid T[i,j])\Big] \\ &= \sum_{i<j} \begin{bmatrix} f_{\mathrm{n}}(x_{i:j}, T[i,j]) \\ -\log \sum_{c'} \exp\left[f_{\mathrm{n}}(x_{i:j}, c')\right] \\ +f_{\mathrm{e}}(x_{i:j}, T[i,j], \pi[i,j]) \\ -\log \sum_{l'} \exp\left[f_{\mathrm{e}}(x_{i:j}, T[i,j], l')\right] \end{bmatrix}.\end{aligned} \tag{6}$$

Unlike in the original model, the last log-sum-exp term in Equation 6 *does* depend on the tree $T$, and thus has to be included in the tree score. Our

new tree score is

$$s(T) := \sum_{i<j} \left[ \begin{array}{l} f_{\mathrm{n}}(x_{i:j}, T[i,j]) \\ + \log p(\pi[i,j] \mid T[i,j]) \end{array} \right] \quad (7)$$

where $p(\pi[i,j] \mid T[i,j])$ is the softmax over edge scores as defined in Equation 5. Note that we cannot replace node scores $f_{\mathrm{n}}(x_{i:j}, T[i,j])$ with their log-softmax this way since we want dummy nodes to contribute a score of 0 to the tree score.

To train the model, we again directly minimize the negative log-likelihood of the gold tree $T^*$. The loss function factors into a cross-entropy loss term for each span $(i,j)$ and for each edge in $T^*$.

**Discussion.** The joint modeling of $T[i,j]$ and $\pi[i,j]$ is closely related to the parent annotation technique in syntactic parsing (Johnson, 1998; Klein and Manning, 2003; Petrov et al., 2006), where certain non-terminal labels are split into multiple labels based on their parents in the grammar rule (e.g., `VP` with parent `S` becomes `VP^S`). In our framework, we can view the label and edge scores (Equation 7) as a score over parent-annotated labels $(T[i,j], \pi[i,j])$:

$$s(T) = \sum_{i<j} f_{\mathrm{a}}(x_{i:j}, T[i,j], \pi[i,j]) \quad (8)$$

where $f_{\mathrm{a}}(x_{i:j}, \varnothing, \varnothing) = 0$. From our preliminary experiments, we found that modeling $f_{\mathrm{a}}$ as a combination of label and edge scores, as in Equation 7, is empirically better than directly applying a softmax over possible pairs $(T[i,j], \pi[i,j])$.

One analysis in Gaddy et al. (2018) shows that the span embedding of $x_{i:j}$ is powerful enough to identify the parent label $\pi[i,j]$ with high accuracy. However, in TOP semantic trees, parent and child nodes can have span boundaries that are far apart (e.g., the top intent covering span $(0, 15)$ might have a child slot covering span $(7, 9)$). As an alternative to increasing the power of the span embedder to handle long-range relations, which could lead to overfitting, adding edge scores is a simpler way to model relations between labels.

## 6 Experiments

We evaluate our models on the TOP dataset (Gupta et al., 2018) of hierarchical intent-slot annotations for utterances in navigation and event domains.[3] We use the dataset version with the noisy `IN:UNSUPPORTED_*` intents excluded, which contains 28410 training, 4032 development, and 8241 test examples. The main evaluation metric is exact match accuracy: the fraction of predicted parses that *exactly* match the annotated parses. We also report the labeled bracket F1 score for the parse tree constituents (Black et al., 1991).

**Training details.** We highlight crucial modeling decisions of our models and defer other details to Appendix B.

- For a fair comparison, we use the same token and sequence embedders as Gupta et al. (2018): 200-dimensional GloVe embeddings and 2-layer 164-dimensional biLSTMs. We expect contextual embeddings (Peters et al., 2018; Devlin et al., 2018) and a transformer-based sequence embedder (Kitaev and Klein, 2018) to further improve the results, as observed in Einolghozati et al. (2018).

- The class weight $\alpha$ for empty chains is tuned on the development data: $\alpha = 0.4$ for the basic model and $\alpha = 0.2$ for the model with edge scores. We will later demonstrate how different choices of $\alpha$ affect the results.

**Baselines.** We compare our method to existing approaches proposed for this task in Gupta et al. (2018): a shift-reduce parser based on Recurrent Neural Network Grammars (RNNG) (Dyer et al., 2016), and the best sequence-to-sequence model that produces linearization of the trees based on CNN utterance embeddings (Gehring et al., 2017).[4] We also consider the span-based parsers by Stern et al. (2017) with the additional improvements from Gaddy et al. (2018).[5] The parsers were designed for constituency parsing, are trained with cost-augmented margin loss, and use either a bottom-up CKY decoder (Stern-chart) or a top-down greedy decoder (Stern-greedy).

**Accuracy.** Table 1 shows the exact match accuracy and bracket scores on the test data. The span-based model without edge scores is comparable to the baseline RNNG model. With edge scores, the span-based model outperforms the baselines in both exact match accuracy and labeled F1 scores.

[3] http://fb.me/semanticparsingdialog

[4] https://github.com/facebookresearch/pytext

[5] https://github.com/mitchellstern/minimal-span-parser

| Method | Acc | F1 | Time |
|---|---|---|---|
| seq2seq | 78.24 | 90.78 | 8m |
| RNNG | 80.63 | 92.61 | 1h 16m |
| Stern-chart | 80.66 | 93.03 | 25m |
| Stern-greedy | 80.79 | 92.83 | 22m |
| ours (no $f_e$) | 80.80 | 93.35 | **5m** |
| ours (+ $f_e$) | **81.80** | **93.63** | 9m |

Table 1: The test exact match accuracy, labeled bracket F1, and training time per epoch of different methods.

| Error type | no $f_e$ | + $f_e$ |
|---|---|---|
| wrong top intent | 106 | 99 |
| wrong label except top intent | 381 | 367 |
| wrong non-terminal boundary | 133 | 133 |
| missing a non-terminal | 153 | 154 |
| spurious non-terminal | 198 | 185 |
| joining two gold non-terminals | 20 | 26 |
| splitting a gold non-terminal | 27 | 36 |
| other errors | 6 | 4 |

Table 2: The error breakdown on the development data of our span-based models. Note that an example can have multiple errors.

**Training speed.** Our span-based models can be trained in a highly parallelized fashion without having to run the computationally expensive decoder. Table 1 shows the average wall-clock time used to train the model for one epoch over the training data.

**Error analysis.** We compare the errors made by the span-based parsers on the development set. Table 2 provides a breakdown of the error counts of the models on development data.

**Recall trade-off.** As described in Section 4, the hyperparameter $\alpha$ controls the weight of the loss terms for the class $c = \varnothing$ (i.e., not building a subtree for the span). As such, we can use $\alpha$ to trade off two types of errors: missing non-terminals (over-predicting $c = \varnothing$) and spurious non-terminals (under-predicting $c = \varnothing$). As shown in Figure 2, higher $\alpha$ encourages the model to predict $c = \varnothing$ more frequently, leading to more missing non-terminals (intents and slots) and fewer spurious ones. We can also tune $\alpha$ based on the downstream tasks. For instance, getting a higher slot recall using a small $\alpha$ is arguably better for dialog systems since other downstream modules (e.g., entity linker) can detect and discard spurious slots.

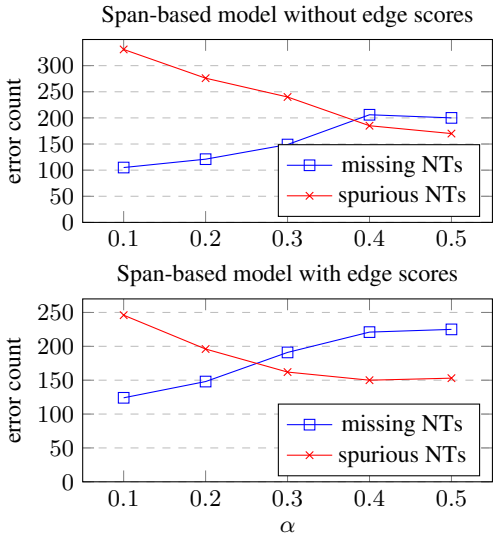


Figure 2: Trade-off between missing and spurious non-terminals with different $\alpha$ (weight for class $c = \varnothing$).

When $\alpha$ is low, one common type of errors is when the models incorrectly split a non-terminal. This usually happens when the gold slot consists of two sub-phrases that can be interpreted as slots on their own (e.g., a `SL:CATEGORY_EVENT` slot *"holiday concerts"* is split into a `SL:DATE_TIME` slot *"holiday"* and a `SL:CATEGORY_EVENT` slot *"concerts"*). As the tree score is the sum of span scores, the parser is biased toward creating two non-terminal nodes instead of one. Luckily, in the context of task-oriented dialogs, this type of errors tends to have small effects on the semantic interpretation, and sometimes even provides more useful information for downstream modules.

## 7 Conclusion

We presented the first span-based parser for parsing utterances into the hierarchical intent-slot representation. The log-likelihood objective allows us to train the model without having to decode a tree in a highly parallelized fashion, while edge scores can explicitly capture the parent-child relationship even when their boundaries are far apart.

Apart from standard accuracy improvement techniques such as better token embeddings and ensembling, possible future directions include a more fine-grained control of the recall trade-off, modeling the tokens outside the non-terminals instead of ignoring them, incorporating the parent's embedding in edge scores, and a more efficient or approximate decoder similar to the greedy decoder from Stern et al. (2017).

## References


Ezra Black, Steven P. Abney, D. Flickenger, Claudia Gdaniec, Ralph Grishman, P. Harrison, Donald Hindle, Robert Ingria, Frederick Jelinek, Judith L. Klavans, Mark Liberman, Mitchell P. Marcus, Salim Roukos, Beatrice Santorini, and Tomek Strzalkowski. 1991. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proceedings of the Workshop on Speech and Natural Language*.

Do Kook Choe and Eugene Charniak. 2016. Parsing as language modeling. In *Empirical Methods in Natural Language Processing (EMNLP)*.

James Cross and Liang Huang. 2016. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Association for Computational Linguistics (ACL)*.

Greg Durrett and Dan Klein. 2015. Neural crf parsing. In *Association for Computational Linguistics (ACL)*.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *North American Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Arash Einolghozati, Panupong Pasupat, Sonal Gupta, Rushin Shah, Mrinal Mohit, Mike Lewis, and Luke Zettlemoyer. 2018. Improving semantic parsing for task oriented dialog. In *Conversational AI Workshop at NeurIPS*.

David Gaddy, Mitchell Stern, and Dan Klein. 2018. What's going on in neural constituency parsers? an analysis. In *North American Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann Dauphin. 2017. Convolutional sequence to sequence learning. In *International Conference on Machine Learning (ICML)*.

Sonal Gupta, Rushin Shah, Mrinal Mohit, Anuj Kumar, and Mike Lewis. 2018. Semantic parsing for task oriented dialog using hierarchical representations. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Luheng He, Kenton Lee, Omer Levy, and Luke S. Zettlemoyer. 2018. Jointly predicting predicates and arguments in neural semantic role labeling. In *Association for Computational Linguistics (ACL)*.

Yulan He and S. Young. 2003. A data-driven spoken language understanding system. In *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*.

Mark Johnson. 1998. Pcfg models of linguistic tree representations. *Computational Linguistics*, 24.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Association for Computational Linguistics (ACL)*.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Association for Computational Linguistics (ACL)*.

Lingpeng Kong, Chris Dyer, and Noah A. Smith. 2016. Segmental recurrent neural networks. In *International Conference on Learning Representations (ICLR)*.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Kenton Lee, Luheng He, Mike Lewis, and Luke S. Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Kenton Lee, Mike Lewis, and Luke S. Zettlemoyer. 2016. Global neural ccg parsing with optimality guarantees. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Jiangming Liu and Yue Zhang. 2017. Shift-reduce constituent parsing with neural lookahead features. *Transactions of the Association for Computational Linguistics (TACL)*, 5.

Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Z. Hakkani-Tür, Xiaodong He, Larry P. Heck, Gökhan Tür, Dong Yu, and Geoffrey Zweig. 2015. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke S. Zettlemoyer. 2018. Deep contextualized word representations. In *North American Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Association for Computational Linguistics (ACL)*.

Christian Raymond and Giuseppe Riccardi. 2007. Generative and discriminative algorithms for spoken language understanding. In *INTERSPEECH*.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Association for Computational Linguistics (ACL)*.

Oriol Vinyals, Lukasz Kaiser, Terry K Koo, Slav Petrov, Ilya Sutskever, and Geoffrey E. Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems (NIPS)*.

Pengcheng Yin and Graham Neubig. 2018. Tranx: A transition-based neural abstract syntax parser for semantic parsing and code generation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

# Enhancing Context Modeling with a Query-Guided Capsule Network for Document-level Translation

**Zhengxin Yang**[123], **Jinchao Zhang**[3], **Fandong Meng**[3]
**Shuhao Gu**[12], **Yang Feng**[12⋆], **Jie Zhou**[3]
[1] University of Chinese Academy of Sciences
[2] Key Laboratory of Intelligent Information Processing
Institute of Computing Technology, Chinese Academy of Sciences (ICT/CAS)
[3] Pattern Recognition Center, WeChat AI, Tencent Inc, China
{yangzhengxin17z,gushuhao17g,fengyang}@ict.ac.cn
{dayerzhang,fandongmeng,withtomzhou}@tencent.com

## Abstract

Context modeling is essential to generate coherent and consistent translation for Document-level Neural Machine Translations. The widely used method for document-level translation usually compresses the context information into a representation via hierarchical attention networks. However, this method neither considers the relationship between context words nor distinguishes the roles of context words. To address this problem, we propose a query-guided capsule networks to cluster context information into different perspectives from which the target translation may concern. Experiment results show that our method can significantly outperform strong baselines on multiple data sets of different domains.

## 1 Introduction

The encoder-decoder based Neural machine translation (NMT) models (Sutskever et al., 2014; Bahdanau et al., 2014; Wu et al., 2016; Vaswani et al., 2017; Zhang et al., 2019) have made great progresses and drawn much attention in recent years. In practical applications, NMT systems are often fed with a document-level input which requires reference resolution, the consistency of tenses and noun expressions and so on. Many researchers have proven that contextual information is essential to generate coherent and consistent translation for document-level translation (Hardmeier, 2012; Meyer and Webber, 2013; Sim Smith, 2017; Jean et al., 2017; Maruf and Haffari, 2018; Miculicich et al., 2018; Zhang et al., 2018a; Voita et al., 2018; Wang et al., 2017; Tu et al., 2018; Maruf et al., 2019). Despite the great success of the above models, they are designed for sentence-level translation tasks and exclude contextual information in the model architecture.

On these grounds, the widely used hierarchical attention networks (HAN) was proposed to integrate contextual information in document-level translation (Miculicich et al., 2018). In this method, the context sentences are considered in the form of hierarchical attentions retrieved by the current generation. That is, it utilizes a word-level attention to represent a sentence and then a sentence-level attention to represent all the involved context. In this way, the final attention representation has to encode all the information needed for coherent and consistent translation, including reference information, tenses, expressions and so on. To get the multi-perspective information, it is necessary to distinguish the role of each context word and model their relationship especially when one context word could take on multiple roles (Zhang et al., 2018b). However, this is difficult to realize for the HAN model as its final representation for the context is produced with an isolated relevance with the query word which ignores relations with other context words.

To address the problem, we introduce Capsule Networks into document-level translation which have proven good at modelling the *parts-wholes* relations between low-level capsules and high-level capsules (Hinton et al., 2011; Xiao et al., 2018; Sabour et al., 2017; Hinton et al., 2018; Gu and Feng, 2019). With capsule networks, the words in a context source sentence is taken as low-level capsules and the information of different perspectives is treated as high-level capsules. Then in the dynamic routing process of capsule networks, all the low-level capsules trade off against each other and consider over all the high-level capsules and drop themselves at a proper proportion to the high-level capsules. In this way, the relation among low-level capsules and that between low-

Joint work with Pattern Recognition Center, WeChat AI, Tencent Inc, China.
⋆ Corresponding Author



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1527–1537,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

level capsules and high-level capsules are both explored. In order to make sure high-level capsules indeed cluster information needed by the target translation, we apply capsule networks to both sides of the current sentence and add a regularization layer using Pearson Correlation Coefficients to force the high-level capsules on the two sides to approach to each other. In addition, we still need to ensure the final output of capsule networks is relevant to the current sentence. Therefore we propose a Query-guided Capsule Network (QCN) to have the current source sentence to take part in the routing process so that high-level capsules can retain information related to the current source sentence.

To the best of our knowledge, this is the first work which applies capsule networks to document-level translation tasks and QCN is also the first attempt to customize attention for capsule networks in translation tasks. We conducted experiments on three English-German translation data sets in different domains and the results demonstrate that our method can significantly improve the performance of document-level translation compared with strong baselines.

## 2 Background

### 2.1 Sentence-level NMT and Transformer Model

Sentence-level NMTs are generally based on an encoder-decoder framework (Sutskever et al., 2014; Bahdanau et al., 2014; Wu et al., 2016; Vaswani et al., 2017; Zhang et al., 2019; Meng and Zhang, 2019). In this framework, the encoder encodes the source sentence $\boldsymbol{x} = \{x_1, x_2, \cdots, x_M\}$ into a sequence of continuous representations $\boldsymbol{z} = \{z_1, z_2, \cdots, z_M\}$. Given $\boldsymbol{z}$, the decoder predicts target words in order and returns the target translation $\boldsymbol{y} = \{y_1, y_2, \cdots, y_N\}$. The objective function of NMT is to maximize the log-likelihood of a set of source-target language sentence pairs as

$$\mathcal{L}(\Theta; \boldsymbol{x}, \boldsymbol{y}) = \frac{1}{|\mathcal{S}|} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{S}} \log P(\boldsymbol{y}|\boldsymbol{x}; \Theta) \quad (1)$$

where $\mathcal{S}$ is the training set.

As we implement our approaches based on the Transformer architecture (Vaswani et al., 2017), which is a strong sentence-level NMT baseline, we give a brief description of the Transformer model. The encoder and decoder are composed of similar layers which consists of two general types of sub-layers: multi-head attention mechanism and point-wise fully connected feed-forward network (FFN).

**Encoder**: The encoder of the Transformer is composed of $N$ identical layers. Each of these layers includes a multi-head self-attention mechanism that allows each position of the output of previous encoder layer to attend to all other positions, as well as a position-wise FFN, which is stacked on top of the multi-head self-attention, composed of two linear transformations and a ReLU activation function.

**Decoder**: The architecture of the decoder is similar to the encoder, however, it employs an additional multi-head attention sub-layer over the encoder output between the multi-head self-attention sub-layer and position-wise FFN sub-layer. The multi-head self-attention sub-layer needs to mask the input target tokens in the future.

### 2.2 Document-level NMT

The document-level translation task is to translate each source sentence with consideration of the previous context in the document. Formally, the translation of a document $\boldsymbol{D}$ containing $|\boldsymbol{D}| = J$ sentence pairs can be defined as given the source document $\boldsymbol{X}$ in order and the translation system generates each translation $\boldsymbol{y^j} \in \boldsymbol{Y}$ in order. The document translation probability can be defined as:

$$P(\boldsymbol{Y}|\boldsymbol{X}; \Theta) = \prod_{j=1}^{J} P(\boldsymbol{y}^j|\boldsymbol{x}^j, \boldsymbol{D}_{<j}; \Theta) \quad (2)$$

where $\boldsymbol{x}^j$, $\boldsymbol{y}^j$ denote the $j_{th}$ source and target sentence respectively, and $\boldsymbol{D}_{<j}$ denotes all of the previous sentence pairs in document. For each sentence $\boldsymbol{x}^j$, each target word is generated according to the source representation and the generated target hypothesis, therefore the Eq. (2) can be formulated as:

$$P(\boldsymbol{Y}|\boldsymbol{X}; \Theta) = \prod_{j=1}^{J} \prod_{i=1}^{I^j} P(y_i^j|, \boldsymbol{y}_{<i}^j, \boldsymbol{x}^j, \boldsymbol{D}_{<j}; \Theta) \quad (3)$$

where $y_i^j$ denotes the $i_{th}$ word of the $j_{th}$ translation $\boldsymbol{y}^j$ with the length as $I^j$ and $\boldsymbol{y}_{<i}^j$ denotes the generated target hypothesis.

The training objective of document-level NMTs is to maximize the log-likelihood of translations in

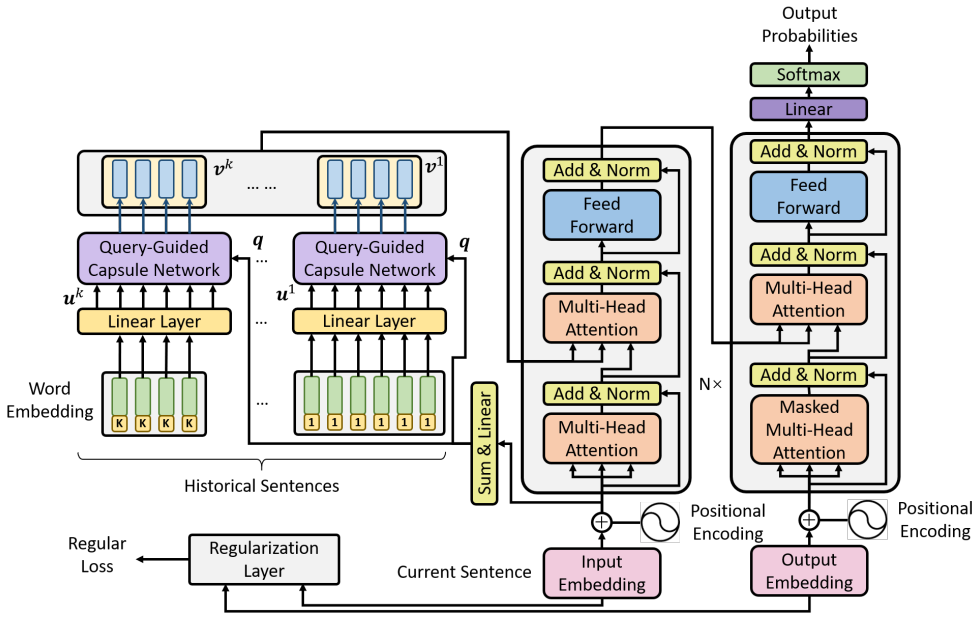


Figure 1: The overall architecture consists of three modules: the *Query-guided Capsule Network* in the upper left of this figure, the *Regularization Layer* in the lower left of this figure and the *Sub-layer-expanded Transformer* in the right of this figure.

document context as following:

$$\mathcal{L}(\Theta; \boldsymbol{X}, \boldsymbol{Y}) = \frac{1}{|\mathcal{D}|} \sum_{(\boldsymbol{X},\boldsymbol{Y})\in\mathcal{D}} \log P(\boldsymbol{Y}|\boldsymbol{X}; \Theta) \tag{4}$$

where $\mathcal{D}$ is the training set of the DocNMT.

Compared with the sentence-level NMTs, the critical part of document-level NMTs is to effectively capture and utilize the related contextual information when translating the to-be-translated source sentence.

## 3 Our Approach

In this section, we introduce the proposed Query-guided Capsule Network (QCN) for enhancing the document-level NMT. First, we present the overall architecture of the network, and then we describe the QCN in detail.

### 3.1 Overall Architecture

We aim to enhance the document-level NMT performance through effectively capturing and employing the contextual features in each historical sentence that related to the current sentence. We integrating a novel Query-guided Capsule Network into the sentence-level Transformer-based NMT (Vaswani et al., 2017) to capture the document-level contextual information for translating the current source sentence.

As shown in Figure 1, the overall architecture of our translation model is composed of three modules:

- **The Query-guided Capsule Network** takes the to-be-translated source sentence as the query to guide the procedure of retrieving related and helpful contextual features from historical sentences with a novel dynamic routing algorithm.

- **Sub-layer-expanded Transformer** contains a new sub-layer that attending the contextual features extracted from the QCN to effectively utilize them for translation.

- **Regularization Layer** contains two conventional Capsule Networks to unify the source sentence and the target sentence into an identical semantic space through computing an extra PCCs loss item at the training stage.

### 3.2 Query-guided Capsule Network

The Capsule Network (CapsNet) (Sabour et al., 2017) was proposed to build parts-wholes relationships in the iterative routing procedure, which can be used to capture features in historical sentences from low level to high level. Capsules in the lower layer vote for those in the higher layer by aggregating their transformations with iteratively updated

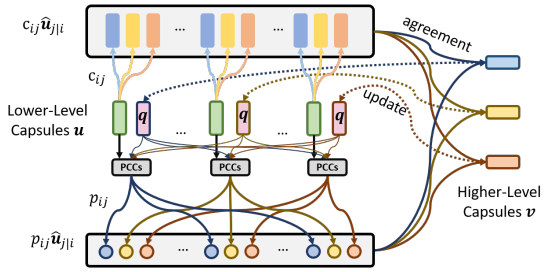


Figure 2: Query-guided Capsule Network, the green blocks in the middle of the figure are lower-level capsules $\boldsymbol{u}_i$, right-side blocks indicate the higher-level capsules $\boldsymbol{v}_j$ and the query vector $\boldsymbol{q}$ is initially tiled to length $|\boldsymbol{v}|$ and updated with corresponding higher-level capsules. Here shows a Query-guided Capsule Network with 3 high-level (feature) capsules. In each iteration, the PPCs is computed according to the input vector $\boldsymbol{u}_i$ and the query vector $\boldsymbol{q}_j$, then $p_{ij}\hat{\boldsymbol{u}}_{j|i}$ and $c_{ij}\hat{\boldsymbol{u}}_{j|i}$ are added to obtain the higher-level capsules which will be used to update corresponding query later.

coupling coefficients. However, there exists an obvious drawback of directly applying the Capsule Network into the document-level NMT for capturing contextual features. The reason is that the CapsNet can only extract internal features without considering whether features are related to the to-be-translated source sentence.

To address this issue, we proposed the Query-guided Capsule Network (QCN), which employ the representation of the to-be-translated source sentence as the query to guide the feature extraction procedure of the Capsule Network. In this way, contextual features that generated from the QCN are based on the internal semantic relations among each historical sentence and the external semantic relations between historical sentences and the to-be-translated source sentence. The QCN is based on an improved dynamic routing algorithm which will be detailed introduced in the next section.

### Improved Dynamic Routing of QCN

Given a query vector $\boldsymbol{q}$ and a set of input capsules $\boldsymbol{u} = \{\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_n\}$, the dynamic routing algorithm iteratively calculates the correlation between the query vector and each input capsule and updates output capsules. Specifically, query vector $\boldsymbol{q}$ is the representation of source to-be-translated sentence and input capsules $\boldsymbol{u}$ are all word embeddings in previous sentences, which can be formulated as:

$$\boldsymbol{q} = \mathrm{g}(\sum_{x \in \boldsymbol{x}} \mathrm{embedding}(x)) \quad (5)$$

$$\boldsymbol{u}_i^{-k} = \mathrm{f}([\mathrm{embedding}(x_i^{-k}); \mathrm{onehot}(k)]) \quad (6)$$

where $f$ and $g$ are both linear transformation functions and $k$ indicates the distance of historical sentence from the to-be-translated sentence. Each word embedding in historical sentences is concatenated with a distance-determined one-hot vector to provide positional markers. Compared to the dynamic routing method proposed by (Sabour et al., 2017), our improved dynamic routing method of QCN can model information that related to the query among input capsules.

Algorithm 1 shows details of the algorithm and Figure 2 illustrates the interaction of the various components in the QCN. Initially, the improved dynamic routing algorithm of QCN recieves a sequence of lower-level capsules $\boldsymbol{u} = \{\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_n\}$ and a query vector $\boldsymbol{q}$ and

**Algorithm 1** Improved Dynamic Routing

**Input:** $r$ , $\boldsymbol{q}$ and $\boldsymbol{u} = \{\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_n\}$
**Output:** $\boldsymbol{v} = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_m\}$

// Initialization
**for** each output capsule $\boldsymbol{v}_j$ in higher-layer **do**
  **for** each input capsule $\boldsymbol{u}_i$ in lower-layer **do**
    $\hat{\boldsymbol{u}}_{j|i} \leftarrow \boldsymbol{W}_{ij}\boldsymbol{u}_i$
    $\alpha_{ij} \leftarrow 0$
    $\boldsymbol{q}_j \leftarrow \boldsymbol{q}$
    $p_{ij} \leftarrow \tanh(\mathrm{PCCs}(\boldsymbol{u}_i, \boldsymbol{q}_j))$
    ▷ PCCs computes Eq 7
  **end for**
**end for**

// Iteration
$itr \leftarrow 0$
**repeat**
  **for** each input capsule $\boldsymbol{u}_i$ in lower-layer **do**
    $\boldsymbol{c}_i \leftarrow \mathrm{softmax}(\boldsymbol{\alpha}_i)$
  **end for**

  **for** each output capsule $\boldsymbol{v}_j$ in higher-layer **do**
    $\boldsymbol{s}_j \leftarrow \sum_{i=1}^{n}(c_{ij} + p_{ij})\hat{\boldsymbol{u}}_{j|i}$
    $\boldsymbol{v}_j \leftarrow \mathrm{squash}(\boldsymbol{s}_j)$
    ▷ squash computes Eq 8
  **end for**

  **for** each output capsule $\boldsymbol{v}_j$ in higher-layer **do**
    **for** each input capsule $\boldsymbol{u}_i$ in lower-layer **do**
      $\alpha_{ij} \leftarrow \alpha_{ij} + p_{ij}\hat{\boldsymbol{u}}_{j|i}\boldsymbol{v}_j$
    **end for**
    $\boldsymbol{q}_j \leftarrow \frac{\boldsymbol{q}_j + \boldsymbol{v}_j}{2}$
    **for** each input capsule $\boldsymbol{u}_i$ in lower-layer **do**
      $p_{ij} \leftarrow \tanh(\mathrm{PCCs}(\boldsymbol{u}_i, \boldsymbol{q}_j))$
    **end for**
  **end for**
  $itr \leftarrow itr + 1$
**until** $itr = r$

then calculates a Pearson Correlation Coefficients (PCCs) between $\boldsymbol{q}$ and each input capsule $\boldsymbol{u}_i$ (line 7). The PCCs is a measure of the linear correlation between two variables. When PCCs is close to +1, it means they have very strong positive linear correlation, and close to -1 means total negative correlation. Given a pair of variables $(\boldsymbol{A}, \boldsymbol{B})$, the formula for computing PPCs is

$$\begin{aligned}\text{PCCs}(\boldsymbol{A}, \boldsymbol{B}) &= \frac{\text{Cov}(\boldsymbol{A}, \boldsymbol{B})}{\sigma_{\boldsymbol{A}}\sigma_{\boldsymbol{B}}} \\ &= \frac{(\boldsymbol{A} - \frac{1}{n}\sum_{i=1}^{n} a_i)^T(\boldsymbol{B} - \frac{1}{n}\sum_{i=1}^{n} b_i)}{\|\boldsymbol{A}\| \cdot \|\boldsymbol{B}\|} \end{aligned} \quad (7)$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ are both $n$-dimension vectors, Cov is the covariance and $\sigma_{\boldsymbol{A}}, \sigma_{\boldsymbol{B}}$ is the standard deviation of $\boldsymbol{A}$ and $\boldsymbol{B}$ respectively.

The routing iteration process then computes coupling coefficients, denoted as $\boldsymbol{c}_i$(line 16), with regard to a input capsule $\boldsymbol{u}_i$ and all the higher-level capsules $\boldsymbol{v}$. In the original dynamic routing algorithm (Sabour et al., 2017), coupling coefficients are only determined by the cumulative "agreement", which are the prior probabilities that capsule $\boldsymbol{u}_i$ should be coupled to capsule $\boldsymbol{v}_j$. However, in DocNMT situation, it is far from enough to cluster the high-level information from the capsules in lower-level that related to query vector according to a naive "agreement". To address this issue, we reduce the "agreement", when PCCs show a negative linear correlation between the query vector $\boldsymbol{q}$ and the input vector $\boldsymbol{u}_i$. Instead, we increase the "agreement", when PCCs are positive (line 27). The query vector $\boldsymbol{q}$ is initially tiled to length $|\boldsymbol{v}|$ and updated with the corresponding higher-level capsules $\boldsymbol{v}_j$ in each iteration (line 29).

Our routing iteration updates higher-level capsules by adding $\sum_{i=1}^{n} p_{ij}\hat{\boldsymbol{u}}_{j|i}$ to $\boldsymbol{s}_j$. This step can add more information related to the query vector and cut off the unrelated features (line 20). It is necessary to get different length of output capsules shrunk into the 0 to 1 interval using "squash" function (line 21) proposed by Sabour et al. (2017) which is shown in Eq.(8).

$$\text{squash}(\boldsymbol{t}) = \frac{||\boldsymbol{t}||^2}{1 + ||\boldsymbol{t}||^2}\frac{\boldsymbol{t}}{||\boldsymbol{t}||} \quad (8)$$

where $\boldsymbol{t}$ can be the initial input capsule $\boldsymbol{u}_i$ or the vector $\boldsymbol{s}_j$ for predicting the output capsule.

### 3.3 Sub-layer-expanded Transformer

To effectively utilize contextual features extracted from each historical sentence by QCN, we introduce an additional context-aware multi-head attention sub-layer in each layer of the Transformer encoder. The multi-head attention can attend to all the positions of the contextual features with outputs of the previous sub-layer as a query. Then the output of the context-aware multi-head attention sub-layer is fed into the point-wise feed-forward sub-layer in each layer of the encoder. The right part in Figure 1 shows details of the Sub-layer-expanded Transformer. Specifically, the equation of multi-head attention computing procedure is as following:

$$\begin{aligned}\text{MultiHead}(Q,K,V) &= \text{Concat}(H_1,\cdots,H_h)W^O \\ H_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \quad (9)$$

where $W_i^Q$, $W_i^K$ and $W_i^V$ are the parameter matrices, $Q$, $K$ and $V$ indicates the query, key and value representations. The computation of the attention function is as following:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \quad (10)$$

where $d_k$ indicates the dimension of queries $Q$ and keys $K$.

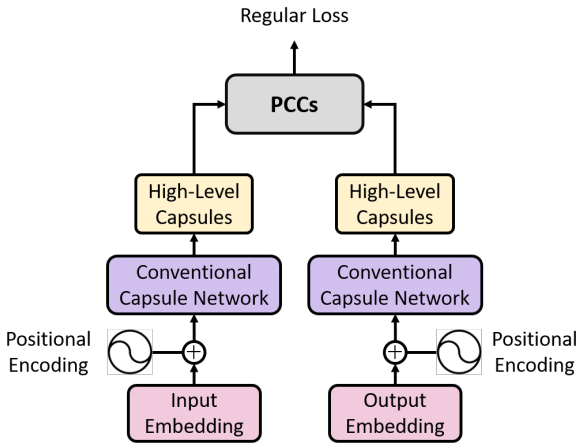


Figure 3: Regularization Layer. In the training stage, this layer takes the inputs of the encoder (left part of the figure) and decoder (right part of the figure) as the input capsules of two capsule networks separately, and computes the PCCs between the high-level capsules of two networks.

### 3.4 Regularization Layer

To better modeling the document-level translation task, we incorporate a regularization layer (Figure 3) into the whole architecture to restrict the source sentence and target sentence to an identical semantic space. This layer separately feeds the inputs

of encoder and decoder into two capsule networks $\text{Caps}_{enc}$ and $\text{Caps}_{dec}$, and computes the PCCs between the outputs of two networks. We regard the PCCs as an extra regularization term in the final objective at the training stage. The loss function of our model can be formulated as:

$$\mathcal{L}(\Theta; \boldsymbol{X}, \boldsymbol{Y}) = \frac{1}{|\mathcal{D}|} \times \sum_{j=1}^{J} \sum_{i=1}^{I^j} \{\log P(y_i^j | , \boldsymbol{y}_{<i}^j, \boldsymbol{x}^j, \boldsymbol{D}_{<j}; \Theta) + \text{PCCs}(\text{Caps}_{enc}(\boldsymbol{x}^j), \text{Caps}_{dec}(\boldsymbol{y}^j))\} \quad (11)$$

where $\Theta$ are parameters of the model, $\boldsymbol{D}_{<j}$ are historical sentences of the to-be-translated source sentence, $\boldsymbol{x}^j$ is the to-be-translated sentence and $\boldsymbol{y}_{<i}^j$ denotes the generated target hypothesis.

## 4 Experiments

### 4.1 Settings

#### Datasets and Evaluation Metrics

We carry out experiments on English-German translation tasks in three different domains: talks, news, and speeches. The corpora statistics are shown in Table 1.

- **TED.** This corpus is a Machine Translation part of the IWSLT 2017 (Cettolo et al., 2012) evaluation compaigns[1], each TED talk is considered to be a document. we take the *tst2016-207* as the test set, and other as our development set.
- **News.** We take the sentence-aligned document-delimited News Commentary v11 corpus[2] as our training set. The WMT'16 *news-test2015* and *news-test2016* are used for development and testing respectively.
- **Europarl.** The corpus are extracted from the Europarl v7 (Koehn, 2005) according to the method mentioned in Maruf and Haffari (2018). The training, development and test sets are obtained through randomly splitting the corpus.

We download all of above extracted corpora[3] from Maruf et al. (2019). The tokenization and truecase pre-processing are implemented on all datasets using the scripts of the Moses Toolkit[4] (Koehn et al., 2007). We also apply segmentation into BPE subword units[5] (Sennrich et al., 2016) with 30K merge operations.

We use two metrics: BLEU (Papineni et al., 2002) and Meteor (Lavie and Agarwal, 2007) to evaluate the translation quality.

#### Models and Baselines

We use the Transformer architecture as our context-agnostic baseline and adopt three context-aware baselines (Zhang et al., 2018a; Miculicich et al., 2018; Maruf and Haffari, 2018).

We performed the same configuration on our models according to the settings of the Maruf and Haffari (2018). Specifically, for the Transformer, we set the hidden size and point-wise FFN size as 512 and 2048 respectively. We use 4 layers and 8 attention heads in both encoder and decoder. All dropout rates are set to 0.1 for context-agnostic model and 0.2 for context-aware model.

In the training phase, we use the default Adam optimizer (Kingma and Ba, 2014) with a fixed learning rate of 0.0001. The batch size is 1500 on TED dataset and 900 on both News and Europarl datasets.

### 4.2 Results and Analysis

#### Main Results

Table 2 shows that our model surpasses all the context-agnostic(Vaswani et al., 2017) and context-aware(Zhang et al., 2018a; Miculicich et al., 2018; Maruf and Haffari, 2018) baselines on TED and Europarl datasets. For TED dataset, the performance of our model greatly exceeds that of all other baselines, and is better than Miculicich et al. (2018) with a gain of +0.59 BLEU and +0.61 Meteor. For Europarl dataset, our model got improvements with a gain of +0.07 on BLEU metric, but the Meteor score is +0.64 higher than Maruf et al. (2019) which utilize the whole document as the contextual information, whereas we only using 3 previous sentences.

Results on the sequence-level Transformer and our DocNMTs show that the captured contextual features provide helpful semantic information for enhancing the translation quality. The regularization term that we proposed can effectively

[1] https://wit3.fbk.eu/

[2] http://www.casmacat.eu/corpus/news-commentary.html

[3] https://github.com/sameenmaruf/selective-attn/tree/master/data

[4] https://github.com/moses-smt/mosesdecoder/tree/master/scripts

[5] https://pypi.org/project/subword-nmt/

| | TED | | News | | Europarl | |
|---|---|---|---|---|---|---|
| | Sent No. | Doc len avg | Sent No. | Doc len avg | Sent No. | Doc len avg |
| Training | 206,126 | 121.39 | 236,287 | 38.93 | 1,666,904 | 14.14 |
| Development | 8,967 | 96.42 | 2,169 | 26.78 | 3,587 | 14.95 |
| Test | 2,271 | 98.74 | 2,999 | 19.35 | 5,134 | 14.26 |

Table 1: The statistics of the training/development/test corpora in number of sentence pairs and the average document length (in sentences).

| | TED | | News | | Europarl | |
|---|---|---|---|---|---|---|
| **Model** | **BLEU** | **Meteor** | **BLEU** | **Meteor** | **BLEU** | **Meteor** |
| Transformer-DocNMT (Zhang et al., 2018a) | 24.00 | 44.69 | 23.08 | 42.40 | 29.32 | 46.72 |
| HAN-DocNMT (Miculicich et al., 2018) | 24.58 | 45.48 | **25.03** | **44.02** | 28.60 | 46.09 |
| SAN-DocNMT (Maruf et al., 2019) | 24.42 | 45.38 | 24.84 | 44.27 | 29.75 | 47.22 |
| Transformer | 23.28 | 44.17 | 21.67 | 41.11 | 28.72 | 46.22 |
| Transformer + Regularization Term | 24.55 | 45.57 | 22.09 | 41.77 | 29.42 | 47.59 |
| Transformer + QCN | 24.41 | **46.09** | 22.22 | 41.90 | 29.48 | 47.49 |
| Transformer + QCN + Regularization Term | **25.19** | 45.91 | 22.37 | 41.88 | **29.82** | **47.86** |

Table 2: BLEU and Meteor scores of models. There exists three context-aware baseline models and a context-agnostic model. "Regularization Term" denotes integrating the regularization layer into the Transformer model and "QCN" denotes incorporating Query-guided Capsule Network in the Transformer model.

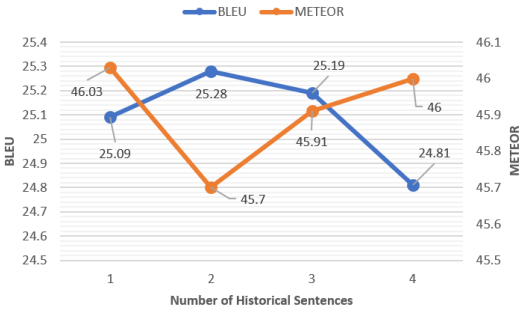


Figure 4: Effect of Contextual Information Scope. It is not that the more historical sentences we utilize, the better translation performance is. Effect of scope on BLEU and Meteor scores are inconsistent to some degree.

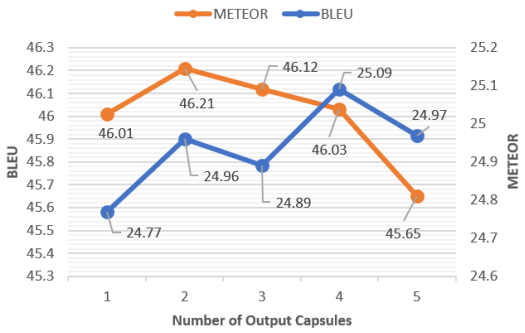


Figure 5: Effect of Feature Capsule Number. Appropriate capsule number is important for model performance.

further improve the model performance on TED and Europarl datasets. For the restriction of the GPU memory, we have to filter long sentences to keep our model running. Although, it hurts the model performance on the "News" dataset (contains many long sentences), the QCN module and regularization term still bring improvements.

### Effect of Contextual Information Scope

To investigate the effect of contextual information scope, we carry on the number of historical sentences hyper-parameter experiments on the TED talk dataset. We fix the hyper-parameters of the QCN by setting both the number of higher-level capsules and routing iteration to four, and investigating the impact of changes in the number of historical sentences on BLEU and Meteor scores. Figure 4 shows that using one historical sentence in QCN can obtain the best Meteor score while the highest BLEU score is presented when we utilize two historical sentences. We found that the growth of Meteor and BLEU scores are opposite. Therefore, the choice of utilizing how many historical sentences is a trade-off between both scores. We choose three historical sentences as our final setting. Through experimentation, we also found it is not that the more historical sentences we utilize, the better translation performance is.

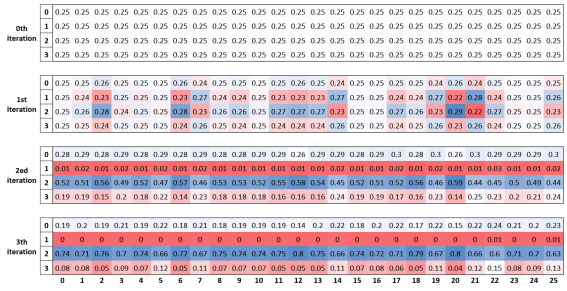


Figure 6: Agreement distribution of a sentence (26 words) as inputs of the QCN and 4 contextual features as output. Blue color means higher agreement and the red means lower. From top to bottom are four heat maps in $0_{th}$ to $3_{th}$ iterations.

### Effect of Feature Capsule Number

QCN is the crucial part of our overall architecture and the positive and negative impact depends on the configuration of the QCN. Therefore, We also investigate the effect of the hyper-parameter of QCN: the number of feature capsules.

We set the number of historical sentence as 3 according to the previously experimental results, and the number of routing iteration is set to 4. Figure 5 shows that Meteor score become highest when the number of higher-level capsules is set as 2, but BLEU score can obtain best score at 4. We finally choose 4 as the final setting because both BLEU and Meteor can obtain relatively good results.

### Visualization of Agreement and PCCs

Coupling Coefficients (CCs) can indirectly reflect the variation of the "agreement", so we visualize the coefficients in each routing iteration at the stage of decoding as shown in Figure 6. In the first iteration, all coupling coefficients are initialized in a uniform distribution, all higher-level capsules are voted by lower-level capsules equally. Then, lower-level capsules are iteratively trained to send more information to the proper higher-level capsule. Figure 6 shows that most of input capsules are tend to vote the $2_{ed}$ feature capsule finally.

Different from the CCs, PCCs show the linear correlation between query and inputs. Initially, the query vector is tiled with the number of feature capsules and is used to calculate the PCCs with each input capsules, as the Figure 7 shows that the color of every column is identical. As the iterative routing begins, each query is updated according to the higher-level capsules. The function of the PPCs is to increase or decrease the coupling coefficients value according to the positive or negative

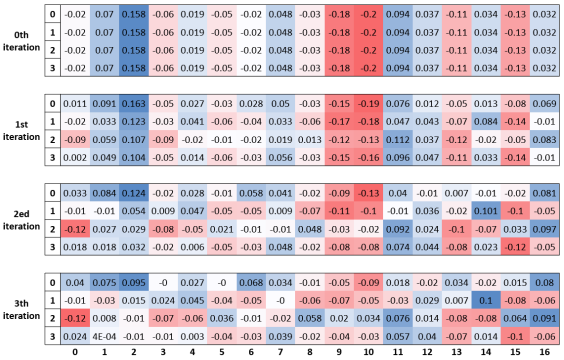


Figure 7: PCCs distribution of a sentence has 17 words as inputs of the QCN and 4 duplicated query. Blue color denotes positive correlation and red means negative. From top to bottom are four heat maps in $0_{th}$ to $3_{th}$ iterations.

value. See Figure 7, we can find that PCCs varies as iteration changes.

## 5 Related Work

### Document-level Machine Translation

Document-level machine translation became a hot research direction in the later stage of statistical machine translation era. Hardmeier and Federico (2010) represented the links between word pairs in the context using a word dependency model for SMT to improve the translation of anaphoric pronouns. Hardmeier et al. (2012, 2013) first proposed a new document-level SMT paradigm that translates whole documents as units. However, in this period, most of the work has not achieved too many compelling results or has been only focused on a part of difficulties.

With the coming of the era of Neural Machine Translation, many works began to focus on Document-level NMT tasks. Xiong et al. (2019) trained a reward teacher to refine the translation quality from a document perspective. Tiedemann and Scherrer (2017) simply concatenated sentences in one document as models' input or output. Jean et al. (2017) used additional context encoder to capture larger-context information. Kuang et al. (2017); Tu et al. (2018) used a cache to memorize most relevant words or features in previous sentences or translations.

Recently, several studies integrated additional modules into the Transformer-based NMTs for modeling contextual information. (Voita et al., 2018; Zhang et al., 2018a), Maruf and Haffari (2018) proposed a document-level NMT using a

memory-networks, and Wang et al. (2017) and Miculicich et al. (2018) integrated hierarchical attention network in RNN-based NMT or Transformer to model the document-level information. Maruf et al. (2019) used the whole document as the contextual information and firslty divided document-level translation tasks into two types: offline and online.

### Capsule Networks

Hinton et al. (2011) proposed the capsule conception to use vector for describing the pose of an object. The dynamic routing algorithm was proposed by Sabour et al. (2017) to build the part-whole relationship through the iterative routing procedure. Hinton et al. (2018) designed a new routing style based on the EM algorithm. Some researchers investigated to apply the capsule network for various tasks. Wang et al. (2018) investigated a novel capsule network with dynamic routing for linear time NMT. Yang et al. (2018) explored capsule networks for text classification with strategies to stabilize the dynamic routing process. Gu and Feng (2019) introduces capsule networks into Transformer to model the relations between different heads in multi-head attention. We specifically investigated dynamic routing algorithms for the document-level NMT.

## 6 Conclusion

We have proposed a novel Query-guided Capsule Network with an improved dynamic routing algorithm for enhancing context modeling for the document-level Neural Machine Translation Model. Experiments on English-German in different domains showed our model significantly outperforms sentence-level NMTs and achieved state-of-the-art performance on two of three datasets, which proved the effectiveness of our approaches.

## Acknowledgments

We thank the anonymous reviewers for their insightful comments. This work was supported by the National Natural Science Foundation of China (NO.61662077, NO.61876174) and National Key R&D Program of China (NO.2017YFE9132900).

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. Wit$^3$: Web inventory of transcribed and translated talks. In *Proceedings of the 16$^{th}$ Conference of the European Association for Machine Translation (EAMT)*, pages 261–268, Trento, Italy.

Shuhao Gu and Yang Feng. 2019. Improving multi-head attention with capsule networks. In *Proceedings of the 8th CCF International Conference on Natural Language Processing and Chinese Computing*.

Christian Hardmeier. 2012. Discourse in statistical machine translation. a survey and a case study. *Discours. Revue de linguistique, psycholinguistique et informatique. A journal of linguistics, psycholinguistics and computational linguistics*, (11).

Christian Hardmeier and Marcello Federico. 2010. Modelling pronominal anaphora in statistical machine translation. In *IWSLT (International Workshop on Spoken Language Translation); Paris, France; December 2nd and 3rd, 2010.*, pages 283–289.

Christian Hardmeier, Joakim Nivre, and Jörg Tiedemann. 2012. Document-wide decoding for phrase-based statistical machine translation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1179–1190, Jeju Island, Korea. Association for Computational Linguistics.

Christian Hardmeier, Sara Stymne, Jörg Tiedemann, and Joakim Nivre. 2013. Docent: A document-level decoder for phrase-based statistical machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 193–198, Sofia, Bulgaria. Association for Computational Linguistics.

Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. 2011. Transforming auto-encoders. In *International Conference on Artificial Neural Networks*, pages 44–51. Springer.

Geoffrey E Hinton, Sara Sabour, and Nicholas Frosst. 2018. Matrix capsules with EM routing. In *International Conference on Learning Representations*.

Sébastien Jean, Stanislas Lauly, Orhan Firat, and Kyunghyun Cho. 2017. Does neural machine translation benefit from larger context? *CoRR*, abs/1704.05135.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the association for computational linguistics companion volume proceedings of the demo and poster sessions*, pages 177–180.

Shaohui Kuang, Deyi Xiong, Weihua Luo, and Guodong Zhou. 2017. Cache-based document-level neural machine translation. *arXiv preprint arXiv:1711.11221*.

Alon Lavie and Abhaya Agarwal. 2007. Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 228–231. Association for Computational Linguistics.

Sameen Maruf and Gholamreza Haffari. 2018. Document context neural machine translation with memory networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1275–1284, Melbourne, Australia. Association for Computational Linguistics.

Sameen Maruf, André FT Martins, and Gholamreza Haffari. 2019. Selective attention for context-aware neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Minneapolis, Minnesota. Association for Computational Linguistics.

Fandong Meng and Jinchao Zhang. 2019. Dtmt: A novel deep transition architecture for neural machine translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 224–231.

Thomas Meyer and Bonnie Webber. 2013. Implicitation of discourse connectives in (machine) translation. In *Proceedings of the Workshop on Discourse in Machine Translation*, pages 19–26.

Lesly Miculicich, Dhananjay Ram, Nikolaos Pappas, and James Henderson. 2018. Document-level neural machine translation with hierarchical attention networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2947–2954, Brussels, Belgium. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. 2017. Dynamic routing between capsules. In *Advances in neural information processing systems*, pages 3856–3866.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Karin Sim Smith. 2017. On integrating discourse in machine translation. In *Proceedings of the Third Workshop on Discourse in Machine Translation*, pages 110–121, Copenhagen, Denmark. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Jörg Tiedemann and Yves Scherrer. 2017. Neural machine translation with extended context. In *Proceedings of the Third Workshop on Discourse in Machine Translation*, pages 82–92, Copenhagen, Denmark. Association for Computational Linguistics.

Zhaopeng Tu, Yang Liu, Shuming Shi, and Tong Zhang. 2018. Learning to remember translation history with a continuous cache. *Transactions of the Association for Computational Linguistics*, 6:407–420.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Elena Voita, Pavel Serdyukov, Rico Sennrich, and Ivan Titov. 2018. Context-aware neural machine translation learns anaphora resolution. In *ACL*.

Longyue Wang, Zhaopeng Tu, Andy Way, and Qun Liu. 2017. Exploiting cross-sentence context for neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2826–2831, Copenhagen, Denmark. Association for Computational Linguistics.

Mingxuan Wang, Jun Xie, Zhixing Tan, Jinsong Su, Deyi Xiong, and Chao Bian. 2018. Towards linear time neural machine translation with capsule networks. *CoRR*, abs/1811.00287.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between

human and machine translation. *arXiv preprint arXiv:1609.08144*.

Liqiang Xiao, Honglun Zhang, Wenqing Chen, Yongkun Wang, and Yaohui Jin. 2018. Mcapsnet: Capsule network for text with multi-task learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4565–4574.

Hao Xiong, Zhongjun He, Hua Wu, and Haifeng Wang. 2019. Modeling coherence for discourse neural machine translation. *CoRR*, abs/1811.05683.

Min Yang, Wei Zhao, Jianbo Ye, Zeyang Lei, Zhou Zhao, and Soufei Zhang. 2018. Investigating capsule networks with dynamic routing for text classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3110–3119, Brussels, Belgium. Association for Computational Linguistics.

Jiacheng Zhang, Huanbo Luan, Maosong Sun, Feifei Zhai, Jingfang Xu, Min Zhang, and Yang Liu. 2018a. Improving the transformer translation model with document-level context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 533–542, Brussels, Belgium. Association for Computational Linguistics.

Wen Zhang, Yang Feng, Fandong Meng, Di You, and Qun Liu. 2019. Bridging the gap between training and inference for neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4334–4343. Association for Computational Linguistics.

Wen Zhang, Jiawei Hu, Yang Feng, and Qun Liu. 2018b. Refining source representations with relation networks for neural machine translation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1292–1303.

# Simple, Scalable Adaptation for Neural Machine Translation

**Ankur Bapna** **Orhan Firat**
Google AI
{ankurbpn,orhanf}@google.com

## Abstract

Fine-tuning pre-trained Neural Machine Translation (NMT) models is the dominant approach for adapting to new languages and domains. However, fine-tuning requires adapting and maintaining a separate model for each target task. We propose a simple yet efficient approach for adaptation in NMT. Our proposed approach consists of injecting tiny task specific adapter layers into a pre-trained model. These lightweight adapters, with just a small fraction of the original model size, adapt the model to multiple individual tasks simultaneously.

We evaluate our approach on two tasks: (i) Domain Adaptation and (ii) Massively Multilingual NMT. Experiments on domain adaptation demonstrate that our proposed approach is on par with full fine-tuning on various domains, dataset sizes and model capacities. On a massively multilingual dataset of 103 languages, our adaptation approach bridges the gap between individual bilingual models and one massively multilingual model for most language pairs, paving the way towards universal machine translation.

## 1 Introduction

Recent developments in deep learning have led to significantly improved quality on Neural Machine Translation (NMT) (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Bahdanau et al., 2014; Vaswani et al., 2017). While NMT performance on sentence level translation for high resource languages seems to be dramatically improved (Wu et al., 2016; Hassan et al., 2018), performance on out-of-domain data or low resource languages, still remains pretty poor (Duh et al., 2013; Koehn and Knowles, 2017; Farajian et al., 2017; Dong et al., 2015; Zoph et al., 2016). This has generated significant interest in adaptation approaches, that can leverage the huge amounts of parallel data available for high resource languages, to improve translation performance on low resource tasks. In this work we focus on two adaptation tasks: (i) Domain Adaptation, to improve the performance of in-domain translation by leveraging out-of-domain datasets (Luong and Manning, 2015; Freitag and Al-Onaizan, 2016), and (ii) Multilingual NMT, to improve the translation quality on low resource languages via co-training with similar languages (Dong et al., 2015; Firat et al., 2016; Johnson et al., 2016; Zoph et al., 2016; Neubig and Hu, 2018; Aharoni et al., 2019; Arivazhagan et al., 2019).

While several approaches have been explored in literature (Chu and Wang, 2018), full fine-tuning of model parameters remains the dominant approach for adapting to new domains and languages (Luong and Manning, 2015; Neubig and Hu, 2018). However, fine-tuning requires training and maintaining a separate model for every language, for every domain. As the number of languages and domains grow, training, maintaining and serving a separate model for every task becomes infeasible. This is further exacerbated by the increasing model capacity of state-of-the-art NMT models (Shazeer et al., 2018; Bapna et al., 2018; Huang et al., 2018); full fine-tuning is just too parameter inefficient. In addition to the growing number of models, fine-tuning requires very careful hyper-parameter tuning (eg. learning rate, regularization knobs etc.) during adaptation, and is prone to rapid over-fitting (Sennrich et al., 2016; Miceli Barone et al., 2017). This sensitivity to hyper-parameters and over-fitting to the adaptation corpus become worse in the setting of high capacity models.

These weaknesses beckon the need for parameter-efficient, scalable and hyper-parameter insensitive approaches for adaptation. The ideal adaptation approach should also offer the flexi-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1538–1548, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

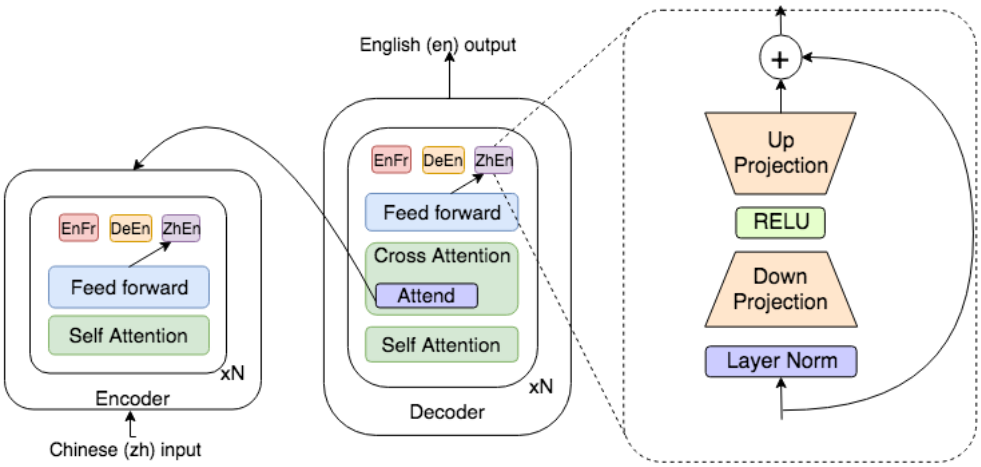


Figure 1: Diagrams depicting (i) Left: the proposed layout of a Transformer enhanced with language specific adapters (ii) Right: the architecture of each residual adapter layer. While the figure depicts use of adapters for multilingual NMT, the same formulation can be used for domain adaptation.

bility to adapt to tasks of varying complexity and adaptation corpus sizes, within a single model.

In this work we propose using light-weight adapter layers, which are transplanted between the layers of a pre-trained network and fine-tuned on the adaptation corpus. Adapting only the light-weight layers enables our approach to be parameter efficient, and eases the scalability of the approach to large models. The capacity of these adapters can be adjusted to match the requirements of the target task, making them suitable for a variety of adaptation tasks. By separating the parameters of the original network and each adaptation task, our approach circumvents catastrophic interference (McCloskey and Cohen, 1989) with the original model parameters, and allows us to simultaneously adapt a single model to multiple domains and languages, while retaining the quality on the source languages and domains.

We make three major contributions in this work: (i) we propose a formulation of adapter layers for NMT adaptation that enables us to tune their capacity according to the target task complexity and corpus size, (ii) we evaluate our approach on domain adaptation, and demonstrate that light-weight adapters match the performance of full fine-tuning based adaptation at a fraction of the per-domain parameter cost, and (iii) we use adapters to train a massively multilingual model on 103 languages, and demonstrate that it is possible to train a single model that significantly improves transfer performance on low resource languages, without huge regression on high resource language pairs.

By demonstrating the effectiveness of adapters on domain adaptation and massively multilingual translation, we make progress towards a flexible universal translation model for all languages and domains.

## 2 Related Work

Several approaches have been proposed in recent literature that try to address the shortcomings of full fine-tuning when applied to domain adaptation (Chu and Wang, 2018). Michel and Neubig (2018) proposed a space efficient approach to adaptation that introduces domain-specific biases to the output vocabulary, enabling extreme personalization in settings where small amounts of data are available for a lot of different domains. Thompson et al. (2018) fine-tune selected components of the base model architecture, in order to determine how much fine-tuning each component contributes to the final adaptation performance. Wuebker et al. (2018) propose introducing sparse offsets from the base model parameters for every domain, reducing the memory complexity of loading and unloading domain specific parameters in real world settings. Bapna and Firat (2019) train the base model to utilize neighboring samples from the training set, enabling the model to adapt to new domains without the need for additional parameter updates. Learning Hidden Unit Contribution (LHUC) (Vilar, 2018) is perhaps closest to our work in spirit.

They introduce domain specific gates that control the contribution of hidden units feeding into the next layer. However, they introduce a limited amount of per-domain capacity which doesn't scale well when a lot of domain specific data is available.

Residual Adapters were first introduced for adapting vision models in Rebuffi et al. (2017), but their formulation used a single projection layer, without any tunable hyper-parameters that could be used to adjust capacity based on the target domain. Houlsby et al. (2019) utilized a new formulation of adapters to adapt BERT (Devlin et al., 2018) to multiple tasks simultaneously. Our formulation of adapters is motivated by theirs, but differs in a few respects. Houlsby et al. (2019) introduce adapters after every sub-layer (self-attention, feed-forward) within a transformer layer, and re-train existing layer normalization parameters for every new domain. We simplify this formulation by leaving the parameters frozen, and introducing new layer normalization parameters for every task, essentially mimic-ing the structure of the transformer feed-forward layer.

## 3 Approach

Our approach consists of two phases: (i) Training a generic base model, and (ii) adapting it to new tasks with added small network modules. We first take a standard NMT model which is trained on a large source corpus. Following convergence,[1] all model parameters are frozen, preserving the information learned during this pre-training phase. Next, per-task light-weight adapter layers (see Fig. 1 right pane) are introduced after every layer in the encoder and the decoder (see Fig. 1 left pane). We fine-tune the parameters of these task-specific adapters on the adaptation corpus. This procedure can be followed for every additional task, allowing us to train a single model for all tasks simultaneously, with a small set of task-specific adapters.

**Adapter Modules** Our design principles for adapter modules are simplicity and flexibility. We propose a simple single hidden-layer feed-forward network formulation for adapters, with a non-linear activation function between the two projection layers. The inner dimension of these two projections is the only knob to tune. This allows us to adjust the capacity of the adapter module easily, depending on the complexity of the target task or domain. Additionally, we normalize the input of the adapters, in order to make the module plug-able into any part of the base network, irrespective of the variations in the activation patterns/distributions. This parametrized normalization layer allows the module to learn the activation pattern of the layer it's injected into. Finally, to allow the adapter module to represent a no-op if necessary, we wrap it with a residual connection.

**Formulation** Let $z_i$ be the output of the $i$-th layer, of dimension $d$. We first apply layer-normalization (Ba et al., 2016) to the inputs of the adapter corresponding to task $T$.

$$\tilde{z}_i^T = LN_T(z_i). \tag{1}$$

This is followed by a projection layer of dimension $b$. The dimension of the projection can be tuned based on the task complexity and the size of the adaptation corpus. This now allows us to make it a bottleneck layer for compression, or over-parametrize it with a dimension larger than the input dimension.[2]

$$h_i^T = relu(W_{bd}^T \tilde{z}_i^T). \tag{2}$$

Lastly, the inner representation is projection back to the input dimension $d$, and combined with a residual connection (He et al., 2015):

$$x_i^T = W_{db}^T h_i^T + z_i. \tag{3}$$

Our proposed formulation for adapters, and their incorporation into Transformers (Vaswani et al., 2017) is illustrated in Figure 1. This self-contained adapter module can be injected between any two layers of the network, without disrupting the original operation.

## 4 Domain Adaptation

We first compare the adaptation performance of the light-weight residual adapters against full fine-tuning and LHUC (Vilar, 2018) on a large scale English-French domain adaptation task.

[1] We leave the methodology for defining convergence task specific, e.g. early stopping on validation set accuracy or number of training steps.

[2] Following the wiring choices of Transformer feed-forward network (Vaswani et al., 2017).

### 4.1 Dataset

We use the WMT En-Fr training set (36M pairs) as our out-of-domain (source) training corpus. NMT models trained on WMT are then adapted to the IWSLT'15 En-Fr corpus, consisting of 237k sentence pairs. We also evaluate adaptation performance on the JRC-Acquis dataset [3], which is an extremely narrow domain dataset containing 797k sentence pairs in the training split. For IWSLT, we use the test corpora from 2012-14 for validation, and the test corpus from 2015 as the test set. For JRC-Acquis the test and validation set contain 6574 and 5121 sentence pairs respectively. We also evaluate the translation performance of the non-adapted base model on newstest-2014.

| Dataset | Base | FT | LHUC | Adap. |
|---|---|---|---|---|
| WMT'14 | 42.80 | - | - | - |
| IWSLT'15 | 41.33 | **44.59** | 43.33 | **44.63** |
| JRC | 54.60 | **64.13** | 57.10 | 63.48 |

Table 1: Domain adaptation performance with different adaptation strategies. Base refers to the baseline NMT model trained on the WMT'14 En-Fr training corpus. FT refers to the fine-tuning upper bound, adapting all the model parameters by incrementally training on in-domain training data. LHUC adds additional task-specific gating parameters to the pre-trained model, which are trained on the in-domain data, as described in Vilar (2018). Adap. is the proposed adaptation approach, adding domain specific adapter layers trained on the in-domain training corpus.

### 4.2 Using Adapters for Domain Adaptation

When using adapters for domain adaptation, we follow the following two step approach:

- Pre-training: Pre-train the NMT model on a large open-domain corpus. Freeze all the parameters of this pre-trained model.
- Adaptation: Inject a set of domain-specific adapter layers for every target domain. These adapters are then fine-tuned to maximize performance on the corresponding domains. This step can be applied any time a new domain is added to the model.

As discussed in Section 5.2, we follow a slightly different approach when using adapters for multilingual NMT.

[3] http://opus.nlpl.eu/JRC-Acquis.php

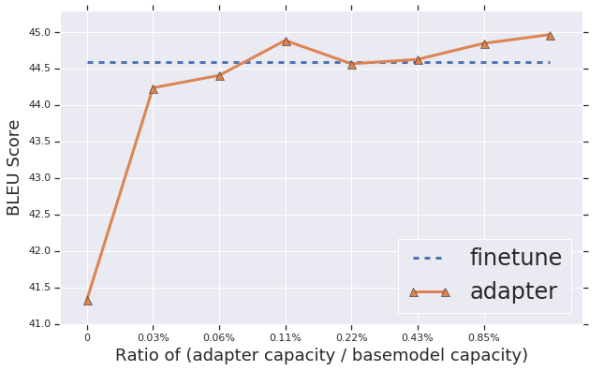


Figure 2: IWSLT Adaptation performance vs adapter capacity in terms of percentage of the base model's capacity. The range of model capacity plotted here corresponds to adapter bottleneck dimensions of 4, 8...256 respectively.

### 4.3 Models and Hyper-parameters

We use a larger version of Transformer Big containing 375M parameters as our base model. Our model is identical to Vaswani et al. (2017), having 6 encoder and decoder layers (12 in total), except that we use hidden layers of size 8192 instead of 4096, and a learning rate schedule of (3.0, 40K)[4], following Chen et al. (2018).

For full fine-tuning, we continue training on the in-domain corpus without resetting the optimizer accumulators or the learning rate schedule. This allows us to fine-tune gradually, avoiding rapidly over-fitting to the target domain. We also conducted fine-tuning experiments with SGD, but report results with the approach described above, based on better performance on the validation sets. When adapting with LHUC or lightweight adapters, we train using the same learning rate schedule and optimizer used during pre-training, but restart from the 0-th step, resetting the optimizer accumulators. BLEU scores are computed on the checkpoint with the best validation performance, on tokenized, true-cased output and references using *multi-bleu.perl* from Moses. All our experiments were performed using the open source Tensorflow Lingvo (Shen et al., 2019) framework.

### 4.4 Results and Analysis

The results of our adaptation experiments are documented in Table 1. On both, IWSLT and JRC-

[4] (3.0, 40K) schedule is the shorthand for a learning rate of 3.0, with 40K warm-up steps for the schedule, which is decayed with the inverse square root of the number of training steps after warm-up.

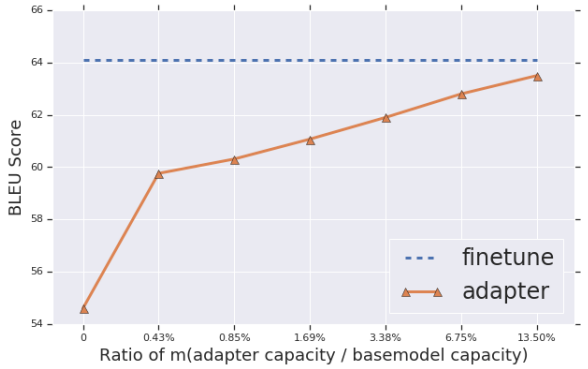


Figure 3: JRC-Acquis Adaptation performance vs adapter capacity in terms of percentage of the base model's capacity. The range of model capacity plotted here corresponds to adapter bottleneck dimensions of 64, 128...2048 respectively.

Acquis, full model fine-tuning (Full-FT columns in Table 1) on in-domain data significantly improves translation performance compared to the base, non-adapted Transformer Big by a huge margin, 3 BLEU points for IWSLT and 9 BLEU points for JRC. LHUC also improves performance over the base model, but lags behind a fully fine-tuned model for both domains and model capacities.

On IWSLT, adapters match the performance of the fine-tuning upper bound within error margins, while adding less than 0.11% of the original model parameters. On JRC-Acquis adapters recover around 90% of fine-tuning improvements without updating any existing parameters, while adding around 13.5% additional parameters.

To demonstrate the flexibility of our approach, we quantify the trade-off between adapter capacity and adaptation performance on both IWSLT and JRC-Acquis. In Figures 2 and 3, we plot the adaptation performance on IWSLT and JRC-Acquis respectively, while varying adapter capacity. On IWSLT, we notice that residual adapters reach within 0.5 BLEU of the full fine-tuning upper bound with just 0.03% of the model capacity, corresponding to a hidden dimension of size 4. By increasing capacity further we were able to improve over the full fine-tuning baseline by around 0.5 BLEU. On the other hand, on JRC-Acquis, adapter capacity had to be increased up to 13.5% of the total model capacity, corresponding to a hidden dimension of size 2048, before we were within 0.5 BLEU of the full fine-tuning performance. This highlights a key strength of the approach: by varying adapter capacity it is possible to adapt the same model to domains of varying complexity and amounts of data.

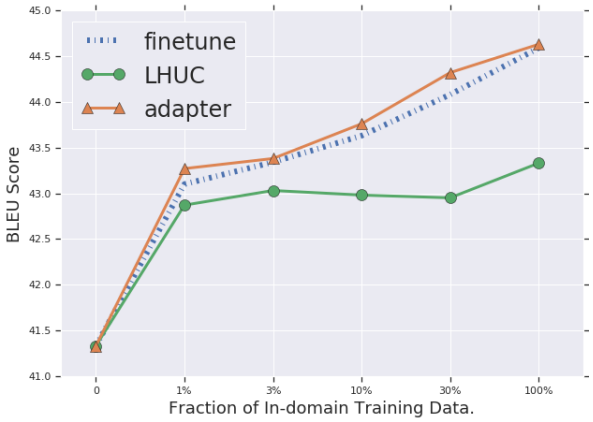


Figure 4: IWSLT Adaptation performance vs fraction of the in-domain training corpus used for adaptation. The blue solid line plots the performance of fine-tuning. The red dotted line tracks the performance of LHUC, while the yellow dashed line tracks the performance of the proposed residual adapter based approach with varying amounts of training data.

To evaluate the effectiveness of adapters when adapting with small in-domain corpora, we further compare the performance of adapters with fine-tuning on varying amounts of training data. In Figure 6 we plot the adaptation performance on IWSLT, when using different fractions of the training corpus for adaptation. While LHUC is competitive with full fine-tuning and light-weight adapters for extremely small fractions, the lack of capacity limits the applicability of the approach when larger quantities of adaptation data are available. On the other hand, by tuning the capacity of adapters to match the requirements for the adaptation corpus size, we are able to match and outperform fine-tuning on almost all evaluated datapoints.

In order to monitor the learning behavior of light-weight adapters, we compare the validation BLEU scores during the course of the fine-tuning process. Figure 5 illustrates the comparison of the two approaches, full fine-tuning and light-weight adapters. We notice that for a reasonably small adapter size, adapter performance gradually converges to its peak and stays steady, with almost no over-fitting for a long enough period, easing final model selection. On the other hand, with full fine-tuning, optimal model selection becomes challenging due to rapid over-fitting on the adaptation corpus. This, in fact, can be remedied by care-

fully tuning the learning rate (and/or batch size) during adaptation, but is not trivial and needs to be done individually for every different domain, model and corpus size, favoring the simplicity of our proposed approach.

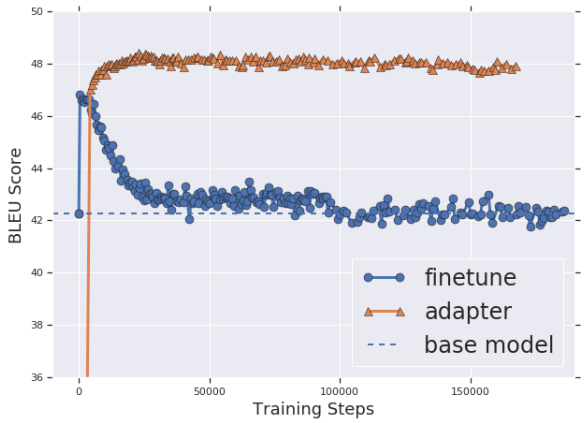


Figure 5: IWSLT dev performance vs number of in-domain adaptation steps when adapting with fine-tuning vs adapters.

## 5 Massively Multilingual Machine Translation

To stress test our adapters based approach, we apply this to a massively multilingual translation task on a real world dataset (Arivazhagan et al., 2019). Most previous literature in multilingual NMT focuses on improving the performance of low resource languages (Zoph et al., 2016; Firat et al., 2016; Neubig and Hu, 2018), often ignoring the source language performance of the adapted model. However, the goal of our work is to enable training a single model for all language pairs, in order to get benefits of transfer on low resource language pairs, without losing performance in the high resource setting.

### 5.1 Dataset

To highlight the flexibility of an adapters based approach, we study multilingual NMT on a massive scale, using a corpus generated by crawling and extracting parallel sentences from the web. Our corpus contains parallel documents for 102 languages, to and from English, containing a total of 25 billion sentence pairs (Arivazhagan et al., 2019).[5] The number of parallel sentences per language in our corpus ranges from around 10s of thousands to almost 2 billion. Figure 6 illustrates the data distribution across languages for all 102 languages studied in this paper.

### 5.2 Using Adapters for multilingual NMT

Our approach to using adapters for multilingual NMT diverges from domain adaptation, owing to the differences in the two tasks.

In the domain adaptation setting, while the input and output distributions of the adaptation domain might differ from that of the base, the set of inputs and outputs is pretty much the same. In mathematical terms, both the base and adaptation domain distributions, $D_S$ and $D_T$ respectively, are defined on the same support set $\{X, Y\}$.

On the other hand, in multilingual NMT, the support sets of different language pairs have very little overlap. In this setting, adapting a model to a new language pair without learning the embeddings and softmax parameters (which correspond to the input and output support sets) would be an extremely difficult task. Following the approach used for domain adaptation in Section 4.2 might not be possible here. We modify our approach to expand the input-output distribution of our initial pre-trained model to all language pairs we are interested in supporting, i.e. we can't add any new language pairs to the model during adaptation, but we use the adaptation stage to improve performance on languages learnt during pre-training.

For multilingual NMT, we follow the following two step approach:

- Global training: Train a fully shared model on all language pairs, with the goal of maximizing transfer to low resource languages.
- Refinement: Fine-tuning language pair specific adapters for all high resource languages, to recover lost performance during step 1. This step can only be applied for language pairs learned during global training.

### 5.3 Models and Hyper-parameters

We first train dedicated bilingual models on all language pairs to ground our multilingual analyses. We perform all our experiments with variants of the Transformer architecture (Vaswani et al., 2017). For most bilingual experiments, we use a larger version of Transformer Big containing 375M parameters (Chen et al., 2018), and a shared source-target sentence-piece model (SPM) (Kudo

[5] Limited to approximately this amount for experimentation.

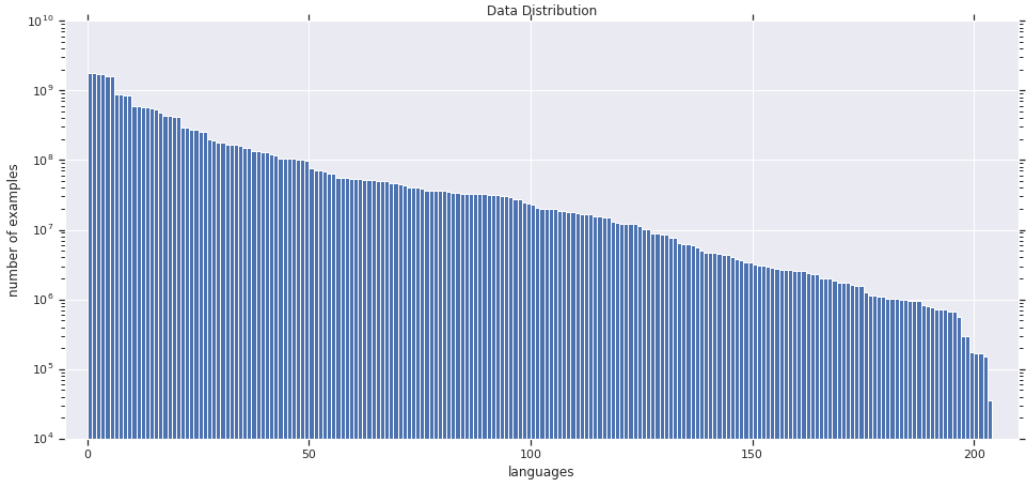


Figure 6: Per language pair data distribution of the dataset used for our multilingual experiments (for 102 languages, 204 language pairs to and from English). The y-axis depicts the number of training examples available per language pair on a logarithmic scale. Dataset sizes range from the order of $10^4$ for the lowest resource language pairs to the order of $10^9$ for the largest.

and Richardson, 2018) vocabulary with 32k tokens. We tune different values of dropout (Srivastava et al., 2014), depending on the dataset size for each language pair. For most medium and low resource languages we also experiment with Transformer Base. All our models are trained with Adafactor (Shazeer and Stern, 2018) with momentum factorization, a learning rate schedule of (3.0, 40K), and a per-parameter norm clipping threshold of 1.0. For Transformer Base models, we use a learning rate schedule of (2.0, 8K). BLEU scores are computed on the checkpoint with the best validation performance, on true-cased output and references.[6]

We now describe our approach for training the multilingual models. Due to the large imbalance in our training dataset (Figure 6), we first design a sampling strategy to simultaneously train a single model on all 204 language pairs. Sampling directly from the data distribution results in good performance on high resource languages, but low resource languages get starved. Sampling equally from all language pairs results in huge boost in low resource translation performance, but high resource languages perform significantly worse than their bilingual baselines.

To balance between high and low resource language pairs, we use a temperature based sampling strategy (Arivazhagan et al., 2019). For a given language pair, $l_{12}$, let $D_{l12}$ be the size of the available parallel corpus. Then if we sample from the union of the datasets, the probability of the sample being from language pair $l_{12}$ is $p_{l_{12}} = \frac{D_{l_{12}}}{\Sigma_{l_{12}} D_{l_{12}}}$. We set the probability of our sampled distribution to be proportional to $p_{l_{12}}^{\frac{1}{T}}$, where $T$ is the sampling temperature. Now, $T = 1$ corresponds to true data distribution and $T = 100$ corresponds to an (almost) equal number of samples for each language. We use $T = 5$ for our multilingual model.

We train a single Transformer Big simultaneously on all 204 language pairs (102 languages to and from English), with the same hyper-parameter settings as the bilingual model. However, we use a shared SPM vocab with 64K tokens, generated using the same sampling distribution ($T = 5$) used during training. We additionally use character coverage of 0.999995 to ensure our vocab contains most of the alphabets for all 103 languages. Please refer (Arivazhagan et al., 2019) for additional training details for the base multilingual model.

Following global pre-training on all language pairs, we inject and fine-tune language pair specific adapters. The fine-tuning stage is performed separately for each language pair to reduce the device memory needed for the training process. The fine-tuned adapters can then be combined together into a single model after this stage. For fine-tuning, we use the same hyper-parameter used during global pre-training, but reset our optimizer

[6]We used an in-house implementation of mteval-v13a.pl from Moses to evaluate BLEU scores for our multilingual experiments.

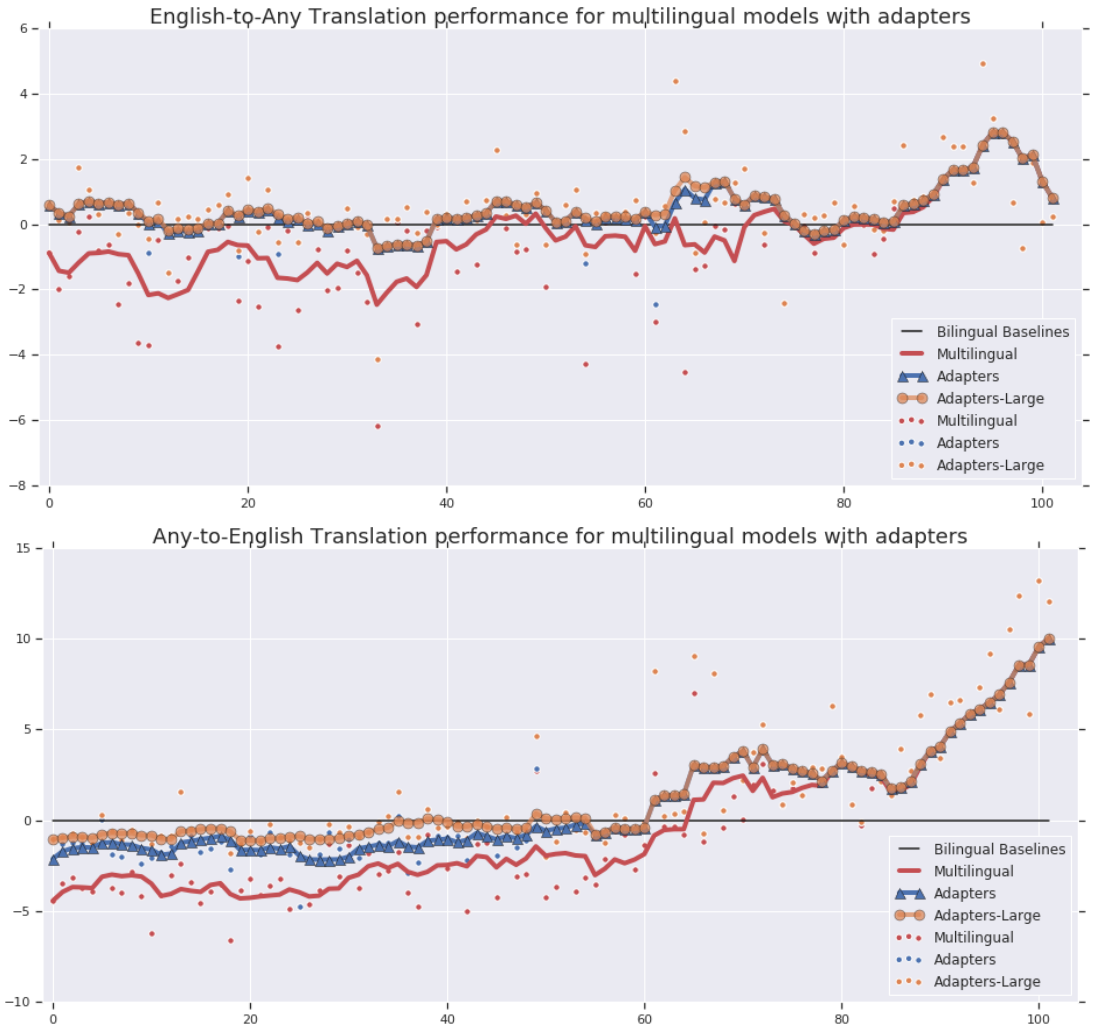


Figure 7: Trendlines depicting translation performance improvement in multilingual models with residual adapters. From left to right, languages are arranged in decreasing order of available training data. y-axis depicts the BLEU score relative to the bilingual baseline trained on the corresponding language pair. The plots correspond to the following models: (1.) Red: Multilingual model trained with sampling temperature, $T = 5$ (2.) Blue: Multilingual model + Small adapters, $b = 2048$ (3.) Pink: Multilingual model + Large adapters, $b = 4096$.
Note: For adapter experiments, we choose the best performance between $b = 0$, $b = 2048$ and $b = 4096$.

accumulators and restart from step 0. For our experiments we use the same bottle-neck dimension, $b = 2048$, for all language pairs. This was meant to reduce the number of experiments given the large number of language pairs in our setup. For language pairs that were worse than their bilingual models after adding adapters with $b = 2048$, we re-run fine-tuning with larger adapters, with $b = 4096$. In an ideal setting, the bottle-neck could be larger for the highest resource languages and $b = 0$ (no adapters) for the smallest languages.

### 5.4 Results and Analysis

We plot the translation quality on different language pairs in Figure 7. As we can see, the multilingual model significantly out-performs the bilingual baselines in the extremely low resource setting. These gains are even more amplified when translating into English, agreeing with previous work in multilingual NMT (Neubig and Hu, 2018; Aharoni et al., 2019). However, owing to the huge training corpus, we observe significant performance deterioration in the high resource languages. We attribute this deterioration to two factors: (i) Languages compete for capacity given the limited model size, and (ii) The model converges much before it trains on significant portions of the high resource datasets.

As is apparent from Figure 7, performance on high and medium resource languages improves

by huge margins after the second stage of training (adapter based refinement). Fine-tuning with adapters allows the model to see larger portions of the training data for high resource languages, and converges faster than training a model from scratch since it only updates a very small fraction of the model parameters (for most language pairs, the second stage converges within 20-50k steps, depending on the corpus size). For high resource languages, especially when translating into English, we observe further performance improvements when increasing adapter size. This again highlights the flexibility of adapters, it is possible to adjust the adapter capacity to match the complexity and resource size of the target task.

While adapters help us bridge most of the gap between bilingual and multilingual models, we still observe a minor regression for high resource languages translating into English, compared to the bilingual baselines. Although it might be possible to reduce this gap further by increasing the adapter size beyond $b = 4096$, there might be more efficient ways to approach this problem, including more expressive network architectures for adapters, joint fine-tuning of adapters and global model parameters, etc. However, we leave these studies to future work.

## 6 Conclusion

In this work, we proposed *light-weight adapters*, a simple yet efficient way for adapting large scale neural machine translation models. Our proposed approach, by injecting small task-specific adapter layers between the frozen base model layers during adaptation, enables the final network to adapt to multiple target tasks simultaneously, without forgetting the original parameters.

We evaluate light-weight adapters on two different adaptation tasks, domain adaptation and multilingual NMT. Experiments support the flexibility and scalability of *light-weight adapters*, (i) yielding comparable or better results when compared with the standard full fine-tuning or bilingual baselines, (ii) without the need for any hyper-parameter tuning across varying adaptation dataset sizes and model capacities.

With a large set of globally shared parameters and small interspersed task-specific layers, adapters allow us to train and adapt a single model for a huge number of languages and domains. We hope that this work would motivate further research into massively multitask and universal translation models.

## Acknowledgments

We would like to thank the Google Translate and Lingvo development teams for their foundational contributions to this project. We would also like to thank Neil Houlsby, Anjuli Kannan and Yonghui Wu for helpful discussions early on in the project, and Wolfgang Macherey, Markus Freitag, Ciprian Chelba, Naveen Arivazhagan, Zhifeng Chen and the anonymous EMNLP reviewers for their insightful comments.

## References

Roee Aharoni, Melvin Johnson, and Orhan Firat. 2019. Massively multilingual neural machine translation. *CoRR*, abs/1903.00089.

Naveen Arivazhagan, Ankur Bapna, Orhan Firat, Dmitry Lepikhin, Melvin Johnson, Maxim Krikun, Mia Xu Chen, Yuan Cao, George Foster, Colin Cherry, Wolfgang Macherey, Zhifeng Chen, and Yonghui Wu. 2019. Massively multilingual neural machine translation in the wild: Findings and challenges.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Ankur Bapna, Mia Xu Chen, Orhan Firat, Yuan Cao, and Yonghui Wu. 2018. Training deeper neural machine translation models with transparent attention. *arXiv preprint arXiv:1808.07561*.

Ankur Bapna and Orhan Firat. 2019. Non-parametric adaptation for neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1921–1931.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Niki Parmar, Mike Schuster, Zhifeng Chen, et al. 2018. The best of both worlds: Combining recent advances in neural machine translation. *arXiv preprint arXiv:1804.09849*.

Chenhui Chu and Rui Wang. 2018. A survey of domain adaptation for neural machine translation. *arXiv preprint arXiv:1806.00258*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 1723–1732.

Kevin Duh, Graham Neubig, Katsuhito Sudoh, and Hajime Tsukada. 2013. Adaptation data selection using neural language models: Experiments in machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 678–683.

M Amin Farajian, Marco Turchi, Matteo Negri, and Marcello Federico. 2017. Multi-domain neural machine translation through unsupervised adaptation. In *Proceedings of the Second Conference on Machine Translation*, pages 127–137.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. *arXiv preprint arXiv:1601.01073*.

Markus Freitag and Yaser Al-Onaizan. 2016. Fast domain adaptation for neural machine translation.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, et al. 2018. Achieving human parity on automatic chinese to english news translation. *arXiv preprint arXiv:1803.05567*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp.

Yanping Huang, Yonglong Cheng, Dehao Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, and Zhifeng Chen. 2018. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *arXiv preprint arXiv:1811.06965*.

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2016. Google's multilingual neural machine translation system: enabling zero-shot translation. *arXiv preprint arXiv:1611.04558*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington, USA. Association for Computational Linguistics.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. *arXiv preprint arXiv:1706.03872*.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.

Minh-Thang Luong and Christopher D. Manning. 2015. Stanford neural machine translation systems for spoken language domain. In *International Workshop on Spoken Language Translation*, Da Nang, Vietnam.

Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier.

Antonio Valerio Miceli Barone, Barry Haddow, Ulrich Germann, and Rico Sennrich. 2017. Regularization techniques for fine-tuning in neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1489–1494. Association for Computational Linguistics.

Paul Michel and Graham Neubig. 2018. Extreme adaptation for personalized neural machine translation.

Graham Neubig and Junjie Hu. 2018. Rapid adaptation of neural machine translation to new languages. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 875–880. Association for Computational Linguistics.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2017. Learning multiple visual domains with residual adapters. In *Advances in Neural Information Processing Systems*, pages 506–516.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Noam Shazeer, Youlong Cheng, Niki Parmar, Dustin Tran, Ashish Vaswani, Penporn Koanantakool, Peter Hawkins, HyoukJoong Lee, Mingsheng Hong, Cliff Young, Ryan Sepassi, and Blake Hechtman. 2018. Mesh-tensorflow: Deep learning for supercomputers.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. *arXiv preprint arXiv:1804.04235*.

Jonathan Shen, Patrick Nguyen, Yonghui Wu, Zhifeng Chen, Mia X. Chen, Ye Jia, Anjuli Kannan, Tara Sainath, Yuan Cao, Chung-Cheng Chiu, Yanzhang He, Jan Chorowski, Smit Hinsu, Stella Laurenzo, James Qin, Orhan Firat, Wolfgang Macherey, Suyog Gupta, Ankur Bapna, Shuyuan Zhang, Ruoming Pang, Ron J. Weiss, Rohit Prabhavalkar, Qiao Liang, Benoit Jacob, Bowen Liang, HyoukJoong Lee, Ciprian Chelba, Sbastien Jean, Bo Li, Melvin Johnson, Rohan Anil, Rajat Tibrewal, Xiaobing Liu, Akiko Eriguchi, Navdeep Jaitly, Naveen Ari, Colin Cherry, Parisa Haghani, Otavio Good, Youlong Cheng, Raziel Alvarez, Isaac Caswell, Wei-Ning Hsu, Zongheng Yang, Kuan-Chieh Wang, Ekaterina Gonina, Katrin Tomanek, Ben Vanik, Zelin Wu, Llion Jones, Mike Schuster, Yanping Huang, Dehao Chen, Kazuki Irie, George Foster, John Richardson, Klaus Macherey, Antoine Bruguier, Heiga Zen, Colin Raffel, Shankar Kumar, Kanishka Rao, David Rybach, Matthew Murray, Vijayaditya Peddinti, Maxim Krikun, Michiel A. U. Bacchiani, Thomas B. Jablin, Rob Suderman, Ian Williams, Benjamin Lee, Deepti Bhatia, Justin Carlson, Semih Yavuz, Yu Zhang, Ian McGraw, Max Galkin, Qi Ge, Golan Pundak, Chad Whipkey, Todd Wang, Uri Alon, Dmitry Lepikhin, Ye Tian, Sara Sabour, William Chan, Shubham Toshniwal, Baohua Liao, Michael Nirschl, and Pat Rondon. 2019. Lingvo: a modular and scalable framework for sequence-to-sequence modeling. *arXiv preprint arXiv:1902.08295*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Brian Thompson, Huda Khayrallah, Antonios Anastasopoulos, Arya D. McCarthy, Kevin Duh, Rebecca Marvin, Paul McNamee, Jeremy Gwinnup, Tim Anderson, and Philipp Koehn. 2018. Freezing subnetworks to analyze domain adaptation in neural machine translation.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

David Vilar. 2018. Learning hidden unit contribution for adapting neural machine translation models. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 500–505.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Joern Wuebker, Patrick Simianer, and John DeNero. 2018. Compact personalized models for neural machine translation. *arXiv preprint arXiv:1811.01990*.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1568–1575.

# Controlling Text Complexity in Neural Machine Translation

**Sweta Agrawal**
Department of Computer Science
University of Maryland
sweagraw@cs.umd.edu

**Marine Carpuat**
Department of Computer Science
University of Maryland
marine@cs.umd.edu

## Abstract

This work introduces a machine translation task where the output is aimed at audiences of different levels of target language proficiency. We collect a high quality dataset of news articles available in English and Spanish, written for diverse grade levels and propose a method to align segments across comparable bilingual articles. The resulting dataset makes it possible to train multi-task sequence-to-sequence models that translate Spanish into English targeted at an easier reading grade level than the original Spanish. We show that these multi-task models outperform pipeline approaches that translate and simplify text independently.

## 1 Introduction

Generating text at the right level of complexity can make machine translation (MT) more useful for a wide range of users. As Xu et al. (2015) note, simplifying text makes it possible to develop reading aids for people with low-literacy (Watanabe et al., 2009; De Belder and Moens, 2010), for non-native speakers and language learners (Petersen and Ostendorf, 2007; Allen, 2009), for people who suffer from language impairments (Carroll et al., 1999; Canning et al., 2000; Inui et al., 2003), and for readers lacking expert knowledge of the topic discussed (Elhadad and Sutaria, 2007; Siddharthan and Katsos, 2010). Such readers would also benefit from MT output that is better targeted to their needs by being easier to read than the original.

Prior work on text complexity has focused on simplifying input text in one language, primarily English (Chandrasekar et al., 1996; Coster and Kauchak, 2011; Siddharthan, 2014; Xu et al., 2015; Zhang and Lapata, 2017; Scarton and Specia, 2018; Kriz et al., 2019; Nishihara et al., 2019). Simplification has been used to improve MT by restructuring complex sentences into shorter and simpler segments that are easier to translate (Gerber and Hovy, 1998; Štajner and Popovic, 2016; Hasler et al., 2017). Contemporaneously to our work, Marchisio et al. (2019) show that tagging the English side of parallel corpora with automatic readability scores can help translate the same input in a simpler or more complex form. Our work shares the goal of controlling translation complexity, but considers a broader range of reading grade levels and simplification operations grounded in professionally edited text simplification corpora.

Building a model for this task ideally requires rich annotation for evaluation and supervised training that is not available in bilingual parallel corpora typically used in MT. Controlling the complexity of Spanish-English translation ideally requires examples of Spanish sentences paired with several English translations that span a range of complexity levels. We collect such a dataset of English-Spanish segment pairs from the Newsela website, which provides professionally edited simplifications and translations. By contrast with MT parallel corpora, the English and Spanish translations at different grade levels are only comparable. They differ in length and sentence structure, reflecting complex syntactic and lexical simplification operations.

We adopt a multi-task approach to control complexity in neural MT and evaluate it on complexity-controlled Spanish-English translation. Our extensive empirical study shows that multitask models produce better and simpler translations than pipelines of independent translation and simplification models. We then analyze the strengths and weaknesses of multitask models, focusing on the degree to which they match the target complexity, and the impact of training data types and reading grade level annotation.[1]

[1]Researchers can request the bilingual Newsela data



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1549–1564,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

## 2 Background

Given corpora of parallel complex-simple segments, text simplification can naturally be framed as a translation task, borrowing and adapting model architectures originally designed for MT. Xu et al. (2016) provide a thorough study of statistical MT techniques for English text simplification, and introduce novel objectives to measure simplification quality. Interestingly, they indirectly make use of parallel translation corpora to derive simplification paraphrasing rules by bilingual pivoting (Callison-Burch, 2007). Zhang and Lapata (2017) train sequence-to-sequence models to translate from complex to simple English using reinforcement learning to directly optimize the metrics that evaluate complexity (SARI) and fluency and adequacy (BLEU). Scarton and Specia (2018) address the task of producing text of varying levels of complexity for different target audiences. They show that neural sequence-to-sequence models informed by target-complexity tokens inserted in the input sequence perform well on this task. While the vast majority of text simplification work has focused on English, Spanish (Štajner et al., 2015), Italian (Brunato et al., 2016; Aprosio et al., 2019) and German (Klaper et al., 2013) have also received some attention.

While most MT approaches only indirectly capture style properties (e.g., via domain adaptation), a growing number of studies share the goal of considering source texts and their translation in their pragmatic context. Mirkin and Meunier (2015) introduce personalized MT. Rabinovich et al. (2016) and Vanmassenhove et al. (2018) suggest that the gender of the author is implicitly marked in the source text and that dedicated statistical and neural systems better preserve gender traits in MT output. Neural MT has enabled more flexible ways to control stylistic properties of MT output. Sennrich et al. (2016) first propose to append a special token to the source that neural MT models can attend to and to select the formal (Sie) or informal (du) version of second person pronouns when translating into German. Niu et al. (2018) show that multi-task models can jointly translate between languages and styles, producing formal and informal translations with broader lexical and phrasal changes than the local pronoun changes in Sennrich et al. (2016). Closest to our goal, Marchisio et al. (2019) address the task of producing either simple or complex translations of the same input, using automatic readability scoring of parallel corpora. They show that training distinct decoders for simple and complex language allows better complexity control than using the target complexity as a side-constraint. By contrast, our approach exploits text simplification corpora for richer supervision for both training and evaluation.

## 3 A Multi-Task Approach to Complexity Controlled MT

**Task** We define **complexity controlled MT** as a task that takes two inputs: an input language segment $s_i$ and a target complexity $c$. The goal is to produce a translation $s_o$ in the output language that has complexity $c$. For instance, given input Spanish sentences in Table 1, complexity controlled MT aims to produce English translations at a specific level of complexity, which might differ from the complexity of the original Spanish.

**Model** We model $P(s_o|s_i, c; \theta)$ as a neural encoder-decoder with attention (Bahdanau et al., 2015). This architecture has been used successfully for the two related tasks of text simplification (Wang et al., 2016; Zhang and Lapata, 2017; Nisioi et al., 2017; Scarton and Specia, 2018) and machine translation (Bahdanau et al., 2015). The encoder constructs hidden representation for each word in the input sequence, while the decoder generates the target sequence, conditioned on hidden source representations. We hypothesize that training a single encoder-decoder model to perform the two distinct tasks of machine translation and text simplification will yield a model that can perform complexity controlled MT. We adopt the multi-task framework proposed by Johnson et al. (2016) to train multilingual neural MT systems.

**Representing target complexity** Target complexity $c$ can be incorporated in sequence-to-sequence models as a special token appended to the beginning of the input sequence, which acts as a side constraint. The encoder encodes this token in its hidden states as any other vocabulary token, and the decoder can attend to this representation to guide the generation of the output sequence. This simple strategy has been used in MT to control second person pronoun forms when translat-

at https://Newsela.com/data/. Scripts to replicate our model configurations and our cross-lingual segment aligner are available at https://github.com/sweta20/ComplexityControlledMT.

| $c_i$ | Spanish ($s_i$) | $c_o$ | English ($s_o$) | Operation |
|---|---|---|---|---|
| 9 | Doug Ratliff, un empresario de 67 años de edad de Richlands, Virginia, dijo que la elección de Trump sería uno de los días más felices de su vida. | 3 | Doug Ratliff is a businessman from Virginia. Ratliff said Trump's election would be one of the happiest days of his life. | Splitting; Deletion |
| 12 | Incluso antes de haber nacido, Daliyah Marie Arana, según dicen sus padres, estaba aprendiendo a leer. | 4 | Daliyah Marie Arana has been learning to read since before she was born. | Paraphrasing |
| 9 | Kes Gray es el escritor de la serie de cuentos de animales "Oi Frog and Friends". A él no le interesaron mucho los descubrimientos del estudio. Los autores y los ilustradores solo necesitan que los personajes principales animales de sus historias sean adorables, concluyó. | 5 | Kes Gray is the writer of the rhyming animal series "Oi Frog and Friends." He was not bothered by the study's findings. Writers and artists just need to keep the main animal characters in their stories cuddly, he said. | Lexical substitution |

Table 1: Cross-lingual Newsela examples: the Spanish text $s_i$ of complexity, or reading grade level, $c_i$ is automatically aligned to English text $s_o$ of $c_o$. Simplification transformations range from sentence splitting and deletions to paraphrasing and lexical substitution.

ing into German (Sennrich et al., 2016), to indicate the target language in multilingual MT (Johnson et al., 2016), and to obtain formal or informal translations of the same input (Niu et al., 2018). In monolingual text simplification tasks (Scarton and Specia, 2018), the reading grade level has been encoded as such a special token.

**Training Data and Objectives** Fully supervised training would ideally require translation samples with outputs representing different levels of complexity for the same input segment. However, constructing such data at the scale required to train deep neural networks is expensive and unrealistic. Our multi-task training configuration lets us exploit different types of training examples to train shared encoder-decoder parameters $\theta$. We use the following samples/tasks:

- Complexity controlled MT samples $(s_i, c_o, s_o)$: These are the closest samples to the task at hand, but are hard to obtain. They are used to defined the complexity-controlled MT loss

$$\mathcal{L}_{CMT} = \sum_{(s_i, c_o, s_o)} \log P(s_o|s_i, c_o; \theta) \quad (1)$$

- MT samples $(s_i, s_o)$: These are sentence pairs drawn from parallel corpora. They are available in large quantities for many language pairs (Tiedemann, 2012) and are used to define the MT loss

$$\mathcal{L}_{MT} = \sum_{(s_i, s_o)} \log P(s_o|s_i; \theta) \quad (2)$$

- Text simplification samples in the MT target language $(s_o, c_{s'_o}, s'_o)$ where $s'_o$ is a simplified version of complexity $c_{s'_o}$ for input $s_o$, which are likely to be available in much smaller quantities than MT samples.

$$\mathcal{L}_{Simplify} = \sum_{(s_o, c_{s'_o}, s'_o)} \log P(s_o|s'_o c_{s'_o}; \theta) \quad (3)$$

The multi-task loss is simply obtained by summing the losses from individual tasks: $\mathcal{L}_{CMT} + \mathcal{L}_{MT} + \mathcal{L}_{Simplify}$.

## 4 The Newsela Cross-Lingual Simplification Dataset

We build on prior work that used the Newsela dataset for English or Spanish text simplification by automatically aligning English and Spanish segments of different complexity to enable complexity-controlled machine translation.

The Newsela website provides high quality data to study text simplification. Xu et al. (2015) argue that text simplification research should be grounded in texts that are simplified by professional editors for specific target audiences, rather than more general-purpose crowd-sourced simplifications such as those available on Wikipedia. They show that Wikipedia is prone to sentence alignment errors, contains a non-negligible amount of inadequate simplifications, and does not generalize well to other text genres. By contrast, Newsela is an instructional content platform meant to help teachers prepare curriculum that match the language skills required at each grade level. The Newsela corpus consists of English articles in their original form, 4 or 5 different versions rewritten by professionals to suit different grade levels as well as optional translations of original and/or simplified English articles into Spanish resulting in 23,130 English and 5,320 Spanish articles with grade annotations respectively.

This section introduces our method to align English and Spanish segments across complexity levels, and the resulting bilingual dataset.

### 4.1 Cross-Lingual Segment Alignment

Extracting training examples from this corpus requires aligning segments within documents. This is challenging because text is neither simplified nor translated sentence by sentence, and as a result, equivalent content might move from one sentence to the next. Past work has introduced techniques to align segments of different complexity within documents of the same language (Xu et al., 2015; Paetzold et al., 2017; Štajner et al., 2018).

Complexity controlled MT requires aligning segments of different complexity in English and Spanish. Existing methods for aligning sentences in English and Spanish parallel corpora are not well suited to this task. For instance, the Gale-Church algorithm (Gale and Church, 1993) assumes that aligned sentences should have similar length. This assumption does not hold if the English article is a simplification of the Spanish article. Consider the following Spanish text and its English translation in Newsela:

**Spanish**: LA HAYA, Holanda - Te has tomado alguna vez una selfie?, Hoy en día es muy fácil. Solo necesitas un teléfono inteligente.

**Google Translated English**: THE HAGUE, Netherlands - Have you ever taken a selfie? Today is very easy. You only need a smart phone.

**Original English Version**: THE HAGUE, Netherlands - All you need is a smartphone to take a selfie. It is that easy.

As a result, we adapt a monolingual text simplification aligner for cross-lingual alignment. MASSAlign (Paetzold et al., 2017) is a Python library designed to align segments of different length within comparable corpora of the same language. It employs a vicinity-driven search approach, based on the assumption that the order in which information appears is roughly constant in simple and complex texts. A similarity matrix is created between the paragraphs/sentences of aligned documents/paragraphs using a standard bag-of-words TF-IDF model. It finds a starting point to begin the search for an alignment path, allowing long-distance alignment skips, capturing 1-N and N-1 alignments. To leverage this alignment flexibility, we apply MASSAlign to English articles and Spanish articles machine translated into English by Google translate.[2] An important property of Google translated articles is that they are aligned 1-1 at the sentence level. This lets us deterministically find the Spanish replacement for the aligned Google translated English version returned by MASSAlign. Translation quality is high for this language pair, and even noisy translated articles contain enough signal to construct the similarity matrix required by MASSAlign.

### 4.2 Resulting Dataset

We thus create: both samples for complexity controlled MT $(s_i, c_o, s_o)$ and traditional monolingual text simplification samples $(s_o, c_{s'_o}, s'_o)$ that can be used by the multi-task model (Section 3). Since the properties of Newsela monolingual simplification samples have been studied thoroughly by Xu et al. (2015), we present key statistics for the cross-lingual simplification examples only. Table 2 contrasts Newsela parallel segments with bilingual parallel sentences drawn from the OPUS corpus (Tiedemann, 2009). We use Global Voices and News Commentary from OPUS corpus as it has the most similar domain to the Newsela data. Aligned segments in Newsela are about twice as long as segments in parallel corpora, and contain more than two sentences on each side on average. By contrast, parallel corpora samples align sentences one-to-one on average.

[2] https://translate.google.com/

| Dataset | # tokens/segment | #sents/segment | # of types | # of tokens |
|---|---|---|---|---|
| *Spanish* | | | | |
| Newsela | 50.13 | 2.17 | 57,361 | 7,792,285 |
| Global Voices | 22.96 | 1.03 | 254,111 | 15,921,948 |
| News | 26.73 | 1.03 | 80,840 | 5,587,307 |
| *English* | | | | |
| Newsela | 43.37 | 2.65 | 39,012 | 7,139,717 |
| Global Voices | 21.93 | 1.06 | 222,383 | 15,208,054 |
| News | 23.76 | 1.04 | 49,589 | 4,939,085 |

Table 2: Comparisons of the English-Spanish Newsela corpus with machine translation corpora from OPUS drawn from Global Voices and News Commentary.

Articles are distributed across reading levels spanning grades 2 to 12 for both English and English-Spanish pairs. Table 3 highlights the vocabulary differences among the different grade levels for the Newsela Spanish-English corpus. The vocabulary size of the corpus corresponding to lower grade level is smaller as compared to higher complexity levels. Also, complex sentences have more words per sentence on average but fewer sentences per segment compared to their simplified counterparts. Simple sentences differ from complex sentences in various ways, ranging from sentence splitting and content deletion to paraphrasing and lexical substitutions, as illustrated in Table 1.

## 5 Experiment Settings

We evaluate **complexity controlled MT** using a subset of the 150k Spanish-English segment pairs extracted from Newsela as described in Section 4. We select Spanish and English segments that have different reading grade levels, so that given a Spanish input, the task consists in producing an English translation which is simpler (lower reading grade level) than the Spanish input.The train/development/test split ensures that there is no overlap between articles held out for testing and articles used for training. We refer to the corresponding training examples as **MT+simplify** since it represents the joint task of translation and simplification.

### 5.1 Evaluation Metrics

We evaluate the truecased detokenized output of our models using three automatic evaluation metrics, drawing from both machine translation and text simplification evaluation.

BLEU (Papineni et al., 2002) estimates translation quality based on $n$-gram overlap between system output and references. However it does not separate mismatches due to meaning errors and mismatches due to simplification errors.

SARI (Xu et al., 2016)[3] is designed to evaluate text simplification systems by comparing system output against references and against the input sentence. It explicitly measures the goodness of words that are added, deleted and kept by the systems. Xu et al. (2016) showed that BLEU shows high correlation with human scores for grammaticality and meaning preservation and SARI shows high correlation with human scores for simplicity. In the cross-lingual setting, we cannot directly compare the Spanish input with English hypotheses and references, therefore we use the baseline machine translation of Spanish into English as a pseudo-source text. The resulting SARI score directly measures the improvement over baseline machine translation.

In addition to BLEU and SARI, we report Pearson's correlation coefficient (PCC) to measure the strength of the linear relationship between the complexity of our system outputs and the complexity of reference translations. Heilman et al. (2008) use it to evaluate the performance of reading difficulty prediction. Here we estimate the reading grade level complexity of MT outputs and reference translations using the Automatic Readability Index (ARI)[4] score, which combines evidence from the number of characters per word and number of words per sentence using hand-tuned

[3] https://github.com/cocoxu/simplification

[4] https://github.com/mmautner/readability

| | Source (Spanish) | | | Target (English) | | |
|---|---|---|---|---|---|---|
| Grade | word types | tokens/segment | sents/segment | word types | tokens/segment | sents/segment |
| 2 | - | - | - | 3749 | 34.31 | 3.76 |
| 3 | 2,628 | 38.59 | 3.52 | 10,615 | 35.57 | 3.28 |
| 4 | 8,431 | 39.33 | 2.95 | 10,414 | 39.38 | 3.09 |
| 5 | 17,082 | 40.96 | 2.59 | 18,508 | 42.18 | 2.87 |
| 6 | 16,945 | 42.78 | 2.25 | 16,613 | 44.21 | 2.62 |
| 7 | 22,352 | 46.65 | 2.19 | 23,617 | 47.54 | 2.57 |
| 8 | 19,317 | 47.07 | 1.96 | 17,746 | 46.66 | 2.24 |
| 9 | 24,846 | 50.08 | 1.87 | 22,230 | 50.08 | 2.25 |
| 10 | 482 | 49.19 | 1.90 | 341 | 38.23 | 1.70 |
| 12 | 42,355 | 53.98 | 1.96 | - | - | - |

Table 3: Grade level Statistics of the Newsela Spanish-English corpus. Vocabulary size decreases with the reading grade level. Simpler segments contain fewer sentences and are often shorter that complex segments.

weights (Senter and Smith, 1967):

$$ARI = 4.71(\frac{chars}{words}) + 0.5(\frac{words}{sents}) - 21.43 \quad (4)$$

### 5.2 Training Data

In addition to the Newsela **MT+Simplify** training examples described above, which are of the form $(s_i, c_o, s_o)$, we use monolingual English simplification data, bilingual parallel training data and Spanish simplification data.

**Newsela Simplification** provides training examples of the form $(s_o, c_{s'_o}, s'_o)$, where $s_o$ and $s'_o$ are in the same language. We refer to this data as **Simplify** data. It is used for training multi-task models and for auxiliary evaluation on English only. Our version of this corpus has 513k English segment pairs extracted using the method by Paetzold and Specia (2016). Similar to Scarton and Specia (2018), an original article 0 can be aligned to up to four simplified versions: 1,2,3 and 4. Here 4 denotes the least simplified level and 0 represents the most simplified level. The train split consists of 460k instance pairs whereas the development and test sets consist of roughly 20K instances, drawn from the same articles as the MT+preserve and MT+simplify test set. For Spanish, we have 110k segment pairs, which will be used to train the Spanish simplification baseline.

**Bilingual Parallel Data (Newsela)** We also extract parallel Spanish-English segments from Newsela based on aligned segments between Spanish and English articles that have the same reading grade level. We use this dataset to provide in-domain MT training examples which includes roughly 70k instances.

All datasets are pre-processed using Moses tools for normalization, tokenization and true-casing (Koehn et al., 2007). We further segment tokens into subwords using a joint source-target byte pair encoding model with 32,000 operations (Sennrich et al., 2015).

### 5.3 Sequence-to-Sequence Model Configuration

We use the standard encoder-decoder architecture implemented in the Sockeye toolkit (Hieber et al., 2017). Both encoder and decoder have two Long Short Term Memory (LSTM) layers (Bahdanau et al., 2015), hidden states of size 500 and dropout of 0.3 applied to the RNNs of the encoder and decoder which is same as what was used by Scarton and Specia (2018). We observe that dot product based attention works best in our scenario, perhaps indicating that the task of complexity controlled translation requires mostly local changes that do not lead to long distance reorderings across sentences. We train using the Adam (Kingma and Ba, 2014) optimizer with a batch size of 256 segments and checkpoint the model every 1000 updates. Training stops after 8 checkpoints without improvement of validation perplexity. The vocabulary size is limited to 50000. We decode with a beam size of 5. Grade side-constraints are defined using a distinct special token for each grade level (from 2 to 12). The constraint token corresponds to the grade level of the target instance.

### 5.4 Baseline

We contrast the multi-task system with pipeline based approaches, where translation and simplification are treated as independent consecutive steps. We train a neural MT model to perform translation from Spanish to English and other neural MT models to perform monolingual text simplification for Spanish and English respectively. In the first pipeline setup, the output from the translation model is fed as input to an English simplification model while in the other, the output from the Spanish simplification model is fed as input to an translation model. As Scarton and Specia (2018), we simply use grade level tokens as side constraints on English simplification examples to control output complexity.[5]

## 6 Evaluation of Complexity Controlled MT

We compare pipeline and multitask models on the Newsela complexity controlled MT task (Table 4). Overall, results show that compared to pipeline models, multitask models produce complexity controlled translations that better match human references according to BLEU. SARI suggests that multitask translations are simpler than baseline translations, and their resulting complexity correlates better with reference grade levels according to PCC.

The two pipeline models use the same MT system, therefore the difference between them comes from text simplification: using English simplification (first pipeline) outperforms Spanish simplification (second pipeline) according to BLEU and PCC, but not SARI. This can be explained by the smaller amount of Spanish simplification training data, which yields a model that generalizes poorly.

The "All tasks" model highlights the strengths of the multi-task approach: combining training samples from many tasks yields improvements over the "Translate and Simplify" multitask model which is trained on the exact same data as the pipelines. However, even without additional training data, the multitask "Translate and Simpify" model improves over baselines mainly by simplifying the output more, which suggests that the simplification component of the multitask model benefits from the additional MT training data.

[5] Additional constraints based on simplification operations were also used in that work but did not provide substantial benefits when operations are predicted based on the input.

Qualitative analysis suggests that the multi-task model is capable of distinguishing among different grade levels and the simplification operations performed for different grade levels are gradual. Table 5 illustrates simplification operations observed for a fixed grade 12 Spanish input into English with target grade levels ranging from 9 to 3. When translating to a nearby grade level, for example 9, the model roughly translates the entire input. For lower grade levels such as 7 and 5, lexical simplification and sentence splitting is observed. For the simplest grade level, the model deletes additional content. More examples are provided in the Appendix ( Table 13).

| Complexity cont. MT | BLEU | SARI | PCC |
|---|---|---|---|
| *Pipeline Baselines* | | | |
| Translate then Simplify | 21.98 | 30.4 | 0.436 |
| Simplify then Translate | 17.09 | 37.4 | 0.275 |
| *Multitask Models* | | | |
| Translate and Simplify | 22.51 | 44.8 | 0.572 |
| All Tasks | **22.75** | **45.0** | **0.608** |

Table 4: Compared to pipeline models, multitask models produce complexity controlled translations that better match human references (BLEU), that are simpler (SARI), and whose resulting complexity correlates better with the target grade level (PCC). Pipeline models are trained on Newsela Simplification data and MT parallel data from Newsela and OPUS. "Translate and Simplify" uses the exact same data in a multi-task model. The "All tasks" model uses all data available, including Newsla MT+Simplify examples.

## 7 Analysis

### 7.1 Output Grade Analysis

We aim to better understand to what degree models simplify the input text: how often does the output complexity exactly matches that of the reference? Does this change depend on the distance between input and output complexity levels? Table 6 compiles Adjacency Accuracy scores (Heilman et al., 2008), which represent the percentage of sentences where the system output complexity is within 1 or 2 grades of the reference text. We derive the reading grade levels from ARI (Senter and Smith, 1967) and conduct this analysis for the best pipeline ("Translate then Simplify") and multitask models ("All Tasks"). These adjacency scores are broken down according to the distance between input and target grade levels.

| | |
|---|---|
| 12 | Ahora el museo Mauritshuis está por inaugurar una exposición dedicada a los autorretratos del siglo XVII, que destaca las similitudes y diferencias entre las fotos modernas y las obras de arte históricas. |
| 9 | Now the museum Mauritois is launching an exhibition dedicated to the 18th century authoritations, highlighting the similarities and differences between modern photos and historical artworks. |
| 7 | The museum is **now set to open** an exhibition dedicated to the 18th century authoritations, highlighting the similarities and differences between modern photos and historical artworks. |
| 5 | The museum is now set to open an exhibit dedicated to the 18th century. **It highlights** the similarities and differences between modern photos and historical artworks. |
| 3 | The museum is now set to open an exhibit dedicated to the 18th century. It **shows** the similarities and differences between modern photos and art works. |

Table 5: Example of multi-task model outputs when translating grade 12 Spanish into increasingly simpler English.

| Adj. | | Source Grade - Target Grade | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| level | Model | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | Pipeline | **0.593** | **0.629** | 0.594 | 0.556 | 0.524 | 0.493 | 0.472 | 0.457 | 0.448 | 0.444 |
| | Multitask | 0.59 | 0.626 | 0.594 | **0.561** | **0.529** | **0.504** | **0.482** | **0.467** | **0.458** | **0.453** |
| 2 | Pipeline | **0.717** | **0.759** | **0.786** | 0.747 | 0.713 | 0.678 | 0.654 | 0.637 | 0.626 | 0.621 |
| | Multitask | 0.711 | 0.755 | 0.784 | **0.753** | **0.725** | **0.696** | **0.67** | **0.653** | **0.642** | **0.636** |

Table 6: Adjacency ARI accuracy within grade level given by Adjacency level for the system output with respect to the target grade: Multitask model is able to better capture the target grade than the pipeline model when the difference between the source and the target grade is greater than 3.

When the source and target grades are close, roughly 60% of system outputs that are within a $\pm 1$ window of the correct grade level. The pipeline model matches the target grade slightly better than the multitask model. However, in the more difficult case where the difference between source and target grades is larger than three, the multitask model outperforms the pipeline. Increasing the adjacency window to $\pm 2$ pushes the percentage of matches in the 70s.

Overall these results show that multitask and pipeline models are able to translate and simplify, but that they do not yet fully succeed at precisely controlling the complexity of their output to match a specific target reading grade.

### 7.2 Ablation Experiments

Table 7 shows the impact of different training data types on the multitask model using ablation experiments. OPUS improves BLEU and SARI performance across the board. However, using OPUS without any Newsela MT data (Row 4) hurts the correlation score, indicating the importance of in-domain MT data to control complexity. The difference between the performance when using joint translation and simplification (MT+S) examples (Row 2) vs. simplification only (S in Row 3) is small in terms of BLEU (+0.11) and PCC (0.012), indicating that the monolingual simplification dataset can provide simplification supervision when MT+Simplify data is unavailable. The overall best performance for the task is obtained by using all types of training examples.[6]

### 7.3 Evaluation on Auxiliary Tasks

In addition to complexity controlled MT, the multi-task model can be used to simplify English text, and to translate from Spanish-to-English without changing the complexity. For completeness, we evaluate on these two auxiliary tasks.

Table 8 summarizes the results: the multitask model slightly outperforms a dedicated simplification model on English simplification, showing the benefits of the additional training data from other tasks. By contrast, on the resource-rich MT task, the standalone translation system performs better.

[6] A random sample of outputs from the best model configuration are provided in the Appendix (Table 14).

| Newsela | | | OPUS | Evaluation Metrics | | |
|---|---|---|---|---|---|---|
| S | MT+S | MT | MT | BLEU | SARI | PCC |
| ✓ | ✓ | ✓ | ✓ | 22.75 | 45.0 | 0.608 |
| | ✓ | ✓ | ✓ | 22.62 | 44.5 | 0.584 |
| ✓ | | ✓ | ✓ | 22.51 | 44.8 | 0.572 |
| ✓ | | | ✓ | 19.16 | 43.4 | 0.468 |
| ✓ | ✓ | ✓ | | 14.65 | 41.4 | 0.521 |

Table 7: Data ablation experiments showing the impact of different types of training examples on multi-task model. The OPUS parallel corpus is essential to good performance. Simplification data (S) can be used for simplification supervision when joint translation and simplification examples (MT+S) are unavailable.

This can be explained by the fact that the standalone system is only responsible for text translation, while the multi-task model is exposed to samples of more diverse complexity levels during training which damage its ability to preserve complexity.

| Task | BLEU | SARI | PCC |
|---|---|---|---|
| *English Simplification* | | | |
| Simplify | 55.76 | 41.7 | 0.736 |
| Translate and Simplify | **56.47** | 41.3 | 0.730 |
| All Tasks | 56.05 | **42.1** | 0.736 |
| *Machine Translation* | | | |
| Translate | **29.09** | - | **0.769** |
| Translate and Simplify | 27.33 | - | 0.647 |
| All Tasks | 27.63 | - | 0.658 |

Table 8: Evaluation on auxiliary tasks: Multitask models trained on both the translation and simplification dataset improves the performance for the task of English Simplification.

### 7.4 Provenance of Reading Grade Level

Our models control complexity using the gold reading grade level assigned by professional Newsela editors. We investigate the impact of replacing these gold labels by automatic predictions from the ARI metric. ARI can be computed for any English segment, including for MT parallel corpora that are not annotated for complexity.

Table 9 shows that ARI provides an adequate substitute for manually annotated reading grade levels, as BLEU and SARI score remain close when Newsela reading grade levels are replaced by ARI-based tags. However, annotating all data with ARI grades, including the OPUS parallel corpus, hurts BLEU. We attribute this result to the differences in length and number of sentences per segment in OPUS vs. Newsela (Table 2): segments of vastly different lengths can have the same ARI score (Equation 4), thus confusing the multi-task model.

| Complexity cont. MT | BLEU | SARI | PCC |
|---|---|---|---|
| Newsela reading grade | 22.51 | 44.80 | 0.572 |
| ARI on Newsela data | 22.26 | **45.12** | **0.581** |
| ARI on all data | 20.91 | 44.75 | 0.577 |

Table 9: Complexity controlled MT with automatic vs. manual reading grade level tags: ARI provides an adequate substitute for manually assigned grade levels.

## 8 Conclusion

We introduce a new task that aims to control complexity in machine translation output, as a proxy for producing translations targeted at audiences with different reading proficiency levels. We construct a Spanish-English dataset drawn from the Newsela corpus for training and evaluation, and adopt a sequence-to-sequence model trained in a multitask fashion.

We show that the multitask model improves performance over translation and simplification pipelines, according to both machine translation and simplification metrics. The reading grade level of the multi-task outputs correlate better with target grade levels than with pipeline outputs. Analysis shows that these benefits come from their ability to combine larger training data from different tasks. Manual inspection also shows that the multi-task model successfully produces different translations for increasingly lower grades given the same Spanish input.

However, even when simplifying translations, multitask models are not yet able to exactly match the desired complexity level, and the gap between the complexity achieved and the target complexity increases with the amount of simplification required. Our datasets and models thus provide a foundation to investigate strategies for a tighter control on output complexity in future work.

## References

David Allen. 2009. A study of the role of relative clauses in the simplification of news texts for learn-

ers of english. *System*, 37(4):585–599.

Alessio Palmero Aprosio, Sara Tonelli, Marco Turchi, Matteo Negri, and Mattia A Di Gangi. 2019. Neural text simplification in low-resource conditions using weak supervision. In *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation*, pages 37–44.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *International Conference on Learning Representations (ICLR)*.

Dominique Brunato, Andrea Cimino, Felice Dell'Orletta, and Giulia Venturi. 2016. Paccss-it: A parallel corpus of complex-simple sentences for automatic text simplification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 351–361.

Chris Callison-Burch. 2007. *Paraphrasing and Translation*. Ph.D. thesis, University of Edinburgh.

Yvonne Canning, John Tait, Jackie Archibald, and Ros Crawley. 2000. Cohesive generation of syntactically simplified newspaper text. In *International Workshop on Text, Speech and Dialogue*, pages 145–150. Springer.

John Carroll, Guido Minnen, Darren Pearce, Yvonne Canning, Siobhan Devlin, and John Tait. 1999. Simplifying text for language-impaired readers. In *Ninth Conference of the European Chapter of the Association for Computational Linguistics*.

R. Chandrasekar, Christine Doran, and B. Srinivas. 1996. Motivations and Methods for Text Simplification. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 2*, COLING '96, pages 1041–1044, Stroudsburg, PA, USA.

William Coster and David Kauchak. 2011. Simple English Wikipedia: A New Text Simplification Task. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11, pages 665–669, Stroudsburg, PA, USA.

Jan De Belder and Marie-Francine Moens. 2010. Text simplification for children. In *Proceedings of the SIGIR workshop on accessible search systems*, pages 19–26. ACM Press New York.

Noemie Elhadad and Komal Sutaria. 2007. Mining a lexicon of technical terms and lay equivalents. In *Proceedings of the Workshop on BioNLP 2007: Biological, Translational, and Clinical Language Processing*, pages 49–56. Association for Computational Linguistics.

William A Gale and Kenneth W Church. 1993. A program for aligning sentences in bilingual corpora. *Computational linguistics*, 19(1):75–102.

Laurie Gerber and Eduard Hovy. 1998. Improving Translation Quality by Manipulating Sentence Length. In *Conference of the Association for Machine Translation in the Americas (AMTA)*.

Eva Hasler, Adrià de Gispert, Felix Stahlberg, Aurelien Waite, and Bill Byrne. 2017. Source sentence simplification for statistical machine translation. *Computer Speech & Language*, 45(Supplement C):221–235.

Michael Heilman, Kevyn Collins-Thompson, and Maxine Eskenazi. 2008. An analysis of statistical models and features for reading difficulty prediction. In *Proceedings of the third workshop on innovative use of NLP for building educational applications*, pages 71–79. Association for Computational Linguistics.

Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, and Matt Post. 2017. Sockeye: A toolkit for neural machine translation. *arXiv preprint arXiv:1712.05690*.

Kentaro Inui, Atsushi Fujita, Tetsuro Takahashi, Ryu Iida, and Tomoya Iwakura. 2003. Text simplification for reading assistance: A project note. In *Proceedings of the Second International Workshop on Paraphrasing*, pages 9–16, Sapporo, Japan.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation. *arXiv:1611.04558 [cs]*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

David Klaper, Sarah Ebling, and Martin Volk. 2013. Building a german/simple german parallel corpus for automatic text simplification. In *Proceedings of the Second Workshop on Predicting and Improving Text Readability for Target Reader Populations*, pages 11–19.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Annual Meeting of the Association for Computational Linguistics (ACL), Demonstration Session*, Prague, Czech Republic.

Reno Kriz, João Sedoc, Marianna Apidianaki, Carolina Zheng, Gaurav Kumar, Eleni Miltsakaki, and Chris Callison-Burch. 2019. Complexity-weighted loss and diverse reranking for sentence simplification. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for*

*Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3137–3147.

Kelly Marchisio, Jialiang Guo, Cheng-I Lai, and Philipp Koehn. 2019. Controlling the reading level of machine translation output. In *Proceedings of Machine Translation Summit XVII Volume 1: Research Track*, pages 193–203.

Shachar Mirkin and Jean-Luc Meunier. 2015. Personalized Machine Translation: Predicting Translational Preferences. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2019–2025, Lisbon, Portugal.

Daiki Nishihara, Tomoyuki Kajiwara, and Yuki Arase. 2019. Controllable text simplification with lexical constraint loss. In *Proceedings of the 57th Conference of the Association for Computational Linguistics: Student Research Workshop*, pages 260–266.

Sergiu Nisioi, Sanja Štajner, Simone Paolo Ponzetto, and Liviu P. Dinu. 2017. Exploring neural text simplification models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 85–91, Vancouver, Canada.

Xing Niu, Sudha Rao, and Marine Carpuat. 2018. Multi-Task Neural Models for Translating Between Styles Within and Across Languages. In *27th International Conference on Computational Linguistics (COLING 2018)*.

Gustavo Paetzold, Fernando Alva-Manchego, and Lucia Specia. 2017. Massalign: Alignment and annotation of comparable documents. *Proceedings of the IJCNLP 2017, System Demonstrations*, pages 1–4.

Gustavo Henrique Paetzold and Lucia Specia. 2016. Semeval 2016 task 11: Complex word identification. *Proceedings of SemEval*, pages 560–569.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, PA.

Sarah E Petersen and Mari Ostendorf. 2007. Text simplification for language learners: a corpus analysis. In *Workshop on Speech and Language Technology in Education*.

Ella Rabinovich, Shachar Mirkin, Raj Nath Patel, Lucia Specia, and Shuly Wintner. 2016. Personalized Machine Translation: Preserving Original Author Traits. *arXiv:1610.05461 [cs]*.

Carolina Scarton and Lucia Specia. 2018. Learning Simplifications for Specific Target Audiences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 712–718.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Controlling Politeness in Neural Machine Translation via Side Constraints. pages 35–40.

RJ Senter and Edgar A Smith. 1967. Automated readability index. Technical report, CINCINNATI UNIV OH.

Advaith Siddharthan. 2014. A survey of research on text simplification. *ITL - International Journal of Applied Linguistics*, 165(2):259–298.

Advaith Siddharthan and Napoleon Katsos. 2010. Reformulating discourse connectives for non-expert readers. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1002–1010. Association for Computational Linguistics.

Sanja Štajner, Iacer Calixto, and Horacio Saggion. 2015. Automatic text simplification for spanish: Comparative evaluation of various simplification strategies. In *Proceedings of the international conference recent advances in natural language processing*, pages 618–626.

Sanja Štajner, Marc Franco-Salvador, Paolo Rosso, and Simone Paolo Ponzetto. 2018. Cats: A tool for customized alignment of text simplification corpora. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*.

Sanja Štajner and Maja Popovic. 2016. Can text simplification help machine translation? In *Proceedings of the 19th Annual Conference of the European Association for Machine Translation*, pages 230–242.

Jörg Tiedemann. 2009. News from OPUS-A collection of multilingual parallel corpora with tools and interfaces. In *Recent Advances in Natural Language Processing*, volume 5, pages 237–248.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in opus. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey.

Eva Vanmassenhove, Christian Hardmeier, and Andy Way. 2018. Getting Gender Right in Neural Machine Translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3003–3008.

Tong Wang, Ping Chen, John Rochford, and Jipeng Qiang. 2016. Text simplification using neural machine translation. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Willian Massami Watanabe, Arnaldo Candido Junior, Vinícius Rodriguez Uzêda, Renata Pontin de Mattos Fortes, Thiago Alexandre Salgueiro Pardo, and Sandra Maria Aluísio. 2009. Facilita: reading assistance for low-literacy readers. In *Proceedings of the 27th ACM international conference on Design of communication*, pages 29–36. ACM.

Wei Xu, Chris Callison-Burch, and Courtney Napoles. 2015. Problems in current text simplification research: New data can help. *Transactions of the Association for Computational Linguistics*, 3:283–297.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415.

Xingxing Zhang and Mirella Lapata. 2017. Sentence Simplification with Deep Reinforcement Learning. *arXiv:1703.10931 [cs]*.

## A Supplemental Material

Table 10 and 11 provides the statistics of grade pair distribution in the Newsela English and Newsela Spanish-English dataset.

| Src / Tgt | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| 3 | 2652 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 4984 | 8212 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 2287 | 19589 | 23775 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 1914 | 7625 | 21022 | 21380 | 0 | 0 | 0 | 0 | 0 |
| 7 | 608 | 8897 | 14249 | 33466 | 10944 | 0 | 0 | 0 | 0 |
| 8 | 623 | 3710 | 13267 | 17347 | 22745 | 12006 | 0 | 0 | 0 |
| 9 | 130 | 5058 | 5031 | 19834 | 4684 | 30929 | 2144 | 0 | 0 |
| 10 | 6 | 40 | 224 | 320 | 382 | 289 | 400 | 142 | 0 |
| 11 | 0 | 0 | 15 | 19 | 11 | 16 | 28 | 0 | 0 |
| 12 | 1069 | 6818 | 18430 | 34232 | 28532 | 41561 | 29836 | 31327 | 97 |

Table 10: Number of text segments per grade level pair in our Newsela English Corpus

| Src / Tgt | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| 3 | 293 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 670 | 1305 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 251 | 3383 | 1957 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 223 | 1124 | 2090 | 2833 | 0 | 0 | 0 | 0 | 0 |
| 7 | 60 | 1249 | 926 | 4986 | 1244 | 0 | 0 | 0 | 0 |
| 8 | 96 | 548 | 1016 | 1804 | 3705 | 1221 | 0 | 0 | 0 |
| 9 | 16 | 717 | 211 | 3074 | 189 | 6135 | 263 | 0 | 0 |
| 10 | 0 | 3 | 5 | 15 | 26 | 1 | 46 | 0 | 0 |
| 12 | 189 | 1288 | 1902 | 5312 | 4708 | 7796 | 4995 | 7077 | 30 |

Table 11: Number of text segments per grade level pair in our Newsela English-Spanish Corpus

| Model | Bleu | SARI | Flesch |
|---|---|---|---|
| *Results copied from Scarton and Specia (2018)* | | | |
| seq2seq w/ side-constraint | 62.91 | 41.01 | 82.91 |
| *Reimplementation evaluated on our Newsela download* | | | |
| seq2seq w/ side-constraint | 58.61 | 39.81 | 70.44 |
| seq2seq w/ side-constraint + BPE | 61.87 | 52.78 | 66.98 |

Table 12: Comparison with previously published results on Newsela English text simplification. Our implementation yields BLEU and SARI score that are close to those reported in Scarton and Specia (2018). The difference in Flesch score can be attributed to changes in the number and complexity of articles available in newsela at the time the datasets were extracted.

| | |
|---|---|
| 12 | Se estima que 75 personas han expresado interés en alojar al menos a un solicitante de asilo, dijo Cronk. Algunas de estas personas viven en las principales áreas metropolitanas de California y Nueva York. Otros son de zonas remotas y rurales de Montana y Dakota del Norte. |
| 8 | An estimated 75 people have expressed interest in hosting at least one asylum-seeker said Cronk. Some of these people live in major California and New York area. Others are from remote and rural areas of Montana and North Dakota. |
| 6 | An estimated 75 people have expressed interest in hosting at least one asylum-seeker said Cronk. Some of these people live in major California and New York area. Others are from remote and rural areas of Montana and North Dakota. |
| 4 | An estimated 75 people have expressed interest in hosting at least one asylum-seeker said Cronk. Some of these people live in the **main** areas of California and New York. Others are from remote and rural areas of Montana and North Dakota. |
| 2 | **Many people live in the** United States and New York City. Some are from remote areas of Montana and North Dakota. |
| 12 | El gobierno federal realizó un contrato con el centro de detención juvenil en Vicennes, Indiana, desde el año 2004 hasta el año 2010, para que alojara a aquellos niños inmigrantes considerados como los más peligrosos. |
| 9 | The federal government conducted a contract with the youth detention center at Vicennes, Indiana, since 2004 to 2010 to host those immigrant children considered as the most dangerous ones. |
| 7 | The federal government conducted a contract with the youth detention center at Vicennes, Indiana, since 2004 to 2010, **so that they hosted** those immigrant children considered as the most dangerous ones. |
| 5 | The federal government **made** a contract with the youth detention center in Vice-Year. **It is in the** United States since 2004 to 2010. |
| 3 | The government made a deal with the youth detention center in Vice-Year. It is in the United States since 2004 to 2010. **It was to host** those immigrant children as the most dangerous in 2010. |
| 12 | Poco después de que el dron despegó durante la prueba de Verizon en Cape May, una "aeronave de seguimiento" salió tras él, para garantizar que el dron pudiera evadir otros aviones en caso de volar dentro de un espacio aéreo designado. |
| 9 | Shortly after the drone took off from Verizon in Cape May, a "unmanned aircraft following" came out after him, to ensure that the drone could evade other planes in case of flying inside a designated airspace. |
| 6 | Shortly after the drone took off in Cape May, a "unmanned aircraft force" came out **behind** him. **They could** ensure that the drone could evade other planes in case of flying inside a designated airspace. |
| 2 | **Not long** after the drone took off from Verizon in Cape May, a "unmanned aircraft help" came out after him. **It is to make sure** the drone could evade other planes in case of flying inside a air space. |

Table 13: Examples of simplification operations observed when simplifying from a higher Grade level into different lower grade levels using the multitask model (All tasks).

| Src_G | Tgt_G | Source | Target | Predicted |
|---|---|---|---|---|
| 8 | 5 | Los partidarios de la reserva dicen que marca un precedente para muchos países que trabajan conjuntamente para proteger una gran porción del océano que no se halla dentro de la jurisdicción específica de ningún país. | Supporters of the reserve say it sets an example for multiple countries working together to protect a large area of ocean. The area is not controlled by any single nation. | Supporters of the reserve say that marks a precedent for many countries that work together to protect a large portion of the ocean. |
| 5 | 3 | Los padres no respondieron bien al escuchar que sus hijas estaban surfeando porque les preocupaba el peligro y la reputación de las niñas. | The parents did not react well to their daughters surfing. They worried about the danger and what other people would think. | Parents did not want to hear that their daughters were surfing their daughters because they were worried about the danger and their reputation. |
| 12 | 5 | Según dijo el actor Eds Eramiha, esta escena habría sido difícil de imaginar hace tan solo dos décadas, cuando el idioma te reo maorí era ampliamente considerado una lengua moribunda que no valía la pena enseñar. | Eds Eramiha is an actor in the play. He says it would have been difficult to imagine this scene just 20 years ago. Back then, te reo Maori was widely regarded as a dying language not worth teaching. | Actress Eds Eramiha said this scene would have been hard to imagine only two decades ago. The language was widely viewed as a dying tongue that would not be worth teaching. |
| 5 | 4 | Dos astrónomos han planteado una fascinante pregunta: ¿Pueden las lunas tener lunas? La interesante respuesta es que sí, una luna puede tener su propia luna. | Two astronomers asked a question: Can moons have moons? The interesting answer is yes, a moon can have its own moon. | Two astronomers have asked a fascinating question: Can Mondays have Monday? The interesting answer is that yes, a moon can have its own. |
| 9 | 3 | Como hijo de un sastre y modisto de pocos ingresos, consiguió trabajo en el taller de un sastre en Brooklyn a la edad de 15 años, terminó su secundaria estudiando por las noches y luego se dispuso a triunfar en la vida. | Campos is the son of a tailor and a poor dressmaker. He found work in a Brooklyn clothing shop at age 15 and He finished high school at night. Then he set out to succeed. | The son of a saga and a modest age of a few incomes, he got work at the age of 15 years old. He graduated from the evenings and then turned into life in life. |

Table 14: Example translations produced by our best multitask model. Refer Table 7 (Row 1).

# Investigating Multilingual NMT Representations at Scale

**Sneha Kudugunta** **Ankur Bapna** **Isaac Caswell** **Orhan Firat**
Google AI
{snehark,ankurbpn,icaswell,orhanf}@google.com

## Abstract

Multilingual Neural Machine Translation (NMT) models have yielded large empirical success in transfer learning settings. However, these black-box representations are poorly understood, and their mode of transfer remains elusive. In this work, we attempt to understand massively multilingual NMT representations (with 103 languages) using Singular Value Canonical Correlation Analysis (SVCCA), a representation similarity framework that allows us to compare representations across different languages, layers and models. Our analysis validates several empirical results and long-standing intuitions, and unveils new observations regarding how representations evolve in a multilingual translation model. We draw three major conclusions from our analysis, with implications on cross-lingual transfer learning: (i) Encoder representations of different languages cluster based on linguistic similarity, (ii) Representations of a source language learned by the encoder are dependent on the target language, and vice-versa, and (iii) Representations of high resource and/or linguistically similar languages are more robust when fine-tuning on an arbitrary language pair, which is critical to determining how much cross-lingual transfer can be expected in a zero or few-shot setting. We further connect our findings with existing empirical observations in multilingual NMT and transfer learning.

## 1 Introduction

Multilingual Neural Machine Translation (NMT) models have demonstrated great improvements for cross-lingual transfer, on tasks including low-resource language translation (Zoph et al., 2016; Nguyen and Chiang, 2017; Neubig and Hu, 2018) and zero or few-shot transfer learning for downstream tasks (Eriguchi et al., 2018; Lample and Conneau, 2019; Wu and Dredze, 2019). A possible explanation is the ability of multilingual models to encode text from different languages in a shared representation space, resulting in similar sentences being aligned together (Firat et al., 2016; Johnson et al., 2017; Aharoni et al., 2019; Arivazhagan et al., 2019b). This is justified by the success of multilingual representations on tasks like sentence alignment across languages (Artetxe and Schwenk, 2018), zero-shot cross-lingual classification (Eriguchi et al., 2018) and XNLI (Lample and Conneau, 2019). Although there exist empirical results that suggest that transfer is stronger across similar languages (Zoph et al., 2016; Neubig and Hu, 2018; Wu and Dredze, 2019), the mechanism of generalization in multilingual representations is poorly understood.

While interpretability is still a nascent field, there has been some work on investigating the learning dynamics and nature of representations learnt by neural models (Olah et al., 2018). Singular Value Canonical Correlation Analysis (SVCCA) is one such method that allows us to analyze the similarity between noisy, high-dimensional representations of the same datapoints learnt across different models, layers and tasks (Raghu et al., 2017). SVCCA has been used to understand the learning dynamics and representational similarity in a variety of computer vision (Morcos et al., 2018), language models (Saphra and Lopez, 2018) and NMT (Bau et al., 2018).

In this work, we attempt to peek into the black-box of massively multilingual NMT models, trained on 103 languages, with the lens of SVCCA. We attempt to answer:

- Which factors determine the extent of overlap in the learned representations?
- Is the extent of representational overlap similar throughout the model?



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1565–1575, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

- How robust are multilingual NMT representations to fine-tuning on an arbitrary other language?

Answers to the above questions might have large implications on how we approach multilingual models and cross lingual transfer learning. Our work is the first that attempts to understand the nature of multilingual representations and cross-lingual transfer in deep neural networks, based on analyzing a model trained on 103 languages simultaneously.

We structure the study into these sections: In Section 2, we describe the experimental setup and tools used to train and analyze our multilingual NMT model. Following that, we attempt to answer each of the above questions in Sections 3 and 4. Finally in Section 5 we summarize our findings with a discussion of the potential implications.[1]

## 2 Experimental Setup

### 2.1 Data and Model

We study multilingual NMT on a massive scale, using an in-house training corpus (Arivazhagan et al., 2019b) generated by crawling and extracting parallel sentences from the web (Uszkoreit et al., 2010). Our dataset contains more than 25 billion sentence pairs for 102 languages to and from English, adding up to 204 direct language pairs.

Having being crawled from the web, our dataset has some important characteristics worth mentioning.

1. **Heavy imbalance between language pairs:** The number of parallel sentences per language pair ranges between $10^4$ to $10^9$. Figure 1 illustrates the data distribution for all the language pairs we study. Although this skew introduces optimization challenges (see Appendix A.1), it also creates a plausible setup for maximizing the positive language transfer from high-resource to low-resource language pairs, making it possible to study low-resource languages, that would otherwise have been very low quality.[2]

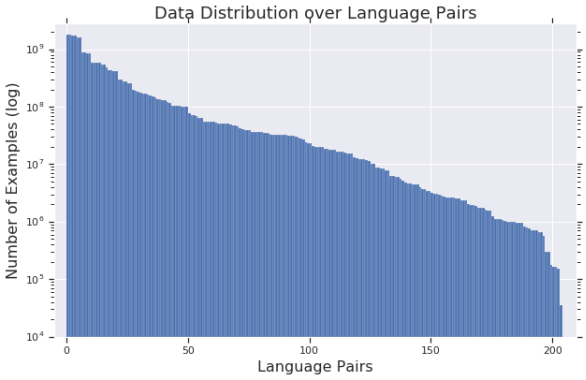


Figure 1: Per language pair data distribution of the dataset used to train our multilingual model. The y-axis depicts the number of training examples available per language pair on a logarithmic scale. Dataset sizes range from $10^4$ for the lowest resource language pairs to $10^9$ for the largest.

2. **Diversity:** Our corpus has languages belonging to a wide variety of scripts and linguistic families. These characteristics of our dataset make the problem that we study as realistic as possible. Models trained on this massive open-domain dataset are expected to yield rich, complex representations which we attempt to study in this paper.

To minimize confounding factors and control the evaluation set size and domain, we created a multi-way aligned evaluation set containing nearly 3k sentence pairs for all languages.[3] This also allows us to analyze the representational difference and similarity while controlling for semantics.

We use the Transformer-Big (Vaswani et al., 2017) architecture described in Artetxe and Schwenk (2018) for our experiments and share all parameters across language pairs including softmax layer and input/output word embeddings. For vocabulary, we use a Sentence Piece Model (Kudo and Richardson, 2018) with 64k tokens shared on both the encoder and decoder side.

Each set of parallel sentences has a `<2xx>` token prepended to the source sentence to indicate the target language, as in (Johnson et al., 2017). [4]

### 2.2 SVCCA

In this study we use Singular Value Canonical Correlation Analysis (SVCCA) (Raghu et al., 2017)

[1] Tools for online visualization and representation similarity used in our work will be open-sourced to facilitate further analysis.

[2] We provide baseline BLEU scores for all languages in Appendix A.2, notice the improvements for low-resource languages in our multilingual setup.

[3] Each sentence in our evaluation set is semantically identical across all other languages.

[4] Further details of the model and training routines are given in Appendix A.1.

as our investigation tool. SVCCA is a technique to compare vector representations in a way that is both invariant to affine transformations and fast to compute. We express a layer's representations as its activations on a set of $n$ examples, $X = \{x_1, \ldots, x_n\}$. Let $l_1 \in \mathbb{R}^{n \times d_1}$ and $l_2 \in \mathbb{R}^{n \times d_2}$ be the representations of two neural network layers, with $d_1$ and $d_2$ being the dimensions of the layers corresponding to $l_1$ and $l_2$ respectively. Given layer activations over the set $X$, SVCCA does the following:

1. Computes SVD decompositions of $l_1$ and $l_2$ to get subspaces $l_1' \in \mathbb{R}^{n \times d_1'}$ and $l_2' \in \mathbb{R}^{n \times d_2'}$ that capture most of the variance in the subspace.[5]

2. Uses Canonical Correlation Analysis (CCA) (Hardoon et al., 2004) to linearly transform $l_1'$ and $l_2'$ to be as aligned as possible, i.e., CCA computes $\tilde{l_1} = W_1 l_1'$ and $\tilde{l_2} = W_2 l_2'$ to maximize the correlations $\bar{\rho} = \{\rho_1, ..., \rho_{min(d_1', d_2')}\}$ between the new subspaces.

As done in (Raghu et al., 2017), we use the mean of the correlations, $\bar{\rho}$, as an approximate measure of the relatedness of representations.

### SVCCA for Sequences

Recent work on interpretability for NLU tasks uses methods such as diagnostic tasks (Belinkov et al., 2017; Tenney et al., 2019; Belinkov et al., 2018), attention based methods (Raganato and Tiedemann, 2018) or task analysis (Zhang and Bowman, 2018) and is primarily focused on understanding the linguistic features encoded by a trained model. Some recent work has applied SVCCA (or CCA) to language modeling (Morcos et al., 2018; Saphra and Lopez, 2018; Dalvi et al., 2019) and NMT (Bau et al., 2018; Dalvi et al., 2019). However, while Raghu et al. (2017) compare the learning dynamics of different classes in an image classifier, to the best of our knowledge, we are the first to apply SVCCA to a multilingual NMT or multitask setting. SVCCA was originally proposed for feed-forward neural networks, but our domain requires comparing unaligned sequences in different languages.

Sahbi (2018) proposes an alignment agnostic CCA approach to comparing unaligned data. However, the likelihood matrix $D$ specifying alignment of datapoints across different datasets (say, language $A$ and $B$) is application specific and infeasible to obtain in a multilingual setting. A simple heuristic is to summarize a set of activations by applying a pooling operation over the set. This is equivalent to assuming that a given token in a sentence from language $A$ is equally likely to be aligned to each token in an equivalent sentence in language $B$. We perform SVCCA on the hidden representations of the model, averaged over sequence time-steps, for each sentence in our evaluation set. We compare this approach with a naive token level SVCCA strategy in A.3.

### SVCCA Across Languages

In all known work using SVCCA, representations of the same data are used for analysis. However, in order to compare representations across languages, we leverage our multi-way parallel evaluation set to compare representations across different languages, since each data point is semantically equivalent across languages.

## 3 Multilingual NMT Learns Language Similarity

In this section, we use SVCCA to examine the relationship between representations of different languages learned by our massively multilingual NMT model. We compute SVCCA scores of layer-wise activations of a fully trained model between 103 languages pairs in both the Any-to-English and English-to-Any directions.[6]

**Visualizing the Representations** We first visualize the relationship between languages in their representation space for each layer using Spectral Embeddings (Belkin and Niyogi, 2003) [7]. In our case, we use mean SVCCA scores described in Section 2.2 as a similarity measure. Due to the differing nature of translating multiple languages to English and vice versa, the representation space of these two sets of languages, All-to-English and English-to-Any, behave differently and separate quite clearly (Figure 11 in the Appendix). We

[5] We retain 99% of the variance in our studies.

[6] Our multilingual NMT model is trained on the available training data which is English centric, hence an All-to-All multilingual model internally decomposes into All-to-English (X-En) and English-to-All (En-X) translation bundles, excluding zero-shot directions.

[7] We use the Laplacian Eigenmaps implementation of Spectral Embeddings in scikit-learn as of August 2019.

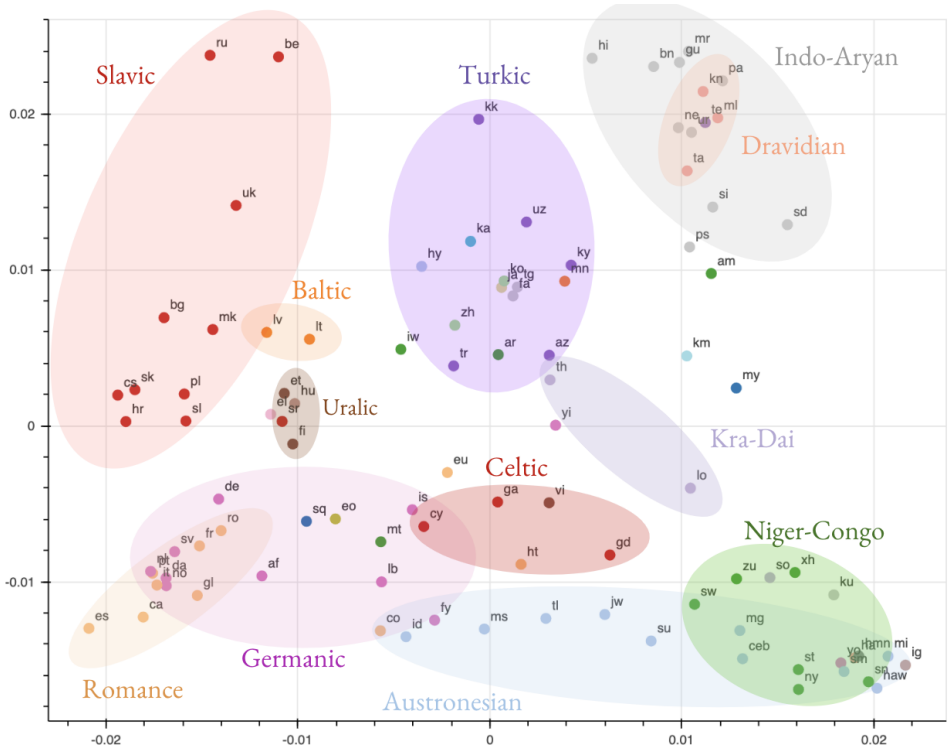


Figure 2: Visualizing clustering of the encoder representations of all languages, based on their SVCCA similarity. Languages are color-coded by their linguistic family. Best viewed in color.

first visualize the encoder representation of all languages in the All-to-English language pair set in Figure 2. For the sake of analysis, we then visualize subsets of the aforementioned 103 languages in Figures 3 and 4. We include visualizations of representations extracted from the embeddings and top layers of the encoder and decoder in the Appendix.

### 3.1 What is Language Similarity?

In the following sections we draw comparisons between the representational similarity of languages learned by our models, and the linguistic similarity between those languages. While there are entire sub-fields in linguistics devoted to studying similarity (e.g. Comparative Linguistics and Linguistic Typology (Ivić, 2011)), in this paper we define language similarity in terms of membership in the same language family (e.g. Turkic languages), or branch within that family (e.g. Oghuz Turkic languages). Families are groups of languages believed to share a common ancestor, and therefore tend to have similar vocabulary and grammatical constructs. An example phylogenetic language tree is given in Figure 3, on the right.

We also discuss writing systems, including scripts like Cyrillic, Roman, and Ge'ez. While similar languages frequently share the same script, that is not always true. Note that all of these categories[8] are muddled by various factors that are difficult to tease apart, and might be affected by the web-crawled data that we train on. For instance, languages sharing a script may also be part of the same political bloc, influencing what text is on the web. This and other confounding factors make a rigorous comparison exceedingly difficult. For brevity, we label languages in images with their BCP-47 language codes (Phillips and Davis, 2009), which are enumerated in the Appendix, Table 3.

### 3.2 Representations cluster by language similarity

We first visualize a clustering for all languages together in Figure 2. While there are a few outliers, we can observe some overlapping clusters, including the Slavic cluster on the top-left, the Germanic and Romance clusters on the bottom-left, the Indo-Aryan and Dravidian clusters on the top-right, etc. To analyze language clustering in more detail we

[8] When we refer to languages from a certain category, we only list those that are in our dataset. For example, when listing Turkic languages we exclude Bashkir, because we do not have English-Bashkir parallel data.

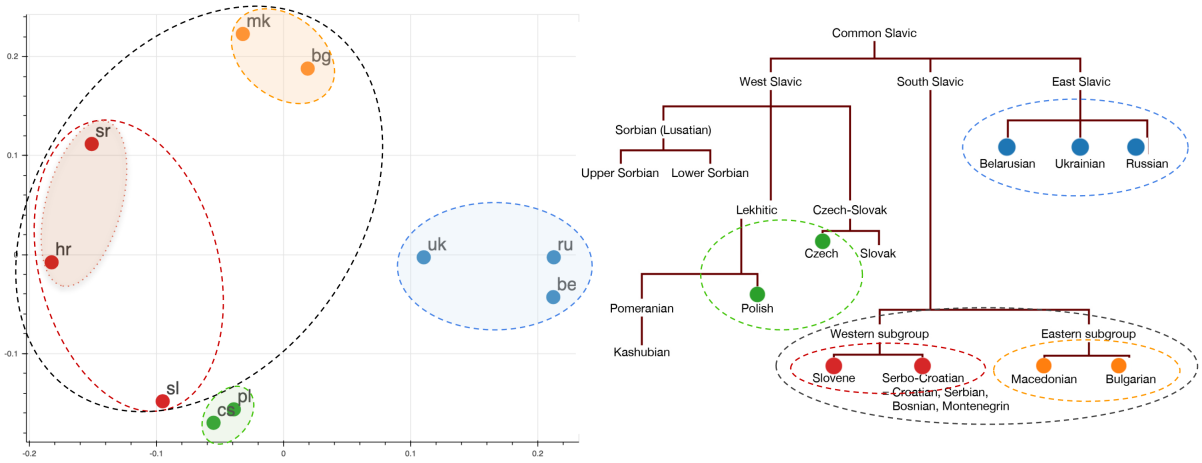


Figure 3: Comparing clusterings in the subword embeddings of the Slavic languages in our dataset with their family tree, which is the result of centuries of scholarship by linguists ((Browne and Ivanov)). Best seen in color.

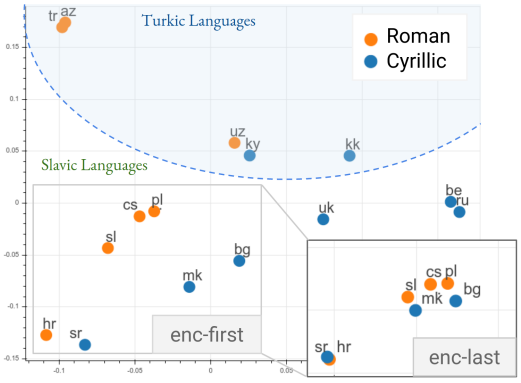


Figure 4: Representations of the Turkic and Slavic languages at the subword embeddings, compared with the top of the encoder, and colored by script. The inset shows a portion of the same image at the top of the encoder, focusing on the South-Western Slavic languages. Both images are at the same scale, see Appendix-Fig. 14 for more details. Best seen in color.

visualize sub-sets of the above languages.

In Figures 3 and 4, we visualize the Slavic and Turkic languages in our dataset. These languages come from two distinct families with very different linguistic properties, and within each family there are languages that are written in Cyrillic and Roman alphabets. This makes them ideal for understanding the interaction between superficial similarity (having the same alphabet and thus sharing many subwords) and linguistic similarity (sharing similar grammatical properties).

The first remarkable phenomenon we observe is that the clusters resulting from our model are grouped not only by family (Slavic), but branches within it (e.g. South Slavic), branches within those branches (e.g. Western Subgroup), and dialects within those (e.g. Serbo-Croatian). Figure 3 provides a more detailed look into the Slavic languages, and how this compositionality maps to the established family tree for Slavic languages. As can be seen in Figure 4, this phenomenon can also be observed for Turkic languages, with the Oghuz languages (Turkish and Azeri) forming one cluster, and the two Eastern branches Kipchak and Karluk (Uzbek, Kyrgyz, Kazakh) forming another.

A point worth special notice is the closeness between Serbian (sr) and Croatian (hr). Although these two are widely considered registers of the same language (Sussex and Cubberley), Serbian is written in Cyrillic, whereas Croatian is written in the Roman script. However, we see in both Figure 3 (middle left of plot) and Figure 4 (bottom left of plot) that they are each others' closest neighbors. Since they have no overlap in subword vocabulary, we conclude that they cluster purely based on distributional similarity – even at the level of sub-word embeddings.

Although we see strong clustering by linguistic family, we also notice the importance of script and lexical overlap, especially (and unsurprisingly) in the embeddings. In Figure 4 we visualize the Turkic and Slavic languages, and color by script. Although the linguistic cluster is probably stronger, there is also a distinct grouping by script, with the Roman-scripted languages on the left and the Cyrillic-scripted languages on the right. However, as we move up the encoder, the script associations become weaker and the language family as-

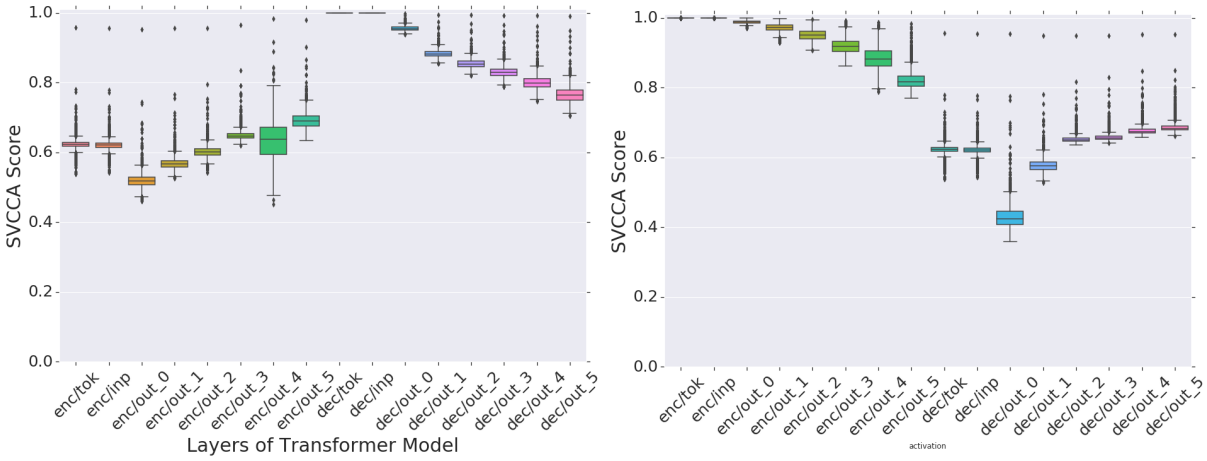


(a) X-En Language Pairs

(b) En-X Language Pairs

Figure 5: The change in distribution of pairwise SVCCA scores between language pairs across layers of a multilingual NMT model, with SVCCA scores between English-to-Any and Any-to-English language pairs visualized separately. We see that while the encoder in (a) and decoder in (b) have dissimilar representations across languages, the English representations of the decoder in (a) and the encoder in (b) diverge depending on the language X.

sociations become stronger. The inset in Figure 4 shows the seven South-Western Slavic languages at the top of the encoder, where they have clustered closer together. Again, Serbian and Croatian are an excellent example: *by the top of the encoder, they have become superimposed, diminishing the effect of the difference in script*.

We find that the trends discussed above are generally true for other language groupings too. The Appendix shows an example with the Dravidian, Indo-Aryan, and Iranian language families, demonstrating the same phenomena discussed above (Appendix Figure 12). Section A.5 further analyzes how the nearest neighbors of languages in SVCCA space become more linguistically coherent as one moves up the encoder.

### 3.3 Representational Similarity evolves across Layers

To visualize the how the representational overlap across languages evolves in the model, we plot how the distribution of pairwise SVCCA scores change across layers. For each layer, we first compute the pair-wise similarity between all pairs of languages. These similarity scores are then aggregated into a distribution and represented in Figures 5a and 5b, separately for the Any-to-English (X-En) and English-to-Any (En-X) language pairs.

For the Any-to-English (X-En) language pairs (Figure 5a), we notice that similarity between the source languages (X) increase as we move up the encoder, from embeddings towards higher level encoder layers, suggesting that the encoder attempts to learn a common representation for all source languages. This might also be motivated by training on a shared target language (En). However, representations at the top of the encoder are far from perfectly aligned, possibly indicating that different languages are represented in only partially overlapping sub-spaces. On the other hand, as the decoder incorporates more information from the source language (X), representations of the target (En) diverge. This is also in line with some findings of studies on translationese (Koppel and Ordan, 2011), where the authors show that that the translated text is predictive of the source language. For English-to-Any (En-X) language pairs (Figure 5b) we observe a similar trend. Representations of the source language (En) diverge as we move up the encoder, indicating that the representations of English sentences separate conditioned on the target language.

While it is a natural assumption that the encoder in a seq2seq model encodes the source, and the decoder decodes it into the target (Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2015), our results indicate that this change from source to target might be more gradual, and the boundary between encoder and decoder, in terms of the localization of the representation is blurry.

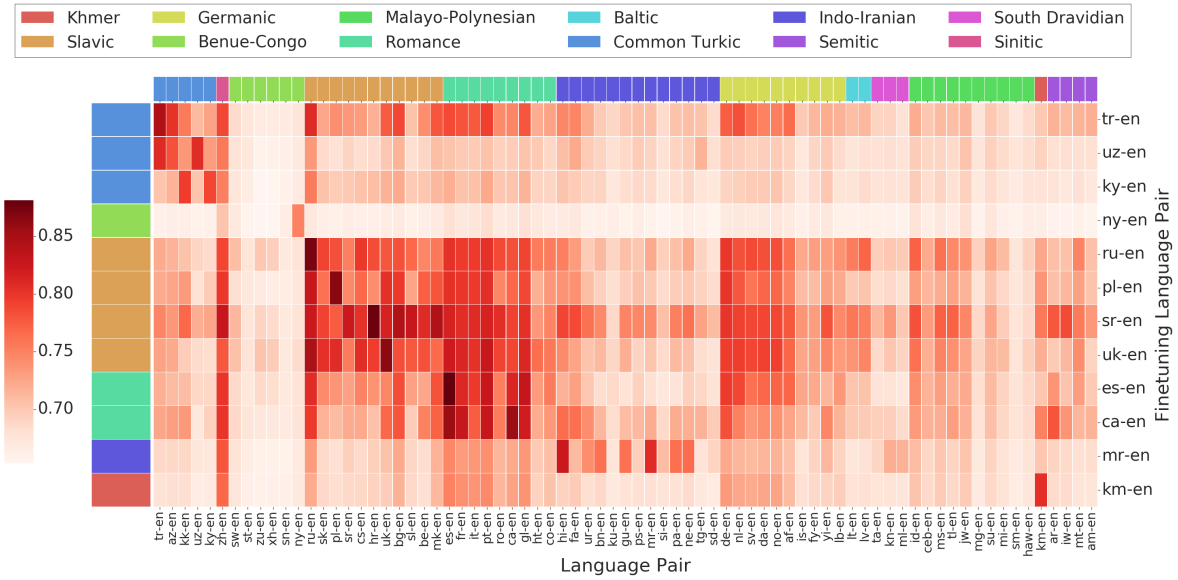


(a) SVCCA scores between the representations (top encoder layer) of *xx-en* language pairs before and after finetuning on various X-En language pairs. Darker cell means less change in representation (and higher SVCCA score) of *xx-en* on finetuning with X-En.

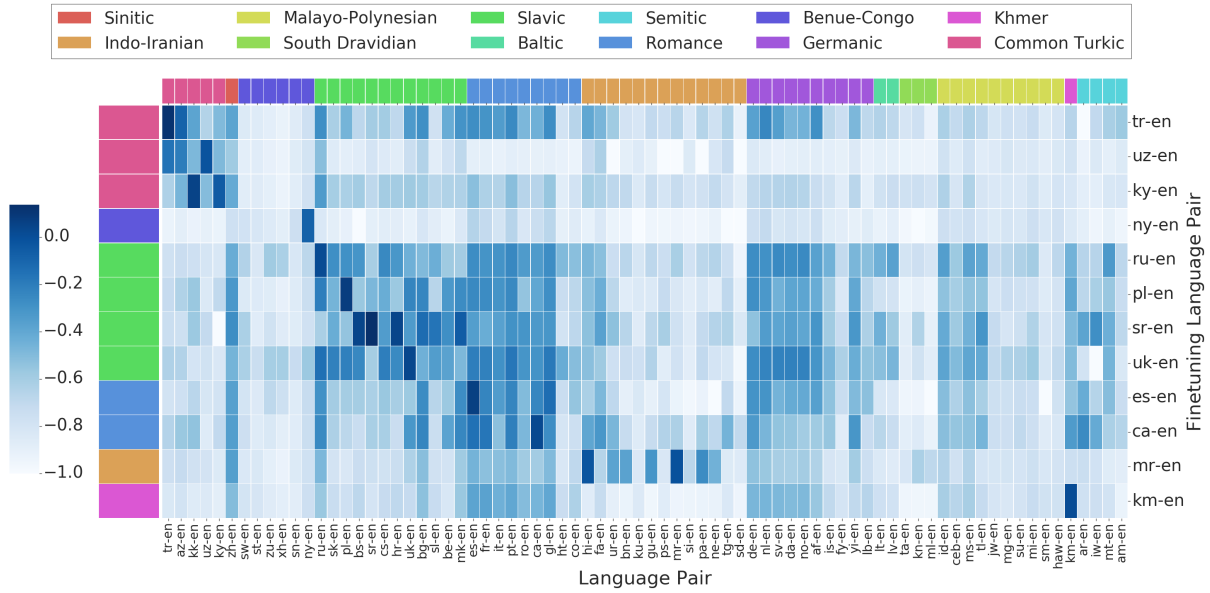


(b) Relative change in BLEU scores of *xx-en* language pairs after finetuning with various language pairs of the form X-En. Darker cell means less change in BLEU score of *xx-en* on finetuning with X-En.

Figure 6: Visualization depicting the (a) change in representations (using SVCCA) and (b) relative change in performance (in terms of test BLEU) of *xx-en* language pairs (x-axis), after fine-tuning a large multilingual model on various X-En language pairs (y-axis). Language sub-families on the axes are color-coded. We notice that representations of high resource languages are relatively robust to fine-tuning on any language. Additionally, languages linguistically similar to the one being fine-tuned on observe less distortion from their original representations.

## 4 Analyzing representation robustness to fine-tuning

In this section, we try to analyze the robustness of encoder representations, when fine-tuning a multilingual model on a single language pair. Note that here we define robustness to mean representational rigidity to fine-tuning, aka robustness to catastrophic forgetting (McCloskey and Cohen, 1989). Understanding the factors that affect robustness during fine-tuning is critical to determining how much cross-lingual transfer can be expected for individual languages in a zero or few-shot setting.

**Analysis Setup**: We fine-tune a fully trained multilingual model separately on 12 Any-to-English language pairs for 40k steps using the op-

timizer as described in A.1. These selected languages form a mix of high and low resource language pairs, from 6 language sub-families.[9]

We first attempt to quantify the extent of distortion in language representations caused by the fine-tuning process. To this end, we calculate the SVCCA similarity between the encoder representations of a language, before and after fine-tuning. We do this for all languages, in order to understand which factors determine the extent of distortion. We visualize these changes in Figure 6a for the final encoder layer, for all X-En language pairs. To complement our analysis of representational similarity, we visualize the relative change in BLEU score after fine-tuning in Figure 6b.

**Observations**: The first observation from Figures 6a and 6b is that the variations in SVCCA scores correspond very well with changes in BLEU; degradation in translation quality is strongly correlated with the magnitude of change in representations during fine-tuning.

We find that representations of high resource languages are quite robust to fine-tuning on any language. In Figure 6a, we see that high resource languages such as Chinese, German, Russian and Italian do not change much, irrespective of the language the model is fine-tuned on.

In general, we find that language representations are relatively robust to fine-tuning on a language pair from the same linguistic family. For example, on fine-tuning with tr-en (Turkish) or ky-en (Kyrgyz), the Turkic language group does not experience much shift in representation. We see a similar pattern with models fine-tuned on es-en (Spanish), ca-en (Catalan) and the Romance languages, uk-en (Ukranian), sr-en (Serbian), ru-en (Russian) and the Slavic languages.

An exception to these general trends seems to be fine-tuning on ny-en (Nyanja: Benue-Congo subfamily); all language pairs degrade by roughly the same extent, irrespective of language similarity or resource size. It's worth noting that all of the languages from the Benue-Congo sub-family are low-resource in our corpus.

These observations suggest that high resource languages might be responsible for partitioning the representation space, while low-resource languages become closely intertwined with linguistically similar high-resource languages. Low resource languages unrelated to any high resource languages have representations spread out across multiple partitions.

While these observations are based on representations from the top of the encoder, we further analyze sensitivity of representations to fine-tuning across different layers in the Appendix A.6. One key observation from that analysis is the robustness of embeddings to fine-tuning on any individual language; there is no significant change in the embedding representations (Correlation between representation of any language before and after finetuning $\bar{\rho} > 0.98$).

## 5 Discussion

Our work uncovers a few observations that might be of interest to practitioners working in multilingual NMT and cross-lingual transfer.

Our analysis reveals that language representations cluster based on language similarity. While language similarity has been exploited for adaptation previously (Neubig and Hu, 2018), our work is the first to concretely highlight which factors affect the overlap in representations across languages. This has potential implications for transfer learning: for example, it is possible to identify and exploit the nearest neighbors of a low resource language to maximize adaptation performance.

We also highlight how representation overlap evolves across layers, which is, again, of interest for cross-lingual transfer. For example, our analysis reveals that embeddings of different languages are less overlapping than the final encoder outputs. This hints that it might not be might not be effective to utilize input embeddings learned in multilingual NMT, since they don't overlap as much as the final encoder outputs. We also notice that encoder representation overlap across languages is not perfect, which explains why explicit language alignment or consistency losses might be needed to enable zero-shot NMT (Arivazhagan et al., 2019a; Al-Shedivat and Parikh, 2019).

We further analyze the robustness of language representations to fine-tuning, and notice that high-resource and linguistically similar languages are more robust to fine-tuning on an arbitrary language. This might help explain why linguistically distant languages typically result in poor zero-shot transfer (Wu and Dredze, 2019). Applying explicit losses, like elastic-weight consolidation (Kirkpatrick et al., 2017), to force language rep-

[9] More details on the relative resource sizes of different language pairs can be found in the Appendix A.6.

resentations of distant languages from getting distorted might help improve transfer performance.

## 6 Conclusion

To conclude, we analyzed factors that affect the overlap in representations learned by multilingual NMT models. We used SVCCA to show that multilingual neural networks share representations across languages strongly along the lines of linguistic similarity, and encoder representations diverge based on the target language. With this work we hope to inspire future work on understanding multitask and multilingual NLP models.

## Acknowledgments

First, we would like to thank Naveen Arivazhagan for foundational contributions to the project. We would like to thank Aditya Siddhant, Dmitry (Dima) Lepikhin, Klaus Macherey and Macduff Hughes for their helpful feedback on the draft. We would also like to thank Maithra Raghu for multiple insightful discussions about our methodology.

## References

Roee Aharoni, Melvin Johnson, and Orhan Firat. 2019. Massively multilingual neural machine translation. *arXiv preprint arXiv:1903.00089*.

Maruan Al-Shedivat and Ankur P. Parikh. 2019. Consistency by agreement in zero-shot neural machine translation. *CoRR*, abs/1904.02338.

Naveen Arivazhagan, Ankur Bapna, Orhan Firat, Roee Aharoni, Melvin Johnson, and Wolfgang Macherey. 2019a. The missing ingredient in zero-shot neural machine translation. *arXiv preprint arXiv:1903.07091*.

Naveen Arivazhagan, Ankur Bapna, Orhan Firat, Dmitry Lepikhin, Melvin Johnson, Maxim Krikun, Mia Xu Chen, Yuan Cao, George Foster, Colin Cherry, Wolfgang Macherey, Zhifeng Chen, and Yonghui Wu. 2019b. Massively multilingual neural machine translation in the wild: Findings and challenges.

Mikel Artetxe and Holger Schwenk. 2018. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *arXiv preprint arXiv:1812.10464*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Anthony Bau, Yonatan Belinkov, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2018. Identifying and controlling important neurons in neural machine translation. *arXiv preprint arXiv:1811.01157*.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017. What do neural machine translation models learn about morphology? *arXiv preprint arXiv:1704.03471*.

Yonatan Belinkov, Lluís Màrquez, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2018. Evaluating layers of representation in neural machine translation on part-of-speech and semantic tagging tasks. *arXiv preprint arXiv:1801.07772*.

Mikhail Belkin and Partha Niyogi. 2003. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396.

Wayles Browne and Vyacheslav Vsevolodovich Ivanov. Slavic languages' family tree. Encyclopdia Britannica.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Zhifeng Chen, Yonghui Wu, and Macduff Hughes. 2018. The best of both worlds: Combining recent advances in neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 76–86, Melbourne, Australia. Association for Computational Linguistics.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Sandagomi Coperahawa. 2007. Language Contact and Linguistic Area: the Sinhala - Tamil Contact Situation. *Journal of the Royal Asiatic Society of Sri Lanka*, 53:133 – 152.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Yonatan Belinkov, D Anthony Bau, and James Glass. 2019. What is one grain of sand in the desert? analyzing individual neurons in deep nlp models. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 7.

Akiko Eriguchi, Melvin Johnson, Orhan Firat, Hideto Kazawa, and Wolfgang Macherey. 2018. Zero-shot cross-lingual classification using multilingual neural machine translation. *arXiv preprint arXiv:1809.04686*.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. *arXiv preprint arXiv:1601.01073*.

Harald Hammarström, Robert Forkel, and Martin Haspelmath. 2017. *Hindustani*.

David R Hardoon, Sandor Szedmak, and John Shawe-Taylor. 2004. Canonical correlation analysis: An overview with application to learning methods. *Neural computation*, 16(12):2639–2664.

Marvin Irving Herzog. West Germanic Languages.

Pavle Ivić. 2011. Encyclopaedia Britannica.

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2017. Googles multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

Moshe Koppel and Noam Ordan. 2011. Translationese and its dialects. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1318–1326. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*.

Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier.

Etukoori Balaraama Moorti. 2011. Proto Dravidian. *Study of Dravidian Linguistics and Civilization*.

Ari S. Morcos, Maithra Raghu, and Samy Bengio. 2018. Insights on representational similarity in neural networks with canonical correlation. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 5732–5741, USA. Curran Associates Inc.

Graham Neubig and Junjie Hu. 2018. Rapid adaptation of neural machine translation to new languages. *arXiv preprint arXiv:1808.04189*.

Toan Q. Nguyen and David Chiang. 2017. Transfer learning across low-resource, related languages for neural machine translation. In *Proc. IJCNLP*, volume 2, pages 296–301.

Chris Olah, Arvind Satyanarayan, Ian Johnson, Shan Carter, Ludwig Schubert, Katherine Ye, and Alexander Mordvintsev. 2018. The building blocks of interpretability. *Distill*, 3(3):e10.

A. Phillips and M Davis. 2009. Tags for Identifying Languages. RFC 5646, RFC Editor.

Alessandro Raganato and Jörg Tiedemann. 2018. An analysis of encoder representations in transformer-based machine translation. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 287–297.

Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. 2017. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In *Advances in Neural Information Processing Systems*, pages 6076–6085.

Hichem Sahbi. 2018. Learning cca representations for misaligned data. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0.

Naomi Saphra and Adam Lopez. 2018. Understanding learning dynamics of language models with svcca. *arXiv preprint arXiv:1811.00225*.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. *arXiv preprint arXiv:1804.04235*.

Mohammad Tahsin Siddiqi. 1994. *Hindustani-English code-mixing in modern literary texts*. University of Wisconsin.

Roland Sussex and Paul Cubberley. The Slavic Languages.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R Bowman, Dipanjan Das, et al. 2019. What do you learn from context? probing for sentence structure in contextualized word representations. *arXiv preprint arXiv:1905.06316*.

Jakob Uszkoreit, Jay M Ponte, Ashok C Popat, and Moshe Dubiner. 2010. Large scale parallel document mining for machine translation. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1101–1109. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Shijie Wu and Mark Dredze. 2019. Beto, bentz, becas: The surprising cross-lingual effectiveness of BERT. *CoRR*, abs/1904.09077.

Kelly W Zhang and Samuel R Bowman. 2018. Language modeling teaches you more syntax than translation does: Lessons learned through auxiliary task analysis. *arXiv preprint arXiv:1809.10040*.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1568–1575.

# Hierarchical Modeling of Global Context for Document-Level Neural Machine Translation

**Xin Tan, Longyin Zhang, Deyi Xiong, Guodong Zhou***
School of Computer Science and Technology, Soochow University, Suzhou, China
{annieT.x,zzlynx}@outlook.com
{dyxiong,gdzhou}@suda.edu.cn

## Abstract

Document-level machine translation (MT) remains challenging due to the difficulty in efficiently using document context for translation. In this paper, we propose a hierarchical model to learn the global context for document-level neural machine translation (NMT). This is done through a sentence encoder to capture intra-sentence dependencies and a document encoder to model document-level inter-sentence consistency and coherence. With this hierarchical architecture, we feedback the extracted global document context to each word in a top-down fashion to distinguish different translations of a word according to its specific surrounding context. In addition, since large-scale in-domain document-level parallel corpora are usually unavailable, we use a two-step training strategy to take advantage of a large-scale corpus with out-of-domain parallel sentence pairs and a small-scale corpus with in-domain parallel document pairs to achieve the domain adaptability. Experimental results on several benchmark corpora show that our proposed model can significantly improve document-level translation performance over several strong NMT baselines.

## 1 Introduction

Due to its flexibility and much less demand of manual efforts for feature engineering, neural machine translation (NMT) has achieved remarkable progress and become the *de-facto* standard choice in machine translation. During the last few years, a variety of NMT models have been proposed to reduce the quality gap between human translation and machine translation (Sutskever et al., 2014; Bahdanau et al., 2015; Gehring et al., 2017; Vaswani et al., 2017). Among them, the Transformer model (Vaswani et al., 2017) has achieved the state-of-the-art performance in sentence-level translation and results on news benchmark test sets have shown its "translation quality at human parity when compared to professional human translators" (Hassan et al., 2018). However, when turning to document-level translation, even the Transformer model yields a low performance as it translates each sentence in the document independently and suffers from the problem of ignoring document context.

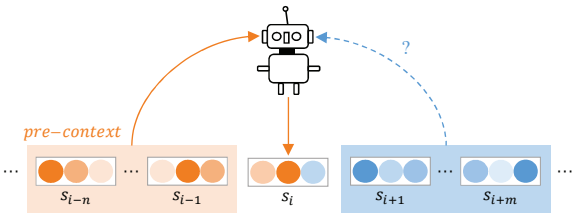


Figure 1: An illustration of document-level translation under the guidance of context.

To address above challenge, various extraction-based methods (Maruf and Haffari, 2018; Wang et al., 2017; Zhang et al., 2018; Miculicich et al., 2018) have been proposed to extract previous context (*pre-context*) to guide the translation of the current sentence $s_i$, as shown in Figure 1. However, when there exists a huge gap between the *pre-context* and the context after the current sentence $s_i$, the guidance from *pre-context* is not sufficient for the NMT model to fully disambiguate the sentence $s_i$. On the one hand, the translation of the current sentence $s_i$ may be inaccurate due to the one-sidedness of partial context. On the other hand, translating the succeeding sentences in the document may much suffer from the semantic bias due to the transmissibility of the improper *pre-context*.

To alleviate the aforementioned issues, we propose to improve document-level translation with the aid of global context, which is hierarchically extracted from the entire document with a sen-

* Corresponding author.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1576–1585, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

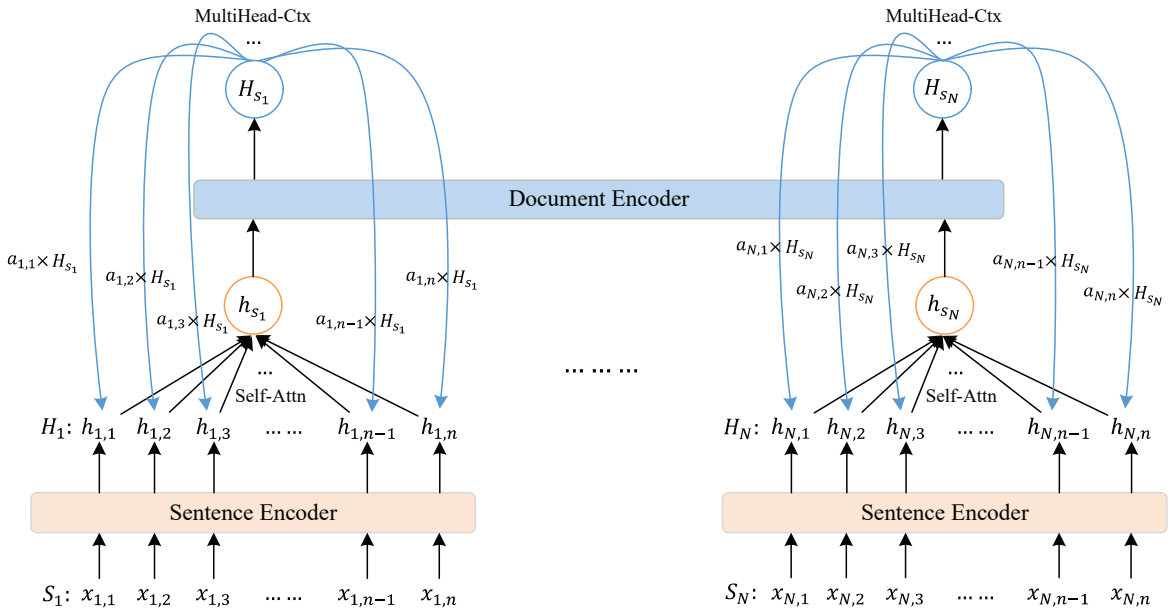


Figure 2: Diagram of the proposed hierarchical modeling of global document context (HM-GDC).

tence encoder modeling intra-sentence dependencies and a document encoder modeling document-level inter-sentence context. To avoid the issue of translation bias propagation caused by improper *pre-context*, we propose to extract global context from all sentences of a document once for all. Additionally, we propose a novel method to feed back the extracted global document context to each word in a top-to-down manner to clarify the translation of words in specific surrounding contexts. In this way, the proposed model can better translate each sentence under the guidance of the global context, thus effectively avoiding the transmissibility of improper *pre-context*. Furthermore, motivated by Zhang *et al.* (2018) and Miculicich *et al.* (2018) who exploit a large amount of sentence-level parallel pairs to improve the performance of document-level translation, we employ a two-step training strategy in taking advantage of a large-scale corpus of out-of-domain sentence-level parallel pairs to pre-train the model and a small-scale corpus of in-domain document-level parallel pairs to fine-tune the pretrained model.

We conduct experiments on both the traditional RNNSearch model and the state-of-the-art Transformer model. Experimental results on Chinese-English and German-English translation show that our proposed model can achieve the state-of-the-art performance due to its ability in well capturing global document context. It is also inferential to notice that our proposed model can explore a large dataset of out-of-domain sentence-level parallel pairs and a small dataset of in-domain document-level parallel pairs to achieve domain adaptability.

## 2 NMT with Hierarchical Modeling of Global Document Context

In this work, our ultimate goal is to incorporate the global document context into NMT to improve the performance of document-level translation. This is first achieved with the hierarchical modeling of global document context (HM-GDC) based on sentence-level hidden representation and document-level consistency and coherence modeling. Then, we integrate the proposed HM-GDC model into NMT models to help improve the performance of document-level translation.

### 2.1 Hierarchically Modeling Global Context

To avoid the one-sidedness of partial context and the transmissibility of the improper *pre-context* in previous studies, we take all sentences of the document into account and extract the global context once for all. Inspired by Sordoni *et al.* (2015), we build our HM-GDC model in a hierarchical way which contains two levels of encoder structure, i.e., the bottom sentence encoder layer to capture intra-sentence dependencies and the upper document encoder layer to capture document-level context. In this way, the global document context is captured for NMT. In order to make the translation of each word in specific surrounding context more robust, we propose to equip each word with global

document context. This is done by backpropagating the extracted global context to each word in a top-down fashion, as shown in Figure 2. The following is the detailed description of the proposed HM-GDC model.

**Sentence encoder.** Given an input document with $N$ sentences $(S_1, S_2, ..., S_N)$, the sentence encoder maps each word $x_{i,k}$ in the sentence $S_i$ into the corresponding hidden state $h_{i,k}$, obtaining:

$$H_i = \text{SentEnc}(S_i) \quad (1)$$

where $S_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,n})$ is the $i^{th}$ sentence in the document, SentEnc is the sentence encoder function (corresponding to Bi-RNNs and multi-head self-attention (Vaswani et al., 2017) for the RNNSearch model and the Transformer model respectively), and $H_i = (h_{i,1}, h_{i,2}, \ldots, h_{i,n}) \in \mathbb{R}^{D \times n}$ is the output hidden state.

**Document encoder.** Following the Transformer model (Vaswani et al., 2017), we employ the multi-head self-attention mechanism to determine the relative importance of different $H_i$. The model architecture of the document encoder here is the same as the sentence-level encoding stated before. And the document context is formulated as:

$$h_{S_i} = \text{MultiHead-Self}(H_i, H_i, H_i) \quad (2)$$

$$\tilde{h}_{S_i} = \sum\nolimits_{h \in h_{S_i}} h \quad (3)$$

$$H_S = \text{DocEnc}(\tilde{h}_S) \quad (4)$$

where MultiHead-Self$(\mathbf{Q}, \mathbf{K}, \mathbf{V})$ is the multi-head self-attention mechanism corresponding to Self-Attn in Figure 2, $h_{S_i} \in \mathbb{R}^{D \times n}$, $\tilde{h}_{S_i} \in \mathbb{R}^{D \times 1}$, $\tilde{h}_S = (\tilde{h}_{S_1}, \tilde{h}_{S_2}, \ldots, \tilde{h}_{S_N}) \in \mathbb{R}^{D \times N}$, DocEnc is the document-level encoding function (corresponding to Bi-RNNs and multi-head self-attention for the RNNSearch model and the Transformer model respectively), and $H_S = (H_{S_1}, H_{S_2}, \ldots, H_{S_N}) \in \mathbb{R}^{D \times N}$ is the global document context.

**Backpropagation of global context.** To equip each word with global document context, we propose to assign the context information to each word in the sentence using another multi-head attention (Vaswani et al., 2017), which we refer to as the multi-head context attention (MultiHead-Ctx in Figure 2). And the context information assigned to the $j^{th}$ word in the sentence $S_i$ is detailed as:

$$\alpha_{i,j} = \text{MultiHead-Ctx}(h_{i,j}, h_{i,j}, H_{S_i}) \quad (5)$$

$$d\_ctx_{i,j} = \alpha_{i,j} H_{S_i} \quad (6)$$

where $\alpha_{i,j}$ is the attention weight assigned to the word and $d\_ctx_{i,j}$ is the corresponding context information distributed to the word.

## 2.2 Integrating the HM-GDC model into NMT

Different from previous works, we equip the representation of each word with global document context. The word representations are sequential in format, which makes it easy to integrate our proposed HM-GDC model into sequential models like RNNSearch and Transformer. In this section, we mainly introduce the process of integrating HM-GDC into the state-of-the-art Transformer model.

### Integration in the Encoding Phase

Consider that the global document context is first extracted during the encoding phase and then distributed to each word in the document, as stated in Section 2.1. On this basis, we propose to employ the residual connection function (He et al., 2016) to incorporate the extracted global context information into the word representation. And the integrated representation of the $j^{th}$ word in the $i^{th}$ sentence is detailed as:

$$h\_ctx_{i,j} = h_{i,j} + \text{ResidualDrop}(d\_ctx_{i,j}, P) \quad (7)$$

where ResidualDrop is the residual connection function, $P = 0.1$ is the rate of residual dropout, $h_{i,j}$ is the corresponding hidden state of the word during the sentence encoding phase, $d\_ctx_{i,j}$ is the global document context assigned to the word, and $h\_ctx_{i,j}$ is the integrated representation of the word.

### Integration in the Decoding Phase

With the help of the multi-head attention sub-layer in the decoder, the Transformer model is capable of well employing the information obtained from the encoder. Inspired by this, we introduce an additional sub-layer into the decoder that performs multi-head attention over the output of the document encoder, which we refer to as DocEnc-Decoder attention (shown in Figure 3). Different from (Vaswani et al., 2017), the keys and values of our DocEnc-Decoder attention come from the output of the document encoder. In this way, the global document context is well employed to supervise the decoding process. And Specially,

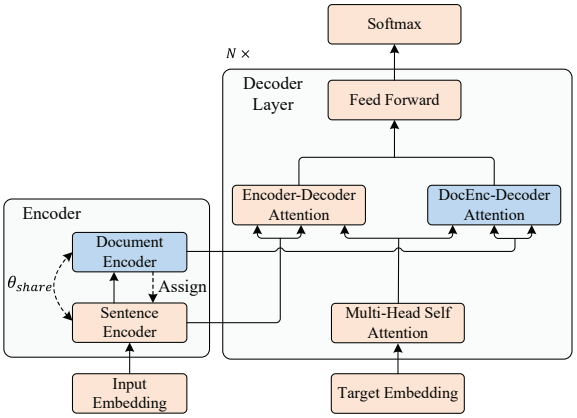


Figure 3: Integration of global document context into the decoder of the Transformer model.

the additional DocEnc-Decoder attention is formulated as:

$$C = [h\_ctx_1; h\_ctx_2; ...; h\_ctx_N] \quad (8)$$

$$G^{(n)} = \text{MultiHead-Attn}(T^{(n)}, C^{(n)}, C^{(n)}) \quad (9)$$

where $h\_ctx_i$ denotes the integrated representation of the $i^{th}$ sentence, $C^{(0)} = C$ is the concatenated global document context, $T^{(n)}$ is the output of the multi-head self-attention sub-layer in the decoder. On this basis, we combine the outputs of both the Encoder-Decoder attention sub-layer and the DocEnc-Decoder attention sub-layer into one single output $H^{(n)}$:

$$H^{(n)} = E^{(n)} + G^{(n)} \quad (10)$$

where $E^{(n)}$ is the output of the Encoder-Decoder attention sub-layer, and $G^{(n)}$ is the output of the DocEnc-Decoder attention sub-layer.

### 2.3 Model Training

In document-level translation, the standard training objective is to maximize the log-likelihood of the document-level parallel corpus. However, due to the size limitation of document-level parallel corpora, previous studies (Zhang et al., 2018; Miculicich et al., 2018; Shen et al., 2016) use two-step training strategies to take advantage of large-scale sentence-level parallel pairs. Following their studies, we also take a two-step training strategy to train our model. Specially, we borrow a large-scale corpus with out-of-domain sentence-level parallel pairs $D_s$ to pre-train our model first, and then use a small-scale corpus with in-domain document-level parallel pairs $D_d$ to fine-tune it.

In this work, we follow Voita *et al.* (2018) to make the sentence and document encoders share the same model parameters. For the RNNSearch model, we share the parameters in the hidden layers of Bi-RNNs in the sentence and document encoders. For the Transformer model, we share the parameters of the multi-head self-attention layers in the sentence and document encoders.

During training, we first optimize the sentence-level parameters $\theta_s$ (colored in wheat in Figure 3) with the large-scale sentence-level parallel pairs $D_s$:

$$\hat{\theta}_s = \arg\max_{\theta_s} \sum_{<x,y>\in D_s} \log P(y|x;\theta_s) \quad (11)$$

Then, we optimize the document-level parameters $\theta_d$ (colored in pale blue in Figure 3) with the document-level parallel corpus $D_d$ and fine-tune the pre-trained sentence-level parameters $\hat{\theta}_s$ as follows:

$$\hat{\theta}_d = \arg\max_{\theta_d} \sum_{<x,y>\in D_d} \log P(y|x;\theta_d,\hat{\theta}_s) \quad (12)$$

## 3 Experimentation

To examine the effect of our proposed HM-GDC model, we conduct experiments on both Chinese-English and German-English translation.

### 3.1 Experimental Settings

#### Datasets

For Chinese-English translation, we carry out experiments with sentence- and document-level corpora on two different domains: news and talks. For the sentence-level corpus, we use 2.8M sentence pairs from news corpora LDC2003E14, LDC2004T07, LDC2005T06, LDC2005T10 and LDC2004T08 (Hongkong Hansards/Laws/News). We use the Ted talks corpus from the IWSLT 2017 (Cettolo et al., 2012) evaluation campaigns[1] as the document-level parallel corpus, including 1,906 documents with 226K sentence pairs. We use *dev2010* which contains 8 documents with 879 sentence pairs for development and *tst2012-tst2015* which contain 62 documents with 5566 sentence pairs for testing.

For German-English translation, we use the document-level parallel Ted talks corpus from the IWSLT 2014 (Cettolo et al., 2012) evaluation campaigns, which contain 1,361 documents with 172K

[1] https://wit3.fbk.eu

| Method | *pre-training* | tst12 | tst13 | tst14 | tst15 | Avg |
|---|---|---|---|---|---|---|
| RNNSearch (Bahdanau et al., 2015) | × | 15.50 | 16.40 | 14.69 | 16.88 | 15.87 |
| Wang *et al.* (2017) | × | 15.90 | 16.99 | 14.50 | 17.33 | 16.18 |
| Ours | × | 16.33 | 17.52 | 15.80 | 18.10 | **16.94** |
| Transformer (Vaswani et al., 2017) | × | 15.87 | 16.51 | 14.67 | 17.27 | 16.08 |
| Zhang *et al.* (2018) | × | 11.31 | 12.58 | 10.22 | 13.35 | 11.87 |
| Ours | × | 16.58 | 17.03 | 15.22 | 17.96 | **16.70** |
| Transformer (Vaswani et al., 2017) | ✓ | 14.63 | 16.72 | 14.43 | 16.25 | 15.51 |
| Zhang *et al.* (2018) | ✓ | 16.46 | 17.80 | 15.85 | 18.24 | 17.09 |
| Ours | ✓ | 16.94 | 18.31 | 16.21 | 19.07 | **17.63** |

Table 1: Performance (BLEU scores) comparison with the four baseline models on Chinese-English document-level translation. The first three rows are based on RNNSearch and the remaining rows are on top of Transformer. And the *p-value* between Ours and the other models are all less than 0.01.

sentence pairs as training data. We use *dev2012* which contains 7 documents with 1172 sentence pairs for development and *tst2013-tst2014* which contain 31 documents with 2329 sentence pairs for testing.

**Model Settings**

We integrate our proposed HM-GDC into the original Transformer model implemented by OpenNMT (Klein et al., 2017). Following the Transformer model (Vaswani et al., 2017), the hidden size and filter size are set to 512 and 2048 respectively. The numbers of layers in encoder and decoder are all set to 6. The multi-head attention mechanism of each layer contains 8 individual attention heads. We set both the source and target vocabulary size as 50K and each batch contains 4096 tokens. The beam size and dropout (Srivastava et al., 2014) rate are set to 5 and 0.1 respectively. Other settings with the Adam (Kingma and Ba, 2015) optimization and regularization methods are the same as the default Transformer model.

To comprehensively evaluate the performance of our proposed HM-GDC model, we integrate the HM-GDC into the standard RNNSearch model to serve as a supplementary experiment. For the RNNSearch network, we borrow the implementation from OpenNMT (Klein et al., 2017). The encoder and decoder layers are all set to 2, the size of the hidden layer is set to 500, and the batch size is set to 64. Same as the Transformer model, we use the most frequent 50K words for both source and target vocabularies. We borrow other settings from (Bahdanau et al., 2015). The evaluation metric for both tasks is case-insensitive BLEU (multi-bleu) (Papineni et al., 2002).

### 3.2 Experimental Results

In this paper, we compare our model with four strong baselines as shown in Table 1. Among them, the RNNSearch (Bahdanau et al., 2015) is a traditional RNN-based encoder-decoder model. Wang *et al.* (2017) propose to use a hierarchical model to extract partial document context based on the RNNSearch model. To compare with these two RNN-based works, we integrate our proposed HM-GDC model into the encoder of the RNNSearch model using the same method in Section 2.2 and keep other settings the same as the basic RNNSearch model. Different from RNN-based works, Vaswani *et al.* (2017) propose the Transformer model, which achieves the state-of-the-art performance in sentence-level translation with only attention mechanism. On this basis, Zhang *et al.* (2018) add an additional multi-head attention to extract partial document context to improve the Transformer model in document-level translation. In particular, Zhang *et al.* (2018) use a two-step training strategy in their work, so we also report the performance comparison with respect to the training strategy. For the RNNSearch and Transformer models, we run them with their default settings. And we reimplement the models of Wang *et al.* (2017) and Zhang *et al.* (2018) to conduct experiments on our datasets.

As shown in Table 1, we divide the results into two main groups, i.e., in the framework of RNNSearch (the first three rows) and Transformer (the remaining rows). The results in the first group reveal that our proposed model can significantly improve the RNNSearch model and can further improve the model of Wang *et al.* (2017) in

document-level translation by 0.76 BLEU points. In addition, the results in the second group is further split into two parts depending on whether the *pre-training* strategy is used. For the first part, we train our model and the two baselines with only the small-scale document-level parallel corpus without *pre-training* (the first three rows in the second group). From the results, the model of Zhang *et al.* (2018) achieves much worse results ($-4.21$ BLEU points) when compared with the standard Transformer model, which is consistent with what they state in their paper. By contrast, our proposed model can achieve 0.62 BLEU points over the Transformer model, which indicates the robustness of our model. For the second part, to further compare with (Zhang et al., 2018), we use the *pre-training* strategy to take advantage of large-scale sentence-level parallel corpus $D_s$ for these models (the last three rows). From the results, our proposed HM-GDC can significantly improve the performance of the Transformer model by 2.12 BLEU points and can further improve the performance of (Zhang et al., 2018) by 0.54 BLEU points.

From the overall results, it's not difficult to find that using a large-scale corpus with out-of-domain parallel pairs $D_s$ to pre-train the Transformer model results in worse performance due to domain inadaptability (the first and the fourth row in the second group). By contrast, our proposed model can effectively eliminate this domain inadaptability (the third and sixth row in the second group). In general, our proposed HM-GDC model is robust when integrated into frameworks like RNNSearch and Transformer and it can help improve the performance of document-level translation.

## 4 Analysis and Discussion

To further demonstrate the effectiveness of our proposed HM-GDC model, we illustrate several experimental results in this section and give our analysis on them.

### 4.1 The Effect of HM-GDC Integration

As the state-of-the-art Transformer model is well-designed in the model structure, an analysis of the integration in Transformer is thus necessary. Therefore, we perform experiments on Chinese-English document-level translation for analyzing. Table 2 illustrates the effectiveness of integrating our proposed HM-GDC model into the encoder, decoder and both sides of the Transformer model with respect to the number of layers in the multi-head self-attention in the document encoder (see Section 2.1).

| $N$ | Encoder | Decoder | Both |
|---|---|---|---|
| 1 | 17.34 | **17.54** | 17.49 |
| 2 | 17.30 | 17.43 | 17.56 |
| 3 | **17.39** | **17.54** | 17.45 |
| 4 | 17.27 | 17.45 | 17.55 |
| 5 | 17.31 | 17.49 | **17.63** |
| 6 | 17.25 | 17.40 | 17.58 |

Table 2: The effect of integrating HM-GDC into Transformer with respect to the layer number ($N$) of the self-attention in the document encoder. The results here refer to the average BLEU scores of test sets.

| Model | tst13 | tst14 | Avg |
|---|---|---|---|
| Baseline | 27.89 | 23.75 | 25.82 |
| Ours | 28.58 | 24.85 | **26.72** |

Table 3: Comparison with the Transformer model on German-English document-level translation. And the *p-value* between Ours and Baseline is less than 0.01.

From the results, the overall performance of integrating HM-GDC into both the encoder and decoder is better than integrating it into the encoder or decoder only. However, the layer number of the multi-head self-attention does not make much difference in our experiments. It shows that when the HM-GDC is integrated into both the encoder and decoder and the layer number equals to 5, the Transformer model can achieve a relatively better performance.

### 4.2 Different Language Pairs

In this paper, we aim to propose a robust document context extraction model. To achieve this goal, we perform experiments on different language pairs to further illustrate the effectiveness of our proposed HM-GDC model. Table 3 shows the performance of our model on German-English document-level translation and the baseline here refers to the Transformer model. For clarity, we only use the German-English document-level parallel corpus to train these two models without *pre-training*. From the results, our proposed HM-GDC model can help improve the Transformer model on German-English document-level translation by 0.90 BLEU points. The experimental re-

| *sent* | *doc* | tst12 | tst13 | tst14 | tst15 | Avg |
|---|---|---|---|---|---|---|
| × | × | 16.58 | 17.03 | 15.22 | 17.96 | 16.70 |
| ✓ | × | 16.94 | 18.31 | 16.21 | 19.07 | **17.63** |
| ✓ | ✓ | 20.08 | 21.04 | 19.48 | 22.46 | **20.77** |

Table 4: Results of our model with and without *pre-training* on Chinese-English document-level translation. *sent* refers to the sentence-level parallel corpus and *doc* refers to the document-level parallel corpus. ✓means that the corresponding corpus is used for pre-training while × means not.

| | Chinese-English | | | | | German-English | | |
|---|---|---|---|---|---|---|---|---|
| Model | tst12 | tst13 | tst14 | tst15 | Avg | tst13 | tst14 | Avg |
| Baseline | 58.06 | 55.10 | 51.71 | 52.48 | 54.34 | 86.47 | 84.78 | 85.63 |
| Ours | 59.29 | 55.48 | 52.95 | 53.22 | **55.24** | 87.36 | 85.64 | **86.50** |

Table 5: Evaluation on pronoun translation of Chinese-English and German-English document-level translation. The baseline model refers to the Transformer.

sults further validate the robustness of our model in different language pairs.

### 4.3 The Effect of Pre-training

Due to the size limitation of document-level parallel corpora, previous studies (Zhang et al., 2018; Miculicich et al., 2018; Shen et al., 2016) use two-step training strategies to take advantage of a large-scale corpus with sentence-level parallel pairs. Inspired by them, we use a two-step training strategy to train our model, which we refer to as the *pre-training* strategy (see Section 2.3). In this section, we perform the *pre-training* strategy on the HM-GDC integrated Transformer model to further analysis the ability of our model in utilizing resources of different domains. The results of our model with and without the *pre-training* strategy are shown in Table 4. The first row in the table gives the result of our model without *pre-training*, where only the talks-domain document-level parallel corpus $D_d$ are used to train the model. The remaining rows give the results of our model with the *pre-training* strategy, where we first use the large-scale sentence-level parallel pairs $D_s$ to pre-train the model and then the small-scale talks-domain document-level parallel corpus $D_d$ to fine-tune the entire model.

From the results, the performance of our model is significantly improved by 0.93 BLEU points (the first two rows in Table 4) when the large-scale sentence-level parallel corpus is used for the pre-training process. In particular, when we use the mixed data of both sentence- and document-level parallel corpora[2] to first pre-train our model, the performance of our model is significantly improved by 5.26 BLEU points (the last row in Table 4). The overall results prove that our proposed model is robust and promising. It can significantly improve the performance of document-level translation when a two-step training strategy is used.

### 4.4 Pronoun & Noun Translation

To intuitively illustrate how the translation performance is improved by our proposed HM-GDC model, we conduct a further analysis on pronoun and noun translation.

For the pronoun translation, we evaluate the coreference and anaphora using the reference-based metric: the accuracy of pronoun translation (Miculicich Werlen and Popescu-Belis, 2017) in Chinese-English and German-English translation as shown in Table 5. From the results, our proposed HM-GDC model can well improve the performance of pronoun translation in both corpora due to the well captured global document context assigned to each word. Correspondingly, we display a translation example in Table 6 to further illustrate this. From the example, given the surrounding context, our proposed HM-GDC model can well infer the latent pronoun *it* and thus improve the translation performance of the Transformer model.

For the analysis of noun translation, we display another example in Table 7. From the example,

[2]We shuffle sentences in $D_d$ to get sentence-level parallel pairs.

| | |
|---|---|
| Source | dan xifang zhengfu ye tongyang dui tamen ziji zheyang zuo **ta**. |
| Reference | But western governments are doing **it** to themselves as well. |
| Baseline | But the western governments do the same for themselves. |
| Ours | But the western governments do **it** the same for themselves. |

Table 6: An example of pronoun translation in the Chinese-English document-level translation. The word "ta" in bold in the source sentence is an omitted pronoun.

| | |
|---|---|
| Pre-context | $\cdots$ renhe yige youxiu de **chengxuyuan** dou hui zuo de $\cdots$ |
| Rear-context | $\cdots$ zai xiaweiyi de **IT** bumen shangban de ren kandao le $\cdots$ |
| Source | ta xie le yige xiangduilaishuo bijiao xiao de **chengxu** |
| Reference | he wrote a modest little **app** |
| Baseline | he wrote a relatively small **procedure** |
| Ours | he wrote a relative smaller **program** |

Table 7: An example of noun translation in the Chinese-English document-level translation. The baseline model here refers to the Transformer.

the word *chengxu* is translated into *procedure* and *program* by the Transformer model and the model integrated with HM-GDC respectively. Comparing with the reference translation, the word *program* translated by our model is more appropriate. Although the words *chengxuyuan* and *IT* in the global context provide essential evidence for an accurate translation of *chengxu*, it is hard for the baseline model to obtain the information. Different from previous works which do not use or use only partial document context, we propose to incorporate the HM-GDC into the NMT model to take global context into consideration and thus it can safely disambiguate those multi-sense words like *chengxu*.

## 5 Related Work

Recent years have witnessed a variety of approaches proposed for document-level machine translation. Most of existing studies aim to improve overall translation quality with the aid of document context. Among them, Maruf and Haffrai (2018), Wang *et al.* (2017), Zhang *et al.* (2018) and Miculicich *et al.* (2018) use extraction-based models to extract partial document context from previous sentences of the current sentence. In addition, Tu *et al.* (2018) and Kuang *et al.* (2018) employ cache-based models to selectively memorize the most relevant information in the document context. Different from above extraction-based models and cache-based models, there are also some works (Bawden et al., 2018; Voita et al., 2018) that pay much attention to discourse phenomena (Mitkov, 1999) related to document-level translation.

Although these approaches have achieved some progress in document-level machine translation, they still suffer from incomplete document context. Further more, most of previous works are based on the RNNSearch model, and only few exceptions (Zhang et al., 2018; Miculicich et al., 2018) are on top of the state-of-the-art Transformer model.

## 6 Conclusion

We have presented a hierarchical model to capture the global document context for document-level NMT. The proposed model can be integrated into both the RNNSearch and the state-of-the-art Transformer frameworks. Experiments on two benchmark corpora show that our proposed model can significantly improve document-level translation performance over several strong document-level NMT baselines. Additionally, we observe that pronoun and noun translations are significantly improved by our proposed HM-GDC model. In our future work, we plan to enrich our HM-GDC model to solve discourse phenomena such as (zero) anaphora.

## Acknowledgments

This work was supported by the National Natural Science Foundation of China (NSFC) via Grant Nos. 61751206, 61673290, 61622209 and A Project Funded by the Priority Academic Program Development of Jiangsu Higher Education

Institutions (PAPD). Also, we would like to thank the anonymous reviewers for their insightful comments.

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *In Proceedings of the Third International Conference on Learning Representations (ICLR2015)*.

Rachel Bawden, Rico Sennrich, Alexandra Birch, and Barry Haddow. 2018. Evaluating discourse phenomena in neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1304–1313, New Orleans, Louisiana. Association for Computational Linguistics.

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. Wit$^3$: Web inventory of transcribed and translated talks. In *Proceedings of the 16$^{th}$ Conference of the European Association for Machine Translation (EAMT)*, pages 261–268, Trento, Italy.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1243–1252. JMLR. org.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, Shujie Liu, Tie-Yan Liu, Renqian Luo, Arul Menezes, Tao Qin, Frank Seide, Xu Tan, Fei Tian, Lijun Wu, Shuangzhi Wu, Yingce Xia, Dongdong Zhang, Zhirui Zhang, and Ming Zhou. 2018. Achieving human parity on automatic chinese to english news translation. *CoRR*, abs/1803.05567.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada. Association for Computational Linguistics.

Shaohui Kuang, Deyi Xiong, Weihua Luo, and Guodong Zhou. 2018. Modeling coherence for neural machine translation with dynamic and topic caches. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 596–606, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Sameen Maruf and Gholamreza Haffari. 2018. Document context neural machine translation with memory networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL) (Volume 1: Long Papers)*, pages 1275–1284, Melbourne, Australia. Association for Computational Linguistics.

Lesly Miculicich, Dhananjay Ram, Nikolaos Pappas, and James Henderson. 2018. Document-level neural machine translation with hierarchical attention networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2947–2954, Brussels, Belgium. Association for Computational Linguistics.

Lesly Miculicich Werlen and Andrei Popescu-Belis. 2017. Validation of an automatic metric for the accuracy of pronoun translation (APT). In *Proceedings of the Third Workshop on Discourse in Machine Translation*, pages 17–25, Copenhagen, Denmark. Association for Computational Linguistics.

Ruslan Mitkov. 1999. Introduction: special issue on anaphora resolution in machine translation and multilingual nlp. *Machine translation*, 14(3-4):159–161.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL) (Volume 1: Long Papers)*, pages 1683–1692, Berlin, Germany. Association for Computational Linguistics.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 553–562. ACM.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Zhaopeng Tu, Yang Liu, Shuming Shi, and Tong Zhang. 2018. Learning to remember translation history with a continuous cache. *Transactions of the Association for Computational Linguistics*, 6:407–420.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Elena Voita, Pavel Serdyukov, Rico Sennrich, and Ivan Titov. 2018. Context-aware neural machine translation learns anaphora resolution. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL) (Volume 1: Long Papers)*, pages 1264–1274, Melbourne, Australia. Association for Computational Linguistics.

Longyue Wang, Zhaopeng Tu, Andy Way, and Qun Liu. 2017. Exploiting cross-sentence context for neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2826–2831, Copenhagen, Denmark. Association for Computational Linguistics.

Jiacheng Zhang, Huanbo Luan, Maosong Sun, Feifei Zhai, Jingfang Xu, Min Zhang, and Yang Liu. 2018. Improving the transformer translation model with document-level context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 533–542, Brussels, Belgium. Association for Computational Linguistics.

# Cross-Lingual Machine Reading Comprehension

**Yiming Cui[†‡], Wanxiang Che[†], Ting Liu[†], Bing Qin[†], Shijin Wang[‡§], Guoping Hu[‡]**
[†]Research Center for Social Computing and Information Retrieval (SCIR),
Harbin Institute of Technology, Harbin, China
[‡]State Key Laboratory of Cognitive Intelligence, iFLYTEK Research, China
[§]iFLYTEK AI Research (Hebei), Langfang, China
[†]{ymcui,car,tliu,qinb}@ir.hit.edu.cn
[‡§]{ymcui,sjwang3,gphu}@iflytek.com

## Abstract

Though the community has made great progress on Machine Reading Comprehension (MRC) task, most of the previous works are solving English-based MRC problems, and there are few efforts on other languages mainly due to the lack of large-scale training data. In this paper, we propose Cross-Lingual Machine Reading Comprehension (CLMRC) task for the languages other than English. Firstly, we present several back-translation approaches for CLMRC task, which is straightforward to adopt. However, to accurately align the answer into another language is difficult and could introduce additional noise. In this context, we propose a novel model called Dual BERT, which takes advantage of the large-scale training data provided by rich-resource language (such as English) and learn the semantic relations between the passage and question in a bilingual context, and then utilize the learned knowledge to improve reading comprehension performance of low-resource language. We conduct experiments on two Chinese machine reading comprehension datasets CMRC 2018 and DRCD. The results show consistent and significant improvements over various state-of-the-art systems by a large margin, which demonstrate the potentials in CLMRC task. [1]

## 1 Introduction

Machine Reading Comprehension (MRC) has been a popular task to test the reading ability of the machine, which requires to read text material and answer the questions based on it. Starting from cloze-style reading comprehension, various neural network approaches have been proposed and massive progresses have been made in creating large-scale datasets and neural models (Hermann et al., 2015; Hill et al., 2015; Kadlec et al., 2016; Cui et al., 2017; Rajpurkar et al., 2016; Dhingra et al., 2017). Though various types of contributions had been made, most works are dealing with English reading comprehension. Reading comprehension in other than English has not been well-addressed mainly due to the lack of large-scale training data.

To enrich the training data, there are two traditional approaches. Firstly, we can annotate data by human experts, which is ideal and high-quality, while it is time-consuming and rather expensive. One can also obtain large-scale automatically generated data (Hermann et al., 2015; Hill et al., 2015; Liu et al., 2017), but the quality is far beyond the usable threshold. Another way is to exploit cross-lingual approaches to utilize the data in rich-resource language to implicitly learn the relations between <passage, question, answer>.

In this paper, we propose the Cross-Lingual Machine Reading Comprehension (CLMRC) task that aims to help reading comprehension in low-resource languages. First, we present several back-translation approaches when there is no or partially available resources in the target language. Then we propose a novel model called Dual BERT to further improve the system performance when there is training data available in the target language. We first translate target language training data into English to form pseudo bilingual parallel data. Then we use multilingual BERT (Devlin et al., 2019) to simultaneously model the <passage, question, answer> in both languages, and fuse the representations of both to generate final predictions. Experimental results on two Chinese reading comprehension dataset CMRC 2018 (Cui et al., 2019) and DRCD (Shao et al., 2018) show that by utilizing English resources could substantially improve system performance and the proposed Dual BERT achieves state-of-the-art performances on both datasets, and even surpass human performance on some metrics. Also, we

[1]Resources available: https://github.com/ymcui/Cross-Lingual-MRC.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1586–1595,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

conduct experiments on the Japanese and French SQuAD (Asai et al., 2018) and achieves substantial improvements. Moreover, detailed ablations and analysis are carried out to demonstrate the effectiveness of exploiting knowledge from rich-resource language. To best of our knowledge, this is the first time that the cross-lingual approaches applied and evaluated on realistic reading comprehension data. The main contributions of our paper can be concluded as follows.

- We present several back-translation based reading comprehension approaches and yield state-of-the-art performances on several reading comprehension datasets, including Chinese, Japanese, and French.
- We propose a model called Dual BERT to simultaneously model the <passage, question> in both source and target language to enrich the text representations.
- Experimental results on two public Chinese reading comprehension datasets show that the proposed cross-lingual approaches yield significant improvements over various baseline systems and set new state-of-the-art performances.

## 2 Related Works

Machine Reading Comprehension (MRC) has been a trending research topic in recent years. Among various types of MRC tasks, span-extraction reading comprehension has been enormously popular (such as SQuAD (Rajpurkar et al., 2016)), and we have seen a great progress on related neural network approaches (Wang and Jiang, 2016; Seo et al., 2016; Xiong et al., 2016; Cui et al., 2017; Hu et al., 2019), especially those were built on pre-trained language models, such as BERT (Devlin et al., 2019). While massive achievements have been made by the community, reading comprehension in other than English has not been well-studied mainly due to the lack of large-scale training data.

Asai et al. (2018) proposed to use runtime machine translation for multilingual extractive reading comprehension. They first translate the data from the target language to English and then obtain an answer using an English reading comprehension model. Finally, they recover the corresponding answer in the original language using soft-alignment attention scores from the NMT model. However, though an interesting attempt has been made, the zero-shot results are quite low, and alignments between different languages, especially for those have different word orders, are significantly different. Also, they only evaluate on a rather small dataset (hundreds of samples) that was translated from SQuAD (Rajpurkar et al., 2016), which is not that realistic.

To solve the issues above and better exploit large-scale rich-resourced reading comprehension data, in this paper, we propose several zero-shot approaches which yield state-of-the-art performances on Japanese and French SQuAD data. Moreover, we also propose a supervised approach for the condition that there are training samples available for the target language. To evaluate the effectiveness of our approach, we carried out experiments on two realistic public Chinese reading comprehension data: CMRC 2018 (simplified Chinese) (Cui et al., 2019) and DRCD (traditional Chinese) (Shao et al., 2018). Experimental results demonstrate the effectiveness by modeling training samples in a bilingual environment.

## 3 Back-Translation Approaches

In this section, we illustrate back-translation approaches for cross-lingual machine reading comprehension, which is natural and easy to implement. Before introducing these approaches in detail, we will clarify crucial terminologies in this paper for better understanding.

- **Source Language**: Rich-resourced and has sufficient large-scale training data that we aim to extract knowledge from. We use subscript $S$ for variables in the source language.
- **Target Language**: Low-resourced and has only a few training data that we wish to optimize on. We use subscript $T$ for variables in the target language.

In this paper, we aim to improve the machine reading comprehension performance in Chinese (target language) by introducing English (source language) resources. The general idea of back-translation approaches is to translate <passage, question> pair into the source language and generate an answer using a reading comprehension system in the source language. Finally, the generated answer is back-translated into the target language. In the following subsections, we will introduce several back-translation approaches

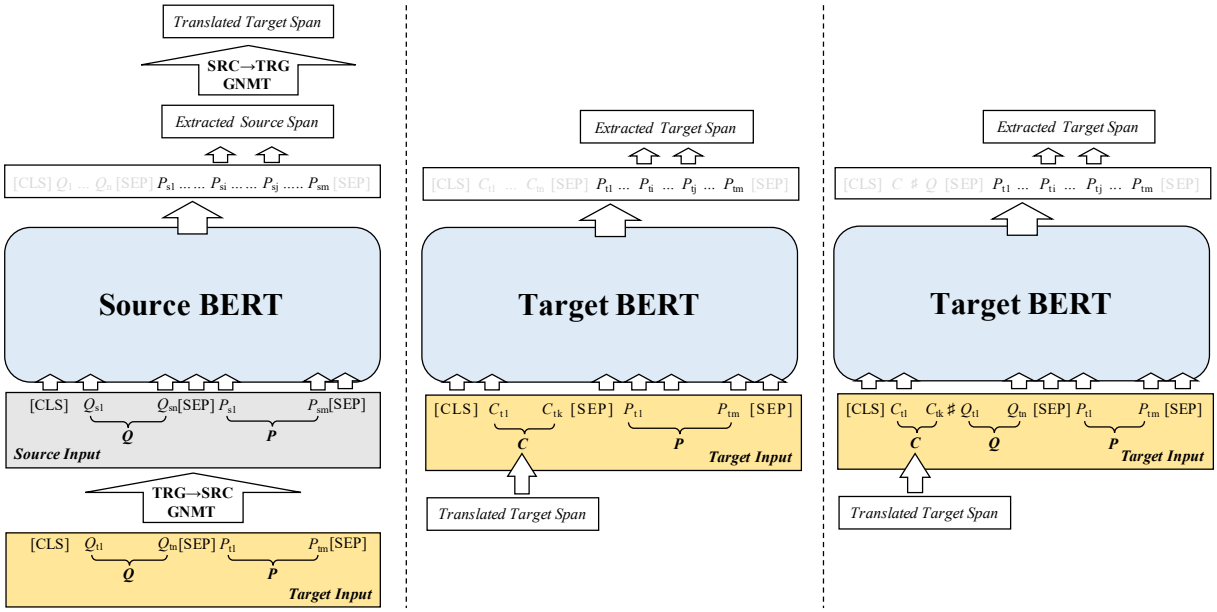


Figure 1: Back-translation approaches for cross-lingual machine reading comprehension (Left: GNMT, Middle: Answer Aligner, Right: Answer Verifier)

for cross-lingual machine reading comprehension task. The architectures of the proposed back-translation approaches are depicted in Figure 1.

### 3.1 GNMT

To build a simple cross-lingual machine reading comprehension system, it is straightforward to utilize translation system to bridge source and target language (Asai et al., 2018). Briefly, we first translate the target sample to the source language. Then we use a source reading comprehension system, such as BERT (Devlin et al., 2019), to generate an answer in the source language. Finally, we use back-translation to get the answer in the target language. As we do not exploit any training data in the target language, we could regard this approach as a *zero-shot* cross-lingual baseline system.

Specifically, we use Google Neural Machine Translation (GNMT) system for source-to-target and target-to-source translations. One may also use advanced and domain-specific neural machine translation system to achieve better translation performance, while we leave it for individuals, and this is beyond the scope of this paper.

However, for span-extraction reading comprehension task, a major drawback of this approach is that the translated answer may not be the exact span in the target passage. To remedy this, we propose three simple approaches to improve the quality of the translated answer in the target language.

### 3.2 Simple Match

We propose a simple approach to align the translated answer into extract span in the target passage. We calculate character-level text overlap (for Chinese) between translated answer $A_{trans}$ and arbitrary sliding window in target passage $\mathcal{P}_{T[i:j]}$. The length of sliding window ranges $len(A_{trans}) \pm \delta$, with a relax parameter $\delta$. Typically, the relax parameter $\delta \in [0, 5]$ as the length between ground truth and translated answer does not differ much in length. In this way, we would calculate character-level F1-score of each candidate span $\mathcal{P}_{T[i:j]}$ and translated answer $A_{trans}$, and we could choose the best matching one accordingly. Using the proposed SimpleMatch could ensure the predicted answer is an exact span in target passage. As SimpleMatch does not use target training data either, it could also be a pipeline component in *zero-shot* settings.

### 3.3 Answer Aligner

Though we could use unsupervised approaches for aligning answer, such as the proposed SimpleMatch, it stops at token-level and lacks semantic awareness between the translated answer and ground truth answer. In this paper, we also propose two supervised approaches for further improving the answer span when there is training data available in the target language.

The first one is Answer Aligner, where we feed

translated answer $\mathcal{A}_{trans}$ and target passage $\mathcal{P}_T$ into the BERT and outputs the ground truth answer span $\mathcal{A}_T$. The model will learn the semantic relations between them and generate improved span for the target language.

### 3.4 Answer Verifier

In Answer Aligner, we did not exploit question information in target training data. One can also utilize question information to transform Answer Aligner into Answer Verifier, as we use complete $\langle \mathcal{P}_T, \mathcal{Q}_T, \mathcal{A}_T \rangle$ in the target language and additional translated answer $\mathcal{A}_{trans}$ to verify its correctness and generate improved span.

## 4 Dual BERT

One disadvantage of the back-translation approaches is that we have to recover the source answer into the target language. To remedy the issue, in this paper, we propose a novel model called Dual BERT to simultaneously model the training data in both source and target language to better exploit the relations among <passage, question, answer>. The model could be used when there is training data available for the target language, and we could better utilize source language data to enhance the target reading comprehension system. The overall neural architecture for Dual BERT is shown in Figure 2.

### 4.1 Dual Encoder

Bidirectional Encoder Representation from Transformers (BERT) has shown marvelous performance in various NLP tasks, which substantially outperforms non-pretrained models by a large margin (Devlin et al., 2019). In this paper, we use multi-lingual BERT for better encoding the text in both source and target language. Formally, given target passage $\mathcal{P}_T$ and question $\mathcal{Q}_T$, we organize the input $X_T$ for BERT as follows.

$$\texttt{[CLS]}\ \mathcal{Q}_T\ \texttt{[SEP]}\ \mathcal{P}_T\ \texttt{[SEP]}$$

Similarly, we can also obtain source training sample by translating target sample with GNMT, forming input $X_S$ for BERT. Then we use $X_T$ and $X_S$ to obtain deep contextualized representations through a *shared* multi-lingual BERT, forming $B_T \in \mathbb{R}^{L_T * h}, B_S \in \mathbb{R}^{L_S * h}$, where $L$ represents the length of input and $h$ is the hidden size (768 for multi-lingual BERT).

### 4.2 Bilingual Decoder

Typically, in the reading comprehension task, attention mechanism is used to measure the relations between the passage and question. Moreover, as Transformers are fundamental components of BERT, multi-head self-attention layer (Vaswani et al., 2017) is used to extract useful information within the input sequence.

Specifically, in our model, to enhance the target representation, we use a multi-head self-attention layer to extract useful information in source BERT representation $B_S$. We aim to generate target span by not only relying on target representation but also on source representation to simultaneously consider the <passage, question> relations in both languages, which can be seen as a bilingual decoding process.

Briefly, we regard target BERT representation $B_T$ as *query* and source BERT representation $B_S$ as *key* and *value* in multi-head attention mechanism. In original multi-head attention, we calculate a raw dot attention as follows. [2] This will result in an attention matrix $A_{TS}$ that indicate raw relations between each source and target token.

$$A_{TS} = B_T \cdot B_S^\top, \quad A_{TS} \in \mathbb{R}^{L_T * L_S} \tag{1}$$

To combine the benefit of both inter-attention and self-attention, instead of using Equation 1, we propose a simple modification on multi-head attention mechanism, which is called **Self-Adaptive Attention (SAA)**. First, we calculate self-attention of $B_T$ and $B_S$ and apply the softmax function, as shown in Equation 2 and 3. This is designed to use self-attention to filter the irrelevant part within each representation firstly, and inform the raw dot attention on paying more attention to the self-attended part, making the attention more precise and accurate.

$$A_T = \mathbf{softmax}(B_T \cdot B_T^\top) \tag{2}$$

$$A_S = \mathbf{softmax}(B_S \cdot B_S^\top) \tag{3}$$

Then we use self-attention $A_T$ and $A_S$, inter-attention $A_{TS}$ to get self-attentive attention $\tilde{A}_{TS}$. We calculate dot product between $A_{ST}$ and $B_S$ to obtain attended representation $R' \in \mathbb{R}^{L_T * h}$.

$$\tilde{A}_{TS} = A_T \cdot A_{TS} \cdot {A_S}^\top, \tilde{A}_{TS} \in \mathbb{R}^{L_T * L_S} \tag{4}$$

$$R' = \mathbf{softmax}(\tilde{A}_{TS}) \cdot B_S \tag{5}$$

[2] We omit rather extensive formulations of representation transformations and kindly advise the readers refer to the attention implementation in BERT: https://github.com/google-research/bert/blob/master/modeling.py#L558

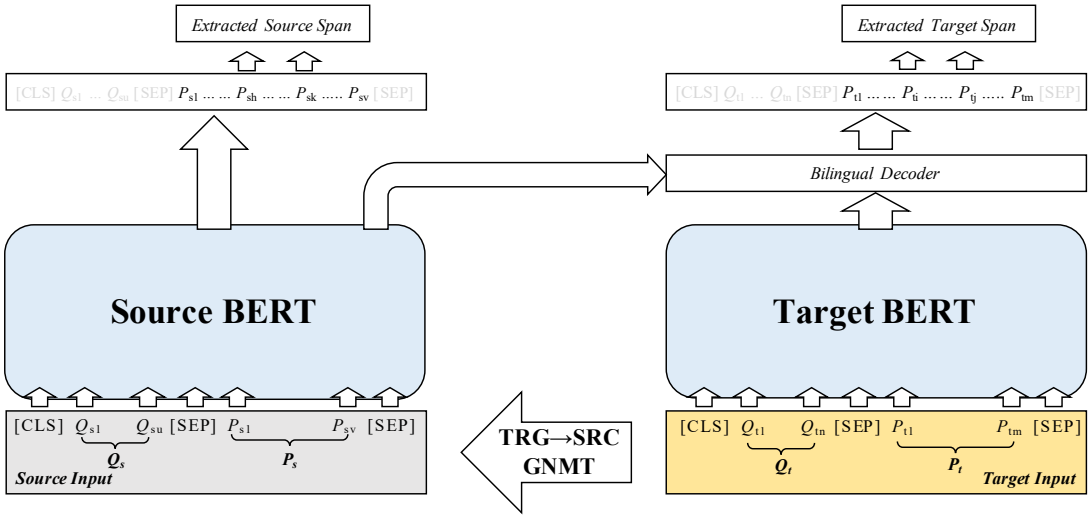


Figure 2: System overview of the Dual BERT model for cross-lingual machine reading comprehension task.

After obtaining attended representation $R'$, we use an additional fully connected layer with residual layer normalization which is similar to BERT implementation.

$$R = W_r R' + b_r, \quad W_r \in \mathbb{R}^{h*h} \tag{6}$$

$$H_T = concat[B_T, \mathbf{LayerNorm}(B_T + R)] \tag{7}$$

Finally, we calculate weighted sum of $H_T$ to get final span prediction $P_T^s, P_T^e$ (superscript $s$ for start, $e$ for end). For example, the start position $P_T^s$ is calculated by the following equation.

$$P_T^s = \mathbf{softmax}(W_T^\top H_T + b), \quad W_T \in \mathbb{R}^{2h} \tag{8}$$

We calculate standard cross entropy loss for the start and end predictions in the target language.

$$\mathcal{L}_T = -\frac{1}{N}\sum_{i=1}^{N}(y_T^s \log(P_T^s) + y_T^e \log(P_T^e)) \tag{9}$$

### 4.3 Auxiliary Output

In order to evaluate how translated sample behaves in the source language system, we also generate span prediction for source language using BERT representation $B_S$ directly without further calculation, resulting in the start and target prediction $P_S^s, P_S^e$ (similar to Equation 8). Moreover, we also calculate cross-entropy loss $\mathcal{L}_{aux}$ for translated sample (similar to Equation 9), where a $\lambda$ parameter is applied to this loss.

Instead of setting $\lambda$ with heuristic value, in this paper, we propose a novel approach to better adjust $\lambda$ automatically. As the sample was generated by the machine translation system, there would be information loss during the translation process. Wrong or partially translated samples may harm the performance of reading comprehension system. To measure how the translated samples assemble the real target samples, we calculate cosine similarity between the *ground truth* span representation in source and target language (denoted as $\tilde{H}_S$ and $\tilde{H}_T$). When the ground truth span representation in the translated sample is similar to the real target samples, the $\lambda$ increase; otherwise, we only use target span loss as $\lambda$ may decrease to zero.

The span representation is the concatenation of three parts: BERT representation of ground truth start $B^s \in \mathbb{R}^h$, ground truth end $B^e \in \mathbb{R}^h$, and self-attended span $B^{att} \in \mathbb{R}^h$, which considers both boundary information (start/end) and mixed representation of the whole ground truth span. We use BERT representation $B$[3] to get a self-attended span representation $B^{att}$ using a simple dot product with average pooling, to get a 2D-tensor.

$$\tilde{H}_S = concat[B_S^s, B_S^e, B_S^{att}] \tag{10}$$

$$\tilde{H}_T = concat[B_T^s, B_T^e, B_T^{att}] \tag{11}$$

$$\lambda = \max\{0, \cos < \tilde{H}_S, \tilde{H}_T >\} \tag{12}$$

The overall loss for Dual BERT is composed by two parts: target span loss $\mathcal{L}_T$ and auxiliary span loss in source language $\mathcal{L}_{aux}$.

$$\mathcal{L} = \mathcal{L}_T + \lambda \mathcal{L}_{aux} \tag{13}$$

[3]We mask out the values that out of span.

## 5 Experiments

### 5.1 Experimental Setups

We evaluate our approaches on two public Chinese span-extraction machine reading comprehension datasets: CMRC 2018 (simplified Chinese) (Cui et al., 2019)[4] and DRCD (traditional Chinese) (Shao et al., 2018)[5]. The statistics of the two datasets are listed in Table 1.

| | Train | Dev | Test | Challenge |
|---|---|---|---|---|
| ***CMRC 2018*** | | | | |
| Question # | 10,321 | 3,219 | 4,895 | 504 |
| Answer # | 1 | 3 | 3 | 3 |
| ***DRCD*** | | | | |
| Question # | 26,936 | 3,524 | 3,493 | - |
| Answer # | 1 | 2 | 2 | - |

Table 1: Statistics of CMRC 2018 and DRCD.

Note that, since the test and challenge sets are preserved by CMRC 2018 official to ensure the integrity of the evaluation process, we submitted our best-performing systems to the organizers to get these scores. The resource in source language was chosen as SQuAD (Rajpurkar et al., 2016) training data. The settings of the proposed approaches are listed below in detail.

- **Tokenization**: Following the official BERT implementation, we use WordPiece tokenizer (Wu et al., 2016) for English and character-level tokenizer for Chinese.

- **BERT**: We use pre-trained English BERT on SQuAD 1.1 (Rajpurkar et al., 2016) for initialization, denoted as SQ-$B_{en}$ (base) and SQ-$L_{en}$ (large) for back-translation approaches. For other conditions, we use multi-lingual BERT as default, denoted as $B_{mul}$ (and SQ-$B_{mul}$ for those were pre-trained on SQuAD).[6]

- **Translation**: We use Google Neural Machine Translation (GNMT) system for translation.[7] We evaluated GNMT system on NIST MT02/03/04/05/06/08 Chinese-English set and achieved an average BLEU score of 43.24, compared to previous best work (43.20) (Cheng et al., 2018), yielding state-of-the-art performance.

- **Optimization**: Following original BERT implementation, we use ADAM with weight decay optimizer (Kingma and Ba, 2014) using an initial learning rate of 4e-5 and use cosine learning rate decay scheme instead of the original linear decay, which we found it beneficial for stabilizing results. The training batch size is set to 64, and each model is trained for 2 epochs, which roughly takes 1 hour.

- **Implementation**: We modified the TensorFlow (Abadi et al., 2016) version `run_squad.py` provided by BERT. All models are trained on Cloud TPU v2 that has 64GB HBM.

### 5.2 Overall Results

The overall results are shown in Table 2. As we can see that, without using any alignment approach, the zero-shot results are quite lower regardless of using English BERT-base (#1) or BERT-large (#2). When we apply SimpleMatch (#3), we observe significant improvements demonstrating its effectiveness. The Answer Aligner (#4) could further improve the performance beyond SimpleMatch approach, demonstrating that the machine learning approach could dynamically adjust the span output by learning the semantic relationship between translated answer and target passage. Also, the Answer Verifier (#5) could further boost performance and surpass the multi-lingual BERT baseline (#7) that only use target training data, demonstrating that it is beneficial to adopt rich-resourced language to improve machine reading comprehension in other languages.

When we do not use SQuAD pre-trained weights, the proposed Dual BERT (#8) yields significant improvements (all results are verified by p-test with $p < 0.05$) over both Chinese BERT (#6) and multi-lingual BERT (#7) by a large margin. If we only train the BERT with SQuAD (#9), which is a zero-shot system, we can see that it achieves decent performance on two Chinese reading comprehension data. Moreover, we can also pursue further improvements by continue training (#10) with Chinese data starting from the system #9, or mixing Chinese data with SQuAD and training from initial multi-lingual BERT (#11). Under powerful SQuAD pre-trained baselines, Dual BERT (#12) still gives moderate and consistent improvements over Cascade Training (#10) and Mixed Training (#11) baselines

[4] https://github.com/ymcui/cmrc2018/
[5] https://github.com/DRCSolutionService/DRCD/
[6] https://github.com/google-research/bert
[7] https://cloud.google.com/translate/

| # | System | CMRC 2018 Dev EM | CMRC 2018 Dev F1 | CMRC 2018 Test EM | CMRC 2018 Test F1 | CMRC 2018 Challenge EM | CMRC 2018 Challenge F1 | DRCD Dev EM | DRCD Dev F1 | DRCD Test EM | DRCD Test F1 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | *Human Performance* | *91.1* | *97.3* | *92.4* | *97.9* | *90.4* | *95.2* | - | - | *80.4* | *93.3* |
| | P-Reader (single model)† | 59.9 | 81.5 | 65.2 | 84.4 | 15.1 | 39.6 | - | - | - | - |
| | Z-Reader (single model)† | 79.8 | 92.7 | 74.2 | 88.1 | 13.9 | 37.4 | - | - | - | - |
| | MCA-Reader (ensemble)† | 66.7 | 85.5 | 71.2 | 88.1 | 15.5 | 37.1 | - | - | - | - |
| | RCEN (ensemble)† | 76.3 | 91.4 | 68.7 | 85.8 | 15.3 | 34.5 | - | - | - | - |
| | r-net (single model)† | - | - | - | - | - | - | - | - | 29.1 | 44.4 |
| | DA (Yang et al., 2019) | 49.2 | 65.4 | - | - | - | - | 55.4 | 67.7 | - | - |
| 1 | GNMT+BERT$_{SQ-B_{en}}$ ♠ | 15.9 | 40.3 | 20.8 | 45.4 | 4.2 | 20.2 | 28.1 | 50.0 | 26.6 | 48.9 |
| 2 | GNMT+BERT$_{SQ-L_{en}}$ ♠ | 16.8 | 42.1 | 21.7 | 47.3 | 5.2 | 22.0 | 28.9 | 52.0 | 28.7 | 52.1 |
| 3 | GNMT+BERT$_{SQ-L_{en}}$+SimpleMatch♠ | 26.7 | 56.9 | 31.3 | 61.6 | 9.1 | 35.5 | 36.9 | 60.6 | 37.0 | 61.2 |
| 4 | GNMT+BERT$_{SQ-L_{en}}$+Aligner | 46.1 | 66.4 | 49.8 | 69.3 | 16.5 | 40.9 | 60.1 | 70.5 | 59.5 | 70.7 |
| 5 | GNMT+BERT$_{SQ-L_{en}}$+Verifier | 64.7 | 84.7 | 68.9 | 86.8 | 20.0 | 45.6 | 83.5 | 90.1 | 82.6 | 89.6 |
| 6 | BERT$_{B_{cn}}$ | 63.6 | 83.9 | 67.8 | 86.0 | 18.4 | 42.1 | 83.4 | 90.1 | 81.9 | 89.0 |
| 7 | BERT$_{B_{mul}}$ | 64.1 | 84.4 | 68.6 | 86.8 | 18.6 | 43.8 | 83.2 | 89.9 | 82.4 | 89.5 |
| 8 | **Dual BERT** | 65.8 | 86.3 | 70.4 | 88.1 | 23.8 | 47.9 | 84.5 | 90.8 | 83.7 | 90.3 |
| 9 | BERT$_{SQ-B_{mul}}$ ♠ | 56.5 | 77.5 | 59.7 | 79.9 | 18.6 | 41.4 | 66.7 | 81.0 | 65.4 | 80.1 |
| 10 | BERT$_{SQ-B_{mul}}$ + Cascade Training | 66.6 | 87.3 | 71.8 | 89.4 | 25.6 | 52.3 | 85.2 | 91.4 | 84.4 | 90.8 |
| 11 | BERT$_{B_{mul}}$ + Mixed Training | 66.8 | 87.5 | 72.6 | 89.8 | 26.7 | 53.4 | 85.3 | 91.6 | 84.7 | 91.2 |
| 12 | **Dual BERT (w/ SQuAD)** | 68.0 | 88.1 | 73.6 | 90.2 | 27.8 | 55.2 | 86.0 | 92.1 | 85.4 | 91.6 |

Table 2: Experimental results on CMRC 2018 and DRCD. † indicates unpublished works (some of the systems are using development set for training, which makes the results not directly comparable.). ♠ indicates zero-shot approach. We mark our system with an ID in the first column for reference simplicity.

and set new state-of-the-art performances on both datasets, demonstrating the effectiveness of using machine-translated sample to enhance the Chinese reading comprehension performance.

### 5.3 Results on Japanese and French SQuAD

In this paper, we propose a simple but effective approach called SimpleMatch to align translated answer to original passage span. While one may argue that using neural machine translation attention to project source answer to original target passage span is ideal as used in Asai et al. (2018). However, to extract attention value in neural machine translation system and apply it to extract the original passage span is bothersome and computationally ineffective. To demonstrate the effectiveness of using SimpleMatch instead of using NMT attention to extract original passage span in zero-shot condition, we applied SimpleMatch to Japanese and French SQuAD (304 samples for each) which is what exactly used in Asai et al. (2018). The results are listed in Table 3.

From the results, we can see that, though our baseline (GNMT+BERT$_{L_{en}}$) is higher than previous work (Back-Translation (Asai et al., 2018)), when using SimpleMatch to extract original passage span could obtain competitive of even larger

| | Japanese EM | Japanese F1 | French EM | French F1 |
|---|---|---|---|---|
| Back-Translation† | 24.8 | 42.6 | 23.5 | 44.0 |
| +Runtime MT† | 37.0 | 52.2 | 40.7 | 61.9 |
| GNMT+BERT$_{L_{en}}$ | 26.9 | 46.2 | 39.1 | 67.0 |
| +SimpleMatch | 37.3 | 58.0 | 47.4 | 71.5 |
| BERT$_{SQ-B_{mul}}$ | 61.3 | 73.4 | 57.6 | 77.1 |

Table 3: Zero-shot cross-lingual machine reading comprehension results on Japanese and French SQuAD data. † are extracted in Asai et al. (2018).

improvements. In Japanese SQuAD, the F1 score improved by 9.6 in Asai et al. (2018) using NMT attention, while we obtain larger improvement with 11.8 points demonstrating the effectiveness of the proposed method. BERT with pre-trained SQuAD weights yields the best performance among these systems, as it does not require the machine translation process and has unified text representations for different languages.

### 5.4 Ablation Studies

In this section, we ablate important components in our model to explicitly demonstrate its effectiveness. The ablation results are depicted in Table 4.

As we can see that, removing SQuAD pre-

| | EM | F1 |
|---|---|---|
| **Dual BERT (w/ SQuAD)** | **68.0** | **88.1** |
| w/o Auxiliary Loss | 67.5 (-0.5) | 87.7 (-0.4) |
| w/o Dynamic Lambda | 67.3 (-0.7) | 87.5 (-0.6) |
| w/o Self-Adaptive Att. | 67.2 (-0.8) | 87.5 (-0.6) |
| w/o Source BERT | 66.6 (-1.4) | 87.3 (-0.8) |
| w/o SQuAD Pre-Train | 65.8 (-2.2) | 86.3 (-1.8) |

Table 4: Ablations of Dual BERT on the CMRC 2018 development set.

trained weights (i.e., using randomly initialized BERT) hurts the performance most, suggesting that it is beneficial to use pre-trained weights though the source and the target language is different. Removing source BERT will degenerate to cascade training, and the results show that it also harms overall performance, demonstrating that it is beneficial to utilize translated sample for better characterizing the relations between <passage, question, answer>. The other modifications seem to also consistently decrease the performance to some extent, but not as salient as the data-related components (last two lines), indicating that data-related approaches are important in cross-lingual machine reading comprehension task.

## 6 Discussion

In our preliminary cross-lingual experiments, we adopt English as our source language data. However, one question remains unclear.

*Is it better to pre-train with larger data in a distant language (such as English, as oppose to Simplified Chinese), or with smaller data in closer language (such as Traditional Chinese)?*

To investigate the problem, we plot the multi-lingual BERT performance on the CMRC 2018 development data using different language and data size in the pre-training stage. The results are depicted in Figure 3, and we come to several observations.

Firstly, when the size of pre-training data is under 25k (training data size of DRCD), we can see that there is no much difference whether we use Chinese or English data for pre-training, and even the English pre-trained models are better than Chinese pre-trained models in most of the times, which is not expected. We suspect that, by using multi-lingual BERT, the model tend to provide universal representations for the text and learn the language-independent semantic relations among the inputs which is ideal for cross-lingual tasks,

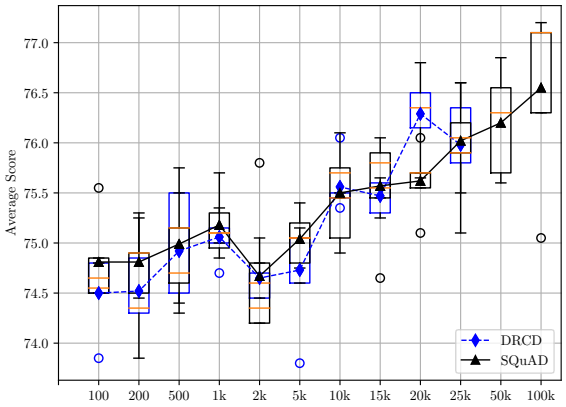


Figure 3: BERT performance (average of EM and F1) with different amount of pre-training SQuAD (English) or DRCD (Traditional Chinese).

thus the model is not that sensitive to the language in the pre-training stage. Also, as training data size of SQuAD is larger than DRCD, we could use more data for pre-training. When we add more SQuAD data (>25k) in the pre-training stage, the performance on the downstream task (CMRC 2018) continues to improve significantly. In this context, we conclude that,

- When the pre-training data is not abundant, there is no special preference on the selection of source (pre-training) language.
- If there are large-scale training data available for several languages, we should select the source language as the one that has the largest training data rather than its linguistic similarity to the target language.

Furthermore, one could also take advantages of data in various languages, but not only in a bilingual environment, to further exploit knowledge from various sources, which is beyond the scope of this paper and we leave this for future work.

## 7 Conclusion

In this paper, we propose Cross-Lingual Machine Reading Comprehension (CLMRC) task. When there is no training data available for the target language, firstly, we provide several zero-shot approaches that were initially trained on English and transfer to other languages, along with three methods to improve the translated answer span by using unsupervised and supervised approaches. When there is training data available for the target language, we propose a novel model

called Dual BERT to simultaneously model the <passage, question, answer> in source and target languages using multi-lingual BERT. The proposed method takes advantage of the large-scale training data by rich-resource language (such as SQuAD) and learns the semantic relations between the passage and question in both source and target language. Experiments on two Chinese machine reading comprehension datasets indicate that the proposed model could give consistent and significant improvements over various state-of-the-art systems by a large margin and set baselines for future research on CLMRC task.

Future studies on cross-lingual machine reading comprehension will focus on 1) how to utilize various types of English reading comprehension data; 2) cross-lingual machine reading comprehension without the translation process, etc.

## Acknowledgments

We would like to thank all anonymous reviewers for their thorough reviewing and providing constructive comments to improve our paper. The first author was partially supported by the Google TensorFlow Research Cloud (TFRC) program for Cloud TPU access. This work was supported by the National Natural Science Foundation of China (NSFC) via grant 61976072, 61632011, and 61772153.

## References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283.

Akari Asai, Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2018. Multilingual extractive reading comprehension by runtime machine translation. *arXiv preprint arXiv:1809.03275*.

Yong Cheng, Zhaopeng Tu, Fandong Meng, Junjie Zhai, and Yang Liu. 2018. Towards robust neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1756–1766. Association for Computational Linguistics.

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. 2017. Attention-over-attention neural networks for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 593–602. Association for Computational Linguistics.

Yiming Cui, Ting Liu, Wanxiang Che, Li Xiao, Zhipeng Chen, Wentao Ma, Shijin Wang, and Guoping Hu. 2019. A span-extraction dataset for chinese machine reading comprehension. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing*. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William Cohen, and Ruslan Salakhutdinov. 2017. Gated-attention readers for text comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1832–1846. Association for Computational Linguistics.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1684–1692.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The goldilocks principle: Reading children's books with explicit memory representations. *arXiv preprint arXiv:1511.02301*.

Minghao Hu, Furu Wei, Yuxing Peng, Zhen Huang, Nan Yang, and Dongsheng Li. 2019. Read + verify: Machine reading comprehension with unanswerable questions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):6529–6537.

Rudolf Kadlec, Martin Schmid, Ondřej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 908–918. Association for Computational Linguistics.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ting Liu, Yiming Cui, Qingyu Yin, Wei-Nan Zhang, Shijin Wang, and Guoping Hu. 2017. Generating and exploiting large-scale pseudo training data for zero pronoun resolution. In *Proceedings of the 55th*

*Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 102–111. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392. Association for Computational Linguistics.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hananneh Hajishirzi. 2016. Bi-directional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Chih Chieh Shao, Trois Liu, Yuting Lai, Yiying Tseng, and Sam Tsai. 2018. Drcd: a chinese machine reading comprehension dataset. *arXiv preprint arXiv:1806.00920*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Shuohang Wang and Jing Jiang. 2016. Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604*.

Wei Yang, Yuqing Xie, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019. Data augmentation for bert fine-tuning in open-domain question answering. *arXiv preprint arXiv:1904.06652*.

# A Multi-Type Multi-Span Network for Reading Comprehension that Requires Discrete Reasoning

**Minghao Hu, Yuxing Peng, Zhen Huang, Dongsheng Li**
National University of Defense Technology, Changsha, China
{huminghao09,pengyuxing,huangzhen,dsli}@nudt.edu.cn

## Abstract

Rapid progress has been made in the field of reading comprehension and question answering, where several systems have achieved human parity in some simplified settings. However, the performance of these models degrades significantly when they are applied to more realistic scenarios, where answers are involved with various types, multiple text strings are correct answers, or discrete reasoning abilities are required. In this paper, we introduce the Multi-Type Multi-Span Network (MTMSN), a neural reading comprehension model that combines a multi-type answer predictor designed to support various answer types (e.g., span, count, negation, and arithmetic expression) with a multi-span extraction method for dynamically producing one or multiple text spans. In addition, an arithmetic expression reranking mechanism is proposed to rank expression candidates for further confirming the prediction. Experiments show that our model achieves 79.9 F1 on the DROP hidden test set, creating new state-of-the-art results. Source code[1] is released to facilitate future work.

## 1 Introduction

This paper considers the reading comprehension task in which some *discrete-reasoning* abilities are needed to correctly answer questions. Specifically, we focus on a new reading comprehension dataset called DROP (Dua et al., 2019), which requires Discrete Reasoning Over the content of Paragraphs to obtain the final answer. Unlike previous benchmarks such as CNN/DM (Hermann et al., 2015) and SQuAD (Rajpurkar et al., 2016) that have been well solved (Chen et al., 2016; Devlin et al., 2019), DROP is substantially more challenging in three ways. First, the answers to the questions involve a wide range of types such as numbers, dates, or text strings. Therefore, various kinds of prediction strategies are required to successfully find the answers. Second, rather than restricting the answer to be a span of text, DROP loosens the constraint so that answers may be a set of multiple text strings. Third, for questions that require discrete reasoning, a system must have a more comprehensive understanding of the context and be able to perform numerical operations such as addition, counting, or sorting.

Existing approaches, when applied to this more realistic scenario, have three problems. First, to produce various answer types, Dua et al. (2019) extend previous one-type answer prediction (Seo et al., 2017) to multi-type prediction that supports span extraction, counting, and addition/subtraction. However, they have not fully considered all potential types. Take the question "*What percent are not non-families?*" and the passage snippet "*39.9% were non-families*" as an example, a *negation* operation is required to infer the answer. Second, previous reading comprehension models (Wang et al., 2017; Yu et al., 2018; Hu et al., 2018) are designed to produce one single span as the answer. But for some questions such as "*Which ancestral groups are smaller than 11%?*", there may exist several spans as correct answers (e.g., "*Italian*", "*English*", and "*Polish*"), which can not be well handled by these works. Third, to support numerical reasoning, prior work (Dua et al., 2019) learns to predict signed numbers for obtaining an arithmetic expression that can be executed by a symbolic system. Nevertheless, the prediction of each signed number is isolated, and the expression's context information has not been considered. As a result, obviously-wrong expressions, such as all predicted signs are either minus or zero, are likely produced.

To address the above issues, we introduce

[1] https://github.com/huminghao16/MTMSN



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1596–1606, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

the Multi-Type Multi-Span Network (MTMSN), a neural reading comprehension model for predicting various types of answers as well as dynamically extracting one or multiple spans. MTMSN utilizes a series of pre-trained Transformer blocks (Devlin et al., 2019) to obtain a deep bidirectional context representation. On top of it, a multi-type answer predictor is proposed to not only support previous prediction strategies such as span, count number, and arithmetic expression, but also add a new type of logical negation. This results in a wider range of coverage of answer types, which turns out to be crucial to performance. Besides, rather than always producing one single span, we present a multi-span extraction method to produce multiple answers. The model first predicts the number of answers, and then extracts non-overlapped spans to the specific amount. In this way, the model can learn to dynamically extract one or multiple spans, thus being beneficial for multi-answer cases. In addition, we propose an arithmetic expression reranking mechanism to rank expression candidates that are decoded by beam search, so that their context information can be considered during reranking to further confirm the prediction.

Our MTMSN model outperforms all existing approaches on the DROP hidden test set by achieving 79.9 F1 score, a 32.9% absolute gain over prior best work at the time of submission. To make a fair comparison, we also construct a baseline that uses the same BERT-based encoder. Again, MTMSN surpasses it by obtaining a 13.2 F1 increase on the development set. We also provide an in-depth ablation study to show the effectiveness of our proposed methods, analyze performance breakdown by different answer types, and give some qualitative examples as well as error analysis.

## 2 Task Description

In the reading comprehension task that requires discrete reasoning, a passage and a question are given. The goal is to predict an answer to the question by reading and understanding the passage. Unlike previous dataset such as SQuAD (Rajpurkar et al., 2016) where the answer is limited to be a single span of text, DROP loosens the constraint so that the answer involves various types such as number, date, or span of text (Figure 1). Moreover, the answer can be multiple text strings instead of single continuous span ($\mathbf{A}_2$). To successfully find the answer, some discrete reasoning abilities, such as sorting ($\mathbf{A}_1$), subtraction ($\mathbf{A}_3$), and negation ($\mathbf{A}_4$), are required.

**Passage**: As of the census of 2000, there were 218,590 people, 79,667 households, ... 22.5% were of German people, 13.1% Irish people, 9.8% Italian people, ...
$\mathbf{Q}_1$: Which group from the census is larger: German or Irish?
$\mathbf{A}_1$: German
$\mathbf{Q}_2$: Which ancestral groups are at least 10%?
$\mathbf{A}_2$: German, Irish
$\mathbf{Q}_3$: How many more people are there than households?
$\mathbf{A}_3$: 138,923
$\mathbf{Q}_4$: How many percent were not German?
$\mathbf{A}_4$: 77.5

Figure 1: Question-answer pairs along with a passage from the DROP dataset.

## 3 Our Approach

Figure 2 gives an overview of our model that aims to combine neural reading comprehension with numerical reasoning. Our model uses BERT (Devlin et al., 2019) as encoder: we map word embeddings into contextualized representations using pre-trained Transformer blocks (Vaswani et al., 2017) (§3.1). Based on the representations, we employ a multi-type answer predictor that is able to produce four answer types: (1) span from the text; (2) arithmetic expression; (3) count number; (4) negation on numbers (§3.2). Following Dua et al. (2019), we first predict the answer type of a given passage-question pair, and then adopt individual prediction strategies. To support multi-span extraction (§3.3), the model explicitly predicts the number of answer spans. It then outputs non-overlapped spans until the specific amount is reached. Moreover, we do not directly use the arithmetic expression that possesses the maximum probability, but instead re-rank several expression candidates that are decoded by beam search to further confirm the prediction (§3.4). Finally, the model is trained under weakly-supervised signals to maximize the marginal likelihood over all possible annotations (§3.5).

### 3.1 BERT-Based Encoder

To obtain a universal representation for both the question and the passage, we utilize BERT (Devlin et al., 2019), a pre-trained deep bidirectional Transformer model that achieves state-of-the-art performance across various tasks, as the encoder.

Specifically, we first tokenize the question and

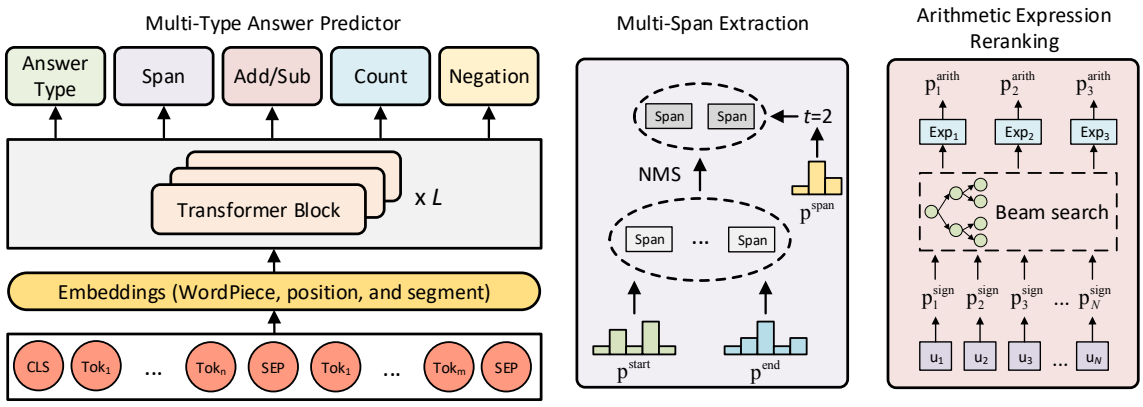


Figure 2: An illustration of MTMSN architecture. The multi-type answer predictor supports four kinds of answer types including span, addition/subtraction, count, and negation. A multi-span extraction method is proposed to dynamically produce one or several spans. The arithmetic expression reranking mechanism aims to rank expression candidates that are decoded by beam search for further validating the prediction.

the passage using the WordPiece vocabulary (Wu et al., 2016), and then generate the input sequence by concatenating a `[CLS]` token, the tokenized question, a `[SEP]` token, the tokenized passage, and a final `[SEP]` token. For each token in the sequence, its input representation is the element-wise addition of WordPiece embeddings, positional embeddings, and segment embeddings (Devlin et al., 2019). As a result, a list of input embeddings $\mathbf{H}_0 \in \mathbb{R}^{T \times D}$ can be obtained, where $D$ is the hidden size and $T$ is the sequence length. A series of $L$ pre-trained Transformer blocks are then used to project the input embeddings into contextualized representations $\mathbf{H}_i$ as:

$$\mathbf{H}_i = \text{TransformerBlock}(\mathbf{H}_{i-1}), \forall i \in [1, L]$$

Here, we omit a detailed introduction of the block architecture and refer readers to Vaswani et al. (2017) for more details.

### 3.2 Multi-Type Answer Predictor

Rather than restricting the answer to always be a span of text, the discrete-reasoning reading comprehension task involves different answer types (e.g., number, date, span of text). Following Dua et al. (2019), we design a multi-type answer predictor to selectively produce different kinds of answers such as span, count number, and arithmetic expression. To further increase answer coverage, we propose adding a new answer type to support logical negation. Moreover, unlike prior work that separately predicts passage spans and question spans, our approach directly extracts spans from the input sequence.

**Answer type prediction** Inspired by the Augmented QANet model (Dua et al., 2019), we use the contextualized token representations from the last four blocks ($\mathbf{H}_{L-3}$, ..., $\mathbf{H}_L$) as the inputs to our answer predictor, which are denoted as $\mathbf{M}_0$, $\mathbf{M}_1$, $\mathbf{M}_2$, $\mathbf{M}_3$, respectively. To predict the answer type, we first split the representation $\mathbf{M}_2$ into a question representation $\mathbf{Q}_2$ and a passage representation $\mathbf{P}_2$ according to the index of intermediate `[SEP]` token. Then the model computes two vectors $\mathbf{h}^{\mathbf{Q}_2}$ and $\mathbf{h}^{\mathbf{P}_2}$ that summarize the question and passage information respectively:

$$\boldsymbol{\alpha}^Q = \text{softmax}(\mathbf{W}^Q \mathbf{Q}_2), \quad \mathbf{h}^{\mathbf{Q}_2} = \boldsymbol{\alpha}^Q \mathbf{Q}_2$$

where $\mathbf{h}^{\mathbf{P}_2}$ is computed in a similar way over $\mathbf{P}_2$.

Next, we calculate a probability distribution to represent the choices of different answer types as:

$$\mathbf{p}^{\text{type}} = \text{softmax}(\text{FFN}([\mathbf{h}^{\mathbf{Q}_2}; \mathbf{h}^{\mathbf{P}_2}; \mathbf{h}^{\text{CLS}}]))$$

Here, $\mathbf{h}^{\text{CLS}}$ is the first vector in the final contextualized representation $\mathbf{M}_3$, and FFN denotes a feed-forward network consisting of two linear projections with a GeLU activation (Hendrycks and Gimpel, 2016) followed by a layer normalization (Lei Ba et al., 2016) in between.

**Span** To extract the answer either from the passage or from the question, we combine the gating mechanism of Wang et al. (2017) with the standard decoding strategy of Seo et al. (2017) to predict the starting and ending positions across the entire sequence. Specifically, we first compute three vectors, namely $\mathbf{g}^{\mathbf{Q}_0}$, $\mathbf{g}^{\mathbf{Q}_1}$, $\mathbf{g}^{\mathbf{Q}_2}$, that summarize

the question information among different levels of question representations:

$$\boldsymbol{\beta}^{Q} = \text{softmax}(\text{FFN}(\mathbf{Q}_2), \quad \mathbf{g}^{\mathbf{Q}_2} = \boldsymbol{\beta}^{Q}\mathbf{Q}_2$$

where $\mathbf{g}^{\mathbf{Q}_0}$ and $\mathbf{g}^{\mathbf{Q}_1}$ are computed over $\mathbf{Q}_0$ and $\mathbf{Q}_1$ respectively, in a similar way as described above.

Then we compute the probabilities of the starting and ending indices of the answer span from the input sequence as:

$$\begin{aligned}
\bar{\mathbf{M}}^{\text{start}} &= [\mathbf{M}_2; \mathbf{M}_0; \mathbf{g}^{\mathbf{Q}_2} \otimes \mathbf{M}_2; \mathbf{g}^{\mathbf{Q}_0} \otimes \mathbf{M}_0], \\
\bar{\mathbf{M}}^{\text{end}} &= [\mathbf{M}_2; \mathbf{M}_1; \mathbf{g}^{\mathbf{Q}_2} \otimes \mathbf{M}_2; \mathbf{g}^{\mathbf{Q}_1} \otimes \mathbf{M}_1], \\
\mathbf{p}^{\text{start}} &= \text{softmax}(\mathbf{W}^{S}\bar{\mathbf{M}}^{\text{start}}), \\
\mathbf{p}^{\text{end}} &= \text{softmax}(\mathbf{W}^{E}\bar{\mathbf{M}}^{\text{end}})
\end{aligned}$$

where $\otimes$ denotes the outer product between the vector $\mathbf{g}$ and each token representation in $\mathbf{M}$.

**Arithmetic expression** In order to model the process of performing addition or subtraction among multiple numbers mentioned in the passage, we assign a three-way categorical variable (plus, minus, or zero) for each number to indicate its sign, similar to Dua et al. (2019). As a result, an arithmetic expression that has a number as the final answer can be obtained and easily evaluated.

Specifically, for each number mentioned in the passage, we gather its corresponding representation from the concatenation of $\mathbf{M}_2$ and $\mathbf{M}_3$, eventually yielding $\mathbf{U} = (\mathbf{u}_1, ..., \mathbf{u}_N) \in \mathbb{R}^{N \times 2*D}$ where $N$ numbers exist. Then the probabilities of the $i$-th number being assigned a plus, minus or zero is computed as:

$$\mathbf{p}_i^{\text{sign}} = \text{softmax}(\text{FFN}([\mathbf{u}_i; \mathbf{h}^{\mathbf{Q}_2}; \mathbf{h}^{\mathbf{P}_2}; \mathbf{h}^{\text{CLS}}]))$$

**Count** We consider the ability of counting entities and model it as a multi-class classification problem. To achieve this, the model first produces a vector $\mathbf{h}^{\mathbf{U}}$ that summarizes the important information among all mentioned numbers, and then computes a counting probability distribution as:

$$\begin{aligned}
\boldsymbol{\alpha}^{U} &= \text{softmax}(\mathbf{W}^{U}\mathbf{U}), \quad \mathbf{h}^{\mathbf{U}} = \boldsymbol{\alpha}^{U}\mathbf{U}, \\
\mathbf{p}^{\text{count}} &= \text{softmax}(\text{FFN}([\mathbf{h}^{\mathbf{U}}; \mathbf{h}^{\mathbf{Q}_2}; \mathbf{h}^{\mathbf{P}_2}; \mathbf{h}^{\text{CLS}}]))
\end{aligned}$$

**Negation** One obvious but important linguistic phenomenon that prior work fails to capture is *negation*. We find there are many cases in DROP that require to perform logical negation on numbers. The question ($\mathbf{Q}_4$) in Figure 1 gives a qualitative example of this phenomenon. To model this phenomenon, we assign a two-way categorical variable for each number to indicate whether a negation operation should be performed. Then we compute the probabilities of logical negation on the $i$-th number as:

$$\mathbf{p}_i^{\text{negation}} = \text{softmax}(\text{FFN}([\mathbf{u}_i; \mathbf{h}^{\mathbf{Q}_2}; \mathbf{h}^{\mathbf{P}_2}; \mathbf{h}^{\text{CLS}}]))$$

### 3.3 Multi-Span Extraction

Although existing reading comprehension tasks focus exclusively on finding one span of text as the final answer, DROP loosens the restriction so that the answer to the question may be several text spans. Therefore, specific adaption should be made to extend previous single-span extraction to multi-span scenario.

To do this, we propose directly predicting the number of spans and model it as a classification problem. This is achieved by computing a probability distribution on span amount as

$$\mathbf{p}^{\text{span}} = \text{softmax}(\text{FFN}([\mathbf{h}^{\mathbf{Q}_2}; \mathbf{h}^{\mathbf{P}_2}; \mathbf{h}^{\text{CLS}}]))$$

To extract non-overlapped spans to the specific amount, we adopt the non-maximum suppression (NMS) algorithm (Rosenfeld and Thurston, 1971) that is widely used in computer vision for pruning redundant bounding boxes, as shown in Algorithm 1. Concretely, the model first proposes a set of top-$K$ spans $\mathbf{S}$ according to the descending order of the span score, which is computed as $\mathbf{p}_k^{\text{start}}\mathbf{p}_l^{\text{end}}$ for the span $(k, l)$. It also predicts the amount of extracted spans $t$ from $\mathbf{p}^{\text{span}}$, and initializes a new set $\tilde{\mathbf{S}}$. Next, we add the span $\mathbf{s}_i$ that possesses the maximum span score to the set $\tilde{\mathbf{S}}$, and remove it from $\mathbf{S}$. We also delete any remaining span $\mathbf{s}_j$ that overlaps with $\mathbf{s}_i$, where the degree of overlap is measured using the text-level F1 function. This process is repeated for remaining spans in $\mathbf{S}$, until $\mathbf{S}$ is empty or the size of $\tilde{\mathbf{S}}$ reaches $t$.

### 3.4 Arithmetic Expression Reranking

As discussed in §3.2, we model the phenomenon of discrete reasoning on numbers by learning to predict a plus, minus, or zero for each number in the passage. In this way, an arithmetic expression composed of signed numbers can be obtained, where the final answer can be deduced by performing simple arithmetic computation. However, since the sign of each number is only determined by the number representation and some coarse-grained global representations, the context information of the expression itself has not been considered. As a result, the model may predict some

**Algorithm 1** Multi-span extraction

**Input:** $\mathbf{p}^{\text{start}}$; $\mathbf{p}^{\text{end}}$; $\mathbf{p}^{\text{span}}$

Generate the set $\mathbf{S}$ by extracting top-$K$ spans
Sort $\mathbf{S}$ in descending order of span scores
$t = \arg\max \mathbf{p}^{\text{span}} + 1$
Initialize $\tilde{\mathbf{S}} = \{\}$
**while** $\mathbf{S} \neq \{\}$ and $|\tilde{\mathbf{S}}| < t$ **do**
    **for** $\mathbf{s}_i$ in $\mathbf{S}$ **do**
        Add span $\mathbf{s}_i$ to $\tilde{\mathbf{S}}$
        Remove span $\mathbf{s}_i$ from $\mathbf{S}$
        **for** $\mathbf{s}_j$ in $\mathbf{S}$ **do**
            **if** $\text{f1}(\mathbf{s}_i, \mathbf{s}_j) > 0$ **then**
                Remove span $\mathbf{s}_j$ from $\mathbf{S}$
**return** $\tilde{\mathbf{S}}$

obviously wrong expressions (e.g., the signs that have maximum probabilities are either minus or zero, resulting in a large negative value). Therefore, in order to further validate the prediction, it is necessary to rank several highly confident expression candidates using the representation summarized from the expression's context.

Specifically, we use beam search to produce top-ranked arithmetic expressions, which are sent back to the network for reranking. Since each expression consists of several signed numbers, we construct an expression representation by taking both the numbers and the signs into account. For each number in the expression, we gather its corresponding vector from the representation $\mathbf{U}$. As for the signs, we initialize an embedding matrix $\mathbf{E} \in \mathbb{R}^{3\times 2*D}$, and find the sign embeddings for each signed number. In this way, given the $i$-th expression that contains $M$ signed numbers at most, we can obtain number vectors $\mathbf{V}_i \in \mathbb{R}^{M\times 2*D}$ as well as sign embeddings $\mathbf{C}_i \in \mathbb{R}^{M\times 2*D}$. Then the expression representation along with the reranking probability can be calculated as:

$$\boldsymbol{\alpha}_i^V = \text{softmax}(\mathbf{W}^V(\mathbf{V}_i + \mathbf{C}_i)),$$
$$\mathbf{h}_i^{\mathbf{V}} = \boldsymbol{\alpha}_i^V(\mathbf{V}_i + \mathbf{C}_i),$$
$$\mathbf{p}_i^{\text{arith}} = \text{softmax}(\text{FFN}([\mathbf{h}_i^{\mathbf{V}}; \mathbf{h}^{\mathbf{Q}_2}; \mathbf{h}^{\mathbf{P}_2}; \mathbf{h}^{\text{CLS}}]))$$

### 3.5 Training and Inference

Since DROP does not indicate the answer type but only provides the answer string, we therefore adopt the weakly supervised annotation scheme, as suggested in Berant et al. (2013); Dua et al. (2019). We find all possible annotations that point to the gold answer, including matching spans, arithmetic expressions, correct count numbers, negation operations, and the number of spans. We use simple rules to search over all mentioned numbers to find potential negations. That is, if 100 minus a number is equal to the answer, then a negation occurs on this number. Besides, we only search the addition/subtraction of three numbers at most due to the exponential search space.

To train our model, we propose using a two-step training method composed of an inference step and a training step. In the first step, we use the model to predict the probabilities of sign assignments for numbers. If there exists any annotation of arithmetic expressions, we run beam search to produce expression candidates and label them as either correct or wrong, which are later used for supervising the reranking component. In the second step, we adopt the marginal likelihood objective function (Clark and Gardner, 2018), which sums over the probabilities of all possible annotations including the above labeled expressions. Notice that there are two objective functions for the multi-span component: one is a distantly-supervised loss that maximizes the probabilities of all matching spans, and the other is a classification loss that maximizes the probability on span amount.

At test time, the model first chooses the answer type and then performs specific prediction strategies. For the span type, we use Algorithm 1 for decoding. If the type is addition/subtraction, arithmetic expression candidates will be proposed and further reranked. The expression with the maximum product of cumulative sign probability and reranking probability is chosen. As for the counting type, we choose the number that has the maximum counting probability. Finally, if the type is negation, we find the number that possesses the largest negation probability, and then output the answer as 100 minus this number.

## 4 Experiments

### 4.1 Implementation Details

**Dataset** We consider the reading comprehension benchmark that requires Discrete Reasoning Over Paragraphs (DROP) (Dua et al., 2019) to train and evaluate our model. DROP contains crowdsourced, adversarially-created, 96.6K question-answer pairs, with 77.4K for training, 9.5K for validation, and another 9.6K hidden examples for testing. Passages are extracted from Wikipedia articles and the answer to each question involves various types such as number, date, or text string. Some answers may even be a set of multiple spans of text in the passage. To find the answers, a com-

| Model | Dev | | Test | |
|---|---|---|---|---|
| | EM | F1 | EM | F1 |
| Heuristic Baseline (Dua et al., 2019) | 4.28 | 8.07 | 4.18 | 8.59 |
| Semantic Role Labeling (Carreras and Màrquez, 2004) | 11.03 | 13.67 | 10.87 | 13.35 |
| BiDAF (Seo et al., 2017) | 26.06 | 28.85 | 24.75 | 27.49 |
| QANet+ELMo (Yu et al., 2018) | 27.71 | 30.33 | 27.08 | 29.67 |
| $\text{BERT}_{\text{BASE}}$ (Devlin et al., 2019) | 30.10 | 33.36 | 29.45 | 32.70 |
| NAQANet (Dua et al., 2019) | 46.20 | 49.24 | 44.07 | 47.01 |
| $\text{NABERT}_{\text{BASE}}$ | 55.82 | 58.75 | - | - |
| $\text{NABERT}_{\text{LARGE}}$ | 64.61 | 67.35 | - | - |
| $\text{MTMSN}_{\text{BASE}}$ | 68.17 | 72.81 | - | - |
| $\text{MTMSN}_{\text{LARGE}}$ | **76.68** | **80.54** | **75.85** | **79.88** |
| Human Performance (Dua et al., 2019) | - | - | 92.38 | 95.98 |

Table 1: The performance of MTMSN and other competing approaches on DROP dev and test set.

| Model | BASE | | LARGE | |
|---|---|---|---|---|
| | EM | F1 | EM | F1 |
| MTMSN | 68.2 | 72.8 | 76.7 | 80.5 |
| w/o Add/Sub | 46.7 | 51.3 | 53.8 | 58.0 |
| w/o Count | 62.5 | 66.4 | 71.8 | 75.6 |
| w/o Negation | 59.4 | 63.6 | 67.2 | 70.9 |
| w/o Multi-Span | 67.5 | 70.7 | 75.6 | 78.4 |
| w/o Reranking | 66.9 | 71.2 | 74.9 | 78.7 |

Table 2: Ablation tests of base and large models on the DROP dev set.

| Model | EM | F1 |
|---|---|---|
| MTMSN | 76.7 | 80.5 |
| w/o Q/P Vectors | 75.1 | 79.2 |
| w/o CLS Vector | 74.0 | 78.4 |
| Q/P Vectors Using Last Hidden | 76.5 | 80.2 |
| w/o Gated Span Prediction | 75.8 | 79.7 |
| Combine Add/Sub with Negation | 75.5 | 79.4 |

Table 3: Ablation tests of different architecture choices using $\text{MTMSN}_{\text{LARGE}}$.

prehensive understanding of the context as well as the ability of numerical reasoning are required.

**Model settings** We build our model upon two publicly available uncased versions of BERT: $\text{BERT}_{\text{BASE}}$ and $\text{BERT}_{\text{LARGE}}$[2], and refer readers to Devlin et al. (2019) for details on model sizes. We use Adam optimizer with a learning rate of 3e-5 and warmup over the first 5% steps to train. The maximum number of epochs is set to 10 for base models and 5 for large models, while the batch size is 12 or 24 respectively. A dropout probability of 0.1 is used unless stated otherwise. The number of counting class is set to 10, and the maximum number of spans is 8. The beam size is 3 by default, while the maximum amount of signed numbers $M$ is set to 4. All texts are tokenized using WordPiece vocabulary (Wu et al., 2016), and truncated to sequences no longer than 512 tokens.

**Baselines** Following the implementation of Augmented QANet (NAQANet) (Dua et al., 2019), we introduce a similar baseline called Augmented BERT (NABERT). The main difference is that we replace the encoder of QANet (Yu et al., 2018) with the pre-trained Transformer blocks (Devlin et al., 2019). Moreover, it also supports the prediction of various answer types such as span, arithmetic expression, and count number.

### 4.2 Main Results

Two metrics, namely Exact Match (EM) and F1 score, are utilized to evaluate models. We use the official script to compute these scores. Since the test set is hidden, we only submit the best single model to obtain test results.

Table 1 shows the performance of our model and other competitive approaches on the develop-

[2] $\text{BERT}_{\text{BASE}}$ is the original version while $\text{BERT}_{\text{LARGE}}$ is the model augmented with n-gram masking and synthetic self-training: https://github.com/google-research/bert.

| Type | (%) | NABERT | | MTMSN | |
|---|---|---|---|---|---|
| | | EM | F1 | EM | F1 |
| Date | 1.6 | 55.7 | 60.8 | 55.7 | 69.0 |
| Number | 61.9 | 63.8 | 64.0 | 80.9 | 81.1 |
| Single Span | 31.7 | 75.9 | 80.6 | 77.5 | 82.8 |
| Multi Span | 4.8 | 0 | 22.7 | 25.1 | 62.8 |

Table 4: Performance breakdown of $\text{NABERT}_{\text{LARGE}}$ and $\text{MTMSN}_{\text{LARGE}}$ by gold answer types.

| Type | NABERT | | | MTMSN | | |
|---|---|---|---|---|---|---|
| | (%) | EM | F1 | (%) | EM | F1 |
| Span | 43.0 | 67.9 | 74.2 | 42.7 | 72.2 | 81.0 |
| Add/Sub | 43.6 | 62.0 | 62.1 | 32.4 | 78.1 | 78.2 |
| Count | 13.4 | 62.4 | 62.4 | 13.4 | 70.4 | 70.4 |
| Negation | 0 | 0 | 0 | 11.5 | 96.3 | 96.3 |

Table 5: Performance breakdown of $\text{NABERT}_{\text{LARGE}}$ and $\text{MTMSN}_{\text{LARGE}}$ by predicted answer types.

ment and test sets. MTMSN outperforms all existing approaches by a large margin, and creates new state-of-the-art results by achieving an EM score of 75.85 and a F1 score of 79.88 on the test set. Since our best model utilizes $\text{BERT}_{\text{LARGE}}$ as encoder, we therefore compare $\text{MTMSN}_{\text{LARGE}}$ with the $\text{NABERT}_{\text{LARGE}}$ baseline. As we can see, our model obtains 12.07/13.19 absolute gain of EM/F1 over the baseline, demonstrating the effectiveness of our approach. However, as the human achieves 95.98 F1 on the test set, our results suggest that there is still room for improvement.

### 4.3 Ablation Study

**Component ablation** To analyze the effect of the proposed components, we conduct ablation studies on the development set. As illustrated in Table 2, the use of addition and subtraction is extremely crucial: the EM/F1 performance of both the base and large models drop drastically by more than 20 points if it is removed. Predicting count numbers is also an important component that contributes nearly 5% gain on both metrics. Moreover, enhancing the model with the negation type significantly increases the F1 by roughly 9 percent on both models. In brief, the above results show that multi-type answer prediction is vitally important for handling different forms of answers, especially in cases where discrete reasoning abilities are required.

We also report the performance after removing the multi-span extraction method. The results reveal that it has a more negative impact on the F1 score. We interpret this phenomenon as follows: producing multiple spans that are partially matched with ground-truth answers is much easier than generating an exactly-matched set of multiple answers. Hence for multi-span scenarios, the gain of our method on F1 is relatively easier to obtain than the one on EM. Finally, to ablate arithmetic expression reranking, we simply use the arithmetic expression that has the maximum cumulative sign probability instead. We find that our reranking mechanism gives 1.8% gain on both metrics for the large model. This confirms that validating expression candidates with their context information is beneficial for filtering out highly-confident but wrong predictions.

**Architecture ablation** We further conduct a detailed ablation in Table 3 to evaluate our architecture designs. First, we investigate the effects of some "global vectors" used in our model. Specifically, we find that removing the question and passage vectors from all involved computation leads to 1.3 % drop on F1. Ablating the representation of `[CLS]` token leads to even worse results. We also try to use the last hidden representation (denoted as $\mathbf{M}_3$) to calculate question and passage vectors, but find that does not work. Next, we remove the gating mechanism used during span prediction, and observe a nearly 0.8% decline on both metrics. Finally, we share parameters between the arithmetic expression component and the negation component, and find the performance drops by 1.1% on F1.

### 4.4 Analysis and Discussion

**Performance breakdown** We now provide a quantitative analysis by showing performance breakdown on the development set. Table 4 shows that our gains mainly come from the most frequent number type, which requires various types of symbolic, discrete reasoning operations. Moreover, significant improvements are also obtained in the multi-span category, where the F1 score increases by more than 40 points. This result further proves the validity of our multi-span extraction method.

We also give the performance statistics that are categorized according to the predicted answer types in Table 5. As shown in the Table, the main improvements are due to the addition/subtraction and negation types. We conjecture that there are two reasons for these improvements. First, our

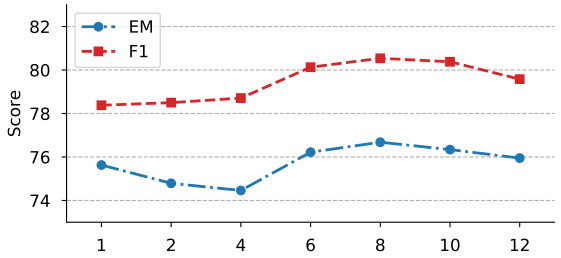


Figure 3: EM/F1 scores of $\text{MTMSN}_{\text{LARGE}}$ with different maximum numbers of spans.

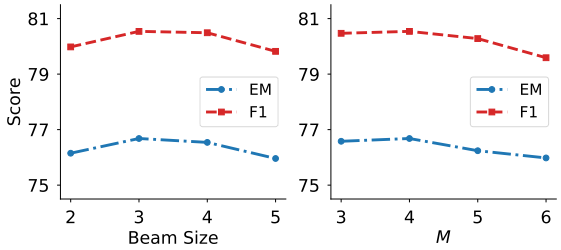


Figure 4: EM/F1 scores of $\text{MTMSN}_{\text{LARGE}}$ with different beam sizes and amounts of signed numbers ($M$).

proposed expression reranking mechanism helps validate candidate expressions. Second, a new inductive bias that enables the model to perform logical negation has been introduced. The impressive performance on the negation type confirms our judgement, and suggests that the model is able to find most of negation operations. In addition, we also observe promising gains brought by the span and count types. We think the gains are mainly due to the multi-span extraction method as well as architecture designs.

**Effect of maximum number of spans** To investigate the effect of maximum number of spans on multi-span extraction, we conduct an experiment on the dev set and show the curves in Figure 3. We vary the value from 2 to 12, increased by 2, and also include the extreme value 1. According to the Figure, the best results are obtained at 8. A higher value could potentially increase the answer recall but damage the precision by making more predictions, and a smaller value may force the model to produce limited number of answers, resulting in high precision but low recall. Therefore, a value of 8 turns out to be a good trade-off between recall and precision. Moreover, when the value decreases to 1, the multi-span extraction degrades to previous single-span scenario, and the performance drops significantly.

| Configuration | Skipped | Kept | Ratio (%) | F1 |
|---|---|---|---|---|
| Span | 33752 | 43657 | 56.4 | 38.9 |
| + ♣ | 6384 | 71025 | 91.7 | 59.2 |
| + ♣ + ♠ | 4282 | 73127 | 94.4 | 63.6 |
| + ♣ + ♠ + ♡ | 1595 | 75814 | 97.9 | 72.8 |

Table 6: Annotation statistics under different combinations of answer types in the DROP train set. "Kept" and "Skipped" mean the number of examples with or without annotation, respectively. ♣ refers to Add/Sub, ♠ denotes Count, and ♡ indicates Negation. F1 scores are benchmarked using $\text{MTMSN}_{\text{BASE}}$ on the dev set.

**Effect of beam size and $M$** We further investigate the effect of beam size and maximum amount of signed numbers in Figure 4. As we can see, a beam size of 3 leads to the best performance, likely because a larger beam size might confuse the model as too many candidates are ranked, on the other hand, a small size could be not sufficient to cover the correct expression. In addition, we find that the performance constantly decreases as the maximum threshold $M$ increases, suggesting that most of expressions only contain two or three signed numbers, and setting a larger threshold could bring in additional distractions.

**Annotation statistics** We list the annotation statistics on the DROP train set in Table 6. As we can see, only annotating matching spans results in a labeled ratio of 56.4%, indicating that DROP includes various answer types beyond text spans. By further considering the arithmetic expression, the ratio increase sharply to 91.7%, suggesting more than 35% answers need to be inferred with numeral reasoning. Continuing adding counting leads to a percentage of 94.4%, and a final 97.9% coverage is achieved by additionally taking negation into account. More importantly, the F1 score constantly increases as more answer types are considered. This result is consistent with our observations in ablation study.

**Error analysis** Finally, to better understand the remaining challenges, we randomly sample 100 incorrectly predicted examples based on EM and categorize them into 7 classes. 38% of errors are incorrect arithmetic computations, 18% require sorting over multiple entities, 13% are due to mistakes on multi-span extraction, 10% are single-span extraction problems, 8% involve miscounting, another 8% are wrong predictions on span number, the rest (5%) are due to various reasons

such as incorrect preprocessing, negation error, and so on. See Appendix for some examples of the above error cases.

## 5 Related Work

**Reading comprehension benchmarks** Promising advancements have been made for reading comprehension due to the creation of many large datasets. While early research used cloze-style tests (Hermann et al., 2015; Hill et al., 2016), most of recent works (Rajpurkar et al., 2016; Joshi et al., 2017) are designed to extract answers from the passage. Despite their success, these datasets only require shallow pattern matching and simple logical reasoning, thus being well solved (Chen et al., 2016; Devlin et al., 2019). Recently, Dua et al. (2019) released a new benchmark named DROP that demands discrete reasoning as well as deeper paragraph understanding to find the answers. Saxton et al. (2019) introduced a dataset consisting of different types of mathematics problems to focuses on mathematical computation. We choose to work on DROP to test both the numerical reasoning and linguistic comprehension abilities.

**Neural reading models** Previous neural reading models, such as BiDAF (Seo et al., 2017), R-Net (Wang et al., 2017), QANet (Yu et al., 2018), Reinforced Mreader (Hu et al., 2018), are usually designed to extract a continuous span of text as the answer. Dua et al. (2019) enhanced prior single-type prediction to support various answer types such as span, count number, and addition/subtraction. Different from these approaches, our model additionally supports a new negation type to increase answer coverage, and learns to dynamically extract one or multiple spans. Morevoer, answer reranking has been well studied in several prior works (Cui et al., 2016; Wang et al., 2018a,b,c; Hu et al., 2019). We follow this line of work, but propose ranking arithmetic expressions instead of candidate answers.

**End-to-end symbolic reasoning** Combining neural methods with symbolic reasoning was considered by Graves et al. (2014); Sukhbaatar et al. (2015), where neural networks augmented with external memory are trained to execute simple programs. Later works on program induction (Reed and De Freitas, 2016; Neelakantan et al., 2016; Liang et al., 2017) extended this idea by using several built-in logic operations along with a key-value memory to learn different types of compositional programs such as addition or sorting. In contrast to these works, MTMSN does not model various types of reasoning with a universal memory mechanism but instead deals each type with individual predicting strategies.

**Visual question answering** In computer vision community, the most similar work to our approach is Neural Module Networks (Andreas et al., 2016b), where a dependency parser is used to lay out a neural network composed of several pre-defined modules. Later, Andreas et al. (2016a) proposed dynamically choosing an optimal layout structure from a list of layout candidates that are produced by off-the-shelf parsers. Hu et al. (2017) introduced an end-to-end module network that learns to predict instance-specific network layouts without the aid of a parser. Compared to these approaches, MTMSN has a static network layout that can not be changed during training and evaluation, where pre-defined "modules" are used to handle different types of answers.

## 6 Conclusion

We introduce MTMSN, a multi-type multi-span network for reading comprehension that requires discrete reasoning over the content of paragraphs. We enhance a multi-type answer predictor to support logical negation, propose a multi-span extraction method for producing multiple answers, and design an arithmetic expression reranking mechanism to further confirm the prediction. Our model achieves 79.9 F1 on the DROP hidden test set, creating new state-of-the-art results. As future work, we would like to consider handling additional types such as sorting or multiplication/division. We also plan to explore more advanced methods for performing complex numerical reasoning.

## Acknowledgments

We would like to thank the anonymous reviewers for their thoughtful comments and insightful feedback. This work was supported by the National Key Research and Development Program of China (2016YFB100101).

## References

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016a. Learning to compose neural net-

works for question answering. In *Proceedings of NAACL*.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016b. Neural module networks. In *Proceedings of CVPR*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of EMNLP*.

Xavier Carreras and Lluís Màrquez. 2004. Introduction to the conll-2004 shared task: Semantic role labeling. In *Proceedings of CONLL*.

Danqi Chen, Jason Bolton, and Christopher D Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. *arXiv preprint arXiv:1606.02858*.

Christopher Clark and Matt Gardner. 2018. Simple and effective multi-paragraph reading comprehension. In *Proceedings of ACL*.

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. 2016. Attention-over-attention neural networks for reading comprehension. *arXiv preprint arXiv:1607.04423*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL*.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of NAACL*.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Dan Hendrycks and Kevin Gimpel. 2016. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR, abs/1606.08415*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proceedings of NIPS*.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The goldilocks principle: Reading childrens books with explicit memory representations. In *Proceedings of ICLR*.

Minghao Hu, Yuxing Peng, Zhen Huang, and Dongsheng Li. 2019. Retrieve, read, rerank: Towards end-to-end multi-document reading comprehension. In *Proceedings of ACL*.

Minghao Hu, Yuxing Peng, Zhen Huang, Xipeng Qiu, Furu Wei, and Ming Zhou. 2018. Reinforced mnemonic reader for machine reading comprehension. In *Proceedings of IJCAI*.

Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. 2017. Learning to reason: End-to-end module networks for visual question answering. In *Proceedings of ICCV*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of ACL*.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *Proceedings of ACL*.

Arvind Neelakantan, Quoc V Le, and Ilya Sutskever. 2016. Neural programmer: Inducing latent programs with gradient descent. In *Proceedings of ICLR*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of EMNLP*.

Scott Reed and Nando De Freitas. 2016. Neural programmer-interpreters. In *Proceedings of ICLR*.

Azriel Rosenfeld and Mark Thurston. 1971. Edge and curve detection for visual scene analysis. *IEEE Transactions on computers*, (5):562–569.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. 2019. Analysing mathematical reasoning abilities of neural models. In *Proceedings of ICLR*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *Proceedings of ICLR*.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Proceedings of NIPS*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of NIPS*.

Shuohang Wang, Mo Yu, Jing Jiang, Wei Zhang, Xiaoxiao Guo, Shiyu Chang, Zhiguo Wang, Tim Klinger, Gerald Tesauro, and Murray Campbell. 2018a. Evidence aggregation for answer re-ranking in open-domain question answering. In *Proceedings of ICLR*.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of ACL*.

Yizhong Wang, Kai Liu, Jing Liu, Wei He, Yajuan Lyu, Hua Wu, Sujian Li, and Haifeng Wang. 2018b. Multi-passage machine reading comprehension with cross-passage answer verification. In *Proceedings of ACL*.

Zhen Wang, Jiachen Liu, Xinyan Xiao, Yajuan Lyu, and Tian Wu. 2018c. Joint training of candidate extraction and answer selection for reading comprehension. In *Proceedings of ACL*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Adams Wei Yu, David Dohan, Quoc Le, Thang Luong, Rui Zhao, and Kai Chen. 2018. Fast and accurate reading comprehension by combining self-attention and convolution. In *Proceedings of ICLR*.

# Neural Duplicate Question Detection without Labeled Training Data

**Andreas Rücklé** and **Nafise Sadat Moosavi** and **Iryna Gurevych**
Ubiquitous Knowledge Processing Lab (UKP)
Department of Computer Science, Technische Universität Darmstadt
www.ukp.tu-darmstadt.de

## Abstract

Supervised training of neural models to duplicate question detection in community Question Answering (cQA) requires large amounts of labeled question pairs, which are costly to obtain. To minimize this cost, recent works thus often used alternative methods, e.g., adversarial domain adaptation. In this work, we propose two novel methods: (1) the automatic generation of duplicate questions, and (2) weak supervision using the title and body of a question. We show that both can achieve improved performances even though they do not require any labeled data. We provide comprehensive comparisons of popular training strategies, which provides important insights on how to 'best' train models in different scenarios. We show that our proposed approaches are more effective in many cases because they can utilize larger amounts of unlabeled data from cQA forums. Finally, we also show that our proposed approach for weak supervision with question title and body information is also an effective method to train cQA answer selection models without direct answer supervision.

## 1 Introduction

The automatic detection of question duplicates in community Question Answering (cQA) forums is an important task that can help users to more effectively find existing questions and answers (Nakov et al., 2017; Cao et al., 2012; Xue et al., 2008; Jeon et al., 2005), and to avoid posting similar questions multiple times. Neural approaches to duplicate detection typically require large quantities of labeled question pairs for supervised training—i.e., labeled pairs of duplicate questions that can be answered with the same information.[1]

In practice, it is often difficult to obtain such data because of the immense manual effort that is required for annotation. A large number of cQA forums thus do not contain enough labeled data for supervised training of neural models.[2] Therefore, recent works have used alternative training methods. This includes weak supervision with question-answer pairs (Qiu and Huang, 2015), semi-supervised training (Uva et al., 2018), and adversarial domain transfer (Shah et al., 2018). An important limitation of these methods is that they still rely on substantial amounts of labeled data—either thousands of duplicate questions (e.g., from a similar source domain in the case of domain transfer) or large numbers of question-answer pairs. Furthermore, unsupervised methods rely on encoder-decoder architectures that impose limitations on the model architectures and they often fall short of the performances that are achieved with supervised training (Lei et al., 2016), or they need to be combined with complex features to achieve state-of-the-art results (Zhang and Wu, 2018). To train effective duplicate question detection models for the large number of cQA forums without labeled duplicates we thus need other methods that do not require any annotations while performing on-par with supervised in-domain training.

In this work, we propose two novel methods for scenarios where we only have access to unlabeled questions (title-body), including (1) automatic duplicate question generation (*DQG*); and (2) weak supervision with the title-body pairs (*WS-TB*). Because a question body typically provides additional important information that is not included in the title (Wu et al., 2018), we hypothesize that titles and bodies have similar properties as duplicate ques-

[1] For example, "Nautilus shortcut for new blank files?" and "How do you create a new document keyboard shortcut?" are titles of labeled duplicate questions from AskUbuntu.com.

[2] Shah et al. (2018) argue that even larger StackExchange sites do not offer enough duplicates for supervised training. Further, there exist many platforms that do not contain any labeled duplicates (e.g., https://gutefrage.net).



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1607–1617,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

tions. For instance, they are only partially redundant but fundamentally describe the same question (see Figure 1 for an example). As a consequence, we can use the information from titles and bodies together with their relations to train our models.

In DQG, we use question generation models to generate a new question title from the body and then consider the generated title as a duplicate to the question's original title. In WS-TB, we take this one step further and directly train models on title-body pairs—i.e., learning to predict whether both texts belong to the same question. The advantage of our proposed methods, compared to previous work, is that they can make use of the large number of unlabeled questions (titles and bodies) in cQA forums, which is typically an order of magnitude more data than is available for supervised training.[3]

In our experiments, we evaluate common question retrieval and duplicate detection models such as RCNN (Lei et al., 2016) and BiLSTM and compare a wide range of training methods: DQG, WS-TB, supervised training, adversarial domain transfer, weak supervision with question-answer pairs, and unsupervised training. We perform extensive experiments on multiple datasets and compare the different training methods in different scenarios, which provides important insights on how to 'best' train models with varying amounts training data. We show that:

1. Training models with title-body information is very effective. With larger amounts of unlabeled questions, WS-TB and DQG outperform adversarial domain transfer from similar source domains by more than 5.8pp on average. Because the amounts of labeled question duplicates is often limited, WS-TB and DQG can in some cases achieve better performances than supervised training.

2. DQG transfers well across domains, i.e., question generation models can be applied to novel target domains to obtain generated duplicates that are suitable for model training.

3. Our training methods are effective when being used to fine-tune more recent models such as BERT (Devlin et al., 2018a).

[3] Question titles and bodies are common in all StackExchange sites, popular platforms in other languages (e.g., GuteFrage.net), and forums such as Reddit. A counterexample is Quora, which only contains question titles. However, there exists a large annotated corpus of question pairs for this forum.

**TITLE**

How to customize each Firefox window icon individually?

**BODY (1st PARAGRAPH)**

I'm a tab hoarder and I admit it. But at least I've sorted them into contextual windows now, and I'd love to have different icons for each window in the Windows task bar (not the tab bar, which is governed by the favicons). How can this be achieved?

**ANSWER**

This can be done using the free AutoHotkey. Create a .ahk text file and enter these contents: ( . . . )

Figure 1: An example question, the first paragraph of its body, and the first answer (from SuperUser[4]).

4. WS-TB can also be used to train cQA answer selection models without direct answer supervision. This shows that our methods can have broader impact on related tasks and beyond duplicate question detection.

## 2 Related Work

Duplicate question detection is closely related to question-question similarity and question retrieval. Early approaches use translation models (Jeon et al., 2005; Xue et al., 2008; Zhou et al., 2011) that were further enhanced with question category information (Cao et al., 2012) and topic models (Ji et al., 2012; Zhang et al., 2014).

More recent works in the context of the SemEval cQA challenges (Nakov et al., 2017) improve upon this and use tree kernels (TK) (Da San Martino et al., 2016), TK with neural networks (Romeo et al., 2016), neural networks with multi-task learning (Bonadiman et al., 2017), and encoder-decoder architectures together with shallow lexical matching and mismatching (Zhang and Wu, 2018). Common neural models such as CNNs achieved superior performance compared to TK when they were trained on sufficiently large numbers of labeled question pairs (Uva et al., 2018).

Similarly, neural representation learning methods have proved to be most effective in technical cQA domains. Santos et al. (2015), for example, learn representations of questions with CNNs and compare them with cosine similarity for scoring. Lei et al. (2016) propose RCNN, which extends CNN with a recurrent mechanism (adaptive gated

[4] https://superuser.com/q/1393090

| Method | Duplicates | Answers | Bodies |
|---|---|---|---|
| Supervised | × | - | (×) |
| WS-QA | - | × | (×) |
| Domain Transfer | ×* | - | (×) |
| DQG | - | - | × |
| WS-TB | - | - | × |

Table 1: The different training methods and the data they use. Models typically also use text from the bodies during training and evaluation, which we indicate with (×). ×* = domain transfer requires duplicates from a sufficiently similar source domain.

decay). This approach was further extended with question-type information (Gupta et al., 2018).

If in-domain training data is scarce—i.e., if the cQA platform does not offer enough labeled duplicates—alternative training strategies are required. If there exist some labeled question pairs (thousands), one can first train a less data-hungry non-neural model and use it for supervised training of neural models (Uva et al., 2018). Further, if there exist large numbers of labeled question-answer pairs, we can use them for weakly-supervised training (Wang et al., 2017; Qiu and Huang, 2015).

More related to our work are methods that do not rely on any labeled data in the target domain. Existing methods use unsupervised training with encoder-decoder architectures (Lei et al., 2016; Zhang and Wu, 2018), and adversarial domain transfer where the model is trained on a source domain and adversarially adapted to a target domain (Shah et al., 2018). However, such approaches typically fall short of the performances that are being achieved with in-domain supervised training.

In contrast, we propose two novel methods, DQG and WS-TB, that do not require any annotations for model training and in some cases perform better than in-domain supervised training with duplicate questions. While WS-TB is related to the approaches mentioned before, DQG is is also related to question generation (QG). Most of the previous work in QG is in the context of reading comprehension (e.g., Du et al., 2017; Subramanian et al., 2018; Zhao et al., 2018; Du and Cardie, 2018) or QG for question answering (Duan et al., 2017). They substantially differ from our approach because they generate questions based on specific answer spans, while DQG generates a new title from a question's body that can be used as a question duplicate.

## 3 Training Methods

Given a pair of questions, our goal is to determine whether they are duplicates or not. In practice, the model predictions are often used to rank a list of potentially similar questions in regard to a new user question, e.g., to retrieve the most likely duplicate for automatic question answering.

To train models, we obtain a set of examples $\{(x_1, y_1), \ldots, (x_N, y_N)\}$ in which each $x_n \in \mathcal{X}$ is an instance, i.e., a tuple containing texts such as two questions, and $y_n \in \{-1, +1\}$ is its corresponding binary label, e.g., duplicate/no-duplicate. Obtaining instances with positive labels $\mathcal{X}^+ = \{x_n^+ \in \mathcal{X} | y_n = 1\}$ is generally more difficult than obtaining $\mathcal{X}^-$ because instances with negative labels can be automatically generated (e.g., by randomly sampling unrelated questions).

In the following, we outline three existing training methods that use different kinds of instances, and in §3.2 we present our two novel methods: duplicate question generation, and weak supervision with title-body pairs. Both do not require any annotations in $\mathcal{X}^+$, and can therefore use larger amounts of data from the cQA forums. Table 1 gives an overview of the different training methods.

### 3.1 Existing Methods

**Supervised (in-domain) training** is the most common method, which requires labeled question duplicates, i.e., $x_n^+ = (q_n, \tilde{q}_n)$. Unrelated questions can be randomly sampled. With this data, we can train representation learning models (e.g., Lei et al., 2016) or pairwise classifiers (e.g., Uva et al., 2018). Most models combine the titles and bodies of the questions during training and evaluation (e.g., by concatenation), which can improve performances (Lei et al., 2016; Wu et al., 2018).

**Weak supervision with question-answer pairs (WS-QA)** is an alternative to supervised training for larger platforms without duplicate annotations (Qiu and Huang, 2015). WS-QA trains models with questions $q_n$ and accepted answers $a_n$, and therefore $x_n^+ = (q_n, a_n)$. Instances in $X^-$ can be obtained by randomly sampling unrelated answers for a question. An advantage of this method is that there typically exist more labeled answers than duplicate questions. For instance, Yahoo! answers has accepted answers but it does not contain labeled duplicate questions.

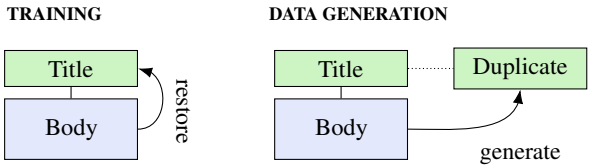


Figure 2: During training we restore the original question title from its body. During data generation we consider the generated title as a new duplicate question.

**Domain transfer** performs supervised training in a source domain and applies the trained model to a different target domain in which no labeled duplicate questions exist. Shah et al. (2018) use this method with adversarial training to learn domain-invariant question representations prior to transfer. They show that adversarial training can considerably improve upon direct transfer, but their method requires sufficiently similar source and target domains. For instance, they could not successfully transfer models between technical and other non-technical domains.

### 3.2 Proposed Methods with Unlabeled Data

The disadvantage of the existing methods is that they require labeled question duplicates, accepted answers, or similar source and target domains for transfer. We could alternatively use unsupervised training within an encoder-decoder framework, but this imposes important limitations on the network architecture, e.g., a question can only be encoded independently (no inter-attention).

Our proposed methods do not suffer from these drawbacks, i.e., they do not require labeled data and they do not impose architectural limitations.

**Duplicate question generation (DQG)** generates new question titles from question bodies, which we then consider as duplicates to the original titles. Our overall approach is depicted in Figure 2. First, we train a question generation model QG to maximize $P(\text{title}(q_n)|\text{body}(q_n))$. This is similar to news headline generation or abstractive summarization (Rush et al., 2015; Chopra et al., 2016) because QG needs to identify the most relevant aspects in the body that best characterize the question. However, restoring the exact title is usually not possible because titles and bodies often contain complementary information (see, e.g., Figure 1). We therefore consider $\text{QG}(\text{body}(q_n))$ as a duplicate of $\text{title}(q_n)$ and obtain positive labeled instances $x_n^+ = (\text{title}(q_n), \text{QG}(\text{body}(q_n)))$.

| Dataset | Train / Dev / Test | \|Q\| | \|A\| |
|---|---|---|---|
| AskUbuntu-Lei | 12,584 / 200 / 200 | 288k | 84k |
| AskUbuntu | 9106 / 1000 / 1000 | 288k | 84k |
| SuperUser | 9106 / 1000 / 1000 | 377k | 142k |
| Apple | - / 1000 / 1000 | 89k | 29k |
| Android | - / 1000 / 1000 | 47k | 14k |

Table 2: The dataset statistics. Numbers for Train/Dev/Test refer to the number of questions with duplicates. |Q| refers to the number of unlabeled questions, and |A| refers to the number of accepted answers.

Because DQG requires no annotated data, we can use this method to train duplicate detection models for all cQA forums that offer a reasonable number of unlabeled title-body pairs to obtain a suitable QG model (the smallest number of questions we tried for training of question generation models is 23k, see §5). An important advantage is that we can make use of *all questions* (after some basic filtering), which is often an order of magnitude more training data than annotated duplicates.

We can use any sequence to sequence model for QG, and we performed experiments with a Transformer (Vaswani et al., 2017) and MQAN (McCann et al., 2018).

**Weak supervision with title-body pairs (WS-TB)** takes the assumption of DQG one step further. If question titles and question bodies have similar attributes as duplicates, we could also just train duplicate detection models directly on this data without prior question generation.

In WS-TB, we thus train models to predict whether a given title and body are related, i.e., whether they belong to the same question. Therefore, $x^+ = (\text{title}(q_n), \text{body}(q_n))$.

This method considerably simplifies the sourcing of training data because it requires no separate question generation model. However, it also means that the duplicate detection model must be able to handle texts of considerably different lengths during training (for instance, bodies in SuperUser.com have an average length of 125 words). This might not be suitable for some text matching models, e.g., ones that were designed to compare two sentences.

## 4 Experiments

### 4.1 Experimental Setup

We use models and data from previous literature to obtain comparable results for evaluation, and we rely on their official implementations, default

hyperparameters, and evaluation measures. An overview of the datasets is given in Table 2, which also shows that they considerably differ in the amounts of data that is available for the different training methods.

The evaluation setup is the same for all datasets: given a user question $q$ and a list of potentially related questions, the goal is to re-rank this list to retrieve duplicates of $q$ (one or more potential related questions are labeled as duplicates). Even though some training methods do not use bodies during training, e.g., WS-DQG, during evaluation they use the same data (annotated pairs of questions with titles and bodies).[5]

**AskUbuntu-Lei.** First, we replicate the setup of Lei et al. (2016), which uses RCNN to learn dense vector representations of questions and then compares them with cosine similarity for scoring. Besides supervised training, this also includes unsupervised training with the encoder-decoder architecture. We report precision at 5 (P@5), i.e., how many of the top-5 ranked questions are actual duplicates. The dataset is based on the AskUbuntu data of Santos et al. (2015) with additional manual annotations for dev/test splits (user questions have an average of 5.7 duplicates).

**Android, Apple, AskUbuntu, and Superuser.** Second, we replicate the setup of Shah et al. (2018), which uses BiLSTM to learn question representations. This setup also includes adversarial domain transfer. The data is from the AskUbuntu, Superuser, Android, and Apple sites of StackExchange, and different to AskUbuntu-Lei, each question has only one duplicate. We measure AUC(0.05), which is the area under curve with a threshold for false positives—Shah et al. (2018) argue that this is more stable when there are many unrelated questions.

**Questions and answers.** To train the models with WS-TB and WS-QA, we use questions and answers from publicly available data dumps[6] of the StackExchange platforms. We obtain our new training sets as specified in §3.2. For instance, for WS-TB we replace every annotated duplicate $(q_n, \tilde{q}_n)$ from the original training split with $(\text{title}(q_n), \text{body}(q_n))$, and we randomly sample unrelated bodies to obtain training instances with negative labels.

It is important to note that the number of questions and answers is much larger than the number of annotated duplicate questions. Therefore, we can add more instances to the training splits with these methods. However, if not otherwise noted, we use the same number of training instances as in the original training splits with duplicates.

**DQG setup.** To train question generation models, we use the same StackExchange data. We filter the questions to ensure that the bodies contain multiple sentences. Further, if a body contains multiple paragraphs, we only keep the one with the highest similarity to the title. Details of the filtering approach are included in the Appendix. Less than 10% of the questions are discarded on average.

We train a MQAN (Multi-task Question Answering Network) model, which was proposed as a very general network architecture to solve a wide variety of tasks as part of the Natural Language Decathlon (McCann et al., 2018). The model first encodes the input with LSTMs and applies different attention mechanisms, including multi-headed self-attention. MQAN also includes pointer-generator networks (See et al., 2017), which allow it to copy tokens from the input text depending on the attention distribution of an earlier layer.

We performed the same experiments with a Transformer sequence to sequence model (Vaswani et al., 2017), but on average MQAN performed better because of its ability to copy words and phrases from the body. We include the Transformer results and a comparison with MQAN in the Appendix.

We use all available questions from a cQA forum to train the question generation model. We perform early stopping using BLEU scores to avoid overfitting. To generate duplicate questions, we then apply the trained model on all questions from the same cQA forum. We do not use a separate heldout set because this would considerably limit both the question generation training data and the number of generated duplicates. We did not observe negative effects from using this procedure.

### 4.2 Experimental Results

The results are given in Table 3. For domain transfer, we report the best scores from Shah et al. (2018), which reflects an optimal transfer setup from a similar source domain.

[5] It has been shown that including bodies in the experimental setup can lead to improved performances (Lei et al., 2016). In initial experiments, we found that the performances are mostly impacted by having access to bodies during evaluation.

[6] https://archive.org/download/stackexchange

| Measuring P@5. Results (dev / test) for RCNN | **AskUbuntu-Lei** | **Android** | **Apple** | **AskUbuntu** | **Superuser** | Average |
|---|---|---|---|---|---|---|
| | | Measuring AUC(0.05). Results for BiLSTM | | | | |
| **Trained on 1x data** (all methods use the same number of training instances as in supervised training) | | | | | | |
| Supervised (in-domain) | **48.0** / 45.0 | - | - | **0.848** | **0.944** | - |
| Unsupervised | 42.6 / 42.0 | - | - | - | - | - |
| Direct Transfer (best) | - | 0.770 | 0.828 | 0.730 | 0.908 | 0.809 |
| Adversarial Transfer (best) | - | 0.790 | 0.861 | 0.796 | 0.911 | 0.840 |
| WS-QA | 47.2 / 45.3 | 0.780 | **0.894** | 0.790 | 0.919 | 0.846 |
| DQG | 46.4 / 44.8 | 0.793 | 0.870 | 0.801 | 0.921 | 0.846 |
| WS-TB | 46.4 / **45.4** | **0.811** | 0.866 | 0.804 | 0.913 | **0.849** |
| **Trained on all available data** | | | | | | |
| Unsupervised | 43.0 / 41.8 | - | - | - | - | - |
| WS-QA | 47.3 / 44.2 | 0.814 | 0.901 | 0.828 | 0.951 | 0.874 |
| DQG | **47.4** / 44.3 | 0.833 | 0.911 | 0.855 | 0.944 | 0.886 |
| WS-TB | 47.3 / **45.3** | 0.852 | 0.910 | **0.871** | **0.952** | 0.896 |
| DQG + WS-TB (combined) | 46.4 / 44.0 | **0.863** | **0.916** | 0.866 | 0.946 | **0.898** |

Table 3: Results of the models with different training strategies. Android and Apple datasets do not contain labeled duplicates for supervised in-domain training.

**Supervised training.** As we expect, supervised in-domain training with labeled duplicates achieves better scores compared to other training methods when we consider the same number of training instances. An exception is on AskUbuntu-Lei where DQG, WS-TB, and WS-QA can achieve results that are on the same level on test or marginally worse on dev.

One reason for the better performances with labeled duplicates is that they contain more information, i.e., a pair of questions consist of two titles and two bodies compared to just one title and body for each training instance in WS-TB. However, the results show that all weakly supervised techniques as well as DQG are effective training methods.

**DQG, WS-TB, and WS-QA.** All methods outperform direct transfer from a similar source domain as well as the encoder-decoder approach on AskUbuntu-Lei. On average, WS-TB is the most effective method, and it consistently outperforms adversarial domain transfer (0.9pp on average).

We otherwise do not observe large differences between the three methods DQG, WS-TB, and WS-QA, which shows that (1) the models we use can learn from different text lengths (title-body, question-answer); and (2) the information that we extract in DQG is suitable for training (examples are given in §6). The good results of WS-TB might suggest that question generation as separate step is not required, however we argue that it can be important in a number of scenarios, e.g., when we need to train sentence matching models that would otherwise not be able to handle long texts.

**Using all available data.** One of the biggest advantages of our proposed methods is that they can use larger amounts of training data. This greatly improves the model performances for BiLSTM, where we observe average improvements of up to 4.7pp (for WS-TB). In many cases our methods now perform better than supervised training. We observe smaller improvements for WS-QA (2.8pp on avg) because it has access to fewer training instances. The performances for RCNN on AskUbuntu-Lei are mostly unchanged with minor improvements on dev. The reason is that the performances were already close to supervised training with the same data sizes.

In Figure 3 we plot the performance scores of BiLSTM averaged over the four StackExchange datasets in relation to the available training data with WS-TB. We see that the model performance consistently improves when we increase the training data (we observe similar trends for DQG and WS-QA). Thus, it is crucial to make use of all available data from the cQA forums.

We also explored a combination of our two proposed approaches where we merge their respective training sets. We find that this helps mostly for smaller cQA platforms with fewer questions (where larger training sets would be most necessary), e.g., the performances on Android and Apple improve by 0.6–1.1pp compared to WS-TB. Even though the combination does not introduce new information because both use the same question data, complementing WS-TB with DQG can provide additional variation with the generative component.

In summary, our results show that even when we

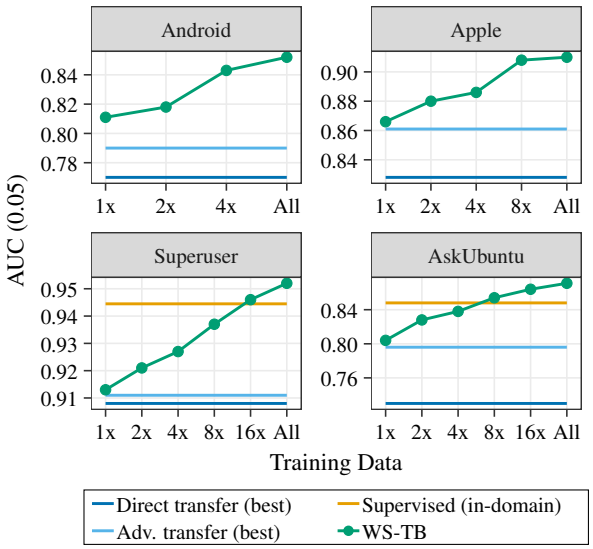


Figure 3: Performances of BiLSTM as a function of the available training data. '2x' means that there are twice as many (unlabeled) questions available to WS-TB than there are annotated duplicate questions in the original dataset (1x = 9106).

have access to sufficient numbers of labeled duplicates, the 'best' method is not always supervised training. When we use larger numbers of title-body pairs, DQG and WS-TB can achieve better performances.

## 5 Further Application Scenarios

To test if our methods are applicable to other scenarios with high practical relevance, we explore (1) whether DQG can be used in cQA forums with fewer unlabeled title-body pairs, (2) if we can use WS-TB to train answer selection models without labeled question-answer pair, and (3) how well large pre-trained language models perform when being fine-tuned with our methods.

### 5.1 DQG for Small-Scale cQA Forums

In our previous experiments, we assumed that there exist enough unlabeled questions to train the question generation model (at least 47k questions, see Table 2). To simulate a more challenging scenario with fewer in-domain questions, we explore the effects of cross-domain question generation. This is highly relevant for DQG because in such scenarios the generated duplicates could be combined with WS-TB to obtain more training data.

We replicate the transfer setup of Shah et al. (2018) where they originally transfer the duplicate question detection model from a source to a target domain. For DQG we instead train the question generation model on the source domain and generate duplicates for the target domain, with which we then train the duplicate detection model. To provide a fair comparison against adversarial domain transfer, we always use the same number of 9106 duplicates to train the duplicate detection models.

| Source | Target | Adv. DT | DQG | Δ |
|---|---|---|---|---|
| AskUbuntu | Android | 0.790 | **0.797** | +0.004 |
| | Apple | 0.855 | **0.861** | −0.009 |
| | SuperUser | 0.911 | **0.916** | −0.005 |
| SuperUser | Android | 0.790 | **0.794** | +0.001 |
| | Apple | **0.861** | **0.861** | −0.009 |
| | AskUbuntu | 0.796 | **0.809** | +0.008 |
| Academia | Android | - | 0.776 | −0.017 |
| | Apple | - | 0.854 | −0.016 |
| | SuperUser | - | 0.912 | −0.009 |
| | AskUbuntu | - | 0.760 | −0.039 |

Table 4: The domain transfer performances. Δ denotes the difference to the setup with in-domain DQG.

Results for the transfer from SuperUser and AskUbuntu to other domains are given in Table 4. They show that the question generation model for DQG can be successfully transferred across similar domains with only minor effects on the performances. Importantly, DQG still performs better than adversarial domain transfer with the same number of training instances.

To test an even more extreme case, we also transfer from StackExchange Academia (only 23k title-body pairs to train question generation) to the technical target domains. This could, e.g., be realistic for other languages with fewer cQA forums. Most notably, the performance of DQG decreases only mildly, which demonstrates its practical applicability in even more challenging scenarios. This is mostly due to the copy mechanism of MQAN, which is stable across domains (see §6).

### 5.2 Answer Selection

In answer selection we predict whether a candidate answer is relevant in regard to a question (Tay et al., 2017; Nakov et al., 2017; Tan et al., 2016; Rücklé and Gurevych, 2017), which is similar to duplicate question detection.

To test whether our strategy to train models with title-body pairs is also suitable for answer selection, we use the data and code of Rücklé et al. (2019a) and train two different types of models with WS-TB on their five datasets that are based on StackExchange Apple, Aviation, Academia, Cooking, and Travel. We train (1) a siamese BiLSTM, which

| Model | Supervised | WS-TB | WS-TB (all) |
|---|---|---|---|
| BiLSTM | 35.3 | 37.5 | 42.5 |
| COALA | 44.7 | 45.2 | 44.5 |

Table 5: Answer selection performances (averaged over five datasets) when trained with question-answer pairs vs. WS-TB.

learns question and answer representations; and (2) their neural relevance matching model COALA. Both are evaluated by how well they re-rank a list of candidate answers in regard to a question.

The results are given in Table 5 where we report the accuracy (P@1), averaged over the five datasets. Interestingly, we do not observe large differences between supervised training and WS-TB for both models when they use the same number of positive training instances (ranging from 2.8k to 5.8k). Thus, using title-body information instead of question-answer pairs to train models without direct answer supervision is feasible and effective. Further, when we use all available title-body pairs, the BiLSTM model substantially improves by 5pp, which is only slightly worse than COALA (which was designed for smaller training sets). We hypothesize that one reason is that BiLSTM can learn improved representations with the additional data. Further, title-body pairs have a higher overlap than question-answer pairs (see §6) which provides a stronger training signal to the siamese network.

These results demonstrate that our work can have broader impact to cQA, e.g., to train models on other tasks beyond duplicate question detection.

### 5.3 BERT Fine-Tuning

Large pre-trained language models such as BERT (Devlin et al., 2018b) and RoBERTa (Liu et al., 2019) have recently led to considerable improvements across a wide range of NLP tasks. To test whether our training strategies can also be used to fine-tune such models, we integrate BERT in the setups of our previous experiments.[7] We fine-tune a pre-trained BERT-base (uncased) model with supervised training, WS-TB (1x), and WS-TB (8x).

The results are given in Table 6. We observe similar trends as before but with overall better results. When increasing the number of training examples, the model performances consistently improve. We note that we have also conducted preliminary experiments with larger BERT models where we observed further improvements.

[7]We add the AskUbuntu-Lei dataset to the framework of Rücklé et al. (2019a) for our BERT experiments. Details are given in the Appendix.

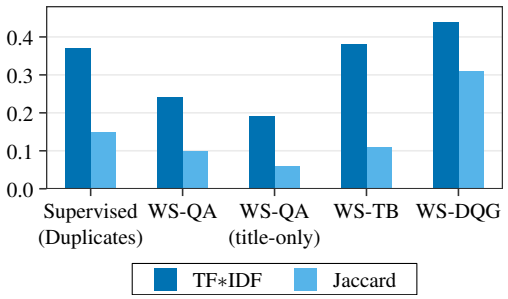


Figure 4: Average overlap of texts from positive training instances (words were stemmed and lowercased).

## 6 Analysis

### 6.1 Overlap

To analyze the differences in the training methods we calculate the overlap between the texts of positive training instances (e.g., question-question, title-body, question-answer etc.). For questions, we concatenate titles and bodies.

Figure 4 shows the Jaccard coefficient and the TF∗IDF score averaged over all instances in the four StackExchange datasets of §4.2. We observe that the overlap in WS-TB is similar to the overlap of actual duplicate questions in supervised training. The WS-DQG overlap is higher, because generated titles only contain relevant content (e.g., no conversational phrases). We also found that the BLEU scores of the MQAN model for QG are not very high (between 13.3–18.9 BLEU depending on the dataset), which shows that the texts are still sufficiently different. The overlap shows that both our methods use suitable training data with sufficiently similar, but not fully redundant texts.

Interestingly, the overlap scores of question-answer pairs are lower, especially when considering title-answer pairs as it is the case in the answer selection experiments (§5.2). This could explain one factor that may contribute to the better scores that we achieve with WS-TB for BiLSTM in this scenario. Because the overlap of title-body pairs is higher, the siamese network can receive a stronger training signal for positive instances, which could lead to better representations for similarity scoring.

### 6.2 Qualitative Analysis

To better understand the results for DQG and WS-QA, we manually checked a random sample of 200 generated questions and title-body pairs from multi-

| | AskUbuntu-Lei | Android | Apple | AskUbuntu | Superuser | Answer Selection |
|---|---|---|---|---|---|---|
| | Measuring P@5. Results for dev / test | Measuring AUC(0.05) | | | | Measuring Accuracy |
| Supervised (in-domain) | **54.0 / 52.3** | - | - | 0.862 | 0.954 | 56.8 |
| WS-TB (1x) | 47.8 / 47.2 | 0.857 | 0.908 | 0.841 | 0.932 | 55.5 |
| WS-TB (8x) | 50.4 / 49.6 | **0.896** | **0.933** | **0.897** | **0.971** | **59.7** |

Table 6: Results of fine-tuned BERT models with different training strategies.

**Title:** 14.10, 15.04 - HDMI audio not working on Dell Vostro 3750 - nVidia card not detected by aplay -l

**DQG:** ALSA not detected in nVidia

**Title:** Installing ubuntu 12.04.02 in uefi mode

**DQG:** Ubuntu 16.04 LTS boot loader not working

**Title:** Grub2 not updating

**DQG:** How to fix Grub2 error

Figure 5: Random samples of titles and DQG output.

ple platforms. Three titles and generated duplicates from AskUbuntu are shown in Figure 5.

For DQG we found that most of the generated duplicates are sensible, and most of the error cases fall into one of the following two categories:

(1) Some generated questions are somewhat off-topic because they contain information that was generated from a body that has minimal overlap with the title (see example 4 in the Appendix).

(2) A number of questions include wrong version numbers or wrong names (see example 5 in the Appendix, or the second example in Figure 5).

Generally, however, we find that many of the generated titles introduce novel information, as can be seen in Figure 5 (e.g., 'ALSA', 'boot loader' etc). The same drawbacks and benefits also apply to title-body information in WS-TB, with the exception that they are less noisy (i.e., not generated) but contain conversational phrases and many details.

We also checked the training data of the difficult DQG domain transfer case to explore reasons for the small performance decreases when transferring the question generation model. Most importantly, we find that the model often falls back to copying important phrases from the body and sometimes generates additional words from the source domain. We note that this is not the case for models without copy mechanisms, e.g., Transformer often generates unrelated text (examples are in the Appendix).

## 7 Conclusion

In this work, we have trained duplicate question detection models without labeled training data. This can be beneficial for a large number of cQA forums that do not contain enough annotated duplicate questions or question-answer pairs to use existing training methods. Our two novel methods, duplicate question generation and weak supervision with title-body pairs, only use title-body information of unlabeled questions and can thus utilize more data during training. While both are already highly effective when using the same number of training instances as other methods (e.g., outperforming adversarial domain transfer), our experiments have shown that we can outperform even supervised training when using larger amounts of unlabeled questions.

Further, we have demonstrated that weak supervision with title-body pairs is well-suited to train answer selection models without direct answer supervision. This shows that our work can potentially benefit a much wider range of related tasks beyond duplicate question detection. For instance, future work could extend upon this by using our methods to obtain more training data in cross-lingual cQA setups (Joty et al., 2017; Rücklé et al., 2019b), or by combining them with other training strategies, e.g., using our methods for pre-training.

The source code and the data of our experiments are publicly available: `http://github.com/UKPLab/emnlp2019-duplicate_question_detection`.

## Acknowledgements

This work has been supported by the German Research Foundation (DFG) as part of the QA-EduInf project (grant GU 798/18-1 and grant RI 803/12-1), by the German Federal Ministry of Education and Research (BMBF) under the promotional reference 03VP02540 (ArgumenText), and by the German Federal Ministry of Education and Research (BMBF) as part of the Software Campus program under the promotional reference 01IS17050.

## References


Daniele Bonadiman, Antonio Uva, and Alessandro Moschitti. 2017. Effective shared representations with multitask learning for community question answering. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017)*, pages 726–732.

Xin Cao, Gao Cong, Bin Cui, Christian S. Jensen, and Quan Yuan. 2012. Approaches to exploring category information for question retrieval in community question-answer archives. *ACM Transactions on Information Systems (TOIS)*, 30(2):7:1–7:38.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2016)*, pages 93–98.

Giovanni Da San Martino, Alberto Barrón Cedeño, Salvatore Romeo, Antonio Uva, and Alessandro Moschitti. 2016. Learning to re-rank questions in community question answering using advanced features. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM 2016)*, pages 1997–2000.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018a. BERT: pre-training of deep bidirectional transformers for language understanding. *arXiv preprint*, abs/1810.04805.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018b. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Xinya Du and Claire Cardie. 2018. Harvesting paragraph-level question-answer pairs from wikipedia. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pages 1907–1917.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 1342–1352.

Nan Duan, Duyu Tang, Peng Chen, and Ming Zhou. 2017. Question generation for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, pages 866–874.

Deepak Gupta, Rajkumar Pujari, Asif Ekbal, Pushpak Bhattacharyya, Anutosh Maitra, Tom Jain, and Shubhashis Sengupta. 2018. Can taxonomy help? improving semantic question matching using question taxonomy. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING 2018)*, pages 499–513.

Jiwoon Jeon, W. Bruce Croft, and Joon Ho Lee. 2005. Finding similar questions in large question and answer archives. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management (CIKM 2005)*, pages 84–90.

Zongcheng Ji, Fei Xu, Bin Wang, and Ben He. 2012. Question-answer topic model for question retrieval in community question answering. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM 2012)*, pages 2471–2474.

Shafiq Joty, Preslav Nakov, Lluís Màrquez, and Israa Jaradat. 2017. Cross-language learning with adversarial neural networks. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 226–237.

Tao Lei, Hrishikesh Joshi, Regina Barzilay, Tommi Jaakkola, Kateryna Tymoshenko, Alessandro Moschitti, and Lluís Màrquez. 2016. Semi-supervised question retrieval with gated convolutions. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2016)*, pages 1279–1289.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*.

Preslav Nakov, Doris Hoogeveen, Lluís Màrquez, Alessandro Moschitti, Hamdy Mubarak, Timothy Baldwin, and Karin Verspoor. 2017. Semeval-2017 task 3: Community question answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 27–48.

Xipeng Qiu and Xuanjing Huang. 2015. Convolutional neural tensor network architecture for community-based question answering. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI 15)*, pages 1305–1311.

Salvatore Romeo, Giovanni Da San Martino, Alberto Barrón-Cedeño, Alessandro Moschitti, Yonatan Belinkov, Wei-Ning Hsu, Yu Zhang, Mitra Mohtarami, and James Glass. 2016. Neural attention for learning to rank questions in community question answering. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING 2016)*, pages 1734–1745.

Andreas Rücklé, Steffen Eger, Maxime Peyrard, and Iryna Gurevych. 2018. Concatenated power mean embeddings as universal cross-lingual sentence representations. *arXiv preprint arXiv:1803.01400*.

Andreas Rücklé and Iryna Gurevych. 2017. Representation learning for answer selection with lstm-based importance weighting. In *Proceedings of the 12th International Conference on Computational Semantics (IWCS 2017)*.

Andreas Rücklé, Nafise Sadat Moosavi, and Iryna Gurevych. 2019a. Coala: A neural coverage-based approach for long answer selection with small data. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI 2019)*, pages 6932–6939.

Andreas Rücklé, Krishnkant Swarnkar, and Iryna Gurevych. 2019b. Improved cross-lingual question retrieval for community question answering. In *Proceedings of the World Wide Web Conference (WWW 2019)*, pages 3179–3186.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 379–389.

Cicero dos Santos, Luciano Barbosa, Dasha Bogdanova, and Bianca Zadrozny. 2015. Learning hybrid representations to retrieve semantically equivalent questions. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP 2015)*, pages 694–699.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 1073–1083.

Darsh Shah, Tao Lei, Alessandro Moschitti, Salvatore Romeo, and Preslav Nakov. 2018. Adversarial domain adaptation for duplicate question detection. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*, pages 1056–1063.

Sandeep Subramanian, Tong Wang, Xingdi Yuan, Saizheng Zhang, Adam Trischler, and Yoshua Bengio. 2018. Neural models for key phrase extraction and question generation. In *Proceedings of the Workshop on Machine Reading for Question Answering (MRQA 2018)*, pages 78–88.

Ming Tan, Cicero Dos Santos, Bing Xiang, and Bowen Zhou. 2016. Improved representation learning for question answer matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 464–473.

Yi Tay, Minh C. Phan, Luu Anh Tuan, and Siu Cheung Hui. 2017. Learning to rank question answer pairs with holographic dual lstm architecture. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2017)*, pages 695–704.

Antonio Uva, Daniele Bonadiman, and Alessandro Moschitti. 2018. Injecting relational structural representation in neural networks for question similarity. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pages 285–291.

Ashish Vaswani, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan N. Gomez, Stephan Gouws, Llion Jones, ukasz Kaiser, Nal Kalchbrenner, Niki Parmar, Ryan Sepassi, Noam Shazeer, and Jakob Uszkoreit. 2018. Tensor2tensor for neural machine translation. *arXiv preprint arXiv:1803.07416*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS 2017)*, pages 5998–6008.

Pengwei Wang, Yong Zhang, Lei Ji, Jun Yan, and Lianwen Jin. 2017. Concept embedded convolutional semantic model for question retrieval. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining (WSDM 2017)*, pages 395–403.

Wei Wu, Xu Sun, and Houfeng Wang. 2018. Question condensing networks for answer selection in community question answering. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pages 1746–1755.

Xiaobing Xue, Jiwoon Jeon, and W. Bruce Croft. 2008. Retrieval models for question and answer archives. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2008)*, pages 475–482.

Kai Zhang, Wei Wu, Haocheng Wu, Zhoujun Li, and Ming Zhou. 2014. Question retrieval with high quality answers in community question answering. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM 2014)*, pages 371–380.

Minghua Zhang and Yunfang Wu. 2018. An unsupervised model with attention autoencoders for question retrieval. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI 2018)*, pages 4978–4986.

Yao Zhao, Xiaochuan Ni, Yuanyuan Ding, and Qifa Ke. 2018. Paragraph-level neural question generation with maxout pointer and gated self-attention networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*, pages 3901–3910.

Guangyou Zhou, Li Cai, Jun Zhao, and Kang Liu. 2011. Phrase-based translation model for question retrieval in community question answer archives. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL 2011)*, pages 653–662.

# Asking Clarification Questions in Knowledge-Based Question Answering

**Jingjing Xu**[1]*, **Yuechen Wang**[2], **Duyu Tang**[3], **Nan Duan**[3], **Pengcheng Yang**[1], **Qi Zeng**[1], **Ming Zhou**[3], **Xu Sun**[1]
[1] MOE Key Lab of Computational Linguistics, School of EECS, Peking University
[2] University of Science and Technology of China
[3] Microsoft Research Asia, Beijing, China
{jingjingxu,yang_pc,pkuzengqi,xusun}@pku.edu.cn
wyc9725@mail.ustc.edu.cn
{dutang,nanduan,mingzhou}@microsoft.com

## Abstract

The ability to ask clarification questions is essential for knowledge-based question answering (KBQA) systems, especially for handling ambiguous phenomena. Despite its importance, clarification has not been well explored in current KBQA systems. Further progress requires supervised resources for training and evaluation, and powerful models for clarification-related text understanding and generation. In this paper, we construct a new clarification dataset, CLAQUA, with nearly 40K open-domain examples. The dataset supports three serial tasks: given a question, identify whether clarification is needed; if yes, generate a clarification question; then predict answers base on external user feedback. We provide representative baselines for these tasks and further introduce a coarse-to-fine model for clarification question generation. Experiments show that the proposed model achieves better performance than strong baselines. The further analysis demonstrates that our dataset brings new challenges and there still remain several unsolved problems, like reasonable automatic evaluation metrics for clarification question generation and powerful models for handling entity sparsity.[1]

## 1 Introduction

Clarification is an essential ability for knowledge-based question answering, especially when handling ambiguous questions (Demetras et al., 1986). In real-world scenarios, ambiguity is a common phenomenon as many questions are not clearly articulated, e.g. *"What are the languages used to create the source code of Midori?"* in Figure 1. There are two "*Midori*" using different programming languages, which confuses the system.

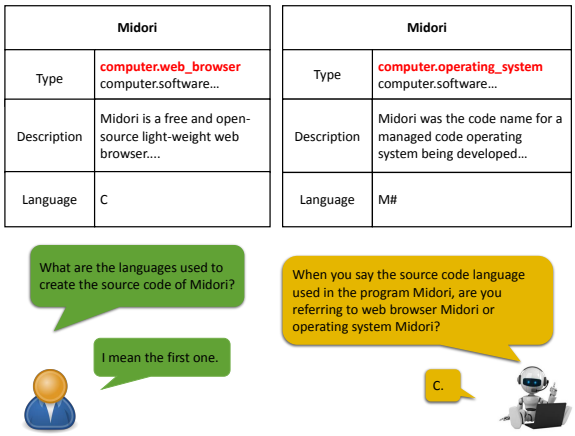


Figure 1: An example of a clarification question in KBQA. There are two entities named "Midori" using different programming languages, which confuses the system.

For these ambiguous or confusing questions, it is hard to directly give satisfactory responses unless systems can ask clarification questions to confirm the participant's intention. Therefore, this work explores how to use clarification to improve current KBQA systems.

We introduce an open-domain clarification corpus, CLAQUA, for KBQA. Unlike previous clarification-related datasets with limited annotated examples (De Boni and Manandhar, 2003; Stoyanchev et al., 2014) or in specific domains (Li et al., 2017; Rao and III, 2018), our dataset covers various domains and supports three tasks. The comparison of our dataset with relevant datasets is shown in Table 1. Our dataset considers two kinds of ambiguity for single-turn and multi-turn questions. In the single-turn case, an entity name refers to multiple possible entities in a knowledge base while the current utterance lacks necessary identifying information. In the multi-turn case, ambiguity mainly comes from the omission where a pronoun refers to multiple possible entities from the previous conversation turn. Unlike CSQA

* The work was done while Jingjing Xu and Yuechen Wang were interns in Microsoft Research, Asia.

[1]The dataset and code will be released at https://github.com/msra-nlc/MSParS_V2.0



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1618–1629,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| Dataset | Domain | Size | Task |
|---|---|---|---|
| De Boni and Manandhar (2003) | Open domain | 253 | Clarification question generation. |
| Stoyanchev et al. (2014) | Open domain | 794 | Clarification question generation. |
| Li et al. (2017) | Movie | 180K | Learning to generate responses based on previous clarification questions and user feedback in dialogue. |
| Guo et al. (2017) | Synthetic | 100K | Learning to ask clarification questions in reading comprehension. |
| Rao and III (2018) | Operating system | 77K | Ranking clarification questions in an online QA forum, StackExchange[2]. |
| **CLAQUA (Our dataset)** | Open domain | 40K | Clarification in KBQA, supporting clarification identification, clarification question generation, and clarification-based question answering. |

Table 1: The comparison of our dataset with relevant datasets. The first and second datasets focus on generating clarification questions while the small size limits their applications. The middle three tasks are either not designed for KBQA or limited in specific domains. Unlike them, we present an open-domain dataset for KBQA.

dataset (Saha et al., 2018) that constructs clarification questions base on predicate-independent templates, our clarification questions are predicate-aware and more diverse. Based on the clarification raising pipeline, we formulate three tasks in our dataset, including clarification identification, clarification question generation, and clarification-based question answering. These tasks can naturally be integrated into existing KBQA systems. Since asking too many clarification questions significantly lowers the conversation quality, we first design the task of identifying whether clarification is needed. If no clarification is needed, systems can directly respond. Otherwise, systems need to generate a clarification question to request more information from the participant. Then, external user feedback is used to predict answers.

Our main contribution is constructing a new clarification dataset for KBQA. To build high-quality resources, we elaborately design a data annotation pipeline, which can be divided into three steps: sub-graph extraction, ambiguous question annotation, and clarification question annotation. We design different annotation interfaces for single-turn and multi-turn cases. We first extract "ambiguous sub-graphs" from a knowledge base as raw materials. To enable systems to perform robustly across domains, the sub-graphs are extracted from an open-domain KB, covering domains like music, tv, book, film, etc. Based on the sub-graphs, annotators are required to write ambiguous questions and clarification questions.

We also contribute by implementing representative neural networks for the three tasks and further developing a new coarse-to-fine model for clarification question generation. Take multi-turn as an example. In the task of clarification identification, the best performing system obtains an accuracy of 86.6%. For clarification question generation, our proposed coarse-to-fine model achieves a BLEU score of 45.02, better than strong baseline models. For clarification-based question answering, the best accuracy is 74.7%. We also conduct a detailed analysis for three tasks and find that our dataset brings new challenges that need to be further explored, like reasonable automatic evaluation metrics and powerful models for handling the sparsity of entities.

## 2 Data Collection

The construction process consists of three steps, sub-graph extraction, ambiguous question annotation, and clarification question annotation. We design different annotation interfaces for single-turn and multi-turn cases.

### 2.1 Single-Turn Annotation

**Sub-graph Extraction.** As shown in Figure 2, we extract ambiguous sub-graphs from an open-domain knowledge base[3], like FreeBase. For simplification, we set the maximum number of ambiguous entities to 2. In single-turn cases, we focus on shared-name ambiguity where two entities have the same name and there is a lack of necessary distinguishing information. To construct such cases, we extract two entities sharing the same entity name and the same predicate. Predicate represents the relation between two entities. The sub-graphs provide reference for human annotators to write diverse ambiguous questions based on the shared predicates.

[2]An online QA community.

[3]To persevere anonymity, the name of the used KB is hidden for now. We will release our dataset and codes.

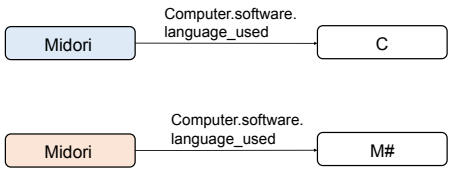


Figure 2: An extracted ambiguous sub-graph in the single-turn case. Two entities share the same name "Midori" and the same predicate "Computer.software,language_used".

| | |
|---|---|
| Shared Name | Midori |
| Predicate # 1 | computer.software.language_used |
| Predicate Description | A property relating an instance of computer.software to a programming language that was used to create the source code for the software. |
| $Q_a$: What are the languages used to create the source code of Midori? | |

Table 2: An ambiguous question annotation example in the single-turn case.

| | |
|---|---|
| $E_1$ Name | **Midori** |
| $E_1$ Type | computer.software<br>**computer.web_browser**<br>media_common.creative_work<br>... ... |
| $E_1$ Description | Midori is a free and open-source light-weight **web browser**. It uses the WebKit rendering engine and the GTK+ 2 or GTK+ 3 ... ... |
| $E_2$ Name | **Midori** |
| $E_2$ Type | **computer.operating_system**<br>computer.software |
| $E_2$ Description | Midori was the code name for a managed code **operating system** being developed by Microsoft with joint effort of Microsoft Research... ... |
| $Q_a$ | What are the languages used to create the source code of Midori? |
| $Q_c$: When you say the source code language used in the program Midori, are you referring to **[web browser Midori]** or **[operating system Midori]**? | |

Table 3: A clarification question annotation example in the single-turn case.

**Ambiguous Question Annotation.** In this step, the input is a table listing the shared entity name, predicate name, and predicate description. Based on this table, annotators need to write ambiguous questions, e.g., *"What are the languages used to create the source code of Midori?"*. An annotation example is shown in Table 2. For diversity, annotators are encouraged to paraphrase the intention words in the predicate description.

**Clarification Question Annotation.** Based on entities and the annotated ambiguous question, annotators are required to summarize distinguishing information and write a multi-choice clarification question with a special marker for separating entity information and pattern information, e.g., *"When you say the source code language used in the program Midori, are you referring to [web browser Midori] or [operating system Midori]?"*. Annotators are asked to write predicate-aware clarification questions instead of general questions, like *"Which one do you mean, A or B?"* or *"do you mean A?"* as adopted in (Saha et al., 2018). An example is shown in Table 3.

We use multi-choice as our basic type of clarification. There are three possible clarification question types, including zero-choice type (e.g., *I don't understand, can you provide more details?*), single-choice type (e.g., *Do you mean A?*), and multi-choice type (e.g., *Which one do you mean, A or B?*). The zero-choice type means that the system does not understand the question and expects more details from the participant. Though simple, it costs more user efforts as it pushes the participant to figure out what confuses the system. In comparison, the advantage of single-choice type lies in the control of conversation direction. However, if the system cannot provide user-expected choices, conversation may be longer. As for the multi-choice type, its advantage is less conversation turns to ask for more valuable information while it requires more annotation work.

## 2.2 Multi-Turn Annotation

**Sub-graph Extraction.** In multi-turn cases, ambiguity comes from the omission of the target entity name from the previous conversation turn. We first extract two connected entities. If they also share another predicate, we extract all related information as an ambiguous sub-graph, as shown in Figure 3. By asking the shared predicate, we get an ambiguous question.

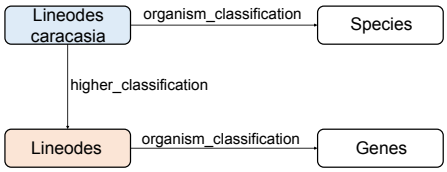


Figure 3: An extracted ambiguous sub-graph in the multi-turn case. "Lineodes caracasia" and "Lineodes" are linked and share the same relation "organism_classification".

| | |
|---|---|
| $E_1$ Name | **Lineodes caracasia** |
| $E_2$ Name | **Lineodes** |
| Predicate #1 | biology.higher_classification |
| Predicate Description | Property relating an biology.organism_classification to its higher or parent biology.organism_classification. |
| Predicate #2 | biology.organism_classification |
| Predicate Description | Relating an biology.organism to its biology.organism_classification, the formal or informal taxonomy grouping for the organism. |
| $Q_h$: What is the higher classification for **Lineodes caracasia?**<br>$R_h$: **Lineodes.**<br>$Q_a$: What is the biological classification? | |

Table 4: An ambiguous question annotation example in the multi-turn case.

| | |
|---|---|
| $E_1$ Name | **Lineodes caracasia** |
| $E_1$ Type | media_common.cataloged_instance<br>biology.organism_classification<br>... ... |
| $E_1$ Description | Lineodes caracasia is a moth in the family Crambidae... ... |
| $E_2$ Name | **Lineodes** |
| $E_2$ Type | media_common.cataloged_instance<br>biology.organism_classification |
| $E_2$ Description | Lineodes is a genus of moths of the Crambidae family. ... ... |
| $Q_h$<br>$R_h$<br>$Q_a$ | What is the higher classification for Lineodes caracasia?<br>Lineodes.<br>What is the biological classification? |
| $Q_c$: Are you referring to **[Lineodes caracasia]** or **[Lineodes]**, when you ask the biological classification? | |

Table 5: A clarification question annotation example in the multi-turn case.

**Ambiguous Question Annotation.** An annotation example is shown in Table 4. Based on two entities in the extracted sub-graph, annotators construct a conversation turn and then write an ambiguous question where the entity name is omitted, e.g, *"What is the biological classification?"*.

**Clarification Question Annotation.** The annotation guideline of this step is the same as in single-turn annotation. The input includes two candidate entities and the annotated ambiguous question. The output is a clarification question. An annotation example is shown in Table 5.

## 3 Tasks

Our dataset aims to enable systems to ask clarification questions in open-domain question answering. Three tasks are designed in this work, including clarification identification, clarification question generation, and clarification-based question answering. Figure 4 shows the distribution of the top-10 domains in our dataset.

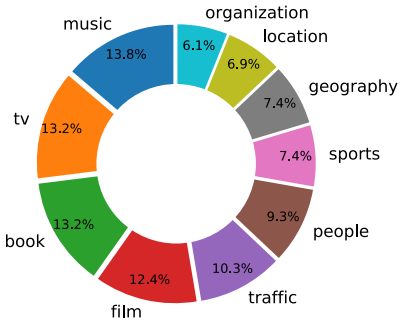


Figure 4: The distribution of top-10 domains.

### 3.1 Clarification Identification

Clarification identification can be regarded as a classification problem. It takes conversation context and candidate entity information as input. The output is a label identifying whether clarification is needed. Specifically, the input is $\{Q_a, E_1, E_2\}$ for single-turn cases or $\{Q_p, R_p, Q_a, E_1, E_2\}$ for multi-turn cases. $Q_a$ represents the current question. $E_1$ and $E_2$ represent the candidate entities. $Q_p$ and $R_p$ are question and response from the previous conversation turn. The output is a binary label from set $Y = \{0, 1\}$ where $1$ indicates that the input question is ambiguous and $0$ indicates the opposite.

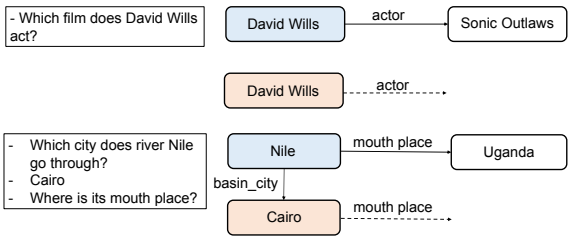


Figure 5: Negative examples of unambiguous sub-graphs and annotated conversations. The dotted line means the non-existent relation. The upper figure is in the single-turn case and the lower one is in the multi-turn case.

As Figure 5 shows, negative examples are annotated in the same way as the positive (ambiguous) examples do, but without the clarification related steps. In their sub-graphs, one of the entities has its unique predicate. The unique predicates are included in the user questions like "Where is its mouth place?". As only river "Nile" has mouth places, this question is unambiguous.

| | Single-Turn | | Multi-Turn | |
|---|---|---|---|---|
| | Positive | Negative | Positive | Negative |
| Train | 8,139 | 6,841 | 12,173 | 8,289 |
| Dev | 487 | 422 | 372 | 601 |
| Test | 637 | 673 | 384 | 444 |

Table 6: Statistics of the dataset. It is important to note that three tasks share the same split. Since clarification question generation and clarification-based question answering are built upon ambiguous situations, they only use the positive data in their training, development, and test sets.

The statistics of our dataset are shown in Table 6. It is important to note that three tasks share the same split. Since clarification question generation and clarification-based question answering are built upon ambiguous situations, they only use the positive (ambiguous) data in their training, development, and test sets. For generalization, we add some examples with unseen entities and predicates into the development and test sets.

### 3.2 Clarification Question Generation

This text generation task takes ambiguous context and entity information as input, and then outputs a clarification question. For single-turn cases, the input is $\{Q_a, E_1, E_2\}$. For multi-turn cases, the input is $\{Q_p, R_p, Q_a, E_1, E_2\}$. In both cases, the output is a clarification question $Q_c$. We use BLEU as the automatic evaluation metric.

### 3.3 Clarification-Based Question Answering

As our dataset focuses on simple questions, this task can be simplified as the combination of entity identification and predicate identification. Entity identification is to extract entity $e$ from candidate entities based on current context. Predicate identification is to choose predicate $p$ that describes the ambiguous question. The extracted entity $e$ and predicate $p$ are combined to query the knowledge triple $(e, p, o)$. The model is correct when both tasks are successfully completed.

Additional user responses toward clarification questions are also necessary for this task. Due to the strong pattern in responses, we use a template-based method to generate the feedback. We first design four templates, "I mean the first one", "I mean the second one", "I mean [entity name]", "I mean [entity type] [entity name]". Then for each clarification example, we randomly select a candidate entity and fill its information into the template as the user response, e.g., "I mean web browser Midori." for the clarification question in Table 3. For entity identification, we use the selected entity as the gold label. For predicate prediction, we use the predicate in the current ambiguous question as the gold label. In muli-turn cases, the predicate label is *Predicate #1* as shown in Table 2. In single-turn cases, the predicate label is *Predicate #2* as shown in Table 4.

Entity identification and predicate identification are both classification tasks. They take the same information as input, including ambiguous context, clarification question, and user response. For single-turn cases, the input is $\{Q_a, E_1, E_2, Q_c, R_c\}$ where $R_c$ is the feedback of the participant toward clarification question $Q_c$. For multi-turn cases, the input is $\{Q_p, R_p, Q_a, E_1, E_2, Q_c, R_c\}$. For entity identification, the output is a label from set $Y_E = \{E_1, E_2\}$. For predicate identification, we randomly choose 1 negative predicates from the training set and combine them with the gold predicate together as the candidate set $Y_P$. The output of predicate identification is the index of the gold predicate in $Y_P$.

## 4 Approach

For three tasks, we implement several representative neural networks as baselines. Here we introduce these models in detail.

### 4.1 Classification Identification Models

The input of classification models is the ambiguous context and entity information. For simplification, we use a special symbol, `[S]`, to concatenate all the entity information together. These two parts can be regarded as different source information. Based on whether the inter-relation between different source information is explicitly explored, the classification models can be classified into two categories, unstructured models and structured models.

Unstructured models concatenate different source inputs into a long sequence with a special separation symbol `[SEL]`. We implement several widely-used sequence classification models, including Convolutional Neural Network (CNN) (Kim, 2014), Long-Short Term Memory Network (LSTM) (Schuster and Paliwal, 1997), and Recurrent Convolutional Neural Network (RCNN) (Lai et al., 2015), Transformer. The details of the models are shown in Supplementary Materials.

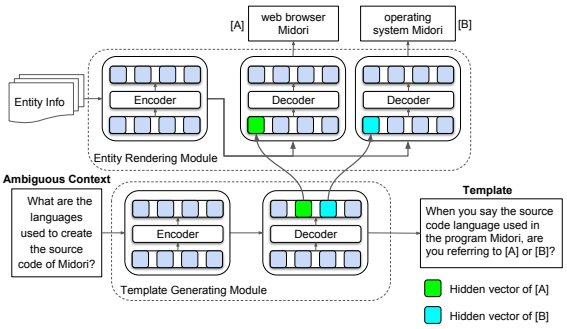


Figure 6: An illustration of the proposed coarse-to-fine model. The proposed model consists of a template generating module and an entity rendering module. The former is used to generate a clarification template based on ambiguous question, e.g., "When you say the source code language used in the program Midori, are you referring to [A] or [B]?". The latter is used to fill up the generated template with detailed entity information.

Structured models use separate structures to encode different source information and adopt an additional structure to model the inter-relation of the source information. Specifically, we use two representative neural networks, Hierarchical Attention Network (HAN) (Yang et al., 2016) and Dynamic Memory Network (DMN) (Kumar et al., 2016), as our structured baselines. The details of the models are shown in Supplementary Materials.

### 4.2 Clarification Question Generation Models

The input of the generation model is the ambiguous context and entity information. In single-turn cases, the ambiguous context is current question $Q_a$. In multi-cases, the input is current question and the previous conversation turn $\{Q_p, R_p, Q_a\}$. We use `[S]` to concatenate all entity information together, and use `[SEL]` to concatenate entity information and context information into a long sequence. We adopt Seq2Seq (Bahdanau et al., 2015) and Transformer (Vaswani et al., 2017) as our baselines and further develop a new generation model. The details of baselines can be found in Supplementary Materials.

**Coarse-to-fine Model.** Generally speaking, a clarification question contains two parts, entity phrases (e.g., "*web browser Midori*") and pattern phrases ( e.g. *"When you say the source code language used in the program Midori, are you referring to [A] or [B]?*"). The entity phrase is summarized from the given entity information for distinguishing between two entities. The pattern phrase is used to locate the position of ambiguity, which is closely related with the context. In summary, two kinds of phrases refer to different source information. Based on this feature, we propose a new coarse-to-fine model, as shown in Figure 6. Similar ideas have been successfully applied to semantic parsing (Dong and Lapata, 2018).

The proposed model consists of a template generating module $T_\theta$ and an entity rendering module $R_\phi$. $T_\theta$ first generates a template containing pattern phrases and the symbolic representation of the entity phrases. Then, the symbolized entities contained in the generated template are further properly rendered by the entity rendering module $R_\phi$ to reconstruct complete entity information. Since the annotated clarification questions explicitly separate entity phrases and pattern phrases, we can easily build training data for these two modules. For clarity, the template is constructed by replacing entity phrases in a clarification question with special symbols, `[A]` and `[B]`, which represent the positions of two entities.

The template generating module $T_\theta$ uses Transformer (Vaswani et al., 2017) as implementation. The input is the ambiguous context and the output is a clarification template.

Similar to $T_\theta$, the entity rendering module $R_\phi$ is also implemented with a Transformer structure. More specifically, for each symbolized entity in the template, we add the hidden state of the decoder of $T_\theta$ corresponding to this symbolized entity into the decoder input.

### 4.3 Clarification-Based Question Answering Models

This task contains two sub-tasks, entity identification and predicate identification. They share the same input. In single-turn cases, the input is $\{Q_a, Q_c, R_c, E_1, E_2\}$. In multi-cases, the input is $\{Q_p, R_p, Q_a, Q_c, R_c, E_1, E_2\}$. Considering these two sub-tasks both belong to classification tasks, we use the same models as used in the task of clarification identification for implementation.

## 5 Experiments

We report the experiment results on three formulated tasks and give a detailed analysis. More details of hyper-parameters are in Supplementary Materials.

| Model | Single-Turn | Multi-Turn |
|---|---|---|
| CNN | 83.3 | 80.0 |
| RNN | 83.8 | 72.7 |
| Transformer | 84.0 | 72.8 |
| HAN | 83.1 | 73.0 |
| DMN | **86.6** | **80.5** |

Table 7: Results of clarification identification models.

### 5.1 Experiment Results

**Clarification Identification.** As shown in Table 7, the structured models have the best performance, with a 86.6% accuracy on single-turn cases and a 80.5% accuracy on multi-turn cases. Compared to the best-performing unstructured model, the structured architecture brings obvious performance improvements, with 2.6% and 6.7% accuracy increases. It indicates that the structured architectures have learned the inter-relation between the context and entity details, which is a vital part for reasoning whether a question is ambiguous. We also find that the models all achieve better results on multi-turn cases. Multi-turn cases contain additionally previous conversation turn, which can help the models capture more key phrases from the entity information.

**Clarification Question Generation.** Table 8 shows that Seq2Seq achieves low BLEU scores, which indicates its tendency to generate irrelevant text. Transformer achieves higher performance than Seq2Seq. Our proposed coarse-to-fine model demonstrates a new state of the art, improving the current highest baseline result by 3.35 and 0.60 BLEU scores, respectively.

We also find that multi-turn cases generally obtain higher BLEU scores than single-turn cases. In single-turn cases, the same-name characteristic makes it hard to summarize key distinguishing information. As a comparison, in multi-turn cases, two candidate entities usually have different names and it is easier to generate clarification questions. It is due to the flexible structure of ambiguous questions, which makes it hard to define universal rules to identify the asked objects. Table 9 shows a an example generated by the proposed model.

**Clarification-Based Question Answering.** As Table 10 shows, the unstructured models perform better than the structured models, indicating that this task tends to be over-fitting and small models have better the generalization ability.

| Model | Single-Turn | Multi-Turn |
|---|---|---|
| Seq2Seq | 18.84 | 31.62 |
| Transformer | 20.69 | 44.42 |
| **The proposed Model** | **24.04** | **45.02** |

Table 8: Results of clarification question generation models.

| | |
|---|---|
| $E_1$ Name | Lineodes interrupta |
| $E_1$ Type | biology.organism_classification |
| $E_1$ Description | Lineodes interrupta is a moth in the Crambidae family. It was described by Zeller in 1873. It is found in Mexico and the United States, where it has been recorded from Florida, Louisiana, Oklahoma and Texas. |
| $E_2$ Name | Lineodes |
| $E_2$ Type | biology.organism_classification |
| $E_2$ Description | Lineodes is a genus of moths of the Crambidae family. |
| $Q_p$ | What is higher classification of Lineodes interrupta? |
| $R_p$ | Lineodes. |
| $Q_a$ | Biologically speaking, what is the classification? |
| **Output**: Are you referring to Lineodes interrupta or Lineodes, when you ask the biological classification? | |

Table 9: An example of the generated clarification questions.

### 5.2 Discussion

**Automatic Evaluation.** In the task of clarification question generation, we use BLEU, a widely used evaluation metric on language generation tasks, as the automatic evaluation metric. BLEU evaluates the generated text by computing its similarity with the gold text. However, we find this metric not suitable for our task. A good answer can ask different aspects as long as it can distinguish the entities. The given answer in our dataset is one of possible answers but not the only correct answer. A good answer may get a low BLEU score. Therefore, a more reasonable evaluation metric needs to be explored in the future, like paraphrase-based BLEU.

**Error Analysis.** In clarification question generation, although our proposed model has achieved the best performance, it still generates some low-quality questions. We conduct a human evaluation on 100 generated clarification questions and summarize two error categories. The first one is the entity error where the generated clarification question has a correct multi-choice structure but with irrelevant entity information. Here we present one example in Table 11. In this example, our model generates entity phrases irrelevant to the input am-

| Model | Single-Turn | Multi-Turn |
|---|---|---|
| CNN | **60.8** | 74.7 |
| RNN | 56.4 | **76.0** |
| Transformer | 52.1 | 58.9 |
| DMN | 50.9 | 54.5 |
| HAN | 51.7 | 54.7 |

Table 10: Results of clarification-based question answering models.

| | |
|---|---|
| $E_1$ Name | **Come Back, Little Sheba** |
| $E_1$ Type | award.winning_work<br>award.nominated_work<br>... ... |
| $E_1$ Description | Come Back, Little Sheba is a 1950 theater production of the play by William Inge... ... |
| $E_2$ Name | **Come Back, Little Sheba** |
| $E_2$ Type | theater.production<br>award.nominated_work |
| $E_2$ Description | Come Back, Little Sheba is a 2008 theater production of the play by William Inge.... ... |
| $Q_h$: Who directed Come Back, Little Sheba?<br>**Output**: Which one do you mean, winning work visual artist Claudia Coleman or winning work Elmer Gantry, when you ask the directors? | |

Table 11: A bad generated case.

biguous question. This failure is mainly due to the unseen entity name "Come Back, Little Sheba", which leads the model to wrong generation decisions. Since there are lots of entities with different names and descriptions, how to handle the sparse information is a non-trivial problem. The second one is the grammar error where models sometimes generate low-quality phrases after generating a low-frequency entity word. These errors both can be attributed to the sparsity of entities to some extent. How to deal with the sparsity of entity information still needs further exploration.

## 6 Related Work

Our work is related to clarification question and question generation.

### 6.1 Clarification Question in Other Tasks

There are several studies on asking clarification questions. Stoyanchev et al. (2014) randomly drop one phrase from a question and require annotators to ask a clarification question toward the dropped information, e.g.*"Do you know the birth data of XXX"*. However, the small dataset size makes it hard to help downstream tasks. Following this work, Guo et al. (2017) provide a larger synthetic dataset QRAQ by replacing some of entities with variables. Li et al. (2017) use misspelled words to replace entities in questions to build ambiguous questions. Although these studies are good pioneering studies, the synthetic constructing way makes them unnatural and far from real world questions.

Different from these studies, Braslavski et al. (2017) investigate a community question answering (CQA) dataset and study how to predict the specific subject of a clarification question. Similarly, Rao and III (2018) focus on learning to rank human-written clarification questions in an online QA forum, StackExchange. Our work differs from these two studies in that we have a knowledge graph at the backend, and the clarification-related components are able to facilitate KBQA.

### 6.2 Question Generation

For different purposes, there are various question generation tasks. Hu et al. (2018) aim to ask questions to play the 20 question game. Dhingra et al. (2017) teach models to ask questions to limit the number of answer candidates in task-oriented dialogues. Ke et al. (2018) train models to ask questions in open-domain conversational systems to better interact with people. Guo et al. (2018) develop a sequence-to-sequence model to generate natural language questions.

## 7 Conclusion and Future Work

In this work, we construct a clarification question dataset for KBQA. The dataset supports three tasks, clarification identification, clarification question generation, and clarification-based question answering. We implement representative neural networks as baselines for three tasks and propose a new generation model. The detailed analysis shows that our dataset brings new challenges. More powerful models and reasonable evaluation metrics need further explored.

In the future, we plan to improve our dataset by including more complex ambiguous situations for both single-turn and multi-turn questions, such as multi-hop questions, aggregation questions, etc. We also plan to integrate the clarification-based models into existing KBQA system, and study how to iteratively improve the model based on human feedback.

## Acknowledgments

We thank all reviewers for providing the thoughtful and constructive suggestions. This work was

supported in part by National Natural Science Foundation of China (No. 61673028). Jingjing Xu and Xu Sun are the corresponding authors of this paper.

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Pavel Braslavski, Denis Savenkov, Eugene Agichtein, and Alina Dubatovka. 2017. What do you mean exactly?: Analyzing clarification questions in cqa. In *Proceedings of the 2017 Conference on Conference Human Information Interaction and Retrieval*, pages 345–348. ACM.

Marco De Boni and Suresh Manandhar. 2003. An analysis of clarification dialogue for question answering. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–55. Association for Computational Linguistics.

Marty J Demetras, Kathryn Nolan Post, and Catherine E Snow. 1986. Feedback to first language learners: The role of repetitions and clarification questions. *Journal of child language*, 13(2):275–292.

Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2017. Towards end-to-end reinforcement learning of dialogue agents for information access. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 484–495.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. *arXiv preprint arXiv:1805.04793*.

Daya Guo, Yibo Sun, Duyu Tang, Nan Duan, Jian Yin, Hong Chi, James Cao, Peng Chen, and Ming Zhou. 2018. Question generation from sql queries improves neural semantic parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1597–1607. Association for Computational Linguistics.

Xiaoxiao Guo, Tim Klinger, Clemens Rosenbaum, Joseph P Bigus, Murray Campbell, Ban Kawas, Kartik Talamadupula, Gerry Tesauro, and Satinder Singh. 2017. Learning to query, reason, and answer questions on ambiguous texts. In *ICLR 2017*.

Huang Hu, Xianchao Wu, Bingfeng Luo, Chongyang Tao, Can Xu, Wei Wu, and Zhan Chen. 2018. Playing 20 question game with policy-based reinforcement learning. *arXiv preprint arXiv:1808.07645*.

Pei Ke, Jian Guan, Minlie Huang, and Xiaoyan Zhu. 2018. Generating informative responses with controlled sentence function. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 1499–1508.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *International Conference on Machine Learning*, pages 1378–1387.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *AAAI*, volume 333, pages 2267–2273.

Jiwei Li, Alexander H. Miller, Sumit Chopra, Marc'Aurelio Ranzato, and Jason Weston. 2017. Dialogue learning with human-in-the-loop. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Sudha Rao and Hal Daumé III. 2018. Learning to ask good questions: Ranking clarification questions using neural expected value of perfect information. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 2736–2745.

Amrita Saha, Vardaan Pahuja, Mitesh M Khapra, Karthik Sankaranarayanan, and Sarath Chandar. 2018. Complex sequential question answering: Towards learning to converse over linked question answer pairs with a knowledge graph. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Svetlana Stoyanchev, Alex Liu, and Julia Hirschberg. 2014. Towards natural clarification questions in dialogue systems. In *AISB symposium on questions, discourse and dialogue*, volume 20.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6000–6010.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

# A Supplementary Materials

## A.1 Classification Models

| Model | Hyper-parameter | Setting |
|---|---|---|
| **CNN** | Batch size | 64 |
| | Embedding size | 128 |
| | Hidden size | 128 |
| | Filter size | 128 |
| | Kernel size | 100 |
| | Learning rate | 0.0003 |
| | Optimizer | Adam |
| **RNN** | Batch size | 8 |
| | Embedding size | 128 |
| | Hidden size | 128 |
| | Learning rate | 2 |
| | Optimizer | Adam |
| **Transformer** | Batch size | 8 |
| | Embedding size | 128 |
| | Hidden size | 128 |
| | Learning rate | 2 |
| | Dropout | 0.1 |
| | Optimizer | Adam |
| **HAN** | Batch size | 64 |
| | Embedding size | 100 |
| | Hidden size | 100 |
| | Learning rate | 0.01 |
| | Optimizer | Adam |
| **DMN** | Batch size | 64 |
| | Embedding size | 300 |
| | Hidden size | 300 |
| | Learning rate | 0.005 |
| | Optimizer | Adam |

Table 12: Settings of classification models.

Table 12 shows the detailed hyper-parameter settings of these classification models. Here we introduce the classification models used in our experiments.

**CNN.** It first feeds the input embedding sequence into a convolutional layer. The convolutionial layer contains $K$ filters $f \in R^{s,d}$ where $s$ is the size of filters and $d$ is the dimension of word embeddings. We use different sizes of filters to get rich features. Then, a max pooling layer takes $K$ vectors as input and generates a distributed input representation $\boldsymbol{h}$. Finally, the vector $\boldsymbol{h}$ is projected to the probability of labels.

**RNN.** It is built on a traditional bi-directional gated recurrent unit (GRU) structure, which is used to capture global and local dependencies inside the input sequence. The last output of GRU is then fed into the output layer to generate the probability of labels.

**Transformer.** A model similar with RNNs. It removes the recurrent unit in RNNs and replaces it with a special attention mechanism. called multi-head attention. The model is composed of a stack of 2 identical layers. Each identical layer has two sub-layers. The first sub-layer is a multi-head self-attention layer, and the second is a position-wise fully connected feed-forward layer. Since input nodes are not sequentially in order, the positional embedding layer is removed from the model. The decoder is also composed of a stack of 2 identical layers.

| Model | Hyper-parameter | Setting |
|---|---|---|
| **Seq2Seq** | Batch size | 16 |
| | Embedding size | 32 |
| | Hidden size | 32 |
| | Learning rate | 2 |
| | Beam size | 1 |
| | Dropout | 0.1 |
| | Optimizer | Adam |
| **Transformer** | Batch size | 16 |
| | Embedding size | 32 |
| | Hidden size | 32 |
| | Learning rate | 2 |
| | Beam size | 1 |
| | Dropout | 0.1 |
| | Optimizer | Adam |
| **Our Model** | Batch size | 16 |
| | Embedding size | 32 |
| | Hidden size | 32 |
| | Learning rate | 2 |
| | Beam size | 1 |
| | Dropout | 0.1 |
| | Optimizer | Adam |

Table 13: Settings of generation models.

**HAN.** It is built upon two self-attention layers. The first layer is responsible for encoding context information and entity information. Then, the second layer uses a self-attention mechanism to learn the inter-relations between them. Finally, the output of the second layer is fed into the output layer for predicting labels.

**DMN.** This model regards context information as query and entity information as input to learn their inter-relations. It consists of four modules, an input module, a question module, an episodic memory module, and an answer module. The input module and thed question module encode query and input into distributed vector representations. The episodic memory module decodes which parts of the entity information to focus on through the attention mechanism. Finally, the answer module generates an answer based on the final memory vector of the memory module.

### A.2 Clarification Question Generation Models

Here is the detailed introduction about the generation baselines.

**Seq2Seq (Bahdanau et al., 2015).** This model is based on a traditional encoder-decoder framework which encodes the input sequence into a dense vector and then decodes the target sequence word by word. Attention is used to capture global dependencies between input and output.

**Transformer (Vaswani et al., 2017).** It is based solely on attention mechanism to capture global dependencies. Similar to Seq2Seq, it uses an encoder-decoder framework but with different implementation.

Table 13 shows the detailed hyper-parameter settings of generation models.

# Multi-View Domain Adapted Sentence Embeddings for Low-Resource Unsupervised Duplicate Question Detection

**Nina Poerner**[1,2] **and Hinrich Schütze**[1]
[1]Center for Information and Language Processing, LMU Munich, Germany
[2]Corporate Technology Machine Intelligence (MIC-DE), Siemens AG Munich, Germany
poerner@cis.uni-muenchen.de | inquiries@cislmu.org

## Abstract

We address the problem of Duplicate Question Detection (DQD) in low-resource domain-specific Community Question Answering forums. Our multi-view framework MV-DASE combines an ensemble of sentence encoders via Generalized Canonical Correlation Analysis, using unlabeled data only. In our experiments, the ensemble includes generic and domain-specific averaged word embeddings, domain-finetuned BERT and the Universal Sentence Encoder. We evaluate MV-DASE on the CQADupStack corpus and on additional low-resource Stack Exchange forums. Combining the strengths of different encoders, we significantly outperform BM25, all single-view systems as well as a recent supervised domain-adversarial DQD method.

## 1 Introduction

Duplicate Question Detection is the task of finding questions in a database that are equivalent to an incoming query. Many Community Question Answering (CQA) forums leave this task to the collective memory of their users. This results in unnecessary manual work for community members as well as delayed access to answers (Hoogeveen et al., 2015).

Automatic DQD is often approached as a supervised problem with community-generated training labels. However, smaller CQA forums may suffer from label sparsity: On Stack Exchange, 50% of forums have fewer than 160 user-labeled duplicates, and 25% have fewer than 50 (see Figure 1).[1]

To overcome this problem, two avenues have been explored: The first is supervised domain-adversarial training on a label-rich source forum (Shah et al., 2018), which works best when source and target domains are related. The second is unsupervised DQD via representation learning (Charlet and Damnati, 2017; Lau and Baldwin, 2016), which requires only unlabeled questions. In this paper, we take the unsupervised avenue.

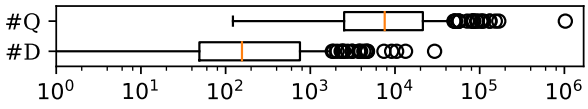


Figure 1: Distribution (log-scale box plot) of number of questions (#Q) and number of labeled duplicates (#D) on Stack Exchange. $N = 165$ forums.

A major challenge in the context of domain-specific CQA forums is that language usage may differ from the "generic" domains of existing representations. To illustrate this point, compare the following Nearest Neighbor lists of the word "tree", based either on generic GloVe embeddings (Pennington et al., 2014) or on FastText embeddings (Bojanowski et al., 2017) that were trained on specific CQA forums:

**generic (GloVe):** trees, branches, leaf
**chess:** searches, prune, modify
**outdoors:** trees, trunk, trunks
**gis:** strtree, rtree, btree
**wordpress:** trees, hierachy, hierarchial
**gaming:** trees, treehouse, skills

Charlet and Damnati (2017) and Lau and Baldwin (2016) report that representations trained on in-domain data perform better on unsupervised DQD than generic representations. But in a low-resource setting, the amount of unlabeled in-domain data is limited. This can result in low coverage or quality, as illustrated by the in-domain embedding neighbors of "tree" in the smallest forum from our dataset:

**windowsphone:** dreamspark, l535ds, generally

[1]archive.org/details/stackexchange [data dump: 2018-12-20]



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1630–1641, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| | generic | domain-specific |
|---|---|---|
| **contextualized** | $f_G$: GloVe | $f_D$: FastText (in-domain) |
| **noncontextualized** | $f_U$: USE | $f_B$: BERT (domain-finetuned) |

Table 1: Ensemble used in our experiments.

It is therefore desirable to combine the overall quality and coverage of generic representations with the domain-specificity of in-domain representations via multi-view learning. There is a large body of work on multi-view word embeddings (see Section 2.3), including domain adapted word embeddings (Sarma et al., 2018).

Recent representation learning techniques go beyond the word level and embed larger contexts (e.g., sentences) jointly (Peters et al., 2018; Devlin et al., 2019; Cer et al., 2018). To reflect this paradigm shift, we take multi-view representation learning from the word to the sentence level and propose MV-DASE (Multi-View Domain Adapted Sentence Embeddings), a framework that combines an ensemble of sentence encoders via Generalized Canonical Correlation Analysis (see Section 3.1).

MV-DASE uses **unlabeled in-domain data only**, making it applicable to the problem of unsupervised DQD. As a framework, it is **agnostic** to the internal specifics of its ensemble. In Section 3.2, we describe an ensemble of different sentence encoders: domain-specific and generic, contextualized and noncontextualized (see Table 1). In Sections 4 and 5, we demonstrate that MV-DASE is effective at **duplicate retrieval** on the CQADupStack corpus (Hoogeveen et al., 2015) and on additional low-resource Stack Exchange forums. **Significance tests** show significant gains over BM25, all single-view systems and domain-adversarial supervised training as proposed by Shah et al. (2018). In Sections 6 and 7, we successfully evaluate MV-DASE on **two additional benchmarks**: the SemEval-2017 DQD shared task (Nakov et al., 2017) as well as the unsupervised STS Benchmark (Cer et al., 2017).

## 2 Related Work

### 2.1 Duplicate Question Detection

Most prior work on DQD (e.g., Bogdanova et al. (2015); Dos Santos et al. (2015); Baldwin et al. (2016); Zhang et al. (2017); Rodrigues et al. (2017); Hoogeveen et al. (2018)) focuses on supervised architectures. As discussed, these approaches are not applicable to forums with few or no labeled duplicates.

Shah et al. (2018) tackle label sparsity by domain-adversarial training (ADA). More specifically, they train a bidirectional Long-Short Term Memory Network (LSTM) (Hochreiter and Schmidhuber, 1997) on a label-rich source forum, while minimizing the distance between source and target domain representations. Their approach beats BM25 and a simple transfer baseline in cases where source and target domain are closely related (e.g., *AskUbuntu→SuperUser*), but not on more distant pairings. This is not ideal, as the existence of a big related source forum is not guaranteed.

An alternative is unsupervised DQD via representation learning, which does not require any labels. Charlet and Damnati (2017) use a word embedding-based soft cosine distance for duplicate ranking. In a recent DQD shared task (SemEval-2017 task 3B, Nakov et al. (2017)), their best unsupervised system trails the best supervised system by only 2% Mean Average Precision (MAP). This seems reasonable, given that the implicit objective of many representation learning methods (similar representations for similar objects) is closely related to the notion of a duplicate.

Charlet and Damnati (2017) report overall better results when embeddings are trained on domain-specific data rather than Wikipedia. However, they make no attempts to combine the two domains. Lau and Baldwin (2016) evaluate two representation learning techniques (doc2vec (Le and Mikolov, 2014) and word2vec (Mikolov et al., 2013a)) on CQADupStack. They also report better results when representations are learned on domain-specific rather than generic data.

### 2.2 Sentence embeddings and STS

Unsupervised DQD is related to the task of unsupervised Semantic Textual Similarity (STS), i.e., sentence similarity scoring (Cer et al., 2017). Arora et al. (2017) show that a weighted average over pre-trained word embeddings, followed by principal component removal, is a strong baseline for STS. We use their weighting scheme, Smooth Inverse Frequency (SIF), in Section 3.2.

Averaged word embeddings are insensitive to word order. This stands in contrast to contextualized encoders, such as LSTMs or Transform-

ers (Vaswani et al., 2017). Contextualized encoders are typically trained as unsupervised language models (Peters et al., 2018; Devlin et al., 2019) or on supervised transfer tasks (Conneau et al., 2017; Cer et al., 2018). At the time of writing, weighted averaged word embeddings achieve better results than contextualized encoders on unsupervised STS.[2]

### 2.3 Multi-view word embeddings

Multi-view representation learning is an umbrella term for methods that transform different representations of the same entities into a common space. In NLP, it has typically been applied to word embeddings. A famous example is the cross-lingual projection of word embeddings (Mikolov et al., 2013b; Faruqui and Dyer, 2014). Monolingually, Rastogi et al. (2015) use Generalized Canonical Correlation Analysis (GCCA) to project different word representations into a common space. Yin and Schütze (2016) combine word embeddings by concatenation, truncated Singular Value Decomposition and linear projections; Bollegala and Bao (2018) use autoencoders. Sarma et al. (2018) correlate generic and domain-specific word embeddings by Canonical Correlation Analysis (CCA).

All of these methods are post-training, i.e., they are applied to fully trained word embeddings. MV-DASE falls into the same category, albeit at the sentence level (see Section 3.1). Other methods, which we will call in-training, encourage the alignment of embeddings during training (e.g., Bollegala et al. (2015); Yang et al. (2017)).

### 2.4 Multi-view sentence embeddings

Multi-view sentence embeddings are less frequently explored than multi-view word embeddings. One exception is Tang and de Sa (2019), who train a recurrent neural network and an average word embedding encoder jointly on an unlabeled corpus. This method is in-training, i.e., it cannot be used to combine pre-existing encoders.

Kiela et al. (2018) dynamically integrate an ensemble of word embeddings into a task-specific LSTM. They require labeled data and the resulting embeddings are task-specific.

Sarma et al. (2018) marry domain-adapted word embeddings (see Section 2.3) with InferSent (Conneau et al., 2017), a bidirectional LSTM sentence encoder trained on Stanford Natural Language Inference (SNLI) (Bowman et al., 2015). They initialize InferSent with the adapted embeddings and then retrain it on SNLI. Note that this approach is not feasible when the training regime of an encoder cannot be reproduced, e.g., when the original training data is not publicly available.

[2] http://ixa2.si.ehu.es/stswiki/index.php/STSbenchmark

## 3 Method

We now describe MV-DASE as a general framework. For details on the ensemble used in this paper, see Section 3.2.

### 3.1 Framework

**GCCA basics.** Given zero-mean random vectors $\mathbf{x}_1 \in \mathbb{R}^{d_1}, \mathbf{x}_2 \in \mathbb{R}^{d_2}$, Canonical Correlation Analysis (CCA) finds linear transformations $\boldsymbol{\theta}_1 \in \mathbb{R}^{d_1}, \boldsymbol{\theta}_2 \in \mathbb{R}^{d_2}$ such that $\boldsymbol{\theta}_1^T \mathbf{x}_1$ and $\boldsymbol{\theta}_2^T \mathbf{x}_2$ are maximally correlated. Bach and Jordan (2002) show that CCA reduces to a generalized eigenvalue problem. A generalized eigenvalue problem finds scalar-vector pairs $(\rho, \boldsymbol{\theta})$ that satisfy $\mathbf{A}\boldsymbol{\theta} = \rho\mathbf{B}\boldsymbol{\theta}$ for matrices $\mathbf{A}, \mathbf{B}$. Here, $\mathbf{A}, \mathbf{B}$ are the following block matrices:

$$\begin{bmatrix} \mathbf{0} & \boldsymbol{\Sigma}_{1,2} \\ \boldsymbol{\Sigma}_{2,1} & \mathbf{0} \end{bmatrix} \boldsymbol{\theta} = \rho \begin{bmatrix} \boldsymbol{\Sigma}_{1,1} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_{2,2} \end{bmatrix} \boldsymbol{\theta} \quad (1)$$

where $\boldsymbol{\Sigma}_{1,1}, \boldsymbol{\Sigma}_{2,2}$ are the covariance matrices of $\mathbf{x}_1, \mathbf{x}_2$ and $\boldsymbol{\Sigma}_{1,2}, \boldsymbol{\Sigma}_{2,1}$ are their cross-covariance matrices. We stack all $d$ eigenvectors into an operator $\boldsymbol{\Theta} \in \mathbb{R}^{d \times d_1 + d_2}$. Using this operator, multi-view representations are projected as:

$$\mathbf{x}_{\text{mv}} = \boldsymbol{\Theta} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} \quad (2)$$

Generalized CCA (GCCA) generalizes CCA to three or more random vectors $\mathbf{x}_1 \ldots \mathbf{x}_J$. There are several variants of GCCA (Kettenring, 1971); we follow Bach and Jordan (2002) and solve a multi-view version of Equation 1:

$$\begin{aligned} &\begin{bmatrix} \mathbf{0} & \boldsymbol{\Sigma}_{\ldots} & \boldsymbol{\Sigma}_{1,J} \\ \boldsymbol{\Sigma}_{\ldots} & \mathbf{0} & \boldsymbol{\Sigma}_{\ldots} \\ \boldsymbol{\Sigma}_{J,1} & \boldsymbol{\Sigma}_{\ldots} & \mathbf{0} \end{bmatrix} \boldsymbol{\theta} \\ &= \rho \begin{bmatrix} \boldsymbol{\Sigma}_{1,1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_{\ldots} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\Sigma}_{J,J} \end{bmatrix} \boldsymbol{\theta} \end{aligned} \quad (3)$$

For stability, we add $\tau\sigma_j \mathbf{I}_j$ to every covariance matrix $\boldsymbol{\Sigma}_{j,j}$, where $\tau$ is a hyperparameter (here: $\tau = 0.1$), $\mathbf{I}_j$ is the identity matrix and $\sigma_j$ is the

average variance of $\mathbf{x}_j$. Like in the two-view case, we stack all $d$ eigenvectors into an operator: $\mathbf{\Theta} \in \mathbb{R}^{d \times \sum_j d_j}$.

**GCCA application.** Assume that we have an ensemble of $J$ sentence encoders. The $j$'th encoder is denoted $f_j : \mathbb{S} \rightarrow \mathbb{R}^{d_j}$, where $\mathbb{S}$ is the set of all possible in-domain strings (here: in-domain questions) and $d_j$ is determined by $f_j$. Assume also that we have a sample from $\mathbb{S}$, i.e., a corpus of unlabeled in-domain strings, denoted $S = \{s_1, \ldots, s_N\}$. From this corpus, we create one training matrix $\mathbf{X}_j$ per encoder:

$$\mathbf{X}_j \in \mathbb{R}^{N \times d_j} = \begin{bmatrix} — & f_j(s_1) & — \\ & \vdots & \\ — & f_j(s_N) & — \end{bmatrix} \quad (4)$$

From $\mathbf{X}_j$ we estimate mean vector $\bar{\mathbf{x}}_j \in \mathbb{R}^{d_j}$, covariance matrix $\mathbf{\Sigma}_{j,j} \in \mathbb{R}^{d_j \times d_j}$ and cross-covariance matrices $\mathbf{\Sigma}_{j,j'} \in \mathbb{R}^{d_j \times d_{j'}}$. We then apply GCCA as described before, yielding $\mathbf{\Theta} \in \mathbb{R}^{d \times \sum_j d_j}$. The multi-view embedding of a new input $q$ (e.g., a test query) is:

$$f_{\text{mv}}(q) = \mathbf{\Theta} \begin{bmatrix} f_1(q) - \bar{\mathbf{x}}_1 \\ \vdots \\ f_J(q) - \bar{\mathbf{x}}_J \end{bmatrix} \quad (5)$$

### 3.2 Ensemble

We use MV-DASE on the following ensemble:

- weighted averaged generic GloVe vectors (Pennington et al., 2014)
- weighted averaged domain-specific FastText vectors (Bojanowski et al., 2017)
- Universal Sentence Encoder (USE) (Cer et al., 2018)
- domain-finetuned BERT (Devlin et al., 2019)

In this section, we describe the encoders in detail. Note that the choice of encoders is orthogonal to the framework and other resources could be used. Where possible, we base our selection on the literature: We choose USE over InferSent due to better performance on STS (Perone et al., 2018), and BERT over ELMo (Peters et al., 2018) due to better performance on linguistic probing tasks (Liu et al., 2019a). The choice of GloVe for generic word embeddings is based on Sarma et al. (2018).

**Weighted averaged word embeddings.** We denote generic and domain-specific word embeddings of some word type $i$ as $\mathbf{w}_{G,i} \in \mathbb{R}^{d_G}$ and $\mathbf{w}_{D,i} \in \mathbb{R}^{d_D}$. For $\mathbf{w}_{G,i}$, we use pre-trained 300-d GloVe vectors.[3] $\mathbf{w}_{D,i}$ are trained using skipgram FastText[4] (100-d, default parameters) on the in-domain corpus $S$. We SIF-weight all word embeddings by $a \cdot (a + p(i))^{-1}$, where $p(i)$ is the unigram probability of the word type and the smoothing factor (here: $a = 10^{-3}$) is taken from Arora et al. (2017). We find that probabilities estimated on $S$ produce better results than the Wikipedia-based probabilities provided by Arora et al. (2017) (see Table 2, top), hence this is what we use below. After weighting, we perform top-3 principal component removal on the embedding matrices, which is beneficial for word-level similarity tasks (Mu et al., 2018). We denote the new embeddings of word type $i$ as $\hat{\mathbf{w}}_{G,i}, \hat{\mathbf{w}}_{D,i}$. The embedding of a tokenized string $s = (s_1, \ldots, s_T)$ is computed by averaging:

$$f_G(s) = \frac{1}{T} \sum_{t=1}^{T} \hat{\mathbf{w}}_{G,s_t} \quad f_D(s) = \frac{1}{T} \sum_{t=1}^{T} \hat{\mathbf{w}}_{D,s_t}$$

**Contextualized encoders.** USE and BERT are downloaded as pre-trained models.[5] [6] USE is a Transformer trained on SkipThought (Kiros et al., 2015), conversation response prediction (Henderson et al., 2017) and SNLI. It outputs a single 512-d sentence embedding, which we use as-is. Below, USE is denoted $f_U$.

| | $f_G$ (GloVe) | $f_D$ (FastText) | $f_B$ (BERT) |
|---|---|---|---|
| no SIF | .089 | .083 | .134 |
| wiki SIF | .128 | .100 | .159 |
| in-domain SIF | .147 | .104 | .176 |

| | $f_B$ (BERT) | ELMo |
|---|---|---|
| generic | .138 | .103 |
| domain-finetuned | .176 | .155 |

Table 2: Mean Average Precision (MAP) averaged over heldout forums. Top: MAP as a function of whether and where SIF weights are estimated. Bottom: MAP of generic vs. domain-finetuned BERT and ELMo. Evaluation setup is as described in Section 4, using four heldout forums. Gray: best in column.

[3] nlp.stanford.edu/data/glove.42B.300d.zip

[4] github.com/facebookresearch/fastText

[5] tfhub.dev/google/universal-sentence-encoder-large/3

[6] tfhub.dev/google/bert_uncased_L-12_H-768_A-12/1

BERT is a Transformer that was pre-trained as a masked language model with next sentence prediction. We find that domain-finetuning BERT on $S$ results in improvements over generic BERT (see Table 2, bottom). Note that domain-finetuning refers to unsupervised training as a masked language model, i.e., we only require unlabeled data (Howard and Ruder, 2018). We use default parameters[7] except for a reduced batch size of 8.

At test time, we take the following approach: BERT segments a token sequence $s = (s_1, \ldots, s_T)$ into a subword sequence $s' = ([\text{CLS}], s'_1, \ldots, s'_{T'}, [\text{SEP}])$, where [CLS] and [SEP] are special tokens that were used during pre-training, and $T' \geq T$. BERT produces one 768-d vector $\mathbf{v}_{l,t}$ per subword $s'_t$ and layer $l \in [1, \ldots, L]$, where $L$ is the total number of layers (here: 12). We SIF-weight all vectors according to the probability of their subword (estimated on $S$) and average over layers and subwords, excluding the special tokens:

$$f_B(s) = \frac{1}{T' \cdot L} \sum_{t=1}^{T'} \sum_{l=1}^{L} \frac{a}{a + p(s'_t)} \mathbf{v}_{l,t}$$

## 4 Evaluation on Stack Exchange

### 4.1 Data

**Corpora.** We evaluate MV-DASE on the CQADupStack corpus (Hoogeveen et al., 2015), which is based on a 2014 Stack Exchange dump. CQADupStack contains 12 forums that have enough duplicates for supervised training; as a consequence, it may not be representative of low-resource domains. We therefore supplement it with 12 low-resource forums from the 2018-12-20 Stack Exchange dump.[8] For our purposes, low-resource means a forum with 100–200 duplicates, which we consider sufficient for evaluation but insufficient for supervised training. All duplicates in the datasets were labeled by unpaid community members. As a result, false negatives (i.e., unflagged duplicates) are common in the gold standard (Hoogeveen et al., 2016). While we do not explicitly filter for language, the vast majority of the data is in English.

[7] github.com/google-research/bert/blob/master/run_pretraining.py

[8] Preprocessed low-resource data can be downloaded here: github.com/npoe/lowresourcecqa

| | | forum | #Q | #D | #T |
|---|---|---|---|---|---|
| CQADupStack forums | and | android | 23697 | 1579 | 2.4M |
| | eng | english | 41791 | 3506 | 3.4M |
| | gam | gaming | 46896 | 2207 | 4.0M |
| | gis | gis | 38522 | 1099 | 4.6M |
| | mat | mathematica | 17509 | 1271 | 2.6M |
| | phy | physics | 39355 | 1769 | 6.1M |
| | prg | programmers | 33052 | 1538 | 5.6M |
| | sta | stats | 42921 | 890 | 7.2M |
| | tex | tex | 71090 | 4939 | 7.4M |
| | uni | unix | 48454 | 1648 | 5.5M |
| | web | webmasters | 17911 | 1143 | 2.0M |
| | wor | wordpress | 49146 | 719 | 5.6M |
| low-resource forums | bud | buddhism | 5350 | 120 | 670K |
| | che | chess | 4539 | 154 | 500K |
| | cog | cogsci | 5687 | 126 | 800K |
| | law | law | 11059 | 126 | 1.7M |
| | net | networkengineering | 11386 | 154 | 1.5M |
| | out | outdoors | 4651 | 124 | 580K |
| | pro | productivity | 2508 | 127 | 380K |
| | rev | reverseengineering | 15619 | 119 | 790K |
| | sit | sitecore | 5605 | 130 | 680K |
| | spo | sports | 4531 | 127 | 430K |
| | sqa | sqa | 8360 | 166 | 950K |
| | win | windowsphone | 3490 | 192 | 290K |

Table 3: Forum statistics. #Q: total number of questions, #D: number of labeled duplicates, #T: number of tokens in training set $S$. Gray: heldout forums.

**Data split.** We split every forum into a test and training set, such that the test set contains all duplicates and the training set contains the remaining unlabeled questions.[9] The unlabeled training set is used for FastText training, BERT domain-finetuning, SIF weight estimation and GCCA. Test queries are never seen during training, not even in an unsupervised way. For hyperparameter choices, we hold out two high-resource and two low-resource forums (highlighted in Table 3). They are not used for the final evaluation and significance tests.

**Preprocessing.** Every question object consists of a title (average length 9 words), a body (average length 125 words), any number of answers or comments, and metadata (e.g., upvotes, view counts). We preprocess the data with the CQADupStack package.[10] To calculate question representations, we use the concatenation of question title and body. We always ignore answers, comments and metadata, as this information is not usually available at the time a question is posted.

[9] We do not use the official CQADupStack train / test split, as it is meant for supervised training and has comparatively few duplicates per test set. Since MV-DASE is unsupervised, we can afford a more robust evaluation on all labeled duplicates.

[10] github.com/D1Doris/CQADupStack

| | | heldout | | test forums | | | | | | | | | | MAP | AUC(.05) | NDCG | P@3 | R@3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | and | eng | gam | gis | mat | prg | phy | sta | tex | uni | web | wor | average over test forums | | | | |
| CQADupStack forums | 1 BM25 | .175 | .162 | .310 | .264 | .132 | .119 | .216 | .212 | .116 | .171 | .103 | .171 | .181 | .821 | .314 | .067 | .196 |
| | 2 $f_G$ (GloVe) | .121 | .093 | .202 | .148 | .056 | .084 | .152 | .153 | .063 | .120 | .085 | .093 | .115 | .755 | .233 | .042 | .123 |
| | 3 $f_D$ (FastText) | .123 | .083 | .211 | .169 | .079 | .091 | .172 | .175 | .085 | .136 | .084 | .107 | .131 | .817 | .261 | .047 | .138 |
| | 4 $f_U$ (USE) | .183 | .113 | .347 | .156 | .081 | .146 | .195 | .165 | .071 | .142 | .110 | .117 | .153 | .832 | .285 | .056 | .163 |
| | 5 $f_B$ (BERT) | .141 | .129 | .262 | .196 | .103 | .099 | .190 | .179 | .090 | .135 | .109 | .134 | .150 | .805 | .276 | .055 | .159 |
| | 6 MV-DASE | .211 | .177 | .371 | .274 | .149 | .181 | .259 | .236 | .135 | .206 | .145 | .183 | .214 | .904 | .362 | .080 | .232 |
| | 7 InferSent | .069 | .047 | .145 | .123 | .041 | .041 | .105 | .121 | .042 | .078 | .053 | .072 | .082 | .667 | .182 | .029 | .085 |
| | 8 doc2vec | .102 | .057 | .141 | .150 | .064 | .069 | .138 | .170 | .067 | .125 | .083 | .111 | .112 | .799 | .234 | .040 | .116 |
| | 9 ELMo | .141 | .116 | .251 | .179 | .081 | .097 | .184 | .182 | .087 | .147 | .097 | .117 | .142 | .835 | .274 | .051 | .149 |
| | 10 word-level CCA | .149 | .109 | .253 | .202 | .101 | .111 | .190 | .189 | .096 | .156 | .103 | .125 | .153 | .851 | .290 | .055 | .161 |
| | 11 upper bound | 1.00 | | | | | | | | | | | | | | | .351 | .999 |
| | | bud | che | cog | law | net | out | pro | rev | sit | spo | sqa | win | average over test forums | | | | |
| low-resource forums | 12 BM25 | .276 | .195 | .269 | .345 | .167 | .373 | .196 | .186 | .430 | .465 | .275 | .349 | .306 | .842 | .461 | .116 | .345 |
| | 13 $f_G$ (GloVe) | .249 | .125 | .209 | .312 | .103 | .260 | .110 | .134 | .237 | .363 | .166 | .239 | .213 | .781 | .359 | .079 | .234 |
| | 14 $f_D$ (FastText) | .142 | .064 | .132 | .255 | .111 | .168 | .067 | .101 | .243 | .239 | .173 | .136 | .163 | .767 | .314 | .060 | .180 |
| | 15 $f_U$ (USE) | .332 | .247 | .384 | .458 | .152 | .513 | .214 | .144 | .282 | .448 | .221 | .244 | .306 | .880 | .470 | .119 | .352 |
| | 16 $f_B$ (BERT) | .261 | .173 | .221 | .335 | .137 | .348 | .171 | .143 | .324 | .489 | .194 | .257 | .262 | .812 | .411 | .099 | .294 |
| | 17 MV-DASE | .378 | .259 | .384 | .447 | .184 | .495 | .233 | .241 | .427 | .523 | .289 | .352 | .358 | .924 | .524 | .137 | .407 |
| | 18 InferSent | .154 | .073 | .117 | .236 | .079 | .194 | .089 | .078 | .192 | .312 | .123 | .161 | .158 | .701 | .281 | .054 | .162 |
| | 19 doc2vec | .133 | .058 | .117 | .239 | .146 | .145 | .057 | .080 | .192 | .141 | .140 | .090 | .135 | .759 | .279 | .048 | .143 |
| | 20 ELMo | .222 | .140 | .228 | .332 | .137 | .248 | .136 | .092 | .247 | .433 | .171 | .278 | .230 | .837 | .387 | .084 | .252 |
| | 21 word-level CCA | .260 | .142 | .237 | .325 | .146 | .274 | .111 | .186 | .312 | .327 | .218 | .194 | .233 | .844 | .391 | .086 | .254 |
| | 22 ADA | .229 | .164 | .161 | .250 | .132 | .207 | .117 | .147 | .225 | .299 | .193 | .218 | .195 | .823 | .347 | .068 | .201 |
| | 23 upper bound | 1.00 | | | | | | | | | | | | | | | .341 | .999 |

Table 4: Main results. Left: MAP on individual forums (heldout and test forums). Rightmost five columns: all metrics averaged over test forums (excluding heldout forums). Gray: best in column.

### 4.2 Evaluation and Metrics

Given a test query $q$, we rank all candidates $c \neq q$ from the same forum by $\cos(f(q), f(c))$, where $f$ is an encoder (e.g., MV-DASE). Our metrics are MAP, AUC(.05), Normalized Discounted Cumulative Gain (NDCG), Recall@3 (R@3) and Precision@3 (P@3). AUC(.05), the area under the ROC curve up to a false positive rate of .05, is used by Shah et al. (2018). Note that upper bounds on P@3 and R@3 are not 1, since most duplicates have only one original and a few have more than three.

### 4.3 Baselines

**Unsupervised.** Our IR baseline is BM25 (Robertson et al., 1995) as implemented in Elasticsearch 6.5.4 (Gormley and Tong, 2015) with default parameters. We test against all single-view encoders from our ensemble. The remaining unsupervised baselines are:

- ELMo (Peters et al., 2018).[11] We treat ELMo like BERT (Section 3.2), i.e., we finetune[12] the language model on the in-domain corpus (3 epochs, batch size 8), SIF-weight all vectors according to in-domain word probability and then average over layers and tokens.
- Doc2vec (Le and Mikolov, 2014) trained on the in-domain corpus, using the best DQD hyperparameters reported in Lau and Baldwin (2016).
- InferSent V.1.[13] (Conneau et al., 2017)
- Our re-implementation of domain-adapted CCA word embeddings (Sarma et al. (2018), see Section 2.3). We use the same word embeddings, SIF weights and component removal described in Section 3.2. Denoted "word-level CCA" below.

[11] tfhub.dev/google/elmo/2

[12] github.com/allenai/bilm-tf/blob/master/bin/restart.py

[13] github.com/facebookresearch/InferSent

| | MAP | AUC(.05) |
|---|---|---|
| 1 | MV-DASE, $\neg f_G, \neg f_D$ | MV-DASE |
| 2 | $\neg f_B, \neg(f_G, f_D)$ | $\neg f_B$, $\neg f_D$, $\neg f_G$, $\neg(f_G, f_D)$ |
| 3 | BM25, avg, concat, ADA, word-level CCA, ELMo, $\neg f_U$, $f_D$, $f_G$, $f_B$, $\neg(f_B, f_U)$, $f_U$ | BM25, avg, concat, doc2vec, ADA, word-level CCA, ELMo, $f_B$, $f_D$, $f_G$, $\neg f_U$, $f_U$, $\neg(f_B, f_U)$ |
| 4 | InferSent, doc2vec | InferSent |

Table 5: Group rankings by transitive closure of paired t-tests. $\neg f_j$ is MV-DASE without $f_j$ (see Table 6). No particular order inside groups.

| | | | MAP | AUC(.05) | NDCG | P@3 | R@3 |
|---|---|---|---|---|---|---|---|
| CQADupStack | 1 | $\neg f_G$ | .002 | .000 | .000 | .000 | -.001 |
| | 2 | $\neg f_D$ | -.002 | -.008 | -.004 | -.001 | -.002 |
| | 3 | $\neg f_U$ | -.030 | -.032 | -.037 | -.012 | -.035 |
| | 4 | $\neg f_B$ | -.010 | -.006 | -.011 | -.003 | -.008 |
| | 5 | $\neg(f_U, f_B)$ | -.056 | -.042 | -.066 | -.022 | -.065 |
| | 6 | $\neg(f_G, f_D)$ | -.012 | -.016 | -.017 | -.005 | -.015 |
| | 7 | concat | -.042 | -.071 | -.058 | -.017 | -.047 |
| | 8 | avg | -.046 | -.075 | -.064 | -.018 | -.051 |
| low-resource | 9 | $\neg f_G$ | -.008 | -.005 | -.010 | -.006 | -.016 |
| | 10 | $\neg f_D$ | .007 | -.001 | .012 | .002 | .006 |
| | 11 | $\neg f_U$ | -.058 | -.050 | -.063 | -.023 | -.067 |
| | 12 | $\neg f_B$ | -.017 | -.006 | -.014 | -.010 | -.028 |
| | 13 | $\neg(f_U, f_B)$ | -.120 | -.069 | -.130 | -.054 | -.160 |
| | 14 | $\neg(f_G, f_D)$ | -.010 | -.005 | -.007 | -.005 | -.014 |
| | 15 | concat | -.058 | -.078 | -.070 | -.023 | -.069 |
| | 16 | avg | -.067 | -.081 | -.080 | -.028 | -.082 |

Table 6: Ablation study. Deltas relative to MV-DASE. Metrics were averaged over test forums before calculating deltas. concat/avg are naive view concatenation and averaging. Gray: better than MV-DASE.

**ADA.** We evaluate the supervised domain-adversarial method of Shah et al. (2018) (ADA) on the low-resource forums. Recall that ADA requires a related labeled source domain. To achieve this, we pair every low-resource forum (target) with the CQADupStack forum (source) with which it has the highest word trigram overlap. See supplementary material for more details and a table of all source-target mappings.[14]

### 4.4 Ablation studies

We perform a set of experiments where we omit views from the ensemble. We also replace GCCA with naive view concatenation or view averaging. When averaging, we pad lower-dimensional vectors (Coates and Bollegala, 2018).

### 4.5 Significance tests

We perform paired t-tests, using the 20 test set forums as data points.[15] We then find groups of equivalent methods by transitive closure of $a \sim b \equiv p \geq .05$. Group $A$ being ranked higher than group $B$ means that every method in $A$ performs significantly better than every method in $B$. Two methods in the same group may differ significantly, but there exists a chain between them of methods with insignificant differences.

## 5 Discussion

### 5.1 Comparison with baselines

**BM25.** BM25 is a tough baseline for DQD: In terms of MAP, it is better than or comparable to every single view (see Table 5). MV-DASE on the other hand, which is built from the same views, outperforms BM25 significantly and almost consistently (19 out of 20 test forums), regardless of the metric. This underlines the usefulness of our multi-view approach.

**Single views.** MV-DASE outperforms the views that make up its ensemble significantly and almost consistently. There are two exceptions (out of 20 test forums): On *law* and *outdoors*, $f_U$ (USE) performs slightly better on its own (Table 4, row 15). Since these forums are less "technical" than most, we hypothesize that they may be less in need of domain adaptation.

**Word-level CCA.** The word-level CCA baseline by Sarma et al. (2018) outperforms $f_G$ and $f_D$ on their own (see Table 4, rows 10, 21), which validates the approach. The method is directly comparable to MV-DASE$\neg(f_U, f_B)$, i.e., MV-DASE on generic and domain-specific averaged word embeddings (see Table 6). The main differences between them are (a) the order in which CCA and averaging are performed and (b) whether the CCA "vocabulary" is composed of word types or sentences. Note that in contrast to MV-DASE, word-level CCA is incompatible with contextualized embeddings, since it requires a context-independent one-to-one mapping between word types and vectors.

[14] We also experiment with Multinomial Adversarial Networks (MAN) (Chen and Cardie, 2018), a multi-source multi-target framework that can be trained on all 24 forums jointly. Initial results were not competitive with ADA, so we do not include them here. See supplementary material for details.

[15] Ten forums for t-tests involving ADA.

**ADA.** Supervised domain-adversarial ADA performs significantly worse than unsupervised MV-DASE (see Table 5). It is comparable to BM25 in terms of AUC(.05) (the metric used by Shah et al. (2018)), but not in terms of MAP.

Recall that we restricted the choice of source domains to the 12 CQADupStack forums. As a consequence, some target forums were paired with non-ideal source forums (e.g., *english→buddhism*). It is possible that the baseline would have performed better with a wider choice of source domains. Nonetheless, this observation highlights a key advantage of our approach: It does not depend on the availability of a label-rich related source domain (or indeed, any labels at all).

**Other baselines.** InferSent performs poorly on the DQD task, which is surprising given its similarity to USE. Recall that InferSent and USE are pre-trained on sentence-level SNLI, but that the training regime of USE also contains conversation response prediction. So USE is expected to be better equipped to handle (a) multi-sentence documents and (b) forum-style language.

Doc2vec is trained on the same data as $f_D$, but performs significantly worse. The difference between them may be due to the ability of FastText to exploit orthography. Domain-finetuned ELMo performs comparably to domain-finetuned BERT on some forums but not consistently.

### 5.2 Ablation study

**View ablation.** On the low-resource forums, omitting $f_D$ has a beneficial effect (Table 6, row 10). This suggests that the in-domain FastText embeddings have insufficient quality when learned on the smallest forums and / or that domain-finetuned BERT subsumes any positive effect. On the high-resource CQADupStack forums, domain-specific embeddings contribute positively, while generic GloVe does not (rows 1,2). Table 5 shows that omitting either $f_G$ or $f_D$ from the ensemble does not lead to a significant drop in MAP, but omitting both does.

USE has the biggest positive effect on MV-DASE (Table 6, rows 3,11), also evidenced by the fact that omitting it is significantly more harmful than omitting any other single view (Table 5). Recall from Section 3.2 that USE is trained on supervised transfer tasks, while the remaining encoders are fully unsupervised.

**GCCA ablation.** The naive concatenation or averaging of views is significantly less effective than view correlation by GCCA (Table 6, rows 7,8,15,16, and Table 5). This underlines that multi-view learning is not just about *which* views are combined, but also about *how*. Intuitively, GCCA discovers which features from the different encoders "mean the same thing" in the domain. By contrast, concatenation treats views as orthogonal, while averaging mixes them in an unstructured way.

## 6 Evaluation on SemEval-2017 3B

In this section, we evaluate MV-DASE on SemEval-2017 3B, a DQD shared task based on the QatarLiving CQA forum. The benchmark provides manually labeled question pairs for training as well as additional unlabeled in-domain data. Since MV-DASE is unsupervised, we discard all training labels and concatenate training and unlabeled data into a text corpus ($\approx$ 1.5M tokens). This corpus is used for FastText training, BERT domain-finetuning, SIF weight estimation and GCCA, as described in Section 3.

The test set contains 88 queries $q$ with ten candidates $c_1 \dots c_{10}$ each. We preprocess all data with the CQADupStack package and concatenate question subjects and bodies, before encoding them. We rank candidates by $\cos(f(q), f(c))$ and evaluate the result with the official shared task scorer.[16] In keeping with the original leaderboard, we report MAP and MRR (Mean Reciprocal Rank). We compare against previous literature as well as all individual views, view concatenation and averaging. See Table 7 for results. Like we observed on the Stack Exchange data, MV-DASE outperforms its individual views, their concatenation and average. It beats the previous State of the Art (a supervised system) by a margin of 2.5% MAP.

## 7 Evaluation on unsupervised STS

While this paper focuses on Duplicate Question Detection, MV-DASE is also applicable to other unsupervised sentence-pair tasks. As proof of concept, we test it on the unsupervised STS Benchmark (Cer et al., 2017). Here, the task is to predict similarity scores $y \in \mathbb{R}$ for sentence pairs $(s_1, s_2)$.

[16] alt.qcri.org/semeval2017/task3/data/uploads/semeval2017_task3_submissions_and_scores.zip

| | | MAP | MRR |
|---|---|---|---|
| 1 | $f_G$ (GloVe) | 43.13 | 47.39 |
| 2 | $f_D$ (FastText) | 43.38 | 47.67 |
| 3 | $f_U$ (USE) | 48.22 | 52.73 |
| 4 | $f_B$ (BERT) | 43.51 | 48.52 |
| 5 | MV-DASE | 51.56 | 56.48 |
| 6 | concat | 44.66 | 49.84 |
| 7 | avg | 44.95 | 49.76 |
| 8 | Filice et al. (2017)* | 49.00 | 52.41 |
| 9 | Charlet and Damnati (2017)* | 47.87 | 50.97 |
| 10 | Goyal (2017)* | 47.20 | 53.22 |
| 11 | Zhang and Wu (2018) | 48.53 | 52.75 |
| 12 | Yang et al. (2018) | 48.97 | - |
| 13 | Gonzalez et al. (2018) | 48.56 | 52.41 |
| 14 | IR baseline* | 41.85 | 46.42 |
| 15 | Random baseline* | 29.81 | 33.02 |

Table 7: MAP and MRR (percentages) on SemEval-2017 3B test set. *Shared task top teams (best run out of three) and baselines as reported in Nakov et al. (2017), Table 6. Gray: best in column.

We treat the benchmark training set as an unlabeled corpus, i.e., we discard all labels and destroy the original sentence pairings by shuffling. The resulting corpus is used for BERT domain-finetuning, SIF weight estimation and GCCA. At test time, we measure Pearson's $r$ between $\cos(f(s_1), f(s_2))$ and $y$, where $f$ is an encoder (e.g., MV-DASE) and $y$ is the ground truth similarity of test set pair $(s_1, s_2)$.

In this experiment, the ensemble contains USE ($f_U$), domain-finetuned BERT ($f_B$) and $f_G$. For $f_G$, we either use SIF-weighted averaged GloVe vectors (Section 3.2), or unweighted averaged ParaNMT[17] word and trigram vectors (Wieting and Gimpel, 2018), which are the current State of the Art on the unsupervised STS Benchmark test set (Ethayarajh, 2018). The unlabeled training set is very small (64K tokens); hence, we do not include $f_D$ in the ensemble, and we finetune the BERT language model for 10K rather than 100K steps to avoid overfitting. Like on the DQD tasks, MV-DASE beats its individual views as well as naive view concatenation and averaging (see Table 8). After adding ParaNMT to the ensemble, we achieve competitive results.

## 8 Future Work

**Non-Linear GCCA.** In Section 3.1, we assumed that relationships between representations are linear. This is probably reasonable for word embeddings (most cross-lingual word embeddings are linear projections, e.g. Artetxe et al. (2018)), but it is unclear whether it holds for sentence embeddings. Potential avenues for non-linear GCCA include Kernel GCCA (Tenenhaus et al., 2015) and Deep GCCA (Benton et al., 2017).

[17] github.com/jwieting/para-nmt-50m

| | | $f_G$ = GloVe | $f_G$ = ParaNMT |
|---|---|---|---|
| 1 | $f_G$ | .731 / .647 | .817 / .799 |
| 2 | $f_U$ (USE) | .793 / .762 | .793 / .762 |
| 3 | $f_B$ (BERT) | .779 / .718 | .779 / .718 |
| 4 | MV-DASE | .825 / .771 | .842 / .804 |
| 5 | concat | .791 / .730 | .826 / .772 |
| 6 | avg | .790 / .729 | .823 / .771 |

Table 8: Pearson's $r$ (dev / test) on the unsupervised STS Benchmark, using different embeddings for $f_G$. Gray: best in column. Underlined: current unsupervised SoTA on test set (Wieting and Gimpel, 2018).

**More views.** A major advantage of MV-DASE is that it is agnostic to the number and specifics of its views. We plan to investigate whether additional or different views (e.g., encoders learned on related domains) can increase performance.

## 9 Conclusion

We have presented a multi-view approach to unsupervised Duplicate Question Detection in low-resource, domain-specific Community Question Answering forums. MV-DASE is a multi-view sentence embedding framework based on Generalized Canonical Correlation Analysis. It combines domain-specific and generic weighted averaged word embeddings with domain-finetuned BERT and the Universal Sentence Encoder, using unlabeled in-domain data only.

Experiments on the CQADupStack corpus and additional low-resource forums show significant improvements over BM25 and all single-view baselines. MV-DASE sets a new State of the Art on a recent DQD shared task (SemEval-2017 3B), with a 2.5% MAP improvement over the best supervised system. Finally, an experiment on the STS Benchmark suggests that MV-DASE has potential on other unsupervised sentence-pair tasks.

## Acknowledgements

We thank Bernt Andrassy and Pankaj Gupta at Siemens MIC-DE, as well as our anonymous reviewers, for their helpful comments. This research was funded by Siemens AG.

## References


Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *ICLR*, Toulon, France.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018. A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. In *ACL*, pages 789–798, Melbourne, Australia.

Francis R Bach and Michael I Jordan. 2002. Kernel independent component analysis. *JMLR*, 3:1–48.

Timothy Baldwin, Huizhi Liang, Bahar Salehi, Doris Hoogeveen, Yitong Li, and Long Duong. 2016. UniMelb at SemEval-2016 Task 3: Identifying similar questions by combining a CNN with string similarity measures. In *International Workshop on Semantic Evaluation*, pages 851–856, San Diego, USA.

Adrian Benton, Huda Khayrallah, Biman Gujral, Dee Ann Reisinger, Sheng Zhang, and Raman Arora. 2017. Deep generalized canonical correlation analysis. *arXiv preprint arXiv:1702.02519*.

Dasha Bogdanova, Cicero dos Santos, Luciano Barbosa, and Bianca Zadrozny. 2015. Detecting semantically equivalent questions in online user forums. In *CoNLL*, pages 123–131, Beijing, China.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *TACL*, 5:135–146.

Danushka Bollegala and Cong Bao. 2018. Learning word meta-embeddings by autoencoding. In *COLING*, pages 1650–1661, Santa Fe, USA.

Danushka Bollegala, Takanori Maehara, and Ken-ichi Kawarabayashi. 2015. Unsupervised cross-domain word representation learning. In *ACL*, pages 730–740, Beijing, China.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*, pages 632–642, Lisbon, Portugal.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *International Workshop on Semantic Evaluation*, pages 1–14, Vancouver, Canada.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal Sentence Encoder for English. In *EMNLP*, pages 169–174, Brussels, Belgium.

Delphine Charlet and Geraldine Damnati. 2017. SimBow at SemEval-2017 Task 3: Soft-cosine semantic similarity between questions for community question answering. In *International Workshop on Semantic Evaluation*, pages 315–319, Vancouver, Canada.

Xilun Chen and Claire Cardie. 2018. Multinomial adversarial networks for multi-domain text classification. In *NAACL-HLT*, pages 1226–1240, New Orleans, USA.

Joshua Coates and Danushka Bollegala. 2018. Frustratingly easy meta-embedding – computing meta-embeddings by averaging source word embeddings. In *NAACL-HLT*, pages 194–198, New Orleans, USA.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*, pages 670–680, Copenhagen, Denmark.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186, Minneapolis, USA.

Cicero Dos Santos, Luciano Barbosa, Dasha Bogdanova, and Bianca Zadrozny. 2015. Learning hybrid representations to retrieve semantically equivalent questions. In *ACL*, pages 694–699, Beijing, China.

Kawin Ethayarajh. 2018. Unsupervised random walk sentence embeddings: A strong but simple baseline. In *Workshop on Representation Learning for NLP*, pages 91–100, Melbourne, Australia.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *EACL*, pages 462–471, Gothenburg, Sweden.

Simone Filice, Giovanni Da San Martino, Alessandro Moschitti, and Roberto Basili. 2017. KeLP at SemEval-2017 Task 3: Learning pairwise patterns in community question answering. In *International Workshop on Semantic Evaluation*, pages 326–333, Vancouver, Canada.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *JMLR*, 17:2096–2030.

Ana Gonzalez, Isabelle Augenstein, and Anders Søgaard. 2018. A strong baseline for question relevancy ranking. In *EMNLP*, pages 4810–4815, Brussels, Belgium.

Clinton Gormley and Zachary Tong. 2015. *Elasticsearch: The definitive guide: A distributed real-time search and analytics engine*. O'Reilly Media.

Naman Goyal. 2017. LearningToQuestion at semeval 2017 Task 3: Ranking similar questions by learning to rank using rich features. In *International Workshop on Semantic Evaluation*, pages 326–333, Vancouver, Canada.

Matthew Henderson, Rami Al-Rfou, Brian Strope, Yun-hsuan Sung, Laszlo Lukacs, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. 2017. Efficient natural language response suggestion for smart reply. *arXiv preprint arXiv:1705.00652*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Doris Hoogeveen, Andrew Bennett, Yitong Li, Karin M Verspoor, and Timothy Baldwin. 2018. Detecting misflagged duplicate questions in community question-answering archives. In *ICWSM*, Stanford, USA.

Doris Hoogeveen, Karin M Verspoor, and Timothy Baldwin. 2015. CQADupStack: A benchmark data set for community question-answering research. In *Australasian Document Computing Symposium*, Parramatta, Australia.

Doris Hoogeveen, Karin M Verspoor, and Timothy Baldwin. 2016. CQADupStack: Gold or silver. In *Workshop on Web Question Answering Beyond Factoids*, volume 16, Pisa, Italy.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *ACL*, pages 328–339, Melbourne, Australia.

Jon R Kettenring. 1971. Canonical analysis of several sets of variables. *Biometrika*, 58(3):433–451.

Douwe Kiela, Changhan Wang, and Kyunghyun Cho. 2018. Dynamic meta-embeddings for improved sentence representations. In *EMNLP*, pages 1466–1477, Brussels, Belgium.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *NeurIPS*, pages 3294–3302, Montreal, Canada.

Jey Hand Lau and Timothy Baldwin. 2016. An empirical evaluation of doc2vec with practical insights into document embedding generation. In *Workshop on Representation Learning for NLP*, pages 78–86, Berlin, Germany.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*, pages 1188–1196, Beijing, China.

Nelson F Liu, Matt Gardner, Yonatan Belinkov, Matthew Peters, and Noah A Smith. 2019a. Linguistic knowledge and transferability of contextual representations. In *NAACL-HLT*, pages 1073–1094, Minneapolis, USA.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Jiaqi Mu, Suma Bhat, and Pramod Viswanath. 2018. All-but-the-top: Simple and effective postprocessing for word representations. In *ICLR*, Vancouver, Canada.

Preslav Nakov, Doris Hoogeveen, Lluıs Marquez, Allessandro Moschitti, Hamdy Mubarak, Timothy Baldwin, and Karin Verspoor. 2017. SemEval-2017 Task 3: Community question answering. In *International Workshop on Semantic Evaluation*, pages 27–48, Vancouver, Canada.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *EMNLP*, pages 1532–1543, Doha, Qatar.

Christian S Perone, Roberto Silveira, and Thomas S Paula. 2018. Evaluation of sentence embeddings in downstream and linguistic probing tasks. *arXiv preprint arXiv:1806.06259*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*, pages 2227–2237, New Orleans, USA.

Pushpendre Rastogi, Benjamin Van Durme, and Raman Arora. 2015. Multiview LSA: Representation learning via generalized CCA. In *NAACL-HLT*, pages 556–566, Denver, USA.

Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. 1995. Okapi at TREC-3. *Text REtrieval Conference (TREC)*, pages 109–126.

João António Rodrigues, Chakaveh Saedi, Vladislav Maraev, Joao Silva, and António Branco. 2017. Ways of asking and replying in duplicate question detection. In *Joint Conference on Lexical and Computational Semantics*, pages 262–270, Vancouver, Canada.

Prathusha K Sarma, Yingyu Liang, and William A Sethares. 2018. Domain adapted word embeddings for improved sentiment classification. In *Workshop on Deep Learning Approaches for Low-Resource NLP*, pages 51–59, Melbourne, Australia.

Darsh Shah, Tao Lei, Alessandro Moschitti, Salvatore Romeo, and Preslav Nakov. 2018. Adversarial domain adaptation for duplicate question detection. In *EMNLP*, pages 1056–1063, Brussels, Belgium.

Shuai Tang and Virginia R de Sa. 2019. Improving sentence representations with multi-view frameworks. In *Interpretability and Robustness for Audio, Speech and Language Workshop*, Montreal, Canada.

Arthur Tenenhaus, Cathy Philippe, and Vincent Frouin. 2015. Kernel generalized canonical correlation analysis. *Computational Statistics & Data Analysis*, 90:114–131.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*, pages 5998–6008, Long Beach, USA.

John Wieting and Kevin Gimpel. 2018. ParaNMT-50M: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. In *ACL*, pages 451–462, Melbourne, Australia.

Wei Yang, Wei Lu, and Vincent W Zheng. 2017. A simple regularization-based algorithm for learning cross-domain word embeddings. In *EMNLP*, pages 2898–2904, Copenhagen, Denmark.

Ziyi Yang, Chenguang Zhu, and Weizhu Chen. 2018. Zero-training sentence embedding via orthogonal basis. *arXiv preprint arXiv:1810.00438*.

Wenpeng Yin and Hinrich Schütze. 2016. Learning word meta-embeddings. In *ACL*, pages 1351–1360, Berlin, Germany.

Mingua Zhang and Yunfang Wu. 2018. An unsupervised model with attention autoencoders for question retrieval. In *AAAI*, pages 4978–4986, New Orleans, USA.

Wei Emma Zhang, Quan Z Sheng, Jey Han Lau, and Ermyas Abebe. 2017. Detecting duplicate posts in programming QA communities via latent semantics and association rules. In *WWW*, pages 1221–1229.

# Multi-label Categorization of Accounts of Sexism using a Neural Framework

**Pulkit Parikh**[1], **Harika Abburi**[1], **Pinkesh Badjatiya**[1], **Radhika Krishnan**[1], **Niyati Chhaya**[1,2], **Manish Gupta**[1*], **Vasudeva Varma**[1]
[1]IIIT-Hyderabad, India
[2]Adobe Research, India
{pulkit.parikh,harika.a,pinkesh.badjatiya}@research.iiit.ac.in,
radhika.krishnan@iiit.ac.in, nchhaya@adobe.com, {manish.gupta,vv}@iiit.ac.in

## Abstract

Sexism, an injustice that subjects women and girls to enormous suffering, manifests in blatant as well as subtle ways. In the wake of growing documentation of experiences of sexism on the web, the automatic categorization of accounts of sexism has the potential to assist social scientists and policy makers in utilizing such data to study and counter sexism better. The existing work on sexism classification, which is different from sexism detection, has certain limitations in terms of the categories of sexism used and/or whether they can co-occur. To the best of our knowledge, this is the first work on the multi-label classification of sexism of any kind(s), and we contribute the largest dataset for sexism categorization. We develop a neural solution for this multi-label classification that can combine sentence representations obtained using models such as BERT with distributional and linguistic word embeddings using a flexible, hierarchical architecture involving recurrent components and optional convolutional ones. Further, we leverage unlabeled accounts of sexism to infuse domain-specific elements into our framework. The best proposed method outperforms several deep learning as well as traditional machine learning baselines by an appreciable margin.

## 1 Introduction

Sexism, discrimination on the basis of one's sex, prevails in our society in numerous forms, causing immense suffering to women and girls. Online forums have enabled victims of sexism to share their experiences freely and widely by facilitating anonymity and connecting far-away people. A meaningful categorization of these accounts of sexism can play a part in analyzing sexism with a view to developing sensitization programs, systemic safeguards, and other mechanisms against this injustice. Given the rising volume of such information on digital media, automated sexism categorization can aid social scientists and policy makers in combating sexism by conducting such analyses efficiently.

While sexism is detected as a category of hate in some of the hate speech classification work (Badjatiya et al., 2017; Waseem and Hovy, 2016), it does not perform sexism classification. Except the work on categorizing sexual harassment by Karlekar and Bansal (2018), the prior work on classifying sexism assumes the categories to be mutually exclusive (Anzovino et al., 2018; Jha and Mamidi, 2017). Moreover, the existing category sets number between 2 to 5. In this paper, we focus on the new problem of the multi-label categorization of an account of sexism reporting any type(s) of sexism. We create a dataset comprising 13023 accounts of sexism, including first-person accounts from survivors, each tagged with at least one of 23 categories of sexism. The categories were defined keeping in mind the discourse and campaigns on gender-related issues along with potential policy implications, under the guidance of a social scientist. Ten annotators, most of whom have formally studied topics related to gender and/or sexuality, were recruited to label textual accounts of sexism. The accounts are drawn from the Everyday Sexism Project website[1], where voluntary contributors from all over the world document experiences of sexism suffered or witnessed by them. For classification experiments, the categories found in less than 400 accounts in our dataset are appropriately merged with others, resulting in 14 categories.

The rationale for formulating this classification as multi-label is that many experiences inherently involve multiple types of sexism. For instance, *"I*

*The author is also an applied researcher at Microsoft.

[1]https://everydaysexism.com



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1642–1652,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

*overheard a co-worker saying that I should be in more team events and photos because I am pleasing to the eye! Disgusting.*" is an experience of sexism wherein the victim was subjected to three types of sexism, namely hyper-sexualization, sexual harassment, and hostile work environment.

We develop a novel neural architecture for the multi-label classification of accounts of sexism that enables flexibly combining sentence representations created using models such as BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2018) with distributional word embeddings like ELMo (Embeddings from Language Models) (Peters et al., 2018) and Global Vectors (GloVe) (Pennington et al., 2014) and a linguistic feature representation through hierarchical convolutional and/or recurrent operations. Leveraging general-purpose models such as BERT for encoding sentences likely makes our model better equipped to capture semantic aspects effectively, since they are trained on substantially larger textual data than the domain-specific labeled data that we have. Moreover, we adapt a BERT model for the domain of instances of sexism using unlabeled data. Embeddings from sentence encoders are complemented by sentence representations built from word embeddings as a function of trainable neural network parameters. We explore multiple ways to deal with the multi-label aspect. The adopted method produces label-wise probabilities directly and simultaneously using shared weights and a joint loss function. Our experimentation finds multiple instances of the proposed framework outperforming several diverse baselines on established multi-label classification metrics.

Our key contributions are summarized below.

- We propose a neural framework for the multi-label classification of accounts of sexism that can combine sentence representations built from word embeddings of different kinds through learnable model parameters with those created using pre-trained models. It yields results superior to many deep learning and traditional machine learning baselines.
- To the best of our knowledge, this is the first work on classifying an account recounting any type(s) of sexism without the assumption of the mutual exclusivity of classes.
- We provide a dataset consisting of 13023 accounts of sexism by survivors and observers annotated with one or more of 23 carefully formulated categories of sexism.

## 2 Related Work

Substantial work has been directed to hate speech detection in recent years. Since some of it involves the detection of sexism (Badjatiya et al., 2017; Waseem and Hovy, 2016), we review it along with the work on sexism classification.

### 2.1 Hate Speech Detection

Warner and Hirschberg (2012) identify anti-semitic hate speech using SVM. Gao et al. (2017) perform hate speech detection in a weakly supervised fashion. Nobata et al. (2016) distinguish abusive comments from clean ones through various NLP and embedding-derived features. Burnap and Williams (2016) classify cyber hate with respect to race, disability, and sexual orientation using text parsing to extract typed dependencies. Waseem and Hovy (2016) explore the role of extra-linguistic features along with character n-grams in classifying tweets as racist, sexist or neither. Badjatiya et al. (2017) experiment with various deep learning approaches for the same three-way classification. Zhang and Luo (2018) explore skipped CNN and a combination of CNN and GRU for hate speech detection. Unlike these papers, we seek to categorize accounts of sexism, a specific form of discrimination or hate.

### 2.2 Sexism Categorization

Anzovino et al. (2018) use features such as (Part of Speech) n-grams and text embeddings for the 5-class categorization of misogynistic language. Jafarpour et al. (2018) classify sexist tweets into one of four categories, which deal with harassment and threats. Jha and Mamidi (2017) experiment with SVM, biLSTM with attention, and fastText to categorize tweets as benevolent, hostile, or non-sexist. Their way of categorizing sexism relates to the manner in which sexism is stated. While it is applicable to all sexist remarks, narrated accounts of sexism may not always capture how the perpetrators stated sexism. Karlekar and Bansal (2018) focus on accounts of sexual harassment, exploring CNN and/or RNN for their 3-class classification. As far as we know, our paper presents the first attempt to categorize accounts involving any type(s) of sexism in a multi-label way. Moreover, we pro-

vide a larger dataset and significantly more extensive or finer-grained categorization scheme than these papers.

## 3 Dataset Construction

Creating our multi-label sexism account categorization dataset entailed two parts: textual data collection and data annotation. To collect data, we crawled the Everyday Sexism Project website, which receives numerous accounts of sexism from survivors themselves as well as observers. After removing entries with less than 7 words, around 20000 entries were shortlisted for annotation; we prioritized shorter ones and tried to approximate the tag distribution on the website. Though shorter entries were preferred keeping in mind the potential future work of transfer learning to Twitter content, our neural framework is devised in a size-agnostic way.

Under the direction of a social scientist, 23 categories of sexism were formulated taking into account gender-related discourse and campaigns (Dutta and Sircar, 2013; Eccles et al., 1990; Mead, 1963; Menon, 2012) as well as possible impact on public policy. Table 1 provides succinct descriptions for the categories.

We followed a three-phase annotation process to ensure that the categorization of each account of sexism in the final dataset involved the labeling of it by at least two of our 10 annotators, most of whom have studied topics related to gender and/or sexuality formally. The annotators were given detailed guidelines, which evolved during the course of their work. Each annotator was given training, which included a pilot round involving evaluation and feedback. Phase 1 involved identifying one or more textual portions capturing distinct accounts of sexism from an entry obtained from Everyday Sexism Project and subsequently tagging each portion with at least one of the 23 categories of sexism, producing over 23000 labeled accounts. In phases 2 and 3, we sought redundancy of annotations for improved quality, as permitted by the availability of annotators adequately knowledgeable about sexism. Over 21000 accounts were categorized again in phase 2 such that the annotators for phases 1 and 2 were different. The inter-annotator agreement across phases 1 and 2, measured by the average of the Cohen's Kappa (Cohen, 1960; Artstein and Poesio, 2008) scores for the per-category pairs of binary label vectors, is 0.584. Each account for which the label sets annotated across phases 1 and 2 were identical was included in the dataset along with the associated label set. In phase 3, some of the accounts for which there was a mismatch between the phase 1 and phase 2 annotations were selected. For each account, the annotators were presented with only the mismatched categories and asked to select or reject each. Duplicates and records for which the Everyday Sexism Project entry numbers match but the accounts do not fully match were removed at multiple stages. In order to improve the annotation reliability further, some records for which the annotations differed across phases 1 and 2 were discarded based on the annotators involved and sensitivities of the categories, resulting in a multi-label sexism categorization dataset of 13023 accounts. For our automated sexism classification experiments, we merge the categories found in less than 400 records with others as follows, resulting in 14 categories. 'Menstruation-related discrimination' and 'Motherhood-related discrimination' are merged into 'Motherhood and menstruation related discrimination'; 'Mansplaining', 'Gaslighting', 'Religion-based sexism', 'Physical violence (excluding sexual violence)', and 'Other' are merged into 'Other'; 'Pay gap' and 'Hostile work environment (excluding pay gap)' are merged into 'Hostile work environment'; 'Tone policing', 'Moral policing (excluding tone policing)', and 'Victim blaming' are merged into 'Moral policing and victim blaming'; 'Rape' and 'Sexual assault (excluding rape)' are merged into 'Sexual assault'. Our dataset, however, retains all 23 categories. Fig. 1 shows the frequency distribution of the number of labels per account in the dataset, demonstrating the multi-label nature of instances of sexism.

**Caution**: 1) The category frequencies in our dataset (used for merging categories) do not represent real-world instances of sexism, as they are affected by several factors including the bias of our sampling scheme toward smaller posts and the small size of our dataset relative to the immense degree of prevalence of sexism in the world. 2) Labeling categories of sexism can be complex in many cases. Hence, despite our best efforts, our labeled data may contain inaccuracies or discrepancies. We also recognize that our categorization scheme could be improved.

Table 1: Descriptions of the categories of sexism used in our dataset

| Category | Description |
|---|---|
| Role stereotyping | Socially constructed false generalizations about certain roles being more appropriate for women; also applies to such misconceptions about men |
| Attribute stereotyping | Mistaken linkage of women with some physical, psychological, or behavioral qualities or likes/dislikes; also applies to such false notions about men |
| Body shaming | Objectionable comments or behaviour concerning appearance including the promotion of certain body types or standards |
| Hyper-sexualization (excluding body shaming) | Unwarranted focus on physical aspects or sexual acts |
| Internalized sexism | The perpetration of sexism by women via comments or other actions |
| Pay gap | Unequal salaries for men and women for the same work profile |
| Hostile work environment (excluding pay gap) | Sexism encountered by an employee at the workplace; also applies when a sexist misdeed committed outside the workplace by a co-worker makes working uncomfortable for the victim |
| Denial or trivialization of sexist misconduct | Denial or downplaying of sexist wrongdoings |
| Threats | All threats including wishing for violence or joking about it, stalking, threatening gestures, or rape threats |
| Rape | FBI's expanded definition of rape |
| Sexual assault (excluding rape) | Any sexual contact without consent; unwanted touching |
| Sexual harassment (excluding assault) | Any sexually objectionable behaviour |
| Tone policing | Comments or actions that cause or aggravate restrictions on how women communicate |
| Moral policing (excluding tone policing) | The promotion of discriminatory codes of conduct for women in the guise of morality; also applies to statements that feed into such codes and narratives |
| Victim blaming | The act of holding the victim responsible (fully or partially) for sexual harassment, violence, or other sexism perpetrated against her |
| Slut shaming | Inappropriate comments made about women 1) deviating from conservative expectations relating to sex or 2) dressing in a certain way when it gets linked to sexual availability |
| Motherhood-related discrimination | Shaming, prejudices, or other discrimination or misconduct related to the notion of motherhood; also applies to the violation of reproductive rights |
| Menstruation-related discrimination | Shaming, prejudices, or other discrimination or wrongdoings related to periods |
| Religion-based sexism | Sexist discrimination or prejudices stemming from religious scriptures or constructs |
| Physical violence (excluding sexual violence) | Domestic abuse, murder, kidnapping, confinement, or other physical acts of violence linked to sexism |
| Mansplaining | A woman being condescendingly talked down to by a man; also applies when a man gives an unsolicited advice or explanation to a woman related to something she knows well that she disapproves of |
| Gaslighting | Sexist manipulation of the victim through psychological means into doubting her own sanity |
| Other | Any type of sexism not covered by the above categories |

### 3.1 Ethical Data Use and Release

We are committed to following ethical practices, which includes protecting the privacy and anonymity of the victims. We only use accounts of sexism and tags from entries on the Everyday Sexism Project website (ESP). The entry titles, which could contain sensitive information related to the names or locations of the victims (or contributors), are not saved or used at all.

Our dataset can be requested for academic purposes only by providing some prerequisites as recommended by an ethics committee and agreeing to certain terms through our website[2]. The requesters who fulfill these conditions will be emailed 1) the data comprising only numerical placeholders and labels, 2) a script that fetches only accounts of sexism from ESP to obtain the account for each placeholder, and 3) the annotation guidelines used. We have devised this method to ensure that if an entry gets removed from ESP by a victim (or contributor), any and all parts of it in our dataset will also be removed.

[2] https://irel.iiit.ac.in/sexism-classification

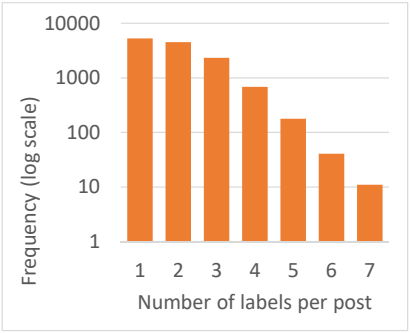


Figure 1: Frequency distribution of #labels per post

## 4 Sexism Categorization Approach

Given an account of sexism (post), our objective is to predict a list of up to 14 applicable categories of sexism, making this a multi-label multi-class classification task. In this section, we detail our proposed framework, which enables combining sentence representations derived from word embeddings using trainable model parameters with those obtained using general-purpose models. Our architecture is depicted in Fig. 2. We also discuss how we tap unlabeled data and loss functions.

### 4.1 Sexism Categorization Architecture

Let each post contain a maximum of $|S|$ sentences with a maximum of $|W|$ words per sentence. Ev-

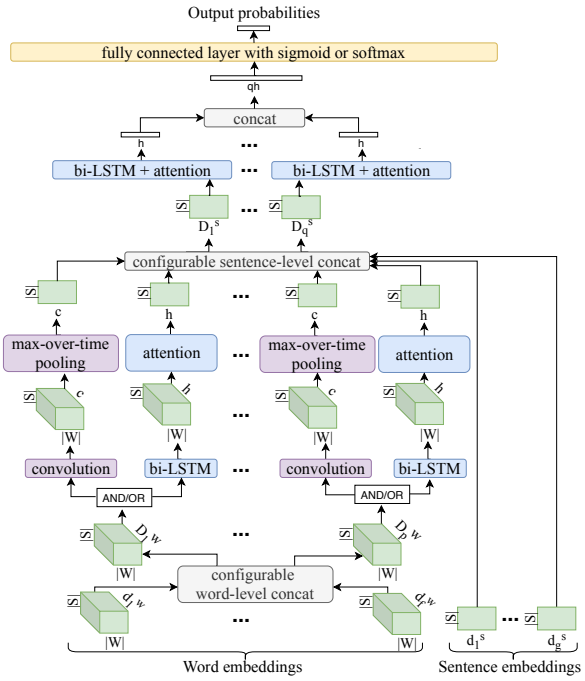


Figure 2: Proposed sexism categorization architecture

ery word (or sentence) can be represented using multiple word (or sentence) embedding methods. Let $f$ (or $g$) be the number of word (or sentence) embedding methods chosen. Let $d_i^w$ (or $d_j^s$) be the embedding dimension for the $i^{th}$ (or $j^{th}$) word (or sentence) embedding scheme. Each post is represented using two kinds of tensors: (a) $f$ tensors $\in \mathcal{R}^{|S|\times|W|\times d_i^w}$ created using different word embeddings, and (b) $g$ tensors $\in \mathcal{R}^{|S|\times d_j^s}$ constructed using different sentence encoders.

First, subsets of the $f$ tensors based on word embeddings are concatenated in a configurable manner (*configurable word-level concat* in Fig. 2), producing $p$ tensors $\in \mathcal{R}^{|S|\times|W|\times D_i^w}$, where $D_i^w$ is the dimension resulting from the $i^{th}$ concatenation. Next, we construct vector representations for the sentences word-embedded in each of the $p$ tensors using CNN-based and/or LSTM-based operations as configured. The CNN-based operations begin with convolutional filters being applied along the word dimension (Kim, 2014) to generate many bigram, trigram and 4-gram based features. This is followed by max-over-time pooling, which picks the largest value for each filter and produces a sentence representing tensor $\in \mathcal{R}^{|S|\times c}$, where $c$ is the total number of convolutional filters used. The LSTM-based components include biLSTM followed by an attention mechanism (Yang et al., 2016) through which the LSTM outputs across time steps are aggregated into a vector representation for each sentence, resulting in a tensor $\in \mathcal{R}^{|S|\times h}$, where $h$ is the bi-LSTM output length. At this stage, we have three types of sentence representing tensors if both CNN-based and RNN-based operations are chosen to be applied on all word embedding tensors: (a) $p$ tensors $\in \mathcal{R}^{|S|\times c}$ from the CNN-based processing, (b) $p$ tensors $\in \mathcal{R}^{|S|\times h}$ from the LSTM-based processing, and (c) $g$ tensors $\in \mathcal{R}^{|S|\times d_j^s}$ obtained using general-purpose sentence encoders.

From these sentence representing tensors, subsets are concatenated to produce $q$ tensors $\in \mathcal{R}^{|S|\times D_j^s}$ (*configurable sentence-level concat* in Fig. 2), where $D_j^s$ is the dimension stemming from the $j^{th}$ concatenation. The sequence of sentence vectors in each of these $q$ tensors is then passed through bi-LSTM followed by attention-based aggregation, producing $q$ representations for a post collectively. These vectors are then concatenated to produce the overall post representation. The final step involves a fully connected layer with a sigmoid or softmax non-linearity depending on the loss function used, generating the output probabilities.

### 4.2 Word and Sentence Representations

We model a post using both word embeddings and sentence embeddings. We experiment with three distributional word vectors, namely ELMo (Peters et al., 2018), GloVe (Pennington et al., 2014), and fastText (Bojanowski et al., 2017), and a linguistic feature vector. Our linguistic feature representation comprises a variety of features, namely features from the biased language detection work (assertive verb, implicative verb, hedges, factive verb, report verb, entailment, strong subjective, weak subjective, positive word, and negative word) (Recasens et al., 2013), PERMA (Positive Emotion, Engagement, Relationships, Meaning, and Accomplishments) features for both polarities (Schwartz et al., 2016), associations with eight basic emotions (anger, fear, anticipation, trust, surprise, sadness, joy, and disgust) and two sentiments (negative and positive) from the NRC emotion lexicon (Mohammad and Turney, 2013), and affect (valence, arousal, and dominance) scores (Mohammad, 2018). Missing values are filled with zero for binary features and with the mean for non-binary ones.

We explore the following for creating sentence embeddings: BERT (Devlin et al., 2018), Universal Sentence Encoder (USE) (Cer et al., 2018), and

InferSent (Conneau et al., 2017). Our choice of utilizing these models is warranted by the fact that the corpora that they are trained on are considerably bigger than the textual data that we have for supervised learning and hence likely contain greater semantic diversity.

### 4.3 Utilizing Unlabeled Data

Models such as BERT are not trained to generate representations tuned to a specific domain. We use over 90000 entries crawled from Everyday Sexism Project's website to tailor a pre-trained BERT model for obtaining more effective representations for our model. After removing the unlabeled entries corresponding to the posts in the test and validation data, we use the rest to tune the BERT parameters using its masked language modeling and next sentence prediction tasks. We henceforth refer to this refined model as tBERT (tuned BERT).

### 4.4 Loss Function Choice

Since the popular cross-entropy loss is inapt for our multi-label classification task in its standard form, we explore two alternatives. Binary (multi-hot) target vectors are used for both.

#### 4.4.1 Extended Binary Cross Entropy Loss

We adopt an Extended version of the Binary Cross Entropy loss ($EBCE$), formulated as a weighted mean of label-wise binary cross entropy values in order to neutralize class imbalance.

$$L_{EBCE} = -\frac{1}{n}\sum_{i=1}^{n}\frac{1}{L}\sum_{j=1}^{L} w_{jy_{ij}}\left\{y_{ij}\log(\hat{p}_{ij}^{\sigma}) + (1-y_{ij})\log(1-\hat{p}_{ij}^{\sigma})\right\} \quad (1)$$

Here, $n$ and $L$ denote the number of samples (posts) and the number of classes respectively. $y_{ij}$ is 1 if label $l_j$ applies to post $x_i$ and 0 otherwise. $\hat{p}_{ij}^{\sigma}$ is the estimated probability of label $l_j$ being applicable to post $x_i$ computed using a sigmoid activation atop the fully connected layer with $L$ output units. The weights for correcting class imbalance $w_{jv}$ are computed as follows.

$$w_{jv} = \frac{n}{2|\{x_i \mid y_{ij} = v, 1 \leq i \leq n\}|} \quad (2)$$

#### 4.4.2 Normalized Cross Entropy Loss

We also experiment with a Normalized variant of the Cross Entropy loss tailored for a multi-label problem configuration also mitigating class imbalance (referred to as NCE).

$$L_{NCE} = -\frac{1}{n}\sum_{i=1}^{n}\frac{1}{|\mathbf{y_i^+}|}\sum_{j=1}^{L} w_j^c\{y_{ij}\log(\hat{p}_{ij})\} \quad (3)$$

Here, $\mathbf{y_i^+}$ is the set of labels applicable to post $x_i$. $\hat{p}_{ij}$ denotes the estimated probability of label $l_j$ being applicable to post $x_i$ computed through a softmax function. The class imbalance negating weights $w_j^c$ are generated as follows.

$$w_j^c = \frac{n}{\sum_{i=1}^{n}\frac{y_{ij}}{|\mathbf{y_i^+}|}} \quad (4)$$

Unlike in single-label multi-class classification, wherein $\arg\max$ can be applied to the probability vector generated by softmax to make the prediction, we could apply a threshold on probability vector $\hat{\mathbf{p}}_\mathbf{i}$ to find the (potentially) multiple applicable classes for post $x_i$. Instead of using a fixed, manually tuned threshold-related parameter, we devise an automated method for estimating a per-sample cut-off position. For each sample, we descendingly sort the probability scores, compute the differences between successive (sorted) score pairs, find the index $m$ corresponding to the maximum value in the list of differences, and select the classes corresponding to indices $[1..m]$. Note that when sigmoid (along with $EBCE$ loss) is used instead of softmax, the prediction is made by rounding the probability vector, since it comprises the class-wise binary prediction probabilities.

#### 4.4.3 Discussion on Single-label Transformations

Traditional approaches to multi-label classification include transforming the problem to one or more single-label classification problems. The Label Powerset (LP) method (Boutell et al., 2004) treats each distinct combination of classes existing in the training set as a separate class. The standard cross-entropy loss can then be used along with softmax. This transformative method may impose a greater computational cost than the direct approach using the $EBCE$ loss since the cardinality of the transformed label set may be relatively high. Moreover, LP does not generalize to label combinations not covered in the training set. Another approach based on problem transformation is binary relevance (BR) (Boutell et al., 2004). An independent binary classifier is trained to predict the applicability of each label in this method. This entails training a total of $L$ classifiers, making BR computationally very expensive. Additionally, its performance is affected by the fact that it disregards correlations existing between labels.

## 5 Experiments

We evaluate the proposed framework against several baselines and provide qualitative and quantitative analyses. Our code is available on GitHub[3]. Our implementation utilizes parts of the code from (Agrawal and Awekar, 2018; Pattisapu et al., 2017; Liao, 2017) and libraries Keras and Scikit-learn (Pedregosa et al., 2011). We reserve 15% of the data for testing and validation each.

### 5.1 Evaluation Metrics

Owing to the multi-label nature of this classification, standard metrics used in single-label multi-class classification are unsuitable. We adopt established example (instance) based metrics, namely F1 ($F_I$) and accuracy ($Acc_I$), and label-based metrics, namely F1 macro ($F_{macro}$) and F1 micro ($F_{micro}$) used in multi-label classification (Zhang and Zhou, 2014).

### 5.2 Baselines

Random

Labels are selected randomly as per their normalized frequencies in the training data for each test sample.

Traditional Machine Learning (ML)

We experiment with Support Vector Machine (SVM), Random Forests (RF), and Logistic Regression (LR). The features explored include TF-IDF on character n-grams (1-5 characters), TF-IDF on word unigrams and bigrams, the mean of the ELMo vectors for the words in a post, and the composite set of features similar to (Anzovino et al., 2018) comprising n-gram based, POS-based, and doc2vec (Le and Mikolov, 2014) features, the post length, and the adjective count.

LSTM-based Architectures

**biLSTM**: The word embeddings for all words in a post are fed to bidirectional LSTM.

**biLSTM-Attention**: Same as biLSTM but with the attention mechanism by Yang et al. (2016).

**Hierarchical-biLSTM-Attention**: For the words in each sentence, the word embeddings are passed through biLSTM with attention to create a sentence embedding. These sentence embeddings are in turn fed to another instance of biLSTM with attention. This broadly follows the architecture proposed for document classification by Yang et al. (2016) with GRUs replaced with LSTMs.

[3] https://github.com/pulkitparikh/sexism_classification

**Sentence embeddings with biLSTM-Attention**: Sentence representations generated using a generic encoder (BERT using bert-as-service (Xiao, 2018), USE, and InferSent) are passed through biLSTM with attention.

CNN and CNN-LSTM based Architectures

**CNN-Kim**: Similar to (Kim, 2014), this involves applying convolutional filters followed by max-over-time pooling to the word vectors for a post.

**C-biLSTM**: In this variant of the C-LSTM architecture (Zhou et al., 2015) somewhat related to an approach used by Karlekar and Bansal (2018) for multi-label sexual harassment classification, after applying convolution on the word vectors for a post, the feature maps are stacked along the filter dimension to create a sequence of window vectors, which are then fed to biLSTM.

**CNN-biLSTM-Attention**: For each sentence, convolutional and max-over-time pooling layers are applied on the embeddings of its words. The resultant sentence representations are put through bi-LSTM with the attention mechanism. This approach is similar to (Wang et al., 2016) with the attention scheme from (Yang et al., 2016) added.

The architectures of the deep learning baselines have a fully connected layer with the sigmoid or softmax non-linearity (depending on the loss function used) at the end.

### 5.3 Results

Table 2 shows results produced using traditional ML methods (SVM, RF, and LR) across four different feature sets (word n-grams, character n-grams, averaged ELMo vectors, and composite features). We use Label Powerset for these methods, since the direct (non-transformative) formulation cannot be used with them. Among these combinations, logistic regression with averaged ELMo embeddings as features performs the best.

Table 3 contains results for the random and deep learning baselines and different variants of the proposed framework. For each method, the average over three runs is reported for each metric. We find ELMo to be better than GloVe and fastText for word embeddings across multiple baselines and hence show only ELMO-based results for the baselines. We report all results with the EBCE loss; the NCE loss produced inferior results across multiple methods. For our framework, s() denotes sentence-level concatenation; wl() de-

Table 2: Results with traditional machine learning (Label Powerset)

| Features⟶ | Word n-grams | | | | Character n-grams | | | | Averaged ELMo vectors | | | | Composite features | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Classifier↓ | $F_I$ | $F_{macro}$ | $Acc_I$ | $F_{micro}$ | $F_I$ | $F_{macro}$ | $Acc_I$ | $F_{micro}$ | $F_I$ | $F_{macro}$ | $Acc_I$ | $F_{micro}$ | $F_I$ | $F_{macro}$ | $Acc_I$ | $F_{micro}$ |
| SVM | 0.448 | 0.373 | 0.324 | 0.410 | 0.449 | 0.374 | 0.331 | 0.416 | 0.546 | 0.430 | 0.431 | 0.500 | 0.178 | 0.094 | 0.116 | 0.174 |
| LR | 0.357 | 0.315 | 0.236 | 0.349 | 0.357 | 0.311 | 0.230 | 0.352 | **0.595** | **0.479** | **0.478** | **0.549** | 0.438 | 0.370 | 0.311 | 0.421 |
| RF | 0.531 | 0.398 | 0.438 | 0.476 | 0.395 | 0.205 | 0.325 | 0.349 | 0.375 | 0.164 | 0.305 | 0.331 | 0.460 | 0.311 | 0.380 | 0.415 |

notes word level concatenation and LSTM-based processing; wc() denotes word level concatenation and CNN-based processing. We note that the results are reported for only some of the many instances that can arise from our configurable architecture. Our framework provides the ability to explore different configurations such as those with multiple s() operations, depending on the problem at hand.

We observe the following: (1) The random baseline performs poorly, confirming the complexity of the problem. (2) biLSTM-Attention and Hierarchical-biLSTM-Attention are the two best baselines. (3) Several variants of the proposed framework outperform all baselines. Based on $F_I$ and $F_{macro}$, our best method is s(wl(ELMo), wl(GloVe), tBERT), though adding linguistic features (Ling) to it slightly improves some metrics. (4) BERT tuned on unlabeled instances of sexism (tBERT) works better than the vanilla BERT counterpart and other sentence encoders. (5) Combining tBERT sentence representations with those generated from ELMo word vectors using biLSTM with attention works better than using either individually. (6) Along with tBERT, concatenating ELMo and GloVe at the sentence level (s(wl(ELMo), wl(GloVe), tBERT)) is better than concatenating them at the word level (s(wl(ELMo, GloVe), tBERT)) while processing word vectors using biLSTM with attention. (7) The LSTM based processing of word embeddings produces better results than the CNN based counterpart.

Table 3: Results for the proposed methods and deep learning baselines (using ELMo embeddings) with the EBCE loss and Random

| | Approach | $F_I$ | $F_{macro}$ | $Acc_I$ | $F_{micro}$ |
|---|---|---|---|---|---|
| Baselines | Random | 0.042 | 0.141 | 0.027 | 0.193 |
| | biLSTM | 0.697 | 0.616 | 0.563 | 0.658 |
| | biLSTM-Attention | 0.728 | 0.650 | 0.601 | 0.688 |
| | Hierarchical-biLSTM-Attention | 0.725 | 0.650 | 0.604 | 0.688 |
| | BERT-biLSTM-Attention | 0.656 | 0.555 | 0.502 | 0.611 |
| | USE-biLSTM-Attention | 0.628 | 0.549 | 0.468 | 0.594 |
| | InferSent-biLSTM-Attention | 0.418 | 0.37 | 0.274 | 0.399 |
| | CNN-biLSTM-Attention | 0.714 | 0.628 | 0.586 | 0.671 |
| | CNN-Kim | 0.701 | 0.622 | 0.574 | 0.669 |
| | C-biLSTM | 0.708 | 0.631 | 0.583 | 0.674 |
| Proposed methods | tBERT-biLSTM-Attention | 0.688 | 0.589 | 0.539 | 0.644 |
| | s(wl(ELMo), tBERT) | 0.747 | 0.675 | 0.628 | 0.710 |
| | s(wl(ELMo, GloVe), tBERT) | 0.743 | 0.667 | 0.618 | 0.703 |
| | s(wc(ELMo), wc(GloVe), tBERT) | 0.738 | 0.654 | 0.614 | 0.698 |
| | s(wl(ELMo), wl(GloVe), tBERT) | **0.756** | **0.684** | **0.635** | **0.715** |
| | s(wl(ELMo), wl(GloVe), tBERT, USE) | 0.753 | 0.673 | 0.632 | 0.715 |
| | s(wl(ELMo), wl(GloVe), wl(Ling), tBERT) | **0.753** | **0.685** | **0.636** | **0.718** |
| | s(wc(ELMo), wl(ELMo), wc(GloVe), wl(GloVe), tBERT) | 0.741 | 0.664 | 0.625 | 0.705 |

The pre-processing steps that we perform for all deep learning methods include removing certain non-alpha-numeric characters and extra spaces, lower-casing, and zero-padding input tensors as appropriate. While breaking a post into sentences, each sentence containing more than 35 words is split into multiple sentences, ensuring the maximum sentence length of 35 words.

Using experiments on a validation set, which was merged into the training set during the test runs, for each method, we choose the values of three hyper-parameters: the LSTM dimension, the attention dimension, and the number CNN filters for kernel sizes 2, 3, and 4 each. The values used for instances of our framework and the deep learning baselines are provided in Table 5.

We employ 0.25 dropouts after each input and before the final, fully connected layer. The learning rate was set to 0.001 and the number of epochs to 10. We use a batch size of 64. These fixed parameters were kept unchanged across methods.

The hyper-parameter values for the traditional ML methods are as follows. For SVM, soft margin (C) is set to 1.0. For RF (Random Forest), the number of estimators is 100. For extracting character and word n-grams, the maximum number of features used, word n-gram range, and character n-gram range are 10000, (1,2), and (1,5) respectively. For SVM and LR (Logistic Regression), we apply class imbalance correction.

For tapping unlabeled data, we pre-train the 'BERT-Base, Uncased' model[4] for 100000 steps with a batch size of 25. For vannila BERT, we use the bigger 'BERT-Large, Uncased' model, which we could not use for pre-training because of computational constraints. For generating GloVe word embeddings, we use the 840B-token, cased model.

Table 4 lists accounts of sexism from the test set for which our best method made the right predictions but the best baseline did not, along with the labels. It also highlights the top two words per sentence based on the word-level attention weights for wl(ELMo) and wl(GloVe) combined through element-wise max operations. For the first ac-

[4] https://github.com/google-research/bert

Table 4: Attention analysis for some test samples

| Account of sexism | Two most-attended-to words / sentence | Labels |
|---|---|---|
| am in social services. my male coworker and I have the same job title. Everyone assumes I am his assistant or an intern. It's not just clients who assume he's my boss, it's other agencies. The same ppl who think nothing calling me honey at work. | (services, am), (coworker, title), (intern, assistant), (boss, agencies), (honey, me) | Role stereotyping, Hostile work environment, Sexual harassment (excluding assault) |
| being told I should take cat calls as compliments by my father | (compliments, cat) | Denial or trivialisation of sexist misconduct, Sexual harassment (excluding assault) |
| I didn't appreciate it when my own father walked into the house one day while I was doing laundry and told me that "it's nice to see you finally doing women's work." | (womens, work) | Role stereotyping, Moral policing and victim blaming |
| Referred to as 'not a girl' because I have short hair and don't wear noticeable makeup. | (makeup, hair) | Attribute stereotyping, Body shaming |
| At our school girls are forbidden to wear tight trowsers or remove their blazers- in fear of distracting the boys. | (tight, trowsers) | Moral policing and victim blaming, Hyper-sexualization (excluding body shaming) |

Table 5: Tuned hyper-parameter values for the deep learning baselines (ELMo embeddings) and proposed methods with the EBCE loss

| Approach | LSTM dimension | Attention dimension | CNN Filters of each kernel size |
|---|---|---|---|
| Hierarchical-biLSTM-Attention | 300 | 400 | N.A. |
| CNN-biLSTM-Attention | 300 | 400 | 100 |
| BERT-biLSTM-Attention | 200 | 500 | N.A. |
| USE-biLSTM-Attention | 300 | 600 | N.A. |
| InferSent-biLSTM-Attention | 100 | 200 | N.A. |
| C-biLSTM | 300 | N.A. | 150 |
| biLSTM-Attention | 200 | 300 | N.A. |
| biLSTM | 300 | N.A. | N.A. |
| CNN-Kim | N.A. | N.A. | 150 |
| tBERT-biLSTM-Attention | 300 | 600 | N.A. |
| s(wl(ELMo), tBERT) | 300 | 600 | N.A. |
| s(wl(ELMo, GloVe), tBERT) | 100 | 100 | N.A. |
| s(wl(ELMo), wl(GloVe), tBERT) | 100 | 200 | N.A. |
| s(wl(ELMo), wl(GloVe), tBERT, USE) | 300 | 600 | N.A. |
| s(wl(ELMo), wl(GloVe), wl(Ling), tBERT) | 300 | 600 | N.A. |
| s(wc(ELMo), wc(GloVe), tBERT) | 300 | 600 | 100 |
| s(wc(ELMo), wl(ELMo), wc(GloVe), wl(GloVe), tBERT) | 300 | 500 | 100 |

Table 6: Performance variation across #labels per post

| #Labels per post | $F_I$ | $F_{macro}$ | $Acc_I$ | $F_{micro}$ |
|---|---|---|---|---|
| 1 | 0.729 | 0.527 | 0.632 | 0.637 |
| 2 | 0.754 | 0.675 | 0.631 | 0.722 |
| 3 | 0.781 | 0.721 | 0.652 | 0.764 |
| 4 | 0.743 | 0.722 | 0.604 | 0.735 |
| 5 | 0.739 | 0.592 | 0.592 | 0.735 |

count of sexism, our model produces words like "intern", "assistant" and "boss", associated with role stereotyping, among the top two words across sentences. Likewise, "honey" related to sexual harassment and "boss", "coworker", and "services" related to hostile work environments also surface. Moreover, the top two sentences based on the sentence-level attention weights of our model are the last two, which evidence all category labels. For other posts too, the model produces category-relevant top two words per sentence; "womens" and "work" relate to role stereotyping and moral policing; "tight" and "trowsers" relate to hyper-sexualization and moral policing; "makeup" relates to attribute stereotyping; "hair" from "short hair" relates to body shaming; "cat" from "cat calls" relates to sexual harassment; "compliments" from "take cat calls as compliments" relates to denial or trivialization of sexist misconduct.

Table 6 shows results for one run of our best method across different numbers of labels per post (1 to 5). Entries for values 6 and 7, which have less than 10 associated test samples, are omitted. The best results are observed for values 2 to 4, suggesting that our approach performs better on multi-label samples.

## 6 Conclusion

We explored classifying an account reporting any kind(s) of sexism such that the categories can co-occur. We developed a neural framework that outperforms many deep learning and traditional ML baselines for this multi-label sexism classification. Moreover, we provided the largest dataset for sexism classification, linked with 23 categories. Directions for future work include devising approaches that perform sexism classification more accurately, enhancing the categorization scheme, and developing other ways to help counter sexism. We hope that this paper will give rise to further work aimed at fighting sexism.

## 7 Acknowledgments

We are really grateful to Everyday Sexism Project. We also thank Bhavapriya Thottakad, Himakshi Baishya, Aanchal Gupta, Subhashini R, Shalini Pathi, Hadia Ahsan, and Reshma Kalpana, without whose efforts in analyzing accounts of sexism this paper would not have materialized. We are grateful to Ponnurangam Kumaraguru, too, for his time and inputs.

## References


Sweta Agrawal and Amit Awekar. 2018. Deep learning for detecting cyberbullying across multiple social media platforms. In *European Conference on Information Retrieval*, pages 141–153. Springer.

Maria Anzovino, Elisabetta Fersini, and Paolo Rosso. 2018. Automatic identification and classification of misogynistic language on twitter. In *International Conference on Applications of Natural Language to Information Systems*, pages 57–64. Springer.

Ron Artstein and Massimo Poesio. 2008. Inter-coder agreement for computational linguistics. *Computational Linguistics*, 34(4):555–596.

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 759–760. International World Wide Web Conferences Steering Committee.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Matthew R Boutell, Jiebo Luo, Xipeng Shen, and Christopher M Brown. 2004. Learning multi-label scene classification. *Pattern recognition*, 37(9):1757–1771.

Pete Burnap and Matthew L Williams. 2016. Us and them: identifying cyber hate on twitter across multiple protected characteristics. *EPJ Data Science*, 5(1):11.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 670–680.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Debolina Dutta and Oishik Sircar. 2013. India's winter of discontent: Some feminist dilemmas in the wake of a rape. *Feminist Studies*, 39(1):293–306.

Jacquelynne S Eccles, Janis E Jacobs, and Rena D Harold. 1990. Gender role stereotypes, expectancy effects, and parents' socialization of gender differences. *Journal of social issues*, 46(2):183–201.

Lei Gao, Alexis Kuppersmith, and Ruihong Huang. 2017. Recognizing explicit and implicit hate speech using a weakly supervised two-path bootstrapping approach. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 774–782.

Borna Jafarpour, Stan Matwin, et al. 2018. Boosting text classification performance on sexist tweets by text augmentation and text generation using a combination of knowledge graphs. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 107–114.

Akshita Jha and Radhika Mamidi. 2017. When does a compliment become sexist? analysis and classification of ambivalent sexism using twitter data. In *Proceedings of the second workshop on NLP and computational social science*, pages 7–16.

Sweta Karlekar and Mohit Bansal. 2018. Safecity: Understanding diverse forms of sexual harassment personal stories. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2805–2811.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196.

Richard Liao. 2017. textclassifier. https://github.com/richliao/textClassifier.

Margaret Mead. 1963. *Sex and temperament in three primitive societies*, volume 370. Morrow New York.

Nivedita Menon. 2012. *Seeing like a feminist*. Penguin UK.

Saif Mohammad. 2018. Obtaining reliable human ratings of valence, arousal, and dominance for 20,000 english words. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 174–184.

Saif M Mohammad and Peter D Turney. 2013. Crowdsourcing a word–emotion association lexicon. *Computational Intelligence*, 29(3):436–465.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive language detection in online user content. In *Proceedings of the 25th international conference on world wide web*, pages 145–153. International World Wide Web Conferences Steering Committee.

Nikhil Pattisapu, Manish Gupta, Ponnurangam Kumaraguru, and Vasudeva Varma. 2017. Medical persona classification in social media. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, pages 377–384. ACM.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Marta Recasens, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. 2013. Linguistic models for analyzing and detecting biased language. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1650–1659.

H Andrew Schwartz, Maarten Sap, Margaret L Kern, Johannes C Eichstaedt, Adam Kapelner, Megha Agrawal, Eduardo Blanco, Lukasz Dziurzynski, Gregory Park, David Stillwell, et al. 2016. Predicting individual well-being through the language of social media. In *Biocomputing 2016: Proceedings of the Pacific Symposium*, pages 516–527. World Scientific.

Jin Wang, Liang-Chih Yu, K Robert Lai, and Xuejie Zhang. 2016. Dimensional sentiment analysis using a regional cnn-lstm model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 225–230.

William Warner and Julia Hirschberg. 2012. Detecting hate speech on the world wide web. In *Proceedings of the second workshop on language in social media*, pages 19–26. Association for Computational Linguistics.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL student research workshop*, pages 88–93.

Han Xiao. 2018. bert-as-service. https://github.com/hanxiao/bert-as-service.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Min-Ling Zhang and Zhi-Hua Zhou. 2014. A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering*, 26(8):1819–1837.

Ziqi Zhang and Lei Luo. 2018. Hate speech detection: A solved problem? the challenging case of long tail on twitter. *Semantic Web*, pages 1–21.

Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis Lau. 2015. A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630*.

# The Trumpiest Trump? Identifying a Subject's Most Characteristic Tweets

**Charuta Pethe**
Department of Computer Science,
Stony Brook University, NY, USA
cpethe@cs.stonybrook.edu

**Steven Skiena**
Department of Computer Science,
Stony Brook University, NY, USA
skiena@cs.stonybrook.edu

## Abstract

The sequence of documents produced by any given author varies in style and content, but some documents are more typical or representative of the source than others. We quantify the extent to which a given short text is characteristic of a specific person, using a dataset of tweets from fifteen celebrities. Such analysis is useful for generating excerpts of high-volume Twitter profiles, and understanding how representativeness relates to tweet popularity. We first consider the related task of binary author detection (is $x$ the author of text $T$?), and report a test accuracy of 90.37% for the best of five approaches to this problem. We then use these models to compute characterization scores among all of an author's texts. A user study shows human evaluators agree with our characterization model for all 15 celebrities in our dataset, each with p-value $< 0.05$. We use these classifiers to show surprisingly strong correlations between characterization scores and the popularity of the associated texts. Indeed, we demonstrate a statistically significant correlation between this score and tweet popularity (likes/replies/retweets) for 13 of the 15 celebrities in our study.

## 1 Introduction

Social media platforms, particularly microblogging services such as Twitter, have become increasingly popular (Statista, 2019) as a means to express thoughts and opinions. Twitter users emit tweets about a wide variety of topics, which vary in the extent to which they reflect a user's personality, brand and interests. This observation motivates the question we consider here, of how to quantify the degree to which tweets are characteristic of their author?

People who are familiar with a given author appear to be able to make such judgments confidently. For example, consider the following pair of tweets written by US President Donald Trump, at the extreme sides of our characterization scores (0.9996 vs. 0.0013) for him:

> **Tweet 1:** Thank you for joining us at the Lincoln Memorial tonight- a very special evening! Together, we are going to MAKE AMERICA GREAT AGAIN!

> **Tweet 2:** "The bend in the road is not the end of the road unless you refuse to take the turn." - Anonymous

Although both these tweets are from the same account, we assert that Tweet 1 sounds more characteristic of Donald Trump than Tweet 2. We might also guess that the first is more popular than second. Indeed, Tweet 1 received 155,000 likes as opposed to only 234 for Tweet 2.

Such an author characterization score has many possible applications. With the ability to identify the most/least characteristic tweets from a person, we can generate reduced excerpts for high-volume Twitter profiles. Similarly, identifying the least characteristic tweets can highlight unusual content or suspicious activity. A run of sufficiently unrepresentative tweets might be indicative that a hacker has taken control of a user's account.

But more fundamentally, our work provides the necessary tool to study the question of how "characteristic-ness" or novelty are related to tweet popularity. Do tweets that are more characteristic of the user get more likes, replies and retweets? Is such a relationship universal, or does it depend upon the personality or domain of the author? Twitter users with a large follower base can employ our methods to understand how characteristic a new potential tweet sounds, and obtain an estimate of how popular it is likely to become.

To answer these questions, we formally define the problem of author representativeness testing, and model the task as a binary classification prob-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1653–1663,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

lem. Our primary contributions in this paper include:

- **Five approaches to authorship verification:** As a proxy for the question of representativeness testing (which has no convincing source of ground truth without extensive human annotation), we consider the task of distinguishing tweets written by a given author from others they did not write. We compare five distinct computational approaches to such binary tweet classification (user vs. non-user). Our best model achieves a test accuracy of 90.37% over a dataset of 15 Twitter celebrities. We use the best performing model to compute a score (the probability of authorship), which quantifies how characteristic of the user a given tweet is.

- **Human evaluation study:** To verify that our results are in agreement with human judgment of how 'characteristic' a tweet is, we ask human evaluators which of a pair of tweets sounds more characteristic of the given celebrity. The human evaluators are in agreement with our model 70.40% of the time, significant above the 0.05 level for each of our 15 celebrities.

- **Correlation analysis for popularity:** Our characterization score exhibits strikingly high absolute correlation with popularity (likes, replies and retweets), despite the fact that tweet text is the only feature used to train the classifier which yields these scores.

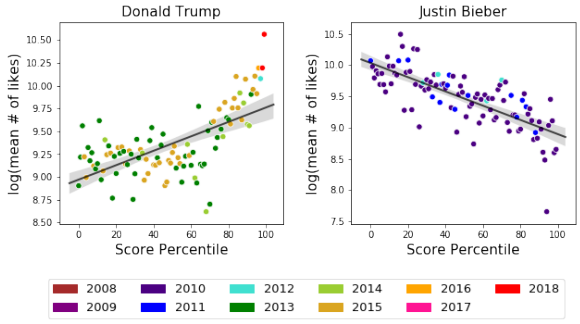


Figure 1: Plot of log mean number of likes against tweet score percentile for Donald Trump and Justin Bieber. Node color denotes the year for which the maximum number of tweets are present in each percentile bucket, demonstrating that this is not merely a temporal correlation.

For 13 of the 15 celebrities in our dataset, we observe a statistically significant correlation between characterization score and popularity. Figure 1 shows the relation between tweet score and tweet popularity for Donald Trump and Justin Bieber respectively. The figure shows that the sign of this association differs for various celebrities, reflecting whether their audience seeks novelty or reinforcement.

- **Iterative sampling for class imbalance:** Our task requires distinguishing a user's tweets (perhaps 1,000 positive training examples) from the sea of all other user's tweets (implying billions of possible negative training examples). We present an iterative sampling technique to exploit this class imbalance, which improves the test accuracy for negative examples by 2.62%.

## 2 Problem Formulation

We formally define the author representativeness problem as follows:

**Input:** A Twitter author $U$ and the collection of their tweets, and a new tweet $T$.
**Problem:** Compute score$(T, U)$, the probability that $T$ was written by $U$. This score quantifies how characteristic of writer $U$, tweet $T$ is.

### 2.1 Methodology

In order to obtain this representativeness score, we model our task as a classification problem, where we seek to distinguish tweets from $U$ against tweets from all other users.

By modeling this as a binary classification problem, it becomes possible to quantify how characteristic of a writer a tweet is, as a probability implied by its distance from the decision boundary. Thus, we obtain a characterization score between 0 and 1 for each tweet.

**Challenges:** In training a classifier to distinguish between user and non-user tweets, we should ideally have an equal amount of examples of both classes. User tweets are simply all the tweets from that user's Twitter account, and measure perhaps in the thousands. Indeed, the number of tweets per user per day is limited to 2400 per day by current Twitter policy (https://help.twitter.com/en/rules-and-policies/twitter-limits). The negative examples consist of all tweets written by other Twitter users, a total of approximately 500 million per day

(https://business.twitter.com). Thus there is an extreme class imbalance between user and non-user tweets. Moreover, the nature of language used on Twitter does not conform to formal syntactic or semantic rules. The sentences tend to be highly unstructured, and the vocabulary is not restricted to a particular dictionary.

### 2.2 Data

For the binary classification task described in Section 2.1, we term tweets from $U$ as positive examples, and tweets from other users as negative examples.

- **Positive examples:** We take tweets written by 15 celebrities from various domains, from 01-Jan-2008 to 01-Dec-2018, as positive examples. Properties of these Twitter celebrities are provided in Table 1.

- **Negative examples:** We have collected 1% of tweets from Twitter's daily feed using the Twitter API (https://developer.twitter.com/en/docs.html) to use as negative examples.

| User | Tweet count (Before) | Tweet count (After) | Domain | Foll. |
|---|---|---|---|---|
| Amitabh Bachchan | 49437 | 10342 | Acting | 37.0 |
| Ariana Grande | 37738 | 13657 | Music | 62.3 |
| Barack Obama | 11350 | 6772 | Politics | 106.0 |
| Bill Gates | 2699 | 1754 | Business | 47.1 |
| Donald Trump | 35549 | 18295 | Politics | 59.9 |
| Ellen DeGeneres | 17317 | 9616 | TV | 77.6 |
| J K Rowling | 6037 | 2634 | Author | 14.7 |
| Jimmy Fallon | 10698 | 3596 | TV | 51.1 |
| Justin Bieber | 18044 | 5193 | Acting | 105.0 |
| Kevin Durant | 22532 | 4146 | Sports | 17.5 |
| Kim Kardashian | 24541 | 7943 | Modeling | 60.5 |
| Lady Gaga | 7239 | 3767 | Music | 78.5 |
| LeBron James | 5145 | 2102 | Sports | 42.6 |
| Narendra Modi | 17613 | 6672 | Politics | 47.0 |
| Oprah Winfrey | 11685 | 4588 | TV | 42.2 |

Table 1: Twitter celebrities in our dataset, with tweet counts before and after filtering (Foll. denotes followers in millions)

**Preprocessing and Filtering:** We have preprocessed and filtered the data to remove tweets that are unrepresentative or too short for analysis. All text has been converted to lowercase, and stripped of punctuation marks and URLs. This is because our approaches are centered around word usage. However, in future models, punctuation may prove effective as a feature. Further, we restrict analysis to English language tweets containing no attached images. We select only tweets which are more than 10 words long, and contain at least 5 legitimate (dictionary) English words. We define an unedited transfer of an original tweet as a retweet, and remove these from our dataset. Since comments on retweets are written by the user themselves, we retain these in our dataset.

We note that celebrity Twitter accounts can be handled by PR agencies, in addition to the owner themselves. Because our aim is to characterize Twitter profiles as entities, we have not attempted to distinguish between user-written and agency-written tweets. However, this is an interesting direction for future research.

We use a train-test split of 70-30% on the positive examples, and generate negative training and test sets of the same sizes for each user, by randomly sampling from the large set of negative examples.

## 3 Related work

### 3.1 Author identification and verification

The challenge of author identification has a long history in NLP. PAN 2013 (Juola and Stamatatos, 2013) introduced the question: "Given a set of documents by the same author, is an additional (out-of-set) document also by that author?" The corpus is comprised of text pieces from textbooks, newspaper articles, and fiction. Submissions to PAN 2014 (Stamatatos et al., 2014) also model authorship verification as binary classification, by using non-author documents as negative examples. The best submission (Seidman, 2013) in PAN 2013 uses the General Impostors (GI) method, which is a modification of the Impostors Method (Koppel et al., 2012). The best submission (Khonji and Iraqi, 2014) in PAN 2014 presents a modification of the GI method. These methods are based on the impostors framework (Koppel and Winter, 2014).

Veenman and Li (2013) used compression distance as a document representation, for authorship verification in PAN 2013. HB et al. (2015) present a global feature extraction approach and achieve state-of-the-art accuracy for the PAN 2014 corpus. The best submission (Bagnall, 2015) in PAN 2015 (Stamatatos et al., 2015) uses a character-level RNN model for author identification, in which each author is represented as a sub-model, and the

recurrent layer is shared by all sub-models. This is useful if the number of authors is fixed, and the problem is modeled as multi-class classification. Mohsen et al. (2016) also approach multi-class author identification, using deep learning for feature extraction, and Nirkhi et al. (2016) using hierarchical clustering.

Potha and Stamatatos (2018) propose an intrinsic profile-based verification method that uses latent semantic indexing (LSI), which is effective for longer texts. Koppel and Schler (2004) and Luyckx and Daelemans (2008) explore methods for authorship verification for larger documents such as essays and novels. Nizamani and Memon (2013) and Brocardo et al. (2013) explore author identification for emails, and Chen and Sun (2017) for scientific papers. Azarbonyad et al. (2015) make use of temporal changes in word usage to identify authors of tweets and emails. Fissette (2010), Green and Sheppard (2013), and Zhang et al. (2014) evaluate the utility of various features for this task. Stamatatos (2008) proposes text sampling to address the lack of text samples of undisputed authorship, to produce a desirable distribution over classes.

Koppel et al. (2009) compare methods for variants of the authorship attribution problem. Bhargava et al. (2013) apply stylometric analysis to tweets to determine the author. López-Monroy et al. (2015) propose a document representation capturing discriminative and subprofile-specific information of terms. Rocha et al. (2016) review methods for authorship attribution for social media forensics. Peng et al. (2016a) use bit-level n-grams for determining authorship for online news. Peng et al. (2016b) apply this method to detect astroturfing on social media. Theóphilo et al. (2019) employ deep learning specifically for authorship attribution of short messages.

### 3.2 Predicting tweet popularity

Suh et al. (2010) leverages features such as URL, number of hashtags, number of followers and followees etc. in a generalized linear model, to predict the number of retweets. Naveed et al. (2011) extend this approach to perform content-based retweet prediction using several features including sentiments, emoticons, punctuations etc. Bandari et al. (2012) apply the same approach for regression as well as classification, to predict the number of retweets specifically for news articles. Zaman et al. (2014) present a Bayesian model for retweet prediction using early retweet times, retweets of other tweets, and the user's follower graph. Tan et al. (2014) analyze whether different wording of a tweet by the same author affects its popularity. SEISMIC (Zhao et al., 2015) and PSEISMIC (Chen and Li, 2017) are statistical methods to predict the final number of retweets. Zhang et al. (2018) approach retweet prediction as a multi-class classification problem, and present a feature-weighted model, where weights are computed using information gain.

### 3.3 Training with imbalanced datasets

Various methods to handle imbalanced datasets have been described by Kotsiantis et al. (2006). These include undersampling (Kotsiantis and Pintelas, 2003), oversampling, and feature selection (Zheng et al., 2004) at the data level. However, due to random undersampling, potentially useful samples can be discarded, while random oversampling poses the risk of overfitting. This problem can be handled at the algorithmic level as well: the threshold method (Weiss, 2004) produces several classifiers by varying the threshold of the classifier score. One-class classification can be performed using a divide-and-conquer approach, to iteratively build rules to cover new training instances (Cohen, 1995). Cost-sensitive learning (Domingos, 1999) uses unequal misclassification costs to address the class imbalance problem.

## 4 Approaches to authorship verification

As described in Section 2.1, we build classification models to distinguish between user and non-user tweets. We have explored five distinct approaches to build such models.

### 4.1 Approach 1: Compression

This approach is inspired from Kolmogorov complexity (Li and Vitányi, 2013), which argues that the compressibility of a text reflects the quality of the underlying model. We use the Lempel-Ziv-Welch (LZW) compression algorithm (Welch, 1984) to approximate Kolmogorov complexity by dynamically building a dictionary to encode word patterns from the training corpus. The longest occurring pattern match present in the dictionary is used to encode the text.

We hypothesize that the length of a tweet $T$ from user $U$, compressed using a dictionary built

from positive examples, will be less than the length of the same tweet compressed using a dictionary built from negative examples.

We use the following setup to classify test tweets for each Twitter user in our dataset:

1. Build an encoding dictionary using positive examples ($\text{train}_{\text{pos}}$), and an encoding dictionary using negative examples ($\text{train}_{\text{neg}}$).

2. Encode the new tweet $T$ using both these dictionaries, to obtain $T_{\text{pos}} = \text{encode}_{\text{pos}}(T)$ and $T_{\text{neg}} = \text{encode}_{\text{neg}}(T)$ respectively.

3. If the length of $T_{\text{pos}}$ is less than that of $T_{\text{neg}}$, classify $T$ as positive; else, classify it as negative.

This gives us the class label for each new tweet $T$. In addition, we compute the characterization score of tweet $T$ with respect to user $U$, as described in Equation 1.

$$\text{score}(T, U) = 1 - \frac{\text{len}(T_{\text{pos}})}{\text{len}(T)} \quad (1)$$

Thus the shorter the length of the encoded tweet, the more characteristic of the user $T$ is.

### 4.2 Approach 2: Topic modeling

We hypothesize that each user writes about topics with a particular probability distribution, and that each tweet reflects the probability distribution over these topics. We train a topic model using Latent Dirichlet Allocation (LDA) (Blei et al., 2003) on a large corpus of tweets, and use this topic model to compute topic distributions for individual tweets. We then use these values as features. We experiment with two types of classifiers: Logistic Regression (LR), and Multi Linear Perceptron (MLP) of size $(5, 5, 5)$. We represent each tweet as a distribution over $n = 500$ topics.

The characterization score of a tweet $T$ is given by the classifier's confidence that $T$ belongs to the positive class.

### 4.3 Approach 3: n-gram probability

We hypothesize that a Twitter user can be characterized by usage of words and their frequencies in tweets, and model this using n-gram frequencies.

We use the following setup to classify test tweets for each Twitter user in our dataset:

1. Build a frequency dictionary of all n-grams in positive examples ($\text{train}_{\text{pos}}$), and a frequency dictionary of all n-grams in negative examples ($\text{train}_{\text{neg}}$).

2. Compute the average probability of all n-gram sequences in the new tweet $T$ using both these dictionaries, to obtain $\text{prob}_{\text{pos}}(T)$ and $\text{prob}_{\text{neg}}(T)$ respectively. Here, we use add-one smoothing and conditional backoff to compute these probability values.

3. If $\text{prob}_{\text{pos}}(T)$ is greater than $\text{prob}_{\text{neg}}(T)$, classify $T$ as positive; else, classify it as negative.

The characterization score of tweet $T$ is given by the average n-gram probability computed using the frequency dictionary of $\text{train}_{\text{pos}}$. We experiment with $n = 1$ (unigrams) and $n = 2$ (bigrams).

### 4.4 Approach 4: Document embeddings

We hypothesize that if we obtain latent representations of tweets as documents, tweets from the same author will cluster together, and will be differentiable from tweets from others. To that end, we use the following setup:

1. We obtain representations of tweets as document embeddings. We experiment with two types of document embeddings: FastText (Facebook-Research, 2016) (embedding size = 100) and BERT-Base, uncased (Devlin et al., 2018) (embedding size = 768).

2. We then use these embeddings as features to train a classification model. We experiment with two types of classifiers: Logistic Regression (LR) and Multi Linear Perceptron (MLP) of size $(5, 5, 5)$.

The characterization score of tweet $T$ is given by the classifier's confidence that $T$ belongs to the positive class.

**Iterative sampling:** As described in Section 2.1, there exists an extreme class imbalance for this binary classification task, in that the number of negative examples is far more than the number of positive examples. Here, we explore an iterative sampling technique to address this problem. We train our classifier for multiple iterations, coupling the same $\text{train}_{\text{pos}}$ with a new randomly sampled $\text{train}_{\text{neg}}$ set in each iteration.

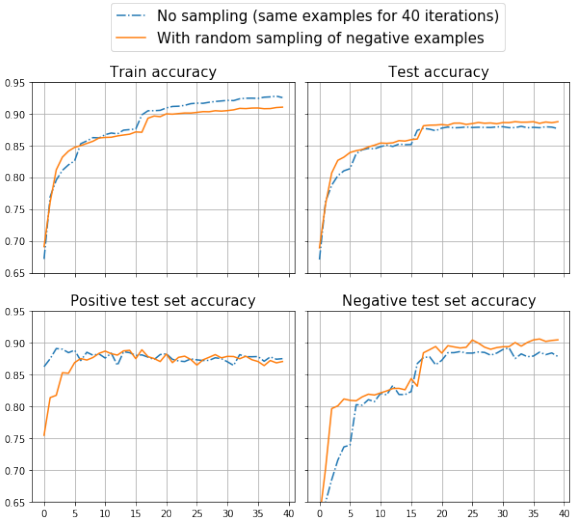


Figure 2: Mean accuracy of the BERT + MLP classifier for all users over 40 iterations

We conduct this experiment for all users with the best performing model for this approach, i.e. we use BERT embeddings as features, and MLP for classification. We train this classifier for 40 iterations, and compare the model's performance when we use the same set of negative examples vs. when we randomly sample new negative examples in each iteration.

Figure 2 shows the mean train and test accuracy for all users over 40 iterations. As expected, the training accuracy is higher if we do not sample, as the model gets trained on the same data repeatedly in each iteration. However, if we perform random sampling, the model is exposed to a larger number of negative examples, which results in a higher test accuracy (+ 1.08%), specifically for negative test examples (+ 2.62%).

### 4.5 Approach 5: Token embeddings and sequential modeling

In this approach, we tokenize each tweet, and obtain embeddings for each token. We then sequentially give these embeddings as input to a classifier.

We use a pretrained model (BERT-Base, Uncased: 12-layer, 768-hidden, 12-heads, 110M parameters) to generate token embeddings of size 768, and pass these to a Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) classifier. We use an LSTM layer with 768 units with dropout and recurrent dropout ratio 0.2, followed by a dense layer with sigmoid activation.

| User | (1) Compression | (2) LDA + LR | (2) LDA + MLP | (3) Bigram | (3) Unigram | (4) FT + LR | (4) FT + MLP | (4) BERT + LR | (4) BERT + MLP | (5) BERT + LSTM |
|---|---|---|---|---|---|---|---|---|---|---|
| Amitabh Bachchan | 72.93 | 69.47 | 74.91 | 84.45 | 90.16 | 84.58 | 87.13 | 93.59 | 93.73 | **96.32** |
| Ariana Grande | 71.98 | 76.76 | 80.57 | 73.89 | 85.62 | 84.00 | 85.28 | 87.36 | 87.98 | **90.20** |
| Barack Obama | 78.85 | 80.09 | 87.20 | 82.47 | 92.58 | 91.75 | 93.06 | 95.28 | 95.57 | **96.57** |
| Bill Gates | 74.29 | 70.78 | 81.78 | 81.21 | 87.10 | 86.34 | 83.97 | 91.56 | 92.41 | **92.41** |
| Donald Trump | 72.11 | 77.38 | 81.91 | 77.68 | 89.70 | 87.72 | 88.88 | 90.94 | 91.62 | **93.40** |
| Ellen DeGeneres | 70.75 | 69.39 | 74.53 | 71.90 | 84.40 | 81.27 | 83.11 | 87.38 | 88.60 | **91.12** |
| J K Rowling | 63.39 | 64.12 | 70.84 | 71.27 | 77.08 | 79.40 | **80.75** | 79.34 | 80.68 | 79.71 |
| Jimmy Fallon | 67.39 | 72.04 | 73.89 | 78.41 | 85.59 | 82.25 | 83.06 | 85.82 | 86.95 | **88.62** |
| Justin Bieber | 73.16 | 70.99 | 79.84 | 75.06 | 85.72 | 85.50 | 86.76 | 89.53 | 89.92 | **92.89** |
| Kevin Durant | 68.78 | 76.56 | 80.24 | 74.61 | 86.21 | 85.76 | 86.93 | 84.75 | 85.84 | **88.62** |
| Kim Kardashian | 69.73 | 72.10 | 76.39 | 71.49 | 83.89 | 80.92 | 82.49 | 84.12 | 85.25 | **88.35** |
| Lady Gaga | 66.01 | 67.07 | 72.40 | 71.44 | 81.14 | 76.91 | 79.90 | 81.46 | 83.21 | **84.54** |
| LeBron James | 67.95 | 66.21 | 73.33 | 74.17 | 82.42 | 81.97 | 77.05 | 83.48 | 84.77 | **85.53** |
| Narendra Modi | 82.78 | 84.40 | 89.51 | 90.75 | 94.39 | 94.69 | 95.71 | 97.21 | **97.41** | 97.33 |
| Oprah Winfrey | 68.61 | 61.74 | 70.47 | 75.37 | 83.88 | 83.69 | 83.51 | 86.37 | 87.07 | **90.01** |
| **Mean** | 71.25 | 71.94 | 77.86 | 76.95 | 85.99 | 84.45 | 85.17 | 87.88 | 88.73 | **90.37** |

Table 2: Test accuracy (%) of five approaches to classify user vs. non-user tweets (The best performing approach is shown in **bold** for each user) [Note that for each user, the test set contains an equal number of positive and negative examples.]

We train this model using the Adam optimizer (Kingma and Ba, 2014) and binary cross-entropy loss, with accuracy as the training metric.

### 4.6 Results and Comparison

Table 2 presents the user-wise test accuracy of the five approaches under the specified configurations. Note that the test set contains an equal number of positive and negative examples for each author.

Other baselines that we attempted to compare against include the best submissions to the PAN 2013 and 2014 author verification challenge: Seidman (2013) and Khonji and Iraqi (2014), which are variants of the Impostors Method. This challenge employed significantly longer documents (with an average of 1039, 845, and 4393 words per document for articles, essays and novels respectively, as opposed to an average of 19 words per tweet) and significantly fewer documents per author (an average of 3.2, 2.6 and 1 document/s per author, as opposed to an average of 6738 tweets per user). Our experiments with the authorship verification classifier (Eder et al., 2016) showed that the Impostors Method is prohibitively expensive on larger corpora, and also performed too inaccurately on short texts to provide a meaningful baseline.

For 13 of the 15 users in our dataset, Approach 4.5 (token embeddings followed by sequential modeling) has the highest accuracy. This model correctly identifies the author of 90.37% of all tweets in our study, and will be used to define the characterization score for our subsequent studies.

## 5 User study

To verify whether human evaluators are in agreement with our characterization model, we conducted a user study using MTurk (Amazon, 2005).

### 5.1 Setup

For each user in our dataset, we build a set of 20 tweet pairs, with one tweet each from the 50 top-scoring and bottom-scoring tweets written by the user. We ask the human evaluator to choose which tweet sounds more characteristic of the user. To validate that the MTurk worker knows enough about the Twitter user to pick a characteristic tweet, we use a qualification test containing a basic set of questions about the Twitter user. We were unable to find equal numbers of Turkers familiar with each subject, so our number of evaluators $n$ differs according to author.

### 5.2 Results

Table 3 describes the results obtained in the user study: the mean and standard deviation of percentage of answers in agreement with our model, the p-value, and the number of MTurk workers who completed each task. We find that the average agreement of human evaluators with our model is 70.40% over all 15 users in our dataset.

| User | Mean(%) | $\sigma$(%) | p-value | $n$ |
|---|---|---|---|---|
| Amitabh Bachchan | 67.08 | 16.44 | **1.30e-07** | 12 |
| Ariana Grande | 67.19 | 24.01 | **7.60e-10** | 16 |
| Barack Obama | 55.75 | 17.04 | 2.43e-02 | 20 |
| Bill Gates | 70.26 | 14.19 | **1.72e-15** | 19 |
| Donald Trump | 83.85 | 8.87 | **2.52e-58** | 26 |
| Ellen DeGeneres | 73.75 | 14.22 | **5.44e-22** | 20 |
| J K Rowling | 65.79 | 10.04 | **7.51e-10** | 19 |
| Jimmy Fallon | 80.00 | 21.93 | **5.76e-25** | 14 |
| Justin Bieber | 71.94 | 22.57 | **3.97e-17** | 18 |
| Kevin Durant | 64.38 | 15.04 | **3.04e-07** | 16 |
| Kim Kardashian | 71.25 | 14.95 | **9.27e-18** | 20 |
| Lady Gaga | 85.00 | 10.31 | **1.45e-22** | 9 |
| LeBron James | 63.50 | 12.15 | **7.38e-08** | 20 |
| Narendra Modi | 60.45 | 13.68 | 2.34e-03 | 11 |
| Oprah Winfrey | 75.79 | 18.12 | **1.25e-24** | 19 |

Table 3: MTurk user study results: For each of these 15 celebrities, human evaluators support our representativeness scores with a significance level above 0.05. (p-values $< 10^{-5}$ are shown in **bold**.)

For each of the 15 celebrities, the human evaluators agree with our model above a significance level of 0.05, and in 13 of 15 cases above a level of $10^{-5}$. This makes clear our scores are measuring what we intend to be measuring.

## 6 Mapping with popularity

### 6.1 Correlation

We now explore the relationship between characterization score and tweet popularity for each of the users in our dataset. To analyze this relationship, we perform the following procedure for each author $U$:

1. Sort all tweets written by $U$ in ascending order of characterization score.
2. Bucket the sorted tweets by percentile score (1 to 100).
3. For each bucket, calculate the mean number of likes, replies, and retweets.

4. Compute the correlation of this mean and the percentile score.

| User | Likes | Replies | Retweets |
|---|---|---|---|
| Donald Trump | **0.64** | **0.63** | **0.55** |
| Amitabh Bachchan | **0.58** | **0.81** | **0.69** |
| Narendra Modi | **0.46** | 0.01 | 0.22 |
| Jimmy Fallon | **0.29** | **0.54** | **0.41** |
| J K Rowling | 0.21 | **0.32** | 0.14 |
| Lady Gaga | 0.05 | 0.12 | -0.01 |
| Bill Gates | -0.05 | -0.11 | -0.21 |
| LeBron James | -0.22 | **-0.27** | -0.24 |
| Oprah Winfrey | **-0.30** | **-0.41** | -0.17 |
| Ellen DeGeneres | **-0.34** | **-0.29** | **-0.40** |
| Barack Obama | **-0.45** | **-0.46** | **-0.45** |
| Kevin Durant | **-0.57** | **-0.67** | **-0.53** |
| Kim Kardashian | **-0.71** | **-0.72** | **-0.70** |
| Justin Bieber | **-0.73** | **-0.50** | **-0.71** |
| Ariana Grande | **-0.74** | **-0.77** | **-0.75** |

Table 4: Pearson correlation coefficients between mean popularity measure and percentile, for each user (Coefficients with p-value $< 0.01$ are shown in **bold** color). Green values exhibit significant positive correlation, and red values significant negative correlation.

The Pearson correlation coefficients (r-values) are listed in Table 4. The users at the top (Trump, Bachchan, Modi) all display very strong positive correlation. We name this group UPC (Users with Positive Correlation), and the group of users at the bottom (Grande, Bieber, Kardashian) as UNC (Users with Negative Correlation).

## 6.2 Interpretation

For users with positive correlation, the higher the tweet's characterization score, the more popular it becomes, i.e. the more likes, replies, and retweets it receives. In contrast, for users with negative correlation, the higher the tweet score, the less popular it becomes.

Figure 3 shows the plot of log mean number of likes per bucket vs. tweet score percentile, for users with the highest positive correlation. Similarly, Figure 4 shows the plot of log mean number of likes per bucket vs. tweet score percentile, for users with the highest negative correlation.

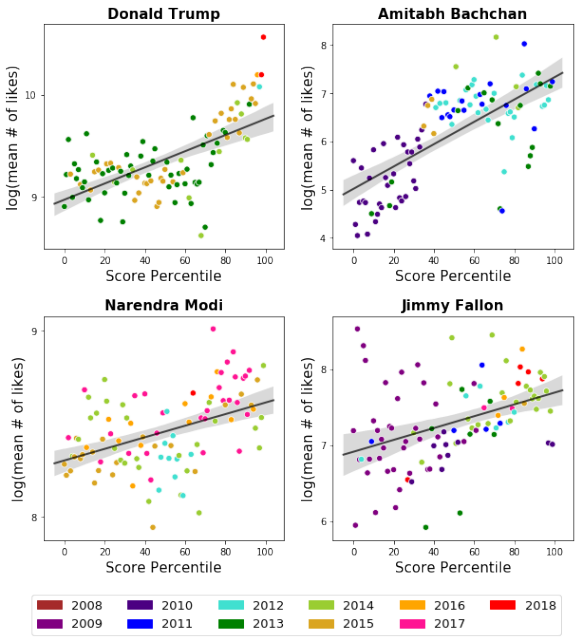


Figure 3: Log mean likes vs. percentile for users of positive correlation (The color denotes the year for which maximum tweets are present in the percentile bucket).

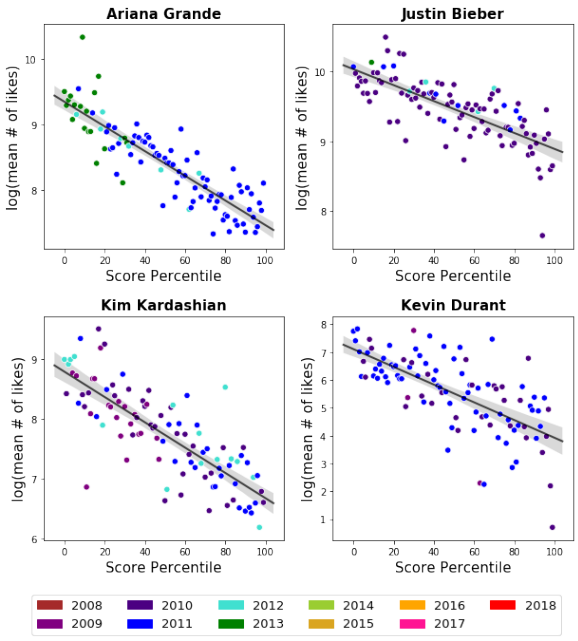


Figure 4: Log mean likes vs. percentile for users of negative correlation (The color denotes the year for which maximum tweets are present in the percentile bucket).

One may question whether these results are due to temporal effects: user's popularity vary with time, and perhaps the model's more characteristic tweets simply reflect periods of authorship. Figures 3 and 4 disprove this hypothesis.

Here the color of each point denotes the year for which most tweets are present in the corresponding bucket. Since the distribution of colors over time is not clustered, we infer that the observed result is not an artifact of temporal effects. In both cases, there is a strong trend in tweet popularity based on tweet score. We note that the plots are presented on the log scale, meaning the trends here are exponential.

### 6.3 Qualitative Analysis

We present examples of the most and least characteristic tweets for celebrities from three categories, along with their corresponding characterization scores computed using Approach 4.5.

#### 6.3.1 Users with Positive Correlation (UPC)

**Donald Trump**

| Tweet | Score |
|---|---|
| Prior to the election it was well known that I have interests in properties all over the world. Only the crooked media makes this a big deal! | 0.9998 |
| Today is the first day of the rest of your life - make the most of it! | 0.0001 |

**Amitabh Bachchan**

| Tweet | Score |
|---|---|
| T 2843 - The work is demanding .. the crew binding .. the city exciting .. and the dialogues expanding .. 'BADLA' is grinding .. !! | 0.9996 |
| hahaha .. now i dont have a HD .. but ya a car ride is on .. | 0.0002 |

The characterization score appears to have correctly captured aspects of the user's personality from their corpus of tweets. For these celebrities, high scoring tweets generally prove more popular (In this example - Donald Trump: 70.5K vs. 693 likes; Amitabh Bachchan: 7.1K vs. 9 likes), as reflected in their positive correlation coefficients.

#### 6.3.2 Users with Negative Correlation (UNC)

**Ariana Grande**

| Tweet | Score |
|---|---|
| Finalizing the set list for Fresno! Getting so excited.. Can't believe the show is already almost sold out, you guys are amazing. Xoxo! | 0.9997 |
| The first thing I do when I get to a new city is look up how close the nearest Whole Foods is. | 0.0002 |

**Justin Bieber**

| Tweet | Score |
|---|---|
| grateful to everyone who came out and to my band, dancers, and whole crew. The energy last night was incredible and cant wait to tour | 0.9999 |
| Less cantaloupe, more berries. I'm talking to you, pre-packaged fruit salads. Don't play me like that. | 0.00002 |

Again, high scoring tweets appear more characteristic of their respective users. But here, low scoring tweets are generally more popular (In this example - Ariana Grande: 622 vs. 2.4K likes; Justin Bieber: 454 vs. 13.8K likes), as reflected in their negative correlation coefficients.

#### 6.3.3 Users with no significant correlation

**Bill Gates**

| Tweet | Score |
|---|---|
| I recently visited a lab doing super-cool energy work-a good reminder of why governments should sponsor R&D | 0.9986 |
| There's a lot of green on this map-which is good-but still not enough. | 0.0027 |

Here, tweets from extreme ends of the spectrum have similar content, so little variation can be expected in their popularity. For this celebrity, there is no significant correlation between characterization score and popularity.

## 7 Conclusions

We have presented and evaluated measures of binary author classification, to obtain a user-specific characterization score for each tweet. We demonstrate that sequential modeling on word embeddings yields the best result of 90.37% mean test accuracy, and that human evaluators are in agreement with our model 70.40% of the time. Our work demonstrates that representativeness scores correlate with popularity, and opens new research directions concerning virality on social media.

## Acknowledgments

We are grateful to the anonymous reviewers for their helpful feedback. We also thank Niranjan Balasubramanian and H. Andrew Schwartz for their comments and suggestions. This work was partially supported by NSF grants IIS-1546113 and IIS-1927227.

## References


Amazon. 2005. MTurk. (https://www.mturk.com/).

Hosein Azarbonyad, Mostafa Dehghani, Maarten Marx, and Jaap Kamps. 2015. Time-aware Authorship Attribution for Short Text Streams. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 727–730. ACM.

Douglas Bagnall. 2015. Author identification using multi-headed recurrent neural networks. *arXiv preprint arXiv:1506.04891*.

Roja Bandari, Sitaram Asur, and Bernardo A Huberman. 2012. The Pulse of News in Social Media: Forecasting Popularity. In *Sixth International AAAI Conference on Weblogs and Social Media*.

Mudit Bhargava, Pulkit Mehndiratta, and Krishna Asawa. 2013. Stylometric Analysis for Authorship Attribution on Twitter. In *International Conference on Big Data Analytics*, pages 37–47. Springer.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022.

Marcelo Luiz Brocardo, Issa Traore, Sherif Saad, and Isaac Woungang. 2013. Authorship verification for short messages using stylometry. In *2013 International Conference on Computer, Information and Telecommunication Systems (CITS)*, pages 1–6. IEEE.

Hsin-Yu Chen and Cheng-Te Li. 2017. PSEISMIC: A Personalized Self-Exciting Point Process Model for Predicting Tweet Popularity. In *2017 IEEE International Conference on Big Data*, pages 2710–2713. IEEE.

Ting Chen and Yizhou Sun. 2017. Task-Guided and Path-Augmented Heterogeneous Network Embedding for Author Identification. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 295–304. ACM.

William W Cohen. 1995. Fast effective rule induction. In *Machine Learning Proceedings 1995*, pages 115–123. Elsevier.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*.

Pedro Domingos. 1999. MetaCost: A General Method for Making Classifiers Cost-Sensitive. In *KDD*, volume 99, pages 155–164.

Maciej Eder, Jan Rybicki, and Mike Kestemont. 2016. Stylometry with R: a Package for Computational Text Analysis. *R journal*, 8(1).

Facebook-Research. 2016. FastText. (https://research.fb.com/fasttext/).

MVM Fissette. 2010. Author Identification in Short Texts.

Rachel M Green and John W Sheppard. 2013. Comparing frequency-and style-based features for twitter author identification. In *FLAIRS Conference*.

Barathi Ganesh HB, U Reshma, et al. 2015. Author identification based on word distribution in word space. In *2015 ICACCI*, pages 1519–1523. IEEE.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-term Memory. *Neural computation*, 9(8):1735–1780.

Patrick Juola and Efstathios Stamatatos. 2013. Overview of the Author Identification Task at PAN 2013. In *CLEF (Working Notes)*.

Mahmoud Khonji and Youssef Iraqi. 2014. A Slightly-modified GI-based Author-verifier with Lots of Features (ASGALF). *CLEF (Working Notes)*, 1180:977–983.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*.

Moshe Koppel and Jonathan Schler. 2004. Authorship verification as a one-class classification problem. In *Proceedings of the Twenty-First International Conference on Machine learning*, page 62. ACM.

Moshe Koppel, Jonathan Schler, and Shlomo Argamon. 2009. Computational Methods in Authorship Attribution. *Journal of the American Society for information Science and Technology*, 60(1):9–26.

Moshe Koppel, Jonathan Schler, Shlomo Argamon, and Yaron Winter. 2012. The "Fundamental Problem" of Authorship Attribution. *English Studies*, 93(3):284–291.

Moshe Koppel and Yaron Winter. 2014. Determining if Two Documents Are Written by the Same Author. *Journal of the Association for Information Science and Technology*, 65(1):178–187.

SB Kotsiantis and PE Pintelas. 2003. Mixture of Expert Agents for Handling Imbalanced Data Sets. *Annals of Mathematics, Computing & Teleinformatics*, 1(1):46–55.

Sotiris Kotsiantis, Dimitris Kanellopoulos, Panayiotis Pintelas, et al. 2006. Handling imbalanced datasets: A review. *GESTS International Transactions on Computer Science and Engineering*, 30(1):25–36.

Ming Li and Paul Vitányi. 2013. *An introduction to Kolmogorov complexity and its applications*. Springer Science & Business Media.

A Pastor López-Monroy, Manuel Montes-y Gómez, Hugo Jair Escalante, Luis Villaseñor-Pineda, and Efstathios Stamatatos. 2015. Discriminative subprofile-specific representations for author profiling in social media. *Knowledge-Based Systems*, 89:134–147.

Kim Luyckx and Walter Daelemans. 2008. Authorship Attribution and Verification with Many Authors and Limited Data. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 513–520. ACL.

Ahmed M Mohsen, Nagwa M El-Makky, and Nagia Ghanem. 2016. Author Identification using Deep Learning. In *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 898–903. IEEE.

Nasir Naveed, Thomas Gottron, Jérôme Kunegis, and Arifah Che Alhadi. 2011. Bad News Travel Fast: A Content-based Analysis of Interestingness on Twitter. In *Proceedings of the 3rd International Web Science Conference*, page 8. ACM.

Smita Nirkhi, RV Dharaskar, and VM Thakare. 2016. Authorship Verification of Online Messages for Forensic Investigation. *Procedia Computer Science*, 78:640–645.

Sarwat Nizamani and Nasrullah Memon. 2013. CEAI: CCM-based email authorship identification model. *Egyptian Informatics Journal*, 14(3):239–249.

Jian Peng, Kim-Kwang Raymond Choo, and Helen Ashman. 2016a. Bit-level n-gram based forensic authorship analysis on social media: Identifying individuals from linguistic profiles. *Journal of Network and Computer Applications*, 70:171–182.

Jian Peng, Raymond Kim-Kwang Choo, and Helen Ashman. 2016b. Astroturfing Detection in Social Media: Using Binary n-Gram Analysis for Authorship Attribution. In *2016 IEEE Trustcom/BigDataSE/ISPA*, pages 121–128. IEEE.

Nektaria Potha and Efstathios Stamatatos. 2018. Intrinsic Author Verification Using Topic Modeling. In *Proceedings of the 10th Hellenic Conference on Artificial Intelligence*, page 20. ACM.

Anderson Rocha, Walter J Scheirer, Christopher W Forstall, Thiago Cavalcante, Antonio Theophilo, Bingyu Shen, Ariadne RB Carvalho, and Efstathios Stamatatos. 2016. Authorship Attribution for Social Media Forensics. *IEEE Transactions on Information Forensics and Security*, 12(1):5–33.

Shachar Seidman. 2013. Authorship Verification Using the Impostors Method. In *CLEF 2013 Evaluation Labs and Workshop-Online Working Notes*. Citeseer.

Efstathios Stamatatos. 2008. Author identification: Using text sampling to handle the class imbalance problem. *Information Processing & Management*, 44(2):790–799.

Efstathios Stamatatos, Walter Daelemans, Ben Verhoeven, Patrick Juola, Aurelio López-López, Martin Potthast, and Benno Stein. 2015. Overview of the Author Identification Task at PAN 2015. In *CLEF 2015 Evaluation Labs and Workshop – Working Notes Papers*, Toulouse, France. CEUR.

Efstathios Stamatatos, Walter Daelemans, Ben Verhoeven, Martin Potthast, Benno Stein, Patrick Juola, Miguel A Sanchez-Perez, and Alberto Barrón-Cedeño. 2014. Overview of the Author Identification Task at PAN 2014. In *CLEF 2014 Evaluation Labs and Workshop Working Notes Papers, Sheffield, UK, 2014*, pages 1–21.

Statista. 2019. Twitter: Number of active users 2010-2018. (https://www.statista.com/statistics/282087/number-of-monthly-active-twitter-users/).

Bongwon Suh, Lichan Hong, Peter Pirolli, and Ed H Chi. 2010. Want to be Retweeted? Large Scale Analytics on Factors Impacting Retweet in Twitter Network. In *2010 IEEE Second International Conference on Social Computing*, pages 177–184. IEEE.

Chenhao Tan, Lillian Lee, and Bo Pang. 2014. The effect of wording on message propagation: Topic- and author-controlled natural experiments on Twitter. *arXiv preprint arXiv:1405.1438*.

Antônio Theóphilo, Luís AM Pereira, and Anderson Rocha. 2019. A Needle in a Haystack? Harnessing Onomatopoeia and User-specific Stylometrics for Authorship Attribution of Micro-messages. pages 2692–2696.

Cor J Veenman and Zhenshi Li. 2013. Authorship Verification with Compression Features. In *CLEF (working notes)*.

Gary M Weiss. 2004. Mining with Rarity: a Unifying Framework. *ACM SIGKDD Explorations Newsletter*, 6(1):7–19.

Terry A. Welch. 1984. Technique for High-Performance Data Compression. *Computer*, 17.

Tauhid Zaman, Emily B Fox, Eric T Bradlow, et al. 2014. A Bayesian approach for predicting the popularity of tweets. *The Annals of Applied Statistics*, 8(3):1583–1611.

Chunxia Zhang, Xindong Wu, Zhendong Niu, and Wei Ding. 2014. Authorship identification from unstructured texts. *Knowledge-Based Systems*, 66:99–111.

Yang Zhang, Zhiheng Xu, and Qing Yang. 2018. Predicting Popularity of Messages in Twitter using a Feature-weighted Model. *http://ww.nlp.ia.ac.cn/2012papers/gjhy/gh154.pdf*, 20.

Qingyuan Zhao, Murat A Erdogdu, Hera Y He, Anand Rajaraman, and Jure Leskovec. 2015. SEISMIC: A Self-Exciting Point Process Model for Predicting Tweet Popularity. In *Proceedings of the 21th ACM SIGKDD*, pages 1513–1522. ACM.

Zhaohui Zheng, Xiaoyun Wu, and Rohini Srihari. 2004. Feature Selection for Text Categorization on Imbalanced Data. *ACM SIGKDD Explorations Newsletter*, 6(1):80–89.

# Finding Microaggressions in the Wild: A Case for Locating Elusive Phenomena in Social Media Posts

**Luke M. Breitfeller**[♠] **Emily Ahn**[♠] **David Jurgens**[♢] **Yulia Tsvetkov**[♠]
[♠]Carnegie Mellon University [♢]University of Michigan
{mbreitfe, eahn1, ytsvetko}@cs.cmu.edu, jurgens@umich.edu

## Abstract

Microaggressions are subtle, often veiled, manifestations of human biases. These uncivil interactions can have a powerful negative impact on people by marginalizing minorities and disadvantaged groups. The linguistic subtlety of microaggressions in communication has made it difficult for researchers to analyze their exact nature, and to quantify and extract microaggressions automatically. Specifically, the lack of a corpus of real-world microaggressions and well-defined criteria for annotating them have prevented researchers from addressing these problems at scale. In this paper, we devise a general but nuanced, computationally operationalizable typology of microaggressions based on a small subset of microaggression data that we have. We then create two datasets: one with examples of diverse types of microaggressions recollected by their targets, and another with gender-based microaggressions in public conversations on social media. We introduce a new, more objective criterion for annotation and an active-learning based procedure that increases the likelihood of surfacing posts containing microaggressions. Finally, we analyze the trends that emerge from these new datasets.

## 1 Introduction

Toxicity and offensiveness are not always expressed with toxic language. While a substantial community effort has rightfully focused on identifying, preventing, and mitigating overtly toxic, profane, and hateful language (Schmidt and Wiegand, 2017), offensiveness spans a far larger spectrum that includes comments with more implicit and subtle signals that are no less offensive (Jurgens et al., 2019). One significant class of subtle-but-offensive comments includes *microaggressions* (Sue et al., 2007, MAs), defined in Merriam-Webster as "a comment or action that subtly or often unconsciously expresses a prejudiced attitude toward a member of a marginalized group such as a racial minority." Though subtle, MAs have been shown to have lasting harmful impacts on their targets. Qualitative interviews suggest that the subtlety of MAs may cause even greater levels of situational stress than overt aggression (Sue, 2010; Nadal et al., 2014).

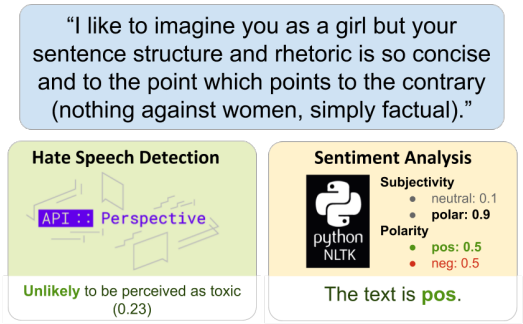


Figure 1: Existing state-of-the-art tools for hate speech detection and sentiment analysis cannot identify the veiled offensiveness of microaggressions (MAs) like the one in this real comment, because in many cases, the framing of a MA includes stylistic markers of positive language. This motivates the need for a corpus and new tools for detecting MAs.

Despite a public effort to recognize and reduce—if not eliminate—their occurrence (Kim, 2013; Neff, 2015), there has been no computational work to detect and *analyze* MAs *at scale*. Instead, much of the recent work has focused on explicitly toxic language (e.g., Waseem et al., 2017), with surveys of the area also overlooking this important and challenging task of recognizing this subtle toxicity (van Aken et al., 2018; Salminen et al., 2018; Fortuna and Nunes, 2018). Indeed, as Figure 1 suggests, current popular tools for toxic language detection do not recognize the toxicity of MAs and further, sentiment tools can label these comments as being positive. As a result, applications using such techniques to promote inoffensive content may potentially *promote* MAs. Such biased online content is then used



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1664–1674, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

as training data in NLP tools—dialogue systems, question answering, and others—thereby perpetuating and amplifying biases (Zhao et al., 2017).

Computational modeling of MAs is challenging for many reasons. MAs are subjective, context-sensitive, and expressed subtly in language (as shown in Figure 1); there are no well-defined annotation guidelines, no corpus of MAs for training and evaluating a model, and prior computational approaches to detecting overtly offensive speech are likely not suitable for classifying MAs. Moreover, they are diluted in the ocean of social media content, and it would be infeasible to create a corpus of diverse types of MAs just by crowdsourcing annotations of randomly sampled social media posts. This paper makes a first step towards closing this gap, providing (i) a typology of MAs, (ii) a corpus, and (iii) a first computational model to surfacing MAs.

Here, we introduce the first computational analysis of MAs with three main contributions. First, we create a seed corpus through collecting self-reported accounts of MAs online; we analyze this data using prior research in social science on MAs in order to derive an operationalizable classification scheme (§2.2), which we show can be used as training data to classify types of MAs (§2.4). Second, we develop a new active-learning based crowdsourcing technique to identify candidate microaggression messages for annotation (§3). In a large-scale study of gender-based MAs on Reddit, we show that our method is effective at surfacing MAs. Our novel approach surfaces MAs not found through simply looking for offensive posts (§3.3). Finally, we compare the types of MAs from self-reported accounts and those observed in social media comments, and we draw new conclusions about how MAs behave and may be identified in online settings (§4). [1]

## 2 A Corpus of Microaggression Classes

Previous examinations of MAs have typically focused on small scale studies with interviews, surveys, or discussions with clinicians (e.g., Shelton and Delgado-Romero, 2013; Campbell and Manning, 2014; Forrest-Bank and Jenson, 2015), aiming at understanding the causes and effects of MAs on their targets (Wong et al., 2014). A crucial gap in existing MAs research is the lack of systematic analysis of reports (Lilienfeld, 2017), which has contributed to the difficulty of understanding what constitutes a microagression. Here, we aim to address this gap by introducing a new dataset of MAs and then use this data and prior qualitative research to construct a new typology of MAs. This new typology enables us to ground the notion of what is a microagression, which also addresses criticism that the concept is too slippery. The typology is critical for downstream analyses of determining who is affected and how they are being targeted. Further, we use the typology as a practical heuristic for identifying which types of MAs are the highest impact for computational approaches. As a test, we develop a classifier using the typology and show that different types of MAs are readily distinguished.

### 2.1 SELFMA: microaggressions.com Dataset

The Tumblr website www.microaggressions.com curates a collection of self-reported accounts of MAs. They were collected by asking people to fill out an online form with three questions: the story of the microaggression, the context, and how the microaggression made them feel. As such, the microaggression accounts in this corpus come from individuals who self-report the incident, and who, presumably, are at least familiar with the term "microaggressions."

From this website, we collected 2,934 accounts of MAs (henceforth, *posts*). Most posts are manifestations of bias, targeting social groups frequently discriminated against, including gender (1,314 posts), race (1,278 posts), sexuality (461 posts), and religion (88 posts), among others. Some are formatted to show the conversation structure of the recollection of the incidents, while some others are shown as plain narratives. We refer to this dataset as SELFMA.

### 2.2 A New Typology of Microaggressions

As originally specified in Sue et al. (2007), MAs were categorized into 13 themes across specific axes of discrimination such as racism and sexism. These themes were developed qualitatively from transcripts with therapists. However, while they were comprehensive in the clinical domain, here, we aim to create a more general framework for classifying MAs in online interactions. Specifically, our new typology builds upon the work of Sue et al. (2007) with three goals in mind: the ty-

[1] All data and code produced through this work is available at https://bit.ly/2lVv3BG. This study was approved by the Institutional Review Board (IRB).

pology should be (1) generalizable across different axes of discrimination, (2) sufficiently represented in our corpus, and (3) comprehensive over all distinct microaggression types in the corpus.

Four key themes were identified in our analysis of the MAS data: Attributive, Institutionalized, Teaming, and Othering (see Table 1). We discuss each of these next.[2]

The **Attributive** theme covers instances where a microaggression attributes a stereotype to an individual based on their identity. These stereotypes may have inherently negative connotations ("lazy"), but may also be neutral ("liking pink") or positive ("strong"), which complements recent work on benevolent sexism (Jha and Mamidi, 2017). The **Institutionalized** theme reflects larger institutionalized biases, such as in employment or law enforcement. The **Teaming** theme is derived from the term *forced teaming*, coined by de Becker (1997) to describe a strategy of abuse where the abusers frames themselves as being on the same team as the victim. The **Othering** theme covers MAS which revolve around framing the target in relation to some "othered" group. This theme often co-exists with "Attribution of Stereotype," but is distinct in that its focus is on redefining the target's sense of identity.

### 2.3 Annotation Results

Three annotators familiar with the theoretical background and prior research on microagressions performed an open coding procedure after examining the SELFMA data to codify the typology and determine annotation guidelines. All three labeled 200 instances of the dataset to estimate agreement.

Despite the difficulty of the task, annotators had moderate agreement, as shown in Table 2. While lower than what is considered high agreement (Artstein and Poesio, 2008), given the potentially-subjective nature of MAS and criticism for lack of objectivity (Lilienfeld, 2017), we view this result as a strongly positive sign that reliable annotation is possible despite the challenge. This moderate agreement is on-par with other difficult annotation tasks, such as those for connotation frames (Rashkin et al., 2016), which had 0.52 percentage agreement for rating the polarity of the sentence towards a target, dimensions of social relationships (Rashid and Blanco, 2017), which had $\kappa$ values as low as 0.59, and Word Sense Disambiguation (Passonneau et al., 2012), which reported $\alpha$ values for some words below 0.30 at determining meaning.

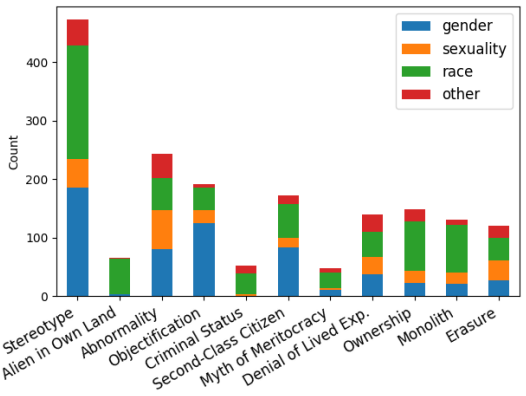


Figure 2: The distribution of four axes of discrimination (gender, race, sexuality, and other) in each subtheme.

The final dataset was determined by first retaining all posts where at least two annotators agreed (183 posts). And where there were no agreement (17 posts), the annotators determine the labels through a follow-up adjudication process.[3] After this process, another 1,100 posts were singly annotated, distributed across the three annotators, for a total of 1,300 posts. Relative percentages of each theme and examples comments are shown in Table 3. We also show the distribution of different axes of discrimination among the subthemes in Figure 2.

### 2.4 Classifying Types of Microaggressions

As a further examination of the typology, we perform a study where a model is trained to classify a comment into one of the four themes. Linear SVM classifiers were trained in a one-versus-rest setup using (i) word unigrams and bigrams on the tokenized version of the posts,[4] (ii) categories in the 2015 LIWC lexicon (Pennebaker et al., 2015), (iii) sentiment lexicons, and (iv) the topic distribution for a 40-topic LDA model (Blei et al., 2003). They were trained on 1,000 posts and evaluated using a held-out test set of 300 posts.

Classifier performance, measured as F1, was on

[2]The full description of themes, sub-themes, and examples can be found in Supplementary Material §2.

[3]There were 26 posts deemed to be exclusively overt aggressions (which could be detected using existing hate speech classifiers), and these occurred only in the training set. We removed them from our corpus.

[4]We used NLTK `wordpunct_tokenize` on the concatenated texts from the structured and unstructured format of the posts, normalizing two or more repeated characters into two. We only take those unigram and bigram features which appear in at least two posts as our feature set. This gives us 8,853 distinct features in the training data.

| | Sub-theme | Definition and Example |
|---|---|---|
| Attributive | Attribution of Stereotype | Link some attributes to an individual based on their identity.<br>"Girls just aren't good at math." |
| | Alien in Own Land | Marginalized individuals are foreign.<br>"But where are you from, originally?" |
| | Abnormality | Marginalized individuals are abnormal.<br>"Why do we need the word cisgender? That's just normal people." |
| Institutionalized | Objectification | Diminish the humanity of marginalized individuals.<br>"If you don't want to get hit on, wear a longer skirt." |
| | Criminal Status | Link a persons identity to criminality, danger, or illness.<br>"You look like a terrorist with that beard." |
| | Second-Class Citizen | Marginalized individuals belong to low-status positions in society.<br>"Oh, you work at an office? I bet you're a secretary." |
| Forced Teaming | Myth of Meritocracy | Differences in treatment are due to ones merit.<br>"They just cast actors who are best for it. Why does it matter if they're all white?" |
| | Denial of Lived Experience | Minimize the experiences of a marginalized individuals.<br>"It was just a joke! You're too sensitive." |
| | Ownership | Anyone can have some claim to a marginalized groups experiences.<br>"Why is it offensive for a white person to wear a bindi? It's just jewelry." |
| Othering | Monolith | All members of a marginalized group are identical.<br>"My gay friend doesn't have a problem with this show. I don't get why you're mad." |
| | Erasure | Anyone can claim that an individual does not belong to that group.<br>"Your mom is white, so it's not like you're really black, though." |

Table 1: Sub-themes that we developed based on the data from www.microaggressions.com, and short excerpts that show the main claim or assumption of each sub-theme. Examples demonstrate that MAs can harm and invalidate in a subtle way, typically targeting disadvantaged populations. The phenomenon is complex, different from overt profanity or negative sentiment.

| Theme | Fleiss' $\kappa$ |
|---|---|
| Attributive | 0.4156 |
| Institutionalized | 0.5058 |
| Teaming | 0.5048 |
| Othering | 0.4300 |
| *Overall* | 0.4641 |

Table 2: Inter-annotator agreement across three annotators on the four themes of microaggressions.

par with annotator agreement at 44.25. However, individual themes varied substantially in their performance, indicating not all were easily recognizable. Attributive and Institutionalized had the highest performance at 58.33 and 44.76, respectively; in contrast, Teaming and Othering had far lower F1 scores at 32.69 and 15.87, respectively. Both Teaming and Othering were the least frequent themes in the corpus, increasing the difficulty of recognizing them from limited data.

## 3 Locating Microaggressions in the Wild

The vast majority of microaggression data comes from self-reported recollections, including the MAs corpus that we collected. Self-reported data is crucial for studying the phenomenon and its effects, but is arguably not suitable for training classifiers, due to two critical limitations: (i) biases in how the offending comment was originally made versus how it was recalled, and (ii) biases in which kinds of microaggression comments are recalled. These make future computational attempts at detecting MAs more difficult by skewing a model away from linguistic signals used in real-world—rather than recalled—behavior.

A natural solution is to collect and annotate examples of MAs, scaling this process by using crowdsourcing. There are, however, two critical obstacles to using the straightforward approach of asking crowdworkers to identify random social media posts as MAs: (1) untrained crowdworkers are unlikely to reliably identify these posts due to the potential *subjectivity* of MAs, and (2) a naive random sampling of social media posts is unlikely to uncover MAs due to their *sparsity*, and since MAs are veiled manifestations of biases, it is also not clear how to use lexicons as a sieve for a stratified selection of posts to annotate.

In this second part of the paper, we devise a novel methodology to address the two challenges to curate microaggression corpora from social media posts with a two-pronged approach: operationalizing and surfacing MAs.

**Operationalizing Microaggressions** As the first study to crowdsource MAs, we focus on finding examples of gender-based MAs. By focusing

| Category | % | Examples | Toxicity |
|---|---|---|---|
| Attributive | 50 | For a female, you've got backbone. | .12 |
| | | There's fighting, for you boys, and romance, for you ladies | .10 |
| Institutionalized | 28 | So you make the coffee and he does the work? | .07 |
| | | Summer's coming! Gotta fit into that tight swimsuit. | .23 |
| Teaming | 23 | Most workplace harassment is an example of women being oversensitive to advances of men | .27 |
| | | So were you good enough to get in or did you get in through a women's quota? | .19 |
| Othering | 17 | I don't want another woman boss. Already have one and she has me running around all the time | .17 |

Table 3: The final microaggression typology after analyzing 1,300 self-reported microaggressions in SELFMA dataset. As shown by the percentages of instances covered each category, this typology effectively summarizes the broad themes (note that one instance may be labeled with multiple categories). For each category, the overall toxicity score (measured in [0,1] using the Google Perspective API) is low, indicating that currently none are reliably recognized as being offensive.

on gender-based MAs, the dominant group (male annotators) and the marginalized groups (women and non-binary annotators)[5] are easily elicited from the annotators through self-reporting. Since training crowdworkers to directly identify MAs is infeasible, we look for a more objective proxy that can identify, or at least help surface, posts containing MAs. A key property of MAs is that they often result from different lived experience between the individuals in the dominant group and those in the marginalized groups. Thus, our hypothesis is: *there will be a discrepancy of perceived offensiveness between the dominant group and the marginalized groups for* MAs. If true, we can then ask annotators to annotate the perceived offensiveness of a statement, and use this as a proxy to surface MAs from social media posts. Based on Figure 2, we pick gender as the discrimination axis for this work, since it covers a large number of our posts.

**Surfacing Microaggressions** To alleviate the sparsity issue, we treat the annotation process as an active learning procedure, where batches of posts to be annotated are provided by a classifier that is trained to predict the discrepancy of perceived offensiveness of a post, which we have claimed above to be a proxy to MAs. This classifier can then be iteratively updated using the results of all previous batches. Assuming the classifier is good enough in predicting the proxy, the classifier will help surface the most likely candidates of posts that contain MAs. In what follows, we describe the annotation process in detail and the resulting crowdsourced dataset.

### 3.1 Crowdsourcing Microaggressions

The full data collection process is set-up using a *multi-round annotation scheme*, summarized as follows. We first select a random sample of Reddit posts from relevant subreddits. These posts are then given to crowdworkers to be annotated for perceived offensiveness. This first round of annotations is then used to train a classifier for a MA-relavent task. The classifier's ratings are used to pick the next batch of Reddit posts to be annotated, in order to surface more instances than random sampling.

**Task** Crowdworkers are shown comments in the context of an internet forum discussion. Crowdworkers are asked to imagine they have written the displayed post, and then after a button click, we reveal how someone (a replier) has commented to that post. They are then asked rate if they found the response offensive and if so, to what degree was it offensive on a seven-point Likert scale. This scale of offensiveness allows us to capture cases where a microaggression is perceived as offensive by both annotator groups (genders), but to different degrees. To test our hypothesis and quantify how certain posts could be perceived differently by annotators of different genders, each annotation task includes a demographic question.[6]

**Data** To focus the data on interactions where gender may be a salient variable, Reddit data was drawn from RtGender (Voigt et al., 2018), which has a post-and-reply interactions where the gender of each author has been inferred with high confidence. After preliminary analyses, the initial data was randomly selected from 28 subreddits based on their focus around gender issues (e.g,. Relationships, AskWomen).[7]

For the **first round of annotation**, we used these four subsets of posts to be annotated for of-

[5]Even this interpretation of gender is limited, as women and nonbinary individuals experience gendered discrimination in different ways, and transgender men also have a marginalized gender identity.

[6]Following best practices in gender elicitation (Jaroszewski et al., 2018), we include a third option of "nonbinary, genderqueer, or otherwise", which has a free-form text box. We opted not to ask about cis or trans status in the demographic survey.

[7]See Supplementary Material §1 for the complete list.

fensiveness by the crowdworkers:

1. (SELFMA) Posts from our MAS corpus (§2) that contain gender-based MAS ($n = 59$);
2. (NON-OFFENSIVE) Posts manually picked from the Reddit dataset that should quite clearly be perceived as not offensive by everyone ($n = 36$);
3. (OFFENSIVE) Posts manually picked from the Reddit dataset, filtering by profane lexicons targeting women, and a corpus of Twitter hate speech (Davidson et al., 2017) that should quite clearly be perceived as offensive ($n = 21$);
4. (RANDOM) Posts randomly selected from the 28 subreddits ($n = 1007$).

The first three sets of posts serve as controls both to verify the soundness of our hypothesis (SELFMA) and to filter out untrusted crowdworkers (NON-OFFENSIVE and OFFENSIVE).

**Adaptive Data Selection** As described previously, after the first round of annotations, we use a classifier trained to predict discrepancy in perceived offensiveness to pick the next batch of posts for the **second round of annotation**. The classifier was given, as positive examples, posts with $\geq 0.25$ discrepancy[8] between the averaged perceived offensiveness of the dominant group and the marginalized groups. Other posts are considered negative examples. The feature sets that we used are motivated by prior work on gender bias and power dynamics. The following seven feature categories are used: (1) unigram and bigram features to capture lexical patterns, (2) two categories of formal and informal words derived from the formality corpus of Pavlick and Tetreault (2016), (3) the gendered occupations lexicon of Bolukbasi et al. (2016), grouped across definitional and stereotypical gender (male, female, neutral), (4) the gender stereotype lexicon of Fast et al. (2016), (5) the gender lexicon for social media from (Sap et al., 2014), (6) a manually-compiled corpus of gendered words, extended from seed list from https://www.hrc.org/resources/glossary-of-terms and (7) a manually-compiled sentiment lexicon, inspired by LIWC. To facilitate reproducibility, all lexicons will be available in the software release.

[8]The threshold was chosen empirically to ensure enough number of positive examples.

### 3.2 Experimental Setup

We designed the annotation task with two purposes in mind: (1) We want to test our hypothesis that there is discrepancy in how annotators of different genders perceive MAS. By comparing the distribution of discrepancy of perceived offensiveness in SELFMA, which consists solely of MAS, and of that in OFFENSIVE, which consists solely of offensive posts, we can see whether the discrepancy test actually is an **operationalizable criteria** to distinguish MAS from offensive posts in general. (2) We want to test our hypothesis that the discrepancy can be used **to surface MAS**, better than the alternatives. For the purpose of this experiment, we also trained a classifier to predict the offensiveness level of a post, and use that to provide a batch of posts to be annotated as well. By comparing the actual number of MAS by these two classifiers, we can test whether discrepancy can be used to surface more MAS compared to offensiveness in general.

**Platform Setup** Data was crowdsourced using Figure Eight with only Level 3 workers, a small "group of most experienced, highest accuracy contributors". Workers were shown 10 posts and replies per page, one of which was a test question. Workers whose ratings were substantially outside of the rating range of our ground truth annotations in more than 60% of cases were removed. Workers were paid $0.40 per page. To control for language and cultural norms, workers were restricted to being from the US, UK, Canada, and Australia.

Much like Mechanical Turk (Hara et al., 2019), Figure Eight has substantial representation of female-identifying individuals (Posch et al., 2018), with slightly more than 60% of users in the US identifying as female on both platforms (a higher rate than other countries). Studying gender-based MAS on these platforms means we are more likely to get a gender-balanced sample than compared with other social categories subject to microaggressions such as religion, age, or sexual orientation, which would likely require targeted recruiting of respondents or a massive collection of responses in order to achieve a balanced sample.

### 3.3 Results

For the first experiment, Figure 3 shows the distribution of perceived offensiveness discrepancy between the dominant group (men) and the marginalized group (women and non-binary) an-




Figure 3: Difference in perceived offensiveness between annotator genders show that MAs in the SELFMA were perceived as substantially more offensive by annotators identifying as women (shown through a positive rating discrepancy on y-axis) than offensive comments in OFFENSIVE. Difference is significant under t-test ($p < 0.001$).

notators, which supports our hypothesis that MAs in SELFMA are perceived as substantially more offensive by annotators identifying as women and non-binary.

For the second experiment, we asked crowdworkers to annotate 2,000 posts with the highest scores as predicted by the classifier, and then we took the top 6% of Reddit posts with the highest discrepancies scores and manually annotated them for MAs. We predicted that these posts contain a higher percentage of MA posts than those selected randomly, and this hypothesis is supported: Out of these aggregate 120 posts, 13 are MAs (10.8%),[9] compared to a manually annotated 200 random posts from the RANDOM batch, where only 6 of them (3%) were MAs. Examples are given in Table 4.

A potential next hypothesis suggested by our original hypothesis is that the classifiers trained on annotator discrepancy (with a higher level of offense for female and non-binary annotators) would be more successful in locating MAs than the classifiers trained on pure offensiveness. However, we did find equal performance or slight gains in microaggression location when using the offensiveness classifier. Based on analysis of the SELFMA data, we believe the cause may be that gender-based MAs were not always annotated as being more offensive to female annotators–many had strong discrepancies with male annotators being more offended. As our annotator discrepancy classifier was trained on samples more offensive to women, these types of MAs would be filtered out and decrease the model's overall effectiveness at surfacing MAs.

[9]The 2,000 posts were selected by taking the top 500 posts from the discrepancy and offensiveness classifiers on two different subsets of the Reddit data. Out of the 13 MAs, 6 come from the discrepancy classifiers, and 7 come from the offensiveness classifier.

## 4 Reported vs Observed Themes

How do self-reported and passively-observed MAs differ in thematic frequencies? Here, we use our thematic classifier (§2.4) to label all the microaggression data in Reddit. Analyzing the thematic differences reveals substantial differences between the two datasets. The Stereotype theme was by far the most common in our SELFMA data (53%), but it rarely occurs in our Reddit (3%). Instead, Objectification is the most common, as shown through the Institutionalized category (79%). Analyzing *where* the MAs occurred in Reddit, we observed that of the 19 instances of MAs identified by our annotators, many came from strongly male-focused subreddits (e.g., MensRights, Tinder, seduction, justneckbeardthings), where it would be expected that one male poster is talking to another. These online settings mirror offline highly-gendered settings where casually sexist language is known to happen, e.g., "locker room talk" (Katz, 1995). This is embodied in our Institutionalized category (specifically the subcategory "Objectification") having a much stronger presence in the Reddit MAs than the SELFMA ones in Table 5. While such language may not be perceived as offensive in the original dialog context, other readers who encounter it would likely feel offended and potentially marginalized.

Our work further points to two potential limitation of self-reported statements. First, the baseline rates of MAs may not be reliably estimated since the offensive comments are made elsewhere (outside of direct experience) and would only be discovered to through observational analysis. Second, the difference in baseline rates suggests that the relative differences in perceived offensiveness by theme may result in different reported quantities of microagressions; as not all comments and themes are equally offensive, people are more likely to recall the extremes of events (Tversky and Kahneman, 1973), potentially contributing to the difference in thematic rates in our two datasets.

## 5 Discussion and Limitations

As a first computational study of MAs, our work comes with some notable limitations that each provide clear avenues for future work. First, the current dataset contains just over 6.5K instances. While useful, this corpus does is likely not fully

| Classifier | Category | Subreddit : Examples |
|---|---|---|
| Discrepancy | Institutionalized | r/skateboarding: "this isn't a real competitor though. She's just a gross bimbo looking for attention [...]" |
| Discrepancy | Teaming | r/changemyview: "[...] Don't you think you ignored the real points I was making? Puting him in a "pro"-molester group, when he maybe just be pro tolerance?" |
| Offensive | Institutionalized | r/Tinder: "I'd f*** a stupid girl, but I definitely won't date or marry her. Other intelligent men are the same." |
| Offensive | Teaming | r/MensRights: "[...] I myself don't want to get the police involved over some girl saying she'll dump a guy or have an affair if he doesn't meet her sexual standards." |

Table 4: Examples of MAs surfaced by the discrepancy/offensiveness classifiers, with corresponding manually-annotated labels of their typology category.

| | Percentage | |
|---|---|---|
| Category | SELFMA (§2.1) | Reddit |
| Attributive | 53 | 3 |
| Institutionalized | 59 | 79 |
| Teaming | 10 | 21 |
| Othering | 7 | 10 |

Table 5: Gender-based MAs crawled in SELFMA ($n = 59$) and found in Reddit ($n = 19$) after the second round of annotation, according to their category from our proposed typology (§2.2).

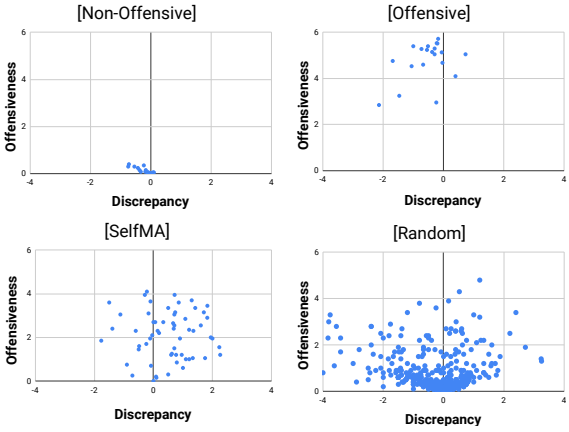


Figure 4: Offensiveness (y-axis) vs Discrepancy (x-axis) of perceived offensiveness between annotator gender. for the four sets of posts (§3.1). Note that compared to clearly offensive data [OFFENSIVE] and clearly non-offensive data [NON-OFFENSIVE], the SELFMA data (§2.1) identified as gendered MAs are annotated as somewhat offensive (indicated by the distribution around the middle of the y-axis), and having more discrepancy in perceived offensiveness between annotator genders (indicated by the higher degree of spread on the +x-axis). RANDOM data collected from Reddit sit all along the two axes of offensiveness and discrepancy.

representative of all types of MAs. However, our crowdsourcing approach does provide an effective way to surface MAs and our dual objective approach, which uses both annotator discrepancy and offensiveness, provides complementary views into what statements could be perceived as MAs. Additional iterations of this procedure are likely to improve microaggression recognition and substantially increase the seize of the corpus. We note that one option is to have workers *generate* examples, rather than rate (e.g., Xu et al., 2013; Su et al., 2016; Jiang et al., 2018); however, such a process raises ethical concerns of having crowdworkers generate toxic statements towards others.

Second, our current focus is on gender-based MAs. This choice was motivated by the observation that gender-based MAs are the largest category in the SELFMA data and, given that prior studies have shown substantial gender disparity online, with women receiving more negative behaviors (Duggan, 2017), this choice has the potential for highest impact. Our work builds upon a growing body of literature focused on identifying and mitigating gender disparity through computational means (e.g., Magno and Weber, 2014; Garimella and Mihalcea, 2016; Li et al., 2018; Field et al., 2019; Field and Tsvetkov, 2019). Further, our focus on gender also allows us to reliably recruit crowdworkers across the gender spectrum, whereas other social categories such as race or religion are more difficult to recruit in a balanced proportion through traditional mechanisms. However, despite this current focus, both the typology and annotation approach are designed for and readily amenable to other societal dimensions of MAs.

Third, performance could still be improved even for the targeted case of gender-based MAs. For example, Figure 4 plots the distribution of posts according to the averaged perceived offensiveness (y-axis) and the discrepancy between the dominant group and the marginalized groups (x-axis). This shows that offensiveness and discrepancy in conjunction are better features than using just discrepancy in separating MAs from simply offensive and non-offensive posts. A possible future direction could be to train a multi-task classifier to predict both features in order to choose posts to annotate.

A final concern is that our approach may not truly be an "objective" replacement to current MAIdentification, as annotator bias is still present in this process. By choosing perceived offensiveness and discrepancies in perception as proxy measures for MAs, we hope to shift studies of

MAs from a judgment call on an individual's intent to a description of how it affects its targets, which can be described as an objective measure of human behavior that can be validated through study. However, it is true that without that validation, it is difficult to prove the behavior of our crowdsourced annotators represents inherent, objective truths about how groups react to MAs.

## 6 Conclusion

Offensive language can take many forms: seemingly positive comments can in reality be dismissive of the experiences and validity of other persons. MAs, everyday comments that marginalize others on the basis of their identity, are a common form of this linguistically subtle offensive language. While prior work in NLP has largely focused on overtly offensive language, efforts to recognize this more insidious and equally impactful language has been stymied by the lack of data and methods for effectively annotating large corpora. In this work, we posit that covert toxicity can be detected through contextualized analysis of subtle linguistic cues: pronoun uses, adjectives, mentions of social groups, stylistic cues, etc. We thus view the task of detection of MAs as a yet unexplored, socially-relevant NLP task, somewhat similar to (but possibly harder than) sentiment analysis and detection of hate speech.

This work makes three key contributions. First, we introduce a new thematic classification of MAs and show that annotations can reach moderate agreement on a highly difficult task–countering claims that microaggressions are too subjective to be reliable–and also showing that computational methods can distinguish these themes. Second, we introduce a new active-learning crowdsourcing approach for surfacing potential MAs from social media using not only predictions of offensiveness but also predictions of rating discrepancy by gender within the annotators themselves. We show that both classifiers are effective at increasing rates of finding microaggression *and* that each finds different types. Finally, we show that self-reported and observed microaggression differ substantially in their types, suggesting computational mechanisms are needed to precisely estimate their true pervasiveness. Our work and corresponding data and models fill a key missing gap for expanding the range of offensive language detection to linguistically subtle comments.

## Acknowledgments

The authors would like to thank the three reviewers for their very helpful and insightful feedback, Figure Eight for providing funding as a part of their AI for Everyone project, and the Washington Post data science team for data used during initial pilot studies of this project. YT gratefully acknowledges the support of the Okawa Foundation and of the NSF grant no. IIS1812327. EA was supported by NSF GRFP under grant no. DGE1745016. The authors would also like to acknowledge the people who helped with the pilot study: Gayatri Bhat, Alexander Coda, Anjalie Field, Shirley Anugrah Hayati, Arnav Kumar, Sachin Kumar, Shrimai Prabhumoye, Qinlan Shen, Sumeet Singh, Craig Stewart, Michael Yoder, Yiheng Zhou.

## References

Betty van Aken, Julian Risch, Ralf Krestel, and Alexander Löser. 2018. Challenges for toxic comment classification: An in-depth error analysis. In *Proceedings of the 2nd Workshop on Abusive Language Online*.

Ron Artstein and Massimo Poesio. 2008. Inter-coder agreement for computational linguistics. *Computational Linguistics*, 34(4):555–596.

Gavin de Becker. 1997. *The Gift of Fear*. Little, Brown.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research (JMLR)*, 3(Jan):993–1022.

Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Advances in Neural Information Processing Systems*, pages 4349–4357.

Bradley Campbell and Jason Manning. 2014. Microaggression and moral cultures. *Comparative sociology*, 13(6):692–726.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Proc. ICWSM*, ICWSM '17, pages 512–515.

Maeve Duggan. 2017. Online harassment 2017. *Pew Research Center*.

Ethan Fast, Tina Vachovsky, and Michael S Bernstein. 2016. Shirtless and dangerous: Quantifying linguistic signals of gender bias in an online fiction writing community. In *ICWSM*, pages 112–120.

Anjalie Field, Gayatri Bhat, and Yulia Tsvetkov. 2019. Contextual affective analysis: A case study of people portrayals in online #metoo stories. In *Proc. ICWSM*.

Anjalie Field and Yulia Tsvetkov. 2019. Entity-centric contextual affective analysis. In *Proc. ACL*.

Shandra Forrest-Bank and Jeffrey M Jenson. 2015. Differences in experiences of racial and ethnic microaggression among asian, latino/hispanic, black, and white young adults. *J. Soc. & Soc. Welfare*, 42:141.

Paula Fortuna and Sérgio Nunes. 2018. A survey on automatic detection of hate speech in text. *ACM Computing Surveys (CSUR)*, 51(4):85.

Aparna Garimella and Rada Mihalcea. 2016. Zooming in on gender differences in social media. In *Proceedings of the COLING Workshop on Computational Modeling of People's Opinions, Personality, and Emotions in Social Media (PEOPLES)*, pages 1–10.

Kotaro Hara, Abigail Adams, Kristy Milland, Saiph Savage, Benjamin V Hanrahan, Jeffrey P Bigham, and Chris Callison-Burch. 2019. Worker demographics and earnings on amazon mechanical turk: An exploratory analysis. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*.

Samantha Jaroszewski, Danielle Lottridge, Oliver L Haimson, and Katie Quehl. 2018. Genderfluid or attack helicopter: Responsible HCI research practice with non-binary gender variation in online communities. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 307:1–307:15. ACM.

Akshita Jha and Radhika Mamidi. 2017. When does a compliment become sexist? analysis and classification of ambivalent sexism using twitter data. In *Proceedings of the Second Workshop on NLP and Computational Social Science*, pages 7–16.

Youxuan Jiang, Catherine Finegan-Dollak, Jonathan K Kummerfeld, and Walter Lasecki. 2018. Effective crowdsourcing for a new type of summarization task. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 628–633.

David Jurgens, Eshwar Chandrasekharan, and Libby Hemphill. 2019. A just and comprehensive strategy for using nlp to address online abuse. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Jackson Katz. 1995. Reconstructing masculinity in the locker room: The mentors in violence prevention project. *Harvard Educational Review*, 65(2):163–175.

Kiyun Kim. 2013. Racial Microaggressions. http://nortonism.tumblr.com.

Jia Li, Xuan Liu, and Min Sun. 2018. Research on gender differences in online health communities. *International Journal of Medical Informatics*.

Scott O. Lilienfeld. 2017. Microaggressions: Strong Claims, Inadequate Evidence. *Perspectives on Psychological Science*, 12(1):138–169.

Gabriel Magno and Ingmar Weber. 2014. International gender differences and gaps in online social networks. In *Proceedings of the International Conference on Social Informatics (SocInfo)*, pages 121–138. Springer.

Kevin L. Nadal, Katie E. Griffin, Yinglee Wong, Sahran Hamit, and Morgan Rasmus. 2014. The impact of racial microaggressions on mental health: Counseling implications for clients of color. *Journal of Counseling and Development*, 92(1):57–66.

Blake Neff. 2015. California trains individuals to avoid microaggressions.. http://dailycaller.com/2015/06/10/california-trains-professors-to-avoid-microaggressions/.

Rebecca J Passonneau, Collin Baker, Christiane Fellbaum, and Nancy Ide. 2012. The masc word sense sentence corpus. In *Proceedings of LREC*.

Ellie Pavlick and Joel Tetreault. 2016. An empirical analysis of formality in online communication. *Transactions of the Association of Computational Linguistics*, 4(1):61–74.

James W Pennebaker, Roger J Booth, Ryan L Boyd, and Martha E Francis. 2015. Linguistic inquiry and word count: LIWC 2015 [computer software]. pennebaker conglomerates.

Lisa Posch, Arnim Bleier, Fabian Flöck, and Markus Strohmaier. 2018. Characterizing the global crowd workforce: A cross-country comparison of crowdworker demographics. *arXiv preprint arXiv:1812.05948*.

Farzana Rashid and Eduardo Blanco. 2017. Dimensions of interpersonal relationships: Corpus and experiments. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2307–2316, Copenhagen, Denmark. Association for Computational Linguistics.

Hannah Rashkin, Sameer Singh, and Yejin Choi. 2016. Connotation Frames: A Data-Driven Investigation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 5, pages 311–321, Stroudsburg, PA, USA. Association for Computational Linguistics.

Joni Salminen, Hind Almerekhi, Milica Milenković, Soon-gyo Jung, Jisun An, Haewoon Kwak, and Bernard J Jansen. 2018. Anatomy of online hate:

developing a taxonomy and machine learning models for identifying and classifying hate in online news media. In *Twelfth International AAAI Conference on Web and Social Media*.

Maarten Sap, Gregory Park, Johannes Eichstaedt, Margaret Kern, David Stillwell, Michal Kosinski, Lyle Ungar, and Hansen Andrew Schwartz. 2014. Developing age and gender predictive lexica over social media. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1146–1151.

Anna Schmidt and Michael Wiegand. 2017. A Survey on Hate Speech Detection using Natural Language Processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10.

Kimber Shelton and Edward A Delgado-Romero. 2013. Sexual orientation microaggressions: The experience of lesbian, gay, bisexual, and queer clients in psychotherapy. *Psychology of Sexual Orientation and Gender Diversity*.

Yu Su, Huan Sun, Brian Sadler, Mudhakar Srivatsa, Izzeddin Gur, Zenghui Yan, and Xifeng Yan. 2016. On generating characteristic-rich question sets for qa evaluation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 562–572.

Derald Wing Sue. 2010. *Microaggressions in Everyday Life: Race, Gender, and Sexual Orientation*. Wiley, Hoboken, NJ.

Derald Wing Sue, Christina M Capodilupo, Gina C Torino, Jennifer M Bucceri, Aisha M.B. B. Holder, Kevin L Nadal, and Marta Esquilin. 2007. Racial microaggressions in everyday life: Implications for clinical practice. *American Psychologist*, 62(4):271–286.

Amos Tversky and Daniel Kahneman. 1973. Availability: A heuristic for judging frequency and probability. *Cognitive psychology*, 5(2):207–232.

Rob Voigt, David Jurgens, Vinodkumar Prabhakaran, Dan Jurafsky, and Yulia Tsvetkov. 2018. Rtgender: A corpus for studying differential responses to gender. In *LREC*.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding Abuse: A Typology of Abusive Language Detection Subtasks. pages 78–84.

Gloria Wong, Annie O Derthick, EJR David, Anne Saw, and Sumie Okazaki. 2014. The what, the why, and the how: A review of racial microaggressions research in psychology. *Race and social problems*, 6(2):181–200.

Wei Xu, Alan Ritter, and Ralph Grishman. 2013. Gathering and generating paraphrases from twitter with application to normalization. In *Proceedings of the sixth workshop on building and using comparable corpora*, pages 121–128.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2017. Men also like shopping: Reducing gender bias amplification using corpus-level constraints. In *Proc. of EMNLP*, pages 2979–2989.

# Reinforced Product Metadata Selection for Helpfulness Assessment of Customer Reviews

**Miao Fan, Chao Feng, Mingming Sun, Ping Li**
Cognitive Computing Lab
Baidu Research
No.10 Xibeiwang East Road, Beijing, 10085, China
10900 NE 8th St, Bellevue, WA 98004, USA
{fanmiao, v_fegchao, sunmingming01, liping11}@baidu.com

## Abstract

To automatically assess the helpfulness of a customer review online, conventional approaches generally acquire various linguistic and neural embedding features solely from the textual content of the review itself as the evidence. We, however, find out that a helpful review is largely concerned with the metadata (such as the name, the brand, the category, etc.) of its target product. It leaves us with a challenge of how to choose the correct *key-value* product metadata to help appraise the helpfulness of *free-text* reviews more precisely. To address this problem, we propose a novel framework composed of two mutual-benefit modules. Given a product, a selector (agent) learns from both the *keys* in the product metadata and one of its reviews to take an action that selects the correct *value*, and a successive predictor (network) makes the *free-text* review attend to this *value* to obtain better neural representations for helpfulness assessment. The predictor is directly optimized by SGD with the loss of helpfulness prediction, and the selector could be updated via policy gradient rewarded with the performance of the predictor. We use two real-world datasets from Amazon.com and Yelp.com, respectively, to compare the performance of our framework with other mainstream methods under two application scenarios: helpfulness identification and regression of customer reviews. Extensive results demonstrate that our framework can achieve state-of-the-art performance with substantial improvements.

## 1 Introduction

The massive number of reviews left by many experienced consumers on online products are our priceless treasure in e-commerce. We believe that online customer reviews can provide more subjective and informative opinions from various perspectives on the products besides the objective descriptions given by their merchants. Hence, we prefer browsing the reviews online for the sake of finding our desirable products. However, it is quite time-consuming for potential buyers to sift through millions of online reviews with uneven qualities to make purchase decisions.

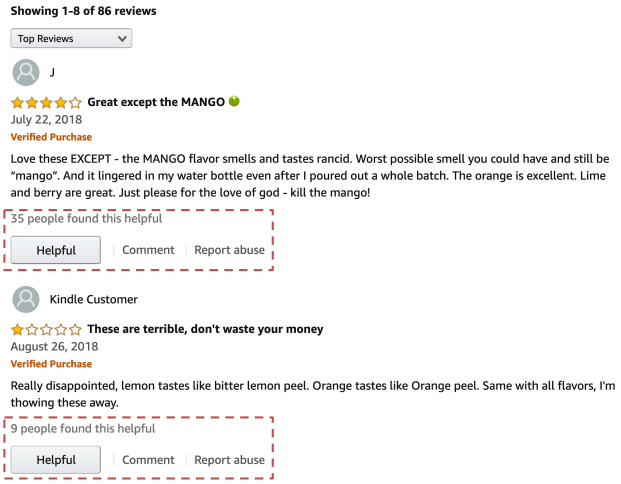


Figure 1: A screenshot of the top review on a product sold in Amazon.com: *SodaStream Fruit Drops Variety Pack, 1.67 Pound*. Online users can vote for the helpfulness of each customer review by clicking the "Helpful" button underneath.

To discover and surface helpful reviews for customers, some well-known websites (e.g., Amazon.com) have launched a module illustrated by Figure 1 which asks for feedback on the helpfulness of online reviews. And a recent study[1] reported that this featured module can increase the revenue of Amazon with an estimated 27 billion U.S. dollars annually. Although this crowdsourcing module is quite useful to find a fraction of helpful reviews, statistics (Fan et al., 2018) indicate that roughly **60%** online reviews still do not receive any vote of helpfulness or unhelpful-

[1] articles.uie.com/magicbehindamazon/



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1675–1683,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

ness. This phenomenon on unknown helpfulness is even more prominent in low-traffic items including those less popular and new arrival products.

We believe it is a promising study that establishes an automatic helpfulness assessment system for online reviews, which can be as useful as a product recommendation engine (Park et al., 2012) in e-commerce. Moreover, review helpfulness assessment (Diaz and Ng, 2018) and sentiment analysis (Liu and Zhang, 2012) are two different but related lines of work to study on the user-generated content (UGC). Sentiment analysis mainly focuses on identifying the opinion orientation of the reviewer himself/herself on the target product. A reviewer can express several emotional words in his/her comment. However, this comment may contain less information on the target product and may not be helpful to other potential consumers. Therefore, review helpfulness assessment concerns more about whether a comment is useful/helpful to other potential consumers.

So far as we know, a series of work on review helpfulness prediction has been proposed from two perspectives: 1) some work leveraged domain-specific knowledge to extract a wide range of hand-crafted features (including structural (Mudambi and Schuff, 2010; Xiong and Litman, 2014), lexical (Kim et al., 2006; Xiong and Litman, 2011), syntactic (Kim et al., 2006), emotional (Martin and Pu, 2014), semantic (Yang et al., 2015) and even argument features (Liu et al., 2017)) from the raw text of reviews as the evidence fed to off-the-shelf learning tools for helpfulness prediction; and 2) recent studies (Chen et al., 2018a,b; Fan et al., 2018) took advantages of deep neural nets by modifying the convolutional neural network (Kim, 2014) to acquire the low-dimensional representations of helpful reviews without the aid of feature engineering. Overall, these mainstream methods extract various linguistic and neural features solely from the raw text of a review as the evidence for helpfulness assessment.

We, on the other hand, consider that identifying the helpfulness of a review should be fully aware of the metadata (such as the name, the brand, the place of origin, the category, the description, etc.) of the target product besides the textual content of the review itself. To illustrate our idea, Figure 2 shows an example of two customer reviews in Amazon.com with diverse helpfulness scores on

**PRODUCT METADATA**

| | |
|---|---|
| ***Title:*** | Sodastream Sodamix Variety Pack (6 diet and 6 trial portion packs) |
| ***Brand:*** | SodaStream |
| ***Description:*** | SodaStream Soda mixes let you add new flavors to your repertoire for parties, dinners and other gatherings. This variety pack includes 12 single-use samples to try some of SodaStream's most popular flavors... |
| ***Categories:*** | [Grocery, Gourmet Food] |

==========

**CUSTOMER REVIEWS**

I did not need this but I read a review about it that highly recommended getting it. Waste of my money. (**Helpfulness: [0, 17]**)

I recently bought a Sodastream Genesis for the purpose of creating my own club soda. It a great machine. As an additional experiment, I've tried a few of the flavors from this sampler pack ... (**Helpfulness: [104, 114]**)

Figure 2: Two customer reviews with diverse helpfulness scores on the same product in Amazon.com.

the same product online. We can easily figure out that the helpful review (104 of 114 people found it helpful) may concern the key of "description" in the metadata of the product, while the unhelpful review (0 of 17 people found it helpful) talks nothing about the product but expresses deep remorse. Without direct supervision from human beings, it is difficult for machines to infer the correct key/aspect in the product metadata that helpful reviews really concern.

It leaves us with a problem of how to teach machines learning to choose the correct *key-value* product metadata to help assess the helpfulness of *free-text* reviews more precisely. To address the issue, this paper introduces a novel framework composed of two mutual-benefit modules: a product metadata selector (agent) and a review helpfulness predictor (network). This work is proposed to be deployed in a real-world system. Therefore, we suggest to decouple the two modules (i.e., the selector and the predictor). Leveraging the attention mechanism is an alternative approach on an end-to-end framework. However, the decoupled modules are preferred in an industrial system.

Given a product, the selector **explores** the connection between the *keys* in the product metadata and one of its reviews to take an action that selects the correct *value*, and the successive predictor **exploits** this *value* attended by the *free-text* review to acquire better neural representations for helpfulness prediction. The predictor is directly

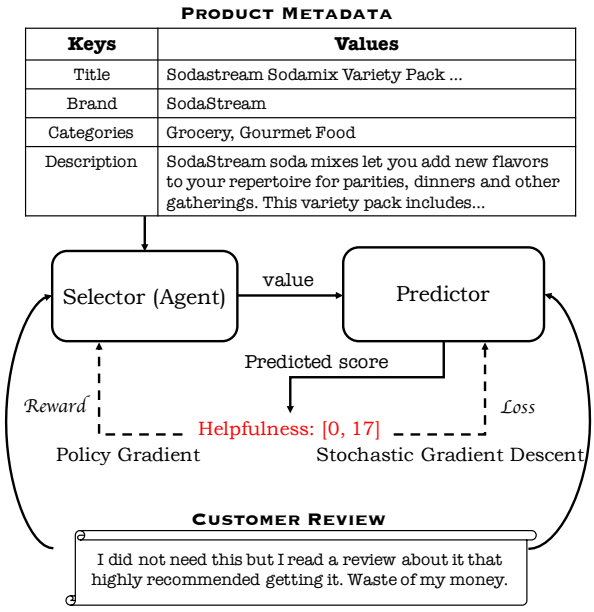


Figure 3: Our reinforced framework for review helpfulness prediction: R$^2$HP.

optimized by the stochastic gradient descent algorithm (Ruder, 2016) with the loss of prediction, and the selector can be updated via the policy gradient method (Sutton et al., 2000) rewarded with the performance of the predictor.

We use two real-world datasets from Amazon.com and Yelp.com, to compare the performance with other mainstream approaches on two application tasks: helpfulness identification and regression of customer reviews. The experimental results reveal that our framework can reach state-of-the-art performance on both tasks with substantial improvements. In addition, our framework can help acquire the embeddings of the keys in product metadata, and visualization results illustrate that they can capture various aspects on customer reviews.

## 2 Framework

The intuition of our framework (named *reinforced review helpfulness prediction*, abbr. as R$^2$HP) is to predict the helpfulness of a customer review which is fully aware of the correct product metadata selected by a reinforced selector (agent). This work is proposed to be deployed in a real-world system which is responsible for recommending helpful reviews to millions of customers. Therefore, we suggest decoupling the two modules (selector and predictor). Leveraging the attention mechanism (Vaswani et al., 2017) is an alternative approach to establishing an end-to-end framework. However, the decoupled modules are preferred by the industry.

As shown by Figure 3, the selector (agent) learns from both the *keys* in the product metadata and one of its reviews to take an action that selects the correct *value*, and a successive predictor (network) makes the *free-text* review attend to this *value* to obtain better neural representations for helpfulness prediction. The predictor is directly optimized by SGD with the loss of prediction, and the selector could be updated via policy gradient rewarded with the performance of the predictor.

### 2.1 Reinforced Product Metadata Selection

Given a customer review and the key-value formed product metadata, our reinforced neural selector $\pi$ takes them as input, to output a policy $\mathbf{p}$ which is the probability distribution over the keys. Suppose that the product metadata contains $k$ keys each of which is represented by an $l$-dimensional vector. Then we can achieve the embeddings of the keys: $\mathbf{K} \in \mathbb{R}^{l \times k}$.

Assume that there are $n$ words/tokens in the customer review $c$. We align each word/token with the embedding dictionary acquired by the word embedding approaches such as Word2Vec (Mikolov et al., 2013), Glove (Pennington et al., 2014) or Elmo (Peters et al., 2018) to initialize the distributed representations of the customer review $\mathbf{C} \in \mathbb{R}^{l \times n}$. To achieve the local contextual embeddings of the customer review $c$, we use a Bi-LSTM network (Schuster and Paliwal, 1997) which takes the word embeddings of the customer review $\mathbf{C}$ as input:

$$\mathbf{H}_c = \text{Bi-LSTM}(\mathbf{C}). \quad (1)$$

$\mathbf{H}_c \in \mathbb{R}^{2l \times n}$ stands for the contextual embeddings where each word can obtain two hidden units with the length of $2l$ encoding both the backward and the forward contextual information of the customer review locally.

We use $\mathbf{B} \in \mathbb{R}^{k \times n}$ to obtain the bilinear relationship between the embeddings of the keys $\mathbf{K}$ and the local contextual embeddings of the customer review $\mathbf{H}_c$:

$$\mathbf{B} = \text{ReLU}(\mathbf{K}^{\mathrm{T}}\mathbf{W}\mathbf{H}_c), \quad (2)$$

where $\mathbf{W} \in \mathbb{R}^{l \times 2l}$ is the weight matrix for the Rectifier Linear Unit (ReLU). The $i$-th row of $\mathbf{B}$ contains the aspect/topic feature aligned by the local contextual embeddings of the customer review.

We apply the *reduce_max* strategy to each row of $\mathbf{B}$ to keep the most significant feature for each metadata key, and use the *softmax* function to gain the policy $\mathbf{p} \in \mathbb{R}^k$:

$$\mathbf{p} = \text{Softmax}(\text{Reduce\_max}(\mathbf{B}, axis = 1)). \quad (3)$$

Then our reinforced selector (agent) can select a metadata value $v \sim \pi(v|\mathbf{K}, \mathbf{C})$ in terms of $\mathbf{p}$.

### 2.2 Product-aware Review Helpfulness Prediction

In this part, we elaborate our neural predictor for review helpfulness prediction. It is devised based on the motivation that the helpfulness of an online review should be fully aware of the selected metadata of its target product besides the textual content of the customer review itself. The predictor is composed of two components: 1) the local contextual embeddings of a review and 2) the product-aware distributed representations of the review. We have explained how to obtain the local contextual embeddings of a review $\mathbf{H}_c \in \mathbb{R}^{2l \times n}$ in the previous subsection. Here we will describe how to achieve the product-aware distributed representations of the review, denoted by $\overline{\mathbf{H}}_c$.

We use $m$ to stand for the number of tokens/words in the selected metadata value $v$ provided by our reinforced selector. Similarly, we can refine the word embeddings of the value $\mathbf{V}$ of the product metadata via another Bi-LSTM network:

$$\mathbf{H}_v = \text{Bi-LSTM}(\mathbf{V}), \quad (4)$$

and achieve the contextual embeddings of the selected product metadata $\mathbf{H}_v \in \mathbb{R}^{2l \times m}$. To make the contextual embeddings of the customer review fully aware of the product metadata, we design a word-level matching mechanism as follows,

$$\mathbf{Q} = \text{ReLU}(\mathbf{W}'\mathbf{H}_v + \mathbf{b}' \otimes \mathbf{e})^{\mathrm{T}}\mathbf{H}_c, \quad (5)$$

where $\mathbf{W}' \in \mathbb{R}^{2l \times 2l}$ is the weight matrix and $\mathbf{b}' \in \mathbb{R}^{2l}$ is the bias vector. The outer product $\otimes$ copys the bias vector $\mathbf{b}'$ for $m$ times (i.e., $\mathbf{e} \in \mathbb{R}^m$) to generate a $2l \times m$ matrix. Then $\mathbf{Q} \in \mathbb{R}^{m \times n}$ is the sparse matrix that holds the word-level matching information between the value $v$ of the product metadata and the customer review $c$.

If we further apply the *softmax* function to each column of $\mathbf{Q}$, we will obtain $\mathbf{G} \in \mathbb{R}^{m \times n}$, the $i$-th column of which represents the normalized attention weights over all the words in the metadata value $v$ for the $i$-th word in the customer review $c$:

$$\mathbf{G} = \text{Softmax}(\mathbf{Q}). \quad (6)$$

Then we can use the attention matrix $\mathbf{G} \in \mathbb{R}^{m \times n}$ and the contextual embeddings of the product metadata $\mathbf{H}_v \in \mathbb{R}^{2l \times m}$ to re-form the product-aware review representation $\overline{\mathbf{H}}_c \in \mathbb{R}^{2l \times n}$:

$$\overline{\mathbf{H}}_c = \mathbf{H}_v\mathbf{G}. \quad (7)$$

Driven by original motivation, we need to join the local contextual embeddings of the review ($\mathbf{H}_c$) and the product-aware distributed representations of the review ($\overline{\mathbf{H}}_c$) together for predicting its helpfulness with the feature matrix $\mathbf{H} \in \mathbb{R}^{2l \times n}$:

$$\mathbf{H} = \mathbf{H}_c + \overline{\mathbf{H}}_c. \quad (8)$$

$\mathbf{H}$ can also benefit from the idea of ResNet (He et al., 2016) that efficiently acquires the residual between $\mathbf{H}_c$ and $\overline{\mathbf{H}}_c$, and provides a highway to update $\mathbf{H}_c$ if the residual is tiny.

Generally speaking, we define a loss function[2] $L(s_g|\mathbf{H}_c)$ which takes $\mathbf{H}_c$ as the feature to predict a helpfulness score $s_p$ judged by the ground-truth score $s_g$. Given the value $v$ is selected by $\pi$, our objective is to minimize the expectation:

$$J(\Theta) = \mathbb{E}_{v \sim \pi(v|\mathbf{K},\mathbf{C})}[L(s^g|v, \mathbf{H}^c)], \quad (9)$$

where $\Theta$ are parameters to be learned. The gradient of $J(\Theta)$ with respect to $\Theta$ is:

$$\begin{aligned} &\nabla_\Theta J(\Theta) \quad (10)\\ &= \nabla_\Theta \sum_v \pi(v|\mathbf{K}, \mathbf{C})[L(s^g|v, \mathbf{H}^c)] \\ &= \sum_v (L(s^g|v, \mathbf{H}^c)\nabla_\Theta \pi(v|\mathbf{K}, \mathbf{C}) + \pi(v|\mathbf{K}, \mathbf{C})\nabla_\Theta L(s^g|v, \mathbf{H}^c)) \\ &= \mathbb{E}_{v \sim \pi(v|\mathbf{K},\mathbf{C})}[L(s^g|v, \mathbf{H}^c)\nabla_\Theta \log \pi(v|\mathbf{K}, \mathbf{C}) + \nabla_\Theta L(s^g|v, \mathbf{H}^c)], \end{aligned}$$

where $\nabla_\Theta L(s^g|v, \mathbf{H}^c)$ refers to training the predictor by SGD and we use the REINFORCE algorithm (Williams, 1992) to update the selector with the gradient of $\log \pi(v|\mathbf{K}, \mathbf{C})$ and the reward $L(s^g|v, \mathbf{H}^c) \in (0.0, 1.0]$.

## 3 Experiments

### 3.1 Real-world Datasets

We look up two well-formatted JSON resources online which contain plenty of product metadata (including titles, brands, categories, and descriptions) and numerous customer reviews. One is the data collection[3] of Amazon.com crawled by

[2] We usually use the mean square error (MSE) as the loss function for helpfulness score regression and the cross-entropy as the loss function for helpfulness identification.

[3] The data collection is publicly available at http://jmcauley.ucsd.edu/data/amazon/links.html

| Category (Amazon) | Training Set | | | Test Set | | |
|---|---|---|---|---|---|---|
| | # (P.) | # (R. $\geq$ 1v.) | # (R. $\geq$ 0.75h.r.) | # (P.) | # (R. $\geq$ 1v.) | # (R. $\geq$ 0.75h.r.) |
| *Clothing, Shoes & Jewelry* | 8,789 | 52,402 | 37,036 | 964 | 6,630 | 4,668 |
| *Electronics* | 16,456 | 215,512 | 136,798 | 1,858 | 21,706 | 13,845 |
| *Grocery & Gourmet Food* | 30,617 | 231,131 | 151,828 | 3,392 | 25,984 | 16,982 |
| *Health & Personal Care* | 19,817 | 216,839 | 135,501 | 2,198 | 23,517 | 14,565 |
| *Home & Kitchen* | 17,999 | 202,549 | 147,324 | 1,978 | 24,163 | 17,896 |
| *Movies & TV* | 7,518 | 316,235 | 143,790 | 847 | 35,880 | 15,973 |
| *Pet Supplies* | 18,189 | 167,775 | 123,287 | 2,031 | 21,170 | 15,650 |
| *Tools & Home Improvement* | 30,105 | 217,397 | 150,483 | 3,454 | 24,971 | 17,426 |
| *TOTAL* | 149,490 | 1,619,840 | 1,026,047 | 16,722 | 184,021 | 117,005 |

Table 1: The statistics of the Amazon dataset for helpfulness prediction of online reviews. # (P.): the number of products; # (R. $\geq$ 1v.): the number of the reviews, each receiving at least 1 vote regardless of helpfulness/unhelpfulness; # (R. $\geq$ 0.75h.r.): the number of the reviews, each regarded as helpful by at least 75% votes.

| Category (Yelp) | Training Set | | | Test Set | | |
|---|---|---|---|---|---|---|
| | # (P.) | # (R. $\geq$ 1v.) | # (R. $\geq$ 0.75h.r.) | # (P.) | # (R. $\geq$ 1v.) | # (R. $\geq$ 0.75h.r.) |
| *Beauty & Spas* | 8,552 | 101,383 | 56,711 | 949 | 10,007 | 5,659 |
| *Health & Medical* | 10,079 | 90,045 | 57,967 | 1,132 | 10,434 | 6,758 |
| *Home Services* | 7,675 | 71,247 | 47,369 | 876 | 7,011 | 4,575 |
| *Restaurants* | 3,432 | 100,613 | 41,139 | 383 | 11,668 | 4,779 |
| *Shopping* | 11,706 | 104,555 | 49,833 | 1,295 | 12,892 | 5,969 |
| *TOTAL* | 41,444 | 467,843 | 253,019 | 4,635 | 52,012 | 27,740 |

Table 2: The statistics of the Yelp dataset for helpfulness prediction of online reviews. # (P.): the number of products; # (R. $\geq$ 1v.): the number of the reviews, each receiving at least 1 vote regardless of helpfulness/unhelpfulness; # (R. $\geq$ 0.75h.r.): the number of the reviews, each regarded as helpful by at least 75% votes.

(He and McAuley, 2016) up to July 2014. The other one is the dump file[4] directly provided by Yelp.com for academic purposes. We use the product ids (i.e., "asin" in Amazon and "business_id" in Yelp) as the foreign keys to align the metadata of products with customer reviews. 80% products with online reviews are randomly picked as the training set, leaving the rest as the test set. In this way, two real-world datasets, i.e., *Amazon* and *Yelp*, are built, and the statistics of them are shown by Table 1 and Table 2, respectively.

In this study, we regard the reviews which receive at least 1 vote for helpfulness/unhelpfulness, i.e., the column # (R.) $\geq$ 1v. in Table 1 and Table 2, as the experimental samples. In *Amazon*, the crowd-sourcing module for voting helpful reviews provides an "$X$ of $Y$" helpfulness score, where "$Y$" stands for the total number of users who participate in voting, and "$X$" denotes the number of users who think the review is helpful. *Yelp* offers more options: *useful*: $X$, *cool*: $Y$, and *funny*: $Z$, to the users who are willing to give feedback. Regardless of the difference, we generally consider the reviews which receive at least 0.75 ratio of helpfulness/usefulness (helpfulness/usefulness score $\geq 0.75$) as positive samples, leaving the others as the negative samples for classification.

### 3.2 Comparison Methods

We compare our framework (R$^2$HP) with a wide range of prior arts. Specifically, we re-implement the methods of learning from deep neural networks and with hand-crafted features. The up-to-date neural approaches involve the embedding-gated CNN (EG-CNN) (Chen et al., 2018a,b) and the multi-task neural learning (MTNL) architecture (Fan et al., 2018) for review helpfulness prediction. The hand-crafted features include the structural features (STR) (Mudambi and Schuff, 2010; Xiong and Litman, 2014), the lexical features (LEX) (Xiong and Litman, 2011), the emotional features (GALC) (Martin and Pu, 2014) and the semantic features (INQUIRER) (Yang et al., 2015). We also add two more experiments on integrating all the hand-crafted features via the Support Vector Machines (SVM) and the Random

[4]The dump file can be downloaded from https://www.yelp.com/dataset

Forest (R.F.) model for review helpfulness assessment.

### 3.3 Application Scenarios

Previous studies mostly reported their performance on either the application scenario of review helpfulness identification or regression. Therefore, we conduct extensive experiments comparing the performance of our framework with all the other approaches under both scenarios.

*Identification of helpful reviews*: As both the training and test sets are imbalanced, we adopt the Area under Receiver Operating Characteristic (AUROC) as the metric to evaluate the performance of all the approaches on helpful review identification. In line with Table 3 and Table 4, MTNL (Fan et al., 2018) achieves the best performance up-to-date on this classification task among the baseline approaches as it achieves the best performance on 12 of 14 categories in *Amazon* and *Yelp* datasets. R$^2$HP surpasses MTNL on both datasets and obtains state-of-the-art (micro-averaged) results of 67.5% AUROC (*Amazon*) and 75.1% AUROC (*Yelp*) with absolute improvements of **4.9%** AUROC and **4.7%** AUROC, respectively.

*Regression of helpfulness score*: In this task, all the approaches are required to predict the fraction of helpful votes that each review receives. We use the data in the column # (R. $\geq$ 1v.) in Table 1 and Table 2 as the training and test sets. The Squared Correlation Coefficient (R$^2$-score) is adopted as the metric to evaluate the performance of all the approaches on helpfulness score regression. Table 5 and Table 6 show that MTNL (Fan et al., 2018) achieves state-of-the-art performance on this regression task among the baseline approaches. Our framework outperforms MTNL on both datasets and obtains state-of-the-art (micro-averaged) results of 62.3% R$^2$-score (*Amazon*) and 74.0% R$^2$-score (*Yelp*) with absolute improvements of **5.4%** R$^2$-score and **5.8%** R$^2$-score, respectively.

## 4 Discussions

### 4.1 Ablation Study on Metadata Selector

In this part, we conduct the ablation study on different metadata selectors: i.e., random selector, heuristic selector, and our reinforced selector. The random selector requires no prior knowledge but randomly picks one pair of (key, value) from the product metadata with uniform distribution. The heuristic selector keeps choosing the pair of (key, value) in which the value contains the longest text[5]. Our reinforced selector learns from the reward given by the helpfulness predictor and makes a wise decision on the metadata selection. The values of product metadata selected by the three selectors are fed into the same predictor.

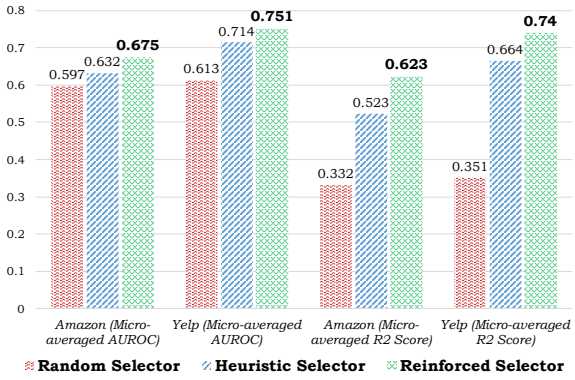


Figure 4: Performances of review helpfulness identification and regression supported by the random, the heuristic, and our reinforced selector on the *Amazon* and *Yelp* datasets.

Figure 4 shows the performance of both identification and regression of review helpfulness supported by the three different selectors, and the results demonstrate that our reinforced selector surpasses the other policies (the random selector and the heuristic selector) for metadata selection.

### 4.2 Case Study on Key Embeddings

Our framework also helps to acquire the distributed representations of the keys in product metadata. We believe these key embeddings can capture various aspects/topics on customer reviews and lead to the correct value for review helpfulness prediction. With the help of t-SNE (Maaten and Hinton, 2008), we can map the embeddings of the metadata keys from the *Amazon* and *Yelp* datasets into 2D vectors and illustrate them in Figure 5.

It shows that the key embeddings in *Amazon* locate at the bottom-right and the key embeddings in *Yelp* are generally at upper-left. The keys which express similar meanings within the same dataset are close to each other, such as the keys "city", "state" and "address" in *Yelp*, and the keys "title", "description" and "brand" in *Amazon*. Even across

[5]In most cases, the "name" or the "description" of a product is selected by the heuristic selector.

| Category (Amazon) | Area under Receiver Operating Characteristic (AUROC) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | STR | LEX | GALC | INQUIRER | FUSION (SVM) | FUSION (R.F.) | EG-CNN | MTNL | $R^2$HP |
| *Clothing, Shoes & Jewelry* | 0.548 | 0.536 | 0.562 | *0.601** | 0.579 | 0.550 | 0.583 | 0.589 | **0.623** (+0.022) |
| *Electronics* | 0.583 | 0.549 | 0.588 | *0.616** | 0.577 | 0.580 | 0.611 | 0.613 | **0.661** (+0.045) |
| *Grocery & Gourmet Food* | 0.536 | 0.526 | 0.553 | 0.602 | 0.532 | 0.546 | 0.611 | *0.626** | **0.657** (+0.031) |
| *Health & Personal Care* | 0.558 | 0.523 | 0.559 | 0.610 | 0.591 | 0.562 | 0.613 | *0.620** | **0.683** (+0.063) |
| *Home & Kitchen* | 0.568 | 0.537 | 0.565 | 0.597 | 0.569 | 0.573 | 0.603 | *0.610** | **0.646** (+0.036) |
| *Movies & TV* | 0.603 | 0.558 | 0.621 | 0.634 | 0.603 | 0.607 | 0.648 | *0.652** | **0.713** (+0.061) |
| *Pet Supplies* | 0.560 | 0.542 | 0.585 | 0.603 | 0.548 | 0.558 | 0.580 | *0.629** | **0.692** (+0.063) |
| *Tools & Home Improvement* | 0.584 | 0.558 | 0.580 | 0.592 | 0.575 | 0.586 | 0.617 | *0.624** | **0.672** (+0.048) |
| *MACRO AVERAGE* | 0.568 | 0.541 | 0.577 | 0.607 | 0.572 | 0.570 | 0.608 | *0.620** | **0.668** (+0.048) |
| *MICRO AVERAGE (Primary)* | 0.571 | 0.543 | 0.580 | 0.609 | 0.573 | 0.574 | 0.614 | *0.625** | **0.675** (+0.049) |

Table 3: Results of performance (AUROC) of up-to-date methods on identifying helpful reviews evaluated by the test sets of Amazon. ($italic\ fonts^*$: the best performance among the baseline approaches; **bold fonts**: the state-of-the-art performance of all the approaches)

| Category (Yelp) | Area under Receiver Operating Characteristic (AUROC) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | STR | LEX | GALC | INQUIRER | FUSION (SVM) | FUSION (R.F.) | EG-CNN | MTNL | $R^2$HP |
| *Beauty & Spas* | 0.573 | 0.610 | 0.627 | 0.674 | 0.651 | 0.662 | 0.693 | *0.702** | **0.742** (+0.040) |
| *Health & Medical* | 0.585 | 0.617 | 0.638 | 0.676 | 0.632 | 0.631 | 0.680 | *0.697** | **0.725** (+0.028) |
| *Home Services* | 0.582 | 0.596 | 0.625 | 0.684 | 0.635 | 0.638 | 0.663 | *0.704** | **0.762** (+0.058) |
| *Restaurants* | 0.595 | 0.616 | 0.652 | 0.682 | 0.669 | 0.654 | 0.681 | *0.695** | **0.738** (+0.043) |
| *Shopping* | 0.582 | 0.618 | 0.660 | 0.699 | 0.672 | 0.685 | 0.682 | *0.719** | **0.784** (+0.065) |
| *MACRO AVERAGE* | 0.583 | 0.611 | 0.640 | 0.683 | 0.652 | 0.654 | 0.680 | *0.703** | **0.750** (+0.047) |
| *MICRO AVERAGE (Primary)* | 0.584 | 0.613 | 0.643 | 0.684 | 0.654 | 0.656 | 0.681 | *0.704** | **0.751** (+0.047) |

Table 4: Results of performance (AUROC) on identifying helpful reviews evaluated by the test sets of Yelp.

| Category (Amazon) | Squared Correlation Coefficient ($R^2$-score) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | STR | LEX | GALC | INQUIRER | FUSION (SVM) | FUSION (R.F.) | EG-CNN | MTNL | $R^2$HP |
| *Clothing, Shoes & Jewelry* | 0.265 | 0.337 | 0.398 | 0.588 | 0.490 | 0.506 | 0.558 | *0.613** | **0.678** (+0.065) |
| *Electronics* | 0.254 | 0.303 | 0.356 | 0.550 | 0.496 | 0.497 | 0.534 | *0.593** | **0.636** (+0.043) |
| *Grocery & Gourmet Food* | 0.242 | 0.319 | 0.408 | 0.450 | 0.426 | 0.425 | 0.469 | *0.506** | **0.570** (+0.064) |
| *Health & Personal Care* | 0.237 | 0.242 | 0.376 | 0.453 | 0.420 | 0.445 | 0.489 | *0.519** | **0.583** (+0.064) |
| *Home & Kitchen* | 0.236 | 0.298 | 0.330 | 0.464 | 0.439 | 0.461 | 0.502 | *0.588** | **0.644** (+0.056) |
| *Movies & TV* | 0.253 | 0.328 | 0.412 | 0.487 | 0.460 | 0.452 | 0.523 | *0.594** | **0.636** (+0.042) |
| *Pet Supplies* | 0.237 | 0.337 | 0.385 | 0.420 | 0.401 | 0.439 | 0.501 | *0.573** | **0.657** (+0.084) |
| *Tools & Home Improvement* | 0.234 | 0.301 | 0.387 | 0.437 | 0.447 | 0.502 | 0.532 | *0.591** | **0.624** (+0.033) |
| *MACRO AVERAGE* | 0.245 | 0.308 | 0.382 | 0.481 | 0.447 | 0.466 | 0.514 | *0.572** | **0.629** (+0.057) |
| *MICRO AVERAGE (Primary)* | 0.243 | 0.307 | 0.382 | 0.471 | 0.444 | 0.461 | 0.510 | *0.569** | **0.623** (+0.054) |

Table 5: Results of performance ($R^2$-score) on helpfulness voting regression evaluated by the test sets of Amazon.

| Category (Yelp) | Squared Correlation Coefficient ($R^2$-score) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | STR | LEX | GALC | INQUIRER | FUSION (SVM) | FUSION (R.F.) | EG-CNN | MTNL | $R^2$HP |
| *Beauty & Spas* | 0.383 | 0.484 | 0.511 | 0.537 | 0.549 | 0.518 | 0.650 | *0.662** | **0.727** (+0.065) |
| *Health & Medical* | 0.353 | 0.454 | 0.533 | 0.557 | 0.598 | 0.579 | 0.672 | *0.686** | **0.735** (+0.049) |
| *Home Services* | 0.364 | 0.452 | 0.516 | 0.554 | 0.592 | 0.601 | 0.670 | *0.695** | **0.741** (+0.046) |
| *Restaurants* | 0.396 | 0.438 | 0.483 | 0.501 | 0.556 | 0.597 | 0.610 | *0.623** | **0.682** (+0.059) |
| *Shopping* | 0.406 | 0.447 | 0.537 | 0.623 | 0.634 | 0.709 | 0.737 | *0.742** | **0.805** (+0.063) |
| *MACRO AVERAGE* | 0.380 | 0.455 | 0.516 | 0.554 | 0.586 | 0.601 | 0.668 | *0.682** | **0.738** (+0.056) |
| *MICRO AVERAGE (Primary)* | 0.383 | 0.454 | 0.516 | 0.557 | 0.587 | 0.606 | 0.670 | *0.682** | **0.740** (+0.058) |

Table 6: Results of performance ($R^2$-score) on helpfulness voting regression evaluated by the test sets of Yelp. ($italic\ fonts^*$: the best performance among the baseline approaches; **bold fonts**: the state-of-the-art performance of all the approaches)

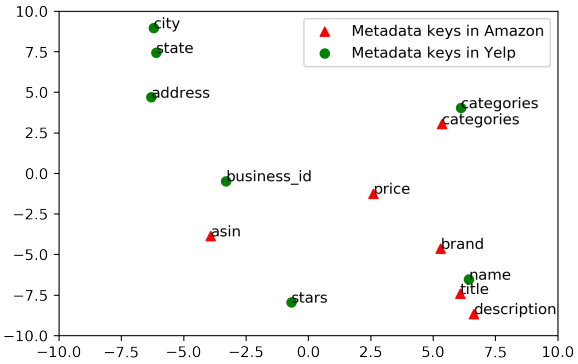


Figure 5: The 2D visualization of the embeddings of the metadata keys in *Amazon* and *Yelp*.

different datasets, the key embeddings draw near to each other if they are close in meaning, e.g., "title" (*Amazon*) and "name" (*Yelp*), or "asin" (*Amazon*) and "business_id" (*Yelp*).

## 5 Conclusion and Future Work

Driven by the intuition that the helpfulness of an online review should be fully aware of the metadata of its target product besides the textual content of the review itself, we take on the challenge of selecting the correct *key-value* product metadata to help predict the helpfulness of *free-text* reviews more precisely. To address the problem, we propose a novel framework in this paper, which is composed of two interdependent modules. Given a product, an agent (selector) learns from both the *keys* in the product metadata and one of its reviews to take an action that selects the correct *value*, and a successive network (predictor) makes the *free-text* review attend to this *value* to produce better neural representations for helpfulness prediction. We use two real-world datasets from Amazon and Yelp, respectively, to compare the performance of our framework with other mainstream methods on two tasks: helpfulness identification and regression of online reviews. Extensive results show that our framework can achieve state-of-the-art performance with substantial improvements. Further discussions demonstrate that it can not only provide better policies on selecting the correct *value* of product metadata but also acquire the embeddings of the *keys* in the product metadata.

We also believe the study on review helpfulness assessment could be as important as the topic of product recommendation, and several open problems are deserved to be explored in the future:

*User-specific and explainable recommendation of helpful reviews*: As different users may concern about various aspects of the products online, helpful review recommendation needs to be more user-specific and self-explainable.

*Enhancing the prediction of helpful reviews with unlabeled data*: As a small proportion of reviews could be heuristically regarded as helpful or unhelpful, it thus becomes a promising study to automatically predict the helpfulness of online reviews based on the small amount of labeled data and a vast amount of unlabeled data.

*Cross-domain helpfulness prediction of online reviews* (Chen et al., 2018b): Given that it costs a lot on manually annotating a sufficient number of helpful reviews in a new domain, we should explore effective approaches on transferring useful knowledge from limited labeled samples in another domain.

## References

Cen Chen, Minghui Qiu, Yinfei Yang, Jun Zhou, Jun Huang, Xiaolong Li, and Forrest Sheng Bao. 2018a. Review helpfulness prediction with embedding-gated CNN. *CoRR*, abs/1808.09896.

Cen Chen, Yinfei Yang, Jun Zhou, Xiaolong Li, and Forrest Sheng Bao. 2018b. Cross-domain review helpfulness prediction based on convolutional neural networks with auxiliary domain discriminators. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL'18)*, pages 602–607. Association for Computational Linguistics.

Gerardo Ocampo Diaz and Vincent Ng. 2018. Modeling and prediction of online product review helpfulness: a survey. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL'18)*, pages 698–708.

Miao Fan, Yue Feng, Mingming Sun, Ping Li, Haifeng Wang, and Jianmin Wang. 2018. Multi-task neural learning architecture for end-to-end identification of helpful reviews. In *Proceedings of the 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM'18)*, pages 343–350.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'16)*, pages 770–778.

Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion

trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web (WWW'16)*, pages 507–517. International World Wide Web Conferences Steering Committee.

Soo-Min Kim, Patrick Pantel, Tim Chklovski, and Marco Pennacchiotti. 2006. Automatically assessing review helpfulness. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP'06)*, pages 423–430. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP'14)*, pages 1746–1751. Association for Computational Linguistics.

Bing Liu and Lei Zhang. 2012. A survey of opinion mining and sentiment analysis. In *Mining text data*, pages 415–463. Springer.

Haijing Liu, Yang Gao, Pin Lv, Mengxue Li, Shiqiang Geng, Minglan Li, and Hao Wang. 2017. Using argument-based features to predict and analyse review helpfulness. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP'17)*, pages 1369–1374. Association for Computational Linguistics.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research (JMLR)*, 9(Nov):2579–2605.

Lionel Martin and Pearl Pu. 2014. Prediction of helpful reviews using emotions extraction. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI'14)*, pages 1551–1557. AAAI Press.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems (NIPS'13)*, pages 3111–3119.

Susan M. Mudambi and David Schuff. 2010. What makes a helpful online review? a study of customer reviews on amazon.com. *MIS Quarterly*, 34(1):185–200.

Deuk Hee Park, Hyea Kyeong Kim, Il Young Choi, and Jae Kyeong Kim. 2012. A literature review and classification of recommender systems research. *Expert Systems with Applications*, 39(11):10059–10072.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP'14)*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL'18)*, pages 2227–2237. Association for Computational Linguistics.

Sebastian Ruder. 2016. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems (NIPS'00)*, pages 1057–1063.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS'17)*, pages 5998–6008.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Wenting Xiong and Diane Litman. 2011. Automatically predicting peer-review helpfulness. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL'11)*, pages 502–507. Association for Computational Linguistics.

Wenting Xiong and Diane J Litman. 2014. Empirical analysis of exploiting review helpfulness for extractive summarization of online reviews. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING'14)*, pages 1985–1995. Association for Computational Linguistics.

Yinfei Yang, Yaowei Yan, Minghui Qiu, and Forrest Bao. 2015. Semantic analysis and helpfulness prediction of text for online product reviews. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL'15)*, pages 38–44. Association for Computational Linguistics.

# Learning Invariant Representations of Social Media Users

**Nicholas Andrews and Marcus Bishop**
Human Language Technology Center of Excellence
Johns Hopkins University
{noa,marcus.bishop}@jhu.edu

## Abstract

The evolution of social media users' behavior over time complicates user-level comparison tasks such as verification, classification, clustering, and ranking. As a result, naïve approaches may fail to generalize to new users or even to future observations of previously known users. In this paper, we propose a novel procedure to learn a mapping from short episodes of user activity on social media to a vector space in which the distance between points captures the similarity of the corresponding users' invariant features. We fit the model by optimizing a surrogate metric learning objective over a large corpus of unlabeled social media content. Once learned, the mapping may be applied to users not seen at training time and enables efficient comparisons of users in the resulting vector space. We present a comprehensive evaluation to validate the benefits of the proposed approach using data from Reddit, Twitter, and Wikipedia.

## 1 Introduction

Social media presents a number of challenges for characterizing user behavior, chief among them that the topics of discussion and their participants evolve over time. This makes it difficult to understand and combat harmful behavior, such as election interference or radicalization (Thompson, 2011; Mihaylov and Nakov, 2016; Ferrara et al., 2016; Keller et al., 2017).

This work focuses on the fundamental problem of learning to compare social media users. We propose a procedure to learn embeddings of small samples of users' online activity, which we call *episodes*. This procedure involves learning the embedding using a metric learning objective that causes episodes by the same author to map to nearby points. Through this embedding users may be efficiently compared using cosine similarity. This representation immediately enables several tasks:

**Verification.** Determining if two episodes have the same author.

**Classification.** Labeling authors via their k-nearest neighbors.

**Clustering.** Grouping users via off-the-shelf methods like k-means or agglomerative clustering.

**Ranking and retrieval.** Sorting episodes according to their distances to a given episode.

The problem considered in this paper is most closely related to author attribution on social media. However, prior work in this area has primarily focused on classifying an author as a member of a closed and typically small set of authors (Stamatatos, 2009; Schwartz et al., 2013; Shrestha et al., 2017). In this paper, we are concerned with an *open-world* setting where we wish to characterize an *unbounded* number of users, some observed at training time, some appearing only at test time. A further challenge is that the episodes being compared may be drawn from different time periods. With these challenges in mind, the primary contributions described in this paper are as follows:

§**3** A training strategy in which a user's history is dynamically sampled at training time to yield multiple short episodes drawn from different time periods as a means of learning invariant features of the user's identity;

§**4** A user embedding that can be trained end-to-end and which incorporates text, timing, and context features from a sequence of posts;

§**5** Reddit and Twitter benchmark corpora for open-world author comparison tasks, which are substantially larger than previously considered;

§**6** Large-scale author ranking and clustering experiments, as well as an application to Wikipedia sockpuppet verification.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1684–1695, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

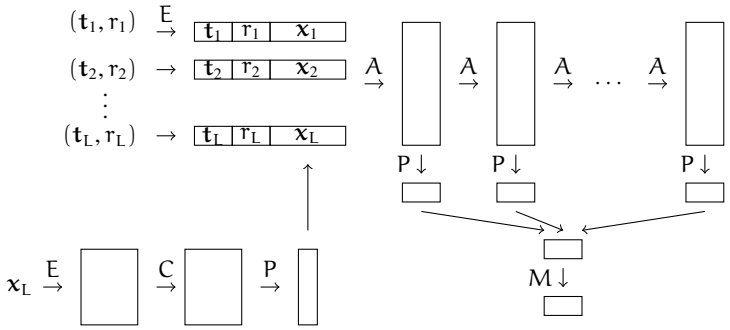


Figure 1: The map $f_\theta$ takes an episode as input and outputs a vector. Here A denotes a multi-head self-attention layer, C a stack of 1D convolutions, E an embedding lookup, M an MLP, and P a pooling layer.

## 2 Preliminaries

Broadly speaking, a corpus of social media data consists of the *actions* of a number of users. Each action consists of all available information from a given platform detailing what exactly the user did, which for purposes of this work we take to include: (1) a timestamp recording when the action occurred, from which we extract a tuple $\mathbf{t}$ of temporal features, (2) unstructured text content $\mathbf{x}$ of the action, and (3) a categorical feature $r$ specifying the context of the action. Thus an action is a tuple of the form $(\mathbf{t}, \mathbf{x}, r)$. This formulation admits all three platforms considered in this work and therefore serves as a good starting point. However, incorporating features specific to particular platforms, such as image, network, and moderation features, might also provide useful signal.

In our experiments we use a data-driven subword representation (Kudo, 2018) of $\mathbf{x}$, which admits multilingual and non-linguistic content, as well as misspellings and abbreviations, all of which useful in characterizing authors. We use a simple discrete time feature for $\mathbf{t}$, namely the hour of the day, although others might be helpful, such as durations between successive actions. In our Reddit experiments we take $r$ to be the subreddit to which a comment was posted. On Twitter we take $r$ to be a flag indicating whether the post was a tweet or a retweet.

## 3 Learning Invariant Representations

We organize the actions of each user into short sequences of chronologically ordered and ideally contiguous actions, which we call *episodes*. This paper is concerned with devising a notion of distance between episodes for which episodes by the same author are closer to one another than episodes by different authors. Such a distance function must necessarily be constructed on the basis of *past* social media data. But in the future, authors' behavior will evolve and new authors will emerge.

We would like episodes by the same author to be nearby, irrespective of when those episodes took place, possibly *future* to the creation of the distance function. A given user will discuss different topics, cycle through various moods, develop new interests, and so on, but distinctive features like uncommon word usage, misspellings, or patterns of activity will persist for longer and therefore provide useful signal for the distance function.

We would also like the distance to be meaningful when applied to episodes by users who didn't exist when the distance function was created. To this end, the features it considers must necessarily generalize to new users. For example, common stylometric features will be shared by many users, including new users, but their particular combination is distinctive of particular users (Orebaugh and Allnutt, 2009; Layton et al., 2010).

Rather than heuristically defining such a distance function, for example, based on word overlap between the textual content of the episodes, we instead introduce a parameterized embedding $f_\theta$ shown in Figure 1 that provides a vector representation of an episode. Then the desired distance between episodes can be taken to be the distance between the corresponding vectors. We fit the embedding $f_\theta$ using *metric learning* to simultaneously decrease the distance between episodes by the same user and increase the distance between

episodes by different users (Bromley et al., 1994; Wang et al., 2014).

But doing so requires knowledge of the true author of an episode, something which is not generally available. Therefore we take account names to be an approximation of latent authorship. Of course, account names are not always a reliable indicator of authorship on social media, as the same individual may use multiple accounts, and multiple individuals may use the same account. As such, we expect a small amount of label noise in our data, to which neural networks have proven robust in several domains (Krause et al., 2016; Rolnick et al., 2017).

We fit $f_\theta$ to a corpus of social media data using stochastic gradient descent on batches of examples, where each example consists of an episode of a given length drawn uniformly at random from the *full history* of each user's actions.[1] By construction, a metric learning objective with this batching scheme will encourage the embedding of episodes drawn from the same user's history to be close. In order to accomplish this, the model will need to distinguish between ephemeral and invariant features of a user. The invariant features are those that enable the model to consistently distinguish a given users' episodes from those of *all other users*.

## 4 The Model

We now describe a mapping $f_\theta$ parameterized by a vector $\theta$ from the space of user episodes to $\mathbb{R}^D$. The model is illustrated in Figure 1. This embedding induces a notion of distance between episodes that depends on which of the two proposed loss functions from §4.2 is used to train $f_\theta$. We illustrate the embeddings resulting from both losses in Figure 2.

### 4.1 The encoder

One approach to define $f_\theta$ might be to manually define features of interest, such as stylometric or surface features (Solorio et al., 2014; Sari et al., 2018). However, when large amounts of data are available, it is preferable to use a data-driven approach to representation learning. Therefore we define $f_\theta$ using a neural network as follows. The network is illustrated in Figure 1.

[1]Different metric learning methods will sample users in different ways, for example to ensure a given ratio of examples of the same class. In this work we simply sample users uniformly at random.

**Encoding actions.** First, we embed each action $(t, x, r)$ of an episode. We encode the time features $t$ and the context $r$, both assumed to be discrete, using a learned embedding lookup. We next embed every symbol of $x$, again using a learned embedding lookup, and apply one-dimensional convolutions of increasing widths over this list of vectors, similar to Kim (2014); Shrestha et al. (2017). We then apply the relu activation and take the componentwise maximum of the list of vectors to reduce the text content to a single, fixed-dimensional vector. We optionally apply dropout at this stage if training. Finally, we concatenate the time, text, and context vectors to yield a single vector representing the action.

**Embedding episodes.** Next we combine the vector representations of the actions of an episode. For this purpose, one option is a recurrent neural network (RNN). However, recurrent models are biased due to processing inputs sequentially, and suffer from vanishing and exploding gradients. Therefore we propose the use of self-attention layers, which avoid the sequential biases of RNNs and admit efficient implementations.

In our particular formulation, we use several layers of multi-head self-attention, each taking the output of the previous layer as input; architectural details of the encoder layers follow those of the Transformer architecture proposed by Vaswani et al. (2017). We apply mean pooling after every layer to yield layer-specific embeddings, which we concatenate. We project to the result to the desired embedding dimension $D$ using an MLP, both its input and output batch normalized (Ioffe and Szegedy, 2015).

### 4.2 The loss function

For the purpose of training the embedding $f_\theta$ we compose it with a discriminative classifier $g_\phi : \mathbb{R}^D \to \mathbb{R}^Y$ with parameters $\phi$ predicting the author of an episode, where $Y$ is the number of authors in the training set. We estimate $\theta$ and $\phi$ jointly using a standard cross-entropy loss on a corpus of examples with their known authors. Once the model is trained, the auxiliary projection $g_\phi$ is discarded. Two possibilities for $g_\phi$ are proposed below.

**Softmax (SM).** We introduce a weight matrix $W \in \mathbb{R}^{Y \times D}$ and define the map $g_\phi(z) = \mathrm{softmax}(Wz)$ with parameters $\phi = W$. When using this loss function, one compares embed-

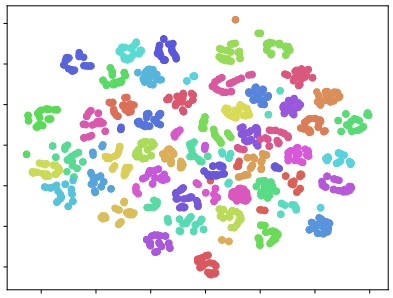

(a) Embeddings obtained using **SM** loss.

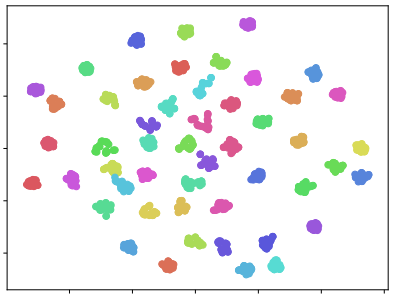

(b) Embeddings obtained using **AM** loss.

Figure 2: Projections of embeddings of user episodes. Each point is the result of mapping an episode to a single point in $\mathbb{R}^{512}$ and projected to $\mathbb{R}^2$ using t-SNE. The colors of the points correspond with the 50 different authors of the underlying episodes. We emphasize that the episodes shown here were not seen by the model at training time.

dings using Euclidean distance.

**Angular margin (AM).** Following Deng et al. (2019) we again introduce a weight matrix $\boldsymbol{W} \in \mathbb{R}^{Y \times D}$ whose rows now serve as *class centers* for the training authors. Given the embedding $\boldsymbol{z} \in \mathbb{R}^D$ of an episode, let $\boldsymbol{z}' = \frac{\boldsymbol{z}}{\|\boldsymbol{z}\|}$ be the normalization of $\boldsymbol{z}$ and let $\boldsymbol{W}'$ be obtained from $\boldsymbol{W}$ by normalizing its rows. Then the entries of $\boldsymbol{w} = \boldsymbol{W}'\boldsymbol{z}'$ give the cosines of the angles between $\boldsymbol{z}$ and the class centers. Let $\boldsymbol{w}'$ be obtained from $\boldsymbol{w}$ by modifying the entry corresponding with the correct author by adding a fixed margin $m > 0$ to the corresponding angle.[2] Finally, define $g_\Phi(\boldsymbol{z}) = \text{softmax}(s\boldsymbol{w}')$ where $s > 0$ is a fixed *scale constant*. When using this loss function, one compares embeddings using cosine similarity.

## 5 Corpora for Large-Scale Author Identification

### 5.1 Reddit benchmark

Reddit is a large, anonymous social media platform with a permissive public API. Using Reddit consists of reading and posting *comments*, which consist of informal text, primarily in English, each appearing within a particular *subreddit*, which we treat as a categorical feature providing useful contextual signal in characterizing users.

We introduce a new benchmark author identification corpus derived from the API (Gaffney and Matias, 2018) containing Reddit comments by 120,601 active users for training and 111,396 held-out users for evaluation. The training split contains posts published in `2016-08` while the evaluation split contains posts published in `2016-09`. In both cases, we restrict to users publishing at least 100 comments but not more than 500. The lower bound ensures that we have sufficient evidence for any given user for training, while the upper bound is intended to mitigate the impact of bots and atypical users. The evaluation split is disjoint from the training split and contains comments by 42,121 novel authors not contributing to the training split.

**Validation**. For model selection, we use the first 75% of each user's chronologically ordered posts from the training set, with the final 25% reserved for validation. For example, in our ranking experiments described in §6.3 we use these held-out comments as candidate targets, using ranking performance to inform hyper-parameter choice.

### 5.2 Twitter benchmark

The microblog domain is sufficiently distinct from Reddit that it is suitable as an additional case study. For this purpose, we sample 169,663 active Twitter users from three months of 2016 as separate training, development, and test sets (`2016-08` through `2016-10`). We use three months because we rely on a sub-sampled collection of Twitter, as little as 1% of all posts published, resulting in significantly fewer posts by each user than on Reddit. Another consequence of this sub-sampling is that the collection violates our assumptions regarding *contiguous* user actions.

[2] One way to calculate $\cos(\theta + m)$ from $\cos\theta$ is $\cos\theta\cos m - \sin\theta\sin m$ where $\sin\theta$ is calculated as $\sqrt{1 - \cos^2\theta}$. Note that this calculation discards the sign of $\theta$.

## 6 Experiments

In the experiments described below, we refer to our method as IUR for Invariant User Representations.

### 6.1 Baseline methods

In order to validate the merit of each of our modeling contributions, we compare against three baseline models described below. To the best of our knowledge, we are the first to consider using metric learning to learn embeddings from episodes of user activity. We are also the first to consider doing so in open-world and large-scale settings. As such, the neural baseline described below uses the training scheme proposed in this paper, and was further improved to be more competitive with the proposed model.

**Neural author identification.** We use the architecture proposed by Shrestha et al. (2017) for closed-set author attribution in place of our $\mathtt{f}_\theta$. At the level of individual posts this architecture is broadly similar to ours in that it applies 1D convolutions to the text content. To extend it to episodes of comments, we simply concatenate the text content into a single sequence with a distinguished end-of-sequence marker. Note that the timing and context features may also be viewed as sequences, and in experiments with these features we run a separate set of one-dimensional filters over them. All max-over-time pooled features are concatenated depthwise. By itself, this model failed to produce useful representations; we found it necessary to apply the batch-normalized MLP described in §4.1 to the output layer before the loss. To train the model, we follow the procedure described in §4.2 to compose the embedding with the **SM** loss function, optimize the composition using cross-entropy loss, and discard the **SM** factor after training.

**Document vectors.** By concatenating all the textual content of an episode we can view the episode as a single document. This makes it straightforward to apply classical document indexing methods to the resulting pseudo-document. As a representative approach, we use TFIDF with cosine distance (Robertson, 2004). We note that TFIDF is also well-defined with respect to arbitrary bags-of-items, and we make use of this fact to represent a user according to the sequence of subreddits to which they post as a further baseline in §6.3.

**Author verification models.** We use the SCAP $\mathfrak{n}$-gram profile method of Frantzeskou et al. (2007). Two episodes are compared by calculating the size of the intersection of their $\mathfrak{n}$-gram profiles. We use profiles of fixed length 64 in our experiments.

### 6.2 Model hyperparameters and training

Below we list our hyperparameter choices for the IUR model, which we define in §4.

For both Twitter and Reddit, we estimate the sub-word vocabulary on training data using an inventory of 65,536 word pieces, including a distinguished end-of-sequence symbol. We truncate comments to 32 word pieces, padding if necessary.[3] We restrict to the 2048 most popular subreddits, mapping all others to a distinguished `unk` symbol. We encode word pieces and subreddits as 256-long vectors. The architecture for the text content uses four convolutions of widths 2, 3, 4, 5 with 256 filters per convolution. We use two layers of self-attention with 4 attention heads per layer, and hidden layers of size 512. Other details such as use of layer normalization match the recommendations of Vaswani et al. (2017).

We train all variations of the IUR for a fixed budget of 200,000 iterations of stochastic gradient descent with momentum 0.9 and a piecewise linear learning rate schedule that starts at 0.1 and is decreased by a factor of 10 at 100,000 and 150,000 iterations. The final MLP has one hidden layer of dimension 512 with output also of dimension $\mathrm{D} = 512$. For the angular margin loss we take $\mathfrak{m} = 0.5$ and $\mathrm{s} = 64$ as suggested in Deng et al. (2019).

### 6.3 Reddit ranking experiment

Given a query episode by a known user, our author ranking experiment consists of returning a list of target episodes ranked according to their similarity to the query. The problem arises in the moderation of social media content, when say, a user attempts to circumvent an account ban by using another account.

**Experimental setup.** Recall that we train all Reddit models on the `2016-08` split. In this experiment we draw episodes from the first half of `2016-09` as *queries* and the second half of `2016-09` as *targets*. Specifically, for

[3] In experiments not reported here, we have found that increasing the number of subwords per action increases performance but at the cost of slower training.

| Input Features | Method | MRR (⇑) | MR (⇓) | R@1 (⇑) | R@2 | R@4 | R@8 |
|---|---|---|---|---|---|---|---|
| text only | SCAP | 0.0057 | 31292 | 0.0035 | 0.004 | 0.0075 | 0.0085 |
| | TF-IDF (word) | 0.071 | 5548 | 0.048 | 0.065 | 0.084 | 0.11 |
| | TF-IDF (char trigram) | 0.07 | 6264 | 0.05 | 0.066 | 0.081 | 0.1 |
| | Shrestha et al. (2017) | 0.081 | 660 | 0.052 | 0.071 | 0.094 | 0.12 |
| | IUR | **0.2** | **88** | **0.15** | **0.19** | **0.24** | **0.29** |
| subreddit only | TF-IDF | 0.1 | 305 | 0.068 | 0.091 | 0.12 | 0.16 |
| | Shrestha et al. (2017) | 0.18 | 110 | 0.12 | 0.16 | 0.21 | 0.26 |
| | IUR | **0.21** | **64** | **0.15** | **0.2** | **0.24** | **0.3** |
| text, subreddit, time | Shrestha et al. (2017) | 0.39 | 8 | 0.31 | 0.38 | 0.45 | 0.51 |
| | IUR (softmax loss) | 0.38 | 9 | 0.31 | 0.38 | 0.44 | 0.49 |
| | IUR (recurrent encoder) | 0.34 | 17 | 0.27 | 0.33 | 0.39 | 0.44 |
| | IUR (without time) | 0.48 | 3 | 0.41 | 0.48 | 0.55 | 0.61 |
| | IUR | **0.52** | **2** | **0.44** | **0.52** | **0.59** | **0.65** |

Table 1: Reddit author ranking results with 111,396 possible targets. The best results for each feature group are in printed in **bold**. The proposed Invariant User Representations are denoted IUR, with variations of the full model noted in parenthesis. MRR stands for the mean reciprocal rank, MR for median rank, and R@k stands for recall at the top k ranked episodes. Larger numbers are better (⇑) except for MR where lower rank is better (⇓). Metrics are computed over 25,000 queries.

each of 25,000 randomly selected users from the `2016-09` split we randomly draw a query episode of length 16 from among those posts published by that user before `2016-09-15`. Then for each of the 111,396 users in the `2016-09` split we randomly draw a target episode of length 16 from among those posts published by that user on or after `2016-09-15`. For each query, the goal of the experiment is to rank the targets according to their likelihoods of being the *unique* target composed by the author of the query.

We compare models using mean reciprocal rank (MRR), median rank (MR), and recall-at-k (R@k) for various k. The MRR is the mean over all 25,000 queries of the reciprocal of the position of the correct target in the ranked list. The MR is the median over the queries of the position of the correct target. The R@k is the proportion of the queries for which the correct target appears among the first k ranked targets.

**Results.** The results of this experiment are shown in Table 1. For each combination of features considered, the rankings based on the proposed IUR embeddings consistently outperform all methods considered, both neural and classical. We also report results on several variations of our model, noted in parenthesis. First, using the proposed architecture for $f_\theta$ but the softmax loss results in ranking performance comparable to the baseline system. Second, using a recurrent architecture rather than self-attention to aggregate information across an episode results in significantly worse performance.[4] Finally, omitting time features results in worse performance.

**Performance on novel users.** As described above, the experiments presented in Table 1 involved ranking episodes by test authors, some of whom had been seen during training, and some new to the model. To better understand the ability of the proposed embedding to generalize to new users, we performed a further evaluation in which authors were restricted to those *not* seen at training time. For the IUR incorporating all features, this yielded a MRR of 0.50, while our extension of Shrestha et al. (2017) obtains 0.38 for the same queries. Both methods produce salient embeddings of novel users, but IUR retains an edge over the baseline.

**Varying episode length.** As described above, the experiments presented in Table 1 involved episodes of length exactly 16. In Figure 3, we report results of a further ranking experiment in which we vary the episode length, both at training time and at ranking time. For both the proposed IUR and our extension of Shrestha et al. (2017), performance increases as episode length increases. Furthermore, even for the shortest episodes considered, the proposed approach performs better. This illustrates that the choice of episode length should be decided on an application-specific basis. For example, for social media moderation, it

[4]We choose RNN hyper-parameters such that the numbers of parameters of both models are on the same order of magnitude.

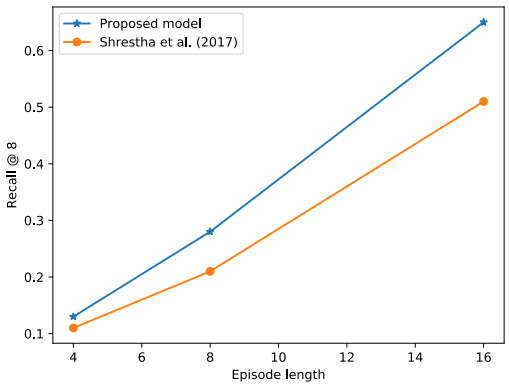


Figure 3: We report Recall@8 for different episode lengths using all features.

may be desirable to quickly identify problematic users on the basis of as few posts as possible.

### 6.4 Twitter ranking experiment

We repeat the experiment described in §6.3 using data from Twitter in place of Reddit, and with the further difference that the queries were drawn from `2016-08` and the targets from `2016-10` as a mitigation of Twitter's 1% censorship. Unlike the Reddit dataset, all three data splits contain posts by exactly the same authors. The results are shown in Table 2.

### 6.5 Wikipedia sockpuppet verification

In this section we describe an experiment for the task of sockpuppet verification on Wikipedia using the dataset collected by Solorio et al. (2014). Wikipedia allows editors to open cases against other editors for using suspected *sockpuppet* accounts to promote their contributions. We have reorganized the dataset into pairs of episodes by different accounts. Half of our examples contain a pair deemed by the community to have the same author, while half have been deemed to have different authors. The task is to predict whether a pair of episodes was composed by the same author.

We are interested in whether the text-only version of our IUR model, trained on Reddit data, is able to transfer effectively to this task. This domain is challenging because in many cases sockpuppet accounts are trying to hide their identity, and furthermore, Wikipedia talk pages contain domain-specific markup which is difficult to reliably strip or normalize. Naturally we expect that the identities of Wikipedia editors do not overlap with Reddit authors seen at training time, since the data is drawn from different time periods and from different platforms.

As a baseline, we compare to BERT, a generic text representation model trained primarily on Wikipedia article text (Devlin et al., 2018). While BERT is not specifically trained for author recognition tasks, BERT has obtained state-of-the-art results in many pairwise text classification tasks including natural language inference, question pair equivalence, question answering, and paraphrase recognition. The BERT model used here has 110 million parameters compared to 20 million for our embedding.

**Setup.** Because many comments are short, we preprocess the data to ensure that each comment has at least 5 whitespace-separated tokens. We restrict to users contributing at least 8 such comments. This left us with 180 cases which we split into 72 for training, and 54 each for validation and testing. We fine-tune both the cased and uncased pretrained English BERT models for our sockpuppet detection task using public models and software.[5] In order to combine the comments comprising an episode for BERT, we explored different strategies, including encoding each comment separately. We found that simply combining comments together and using a long sequence length of 512 gave the best validation performance. For our model, we fine-tune by fitting an MLP on top of our embeddings using binary cross entropy and keeping other parameters fixed. Both methods are tuned on validation data, and the best hyperparameter configuration is then evaluated on held-out test data.

**Results.** Results are reported in Table 3. The best validation performance is obtained by the cased BERT model. However, both BERT models appear to overfit the training data as test performance is significantly lower. Regarding the proposed IUR model, we see that its performance on validation data is comparable to BERT while generalizing better to held-out test data. For reference, Solorio et al. (2013) report accuracy of 68.83 using the same data using a SVM with hand-crafted features; however, neither their experimental splits nor their model are available for purposes of a direct comparison.

[5] https://github.com/google-research/bert

| Method | MRR (⇑) | MR (⇓) | R@1 (⇑) | R@2 | R@4 | R@8 | R@16 | R@32 |
|---|---|---|---|---|---|---|---|---|
| TF-IDF (word) | 0.060 | 4447 | 0.048 | 0.057 | 0.067 | 0.077 | 0.092 | 0.110 |
| TF-IDF (char trigram) | 0.070 | 1622 | 0.052 | 0.064 | 0.078 | 0.095 | 0.120 | 0.140 |
| SCAP | 0.049 | 3582 | 0.037 | 0.044 | 0.053 | 0.065 | 0.08 | 0.098 |
| Shrestha et al. (2017) | 0.056 | 577 | 0.030 | 0.050 | 0.070 | 0.090 | 0.130 | 0.140 |
| IUR (text, time, context) | 0.113 | 179 | 0.073 | 0.100 | 0.130 | 0.170 | 0.224 | 0.287 |
| IUR (text only) | 0.117 | 161 | 0.077 | 0.100 | 0.133 | 0.176 | 0.228 | 0.293 |
| IUR (text, time) | **0.119** | **154** | **0.078** | **0.103** | **0.137** | **0.178** | **0.234** | **0.305** |

Table 2: Twitter ranking results with 25,000 queries and with 169,663 possible targets.

| | Validation | Test |
|---|---|---|
| Majority baseline | 0.5 | 0.5 |
| BERT (uncased) | 0.72 | 0.65 |
| BERT (cased) | **0.76** | 0.61 |
| IUR (text-only) | 0.74 | **0.72** |

Table 3: Validation and test accuracy for the Wikipedia sockpuppet task. Best results in **bold**.

| | NMI | H | C |
|---|---|---|---|
| Shrestha et al. (2017) | 0.54 | 0.39 | 0.74 |
| IUR | **0.76** | **0.70** | **0.84** |

Table 4: Clustering performance on Reddit episodes using embeddings obtained with different methods.

### 6.6 Clustering users

For certain tasks it is useful to identify groups of accounts shared by the same author or to identify groups of accounts behaving in a similar fashion (Solorio et al., 2013; Tsikerdekis and Zeadally, 2014). To this end, we experiment with how well a clustering algorithm can partition authors on the basis of the cosine similarity of their IUR episode embeddings.

**Procedure.** Using the pre-trained Reddit IUR model, we embed five episodes of length 16 by 5000 users selected uniformly at random, all drawn from the held-out `2016-09` split. The embeddings are clustered using affinity propagation, hiding both the identities of the users as well as the true number of users from the algorithm (Frey and Dueck, 2007). Ideally the algorithm will arrive at 5000 clusters, each containing exactly five episodes by same author. Clustering performance is evaluated using mutual information (NMI), homogeneity (H), and completeness (C) (Rosenberg and Hirschberg, 2007). NMI involves a ratio of the mutual information of the clustering and ground truth. Homogeneity is a measure of cluster purity. Completeness measures the extent to which data points by the same author are elements of the same cluster. All three measures lie in interval $[0, 1]$ where 1 is best. The results are shown in Table 4.

## 7 Related Work

This work considers the problem of learning to compare users on social media. A related task which has received considerably more attention is predicting user attributes (Han et al., 2014; Sap et al., 2014; Dredze et al., 2013; Culotta et al., 2015; Volkova et al., 2015; Goldin et al., 2018). The inferred user attributes have proven useful for social science and public health research (Mislove et al., 2011; Morgan-Lopez et al., 2017). While author attributes like gender or political leaning may be useful for population-level studies, they are inadequate for identifying particular users.[6]

More generally, learning representations for downstream tasks using unsupervised training has recently emerged as an effective way to mitigate the lack of task-specific training data (Peters et al., 2018; Devlin et al., 2018). In the context of social media data, unsupervised methods have also been explored to obtain vector representations of individual posts on Twitter (Dhingra et al., 2016; Vosoughi et al., 2016). Our approach is distinguished from this prior work in several respects. First, we embed episodes consisting of multiple documents, which involves aggregating features. Second, for each document, we encode both textual features as well as associated meta-data. Finally, our training procedure is discriminative, embedding episodes into a vector space with an im-

[6]We leave as future work the question of whether the episode embeddings proposed in this paper are useful for attribute prediction.

mediately meaningful distance.

When social network structure is available, for example on Twitter via *followers*, it may be used to learn user embeddings (Tang et al., 2015; Grover and Leskovec, 2016; Kipf and Welling, 2016). Graph representations have successfully been combined with content-based features; for example, Benton et al. (2016) propose matrix decomposition methods that exploit complementary features of Twitter authors. Graph-based embeddings have proven useful in downstream applications such as entity linking (Yang et al., 2016). However, such methods are not applicable when network structure is unavailable or unreliable, such as with new users or on social media platforms like Reddit. In this work, we are motivated in part by adversarial settings such as moderation, where it is desirable to quickly identify the authorship of novel users on the basis of sparse evidence.[7]

The most closely related work is author identification on social media. However, previous work in this area has largely focused on distinguishing among small, closed sets of authors rather than the open-world setting of this paper (Mikros and Perifanos, 2013; Ge et al., 2016). For example, Schwartz et al. (2013) consider the problem of assigning single tweets to one of a closed set of 1000 authors. Overdorf and Greenstadt (2016) consider the problem of cross-domain authorship attribution and consider 100 users active on multiple platforms. In a different direction, Sari et al. (2018) seek to identify stylistic features contributing to successful author identification and consider a closed set of 62 authors. In contrast, the present work is concerned with problems involving several orders of magnitude more authors. This scale precludes methods where similarity between examples is expensive to compute, such as the method of Koppel and Winter (2014).

Prior work on detecting harmful behavior like hate speech has focused on individual documents such as blog posts or comments (Spertus, 1997; Magu et al., 2017; Pavlopoulos et al., 2017; Davidson et al., 2017; de la Vega and Ng, 2018; Basile et al., 2019; Zampieri et al., 2019). Recently, there have been some efforts to incorporate user-level information. For example, for the supervised task of abuse detection, Mishra et al. (2018) find consistent improvements from incorporating user-level features.

## 8 Conclusion

Learning meaningful embeddings of social media users on the basis of short episodes of activity poses a number of challenges. This paper describes a novel approach to learning such embeddings using metric learning coupled with a novel training regime designed to learn invariant user representations. Our experiments show that the proposed embeddings are robust with respect to both novel users and data drawn from future time periods. To our knowledge, we are the first to tackle open-world author ranking tasks by learning a vector space with a meaningful distance.

There are several natural extensions of this work. An immediate extension is to further scale up the experiments to Web-scale datasets consisting of millions of users, as has been successfully done for face recognition (Kemelmacher-Shlizerman et al., 2016). Sorting episodes according to their distances to a query can be made efficient using a number of approximate nearest neighbor techniques (Indyk and Motwani, 1998; Andoni and Indyk, 2006).

We are also considering further applications of the proposed approach beyond those in this paper. For example, by restricting the features considered in the encoder to text-alone or text and temporal features, it would be interesting to explore cross-domain author attribution (Stamatatos et al., 2018). It would also be interesting to explore community composition on the basis of the proposed embeddings (Newell et al., 2016; Waller and Anderson, 2019).

Finally, it bears mentioning that the proposed model presents a double-edged sword: methods designed to identify users engaging in harmful behavior could also be used to identify authors with legitimate reasons to remain anonymous, such as political dissidents, activists, or oppressed minorities. On the other hand, methods similar to the proposed model could be developed for such purposes and not shared with the broader community. Therefore, as part of our effort to encourage positive applications, we release source code to reproduce our key results.[8]

[7] Incorporating social network information in our model as additional features is in principle straightforward, requiring only minor architectural changes to the model; the metric learning procedure would otherwise remain the same.

[8] http://github.com/noa/iur.

## References


Alexandr Andoni and Piotr Indyk. 2006. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 459–468. IEEE Computer Society.

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Manuel Rangel Pardo, Paolo Rosso, and Manuela Sanguinetti. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 54–63.

Adrian Benton, Raman Arora, and Mark Dredze. 2016. Learning multiview embeddings of twitter users. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 14–19.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1994. Signature verification using a "siamese" time delay neural network. In *Advances in neural information processing systems*, pages 737–744.

Aron Culotta, Nirmal Ravi Kumar, and Jennifer Cutler. 2015. Predicting the demographics of twitter users from website traffic data. In *AAAI*, pages 72–78.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Eleventh International AAAI Conference on Web and Social Media*.

Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. 2019. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4690–4699.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Bhuwan Dhingra, Zhong Zhou, Dylan Fitzpatrick, Michael Muehl, and William Cohen. 2016. Tweet2vec: Character-based distributed representations for social media. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 269–274.

Mark Dredze, Michael J Paul, Shane Bergsma, and Hieu Tran. 2013. Carmen: A twitter geolocation system with applications to public health. In *Workshops at the Twenty-Seventh AAAI Conference on Artificial Intelligence*.

Emilio Ferrara, Onur Varol, Filippo Menczer, and Alessandro Flammini. 2016. Detection of promoted social media campaigns. In *tenth international AAAI conference on web and social media*.

Georgia Frantzeskou, Efstathios Stamatatos, Stefanos Gritzalis, Carole E Chaski, and Blake Stephen Howald. 2007. Identifying authorship by byte-level n-grams: The source code author profile (scap) method. *International Journal of Digital Evidence*, 6(1):1–18.

Brendan J Frey and Delbert Dueck. 2007. Clustering by passing messages between data points. *science*, 315(5814):972–976.

Devin Gaffney and J Nathan Matias. 2018. Caveat emptor, computational social science: Large-scale missing data in a widely-published reddit corpus. *PloS one*, 13(7):e0200162.

Zhenhao Ge, Yufang Sun, and Mark JT Smith. 2016. Authorship attribution using a neural network language model. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Gili Goldin, Ella Rabinovich, and Shuly Wintner. 2018. Native language identification with user generated content. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3591–3601, Brussels, Belgium. Association for Computational Linguistics.

Aditya Grover and Jure Leskovec. 2016. Node2vec: Scalable feature learning for networks. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 855–864, New York, NY, USA. ACM.

Bo Han, Paul Cook, and Timothy Baldwin. 2014. Text-based twitter user geolocation prediction. *Journal of Artificial Intelligence Research*, 49:451–500.

Piotr Indyk and Rajeev Motwani. 1998. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613. ACM.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Franziska B Keller, David Schoch, Sebastian Stier, and JungHwan Yang. 2017. How to manipulate social media: Analyzing political astroturfing using ground truth data from south korea. In *Eleventh International AAAI Conference on Web and Social Media*.

Ira Kemelmacher-Shlizerman, Steven M Seitz, Daniel Miller, and Evan Brossard. 2016. The megaface benchmark: 1 million faces for recognition at scale. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4873–4882.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Moshe Koppel and Yaron Winter. 2014. Determining if two documents are written by the same author. *Journal of the Association for Information Science and Technology*, 65(1):178–187.

Jonathan Krause, Benjamin Sapp, Andrew Howard, Howard Zhou, Alexander Toshev, Tom Duerig, James Philbin, and Li Fei-Fei. 2016. The unreasonable effectiveness of noisy data for fine-grained recognition. In *European Conference on Computer Vision*, pages 301–320. Springer.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. *arXiv preprint arXiv:1804.10959*.

Robert Layton, Paul Watters, and Richard Dazeley. 2010. Authorship attribution for twitter in 140 characters or less. In *2010 Second Cybercrime and Trustworthy Computing Workshop*, pages 1–8. IEEE.

Rijul Magu, Kshitij Joshi, and Jiebo Luo. 2017. Detecting the hate code on social media. In *Eleventh International AAAI Conference on Web and Social Media*.

Todor Mihaylov and Preslav Nakov. 2016. Hunting for troll comments in news community forums. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 399–405.

George K Mikros and Kostas Perifanos. 2013. Authorship attribution in greek tweets using author's multilevel n-gram profiles. In *2013 AAAI Spring Symposium Series*.

Pushkar Mishra, Marco Del Tredici, Helen Yannakoudakis, and Ekaterina Shutova. 2018. Author profiling for abuse detection. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1088–1098.

Alan Mislove, Sune Lehmann, Yong-Yeol Ahn, Jukka-Pekka Onnela, and J Niels Rosenquist. 2011. Understanding the demographics of twitter users. In *Fifth international AAAI conference on weblogs and social media*.

Antonio A Morgan-Lopez, Annice E Kim, Robert F Chew, and Paul Ruddle. 2017. Predicting age groups of twitter users based on language and metadata features. *PloS one*, 12(8):e0183537.

Edward Newell, David Jurgens, Haji Mohammad Saleem, Hardik Vala, Jad Sassine, Caitrin Armstrong, and Derek Ruths. 2016. User migration in online social networks: A case study on reddit during a period of community unrest. In *Tenth International AAAI Conference on Web and Social Media*.

Angela Orebaugh and Jeremy Allnutt. 2009. Classification of instant messaging communications for forensics analysis. *The International Journal of Forensic Computer Science*, 1:22–28.

Rebekah Overdorf and Rachel Greenstadt. 2016. Blogs, twitter feeds, and reddit comments: Cross-domain authorship attribution. *Proceedings on Privacy Enhancing Technologies*, 2016(3):155–171.

John Pavlopoulos, Prodromos Malakasiotis, and Ion Androutsopoulos. 2017. Deep learning for user comment moderation. *arXiv preprint arXiv:1705.09993*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237.

Stephen Robertson. 2004. Understanding inverse document frequency: on theoretical arguments for idf. *Journal of documentation*, 60(5):503–520.

David Rolnick, Andreas Veit, Serge Belongie, and Nir Shavit. 2017. Deep learning is robust to massive label noise. *arXiv preprint arXiv:1705.10694*.

Andrew Rosenberg and Julia Hirschberg. 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*.

Maarten Sap, Gregory Park, Johannes Eichstaedt, Margaret Kern, David Stillwell, Michal Kosinski, Lyle Ungar, and Hansen Andrew Schwartz. 2014. Developing age and gender predictive lexica over social media. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1146–1151.

Yunita Sari, Mark Stevenson, and Andreas Vlachos. 2018. Topic or style? exploring the most useful features for authorship attribution. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 343–353.

Roy Schwartz, Oren Tsur, Ari Rappoport, and Moshe Koppel. 2013. Authorship attribution of micro-messages. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1880–1891.

Prasha Shrestha, Sebastian Sierra, Fabio Gonzalez, Manuel Montes, Paolo Rosso, and Thamar Solorio. 2017. Convolutional neural networks for authorship

attribution of short texts. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 669–674.

Thamar Solorio, Ragib Hasan, and Mainul Mizan. 2013. A case study of sockpuppet detection in wikipedia. In *Proceedings of the Workshop on Language Analysis in Social Media*, pages 59–68.

Thamar Solorio, Ragib Hasan, and Mainul Mizan. 2014. Sockpuppet detection in wikipedia: A corpus of real-world deceptive writing for linking identities. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland. European Language Resources Association (ELRA).

Ellen Spertus. 1997. Smokey: Automatic recognition of hostile messages. In *AAAI/IAAI*, pages 1058–1065.

Efstathios Stamatatos. 2009. A survey of modern authorship attribution methods. *J. Am. Soc. Inf. Sci. Technol.*, 60(3):538–556.

Efstathios Stamatatos, Francisco Rangel, Michael Tschuggnall, Benno Stein, Mike Kestemont, Paolo Rosso, and Martin Potthast. 2018. Overview of pan 2018. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 267–285. Springer.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077. International World Wide Web Conferences Steering Committee.

Robin Thompson. 2011. Radicalization and the use of social media. *Journal of strategic security*, 4(4):167–190.

Michail Tsikerdekis and Sherali Zeadally. 2014. Multiple account identity deception detection in social media using nonverbal behavior. *IEEE Transactions on Information Forensics and Security*, 9(8):1311–1321.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Luis Gerardo Mojica de la Vega and Vincent Ng. 2018. Modeling trolling in social media conversations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*.

Svitlana Volkova, Yoram Bachrach, Michael Armstrong, and Vijay Sharma. 2015. Inferring latent user properties from texts published in social media. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Soroush Vosoughi, Prashanth Vijayaraghavan, and Deb Roy. 2016. Tweet2vec: Learning tweet embeddings using character-level cnn-lstm encoder-decoder. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, pages 1041–1044, New York, NY, USA. ACM.

Isaac Waller and Ashton Anderson. 2019. Generalists and specialists: Using community embeddings to quantify activity diversity in online platforms. In *The World Wide Web Conference*, WWW '19, pages 1954–1964, New York, NY, USA. ACM.

Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu. 2014. Learning fine-grained image similarity with deep ranking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1386–1393.

Yi Yang, Ming-Wei Chang, and Jacob Eisenstein. 2016. Toward socially-infused information extraction: Embedding authors, mentions, and entities. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1452–1461.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 75–86.

# (Male, Bachelor) and (Female, Ph.D) have different connotations: Parallelly Annotated Stylistic Language Dataset with Multiple Personas

**Dongyeop Kang** **Varun Gangal** **Eduard Hovy**
Carnegie Mellon University, Pittsburgh, PA, USA
{dongyeok,vgangal,hovy}@cs.cmu.edu

## Abstract

Stylistic variation in text needs to be studied with different aspects including the writer's personal traits, interpersonal relations, rhetoric, and more. Despite recent attempts on computational modeling of the variation, the lack of parallel corpora of style language makes it difficult to systematically control the stylistic change as well as evaluate such models. We release PASTEL, the parallel and annotated stylistic language dataset, that contains $\approx$ 41K parallel sentences (8.3K parallel stories) annotated across different personas. Each persona has different styles in conjunction: gender, age, country, political view, education, ethnic, and time-of-writing. The dataset is collected from human annotators with solid control of input denotation: not only preserving original meaning between text, but promoting stylistic diversity to annotators. We test the dataset on two interesting applications of style language, where PASTEL helps design appropriate experiment and evaluation. First, in predicting a target style (e.g., male or female in gender) given a text, multiple styles of PASTEL make other external style variables controlled (or fixed), which is a more accurate experimental design. Second, a simple supervised model with our parallel text outperforms the unsupervised models using non-parallel text in style transfer. Our dataset is publicly available[1].

## 1 Introduction

Hovy (1987) claims that appropriately varying the style of text often conveys more information than is contained in the literal meaning of the words. He defines the roles of styles in text variation by pragmatics aspects (e.g., relationship between them) and rhetorical goals (e.g., formality), and provides example texts of how they are tightly coupled in practice. Similarly, Biber (1991) categorizes components of conversational situation by participants' characteristics such as their roles, personal characteristics, and group characteristics (e.g., social class). Despite the broad definition of style, this work mainly focuses on one specific aspect of style, *pragmatics aspects in group characteristics of speakers*, which is also called *persona*. Particularly, we look at multiple types of group characteristics in conjunction, such as gender, age, education level, and more.

Stylistic variation in text primarily manifest themselves at the different levels of textual features: lexical features (e.g., word choice), syntactic features (e.g., preference for the passive voice) and even pragmatics, while preserving the original meaning of given text (DiMarco and Hirst, 1990). Connecting such textual features to someone's persona is an important study to understand stylistic variation of language. For example, do highly educated people write longer sentences (Bloomfield, 1927)? Are Hispanic and East Asian people more likely to drop pronouns (White, 1985)? Are elder people likely to use lesser anaphora (Ulatowska et al., 1986)?

To computationally model a meaning-preserved variance of text across styles, many recent works have developed systems that transfer styles (Reddy and Knight, 2016; Hu et al., 2017; Prabhumoye et al., 2018) or profiles authorships from text (Verhoeven and Daelemans, 2014; Koppel et al., 2009; Stamatatos et al., 2018) without parallel corpus of stylistic text. However, the absence of such a parallel dataset makes it difficult both to systematically learn the textual variation of multiple styles as well as properly evaluate the models.

In this paper, we propose a *large scale, human-annotated, parallel stylistic dataset* called PASTEL, with focus on *multiple types of personas in conjunction*. Ideally, annotations for a parallel

[1] https://github.com/dykang/PASTEL



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1696–1706, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

style dataset should preserve the original meaning (i.e., *denotation*) between reference text and stylistically transformed text, while promoting diversity for annotators to allow their own styles of persona (i.e., *connotation*). However, if annotators are asked to write their own text given a reference sentence, they may simply produce arbitrarily paraphrased output which does not exhibit a stylistic diversity. To find such a proper input setting for data collection, we conduct a denotation experiment in §3. PASTEL is then collected by crowd workers based on the most effective input setting that balances both meaning preservation and diversity metrics (§4).

PASTEL includes stylistic variation of text at two levels of parallelism: ≈8.3K annotated, parallel stories and ≈41K annotated, parallel sentences, where each story has five sentences and has 2.63 annotators on average. Each sentence or story has the seven types of persona styles in conjunction: gender, age, ethnics, countries to live, education level, political view, and time of the day.

In §5, we introduce two interesting applications of style language using PASTEL: controlled style classification and supervised style transfer. The former application predicts a category (e.g., male or female) of target style (i.e., gender) given a text. Multiplicity of persona styles in PASTEL makes other style variables controlled (or fixed) except the target, which is a more accurate experimental design. In the latter, contrast to the unsupervised style transfer using non-parallel corpus, simple supervised models with our parallel text in PASTEL achieve better performance, being evaluated with the parallel, annotated text.

We hope PASTEL sheds light on the study of stylistic language variation in developing a solid model as well as evaluating the system properly.

## 2 Related Work

Transferring styles between text has been studied with and without parallel corpus:

*Style transfer without parallel corpus:* Prior works transfer style between text on single type of style aspect such as sentiment (Fu et al., 2018; Shen et al., 2017; Hu et al., 2017), gender (Reddy and Knight, 2016), political orientation (Prabhumoye et al., 2018), and two conflicting corpora (e.g, paper and news (Han et al., 2017), or real and synthetic reviews (Lipton et al., 2015)). They use different types of generative models in the same way as style transfer in images, where meaning preservation is not controlled systematically. Prabhumoye et al. (2018) proposes back-translation to get a style-agnostic sentence representation. However, they lack parallel ground truth for evaluation and present limited evaluation for meaning preservation.

*Style transfer with parallel corpus:* Few recent works use parallel text for style transfer between modern and Shakespearean text (Jhamtani et al., 2017), sarcastic and literal tweets (Peled and Reichart, 2017), and formal and informal text (Heylighen and Dewaele, 1999; Rao and Tetreault, 2018). Compared to these, we aim to understand and demonstrate style variation owing to multiple demographic attributes.

Besides the style transfer, other applications using stylistic features have been studied such as poetry generation (Ghazvininejad et al., 2017), stylometry with demographic information (Verhoeven and Daelemans, 2014), modeling style bias (Vogel and Jurafsky, 2012) and modeling biographic attributes (Garera and Yarowsky, 2009). A series of works by (Koppel et al., 2011, 2009; Argamon et al., 2009; Koppel and Winter, 2014) and their shared tasks (Stamatatos et al., 2018) show huge progress on author profiling and attribute classification tasks. However, none of the prior works have collected a stylistic language dataset to have multiple styles in conjunction, parallely annotated by a human. The multiple styles in conjunction in PASTEL enable an appropriate experiment setting for controlled style classification task in Section 5.1.

## 3 Denotation Experiment

We first provide a preliminary study to find the best input setting (or denotation) for data collection to balance between two trade-off metrics: meaning preservation and style diversity.

### 3.1 Preliminary Study

Table 1 shows output texts produced by annotators given different input denotation settings. The basic task is to provide an input denotation (e.g., a sentence only, a sequence of images) and then ask them to reproduce text maintaining the meaning of the input but with their own persona.

For instance, if we provide a *single reference sentence*, annotators mostly repeat the input text with a little changes of the lexical terms. This

| **Denotation**: | **Produced sentences**: |
|---|---|
| *single ref. sentence* | the old door with wood was the only direction to the courtyard |
| *story(imgs)* | The old wooden door in the stonewall looks like a portal to a fairy tale. |
| *story(imgs.+keyw words)* | Equally so, he is intrigued by the heavy wooden door in the courtyard. |
| **Reference sentence**: the old wooden door was only one way into the courtyard. | |

Table 1: Textual variation across different denotation settings. Each sentence is produced by a same annotator. Note that providing reference sentence increases fidelity to the reference while decreases diversity.

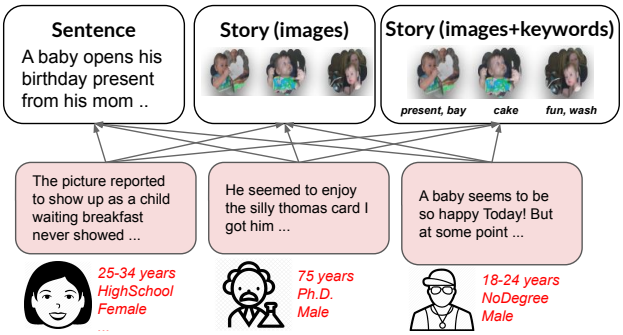


Figure 1: Denotation experiment finds the best input setting for data collection, that preserves meaning but diversifies styles among annotators with different personas.

setup mostly preserves the meaning by simply paraphrasing the sentence, but annotators' personal style does not reflect the variation. With a *single image*, on the other hand, the outputs produced by annotators tend to be diverse. However, the image can be explained with a variety of contents, so the output meaning can drift away from the reference sentence.

If a series of consistent images (i.e., a story) is given, we expect a stylistic diversity can be more narrowed down, by grounding it to a specific event or a story. In addition to that, some keywords added to each image of a story help deliver more concrete meaning of content as well as the style diversity.

### 3.2 Experimental Setup

In order to find the best input setting that preserves meaning as well as promotes a stylistic diversity, we conduct a denotation experiment as described in Figure 1. The experiment is a subset of our original dataset, which have only 100 samples of annotations.

A basic idea behind this setup is to provide (1) a perceptually common denotation via sentences or images so people share the same context (i.e., denotation) given, (2) a series of them as a "story" to limit them into a specific event context, and (3) two modalities (i.e., text and image) for better disambiguation of the context by grounding them to each other.

We test five different input settings[2]: *Single reference sentence*, ***Story*** *(images)*, ***Story*** *(images)* + *global keywords*, ***Story*** *(images + local keywords)*, and ***Story*** *(images + local keywords + ref. sentence)*.

For the keyword selection, we use RAKE algorithm (Rose et al., 2010) to extract keywords and rank them for each sentence by the output score. Top five uni/bigram keywords are chosen at each story, which are called *global keywords*. On the other hand, another top three uni/bigram keywords are chosen at each image/sentence in a story, which are called *local keywords*. Local keywords for each image/sentence help annotators not deviate too much. For example, *local keywords* look like *(restaurant, hearing, friends)* $\rightarrow$ *(pictures, menu, difficult)* $\rightarrow$ *(salad, corn, chose)* for three sentences/images, while *global keywords* look like *(wait, salad, restaurant)* for a story of the three sentences/images.

We use Visual Story Telling (ViST) (Huang et al., 2016) dataset as our input source. The dataset contains stories, and each story has five pairs of images and sentences. We filter out stories that are not temporally ordered using the timestamps of images. The final number of stories after filtering the non-temporally-ordered stories is 28,130. For the denotation experiment, we only use randomly chosen 100 stories. The detailed pre-processing steps are described in Appendix.

### 3.3 Measuring Meaning Preservation & Style Diversity across Different Denotations

For each denotation setting, we conduct a quantitative experiment to measure the two metrics: meaning preservation and style diversity. The two metrics pose a trade-off to each other. The best input setting then is one that can capture both in appropriate amounts. For example, we want meaning of the input preserved, while lexical or syn-

[2]Other settings like *Single reference image* are tested as well, but they didn't preserve the meaning well.

| | *denotation settings* | **Style Diversity** | **Meaning Preservation** | |
|---|---|---|---|---|
| | | E(GM) | METEOR | VectorExtrema |
| **sentence-level** | *single ref. sentence* | <u>2.98</u> | **0.37** | **0.70** |
| | ***story**(images)* | 2.86 | 0.07 | 0.38 |
| | ***story**(images) + global keywords* | 2.85 | 0.07 | 0.39 |
| | ***story**(images + local keywords)* | **3.07** | 0.17 | <u>0.53</u> |
| | ***story**(images + local keywords + ref. sentence)* | 2.91 | <u>0.21</u> | 0.43 |
| **story-level** | ***story**(images)* | 4.43 | 0.1 | 0.4 |
| | ***story**(images) + global keywords* | 4.43 | 0.1 | 0.42 |
| | ***story**(images + local keywords)* | **4.58** | <u>0.19</u> | **0.55** |
| | ***story**(images + local keywords + ref. sentence)* | <u>4.48</u> | **0.22** | <u>0.44</u> |

Table 2: Denotation experiment to find the best input setting (i.e., meaning preserved but stylistically diverse). **story-level** measures the metrics for five sentences as a story, and **sentence-level** per individual sentence. Note that *single reference sentence* setting only has sentence level. For every metrics in both meaning preservation and style diversity, the higher the better. The **bold** number is the highest, and the <u>underlined</u> is the second highest.




Figure 2: Final denotation setting for data collection: an event that consists of a series of five images with a handful number of keywords. We ask annotators to produce text about the event for each image.

tactic features (e.g., POS tags) can vary depending on annotator's persona. We use the following automatic measures for the two metrics:

**Style Diversity** measures how much produced sentences (or stories) differ amongst themselves. Higher the diversity, better the stylistic variation in language it contains. We use an entropy measure to capture the variance of n-gram features between annotated sentences: Entropy (Gaussian-Mixture) that combines the N-Gram entropies (Shannon, 1951) using Gaussian mixture model (N=3).

**Meaning Preservation** measures semantic similarity of the produced sentence (or story) with the reference sentence (or story). Higher the similarity, better the meaning preserved. We use a hard-measure, METEOR (Banerjee and Lavie, 2005), that calculates F-score of word overlaps between the output and reference sentences[3]. Since the hard measures do not take into account all semantic similarities [4], we also use a soft measure, VectorExtrema (VecExt) (Liu et al., 2016). It computes cosine similarity of averaged word embeddings (i.e., GloVe (Pennington et al., 2014)) between the output and reference sentences.

Table 2 shows results of the two metrics across different input settings we define. For the sentence level, as expected, *single reference sentence* has the highest meaning preservation across all the metrics because it is basically paraphrasing the reference sentence. In general, ***Story*** (*images + local keywords*) shows a great performance with the highest diversity regardless of the levels, as well as the highest preservation at the soft measure on the story-level. Thus, we use ***Story***(*images+local keywords*) as the input setting for our final data collection, which has the most balanced performance on both metrics. Figure 2 shows an example of our input setting for crowd workers.

## 4 PASTEL: A Parallelly Annotated Dataset for Stylistic Language Dataset

We describe how we collect the dataset with human annotations and provide some analysis on it.

[3]Other measures (e.g., BLEU (Papineni et al., 2002), ROUGE (Lin and Hovy, 2003)) show relatively similar performance.

[4]METEOR does consider synonymy and paraphrasing but is limited by its predefined model/dictionaries/resources for the respective language, such as Wordnet

### 4.1 Annotation Schemes

Our crowd workers are recruited from the Amazon Mechanical Turk (AMT) platform. Our annotation scheme consists of two steps: (1) ask annotator's demographic information (e.g., gender, age) and (2) given an input denotation like Figure 2, ask them to produce text about the denotation with their own style of persona (i.e., connotation).

In the first step, we use seven different types of persona styles; *gender*, *age*, *ethnic*, *country*, *education level*, and *political orientation*, and one additional context style *time-of-day* (tod). For each type of persona, we provide several categories for annotators to choose. For example, *political orientation* has three categories: Centrist, Left Wing, and Right Wing. Categories in other styles are described in the next sub-section.

In the second step, we ask annotators to produce text that describes the given input of denotation. We again use the pre-processed ViST (Huang et al., 2016) data in §3 for our input denotations. To reflect annotators' persona, we explicitly ask annotators to reflect their own persona in the stylistic writing, instead of pretending others' persona. We attach detailed annotation schemes at Figure 5 in Appendix.

To amortize both costs and annotators' effort at answering questions, each HIT requires the participants to annotate three stories after answering demographic questions. One annotator was paid $0.11 per HIT. For English proficiency, the annotators were restricted to be from USA or UK. A total 501 unique annotators participated in the study. The average number of HIT per annotator was 9.97.

| | Number of Sentences | Number of Stories |
|---|---|---|
| Train | 33,240 | 6,648 |
| Valid | 4,155 | 831 |
| Test | 4,155 | 831 |
| total | 41,550 | 8,310 |

Table 3: Data statistics of the PASTEL.

Once we complete our annotations, we filter out noisy responses such as stories with missing images and overtly short sentences (i.e., minimum sentence length is 5). The dataset is then randomly split into train, valid, and test set by 0.8, 0.1, and 0.1 ratios, respectively. Table 3 shows the final number of stories and sentences in our dataset.

### 4.2 Analysis and Examples

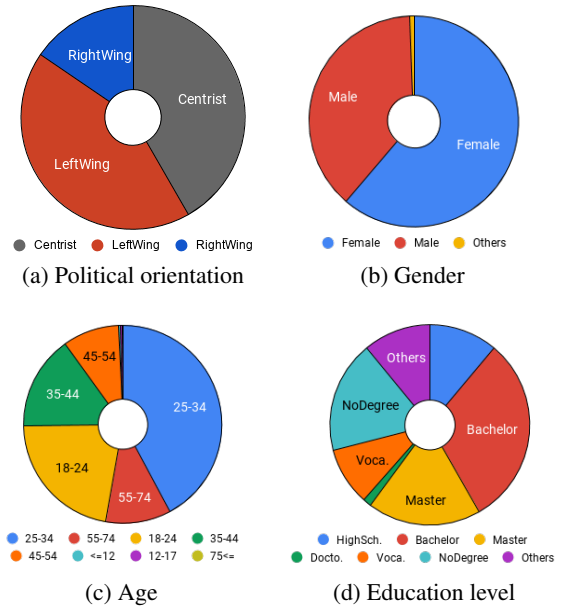


(a) Political orientation (b) Gender

(c) Age (d) Education level

Figure 3: Distribution of annotators for each personal style in PASTEL. Best viewed in color.

Figure 3 shows demographic distributions of the annotators. Education-level of annotators is well-balanced, while gender and political view are somewhat biased (e.g., 68% of annotators are Female, only 18.6% represent themselves as right-wing). Table 8 in Appendix includes the categories in other styles and their distributions.

Table 4 shows few examples randomly chosen from our dataset: two at sentence level (top, middle) and one at story level (bottom). Due to paucity of space, we only show a few types of persona styles. For example, we observe that Education level (e.g., NoDegree vs. Graduate) actually reflects a certain degree of formality in their writing at both sentence and story levels. In §5.1, we conduct an in-depth analysis of textual variation with respect to the persona styles in PASTEL.

## 5 Applications with PASTEL

PASTEL can be used in many style related applications including style classification, stylometry (Verhoeven and Daelemans, 2014), style transfer (Fu et al., 2018), visually-grounded style transfer, and more. Particularly, we chose two applications, where PASTEL helps design appropriate experiment and evaluation: controlled style classification (§5.1) and supervised style transfer (§5.2).

| **Reference Sentence:** went to an art museum with a group of friends. | |
|---|---|
| *edu*:HighSchoolOrNoDiploma | My friends and I went to a art museum yesterday . |
| *edu*:Bachelor | I went to the museum with a bunch of friends. |

| **Reference Sentence:** the living room of our new home is nice and bright with natural light. | |
|---|---|
| *edu*:NoDegree, *gender*:Male | The natural lightning made the apartment look quite nice for the upcoming tour . |
| *edu*:Graduate, *gender*:Female | The house tour began in the living room which had a sufficient amount of natural lighting. |

**Reference Story:** Went to an art museum with a group of friends . We were looking for some artwork to purchase, as sometimes artist allow the sales of their items . There were pictures of all sorts , but in front of them were sculptures or arrangements of some sort . Some were far out there or just far fetched . then there were others that were more down to earth and stylish. this set was by far my favorite.very beautiful to me .

| | |
|---|---|
| *edu*:HighSchool, *ethnic*:Caucasian, *gender*:Female | My friends and I went to a art museum yesterday . There were lots of puchases and sales of items going on all day . I loved the way the glass sort of brightened the art so much that I got all sorts of excited . After a few we fetched some grub . My favorite set was all the art that was made out of stylish trash . |
| *edu*:Bachelor, *ethnic*:Caucasian, *gender*:Female | I went to the museum with a bunch of friends . There was some cool art for sale . We spent a lot of time looking at the sculptures . This was one of my favorite pieces that I saw . We looked at some very stylish pieces of artwork . |

Table 4: Two sentence-level (top, middle) and one story-level (bottom) annotations in PASTEL. Each text produced by an annotator has their own persona values (underline) for different types of styles (italic). Note that the reference sentence (or story) is given for comparison with the annotated text. Note that misspellings of the text are made by annotators.

### 5.1 Controlled Style Classification

A common mistake in style classification datasets is not controlling external style variables when predicting the category of the target style. For example, when predicting a gender type given a text $P(\textit{gender}=\underline{Male}|\texttt{text})$, the training data is only labeled by the target style *gender*. However, the `text` is actually produced by a person with not only *gender*=$\underline{Male}$ but also other persona styles such as *age*=55-74 or *education*=HighSchool. Without controlling the other external styles, the classifier is easily biased against the training data.

We define a task called *controlled style classification* where all other style variables are fixed[5], except one to classify. Here we evaluate (1) which style variables are relatively difficult or easy to predict from the text given, and (2) what types of textual features are salient for each type of style classification.

**Features.** Stylistic language has a variety of features at different levels such as lexical choices, syntactic structure and more. Thus, we use following features:

- **lexical** features: ngram's frequency (n=3), number of named entities, number of stop-words
- **syntax** features: sentence length, number of each Part-of-Speech (POS) tag, number of out-of-vocabulary, number of named entities
- **deep** features: pre-trained sentence encoder using BERT (Devlin et al., 2019)
- **semantic** feature: sentiment score

where named entities, POS tags, and sentiment scores are obtained using the off-the-shelf tools such as Spacy[6] library. We use $70K$ n-gram lexical features, 300 dimensional embeddings, and 14 hand-written features.

[5]The distribution of number of training instances per variable is given in Appendix

[6]https://spacy.io/

**Models.** We train a binary classifier for each personal style with different models: logistic regression, SVM with linear/RBF kernels, Random Forest, Nearest Neighbors, Multi-layer Perceptron, AdaBoost, and Naive Bayes. For each style, we choose the best classifiers on the validation. Their F-scores are reported in Figure 4. We use sklearn's implementation of all models (Pedregosa et al., 2011).[7] We consider various regularization parameters for SVM and logistic regression (e.g., c=[0.01, 0.1, 0.25, 0.5, 0.75, 1.0].

We use neural network based baseline models: deep averaging networks (DAN, Iyyer et al., 2015) of GloVe word embeddings (Pennington et al., 2014)[8]. We also compare with the non-controlled model (Combined) which uses a combined set of samples across all other variables except for one to classify using the same features we used.

**Setup.** We tune hyperparameters using 5-fold cross validation. If a style has more than two categories, we choose the most conflicting two: *gender*:{Male, Female}, *age*: {18-24, 35-44}, *education*: {Bachelor, No Degree}, and *politics*: {LeftWing, RightWing}. To classify one style, all possible combinations of other styles ($2 * 2 * 2$=8) are separately trained by different models. We use the macro-averaged F-scores among the separately trained models on the same test set for every models.

**Results.** Figure 4 shows F-scores (a) among different styles and (b) between sentences and stories. In most cases, multilayer perceptron (MLP) outperforms the majority classifier and other models by large margins. Compared to the neural baselines and the combined classifier, our models show better performance. In comparison between controlled and combined settings, controlled setting achieves higher improvements, indicating that fixing external variables helps control irrelevant features that come from other variables. Among different styles, gender is easier to predict from the text than ages or education levels. Interestingly, a longer context (i.e., story) is helpful in predicting age or education, whereas not for political view and gender.

In our ablation test among the feature types, the combination of different features (e.g., lexical, syntax, deep, semantic) is very complementary and effective. Lexical and deep features are two most significant features across all style classifiers, while syntactic features are not.

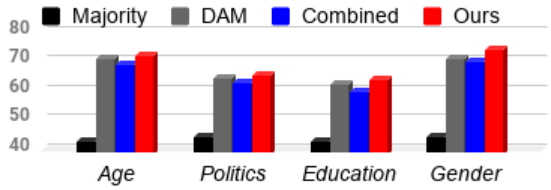


(a) different style types

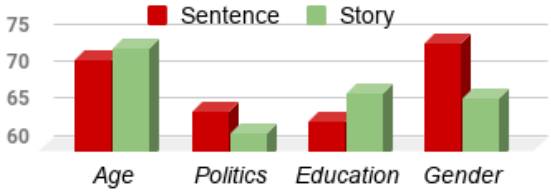


(b) sentences vs stories

Figure 4: Controlled style classification: F-scores on (a) different types of styles on sentences and on (b) our best models between sentences and stories. Best viewed in color.

| *Gender*:Male | *Gender*:Female |
|---|---|
| PROPN, ADJ, #_ENTITY, went, party, SENT_LEN | happy, day, end, group, just, snow, NOUN |
| *Politics*:LeftWing | *Politics*:RightWing |
| female, time, NOUN, ADP, VERB, porch, day, loved | SENT_LENGTH, PROPN, #_ENTITY, n't, ADJ, NUM |
| *Education*:Bachelor | *Education*:NoDegree |
| food, went, #_STOPWORDS, race, ADP | !, just, came, love, lots, male, fun, n't, friends, happy |
| *Age*:18-24 | *Age*:35-44 |
| ADP, come, PROPN, day, ride, playing, sunset | ADV, did, town, went, NOUN, #_STOPWORDS |

Table 5: Most salient lexical (lower cased) and syntactic (upper cased) features on story-level classification. Each feature is chosen by the highest coefficients in the logistic regression classifier.

Table 5 shows the most salient features for classification of each style. Since we can't interpret deep features, we only show lexical and syntactic features. The salience of features are ranked by coefficients of a logistic regression classifier. Interestingly, female annotators likely write more nouns and lexicons like 'happy', while male annotators likely use pronouns, adjectives, and named

[7] http://scikit-learn.org/stable/

[8] Other architectures such as convolutional neural networks (CNN, Zhang et al., 2015) and recurrent neural networks (LSTM, Hochreiter and Schmidhuber, 1997) show comparable performance as DAN.

entities. Annotators on left wing prefer to use 'female', nouns and adposition, while annotators on right wing prefer shorter sentences and negative verbs like 'n't'. Not many syntactic features are observed from annotators without degrees compared to with bachelor degree.

### 5.2 Supervised Style Transfer

The style transfer is defined as $(S, \alpha) \rightarrow \hat{S}$: We attempt to alter a given source sentence $S$ to a given target style $\alpha$. The model generates a candidate target sentence $\hat{S}$ which preserves the meaning of $S$ but is more faithful to the target style $\alpha$ so being similar to the target annotated sentence $\bar{S}_\alpha$. We evaluate the model by comparing the predicted sentence $\hat{S}$ and target annotated sentence $\bar{S}_\alpha$. The sources are from the original reference sentences, while the targets are from our annotations.

**Models.** We compare five different models:

- AsItIs: copies over the source sentence to the target, without any alterations.
- WordDistRetrieve: retrieves a training source-target pair that has the same target style as the test pair and is closest to the test source in terms of word edit distance (Navarro, 2001). It then returns the target of that pair.
- EmbDistRetrieve: Similar to WordDistRetrieve, except that a continuous bag-of-words (CBOW) is used to retrieve closest source sentence instead of edit distance.
- Unsupervised: use unsupervised style transfer models using Variational Autoencoder (Shen et al., 2017) and using additional objectives such as cross-domain and adversarial losses (Lample et al., 2017)[9]. Since unsupervised models can't train multiple styles at the same time, we train separate models for each style and macro-average their scores at the end. In order not to use the parallel text in PASTEL, we shuffle the training text of each style.
- Supervised: uses a simple attentional sequence-to-sequence (S2S) model (Bahdanau et al., 2014) extracting the parallel text from PASTEL. The model jointly trains different styles in conjunction by concatenating them to the source sentence at the beginning.

We avoid more complex architectural choices for Supervised models like adding a pointer component or an adversarial loss, since we seek to establish a minimum level of performance on this dataset.

[9]We can't directly compare with Hu et al. (2017); Prabhumoye et al. (2018) since their performance highly depends on the pre-trained classifier that often shows poor performance.

**Setup.** We experiment with both Softmax and Sigmoid non-linearities to normalize attention scores in the sequence-to-sequence attention. Adam (Kingma and Ba, 2014) is used as the optimizer. Word-level cross entropy of the target is used as the loss. The batch size is set to 32. We pick the model with lowest validation loss after 15 training epochs. All models are implemented in PyTorch (Paszke et al., 2017).

For an evaluation, in addition to the same hard and soft metrics used for measuring the meaning preservation in §3, we also use $\text{BLEU}_2$ (Papineni et al., 2002) for unigrams and bigrams, and ROUGE (Lin and Hovy, 2003) for hard metric and Embedding Averaging (EA) similarity (Liu et al., 2016) for soft metric.

| | Hard $(\hat{S},\bar{S}_\alpha)$ | | | Soft $(\hat{S},\bar{S}_\alpha)$ | |
|---|---|---|---|---|---|
| Models: $(S, \alpha) \rightarrow \hat{S}$ | $\mathbf{B_2}$ | **M** | **R** | **EA** | **VE** |
| AsItIs | **35.41** | **12.38** | 21.08 | 0.649 | 0.393 |
| WordDistRetrieve | 30.64 | 7.27 | 22.52 | 0.771 | 0.433 |
| EmbDistRetrieve | 33.00 | 8.29 | 24.11 | 0.792 | 0.461 |
| Unsupervised | | | | | |
| · Shen et al. (2017) | 23.78 | 7.23 | 21.22 | 0.795 | 0.353 |
| · Lample et al. (2017) | 24.52 | 6.27 | 19.79 | 0.702 | 0.369 |
| Supervised | | | | | |
| · S2S | 26.78 | 7.36 | 25.57 | 0.773 | 0.455 |
| · S2S+GloVe | 31.80 | 10.18 | 29.18 | 0.797 | 0.524 |
| · S2S+GloVe+PreTr. | 31.21 | 10.29 | **29.52** | **0.804** | **0.529** |

Table 6: Supervised style transfer. GloVe initializes with pre-trained word embeddings. PreTr. denotes pre-training on YAFC. Hard measures are $\mathbf{BLEU_2}$, **M**ETEOR, and **R**OUGE, and soft measures are **E**mbeding**A**veraging and **V**ector**E**xtrema.

**Results.** Table 6 shows our results on style tranfer. We observe that initializing both en/decoder's word embeddings with GloVe (Pennington et al., 2014) improves model performance on most metrics. Pretraining (PreTr.) on the formality style transfer data YAFC (Rao and Tetreault, 2018) further helps performance. All supervised S2S approaches outperform both retrieval-based baselines on all measures. This illustrates that the performance scores achieved are not simply a result of memorizing the training set. S2S methods surpass AsItIs on both soft measures and ROUGE. The significant gap that remains on BLEU remains

| |
|---|
| **Source** ($S$): I'd never seen so many beautiful flowers. |
| **Style** ($\alpha$): (Morning, HighSchool)<br>$S + \alpha \rightarrow \hat{S}$: the beautiful flowers were beautiful.<br>$\bar{S}_\alpha$: the flowers were in full bloom.<br>**Style** ($\alpha$): (Afternoon, NoDegree)<br>$S + \alpha \rightarrow \hat{S}$: The flowers were very beautiful.<br>$\bar{S}_\alpha$: Tulips are one of the magnificent varieties of flowers. |
| **Source** ($S$): she changed dresses for the reception and shared food with her new husband. |
| **Style** ($\alpha$): (Master, Centrist)<br>$S + \alpha \rightarrow \hat{S}$: The woman had a great time with her husband<br>$\bar{S}_\alpha$: Her husband shared a cake with her during reception |
| **Style** ($\alpha$): (Vocational, Right)<br>$S + \alpha \rightarrow \hat{S}$: The food is ready for the reception<br>$\bar{S}_\alpha$: The new husband shared the cake at the reception |

Table 7: Examples of style transferred text by our supervised model (S2S+GLOVE+PRETR.) on PASTEL. Given source text ($S$) and style ($\alpha$), the model predicts a target sentence $\hat{S}$ compared to annotated target sentence $\bar{S}_\alpha$.

a point of exploration for future work. The significant improvement against the unsupervised methods (Shen et al., 2017; Lample et al., 2017) indicates the usefulness of the parallel text in PASTEL.

Table 7 shows output text $\hat{S}$ produced by our model given a source text $S$ and a style $\alpha$. We observe that the output text changes according to the set of styles.

## 6 Conclusion and Future Directions

We present PASTEL, a parallelly annotated stylistic language dataset. Our dataset is collected by human annotation using a proper denotation setting that preserves the meaning as well as maximizes the diversity of styles. Multiplicity of persona styles in PASTEL makes other style variables controlled (or fixed) except the target style for classification, which is a more accurate experimental design. Our simple supervised models with our parallel text in PASTEL outperforms the unsupervised style transfer models using non-parallel corpus. We hope PASTEL can be a useful benchmark to both train and evaluate models for style transfer and other related problems in text generation field.

We summarize some directions for future style researches:

- In our ablation study, salient features for style classification are not only syntactic or lexical features but also content words (e.g., love, food). This is a counterexample to the hypothesis implicit in much of recent style research: *style* needs to be *separately modeled* from *content*. We also observe that some texts remain similar across different annotator personas or across outputs from our transfer models, indicating that some content is stylistically invariant. Studying these and other aspects of the content-style relationship in PASTEL could be an interesting direction.
- Does any external variable co-varying with the text qualify to be a style variable/facet? What are the categories of style variables/facets? Do architectures which transfer well across one style variable (e.g gender) generalize to other style variables (e.g age)? We opine that these questions are largely overlooked by current style transfer work. We hope that our consideration of some of these questions in our work, though admittedly rudimentary, will lead to them being addressed extensively in future style transfer work.

## Acknowledgements

This work would not have been possible without the ViST dataset and helpful suggestions with Ting-Hao Huang. We also thank Alan W Black, Dan Jurafsky, Wei Xu, Taehee Jung, and anonymous reviewers for their helpful comments.

## References


Shlomo Argamon, Moshe Koppel, James W Pennebaker, and Jonathan Schler. 2009. Automatically profiling the author of an anonymous text. *Commun. ACM*, 52(2):119–123.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Douglas Biber. 1991. *Variation across speech and writing*. Cambridge University Press.

Leonard Bloomfield. 1927. Literate and illiterate speech. *American speech*, 2(10):432–439.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Chrysanne DiMarco and Graeme Hirst. 1990. Accounting for style in machine translation. In *Third International Conference on Theoretical Issues in Machine Translation, Austin*.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2018. Style transfer in text: Exploration and evaluation. *CoRR*, abs/1711.06861.

Nikesh Garera and David Yarowsky. 2009. Modeling latent biographic attributes in conversational genres. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 710–718. Association for Computational Linguistics.

Marjan Ghazvininejad, Xing Shi, Jay Priyadarshi, and Kevin Knight. 2017. Hafez: an Interactive Poetry Generation System. *Proceedings of ACL 2017, System Demonstrations*, pages 43–48.

Mengqiao Han, Ou Wu, and Zhendong Niu. 2017. Unsupervised Automatic Text Style Transfer using LSTM. In *National CCF Conference on Natural Language Processing and Chinese Computing*, pages 281–292. Springer.

Francis Heylighen and Jean-Marc Dewaele. 1999. Formality of language: definition, measurement and behavioral determinants. *Interner Bericht, Center Leo Apostel, Vrije Universiteit Brüssel*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Eduard Hovy. 1987. Generating natural language under pragmatic constraints. *Journal of Pragmatics*, 11(6):689–719.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *International Conference on Machine Learning*, pages 1587–1596.

Ting-Hao (Kenneth) Huang, Francis Ferraro, Nasrin Mostafazadeh, Ishan Misra, Aishwarya Agrawal, Jacob Devlin, Ross Girshick, Xiaodong He, Pushmeet Kohli, Dhruv Batra, C. Lawrence Zitnick, Devi Parikh, Lucy Vanderwende, Michel Galley, and Margaret Mitchell. 2016. Visual storytelling. In *NAACL*.

Shankar Iyer, Nikhil Dandekar, and Kornél Csernai. 2017. First quora dataset release: Question pairs.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proc. of ACL-IJCNLP*, volume 1, pages 1681–1691.

Harsh Jhamtani, Varun Gangal, Eduard H. Hovy, and Eric Nyberg. 2017. Shakespearizing modern language using copy-enriched sequence-to-sequence models. *CoRR*, abs/1707.01161.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Moshe Koppel, Jonathan Schler, and Shlomo Argamon. 2009. Computational methods in authorship attribution. *Journal of the American Society for information Science and Technology*, 60(1):9–26.

Moshe Koppel, Jonathan Schler, and Shlomo Argamon. 2011. Authorship attribution in the wild. *Language Resources and Evaluation*, 45:83–94.

Moshe Koppel and Yaron Winter. 2014. Determining if two documents are written by the same author. *Journal of the Association for Information Science and Technology*, 65(1):178–187.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2017. Unsupervised machine translation using monolingual corpora only. *arXiv preprint arXiv:1711.00043*.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 71–78. Association for Computational Linguistics.

Zachary C Lipton, Sharad Vikram, and Julian McAuley. 2015. Generative concatenative nets jointly learn to Write and Classify reviews. *arXiv preprint arXiv:1511.03683*.

Chia-Wei Liu, Ryan Lowe, Iulian V Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *arXiv preprint arXiv:1603.08023*.

Gonzalo Navarro. 2001. A guided tour to approximate string matching. *ACM computing surveys (CSUR)*, 33(1):31–88.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Soumith Chintala, and Gregory Chanan. 2017. Pytorch.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

Lotem Peled and Roi Reichart. 2017. Sarcasm sign: Interpreting sarcasm with sentiment based monolingual machine translation. In *ACL*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W. Black. 2018. Style transfer through back-translation. In *ACL*.

Sudha Rao and Joel R. Tetreault. 2018. Dear sir or madam, may i introduce the gyafc dataset: Corpus, benchmarks and metrics for formality style transfer. In *NAACL-HLT*.

Sravana Reddy and Kevin Knight. 2016. Obfuscating gender in social media writing. In *Proceedings of the First Workshop on NLP and Computational Social Science*, pages 17–26.

Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. Automatic keyword extraction from individual documents. *Text Mining: Applications and Theory*, pages 1–20.

Claude E Shannon. 1951. Prediction and entropy of printed English. *Bell Labs Technical Journal*, 30(1):50–64.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in Neural Information Processing Systems*, pages 6830–6841.

Efstathios Stamatatos, Francisco Rangel, Michael Tschuggnall, Benno Stein, Mike Kestemont, Paolo Rosso, and Martin Potthast. 2018. Overview of pan 2018. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 267–285. Springer.

Hanna K Ulatowska, Mari M Hayashi, Michael P Cannito, and Susan G Fleming. 1986. Disruption of reference in aging. *Brain and language*, 28(1):24–41.

Ben Verhoeven and Walter Daelemans. 2014. CLiPS Stylometry Investigation (CSI) corpus: A Dutch corpus for the detection of age, gender, personality, sentiment and deception in text. In *LREC 2014-NINTH INTERNATIONAL CONFERENCE ON LANGUAGE RESOURCES AND EVALUATION*, pages 3081–3085.

Adam Vogel and Dan Jurafsky. 2012. He said, she said: Gender in the ACL anthology. In *Proceedings of the ACL-2012 Special Workshop on Rediscovering 50 Years of Discoveries*, pages 33–41. Association for Computational Linguistics.

Lydia White. 1985. The pro-drop parameter in adult second language acquisition. *Language learning*, 35(1):47–61.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proc. of NIPS*.

# Movie Plot Analysis via Turning Point Identification

**Pinelopi Papalampidi** **Frank Keller** **Mirella Lapata**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
p.papalampidi@sms.ed.ac.uk, {keller,mlap}@inf.ed.ac.uk

## Abstract

According to screenwriting theory, turning points (e.g., change of plans, major setback, climax) are crucial narrative moments within a screenplay: they define the plot structure, determine its progression and thematic units (e.g., setup, complications, aftermath). We propose the task of turning point identification in movies as a means of analyzing their narrative structure. We argue that turning points and the segmentation they provide can facilitate processing long, complex narratives, such as screenplays, for summarization and question answering. We introduce a dataset consisting of screenplays and plot synopses annotated with turning points and present an end-to-end neural network model that identifies turning points in plot synopses and projects them onto scenes in screenplays. Our model outperforms strong baselines based on state-of-the-art sentence representations and the expected position of turning points.

## 1 Introduction

Computational literary analysis works at the intersection of natural language processing and literary studies, aiming to evaluate various theories of storytelling (e.g., by examining a collection of works within a single genre, by an author, or topic) and to develop tools which aid in searching, visualizing, or summarizing literary content.

Within natural language processing, computational literary analysis has mostly targeted works of fiction such as novels, plays, and screenplays. Examples include analyzing characters, their relationships, and emotional trajectories (Chaturvedi et al., 2017; Iyyer et al., 2016; Elsner, 2012), identifying enemies and allies (Nalisnick and Baird, 2013), villains or heroes (Bamman et al., 2014, 2013), measuring the memorability of quotes (Danescu-Niculescu-Mizil et al., 2012), characterizing gender representation in dialogue (Agarwal et al., 2015; Ramakrishna et al., 2015; Sap et al., 2017), identifying perpetrators in crime series (Frermann et al., 2018), summarizing screenplays (Gorinski and Lapata, 2018), and answering questions about long and complex narratives (Kočiskỳ et al., 2018).

| Turning Point | Description |
|---|---|
| 1. Opportunity | Introductory event that occurs after the presentation of the setting and the background of the main characters. |
| 2. Change of Plans | Event where the main goal of the story is defined. From this point on, the action begins to increase. |
| 3. Point of No Return | Event that pushes the main character(s) to fully commit to their goal. |
| 4. Major Setback | Event where everything falls apart (temporarily or permanently). |
| 5. Climax | Final event of the main story, moment of resolution and the "biggest spoiler". |

Table 1: Turning points and their definitions.

In this paper we are interested in the automatic analysis of narrative structure in screenplays. Narrative structure, also referred to as a storyline or plotline, describes the framework of how one tells a story and has its origins to Aristotle who defined the basic triangle-shaped plot structure representing the beginning (protasis), middle (epitasis), and end (catastrophe) of a story (Pavis, 1998). The German novelist and playwright Gustav Freytag modified Aristotle's structure by transforming the triangle into a pyramid (Freytag, 1896). In his scheme, there are five acts (*introduction, rising movement, climax, return*, and *catastrophe*). Several variations of Freytag's pyramid are used today in film analysis and screenwriting (Cutting, 2016).

In this work, we adopt a variant commonly employed by screenwriters as a practical guide for producing successful screenplays (Hague, 2017). According to this scheme, there are six stages (acts) in a film, namely *the setup*, *the new situation*, *progress*, *complications and higher stakes*, *the final push*, and *the aftermath*, separated by five *turning points* (TPs). TPs are narrative moments from which the plot goes in a different direction



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1707–1717, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

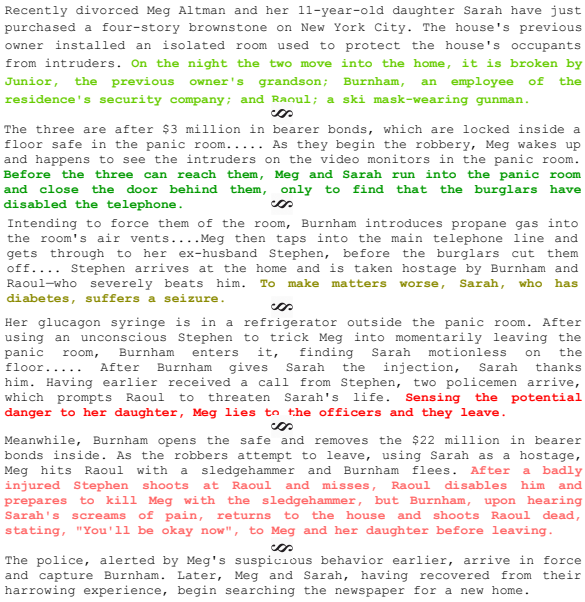


Figure 1: Example of turning point annotations (TP1, TP2, TP3, TP4, TP5, respectively) for the synopsis of the movie "Panic Room".

(Thompson, 1999), and by definition they occur at the junctions of acts. Aside from changing narrative direction, TPs define the movie's structure, tighten the pace, and prevent the narrative from drifting. The five TPs and their definitions are given in Table 1.

We propose the task of turning point identification in movies as a means of analyzing their narrative structure. TP identification provides a sequence of key events in the story and segments the screenplay into thematic units. Common approaches to summarization and QA of long or multiple documents (Chen et al., 2017; Yang et al., 2018; Kratzwald and Feuerriegel, 2018; Elgohary et al., 2018) include a retrieval system as the first step, which selects a subset of relevant passages for further processing. However, Kočiskỳ et al. (2018) demonstrate that these approaches do not perform equally well for extended narratives, since individual passages are very similar and the same entities are referred to throughout the story. We argue that this challenge can be addressed by TP identification, which finds the most important events and segments the narrative into thematic units. Downstream processing for summarization or question answering can then focus on those segments that are relevant to the task.

Problematically for modeling purposes, TPs are latent in screenplays, there are no scriptwriting conventions (like character cues or scene headings) to denote where TPs occur, and their exact manifestation varies across movies (depending on genre and length), although there are some rules of thumb indicating where to expect a TP (e.g., the Opportunity occurs after the first 10% of a screenplay, Change of Plans is approximately 25% in). To enable automatic TP identification, we develop a new dataset which consists of screenplays, plot synopses, and turning point annotations. To save annotation time and render the labeling task feasible, we collect TP annotations at the plot synopsis level (synopses are a few paragraphs long compared to screenplays which are on average 120 pages long). An example is given in Figure 1. We then project the TP annotations via distant supervision onto screenplays and propose an end-to-end neural network model which identifies TPs in full length screenplays.

Our contributions can be summarized as follows: (a) we introduce TP identification as a new task for the computational analysis of screen plays that can benefit applications such as QA and summarization; (b) we create and make publicly available the **TuRnIng POint Dataset** (TRIPOD) [1] which contains 99 movies (3,329 synopsis sentences and 13,403 screenplay scenes) annotated with TPs; and (c) we present an end-to-end neural network model that identifies turning points in plot synopses and projects them onto scenes in screenplays, outperforming strong baselines based on state-of-the-art sentence representations and the expected position of TPs.

## 2 Related Work

Recent years have seen increased interest in the automatic analysis of long and complex narratives. Specifically, Machine Reading Comprehension (MRC) and Question Answering (QA) tasks are transitioning from investigating single short and clean articles or queries (Rajpurkar et al., 2016; Nguyen et al., 2016; Trischler et al., 2016) to large scale datasets that consist of complex stories (Tapaswi et al., 2016; Frermann et al., 2018; Kočiskỳ et al., 2018; Joshi et al., 2017) or require reasoning across multiple documents (Welbl et al., 2018; Wang et al., 2018; Dua et al., 2019; Yang et al., 2018). Tapaswi et al. (2016) introduce a multi-modal dataset consisting of questions over 140 movies, while Frermann et al. (2018) attempt to answer a single question, namely who is the perpetrator in 39 episodes of the well-known crime series CSI, again based on multi-modal informa-

[1] https://github.com/ppapalampidi/TRIPOD

tion. Finally, Kočiskỳ et al. (2018) recently introduced a dataset consisting of question-answer pairs over 1,572 movie screenplays and books.

Previous approaches have focused on fine-grained story analysis, such as inducing character types (Bamman et al., 2013, 2014) or understanding relationships between characters (Iyyer et al., 2016; Chaturvedi et al., 2017). Various approaches have also attempted to analyze the goal and structure of narratives. Black and Wilensky (1979) evaluate the functionality of story grammars in story understanding, Elson and McKeown (2009) develop a platform for representing and reasoning over narratives, and Chambers and Jurafsky (2009) learn fine-grained chains of events.

In the context of movie summarization, Gorinski and Lapata (2018) automatically generate an overview of the movie's genre, mood, and artistic style based on screenplay analysis. Gorinski and Lapata (2015) summarize full length screenplays by extracting an optimal chain of scenes via a graph-based approach centered around the characters of the movie. A similar approach has also been adopted by Vicol et al. (2018), who introduce the MovieGraphs dataset consisting of 51 movies and describe video clips with character-centered graphs. Other work creates animated story-boards using the action descriptions of screenplays (Ye and Baldwin, 2008), extracts social networks from screenplays (Agarwal et al., 2014a), or creates *xkcd* movie narrative charts (Agarwal et al., 2014b).

Our work also aims to analyze the narrative structure of movies, but we adopt a high-level approach. We advocate TP identification as a precursor to more fine-grained analysis that unveils character attributes and their relationships. Our approach identifies key narrative events and segments the screenplay accordingly; we argue that this type of preprocessing is useful for applications which might perform question answering and summarization over screenplays. Although our experiments focus solely on the textual modality, turning point analysis is also relevant for multimodal tasks such as trailer generation and video summarization.

## 3 The TRIPOD Dataset

The TRIPOD dataset contains 99 screenplays, accompanied with cast information (according to IMDb), and Wikipedia plot synopses annotated with turning points. The movies were selected from the Scriptbase dataset (Gorinski and Lapata, 2015) based on the following criteria: (a) maintaining a variation across different movie genres (e.g., action, romance, comedy, drama) and narrative types (e.g., flashbacks, time shifts); and (b) including screenplays that are faithful to the released movies and their synopses as much as possible. In Table 2, we present various statistics of the dataset.

| | Train | Test |
|---|---|---|
| movies | 84 | 15 |
| turning points | 420 | 75 |
| synopsis sentences | 2,821 | 508 |
| screenplay scenes | 11,320 | 2,083 |
| synopsis vocabulary | 7.9k | 2.8k |
| screenplay vocabulary | 37.8k | 16.8k |
| *per synopsis* | | |
| tokens | 729.8 (165.5) | 698.4 (187.4) |
| sentences | 35.4 (8.4) | 33.9 (9.9) |
| sentence tokens | 20.6 (9.5) | 20.6 (9.3) |
| *per screenplay* | | |
| tokens | 23.0k (6.6) | 20.9k (4.5) |
| sentences | 3.0k (0.9) | 2.8k (0.6) |
| scenes | 133.0 (61.1) | 138.9 (50.7) |
| *per scene* | | |
| tokens | 173.0 (235.0) | 150.5 (198.3) |
| sentences | 22.2 (31.5) | 19.9 (26.9) |
| sentence tokens | 7.8 (6.0) | 7.6 (6.4) |

Table 2: Statistics of the TRIPOD dataset; all means are shown with standard deviation in brackets.

Our motivation for obtaining TP annotations at the synopsis level (coarse-grained), instead of at the screenplay level (fine-grained) was twofold. Firstly, on account of being relatively short, synopses are easier to annotate than full-length screenplays, allowing us to scale the dataset in the future. Secondly, we would expect synopsis-level annotations to be more reliable and the degree of inter-annotator agreement higher; asking annotators to identify precisely where a turning point occurs might seem like looking for a needle in a haystack. An example of a synopsis with TP annotations is shown in Figure 1 for the movie "Panic Room". Each TP is colored differently, and both the chain of key events (colored text) and resulting segmentation (∞) are illustrated.

In an initial pilot study, the three authors acted as annotators for identifying TPs in movie synopses. They selected exactly one sentence per TP, under the assumption that all TPs are present. Based on the pilot, annotation instructions were devised and an annotation tool was created which allows to label synopses with TPs sentence-by-sentence. After piloting the annotation scheme on 30 movies, two new annotators were trained using our instructions and in a second study, they doubly annotated five movies. The remaining movies

in the dataset were then single annotated by the new annotators.

We computed inter-annotator agreement using two different metrics: (a) total agreement (TA), i.e., the percentage of TPs that two annotators agree upon by selecting the exact same sentence; and (b) annotation distance, i.e., the distance $d[p_i, tp_i]$ between two annotations for a given TP, normalized by synopsis length:

$$d[p_i, tp_i] = \frac{1}{N}|p_i - tp_i| \quad (1)$$

where $N$ is the number of synopsis sentences and $tp_i$ and $p_i$ are the indices of the sentences labeled with TP $i$ by two annotators. The mean annotation distance $D$ is then computed by averaging distances $d[p_i, tp_i]$ across all annotated TPs.

The TA between the two annotators in our second study was 64.00% and the mean annotation distance was 4.30% (StDev 3.43%). The annotation distance per TP is presented in Table 5 (last line), where it is compared with the automatic TP identification results (to be explained later).

We also asked our annotators to annotate the screenplays (rather than synopses) for a subset of 15 movies. This subset serves as our goldstandard test set. Annotators were given synopses annotated with TPs and were instructed to indicate for each TP which scenes in the screenplay correspond to it. Six of the 15 movies were doubly annotated, so that we could measure agreement. Since annotators were allowed to choose a variable number of scenes for each TP, this changes slightly our agreement metrics.

Total Agreement (TA) now is the percentage of TP scenes the annotators agree on:

$$TA = \frac{1}{T \cdot L} \sum_{i=1}^{T \cdot L} \frac{|S_i \cap G_i|}{|S_i \cup G_i|} \quad (2)$$

where $T$, $L$ are the TPs identified per annotator in a screenplay, and $S_i$ and $G_i$ are the indices of the scenes selected for TP $i$ by the two annotators. Partial Agreement (PA) is the percentage of TPs where there is an overlap of at least one scene:

$$PA = \frac{1}{T \cdot L} \sum_{i=1}^{T \cdot L} [S_i \cap G_i \neq \emptyset] \quad (3)$$

And annotation distance $D$ becomes the mean of the distances[2] $d[S_i, G_i]$ between two annotators normalized by $M$, the length of the screenplay:

$$d[S_i, G_i] = \frac{1}{M} \min_{(s \in S_i, g \in G_i)} |s - g| \quad (4)$$

The TA and PA between the two annotators were 35.48% and 56.67%, respectively. The mean annotation distance was 1.48% (StDev 2.93%). The TA shows that the annotators rarely indicate the same scenes, even if they are asked to annotate an event in the screenplay that is described by a specific synopsis sentence. However, they identify scenes which are in close proximity in the screenplay, as PA and annotation distance reveal. This analysis validates our assumption that annotating the synopses first limits the degree of overall disagreement.

[2]We compute the minimum distance between the two sets of scenes, since non-sequential scenes may be included in the same set. Hence, considering the center of the sets is not always representative of the TP scenes.

## 4 Turning Point Prediction Models

In this work, we aim to detect text segments which act as TPs. We first identify which *sentences* in plot synopses are TPs (Section 4.1); next, we identify which *scenes* in screenplays act as TPs via projection of *goldstandard* TP labels (Section 4.2); finally, we build an end-to-end system which identifies TPs in screenplays based on *predicted* TP synopsis labels (Section 4.3).

All models we propose in this paper have the same basic structure; they take text segments $i$ (sentences or scenes) as input and predict whether these act as TPs or not. Since the sequence, number, and labels of TPs are fixed (see Table 1), we treat TP identification as a binary classification problem (where 1 indicates that the text is a TP and 0 otherwise). Each segment is encoded into a multi-dimensional feature space $x_i$ which serves as input to a fully-connected layer with a single neuron representing the probability that $i$ acts as a TP. In the following, we describe several models which vary in the way input segments are encoded.

### 4.1 Identifying Turning Points in Synopses

**Context-Aware Model (CAM)** A simple baseline model would compute the semantic representation of each sentence in the synopsis using a pretrained sentence encoder. However, classifying segments in isolation without considering the context in which they appear, might yield inferior semantic representations. We therefore obtain richer representations for sentences by modeling their surrounding context. We encode the synopsis with a Bidirectional Long Short-Term Memory (BiLSTM; Hochreiter and Schmidhuber 1997) network; and obtain contextualized representation $cp_i$

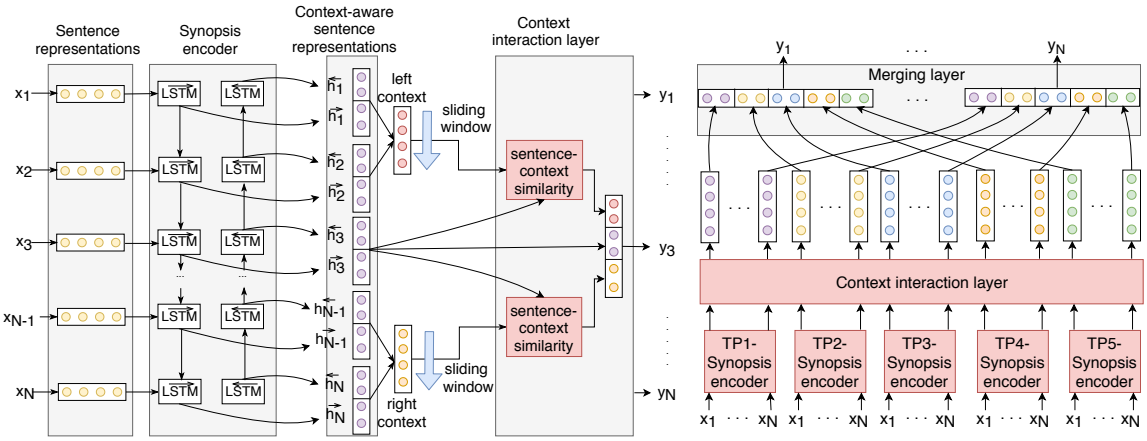


(a) Topic-Aware Model (TAM) (b) Multi-view TAM

Figure 2: Model overview for TP identification in synopses. On the left, sentence representations $x_i$ are contextualized via a synopsis encoder (BiLSTM layer) and after interacting with the left and right windows in the context interaction layer, the final sentence representation $y_i$ is computed. On the right, five different synopsis encoders are utilized, one per TP, and these different views of a synopsis sentence $x_i$ are combined in the merging layer.

for sentence $x_i$ by concatenating the hidden layers of the forward $\overrightarrow{h_i}$ and backward $\overleftarrow{h_i}$ LSTM, respectively: $cp_i = h_i = [\overrightarrow{h_i}; \overleftarrow{h_i}]$ (for a more detailed description, see the Appendix). Representation $cp_i$ is the input feature vector for our binary classifier. The model is illustrated in Figure 2a.

**Topic-Aware Model (TAM)** TPs by definition act as boundaries between different thematic units in a movie. Furthermore, long documents are usually comprised of topically coherent text segments, each of which contains a number of text passages such as sentences or paragraphs (Salton et al., 1996). Inspired by text segmentation approaches (Hearst, 1997) which measure the semantic similarity between sequential context windows in order to determine topic boundaries, we enhance our representations with a *context interaction* layer. The objective of this layer is to measure the similarity of the current sentence with its preceding and following context, thereby encoding whether it functions as a boundary between thematic sections. The enriched model with the context interaction layer is illustrated in Figure 2a.

After calculating contextualized sentence representations $cp_i$, we compute the representation of the left $lc_i$ and right $rc_i$ contexts of sentence $i$ (see Figure 2a, right-hand side). We select windows of fixed length $l$ and calculate $lc_i$ and $rc_i$ by averaging the sentence representations within each window. Next, we compute the semantic similarity of the current sentence with each context representation. Specifically, we consider the element-wise product $b_i$, cosine similarity $c_i$ and pairwise distance $u_i$ as similarity metrics:

$$b_i = cp_i \odot lc_i \qquad c_i = \frac{cp_i \cdot lc_i}{\|cp_i\| \|lc_i\|} \tag{5}$$

$$u_i = \frac{cp_i \cdot lc_i}{\max(\|cp_i\|_2 \cdot \|lc_i\|_2)} \tag{6}$$

The interaction representation of sentence $cp_i$ with its left context is the concatenation of $cp_i$, $fl_i$, and the above similarity values (i.e., $b_i, c_i, u_i$):

$$fl_i = [cp_i; lc_i; b_i; c_i; u_i] \tag{7}$$

The interaction representation $fr_i$ for the right context $rc_i$ is computed analogously. We obtain the final representation of sentence $i$ via concatenating $fl_i$ and $fr_i$: $y_i = [fl_i; fr_i; cp_i]$.

**TP-Specific Information** Another variation of our model is to use TP-specific encoders instead of a single one (see Figure 2b). In this case, we employ five different encoders for calculating five different representations for the current synopsis sentence $x_i$, each one with respect to a specific TP. These representations can be considered multiple views of the same sentence. We calculate the interaction of each view with the left and right context window, as previously, via the context interaction layer. Finally we compute the sentence representation $y_i$ by concatenating its individual context-enriched TP representations.

**Entity-Specific Information** We also enrich our model with information about entities. We first apply co-reference resolution to the plot synopses using the Stanford CoreNLP toolkit (Manning et al.,

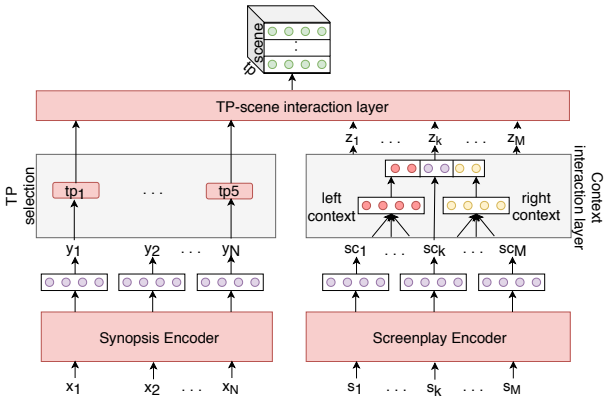


Figure 3: TAM overview for TP identification in screenplays. The synopsis and screenplay encoders contextualize synopsis sentences $x_i$ and screenplay scenes $s_i$, respectively. TPs are selected from contextualized synopsis sentences $y_i$ and a richer representation $sc_i$ is computed for $s_i$ via the context interaction layer. The similarity between sentence $tp_i$ and scene $z_i$ is computed by the TP–scene interaction layer.

2014) and substitute mentions of named entities whenever these are included in the IMDb cast list. We then obtain entity-specific sentence representations as follows. Our encoder uses a word embedding layer initialized with pre-trained entity embeddings and a BiLSTM for contextualizing word representations. We add an attention mechanism on top of the LSTM, which assigns a weight to each word representation. We compute the entity-specific representation $e_i$ for synopsis sentence $i$ as the weighted sum of its word representations (for more details, see the Appendix). Finally, entity enriched sentence representations $x'_i$ are obtained by concatenating generic vectors $x_i$ with entity-specific ones $e_i$: $x'_i = [x_i; e_i]$.

## 4.2 Identifying Turning Points in Screenplays

Identifying TPs in synopses serves as a testbed for validating some of the assumptions put forward in this work, namely that turning points mark narrative progression and can be identified automatically based on their lexical makeup. Nevertheless, we are mainly interested in the real-world scenario where TPs are detected in longer documents such as screenplays. Screenplays are naturally segmented into scenes, which often describe a self-contained event that takes place in one location, and revolves around a few characters. We therefore assume that scenes are suitable textual segments for signaling TPs in screenplays.

Unfortunately, we do not have any goldstandard information about TPs in screenplays. We provide distant supervision by constructing noisy labels based on goldstandard TP annotations in synopses (see the description below). Given sentences labeled as TPs in a synopsis, we identify scenes in the corresponding screenplay which are semantically similar to them. We formulate this task as a binary classification problem, where a sentence-scene pair is deemed either "relevant" or "irrelevant" for a given TP.

| | TP1 | TP2 | TP3 | TP4 | TP5 |
|---|---|---|---|---|---|
| theory | 10.00 | 25.00 | 50.00 | 75.00 | 94.50 |
| $\mu$ | 11.39 | 31.86 | 50.65 | 74.15 | 89.43 |
| $\sigma$ | 6.72 | 11.26 | 12.15 | 8.40 | 4.74 |

Table 3: Expected TP position based on screenwriting theory; mean position $\mu$ and standard deviation $\sigma$ in goldstandard synopses of our training set.

**Distant Supervision** Based on the screenwriting scheme of Hague (2017), TPs are expected to occur in specific parts of a screenplay (e.g., the Climax is likely to occur towards the end). We exploit this knowledge as a form of distant supervision. We estimate the mean position for each TP using the gold standard annotation of the plot synopses in our training set (normalized by the synopsis length). The results are shown in Table 3, together with the TP positions postulated by screenwriting theory. We observe that our estimates agree well with the theoretical predictions, but also that some TPs (e.g., TP2 and TP3) are more variable in their position than others (e.g., TP1 and TP5). This leads us to the following hypothesis: each TP is situated within a specific window in a screenplay. Scenes that lie within the window are semantically related to the TP, whereas all other scenes are unrelated. In experiments we calculate a window $\mu \pm \sigma$ based on our data (see Table 3).

We compute scene representations based on the sequence of sentences that comprise it using a BiLSTM equipped with an attention mechanism (see Section 4.1). The final scene representation $s$ is the weighted sum of the representations of the scene sentences. Next, the TP–scene interaction layer enriches scene representations with similarity values with each marked TP synopsis sentence $tp$ as shown in Equations (5)–(7).

We again augment the above-described base model with contextualized sentence and scene representations using a synopsis and screenplay encoder. The synopsis encoder is the same one used for our sentence-level TP prediction task (see Section 4.1). The screenplay encoder works in a sim-

ilar fashion over scene representations.

**Topic-Aware Model (TAM)** TAM enhances our screenplay encoder with information about topic boundaries. Specifically, we compute the representations of the left $lc_i$ and right $rc_i$ context window of the $i^{th}$ scene in the screenplay as described in Section 4.1. Next, we compute the final representation $z_i$ of scene $sc_i$ by concatenating the representations of the context windows $lc_i$ and $rc_i$ and the current scene $sc_i$: $z_i = [lc_i; sc_i; rc_i]$. There is no need to compute the similarity between scenes and context windows here as we now have goldstandard TP representations in the synopsis and employ the TP–scene interaction layer for the computation of the similarity between TPs and enriched scene representations $z_i$. Hence, we directly calculate in this layer a scene-level feature vector that encodes information about the scene, its similarity to TP sentences, and whether these function as boundaries between topics in the screenplay.

**Entity-Specific information** We can also employ an entity-specific encoder (see Section 4.1) for the representing the synopsis and scene sentences. Again, generic and entity-specific representations are combined via concatenation.

### 4.3 End-to-end TP Identification

Our ultimate goal is to identify TPs in screenplays without assuming any goldstandard information about their position in the synopsis. We address this with an end-to-end model which first predicts the sentences that act as TPs in the synopsis (e.g., TAM in Section 4.1) and then feeds these predictions to a model which identifies the corresponding TP scenes (e.g., TAM in Section 4.2).

## 5 Experimental Setup

**Training** We used the Universal Sentence Encoder (USE; Cer et al. 2018) as a pre-trained sentence encoder for all models and tasks; its performance was superior to BERT (Devlin et al., 2018) and other related pre-trained encoders (for more details, see the Appendix). Since the binary labels in both prediction tasks are imbalanced, we apply class weights to the loss function of our models. We weight each class by its inverse frequency in the training set (for more implementation details, see the Appendix).

**Inference** During inference in our first task (i.e., identification of TPs in synopses), we select one sentence per TP. Specifically, we want to track the five sentences with the highest posterior probability of being TPs and sequentially assign them TP labels based on their position. However, it is possible to have a cluster of neighboring sentences with high probability, even though they all belong to the same TP. We therefore constrain the sentence selection for each TP within the window of its expected position, as calculated in the distribution baseline (Section 4.2).

For models which predict TPs in screenplays, we obtain a probability distribution over all scenes in a screenplay indicating how relevant each is to the TPs of the corresponding plot synopsis. We find the peak of each distribution and select a neighborhood of scenes around this peak as TP-relevant ones. Based on the goldstandard annotation, each TP corresponds to 1.77 relevant scenes on average (StDev 1.23). We therefore consider a neighborhood of three relevant scenes per TP.

| | *TA* | *D* |
|---|---|---|
| Baseline | 31.00 | 9.65 (4.41) |
| CAM | 33.00 | 7.44 (8.09) |
| TAM | 36.00 | 7.11 (7.98) |
| + TP views | **39.00** | **6.52** (**7.72**) |
| + entities | 38.00 | 6.91 (7.65) |

(a) Development set

| | *TA* | *D* |
|---|---|---|
| Random | 2.00 | 37.79 (25.33) |
| Theory baseline | 22.00 | 7.47 (6.75) |
| Distribution baseline | 28.00 | 7.28 (6.23) |
| TAM | 34.67 | **6.80** (**5.19**) |
| + TP views | 38.57 | 7.47 (7.48) |
| + entities | **41.33** | 7.30 (7.21) |
| *Human agreement* | 64.00 | 4.30 (3.43) |

(b) Test set

Table 4: Identification of TPs in plot synopses; results are shown in percent (*TA*: mean Total Agreement; *D*: annotation distance; standard deviation in brackets).

| TAM | TP1 | TP2 | TP3 | TP4 | TP5 |
|---|---|---|---|---|---|
| + TP views | **6.09** | 9.45 | 10.72 | 6.91 | 4.26 |
| + entities | 7.18 | **9.35** | **9.86** | **5.23** | **3.48** |
| *Human agreement* | 3.33 | 5.00 | 10.58 | 1.07 | 1.53 |

Table 5: Mean annotation distance *D* (test set); results are shown per TP on the synopsis identification task.

## 6 Results

**TP Identification in Synopses** Table 4a reports our results on the development set (we extracted 20 movies from the original training set) which aim at comparing various model instantiations for the TP identification task. Specifically, we report the performance of a baseline model which is nei-

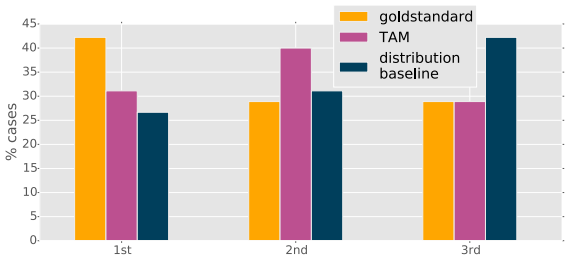


Figure 4: Rankings (shown as proportions) of synopsis highlights produced by aggregating goldstandard TP annotations, those predicted by the distribution baseline, and our model (TAM + TP views).

| | *TA* | *PA* | *D* |
|---|---|---|---|
| Theory baseline | 8.66 | 10.67 | 10.45 (9.14) |
| Distribution baseline | 6.67 | 9.33 | 10.84 (8.94) |
| tf*idf similarity | 0.74 | 1.33 | 53.07 (31.83) |
| tf*idf + distribution | 4.44 | 6.67 | 13.33 (11.51) |
| CAM | 11.11 | 16.00 | 10.23 (11.23) |
| + entities | **14.18** | **17.33** | 12.77 (12.61) |
| TAM | 10.63 | 13.33 | **8.94** (**9.39**) |
| + entities | 10.63 | 13.33 | 10.15 (10.56) |
| TAM End2end | 7.87 | 9.33 | 10.16 (10.74) |
| *Human agreement* | *35.48* | *56.67* | *1.48* (*2.93*) |

Table 6: Identification of TPs in screenplays; results are shown in percent using five-fold cross validation (*TA*: mean Total Agreement; *PA*: Partial Agreement; *D*: annotation distance *D*; standard deviation in brackets).

ther context-aware nor utilizes topic boundary information against CAM and TAM. We also show two variants of TAM enhanced with TP-specific encoders (+ TP views) and entity-specific information (+ entities). Model performance is measured using the evaluation metrics of Total Agreement (*TA*) and annotation distance (*D*), normalized by synopsis length (equation (1)).

The baseline model presents the lowest performance among all variants which suggests that state-of-the-art sentence representations on their own are not suitable for our task. Indeed, when contextualizing the synopsis sentences via a BiLSTM layer we observe an absolute increase of 4.00% in terms of *TA*. Moreover, the addition of a context interaction layer (see TAM row in Table 4a) yields an absolute *TA* improvement of 4.00% compared to CAM. Combining different TP views further improves by 3.00%, reaching a *TA* of 39.00%, and reducing *D* to 6.52%.

Table 4b shows our results on the test set. We compare TAM, our best performing model against two strong baselines. The first one selects sentences that lie on the expected positions of TPs according to screenwriting theory; while the second one selects sentences that lie on the peaks of the empirical TP distributions in the training set (Section 4.2). As we can see, TAM (+ TP views) achieves a *TA* of 38.57% compared to 22.00% for the distribution baseline. And although entity-specific information does not have much impact on the development set, it yields a 2.76% improvement on the test set. A detailed break down of results per TP is given in Table 5. Interestingly, our model resembles human behavior (see row *Human agreement*): TPs 1, 4, and 5 are easiest to distinguish, whereas TPs 2 and 3 are hardest and frequently placed at different points in the synopsis.

We also conducted a human evaluation experiment on Amazon Mechanical Turk (AMT). AMT workers were presented with a synopsis and "highlights", i.e., five sentences corresponding to TPs. We obtained highlights from goldstandard annotations, the distribution baseline, and TAM (+ TP views). AMT workers were asked to read the synopsis and rank the highlights from best to worst according to the following criteria: (1) the quality of the plotline that they form; (2) whether they include the most important events and plot twists of the movie; and (3) whether they provide some description of the events in the beginning and end of the movie. In Figure 4 we show, proportionally, how often our participants ranked each model 1st, 2nd, and so on. Perhaps unsurprisingly, goldstandard TPs were considered best (and ranked 1st 42% of the time). TAM is ranked best 30% of the time, followed by the distribution baseline which was only ranked first 26% of the time. Overall, the average ranking positions for the goldstandard, TAM, and the baseline are 1.87, 1.98, and 2.16, respectively. Human evaluation therefore validates that our model outperforms the position-based baselines.

**TP Identification in Screenplays** Our results are summarized in Table 6. For this task, we performed five-fold crossvalidation over our original goldstandard set to obtain a test-development split (recall we do not have goldstandard annotations for training). We report Total Agreement (*TA*), Partial Agreement (*PA*), and annotation distance *D*, normalized by screenplay length (Equations (2)–(4)).

Aside from the theory and distribution-based baselines, we also experimented[3] with a com-

[3]Common segmentation approaches such as TextTiling (Hearst, 1997) perform poorly on our task and we do not report them due to space constraints.

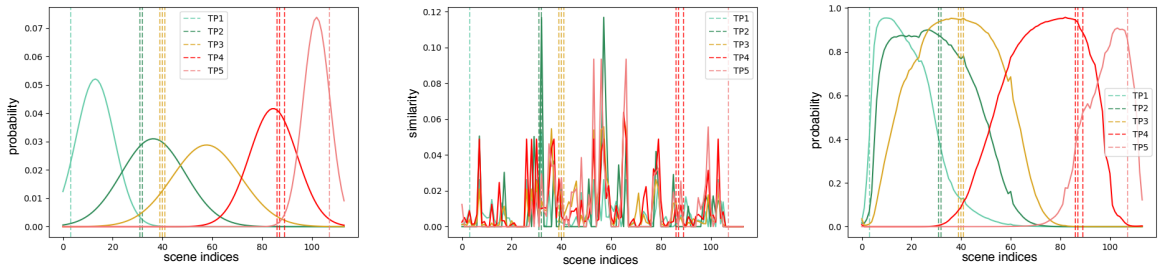


Distribution baseline tf*idf similarity TAM

Figure 5: Probability distributions over the scenes of the screenplay for the movie "Juno"; x-axis: scene indices, y-axis: probability that the scene is relevant to a specific TP. Vertical dashed lines are goldstandard TP scenes.

mon IR baseline which considers TP synopsis sentences as queries and retrieves a neighborhood of semantically similar scenes from the screenplay using tf*idf similarity. Specifically, we compute the maximum tf*idf similarity for all sentences included in the respective scene. We empirically observed that tf*idf's behavior can be erratic selecting scenes in completely different sections of the screenplay, and therefore constrain it by selecting scenes only within the windows determined by the position distributions ($\mu \pm \sigma$) for each TP. As far as our own models are concerned, we report results with goldstandard TP labels for CAM and TAM on their own and enriched with entity information. We also built and end-to-end system based on TP predictions from TAM.

As can be seen in Table 6, tf*idf approaches perform worse than position-related baselines. Overall, similar vocabulary across scenes and mentions of the same entities throughout the screenplay make tf*idf approaches insufficient for our tasks. The best performing model is TAM confirming our hypothesis that TPs are not just isolated key events, but also mark boundaries between thematic units and, therefore, segmentation-inspired approaches can be beneficial for the task. Results for entities are somewhat mixed; for CAM, the entity-specific information improves *TA* and *PA* but increases *D*, while it does not seem to make much difference for TAM. The performance of the end-to-end TAM model drops slightly compared to the same model using goldstandard TP annotations. However, it still remains competitive against the baselines, indicating that tracking TPs in screenplays fully automatically is feasible.

In Figure 5, we visualize the posterior distribution of various models over the scenes of the screenplay for the movie "Juno". The first panel shows the distribution baseline alongside goldstandard TP scenes (vertical lines). We observe that the distribution baseline provides a good approximation of relevant TP positions (which validates its use in the construction of noisy labels, Section 4.2), even though it is not always accurate. For example, TPs 1 and 3 lie outside the expected window in "Juno".

The second panel presents the TP predictions according to tf*idf similarity. We observe that scenes located in entirely different parts of the screenplay present high similarity scores with respect to a given TP due to vocabulary uniformity and mentions of the same entities throughout the screenplay. In the next panel we present the predictions of TAM. Adding synopsis and screenplay encoders yields smoother distributions increasing the probability of selecting TP scenes inside distinct regions of the screenplay, with sharper peaks and higher confidence.

## 7 Conclusions

We proposed the task of turning point identification in screenplays as a means of analyzing their narrative structure. We demonstrated that automatically identifying a sequence of key events and segmenting the screenplay into thematic units is feasible via an end-to-end neural network model. In future work, we will investigate the usefulness of TPs for summarization and question answering. We will also scale the TRIPOD dataset and move to a multi-modal setting where TPs are identified directly in video data.

## Acknowledgments

We thank the anonymous reviewers for their feedback. We gratefully acknowledge the support of the European Research Council (Lapata; award 681760, "Translating Multiple Modalities into Text") and of the Leverhulme Trust (Keller; award IAF-2017-019).

## References


Apoorv Agarwal, Sriramkumar Balasubramanian, Jiehan Zheng, and Sarthak Dash. 2014a. Parsing Screenplays for Extracting Social Networks from Movies. In *Proceedings of the 3rd Workshop on Computational Linguistics for Literature*, pages 50–58, Gothenburg, Sweden.

Apoorv Agarwal, Sarthak Dash, Sriramkumar Balasubramanian, and Jiehan Zheng. 2014b. Using Determinantal Point Processes for Clustering with Application to Automatically Generating and Drawing xkcd Movie Narrative Charts. In *Proceedings of the 2nd Academy of Science and Engineering International Conference on Big Data Science and Computing*, Stanford, California.

Apoorv Agarwal, Jiehan Zheng, Shruti Kamath, Sriramkumar Balasubramanian, and Shirin Ann Dey. 2015. Key Female Characters in Film Have More to Talk About Besides Men: Automating the Bechdel Test. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 830–840, Denver, Colorado.

David Bamman, Brendan OConnor, and Noah A Smith. 2013. Learning latent personas of film characters. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 352–361, Sofia, Bulgaria.

David Bamman, Ted Underwood, and Noah A Smith. 2014. A bayesian mixed effects model of literary character. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 370–379.

John B Black and Robert Wilensky. 1979. An evaluation of story grammars. *Cognitive science*, 3(3):213–229.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 602–610. Association for Computational Linguistics.

Snigdha Chaturvedi, Mohit Iyyer, and Hal Daume III. 2017. Unsupervised learning of evolving relationships between literary characters. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*.

James E Cutting. 2016. Narrative theory and the dynamics of popular movies. *Psychonomic bulletin & review*, 23(6):1713–1743.

Cristian Danescu-Niculescu-Mizil, Justin Cheng and-Jon Kleinberg, and Lillian Lee. 2012. You had me at hello: How phrasing affects memorability. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers–Volume 1*, pages 892–901, ??

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. *arXiv preprint arXiv:1903.00161*.

Ahmed Elgohary, Chen Zhao, and Jordan Boyd-Graber. 2018. A dataset and baselines for sequential open-domain question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1077–1083.

Mich Elsner. 2012. Character-based kernels for novelistic plot structure. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 634–644, Avignon, France.

David Elson and Kathleen McKeown. 2009. Extending and evaluating a platform for story understanding. In *Proceedings of the AAAI 2009 Spring Symposium on Intelligent Narrative Technologies II*, page ??, ??

Lea Frermann, Shay B Cohen, and Mirella Lapata. 2018. Whodunnit? crime drama as a case for natural language understanding. *Transactions of the Association of Computational Linguistics*, 6:1–15.

Gustav Freytag. 1896. *Freytag's technique of the drama: an exposition of dramatic composition and art*. Scholarly Press.

Philip John Gorinski and Mirella Lapata. 2015. Movie script summarization as graph-based scene extraction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1066–1076.

Philip John Gorinski and Mirella Lapata. 2018. Whats this movie about? a joint neural network architecture for movie content analysis. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1770–1781.

Michael Hague. 2017. *Storytelling Made Easy: Persuade and Transform Your Audiences, Buyers, and Clients – Simply, Quickly, and Profitably*. Indie Books International.

Marti A Hearst. 1997. Texttiling: Segmenting text into multi-paragraph subtopic passages. *Computational linguistics*, 23(1):33–64.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Mohit Iyyer, Anupam Guha, Snigdha Chaturvedi, Jordan Boyd-Graber, and Hal Daumé III. 2016. Feuding families and former friends: Unsupervised learning for dynamic fictional relationships. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1534–1544.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gáabor Melis, and Edward Grefenstette. 2018. The narrativeqa reading comprehension challenge. *Transactions of the Association of Computational Linguistics*, 6:317–328.

Bernhard Kratzwald and Stefan Feuerriegel. 2018. Adaptive document retrieval for deep question answering. *arXiv preprint arXiv:1808.06528*.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

Eric T. Nalisnick and Henry S. Baird. 2013. Character-to-character sentiment analysis in shakespeare's plays. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 479–483, Sofia, Bulgaria.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

Patrice Pavis. 1998. *Dictionary of the theatre: Terms, concepts, and analysis*. University of Toronto Press.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Anil Ramakrishna, Nikolaos Malandrakis, Elizabeth Staruk, and Shrikanth Narayanan. 2015. A quantitative analysis of gender differences in movies using psycholinguistic normatives. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing.*, pages 1996–2001, Lisbon, Portugal.

Gerard Salton, Amit Singhal, Chris Buckley, and Mandar Mitra. 1996. Automatic text decomposition using text segments and text themes. In *Proceedings of the the seventh ACM conference on Hypertext*, pages 53–65, Bethesda, Maryland.

Maarten Sap, Marcella Cindy Prasettio, Ari Holtzman, Hannah Rashkin, and Yejin Choi. 2017. Connotation frames of power and agency in modern films. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2329–2334, Copenhagen, Denmark.

Makarand Tapaswi, Yukun Zhu, Rainer Stiefelhagen, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2016. Movieqa: Understanding stories in movies through question-answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4631–4640.

Kristin Thompson. 1999. *Storytelling in the new Hollywood: Understanding classical narrative technique*. Harvard University Press.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2016. Newsqa: A machine comprehension dataset. *arXiv preprint arXiv:1611.09830*.

Paul Vicol, Makarand Tapaswi, Lluis Castrejon, and Sanja Fidler. 2018. Moviegraphs: Towards understanding human-centric situations from videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8581–8590.

Yizhong Wang, Kai Liu, Jing Liu, Wei He, Yajuan Lyu, Hua Wu, Sujian Li, and Haifeng Wang. 2018. Multi-passage machine reading comprehension with cross-passage answer verification. *arXiv preprint arXiv:1805.02220*.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association of Computational Linguistics*, 6:287–302.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

Patrick Ye and Timothy Baldwin. 2008. Towards Automatic Animated Storyboarding. In *Proceedings of the 23rd Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence*, pages 578–583, Chicago, Illinois.

# Latent Suicide Risk Detection on Microblog via Suicide-Oriented Word Embeddings and Layered Attention

**Lei Cao**[1,2] and **Huijun Zhang**[1,2] and **Ling Feng**[1,2] and **Zihan Wei**[3]
**Xin Wang**[1,2] and **Ningyun Li**[1,2] and **Xiaohao He**[1,2]
[1]Department of Computer Science and Technology, Tsinghua University, Beijing, China
[2]Beijing National Research Center for Information Science and Technology, Beijing, China
[3]School of Software, Beihang University, Beijing, China
{cao-l17,zhang-hj17,xin-wang18,liny18}@mails.tsinghua.edu.cn,
fengling@mail.tsinghua.edu.cn, sy1721114@buaa.edu.cn,
hexh17@mails.tsinghua.edu.cn

## Abstract

Despite detection of suicidal ideation on social media has made great progress in recent years, people's implicitly and anti-real contrarily expressed posts still remain as an obstacle, constraining the detectors to acquire higher satisfactory performance. Enlightened by the hidden "tree holes" phenomenon on microblog, where people at suicide risk tend to disclose their inner real feelings and thoughts to the microblog space whose authors have committed suicide, we explore the use of tree holes to enhance microblog-based suicide risk detection from the following two perspectives. (1) We build suicide-oriented word embeddings based on tree hole contents to strength the sensibility of suicide-related lexicons and context based on tree hole contents. (2) A two-layered attention mechanism is deployed to grasp intermittently changing points from individual's open blog streams, revealing one's inner emotional world more or less. Our experimental results show that with suicide-oriented word embeddings and attention, microblog-based suicide risk detection can achieve over 91% accuracy. A large-scale well-labelled suicide data set is also reported in the paper.

## 1 Introduction

Suicide is a growing problem in today's society. Each year nearly 800,000 people worldwide commit suicide, which is one person every 40 seconds, and there are many more who attempt it (Organization et al., 2014). Suicide prevention will conduce to human's well-being, of which timely sensing suicide ideation is an essential task.

*Existing Solutions.* Traditional suicide risk assessment like Suicide Probability Scale (Bagge and Osman, 1998), Adult Suicide Ideation Questionnaire (Fu et al., 2007), Suicidal Affect-Behavior-Cognition Scale (Harris et al., 2015), etc. requires respondents to either fill in a questionnaire or participate in a professional interview. However, they are applicable to a small group of people. Particularly for the people who are suffering but tend to hide inmost thoughts and refuse to seek helps from others, the approaches cannot function (Essau, 2005; Rickwood et al., 2007; Zachrisson et al., 2006).

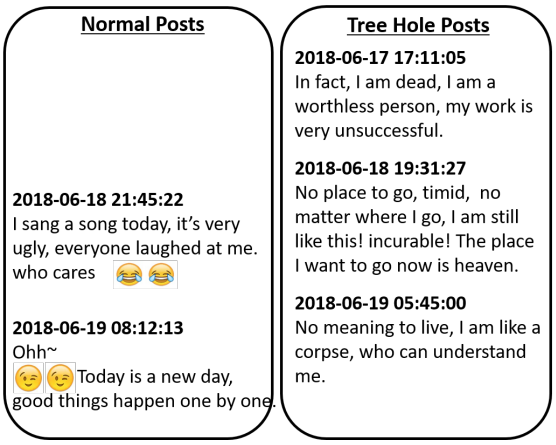


Figure 1: An example of one user's normal posts vs. his/her hidden tree hole posts on microblog.

Recently, the penetration of social media (like forums and microblogs) and its large-scale, low-cost, and open advantages enable researchers to overcome the limitation and timely detect individual's suicide ideation. Despite great efforts have been made (Alambo et al., 2019; Cheng et al., 2017; Du et al., 2018; Sawhney et al., 2018; Coppersmith et al., 2018; Viouls et al., 2018), the social media based detection performance is constrained due to implicitly and anti-real contrarily expressed posts from people who hide their inmost feelings and thoughts on social media. To illustrate, let's see a user's normal blogs vs. his/her hidden posts in a microblog tree hole in Figure 1. Usually, people of suicidal tendency (referred to as suicidal people in the study) tend to disclose their real inner feelings on the microblog space whose authors have committed suicide. Hundreds of such



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1718–1728,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| | Normal Posts | Tree Hole Posts |
|---|---|---|
| Avg proportion of self-concern words per post | 14% | 50% |
| Avg proportion of others-concern words per post | 68% | 12% |
| Avg proportion of suicide-related words per post | 5% | 95% |
| Avg number of posts per user in a year | 69.3 | 52.1 |
| Total number of posts from all users in a year | 252,901 | 190,087 |

Table 1: Statistics of suicidal users' normal posts and hidden tree hole posts on Microblog based on 3652 users from May 1, 2018 to April 30, 2019.

tree holes exist on Sina microblog. An example tree hole contains 1,700,000 posts from suicide attempts. In Figure 1, we can sense a severe hopelessness from the tree hole posts, but not from the normal posts, and the user even expresses an uplift feeling. After cross-examining suicidal users' normal and hidden posts as shown in table 1, we discover that users' hidden posts in the tree hole have more self-concerns, less others-concerns, and illustrate suicidal thoughts more directly. In comparison, suicidal users normal posts contain much less suicide-related words, and the users are reluctant to show their suicidal feelings in their normal posts. Moreover, the data of self-concern and others-concern shown in the first two rows even indicate that people with suicide risk are not willing to talk about themselves in their normal posts, which takes great challenges to detect suicide risk from users' open normal posts.

*Our Work.* The aim of the study is to break through the above limitation to achieve a new state-of-art performance on latent suicide risk detection from one's open normal microblogs. We leverage tree hole posts from the following two perspectives. (1) We construct suicide-oriented word embeddings based on tree hole contents to strength the sensibility of suicide-related lexicons and context based on tree hole contents. (2) A two-layered attention mechanism is deployed to grasp intermittently changing points from individual's open blog streams, revealing one's inner emotional world more or less. Our experimental results on 252,901 open normal microblogs show that with suicide-oriented word embeddings and two-layered attention, latent suicide risk detection can achieve over 91% accuracy.

In summary, the paper makes the following contributions.

- We build effective suicide-oriented word embeddings to better understand the implicit meanings of words contained in users' normal posts, and propose a two-layered attention model to capture the changing points which reveal suicide risk from individuals' blog streams. Our latent suicide risk detection from users normal posts not only outperforms the state-of-the-art approaches, but also are powerful enough in detecting implicitly and anti-real contrarily expressed posts.
- We construct a large-scale data set from 3652 suicidal people in the period of [May 1, 2018 to April 30, 2019], containing 252,901 normal posts on Sina microblog. The data set can further facilitate people's well-being studies in the computer science and psychology fields.

## 2 Related Work

### 2.1 Traditional Questionnaire-based Suicide Risk Assessment

Researchers have developed a number of psychological measurements to access individual's suicide risk (Pestian et al., 2017), such as Suicide Probability Scale (SPS) (Bagge and Osman, 1998), Depression Anxiety Stress Scales-21 (DASS-21) (Crawford and Henry, 2003; Henry and Crawford, 2005), Adult Suicide Ideation Questionnaire (Fu et al., 2007), Suicidal Affect-Behavior-Cognition Scale (Harris et al., 2015), functional Magnetic Resonance Imaging (fMRI) signatures (Just et al., 2017), and so on. While these measurements are professional and effective, they require respondents to either fill in a questionnaire or participate in a professional interview, constraining its touching to suicidal people who have low motivations to seek help from professionals (Essau, 2005; Rickwood et al., 2007; Zachrisson et al., 2006). A recent study found out that taking a suicide assessment may bring negative effect to individuals with depressive symptoms (Harris and Goh, 2017).

### 2.2 Suicide Risk Detection from Social Media

Recently, detection of suicide risk from social media is making great progress due to the advantages of reaching massive population, low-cost, and real-time (Braithwaite et al., 2016). Harris et al. (2014) reported that suicidal users tend to spend more time online, have greater likelihood of developing online personal relationships, and greater use of online forums.

***Suicide Risk Detection from Suicide Notes.*** Pestian et al. (2010) built a suicide note classifier used machine learning techniques, which performs better than human psychologists in distinguishing fake online suicide notes from real ones. Huang et al. (2007) hunted suicide notes based on lexicon-based keyword matching on MySpace.com (a popular site for adolescents and young adults, particularly sexual minority adolescents with over 1 billion registered users worldwide) to check whether users have an intent to commit suicide.

***Suicide Risk Detection from Community Forums.*** Li et al. (2013) applied textual sentiment analysis and summarization techniques to users' posts and posts' comments in a Chinese web forum in order to identify suicide expressions. Masuda et al. (2013) examined online forums in Japan, and discovered that the number of communities which a user belongs to, the intransitivity, and the fraction of suicidal neighbors in the social network contributed the most to suicide ideation. De Choudhury et al. (2016) built a logistic regression framework to analyze Reddit users' shift tendency from mental health sub-communities to a suicide support sub-community. heightened self-attentional focus, poor linguistic coherence and coordination with the community, reduced social engagement and manifestation of hopelessness, anxiety, impulsiveness and loneliness in shared contents are distinct markers characterizing these shifts. Based on the suicide lexicons detailing suicide indicator, suicide ideation, suicide behavior, and suicide attempt, Alambo et al. (2019) built four corresponding semantic clusters to group semantically similar posts on Reddit and questions in a questionnaire together, and used the clusters to assess the aggregate suicide risk severity of a Reddit post.

***Suicide Risk Detection from Microblogs.*** Jashinsky et al. (2014) used search keywords and phrases relevant to suicide risk factors to filter potential suicide-related tweets, and observed a strong correlation between Twitter-derived suicide data and real suicide data, showing that Twitter can be viewed as a viable tool for real-time monitoring of suicide risk factors on a large scale. The correlation study between suicide-related tweets and suicidal behaviors was also conducted based on a cross-sectional survey (Sueki, 2015), where participants answered a self-administered online questionnaire, containing questions about Twitter use, suicidal behaviour, depression and anxiety, and demographic characteristics. The survey result showed that Twitter logs could help identify suicidal young Internet users.

Based on eight basic emotion categories (joy, love, expectation, anxiety, sorrow, anger, hate, and surprise), Ren et al. (2015) examined three accumulated emotional traits (i.e., emotion accumulation, emotion covariance, and emotion transition) as the special statistics of emotions expressions in blog streams for suicide risk detection. A linear regression algorithm based on the three accumulated emotional traits was employed to examine the relationship between emotional traits and suicide risk. The experimental result showed that by combining all of three emotion traits together, the proposed model could generate more discriminative suicidal prediction performance.

Natural language processing and machine learning techniques, such as Latent Dirichlet Allocation (LDA), Logistic Regression, Random Forest, Support Vector Machine, Naive Bayes, Decision Tree, etc., were applied to identify users' suicidal ideation based on their linguistic contents and online behaviors on Sina Weibo (Guan et al., 2014; Zhang et al., 2014a; Huang et al., 2014; Zhang et al., 2014b; Huang et al., 2015; Guan et al., 2015; Cheng et al., 2017) and Twitter (Abboute et al., 2014; Burnap et al., 2015; O'Dea et al., 2015; Coppersmith et al., 2015). Deep learning based architectures like Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Long Short-Term Memory Neural Network (LSTM), etc., were also exploited to detect users' suicide risk on social media (Du et al., 2018; Sawhney et al., 2018; Coppersmith et al., 2018). Viouls et al. (2018) detected users' change points in emotional well-being on Twitter through a martingale framework, which is widely used for change detection in data streams.

| Category | Suicide Ideation | Suicide behavior | Psychache | Mental illness | Hopeless |
|---|---|---|---|---|---|
| Number | 586 | 88 | 403 | 48 | 188 |
| Words/phrases | want to die<br>escape | seppuku<br>hypnotics | want to cry<br>loneliness | depression<br>hallucination | dead end<br>despair |

Table 2: Representative words/phrases of Chinese suicide dictionary

## 3 Suicide-oriented Word Embeddings

Although there are some good works on word embeddings (Mikolov et al., 2013; Pennington et al., 2014; Joulin et al., 2016; Devlin et al., 2018), lack of domain information limits their performance on suicide detection.

Given a serious of pre-trained word embeddings and suicide-related dictionary, we aim to generate suicide-related word embeddings which can strengthen the sensibility of suicide-related lexicons and context. In this study, we call them Suicide-oriented Word Embeddings, as we take advantage of the information from Tree Hole's data set which can be regarded as a kind of latent emotional expressions of individuals. As suicidal individuals in social media often use some suicide-related words/phrases in their posts, we employ Chinese suicide dictionary (Lv et al., 2015) to generate suicide domain associated embeddings. The Chinese suicide dictionary analyzes 1.06 million active blog users' posts and lists 2168 words/phrases related to suicidal ideation. These words/phrases belong to 13 categories and each word/phrase is assigned with a risk weight from 1 to 3 which indicates the relevance of suicide. We list 5 representative categories in table 2.

Since pre-trained word embeddings already contain rich semantic information and contextual information, we only need to enrich existing word embeddings with suicide-related information.

We employ a masked classification task to do this. Generally, a sentence should contains suicide-related words/phrases if it express suicidal ideation. Hence, We select 10,000 sentences[1] from Tree Hole's data set and ensure every sentence contains more than one word/phrase appeared in Chinese suicide dictionary.

Moreover, we utilize the selected sentences to do a suicidal expression classification. A sentence is regarded as suicidal expression only if it includes at least one suicide-related word/phrase. In this way, we do a sentence-level classification to refine pre-trained word embeddings and let them understand which word/phrase is relevant to suicide expression.

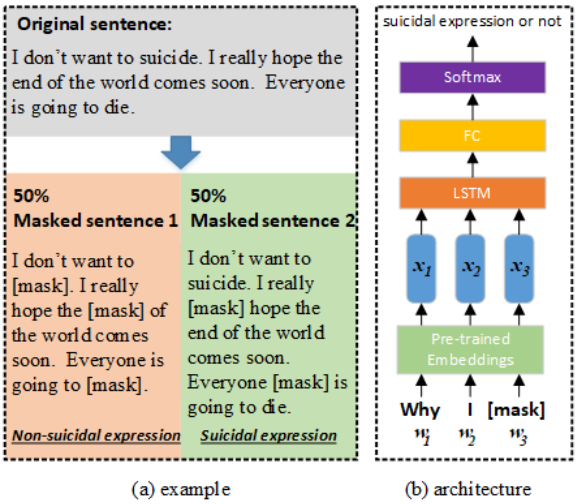


(a) example (b) architecture

Figure 2: An example of the strategy for embeddings training is shown on the left and the architecture of the masked classification task to train the suicide-oriented word embeddings is on the right.

In training details, for each epoch, we randomly select 50% sentences to replace all suicide-related words/phrases with "[mask]". Especially, for the rest 50% sentences, we randomly insert two "[mask]" into every sentence to avoid the suicidal expression classifier classifying sentence only based on whether it contains word "[mask]".

An example is given in Figure 2 (a). **Masked sentence 1** is the sentence that we replace all suicide-related words/phrases with "[mask]" and **Masked sentence 2** is the sentence that we randomly insert two "[mask]". We label **Masked sentence 1** as 0 (non-suicide) and **masked sentence 2** as 1 (suicide).

As there is no clear boundary between suicide-related words/phrases and others in pre-trained word embeddings, through this suicidal expression classification we force suicide-related words/phrases to be enriched with domain information and let all the suicide-related word/phrases

[1]Through out this work, a "sentence" can be a piece of text, rather than an actual linguistic sentence. It may contains more than one actual sentence.

contain the relationship with suicidal ideation. After classification model converge in Tree Hole's data set, we obtain suicide-oriented word embeddings which contain both semantic information from pre-trained word embeddings and suicide-information from suicide dictionary.

As illustrated in Figure 2, given a sentence $A = \{w_1, w_2, .., w_n\}$ written by a user in Tree Hole, where $n$ is the length of a sentence, the aim of suicidal expression classification is to classify whether this sentence contains expression about suicidal ideation or not. In this case, we define $X = \{x_1, x_2, .., x_n\} \in R^{n \times d_e}$ as the word embeddings of $A$, where $d_e$ is the dimension of embeddings. Figure 2 shows the architecture of the suicidal expression classification Model.

We employ a LSTM layer to extract text feature from $A$ followed by a fully connected layer for classification. We feed the word embeddings $X$ into the LSTM as following:

$$\begin{aligned} h_t &= \textbf{LSTM}(x_i, h_{t-1}), \\ [k_1, k_2] &= softmax((H^a W_1 + b_1)^T W_2 + b_2) \end{aligned} \tag{1}$$

where $h_{t-1}, h_t$ represent the hidden states at time $t-1$ and $t$, $H^a = \{h_1, h_2, ..., h_n\} \in R^{n \times d_e}$ is sentence representation of $A$, $[k_1, k_2]$ stand for the possibility of whether the sentence contains expression about suicidal ideation or not. $W_1 \in R^{d_e \times 1}$, $W_2 \in R^{n \times 2}$, $b_1 \in R^{1 \times 1}$ and $b_2 \in R^{1 \times 2}$ are trainable parameters.

## 4 Suicide Detection Model (SDM) based on Suicide-oriented Word Embeddings and Attention Mechanism

Given a sequential of posts $\hat{T}$ from one user, $\hat{T} = \{(s_1, p_1), (s_2, p_2), ..., (s_m, p_m)\}$ , where $m$ denotes the number of posts, $(s_i, p_i)$ stand for text and picture from i-th post. The aim is to detect whether the user at risk of suicide or not. Let $\hat{X} = \{x_1, x_2, .., x_n\} \in R^{n \times d_e}$ be the word embeddings of $s_i$, where $n$ represents the length of $s_i$ and $d_e$ is the dimension of embeddings. Figure 3 shows the architecture of the proposed two-layered attention model.

### 4.1 Feature Extraction

**Text Feature Extraction.** We employ a LSTM layer and attention mechanism to extract text feature from $s_i$. We feed the word embeddings $\hat{X}$ into

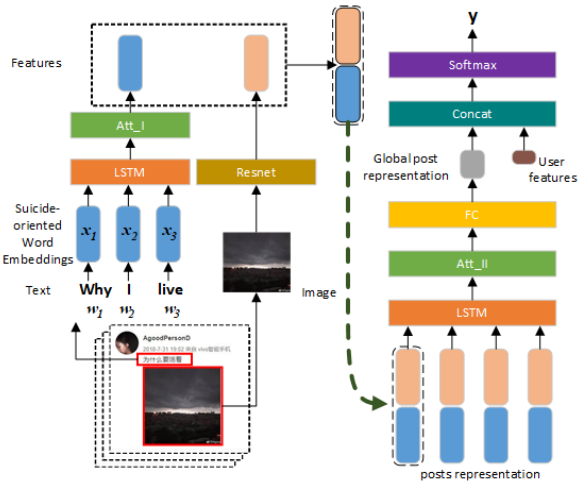


Figure 3: Architecture of the Suicide Risk Detection model.

the LSTM as following:

$$h_t = \textbf{LSTM}(x_t, h_{t-1}) \tag{2}$$

where $h_{t-1}, h_t$ represent the hidden states at time $t-1$ and $t$. We obtain the primary textual representation $H_i^p = \{h_1, h_2, ..., h_n\} \in R^{n \times d_e}$ of $s_i$ after LSTM layer. To gain the critical suicide-related textual information of $H_i^p$, we apply the **attention mechanism *Att_I*** :

$$Att\_I = softmax(H_i^p W_3 + b_3) \tag{3}$$

where $Att\_I \in R^{n \times 1}$ is the attention vector that demonstrates the distribution of the weights for each word of primary textual representation, $W_3 \in R^{d_e \times 1}$ and $b_3 \in R^{1 \times 1}$ are trainable parameters. Then we make multiplication between the attention vector $Att\_I$ and $H_i^p$ to get the final textual representation $\hat{H}_i \in R^{d_e \times 1}$ of text $s_i$,

$$\hat{H}_i = (H_i^p)^T Att\_I \tag{4}$$

**Image Feature Extraction.** We extract image features from a 34 layer pre-trained ResNet (He et al., 2016). For the convenience of calculation, we convert the last fully connected layer of ResNet from $512 \times 1000$ to $512 \times d_e$:

$$I_i = tanh(O\, W_4 + b_4) \tag{5}$$

where $O \in R^{1 \times 512}$ is the input of the last fc-layer, $W_4 \in R^{512 \times d_e}$ and $b_4 \in R^{1 \times d_e}$ are trainable parameters. Then, $I_i \in R^{1 \times d_e}$ is the visual representation of picture $p_i$.

**User's Feature Extraction.**

| Feature | Dimension | Description |
|---|---|---|
| Gender | 3 | (1,0,0)for male, (0,1,0)for female and (0,0,1) for 'Unknown'. |
| Screen name length | 1 | The length of screen name. |
| Post Count | 1 | The number of posts. |
| Follower | 1 | The number of followers. |
| Following | 1 | The number of following. |
| Picture | 1 | The number of posts with picture. |
| Post time | 4 | The four dimensions are proportions of posts posted in (0:00-5:59), (6:00-11:59), (12:00-17:59), (18:00-23:59) of a day. |

Table 3: Summary of user's features.

As illustrated in table 3, we extract 12 features $F \in R^{12\times1}$ from user' profile and posting behaviour. Since not every one has tree hole's data, in this study we do not consider tree hole's data in suicide risk detection model.

### 4.2 Suicide Risk Detection

Given textual representation $\hat{H}_i$ and visual representation $I_i$, we employ a concatenate operation $\oplus$ to obtain the post representation $E_i \in R^{2d_e\times1}$ for a single post $(s_i, p_i)$:

$$E_i = \hat{H}_i \oplus I_i^T \tag{6}$$

Similar with above, we employ another LSTM layer and attention mechanism to generate global post representation $G \in R^{30\times1}$:

$$\begin{aligned} h_t &= \textbf{LSTM}(E_i, h_{t-1}), \\ Att\_II &= softmax(H^g W_5 + b_5), \\ G &= tanh(((Att\_II)^T \times H^g)W_6 + b_6), \end{aligned} \tag{7}$$

where $h_{t-1}, h_t$ represent the hidden states at time $t-1$ and $t$. $H^g = \{h_1, h_2, ..., h_m\} \in R^{m\times d_e}$ represents the primary post representation of a user after LSTM layer. As not every post of a user expresses the ideation of suicide, we apply the **attention mechanism *Att_II*** to gain the high suicide risk post information of $H^e$. An attention vector $Att\_II \in R^{m\times1}$ was computed to present the different risk weight of posts, where $W_5 \in R^{d_e\times1}$ and $b_5 \in R^{1\times1}$ stand for trainable parameters. Then based on attention $Att\_II$, we obtain the global post representation $G$ for a user, where $W_6 \in R^{d_e\times30}$ and $b_6 \in R^{1\times30}$ stand for trainable parameters.

Finally, we apply a concatenate operation to jointly consider $G$ and $F$, and through a fully connected layer to compute the possibility of suicide:

| | Users | Posts | Posts with image |
|---|---|---|---|
| suicide | 3,652 | 252,901 | 93,461 |
| non-suicide | 3,677 | 491,130 | 260,667 |

Table 4: Statistic of suicide data set.

$$[y_1, y_0] = softmax((G \oplus F)^T W_7 + b_7) \tag{8}$$

where $y_1, y_0$ represent the possibility of a user at risk of suicide or not, $W_7 \in R^{42\times2}$ and $b_7 \in R^{1\times2}$ stand for trainable parameters.

## 5 Experiments

### 5.1 Data Collection

To make suicide risk detection via social media, we construct two data sets: one from Tree Hole and another from Weibo.

**Tree Hole's data set.** We studied a suicidal community which exists in the comments of a Chinese student's last posting before the student committed suicide. In March 17, 2012, this student which screen name is "Zoufan" left the last word on Weibo and then committed suicide. For the past seven years, More than 160,000 people gather here and write over 1,700,000 comments which is still continuing to grow. They express their suicidal thoughts, show their tragic experiences and demonstrate their plans of suicide behaviors. In psychology, we can understand this community as a Tree Hole. We crawled all comments from May 1,2018 to April 30, 2019 and selected top 4,000 active users. After that, four doctoral students major in computational mental healthcare were employed to annotate users that whether they are at risk of suicide or not. Specifically, We only decide the user "at suicide risk" based on self-report of

his/her tree hole posts. If a user express clear suicidal thoughts like *"At this moment, I especially want to die. I feel very tired. I really want to be free."* more than 5 times in different days, then we label him/her at suicide risk. Finally we get 190,087 sentences of 3,652 users and the average length per post is 11.96 words.

**Suicide data set.** To collect users at suicide risk, we crawl user's profile and all created posts in Weibo according to user list from Tree Hole' data set. Besides, we select the users who never submit any post containing expression about suicidal ideation and label them as non-suicide risk. In this case, we discard users whose fans more than 1,500 or posts more than 2,000 because that they may be public figure or organization. The statistic of suicide data set are illustrated in Table 4.

### 5.2 Data Preprocessing

We carry out the following data preprocessing procedures: 1) **Emoji**. We replace emoji with corresponding word like "happy", "cry" to facilitate our model to understand the emotion of user's post. 2) **URL.** As URL has no use for our detection, we simply remove them from sentences. 3) **Image.** All images posted by users were adjusted to $224 \times 224$ for normalized input.

### 5.3 Experimental Setup

In suicide detection task, we treat recent 100 posts from one user as one sample. After sum up $D_2$ and $D_3$, we obtain 7,329 microblog users and training set, validation set and test set contain 6,129, 600, 600 respectively. All sentences are padded to the length of the longest sentence in the data set with word "<PAD>". Batch size is 16 during training process and we use 0.001 as learning rate. Adam Kingma and Ba (2015) is adopted as the optimizer.

We compare suicide-oriented word embeddings with following well-developed word embeddings.

(1) **Word2vec**: The fundamental work for considering the local semantic information of words. We get pre-trained Word2vec word embeddings from Li et al. (2018).

(2) **GloVe**: Context-based unsupervised algorithm, which apply co-occurrence matrix to jointly consider the local and global semantic information. We apply open source tool [2] to train word embeddings on all sentences from Tree Hole's data set.

(3) **Fasttext**: A fast text classification and representation learning model based on Word2vec and hierarchical softmax. Pre-trained FastText word embeddings were obtained from official project [3].

(4) **Bert**: Latest language model based on transformer. We acquire pre-trained Bert model from official project[4] and generate word embedding of each word in a sentence dynamically.

Also, we compare our suicide risk detection model with following well-designed methods.

(1) **LSTM** (Coppersmith et al., 2018): An attention mechanism based Long Short-Term Memory model which can capture contextual information between suicide-related words and others.

(2) **Naive Bayesian (NB) and Support Vector Machine (SVM)** (Pedregosa et al., 2011): Two representative machine learning methods with well-designed features. We use SC-LIWC information (Cheng et al., 2017) as textual features, saturation, brightness, warm/clear color and five-color theme information (Shen et al., 2018) from picture as visual features and user's behaviour features from table 3.

### 5.4 Results

Three sets of tests were conducted to evaluate the performance of suicide risk detection model with suicide-oriented word embeddings.

#### 5.4.1 Effectiveness of Suicide-oriented Word Embeddings

We compare the performance of LSTM and SDM with seven word embeddings as illustrated in table 5. We find that without suicide-related dictionary, Bert outperforms other three word embeddings with 2% higher accuracy and 1.5% higher F1-score on both models. After leveraging suicide-related dictionary, suicide-oriented word embeddings based on FastText achieves the best performance with accuracy 88.00% 91.33%, F1-score 88.14%, 90.92% on two models. Obviously, there is a gap between suicide-oriented word embeddings and normal word embeddings which can verify the effectiveness of the former.

#### 5.4.2 Effectiveness of Suicide Risk Detection Model

We compare the performance of four model as shown in table 6. In this case, LSTM and SDM

[2]https://github.com/stanfordnlp/GloVe

[3]https://github.com/facebookresearch/fastText

[4]https://github.com/google-research/bertpre-trained-models

| | | Word2vec | GloVe | FastText | Bert | So-W2v | So-GloVe | So-FastText |
|---|---|---|---|---|---|---|---|---|
| LSTM | Acc(%) | 79.21 | 80.17 | 82.59 | 85.15 | 86.00 | 86.45 | **88.00** |
| | F1(%) | 78.58 | 79.98 | 82.18 | 85.69 | 86.17 | 86.69 | **88.14** |
| SDM | Acc(%) | 86.54 | 86.55 | 87.08 | 88.89 | 90.83 | 91.00 | **91.33** |
| | F1(%) | 86.63 | 85.13 | 86.91 | 87.44 | 90.55 | 90.56 | **90.92** |

Table 5: Performance comparison for different word embedding and different detection model, where "So-W2v", "So-Glove" and "So-FastText" represent suicide-oriented word embeddings based on Word2vec, GloVe and FastText respectively. Acc and F1 represent accuracy and F1-score.

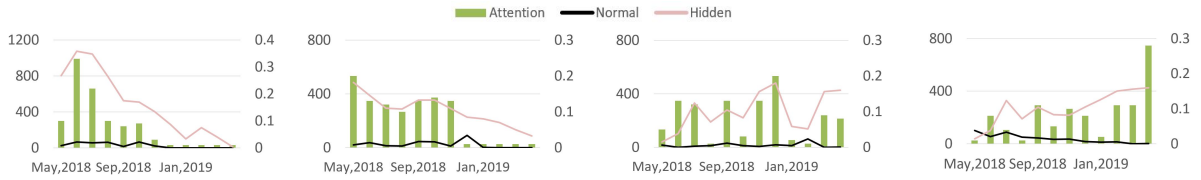


(a) *User 1 with* $\rho_{nh} = 0.84$ *and* $\rho_{ah} = 0.88$ (b) *User 2 with* $\rho_{nh} = 0.34$ *and* $\rho_{ah} = 0.91$ (c) *User 3 with* $\rho_{nh} = -0.12$ *and* $\rho_{ah} = 0.6$ (d) *User 4 with* $\rho_{nh} = -0.7$ *and* $\rho_{ah} = 0.6$

Figure 4: Correlation between normal posts, hidden posts and attention on four representative suicide risk users. The x-axis shows dates from May, 2018 to April, 2019. The left y-axis represent number of suicde-related words/phrases and the right represent attention weight.

| | Full testset | | Harder sub-testset | |
|---|---|---|---|---|
| | Acc | F1 | Acc | F1 |
| SVM | 70.34 | 69.01 | 61.17 | 64.11 |
| NB | 69.59 | 70.12 | 65.14 | 62.20 |
| LSTM | 88.00 | 88.14 | 76.89 | 75.32 |
| SDM | **91.33** | **90.92** | **85.51** | **84.77** |

Table 6: Performance comparison between different suicide risk detection model, where Acc and F1 represent accuracy and F1-score respectively.

| Inputs | Accuracy | F1-score |
|---|---|---|
| Text | 88.56 | 87.99 |
| Text+Image | 89.22 | 89.22 |
| Text+User's feature | 90.66 | 90.17 |
| Text+Image +User's feature | **91.33** | **90.92** |

Table 7: Ablation test for SDM with different inputs.

employs So-FastText word embeddings as their input. SDM improves the accuracy by over 3.33% and obtains 2.78% higher F1-score on full data set.

#### 5.4.3 Performance on *Harder sub-testset*

To verify the effectiveness of models in dealing with peoples implicit and anti-real contrary expressions on microblog posts, we filter out 130 suicide risk people from test set who do not show obvious suicidal ideation on normal posts. Those 130 people construct a subset of test set named *Harder sub-testset*. After observing the performance of four models as shown in table 6, SDM can keep 8% higher value both in accuracy and F1-score on *Harder sub-testset*, compared with other models, and the decline is smaller than other models. This suggests that SDM performs better than existing models in dealing with people's implicit and anti-real contrary expressions.

#### 5.4.4 Ablation Test for Suicide Risk Detection Model

To show the contribution of different input to the final classification performance, we design an ablation test of SDM after removing different input. All SDMs are based on embedding So-Fasttext. Since not every post contains image and user's features contain missing value, we do not only use images nor user's feature as input of SDMs. As illustrated in table 7, we can see that textual information is a crucial input of our SDM. Besides, user's features play a more important role than visual information. The more modalities we use, the better performance we get.

### 5.5 Discovery

To further explore the correlation between normal posts and hidden posts from same user, we import *Pearson Correlation Coefficient* (Tutorials, 2014) to manage it. For each user, we obtain a normal vector $V_i^n = \{v_{i,Jan}^n, v_{i,Feb}^n, ..., v_{i,Dec}^n\} \in$

$R^{12}$, where $v^n_{i,Jan}$ shows the total number of times words/phrases appear in posts for user *i* in January. In a similar way we get a hidden vector $V^h_i = \{v^h_{i,Jan}, v^h_{i,Feb}, ..., v^h_{i,Dec}\} \in R^{12}$. For each normal posts, we also have an attention weight from $Att_{II}$ which represents the suicide risk. Then, similar as above, an attention risk vector $V^a_i = \{v^a_{i,Jan}, v^a_{i,Feb}, ..., v^a_{i,Dec}\} \in R^{12}$ was computed, where $v^a_{i,Jan}$ shows the total suicide risk for user *i* in January. We donate the pearson correlation coefficient $\rho_{n,h,i}$ of $V^n_i$ and $V^h_i$, $\rho_{a,h,i}$ of $V^a_i$ and $V^h_i$ as the correlation between normal posts and hidden posts , attention and hidden posts for user *i*.

As shown in figure 4, we find that there are high positive linear correlation between normal posts and hidden posts from user 1 with $\rho_{nh} = 0.84$ and high negative linear correlation from user 4 with $\rho_{nh} = -0.7$. For other two suicide risk users, there are not obvious linear correlations with $\rho_{nh} = 0.33, -0.12$ respectively. The phenomenon that correlations $\rho_{ah}$ between attention and hidden posts from four users all higher than 0.6 which means high positive linear correlation, which verify the ability of the two-layered attention mechanism to reveal ones' inner emotional world.

## 6 Conclusion

In this paper, we explore the uses of tree holes to enhance microblog-based suicide risk detection. Suicide-oriented word embeddings based on tree hole contents are built to strengthen the sensibility of suicide-related lexicons and a two-layered attention mechanism is deployed to grasp intermittently changing points from individuals open blog streams. Based on above word embeddings and attention mechanism, we propose a suicide risk detection model which outperforms the well-designed approaches on benchmark data set. Through experimental results we also find that, our model also performs well on people's implicit and anti-real contrary expressions.

## Acknowledgments

The work is supported by National Natural Science Foundation of China (61872214, 61532015, 61521002) and Chinese Major State Basic Research Development 973 Program (2015CB352301).

## References

Amayas Abboute, Yasser Boudjeriou, Gilles Entringer, Jérôme Azé, Sandra Bringay, and Pascal Poncelet. 2014. Mining twitter for suicide prevention. In *International Conference on Applications of Natural Language to Data Bases/Information Systems*, pages 250–253. Springer.

Amanuel Alambo, Manas Gaur, Usha Lokala, Ugur Kursuncu, Krishnaprasad Thirunarayan, Amelie Gyrard, Amit Sheth, Randon S Welton, and Jyotishman Pathak. 2019. Question answering for suicide risk assessment using reddit. In *2019 IEEE 13th International Conference on Semantic Computing (ICSC)*, pages 468–473. IEEE.

Courtney Bagge and Augustine Osman. 1998. The suicide probability scale: Norms and factor structure. *Psychological reports*, 83(2):637–638.

Scott R Braithwaite, Christophe Giraud-Carrier, Josh West, Michael D Barnes, and Carl Lee Hanson. 2016. Validating machine learning algorithms for twitter data against established measures of suicidality. *JMIR mental health*, 3(2):e21.

Pete Burnap, Walter Colombo, and Jonathan Scourfield. 2015. Machine classification and analysis of suicide-related communication on twitter. In *Proceedings of the 26th ACM conference on hypertext & social media*, pages 75–84. ACM.

Qijin Cheng, Tim Mh Li, Chi Leung Kwok, Tingshao Zhu, and Paul Sf Yip. 2017. Assessing suicide risk and emotional distress in chinese social media: a text mining and machine learning study. *Journal of Medical Internet Research*, 19(7):e243.

Glen Coppersmith, Ryan Leary, Patrick Crutchley, and Alex Fine. 2018. Natural language processing of social media as screening for suicide risk. *Biomedical informatics insights*, 10:1178222618792860.

Glen Coppersmith, Ryan Leary, Eric Whyne, and Tony Wood. 2015. Quantifying suicidal ideation via language usage on social media. In *Joint Statistics Meetings Proceedings, Statistical Computing Section, JSM*.

John R Crawford and Julie D Henry. 2003. The depression anxiety stress scales (dass): Normative data and latent structure in a large non-clinical sample. *British journal of clinical psychology*, 42(2):111–131.

Munmun De Choudhury, Emre Kiciman, Mark Dredze, Glen Coppersmith, and Mrinal Kumar. 2016. Discovering shifts to suicidal ideation from mental health content in social media. In *Proceedings of the 2016 CHI conference on human factors in computing systems*, pages 2098–2110. ACM.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jingcheng Du, Yaoyun Zhang, Jianhong Luo, Yuxi Jia, Qiang Wei, Cui Tao, and Hua Xu. 2018. Extracting psychiatric stressors for suicide from social media using deep learning. *BMC medical informatics and decision making*, 18(2):43.

Cecilia A Essau. 2005. Frequency and patterns of mental health services utilization among adolescents with anxiety and depressive disorders. *Depression and anxiety*, 22(3):130–137.

King-wa Fu, Ka Y Liu, and Paul SF Yip. 2007. Predictive validity of the chinese version of the adult suicidal ideation questionnaire: Psychometric properties and its short version. *Psychological Assessment*, 19(4):422.

Li Guan, Bibo Hao, Qijin Cheng, Paul SF Yip, and Tingshao Zhu. 2015. Identifying chinese microblog users with high suicide probability using internet-based profile and linguistic features: classification model. *JMIR mental health*, 2(2):e17.

Li Guan, Bibo Hao, and Tingshao Zhu. 2014. How did the suicide act and speak differently online? behavioral and linguistic features of china's suicide microblog users. *arXiv preprint arXiv:1407.0466*.

Keith M Harris and Melissa Ting-Ting Goh. 2017. Is suicide assessment harmful to participants? findings from a randomized controlled trial. *International journal of mental health nursing*, 26(2):181–190.

Keith M Harris, John P McLean, and Jeanie Sheffield. 2014. Suicidal and online: How do online behaviors inform us of this high-risk population? *Death studies*, 38(6):387–394.

Keith M Harris, Jia-Jia Syu, Owen D Lello, YL Eileen Chew, Christopher H Willcox, and Roger HM Ho. 2015. The abcs of suicide risk assessment: Applying a tripartite approach to individual evaluations. *PLoS One*, 10(6):e0127442.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Julie D Henry and John R Crawford. 2005. The short-form version of the depression anxiety stress scales (dass-21): Construct validity and normative data in a large non-clinical sample. *British journal of clinical psychology*, 44(2):227–239.

Xiaolei Huang, Xin Li, Tianli Liu, David Chiu, Tingshao Zhu, and Lei Zhang. 2015. Topic model for identifying suicidal ideation in chinese microblog. In *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation*, pages 553–562.

Xiaolei Huang, Lei Zhang, David Chiu, Tianli Liu, Xin Li, and Tingshao Zhu. 2014. Detecting suicidal ideation in chinese microblogs with psychological lexicons. In *2014 IEEE 11th Intl Conf on Ubiquitous Intelligence and Computing and 2014 IEEE 11th Intl Conf on Autonomic and Trusted Computing and 2014 IEEE 14th Intl Conf on Scalable Computing and Communications and Its Associated Workshops*, pages 844–849. IEEE.

Yen-Pei Huang, Tiong Goh, and Chern Li Liew. 2007. Hunting suicide notes in web 2.0-preliminary findings. In *Ninth IEEE International Symposium on Multimedia Workshops (ISMW 2007)*, pages 517–521. IEEE.

Jared Jashinsky, Scott H Burton, Carl L Hanson, Josh West, Christophe Giraud-Carrier, Michael D Barnes, and Trenton Argyle. 2014. Tracking suicide risk factors through twitter in the us. *Crisis*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. 2016. Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.

Marcel Adam Just, Lisa Pan, Vladimir L Cherkassky, Dana L McMakin, Christine Cha, Matthew K Nock, and David Brent. 2017. Machine learning of neural representations of suicide and emotion concepts identifies suicidal youth. *Nature human behaviour*, 1(12):911.

Diederik Kingma and Jimmy Ba. 2015. ADAM: A method for stochastic optimization. In *Proc. of ICLR*.

Shen Li, Zhe Zhao, Renfen Hu, Wensi Li, Tao Liu, and Xiaoyong Du. 2018. Analogical reasoning on chinese morphological and semantic relations. *arXiv preprint arXiv:1805.06504*.

Tim MH Li, Ben CM Ng, Michael Chau, Paul WC Wong, and Paul SF Yip. 2013. Collective intelligence for suicide surveillance in web forums. In *Pacific-Asia Workshop on Intelligence and Security Informatics*, pages 29–37. Springer.

Meizhen Lv, Ang Li, Tianli Liu, and Tingshao Zhu. 2015. Creating a chinese suicide dictionary for identifying suicide risk on social media. *PeerJ*, 3:e1455.

Naoki Masuda, Issei Kurahashi, and Hiroko Onari. 2013. Suicide ideation of individuals in online social networks. *PloS one*, 8(4):e62262.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Bridianne O'Dea, Stephen Wan, Philip J Batterham, Alison L Calear, Cecile Paris, and Helen Christensen. 2015. Detecting suicidality on twitter. *Internet Interventions*, 2(2):183–188.

World Health Organization et al. 2014. *Preventing suicide: A global imperative*. World Health Organization.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

John Pestian, Henry Nasrallah, Pawel Matykiewicz, Aurora Bennett, and Antoon Leenaars. 2010. Suicide note classification using natural language processing: A content analysis. *Biomedical informatics insights*, 3:BII–S4706.

John P Pestian, Michael Sorter, Brian Connolly, Kevin Bretonnel Cohen, Cheryl McCullumsmith, Jeffry T Gee, Louis-Philippe Morency, Stefan Scherer, Lesley Rohlfs, and STM Research Group. 2017. A machine learning approach to identifying the thought markers of suicidal subjects: a prospective multicenter trial. *Suicide and Life-Threatening Behavior*, 47(1):112–121.

Fuji Ren, Xin Kang, and Changqin Quan. 2015. Examining accumulated emotional traits in suicide blogs with an emotion topic model. *IEEE journal of biomedical and health informatics*, 20(5):1384–1396.

Debra J Rickwood, Frank P Deane, and Coralie J Wilson. 2007. When and how do young people seek professional help for mental health problems? *Medical journal of Australia*, 187(S7):S35–S39.

Ramit Sawhney, Prachi Manchanda, Puneet Mathur, Rajiv Shah, and Raj Singh. 2018. Exploring and learning suicidal ideation connotations on social media with deep learning. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 167–175.

Tiancheng Shen, Jia Jia, Guangyao Shen, Fuli Feng, Xiangnan He, Huanbo Luan, Jie Tang, Thanassis Tiropanis, Tat-Seng Chua, and Wendy Hall. 2018. Cross-domain depression detection via harvesting social media. In *IJCAI*, pages 1611–1617.

Hajime Sueki. 2015. The association of suicide-related twitter use with suicidal behaviour: a cross-sectional study of young internet users in japan. *Journal of affective disorders*, 170:155–160.

SPSS Tutorials. 2014. Pearson correlation. *Retrieved on February*.

M. Johnson Viouls, B. Moulahi, J. Az, and S. Bringay. 2018. Detection of suicide-related posts in twitter data streams. *Ibm Journal of Research Development*, 62(1):7:1–7:12.

Henrik D Zachrisson, Kjetil Rödje, and Arnstein Mykletun. 2006. Utilization of health services in relation to mental health problems in adolescents: a population based survey. *BMC public health*, 6(1):34.

Lei Zhang, Xiaolei Huang, Tianli Liu, Ang Li, Zhenxiang Chen, and Tingshao Zhu. 2014a. Using linguistic features to estimate suicide probability of chinese microblog users. In *International Conference on Human Centered Computing*, pages 549–559. Springer.

Lei Zhang, Xiaolei Huang, Tianli Liu, Ang Li, Zhenxiang Chen, and Tingshao Zhu. 2014b. Using linguistic features to estimate suicide probability of chinese microblog users. In *International Conference on Human Centered Computing*, pages 549–559. Springer.

# Deep Ordinal Regression for Pledge Specificity Prediction

**Shivashankar Subramanian** **Trevor Cohn** **Timothy Baldwin**
School of Computing and Information Systems
The University of Melbourne
shivashankar@student.unimelb.edu.au {t.cohn,tbaldwin}@unimelb.edu.au

## Abstract

Many pledges are made in the course of an election campaign, forming important corpora for political analysis of campaign strategy and governmental accountability. At present, there are no publicly available annotated datasets of pledges, and most political analyses rely on manual analysis. In this paper we collate a novel dataset of manifestos from eleven Australian federal election cycles, with over 12,000 sentences annotated with specificity (e.g., rhetorical vs. detailed pledge) on a fine-grained scale. We propose deep ordinal regression approaches for specificity prediction, under both supervised and semi-supervised settings, and provide empirical results demonstrating the effectiveness of the proposed techniques over several baseline approaches. We analyze the utility of pledge specificity modeling across a spectrum of policy issues in performing ideology prediction, and further provide qualitative analysis in terms of capturing party-specific issue salience across election cycles.

## 1 Introduction

Election manifestos play a critical role in structuring political campaigns. Campaign communication can influence a party's reputation, credibility, and competence, which are primary factors in voter decision making (Fernandez-Vazquez, 2014). Among the various campaign-related functions fulfilled by manifestos (Eder et al., 2017), perhaps the most important is the contract they represent between parties and voters in terms of pledges and prioritisation of political issues (Royed et al., 2019). Political scientists have long studied how specific pledges translate into government programs and actual policy (Royed, 1996; Thomson, 2001; Naurin, 2011; Schermann and Ennser-Jedenastik, 2014). Other work relates specific pledges to the issue clarity of a political party through selective emphasis, which complements salience theory (Robertson et al., 1976; Budge and Farlie, 1983; Praprotnik, 2017). For example:

> *we commit ... 30 per cent tax rebate or cash benefit on the cost of private health insurance premiums*

conveys the party's support for private health insurance, and is more verifiable than:

> *we will improve the health system.*

Issue clarity has also been shown to be influenced by a party's ideological position and its role in government (Praprotnik, 2017).

Although pledge specificity prediction is an important task for the analysis of party position, priorities, and post-election policy framing, to date, almost all research has relied on manual analysis. Subramanian et al. (2019) is a recent exception to this, in performing speech act classification over political campaign text, where the class schema includes the distinction between specific and vague pledges (binary specificity class).

In this paper, we perform fine-grained pledge specificity prediction, which is more expressive than binary levels (Li et al., 2016; Gao et al., 2019). We use a class schema proposed by Pomper and Lederman (1980) as detailed in Table 1, which captures seven levels of specificity, forming a non-linear increasing order of commitment and specificity (Pomper and Lederman, 1980). Given the non-linear nature of the scale, we use deep ordinal regression models for this task, with distributional loss (Imani and White, 2018), where we model the output as a uni-modal distribution (Beckham and Pal, 2017). Our goal is to capture the intuition that a pledge with specificity level $k$, has higher commitment than all the levels $< k$, producing a smoothly varying prediction over the ordinal classes. This can be modeled as a uni-modal



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1729–1740, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

distribution which has a probability mass that decreases on both sides of the most probable class. Lastly, as it is expensive to obtain large-scale annotations, in addition to developing a novel annotated dataset, we also experiment with a semi-supervised approach by using unlabeled text.

The contributions of this paper are as follows: (1) we develop and release a dataset[1] for fine-grained pledge specificity prediction based on election manifestos covering eleven Australian federal election cycles (1980–2016), from the two major political parties — Labor and Liberal; (2) we propose to use deep ordinal regression models for the prediction task, and evaluate the model under sparse supervision scenarios using the teacher–student framework; and (3) we evaluate the utility of pledge specificity towards ideology prediction, and provide further qualitative analysis by correlating model predictions with party-specific issue salience across major policy areas.

## 2 Related Work

Political manifesto text analysis is a relatively novel application, at the intersection of Political Science and NLP. Research has focused primarily on fine-grained policy topic classification and overall ideology prediction tasks (Volkens et al., 2017; Verberne et al., 2014; Zirn et al., 2016; Subramanian et al., 2018). Most work dealing with pledge specificity analysis in manifestos has been based on manual analysis, as outlined in Section 1.

Specificity is a pragmatic property of text which has been studied across various fields of research. In cognitive linguistics, Dixon (1987) showed that specificity of information in text impacts reading comprehension speed. In Political Science, it has been used to analyze salience, party position and post-election policy framing (see Section 1). There has also been research on the association between text specificity and communication style. In terms of automated specificity analysis, Cook (2016) found specificity in the context of congressional hearings to vary between speakers belonging to the same vs. different ideologies. Namely, it was shown that specificity increases as the ideological distance between the committee chair and the witness decreases. Subramanian et al. (2019) addressed two levels of pledge specificity, as part of speech act classification task. Specificity has also been studied in news (Louis and Nenkova, 2011) and classroom discussion domains (Luo and Litman, 2016; Lugini and Litman, 2017).

These studies have dealt with a restrictive coarse-level analysis (2–3 categories), whereas a fine-grained scale better captures and allows for comparison of election manifestos (Pomper and Lederman, 1980). Gao et al. (2019) was the first attempt at fine-grained text specificity prediction, in the context of social media posts. Here, we target the novel task of fine-grained pledge specificity prediction, which can be used in a range of downstream applications, including capturing party priorities (salience) and ideological position across election cycles.

All the text specificity analysis work in NLP has modeled the task as classification or regression. As the 7-step pledge specificity levels used in this research (Pomper and Lederman, 1980) do not form a single real-valued scale, we model it as an ordinal regression task. Some examples of ordinal regression tasks include sentiment rating prediction (Rosenthal et al., 2017), stages of disease prediction (Gentry et al., 2015), and age prediction (Eidinger et al., 2014). Recent work has shown that adding a distributional (auxiliary) loss alongside a regression loss, and using expectation to obtain the predicted value (Imani and White, 2018), provides label smoothing and improves regression performance (Gao et al., 2017). Approaches based on a uni-modal probability distribution (e.g., Poisson) as output (da Costa et al., 2008; Beckham and Pal, 2017) can be seen as related to the former approach (Imani and White, 2018) where the discrete probability mass function replaces the histogram density. We propose to use a uni-modal distributional loss-based ordinal regression for pledge specificity prediction.

Secondly, as it is difficult to obtain large amounts of labeled data, existing approaches have used semi-supervised learning (Li and Nenkova, 2015; Subramanian et al., 2019). Here we use a cross-view training approach (Clark et al., 2018; Subramanian et al., 2019), where we enforce consensus between the intermediate class distributions or the final real-valued output.

## 3 Pledge Specificity Dataset

We annotated 22 election manifestos from the Australian Labor and Liberal parties, covering eleven Australian federal election cycles from

[1] https://github.com/shivashankarrs/Pledge-Specificity

| Category | Definition | Example | # |
|---|---|---|---|
| Not a pledge | Provide facts; greetings; approval or criticism of policies | *We have modernised Australia's industrial relations system, particularly through the 1996 Workplace Relations Act* | 1 |
| Rhetorical pledge | Based on moral values and applies to all irrespective of the party | *We will put our country first* | 2 |
| General pledge | Specify intangible goals, and also not the ways to achieve them | *Labor will build a stronger and more productive economy* | 3 |
| Continuity pledge | Commit to the maintenance of currently functioning policy | *We will retain the voluntary health insurance system which now covers the great majority of Australians* | 4 |
| Goal pledge | Provide tangible outcomes and goals, without providing the means to achieve them | *A Shorten Labor Government will create 2000 jobs in Adelaide* | 5 |
| Action pledge | Provide means to achieve the objective, but don't reveal specific details | *We pledge to support effective voluntary family planning, and to recognize officially the link between social and economic development and the willingness of the individual to limit family size* | 6 |
| Detailed pledge | Provide clear details of action to achieve an objective | *A re-elected Coalition Government will invest $1 million to support the Exeter Community Precinct* | 7 |

Table 1: Pledge specificity category, definition, example manifesto sentence, and the corresponding specificity value (#).

| Specificity | # Sentences | % | Avg. Length |
|---|---|---|---|
| 1 | 8165 | 67.00 | 19.5 |
| 2 | 423 | 3.47 | 22.0 |
| 3 | 950 | 7.80 | 23.4 |
| 4 | 300 | 2.46 | 24.6 |
| 5 | 473 | 3.88 | 24.8 |
| 6 | 901 | 7.39 | 25.2 |
| 7 | 973 | 7.99 | 28.5 |

Table 2: Distribution and length statistics across specificity categories.

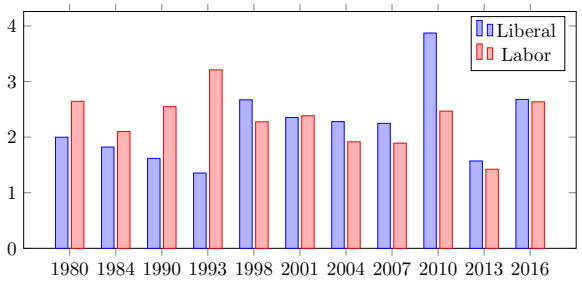


Figure 1: Average pledge specificity of Labor and Liberal parties over different election cycles

1980–2016. The dataset has 12,185 sentences annotated with seven levels of specificity (Pomper and Lederman, 1980). See Table 1 for class definitions and an example of each class. We obtained annotations using the Figure Eight crowdsourcing platform. For each sentence we provided the previous two sentences from the manifesto (as context), the party which published the manifesto, election year, and incumbent and opposition party details. Each sentence was annotated by at least 3 workers after passing quality control (at least 70% accuracy on test questions). After obtaining annotations, the label which has the highest confidence score is chosen for each sentence. Confidence is the level of agreement between multiple contributors, weighted by the contributors' trust scores. Overall agreement based on the Krippendorf's Alpha is $\alpha = 0.58$, indicating moderate agreement (Artstein and Poesio, 2008; Krippendorff, 2011), on par with related studies (Gao et al., 2019). The class distribution in the final dataset is given in Table 2, alongside the average sentence length in tokens. It can be seen that more specific pledge categories have higher average length. Average specificity values of Labor and Liberal party manifestos across elections are given in Figure 1. The length of the manifesto (in terms of number of sentences) influences average specificity values, with exceptions such as the Liberal party's 2010 election manifesto which is the shortest document but has the highest average pledge specificity value. Further detailed analysis, to decipher the pledge specificity trends in general, is a potential task for future work.

## 4 Proposed Approach

### 4.1 Base Model

We first obtain representations for each sentence via a sequence of word embeddings, to which we apply a bidirectional GRU ("biGRU": Cho et al. (2014)), and concatenate the final hidden state of both the forward and backward GRUs, $\mathbf{h}_i = \left[\overrightarrow{\mathbf{h}}_i, \overleftarrow{\mathbf{h}}_i\right]$. Rather than using a linear activation layer for the output, we study the effect of learning a distribution over ordinal classes, and using an expectation layer to get the final prediction, which we now expound upon.

### 4.2 Distributional Loss

Let us assume that the continuous target variable $Y$ is normally distributed, conditioned on inputs $x \in \mathbb{R}^d$, $Y \sim \mathcal{N}(\mu = f(x), \sigma^2)$ for a fixed variance $\sigma^2 > 0$. In regression, the maximum likelihood function $f$ for $n$ samples $\{x_i, y_i\}$ corresponds to minimizing $l_2$ loss, such that $f(x) = \mathbb{E}(Y|x)$. Alternatively, we can learn a categorical distribution ($q_x$) over the ordinal classes $\mathcal{Y}$, and use the expected value as the prediction, $f(x)$ (Rothe et al., 2018). In this work, we follow the latter method, but parameterise the categorical distribution based on uni-modal probability distribution, a technique which has been shown to perform well for ordinal regression tasks (Beckham and Pal, 2017). This modification converts the problem to a more difficult (multi-task) problem, that promotes generalization and reduces over-fitting (Imani and White, 2018). The overall objective is to jointly minimize the squared loss for the regression task ($\mathcal{L}_S$), and cross-entropy for the distributional loss over $\mathcal{Y}$ ($\mathcal{L}_D$), based on the objective $\mathcal{L}_J = \alpha\mathcal{L}_S + \mathcal{L}_D$, where the hyper-parameter $\alpha$ is tuned using a validation set. We experiment with different distributions in generating the intermediate representations $q_x$, including categorical (as a baseline approach, see Section 5: Beckham and Pal (2016); Gao et al. (2017); Rothe et al. (2018)), discrete uni-modal (Binomial and Poisson: Beckham and Pal (2017)), and truncated Gaussian (Imani and White, 2018). The final prediction is obtained using expectation, which has been shown to be effective for various regression tasks in the vision domain. Here we study the use of uni-modal distributional loss-based ordinal regression approaches (Beckham and Pal, 2017; Imani and White, 2018) for text specificity analysis (Section 5 has results demonstrating its superiority over the other choices). We detail the different ways to obtain $q_x$, and the corresponding loss functions $\mathcal{L}_D$ below, and provide an overall summary in Figure 2.

#### 4.2.1 BINOMIAL

With the biGRU model, we estimate the parameter ($p$) of the Binomial distribution (with a sigmoid output), based on which the distribution over classes can be obtained via the probability mass function,

$$p(k; K-1, p) = \binom{K-1}{k} p^k (1-p)^{K-1-k},$$

$k \in \{0, 1, \ldots K-1\}$, where $p \in [0, 1]$. As the final layers (post sigmoid) are under-parametrized, we have a softmax layer with $\tau$ after obtaining the probability masses,

$$q_k = \frac{\exp(\phi_k/\tau)}{\sum_{j=0}^{K-1} \exp(\phi_j/\tau)},$$

where $\tau \sim \text{SoftPlus}(\tau')$, and $\tau'$ is learned by the deep net, conditioned on the input ($x$). We then have an expectation layer to obtain the final output $f(x)$. Output of the softmax layer is fit to the one-hot encoded ordinal classes for each input ($\mathbf{y}$), by minimizing the cross-entropy loss ($\mathcal{L}_{\mathcal{D}\text{BINOMIAL}}$).

#### 4.2.2 POISSON

POISSON is similar to the binomial case, in that we obtain the parameter ($\lambda$) of the Poisson distribution using the biGRU, with a SoftPlus activation. We then use the probability mass function of the Poisson distribution to get the probabilities over different classes, $k \in \mathcal{Y}$,

$$p(k; \lambda) = \frac{\lambda^k e^{-\lambda}}{k!},$$

which is again passed through a softmax layer to obtain $q_x$, fit by minimizing cross-entropy loss ($\mathcal{L}_{\mathcal{D}\text{POISSON}}$), and an expectation layer is used to obtain the final prediction.

#### 4.2.3 Gaussian (GAUSS)

To compute $\mathbb{E}(Y|x)$ ($\mu$ of the Gaussian), here we fit the intermediate distribution $q_x$ directly to histogram density of a truncated Gaussian distribution with support $[1, K]$ (target distribution: $p_*$). We achieve this by learning a prediction distribution with the biGRU model, $q_x : \mathcal{Y} \rightarrow [0, 1]$. For this, the ordinal label of training instances is transformed into a truncated Gaussian PDF. The mean

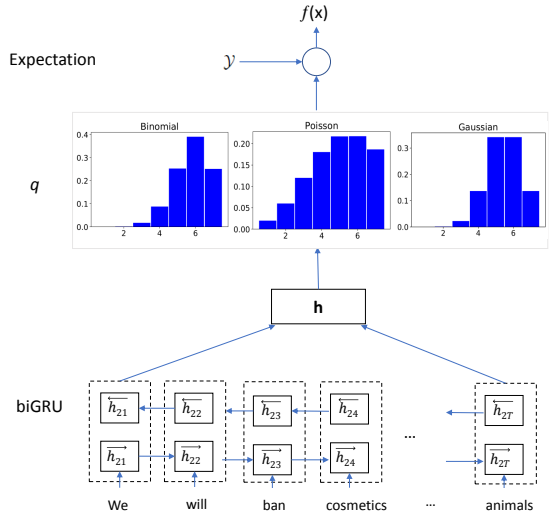


Figure 2: Illustration of the model architecture, comprising a biGRU over sentence tokens, to compute the parameter of one of the three distributions: $p$ for Binomial (with $n = K-1$) and $\lambda$ for Poisson. *Pmf* of these distributions are then used to define a categorical distribution over the ordinal classes $\mathcal{Y}$. For learning, we use the categorical cross-entropy against the gold *one-hot* $y$, as well as the squared error $(y - f(x))^2$, where $f(x)$ is given by expectation taken under the predicted categorical distribution $q$. Gaussian uses a different mechanism, as described in Section 4.2.3 where the categorical distribution ($q$) is predicted directly using a $K$-dimensional softmax output, and the cross-entropy is computed between $q$ and a Gaussian histogram density centred at $\mu = y$, discretised by way of integration of the PDF between adjacent label indices.

$\mu$ for this Gaussian is the target $y$ of each datapoint, with fixed variance $\sigma^2$, which we set to the radius of the bins in $\mathcal{Y}$ (1 in this case). The CDF for the chosen target distribution is computed as $\frac{1}{2}(1 + \text{erf}(\frac{x-\mu}{\sigma\sqrt{2}}))$ and $p_*$ is obtained for each class, k $\in \mathcal{Y}$, as,

$$\frac{1}{2}\left(\text{erf}\left(\frac{k-\mu}{\sigma\sqrt{2}}\right) - \text{erf}\left(\frac{k-1-\mu}{\sigma\sqrt{2}}\right)\right).$$

This formulation allows efficient computation of divergence between $p_*$ and $q_x$ for optimization, which results in cross-entropy minimization ($\mathcal{L}_{\mathcal{D}\text{GAUSS}}$: Imani and White (2018)). Note that the training target $p_*$ is uni-modal, and no constraints are *explicitly* enforced on the shape of $q_x$.

### 4.3 Incorporating Context

We incorporate context in the form of information from adjacent sentences following the approach of Liu et al. (2017): for each training sentence, we use the predicted (intermediate) probability distribution across ordinal classes of the previous $L$ sentences as context. A new biGRU model is trained with the sentence and the additional contextual information, concatenated to $\mathbf{h}_i$. We refer to this model as biGRU$_{\text{ORD+CONTEXT}}$. In the test phase, biGRU$_{\text{ORD}}$ provides contextual information, and the newly trained model (biGRU$_{\text{ORD+CONTEXT}}$) is used to predict the test sentence output.

### 4.4 Semi-supervised Learning

As it is expensive to get large-scale specificity annotations we employ a cross-view training approach (Clark et al., 2018; Subramanian et al., 2019) for semi-supervised learning, which can leverage additional unlabeled text. Cross-view training is a kind of teacher–student method, whereby the model "teaches" a "student" model to classify unlabelled data. The student has a restricted view over the data, e.g., through the application of noise (Sajjadi et al., 2016; Wei et al., 2018). We use biGRU$_{\text{ORD+CONTEXT}}$ with word-level dropout and zero vector set to contextual information as the auxiliary model. This procedure regularizes the learning of the teacher to be more robust, as well as increasing its exposure to unlabeled text. We augment our dataset with over 32k sentences from UK and US election manifestos released from the same time period. On these unlabeled examples, the model's output is used to fit the auxiliary model by enforcing consensus in their predictions. This consensus loss $\mathcal{L}_{\text{U}}$ is added to the supervised training objective ($\mathcal{L}_J$). Under the semi-supervised setting, we evaluate the following approaches:

- MSE: use the final regression output of the teacher model ($f(x)$) to fit an auxiliary model, thereby enforcing consensus using a squared loss, MSE($\mathbb{E}_{q_\theta}[\mathcal{Y}], \mathbb{E}_{q_\omega}[\mathcal{Y}]$) where $\mathcal{Y}$ is a fixed class vector; denoted as "$\mathcal{L}_{\text{UMSE}}$".
- KLD: an intermediate distribution over targets $q_\theta(\mathcal{Y}|s)$ is used to fit an auxiliary model, $q_\omega(\mathcal{Y}|s)$, by minimising the Kullback-Leibler (KL) divergence, KL($q_\theta(\mathcal{Y}|s), q_\omega(\mathcal{Y}|s)$); denoted as "$\mathcal{L}_{\text{UKLD}}$".[2]
- EMD: $q_\theta(\mathcal{Y}|s)$ is again used to fit the auxiliary model, $q_\omega(\mathcal{Y}|s)$, by minimising the earth

[2]This is the closest setting to Subramanian et al. (2019), which minimizes KL divergence between output distributions in a classification setting. But the overall objective is different in our case, in that we have an expectation layer over $q$ to obtain the target regression output.

| Category | Approach | MMAE | $\rho$ |
|---|---|---|---|
| | Majority | 3 | - |
| | Length | - | 0.21 |
| | Speciteller | - | 0.18 |
| | $\text{NN}_{\text{REG}}$ | 2.05 | 0.33 |
| | $\text{biGRU}_{\text{REG}}$ | 1.83 | 0.47 |
| | $\text{biGRU}_{\text{CLASS}}$ | 2.17 | 0.40 |
| | $\text{biGRU}_{\text{REG}_{l_1}}$ | 1.99 | 0.46 |
| Ordinal | $\text{biGRU}_{\text{CATEGORICAL}}$ | 1.80 | 0.48 |
| Ordinal | $\text{biGRU}_{\text{BINOMIAL}}$ | 1.78 | 0.48 |
| Ordinal | $\text{biGRU}_{\text{POISSON}}$ | 1.90 | 0.41 |
| Ordinal | $\text{biGRU}_{\text{GAUSS}}$ | 1.72 | **0.49** |
| Ordinal | $\text{biGRU}_{\text{GAUSS + CONTEXT}}$ | **1.71** | **0.49** |

Table 3: Specificity prediction performance; the best approach is given in bold.

mover's distance, $\text{EMD}(q_\theta(\mathcal{Y}|s),\ q_\omega(\mathcal{Y}|s))$; denoted as "$\mathcal{L}_{\text{UEMD}}$". EMD is defined as $\text{EMD}(q_\theta, q_\omega) = \frac{1}{K}^{\frac{1}{l}} \|\text{cmf}(q_\theta) - \text{cmf}(q_\omega)\|_l$, where $\text{cmf}(\cdot)$ is the cumulative mass function for the predicted (intermediate) probability distribution $q$, and we use $l = 2$. EMD considers distance between classes, and is more suitable for ordinal tasks (Hou et al., 2016).

## 5 Experimental Results

To evaluate model performance we use macro-averaged mean absolute error (MMAE: Rosenthal et al. (2017)) given the class imbalance, and Spearman's $\rho$. MMAE is given as $\frac{1}{|K|}\sum_{j=1}^{|K|}\frac{1}{|S_j|}\sum_{x_i \in S_j}|f(x_i) - y_i|$, where $S_j$ denotes the subset of instances annotated with (true) ordinal class $j$. We consider the following baselines:

**Majority**: assign the majority class in the training set to all test instances.

**Length**: use sentence length as the specificity score.

**Speciteller**: co-training model of Li and Nenkova (2015), used by Cook (2016) for congressional hearings specificity analysis.

**$\text{NN}_{\text{REG}}$**: bag-of-words term-frequency representation, fed into a feed-forward neural network model (Gao et al., 2019).

**$\text{biGRU}_{\text{REG}}$**: biGRU model trained with a mean squared loss objective.

**$\text{biGRU}_{\text{CLASS}}$**: biGRU model trained with a cross-entropy objective (Subramanian et al., 2019).

**$\text{biGRU}_{\text{REG}_{l_1}}$**: biGRU regression model with mean absolute error objective ($l_1$ loss).

All the baseline and proposed biGRU models use ELMo embeddings (Peters et al., 2018). The regression models minimize $l_2$ loss, unless otherwise specified. We compare the average performance across five runs with an 80:20 train:test split. We randomly choose 10% of instances from the training set as validation data. We compare the baseline approaches with our proposed ordinal approaches, which have an intermediate distributional loss in conjunction with the final prediction loss: $\text{biGRU}_{\text{GAUSS}}$ (Section 4.2.3), $\text{biGRU}_{\text{BINOMIAL}}$ (Section 4.2.1), or $\text{biGRU}_{\text{POISSON}}$ (Section 4.2.2). We also evaluate $\text{biGRU}_{\text{CATEGORICAL}}$, where the softmax layer is fitted to one-hot encoded class labels (Gao et al., 2017; Rothe et al., 2018). Note that this is not uni-modal.

Gao et al. (2019) used a combination of bag-of-words representation, surface features, social-media-specific features (eg., Tweet mentions), and emotion-related features, with a support vector regression model which minimizes squared loss. Social media and emotion-related attributes are not relevant to our data, and other surface features did not provide improvements. Hence we show the performance of the bag-of-words representation with squared loss objective ($\text{NN}_{\text{REG}}$ in Table 3). From the results in Table 3, we can see that sequential models with ELMo embeddings (biGRU) perform better than neural bag-of-words models ($\text{NN}_{\text{REG}}$). The $l_2$ regression model ($\text{biGRU}_{\text{REG}}$) performs better than $l_1$ regression ($\text{biGRU}_{\text{REG}_{l_1}}$) and classification ($\text{biGRU}_{\text{CLASS}}$).

With respect to deep ordinal approaches, $\text{biGRU}_{\text{POISSON}}$ performs better than classification ($\text{biGRU}_{\text{CLASS}}$), but does not improve upon regression ($\text{biGRU}_{\text{REG}}$). The Binomial performs better than the Poisson, consistent with previous work (Beckham and Pal, 2017; da Costa et al., 2008). $\text{biGRU}_{\text{CATEGORICAL}}$ performs better than $\text{biGRU}_{\text{REG}}$, but not over unimodal approaches ($\text{biGRU}_{\text{BINOMIAL}}$, $\text{biGRU}_{\text{GAUSS}}$). Overall, the model which fits intermediate distribution to a truncated Gaussian (histogram density) target distribution provides the best performance. It gives over 6% improvement in terms of MMAE, and over 4% in $\rho$, compared to $\text{biGRU}_{\text{REG}}$. Adding context to $\text{biGRU}_{\text{GAUSS}}$ ($\text{biGRU}_{\text{GAUSS+CONTEXT}}$) provides a slight reduction in error.

### 5.1 Semi-supervised Learning

We next compare the performance of $\text{biGRU}_{\text{GAUSS+CONTEXT}}$ (SUP) and the semi-

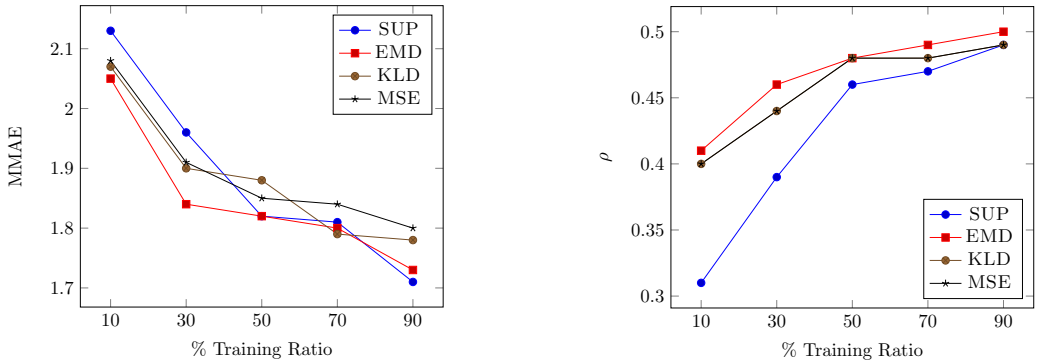


Figure 3: Prediction performance across different training ratios. Note that 90% = all the training data, as 10% is used for validation. The supervised ordinal model (SUP) and semi-supervised teacher–student models (MSE, KLD, EMD, given in Section 4.4) are compared on MMAE and Spearman's $\rho$.

supervised extensions of it (Section 4.4) which leverage additional unlabeled data: minimizing $\mathcal{L}_{\text{UMSE}}$ (MSE), $\mathcal{L}_{\text{UKLD}}$ (KLD), and $\mathcal{L}_{\text{UEMD}}$ (EMD). The amount of labeled data in the training split is varied from 10% to 90%. Results are presented in Figure 3 for MMAE and $\rho$. From the results, semi-supervised approaches provide large gains in terms of both MMAE and $\rho$, especially when training with fewer instances, ratio $\leq$ 30%. Overall, the semi-supervised learning approach which minimizes EMD performs best across all training ratios compared to both supervised and other semi-supervised approaches. It provides .10 and .06 absolute improvements in $\rho$ under sparse supervision scenarios (10% and 30% of training data, resp.). Even under richer supervision settings ($\geq$ 70 %), it provides higher $\rho$.

## 6 Political Analysis Using the Models

Political scientists utilize pledge specificity for a variety of applications (see Section 1). Here, we extrinsically evaluate our specificity model using two tasks related to campaign strategy: (1) party position or ideology prediction (Section 6.1), and (2) issue salience analysis (Section 6.2). For both tasks, we compare the use of pledge specificity across policy issues vs. a count-based representation of policy mentions.

### 6.1 Ideology Prediction

Estimating the manifesto-level ideology score on the left–right spectrum using sentence-level policy topic annotations is a popular task (Slapin and Proksch, 2008; Lowe et al., 2011; Däubler and Benoit, 2017; Subramanian et al., 2018), for which the policy scheme provided by CMP (Volkens et al., 2017) is commonly used. It has 57 political themes, across 7 major categories. Among those approaches, the RILE index is the most widely adopted (Merz et al., 2016; Jou and Dalton, 2017), and has been shown to correlate highly with other popular scores (Lowe et al., 2011). RILE is defined as the difference between count of (pre-determined) right and left policy theme mentions across sentences in a manifesto (Volkens et al., 2013). Here we evaluate the effectiveness of using the proposed specificity modeling across those policy issues, compared to using RILE-based party position scores (Volkens et al., 2013).

We compute the specificity weight (Pomper and Lederman, 1980) from the average specificity score across sentences, $\frac{1}{|I|}\sum_{S_i \in I} \text{Spec}(S_i)$ for each policy issue ($I$). With specificity weight as the basic feature, we also model global signals such as party coalition and temporal dependencies across elections, which can enforce smoothness in manifesto positions (Greene, 2016; Subramanian et al., 2018) based on probabilistic soft logic.

#### 6.1.1 Probabilistic Soft Logic

To address this, we propose an approach using hinge-loss Markov random fields ("HL-MRFs"), a scalable class of continuous, conditional graphical models (Bach et al., 2013). These models can be specified using Probabilistic Soft Logic ("PSL": Bach et al. (2017)), a weighted first order logical template language. An example of a PSL rule is $\lambda : \texttt{P(a)} \wedge \texttt{Q(a,b)} \rightarrow \texttt{R(b)}$, where P, Q, and R are predicates, a and b are variables, and $\lambda$ is the weight associated with the rule. PSL uses soft

truth values for predicates in the interval $[0, 1]$. The degree of ground rule satisfaction is determined using the Lukasiewicz $t$-norm and its corresponding co-norm as the relaxation of the logical AND and OR, respectively. The weight of the rule indicates its importance in the HL-MRF probabilistic model, which defines a probability density function of the form:

$$P(\mathbf{Y}|\mathbf{X}) \propto \exp\left(-\sum_{r=1}^{M} \lambda_r \phi_r(\mathbf{Y}, \mathbf{X})\right),$$

where $\phi_r(\mathbf{Y}, \mathbf{X}) = \max\{l_r(\mathbf{Y}, \mathbf{X}), 0\}^{\rho_r}$ is a hinge-loss potential corresponding to an instantiation of a rule, and is specified by a linear function $l_r$ and optional exponent $\rho_r \in \{1, 2\}$.

#### 6.1.2 PSL Model

Here we elaborate on our PSL model based on manifesto content-based features (specificity weight across 57 policy issues), coalition information, and temporal dependencies. Our target `pos` (left–right position) is a continuous variable $[0, 1]$, where 1 indicates an extreme right position, 0 denotes an extreme left position, and 0.5 indicates center. We also model the social and economic positions explicitly (`socpos` and `econpos`), which influence the overall `pos`. Each instance of a manifesto, its party affiliation and policy issues, are denoted by the predicates `Manifesto`, `Party` and `Policy`. Other predicates are given as follows:

- **Specificity weight** of each policy issue in the given manifesto (`Specw`).
- **Relative specificity scale**: ratio of specificity weight for each policy issue given a party's manifesto, to maximum specificity weight for the same policy issue across parties from the same country and election (`SpecScale`).
- **Policy issue mapping**: 26 out of the 57 policy themes are categorized as social and economic left–right issues by Benoit and Laver (2007) (`IdeologyMap`).
- **Coalition**: captures the strength of ties between two parties, given by a logistic transformation of the number of times two parties have been in a coalition in the past (`Coalition`).
- **Temporal dependency** between a party's current manifesto position and its previous manifesto position (`PreviousManifesto`).

Representative rules of our PSL model, based on the predicates presented above, are given in Table 4. They include:

- **Specificity**: if a manifesto contains more specific pledges related to social (or economic) left/right policies, then it will more likely be a social (or economic) left/right-aligned manifesto.
- **Overall position**: social and economic position influences the overall position, and this allows the model to place different weights on the influence of social and economic policies on the overall position, which is found to be necessary by Benoit and Laver (2007).
- **Global signals**: coalition and temporal dependencies to enforce smoothness in manifesto positions.
- **Relative specificity**: `SpecScale` of a left (or right) policy during an election amplifies its overall position scores.

#### 6.1.3 Evaluation

We use manifestos from Australia and UK for our analysis. We use data from Voter Survey (Cameron and McAllister, 2019) for Australia and CHES Expert Survey (Bakker et al., 2015) for the UK as the gold-standard party position. A primary step (related to the model given in Section 6.1.2) is to obtain policy topic classification for sentences in each manifesto. If annotations are not available from Volkens et al. (2017), one out of 57 political themes are predicted using the method of Subramanian et al. (2018). Specificity scores of sentences are obtained using the proposed ordinal regression approach (biGRU$_{\text{GAUSS+CONTEXT}}$). Using social, economic and a combined list of left–right policy themes (`IdeologyMap`), and with the RILE formulation, we bootstrap `socpos`, `econpos` and `pos`. We then use the PSL model (Table 4) to recalibrate the scores based on specificity scores and the global signals.

We compare the performance of bootstrapped `pos` (RILE or policy count-based) with the PSL model. Principal component analysis ("PCA": Gabel and Huber (2000)) on the frequency distribution, and projection on its principal component, is used as an additional baseline. Spearman's correlation ($\rho$) against the gold-standard positions is given in Table 5. Overall, pledge specificity, especially on a relative scale (which differentiates emphasis between parties) provides large gains, and global signals give only mild improvements.

| **Specificity Weight — Model I** |
|---|
| $\texttt{Manifesto(x)} \wedge \texttt{Policy(I)} \wedge \texttt{Specw(x, I)} \wedge \texttt{IdeologyMap(I, social left/right)} \rightarrow \texttt{socpos(x)}$<br>$\texttt{Manifesto(x)} \wedge \texttt{Policy(I)} \wedge \texttt{Specw(x, I)} \wedge \texttt{IdeologyMap(I, economic left/right)} \rightarrow \texttt{econpos(x)}$ |
| **Overall position — Model II** |
| $\texttt{Manifesto(x)} \wedge \texttt{socpos(x)} \rightarrow \texttt{pos(x)}$<br>$\texttt{Manifesto(x)} \wedge \texttt{econpos(x)} \rightarrow \texttt{pos(x)}$ |
| **Global signals — Model III** |
| $\texttt{Manifesto(x)} \wedge \texttt{Party(x, a)} \wedge \texttt{Manifesto(y)} \wedge \texttt{Party(y, b)} \wedge \texttt{Coalition(a, b)} \wedge \texttt{pos(x)} \rightarrow \texttt{pos(y)}$<br>$\texttt{Manifesto(x)} \wedge \texttt{Party(x, a)} \wedge \texttt{PreviousManifesto(x, t)} \wedge \texttt{Party(t, a)} \wedge \texttt{pos(t)} \rightarrow \texttt{pos(x)}$ |
| **Relative specificity — Model IV** |
| $\texttt{Manifesto(x)} \wedge \texttt{Policy(I)} \wedge \texttt{SpecScale(x, I)} \wedge \texttt{IdeologyMap(I, social left/right)} \rightarrow \texttt{socpos(x)}$<br>$\texttt{Manifesto(x)} \wedge \texttt{Policy(I)} \wedge \texttt{SpecScale(x, I)} \wedge \texttt{IdeologyMap(I, economic left/right)} \rightarrow \texttt{econpos(x)}$ |

Table 4: PSL Model: Representative rules. `left/right` in the `IdeologyMap` predicate indicates policy issues mapped to left/right categories, which is implemented as two separate rules — one for left and another for right.

| Model | Aus | UK |
|---|---|---|
| bootstrapped pos | 0.66 | 0.39 |
| PCA | 0.11 | 0.39 |
| Model I + II | 0.63 | 0.33 |
| Model I + II + III | 0.65 | 0.33 |
| Model I + II + III + IV | **0.71** | **0.45** |

Table 5: Spearman's $\rho$ for prediction of party position based on the different models.

| Policy Area | **C** | **S** |
|---|---|---|
| Health | 824.92 | 905.19 |
| Education | 781.15 | 905.54 |
| Environment | 841.89 | 897.53 |
| Tax | 735.97 | 824.69 |
| Economy | 258.30 | 276.71 |

Table 6: Log-likelihood with pledges specificity weight (**S**) and count of sentences (**C**) across 57 policy themes as independent variables. Log-likelihood values using **S** are better than **C** across all the policy areas.

### 6.2 Capturing Issue Salience

For the Australian manifestos (from the Greens, Labor, Liberal, and National parties) we perform a qualitative study of specificity weight across policy themes, by correlating it against the salience of major policy areas given by the Voter Survey (Cameron and McAllister, 2019). Again we compare its utility over the use of counts across policy themes in a manifesto. Using sentences classified with policy themes and specificity scores using our proposed approach, we construct the following $|\text{Manifestos}| \times 57$ features — frequency distribution (**C**) and pledge specificity weight (**S**) across policy themes. The features are used as independent variables, and voter survey salience scores across major policy areas — health, education, environment, tax, and economy — are treated as dependent variables. Note that the voter survey scores are available for each party and election cycle across policy areas. We build separate multivariate linear regression models and compare them based on the goodness of fit (log-likelihood). Log-likelihood values are given in Table 6: across all policy areas, pledge specificity better captures salience than a count-based representation.

## 7 Conclusion and Future Work

In this work we present a new dataset of election campaign texts, annotated with pledge specificity on a fine-grained scale. We study the use of deep ordinal regression approaches using an auxiliary uni-modal distributional loss for this task. The proposed approaches provide large gains in performance under both supervised and semi-supervised settings. Specificity weight across policy issues benefits ideology prediction and also better captures issue salience, compared to the traditional policy theme count-based representation. This aligns with previous studies done based on manual annotations (Praprotnik, 2017). In future work, we aim to expand this study to multiple languages.

## Acknowledgements

We thank Robert Thomson for his inputs on the task definition. This work was funded in part by the Australian Government Research Training Program Scholarship, and the Australian Research Council.

## References


Ron Artstein and Massimo Poesio. 2008. Inter-coder agreement for computational linguistics. *Computational Linguistics*, 34(4):555–596.

Stephen H Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. 2017. Hinge-loss Markov random fields and probabilistic soft logic. *Journal of Machine Learning Research*, 18(109):1–67.

Stephen H. Bach, Bert Huang, Ben London, and Lise Getoor. 2013. Hinge-loss Markov random fields: Convex inference for structured prediction. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, pages 32–41.

Ryan Bakker, Catherine De Vries, Erica Edwards, Liesbet Hooghe, Seth Jolly, Gary Marks, Jonathan Polk, Jan Rovny, Marco Steenbergen, and Milada Anna Vachudova. 2015. Measuring party positions in Europe: The Chapel Hill expert survey trend file, 1999–2010. *Party Politics*, 21(1):143–152.

Christopher Beckham and Christopher Pal. 2016. A simple squared-error reformulation for ordinal classification. *arXiv preprint arXiv:1612.00775*.

Christopher Beckham and Christopher Pal. 2017. Unimodal probability distributions for deep ordinal classification. In *International Conference on Machine Learning*, pages 411–419.

Kenneth Benoit and Michael Laver. 2007. Estimating party policy positions: Comparing expert surveys and hand-coded content analysis. *Electoral Studies*, 26(1):90–107.

Ian Budge and Dennis Farlie. 1983. Party competition: selective emphasis or direct confrontation?: an alternative view with data. *Western European party systems : continuity & change*, pages 267–305.

Sarah Cameron and Ian McAllister. 2019. Australian election study, 1987-2016 trends.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734.

Kevin Clark, Minh-Thang Luong, Christopher D Manning, and Quoc V Le. 2018. Semi-supervised sequence modeling with cross-view training. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1914–1925.

Ian Cook. 2016. *Content and Context: Three Essays on Information in Politics*. Ph.D. thesis, University of Pittsburgh.

Joaquim F Pinto da Costa, Hugo Alonso, and Jaime S Cardoso. 2008. The unimodal model for the classification of ordinal data. *Neural Networks*, 21(1):78–91.

Thomas Däubler and Kenneth Benoit. 2017. Estimating better left-right positions through statistical scaling of manual content analysis. *Retrieved from* `http://kenbenoit.net/pdfs/text_in_context_2017.pdf`.

Peter Dixon. 1987. The processing of organizational and component step information in written directions. *Journal of memory and language*, 26(1):24–35.

Nikolaus Eder, Marcelo Jenny, and Wolfgang C Müller. 2017. Manifesto functions: How party candidates view and use their party's central policy document. *Electoral Studies*, 45:75–87.

Eran Eidinger, Roee Enbar, and Tal Hassner. 2014. Age and gender estimation of unfiltered faces. *IEEE Transactions on Information Forensics and Security*, 9(12):2170–2179.

Pablo Fernandez-Vazquez. 2014. And yet it moves: The effect of election platforms on party policy images. *Comparative Political Studies*, 47(14):1919–1944.

Matthew J Gabel and John D Huber. 2000. Putting parties in their place: Inferring party left-right ideological positions from party manifestos data. *American Journal of Political Science*, pages 94–103.

Bin-Bin Gao, Chao Xing, Chen-Wei Xie, Jianxin Wu, and Xin Geng. 2017. Deep label distribution learning with label ambiguity. *IEEE Transactions on Image Processing*, 26(6):2825–2838.

Yifan Gao, Yang Zhong, Daniel Preotiuc-Pietro, and Junyi Jessy Li. 2019. Predicting and analyzing language specificity in social media posts. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*.

AE Gentry, CK Jackson-Cook, DE Lyon, and KJ Archer. 2015. Penalized ordinal regression methods for predicting stage of cancer in high-dimensional covariate spaces. *Cancer informatics*, 14(Suppl 2):201–208.

Zachary Greene. 2016. Competing on the issues: How experience in government and economic conditions influence the scope of parties' policy messages. *Party Politics*, 22(6):809–822.

Le Hou, Chen-Ping Yu, and Dimitris Samaras. 2016. Squared earth mover's distance-based loss for training deep neural networks. *arXiv preprint arXiv:1611.05916*.

Ehsan Imani and Martha White. 2018. Improving regression performance with distributional losses. In *Proceedings of the International Conference on Machine Learning*, pages 2162–2171.

Willy Jou and Russell J. Dalton. 2017. Left-right orientations and voting behavior. *Oxford Research Encyclopedia of Politics*.

Klaus Krippendorff. 2011. Computing Krippendorff's alpha-reliability. *Technical Report, University of Pennsylvania.*

Junyi Jessy Li and Ani Nenkova. 2015. Fast and accurate prediction of sentence specificity. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2281–2287. AAAI Press.

Junyi Jessy Li, Bridget O'Daniel, Yi Wu, Wenli Zhao, and Ani Nenkova. 2016. Improving the annotation of sentence specificity. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 3921–3927.

Yang Liu, Kun Han, Zhao Tan, and Yun Lei. 2017. Using context information for dialog act classification in DNN framework. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2170–2178.

Annie Louis and Ani Nenkova. 2011. Automatic identification of general and specific sentences by leveraging discourse annotations. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 605–613.

Will Lowe, Kenneth Benoit, Slava Mikhaylov, and Michael Laver. 2011. Scaling policy preferences from coded political texts. *Legislative studies quarterly*, 36(1):123–155.

Luca Lugini and Diane Litman. 2017. Predicting specificity in classroom discussion. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 52–61.

Wencan Luo and Diane Litman. 2016. Determining the quality of a student reflective response. In *The Twenty-Ninth International FLAIRS Conference*.

Nicolas Merz, Sven Regel, and Jirka Lewandowski. 2016. The Manifesto Corpus: A new resource for research on political parties and quantitative text analysis. *Research & Politics*, 3(2).

E. Naurin. 2011. *Election Promises, Party Behaviour and Voter Perception*. Palgrave Macmillan.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2227–2237.

Gerald M. Pomper and Susan S. Lederman. 1980. *Elections in America: Control and Influence in Democratic Politics*. Longman.

Katrin Praprotnik. 2017. Issue clarity in electoral competition. Insights from Austria. *Electoral Studies*, 48:121–130.

David Bruce Robertson et al. 1976. *A theory of party competition*. Wiley London.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. Semeval-2017 task 4: Sentiment analysis in Twitter. In *Proceedings of the 11th international workshop on semantic evaluation (SemEval-2017)*, pages 502–518.

Rasmus Rothe, Radu Timofte, and Luc Van Gool. 2018. Deep expectation of real and apparent age from a single image without facial landmarks. *International Journal of Computer Vision*, 126(2-4):144–157.

Terry J Royed. 1996. Testing the mandate model in Britain and the United States: Evidence from the Reagan and Thatcher eras. *British Journal of Political Science*, 26(1):45–80.

Terry J Royed, Elin Naurin, and Robert Thomson. 2019. *Making and Keeping Pledges*. New Comparative Politics.

Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. 2016. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 1163–1171.

Katrin Schermann and Laurenz Ennser-Jedenastik. 2014. Coalition policy-making under constraints: Examining the role of preferences and institutions. *West European Politics*, 37(3):564–583.

Jonathan B Slapin and Sven-Oliver Proksch. 2008. A scaling model for estimating time-series party positions from texts. *American Journal of Political Science*, 52(3):705–722.

Shivashankar Subramanian, Trevor Cohn, and Timothy Baldwin. 2018. Hierarchical structured model for fine-to-coarse manifesto text analysis. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1964–1974.

Shivashankar Subramanian, Trevor Cohn, and Timothy Baldwin. 2019. Target based speech act classification in political campaign text. In *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (* SEM 2019)*, pages 273–282.

Robert Thomson. 2001. The programme to policy linkage: The fulfilment of election pledges on socio–economic policy in the Netherlands, 1986–1998. *European Journal of Political Research*, 40(2):171–197.

Suzan Verberne, Eva D'hondt, Antal van den Bosch, and Maarten Marx. 2014. Automatic thematic classification of election manifestos. *Information Processing & Management*, 50(4):554–567.

Andrea Volkens, Judith Bara, Budge Ian, and Simon Franzmann. 2013. Understanding and validating the left-right scale (RILE). In *Mapping Policy Preferences From Texts: Statistical Solutions for Manifesto Analysts*, chapter 6. Oxford University Press.

Andrea Volkens, Pola Lehmann, Theres Matthieß, Nicolas Merz, Sven Regel, and Bernhard Weßels. 2017. *The Manifesto Data Collection. Manifesto Project (MRG/CMP/MARPOR). Version 2017b*. Wissenschaftszentrum Berlin für Sozialforschung.

Xiang Wei, Boqing Gong, Zixia Liu, Wei Lu, and Liqiang Wang. 2018. Improving the improved training of Wasserstein GANs: A consistency term and its dual effect. In *Proceedings of the International Conference on Learning Representations*.

Cäcilia Zirn, Goran Glavaš, Federico Nanni, Jason Eichorts, and Heiner Stuckenschmidt. 2016. Classifying topics and detecting topic shifts in political manifestos. In *Proceedings of the International Conference on the Advances in Computational Analysis of Political Text*, pages 88–93.

# Data-Efficient Goal-Oriented Conversation with Dialogue Knowledge Transfer Networks

**Igor Shalyminov**[†], **Sungjin Lee**[‡], **Arash Eshghi**[†], **and Oliver Lemon**[†]
[†]Heriot-Watt University, UK
[‡]Amazon Alexa AI, USA
[†]{is33, a.eshghi, o.lemon}@hw.ac.uk, [‡]sungjinl@amazon.com

## Abstract

Goal-oriented dialogue systems are now being widely adopted in industry where it is of key importance to maintain a rapid prototyping cycle for new products and domains. Data-driven dialogue system development has to be adapted to meet this requirement — therefore, reducing the amount of data and annotations necessary for training such systems is a central research problem.

In this paper, we present the Dialogue Knowledge Transfer Network (DiKTNet), a state-of-the-art approach to goal-oriented dialogue generation which only uses a few example dialogues (i.e. few-shot learning), none of which has to be annotated. We achieve this by performing a 2-stage training. Firstly, we perform *unsupervised dialogue representation pre-training* on a large source of goal-oriented dialogues in multiple domains, the MetaLWOz corpus. Secondly, at the *transfer stage*, we train DiKTNet using this representation together with 2 other textual knowledge sources with different levels of generality: ELMo encoder and the main dataset's source domains.

Our main dataset is the Stanford Multi-Domain dialogue corpus. We evaluate our model on it in terms of BLEU and Entity F1 scores, and show that our approach significantly and consistently improves upon a series of baseline models as well as over the previous state-of-the-art dialogue generation model, ZSDG. The improvement upon the latter — up to **10%** in Entity F1 and the average of **3%** in BLEU score — is achieved using only **10%** equivalent of ZSDG's in-domain training data.

## 1 Introduction

Machine learning-based dialogue systems, while still being a relatively new research direction, are experiencing increasingly wide adoption in industry. Large-scale dialogue assistant platforms such as *Google Assistant, Amazon Alexa, and Apple Siri* provide a unified conversational user interface (CUI) for third-party applications and services. Furthermore, products like *Google Dialogflow, Wit.ai, Microsoft LUIS, and Rasa* offer means for rapid development of a dialogue system's core modules. In addition, with the recently adopted technique of training dialogue systems end-to-end *data-efficiency* of such systems becomes the key question in their adoption in practical applications. Currently, while being extremely flexible and requiring little to no programming of in-domain business logic (see e.g. Ultes et al. (2018); Wen et al. (2017); Rojas-Barahona et al. (2017)), such systems have too high data consumption — including both collection and annotation effort — in order for them to be used in rapidly paced industrial product cycles. Therefore, approaches to training such systems with extremely limited data (i.e. zero-, one- and few-shot training) are a priority research direction in the dialogue systems area.

In this paper, we present the *Dialogue Knowledge Transfer Network* (or DiKTNet), a generative goal-oriented dialogue model designed for few-shot learning, i.e. training only using a small number of complete in-domain dialogues. The key underlying concept of this model is transfer learning: DiKTNet makes use of the latent text representation learned from several sources ranging from large-scale general-purpose textual corpora to similar dialogues in the domains different to the target one. We use the evaluation framework of Zhao and Eskénazi (2018) and the same dataset, and mainly compare our approach to theirs. While their method doesn't require complete in-domain dialogues and uses annotated utterances instead (and is therefore described as "zero-shot"), we show that our model achieves superior performance with roughly the same amount of data (with



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1741–1751,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

respect to in-domain utterances) while requiring no annotations whatsoever.

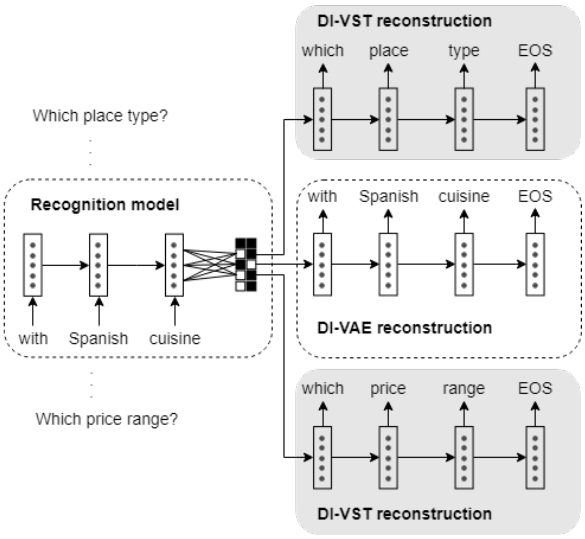


Figure 1: DI-VAE and DI-VST (DiKTNet Stage 1)

## 2 Related Work

The problem of data efficiency of dialogue systems has been extensively researched in the past. Starting with domain adaptation of a dialogue state tracker (Henderson et al., 2014) approached using Bayesian Processes (Gasic et al., 2017) and Recurrent Neural Networks (Mrksic et al., 2015), there has been significant work on training different dialogue system components using as little data as possible. As such, Williams et al. (2017) introduced a dialogue management model designed for bootstrapping from limited training data and further fine-tuning. A recent paper by Vlasov et al. (2018) introduced a dialogue management model which uses a unified embedding space for user and system turns allowing efficient cross-domain knowledge transfer.

There also exist approaches to end-to-end dialogue generation. Eshghi et al. (2017) proposed a linguistically informed model based on an incremental semantic parser (Eshghi et al., 2011) combined with a reinforcement learning-based agent. The parser was used for both maintaining the agent's state and pruning the agent's incremental, word-level generation actions (only the actions leading to syntactically correct word sequences were allowed for the agent to take). While outperforming end-to-end dialogue models on bAbI Dialog Tasks in a zero-shot setup (Shalyminov et al., 2017) due to its prior linguistic knowledge in the form of a dialogue grammar, this method inherited the limitations of it as well. Specifically, it's limited to a single domain until a wide-coverage grammar is available.

Meta-learning has also gained a lot of attention as a way to train models for maximally efficient adaptation to new data. As such, Qian and Yu (2019) presented such approach for fast adaptation of a dialogue model to a new domain. While highly promising, its main result was achieved on a synthetic dataset and would ideally need more testing on real data.

Finally, the method we directly compare our approach to is that of Zhao and Eskénazi (2018) who introduced the Zero-Shot Dialogue Generation (ZSDG) task and the corresponding model. In their work, they use a unified latent space for user utterances, system turns, and *domain descriptions* in the form of utterance-annotation pairs. Since they only used such utterances and no full dialogues for the target domain, they presented this approach as "zero-shot" learning. In our approach, we do use complete in-domain dialogues, but with significantly less data with respect to the number of in-domain utterances. Moreover, our method requires no annotation whatsoever.

Recent research in Natural Language Processing has shown that the transfer of text representation learned on larger data sources benefits target models' performance, just as was the case with ImageNet-based computer vision models (Deng et al., 2009).

For text, the main means for transfer was Word2Vec and GloVe embeddings (Mikolov et al., 2013; Pennington et al., 2014) recently extended with context-aware models like ELMo (Peters et al., 2018) and BERT (Devlin et al., 2018). Trained on large and diverse textual corpora, they were shown to improve target models' performance on a number of Natural Language Processing tasks. Although highly beneficial, those models' use may not be sufficient for the case of dialogue as response generation for goal-oriented dialogue from extremely limited data requires specialized tools. General-purpose embeddings lack specificity for close dialogue domains since they have been learned from very heterogeneously distributed data: in dialogue, the distribution of word sequences is highly specific to a given domain or task, i.e. word sequences in dialogue can take on an astonishingly wide variety of meanings in dif-

ferent contexts.

In this paper, we will work with *autoencoders*, a class of unsupervised text representation models working via reconstructing the input — specifically, a Variational Autoencoder (VAE) was considered the main means to learn robust text representations (Bowman et al., 2016). Although the model itself was challenging to train and was mainly used with plenty of workarounds, and recently there started to appear variants of this model with improved stability. One such model we will use in this paper is that of Zhao et al. (2018) (see Section 4.1 for more detail).

## 3 Few-Shot Dialogue Generation

We first describe the task we are addressing in this paper, and the corresponding base model. Specifically, we have a set of dialogues in source domains and just a few seed dialogues in the target domain. And the model's task is, having been trained on all the available *source data*, to fine-tune on the *target data* to be further evaluated on the full set of target-domain dialogues.

We are basing our model for this task on a Hierarchical Encoder-Decoder (HRED) architecture with attention-based copying (Merity et al., 2017). The base optimization objective is as follows:

$$\mathcal{L}_{\text{HRED}} = \log p_{\mathcal{F}^d}(\boldsymbol{x}_{\text{sys}} \mid \mathcal{F}^e(\boldsymbol{c}, \boldsymbol{x}_{\text{usr}})) \quad (1)$$

where $\boldsymbol{x}_{\text{usr}}$ is user's query, $\boldsymbol{x}_{\text{sys}}$ is the system's response, $\boldsymbol{c}$ is the dialogue context, and $\mathcal{F}^e$ and $\mathcal{F}^d$ are respectively hierarchical encoder and decoder.

We work with goal-oriented dialogues, so it's natural in our setting to take into account an underlying Knowledge Base (or API) providing results on the user's queries. Given that such KB information may contain unseen token sequences for the most part, especially in the target domain, we use a copy mechanism in order to be able to use this information in the system's responses. More specifically, we represent KB info as token sequences and concatenate it to the dialogue context similarly to CopyNet setup of Eric et al. (2017). Our copy mechanism's implementation is the Pointer-Sentinel Mixture Model (Merity et al., 2017; Zhao and Eskénazi, 2018):

$$p(w_t \mid s_t) = g p_{\text{vocab}}(w_t \mid s_t) + (1 - g) p_{\text{ptr}}(w_t \mid s_t) \quad (2)$$

In the formula above, $w_t$ and $s_t$ are respectively the output word and the decoder state at step $t$; $p_{\text{ptr}}$ is the probability of attention-based copying of the word $w_t$, and $g$ is the mixture weight:

$$p_{\text{ptr}}(w_t \mid s_t) = \sum_{k_j \in I(w, \boldsymbol{x})} \alpha_{k_j, t} \quad (3)$$

$$g = \text{Softmax}(u^T \tanh(W_\alpha s_i)) \quad (4)$$

where $\alpha_{k_j,t}$ is the attention weight for $k$th token in flattened dialogue context at the decoding step $t$ and $u$ is the sentinel vector — for more detail, see (Zhao and Eskénazi, 2018).

## 4 Dialogue Knowledge Transfer Network

Transfer learning is considered the key means for efficient training with minimal data, and our DiKTNet model essentially introduces several knowledge-transfer augmentations to the base HRED model described above. DiKTNet training is performed in two stages described below.

### 4.1 Stage 1. Dialogue representation pre-training

Dialogue structure — e.g. word sequences — is highly specific to a given domain or task, and the meaning of conversational utterances is highly contextual, i.e. similar utterances may have different meanings depending on the context. Nevertheless, there is a lot of similarity in dialogue structure — i.e. sequences of dialogue actions — across domains, e.g. a conversation normally starts with a mutual greeting and a question is very often followed by an answer. Here, we propose to exploit this phenomenon in the form of learning a latent dialogue action representation in order to better capture the dialogue structure by abstracting away from surface forms. Crucially, we learn such representation from MetaLWOz (Lee et al., 2019), a dataset specifically created for the purposes of meta-learning and transfer learning and consisting of human-human conversations in 51 unique domains (for more detail, see Section 6).

For this stage of training we use unsupervised, variational autoencoder-based (VAE) representation learning following the *Latent Action Encoder-Decoder (LAED)* approach of Zhao et al. (2018). LAED's underlying model is called *Discrete Information VAE (DI-VAE)*, a variant of a VAE with

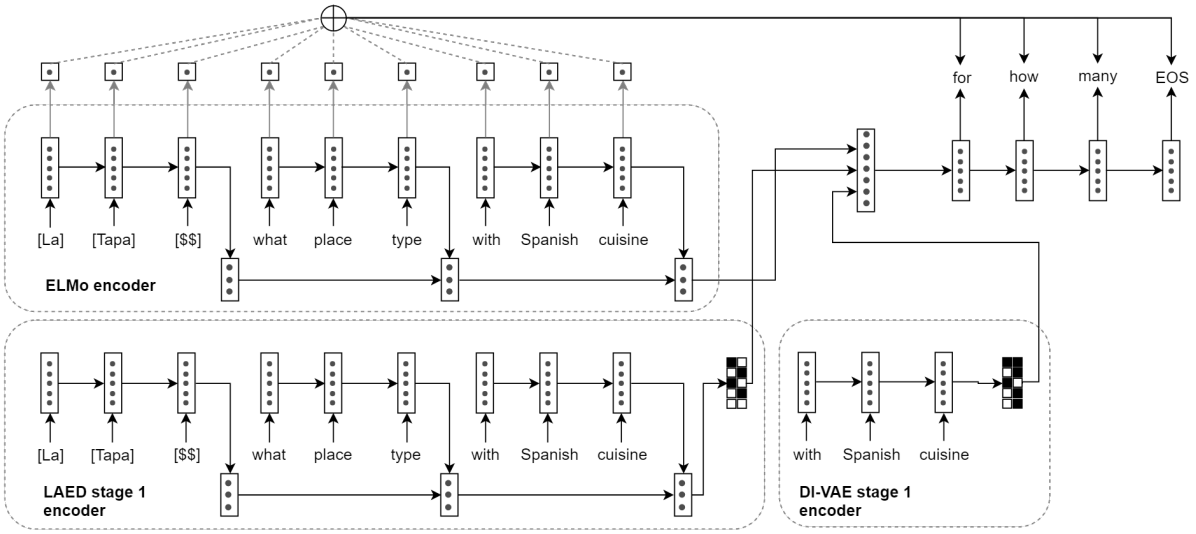


Figure 2: DiKTNet Stage 2 (tokens in brackets are KB data)

two modifications. Firstly, its optimization objective accounts for the mutual information $I$ between the input and the latent variable which is implicitly discouraged in the original VAE objective (see Eqs. 5 and 6).

$$\begin{aligned}\mathcal{L}_{VAE} &= \mathbb{E}_x[\mathbb{E}_{q_\mathcal{R}(\boldsymbol{z}|\boldsymbol{x})}[\log p_\mathcal{G}(\boldsymbol{x} \mid \boldsymbol{z})] \\ &\quad - KL(q(\boldsymbol{z})\|p(\boldsymbol{z}))] = \\ &\quad \mathbb{E}_{q(\boldsymbol{z}|x)p(x)}[\log p_\mathcal{G}(\boldsymbol{x} \mid \boldsymbol{z})] \\ &\quad - I(Z, X) - KL(q(\boldsymbol{z})\|p(\boldsymbol{z})),\end{aligned} \tag{5}$$

$$\begin{aligned}\mathcal{L}_{DI\text{-}VAE} &= \mathcal{L}_{VAE} + I(Z, X) \\ &= \mathbb{E}_{q_\mathcal{R}(\boldsymbol{z}|\boldsymbol{x})p(\boldsymbol{x})}[\log p_\mathcal{G}(\boldsymbol{x} \mid \boldsymbol{z})] \\ &\quad - KL(q(\boldsymbol{z})\|p(\boldsymbol{z}))\end{aligned} \tag{6}$$

where $\boldsymbol{x}$ is the input utterance, $\boldsymbol{z}$ is the latent variable ($X$ and $Z$ corresponding to their batch-wise vectors), $\mathcal{R}$ and $\mathcal{G}$ are the recognition and generation models (implemented as RNNs) respectively, and $q(\boldsymbol{z}) = \mathbb{E}_x[q_\mathcal{R}(\boldsymbol{z} \mid x)]$.

Secondly, the latent variable $\boldsymbol{z}$ in DI-VAE is *discrete* as opposed to the continuous one in a vanilla VAE. The discrete latent code lends itself well to interpretation and can be viewed as a form of unsupervised dialogue act tagging. The discrete nature also makes the calculation of the KL-term more tractable via the *Batch Prior Regularization* technique (Zhao et al., 2018):

$$KL(q'(\boldsymbol{z})\|p(\boldsymbol{z})) = \sum_{k=1}^{K} q'(\boldsymbol{z}=k) \log \frac{q'(\boldsymbol{z}=k)}{p(\boldsymbol{z}=k)} \tag{7}$$

where $K$ is the number of $\boldsymbol{z}$'s possible values and $q'(\boldsymbol{z})$ is the approximation to $q(\boldsymbol{z})$ over $N$ data points:

$$q'(\boldsymbol{z}) = \frac{1}{N}\sum_{n=1}^{N} q_\mathcal{R}(\boldsymbol{z} \mid x_n) \tag{8}$$

In addition, we employ ***DI-VST***, DI-VAE's counterpart working in a Variational Skip-Thought manner (Hill et al., 2016) and reconstructing the input $\boldsymbol{x}$'s previous ($\boldsymbol{x_p}$) and next ($\boldsymbol{x_n}$) context utterances instead:

$$\begin{aligned}\mathcal{L}_{DI\text{-}VST} &= \\ &\mathbb{E}_{q_\mathcal{R}(\boldsymbol{z}|\boldsymbol{x})p(\boldsymbol{x})}[\log p^n_\mathcal{G}(\boldsymbol{x_n} \mid \boldsymbol{z}) p^p_\mathcal{G}(\boldsymbol{x_p} \mid \boldsymbol{z})] \\ &- KL(q(\boldsymbol{z})\|p(\boldsymbol{z}))\end{aligned} \tag{9}$$

DI-VAE and DI-VST models are visualized in Figure 1.

In the downstream DiKTNet model, we use DI-VAE autoencoder in order to obtain the representation of the user's query: $\boldsymbol{z}_{\text{usr}} = \text{DI-VAE}(\boldsymbol{x}_{\text{usr}})$.

In turn, DI-VST is used to obtain a prediction of the system's action $\boldsymbol{z}_{\text{sys}}$ in the discretized latent form given the user's input $\boldsymbol{x}_{\text{usr}}$ as well as the full dialogue context $\boldsymbol{c}$. For that, DI-VST autoencoder is used as part of a hierarchical, context-aware encoder-decoder response generation model (we refer to it as ***LAED*** itself). Its optimization

objective is as follows:

$$\mathcal{L}_{LAED}(\theta_{\mathcal{F}}, \theta_{\pi}) = \mathbb{E}_{q_{\mathcal{R}}(\boldsymbol{z}_{\text{sys}}|\boldsymbol{x}_{\text{sys}})p(\boldsymbol{x}_{\text{sys}},\boldsymbol{c})}[ \log p_{\pi}(\boldsymbol{z}_{\text{sys}} \mid \boldsymbol{c}) + \log p_{\mathcal{F}}(\boldsymbol{x}_{\text{sys}} \mid \boldsymbol{z}_{\text{sys}}, \boldsymbol{c})] \quad (10)$$

where $\theta_{\mathcal{F}}$ is the set of parameters of the context-aware encoder and decoder, $\theta_{\pi}$ is the set of parameters of the policy $\theta_{\pi}$. $\theta_{\pi}$ is the component trained to directly predict $\boldsymbol{z}_{\text{sys}}$ from the context $\boldsymbol{c}$.

We use different models for different aspects of the dialogue: DI-VAE for user's utterance representation, and DI-VST-based LAED — for the system's action prediction. In that, we follow the intuition of Zhao and Eskénazi (2018) who said that DI-VAE is better at capturing specific words of an utterance, while DI-VST represents the overall dialogue action better.

We train these two models on MetaLWOz in an unsupervised way with the objectives as described above, and use their discretized latent codes $\boldsymbol{z}_{\text{usr}}$ and $\boldsymbol{z}_{\text{sys}}$ respectively in the downstream model at the next stage of training.

### 4.2 Stage 2. Transfer

At this stage, we train directly for our target task, few-shot dialogue generation, and thus go back to the model described in Section 3. While the training procedure of this model naturally assumes *domain transfer*, we will provide it with more sources of textual and dialogue knowledge of varying generality described below.

As opposed to direct domain transfer, we incorporate domain-general dialogue understanding from the LAED representation trained on MetaLWOz at the previous stage. LAED captures the background top-down dialogue structure: sequences of dialogue acts in a cooperative conversation, latent dialog act-induced clustering of utterances, and the overall phrase structure of spoken utterances. We incorporate this information into the model by conditioning HRED's decoder on the combined latent codes from Stage 1 and refer to this model as ***HRED+LAED***.

$$\mathcal{L}_{\text{HRED+LAED}} = \mathbb{E}_{p(\boldsymbol{x}_{\text{usr}},\boldsymbol{c})p(\boldsymbol{z}_{\text{usr}}|\boldsymbol{x}_{\text{usr}})p_{\pi}(\boldsymbol{z}_{\text{sys}}|\boldsymbol{x}_{\text{usr}},\boldsymbol{c})}[ \log p_{\mathcal{F}^d}(\boldsymbol{x}_{\text{sys}} \mid \{\mathcal{F}^e(\boldsymbol{x}_{\text{usr}}, \boldsymbol{c}), \boldsymbol{z}_{\text{usr}}, \boldsymbol{z}_{\text{sys}}\})]. \quad (11)$$

where $\boldsymbol{z}_{\text{usr}}$ and $\boldsymbol{z}_{\text{sys}}$ are respectively samples obtained from the DI-VAE user utterance model and LAED/DI-VST system action model, and $\{\}$ is the concatenation operator.

The last, most general source of knowledge we use is a pre-trained ELMo model (Peters et al., 2018). Apart from using an underlying bidirectional RNN encoder, ELMo captures both token-level and character-level information which is especially crucial in understanding unseen tokens and KB items in the underrepsesented target domain. HRED model with ELMo as the utterance-level encoder is referred to as ***HRED+ELMo***.

Finally, ***DiKTNet*** is HRED augmented with both ELMo encoder and LAED representation.

DiKTNet is visualized in Figure 2. The model (as well as its variants listed above) is implemented in PyTorch (Paszke et al., 2017), and the code is openly available[1].

## 5 Baselines

We perform an exhaustive ablation study of DiKTNet by comparing it to all its variations mentioned above: HRED, HRED+ELMo, and HRED+LAED. In addition to that, we have the ***HRED+VAE*** —a version of HRED+LAED for which we use a regular, continuous VAE behind DI-VAE and DI-VST in order to see the impact of discretized latent codes (see Eq 5 for the corresponding objective function).

Furthermore, we compare DiKTNet to the previous state-of-the-art approach, Zero-Shot Dialogue Generation (Zhao and Eskénazi, 2018). This model didn't use any complete in-domain dialogues but instead it relied on annotated utterances in all of the domains. We use it as-is (***ZSDG***) as well its variation as follows.

We make use of its central idea of 'domain descriptions' bridging dialogue understanding across domains, but instead of using manually annotated utterances, we employ automatic Natural Language Understanding markup. Our NLU annotations include:

- Named Entity Recognition — Stanford NER model ensemble of case-sensitive and caseless models (Finkel et al., 2005),
- date/time markup — Stanford SUTime (Chang and Manning, 2012),
- Wikidata entity linking — Yahoo FEL (Blanco et al., 2015; Pappu et al., 2017).

[1] https://bit.ly/fsdg_emnlp2019

| Model \ Domain | Navigation | | Weather | | Schedule | |
|---|---|---|---|---|---|---|
| | BLEU, % | Entity F1, % | BLEU, % | Entity F1, % | BLEU, % | Entity F1, % |
| ZSDG | 5.9 | 14.0 | 8.1 | 31 | 7.9 | 36.9 |
| NLU_ZSDG | $6.1 \pm 2.2$ | $12.7 \pm 3.3$ | $5.0 \pm 1.6$ | $16.8 \pm 6.7$ | $6.0 \pm 1.7$ | $26.5 \pm 5.4$ |
| NLU_ZSDG+LAED | $7.9 \pm 1$ | $12.3 \pm 2.9$ | $8.7 \pm 0.6$ | $21.5 \pm 6.2$ | $8.3 \pm 1$ | $20.7 \pm 4.8$ |
| HRED@1% | $6.0 \pm 1.8$ | $9.8 \pm 4.8$ | $6.9 \pm 1.1$ | $22.2 \pm 10.7$ | $5.5 \pm 0.8$ | $25.6 \pm 8.2$ |
| HRED@3% | $7.9 \pm 0.7$ | $11.8 \pm 4.4$ | $9.6 \pm 1.8$ | $39.8 \pm 7$ | $8.2 \pm 1.1$ | $34.8 \pm 4.4$ |
| HRED@5% | $8.3 \pm 1.3$ | $15.3 \pm 6.3$ | $11.5 \pm 1.6$ | $38.0 \pm 10.5$ | $9.7 \pm 1.4$ | $37.6 \pm 8.0$ |
| HRED@10% | $9.8 \pm 0.8$ | $19.2 \pm 3.2$ | $12.9 \pm 2.4$ | $40.4 \pm 11.0$ | $12.0 \pm 1.0$ | $38.2 \pm 4.2$ |
| HRED+VAE@1% | $3.6 \pm 2.6$ | $9.3 \pm 4.1$ | $6.8 \pm 1.3$ | $23.2 \pm 10.1$ | $4.6 \pm 1.6$ | $28.9 \pm 7.3$ |
| HRED+VAE@3% | $6.9 \pm 1.9$ | $15.6 \pm 5.8$ | $9.5 \pm 2.6$ | $32.2 \pm 11.8$ | $6.6 \pm 1.7$ | $34.8 \pm 7.7$ |
| HRED+VAE@5% | $7.8 \pm 1.9$ | $12.7 \pm 4.2$ | $10.1 \pm 2.1$ | $40.3 \pm 10.4$ | $8.2 \pm 1.7$ | $34.2 \pm 8.7$ |
| HRED+VAE@10% | $9.0 \pm 2.0$ | $18.0 \pm 5.8$ | $12.9 \pm 2.2$ | $40.1 \pm 7.6$ | $11.6 \pm 1.5$ | $39.9 \pm 6.9$ |
| HRED+LAED@1% | $7.1 \pm 0.8$ | $10.1 \pm 4.5$ | $10.6 \pm 2.1$ | $31.4 \pm 8.1$ | $7.4 \pm 1.2$ | $29.1 \pm 6.6$ |
| HRED+LAED@3% | $9.2 \pm 0.8$ | $14.5 \pm 4.8$ | $13.1 \pm 1.7$ | $40.8 \pm 6.1$ | $9.2 \pm 1.2$ | $32.7 \pm 6.1$ |
| HRED+LAED@5% | $10.3 \pm 1.2$ | $15.6 \pm 4.5$ | $14.5 \pm 2.2$ | $40.9 \pm 8.6$ | $11.8 \pm 1.9$ | $37.6 \pm 6.1$ |
| HRED+LAED@10% | $12.3 \pm 0.9$ | $17.3 \pm 4.5$ | $17.6 \pm 1.9$ | $47.5 \pm 6.0$ | $15.2 \pm 1.6$ | $38.7 \pm 8.4$ |
| HRED+ELMo@1% | $5.8 \pm 1.9$ | $18.2 \pm 3.8^{\star}$ | $7.3 \pm 2.6$ | $38.5 \pm 11.1$ | $6.3 \pm 2.6$ | $36.3 \pm 9.2$ |
| HRED+ELMo@3% | $8.0 \pm 1.3$ | $17.2 \pm 4.2$ | $10.6 \pm 1.1$ | $42.0 \pm 11.0$ | $9.5 \pm 2.0$ | $39.6 \pm 9.2$ |
| HRED+ELMo@5% | $9.4 \pm 0.8$ | $21.5 \pm 7.3$ | $12.1 \pm 2.0$ | $39.0 \pm 12.8$ | $11.3 \pm 2.1$ | $40.0 \pm 5.6$ |
| HRED+ELMo@10% | $9.9 \pm 1.1$ | $24.3 \pm 5.7$ | $14.9 \pm 2.7$ | $41.4 \pm 12.0$ | $14.5 \pm 1.4$ | $43.4 \pm 3.9$ |
| **DiKTNet@1%** | $\mathbf{8.4 \pm 0.7^{\star}}$ | $\mathbf{15.2 \pm 4.0}$ | $\mathbf{11.5 \pm 1.7^{\star}}$ | $\mathbf{43.0 \pm 10.5^{\star}}$ | $\mathbf{8.1 \pm 0.8^{\star}}$ | $\mathbf{40.5 \pm 6.3^{\star}}$ |
| DiKTNet@3% | $10.4 \pm 1.2$ | $19.2 \pm 4.8$ | $15.7 \pm 2.1$ | $44.0 \pm 11.7$ | $11.1 \pm 1.3$ | $38.2 \pm 5.8$ |
| DiKTNet@5% | $11.5 \pm 1.1$ | $23.9 \pm 2.9$ | $15.5 \pm 2.1$ | $39.5 \pm 14.8$ | $13.7 \pm 2.0$ | $41.1 \pm 3.8$ |
| DiKTNet@10% | $12.9 \pm 1.0$ | $26.8 \pm 4.2$ | $20.4 \pm 1.2$ | $48.0 \pm 5.6$ | $17.5 \pm 1.3$ | $42.8 \pm 2.6$ |

Table 1: Evaluation results. Marked with asterisks are individual results higher than ZSDG's performance which are achieved with the minimum amount of training data. In bold is the model consistently outperforming ZSDG in all domains and metrics with minimum data.

We serialize annotations from these sources into token sequences and make domain description tuples out of all the utterances in the source and target domains. This way, most of our domain descriptions share the structure and content of the original ones.

For example, for the phrase *'Will it be cloudy in Los Angeles on Thursday?'*, the original ZSDG annotation is of the form `"request #goal cloudy #location Los Angeles #date Thursday"`. Our NLU annotation for this phrase is `"LOCATION Los Angeles DATE Thursday"`.

We have two models in this setup, with (***NLU_ZSDG+LAED***) and without the use of LAED representation (***NLU_ZSDG***) respectively.

## 6 Datasets

| | |
|---|---|
| Number of Domains: | 51 |
| Number of Dialogues: | 40,388 |
| Mean dialogue length: | 11.91 |

Table 2: MetaLWOz dataset statistics

We use the Stanford Multi-Domain (SMD) dialogue dataset (Eric et al., 2017) containing human-human goal-oriented dialogues in three domains: appointment scheduling, city navigation, and weather information. Each dialogue has to do with a single task queried by the user and thus comes with additional knowledge base information coming from implicit querying of the underlying domain-specific API. Although sharing some common features (the setting of an intelligent in-car assistant and the use of the underlying KB), the dialogues differs significantly across domains which makes doamin transfer sufficiently challenging.

| Statistic \ Domain | Navigation | Weather | Schedule |
|---|---|---|---|
| Dialogues | 800 | 797 | 828 |
| Utterances | 5248 | 4314 | 3170 |
| Avg. dialogue length | 6.56 | 5.41 | 3.83 |

Table 3: Stanford multi-domain dataset statistics (trainset)

For the latent representation learning, we use MetaLWOz, a goal-oriented dialogue dataset containing human-human dialogues in diverse domains and several tasks in each of those. The dialogues are collected in a Wizard-of-Oz method where human participants were given a problem domain and a specific task in it, and were asked to complete the task via dialogue. No domain-specific APIs or knowledge bases were available

for the participants, and in the actual dialogues they were free to use fictional names and entities in a consistent way. The dataset's statistics are shown in Table 2. All the domains available in the MetaLWOz dataset are listed in the Table 6 of the Appendix A.

## 7 Experimental setup and evaluation

Our few-shot setup is as follows. Given the target domain, we first train LAED model(s) on the MetaLWOz data — here we exclude from training every domain that might overlap with the target one. Specifically, for the *Navigation* domain in SMD, it's *Store Details*, for *Weather* it's *Weather Check*, and for *Schedule* it's *Update Calendar* and *Appointment Reminder*.

In our final setup, at Stage 1 we used a DI-VST-based LAED and a DI-VAE, both of the size $10 \times 5$.

Next, having trained and frozen Stage 1 models, we train DiKTNet on all the source domains from the SMD dataset. We use a random sample of the target domain utterances together with their contexts and KB info, varying the amount of those from 1% to 10% of all available target data.

For the NLU_ZSDG setup, we annotated all available SMD data and randomly selected a subset of 1000 utterances from each source domain, and 200 utterances from the target domain. For source domains, this number amounts to roughly a quarter of all available training data — we chose it in order to make use of as much annotated data as possible while keeping the domain description task secondary. For the target domain, we made sure to keep under roughly the same in-domain data requirements as the ZSDG baseline.

For evaluation, we follow the approach of Zhao and Eskénazi (2018) and report BLEU and Entity F1 scores. Given the non-deterministic nature of our training setup, we report means and variances of our results over 10 runs with different random seeds.

We also perform an additional evaluation of DiKTNet's performance with extended amounts of target data and compare it to the original Key-Value Retrieval Network (***KVRet***) by Eric et al. (2017) which was originally trained with all the available data. In this case we average BLEU scores across all 3 SMD domains in order to be consistent with the form the corresponding results are presented in the original paper.

We train our models with the Adam optimizer (Kingma and Ba, 2014) with learning rate 0.001. Our hierarchical models' utterance encoder is an LSTM cell (Hochreiter and Schmidhuber, 1997) of size 256, and the dialog-level encoder is a GRU (Cho et al., 2014) of size 512.

## 8 Results and discussion

Our results are shown in Table 1 — our objective here is maximum accuracy with minimum training data required.

### 8.1 Results for the few-shot setup

It can be seen that few-shot models with LAED representation are the best performing models for this objective. While improvements upon ZSDG can already be seen with simple HRED in a few-shot setup, the use of the LAED representation and domain-general ELMo encoding helps significantly reduce the amount of in-domain training data needed: at 1% of in-domain dialogues, we see that DiKTNet consistently and significantly improves upon ZSDG in every domain. In SMD, with its average dialogue length of 5.25 turns, 1% of training dialogues amounts to approximately 40 in-domain training utterances. In contrast, the ZSDG setup used approximately 150 training utterance-annotation pairs for each domain, including the target one, totalling about *450 annotated* utterances.

Although in our few-shot approach we use full in-domain dialogues, we end up having significantly less in-domain training data, with the crucial difference that none of those has to be annotated for our approach. Therefore, the method we introduced attains state-of-the-art in both accuracy and data-efficiency.

In turn, the results of the ZSDG_NLU setup demonstrate that single utterance annotations, if not domain-specific and produced by human experts, don't provide as much signal as full dialogues, even without annotations at all. Even the significant number of such annotated utterances per domain didn't make a difference in this case.

We would also like to point out that, as can be seen in the table, our results have quite high variance — the main source of it is the nature of our training/evaluation setup where we average over 10 runs with 10 different sets of seed dialogues. However, in the majority of cases with comparable means, DiKTNet has a lower variance than

| Domain | | Context | Gold response | Predicted response |
|---|---|---|---|---|
| schedule | <usr> | Remind me to take my pills | Ok setting your medicine appointment for 7pm | Okay, setting *a reminder to take your pills at 7 pm.* |
| | <sys> | What time do you need to take your pills? | | |
| | <usr> | I need to take my pills at 7 pm. | | |
| navigate | <usr> | Find the address to a hospital | Have a good day | *No problem.* |
| | <sys> | Stanford Express Care is at 214 El Camino Real. | | |
| | <usr> | Thank you. | | |
| weather | <usr> | What is the weather forecast for the weekend? | For what city would you like to know that? | For what city would you like *the weekend forecast for?* |

Table 4: DiKTNet's selected responses

| |
|---|
| Where can I go shopping? |
| Where does my friend live? |
| Where can I get Chinese food? |
| Where can I go to eat? |
| Can you please take me to a coffee house? |
| I'd like to set a reminder for my meeting at 2pm later this month please. |
| What is the time and agenda for my meeting, and who is attending? |
| Schedule a lab appointment with my aunt for the 7th at 1pm. |
| Schedule a calendar reminder for yoga with Jeff at 6pm on the 5th. |
| Car I'm desiring to do some shopping: which one is it the nearest shopping ... |
| ... center? Anything within 4 miles? |
| Get the address to my friend's house that i could get to the fastest |
| Car I need to get to my friends house, it should be within 4 miles from here |

Table 5: Selected clusters of utterances sharing the same LAED codes

the alternative models at the same percentage of seed data. And in the extreme case with 1% target data, DiKTNet improves on all the other models in terms of both means and variances.

### 8.2 Discussion of the latent representations

The comparison of the setups with different latent representations also gives us some insight: while the VAE-powered HRED model improves on the baseline in multiple cases, it lacks generalization potential compared to the LAED setup. The reason for that might be inherently more stable training of LAED due to its modified objective function which in turn results in a more informative representation providing better generalization. In order to have a glimpse into the LAED-produced clustering, in Table 5 we present a snippet of the utterance clusters sharing the same, most frequent latent codes throughout the dataset (the clustering is obtained with LAED model trained on every domain but 'Store details', i.e. the one for the evaluation on 'Navigate' SMD domain). From this snippet, it can be seen that those clusters work well for domain separation, as well as capturing dialogue intents.

### 8.3 Results with extended data

We performed an additional experiment with extended target data (see Figure 3 of Appendix A). It showed that DiKTNet, when trained with as little as 5% of target data, can outperform a KVRet trained using the entire dataset. Furthermore, with 50% of the target data, DiKTNet becomes more than twice as good as KVRet in terms of overall language generation.

However, goal-oriented metrics such as Entity F1 are more challenging to bootstrap. As such, DiKTNet outperforms KVRet on 'Weather' domain starting at 10% of the target data, but only has a trend on narrowing down the performance gap with KVRet on 'Navigate', and certainly needs more training data in 'Schedule' domain.

The explanation to that might be that most of the dialogue entities come from the KB snippets which are the least represented resource in our setup. They aren't available in MetaLWOz, and in SMD, KB snippets share little in common across domains. Therefore, in order to increase Entity F1, KB information should be directly copied to the output more efficiently — and increasing the robustness of the copy-augmented decoder is one of our future research directions.

### 8.4 Discussion of the evaluation metrics

We use BLEU as one of the main evaluation metrics in this paper — we do it in order to fully conform with the setup of Zhao and Eskénazi (2018) which we base our work on. But while being widely adopted as a general-purpose language generation metric, BLEU might not be sufficient in the dialogue settings (see Novikova et al. (2017) for a review). Specifically, we have observed several cases where the model would produce an overall grammatical response with the correct dialogue intent (e.g. "You are welcome! Anything else?"), but BLEU would output a lower score for it due to word mismatch (e.g. "You're welcome!"; see more examples in Table 4). This is a general issue in dialogue model evaluation since the variability of possible responses equivalent in meaning is very high in dialogue. In future work, we will put more emphasis on the meaning of utterances, for example by incorporating external dialogue act tagging resources in the evaluation setup which, together with general language generation metrics like perplexity, can make for more robust evaluation criteria than word overlap.

## 9 Conclusion and future work

In this paper, we have introduced DiKTNet, a model achieving state-of-the-art dialogue generation performance in a few-shot setup, without using any annotated data. By transferring latent dialogue knowledge from multiple sources of varying generality, we obtained a model with superior generalization to an underrepresented domain.

Specifically, we showed that our few-shot approach achieves state-of-the art results on the Stanford Multi-Domain dataset while being more data-efficient than the previous best model, by requiring significantly less data none of which has to be annotated.

While being state-of-the-art, the accuracy scores themselves still suggest that our technique is not ready for immediate adoption for real-world production purposes, and the task of few-shot generalization to a completely new dialogue domain remains an area of active research. In our own future work, we will try and find ways to improve the unsupervised representation (Shi et al., 2019) in order to increase the transfer potential. We will also explore ways to enable more efficient copying from the input which is crucial for correctly handling entities and therefore attaining high goal-oriented performance of the system.

Apart from that, we will consider alternative evaluation criteria to account for rich surface variability of natural speech.

## References


Roi Blanco, Giuseppe Ottaviano, and Edgar Meij. 2015. Fast and space-efficient entity linking in queries. In *Proceedings of the Eight ACM International Conference on Web Search and Data Mining*, WSDM 15, New York, NY, USA. ACM.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, pages 10–21.

Angel X. Chang and Christopher D. Manning. 2012. Sutime: A library for recognizing and normalizing time expressions. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation, LREC 2012, Istanbul, Turkey, May 23-25, 2012*, pages 3735–3740.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Mihail Eric, Lakshmi Krishnan, Francois Charette, and Christopher D. Manning. 2017. Key-value retrieval networks for task-oriented dialogue. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue, Saarbrücken, Germany, August 15-17, 2017*, pages 37–49.

A. Eshghi, M. Purver, and Julian Hough. 2011. Dylan: Parser for dynamic syntax. Technical report, Queen Mary University of London.

Arash Eshghi, Igor Shalyminov, and Oliver Lemon. 2017. Interactional Dynamics and the Emergence of Language Games. In *Proceedings of the ESSLLI 2017 workshop on Formal approaches to the Dynamics of Linguistic Interaction*, Barcelona.

Jenny Rose Finkel, Trond Grenager, and Christopher D. Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL 2005, 43rd Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, 25-30 June 2005, University of Michigan, USA*, pages 363–370.

Milica Gasic, Nikola Mrksic, Lina Maria Rojas-Barahona, Pei-Hao Su, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve J. Young. 2017. Dialogue manager domain adaptation using gaussian process reinforcement learning. *Computer Speech & Language*, 45:552–569.

Matthew Henderson, Blaise Thomson, and Jason D. Williams. 2014. The third dialog state tracking challenge. In *2014 IEEE Spoken Language Technology Workshop, SLT 2014, South Lake Tahoe, NV, USA, December 7-10, 2014*, pages 324–329.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 1367–1377.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Sungjin Lee, Hannes Schulz, Adam Atkinson, Jianfeng Gao, Kaheer Suleman, Layla El Asri, Mahmoud Adada, Minlie Huang, Shikhar Sharma, Wendy Tay, and Xiujun Li. 2019. Multi-domain task-completion dialog challenge.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 3111–3119.

Nikola Mrksic, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gasic, Pei-hao Su, David Vandyke, Tsung-Hsien Wen, and Steve J. Young. 2015. Multi-domain dialog state tracking using recurrent neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*, pages 794–799.

Jekaterina Novikova, Ondrej Dusek, Amanda Cercas-Curry, and Verena Rieser. 2017. Why we need new evaluation metrics for nlg. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Aasish Pappu, Roi Blanco, Yashar Mehdad, Amanda Stent, and Kapil Thadani. 2017. Lightweight multilingual entity extraction and linking. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, WSDM 17, New York, NY, USA. ACM.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NIPS-W*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 2227–2237.

Kun Qian and Zhou Yu. 2019. Domain adaptive dialog generation via meta learning. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 2639–2649.

Lina Maria Rojas-Barahona, Milica Gasic, Nikola Mrksic, Pei-Hao Su, Stefan Ultes, Tsung-Hsien Wen, Steve J. Young, and David Vandyke. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*, pages 438–449.

Igor Shalyminov, Arash Eshghi, and Oliver Lemon. 2017. Challenging neural dialogue models with natural data: Memory networks fail on incremental phenomena. In *Proceedings of the 21st Workshop on the Semantics and Pragmatics of Dialogue (SemDial 2017 - SaarDial)*, Barcelona.

Weiyan Shi, Tiancheng Zhao, and Zhou Yu. 2019. Unsupervised dialog structure learning. *Proceedings of NAACL*.

Stefan Ultes, Pawel Budzianowski, Iñigo Casanueva, Lina Maria Rojas-Barahona, Bo-Hsiang Tseng, Yen-Chen Wu, Steve J. Young, and Milica Gasic. 2018. Addressing objects and their relations: The conversational entity dialogue model. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue, Melbourne, Australia, July 12-14, 2018*, pages 273–283.

Vladimir Vlasov, Akela Drissner-Schmid, and Alan Nichol. 2018. Few-shot generalization across dialogue tasks. *CoRR*, abs/1811.11707.

Tsung-Hsien Wen, Yishu Miao, Phil Blunsom, and Steve J. Young. 2017. Latent intention dialogue models. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 3732–3741.

Jason D. Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 665–677.

Tiancheng Zhao and Maxine Eskénazi. 2018. Zero-shot dialog generation with cross-domain latent actions. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue, Melbourne, Australia, July 12-14, 2018*, pages 1–10.

Tiancheng Zhao, Kyusong Lee, and Maxine Eskénazi. 2018. Unsupervised discrete sentence representation learning for interpretable neural dialog generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 1098–1107.

# Multi-Granularity Representations of Dialog

**Shikib Mehri and Maxine Eskenazi**
Language Technologies Institute, Carnegie Mellon University
{amehri,max+}@cs.cmu.edu

## Abstract

Neural models of dialog rely on generalized latent representations of language. This paper introduces a novel training procedure which explicitly learns multiple representations of language at several levels of granularity. The multi-granularity training algorithm modifies the mechanism by which negative candidate responses are sampled in order to control the granularity of learned latent representations. Strong performance gains are observed on the next utterance retrieval task using both the MultiWOZ dataset and the Ubuntu dialog corpus. Analysis significantly demonstrates that multiple granularities of representation are being learned, and that multi-granularity training facilitates better transfer to downstream tasks.

## 1 Introduction

Producing generalized representations of language is a well-studied problem in natural language processing (NLP) (Montague, 1973; Davidson and Harman, 2012). Neural models typically encode an input into a latent vector, which is then used by upper layers. As such, improving the quality or generality of the learned representations will typically improve performance on the final task due to the increased representative power of the model.

Constructing meaningful representations of dialog is challenging. To effectively represent the dialog context, a latent dialog representation must contain the information necessary to (1) estimate a belief state over user goals (Williams et al., 2013), (2) track entity mentions (Zhao et al., 2017), (3) resolve anaphora co-references (Mitkov, 2014), (4) model the communicative purpose of an utterance (Core and Allen, 1997) and (5) resolve ambiguity in natural language. A large focus area of dialog research is the development of neural architectures which learn effective representations of the input (Zhou et al., 2016; Wu et al., 2016; Zhou et al., 2018). With the goal of training a model for next utterance retrieval, Zhou et al. (2018) use a deep self-attention network to produce a representation of each utterance within a dialog and follow it with an attention between utterances and 3-D convolutional layers.

Recent work has explored the use of large-scale self-supervised pre-training on very large corpora (Kiros et al., 2015; Peters et al., 2018; Devlin et al., 2018; Radford et al., 2018) as a means of improving natural language representations. These pre-trained models have yielded state-of-the-art results on several downstream NLP tasks (Wang et al., 2018): text classification, natural language inference, and question answering. Though such methods have proven useful across several downstream tasks (Wang et al., 2018), using them for dialog requires expensive fine-tuning of the complex models (Dinan et al., 2019; Alberti et al., 2019). The need for this fine-tuning is due to the pre-training procedure. First, the domain and style of dialog corpora differ significantly from the majority of the data used during pre-training. This necessitates fine-tuning in order to adapt the representations to more varied input. Second, the pre-trained representations, which are all obtained through various language modelling objectives, do not necessarily capture properties of dialog at several levels of granularity (e.g., belief state, entities, co-references, high-level user goals).

Though large-scale pre-training improves the strength and generality of latent representations, this effect is minimized when transferring to dialog tasks or out-of-domain data. To this end, this paper explores an alternate mechanism of learning strong and general representations for the task of next utterance retrieval (Lowe et al., 2015). We propose Multi-Granularity Training (MGT), which simultaneously trains multiple levels of representation. It later combines these latent repre-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1752–1761, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

sentations to obtain more general models of dialog. Different granularities of representation capture different properties of the input. For example, a high-granularity representation will capture specific words and entities mentioned in the dialog context. A low-granularity representation will instead capture more abstract properties of the dialog, such as the domain of the conversation or the high-level user goal. MGT combines representations at several levels of granularity, resulting in stronger and more general representations of dialog. The strength of representations is a consequence of learning the dedicated representations at each level of granularity. The generality results from learning several diverse representations across multiple granularities, thereby encompassing a wider amount of information. Since the representations are learned on dialog data and for the final task, this method does not suffer from the aforementioned shortcomings of pre-training.

The specific MGT procedure is motivated by the fact that observing different negative examples during training results in different representations. A model trained to select the correct response out of a set of lexically similar candidates will likely learn fine-grained representations of each word in an effort to identify minute differences between the candidates. On the other hand, a model trained to select a response from a set of topically diverse candidates will likely learn broader and more abstract representations of each utterance. Typically, negative examples are randomly sampled which results in learned representations that fit the average training example. MGT relies on an algorithm for controlled sampling of negative candidate responses, which allows for the construction of multiple training sets in order to learn multiple levels of granularity.

MGT is agnostic to the underlying model architecture. Though the majority of experiments in this paper are carried out with a dual encoder (Lowe et al., 2015) as the base model, MGT is also applied on top of Deep Attention Matching networks (Zhou et al., 2018) and obtains strong performance gains.

MGT is evaluated using the MultiWOZ dataset (Budzianowski et al., 2018) and the Ubuntu dialog corpus (Lowe et al., 2015) to train models for next utterance retrieval. Results show that MGT obtains better performance than ensembling (Perrone and Cooper, 1992) multiple baseline models. At the same time, it also serves as a better downstream representation of dialogs. The contributions of this paper are: (1) a training procedure which learns multiple granularities of latent representations for a task, (2) improved performance on next utterance retrieval across two diverse datasets, (3) an analysis significantly demonstrating that multiple granularities of representation have indeed been learned.

## 2 Related Work

This section discusses two areas of related work: language representations and the next utterance retrieval task.

### 2.1 Language Representations

Recent work has focused on improving latent representations of language through the use of large-scale self-supervised pre-training on very large corpora. Kiros et al. (2015) trains a sequence-to-sequence model (Sutskever et al., 2014) to predict the surrounding sentences, and uses the final encoder hidden state as a generic sentence representation. ELMo (Peters et al., 2018) trained a bi-directional language model on a large corpus in order to obtain strong contextual representations of words. OpenAI's GPT (Radford et al., 2018) produces latent representations of language by training a large transformer (Vaswani et al., 2017) with a language modelling objective. Devlin et al. (2018) further improves on this line of research by introducing the masked language modelling objective and a multi-tasking pre-training loss. Each of these methods has obtained state-of-the-art results on the GLUE benchmark (Wang et al., 2018), suggesting that they are strong and general representations of language.

These pre-trained representations of language have been applied to numerous tasks. Of particular interest are applications of these representations to dialog tasks. As part of the 2nd ConvAI challenge (Dinan et al., 2019), the best performing models on both human and automated evaluations (Wolf et al., 2019) were fine-tuned versions of OpenAI's GPT (Radford et al., 2018). Despite strong performance gains, transferring OpenAI's GPT required fine-tuning the full model because the dialog data was in a different domain and required different information to be contained in the representations. Recently, Mehri et al. (2019) introduce several dialog specific pre-training objec-

tives that obtain strong performance gains across multiple downstream dialog tasks.

### 2.2 Next Utterance Retrieval

Lowe et al. (2015) construct Ubuntu, the largest retrieval corpus for dialog, and present the dual encoder architecture as a baseline architecture. Kadlec et al. (2015) present several strong baseline architectures for this dataset. Zhou et al. (2016) present the Multiview architecture which, with the aim of constructing broader representation, learns both word-level representations and utterance-level representations. Sequential Matching Networks (SMN) (Wu et al., 2016) represent each utterance in the dialog context and construct segment-segment matching matrices between the response and each utterance in the context. Deep Attention Matching (DAM) (Zhou et al., 2018) uses deep transformers (Vaswani et al., 2017) to construct representations of each utterance in a dialog context, followed by cross-attention and convolutional layers.

Previous work on next utterance retrieval has proposed architectural modifications in an effort to improve the representative powers of the models. This paper presents a training algorithm applicable to any neural architecture, which explicitly forces the model to learn different granularities of representation.

## 3 Methods

This section describes three methods used for next utterance retrieval: a strong baseline dual encoder architecture, an ensemble of dual encoders, and an ensemble of dual encoders with multi-granularity training.

### 3.1 Dual Encoder

Given a dialog context, next utterance retrieval selects the correct response from a set of $k$ candidates. The retrieval baselines presented by Kadlec et al. (2015) first encode the dialog context and a candidate response. Then they use the product of the latent representations to output a probability. This baseline architecture consists of two encoders, one to encode the context and one for the response.

Previous approaches using Ubuntu were trained for binary prediction (i.e., predict the probability of a particular response), and used during testing to select from a candidate set. To mitigate the discrepancy between training and testing, our baseline is *trained* to select the correct response from a candidate set. Since the Ubuntu training set consists of 0/1 labels, the training set was modified by considering only the positive-labeled examples, and uniformly sampling $k-1$ negative candidates.

Let $c_{1,\dots,N}$ denote the words of the dialog context, $r^i_{1,\dots,M_i}$ denote the words of the $i$-th candidate response and $r_{gt}$ denote the ground-truth response. Given $f_c$, the LSTM encoder of the context, and $f_r$, the LSTM encoder of the candidate responses, the forward propagation of the dual encoder is described by:

$$\mathbf{c} = f_{\mathrm{c}}(c_i) \quad i \in [1, N] \tag{1}$$

$$\mathbf{r_i} = f_{\mathrm{r}}(r^i_j) \quad j \in [1, M_i] \tag{2}$$

$$\mathbf{r_{gt}} = f_{\mathrm{r}}(r^{gt}_j) \quad j \in [1, M_{gt}] \tag{3}$$

$$\alpha_{gt} = \mathbf{c}^T \mathbf{r_{gt}} \tag{4}$$

$$\alpha_i = \mathbf{c}^T \mathbf{r_i} \tag{5}$$

The final loss function is:

$$\begin{aligned}\mathcal{L} &= -\log p(r^i_{1,\dots,M_i} | c_{1,\dots,N}) \\ &= -\log\left(\frac{\exp(\alpha_{gt})}{\exp(\alpha_{gt}) + \sum_{j=1}^{K}\exp(\alpha_j)}\right)\end{aligned} \tag{6}$$

### 3.2 Ensemble of Dual Encoders

Ensembling multiple models (Perrone and Cooper, 1992) has been empirically shown to improve performance, since it maintains a low model bias while significantly reducing the model variance. In ensembling, multiple models are trained and their predictions are averaged during inference. Specifically, if $\alpha^l$ denotes the output of model $l \in [1, L]$, the output probability is defined as:

$$p(r^i_{1,\dots,M_i} | c_{1,\dots,N}) = \frac{1}{L}\sum_{l=1}^{L}\frac{exp(\alpha^l_i)}{\sum_{j=1}^{K} exp(\alpha^j_i)} \tag{7}$$

Since ensembling reduces the model variance while maintaining low bias, it is most effective when the models are diverse and each model excels at a particular type of input. In typical ensemble training, the different models are either obtained through different random initializations or at different checkpoints from the same training run. In such an approach, there is no mechanism which explicitly enforces diversity between the models.

### 3.3 Multi-Granularity Training

During baseline model training, the negative response candidates were uniformly sampled from $R$, the set of all responses in the training set. MGT is proposed in an effort to explicitly model different granularities of representation through a controlled method of sampling negative candidates.

Consider a training corpus consisting of a set of dialog contexts and ground-truth responses, $T = (C, R^{gt})$. In the baseline training, $k-1$ negative response candidates are uniformly sampled from the set of all responses, $R$:

$$T_i = (C_i, R_i^{gt}, [N_{i,1}, N_{i,2}, \ldots, N_{i,k-1}]) \quad (8)$$
$$\forall j \in [1, k-1] \quad N_{i,j} \sim \text{Uniform}(R)$$

MGT is motivated by the idea that observing different types of negative candidate response sets will result in different representations. Negative candidates which are lexically similar to the ground truth response should result in models that carefully consider each word in order to produce fine-grained representations and identify minute differences between candidate responses. On the other hand, very semantically distant candidate responses should result in very broad and abstract representations of language. While there may be many methods of sampling negative responses to influence what the model learns, this paper focuses on using the semantic similarity of the candidate responses as a means of controlling the granularity of learned representations.

Given the LSTM response encoder, $f_r$, the measure of semantic similarity is defined as:

$$\mathbf{r_i} = f_r(R_{i,j}) \quad j \in [1, M_i] \quad (9)$$
$$\mathbf{r_k} = f_r(R_{k,j}) \quad j \in [1, M_k] \quad (10)$$
$$d(R_i, R_k) = \frac{\mathbf{r_i}^T \mathbf{r_k}}{||\mathbf{r_i}|| \cdot ||\mathbf{r_k}||} \quad (11)$$

This approach relies on a cosine-similarity as a measure of semantic distance between dialog utterances. While not a perfect measure, for the purposes of the MGT algorithm it appears to be a sufficient measure. Since the training algorithm groups together similarly distant negative candidates, it is robust to noise in the measure of semantic distance. Future work may explore whether a better distance measure improves the MGT algorithm.

A distance matrix $D$ is constructed between all of the responses in $R$, such that $D_{i,j} = d(R_i, R_j)$. The objective of MGT is to train $L$ models at $L$ different levels of granularity. For a particular response $R_i$, rather than sampling negative candidates from the entire set of $R$, the set of responses $R$ is split into $L$ segments based on distance from $R_i$. Define a function $\mathtt{b}(D_i, l)$ which considers a list of distances and returns the maximum distance in the $l$-th segment of a total of $L$ segments. This is equivalent to sorting $D_i$ and taking the $\left(|R| \times \frac{l}{L}\right)$-th value.

The distance matrix, $D$, is used to segment the set of potential negative candidates, $R$, for each training example $(C_i, R_i^{gt})$, into $L$ buckets: $P_i^1, \ldots, P_i^L$. Given the definition of segmentation provided above, $P_i^1$ will consist of responses that are strictly closer (as defined by $d$) to $R_i$ than the responses in $P_i^2$. When training the $l$-th model at the $l$-th level of granularity, the negative responses for $R_i$ are sampled from $P_i^l$ rather than $R$. $P_i^l$ is constructed using $\mathtt{b}(D_i, l)$, which was defined to return the maximum value in the $l$-th segment.

This method is used to construct $L$ different training corpora, $T^1, \ldots, T^L$. A particular $T^l$ is constructed as follows:

$$T_i^l = (C_i, R_i^{gt}, [N_{i,1}^l, \ldots, N_{i,k-1}^l]) \quad (12)$$
$$P_i^l = \{r \in R \mid d(R_i, r) \in (\mathtt{b}(D_i, l-1), \mathtt{b}(D_i, l)\}$$
$$\forall j \in [1, k-1] \quad N_{i,j}^l \sim \text{Uniform}(P_i^l)$$

After the $L$ different training corpora, $L$ different models are trained. Models trained on closer candidate sets should learn more granular representations while models trained on more distant candidate sets should learn more abstract representations of dialog. Upon obtaining $L$ different models, the output probability is produced by the ensembling method described in Equation 7.

## 4 Experiments

This section describes the datasets and presents experimental procedures aimed at evaluating the different approaches to next utterance retrieval.

### 4.1 Datasets

Two retrieval corpora, MultiWOZ (Budzianowski et al., 2018) and Ubuntu (Lowe et al., 2015) were used. MultiWOZ contains task-oriented conversations between a tourist and a Wizard-of-Oz, while

Ubuntu contains both open-domain and technical dialog snippets collected from Internet Relay Chat (IRC). The diversity of these two datasets provides insight into the general applicability of MGT.

#### 4.1.1 MultiWOZ

The MultiWOZ dataset (Budzianowski et al., 2018) was converted into a retrieval corpus. MultiWOZ contains 8422 dialogs for training, 1000 for validation and 1000 for testing. There are 20 candidate responses for each dialog context.

#### 4.1.2 Ubuntu Dialog Corpus

The original Ubuntu corpus (Lowe et al., 2015) has 1,000,000 training examples. Typical interactions include individuals asking for technical assistance in a conversational manner. The subject of conversation is not explicitly bounded and may be any topic. As described in Section 3.1, the training corpus is modified in order to train as a retrieval task rather than as a binary prediction task. Negative training examples (500,127) are filtered out. The size of the new training dataset is 499,873. There are a total of 10 candidate responses for each context. The validation and test sets remain unchanged, with 19,561 validation examples and 18,921 test examples.

### 4.2 Experimental Setup

Unless otherwise specified, the size of ensembles and the number of models in MGT is $L = 5$. For MGT, the highest performing checkpoint at each granularity is selected using the validation score. For the ensemble method, the top performing checkpoints are selected from a single run.

#### 4.2.1 MultiWOZ Setup

Two distinct encoders are trained, one to encode the dialog context and the other for the candidate responses. Each encoder is a single layer, unidirectional LSTM with an embedding dimension of 50 and a hidden size of 150. These hidden sizes match the best performing hyperparameters identified by Budzianowski et al. (2018). The Adam optimizer (Kingma and Ba, 2015) with a learning rate of 0.005 is used to train the model for 20 epochs. The vocabulary is 1261 words, the batch size is 32, and gradients are clipped to 5.0. A checkpoint is saved after each epoch, and the best checkpoint is selected using performance on the validation set.

#### 4.2.2 Ubuntu Setup

Each encoder is a single layer, uni-directional LSTM with an embedding dimension of 300 and a hidden size of 150. The Adam optimizer (Kingma and Ba, 2015) with a learning rate of 0.005 is used to train the model for 20 epochs. The vocabulary is 10002 words, the batch size is 128, and gradients are clipped to 5.0. Only the last 160 words of each dialog context are used. The word embeddings are initialized with pre-trained GloVe embeddings (Pennington et al., 2014). A checkpoint is saved after each epoch, and the best checkpoint is selected using performance on the validation set.

#### 4.2.3 Deep Attention Matching Experiment

MGT is a training procedure which is agnostic to the underlying model architecture. Though the majority of the experiments presented in this paper use the dual encoder architecture (Lowe et al., 2015), MGT is applied on top of the state-of-the-art architecture for Ubuntu: the Deep Attention Matching Network (DAM) (Zhou et al., 2018). When applying MGT to DAM, the sampling of negative candidates is done using the baseline dual encoder architecture.

### 4.3 Multi-Granularity Training Example

Table 1 shows an example dialog context, ground-truth response and the negative candidate responses sampled at several levels of cosine distance, as per Equation 11. These negative candidate responses are retrieved by MGT's sampling algorithm, and are used to train multiple models at different levels of granularity.

The negative candidates for the higher granularity models are much closer to the ground-truth response than the candidates for the lower granularity models. As such, models trained at higher granularities will learn more granular representations of dialog in order to identify minute differences between responses. Examples for lower granularity models are more distant in meaning, and models may instead learn to represent higher-level attributes of the dialog.

## 5 Results

Multi-granularity training is proposed in order to learn strong and general latent representations of dialog. To evaluate the strength and generality of the learned representations, experiments are conducted to evaluate three different properties of

| |
|---|
| **Dialog Context:** |
| USER: i need an expensive place to dine in the center of town . |
| SYS: what type of food are you wanting at this restaurant ? |
| USER: type of food does not matter to me . |
| **Ground-Truth Response:** |
| SYS: yes there are 33 such restaurant -s . would you like me to book 1 for you ? |
| **Closest Neg. Cand. (for Highest Granularity Model):** |
| SYS: i have 5 indian restaurant -s in the west . if i may , might i suggest the rice boat . i can reserve a table for you if you would like . |
| **2nd Closest Neg. Cand. (for 2nd Highest Granularity Model):** |
| SYS: how about the avalon ? |
| **3rd Closest Neg. Cand. (for 3nd Highest Granularity Model):** |
| SYS: i would be happy to book that for you .are you travelling alone ? |
| **2nd Furthest Neg. Cand. (for 2nd Lowest Granularity Model):** |
| SYS: i apologize , there s been a misunderstanding . before we can book your taxi , we need to know which college . there are 18 colleges . what area are you looking for ? |
| **Furthest Granularity Neg. Cand (for Lowest Granularity Model):** |
| SYS: i need to know when you want to leave and arrival time in order for me to order you a taxi . |

Table 1: An example dialog context from the training set. Along with the ground-truth response, negative candidates sampled at five different levels of semantic distance are shown. The retrieval models are trained to differentiate between the ground-truth response and the different negative candidates.

MGT: (1) improved performance on the task of next utterance retrieval, (2) explicit modelling of different granularities, and (3) improved generality and transferability to other dialog tasks.

### 5.1 Next Utterance Retrieval

Next utterance retrieval is reliant on latent representations of dialog. Several experiments are conducted to evaluate whether MGT improves the representative power of models and results in better performance on the task of next utterance retrieval. MGT is expected to outperform standard ensembling, since MGT explicitly models multiple granularities and trains more diverse models. The performance of MGT is evaluated using both MultiWOZ (Budzianowski et al., 2018) and Ubuntu (Lowe et al., 2015). Experiments are conducted using two different underlying architectures, a dual encoder baseline (Lowe et al., 2015) and a Deep Attention Matching network (Zhou et al., 2018).

#### 5.1.1 MultiWOZ

Performance on the MultiWOZ retrieval task is evaluated with mean reciprocal rank (MRR), and Hits@1 (H@1). Mean reciprocal rank is defined as follows:

$$MRR = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{rank_i} \tag{13}$$

Hits@1 is equivalent to accuracy. It measures how often the ground-truth response is selected from the $K = 20$ candidates.

| Model Name | MRR | Hits@1 |
|---|---|---|
| Dual Encoder | 79.55 | 66.13% |
| Ensemble (5) | 81.53 | 69.47% |
| Multi-Granularity (5) | **82.74** | **72.18%** |

Table 2: Performance on MultiWOZ. MGT is compared to a baseline dual encoder, and an ensemble of dual encoders with an identical number of parameters. All bold-face results are statistically significant to $p < 0.01$.

The results in Table 2 demonstrate the strong performance gains obtained with MGT. With $L = 5$ granularities, MGT outperforms a similarly sized ensemble of dual encoders. These results demonstrate that explicitly enforcing the policy that makes models learn multiple granularities of representation improves the representative power and performance on next utterance retrieval.

| Model Name | MRR | $R_{10}@1$ | $R_2@1$ |
|---|---|---|---|
| **Previous Research** | | | |
| Dual Encoder (Lowe et al., 2015) | - | 63.8 | 90.1 |
| MV-LSTM (Pang et al., 2016) | - | 65.3 | 90.6 |
| Match-LSTM (Wang and Jiang, 2016) | | 65.3 | 90.4 |
| Multiview (Zhou et al., 2016) | - | 66.2 | 90.8 |
| DL2R (Yan et al., 2016) | - | 62.6 | 89.9 |
| SMN (Wu et al., 2016) | - | 72.6 | 92.6 |
| DAM (Zhou et al., 2018) | - | **76.7** | **93.8** |
| **Dual Encoder Experiments** | | | |
| Dual Encoder (Lowe et al., 2015) | 76.84 | 63.6 | 90.9 |
| Ensemble (5) | 78.91 | 66.9 | 91.7 |
| Multi-Granularity (5) | **80.10** | **68.7** | **91.9** |
| **Deep Attention Matching Experiments** | | | |
| DAM (Zhou et al., 2018) (re-trained) | 83.74 | 74.54 | 93.08 |
| Ensemble (5) | 84.03 | 74.95 | 93.27 |
| Multi-Granularity (5) | **84.26** | **75.30** | **93.45** |

Table 3: Results for next utterance retrieval on the Ubuntu dialog corpus. This table shows previous work, and experimental results with two underlying architectures: a dual encoder model and Deep Attention Matching networks. The results shown in the DAM experiments section are performed with the open-sourced implementation of Zhou et al. (2018), which obtains slightly worse performance than they report. All bold-face results are statistically significant to $p < 0.01$.

#### 5.1.2 Ubuntu

Previous research used several variations of the $R_N@k$ metric to evaluate retrieval performance on the Ubuntu dialog dataset. $R_N@k$ refers to the percentage of the time that the ground truth response was within the top-$k$ predictions for a candidate set size of $N$ utterances. $R_{10}@1$ on Ubuntu is equivalent to Hits@1 and accuracy. In addition to MRR, we report $R_{10}@1$ and $R_2@1$, top-1 accuracy with a candidate set size of 10 and 2, respectively.

MGT is applied on top of the dual encoder baseline (Lowe et al., 2015) and Deep Attention Matching networks (Zhou et al., 2018). The results shown in Table 3 show the performance of MGT using two different underlying architectures, as well as previous work. Across both base architectures, MGT outperforms ensembling. The primary difference between these two methods is that MGT explicitly ensures that several granularities of representation are learned. As such, these results reaffirm the hypothesis that learning multiple granularities of representation leads to more diverse models, and more general representations of dialog.

Even with the dual encoder as the underlying model, MGT outperforms all previous work except for Sequential Matching Networks (SMN) (Wu et al., 2016) and Deep Attention Matching networks (DAM) (Zhou et al., 2018). The Deep Attention Matching experiment performs MGT using DAM[1] as the underlying architecture. MGT has good performance improvement on top of DAM, roughly double the improvement obtained by ensembling. This suggests that MGT can be used as a general purpose training algorithm which learns multiple-granularities of representation and thereby produces stronger and more general models.

### 5.2 Explicit Granularity Modelling

Multi-granularity training learns multiple granularities of representation. However, strong performance on next utterance retrieval, does not necessarily prove that several granularities are explicitly modelled. To analyze whether the models operate at different levels of granularity, the content of the representations must be considered. The $L = 5$ trained models, each at a different granularity, have their weights frozen. These frozen

[1]It should be noted that the open-source implementation provided by Zhou et al. (2018) was used, however performance was slightly lower than the results they reported. We speculate that given a DAM implementation that matches their reported results, MGT would obtain a similarly-sized improvement (+0.76 R@1).

| Model Name | BoW (F-1) | DA (F-1) |
|---|---|---|
| Highest Abstraction | 57.00 | **19.24** |
| 2nd Highest Abs. | 57.69 | 19.14 |
| Medium | 58.49 | 18.31 |
| 2nd Highest Gran. | 58.38 | 16.88 |
| Highest Granularity | **59.43** | 15.46 |

Table 4: Results of the granularity analysis experiment. $L = 5$ models trained to capture different granularities of representation. All bold-face results are statistically significant to $p < 0.01$.

models are then used to obtain a latent representation of all the dialog contexts in MultiWOZ. A linear layer is then trained on top of these representations for a downstream task. During this training, only the weights of the linear layer are updated. This evaluates the information contained in these learned representations.

Two different downstream tasks are considered; bag-of-words prediction and dialog act prediction. Bag-of-words prediction is the task of predicting a binary vector corresponding to the words present in the last utterance of the dialog context. This task requires very granular representations of language, and therefore the models trained to capture high granularity representations should have the highest performance. Dialog act prediction is the task of predicting the set of dialog acts for the next system response. This is a high-level task that requires abstract representations of language, therefore the models with the lowest granularity should do well.

The results in Table 4 confirm the hypothesis that MGT results in models that learn different granularities of representation. It is clear that higher granularity models better capture the information necessary for the bag-of-words task, while higher abstraction (lower granularity) models better capture information for dialog act prediction.

### 5.3 Generalizability and Task Transfer

One motivation of MGT is to improve the generality of representation, and facilitate easy transfer to various tasks. Truly general representations of language would require no fine-tuning of the model, and we would only need to learn a linear layer in order to extract the relevant information from the representation. Bag-of-words prediction and dialog act prediction are again used to evaluate the ability of MGT to transfer without any fine-tuning.

The results shown in Table 5 demonstrate

| Model Name | BoW (F-1) | DA (F-1) |
|---|---|---|
| Dual Encoder | 60.13 | 19.09 |
| Ensemble (5) | 64.11 | 22.39 |
| Multi-Granularity (5) | **67.51** | **22.85** |
| Fine-tuned | 90.33 | 28.75 |

Table 5: Experimental results demonstrating performance on two downstream tasks, without any fine-tuning of the latent representations. All bold-face results are statistically significant to $p < 0.01$.

| Model Name | DA (F-1) |
|---|---|
| Random Init | 28.75 |
| Dual Encoder | 32.63 |
| Ensemble (5) | 31.71 |
| Multi-Granularity (5) | **33.46** |

Table 6: Experimental results demonstrating performance on the downstream task of dialog act prediction, when the model is fine-tuned on all available data. All bold-face results are statistically significant to $p < 0.01$.

that MGT results in more general representations of language, thereby facilitating better transfer. However, there is room for improvement when comparing to models fine-tuned on the downstream task. This suggests that additional measures can be taken to improve the representative power of these models.

The results in Table 6 demonstrate that MGT learns general representations which effectively transfer to downstream tasks, especially more difficult tasks such as dialog act prediction. Fine-tuning the latent representations learned by MGT, results in improved performance on dialog act prediction.

## 6 Conclusions and Future Work

This paper presents multi-granularity training (MGT), a mechanism for learning strong and general representations for next utterance retrieval. Through the use of a sampling algorithm to select negative candidate responses, multiple granularities of representation are learned during training. Strong performance gains are observed on the task of next utterance retrieval on both MultiWOZ and Ubuntu. Experiments show that MGT is a generally applicable training procedure which can be applied to multiple underlying model architectures. Quantitative analytic experiments demonstrate that multiple granularities of representation

are in fact being learned, and that MGT facilitates better transfer to downstream tasks both with and without fine-tuning.

There are several avenues for future work. First, this method is general and broadly applicable, which suggests that it may improve performance on other tasks and domains. A particularly interesting application would be to generalize this method to language generation tasks. Second, a useful improvement on top of MGT would be a more sophisticated method of combining the multiple granularities of representations. Third, while this paper focuses on capturing multiple representations at different levels of granularity, it would be interesting to generalize MGT to learning multiple representations along several different axes (e.g., domains, styles, intents, etc.).

## References


Chris Alberti, Kenton Lee, and Michael Collins. 2019. A bert baseline for the natural questions. *arXiv preprint arXiv:1901.08634*.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. Multiwoz-a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. *arXiv preprint arXiv:1810.00278*.

Mark G Core and James Allen. 1997. Coding dialogs with the damsl annotation scheme. In *AAAI fall symposium on communicative action in humans and machines*, volume 56. Boston, MA.

Donald Davidson and Gilbert Harman. 2012. *Semantics of natural language*, volume 40. Springer Science & Business Media.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Emily Dinan, Varvara Logacheva, Valentin Malykh, Alexander Miller, Kurt Shuster, Jack Urbanek, Douwe Kiela, Arthur Szlam, Iulian Serban, Ryan Lowe, et al. 2019. The second conversational intelligence challenge (convai2). *arXiv preprint arXiv:1902.00098*.

Rudolf Kadlec, Martin Schmid, and Jan Kleindienst. 2015. Improved deep learning baselines for ubuntu corpus dialogs. *arXiv preprint arXiv:1510.03753*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. *arXiv preprint arXiv:1506.08909*.

Shikib Mehri, Evgeniia Razumovsakaia, Tiancheng Zhao, and Maxine Eskenazi. 2019. Pretraining methods for dialog context representation learning. *arXiv preprint arXiv:1906.00414*.

Ruslan Mitkov. 2014. *Anaphora resolution*. Routledge.

Richard Montague. 1973. The proper treatment of quantification in ordinary english. In *Approaches to natural language*, pages 221–242. Springer.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016. Text matching as image recognition. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Michael P Perrone and Leon N Cooper. 1992. When networks disagree: Ensemble methods for hybrid neural networks. Technical report, BROWN UNIV PROVIDENCE RI INST FOR BRAIN AND NEURAL SYSTEMS.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Alex Wang, Amapreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform

for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Shuohang Wang and Jing Jiang. 2016. Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905*.

Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. 2013. The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 404–413.

Thomas Wolf, Victor Sanh, Julien Chaumond, and Clement Delangue. 2019. Transfertransfo: A transfer learning approach for neural network based conversational agents. *arXiv preprint arXiv:1901.08149*.

Yu Wu, Wei Wu, Chen Xing, Ming Zhou, and Zhoujun Li. 2016. Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots. *arXiv preprint arXiv:1612.01627*.

Rui Yan, Yiping Song, and Hua Wu. 2016. Learning to respond with deep neural networks for retrieval-based human-computer conversation system. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 55–64. ACM.

Tiancheng Zhao, Allen Lu, Kyusong Lee, and Maxine Eskenazi. 2017. Generative encoder-decoder models for task-oriented spoken dialog systems with chatting capability. *arXiv preprint arXiv:1706.08476*.

Xiangyang Zhou, Daxiang Dong, Hua Wu, Shiqi Zhao, Dianhai Yu, Hao Tian, Xuan Liu, and Rui Yan. 2016. Multi-view response selection for human-computer conversation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 372–381.

Xiangyang Zhou, Lu Li, Daxiang Dong, Yi Liu, Ying Chen, Wayne Xin Zhao, Dianhai Yu, and Hua Wu. 2018. Multi-turn response selection for chatbots with deep attention matching network. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1118–1127.

# Are You for Real? Detecting Identity Fraud via Dialogue Interactions

**Weikang Wang**[1,2], **Jiajun Zhang**[1,2], **Qian Li**[3], **Chengqing Zong**[1,2,4] and **Zhifei Li**[3]
[1] National Laboratory of Pattern Recognition, Institute of Automation, CAS, Beijing, China
[2] University of Chinese Academy of Sciences, Beijing, China
[3] Mobvoi, Beijing, China
[4] CAS Center for Excellence in Brain Science and Intelligence Technology, Beijing, China
{weikang.wang, jjzhang, cqzong}@nlpr.ia.ac.cn
{qli, zfli}@mobvoi.com

## Abstract

Identity fraud detection is of great importance in many real-world scenarios such as the financial industry. However, few studies addressed this problem before. In this paper, we focus on identity fraud detection in loan applications and propose to solve this problem with a novel interactive dialogue system which consists of two modules. One is the knowledge graph (KG) constructor organizing the personal information for each loan applicant. The other is structured dialogue management that can dynamically generate a series of questions based on the personal KG to ask the applicants and determine their identity states. We also present a heuristic user simulator based on problem analysis to evaluate our method. Experiments have shown that the trainable dialogue system can effectively detect fraudsters, and achieve higher recognition accuracy compared with rule-based systems. Furthermore, our learned dialogue strategies are interpretable and flexible, which can help promote real-world applications.[1]

## 1 Introduction

Identity fraud is one person using another person's personal information or combining a few pieces of real data with bogus information to deceive a third person. Nowadays, identity fraud is becoming an increasingly prevalent issue and has left many financial firms nursing huge losses. Besides, for persons whose identities have been stolen, they may receive unexpected bills and their credit will also be affected. Although identity fraud is a very serious problem in modern society, there are no effective fraud detection methods at present and little attention has been paid to this problem.

Intuitively, a simple way to detect identity fraud in loan applications is directly asking applicants about their personal information. However, as shown in Fig. 1, this method is prone to errors because fraudsters may well know the fake information. Fortunately, we find fraudsters generally are not clear about answers to questions that are related to the fake information.[2] We refer to these questions as *derived questions*, which can be constructed based on triplets where the head entity is the personal information entity. For example, the first derived question about "Nanjing University" is based on (Nanjing University, FoundedDate, 1902). In Fig. 1, the applicant claims to graduate from "Nanjing University" but can not answer derived questions about this school. This fact indicates that the applicant is likely to be a fraudster.

The **fake** personal information of a loan applicant:

| School | Company | Residence |
|---|---|---|
| Nanjing University | Baidu | No.3 Gulou Street |

*System: Which university did you graduate from?*
Applicant: Nanjing University.
*System: Which company do you work for?*
Applicant: Baidu.
*System: Where do you live?*
Applicant: No.3 Gulou Street.
*Decision: Non-Fraud (Wrong)*

*System: When was Nanjing University founded?*
Applicant: I am not sure.
*System: Which is the nearest subway station to Nanjing University?*
Applicant: I am not quite clear.
*Decision: Fraud (Correct)*

Figure 1: Dialogue examples of two possible fraud detection methods. The first one is directly asking applicants about their personal information. The second one is asking applicants about questions that are related to their personal information.

Based on the above finding, we aim to design a

[1] https://github.com/Leechikara/Dialogue-Based-Anti-Fraud

[2] This finding is based on the premise that loan applicants answer questions without any help (e.g., using automatic QA systems or information retrieval tools). In fact, this premise is reasonable in many real scenarios, such as dialogue with video calls and phone calls in which we can monitor the applicants with a camera or require them to answer questions within few seconds (e.g., 5 seconds).



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1762–1771, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

dialogue system to detect identity fraud by asking derived questions. However, there are three major challenges in achieving this goal.

First, designing derived questions requires a high-quality KG. However, owing to the sparseness problem (Ji et al., 2016; Trouillon et al., 2017) of the KG, many entities have no triplets for derived question generation. Second, randomly selecting triplets to generate questions is feasible but it is not the optimal questioning strategies to detect fraudsters. Third, because of privacy issues, evaluating anti-fraud systems with real applicants is not practical. And existing user simulation methods (Li et al., 2016; Georgila et al., 2006; Pietquin and Dutoit, 2006) do not apply to our task. Hence, how to evaluate our systems efficiently is a problem.

To address the above problems, we first complete an existing KG with geographic information in an electronic map (Section 2). In the new KG, nearly all personal information entities can find triplets for derived question generation. Then, based on the KG, we present structured dialogue management (Section 3) to explore the optimal dialogue strategy with reinforcement learning. Specifically, our dialogue management consists of (1) the *KG-based dialogue state tracker* (KG-DST) that treats embeddings of nodes in the KG as dialogue states and (2) the *hierarchical dialogue policy* (HDP) where high-level and low-level agents unfold the dialogue together. Finally, based on intuitive analysis, we find the applicants' behavior is related to some factors (Section 5.1). Thus, we introduce hypotheses to formalize the effect of these factors on the applicants' behavior and propose a heuristic user simulator to evaluate our systems.

Experiments have shown that the data-driven system significantly outperforms rule-based systems in the fraud detection task. Besides, the ablation study proves that the proposed dialogue management can improve the recognition accuracy and learning efficiency because of its ability to model structured information. We also analyze the behavior of our system and find the learned anti-fraud policy is interpretable and flexible.

To summarise, our main contributions are threefold: (1) As far as we know, this is the first work to detect identity fraud through dialogue interactions. (2) We point out three major challenges of identity fraud detection and propose corresponding solutions. (3) Experiments have shown that our approach can detect identity fraud effectively.

## 2 Knowledge Graph Constructor

There are four types of personal information in a Chinese loan application form: "School", "Company", "Residence" and "BirthPlace". To generate derived questions, we link all personal information entities to nodes in an existing Chinese KG[3] and crawl triplets that are directly related to them. However, owing to the fact that the KG is largely sparse, nearly a half of entities[4] cannot be linked. Thus we use wealthy geographic information about organizations and locations in electronic maps (e.g., Amap[5]) to complete the KG.

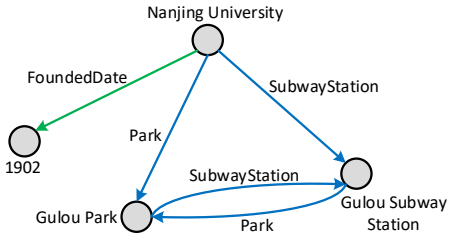


Figure 2: An example of the KG for Nanjing University. The green edge represents the triplet crawled from the existing KG and the blue edges represent the triplets generated based on a navigation electronic map.

Specifically, for each personal information entity, we first crawl its points of interest (POI[6]) within one kilometer and the POI types in the Amap. If there are multiple POI for the same type, we only keep the nearest one. Then we generate triplets in the form of ($Personal Information Entity$, $POI type$, $POI$) to indicate the fact that the nearest $POI type$ to the $Personal Information Entity$ is $POI$. Besides, for any two entities, if the distance between them is less than 100 meters, we generate two triplets to represent the bi-directional adjacency relation between them. In the end, as shown in Fig. 2, we combine triplets from the two information sources (the Chinese KG and the electronic map) to construct a new KG. In this KG, nearly all personal information entities can be linked. And for each relation[7], we design a language template for the question generation.

## 3 Dialogue System Design

The overview of our system is shown in Fig. 3. The core of the system is dialogue management

[3] https://www.ownthink.com

[4] Most of them are "Residence" and "BirthPlace".

[5] https://www.amap.com

[6] The POI are the specific locations (e.g., subway stations) that someone may find useful in navigation systems.

[7] After the data cleaning, there are 40 relations in all.

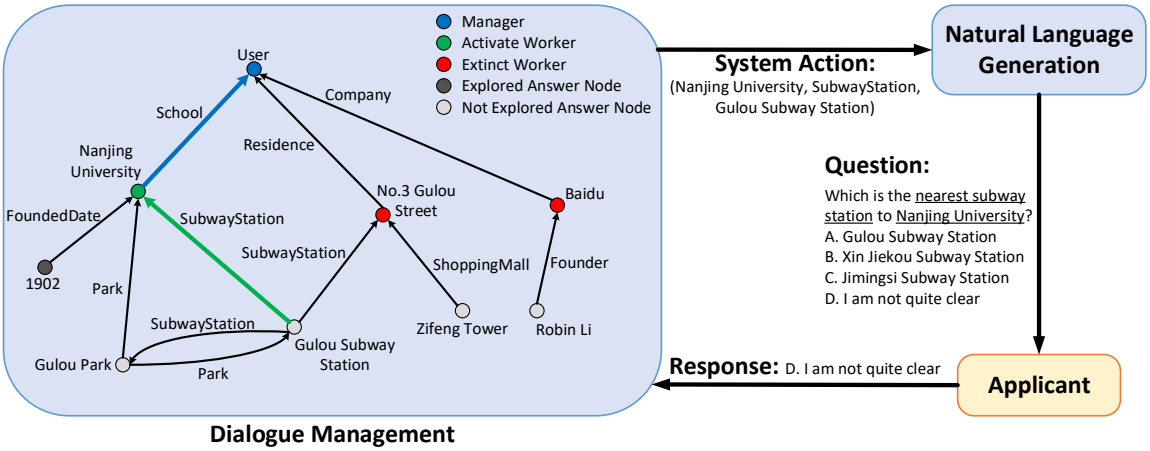


Figure 3: Overview of our approach. To build the directed graph for dialogue management, we reverse directions of all edges in the original personal KG to make the head entity read information from its tail entities. Besides, we add a special node "User" and new edges to represent the applicant's personal information. In this graph, the direction of each edge is the direction of message passing in KG-DST. The blue and green edges indicate that two agents select nodes to unfold the dialogue according to HDP.

which is organized as a directed graph $G(\mathbb{V}, \mathbb{E})$. In each turn, our system first infers dialogue states with the *KG-based dialogue state tracker* by computing embeddings of nodes. In this graph, the embedding of "User" node is the dialogue state of a high-level agent (manager), and the embeddings of nodes adjacent to "User" (named as personal information nodes) are the dialogue states of low-level agents (workers). Then our system unfolds the dialogue according to the *hierarchical dialogue policy*. Concretely, the manager first selects a personal information node (e.g., "Nanjing University") as the worker, and then the worker will select a node (e.g., "Gulou Subway Station") from its predecessors (named as answer nodes). After that, the sampled nodes of two agents form the final system action (a triplet). Next, based on the triplet and a predefined template, the natural language generation module will give a multiple-choice question to the applicant. After the applicant gives a response, the embeddings of all nodes will be updated to generate new dialogue states for the next turn.

### 3.1 KG-based Dialogue State Tracker

There are three types of nodes in $G(\mathbb{V}, \mathbb{E})$: the "User" node $v_u$, the personal information node $v_p \in \mathbb{V}_p$ and the answer node $v_a \in \mathbb{V}_a$. In the $t$-th turn, KG-DST first gives an initial embedding to $v \in \mathbb{V}_p \cup \mathbb{V}_a$. The initial embedding is the concatenation of *static features* and *dialogue features*. Then, $v$ will gather information from its predecessors $N(v)$. After multiple *message passings*, we get its final embedding $E_t(v)$. Next, $v_u$ will aggregate information from $\mathbb{V}_p$ to generate its embedding $E_t(v_u)$. Finally, $E_t(v_u)$ and $E_t(v_p)$ are the dialogue states of the manager and worker respectively.

**Static Features.** Specifically, for $v \in \mathbb{V}_p \cup \mathbb{V}_a$, the static features include the degree and type. Besides, for $v_a \in \mathbb{V}_a$, we use the "spread degree on the internet" to distinguish different answer nodes because we find there is an obvious correlation between this "spread degree" feature and applicants' behavior in our human experiments (Section 5.1). To get the "spread degree" feature, we first treat the answer node $v_a$ and its adjacent personal information node $v_p$ as the keyword[8], and then search it in the search engine. The number[9] of the retrieved results will be the "spread degree" feature of $v_a$. In the end, each static feature is encoded as a one-hot vector and they are concatenated to form a vector $S_t(v)$.

**Dialogue Features.** The dialogue features record the dynamic information of $v \in \mathbb{V}_p \cup \mathbb{V}_a$ during the dialogue. Specifically, dialogue features include whether the node has been explored by the manager or workers and whether the node appeared in the system action of the last turn. In addition, for $v_p \in \mathbb{V}_p$, the dialogue features include the interaction turns of the corresponding worker and the number of correctly/incorrectly answered questions about $v_p$. For $v_a \in \mathbb{V}_a$, the dialogue features

[8] In fact, the keyword is the head entity and tail entity of a triplet. For example, for the answer node "1902", the keyword is "Nanjing University 1902".

[9] If there are multiple keywords for an answer node (e.g., "Gulou Subway Station"), we take the average.

include whether applicants know $v_a$ is the answer to a derived question. Similarly, dialogue features will be encoded as a one-hot vector $D_t(v)$.

**Message Passing.** In Fig. 3, the applicant does not know "Gulou Subway Station" is the nearest subway station to "Nanjing University". In such case, the personal information about "School" may be fake. Besides, for another question "What's the nearest park to Nanjing University?", the applicant may not know the answer because the distance between "Gulou Park" and "Gulou Subway Station" is less than 100 meters. Thus, we want the known information of "Gulou Subway Station" to be sent to its successors.

Specifically, for $v \in \mathbb{V}_p \cup \mathbb{V}_a$, we compute its embedding recursively as follows:

$$E_t^k(v) = \max_{v' \in N(v)} \tanh\left(\mathbf{W}^k E_t^{k-1}(v')\right) \quad (1)$$

where $E_t^k(v)$ is the depth-$k$ node embedding in the $t$-th turn, $N(v)$ denotes the set of nodes adjacent to $v$, $\mathbf{W}^k$ is the parameter in the $k$-th iteration and the aggregate function is the element-wise max operation. The final node embedding is the concatenation of embeddings at each depth:

$$E_t(v) = \left[E_t^0(v), ..., E_t^K(v)\right] \quad (2)$$

where $E_t^0(v) = [S_t(v), D_t(v)]$ and $K$ is a hyper-parameter.

After getting the embedding of $v_p \in \mathbb{V}_p$, we compute the embedding of $v_u$ by aggregating information from $\mathbb{V}_p$:

$$E_t(v_u) = \max_{v_p \in \mathbb{V}_p} \tanh\left(\mathbf{W}^p E_t(v_p)\right) \quad (3)$$

where $\mathbf{W}^p$ is the parameter.

In the end, $E_t(v_p)$ is the worker's dialogue state which contains information of a part of the graph and $E_t(v_u)$ is the manager's dialogue state which contains information of the whole graph.

### 3.2 Hierarchical Dialogue Policy

After obtaining the dialogue states and node embeddings, our system will unfold the dialogue according to a hierarchical policy.

Specifically, the manager first selects $v_p \in \mathbb{V}_p$ as a worker to verify the identity state of $v_p$ according to a high-level policy $\pi^m$. Then, the worker will choose some answer nodes from its predecessors $N(v_p)$ to generate questions about $v_p$ according to a low-level policy $\pi^w$. If the worker gives the decision $d^w \in \{\text{Fraud}, \text{Non-Fraud}\}$ about the identity state of $v_p$, $\pi^w$ will end and the manager will select a new worker again or give the final decision. If the manager gives the final decision $d^m \in \{\text{Fraud}, \text{Non-Fraud}\}$ about the applicant's identity state, $\pi^m$ will end. Formally, $\pi^m$ and $\pi^w$ are defined as follows:

$$\begin{aligned}
\pi_t^m(v_p|E_t(v_u)) &\propto \exp\left(\mathbf{W}^m\left[E_t(v_u), E_t(v_p)\right] + b^m\right) \\
\pi_t^m(d^m|E_t(v_u)) &\propto \exp\left(\mathbf{W}^m\left[E_t(v_u), E(d^m)\right] + b^m\right) \\
\pi_t^w(v_a|E_t(v_p)) &\propto \exp\left(\mathbf{W}^w\left[E_t(v_p), E_t(v_a)\right] + b^w\right) \\
\pi_t^w(d^w|E_t(v_p)) &\propto \exp\left(\mathbf{W}^w\left[E_t(v_p), E(d^w)\right] + b^w\right)
\end{aligned} \quad (4)$$

where $\{\mathbf{W}^m, \mathbf{W}^w, b^m, b^w, E(d^m), E(d^w)\}$ are parameters, $E_t(v_u)$ and $E_t(v_p)$ are dialogue states of the manager and worker in the $t$-th turn, $E(d^m)$ is the encoding of the manager's terminal action which has the same dimension as $E_t(v_p)$, and $E(d^w)$ is the encoding of the worker's terminal action which has the same dimension as $E_t(v_a)$.

Besides, to prevent the two agents from making decisions in haste, domain rules are applied to their dialogue policies by "Action Mask" (Williams et al., 2017). Specifically, domain rules are defined as follows. First, only after all or at least three answer nodes related to a worker have been explored can the worker make the decision. Second, only after all workers have made decisions or at least one worker's decision is "Fraud" can the manager make the final decision.

## 4 Training

### 4.1 Reward Function

We expect the system can give correct decisions about applicants within minimum turns. Thus, at the end of each dialogue, the manager receives a positive reward $r_{crt}^m$ for correct decision, or a negative reward $-r_{wrg}^m$ for wrong decision. If the manager selects a worker to unfold the dialogue in the $t$-th turn and the worker gives $n_t^w$ questions to the applicant, the manager will receive a negative reward $-n_t^w * r_{turn}$. Besides, we provide an internal reward to optimize the low-level policy. Specifically, if the worker gives a correct decision about the corresponding personal information, it will receive a positive reward $r_{crt}^w$. Otherwise, it will receive a negative reward $-r_{wrg}^w$. And in each turn, the worker receives a negative reward $-r_{turn}$ to encourage shorter interactions.

### 4.2 Reinforcement Learning

The two agents can be trained with policy gradient (Williams, 1992) approach as follows:

$$\nabla \pi_t^m = \Big(R_t^m - V_t^m\big(E_t(v_u)\big)\Big) \nabla \log \pi_t^m\big(a_t^m | E_t(v_u)\big)$$
$$\nabla \pi_t^w = \Big(R_t^w - V_t^w\big(E_t(v_p)\big)\Big) \nabla \log \pi_t^w\big(a_t^w | E_t(v_p)\big) \tag{5}$$

where $R_t^m$ and $R_t^w$ are the discounted returns of two agents, $a_t^m$ and $a_t^w$ are their sampled actions, $V_t^m\big(E_t(v_u)\big)$ and $V_t^w\big(E_t(v_p)\big)$ are value networks which are optimized by minimizing mean-square errors to $R_t^m$ and $R_t^w$ respectively.

### 4.3 Pre-Training

Before reinforcement learning (RL), supervised learning (SL) is applied to mimic dialogues provided by a rule-based system. Rules are defined as follows. First, the manager selects a worker randomly. Then, the worker will select answer nodes randomly to generate questions. Let $n_{crt}/n_{wrg}$ denotes the number of correctly/incorrectly answered questions in this worker's decision process. If $|n_{crt} - n_{wrg}| \geq 3$ or all answer nodes related to this worker have been explored, the worker will give its decision. If $n_{crt} < n_{wrg}$, the worker's decision will be "Fraud" and the manager's decision will be "Fraud" too. Otherwise, the worker's decision will be "Non-Fraud" and the manager will choose a new worker to continue the dialogue. In the end, if all workers' decisions are both "Non-Fraud", the manager's decision will be "Non-Fraud".

## 5 Experiments and Results

### 5.1 User Simulator and Human Experiments

Simulating users' behavior is an efficient way to evaluate dialogue systems. In our task, the applicants' behavior is answering derived questions. Thus, the key of user simulator is to estimate the probability $p(\boldsymbol{k}_i)$, where $\boldsymbol{k}_i$ is a binary random variable which denotes whether or not the applicant knows the triplet fact $\boldsymbol{t}_i$ behind a question $\boldsymbol{q}_i$.

Intuitively, $p(\boldsymbol{k}_i)$ depends on three factors. First, if the applicant's identity state is "Non-Fraud", $p(\boldsymbol{k}_i = 1)$ will be greater than $p(\boldsymbol{k}_i = 0)$. Second, the wider a triplet fact $\boldsymbol{t}_i$ spreads on the internet, the more likely applicants know it. For example, almost all of applicants know (Baidu, Founder, Robin Li) because there are a lot of web pages containing this fact on the internet. Third, if applicants know other triplets that are related to $\boldsymbol{t}_i$, they may well know $\boldsymbol{t}_i$ because it is easy to deduce $\boldsymbol{t}_i$ based on what they know. For example, if applicants know (Nanjing University, Park, Gulou Park) and (Gulou Park, SubwayStation, Gulou Subway Station), they may well know (Nanjing University, SubwayStation, Gulou Subway Station).

To formalize the effect of the three factors on applicants' behavior, we introduce three hypotheses: (1) For both fraudsters and normal applicants, $p(\boldsymbol{k}_i = 1)$ is proportional to the "spread degree" of $\boldsymbol{t}_i$. (2) The "spread degree" of $\boldsymbol{t}_i$ can be approximated by the number of retrieved results (denoted as $\text{Freq}(e_i^h, e_i^t)$) in search engine where the keyword is the head entity $e_i^h$ and the tail entity $e_i^t$ of $\boldsymbol{t}_i$. (3) For any three triplets, if they form a closed loop (regardless of directions) and applicants know two of them, the applicants must know all of them.

To generate simulated loan applicants, we first estimate the function relations between $p(\boldsymbol{k}_i = 1)$ and $\text{Freq}(e_i^h, e_i^t)$ via human experiments. Specifically, we ask 31 volunteers to answer derived questions[10] about their own and others' personal information. And then, for the question $\boldsymbol{q}_i$, we place it into a discrete bin according to the logarithm of $\text{Freq}(e_i^h, e_i^t)$. In each bin, we use the ratio of correctly answered questions to approximate $p(\boldsymbol{k}_i = 1)$. In the end, the relations are shown in Fig. 4. We can find that the statistical distributions of real behavior patterns of normal applicants and fraudsters are distinguishable and the results agree with our first two intuitions.

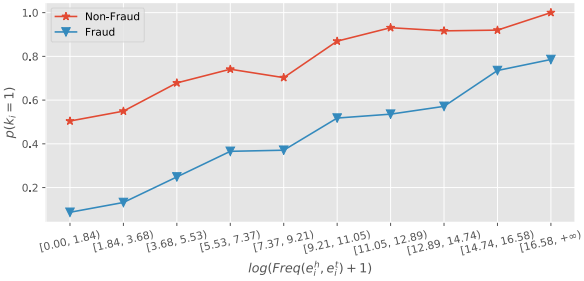


Figure 4: The relations between $\text{Freq}(e_i^h, e_i^t)$ and $p(\boldsymbol{k}_i)$ for two kinds of applicants. $\text{Freq}(e_i^h, e_i^t)$ is used to approximate the "spread degree" of $\boldsymbol{t}_i$. $p(\boldsymbol{k}_i = 1)$ indicates the probability that applicants know $\boldsymbol{t}_i$.

Then, we get simulated loan applicants[11] following a "*sampling and calibration*" manner. Specifically, given an applicant's personal information, we first sample the identity state randomly. If the

[10] There are 1516 derived questions in all.

[11] Note that we can generate any number of simulated applicants based on one applicant's personal information.

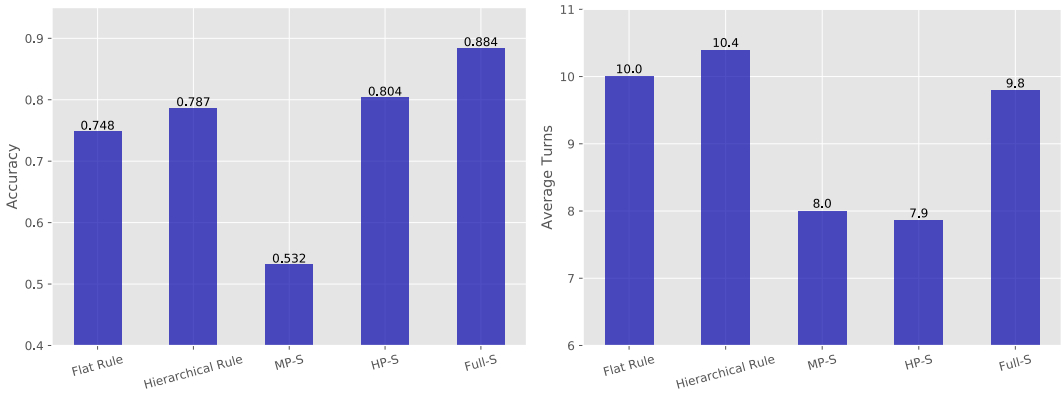


Figure 5: Performance of different systems. Tested on 10 epochs using the best model during training.

sampling result is "Fraud", we will sample $1 \sim 4$ information item(s) randomly to be the fake information. Generally, forging information about "School" and "Company" may result in a larger loan. Thus, when sampling the fake information, the sampling probability of "School" and "Company" is twice the sampling probability of "Residence" and "BirthPlace". Then, for each personal information and its related triplet $\boldsymbol{t_i}$, we sample $\boldsymbol{k_i}$ based on (1) whether the personal information is fake (2) Freq$(e_i^h, e_i^t)$ and (3) the corresponding function relation in Fig. 4. Because the sampling results $\{\boldsymbol{k_1}, ..., \boldsymbol{k_n}\}$ are independent from each other, there may be the situations where the sampling results do not satisfy the rule defined in our third hypothesis. If that happens, we calibrate it until all sampling results agree with the hypothesis. Finally, if $\boldsymbol{k}_i = 1$, the applicant will give the correct answer to the question $\boldsymbol{q}_i$. Otherwise, the applicant's response is "D. I am not quite clear.".

### 5.2 Baselines

We compare our model (denoted as Full-S) with two rule-based baselines. In addition, to study the effect of message passing and hierarchical policy on the model training, we compare Full-S with two neural baselines for the ablation study.

- Flat Rule: The system selects 10 questions randomly to ask applicants. If the number of correctly answered questions is fewer than the number of incorrectly answered questions, the system's decision will be "Fraud". Otherwise, the system's decision will be "Non-Fraud".
- Hierarchical Rule: A rule-based system which uses a hand-crafted hierarchical policy to unfold dialogues. As shown in Section 4.3, we use this system to pre-train Full-S.
- MP-S: A neural dialogue system which uses message passing to infer dialogue states but uses a flat policy to unfold dialogues. That is, the manager selects answer nodes directly to generate derived questions.
- HP-S: A neural dialogue system which uses the hierarchical policy to unfold dialogues but does not use message passing to infer dialogue states. That is, $K$ is 0 in Eq. 2.

### 5.3 Implementation Details

We collect 906 applicants' personal information, and randomly select 706 for training, 100 for dev, and 100 for test. In each batch, we sample 32 applicants' information for simulation. The maximum interaction turns of the system and the worker are 40 and 10 respectively. The iteration depth $K$ is 2 in message passing. In the reward function, $r_{crt}^m = 3$, $r_{wrg}^m = 3$, $r_{crt}^w = 1$, $r_{wrg}^w = 1$, $r_{turn} = 0.1$. The discount factors are 0.999 and 0.99 for the manager and worker respectively. All neural dialogue systems are both pre-trained with rule-based systems for 20 epochs. We pre-train MP-S with Flat Rule because they both use the flat policy. Besides, we pre-train HP-S and Full-S with Hierarchical Rule because they both use the hierarchical policy. In the RL stage, all neural dialogue systems are trained for 300 epochs. When testing, we repeat 10 epochs and take the average.

### 5.4 Test Performance

We compare Full-S with baselines in terms of two metrics: recognition accuracy and average turns.

Fig. 5 shows the test performance. We can see that the accuracy of Flat Rule is lower than Hierarchical Rule, and the accuracy of the data-driven counterpart of Flat Rule (MP-S) is just slightly

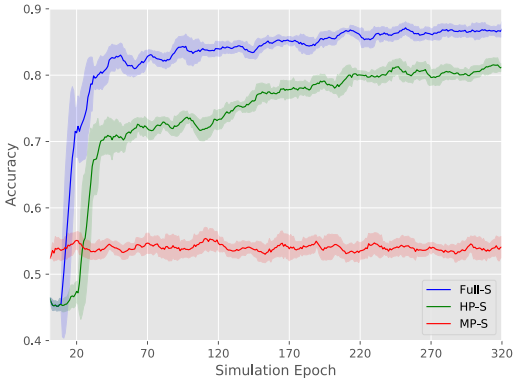


Figure 6: Accuracy curves of different neural models in dev set. The first "20 epochs" indicates the pre-training stage. The last "300 epochs" indicates the RL stage.

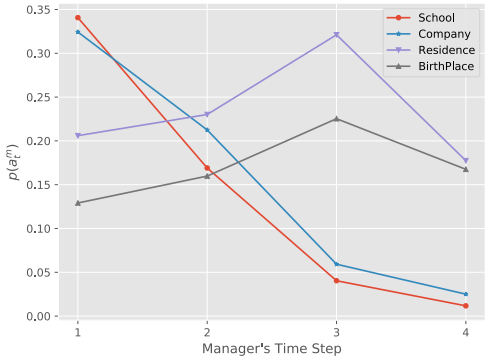


Figure 7: Manager's action probability curves. Each curve indicates the probability of selecting a piece of personal information to verify. For each curve, we take the average of all dialogues during testing.

higher than randomly guessing. It means that using the hierarchical policy to unfold dialogues is necessary for our task. Besides, HP-S achieves a higher accuracy than its rule-based counterpart (Hierarchical Rule) within much fewer turns. It proves that the data-driven system is more efficient than the rule-based system. Finally, equipped with message passing and hierarchical policy, Full-S achieves the best accuracy. And it is interesting to note that Full-S requires more turns but achieves much higher accuracy than HP-S. One possbile reason is that HP-S may easily trap in local optimum without message passing to infer dialogue states.

### 5.5 Ablation Study

To study the effect of message passing and hierarchical policy, we show the learning curves of three neural dialogue systems in Fig. 6. Each learning curve is averaged on 10 epochs.

We find that, compared with Full-S and HP-S, MP-S is unable to learn any useful dialogue policy during training. There are two reasons for this. First, the action space of flat policy is too large, which results that MP-S suffers from the sparse reward and long horizon issues. Second, without explicitly modeling the logic relation between the manager and workers, MP-S is prone to errors. Besides, we can see that the convergence speed of Full-S is faster than HP-S in both the pre-training and the RL stages. This is because message passing can model structured information of the KG, and hence Full-S is more efficient in policy learning.

### 5.6 Manager's Policy Analysis

To better understand the high-level dialogue policy, we analyze the manager's behavior in Full-S.

First, we show the manager's action probability curves in Fig. 7. We can see that selecting "School" and "Company" to verify personal information has a priority over "Residence" and "BirthPlace" in the first decision step. And in the following two decision steps, the probabilities of selecting "Residence" and "BirthPlace" will increase. This is because simulated applicants tend to forge personal information about "School" and "Company" for a larger loan. Consequently, to discover fake information faster, the manager learns to prioritize different information items.

Second, intuitively the manager's policy should follow two logic rules in our task:

**Rule1:** If a worker's decision is "Fraud" ($Cond1$), the dialogue should end immediately and the manager's decision will be "Fraud" ($RS1$).

**Rule2:** If all workers' decisions are both "Non-Fraud" ($Cond2$), the manager's decision will be "Non-Fraud" ($RS2$).

To test whether the manager follows the two rules, we calculate the probabilities of $RS1$ and $RS2$ under $Cond1$ and $Cond2$ respectively. Specifically, in the test data, $p(RS1|Cond1) = 0.95$ and $p(RS2|Cond2) = 0.96$. It proves that the manager will adopt workers' suggestions in most situations.

Meanwhile, we study cases where the manager does not follow the two rules and find some interesting phenomena. Specifically, if only one worker's decision is "Fraud" and the applicant can answer a few questions given by this worker, the manager's decision may be "Non-Fraud". Besides, if all workers' decisions are both "Non-Fraud" but the applicant can not answer most of the questions given

| All triplets that are related to "Shanghai Sports University" (replaced with $School$ for short): | |
|---|---|
| ($School$, SuperMarket, Educational Supermarket)<br>($School$, LocatedIn, Shanghai)<br>($School$, DigitalMall, JinLu Security)<br>($School$, ConvenienceStore, HaoDe)<br>(HaoDe, FruitShop, Xiao Liu Fruit) | ($School$, PetMarket, Seasons Garden)<br>($School$, FoundedDate, 2002)<br>($School$, FruitShop, Xiao Liu Fruit)<br>(Xiao Liu Fruit, ConvenienceStore, HaoDe) |
| HP-S | Full-S |
| *System: Which is the nearest pet market to $School$?*<br>Applicant: I am not quite clear.<br>*System: Which is the nearest digital mall to $School$?*<br>Applicant: I am not quite clear.<br>*System: Where is $School$ located?*<br>Applicant: Shanghai. | *System: Which is the nearest pet market to $School$?*<br>Applicant: I am not quite clear.<br>*System: Which is the nearest digital mall to $School$?*<br>Applicant: I am not quite clear.<br>*System: Which is the nearest fruit shop to $School$?*<br>Applicant: Xiao Liu Fruit<br>*System: Which is the nearest supermarket to $School$?*<br>Applicant: Educational Supermarket<br>*System: When was $School$ founded?*<br>Applicant: 2002 |
| Decison: Fraud (Wrong) | Decison: Non-Fraud (Correct) |

Table 1: Examples of the low-level policies in two systems. Note that the information about "School" is not fake.

by one worker, the manager's decision may still be "Fraud". In fact, when the two cases happen, the worker may make the wrong decision. However, the manager can still give the correct decision. It means the manager is robust to workers' mistakes.

### 5.7 Worker's Policy Analysis

To better understand the low-level dialogue policy and the effect of message passing on it, we compare workers' behaviors in HP-S and Full-S.

Table 1 shows an example of verifying personal information about "School" in HP-S and Full-S. We can see that the two systems give the same two questions in the first two turns. This is because the triplets behind the two questions are rarely known to fraudsters. It means that the low-level policies learn to give priority to such triplets for better distinguishing fraudsters from normal applicants. In the third turn, HP-S gives a question that is easy to answer for fraudsters and makes the wrong decision. However, Full-S notices the applicant gives the correct answer to a question that is hard to answer for fraudsters. Thus, Full-S does not make the decision in haste but continue the dialogue. Besides, it is worth noting that Full-S has not chosen ($School$, ConvenienceStore, HaoDe) to generate the derived question. This is because the message passing mechanism models the relation between "HaoDe" and "Xiao Liu Fruit". Specifically, because the two entities are closely related to each other, if applicants know "Xiao Liu Fruit", they may well know "HaoDe". Thus, there is no need to select this triplet anymore.

## 6 Related work

As far as we know, there is no published work about detecting identity fraud via interactions. We describe the two most related directions as follows:

**Deception Detection.** Detecting deception is a longstanding research goal in many artificial intelligence topics. Existing work has mainly focused on extracting useful features from non-verbal behaviors (Meservy et al., 2005; Lu et al., 2005; Bhaskaran et al., 2011), speech cues (Levitan et al., 2018; Graciarena et al., 2006) or both (Krishnamurthy et al., 2018; Pérez-Rosas et al., 2015) to train a classification model. In their work, the definition of deception is telling a lie. Besides, existing work requires labeled data, which is often hard to get. In contrast, we focus on detecting identity fraud through multi-turn interactions and use reinforcement learning to explore the anti-fraud policy without any labeled data.

**Dialogue System.** Our work is also related to task-oriented dialogue systems (Young et al., 2013; Wen et al., 2017; Li et al., 2017; Gašić et al., 2011; Wang et al., 2018, 2019). Existing systems have mainly focused on slot-filling tasks (e.g., booking a hotel). In such tasks, a set of system actions can be pre-defined based on the business logic and slots. In contrast, the system actions in our task are selecting nodes in the KG to generate questions. Thus, the structured information is important in our task. Besides, some works also try to model structured information in dialogue systems. For example, Peng et al. (2017) used hierarchical reinforcement learn-

ing (Vezhnevets et al., 2017; Kulkarni et al., 2016; Florensa et al., 2017) to design multi-domain dialogue management. Chen et al. (2018) used graph neural networks (Battaglia et al., 2018; Li et al., 2015; Scarselli et al., 2009; Niepert et al., 2016) to improve the sample-efficiency of reinforcement learning. He et al. (2017) used DynoNet to incorporate structured information in the collaborative dialogue setting. Compared with them, our method is a combination of the graph neural networks and hierarchical reinforcement learning, and experiments prove that they both work in the novel dialogue task.

## 7 Conclusion

This paper proposes to detect identity fraud automatically via dialogue interactions. To achieve this goal, we present structured dialogue management to explore anti-fraud dialogue strategies based on a KG with reinforcement learning and a heuristic user simulator to evaluate our systems. Experiments have shown that end-to-end systems outperform rule-based systems and the proposed dialogue management can learn interpretable and flexible dialogue strategies to detect identity fraud more efficiently. We believe that this work is a basic first step in this promising research direction and will help promote many real-world applications.

## 8 Acknowledgments

The research work described in this paper has been supported by the National Key Research and Development Program of China under Grant No. 2017YFB1002103. We would like to thank Shaonan Wang, Yang Zhao, Haitao Lin, Cong Ma, Lu Xiang and Junnan Zhu for their suggestions on this paper and Fenglv Lin for his help in the POI dataset construction.

## References

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

Nisha Bhaskaran, Ifeoma Nwogu, Mark G Frank, and Venu Govindaraju. 2011. Lie to me: Deceit detection via online behavioral learning. In *Face and Gesture 2011*, pages 24–29. IEEE.

Lu Chen, Bowen Tan, Sishan Long, and Kai Yu. 2018. Structured dialogue policy with graph neural networks. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1257–1268.

Carlos Florensa, Yan Duan, and Pieter Abbeel. 2017. Stochastic neural networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1704.03012*.

Milica Gašić, Filip Jurčíček, Blaise Thomson, Kai Yu, and Steve Young. 2011. On-line policy optimisation of spoken dialogue systems via live interaction with human subjects. In *2011 IEEE Workshop on Automatic Speech Recognition & Understanding*, pages 312–317. IEEE.

Kallirroi Georgila, James Henderson, and Oliver Lemon. 2006. User simulation for spoken dialogue systems: Learning and evaluation. In *Ninth International Conference on Spoken Language Processing*.

Martin Graciarena, Elizabeth Shriberg, Andreas Stolcke, Frank Enos, Julia Hirschberg, and Sachin Kajarekar. 2006. Combining prosodic lexical and cepstral systems for deceptive speech detection. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, volume 1, pages I–I. IEEE.

He He, Anusha Balakrishnan, Mihail Eric, and Percy Liang. 2017. Learning symmetric collaborative dialogue agents with dynamic knowledge graph embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1766–1776, Vancouver, Canada. Association for Computational Linguistics.

Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. 2016. Knowledge graph completion with adaptive sparse transfer matrix. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Gangeshwar Krishnamurthy, Navonil Majumder, Soujanya Poria, and Erik Cambria. 2018. A deep learning approach for multimodal deception detection. *arXiv preprint arXiv:1803.00344*.

Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. 2016. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in neural information processing systems*, pages 3675–3683.

Sarah Ita Levitan, Angel Maredia, and Julia Hirschberg. 2018. Acoustic-prosodic indicators of deception and trust in interview dialogues. *Proc. Interspeech 2018*, pages 416–420.

Xiujun Li, Yun-Nung Chen, Lihong Li, Jianfeng Gao, and Asli Celikyilmaz. 2017. End-to-end task-completion neural dialogue systems. *arXiv preprint arXiv:1703.01008*.

Xiujun Li, Zachary C Lipton, Bhuwan Dhingra, Lihong Li, Jianfeng Gao, and Yun-Nung Chen. 2016. A user simulator for task-completion dialogues. *arXiv preprint arXiv:1612.05688*.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. 2015. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*.

Shan Lu, Gabriel Tsechpenakis, Dimitris N Metaxas, Matthew L Jensen, and John Kruse. 2005. Blob analysis of the head and hands: A method for deception detection. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, pages 20c–20c. IEEE.

Thomas O Meservy, Matthew L Jensen, John Kruse, Judee K Burgoon, Jay F Nunamaker, Douglas P Twitchell, Gabriel Tsechpenakis, and Dimitris N Metaxas. 2005. Deception detection through automatic, unobtrusive analysis of nonverbal behavior. *IEEE Intelligent Systems*, 20(5):36–43.

Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. 2016. Learning convolutional neural networks for graphs. In *International conference on machine learning*, pages 2014–2023.

Baolin Peng, Xiujun Li, Lihong Li, Jianfeng Gao, Asli Celikyilmaz, Sungjin Lee, and Kam-Fai Wong. 2017. Composite task-completion dialogue policy learning via hierarchical deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2231–2240.

Verónica Pérez-Rosas, Mohamed Abouelenien, Rada Mihalcea, Yao Xiao, CJ Linton, and Mihai Burzo. 2015. Verbal and nonverbal clues for real-life deception detection. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2336–2346.

Olivier Pietquin and Thierry Dutoit. 2006. A probabilistic framework for dialog simulation and optimal strategy learning. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(2):589–599.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

Théo Trouillon, Christopher R Dance, Éric Gaussier, Johannes Welbl, Sebastian Riedel, and Guillaume Bouchard. 2017. Knowledge graph completion via complex tensor factorization. *The Journal of Machine Learning Research*, 18(1):4735–4772.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. 2017. Feudal networks for hierarchical reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3540–3549. JMLR. org.

Weikang Wang, Jiajun Zhang, Qian Li, Mei-Yuh Hwang, Chengqing Zong, and Zhifei Li. 2019. Incremental learning from scratch for task-oriented dialogue systems. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3710–3720, Florence, Italy. Association for Computational Linguistics.

Weikang Wang, Jiajun Zhang, Han Zhang, Mei-Yuh Hwang, Chengqing Zong, and Zhifei Li. 2018. A teacher-student framework for maintainable dialog manager. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3803–3812, Brussels, Belgium. Association for Computational Linguistics.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gasic, Lina M Rojas Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 438–449.

Jason D Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 665–677.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

# Hierarchy Response Learning for Neural Conversation Generation

**Bo Zhang**[1] and **Xiaoming Zhang**[1,2]
[1] School of Cyber Science and Technology
[2] Hefei Innovation Research Institute
Beihang University, China
{zhangnet,yolixs}@buaa.edu.cn

## Abstract

The neural encoder-decoder models have shown great promise in neural conversation generation. However, they cannot perceive and express the intention effectively, and hence often generate dull and generic responses. Unlike past work that has focused on diversifying the output at word-level or discourse-level with a flat model to alleviate this problem, we propose a hierarchical generation model to capture the different levels of diversity using the conditional variational autoencoders. Specifically, a hierarchical response generation (HRG) framework is proposed to capture the conversation intention in a natural and coherent way. It has two modules, namely, an expression reconstruction model to capture the hierarchical correlation between expression and intention, and an expression attention model to effectively combine the expressions with contents. Finally, the training procedure of HRG is improved by introducing reconstruction loss. Experiment results show that our model can generate the responses with more appropriate content and expression.

## 1 Introduction

Neural conversation generation (Xu et al., 2017; Zhang et al., 2018), focusing on responding to humans intelligently on a variety of topics, has drawn great attention from both academic and industry. The sequence-to-sequence model (Seq2Seq) (Sutskever et al., 2014) is one type of neural generation model that maximizes the probability of generating a response given the dialogue context. It enables the incorporation of rich context to generate coherent responses in an unsupervised manner. However, it was found that Seq2Seq models suffer from so-called safe response problem (Xu et al., 2017) , i.e., they tend to generate some dull and generic repetitive responses (e.g., *"I think so"*, *"I don't know'*, etc.), rather than meaningful and conscious expression. Xu et al. (2017) ascribed this to the fundamental nature of statistical models since the distribution of most pieces of information are relatively sparser when compared to the safe response patterns in the open domain conversations. Some works attempted to improve the architecture of Seq2Seq models, including introducing reinforcement learning (Zhang et al., 2018), encouraging responses that have long-term payoff, etc. The other important reason is that the response generation model cannot express the intention and emotion internally. Thus, one line of research has focused on forcing the model to simulate some human's skills by augmenting the input with rich meta information. For example, some recent works biased the responses to some specific personas (Li et al., 2016b) or emotions (Huber et al., 2018).

Usually, in the process of human conversation, a speaker participates in the dialogue including the following steps. The speaker is firstly required to decide what the intention is to reflect the inner feelings or opinions. In the speaker's knowledge base, there may be varieties of appropriate expressions that can be found to represent his current intention. Therefore, a meaningful response can be produced by choosing one of the expressions and filling it with relevant content. For example, if a man wants to ask the way to the Park, he first needs to select an appropriate expression from a cluster of the expressions, e.g., *"Where is the . . . ?"*, *"How do I get to the . . . ?"* and *"Is the . . . far from here?"*, and then replaces ". . . " as the destination, i.e., the Park. As a crucial feature in natural conversation, **dialog acts** (Poesio and Traum, 1998) have been widely used in the dialogue managers to represent the intentions. Existing works introduce dialog acts to label a cluster of responses and a latent variable is learned to select a dialog act for response generation (Zhao et al., 2017; Serban



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1772–1781, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

et al., 2017a). However, it is not effective to capture the output diversity since the natural correlation between the expression patterns and dialog acts is not learnt. Intuitively, another latent level can be introduced to generate different expressions from the same dialog act, and a hierarchical structure can be used to model the response generation process. That is, the knowledge base is first constructed over the pairs of expressions and dialog acts to capture the latent correlation between them. Then, varieties of expressions can be selected from the knowledge base to be filled with the response content based on the latent correlation.

To learn the hierarchical model, it is quite challenging in large-scale conversation generation due to the following reasons. **First**, the semantic world is populated with a vast number of expressions, each of which corresponds to a specified label that reflects a kind of dialog act. Obtaining high-quality expression-act data is impractical particularly in open domain conversations. **Second**, it is difficult to incorporate expression and content into the generation model in a nature and coherent way because they have different semantic representation patterns. **Last**, this process cannot be efficiently optimized using stochastic gradient descent (SGD) akin to backpropagation on feedforward neural networks.

To tackle the challenges, we propose to take advantage of the hierarchical nature of response generation. In particular, we investigate: (1) how to automatically learn a hierarchical model to naturally capture the response generation process; (2) how to adaptively learn and adjust the influence ratio between expression and content. Our solutions to these questions result in a new architecture for neural response generation. In particular, a novel hierarchical response generation (HRG) framework is proposed to effectively capture the process of response generation. An expression reconstruction model with a two-level probability structure is introduced to randomly generate the expressions, and an expression attention model is proposed to effectively fill the expressions with content. Finally, an efficient training method is proposed to learn the model within the framework of conditional variational autoencoders (CVAE) (Doersch, 2016). The main contributions are outlined as:

- We propose to investigate the problem of generating variety and meaningful responses by imitating the human response process with a hierarchical response generation model.
- We propose an end-to-end framework to incorporate the expression and response content into the dialog generation. Our model is interpretable and even controllable compared to traditional generation model.
- We empirically demonstrate that our approach can generate responses with better expressions and content than traditional generation model.

## 2 Problem Statement

Our problem is formulated as follows: Given a dialog context $C$ and a dialog act $a$ of the response to be generated, the goal is to generate a response $y = (y_1, y_2, ..., y_n)$ that is coherent with the dialog act $a$. Essentially, the model estimates the probability: $P(y|C, a) = \prod_t P(y_t|y_{<t}, C, a)$. A simple implementation is to directly embed the act information into the Seq2Seq model. However, as shown in our experiments, it still suffers from safe response problem.

In this paper, we propose a novel hierarchical model to imitate the human thinking process in the conversation generation. The hierarchical generation process is: (1) for each dialog act $a$, a set $\Omega(a)$ is constructed to contain all the corresponding expressions of $a$; (2) to generate an expression, a dialog act $a$ is first selected, and then an appropriate expression $e$ is also selected from $\Omega(a)$; (3) a response is obtained by filling the expression $e$ with relevant content according to the dialog context $C$. This hierarchical model allows us to express the responses with diverse expression templates of the same dialog act by drawing different samples from $\Omega(a)$. However, in addition to the difficulty of constructing high-quality set $\Omega(a)$, it is also needed to maximize the probability of each $y$ in the training set with the objective: $P(y|C, a) = \int_{e \in \Omega(a)} P(y|C, e)\, de$, which is also difficult to compute by the numeric methods.

In our approach, the expression $e$ is modeled as a conditional distribution over the dialog act $a$, i.e., $p_\theta(e|a)$. The response is then generated by feeding the expression $e$ obtained based on $p_\theta(e|a)$ into the model, i.e., $P(y|C, e), e \sim p_\theta(e|a)$. Now, the training objective is simplified as follows:

$$P(y|C, a) = \int P(y|C, e)\, p_\theta(e|a)\, de. \quad (1)$$

This objective can be transformed as the variational lower bound of CVAE (Doersch, 2016), and thus can be optimized efficiently. Specifically, the variational approximation $q_\phi(e|y)$ is constructed to approximate the intractable posterior $p_\theta(e|a)$. Assuming that the meaning of expression $e$ is independent of $C$, we train the model by maximizing the variational lower bound,

$$\begin{aligned} L(\theta, \phi; y, C, a) &= -KL(q_\phi(e|y)||p_\theta(e|a)) \\ &+ \mathbb{E}_{e\sim q_\phi(e|y)}[\log p_\theta(y|C, e, a)] \\ &\leq \log p(y|C, a) \end{aligned} \quad (2)$$

where $KL(\cdot||\cdot)$ is the Kullback-Leibler divergence to measure the distance between two distributions.

Note that in our problem statement, we assume that the dialog act of the to-be-generated response is given in advance, rather than predicted depending on the context. Many existing researches (Sacks et al., 1978; Young et al., 2013; Daniel Jurafsky, 2017) have explored dialog act interactions with dialog system and proposed some methods to decide the most appropriate dialog act for the response. In this paper, we only focus on response generation. During the testing process, we simply specify a dialog act to the model. We leave this study to our future work.

## 3 Hierarchical Response Generation

Building upon the encoder-decoder models (e.g., Seq2Seq, HRED(Serban et al., 2016)), a Hierarchical Response Generation (HRG) framework is proposed to effectively generate more diverse expressions for conversation generation. As shown in Figure 1, HRG contains two main modules: Expression Reconstruction model and Expression Attention model. A training method is proposed to learn the hierarchical HRG model in Section 3.3.

### 3.1 Expression Reconstruction

To maximize the objective Eq.(2), we are first required to model the networks $q_\phi(e|y)$ and $p_\theta(e|a)$. The task of network $q_\phi(e|y)$ is to capture the expression representations from the responses while $p_\theta(e|a)$ to sample an expression representation from the distributions associated with the specified dialog acts. As shown in Figure 2, in the framework of CVAE, the response is first encoded as a latent variable, and then a decoder is introduced to reconstruct the response. But this generation process is not interpretable and controllable. Thus, we

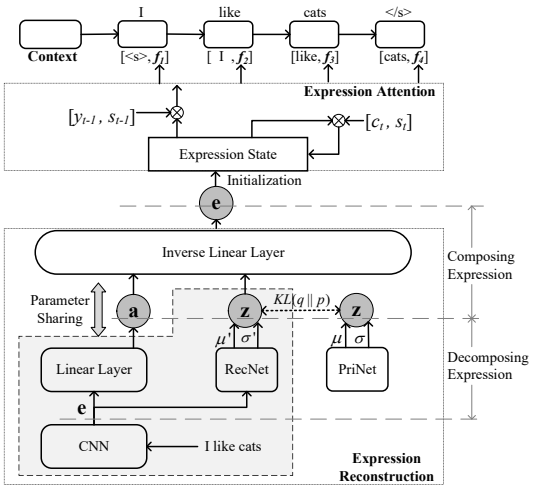


Figure 1: The overview of Hierarchical Response Generation (HRG). The components in the dashed box are removed during testing. The dialog act $a$ is specified manually and the random variable $z$ is generated by PriNet during testing.

propose a novel method to reconstruct the expression with multiple latent variables by establishing links between $q_\phi(e|y)$ and $p_\theta(e|a)$.

*Modeling* $q_\phi(e|y)$. It was found that the convolutional layer can extract the common patterns within the local regions of the input utterance (Kim, 2014). Therefore, the text convolutional network proposed in (Kim, 2014) is leveraged to mine the relationship between expression patterns and responses. Particularly, the network $q_\phi(e|y)$ consists of several convolutional filtering, local contrast normalization, and max-pooling layers, followed by several connected linear layers. Formally, given a response $y$, the expression representation can be described as

$$e = \text{CNN}(y). \quad (3)$$

In the experiments, we found that the convolutional layers can effectively extract the expression representation by discarding the content-related information.

*Modeling* $p_\theta(e|a)$. Given a dialog act $a$, the network $p_\theta(e|a)$ outputs an expression representation associated with $a$. To capture output diversity, it is also necessary to generate different expressions each time given the same dialog act. The classical models (e.g., linear layer) are not capable of representing this feature. Instead, the network $p_\theta(a|e)$ is easy to model by a linear layer, where

$$a = g(e) = \boldsymbol{W_f} \cdot e + \boldsymbol{b_f}. \quad (4)$$

Motivated by this observation, we propose an Inverse Linear Layer to model the network $p_\theta(e|a)$ where the dialog act $a$ is mapped inversely into the expression representation $e$ by solving $g^{-1}$.

Penrose Theorem (Ben-Lsrael and Greville, 1976) gives a general solution of equation $AX = B$ and proofs it. To solve $g^{-1}$, we simplify Penrose Theorem under an extra constraint and give a Corollary:

**Corollary 1** *Let $A \in \mathbb{C}^{m\times n}$, $b \in \mathbb{C}^m$. The general solution of the equation $Ax = b$ is*

$$x = A^+b + (I - A^+A)z \tag{5}$$

*for arbitrary $z \in \mathbb{C}^n$ (if $0 < rank(A) < n$), where $A^+$ is Moore–Penrose inverse of $A$ and $I$ is an identity matrix.*

The transformation of $Ax$ inevitably leads to the information loss of $x$ and thus the random vector $z$ should be supplemented to the solution.

According to Corollary 1, the expression $e$ can be represented by an inverse linear layer as

$$e = g^{-1}(a, z) = \boldsymbol{W_f}^+(a - \boldsymbol{b_f}) + (I - \boldsymbol{W_f}^+\boldsymbol{W_f})z. \tag{6}$$

The expression is uniquely determined by two independent variables, i.e., the dialog act $a$ and the latent variable $z$. Given the same or similar dialog context, there may exist many valid expressions for the responses with the same dialog act $a$, each corresponding to a certain configuration of $z$. This representation allows us to express responses with diverse expression templates of specified dialog act by drawing samples from the learned distribution of $z$. The network $p_\theta(e|a)$ can be easily computed by setting $p_\theta(a, z) = p_\theta(a)p_\theta(z)$, where $a$ and $z$ are uncorrelated.

*Reconstructing Expression.* As shown in Figure 1, during training, the expression representation $e$ is first captured from the ground-truth response by the network $q_\phi(e|y)$. The expression $e$ is decomposed into multiple independent variables, i.e., $a$ and $z$, and then these variables are composed to reconstruct the expression $e$ by the network $p_\theta(e|a)$. These variables provide different discourse-level information to force the decoder to focus on multiple global information simultaneously. As shown in Figure 2, different from CVAE, some variables (i.e., $a$) in HRG are interpretable to make the model controllable while some (i.e., $z$) are continuous to reflect the latent feature. During testing, the model is controllable by specifying an appropriate dialog act to express the intention.

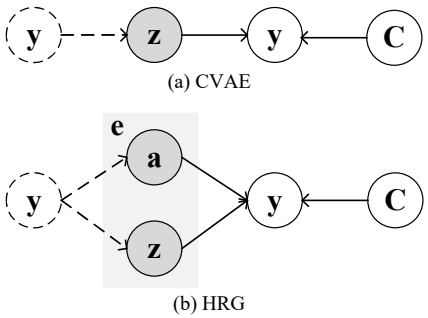


Figure 2: Graphical models for (a) CVAE and (b) HRG. Solid lines denote the generative models (a) $p(z)p(y|z, C)$ and (b) $p(e|a)p(y|e, C) = p(a)p(z)p(y|a, z, C)$; dashed lines denote the variational approximation (a) $q(z|y)$ and (b) $q(e|y) = q(a|y)q(z|y)$.

### 3.2 Expression Attention

There exist two main methods to incorporate the expression representation into the decoder. First, the concatenation of the context and expression representation $e$ is used to initialize the recurrent of the decoder RNN with a nonlinear transformation. During the decoding, the decoder RNN decodes words based on the current state and previous word embeddings $w$. The second way is that the concatenation of fixed $e$ and $w$ is fed into decoder to update its state at each step. Formally, the first way updates its state only according to $[w, 0 \cdot e]$, while the second according to $[w, 1 \cdot e]$. However, these methods may cause that the content (expression) is so powerful that the responses are without any effective expression (meaningful content). In this paper, we propose an expression attention (EA) model to attend on different parts of expression representations each step by learning a vector $\alpha = \{\alpha_i \in (0, 1)\}$ adaptively. A balance between content and expression is effectively kept by feeding $[w, \alpha \cdot e]$ into decoder.

In particular, before decoding, an expression state is initialized as $q_0 = e$ to record the current expression representation. At step $t$, a strength gate $\beta_t$ is computed based on the input of the previously decoded word $y_{t-1}$ and the previous decoder state $s_{t-1}$. The expression state is weakened by a certain amount $\beta_t$ at each step,

$$\beta_t = \text{Sigmoid}(\boldsymbol{W_s}[y_{t-1}, s_{t-1}]), \tag{7}$$

$$f_t = q_{t-1} \otimes \beta_t \tag{8}$$

where $\otimes$ is element-wise multiplication and $\text{Sigmoid}(x) = 1/(1 + exp(-x))$. The decoder updates its state conditioned on the previous token $y_{t-1}$ and the current output $f_t$ as follows:

$$s_t = \text{RNN}^d(s_{t-1}, [y_{t-1}, f_t]). \quad (9)$$

It is a dynamic process that the expression is adjusted adaptively according to the current environment and model behaviors compared to the two existing methods above. After step $t$, a self-update strategy is designed to update the expression state based on the context vector $c_t$ (computed by Attention Mechanism (Luong et al., 2015)) and the current decoder state $s_t$. This process is formulated as

$$\gamma_t = \text{Sigmoid}(\boldsymbol{W_u}[c_t, s_t]), \quad (10)$$

$$q_t = q_{t-1} \otimes \gamma_t. \quad (11)$$

The expression representation is integrated into the decoder gradually until the expression state decays to zero through multiple iterations. The expression reconstruction and attention models respectively provide discourse-level and token-level randomness respectively, which can avoid the decoder generates the next token only depending on Neural Probabilistic Language Model instead of the dialog context and the current decoding state.

### 3.3 Reconstruction Training

The learnt parameters include the embeddings of vocabulary, and those in the encoder-decoder component and HRG. According to Section 3.1, we first identify two key assumptions that are essential: Both $z$ and $a$ are the indigenous properties of the expression $e$; The meaning of $z$ is independent of the dialog act $a$. Based on them, we update the objective Eq.2 as

$$\begin{aligned} L(\theta, \phi; y, C, a) = & -KL(q_\phi(z|y)||p_\theta(z)) \\ & - KL(q_\phi(a|y)||p_\theta(a)) \\ & + \mathbb{E}_{z\sim q_\phi(z|y), a\sim q_\phi(a|y)}[\log p_\theta(y|C, a, z)]. \end{aligned} \quad (12)$$

Assuming that $z$ follows isotropic Gaussian distribution, the prior network (PriNet) $p_\theta(z) = \mathcal{N}(\mu, \sigma^2\boldsymbol{I})$ and the recognition network (RecNet) $q_\phi(z|y) = \mathcal{N}(\mu', \sigma'^2\boldsymbol{I})$, where

$$\begin{bmatrix} \mu' \\ \log(\sigma'^2) \end{bmatrix} = \boldsymbol{W_r}\text{CNN}(y) + \boldsymbol{b_r}. \quad (13)$$

Now, the term $KL(q_\phi(z|y)||p_\theta(z))$ is a KL-divergence between two multivariate Gaussian distributions which can be computed in a closed form (Doersch, 2016). Different from $z$, the dialog act $a$ follows discrete distribution. Minimizing the term $KL(q_\phi(a|y)||p_\theta(a))$ is much simpler than the continuous one, which can be evaluated by

$$\sum_{k=1}^{\mathcal{A}} q_\phi(a = k|y) \log \frac{q_\phi(a = k|y)}{p_\theta(a = k) + \epsilon} \quad (14)$$

where $\mathcal{A}$ is the number of dialog acts and $\epsilon = 10^{-6}$ is used to prevent division by zero. We denote the network $q_\phi(a|y)$ as the act classifier and its probability is evaluated by $softmax(\boldsymbol{W_f}\text{CNN}(y) + \boldsymbol{b_f})$. As shown in Figure 1, note here that the inverse linear layer shares the same parameters $\boldsymbol{W_f}$ and $\boldsymbol{b_f}$ with those in $q_\phi(a|y)$.

In the training process, by introducing the re-parameterization trick (Kingma and Welling, 2014), we obtain the variables $z$ and $a$ from RecNet $q_\phi(z|y)$ and act classifier $q_\phi(a|y)$, and then feed them into the inverse linear layer to capture expression representations. During testing, by specifying a dialog act $a$, the decoder generates the response according to the sample from the PriNet $p_\theta(z)$.

## 4 Experiment

### 4.1 Implementation Details

The dataset DailyDialog Corpus (Li et al., 2017b) is used to evaluate the proposed model. It contains 13,118 multi-turn human-human dialogs annotated with dialog acts and emotions, and covers 10 main topics about daily life. In this Corpus, the dialog act categories are {*Inform, Question, Directive, Commissive*}. In our experiments, HRG is combined into HRED model (Serban et al., 2016) as the expression-aware chatting machine (ECM). PyTorch[1] is used to implement the proposed model. All the RNN modules have 2-layer gated recurrent units (GRU) (Cho et al., 2014) structures with 500 hidden cells for each layer and are set with different parameters. Word embedding has size 300 and is initialized from Glove embedding[2]. The size of the latent variable $z$ is set to be 300. The maximum dialog turn is 5 (10 utterances).

The models are trained end-to-end using Adam optimizer (Kingma and Ba, 2015) with batch size

[1] https://pytorch.org/

[2] https://nlp.stanford.edu/projects/glove/

of 30, learning rate of 0.001 and gradient clipping at 50. To overcome the *latent variable vanishing problem* in CVAE, we use the heuristic method (Bowman et al., 2016) to encode the meaningful information in $z$. That is, we multiply the first KL term in Eq.12 by a scalar, which starts at 0 and linearly increases to 1 over the first 10,000 batches.

### 4.2 Baselines

We compare our hierarchical model with two popular baselines: (1) **HAE**: a HRED-based model which we design to embed the dialog act information in the decoder; (2) **kgCVAE**: a knowledge-guided model that introduces dialog acts to guide the learning of the CVAEs (Zhao et al., 2017). A variant of the proposed model is implemented to verify the effectiveness of expression attention (EA) model. We denote the model without EA as **w/o EA**. The hyperparameters of the baselines and variants are the same as ECM.

As for **HAE**, we initialize the embedding of dialog acts using three different methods: (1) **RD** (random): initializing the embedding randomly; (2) **LG** (logic-related): training a Skip-Gram model (Mikolov et al., 2013) to maximize the co-occurrence probability among the acts that appear within a window, $w$, in the sequence of dialog acts for each dialog (set $w$ to 1); (3) **CT** (content-related): training an act classifier $q_\phi(a|y)$ with the pairs of utterances and dialog acts in training set, and use each row in $\boldsymbol{W_f}$ as the embeddings. The size of act embeddings is set to 300, as the same with the output of EA. The concatenation of dialog act embedding and the previous word embedding is fed into the decoder of HRED to update its state at each step during decoding. HAE is trained to minimize the standard cross entropy loss of the decoder RNN model without any auxiliary loss.

### 4.3 Quantitative Analysis

Automatically evaluating the quality of the dialog model remains an open question. To evaluate how semantically relevant the response is, we report the results for three word embedding-based similarity metrics proposed by Liu et al. (2016): *Greedy Matching* (GDY), *Embedding Average* (AVG) and *Vector Extrema* (EXT). To evaluate whether the response follows the dialog act, we adopt act accuracy (ACC) as the agreement between the ground-truth dialog act and the dialog act predicted by an act classifier. We trained the act classifier and its precision and average recall in the testing set are 83.4% and 74.3% respectively.

In addition to automatic evaluation metrics, a manual evaluation metric (MUL) is also given to evaluate both the response content and expression, where three workers are employed to score a response in terms of *Content* (rating scale is 0,1,2) and *Act* (rating scale is 0,1). *Content* is evaluated based on whether the response is appropriate and natural to the dialog context, while *Act* based on whether the expression agrees with the ground-truth act. *Content* rating is a widely accepted metric proposed by Shang et al. (2015). And, the workers can easily evaluate *Act* rating based on the context since the number of acts is few in our experiment.

During testing, to efficiently measure output diversity, we generate $N$ responses from HAE models by introducing beam search. For kgCVAE and ECM, we sample $N$ times from the latent variable and only use greedy decoders. Meanwhile, for HAEs and ECM, we specify $a$ as the act of the ground-truth response.

The automatic evaluation metrics focus on comparing the generated responses $r_j$ with the ground truth $g_i$ of the conversation. We compute the scores of models based on all the $M$ test samples as follows:

$$AM = \frac{1}{M}\sum_{i=1}^{M}\max_{j\in[1,N]} d(g_i, r_j) \qquad (15)$$

where $d(\cdot)$ is one of automatic metrics described above, and $N$ is empirically set to 10. Note here that the maximum metric in Eq.15 is more appropriate to measure the output diversity than average one. This is because that taking average metrics may cause that the safety responses get higher scores than meaningful and diverse responses if most of these valid responses are not related to the ground-truth. The maximum metric can greatly reduce the error by increasing the number of samples. The evaluation of MUL metric is unrelated to the ground truth responses. To evaluate MUL metric, we randomly selected 40 dialog contexts from the test set and then generate 400 responses for each model. Each response is evaluated with a rating of *Content-Act* by workers.

The automatic evaluation results are given in Table 1. According to the word embedding-based similarity metrics, responses generated by ECM are substantially more coherent and relevant to

| Method (%) | GDY | AVG | EXT | ACC |
|---|---|---|---|---|
| HAE-RD | 11.6 | 57.7 | 33.8 | 64.2 |
| HAE-LG | 14.7 | 66.0 | 30.0 | 74.2 |
| HAE-CT | 18.2 | 72.0 | 35.5 | 77.0 |
| kgCVAE | 23.6 | 75.3 | **39.8** | — |
| ECM | **28.1** | **84.6** | 37.9 | 80.4 |
| w/o EA | 20.6 | 78.0 | 34.8 | **82.6** |

Table 1: Automatic Evaluation Result.

the topic compared to HAE models and kgCVAE. ECM obtains higher act accuracy score than HAE as well since the second KL term of Eq.(12) forces the predicted act distribution to approximate the ground-truth. ECM without EA (w/o EA) achieves the best performance in act accuracy but poor performance in embedding-based similarity metrics. It indicates that EA is an efficient model to balance the expression and the content dynamically. On the other hand, HAE-CT gets higher scores both in embedding-based metrics and in accuracy than other HAE models, which suggests that the act classifier can preserve act-related information effectively. Note here that the act accuracy of kgCVAE is not given because the response act is an internal parameter predicted by the dialog context rather than an input during testing. Compared to ECM, kgCVAE may give the decoder a wrong direction to approximate the ground-truth responses with different dialog acts.

| Method (%) | 2-1 | 2-0 | 1-1 | 1-0 | 0-1 | 0-0 |
|---|---|---|---|---|---|---|
| HAE-CT | 13.5 | 8.0 | 23.5 | 9.0 | 31.5 | 14.5 |
| ECM | 26.5 | 15.5 | 32.0 | 7.0 | 16.0 | 3.0 |
| w/o EA | 21.0 | 6.0 | 29.0 | 10.0 | 28.5 | 5.5 |

Table 2: Manual evaluation result. The percentage of responses with the ratings of *Content-Act*. For instance, 2-1 means *Content* rating is 2 and *Act* rating is 1.

Table 2 shows the manual evaluation result where the content and expression are considered simultaneously. As we can see, responses generated by w/o EA tend to contain obvious act information but a little of content, while HAE generates the responses with lower scores of *Content-Act*. Compared to other methods, responses generated by ECM keep a good balance between *Content* and *Act*. In our experiment, we also find that HAE-CT still faces serious safe response problems. However, EA provides token-level randomness to avoid that the decoder generates the next token only depending on Neural Probabilistic Language Model.

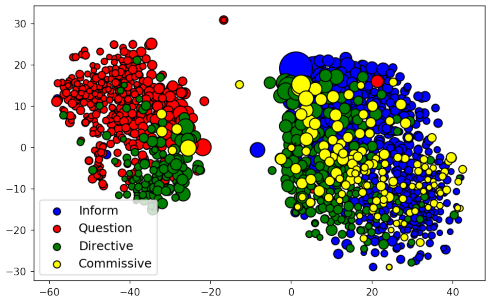


Figure 3: Visualization of expression representations for utterances with dialog acts. The size of circle represents the response length.

*Discussion.* ECM performs well than w/o EA not in automatic evaluation but also in manual evaluation obviously. Although ECM includes more trainable parameters in EA than w/o EA, the improvement of performance is mainly due to the effective architecture of EA. The EA model only involves two learnt matrices, i.e, $\boldsymbol{W_s}$ and $\boldsymbol{W_u}$, described in Section 3.2. Compared to the parameters in the multi-layer HRED, expression reconstruction, and the embedding of vocabulary, the number of parameters in EA can be ignored. Therefore, it can be concluded that EA plays a key role in improving output diversity with few parameters due to its efficient architecture.

### 4.4 Qualitative Analysis

Table 3 shows the responses generated by kgCVAE and ECM. In Example 1, speaker "A" begins with an open domain demand (directive). ECM generated highly diverse answers that cover multiple dialog acts which were fed into the model in advance during decoding. Further, we notice that the generated response with *inform* act (i.e., sample 1) has similar expression with the ground-truth one, implying that the latent $z$ is able to capture the expression-sensitive variations. It verifies the effectiveness of the hierarchical generation process. ECM can obtain effective expression representation, and fill it with appropriate content obtained from the dialog context. Example 2 is a situation where the waiter "A" tells the customer "B" that the order has done. ECM takes the *directive* act as input and generate multiple responses to give "A" some suggestions (or commands). All the responses reflect the similar behaviors with different expression styles. On the contrary, kgCVAE is capable of generating some diverse responses, but cannot accurately understand the intention of

| | |
|---|---|
| **Example 1**: **History** (Directive): **A**: Tell me a little bit about yourself , please . **Target** (Inform): **B**: My name is Dunlin and I live in Beijing . I was born in 1980 . I will graduate from Peking University this July . | |
| **kgCVAE** + $z$ Samplings | **ECM** + Dialog Acts + $z$ Samplings |
| 1. i've been in china for two weeks . | 1. (Inform) i graduated from hebes university . |
| 2. how do you wish i can speak to the teacher's letter ? | 2. (Commissive) of course , i ' ve been visiting a new company . i ' m looking for a job . |
| 3. i'd like to speak a little more , but i know him . | 3. (Question) yes , what ' s the matter ? |
| 4. excuse me ? | 4. (Directive) can you give me a example ? |
| **Example 2**: **History** (Inform): **waiter A**: good morning , sir . i ve got breakfast your ordered . **Target** (Directive): **customer B**: just put it on the table please . | |
| **kgCVAE** + $z$ Samplings | **ECM** + *Directive* Act + $z$ Samplings |
| 1. oh , no . | 1. put it on . |
| 2. yes , i'd like to have a look . it s a good choice . | 2. please check out this time . |
| 3. that's ok . it's got wireless only , and we're over . | 3. please . i am sorry to have kept you waiting . |
| 4. good morning , thanks . | 4. please wait a moment . |

Table 3: Generated responses from kgCVAE and ECM in two examples.

"A" and thus the responses lack of coherency. The human-human dialogues in the dataset follow some dialog flow patterns, such as *Question-Inform*, *Directive-Commissive* (Li et al., 2017b). kgCVAE predicts the dialog act exactly in example 1 but wrongly in Example 2 since the pattern *Inform-Directive* is not common.

In our work, a CNN module is leveraged to filter the content-related information of utterances and get a discourse-level representation, i.e. expression vector, where meaningful expression information is preserved. CNN models have been shown to be efficient for NLP and have achieved excellent results in sentence modeling and classification. So we conjecture that the expression vectors are highly correlated with the dialog acts, and each one reflects a concrete expression representation of the specified dialog act. Figure 3 visualizes the expression vectors in the test dataset in 2D space using t-SNE (Der Maaten and Hinton, 2008). We find that the expression vectors are clustered into meaningful groups associated with the dialog acts, which confirms that CNN is an efficient tool to extract the expression information.

## 5 Related Works

Vanilla Seq2Seq model usually ends up with generic and dull responses. To tackle this problem, one line of research has focused on forcing the model to imitate some human's skills by augmenting the input with rich meta information. For example, some works separately gave chatbots the ability of emotions (Zhou et al., 2018), persona (Li et al., 2016b), vision (Huber et al., 2018; Wu et al., 2018) and thinking over the knowledge base (Liu et al., 2018; Zhu et al., 2017). In this work, we consider open domain dialogue generation with dialog acts. But, only a little works (Zhao et al., 2017; Serban et al., 2017a) on open domain end-to-end modeling take dialog acts into account.

On the other hand, many attempts have also been made to improve the architecture of Seq2Seq models by changing the training methods. Li et al. (2016a) attributed safe response problems to the use of MLE objective. Some works separately attempted to replace the MLE method with maximum mutual information (Li et al., 2016a), reinforcement learning (Zhang et al., 2018; Li et al., 2016c) and adversarial learning (Xu et al., 2017; Li et al., 2017a). Serban et al. (2017b) viewed the dialog context as prior knowledge and combined HRED model into the CVAE framework. Zhao et al. (2017) further introduced dialog acts to guide the learning of CVAE. In our paper, we use CVAE to learn the hierarchy generation model.

## 6 Conclusion and Future Work

In this paper, we investigate the problem of generating meaningful responses by imitating the hierarchical process of human response. Specifically, a hierarchical response generation model is proposed to hierarchically generate the expressions and fill them with appropriate content naturally and coherently. The experiment results show that HRED model equipped with HRG can generate responses appropriate not only in content but also in expression. Different from existing works, our model is interpretable and controllable.

In the future work, we will explore the act interactions with HRG. Instead of specifying a dialog act manually, the most appropriate one can be decided automatically.

## Acknowledgments

This work was supported in part by National Natural Science Foundation of China (No.U1636210, No.U1636211, No.61602025) and Beijing Natural Science Foundation of China (No.4182037).

## References

Adi Ben-Lsrael and Thomas N. E. Greville. 1976. Generalized inverse: Theory and applications. *Journal of the Royal Statistical Society*, 139(1).

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *conference on computational natural language learning*, pages 10–21.

Kyunghyun Cho, Bart Van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP*, pages 1724–1734.

James H. Martin Daniel Jurafsky. 2017. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall series in artificial intelligence. Prentice Hall, Pearson Education International.

Laurens Van Der Maaten and Geoffrey E Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605.

Carl Doersch. 2016. Tutorial on variational autoencoders. *CoRR*, abs/1606.05908.

Bernd Huber, Daniel Mcduff, Chris Brockett, Michel Galley, and Bill Dolan. 2018. Emotional dialogue generation using image-grounded language models. In *CHI*, page 277.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *ICLR*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *HLT-NAACL*, pages 110–119.

Jiwei Li, Michel Galley, Chris Brockett, Georgios P Spithourakis, Jianfeng Gao, and Bill Dolan. 2016b. A persona-based neural conversation model. In *ACL*, volume 1, pages 994–1003.

Jiwei Li, Will Monroe, Alan Ritter, Daniel Jurafsky, Michel Galley, and Jianfeng Gao. 2016c. Deep reinforcement learning for dialogue generation. In *EMNLP*, pages 1192–1202.

Jiwei Li, Will Monroe, Tianlin Shi, Sebastien Jean, Alan Ritter, and Daniel Jurafsky. 2017a. Adversarial learning for neural dialogue generation. In *EMNLP*, pages 2157–2169.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017b. Dailydialog: A manually labelled multi-turn dialogue dataset. In *IJCNLP*, pages 986–995.

Chiawei Liu, Ryan Lowe, Iulian Vlad Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *EMNLP*, pages 2122–2132.

Shuman Liu, Hongshen Chen, Zhaochun Ren, Yang Feng, Qun Liu, and Dawei Yin. 2018. Knowledge diffusion for neural dialogue generation. In *ACL*, pages 1489–1498.

Minh Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*, pages 1412–1421.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *ICLR, Workshop Track Proceedings*.

Massimo Poesio and David Traum. 1998. Towards an axiomatization of dialogue acts. In *Proceedings of the Twente Workshop on the Formal Semantics and Pragmatics of Dialogues (13th Twente Workshop on Language Technology*. Citeseer.

Harvey Sacks, Emanuel A Schegloff, and Gail Jefferson. 1978. A simplest systematics for the organization of turn taking for conversation. In *Studies in the organization of conversational interaction*, pages 7–55. Elsevier.

Iulian V. Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, pages 3776–3783.

Iulian Vlad Serban, Chinnadhurai Sankar, Mathieu Germain, Saizheng Zhang, Zhouhan Lin, Sandeep Subramanian, Taesup Kim, Michael Pieper, Sarath Chandar, Nan Rosemary Ke, et al. 2017a. A deep reinforcement learning chatbot. *CoRR*, abs/1709.02349.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017b. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*, pages 3295–3301.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *ACL*, pages 1577–1586.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112.

Qi Wu, Peng Wang, Chunhua Shen, Ian D Reid, and Anton Van Den Hengel. 2018. Are you talking to me? reasoned visual dialog generation through adversarial learning. In *CVPR*, pages 6106–6115.

Zhen Xu, Bingquan Liu, Baoxun Wang, Chengjie Sun, Xiaolong Wang, Zhuoran Wang, and Chao Qi. 2017. Neural response generation via gan with an approximate embedding layer. In *EMNLP*, pages 617–626.

Steve Young, Milica Gasic, Blaise Thomson, and J. D. Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

Hainan Zhang, Yanyan Lan, Jiafeng Guo, Jun Xu, and Xueqi Cheng. 2018. Reinforcing coherence for sequence to sequence model in dialogue generation. In *IJCAI*, pages 4567–4573.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. *ACL*, 1:654–664.

Hao Zhou, Minlie Huang, Tianyang Zhang, Xiaoyan Zhu, and Bing Liu. 2018. Emotional chatting machine: Emotional conversation generation with internal and external memory. In *AAAI*, pages 730–739.

Wenya Zhu, Kaixiang Mo, Yu Zhang, Zhangbin Zhu, Xuezheng Peng, and Qiang Yang. 2017. Flexible end-to-end dialogue system for knowledge grounded conversation. *CoRR*, abs/1709.04264.

# Knowledge Aware Conversation Generation with Explainable Reasoning over Augmented Graphs

**Zhibin Liu, Zheng-Yu Niu, Hua Wu, Haifeng Wang**
Baidu Inc., Beijing, China
{liuzhibin05,niuzhengyu,wu_hua, wanghaifeng}@baidu.com

## Abstract

Two types of knowledge, triples from knowledge graphs and texts from documents, have been studied for knowledge aware open-domain conversation generation, in which graph paths can narrow down vertex candidates for knowledge selection decision, and texts can provide rich information for response generation. Fusion of a knowledge graph and texts might yield mutually reinforcing advantages, but there is less study on that. To address this challenge, we propose a knowledge aware chatting machine with three components, an *augmented knowledge graph* with both triples and texts, knowledge selector, and knowledge aware response generator. For knowledge selection on the graph, we formulate it as a problem of *multi-hop graph reasoning* to effectively capture conversation flow, which is more explainable and flexible in comparison with previous work. To fully leverage long text information that differentiates our graph from others, we improve a state of the art reasoning algorithm with *machine reading comprehension technology*. We demonstrate the effectiveness of our system on two datasets in comparison with state-of-the-art models[1].

## 1 Introduction

One of the key goals of AI is to build a machine that can talk with humans when given an initial topic. To achieve this goal, the machine should be able to understand language with background knowledge, recall knowledge from memory or external resource, reason about these concepts together, and finally output appropriate and informative responses. Lots of research efforts have been devoted to chitchat oriented conversation generation (Ritter et al., 2011; Shang et al., 2015).

However, these models tend to produce generic responses or incoherent responses for a given topic, since it is quite challenging to learn semantic interactions merely from dialogue data without help of background knowledge. Recently, some previous studies have been conducted to introduce external knowledge, either unstructured knowledge texts (Ghazvininejad et al., 2018; Vougiouklis et al., 2016) or structured knowledge triples (Liu et al., 2018; Young et al., 2018; Zhou et al., 2018) to help open-domain conversation generation by producing responses conditioned on selected knowledge.

In the first research line, their knowledge graph can help narrowing down knowledge candidates for conversation generation with the use of prior information, e.g., triple attributes or graph paths. Moreover, these prior information can enhance generalization capability of knowledge selection models. But it suffers from information insufficiency for response generation since there is simply a single word or entity to facilitate generation. In the second line, their knowledge texts, e.g., comments about movies, can provide rich information for generation, but its unstructured representation scheme demands strong capability for models to perform knowledge selection or attention from the list of knowledge texts. Fusion of graph structure and knowledge texts might yield mutually reinforcing advantages for knowledge selection in dialogue systems, but there is less study on that.

To bridge the gap between the two lines of studies mentioned above, we present an Augmented Knowledge Graph based open-domain Chatting Machine (denoted as **AKGCM**), which consists of knowledge selector and knowledge aware response generator. This two-stage architecture and graph based knowledge selection make our system to be explainable. Explainability is very important

[1]Data and codes are available at https://github.com/PaddlePaddle/models/tree/develop/PaddleNLP/Research/EMNLP2019-AKGCM



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1782–1792,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

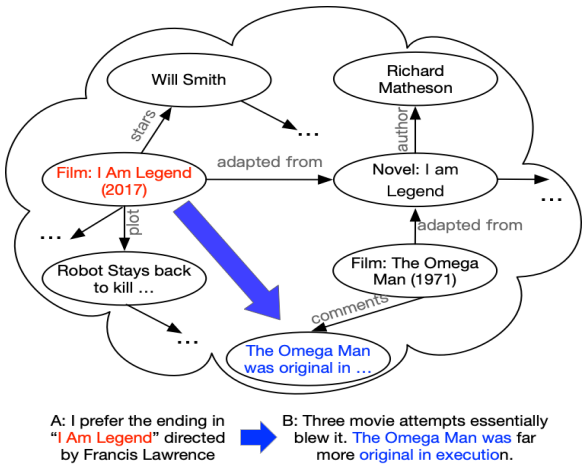


Figure 1: Sample conversation with explainable reasoning, reflecting conversation flow, on an augmented knowledge graph.

in e.g., information oriented chatting scenarios, where a user needs to know how new knowledge in chatbot's responses is linked to the knowledge in their utterances, or business scenarios, where a user cannot make a business decision without justification.

To integrate texts into a knowledge graph, we take a factoid knowledge graph (KG) as its backbone, and align unstructured sentences of non-factoid knowledge with the factoid KG by linking entities from these sentences to vertices (containing entities) of the KG. Thus we augment the factoid KG with non-factoid knowledge, and retain its graph structure. Then we use this augmented KG to facilitate knowledge selection and response generation, as shown in Figure 1.

For knowledge selection on the graph, we adopt a deep reinforcement learning (RL) based reasoning model(Das et al., 2018), MINERVA, in which the reasoning procedure greatly reflects conversation flow as shown in Figure 1. It is as robust as embedding based neural methods, and is as explainable as path based symbolic methods. Moreover, our graph differs from previous KGs in that: some vertices in ours contain long texts, not a single entity or word. To fully leverage this long text information, we improve the reasoning algorithm with machine reading comprehension (MRC) technology (Seo et al., 2017) to conduct fine-grained semantic matching between an input message and candidate vertices.

Finally, for response generation, we use an encoder-decoder model to produce responses conditioned on selected knowledge.

In summary, we make following contributions:

- This work is the first attempt that unifies knowledge triples and texts as a graph, and conducts flexible multi-hop knowledge graph reasoning in dialogue systems. Supported by such knowledge and knowledge selection method, our system can respond more appropriately and informatively.
- Our two-stage architecture and graph based knowledge selection mechanism provide better model explainability, which is very important for some application scenarios.
- For knowledge selection, to fully leverage long texts in vertices, we integrate machine reading comprehension (MRC) technology into the graph reasoning process.

## 2 Related Work

**Conversation with Knowledge Graph:** There are growing interests in leveraging factoid knowledge (Han et al., 2015; Liu et al., 2018; Zhu et al., 2017) or commonsense knowledge (Young et al., 2018; Zhou et al., 2018) with graph based representation for generation of appropriate and informative responses. Compared with them, we augment previous KGs with knowledge texts and integrate more explainable and flexible multi-hop graph reasoning models into conversation systems. Wu et al. (2018) used document reasoning network for modeling of conversational contexts, but not for knowledge selection.

**Conversation with Unstructured Texts:** With availability of a large amount of knowledge texts from Wikipedia or user generated content, some work focus on either modeling of conversation generation with unstructured texts (Ghazvininejad et al., 2018; Vougiouklis et al., 2016; Xu et al., 2017), or building benchmark dialogue data grounded on knowledge (Dinan et al., 2019; Moghe et al., 2018). In comparison with them, we adopt a graph based representation scheme for unstructured texts, which enables better explainability and generalization capability of our system.

**Knowledge Graph Reasoning:** Previous studies on KG reasoning can be categorized into three lines, path-based symbolic models (Das et al., 2017a; Lao et al., 2011), embedding-based neural models (Bordes et al., 2013; Wang et al., 2014), and models in unifying embedding and path-based

technology (Das et al., 2018; Lin et al., 2018; Xiong et al., 2017), which can predict missing links for completion of KG. In this work, for knowledge selection on a graph, we follow the third line of works. Furthermore, our problem setting is different from theirs in that some of our vertices contain long texts, which motivates the use of machine reading technology for graph reasoning.

**Fusion of KG triples and texts:** In the task of QA, combination of a KG and a text corpus has been studied with a strategy of late fusion (Gardner and Krishnamurthy, 2017; Ryu et al., 2014) or early fusion (Das et al., 2017b; Sun et al., 2018), which can help address the issue of low coverage to answers in KG based models. In this work, we conduct this fusion for conversation generation, not QA, and our model can select sentences as answers, not restricted to entities in QA models.

## 3 The Proposed Model

### 3.1 Problem Definition and Model Overview

Our problem is formulated as follows: Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{L}^{\mathcal{E}}\}$ be an augmented KG, where $\mathcal{V}$ is a set of vertices, $\mathcal{E}$ is a set of edges, and $\mathcal{L}^{\mathcal{E}}$ is a set of edge labels (e.g., triple attributes, or vertex categories). Given a message $X = \{x_1, x_2, ..., x_m\}$ and $\mathcal{G}$, the goal is to generate a proper response $Y = \{y_1, y_2, ..., y_n\}$ with supervised models. Essentially, the system consists of two stages: (1) knowledge selection: we select the vertex that maximizes following probability as an answer, which is from vertex candidates connected to $v_X$:

$$v_Y = \arg\max_v P_{KS}(v|v_X, \mathcal{G}, X). \quad (1)$$

$v_X$ is one of vertices retrieved from $\mathcal{G}$ using the entity or words in $X$, and it is ranked as top-1 based on text similarity with $X$. Please see Equation 8 and 10 for computation of $P_{KS}(*)$; (2) response generation: it estimates the probability:

$$P_{RG}(Y|X, v_Y) = \prod_{t=1}^{n} P(y_t|y_{<t}, X, v_Y). \quad (2)$$

The overview of our Augmented Knowledge Graph based Chatting Machine (AKGCM) is shown in Figure 2. The knowledge selector first takes as input a message $X = \{x_1, x_2, ..., x_m\}$ and retrieves a starting vertex $v_X$ from $\mathcal{G}$ that is closely related to $X$, and then performs multi-hop graph reasoning on $\mathcal{G}$ and finally arrives at a vertex $v_Y$ that has the knowledge being appropriate for response generation. The knowledge aware response generator produces a response $Y = \{y_1, y_2, ..., y_n\}$ with knowledge from $v_Y$. At each decoding position, it attentively reads the selected knowledge text, and then generates a word in the vocabulary or copies a word in the knowledge text.

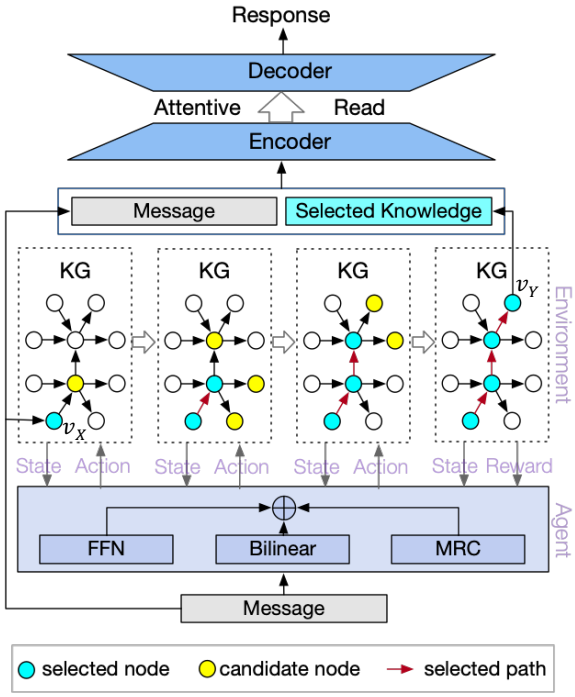


Figure 2: The architecture of AKGCM.

For model training, each pair of [message, response] in training data is associated with ground-truth knowledge and its vertex ID (ground-truth vertex) in $\mathcal{G}$ for knowledge grounding. These vertex IDs will be used as ground-truth for training of knowledge selector, while the triples of [message, knowledge text, response] will be used for the training of knowledge aware generator.

### 3.2 Augmented Knowledge Graph

Given a factoid KG and related documents containing non-factoid knowledge, we take the KG as a backbone, where each vertex contains a single entity or word, and each edge represents an attribute or a relation. Then we segment the documents into sentences and align each sentence with entries of the factoid KG by mapping entities from these sentences to entity vertices of the KG. Thus we augment the factoid KG with non-factoid knowledge, and retain its structured representation.

### 3.3 Knowledge Selection on the Graph

**Task Definition:** We formulate knowledge selection on $\mathcal{G}$ as a finite horizon sequential decision making problem. It supports more flexible multi-hop walking on graphs, not restricted to one-hop walking as done in previous work (Han et al., 2015; Zhou et al., 2018; Zhu et al., 2017).

As shown in Figure 2, we begin by representing the environment as a deterministic partially observed Markov decision process (POMDP) on $\mathcal{G}$ built in Section 3.2. Our RL based agent is given an input query of the form $(v_X, X)$. Starting from vertex $v_X$ corresponding to $X$ in $\mathcal{G}$, the agent follows a path in the graph, and stops at a vertex that it predicts as the answer $v_Y$. Using a training set of known answer vertices for message-response pairs, we train the agent using policy gradients (Williams, 1992) with control variates.

The difference between the setting of our problem and previous KG reasoning lies in that: (1) the content of our input queries is not limited to entities and attributes; (2) some vertices in our graph contains long texts, while vertices in previous KGs just contain a single entity or short text. It motivates us to make a few improvements on previous models, as shown in Equation (5), (6), and (7).

Next we elaborate the 5-tuple $(\mathcal{S}, \mathcal{O}, \mathcal{A}, \delta, \mathcal{R})$ of the environment, and policy network.

**States:** A state $S_t \in \mathcal{S}$ at time step $t$ is represented by $S_t = (v_t, v_X, X, v_{gt})$ and the state space consists of all valid combinations in $\mathcal{V}\times\mathcal{V}\times\mathcal{X}\times\mathcal{V}$, where $v_t$ is current location of the RL agent, $v_{gt}$ is the ground-truth vertex, and $\mathcal{X}$ is the set of all possible $X$.

**Observations:** The complete state of the environment cannot be observed. Intuitively, the agent knows its current location $(v_t)$ and $(v_X, X)$, but not the ground-truth one $(v_{gt})$, which remains hidden. Formally, the observation function $\mathcal{O} : \mathcal{S}\rightarrow\mathcal{V} \times \mathcal{V} \times \mathcal{X}$ is defined as $\mathcal{O}(S_t = (v_t, v_X, X, v_{gt})) = (v_t, v_X, X)$.

**Actions:** The set of possible actions $\mathcal{A}_{S_t}$ from a state $S_t$ consists of all outgoing edges of the vertex $v_t$ in $\mathcal{G}$. Formally $\mathcal{A}_{S_t} = \{(v_t, l_e, v_d) \in \mathcal{E} : S_t = (v_t, v_X, X, v_{gt}), l_e \in \mathcal{L}^{\mathcal{E}}, v_d \in \mathcal{V}\} \cup \{(S_t, \emptyset, S_t)\}$. It means an agent at each state has option to select which outgoing edge it wishes to take with the label of the edge $l_e$ and destination vertex $v_d$. We limit the length of the action sequence (horizon length) up to a fixed number (e.g., $T$) of time steps. Moreover, we augment each vertex with a special action called 'NO_OP' which goes from a vertex to itself. This decision allows the agent to remain at a vertex for any number of time steps. It is especially helpful when the agent has managed to reach a correct vertex at a time step $t < T$ and can continue to stay at the vertex for the rest of the time steps.

**Transition:** The environment evolves deterministically by just updating the state to the new vertex according to the edge selected by the agent. Formally, the transition function $\delta : \mathcal{S} \times \mathcal{A}\rightarrow\mathcal{S}$ is defined by $\delta(S_t, A) = (v_d, v_X, X, v_{gt})$, where $S_t = (v_t, v_X, X, v_{gt})$ and $A = (v_t, l_e, v_d)$. $l_e$ is the label of an edge connecting $v_t$ and $v_d$, and $v_d$ is destination vertex.

**Rewards:** After $T$ time steps, if the current vertex is the ground-truth one, then the agent receives a reward of $+1$ otherwise 0. Formally, $\mathcal{R}(S_T) = I\{v_T = v_{gt}\}$, where $S_T = (v_T, v_X, X, v_{gt})$ is the final state.

**Policy Network:** We design a randomized non-stationary policy $\boldsymbol{\pi} = (\boldsymbol{d_0}, \boldsymbol{d_1}, ..., \boldsymbol{d_{T-1}})$, where $d_t = P(\mathcal{A}_{S_t})$ is a policy at time step $t$. In this work, for each $d_t$, we employ *a policy network with three components* to make the decision of choosing an action from all available actions $(\mathcal{A}_{S_t})$ conditioned on $X$.

The first component is a history dependent *feed-forward network* (FFN) based model proposed in (Das et al., 2018). We first employ a LSTM to encode the history $H_t = (H_{t-1}, A_{t-1}, O_t)$ as a continuous vector $\boldsymbol{h_t} \in \mathbb{R}^{2d}$, where $H_t$ is the sequence of observations and actions taken. It is defined by:

$$\boldsymbol{h_t} = LSTM(\boldsymbol{h_{t-1}}, [\boldsymbol{a_{t-1}}; \boldsymbol{o_t}]), \qquad (3)$$

where $\boldsymbol{a_{t-1}}$ is the embedding of the relation corresponding to the label of the edge the agent chose at time $t-1$ and $\boldsymbol{o_t}$ is the embedding of the vertex corresponding to the agent's state at time $t$.

Recall that each possible action represents an outgoing edge with information of the edge relation label $l_e$ and destination vertex $v_d$. So let $[\boldsymbol{l_e}; \boldsymbol{v_d}]$ denote an embedding for each action $A \in \mathcal{A}_{S_t}$, and we obtain the matrix $\boldsymbol{A_t}$ by stacking embeddings for all the outgoing edges. Then we build a two-layer feed-forward network with ReLU nonlinearity which takes in the current history representation $h_t$ and the representation of $X$ $(\boldsymbol{e_X^{new}})$. We use another single-layer *feed-forward network* for computation of $\boldsymbol{e_X^{new}}$, which accepts the original sentence embedding of $X$ $(\boldsymbol{e_X})$ as input. The

updated FFN model for action decision is defined by:

$$P_{FFN}(\mathcal{A}_{S_t}) = \boldsymbol{A_t}(\boldsymbol{W_2}ReLU(\boldsymbol{W_1}[\boldsymbol{h_t}; \boldsymbol{o_t}; \boldsymbol{e_X^{new}}] + \boldsymbol{b_1}) + \boldsymbol{b_2}), \tag{4}$$

$$\boldsymbol{e_X^{new}} = ReLU(\boldsymbol{W_X}\boldsymbol{e_X} + \boldsymbol{b_X}). \tag{5}$$

Recall that in our graph, some vertices contain long texts, differentiating our graph from others in previous work. The original reasoning model (Das et al., 2018), MINERVA, cannot effectively exploit the long text information within vertices since it just learns embedding representation for the whole vertex, without detailed analysis of text in vertices. To fully leverage the long text information in vertices, we employ two models, *a machine reading comprehension model* (MRC) (Seo et al., 2017) and *a bilinear model*, to score each possible $v_d$ from both global and local view.

For scoring from global view, (1) we build a document by collecting sentences from all possible $v_d$, (2) we employ the MRC model to predict an answer span ($span_{aw}$) from the document, (3) we score each $v_d$ by calculating a ROUGE-L score vector of $v_d$'s sentence with $span_{aw}$ as the reference, shown as follows:

$$P_{MRC}(\mathcal{A}_{S_t}) = ROUGE(Text(V_d), span_{aw}). \tag{6}$$

Here, $Text(\cdot)$ represents operation of getting text contents, and $ROUGE(\cdot)$ represents operation of calculating ROUGE-L score. We see that the MRC model can help to determine which $v_d$ is the best based on global information from the whole document.

For scoring from local view, we use another *bilinear model* to calculate similarity between $X$ and $v_d$, shown as follows:

$$P_{Bi}(\mathcal{A}_{S_t}) = \boldsymbol{V_d}\boldsymbol{W_B}\boldsymbol{e_X}. \tag{7}$$

Finally, we calculate a sum of outputs of the three above-mentioned models and outputs a probability distribution over the possible actions from which a discrete action is sampled, defined by:

$$P(\mathcal{A}_{S_t}) = softmax(\alpha P_{FFN}(\mathcal{A}_{S_t}) + \beta P_{Bi}(\mathcal{A}_{S_t}) + \gamma P_{MRC}(\mathcal{A}_{S_t})), \tag{8}$$

$$A_t \sim Sample(P(\mathcal{A}_{S_t})), \tag{9}$$

$$P_{KS}(v_d|v_X, \mathcal{G}, X) = P(\mathcal{A}_{S_{T-1}}). \tag{10}$$

Please see Section 3.1 for definition of $P_{KS}(*)$. When the agent finally arrives at $S_T$, we obtain $v_T$ as the answer $v_Y$ for response generation.

**Training:** For the policy network ($\pi_\theta$) described above, we want to find parameters $\theta$ that maximize the expected reward:

$$J(\theta) = \mathbb{E}_{(v_0,X,v_{gt})\sim D}\mathbb{E}_{A_0,\ldots,A_{T-1}\sim\pi_\theta}[\mathcal{R}(S_T)|S_0 = (v_0, v_0, X, v_{gt})], \tag{11}$$

where we assume there is a true underlying distribution $D$, and $(v_0, X, v_{gt}) \sim D$.

### 3.4 Knowledge Aware Generation

Following the work of Moghe et al. (2018), we modify a text summarization model (See et al., 2017) to suit this generation task.

In the summarization task, its input is a document and its output is a summary, but in our case the input is a [selected knowledge, message] pair and the output is a response. Therefore we introduce two RNNs: one is for computing the representation of the selected knowledge, and the other for the message. The decoder accepts the two representations and its own internal state representation as input, and then compute (1) a probability score which indicates whether the next word should be generated or copied, (2) a probability distribution over the vocabulary if the next word needs to be generated, and (3) a probability distribution over the input words if the next word needs to be copied. These three probability distributions are then combined, resulting in $P(y_t|y_{<t}, X, v_Y)$, to produce the next word in the response.

## 4 Experiments and Results

### 4.1 Datasets

We adopt two knowledge grounded multi-turn dialogue datasets for experiments, shown as follows:

**EMNLP dialog dataset (Moghe et al., 2018)** This Reddit dataset contains movie chats from two participants, wherein each response is explicitly generated by copying or modifying sentences from background knowledge such as IMDB's facts/plots, or Reddit's comments about movies. We follow their data split for training, validation and test[2]. Their statistics can be seen in Table 1.

**ICLR dialog dataset (Dinan et al., 2019)** This wizard-of-wiki dataset contains multi-turn conver-

[2]We use the single-reference mixed-short test set for evaluation. Please see their paper for more details.

| EMNLP dialog dataset | | | |
|---|---|---|---|
| Conversational Pairs | | Augmented KG | |
| #Train. pairs | 34486 | #Vertices | 117373 |
| #Valid. pairs | 4388 | #Relations | 11 |
| #Test pairs | 4318 | #Triples | 251138 |
| Factoid know. | | Non-f. know. | |
| #Total Ver. | 21028 | #Total Ver. | 96345 |
| #Used Ver. | 2620 | #Used Ver. | 27586 |
| ICLR dialog dataset | | | |
| Conversational Pairs | | Augmented KG | |
| #Train. pairs | 19484 | #Vertices | 290075 |
| #Valid. pairs | 1042 | #Relations | 5 |
| #Test pairs | 1043 | #Triples | 2570227 |
| Factoid know. | | Non-f. know. | |
| #Total Ver. | NA | #Total Ver. | 290075 |
| #Used Ver. | NA | #Used Ver. | 10393 |

Table 1: The upper/lower two tables show statistics of the EMNLP/ICLR datasets and corresponding augmented knowledg graphs. #Used Ver. means total num of vertices that used for response generation. For more details, please visit our data sharing URL.

sations from two participants. One participant selects a beginning topic, and during the conversation the topic is allowed to naturally change. The two participants are not symmetric: one will play the role of a knowledgeable expert while the other is a curious learner. We filter their training data and test data by removing instances without the use of knowledge and finally keep 30% instances[3] for our study since we focus on knowledge selection and knowledge aware generation. Their statistics can be seen in Table 1. For models (Seq2Seq, HRED) without the use of knowledge, we keep the original training data for them.

### 4.2 Experiment Settings

We follow the existing work to conduct both automatic evaluation and human evaluation for our system. We also compare our system with a set of carefully selected baselines, shown as follows.

**Seq2Seq:** We implement a sequence-to-sequence model (Seq2Seq) (Sutskever et al., 2014), which is widely used in open-domain conversational systems.

**HRED:** We implement a hierarchical recurrent encoder-decoder model (Serban et al., 2016).

**MemNet:** We implement an end-to-end knowledge-MemNet based conversation model (Ghazvininejad et al., 2018).

**GTTP:** It is an end-to-end text summarization model (See et al., 2017) studied on the EMNLP data. We use the code[4] released by Moghe et al. (2018), where they modify GTTP to suit knowledge aware conversation generation.

**BiDAF+G:** It is a Bi-directional Attention Flow based QA Model (BiDAF) (Seo et al., 2017) that performs best on the EMNLP dataset. We use the code[4] released by Moghe et al. (2018), where they use it to find the answer span from a knowledge document, taking the input message as the query. Moreover, we use a response generator (as same as ours) for NLG with the predicted knowledge span.

**TMemNet:** It is a two-stage transformer-MemNet based conversation system that performs best on the ICLR dataset (Dinan et al., 2019). We use the code[5] released by the original authors.

**CCM:** It is a state-of-the-art knowledge graph based conversation model (Zhou et al., 2018). We use the code[6] released by the original authors and then modify our graph to suit their setting by selecting each content word from long text as an individual vertex to replace our long-text vertices.

**AKGCM:** It is our two-stage system presented in Section 3. We implement our knowledge selection model based on the code[7] by(Das et al., 2018) and that[4] by (Moghe et al., 2018). We use BiDAF as the MRC module, shown in Equation (6), and we train the MRC module on the same training set for our knowledge selection model. We implement the knowledge aware generation model based on the code of GTTP[4] released by (Moghe et al., 2018). We also implement a variant AKGCM-5, in which top five knowledge texts are used for generation, and other setting are not changed.

### 4.3 Automatic Evaluations

**Metrics:** Following the work of (Moghe et al., 2018), we adopt BLEU-4 (Papineni et al., 2002), ROUGE-2 (Lin, 2004) and ROUGE-L (Lin and Och, 2004) to evaluate how similar the output response is to the reference text. We use Hit@1 (the top 1 accuracy) to evaluate the performance of knowledge selection.

[3] Their ground-truth responses should have high ROUGE-L scores with corresponding ground-truth knowledge texts.

[4] https://github.com/nikitacs16/Holl-E

[5] https://parl.ai/projects/wizard_of_wikipedia/

[6] https://github.com/tuxchow/ccm

[7] https://github.com/shehzaadzd/MINERVA

| Model | EMNLP dialog dataset | | | | ICLR dialog dataset | | | |
|---|---|---|---|---|---|---|---|---|
| | BLEU-4 (%) | ROUGE-2 (%) | ROUGE-L (%) | Hit@1 (%) | BLEU-4 (%) | ROUGE-2 (%) | ROUGE-L (%) | Hit@1 (%) |
| Seq2seq | 1.59 | 5.73 | 14.49 | NA | 0.17 | 1.01 | 7.02 | NA |
| HRED | 2.08 | 8.83 | 18.13 | NA | 0.23 | 1.08 | 7.32 | NA |
| MemNet | 5.86 | 10.64 | 18.48 | NA | 0.89 | 2.33 | 11.84 | NA |
| GTTP | 11.05 | 17.70 | 25.13 | NA | 6.74 | 7.18 | **17.11** | NA |
| CCM | 2.40 | 4.84 | 17.70 | NA | 0.86 | 1.68 | 12.74 | NA |
| BiDAF+G | **32.45** | **31.28** | **36.95** | 40.80 | 6.48 | 6.54 | 15.56 | 17.40 |
| TMemNet | 8.92 | 13.15 | 19.97 | 38.10 | 1.09 | 1.86 | 8.51 | 16.80 |
| AKGCM-5 | 13.29 | 13.12 | 21.22 | **42.04** | **6.94** | **7.38** | 17.02 | **18.24** |
| AKGCM | 30.84 | 29.29 | 34.72 | **42.04** | 5.52 | 6.10 | 15.46 | **18.24** |

Table 2: Results of automatic evaluations on the two datasets.

| Model | EMNLP dialog dataset | | ICLR dialog dataset | |
|---|---|---|---|---|
| | Appr. * vs. AKGCM Win/Tie/Lose | Infor. * vs. AKGCM Win/Tie/Lose | Appr. * vs. AKGCM-5 Win/Tie/Lose | Infor. * vs. AKGCM-5 Win/Tie/Lose |
| Seq2seq | 0.04/0.42/0.54 | 0.05/0.21/0.74 | 0.00/0.10/0.90 | 0.00/0.11/0.89 |
| HRED | 0.03/0.50/0.47 | 0.03/0.27/0.70 | 0.01/0.14/0.85 | 0.01/0.14/0.85 |
| MemNet | 0.03/0.43/0.54 | 0.03/0.23/0.74 | 0.00/0.19/0.81 | 0.00/0.17/0.83 |
| GTTP | 0.03/0.52/0.45 | 0.10/0.42/0.48 | 0.07/0.73/0.20 | 0.12/0.68/0.20 |
| CCM | 0.01/0.18/0.81 | 0.01/0.15/0.84 | 0.00/0.17/0.83 | 0.00/0.16/0.84 |
| BiDAF+G | 0.04/0.83/0.13 | 0.07/0.79/0.14 | 0.04/0.61/0.35 | 0.04/0.56/0.40 |
| TMemNet | 0.04/0.50/0.46 | 0.05/0.36/0.59 | 0.01/0.25/0.74 | 0.01/0.21/0.78 |

Table 3: Results of human evaluations on the two datasets. AKGCM (or AKGCM-5) outperforms all the baselines significantly (sign test, p-value $< 0.05$) in terms of the two metrics.

**Results:** As shown in Table 2, AKGCM (or AKGCM-5) can obtain the highest score on test set in terms of Hit@1, and the second highest scores in terms of BLEU-4, ROUGE-2 and ROUGE-L, surpassing other models, except BiDAF, by a large margin. It indicates that AKGCM has a capability of knowledge selection better than BiDAF and TMemNet, and generates more informative and grammatical responses. We notice that from EMNLP dataset to ICLR dataset, there is a significant performance drop for almost all the models. It is probably due to that the quality of ICLR dataset is worse than that of EMNLP dataset. A common phenomenon of ICLR dataset is that the knowledge used in responses is loosely relevant to input messages, which increases the difficulty of model learning.

### 4.4 Human Evaluations

**Metrics:** We resort to a web crowdsourcing service for human evaluations. We randomly sample 200 messages from test set and run each model to generate responses, and then we conduct pair-wise comparison between the response by AKGCM and the one by a baseline for the same message. In total, we have 1400 pairs on each dataset since there are seven baselines. For each pair, we ask five evaluators to give a preference between the two responses, in terms of the following two metrics: (1) appropriateness (Appr.), e.g., whether the response is appropriate in relevance, and logic, (2) informativeness (Infor.), whether the response provides new information and knowledge in addition to the input message, instead of generic responses such as "This movie is amazing". Tie is allowed. Notice that system identifiers are masked during evaluation.

**Annotation Statistics:** We calculate the agreements to measure inter-evaluator consistency. For appropriateness, the percentage of test instances that at least 2 evaluators give the same label (2/3 agreement) is 98%, and that for at least 3/3 agreement is 51%. For informativeness, the percentage for at least 2/3 agreement is 98% and that for at

least 3/3 agreement is 55.5%.

**Results:** In Table 3, each score for win/tie/lose is the percentage of messages for which AKGCM (or AKGCM-5) can generate better/almost same/worse responses, in comparison with a baseline. We see that our model outperforms all the baselines significantly (sign test, p-value $< 0.05$) in terms of the two metrics on the two datasets. Furthermore, our model can beat the strongest baseline, BiDAF. It demonstrates the effectiveness of our graph reasoning mechanism that can use global graph structure information and exploit long text information. Our data analysis shows that both Seq2Seq and HRED tend to generate safe responses starting with "my favorite character is" or "I think it is". Both Memnet and TMemnet can generate informative responses. But the knowledge in their responses tends to be incorrect, which is a serious problem for knowledge aware conversation generation. Our results show that GTTP and BiDAF are very strong baselines. It indicates that the attention mechanism (from machine reading) for knowledge selection and the copy mechanism can bring benefits for knowledge aware conversation generation. Although CCM have the mechanisms mentioned above, this model is good at dealing with structured triples rather than long texts. It may explain the inferior performance of CCM in our problem setting.

We find that many responses are likely to be a simple copy of the selected knowledge, which is a reflection of the characteristics of the datasets. In the two datasets, some sentences in the background knowledge are directly used as responses to the messages. Therefore, the NLG module is likely to copy content from the selected knowledge as much as possible for generation. Moreover, the summarization model GTTP tends to copy words from its input message as its output due to its generation mechanism. Table 5 presents the examples in which AKGCM (AKGCM-5) performs better than other models on two dataset.

### 4.5 Model Analysis

**AKGCM without (w/o) Non-factoid Knowledge:** To verify contribution of non-factoid knowledge, we remove non-factoid knowledge from augmented KG in test procedure, and report the performance of our system with only factoid knowledge in Table 4. We see that without non-factoid knowledge from EMNLP dataset, the performance of our system drops significantly in terms of BLEU and ROUGE. It indicates that non-factoid knowledge is essential for knowledge aware conversation generation.

| Model variant | BLEU-4 (%) | ROUGE-2 (%) | ROUGE-L (%) |
|---|---|---|---|
| w/o non-factoid knowledge | 0.61 | 0.84 | 1.52 |
| w/o Bilinear + MRC | 15.12 | 14.15 | 21.09 |
| w/o MRC | 18.41 | 17.79 | 24.59 |
| w/o Bilinear | 28.62 | 28.19 | 32.84 |
| Full model | 30.84 | 29.29 | 34.72 |

Table 4: Results of ablation study for AKGCM on EMNLP dataset. We also include results of the full model for comparison.

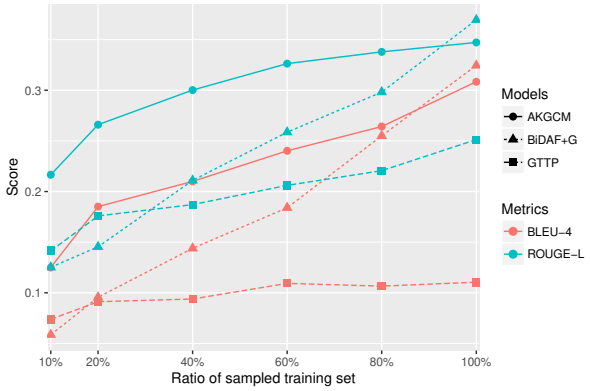


Figure 3: Results of how each model performs on EMNLP dataset when we gradually reduce the ratio of sampled training data from 100% to 10% (from right to left).

**AKGCM w/o the MRC Model or Bilinear One:** For ablation study, we implement a few variants of our system without the bilinear model or MRC for knowledge selection. Results of these variants are reported in Table 4. If we compare the performance of our full model with its variants, we find that both MRC and the bilinear model can bring performance improvement to our system. It indicates that the full interaction between messages and knowledge texts by their attention mechanism is effective to knowledge selection.

**Model Generalization:** As shown in Figure 3, we gradually reduce the size of training data, and then AKGCM can still manage to achieve acceptable performance, even when given extremely small training data (around 3,400 u-r pairs at the x-axis point of 10%). But the performance of

| Dataset | EMNLP dataset (Movie Name: Monsters University) | ICLR dataset (Topic Name: Green) |
|---|---|---|
| Message | I love comic scene in movies though it makes me want to see it over again | How is green color produced? |
| Ground-truth response | I will be definitely be seeing it a second and possibly third time . | Green is evoked by light, a dominant wavelength of about 495570 nm. it is between blue and yellow. |
| Seq2seq | I think it 's just plain funny | Yellow is the color between green and orange on the spectrum of visible light |
| HRED | A stone cold classic ! | There are different types of UNK there are different types of UNK |
| MemNet | I thought it was a great movie, but it is a great film. | It lies between violet and green on the spectrum of light. |
| GTTP | I think this movie was worth it , for i will be definitely be seeing it a second and possibly third time. | Yes , green is the color between blue and yellow on the visible spectrum . |
| CCM | I think it was a worth _UNK , but I still enjoyed it intense than the fraternities . | The energy is chemical on the Several minerals and tradition . |
| BiDAF+G | It not only serves as a great climax , but something that we can once again commend Pixar for telling great stories in out of the box thinking that we should think paying for the $ 10 dollar ticket is worth it . | It 's the color between blue and the color yellow . |
| Selected knowledge | It not only serves as a great climax, but something that we can once again commend Pixar for telling great stories in out of the box thinking that we should think paying for the $10 dollar ticket is worth it. And boy did I think this movie was worth it, ... | Green is the color between blue and yellow on the visible spectrum. |
| TMemNet | Man and i think this movie was worth it , i will definitely be seeing if it was a second time . | It ' s been around since the middle ages , it was associated with royalty . |
| AKGCM (AKGCM-5) | And boy did I think this movie was worth it , for I will be definitely be seeing it a second and possibly third time . | Green is evoked by light which has a dominant wavelength of roughly 495570 nm . |
| Selected knowledge | And boy did I think this movie was worth it, for I will be definitely be seeing it a second and possibly third time. | ... It is evoked by light which has a dominant wavelength of roughly 495570 nm. Several minerals have a green color, including the emerald, ... |

Table 5: Examples in which AKGCM performs better than other models on two dataset.

the strongest baseline, BIDAF+G, drops more dramatically in comparison with AKGCM. It indicates that our graph reasoning mechanism can effectively use the graph structure information for knowledge selection, resulting in better generalization capability of AKGCM.

**Model Explainability:** We check the graph paths traversed by our system for knowledge selection and try to interpret what heuristics have been learned. We find that our system can learn to visit different types of vertices conditioned on conversational contexts, e.g.. selecting comment vertices as responses for utterances starting with "what do you think". These results suggest that AKGCM may also be a good assistive tool for discovering new algorithms, especially in cases when the graph reasoning are new and less well-studied for conversation modeling.

## 5 Conclusion

In this paper, we propose to augment a knowledge graph with texts and integrate it into an open-domain chatting machine with both graph reasoning based knowledge selector and knowledge aware response generator. Experiments demonstrate the effectiveness of our system on two datasets compared to state-of-the-art approaches.

## Acknowledgments

We would like to thank the reviewers for their insightful comments. This work was supported by the Natural Science Foundation of China (No.61533018).

## References


Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multirelational data. In *Proceedings of NIPS*, pages 2787—-2795.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2018. Go for a walk and arrive at the answer: reasoning over paths in knowledge bases using reinforcement learning. In *Proceedings of ICLR*, pages 1–18.

Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. 2017a. Chains of reasoning over entities, relations, and text using recurrent neural networks. In *Proceedings of EACL*, pages 132—-141.

Rajarshi Das, Manzil Zaheer, Siva Reddy, and Andrew McCallum. 2017b. Question answering on knowledge bases and text using universal schema and memory networks. In *Proceedings of ACL*, pages 358—-365.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2019. Wizard of wikipedia: knowledge-powered conversational agents. In *Proceedings of ICLR*.

Matt Gardner and Jayant Krishnamurthy. 2017. Open-vocabulary semantic parsing with both distributional statistics and formal knowledge. In *Proceedings of AAAI*, pages 3195—-3201.

Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen tau Yih, and Michel Galley. 2018. A knowledge-grounded neural conversation model. In *Proceedings of AAAI 2018*, pages 5110–5117.

Sangdo Han, Jeesoo Bang, Seonghan Ryu, and Gary Geunbae Lee. 2015. Exploiting knowledge base to generate responses for natural language dialog listening agents. In *Proceedings of SIGDIAL*, pages 129—-133.

Ni Lao, Tom M. Mitchell, and William W. Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of EMNLP*, pages 529—-539.

Chin-Yew Lin. 2004. Rouge: a package for automatic evaluation of summaries. In *Proceedings of the Workshop on Text Summarization Branches Out (WAS 2004)*.

Chin-Yew Lin and Franz Josef Och. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of ACL*, pages 605–612.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2018. Multi-hop knowledge graph reasoning with reward shaping. In *Proceedings of EMNLP*, pages 3243—-3253.

Shuman Liu, Hongshen Chen, Zhaochun Ren, Yang Feng, Qun Liu, and Dawei Yin. 2018. Knowledge diffusion for neural dialogue generation. In *Proceedings of ACL*, pages 1489—-1498.

Nikita Moghe, Siddhartha Arora, Suman Banerjee, and Mitesh M. Khapra. 2018. Towards exploiting background knowledge for building conversation systems. In *Proceedings of EMNLP*, pages 2322—-2332.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*, pages 311–318.

Alan Ritter, Colin Cherry, and William B. Dolan. 2011. Data-driven response generation in social media. In *Proceedings of EMNLP*, pages 583—-593.

Pum-Mo Ryu, Myung-Gil Jang, and Hyun-Ki Kim. 2014. Open domain question answering using wikipedia-based knowledge model. In *Information Processing and Management*, pages 50(5):683–692.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: summarization with pointer-generator networks. In *Proceedings of ACL*, pages 1073—-1083.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *Proceedings of ICLR*.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of AAAI*, pages 3776—-3784.

Lifeng Shang, Zheng dong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of ACL*, pages 1577—-1586.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William W. Cohen. 2018. Open domain question answering using early fusion of knowledge bases and text. In *Proceedings of EMNLP*, pages 4231—-4242.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of NIPS*, pages 3104—-3112.

Pavlos Vougiouklis, Jonathon Hare, and Elena Simperl. 2016. A neural network approach for knowledge-driven response generation. In *Proceedings of COLING 2016*, pages 3370—-3380.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of AAAI*, pages 1112—-1119.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Machine learning*, pages 8(3–4):229—-256.

Xianchao Wu, Ander Martınez, and Momo Klyen. 2018. Dialog generation using multi-turn reasoning neural networks. In *Proceedings of NAACL-HLT*, pages 2049—-2059.

Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. Deeppath: a reinforcement learning method for knowledge graph reasoning. In *Proceedings of EMNLP*, pages 564–573.

Zhen Xu, Bingquan Liu, Baoxun Wang, Chengjie Sun, and Xiaolong Wang. 2017. Incorporating loose-structured knowledge into lstm with recall gate for conversation modeling. In *Proceedings of IJCNN*, pages 3506–3513.

Tom Young, Erik Cambria, Iti Chaturvedi, Hao Zhou, Subham Biswas, and Minlie Huang. 2018. Augmenting end-to-end dialogue systems with commonsense knowledge. In *Proceedings of AAAI 2018*, pages 4970–4977.

Hao Zhou, Tom Young, Minlie Huang, Haizhou Zhao, Jingfang Xu, and Xiaoyan Zhu. 2018. Commonsense knowledge aware conversation generation with graph attention. In *Proceedings of IJCAI-ECAI*, pages 4623–4629.

Wenya Zhu, Kaixiang Mo, Yu Zhang, Zhangbin Zhu, Xuezheng Peng, and Qiang Yang. 2017. Flexible end-to-end dialogue system for knowledge grounded conversation. In *CoRR, abs/1709.04264*.

# Adaptive Parameterization for Neural Dialogue Generation

**Hengyi Cai**[†,§]*, **Hongshen Chen**[‡], **Cheng Zhang**[†], **Yonghao Song**[†], **Xiaofang Zhao**[†] and **Dawei Yin**[‡]
[†]Institute of Computing Technology, Chinese Academy of Sciences
[§]University of Chinese Academy of Sciences
[‡]Data Science Lab, JD.com
caihengyi@ict.ac.cn, ac@chenhongshen.com,
{zhangcheng, songyonghao, zhaoxf}@ict.ac.cn, yindawei@acm.org

## Abstract

Neural conversation systems generate responses based on the sequence-to-sequence (SEQ2SEQ) paradigm. Typically, the model is equipped with a single set of learned parameters to generate responses for given input contexts. When confronting diverse conversations, its adaptability is rather limited and the model is hence prone to generate generic responses. In this work, we propose an **Ada**ptive **N**eural **D**ialogue generation model, ADAND, which manages various conversations with conversation-specific parameterization. For each conversation, the model generates parameters of the encoder-decoder by referring to the input context. In particular, we propose two adaptive parameterization mechanisms: a context-aware and a topic-aware parameterization mechanism. The context-aware parameterization directly generates the parameters by capturing local semantics of the given context. The topic-aware parameterization enables parameter sharing among conversations with similar topics by first inferring the latent topics of the given context and then generating the parameters with respect to the distributional topics. Extensive experiments conducted on a large-scale real-world conversational dataset show that our model achieves superior performance in terms of both quantitative metrics and human evaluations.

## 1 Introduction

Open-domain dialogue models, usually called chit-chat systems, draw increasing attention from both academia and industry in recent years. Building on the successful sequence-to-sequence (SEQ2SEQ) paradigm (Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2015), contemporary mainstream open-domain dialogue generation models (Shang et al., 2015; Serban et al., 2016, 2017; Shen et al., 2017; Clark and Cao, 2017; Zhao et al., 2017; Xing et al., 2017; Zhao et al., 2018), trained on a large number of context-response pairs, attempt to generate an appropriate response for the given context based on a single set of the model parameters.

Because of its great potential in understanding and modeling conversations, SEQ2SEQ has been widely applied in different kinds of conversation scenarios including technical supporting, movie discussions, and social entertainment, etc. However, when confronting conversations with diverse topics or themes, SEQ2SEQ is usually prone to make generic meaningless responses due to its over-simplified parameterization. To tackle this issue, Xing et al. (2017) proposed a topic-aware response generation model, which utilizes a single encoder-decoder, augmented with topic information obtained from a pre-trained LDA model. Though effective, the model heavily relies on the outsourcing topic information to capture the topic variations of different conversations. Another approach, per-topic/theme encoder-decoder model (Choudhary et al., 2017), uses separate encoder-decoder model for each topic or theme. This method needs the preorganized topic/theme annotations for each conversation, which are prohibitively expensive to obtain. Furthermore, building multiple separate topic/theme-specific encoder-decoders not only weakens the applicability and efficiency of the system, but also prevents parameter sharing across domains, which leads to overparameterization due to the excessive amount of model parameters.

To gather the benefits of both approaches, in this paper, we propose an adaptive neural dialogue generation model which utilizes a single encoder-decoder for diverse conversations, meanwhile, the encoder-decoder is specifically parameterized according to each conversation. In particular, we propose two adaptive parameterization mechanisms: 1) context-aware parameterization directly generates parameters of the encoder-decoder model by capturing local semantics of the given context; 2)

*Work done at Data Science Lab, JD.com.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1793–1802, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

topic-aware parameterization enables parameter sharing among conversations with similar topic distributions by first inferring the latent topics of the input context, and then generating the parameters with respect to the inferred latent topics. Equipped with both the context-aware and topic-aware parameterization mechanisms, the model is capable of generating responses for diverse conversations with a single encoder-decoder through a more flexible and efficient approach. Moreover, our model is trained in an end-to-end fashion without any costly external or labeled topic annotations.

We empirically evaluate our approach on a large scale real-world conversational dataset. Extensive experiments show that our proposed ADAND outperforms existing dialogue generation models in terms of both the automatic evaluation metrics and human judgements.

## 2 Neural Dialogue Generation Model

Conventionally, neural dialogue generation model follows the encoder-decoder paradigm. Given the context $\mathbf{x} = \{x_1, x_2, \cdots, x_{T_\mathbf{x}}\}$ and the target response $\mathbf{y} = \{y_1, y_2, \cdots, y_{T_\mathbf{y}}\}$, the model learns to maximize the following conditional probability:

$$p(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^{T_\mathbf{y}} p(y_t|\mathbf{y}_{<t}, \mathbf{x}), \tag{1}$$

where $\mathbf{y}_{<t} = y_1 \cdots y_{t-1}$.

Typically, the probability $p(y_t|\mathbf{y}_{<t}, \mathbf{x})$ is modeled as follows:

$$p(y_t|\mathbf{y}_{<t}, \mathbf{x}) = p(y_t|s_{t-1}), \tag{2}$$

where $s_{t-1}$ is the decoding hidden state up to time step $t-1$, depending on $\mathbf{y}_{<t}$ and $\mathbf{x}$. The $s_t$ can be defined as:

$$s_t = \mathcal{F}(y_t, s_{t-1}, \mathcal{G}(\mathbf{x})), \tag{3}$$

where the encoder $\mathcal{G}$ and the decoder $\mathcal{F}$ can be implemented as recurrent neural networks such as LSTMs (Hochreiter and Schmidhuber, 1997) or GRUs (Cho et al., 2014), or the transformer (Vaswani et al., 2017) (with the attention mechanism (Bahdanau et al., 2015) or self-attention mechanism).

In this work, we employ the LSTM-based encoder-decoder dialogue generation model. The

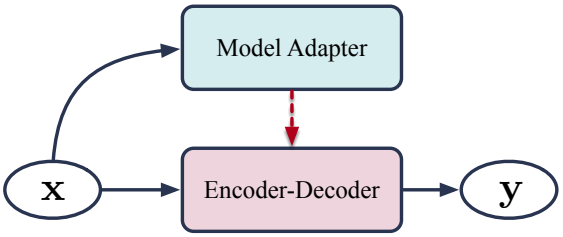


Figure 1: General model architecture. Black solid lines denote information flow, and the red dashed line indicates the adaptive parameterization operation.

LSTM unit is formulated as:

$$\begin{aligned}
i_t &= W_h^i h_{t-1} + W_x^i x_t + b^i \\
g_t &= W_h^g h_{t-1} + W_x^g x_t + b^g \\
f_t &= W_h^f h_{t-1} + W_x^f x_t + b^f \\
o_t &= W_h^o h_{t-1} + W_x^o x_t + b^o \\
c_t &= \sigma(f_t) \odot c_{t-1} + \sigma(i_t) \odot \tanh(g_t) \\
h_t &= \sigma(o_t) \odot \tanh(c_t),
\end{aligned} \tag{4}$$

where $\sigma$ is the sigmoid operator and $\odot$ stands for Hadamard product. $h_{t-1}$ is the previous hidden state and $x_t$ is the input embedding at step $t$. $W = \{W_h^i, W_h^g, W_h^f, W_h^o, W_x^i, W_x^g, W_x^f, W_x^o\}$ and $b = \{b^i, b^g, b^f, b^o\}$ are the LSTM parameters. The model parameters are tuned on the training corpus.

When testing, given the input context, it generates a response with the learned parameters. This architecture shows great success in neural dialogue generation (Shang et al., 2015; Serban et al., 2016, 2017; Shen et al., 2017; Clark and Cao, 2017; Xing et al., 2017; Choudhary et al., 2017; Zhao et al., 2018). However, with a single set of model parameters and the oversimplified model architecture, the flexibility of the model is rather limited, especially when confronting conversations with diverse topics or themes.

## 3 Adaptive Neural Dialogue Generation Model

In this work, we propose an adaptive neural dialogue generation model which utilizes a single encoder-decoder model and a set of dynamical parameters to balance the model's flexibility and efficiency. As depicted in Figure 1, we utilize the model adapter to parameterize the encoder-decoder for each conversation. It takes the dialogue context as its input, and generates parameters of the encoder-decoder model through two adaptive parameterization mechanisms; and then the resultant

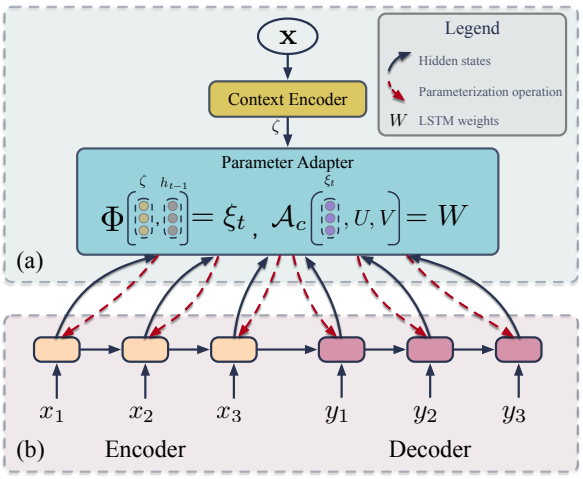


Figure 2: General architecture of the context-aware parameterization. (a) Context-aware parameter adapter. (b) One layer of the dialogue generation model. Black solid lines denote information flow, and red dashed lines indicate adaptive parameterization operations. $\mathcal{A}_c$ stands for the context-aware parameterization function (Eq.(7)).

encoder-decoder model generates the response with a specific set of model parameters.

### 3.1 Context-aware Parameterization

Context-aware parameterization parameterizes the dialogue encoder-decoder with respect to the local semantics of the given context. As shown in Figure 2, we first obtain the semantic representation of the context through a context encoder. Then, the parameter adapter dynamically adapts the parameters of the encoder-decoder at each time step. Here, we utilize a bidirectional LSTM to transform the input context into the semantic hidden representation $\zeta$.

The parameter adapter then generates the weights of LSTM units as:

$$W = M^{[1:N_r]}\zeta, \tag{5}$$

where $W \in \mathbb{R}^{N_r \times N_c}$ and $M^{[1:N_r]} \in \mathbb{R}^{N_r \times N_c \times N_\zeta}$ is a tensor. $N_r = 4N_h$ and $N_c = N_h + N_x$, in which $N_h$ is the hidden size of the LSTM and $N_x$ is the embedding size. $\zeta \in \mathbb{R}^{N_\zeta}$ is the context representation. The product $M^{[1:N_r]}\zeta$ results in a weight matrix $W$, where each row is computed by one slice $j = 1, 2, \cdots, N_r$ of the tensor: $W_j = (M^j\zeta)^{\mathrm{T}}$.

Although such parameterization seems to be straightforward, due to the quadratic size of $M^{[1:N_r]}$, the parameter size of such parameter adapter is $N_\zeta$ times larger than the basic encoder-decoder model. Therefore, overfitting and expensive computational cost make it infeasible (Bertinetto et al., 2016; Ha et al., 2017). Following Flennerhag et al. (2018), we reduce the parameter space by factorizing the weights as:

$$\begin{aligned} W &= \mathcal{A}_c(\xi_t, U, V) \\ \xi_t &= \Phi(\zeta, h_{t-1}), \end{aligned} \tag{6}$$

where $\Phi$ is implemented as a LSTM unit, $h_{t-1}$ is the previous encoder or decoder hidden state, and $\xi_t \in \mathbb{R}^{N_\zeta}$ is the context representation at time step $t$. $\mathcal{A}_c$ denotes the context-aware parameterization function, defined as:

$$\mathcal{A}_c(\xi_t, U, V) = U\xi_t V^{\mathrm{T}}, \tag{7}$$

where $U \in \mathbb{R}^{N_r \times N_\zeta}$ and $V \in \mathbb{R}^{N_c \times N_\zeta}$ are learnable weights.

The context-aware parameterization function $\mathcal{A}_c$ is reminiscent of the Singular Value Decomposition (SVD). Here, $\mathcal{A}_c$ composes a projection by adapting the dialogue context to the weight matrices and does not perform matrix decomposition actually. The number of parameters in this formulation is $L = N_r \times N_\zeta + N_c \times N_\zeta$ and the total parameter number of the model is linear with $L$, so that the total number of model parameters does not explode.

### 3.2 Topic-aware Parameterization

Context-aware parameterization adapts the encoder-decoder parameters regarding each input context. As a result, the adapted encoder-decoder is sensitive to the input context. To enable the parameter sharing among similar topics, we further propose a topic-aware parameterization mechanism. As shown in the Figure 3, the topic inferrer first distills the topic distribution $\theta$ from the context (Figure 3 (a)), and then the parameters of the encoder-decoder model are constructed upon $\theta$ (Figure 3 (b)).

#### 3.2.1 Latent Topic Inference

We introduce a variational topic inferrer to infer the topic distribution $\theta$ of the conversation. Drawing inspirations from neural topic model (Miao et al., 2017), as illustrated in Figure 3 (a), the generation process of the variational topic inferrer is formulated as follows:

(i) A latent variable $\nu$ is inferred to convey the underlying semantics of the given context.

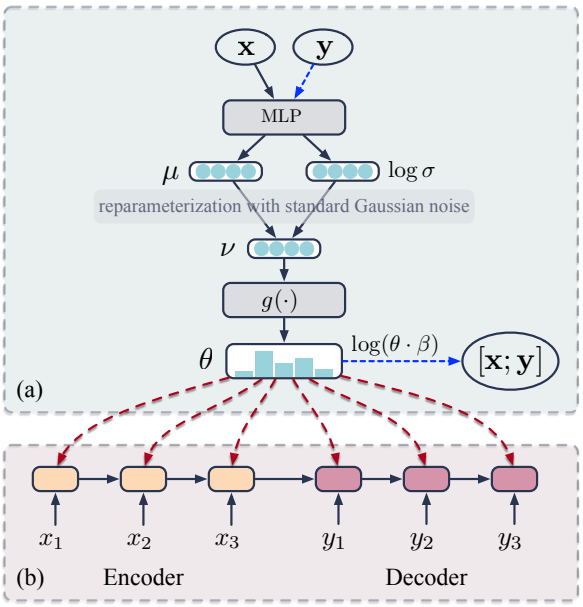


Figure 3: Illustration of the topic-aware parameterization. (a) Topic inferrer. (b) One layer of the dialogue generation model. Black solid lines denote information flow, blue dashed lines only appear in training, and red dashed lines indicate adaptive parameterization operations.

(ii) The topic distribution $\theta$ is constructed from the latent variable $\nu$ through a softmax function.

(iii) The dialogue $\mathbf{d}$, composing of a context-response pair $(\mathbf{x}, \mathbf{y})$, is drawn from the topic distribution over words $p(w_i|\beta_{z_i})$, where $z_i$ is a topic assignment sampled from a multinomial distribution parameterized by the topic distribution $\theta$, and $\beta_{z_i}$ is the topic-word distribution of topic assignment $z_i$.

Given a context $\mathbf{x}$, the latent variable $\nu$ is sampled from $P(\nu|\mathbf{x}) = \mathcal{N}(\mu_{prior}, \sigma^2_{prior})$, and $\mathcal{N}(\mu_{prior}, \sigma^2_{prior})$ is the multivariate normal distribution with mean $\mu_{prior}$ and diagonal covariance matrix $\sigma^2_{prior}$. In practice, $\nu$ is reparameterized as: $\nu = \mu_{prior} + \epsilon \cdot \sigma_{prior}$ and $\epsilon$ is a standard Gaussian noise. The Gaussian parameters $\mu_{prior}$ and $\sigma^2_{prior}$ are the outputs of multilayer perceptrons (MLP) given the bag-of-words (BoW) representation of the context as input. To reduce the encoding noise of stop words, here we choose the BoW instead of LSTM for context representations, following Miao et al. (2017). To implement neural variational inference, we utilize an inference network $Q(\nu|\mathbf{d}) = \mathcal{N}(\mu_{posterior}, \sigma^2_{posterior})$ to approximate the intractable true posterior $p(\nu|\mathbf{d})$, where $\mu_{posterior}$ and $\sigma^2_{posterior}$ are computed in the same way as the prior, taking the bag-of-words representation of dialogue $\mathbf{d}$ as input. The dialogue $\mathbf{d}$ consists of the context-response pair $(\mathbf{x}, \mathbf{y})$.

The topic distribution $\theta$ is constructed from the latent variable $\nu$ through a softmax function:

$$\theta = g(\nu) = \text{softmax}(\nu \cdot \mathbf{W}_\nu), \tag{8}$$

where $\mathbf{W}_\nu$ is a linear transformation and the bias terms are left out for brevity.

Then, the dialogue $\mathbf{d}$ is generated based on the topic distribution $\theta$. Given $\theta$, the marginal likelihood of a dialogue $\mathbf{d}$ is formulated as:

$$p(\mathbf{d}) = \int_\theta p(\theta) \prod_{i=1}^{|\mathbf{d}|} \sum_{z_i} p(w_i|\beta_{z_i}) p(z_i|\theta) d\theta. \tag{9}$$

In addition, the topic assignment $z_i$ can be integrated out and the log-likelihood of a word $w_i$ in dialogue $\mathbf{d}$ can be factorized as:

$$\begin{aligned} \log p(w_i|\beta, \theta) &= \log \sum_{z_i} [p(w_i|\beta_{z_i}) p(z_i|\theta)] \\ &= \log(\theta \cdot \beta^{\mathrm{T}}). \end{aligned} \tag{10}$$

The topic-word distribution $\beta_k$ is defined by:

$$\beta_k = \text{softmax}(F \cdot \Lambda_k^{\mathrm{T}}), \tag{11}$$

where $F \in \mathbb{R}^{C \times H}$ is the topical word embedding matrix and $\Lambda \in \mathbb{R}^{K \times H}$ is the topic embedding matrix, $K$ is the number of topics, $C$ is the number of topical words, $H$ is the embedding size and $\beta = \{\beta_1, \beta_2, \cdots, \beta_K\}$.

### 3.2.2 Parameterization with Latent Topics

We parameterize the encoder-decoder with the inferred topic distribution $\theta$. In context-aware parameterization, the parameters of the encoder-decoder are adapted dynamically at each time step, whereas in topic-aware parameterization, as illustrated in Figure 3 (b), we generate only one set of parameters for each conversation.

Similar to the context-aware parameterization function in Eq.(7), given the topic distribution $\theta$, the topic-aware parameterization function $\mathcal{A}_\kappa$ constructs the LSTM weight $W$ as follows:

$$\begin{aligned} W &= \mathcal{A}_\kappa(\theta, U, V) \\ \mathcal{A}_\kappa(\theta, U, V) &= U\theta V^{\mathrm{T}}, \end{aligned} \tag{12}$$

where $U \in \mathbb{R}^{N_r \times K}$ and $V \in \mathbb{R}^{N_c \times K}$ are learnable parameters. $K$ is the number of latent topics.

### 3.3 Parameterization with Both Context and Topics

Intuitively, context-aware parameterization is more adept at capturing local semantics of the input context while topic-aware parameterization enables parameter sharing between conversations with similar topic distributions. To benefit the model parameterization with both the local and global information, we further adapt parameters of the encoder-decoder by utilizing both the context representations $\xi_t$ and the topic distribution $\theta$. In particular, the LSTM weight $W$ at time step $t$ is adapted as follows:

$$\begin{aligned} W =& \Psi_t \mathcal{A}_c(\xi_t, U_c, V_c) + \\ & (1-\Psi_t)\mathcal{A}_\kappa(\theta, U_\kappa, V_\kappa) \\ \Psi_t =& \sigma(\xi_t, \theta), \end{aligned} \tag{13}$$

where $\Psi_t$ is the gating function deciding whether the parameterization relies more on the context or the topics. $U_c, V_c, U_\kappa$ and $V_\kappa$ are learnable weights. $\mathcal{A}_c$ and $\mathcal{A}_\kappa$ denote the context-aware and topic-aware parameterization function respectively. $\sigma$ is the sigmoid function.

### 3.4 Learning

To enable the joint optimization of latent topic inference, adaptive model parameterization, and response generation in ADAND, given the definitions in Eq.(9), similar to Kingma and Welling (2014) and Miao et al. (2017), we derive a variational lower bound for the generation likelihood:

$$\begin{aligned} \mathcal{J} =& \mathbb{E}_{Q(\theta|\mathbf{d})}[\sum_{i=1}^{|\mathbf{d}|}[\log \sum_{z_i}[p(w_i|\beta_{z_i})p(z_i|\theta)]] + \log p(\mathbf{y}|\mathbf{x})] \\ & - D_{KL}(Q(\theta|\mathbf{d})||P(\theta|\mathbf{x})) \\ =& \mathbb{E}_{Q(\theta|\mathbf{d})}[\sum_{i=1}^{|\mathbf{d}|}\log p(w_i|\beta,\theta) + \sum_{t=1}^{T_\mathbf{y}}\log p(y_t|s_{t-1})] \\ & - D_{KL}(Q(\theta|\mathbf{d})||P(\theta|\mathbf{x})) \\ =& \mathbb{E}_{Q(\nu|\mathbf{d})}[\sum_{i=1}^{|\mathbf{d}|}\log p(w_i|\beta,\nu) + \sum_{t=1}^{T_\mathbf{y}}\log p(y_t|s_{t-1})] \\ & - D_{KL}(Q(\nu|\mathbf{d})||P(\nu|\mathbf{x})) \\ \approx & \sum_{i=1}^{|\mathbf{d}|}\log p(w_i|\beta,\nu) + \sum_{t=1}^{T_\mathbf{y}}\log p(y_t|s_{t-1}) \\ & - D_{KL}(Q(\nu|\mathbf{d})||P(\nu|\mathbf{x})), \end{aligned} \tag{14}$$

where $\mathbf{y} = \{y_1, y_2, \cdots, y_{T_y}\}$, $P(\nu|\mathbf{x})$ is the prior estimation of the latent variable $\nu$ which approximates the posterior $Q(\nu|\mathbf{d})$. The prior $P(\theta|\mathbf{x}) = P(g(\nu)|\mathbf{x}) = P(\nu|\mathbf{x})$, and the posterior $Q(\theta|\mathbf{d}) = Q(g(\nu)|\mathbf{d}) = Q(\nu|\mathbf{d})$. The first term is the dialogue generation objective in the latent topic inferrer, the second term is the response generation objective, and the third term is the KL divergence between two Gaussian distributions. All the parameters are learned by optimizing Eq.(14) and updated with back-propagation.

The following previously proposed strategies (Bowman et al., 2016; Zhao et al., 2017) are adopted in training to alleviate the vanishing latent variable problem: (1) KL annealing: the weight of the KL divergence term is gradually increasing from 0 to 1 during training; (2) Bag-of-words loss: the bag-of-words loss requires the latent variable $\nu$, together with the dialogue context, to reconstruct the response bag-of-words representation $y_b$.

## 4 Experiments

### 4.1 Dataset and Competitor Baselines

To ascertain the effectiveness of the proposed model, we construct an open-domain conversation corpus covering a broad range of resources including a movie discussions dataset collected from Reddit (Dodge et al., 2015), an Ubuntu technical corpus (Lowe et al., 2015), and a chit-chat dataset (Zhang et al., 2018). 87,468 context-response pairs were sampled for training, 4,460 for validation and 4,468 for testing.

The code and corpus are available at `http://github.com/hengyicai/AdaND`.

The following state-of-the-art models are adopted as our comparison systems.

**SEQ2SEQ** The attention-based sequence-to-sequence model (Bahdanau et al., 2015), which is a representative baseline.

**CVAE** A latent variable conversation model in which it incorporates a latent variable at the sentence-level to inject stochasticity and diversity (Clark and Cao, 2017; Zhao et al., 2017).

**LAED** A recurrent encoder-decoder conversation model using discrete latent actions for interpretable neural dialogue generation (Zhao et al., 2018).

**TA-SEQ2SEQ** TA-SEQ2SEQ incorporates the outsourcing topic information into the response generation, where the topics are learned from a separate LDA model to enrich the context (Xing et al., 2017).

**DOM-SEQ2SEQ** A domain-aware conversation model consisting of multiple domain-targeted

| Models | Relevance (%) | | | | Informativeness (%) | | |
|---|---|---|---|---|---|---|---|
| | BLEU | Average | Greedy | Extrema | Distinct-1 | Distinct-2 | Distinct-3 |
| **SEQ2SEQ** | 0.845 | 69.60 | 64.94 | 45.29 | 0.2822 | 0.5922 | 0.7873 |
| **CVAE** | 1.546 | 71.23 | 66.67 | 47.14 | 0.5465 | 1.716 | 2.731 |
| **LAED** | 0.7545 | 69.91 | 63.55 | 43.12 | 0.3890 | 0.9165 | 1.243 |
| **TA-SEQ2SEQ** | 1.465 | 72.47 | 65.9 | 45.19 | 0.3593 | 0.7994 | 1.016 |
| **DOM-SEQ2SEQ** | 1.189 | 74.42 | 66.6 | 48.47 | 0.4977 | 1.294 | 1.814 |
| **ADAND** (***with context para.***) | 1.94 | 74.03 | 66.76 | 49.23 | 0.6493 | 1.889 | 2.745 |
| **ADAND** (***with topic para.***) | **2.051** | 74.17 | 66.65 | 49.04 | 0.5919 | 1.699 | 2.438 |
| **ADAND** (***with both***) | 1.90 | **75.59** | **67.25** | **51.17** | **0.7092** | **2.10** | **3.108** |

Table 1: Quantitative evaluation results (%).

SEQ2SEQ models for response generation (Choudhary et al., 2017).

### 4.2 Evaluation

Following the evaluation procedure in previous work (Li et al., 2016; Xing et al., 2017; Chen et al., 2018), experimental results of all models are reported in terms of the relevance and informativeness. To evaluate the semantic relevance between the generated response and the ground-truth response, we adopted the BLEU metric (Papineni et al., 2002) and three embedding-based similarity metrics proposed in Liu et al. (2016): Embedding Average (**Average**), Embedding Extrema (**Extrema**) and Embedding Greedy (**Greedy**). To measure informativeness and diversity of the response, we exploited the Distinct-1, Distinct-2 and Distinct-3 metrics. A higher ratio of distinct n-grams implies more informative and diverse responses.

### 4.3 Implementation and Reproducibility

We implemented our model with ParlAI (Miller et al., 2017). The sequence lengths are truncated at 50. We used Adam (Kingma and Ba, 2014) with an initial learning rate of 0.001 to optimize the model. For all the experiments, we employed a 2-layer bidirectional LSTM as the encoder and a unidirectional one as the decoder. The hidden size and the word embedding dimension are both set to 300. The latent variable size is set to 64. The topic number $K$ in our model is set to 5 and the most frequent 3,159 words are taken as the topical words vocabulary by stemming, filtering stop-words from the training set. The batch size is set to 128 for all models. We trained a Twitter LDA model to obtain the topical words for TA-SEQ2SEQ and set its model-specific parameters following the original paper (Xing et al., 2017). For regularization and preventing over-fitting, a dropout of 0.1 is applied and the weight decay is set to $3 \times 10^{-5}$. We used the pretrained word embeddings (Pennington et al., 2014) of 300 dimensions, and the vocabulary size is set to 20,000. All models are trained with early stopping, i.e., if the loss does not decrease after 10 validations. The loss is computed on the validation set at every 0.5 epochs and we save the parameters for the top model on the validation set. We finally report evaluation scores on the test set from the saved model.

### 4.4 Overall Performance

Table 1 lists the performance of our system and the comparison systems. CVAE and LAED inject SEQ2SEQ with stochastic latent variable, resulting in more informative responses and better performance on Distinct-{1, 2, 3}. TA-SEQ2SEQ incorporates SEQ2SEQ with the outsourcing topic information from LDA. It is not surprising that it performs much better on the response relevance (BLEU, Average, Greedy, Extrema), while its improvements on the informativeness are limited. DOM-SEQ2SEQ builds multiple domain-specific encoder-decoders. It gains improvements on both the relevance metrics and informativeness metrics.

In general, with both the context-aware and topic-aware parameterization, our model outperforms all the competitive baselines in terms of the response relevance and informativeness.

### 4.5 Context-aware *vs* Topic-aware Parameterization

Context-aware parameterization captures local semantics of the given context, while topic-aware parameterization enables parameters sharing among conversations with similar topics. As shown in Ta-

| Opponent | Win | Loss | Tie | Kappa |
|---|---|---|---|---|
| **ADAND vs. SEQ2SEQ** | 62.5% | 2.52% | 34.98% | 46.46 |
| **ADAND vs. CVAE** | 36.98% | 7.56% | 55.46% | 51.59 |
| **ADAND vs. LAED** | 44.54% | 6.72% | 48.74% | 55.57 |
| **ADAND vs. TA-SEQ2SEQ** | 54.62% | 5.88% | 39.5% | 54.91 |
| **ADAND vs. DOM-SEQ2SEQ** | 32.78% | 10.08% | 57.14% | 55.53 |

Table 2: The results of human evaluation.

| Models | Speed (cases/ms) |
|---|---|
| **SEQ2SEQ** | 0.740 |
| **CVAE** | 0.541 |
| **LAED** | 0.435 |
| **TA-SEQ2SEQ** | 0.172 |
| **DOM-SEQ2SEQ** | 0.106 |
| **ADAND** | 0.407 |

Table 3: Speed test.

ble 1, both parameterization mechanisms perform much better than the original SEQ2SEQ model, while context-aware parameterization is slightly better in terms of informativeness. When jointly utilizing both the context-aware and topic-aware parameterization mechanisms, we observe the best performance, indicating that these two mechanisms are both beneficial and complementary.

### 4.6 Human Evaluation

We conducted human evaluations on the test set to further validate the effectiveness of the model. We randomly selected 500 samples from the test set. Three well-educated students were invited to conduct the evaluation. For each case, we provided annotators with triplets (sample, response$_1$, response$_2$) whereby one response is generated by ADAND, and the other is generated by a competitor model. The annotators, who have no knowledge about which system the response is from, are then required to independently rate among win (response$_1$ is better), loss (response$_2$ is better) and tie (they are equally good or bad), considering four factors: context relevance, logical consistency, fluency and informativeness. Note that if annotators rate different options, this triplet will be counted as "tie". Table 2 reveals the results of subjective evaluation. The kappa scores indicate that the annotators came to a fair agreement in the judgment.

As expected, ADAND outperforms the other baselines and enjoys a large margin over the existing models. The relative performance of the competitors is consistent with the quantitative evaluation results, confirming the superior performance of our proposed method.

| Movie | Politics | Ubuntu | Food | Network |
|---|---|---|---|---|
| imax | trump | install | food | router |
| movie | people | grub | vegetarian | wireless |
| youtube | hillary | apt | seafood | ip |
| scene | vote | kernel | restaurants | address |
| marvel | clinton | nvidia | cotto | phone |
| hulk | election | cd | gourmet | network |
| avengers | debate | sudo | serves | card |
| nolan | donald | ssh | starbucks | eth0 |
| comics | support | boot | breakfast | dhcp |
| batman | working | ubuntu | pizzeria | wifi |

Table 4: Topics by the words ($\beta$ in Eq.(10)) with top-10 highest probability discovered by the latent topic inferrer.

### 4.7 Speed Test

We conducted speed test to verify the efficiency of the ADAND model empirically in Table 3. Augmented with auxiliary components, all the extension models exhibit higher time cost than the original SEQ2SEQ model. We observe that the decoding speeds of CVAE and LAED are relatively comparable with our model. However, when comparing with TA-SEQ2SEQ and DOM-SEQ2SEQ that also elaborately and explicitly model conversations with diverse topics or themes, ADAND shows a clear superiority in decoding speed. For TA-SEQ2SEQ, it relies on an outside LDA model to obtain the topic information. The joint attention and copying mechanism also reduce its efficiency. For DOM-SEQ2SEQ, it is not surprising that the time complexity of multiple topic/theme-specific encoder-decoders is much higher than all-other comparison models. ADAND utilizes a single encoder-decoder and is parameterized dynamically regarding the input context, which ensures its flexibility and efficiency.

### 4.8 Analysis & Case Study

To get some insights of how topic-aware parameterization performs, we present the topics by the words ($\beta$ in Eq.(10)) with top-10 highest probabilities in Table 4. The discernible clusters of the topical words ($F$ in Eq.(11)) are illustrated in Figure 4. These evidences demonstrate that the topic inferrer in topic-aware parameterization effectively distills the latent topic distribution of each conversation, which enables the parameter sharing among conversations with similar topics.

We also investigate the orthogonality of the learned $U$ and $V$ matrices in Eq.(7) and Eq.(12). We trained our model multiple times with differ-

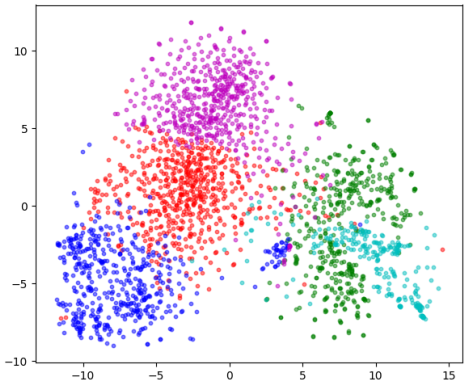

Figure 4: t-SNE (van der Maaten and Hinton, 2008) projection of topical word embeddings ($F$ in Eq.(11)) in the latent topic inferrer. Words with similar topics are in the same color.

ent parameter initialization methods (drawn values from normal distribution or uniform distribution). We observe that $UU^{\mathrm{T}}$ and $VV^{\mathrm{T}}$ approximate identity matrices. We conjecture that such SVD-alike parameterization implicitly enforces orthogonality during training.

We list several examples generated by different models in Table 5. The inferred latent topic distributions are also presented in the table. It can be observed that responses generated by the original SEQ2SEQ model are more generic. Latent variable conversation models (CVAE and LAED) generate more diverse but sometimes irrelevant responses, TA-SEQ2SEQ tends to produce short responses while DOM-SEQ2SEQ does not perform obviously better than TA-SEQ2SEQ. The responses generated by ADAND are not only relevant but also informative.

## 5 Related Work

Our work is closely related to the research of dialogue generation in diverse conversations. Previous work relies on external pre-organized topic information (Xing et al., 2017; Wang et al., 2017) or predicted keywords (Yao et al., 2017; Wang et al., 2018) to boost the response informativeness and coherence. Choudhary et al. (2017) further leveraged the topic/theme annotations to build multiple separate encoder-decoder models for topic/theme-aware response generation. In contrast, we do not exploit any outsourcing or labeled topic information. The proposed model directly infers the latent topics of each conversation and is trained in an end-to-end manner. Another difference is that we maintain a single encoder-decoder for various conversations whereas the model is dynamically and specially parameterized.

The second line of related work is parameterization in NLP. Ha et al. (2017) proposed to train a small network to generate the parameters for another larger network. Such adaptive parameteriza-

| | |
|---|---|
| 1 | ***Context:*** *Hi, can you help me find a good movie to watch?*<br>***Reference:*** *Hi, I would like to help out.*<br>***SEQ2SEQ:*** *I'm sorry, I'm sorry, I'm sorry.*<br>***CVAE:*** *I'm looking for a good comedy movie.*<br>***LAED:*** *I'm sorry, I'm not sure.*<br>***TA-SEQ2SEQ:*** *I am looking for a good movie.*<br>***DOM-SEQ2SEQ:*** *I sent the info on your screen, drive carefully!*<br>**ADAND:** *What kind of movies do you like?*<br>***Inferred topic distribution:***  |
| 2 | ***Context:*** *I love to sit on the couch and play video games myself.*<br>***Reference:*** *My brother does too, especially since our parents moved to florida recently.*<br>***SEQ2SEQ:*** *I'm sorry, I'm not a good.*<br>***CVAE:*** *I'm not sure that is my favorite.*<br>***LAED:*** *I'm not a good. I'm a good.*<br>***TA-SEQ2SEQ:*** *That's cool.*<br>***DOM-SEQ2SEQ:*** *I'm a good. I'm a __unk__*<br>**ADAND:** *That's cool. What do you do for a living?*<br>***Inferred topic distribution:***  |
| 3 | ***Context:*** *We go back to celebration in Florida often. It is our home town.*<br>***Reference:*** *Oh how nice! I love taking my dog to the beach!*<br>***SEQ2SEQ:*** *I'm sorry, I'm not a good.*<br>***CVAE:*** *I'm a little more than a lot of time.*<br>***LAED:*** *I'm sorry, I'm sorry, but there are no results for any other restaurants in the centre. Would you like to try another area?*<br>***TA-SEQ2SEQ:*** *:)*<br>***DOM-SEQ2SEQ:*** *I have a 4 star hotel near a park, museum, and theatre, 8.86/10 guest rating, free parking, breakfast, and wifi, from September 2nd to 13th, business class flights, __unk__ 86.*<br>**ADAND:** *That's cool. Do you have any hobbies?*<br>***Inferred topic distribution:*** 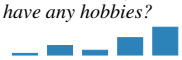 |
| 4 | ***Context:*** *Oh, that's all right. No need to reserve my ticket. But I do need the travel time, please.*<br>***Reference:*** *Your travel time is 60 minutes. Is there anything else that I can do for you?*<br>***SEQ2SEQ:*** *I have booked you for the reference number is __unk__*<br>***CVAE:*** *The travel time is 105 minutes.*<br>***LAED:*** *I' m sorry, I' m sorry, I' m sorry, but there are no results for that time. Would you like to try another day or time?*<br>***TA-SEQ2SEQ:*** *The train id is __unk__*<br>***DOM-SEQ2SEQ:*** *You're welcome. Have a great day!*<br>**ADAND:** *The travel time is 163 minutes. Is there anything else i can help you with?*<br>***Inferred topic distribution:***  |
| 5 | ***Context:*** *How is gonna be the weather forecast in San Mateo for the next weekend?*<br>***Reference:*** *In San Mateo it will be stormy on Saturday, and hail on Sunday.*<br>***SEQ2SEQ:*** *The address is a park, and the address is a park.*<br>***CVAE:*** *It will be a low of 60F and a high of 70F.*<br>***LAED:*** *what city would you like to know the weather for?*<br>***TA-SEQ2SEQ:*** *Do you have a budget?*<br>***DOM-SEQ2SEQ:*** *It will not be cloudy this week.*<br>**ADAND:** *It will not be cloudy in San Mateo next week.*<br>***Inferred topic distribution:***  |

Table 5: Test samples of our model (ADAND) and the baselines. The latent topic distributions inferred by ADAND are also presented. The reference is the ground-truth response in the dataset.

tion has been shown to be successful in many NLP tasks, including language modeling (Suarez, 2017; Flennerhag et al., 2018), sequence generation (Ha and Eck, 2018; Peng et al., 2019), and neural machine translation (Platanios et al., 2018). In our work, we parameterize the encoder-decoder with respect to both the context and the latent topics.

Regarding latent variable conversation models, prior researches strive to learn meaningful latent variables for dialogue systems, and reveal that latent variables befit the neural dialogue models with more diverse response generations (Serban et al., 2017; Zhao et al., 2017; Clark and Cao, 2017; Chen et al., 2018) and interpretable dialogue actions (Zhao et al., 2018). In our model, instead of directly injecting the latent variable into dialogue models, we distill the latent topics through neural variational inference, offering a more interpretable latent variable. Moreover, we parameterize the encoder-decoder with the inferred latent topics, which allows parameter sharing among conversations with similar topics.

## 6 Conclusion

This paper presents an adaptive neural dialogue generation model—ADAND, which allows the dynamical parameterization of the model to each conversation and enables the generation of appropriate responses in diverse conversations. Specially, we propose two adaptive parameterization approaches: context-aware parameterization which captures local semantics of the input context; and topic-aware parameterization which enables parameter sharing by first inferring the latent topics of the given context and then generating the parameters with the inferred latent topics. The proposed approaches are assessed on a large-scale conversational dataset and the results show that our model achieves superior performance and higher efficiency. It should be noted that our approach is not isolated to only LSTMs. We would like to explore the effectiveness of the approach regarding other structures in future work.

## Acknowledgments

We would like to thank all the reviewers for their insightful and valuable comments and suggestions. Hongshen Chen and Cheng Zhang are the corresponding authors.

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Luca Bertinetto, João F. Henriques, Jack Valmadre, Philip H. S. Torr, and Andrea Vedaldi. 2016. Learning feed-forward one-shot learners. In *NIPS*, pages 523–531.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *SIGNLL*, pages 10–21.

Hongshen Chen, Zhaochun Ren, Jiliang Tang, Yihong Eric Zhao, and Dawei Yin. 2018. Hierarchical variational memory network for dialogue generation. In *WWW*, pages 1653–1662.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP*, pages 1724–1734.

Sajal Choudhary, Prerna Srivastava, Lyle H. Ungar, and João Sedoc. 2017. Domain aware neural dialog system. *CoRR*, abs/1708.00897.

Stephen Clark and Kris Cao. 2017. Latent variable dialogue models and their diversity. In *EACL*, pages 182–187.

Jesse Dodge, Andreea Gane, Xiang Zhang, Antoine Bordes, Sumit Chopra, Alexander H. Miller, Arthur Szlam, and Jason Weston. 2015. Evaluating prerequisite qualities for learning end-to-end dialog systems. *CoRR*, abs/1511.06931.

Sebastian Flennerhag, Hujun Yin, John A. Keane, and Mark Elliot. 2018. Breaking the activation function bottleneck through adaptive parameterization. In *NeurIPS*, pages 7750–7761.

David Ha, Andrew M. Dai, and Quoc V. Le. 2017. Hypernetworks. In *ICLR*.

David Ha and Douglas Eck. 2018. A neural representation of sketch drawings. In *ICLR*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *ICLR*.

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *ICLR*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *NAACL-HLT*, pages 110–119.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *EMNLP*, pages 2122–2132.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *SIGDIAL*, pages 285–294.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605.

Yishu Miao, Edward Grefenstette, and Phil Blunsom. 2017. Discovering discrete latent topics with neural variational inference. In *ICML*, volume 70, pages 2410–2419.

A. H. Miller, W. Feng, A. Fisch, J. Lu, D. Batra, A. Bordes, D. Parikh, and J. Weston. 2017. Parlai: A dialog research software platform. *arXiv preprint arXiv:1705.06476*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318.

Hao Peng, Ankur P. Parikh, Manaal Faruqui, Bhuwan Dhingra, and Dipanjan Das. 2019. Text generation with exemplar-based adaptive decoding. In *NAACL-HLT*, pages 2555–2565.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Emmanouil Antonios Platanios, Mrinmaya Sachan, Graham Neubig, and Tom M. Mitchell. 2018. Contextual parameter generation for universal neural machine translation. In *EMNLP*, pages 425–435.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, pages 3776–3784.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*, pages 3295–3301.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *ACL*, pages 1577–1586.

Xiaoyu Shen, Hui Su, Yanran Li, Wenjie Li, Shuzi Niu, Yang Zhao, Akiko Aizawa, and Guoping Long. 2017. A conditional variational framework for dialog generation. In *ACL*, pages 504–509.

Joseph Suarez. 2017. Character-level language modeling with recurrent highway hypernetworks. In *NIPS*, pages 3269–3278.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*, pages 5998–6008.

Di Wang, Nebojsa Jojic, Chris Brockett, and Eric Nyberg. 2017. Steering output style and topic in neural response generation. In *EMNLP*, pages 2140–2150.

Wenjie Wang, Minlie Huang, Xin-Shun Xu, Fumin Shen, and Liqiang Nie. 2018. Chat more: Deepening and widening the chatting topic via A deep model. In *SIGIR*, pages 255–264.

Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2017. Topic aware neural response generation. In *AAAI*, pages 3351–3357.

Lili Yao, Yaoyuan Zhang, Yansong Feng, Dongyan Zhao, and Rui Yan. 2017. Towards implicit content-introducing for generative short-text conversation systems. In *EMNLP*, pages 2190–2199.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? In *ACL*, pages 2204–2213.

Tiancheng Zhao, Kyusong Lee, and Maxine Eskénazi. 2018. Unsupervised discrete sentence representation learning for interpretable neural dialog generation. In *ACL*, pages 1098–1107.

Tiancheng Zhao, Ran Zhao, and Maxine Eskénazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *ACL*, pages 654–664.

# Towards Knowledge-Based Recommender Dialog System

**Qibin Chen**[1], **Junyang Lin**[2], **Yichang Zhang**[2], **Ming Ding**[1], **Yukuo Cen**[1], **Hongxia Yang**[2], **Jie Tang**[1]
[1]Department of Computer Science and Technology, Tsinghua University
[2]DAMO Academy, Alibaba Group
{cqb19,dm18,cyk18}@mails.tsinghua.edu.cn
{junyang.ljy,yichang.zyc,yang.yhx}@alibaba-inc.com
jietang@tsinghua.edu.cn

## Abstract

In this paper, we propose a novel end-to-end framework called **KBRD**, which stands for **K**nowledge-**B**ased **R**ecommender **D**ialog System. It integrates the recommender system and the dialog generation system. The dialog system can enhance the performance of the recommendation system by introducing knowledge-grounded information about users' preferences, and the recommender system can improve that of the dialog generation system by providing recommendation-aware vocabulary bias. Experimental results demonstrate that our proposed model has significant advantages over the baselines in both the evaluation of dialog generation and recommendation. A series of analyses show that the two systems can bring mutual benefits to each other, and the introduced knowledge contributes to both their performances.[1]

## 1 Introduction

Dialog in e-commerce has great commercial potential. In conventional recommender systems, personalized recommendation is highly based on the previous actions of users, including searching, clicking and purchasing. These actions can be regarded as users' feedbacks that reflect users' interest. However, due to its implicitness, such feedback can only reflect a part of users' interest, causing inaccuracy in recommendation. Another information source about user preferences is the dialog between users and services. In such dialog, users often provide more information about their preferences. They often ask for tips or recommendation in the dialog. In this process, services can guide them to speak out their interests in order to solve users' problems and meet their requirements. Compared with the implicit feedback, the feedback from the dialog is more explicit and more related to users' preferences. Therefore, a recommender dialog system possesses high commercial potential.

| | |
|---|---|
| **USER**: | Hello! |
| **RECOMMENDER**: | What kind of movies do you like? |
| **USER**: | I am looking for a movie recommendation. When I was younger I really enjoyed the *A Nightmare on Elm Street (1984)*. |
| **BASELINE**: | Have you seen *It (2017)*? |
| **OURS**: | I like horror movies too! Have your seen *Halloween (1978)* ? |
| **HUMAN**: | Oh, you like scary movies? I recently watched *Happy Death Day (2017)*. |

Table 1: **An example of the outputs of recommender dialog systems.** The model recommends items (italic) while maintaining the dialog with the user. Compared with the baseline, our dialog system gives a more diverse and consistent response.

A recommender dialog system can be regarded as a combination of a recommender system and a dialog system. A dialog system should respond to users' utterances with informative natural language expressions, and a recommender system should provide high-quality recommendation based on the content of users' utterances. We demonstrate an example in Table 1. In brief, a recommender dialog system should perform well in both tasks.

An ideal recommender dialog system is an end-to-end framework that can effectively integrate the two systems so that they can bring mutual benefits to one another. In this setting, information from the recommender system can provide vital information to maintain multi-turn dialog, while information from the dialog system that contains implication of users' preferences can enhance the qual-

[1]Code will be available at https://github.com/qibinc/KBRD.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1803–1813,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

ity of recommendation. Besides, the incorporation of external knowledge can strengthen the connections between systems and enhance their performances. Therefore, driven by the motivations, we propose a novel end-to-end framework that integrates the two systems. We name it **KBRD**, standing for **K**nowledge-**B**ased **R**ecommender **D**ialog System.

Specifically, the dialog generation system provides contextual information about items to the recommender system. For instance, for a movie recommendation system, contextual information can be director, actor/actress and genre. Thus, even with no item mentioned in the dialog, the recommender system can still perform high-quality recommendation based on the contextual information. In return, the recommender system provides recommendation information to promote the dialog, such as recommendation-aware vocabulary bias. Furthermore, we incorporate external knowledge into our framework. The knowledge graph helps bridge the gap between systems and enhances both their performances.

We conduct a series of experiments that demonstrate the effects of our framework in both the evaluation of recommendation and dialog generation. Moreover, the analyses show that dialog information effectively tackles the cold-start problem in recommendation, and the recommendation-aware vocabulary bias from the recommender system improves the quality of generated dialogs. Also, the biased words can be parts of reasons to explain the system's decisions for recommendation.

## 2 Preliminary

Before we introduce our proposed framework, we provide an illustration of the basic framework of the recommender dialog system to show how the recommendation system and the dialog system are organized for end-to-end training.

### 2.1 Recommender System

Provided with a user's information, a recommender system is aimed at retrieving a subset of items that meet the user's interest from all the items. In a cold-start setting, the recommender system initially has no knowledge about the user. With the progress of the dialog, the recommender system accumulates user's information and builds a user profile. Thus, it can provide reasonable recommendation based on the user preferences reflected in the conversation.

To implement an effective recommender system in this task, it is available to build a recommender system based on conventional collaborative filtering algorithms (Sarwar et al., 2001) or based on neural networks (He et al., 2017). For example, Li et al. (2018) applies a user-based autoencoder (Sedhain et al., 2015) to recommend new items based on previously mentioned items in the dialog.

### 2.2 Dialog System

The dialog system in the basic framework is in charge of generating multi-turn dialog with a natural language generation model. The pioneering work (Li et al., 2018) on conversational recommendation task adopted Hierarchical Recurrent Encoder Decoder (HRED) (Sordoni et al., 2015b,a; Serban et al., 2016) for this part. The HRED is an encoder-decoder framework for sequence-to-sequence learning (Sutskever et al., 2014). In the framework, an encoder receives the dialog history as input and encodes it to high-level representation, while a decoder generates responses based on the encoded representation. By recursively encoding and decoding the previous information, the system makes utterances in the multi-turn dialog.

### 2.3 End-to-End System

In order to perform end-to-end training, we demonstrate the combination of the recommender system and conversation system. Specifically, the input of the recommender system is constructed based on the dialog history, which is a representation of mentioned items in the dialog. The output of the recommender system $P_{\text{rec}}$, which is a probability distribution over the item set, can be combined with the output of the dialog system $P_{\text{dialog}} \in \mathbb{R}^{|V|}$, where $V$ refers to the vocabulary. A switching mechanism (Gulcehre et al., 2016) controls the decoder to decide whether it should generate a word from the vocabulary or an item from the recommender output at a certain timestep.

$$P(w) = p_s P_{\text{dialog}}(w) + (1 - p_s)\, P_{\text{rec}}(w) \quad (1)$$

$$p_s = \sigma\left(w_s o + b_s\right) \quad (2)$$

where $w$ represents either a word from the vocabulary or an item from the item set, $o$ is the hidden representation in the final layer of the dialog

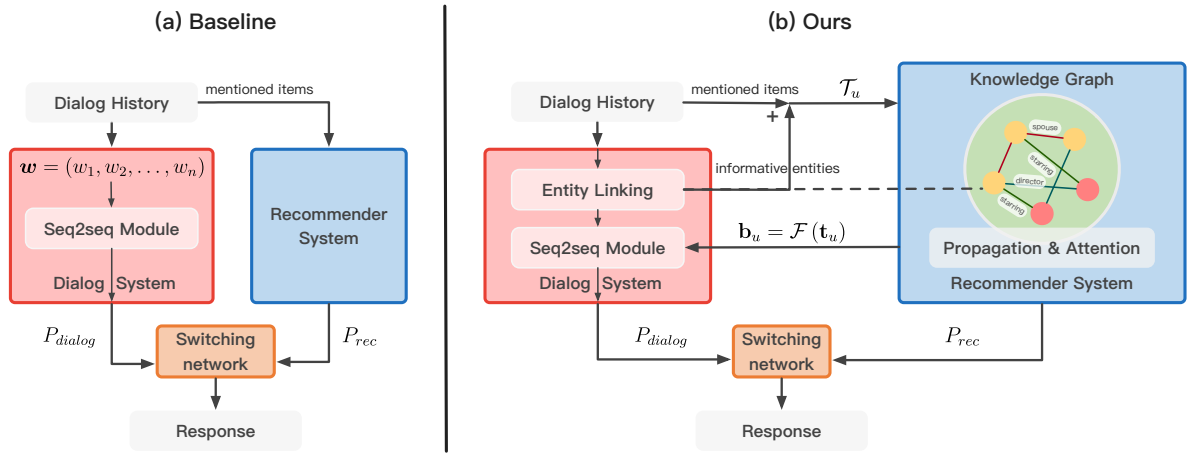


Figure 1: **Comparative illustration on modules of the existing baseline framework and our proposed KBRD framework.** (a) The connection between the recommender system and the dialog system in the baseline framework is weak. The dialog system takes the plain text of the dialog history as input and the recommender only considers mentioned items in the dialog. (b) Our framework enables interaction between the two systems. First, informative entities are linked to an external knowledge graph and sent to the recommender besides items. They are propagated on the KG via a relational graph convolutional network, enriching the representation of user interest. Second, the knowledge-enhanced user representation is sent back to the dialog system in the form of vocabulary bias, enabling it to generate responses that are consistent with the user's interest.

system. $w_s \in \mathbb{R}^d$ and $b_s \in \mathbb{R}$ are the switcher's parameters and $\sigma$ refers to the sigmoid function. Therefore, the whole system can be trained in an end-to-end fashion.

## 3 Proposed Model

In this section, we introduce our proposed framework **KBRD** that integrates the recommender system and the dialog system effectively via knowledge propagation. We show how knowledge connects the two systems and how they bring mutual benefits to each other.

### 3.1 Dialog-Aware Recommendation with Knowledge

Recommendation of the basic framework is solely based on the mentioned items in the dialog history. Such recommendation ignores contextual information in dialog that often indicates users' preferences.

Here we propose to make use of the dialog contents, including the non-item information, in the process of recommendation. Furthermore, to effectively recommend items from the non-item information, we introduce an external knowledge graph from DBpedia (Lehmann et al., 2015) to our system. The knowledge can build a connection between dialog contents and items.

**Incorporating Dialog Contents** Specifically, we have a knowledge graph $\mathcal{G}$ consisting of triples $(h, r, t)$ where $h, t \in \mathcal{E}$ and $r \in \mathcal{R}$. $\mathcal{E}$ and $\mathcal{R}$ denote the sets of entities and relations in the knowledge graph. We first match each item in the item set to entities in $\mathcal{E}$ by name.[2] We then perform entity linking (Daiber et al., 2013) on dialog contents and thus informative non-item entities appearing in dialog contents are matched to $\mathcal{E}$.[3] Therefore, we can represent a user as $\mathcal{T}_u = \{e_1, e_2, \cdots, e_{|\mathcal{T}_u|}\}$, where $e_i \in \mathcal{E}$. To be more specific, it is a set of mentioned items plus non-item entities extracted from the dialog contents, linked to the knowledge graph.

**Relational Graph Propagation** Inspired by Schlichtkrull et al. (2018), we apply Relational Graph Convolutional Networks (R-GCNs) to encode structural and relational information in the knowledge graph to entity hidden representations. An intuition behind this is that neighboring nodes in knowledge graph may share similar features that are useful for recommendation. For exam-

[2]For example, a movie item "star wars" is matched to http://dbpedia.org/resource/Star_Wars_(film) in the dbpedia knowledge graph.

[3]Utterance "I like science fiction movies." is associated with http://dbpedia.org/resource/Science_fiction_film. An utterance may be associated with one or multiple entities.

ple, when a user speaks of his/her preference on an actor/actress, the recommender should provide movies that have a close connection to that person. In addition, by taking different relations into consideration, the system models different types of neighbors more accurately.

Formally, at layer 0, we have a trainable embedding matrix $\mathbf{H}^{(0)} \in \mathbb{R}^{|\mathcal{E}| \times d^{(0)}}$ for nodes (i.e., entities) on the knowledge graph. Then, for each node $v$ in $\mathcal{E}$ at layer $l$, we compute:

$$h_v^{(l+1)} = \sigma\left(\sum_{r \in \mathcal{R}} \sum_{w \in \mathcal{N}_v^r} \frac{1}{c_{v,r}} W_r^{(l)} h_w^{(l)} + W_0^{(l)} h_v^{(l)}\right)$$

where $h_v^{(l)} \in \mathbb{R}^{d^{(l)}}$ denotes the hidden representation of node $v$ at the $l$-th layer of the graph neural network, and $d^{(l)}$ denotes the dimensionality of the representation at the layer. $\mathcal{N}_v^r$ denotes the set of neighbor indices of node $v$ under relation $r \in \mathcal{R}$. $W_r^l$ is a learnable relation-specific transformation matrix for vectors from neighboring nodes with relation $r$. $W_0^l$ is a learnable matrix for transforming the nodes' representation at the current layer. $c_{v,r}$ is a normalization constant that can either be learned or chosen in advance (e.g., $c_{v,r} = |\mathcal{N}_v^r|$).

For each node on the graph, it receives and aggregates the messages from its neighboring nodes after relation-specific transformation. Then it combines the information with its hidden representation to form its updated representation at the next layer.

Finally, at the last layer $L$, structural and relational information is encoded into the entity representation $h_v^{(L)}$ for each $v \in \mathcal{E}$. We denote the resulting knowledge-enhanced hidden representation matrix for entities in $\mathcal{E}$ as $\mathbf{H}^{(L)} \in \mathbb{R}^{|\mathcal{E}| \times d^{(L)}}$. We omit the $(L)$ in the following paragraphs for simplicity.

**Entity Attention** The next step is to recommend items to users based on knowledge-enhanced entity representations. While an item corresponds to an entity on the knowledge graph, a user may have interacted with multiple entities. Given $\mathcal{T}_u$, we first look up the knowledge-enhanced representation of entities in $\mathcal{T}_u$ from $\mathbf{H}$, and we have:

$$\mathbf{H}_u = (h_1, h_2, \cdots, h_{|\mathcal{T}_u|}) \tag{3}$$

where $h_i \in \mathbb{R}^d$ is the hidden vector of entity $e_i$. Here our objective is to encode this vector set of variable size to a vector of fixed size so that we can compute the similarity between user and item. Instead of simply averaging these vectors, we choose a linear combination of the $|\mathcal{T}_u|$ vectors. Specifically, we apply self-attention mechanism (Lin et al., 2017) that takes $\mathbf{H}_u$ as input and outputs a distribution $\alpha_u$ over $|\mathcal{T}_u|$ vectors:

$$\alpha_u = \text{softmax}\left(w_{a2} \tanh\left(W_{a1} \mathbf{H}_u^T\right)\right) \tag{4}$$

where $W_{a1} \in \mathbb{R}^{d_a \times d}$ is a weight matrix and $w_{a2}$ is a vector of parameters with size $d_a$. The final representation of user $u$ is computed as follows:

$$t_u = \alpha_u \mathbf{H}_u \tag{5}$$

This enables the recommender system to consider the importance of different items and non-item entities in the dialog. Finally, the output of our recommender is computed as follows:

$$P_{\text{rec}} = \text{softmax}(\text{mask}(t_u \mathbf{H}^T)) \tag{6}$$

where mask is an operation that sets the score of non-item entities to $-\infty$. The masking operation ensures that the recommendations are all items.

### 3.2 Recommendation-Aware Dialog

Instead of applying HRED, we introduce the Transformer framework to the dialog system in this task. Transformer (Vaswani et al., 2017) can reach significantly better performances in many tasks, such as machine translation (Vaswani et al., 2017; Ott et al., 2018), question answering (Rajpurkar et al., 2016; Yang et al., 2018; Ding et al., 2019) and natural language generation (Liu et al., 2018; Chen et al., 2019). In our preliminary experiments, we have found that Transformer can also achieve better performance than HRED in this task, and thus we apply this framework to the dialog system.

The Transformer is also an encoder-decoder framework for sequence-to-sequence learning. The Transformer encoder consists of an embedding layer and multiple encoder layers. Each encoder layer has a self-attention module and a Point-Wise Feed-Forward Network (FFN). The encoder encodes the dialog history $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ to high-level representations $\boldsymbol{s} = (s_1, s_2, \ldots, s_n)$. Similarly, the Transformer decoder contains an embedding layer and multiple decoder layers with self-attention and FFN. Moreover, each of them contains a multi-head context

attention to extract information from the source-side context. The decoder generates a representation $o$ at each decoding time step.

In order to predict a word at each decoding time step, the top layer of the decoder, namely the output layer, generates a probability distribution over the vocabulary:

$$P_{\text{dialog}} = \text{softmax}\left(Wo + b\right) \tag{7}$$

where $W \in \mathbb{R}^{|V| \times d}$ and $b \in \mathbb{R}^{|V|}$ are weight and bias parameters, and $V$ refers to the vocabulary.

However, so far the dialog system is completely conditioned on the plain text of the dialog contents. By further introducing the recommender system's knowledge of the items that have appeared in dialog, we guide the dialog system to generate responses that are more consistent with the user's interests. Specifically, we add a vocabulary bias $b_u$ to the top layer of the decoder inspired by Michel and Neubig (2018). Different from their work, $b_u$ is computed based on the recommender system's hidden representation of user $u$:

$$b_u = \mathcal{F}(t_u) \tag{8}$$

where $\mathcal{F} : \mathbb{R}^d \rightarrow \mathbb{R}^{|V|}$ represents a feed-forward neural network and $t_u$ is the user representation in the recommendation context introduced in Equation 5.

Therefore, the computation of the top layer of the decoder becomes:

$$P_{\text{dialog}} = \text{softmax}\left(Wo + b + b_u\right) \tag{9}$$

So far, we have built an end-to-end framework that bridges the recommender system and the dialog system, which enables mutual benefits between the systems.

## 4 Experiments

In this section, we provide an introduction to the details of our experiments, including dataset, setting, evaluation as well as further analyses.

### 4.1 Dataset

*REcommendations through DIALog (REDIAL)* is a dataset for conversational recommendation. Li et al. (2018) collected the dialog data and built the dataset through Amazon Mechanical Turk (AMT). With enough instructions, the workers on the platform generated dialogs for recommendation on movies. Furthermore, in order to achieve and dialog-aware recommendation, besides movies, we introduce the relevant entities, such as director and style, from DBpedia. The number of conversations is 10006 and the number of utterances is 182150. The total number of users and movies are 956 and 51699 respectively.

### 4.2 Setting

We implement the models in PyTorch and train on an NVIDIA 2080Ti. For the recommender, both the entity embedding size $d^{(0)}$ and the hidden representation size $d^{(l)}$ are set to 128. We choose the number of R-GCN layers $L = 1$ and the normalization constant $c_{v,r}$ to 1. For Transformer, all input embedding dimensions and hidden sizes are set to 300. During training, the batch size is set to 64. We use Adam optimizer (Kingma and Ba, 2015) with the setting $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 1 \times 10^{-8}$. The learning rate is 0.003 for the recommender and 0.001 for the Transformer. Gradient clipping restricts the norm of the gradients within [0, 0.1].

### 4.3 Evaluation Metrics

The evaluation of dialog consists of automatic evaluation and human evaluation. The metrics for automatic evaluation are perplexity and distinct n-gram. Perplexity is a measurement for the fluency of natural language. Lower perplexity refers to higher fluency. Distinct n-gram is a measurement for the diversity of natural language. Specifically, we use distinct 3-gram and 4-gram at the sentence level to evaluate the diversity. As to human evaluation, we collect ten annotators with knowledge in linguistics and require them to score the candidates on the consistency with the dialog history. We sample 100 multi-turn dialogs from the test set together with the models' corresponding responses, and require them to score the consistency of the responses.[4] The range of score is 1 to 3.

The evaluation for recommendation is Recall@K. We evaluate that whether the top-k items selected by the recommender system contain the ground truth recommendation provided by human recommenders. Specifically, we use Recall@1, Recall@10, and Recall@50 for the evaluation.

[4]Note that we did not provide the utterances of the baseline Transformer to annotators. Based on our observation, the generations of the Transformer-based models are significantly different from those of REDIAL. In case that annotators had knowledge about models, we did not require them to score the utterances of Transformer.

| Model | R@1 | R@10 | R@50 |
|---|---|---|---|
| REDIAL | 2.3±0.2 | 12.9±0.7 | 28.7±0.9 |
| **KBRD (D)** | 2.7±0.2 | 14.0±0.6 | 30.6±0.7 |
| **KBRD (K)** | 2.6±0.2 | 14.4±0.9 | 31.0±1.2 |
| **KBRD** | **3.0**±0.2 | **16.3**±0.3 | **33.8**±0.7 |

Table 2: **Evaluation of the recommender system.** We report the results of Recall@1, Recall@10 and Recall@50 of the models ($p \ll 0.01$). **KBRD (D)** stands for only incorporating the dialog contents. **KBRD (K)** stands for only incorporating knowledge. The results demonstrate that both the interaction with the dialog system and the external knowledge are helpful for the improvement of model performance, and our proposed model reaches the best performance on the three metrics.

| Model | PPL | Dist-3 | Dist-4 | CSTC |
|---|---|---|---|---|
| REDIAL | 28.1 | 0.11 | 0.13 | 1.73 |
| Transformer | 18.0 | 0.27 | 0.39 | - |
| **KBRD** | **17.9** | **0.30** | **0.45** | **1.99** |

Table 3: **Automatic and human evaluation of dialog generation.** For automatic evaluation, we evaluate the perplexity (PPL) and distinct n-gram (Dist-3 and Dist-4 refer to distinct 3-gram and 4-gram respectively) of the generated dialogs. For human evaluation, we ask human annotators to evaluate the consistency (CSTC) of the generated utterances with the dialog history. Our proposed method performs the best in all evaluations compared with the baselines.

### 4.4 Baselines

The baseline models for the experiments are illustrated in the following:

- **REDIAL** This is a basic model for conversational recommendation. It basically consists of a dialog generation system based on HRED (Sordoni et al., 2015a; Serban et al., 2016), a recommendation system based on autoencoder and a sentiment analysis module.

- **Transformer** We name our implemented baseline model Transformer. It is similar to REDIAL, but its dialog generation system is based on the model Transformer (Vaswani et al., 2017). Except for that, the others remain the same.

### 4.5 Results

In the following, we present the results of our experiments, including the model performances in recommendation and dialog generation.

**Recommendation** To evaluate the effects of our recommendation system, we conduct an evaluation of Recall@K. We present the results in Table 2. From the results, it can be found that our proposed model reaches the best performances in the evaluation of Recall@1, Recall@10 and Recall@50. Furthermore, we also demonstrate an ablation study to observe the contribution of the dialog system and the introduced knowledge. It can be found that either dialog or knowledge can bring improvement to the performance of the recommendation system. Their combination improves the performance the most by +0.7 Recall@1, +3.4 Recall@10 and +5.1 Recall@50, which are advantages of 30.4%, 26.4% and 17.8% respectively. This shows that the information from both sources is contributive. The dialog contains users' preferred items as well as attributes, such as movie director and movie style, so that the system can find recommendation based on these inputs. The knowledge contains important features of the movie items so that the system can find items with similar features. Further, the combination brings an advantage even greater than sum of the two parts, which proves the effectiveness of our model.

**Dialog** Table 3 shows the results of the evaluation of the baseline models and our proposed method in dialog generation. In the evaluation of perplexity, Transformer has much lower perplexity (18.0) compared to REDIAL (28.1), and **KBRD** can reach the best performance in perplexity. This demonstrates the power of Transformer in modeling natural language. In the evaluation of diversity, we find that the models based on Transformer significantly outperform REDIAL from the results of distinct 3-gram and 4-gram. Besides, it can be found that **KBRD** has a clear advantage in diversity over the baseline Transformer. This shows that our model can generate more diverse contents without decreasing fluency.

As to the human evaluation, we ask human annotators to score the utterances' consistency with their dialog history. Compared with REDIAL **KBRD** reaches better performance by +0.22 consistency score, which is an advantage of 15%. Moreover, considering the range is between 1 and 3, this is a large gap between the performances

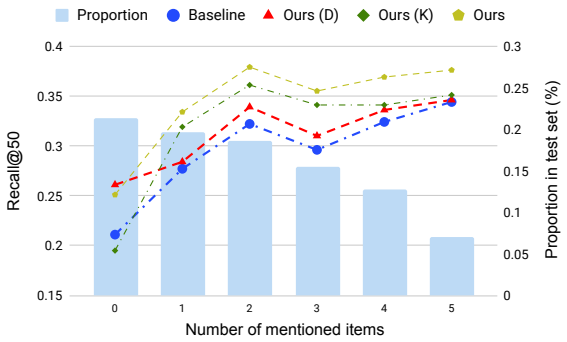


Figure 2: **Performance of the recommender system with different numbers of mentioned items.** The x-axis refers to the number of mentioned items in the dialog, the y-axis for the line chart (on the left) refers to the model performance on the Recall@50 evaluation, and the y-axis for the histogram (on the right) refers to proportion in the test set. This shows recommendation is much more difficult with few items mentioned (i.e., at the first few rounds in dialog). Leveraging dialog contents makes a great difference in this situation.

of the two models in this evaluation. To make a consistent response in a dialog, the model should understand the dialog history and better learn the user's preference. The baseline REDIAL does not have a strong connection between the dialog system and user representation. Instead, in our framework, the recommender system provides the recommendation-aware vocabulary bias $b_u$, which is based on the user representation $t_u$, to the dialog system. Thus the dialog system gains knowledge about the user's preference and generates a consistent response.

## 5 Discussion

In this section, we conduct a series of analyses to observe the effects of our proposed model. We discuss how dialog can improve the recommendation performance and how recommendation can enhance the dialog quality.

### 5.1 Does dialog help recommendation?

We first evaluate whether the dialog contents can benefit the recommendation system. The results of the evaluation are demonstrated in Figure 2. From the histogram in the figure, we observe that most of the dialogs contain only a few mentioned movies. The dialogs with only 0-2 mentioned movies take up a proportion of 62.8% of the whole testing dataset. Therefore, it is important for the system to perform high-quality recommendation with only a small number of mentioned movies. This also corresponds to the classical problem "cold start" (Schein et al., 2002) in the recommender system. In real applications, we also expect that the system can perform high-quality recommendation with fewer rounds. This represents the efficiency of the recommender system, which can save users' time and efforts.

Specifically, we demonstrate the performances of four systems in Figure 2. They are the basic framework, the one only with the interaction with the dialog system, the one only with the external knowledge and **KBRD** with both dialog and knowledge incorporation. From the figure, it can be found that while there is no mentioned item in the dialog, the baseline and the one only with knowledge perform the worst. In contrast, the two models with dialog incorporation perform significantly better. This shows that the context in the dialog contains much useful non-item information about users' preferences, such as director, actor/actress in movie recommendation. Therefore, while there is no mentioned item, the recommender system can still perform high-quality recommendation based on the contextual information. With the increase of mentioned items, the contribution of knowledge becomes more significant than the dialog. On average, the system with both information sources performs the best. Dialog introduces contextual information and knowledge introduces movie features and structural connection with other movies.

### 5.2 Does recommendation help dialog?

In Section 4.5, we present the performances of the baselines and our model **KBRD** in dialog generation. It can be found that the interaction with the recommendation system can enhance the performance of the dialog system in both automatic evaluation and human evaluation. Also, an example of the responses of different models is shown in Table 1. With the dialog history, the baseline REDIAL simply uses a generic response with a recommended movie. Instead, **KBRD** has more concern about the mentioned items apart from the plain text of dialog history. The user representation from our recommender system contains such information, which is sent to the dialog system to form a vocabulary bias. With such information, **KBRD** has a better understanding of both the dialog history as well as the user's preference, and thus generates a consistent response.

| Movie | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| Star Wars | space | alien | sci-fi | star | sci | robot | smith | harry |
| The Shining | creepy | stephen | gory | horror | scary | psychological | haunted | thriller |
| The Avengers (2012) | marvel | superhero | super | dc | wait | batman | thor | take |
| Beauty and the Beast | cute | disney | animated | live | music | child | robin | kids |

Table 4: **Examples of top 8 vocabulary bias**. Given a mentioned movie, we visualize words with high probability based on the recommendation-aware vocabulary bias. The examples show that the biased words have strong connections with their corresponding movies. While the vocabulary bias benefits the dialog system to generate responses that are more consistent with user interest, they also contribute to the explainability of recommendation.

To further study the effects of the recommender system on dialog generation, we display the top biased words from the vocabulary bias. Note that in **KBRD** a connection between the recommender system and dialog system is the recommendation-aware vocabulary bias $b_u$. To be specific, we compute the recommendation-aware bias $b_u$ in dialog and select the components with the top-8 largest values.[5] Then we record the corresponding words and observe whether these words are related to the mentioned movies. We present several examples in Table 4. From the table, we observe that the words are highly related to the mentioned movies. For example, when "The Shining" is mentioned, some of the top biased words are "creepy", "gory" and "scary", which are consistent with the style of the horror movie, and "stephen", who is the original creator of the movie. Therefore, it can be suggested that the recommendation system conveys important information to the dialog system in the form of a vocabulary bias. Furthermore, these biased words can also serve as explicit explanation to recommendation results. From this perspective, this shows the interpretability of our model.

## 6 Related Work

Recommender systems aim to find a small set of items that meet users' interest based on users' historical interactions. Traditional recommender systems rely on collaborative filtering (Resnick et al., 1994; Sarwar et al., 2001), and recent advances in this field rely much on neural networks (Wang et al., 2015; He et al., 2017; Ying et al., 2018). To deal with the cold-start problem and the sparsity of user-item interactions which these methods usually suffer, researchers have proposed methods to incorporate external information, such as heterogeneous information networks (Yu et al., 2014), knowledge bases (Zhang et al., 2016; Wang et al., 2018a) and social networks (Jamali and Ester, 2010). Besides accuracy, explainability is also an important aspect when evaluating recommender systems (Zhang et al., 2014; Zhang and Chen, 2018; Wang et al., 2018b).

End-to-end dialog systems based on neural networks have shown promising performance in open-ended settings (Vinyals and Le, 2015; Sordoni et al., 2015b; Dodge et al., 2016; Wen et al., 2015) and goal-oriented applications (Bordes et al., 2017). Recent literature also explores the intersection of end-to-end dialog systems with other intelligence systems and creates new tasks such as visual dialog (Das et al., 2017; De Vries et al., 2017), conversational recommendation (Li et al., 2018). In particular, Li et al. (2018) collects a dataset of conversations focused on providing movie recommendations and proposes a baseline model for end-to-end training of recommender and dialog systems. Earlier studies in this field focus on different tasks such as minimizing the number of user queries (Christakopoulou et al., 2016), training the dialog agent to ask for facet values for recommendation (Sun and Zhang, 2018). Related literature can also be found in Thompson et al. (2004), Mahmood and Ricci (2009), Chen and Pu (2012), Widyantoro and Baizal (2014) and Liao et al. (2019).

## 7 Conclusion

In this paper, we propose a novel end-to-end framework, **KBRD**, which bridges the gap between the recommender system and the dialog system via knowledge propagation. Through a series of experiments, we show that **KBRD** can reach better performances in both recommendation and dialog generation in comparison with the baselines. We also discuss how the two systems benefit each other. Dialog information is effective for the recommender system especially in the setting of cold start, and the introduction of knowledge can strengthen the recommendation performance

[5] After stop words filtering.

significantly. Information from the recommender system that contains the user preference and the relevant knowledge can enhance the consistency and diversity of the generated dialogs.

## Acknowledgements

The work is supported by NSFC for Distinguished Young Scholar (61825602), NSFC (61836013), and a research fund supported by Alibaba. The authors would like to thank Lei Li and Chang Zhou for their insightful feedback, and responsible reviewers of EMNLP-IJCNLP 2019 for their valuable suggestions. Jie Tang is the corresponding author.

## References

Antoine Bordes, Y.-Lan Boureau, and Jason Weston. 2017. Learning end-to-end goal-oriented dialog. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Li Chen and Pearl Pu. 2012. Critiquing-based recommenders: survey and emerging trends. *User Modeling and User-Adapted Interaction*, 22(1-2):125–150.

Qibin Chen, Junyang Lin, Yichang Zhang, Hongxia Yang, Jingren Zhou, and Jie Tang. 2019. Towards knowledge-based personalized product description generation in e-commerce. *arXiv preprint arXiv:1903.12457*.

Konstantina Christakopoulou, Filip Radlinski, and Katja Hofmann. 2016. Towards conversational recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 815–824. ACM.

Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N Mendes. 2013. Improving efficiency and accuracy in multilingual entity extraction. In *Proceedings of the 9th International Conference on Semantic Systems*, pages 121–124. ACM.

Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José MF Moura, Devi Parikh, and Dhruv Batra. 2017. Visual dialog. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 326–335.

Harm De Vries, Florian Strub, Sarath Chandar, Olivier Pietquin, Hugo Larochelle, and Aaron Courville. 2017. Guesswhat?! visual object discovery through multi-modal dialogue. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5503–5512.

Ming Ding, Chang Zhou, Qibin Chen, Hongxia Yang, and Jie Tang. 2019. Cognitive graph for multi-hop reading comprehension at scale. In *ACL (1)*, pages 2694–2703. Association for Computational Linguistics.

Jesse Dodge, Andreea Gane, Xiang Zhang, Antoine Bordes, Sumit Chopra, Alexander H. Miller, Arthur Szlam, and Jason Weston. 2016. Evaluating prerequisite qualities for learning end-to-end dialog systems. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 140–149.

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, pages 173–182. International World Wide Web Conferences Steering Committee.

Mohsen Jamali and Martin Ester. 2010. A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain, September 26-30, 2010*, pages 135–142. ACM.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015*.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195.

Raymond Li, Samira Ebrahimi Kahou, Hannes Schulz, Vincent Michalski, Laurent Charlin, and Chris Pal. 2018. Towards deep conversational recommendations. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018.*, pages 9748–9758.

Lizi Liao, Ryuichi Takanobu, Yunshan Ma, Xun Yang, Minlie Huang, and Tat-Seng Chua. 2019. Deep conversational recommender in travel. *CoRR*, abs/1907.00710.

Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *5th International Conference on Learning Representations, ICLR 2017*.

Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating wikipedia by summarizing long sequences. In *6th International Conference on Learning Representations, ICLR 2018*.

Tariq Mahmood and Francesco Ricci. 2009. Improving recommender systems with adaptive conversational strategies. In *Proceedings of the 20th ACM conference on Hypertext and hypermedia*, pages 73–82. ACM.

Paul Michel and Graham Neubig. 2018. Extreme adaptation for personalized neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018*, pages 312–318.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. 2018. Scaling neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers, WMT 2018*, pages 1–9.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *EMNLP*, pages 2383–2392. The Association for Computational Linguistics.

Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. 1994. Grouplens: An open architecture for collaborative filtering of netnews. In *CSCW '94, Proceedings of the Conference on Computer Supported Cooperative Work, Chapel Hill, NC, USA, October 22-26, 1994*, pages 175–186. ACM.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW'01*, pages 285–295.

Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. 2002. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260. ACM.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer.

Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th International Conference on World Wide Web Companion, WWW 2015*, pages 111–112.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 3776–3784.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015a. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015*, pages 553–562.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015b. A neural network approach to context-sensitive generation of conversational responses. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 196–205.

Yueming Sun and Yi Zhang. 2018. Conversational recommender system. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 235–244. ACM.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014*, pages 3104–3112.

Cynthia A Thompson, Mehmet H Goker, and Pat Langley. 2004. A personalized system for conversational recommendations. *Journal of Artificial Intelligence Research*, 21:393–428.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, pages 6000–6010.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1235–1244.

Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2018a. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 417–426. ACM.

Xiting Wang, Yiru Chen, Jie Yang, Le Wu, Zhengtao Wu, and Xing Xie. 2018b. A reinforcement learning framework for explainable recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 587–596. IEEE.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721.

Dwi H Widyantoro and ZKA Baizal. 2014. A framework of conversational recommender system based on user functional requirements. In *2014 2nd international conference on information and communication technology (ICoICT)*, pages 160–165. IEEE.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*, pages 2369–2380. Association for Computational Linguistics.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 974–983. ACM.

Xiao Yu, Xiang Ren, Yizhou Sun, Quanquan Gu, Bradley Sturt, Urvashi Khandelwal, Brandon Norick, and Jiawei Han. 2014. Personalized entity recommendation: A heterogeneous information network approach. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 283–292. ACM.

Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 353–362. ACM.

Yongfeng Zhang and Xu Chen. 2018. Explainable recommendation: A survey and new perspectives. *arXiv preprint arXiv:1804.11192*.

Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. 2014. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 83–92. ACM.

# Structuring Latent Spaces for Stylized Response Generation

**Xiang Gao** **Yizhe Zhang** **Sungjin Lee** *
**Michel Galley** **Chris Brockett** **Jianfeng Gao** **Bill Dolan**
Microsoft Research, Redmond, WA, USA
{xiag,billdol}@microsoft.com

## Abstract

Generating responses in a targeted style is a useful yet challenging task, especially in the absence of parallel data. With limited data, existing methods tend to generate responses that are either less stylized or less context-relevant. We propose StyleFusion, which bridges conversation modeling and non-parallel style transfer by sharing a structured latent space. This structure allows the system to generate stylized relevant responses by sampling in the neighborhood of the conversation model prediction, and continuously control the style level. We demonstrate this method using dialogues from Reddit data and two sets of sentences with distinct styles (arXiv and Sherlock Holmes novels). Automatic and human evaluation show that, without sacrificing appropriateness, the system generates responses of the targeted style and outperforms competitive baselines. [1]

## 1 Introduction

A social chatbot designed to establish long-term emotional connections with users must generate responses that not only match the *content* of user input and context, but also do so in a desired target *style* (Zhou et al., 2018; Li et al., 2016b; Luan et al., 2016; Gao et al., 2019a). A conversational agent that speaks in a polite, professional tone is likely to facilitate service in customer relationship scenarios; likewise, an agent that sounds like an cartoon character or a superhero can be more engaging in a theme park. The master of response style is also an important step towards human-like chatbots. As highlighted in social psychology studies (Niederhoffer and Pennebaker, 2002a,b), when two people are talking, they tend to match linguistic style of each other, sometime even regardless of their intentions. Achieving this level of performance, however, is challenging. Lacking parallel data in different conversational styles, researchers often resort to what we will term *style datasets* that are in non-conversational format (e.g. news, novels, blogs). Since the contents and formats of these are quite different from conversation data, existing approaches tend to generate responses that are either less style-specific (Luan et al., 2017) or less context-relevant (Niu and Bansal, 2018).

We suggest that this trade-off between appropriateness and style stems from profound differences

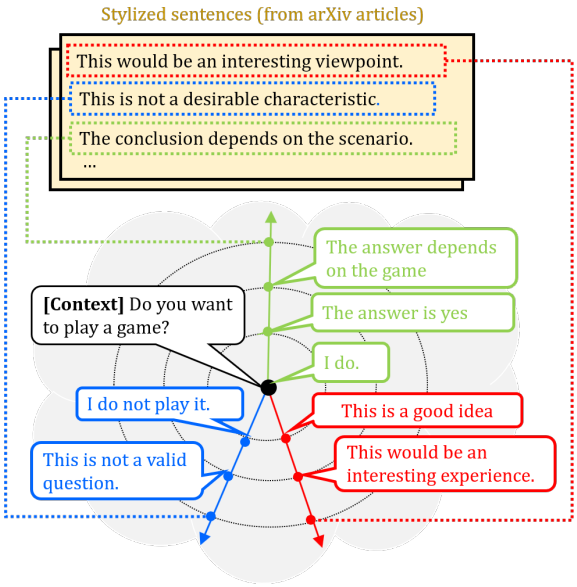


Figure 1: StyleFusion helps conversational model to distill style from non-conversational, non-parallel sentences by mapping them to points surrounding the related conversations in the structured latent space. Direction and distance from the model prediction (center black dot) roughly correspond to contents and style intensity, respectively, illustrated by examples taken from Table 2.

*Now at Alexa AI, Amazon.

[1] An implementation of our model and the scripts to generate the datasets are available at https://github.com/golsun/StyleFusion.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1814–1823, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

between conversation and style datasets in format, style and contents that impede joint learning. One approach has been to combine these only during decoding: Niu and Bansal (2018) trained two models separately, a Sequence-to-Sequence (S2S) (Sutskever et al., 2014) on a conversation dataset and a language model (LM) on a style dataset. At inference time, they take a weighted average of the token probability distribution of the two models to predict the next token. This forced bias, however, degrades output relevance. An alternative approach attempts to map the two datasets into the same latent space: Luan et al. (2017) use multi-task learning to train a S2S model on a conversation dataset and an autoencoder (AE) on a style dataset. Gao et al. (2019b) point out that the two datasets still form separate clusters in the latent space; below we observe that this leads to a low style intensity in generated responses (Section 5).

We propose to bridge conversation modeling and non-parallel style transfer by structuring a shared latent space using novel regularization techniques, that we dub StyleFusion. In contrast to Luan et al. (2017), the two datasets are well aligned in the latent space by generalizing SpaceFusion [2] (Gao et al., 2019b), an approach that aligns latent spaces for paired samples, to non-parallel datasets. In the structured shared latent space, stylized sentences are nudged adjacent to semantically related conversations, thereby allowing the system to generate style-specific relevant responses by sampling in the neighborhood of the model prediction. Distance and direction from the model prediction roughly match the style intensity and content diversity of generated responses, respectively, as illustrated in Fig. 1

We demonstrate this method using dialogues from Reddit data and two sets of sentences with distinct styles (arXiv and Sherlock Holmes novels). Automatic and human evaluation show that, without sacrificing appropriateness, our system can generate responses in a targeted style and outperforms competitive baselines.

Our contribution can be summarized thus: 1) We introduce an end-to-end approach that generates style-specific responses from conversational data and non-parallel non-conversation style data. 2) We generalize the SpaceFusion model of (Gao et al., 2019b) to non-parallel data by a new regularization method. 3) We present a visualization analysis that provides intuitive insights into the drawbacks of alternative approaches.

[2]Integrated into Microsoft Icecaps toolkit (Shiv et al., 2019) https://github.com/microsoft/icecaps.

## 2 Related Work

**Text style transfer** is a related but distinct task. It usually preserves the content (Yang et al., 2018; Hu et al., 2017; Fu et al., 2018; Shen et al., 2017; Gong et al., 2019). In contrast, content of conversational responses in a given context can be semantically diverse. Various approaches have been proposed for non-parallel data setup. Fu et al. (2018) proposed to use separate decoders for different styles and a classifier to measure style strength. Shen et al. (2017) proposed to map texts of two different styles into a shared latent space where the "content" information is preserved and "style" information is discarded. An adversarial discriminator is used to align the latent spaces of two different styles. However, Yang et al. (2018) point out the difficulty of training an adversarial discriminator and proposed instead the use of language models as discriminator. Like Shen et al. (2017); Yang et al. (2018), we align latent spaces for different styles. However we also align latent spaces encoded by different models (S2S and AE).

**Stylized response generation** is a relatively new task. Akama et al. (2017) use a stylized conversation corpus to fine-tune a conversation model pretrained on a background conversation dataset. However, stylized texts are usually in non-conversational format, as in the present setting. Niu and Bansal (2018) proposed a method that takes the weighted average of the token probability distribution predicted by a S2S trained on background conversational dataset and that predicted by a LM trained on style dataset as the token probability. They observed reduced relevance and attributed this to the fact that the LM was not trained to attend to conversation context and S2S was not trained to learn style during training. In contrast, we jointly learn from conversation and style datasets during training. Niu and Bansal (2018) have proposed label-fine-tuning, but this is limited to scenarios where a reasonable portion of the conversational dataset is in the target style, which is not always the case.

**Persona-grounded conversation modeling** Li et al. (2016b); Luan et al. (2017) aim to generate responses mimicking a speaker. It is closely re-

lated to the present task, since persona is, broadly speaking, the manifestation of a type of style. Li et al. (2016b) feeds a speaker ID to the decoder to promote generation of response for that target speaker. However non-conversational data cannot be used. Luan et al. (2017) applied a multi-task learning approach to utilize non-conversational data. A S2S model, taking in conversational data, and an autoencoder (AE), taking in non-conversational data, share the decoder and are trained alternately. However, Gao et al. (2019b) observed that sharing the decoder may not truly allow S2S and AE to share the latent space, and thus S2S may not fully utilize what is learned by AE. Unlike Li et al. (2016b) using labelled persona IDs, Zhang et al. (2019) have proposed using a self-supervised method to extract persona features from conversation history. This allows modeling persona dynamically, which agrees with the fact that even the same person can speak in different style in different scenarios.

**Multi-task learning** McCann et al. (2018); Liu et al. (2019); Luan et al. (2017); Gao et al. (2019b); Zhang et al. (2017) aggregates the strengths of each specific task, and induces regularization effects (Liu et al., 2019) as the model is trained to learn a more universal representation. However a simple multi-task approach (Luan et al., 2017) may learn separate representations for each dataset (Gao et al., 2019b). To address this, in previous work (Gao et al., 2019b), we proposed the SPACEFUSION model featuring a regularization technique that explicitly encourages alignment of latent spaces for a universal representation. SPACEFUSION, however, is only designed for paired samples. We generalize SPACEFUSION to non-parallel datasets in this paper.

## 3 The STYLEFUSION Model

### 3.1 Problem statement

Let $\mathcal{D}_{conv} = [(x_0, y_0), (x_1, y_1), \cdots, (x_n, y_n)]$ denote a conversation dataset, where $x_i$ and $y_i$ are context sentences and a corresponding response, respectively. $x_i$ consists of one or more utterances and $y_i$ is only one utterance. $\mathcal{D}_{style} = [s_0, s_1, \cdots, s_m]$ denotes a non-conversational style dataset, where $s_i$ is a sentence sampled from a corpus of the targeted style. Samples from $\mathcal{D}_{style}$ do not have a labelled corresponding relation with samples from $\mathcal{D}_{conv}$ (thus

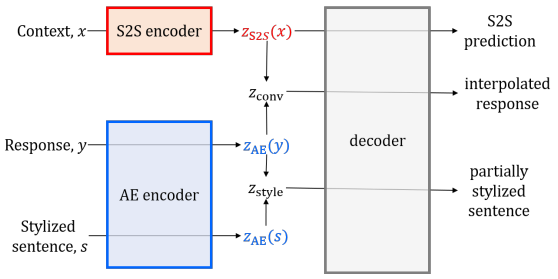


Figure 2: STYLEFUSION model architecture.

"non-parallel"). Our aim is to train a model jointly on $\mathcal{D}_{style}$ and $\mathcal{D}_{conv}$ to generate appropriate responses in the style similar to sentences from $\mathcal{D}_{style}$, to a given context. The iven context may or may not be in the target style.

### 3.2 Training

In contrast to SPACEFUSION(Gao et al., 2019b), which only fuses context-response pairs, our goal is to additionally map related stylized sentences to points surrounding the context in the shared latent representation space. The system can then generate relevant stylized responses by sampling in the neighborhood of the prediction based on the context.

**Model overview.** As illustrated in figure 2, the model consists of a sequence-to-sequence (S2S) module and an autoencoder (AE) module that shares a decoder. We use S2S to produce the prediction representation $z_{S2S}(x_i)$, and AE to obtain the representation of the corresponding responses $z_{AE}(y_i)$ and stylized sentences $z_{AE}(s_j)$. We use generalized regularization terms, fusion and smoothness, to align the three latent spaces $z_{S2S}(x_i)$, $z_{AE}(y_i)$, and $z_{AE}(s_j)$.

**Fusion objectives** encourage different latent spaces to be close to each other. Accordingly, we define the cross-latent-space distances to be minimized. For response appropriateness, as $x_i$ and $y_i$ are paired as context and response, we use their pair-wise dissimilarity, following Gao et al. (2019b)

$$d_{\text{conv}} = \sum_{i \in \text{batch}} \frac{d_E(z_{\text{S2S}}(x_i), z_{\text{AE}}(y_i))}{n\sqrt{l}} \quad (1)$$

where $n$ is the batch size, $l$ is the dimension of latent space , and we use $d_E$, the Euclidean distance in latent space, as the dissimilarity.

For style transfer, however, $x_i$ and $s_j$ is not paired. Thus, we instead minimize the distance

between a point and its nearest neighbor from another dataset to pull these two datasets close to each other in the shared latent space.

$$d_{\text{style}} = \frac{1}{2} d_{\text{NN}}^{\text{cross}}(\{z_{\text{S2S}}(x_i)\}, \{z_{\text{AE}}(s_i)\}) + \frac{1}{2} d_{\text{NN}}^{\text{cross}}(\{z_{\text{AE}}(s_i)\}, \{z_{\text{S2S}}(x_i)\}) \quad (2)$$

where $d_{NN}^{\text{cross}}(\{a_i\}, \{b_i\})$ is the batch average of the distance between $a_i$ and $b_{\text{NN of } a_i}$ – the nearest neighbor (NN) of $a_i$ from set $\{b_i\}$

$$d_{\text{NN}}^{\text{cross}}(\{a_i\}, \{b_i\}) = \sum_{i \in \text{batch}} \frac{d_E(a_i, b_{\text{NN of } a_i})}{n\sqrt{l}} \quad (3)$$

While minimizing the cross-latent-space distances, $d_{\text{conv}}$ and $d_{\text{style}}$, we want the samples from the same latent space spread out, following Gao et al. (2019b). For this purpose, Gao et al. (2019b) maximized the average of capped distance between points from the same latent space. However, we found that the results are sensitive to the cap value. Instead, we define the following nearest-neighbor-based characteristic distance

$$d_{\text{spread-out}} = \min[d_{\text{NN}}^{\text{same}}(\{z_{\text{AE}}(y_i)\}), d_{\text{NN}}^{\text{same}}(\{z_{\text{AE}}(s_i)\}), d_{\text{NN}}^{\text{same}}(\{z_{\text{S2S}}(x_i)\})] \quad (4)$$

$$d_{\text{NN}}^{\text{same}}(\{a_i\}) = \sum_{i \in \text{batch}} \frac{d_E(a_i, a_{\text{NN of } a_i})}{n\sqrt{l}} \quad (5)$$

Combining these loss terms we have the following two objectives:

$$\mathcal{L}_{\text{fuse,conv}} = d_{\text{conv}} - d_{\text{spread-out}} \quad (6)$$

$$\mathcal{L}_{\text{fuse,style}} = d_{\text{style}} - d_{\text{spread-out}} \quad (7)$$

**Smoothness objective** encourages smooth semantic transition in the shared latent space. For response appropriateness, following Gao et al. (2019b), we encourage the interpolation between the prediction $z_{S2S}(x_i)$ and the target response $z_{AE}(y_i)$ to generate the target response $y_i$.

$$z_{\text{conv}} = (1-u) z_{\text{AE}}(y) + u z_{\text{S2S}}(x) + \epsilon \quad (8)$$

$$\mathcal{L}_{\text{smooth,conv}} = -\frac{1}{|y|} \log p(y|z_{\text{conv}}) \quad (9)$$

where $u \sim U(0,1)$ is a uniformly distributed random variable, and $\epsilon$ is a Gaussian noise with zero mean and covariance matrix of $\sigma^2 I$.

For style transfer, as we move from a non-stylized sentence $z_{\text{AE}}(x)$ to a random stylized sentence $z_{\text{AE}}(s)$, we expect to generate a partially stylized sentence and encourage the generated sentence to gradually change from $x$ to $s$.

$$z_{\text{style}} = (1-u) z_{\text{AE}}(x) + u z_{\text{AE}}(s) + \epsilon \quad (10)$$

$$\mathcal{L}_{\text{smooth,style}} = -(1-u)\frac{1}{|x|} \log p(x|z_{\text{style}}) - u\frac{1}{|s|} \log p(s|z_{\text{style}}) \quad (11)$$

**Training objective** to be minimized is a combination of a vanilla S2S and the above regularization terms [3]. $\mathcal{L}_{\text{smooth,style}}$ and $\mathcal{L}_{\text{fuse,style}}$ are new terms not existing in (Gao et al., 2019b). A more compact definition $\mathcal{L}_{\text{conv}} = \mathcal{L}_{\text{smooth,conv}} + \mathcal{L}_{\text{fuse,conv}}$ and $\mathcal{L}_{\text{style}} = \mathcal{L}_{\text{smooth,style}} + \mathcal{L}_{\text{fuse,style}}$ yields

$$\mathcal{L} = -\frac{1}{|y|} \log p(y|z_{\text{S2S}}) + \mathcal{L}_{\text{conv}} + \mathcal{L}_{\text{style}} \quad (12)$$

For the case $\mathcal{D}_{style}$ is much smaller than $\mathcal{D}_{conv}$, as in the present work, the model may overfit on $\mathcal{D}_{style}$. We propose to firstly pretrain the model on $\mathcal{D}_{conv}$ only [4], then continue training on both $\mathcal{D}_{conv}$ and $\mathcal{D}_{style}$. Furthermore, to reduce overfitting, we applied a data augmentation technique by randomly mask tokens in $s_i$ by a special out-of-vocab token. The masking probability of a token is inversely proportional to its frequency in training data.

### 3.3 Inference

Following (Gao et al., 2019b), we sample in the neighborhood of $z_{S2S}(x)$ by adding a noise $r$ of a given length $|r|$ towards a direction randomly drawn from the uniform distribution.

$$z = z_{\text{S2S}}(x) + r \quad (13)$$

As $r$ depends on $l$, the dimension of $z$, We define a normalized value

$$\rho = |r|/(\sigma\sqrt{l}) \quad (14)$$

As the stylized texts are usually sparse, it is possible to generate non-stylized hypothesis as we vary $\rho$ along some direction. Thus we rank the hypotheses considering both relevance and style intensity.

$$\text{score}(h_i) = (1-\lambda) P(h_i|z_{\text{S2S}}(x)) + \lambda P_{\text{style}}(h_i) \quad (15)$$

[3] More generally, one may use a weighted sum of these terms instead. We set them equally weighted for simplicity

[4] by setting terms $\mathcal{L}_{\text{smooth,style}}$ and $\mathcal{L}_{\text{fuse,style}}$ as zero

where $\lambda = 0.5$ unless otherwise specified, $P(h_i|z_{\text{S2S}}(x))$ estimates the relevancy, and $P_{style}(h_i)$ is the probability of hypothesis $h_i$ being targeted style predicted by pretrained classifiers.

We considered two style classifiers: a "**neural**" based on two stacked GRU (Cho et al., 2014) cells, and a "**ngram**" classifier which is a logistic regressor using ngram (n=1,2,3,4) multi-hot features. Both classifiers are trained using $\{y_i\}$ as negative samples and $\{s_i\}$ as positive samples. $P_{style}(h_i)$ is computed by taking average of the prediction of these two classifiers.

## 4 Experiment Setup

### 4.1 Tasks and datasets

We experiments with two tasks: generating arXiv-like and Holmes-like responses, respectively, using the datasets summarized in Table 1

| Task | Training $\mathcal{D}_{\text{conv}}$ | $\mathcal{D}_{\text{style}}$ | Testing $\mathcal{D}_{\text{test}}$ |
|---|---|---|---|
| arXiv-like | Reddit | arXiv | arXiv-like Reddit |
| Holmes-like | Reddit | Holmes | Holmes-like Reddit |

Table 1: Summary of tasks and datasets

(i) *Reddit* is a conversation dataset constructed from posts and comments on Reddit.com [5] during 2011, consisting of 10M pairs of context and response . (ii) *arXiv* is a non-conversational dataset extracted from articles on arXiv.org [6] from 1998 to 2002, consisting of 1M sentences . (iii) *Holmes* refers to another non-conversational dataset extracted from Sherlock Holmes novel series[7] by Arthur Conan Doyle, with 38k sentences.

$\mathcal{D}_{\text{test}}$ is the test set with stylized reference responses, constructed by filtering the Reddit dataset from year 2013 using the trained neural and ngram classifiers. For each context, there are at least 4 reference responses approximately in the targeted style ($P_{\text{style}} > 0.3$). The style intensity of the context is not filtered.

### 4.2 Human evaluation

We designed the following two tasks.

**Response appropriateness** measurement task presents a context and a set of hypotheses (from the present and baseline systems), and for each hypothesis annotators choose from one of the following options that best fits the quality of the response: ok, marginal, bad (generic or irrelevant), and then map them to numerical score 1, 0.5, and 0, respectively.

**Style classification** task presents a hypothesis and two groups of example sentences, from Reddit and style corpus (Holmes or arXiv). Then crowdsourced annotators judge whether the hypothesis is more similar to the Reddit group, not sure, or more similar to the style corpus group. We then map these to numerical scores 0, 0.5, and 1, respectively.

For all tasks, the hypotheses of different systems of the same set of 500 randomly selected $x_i$ are presented in random order and the identity of the system is invisible to annotators. Each sample is judged by 5 annotators individually.

### 4.3 Automatic evaluation

We measure relevance using multi-reference BLEU Papineni et al. (2002), and diversity by entropy 4-gram (Zhang et al., 2018), and distinct 1,2-gram (Li et al., 2016a).

For style intensity evaluation, besides the **neural** and **ngram** classifier prediction (Section 3.3), we also use simple word-counting (hereafter **count** metric) to minimize model-specific effects. We first construct a training corpus with balanced positive (from $\mathcal{D}_{\text{style}}$) and negative (responses sampled from $\mathcal{D}_{\text{conv}}$) samples. Then, for each word that appears in more than 5 sentences in the training corpus, we compute the average style intensity of sentences that contain this word. The top $k$ words of highest style intensity are chosen as the keywords in this style. For a test corpus, we compute the average ratio of words that are keywords of a style, as its "count" style metric.

Besides the overall style comparisons (Reddit vs. Holmes, and Reddit vs. arXiv), we also crowdsourced three sets of sentences with human labeled levels in three finer styles: how formal, emotional, and technical each sentence is, and build the keyword list for the count metric.

### 4.4 Baselines

We compare the following baseline systems.

[5] using raw data collected by a third party http://files.pushshift.io/reddit/comments/

[6] from KDD 2003 http://www.cs.cornell.edu/projects/kddcup/datasets.html

[7] from https://gutenberg.org

The first category is generative models. (i) *MTask* refers to the vanilla multi-task learning model proposed in (Luan et al., 2017) trained on both $\mathcal{D}_{\text{conv}}$ and $\mathcal{D}_{\text{style}}$. (ii) *S2S+LM* refers to the method proposed by Niu and Bansal (2018)[8], which uses the weighted average of a S2S model, trained on $\mathcal{D}_{\text{conv}}$, and a LM model, trained on $\mathcal{D}_{\text{style}}$, as the token probability distribution at inference time.

The second category draws a training sample as hypothesis. (iii) *Retrieval* refers to a simple retrieval system which returns the sentence from $\mathcal{D}_{\text{style}}$ of highest generation probability by the MTask model. (iv) *Rand* is a system that randomly picks a sentence from $\mathcal{D}_{\text{style}}$. (v) *Human* refers to the system randomly picks one of the multiple reference responses in the given context from $\mathcal{D}_{\text{test}}$.

### 4.5 Model setup

StyleFusion and trainable baselines, MTask and S2S+LM, use two stacked GRU (Cho et al., 2014) cells for encoders and decoders with $l = 1000$. The word embedding is also 1000 dimension, trained from random initialization. The variance of the noise $\epsilon$ is set to $\sigma^2 = 0.1^2$. The state of the top layer of encoder GRU cell is used as $z$. $z$ is used as the initial state of all layers of the decoder. All trainable models are trained with the ADAM method (Kingma and Ba, 2014) with a learning rate of 0.0003. For StyleFusion and MTask, we first train on $\mathcal{D}_{\text{conv}}$ for 2 epochs, and then continue the training on both $\mathcal{D}_{\text{conv}}$ and $\mathcal{D}_{\text{style}}$ until convergence [9]. For all systems except "Rand" and "Retrieval", we use the ranking method Eq 15 to select the top one hypothesis from 100 candidates.

## 5 Results and Analysis

### 5.1 Modulating the style

By leveraging the structure of the shared latent space, we can modulate the style intensity by $\rho$, as illustrated by examples in Table 2. For example, given the context *"Do you want to play a game"*, the hypothesis generated from $\rho = 0$ is *"I do"*, which is non-stylized. While moving towards $z_{\text{AE}}$ of an arXiv-style sentence *"This would be an interesting viewpoint"*, the responses gradually change to "This would be an interesting experience" at $\rho = 1.0$, which remains relevant but is more similar to the target style. Similar trends can be observed when moving in the other direction *"The conclusion depends on the scenario"* and *"This is not a desirable characteristic"*. It also shows that the contents are affected by the direction, a desired property inherited from SpaceFusion models.

| **context** | Do you want to play a game? |
|---|---|
| **towards** | The conclusion depends on the scenario . |
| $\rho = 0.0$ | I do. |
| $\rho = 0.5$ | The answer is yes. |
| $\rho = 1.0$ | The answer depends on the game. |
| **towards** | This would be an interesting viewpoint. |
| $\rho = 0.4$ | This is a good idea. |
| $\rho = 0.9$ | This would be an interesting experience |
| **towards** | This is not a desirable characteristic. |
| $\rho = 0.5$ | I don't play it. |
| $\rho = 1.0$ | This is not a valid question. |

Table 2: Example StyleFusion outputs for arXiv-like response generation task at different distance $\rho$ and direction (towards $z_{\text{AE}}$ of a given sentence)

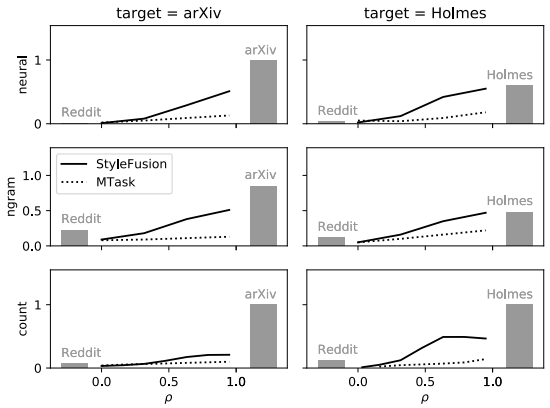


Figure 3: Change of the overall style intensity with $\rho$, as measured by two classifiers "neural" and "ngram" (Section 3.3) and the "count" metric. The "count" metric is normalized by the value of the target style corpus. The barplot shows the desired trend (from Reddit to arXiv or Holmes), and the lines the actual trends.

The relation between style intensity and $\rho$ is further confirmed by automatic measurement. As illustrated in Fig. 3, as $\rho$ increases, responses come to resemble the targeted style within the depicted range. In contrast, the style intensity of MTask outputs rises only slightly as $\rho$ increases.

The increase of overall style intensity is coupled with change in the style's finer granularity, as illustrated in Fig. 4. Compared to Reddit, arXiv is less emotional, and more formal and technical. Consistent with this, StyleFusion outputs exhibit

[8] This method was referred as "Fusion" in (Niu and Bansal, 2018) but to avoid confusing readers with our StyleFusion method, we refer it as "S2S+LM"

[9] approximately one pass of arXiv and 5 passes of Holmes

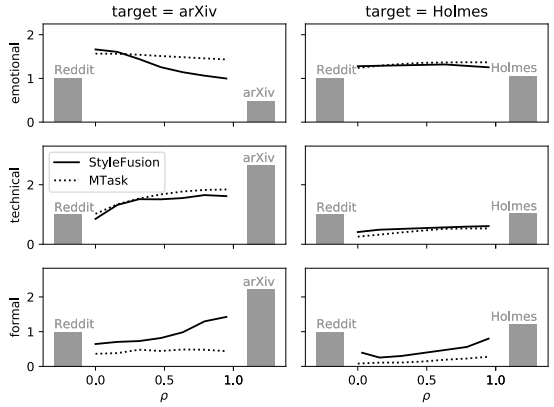


Figure 4: Change of the styles at finer granularity of $\rho$, measured by the "count" metric normalized by the value of Reddit dataset. The barplot shows the desired trend (from Reddit to arXiv or Holmes), and the lines the actual trends.

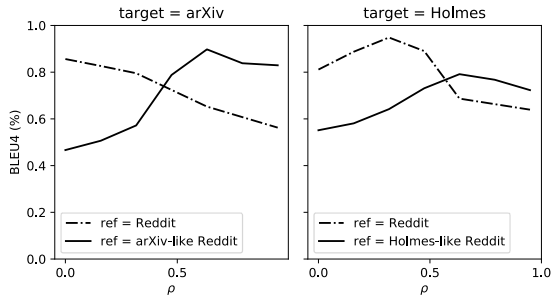


Figure 5: Relevancy of the StyleFusion outputs at different $\rho$ as measured by BLEU4 with references of different styles

less emotion, but become much more technical and formal as $\rho$ increases. MTask, however, tends only to show increased technical level, but fails to be less emotional and more formal, inconsistent with the target style. Where Holmes is the target, the emotional and technical levels do not significantly change compared to Reddit, but Holmes is stylistically more formal. StyleFusion captures these trends, whereas MTask outputs are insufficiently formal, shown in Fig. 4(lowest panel).

We also measured the BLEU4 score at different $\rho$ as shown in Fig. 5. Besides $\mathcal{D}_{\text{test}}$ (stylized references), we also tested on held-out Reddit data (i.e. non-stylized references). We observed that at smaller $\rho$, the StyleFusion outputs are relevant to context and style is more similar to Reddit, indicated by the relatively high BLEU4 computed using non-stylized references. At greater radius, BLEU4 rises when calculated on arXiv-like and Holmes-like references, indicating that the responses generated remain relevant but are closer to the targeted style. Combined with the case at small radius, the implication is that, although the style gradually changes, the responses generated by StyleFusion remain relevant at a relatively wide range of radius.

| | |
|---|---|
| **context** | Okay, but can we host it in the cloud? |
| StyleFusion | It would be an interesting experiment. It is a possibility. |
| S2S | I think it might be a bit of a stretch. |
| MTask | Yes, yes you can. |
| S2S+LM | The relevant question is , as to this , is you referring to the casimir energy program? |
| Retrieval | Even though this issue can be analyzed in more detail, we shall content ourselves with some simple statements. |
| rand | In particular, it involves the combination equation |
| human | To the cloud ! |
| **context** | The universe has no edge. |
| StyleFusion | Does this mean the universe is expanding? |
| S2S | The universe is infinite. |
| MTask | The universe is finite. |
| S2S+LM | The universe can be on the infinite loop. |
| Retrieval | Is there really any difference in the low energy physics? |
| rand | In other words , the wave function in equation. |
| human | So... a sphere? |

Table 3: Example system outputs on arXiv-like response generation task at $\rho = 1$

### 5.2 Fused latent space

As illustrated by the MDS (Borg and Groenen, 2003) visualization in Fig. 6, MTask cannot align different latent spaces, not only those from different model ($z_{\text{AE}}(y)$ and $z_{\text{S2S}}(x)$), but also for those from same model that have different styles ($z_{\text{AE}}(y)$ and $z_{\text{AE}}(s)$). SpaceFusion (Gao et al., 2019b) can align $z_{\text{AE}}(y)$ and $z_{\text{S2S}}(x)$ better, but $z_{\text{AE}}(s)$ forms a separate cluster, indicating that the conversation dataset and style dataset remain unaligned in the latent space. This is because SpaceFusion was not designed to align non-parallel samples. The separation between the conversation dataset and style dataset in latent space, as is the case for MTask and SpaceFusion, makes it difficult for the conversation model to use style knowledge. In contrast, StyleFusion aligns all three latent spaces well as evidenced by Fig. 6.

### 5.3 Human evaluation

Human evaluation results are presented in Table 5. As in the automatic evaluation results, StyleFusion and MTask show the highest appropriate-

| | |
|---|---|
| **context** | First step: go check behind the cabinet and see what's back there. |
| STYLE-FUSION | I would advise against this. |
| S2S | I'm going to try this next time. |
| MTask | I'll be back in a few minutes. |
| S2S+LM | I've got a little time that i had not been engaged yet. |
| Retrieval | i was thinking the same thing. |
| Rand | Holmes shrugged his shoulders. |
| human | Then there would be nothing , and that would be the answer. |
| **context** | that 's a surprisingly large amount of police coming to see a *** teenager. |
| STYLE-FUSION | What is this crime you speak of? |
| S2S | I'm pretty sure that 's the only thing that 's going to happen. |
| MTask | I'm not sure what you mean by that. |
| S2S+LM | I should be on the same business. |
| Retrieval | well, yes. |
| Rand | I shall be back in an hour or two. |
| human | Must have feared what he was packin' |

Table 4: Example system outputs on Holmes-like response generation task at $\rho = 1$

| target | model | appropr-iateness | style intensity | harmonic mean |
|---|---|---|---|---|
| arXiv | STYLEFUSION | **0.17** | 0.26 | **0.20** |
| | MTask | **0.17** | 0.11 | 0.14 |
| | S2S+LM | 0.09 | 0.51 | 0.15 |
| | Retrieval | 0.07 | 0.71 | 0.14 |
| | Rand | 0.04 | **0.96** | 0.07 |
| | Human | 0.51 | 0.28 | 0.36 |
| Holmes | STYLEFUSION | **0.22** | 0.28 | **0.25** |
| | MTask | **0.23** | 0.15 | 0.18 |
| | S2S+LM | 0.03 | 0.55 | 0.05 |
| | Retrieval | 0.14 | 0.30 | 0.19 |
| | Rand | 0.08 | **0.69** | 0.14 |
| | Human | 0.63 | 0.26 | 0.37 |

Table 5: Human evaluation results. The top models (and those models that are not statistically different, except "human") are in **bold**.

ness (not statistically different) apart from the Human system. However STYLEFUSION outputs are much more stylized. Rand, Retrieval and S2S+LM tend to generate stylized but irrelevant responses. To make the overall trends sharper, following (Gao et al., 2019b), we compute the harmonic mean of appropriateness and style intensity, in terms of which STYLEFUSION outperforms all baselines except the Human system. Additional examples of the system outputs and human responses are provided in Table 3 and Table 4

### 5.4 Ablation study and automatic evalution

The automatic evaluation results for arXiv-like and Holmes-like response generation tasks are presented in Table 6. In both instances, STYLEFUSION achieved relatively high BLEU, and showed high style intensity. The Rand baseline has the highest style intensity but lowest relevance. S2S+LM has the comparable style intensity to STYLEFUSION but BLEU is much lower, consistent with the observation made by (Niu and Bansal, 2018). MTask shows significantly less style intensity than STYLEFUSION. Moreover, MTask's diversity, as measured by entropy4 and distinct1,2, is much lower, indicating that outputs of this model tend to be bland. Adding $\mathcal{L}_{\text{conv}}$ regularization, which is SPACEFUSION, increases diversity, relevance and style intensity slightly, consistent with the finding in (Gao et al., 2019b). Style intensity is further boosted by the addition of term $\mathcal{L}_{\text{style}}$. relevancy and diversity are not significantly affected by the addition of $\mathcal{L}_{\text{style}}$.

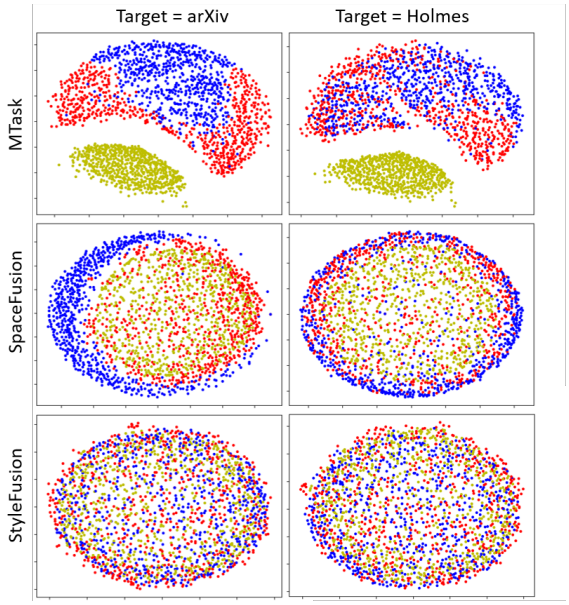


Figure 6: MDS visualization of learned latent spaces. **yellow** dots for $z_{\text{S2S}}(x)$, **blue** dots for $z_{\text{AE}}(s)$ and **red** dots for $z_{\text{AE}}(y)$

## 6 Conclusion

We propose a regularized multi-task learning approach, STYLEFUSION, that bridges conversation models and non-parallel style transfer by structuring a shared latent space. This structure allows the system to generate stylized relevant responses by sampling in the neighborhood of the model prediction, and to continuously control style intensity by modulating the sampling radius. We demonstrate this method in two tasks: generating arXiv-like and Holmes-like conversational responses. Automatic and human evaluation show that, with-

| system | style intensity | | | relevancy | | | | diversity | | |
|---|---|---|---|---|---|---|---|---|---|---|
| sampled ($\in$) or generated | neural | ngram | count | BLEU1 | BLEU2 | BLEU3 | BLEU4 | entropy4 | distinct1 | distinc2 |
| target = arXiv | | | | | | | | | | |
| Rand ($\in \mathcal{D}_{\text{style}}$) | 0.99 | 0.85 | 1.00 | 12.1 | 1.7 | 0.6 | 0.34 | 9.4 | 0.13 | 0.56 |
| Retrieval ($\in \mathcal{D}_{\text{style}}$) | 0.84 | 0.77 | 0.59 | 15.5 | 2.3 | 0.8 | 0.49 | 7.6 | 0.06 | 0.19 |
| Human ($\in \mathcal{D}_{\text{test}}$) | 0.43 | 0.47 | 0.35 | 29.0 | 16.3 | 10.6 | 7.44 | 8.6 | 0.31 | 0.81 |
| S2S+LM | 0.36 | **0.48** | **0.34** | 14.2 | 2.5 | 0.7 | 0.41 | **9.4** | **0.11** | **0.54** |
| MTask | 0.13 | 0.13 | 0.10 | 16.5 | 2.9 | 1.2 | 0.66 | 5.7 | 0.04 | 0.13 |
| +$\mathcal{L}_{\text{conv}}$ (SPACEFUSION) | 0.27 | 0.41 | 0.17 | **18.1** | 3.9 | 1.4 | 0.75 | 6.9 | 0.04 | 0.13 |
| +$\mathcal{L}_{\text{style}}$ (STYLEFUSION) | **0.40** | 0.46 | 0.21 | 17.9 | **4.4** | **1.6** | **0.83** | 7.9 | 0.05 | 0.20 |
| target = Holmes | | | | | | | | | | |
| Rand ($\in \mathcal{D}_{\text{style}}$) | 0.60 | 0.48 | 1.00 | 13.1 | 1.9 | 0.6 | 0.37 | 9.0 | 0.15 | 0.62 |
| Retrieval ($\in \mathcal{D}_{\text{style}}$) | 0.20 | 0.21 | 0.10 | 10.7 | 1.7 | 0.7 | 0.45 | 6.5 | 0.04 | 0.15 |
| Human ($\in \mathcal{D}_{\text{test}}$) | 0.46 | 0.43 | 0.67 | 26.5 | 13.7 | 9.2 | 6.65 | 9.3 | 0.16 | 0.60 |
| S2S+LM | 0.50 | 0.44 | **0.59** | 16.3 | 3.0 | 0.8 | 0.44 | **8.7** | **0.07** | **0.38** |
| MTask | 0.17 | 0.22 | 0.14 | 19.5 | 4.5 | 1.5 | 0.73 | 6.9 | 0.03 | 0.12 |
| +$\mathcal{L}_{\text{conv}}$ (SPACEFUSION) | 0.39 | 0.33 | 0.22 | 18.9 | 4.6 | 1.5 | **0.76** | 7.7 | 0.04 | 0.17 |
| +$\mathcal{L}_{\text{style}}$ (STYLEFUSION) | **0.55** | **0.48** | 0.47 | **20.6** | **5.1** | **1.6** | 0.73 | 7.8 | 0.04 | 0.17 |

Table 6: Performance of each model on automatic measures. The highest score for generative models in each column is in **bold** for each target. the "count" metric is normalized by the value of the targeted style dataset. Note that the unit for BLEU is percentage.

out sacrificing relevance, the system generates responses of the targeted style and outperforms competitive baselines. In future work, we will use this technique to distill information from other non-parallel datasets, such as external informative text (Qin et al., 2019; Galley et al., 2019).

## References


Reina Akama, Kazuaki Inada, Naoya Inoue, Sosuke Kobayashi, and Kentaro Inui. 2017. Generating stylistically consistent dialog responses with transfer learning. In *IJCNLP*, pages 408–412.

Ingwer Borg and P Groenen. 2003. Modern multidimensional scaling: theory and applications. *Journal of Educational Measurement*, 40(3):277–280.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *SSST-8*, pages 103–111.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2018. Style transfer in text: Exploration and evaluation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Michel Galley, Chris Brockett, Xiang Gao, Jianfeng Gao, and Bill Dolan. 2019. Grounded response generation task at dstc7. In *AAAI Dialog System Technology Challenges Workshop*.

Jianfeng Gao, Michel Galley, and Lihong Li. 2019a. Neural approaches to conversational ai. *Foundations and Trends in Information Retrieval*, 13(2-3):127–298.

Xiang Gao, Sungjin Lee, Yizhe Zhang, Chris Brockett, Michel Galley, Jianfeng Gao, and Bill Dolan. 2019b. Jointly optimizing diversity and relevance in neural response generation. *NAACL-HLT 2019*.

Hongyu Gong, Suma Bhat, Lingfei Wu, Jinjun Xiong, and Wen-mei Hwu. 2019. Reinforcement learning based text style transfer without parallel training corpus. *arXiv preprint arXiv:1903.10671*.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *ICML*, pages 1587–1596. JMLR. org.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *NAACL*, pages 110–119.

Jiwei Li, Michel Galley, Chris Brockett, Georgios Spithourakis, Jianfeng Gao, and Bill Dolan. 2016b. A persona-based neural conversation model. In *ACL*, volume 1, pages 994–1003.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*.

Yi Luan, Chris Brockett, Bill Dolan, Jianfeng Gao, and Michel Galley. 2017. Multi-task learning for speaker-role adaptation in neural conversation models. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 605–614.

Yi Luan, Yangfeng Ji, Hannaneh Hajishirzi, and Boyang Li. 2016. Multiplicative representations for unsupervised semantic role induction. In *ACL*.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*.

Kate G Niederhoffer and James W Pennebaker. 2002a. Linguistic style matching in social interaction. *Journal of Language and Social Psychology*, 21(4):337–360.

Kate G Niederhoffer and James W Pennebaker. 2002b. Sharing one's story: On the benefits of writing or talking about emotional experience. *Handbook of positive psychology*.

Tong Niu and Mohit Bansal. 2018. Polite dialogue generation without parallel data. *Transactions of the Association of Computational Linguistics*, 6:373–389.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Lianhui Qin, Michel Galley, Chris Brockett, Xiaodong Liu, Xiang Gao, Bill Dolan, Yejin Choi, and Jianfeng Gao. 2019. Conversing by reading: Contentful neural conversation with on-demand machine reading. *arXiv preprint arXiv:1906.02738*.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *NIPS*, pages 6830–6841.

Vighnesh Leonardo Shiv, Chris Quirk, Anshuman Suri, Xiang Gao, Khuram Shahid, Nithya Govindarajan, Yizhe Zhang, Jianfeng Gao, Michel Galley, Chris Brockett, et al. 2019. Microsoft icecaps: An open-source toolkit for conversation modeling. In *Proceedings of the 57th Conference of the Association for Computational Linguistics: System Demonstrations*, pages 123–128.

I. Sutskever, O. Vinyals, and Q. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Zichao Yang, Zhiting Hu, Chris Dyer, Eric P Xing, and Taylor Berg-Kirkpatrick. 2018. Unsupervised text style transfer using language models as discriminators. In *Advances in Neural Information Processing Systems*, pages 7287–7298.

Yizhe Zhang, Michel Galley, Jianfeng Gao, Zhe Gan, Xiujun Li, Chris Brockett, and Bill Dolan. 2018. Generating informative and diverse conversational responses via adversarial information maximization. In *Advances in Neural Information Processing Systems*, pages 1813–1823.

Yizhe Zhang, Xiang Gao, Sungjin Lee, Chris Brockett, Michel Galley, Jianfeng Gao, and Bill Dolan. 2019. Consistent dialogue generation with self-supervised feature learning. *arXiv preprint arXiv:1903.05759*.

Yizhe Zhang, Dinghan Shen, Guoyin Wang, Zhe Gan, Ricardo Henao, and Lawrence Carin. 2017. Deconvolutional paragraph representation learning. In *Advances in Neural Information Processing Systems*, pages 4169–4179.

Li Zhou, Jianfeng Gao, Di Li, and Heung-Yeung Shum. 2018. The design and implementation of xiaoice, an empathetic social chatbot. *arXiv preprint arXiv:1812.08989*.

# Improving Open-Domain Dialogue Systems via Multi-Turn Incomplete Utterance Restoration

**Zhufeng Pan**[1,2*] **Kun Bai**[2*] **Yan Wang**[2 †] **Lianqiang Zhou**[2] **Xiaojiang Liu**[2]
[1]University of California, Los Angeles [2]Tencent AI Lab
panzhufeng@cs.ucla.edu
{bookerbai, brandenwang, tomcatzhou, kieranliu}@tencent.com

## Abstract

In multi-turn dialogue, utterances do not always take the full form of sentences. These incomplete utterances will greatly reduce the performance of open-domain dialogue systems. Restoring more incomplete utterances from context could potentially help the systems generate more relevant responses. To facilitate the study of incomplete utterance restoration for open-domain dialogue systems, a large-scale multi-turn dataset *Restoration-200K*[1] is collected and manually labeled with the explicit relation between an utterance and its context. We also propose a "pick-and-combine" model to restore the incomplete utterance from its context. Experimental results demonstrate that the annotated dataset and the proposed approach significantly boost the response quality of both single-turn and multi-turn dialogue systems.

## 1 Introduction

Dialogue systems have attracted increasing attention due to the promising potentials on applications like virtual assistants or customer support systems (Hauswald et al., 2015; Poulami Debnath, 2018). However, studies (Carbonell, 1983) show that users of dialogue systems tend to use succinct language which often omits entities or concepts made in previous utterances. (also known as non-sentential utterances, (Fernández et al., 2005, 2007)). To make appropriate responses, dialogue systems must be equipped with the ability to understand these incomplete utterances.

Take Example 1 in Table 1 for instance, contents in parentheses are information omitted in the utterance. Humans are capable of comprehending those incomplete utterances based on previous utterances. For example, $A_3$ means what kind of *dessert* matches B's taste, instead of what kind of *shop* B likes. Failing to understand this utterance would be a disaster for dialogue systems to generate a relevant and coherent response. According to our survey (details in Table 2), in about 60% conversations, fully comprehending current utterance depends on previous context. We will refer conversation history ($A_1$ to $B_2$ in above example) as *previous utterances*, the utterance to be restored ($A_3$) as *original utterance*, and the complete form of $A_3$ as *restored utterance*.

Studies show that restoring the incomplete questions could help question-answering systems better understand users' intention (Raghu et al., 2015; Kumar and Joshi, 2017). It inspires us to improve the performance of open-domain dialogue systems via incomplete utterance restoration. However, most existing multi-turn dialogue datasets only provide sets of utterances, without any information about relations between utterances. In other words, they lack the necessary supervisions to restore the incomplete utterance.

To make dialogue systems better understand incomplete utterances, we collect a multi-turn conversation dataset from internet communities, and each of the conversations contains at least six utterances. Then we hire an annotation team to (1) label whether an utterance is related to its context or not, and (2) restore an incomplete utterance to a complete and context-free form based on its context. Finally, we get a high-quality and large-scale dataset with 200K annotated conversations. Such a dataset offers a new way of modelling utterance relations and improving the context-understanding ability of dialogue systems. We hope it would benefit the research of context understanding for multi-turn dialogue systems in the future.

With the annotated dataset above, we first attempt to restore the incomplete utterance using

[*]Both authors contributed equally to the work. The work was conducted when Zhufeng Pan was interning at Tencent AI Lab.

[†]Corresponding author

[1]The dataset is available at: https://ai.tencent.com/ailab/nlp/dialogue/datasets/



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1824–1833, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| | Example 1 | Example 2 | Example 3 |
|---|---|---|---|
| $A_1$ | 我能在巴黎哪个地方学做甜品？<br>Where can I learn to make dessert in Paris? | 今天买了一堆**桌游**有爱**玩**的可以一起<br>I bought a bunch of **board game**. Welcome anybody who also likes to **play** it | 我们一起过个情人节吧<br>Shall we spend Valentine's Day together |
| $B_1$ | 为什么(你想学做甜品)啊？<br>Why (do you want to learn to make dessert)? | 我比较喜欢卡卡颂和现代艺术<br>I like Carcassonne and Modern Art | 头像都一样在一起吧。<br>Let's date since we have the same avatar |
| $A_2$ | 因为我想在(巴黎)这儿开个甜品店<br>Because I want to open a dessert shop here (in Paris) | 听说过不过没买<br>Heard of it. But I haven't bought it | 在一起不错的选择<br>Dating is a good choice |
| $B_2$ | (在巴黎开甜品店)不错啊，我很喜欢**甜点**！<br>(Opening a dessert shop in Paris) Sounds great, I love dessert! | 我有<br>No problem, I have | 赞<br>Cool |
| $A_3$ | 你最喜欢哪一种?<br>Which kind matches your taste most? | 一起啊<br>Let's do it together | 人呢<br>Where are you? |
| Label | 1 | 1 | 0 |
| Reference | 你最喜欢哪一种**甜品**?<br>Which kind **of dessert** matches your taste most? | 一起**玩桌游**啊<br>Let's **play board game** together | 人呢<br>Where are you? |

Table 1: Examples from the dataset. If annotators judge the fifth utterance ($A_3$) omits concepts or entities made in previous utterances like Example 1 and 2, they rewrite the utterance as ground truth reference.

two vanilla models, Sequence-to-Sequence model (Seq2Seq) and Pointer Generative Network. Then a cascaded pick-and-combine model is further proposed to first "Pick" omitted words from context and then "Combine" them with the incomplete utterance. To better evaluate the restoration performance, an evaluation metric is also designed. In the experiment, both automatic metrics and human evaluation show that the proposed approach could achieve promising results and significantly improve the response relevance of both single-turn dialogue systems and multi-turn dialogue systems.

The proposed approach enables single-turn dialogue systems with the capability to comprehend the dialogue context. It also facilitates multi-turn dialogue systems to model the relation between the query (original utterance) and context (previous utterances) explicitly in a supervised manner, in contrast to modelling the relation implicitly and without extra guidance (Serban et al., 2016; Wu et al., 2017).

Our contributions are summarized as below:

1) A large-scale Chinese dataset with 200K multi-turn conversations are collected and manually labeled with the explicit relations between an utterance and its context.

2) A cascaded pick-and-combine model is proposed, which achieves promising results on both automatic metrics and human evaluation.

3) Experimental results demonstrate that the incomplete utterance restoration model could be complementary to existing dialogue systems and is conducive for improving response quality.

In the remaining part, we first describe the collected dataset in Section 2. In Section 3, several models and a new metric are presented for incomplete utterance restoration. Section 4 shows experimental results on both automatic metrics and human evaluation for the proposed method. In Section 5 the proposed model is applied to dialogue systems to evaluate its effectiveness. Section 6 introduces related work. We conclude our work and discuss future directions in Section 7.

## 2 Restoration-200K

### 2.1 Data Collection

We collect open-domain dialogues from Douban group[2], a well-known Chinese online community which is a common data source for dialogue systems (Wu et al., 2017). To determine if one utterance is complete or not, we take four utterances before it as the conversation context. Crawled conversations with less than six (extra one to assist annotation) utterances are filtered out. For conversations with more than six utterances, only the first six are reserved. We construct a conversation by the identification of reply tags in comments under each post.

### 2.2 Data Annotation

To ensure the annotation quality, five professional data annotators are hired to annotate the dataset instead of using crowd-sourcing platforms like MTurk. It took six months to finish the annotation. As shown in Figure 1, annotators first discern whether the original utterance (the fifth utterance in conversation) omits concepts or entities made in previous utterances. Since the boundary

[2]https://www.douban.com/group

| | train | val | test |
|---|---|---|---|
| # conversations | 194k | 5k | 5k |
| Incomplete ratio (%) | 60.1 | 59.4 | 58.8 |
| Vocabulary size | 80k | 15k | 15k |
| Avg. context length | 25.9 | 25.8 | 25.7 |
| Avg. utterance length | 8.62 | 8.53 | 8.60 |
| Avg. reference length | 12.4 | 12.3 | 12.4 |

Table 2: Statistics of Restoration-200K. The incomplete ratio refers to the ratio of conversations that contains the incomplete utterance. Vocabulary size is counted after Chinese word segmentation and the average length is counted on character level.

between yes and no is vague under certain circumstances, annotators are allowed to skip and discard the instance if it is hard to make a decision. The sixth utterance is also provided to assist annotators in comprehending the conversation. When restoration is needed, they rewrite the original utterance to a context-free restored utterance which contains all necessary information for utterance understanding, as shown in the last row of Table 1.

To reduce the diversity of rewritten sentences, annotators are instructed to use words from previous utterances wherever possible. A small commonly used word list is provided, and annotators could use those words to ensure the fluency of rewritten sentences. In other words, all words in the restored utterance come from previous and original utterances or the extra word list. Our survey shows only about 4.8% of conversations that need to be restored in the dataset do not satisfy such condition. All of them are discarded as well.

### 2.3 Data Statistics

The statistics of train, validation and test set are shown in Table 2. Kumar and Joshi (2016) collect 6K incomplete questions for QA systems, where each utterance only consists of 3.52 words on average[3]. In comparison, *Restoration-200K* has a longer utterance length, which indicates covering a broader range of topics and more informative contents.

## 3 Methodology

In this section, we try to tackle the incomplete utterance restoration problem by using the vanilla Seq2Seq model with attention and pointer generative network. Since these two models are easily dominated by a simple copy mechanism that directly regenerates the original utterance as the restored utterance, we further propose a cascaded pick-and-combine (PAC) model to restore the original utterance.

[3]The average sentence length is estimated based on the released test set.

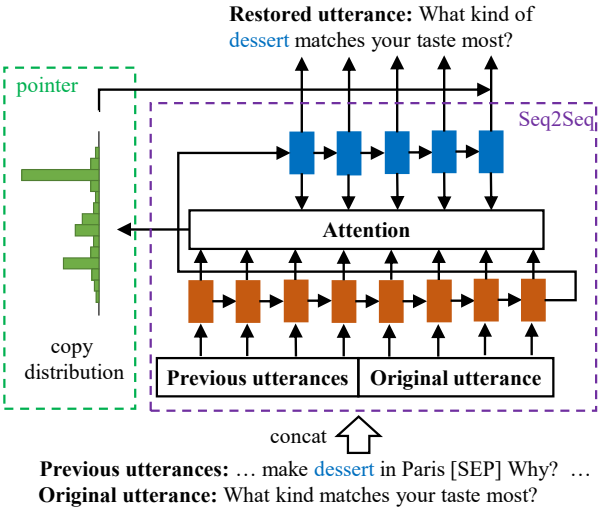


Figure 1: The architecture of Seq2Seq model and pointer generative network.

### 3.1 Vanilla Models

**Seq2Seq Model with Attention:** the Seq2Seq framework has been widely used in sequence generation tasks. The encoder encodes the input sequence into a vector representation $c$ and the decoder generates the target sequence based on representation vector $c$, and all the previous words. As shown in Figure 1, to restore the incomplete utterance, we concatenate previous utterances $C$ and the original utterance $X$ as the input sequence. By inserting a special token $[sep]$ to separate $C$ and $X$, the input sequence will be: $[c_1, \ldots, c_{t1}, [sep], x_1, \ldots x_{t2}]$.

**Pointer Generative Network:** as analyzed in Section 2.2, the incomplete utterance restoration problem has a unique characteristic: most generated words come from previous utterances or original utterance. To exploit this constraint, we propose using the pointer generative network to directly copy words from the input sequence. It has the capability of copying words from source sequence by directly taking the attention score $a_k$ as prediction probability, as shown in the green part of Figure 1. The generation distribution $P_{gen}$ can be calculated as: $P_{gen} = f(h_t^*, s_t, y_t)$, $p_{gen} = g(h_t^*, s_t, y_t)$. And the final vocabulary distribution

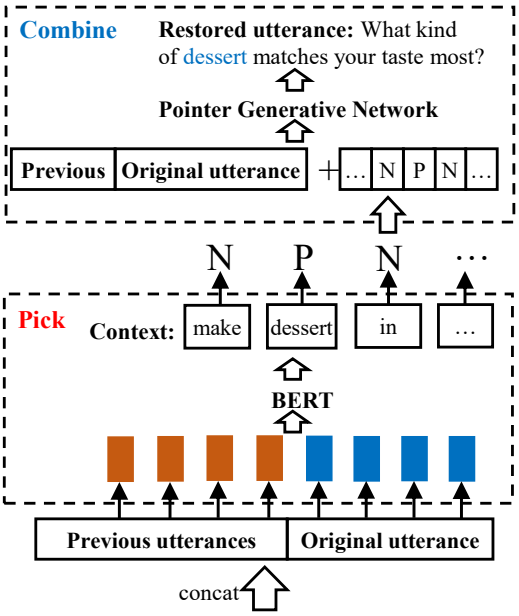


**Previous utterances:** … make dessert in Paris [SEP] Why?…

**Original utterance:** What kind matches your taste most?

Figure 2: The pick stage predicts which words in previous utterances are omitted by the original utterance. P represents the label of positive, namely the omitted words. In combine stage, the picking result is appended to original conversation as extra guidance for the complete utterance sequence generation.

$P_{vocab}$ is defined as:

$$P_{vocab} = p_{gen} * P_{gen} + (1 - p_{gen}) * P_{copy}$$

### 3.2 Pick-and-Combine Model

For most utterances to be restored in the corpus, the reference only differs from the original utterance in few words. Our study shows on average only 17.7% words in previous utterances overlap with the restored utterance, while 100% words in the original utterance are included in the restored utterance. Such *unbalanced* probability makes models mentioned above tend to simply regenerate the original utterance to maximize the conditional probability of the generated sentence during the beam-search process. In other words, the Seq2Seq model and pointer generative network tend to regenerate the original utterance and cannot effectively restore the original utterance. The third conversation in Table 5 is a typical example.

To mitigate this problem, we propose to decompose the incomplete utterance restoration task into a cascaded process. The first stage is the *Pick* process that identifies omitted words in previous utterances. The second is the *Combine* stage that restores the original utterance based on the identified omitted words.

**Pick:** inspired by recent advances in transfer learning for language representation, we fine-tune the pre-trained deep bidirectional transformers for language understanding model (BERT) (Devlin et al., 2019) as a classifier to select omitted words from previous utterances. Specifically, instead of discrete classifications, we first formulate this word selection problem as a sequence tagging problem. Each word in previous utterances will be determined to be positive $P$ (should be identified as an omitted word) or negative $N$ (not an omitted word).

**Combine:** the combine stage is straightforward. The selected omitted words are appended to the input sequence as extra guidance. And the two sequences are taken as input of the pointer generative network mentioned in Section 3.1. We also tried to fine-tune the BERT to directly generate the restored utterance, but the result is far from satisfactory. Therefore, we directly append selected words to the original input sequence as extra guidance for generating restored sentences.

### 3.3 Task Evaluation

BLEU (Papineni et al., 2002) and ROUGE (Lin, 2004) are commonly used metrics in machine translation and summarization. However, for the task of incomplete utterance restoration, these metrics do not differentiate words in the original utterance from those in previous utterances, and each gram is equally important. The statistics in Table 2 show that most words in the reference overlap with words in the original utterance. Thus, a simple copy mechanism which regenerates the original utterance as restored utterance would achieve a high score on these metrics.

To alleviate this issue, we propose the *restoration score* to evaluate the performance of incomplete utterance restoration model. This metric focuses on n-grams that contain at least one restored word, excluding other n-grams. Specifically, the n-gram restoration precision, recall, and F-score can be calculated as:

$$p_n = \frac{|\{\text{restored n-grams}\} \cap \{\text{n-grams in ref}\}|}{|\{\text{restored n-grams}\}|}$$

$$r_n = \frac{|\{\text{restored n-grams}\} \cap \{\text{n-grams in ref}\}|}{|\{\text{n-grams in ref}\}|}$$

$$f_n = 2 \cdot \frac{p_n \cdot r_n}{p_n + r_n}$$

| Model | $p_1$ | $r_1$ | $f_1$ | $p_2$ | $r_2$ | $f_2$ | $p_3$ | $r_3$ | $f_3$ | $B_1$ | $B_2$ | $R_1$ | $R_2$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Syntactic | 67.4 | 37.2 | 47.9 | 53.9 | 30.3 | 38.8 | 45.3 | 25.3 | 32.5 | 84.1 | 81.2 | 89.3 | 80.6 |
| Seq2Seq | 65.5 | 40.8 | 50.3 | 52.2 | 32.6 | 40.1 | 43.6 | 27.0 | 33.4 | 84.9 | 81.7 | 88.8 | 80.3 |
| Pointer | 66.6 | 40.4 | 50.3 | 54.0 | 33.1 | 41.1 | **45.9** | 28.1 | 34.9 | 84.7 | 81.7 | 89.0 | 80.9 |
| PAC | **70.5** | **58.1** | **63.7** | **55.4** | **45.1** | **49.7** | 45.2 | **36.6** | **40.4** | **89.9** | **86.3** | **91.6** | **82.8** |

Table 3: $p$, $r$, $f$ here represent the restoration precision, recall and F-score that we propose in Section 3.3. $B_n$ represents n-gram BLEU score and $R_n$ represents n-gram ROUGE score.

where "restored n-grams" refer to the n-grams in restored utterance that contain at least one restored words, and "n-grams in ref" refer to the n-grams in reference that contain at least one restored words, $|X|$ refers to the number of elements in set $X$.

## 4 Experiment

### 4.1 Compared methods

We compare the performance of the following methods on the collected dataset:

- **Syntactic:** model proposed by Kumar (Kumar and Joshi, 2016). Specifically, they replace each out-of-vocabulary word in the corpus with a numbered unknown token. The number is based on its relative position in the conversation. The model itself is a Seq2Seq model with attention.
- **Seq2Seq:** the Seq2Seq model with attention introduced in Section 3.1.
- **Pointer:** the pointer generative network introduced in Section 3.1.
- **PAC:** the pick-and-combine model introduced in Section 3.2.

We adopt one layer unidirectional LSTM as the encoder and decoder of each model. During training, the vocabulary size is set to 10K. The size of each mini-batch is 64. Parameters are updated by Adam algorithm (P and Ba, 2014) with the betas set to 0.9 and 0.999 and the eps set to 1e-8. The learning rate is 0.25 and the clipping threshold of gradients is 0.1. The dropout rate is set to 0.5. The word embedding size, encoder hidden size and decoder hidden size are all set to 512. During the inference stage, the checkpoint with smallest validation loss is chosen and the beam-search size is set to 5 for all methods. For the pick stage in the proposed approach, we use a BERT model trained on 200G high-quality news data from Tencent AI Lab.

| Method | Quality | Fluency |
|---|---|---|
| Syntactic | 2.25 | **2.58** |
| Seq2Seq | 2.45 | 2.37 |
| Pointer | 2.53 | 2.50 |
| PAC | **2.89** | 2.51 |

Table 4: Human evaluation on the restoration quality and language fluency. The quality score adopts a 4-point scale, and fluency score adopts a 3-point scale. A higher score is better for both.

### 4.2 Evaluation metrics

The n-gram restoration score proposed in Section 3.3 is adopted as the automatic evaluation metrics. Then human evaluation is further conducted. Specifically, human annotators are instructed to evaluate the quality and fluency of restored utterances. The quality score evaluates if the restored utterances can be better understood without context than the original utterance. A restored utterance should be scored 4 if it can be perfectly understood without previous utterances, 3 if it is not perfect but still better than the original utterance, 2 if it hasn't been restored or it cannot help better the understanding, and 1 if it is worse than the original utterance. The fluency score is used to evaluate the fluency of a restored utterance. A 3-point scale is adopted and higher is better. We randomly select three hundred examples from the test set for human evaluation. Five professional annotators who have at least one year's experience on text annotation participate in the experiment.

### 4.3 Results

#### Automatic Metrics

Scores of each method on the automatic metric are shown on Table 3. PAC model achieves the best performance on the restoration precision, recall, and F-score from 1-gram to 3-grams. The restoration recall reflects how often does the model choose to restore the original utterance. As discussed in Section 3.2, Seq2Seq model and pointer

| | Example 1 | Example 2 | Example 3 |
|---|---|---|---|
| $A_1$ | 我这个样子也来相个亲喃<br>I am here for a blind date | 晚上了在上**夜班**<br>It's late. I am working in **night shift** | 谁给你拍的<br>Who took it for you? |
| $B_1$ | **会煮饭**加100分<br>**Being able to cook** will add 100 points for you | 早安世界<br>Morning, the world | 朋友抓拍<br>My friend took the snapshot |
| $A_2$ | 这个技能竟然100分<br>I can't believe this skill is worth 100 points | 早<br>Morning | 抓拍的够写意给了别人脑补的空间<br>The snapshot is so atmospheric that people could free their imagination |
| $B_2$ | 我喜欢**这个技能**<br>I like **the skill** | 乃是要通宵么<br>Are you going to work overnight? | **要的就是这个效果**<br>**That is what I want** |
| $A_3$ | 我也喜欢希望能做好<br>I also like it and hope I could do well | 上到8点钟<br>Until 8 a.m. | 你确定吗我怎么觉得其实是没拍好<br>Are you sure? Why do I feel it was just a mistake |
| Reference | 我也喜欢**会煮饭这个技能**希望能做好<br>I also like **the cooking skill** and hope I could do well | 上**夜班**到8点钟<br>**The night shift** is until 8 a.m. | 你确定**要的就是这种效果**吗我怎么觉得其实是没拍好<br>Are you sure **that is what you want**? Why do I feel it was just a mistake |
| PAC | 我也喜欢**煮饭这个技能**希望能做好<br>I also like **the cooking skill** and hope I could do well | **夜班通宵**上到8点钟<br>**The night shift** requires **working overnight** until 8 a.m. | 你确定**要的就是这种效果**吗我怎么觉得其实是没拍好<br>Are you sure **that is what you want**? Why do I feel it was just a mistake |
| Pointer | 我也喜欢**这个技能**希望能做好<br>I also like **the skill** and hope I could do well | 上到8点钟**通宵**<br>**Working overnight** until 8 a.m. | 你确定吗我怎么觉得其实是没拍好<br>Are you sure? Why do I feel it was just a mistake |
| Syntactic | 我也喜欢**这个技能100分**希望能做好<br>I also like **the skill 100 points** and hope I could do well | 上到8点钟<br>Until 8 a.m. | 你确定吗我怎么觉得其实是没拍好<br>Are you sure? Why do I feel it was just a mistake |
| Seq2Seq | 我也喜欢希望能做好<br>I also like it and hope I could do well | **通宵**上到8点钟<br>**Working overnight** until 8 a.m. | 你确定吗我怎么觉得其实是没拍好<br>Are you sure? Why do I feel it was just a mistake |

Table 5: Examples for incomplete utterance restoration. $A_1$ to $B_2$ are previous utterances and $A_3$ is the original utterance. The proposed PAC model could accurately identify pertinent words even those words appear at the beginning of the conversation like Example 2.

generative network tend to directly copy the original utterance, instead of taking risks to choose words from previous utterances. Restoration recall scores of these models can prove this claim. PAC model is also higher than other models in restoration precision. It demonstrates that the words selected by the PAC model are more accurate than that of other models.

We also report the BLEU and ROUGE score of each model. PAC model also achieves the highest scores in these metrics. However, scores of other models are comparatively high because these two metrics take each gram in the reference as equally important. It illustrates the necessity of the proposed evaluation metrics.

### Human Evaluation

Results of the human evaluation are shown in Table 4. The proposed method has the highest quality score among all methods. We find the model with the higher score in human evaluation also has a higher restoration score. This tendency demonstrates that the proposed metric makes sense and is practical in this task. As for the fluency, all models except the Seq2Seq model achieve a similar score. The reason why the score of the PAC model is slightly lower than the syntactic model is that the fluency score follows a principle that the more you do, the more mistakes you make. Fluency of original utterances is high, so models with lower recall may get higher fluency.

### 4.4 Case study

Table 5 shows some examples of the incomplete utterance restoration among different models. At a closer look to the restored utterances, we find PAC model is capable of understanding the conversation context and correctly identify omitted concepts from multiple possible choices. In Example 1, none except PAC model can successfully restore the word "cooking" from previous utterances. In Example 2, most models can only extract the word "stay up" from previous utterances, and they fail to generate a fluent utterance. In contrast, the PAC model picks the correct words from previous utterances and generates a more fluent utterance. As for Example 3, only the PAC model can correctly restore the original utterance. This scenario also supports our analysis at the beginning of section 3.2 that vanilla models have a tendency to regenerate the original utterance.

We conclude an interesting phenomenon from this case study: only the PAC model owns the capability to extract omitted words from very early context. Other models merely choose words from the latest utterance exactly before the original one, namely the $B_2$ in Table 5. This great context comprehension ability is benefit from the pick process that identifies words from previous utterances.

## 5 Improving Existing Dialogue Systems

The main motivation for incomplete utterance restoration is to facilitate dialogue systems better

understanding conversation context and generate better responses. Thus, we conduct another experiment to evaluate if the response quality of single-turn and multi-turn dialogue systems improves after incomplete utterance restoration.

### 5.1 Settings

We applied the proposed PAC model to a single-turn and a multi-turn response generation model to evaluate its effectiveness on dialogue systems. Details are shown as follows:

- **MMI:** a state-of-the-art single-turn response generation model (Li et al., 2016). It adopts the maximum mutual information (MMI) as the objective function to prevent generating general responses. Note MMI model does not have access to the dialogue context. It only takes the original utterance as the input.

- **SMN:** the sequential matching network (SMN) is a state-of-the-art multi-turn reranking modeling proposed by Wu et al. (2017). The query expanded with keywords is used to retrieve top 100 response candidates. When reranking, SMN first matches a response with each utterance and distills matching information into a vector. The vectors are then accumulated through a recurrent neural network (RNN). The hidden states of the RNN is used to calculate the final matching score. In our experiments, SMN is applied to rerank candidates by taking the original utterance and conversation context as model inputs. The candidate with the highest matching score is chosen as the response.

- **MMI after Restoration:** The MMI model takes the restored utterance as the input.

- **SMN after Restoration:** The SMN model takes the restored utterance and conversation context as inputs.

Three hundred randomly selected conversations from the test set are selected and restored by all models. Five annotators are asked to evaluate whether responses to restored utterances are more relevant and appropriate than those of the original utterances. There are three choices for the annotators: better, similar or worse. If the response gets more appropriate *or* more relevant to the context, it will be judged as "better". If the response quality does not change much after the restoration, it will be regarded as "similar". Otherwise, it will be judged as "worse". Utterances that are not restored are judged as "Not Restored" (NR).

| MMI | Better | Similar | Worse | NR |
|---|---|---|---|---|
| Syntactic | 14.97 | 12.16 | **2.81** | 70.06 |
| Seq2Seq | 19.45 | 15.95 | 5.32 | 59.28 |
| Pointer | 21.25 | 16.51 | 5.35 | 56.89 |
| PAC | **28.82** | 21.95 | 6.12 | **43.11** |

Table 6: Human evaluation on response quality from the single-turn dialogue system after restoration. All numbers are in percentage (%), and NR represents *Not Restored*. Both are same for Table 7.

| SMN | Better | Similar | Worse | NR |
|---|---|---|---|---|
| Syntactic | 13.17 | 10.18 | **6.59** | 70.06 |
| Seq2Seq | 13.77 | 18.56 | 8.38 | 59.28 |
| Pointer | 16.17 | 16.76 | 10.18 | 56.89 |
| PAC | **27.74** | 17.37 | 11.78 | **43.11** |

Table 7: Human evaluation on response quality from the multi-turn dialogue system after restoration.

Note that we also train a generative multi-turn response generation model (Serban et al., 2016) on the collected dataset. But generated responses are not clear and coherent enough to understand. As a result, no human study is further conducted for such approach.

### 5.2 Results

Evaluation results of single-turn and multi-turn dialogue systems are shown in Table 6 and Table 7, respectively. Both systems get better responses after the restoration process, and PAC model is the most effective one. The PAC model achieves the lowest NR, which means it can restore more utterances than other models. Meanwhile, for single-turn dialogue systems, more than $50\%$ queries get better responses when they are restored, and only $10\%$ get worse responses. For multi-turn dialogue systems, although the SMN already has the ability to understand conversation context, the restoration still improves the response quality of nearly half of quires, and only about $20\%$ responses get worse. The results prove that the proposed dataset and approach significantly improve the response quality.

| | Example 1 | Example 2 | Example 3 |
|---|---|---|---|
| $A_1$ | 下午去哪儿逛逛我的小单反<br>Where shall I go with my DSLR camera this afternoon | 明天就直奔**西安**来了<br>Heading for **Xi'an** tomorrow | 这是啤酒节啊<br>What a beer festival |
| $B_1$ | 火车站<br>Rail station | 最近雨后会暴热 西安的定律<br>It''l become very hot after the rain. It''s a rule in Xi'an. | 只有我一个人关注啤酒不错的吗<br>Am I the only one who finds the beer seems to be great? |
| $A_2$ | 不去那人太多了<br>No, there are too many people | 要不要这么悲催的<br>How miserable it is | 你也可以去的<br>You can go as well |
| $B_2$ | 哈哈**拍人头**<br>Hah, you can **take photos of people** | 肿么了你还没到<br>What, you haven't arrived yet? | **不喝酒**<br>**No drinking** |
| $A_3$ | (拍人头)不行不行镜头达不到<br>No, the lens is not good enough (for taking photos of people) | 明早到(西安)呢现在在家<br>Will arrive (in Xi'an) tomorrow. Now I am home | (不喝酒)你可以去吃东西<br>(If no drinking,) You can go and eat something |
| MMI | 你达不到？<br>Are you not good enough? | 你在哪上班呢?<br>Where do you work? | *没东西吃<br>*There is nothing to eat |
| MMI+PAC | *那就别拍了<br>*Then try to take photos of something else | *你在西安干嘛呢?<br>*What are you doing in Xi'an? | 我不喝酒的<br>I do not drink |
| SMN | 不行呗<br>Just cannot do it | 你现在好好的啊<br>You look cool now | *啥东西吃?<br>*What to eat? |
| SMN+PAC | *不行也得行<br>*Yes, you can do it | *带你高校一日游外加兴庆公园赏雪哈哈<br>*I will take you to enjoy the one day trip to colleges and the beauty of snow in Park of Xingqing Palace | 不喝酒那喝什么?<br>What to drink if not the beer? |

Table 8: A comparison between taking original and restored utterance as the input query to a response generation model. In $A_3$, words in parentheses are restored by PAC model. Responses annotated with * are better.

Some examples are shown in Table 8. Example 1 and 2 are cases where the response quality improves after incomplete utterance restoration. One interesting response is the one from *SMN+PAC* in Example 2. Note that the Park of Xingqing Palace is a famous historic relic in Xi'an. The dialogue system successfully generates a much more concrete and relevant response after restoring the city name *Xi'an*. There are also cases where responses become worse even if the model correctly restores omitted words, as shown in Example 3. One possible explanation is in this case the restored part is less important than the original utterance. This motivates us to find more strategies to alleviate this issue in future studies, such as training the restoration model and response generation model in an end-to-end manner or deploying a ranking model to choose the better response.

# 6 Related Work

## 6.1 Non-sentential Utterance Resolution

In addition to a further corpus and taxonomy study, Fernández et al. (2007) design a series of linguistic features to determine the NSU class. Dragone (2015); Dragone and Lison (2016) extend these features for NSU classification. Raghu et al. (2015) propose generating NSU resolution from templates. Poulami Debnath (2018) design a set of rules to classify and restore NSUs for customer support systems. All methods above heavily rely on the syntactic structure or frequent patterns observed empirically, which may not scale well for unseen domains. Kumar and Joshi (2016) approach NSU resolution as a Seq2Seq learning problem. One type of NSU that draws extra attention is ellipsis (Merchant, 2016; Kenyon-Dean et al., 2016; McShane and Babkin, 2016).

## 6.2 Multi-turn Dialogue Systems

Many efficient approaches have been proposed for developing intelligent dialogue systems (He et al., 2017; Shang et al., 2018; Tian et al., 2019; Cai et al., 2019). For multi-turn dialogue systems, Serban et al. (2016) and Xing et al. (2018) adopt hierarchical neural networks to model context. Tian et al. (2017) conduct an extensive comparison among existing methods and found context information is conducive for neural networks to generate longer, more meaningful and diverse replies. Wu et al. (2017) and Zhang et al. (2018) utilize the context information via a matching network and rerank the retrieved responses. Zhou et al. (2018) investigate matching a response with its multi-turn context using dependency information based entirely on attention. Most of these studies model the dialogue context information as an extra input in the response generation or response matching process. Our method, on the other hand, tries to understand the context by restoring the incomplete utterance to a context-free and complete form. It can be complementary to above studies.

# 7 Conclusion

In this paper, we propose to facilitate the comprehension of conversation context by restoring incomplete utterances. A large-scale dataset Restoration-200K, which consists of multi-turn

conversations for open-domain dialogue systems, is collected and manually annotated. Based on this dataset, the proposed pick-and-combine method achieves promising results on the incomplete utterance restoration task. Experimental results also demonstrate that the annotated dataset and the proposed approach significantly improve response quality for existing dialogue systems.

## References


Deng Cai, Yan Wang, Wei Bi, Zhaopeng Tu, Xiaojiang Liu, Wai Lam, and Shuming Shi. 2019. Skeleton-to-response: Dialogue generation guided by retrieval memory. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1219–1228.

Jaime G Carbonell. 1983. Discourse pragmatics and ellipsis resolution in task-oriented natural language interfaces. In *Proceedings of the 21st annual meeting on Association for Computational Linguistics*, pages 164–168. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Paolo Dragone. 2015. Non-sentential utterances in dialogue: Experiments in classification and interpretation. Master's thesis, Sapienza University of Rome.

Paolo Dragone and Pierre Lison. 2016. Classification and resolution of non-sentential utterances in dialogue. *Italian Journal of Computational Linguistics*, 2(1):45–62.

Raquel Fernández, Jonathan Ginzburg, and Shalom Lappin. 2005. Using machine learning for non-sentential utterance classification. In *6th SIGdial Workshop on Discourse and Dialogue*.

Raquel Fernández, Jonathan Ginzburg, and Shalom Lappin. 2007. Classifying non-sentential utterances in dialogue: A machine learning approach. *Computational Linguistics*, 33(3):397–427.

Johann Hauswald, Michael A Laurenzano, Yunqi Zhang, Cheng Li, Austin Rovinski, Arjun Khurana, Ronald G Dreslinski, Trevor Mudge, Vinicius Petrucci, Lingjia Tang, et al. 2015. Sirius: An open end-to-end voice and vision personal assistant and its implications for future warehouse scale computers. In *ACM SIGPLAN Notices*, volume 50, pages 223–238. ACM.

He He, Anusha Balakrishnan, Mihail Eric, and Percy Liang. 2017. Learning symmetric collaborative dialogue agents with dynamic knowledge graph embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1766–1776.

Kian Kenyon-Dean, Jackie Chi Kit Cheung, and Doina Precup. 2016. Verb phrase ellipsis resolution using discriminative and margin-infused algorithms. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1734–1743.

Vineet Kumar and Sachindra Joshi. 2016. Non-sentential question resolution using sequence to sequence learning. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2022–2031.

Vineet Kumar and Sachindra Joshi. 2017. Incomplete follow-up question resolution using retrieval based sequence to sequence learning. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 705–714. ACM.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Marjorie McShane and Petr Babkin. 2016. Detection and resolution of verb phrase ellipsis. *LiLT (Linguistic Issues in Language Technology)*, 13.

Jason Merchant. 2016. Ellipsis: A survey of analytical approaches. *Handbook of ellipsis. Oxford: Oxford University Press. Ms. University of Chicago*.

Kingma Diederik P and Jimmy Ba. 2014. Adam: A method for stochastic optimization.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Harshawardhan M. Wabgaonkar Poulami Debnath, Shubhashis Sengupta. 2018.

Dinesh Raghu, Sathish Indurthi, Jitendra Ajmera, and Sachindra Joshi. 2015. A statistical approach for non-sentential utterance resolution for interactive qa system. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 335–343.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Mingyue Shang, Zhenxin Fu, Nanyun Peng, Yansong Feng, Dongyan Zhao, and Rui Yan. 2018. Learning to converse with noisy data: Generation with calibration. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 4338–4344. International Joint Conferences on Artificial Intelligence Organization.

Zhiliang Tian, Wei Bi, Xiaopeng Li, and Nevin L Zhang. 2019. Learning to abstract for memory-augmented conversational response generation. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 3816–3825.

Zhiliang Tian, Rui Yan, Lili Mou, Yiping Song, Yansong Feng, and Dongyan Zhao. 2017. How to make context more useful? an empirical study on context-aware neural conversational models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 231–236.

Yu Wu, Wei Wu, Chen Xing, Ming Zhou, and Zhoujun Li. 2017. Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 496–505.

Chen Xing, Yu Wu, Wei Wu, Yalou Huang, and Ming Zhou. 2018. Hierarchical recurrent attention network for response generation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Zhuosheng Zhang, Jiangtong Li, Pengfei Zhu, Hai Zhao, and Gongshen Liu. 2018. Modeling multi-turn conversation with deep utterance aggregation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3740–3752.

Xiangyang Zhou, Lu Li, Daxiang Dong, Yi Liu, Ying Chen, Wayne Xin Zhao, Dianhai Yu, and Hua Wu. 2018. Multi-turn response selection for chatbots with deep attention matching network. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1118–1127.

# Unsupervised Context Rewriting for Open Domain Conversation

**Kun Zhou**[1*], **Kai Zhang**[2], **Yu Wu**[3], **Shujie Liu**[3], and **Jingsong Yu**[1]
[1]School of Software and Microelectronics, Peking University
[2]AI and Research Microsoft, Beijing, China
[3]Microsoft Research, Beijing, China
franciszhou@pku.edu.cn
{kaizh,Wu.Yu,shujliu}@microsoft.com
yjs@ss.pku.edu.cn

## Abstract

Context modeling has a pivotal role in open domain conversation. Existing works either use heuristic methods or jointly learn context modeling and response generation with an encoder-decoder framework. This paper proposes an explicit context rewriting method, which rewrites the last utterance by considering context history. We leverage pseudo-parallel data and elaborate a context rewriting network, which is built upon the CopyNet with the reinforcement learning method. The rewritten utterance is beneficial to candidate retrieval, explainable context modeling, as well as enabling to employ a single-turn framework to the multi-turn scenario. The empirical results show that our model outperforms baselines in terms of the rewriting quality, the multi-turn response generation, and the end-to-end retrieval-based chatbots.

## 1 Introduction

Recent years have witnessed remarkable progress in open domain conversation (non-task oriented dialogue system) (Ji et al., 2014; Li et al., 2016a) due to the easy-accessible conversational data and the development of deep learning techniques (Bahdanau et al., 2014). One of the most difficult problems for open domain conversation is how to model the conversation context.

A conversation context is composed of multiple utterances, which raises some challenges not existing in the sentence modeling, including: 1) topic transition; 2) plenty of coreferences (he, him, she, it, they); and 3) long term dependency. To tackle these problems, existing works either refine the context by appending keywords to the last turn utterance (Yan et al., 2016), or learn a vector representation with neural networks (Serban et al., 2017b). However, these methods have drawbacks, for instance, correct keywords cannot be selected by heuristics rules, and a fix-length vector is not able to handle a long context.

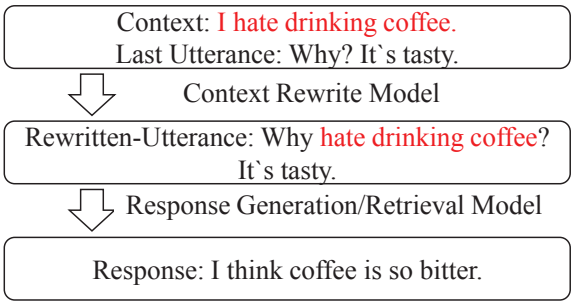


Figure 1: An Example of Context Rewriting.

We propose a context rewriting method, which explicitly rewrites the last utterance by considering the contextual information. Our goal is to generate a self-contained utterance, which neither has coreferences nor depends on other utterances in history. By this means, we change the input of chatbots from an entire conversation session to a rewritten sentence, which significantly reduces the difficulty of response generation/selection since the rewritten sentence is shorter and does not has redundant information. Figure 1 gives an example to further illustrate our idea.

The last utterance contains the word "it" which refers to the coffee in context. Moreover, "Why?" is an elliptical interrogative sentence, which is a shorter form of "Why hate drinking coffee?". We rewrite the context and yield a self-contained utterance "Why hate drinking coffee? It's tasty." Compared to previous methods, our method enjoys the following advantages: 1) The rewriting process is friendly to the retrieval stage of retrieval-based chatbots. Retrieval-based chatbots consists of two components: candidate retrieval and candidate reranking. Traditional works (Yan et al., 2016; Wu et al., 2017) pay little attention to the

*This work was done when the first author was an intern at Microsoft XiaoIce.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1834–1844, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

retrieval stage, which regards the entire context or context rewritten with heuristic rules as queries so noise is likely to be introduced; 2) It makes a step toward explainable and controllable context modeling, because the explicit context rewriting results are easy to debug and analyze. 3) Rewritten results enable us to employ a single-turn framework to solve the multi-turn conversation task. The single-turn conversation technology is more mature than the multi-turn conversation technology, which is able to achieve higher responding accuracy.

To this end, we propose a context rewriting network (CRN) to integrate the key information of the context and the original last utterance to build a rewritten one, so as to improve the answer performance. Our CRN model is a sequence-to-sequence network (Ilya Sutskever, 2014) with a bidirectional GRU-based encoder, and a GRU-based decoder enhanced with the CopyNet (Gu et al., 2016), which helps the CRN to directly copy words from the context. Due to the absence of the real written last utterance, unsupervised methods are used with two training stages, a pre-training stage with pseudo rewritten data, and a fine-tuning stage using reinforcement learning (RL) (Sutton et al., 1998) to maximize the reward of the final answer. Without the pre-training part, RL is unstable and slow to converge, since the randomly initialized CRN model cannot generate reasonable rewritten last utterance. On the other hand, only the pre-training part is not enough, since the pseudo data may contain errors and noise, which restricts the performance of our CRN.

We evaluate our method with four tasks, including the rewriting quality, the multi-turn response generation, the multi-turn response selection, and the end-to-end retrieval-based chatbots. Empirical results show that the outputs of our method are closer to human references than baselines. Besides, the rewriting process is beneficial to the end-to-end retrieval-based chatbots and the multi-turn response generation, and it shows slightly positive effect on the response selection.

## 2 Related Work

Recently, data-driven approaches for chatbots (Ritter et al., 2011; Ji et al., 2014) has achieved promising results. Existing work along this line includes retrieval-based methods (Hu et al., 2014; Ji et al., 2014; Wang et al., 2015; Yan et al., 2016; Zhou et al., 2016) and generation-based methods (Shang et al., 2015; Serban et al., 2016; Vinyals and Le, 2015; Li et al., 2016a,b; Xing et al., 2017; Serban et al., 2017a).

Early research into retrieval-based chatbots (Wang et al., 2013; Hu et al., 2014; Wang et al., 2015) only considers the last utterances and ignores previous ones, which is also called Short Text Conversation (STC). Recently, several studies (Lowe et al., 2015; Yan et al., 2016; Wu et al., 2017, 2018b) have investigated multi-turn response selection, and obtained better results in a comparison with STC. A common practice for multi-turn retrieval-based chatbots first retrieve candidates from a large index with a heuristic context rewriting method. For example, (Wu et al., 2017) and (Yan et al., 2016) refine the last utterance by appending keywords in history, and retrieve candidates with the refined utterance. Then, response selection methods are applied to measure the relevance between history and candidates.

A number of studies about generation-based chatbots have considered multi-turn response generation. Sordoni et al. (2015) is the pioneer of this type of research, it encodes history information into a vector and feeds to the decoder. Shang et al. (2015) propose three types of attention to utilize the context information. In addition, Serban et al. (2016) propose a Hierarchical Recurrent Encoder-Decoder model (HRED), which employs a hierarchical structure to represent the context. After that, latent variables (Serban et al., 2017b) and hierarchical attention mechanism (Xing et al., 2018) have been introduced to modify the architecture of HRED. Compared to previous work, the originality of this study is that it proposes a principle way instead of heuristic rules for context rewriting, and it does not depend on parallel data.

## 3 Model

Given a dialogue data set $\mathcal{D} = \{(U, r)_z\}_{z=1}^{N}$, where $U = \{u_0, \cdots, u_n\}$ represents a sequence of utterances and $r$ is a response candidate. We denote the last utterance as $q = u_n$ for simplicity, which is especially important to produce the response, and other utterances as $c = \{u_0, \cdots, u_{n-1}\}$. The goal of our paper is to rewrite $q$ as a self-contained utterance $q^*$ using useful information from $c$, which can not only reduce the noise in multi-turn context but also leverage a more simple single-turn framework to solve the multi-turn end-to-end tasks. We focus on the

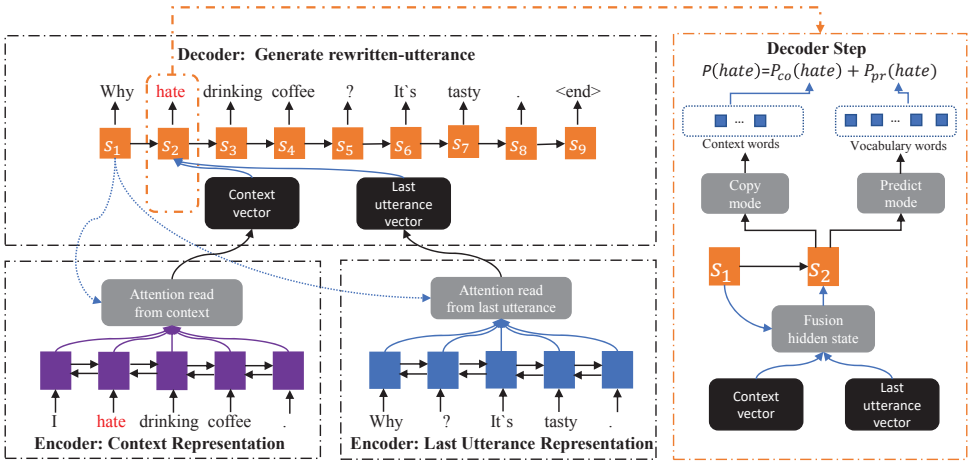


Figure 2: The Detail of CRN

multi-turn response generation and selection tasks.

To rewrite the last utterance $q$ with the help of context $c$, we propose a context rewriting network (CRN), which is a popularly used sequence-to-sequence network, equipped with a CopyNet to copy words from the original context $c$ (Section 3.1). Without the real paired data (pairs of the original and rewritten last utterance), our CRN model is firstly pre-trained with the pseudo data, generated by inserting extracted keywords from context into the original last utterance $q$ (Section 3.2). To let the final response to influence the rewriting process, reinforcement learning is leveraged to further enhance our CRN model, using the rewards from the response generation and selection tasks respectively (Section 3.3).

## 3.1 Context Rewriting Network

As shown in Figure 2, our context rewriting network (CRN) follows the sequence to sequence framework, consisting of three parts: one encoder to learn the context ($c$) representation, another encoder to learn the last utterance ($q$) representation, and a decoder to generate the rewritten utterance $q^*$. Attention is also used to focus on different words in the last utterance $q$ and the context $c$, and the copy mechanism is introduced to copy important words from the context $c$.

### 3.1.1 Encoder

To encode the context $c$ and the last utterance $q$, bidirectional GRU is leveraged to take both the left and right words in the sentence into consideration, by concatenating the hidden states of two GRU networks in positive time direction and negative time direction. With the bidirectional GRU, the last utterance $q$ is encoding into $H_Q = [h_{q_1}, \ldots, h_{q_{nq}}]$, and the context $c$ is encoding into $H_C = [h_{c_1}, \ldots, h_{c_{nc}}]$.

### 3.1.2 Decoder

The GRU network is also leveraged as decoder to generate the rewritten utterance $q^*$, in which the attention mechanism is used to extract useful information from the context $c$ and the last utterance $q$, and the copy mechanism is leveraged to directly copy words from the context $c$ into $q^*$. At each time step $t$, we fuse the information from $c$, $q$ and last hidden state $s_t$ to generate the input vector $z_t$ of GRU as following

$$z_t = W_f^T[s_t; \sum_{i=1}^{nq} \alpha_{q_i} h_{q_i}; \sum_{i=1}^{nc} \alpha_{c_i} h_{c_i}] + b \quad (1)$$

where [;] is the concatenation operation. $W_f$ and $b$ are trainable parameters, $s_t$ is the last hidden state of decoder GRU in step $t$, $\alpha_q$ and $\alpha_c$ are the weights of the words in $q$ and $c$ respectively, derived by the attention mechanism as following

$$\alpha_i = \frac{exp(e_i)}{\sum_{j=1}^{n} exp(e_j)} \quad (2)$$

$$e_i = h_i W_a s_t \quad (3)$$

where $h_i$ is the encoder hidden state of the $i^{th}$ word in $q$ or $c$, $W_a$ is the trainable parameter.

The copy mechanism is used to predict the next target word according to the probability of $p(y_t|s_t, H_Q, H_C)$, which is computed as

$$\begin{aligned} p(y_t|s_t, H_Q, H_C) = & \, p_{pr}(y_t|z_t) \cdot p_m(pr|z_t) \\ & + p_{co}(y_t|z_t) \cdot p_m(co|z_t) \end{aligned} \quad (4)$$

where $y_t$ is the $t-th$ word in response, $pr$ and $co$ stand for the predict-mode and the copy-mode, $p_{pr}(y_t|z_t)$ and $p_{co}(y_t|z_t)$ are the distributions of vocabulary word and context word which are im-

plemented by two MLP (multi layer perceptron) classifiers, respectively. And $p_m(\cdot|\cdot)$ indicates the probability to choose the two modes, which is a MLP (multi layer perceptron) classifier with softmax as the activation function:

$$p_m(pr|z_t) = \frac{e^{\psi_{pr}(y_t,H_Q,H_C)}}{e^{\psi_{pr}(y_t,H_Q,H_C)} + e^{\psi_{co}(y_t,H_Q,H_C)}} \tag{5}$$

where $\psi_{pr}(\cdot)$, $\psi_{co}(\cdot)$ are score functions for choosing the predict-mode and copy-mode with different parameters.

### 3.2 Pre-training with Pseudo Data

Instead of directly leverage RL to optimize our model CRN, which could be unstable and slow to converge, we pre-train our model CRN with pseudo-parallel data. Cross-entropy is selected as the training loss to maximize the log-likelihood of the pseudo rewritten utterance.

$$L_{MLE} = -\frac{1}{N}\sum_{i=1}^{n} \log(p(y_t|s_t, H_Q, H_C)) \tag{6}$$

The main challenge for the pre-training stage is how to generate good pseudo data, which can integrate suitable keywords from context and the original last utterance to form a better one to generate a good response. Given the context $c$, we extract keywords $w^*_{c_{1:n}}$ using pointwise mutual information (PMI) (in Section 3.2.1). With the extracted keywords $w^*_{c_{1:n}}$, language model is leveraged to find suitable positions to insert them into the original last utterance to generate rewritten candidates $s^*$, which will be re-ranked leveraging the information from following process (the response generation/selection) to get final pseudo rewritten utterance $Q^*$ (in Section 3.2.2). In the following of this section, we will introduce our pseudo data creation method in detail.

#### 3.2.1 Key Words Extraction

To penalize common and low frequent words, and prefers the "mutually informative" words, PMI is used to extract the keywords in the context $c$. Given a context word $w_c$, and a word $w_r$ in response $r$, it is the divides the prior distribution $p_c(w_c)$ by the posterior distribution $p(w_c|w_r)$ as shown as:

$$\text{PMI}(w_c, w_r) = -\log\frac{p_c(w_c)}{p(w_c|w_r)} \tag{7}$$

In order to select the keywords which contribute to the response, and are suitable to be shown in the last utterance, we also calculate $\text{PMI}(w_c, w_q)$ between the context word $w_c$ and any word $w_q$ in the last utterance. The final contribution score $\text{PMI}(w_c, q, r)$ for the context word $w_c$ to the last utterance $q$ and the response $r$ is calculated as

$$\text{norm}(\text{PMI}(w_c, q)) + \text{norm}(\text{PMI}(w_c, r)) \tag{8}$$

where $\text{norm}(\cdot)$ is the min-max normalization among all words in $c$, and $\text{PMI}(w_c, q)$ (similar for $\text{PMI}(w_c, r)$) is calculated as

$$\text{PMI}(w_c, q) = \sum_{w_q \in q} \text{PMI}(w_c, w_q). \tag{9}$$

The keywords $w^*_c$ with top-20% contribution score $\text{PMI}(w_c, q, r)$ against $r$ and $q$ are selected to insert into the last utterance $q$.[1]

#### 3.2.2 Pseudo Data Generation

Together with the extract candidate keyword, the words nearby are also extracted to form a continuous span to introduce more information, of which, at most 2 words before and after are considered. For one keyword, there are at most $C_3^1 * C_3^1 = 9$ span candidates. We apply a multi-layer RNN language model to insert the extracted key phrase to a suitable position in the last utterance. Top-3 generated sentences with high language model scores are selected as the rewritten candidates $s^*$.

With the information from the response, a rerank model is used to select the best one from the candidates $s^*$. For end-to-end generation task, the quality of candidates is measured with the cross-entropy of a single-turn attention based encoder-decoder model $M_{s2s}$, hoping that the good rewritten utterance can help to generate the proper response. For the end-to-end response selection task, the quality of the candidates is measured by the rank loss of a single-turn response selection model $M_{ir}$, hoping that the good one can distinguish the positive and negative responses.

$$L_{M_{s2s}}(r|s^*) = -\frac{1}{n}\sum_{i=1}^{n} \log p(r_1, \ldots, r_n|s^*) \tag{10}$$

$$L_{M_{ir}}(po, ne, s^*) = M_{ir}(po, s^*) - M_{ir}(ne, s^*) \tag{11}$$

In equation 10, $r_i$ is the $i-th$ word in response. In equation 11 $po$ is the positive response, and $ne$ are the negative one.

[1] This threshold of PMI is based on the observation on the development set.

### 3.3 Fine-Tuning with Reinforcement Learning

Since the generated pseudo data inevitably contains errors and noise, which limits the performance of the pre-trained model, we leverage the reinforcement learning method to build the direct connection between the context rewrite model CRN and different tasks. We first generate rewritten utterance candidates $q_r$ with our pre-trained model, and calculate the reward $R(q_r)$ which will be maximized to optimize the network parameters of our CRN. Due to the discrete choices of words in sequential generation, the policy gradient is used to calculate the gradient.

$$\nabla_\theta J(\theta) = E[R \cdot \nabla \log(P(y_t|x))] \quad (12)$$

For reinforcement learning in sequential generation task, instability is a serious problem. Similar to other works (Wu et al., 2018a), we combine MLE training objective with RL objective as

$$L_{com} = L^*_{rl} + \lambda L_{MLE} \quad (13)$$

where $\lambda$ is a harmonic weight.

By directly maximizing the reward from end tasks (response generation and selection), we hope that our CRN can correct the errors in the pseudo data and generate better rewritten last utterance. Two different rewards are used to fine-tune our CRN for the tasks of response generation and selection respectively. We will introduce them in detail in the following.

#### 3.3.1 End-to-end response generation reward

Similar as we do in Section 3.2.2, for end-to-end response generation task, we use the cross-entropy loss of a single-turn attention based encoder-decoder model $M_{s2s}$ to evaluate the quality of rewritten last utterance $q_r$ as

$$R_g(r, q^*, q_r) = L_{M_{s2s}}(r|q^*) - L_{M_{s2s}}(r|q_r) \quad (14)$$

where $L_{M_{s2s}}$ is defined in Equation 10, $r$ is the response candidate, $q_r$ is the generated candidate of our CRN, and $q^*$ is the pseudo rewritten candidate as introduced in Section 3.2.2. If $q_r$ can bring more useful information from context, it will get lower cross-entropy to generate $r$ than the original pseudo rewritten one $q^*$.

#### 3.3.2 End-to-end response selection reward

For end-to-end response selection task, we use a single-turn response selected model $M_{ir}$ to evaluate the quality of the generated candidate $q_r$ by the rank loss, it is calculated as

$$\begin{aligned} R_{ir}(po, ne, q^*, q_r) = L_{M_{ir}}(po, ne, q_r) \\ - L_{M_{ir}}(po, ne, q^*) \end{aligned} \quad (15)$$

where $L_{M_{ir}}$ is defined in Equation 11, $q_r$ is the generated candidate of our model, and $q^*$ is the pseudo candidate as introduced in Section 3.2.2. Similar to Equation 14, if $q_r$ can bring more useful information, it will do better to distinguish the negative and positive responses.

## 4 Experiment

We conduct four experiments to validate the effectiveness of our model, including the rewriting quality, the multi-turn response generation, the multi-turn response selection, and the end-to-end retrieval-based chatbots.

We crawl human-human context-response pairs from Douban Group which is a popular forum in China and remove duplicated pairs and utterances longer than 30 words. We create pseudo rewritten context as described in Section 3.1. Because most of the responses are only relevant with the last two turn utterances, following Li et al. (2016c), we remove the utterances beyond the last two turns. We finally split 6,844,393 $(c_i, q_i, q^*_i, r_i)$ quadruplets for training[2], 1000 for validation and 1074 for testing, and the last utterance in test set are selected by human and they all require rewriting to enhance information[3]. In the data set, the ratio between rewritten last utterance and un-rewritten[4] the last utterance is $1.426:1$. The average length of context $c_i$, last utterance $q_i$, response $r_i$, and rewritten last utterance $q^*_i$ are 12.69, 11.90, 15.15 and 14.27 respectively.

We pre-train the CRN with the pseudo-parallel data until it coverages, then we use the reinforcement learning technique described in Section 3.3 to fine-tune the CRN. The specific details of the model hyper-parameters and optimization algorithms can be found in the **Supplementary**.

### 4.1 Rewriting Quality Evaluation

The detail of the training process is the same as Section 4.2.1. We evaluate the rewriting quality by calculating the BLEU-4 score (Papineni et al.,

[2]The data in the training set do not overlap with the test data of the four tasks.

[3]Our human-annotated test set is available at https://github.com/lovelife/chat

[4]Un-rewritten ones can handle utterances do not rely on their contexts.

| | BLEU-4 |
|---|---|
| Last Utterance | 34.2 |
| Last Utterance + Context | 37.1 |
| Last Utterance + Keyword | 49.8 |
| CRN | 50.9 |
| CRN + RL | **54.2** |

Table 1: The result of rewriting quality.

2002), a sequence order sensitive metric, between the system outputs and human references. Such references are rewritten by a native speaker who considers the information in context. It is required that the rewritten last utterance is self-contained.

We compare our models CRN with three baselines. Firstly, we report the BLEU-4 scores of the origin last utterance and the combination of the last utterance and context. Additionally, following Wu et al. (2017), we append five keywords to the last utterance, where the keywords are selected from the context by TF-IDF weighting, which is named by Last Utterance + Keyword. The IDF score is computed on the entire training corpus.

Table 1 shows the experiment result, which indicates that our rewriting method outperforms heuristic methods. Moreover, a 54.2 BLEU-4 score means that the rewritten sentences are very similar to the human references. CRN-RL has a higher score than CRN-Pre-train on BLEU-4, it proves reinforcement learning promotes our model effectively.

### 4.2 Multi-turn Response Generation

Section 4.1 demonstrates the outputs of our model are more similar to the human rewritten references. In this part, we will show the influence of the context rewriting for response generation.

We use the same test data in Section 4.1 to evaluate our model in the multi-turn response generation task. The multi-turn response generation is defined as, given an entire context consisting of multiple utterances, a model should generate informative, relevant, and fluent responses. We compare against the following previous works:

**S2SA:** We adopt the well-known Seq2Seq with attention (Bahdanau et al., 2014) model to generate responses by feeding the last utterances $q$ as source sentences.

**HRED:** Serban et al. (2016) propose using a hierarchical encoder-decoder model to handle the multi-turn response generation problem, where each utterance and the entire session are represented by different networks.

**Dynamic, Static:** Zhang et al. (2018) propose two state-of-the-art hierarchical recurrent attention networks for response generation. The dynamic model dynamically weights utterances in the decoding process, while the static model weights utterances before the decoding process.

#### 4.2.1 Implementation Details

Given a context $c$ and last utterance $q$, we first rewrite them with the CRN. Then the rewritten last utterance $q'$ is fed to a single-turn generation model. The details of the model can be found in **Supplementary** and we set the same sizes of hidden states and embedding in all models. We regard two adjacent utterances in our training data to construct the training dataset (5,591,794 utterance-response pairs) for the single-turn generation model. We do not use the rewritten context as the input in the training phase, since we would like to guarantee the gain only comes from the rewriting mechanism at the inference stage.

#### 4.2.2 Evaluation Metrics

We regard the human response as the ground truth, and use the following metrics:

**Word overlap based metrics**: We report BLEU score (Papineni et al., 2002) between model outputs and human references.

**Embedding based metrics**: As BLEU is not correlated with the human annotation perfectly, following (Liu et al., 2016), we employ embedding based metrics, Embedding Average (Average), Embedding Extrema (Extrema), and Embedding Greedy (Greedy) to evaluate results. The word2vec is trained on the training data set, whose dimension is 200.

**Diversity**: We evaluate the response diversity based on the ratios of distinct unigrams and bigrams in generated responses, denoted as Distinct-1 and Distinct-2 (Li et al., 2016a).

**Human Annotation**: We ask three native speakers to annotate the quality of generated responses. We compare the quality of our model with HRED and S2SA. We conduct 5-scale rating: +3, +2, +1, 0 and -1. +3: the response is natural, informative and relevant with context; +2: the response is natural, informative, but might not be relevant enough; +1: the response is natural, but might not be informative and relevant enough (e.g., I don't know); 0: The response makes no sense, irrelevant, or grammatically broken; -1: The response or utterances cannot be understood.

| | BLEU-1 | BLEU-2 | BLEU-3 | Average | Extrema | Greedy | Distinct-1 | Distinct-2 |
|---|---|---|---|---|---|---|---|---|
| S2SA | 5.72 | 2.80 | 1.37 | 11.14 | 8.58 | 13.15 | 25.55 | 58.89 |
| HRED | 10.10 | 5.53 | 2.75 | 27.45 | 21.71 | 27.43 | 15.22 | 32.19 |
| Dynamic | 7.05 | 3.54 | 1.75 | 17.77 | 14.20 | 18.94 | 6.22 | 15.08 |
| Static | 9.31 | 5.01 | 2.77 | 21.32 | 17.36 | 22.49 | 6.59 | 17.31 |
| CRN | 13.26 | 8.43 | 4.64 | 32.31 | 26.97 | 33.59 | **31.48** | **67.02** |
| CRN + RL | **13.63** | **8.69** | **4.88** | **33.14** | **27.49** | **34.68** | 31.42 | 65.10 |

Table 2: Automatic evaluation results.

| | 3 | 2 | 1 | 0 | -1 | avg |
|---|---|---|---|---|---|---|
| S2SA | 14.48% | **48.56%** | 31.01% | 5.48% | 0.46% | 1.72 |
| HRED | 13.28% | 16.90% | **65.00%** | 3.99% | 0.46% | 1.40 |
| CRN+S2SA | 37.05% | 31.01% | 25.07% | 6.04% | 0.46% | 1.99 |
| CRN+S2SA+RL | **42.43%** | 29.25% | 15.04% | **12.63%** | 0.46% | **2.02** |

Table 3: The distribution of human evaluation in response generation model.

#### 4.2.3 Evaluation Results

Table 2 presents the automatic evaluation results, showing that our models outperform baselines on relevance and diversity. Table 3 gives the human annotation results, which also demonstrates the superiority of our models. Our models significantly improve response diversity, mainly because the rewritten sentence contains rich information that is capable of guiding the model to generate a specific output. After reinforcement learning our model promotes on BLEU and embedding metrics, it is because reinforcement learning can build the connection between the utterance-rewrite model with the response generation model for exploring better rewritten-utterance. But our model drops a little on the diversity metrics after reinforcement learning, this owes to the fact that the reward is biased to relevance rather than diversity. The similar phenomenon can be observed in the comparison of HRED and S2SA, which means that although relevance can increase by considering context information, general responses become more frequently concurrently.

Table 3 presents the distribution of score in human evaluation, we can observe that most of the responses generated by HRED and S2SA get 1 or 2 in human evaluation, while most of the responses generated by our model can get 2 or 3. It proves that our model can reduce noisy from context and construct an informative utterance to generate high-quality response. However, after reinforcement learning our model gets more 0 and 3 score, that is because after reinforcement learning, our model becomes unstable and prefers to extract more words from context. The score of one candidate will increase or decrease a lot if useful keywords or wrong keywords are inserted into the last utterance, respectively. In fact, more utterances are rewritten better after reinforcement learning so the average evaluation score improves.

### 4.3 Multi-turn Response Selection

We also evaluate the multi-turn response selection task of retrieval-based chatbots, which aims to select proper responses from a candidate pool by considering the context. We use the Douban Conversation Corpus released by Wu et al. (2017), which is created by crawling a popular Chinese forum, the Douban Group [5], covering various topics. Its training set contains 0.5 million conversational sessions, and the validation set contains 50,000 sessions. The negative instances in both sets are randomly sampling with a 1:1 positive-negative ratio. The test set contains 1000 conversation contexts, and each context has 10 response candidates with human annotations.

We split the last utterance from each context in the training data, and forms 0.5 million of $(q, r)$ pairs. Subsequently, we train a single-turn Deep Attention Matching Network (Zhou et al., 2018) consuming the pair as an input, which is denoted as DAM$_{single}$. The DAM$_{single}$ model is treated as a rank model in Section 3.2.2 and a reward function in Section 3.3.2. In the testing stage, we use the CRN and the DAM$_{single}$ to assign a score for each candidate. Notably, the original DAM takes a context-response pair as an input, which is set as a baseline method. The parameters of the DAM is the same as its original paper.

[5] https://www.douban.com/group/explore

| | MAP | MRR | P@1 | $R_{10}$@1 | $R_{10}$@2 | $R_{10}$@5 |
|---|---|---|---|---|---|---|
| TF-IDF (Lowe et al., 2015) | 0.331 | 0.359 | 0.180 | 0.096 | 0.172 | 0.405 |
| RNN (Lowe et al., 2015) | 0.390 | 0.422 | 0.208 | 0.118 | 0.223 | 0.589 |
| CNN (Lowe et al., 2015) | 0.417 | 0.440 | 0.226 | 0.121 | 0.252 | 0.647 |
| LSTM (Lowe et al., 2015) | 0.485 | 0.527 | 0.320 | 0.187 | 0.343 | 0.720 |
| BiLSTM (Lowe et al., 2015) | 0.479 | 0.514 | 0.313 | 0.184 | 0.330 | 0.716 |
| Multi-View (Zhou et al., 2016) | 0.505 | 0.543 | 0.342 | 0.202 | 0.350 | 0.729 |
| DL2R (Yan et al., 2016) | 0.488 | 0.527 | 0.330 | 0.193 | 0.342 | 0.705 |
| MV-LSTM (Pang et al., 2016) | 0.498 | 0.538 | 0.348 | 0.202 | 0.351 | 0.710 |
| Match-LSTM (Wang and Jiang, 2017) | 0.500 | 0.537 | 0.345 | 0.202 | 0.348 | 0.720 |
| Attentive-LSTM (Tan et al., 2016) | 0.495 | 0.523 | 0.331 | 0.192 | 0.328 | 0.718 |
| SMN (Wu et al., 2017) | 0.529 | 0.569 | 0.397 | 0.233 | 0.396 | 0.724 |
| DAM (Zhou et al., 2018) | 0.550 | 0.601 | 0.427 | 0.254 | 0.410 | **0.757** |
| $DAM_{single}$ | 0.543 | 0.592 | 0.414 | 0.255 | 0.427 | 0.725 |
| CRN + $DAM_{single}$ | 0.548 | 0.603 | 0.428 | 0.262 | 0.439 | 0.727 |
| CRN + RL + $DAM_{single}$ | **0.552** | **0.605** | **0.431** | **0.267** | **0.445** | 0.729 |

Table 4: Evaluation results on multi-turn response selection. The numbers of baselines are copied from (Zhou et al., 2018)

| | Win | Loss | Tie |
|---|---|---|---|
| CRN vs baseline | 51.3% | 29.7% | 19.0% |
| CRN + RL vs baseline | 35.7% | 21.8% | 42.5% |

Table 5: The result of end-to-end response selection subjective evaluation.

#### 4.3.1 Evaluation Results

Table 4 shows the response selection performances of different methods. We can see that our model achieves a comparable performance with state-of-the-art DAM model, but only consuming a rewritten utterance rather than the whole context. This indicates that our model is able to recognize important content in context and generate a self-contained sentence. This argument is also verified by 1 point promotion compared with $DAM_{single}$ which only uses the last utterance as an input. Additionally, $DAM_{single}$ just underperforms DAM 1 point, meaning that the last utterance is very important for response selection. It supports our assumption that the last utterance is important which is a good prototype for context rewriting.

### 4.4 End-to-End Multi-turn Response Selection

In practice, a retrieval-based chatbot first retrieves a number of response candidates from an index, then re-ranks the candidates with the aforementioned response selection methods. Previous works pay little attention to the retrieval stage, which just appends some keywords to the last utterance to collect candidates (Wu et al., 2017).

Because our model is able to rewrite context and generate a self-contained sentence. We expect it could retrieve better candidates at the first step, benefiting to the end-to-end performance. Since it is hard to evaluate the retrieval-stage, we evaluate the end-to-end response selection performance. Specifically, we first rewrite the contexts in the test set with CRN, and then retrieve 10 candidates with the rewritten context from the index[6]. $DAM_{single}$ is employed to compute relevance scores with the rewritten utterance and the candidates. The candidate with the top score is selected as the final response. The baseline model appends keywords from context to the last utterance for retrieval and use the original DAM with all context as the input to select final response.

We recruit three annotators to do a side-by-side evaluation, and the model outputs are shuffled before human evaluation. The majority of the three judgments are selected as a result. If both outputs are hard to distinguish, we choose Tie as the result.

#### 4.4.1 Evaluation Results

We list the side-by-side evaluation results in Table 5. Human annotators prefer the outputs of our model. On account of the reranking modules are comparable, we can infer that the gain comes from the better retrieval candidates. However, reinforcement learning does not have a positive effect on this task. We find our reinforced model becomes more conservative, it tends to generate shorter rewritten utterance than our pre-training model. That may be beneficial for response rerank, but if wrong keywords or noise words are extracted from context. It will reduce the quality of retrieved candidates, leading to an undesired end-to-end result.

[6]The authors (Wu et al., 2017) share their index data with us

| | | | |
|---|---|---|---|
| Context | 我十九号昆明飞厦门<br>I will fly from Kunming to Xiamen in 19th | 上了豆瓣更无聊<br>Douban is so boring | 你5点半下班?<br>Do you go off duty in 5:30? |
| Last Utterance | 我也想去<br>I want to go, too | 好像是这样的<br>Yes | 是的呀，坐上班车了都<br>Yes, I am in bus now |
| Rewritten-Utterance | 昆明飞厦门我也想去<br>I want to fly from Kunming to Xiamen, too | 豆瓣好像是这样的<br>Yes, Douban is. | 是的呀，坐上班车了都下班?<br>Yes, I am now in bus off duty? |
| Rewritten-Utterance+RL | 我也想去厦门<br>I want to go to Xiamen, too | 好像是这样的豆瓣更无聊<br>Yes, Douban is so boring | 是的呀5点半下班，坐上班车了都<br>Yes, go off duty in 5:30, I am now in bus |
| S2SA | 那就出发吧<br>Let's go | 你是双子<br>You are Gemini | 5点半<br>5:30 |
| HRED | 我也是<br>Me too | 是啊<br>Yes | 好吧<br>Okay |
| Our Model | 昆明大理丽江<br>Kunming, Dali, Lijiang | 豆瓣毁一生<br>Douban can ruin whole life. | 下班了吗<br>Do you go off? |
| Our Model+RL | 厦门欢迎你<br>Welcome to Xiamen | 无聊到爆<br>I'm bored to death | 我也是坐班车<br>I am taking the bus, too |

Table 6: The examples of end-to-end response generation.

### 4.5 Case Study

We list the generated examples of our models and baseline models for End-to-end Generation Chatbot and Retrieval Chatbot. Because the submission space is quite limited, we put the case study of Retrieval Chatbot in the Supplementary Material.

#### 4.5.1 End-to-end Generation Chatbots

Table 6 presents the generated examples of our models and baselines, our model can extract the keywords from the context which is helpful to generate an informative response, but the HRED model often generates safe responses like "$Metoo$" or "$Yes$". It is because the input information from context and last utterance contain so much noise, some of the context words are useless for the last utterance to generate responses. Our model can extract important keywords from noisy context and insert them into the last utterance, it is not only easy to control and explain in a chat-bot system, but also transmit useful information directly to last utterance. The input of S2SA model is the last utterance, so it can generate diverse response due to less noise, but its relevancy with context is low. Our model succeeds fusing advantage from both models and get a significant promotion. Comparing the generated responses by our pre-training model and reinforced model, the rewritten-utterance inferred by our pre-training model may be more informative, but the final generated response may be unrelated to context and last utterance. It is because reinforcement learning can build the connection between the utterance-rewrite model with response generation model for exploring better rewritten-utterance. A better rewritten-utterance should be helpful to generate a context-related response, Too much information inserted will add noise and too little will be useless.

## 5 Conclusion

This paper investigates context modeling in open domain conversation. It proposes an unsupervised context rewriting model, which benefits to candidate retrieval and controllable conversation. Empirical results show that the rewriting contexts are similar to human references, and the rewriting process is able to improve the performance of multi-turn response selection, multi-turn response generation, and end-to-end retrieval chatbots.

## Acknowledgement

We are thankful to Yue Liu, Sawyer Zeng and Libin Shi for their supportive work. We also gratefully thank the anonymous reviewers for their insightful comments.

## References


Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems*, pages 2042–2050.

Quoc V. Le Ilya Sutskever, Oriol Vinyals. 2014. Sequence to sequence learning with neural networks. In *Nips*, pages 3104–3112.

Zongcheng Ji, Zhengdong Lu, and Hang Li. 2014. An information retrieval approach to short text conversation. *ArXiv*, abs/1408.6988.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 110–119.

Jiwei Li, Michel Galley, Chris Brockett, Georgios P. Spithourakis, Jianfeng Gao, and William B. Dolan. 2016b. A persona-based neural conversation model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016c. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1192–1202.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2122–2132.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *Proceedings of the SIGDIAL 2015 Conference, The 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 2-4 September 2015, Prague, Czech Republic*, pages 285–294.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016. Text matching as image recognition. In *Proceedings of the thirtieth AAAI Conference on Artificial Intelligence*, pages 2793–2799.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318. Association for Computational Linguistics.

Alan Ritter, Colin Cherry, and William B Dolan. 2011. Data-driven response generation in social media. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 583–593. Association for Computational Linguistics.

Iulian Vlad Serban, Tim Klinger, Gerald Tesauro, Kartik Talamadupula, Bowen Zhou, Yoshua Bengio, and Aaron C. Courville. 2017a. Multiresolution recurrent neural networks: An application to dialogue response generation. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3288–3294.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 3776–3784.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017b. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*, pages 3295–3301.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1577–1586.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 196–205.

Richard S Sutton, Andrew G Barto, et al. 1998. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge.

Ming Tan, Cícero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2016. Improved representation learning for question answer matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Oriol Vinyals and Quoc V. Le. 2015. A neural conversational model. *CoRR*, abs/1506.05869.

Hao Wang, Zhengdong Lu, Hang Li, and Enhong Chen. 2013. A dataset for research on short-text conversations. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 935–945.

Mingxuan Wang, Zhengdong Lu, Hang Li, and Qun Liu. 2015. Syntax-based deep matching of short texts. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Shuohang Wang and Jing Jiang. 2017. Machine comprehension using match-lstm and answer pointer. *ICLR*.

Lijun Wu, Fei Tian, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2018a. A study of reinforcement learning for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018, Brussels, Belgium, November 2-4, 2018*.

Yu Wu, Furu Wei, Shaohan Huang, Yunli Wang, Zhoujun Li, and Ming Zhou. 2018b. Response generation by context-aware prototype editing. In *AAAI*.

Yu Wu, Wei Wu, Chen Xing, Ming Zhou, and Zhoujun Li. 2017. Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 496–505.

Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2017. Topic aware neural response generation. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3351–3357.

Chen Xing, Yu Wu, Wei Wu, Yalou Huang, and Ming Zhou. 2018. Hierarchical recurrent attention network for response generation. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5610–5617.

Rui Yan, Yiping Song, and Hua Wu. 2016. Learning to respond with deep neural networks for retrieval-based human-computer conversation system. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, SIGIR 2016, Pisa, Italy, July 17-21, 2016*, pages 55–64.

Wei-Nan Zhang, Yiming Cuiy, Yifa Wang, Qingfu Zhu, Lingzhi Li, Lianqiang Zhouz, and Ting Liu. 2018. Context-sensitive generation of open-domain conversational responses. In *COLING*, pages 2437–2447.

Xiangyang Zhou, Daxiang Dong, Hua Wu, Shiqi Zhao, Dianhai Yu, Hao Tian, Xuan Liu, and Rui Yan. 2016. Multi-view response selection for human-computer conversation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 372–381.

Xiangyang Zhou, Lu Li, Daxiang Dong, Yi Liu, Ying Chen, Wayne Xin Zhao, Dianhai Yu, and Hua Wu. 2018. Multi-turn response selection for chatbots with deep attention matching network. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 1118–1127.

# Dually Interactive Matching Network for Personalized Response Selection in Retrieval-Based Chatbots

**Jia-Chen Gu**[1], **Zhen-Hua Ling**[1], **Xiaodan Zhu**[2], **Quan Liu**[1,3]
[1]National Engineering Laboratory for Speech and Language Information Processing,
University of Science and Technology of China, Hefei, China
[2]ECE, Queen's University, Kingston, Canada
[3]State Key Laboratory of Cognitive Intelligence, iFLYTEK Research, Hefei, China
gujc@mail.ustc.edu.cn, zhling@ustc.edu.cn,
xiaodan.zhu@queensu.ca, quanliu@ustc.edu.cn

## Abstract

This paper proposes a dually interactive matching network (DIM) for presenting the personalities of dialogue agents in retrieval-based chatbots. This model develops from the interactive matching network (IMN) which models the matching degree between a context composed of multiple utterances and a response candidate. Compared with previous *persona* fusion approaches which enhance the representation of a context by calculating its similarity with a given persona, the DIM model adopts a dual matching architecture, which performs interactive matching between responses and contexts and between responses and personas respectively for ranking response candidates. Experimental results on PERSONA-CHAT dataset show that the DIM model outperforms its baseline model, i.e., IMN with *persona* fusion, by a margin of 14.5% and outperforms the current state-of-the-art model by a margin of 27.7% in terms of top-1 accuracy **hits**@1.

## 1 Introduction

Building a conversation system with intelligence is challenging. Response selection, which aims to select a potential response from a set of candidates given the context of a conversation, is an important technique to build retrieval-based chatbots (Zhou et al., 2018). Many previous studies on single-turn (Wang et al., 2013) or multi-turn response selection (Lowe et al., 2015; Zhou et al., 2018; Gu et al., 2019) rank response candidates according to their semantic relevance with the given context.

With the emergence and popular use of personal assistants such as Apple Siri, Google Now and Microsoft Cortana, the techniques of making personalized dialogues has attracted much research attention in recent years (Li et al., 2016; Zhang et al., 2018; Mazaré et al., 2018). Zhang et al. (2018) constructed a PERSONA-CHAT dataset for building personalized dialogue agents, where each *persona* was represented as multiple sentences of profile description. An example dialogue conditioned on given profiles from this dataset is given in Table 1 for illustration.

A persona fusion method for personalized response selection was also proposed by Zhang et al. (2018). In this method, given a context and a persona composed of several profile sentences, the similarities between the context representation and all profile sentences are computed first using attention to get the persona representation. Then, the persona representation is applied to enhance the context representation by a simple concatenation or addition operation. Finally, the enhanced context representation is used to rank response candidates. This method has two main deficiencies. First, the context is treated as a whole for calculating its attention towards profile sentences. However, each context is composed of multiple utterances and these utterances may play different roles when matching different profile sentences. Second, the interactions between the persona and each response candidate are ignored when deriving the persona representation.

In this paper, the interactive matching network (IMN) (Gu et al., 2019) is adopted as the fundamental architecture to build our baseline and improved models for personalized response selection. The baseline model follows the persona fusion method proposed by Zhang et al. (2018) and two improved models are then proposed. First, an IMN-based persona fusion model with fine-grained context-persona interaction is designed. In this model, each utterance in a context, instead of the whole context, is used to calculate its similarity with each profile sentence in a persona. Second, a dually interactive matching network (DIM) is proposed by formulating the task of personalized response selection as a dual matching problem,



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1845–1854,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| | Persona 1 | | Persona 2 |
|---|---|---|---|
| Original | I just bought a brand new house.<br>I like to dance at the club.<br>I run a dog obedience school.<br>I have a big sweet tooth.<br>I like taking and posting selkies. | Original | I love to meet new people.<br>I have a turtle named timothy.<br>My favorite sport is ultimate frisbee.<br>My parents are living in bora bora.<br>Autumn is my favorite season. |
| Revised | I have purchased a home.<br>Just go dancing at the nightclub, it is fun!<br>I really enjoy animals.<br>I enjoy chocolate.<br>I pose for pictures and put them online. | Revised | I like getting friends.<br>Reptiles make good pets.<br>I love to run around and get out my energy.<br>My family lives on a island.<br>I love watching the leaves change colors. |

**Dialogue**
Person 1: Hello, how are you doing tonight?
Person 2: I am well an loving this interaction how are you?
Person 1: I am great. I just got back from the club.
Person 2: This is my favorite time of the year season wise.
Person 1: I would rather eat chocolate cake during this season.
Person 2: What club did you go to? Me an timothy watched tv.
Person 1: I went to club chino. What show are you watching?
Person 2: LOL oh okay kind of random.
Person 1: I love those shows. I am really craving cake.
Person 2: Why does that matter any? I went outdoors to play frisbee.
Person 1: It matters because I have a sweet tooth.
Person 2: So? LOL I want to meet my family at home in bora.
Person 1: My family lives in alaska. It is freezing down there.
Person 2: I bet it is oh I could not.

Table 1: An example dialogue from the PERSONA-CHAT dataset.

i.e., finding a response that can properly match the given context and persona simultaneously. The DIM model calculates the interactions between the context and the response, and between the persona and the response in parallel, in order to derive the final matching feature for response selection.

We test our proposed methods on the PERSONA-CHAT dataset (Zhang et al., 2018). Results show that the IMN-based utterance-level persona fusion model and the DIM model can obtain a top-1 accuracy **hits**@1 improvement of 2.4% and 14.5%, respectively, over the baseline model, i.e., the IMN-based context-level persona fusion model. Finally, our proposed DIM model outperforms the current state-of-the-art model by a margin of 27.7% in terms of top-1 accuracy **hits**@1 on the PERSONA-CHAT dataset.

In summary, the contributions of this paper are three-fold. (1) An IMN-based fine-grained persona fusion model is designed in order to consider the utterance-level interactions between contexts and personas. (2) A dually interactive matching network (DIM) is proposed by formulating the task of personalized response selection as a dual matching problem, aiming to find a response that can properly match the given context and persona simultaneously. (3) Experimental results on the PERSONA-CHAT dataset demonstrate that our proposed models outperform the baseline and state-of-the-art models by large margins on the accuracy of response selection.

## 2 Related Work

### 2.1 Response Selection

Response selection is an important problem in building retrieval-based chatbots. Existing work on response selection can be categorized into single-turn (Wang et al., 2013) and multi-turn dialogues (Lowe et al., 2015; Zhou et al., 2018; Gu et al., 2019). Early studies have been more on single-turn dialogues, considering only the last utterance of a context for response matching. More recently, the research focus has been shifted to multi-turn conversations, a more practical setup for real applications. Wu et al. (2017) proposed the sequential matching network (SMN) which first matched the response with each context utterance and then accumulated the matching information by a recurrent neural network (RNN). Zhou et al. (2018) proposed the deep attention matching network (DAM) to construct representations at different granularities with stacked self-attention. Gu et al. (2019) proposed the interactive matching network (IMN)

to enhance the representations of the context and response at both the word-level and sentence-level, and to perform the bidirectional and global interactions between the context and response in order to derive the matching feature vector.

### 2.2 Persona for Chatbots

Chit-chat models suffer from a lack of a consistent personality as they are typically trained over many dialogues, each with different speakers, and a lack of explicit long-term memory as they are typically trained to produce an utterance given only a very recent dialogue history. Li et al. (2016) proposed a persona-based neural conversation model to capture individual characteristics such as background information and speaking style. Miller et al. (2016) proposed the key-value memory network, where the keys were dialogue histories, i.e., contexts, and the values were next dialogue utterances. Zhang et al. (2018) proposed the profile memory network by considering the dialogue history as input and then performing attention over the persona to be combined with the dialogue history. Mazaré et al. (2018) proposed the fine-tuned persona-chat (FT-PC) model which first pretrained a model using a large-scale corpus with external knowledge and then fine-tuned it on the PERSONA-CHAT dataset.

In general, all these methods adopted a context-level persona fusion strategy, which first obtained the embedding vector of a context and then computed the similarities between the whole context and each profile sentence to acquire the persona representation. However, such persona fusion is relatively too coarse. The utterance-level representations of contexts are not leveraged. The interactions between the persona and each response candidate are also ignored when deriving the persona representation.

## 3 Task Definition

Given a dialogue dataset $\mathcal{D}$ with personas, an example of the dataset can be represented as $(c, p, r, y)$. Specifically, $c = \{u_1, u_2, ..., u_{n_c}\}$ represents a context with $\{u_m\}_{m=1}^{n_c}$ as its utterances and $n_c$ as the utterance number. $p = \{p_1, p_2, ..., p_{n_p}\}$ represents a persona with $\{p_n\}_{n=1}^{n_p}$ as its profile sentences and $n_p$ as the profile number. $r$ represents a response candidate. $y \in \{0, 1\}$ denotes a label. $y = 1$ indicates that $r$ is a proper response for $(c, p)$; otherwise, $y = 0$.

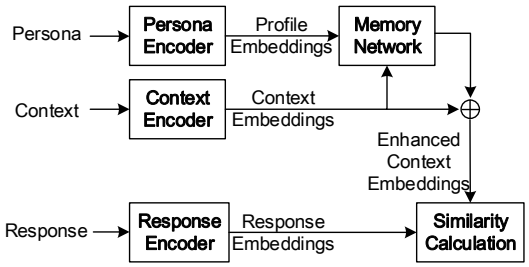


(a) Context-level persona fusion

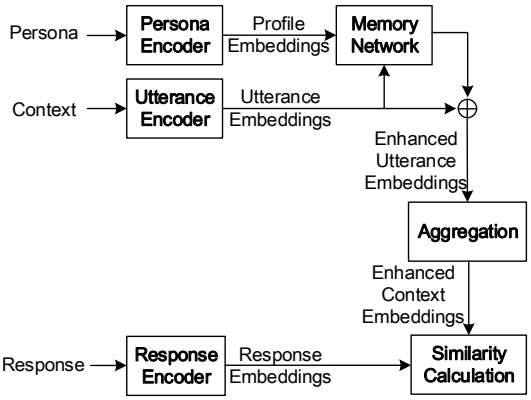


(b) Utterance-level persona fusion

Figure 1: Comparison of the model architectures for (a) context-level persona fusion and (b) utterance-level persona fusion.

Our goal is to learn a matching model $g(c, p, r)$ from $\mathcal{D}$. For any context-persona-response triple $(c, p, r)$, $g(c, p, r)$ measures the matching degree between $(c, p)$ and $r$. A softmax output layer over all response candidates is adopted in this model. The model parameters are trained by minimizing a multi-class cross-entropy loss function on $\mathcal{D}$.

## 4 IMN-Based Persona Fusion

The model architecture used by previous methods with persona fusion (Zhang et al., 2018; Mazaré et al., 2018) is shown in Figure 1(a). It first obtains the context representation and then computes the similarities between the whole context and each profile sentence in a persona. Attention weights are calculated for all profile sentences to obtain the persona representation. Finally, the persona representation is combined with the context representation through concatenation or addition operations.

Formally, the representations of the whole context which is the concatenation of utterances, the context utterances, and the profile sentences are denoted as $\mathbf{c}$, $\{\mathbf{u}_m\}_{m=1}^{n_c}$ and $\{\mathbf{p}_n\}_{n=1}^{n_p}$ respectively,

where $\mathbf{c}$, $\mathbf{u}_m$ and $\mathbf{p}_n \in \mathbb{R}^d$. In previous context-level persona fusion methods, the enhanced context representation $\mathbf{c}^+$ fused with persona information is calculated as

$$\mathbf{c}^+ = \mathbf{c} + \sum_n \textbf{Softmax}(\mathbf{c} \cdot \mathbf{p}_n)\mathbf{p}_n. \quad (1)$$

Then, the similarity between $\mathbf{c}^+$ and the response representation are computed to get the matching degree of $(c, p, r)$.

In this paper, we build our baseline model based on IMN (Gu et al., 2019). After the context and response embeddings are obtained in the IMN model, the context-level persona fusion architecture shown in Figure 1(a) is applied to integrate persona information. All model parameters are estimated in an end-to-end manner. This baseline model is denoted as IMN$_{ctx}$ in this paper.

Considering each context is composed of multiple utterances and these utterances may play different roles when matching different profile sentences, we propose to improve the baseline model by fusing the persona information at a fine-grained utterance-level as shown in Figure 1(b). This model is denoted as IMN$_{utr}$ in this paper. First, the similarities between each context utterance and each profile sentence are computed and the enhanced representation $\mathbf{u}_m^+$ of each context utterance is calculated as

$$\mathbf{u}_m^+ = \mathbf{u}_m + \sum_n \textbf{Softmax}(\mathbf{u}_m \cdot \mathbf{p}_n)\mathbf{p}_n. \quad (2)$$

Then, these enhanced utterance representations are aggregated into the enhanced context representation as

$$\mathbf{c}^+ = \textbf{Aggregation}(\{\mathbf{u}_m^+\}_{m=1}^{n_c}), \quad (3)$$

where either RNN or attention-based aggregation (Gu et al., 2019) can be employed.

# 5 Dually Interactive Matching Network

## 5.1 Model Overview

Previous studies on personalized response selection treat personas as supplementary information to enhance context representations by attention-based interaction. In this paper, we formulate the task of personalized response selection as a dual matching problem. The selected response is expected to properly match the given context and persona respectively. Here, personas are considered as equally important counterparts to contexts for ranking response candidates. The interactive matching between the context and response, and that between the persona and response constitute the dually interactive matching network (DIM).

The DIM model is composed of five layers. Figure 2 shows an overview of the architecture. Details about each layer are provided in the following subsections.

## 5.2 Word Representation Layer

We follow the setting used in IMN (Gu et al., 2019), which constructs word representations by combining general pre-trained word embeddings, those estimated on the task-specific training set, as well as character-level embeddings, in order to deal with the out-of-vocabulary issue.

Formally, embeddings of the *m*-th utterance in a context, the *n*-th profile sentence in a persona and a response candidate are denoted as $\mathbf{U}_m = \{\mathbf{u}_{m,i}\}_{i=1}^{l_{u_m}}$, $\mathbf{P}_n = \{\mathbf{p}_{n,j}\}_{j=1}^{l_{p_n}}$ and $\mathbf{R} = \{\mathbf{r}_k\}_{k=1}^{l_r}$ respectively, where $l_{u_m}$, $l_{p_n}$ and $l_r$ are the numbers of words in $\mathbf{U}_m$, $\mathbf{P}_n$ and $\mathbf{R}$ respectively. Each $\mathbf{u}_{m,i}$, $\mathbf{p}_{n,j}$ or $\mathbf{r}_k$ is an embedding vector of *d*-dimensions.

## 5.3 Sentence Encoding Layer

The context utterances, profile sentences and response candidate are encoded by bidirectional long short-term memories (BiLSTMs) (Hochreiter and Schmidhuber, 1997). We denote the calculations as follows,

$$\bar{\mathbf{u}}_{m,i} = \textbf{BiLSTM}(\mathbf{U}_m, i), i \in \{1, ..., l_{u_m}\}, \quad (4)$$
$$\bar{\mathbf{p}}_{n,j} = \textbf{BiLSTM}(\mathbf{P}_n, j), j \in \{1, ..., l_{p_n}\}, \quad (5)$$
$$\bar{\mathbf{r}}_k = \textbf{BiLSTM}(\mathbf{R}, k), k \in \{1, ..., l_r\}, \quad (6)$$

where $\bar{\mathbf{U}}_m = \{\bar{\mathbf{u}}_{m,i}\}_{i=1}^{l_{u_m}}$, $\bar{\mathbf{P}}_n = \{\bar{\mathbf{p}}_{n,j}\}_{j=1}^{l_{p_n}}$ and $\bar{\mathbf{R}} = \{\bar{\mathbf{r}}_k\}_{j=1}^{l_r}$. The parameters of these three BiLSTMs are shared in our implementation.

## 5.4 Matching Layer

The interactions between the context and the response and those between the persona and the response can provide useful matching information for deciding the matching degree between them. Here, the DIM model adopts the same strategy as in the IMN model (Gu et al., 2019) which considers the global and bidirectional interactions between two sequences.

Take the context-response matching as an example. First, the context representation $\bar{\mathbf{C}} = \{\bar{\mathbf{c}}_i\}_{i=1}^{l_c}$ with $l_c = \sum_{m=1}^{n_c} l_{u_m}$ is formed by

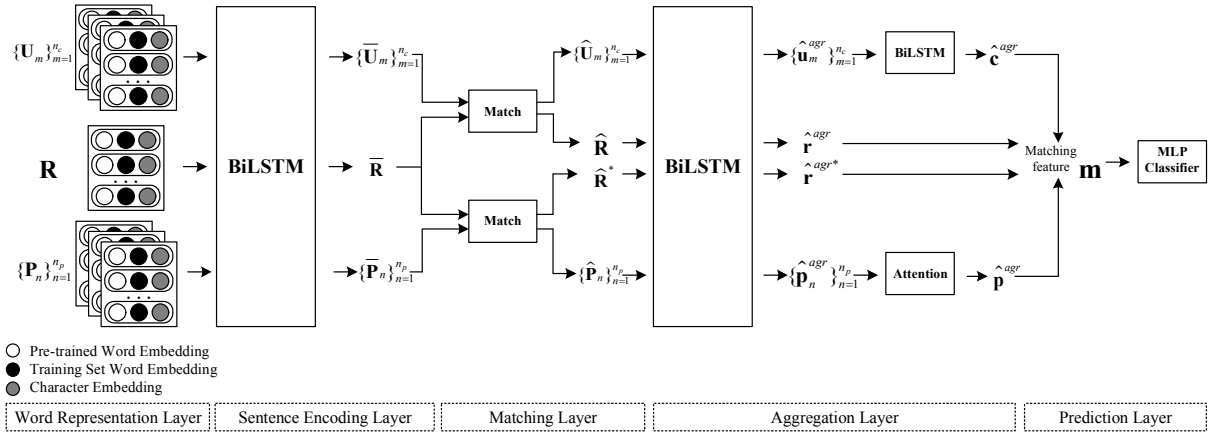


Figure 2: An overview of our proposed DIM model.

concatenating the set of utterance representations $\{\bar{\mathbf{U}}_m\}_{m=1}^{n_c}$. Then, a soft alignment is performed by computing the attention weight between each tuple $\{\bar{\mathbf{c}}_i, \bar{\mathbf{r}}_k\}$ as

$$e_{ik} = (\bar{\mathbf{c}}_i)^\top \cdot \bar{\mathbf{r}}_k. \quad (7)$$

After that, local inference is determined by the attention weights computed above to obtain the local relevance between a context and a response bidirectionally. For a word in the context, its relevant representation carried by the response is identified and composed using $e_{ik}$ as

$$\tilde{\mathbf{c}}_i = \sum_{k=1}^{l_r} \frac{exp(e_{ik})}{\sum_{l=1}^{l_r} exp(e_{il})} \bar{\mathbf{r}}_k, i \in \{1, ..., l_c\}, \quad (8)$$

where the contents in $\{\bar{\mathbf{r}}_k\}_{k=1}^{l_r}$ that are relevant to $\bar{\mathbf{c}}_i$ are selected to form $\tilde{\mathbf{c}}_i$. Then, we define $\tilde{\mathbf{C}} = [\tilde{\mathbf{c}}_1, ..., \tilde{\mathbf{c}}_{l_c}]$. The same calculation is performed for each word in the response to form its relevant representation carried by the context as

$$\tilde{\mathbf{r}}_k = \sum_{i=1}^{l_c} \frac{exp(e_{ik})}{\sum_{l=1}^{l_c} exp(e_{lk})} \bar{\mathbf{c}}_i, k \in \{1, ..., l_r\}, \quad (9)$$

and we define $\tilde{\mathbf{R}} = [\tilde{\mathbf{r}}_1, ..., \tilde{\mathbf{r}}_{l_r}]$. To further enhance the collected information, the differences and element-wise products between $\{\bar{\mathbf{C}}, \tilde{\mathbf{C}}\}$ and between $\{\bar{\mathbf{R}}, \tilde{\mathbf{R}}\}$ are computed, and are then concatenated with the original vectors to obtain the enhanced representations as follows,

$$\widehat{\mathbf{C}} = [\bar{\mathbf{C}}; \tilde{\mathbf{C}}; \bar{\mathbf{C}} - \tilde{\mathbf{C}}; \bar{\mathbf{C}} \odot \tilde{\mathbf{C}}], \quad (10)$$

$$\widehat{\mathbf{R}} = [\bar{\mathbf{R}}; \tilde{\mathbf{R}}; \bar{\mathbf{R}} - \tilde{\mathbf{R}}; \bar{\mathbf{R}} \odot \tilde{\mathbf{R}}]. \quad (11)$$

So far we have collected the relevant information between the context and response. The enhanced context representation $\widehat{\mathbf{C}}$ is further converted back to matching matrices of separated utterances as $\{\widehat{\mathbf{U}}_m\}_{m=1}^{n_c}$.

The persona-response matching is conducted identically to the context-response matching introduced above, where the representations of profile sentences $\{\bar{\mathbf{P}}_n\}_{n=1}^{n_p}$ are used, instead of the representations of context utterances $\{\bar{\mathbf{U}}_m\}_{m=1}^{n_c}$. The results of persona-response matching are denoted as $\{\widehat{\mathbf{P}}_n\}_{n=1}^{n_p}$ and $\widehat{\mathbf{R}}^*$.

### 5.5 Aggregation Layer

The aggregation layer converts the matching matrices of context utterances, profile sentences and response into a final matching feature vector.

First, each matching matrix $\widehat{\mathbf{U}}_m, \widehat{\mathbf{R}}, \widehat{\mathbf{P}}_n$ and $\widehat{\mathbf{R}}^*$ are processed by BiLSTMs as

$$\hat{\mathbf{u}}_{m,i}^{utr} = \textbf{BiLSTM}(\widehat{\mathbf{U}}_m, i), i \in \{1, ..., l_{u_m}\}, \quad (12)$$

$$\hat{\mathbf{r}}_k^{utr} = \textbf{BiLSTM}(\widehat{\mathbf{R}}, k), k \in \{1, ..., l_r\}, \quad (13)$$

$$\hat{\mathbf{p}}_{n,j}^{utr} = \textbf{BiLSTM}(\widehat{\mathbf{P}}_n, j), j \in \{1, ..., l_{p_n}\}, \quad (14)$$

$$\hat{\mathbf{r}}_k^{utr*} = \textbf{BiLSTM}(\widehat{\mathbf{R}}^*, k), k \in \{1, ..., l_r\}, \quad (15)$$

where the four BiLSTMs share the same parameters in our implementation. Then, the aggregated embeddings are calculated by max pooling and last-hidden-state pooling operations as

$$\hat{\mathbf{u}}_m^{agr} = [\hat{\mathbf{u}}_{m,max}^{utr}; \hat{\mathbf{u}}_{m,l_{u_m}}^{utr}], m \in \{1, ..., n_c\}, \quad (16)$$

$$\hat{\mathbf{r}}^{agr} = [\hat{\mathbf{r}}_{max}^{utr}; \hat{\mathbf{r}}_{l_r}^{utr}], \quad (17)$$

$$\hat{\mathbf{p}}_n^{agr} = [\hat{\mathbf{p}}_{n,max}^{utr}; \hat{\mathbf{p}}_{n,l_{p_n}}^{utr}], n \in \{1, ..., n_p\}, \quad (18)$$

$$\hat{\mathbf{r}}^{agr*} = [\hat{\mathbf{r}}_{max}^{utr*}; \hat{\mathbf{r}}_{l_r}^{utr*}]. \quad (19)$$

Next, the sequences of $\hat{\mathbf{u}}_m^{agr}$ and $\hat{\mathbf{p}}_n^{agr}$ are further aggregated to get the embedding vectors for the context and the persona respectively.

**Context aggregation** As the utterances in a context are chronologically ordered, the utterance embeddings $\widehat{\mathbf{U}}^{agr} = \{\hat{\mathbf{u}}_m^{agr}\}_{m=1}^{n_c}$ are sent into another BiLSTM following the chronological order of utterances in the context. Combined max pooling and last-hidden-state pooling operations are then performed to obtain the context embeddings as

$$\hat{\mathbf{u}}_m^{ctx} = \textbf{BiLSTM}(\widehat{\mathbf{U}}^{agr}, m), m \in \{1, ..., n_c\}, \tag{20}$$

$$\hat{\mathbf{c}}^{agr} = [\hat{\mathbf{u}}_{max}^{ctx}; \hat{\mathbf{u}}_{n_c}^{ctx}]. \tag{21}$$

**Persona aggregation** As the profile sentences in a persona are independent to each other, an attention-based aggregation is designed to derive the persona embeddings as follows,

$$\alpha_n = \textbf{ReLU}(\mathbf{w}^\top \cdot \hat{\mathbf{p}}_n^{agr} + b), \tag{22}$$

$$\hat{\mathbf{p}}^{agr} = \sum_{n=1}^{n_p} \frac{e^{\alpha_n}}{\sum_{k=1}^{n_p} e^{\alpha_k}} \hat{\mathbf{p}}_n^{agr}, \tag{23}$$

where $\mathbf{w}$ and $b$ are parameters need to be estimated during training.

Last, the final matching feature vector is the concatenation of context, persona and response embeddings as

$$\mathbf{m} = [\hat{\mathbf{c}}^{agr}; \hat{\mathbf{r}}^{agr}; \hat{\mathbf{p}}^{agr}; \hat{\mathbf{r}}^{agr*}], \tag{24}$$

where the first two features describe the context-response matching, and the last two describe the persona-response matching.

### 5.6 Prediction Layer

The final matching feature vector is then sent into a multi-layer perceptron (MLP) classifier with softmax output. Here, the MLP is designed to predict whether a $(c, p, r)$ triple match appropriately based on the derived matching feature vector. Finally, the MLP returns a probability to denote the matching degree.

## 6 Experiments

### 6.1 Dataset

We tested our proposed methods on the PERSONA-CHAT dataset (Zhang et al., 2018) which contains multi-turn dialogues conditioned on personas. The dataset consists of 8939 complete dialogues for training, 1000 for validation, and 968 for testing. Response selection is performed at every turn of a complete dialogue, which results in 65719 dialogues for training, 7801 for validation, and 7512 for testing in total. Positive responses are true responses from humans and negative ones are randomly sampled. The ratio between positive and negative responses is 1:19 in the training, validation, and testing sets. There are 955 possible personas for training, 100 for validation, and 100 for testing, each consisting of 3 to 5 profile sentences. To make this task more challenging, a version of revised persona descriptions are also provided by rephrasing, generalizing, or specializing the original ones. Since the personas of both speakers in a dialogue are available, the response selection task can be conditioned on the speaker's persona ("self persona") or the dialogue partner's persona ("their persona") respectively.

### 6.2 Evaluation Metrics

We used the same evaluation metrics as in the previous work (Zhang et al., 2018). Each model aimed to select the best-matched response from available candidates for the given context $c$ and persona $p$. We calculated the recall of the true positive replies, denoted as **hits**@1. In addition, the mean reciprocal rank (**MRR**) (Voorhees, 1999) metric was also adopted to take the rank of the correct response over all candidates into consideration.

### 6.3 Training Details

For building the IMN, IMN$_{ctx}$, IMN$_{utr}$ and DIM models, the Adam method (Kingma and Ba, 2015) was employed for optimization with a batch size of 16. The initial learning rate was 0.001 and was exponentially decayed by 0.96 every 5000 steps. Dropout (Srivastava et al., 2014) with a rate of 0.2 was applied to the word embeddings and all hidden layers. A word representation is a concatenation of a 300-dimensional GloVe embedding (Pennington et al., 2014), a 100-dimensional embedding estimated on the training set using the Word2Vec algorithm (Mikolov et al., 2013), and 150-dimensional character-level embeddings with window sizes {3, 4, 5}, each consisting of 50 filters. The word embeddings were not updated during training. All hidden states of the LSTM have 200 dimensions. The MLP at the prediction layer have 256 hidden units with ReLU (Nair and Hinton, 2010) activation. The maximum number of characters in a word, that of words in a context utterance, of utterances in a context, and of words in a response were set to be 18, 20, 15, and

| | **hits**@1 | **MRR** |
|---|---|---|
| IR baseline | 21.4 | - |
| Starspace | 31.8 | - |
| Profile | 31.8 | - |
| KV Profile | 34.9 | - |
| IMN | 63.8 | 75.8 |

Table 2: Evaluation results of the IMN model and previous methods on PERSONA-CHAT dataset without using personas. All the results except ours are copied from Zhang et al. (2018).

20, respectively. We padded with zeros if the number of utterances in a context was less than 15; otherwise, we kept the last 15 utterances. For the IMN$_{ctx}$, IMN$_{utr}$ and the DIM models, the maximum number of words in a profile sentence and that of profile sentences in a persona were set to be 15 and 5, respectively. Similarly, we padded with zeros if the number of profile sentences in a persona was less than 5. The development set was used to select the best model for testing.

All code was implemented in the TensorFlow framework (Abadi et al., 2016) and is published to help replicate our results [1].

### 6.4 Experimental Results

Table 2 presents the evaluation results of our reproduced IMN model (Gu et al., 2019) and previous methods on PERSONA-CHAT dataset without using personas. It can be seen that the IMN model outperformed other models on this dataset by a margin larger than 28.9% in terms of **hits**@1. As introduced above, our proposed models for personalized response selection were all built on IMN.

Table 3 presents the evaluation results of our proposed and previous methods on PERSONA-CHAT under various persona configurations. The t-test shows that the differences between our proposed models, i.e., IMN$_{utt}$ and DIM, and the baseline model, i.e. IMN$_{ctx}$, were both statistically significant with $p$-value $< 0.01$. We can see that the fine-grained persona fusion at the utterance level rendered a **hits**@1 improvement of 2.4% and an **MRR** improvement of 1.9% by comparing IMN$_{ctx}$ and IMN$_{utr}$ conditioned on original self personas. The DIM model outperformed its baseline IMN$_{ctx}$ by a margin of 14.5% in terms of **hits**@1 and 10.5% in terms of **MRR**. Compared with the FT-PC model (Mazaré et al., 2018) which was first pretrained using a large-scale corpus and then fine-tuned on the PERSONA-CHAT dataset, the DIM model outperformed it by a margin of 10.0% in terms of **hits**@1 conditioned on revised self personas. Another advantage of DIM is that it was trained in an end-to-end mode without pre-training and using any external knowledge. Lastly, the DIM model outperforms previous models by margins larger than 27.7% in terms of **hits**@1 conditioned on original self personas.

**Improvement of Using Personas** Examining the numbers which indicate the gains or losses after adding persona conditions in Table 3, we can see that the context-level persona fusion improves the performance of previous models significantly when original self personas are used. However, the gain achieved by the IMN$_{ctx}$ model is limited. One possible reason is that the IMN model performs attention-based interactions between the context and the response in order to get their matching feature for response selection. Thus, the context embeddings shown in Fig. 1(a) contained the information from both the context and the response, which may be inappropriate for the following context-level persona fusion shown in Eq. (1). The improvement achieved by the DIM model is much higher because it adopts a dual matching framework to address this issue.

**Original vs. Revised** Comparing with using original personas, it is more difficult for the models conditioned on the revised personas to extract useful persona information, as shown by the limited improvement achieved by the previous models shown in Table 3. One possible reason is that there are fewer shared words between the response and the persona revised by rephrasing, generalizing, or specializing, which increases the difficulty of understanding the persona and its relationships with the response. For example, it is easier for models to judge the matching degree between the original profile "*Autumn is my favorite season.*" and the response "*This is my favorite time of the year season wise.*" than between the revised profile "*I love watching the leaves change colors.*" and the response. On the contrary, our proposed DIM model still obtains a **hits**@1 improvement of 6.9% and an **MRR** improvement of 5.4% when conditioned on the revised self personas, which can be attributed to the direct and interactive

[1] https://github.com/JasonForJoy/DIM

| | Self Persona | | | | Their Persona | | | |
|---|---|---|---|---|---|---|---|---|
| | Original | | Revised | | Original | | Revised | |
| | **hits**@1 | **MRR** | **hits**@1 | **MRR** | **hits**@1 | **MRR** | **hits**@1 | **MRR** |
| IR baseline | 41.0 (+19.6) | - | 20.7 (-0.7) | - | 18.1 (-3.3) | - | 18.1 (-3.3) | - |
| Starspace | 48.1 (+16.3) | - | 32.2 (+0.4) | - | 24.5 (-7.3) | - | 26.1 (-5.7) | - |
| Profile | 47.3 (+15.5) | - | 35.4 (+3.6) | - | 28.3 (-3.5) | - | 29.4 (-2.4) | - |
| KV Profile | 51.1 (+16.2) | - | 35.1 (+0.2) | - | 29.1 (-5.8) | - | 28.9 (-6.0) | - |
| FT-PC | - | - | 60.7 (-) | - | - | - | - | - |
| IMN$_{ctx}$ | 64.3 (+0.5) | 76.2 (+0.4) | 63.8 (+0.0) | 75.8 (+0.0) | 63.7 (-0.1) | 75.8 (+0.0) | 63.5 (-0.3) | 75.7 (-0.1) |
| IMN$_{utr}$ | 66.7 (+2.9) | 78.1 (+2.3) | 64.0 (+0.2) | 76.0 (+0.2) | 63.9 (+0.1) | 75.9 (+0.1) | 63.7 (-0.1) | 75.7 (-0.1) |
| DIM | **78.8 (+15.0)** | **86.7 (+10.9)** | **70.7 (+6.9)** | **81.2 (+5.4)** | **64.0 (+0.2)** | **76.1 (+0.3)** | **63.9 (+0.1)** | **76.0 (+0.2)** |

Table 3: Performance of the proposed and previous methods on the PERSONA-CHAT under various persona configurations. The meanings of "Self Persona", "Their Persona", "Original", and "revised" can be found in Section 6.1. All results except ours are copied from Zhang et al. (2018); Mazaré et al. (2018). Numbers in parentheses indicate the gains or losses after adding the persona conditions.

persona-response matching used in this model.

**Self vs. Their** As shown in Table 3, no significant gains can be obtained when the models are conditioned on the personas of dialogue partners. Note that there are no significant performance losses with our proposed methods, while the losses of previous models are 2.4% to 7.3% in terms of **hits**@1.

## 7 Analysis

### 7.1 Ablations

| | **hits**@1 | **MRR** |
|---|---|---|
| DIM | 78.8 | 86.7 |
| - persona | 63.8 | 75.8 |
| - context | 48.8 | 60.9 |

Table 4: Ablation tests of removing either persona-response matching or context-response matching in the DIM model conditioned on original self personas.

To demonstrate the importance of the dual matching framework followed by our proposed DIM model, ablation tests were performed using the original self personas and the results are shown in Table 4. We can see that both the persona-response matching and the context-response matching contribute to the performance of the DIM model. It is reasonable that the context-persona matching is more important because contexts provide the fundamental semantic descriptions for response selection. On the other hand, the single persona-response matching can also achieve a **hits**@1 of 48.8% and an **MRR** of 60.9%, which shows the usefulness of utilizing persona information to select the best-matched response.

### 7.2 Interactive Matching in DIM

In order to investigate the effectiveness of the interaction matching between the context and the response and that between the persona and the response in the DIM model, a case study was conducted by visualizing the response-to-context and response-to-persona attention weights used in Eq. (9). The results are shown in Fig. 3. We can see that some important words such as "*dogs*" in the response selected its relevant words such as "*animals*" in the context to derive the context-response matching features. Some important profile texts such as "*I love animals and have two dogs.*" also obtained large attention weights for getting the persona-response matching features. This experimental result confirms our formulation of the task of personalized response selection as a dual matching problem.

### 7.3 Transfer Test

| Train \ Test | Original | Revised |
|---|---|---|
| Original | 78.8 | 66.3 |
| Revised | 77.6 | 70.7 |

Table 5: **hits**@1 results of transfer tests on the DIM model.

Transfer tests were conducted by training and evaluating the DIM model using mismatched types of personas. The results are reported in Table 5. It shows that the DIM model achieved a better performance when testing on the same type of personas as training. Meanwhile, the model trained on the revised personas and tested on the original personas achieved less loss than the ones trained

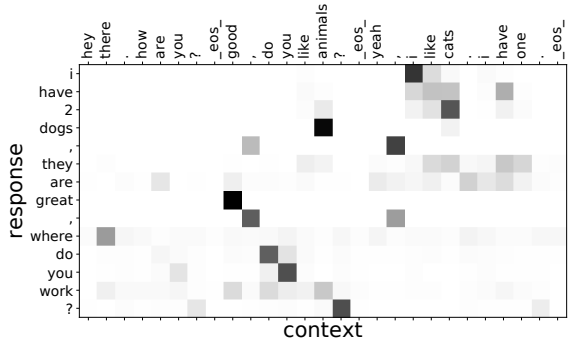


(a) Response-to-context attention weights

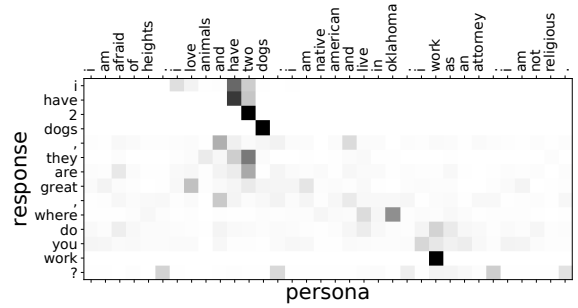


(b) Response-to-persona attention weights

Figure 3: Visualizations of (a) response-to-context or (b) response-to-persona attention weights at the matching layer for a test sample. The darker units correspond to larger values.

on the original personas and tested on the revised personas, which shows that the revised personas can provide a better generalization ability to the DIM model than the original ones.

## 8 Conclusions

In this paper, we formulate the task of personalized response selection as a dual matching problem to search for a response that can properly match the given context and persona simultaneously. A new model named Dually Interactive Matching Network (DIM) is proposed, which performs the interaction matching between the context and response as well as between the persona and response in parallel, in order to derive the final matching features for personalized response selection. Experimental results show that DIM improves over the IMN models with context-level or utterance-level persona fusion, outperforming previous methods and achieving a new state-of-the-art performance on the PERSONA-CHAT dataset. In the future, we will explore models to make better use of dialogue partners' persona for response selection.

## Acknowledgments

We thank the anonymous reviewers for their valuable comments.

## References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek Gordon Murray, Benoit Steiner, Paul A. Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016, Savannah, GA, USA, November 2-4, 2016.*, pages 265–283.

Jia-Chen Gu, Zhen-Hua Ling, and Quan Liu. 2019. Interactive matching network for multi-turn response selection in retrieval-based chatbots. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. ACM.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Jiwei Li, Michel Galley, Chris Brockett, Georgios P. Spithourakis, Jianfeng Gao, and William B. Dolan. 2016. A persona-based neural conversation model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *Proceedings of the SIGDIAL 2015 Conference, The 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 2-4 September 2015, Prague, Czech Republic*, pages 285–294.

Pierre-Emmanuel Mazaré, Samuel Humeau, Martin Raison, and Antoine Bordes. 2018. Training millions of personalized dialogue agents. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2775–2779.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed

representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 3111–3119.

Alexander H. Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1400–1409.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 807–814.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Ellen M. Voorhees. 1999. The TREC-8 question answering track report. In *Proceedings of The Eighth Text REtrieval Conference, TREC 1999, Gaithersburg, Maryland, USA, November 17-19, 1999*.

Hao Wang, Zhengdong Lu, Hang Li, and Enhong Chen. 2013. A dataset for research on short-text conversations. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 935–945.

Yu Wu, Wei Wu, Chen Xing, Ming Zhou, and Zhoujun Li. 2017. Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 496–505.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 2204–2213.

Xiangyang Zhou, Lu Li, Daxiang Dong, Yi Liu, Ying Chen, Wayne Xin Zhao, Dianhai Yu, and Hua Wu. 2018. Multi-turn response selection for chatbots with deep attention matching network. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 1118–1127.

# DyKgChat: Benchmarking Dialogue Generation Grounding on Dynamic Knowledge Graphs

**Yi-Lin Tuan  Yun-Nung Chen  Hung-yi Lee**
National Taiwan University, Taipei, Taiwan
pascaltuan@gmail.com  y.v.chen@ieee.org  hungyilee@ntu.edu.tw

## Abstract

Data-driven, knowledge-grounded neural conversation models are capable of generating more informative responses. However, these models have not yet demonstrated that they can zero-shot adapt to updated, unseen knowledge graphs. This paper proposes a new task about how to apply dynamic knowledge graphs in neural conversation model and presents a novel TV series conversation corpus (DyKgChat) for the task. Our new task and corpus aids in understanding the influence of dynamic knowledge graphs on responses generation. Also, we propose a preliminary model that selects an output from two networks at each time step: a sequence-to-sequence model (Seq2Seq) and a multi-hop reasoning model, in order to support dynamic knowledge graphs. To benchmark this new task and evaluate the capability of adaptation, we introduce several evaluation metrics and the experiments show that our proposed approach outperforms previous knowledge-grounded conversation models. The proposed corpus and model can motivate the future research directions[1].

## 1 Introduction

In the chit-chat dialogue generation, neural conversation models (Sutskever et al., 2014; Sordoni et al., 2015; Vinyals and Le, 2015) have emerged for its capability to be fully data-driven and end-to-end trained. While the generated responses are often reasonable but *general* (without useful information), recent work proposed knowledge-grounded models (Eric et al., 2017; Ghazvininejad et al., 2018; Zhou et al., 2018b; Qian et al., 2018) to incorporate external facts in an end-to-end fashion without hand-crafted slot filling. Effectively combining text and external knowledge

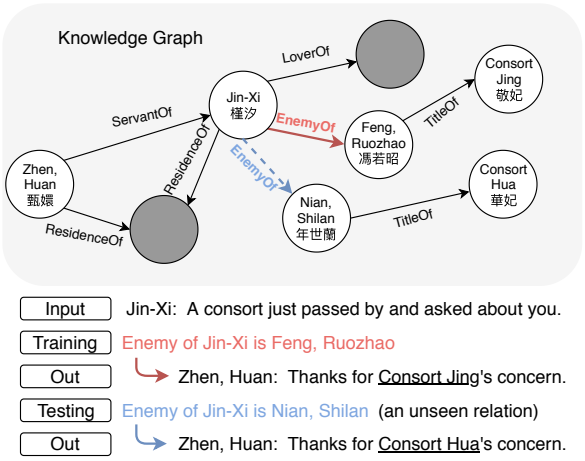


Figure 1: An example of an ideal conversation model with dynamic knowledge graphs.

graphs have also been a crucial topic in *question answering* (Yin et al., 2016; Hao et al., 2017; Levy et al., 2017; Sun et al., 2018; Das et al., 2019). Nonetheless, prior work rarely analyzed the model capability of zero-shot adaptation to dynamic knowledge graphs, where the states/entities and their relations are temporal and evolve as a single time scale process. For example, as shown in Figure 1, the entity *Jin-Xi* was originally related to the entity *Feng, Ruozhao* with the type *EnemyOf*, but then evolved to be related to the entity *Nian, Shilan*.

The goal of this paper is to facilitate knowledge-grounded neural conversation models to learn and zero-shot adapt with dynamic knowledge graphs. To our observation, however, there is no existing conversational data paired with dynamic knowledge graphs. Therefore, we collect a TV series corpus—*DyKgChat*, with facts of the fictitious life of characters. DyKgChat includes a Chinese palace drama *Hou Gong Zhen Huan Zhuan (HGZHZ)*, and an English sitcom *Friends*,which contain dialogues, speakers, scenes (e.g., the places and listeners), and the corresponded knowl-

[1]The data and code are available in https://github.com/Pascalson/DyKGChat.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1855–1865, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| | |
|---|---|
| HGZHZ | **Zhen-Huan**: She must be frightened. It should blame me. I should not ask her to play chess.<br>**甄嬛**: 姐姐定是嚇壞了。都怪臣妾不好，好端端的來叫梅姐姐下棋做什麼。<br>**Doctor-Wen**: Relax, **Concubine-Huan madame**. **Lady Shen** is just injured but is fine.<br>**溫太醫**: **莞嬪娘娘**請放心，**惠貴人**的精神倒是沒有大礙，只是傷口燒得有些厲害。 |
| Friends | **Joey**: C' mon , you're going out with the guy! There's gotta be something wrong with him!<br>**Chandler**: Alright **Joey**, be nice. So does he have a hump? A hump and a hairpiece? |

Table 1: Examples of DyKgChat corpus.

edge graphs including explicit information such as the relations *FriendOf*, *EnemyOf*, and *ResidenceOf* as well as the linked entities. Table 1 shows some examples from DyKgChat.

Prior graph embedding based knowledge-grounded conversation models (Sutskever et al., 2014; Ghazvininejad et al., 2018; Zhu et al., 2017) did not directly use the graph structure, so it is unknown how a changed graph will influence the generated responses. In addition, key-value retrieved-based models (Yin et al., 2016; Eric et al., 2017; Levy et al., 2017; Qian et al., 2018) retrieve only *one-hop* relation paths. As fictitious life in drama, realistic responses often use knowledge entities existing in multi-hop relational paths, e.g., the residence of a friend of mine. Therefore, we propose a model that incorporates multi-hop reasoning (Lao et al., 2011; Neelakantan et al., 2015; Xiong et al., 2017) on the graph structure into a neural conversation generation model. Our proposed model, called quick adaptive dynamic knowledge-grounded neural conversation model (Qadpt), is based on a Seq2Seq model (Sutskever et al., 2014) with a widely-used copy mechanism (Gu et al., 2016; Merity et al., 2017; He et al., 2017; Xing et al., 2017; Zhu et al., 2017; Eric et al., 2017; Ke et al., 2018). To enable multi-hop reasoning, the model factorizes a transition matrix for a *Markov chain*.

In order to focus on the capability of producing reasonable knowledge entities and adapting with dynamic knowledge graphs, we propose two types of automatic metrics. First, given the provided knowledge graphs, we examine if the models can generate responses with proper usage of multi-hop reasoning over knowledge graphs. Second, after randomly replacing some crucial entities in knowledge graphs, we test if the models can accordingly generate correspondent responses. The empirical results show that our proposed model has the desired advantage of zero-shot adaptation with dynamic knowledge graphs, and can serve as a preliminary baseline for this new task. To sum up, the contributions of this paper are three-fold:

- A new task, *dynamic knowledge-grounded conversation generation*, is proposed.
- A newly-collected TV series conversation corpus *DyKgChat* is presented for the target task.
- We benchmark the task by comparing many prior models and the proposed quick adaptive dynamic knowledge-grounded neural conversation model (Qadpt), providing the potential of benefiting the future research direction.

## 2 Task Description

For each single-turn conversation, the input message and response are respectively denoted as $x = \{x_t\}_{t=1}^{m}$ and $y = \{y_t\}_{t=1}^{n}$, where $m$ and $n$ are their lengths. Each turn $(x, y)$ is paired with a knowledge graph $\mathcal{K}$, which is composed of a collection of triplets $(h, r, t)$, where $h, t \in \mathcal{V}$ (the set of entities) and $r \in \mathcal{L}$ (the set of relationships). Each word $y_t$ in a response belongs to either generic words $\mathcal{W}$ or knowledge graph entities $\mathcal{V}$. The task is two-fold:

1. Given an input message $x$ and a knowledge graph $\mathcal{K}$, the goal is to generate a sequence $\{\hat{y}_t\}_{t=1}^{n}$ that is not only as similar as possible to the ground-truth $\{y_t\}_{t=1}^{n}$, but contains *correct knowledge graph entities* to reflect the information.
2. After a knowledge graph is updated to $\mathcal{K}'$, where some triplets are revised to $(h, r, t')$ or $(h, r', t)$, the generated sequence should contain *correspondent knowledge graph entities* in $\mathcal{K}'$ to reflect the updated information.

### 2.1 Evaluation Metrics

To evaluate dynamic knowledge-grounded conversation models, we propose two types of evaluation metrics for validating two aspects described above.

#### 2.1.1 Knowledge Entity Modeling

There are three metrics focusing on the knowledge-related capability.

**Knowledge word accuracy (KW-Acc).** Given the ground-truth sentence as the decoder inputs, at each time step, it evaluates how many knowledge graph entities are correctly predicted.

$$\text{KW-Acc} = \sum_{t=1}^{n} P(\hat{y}_t = y_t \mid y_1 y_2 \dots y_{t-1}, y_t \in \mathcal{V}).$$

For example, after perceiving the partial ground-truth response "*If Jin-Xi not in*" and knowing the next word should be a knowledge graph entity, KW-Acc measures if the model can predict the correct word "*Yongshou Palace*".

**Knowledge and generic word classification (KW/Generic).** Given the ground-truth sentence as the decoder inputs, at each time step, it measures the capability of predicting the correct class (a knowledge graph entity or a generic word) and adopts micro-averaging. The true positive, false negative and false positive are formulated as:

$$\begin{aligned}
\text{TP} &= |\{t \mid \hat{y}_t \in \mathcal{V}, y_t \in \mathcal{V}\}|, \\
\text{FN} &= |\{t \mid \hat{y}_t \in \mathcal{W}, y_t \in \mathcal{V}\}|, \\
\text{FP} &= |\{t \mid \hat{y}_t \in \mathcal{V}, y_t \in \mathcal{W}\}|, \\
\hat{y}_t &\sim P(\cdot \mid y_1 y_2 \dots y_{t-1}).
\end{aligned}$$

**Generated knowledge words (Generated-KW).** Considering the knowledge graph entities in the reference $y = \{y_t\}_{t=1}^{n}$ as positives, in the inference stage, we use the generated knowledge entities to compute true positive, false positive, and true negative, and adopt micro-averaging.

$$\begin{aligned}
\text{TP} &= |\{\hat{y}_t \in \{y_t \in \mathcal{V}\}_{t=1}^{n}, \hat{y}_t \in \mathcal{V}\}_{t=1}^{n}|, \\
\text{FN} &= |\{y_t \notin \{\hat{y}_t \in \mathcal{V}\}_{t=1}^{n}, y_t \in \mathcal{V}\}_{t=1}^{n}|, \\
\text{FP} &= |\{\hat{y}_t \notin \{y_t \in \mathcal{V}\}_{t=1}^{n}, \hat{y}_t \in \mathcal{V}\}_{t=1}^{n}|, \\
\hat{y}_t &\sim P(\cdot \mid \hat{y}_1 \hat{y}_2 \dots \hat{y}_{t-1}).
\end{aligned}$$

For example, after input a sentence "*Where's* **JinXi***?*", if a model generates "*Hi,* **Zhen-Huan***,* **JinXi** *is in* **Yangxin-Palace**." when reference is "**JinXi** *is in* **Yongshou-Palace**.", where bolded words are knowledge entities. Recall is $\frac{1}{2}$ and precision is $\frac{1}{3}$.

#### 2.1.2 Adaptation of Changed Knowledge Graphs

Each knowledge graph is randomly changed by (1) shuffling a batch (*All*), (2) replacing the predicted entities (*Last1*), or (3) replacing the last two steps of paths predicting the generated entities (*Last2*). We have two metrics focusing on the capability of adaptation.

| Metrics | HGZHZ | Friends |
|---|---|---|
| # Dialogues | 1247 | 3092 |
| Total # turns | 17,164 | 57,757 |
| Total # tokens | 462,647 | 838,913 |
| Avg. turns per dialogue | 13.76 | 18.68 |
| Avg. tokens per turn | 26.95 | 14.52 |
| Total unique tokens | 3,624 | 19,762 |
| # KG entities | 174 | 281 |
| # KG relation types | 9 | 7 |
| total # KG entities appear | 46,059 | 176,550 |
| # Dialogues w/ KG entities | 1,166 | 2,373 |
| # turns w/ KG entities | 10,110 | 9,199 |

Table 2: The details of collected DyKgChat.

| | Relation Type (Percentage) |
|---|---|
| **HGZHZ** | IsAliasOf (25%), IsChildOf (5%), IsLoverOf (6%), IsParentOf (5%), IsResidenceOf (16%), IsSiblingOf (2%), IsTitleOf (30%), IsEnemyOf (8%), IsServantOrMasterOf (3%) |
| **Friends** | IsLoverOf (12%), IsWorkplaceOf (2%), IsOccupationOf (8%), IsNameOf (47%), IsRelativeOf (8%), IsFriendOf (4%), IsNicknameOf (19%) |

Table 3: The included relation types in the collect knowledge graphs, and their percentages.

**Change rate.** It measures if the responses are different from the original ones (with original knowledge graphs). The higher rate indicates that the model is more sensitive to a changed knowledge graph. Therefore, the higher rate may not be better, because some changes are worse. The following metric is proposed to deal with the issue, but *change rate* is still reported.

**Accurate change rate.** This measures if the original predicted entities are replaced with the hypothesis set, where this ensures that the updated responses generate knowledge graph entities according to the updated knowledge graphs. (1) In *All*, the hypothesis set is the collection of all entities in the new knowledge graph. (2) In *Last1* and *Last2*, the hypothesis set is the randomly-selected substitutes.

## 3 DyKgChat Corpus

This section introduces the collected DyKgChat corpus for the target knowledge-grounded conversation generation task.

### 3.1 Data Collection

To build a corpus where the knowledge graphs would naturally evolves, we collect TV series conversations, considering that TV series often contain complex relationship evolution, such as

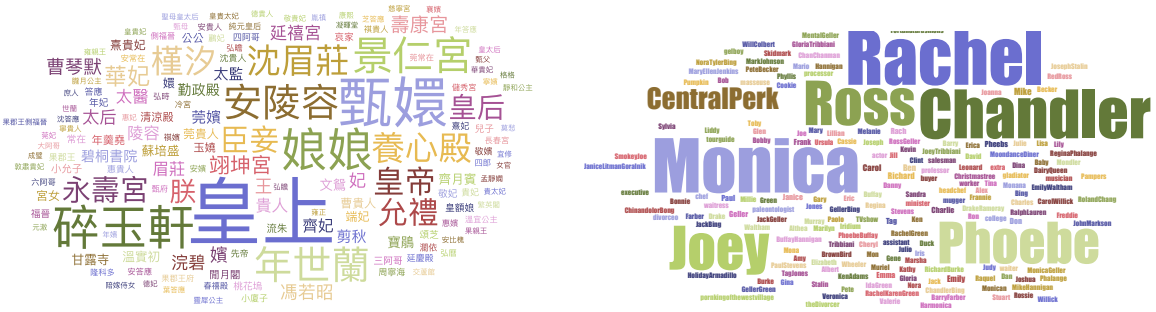


(a) The word cloud of HGZHZ. (b) The word cloud of Friends.

Figure 2: The knowledge entity counts of HGZHZ are more balanced than ones of Friends.

friends, jobs, and residences. We choose TV series with different languages and longer episodes. We download the scripts of a Chinese palace drama "*Hou Gong Zhen Huan Zhuang*" (HGZHZ; with 76 episodes and hundreds of characters) from *Baidu Tieba*, and the scripts of an English sitcom "*Friends*" (with 236 episodes and six main characters)[2]. Their paired knowledge graphs are manually constructed according to their wikis written by fans[3][4]. Noted that the knowledge graph of HGZHZ is mainly built upon the top twenty-five appeared characters.

The datasets are split 5% as validation data and 10% as testing data, where the split is based on multi-turn dialogues and balanced on speakers. The boundaries of dialogues are annotated in the original scripts. The tokenization of HGZHZ considers Chinese characters and knowledge entities; the tokenization of Friends considers space-separated tokens and knowledge entities. The data statistics after preprocessing is detailed in Table 2. The relation types $r \in \mathcal{L}$ of each knowledge graph and their percentages are listed in Table 3, and the knowledge graph entities are plotted as word clouds in Figure 2.

### 3.2 Subgraph Sampling

Due to the excessive labor of building dynamic knowledge graphs aligned with all episodes, we currently collect a fixed knowledge graph $\mathcal{G}$ containing all information that once exists for each TV series. To build the aligned dynamic knowledge graphs, we sample the top-five shortest paths on knowledge graphs from each source to each target, where the sources are knowledge entities in the input message and the scene $\{x_t \in \mathcal{V}\}$, and the targets are knowledge entities in the ground-truth response $\{y_t \in \mathcal{V}\}$. We manually check whether the selected number of shortest paths are able to cover most of the used relational paths. The dynamic knowledge graphs are built based on an ensemble of the following possible subgraphs:

- The sample for each single-turn dialogue.
- The sample for each multi-turn dialogue.
- The manually-annotated subgraph for each period.

While the first rule is adopted for simplicity, the preliminary models should at least work on this type of subgraphs. The subgraphs are defined as the dynamic knowledge graphs $\{\mathcal{K}\}$, which are updated every single-turn dialogue.

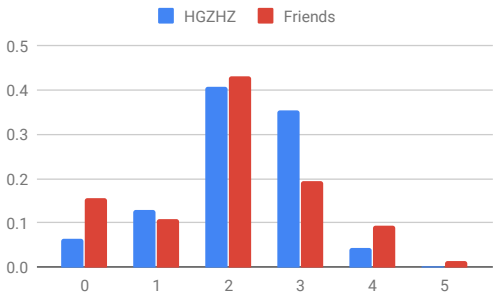


Figure 3: The distribution of lengths of shortest paths.

### 3.3 Data Analysis

**Data imbalance.** As shown in Table 2, the turns with knowledge graph entities are about 58.9% and 15.93% of HGZHZ and Friends respectively. Apparently in Friends, the training data with knowledge graph entities are too small, so fine-

[2] https://github.com/npow/friends-chatbot

[3] https://zh.wikipedia.org/wiki/後宮甄嬛傳_(電視劇)

[4] https://friends.fandom.com/wiki/Friends_Wiki

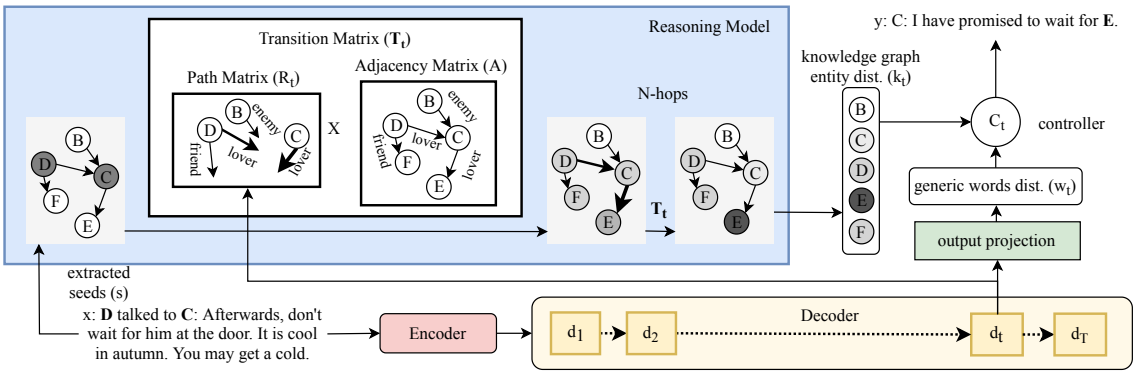


Figure 4: The framework of the proposed model. The node E here is the symbol for the emperor.

tuning on this subset with knowledge graph entities might be required.

**Shortest paths.** The lengths of shortest paths from sources to targets are shown in Figure 3. Most probabilities lie on two and three hops rather than zero and one hop, so key-value extraction based text generative models (Eric et al., 2017; Levy et al., 2017; Qian et al., 2018) are not suitable for this task.On the other hand, multi-hop reasoning might be useful for better retrieving correct knowledge graph entities.

**Dynamics.** The distribution of graph edit distances among dynamic knowledge graphs are $57.24 \pm 24.34$ and $38.16 \pm 15.99$ for HGZHZ and Friends respectively, revealing that the graph dynamics are spread out: some are slightly changed while some are largely changed, which matches our intuition.

## 4 Qadpt: Quick Adaptative Dynamic Knowledge-Grounded Neural Conversation Model

To our best knowledge, no prior work focused on dynamic knowledge-grounded conversation; thus we propose *Qadpt* as the preliminary model. As illustrated in Figure 4, the model is composed of (1) a Seq2Seq model with a controller, which decides to predict knowledge graph entities $k \in \mathcal{V}$ or generic words $w \in \mathcal{W}$, and (2) a reasoning model, which retrieves the relational paths in the knowledge graph.

### 4.1 Sequence-to-Sequence Model

Qadpt is based on a Seq2Seq model (Sutskever et al., 2014; Vinyals and Le, 2015), where the encoder encodes an input message $x$ into a vector $\mathbf{e}(x)$ as the initial state of the decoder. At each time step $t$, the decoder produces a vector $\mathbf{d}_t$ conditioned on the ground-truth or predicted $y_1 y_2 \dots y_{t-1}$. Note that we use gated recurrent unit (GRU) (Cho et al., 2014) in our experiments.

$$\mathbf{e}(x) = \mathbf{GRU}(x_1 x_2 \dots x_m) \quad (1)$$

$$\mathbf{d_t} = \mathbf{GRU}(y_1 y_2 \dots y_{t-1}, \mathbf{e}(x)) \quad (2)$$

Each predicted $\mathbf{d}_t$ is used for three parts: *output projection*, *controller*, and *reasoning*. For output projection, the predicted $\mathbf{d}_t$ is transformed into a distribution $\mathbf{w}_t$ over generic words $\mathcal{W}$ by a projection layer.

### 4.2 Controller

To decide which vocabulary set (knowledge graph entities $\mathcal{V}$ or generic words $\mathcal{W}$) to use, the vector $\mathbf{d}_t$ is transformed to a controller $c_t$, which is a widely-used component (Eric et al., 2017; Zhu et al., 2017; Ke et al., 2018; Zhou et al., 2018b; Xing et al., 2017) similar to copy mechanism (Gu et al., 2016; Merity et al., 2017; He et al., 2017). The controller $c_t$ is the probability of choosing from knowledge graph entities $\mathcal{V}$, while $1 - c_t$ is the probability of choosing from generic words $\mathcal{W}$. Note that we take the controller as a special symbol $KB$ in generic words, so the term $1 - c_t$ is already multiplied to $\mathbf{w}_t$. The controller here can be flexibly replaced with any other model.

$$P(\{KB, \mathcal{W}\} \mid y_1 y_2 \dots y_{t-1}, \mathbf{e}(x)) = \text{softmax}(\phi(\mathbf{d}_t)), \quad (3)$$

$$\mathbf{w}_t = P(\mathcal{W} \mid y_1 y_2 \dots y_{t-1}, \mathbf{e}(x)), \quad (4)$$

$$c_t = P(KB \mid y_1 y_2 \dots y_{t-1}, \mathbf{e}(x)), \quad (5)$$

$$\mathbf{o}_t = \{c_t \mathbf{k}_t; \mathbf{w}_t\}, \quad (6)$$

where $\phi$ is the output projection layer, and $\mathbf{k}_t$ is the predicted distribution over knowledge graph entities $\mathcal{V}$ (detailed in subsection 4.3), and $\mathbf{o}_t$ is the produced distribution over all vocabularies.

### 4.3 Reasoning Model

To ensure that Qadpt can zero-shot adapt to dynamic knowledge graphs, instead of using attention mechanism on graph embeddings (Ghazvininejad et al., 2018; Zhou et al., 2018b), we leverage the concept of multi-hop reasoning (Lao et al., 2011). The reasoning procedure can be divided into two stages: (1) forming a transition matrix and (2) reasoning multiple hops by a Markov chain.

In the first stage, a transition matrix $\mathbf{T}_t$ is viewed as multiplication of a path matrix $\mathbf{R}_t$ and the adjacency matrix $\mathbf{A}$ of a knowledge graph $\mathcal{K}$. The adjacency matrix is a binary matrix indicating if the relations between two entities exist. The path matrix is a linear transformation $\theta$ of $\mathbf{d}_t$, and represents the probability distribution of each head $h \in \mathcal{V}$ choosing each relation type $r \in \mathcal{L}$. Note that a relation type *self-loop* is added.

$$\mathbf{R}_t = \text{softmax}(\theta(\mathbf{d}_t)), \tag{7}$$

$$\mathbf{A}_{i,j,\gamma} = \begin{cases} 1, & (h_i, r_j, t_\gamma) \in \mathcal{K} \\ 0, & (h_i, r_j, t_\gamma) \notin \mathcal{K} \end{cases}, \tag{8}$$

$$\mathbf{T}_t = \mathbf{R}_t \mathbf{A}, \tag{9}$$

where $\mathbf{R}_t \in \mathbb{R}^{|\mathcal{V}|\times 1\times|\mathcal{L}|}$, $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{L}|\times|\mathcal{V}|}$, and $\mathbf{T}_t \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$.

In the second stage, a binary vector $\mathbf{s} \in \mathbb{R}^{|\mathcal{V}|}$ is computed to indicate whether each knowledge entity exists in the input message $x$. First, the vector $\mathbf{s}$ is multiplied by the transition matrix. A new vector $\mathbf{s}^\intercal \mathbf{T}_t$ is then produced to denote the new probability distribution over knowledge entities after one-hop reasoning. After $N$ times reasoning[5], the final probability distribution over knowledge entities is taken as the generated knowledge entity distribution $\mathbf{k}_t$:

$$\mathbf{k}_t = \mathbf{s}^\intercal (\mathbf{T}_t)^N. \tag{10}$$

The loss function is the cross-entropy of the predicted word $\mathbf{o}_t$ and the ground-truth distribution:

$$\mathcal{L}(\psi, \phi, \theta) = -\sum_{t=1}^{n} \log \mathbf{o}_t(y_t), \tag{11}$$

where $\psi$ is the parameters of GRU layers. Compared to prior work, the proposed reasoning approach explicitly models the knowledge reasoning path, so an updated knowledge graphs will definitely change the results without retraining.

[5]We choose $N = 6$ because of the maximum length of shortest paths in Figure 3

### 4.4 Inferring Reasoning Paths

Because this reasoning method is stochastic, we compute the probabilities of the possible reasoning paths by the reasoning model, and infer the one with the largest probability as the retrieved reasoning path.

## 5 Related Work

The proposed task is motivated by prior knowledge-grounded conversation tasks (Ghazvininejad et al., 2018; Zhou et al., 2018b), but further requires the capability to adapt to dynamic knowledge graphs.

### 5.1 Knowledge-Grounded Conversations

The recent knowledge-grounded conversation models (Sordoni et al., 2015; Ghazvininejad et al., 2018; Zhu et al., 2017; Zhou et al., 2018b) generated responses conditioned on conversation history and external knowledge. Ghazvininejad et al. (2018) used memory networks (Weston et al., 2015b,a; Sukhbaatar et al., 2015) to attend on external facts, and added the encoded information to the decoding process.Zhu et al. (2017) added a copy mechanism (Gu et al., 2016; Merity et al., 2017; He et al., 2017) for improving its performance. Zhou et al. (2018b) presented two-level graph attention mechanisms (Veličković et al., 2018) to produce more informative responses.

For knowledge from unstructured texts, Ghazvininejad et al. (2018) used bag-of-word representations and Long et al. (2017) applied a convolutional neural network to encode the whole texts. With structured knowledge graphs, Zhu et al. (2017) and Zhou et al. (2018b) utilized graph embedding methods (e.g., TransE (Bordes et al., 2013)) to encode each triplet.

The above methods generated responses without explicit relationship to each external knowledge triplet. Therefore, when a triplet is added or deleted, it is unknown whether their generated responses can change accordingly. Moon et al. (2019) recently presented a similar concept, walking on the knowledge graph, for response generation. Nonetheless, their purpose is to find explainable path on a large-scaled knowledge graph instead of adaptation with the changed knowledge graphs. Hence, the proposed attention-based graph walker may suffer from the same issue as previous embedding-based methods.

| Model | HGZHZ | | | | | | Friends | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Change Rate | | | Accurate Change Rate | | | Change Rate | | | Accurate Change Rate | | |
| | All | Last1 | Last2 | All | Last1 | Last2 | All | Last1 | Last2 | All | Last1 | Last2 |
| MemNet | 92.98 | 31.78 | 37.46 | 62.19 | 1.17 | 2.92 | 93.27 | 19.22 | 27.53 | 78.23 | 0.91 | 7.66 |
| + multi | 98.69 | 77.87 | 81.96 | **83.82** | 3.40 | 10.74 | 95.31 | 28.09 | 36.65 | **87.28** | 0.69 | 7.63 |
| TAware | 94.38 | 68.33 | 71.86 | **78.88** | 1.95 | 9.26 | 92.93 | 26.52 | 30.96 | **88.07** | 0.35 | 10.63 |
| + multi | 97.74 | 76.68 | 81.00 | **95.30** | 4.03 | 10.75 | 98.31 | 68.87 | 68.22 | **92.29** | 0.87 | 10.09 |
| KAware | 96.91 | 90.89 | 96.91 | 64.80 | 13.06 | 7.22 | 90.93 | 50.92 | 61.08 | 75.57 | 2.77 | 10.00 |
| Qadpt | 95.65 | 77.33 | 78.68 | 59.01 | **66.67** | **16.82** | 92.34 | 38.62 | 36.96 | 81.24 | **30.85** | **16.87** |
| + multi | 99.60 | 83.17 | 87.27 | 56.11 | **61.92** | **18.54** | 98.47 | 48.78 | 63.54 | 86.97 | **26.17** | **17.31** |
| + TAware | 99.02 | 83.14 | 85.59 | 58.82 | **64.12** | **14.90** | 98.45 | 56.77 | 65.25 | 82.52 | **28.34** | **17.68** |

Table 4: The results of change rate and accurate change rate.

| Model | HGZHZ | | | | | Friends | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | KW | KW/Generic | | Generated-KW | | KW | KW/Generic | | Generated-KW | |
| | Acc | Recall | Precision | Recall | Precision | Acc | Recall | Precision | Recall | Precision |
| Seq2Seq | 12.10 | 29.08 | 27.44 | 13.30 | 24.28 | 3.81 | 23.22 | 5.57 | 6.88 | 2.02 |
| MemNet | 22.58 | 39.09 | **100.00** | 39.52 | 67.10 | 22.79 | 37.18 | **100.00** | 46.02 | 53.98 |
| + multi | 35.20 | 54.49 | **100.00** | 60.63 | 83.43 | 34.92 | 47.31 | **100.00** | 60.54 | 69.46 |
| TAware | 50.21 | 44.40 | 35.50 | 49.18 | 76.72 | 62.74 | **50.78** | 22.50 | 57.84 | 62.83 |
| + multi | **59.71** | **68.61** | 28.70 | **70.18** | **85.54** | **72.96** | 42.98 | 25.74 | **71.11** | **77.35** |
| KAware | 20.53 | 40.63 | 36.64 | 24.61 | 43.13 | 13.52 | 30.76 | 28.42 | 15.14 | 18.74 |
| Qadpt | **57.61** | 38.24 | 28.31 | 44.50 | **90.70** | **74.00** | 41.33 | 25.31 | 69.30 | **77.30** |
| + multi | **57.40** | 51.97 | 28.43 | **64.55** | **91.22** | **74.44** | 42.81 | 25.01 | **74.63** | **77.09** |
| + TAware | **56.24** | 53.68 | 31.03 | **63.66** | **88.99** | **73.57** | 47.05 | 25.91 | **74.52** | **78.56** |

Table 5: The results of knowledge graph entities prediction.

### 5.2 Multi-Hop Reasoning

We leverage multi-hop reasoning (Lao et al., 2011) to allow our model to quickly adapt to dynamic knowledge graphs. Recently, prior work used convolutional neural network (Toutanova et al., 2015), recurrent neural network (Neelakantan et al., 2015; Das et al., 2017), and reinforcement learning (Xiong et al., 2017; Das et al., 2018; Chen et al., 2018; Shen et al., 2018) to model multi-hop reasoning on knowledge graphs, and has proved this concept useful in link prediction. These reasoning models, however, have not yet explored on dialogue generation. The proposed model is the first attempt at adapting conversations via a reasoning procedure.

## 6 Experiments

For all models, we use gated recurrent unit (GRU) based Seq2Seq models (Cho et al., 2014; Chung et al., 2014; Vinyals and Le, 2015). Both encoder and decoder for HGZHZ are 256 dimension with 1 layer; ones for Friends are 128 dimension with 1 layer.

We benchmark the task, *dynamic knowledge-grounded dialogue generation*, and corpus *DyKgChat* by providing a detailed comparison between the prior conversational models and our proposed model as the preliminary experiments. We evaluate their capability of quick adaptation by randomized whole, last 1, last 2 reasoning paths as described in Section 2.1.2. We evaluate the produced responses by sentence-level BLEU-2 (Papineni et al., 2002; Liu et al., 2016), perplexity, distinct-n (Li et al., 2016), and our proposed metrics for predicting knowledge entities descrin section 2.1.1.

Because of the significant data imbalance of *Friends*, we first train on whole training data, and then fine-tune the models using the subset containing knowledge entities. Early stopping is adopted in all experiments.

### 6.1 Baselines

We compare our model with prior knowledge-grounded conversation models: the memory network (Ghazvininejad et al., 2018) and knowledge-aware model (KAware) (Zhu et al., 2017; Zhou et al., 2018b). We also leverage the topic-aware model (TAware) (Xing et al., 2017; Wu et al., 2018; Zhou et al., 2018a) by attending on knowledge graphs and using two separate output projection layers (generic words and all knowledge graph entities). In our experiments, MemNet is modified for fair comparison, where the memory pool of MemNet stores TransE embeddings of knowledge triples (Zhou et al., 2018b). The maximum number of the stored triplets are set to the maximum size of all knowledge graphs for

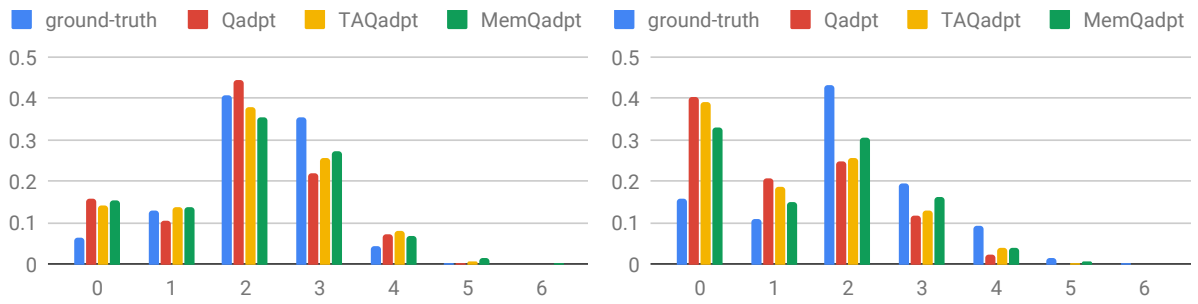


(a) retrieved relation paths distribution of HGZHZ

(b) retrieved relation paths distribution of Friends

Figure 5: The distribution of the lengths of Qadpt inferred relation paths.

| Model | HGZHZ | | | | | | Friends | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | BLEU | PPL | dist-1 | dist-2 | dist-3 | dist-4 | BLEU | PPL | dist-1 | dist-2 | dist-3 | dist-4 |
| Seq2Seq | 14.20 | 94.48 | 0.008 | 0.039 | 0.092 | 0.150 | 15.46 | 73.23 | 0.004 | 0.016 | 0.026 | 0.032 |
| MemNet | 15.73 | 88.29 | 0.012 | 0.062 | 0.150 | 0.240 | 14.61 | 67.58 | 0.005 | **0.023** | 0.040 | 0.049 |
| + multi | 15.88 | 86.76 | 0.010 | 0.058 | 0.138 | 0.224 | 12.97 | **54.67** | **0.006** | 0.022 | 0.032 | 0.036 |
| TAware | **15.97** | **81.54** | 0.013 | 0.068 | 0.153 | 0.223 | 14.78 | 60.61 | 0.002 | 0.007 | 0.013 | 0.016 |
| + multi | 13.34 | 80.48 | **0.022** | 0.122 | 0.239 | 0.304 | 15.74 | 56.67 | 0.003 | 0.011 | 0.019 | 0.023 |
| KAware | 14.14 | 90.11 | 0.011 | 0.061 | 0.135 | 0.198 | 15.70 | 64.70 | 0.002 | 0.009 | 0.017 | 0.021 |
| Qadpt | 14.52 | 88.24 | 0.013 | 0.081 | 0.169 | 0.242 | **17.01** | 68.27 | 0.002 | 0.008 | 0.013 | 0.016 |
| + multi | 15.47 | 86.65 | 0.021 | **0.129** | **0.259** | **0.342** | 14.79 | 66.70 | 0.005 | **0.023** | **0.041** | **0.051** |
| + TAware | 15.05 | 81.75 | 0.022 | 0.123 | 0.246 | 0.332 | 16.85 | 55.46 | 0.003 | 0.012 | 0.020 | 0.024 |

Table 6: The results of responses generation with BLEU, perplexity (PPL), distinct scores (1-gram to 4-gram).

each dataset (176 for hgzhz and 98 for friends). The multi-hop version of MemNet (Weston et al., 2015b) is also implemented (MemNet+multi)[6]. To empirically achieve better performance, we also utilize the attention of MemNet for TAware and KAware. Moreover, we empirically find that multi-hop MemNet deteriorate the performance of KAware (compared to one-hop), while it could enhance the performance of TAware. A standard Seq2Seq model (Vinyals and Le, 2015) is also shown as a baseline without using external knowledge. We also leverage multi-hop MemNet and the attention of TAware to strength Qadpt (+multi and +TAware).

### 6.2 Results

As shown in Table 4, MemNet, TAware and KAware significantly change when the knowledge graphs are largely updated (*All*) and can also achieve good accurate change rate. For them, the more parts updated (*All* $>>$ *Last2* $>$ *Last1*), the more changes and accurate changes. However, when the knowledge graphs are slightly updated (*Last1* and *Last2*), the portion of accurate changes over total changes (e.g., the *Last1* score $1.17/31.78$ for HGZHZ with MemNet model) is significantly low. Among the baselines, KAware has better performance on *Last1*. On the other hand, Qadpt outperforms all baselines when the knowledge graphs slightly change (*Last1* and *Last2*) in terms of accurate change rate. The proportion of accurate changes over total changes also show significantly better performance than the prior models. Figure 5 shows the distribution of lengths of the inferred relation paths for Qadpt models. After combining TAware or MemNet, the distribution becomes more similar to the test data.

Table 5 shows the results of the proposed metrics for correctly predicting knowledge graph entities. On both HGZHZ and Friends, TAware+multi and Qadpt significantly outperform MemNet for KW-Acc and KW/Generic, and MemNet outperforms all other models by KW/Generic precision ($100\%$). This demonstrates that these models can better predict knowledge graph entities, but are slightly worse at making good choices of when to predict generic words (KW/Generic).

Table 6 presents the BLEU-2 scores (as recommended in the prior work (Liu et al., 2016)), perplexity (PPL), and distinct scores. The results show that all models have similar levels of BLEU-2 and PPL, while Qadpt+multi has slightly better distinct scores. The results suggest the same claim as Liu et al. (2016) that BLEU scores are not suit-

[6]Note that *multi-hop* here indicates re-attention on the triplet embeddings of a knowledge graph.

| Model | HGZHZ | | | | | | Friends | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Fluency | | | Information | | | Fluency | | | Information | | |
| | Win | Lose | Kappa | Win | Lose | Kappa | Win | Lose | Kappa | Win | Lose | Kappa |
| Qadpt vs Seq2Seq | **60.0** | 21.3 | .69 | **46.0** | 11.3 | .64 | **55.3** | 23.3 | .55 | **60.0** | 13.3 | .61 |
| Qadpt vs MemNet | **49.3** | 19.3 | .78 | **34.0** | 12.7 | .73 | 35.3 | **38.7** | .62 | 17.3 | **22.7** | .59 |
| Qadpt vs TAware | **48.7** | 18.0 | .72 | **30.0** | 13.3 | .70 | **42.7** | 32.7 | .58 | **20.7** | 18.0 | .62 |
| Qadpt vs KAware | **59.3** | 14.0 | .71 | **58.7** | 8.0 | .78 | **44.7** | 28.7 | .62 | **44.7** | 13.3 | .68 |

Table 7: The results of human evaluation.

able for dialogue generation.

## 7 Human Evaluation

To perform human evaluation, we randomly select examples from the knowledge-related outputs of all models, because it is difficult for human to distinguish which generic response is better. We recruit fifteen annotators to judge the results. Each annotator was randomly assigned with 20 examples, and was guided to rank the results of five models: Seq2Seq, MemNet, TAware, KAware, and Qadpt. They were asked to rank all results according to two criteria: (1) fluency and (2) information. *Fluency* measures which output is more proper as a response to a given input message. *Information* measures which output contains more correct information (in terms of knowledge words here) according to a given input message and a referred response. The evaluation results are classified into "win", "tie", and "lose" for comparison.

The human evaluation results and the annotator agreement in the form of Cohen's kappa (Cohen, 1960) are reported in Table 7. According to a magnitude guideline (Landis and Koch, 1977), most agreements are substantial (0.6-0.8), while some agreements of Friends are moderate (0.4-0.6). In most cases of Table 7, Qadpt outperforms other four models. However, in Friends, Qadpt, MemNet, and TAware tie closely. The reason might be the lower agreements of Friends, or only the similar trend with automatic evaluation metrics. There are two extra spots. First, Qadpt wins MemNet and TAware less times than winning Seq2Seq and KAware, which aligns with Table 5 and Table 6. Second, Qadpt wins baselines more often by fluency than by information, and much more ties happen in the infomation fields than the fluency fields. This is probably due to the selection of knowledge-contained examples. Hence there is no much difference when seeing the information amount of models. Overall, the human evaluation results can be considered as reference because of the substantial agreement among annotators and the similar trend with automatic evaluation.

## 8 Discussion

The results demonstrate that MemNet, TAware and Qadpt generally perform better than than the other two baselines, and they excel at different aspects. MemNet can successfully incorporate knowledge graphs and generate sentences with both appropriate knowledge entities and generic words. In contrast, TAware and Qadpt predict more correct knowledge entities but tend to diminish generic words.

For the scenario of zero-shot adaptation, MemNet and TAware show their ability to update responses when the knowledge graphs are largely changed. On the other hand, Qadpt is better to capture minor dynamic changes ( *Last1* and *Last2*) and updates the responses according to the new knowledge graphs. Some examples are given in Appendix. This demonstrates that MemNet and TAware attend on the whole graph instead of focusing on the most influential part.

## 9 Conclusion

This paper presents a new task, *dynamic knowledge-grounded conversation generation*, and a new dataset *DyKgChat* for evaluation. The dataset is currently provided with a Chinese and an English TV series as well as their correspondent knowledge graphs. This paper also benchmarks the task and dataset by proposing automatic evaluation metrics and baseline models, which can motivate the future research directions.

## Acknowledgements

We would like to thank reviewers for their insightful comments. This work was financially supported from the Young Scholar Fellowship Program by Ministry of Science and Technology (MOST) in Taiwan, under Grant 108-2636-E002-003.

## References


Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

Wenhu Chen, Wenhan Xiong, Xifeng Yan, and William Yang Wang. 2018. Variational knowledge graph reasoning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 1823–1832.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *Presented in NIPS 2014 Deep Learning and Representation Learning Workshop*.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2018. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. *International Conference on Learning Representations*.

Rajarshi Das, Tsendsuren Munkhdalai, Xingdi Yuan, Adam Trischler, and Andrew McCallum. 2019. Building dynamic knowledge graphs from text using machine reading comprehension. In *International Conference on Learning Representations*.

Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. 2017. Chains of reasoning over entities, relations, and text using recurrent neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 132–141.

Mihail Eric, Lakshmi Krishnan, Francois Charette, and Christopher D Manning. 2017. Key-value retrieval networks for task-oriented dialogue. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 37–49.

Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen-tau Yih, and Michel Galley. 2018. A knowledge-grounded neural conversation model. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1631–1640.

Yanchao Hao, Yuanzhe Zhang, Kang Liu, Shizhu He, Zhanyi Liu, Hua Wu, and Jun Zhao. 2017. An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 221–231.

Shizhu He, Cao Liu, Kang Liu, and Jun Zhao. 2017. Generating natural answers by incorporating copying and retrieving mechanisms in sequence-to-sequence learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 199–208.

Pei Ke, Jian Guan, Minlie Huang, and Xiaoyan Zhu. 2018. Generating informative responses with controlled sentence function. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1499–1508.

J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174.

Ni Lao, Tom Mitchell, and William W Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 529–539. Association for Computational Linguistics.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. *CoNLL 2017*, page 333.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of NAACL-HLT*, pages 110–119.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2122–2132.

Yinong Long, Jianan Wang, Zhen Xu, Zongsheng Wang, Baoxun Wang, and Zhuoran Wang. 2017. A knowledge enhanced generative conversational service agent. In *Proceedings of the 6th Dialog System Technology Challenges (DSTC6) Workshop*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. *Proceedings of the 2017 International Conference on Learning Representations*.

Seungwhan Moon, Pararth Shah, Anuj Kumar, and Rajen Subba. 2019. Opendialkg: Explainable conversational reasoning with attention-based walks over knowledge graphs. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 845–854.

Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 2015. Compositional vector space models for knowledge base completion. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 156–166.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Qiao Qian, Minlie Huang, Haizhou Zhao, Jingfang Xu, and Xiaoyan Zhu. 2018. Assigning personality/profile to a chatting machine for coherent conversation generation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 4279–4285. AAAI Press.

Yelong Shen, Jianshu Chen, Po-Sen Huang, Yuqing Guo, and Jianfeng Gao. 2018. Reinforcewalk: Learning to walk in graph with monte carlo tree search. *International Conference on Learning Representations*.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 196–205.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen. 2018. Open domain question answering using early fusion of knowledge bases and text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4231–4242.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. *International Conference on Learning Representation*.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *ICML Deep Learning Workshop 2015*.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015a. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015b. Memory networks. *International Conference on Learning Representations*.

Yu Wu, Wei Wu, Dejian Yang, Can Xu, and Zhoujun Li. 2018. Neural response generation with dynamic vocabularies. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2017. Topic aware neural response generation. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 564–573.

Jun Yin, Xin Jiang, Zhengdong Lu, Lifeng Shang, Hang Li, and Xiaoming Li. 2016. Neural generative question answering. In *Proceedings of the Workshop on Human-Computer Question Answering*, pages 36–42.

Hao Zhou, Minlie Huang, Tianyang Zhang, Xiaoyan Zhu, and Bing Liu. 2018a. Emotional chatting machine: Emotional conversation generation with internal and external memory. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Hao Zhou, Tom Young, Minlie Huang, Haizhou Zhao, Jingfang Xu, and Xiaoyan Zhu. 2018b. Commonsense knowledge aware conversation generation with graph attention. In *IJCAI*, pages 4623–4629.

Wenya Zhu, Kaixiang Mo, Yu Zhang, Zhangbin Zhu, Xuezheng Peng, and Qiang Yang. 2017. Flexible end-to-end dialogue system for knowledge grounded conversation. *arXiv preprint arXiv:1709.04264*.

# Retrieval-guided Dialogue Response Generation via a Matching-to-Generation Framework

**Deng Cai**[1]* **Yan Wang**[2]* **Wei Bi**[2] **Zhaopeng Tu**[2] **Xiaojiang Liu**[2] **Shuming Shi**[2]
[1]The Chinese University of Hong Kong, [2]Tencent AI Lab
thisisjcykcd@gmail.com
{brandenwang,victoriabi,zptu,kieranliu,shumingshi}@tencent.com

## Abstract

End-to-end sequence generation is a popular technique for developing open domain dialogue systems, though they suffer from the *safe response problem*. Researchers have attempted to tackle this problem by incorporating generative models with the returns of retrieval systems. Recently, a skeleton-then-response framework has been shown promising results for this task. Nevertheless, how to precisely extract a skeleton and how to effectively train a retrieval-guided response generator are still challenging. This paper presents a novel framework in which the skeleton extraction is made by an interpretable matching model and the following skeleton-guided response generation is accomplished by a separately trained generator. Extensive experiments demonstrate the effectiveness of our model designs.

## 1 Introduction

Sequence-to-sequence (seq2seq) neural models (Shang et al., 2015; Vinyals and Le, 2015; Sordoni et al., 2015; Serban et al., 2016; Li et al., 2016a) have been popular for single-turn dialogue response generation. However, many of the generated responses (e.g., "I don't know" and "I think so") appear to be generic and dull (safe response problem) (Li et al., 2016a). This problem is avoided in traditional retrieval systems (Ji et al., 2014; Hu et al., 2014) by preceding the selection of informative and engaging responses.

It is of interest to benefit from both the generalization capacity of the seq2seq models and the information richness of the retrieved responses. Following the standard encoder-decoder framework, early attempts have either used an extra encoder for the retrieved response (Song et al., 2016; Pandey et al., 2018; Wu et al., 2019) or a unified encoder for the concatenation of the query and the retrieved response (Weston et al., 2018). To prevent the inflow of erroneous information, Cai et al. (2019) proposed a general framework that first extracts a skeleton from the retrieved response and then generates the response based on the extracted skeleton. Despite their differences, a common issue is that the generation model easily learns to ignore the retrieved response entirely and collapses to a vanilla seq2seq model. As shown in Figure 1, this happens with improper training instances. Given the large space of possible responses, it happens frequently that a retrieved response (extracted skeleton) is suitable for responding to the query, but inconsistent with the current target response.[1] The generation model is thus mistakenly led to be inclined to neglect the retrieval.

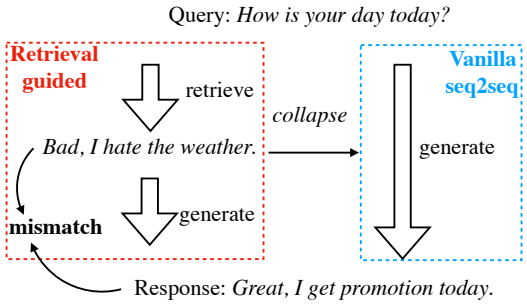


Figure 1: The common problem for training a retrieval-guided generation model in previous work. The model is forced to neglect the retrieved response even though it is a proper response, due to the mismatch between the retrieved response and the target response.

To address the above problem, we present the matching-to-generation method, a more flexible

* This work was mainly done when Deng Cai was an intern at Tencent AI Lab. Yan Wang is the corresponding author.

[1]Previous studies (Weston et al., 2018; Wu et al., 2019; Cai et al., 2019) alleviated the problems by putting hard constraints on the data, which, however, greatly reduces the amount of usable data (e.g., Cai et al. (2019) required the Jaccard distance between the retrieved response and the target response should be in the range [0.3, 0.7]).



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1866–1875, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

framework for retrieval-guided response generation. This framework consists of an interpretable matching model for skeleton extraction and a skeleton-guided response generator for response generation. One novel characteristic of our proposed framework is that the training of the skeleton extractor (i.e., the matching model) and the response generator is decoupled, yet they work cooperatively under the help of a retrieval system. Figure 2 depicts the training and inference procedures of our framework. During training, the skeleton-guided response generator is trained in a similar manner as the denoising autoencoder (Vincent et al., 2008), where the model learns to recover an input pattern that is partially corrupted. Specifically, we employ a random mechanism for generating the skeletons used for training. The generated skeletons are extracted from their corresponding responses with some deliberate disturbance. In this way, we circumvent the aforementioned problem of improper training instances in previous work. Meanwhile, the random mechanism also simulates the actual inference environment where the quality of the input skeleton varies among different queries due to the instability of the retrieval system and the skeleton extractor. The diversity of the training skeletons helps produce a robust response generator that is capable of handling different situations.

Note the separation of the training of skeleton extraction and response generation requires an additional training objective for the skeleton extractor. Given there is no explicit response skeleton in general query-response pairs for training, we propose to use an interpretable matching model for *matching skeleton* extraction. We consider that the matching skeleton for a given query-response pair should be the sub-sequence of the response that is particularly useful in matching the query. The designed interpretable matching model is able to reveal the fine-grained matching scores at token-level whereas it is trained by ordinary query-response pairs.

Experiments show that our method significantly improves the informativeness of the generated responses as well as their relevance to the corresponding queries. In addition, we conduct extensive ablation studies to quantify the improvement from different model designs.

To summarize, our contributions are as follows:

- We propose a flexible framework for retrieval-guided dialogue response generation. The training of our approach is independent of the underlying retrieval system.
- We propose an interpretable matching model for matching skeleton extraction.
- We propose to train a skeleton-guided response generator that can handle skeletons with different qualities.

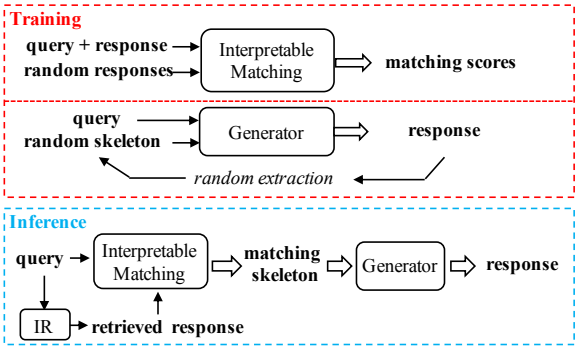


Figure 2: Flow charts during training and inference.

## 2 Models

The whole framework consists of two components: an interpretable matching model and a skeleton-guided response generator. During inference, the matching model is used to derive a matching skeleton by explicitly selecting a subsequence of a retrieved response. The response generator then takes the generated skeleton as an additional input and makes necessary editions to obtain a complete and appropriate response.

### 2.1 Interpretable Matching Model

The goal of the interpretable matching model is to reveal token-level matching information between a query-response pair thus a matching skeleton can be derived from the response. However, the training of the matching model does not rely on such fine-grained annotations. Instead, it is trained to estimate the sequence-level quality of a response for a given query, as an ordinary query-response matching model. The key is that the sequence-level matching score can be decomposed into a set of token-level scores, which will be illustrated later.

The overall architecture of our matching model is illustrated in Figure 3. It consists of two encoders, one for the query and one for the re-

sponse. Both encoders are based on the Transformer architecture (Vaswani et al., 2017). For a query $q = (q_1, q_2, \ldots, q_n)$ and a response $r = (r_1, r_2, \ldots, r_m)$, where $n$ and $m$ are the query length and the response length respectively, we first insert a special token at the beginning of each input sequence. The transformer encoders results in two sequences of hidden state vectors $\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_n$ and $\mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_m$, where $\mathbf{q}_0$ and $\mathbf{r}_0$ are considered as the aggregate summary for the query and the response respectively.

We then use self-attention mechanism for acquiring the final query representation $\mathbf{x}_q$ and the final response representation $\mathbf{x}_r$. For instance, to compute the response representation $\mathbf{x}_r$, we use the sequence-level summary $\mathbf{r}_0$ for weighting the different parts of the input response. First, the sequence-level summary $\mathbf{r}_0$ is projected to another vector space by a linear transformation:

$$\mathbf{r}^w = W^w \mathbf{r}_0 + b^w$$

where $\mathbf{r}^w$ is the weight vector, and $W^w$ and $b^w$ are learnable parameters. The attention score $\omega_i$ of the $i$-th token in the response is then computed as a dot-product between the weight vector $\mathbf{r}^w$ and the token representation $\mathbf{r}_i$:

$$\omega_i = \frac{\exp(\mathbf{r}^w \cdot \mathbf{r}_i)}{\sum_{k=1}^{m} \exp(\mathbf{r}^w \cdot \mathbf{r}_k)}$$

The response representation $\mathbf{x}_r$ is calculated as the weighted sum of the Transformer encoder outputs as well as their initial vector representations (i.e., the sum of tokens and position embeddings)[2]:

$$\mathbf{x}_r = \sum_{k=1}^{m} \omega_i (\mathbf{r}_i + \mathbf{e}_{r_i})$$

The self-attention mechanism for the query has the identical architecture but uses a different set of parameters. Finally, the pair-wise score is calculated by a bilinear function of $\mathbf{x}_q$ and $\mathbf{x}_r$:

$$s(q, r) = {\mathbf{x}_q}^T W^s \mathbf{x}_r$$

where $W^s$ is a trainable parameter. The above equation can be rewritten by a decomposition of the ingredients of $\mathbf{x}_r$:

$$\begin{aligned} s(q, r) &= {\mathbf{x}_q}^T W^s \mathbf{x}_r \\ &= {\mathbf{x}_q}^T W^s \sum_{k=1}^{m} \omega_k (\mathbf{r}_k + \mathbf{e}_{r_k}) \\ &= \sum_{k=1}^{m} \omega_k {\mathbf{x}_q}^T W^s (\mathbf{r}_k + \mathbf{e}_{r_k}) \end{aligned}$$

Let $s_k = {\mathbf{x}_q}^T W^s (\mathbf{r}_k + \mathbf{e}_{r_k})$, we arrive at:

$$s(q, r) = \sum_{k=1}^{m} \omega_k s_k$$

Note that for a given query, $\omega_k$ and $s_k$ are functions of the response $r$ and the position index $k$ only. According to the formulation, we see that $s_k$ and $\omega_k$ are largely impacted by the local information at $r_k$. Therefore, $s_k$, $\omega_k$ can be interpreted as the *local matching score* and the *local importance*, respectively, followed by that $s(q, r)$ is a weighted sum of all local scores.

Once the matching model has been well-trained, we can use $\omega_k$ and $s_k$ to identify the most informative and relevant parts of a retrieved response. In experiments, we show a simple heuristic rule can effectively pick up the skeletons.

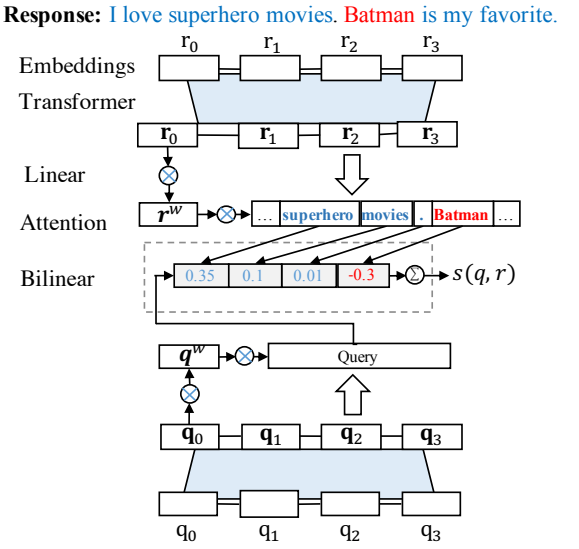


Figure 3: Architecture of the interpretable matching model. The query and response are encoded separately. We compute all attention values for each token simultaneously. The final score is computed by a bilinear function between query representation and response representation. Different colors indicate the different values of the local scores.

[2]We found that adding the initial vector representations $\mathbf{e}_{r_i}$ to be critical in keeping the weighted elements reflect the corresponding local information. Without this operation, the Transformer encoder outputs $\mathbf{r}_i$ tends to be constant regardless of the position $i$, which indicates that the point information about a specific input part is overwhelmed by the global information.

### 2.2 Skeleton-guided Response Generator

The skeleton-guided response generator is devised for generating a fluent and adequate response based on the current query and an input skeleton. To ensure the skeleton-guided response generator does make use of the input skeleton, we extract the training skeleton from the ground-truth response by some randomized strategies. To prevent the response generator from mindlessly copying, we deliberately vary the length and the quality of the training skeletons to create a diverse set of training instances. We note the response generator behaves like a denoising autoencoder (Vincent et al., 2008) with an extra input, i.e. the query. In this way, we learn a robust response generator that is compatible with different types of skeletons.

Specifically, for any golden query-response pair $(q, r)$, we randomly generate a training skeleton through the following procedures.[3]

- All stop words in $r$ are masked in advance. The rest tokens are masked at a mask rate $\gamma$. 90% of the time, $\gamma$ is set to 0.7. 10% of the time, $\gamma$ is uniformly sampled in the range of $[0, 1]$.
- Instead of always replacing the masked token with a special placeholder token, 20% of time, we replace the token with a random word uniformly sampled from the total vocabulary.
- At a chance of 10%, we randomly shuffle the word order in the training skeleton.

The response generator consists of one encoder for the query $q$, one encoder for the skeleton $s$ and one decoder for the response $r$, all implemented by LSTM networks (Hochreiter and Schmidhuber, 1997). The decoder interacts with the two encoders through two separate attention mechanisms accordingly.

### 2.3 Training

The matching model and the response generator are trained separately. Previous studies (Shang et al., 2018; Tao et al., 2018; Mou et al., 2016) formulated the training of matching models as binary classification learning, where negative sampling is used to free human annotation. Specifically, for query $q$ and golden response $r^+$, a negative response $r^-$ can be randomly sampled from other responses in the training set. We extend the binary classification setting into a learning-to-rank fashion for improved performance. Concretely, at each training mini-batch, we randomly sample $M$ query-response pairs. Then we compute the matching scores between all combinations of queries and responses in the mini-batch. As a result, all these scores form a scoring matrix $\mathbf{S} \in \mathbb{R}^{M \times M}$, where $\mathbf{S}_{ij}$ is the score between the $i$-th query and the $j$-th response.

Inspired by Henderson et al. (2017); Lin et al. (2017), we use softmax to compute the ranking scores for candidate responses. Intuitively, for each query, the matching model should give the highest score to the golden response over other $M - 1$ responses (i.e., always rank the golden response at the first place). Thus, we define the training loss as

$$L(\theta) = -\sum_{k=1}^{M} \log \text{softmax}(\mathbf{S}_{k:})_k \qquad (1)$$

where $\mathbf{S}_{k:}$ is $k$-th row of $\mathbf{S}$. Label smoothing (Szegedy et al., 2016) of value $\epsilon_{ls} = 0.1$ is used to improve the performance. Note although there are $M \times M$ scores to compute, each query and each response only needs to be modeled once thanks to the independent encoding of $\mathbf{x}_q$ and $\mathbf{x}_r$. Experiments show that the ranking scheme outperforms the binary classification scheme by a large margin when evaluated by hits@1 metric with 127 randomly sampled responses.

The response generator is trained by the standard maximum likelihood estimate.

### 2.4 Discussion

We note that the most related work is Cai et al. (2019) that also employs a pipeline approach for skeleton extraction and response generation. However, there are some major distinctions in our framework. First, their skeleton extractor is pre-trained by the lexical overlap between the retrieved response and the golden response. However, it is not a proper objective since the mismatch between the retrieved response and the golden response does not imply a mismatch to the target query. In contrast, our interpretable matching model allows extracting a more precise skeleton in semantics. Second, the training of the response generator relies on the output of the learned skeleton extractor, which is by no means aimed for generating the current response, causing a trained

[3] We did not tune much the hyper-parameters in the random strategies as the given setting just works fine.

generator to severely ignore the skeleton. Differently, our response generator is trained with target-specific skeletons.

## 3 Experiments

### 3.1 Dataset and Evaluation Metrics

We use a single-turn conversation dataset collected from popular Chinese social websites such as Douban and Weibo.[4] The dataset contains about six millions query-response pairs. Throughout all experiments, the retrieval system we adopted is a publicly available chatbot API.[5] The related resources can be found at https://github.com/jcyk/seqgen.

It has been argued that existing automatic metrics such as BLEU and METEOR cannot authentically reflect the quality of dialog response. Thus, the main evaluation is done by human annotators. Specifically, we evaluate the quality of a response on three criteria: informativeness, relevance, and fluency. Each aspect is rated on a five-point scale, where 1, 3 and 5 indicate unacceptable, moderate and excellent performance respectively. 2 and 4 are used by annotators in unsure cases. A set of 300 different query samples are used for evaluation. We recruit five experienced annotators and take the average score among them. Besides, we also use *dist-1*/*dist-2* (Li et al., 2016a) to examine a model's ability for generating diverse responses. which is the number of distinct unigrams/bi-grams divided by the total number.

### 3.2 Compared Methods

To show the effectiveness of our proposed methods, we compare it with the following methods.

- *Retrieval* The underlying retrieval system used in our experiments.
- *Seq2Seq* The basic Seq2Seq model (Bahdanau et al., 2014; Luong et al., 2015) that only takes the query as input.
- *Seq2Seq-MMI* A variant of the basic Seq2Seq model that uses Maximum Mutual Information (MMI) for filtering out generic responses (Li et al., 2016a). Concretely, a response-to-query Seq2Seq model is trained and used to rerank the outputs of the top-100 responses of *Seq2Seq*.
- *RetrieveNRefine*$^{++}$ The best performing model used in Weston et al. (2018), which appends the retrieved response to the query in a basic Seq2Seq model.[6] The model's output will be overwritten by the retrieved response once they have a large word overlap (Jaccard distance $> 0.6$).
- *EditVec* The model proposed in Wu et al. (2019). In addition to the retrieved response, the lexical difference (insert words and delete words) between the query and the retrieved query is also encoded (in a so-called edit vector) to feed the decoder.
- *Skeleton-Lex* The best method presented in Cai et al. (2019). We refer to it as *Skeleton-Lex* because its skeleton extractor is pre-trained by the lexical overlap between the retrieved response and the golden response.

### 3.3 Implementation Details

For encoders and decoders in all above baselines, they are implemented by LSTM networks (bidirectional for encoders and unidirectional for decoders) (Hochreiter and Schmidhuber, 1997) with the number of layers and hidden size equal to 2 and 500. The word embeddings are randomly initialized, of which the dimension is 300. Our response generator follows the same settings. The skeleton extractor is implemented by 2-layer Transformer encoder (Vaswani et al., 2017), of which the number of heads and hidden size is 8 and 512. In experiments, we use a simple heuristic rule for extracting skeletons. First, we remove all words with a negative local score $s_k$. Then we compute the average score of the rest part. Lastly, words with a score below the average are also removed.

As the retrieval system can potentially return a large set of results, we allow retrieval-guided generation models (both baselines and ours) make use of the top-10 retrieved results both in training and testing. Therefore, during testing, 10 responses are generated for each query. They are then ranked by the matching model proposed in Section 2.1 and the highest-scored one is used for evaluation.[7]

[4] Douban https://www.douban.com/ and Weibo https://www.weibo.com/

[5] https://ai.qq.com/product/nlpchat.shtml

[6] We take a slightly different approach by treating the two inputs as independent sources to do attention over, as it is also suggested by the original authors.

[7] We train another matching model (not the one we used as skeleton extractor).

| Models | Informativeness | Relevance | Fluency | Dist-1(%) | Dist-2(%) |
|---|---|---|---|---|---|
| *Retrieval* | 2.65 (0.90)† | 2.58 (0.86) | 2.96 (0.72) | **49.10** | **84.19** |
| *Seq2Seq* | 2.01 (0.65) | 2.58 (0.53) | 2.71 (0.43) | 30.38 | 54.52 |
| *Seq2Seq-MMI* | 2.47 (0.70) | 2.79 (0.67) | 2.99 (0.61) | 30.98 | 62.85 |
| *RetrieveNRefine*$^{++}$ | 2.30 (0.79) | 2.62 (0.63) | 2.82 (0.51) | 29.83 | 61.07 |
| *EditVec* | 2.29 (0.61) | 2.62 (0.60) | 2.83 (0.47) | 35.30 | 67.57 |
| *Skeleton-Lex* | 2.45 (0.61) | 2.80 (0.56) | 2.99 (0.46) | 25.70 | 56.61 |
| Ours | **2.69** (0.87) | **3.11** (0.55) | **3.20** (0.55) | **49.01** | **80.36** |

Table 1: Human scores on response quality, depicted in three aspects: informativeness, relevance, and fluency, with standard deviation in parentheses. Sign tests on human scores show that our method is significantly better than all other methods with p-value $<0.01$ with the only exception marked by †. We also present dist-1 and dist-2 for diversity assessment.

### 3.4 Main Results

The evaluation results are given in Table 1. They show that our method outperforms all baseline methods in all three human evaluation aspects. Surprisingly, the informativeness score is even slightly better than the underlying retrieval system, which indicates the retrieved information has been effectively utilized. It can also be verified by the automatic metrics (dist-1 and dist-2), our generation model is the only one that achieves close performance to that of the retrieval system.

For the relevance score, the retrieval-independent *Seq2Seq-MMI* establishes a strong baseline. As for retrieval-guided generation, skeleton-guided methods are better than those who use completely retrieved responses, which confirms that the introduction of the intermediate skeleton prevents the inflow of irrelevant information. Furthermore, our method advances the performance of *Skeleton-Lex* by a large margin, which partly demonstrates that the skeletons extracted by our deep semantic matching model are more precise.

For fluency, our method also achieves much better performance than all baseline methods. We attribute the remarkable improvement to the unique training fashion for our response generator. During training, our response generator receives a diverse set of probably noisy skeletons, which impels it to learn the error correction and better language organization.

### 3.5 More Analysis

To further quantify the contributions made by different components in our model, we turn to ablation tests. Generally, we try to substitute each component of our model with other possible counterparts. The detailed analysis is given below.

| Skeletons | Info. | Relevance | Fluency |
|---|---|---|---|
| Ours | 2.69 | 3.11 | 3.20 |
| *Lexical* | 2.62 | 2.92 | 3.05 |
| *keywords* | 2.56 | 2.90 | 3.03 |
| *PMI* | 2.53 | 2.88 | 3.02 |

Table 2: Ablation study on the skeleton extractor. Info. is short for informativeness in this and following tables.

First, we would like to see if the matching skeleton extracted by our interpretable architecture is beneficial. In order to examine this, we replace our skeleton extractor by several different approaches.

- *Lexical* We use the skeletons extracted by the skeleton extractor in *Skeleton-Lex*.
- *PMI* Point mutual information (PMI) is a popular measure used for finding collocations and associations between words. We compute the PMI between query word and response word through statistics on the training corpus. For a word in the retrieved response, we score it by the sum of the PMIs between it and all words in the target query. Words with the highest scores form the skeleton.
- *Keywords* We generate a skeleton by preserving the most informative words in the retrieved response. Specifically, the words with the highest TF-IDF values are preserved and the others are removed.

For a fair comparison, the lengths of the skeletons (the number of preserving words) generated by *PMI* and *Keywords* are kept as the same with the one generated by our skeleton extractor. In this sense, the comparison with the last two approaches shows how good the token-level score $s_k$

| Model Variants | Info. | Rel. | Flu. |
|---|---|---|---|
| Ours | 2.69 | 3.11 | 3.20 |
| Matching + C19's RG | 2.46 | 2.72 | 2.89 |
| C19's SE + Generator | 2.62 | 2.92 | 3.05 |
| Cai et al. (2019) | 2.45 | 2.80 | 2.99 |

Table 3: A systematic comparison of different component combinations, where C19's SE and C19's RG are short for Cai et al. (2019)'s skeleton extractor and response generator respectively.

is in selecting the most useful words, compared to statistical values such as TF-IDF and PMI.

The result is in shown Table 2. As seen, both two learnable skeleton extractors give better results than non-parametric methods, indicating the task of skeleton extraction is non-trivial and requires deep reasoning. Our semantic-inspired model is far ahead of others in all aspects, while *Lexical* only has a notable improvement in informativeness compared to statistical methods. This suggests that the skeleton extracted by *Lexical* has a relatively low precision, leading to moderate relevance. In addition, it might be a little bit surprising to see that *PMI* and *keywords* give almost the same performance on all three metrics, telling a given query is not that necessary. However, we found lots of skeletons proposed by *PMI* are identical to those of *keywords*. We attribute it to that the keywords in $r$ are often also the keywords in differentiating its context.

To test the ability of the skeleton-based response generator, we use the existing alternative trained in *Skeleton-Lex*, which also takes a skeleton and input query as input. The result displays on how good our response generator is at transforming a skeleton to a proper response.

The results are shown in the first block of Table 3. We see a clear decline in performance after switching to the response generator of *Skeleton-Lex*. We conjecture that the big gap is caused by that their response generator is trained with the output of their skeleton extractor, thus it is highly biased to their specific skeleton extractor and cannot work well with others. This result motivates us to present a systematic examination of different component combinations, as shown by the full content of Table 3. As seen, our response generator is less sensitive to the underlying skeletons. The result of combining our response generator and their skeleton extractor is slightly below our full mode, but even higher than their full model.

| Variants | Info. | Relevance | Fluency |
|---|---|---|---|
| ours | 2.69 | 3.11 | 3.20 |
| *Seq2Seq-MMI* | 2.64 | 3.02 | 3.14 |

Table 4: Ablation study on the ranker.

Lastly, we investigate the improvement brought by our ranker. To this end, we replace our ranker with the *seq2seq-MMI* model (use the sum of the forward and backward generation probability of re-ranking). The results are shown in Table 4. As we can see, our matching model shows superior capability in selecting the best response, especially in terms of relevance.

### 3.6 Case Study

We also present some examples generated by different methods in Table 5. In the first case, the retrieved utterance is very specific with elaborated details. However, it is not a reasonable response due to the sudden topic drift. While other methods directly ignore the retrieved response, our method makes an informative and fluent response by using some useful details. In the second case, our skeleton extractor again successfully locates the most informative and relevant parts of the retrieved response, the response generator then chains them together and generate a meaningful response. In the third case, the retrieved response is also good but not fluent. The skeleton extracted by our skeleton extractor contains a useless word, yet our response generator can ignore the mistake and generate a fluent and comforting response. We can also see another retrieval-guided method (*EditVec*) also attempt to use the retrieved response but capture the wrong parts.

## 4 Related Work

Open domain dialog system has been a long goal for the NLP community since ELIZA (Weizenbaum, 1966). Early data-driven work uses information retrieval techniques (Ji et al., 2014; Hu et al., 2014). Recently, end-to-end neural sequence generation (Vinyals and Le, 2015; Serban et al., 2016; Li et al., 2016a; Sordoni et al., 2015) has attracted the most attention. A major issue of such end-to-end sequence generation method is the *safe response problem*. The generated responses tend to be universal and unengaging (e.g., "I don't know", "I think so" etc.). One of the rea-

| | |
|---|---|
| Query | 喜欢长头发的女生 I like girls with long hair |
| Retrieved response | 可以 把 头发 后面 汤 大 卷 , 前面 是 直 刘海 , 这样 看 的 很 可爱<br>You can try perming the hair , hair with curly ends and straight bangs is very cute. |
| Ours | 我也喜欢长**头发**的女生,**卷**了**直刘海**, 头发就不好看了 I also like girl with long **hair**, I think once the **straight bangs** are **curled**, it doesn't look good any more. |
| *Seq2Seq-MMI* | 我也喜欢长头发的女生 I also like girls with long hair. |
| *RetrieveNRefine*$^{++}$ | 我喜欢长头发的女生 I like girls with long hair. |
| *EditVec* | 我也喜欢长头发的女生 I also like girls with long hair. |
| *Skeleton-Lex* | 我喜欢长头发的女生 I like girls with long hair. |
| Query | 我提现怎么还没到 Why hasn't my withdrawal arrived yet? |
| Retrieved response | 就是 提现 到 你 绑定 的 银行卡 上 了 Withdrawal is in the bound bank card . |
| Ours | 你**提现**的时候**绑定**了**银行卡**吗 Have you **bounded** the **bank card** when **withdrawing** ? |
| *Seq2Seq-MMI* | 我提现也到了 My withdrawal has arrived too. |
| *RetrieveNRefine*$^{++}$ | 你要支付宝干嘛 Why do you need Alipay? |
| *EditVec* | 你是提现的吗 Do you want to withdraw? |
| *Skeleton-Lex* | 你不是已经到了吗 Haven't you arrived ? |
| Query | 我月经不太规律 I have irregular periods. |
| Retrieved response | 去 看 医生 啊 . 最 好看 中医 , 挺准 的 , 别 不好意思 Go to see a doctor , best see a traditional Chinese doctor . Pretty accurate , don't be shy. |
| Ours | 看**医生**吧,**最**简单的方法就是**中医**调理一下了 See a **doctor**, the **easiest** way is with **Chinese medicine** recuperation. |
| *Seq2Seq-MMI* | 我也不规律 I am irregular too. |
| *RetrieveNRefine*$^{++}$ | 有啥不规律的 What is irregular? |
| *EditVec* | 有啥不好意思的 Why you are shy? |
| *Skeleton-Lex* | 我也不规律 I am irregular too. |

Table 5: Example responses from different models, as well as a visualization of the skeleton extraction in our model. Darker color indicates bigger matching scores and the words being selected for skeleton are in red boxes.

sons is that for most queries, the set of possible responses is considerably large and the query alone cannot specify an informative response. Various approaches (Li et al., 2016b; Xing et al., 2017; Ghazvininejad et al., 2018; Zhou et al., 2018; Liu et al., 2018; Bi et al., 2019; Tian et al., 2019; Gao et al., 2019) have been proposed for this problem.

Some previous studies have been about using the results of traditional retrieval systems for informative response generation. Song et al. (2016) introduced an extra encoder for the retrieved response. The encoder's output, together with that of the query encoder, is utilized to feed the decoder. Weston et al. (2018) simply concatenated the original query and the retrieved response as the input to the encoder. Instead of solely using the retrieved response, Wu et al. (2019) further introduced to encodes the lexical differences between the current query and the retrieved query. Besides, Pandey et al. (2018) proposed to weight different training instances by context similarity, yet their work is done in close domain conversation. The idea of editing some prototype materials rather than generating from scratch has also been explored in other text generation tasks. For examples, Guu et al. (2018) proposed a prototype-then-edit model for unconditional text generation. Wiseman et al. (2017, 2018) used either fixed template or learned templates for data-to-text generation. Xu et al. (2018) conditioned the next sentence generation on a skeleton that is extracted from the source input and the already generated text in storytelling. Also for storytelling, Clark et al. (2018) proposed to extract the entities in sentences and use them as additional input. Gu et al. (2018) uses retrieved translation as a reference to the generative translation model.

## 5 Conclusion

In this paper, we presented a novel framework, matching-to-generation, for retrieval-guided response generation. Our method uses an interpretable matching model for response skeleton extraction and a robust response generator for response completion. The two components are trained separately to allow more flexibility. Experiments show our method significantly outperforms several strong baselines.

## References


Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Wei Bi, Jun Gao, Xiaojiang Liu, and Shuming Shi. 2019. Fine-grained sentence functions for short-text conversation. In *ACL*, pages 3984–3993.

Deng Cai, Yan Wang, Wei Bi, Zhaopeng Tu, Xiaojiang Liu, Wai Lam, and Shuming Shi. 2019. Skeleton-to-response: Dialogue generation guided by retrieval memory. In *NAACL*.

Elizabeth Clark, Yangfeng Ji, and Noah A. Smith. 2018. Neural text generation in stories using entity representations as context. In *NAACL*, pages 2250–2260.

Jun Gao, Wei Bi, Xiaojiang Liu, Junhui Li, and Shuming Shi. 2019. Generating multiple diverse responses for short-text conversation. In *AAAI*, volume 33, pages 6383–6390.

Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen-tau Yih, and Michel Galley. 2018. A knowledge-grounded neural conversation model. In *AAAI*, pages 5110–5117.

Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor OK Li. 2018. Search engine guided neural machine translation. In *AAAI*.

Kelvin Guu, Tatsunori B Hashimoto, Yonatan Oren, and Percy Liang. 2018. Generating sentences by editing prototypes. *Transactions of the Association for Computational Linguistics*, 6:437–450.

Matthew Henderson, Rami Al-Rfou, Brian Strope, Yun-hsuan Sung, László Lukács, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. 2017. Efficient natural language response suggestion for smart reply. *arXiv preprint arXiv:1705.00652*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *NIPS*, pages 2042–2050.

Zongcheng Ji, Zhengdong Lu, and Hang Li. 2014. An information retrieval approach to short text conversation. *arXiv preprint arXiv:1408.6988*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *NAACL*, pages 110–119.

Jiwei Li, Michel Galley, Chris Brockett, Georgios P Spithourakis, Jianfeng Gao, and Bill Dolan. 2016b. A persona-based neural conversation model. In *ACL*, pages 994–1003.

Kevin Lin, Dianqi Li, Xiaodong He, Zhengyou Zhang, and Ming-Ting Sun. 2017. Adversarial ranking for language generation. In *NIPS*, pages 3155–3165.

Yahui Liu, Wei Bi, Jun Gao, Xiaojiang Liu, Jian Yao, and Shuming Shi. 2018. Towards less generic responses in neural conversation models: A statistical re-weighting method. In *EMNLP*, pages 2769–2774.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*, pages 1412–1421.

Lili Mou, Yiping Song, Rui Yan, Ge Li, Lu Zhang, and Zhi Jin. 2016. Sequence to backward and forward sequences: A content-introducing approach to generative short-text conversation. In *COLING*, pages 3349–3358.

Gaurav Pandey, Danish Contractor, Vineet Kumar, and Sachindra Joshi. 2018. Exemplar encoder-decoder for neural conversation generation. In *ACL*, pages 1329–1338.

Lisong Qiu, Juntao Li, Wei Bi, Dongyan Zhao, and Rui Yan. 2019. Are training samples correlated? learning to generate dialogue responses with multiple references. In *ACL*, pages 3826–3835.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, volume 16, pages 3776–3784.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *ACL*, pages 1577–1586.

Mingyue Shang, Zhenxin Fu, Nanyun Peng, Yansong Feng, Dongyan Zhao, and Rui Yan. 2018. Learning to converse with noisy data: Generation with calibration. In *IJCAI*, pages 4338–4344.

Yiping Song, Rui Yan, Xiang Li, Dongyan Zhao, and Ming Zhang. 2016. Two are better than one: An ensemble of retrieval-and generation-based dialog systems. *arXiv preprint arXiv:1610.07149*.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *NAACL*, pages 196–205.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *CVPR*, pages 2818–2826.

Chongyang Tao, Lili Mou, Dongyan Zhao, and Rui Yan. 2018. Ruber: An unsupervised method for automatic evaluation of open-domain dialog systems. In *AAAI*.

Zhiliang Tian, Wei Bi, Xiaopeng Li, and Nevin L. Zhang. 2019. Learning to abstract for memory-augmented conversational response generation. In *ACL*, pages 3816–3825.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*, pages 5998–6008.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *ICML*, pages 1096–1103.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. In *ICML (Deep Learning Workshop)*.

Joseph Weizenbaum. 1966. Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45.

Jason Weston, Emily Dinan, and Alexander H Miller. 2018. Retrieve and refine: Improved sequence generation models for dialogue. *arXiv preprint arXiv:1808.04776*.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *EMNLP*, pages 2253–2263.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2018. Learning neural templates for text generation. In *EMNLP*, pages 3174–3187.

Yu Wu, Furu Wei, Shaohan Huang, Zhoujun Li, and Ming Zhou. 2019. Response generation by context-aware prototype editing. In *AAAI*.

Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2017. Topic aware neural response generation. In *AAAI*, pages 3351–3357.

Jingjing Xu, Xuancheng Ren, Yi Zhang, Qi Zeng, Xiaoyan Cai, and Xu Sun. 2018. A skeleton-based model for promoting coherence among sentences in narrative story generation. In *EMNLP*, pages 4306–4315.

Tiancheng Zhao, Kyusong Lee, and Maxine Eskenazi. 2018. Unsupervised discrete sentence representation learning for interpretable neural dialog generation. In *ACL*, pages 1098–1107.

Hao Zhou, Tom Young, Minlie Huang, Haizhou Zhao, Jingfang Xu, and Xiaoyan Zhu. 2018. Commonsense knowledge aware conversation generation with graph attention. In *IJCAI*, pages 4623–4629.

# Scalable and Accurate Dialogue State Tracking via Hierarchical Sequence Generation

**Liliang Ren, Jianmo Ni and Julian McAuley**
University of California, San Diego
La Jolla, CA92093
{lren,jin018,jmcauley}@ucsd.edu

## Abstract

Existing approaches to dialogue state tracking rely on pre-defined ontologies consisting of a set of all possible slot types and values. Though such approaches exhibit promising performance on single-domain benchmarks, they suffer from computational complexity that increases proportionally to the number of pre-defined slots that need tracking. This issue becomes more severe when it comes to multi-domain dialogues which include larger numbers of slots. In this paper, we investigate how to approach DST using a generation framework without the pre-defined ontology list. Given each turn of user utterance and system response, we directly generate a sequence of belief states by applying a hierarchical encoder-decoder structure. In this way, the computational complexity of our model will be a constant regardless of the number of pre-defined slots. Experiments on both the multi-domain and the single domain dialogue state tracking dataset show that our model not only scales easily with the increasing number of pre-defined domains and slots but also reaches the state-of-the-art performance.

## 1 Introduction

A Dialogue State Tracker (DST) is a core component of a modular task-oriented dialogue system (Young et al., 2013). For each dialogue turn, a DST module takes a user utterance and the dialogue history as input, and outputs a belief estimate of the dialogue state. Then a machine action is decided based on the dialogue state according to a dialogue policy module, after which a machine response is generated.

Traditionally, a dialogue state consists of a set of *requests* and *joint goals*, both of which are represented by a set of *slot-value* pairs (e.g. (*request*, phone), (*area*, north), (*food*, Japanese)) (Henderson et al., 2014). In a recently proposed multi-domain dialogue state tracking dataset, MultiWoZ (Budzianowski et al., 2018), a representation of dialogue state consists of a hierarchical structure of domain, slot, and value is proposed. This is a more practical scenario since dialogues often include multiple domains simultaneously.

| DST Models | ITC |
|---|---|
| NBT-CNN (Mrksic et al., 2017) | $O(mn)$ |
| MD-DST (Rastogi et al., 2017) | $O(n)$ |
| GLAD (Zhong et al., 2018) | $O(mn)$ |
| StateNet_PSI (Ren et al., 2018) | $O(n)$ |
| TRADE (Wu et al., 2019) | $O(n)$ |
| HyST (Goel et al., 2019) | $O(n)$ |
| DSTRead (Gao et al., 2019) | $O(n)$ |

Table 1: The Inference Time Complexity (ITC) of previous DST models. The ITC is calculated based on how many times inference must be performed to complete a prediction of the belief state in a dialogue turn, where $m$ is the number of values in a pre-defined ontology list and $n$ is the number of slots.

Many recently proposed DSTs (Zhong et al., 2018; Ramadan et al., 2018) are based on pre-defined ontology lists that specify all possible slot values in advance. To generate a distribution over the candidate set, previous works often take each of the *slot-value* pairs as input for scoring. However, in real-world scenarios, it is often not practical to enumerate all possible slot value pairs and perform scoring from a large dynamically changing knowledge base (Xu and Hu, 2018). To tackle this problem, a popular direction is to build a fixed-length candidate set that is dynamically updated throughout the dialogue development. Table 1 briefly summaries the inference time complexity of multiple state-of-the-art DST models following this direction. Since the inference complexity of all of previous model is at least pro-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1876–1885, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

portional to the number of the slots, these models will *struggle* to scale to multi-domain datasets with much larger numbers of pre-defined slots.

**In this work**, we formulate the dialogue state tracking task as a sequence generation problem, instead of formulating the task as a pair-wise prediction problem as in existing work. We propose the **CO**nditional **ME**mory **R**elation Network (**COMER**), a scalable and accurate dialogue state tracker that has a constant inference time complexity. [1]

Specifically, our model consists of an encoder-decoder network with a hierarchically stacked decoder to first generate the slot sequences in the belief state and then for each slot generate the corresponding value sequences. The parameters are shared among all of our decoders for the scalability of the depth of the hierarchical structure of the belief states. COMER applies BERT contextualized word embeddings (Devlin et al., 2018) and BPE (Sennrich et al., 2016) for sequence encoding to ensure the uniqueness of the representations of the unseen words. The word embeddings for sequence generation are initialized and fixed with the static word embeddings generated from BERT to have the potential of generating unseen words.

## 2 Motivation

Figure 1 shows a multi-domain dialogue in which the user wants the system to first help book a train and then reserve a hotel. For each turn, the DST will need to track the *slot-value* pairs (e.g. (arrive by, 20:45)) representing the user goals as well as the domain that the *slot-value* pairs belongs to (e.g. train, hotel). Instead of representing the belief state via a hierarchical structure, one can also combine the domain and slot together to form a *combined slot-value* pair (e.g. (train; arrive by, 20:45) where the *combined slot* is "train; arrive by"), which ignores the subordination relationship between the domain and the slots.

A typical fallacy in dialogue state tracking datasets is that they make an assumption that the slot in a belief state can only be mapped to a single value in a dialogue turn. We call this the *single value assumption*. Figure 2 shows an example of this fallacy from the WoZ2.0 dataset: Based on the belief state label (food, seafood), it will be impossible for the downstream module in the dialogue system to generate sample responses that return information about Chinese restaurants. A correct representation of the belief state could be (food, seafood > chinese). This would tell the system to first search the database for information about seafood and then Chinese restaurants. The logical operator ">" indicates which retrieved information should have a higher priority to be returned to the user. Thus we are interested in building DST modules capable of generating structured sequences, since this kind of sequence representation of the value is critical for accurately capturing the belief states of a dialogue.

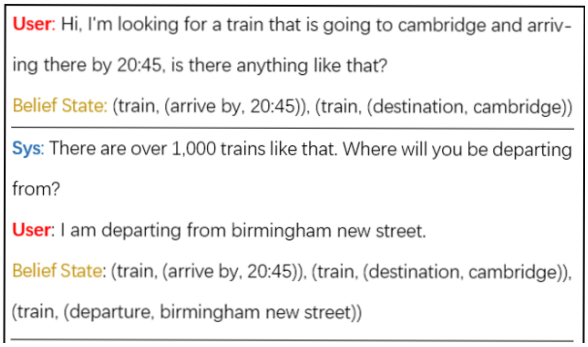


Figure 1: An example dialogue from the multi-domain dataset, MultiWOZ. At each turn, the DST needs to output the belief state, a nested tuple of (DOMAIN, (SLOT, VALUE)), immediately after the user utterance ends. The belief state is accumulated as the dialogue proceeds. Turns are separated by black lines.

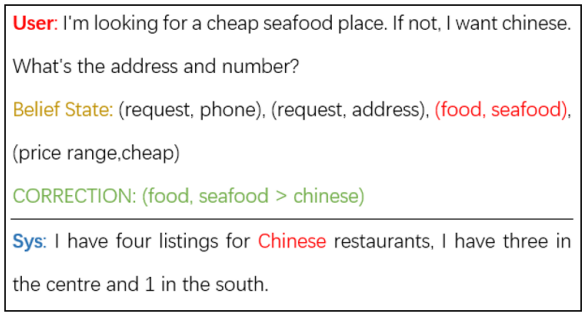


Figure 2: An example in the WoZ2.0 dataset that invalidates the *single value assumption*. It is impossible for the system to generate the sample response about the Chinese restaurant with the original belief state (food, seafood). A correction could be made as (food, seafood > chinese) which has multiple values and a logical operator ">".

## 3 Hierarchical Sequence Generation for DST

Given a dialogue $D$ which consists of $T$ turns of user utterances and system actions, our target is to

[1] The code is released at https://github.com/renll/ComerNet

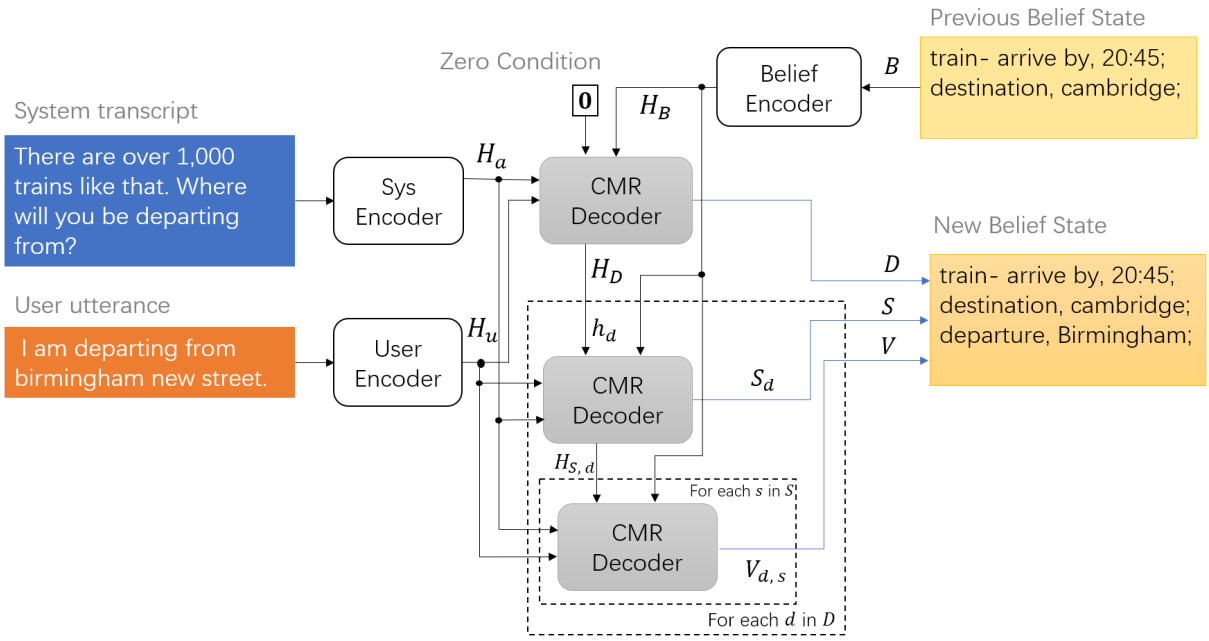


Figure 3: The general model architecture of the Hierarchical Sequence Generation Network. The Conditional Memory Relation (CMR) decoders (gray) share all of their parameters.

predict the state at each turn. Different from previous methods which formulate multi-label state prediction as a collection of binary prediction problems, COMER adapts the task into a sequence generation problem via a Seq2Seq framework.

As shown in Figure 3, COMER consists of three encoders and three hierarchically stacked decoders. We propose a novel Conditional Memory Relation Decoder (CMRD) for sequence decoding. Each encoder includes an embedding layer and a BiLSTM. The encoders take in the user utterance, the previous system actions, and the previous belief states at the current turn, and encodes them into the embedding space. The user encoder and the system encoder use the fixed BERT model as the embedding layer.

Since the *slot value* pairs are un-ordered set elements of a domain in the belief states, we first order the sequence of domain according to their frequencies as they appear in the training set (Yang et al., 2018), and then order the *slot value* pairs in the domain according to the slot's frequencies of as they appear in a domain. After the sorting of the state elements, We represent the belief states following the paradigm: (Domain1- Slot1, Value1; Slot2, Value2; ... Domain2- Slot1, Value1; ...) for a more concise representation compared with the nested tuple representation.

All the CMRDs take the same representations from the system encoder, user encoder and the belief encoder as part of the input. In the procedure of hierarchical sequence generation, the first CMRD takes a zero vector for its condition input $\mathbf{c}$, and generates a sequence of the domains, $D$, as well as the hidden representation of domains $H_D$. For each $d$ in $D$, the second CMRD then takes the corresponding $h_d$ as the condition input and generates the slot sequence $S_d$, and representations, $H_{S,d}$. Then for each $s$ in $S$, the third CMRD generates the value sequence $V_{d,s}$ based on the corresponding $h_{s,d}$. We update the belief state with the new $(d, (s_d, V_{d,s}))$ pairs and perform the procedure iteratively until a dialogue is completed. All the CMR decoders share all of their parameters.

Since our model generates domains and slots instead of taking pre-defined slots as inputs, and the number of domains and slots generated each turn is only related to the complexity of the contents covered in a specific dialogue, the inference time complexity of COMER is $O(1)$ with respect to the number of pre-defined slots and values.

### 3.1 Encoding Module

Let $X$ represent a user utterance or system transcript consisting of a sequence of words $\{w_1, \ldots, w_T\}$. The encoder first passes the sequence $\{[CLS], w_1, \ldots, w_T, [SEP]\}$ into a pre-trained BERT model and obtains its contextual embeddings $E_X$. Specifically, we leverage the output of all layers of BERT and take the average

to obtain the contextual embeddings.

For each domain/slot appeared in the training set, if it has more than one word, such as 'price range', 'leave at', etc., we feed it into BERT and take the average of the word vectors to form the extra slot embedding $E_s$. In this way, we map each domain/slot to a fixed embedding, which allows us to generate a domain/slot as a whole instead of a token at each time step of domain/slot sequence decoding. We also construct a static vocabulary embedding $E_v$ by feeding each token in the BERT vocabulary into BERT. The final static word embedding $E$ is the concatenation of the $E_v$ and $E_s$.

After we obtain the contextual embeddings for the user utterance, system action, and the static embeddings for the previous belief state, we feed each of them into a Bidirectional LSTM (Hochreiter and Schmidhuber, 1997).

$$\begin{aligned}\mathbf{h}_{a_t} &= \text{BiLSTM}(\mathbf{e}_{X_{a_t}}, \mathbf{h}_{a_{t-1}}) \\ \mathbf{h}_{u_t} &= \text{BiLSTM}(\mathbf{e}_{X_{u_t}}, \mathbf{h}_{u_{t-1}}) \\ \mathbf{h}_{b_t} &= \text{BiLSTM}(\mathbf{e}_{X_{b_t}}, \mathbf{h}_{b_{t-1}}) \\ \mathbf{h}_{a_0} &= \mathbf{h}_{u_0} = \mathbf{h}_{b_0} = c_0,\end{aligned} \tag{1}$$

where $c_0$ is the zero-initialized hidden state for the BiLSTM. The hidden size of the BiLSTM is $d_m/2$. We concatenate the forward and the backward hidden representations of each token from the BiLSTM to obtain the token representation $\mathbf{h}_{k_t} \in R^{d_m}$, $k \in \{a, u, b\}$ at each time step $t$. The hidden states of all time steps are concatenated to obtain the final representation of $H_k \in R^{T \times d_m}, k \in \{a, u, B\}$. The parameters are shared between all of the BiLSTMs.

### 3.2 Conditional Memory Relation Decoder

Inspired by Residual Dense Networks (Zhang et al., 2018), End-to-End Memory Networks (Sukhbaatar et al., 2015) and Relation Networks (Santoro et al., 2017), we here propose the Conditional Memory Relation Decoder (CMRD). Given a token embedding, $\mathbf{e}_x$, CMRD outputs the next token, $s$, and the hidden representation, $h_s$, with the hierarchical memory access of different encoded information sources, $H_B$, $H_a$, $H_u$, and the relation reasoning under a certain given condition $\mathbf{c}$,

$$\mathbf{s}, \mathbf{h}_s = \text{CMRD}(\mathbf{e}_x, \mathbf{c}, H_B, H_a, H_u),$$

the final output matrices $S, H_s \in R^{l_s \times d_m}$ are concatenations of all generated $\mathbf{s}$ and $\mathbf{h}_s$ (respectively) along the sequence length dimension, where $d_m$ is the model size, and $l_s$ is the generated sequence length. The general structure of the CMR decoder is shown in Figure 4. Note that the CMR decoder can support additional memory sources by adding the residual connection and the attention block, but here we only show the structure with three sources: belief state representation ($H_B$), system transcript representation ($H_a$), and user utterance representation ($H_u$), corresponding to a dialogue state tracking scenario. Since we share the parameters between all of the decoders, thus CMRD is actually a *2-dimensional* auto-regressive model with respect to both the condition generation and the sequence generation task.

At each time step $t$, the CMR decoder first embeds the token $x_t$ with a fixed token embedding $E \in R^{d_e \times d_v}$, where $d_e$ is the embedding size and $d_v$ is the vocabulary size. The initial token $x_0$ is "[CLS]". The embedded vector $\mathbf{e}_{x_t}$ is then encoded with an LSTM, which emits a hidden representation $\mathbf{h}_0 \in R^{d_m}$,

$$\mathbf{h}_0 = \text{LSTM}(\mathbf{e}_{x_t}, \mathbf{q}_{t-1}).$$

where $\mathbf{q}_t$ is the hidden state of the LSTM. $\mathbf{q}_0$ is initialized with an average of the hidden states of the belief encoder, the system encoder and the user encoder which produces $H_B$, $H_a$, $H_u$ respectively.

$\mathbf{h}_0$ is then summed (element-wise) with the condition representation $\mathbf{c} \in R^{d_m}$ to produce $\mathbf{h}_1$, which is (1) fed into the attention module; (2) used for residual connection; and (3) concatenated with other $\mathbf{h}_i$, $(i > 1)$ to produce the concatenated working memory, $\mathbf{r_0}$, for relation reasoning,

$$\begin{aligned}\mathbf{h}_1 &= \mathbf{h}_0 + \mathbf{c}, \\ \mathbf{h}_2 &= \mathbf{h}_1 + \text{Attn}_{\text{belief}}(\mathbf{h}_1, H_e), \\ \mathbf{h}_3 &= \mathbf{h}_2 + \text{Attn}_{\text{sys}}(\mathbf{h}_2, H_a), \\ \mathbf{h}_4 &= \mathbf{h}_3 + \text{Attn}_{\text{usr}}(\mathbf{h}_3, H_u), \\ \mathbf{r} &= \mathbf{h}_1 \oplus \mathbf{h}_2 \oplus \mathbf{h}_3 \oplus \mathbf{h}_4 \in R^{4d_m},\end{aligned}$$

where $\text{Attn}_k$ ($k \in \{\text{belief}, \text{sys}, \text{usr}\}$) are the attention modules applied respectively to $H_B$, $H_a$, $H_u$, and $\oplus$ means the concatenation operator. The gradients are blocked for $\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3$ during the back-propagation stage, since we only need them to work as the supplementary memories for the relation reasoning followed.

The attention module takes a vector, $\mathbf{h} \in R^{d_m}$, and a matrix, $H \in R^{d_m \times l}$ as input, where $l$ is the

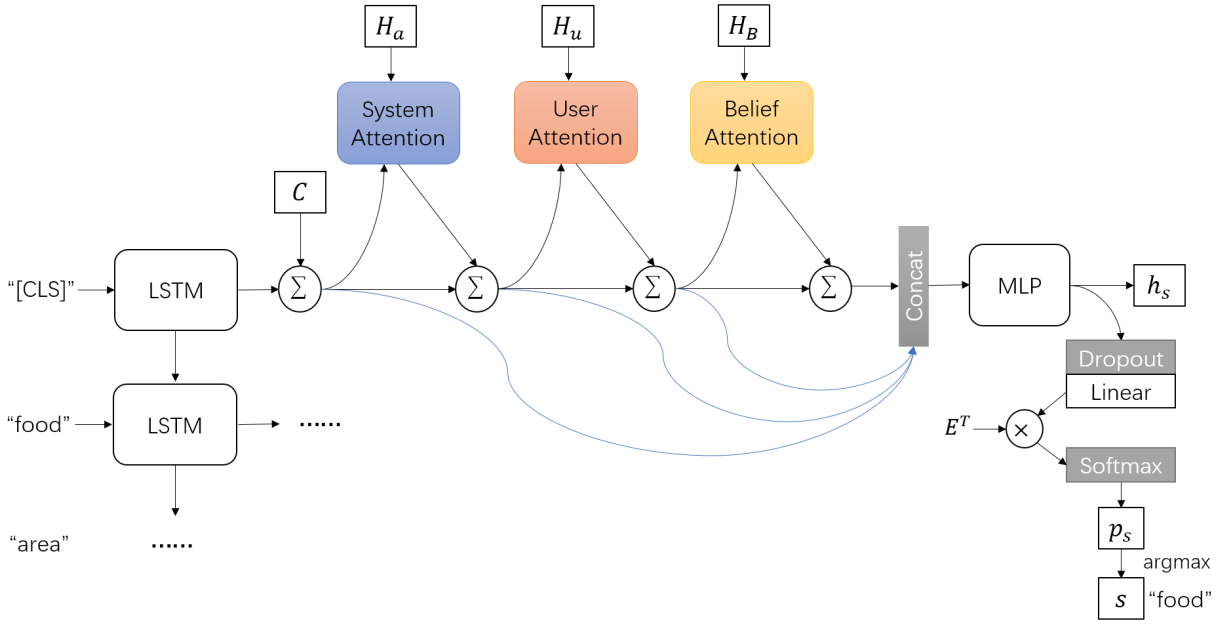


Figure 4: The general structure of the Conditional Memory Relation Decoder. The decoder output, $s$ (e.g. "food"), is refilled to the LSTM for the decoding of the next step. The blue lines in the figure means that the gradients are blocked during the back propagation stage.

sequence length of the representation, and outputs $\mathbf{h}_a$, a weighted sum of the column vectors in $H$.

$$\begin{aligned}
\mathbf{a} &= W_1^T \mathbf{h} + \mathbf{b}_1 && \in R^{d_m}, \\
\mathbf{c} &= \text{softmax}(H^T a) && \in R^{l}, \\
\mathbf{h} &= H\mathbf{c} && \in R^{d_m}, \\
\mathbf{h}_a &= W_2^T \mathbf{h} + \mathbf{b}_2 && \in R^{d_m},
\end{aligned}$$

where the weights $W_1 \in R^{d_m \times d_m}$, $W_2 \in R^{d_m \times d_m}$ and the bias $b_1 \in R^{d_m}$, $b_2 \in R^{d_m}$ are the learnable parameters.

The order of the attention modules, *i.e.,* first attend to the system and the user and then the belief, is decided empirically. We can interpret this hierarchical structure as the internal order for the memory processing, since from the daily life experience, people tend to attend to the most contemporary memories (system/user utterance) first and then attend to the older history (belief states). All of the parameters are shared between the attention modules.

The concatenated working memory, $\mathbf{r}_0$, is then fed into a Multi-Layer Perceptron (MLP) with four layers,

$$\begin{aligned}
\mathbf{r}_1 &= \sigma(W_1^T \mathbf{r}_0 + \mathbf{b}_1), \\
\mathbf{r}_2 &= \sigma(W_2^T \mathbf{r}_1 + \mathbf{b}_2), \\
\mathbf{r}_3 &= \sigma(W_3^T \mathbf{r}_2 + \mathbf{b}_3), \\
\mathbf{h}_s &= \sigma(W_4^T \mathbf{r}_3 + \mathbf{b}_4),
\end{aligned}$$

where $\sigma$ is a non-linear activation, and the weights $W_1 \in R^{4d_m \times d_m}$, $W_i \in R^{d_m \times d_m}$ and the bias $b_1 \in R^{d_m}$, $b_i \in R^{d_m}$ are learnable parameters, and $2 \leq i \leq 4$. The number of layers for the MLP is decided by the grid search.

The hidden representation of the next token, $\mathbf{h}_s$, is then (1) emitted out of the decoder as a representation; and (2) fed into a dropout layer with drop rate $p$, and a linear layer to generate the next token,

$$\begin{aligned}
\mathbf{h}_k &= \text{dropout}(\mathbf{h}_s) && \in R^{d_m}, \\
\mathbf{h}_o &= W_k^T \mathbf{h}_k + \mathbf{b}_k && \in R^{d_e}, \\
\mathbf{p}_s &= \text{softmax}(E^T \mathbf{h}_o) && \in R^{d_v}, \\
s &= \text{argmax}(\mathbf{p}_s) && \in R,
\end{aligned}$$

where the weight $W_k \in R^{d_m \times d_e}$ and the bias $b_k \in R^{d_e}$ are learnable parameters. Since $d_e$ is the embedding size and the model parameters are independent of the vocabulary size, the CMR decoder can make predictions on a dynamic vocabulary and implicitly supports the generation of unseen words. When training the model, we minimize the cross-entropy loss between the output probabilities, $\mathbf{p}_s$, and the given labels.

## 4 Experiments

### 4.1 Experimental Setting

We first test our model on the single domain dataset, WoZ2.0 (Wen et al., 2017). It consists

| Metric | WoZ2.0 | MultiWoZ |
|---|---|---|
| Avg. # turns, $t$ | 7.45 | 13.68 |
| Avg. tokens / turn, $s$ | 11.24 | 13.18 |
| Number of Slots, $n$ | 3 | 35 |
| Number of Values, $m$ | 99 | 4510 |
| Training set size | 600 | 8438 |
| Validation set size | 200 | 1000 |
| Test set size | 400 | 1000 |

Table 2: The statistics of the WoZ2.0 and the MultiWoZ datasets.

of 1,200 dialogues from the restaurant reservation domain with three pre-defined slots: *food*, *price range*, and *area*. Since the *name* slot rarely occurs in the dataset, it is not included in our experiments, following previous literature (Ren et al., 2018; Liu and Perez, 2017). Our model is also tested on the multi-domain dataset, MultiWoZ (Budzianowski et al., 2018). It has a more complex ontology with 7 domains and 25 predefined slots. Since the *combined slot-value* pairs representation of the belief states has to be applied for the model with $O(n)$ ITC, the total number of slots is 35. The statistics of these two datsets are shown in Table 2.

Based on the statistics from these two datasets, we can calculate the theoretical Inference Time Multiplier (ITM), $K$, as a metric of scalability. Given the inference time complexity, ITM measures how many times a model will be slower when being transferred from the WoZ2.0 dataset, $d_1$, to the MultiWoZ dataset, $d_2$,

$$K = h(t)h(s)h(n)h(m)$$

$$h(x) = \begin{cases} 1 & O(x) = O(1), \\ \frac{x_{d_2}}{x_{d_1}} & \text{otherwise}, \end{cases}$$

where $O(x)$ means the Inference Time Complexity (ITC) of the variable $x$. For a model having an ITC of $O(1)$ with respect to the number of slots $n$, and values $m$, the ITM will be a multiplier of 2.15x, while for an ITC of $O(n)$, it will be a multiplier of 25.1, and 1,143 for $O(mn)$.

As a convention, the metric of joint goal accuracy is used to compare our model to previous work. The joint goal accuracy only regards the model making a successful belief state prediction if all of the slots and values predicted are exactly matched with the labels provided. This metric gives a strict measurement that tells how often the DST module will not propagate errors to the downstream modules in a dialogue system. In this work, the model with the highest joint accuracy on the validation set is evaluated on the test set for the test joint accuracy measurement.

### 4.2 Implementation Details

We use the $\text{BERT}_{\text{large}}$ model for both contextual and static embedding generation. All LSTMs in the model are stacked with 2 layers, and only the output of the last layer is taken as a hidden representation. ReLU non-linearity is used for the activation function, $\sigma$.

The hyper-parameters of our model are identical for both the WoZ2.0 and the MultiwoZ datasets: dropout rate $p = 0.5$, model size $d_m = 512$, embedding size $d_e = 1024$. For training on WoZ2.0, the model is trained with a batch size of 32 and the ADAM optimizer (Kingma and Ba, 2015) for 150 epochs, while for MultiWoZ, the AMSGrad optimizer (Reddi et al., 2018) and a batch size of 16 is adopted for 15 epochs of training. For both optimizers, we use a learning rate of 0.0005 with a gradient clip of 2.0. We initialize all weights in our model with Kaiming initialization (He et al., 2015) and adopt zero initialization for the bias. All experiments are conducted on a single NVIDIA GTX 1080Ti GPU.

### 4.3 Results

To measure the actual inference time multiplier of our model, we evaluate the runtime of the best-performing models on the validation sets of both the WoZ2.0 and MultiWoZ datasets. During evaluation, we set the batch size to 1 to avoid the influence of data parallelism and sequence padding. On the validation set of WoZ2.0, we obtain a runtime of 65.6 seconds, while on MultiWoZ, the runtime is 835.2 seconds. Results are averaged across 5 runs. Considering that the validation set of MultiWoZ is 5 times larger than that of WoZ2.0, the actual inference time multiplier is 2.54 for our model. Since the actual inference time multiplier roughly of the same magnitude as the theoretical value of 2.15, we can confirm empirically that we have the $O(1)$ inference time complexity and thus obtain full scalability to the number of slots and values pre-defined in an ontology.

Table 3 compares our model with the previous state-of-the-art on both the WoZ2.0 test set and the MultiWoZ test set. For the WoZ2.0 dataset, we maintain performance at the level of the state-of-

| DST Models | Joint Acc. WoZ 2.0 | Joint Acc. MultiWoZ | ITC |
|---|---|---|---|
| Baselines (Mrksic et al., 2017) | 70.8% | 25.83% | $O(mn)$ |
| NBT-CNN (Mrksic et al., 2017) | 84.2% | - | $O(mn)$ |
| StateNet_PSI (Ren et al., 2018) | **88.9%** | - | $O(n)$ |
| GLAD (Nouri and Hosseini-Asl, 2018) | 88.5% | 35.58% | $O(mn)$ |
| HyST (ensemble) (Goel et al., 2019) | - | 44.22% | $O(n)$ |
| DSTRead (ensemble) (Gao et al., 2019) | - | 42.12% | $O(n)$ |
| TRADE (Wu et al., 2019) | - | **48.62%** | $O(n)$ |
| COMER | 88.6% | 45.72% | $O(1)$ |

Table 3: The joint goal accuracy of the DST models on the WoZ2.0 test set and the MultiWoZ test set. We also include the Inference Time Complexity (ITC) for each model as a metric for scalability. The baseline accuracy for the WoZ2.0 dataset is the Delexicalisation-Based (DB) Model (Mrksic et al., 2017), while the baseline for the MultiWoZ dataset is taken from the official website of MultiWoZ (Budzianowski et al., 2018).

| Model | Joint Acc. |
|---|---|
| COMER | 88.64% |
| - Hierachical-Attn | 86.69% |
| - MLP | 83.24% |

Table 4: The ablation study on the WoZ2.0 dataset with the joint goal accuracy on the test set. For "- Hierachical-Attn", we remove the residual connections between the attention modules in the CMR decoders and all the attention memory access are based on the output from the LSTM. For "- MLP", we further replace the MLP with a single linear layer with the non-linear activation.

| Model | JD Acc. | JDS Acc. | JG Acc. |
|---|---|---|---|
| COMER | 95.53% | 54.74% | 45.72% |
| - ShareParam | 94.96% | 54.40% | 44.38% |
| - Order | 95.55% | 55.06% | 42.84% |
| - Nested | - | 49.58% | 40.57% |
| - BlockGrad | - | 49.36% | 39.15% |

Table 5: The ablation study on the MultiWoZ dataset with the joint domain accuracy (JD Acc.), joint domain-slot accuracy (JDS Acc.) and joint goal accuracy (JG Acc.) on the test set. For "- ShareParam", we remove the parameter sharing mechanism on the encoders and the attention module. For "- Order", we further arrange the order of the slots according to its global frequencies in the training set instead of the local frequencies given the domain it belongs to. For "- Nested", we do not generate domain sequences but generate *combined slot* sequences which combines the domain and the slot together. For "- BlockGrad", we further remove the gradient blocking mechanism in the CMR decoder.

the-art, with a marginal drop of 0.3% compared with previous work. Considering the fact that WoZ2.0 is a relatively small dataset, this small difference does not represent a significant big performance drop. On the muli-domain dataset, MultiWoZ, our model achieves a joint goal accuracy of 45.72%, which is significant better than most of the previous models other than TRADE which applies the copy mechanism and gains better generalization ability on named entity coping.

### 4.4 Ablation Study

To prove the effectiveness of our structure of the Conditional Memory Relation Decoder (CMRD), we conduct ablation experiments on the WoZ2.0 dataset. We observe an accuracy drop of 1.95% after removing residual connections and the hierarchical stack of our attention modules. This proves the effectiveness of our hierarchical attention design. After the MLP is replaced with a linear layer of hidden size 512 and the ReLU activation function, the accuracy further drops by 3.45%. This drop is partly due to the reduction of the number of the model parameters, but it also proves that stacking more layers in an MLP can improve the relational reasoning performance given a concatenation of multiple representations from different sources.

We also conduct the ablation study on the MultiWoZ dataset for a more precise analysis on the hierarchical generation process. For joint domain accuracy, we calculate the probability that all do-

mains generated in each turn are exactly matched with the labels provided. The joint domain-slot accuracy further calculate the probability that all domains and slots generated are correct, while the joint goal accuracy requires all the domains, slots and values generated are exactly matched with the labels. From Table 5, We can further calculate that given the correct slot prediction COMER has 83.52% chance to make the correct value prediction. While COMER has done great job on domain prediction (95.53%) and value prediction (83.52%), the accuracy of the slot prediction given the correct domain is only 57.30%. We suspect that this is because we only use the previous belief state to represent the dialogue history, and the inter-turn reasoning ability on the slot prediction suffers from the limited context and the accuracy is harmed due to the multi-turn mapping problem (Wu et al., 2019). We can also see that the JDS Acc. has an absolute boost of 5.48% when we switch from the *combined slot* representation to the nested tuple representation. This is because the subordinate relationship between the domains and the slots can be captured by the hierarchical sequence generation, while this relationship is missed when generating the domain and slot together via the *combined slot* representation.

### 4.5 Qualitative Analysis

Figure 5 shows an example of the belief state prediction result in one turn of a dialogue on the MultiWoZ test set. The visualization includes the CMRD attention scores over the belief states, system transcript and user utterance during the decoding stage of the slot sequence.

From the system attention (top right), since it is the first attention module and no previous context information is given, it can only find the information indicating the slot "departure" from the system utterance under the domain condition, and attend to the evidence "leaving" correctly during the generation step of "departure". From the user attention, we can see that it captures the most helpful keywords that are necessary for correct prediction, such as "after" for "day" and "leave at", "to" for "destination". Moreover, during the generation step of "departure", the user attention successfully discerns that, based on the context, the word "leave" is not the evidence that need to be accumulated and choose to attend nothing in this step. For the belief attention, we can see that the belief attention module correctly attends to a previous slot for each generation step of a slot that has been presented in the previous state. For the generation step of the new slot "destination", since the previous state does not have the "destination" slot, the belief attention module only attends to the '-' mark after the 'train' domain to indicate that the generated word should belong to this domain.

## 5 Related Work

**Semi-scalable Belief Tracker** Rastogi et al. (2017) proposed an approach that can generate fixed-length candidate sets for each of the slots from the dialogue history. Although they only need to perform inference for a fixed number of values, they still need to iterate over all slots defined in the ontology to make a prediction for a given dialogue turn. In addition, their method needs an external language understanding module to extract the exact entities from a dialogue to form candidates, which will not work if the label value is an abstraction and does not have the exact match with the words in the dialogue.

StateNet (Ren et al., 2018) achieves state-of-the-art performance with the property that its parameters are independent of the number of slot values in the candidate set, and it also supports online training or inference with dynamically changing slots and values. Given a slot that needs tracking, it only needs to perform inference once to make the prediction for a turn, but this also means that its inference time complexity is proportional to the number of slots.

TRADE (Wu et al., 2019) achieves state-of-the-art performance on the MultiWoZ dataset by applying the copy mechanism for the value sequence generation. Since TRADE takes $n$ combinations of the domains and slots as the input, the inference time complexity of TRADE is $O(n)$. The performance improvement achieved by TRADE is mainly due to the fact that it incorporates the copy mechanism that can boost the accuracy on the name slot, which mainly needs the ability in copying names from the dialogue history. However, TRADE does not report its performance on the WoZ2.0 dataset which does not have the name slot.

DSTRead (Gao et al., 2019) formulate the dialogue state tracking task as a reading comprehension problem by asking slot specified questions to the BERT model and find the answer span in the

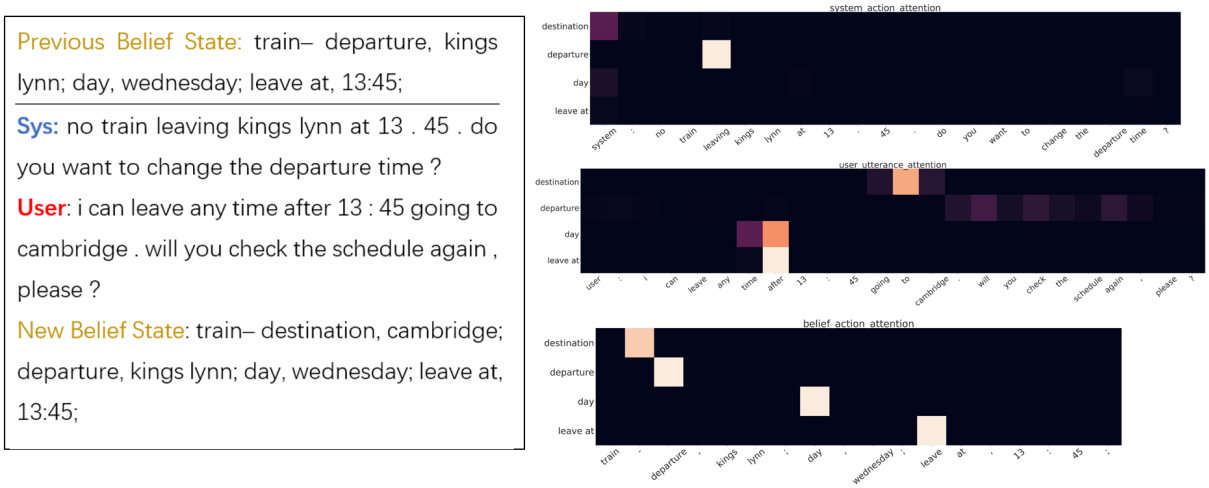


Figure 5: An example belief prediction of our model on the MultiWoZ test set. The attention scores for belief states, system transcript and user utterance in CMRD is visualized on the right. Each row corresponds to the attention score of the generation step of a particular slot under the 'train' domain.

dialogue history for each of the pre-defined *combined slot*. Thus its inference time complexity is still $O(n)$. This method suffers from the fact that its generation vocabulary is limited to the words occurred in the dialogue history, and it has to do a manual combination strategy with another joint state tracking model on the development set to achieve better performance.

**Contextualized Word Embedding (CWE)** was first proposed by Peters et al. (2018). Based on the intuition that the meaning of a word is highly correlated with its context, CWE takes the complete context (sentences, passages, etc.) as the input, and outputs the corresponding word vectors that are unique under the given context. Recently, with the success of language models (e.g. Devlin et al. (2018)) that are trained on large scale data, contextualizeds word embedding have been further improved and can achieve the same performance compared to (less flexible) finely-tuned pipelines.

**Sequence Generation Models.** Recently, sequence generation models have been successfully applied in the realm of multi-label classification (MLC) (Yang et al., 2018). Different from traditional binary relevance methods, they proposed a sequence generation model for MLC tasks which takes into consideration the correlations between labels. Specifically, the model follows the encoder-decoder structure with an attention mechanism (Cho et al., 2014), where the decoder generates a sequence of labels. Similar to language modeling tasks, the decoder output at each time step will be conditioned on the previous predictions during generation. Therefore the correlation between generated labels is captured by the decoder.

## 6 Conclusion

In this work, we proposed the Conditional Memory Relation Network (COMER), the first dialogue state tracking model that has a constant inference time complexity with respect to the number of domains, slots and values pre-defined in an ontology. Besides its scalability, the joint goal accuracy of our model also achieve the similar performance compared with the state-of-the-arts on both the MultiWoZ dataset and the WoZ dataset. Due to the flexibility of our hierarchical encoder-decoder framework and the CMR decoder, abundant future research direction remains as applying the transformer structure, incorporating open vocabulary and copy mechanism for explicit unseen words generation, and inventing better dialogue history access mechanism to accommodate efficient inter-turn reasoning.

**Acknowledgements.** This work is partly supported by NSF #1750063. We thank all the reviewers for their constructive suggestions. We also want to thank Zhuowen Tu and Shengnan Zhang for the early discussions of the project.

## References

Pawel Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. 2018. Multiwoz - a large-

scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In *EMNLP*.

Kyunghyun Cho, Bart van Merrienboer, aglar Glehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Shuyang Gao, Abhishek Sethi, Sanchit Agarwal, Tagyoung Chung, and Dilek Zeynep Hakkani. 2019. Dialog state tracking: A neural reading comprehension approach. *ArXiv*, abs/1908.01946.

Rahul Goel, Shachi Paul, and Dilek Zeynep Hakkani. 2019. Hyst: A hybrid approach for flexible and accurate dialogue state tracking. *ArXiv*, abs/1907.00883.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034.

Matthew Henderson, Blaise Thomson, and Jason D. Williams. 2014. The second dialog state tracking challenge. In *SIGDIAL Conference*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9:1735–1780.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*.

Fei Liu and Julien Perez. 2017. Dialog state tracking, a machine reading approach using memory network. In *EACL*.

Nikola Mrksic, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve J. Young. 2017. Neural belief tracker: Data-driven dialogue state tracking. In *ACL*.

Elnaz Nouri and Ehsan Hosseini-Asl. 2018. Toward scalable neural dialogue state tracking model. *arXiv preprint arXiv:1812.00899*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke S. Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*.

Osman Ramadan, Pawel Budzianowski, and Milica Gasic. 2018. Large-scale multi-domain belief tracking with knowledge sharing. In *ACL*.

Abhinav Rastogi, Dilek Hakkani-Tur, and Larry Heck. 2017. Scalable multi-domain dialogue state tracking. *arXiv preprint arXiv:1712.10224*.

Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. 2018. On the convergence of adam and beyond. In *ICLR*.

Liliang Ren, Kaige Xie, Lu Chen, and Kai Yu. 2018. Towards universal dialogue state tracking. In *EMNLP*.

Adam Santoro, David Raposo, David G. T. Barrett, Mateusz Malinowski, Razvan Pascanu, Peter W. Battaglia, and Timothy P. Lillicrap. 2017. A simple neural network module for relational reasoning. In *NIPS*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. *CoRR*, abs/1508.07909.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *NIPS*.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Lina Maria Rojas-Barahona, Pei hao Su, Stefan Ultes, David Vandyke, and Steve J. Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *EACL*.

Chien-Sheng Wu, Andrea Madotto, Ehsan Hosseini-Asl, Caiming Xiong, Richard Socher, and Pascale Fung. 2019. Transferable multi-domain state generator for task-oriented dialogue systems. In *ACL*.

Puyang Xu and Qi Hu. 2018. An end-to-end approach for handling unknown slot values in dialogue state tracking. In *ACL*.

Pengcheng Yang, Xu Sun, Wei Li, Shuming Ma, Wei Wu, and Houfeng Wang. 2018. Sgm: Sequence generation model for multi-label classification. In *COLING*.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. 2018. Residual dense network for image super-resolution. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2472–2481.

Victor Zhong, Caiming Xiong, and Richard Socher. 2018. Global-locally self-attentive dialogue state tracker. *CoRR*, abs/1805.09655.

# Low-Resource Response Generation with Template Prior

**Ze Yang**[†], **Wei Wu**[◇], **Jian Yang**[†], **Can Xu**[◇], **Zhoujun Li**[†*]
[†]State Key Lab of Software Development Environment, Beihang University, Beijing, China
[◇]Microsoft Corporation, Beijing, China
{tobey, jiaya, lizj}@buaa.edu.cn {wuwei, caxu}@microsoft.com

## Abstract

We study open domain response generation with limited message-response pairs. The problem exists in real-world applications but is less explored by the existing work. Since the paired data now is no longer enough to train a neural generation model, we consider leveraging the large scale of unpaired data that are much easier to obtain, and propose response generation with both paired and unpaired data. The generation model is defined by an encoder-decoder architecture with templates as prior, where the templates are estimated from the unpaired data as a neural hidden semi-markov model. By this means, response generation learned from the small paired data can be aided by the semantic and syntactic knowledge in the large unpaired data. To balance the effect of the prior and the input message to response generation, we propose learning the whole generation model with an adversarial approach. Empirical studies on question response generation and sentiment response generation indicate that when only a few pairs are available, our model can significantly outperform several state-of-the-art response generation models in terms of both automatic and human evaluation.

## 1 Introduction

Human-machine conversation is a long-standing goal of artificial intelligence. Early dialogue systems are designed for task completion with conversations restricted in specific domains (Young et al., 2013). Recently, thanks to the advances in deep learning techniques (Sutskever et al., 2014; Vaswani et al., 2017) and the availability of large amounts of human conversation on the internet, building an open domain dialogue system with data-driven approaches has become the new fashion in the research of conversational AI. Such dialogue systems can generate reasonable responses without any needs on rules, and have powered products in the industry such as Amazon Alexa (Ram et al., 2018) and Microsoft XiaoIce (Shum et al., 2018).

State-of-the-art open domain response generation models are based on the encoder-decoder architecture (Vinyals and Le, 2015; Shang et al., 2015). On the one hand, with proper extensions to the vanilla structure, existing models now are able to naturally handle conversation contexts (Serban et al., 2016; Xing et al., 2018), and synthesize responses with various styles (Wang et al., 2017), emotions (Zhou et al., 2018), and personas (Li et al., 2016a). On the other hand, all the existing success of open domain response generation builds upon an assumption that the large scale of paired data (Shao et al., 2016) or conversation sessions (Sordoni et al., 2015) are available. In this work, we challenge this assumption by arguing that one cannot always obtain enough pairs or sessions for training a neural generation model. For example, although it has been indicated by existing work (Li et al., 2016b; Wang et al., 2018) that question asking in conversation can enhance user engagement, we find that in a public dataset[1] with 5 million conversation sessions crawled from Weibo, only 7.3% sessions have a question response and thus can be used to learn a question generator for responding[2]. When we attempt to generate responses that express positive sentiment, we only get 360k pairs (18%) with positive responses from a dataset with 2 million message-response pairs crawled from Twitter. Indeed, existing big conversation data mix various intentions, styles, emotions, personas, and so on. Thus,

* Corresponding Author

[1] http://tcci.ccf.org.cn/conference/2018/dldoc/trainingdata05.zip

[2] Questions are detected with the rules in (Wang et al., 2018).



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1886–1897, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

we have to face the data sparsity problem, as long as we attempt to create a generation model with constraints on responses.

In this work, we jump out of the paradigm of learning from large scale paired data[3], and investigate how to build a response generation model with only a few pairs at hand. Aside from the paired data, we assume that there are a large number of unpaired data available. The assumption is reasonable since it is much easier to get questions or sentences with positive sentiment than to get such responses paired with messages. We formalize the problem as low-resource response generation from paired and unpaired data, which is less explored by existing work. Since the paired data are insufficient for learning the mapping from a message to a response, the challenge of the task lies in how to effectively leverage the unpaired data to enhance the learning on the paired data. Our solution to the challenge is a two-stage approach where we first distill templates from the unpaired data and then use them to guide response generation. Targeting on an unsupervised approach to template learning, we propose representing the templates as a neural hidden semi-markov model (NHSMM) estimated through maximizing the likelihood of the unpaired data. Such latent templates encode both semantics and syntax of the unpaired data and then are used as prior in an encoder-decoder architecture for modeling the paired data. With the latent templates, the whole model is end-to-end learnable and can perform response generation in an explainable manner. To ensure the relevance of responses regarding input messages and at the same time make full use of the templates, we propose learning the generation model with an adversarial approach.

Empirical studies are conducted on two tasks: question response generation and sentiment response generation. For the first task, we exploit the dataset published in (Wang et al., 2018) and augment the data with questions crawled from Zhihu[4]. For the second task, we build a paired dataset from Twitter by filtering responses with an off-the-shelf sentiment classifier and augment the dataset with tweets in positive sentiment extracted from a large scale tweet dataset published in (Cheng et al., 2010). Evaluation results on both automatic metrics and human judgment indicate that with limited message-response pairs, our model can significantly outperform several state-of-the-art response generation models. The source code is available online. [5]

Our contributions in this work are three-folds: (1) proposal of low-resource response generation with paired and unpaired data for open domain dialogue systems; (2) proposal of encoder-decoder with template prior; and (3) empirical verification of the effectiveness of the model with two large-scale datasets.

## 2 Related Work

Inspired by neural machine translation, early work applies the sequence-to-sequence with attention model (Shang et al., 2015) to open domain response generation, and gets promising results. Later, the basic architecture is extended to suppress generic responses (Li et al., 2015; Zhao et al., 2017; Xing et al., 2017); to model the structure of conversation contexts (Serban et al., 2016); and to incorporate different types of knowledge into generation (Li et al., 2016a; Zhou et al., 2018). In addition to model design, how to learn a generation model (Li et al., 2016c, 2017), and how to evaluate the models (Liu et al., 2016; Lowe et al., 2017; Tao et al., 2018), are drawing attention in the community of open domain dialogue generation. In this work, we study how to learn a response generation model from limited pairs, which breaks the assumption made by existing work. We propose response generation with paired and unpaired data. As far as we know, this is the first work on low-resource response generation for open domain dialogue systems.

Traditional template-based text generation (Becker, 2002; Foster and White, 2004; Gatt and Reiter, 2009) relies on handcrafted templates that are expensive to obtain. Recently, some work explores how to automatically mine templates from plain text and how to integrate the templates into neural architectures to enhance interpretability of generation. Along this line, Duan et al. (2017) mine patterns from related questions in community QA websites and leverage the patterns with a retrieval-based approach and a generation-based approach for question generation. Wiseman et al. (2018) exploit a hidden semi-markov model for joint template extraction and text generation. In

[3]The study in this work starts from response generation for single messages. One can easily extend the proposed approach to handle conversation history.

[4]https://en.wikipedia.org/wiki/Zhihu

[5]https://github.com/TobeyYang/S2S_Temp

addition to structured templates, raw text retrieved from indexes is also used as "soft templates" in various natural language generation tasks (Guu et al., 2018; Pandey et al., 2018; Cao et al., 2018; Peng et al., 2019). In this work, we leverage templates for open domain response generation. Our idea is inspired by (Wiseman et al., 2018), but latent templates estimated from one source are transferred to another source in order to handle the low-resource problem, and the generation model is learned by an adversarial approach rather than by maximum likelihood estimation.

Before us, the low-resource problem has been studied in tasks such as machine translation (Gu et al., 2018b,a), pos tagging (Kann et al., 2018), word embedding (Jiang et al., 2018), automatic speech recognition (Tüske et al., 2014), task-oriented dialogue systems (Tran and Nguyen, 2018; Mi et al., 2019), etc. In this work, we pay attention to low-resource open domain response generation which is untouched by existing work. We propose attacking the problem with unpaired data, which is related to the effort in low-resource machine translation with monolingual data (Gulcehre et al., 2015; Sennrich et al., 2015; Zhang and Zong, 2016). Our method is unique in that rather than using the unpaired data through multi-task learning (Zhang and Zong, 2016) or back-translation (Sennrich et al., 2015), we extract linguistic knowledge from the data as latent templates and use the templates as prior in generation.

## 3 Low-Resource Response Generation

In this section, we first formalize the setting upon which we study low-resource response generation and then elaborate the model of response generation with paired and unpaired data, including how to learn latent templates from the unpaired data, and how to perform generation with the templates.

### 3.1 Problem Formalization

Suppose that we have a dataset $\mathcal{D}_P = \{(X_i, Y_i)\}_{i=1}^n$, where $\forall i$, $(X_i, Y_i)$ is a pair of message-response, and $n$ represents the number of pairs in $\mathcal{D}_P$. Different from existing work, we assume that $n$ is small (e.g., a few hundred thousands) and further assume that there is another set $\mathcal{D}_U = \{T_i\}_{i=1}^N$ with $T_i$ a piece of plain text sharing the same characteristics with $\{Y_i\}_{i=1}^n$ (e.g., both are questions) and $N > n$. Our goal is to learn a generation probability $P(Y|X)$ with both $\mathcal{D}_P$ and $\mathcal{D}_U$. Thus, given a new message $X$, we can generate a response $Y$ for $X$ following $P(Y|X)$.

Since the limited resource in $\mathcal{D}_P$ may not support accurately learning of $P(Y|X)$, we try to transfer the linguistic knowledge in $\mathcal{D}_U$ to response generation. The challenges then lie in two aspects: (1) how to represent the linguistic knowledge in $\mathcal{D}_U$; and (2) how to effectively leverage the knowledge extracted from $\mathcal{D}_U$ for response generation, given that $\mathcal{D}_U$ cannot provide any information of correspondence between a message $X$ and a response $Y$. The remaining part of the section will describe our solutions to the two problems.

### 3.2 Learning Templates from $\mathcal{D}_U$

In the representation of the knowledge in $\mathcal{D}_U$, we hope that both semantic information and syntactic information can be kept. Thus, we consider extracting templates from $\mathcal{D}_U$ as the knowledge. A template segments a piece of text as a structured representation. With the templates, semantically and functionally similar text segments are grouped together. Since the templates encode the structure of language in $\mathcal{D}_U$, they can inform the generation model about how to express a response in a desired way (e.g., as a question or with the specific sentiment). Here, we prefer an unsupervised and parametric approach to learning templates, since "unsupervised" means that the approach is generally applicable to various tasks, and "parameteric" allows us to naturally incorporate the templates into the generation model. Then, a natural choice for template learning is the neural hidden semi-markov model (NHSMM) (Dai et al., 2016; Wiseman et al., 2018).

NHSMM is an HSMM parameterized with neural networks. HSMM (Murphy, 2002) extends HMM by allowing a hidden state to emit a sequence of observations and thus can segment a piece of text with the latent variables and group similar segments by the variables. Formally, given an observed sequence $Y = (y_1, \ldots, y_S)$, the joint distribution of $Y$ and its segmentation is

$$\prod_{t=1}^{S'} P(z_{t+1}, l_{t+1}|z_t, l_t) \prod_{t=1}^{S'} P(y_{i(t-1)+1:i(t)}|z_t, l_t),$$

where $z_t \in \{1, \ldots, K\}$ is the hidden state for step $t$, $l_t \in \{1, \ldots, D\}$ is the duration variable for $z_t$ that represents the number of tokens emitted by $z_t$, $i(t) = \sum_{j=1}^t l_j$ with $i(0) = 0$ and $i(S') = S$, and $y_{i(t-1)+1:i(t)}$ is the sequence

of $(y_{i(t-1)+1}, \ldots, y_{i(t)})$. $P(z_{t+1}, l_{t+1}|z_t, l_t)$ is factorized as $P(z_{t+1}|z_t) \times P(l_{t+1}|z_{t+1})$ where $P(l_{t+1}|z_{t+1})$ is a uniform distribution on $\{1, \ldots, D\}$ and $P(z_{t+1}|z_t)$ can be viewed as a transition matrix $[A(i,j)]_{K\times K}$ which is defined by

$$\begin{aligned} A(i,j) &\triangleq P(z_{t+1}=j|z_t=i) \\ &= \frac{\exp(e_j^\top e_i + b_{i,j})}{\sum_{o=1}^{K}\exp(e_o^\top e_i + b_{i,o})}, \end{aligned}$$

where $e_i, e_j, e_o \in \mathbb{R}^{d_1}$ are embeddings of state $i, j, o$ respectively, and $b_{i,j}, b_{i,o}$ are scalar bias terms. In practice, we set $b_{i,j} = -\infty \Leftrightarrow i = j$ to disable self-transition, because the adjacent states play different syntactic or semantic roles in a desired template. The emission distribution $P(y_{i(t-1)+1:i(t)}|z_t, l_t)$ is defined by

$$\begin{aligned} P(y_{i(t-1)+1:i(t)}|z_t, l_t) &= P(y_{i(t-1)+1}|z_t, l_t) \\ &\times \prod_{j=2}^{l_t} P(y_{i(t-1)+j}|y_{i(t-1)+j-1}, z_t, l_t), \end{aligned}$$

and parameterized with a recurrent neural network with gated recurrent unit (GRU) (Cho et al., 2014). The hidden vector for position $j$ is formulated as

$$\begin{aligned} o_j^t &= \text{GRU}_\text{H}(o_{j-1}^t, [e_{z_t}; e_{y_{i(t-1)+j-1}}]) \\ v_j^t &= g_{z_t} \odot o_j^t, \end{aligned} \quad (1)$$

where $o_j^t \in \mathbb{R}^{d_2}$, $e_{y_{i(t-1)+j-1}} \in \mathbb{R}^{d_3}$ is the embedding of word $y_{i(t-1)+j-1}$, $[\cdot;\cdot]$ is a concatenation operator, $\odot$ refers to element-wise multiplication, and $g_{z_t} \in \mathbb{R}^{d_2}$ is a gate (in total, there are $K$ gate vectors as parameters). $P(y_{i(t-1)+j}|y_{i(t-1)+j-1}, z_t, l_t)$ is then defined by

$$\text{softmax}(W_1 v_j^t + b_1),$$

where $W_1 \in \mathbb{R}^{V\times d_2}$ and $b_1 \in \mathbb{R}^{d_2}$ are parameters with $V$ the vacabulary size.

Following Murphy (2002), the marginal distribution of $Y$ can be obtained by the backward algorithm which is formulated as

$$\begin{aligned} \beta_t(i) &\triangleq P(y_{t+1:S}|q_t=i) = \sum_{j=1}^{K}\beta_t^*(j)A(i,j) \\ \beta_t^*(j) &\triangleq P(y_{t+1:S}|q_{t+1}=j) \\ &= \sum_{d=1}^{D}\Big[\beta_{t+d}(j)P(d|j)P(y_{t+1:t+d}|j,d)\Big] \\ P(Y) &= \sum_{j=1}^{K}\beta_0^*(j)P(q_1=j). \end{aligned} \quad (2)$$

where $q_t$ is the hidden state of the $t$-th word in $Y$, and the base cases $\beta_S(i) = 1, \forall i \in \{1, \ldots, K\}$. Specifically, to learn more reasonable segmentations, we parsed every sentence by stanford parser (Manning et al., 2014) and forced NHSMM not to break syntactic elements such as VP and NP, etc. The parameters of the NHSMM are estimated by maximizing the log-likelihood of $\mathcal{D}_U$ through backpropagation.

### 3.3 Response Generation with Template Prior

We propose incorporating the templates parameterized by the NHSMM learned from $\mathcal{D}_U$ into response generation as prior. Figure 1 illustrates the architecture of the generation model. In a nutshell, the model first samples a chain of states with duration as a template. The template specifies a segmentation of the response to generate. Then, the hidden representations of the segments defined by Equation (1) are fed to an encoder-decoder architecture for response generation, where the hidden states of the decoder are calculated with both attention over the hidden states of the input message given by the encoder and the hidden representations of the segments given by the template prior. The template prior acts as a base and assists the encoder-decoder in response generation regarding to an input message, when paired information is insufficient for learning the correspondence between a message and a response. Note that similar to the conditional variational autoencoder (CVAE) (Zhao et al., 2017), our model also exploits hidden variables for response generation. The difference is that the hidden variables in our model are structured and learned from extra resources, and thus encode more semantic and syntactic information.

Specifically, we segment responses in $\mathcal{D}_P$ with Viterbi algorithm (Zucchini et al., 2016), collect all chains of states as a pool and sample a chain from the pool uniformly. We do not sample states according to the transition matrix $[A(i,j)]_{K\times K}$, since it is difficult to determine the end of a chain. Suppose that the sampled chain is $(z_1, \ldots z_S)$, then $\forall 1 \le t \le S$, we sample an $l_t$ for $z_t$ according to $P(l_t|z_t)$, and finally form a latent template $(< z_1, l_1 >, \ldots < z_S, l_S >)$. Given a message $X = (x_1, \ldots, x_L)$, the encoder exploits a GRU to transform $X$ into a hidden sequence $H_X = (h_{X,1}, \ldots, h_{X,L})$ with the $i$-th hidden state

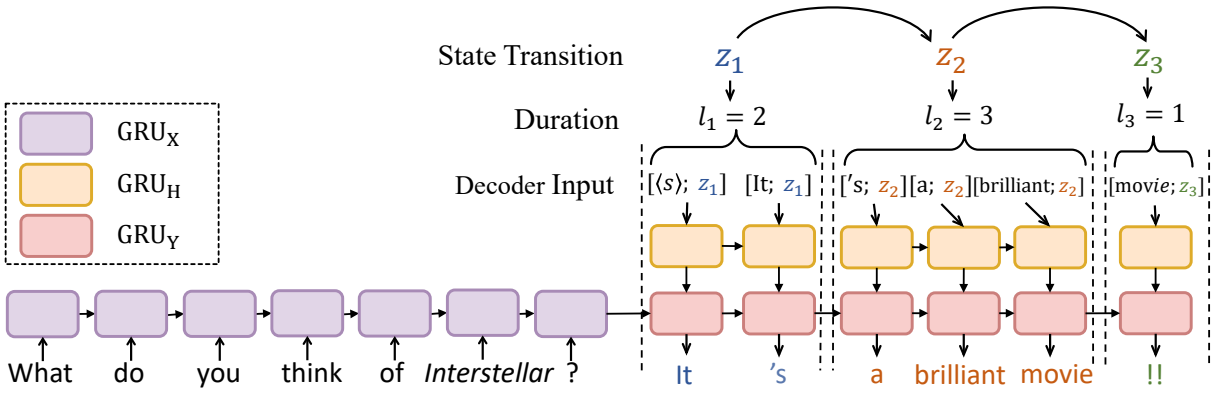


Figure 1: The architecture of the generation model.

$h_{X,i} \in \mathbb{R}^{d_2}$ given by

$$h_{X,i} = \text{GRU}_X(h_{X,i-1}, e_{x_i}),$$

where $e_{x_i} \in \mathbb{R}^{d_3}$ is the embedding of word $x_i$ and $h_{X,0} = 0$. Then when predicting the $t$-th word of the response, the decoder calculates the probability $P(y_t|y_{1:t-1}, X, \mathcal{T})$ via

$$\begin{aligned} P(y_t|y_{1:t-1}, X, \mathcal{T}) =& \text{softmax}(W_2[s_t; c_t] + b_2) \\ s_t =& \text{GRU}_Y(s_{t-1}, v_k^m). \end{aligned}$$

with parameters $W_2 \in \mathbb{R}^{V \times 2d_2}$ and $b_2 \in \mathbb{R}^V$, $s_t \in \mathbb{R}^{d_2}$ and $s_{t-1} \in \mathbb{R}^{d_2}$ are the hidden states of the decoder for step $t$ and step $t-1$ respectively, $v_k^m$ is defined by Equation (1) where $m$ satisfies $i(m-1) < t \leq i(m)$, $k = t - i(m-1)$, and $o_k^m = \text{GRU}_H(o_{k-1}^m, [e_{z_m}; e_{y_{t-1}}])$, and $c_t \in \mathbb{R}^{d_2}$ is a context vector of $X$ obtained via attention over $H_X$ (Bahdanau et al., 2015):

$$\begin{aligned} c_t &= \sum_{i=1}^{L} \alpha_{t,i} h_{X,i}, \\ \alpha_{t,i} &= \frac{\exp(s_{t,i})}{\sum_{j=1}^{L} \exp(s_{t,j})} \\ s_{t,i} &= v^\top \tanh(W s_t + U h_i + b), \end{aligned}$$

where $v, b \in \mathbb{R}^{d_2}$, $W, U \in \mathbb{R}^{d_2 \times d_2}$ are parameters.

## 4 Learning Approach

Intuitively, we can estimate the parameters of the encoder-decoder and fine-tune the parameters of NHSMM by maximizing the likelihood of $\mathcal{D}_P$ (i.e., MLE). However, since $\mathcal{D}_P$ only contains a few pairs, the MLE approach may suffer from a dilemma: (1) if we stop training early, then both the template prior and the encoder-decoder are not sufficiently supervised by the pairs. In that case, the linguistic knowledge in $\mathcal{D}_U$ will play a more important role in response generation and result in irrelevant responses regarding to messages; or (2) if we let the training go deep, then the template prior will be overwhelmed by the pairs in $\mathcal{D}_P$. As a result, the generation model will lose the knowledge obtained from $\mathcal{D}_U$. Since response generation starts from a latent template, we consider learning the model with an adversarial approach (Goodfellow et al., 2014) that can well balance the effect of the latent template and the input message. The learning involves a generator $G$ described in Section 3 and a discriminator $D$. $G$ is updated with REINFORCE algorithm (Williams, 1992) with rewards defined by $D$, and $D$ is updated to distinguish human responses in $\mathcal{D}_P$ from responses generated by $G$.

**Generator Pre-training.** To improve the stability of adversarial learning, we first pre-train $G$ with MLE on $\mathcal{D}_P$. $\forall (X_i, Y_i) \in \mathcal{D}_P$, the template prior $\mathcal{T}_i$ is obtained by running Viterbi algorithm (Zucchini et al., 2016) on $Y_i$ rather than by sampling. Let $Y_i = (y_{i,1}, \ldots, y_{i,S_i})$, then the objective of pre-training is given by

$$\frac{1}{n} \sum_{(X_i, Y_i) \in \mathcal{D}_P} \sum_{t=1}^{S_i} \log P(y_{i,t}|y_{i,1:t-1}, X_i, \mathcal{T}_i). \quad (3)$$

**Discriminator Update.** The discriminator $D$ is defined by a convolutional neural network (CNN) based binary classifier (Kim, 2014). $D$ takes a message-response pair as input and outputs a score that indicates how likely the response is from humans. In the model, the message and the response are separately embedded as vectors by CNNs, and then the concatenation of the two vectors are fed to

a 2-layer MLP to calculate the score. Let $\hat{Y}_i$ be the response generated by $G$ for $X_i$, then $D$ is updated by maximizing the following objective:

$$\sum_{(X_i,Y_i)\in\mathcal{D}_P} \log D(X_i, Y_i) + \log(1 - D(X_i, \hat{Y}_i)) \tag{4}$$

**Generator Update.** The generator $G$ is updated by the policy gradient method (Yu et al., 2017; Li et al., 2017). Let $\hat{y}_{1:t}$ be a partial response generated by $G$ from beam search for message $X$ until step $t$, then we adopt the Monte Carlo (MC) search method and sample $N$ paths that supplement $\hat{y}_{1:t}$ as responses $\{\hat{Y}_i\}_{i=1}^N$. The intermediate reward for $\hat{y}_{1:t}$ is defined as $R_t = \frac{1}{N}\sum_{i=1}^N D(X, \hat{Y}_i)$. The gradient for updating $G$ is given by

$$\nabla J(\theta) \approx \sum_{t=1}^{S} \big[\nabla \log P(\hat{y}_t|\hat{y}_{1:t-1}, X, \mathcal{T}) \cdot R_t\big], \tag{5}$$

where $\theta$ represents the parameters of $G$, and $\mathcal{T}$ is a sampled template. To control the quality of MC search, we sample from top 50 most probable words at each step.

The learning algorithm is summarized in Algorithm 1. Note that in learning of the generation model from $\mathcal{D}_P$, we freeze the embedding of states (i.e., $e_{z_t}$ in Equation (1)) and the embedding of words given by the NHSMM, and update all other parameters in generator pre-training and the following adversarial learning.

**Algorithm 1** Learning a generation model with paired and unpaired data.

**Require:** NHSMM $H$, generator $G$, discriminator $D$, $\mathcal{D}_U$, and $\mathcal{D}_P$.
Initialize $H$, $G$, $D$.
Learn $H$ from $\mathcal{D}_U$ according to Equation (2).
Pre-train $G$ using MLE on $\mathcal{D}_P$.
Pre-train $D$ using $G$ according to Equation (4).
**repeat**
  **for** g-steps **do**
    Sample $(X, Y)$ from $\mathcal{D}_P$.
    Sample a template $\mathcal{T}$ for $(X, Y)$ with $H$.
    Generate $\hat{Y} = (\hat{y}_1, \ldots, \hat{y}_S) \sim G(\cdot|X, \mathcal{T})$.
    **for** $t$ in $1 : S$ **do**
      MC search and compute reward $R_t$ using $D$.
    **end for**
    Update $G$ on $(X, \hat{Y})$ via Equation (5).
  **end for**
  **for** d-steps **do**
    Sample $(X, Y)$ from $\mathcal{D}_P$.
    Sample a template $\mathcal{T}$ for $(X, Y)$ with $H$.
    Generate $\hat{Y} \sim G(\cdot|X, \mathcal{T})$ and pair $(X, \hat{Y})$ with $(X, Y)$.
    Update $D$ by Equation (4).
  **end for**
**until** convergence

## 5 Experiments

We test the proposed approach on two tasks: question response generation and sentiment response generation. The first task requires a model to generate a question as a response to a given message; while in the second task, as a showcase, responses should express the positive sentiment.

### 5.1 Experiment Setup

**Datasets.** For the question response generation task, we choose the data published in (Wang et al., 2018) as the paired dataset. The data are obtained by filtering 9 million message-response pairs mined from Weibo with 20 handcrafted question templates and are split as a training set, a validation set, and a test set with 481k, 5k, and 5k pairs respectively. In addition to the paired data, we crawl 776k questions from Zhihu, a Chinese community QA website featured by high-quality content, as an unpaired dataset. Both datasets are tokenized by Stanford Chinese word segmenter[6]. We keep $20,000$ most frequent words in the two data as a vocabulary for the encoder, the decoder, and the NHSMM. The vocabulary covers 95.8% words appearing in the messages, in the responses, and in the questions. Other words are replaced with "UNK". For the sentiment response generation task, we mine 2 million message-response pairs from Twitter FireHose, filter responses with the positive sentiment using Stanford Sentiment Annotator toolkit (Socher et al., 2013), and obtain 360k pairs as a paired dataset. As pre-processing, we remove URLs and usernames, and transform each word to its lower case. After that, the data is split as a training set, a validation set, and a test set with 350k, 5k, and 5k pairs respectively. Besides, we extract 1 million tweets with positive sentiment from a public corpus (Cheng et al., 2010) as an unpaired dataset. Top $20,000$ most frequent words in the two data are kept as a vocabulary that covers 99.3% words. Words excluded from the vocabulary are treated as "UNK". In both tasks, human responses in the test sets are taken as ground truth for automatic metric calculation. From each test set, we randomly sample 500 distinct messages and recruit human annotators to judge the quality of responses generated for these messages.

[6] https://stanfordnlp.github.io/CoreNLP

**Evaluation Metrics.** We conduct evaluation with both automatic metrics and human judgements. For automatic evaluation, besides BLEU-1 (Papineni et al., 2002) and Rouge-L (Lin, 2004), we follow (Serban et al., 2017) and employ Emebedding Average (Average), Embedding Extrema (Extrema), and Embedding Greedy (Greedy) as metrics. All these metrics are computed by a popular NLG evaluation project available at https://github.com/Maluuba/nlg-eval. In terms of human evaluation, for each task, we recruit 3 well-educated native speakers as annotators, and let them compare our model and each of the baselines. Every time, we show an annotator a message (in total 500) and two responses, one from our model and the other from a baseline model. Both responses are top 1 results in beam search, and the two responses are presented in random order. The annotator then compare the two responses from three aspects: (1) **Fluency**: if the response is fluent without grammatical error; (2) **Relevance**: if the response is relevant to the given message; and (3) **Richness**: if the response contains informative and interesting content, and thus may keep conversation going. For each aspect, if the annotator cannot tell which response is better, he/she is asked to label a "tie". Each pair of responses receive 3 labels on each of the three aspects, and agreements among the annotators are measured by Fleiss' kappa (Fleiss and Cohen, 1973).

## 5.2 Baselines

We compare our model with the following baselines: (1) **Seq2Seq**: the basic sequence-to-sequence with attention architecture (Bahdanau et al., 2015). (2) **CVAE**: the conditional variational autoencoder that represents the relationship between messages and responses with latent variables (Zhao et al., 2017). We use the code published at https://github.com/snakeztc/NeuralDialog-CVAE. (3) **HTD**: the hard typed decoder model proposed in (Wang et al., 2018) that exhibits the best performance on the dataset selected by this work for question response generation. The model estimates distributions over three types of words (i.e., interrogative, topic, and ordinary) and modulates the final distribution during generation. Since our experiments are conducted on the same data as those in (Wang et al., 2018), we run the code shared at https://github.com/victorywys/Learning2Ask_TypedDecoder with the default setting. (4) **ECM**: emotional chatting machine proposed in (Zhou et al., 2018). We implement the model with the code published at https://github.com/tuxchow/ecm. Since the model can handle various emotions, we train the model with the entire 2 million Twitter message-response pairs labeled with a positive, negative, and neutral sentiment. Thus, when we only focus on responses with positive sentiment, ECM actually performs multi-task learning for response generation. In the test, we set the sentiment label as "positive".

| | BLEU-1 | ROUGE-L | AVERAGE | EXTREME | GREEDY |
|---|---|---|---|---|---|
| Seq2Seq | 0.037 | 0.111 | 0.656 | 0.438 | 0.456 |
| CVAE | 0.094 | 0.088 | 0.685 | 0.414 | 0.422 |
| HTD | 0.073 | 0.103 | 0.647 | 0.425 | 0.439 |
| S2S-Temp-MLE | 0.097 | 0.119 | 0.699 | 0.438 | 0.457 |
| S2S-Temp-None | 0.069 | 0.092 | 0.677 | 0.429 | 0.416 |
| S2S-Temp-50% | 0.091 | 0.113 | 0.702 | 0.442 | 0.461 |
| S2S-Temp | **0.102** | **0.128** | **0.710** | **0.451** | **0.469** |

Table 1: Automatic evaluation results for the task of question response generation. Numbers in bold mean that the improvement over the best performing baseline is statistically significant (t-test with $p$-value$< 0.01$).

| | BLEU-1 | ROUGE-L | AVERAGE | EXTREME | GREEDY |
|---|---|---|---|---|---|
| Seq2Seq | 0.065 | 0.118 | 0.726 | 0.474 | 0.582 |
| CVAE | 0.088 | 0.081 | 0.727 | 0.408 | 0.563 |
| ECM | 0.051 | 0.102 | 0.708 | 0.462 | 0.559 |
| S2S-Temp-MLE | 0.103 | 0.124 | 0.732 | 0.458 | 0.593 |
| S2S-Temp-None | 0.078 | 0.089 | 0.687 | 0.479 | 0.501 |
| S2S-Temp-50% | 0.102 | 0.121 | 0.691 | 0.491 | 0.586 |
| S2S-Temp | **0.106** | **0.130** | **0.738** | **0.492** | **0.603** |

Table 2: Automatic evaluation results for the task of sentiment response generation. Numbers in bold mean that the improvement over the best performing baseline is statistically significant (t-test, with $p$-value$< 0.01$).

We name our model **S2S-Temp**. Besides the full model, we also examine three variants in order to understand the effect of unpaired data and the role of adversarial learning: (1) S2S-Temp-None. The proposed model is trained only with the paired data, where the NHSMM is estimated from responses in the paired data; (2) S2S-Temp-50%. The proposed model is trained with 50% unpaired data; and (3) S2S-Temp-MLE. The pre-trained generator described in Section 4. These variants are only involved in automatic evaluation.

## 5.3 Implementation Details

In both tasks, we set the number of states (i.e., $K$) and the the maximal number of emissions (i.e., $D$) in NHSMM as 50 and 4 respectively. $d_1$, $d_2$ and

| Models | Fluency | | | Relevance | | | Richness | | | Kappa |
|---|---|---|---|---|---|---|---|---|---|---|
| | W(%) | L(%) | T(%) | W(%) | L(%) | T(%) | W(%) | L(%) | T(%) | |
| S2S-Temp vs. Seq2Seq | 20.8 | 18.3 | 60.9 | 30.8 | 22.5 | 46.7 | 42.5 | 19.2 | 38.3 | 0.63 |
| S2S-Temp vs. CVAE | 41.7 | 5.7 | 52.6 | 50.8 | 12.5 | 36.7 | 37.5 | 15.8 | 46.7 | 0.71 |
| S2S-Temp vs. HTD | 35.1 | 19.2 | 45.8 | 30.8 | 25.1 | 44.1 | 37.5 | 30.8 | 31.7 | 0.64 |
| S2S-Temp vs. Seq2Seq | 15.6 | 11.5 | 72.9 | 34.4 | 17.2 | 48.4 | 31.9 | 7.3 | 60.8 | 0.68 |
| S2S-Temp vs. CVAE | 48.4 | 9.0 | 42.6 | 31.9 | 5.7 | 62.4 | 31.4 | 8.2 | 60.4 | 0.69 |
| S2S-Temp vs. ECM | 27.1 | 12.3 | 60.6 | 36.9 | 13.9 | 49.2 | 27.9 | 10.6 | 61.5 | 0.78 |

Table 3: Human annotation results. W, L, and T refer to Win, Lose and Tie respectively. The first three rows are results on question response generation, and the last three rows are results on sentiment response generation. The ratios are calculated by combining labels from the three judges.

$d_3$ are set as 600, 300, and 300 respectively. In adversarial learning, we use three types of filters with window sizes 1, 2 and 3 in the discriminator. The number of filters is 128 for each type. The number of samples obtained from MC search (i.e., $N$) at each step is 5. We learn all models using Adam algorithm (Kingma and Ba, 2015), monitor perplexity on the validation sets, and terminate training when perplexity gets stable. In our model, learning rates for NHSMM, the generator, and the discriminator are set as $1 \times 10^{-3}$, $1 \times 10^{-5}$, and $1 \times 10^{-3}$ respectively.

### 5.4 Evaluation Results

Table 1 and Table 2 report the results of automatic evaluation on the two tasks. We can see that on both tasks, S2S-Temp outperforms all baseline models in terms of all metrics, and the improvements are statistically significant (t-test with $p$-value$<$ 0.01). The results demonstrate that when only limited pairs are available, S2S-Temp can effectively leverage unpaired data to enhance the quality of response generation. Although lacking fine-grained check, from the comparison among S2S-Temp-None, S2S-Temp-50%, and S2S-Temp, we can conclude that the performance of S2S-Temp improves with more unpaired data. Moreover, without unpaired data, our model is even worse than CVAE since the structured templates cannot be accurately estimated from such a few data, and as long as half of the unpaired data are available, the model outperforms the baseline models on most metrics. The results further verified the important role the unpaired data plays in learning of a response generation model from low resources. S2S-Temp is better than S2S-Temp-MLE, indicating that the adversarial learning approach can indeed enhance the relevance of responses regarding to messages.

Table 3 shows the results of human evaluation. In terms of all the three aspects, S2S-Temp is better than all the baseline models. The values of kappa are all above 0.6, indicating substantial agreement among the annotators. When the size of paired data is small, the basic Seq2Seq model tends to generate more generic responses. That is why the gap between S2S-Temp and Seq2Seq is much smaller on fluency than those on the other two aspects. With the latent variables, CVAE brings both content and noise into responses. Therefore, the gap between S2S-Temp and CVAE is more significant on fluency and relevance than that on richness. HTD can greatly enrich the content of responses, which is consistent with the results in (Wang et al., 2018), although sometimes the responses might be irrelevant to messages or ill-formed. ECM does not perform well on both automatic evaluation and human judgement.

### 5.5 Case Study

To further understand how S2S-Temp leverages templates for response generation, we show two examples with the test data, one for question response generation in Table 4 and the other for sentiment response generation in Table 5, where subscripts refer to states of the NHSMMs. First, we can see that a template defines a structure for a response. By varying templates, we can have responses with different syntax and semantics for a message. Second, some states may have consistent functions across responses. For example, state 36 in question response generation may refer to pronouns, and "I'm" and "it was" correspond to the same state 23 in sentiment response generation. Finally, some templates provide strong syntactic signals to response generation. For example, the

| | |
|---|---|
| **Message:** | 真的假的？我瘦了16斤<br>Really? I lost 17.6 pounds |
| **Responses:** | [你]$_{36}$ [瘦 了]$_{31}$ [吗 ?]$_{13}$<br>[You]$_{36}$ [lost weight]$_{31}$ [?]$_{13}$<br>[你 是 怎么]$_{14}$ [做到 的]$_{42}$ [?]$_{26}$<br>[How do you]$_{14}$ [make it]$_{42}$ [?]$_{26}$<br>[真的 吗 ?]$_{48}$ [我]$_{36}$ [不 信]$_{32}$<br>[Really ?]$_{48}$ [I]$_{36}$ [don't believe it]$_{32}$ |

Table 4: Question response generation with various templates.

| | |
|---|---|
| **Message:** | One of my favoriate Eddie Murphy movies! |
| **Responses:** | [it 's]$_{29}$ [a brilliant]$_{35}$ [movie]$_{37}$<br>[i screamed]$_{29}$ [! ! !]$_{1}$<br>[it was]$_{23}$ [so underrated]$_{14}$ [!]$_{31}$<br>[honestly .]$_{30}$ [i 'm]$_{23}$ [so pumped]$_{14}$ [to watch it]$_{31}$<br>[yeah ,]$_{47}$ [i was watching]$_{48}$ |

Table 5: Sentiment response generation with various templates.

segmentation of "Really? I don't believe it" given by the template (48, 36, 32) matches the parsing result "FRAG + LS + VP " given by stanford syntactic parser.

## 6 Conclusions

We study low-response response generation for open domain dialogue systems by assuming that paired data are insufficient for modeling the relationship between messages and responses. To augment the paired data, we consider transferring knowledge from unpaired data to response generation through latent templates parameterized as a hidden semi-markov model, and take the templates as prior in generation. Evaluation results on question response generation and sentiment response generation indicate that when limited pairs are available, our model can significantly outperform several state-of-the-art response generation models.

## Acknowledgement

We appreciate the valuable comments provided by the anonymous reviewers. This work is supported in part by the National Natural Science Foundation of China (Grand Nos. U1636211, 61672081, 61370126), and the National Key R&D Program of China (No. 2016QY04W0802).

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Tilman Becker. 2002. Practical, template–based natural language generation with tag. In *Proceedings of the Sixth International Workshop on Tree Adjoining Grammar and Related Frameworks (TAG+ 6)*, pages 80–83.

Ziqiang Cao, Wenjie Li, Sujian Li, and Furu Wei. 2018. Retrieve, rerank and rewrite: Soft template based neural summarization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 152–161.

Zhiyuan Cheng, James Caverlee, and Kyumin Lee. 2010. You are where you tweet: a content-based approach to geo-locating twitter users. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 759–768. ACM.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Hanjun Dai, Bo Dai, Yan-Ming Zhang, Shuang Li, and Le Song. 2016. Recurrent hidden semi-markov model.

Nan Duan, Duyu Tang, Peng Chen, and Ming Zhou. 2017. Question generation for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 866–874, Copenhagen, Denmark. Association for Computational Linguistics.

Joseph L Fleiss and Jacob Cohen. 1973. The equivalence of weighted kappa and the intraclass correlation coefficient as measures of reliability. *Educational and psychological measurement*, 33(3):613–619.

Mary Ellen Foster and Michael White. 2004. Techniques for text planning with xslt. In *Proceeedings of the Workshop on NLP and XML (NLPXML-2004): RDF/RDFS and OWL in Language Technology*.

Albert Gatt and Ehud Reiter. 2009. Simplenlg: A realisation engine for practical applications. In *Proceedings of the 12th European Workshop on Natural Language Generation (ENLG 2009)*, pages 90–93.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron

Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Jiatao Gu, Hany Hassan, Jacob Devlin, and Victor O.K. Li. 2018a. Universal neural machine translation for extremely low resource languages. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 344–354, New Orleans, Louisiana. Association for Computational Linguistics.

Jiatao Gu, Yong Wang, Yun Chen, Victor O. K. Li, and Kyunghyun Cho. 2018b. Meta-learning for low-resource neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3622–3631, Brussels, Belgium. Association for Computational Linguistics.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. *arXiv preprint arXiv:1503.03535*.

Kelvin Guu, Tatsunori B Hashimoto, Yonatan Oren, and Percy Liang. 2018. Generating sentences by editing prototypes. *Transactions of the Association of Computational Linguistics*, 6:437–450.

Chao Jiang, Hsiang-Fu Yu, Cho-Jui Hsieh, and Kai-Wei Chang. 2018. Learning word embeddings for low-resource languages by pu learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1.

Katharina Kann, Johannes Bjerva, Isabelle Augenstein, Barbara Plank, and Anders Søgaard. 2018. Character-level supervision for low-resource pos tagging. In *Proceedings of the Workshop on Deep Learning Approaches for Low-Resource NLP*, pages 1–11.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2015. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119.

Jiwei Li, Michel Galley, Chris Brockett, Georgios Spithourakis, Jianfeng Gao, and Bill Dolan. 2016a. A persona-based neural conversation model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 994–1003.

Jiwei Li, Alexander H Miller, Sumit Chopra, Marc'Aurelio Ranzato, and Jason Weston. 2016b. Learning through dialogue interactions by asking questions. *arXiv preprint arXiv:1612.04936*.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016c. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202.

Jiwei Li, Will Monroe, Tianlin Shi, Sėbastien Jean, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2157–2169.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2122–2132.

Ryan Lowe, Michael Noseworthy, Iulian Vlad Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. 2017. Towards an automatic turing test: Learning to evaluate dialogue responses. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1116–1126.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Fei Mi, Minlie Huang, Jiyong Zhang, and Boi Faltings. 2019. Meta-learning for low-resource natural language generation in task-oriented dialogue systems. *arXiv preprint arXiv:1905.05644*.

Kevin P Murphy. 2002. Hidden semi-markov models (hsmms). *unpublished notes*, 2.

Gaurav Pandey, Danish Contractor, Vineet Kumar, and Sachindra Joshi. 2018. Exemplar encoder-decoder for neural conversation generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1329–1338.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Hao Peng, Ankur P Parikh, Manaal Faruqui, Bhuwan Dhingra, and Dipanjan Das. 2019. Text generation with exemplar-based adaptive decoding. *arXiv preprint arXiv:1904.04428*.

Ashwin Ram, Rohit Prasad, Chandra Khatri, Anu Venkatesh, Raefer Gabriel, Qing Liu, Jeff Nunn, Behnam Hedayatnia, Ming Cheng, Ashish Nagar, et al. 2018. Conversational ai: The science behind the alexa prize. *arXiv preprint arXiv:1801.03604*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709*.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. 2016. End-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, pages 3776–3784.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*, pages 3295–3301.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *ACL*, pages 1577–1586.

Louis Shao, Stephan Gouws, Denny Britz, Anna Goldie, Brian Strope, and Ray Kurzweil. 2016. Generating long and diverse responses with neural conversation models.

Heung-Yeung Shum, Xiaodong He, and Di Li. 2018. From eliza to xiaoice: Challenges and opportunities with social chatbots. *Frontiers of Information Technology & Electronic Engineering*, 19(1):10–26.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 196–205.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Chongyang Tao, Lili Mou, Dongyan Zhao, and Rui Yan. 2018. Ruber: An unsupervised method for automatic evaluation of open-domain dialog systems. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Van-Khanh Tran and Le-Minh Nguyen. 2018. Dual latent variable model for low-resource natural language generation in dialogue systems. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 21–30.

Zoltán Tüske, Pavel Golik, David Nolden, Ralf Schlüter, and Hermann Ney. 2014. Data augmentation, feature combination, and multilingual neural networks to improve asr and kws performance for low-resource languages. In *Fifteenth Annual Conference of the International Speech Communication Association*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Di Wang, Nebojsa Jojic, Chris Brockett, and Eric Nyberg. 2017. Steering output style and topic in neural response generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2140–2150.

Yansen Wang, Chenyi Liu, Minlie Huang, and Liqiang Nie. 2018. Learning to ask questions in open-domain conversational systems with typed decoders. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2193–2203.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2018. Learning neural templates for text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3174–3187.

Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2017. Topic aware neural response generation. In *AAAI*, pages 3351–3357.

Chen Xing, Wei Wu, Yu Wu, Ming Zhou, Yalou Huang, and Wei-Ying Ma. 2018. Hierarchical recurrent attention network for response generation. In *AAAI*, pages 5610–5617.

Stephanie Young, Milica Gasic, Blaise Thomson, and John D Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Jiajun Zhang and Chengqing Zong. 2016. Exploiting source-side monolingual data in neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 654–664.

Hao Zhou, Minlie Huang, Tianyang Zhang, Xiaoyan Zhu, and Bing Liu. 2018. Emotional chatting machine: Emotional conversation generation with internal and external memory. In *AAAI*, pages 730–738.

Walter Zucchini, Iain L MacDonald, and Roland Langrock. 2016. *Hidden Markov models for time series: an introduction using R*. Chapman and Hall/CRC.

# A Discrete CVAE for Response Generation on Short-Text Conversation

**Jun Gao[1]*, Wei Bi[2]†, Xiaojiang Liu[2], Junhui Li[1], Guodong Zhou[1], Shuming Shi[2]**
[1]School of Computer Science and Technology, Soochow University, Suzhou, China
imgaojun@gmail.com, {lijunhui,gdzhou}@suda.edu.cn
[2]Tencent AI Lab, Shenzhen, China
{victoriabi, kieranliu, shumingshi}@tencent.com

## Abstract

Neural conversation models such as encoder-decoder models are easy to generate bland and generic responses. Some researchers propose to use the conditional variational autoencoder (CVAE) which maximizes the lower bound on the conditional log-likelihood on a continuous latent variable. With different sampled latent variables, the model is expected to generate diverse responses. Although the CVAE-based models have shown tremendous potential, their improvement of generating high-quality responses is still unsatisfactory. In this paper, we introduce a discrete latent variable with an explicit semantic meaning to improve the CVAE on short-text conversation. A major advantage of our model is that we can exploit the semantic distance between the latent variables to maintain good diversity between the sampled latent variables. Accordingly, we propose a two-stage sampling approach to enable efficient diverse variable selection from a large latent space assumed in the short-text conversation task. Experimental results indicate that our model outperforms various kinds of generation models under both automatic and human evaluations and generates more diverse and informative responses.

## 1 Introduction

Open-domain response generation (Perez-Marin, 2011; Sordoni et al., 2015) for single-round short text conversation (Shang et al., 2015), aims at generating a meaningful and interesting response given a query from human users. Neural generation models are of growing interest in this topic due to their potential to leverage massive conversational datasets on the web. These generation models such as encoder-decoder models (Vinyals and Le, 2015; Shang et al., 2015; Wen et al., 2015), directly build a mapping from the input query to its output response, which treats all query-response pairs uniformly and optimizes the maximum likelihood estimation (MLE). However, when the models converge, they tend to output bland and generic responses (Li et al., 2016a,c; Serban et al., 2016).

Many enhanced encoder-decoder approaches have been proposed to improve the quality of generated responses. They can be broadly classified into two categories (see Section 2 for details): (1) One that does not change the encoder-decoder framework itself. These approaches only change the decoding strategy, such as encouraging diverse tokens to be selected in beam search (Li et al., 2016a,b); or adding more components based on the encoder-decoder framework, such as the Generative Adversarial Network (GAN)-based methods (Xu et al., 2017; Zhang et al., 2018; Li et al., 2017) which add discriminators to perform adversarial training; (2) The second category modifies the encoder-decoder framework directly by incorporating useful information as latent variables in order to generate more specific responses (Yao et al., 2017; Zhou et al., 2017). However, all these enhanced methods still optimize the MLE of the log-likelihood or the complete log-likelihood conditioned on their assumed latent information, and models estimated by the MLE naturally favor to output frequent patterns in training data.

Instead of optimizing the MLE, some researchers propose to use the conditional variational autoencoder (CVAE), which maximizes the lower bound on the conditional data log-likelihood on a continuous latent variable (Zhao et al., 2017; Shen et al., 2017). Open-domain response generation is a one-to-many problem, in which a query can be associated with many valid responses. The CVAE-based models generally assume the latent

*Work done when Jun Gao was interning at Tencent AI Lab.
†Corresponding author



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1898–1908, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

variable follows a multivariate Gaussian distribution with a diagonal covariance matrix, which can capture the latent distribution over all valid responses. With different sampled latent variables, the model is expected to decode diverse responses. Due to the advantage of the CVAE in modeling the response generation process, we focus on improving the performance of the CVAE-based response generation models.

Although the CVAE has achieved impressive results on many generation problems (Yan et al., 2016; Sohn et al., 2015), recent results on response generation show that the CVAE-based generation models still suffer from the low output diversity problem. That is multiple sampled latent variables result in responses with similar semantic meanings. To address this problem, extra guided signals are often used to improve the basic CVAE. Zhao et al. (2017) use dialogue acts to capture the discourse variations in multi-round dialogues as guided knowledge. However, such discourse information can hardly be extracted for short-text conversation.

In our work, we propose a discrete CVAE (DCVAE), which utilizes a discrete latent variable with an explicit semantic meaning in the CVAE for short-text conversation. Our model mitigates the low output diversity problem in the CVAE by exploiting the semantic distance between the latent variables to maintain good diversity between the sampled latent variables. Accordingly, we propose a two-stage sampling approach to enable efficient selection of diverse variables from a large latent space assumed in the short-text conversation task.

To summarize, this work makes three contributions: (1) We propose a response generation model for short-text conversation based on a DCVAE, which utilizes a discrete latent variable with an explicit semantic meaning and could generate high-quality responses. (2) A two-stage sampling approach is devised to enable efficient selection of diverse variables from a large latent space assumed in the short-text conversation task. (3) Experimental results show that the proposed DCVAE with the two-stage sampling approach outperforms various kinds of generation models under both automatic and human evaluations, and generates more high-quality responses. All our code and datasets are available at https://ai.tencent.com/ailab/nlp/dialogue.

## 2 Related Work

In this section, we briefly review recent advancement in encoder-decoder models and CVAE-based models for response generation.

### 2.1 Encoder-decoder models

Encoder-decoder models for short-text conversation (Vinyals and Le, 2015; Shang et al., 2015) maximize the likelihood of responses given queries. During testing, a decoder sequentially generates a response using search strategies such as beam search. However, these models frequently generate bland and generic responses.

Some early work improves the quality of generated responses by modifying the decoding strategy. For example, Li et al. (2016a) propose to use the maximum mutual information (MMI) to penalize general responses in beam search during testing. Some later studies alter the data distributions according to different sample weighting schemes, encouraging the model to put more emphasis on learning samples with rare words (Nakamura et al., 2018; Liu et al., 2018). As can be seen, these methods focus on either pre-processing the dataset before training or post-processing the results in testing, with no change to encoder-decoder models themselves.

Some other work use encoder-decoder models as the basis and add more components to refine the response generation process. Xu et al. (2017) present a GAN-based model with an approximate embedding layer. Zhang et al. (2018) employ an adversarial learning method to directly optimize the lower bounder of the MMI objective (Li et al., 2016a) in model training. These models employ the encoder-decoder models as the generator and focus on how to design the discriminator and optimize the generator and discriminator jointly. Deep reinforcement learning is also applied to model future reward in chatbot after an encoder-decoder model converges (Li et al., 2016c, 2017). The above methods directly integrate the encoder-decoder models as one of their model modules and still do not actually modify the encoder-decoder models.

Many attentions have turned to incorporate useful information as latent variables in the encoder-decoder framework to improve the quality of generated responses. Yao et al. (2017) consider that a response is generated by a query and a precomputed cue word jointly. Zhou et al. (2017) uti-

lize a set of latent embeddings to model diverse responding mechanisms. Xing et al. (2017) introduce pre-defined topics from an external corpus to augment the information used in response generation. Gao et al. (2019) propose a model that infers latent words to generate multiple responses. These studies indicate that many factors in conversation are useful to model the variation of a generated response, but it is nontrivial to extract all of them. Also, these methods still optimize the MLE of the complete log-likelihood conditioned on their assumed latent information, and the model optimized with the MLE naturally favors to output frequent patterns in the training data. Note that we apply a similar latent space assumption as used in (Yao et al., 2017; Gao et al., 2019), i.e. the latent variables are words from the vocabulary. However, they use a latent word in a factorized encoder-decoder model, but our model uses it to construct a discrete CVAE and our optimization algorithm is entirely different from theirs.

### 2.2 The CVAE-based models

A few works indicate that it is worth trying to apply the CVAE to dialogue generation which is originally used in image generation (Yan et al., 2016; Sohn et al., 2015) and optimized with the variational lower bound of the conditional log-likelihood. For task-oriented dialogues, Wen et al. (2017) use the latent variable to model intentions in the framework of neural variational inference. For chit-chat multi-round conversations, Serban et al. (2017) model the generative process with multiple levels of variability based on a hierarchical sequence-to-sequence model with a continuous high-dimensional latent variable. Zhao et al. (2017) make use of the CVAE and the latent variable is used to capture discourse-level variations. Gu et al. (2019) propose to induce the latent variables by transforming context-dependent Gaussian noise. Shen et al. (2017) present a conditional variational framework for generating specific responses based on specific attributes. Yet, it is observed in other tasks such as image captioning (Wang et al., 2017) and question generation (Fan et al., 2018) that the CVAE-based generation models suffer from the low output diversity problem, i.e. multiple sampled variables point to the same generated sequences. In this work, we utilize a discrete latent variable with an interpretable meaning to alleviate this low output diversity problem on short-text conversation.

We find that Zhao et al. (2018) make use of a set of discrete variables that define high-level attributes of a response. Although they interpret meanings of the learned discrete latent variables by clustering data according to certain classes (e.g. dialog acts), such latent variables still have no exact meanings. In our model, we connect each latent variable with a word in the vocabulary, thus each latent variable has an exact semantic meaning. Besides, they focus on multi-turn dialogue generation and presented an unsupervised discrete sentence representation learning method learned from the context while our concentration is primarily on single-turn dialogue generation with no context information.

## 3 Proposed Models

### 3.1 DCVAE and Basic Network Modules

Following previous CVAE-based generation models (Zhao et al., 2017), we introduce a latent variable $z$ for each input sequence and our goal is to maximize the lower bound on the conditional data log-likelihood $p(\mathbf{y}|\mathbf{x})$, where $\mathbf{x}$ is the input query sequence and $\mathbf{y}$ is the target response sequence:

$$\begin{aligned}\log p(\mathbf{y}|\mathbf{x}) \geq\ & \mathbb{E}_{z\sim q(z|\mathbf{y},\mathbf{x})}[\log p(\mathbf{y}|\mathbf{x},z)] \\ & -D_{KL}(q(z|\mathbf{y},\mathbf{x})||p(z|\mathbf{x})). \quad (1)\end{aligned}$$

Here, $p(z|\mathbf{x})$/$q(z|\mathbf{y},\mathbf{x})$/$p(\mathbf{y}|\mathbf{x},z)$ is parameterized by the prior/posterior/generation network respectively. $D_{KL}(q(z|\mathbf{y},\mathbf{x})||p(z|\mathbf{x}))$ is the Kullback-Leibler (KL) divergence between the posterior and prior distribution. Generally, $z$ is set to follow a Gaussian distribution in both the prior and posterior networks. As mentioned in the related work, directly using the above CVAE formulation causes the low output diversity problem. This observation is also validated in the short-text conversation task in our experiments.

Now, we introduce our basic discrete CVAE formulation to alleviate the low output diversity problem. We change the continuous latent variable $z$ to a discrete latent one with an explicit interpretable meaning, which could actively control the generation of the response. An intuitive way is to connect each latent variable with a word in the vocabulary. With a sampled latent $z$ from the prior (in testing)/posterior network (in training), the generation network will take the query representation together with the word embedding of this latent

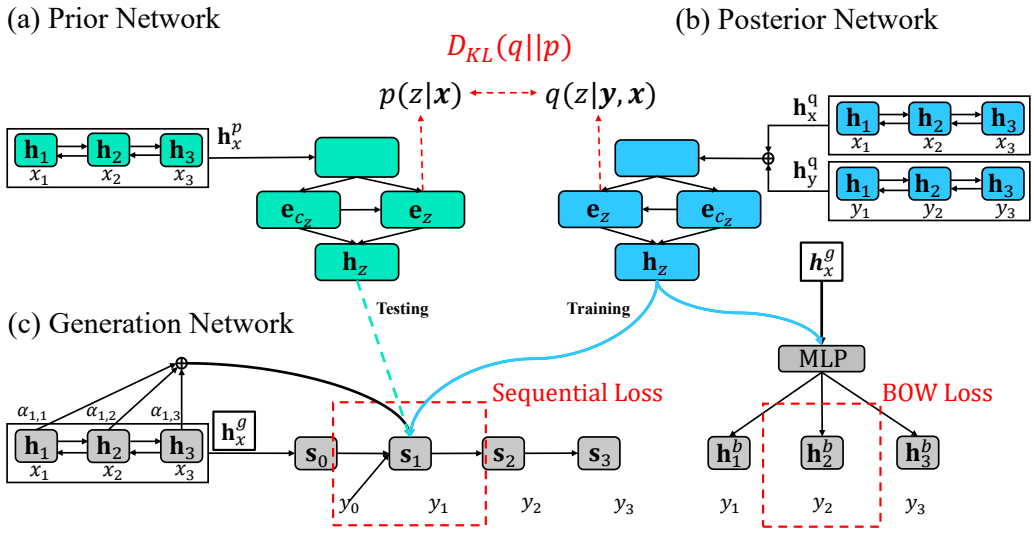


Figure 1: The architecture of the proposed discrete CVAE. $\mathbf{e}_{c_z}$ and $\mathbf{e}_z$ are embeddings of a cluster and a word sampled from the estimated discrete distributions. $\mathbf{e}_{c_z}$ is only applied when the two-stage sampling approach in Section 3.2 is used. If $\mathbf{e}_{c_z}$ is applied, the latent representation $\mathbf{h}_z$ is the sum of $\mathbf{e}_{c_z}$ and $\mathbf{e}_z$; otherwise, $\mathbf{h}_z$ is $\mathbf{e}_z$. $\alpha$ denotes the attention weight. $\oplus$ denotes the sum of input vectors.

variable as the input to decode the response. Here, we assume that a single word is enough to drive the generation network to output diverse responses for short text conversation, in which the response is generally short and compact.

A major advantage of our DCVAE is that for words with far different meanings, their word embeddings (especially that we use a good pre-trained word embedding corpus) generally have a large distance and drive the generation network to decode scattered responses, thus improve the output diversity. In the standard CVAE, $z$'s assumed in a continuous space may not maintain the semantic distance as in the embedding space and diverse $z$'s may point to the same semantic meaning, in which case the generation network is hard to train well with such confusing information. Moreover, we can make use of the semantic distance between latent variables to perform better sampling to approximate the objective during optimization, which will be introduced in Section 3.2.

The latent variable $z$ is thus set to follow a categorical distribution with each dimension corresponding to a word in the vocabulary. Therefore the prior and posterior networks should output categorical probability distributions:

$$p_\theta(z|\mathbf{x}) = \text{softmax}(g_\theta(\mathbf{x})), \quad (2)$$

$$q_\phi(z|\mathbf{y},\mathbf{x}) = \text{softmax}(f_\phi(\mathbf{y},\mathbf{x})), \quad (3)$$

where $\theta$ and $\phi$ are parameters of the two networks respectively. The KL distance of these two distributions can be calculated in a closed form solution:

$$D_{KL}(q(z|\mathbf{y},\mathbf{x})||p(z|\mathbf{x})) = \sum_{z\in Z} q(z|\mathbf{y},\mathbf{x}) \log \frac{q(z|\mathbf{y},\mathbf{x})}{p(z|\mathbf{x})}, \quad (4)$$

where $Z$ contains all words in the vocabulary. In the following, we present the details of the prior, posterior and generation network.

**Prior network** $p(z|\mathbf{x})$: It aims at inferring the latent variable $z$ given the input sequence $x$. We first obtain an input representation $\mathbf{h}_\mathbf{x}^p$ by encoding the input query $\mathbf{x}$ with a bi-directional GRU and then compute $g_\theta(\mathbf{x})$ in Eq. 2 as follows:

$$g_\theta(\mathbf{x}) = \mathbf{W}_2 \cdot \tanh(\mathbf{W}_1 \mathbf{h}_x^p + \mathbf{b}_1) + \mathbf{b}_2, \quad (5)$$

where $\theta$ contains parameters in both the bidirectional GRU and Eq. 5.

**Posterior network** $q(z|\mathbf{y},\mathbf{x})$: It infers a latent variable $z$ given a input query $\mathbf{x}$ and its target response $\mathbf{y}$. We construct both representations for the input and the target sequence by separated bi-directional GRU's, then add them up to compute $f_\phi(\mathbf{y},\mathbf{x})$ in Eq. 3 to predict the probability of $z$:

$$f_\phi(\mathbf{y},\mathbf{x}) = \mathbf{W}_4 \cdot \tanh(\mathbf{W}_3(\mathbf{h}_x^q + \mathbf{h}_y^q) + \mathbf{b}_3) + \mathbf{b}_4, \quad (6)$$

where $\phi$ contains parameters in the two encoding functions and Eq. 6. Note that the parameters of

the encoding functions are not shared in the prior and posterior network.

**Generation network** $p(\mathbf{y}|\mathbf{x}, z)$: We adopt an encoder-decoder model with attention (Luong et al., 2015) used in the decoder. With a sampled latent variable $z$, a typical strategy is to combine its representation, which in this case is the word embedding $\mathbf{e}_z$ of $z$, only in the beginning of decoding. However, many previous works observe that the influence of the added information will vanish over time (Yao et al., 2017; Gao et al., 2019). Thus, after obtaining an attentional hidden state at each decoding step, we concatenate the representation $\mathbf{h}_z$ of the latent variable and the current hidden state to produce a final output in our generation network.

### 3.2 A Two-Stage Sampling Approach

When the CVAE models are optimized, they tend to converge to a solution with a vanishingly small KL term, thus failing to encode meaningful information in $z$. To address this problem, we follow the idea in (Zhao et al., 2017), which introduces an auxiliary loss that requires the decoder in the generation network to predict the bag-of-words in the response $\mathbf{y}$. Specifically, the response $\mathbf{y}$ is now represented by two sequences simultaneously: $\mathbf{y}_o$ with word order and $\mathbf{y}_{bow}$ without order. These two sequences are assumed to be conditionally independent given $z$ and $\mathbf{x}$. Then our training objective can be rewritten as:

$$\begin{aligned} J(\Theta) =& \mathbb{E}_{z\sim q(z|\mathbf{y},\mathbf{x})}[\log p(\mathbf{y}|\mathbf{x}, z)] \\ & - D_{KL}(q(z|\mathbf{y}, \mathbf{x})||p(z|\mathbf{x})) \\ & + \mathbb{E}_{z\sim q(z|\mathbf{y},\mathbf{x})}[\log p(\mathbf{y}_{bow}|\mathbf{x}, z)], \end{aligned} \tag{7}$$

where $p(\mathbf{y}_{bow}|\mathbf{x}, z)$ is obtained by a multilayer perceptron $\mathbf{h}^b = \text{MLP}(\mathbf{x}, z)$:

$$p(\mathbf{y}_{bow}|\mathbf{x}, z) = \prod_{t=1}^{|\mathbf{y}|} \frac{\exp(h^b_{y_t})}{\sum_{j\in V}\exp(h^b_j)}, \tag{8}$$

where $|\mathbf{y}|$ is the length of $\mathbf{y}$, $y_t$ is the word index of $t$-th word in $\mathbf{y}$, and $V$ is the vocabulary size.

During training, we generally approximate $\mathbb{E}_{z\sim q(z|\mathbf{y},\mathbf{x})}[\log p(\mathbf{y}|\mathbf{x}, z)]$ by sampling $N$ times of $z$ from the distribution $q(z|\mathbf{y}, \mathbf{x})$. In our model, the latent space is discrete but generally large since we set it as the vocabulary in the dataset [1]. The vocabulary consists of words that are similar in syntactic or semantic. Directly sampling $z$ from the categorical distribution in Eq. 3 cannot make use of such word similarity information.

Hence, we propose to modify our model in Section 3.1 to consider the word similarity for sampling multiple accurate and diverse latent $z$'s. We first cluster $z \in Z$ into $K$ clusters $c_1, \ldots, c_K$. Each $z$ belongs to only one of the $K$ clusters and dissimilar words lie in distinctive groups. We use the K-means clustering algorithm to group $z$'s using a pre-trained embedding corpus (Song et al., 2018). Then we revise the posterior network to perform a two-stage cluster sampling by decomposing $q(z|\mathbf{y}, \mathbf{x})$ as :

$$\begin{aligned} q(z|\mathbf{y}, \mathbf{x}) &= \sum_k q(z|\mathbf{x}, \mathbf{y}, c_k)q(c_k|\mathbf{x}, \mathbf{y}) \\ &= q(z|\mathbf{x}, \mathbf{y}, c_{k_z})q(c_{k_z}|\mathbf{x}, \mathbf{y}). \end{aligned} \tag{9}$$

That is, we first compute $q(c_{k_z}|\mathbf{y}, \mathbf{x})$, which is the probability of the cluster that $z$ belongs to conditioned on both $\mathbf{x}$ and $\mathbf{y}$. Next, we compute $q(z|\mathbf{x}, \mathbf{y}, c_{k_z})$, which is the probability distribution of $z$ conditioned on the $\mathbf{x}$, $\mathbf{y}$ and the cluster $c_{k_z}$. When we perform sampling from $q(z|\mathbf{x}, \mathbf{y})$, we can exploit the following two-stage sampling approach: first sample the cluster based on $q(c_k|\mathbf{x}, \mathbf{y})$; next sample a specific $z$ from $z$'s within the sampled cluster based on $q(z|\mathbf{x}, \mathbf{y}, c_{k_z})$.

Similarly, we can decompose the prior distribution $p(z|\mathbf{x})$ accordingly for consistency:

$$p(z|\mathbf{x}) = p(z|\mathbf{x}, c_{k_z})p(c_{k_z}|\mathbf{x}). \tag{10}$$

In testing, we can perform the two-stage sampling according to $p(c_k|\mathbf{x})$ and $p(z|\mathbf{x}, c_{k_z})$. Our full model is illustrated in Figure 1.

**Network structure modification:** To modify the network structure for the two-stage sampling method, we first compute the probability of each cluster given $\mathbf{x}$ in the prior network (or $\mathbf{x}$ and $\mathbf{y}$ in the posterior network) with a softmax layer (Eq. 5 or Eq. 6 followed by a softmax function). We then add the input representation and the cluster embedding $\mathbf{e}_{c_z}$ of a sampled cluster $c_z$, and use another softmax layer to compute the probability of each $z$ within the sampled cluster. In the generation network, the representation of $z$ is the sum of the cluster embedding $\mathbf{e}_{c_z}$ and its word embedding $\mathbf{e}_z$.

**Network pre-training:** To speed up the convergence of our model, we pre-extract keywords from

[1] Note that we remove special tokens including UNK (unknown token), BOS (start of sentence) and EOS (end of sentence) in the latent space such that our model will only select meaningful words as the latent variables.

each query using the TF-IDF method. Then we use these keywords to pre-train the prior and posterior networks. The generation network is not pre-trained because in practice it converges fast in only a few epochs.

## 4 Experimental Settings

Next, we describe our experimental settings including the dataset, implementation details, all compared methods, and the evaluation metrics.

### 4.1 Dataset

We conduct our experiments on a short-text conversation benchmark dataset (Shang et al., 2015) which contains about 4 million post-response pairs from the Sina Weibo [2], a Chinese social platforms. We employ the Jieba Chinese word segmenter [3] to tokenize the queries and responses into sequences of Chinese words. We use a vocabulary of 50,000 words (a mixture of Chinese words and characters), which covers 99.98% of words in the dataset. All other words are replaced with <UNK>. We randomly hold out two subsets as the development and test dataset, each containing 900 pairs.

### 4.2 Implementation Details

We use single-layer bi-directional GRU for the encoder in the prior/posterior/generation network, and one-layer GRU for the decoder in the generation network. The dimension of all hidden vectors is 1024. The cluster embedding dimension is 620. Except that the word embeddings are initialized by the word embedding corpus (Song et al., 2018), all other parameters are initialized by sampling from a uniform distribution $[-0.1, 0.1]$. The batch size is 128. We use Adam optimizer with a learning rate of 0.0001. For the number of clusters $K$ in our method, we evaluate four different values $(5, 10, 100, 1000)$ using automatic metrics and set $K$ to 10 which tops the four options empirically. It takes about one day for every two epochs of our model on a Tesla P40 GPU, and we train ten epochs in total. During testing, we use beam search with a beam size of 10.

### 4.3 Compared Methods

In our work, we focus on comparing various methods that model $p(\mathbf{y}|\mathbf{x})$ differently. We compare our proposed discrete CVAE (**DCVAE**) with the two-stage sampling approach to three categories of response generation models:

1. Baselines: **Seq2seq**, the basic encoder-decoder model with soft attention mechanism (Bahdanau et al., 2015) used in decoding and beam search used in testing; **MMI-bidi** (Li et al., 2016a), which uses the MMI to re-rank results from beam search.

2. **CVAE** (Zhao et al., 2017): We adjust the original work which is for multi-round conversation for our single-round setting. For a fair comparison, we utilize the same keywords used in our network pre-training as the knowledge-guided features in this model.

3. Other enhanced encoder-decoder models: Hierarchical Gated Fusion Unit (**HGFU**) (Yao et al., 2017), which incorporates a cue word extracted using pointwise mutual information (PMI) into the decoder to generate meaningful responses; Mechanism-Aware Neural Machine (**MANM**) (Zhou et al., 2017), which introduces latent embeddings to allow for multiple diverse response generation.

Here, we do not compare RL/GAN-based methods because all our compared methods can replace their objectives with the use of reward functions in the RL-based methods or add a discriminator in the GAN-based methods to further improve the overall performance. However, these are not the contribution of our work, which we leave to future work to discuss the usefulness of our model as well as other enhanced generation models combined with the RL/GAN-based methods.

### 4.4 Evaluation

To evaluate the responses generated by all compared methods, we compute the following automatic metrics on our test set:

1. BLEU: BLEU-n measures the average n-gram precision on a set of reference responses. We report BLEU-n with n=1,2,3,4.

2. Distinct-1 & distinct-2 (Li et al., 2016a): We count the numbers of distinct uni-grams and bi-grams in the generated responses and divide the numbers by the total number of generated uni-grams and bi-grams in the test set. These metrics can be regarded as an automatic metric to evaluate the diversity of the responses.

[2] http://weibo.com
[3] https://github.com/fxsjy/jieba

| Method | Automatic Metrics | | | | | | Human Evaluation | | |
|---|---|---|---|---|---|---|---|---|---|
| | BLEU-1 | BLEU-2 | BLEU-3 | BLEU-4 | dist-1 | dist-2 | quality | accept | good |
| Seq2seq | $0.63 \pm 0.14$ | $0.49 \pm 0.16$ | $0.35 \pm 0.14$ | $0.21 \pm 0.12$ | 0.03 | 0.08 | $1.64 \pm 0.30$ | 49% | 15% |
| MMI-bidi | $0.54 \pm 0.17$ | $0.39 \pm 0.18$ | $0.28 \pm 0.15$ | $0.17 \pm 0.13$ | 0.04 | 0.11 | $1.71 \pm 0.34$ | 52% | 19% |
| CVAE | $0.60 \pm 0.13$ | $0.43 \pm 0.16$ | $0.30 \pm 0.14$ | $0.18 \pm 0.11$ | 0.03 | 0.06 | $1.60 \pm 0.33$ | 47% | 13% |
| MANM | $0.62 \pm 0.14$ | $0.48 \pm 0.15$ | $0.34 \pm 0.14$ | $0.22 \pm 0.12$ | 0.05 | 0.14 | $1.73 \pm 0.35$ | 53% | 21% |
| HGFU | $0.52 \pm 0.11$ | $0.38 \pm 0.14$ | $0.27 \pm 0.12$ | $0.16 \pm 0.11$ | 0.08 | 0.27 | $1.63 \pm 0.39$ | 51% | 12% |
| DCVAE | $\mathbf{0.64 \pm 0.14}$ | $\mathbf{0.49 \pm 0.16}$ | $\mathbf{0.35 \pm 0.15}$ | $\mathbf{0.22 \pm 0.13}$ | 0.08 | 0.24 | $\mathbf{2.03 \pm 0.34}$ | **73%** | **30%** |

Table 1: The automatic and human evaluation results of all compared methods. Note that the acceptable ratio is the percentage of responses with 2 or 3 points.

Three annotators from a commercial annotation company are recruited to conduct our human evaluation. Responses from different models are shuffled for labeling. 300 test queries are randomly selected out, and annotators are asked to independently score the results of these queries with different points in terms of their quality: (1) Good (3 points): The response is grammatical, semantically relevant to the query, and more importantly informative and interesting; (2) Acceptable (2 points): The response is grammatical, semantically relevant to the query, but too trivial or generic (e.g.,"我不知道(I don't know)", "我也是(Me too)", "我喜欢(I like it)" etc.); (3) Failed (1 point): The response has grammar mistakes or irrelevant to the query.

## 5 Experimental Results and Analysis

In the following, we will present results of all compared methods and conduct a case study on such results. Then, we will perform further analysis of our proposed method by varying different settings of the components designed in our model.

### 5.1 Results on All Compared Methods

Results on automatic metrics are shown on the left-hand side of Table 1. From the results we can see that our proposed DCVAE achieves the best BLEU scores and the second best distinct ratios. The HGFU has the best dist-2 ratio, but its BLEU scores are the worst. These results indicate that the responses generated by the HGFU are less close to the ground true references. Although the automatic evaluation generally indicates the quality of generated responses, it can not accurately evaluate the generated response and the automatic metrics may not be consistent with human perceptions (Liu et al., 2016). Thus, we consider human evaluation results more reliable.

For the human evaluation results on the right-hand side of Table 1, we show the mean and standard deviation of all test results as well as the percentage of acceptable responses (2 or 3 points) and good responses (3 points only). Our proposed DCVAE has the best quality score among all compared methods. Moreover, DCVAE achieves a much higher good ratio, which means it generates more informative and interesting responses. Besides, the HGFU's acceptable and good ratios are much lower than our model indicating that it may not maintain enough response relevance when encouraging diversity. This is consistent with the results of the automatic evaluation in Table 1. We also notice that the CVAE achieves the worst human annotation score. This validates that the original CVAE for open-domain response generation does not work well and our proposed DCVAE is an effective way to improve the CVAE for better output diversity.

### 5.2 Case Study

Figure 2 shows four example queries with their responses generated by all compared methods. The Seq2seq baseline tends to generate less informative responses. Though MMI-bidi can select different words to be used, its generated responses are still far from informative. MANM can avoid generating generic responses in most cases, but sometimes its generated response is irrelevant to the query, as shown in the left bottom case. Moreover, the latent responding mechanisms in MANM have no explicit or interpretable meaning. Similar results can be observed from HGFU. If the PMI selects irrelevant cue words, the resulting response may not be relevant. Meanwhile, responses generated by our DCVAE are more informative as well as relevant to input queries.

### 5.3 Different Sizes of the Latent Space

We vary the size of the latent space (i.e., sampled word space $Z$) used in our proposed DCVAE. Fig-

| | | |
|---|---|---|
| **Input** | 刚和卫哲通了电话，他心态很平和，<br>说现在有时间找他打球了。<br>I just talked to Wei Zhe on the phone and he was calm.<br>I said now I had time to play ball with him. | 房子，小院，车子，落英缤纷的门前，<br>静谧的生活。<br>The house, the courtyard, the car,<br>the door of fallen petals, quiet life. |
| **Seq2seq** | 是的，我也是这样。<br>Yes, me too. | 这是在哪啊？在哪里？<br>Where is this? Where? |
| **MMI-bidi** | 恭喜恭喜！恭喜恭喜！<br>Congratulations! Congratulations! | 哇，好漂亮，好想去<br>Wow, it's beautiful, I want to go there |
| **MANM** | 我也想知道他是谁<br>I wonder who he is, too | 好喜欢，好喜欢。<br>I like it, I like it. |
| **HGFU** | 我也想知道他是谁[据说]<br>I wonder who he is, too [reputedly] | 好美的地方，好美的地方[学习]<br>What a beautiful place, what a beautiful place [study] |
| **DCVAE** | 哈哈，我也喜欢看你打球[打球]<br>Haha, I like watching you play ball, too [play ball] | 好漂亮的房子，好喜欢[生活]<br>What a beautiful house, I love it [life] |
| **Input** | 这位雷神长得有点眼熟<br>This god of thunder looks a little familiar | 下午媒体公开课！现在睡了…晚上见<br>I have an open media class this afternoon!<br>Now I'm going to bed...See you tonight |
| **Seq2seq** | 哈哈哈哈哈，笑死我了！<br>Hahahahaha, you are killing me! | 辛苦了，注意身体！<br>You've been working hard, take care of your health! |
| **MMI -bidi** | 笑死我了<br>You are killing me | 辛苦了，好好休息！<br>You've been working hard, have a good rest! |
| **MANM** | 这是要闹哪样啊。<br>What is this for. | 辛苦了，注意身体！<br>You've been working hard, take care of your health! |
| **HGFU** | 尼玛，这是要逆天啊！[哇塞]<br>Holy crap, you are going against the world！[wow] | 加油加油加油加油加油[尼玛]<br>Come on, come on, come on [holy crap] |
| **DCVAE** | 雷神长的好漂亮啊。[这位]<br>The god of thunder is so beautiful. [this] | 我也想睡了，明天还要上班[下午]<br>I want to sleep too, I need to work tomorrow [afternoon] |

Figure 2: Examples of the generated responses. The sampled latent words ($z$) are showed in the brackets.

ure 3 shows the automatic and human evaluation results on the latent space setting to the top 10k, 20k, all words in the vocabulary. On the automatic evaluation results, if the sampled latent space is getting larger, the BLEU-4 score increases but the distinct ratios drop. We find out that though the DCVAE with a small latent space has a higher distinct-1/2 ratio, many generated sentences are grammatically incorrect. This is also why the BLEU-4 score decreases. On the human evaluation results, all metrics improve with the use of a larger latent space. This is consistent with our motivation that open-domain short-text conversation covers a wide range of topics and areas, and the top frequent words are not enough to capture the content of most training pairs. Thus a small latent space, i.e. the top frequent words only, is not feasible to model enough latent information and a large latent space is generally favored in our proposed model.

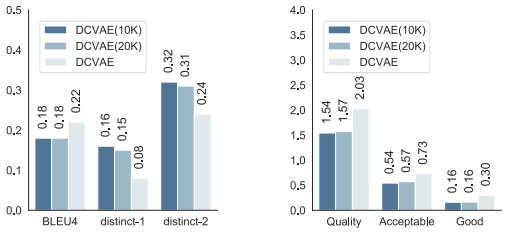


Figure 3: Different sizes of the latent space used in the DCVAE: automatic evaluation (left) and human evaluation (right).

## 5.4 Analysis on the Two-Stage Sampling

We further look into whether the two-stage sampling method is effective in the proposed DCVAE. Here, the One-Stage method corresponds to the basic formulation in Section 3.1 with no use of the clustering information in the prior or posterior network. Results on both automatic and human evaluation metrics are shown in Figure. 4(a) and 4(b). We can observe that the performance of the DCVAE without the two-stage sampling method drops drastically. This means that the proposed two-stage sampling method is important for the DCVAE to work well.

Besides, to validate the effectiveness of clustering, we implemented a modified DCVAE (DCVAE-CD) that uses a pure categorical distribution in which each variable has no exact meaning. That is, the embedding of each latent variable does not correspond to any word embedding. Automatic evaluation results of this modified model are shown in Figure. 4(c). We can see that DCVAE-CD performs worse, which means the distribution on word vocabulary is important in our model.

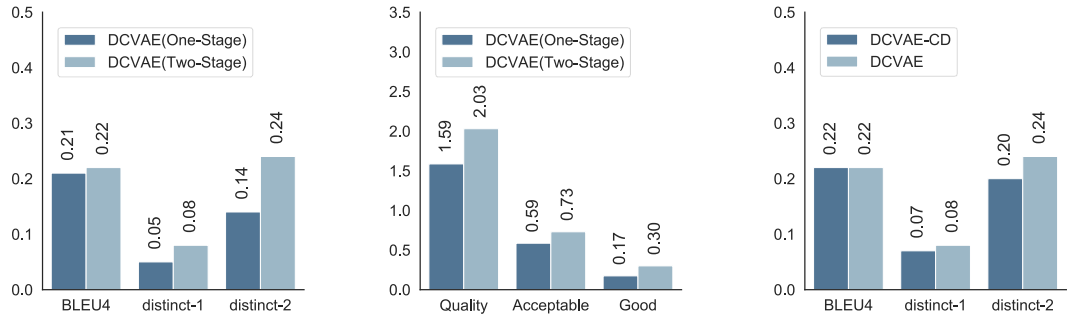


(a) (b) (c)

Figure 4: (a)/(b): Automatic/human evaluation on the DCVAE with/without the two-stage sampling approach. (c): Automatic evaluation on our proposed DCVAE and the modified DCVAE that uses a pure categorical distribution (DCVAE-CD) in which each variable has no exact meaning.

## 6 Conclusion

In this paper, we have presented a novel response generation model for short-text conversation via a discrete CVAE. We replace the continuous latent variable in the standard CVAE by an interpretable discrete variable, which is set to a word in the vocabulary. The sampled latent word has an explicit semantic meaning, acting as a guide to the generation of informative and diverse responses. We also propose to use a two-stage sampling approach to enable efficient selection of diverse variables from a large latent space, which is very essential for our model. Experimental results show that our model outperforms various kinds of generation models under both automatic and human evaluations.

## Acknowledgements

This work was supported by National Natural Science Foundation of China (Grant No. 61751206, 61876120).

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 2015 International Conference on Learning Representations*.

Zhihao Fan, Zhongyu Wei, Siyuan Wang, Yang Liu, and Xuanjing Huang. 2018. A reinforcement learning framework for natural question generation using bi-discriminators. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1763–1774.

Jun Gao, Wei Bi, Xiaojiang Liu, Junhui Li, and Shuming Shi. 2019. Generating multiple diverse responses for short-text conversation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 6383–6390.

Xiaodong Gu, Kyunghyun Cho, Jung-Woo Ha, and Sunghun Kim. 2019. DialogWAE: Multimodal response generation with conditional wasserstein auto-encoder. In *Proceedings of the 2019 International Conference on Learning Representations*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016b. A simple, fast diverse decoding algorithm for neural generation. *arXiv preprint arXiv:1611.08562*.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016c. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202.

Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2157–2169.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2122–2132.

Yahui Liu, Wei Bi, Jun Gao, Xiaojiang Liu, Jian Yao, and Shuming Shi. 2018. Towards less generic responses in neural conversation models: A statistical re-weighting method. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2769–2774.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Ryo Nakamura, Katsuhito Sudoh, Koichiro Yoshino, and Satoshi Nakamura. 2018. Another diversity-promoting objective function for neural dialogue generation. *arXiv preprint arXiv:1811.08100*.

Diana Perez-Marin. 2011. *Conversational Agents and Natural Language Interaction: Techniques and Effective Practices: Techniques and Effective Practices*. IGI Global.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of AAAI Conference on Artificial Intelligence*, pages 3776–3784.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Proceedings of AAAI Conference on Artificial Intelligence*, pages 3295–3301.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 1577–1586.

Xiaoyu Shen, Hui Su, Yanran Li, Wenjie Li, Shuzi Niu, Yang Zhao, Akiko Aizawa, and Guoping Long. 2017. A conditional variational framework for dialog generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 504–509.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. 2015. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*, pages 3483–3491.

Yan Song, Shuming Shi, Jing Li, and Haisong Zhang. 2018. Directional skip-gram: Explicitly distinguishing left and right context for word embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 175–180.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 196–205.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Liwei Wang, Alexander Schwing, and Svetlana Lazebnik. 2017. Diverse and accurate image description using a variational auto-encoder with an additive gaussian encoding space. In *Advances in Neural Information Processing Systems*, pages 5756–5766.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721.

Tsung-Hsien Wen, Yishu Miao, Phil Blunsom, and Steve Young. 2017. Latent intention dialogue models. In *International Conference on Machine Learning*, pages 3732–3741.

Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2017. Topic aware neural response generation. In *Proceedings of AAAI Conference on Artificial Intelligence*, pages 3351–3357.

Zhen Xu, Bingquan Liu, Baoxun Wang, SUN Chengjie, Xiaolong Wang, Zhuoran Wang, and Chao Qi. 2017. Neural response generation via gan with an approximate embedding layer. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 617–626.

Xinchen Yan, Jimei Yang, Kihyuk Sohn, and Honglak Lee. 2016. Attribute2image: Conditional image generation from visual attributes. In *European Conference on Computer Vision*, pages 776–791. Springer.

Lili Yao, Yaoyuan Zhang, Yansong Feng, Dongyan Zhao, and Rui Yan. 2017. Towards implicit content-introducing for generative short-text conversation systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2190–2199.

Yizhe Zhang, Michel Galley, Jianfeng Gao, Zhe Gan, Xiujun Li, Chris Brockett, and Bill Dolan. 2018. Generating informative and diverse conversational responses via adversarial information maximization. In *Advances in Neural Information Processing Systems*, pages 1810–1820.

Tiancheng Zhao, Kyusong Lee, and Maxine Eskenazi. 2018. Unsupervised discrete sentence representation learning for interpretable neural dialog generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 654–664.

Ganbin Zhou, Ping Luo, Rongyu Cao, Fen Lin, Bo Chen, and Qing He. 2017. Mechanism-aware neural machine for dialogue response generation. In *Proceedings of AAAI Conference on Artificial Intelligence*, pages 3400–3407.

# Who Is Speaking to Whom? Learning to Identify Utterance Addressee in Multi-Party Conversations

**Ran Le**[13]*, **Wenpeng Hu**[2*], **Mingyue Shang**[1*], **Zhenjun You**[2*],
**Lidong Bing** [4], **Dongyan Zhao**[13] and **Rui Yan**[13†]
[1] Center for Data Science, AAIS, Peking University, Beijing, China
[2] School of Mathematical Sciences, Peking University, Beijing, China
[3] Wangxuan Institute of Computer Technology, Peking University, Beijing, China,
[4] R&D Center Singapore, Machine Intelligence Technology, Alibaba DAMO Academy
{leran.pku}@gmail.com {l.bing}@alibaba-inc.com
{wenpeng.hu,shangmy,youzhenjunpku,zhaody,ruiyan}@pku.edu.cn

## Abstract

Previous research on dialogue systems generally focuses on the conversation between two participants, yet multi-party conversations which involve more than two participants within one session bring up a more complicated but realistic scenario. In real multi-party conversations, we can observe who is speaking, but the addressee information is not always explicit. In this paper, we aim to tackle the challenge of identifying all the missing addressees in a conversation session. To this end, we introduce a novel who-to-whom (**W2W**) model which models users and utterances in the session jointly in an interactive way. We conduct experiments on the benchmark Ubuntu Multi-Party Conversation Corpus and the experimental results demonstrate that our model outperforms baselines with consistent improvements.

## 1 Introduction

As an essential aspect of artificial intelligence, dialogue systems have attracted extensive attention in recent studies (Vinyals and Le, 2015; Serban et al., 2016). Researchers have paid great efforts to understand conversations between two participants, either single-turn (Li et al., 2016a; Shang et al., 2015; Vinyals and Le, 2015) or multi-turn (Zhou et al., 2016; Yan et al., 2016; Tao et al., 2019a,b), and achieved encouraging results. A more general and challenging scenario is that a conversation may involve more than two interlocutors conversing among each other (Uthus and Aha, 2013; Hu et al., 2019), which is known as multi-party conversation. Ubuntu Internet Relay Chat channel (IRC) is a multi-party conversation scenario as shown in Table 1. Generally, each utterance is associated with a speaker and one or more addressees in the conversation. Such a characteristic leads to complex speaker-addressee interactions. As a result, the speaker and addressee roles associated with utterances are constantly changing among multiple users across different turns. Such speaker and addressee information could be essential in many multi-party conversation scenarios including group meeting, debating and forum discussion. Therefore, compared to two-party conversations, a unique issue of multi-party conversations is to understand who is speaking to whom.

Table 1: An example of the multi-party conversation in the IRC dataset. Not all the addressees are specified.

| Speaker | Utterance | Addressee |
|---|---|---|
| User 1 | "Good point, tmux is the thing I miss." | – |
| User 1 | "Cool thanks for ur help." @User 4 | User 4 |
| User 2 | "Ahha, you r using something like cpanel." | – |
| User 3 | "Yeah 1.4.0 exactly." @User 2 | User 2 |
| User 4 | "my pleasure :)" | – |

In real scenarios of multi-party conversations, an interesting phenomenon is that the speakers do not usually designate an addressee explicitly. This phenomenon also accords with our statistic analysis on the IRC dataset. We found that around 66% utterances missing explicit addressee information. That means when modeling such multi-party conversations, one may have to *guess* who is speaking to whom in order to understand the utterance correspondence as well as the stream structure of multi-party conversations.

Given a multi-party conversation where part of the addressees are unknown, previous work mainly focuses on predicting the addressee of only the last utterance. Ouchi and Tsuboi (2016) proposed to scan the conversation session and track the speaker's state based on the utterance content at each step. On this basis, Zhang et al. (2017) introduced a speaker interaction model that tracks all users' states according to their roles in the session. They both fused the representations of the last speaker and utterance as a query, and a match-

*Equal contribution.
†Corresponding author.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1909–1919, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

ing network is utilized to calculate the matching degree between the query and each listener. The listener with the highest matching score is selected as the predicted addressee.

However, in practice, it is more helpful to predict all the missing addressees rather than only the last one in understanding the whole conversation. And it also benefits for both building a group-based chatbot and clustering users based on what they have said. Therefore, we propose a new task of identifying the addressees of all the missing utterances given a multi-party conversation session where part of the addressees are unspecified. To this end, we propose a novel Who-to-Whom (W2W) model which jointly models users and utterances in the multi-party conversation and predicts all the missing addressees in a uniform framework.[1] Our contributions are as follows:

- We introduce a new task of understanding who speaks to whom given an entire conversation session as well as a benchmark system.
- To capture the correlation within users and utterances in multi-party conversations, we propose an interactive representation learning approach to jointly learn the representations of users and utterances and enhance them mutually.
- The proposed approach (W2W) considers both previous and subsequent information in the session while incorporating the correlation with users and utterances. For conversations with complex structures, W2W models them in a uniform way and could handle any kind of occasion even when all the addressee information is missing.

## 2 Related Work

In this section, we briefly review recent works and progresses on multi-party conversations.

Multi-party conversations, as a general case of multi-turn conversations (Li et al., 2017, 2016c; Yan et al., 2016; Serban et al., 2016) involve more than two participants. In addition to the representation of learning for utterances, another key issue is to model multiple participants in the conversations. It is intuitive to introduce multiple user embeddings for multi-party conversations, either as persona-dependent embeddings (Li et al., 2016b), or as persona-independent embeddings (Ouchi and Tsuboi, 2016; Zhang et al., 2017; Meng et al., 2017). Recently, some researchers utilized users' information based on different roles in conversations, such as *senders* and *recipients* (Chen et al., 2017; Chi et al., 2017; Luan et al., 2016).

In multi-party conversations, identifying the relationship among users is also an important task. It can be categorized into two topics, 1) predicting who will be the next speaker (Meng et al., 2017) and 2) who is the addressee (Ouchi and Tsuboi, 2016; Zhang et al., 2017). For the first topic, Meng et al. (2017) investigated a temporal-based and a content-based method to jointly model the users and context. For the second topic, which is closely related to ours, Ouchi and Tsuboi (2016) proposed to predict the addressee and utterance given a context with all available information. Later, Zhang et al. (2017) proposed a speaker-interactive model, which takes users' role information into consideration and implements a role-sensitively state tracking process.

In our task, the addressee identification problem is quite different from (Ouchi and Tsuboi, 2016) and (Zhang et al., 2017). Both of their studies aimed to make predictions on whom the last speaker addresses to. While in this paper, we focus on the whole session and aim to identify all the missing addressees. By contrast, our task is a more challenging scenario since it relies on the correlation within all users and utterances to identify the speaker-addressee structure of the entire session.

## 3 Overview

### 3.1 Problem Formulation

Given an entire multi-party conversation $S$ with length $T$, the sequence of utterances in it is defined as $\{u_t\}_{t=1}^{T}$. Each utterance is associated with a speaker $a_t^{SPR}$ and an addressee $a_t^{ADR}$. $a_t^{SPR}$ is observable across the entire session while $a_t^{ADR}$ is mostly unspecified as shown in Table 1. Our task is to identify the addressees for all utterances within the conversation session. The predicted addressee is denoted as $\hat{a}_t^{ADR}$. Formally, we have following formulations:

$$\begin{aligned} QUERY: \ & \{(a_t^{\text{SPR}}, u_t)\}_{t=1}^{T} \\ PREDICTIONS: \ & \{\hat{a}_t^{\text{ADR}}\}_{t=1}^{T} \end{aligned} \tag{1}$$

Let $A(S)$ denote the user set in the session $S$, thus $A(S)\backslash\{a_t^{SPR}\}$ denotes the listeners at the $t$-th turn ($a_t^{LSR_j}$ denotes the $j$-th listener). The listeners are also referred as candidate addressees for each turn and the identified addressee $\hat{a}_t^{ADR}$ should be one

[1] To make the model practical in learning, we assume that one utterance is associated with only one addressee.

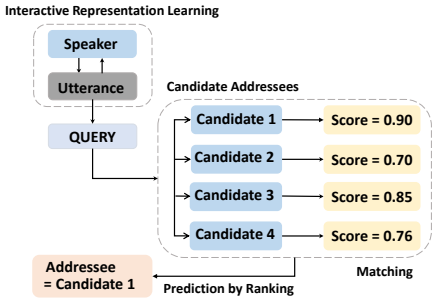


Figure 1: The system architecture of W2W model.

of them.[2]

### 3.2 Architecture Overview of W2W Model

Figure 1 illustrates the overview picture of the proposed W2W model which consists of a representation learning module and a matching module.

Concretely, the representation learning module is designed to jointly learn the representation of users and utterances in an interactive way after initializing them separately. The representations of users (also denoted as user states) and utterance embeddings are mutually enhanced. With the representations of users and utterances, a network is utilized to fuse them up into a query representation. In this way, we jointly capture who is speaking what at each step.

After the representations of users and utterances are learned, we feed them into a matching module. In this module, a matching network is learned to score the matching degrees between the query and each candidate. According to the matching scores, the model ranks all addressee candidates in $A(S)\backslash\{a_t^{SPR}\}$ and selects the one with the highest matching score as the identified addressee. For each utterance in the multi-party conversation, we repeat the above steps until the addressees of all utterances are identified.

## 4 Our W2W Model

In this section, we first describe each part of the W2W model in details: (1) Initialization of utterance and user representations; (2) Interactive representation learning of users and utterances; (3) Matching procedure for identifying the addressee. We finally describe the training procedure of the W2W model.

[2]In this paper, we denote vectors with bold lower-case (like $\mathbf{u}_i$) and matrices with bold upper-case like ($\boldsymbol{U}$ and $\boldsymbol{W}$).

### 4.1 Initialization

W2W models utterance and user embeddings separately before interactive representation learning and gets the representation of each utterance and user as initialization.

#### 4.1.1 Utterance Initialization Encoder

Suppose that in a conversation session $S$ with $T$ utterances denoted as $\{u_1, u_2, \ldots, u_T\}$. An utterance $u_t$ that contains $n$ tokens is denoted as $\{w_1, w_2, \ldots, w_n\}$, where $\{\boldsymbol{w}_i\}$ is word embeddings[3] of the $i$-th token. We first utilize a word level bi-directional RNN with Gated Recurrent Units (GRUs) (Cho et al., 2014) to encode each utterance and take the concatenation of the hidden states of the last step from both sides as the sentence embedding. Then, a sentence level bi-directional GRU is applied with each sentence embedding as input to obtain the global context of the session. The utterance representation $\boldsymbol{u}_t$ is represented by the concatenation of hidden states from both sides at $t$-th time step. [4]

#### 4.1.2 Position-Based User Initialization

In multi-party conversation, position information of different participants in the session is crucial in the addressee identification task. For example, a speaker is more likely to address his direct preceding or subsequent speaker. On this basis, we define the initialization user matrix $\mathbf{A}_{(0)}$ based on the speaking order of users in the session (Ouchi and Tsuboi, 2016). Concretely, all users in a session are sorted in a descending order according to the first time when they speak, and the $i$-th user is assigned with the $i$-th row of $\mathbf{A}_{(0)}$ as $\mathbf{a}^{\mathrm{i}}_{(0)}$. The user matrix $\mathbf{A}_{(0)}$ is trained as parameters along with other weight matrices in the neural network.

Users of the same order in different sessions share the same initialization user embedding. Note that the user representations are independent of each personality (unique user). Such strategy guarantees the initialization user embeddings to carry position information as well as handle new users unseen in training data during addressee identification.

[3] We use GloVe(Pennington et al., 2014), but it can be any word embeddings(Mikolov et al., 2013; Hu et al., 2016).

[4]Such a hierarchical framework (Serban et al., 2016) takes into account the context of all previous and future sentences in the whole session, thus enables the model to learn a strong representation.

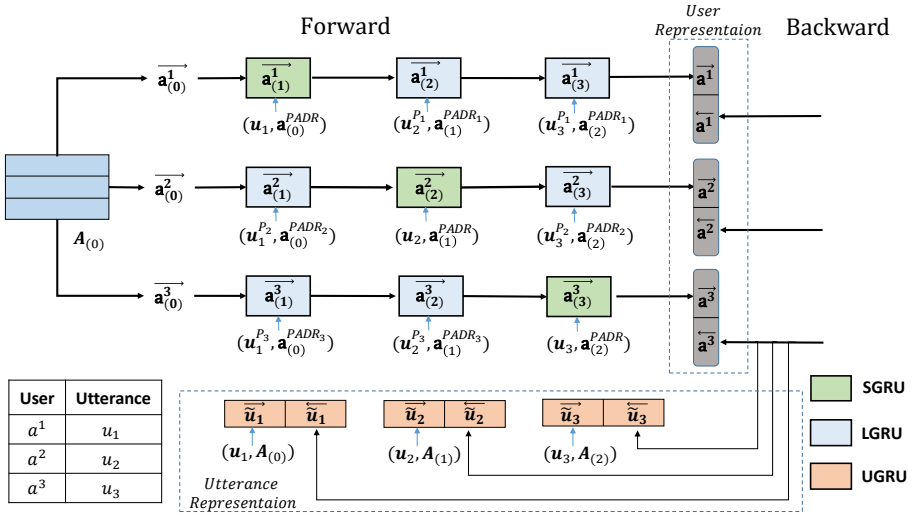


Figure 2: Interactive representation learning in W2W: At each utterance turn, SGRU tracks the speaker's state, LGRU tracks listeners' states and UGRU fuses users' states into utterances. W2W scans the conversation session from two directions and the representation of each user and utterance is the concatenation of both sides.

### 4.2 Interactive Representation Learning

To better capture who speaks what at each turn through the whole session, we propose to interactively learn the representation of utterances and users. Different from prior studies (Ouchi and Tsuboi, 2016; Zhang et al., 2017) which only track the users' states but neglecting the users' impact on utterances. We propose the W2W model which learns user and utterance representations interactively by tracking users' states with utterance embeddings as well as fusing users' states into the utterance embeddings.

#### 4.2.1 Users Representation Learning

**Role-sensitive User State Tracking.** Suggested by (Zhang et al., 2017), an utterance could have different degrees of impact on the states of the corresponding speaker and listeners. In order to capture the users' role information, we utilize two kinds of GRU-based cells represented as Speaker-GRU (SGRU) and Listener-GRU (LGRU) to track the states of the speaker and listeners respectively at each turn of the session.[5] At the $t$-th transition step, the SGRU tracks the speaker representation $\mathbf{a}^{\mathrm{SPR}}_{(\mathrm{t})}$, from the former state of him $\mathbf{a}^{\mathrm{SPR}}_{(\mathrm{t}-1)}$, the utterance representation $\mathbf{u}_t$, as well as a pseudo addressee representation $\mathbf{a}^{\mathrm{PADR}}_{(\mathrm{t}-1)}$ calculated via PAM (Person Attention Mechanism) which is a weighted sum of all the listeners' representations. Details on PAM will be elaborated in the next part. The state tracking procedure for the $i$-th step is formulated as Eq (2). The main idea of SGRU is to incorporate two reset gates, each of which controls the information fusion from the listeners and speaker respectively, denoted as $r_i$ and $p_i$. $\boldsymbol{W}$, $\boldsymbol{U}$ and $\boldsymbol{V}$ are learnable parameters.

$$
\begin{aligned}
\mathbf{r}_i &= \sigma(\boldsymbol{W}_r\boldsymbol{u}_i + \boldsymbol{U}_r\mathbf{a}^{\mathrm{SPR}}_{(\mathrm{i}-1)} + \mathrm{V}_{\mathrm{r}}\mathbf{a}^{\mathrm{PADR}}_{(\mathrm{i}-1)}) \\
\mathbf{p}_i &= \sigma(\boldsymbol{W}_p\boldsymbol{u}_i + \boldsymbol{U}_p\mathbf{a}^{\mathrm{SPR}}_{(\mathrm{i}-1)} + \mathrm{V}_{\mathrm{p}}\mathbf{a}^{\mathrm{PADR}}_{(\mathrm{i}-1)}) \\
\mathbf{z}_i &= \sigma(\boldsymbol{W}_z\boldsymbol{u}_i + \boldsymbol{U}_z\mathbf{a}^{\mathrm{SPR}}_{(\mathrm{i}-1)} + \mathrm{V}_{\mathrm{z}}\mathbf{a}^{\mathrm{PADR}}_{(\mathrm{i}-1)}) \\
\widetilde{\mathbf{a}}^{SPR}_{(i)} &= \tanh(\boldsymbol{W}\boldsymbol{u}_i + \boldsymbol{U}(\mathbf{r}_i \odot \mathbf{a}^{\mathrm{SPR}}_{(\mathrm{i}-1)}) + \mathrm{V}(\mathbf{p}_{\mathrm{i}} \odot \mathbf{a}^{\mathrm{PADR}}_{(\mathrm{i}-1)})) \\
\mathbf{a}^{\mathrm{SPR}}_{(\mathrm{i})} &= \mathbf{z}_i \odot \mathbf{a}^{\mathrm{SPR}}_{(\mathrm{i}-1)} + (1-\mathbf{z}_{\mathrm{i}}) \odot \widetilde{\mathbf{a}}^{\mathrm{SPR}}_{(\mathrm{i})}
\end{aligned}
\tag{2}
$$

Symmetrically, LGRU incorporates the embeddings of a certain listener as well as a pseudo speaker and a pseudo utterance representation (also calculated via PAM) as inputs and tracks the state of each listener. SGRU and LGRU have symmetric updating functions as Eq (2) except for the difference on pseudo representation incorporated in the cell.[6] The parameters of SGRU and LGRU are not shared, which guarantees W2W to learn role-dependent features in users' state tracking procedure. The whole structure of SGRU and LGRU are illustrated in Figure 3.

**Person Attention Mechanism.** We propose a person attention mechanism (PAM) (Eq (3) to model

[5] We denote a user embedding tracked until $t_{th}$ time step as $\mathbf{a}_{(\mathrm{t})}$, with $\mathbf{a}^{\mathrm{SPR}}_{(\mathrm{t})}$ as the representation of the speaker at $t_{th}$ turn and $\mathbf{a}^{\mathrm{LSR_j}}_{(\mathrm{t})}$ as the representation of the $j_{th}$ listener at $t_{th}$ turn.

[6] SGRU incorporates pseudo addressee $\mathbf{a}^{\mathrm{PADR}}_{(\mathrm{i}-1)}$ for speaker state tracking. LGRU incorporates the pseudo speaker $\mathbf{a}^{\mathrm{PSPR_j}}_{(\mathrm{i}-1)}$ and pseudo utterance $\mathbf{u}^j_i$ to track each $j_{th}$ listener

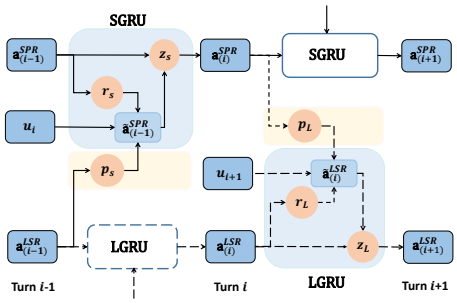


Figure 3: Structures of SGRU and LGRU.

the state tracking process exactly. Each element $\beta_i^j$ measures how likely the model estimates the $j$-th listener to be the addressee for the $i$-th turn based on the user representations tracked until turn $i$. $\boldsymbol{W}_p$ is the parameter.

$$\beta_i^j = \sigma(\mathbf{a}^{\mathrm{LSR_j}}_{(i-1)} \mathbf{W}_{\mathrm{p}} [\mathbf{a}^{\mathrm{SPR}}_{(i-1)}; \mathbf{u}_i]^{\mathsf{T}}) \tag{3}$$

Then for each turn $i$, a pseudo addressee $\mathbf{a}^{\mathrm{PADR}}_{(i-1)}$ is generated as the weighted sum of all listener representations tracked until step $i$ as Eq (4). Intuitively, a listener with a higher matching score is more likely to be the addressee at the current step. The pseudo addressee $\mathbf{a}^{\mathrm{PADR}}_{(i-1)}$ is incorporated into the state tracking of the speaker as Eq (2).

$$\mathbf{a}^{\mathrm{PADR}}_{(i-1)} = \frac{\sum_j \beta_i^j \cdot \mathbf{a}^{\mathrm{LSR_j}}_{(i-1)}}{\sum_j \beta_i^j} \tag{4}$$

$$\mathbf{a}^{\mathrm{PSPR_j}}_{(i-1)} = \beta_i^j \cdot \mathbf{a}^{\mathrm{SPR_j}}_{(i-1)} \tag{5}$$

$$\mathbf{u}_i^{P_j} = \beta_i^j \cdot \mathbf{u}_i \tag{6}$$

Symmetrically, the pseudo speaker $\mathbf{a}^{\mathrm{PSPR_j}}_{(i-1)}$ and pseudo utterance $\mathbf{u}_i^{P_j}$ are generated through Eq (5) and Eq (6) for each listener $j$ at the $i$-th turn of the conversation.

#### 4.2.2 Utterances Representation Learning

We design a UGRU (Utterance-GRU) cell[7], which has the same structure as SGRU/LGRU to fuse the utterance embedding, current speaker embedding and the user-summary vector into an enhanced utterance embedding. The user matrix initialized with $\boldsymbol{A}_{(0)}$ (as described in 4.2.1) and tracked until step $t-1$ is denoted as $\boldsymbol{A}_{(t-1)}$. The user-summary vector is calculated through max-pooling over users on $\boldsymbol{A}_{(t-1)}$ as a summary of all users' current states.

$$\begin{aligned} \boldsymbol{h}_s &= \text{Max-Pool}(\boldsymbol{A}_{(t-1)}) \\ \tilde{\boldsymbol{u}}_t &= \text{UGRU}(\boldsymbol{u}_t, \mathbf{a}^{\mathrm{SPR}}_{(t-1)}, \mathrm{h_s}) \end{aligned} \tag{7}$$

[7]Note that although UGRU has the same structure as SGRU/LGRU, it acts on each utterance for only once instead of recurrently tracking.

**Input:** Initial representations of utterances $\{\boldsymbol{u}_t\}_{t=1}^{T}$
Initial user matrix $\boldsymbol{A}_{(0)}$
**for** $i = 0; i < T; i++$ **do**
  Calculate current matching scores through **PAM** Eq (3);
  Generate pseudo embeddings using Eq (4,5,6);
  Track the speaker state through SGRU using Eq (2);
  Track each listener's state with LGRU using Eq (2);
  Fuse users' information into utterance representation using Eq (7) ;
**end**
**return** *1) User matrix of the last turn* $\overrightarrow{\boldsymbol{A}}_{(T)}$;
*2) Enhanced utterance embeddings* $\{\overrightarrow{\tilde{\boldsymbol{u}}}_t\}_{t=1}^{T}$.

**Algorithm 1:** Interactive Representation Learning Algorithm (Forward Pass).

#### 4.2.3 Forward-and-Backward Scanning.

Considering that the addressee of an utterance can be the speaker of the preceding utterances or the subsequent ones, it is important to capture the dependency from both sides for users and utterances. We propose a forward-and-backward scanning schema to enhance the interactive representation learning. For forward pass, W2W model outputs the forward user matrix of the last time step, denoted as $\overrightarrow{\boldsymbol{A}}_{(T)}$ as well as all the forward-enhanced utterance embeddings $\{\overrightarrow{\tilde{\boldsymbol{u}}}_t\}_{t=1}^{T}$ as illustrated in Algorithm 1. The backward pass initializes users and utterances in the same way as the forward pass and scans the conversation session in the reversed order. Representations from both sides are concatenated correspondingly as the final representation as Eq (8).

$$\tilde{\mathbf{u}}_i = [\overrightarrow{\tilde{\boldsymbol{u}}_i}; \overleftarrow{\tilde{\boldsymbol{u}}_i}]; \quad \mathbf{a}^{\mathrm{j}} = [\overrightarrow{\mathbf{a}^{\mathrm{j}}_{(\mathrm{T})}}; \overleftarrow{\mathbf{a}^{\mathrm{j}}_{(\mathrm{T})}}] \tag{8}$$

### 4.3 Matching

**Matching Network.** We first fuse the speaker embedding and the utterance embedding into a query representation as $\mathbf{q}$, then measures the embedding similarity $s$ between the query and each listener:

$$\begin{aligned} \mathbf{q} &= \tanh(\mathbf{W}_s \mathbf{a}^{\mathrm{SPR}} + \mathbf{W}_{\mathrm{u}} \tilde{\mathbf{u}}) \\ s &= \sigma(\mathbf{a}^{\mathrm{LSR}} \mathbf{W}_{\mathrm{m}} \mathbf{q}^{\mathsf{T}}) \end{aligned} \tag{9}$$

Table 2: Data statistics: sample size of datasets with different session lengths (i.e., 5, 10, 15) from the Ubuntu dataset (Ouchi and Tsuboi, 2016).

| Dataset | Train | Dev | Test |
|---|---|---|---|
| Len-5 | 461,120 | 28,570 | 32,668 |
| Len-10 | 495,226 | 30,974 | 35,638 |
| Len-15 | 489,812 | 30,815 | 35,385 |

where $\boldsymbol{W}_s, \boldsymbol{W}_u, \boldsymbol{W}_m$ denote weight matrices. For simplicity, we use a short-handed $Match(.)$ to denote the Equation (9) when there is no ambiguity.

**Addressee Identification.** For each turn in the session, we score the matching degree between the query and each listener and select the best matched $\hat{a}_i^{ADR}$ as the addressee prediction as Eq (10). $s_i^j$ denotes the matching score between the $j$-th listener $\mathbf{a}^{\mathrm{LSR_j}}$ and the query of the $i$-th turn. For the entire conversation, we repeat the above steps until the addressee for each utterance is identified.

$$\begin{aligned} s_i^j &= \mathrm{Match}(\mathbf{a}_i^{\mathrm{SPR}}, \mathbf{a}_i^{\mathrm{LSR_j}}, \widetilde{\mathbf{u}}_i) \\ \hat{a}_i^{ADR} &= \mathrm{argmax}_j(\{s_i^j\}) \end{aligned} \tag{10}$$

### 4.4 Learning

We utilize the cross-entropy loss to train our model (Ouchi and Tsuboi, 2016). The objective is to minimize the loss as follows:

$$loss = -\sum_k \sum_i \left(\log(s_i^+) + \log\left(1 - s_i^-\right)\right) + \frac{\lambda}{2}\|\theta\|_2 \tag{11}$$

Each subscript $k$ denotes a session, and subscript $i$ is taken from the utterances that have ground truths of addressee information. $s_i^+$ denotes the matching score between the query and the ground truth addressee, $s_i^-$ denotes the score of the negative matching, where the candidate addressee is negatively sampled. All parameters in our W2W model are jointly trained via Back-Propagation (BP) (Rumelhart et al., 1986).

## 5 Experimental Setups

**Dataset.** We run experiments using the benchmark Ubuntu dataset released by Ouchi and Tsuboi (2016). The corpus consists of a huge amount of records including response utterances, user IDs and their posting time. We organize the dataset as samples of conversation sessions.

We filter out the sessions without a single addressee ground truth, which means no label is available in these sessions. We also filter out session samples with one or more blank utterance. Moreover, we separate the conversation sessions into three categories according to the session length. Len-5 indicates the sessions with 5 turns and it is similar for Len-10 and Len-15. Such a splitting strategy is adopted in related studies as (Ouchi and Tsuboi, 2016; Zhang et al., 2017). The dataset is split into train-dev-test sets and the statistics are shown in Table 2.

**Comparison Methods.** We utilize several algorithms including heuristic and state-of-the-art methods as baselines. As there is no existing method that can perform the new task, we have to adapt baselines below into our scenario.

• Preceding: The addressee is designated as the preceding speaker of the current speaker.

• Subsequent: The addressee is designated as the next speaker.

• Dynamic RNN (DRNN): The model is originally designed to predict the addressee of the last utterance given the whole context available (Ouchi and Tsuboi, 2016). We adapt it to our scenario which is to identify addressees for all utterances in the conversation session. Concretely, the representations of users and context are learned in the same way as DRNN. While during the matching procedure, the representations of speaker and context are utilized to calculate the matching degree with candidate addressees at each turn.

• Speaker Interactive RNN (SIRNN): SIRNN is an extension model on DRNN, which is more interaction-sensitive (Zhang et al., 2017). Since all addressee information is totally unknown in our scenario, we also adapt the model with only speaker-role and observer-role into this situation. User states are tracked recurrently according to their roles at each turn, i.e. two distinct networks ($IGRU^S$ and $IGRU^O$) are utilized to track the status of the speaker and observers at each turn. Since there is no addressee-role observable through the session, we also make some adaption on the updating cell here. At each turn, $IGRU^S$ updates the speaker embedding from the previous speaker embedding and the utterance embedding, $IGRU^O$ updates the observer embedding from the previous observer embedding and the utterance embedding. During matching procedure, we make the prediction on each turn instead of only predicting the addressee for the last turn.

**Implementation and Parameters.** For fair comparison, we choose the hyper-parameters spec-

Table 3: Addressee identification results (in %) on datasets (Len-5, Len-10 and Len-15) where $\star$ denotes p-value $< 0.01$ in the significance test against all the baselines.

| | Len-5 | | | | Len-10 | | | | Len-15 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Model | $p$@1 | $p$@2 | $p$@3 | Acc. | $p$@1 | $p$@2 | $p$@3 | Acc. | $p$@1 | $p$@2 | $p$@3 | Acc. |
| Preceding | 63.50 | 90.05 | 98.83 | 40.46 | 56.84 | 80.15 | 91.86 | 21.06 | 54.97 | 77.19 | 88.75 | 13.08 |
| Subsequent | 61.03 | 88.86 | 98.54 | 40.25 | 54.57 | 73.60 | 87.26 | 20.26 | 53.07 | 69.85 | 81.93 | 12.79 |
| DRNN | 72.75 | 93.21 | 99.24 | 58.18 | 65.58 | 85.85 | 94.92 | 34.47 | 62.60 | 82.68 | 92.14 | 22.58 |
| SIRNN | 75.98 | 94.49 | 99.39 | 62.06 | 70.88 | 89.14 | 96.10 | 40.66 | 68.13 | 85.82 | 93.52 | 28.05 |
| **W2W** | **77.55**$^\star$ | **95.11**$^\star$ | **99.57** | **63.81**$^\star$ | **73.52**$^\star$ | **90.33**$^\star$ | **96.64** | **44.14**$^\star$ | **73.42**$^\star$ | **89.44**$^\star$ | **95.51**$^\star$ | **34.23**$^\star$ |

Table 4: Consistency comparison between human inference and model predictions on overlapping rate (%). $\star$ denotes p-value $< 0.01$ in the significance test against all the baselines.

| Model | Len-5 | Len-10 | Len-15 |
|---|---|---|---|
| DRNN | 75.60 | 67.54 | 63.06 |
| SIRNN | 78.94 | 71.59 | 67.22 |
| W2W | **80.86**$^\star$ | **74.05**$^\star$ | **71.14**$^\star$ |

ified by Ouchi and Tsuboi (2016) and Zhang et al. (2017). We represent the words with 300-dimensional GloVe vectors, which are fixed during training. The dimension of speaker embeddings and hidden states are set to 50. The joint cross-entropy loss function with $L_2$ weight decay as 0.001 is minimized by Adam (Kingma and Ba, 2014) with a batch size of 128.

**Evaluation Metrics.** To examine the effectiveness of our model on the addressee identification task, we compare it with baselines in terms of precision@n (i.e. p@n) (Yan et al., 2017). For predicting an addressee of an utterance, our model actually provides a ranking list for all candidate addressees.[8] We also evaluate the performance on the session level: we mark a session as a positive sample if and only if all ground truth labels are correctly identified on the top 1 of rankings, and calculate the ratio as accuracy.

As we discussed before, only a part of utterances in multi-party conversations have explicit addressee, which limits the completeness of automatic evaluation metrics. In order to evaluate the performance on unlabeled utterances, we leverage human inference results and calculate the consistency between the model predictions and human results. Due to the labor cost limit, we randomly sample 100 sessions from the test set of Len-5, Len-10 and Len-15 respectively and recruit three volunteers to annotate the addressee label for unlabeled utterances by reasoning through content and addressee information. We leave blank on the utterance where three annotators give different inference results. Finally around 81.4% of the unlabeled utterances have two or more annotators given them same annotation. With the human inference results and model predictions, we use the overlapping rate[9] as the consistency metric.

## 6 Results and Discussion

We first give an overall comparison between W2W and baselines followed by ablation experiments to demonstrate the effectiveness of each part in W2W model. We then confirm the robustness of W2W with several factors including the numbers of users, the position of the utterance. Furthermore, we evaluate how W2W and baseline models perform on both labeled and unlabeled utterance.

**Overall Performance.** For automatic evaluation shown in Table 3, end-to-end deep learning approaches outperform heuristic ones, which indicates that simple intuition is far from satisfaction and further confirm the value of this work. Among the deep-learning based approaches, our W2W model outperforms the state-of-the-art models by all evaluation metrics. Direct adaption from approaches on identifying the last addressee of the session may not work fine for our scenario.

As shown in Figure 4, the performance of all methods drops as the context length increases (from Len-5 to Len-15) since the task becomes more difficult with more context information to encode and more candidate addressees to rank. However, the improvement of our W2W model is rather more obvious with longer context length. In particular, for the dataset Len-15, W2W improves 5% on p@1 and 10% on session accuracy over SIRNN as shown in Figure 4, which indicate the robustness of W2W model in complex scenarios.

[8] Intuitively, p@1 is the precision at the highest ranked position and should be the most natural way to indicate the performance. We also provide p@2 and p@3 to illustrate the potential of different systems to identify the correct addressee on top of the lists.

[9] The calculation formula of the overlapping rate is described in Appendix.

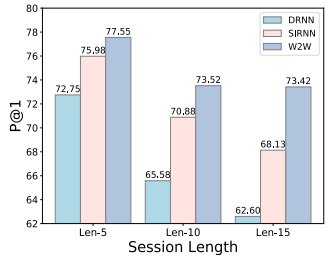


Figure 4: The comparison between W2W model and two state-of-the-art baselines on p@1.

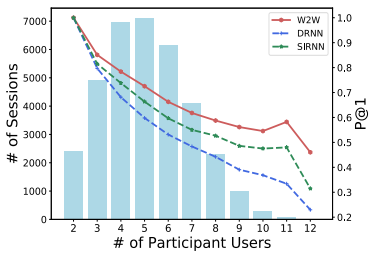


Figure 5: p@1 performance vs. number (#) of participants within a session. Results are tested on Len-15.

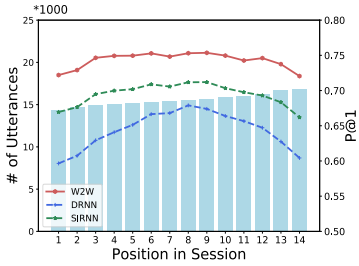


Figure 6: p@1 performance vs. positions of addressees within a session. Results are tested on Len-15.

Table 4 shows the consistency between human inference and model predictions. W2W also outperforms the baselines with a larger margin on longer conversation scenarios, which is consistent with the phenomenon of automatic evaluation. The advantage on unlabeled data of our W2W model demonstrates the superiority for detecting the latent speaker-addressee structure unspecified in the conversation stream, and that it could help find out the relationship between and across users in the session.

**Ablation Test.** Table 5 shows the results of ablation test. First, we replace the bi-directional scanning schema with the forward scanning one. The result shows that the bi-directional scanning schema captures the information in long conversations more sufficiently. Besides, we investigate the effectiveness of PAM by replacing it with the simple mean-pooling approach as Eq (12):

$$\begin{aligned} \mathbf{a}^{\mathrm{PADR}}_{(\mathrm{i}-1)} &= \frac{1}{n} \cdot \sum_{j} \mathbf{a}^{\mathrm{LSR_j}}_{(\mathrm{i}-1)} \\ \mathbf{a}^{\mathrm{PSPR_j}}_{(\mathrm{i}-1)} &= \frac{1}{n} \cdot \mathbf{a}^{\mathrm{SPR_j}}_{(\mathrm{i}-1)} \\ \mathbf{u}_i^{P_j} &= \frac{1}{n} \cdot \mathbf{u}_i \end{aligned} \tag{12}$$

The result shows that it is more beneficial to capture the correlation between the user-utterance pair and each listener and implement the state tracking correspondingly at each turn with our PAM mechanism.

To investigate the effectiveness of interactive representation learning module, we first remove the UGRU cell and fix the utterance representations in the state tracking procedure (referred as w/o Utterance Interaction in Table 5). Symmetrically, we fix the user representations in the session by removing the SGRU and LGRU cell and maintain only the interaction affect from the users to the utterances (referred as w/o User Interaction in Table 5). We also conduct an experiment on taking off the whole interactive representation module by removing UGRU, SGRU and LGRU, where the addressee identification is totally dependent on the initial representations of users and utterances. The result demonstrates that each part of them has an important contribution to our W2W model, especially the interaction affect on users.

Table 5: Ablation test on the effectiveness of W2W model parts in dataset Len-15.

| Model | $p@1$ | $p@2$ | $p@3$ | Acc. |
|---|---|---|---|---|
| W2W w/ Forward Scanning Only | 71.60 | 87.99 | 94.80 | 31.39 |
| W2W w/o PAM | 72.56 | 88.83 | 95.21 | 32.78 |
| W2W w/o Utterance Interaction | 72.94 | 88.89 | 95.28 | 33.04 |
| W2W w/o User Interaction | 49.18 | 72.38 | 86.81 | 18.24 |
| W2W w/o Interaction | 46.39 | 71.66 | 86.14 | 15.15 |
| **W2W** | **73.42** | **89.41** | **95.50** | **33.89** |

**Number of Participants.** The task becomes more difficult as the number of participants involved in the conversation increases since more participants correspond to more complicated speaker-addressee relationship. We investigate how the performance correlates with the number of speakers in the dataset Len-15. The results in terms of p@1 are shown in Figure 5. In conversations with few participants, all methods have rather high performance. As the participant number increases, W2W constantly outperforms the baselines. The performance gap becomes larger especially when there are 6 users and more, which indicates the capability of our W2W model in handling complex conversation situations.

**Position of Addressee-to-Predict.** As mentioned above in 4.1.2, position information of utterances is a crucial factor in identifying addressees for multi-party conversation. We investigate how the system performance correlates with the position of the addressee to be predicted. In

Figure 6, we show the p@1 performance of the W2W and baselines when predicting the addressee of $u_i$ at the $i_{th}$ turn. Again, W2W shows consistently better performance than the other baselines no matter where the turn to predict addressee is.

We can observe that all the methods perform relatively poor at the beginning or the end. Compared with the middle part of a long conversation, the beginning and the ending contains less context information, which makes the addressee of these part more difficult to predict. The result in Figure 6 shows that the gap between W2W and other methods is even larger where the addressee-to-predict is at the beginning or the end, which indicates that W2W is better at capturing key information and has stronger robustness in difficult scenarios.

**Variance of Matching Scores.** In real multi-party conversations, the utterances without addressee information can be divided into two cases. Sometimes an utterance has an explicit addressee while the speaker doesn't specify whom he/she is speaking to. We denote these cases as **NP** which refers to NULL-Positive. Another case is that the utterances don't address to any user in the conversation (denoted as **NN** which means Null-Negative), such as utterances 'Hi, everyone!' and 'Can anyone here help me?' In Ubuntu IRC dataset, unlabeled of the NN case and NP case are mixed and are difficult to distinguish without manual annotation.

Meanwhile, our W2W model and baseline approaches predict matching scores on each listener for every utterance no matter it has addressee information or not. For each utterance, the variance of the matching scores on all listeners represents how certain the model is on its addressee identification decision. A larger variance corresponds to a more confident prediction. Table 6 demonstrates the variance comparison on labeled and unlabeled cases in the test set Len-15. On utterances with addressee labels, the variance of our W2W model is significantly larger than the state-of-the-art baseline, which indicates that W2W has a higher degree of certainty about its own predictions when the conversation content is referring to someone explicitly. For utterances without addressee labels, the difference of variance between W2W and SIRNN is significantly reduced. Considering that unlabeled sets consist of NN ones as well as NP ones on which W2W has much larger variance than SIRNN just as the labled utterance scenario, we can infer that W2W has much lower variance than SIRNN on the NN case. Such phenomenon reflects that our W2W model won't make a prediction recklessly on occasions where there is no clear addressee. Therefore, the variance on matching scores of all listeners in our W2W model, to some extent, provides a signal of whether the utterance has a explicit addressee even if we do not provide any supervision information on this aspect during training.

Table 6: Variance of matching scores on labeled and unlabeled utterances in the set Len-15.

| Model | Labeled | Unlabeled |
|---|---|---|
| W2W | 0.111 | 0.080 |
| SIRNN | 0.088 | 0.077 |

## 7 Conclusion

In this paper, we aim at learning to identify the utterance addressees in multi-party conversations by predicting who is speaking to whom, which is a new task.. To perform the new task, we propose the W2W model which learns the user and utterance representations interactively. To handle the uncertainty in the conversation session, we design PAM which captures matching degree between current query and each candidate addressees. The experimental results show prominent and consistent improvement over heuristic and state-of-the-art baselines.

In the future, we will further investigate better utterance models associated with additional information such as topics or knowledge. With the help of learned addressee structure, we can build a general structural dialogue system for complex multi-party conversations.

## Acknowledgments

We would like to thank the reviewers for their constructive comments. We would also like to thank Bing Liu and Zhangming Chan from Peking University for their suggestion and help on this paper. This work was supported by the National Key Research and Development Program of China (No. 2017YFC0804001), the National Science Foundation of China (NSFC No. 61876196 and NSFC No. 61672058). Rui Yan and Wenpeng Hu were sponsored by Alibaba Innovative Research (AIR) Grant.

## References


Po-Chun Chen, Ta-Chung Chi, Shang-Yu Su, and Yun-Nung Chen. 2017. Dynamic time-aware attention to speaker roles and contexts for spoken language understanding. In *Automatic Speech Recognition and Understanding Workshop (ASRU), 2017 IEEE*, pages 554–560, Okinawa Japan. IEEE.

Ta Chung Chi, Po Chun Chen, Shang-Yu Su, and Yun-Nung Chen. 2017. Speaker role contextual modeling for language understanding and dialogue policy learning. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 163–168, Taipei Taiwan. IJCNLP.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha Qatar. EMNLP.

Wenpeng Hu, Zhangming Chan, Bing Liu, Dongyan Zhao, Jinwen Ma, and Rui Yan. 2019. Gsn: A graph-structured network for multi-party dialogues. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 5010–5016. International Joint Conferences on Artificial Intelligence Organization.

Wenpeng Hu, Jiajun Zhang, and Nan Zheng. 2016. Different contexts lead to different word embeddings. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 762–771, Osaka, Japan. The COLING 2016 Organizing Committee.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 9:15.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119, Berlin Germany. ACL.

Jiwei Li, Michel Galley, Chris Brockett, Georgios Spithourakis, Jianfeng Gao, and Bill Dolan. 2016b. A persona-based neural conversation model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 994–1003, Berlin Germany. ACL.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016c. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202, Berlin Germany. ACL.

Jiwei Li, Will Monroe, Tianlin Shi, Sèbastien Jean, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2157–2169, Copenhagen Denmark. EMNLP.

Yi Luan, Yangfeng Ji, and Mari Ostendorf. 2016. Lstm based conversation models. *arXiv preprint arXiv:1603.09457*, 1.

Zhao Meng, Lili Mou, and Zhi Jin. 2017. Towards neural speaker modeling in multi-party conversation: The task, dataset, and models. *arXiv preprint arXiv:1708.03152*, 1.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Hiroki Ouchi and Yuta Tsuboi. 2016. Addressee and response selection for multi-party conversation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2133–2143, Austin Texas USA. EMNLP.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing*, pages 1532–1543, Doha Qutar. EMNLP.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *nature*, 323(6088):533.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 3776–3783, Phoenix Arizona USA. AAAI Press, AAAI.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 1, pages 1577–1586, Beijing China. ACL.

Chongyang Tao, Wei Wu, Can Xu, Wenpeng Hu, Dongyan Zhao, and Rui Yan. 2019a. Multi-representation fusion network for multi-turn response selection in retrieval-based chatbots. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, WSDM '19, pages 267–275, New York, NY, USA. ACM.

Chongyang Tao, Wei Wu, Can Xu, Wenpeng Hu, Dongyan Zhao, and Rui Yan. 2019b. One time of interaction may not be enough: Go deep with an interaction-over-interaction network for response selection in dialogues. In *Proceedings of the 57th*

*Conference of the Association for Computational Linguistics*, pages 1–11.

David C Uthus and David W Aha. 2013. The ubuntu chat corpus for multiparticipant chat analysis. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence*, Bellevue Washington USA. AAAI.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*, 1.

Rui Yan, Yiping Song, and Hua Wu. 2016. Learning to respond with deep neural networks for retrieval-based human-computer conversation system. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 55–64, Pisa Tuscany Italy. ACM, SIGIR.

Rui Yan, Dongyan Zhao, et al. 2017. Joint learning of response ranking and next utterance suggestion in human-computer conversation system. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 685–694, Tokyo Japan. ACM, SIGIR.

Rui Zhang, Honglak Lee, Lazaros Polymenakos, and Dragomir Radev. 2017. Addressee and response selection in multi-party conversations with speaker interaction rnns. *arXiv preprint arXiv:1709.04005*, 1.

Xiangyang Zhou, Daxiang Dong, Hua Wu, Shiqi Zhao, Dianhai Yu, Hao Tian, Xuan Liu, and Rui Yan. 2016. Multi-view response selection for human-computer conversation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 372–381, Austin Texas USA. EMNLP.

# A Semi-Supervised Stable Variational Network for Promoting Replier-Consistency in Dialogue Generation

**Jinxin Chang**[1,2,*], **Ruifang He**[1,2,3,*†], **Longbiao Wang**[1,2,†], **Xiangyu Zhao**[1,2], **Ting Yang**[1,2], **and Ruifang Wang**[1,2]

[1]Tianjin Key Laboratory of Cognitive Computing and Application, Tianjin, China.
[2]College of Intelligence and Computing, Tianjin University, Tianjin, China.
[3]State Key Laboratory of Cognitive Intelligence, iFLYTEK, P.R. China.
{changjinxin,rfhe,longbiao_wang}@tju.edu.cn
{zhaoxiangyu009,16622898776,wrf276224255}@163.com

## Abstract

Neural sequence-to-sequence models for dialog systems suffer from the problem of favoring uninformative and non replier-specific responses due to lacking of the global and relevant information guidance. The existing methods model the generation process by leveraging the neural variational network with simple Gaussian. However, the sampled information from latent space usually becomes useless due to the KL divergence vanishing issue, and the highly abstractive global variables easily dilute the personal features of replier, leading to a non replier-specific response. Therefore, a novel **S**emi-Supervised **S**table **V**ariational **N**etwork (SSVN) is proposed to address these issues. We use a unit hypersperical distribution, namely the von Mises-Fisher (vMF), as the latent space of a semi-supervised model, which can obtain the stable KL performance by setting a fixed variance and hence enhance the global information representation. Meanwhile, an unsupervised extractor is introduced to automatically distill the replier-tailored feature which is then injected into a supervised generator to encourage the replier-consistency. Experimental results on two large conversation datasets show that our model outperforms the competitive baseline models significantly, and can generate diverse and replier-specific responses.

## 1 Introduction

Dialog systems, aiming at generating relevant and fluent responses in the replier-consistent way, have received increasing attention due to its numerous applications (Grosz, 2016; Chen et al., 2017a). Recently, Seq2Seq neural networks (Sutskever et al., 2014) have demonstrated excellent results on open-domain conversation (Shang et al., 2015; Sordoni et al., 2015; Vinyals and V. Le, 2015; Yao et al., 2015). However, due to lacking of the global and relevant information guidance, they inherently tend to generate trivial and uninformative responses (e.g., "I don't know"), rather than meaningful and replier-specific ones (Serban et al., 2016; Li et al., 2016).

The existing methods based on neural variational methods with Gaussian (Kingma and Welling, 2014; Kingma et al., 2014), are proposed to use a latent variable as the global information in decoder to strengthen the generation (Serban et al., 2017; Zhao et al., 2017; Chen et al., 2018). However, they face the problems of (1) **latent space futility** and (2) **replier-consistency decay**.

(1) The model tends to select more gain from a lower Kullback-Leibler (KL) divergence during training, which encourages the approximate posterior close to Gaussian prior, rendering the latent space of the former unused. Thus, the latent variables on this space become worthless global guidance for decoder. To address this issue, most previous work (Xie et al., 2017; Yang et al., 2017; Chen et al., 2017) has suggested a weaker decoder to match the Gaussian samples, which essentially sacrifice the generative capacity. (2) The speakers in a dyadic conversation have different linguistic characteristics, sentiments and personalities. However, the latent variable is learned conditioned on the holistic context without any distinction between speakers, especially the replier. This will dilute the personal features of replier and lead to a decrease in replier-consistency. Current methods (Li et al., 2016; Zhang et al., 2018) normally recur to artificially scheduled personal information to promote the replier-consistency, but they cannot be migrated to the other datasets.

Inspired by the effectiveness of vMF distribution in solving the KL-vanishing in the unsupervised scene (Xu and Durrett, 2018) and the suc-

* Equal contribution.
† Corresponding author.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1920–1930, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

cess of Variational Auto-Encoder (VAE) in capturing latent feature of the real data (Davidson et al., 2018), we propose a **S**emi-Supervised **S**table **V**ariational **N**etwork (SSVN) framework to address the above issues. It consists of an unsupervised personal feature extractor (a VAE with vMF) and a supervised information-enhanced generator (a CVAE with vMF). To maintain the consistency of replier features, the extractor only encodes the previous utterances from the replier and produces a personally tailored latent variable. On the top of this, the generator fuses the replier-tailored latent variable and the self vMF distributed global information to facilitate the diverse and replier-specific responses.

In general, our contributions are as follows:

- A semi-supervised stable variational network is proposed to solve the latent space futility issue and promote the replier-consistency.
- To the best of our knowledge, our model is the first to use the vMF distribution in a semi-supervised framework for dialogue generation, which can enhance the global information by alleviating the KL divergence vanishing problem.
- An unsupervised personal feature extractor is designed to acquire the replier-specific features automatically.
- The experimental results on two large conversation datasets validate the effectiveness of our model.
- It is shown that the different roles of vMF on extractor and generator. We suprisingly find that the extractor can alleviate the KL-vanishing to some extent.

## 2 Related Work

### 2.1 Neural Variational Network

Variational autoencoder (VAE) (Kingma and Welling, 2014; Rezende et al., 2014) is one of the most popular generative models. The principle idea is to encode the data $x$ to learn a probability distribution $z$, then sample the latent variables from $z$ and inject them into a directed decoder network to reconstruct $x$. The model parameters are optimized by maximizing a reparameterized variational lower bound. Based on this process, the conditional VAE (CVAE) (Sohn et al., 2015) can be conditioned on certain attributes to improve diversity. In diaog generation task, Serban et al. (2017) employs the CVAE to acquire a global latent variable as a holistic representation in a hierarchical setting. Zhao et al. (2017) regards the latent variable as a global dialog act information and directly feed it to the decoder to control the dialog act of a response. To maintain the long-term memory of the previous utterances, Chen et al. (2018) utilizes the higher-level abstract variable to retrieve and update memory cells.

### 2.2 Latent Space Futility

As for the latent space futility issue, also called KL-vanishing in Shen et al. (2018), most previous work has suggested a weaker decoder to encourage the simple Gaussian samples to be leveraged, such as a word drop-out technique in decoder (Xie et al., 2017; Zhao et al., 2017) or a practice of replacing RNN decoder with a CNN counterpart (Yang et al., 2017; Chen et al., 2017; Semeniuta et al., 2017). These methods are contrary to our origiral intention due to generation capacity descending. Other efforts focus on changing prior and posterior: Rezende and Mohamed (2015) and Kingma et al. (2016) utilize a normalizing flow to transform the sampled variables; Shen et al. (2018) introduces an AE module to complicate the quondam distribution. Extending the latter direction yet without increasing the model complexity, we only leverage the vMF distribution instead of the simple Gaussian to strengthen the KL term. Unlike the single vMF-based VAEs implemented in other cases (Davidson et al., 2018; Guu et al.; Xu and Durrett, 2018), we apply vMF into a semi-supervised dialog model to generate diverse and replier-specific responses.

### 2.3 Replier-Consistency Decay

In order to emphasize the replier-consistency, Li et al. (2016) captures personas' characteristics of Twitter users by encapsulating background information and speaking style into the distributed embeddings (one per user), which are used to improve consistency for the same person. Zhang et al. (2018) presents a persona-provided dialogue dataset and trains dialog models conditioned on their given configurable, but persistent profile information. However, the above work relies heavily on manually scheduled persona information and has difficulties in migrating to other common conversation corpora. In contrast to this, our work

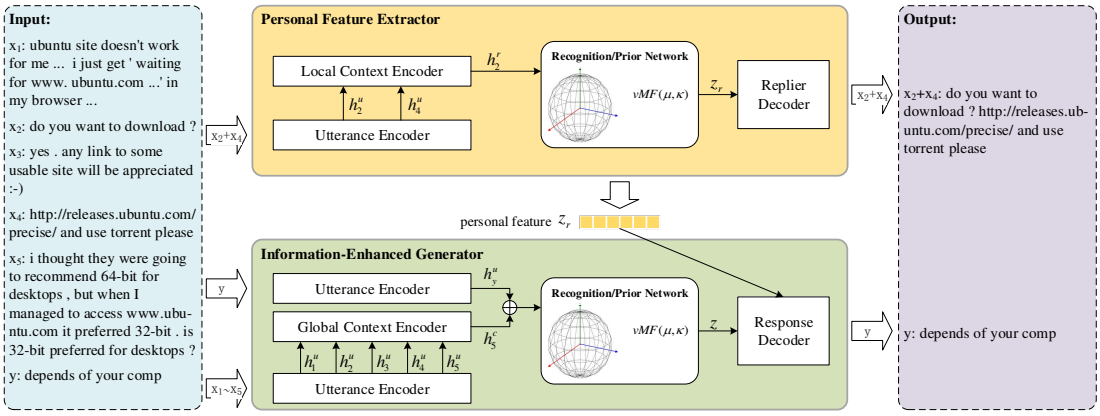


Figure 1: The SSVN framework.

focuses on automatically extracting the individual features of replier from the original conversation text, to enhance the replier-consistency of responses, without any corpus restriction.

## 3 Model

### 3.1 Task Description

Given a series of dialogue context utterances $(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n)$, where $\boldsymbol{x}_i = (w_{i,1}, w_{i,2}, ..., w_{i,N_i})$, our task is to generate a response $\boldsymbol{y} = (w_{y,1}, w_{y,2}, ..., w_{y,N_y})$ that not only rely on the global information but also consider the personally special features from the replier. In this paper, we employ the vMF distribution to stimulate the potential of latent space, impelling the extractor to condense a feature-augmented individual information and the generator to generalize a useful global guidance. The overview of SSVN is illustrated in Figure 1.

### 3.2 von Mises-Fisher

The von Mises-Fisher (vMF) places a distribution over points on the unit hypersphere, parameterized by a direction vector $\mu \in \mathbb{R}^d$ indicating the mean direction and a concentration parameter $\kappa \in \mathbb{R}_{\geq 0}$. The PDF of the vMF distribution for the unit vector $z \in \mathbb{R}^d$ is defined as:

$$f_d(x; \mu, \kappa) = \mathbb{C}_d(\kappa) \exp(\kappa \mu^T x) \quad (1)$$

$$\mathbb{C}_d(\kappa) = \frac{\kappa^{d/2-1}}{(2\pi)^{d/2} I_{d/2-1}(\kappa)} \quad (2)$$

where $\|\mu\| = 1$, $\mathbb{C}_d$ is the normalization constant, and $I_\rho$ stands for the modified Bessel function of the first kind at order $\rho$.

### 3.3 Personal Feature Extractor

To enhance the replier-consistency, the personal feature extractor, implemented by a VAE with vMF, encodes rustically the context utterances from replier $\boldsymbol{x}^r = (\boldsymbol{x}^r_1, \boldsymbol{x}^r_2, ..., \boldsymbol{x}^r_l)$ (e.g., $\boldsymbol{x}^r = (\boldsymbol{x}^r_1, \boldsymbol{x}^r_2) = (\boldsymbol{x}_2, \boldsymbol{x}_4)$ in Figure 1 into a random latent variable $z_r$, based on which the decoder reconstructs $\boldsymbol{x}^r$. Due to an intractable high-dimensional integral problem over the latent variable $z_r$, we set a recognition network $q_{\phi_e}(z_r|\boldsymbol{x}^r)$ as a variational approximation to the true posterior $p(z_r|\boldsymbol{x}^r)$, then apply variational inference to optimize the evidence lower bound (ELBO) as:

$$\begin{aligned} \mathcal{L}_E &= -KL(q_{\phi_e}(z_r|\boldsymbol{x}^r) \| p_{\theta_e}(z_r)) \\ &\quad + \mathbb{E}_{q_{\phi_e}(z_r|\boldsymbol{x}^r)} \log p_{\theta_e}(\boldsymbol{x}^r|z_r) \\ &\leq \log \int_{z_r} p_{\theta_e}(z_r) p(\boldsymbol{x}^r|z_r) dz_r = \log p(\boldsymbol{x}^r) \end{aligned} \quad (3)$$

**Utterance & Local Context Encoder** Concretely, we employ a hierarchical encoder to encode $\boldsymbol{x}^r$: the utterance encoder based on bidirectional RNN (Schuster and Paliwal, 1997) determinitically reads each utterance $\boldsymbol{x}^r_i$ and output a size-fixed real-valued $h^u_i = [\overrightarrow{h^u_i}, \overleftarrow{h^u_i}]$, which the local context encoder takes as input to obtain the final hidden state $h^r_l$ as the summary of $\boldsymbol{x}^r$.

**Prior/Posterior Distribution** Since we assume the latent space follows vMF distribution, the prior $p_{\theta_e}((z_r) \sim vMF(\cdot, \kappa^e_{prior} = 0)$ and the variational posterior $q_{\phi_e}(z_r|\boldsymbol{x}^r) \sim vMF(\mu^e_{pos}, \kappa^e_{pos})$ where $\mu^e_{pos}$ is the output of the recognition net-

work and $\kappa^e_{pos}$ is set to a constant.

$$\tilde{\mu^e_{pos}} = f^e_{pos}(h^r_l) \tag{4}$$

$$\mu^e_{pos} = \tilde{\mu^e_{pos}}/\|\tilde{\mu^e_{pos}}\| \tag{5}$$

where $f^e_{pos}(\cdot)$ is a linear transformation, and $\|\cdot\|$ stands for 2-norm to ensure the normalization.

With the uniform distribution as our prior, the KL divergence can be computed as:

$$\begin{aligned} &KL(vMF(\mu^e_{pos}, \kappa^e_{pos})\|vMF(\cdot, 0)) = \\ &\left(1 - \frac{d}{2}\right)\log 2 - \log I_{d/2-1}(\kappa^e_{pos}) - \log\Gamma\left(\frac{d}{2}\right) \\ &\quad + \kappa^e_{pos}\frac{I_{d/2}(\kappa^e_{pos})}{I_{d/2-1}(\kappa^e_{pos})} + \left(\frac{d}{2} - 1\right)\log\kappa^e_{pos} \end{aligned} \tag{6}$$

Since Eq. 6 only depends on fixed constant $\kappa^e_{pos}$, not on $\mu^e_{pos}$, this term can resolve the latent space futility problem by averting the KL-zeroing.

**Replier Decoder** During reconstruction, the decoder receives the concatenation of replier's context $h^r_l$ and personal latent variable $z_r$ as the initial hidden state, then generates tokens sequentially under the following probability distribution:

$$p_{\theta_e}(\boldsymbol{x}^r|z_r) = \prod_{i=1}^{l}\prod_{j=1}^{N_i} p(w_{i,j}|\boldsymbol{x}^r_{<i}, w_{i,<j}) \tag{7}$$

where $l$ is the number of turns of the replier's context $\boldsymbol{x}^r$; $N_i$ is the length of the $i$-th utterance ($\boldsymbol{x}^r_i$) in $\boldsymbol{x}^r$; $w_{i,j}$ is the $j$-th token in $\boldsymbol{x}^r_i$.

### 3.4 Information-Enhanced Generator

Similar to the extractor, the information-enhanced generator based on CVAE also employs a recognition network $q_{\phi_g}(z|\boldsymbol{x}, \boldsymbol{y})$ to approximate the true posterior $p(z|\boldsymbol{x}, \boldsymbol{y})$, correspondingly, its ELBO can be calculated as:

$$\begin{aligned} \log p(\boldsymbol{y}|\boldsymbol{x}) &= \log\int_z p(\boldsymbol{y}|\boldsymbol{x}, z)p(z|\boldsymbol{x})dz \\ &\geq -KL(q_{\phi_g}(z|\boldsymbol{x}, \boldsymbol{y})\|p_{\theta_g}(z|\boldsymbol{x})) \\ &\quad + \mathbb{E}_{q_{\phi_g}(z|\boldsymbol{x},\boldsymbol{y})}\log p_{\theta_g}(\boldsymbol{y}|\boldsymbol{x}, z) \end{aligned} \tag{8}$$

when considering an external personal feature $z_r$, the ELBO in generator would be rewritten as:

$$\begin{aligned} \mathcal{L}_G &= -KL(q_{\phi_g}(z|\boldsymbol{x}, \boldsymbol{y}, z_r)\|p_{\theta_g}(z|\boldsymbol{x}, z_r)) \\ &\quad + \mathbb{E}_{q_{\phi_g}(z|\boldsymbol{x},\boldsymbol{y},z_r)}\log p_{\theta_g}(\boldsymbol{y}|\boldsymbol{x}, z, z_r) \\ &\leq \log\int_z p(\boldsymbol{y}|\boldsymbol{x}, z, z_r)p(z|\boldsymbol{x}, z_r)dz \\ &= \log p(\boldsymbol{y}|\boldsymbol{x}, z_r) \end{aligned} \tag{9}$$

Notice that $z_r$ only participates in the generation process $p_{\theta_g}(\boldsymbol{y}|\boldsymbol{x}, z, z_r)$, the approximate posterior $q_{\phi_g}(z|\boldsymbol{x}, \boldsymbol{y}, z_r) \sim vMF(\mu^g_{pos}, \kappa^g_{pos})$ is conditioned on dialog context $\boldsymbol{x}$ and the corresponding response $\boldsymbol{y}$, and the prior $p_{\theta_g}(z|\boldsymbol{x}, z_r) \sim vMF(\mu^g_{prior}, \kappa^g_{prior})$ depends on $\boldsymbol{x}$ [1].

**Utterance & Golbal Context Encoder** The hierarchical encoder in this part utilizes the shared utterance encoder from extractor to encode utterances $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n, \boldsymbol{y}$ into the corresponding representations $h^u_1, h^u_2, ..., h^u_n, h^u_y$ orderly. Thereafter, the utterance vectors $h^u_1, h^u_2, ..., h^u_n$ are fed to the global context encoder to compute the representation of the whole dialog context $h^c_n$. Based on these, the approximate posterior and prior can be determined by the following operations:

$$\tilde{\mu^g_{pos}} = f^g_{pos}([h^c_n, h^u_y]) \tag{10}$$

$$\mu^g_{pos} = \tilde{\mu^g_{pos}}/\|\tilde{\mu^g_{pos}}\| \tag{11}$$

$$\tilde{\mu^g_{prior}} = f^g_{prior}(h^c_n) \tag{12}$$

$$\mu^g_{prior} = \tilde{\mu^g_{prior}}/\|\tilde{\mu^g_{prior}}\| \tag{13}$$

where $f^g_{pos}(\cdot)$ and $f^g_{prior}(\cdot)$ are both linear transformations. $\kappa^g_{pos}$ and $\kappa^g_{prior}$ in both distributions are the constants with equal values.

**Prior/Posterior Distribution** Without $vMF(\cdot, 0)$ as the prior, we require to recalculate the KL term in generator as:

$$\begin{aligned} &KL(vMF(\mu^g_{pos}, \kappa^g_{pos})\|vMF(\mu^g_{prior}, \kappa^g_{prior})) = \\ &(d/2 - 1)\log\frac{\kappa^g_{pos}}{\kappa^g_{prior}} + \log\frac{I_{d/2-1}(\kappa^g_{prior})}{I_{d/2-1}(\kappa^g_{pos})} \\ &- \kappa^g_{prior}\mu^g_{prior}{\mu^g_{pos}}^{-1}\frac{I_{d/2}(\kappa^g_{pos})}{I_{d/2-1}(\kappa^g_{pos})} \\ &+ \kappa^g_{pos}\frac{I_{d/2}(\kappa^g_{pos})}{I_{d/2-1}(\kappa^g_{pos})} \end{aligned} \tag{14}$$

**Response Decoder** We employ a RNN decoder similar to the one in extractor, extending it to condition on a personal feature $z_r$ by concatenating $z_r$ to the input of the decoder at each time step. The concrete generative process is as follows:

$$s^R_t = \sigma(s^R_{t-1}, [e_{w_{y,t-1}}, z_r]) \tag{15}$$

$$p_{vocab} = softmax(Vs^R_t + b) \tag{16}$$

[1] The $z_r$ in $q_{\phi_g}(z|\boldsymbol{x}, \boldsymbol{y}, z_r)$ and $p_{\theta_g}(z|\boldsymbol{x}, z_r)$ is introduced formally to satisfy the consistency of the formula derivation, where, in practice, both distributions are equivalent to $q_{\phi_g}(z|\boldsymbol{x}, \boldsymbol{y})$ and $p_{\theta_g}(z|\boldsymbol{x})$ respectively.

where $\sigma$ is the sigmoid function; $e_{w_{y,i}}$ is the word embedding of the $i$-th word in response $\boldsymbol{y}$; $s_t^R$ denotes the hidden state at the time step $t$; $V$ and $b$ are learnable parameters; $p_{vocab}$ stands for the probability distribution over the vocabulary. Then the objective function of the decoder is given by:

$$p_{\theta_g}(\boldsymbol{y}|\boldsymbol{x}, z, z_r) = \prod_{i=1}^{N_y} p_{vocab}(w_{y,i}) \tag{17}$$

where $p_{vocab}(w_{y,i})$ is the probability to generate the word $w_{y,i}$; $N_y$ is the length of the response $\boldsymbol{y}$.

### 3.5 Training Objective

The entire SSVN model integrates two modules in Figure 1, i.e., the unsupervised extractor and the supervised generator, which can be optimized simultaneously in one framework. Thus, the overall objective function of SSVN is to maximize:

$$\mathcal{L} = \lambda \mathcal{L}_G + (1 - \lambda)\mathcal{L}_E \tag{18}$$

where we have a hyperparameter $\lambda$ to control the balance between response generation (generator) and personality reconstruction (extractor).

### 3.6 Sampling Techique for vMF

Similar to Xu and Durrett (2018), we utilize the rejection sampling scheme of Wood (1994) to sample a value $w \in [-1, 1]$, then derive a random unit vector tangent $\upsilon$ on the hypersphere at the mean vector $\mu$. Based on these, our latent variable $z$ can be given by $z = w\mu + \upsilon\sqrt{1 - w^2}$.

## 4 Experiments

### 4.1 Datasets

The proposed model is evaluated on two datasets. The first corpus is *Cornell Movie Dialogs Corpus*[2] (Danescu-Niculescu-Mizil and Lee, 2011) that contains more than 80,000 imagined movie conversations. To normalize the length (turns) of the dialogs, we divide the original conversations into consecutive 3-10 utterances. Our second dataset is *Ubuntu Dialogue Corpus*[3] (Lowe et al., 2015). It contains about 500,000 multi-turn dialogues collected from the Ubuntu Internet Relayed Chat channel, each of which starts with a Ubuntu-related technical problem and follows by the corresponding responses about solutions.

In the above two datasets, the last utterance in a conversation is regarded as the response and the remaining ones are the input context. The detailed statistical information is shown in Table 1.

### 4.2 Baselines

We compare SSVN with the following models:

**S2SA**: the standard Seq2Seq model with the attention mechanism (Vinyals and V. Le, 2015).

**HRED**: a hierarchical encoder framework to model multi-turn dialogs (Serban et al., 2016).

**VHRED**: a hierarchical encoder-decoder with latent stochastic variable (Serban et al., 2017).

**HVMN**: an encoder-decoder network containing the hierarchical structure and the variational memory (Chen et al., 2018).

### 4.3 Experimental Details

Our model is implemented using the Tensorflow framework (Abadi et al., 2016) with the following parameter settings: We set word embeddings to size of 200 and initialize them randomly. The shared utterance encoder is a 2-layer bidirectional GRU structure with 600 hidden neurons for each layer, while the both context encoders and the both decoders are the unidirectional ones with hidden size of 600. The dimensions of the latent variable $z$ and $z_r$ are both set to 50. We use the Adam algorithm (Kingma and Ba, 2014) to update the parameters with an initial learning rate of 0.001. In the training, we employ the early-stop strategy (Caruana et al., 2000) to select the best models using the variational lower-bound on the validation set.

### 4.4 Evaluation Metrics

We use both automatic and human evaluations to analyze the model's performance.

**Automatic Evaluation Metrics** In our experiment, three embedding-based metrics (**Average**, **Greedy**, **Extreme**) [4] (Liu et al., 2016) are employed to measure the semantic relevance between generated responses and ground truths. Besides, we also adopt **Distinct-1** and **Distinct-2** (Li et al., 2016) to evaluate the diversity of responses.

**Human Evaluation** In order to assess how well the models can maintain the replier's consistency, we conduct a human evaluation. Specifically, we randomly sample 300 context from the test set and apply 5 models to generate responses for each context. For each response, three annotators are re-

[2] The dataset is available at `https://www.cs.cornell.edu/~cristian/Cornell_Movie-Dialogs_Corpus.html`.

[3] We use the same train-validation-test split as in Chen et al. (2018).

[4] We use the embeddings trained on Google News Corpus: `https://code.google.com/archive/p/word2vec/`.

| Corpus | Train | Valid | Test | Avg. Utterances | Avg. Words | Vocab | Coverage |
|---|---|---|---|---|---|---|---|
| Cornell | 91271 | 871 | 702 | 5.04 | 16.91 | 10000 | 98.26% |
| Ubuntu | 448833 | 19584 | 18920 | 4.94 | 23.67 | 20000 | 99.12% |

Table 1: Statistical information including number of dialogs in training, validation and test sets, average number of utterances (turns) per dialog, words per utterance, vocabulary size and its proportion in corpus.

| Model | Cornell Movie Dialogs Corpus | | | | | Ubuntu Dialogue Corpus | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Average | Greedy | Extreme | Distinct-1 | Distinct-2 | Average | Greedy | Extreme | Distinct-1 | Distinct-2 |
| S2SA | 0.1979 | 0.1537 | 0.1373 | 19/.0013 | 37/.0092 | 0.2156 | 0.1688 | 0.1265 | 1543/.0047 | 5342/.0260 |
| HRED | 0.4964 | 0.3824 | 0.3317 | 27/.0068 | 56/.0171 | 0.5415 | 0.4117 | 0.3193 | 1924/.0084 | 8549/.0409 |
| VHRED | 0.5148 | 0.3954 | 0.3446 | 126/.0216 | 329/.0642 | 0.5341 | 0.4027 | 0.3062 | 2928/.0151 | 13468/.0772 |
| HVMN | 0.5347 | 0.3876 | 0.3462 | 199/.0399 | 474/.1107 | 0.5584 | 0.4229 | 0.3220 | 6334/.0193 | 25136/.1005 |
| SSVN | **0.6417** | **0.4582** | **0.3732** | **591/.0535** | **2535/.2450** | **0.6089** | **0.4433** | **0.3312** | **9562/.0259** | **62539/.1955** |

Table 2: Automatic evaluation results on two dialogue datasets. The Distinct-* contains the number of distinct n-grams and its ratio over all generated responses. The embedding-based metrics results of baselines on Ubuntu are the same as Chen et al. (2018).

cruited to give a 4-graded judgement with the following criteria: **1**: the response is ungrammatical or semantically irrelevant; or inconsistent with replier's features (e.g., linguistic characteristics, sentiments and personalities); or has wrong logic; **2**: the response is semantically weak related, but it is too trivial (e.g., "I don't know"); **3**: the response is semantically relevant and informative, but has no obvious consistency about the replier's personal features; **4**: the response is not only semantically related and informative, but also consistent with the individual features of replier.

### 4.5 Evaluation Results

**Automatic Evaluation** The metric-based evaluation results are shown in Table 2. From the results, we can observe that:

(1) HRED performs better than S2SA, indicating that the hierarchical structure is benefical.

(2) VHRED outperforms HRED on all metrics on Cornell, which demonstrates that the latent variables are the useful global guidance information. Inversely, VHRED has a worse performance than HRED in terms of three embedding-based metrics on Ubuntu, which is consistent with Chen et al. (2018) due to the domain specific dataset.

(3) On the top of VHRED, HVMN introduces the memory network to enhance the long-term memory, and obtains the best performance among the baseline models.

(4) Compared with all the baselines, our SSVN model achieves the highest scores in terms of all metrics on two datasets, indicating that SSVN can best fit the ground truth semantically and generate more informative responses. Meanwhile, the sign tests show that the improvements of SSVN are statistically significant ($p$-value$<$0.01).

(5) Noticeably, the models trained on Ubuntu consistently have more distinct n-grams than the same models trained on Cornell, while the distinct ratios do not differ much. The reason is that Ubuntu dataset has more words averagely per utterance than Cornell data (as the statistical details shown in Table 1), which forces the models to produce longer responses.

**Human Evaluation** The human evaluation results on Cornell data are shown in Table 4, in which the score distribution values represent the percentages of responses belonging to each grade, and Fleiss' kappa (Fleiss and Cohen, 1973) is employed to evaluate the inter-annotator agreement. From the results, we have the following observations:

(1) The percentage of replier-specific responses (i.e., the grade '4') of SSVN model is 22.69%, which is much higher than that of baselines, indicating that the personal feature extractor can effectively capture the personal feature of replier.

(2) SSVN model generates much more informative responses (i.e., 71.56% labeled as '3+4') and much less generic responses (i.e., 20.28% labeled as '2') than all the baselines. The results are in line with the above results of metric-based evaluation.

(3) Kappa scores of the models are all higher than 0.4, demonstrating that the annotators come to a fair agreement. Meanwhile, the sign tests also show that the human evaluation improvements of SSVN to baselines are significant on Cornell dataset ($p$-value$<$0.01).

| Model | Extractor | Generator | Average | Greedy | Extreme | Distinct-1 | Distinct-2 |
|---|---|---|---|---|---|---|---|
| SVN | – | vMF | 0.6108 | 0.4334 | 0.3456 | 403/.0374 | 1376/.1367 |
| SSVN$_{Gau}$ | Gau | Gau | 0.5563 | 0.4164 | 0.3406 | 470/.0703 | 1859/.3109 |
| SSVN$_{Gau-E}$ | Gau | vMF | 0.6164 | 0.4371 | 0.3648 | 472/.0439 | 2100/.2091 |
| SSVN$_{Gau-G}$ | vMF | Gau | 0.5605 | 0.4195 | 0.3467 | 525/**.0764** | 2097/**.3403** |
| SSVN | vMF | vMF | **0.6417** | **0.4582** | **0.3732** | **591**/.0535 | **2535**/.2450 |

Table 3: Performances of model ablation on Cornell dataset. The 'Gau' and 'vMF' denote the distributions of the latent spaces of extractor or generator, and '–' means no extractor.

| Model | score distribution (%) | | | | | Kappa |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 3+4 | |
| S2SA | 29.38 | 45.11 | 20.47 | 5.04 | 25.51 | 0.41 |
| HRED | 26.31 | 39.38 | 26.2 | 8.11 | 34.31 | 0.44 |
| VHRED | 17.58 | 36.48 | 33.55 | 12.39 | 45.94 | 0.46 |
| HVMN | 14.03 | 29.79 | 39.47 | 16.71 | 56.18 | 0.50 |
| SSVN | **8.16** | **20.28** | **48.87** | **22.69** | **71.56** | 0.45 |

Table 4: Human evaluation results on Cornell dataset. The percentages are calculated by combining the judgements from three annotators together. '3+4' stands for the sum of percentages of the grade '3' and '4' (i.e., the ratio of informative responses).

### 4.6 Discussions

**Model Ablation** To investigate the effect of different parts, we conduct a set of experiments on Cornell by removing the extractor or modifying the distribution of extractor and generator. From the results listed in Table 3, we can observe that:

(1) Removing the extractor (denoted as SVN) makes the distinct ratios and numbers drop dramatically, while the embedding-based metric scores are only slightly lower than that of SSVN. This indicates the personal features learned by the extractor not only maintain the replier-consistency, but also improve the diversity of responses. In addition, SSVN$_{Gau-E}$ (replacing the vMF distribution with a Gaussian in extractor) has a better performance than SVN, but a worse one than SSVN, demonstrating the vMF-Extractor is more effective than Gau-Extractor.

(2) As for the generator, when setting the Gaussian as the latent space (denoted as SSVN$_{Gau-G}$), the embedding-based performance deteriorates dramatically whereas the distinct numbers decrease slightly, indicating that the vMF-Generator is more capable of facilitating the generated responses semantically close to the ground truth than Gau-Generator. Notably, the distinct ratios in SSVN$_{Gau-G}$ rise unexpectedly, which will be investigated in (3).

(3) To figure out this special phenomenon, we conduct an experiment on SSVN$_{Gau}$ composed by Gau-Extractor and Gau-Generator. We can see that the SSVN$_{Gau-G}$ and SSVN$_{Gau}$ obtain the best distinct ratios among the ablation models, but their distinct numbers are not the highest. The results indicate that, whatever the latent space of extractor follows, the Gau-Generator always tends to produce informative but very short responses. Meanwhile, their worst embedding-based scores show that the responses generated by Gau-Generator semantically deviate from the ground truth significantly.

**The Effect of vMF on KL** Besides the metric-based performance, we also evaluate the effectiveness of different settings in sloving the latent space futility problem. Figure 2 visualizes the evolution of the KL loss in both extractor and generator parts. We can see that:

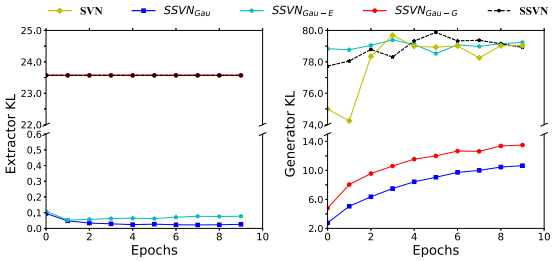


Figure 2: KL cost of different parts in ablation models as training progress.

(1) In Extractor KL, Gau-Extractors (i.e., SSVN$_{Gau}$ and SSVN$_{Gau-E}$) have a KL cost close to 0 at the begining and never recover, while the vMF-Extractors (i.e., SSVN$_{Gau-G}$ and SSVN) can keep a constant KL value as evidenced by Eq.(6). The results indicate that the vMF in extractor can mitigate KL-vanishing and capture the more meaningful personal features.

(2) Surprisingly, in Generator KL, the KL loss presents an upward trend in Gau-Generators (i.e., SSVN$_{Gau}$ and SSVN$_{Gau-G}$). The reason is that, the personal features from the extractor can effectively strengthen the expressiveness of the latent space in the generator, thus the response decoder

is encouraged to exploit the latent variables and the latent space futility problem is alleviated.

(3) Compared with the Gau-Generators in Generator KL, the vMF-Generators (i.e., SVN, $\text{SSVN}_{Gau-E}$ and SSVN) have the much higher KL values, indicating that the vMF is a better selection than Gaussian to solve the KL-vanishing problem. Meanwhile, the KL values are relatively stable, which experimentally demonstrates the KL cost mainly depends on the last term in Eq.(14) and the variable term has little effect on it. Last but not least, KL cost can explictly be changed by setting different kappa values.

**Impact of the Coefficient $\lambda$** Recall that Eq.(18) shows the capacity of SSVN in balancing response generation and personality reconstruction. Here we analyze the effects of different coefficient $\lambda$ on the quality of responses. Figure 3 shows the performances given varying $\lambda$. Notably, the performances of embedding-based metrics are changing in a similar trend, as the same case in Distinct-1 and Distinct-2, thus we only consider Average and Distinct-1 as the major analysis items.

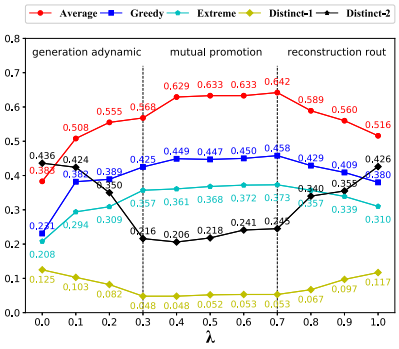


Figure 3: Influences of different $\lambda$ on Average, Greedy, Extreme, Distinct-1 and Distinct-2.

As we can see, the evolutions of Average and Distinct-1 in Figure 3 can be broadly into three stages: generation adynamic stage, mutual promotion stage and reconstruction rout stage.

(1) The first stage shows that as $\lambda$ increases, the Average monotonically increases while diversity decreases. This is because the lower $\lambda$ gives the model less incentive to optimize the generator, which makes the response decoder incapable of utilizing the higher-quality personal features, resulting in the diverse but semantically inappropriate responses.

(2) When $\lambda$ moves to the second stage, the performances of the Average and diversity improve simultaneously, implying that the response generation and personality reconstruction achieve en expected balance.

(3) For the reconstruction rout stage, although the model focuses on response generation, the larger $\lambda$ does not bring an improvement of Average, but instead increases diversity. The result indicates that the unprofitable personal features in thwarted personality reconstruction part, as a random disturbance, can increase the diversity of the responses, but severely bias the generation of response decoder semantically.

Observed from the curves of all metrics, the best performance of embedding-based metrics is achieved at $\lambda = 0.7$, while the diversity reaches the peak in the mutual promotion stage. Thus, we set $\lambda$ to 0.7 in all previous experiments.

**Case Study** Besides the quantitative analysis, we also organize some examples (seen in Table 5) from different models to analyze the performances of the methods qualitatively. They are chosen randomly from the responses produced by the proposed model, and showed together with the corresponding contexts and the outputs of the baselines. From the case 1, we can observe that the SSVN can extract the personal feature of the replier that the speaker A prefers to acquire further information from others, which guides the generator to produce an interrogative response to promote the replier-consistency. Meanwhile, the SSVN can also extract the firm attitude of the replier in case 2 and the pleading tone of the replier in case 3. By contrast, the baselines favor to produce the bad responses, such as containing more 'unk'.

**Error Analysis** To improve the performance of SSVN in the future, we take the worse cases (i.e., the grade '1' and '2') in human judgement as an example to analyze our errors. Specifically, we divide the errors with the grade '1' into grammatical error, replier-nonconsistency and logic contradiction, which occupy $19.79\%$, $31.49\%$ and $48.72\%$, respectively. We can find that 1) logic condradiction scenes make up most of the errors as SSVN pays little attention to this issue. 2) although considering the personal features from replier, there still exists $31.49\%$ replier-inconsistent cases, incidating that only strengthening the VAE with the vMF distribution may not be a perfect approach for personal feature extraction. As for grade '2', the consistency of replier's features improve the of response diversity significantly, but the model still has "safe response" problem as the baseline.

| ID | Context | Response |
|---|---|---|
| 1 | A: Where are we going?<br>B: We're getting out of this place. We've done enough.<br>ground truth: Wait a minute, whoa. We can't go now. What's ⟨unk⟩ got planned with that nerve agent? | SSVN: When we're going out?<br>S2SA: We know he is on you.<br>HRED: I'll ⟨unk⟩ there's no ⟨unk⟩, or there's no this case. ⟨unk⟩ ⟨unk⟩ well, because he is the ⟨unk⟩.<br>VHRED: It motel yet .<br>HVMN: What kind of ⟨unk⟩ were you? |
| 2 | A: There.<br>B: I beg your pardon?<br>A: There wolf! There castle!<br>B: Why are you talking like that?<br>ground truth: I thought you wanted to. | SSVN: Well, I want you to get out of here!<br>S2SA: Me ⟨unk⟩.,<br>HRED: So how do you believe that?<br>VHRED: Fourth ⟨unk⟩ she wasn't ⟨unk⟩!<br>HVMN: Old ⟨unk⟩. These ⟨unk⟩ is ⟨unk⟩ – ⟨unk⟩ ⟨unk⟩. ⟨unk⟩ years ... |
| 3 | A: James ...<br>B: I have to go.<br>A: Then take me with you.<br>B: No. You'll be safe here.<br>ground truth: I don't want to be safe! | SSVN: Please, I have to go. I can't take a walk.<br>S2SA: Tell you now ⟨unk⟩ something of ⟨unk⟩, doesn't ⟨unk⟩!<br>HRED: ⟨unk⟩ just actually, the ⟨unk⟩ ⟨unk⟩ of making the ⟨unk⟩ ⟨unk⟩. She's dead.<br>VHRED: Gross ...<br>HVMN: No. |

Table 5: Generated responses of all the models on Cornell Movie Dialogs Corpus.

The above analysis sheds light on our future directions: 1) modeling the logic consistency between the context and response; 2) exploring advanced methods for extracting personal features; 3) improving the response diversity.

## 5 Conclusion and Future Work

In this work, we propose a semi-supervised stable variational network for addressing the latent space futility and replier-consistency decay issues. Different from the traditional variational dialog models, the proposed model selects the vMF as the prior and posterior to resolve the latent space futility issue, and then integrates a unsupervised extractor to obtain the replier-tailored personal features to ensure the replier-consistency. Experimental results on two dialog datasets demonstrate the effectiveness of our model, especially on replier-consistency in terms of human evaluation. However, the error analysis shows that there are still challenges in dialogue generation, which we would like to explore in the future.

## Acknowledgments

We thank the anonymous reviewers for their valuable feedback. Our work is supported by National Natural Science Foundation of China (61976154), Tianjin Natural Science Foundation (18JCYBJC15500), National Natural Science Foundation of China (61771333), Tianjin Municipal Science and Technology Project (18ZXZNGX00330), and the Foundation of State Key Laboratory of Cognitive Intelligence, iFLYTEK(CIOS-20190001).

## References

Martn Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, and Xiaoqiang Zheng. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv:1603.04467[cs]*, arXiv:1603.04467.

Rich Caruana, Steve Lawrence, and C. Lee Giles. 2000. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances in neural information processing systems 13 (NIPS)*, pages 402–408.

Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. 2017a. A survey on dialogue systems: Recent advances and new frontiers. *ACM SIGKDD Explorations Newsletter*, 19(2):25–35.

Hongshen Chen, Zhaochun Ren, Jiliang Tang, Yihong Eric Zhao, and Dawei Yin. 2018. Hierarchical variational memory network for dialogue generation. In *Proceedings of the 2018 World Wide Web Conference (WWW '18)*, pages 1653–1662.

Xi Chen, Diederik P. Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2017. Variational lossy autoencoder. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Cristian Danescu-Niculescu-Mizil and Lillian Lee. 2011. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In *Proceedings of the 2nd Workshop on Cognitive Modeling and Computational Linguistics (CMCL '11)*, pages 76–87.

Tim R. Davidson, Luca Falorsi, Nicola De Cao, Thomas Kipf, and Jakub M. Tomczak. 2018. Hyperspherical variational auto-encoders. *arXiv:1804.00891[cs]*, arXiv:1804.00891.

Joseph L Fleiss and Jacob Cohen. 1973. The equivalence of weighted kappa and the intraclass correlation coefficient as measures of reliability. *Educational and psychological measurement*, 33(3):613–619.

Barbara J. Grosz. 2016. Ai100 report. https://ai100.stanford. edu/2016-report.

Kelvin Guu, Tatsunori B. Hashimoto, Yonatan Oren, and Percy Liang. Generating sentences by editing prototypes. *Transactions of the Association for Computational Linguistics (TACL)*, 6:437–450.

D. P. Kingma, D. J. Rezende, S. Mohamed, and M. Welling. 2014. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems 27 (NIPS)*, pages 3581–3589.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference for Learning Representations (ICLR)*.

Diederik P. Kingma, Tim Salimans, Rafal Jozefowicz, and Xi Chen. 2016. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems 29 (NIPS)*, pages 4743–4751.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 110–119.

Chia Wei Liu, Ryan Lowe, Iulian V. Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2122–2132.

Ryan Lowe, Nissan Pow, Iulian V. Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 285–294.

Danilo Jimenez Rezende and Shakir Mohamed. 2015. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 1530–1538.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31nd International Conference on Machine Learning (ICML)*, pages 1278–1286.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. 2017. A hybrid convolutional variational autoencoder for text generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 627–637.

Iulian Serban, Alessandro Sordoni, Ryan Joseph Lowe, Laurent Charlin, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*, pages 3295–3301.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*, pages 3776–3784.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1577–1586.

Xiaoyu Shen, Hui Su, Shuzi Niu, and Vera Demberg. 2018. Improving variational encoder-decoders in dialogue generation. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, pages 5456–5463.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. 2015. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems 28 (NIPS)*, pages 3483–3491.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of the 2015 Conference Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 196–205.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27 (NIPS)*, pages 3104–3112.

Oriol Vinyals and Quoc V. Le. 2015. A neural conversational model. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*.

Andrew Wood. 1994. Simulation of the von mises fisher distribution. *Communications in Statistics Simulation and Computation*, 23(1):157–164.

Ziang Xie, Sida I. Wang, Jiwei Li, Daniel Levy, Aiming Nie, Dan Jurafsky, and Andrew Y. Ng. 2017. Data noising as smoothing in neural network language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Jiacheng Xu and Greg Durrett. 2018. Spherical latent spaces for stable variational autoencoders. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4503–4513.

Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kir. 2017. Improved variational autoencoders for text modeling using dilated convolutions. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 3881–3890.

Kaisheng Yao, Geoffrey Zweig, and Baolin Peng. 2015. Attention with intention for a neural network conversation model. *ArXiv*, abs/1510.08565.

Saizheng Zhang, Emily Dinanz, Jack Urbanekz, Arthur Szlamz, Douwe Kielaz, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1–10.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 654–664.

# Modeling Personalization in Continuous Space for Response Generation via Augmented Wasserstein Autoencoders

**Zhangming Chan**[1,3,*], **Juntao Li**[1,3,*], **Xiaopeng Yang**[4], **Xiuying Chen**[1,3], **Wenpeng Hu**[2,3], **Dongyan Zhao**[1,3] and **Rui Yan**[1,3,†]

[1] Center for Data Science, AAIS, Peking University

[2] School of Mathematical Sciences, Peking University

[3] Wangxuan Institute of Computer Technology, Peking University

[4] David R. Cheriton School of Computer Science, Faculty of Mathematics, University of Waterloo

{zhangming.chan,ljt,ruiyan}@pku.edu.cn

## Abstract

Variational autoencoders (VAEs) and Wasserstein autoencoders (WAEs) have achieved noticeable progress in open-domain response generation. Through introducing latent variables in continuous space, these models are capable of capturing utterance-level semantics, e.g., topic, syntactic properties, and thus can generate informative and diversified responses. In this work, we improve the WAE for response generation. In addition to the utterance-level information, we also model user-level information in latent continue space. Specifically, we embed user-level and utterance-level information into two multimodal distributions, and combine these two multimodal distributions into a mixed distribution. This mixed distribution will be used as the prior distribution of WAE in our proposed model, named as PersonaWAE. Experimental results on a large-scale real-world dataset confirm the superiority of our model for generating informative and personalized responses, where both automatic and human evaluations outperform state-of-the-art models.

## 1 Introduction

Over the past decade, a myriad of conversational systems have been proposed in the field of artificial intelligence and achieved remarkable success in various industry scenarios, such as e-commerce assistant (Li et al., 2017) and chit-chat machine XiaoIce (Shum et al., 2018). Based on the domains involved in previous research, existing work can be categorized into two groups, i.e., vertical-domain (Glas et al., 2015) and open-domain (Zhao et al., 2017), where the former group pursues to complete a specific target with limited domain knowledge while the latter one involves massive topics in conversations. In this work, we focus on the latter one and intend to generate a natural and meaningful response for a given conversation context. Most recent works build upon the sequence to sequence model (Bahdanau et al., 2014) and can generate a fluent response. But they suffer from the notorious "universal response" issue, i.e., generating safe and uninformative responses (e.g., I don't know) (Li et al., 2015).

To address the aforementioned shortcoming, advanced conversational systems propose to capture and incorporate extra information from two different levels, i.e., utterance-level and user-level. As for utterance-level information modeling, previous works mainly construct models upon variational autoencoders (VAE) (Kingma and Welling, 2014). By doing so, responses with diverse and informative words can be generated by introducing a latent variable for modeling utterance-level information such as topic, and syntactic structure (Bowman et al., 2015). It is verified in various open-domain response generation situations that conditional variational autoencoders (CVAE) (Serban et al., 2017; Zhao et al., 2017) are effective for addressing the "universal response" issue. In user-level information modeling, existing models either implicitly learn user information from training data such as learning user embedding (Li et al., 2015) or explicitly collect user profiles as the accurate personalization (Zhang et al., 2017; Yang et al., 2018; Zhang et al., 2018). Although obtaining user profiles is more effective and accurate than user embeddings, it is time-consuming and economically costly, or even impossible under the condition of protecting user privacy.

We propose the PersonaWAE model, a novel conversational system which simultaneously captures user-level personalization and utterance-level information as extra hints for generating better responses. Our model is motivated by following two points: 1) existing embedding based per-

[*] Equal contribution. Ordering is decided by a coin flip.

[†] Corresponding author.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1931–1940, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

sona modeling methods cannot discover the common properties among users and train the embedding for different user independently, which cause (equal to learning) a very high-dimensional persona embedding and thus have a low data utilization efficiency or require a large amount of training data for each user. 2) benefited by the semantic capturing ability of WAEs, plenitude persona information can be gathered into the continuous space (Li et al., 2019). To this end, we build our model upon the state-of-the-art conversation model WAE (Gu et al., 2019) to model utterance-level and the user-level information. In the case, user embeddings are utilized as the condition of the prior distribution of the latent variable to formulate a WAE conditional prior. Meanwhile, to further model and fuse the utterance and user-level information, we extend these simple prior distributions to the Gaussian Mixture Distributions (GMDs, more details of the reasons in Section 2.2). After obtaining two GMDs, we combine them into a mixed distribution and regard this mixed distribution as the prior distribution of PersonaWAE. To evaluate the effectiveness of our proposed personalized conversational system, we collect a large dataset with user identifications. Experimental results on both automatic and human evaluation demonstrate that our proposed model can outperform several strong methods, and generate personalized responses for different users.

In a nutshell, our contributions can be summarized as follows:

- We proposed a novel personalized Wasserstein autoencoder (PersonaWAE) for open-domain response generation, which incorporates both utterance-level and user-level information;
- We proposed to mix two different types of Gaussian mixture distribution as the prior distribution of our model for scaling up the capability of the latent variable;
- Experiments performed on a large dataset demonstrate the effectiveness of our proposed model and achieves the new state-of-the-art results.

## 2 Preliminaries

### 2.1 VAE and WAE

Conditional VAE (CVAE) is a popular framework for dialogue generation (Zhao et al., 2017; Shen et al., 2017, 2018). CVAE, as an extension of VAE, supervises the generation process under an extra condition $c$. To train a CVAE model, the log-likelihood objective $\log p_\theta(x|c)$ is maximized through pushing up its variational lower bound:

$$\begin{aligned}\mathbb{O}(\theta,\phi,x,c) = &-\mathbf{KL}(q_\phi(z|x,c)||p_\theta(z|c)) \\ &+\mathbf{E}_{q_\phi(z|x,c)}[\log p_\theta(x|z,c)]\end{aligned} \tag{1}$$

where $q_\theta(z|x,c)$ and $p_\theta(z|c)$ represent the approximated conditional posterior and the conditional prior respectively, $\log p_\theta(x|z,c)$ represents the probability of reconstructing $x$ conditioned on both $z$ and $c$. Herein $KL(\cdot)$ represents the KL-divergence term, which serves as the regularization for encouraging the approximated posterior $q_\phi(z)$ to approach the prior $p_\theta(z)$, i.e. a standard Gaussian distribution. $E[\cdot]$ is the term of reconstruction loss, reflecting how well the decoder performs. The KL-divergence can be replaced by Wasserstein distance which is implemented by Arjovsky et al. (2017) and is proved to be superior to KL-divergence by many experiments. The conditional VAE based on Wasserstein distance is called conditional WAE.

### 2.2 Gaussian Mixture Model

In VAE/WAE frameworks, a variable in latent space is introduced for modeling information in datasets. As demonstrated in Figure 1, each gray point represents a data sample while colorized points refer to noise data. If the latent variable obeys a Gaussian distribution, noise samples (colorized points) will result in inferior responses. Alternatively, a Gaussian mixture distribution can model the datasets more accurately. Conventional VAE and WAE models usually set the prior distribution of the latent variable to a multivariate Gaussian distribution, formulated as

$$z \sim \mathcal{N}\{\mu, \sigma^2 I\} \tag{2}$$

where $\mu$ and $\sigma^2$ represent the mean and variance of $\mathcal{N}$. In our model, we utilize a Gaussian mixture distribution as the prior distribution of the latent variable $z$, written by

$$z \sim \mathcal{N}\{\pi_k, \mu_k, \sigma_k^2 I\}_{k=1}^K \tag{3}$$

where $\mu_k$ and $\sigma_k^2$ is the parameter for the $k$-th gaussian distribution in this multimodal distribution. $\pi_k$ is the weight.

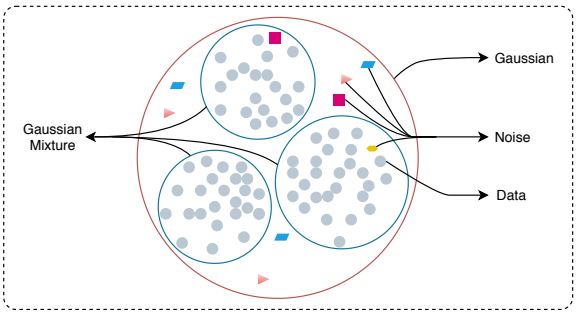


Figure 1: Distributions on latent space. The pink circle represents a Gaussian distribution while three small blue circles refer to a Gaussian mixture distribution, where gray points represent positive samples, and colorized points represent noise data.

### 2.3 Problem Formulation

We follow the conventional personalized conversation generation research (Li et al., 2016) and formulate the response generation task with the following necessary notations. A dataset with user dialogue history content $\mathcal{D} = \{(c_i, r_i, m_i)\}_{i=1}^{N}$ is firstly given, where $c_i$, $r_i$, $m_i$ represent dialogue context, response candidate, and user specific dialogue utterance respectively. Note that we treat the user dialogue utterance for extracting personalization information in multi-turn response generation. Herein, the context is formulated by: $c_i = (s_1, s_2, \cdots, s_j, \cdots, s_{n_i})$ where $s_j$ represents an utterance in the j-th turn of dialogue context and there are $n_i$ utterances in the dialogue context. $m_i$ denotes the user specific dialogue utterance. $r_i = \{r_{i,1}, r_{i,2}, \cdots, r_{i,n_r}\}$, where $n_r$ is the length of a target response $r_i$. Then, our task is defined as learning a mapping function $f(\cdot)$ from the given dataset that can yield a personalized response according to the given dialogue context and the user dialogue history.

## 3 Proposed Model

As in Figure 2, our proposed personalized Wasserstein autoencoder (PersonaWAE) consists of user personalization modeling and WAE, where details are elaborated as follows.

### 3.1 User Personalization Modeling

***Personalization Gaussian Mixture Distribution.*** To model the user-level information in the continue space, we build the Personalization Gaussian Mixture Distribution (Personalization GMD).

We train vector representations of users (Li et al., 2016) from user dialog history $\mathcal{M} = \{m_i\}_{i=1}^{N}$ as the user personalizations to facilitate personalized response generation. We denote the trained user embeddings as $\mathcal{U} = \{\mathbf{u_1}, \mathbf{u_2}, \ldots, \mathbf{u_i}\}$ where $\mathbf{u_i}$ represents the vector representations of $i$-th user (User $\mathbf{i}$).

Based on the user embeddings as $\mathcal{U}$, we utilize learned user personalizations in the latent space. Specifically, the conditional prior distribution of WAE part is a Gaussian mixture distribution (GMD) conditioned on the learned user embeddings, namely personalization GMD. We formulate the conditional prior as:

$$p(z_u|\mathbf{u_i}) = \sum_{k=1}^{K} v_k \mathcal{N}(z_u; \mu_k, \sigma_k^2 I) \tag{4}$$

where $\{\pi_k, \mu_k, \sigma_k^2 I\}_{k=1}^{K}$ represent the GMD (the distribution will deprecate to a Gaussian distribution when $K$=1) and the parameters of $k$-th Gaussian distribution are $\{\mu_k, \sigma_k^2\}$. $v_k$ is a component indicator with class probabilities $\pi_1, \pi_2, \ldots, \pi_K$, where $\pi_k$ is the mixture coefficient of the $k$-th component of the GMD. We follow (Gu et al., 2019) and compute these parameters as:

$$\begin{bmatrix} a_k \\ \mu_k \\ \log \sigma_k^2 \end{bmatrix} = W_k(\mathbf{u_i}) + b_k$$
$$\pi_k = \frac{e^{a_k}}{\sum_{i=1}^{K} e^{a_k}} \tag{5}$$

To obtain $v_k$, we use the Gumbel-Softmax reparametrization to replace the exact sampling:

$$g_i = -\log(-\log(b_i))$$
$$v_k = \frac{e^{(a_k+g_k)/\tau}}{\sum_{i=1}^{K} e^{(a_i+g_i)/\tau}} \tag{6}$$

where $b_i$ is a sample from $U(0, 1)$, and $\tau$ is the softmax temperature to control the sampling process.

***Fusion of Personalization in Decoder.*** We also incorporate the user embeddings into the decoder. Concretely, user personalization is used as the input of each updating step to obtain user-specific information for generating personalized responses. Meanwhile, $\mathbf{u_i}$ is updated by back-propagating loss signal during training. As user personalizations are high-level representations, we further introduce a gating strategy to dynamically balance the user personalization and the current conversation information.

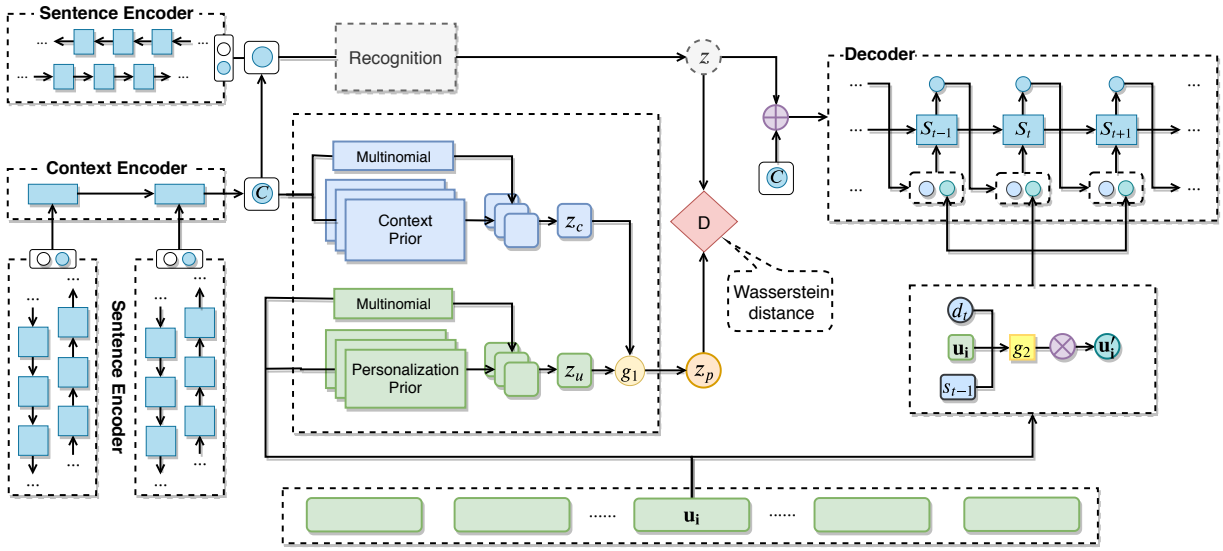


Figure 2: The detailed architecture of our proposed PersonaWAE model.

### 3.2 Personalized Wasserstein Autoencoder

Our proposed PersonaWAE consists of encoder, prior and recognition networks, and decoder.

***Encoder.*** The encoder encodes a given context by a bidirectional RNN with GRU cells following (Chen et al., 2018). Through the encoder, the context $c_i = (u_1, u_2, \cdots, u_j, \cdots, u_{n_i})$ is represented as concatenated forward and backward[1] vectors $V_c = (v_1, v_2, \cdots, v_j, \cdots, v_{n_i})$, where $v_i = [\overrightarrow{v}_i, \overleftarrow{v}_i]$. Similarly, the target response $r_i$ is represented by the concatenation of states from another bi-directional RNN with GRU cells, denoted as $v_{r,i}$. The vectors sequence $V_c$ is further processed by an RNNs and yields a vector representation $v_{c,i}$. Note that $v_{c,i}$ refers to $c$ in Equation 1 while $v_{r,i}$ represents $x$.

***Recognition and Prior Networks.*** We use a recognition network to learn the posterior $q_\theta(z|x,c)$, we hypothesize that the approximated variational posterior follows an isotropic multivariate Gaussian distribution $\mathcal{N}(\mu, \sigma^2 I)$, where $I$ represents the diagonal covariance. Thus modeling $q_\theta(z|x,c)$ is converted to learn $\mu$ and $\log \sigma^2$:

$$\begin{bmatrix} \mu \\ \log \sigma^2 \end{bmatrix} = W_o(\begin{bmatrix} x \\ c \end{bmatrix}) + b \tag{7}$$

which is presented as the recognition network in Figure 2. $W_o$ and $b$ are trainable parameters.

To approach the prior distribution, we superpose two conditional GMD, where the first one is personalization GMD as mentioned before while another conditional GMD that called context GMD is performed on context $c$. Resemble to the personalization GMD, the parameters of the context GMD $p_\phi(z_c|c)$ are defined as $a'_k$, $\mu'_k$ and $\log \sigma'^2_k$, which is learned by:

$$\begin{bmatrix} a'_k \\ \mu'_k \\ \log \sigma'^2_k \end{bmatrix} = W_{r_k}(c) + b'_k \tag{8}$$

***Fusion of two GMDs.*** In fusing personalization and context GMD, we use the weighted addition strategy to superpose these two distributions, where the resulted new distribution is the prior distribution of PersonaWAE (which is also a GMD).

$$\begin{bmatrix} w_c \\ w_u \end{bmatrix} = softmax(W_f(\begin{bmatrix} c \\ \mathbf{u_i} \end{bmatrix})) \tag{9}$$
$$z_p = w_c \cdot z_c + w_u \cdot z_u$$

where $W_f$ is a trainable parameter.

***Decoder.*** The decoder is a one-layer GRU network to output the sentence in the generation, which is shown in the right hand of Figure 2. Taking the generation of response $r_i$ as an example, the initial state of the decoder is calculated as:

$$s_{i,0} = W_d(\begin{bmatrix} z_p \\ c \end{bmatrix}) + b_d \tag{10}$$

where $W_d$ is a trainable matrix for dimension transformation. To facilitate the combination of user personalization $\mathbf{u_i}$ and decoder hidden states,

[1] $\rightarrow$ and $\leftarrow$ refer to forward and backward, respectively.

we incorporate a gate module (Tu et al., 2018) in our model:

$$g = f(\mathbf{U}s_{t-1} + \mathbf{V}d_t + \mathbf{W}\mathbf{u_i})$$
$$o_t = \mathbf{GRU}(s_{t-1}, \begin{bmatrix} d_t \\ g \cdot \mathbf{u_i} \end{bmatrix}) \quad (11)$$

where $f$ is the sigmoid funtion and $o_t$ refers to the decoder output in time step $t$. After processing $o_t$ with the softmax operation, the response $r_i$ is generated.

***Training.*** To train our proposed model, we launch the following objective to simultaneously minimize the Wasserstein distance between $p_\theta(z|c, \mathbf{u_i})$ and $q_\phi(z|x, c)$, and maximize the reconstructing probability of $x$:

$$\mathbb{L}(\theta, \phi; c, x, \mathbf{u_i}) = -\mathbf{E}_{q_\phi(z|x,c,\mathbf{u_i})} \log p(x|z, c, \mathbf{u_i}) + W(q_\phi(z|x, c)||p_\theta(z|c, \mathbf{u_i})) \quad (12)$$

## 4 Experiments

### 4.1 Dataset

To evaluate the effectiveness of our proposed personalized WAE model (PersonaWAE), we collect a dataset from an open online chatting forum, i.e., Weibo [2], which contains massive multi-turn conversation sessions and user identification information. Overall, there are 31,128,520 utterances in the raw dataset with corresponded user identifications. To construct the personalized conversation systems, we retrieve users with more than 14 utterances from the raw Weibo corpus. We also filtrate conversation sessions with less than 2 turns for training multi-turn conversation systems. We use a sliding window with a size of 3 to construct each dialogue session and there are 3 utterances in each dialogue session. By doing so, there are 336,342 conversation sessions in the cleaned corpus. We remove emojis in utterances and utilize NLTK for tokenization. Then, we randomly split the Weibo corpus into 335,342/5,000/5,000 sessions as training/validation/testing sets. For each session, the last utterance is the target response for generation while other utterances are treated as context.

### 4.2 Baselines

In our experiments, we compare our proposed method with the following highly related and strong baselines.

[2] http://www.weibo.com/

| | |
|---|---|
| Readability | Is the response grammatically formed and smooth ? |
| Informativeness | Does the response contains informative words ? |
| Personalization | Does the response resembles with any user history? |

Table 1: Criteria of human evaluation.

**Seq2Seq**, the vanilla schema of the sequence to sequence model with attention mechanism (Bahdanau et al., 2014), which is widely used in various generation tasks.

**Persona**, a typical and recent neural personalized conversation system, which incorporates user-level representations in the generation process (Li et al., 2016).

**Adaptation**, the domain adaptation solution for building personalized conversation systems (Zhang et al., 2017). We adapt the model in our scenario and we use the tf-idf to obtain the personal words as the user information.

**CVAE**, which is the conventional CVAE model and trained by KL-divergence. We change our model to use KL-divergence as the training loss.

**RL-Persona**, the personalized conversational system (Yang et al., 2018), which takes the advantages of deep reinforcement learning. We apply the method into our scenario as same as Adaptation.

**DiaWAE-GMD**, where the former is the state-of-the-art model for open-domain conversation generation (Gu et al., 2019). DiaWAE-GMD employs the Gaussian mixture prior to WAE.

### 4.3 Settings

The dimension of word embeddings is set to 200, which is initialized with pre-trained word2vec vectors [3]. The vocabulary is comprised of the most frequent 31,000 words. The sentence encoder and the context encoder in our PersonaWAE model are two bi-directional RNN with the GRU cells, respectively. The decoder consists of a one-layer RNN with GRUs. The hidden state sizes of both GRU encoder and decoder are set to 256. Each user is allocated a user-level vector representation with dimension size 512. We set the mini-batch size to 100. The SGD optimizer is used to train the autoencoder module with the initial learning rate 1.0, and the learning rate decay strategy is employed. We use RMSprop optimizer (Hinton et al., 2012) to update the parameters of the generator and the discriminator, where the initial learn-

[3] https://github.com/tmikolov/word2vec

| Models | Embedding Metrics | | | BLEU | | | Human Evaluation | | |
|---|---|---|---|---|---|---|---|---|---|
| | Extrema | Greedy | Average | Recall | Precision | F1 | Read. | Info. | P-score |
| **Seq2Seq** | 0.1640 | 0.4098 | 0.4911 | 0.1646 | 0.1646 | 0.1646 | 2.30 | 2.16 | 0.49 |
| **Persona** | 0.1631 | 0.3982 | 0.4871 | 0.1646 | 0.1646 | 0.1646 | 2.31 | 2.15 | 0.50 |
| **Adaptation** | 0.1722 | 0.4038 | 0.5113 | 0.1689 | 0.1689 | 0.1689 | 2.29 | 1.93 | 0.47 |
| **CVAE** | 0.2643 | 0.2911 | 0.5759 | 0.1931 | 0.1636 | 0.1771 | 2.02 | 2.33 | 0.45 |
| **RL-Persona** | 0.1694 | 0.4536 | 0.4906 | 0.1723 | 0.1723 | 0.1723 | 2.21 | 2.22 | 0.63 |
| **DiaWAE-GMD** | 0.4387 | 0.4752 | 0.7573 | **0.3409** | 0.1710 | 0.2277 | 2.31 | 2.35 | 0.50 |
| **PersonaWAE** | **0.4542** | **0.5914** | **0.7585** | 0.3365 | **0.1806** | **0.2350** | **2.33** | **2.37** | **0.66** |
| **ground-truth** | | | | | | | 2.73 | 2.66 | 0.86 |

Table 2: The results of both automatic evaluations and human evaluation. Read., Info., P-score refer to readability, informativeness, personalization scores. The kappa value between human annotators is 0.41, which indicates a moderate inter-rater.

| # Distributions in GMD | Embedding Metrics | | | BLEU-F1 |
|---|---|---|---|---|
| | Extrema | Greedy | Average | |
| **1** | 0.4114 | 0.3110 | 0.7026 | 0.1547 |
| **2** | 0.4228 | 0.4494 | 0.7135 | 0.1543 |
| **3** | **0.4542** | **0.5914** | **0.7585** | 0.2350 |
| **4** | 0.4013 | 0.5764 | 0.7256 | 0.2150 |
| **5** | 0.4382 | 0.5680 | 0.7490 | **0.2480** |

Table 3: The results of different settings of $k$ in personalization GMD, where $k$ denotes number of distributions. When $k = 1$, it is a Gaussian distribution.

| Models | Embedding Metrics | | | BLEU-F1 |
|---|---|---|---|---|
| | Extrema | Greedy | Average | |
| **PersonaWAE** | **0.4542** | **0.5914** | 0.7585 | **0.2350** |
| **w/o prior** | 0.4360 | 0.4848 | 0.7479 | 0.2256 |
| **w/o fusion** | 0.4293 | 0.4813 | **0.7588** | 0.2244 |

Table 4: The results of Ablation Experiments. w/o denotes without. Fusion represents fusion of personalization in decoder.

ing rates are set to $5e$-5 and $1e$-5, respectively. The gradient penalty is used for training discriminator (Gulrajani et al., 2017). The value of $\tau$ in Gumbel softmax is set to 0.1.

### 4.4 Evaluation Metrics

To evaluate the results of the generated responses, we adopt the following metrics widely used in existing research.

**Embedding Metrics.** To capture the semantic matching degrees between generated responses and ground-truth, we perform evaluations on embedding space. In consistent with previous study (Gu et al., 2019), we compute the similarity between the bag-of-words (BOW) embeddings representations of generated results and reference. In particular, we calculate three metrics:1) *Greedy* (BOW-Greedy), i.e., greedily matching words in two utterances based on the cosine similarities, and the total scores is then averaged across all words (Rus and Lintean, 2012); 2) *Average* (BOW-Average), cosine similarity between the averaged word embeddings in the two utterances (Mitchell and Lapata, 2008); 3) *Extrema* (BOW-Extrema), cosine similarity between the largest extreme values among the word embeddings in the two utterances (Forgues et al., 2014). We report the maximum BOW embedding scores of the 10 sampled responses for each testing context.

**Overlap-based Metric.** We utilize BLEU score (Papineni et al., 2002) to measure n-grams overlaps between ground-truth and generated response. Specifically, we follow the conventional setting in previous work (Gu et al., 2019) to compute BLEU scores using smoothing techniques (smoothing 7) [4]. For each testing context, we sample 10 responses from the models and compute their BLEU scores, i.e., n-gram precision (BLEU-Precision), n-gram recall (BLEU-Recall), and n-gram F1 (BLEU-F1).

**Human Evaluation.** We also employ human evaluation to assess the responses generated by our model and the baselines. Three well-educated annotators are hired to evaluate the quality of generated responses, where the evaluation is conducted in a double-blind fashion. Totally, 200 randomly

[4] http://www.nltk.org/_modules/nltk/translate/bleu_score.html

| | |
|---|---|
| -这孩子，太稚嫩了，真想踹一脚<br>(This child is so immature that I really want to kick him.)<br>-wow，你大粗腿，能抬多高啊<br>(Wow, how high can you get with those big legs.) | |
| ground truth | 你要不要试试？见证一下我能办得到！<br>(Try it ? Prove I can do it.) |
| RL-Persona | 哈哈，那就好了<br>(Haha, it's all right.) |
| DiaWAE-GMD | 啊，完了，我就不敢说<br>(Game over, I can't talk it.) |
| PersonaWAE | **我拿小拳拳捶你**<br>**(I'll punch you with my fist.)** |
| -你什么时候回香港?<br>(When are you going back to HongKong?)<br>-签证好了就回去(when visa is ready.) | |
| ground truth | 要半个月嘞<br>(About half a month.) |
| RL-Persona | 我也不知道，我也不知道<br>(I don't know, I don't know.) |
| DiaWAE-GMD | 是因为我没在你宿舍里<br>(Because I'm not in your dormitory.) |
| PersonaWAE | **哈哈，回来约，约个时间**<br>**(Ha, play togther when you are back.)** |

Table 5: Responses generated by baselines and our model.

| | |
|---|---|
| -求种草水乳，性价比高点的。<br>(Please recommend cost-effective make-up water and lotion to me.)<br>-我水乳用的老慢啦哈哈，感觉两年用一套<br>(I use make-up water and lotion so slow that one can be used for 2 years.) | |
| User1 | [UNK]，谢谢，我的小U<br>([UNK], thank you, my xiaoU (name)) |
| User2 | 哈哈，么么哒。<br>(Haha, thank you and I love you.) |
| User3 | 小姐姐我买了(Ok, I buy it) |
| User4 | 是，哈哈(Yes, Haha) |
| -这孩子，太稚嫩了，真想踹一脚<br>(This child is so immature that I really want to kick him.)<br>-wow，你大粗腿，能抬多高啊<br>(Wow, how high can you get with those big legs.) | |
| User1 | 我被风吹走了(I am so thin that i'm almost blown away by the wind.) |
| User2 | 我今天，小仙女(Today, I am a fairy !) |
| User3 | 哈哈，好(Haha, OK.) |
| User4 | 哈哈，我瘦了(Haha, I am thin !.) |
| -哈哈演不下去了<br>(Haha! I can't play anymore.)<br>-早已看穿一切<br>(I have seen everything before.) | |
| User1 | 我看了一遍，我觉得我很帅<br>(I watch it again, I think I am handsome.) |
| User2 | 哈哈，好哒好，等我<br>(Haha, get it, wait for me.) |
| User3 | 哈哈，服(Haha, come on.) |
| User4 | 哈哈，太可怕(Haha, it's terrible.) |

Table 6: Responses generated by our model for four users. We fix the context and test the effects of different user personalizations.

sampled responses generated by each model are rated by each annotator with three different aspects, i.e., readability, informativeness, and personalization. Details of the criteria are illustrated in Table 1. Note that it is very difficult to judge whether a generated response resembles with the style of the corresponding user history utterances, and thus we rate the personalization with {0,1}, representing bad or normal. Other criteria are scored from 1 to 3, i.e., bad, normal, and good.

## 5 Results and Analysis

In this section, we perform automatic evaluations and human evaluation to measure the quality of the generated responses quantitatively. Meanwhile, we also conduct a qualitative study to intuitively analyze the generated results. Table 2 presents the results of automatic and human evaluation.

### 5.1 The Effect of WAE

***WAE can effectively improve the quality of responses but fails to capture personalization.*** As we mentioned before, we intend to improve the response quality by using WAE. Unsurprisingly, Seq2Seq gets the worst performance. Comparing DiaWAE-GMD with CVAE, we can observe that DiaWAE-GMD significantly improves BLEU scores and BOW scores upon CVAE, which is shown in Table 2. Such results indicate that the Wasserstein distance and the adversarial training can enhance model learning and address KL-vanishing issue in VAEs, as a result of which achieves better results of generated responses, which is also confirmed in the previous research (Gu et al., 2019). Besides, human evaluation results in Table 4 further illustrate that DiaWAE-GMD fails to model personalization of different users insomuch as DiaWAE-GMD lacks user-level information learning.

***The number of distributions in conditional Gaussian mixture distribution significantly alter model performance.*** Table 3 presents the ablation results of the influence of $k$ value in personaliza-

tion Gaussian mixture distribution. It is observed that when $k \leq 3$ , model performance improves with the increasing of $k$, which suggests that more distributions in GMD are helpful for modeling user personalization. However, for $k > 3$, model performance slightly drops with the increasing of $k$, The potential reason is GMD with three distributions is effective enough for modeling personalization, and sophisticated GMD might suffer scarce datasets for training.

### 5.2 The Influence of Personalization Modeling

***User embeddings substantially improve the quality of generated response and introduce personalizations for different users.*** Through conducting the comparison between PersonaWAE and DiaWAE-GMD. We can learn that incorporating user personalization in decoding step substantially enhance the personalization score of human evaluations, which means user embeddings and the combination in decoder has a positive influence on response quality.

***Incorporating personalization in the conditional GMD prior is more effective than combing personalization in decoder.*** As shown in Table 2, Persona model only achieves comparable results with Seq2Seq in terms of BLEU scores and BOW scores. For PersonaWAE and DiaWAE-GMD, incorporating personalizations in both decoder and the latent space yields a performance improvement. For the BLEU-Recall, which PersonaWAE does not outperform than DiaWAE-GMD, a possible explanation for this might be that PersonaWAE model the personalization information and generation may be more limited.

### 5.3 Discussion

Overall, PersonaWAE outperforms all other baselines on both automatic and human evaluations. Especially for personalization modeling, PersonaWAE achieves a noticeable achievement over the strong baselines DiaWAE-GMD and RL-Persona. These results support that our proposed PersonaWAE is effective in generating personalized response. We also observe that fusing personalized GMD and context GMD as the conditional prior is also useful, which is proven by the results shown in Table 4.

### 5.4 Case Study

Table 5 illustrates the generated response of different models for a given context. We can observe that our proposed model can generate responses with readability and personalization information. Table 6 shows a few example responses generated by altering the user personalization information. With different user representations, the generated responses change, which supports that personalization representation introduced in our model helps learn user-level information. Although it is difficult to evaluate the personalization of generated response and there exists a gap between generated responses and human-comprised ones, quality improvement of responses is achieved. Moreover, we observe that our proposed model might generate a too long and repeated sentence in extreme cases. The potential reason might be the relative short dialog history for each user. Besides, explicitly extracting knowledge and user personalization from conversation history is also promising. These results point out the direction of future work.

## 6 Related Work

Constructing an automatic conversation system is an attractive and prevalent task within the community of artificial intelligence. Previous studies mainly focus on vertical domains by applying rule- and template-based models (Pieraccini et al., 2009). Later on, with the explosive growth of data, the application of open-domain conversation model is promising. Conventional methods in vertical domains have obstacles to scale to open domain. Given this, various data-driven approaches have been proposed for modeling open-domain conversation, including retrieval-based methods (Yan et al., 2016; Tao et al., 2019), statistical machine translation model (Ritter et al., 2011), and neural networks (Serban et al., 2015; Hu et al., 2019).

Recently, building a personalized conversation system has been attached more attention, e.g., implicitly learning user personalizations from dialog history (Li et al., 2015), explicitly collecting and modeling user profiles as personalizations for generating personalized responses (Zhang et al., 2017, 2018). To improve wording diversity, CVAE models (Serban et al., 2017; Zhao et al., 2017; Shen et al., 2018) are well-investigated for open-domain response generation. As the extension of

CVAE, Wasserstein autoencoder (Gu et al., 2019) is also used for open-domain response generation to solve the issues of posterior collapse and vanishing latent variables. We build our model upon both the advantages of WAE and personalization modeling for personalized response generation.

## 7 Conclusion

Open domain response generation is a challenging task, which involves automatically comprising a response with informative words and personalization. Although prompting progress has been made in wording informativeness, there still exists a noticeable gap between generated response and those created by humans, especially in personalization modeling. To fill this gap, we propose a personalized Wasserstein autoencoder (PersonaWAE) for response generation, where the WAE module improves informativeness by using a continuous latent variable with GMD and user vector representations learned from dialog history is used for introducing personalization information. Experimental results on a large dataset indicate that our proposed model can generate better responses, and outperforms existing models under both automatic and human evaluations.

## Acknowledgments

We would like to thank the reviewers for their constructive comments. This work was supported by the National Key Research and Development Program of China (No. 2017YFC0804001), the National Science Foundation of China (NSFC No. 61876196 and NSFC No. 61672058). Rui Yan was sponsored by Alibaba Innovative Research (AIR) Grant. Rui Yan is the corresponding author.

## References

Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein gan. *arXiv preprint arXiv:1701.07875*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *Computer Science*.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*.

Xiuying Chen, Shen Gao, Chongyang Tao, Yan Song, Dongyan Zhao, and Rui Yan. 2018. Iterative document representation learning towards summarization with polishing. *EMNLP*.

Gabriel Forgues, Joelle Pineau, Jean-Marie Larchevêque, and Réal Tremblay. 2014. Bootstrapping dialog systems with word embeddings. In *Nips, workshop*, volume 2.

Nadine Glas, Ken Prepin, and Catherine Pelachaud. 2015. Engagement driven topic selection for an information-giving agent. In *SemDial*.

Xiaodong Gu, Kyunghyun Cho, Jung-Woo Ha, and Sunghun Kim. 2019. DialogWAE: Multimodal response generation with conditional wasserstein auto-encoder. In *International Conference on Learning Representations*.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. 2017. Improved training of wasserstein gans. In *NIPS*, pages 5767–5777.

Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. 2012. Rmsprop: Divide the gradient by a running average of its recent magnitude. *Neural networks for machine learning, Coursera lecture 6e*.

Wenpeng Hu, Zhangming Chan, Bing Liu, Dongyan Zhao, Jinwen Ma, and Rui Yan. 2019. Gsn: A graph-structured network for multi-party dialogues. *arXiv preprint arXiv:1905.13637*.

Diederik P Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. *stat*, 1050:10.

Feng-Lin Li, Minghui Qiu, Haiqing Chen, Xiongwei Wang, Xing Gao, Jun Huang, Juwei Ren, Zhongzhou Zhao, Weipeng Zhao, Lei Wang, et al. 2017. Alime assist: an intelligent assistant for creating an innovative e-commerce experience. In *CIKM*, pages 2495–2498.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2015. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055*.

Jiwei Li, Michel Galley, Chris Brockett, Georgios Spithourakis, Jianfeng Gao, and Bill Dolan. 2016. A persona-based neural conversation model. In *ACL*.

Juntao Li, Lisong Qiu, Bo Tang, Dongmin Chen, Dongyan Zhao, and Rui Yan. 2019. Insufficient data can also rock ! learning to converse using smaller data with augmentation.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. *NAACL-HLT*, pages 236–244.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318. ACL.

Roberto Pieraccini, David Suendermann, Krishna Dayanidhi, and Jackson Liscombe. 2009. Are we there yet? research in commercial spoken dialog systems. In *International Conference on Text, Speech and Dialogue*, pages 3–13.

Alan Ritter, Colin Cherry, and William B. Dolan. 2011. Data-driven response generation in social media. In *Conference on Empirical Methods in Natural Language Processing*, pages 583–593.

Vasile Rus and Mihai Lintean. 2012. A comparison of greedy and optimal assessment of natural language student input using word-to-word similarity metrics. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 157–162. ACL.

Iulian V. Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2015. Building end-to-end dialogue systems using generative hierarchical neural network models. (4):3776–3783.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*, pages 3295–3301.

Xiaoyu Shen, Hui Su, Yanran Li, Wenjie Li, Shuzi Niu, Yang Zhao, Akiko Aizawa, and Guoping Long. 2017. A conditional variational framework for dialog generation. *arXiv preprint arXiv:1705.00316*.

Xiaoyu Shen, Hui Su, Shuzi Niu, and Vera Demberg. 2018. Improving variational encoder-decoders in dialogue generation. *arXiv preprint arXiv:1802.02032*.

Heung-Yeung Shum, Xiao-dong He, and Di Li. 2018. From eliza to xiaoice: challenges and opportunities with social chatbots. *Frontiers of Information Technology & Electronic Engineering*, 19(1):10–26.

Chongyang Tao, Wei Wu, Can Xu, Wenpeng Hu, Dongyan Zhao, and Rui Yan. 2019. One time of interaction may not be enough: Go deep with an interaction-over-interaction network for response selection in dialogues. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 1–11.

Zhaopeng Tu, Yang Liu, Shuming Shi, and Tong Zhang. 2018. Learning to remember translation history with a continuous cache. *Transactions of the Association of Computational Linguistics*, 6:407–420.

Rui Yan, Song Yiping, and Wu Hua. 2016. Learning to respond with deep neural networks for retrieval-based human-computer conversation system. In *SIGIR*, pages 55–64.

Min Yang, Qiang Qu, Kai Lei, Jia Zhu, Zhou Zhao, Xiaojun Chen, and Joshua Z Huang. 2018. Investigating deep reinforcement learning techniques in personalized dialogue generation. In *SIAM*, pages 630–638.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? In *ACL*, pages 2204–2213.

Wei-Nan Zhang, Qingfu Zhu, Yifa Wang, Yanyan Zhao, and Ting Liu. 2017. Neural personalized response generation as domain adaptation. *WWW*, pages 1–20.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. *ACL*.

# Variational Hierarchical User-based Conversation Model

**JinYeong Bak**
School of Computing
KAIST
jy.bak@kaist.ac.kr

**Alice Oh**
School of Computing
KAIST
alice.oh@kaist.edu

## Abstract

Generating appropriate conversation responses requires careful modeling of the utterances and speakers together. Some recent approaches to response generation model both the utterances and the speakers, but these approaches tend to generate responses that are overly tailored to the speakers. To overcome this limitation, we propose a new model with a stochastic variable designed to capture the speaker information and deliver it to the conversational context. An important part of this model is the network of speakers in which each speaker is connected to one or more conversational partner, and this network is then used to model the speakers better. To test whether our model generates more appropriate conversation responses, we build a new conversation corpus containing approximately 27,000 speakers and 770,000 conversations. With this corpus, we run experiments of generating conversational responses and compare our model with other state-of-the-art models. By automatic evaluation metrics and human evaluation, we show that our model outperforms other models in generating appropriate responses. An additional advantage of our model is that it generates better responses for various new user scenarios, for example when one of the speakers is a known user in our corpus but the partner is a new user. For replicability, we make available all our code and data[1].

## 1 Introduction

In conversation response generation, modeling the speakers in addition to the utterances is important for generating appropriate responses. Knowing information about a speaker, such as her linguistic style or personal information can help predict her response, and knowing more about both speakers from their previous conversations can help predict the contents of their conversation. Some recent work models the speakers in addition to the utterances (Li et al., 2016b; Olabiyi et al., 2018), but these models tend to overly emphasize the speaker such that they generate very similar responses even when the previous utterances are different. Another difficult and under-tackled problem in conversation response generation is the cold start problem, when the training data do not contain one or both of the speakers. It then becomes very difficult to predict the appropriate responses.

In this paper, we propose Variational Hierarchical User-based Conversation Model (VHUCM) which has a stochastic variable conditioned on the speakers and affects the context. We generate the stochastic variable from a prior distribution whose parameters are given by feed-forward neural networks. The inputs of the neural networks are two speakers that are represented as vector embeddings. Then, we use an RNN to infer the conversational context from the utterances and the stochastic variable. During training, we sample the variable from the variational distribution whose inputs are the speakers and utterances and minimize the difference between the distributions. The stochastic variable models the speakers to a more appropriate degree than the previous models because it is generated from the learned distribution. With VHUCM, we devise a simple solution to the cold start problem by initializing the embeddings of the new speakers by combining the embeddings of their conversation partners. This is based on the assumption that two speakers are close in the embedding space when they have conversations because people try to minimize the social difference among themselves when they have conversations (Linell et al., 1991).

To evaluate VHUCM, we build a new corpus that better reflects the real life scenario of multiple

[1] https://github.com/NoSyu/VHUCM



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1941–1950, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

speakers interconnected in a social network. This corpus contains naturally-occurring conversations over a long period, each conversation having more than just everyday greetings. More details about the corpus can be found in Section 3.

We evaluate VHUCM with this new corpus, and in comparison with other conversation models, the responses generated by VHUCM score the highest using automatic evaluation metrics as well as human evaluation. We show two additional advantages of VHUCM: 1) it can generate personalized responses based on the user and conversation partner, and 2) VHUCM with user embeddings can solve the new user cold start problem.

Our contributions in this paper include the following. First, we collect a large and longitudinal conversation corpus from Twitter (Sec 3). Second, we present VHUCM, a new conversation model which captures the speaker information more effectively and leverages the network of the speakers for better speaker embedding (Sec 4). Third, we conduct the response generation experiments with VHUCM and other models and show that VHUCM outperforms the others (Sec 5). Last, we show how to approach the new user problem with VHUCM and speaker embedding (Sec 6).

## 2 Related Work

Dialogue response generation has been extensively studied (Ratnaparkhi, 2002; Ritter et al., 2011), and recently, neural network models, especially sequence-to-sequence models have been widely used (Sordoni et al., 2015; Serban et al., 2017; Park et al., 2018; Du et al., 2018; Gu et al., 2019). One limitation of basic seq2seq models is that they only generate responses to the immediately preceding utterances, whereas people usually respond to the entire dialogue consisting of multiple previous utterances. To overcome this limitation, Hierarchical recurrent encoder-decoder (HRED) (Sordoni et al., 2015) builds one more RNN that models the dependency over the utterances in the conversation. VHUCM also constructs the hierarchical RNN structure to understand the previous utterances.

Recently, latent variable models based on Conditional Variational Auto-Encoder (CVAE) (Kingma et al., 2014) or Generative Adversarial Network (GAN) (Goodfellow et al., 2014) show the better performance for generating response (Serban et al., 2017; Xu et al., 2017; Li et al., 2017a; Park et al., 2018). We adopt CVAE to VHUCM and compare the performance with GAN based model (Gu et al., 2019).

Modeling of speakers in the conversation model has been studied (Li et al., 2016b; Xing and Fernández, 2018). They incorporate the speakers to generate the responses, but Li et al. (2016b) only considers a short context of the conversation. Olabiyi et al. (2018) overcomes this, but the user information is still in the utterance level. This approach tends to generate the same response for the same speaker even when the given utterances are different. This is because it gives too much importance to the speaker rather than the content of the previous utterances. VHUCM differs from these models in that it uses a global stochastic variable which is conditioned on the speakers and affects the context.

| Users | Dyads | Conv's | Utterances |
|---|---|---|---|
| 27,152 | 107,611 | 770,739 | 6,109,469 |

Table 1: Basic statistics of Twitter conversations corpus

## 3 Twitter Conversation Corpus

In this section, we describe our new Twitter conversations corpus. We first explain how we build the corpus, then we compare it with other conversation corpora.

### 3.1 Definition and Basic Statistics

We define a Twitter conversation as a chain of tweets where two users are consecutively replying to each other's tweets using the Twitter reply button. Unlike other research using Twitter conversation corpora (Bak et al., 2014; Li et al., 2016b), we increase the minimum number of the tweets in a conversation from three to five because in many cases, the first few utterances are greetings such as "How are you". To model the users in-depth, we keep conversations only from dyads with ten or more conversations and users with three or more conversational partners.

Our Twitter conversation corpus consists of 27,152 users, 107,611 dyads, 770,739 conversations and 6,109,469 tweets which were posted between April 2007 to June 2013. The average duration of each conversation is around 4.5 hours, and the average duration between the first and the last conversations of each dyad is around 110 days. Table 1 summarizes the corpus.

### 3.2 Other Conversation Corpora

Our corpus has two major differences from existing conversation corpora. First major difference is that our corpus consists of open-domain naturally-occurring conversations. The widely-used Cornell movie corpus (Danescu-Niculescu-Mizil and Lee, 2011) and TV series transcripts (Li et al., 2016b) are made up conversations written by the script writers. The Ubuntu dialog corpus (Lowe et al., 2015) consists of naturally-occurring conversations, but the topics are limited to a specific computer OS. Mazare et al. (2018) creates a corpus perhaps closest to open-domain naturally-occurring conversations, but Reddit comments are in the form of discussions, rather than personal casual conversations.

Another property of our corpus is that it is large and longitudinal. It contains ten or more conversations by the same two users, over several months, and the corpus contains hundreds of thousands of such dyads and their conversations. Existing Twitter conversation corpora (Ritter et al., 2011; Li et al., 2016b) consist of short conversations, with about three turns in each conversation. The DailyDialog corpus (Li et al., 2017b) contains daily conversations from English learners, but the corpus size is relatively small, and it does not have a user indicator to identify the speakers. Zhang et al. (2018) builds a dialogue corpus for personalized responses, but the corpus is generated by crowd workers in a controlled setting, and the corpus is relatively small.

## 4 Variational Hierarchical User-based Conversation Model

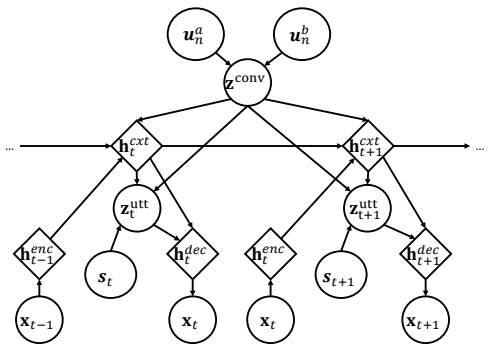


Figure 1: Graphical representation of VHUCM. The global context of the conversation $\mathbf{z}^{conv}$ is inferred by the speakers of the conversation.

This section describes our model, the Variational Hierarchical User-based Conversation Model (VHUCM) which explicitly models the speakers as well as the input utterances. VHUCM is based on the Variational Hierarchical Conversation RNN (VHCR) (Park et al., 2018), but in generating the conversation level latent variable $\mathbf{z}^{conv}$ and the uttrerance level latent variable $\mathbf{z}_t^{utt}$, we incorporate the user embedding. We further propose to initialize the user embedding from the conversation network described in section 4.3.

### 4.1 Notations

We have $N$ conversations in the data $\{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_N\}$, where the $n$-th conversation has $M_n$ utterances by two users $u_n^a$ and $u_n^b$. The $t$-th utterance in the conversation has word sequence $\mathbf{x}_{nt}$ and speaker indicator $s_{nt}$. Hence, $\mathbf{c}_n = \{(\mathbf{x}_{n1}, s_{n1}), \ldots, (\mathbf{x}_{nM_n}, s_{nM_n})\}$ and $s_{nt} \in \{u_n^a, u_n^b\}$ where $t \in \{1, \ldots, M_n\}$.

### 4.2 VHUCM

The structure of VHUCM is similar to the VHCR, which has three RNNs (encoder RNN $f_{\boldsymbol{\theta}}^{enc}$, context RNN $f_{\boldsymbol{\theta}}^{cxt}$, and decoder RNN $f_{\boldsymbol{\theta}}^{dec}$) and two latent random variables ($\mathbf{z}_t^{utt}$ and $\mathbf{z}^{conv}$).

Given previous $t-1$ utterances[2], VHUCM generates the word sequence of the next utterance $\mathbf{x}_t$ as follows:

$$\boldsymbol{h}_{t-1}^{enc} = f_{\boldsymbol{\theta}}^{enc}(\mathbf{x}_{t-1})$$
$$\boldsymbol{h}_t^{cxt} = f_{\boldsymbol{\theta}}^{cxt}(\boldsymbol{h}_{t-1}^{cxt}, \boldsymbol{h}_{t-1}^{enc}, \mathbf{z}^{conv}).$$

We use the embeddings of the two users in the conversation $\boldsymbol{h}_{u^a}^{user}$ and $\boldsymbol{h}_{u^b}^{user}$ to create dyad-specific context $\mathbf{z}^{conv}$ as follows:

$$p_{\boldsymbol{\theta}}(\mathbf{z}^{conv} | \boldsymbol{h}_{u^a}^{user}, \boldsymbol{h}_{u^b}^{user}) = \mathcal{N}(\mathbf{z} | \boldsymbol{\mu}^d, \boldsymbol{\sigma}^d \mathbf{I})$$
$$\boldsymbol{\mu}^d = \text{MLP}_{\boldsymbol{\theta}}(\boldsymbol{h}_{u^a}^{user}, \boldsymbol{h}_{u^b}^{user})$$
$$\boldsymbol{\sigma}^d = \text{Softplus}(\text{MLP}_{\boldsymbol{\theta}}(\boldsymbol{h}_{u^a}^{user}, \boldsymbol{h}_{u^b}^{user})).$$

We also use the user embeddings to understand the users' words as follows:

$$p_{\boldsymbol{\theta}}(\mathbf{z}_t^{utt} | \mathbf{x}_{<t}, \mathbf{z}^{conv}, \boldsymbol{h}_{s_t}^{user}) = \mathcal{N}(\mathbf{z} | \boldsymbol{\mu}_t, \boldsymbol{\sigma}_t \mathbf{I})$$
$$\boldsymbol{\mu}_t = \text{MLP}_{\boldsymbol{\theta}}(\boldsymbol{h}_t^{cxt}, \mathbf{z}^{conv}, \boldsymbol{h}_{s_t}^{user})$$
$$\boldsymbol{\sigma}_t = \text{Softplus}(\text{MLP}_{\boldsymbol{\theta}}(\boldsymbol{h}_t^{cxt}, \mathbf{z}^{conv}, \boldsymbol{h}_{s_t}^{user})).$$

To decode the words from the context, we use the decoder RNN with context-related variables.

$$p_{\boldsymbol{\theta}}(\mathbf{x}_t | \mathbf{x}_{<t}, \mathbf{z}_t^{utt}, \mathbf{z}^{conv}) = f_{\boldsymbol{\theta}}^{dec}(\boldsymbol{h}_t^{cxt}, \mathbf{z}_t^{utt}, \mathbf{z}^{conv})$$

For the inference of $\mathbf{z}^{conv}$, we adopt the idea of VHCR that uses a bi-directional GRU $f^{conv}$ where

[2] In order to be concise, we remove the notation $n$.

the input is the encoder RNN outputs.

$$q_\phi(\mathbf{z}^{conv}|\mathbf{c}, \boldsymbol{h}^{user}_{u^a}, \boldsymbol{h}^{user}_{u^b}) = \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}^{Id}, \boldsymbol{\sigma}^{Id}\mathbf{I})$$

$$\boldsymbol{h}^{conv} = f^{conv}(\boldsymbol{h}^{enc}_1, \ldots, \boldsymbol{h}^{enc}_N)$$

$$\boldsymbol{\mu}^{Id} = \text{MLP}_\phi(\boldsymbol{h}^{conv}, \boldsymbol{h}^{user}_{u^a}, \boldsymbol{h}^{user}_{u^b})$$

$$\boldsymbol{\sigma}^{Id} = \text{Softplus}(\text{MLP}_\phi(\boldsymbol{h}^{conv}, \boldsymbol{h}^{user}_{u^a}, \boldsymbol{h}^{user}_{u^b}))$$

To infer $\mathbf{z}^{utt}_t$, we build the additional networks:

$$q_\phi(\mathbf{z}^{utt}_t|\mathbf{c}, \mathbf{z}^{conv}) = \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}'_t, \boldsymbol{\sigma}'_t\mathbf{I})$$

$$\boldsymbol{\mu}'_t = \text{MLP}_\phi(\mathbf{x}_t, \boldsymbol{h}^{cxt}_t, \mathbf{z}^{conv}, \boldsymbol{h}^{user}_{s_t})$$

$$\boldsymbol{\sigma}'_t = \text{Softplus}(\text{MLP}_\phi(\mathbf{x}_t, \boldsymbol{h}^{cxt}_t, \mathbf{z}^{conv}, \boldsymbol{h}^{user}_{s_t})).$$

The loss function of VHUCM is the ELBO loss and the auxiliary loss. The ELBO loss is the combination of the reconstruction loss, and the KL divergence between $p_\theta$ and $q_\phi$ with KL cost annealing (Bowman et al., 2016). We also add the bag-of-word loss (Zhao et al., 2017) as the auxiliary loss to avoid the vanishing latent variable problem.

### 4.3 VHUCM-PUE

We leverage the fact that people connected in a social network can be modeled with the node2vec algorithm (Grover and Leskovec, 2016), where the nodes are users, and two nodes are connected if they have conversations. We can use the number of conversations as the weight of the edge. We extend the original VHUCM with these pre-trained user embeddings based on the conversation network, and we can call this VHUCM-PUE (VHUCM with Pre-trained User Embeddings). This approach is similar to soc2seq (Bhatia et al., 2017) which embeds the users more elaborately using comments and likes but is a much simpler conversational model based on the encoder-decoder.

## 5 Experiment 1 - Response Quality

This section describes the experiments and results of VHUCM and VHUCM-PUE compared to other state-of-the-art response generation models. We compare the models using various automatic evaluation metrics in section 5.2 and with human evaluation in section 5.3. Section 5.4 presents a qualitative analysis of the generated responses, in particular how the responses can be personalized.

### 5.1 Experiment Setup

**Data** We use the corpus described in section 3 for this experiment. We split the data as 80/10/10 for training/validation/test, because VHUCM learns the user information from the conversation, we split the conversations of each dyad in chronological order and merge each part into overall training, validation and test sets. We further investigate cases of new users that are not in the training data but appear in the test data in section 6.

**Comparison models** We compare VHUCM and VHUCM-PUE with the following models:

- *HRED* (Sordoni et al., 2015): A hierarchical structure of RNNs for encoder, decoder, and context.
- *VHRED* (Serban et al., 2017): A variational autoencoder model that adds latent variables $\mathbf{z}^{utt}_t$ to the HRED.
- *VHCR* (Park et al., 2018): A variational autoencoder model that adds the latent variable $\mathbf{z}^{conv}$ to the VHRED.
- *SpeakAddr* (Li et al., 2016b): A persona seq2seq model. We choose the Speaker-Addressee model as this outperforms the Speaker model.
- *DialogWAE* (Gu et al., 2019): A conditional Wasserstein autoencoder model that uses GAN for training. We use DialogWAE-GMP which outperforms DialogWAE by using the Gaussian mixture prior.

### 5.2 Quantitative Analysis

| Model | Perp | Model | Perp |
|---|---|---|---|
| HRED | 72.6 | SpeakAddr | 85.5 |
| VHRED | 71.2 | DialogWAE | 81.0 |
| VHCR | 71.1 | VHUCM | 65.3 |
| | | VHUCM-PUE | **62.7** |

Table 2: Perplexity per word of the generated responses. VHUCM-PUE outperforms all other methods.

**Metrics** To quantitatively compare the response generation performance of VHUCM-PUE with other models, we use various automatic metrics used in (Park et al., 2018; Du et al., 2018; Olabiyi et al., 2018; Gu et al., 2019).

We first compute the perplexity per word of each model using the reconstruction errors of the test data. For variational models, we calculate the values by importance sampling. As shown in Table 2, VHUCM-PUE outperforms the other methods in terms of word perplexity.

| Model | BLEU | Embedding | | | ROUGE-L | | | Distinct | | Len |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Avg | Ext | Gre | Rec | Prec | F | dist-1 | dist-2 | |
| HRED | 0.090 | 0.577 | 0.364 | 0.357 | 0.064 | 0.162 | 0.066 | 0.019 | 0.072 | 9.4 |
| VHRED | 0.120 | 0.596 | 0.368 | 0.377 | 0.072 | 0.161 | 0.072 | 0.016 | 0.063 | 11.4 |
| VHCR | 0.137 | 0.599 | 0.371 | 0.381 | 0.076 | 0.169 | 0.075 | 0.020 | 0.076 | **12.3** |
| SpeakAddr | 0.037 | 0.567 | **0.384** | 0.337 | 0.052 | **0.218** | 0.055 | 0.016 | 0.031 | 4.8 |
| DialogWAE | 0.127 | 0.586 | 0.345 | 0.369 | 0.079 | 0.132 | 0.080 | 0.012 | 0.104 | 11.5 |
| VHUCM | 0.120 | 0.633 | 0.373 | 0.394 | 0.075 | 0.154 | 0.079 | 0.030 | 0.108 | 10.1 |
| VHUCM-PUE | **0.161** | **0.643** | 0.376 | **0.400** | **0.082** | 0.162 | **0.087** | **0.034** | **0.123** | 10.6 |

Table 3: Comparison of various models using BLEU, Embedding, and ROUGE-L scores, which measure the quality of the generated responses with respect to the ground truth. Embedding Avg, Ext, and Gre are average, extrema and greedy matching by embedding based metrics, respectively. ROUGE-L is ROUGE score using the longest common subsequence. Distinct unigram (dist-1) and bigram (dist-2) measure the degree of diversity among responses. Len is the average length of the generated response. VHUCM-PUE outperforms the other methods for most of the metrics.

We then compute the following metrics, with the results in Table 3:

- *BLEU* (Papineni et al., 2002): We compute the sentence-level BLEU score with the smoothing seven technique (Chen and Cherry, 2014).
- *Embedding based metrics* (Liu et al., 2016): We use three types of the embedding based metrics, *Embedding average*, *Embedding extrema*, and *Embedding greedy matching*. We use pre-trained Google news word embedding (Mikolov et al., 2013) to measure the embedding metrics to avoid dependency between the training data for dialogue response generation and embedding.
- *ROUGE-L* (Lin, 2004): We report three types of ROUGE-L score, *ROUGE-L Rec*, *ROUGE-L Prec*, and *ROUGE-L F* score.
- *Distinct* (Li et al., 2016a): We use *dist-1* and *dist-2* that refer to the diversity of generated responses.
- *Average length*: We examine the average length of the generated responses to show its diversity.

**Results** As shown in Table 3, VHUCM-PUE outperforms the other methods on most metrics. SpeakAddr generally shows the lowest performance, reflecting that this model does not explicitly model the conversation context. ROUGE-L Prec on SpeakAddr shows higher values than the others because it generates shorter responses on average. VHCR performs better than HRED and VHRED and performs similarly to DialogWAE. VHUCM outperforms VHCR, confirming the effectiveness of the user embeddings. Finally, VHUCM-PUE outperforms all other methods for most of the metrics, showing the effectiveness of the conversation network in initializing the user embeddings.

### 5.3 Human Evaluation

| | Wins | Losses | Ties |
|---|---|---|---|
| vs SpeakAddr | **40.7 ± 3.0** | 34.9 ± 3.0 | 24.4 ± 2.4 |
| vs VHCR | **45.5 ± 2.7** | 40.4 ± 2.8 | 14.1 ± 2.1 |
| vs DialogWAE | **52.5 ± 2.9** | 34.5 ± 2.8 | 13.0 ± 2.1 |

Table 4: Human evaluation of the appropriateness of the generated response. We ask MTurkers to pick a more appropriate response from two candidates generated by different models. Five annotators answered each task. We compute the mean preferences with a 90% confidence interval. VHUCM-PUE outperforms the baselines.

We also evaluate the generated responses by human judgment using Amazon Mechanical Turk (MTurk). First, we sample 150 dyads randomly and build the tasks. We select three conversations of the dyad from the training data and one conversation from the test data randomly for each task. During the task, we show the three conversations initially, after which we show the three-turn utterances of a dyad from the test data. We show the two candidate responses from different models and ask the Mturk annotators which is more ap-

propriate. One of the candidates is from VHUCM-PUE, and the other is from one of the baselines (SpeakAddr, VHCR, and DialogWAE). The annotators can use the answer 'Tie' if they cannot readily select an answer. For each task, we take five annotations and compute the mean preference with a 90% confidence interval. Table 4 shows the results, in which VHUCM-PUE competes with three baselines and outperforms all of them.

Table 5 shows an example of the generated responses from SpeakAddr, VHCR, DialogWAE, and VHUCM-PUE for the same three-turn context. Overall, VHUCM-PUE generates more appropriate responses given the context. SpeakAddr makes general responses without the context. VHCR and DialogWAE create more context related responses than SpeakAddr. But, VHUCM-PUE generates more sophisticated responses than other baselines.

### 5.4 Personalized Responses

We test for consistent responses of personal information, such as age when the user is fixed. We devise personal information questions and generate the responses between the dyads. Table 6 shows examples of these questions and responses.

For the question, the answers of "where is your hometown?" from A are identical even when the questioners differ, because A reveals the hometown in the training data. VHUCM-PUE can answer the user-specific context with learned user vector. In the opposite cases when the questioner is A, the responses are based on the answerer.

The answers of A to the question about age are entirely consistent with the experience question. Moreover, the answer of A for question from B is interesting. From the other answers, we know that age of A is 19, and B's age is 18. The answer to the question is correct even it does not generate the exact age number.

To see the relationship between a dyad, we create the third question that asks about the partner, "Do you love me?". The generated responses show that they match the same feeling on each other. For example, the generated response of A & B and A & C dyads are agreed with the question, 'I love you'. The response of A & D dyad are also matched, but negation of the question. We find the reason that A & B and A & C dyads use emotional words and emoticons. But, A & D dyad have question and answers about the computer-related topics perfunctorily. It shows that VHUCM-PUE can learn the relationship of dyads from their conversation word patterns.

Another interesting outcome is that all responses of C contain ':)' and 'xx' words. User C usually uses the words at the end of the tweets in the training data. It also shows that VHUCM-PUE can learn the word preferences of users.

## 6 Experiment 2 - New Users and Dyads

In this section, we investigate the new user problem when generating responses. We simulate various scenarios of new dyads and new users, and show how VHUCM-PUE incorporates the network information to perform better even for an unknown user.

### 6.1 Experiment Setup

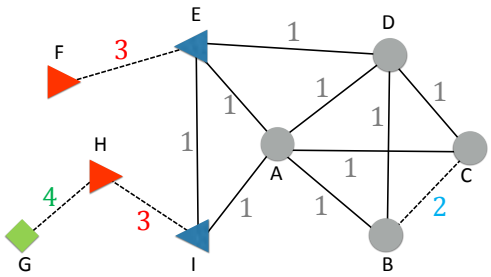


Figure 2: Example of a conversation network. Nodes are users, and a edge between two nodes forms a line if they have conversations. To create the new users and dyads in the test time, we randomly select dyads and remove all conversations between them in the training data (the dashed line in the figure). F, G, and H are new users, B-C dyad is new dyad since they are not in the training data.

We first sort the conversations of a dyad in the order of time, and split the data into training, validation and test data in the same way as in experiment 1. To create new users and dyads, we arbitrarily pick dyads and remove all conversations of those dyads from the training and the validation data. The conversations of these dyads are still in the test data, so we can treat the users in these dyads as new users. Figure 2 shows an example of the conversation network. There are nine users in the figure, and they are connected if they have conversations in the original data. Then, we choose dyads randomly (say B-C, E-F, G-H, and H-I), and remove all conversations of these dyads in the training and validation data. We define four types of dyads with respect to new and known users.

| Context | SpeakAddr | VHCR | DialogWAE | VHUCM-PUE |
|---|---|---|---|---|
| nice ! atl a few months ago and already back again :) how could you not be .. im so excited for the both of them . → whose getting married ? ? → snarky and suzi . got engaged back in dec | i know , i know ! ! ! | hey | ah ha ha , i have to do a lot of time to get to see what happens . | what about u |
| call me when your at the station , i shall be with → am waiting at station 2 caramel macchiato and 2 pain aux raisin in hand ! ! ! x → whoop whoop , , , starting to feel sick | i know , i know ! ! | no . i don't even know what to do , but i don't think so . :( | i know what the hell is , but the is . | :( oh . |
| how was your exam ? x → think it was okay actually :) x → thats good then :) x | you ! | :d :d :d :d :d :d :d i miss you too <3 <3 <3 | hahah ! ! ! im so sorry to tell you when I get home or something lol | you are still the best person who's there ? x |

Table 5: Examples of generated responses from SpeakAddr, VHCR, DialogWAE, and VHUCM-PUE. The → means a changes of turn in the context.

| Questioner | Answerer | Questions | | |
|---|---|---|---|---|
| | | Where is your hometown ? | How old are you ? | Do you love me ? |
| User B | User A | north carolina ! | i'm not sure , but i am a bit older than you | i love you . |
| User C | User A | north carolina . | 19 ! ! ! | yes i do ! |
| User D | User A | north carolina . | i'm 19 . i don't even know what to say | no i do not |
| User A | User B | minnesota . <unk>. | 18 yr old | because i love you . |
| User A | User C | manchester :) xx | nothing much :) | i love you too :) xx |
| User A | User D | i live in <unk>. | i have no idea | no , i don't . |

Table 6: Responses to users' personal information questions from VHUCM-PUE. The questioner asks each question to the answerer, and VHUCM-PUE generates the answerers' responses. The '<unk>' token is an unknown word. VHUCM-PUE generates personalized responses of users.

1. *Known Dyad*: Conversations of the dyad exist in the training and validation data. Examples include A-B, A-D, and E-I.

2. *Known Users*: There are no conversations of the dyad in the training and validation data, but the two users' conversations with other users exist in the training data. An example is B-C.

3. *Known Partner*: Similar to *Known Users*, except there are no conversations of one of the speakers ("new user") in the training data. But the other partner ("known partner") has conversations with other users in the training corpus, so the model can see the partner's conversations during training. Examples are F-E and H-I where F and H are new users, and E and I are known partners.

4. *New Users*: Both users do not have any conversations in the training and validation data. They are new users. An example is G.

For this experiment, we create a small but dense conversation network from our corpus. As in section 3, we filter out dyads with fewer than ten conversations, but here we filter users who have fewer than five conversation friends. The number of users is 2,187, the number of dyads is 3,833, and the number of conversations is 84,295. We randomly select 20% of the dyads and remove all conversations of the dyads in the training and validation data. We conduct the experiment six times with different random seeds.

### 6.2 New User Embedding

We assume that two users are close in the user embedding space when they have conversations since people try to minimize the social difference between the others when they have conversations

| | *Dyads* | *NonDyads* |
|---|---|---|
| VHUCM | $2.449 \pm 0.004$ | $2.453 \pm 0.001$ |
| VHUCM-PUE | $2.879 \pm 0.003$ | $5.444 \pm 0.001$ |

Table 7: Distance between users in the user embedding spaces from VHUCM or VHUCM-PUE. We make two groups of user pairs; pairs who have conversations (*Dyads*), and pairs who do not have any conversations (*NonDyads*). We compute the mean value of the Euclidean distance between each pair in the groups with a 90% confidence interval. VHUCM-PUE distinguishes the two groups significantly better than VHUCM.

(Linell et al., 1991). We investigate how VHUCM and VHUCM-PUE learn this assumption well. We make two groups of user pairs whether they have conversations or not. *Dyads* are user pairs who have conversations, and *NonDyads* are user pairs who do not. And, we compute the Euclidean distance of user embedding between each pair in a group.

Table 7 shows the results. The two groups are statistically significantly different in VHUCM-PUE user embedding, but not in VHUCM. This result shows that VHUCM-PUE learns the assumption better than VHUCM, and this is one reason why VHUCM-PUE outperforms VHUCM.

Accordingly, we can set the vector of a new user to be near the conversation partner. Formally, we compute the average of the conversation partners of the new user and add some random noise to the new user embedding. For example, F is the new user and E is the known partner in Figure 2. We utilize the embedding of F from conversation partner E as follows:

$$\boldsymbol{h}_F^{user} = \sum_{i \in \text{friends of F}} \boldsymbol{h}_i^{user} + \epsilon$$

$$\epsilon \sim \text{Uniform}(u_{min}, u_{max})$$

where $u_{min}$ and $u_{max}$ are min and max values over each dimension of the trained user vectors, and the friends of F $= \{E\}$. We incorporate this method in VHUCM-PUE for new users in the *Known Partner* scenario.

In the *New Users* case, we cannot get the user information during the test. So, we set their embeddings as random values. For example, G is a new user in the *New Users*, thus we set the user embedding values from the uniform distribution over the trained user embedding space as $\boldsymbol{h}_G^{user} \sim \text{Uniform}(u_{min}, u_{max})$. We incorporate this method in VHUCM for all new users and VHUCM-PUE for users who are in the *New Users*.

### 6.3 Results and Discussion

Figure 3 shows the response quality test results for all cases. VHUCM and VHUCM-PUE outperform VHCR in the *Known Dyad* case which is the same situation in section 5. In the *Known Users* case, VHUCM and VHUCM-PUE performs better than VHCR. Although the conversations of the dyad are not in the training data, VHUCM can learn the users from conversations with other partners, and this helps to infer the responses between them.

In the *Known Partner* case, VHUCM-PUE outperforms VHUCM and VHCR. To see the reason of this improvement, we investigate the difference of user embeddings between the models. We run VHUCM-PUE with full data which are not removed the conversations in the training data. Figure 4a shows the two-dimensional plot of the user embedding by projecting t-SNE (Maaten and Hinton, 2008). The new user (▶) and known partner (◀) pairs are closed each other. But the closeness between the dyad is not observed in VHUCM with removed data since it creates the new user embedding randomly (Fig 4b). The new user and known partner dyads are closed in VHUCM-PUE which applies the method in section 6.2 to new users in the space (Fig 4c). This results show the reason why VHUCM-PUE has a better performance in *Known Partner* case.

Finally, *New Users* shows an overlap with regard to the performances of all models. No information about new users exists in the training data and test data; hence VHUCM-PUE is forced to initialize the new users randomly, similarly to VHUCM.

## 7 Conclusion

In this paper, we made large long-term conversation corpus from Twitter. We have presented VHUCM to generate the response given the prior utterances of a conversation and user. To initialize the users to VHUCM, we pre-train the user embedding from the conversation network. We showed that VHUCM-PUE outperforms others in most metrics. We also suggested the way to incorporate new users who have conversations in test time with trained users. We showed that using learned partners' embedding helps to generate better responses for the new users and dyads.

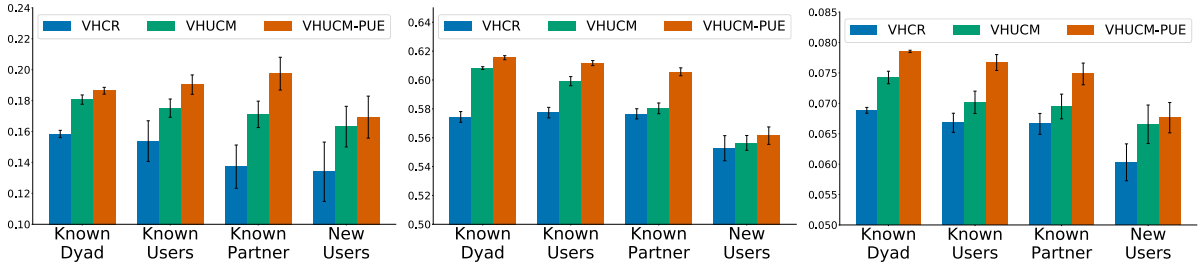


(a) BLEU (b) Embedding Avg (c) ROUGE-L F

Figure 3: Response quality test on four cases of new users and dyads. The range of the vertical error bar is one standard error of the metric values among the experimental trials. VHUCM-PUE outperforms the other models in cases involving new user with a known user whose conversations are in the training data (*Known Partner* case).

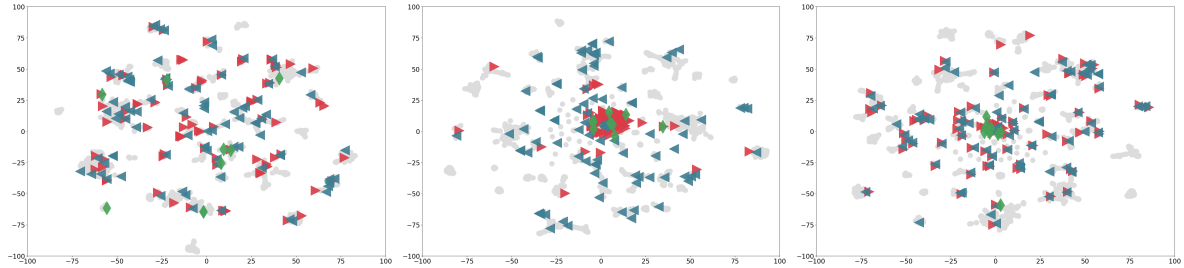

(a) VHUCM-PUE with full data (b) VHUCM with removed data (c) VHUCM-PUE with removed data

Figure 4: Plots of two-dimensions projection of trained user vectors by each model and data by t-SNE. Four types of users: Known Users (●) who are in the *Known Dyad* and *Known Users* cases, New Users (▶) and Known Partners (◀) who are in the *Known Partner* case, and New Users (◆) who are in the *New Users* case. (a) The conversation partners (▶ and ◀) are closed in the embedding space. (b) Many new users (▶ and ◆) are not well distributed. (c) The conversation partners (▶ and ◀) are well paired as similar as (a).

## Acknowledgments

We would like to thank the anonymous reviewers for helpful questions and comments. This work was supported by IITP grant funded by the Korea government (MSIT) (No.2017-0-01779, XAI).

## References

JinYeong Bak, Chin-Yew Lin, and Alice Oh. 2014. Self-disclosure topic model for classifying and analyzing twitter conversations. In *Proceedings of the EMNLP*.

Parminder Bhatia, Marsal Gavaldà, and Arash Einolghozati. 2017. soc2seq: Social embedding meets conversation model. *arXiv preprint*.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of the CoNLL*.

Boxing Chen and Colin Cherry. 2014. A systematic comparison of smoothing techniques for sentence-level bleu. In *Proceedings of the WMT*.

Cristian Danescu-Niculescu-Mizil and Lillian Lee. 2011. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In *Proceedings of the CMCL*.

Jiachen Du, Wenjie Li, Yulan He, Ruifeng Xu, Lidong Bing, and Xuan Wang. 2018. Variational autoregressive decoder for neural response generation. In *Proceedings of the EMNLP*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Proceedings of the NIPS*.

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the SIGKDD*.

Xiaodong Gu, Kyunghyun Cho, Jung-Woo Ha, and Sunghun Kim. 2019. DialogWAE: Multimodal response generation with conditional wasserstein auto-encoder. In *Proceedings of the ICLR*.

Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. 2014. Semi-supervised learning with deep generative models. In *Proceedings of the NIPS*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting ob-

jective function for neural conversation models. In *Proceedings of the NAACL*.

Jiwei Li, Michel Galley, Chris Brockett, Georgios Spithourakis, Jianfeng Gao, and Bill Dolan. 2016b. A persona-based neural conversation model. In *Proceedings of the ACL*.

Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. 2017a. Adversarial learning for neural dialogue generation. In *Proceedings of the EMNLP*.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017b. Dailydialog: A manually labelled multi-turn dialogue dataset. In *Proceedings of the IJCNLP*.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*.

Per Linell, Antony Manstead, et al. 1991. *Contexts of Accommodation: Developments in Applied Sociolinguistics*. Cambridge University Press.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the EMNLP*.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *Proceedings of the SIGDIAL*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*.

Pierre-Emmanuel Mazare, Samuel Humeau, Martin Raison, and Antoine Bordes. 2018. Training millions of personalized dialogue agents. In *Proceedings of the EMNLP*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the NIPS*.

O. O. Olabiyi, A. Khazane, and E. T. Mueller. 2018. A persona-based multi-turn conversation model in an adversarial learning framework. In *Proceedings of the ICMLA*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the ACL*.

Yookoon Park, Jaemin Cho, and Gunhee Kim. 2018. A hierarchical latent structure for variational conversation modeling. In *Proceedings of the NAACL*.

Adwait Ratnaparkhi. 2002. Trainable approaches to surface natural language generation and their application to conversational dialog systems. *Computer Speech & Language*.

Alan Ritter, Colin Cherry, and William B. Dolan. 2011. Data-driven response generation in social media. In *Proceedings of the EMNLP*.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Proceedings of the AAAI*.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the CIKM*.

Yujie Xing and Raquel Fernández. 2018. Automatic evaluation of neural personality-based chatbots. In *Proceedings of the INLG*.

Zhen Xu, Bingquan Liu, Baoxun Wang, Chengjie SUN, Xiaolong Wang, Zhuoran Wang, and Chao Qi. 2017. Neural response generation via gan with an approximate embedding layer. In *Proceedings of the EMNLP*.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the ACL*.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Proceedings of the ACL*.

# Recommendation as a Communication Game: Self-Supervised Bot-Play for Goal-oriented Dialogue

**Dongyeop Kang**[♡] **Anusha Balakrishnan**[♣] **Pararth Shah**[♣]
**Paul Crook**[♣] **Y-Lan Boureau**[♣] **Jason Weston**[♣]
[♡]Carnegie Mellon University, [♣]Facebook AI
dongyeok@cs.cmu.edu anusha.bala28@gmail.com
{pararths,pacrook,ylan,jase}@fb.com

## Abstract

Traditional recommendation systems produce static rather than interactive recommendations invariant to a user's specific requests, clarifications, or current mood, and can suffer from the cold-start problem if their tastes are unknown. These issues can be alleviated by treating recommendation as an interactive dialogue task instead, where an expert recommender can sequentially ask about someone's preferences, react to their requests, and recommend more appropriate items. In this work, we collect a goal-driven recommendation dialogue dataset (GoRecDial), which consists of 9,125 dialogue games and 81,260 conversation turns between pairs of human workers recommending movies to each other. The task is specifically designed as a cooperative game between two players working towards a quantifiable common goal. We leverage the dataset to develop an end-to-end dialogue system that can simultaneously converse and recommend. Models are first trained to imitate the behavior of human players without considering the task goal itself (supervised training). We then fine-tune our models on simulated bot-bot conversations between two paired pre-trained models (bot-play), in order to achieve the dialogue goal. Our experiments show that models fine-tuned with bot-play learn improved dialogue strategies, reach the dialogue goal more often when paired with a human, and are rated as more consistent by humans compared to models trained without bot-play. The dataset and code are publicly available through the ParlAI framework[1].

## 1 Introduction

Traditional recommendation systems factorize users' historical data (i.e., ratings on movies) to extract common preference patterns (Koren et al., 2009; He et al., 2017b). However, besides making it difficult to accommodate new users because of the *cold-start problem*, relying on aggregated history makes these systems static, and prevents users from making specific requests, or exploring a temporary interest. For example, a user who usually likes horror movies, but is in the mood for a fantasy movie, has no way to indicate their preference to the system, and would likely get a recommendation that is not useful. Further, they cannot iterate upon initial recommendations with clarifications or modified requests, all of which are best specified in natural language.

Recommending through dialogue interactions (Reschke et al., 2013; Wärnestål, 2005) offers a promising solution to these problems, and recent work by Li et al. (2018) explores this approach in detail. However, the dataset introduced in that work does not capture higher-level strategic behaviors that can impact the quality of the recommendation made (for example, it may be better to elicit user preferences first, before making a recommendation). This makes it difficult for models trained on this data to learn optimal recommendation strategies. Additionally, the recommendations are not grounded in real observed movie preferences, which may make trained models less consistent with actual users. This paper aims to provide *goal-driven recommendation dialogues grounded in real-world data*. We collect a corpus of goal-driven dialogues grounded in real user movie preferences through a carefully designed gamified setup (see Figure 1) and show that models trained with that corpus can learn a successful recommendation dialogue strategy. The training is conducted in two stages: first, a *supervised* phase that trains the model to mimic human behavior on the task; second, a *bot-play* phase that improves the goal-directed strategy of the model.

The contribution of this work is thus twofold.

[1] https://github.com/facebookresearch/ParlAI



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1951–1961,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

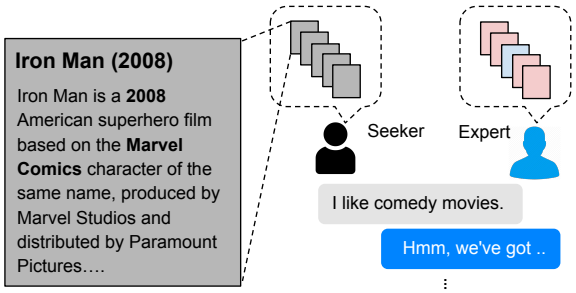


Figure 1: Recommendation as a dialogue game. We collect 81,260 recommendation utterances between pairs of human players (experts and seekers) with a collaborative goal: the expert must recommend the correct (blue) movie, avoiding incorrect (red) ones, and the seeker must accept it. A chatbot is then trained to play the expert in the game.

(1) We provide the first (to the best of our knowledge) large-scale goal-driven recommendation dialogue dataset with specific goals and reward signals, grounded in a real-world knowledge base. (2) We propose a two-stage recommendation strategy learning framework and empirically validate that it leads to better recommendation conversation strategies.

## 2 Recommendation Dialogue Task Design

In this section, we first describe the motivation and design of the dialogue-based recommendation game that we created. We then describe the data collection environment and present detailed dataset statistics.

### 2.1 Dialogue Game: Expert and Seeker

The game is set up as a conversation between a *seeker* looking for a movie recommendation, and an *expert* recommending movies to the seeker. Figure 2 shows an example movie recommendation dialogue between two-paired human workers on Amazon Mechanical Turk.

**Game Setting.** Each worker is given a set of five movies[2] with a description (first paragraph from the Wikipedia page for the movie) including important features such as director's name, year, and genre. The seeker's set represents their watching history (movies they are supposed to have liked) for the game's sake. The expert's set consists of candidate movies to choose from when making recommendations, among which only one is the *correct* movie to recommend. The correct movie is chosen to be similar to the seeker's movie set (see Sec. 2.2), while the other four movies are dissimilar. The expert is not told by the system which of the five movies is the correct one. The expert's goal is to find the correct movie by chatting with the seeker and recommend it after a minimal number of dialogue turns. The seeker's goal is to accept or reject the recommendation from the expert based on whether they judge it to be similar to their set. The game ends when the expert has recommended the correct movie. The system then asks each player to rate the other for engagingness.

**Justification.** Players are asked to provide reasons for recommending, accepting, or rejecting a movie, so as to get insight into human recommendation strategies[3].

**Gamification.** Rewards and penalties are provided to players according to their decisions, to make the task more engaging and incentivize better strategies. Bonus money is given if the expert recommends the correct movie, or if the seeker accepts the correct movie or rejects an incorrect one.

### 2.2 Picking Expert and Seeker movie sets

This section describes how movie sets are selected for experts and seekers.

**Pool of movies** To reflect movie preferences of real users, our dataset uses the MovieLens dataset[4], comprising 27M ratings applied to 58K movies by 280K real users. We obtain descriptive text for each movie from Wikipedia[5] (i.e., the first paragraph). We also extract entity-level features (e.g., directors, actors, year) using the MovieWiki dataset (Miller et al., 2016) (See Figure 1). We filter out less frequent movies and user profiles (see Appendix), resulting in a set of 5,330 movies and 65,181 user profiles with their ratings.

**Movie similarity metric** In order to simulate a natural setting, the movies in the seeker's set

[2]We deliberately restricted the set of movies to make the task more tractable. One may argue that the expert can simply ask these candidates one by one (at the cost of low engagingness). However, this empirically doesn't happen: experts make on average only 1.16 incorrect movie recommendations.

[3]Our model doesn't utilize this or the engagingness scores for learning, but these are potential future directions.

[4]https://grouplens.org/datasets/movielens/

[5]https://dumps.wikimedia.org/

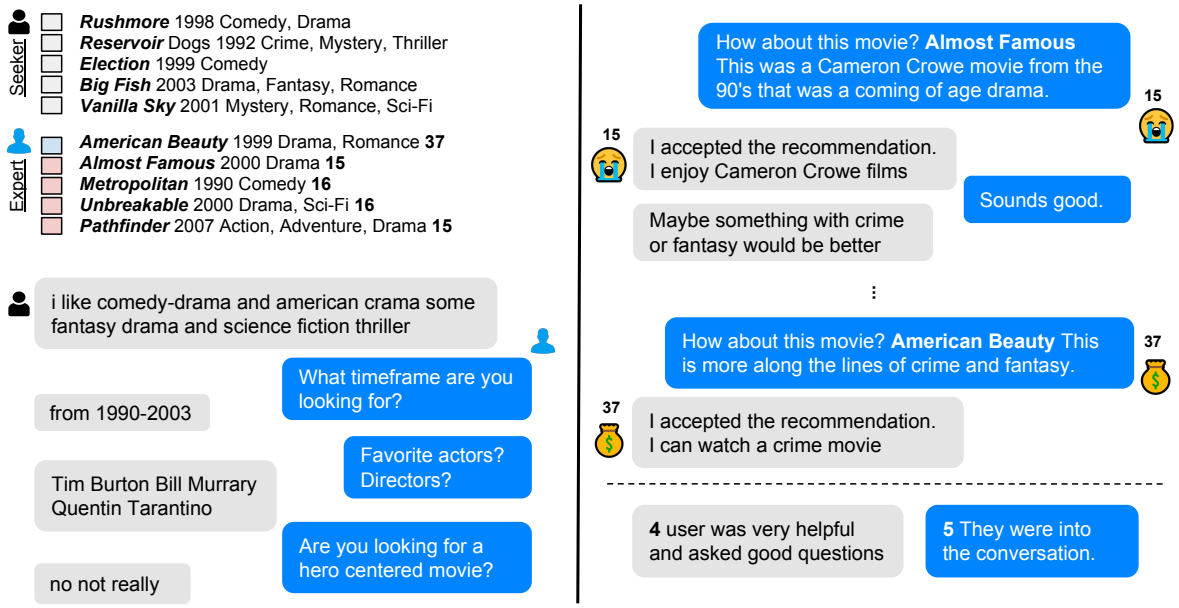


Figure 2: An example dialogue from our dataset of movie recommendation between two human workers: seeker (grey) and expert (blue). The goal is for the expert to find and recommend the correct movie (light blue) out of incorrect movies (light red) which is similar to the seeker movies. Best viewed in color.

should be similar to each other, and the correct movie should be similar to these, according to a metric that reflects coherent empirical preferences. To compute such a metric, we train an embedding-driven recommendation model (Wu et al., 2018).[6] Each movie is represented as an embedding, which is trained so that embeddings of movies watched by the same user are close to each other. The closeness metric between two movies is the cosine similarity of these trained embeddings. A movie is deemed close to a set of movies if its embedding is similar to the average of the movie embeddings in the set.

**Movie Set Selection** Using these trained embeddings, we design seeker and expert sets based on the following criteria (See Figure 3):

- Seeker movies (grey) are a set of five movies which are close to each other, chosen from the set of all movies watched by a real user.
- The correct movie (light blue) is close to the average of the five embeddings of the seeker set.
- The expert's incorrect movies (light red) are far from the seeker set and the correct movie.

We filter out movie sets that are too difficult or easy for the recommendation task (see Appendix), and choose 10,000 pairs of seeker-expert movie sets at random.

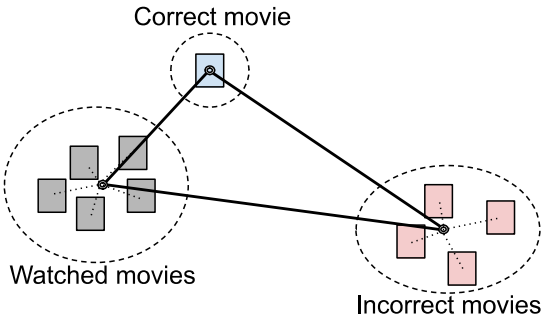


Figure 3: Movie set selection: watched movies for seeker (grey) and correct (light blue) / incorrect (light red) movies for expert.

## 2.3 Data Collection

For each dialogue game, a movie set is randomly chosen without duplication. We collect dialogues using ParlAI (Miller et al., 2017) to interface with Amazon Mechanical Turk. More details about data collection are included in the Appendix.

Table 1 shows detailed statistics of our dataset regarding the movie sets, the annotated dialogues, actions made by expert and seeker, dialogue

[6]We also tried a classical matrix-factorization based recommendation model, which shows comparable performance to the embedding model.

| Dialogue statistics | |
|---|---|
| Number of dialogues | 9,125 |
| Number of utterances | 170,904 |
| Number of unique utterances | 85,208 |
| Avg length of a dialogue | 23.0 |
| Avg duration (minutes) of a dialogue | 5.2 |
| **Expert's utterance statistics** | |
| Avg utterance length | 8.40 |
| Unique tokens | 11,757 |
| Unique utterances | 40,550 |
| **Seeker's utterance statistics** | |
| Avg utterance length | 8.47 |
| Unique tokens | 10,766 |
| Unique utterances | 45,196 |
| **Action statistics** (all scores are averaged) | |
| # of correct/incorrect recs. by expert | 1.0 / 1.16 |
| # of correct/incorrect decisions by seeker | 1.1 / 1.04 |
| **Game statistics** (all scores are averaged) | |
| min/max movie scores | 12.3 / 46.0 |
| correct/incorrect movies | 39.9 / 15.0 |
| real game score by expert/seeker | 61.3 / 50.8 |
| random game score by expert/seeker | 43.2 / 38.1 |
| **Engagingness statistics** (all scores are averaged) | |
| engagingness score by expert/seeker | 4.3 / 4.4 |
| engagingness scores & feedback collected | 18,308 |

Table 1: Data statistics. "correct/incorrect" in the action stats means that the expert recommends the correct/incorrect movie or the seeker correctly accepts/rejects the movie.

games, and engagingness feedback.

The collected dialogues contain a wide variety of action sequences (recommendations and accept/reject decisions). Experts make an average of 1.16 incorrect recommendations, which indicates a reasonable difficulty level. Only 37.6% of dialogue games end at first recommendation, and 19.0% and 10.8% at second and third recommendations, respectively.

Figure 4 shows histogram distributions of (a) expert's decisions between speaking utterance and recommendation utterance and (b) correct and incorrect recommendations over the normalized turns of dialogue. In (a), recommendations increasingly occur after a sufficient number of speaking utterances. In (b), incorrect recommendations are much more frequent earlier in the dialogue, while the opposite is true later on.

## 3 Our Approach

In order to recommend the right movie in the role of the expert, a model needs to combine several perceptual and decision skills. We propose to conduct learning in two stages (See Figure 5): *supervised multi-aspect learning* and *bot-play*.

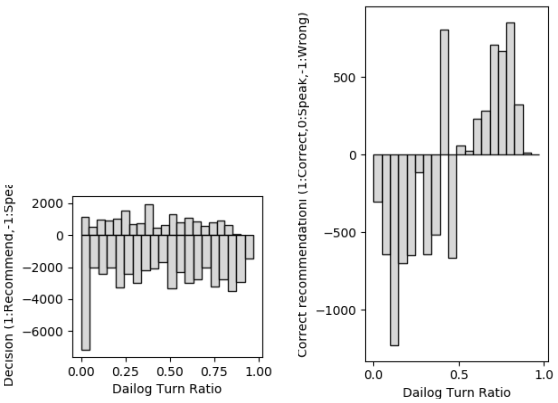


(a) Decision to speak (-1) or recommend (+1)

(b) Correct (+1) or incorrect (-1) recommendations

Figure 4: Histogram distribution of (a) experts' decisions of whether to speak or recommend and (b) correct/incorrect recommendations over the normalized dialogue turns.

### 3.1 Supervised Multi-Aspect Learning

The supervised stage of training the expert model combines three sources of supervision, corresponding to the three following subtasks: (1) GENERATE dialogue utterances to speak with the seeker in a way that matches the utterances of the human speaker, (2) PREDICT the correct movie based on the dialogue history and the movie description representations, and (3) DECIDE whether to recommend or speak in a way that matches the observed decision of the human expert.

Using an LSTM-based model (Hochreiter and Schmidhuber, 1997), we represent the dialogue history context $h_t$ of utterances $x_1$ to $x_t$ as the average of LSTM representations of $x_1, \cdots, x_t$, and the description $m_k$ of the $k$-th movie as the average of the bag-of-word representations[7] of its description sentences. Let $(x_{t+1}, y, d_{t+1})$ denote the ground truth next utterance, correct movie index, and ground truth decision at time $t+1$, respectively. We cast the supervised problem as an end-to-end optimization of the following loss:

$$\mathcal{L}_{sup} = \alpha\mathcal{L}_{gen} + \beta\mathcal{L}_{predict} + (1-\alpha-\beta)\mathcal{L}_{decide}, \quad (1)$$

where $\alpha$ and $\beta$ are weight hyperparameters optimized over the validation set, and $\mathcal{L}_{predict}, \mathcal{L}_{decide}, \mathcal{L}_{gen}$ are negative log-likelihoods

[7] We empirically found that BOW works better than other encoders such as LSTM in this case.

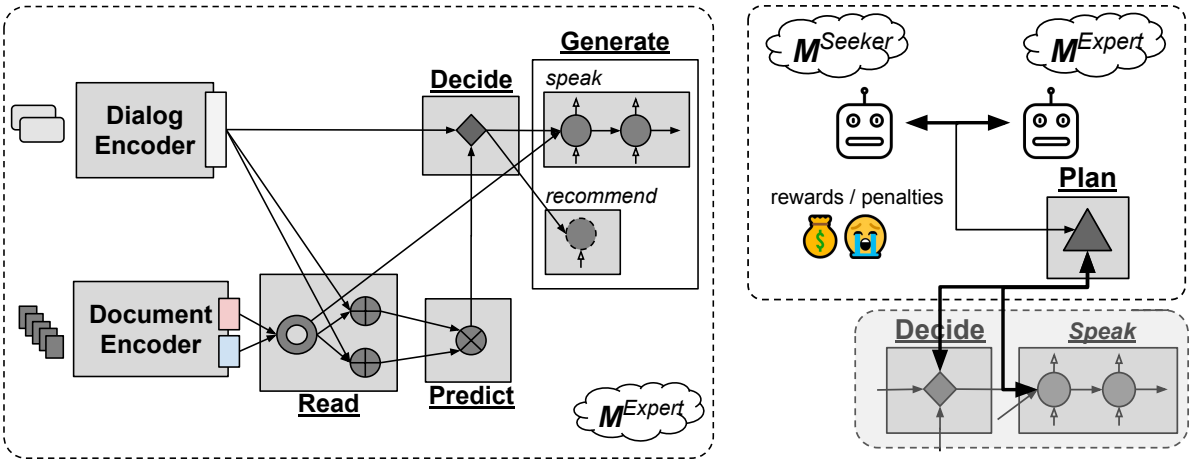


(a) Supervised multi-aspect learning

(b) Bot-play

Figure 5: (a) Supervised learning of the expert model $\mathcal{M}^{expert}$ and (b) bot-play game between the expert $\mathcal{M}^{expert}$ and the seeker $\mathcal{M}^{seeker}$ models. The former imitates multiple aspects of humans' behaviors in the task, while the later fine-tunes the expert model w.r.t the game goal (i.e., recommending the correct movie).

of probability distributions matching each of the three subtasks:

$$\mathcal{L}_{gen} = -\log p_{gen}(x_{t+1}|h_t, m_1, \cdots, m_K), \quad (2)$$

$$\mathcal{L}_{predict} = -\log p(y|c_1, \cdots, c_K), \quad \text{where} \quad (3)$$

$$c_j = h_t \cdot m_j \qquad \text{for } j \in 1..K, \quad (4)$$

$$\mathcal{L}_{decide} = p_{MLP}(d_{t+1}|h_t, c_1, \cdots, c_K), \quad (5)$$

with $p_{gen}$ the output distribution of an attentive seq2seq generative model (Bahdanau et al., 2015), $p$ a softmax distribution over dot products $h_t \cdot m_k$ that capture how aligned the dialogue history $h_t$ is with the description $m_k$ of the $k$-th movie, and $p_{MLP}$ the output distribution of a multi-layer perceptron predictor that takes $c_1, \cdots, c_K$ as inputs[8].

### 3.2 Bot-Play

Motivated by the recent success of self-play in strategic games (Silver et al., 2017; Vinyals et al., 2019; OpenAI, 2018) and in negotiation dialogues (Lewis et al., 2017), we show in this section how we construct a reward function to perform bot-play between two bots in our setting, with the aim of developing a better expert dialogue agent for recommendation.

**Plan** optimizes long-term policies of the various aspects over multiple turns of the dialogue game by maximizing game-specific rewards. We first pre-train expert and seeker models individually: the expert model $\mathcal{M}^{expert}(\theta) = \min_\theta \mathcal{L}_{sup}$ is pre-trained by minimizing the supervised loss in Eq 1, and the seeker model $\mathcal{M}^{seeker}(\phi)$ is a retrieval-based model that retrieves seeker utterances from the training set based on cosine similarity of the preceding dialogue contexts encoded using the BERT pre-trained encoder[9]. $\theta$ and $\phi$ are model parameters of the expert and seeker model, respectively. Then, we make them chat with each other, and fine-tune the expert model by maximizing its reward in the game (See Figure 5, Right).

The dialogue game ends if the expert model recommends the correct movie, or a maximum dialogue length is reached[10], yielding $T$ turns of dialogue; $g = (x_1^{expert}, x_1^{seeker} .. x_T^{expert}, x_T^{seeker})$. Let $T_{REC}$ the set of turns when the expert made a recommendation. We define the expert's reward as:

$$r_t^{expert} = \frac{1}{|T_{REC}|} \cdot \sum_{t \in T_{REC}} \delta^{t-1} \cdot b_t, \quad (6)$$

where $\delta$ is a discount factor[11] to encourage earlier recommendations, $b_t$ is the reward obtained at each recommendation made, and $|T_{REC}|$ is the number of recommendations made. $b_t$ is 0 unless the correct movie was recommended.

[8] We experimented with various other encoding functions, detailed in the Appendix.

[9] See Sec. 4.2 for details on BERT. We also experimented with sequence-to-sequence models for modeling the seeker but performance was much worse.

[10] We restrict the maximum length of a dialogue to 20.

[11] we use $\delta = 0.5$.

We define the reward function $\mathcal{R}$ as follows:

$$\mathcal{R}(x_t) = \sum_{x_t \in X^{expert}} \gamma^{T-t}(r_t^{expert} - \mu) \quad (7)$$

where $\mu = \frac{1}{t}\sum_{1..t} r_t^{expert}$ is the average of the rewards received by the expert until time $t$ and $\gamma$ is a discount factor to diminish the reward of earlier actions. We optimize the expected reward for each turn of dialogue $x_t$ and calculate its gradient using REINFORCE (Williams, 1992). The final role-playing objective $\mathcal{L}_{RP}$ is:

$$\nabla \mathcal{L}_{RP}(\theta; z) = \sum_{x_t \in X^{expert}} \mathbb{E}_{x_t}[\nabla \log p(x_t|x_{<t})\mathcal{R}(x_t)] \quad (8)$$

We optimize the role-playing objective with the pre-trained expert model's decision ($\mathcal{L}_{decide}$) and generation ($\mathcal{L}_{gen}$) objectives at the same time. To control the variance of the RL loss, we alternate optimizing the RL loss and other two supervised losses for each step. We do not fine-tune the prediction loss, in order not to degrade the prediction performance during bot-play.

## 4 Experiments

We describe our experimental setup in §4.1. We then evaluate our supervised and unsupervised models in §4.2 and §4.3, respectively.

### 4.1 Setup

We select 5% of the training corpus as validation set in our training.

All hyper-parameters are chosen by sweeping different combinations and choosing the ones that perform best on the validation set. In the following, the values used for the sweep are given in brackets. Tokens of textual inputs are lower-cased and tokenized using byte-pair-encoding (BPE) (Sennrich et al., 2016) or the Spacy[12] tokenizer. The seq-to-seq model uses 300-dimensional word embeddings initialized with GloVe (Pennington et al., 2014) or Fasttext (Joulin et al., 2017) embeddings, $[1, 2]$ layers of $[256, 512]$-dimensional Uni/Bi-directional LSTMs (Hochreiter and Schmidhuber, 1997) with 0.1 dropout ratio, and soft attention (Bahdanau et al., 2015). At decoding, we use beam search with a beam of size 3, and choose the maximum likelihood output. For each turn, the initial movie text and all previous dialogue turns including seeker's and expert's replies are concatenated as input to the models.

Both supervised and bot-play learning use Adam (Kingma and Ba, 2015) optimizer with batch size 32 and learning rates of $[0.1, 0.01, 0.001]$ with 0.1 gradient clipping. The number of softmax layers (Yang et al., 2018) is $[1, 2]$. For each turn, the initial movie description and all previous dialogue utterances from the seeker and the expert are concatenated as input text to the other modules. Each movie textual description is truncated at 50 words for efficient memory computation.

We use annealing to balance the different supervised objectives: we only optimize the GENERATE loss for the first 5 epochs, and then gradually increase weights for the PREDICT and DECIDE losses. We use the same movie-sets as in the supervised phase to fine-tune the expert model. Our models are implemented using PyTorch and ParlAI (Miller et al., 2017). Code and dataset will be made publicly available through ParlAI[13].

### 4.2 Evaluation of Supervised Models

**Metrics.** We first evaluate our supervised models on the three supervised tasks: dialogue generation, movie recommendation, and per-turn decision to speak or recommend. The dialogue generation is evaluated using the F1 score and BLEU (Papineni et al., 2002) comparing the predicted and ground-truth utterances. The F1 score is computed at token-level. The recommendation model is evaluated by calculating the percentage of times the correct movie is among the top k recommendations (hit@k). In order to see the usefulness of dialogue for recommendation, precision is measured per each expert turn of the dialogue (`Turn@k`) regardless of the decision to speak or recommend, and at the end of the dialogue (`Chat@k`).

**Models.** We compare our models with Information Retrieval (IR) based models and recommendation-only models. The IR models retrieve the most relevant utterances from the set of candidate responses of the training data and rank them by comparing cosine similarities using TFIDF features or BERT (Devlin et al., 2019) encoder features. Note that IR models make no recommendation. The recommendation-only models

[12] https://spacy.io/

[13] https://github.com/facebookresearch/ParlAI

| | | Generation | | Recommendation | | | | Decision |
|---|---|---|---|---|---|---|---|---|
| | | F1 | BLEU | Turn@1 | Turn@3 | Chat@1 | Chat@3 | Acc |
| Baseline | TFIDF-RANKER | 32.5 | **27.8** | - | - | - | - | - |
| | BERT-RANKER | 38.3 | 23.9 | - | - | - | - | |
| | RANDOM RECC. | 3.6 | 0.1 | 21.3 | 59.2 | 23.1 | 62.2 | - |
| | BERT RECC. | 16.5 | 0.2 | 25.5 | 66.3 | 26.4 | 68.3 | - |
| Ours | GENERATE | 39.5 | 26.0 | - | - | - | - | - |
| | +PREDICT | 40.2 | 26.4 | 76.4 | 96.9 | 75.7 | 97.0 | - |
| | +DECIDE | **41.0** | 27.4 | **77.8** | **97.1** | **78.2** | **97.7** | **67.6** |
| | +PLAN | 40.9 | 26.8 | 76.3 | 95.7 | 77.5 | 97.6 | 53.6 |

Table 2: Evaluation on supervised models. We incrementally add different aspects of modules: GENERATE, PREDICT, and DECIDE for supervised multi-aspect learning and PLAN for bot-play fine-tuning.

always produce recommendation utterances following the template (e.g., "how about this movie, [MOVIE]?") where the [MOVIE] is chosen randomly or based on cosine similarities between dialogue contexts and the text descriptions of candidate movies. We use the pre-trained BERT encoder (Devlin et al., 2019) to encode dialogue contexts and movie text descriptions.

We incrementally add each module to our base GENERATE model: PREDICT and DECIDE for supervised learning and PLAN for bot-play fine-tuning. Each model is chosen from the best model in our hyper-parameter sweeping.

**Results.** Table 2 shows performance comparison on the test set. Note that only the full supervised model (+DECIDE) and the fine-tuned model (+PLAN) can appropriately operate every function required of an expert agent such as producing utterances, recommending items, and deciding to speak or recommend.

Compared to recommendation-only models, our prediction PREDICT modules show significant improvements over the recommendation baselines on both per-turn and per-chat recommendations: 52% on Turn@1 and 34% on Turn@3. Chat scores are always higher than Turn, indicating that recommendations get better as more dialogue context is provided. The DECIDE module yields additional improvements over the PREDICT model in both generation and recommendation, with 67.6% decision accuracy, suggesting that the supervised signal of decisions to speak or recommend can contribute to better overall representations.

In generation, our proposed models show comparable performance as the IR baseline models (e.g., BERTRANKER). The +DECIDE model improves on the F1 generation score because it learns when to predict the templated recommendation utterance.

As expected, +PLAN slightly hurts most metrics of supervised evaluation, because it optimizes a different objective (the game objective), which might not systematically align with the supervised metrics. For example, a system optimized to maximize game objective should try to avoid incorrect recommendations even if humans made them. Game-related evaluations are shown in §4.3.

**Analysis** We analyze how each of the supervised modules acts over the dialogue turns on the test set. Figure 6 (a) shows a histogram of the rank of the ground-truth movie over turns. The rank of the model's prediction is very high for the first few turns, then steadily decreases as more utterances are exchanged with the seeker. This indicates that the dialogue context is crucial for finding good recommendations.

The evolution of generation metrics (F1, BLEU) for each turn is shown at the top of Fig. 6(b), and the (accumulated) recommendation and decision metrics (Turn@1/Accuracy) at the bottom[14]. The accumulated recommendation and decision performance sharply rises at the end of the dialogue and variance decreases. The generation performance increases, because longer dialogue contexts helps predict the correct utterances.

[14]For better understanding of the effect of recommendation and decision, we show accumulated values, and per-turn values for generation.

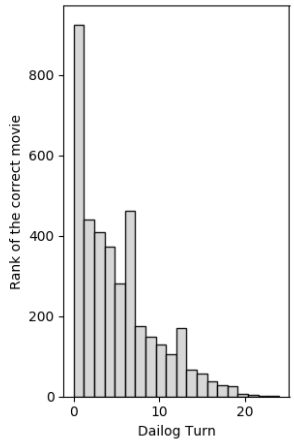


(a) Rank of recommendation

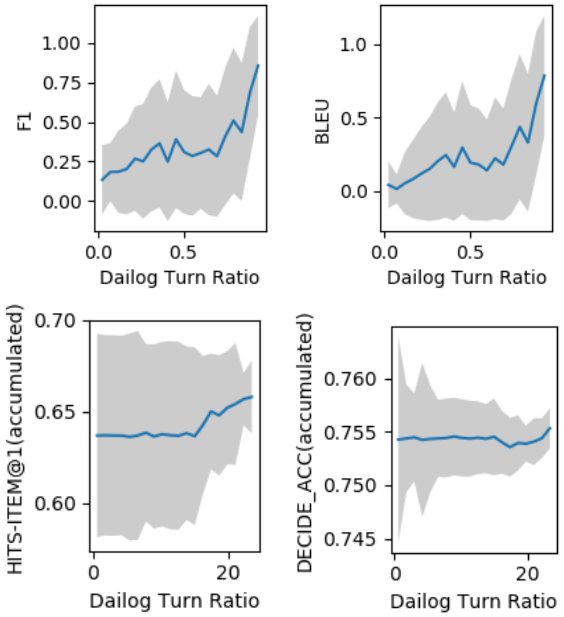


(b) F1/BLEU (top) & Turn@1 /Decision Acc (bottom)

Figure 6: Analysis of the expert's model: as the dialogue continues (x-axis is either fraction of the full dialogue, or index of dialogue turn), y-axis is (a) rank of the correct recommendation (the lower rank, the better) and (b) F1/BLEU/Turn@1/Decision Accuracy (the higher the better) with the variance shown in grey.

### 4.3 Evaluation on Dialogue Games

**Metrics.** In the bot-play setting, we provide game-specific measures as well as human evaluations. We use three automatic game measures: Goal to measure the ratio of dialogue games where the goal is achieved (i.e., recommending the correct movie or not), Score to measure the total game score, and Turn2G to count the number of dialogue turns taken until the goal is achieved.

We conduct human evaluation by making the expert model play with human seekers. We measure automatic metrics as well as dialogue quality scores provided by the player: fluency, consistency, and engagingness (scored between 1 and 5) (Zhang et al., 2018). We use the full test set (i.e., 911 movie sets) for bot-bot games and use 20 random samples from the test set for {bot,human}-human games.

**Models.** We compare our best supervised model with several variants of our fine-tuned bot-play models. We consider bot-play of an expert model with different seeker models such as BERT-Ranker based seeker and Seq-to-Seq based seeker. Each bot-play model is trained on the same train set that is used for training the original supervised model. The seeker model uses retrieval based on BERT pretrained representations of dialogue context (BERT-R) [15].

| Players | | Automatic | | | Human | | |
|---|---|---|---|---|---|---|---|
| **Expert** | **Seeker** | **Goal** | **Sco** | **T2G** | **F** | **C** | **E** |
| Supervised* | BERT-R | 30.9 | 38.3 | 1.4 | - | - | - |
| Bot-play \w S2S | BERT-R | 42.1 | 49.6 | 2.8 | - | - | - |
| Bot-play \w BERT-R | BERT-R | **48.6** | **52.4** | **3.2** | - | - | - |
| Supervised* | Human | 55.0 | 51.2 | 2.1 | 3.1 | 2.2 | **2.0** |
| Bot-play* | Human | **68.5** | **54.7** | **3.1** | **3.2** | **2.6** | **2.0** |
| Human | Human | 95.0 | 64.3 | 8.5 | 4.8 | 4.7 | 4.2 |

Table 3: Evaluation on dialogue recommendation games: bot-bot (top three rows) and {bot,human}-human (bottom three rows). We use automatic game measures (**Goal**, **Score**, **Turn2Goal**) and human quality ratings (**Fluency**, **Consistency**, **Engagingness**).

**Results.** Compared to the supervised model, the self-supervised model fine-tuned by seeker models shows significant improvements in the game-related measures. In particular, the BERT-R model shows a +27.7% improvement in goal success ratio. Interestingly, the number of turns to reach the goal increases from 1.4 to 3.2, indicating that conducting longer dialogues seems to be a better strategy to achieve the game goal throughout our role-playing game.

In dialogue games with human seeker players, the bot-play model also outperforms the supervised one, even though it is still far behind human performance. When the expert bot plays with the human seeker, performance increases compared to

[15] A potential direction for future work may have more solid seeker models and explore which aspect of the model makes the dialogue with the expert model more goal-oriented or human-like.

playing with the bot seeker, because the human seeker produces utterances more relevant to their movie preferences, increasing overall game success.

## 5 Related Work

Recommendation systems often rely on matrix factorization (Koren et al., 2009; He et al., 2017b). Content (Mooney and Roy, 2000) and social relationship features (Ma et al., 2011) have also been used to help with the cold-starting problem of new users. The idea of eliciting users' preference for certain content features through dialogue has led to several works. Wärneståal (2005) studies requirements for developing a conversational recommender system, e.g., accumulation of knowledge about user preferences and database content. Reschke et al. (2013) automatically produces template-based questions from user reviews. However, no conversational recommender systems have been built based on these works due to the lack of a large publicly available corpus of human recommendation behaviors.

Very recently, Li et al. (2018) collected the ReDial dataset, comprising 10K conversations of movie recommendations, and used it to train a generative encoder-decoder dialogue system. In this work, crowdsource workers freely talk about movies and are instructed to make a few movie recommendations before accepting one. Compared to ReDial, our dataset is grounded in real movie preferences (movie ratings from MovieLens), instead of relying on workers' hidden movie tastes. This allows us to make our task goal-directed rather than chit-chat; we can optimize prediction and recommendation strategy based on a known ground truth, and train the PREDICT and PLAN modules of our system. That in turn allows for novel setups such as bot-play.

To the best of our knowledge, Bordes et al. (2016) is the only other goal-oriented dialogue benchmark grounded in a database that has been released with a large-scale publicly available dataset. Compared to that work, our database is made of real (not made-up) movies, and the choice of target movies is based on empirical distances between movies and movie features instead of being arbitrary. This, combined with the collaborative set-up, makes it possible to train a model for the seeker in the bot-play setting.

Our recommendation dialogue game is collaborative. Other dialogue settings with shared objectives have been explored, for example a collaborative graph prediction task (He et al., 2017a), and semi-cooperative negotiation tasks (Lewis et al., 2017; Yarats and Lewis, 2018; He et al., 2018).

## 6 Conclusion and Future Directions

In conclusion, we have posed recommendation as a goal-oriented game between an expert and a seeker, and provided a framework for both training agents in a supervised way by learning to mimic a large set of collected human-human dialogues, as well as by bot-play between trained agents. We have shown that a combination of the two stages leads to learning better expert recommenders.

Our results suggest several promising directions. First, we noted that the recommendation performance linearly increases as more dialogue context is provided. An interesting question is how to learn to produce the best questions that will result in the most informative dialogue context.

Second, as the model becomes better at the game, we observe an increase in the length of dialogue. However, it remains shorter than the average length of human dialogues, possibly because our reward function is designed to minimize it, which worked better in experiments. A potential direction for future work is to study how different game objectives interact with each other.

Finally, our evaluation on movie recommendation is made only within the candidate set of movies given to expert. Future work should evaluate if our training scheme generalizes to a fully open-ended recommendation system, thus making our task not only useful for research and model development, but a useful end-product in itself.

## Acknowledgements

We thank Eduard Hovy, Alan W Black, Dan Jurafsky, Alan Ritter, and anonymous reviewers for their helpful comments.

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Antoine Bordes, Y-Lan Boureau, and Jason Weston. 2016. Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *ACL*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

He He, Anusha Balakrishnan, Mihail Eric, and Percy Liang. 2017a. Learning symmetric collaborative dialogue agents with dynamic knowledge graph embeddings. In *ACL*.

He He, Derek Chen, Anusha Balakrishnan, and Percy Liang. 2018. Decoupling strategy and generation in negotiation dialogues. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2333–2343, Brussels, Belgium. Association for Computational Linguistics.

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017b. Neural collaborative filtering. In *WWW*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *EACL*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Yehuda Koren, Robert M. Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer*, 42.

Mike Lewis, Denis Yarats, Yann Dauphin, Devi Parikh, and Dhruv Batra. 2017. Deal or no deal? end-to-end learning for negotiation dialogues. In *EMNLP*.

Raymond Li, Samira Ebrahimi Kahou, Hannes Schulz, Vincent Michalski, Laurent Charlin, and Chris Pal. 2018. Towards deep conversational recommendations. In *NIPS*, pages 9725–9735.

Hao Ma, Dengyong Zhou, Chao Liu, Michael R. Lyu, and Irwin King. 2011. Recommender systems with social regularization. In *WSDM*.

Kanti V Mardia. 1970. Measures of multivariate skewness and kurtosis with applications. *Biometrika*, 57(3):519–530.

Alexander H. Miller, Will Feng, Dhruv Batra, Antoine Bordes, Adam Fisch, Jiasen Lu, Devi Parikh, and Jason Weston. 2017. Parlai: A dialog research software platform. In *EMNLP*.

Alexander H. Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *EMNLP*.

Raymond J Mooney and Loriene Roy. 2000. Content-based book recommending using learning for text categorization. In *Proceedings of the fifth ACM conference on Digital libraries*, pages 195–204. ACM.

OpenAI. 2018. Openai five. https://blog.openai.com/openai-five/.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Kevin Reschke, Adam Vogel, and Daniel Jurafsky. 2013. Generating recommendation dialogs by extracting information from user reviews. In *ACL*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL*.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. 2017. Mastering the game of go without human knowledge. *Nature*, 550(7676):354.

Oriol Vinyals, Igor Babuschkin, Junyoung Chung, Michael Mathieu, Max Jaderberg, Wojciech M. Czarnecki, Andrew Dudzik, Aja Huang, Petko Georgiev, Richard Powell, Timo Ewalds, Dan Horgan, Manuel Kroiss, Ivo Danihelka, John Agapiou, Junhyuk Oh, Valentin Dalibard, David Choi, Laurent Sifre, Yury Sulsky, Sasha Vezhnevets, James Molloy, Trevor Cai, David Budden, Tom Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Toby Pohlen, Yuhuai Wu, Dani Yogatama, Julia Cohen, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Chris Apps, Koray Kavukcuoglu, Demis Hassabis, and David Silver. 2019. AlphaStar: Mastering the Real-Time Strategy Game StarCraft II. https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/.

Pontus Wärnestål. 2005. Modeling a dialogue strategy for personalized movie recommendations. In *Beyond Personalization Workshop*, pages 77–82.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Ledell Yu Wu, Adam Fisch, Sumit Chopra, Keith Adams, Antoine Bordes, and Jason Weston. 2018. Starspace: Embed all the things! In *AAAI*.

Zhilin Yang, Zihang Dai, Ruslan R. Salakhutdinov, and William W. Cohen. 2018. Breaking the softmax bottleneck: A high-rank rnn language model. In *ICLR*.

Denis Yarats and Mike Lewis. 2018. Hierarchical text generation and planning for strategic dialogue. In *ICML*.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? In *ACL*.

# CoSQL: A Conversational Text-to-SQL Challenge Towards Cross-Domain Natural Language Interfaces to Databases

**Tao Yu**[†] **Rui Zhang**[†] **He Yang Er**[†] **Suyi Li**[†] **Eric Xue**[†] **Bo Pang**[†]
**Xi Victoria Lin**[¶] **Yi Chern Tan**[†] **Tianze Shi**[§] **Zihan Li**[‡] **Youxuan Jiang**[‡]
**Michihiro Yasunaga**[†] **Sungrok Shim**[†] **Tao Chen**[†] **Alexander Fabbri**[†]
**Zifan Li**[†] **Luyao Chen**[‡] **Yuwen Zhang**[‡] **Shreya Dixit**[†] **Vincent Zhang**[†]
**Caiming Xiong**[¶] **Richard Socher**[¶] **Walter S. Lasecki**[‡] **Dragomir Radev**[†]
[†]Yale University [¶]Salesforce Research
[‡]University of Michigan [§]Cornell University
{tao.yu, r.zhang, dragomir.radev}@yale.edu
{xilin, cxiong, rsocher}@salesforce.com

## Abstract

We present CoSQL, a corpus for building cross-domain, general-purpose database (DB) querying dialogue systems. It consists of 30k+ turns plus 10k+ annotated SQL queries, obtained from a Wizard-of-Oz (WOZ) collection of 3k dialogues querying 200 complex DBs spanning 138 domains. Each dialogue simulates a real-world DB query scenario with a crowd worker as a user exploring the DB and a SQL expert retrieving answers with SQL, clarifying ambiguous questions, or otherwise informing of unanswerable questions. When user questions are answerable by SQL, the expert describes the SQL and execution results to the user, hence maintaining a natural interaction flow. CoSQL introduces new challenges compared to existing task-oriented dialogue datasets: (1) the dialogue states are grounded in SQL, a domain-independent executable representation, instead of domain-specific slot-value pairs, and (2) because testing is done on unseen databases, success requires generalizing to new domains. CoSQL includes three tasks: SQL-grounded dialogue state tracking, response generation from query results, and user dialogue act prediction. We evaluate a set of strong baselines for each task and show that CoSQL presents significant challenges for future research. The dataset, baselines, and leaderboard will be released at https://yale-lily.github.io/cosql.

## 1 Introduction

Natural language interfaces to databases (NLIDB) have been studied extensively, with a multitude of different approaches introduced over the past few decades. To this end, considerable progress has been made in querying data via natural language (NL). However, most NL query systems expect the query to be well-formed and stated in a single sentence (Zelle and Mooney, 1996; Li and Jagadish, 2014; Yaghmazadeh et al., 2017; Iyer et al., 2017; Zhong et al., 2017; Xu et al., 2017; Shi et al., 2018;

$D_1$ : Database about student dormitories containing 5 tables

$Q_1$ : What are the names of all the dorms? INFORM_SQL

$S_1$ : `SELECT dorm_name FROM dorm`

$A_1$ : (Result table with many entries)

$R_1$ : This is the list of the names of all the dorms. CONFIRM_SQL

$Q_2$ : Which of those dorms have a TV lounge? INFORM_SQL

$S_2$ : `SELECT T1.dorm_name FROM dorm AS T1 JOIN has_amenity AS T2 ON T1.dormid = T2.dormid JOIN dorm_amenity AS T3 ON T2.amenid = T3.amenid WHERE T3.amenity_name = 'TV Lounge'`

$A_2$ : (Result table with many entries)

$R_2$ : This shows the names of dorms with TV lounges. CONFIRM_SQL

$Q_3$ : What dorms have no study rooms as amenities? AMBIGUOUS

$R_3$ : Do you mean among those with TV Lounges? CLARIFY

$Q_4$ : Yes. AFFIRM

$S_4$ : `SELECT T1.dorm_name FROM dorm AS T1 JOIN has_amenity AS T2 ON T1.dormid = T2.dormid JOIN dorm_amenity AS T3 ON T2.amenid = T3.amenid WHERE T3.amenity_name = 'TV Lounge' EXCEPT SELECT T1.dorm_name FROM dorm AS T1 JOIN has_amenity AS T2 ON T1.dormid = T2.dormid JOIN dorm_amenity AS T3 ON T2.amenid = T3.amenid WHERE T3.amenity_name = 'Study Room'`

$A_4$ : Fawlty Towers

$R_4$ : Fawlty Towers is the name of the dorm that has a TV lounge but not a study room as an amenity. CONFIRM_SQL

...

$Q_8$ : Thanks! THANK_YOU

$R_8$ : You are welcome. WELCOME

Figure 1: A dialog from the CoSQL dataset. Gray boxes separate the user inputs ($Q_i$) querying the database ($D_i$) from the SQL queries ($S_i$), returned answers ($A_i$), and expert responses ($R_i$). Users send an input to the expert, who writes the corresponding SQL query (only seen by the expert) if possible and sends an answer and response description back. Dialogue acts are on the right-hand side (e.g., $Q_3$ is "ambiguous" and $R_3$ is "clarify").



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1962–1979,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

Wang et al., 2018; Yu et al., 2018b,c). In reality, complex questions are usually answered through interactive exchanges (Figure 1). Even for simple queries, people tend to explore the database by asking multiple basic, interrelated questions (Hale, 2006; Levy, 2008; Frank, 2013; Iyyer et al., 2017). This requires systems capable of sequentially processing conversational requests to access information in relational databases. To drive the progress of building a context-dependent NL query system, corpora such as ATIS (Hemphill et al., 1990; Dahl et al., 1994) and SParC (Yu et al., 2019)[1] have been released. However, these corpora assume all user questions can be mapped into SQL queries and do not include system responses.

Furthermore, in many cases, multi-turn interaction between users and NL systems is needed to clarify ambiguous questions (e.g., $Q_3$ and $R_3$ in Figure 1), verify returned results, and notify users of unanswerable or unrelated questions. Therefore, a robust dialogue-based NL query agent that can engage with users by forming its own responses has become an increasingly necessary component for the query process. Such systems have already been studied under task-oriented dialogue settings by virtue of continuous effort of corpus creation (Seneff and Polifroni, 2000; Walker et al., 2002; Raux et al., 2005; Mrksic et al., 2015; Asri et al., 2017; Budzianowski et al., 2018) and modelling innovation (Artzi and Zettlemoyer, 2011; Henderson et al., 2013; Lee and Dernoncourt, 2016; hao Su et al., 2016; Dhingra et al., 2016; Li et al., 2016; Mrksic et al., 2017). The goal of these systems is to help users accomplish a specific task, such as flight or hotel booking or transportation planning. However, to achieve these goals, task-oriented dialogue systems rely on pre-defined slots and values for request processing (which can be represented using simple SQL queries consisting of `SELECT` and `WHERE` clauses). Thus, these systems only operate on a small number of domains and have difficulty capturing the diverse semantics of practical user questions.

In contrast, the goal of dialogue-based NLIDB systems is to support general-purpose exploration and querying of databases by end users. To do so, these systems must possess the ability to (1) detect questions answerable by SQL, (2) ground user questions into executable SQL queries if possible, (3) return results to the user in a way that is easily understood and verifiable, and (4) handle unanswerable questions. The difficulty of constructing dialogue-based NLIDB systems stems from these requirements. To enable modeling advances in this field, we introduce CoSQL, the first large-scale cross-domain **Co**nversational text-to-**SQL** corpus collected under the WOZ setting (Budzianowski et al., 2018). CoSQL contains 3,007 dialogues (more than 30k turns with annotated dialogue acts and 10k expert-labeled SQL queries) querying 200 complex DBs spanning across 138 different domains. For each dialogue, we follow the WOZ setup that involves a crowd worker as a DB user and a college computer science student who is familiar with SQL as an expert (§3).

Like Spider[2] (Yu et al., 2018c) and SParC (Yu et al., 2019), the cross-domain setting in CoSQL enables us to test the ability of systems to generalize on querying different domains via dialogues. We split the dataset in a way that each database only appears in one of train, development, or test set. This setting requires systems to generalize to new domains without additional annotation.

More importantly, unlike most prior work in text-to-SQL systems, CoSQL demonstrates greater language diversity and more frequent user focus changes. It also includes a significant amount of questions that require user clarification and questions that cannot be mapped to SQL queries, introducing the potential to evaluate text-to-SQL dialog act prediction. These features pose new challenges for text-to-SQL systems. Moreover, CoSQL includes system responses that describe SQL queries and the returned results in a way that is easy for users with different backgrounds to understand and verify, as faithful and comprehensible presentation of query results is a crucial component of any NLIDB system.[3]

We introduce three challenge tasks on CoSQL: (1) **SQL-grounded dialogue state tracking** to map user utterances into SQL queries if possible given the interaction history (§5.1), (2) **natural language response generation** based on an executed SQL and its results for user verification

[1] SParC task is available at https://yale-lily.github.io/sparc

[2] Spider task is available at https://yale-lily.github.io/spider

[3] The DB community has developed query visualization (Li and Jagadish, 2014) and other techniques to provide faithful explanation of a SQL query. These explanations are complementary to the NL ones used in our work and future NLIDB systems could integrate them.

(§5.2) and (3) **user dialogue act prediction** to detect and resolve ambiguous and unanswerable questions (§5.3). We provide detailed data analysis and qualitative examples (§4). For each of the three tasks, we benchmark several competitive baseline models (§6). The performances of these models indicate plenty of room for improvement.

## 2 Related Work

**Text-to-SQL generation** Text-to-SQL generation has been studied for decades in both DB and NLP communities. (Warren and Pereira, 1982; Zettlemoyer and Collins, 2005; Popescu et al., 2003; Li et al., 2006; Li and Jagadish, 2014; Iyer et al., 2017; Zhong et al., 2017; Xu et al., 2017; Yu et al., 2018a; Dong and Lapata, 2018; Finegan-Dollak et al., 2018; Guo et al., 2019; Bogin et al., 2019). However, the majority of previous work focus on converting a single, complex question into its corresponding SQL query. Only a few datasets have been constructed for the purpose of mapping context-dependent questions to structured queries. Price (1990); Dahl et al. (1994) collected ATIS that includes series of questions from users interacting with a flight database. Yu et al. (2019) introduced SParC, a large cross-domain semantic parsing in context dataset, consisting of 4k question sequences with 12k questions annotated with SQL queries over 200 complex DBs. Similar to ATIS, SParC includes sequences of questions instead of conversational interactions. An NL question and its corresponding SQL annotation in SParC are constructed by the same expert. Recent works (Suhr et al., 2018; Zhang et al., 2019) have built context-dependent text-to-SQL systems on top of these datasets.

In contrast, CoSQL was collected under a WOZ setting involving interactions between two parties, which contributes to its diverse semantics and discourse covering most types of conversational DB querying interactions (e.g. the system will ask for clarification of ambiguous questions, or inform the user of unanswerable and irrelevant questions). Also, CoSQL includes a natural language system response for the user to understand and verify the systems actions.

**Task-oriented dialog systems** Task-oriented dialog systems (Henderson et al., 2014; Wen et al., 2016; Mrkšić et al., 2017; Budzianowski et al., 2018) have attracted increasing attention especially due to their commercial values. The goal is to help users accomplish a specific task such as hotel reservation, flight booking, or travel information. These systems (Bordes and Weston, 2017; Zhong et al., 2018; Wu et al., 2019) often predefine slot templates grounded to domain-specific ontology, limiting the ability to generalize to unseen domains. In comparison, our work is to build a system for general-purpose DB exploration and querying. The domain-independent intent representation (SQL query) enables the trained system to work on unseen domains (DB schemas).

While most task-oriented dialog systems need to actively poke the user for information to fill in pre-defined slot-value pairs, the primary goal of system responses in CoSQL is to offer users a reliable way to understand and verify the returned results. If a question can be converted into a SQL query, the user is shown the execution result and the system will describe the SQL query and the result in natural language. In case sthe user questions are ambiguous or unanswerable by SQL, the system either requests the user to rephrase or informs them to ask other questions.

**Data-to-Text generation** Response generation in CoSQL takes a structured SQL query and its corresponding result table to generate an NL description of the system's interpretation of the user request. Compared to most dialogue-act-to-text generation tasks, the richer semantics of SQL queries makes our task more challenging – besides generating natural and coherent descriptions, faithfully preserving the logic of a SQL query in an NL response is also crucial in our task. Furthermore, this component is related to previous work on text generation from structured data (McKeown, 1985; Iyer et al., 2016; Wiseman et al., 2017).

## 3 Data Collection

We follow the Wizard-of-Oz setup which facilitates dialogues between DB users and SQL experts to create CoSQL. We recruited Amazon Mechanical Turkers (AMT) to act as DB users and trained 25 graduate- and undergraduate-level computer science students proficient in SQL to act as DB experts. The collection interface (Lasecki et al., 2013) is designed to be easy-to-operate for the experts and intuitive for the users. Detailed explanations of the data collection process is provided below.

**Reference goal selection** We pre-select a reference goal for each dialogue to ensure the interaction is meaningful and to reduce redundancy within the dataset. Users are asked to explore the given DB content to come up with questions that are likely to naturally arise in real-life scenarios and reflect their query intentions as specified by the reference goals. Following Yu et al. (2019), we selected the complex questions classified as medium, hard, and extra hard in Spider (Yu et al., 2018c) as the reference goals.[4] In total, 3,783 questions were selected on 200 databases. After annotation and reviewing, 3,007 of them were finished and kept in the final dataset.

**User setup** We developed online chatting interfaces to pair the user with the expert (Figure 6 and 7 in Appendix). When a data collection session starts, the user is first shown multiple tables from a DB to which a reference goal is groundedand is required to read through them. Once they have examined the data stored in the tables, the reference goal question will be revealed on the same screen. The user is encouraged to use the goal question as a guide to ask interrelated questions, but is also allowed to ask other questions exploring the DB. We require the user to ask at least 3 questions.[5] In each turn, if the user question can be answered by a SQL query, they will be shown the result table, and the expert will write an NL response interpreting the executed SQL query based on their understanding of the user's query intent (Figure 8 Appendix). If the user question is ambiguous or cannot be answered with SQL, they will receive clarification questions or notice to rephrase from the expert (detailed in expert setup).

**Expert setup** Within each session, the expert is shown the same DB content and the reference goal as the user (Figure 8 in Appendix). For each dialogue turn, the expert first checks the user question and labels it using a set of pre-defined user dialog action types (DATs, see Table 4). Then the expert sets the DAT of his response according to the user DAT. Both the user and the expert can have multiple DATs labels in each turn. If the user question is answerable in SQL (labeled as `INFORM_SQL`, e.g. $Q_1$ in Figure 1 ), the expert writes down the SQL query[6], executes it, checks the result table, and sends the result table to the user. The expert then describes the SQL query and result table in natural language and sends the response. If the user question is ambiguous, the expert needs to write an appropriate response to clarify the ambiguity (labeled as `AMBIGUOUS`, e.g. $Q_3$ in Figure 1). Some user questions require the expert to infer the answer based on their world knowledge (labeled as `INFER_SQL`, e.g. $Q_3$ in Figure 10). If the user question cannot be answered by SQL, the expert will inform them to ask well-formed questions (labeled as `NOT_RELATED`, `CANNOT_UNDERSTAND`, or `CANNOT_ANSWER`). In other cases (labeled as `GREETING`, `THANK_YOU`, etc.), the expert responds with general dialogue expressions ($Q_8$ in Figure 1).

**User quality control** Because of the real-time dialogue setting and the expensive annotation procedures on the expert side, conducting quality control on user is crucial for our data collection. We use LegionTools[7] (Lasecki et al., 2014) to post our tasks onto AMT and to recruit and route AMT workers for synchronous real time crowd sourcing tasks. We specify that only workers from the U.S. with 95% approval rates are allowed to accept our task. Before proceeding to the chat room, each AMT worker has to go through a tutorial and pass two short questions[8] to test their knowledge about our task. Only the user who passes the quiz proceeds to the chat room. Throughout the data collection, if a user ignores our instructions in a specific turn, we allow the experts to alert the user through chat and label the corresponding turn as `DROP`. If a user's actions continue to deviate from instructions, the expert can terminate the dialog before it ends. After each dialogue session terminates, we ask the expert to provide a score from 1 to 5 as an evaluation of the user's performance. Dialogues with a score below 3 are dropped and the user will be blocked from future participation.

[4] Yu et al. (2019) also includes 12.9% of the easy questions in Spider in order to increase dataset diversity. In this work we prioritize the complex questions that trigger more interesting interactions and do not include any easy questions.

[5] The worker is paid $1.20 USD for each dialog. To encourage interaction, we offer $0.50 USD bonus for each dialogue if the user asks more than 4 interesting, interrelated questions.

[6] We use the same SQL annotation protocol as Spider (Yu et al., 2018c) to ensure the same SQL pattern was chosen when multiple equivalent queries were available.

[7] https://www.cromalab.net/LegionTools/

[8] One is on how to ask interrelated questions and the other is on how to read multiple tables with reference keys.

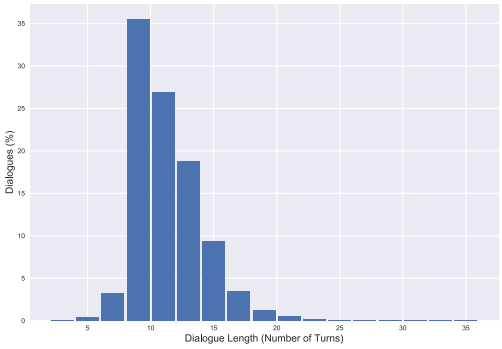


Figure 2: Distributions of dialogue lengths.

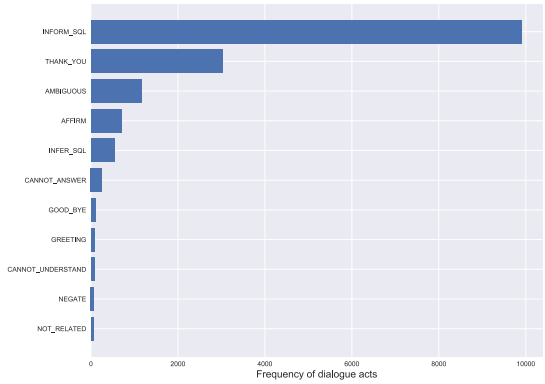


Figure 3: Distributions of user dialog action types.

**Data review and post-process** We conduct a multi-pass data reviewing process.[9] Two student conducted a first-round review. They focus on correcting any errors in the DATs of the users and the experts, checking if the SQL queries match the user's questions, and modifying or rewriting the expert's responses to contain necessary information in the SQL queries in case they miss any of them. Also, they re-evaluate all dialogues based on the diversity of user questions and reject any dialogues that only contain repeated, simple, and thematically-independent user questions (about 6% of the dialogs). After the first-round review, another two student experts reviewed the refined data to double check the correctness of the DATs, the SQL queries, and the expert responses. They also corrected any grammar errors, and rephrased the user's questions and the expert's responses in a more natural way if necessary. Finally, we ran and parsed all annotated SQL queries to make sure they were executable, following the same annotation protocol as the Spider dataset.

## 4 Data Statistics and Analysis

We report the statistics of CoSQL and compare it to other task-oriented dialog and context-dependent text-to-SQL datasets. We also conduct detailed analyses on its contextual, cross-domain nature, and question diversity.

**Data statistics** Table 1 and 2 summarize the statistics of CoSQL. CoSQL contains 3k+ dialogues in total (2,164 in training), which is comparable to or bigger than most commonly used task-oriented dialogue datasets. Figure 2 shows the distribution of dialogue length in the corpus, approximately 80% of dialogues involve 8 or more turns, with a total of 31,148 turns.[10] The average number of tokens in each turn is 11.21. Noticeably, the domain of CoSQL spans over 200 complex databases, overshadowing most other task-oriented dialogue datasets. Comparing to existing context-dependent text-to-SQL datasets, CoSQL contains significantly more turns, out of which 11,039 user utterances are convertible to SQL. In contrast, all NL utterances in ATIS and SParC can be mapped to SQL. CoSQL also has a much larger NL vocabulary.

**Dialogue act distribution** As shown in Figure 3, CoSQL contains a fairly diverse set of user dialogue action types (DATs). Unsurprisingly, `INFORM_SQL` and `THANK_YOU` are the two most commonly seen DATs. Among the rest of DATs, approximately 40% are `AMBIGUOUS`, demonstrating the paramount importance of system clarification in the DB querying process in general. Another 20% of this subgroup is `INFER_SQL`, which signifies questions that cannot be answered without the aid of human inference.

**Semantic complexity** As shown in Table 1, if we consider the column names of the 200 DBs of CoSQL as slots and their entries as values, the number of slot-value pairs far exceed those defined in other task-oriented dialogues. Figure 4 shows the total number of occurrences of different SQL keywords in the SQL queries corresponding to these questions. The SQL queries in CoSQL cover all common SQL keywords as well as complicated syntactic structure such as nesting (Fig-

[9]The review interface is shown in Figure 9 (Appendix).

[10]Following Budzianowski et al. (2018), in the statistics report we define the # turns in a dialogue to be the total # messages in the dialogue.

| | DSTC2 | WOZ 2.0 | KVRET | MultiWOZ | **CoSQL** |
|---|---|---|---|---|---|
| # dialogs | 1,612 | 600 | 2,425 | 8,438 | 2,164 |
| Total # turns | 23,354 | 4,472 | 12,732 | 115,424 | 22,422 |
| Total # tokens | 199,431 | 50,264 | 102.077 | 1,520,970 | 22.8197 |
| Avg. # turns/dialog | 14.49 | 7.45 | 5.25 | 13.68 | 10.36 |
| Avg. # tokens/turn | 8.54 | 11.24 | 8.02 | 13.18 | 11.34 |
| Total # unique tokens | 986 | 2,142 | 2,842 | 24,071 | 7,502 |
| # databases | 1 | 1 | 1 | 7 | 140 |
| # Slots # | 8 | 4 | 13 | 25 | 3,696 |
| # Values # | 212 | 99 | 1,363 | 4,510 | >1,000,000 |

Table 1: Comparison of CoSQL to some commonly used task-oriented dialogue datasets. The numbers are computed for the training part of data in consistency with previous work (Budzianowski et al., 2018).

| | **CoSQL** | SParC | ATIS |
|---|---|---|---|
| # Q sequence | 3,007 | 4298 | 1658 |
| # user questions | 15,598* | 12,726 | 11,653 |
| # databases | 200 | 200 | 1 |
| # tables | 1020 | 1020 | 27 |
| Avg. Q len | 11.2 | 8.1 | 10.2 |
| Vocab | 9,585 | 3794 | 1582 |
| Avg. # Q turns | 5.2 | 3.0 | 7.0 |
| Unanswerable Q | ✓ | ✗ | ✗ |
| User intent | ✓ | ✗ | ✗ |
| System response | ✓ | ✗ | ✗ |

Table 2: Comparison of CoSQL with other context-dependent text-to-SQL datasets. The number are computed over the entire datasets. *For CoSQL we count the total # user utterances.

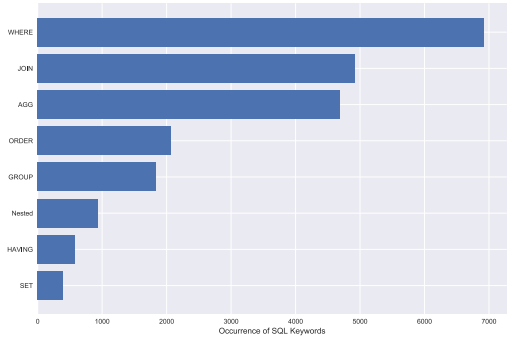


Figure 4: SQL keyword counts.

ure 5).

**Semantic changes by turns** We compute the frequency of occurrences of common SQL keywords in different turns for both CoSQL and SParC and compare them in Figure 5 (upper: CoSQL, lower: SParC). Here we count the turn # based on user utterance only. Since CoSQL and SParC span the same domains, Figure 5 reveals a comparison of semantic changes between context-dependent DB questions issued by end

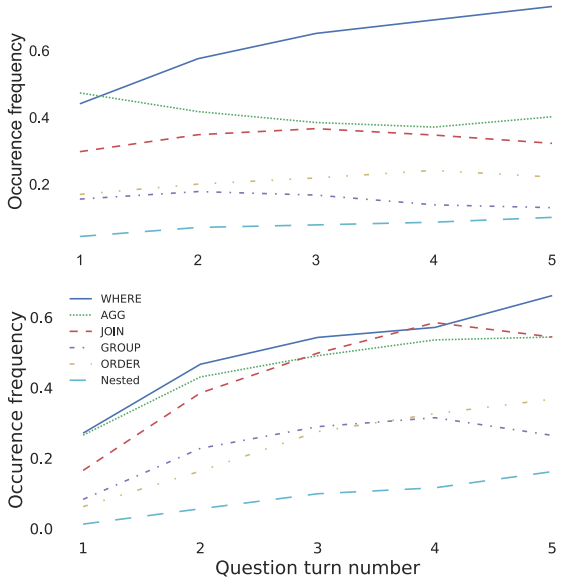


Figure 5: Percentage of question sequences that contain a particular SQL keyword at a specific user utterance turn. The keyword occurrences in CoSQL (upper) slightly fluctuates as the interaction proceeds while that in SParC (lower) demonstrates a clear increasing trend.

users (CoSQL) and expert users (SParC). For CoSQL, the frequencies of all keywords except for `WHERE` do not change significantly throughout the conversation, and the average frequencies of these keywords are in general lower than those of SParC. In addition, `WHERE` occurs slightly more frequently in CoSQL than in SParC. We believe this indicates the exploratory nature of the dialogues we collected, as the users switch their focus more frequently instead of building questions upon previous ones. For example, SQL `AGG` components occur most frequently in the beginning of dialogues, as a result of users familiarizing themselves with the amount of data in the DB or other statistical measures. In contrast, the frequencies of almost all SQL components in SParC increase as the question turn increases. This sug-

gests that questions in SParC have stronger interdependency, as the purpose of this corpus is to study text-to-SQL in context.

**Cross domain** As shown in Table 3, the dialogues in CoSQL are randomly split into train, development and test sets by DB with a ratio of 7:1:2 (the same split as SParC and Spider).

| | Train | Dev | Test |
|---|---|---|---|
| # Dialogs | 2164 | 292 | 551 |
| # Databases | 140 | 20 | 40 |

Table 3: Dataset Split Statistics

## 5 Tasks and Models

CoSQL is meant to be used as the first benchmark for building general-purpose DB querying dialogue systems in arbitrary domains. Such systems take a user question and determine if it can be answered by SQL (user dialogue act prediction). If the question can be answered by SQL, the system translates it into the corresponding SQL query (SQL-grounded dialogue state tracking), executes the query, returns and shows the result to the user. To improve interpretability and trustworthiness of the result, the system describes the predicted SQL query and result tables to the user for their verification (response generation from SQL and result). Finally, the user checks the results and the system responses and decides if the desired information is obtained or additional questions shall be asked.

Some components relevant to the process above are beyond the scope of our work. First, our response generation task only includes turns where the system's dialogue act is `CONFORM_SQL`. In case the system cannot understand the user's question (the system dialogue act is `CANNOT_ANSWER`) or considers it as unanswerable (`CANNOT_ANSWER`), the system will reply in a standard way to inform the user that it needs clarification or cannot answer that question. The same applies to questions that require human inference (e.g., the system confirms with the user which types of dorms he or she was talking about by asking $R_3$ instead of immediately translating $Q_3$ in Figure 1). Currently we do not have a task setup to evaluate the quality of system clarifications. Second, some user questions cannot be directly answered by SQL but are possible to be answered with other type of logical reasoning (e.g., $Q_3$ in Figure 10). We exclude these questions from our task design and leave them for future research.

### 5.1 SQL-Grounded dialogue State Tracking

In CoSQL, user dialogue states are grounded in SQL queries. Dialogue state tracking (DST) in this case is to predict the correct SQL query for each user utterance with `INFORM_SQL` label given the interaction context and the DB schema. In our setup, the system does not have access to gold SQL queries from previous turns, which is different from the traditional DST settings in dialogue management where the history of ground-truth dialogue states is given. Comparing to other context-dependent text-to-SQL tasks such as SParC and ATIS, the DST task in CoSQL also include the ambiguous questions if the user affirms the system clarification of them (e.g., $Q_4$ in Figure 1). In this case, the system clarification is also given as part of the interaction context to predict the SQL query corresponding to the question.[11][12] For instance, to generate $S_4$ in Figure 1, the input consists of all previous questions ($Q_1, Q_2, Q_3$), the current user question ($Q_4$), the DB schema, and the system response $R_3$.

We benchmark the performance of two strong context-dependent neural text-to-SQL models on this the task, which are the baseline models reported on SParC by Yu et al. (2019).

**Context-dependent Seq2Seq (CD-Seq2Seq)** The model is originally introduced by (Suhr et al., 2018) for the ATIS task. It incorporates interaction history and is able to copy segments of previous generated SQL queries. Yu et al. (2019) extends it to encode DB schema information such that it works for the cross-domain setting in SParC. We apply the model to our task without any changes.

**SyntaxSQL-con** SyntaxSQLNet is a SQL-specific syntax-tree based model introduced for Spider (Yu et al., 2018b). Yu et al. (2019) extends it to take previous questions as input when predicting SQL for the current question. We apply the model to our task without any changes.

[11] If a dialogue contains multiple ambiguous questions, the system clarification to all ambiguous questions will be given as input.

[12] The ambiguous questions not confirmed by the user and their system responses are given as part of the conversation history but we do not require a system to predict SQL queries for them.

### 5.2 Response Generation from SQL and Query Results

This task requires generating a natural language description of the SQL query and the result for each system response labeled as CONFORM_SQL. It considers a SQL query, the execution result, and the DB schema. Preserving logical consistency between SQL and NL response is crucial in this task, in addition to naturalness and syntactical correctness. Unlike other SQL-to-text generation tasks (Xu et al., 2018), our task maps the SQL query to a statement and summarizes the result in that statement (instead of just mapping it back to the user question).

We experiment with three baseline methods for this task.

**Template-based** Given the SQL and NL response pairs in the training set, we masked variable values in both the SQL and NL response to form parallel SQL-response templates. Given a new SQL query, we employ rule-based approach to select the closest SQL-response template pair from the set. After that, we fill in the selected response template with the columns, tables, and values of the SQL query and the result to generate the final response (see more in Appendix).

**Seq2Seq** We experiment with a vanilla Seq2Seq model (Sutskever et al., 2014) with attention (Bahdanau et al., 2015), a standard baseline for text generation tasks.

**Pointer-generator** Oftentimes the column or table names in the NL response are copied from the input SQL query. To capture this phenomenon, we experiment with a pointer-generator network (See et al., 2017), which addresses the problem of out-of-vocabulary word generation in summarization and other text generation tasks. We use a modified version of the implementation from Chen and Bansal (2018).

### 5.3 User dialogue Act Prediction

For a real-world DB querying dialogue system, it has to decide if the user question can be mapped to a SQL query or if special actions are needed. We define a series of dialogue acts for the DB user and the SQL expert (Table 4).[13] For example, if the user question can be answered by a SQL query, the dialogue act of the question is INFORM_SQL. Since the system DATs are defined in response to the user DATs, our task does not include system dialogue acts prediction.

We experiment with two baseline models for this task.

**Majority** The dialogue acts of all the user questions are predicted to be the majority dialogue act INFORM_SQL.

**TBCNN-pair** We employ TBCNN-pair (Mou et al., 2016), a tree-based CNN model with heuristics for predicting entailment and contradiction between sentences. We change the two sentence inputs for the model to a user utterance and the DB schema, and follow the same method in SQL-Net (Xu et al., 2017) to encode each column name.

## 6 Results and Discussion

**SQL-grounded dialog state tracking** We use the same evaluation metrics used by the SParC dataset (Yu et al., 2019) to evaluate the model's performance on all questions and interactions (dialogs). The performances of CD-Seq2Seq and SyntaxSQL-con are reported in Table 5. The two models achieve less than 16% question-level accuracy and less than 3% on interaction-level accuracy. Since the two models have been benchmarked on both CoSQL and SParC, we cross-compare their performance on these two datasets. Both models perform significantly worse on CoSQL DST than on SParC. This indicates that CoSQL DST is more difficult than SParC. The possible reasons is that the questions in CoSQL are generated by a more diverse pool of users (crowd workers instead of SQL experts), the task includes ambiguous questions and the context contains more complex intent switches.

**Response generation** Table 6 shows the results of three different baselines on three metrics: BLEU score (Papineni et al., 2002), logic correctness rate (LCR), and grammar. To compute LCR and grammar score, we randomly sampled 100 descriptions generated by each model. Three students proficient in English participated in the evaluation, They were asked to choose a score 0 or 1 for LCR, and 1 to 5 for grammar check (the larger, the better). For LCR, the final score was decided by majority vote. We computed the average grammar score.

[13] §A.1 defines the complete set of dialogue action types.

| Groups | Dialog acts |
|---|---|
| DB user | inform_sql, infer_sql, ambiguous, affirm, negate, not_related, cannot_understand, cannot_answer, greeting, goodbye, thank_you |
| DB expert | conform_sql, clarify, reject, request_more, greeting, sorry, welcome, goodbye |

Table 4: Dialog acts in CoSQL. See § A.1 for the comprehensive definition of each dialogue act.

| Model | Question Match | | Interaction Match | |
|---|---|---|---|---|
| | Dev | Test | Dev | Test |
| CD-Seq2Seq | 13.8 | 13.9 | 2.1 | 2.6 |
| SyntaxSQL-con | 15.1 | 14.1 | 2.7 | 2.2 |

Table 5: Performance of various methods over all questions (*question match*) and all interactions (*interaction match*).

| Model | BLEU | | LCR (%) | Grammar |
|---|---|---|---|---|
| | Dev | Test | Test | Test |
| Template | 9.5 | 9.3 | 41.0 | 4.0 |
| Seq2Seq | 15.3 | 14.1 | 27.0 | 3.5 |
| Pointer-generator | 16.4 | 15.1 | 35.0 | 3.6 |

Table 6: BLEU scores on the development and test sets, and human evaluations of logic correctness rate (LCR) and grammar check on the 100 examples randomly sampled from the test set.

Interestingly, the human evaluation and BLEU scores do not completely agree. While the template-based method is brittle and requires manual effort, it performs significantly better than the two end-to-end neural models in the human evaluation. Because the SQL-question templates provide natural and grammatical sketch of the output, it serves as an advantage in our human evaluation. However, this approach is limited by the small coverage of the training templates and its LCR is only around 40%. On the other hand, the neural models achieve better BLEU scores than the template-based approach. A possible reason for this is that they tend to generate words frequently associated with certain SQL queries. However, the neural models struggle to preserve the SQL query logic in the output. Unsurprisingly, pointer-generator performs better than basic Seq2Seq in terms of both BLEU and human evaluation. The low performances of all methods on LCR show that the task is indeed very challenging.

| Model | Dev | Test |
|---|---|---|
| Majority | 63.3 | 62.8 |
| TBCNN-pair | 84.2 | 83.9 |

Table 7: Accuracy of user dialog act prediction on the development and test sets.

**User dialog act prediction** Table 7 shows the accuracy of the two baselines on predicting user dialog acts. The result of Majority indicates that about 40% of user questions cannot be directly converted into SQL queries. This confirms the necessity of considering a larger set of dialogue actions for building a practical NLIDB system. Even though TBCNN can predict around 85% of user intents correctly, most of the correct predictions are for simple classes such as INFORM_SQL, THANK_YOU, and GOODBYE etc. The F-scores for more interesting and important dialog acts such as INFER_SQL and AMBIGUOUS are around 10%. This indicates that improving the accuracy on user DAT prediction is still important.

## 7 Conclusion and Future Work

In this paper, we introduce CoSQL, the first large-scale cross-domain conversational text-to-SQL corpus collected under a Wizard-of-Oz setup. Its language and discourse diversity and cross-domain setting raise exciting open problems for future research. Especially, the baseline model performances on the three challenge tasks suggest plenty space for improvement. The data and challenge leaderboard will be publicly available at https://yale-lily.github.io/cosql.

**Future Work** As discussed in Section 5, some examples in CoSQL include ambiguous and unanswerable user questions and we do not study how a system can effectively clarify those questions or guide the user to ask questions that are answerable. Also, some user questions cannot be answered with SQL but by other forms of logical reasoning the correct answer can be derived. We urge the community to investigate these problems in future work in order to build practical, robust and reliable conversational natural language interfaces to databases.

## 8 Acknowledgement

This work was supported in part by IBM under the Sapphire Project at the University of Michigan.

## References


Yoav Artzi and Luke S. Zettlemoyer. 2011. Bootstrapping semantic parsers from conversations. In *EMNLP*.

Layla El Asri, Hannes Schulz, Shikhar Sharma, Jeremie Zumer, Justin Harris, Emery Fine, Rahul Mehrotra, and Kaheer Suleman. 2017. Frames: A corpus for adding memory to goal-oriented dialogue systems. *CoRR*, abs/1704.00057.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Ben Bogin, Jonathan Berant, and Matt Gardner. 2019. Representing schema structure with graph neural networks for text-to-sql parsing. In *ACL*.

Antoine Bordes and Jason Weston. 2017. Learning end-to-end goal-oriented dialog. *ICLR*.

Pawel Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. 2018. Multiwoz - a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In *EMNLP*.

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. In *Proceedings of ACL*.

Deborah A. Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. 1994. Expanding the scope of the atis task: The atis-3 corpus. In *Proceedings of the Workshop on Human Language Technology*, HLT '94, Stroudsburg, PA, USA. Association for Computational Linguistics.

Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2016. End-to-end reinforcement learning of dialogue agents for information access. In *ACL*.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 731–742. Association for Computational Linguistics.

Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan Dhanalakshmi Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving text-to-sql evaluation methodology. In *ACL 2018*. Association for Computational Linguistics.

Stefan L. Frank. 2013. Uncertainty reduction as a measure of cognitive load in sentence comprehension. *Topics in cognitive science*, 5 3:475–94.

Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao pei Liu Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. 2019. Towards complex text-to-sql in cross-domain database with intermediate representation. In *ACL*.

John Hale. 2006. Uncertainty about the rest of the sentence. *Cognitive science*, 30 4:643–72.

Charles T. Hemphill, John J. Godfrey, and George R. Doddington. 1990. The atis spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990*.

Matthew Henderson, Blaise Thomson, and Jason D. Williams. 2014. The second dialog state tracking challenge. In *SIGDIAL Conference*.

Matthew Henderson, Blaise Thomson, and Steve Young. 2013. Deep neural network approach for the dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 467–471, Metz, France. Association for Computational Linguistics.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. *CoRR*, abs/1704.08760.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. 2016. Summarizing source code using a neural attention model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2073–2083, Berlin, Germany. Association for Computational Linguistics.

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1821–1831. Association for Computational Linguistics.

Walter Lasecki, Ece Kamar, and Dan Bohus. 2013. Conversations in the crowd: Collecting data for task-oriented dialog learning. In *In Proceedings of the Human Computation Workshop on Scaling Speech and Language Understanding and Dialog through Crowdsourcing at HCOMP 2013.*

Walter S. Lasecki, Mitchell Gordon, Danai Koutra, Malte F. Jung, Steven P. Dow, and Jeffrey P. Bigham. 2014. Glance: Rapidly coding behavioral video with the crowd. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, New York, NY, USA. ACM.

Ji Young Lee and Franck Dernoncourt. 2016. Sequential short-text classification with recurrent and convolutional neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of*

*the Association for Computational Linguistics: Human Language Technologies*, pages 515–520, San Diego, California. Association for Computational Linguistics.

Roger P. Levy. 2008. Expectation-based syntactic comprehension. *Cognition*, 106:1126–1177.

Fei Li and HV Jagadish. 2014. Constructing an interactive natural language interface for relational databases. *VLDB*.

Jiwei Li, Will Monroe, Alan Ritter, Daniel Jurafsky, Michel Galley, and Jianfeng Gao. 2016. Deep reinforcement learning for dialogue generation. In *EMNLP*.

Yunyao Li, Huahai Yang, and HV Jagadish. 2006. Constructing a generic natural language interface for an xml database. In *EDBT*, volume 3896, pages 737–754. Springer.

Kathleen R. McKeown. 1985. *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text*. Cambridge University Press, New York, NY, USA.

Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Natural language inference by tree-based convolution and heuristic matching. In *ACL*.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2017. Neural belief tracker: Data-driven dialogue state tracking. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1777–1788. Association for Computational Linguistics.

Nikola Mrksic, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gasic, Pei hao Su, David Vandyke, Tsung-Hsien Wen, and Steve J. Young. 2015. Multi-domain dialog state tracking using recurrent neural networks. In *ACL*.

Nikola Mrksic, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve J. Young. 2017. Neural belief tracker: Data-driven dialogue state tracking.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318. Association for Computational Linguistics.

Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. 2003. Towards a theory of natural language interfaces to databases. In *Proceedings of the 8th international conference on Intelligent user interfaces*, pages 149–157. ACM.

P. J. Price. 1990. Evaluation of spoken language systems: the atis domain. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990*, pages 91–95.

Antoine Raux, Brian Langner, Dan Bohus, Alan W. Black, and Maxine Eskénazi. 2005. Let's go public! taking a spoken dialog system to the real world. In *INTERSPEECH*.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1073–1083.

Stephanie Seneff and Joseph Polifroni. 2000. Dialogue management in the mercury flight reservation system. In *Proceedings of the 2000 ANLP/NAACL Workshop on Conversational Systems - Volume 3*, ANLP/NAACL-ConvSyst '00, pages 11–16.

Tianze Shi, Kedar Tatwawadi, Kaushik Chakrabarti, Yi Mao, Oleksandr Polozov, and Weizhu Chen. 2018. Incsql: Training incremental text-to-sql parsers with non-deterministic oracles. *arXiv preprint arXiv:1809.05054*.

Pei hao Su, Milica Gasic, Nikola Mrksic, Lina Maria Rojas-Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve J. Young. 2016. Online active reward learning for policy optimisation in spoken dialogue systems. *ACL*.

Alane Suhr, Srinivasan Iyer, and Yoav Artzi. 2018. Learning to map context-dependent sentences to executable formal queries. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2238–2249. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Marilyn A. Walker, Alexander I. Rudnicky, Rashmi Prasad, John S. Aberdeen, Elizabeth Owen Bratt, John S. Garofolo, Helen F. Hastie, Audrey N. Le, Bryan L. Pellom, Alexandros Potamianos, Rebecca J. Passonneau, Salim Roukos, Gregory A. Sanders, Stephanie Seneff, and David Stallard. 2002. Darpa communicator: cross-system results for the 2001 evaluation. In *INTERSPEECH*.

Chenglong Wang, Po-Sen Huang, Alex Polozov, Marc Brockschmidt, and Rishabh Singh. 2018. Execution-guided neural program decoding. In *ICML workshop on Neural Abstract Machines and Program Induction v2 (NAMPI)*.

David HD Warren and Fernando CN Pereira. 1982. An efficient easily adaptable system for interpreting natural language queries. *Computational Linguistics*, 8(3-4):110–122.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Lina Maria Rojas-Barahona, Pei-Hao Su, Stefan Ultes, David Vandyke, and Steve J. Young. 2016. A network-based end-to-end trainable task-oriented dialogue system. *CoRR*.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263. Association for Computational Linguistics.

Chien-Sheng Wu, Andrea Madotto, Ehsan Hosseini-Asl, Caiming Xiong, Richard Socher, and Pascale Fung. 2019. Transferable multi-domain state generator for task-oriented dialogue systems. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

Kun Xu, Lingfei Wu, Zhiguo Wang, Mo Yu, Liwei Chen, and Vadim Sheinin. 2018. Sql-to-text generation with graph-to-sequence model. *EMNLP*.

Xiaojun Xu, Chang Liu, and Dawn Song. 2017. Sqlnet: Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv:1711.04436*.

Navid Yaghmazadeh, Yuepeng Wang, Isil Dillig, and Thomas Dillig. 2017. Sqlizer: query synthesis from natural language. *Proceedings of the ACM on Programming Languages*.

Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. 2018a. Typesql: Knowledge-based type-aware neural text-to-sql generation. In *Proceedings of NAACL*. Association for Computational Linguistics.

Tao Yu, Michihiro Yasunaga, Kai Yang, Rui Zhang, Dongxu Wang, Zifan Li, and Dragomir Radev. 2018b. Syntaxsqlnet: Syntax tree networks for complex and cross-domain text-to-sql task. In *Proceedings of EMNLP*. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018c. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Tao Yu, Rui Zhang, Michihiro Yasunaga, Yi Chern Tan, Xi Victoria Lin, Suyi Li, Irene Li Heyang Er, Bo Pang, Tao Chen, Emily Ji, Shreya Dixit, David Proctor, Sungrok Shim, Vincent Zhang Jonathan Kraft, Caiming Xiong, Richard Socher, and Dragomir Radev. 2019. Sparc: Cross-domain semantic parsing in context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy. Association for Computational Linguistics.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *AAAI/IAAI*, pages 1050–1055, Portland, OR. AAAI Press/MIT Press.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *UAI*.

Rui Zhang, Tao Yu, He Yang Er, Sungrok Shim, Eric Xue, Xi Victoria Lin, Tianze Shi, Caiming Xiong, Richard Socher, and Dragomir Radev. 2019. Editing-based sql query generation for cross-domain context-dependent questions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing*. Association for Computational Linguistics.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.

Victor Zhong, Caiming Xiong, and Richard Socher. 2018. Global-locally self-attentive encoder for dialogue state tracking. In *ACL*.

# A Appendices

This section provides description of dialog actions in A.1, more details on baseline modifications and hyperparameters in A.2, system response guides in A.3, additional dialog examples in CoSQL dataset in Figure 10 and 11, and the DB user (AMT turkers) and the SQL expert (college computer science students) annotation interfaces in Figure 6, 7, 8, and 9.

## A.1 Description of Dialog Acts

For the DB user, we define the following dialog acts:

**`INFORM_SQL`** The user informs his/her request if the users question can be answered by SQL. The system needs to write SQL.

**`INFER_SQL`** If the users question must be answered by SQL+human inference. For example, users questions are are they..? (yes/no question) or the 3rd oldest.... SQL cannot directly (or unnecessarily complicated) return the answer, but we can infer the answer based on the SQL results.

**AMBIGUOUS** The users question is ambiguous, the system needs to double check the user's intent (e.g. what/did you mean by...?) or ask for which columns to return.

**AFFIRM** Affirm something said by the system (user says yes/agree).

**NEGATE** : Negate something said by the system (user says no/deny).

**NOT_RELATED** The users question is not related to the database, the system reminds the user.

**CANNOT_UNDERSTAND** The users question cannot be understood by the system, the system asks the user to rephrase or paraphrase question.

**CANNOT_ANSWER** The users question cannot be easily answered by SQL, the system tells the user its limitation.

**GREETING** Greet the system.

**GOOD_BYE** Say goodbye to the system.

**THANK_YOU** Thank the system.

For the system, we define the following dialog acts:

**CONFIRM_SQL** The system creates a natural language response that describes SQL and result table, and asks the user to confirm if the system understood his/her intention.

**CLARIFY** Ask the user to double check and clarify his/her intention when the users question is ambiguous.

**REJECT** Tell the user you did not understand/cannot answer his/her question, or the user question is not related.

**REQUEST_MORE** Ask the user if he/she would like to ask for more info.

**GREETING** Greet the user.

**SORRY** Apologize to the user.

**WELCOME** Tell the user he/she is welcome.

**GOOD_BYE** Say goodbye to the user.

### A.2 Modifications and Hyperparameters for Baselines

**CD-Seq2Seq** We apply the model with the same settings used in SParC without any changes.

**SyntaxSQL-con** We apply the model with the same settings used in SParC without any changes.

**Template-based** We first create a list of SQL query patterns without values, column and table names that cover the most cases in the train set of CoSQL. And then we manually changed the patterns and their corresponding responses to make sure that table, column, and value slots in the responses have one-to-one map to the slots in the SQL query. Once we have the SQL-response mapping list, during the prediction, new SQL statements are compared with every templates to find the best template to use. A score will be computed to represent the similarity between the SQL and each template. The score is computed based on the number of each SQL key components existing in the SQL and each template. Components of the same types are grouped together to allow more flexible matching, like count, max, min are grouped to aggregate. A concrete example of templates is shown: `SELECT column0 FROM table0 WHERE column1 comparison0 value0.` `column0,1` and `table0` represent column name and table name respectively. `comparison0` represents one of the comparison operator including `>=, <=, <,>,=,!=, and like.` `value0` represents a value the user uses to constrain the query result.

**Seq2Seq** We train a word2vec embedding model on the concatenation of the SQL query and response output of the training data for the embedding layer of our Seq2Seq model. We use an embedding dimension of 128, hidden dimension of 256, a single-layer bi-directional LSTM encoder and uni-directional LSTM decoder with attention. We use a batch size of 32, clip the norm of the gradient at 2.0, and do early stopping on the validation loss with a patience of 5. We perform decoding with greedy search.

**Pointer-generator** We follow the same settings as in the Seq2Seq case with the addition of the copy mechanism during training and testing.

**TBCNN-pair** The model is modified mainly on the sentence embedding part and classifier part. The input of the modified model is a user utterance and the related column names. Therefore, we replace one of the two sentence embedding modules with a database column name encoding module, which generates representations of the col-

umn names related to the sentence. The classifier is modified by adding a label(user dialogue act) number predicting module, which predicts the number of the labels(user dialogue acts) of the user utterance. The label number prediction module is similar to the column number prediction module in SQLNet.

### A.3 System Response Guide

System response should be standard and professional. We follow the rules below to write responses for different system dialog action type:

**`CLARIFY`** "Did you mean...?", "What did you mean by...?", or anything similar.

**`REJECT`** "Sorry, I don't have the answer to your question..." or anything similar.

**`REQUEST_MORE`** "Do you want anything else?" or anything similar.

**`CONFORM_SQL`** We convert SQL written by us back to natural language. (We should use the column names and values in the SQL). Our response has to describe all information in the SQL independently instead of referring to any previous context or subject.

1. If the returned result can be combined with the SQL description, combine them together to generate the response. For example:

   Given SQL:
   SELECT AVG(SALARY) FROM INSTRUCTOR
   Result Returned: 200k
   Your Response:"The average salary of all instructors is 200k."

2. If the returned result is too large and cannot be combined with the SQL description, describe them separately. For example:

   Given SQL:
   SELECT AVG(T1.SALARY),T1.DEPARTMENT_ID FROM INSTRUCTOR AS T1 JOIN DEPARTMENT AS T2 ON T1.DEPARTMENT_ID = T2.ID GROUP BY T1.DEPARTMENT_ID
   Result Returned: a long table
   Your Response:"Here is the result table that shows the average salary in each department. For example, the average of CS professors is 250k."

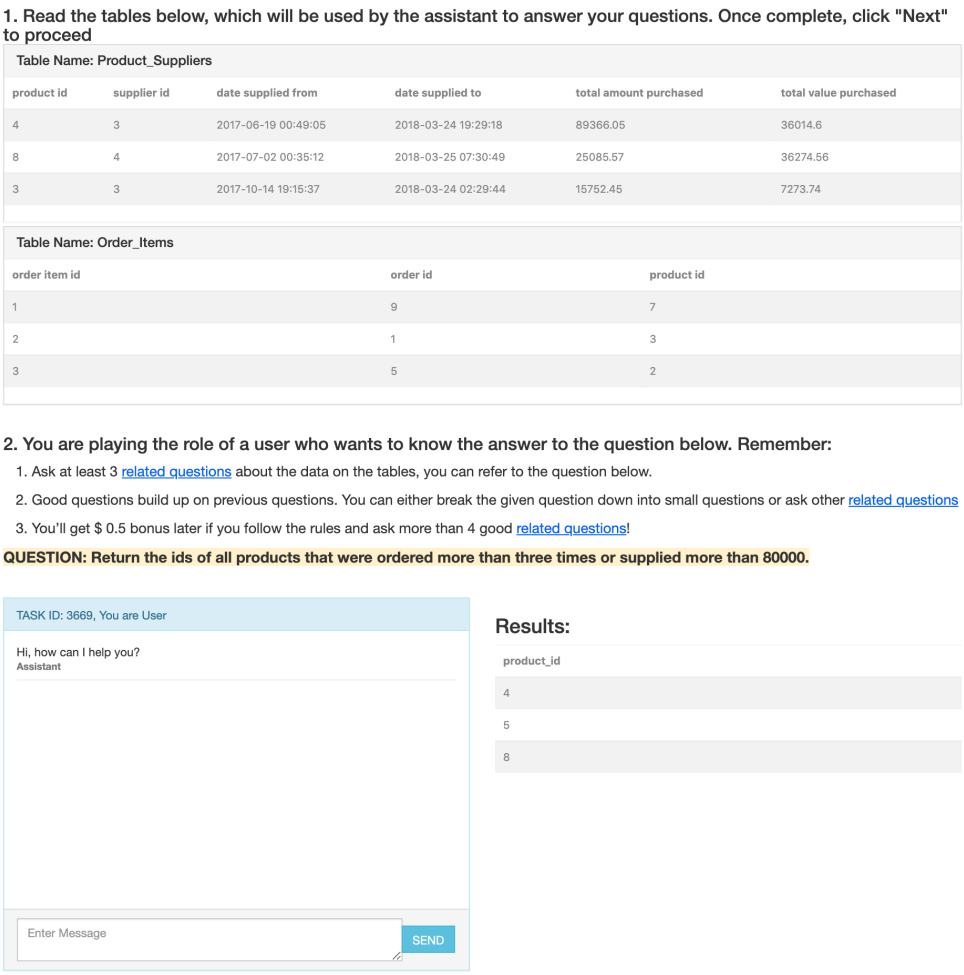


Figure 6: DB User Interface

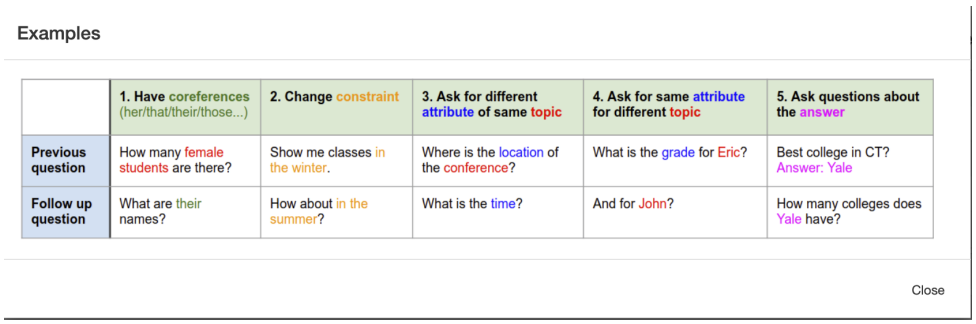


Figure 7: DB User Related Questions: a pop-up window when the user clicks highlighted ”related questions” in the above interface.

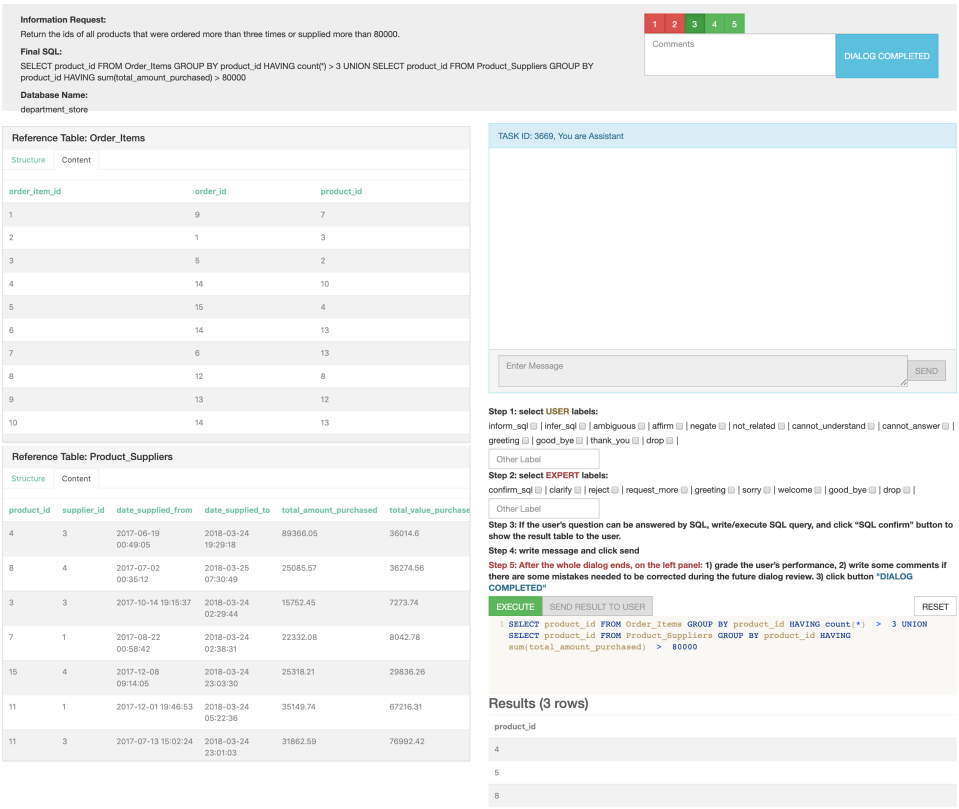


Figure 8: SQL Expert Interface

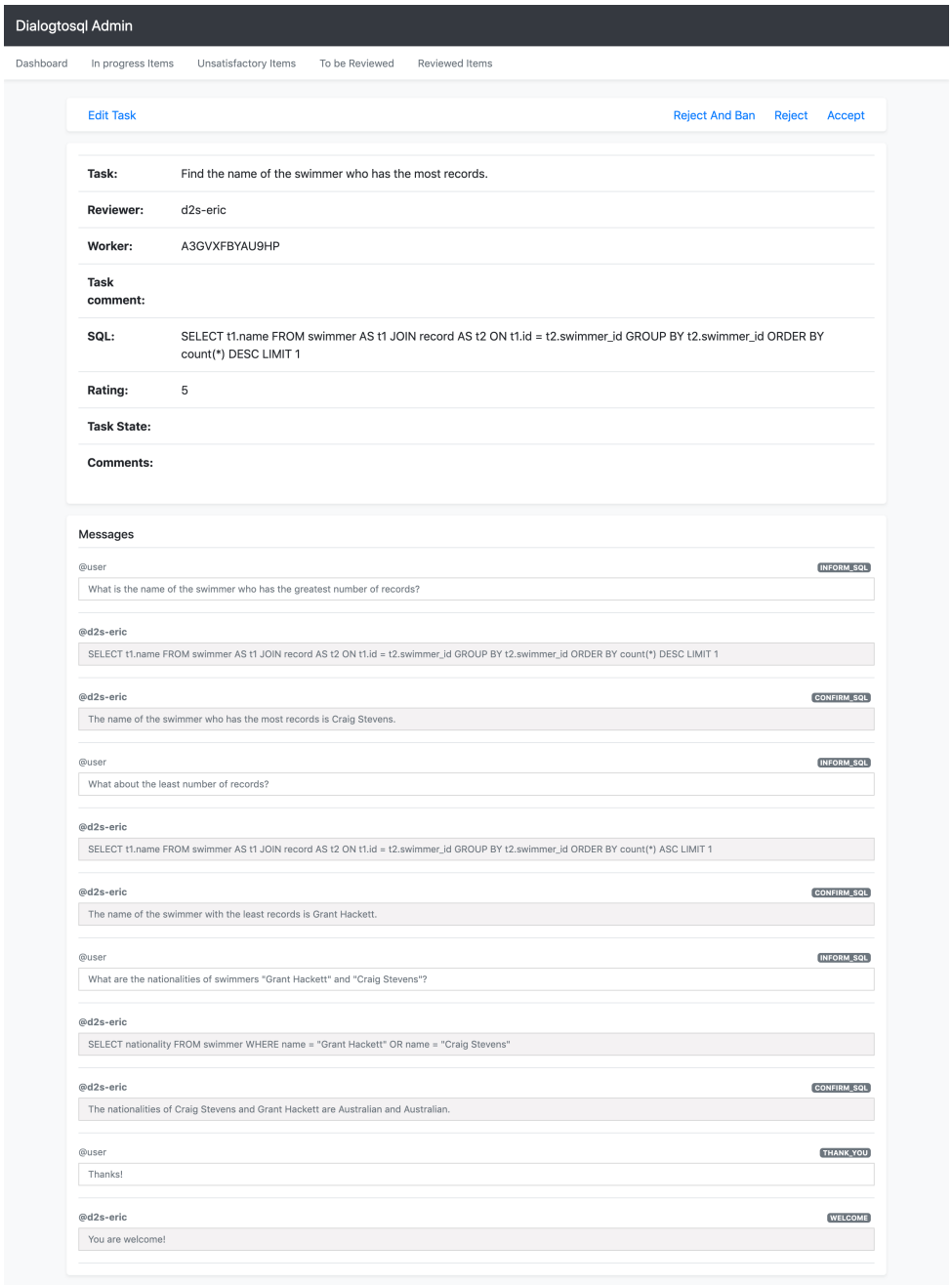


Figure 9: Dialogue Review Interface

$D_1$ : Database about soccer containing 3 tables

...

$Q_2$ : What are their names? `INFORM_SQL`

$S_2$ : `SELECT cName FROM College WHERE enr > 18000`

$A_2$ : OU | FSU

$R_2$ : The names of the colleges that have enrollment values greater than 18000 are OU and FSU `CONFIRM_SQL`

$Q_3$ : Which one has the higher enrollment? `INFER_SQL`

$S_3$ : `SELECT cName FROM College WHERE enr > 18000 ORDER BY enr DESC LIMIT 1`

$A_3$ : OU

$R_3$ : Among the results, OU has a higher enrollment. `CONFIRM_SQL`

...

Figure 10: Part of a dialogue example with INFER_SQL user dialog label

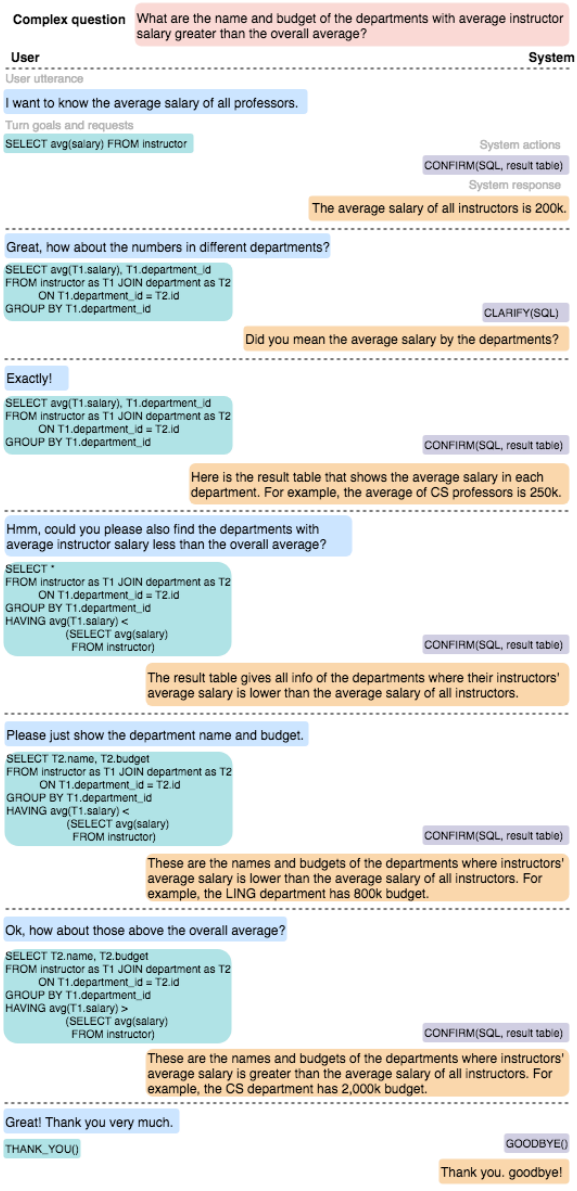


Figure 11: A complete dialogue example

# A Practical Dialogue-Act-Driven Conversation Model for Multi-Turn Response Selection

**Harshit Kumar**
IBM Research,
Delhi, India
harshitk@in.ibm.com

**Arvind Agarwal**
IBM Research,
Delhi, India
arvagarw@in.ibm.com

**Sachindra Joshi**
IBM Research,
Delhi, India
jsachind@in.ibm.com

## Abstract

Dialogue Acts play an important role in conversation modeling. Research has shown the utility of dialogue acts for the response selection task, however, the underlying assumption is that the dialogue acts are readily available, which is impractical, as dialogue acts are rarely available for new conversations. This paper proposes an end-to-end multi-task model for conversation modeling, which is optimized for two tasks, dialogue act prediction and response selection, with the latter being the task of interest. It proposes a novel way of combining the predicted dialogue acts of context and response with the context (previous utterances) and response (follow-up utterance) in a crossway fashion, such that, it achieves at par performance for the response selection task compared to the model that uses actual dialogue acts. Through experiments on two well known datasets, we demonstrate that the multi-task model not only improves the accuracy of the dialogue act prediction task but also improves the MRR for the response selection task. Also, the cross-stitching of dialogue acts of context and response with the context and response is better than using either one of them individually.

## 1 Introduction

Response selection remains at the core of conversation modeling, with the objective of selecting an appropriate response utterance from a set of candidate utterances, for a given conversation history consisting of previous utterances (context). Decades of research in this task includes traditional methods such as (Kitano, 1991; Ritter et al., 2011) and recent deep learning based methods (Ji et al., 2014; Chaudhuri et al., 2018; Xu et al., 2018; Chen et al., 2017; Song et al., 2018; Lowe et al., 2015; Zhao et al., 2018; Wen et al., 2016). Underlying these methods, a fundamental need is to capture the semantics of the context and use it for selecting the appropriate response. While the context provides essential clues as to what could be a follow-up response, research (Kumar et al., 2018) has further shown that any additional information available in the form of dialogue acts can also be helpful for response selection. Such information when used along with the context improves the performance of the response selection task. However, the above method assumes that dialogue acts are available at the time of response selection, which is rarely the case —as dialogue acts are usually not available for new conversations in a live setting— thus making them impractical for practitioners. In this paper, we propose a novel model that bridges this gap between theory and practice. In other words, our proposed model leverages the dialogue acts for response selection, as well as is practical.

In the literature, researchers (Kumar et al., 2018; Xu et al., 2018; Zhao et al., 2017) have proposed deep learning models that use actual dialogue acts in conversation modeling. While actual dialogue acts help in response selection, a natural question is, can we build a system that eliminates the dependency on *actual* dialogue acts at the time of response selection, and rather predict them as an integral part of the model? Second, and a more important question is: Is such a system going to be helpful in response selection, because the dialogue acts predictions will have some error in it, i.e., the underlying prediction model would not be 100% accurate in its predictions? And, if the answer to the second question is positive, then what is the gap - in terms of performance - between the proposed system that uses predicted dialogue acts and the system that uses actual dialogue acts? In this paper, we answer all of the above questions: our proposed model is a multi-task model that has dialogue acts prediction as an integral part of it,



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1980–1989,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

i.e., it does not need the actual dialogue acts to select an appropriate response, rather it predicts the dialogue acts and use them for response selection. Furthermore, our model is by design robust to the errors in dialogue act prediction; our novel way of combining dialogue acts of context and response, is able to compensate for the errors in dialogue act predictions, and performs on par with the model that uses actual dialogue acts.

The main contributions of this paper are as follows:

- We model the task of response selection as a multi-task learning problem, with the objective of performing two tasks in a single end-to-end model: first, learn to predict the dialogue acts of utterances (context and response), and second, use the previous utterances (context) and the predicted dialogue acts of both the context and the response to select a response from a given set of candidate responses.

- While modeling the response selection conditioned on the dialogue acts of the context helps (Zhao et al., 2017), an important contribution is the additional utility of the dialogue act of the response. Our simple yet novel way of combining the dialogue act representations of the context and response with the utterance representations of the context and response promotes cross similarities, and thereby bring in ensemble characteristics in the model. That is, the ensemble model outperforms all other non-ensemble models, and is robust to the errors made by any underlying components of the ensemble.

- We evaluate the proposed model on two dialogue datasets, DailyDialog(Li et al., 2017) and Switchboard Dialogue Act Corpus (SwDA (Jurafsky, 1997)), and show that having dialogue act prediction as an integral part of the model improves the performance of the response selection consistently across both datasets. An important observation is the significant performance boost obtained from the proposed Crossway model (ensemble-model); that is, it not only improves the MRR for the response selection task but also improves the accuracy of the dialogue act prediction task.

## 2 Approach

This section details our approach, i.e. an end-to-end multi-task model for response selection (task1) using predicted dialogue acts (task2) of context and response. For response selection, there are two frameworks that are popular in the literature; one is generative and the other is discriminative. In the generative framework, sequence-to-sequence kind of model is used. It is trained to *generate* an appropriate response given a context. On the other hand, a discriminative model is trained such that among a set of $K$ candidate responses, the correct response has the highest similarity with the context. Since discriminative model is superior than their generative counterpart(Liu et al., 2016), we use discriminative model as a base model.

Before proceeding, we first define the mathematical notation that we use throughout this work. Let $\mathcal{D} = (C^1, C^2, \ldots C^N)$ be a set of $N$ conversations, with $(Y^1, Y^2, \ldots Y^N)$ be their corresponding actual DAs. Each conversation $C^i$ is a sequence of $R_i$ utterances $C^i = (u^i_1, u^i_2, \ldots u^i_{R_i})$ with $Y^i = (y^i_1, y^i_2, \ldots y^i_{R_i})$ being the corresponding actual DAs. For the notational simplicity, we shall ignore the conversation superscript $i$ which should be clear from the context. For each utterance $u_j$ in each conversation, we have an associated DA label $y_j \in \mathcal{Y}$, where $\mathcal{Y}$ is the set of all possible DAs. Each utterance $u_j$ is a sequence of $S_j$ words stringed together, i.e., $u_j = (w_{j1}, w_{j2}, \ldots w_{jS_j})$. In our problem setting, the first $R_{i-1}$ utterances in a conversation $C^i$ form the context, i.e. $context^i = (u^i_1, u^i_2, \ldots u^i_{R_i-1})$; and, the last utterance, $u^i_{R_i}$ is the true response. An illustration of the multi-task model for dialogue act prediction and response selection is illustrated in Figure 1.

Our approach of joint modeling of dialogue act prediction task and response selection task, share a common encoder which encodes the conversation context and response. These representations are then used to predict the dialogue acts and to find the right response from a set of candidate responses. In the following subsections, we provide details of this shared encoder, dialogue act prediction modeling, and response selection modeling.

### 2.1 Shared Context-Response Encoder

In each conversation, the whole sequence of utterances that constitutes a conversation can be con-

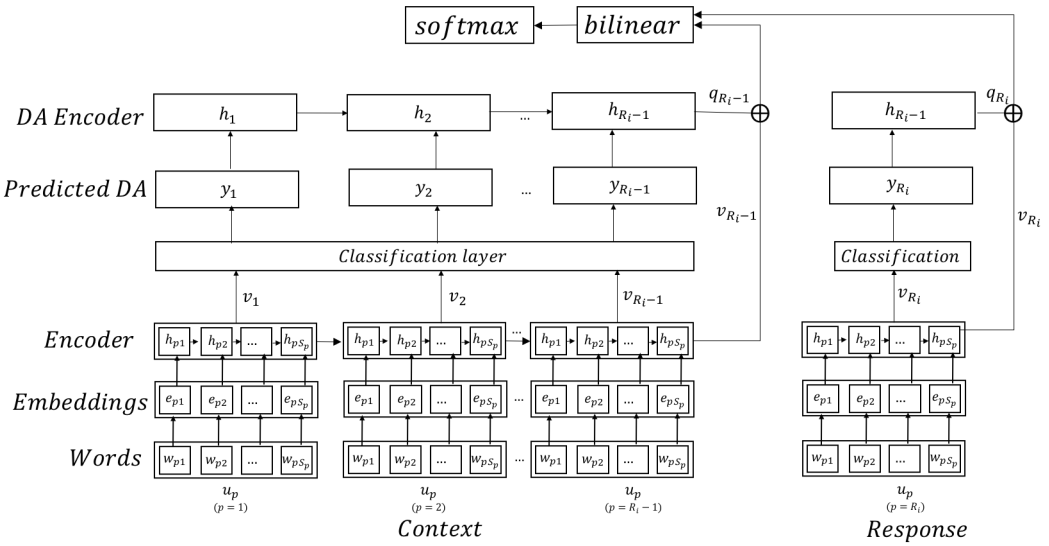


Figure 1: Architecture diagram of the proposed multi-task Crossway model for response selection and dialogue prediction

sidered as a single very long chain of words. This is input to an RNN encoder to obtain a single *unified* representation of the context (and response), and a representation of each utterance in the context (response). Given a conversation consisting of $R_i$ utterances, with each utterance $u_j$ consisting of $S_j$ words, the sequence of operation used in encoder is as follows:

$$e_{jk} = f^1_{embed}(w_{jk}) \quad \forall j \in 1, \dots R_i, \forall k \in 1, \dots S_j$$
$$h_{jk} = f^1_{rnn}(h_{jk-1}, e_{jk}) \quad \forall j \in 1, \dots R_i, \forall k \in 1, \dots S_j \tag{1}$$

where, $f^1_{embed}$ represents the embedding layer, whereas $f^1_{rnn}$ is the encoder (RNN). The representation of each utterance $u_j$, denoted by $v_j$, can be obtained by combining the representations of its constituent words. We take the representation of the last time-step of the encoder as the representation of the entire utterance, i.e. $v_j = h_{jS_j}$. This is because the final time-step contains the context of all the words preceding it, and serves as a good approximation to the representation of the entire utterance. Thus the shared encoder finally gives us the representations of each utterance, i.e. $v_1, v_2, \dots v_{R_i-1}$, corresponding to the $context^i$ consisting of utterances $u_1, u_2, \dots u_{R_i-1}$, with $v_{R_i-1}$ being the representation of the entire context. Since the encoder is shared between context and response, it is also used to get the representation $v_{R_i}$ corresponding to response utterance $u_{R_i}$.

## 2.2 Dialogue Act Prediction Model

Dialogue act prediction (Task1) is a multi-class classification problem, where the goal is to assign a dialogue act to each utterance in a conversation. Following the recent advances in sequence prediction task (Kumar et al., 2017), the dialogue act prediction model is built on top of an RNN network, where each utterance's representation is first obtained using an RNN Encoder ( 2.1), which is then input to a classification layer to predict the appropriate dialogue act of that utterance. Given the representation of an utterance obtained using shared context-response encoder, the probability of predicting the dialogue act of the utterance $u^i_k$ can be written as:

$$p(y|v_k) = \frac{\exp(-W_y^T v_k)}{\sum_{y' \in \mathcal{Y}} \exp(-W_{y'}^T v_k)} \tag{2}$$

where $v_k$ is the encoded representation of utterance $u_k$. $W_y$ is the weight vector associated with the class $y$. The network is optimized to maximize the probability of the gold-standard (actual) dialogue act. For the dialogue acts associated with the utterances in the context, the loss function can be written as following:

$$\mathcal{L}_c = \sum_{C^i \in \mathcal{D}} \sum_{u^i_k \in C_i \setminus u^i_{R_i}} -\log p(y^i_k | v^i_k) \tag{3}$$

where $y^i_k$ is the actual dialogue act of the utterance $u^i_k$. $\mathcal{L}_c$ is the loss (i.e., negative log likelihood) computed from the prediction task of the context. We can compute a similar loss for the response as

following:

$$\mathcal{L}_r = \sum_{C^i \in D} -\log p(y^i_{R_i} | v^i_{R_i}) \quad (4)$$

where $v^i_{R_i}$ is representation of the response utterance $u^i_{R_i}$ obtained from the encoder, and $y^i_{R_i}$ is the corresponding actual dialogue act.

### 2.3 Dialogue-Act Aware Response Selection

The goal of the second task is to select the true candidate response from a set of candidate responses for a given context. This model consists of two modules, the first module is a Dialogue-Act Encoder which gives us two representations: a compositional representation for the sequence of dialogue acts associated with the context; and, a representation for the response dialogue act. The second module is a crossway response selection module which uses both dialogue act representations to select the right response from a set of candidate responses. This module combines the dialogue act representation and utterance representation of context utterances and response in a cross-stitched way using a Siamese network for the response selection task.

#### 2.3.1 Dialogue-Act Encoder Module

In conversation modelling, dialogue acts are treated as an additional sequence of signals that can aid in the learning process. Dialog-Act encoder (DA-encoder), which is based on the same principle as the RNN encoder, takes the sequence of dialogue acts and returns a representation of that sequence. The input to the DA-encoder are one hot encodings of the dialogue acts, which are then passed through an embedding layer ($f^2_{embed}$) to learn DA embeddings. These DA embeddings are sent to an RNN ($f^2_{rnn}$) to learn a representation of the entire DA sequence. For a given sequence of DA of length $K$, the sequence of operations for the DA-encoder is as follows:

$$\begin{aligned} e_k &= f^2_{embed}(y_k) \quad \forall k \in 1, 2, \dots K \\ h_k &= f^2_{rnn}(h_{k-1}, e_k) \quad \forall k \in 1, 2, \dots K \\ q_K &= h_K \end{aligned} \quad (5)$$

where, $q_K$ is the final representation of the dialogue act sequence.

#### 2.3.2 Crossway Response Selection Module

The Crossway Response Selection Module uses the shared context-response encoder to get the representations of the utterances in a context, i.e. $v_{R_i-1}$, and the response, i.e. $v_{R_i}$; and DA encoder to get the representation of the DA sequence associated with the context, i.e $q_{R_i-1}$ and response, i.e. $q_{R_i}$. A typical discriminative model, or in particular Siamese model, consists of two encoders, one encoder encoding the context, while another encoding the response utterance. These two representations are passed to a final layer that computes the probability of candidate being a valid response given the context. In the previous response selection models that use dialogue acts, authors have only used the dialogue act representations of the context and not of the response. We use all four representations in a Crossway fashion, i.e. $v_{R_i-1}$, $v_{R_i}$, $q_{R_i-1}$ and $q_{R_i}$. As we shall see in the experiments, using these four representation adds robustness to the Crossway model. The two representations corresponding to context and its DA sequence, $v_{R_i-1}$ and $q_{R_i-1}$, are concatenated together to obtain a compositional representation of the context. Similarly, the two representations of the response and its associated dialogue act, $v_{R_i}$ and $q_{R_i}$ are concatenated together to obtain a compositional representation of the response utterance. The probability of the association of these representations can be computed using a bilinear function as following:

$$\begin{aligned} d_{R_i-1} &= [v_{R_i-1}, q_{R_i-1}] \\ d_{R_i} &= [v_{R_i}, q_{R_i}] \\ p(s|d_{R_i-1}, d_{R_i}) &= \sigma(d^T_{R_i-1} \cdot A \cdot d_{R_i} + b) \end{aligned} \quad (6)$$

where, the bias $b$ and matrix $A$ are learned model parameters. The model is trained by minimizing the cross-entropy of all labeled conversations including positive and negative examples. Let $\mathcal{D}^-$ be the variation of $\mathcal{D}$ where response utterance $u^i_{R_i}$ is replaced with some random utterance in order to create negative examples. Given the set of positive and negative conversation sets, the loss is computed as follows:

$$\begin{aligned} \mathcal{L}_s = &- \sum_{C^i \in \mathcal{D}} \log p(s^i = 1 | d_{R_i-1}, d_{R_i}) \\ &- \sum_{C^i \in \mathcal{D}^-} \log p(s^i = 0 | d_{R_i-1}, d_{R_i}) \end{aligned} \quad (7)$$

where $s^i$ is 1 for $C^i \in \mathcal{D}$ and $s^i$ is 0 for $C^i \in \mathcal{D}^-$. At the test time, each conversation has a context followed by a set of $n$ candidates responses. The system is tested in its ability to assign a higher score to the true response.

## 3 Multi-task Crossway Model

Dialogue acts have been shown to be useful for response selection task (Kumar et al., 2018). These dialogue acts can either be given to us or can be predicted using an external model. When dialogue acts are given to us, we denote this model as Siamese-ADA+Crossway. The assumption that the dialogue acts would be available at the test time is rather impractical, therefore an alternate way of leveraging dialogue acts is to predict them. We call this model as Siamese-PDA single-task (Siamese-PDA-ST+Crossway), since dialogue act prediction task is trained independent of the response selection task. In this work we hypothesize that joint modeling of dialogue act prediction task and response selection task would be more beneficial than modeling them individually. Under this hypothesis, we propose a multi-task model, Siamese-PDA-MT+Crossway, that uses the same shared context-response encoder for both tasks. For the dialogue act prediction task of the context and the response, the representations obtained from the shared encoder are input to the classification layer (Section 2.2). The loss is computed as the negative likelihood of predicting the correct dialogue acts for each utterance in the context and the response. The dialogue act prediction loss associated with the context and response are given in Equations (3) and (4) respectively. The response selection task in the multi-task learning setting also uses the same representations as used by the dialogue act prediction task, i.e. those obtained from the shared encoder (Section 2.3). The loss of the response selection task is given in Equation (7). The final loss of the end-to-end multitask model is the combined loss of both the tasks, i.e.

$$\mathcal{L} = \mathcal{L}_c + \mathcal{L}_r + \mathcal{L}_s$$

## 4 Experiments

In this section, we provide details of the experiments, i.e. dataset and its preparation, baseline models, experimental setup, results and their analysis including ablation study.

### 4.1 Datasets

While our model does not need the actual dialogue acts at test time, it does require them at the time of training. So in our problem setting, we require a dataset that is of reasonable size and has utterances annotated with the corresponding dialogue acts. We considered several available datasets (Serban et al., 2015), such as DailyDialog (Li et al., 2017), SwDA (Switchboard Dialogue Act Corpus (Jurafsky, 1997)), MRDA (Meeting Recorder Dialogue Act corpus (Janin et al., 2003)), Ubuntu(Lowe et al., 2015), OpenSubtitles(Tiedemann, 2009), etc. Out of these, Ubuntu, OpenSubtitles, MRDA were found to be unsuitable for our problem setting. The first two, i.e. Ubuntu and OpenSubtitles, do not have dialogue act annotations, and the MRDA corpus is too small, it has only 51 conversations. Therefore, we evaluate the performance of our model on SwDA and DailyDialog datasets:

- SwDA: Switchboard Dialogue Act Corpus (Stolcke et al., 2000) is annotated on 1155 human to human telephonic conversations. Each utterance in a conversation is labeled with one of the 42-class compact DAMSL taxonomy (Core and Allen, 1997; Jurafsky, 1997). The dataset has train, validation, and test splits of 1003, 12, and 19 conversations, respectively.

- DailyDialog(Li et al., 2017) consists of utterances annotated with dialogue acts and is large enough for conversation modeling methods to work. Each utterance is annotated with one of the four dialogue acts. The dataset has train, validation, and test splits of 11118, 1000, and 1000 conversations, respectively.

**Dataset Preparation** To prepare the data for training and testing, we followed the procedure mentioned in (Kumar et al., 2018; Lowe et al., 2017, 2015). Examples in our training dataset consists of a context of $K$ utterances, followed by the $K+1$ utterance that acts as a true response (positive examples). The training of response selection models require positive examples and an equal number of negative examples. In-order to prepare negative examples, each conversation is replicated by replacing the last utterance with a random utterance from the rest of the training data. On the other hand, examples in our test dataset consists of a context followed by a set of candidate responses of size $n$. The first response is a true response followed by $n-1$ utterances selected randomly from the rest of the test set. For the DailyDialog and SwDA dataset, the context length $K$ is set to 3 and 10, respectively. For both datasets, the candidate pool consists of 10 utterances. Thus, with

| | DailyDialog | | | | SwDA | | | |
|---|---|---|---|---|---|---|---|---|
| | MRR | R@1 | R@3 | Acc (%) | MRR | R@1 | R@3 | Acc (%) |
| Siamese (Lowe et al., 2017) | 0.867 | 0.818 | 0.880 | NA | 0.615 | 0.463 | 0.687 | NA |
| Siamese-PDA-ST + Crossway | 0.900 | 0.870 | 0.899 | 82.9 | 0.669 | 0.568 | 0.685 | 71.4 |
| Siamese-PDA-MT + Crossway | **0.946** | **0.938** | **0.938** | **86.1** | **0.703** | **0.612** | **0.712** | **72.4** |
| Siamese-ADA + Crossway (upper bound) | 0.956 | 0.944 | 0.955 | 100 | 0.719 | 0.630 | 0.738 | 100 |

Table 1: Results for DailyDialog and SwDA Datasets, PDA is for Predicted Dialogue Act, ADA is for Actual Dialogue Act, ST is Single-Task Learning, and MT is for Multi-Task Learning

this data preparation exercise, the total number of conversations in train, test, and validation for DailyDialog are 61030, 2849, and 2695, respectively. And, for the SwDA dataset, the total number number of conversations in train, test, and validation split are 178736, 3483, and 2245, respectively.

### 4.2 Hyper-parameter Tuning

The validation set is used for fine tuning hyper-parameters, and results are reported on the test set. The maximum batch size is 32; for each batch, the utterances are padded to the maximum length in that batch. We use 300-dimensional Glove embeddings (Pennington et al., 2014) to initialize the word vectors – these word vectors are also updated during training. Both Context-Response encoder and DA-Encoder are GRUs with $rnn\_size$ of 300, after optimizing between 100 to 500 in step of 100. Dropout of 0.1 (optimized over 0.0 to 0.7 in steps of 0.1) was applied to embeddings obtained from the output of the encoder. Models were trained to minimize cross entropy using Adam optimizer with learning rate of 0.0003 (optimized over 0.0001, 0.0003, 0.0005, 0.0007, 0.001). All models were trained for 200 epochs.

### 4.3 Evaluation Metrics

Since our problem formulation is retrieval based, we use standard IR metrics such as Mean Reciprocal Rank (MRR) and Recall@k as our evaluation metrics for the response selection task (Task-2). MRR is calculated as the mean of the reciprocal rank of the true candidate response among other candidate responses. Recall@$k$ measures whether the true candidate response appears in a ranked list of $k$ responses. While we report all of these metrics, in order to make the analysis more explainable, we will keep the MRR as our primary metric. We also report the accuracy of the dialogue act prediction task (Task-1).

### 4.4 Baseline Methods and Proposed Models

Following is the list of baseline model and proposed models that we use in our experiments:

- **Siamese** (Lowe et al., 2017): a siamese model that uses a dual encoder for conversation modeling without any dialogue acts information.
- **Siamese-PDA-ST+Crossway**: model that uses dialogue acts in single-task setting, (i.e. predicted externally) in a crossway fashion. PDA is for Predicted Dialogue Act and ST is for Single-Task.
- **Siamese-ADA+Crossway**: a hypothetical model that uses actual dialogue acts in a crossway fashion (upper bound). ADA is for Actual Dialogue Act
- **Siamese-PDA-MT+Crossway**: the proposed model uses predicted dialogue acts in a crossway fashion in a multi-task setting. MT is for Multi-Task.
- **Siamese-PDA-MT+Context-DA** (Zhao et al., 2017) : this model uses predicted dialogue acts of the context in a multi-task setting, we implemented this model for the discriminative response selection task.

### 4.5 Results and Discussion

In Tables 1 and 2, we report results of our experimental study, providing evidences to support two hypotheses:

1. The joint modeling of dialogue act prediction task and response selection task (multi-task setting) performs better than modelling them independently (single-task setting).
2. Combining the dialogue acts of response and context (Crossway) performs better than using either one of them.

| | DailyDialogue | | | SwDA | | |
|---|---|---|---|---|---|---|
| | Context-DA (Zhao et al., 2017) | Response-DA | Crossway | Context-DA (Zhao et al., 2017) | Response-DA | Crossway |
| Siamese-PDA-ST | **0.912** | 0.900 | 0.900 | 0.639 | 0.649 | **0.669** |
| Siamese-PDA-MT (Zhao et al., 2017) | 0.921 | 0.919 | **0.946** | 0.698 | 0.685 | **0.703** |
| Siamese-ADA (Kumar et al., 2018) | 0.934 | 0.927 | **0.956** | 0.656 | 0.682 | **0.719** |

Table 2: Comparison of MRR when using dialogue acts of Context-only, Response-only and Crossway fashion

**Multi-Task vs Single-Task Modelling:** In Table 1, we report and compare the results of our proposed method with the baselines, for both datasets, i.e. DailyDialog and SwDA, and provide evidence for the first hypothesis outlined above. From these results, we draw several observations. First observation is that all models that use dialogue acts outperform the model that does not use them. The second observation is that the multi-task model (Siamese-PDA-MT+Crossway), that does the joint modeling of both tasks (dialogue act prediction and response selection task), performs better than the single-task model (Siamese-PDA-ST+Crossway) that models them separately. Multi-task modelling not only improves the MRR in the response selection task for both datasets, but also achieves better dialogue act prediction accuracy. Siamese-ADA+Crossway model which uses the actual dialogue acts, is an upper bound (therefore an ideal model) on how good any model can perform if it were to use predicted dialogue acts. And as we can see, the MRR of multi-task model (Siamese-PDA-MT+Crossway) is close to the upper bounds for both datasets as compared to the single-task model (Siamese-PDA-ST+Crossway). An interesting observation is that, for both DailyDialog and SwDA dataset, though the multi-task model has less than ideal dialogue act prediction accuracy (less than 100%), it performs at par with the ideal model for the response selection task. For the DailyDialog dataset, the multi-task model has the dialogue act prediction accuracy of 86.1%, much less than the ideal accuracy of 100%; in-spite of that, it performs at par with the ideal model that uses the actual dialogue acts, i.e. Siamese-ADA+Crossway (MRR of 0.946 with Siamese-PDA-MT+Crossway vs 0.956 with Siamese-ADA+Crossway). Similarly, for the SwDA dataset, the MRR with multi-task model (Siamese-PDA-MT+Crossway) is 0.703, which is very close to the MRR of 0.719 obtained with the ideal model (Siamese-ADA+Crossway). Consistency of such results across both datasets suggests that the Crossway model is robust and is able to compensate for the errors in predictions by leveraging the similarities across dialogue acts and context/response. In the follow up section, we analyze the effect of Crossway in much more detail.

**Crossway vs Response-DA/Context-DA:** Although the dialogue acts have been shown to be useful for the response selection task, existing work has only used the dialogue acts of the context. Whereas, in our experiments, we have found that the model that uses the dialogue acts of both context and response outperforms the models that use the dialogue acts of either context or response. To further analyze the results, we perform an ablation study and show the results of using the dialogue acts of context, response and of both. In Table 2, we report the MRR numbers of several models that use the dialogue acts in different settings. More specifically, we show how the following models i.e., Siamese with actual dialogue act (Siamese-ADA), Siamese with predicted dialogue acts in single-task setting (Siamese-PDA-ST) and Siamese with predicted dialogue act in multi-task setting (Siamese-PDA-MT) perform when they are given the dialogue acts of only context (Context-DA), dialogue acts of only response (Response-DA), and dialogue acts of both (Crossway). Results in Table 2 indicate that the Crossway always outperforms the Context-DA or the Response-DA, for both datasets. For DailyDialog dataset, Context-DA performs better than Response-DA for all three models, whereas in the SwDA dataset, Response-DA does a relatively better job than context-DA (two out of three models). Despite their different behavior for different datasets, when we combine Response-DA and Context-DA in a Crossway fashion, it outperforms the both, giving the best of both worlds. This performance improvement of the Crossway over context-DA and response-DA can also be attributed to the

way Crossway model works. Note that in the

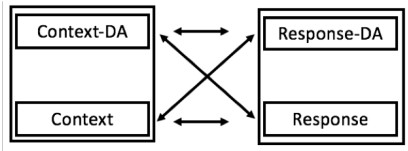


Figure 2: Four implicit similarities being used in Crossway model

Crossway model, there are four similarities that play a role, i.e. context-response, ContextDA-ResponseDA, ContextDA-response and Context-responseDA, graphically depicted in Figure 2. So, in the case of erroneous prediction of either of context DA or response DA, it shall only corrupt two of the four similarities, still leaving two other similarities that can provide strong clues to the underlying model about the correct response belonging to the context.

## 5 Related Work

Researchers have shown that response selection is a promising approach to build a practical conversation system (Gandhe and Traum, 2010; Lowe et al., 2017; Wu et al., 2016). (Gandhe and Traum, 2010) have shown that response selection based approach for conversation modelling is a good approximation of mimicking human dialogue. Response selection based conversation systems are more practical from the implementation perspective because the responses are mined form previous conversation logs and are therefore more natural and semantically correct. (Ji et al., 2014) have used response selection based techniques for modelling short text conversation responses. They conclude that speech act, sentiment or entity associated with the utterances may enhance the accuracy of the underlying model.

Recently, multi-turn response selection has become the focus of conversation modelling. In multi-turn response selection, current utterance including previous $k$ utterances are used to select an appropriate response from a set of candidate responses. (Lowe et al., 2017; Wu et al., 2016) have shown the efficacy of multi-turn response selection in conversation modeling. (Chaudhuri et al., 2018) have further enhanced these models by incorporating additional domain knowledge in the form of domain specific keywords. (Song et al., 2018) have used an ensemble approach (generation-based and selection-based) to build conversational model. Although effective, none of these methods leverage Dialogue Acts for response selection.

The use of Dialogue Acts (DA) (Xu et al., 2018), latent topics(Zhao et al., 2018; Wen et al., 2017), sentiments, entity models can help in grounding or interpretation of the user utterances which can further aid in improvement of conversation modelling. (Kumar et al., 2018) have shown the usefulness of dialogue act for conversation modeling. However, they assume that the dialogue acts are available at the conversation time which is impractical as dialogue acts are rarely available in a real conversation. Our work addresses this limitation and builds an end-to-end dialogue model, where we predict the dialogue acts and use them as an additional signal for response selection. (Xu et al., 2018; Zhao et al., 2017) use dialogue act for conversation modeling, however the focus of their work is on response generation. In addition to the difference in the underlying task, (Xu et al., 2018) uses an in-house dataset with synthetic dialogue acts [1], whereas we have experimented on publicly available datasets. In (Zhao et al., 2017), while authors propose the model for the dialogue generation task, an important difference is that their model uses the dialogue acts of the context whereas our model uses the dialogue acts of both context and response, combined in a cross-way fashion. We however do use this as a baseline, and show our model's superior performance.

## 6 Conclusion

This paper presents an end-to-end multi-task model that eliminates the need of actual dialogue acts at the test time. Our end-to-end model combines the predicted dialogue acts of the context and the response with the context, and use the combined representation to select an appropriate response from a set of candidate responses. Our model has been validated on real-world dialogue datasets; we show that our novel way of combining dialogue acts in a cross-way fashion not only compensates for the errors in the dialogue act prediction model but it performs at par with the response selection model that uses actual dialogue acts.

[1]The dataset and the code is not publicly available to be used as a baseline.

## References


Debanjan Chaudhuri, Agustinus Kristiadi, Jens Lehmann, and Asja Fischer. 2018. Improving response selection in multi-turn dialogue systems. *arXiv preprint arXiv:1809.03194*.

Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. 2017. A survey on dialogue systems: Recent advances and new frontiers. *ACM SIGKDD Explorations Newsletter*, 19(2):25–35.

Mark G Core and James Allen. 1997. Coding dialogs with the damsl annotation scheme. In *AAAI Fall Symposium On Communicative Action In Humans And Machines*.

Sudeep Gandhe and David Traum. 2010. I've said it before, and i'll say it again: an empirical investigation of the upper bound of the selection approach to dialogue. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 245–248. Association for Computational Linguistics.

Adam Janin, Don Baron, Jane Edwards, Dan Ellis, David Gelbart, Nelson Morgan, Barbara Peskin, Thilo Pfau, Elizabeth Shriberg, Andreas Stolcke, et al. 2003. The icsi meeting corpus. In *ICASSP*.

Zongcheng Ji, Zhengdong Lu, and Hang Li. 2014. An information retrieval approach to short text conversation. *arXiv preprint arXiv:1408.6988*.

Dan Jurafsky. 1997. Switchboard swbd-damsl shallow-discourse-function annotation coders manual. *www. dcs. shef. ac. uk/nlp/amities/files/bib/ics-tr-97-02. pdf*.

Hirotani Kitano. 1991. Phi dm-dialog: an experimental speech-to-speech dialog translation system. *Computer*, 24(6):36–50.

Harshit Kumar, Arvind Agarwal, Riddhiman Dasgupta, Sachindra Joshi, and Arun Kumar. 2017. Dialogue act sequence labeling using hierarchical encoder with crf. *arXiv preprint arXiv:1709.04250*.

Harshit Kumar, Arvind Agarwal, and Sachindra Joshi. 2018. Dialogue-act-driven conversation model: An experimental study. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1246–1256.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. Dailydialog: A manually labelled multi-turn dialogue dataset. *arXiv preprint arXiv:1710.03957*.

Chia-Wei Liu, Ryan Lowe, Iulian V Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *arXiv preprint arXiv:1603.08023*.

Ryan Lowe, Nissan Pow, Iulian V Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, page 285.

Ryan Thomas Lowe, Nissan Pow, Iulian Vlad Serban, Laurent Charlin, Chia-Wei Liu, and Joelle Pineau. 2017. Training end-to-end dialogue systems with the ubuntu dialogue corpus. *Dialogue & Discourse*, 8(1):31–65.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Alan Ritter, Colin Cherry, and William B Dolan. 2011. Data-driven response generation in social media. In *Proceedings of the conference on empirical methods in natural language processing*, pages 583–593. Association for Computational Linguistics.

Iulian Vlad Serban, Ryan Lowe, Peter Henderson, Laurent Charlin, and Joelle Pineau. 2015. A survey of available corpora for building data-driven dialogue systems. *arXiv preprint arXiv:1512.05742*.

Yiping Song, Rui Yan, Cheng-Te Li, Jian-Yun Nie, Ming Zhang, and Dongyan Zhao. 2018. An ensemble of retrieval-based and generation-based human-computer conversation systems.

Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, Paul Taylor, Rachel Martin, Carol Van Ess-Dykema, and Marie Meteer. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational linguistics*, 26(3):339–373.

Jörg Tiedemann. 2009. News from opus-a collection of multilingual parallel corpora with tools and interfaces. In *Recent advances in natural language processing*, volume 5, pages 237–248.

Tsung-Hsien Wen, Yishu Miao, Phil Blunsom, and Steve Young. 2017. Latent intention dialogue models. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3732–3741. JMLR. org.

Tsung-Hsien Wen, David Vandyke, Nikola Mrksic, Milica Gasic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2016. A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562*.

Yu Wu, Wei Wu, Chen Xing, Ming Zhou, and Zhoujun Li. 2016. Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots. *arXiv preprint arXiv:1612.01627*.

Can Xu, Wei Wu, and Yu Wu. 2018. Towards explainable and controllable open domain dialogue generation with dialogue acts. *arXiv preprint arXiv:1807.07255*.

Tiancheng Zhao, Kyusong Lee, and Maxine Eskenazi. 2018. Unsupervised discrete sentence representation learning for interpretable neural dialog generation. *arXiv preprint arXiv:1804.08069*.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. *arXiv preprint arXiv:1703.10960*.

# How to Build User Simulators to Train RL-based Dialog Systems

**Weiyan Shi**[*1], **Kun Qian**[*1], **Xuewei Wang**[2] **and Zhou Yu**[1]
[1] University of California, Davis, [2] Carnegie Mellon University
{wyshi, kunqian, joyu}@ucdavis.edu, xueweiwa@andrew.cmu.edu

## Abstract

User simulators are essential for training reinforcement learning (RL) based dialog models. The performance of the simulator directly impacts the RL policy. However, building a good user simulator that models real user behaviors is challenging. We propose a method of standardizing user simulator building that can be used by the community to compare dialog system quality using the same set of user simulators fairly. We present implementations of six user simulators trained with different dialog planning and generation methods. We then calculate a set of automatic metrics to evaluate the quality of these simulators both directly and indirectly. We also ask human users to assess the simulators directly and indirectly by rating the simulated dialogs and interacting with the trained systems. This paper presents a comprehensive evaluation framework for user simulator study and provides a better understanding of the pros and cons of different user simulators, as well as their impacts on the trained systems. [1]

## 1 Introduction

Reinforcement Learning has gained more and more attention in dialog system training because it treats the dialog planning as a sequential decision problem and focuses on long-term rewards (Su et al., 2017). However, RL requires interaction with the environment, and obtaining real human users to interact with the system is both time-consuming and labor-intensive. Therefore, building user simulators to interact with the system before deployment to real users becomes an economical choice (Williams et al., 2017; Li et al., 2016). But the performance of the user simulator has a direct impact on the trained RL policy. Such an intertwined relation between user simulator and dialog system makes the whole process a "chicken and egg" problem. This naturally leads to the question of how different user simulators impact the system performance, and how to build appropriate user simulators for different tasks.

In previous RL-based dialog system literature, people reported their system's performance, such as success rate, on their specific user simulators (Liu and Lane, 2017; Shi and Yu, 2018), but the details of the user simulators are not sufficient to reproduce the results. User simulators' quality can vary in multiple aspects, which could lead to unfair comparison between different trained systems. For instance, RL systems built with more complicated user simulators will have lower scores on the automatic metrics, compared to those built using simpler user simulators. However, the good performance may not necessarily transfer when the system is tested by real users. In fact, models that have a low score but are trained on better simulators may actually perform better in real situations because they have experienced more complex scenarios. In order to obtain a fairer comparison between systems, we propose a set of standardized user simulators. We pick the popular restaurant search task from Multiwoz (Budzianowski et al., 2018) and analyze the pros and cons of different user simulator building methods.

The potential gap between automatic metrics and real human evaluation also makes user simulator hard to build. The ideal evaluator of a dialog system should be its end-users. But as stated before, to obtain real user evaluation is time-consuming. Therefore, many automatic metrics have been studied to evaluate a user simulator (Pietquin and Hastie, 2013; Kobsa, 1994) from different perspectives. However, we do not know how these automatic metrics correlate with human satisfaction. In this paper, we ask human users to

[*] Equal contribution.
[1] The code and data are released at https://github.com/wyshi/user-simulator.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1990–2000, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

both rate the dialogs generated by the user simulators, and interact with the dialog systems trained with them, in order to quantify the gap between the automatic metrics and human evaluation.

This paper presents three contributions: first, we annotate the user dialog acts in the restaurant domain in Multiwoz 2.0; second, we build multiple user simulators in the standard restaurant search domain and publish the code to facilitate further development of RL-based dialog system training algorithms; third, we perform comprehensive evaluations on the user simulators and trained RL systems, including automatic evaluation, human rating simulated dialogs, human interacting with trained systems and cross study between simulators and systems, to measure the gap between automatic dialog completion metrics with real human satisfaction, and provide meaningful insights on how to develop better user simulators.

## 2 Related work

One line of prior user simulator research focuses on agenda-based user simulator (ABUS) (Schatzmann et al., 2006, 2007; Schatzmann and Young, 2009; Li et al., 2016) and it is most commonly used in task-oriented dialog systems. An agenda-based user simulator is built on hand-crafted rules according to an agenda provided at the beginning of a dialog. This mechanism of ABUS makes it easier to explicitly integrate context and agenda into the dialog planning. Schatzmann and Young (2009) presented a statistical hidden agenda user simulator, tested it against real users and showed that a superior result in automatic metrics does not guarantee a better result in the real situation. Li et al. (2016) proposed an agenda-based user simulator in the movie domain and published a generic user simulator building framework. In this work, we build a similar agenda-based user simulator in the restaurant domain, and focus more on analyzing the effects of using different user simulators.

However, it's not feasible to build agenda-based user simulators for more complex tasks without an explicit agenda. Therefore, people have also studied how to build user simulators in a data-driven fashion. He et al. (2018) fit a supervised-learning-based user simulator to perform RL training on a negotiation task. Asri et al. (2016) developed a seq2seq model for user simulation in the restaurant search domain, which took the dialog context into consideration without the help of external data structure. Kreyssig et al. (2018) introduced the Neural User Simulator (NUS) which learned user behaviour from a corpus and generates natural language directly instead of semantic output such as dialog acts. However, unlike in ABUS, how to infuse the agenda into the dialog planning and assure consistency in data-driven user simulators has been an enduring challenge. In this paper, we present a supervised-learning-based user simulator and integrate the agenda into the policy learning. Furthermore, we compare such a data-driven method with its agenda-based counterpart.

Another line of user simulator work treats the user simulator itself as a dialog system, and train the simulator together with the RL system iteratively (Liu and Lane, 2017; Shah et al., 2018). Shah et al. (2018) proposed the Machines Talking To Machines (M2M) framework to bootstrap both user and system agents with dialog self-play. Liu and Lane (2017) presented a method for iterative dialog policy training and address the problem of building reliable simulators by optimizing the system and the user jointly. But such iterative approach requires extra effort in setting up RL and designing reward for the user simulator, which may result in the two agents exploiting the task, and leads to numerical instability.

Another challenging research question is how user simulator performance can be evaluated (Schatztnann et al., 2005; Ai and Litman, 2011a,b; Engelbrecht et al., 2009; Hashimoto et al., 2019). Pietquin and Hastie (2013) conducted a comprehensive survey over metrics that have been used to assess user simulators, such as perplexity and BLEU score (Papineni et al., 2002). However, some of the metrics are designed specifically for language generation evaluation, and as Liu et al. (2016) pointed out, these automatic metrics barely correlate with human evaluation. Therefore, Ai and Litman (2011a) involved human judges to directly rate the simulated dialog. Schatzmann and Young (2009) asked humans to interact with the trained systems to perform indirect human evaluation. Schatztnann et al. (2005) proposed *cross-model evaluation* to compare user simulators since human involvement is expensive. We combine the existing evaluation methods and conduct comprehensive assessments to measure the gap between automatic metrics and human satisfaction.

## 3 Dataset

We choose the restaurant domain in Multiwoz 2.0 (Budzianowski et al., 2018) as our dataset, because it's the most classic domain in task-oriented dialog systems. The system's task is to help users find restaurants, provide restaurant information and make reservations. There are a total of 1,310 dialogs annotated with *informable* slots (e.g. *food, area*) that narrow downs the restaurant choice, and *requestable* slots (e.g. *address, phone*) that track users' detailed requests about the restaurant. But because the original task in Multiwoz was to model the system response, it only contains dialog act annotation on the system-side but not on the user-side. To build user simulators, we need to model user behaviors, and therefore, we annotate the user intent in Multiwoz. In order to build user simulators, we need to model user behavior and therefore, we annotate the user-side dialog act in the restaurant domain of Multiwoz. Two human expert annotators analyze the data and agree on a set of seven user dialog acts ($UserActs$): *"inform restaurant type"*, *"inform restaurant type change"*, *"anything else"*, *"request restaurant info"*, *"make reservation"*, *"make reservation change time"*, and *"goodbye"*. Because the data is relatively clean and constrained in domain, the annotation is performed by designing regular expression first and cleaned by human annotators later. We manually checked 10% of the data (around 500 utterances) and the accuracy for automatic annotations is 94%. These annotated user dialog acts will serve as the foundation of the user simulator action space $UserActs$. The annotated data is released to facilitate user simulator study.

## 4 User Simulator Design

According to Li et al. (2016), user simulator building eventually boils down to two important tasks: building 1) a dialog manager (DM) (Henderson et al., 2014; Cuayáhuitl et al., 2015; Young et al., 2013) that governs the simulator's next move; and 2) a natural language generation module (NLG) (Tran and Nguyen, 2017; Dušek and Jurčíček, 2016) that translates the semantic output from dialog manager into natural language. The user simulator can adopt either *agenda-based* approach or *model-based* approach for the dialog manager. While for NLG, the user simulator can use the dialog act to select pre-defined templates, retrieve user utterances from previously collected dialogs, or generate the surface form utterance directly with pre-trained language model (Jung et al., 2009).

The dialog manager module ensures the intrinsic logical consistency of the user simulator, while the NLG module controls the extrinsic language fluency. DM and NLG play an equally important role in the user simulator design and must go hand-in-hand to imitate user behaviours. Therefore, we propose to test different combinations of DM and NLG methods to answer the question of how to build the best user simulator.

In task-oriented dialog systems, the user simulator's task is to complete a pre-defined goal by interacting with the system. Multiwoz provides detailed goals for each dialog, which serves as the goal database. These goals consist of sub-tasks, such as request information or make reservation. An example goal is, "*You're looking for an Italian restaurant in the moderate price range in the east. Once you find the restaurant, you want to book a table for 5 people at 12:15 on Monday. Make sure you get the reference number.*" During initial RL experiments, we find that similar to supervised learning, the data imbalance in goals will impact the reinforce learning in the simulated tasked-oriented dialog setting. We find that 2/3 of the goals contain the sub-task "ask info" and the rest 1/3 are about "make reservation". Because the user simulators are all goal-driven, the RL policy is only able to experience the "reservation" scenario 1/3 of the time on average, which will result in the model favoring the "ask info" scenario more, especially in the early training stage. This further misleads the policy (Su et al., 2017). Therefore, we augment the goal set with more "make reservation" sub-task from MultiWoz to make the sub-tasks of "make reservation" and "ask info" even. This augmented goal set with more even distribution serves as our goal database. We randomly sample a goal from the goal database during training. A user goal defines the agenda the user simulator needs to follow, so we'll use "goal" and "agenda" interchangeably in this paper.

### 4.1 Dialog Manager

**Agenda-based** We employ the traditional agenda-based stack-like user simulator (Schatzmann and Young, 2009; Li et al., 2016), where the dialog manager chooses a dialog act among the user dialog act set $UserActs$ mentioned in Section 3. The

dialog act transition is governed by hand-set rules and probabilities based on the initial goal. For example, after the system makes a recommendation, the user can go on to the next sub-task, or ask if there is another option. Fig. 1 shows a typical agenda. Because the restaurant task is a user-initiated task, agenda-based simulator's first dialog act is always "inform restaurant type". The dialog history is managed with the user dialog state by pushing and popping important slots to ensure consistency. Although the restaurant search task is simple, designing an agenda-based system for the task is non-trivial, because there are many corner cases to be handled. However, the advantage of building agenda-based system is that it does not require thousands of annotated dialogs.

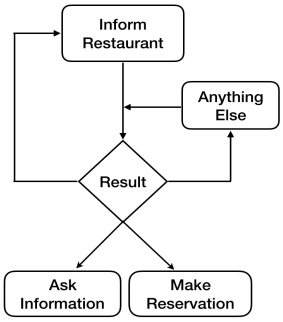


Figure 1: An example user agenda for the restaurant task

**Model-based** It requires specific human expertise to design rules for agenda-based user simulators (compared to more easily accessible annotation), and the process is both labor-intensive and error-prone. Moreover, for complicated tasks such as negotiation, it is not practical to design rules in the policy (He et al., 2018). Therefore, we explore the possibility of building dialog manager with supervised learning methods. Compared to agenda-based simulators which require special expert knowledge, supervised learning methods require less expert involvement. We utilize Sequicity (Lei et al., 2018) to construct model-based user simulator. Sequicity is a simple seq2seq dialog system model with copy and attention mechanism. It also used belief span to track the dialog states. For example, *inform:{Name:"Caffee Uno"; Phone:"01223448620"}* records the information that the system offers and this would be kept in belief span throughout the dialog, while *request:{"food", "price range"}* means the system is asking for more information from user to locate a restaurant, which would be removed from belief span once the request is fulfilled. There are 13 types of system dialog acts. To focus on the valuable information to fit the model, we combine these dialog acts into 5 categories: {*"inform","request","book inform","select", "recommend"*}. Similarly, we define three types of user goals *"inform","request"* and *"book"*, and record them in belief span, denoted as $G$. So, at time $t$, we first update belief span with a seq2seq model, based on current system response $R_t$, previous belief state $B_{t-1}$ and previous user utterance $U_{t-1}$:

$$B_t = \text{seq2seq}(B_{t-1}, U_{t-1}, R_t)$$

Then we incorporate user goal and the context above to generate current user utterance:

$$U_t = \text{seq2seq}(B_{t-1}, U_{t-1}, R_t | B_t, G)$$

As illustrated in Fig. 2, we build a GRU-based encoder for all the $B_{t-1}$, $U_{t-1}$, $R_t$ and the goal $G$. Then we decode the current belief span and user utterance separately. Both decoders are a one-layer GRU with copy and attention mechanism. To evaluate the dialog manager alone, we also modify the Sequicity's second decoder to generate system dialog act ($A_t$) instead of system utterances.

$$A_t = \text{seq2seq}(B_{t-1}, U_{t-1}, R_t | B_t, G)$$

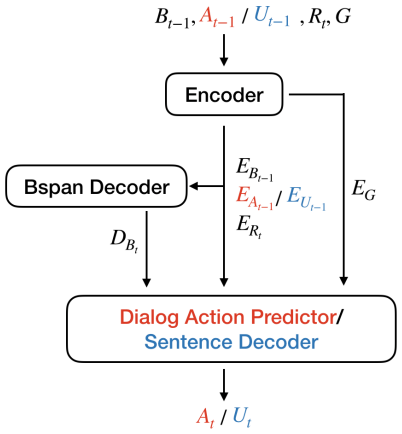


Figure 2: The end-to-end simulator and user dialog act predictor share the most part of their model, colored in black, except the decoder. All the parameters colored in red are related to the dialog act predictor and the parameters in blue color are for sentence decoder.

## 4.2 Natural Language Generation

Dialog act-based NLG is formalized as $U_t = M(A_t)$, where $A_t$ is the selected dialog act by the

dialog manager, $U_t$ is the generated user utterance. We describe three different dialog-act-based NLG methods.

**Template** Template method requires human experts to write a variety of delexicalized templates for each dialog act. By searching in the templates, it translates $A_t$ into human-readable utterances. The quality of the templates have direct impact on the NLG quality.

**Retrieval** Template method suffers from limited vocabulary size and language diversity. An alternative method is Retrieval-based NLG (Wu et al., 2016; Hu et al., 2014). The model retrieves user utterances with $A_t$ as their dialog act in the training dataset. Following He et al. (2018), we represent the context by a TF-IDF weighted bag-of-words vector and compute the similarity score between the candidate's context vector and the current context vector to retrieval $U_t$.

**Generation** Generation method (Wen et al., 2015a,b) does not need expert involvement to rewrite templates, but requires dialog act annotation similar to retrieval method. We build a vanilla seq2seq (Sutskever et al., 2014) model using the annotated data adding $A_t$ in the input.

## 5 Dialog System Training Setting

Traditionally, hand-crafted dialog acts plus slot values are used as the discrete action space in RL training (Raux et al., 2005). Dialog action space can also be on the word-level. However, previous study shows degenerate behavior when using word-level action space (Zhao et al., 2019), as it is difficult to design a reward. We choose the first approach and use the discrete action space with six system dialog acts: "ask restaurant type", "present restaurant search result", "provide restaurant info", "ask reservation info", "inform reservation result", "goodbye". Simple action masks are applied to avoid impossible actions such as making reservation before presenting a restaurant.

We use a 2-layer bidirectional-GRU with 200 hidden units to train a NLU module. For simplicity, we use the template-based method in the system's NLG module. We used policy gradient method to train dialog systems (Williams, 1992). During RL training, a discounted factor of 0.9 is applied to all the experiences with the maximum number of turns to be 10. We also apply the $\epsilon$-greedy exploration strategy (Tokic, 2010). All the RL systems use the same RL state representation, which consists of traditional dialog state and word count vector of the current utterance. The same reward function is used, which is $+1$ for task success, $-1$ for task failure and $-0.1$ for each additional turn to encourage the RL policy to finish the task faster rather than slower. We fix the RL model every 1,000 episodes and test for 100 dialogs to calculate the average success rate, shown in Fig. 3.

Besides RL systems, we also build a rule-based system *Rule-System*, which serves as the third-party system to interact with each user simulator and generate simulated dialogs for human evaluation. The only difference between *Rule-System* and the RL-based systems is their policy selection module, where *Rule-System* uses hand-crafted rules while RL-based systems use RL policy.

## 6 User Simulator Evaluation

Evaluating the quality of a user simulator is an enduring challenge. Traditionally, we report *direct* automatic metrics of the user simulator, such as perplexity (Ai and Litman, 2011b; Pietquin and Hastie, 2013). Besides, the performance of the RL system trained with a specific simulator gives us an *indirect* assessment of the user simulator's ability to imitate user behaviours.

The ultimate goal of the user simulator is to build a task-oriented RL system to serve real users. Therefore, the most ideal evaluation should be conducted by human. Therefore, we first asked human to read the simulated dialogs and rate the user simulator's performance *directly*. We then hired Amazon Mechanic Turkers (AMT) to interact with the RL systems trained with different simulators and rate their performance. Besides, we also performed cross study between user simulators and systems trained with different simulators to see if the systems' performance can be transferred to a different simulated setting. Finally, we measure the gap between the automatic metrics and human evaluation scores, and share insights on how to evaluate user simulator effectively.

### 6.1 Automatic Evaluation

**Perplexity (Direct)** Perplexity measures the language generation quality of the user simulator. The results are shown in Table 1. For each simulator model, we generate 200 dialogs with the third-party *Rule-System* and train a trigram language model with the data. Then we test the model and compute the perplexity with 5000 user utterances

| Simulators | NLU | DM | NLG | PPL | Vocab | Utt | Hu.Fl | Hu.Co | Hu.Go | Hu.Div | Hu.All |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Agenda-Template (AgenT) | SL | Agenda | Template | 10.32 | 180 | 9.65 | 4.07 | 4.56 | 4.88 | 2.4 | 4.50 |
| Agenda-Retrieval (AgenR) | SL | Agenda | Retrieval | 33.90 | **383** | **11.61** | 3.50 | 4.22 | 4.58 | 3.9 | 3.74 |
| Agenda-Generation (AgenG) | SL | Agenda | Generation | **7.49** | 159 | 8.07 | 3.32 | 3.92 | 4.64 | 2.5 | 3.36 |
| SL-Template (SLT) | SL | | Template | 9.32 | 192 | 9.83 | **4.80** | **4.80** | **4.98** | 2.6 | **4.74** |
| SL-Retrieval (SLR) | SL | | Retrieval | 29.36 | 346 | 11.06 | 4.40 | 3.99 | 4.88 | **4.3** | 4.01 |
| SL-End2End (SLE) | End-to-End | | | 13.47 | 205 | 10.95 | 3.32 | 2.62 | 3.18 | 2.7 | 2.64 |

Table 1: Automatic metrics and human evaluation scores of different user simulators. Automatic metrics include, perplexity per word (PPL), vocabulary size (Vocab), average utterance length (Utt). Human evaluation metrics include, sentence fluency (Hu.Fl), coherent (Hu.Co), goal adherence (Hu.Go), language diversity (Hu.Div) and overall score (Hu.All).

sampled from MultiWoz. Although the perplexity for retrieval models is the highest in both agenda-based and SL-based simulators, it also possesses the biggest vocabulary set and the longest average utterance length. Another common automatic metrics used to assess the language model is BLEU, but since this is a user simulator study and we don't have ground truth, BLEU score is not available.

**Vocabulary Size (Direct)** Vocabulary size is a simple and straightforward metric that measures the language diversity. As expected, retrieval-based models have the biggest vocabulary set. However, Agenda-Generation has the smallest vocabulary set. The possible reason behind is that we adopt a vanilla greedy seq2seq that suffers from generating the most frequent words. SL-End2End in Table 1 trains the NLU, DM and NLG jointly, and therefore, the vocabulary size is slightly larger than the template-based methods.

**Average utterance length (Direct)** Average utterance length is another simple metric to assess the language model and language diversity. As expected, retrieval-based methods are doing the best, but SL-End2End is also doing a good job in generating long sentences.

**Success Rate (Indirect)** The success rate is the most commonly used metric in reporting RL dialog system performance. Also, it can reflect the user simulator's certain behaviour. The success rate of various user simulators are shown in Fig. 3. SL-based user simulators converge faster than rule-based simulators. It can be explained by the observation that SL tries to capture the major paths in the original data, and counts those as success, instead of exploring all the possible paths like in the agenda-based simulators. In general, retrieval-based simulators converge slower than other NLG methods because retrieval-based approach has a bigger vocabulary size.

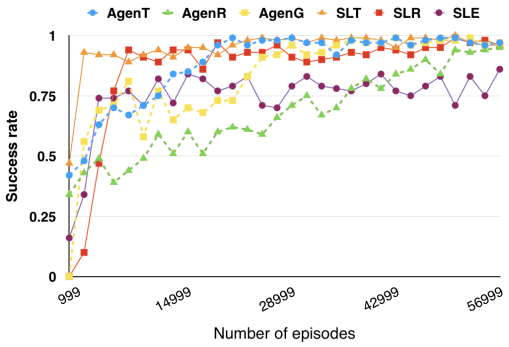


Figure 3: Average success rate during RL training.

## 6.2 Human Direct Evaluation

The direct evaluation of the user simulator is conducted by asking 10 volunteers to read the simulated dialogs between different simulators and the third-party *Rule-System*. Each of the 10 volunteers would rate five randomly-selected dialogs generated from each model, and the average of the total 50 ratings is reported as the final human-evaluation score. The *Rule-System* is built solely based on hand-crafted rules with no knowledge about any of the simulators, and therefore is fair to all of them. We design four metrics to assess the user simulator's behaviour from multiple aspects. The results are shown in Table 1.

**Fluency** focuses on the language quality, such as grammar, within each utterance unit. Agenda-Template (AgenT) and SL-Template (SLT) received the two highest fluency scores because the templates are all written by human.

**Coherence** focuses on the relation quality between different turns within one dialog. SL-Template (SLT) simulator performs the best in coherence, but agenda-based simulators in general are a bit more coherence than SL-based ones.

**Goal Adherence** focuses on the relation between the goal and the simulator-generated utterances. Both agenda-based and SL-based simulators in general stick to the goal with the exception of

| RL System | Solved Ratio | Satisfaction | Efficiency | Naturalness | Rule-likeness | Dialog Length | Auto Success |
|---|---|---|---|---|---|---|---|
| Sys-AgenT | 0.814 ±0.06 | 4.29 ±0.20 | 4.35 ±0.21 | 3.96 ±0.23 | 4.49 ±0.15 | 8.95 ±0.38 | **0.983** ±0.01 |
| Sys-AgenR | **0.906** ±0.05 | **4.52** ±0.15 | 4.45 ±0.16 | 4.23 ±0.19 | 4.59 ±0.14 | **8.73** ±0.31 | 0.925 ±0.02 |
| Sys-AgenG | 0.904 ±0.05 | 4.38 ±0.18 | **4.46** ±0.19 | **4.33** ±0.17 | 4.51 ±0.16 | 9.48 ±0.45 | 0.980 ±0.01 |
| Sys-SLT | 0.781 ±0.07 | 3.87 ±0.22 | 3.81 ±0.22 | 3.63 ±0.22 | **4.08** ±0.21 | 9.61 ±0.76 | 0.978 ±0.01 |
| Sys-SLR | 0.823 ±0.05 | 4.23 ±0.20 | 4.20 ±0.10 | 3.99 ±0.20 | 4.42 ±0.17 | 8.92 ±0.70 | 0.965 ±0.01 |
| Sys-SLE | 0.607 ±0.06 | 3.42 ±0.22 | 3.41 ±0.23 | 3.59 ±0.20 | 4.22 ±0.20 | 9.44 ±0.69 | 0.798 ±0.03 |

Table 2: Human evaluation of RL systems trained with different simulators on AMT with 95% confidence intervals. Each row represents one RL system, e.g. Sys-AgenT means the RL system trained with the AgenT simulator.

SL-End2End (SLE). This may be because SLE is training all the modules together and thus has more difficulty infusing the goal.

**Diversity** focuses on the language diversity between simulated dialogs of the same simulator. Each simulator will be given one diversity score. Retrieval-based methods surpass other methods in diversity, but it is not as good in fluency, while template-based methods outperform in fluency but suffer on diversity as expected. Generative methods suffer from generating generic sentences as mentioned before.

**Overall** We ask the human to rate the overall simulator quality. Except for SL-End2End, SL-based methods are favoured by human over agenda-based methods. Agenda-Template is comparable to SL-based simulators because of its fluent responses and carefully-designed policy.

### 6.3 Human Indirect Evaluation on AMT

The ultimate goal of user simulator building is to train better system policies. Automatic metrics such as success rate can give us a sense on the system's performance, but the ultimate evaluation should be conducted on human so that we can know the real performance of the system policy when deployed.

Motivated by this, we tested the RL systems trained with various user simulators on Amazon Mechanical Turk (AMT) (Miller et al., 2017), and asked Turkers to interact with the system and obtained their opinions. Each system is tested on 100 Turkers. The results are shown in Table 2. The AMT interface is in the Appendix.

We also listed two common automatic metrics in Table 2 to compare. The "Dialog Length" column shows the average dialog length of the Turker-Machine dialogs, which reflects the system's efficiency to some extent. The "Auto Success" column represents the automatic success rate. It's the convergent success rate from Fig. 3, measured by freezing the policy and testing against the user simulator for 100 episodes. Previous approaches have utilized these two automatic metrics to evaluate the system's efficiency and success (Williams et al., 2017; Shi et al., 2019), but we find that due to user individual difference, such automatic metrics have relatively big variances and don't always correlate with efficiency perceived by human. For example, some users tend to provide all slots in one turn, while others provide slots only when necessary; some users would even go off-the-script and ask about restaurants not mentioned in the goal. Therefore, we should caution against relying solely on the automatic metrics to represent user opinion and the best way is to ask the users directly for their thoughts on the system's performance from multiple aspects as follows.

**Solved Ratio.** Each Turker is given a goal at the beginning, the same as in the simulated setting. At the end of the dialog, we ask the Turker if the system has solved his/her problem. There are three types of answers to this question, "Yes" is coded as 1, "Partially solved" is coded as 0.5, and "No" is coded as 0. Sys-AgenR is the system trained with the Agenda-Retrieval (AgenR) simulator and it received the highest score, better than the Sys-AgenT trained with the AgenT simulator, which is reasonable because through retrieval, Agenda-Retrieval (AgenR) simulators present more language diversity to the system during training. When interacting with a real user, the systems that can handle more language variations will do better.The SL-based simulators received relatively low scores. Further investigation on this cause is presented in the discussion section.

"Auto-Success" has been used to reflect the solved ratio previously. However, it's not necessarily correlated with the user-rated solved ratio. For example, Sys-AgenG's Auto-Success rate is much higher than Sys-AgenR's Auto Success rate, but the users think that these two systems perform the same in terms of *Solved Ratio*.

| Usr\Sys | **Sys-AgenT** | **Sys-AgenR** | **Sys-AgenG** | **Sys-SLT** | **Sys-SLR** | **Sys-SLE** |
|---|---|---|---|---|---|---|
| AgenT | 0.975 | 0.960 | 0.790 | 0.305 | 0.300 | 0.200 |
| AgenR | 0.540 | 0.900 | 0.785 | 0.230 | 0.230 | 0.235 |
| AgenG | 0.725 | 0.975 | 0.950 | 0.355 | 0.300 | 0.20 |
| SLT | 0.985 | 0.985 | 0.985 | 0.990 | 0.965 | 0.730 |
| SLR | 0.925 | 0.975 | 0.965 | 0.975 | 0.935 | 0.630 |
| SLE | 0.770 | 0.820 | 0.815 | 0.840 | 0.705 | 0.770 |
| Average | 0.820 | **0.935** | 0.882 | 0.616 | 0.573 | 0.461 |

Table 3: Cross study results. Each row represents one user simulator, each column represents one RL system trained with a specific simulator. Each entry shows the average success rate obtained by having the user simulator interacting with the RL system for 200 times.

**Satisfaction.** Solving the user's problem doesn't necessarily lead to user satisfaction sometimes. It also depends on the system's efficiency and latency. Therefore, besides Solved Ratio, we also directly ask Turkers how satisfied they are with the system. The result shows that among all systems, Sys-AgenR model received the highest score. The positive correlation between the "Solved Ratio" and "Satisfaction" in Table 1 also indicates automatic task completion rate is a good estimator for user satisfaction.

**Efficiency.** We directly ask Turkers how efficient the system is in solving their problems since dialog length doesn't always correlate with the system efficiency. For example, although the dialog length of Sys-AgenG and Sys-SLE are similar to each other, users rated Sys-AgenG to be the most efficient one and Sys-SLE to be the most inefficient one. Again we suspect this is caused by different user communication pattern where some users prefer providing slots across multiple turns while others prefer providing all slots in one turn.

**Naturalness.** We ask the Turkers to rate the naturalness of the system responses. All the systems share the same template-based NLG module designed by human experts, thus there shouldn't be a significant difference in the naturalness score. However, according to Table 2, we find that the naturalness score seems to correlate with the overall system performance. A possible reason is that the end-user is rating the system's naturalness by the overall performance instead of the system responses alone. When the dialog policy is bad, even if the NLG module can generate natural system responses, the users would still think the system is unnatural. This suggests that when designing dialog systems, NLG and policy selection modules should go hand-in-hand in evaluation.

**Rule-likeness** We also ask the users to what extend they think the system is designed by handcrafted rules on a scale from 1 to 5, five means it is heavily handcrafted. Among all the models, Sys-SLT that is trained with the SL-Template simulator receives the lowest score, meaning it's the least rigid system. This is because SL-Template's dialog manager is learned with supervised learning, less rigid than the agenda-based dialog policy, which further leads to a less rigid behaviour of the trained dialog system.

### 6.4 Cross Study of Simulators and Systems

From the last column in Table 2, we find that although the automatic success rates claimed by the user simulator used to train the system are all relatively high, the high automatic success rate doesn't transfer to real human satisfaction. In our setting, each simulator can be viewed as a new user with different communicating habits; therefore, we are curious to see if the performance can transfer to a different simulator when we test the RL system trained with simulator A against simulator B. Table 3 shows a cross study between the six user simulators and the six systems trained with different simulators, where we fix the systems, have each simulator interact with each system for 200 episodes, and calculate the average success rate. The diagonal should reflect the "Auto Success" column in Table 2, but since the 200 episodes are random and the "Auto Success" is the convergent success rate, the exact number won't be the same.

The last row in Table 3 shows the average success rate of each system across user simulators. There are some interesting findings. 1) Sys-AgenR that is trained with the Agenda-Retrieval simulator has the best average success rate, which agrees with the human evaluation on MTurk. 2) A common practice to compare RL systems $S_1$, ... $S_n$ is to fix one user simulator $U$ and then com-

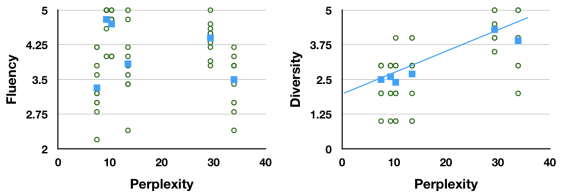


Figure 4: Correlation between sentence fluency and perplexity, and correlation between sentence diversity and perplexity. Green circles represent the human rated score, while the blue squares are the average score over different raters.

pare the success rate of $S_1$, ..., $S_n$ on $U$. However, by looking at the fifth row for the SL-Retrieval simulator, it will prefer Sys-SLT (0.975) over Sys-AgenG (0.965), but actually the average performance of Sys-AgenG (0.882) is better than Sys-SLT (0.616) from the last row. This suggests that when we want to compare two systems but don't have the resource to do human evaluation on the system performance, instead of solely comparing their success rates tested on one simulator, we should build different types of user simulators and test the systems against multiple simulators to get a more holistic view of the systems. 3) The diagonal in the table is usually the highest, meaning that RL policy does a good job optimizing for its own simulator but may not generalize to other user simulators. For example, the upper right corner performs the worst because the systems trained with SL-based simulators are worse in general, whose reason we will discuss later.

### 6.5 Human Correlation Study

To test if the automatic metrics can reflect human evaluation, we compute the correlation between perplexity (PPL) and human evaluated fluency (Hu.Fl) and the correlation between perplexity and human evaluated diversity score (Hu.Div), which are $-0.21$ with $p > 0.05$ and $0.95$ with $p = 0.003$ respectively. We also visualize these metrics in Fig. 4. It shows that as an automatic metric, perplexity is a good estimator for language diversity but not for language fluency.

## 7 Discussion and Future Work

SL-based simulators perform relatively worse than Agenda-based simulators when interacting with real users. We investigate the data and find it's caused by SL-based simulators not exploring all possible paths. We draw the different dialog act distributions on simulated conversations in Fig. 5. For example, in agenda-based simulators, we explicitly have a rule for the dialog act "anything else" (Act6 in Fig. 5) but no such rules exist in SL-based simulators. Therefore, the RL model will experience the "anything else" scenario more in Agenda-based simulators than in SL-based simulators. When real users ask about "anything else", RL systems trained with Agenda-based simulators will have more experiences in handling such a case, compared to systems trained with SL-based simulators.

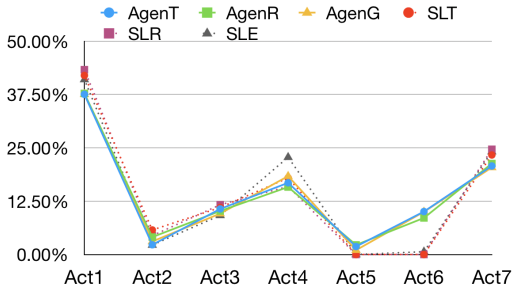


Figure 5: Dialog act distribution comparison. Act1 to Act7 corresponds to the seven user dialog acts, *"inform restaurant type"*, *"inform restaurant type change"*, *"ask info"*, *"make reservation"*, *"make reservation change time"*, *"anything else"*, and *"goodbye"*

In this paper, we perform in-depth studies on the restaurant domain as it's the most well-studied domain in task-oriented dialog systems, yet there's still no standard user simulator available. In the future we plan to include more domain using various domain-adaptive methods (Qian and Yu, 2019; Tran and Nguyen, 2018; Gašić et al., 2017) to support multi-domain dialog system research, and incorporate our work into more and more standardized dialog system platforms (Lee et al., 2019).

## 8 Conclusions

User simulators are essential components in training RL-based dialog systems. However, building user simulators is not a trivial task. In this paper, we surveyed through different ways to build user simulators at the levels of dialog manager and NLG, and analyzed the pros and cons of each method. Further, we evaluated each simulator with automatic metrics and human evaluations both directly and indirectly, and shared insights on better user simulator building based on comprehensive analysis.

## References


Hua Ai and Diane Litman. 2011a. Assessing user simulation for dialog systems using human judges and automatic evaluation measures. *Natural Language Engineering*, 17(4):511–540.

Hua Ai and Diane Litman. 2011b. Comparing user simulations for dialogue strategy learning. *ACM Transactions on Speech and Language Processing (TSLP)*, 7(3):9.

Layla El Asri, Jing He, and Kaheer Suleman. 2016. A sequence-to-sequence model for user simulation in spoken dialogue systems. *arXiv preprint arXiv:1607.00070*.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. Multiwoz-a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. *arXiv preprint arXiv:1810.00278*.

Heriberto Cuayáhuitl, Simon Keizer, and Oliver Lemon. 2015. Strategic dialogue management via deep reinforcement learning. *arXiv preprint arXiv:1511.08099*.

Ondřej Dušek and Filip Jurčíček. 2016. Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings. *arXiv preprint arXiv:1606.05491*.

Klaus-Peter Engelbrecht, Michael Quade, and Sebastian Möller. 2009. Analysis of a new simulation approach to dialog system evaluation. *Speech Communication*, 51(12):1234–1252.

Milica Gašić, Nikola Mrkšić, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2017. Dialogue manager domain adaptation using gaussian process reinforcement learning. *Computer Speech & Language*, 45:552–569.

Tatsunori B Hashimoto, Hugh Zhang, and Percy Liang. 2019. Unifying human and statistical evaluation for natural language generation. *arXiv preprint arXiv:1904.02792*.

He He, Derek Chen, Anusha Balakrishnan, and Percy Liang. 2018. Decoupling strategy and generation in negotiation dialogues. *arXiv preprint arXiv:1808.09637*.

Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014. The third dialog state tracking challenge. In *2014 IEEE Spoken Language Technology Workshop (SLT)*, pages 324–329. IEEE.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in neural information processing systems*, pages 2042–2050.

Sangkeun Jung, Cheongjae Lee, Kyungduk Kim, Minwoo Jeong, and Gary Geunbae Lee. 2009. Data-driven user simulation for automated evaluation of spoken dialog systems. *Computer Speech & Language*, 23(4):479–509.

Alfred Kobsa. 1994. User modeling and user-adapted interaction. In *CHI Conference Companion*, pages 415–416.

Florian Kreyssig, Inigo Casanueva, Pawel Budzianowski, and Milica Gasic. 2018. Neural user simulation for corpus-based policy optimisation for spoken dialogue systems. *arXiv preprint arXiv:1805.06966*.

Sungjin Lee, Qi Zhu, Ryuichi Takanobu, Xiang Li, Yaoqin Zhang, Zheng Zhang, Jinchao Li, Baolin Peng, Xiujun Li, Minlie Huang, et al. 2019. Convlab: Multi-domain end-to-end dialog system platform. *arXiv preprint arXiv:1904.08637*.

Wenqiang Lei, Xisen Jin, Min-Yen Kan, Zhaochun Ren, Xiangnan He, and Dawei Yin. 2018. Sequicity: Simplifying task-oriented dialogue systems with single sequence-to-sequence architectures. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1437–1447.

Xiujun Li, Zachary C Lipton, Bhuwan Dhingra, Lihong Li, Jianfeng Gao, and Yun-Nung Chen. 2016. A user simulator for task-completion dialogues. *arXiv preprint arXiv:1612.05688*.

Bing Liu and Ian Lane. 2017. Iterative policy learning in end-to-end trainable task-oriented neural dialog models. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 482–489. IEEE.

Chia-Wei Liu, Ryan Lowe, Iulian V Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *arXiv preprint arXiv:1603.08023*.

Alexander H Miller, Will Feng, Adam Fisch, Jiasen Lu, Dhruv Batra, Antoine Bordes, Devi Parikh, and Jason Weston. 2017. Parlai: A dialog research software platform. *arXiv preprint arXiv:1705.06476*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Olivier Pietquin and Helen Hastie. 2013. A survey on metrics for the evaluation of user simulations. *The knowledge engineering review*, 28(1):59–73.

Kun Qian and Zhou Yu. 2019. Domain adaptive dialog generation via meta learning. *arXiv preprint arXiv:1906.03520*.

Antoine Raux, Brian Langner, Dan Bohus, Alan W Black, and Maxine Eskenazi. 2005. Let's go public! taking a spoken dialog system to the real world. In *Ninth European conference on speech communication and technology*.

Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young. 2007. Agenda-based user simulation for bootstrapping a pomdp dialogue system. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 149–152. Association for Computational Linguistics.

Jost Schatzmann, Karl Weilhammer, Matt Stuttle, and Steve Young. 2006. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *The knowledge engineering review*, 21(2):97–126.

Jost Schatzmann and Steve Young. 2009. The hidden agenda user simulation model. *IEEE transactions on audio, speech, and language processing*, 17(4):733–747.

Jost Schatztnann, Matthew N Stuttle, Karl Weilhammer, and Steve Young. 2005. Effects of the user model on simulation-based learning of dialogue strategies. In *IEEE Workshop on Automatic Speech Recognition and Understanding, 2005.*, pages 220–225. IEEE.

Pararth Shah, Dilek Hakkani-Tür, Gokhan Tür, Abhinav Rastogi, Ankur Bapna, Neha Nayak, and Larry Heck. 2018. Building a conversational agent overnight with dialogue self-play. *arXiv preprint arXiv:1801.04871*.

Weiyan Shi and Zhou Yu. 2018. Sentiment adaptive end-to-end dialog systems. *arXiv preprint arXiv:1804.10731*.

Weiyan Shi, Tiancheng Zhao, and Zhou Yu. 2019. Unsupervised dialog structure learning. *arXiv preprint arXiv:1904.03736*.

Pei-Hao Su, Pawel Budzianowski, Stefan Ultes, Milica Gasic, and Steve Young. 2017. Sample-efficient actor-critic reinforcement learning with supervised data for dialogue management. *arXiv preprint arXiv:1707.00130*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Michel Tokic. 2010. Adaptive $\varepsilon$-greedy exploration in reinforcement learning based on value differences. In *Annual Conference on Artificial Intelligence*, pages 203–210. Springer.

Van-Khanh Tran and Le-Minh Nguyen. 2017. Semantic refinement gru-based neural language generation for spoken dialogue systems. In *International Conference of the Pacific Association for Computational Linguistics*, pages 63–75. Springer.

Van-Khanh Tran and Le-Minh Nguyen. 2018. Adversarial domain adaptation for variational neural language generation in dialogue systems. *arXiv preprint arXiv:1808.02586*.

Tsung-Hsien Wen, Milica Gasic, Dongho Kim, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. 2015a. Stochastic language generation in dialogue using recurrent neural networks with convolutional sentence reranking. *arXiv preprint arXiv:1508.01755*.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. 2015b. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745*.

Jason D Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. *arXiv preprint arXiv:1702.03274*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Yu Wu, Wei Wu, Chen Xing, Ming Zhou, and Zhoujun Li. 2016. Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots. *arXiv preprint arXiv:1612.01627*.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

Tiancheng Zhao, Kaige Xie, and Maxine Eskenazi. 2019. Rethinking action spaces for reinforcement learning in end-to-end dialog agents with latent variable models. *arXiv preprint arXiv:1902.08858*.

# Low-Rank HOCA: Efficient High-Order Cross-Modal Attention for Video Captioning

**Tao Jin, Siyu Huang*, Yingming Li, and Zhongfei Zhang**
College of Information Science & Electronic Engineering
Zhejiang University, China
{jint_zju, siyuhuang, yingming, zhongfei}@zju.edu.cn

## Abstract

This paper addresses the challenging task of video captioning which aims to generate descriptions for video data. Recently, the attention-based encoder-decoder structures have been widely used in video captioning. In existing literature, the attention weights are often built from the information of an individual modality, while, the association relationships between multiple modalities are neglected. Motivated by this observation, we propose a video captioning model with High-Order Cross-Modal Attention (HOCA) where the attention weights are calculated based on the high-order correlation tensor to capture the frame-level cross-modal interaction of different modalities sufficiently. Furthermore, we novelly introduce Low-Rank HOCA which adopts tensor decomposition to reduce the extremely large space requirement of HOCA, leading to a practical and efficient implementation in real-world applications. Experimental results on two benchmark datasets, MSVD and MSR-VTT, show that Low-rank HOCA establishes a new state-of-the-art.

## 1 Introduction

Video captioning has drawn much attention from natural language processing and computer vision researchers (Venugopalan et al., 2014; Bin et al., 2016; Ramanishka et al., 2016; Zanfir et al., 2016). As videos typically consist of multiple modalities (image, motion, audio, etc.), video captioning is actually a multimodal learning task. The abundant information underlying in the modalities is much beneficial for video captioning models. However, how to effectively learn the association relationships between different modalities is still a challenging problem.

In the context of deep learning based video captioning, multimodal attention mechanisms (Xu et al., 2017; Hori et al., 2017; Wang et al., 2018c) are shown to help deliver superior performance. Bahdanau Attention (Bahdanau et al., 2014) is widely used to calculate the attention weights according to the features of individual modalities, since Bahdanau Attention is originally proposed for machine translation which can be considered as a unimodal task. However, video captioning is a multimodal task and different modalities are able to provide complementary cues to each other when calculating the attention weights. For example, as shown in Fig. 1, a part of the video shows that two men are wrestling and another part shows that the referee coaches the match. With only the image modality, the model may pay equal attention to the frames with the competitors and referee. Considering the additional motion modality, the attention mechanism weighs more on one of them, resulting in a model with more focused attention.



***GT1***: a referee coaches a wrestling match
***GT2***: boys are wrestling in front of a crowd

Figure 1: An example of video captioning, where "GT" denotes ground truth.

Motivated by the above observations, in this paper we propose an attention mechanism called High-Order Cross-Modal Attention (HOCA) for video captioning, which makes a full use of the different modalities in video data by capturing their structure relationships. The key idea of HOCA is to consider the information of the other modalities when calculating the attention weights for one modality, different from Bahdanau Attention. In addition, we propose a low-rank version of HOCA which significantly reduces the com-

*Corresponding author.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2001–2011, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

putational complexity of HOCA. Specifically, the attention weights of HOCA are computed based on the similarity tensor between modalities, fully exploiting the correlation information of different modalities at each time step. Given the fact that the space requirement of the high-order tensor increases exponentially in the number of modalities and inspired by tensor decomposition, we adopt a low-rank correlation structure across modalities into HOCA to enable a good scalability to the increasing number of modalities. Such improvement largely reduces the algorithm complexity of HOCA with good results in empirical study.

Our contributions can be summarized as:

(1) We propose High-Order Cross-Modal Attention (HOCA), which is a novel multimodal attention mechanism for video captioning. Compared with the Bahdanau Attention, HOCA captures the frame-level interaction between different modalities when computing the attention weights, leading to an effective multimodal modeling.

(2) Considering the scalability to the increasing number of modalities, we propose Low-Rank HOCA, which employs tensor decomposition to enable an efficient implementation of High-Order Cross-Modal Attention.

(3) Experimental results show that our method outperforms the state-of-the-art methods on video captioning, demonstrating the effectiveness of our method. In addition, the theoretical and experimental complexity analyses show that Low-Rank HOCA can implement multimodal correlation effeciently with acceptable computing cost.

## 2 Related Work

### 2.1 Attention Mechanism

The encoder-decoder structures have been widely used in sequence transformation tasks. Some models also connect the encoder and decoder through an attention mechanism(Bahdanau et al., 2014; Luong et al., 2015). In natural language processing (NLP), (Bahdanau et al., 2014) first proposes the soft attention mechanism to adaptively learn the context vector of the target keys/values. Such attention mechanisms are used in conjunction with a recurrent network. In the context of video captioning, there are many methods (Hori et al., 2017; Xu et al., 2017) using Bahdanau Attention to learn the context vector of the temporal features of video data, including the recent state-of-the-art method, hierarchically aligned cross-modal attentive network (HACA) (Wang et al., 2018c). Such method ignores the structure relationships between modalities when computing the attention weights.

### 2.2 Video Captioning

Compared with image data, video data have rich multimodal information, such as image, motion, audio, semantic object, text. Therefore, the key is how to utilize the information. In the literature of video captioning, (Xu et al., 2017; Hori et al., 2017) propose the hierarchical multimodal attention, which selectively attends to a certain modality when generating descriptions. (Shen et al., 2017; Gan et al., 2017) adopt multi-label learning with weak supervision to extract semantic features of video data. (Wang et al., 2018b) proposes optimizing the metrics directly with hierarchical reinforcement learning. (Chen et al., 2017) extracts five types of features to develop the multimodal video captioning method and achieves promising results. (Wang et al., 2018a) performs the reconstruction of input video features by utilizing the output of the decoder, increasing the consistency of descriptions and video features. (Wang et al., 2018c) proposes the hierarchically aligned multimodal attention (HACA) to selectively fuse both global and local temporal dynamics of different modalities.

None of the methods mentioned above utilizes the interaction information of different modalities to calculate the attention weights. Motivated by this observation, we present HOCA and Low-Rank HOCA for video captioning. Different from the widely used Bahdanau Attention, our methods fully exploits the video representation of different modalities and their frame-level interaction information when computing the attention weights.

We introduce our methods in the following sections, where Section 3 introduces the details of Bahdanau Attention based multimodal video captioning (background), Section 4 gives the derivations of HOCA and Low-Rank HOCA, and the encoder-decoder structure which we propose for video captioning. Section 5 and 6 show the experimental settings and results of our methods.

## 3 Multimodal Video Captioning

Encoder-decoder structures combined with Bahdanau Attention are widely used in multimodal video captioning. Suppose that the num-

ber of input modalities is $n$, $I_l$ denotes the features of $l$-th modality, the space of $I_l$ is $\mathbb{R}^{d_l \times t_l}$ where $t_l$ denotes the temporal length and $d_l$ denotes the feature dimensions. The corresponding output is word sequence.

### 3.1 Encoder

The input video is fed to multiple feature extractors, which can be pre-trained CNNs for classification tasks such as Inception-Resnet-v2(Szegedy et al., 2017), I3D(Carreira and Zisserman, 2017), VGGish(Hershey et al., 2017), each extractor corresponds to one modality. The extracted features are sent to Bi-LSTM (Hochreiter and Schmidhuber, 1997) which has a capability to process sequence data, capturing the information from both forward and backward directions. The output of Bi-LSTM is kept.

### 3.2 Decoder

The decoder aims to generate word sequence by utilizing the features provided by the encoder and the context information. At time step $t$, the decoder output can be obtained with the input word $y_t$ and the previous output $h_{t-1}$,

$$h_t = \mathrm{LSTM}(h_{t-1}, y_t) \tag{1}$$

We treat $h_t$ as query $q$, the features of different modalities are allocated weights with Bahdanau Attention seperately as shown in Fig. 2(a). The attention weights $\alpha_t^{I_l} \in \mathbb{R}^{t_l}$ of $l$-th modality can be obtained as follows:

$$(\alpha_t^{I_l})_{r_l} = \frac{\exp\left\{(e_t^{I_l})_{r_l}\right\}}{\sum_{n=1}^{t_l} \exp\left\{(e_t^{I_l})_n\right\}} \tag{2}$$

$$(e_t^{I_l})_{r_l} = w_1^{\mathbb{T}} \tanh\left[W_1 h_t + U_1 (I_l)_{r_l} + b_1\right] \tag{3}$$

where $(I_l)_{r_l}$ denotes the $r_l$-th time step of $I_l$ and $(\alpha_t^{I_l})_{r_l}$ is the corresponding weight, we combine the attention weights and features, obtaining the context vector $\varphi_t(I_l)$ of $l$-th modality,

$$\sum_{r_l=1}^{t_l} (\alpha_t^{I_l})_{r_l} = 1 \tag{4}$$

$$\varphi_t(I_l) = \sum_{r_l=1}^{t_l} (\alpha_t^{I_l})_{r_l} (I_l)_{r_l} \tag{5}$$

the context vectors of other modalities can be obtained in the same way. We then integrate the context vectors to predict the word as follows:

$$p_t = \mathrm{softmax}\left[W_p^h h_t + \sum_{i=1}^{n} W_p^{I_i} \varphi_t(I_i) + b_p\right] \tag{6}$$

where $n$ is the number of modalities, $p_t$ denotes the probability distribution of words at time step $t$.

## 4 HOCA and Low-Rank HOCA

Considering multimodal features, Bahdanau Attention and its variants process different modalities separately. In such situation, the interaction between different modalities is ignored. We propose HOCA and Low-Rank HOCA to excavate this information, in addition, the tensor decomposition used in Low-Rank HOCA reduces the complexity of the high-order correlation.

### 4.1 High-Order Cross-Modal Attention (HOCA)

Different modalities in video data are able to provide complementary cues for each other and should be integrated effectively. Inspired by the correlation matrix which is widely used in the natural language understanding (Seo et al., 2016), we use a high-order correlation tensor to model the interaction between different modalities. Fig. 2(b) shows the generalized form of the structure for $n$ modalities, the core module is "tensor multiplication" defined by ourselves. After the nonlinear mapping similar to Eqn. 3 with query $h_t$, the features of the $n$ modalities are in a $d$-dimensional common space. Note that, for convenience, we still use $I_l$ to represent the features of the $l$-th modality (instead of $I_{l,t}$) and omit $t$ which denotes the time step of the decoder in the following derivations of Section 4.1 and 4.2.

Let $\alpha_n^{I_l}$ denote the target attention weights of modality $I_l$. We obtain $\alpha_n^{I_l}$ through the high-order correlation tensor $C_n$ between the $n$ modalities. $C_n$ can be obtained as below:

$$\begin{aligned}(C_n)_{r_1,\ldots,r_n} &= \mathbf{1}_d\left[(I_1)_{r_1} \circ (I_2)_{r_2} \circ \ldots \circ (I_n)_{r_n}\right] \\ &= \mathbf{1}_d\left[\bigwedge_{i=1}^{n} (I_i)_{r_i}\right]\end{aligned} \tag{7}$$

$$C_n = \bigotimes \left\{I_1, I_2, \ldots, I_n\right\} = \bigotimes_{i=1}^{n} I_i \tag{8}$$

where the $(r_1, \ldots, r_n)$-th entry of $C_n$ is the inner-product of the $r_1$-th column (time step) of $I_1$, $r_2$-th column (time step) of $I_2$,...,$r_n$-th column (time step) of $I_n$. $\circ$ denotes element-wise multiplication and $\Lambda$ denotes the element-wise multiplication $\circ$

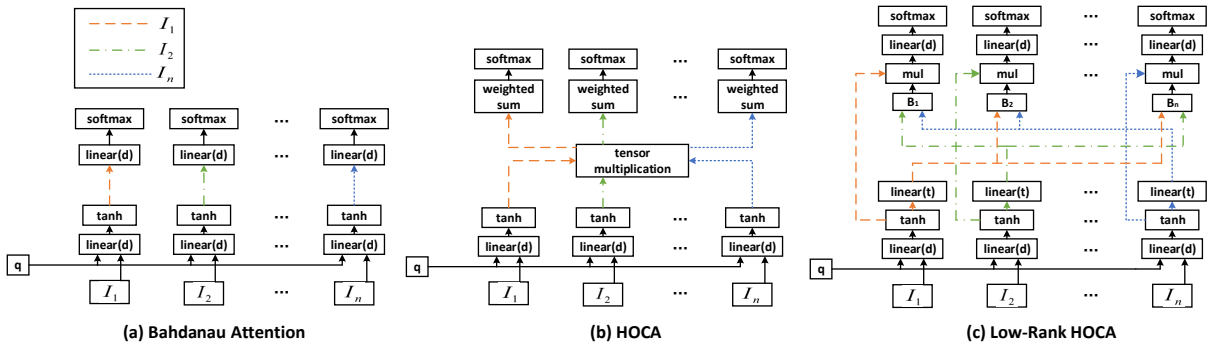


Figure 2: The attention mechanisms for multiple modalities where $n$ denotes the number of modalities. (a) shows Bahdanau Attention for different modalities; $I_l$ represents the $l$-th modality; linear(d) denotes the linear layer connected to the first dimension of $I_l$. (b) is the detailed structure of HOCA where the core module is "tensor multiplication" defined by ourselves. (c) is Low-Rank HOCA, where linear(t) denotes the linear layer in temporal (second) dimension; "mul" denotes multiplication operation; $B_l$ is defined in Eqn. 16.

for a sequence of tensors. $\bigotimes$ is the tensor product operator and we use it to define a new operation $\bigotimes\{,...,\}$ for the "tensor multiplication" of multiple matrices. $\mathbf{1}_d$ with space $\mathbb{R}^d$ consists of 1. We use the vectors $\mathbf{1}_d$ and $\mathbf{1}_{t_i}$ to denote the summation operation along the $d$ and $t_i$ dimensions. In this situation, Eqn. 7 and Eqn. 8 are equivalent. The attention weights of $I_l$ can be calculated as below:

$$(\alpha_n^{I_l})_{r_l} = \frac{\exp\Big\{\sum\Big[W_{n-1}^{I_l} \circ (C_n^{I_l})_{r_l}\Big]\Big\}}{\sum_{o=1}^{t_l}\exp\Big\{\sum\Big[W_{n-1}^{I_l} \circ (C_n^{I_l})_{o}\Big]\Big\}} \tag{9}$$

where $(C_n^{I_l})_{r_l}$ is equal to $(C_n)_{:,:,...,r_l,...,:,:}$, which is an ($n$-1)-order tensor and denotes the correlation values between the $r_l$-th time step of $I_l$ and different time steps of the other $n$-1 modalities. $W_{n-1}^{I_l}$ is also an ($n$-1)-order tensor, which has the same shape and denotes the relative importance of the correlation values. $\sum$ denotes the summation function for the high-order tensor. For simplicity, the "weighted sum" module can be considered as a linear layer in multiple dimensions.

### 4.2 Low-Rank HOCA

One of the main drawbacks of HOCA is the generation of high-order tensor, the size of the high-order tensor will increases exponentially with the number of modalities as $\prod_{i=1}^{n} t_i$, resulting in a lot of computation. Therefore, we implement the multimodal correlation between the different modalities in a more efficient way with low-rank approximation which has been widely used in the community of vision and language (Lei et al., 2015; Liu et al., 2018; Yu et al., 2017). Following the Eqn. 7 and 8, we rewrite $(C_n^{I_l})_{r_l}$ as[1]:

$$\begin{aligned}(C_n^{I_l})_{r_l} &= \bigotimes\Big\{(I_l)_{r_l} \circ I_1, ..., I_{l-1}, I_{l+1}, ..., I_n\Big\} \\ &= \bigotimes_{i=1,\neq l}^{n} (I_l)_{r_l} \star I_i\end{aligned} \tag{10}$$

Element-wise multiplication operator $\circ$ is used for vector $(I_l)_{r_l}$ and matrix $I_1$. Each column of matrix $I_1$ multiplies vector $(I_l)_{r_l}$. $\star$ denotes that $(I_l)_{r_l}$ is multiplied ($\circ$) only when $i = 1$. We assume a low-rank factorization of the tensor $W_{n-1}^{I_l}$. Specifically, $W_{n-1}^{I_l}$ is decomposed into a sum of $k$ rank 1 tensors,

$$W_{n-1}^{I_l} = \sum_{j=1}^{k}\bigotimes_{i=1,\neq l}^{n} w_j^{I_i} \tag{11}$$

where the space of $w_j^{I_i}$ is $\mathbb{R}^{1\times t_i}$. Note that we set $k$ to a constant value and use the recovered low-rank tensor to approximate $W_{n-1}^{I_l}$. The numerator in Eqn. 9 can be further derived as:

$$\begin{aligned}\sum\Big[W_{n-1}^{I_l} \circ (C_n^{I_l})_{r_l}\Big] = \sum\Big[\sum_{j=1}^{k}\bigotimes_{i=1,\neq l}^{n} w_j^{I_i} \\ \circ \bigotimes_{i=1,\neq l}^{n} (I_l)_{r_l} \star I_i\Big]\end{aligned} \tag{12}$$

Since $I_i$ is a matrix and $w_j^{I_i}$ is a vector, we directly multiply $I_i$ and corresponding $w_j^{I_i}$ [1]. Each row of the matrix $I_i$ multiplies vector $w_j^{I_i}$,

$$\sum\Big[W_{n-1}^{I_l} \circ (C_n^{I_l})_{r_l}\Big] = \sum_{j=1}^{k}\Big[\sum\bigotimes_{i=1,\neq l}^{n} (I_l)_{r_l} \star I_i \circ w_j^{I_i}\Big] \tag{13}$$

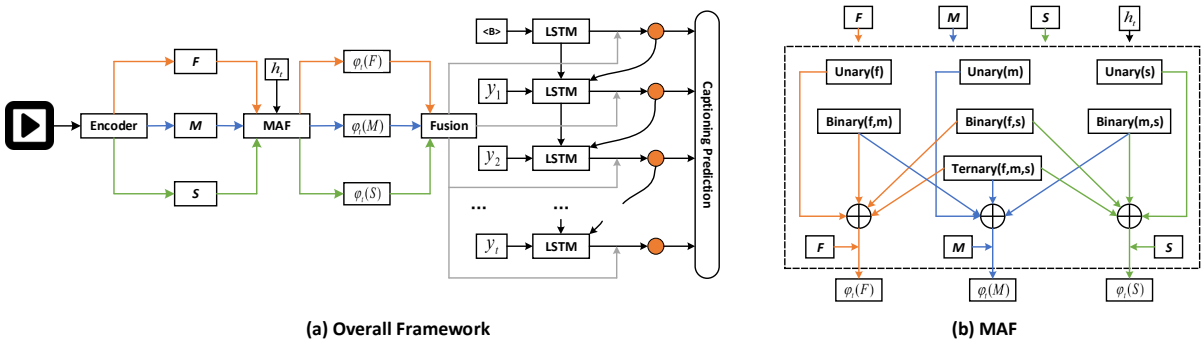


Figure 3: (a) is the overall framework of video captioning where F, M, S represent image, motion, and audio, respectively; $h_t$ denotes the query at time step $t$. We predict words at all the time steps of the decoder with multiple attentive fusion (MAF) module. (b) is the detailed structure of MAF module, each modality has three types of attention weights (unary, binary, and ternary), where "Ternary" denotes that the number of modalities is 3 in HOCA and Low-Rank HOCA, "Binary" and "Unary" correspond to 2 and 1. The unary attention is equal to Bahdanau Attention. In addition, "Binary(f,m)" denotes the binary attention weights with image and motion. We utilize trainable parameters to determine the importance of these attention weights when integrating them.

where we first apply the tensor multiplication $\bigotimes$ to correlating the different time steps of all modalities. During the process, we sum in the $d$ dimension to obtain the elements of the high-order tensor. Second, we sum (inner $\sum$) all the elements. For convenience, we change the operation order, we first sum in the temporal dimension, then sum in the $d$ dimension[1]. Letting $(I_i^{'})_j$ denote the global information of $I_i$ with importance factor $w_j^{I_i}$,

$$(I_i^{'})_j = (I_i \circ w_j^{I_i})\mathbf{1}_{t_i} = I_i(w_j^{I_i})^{\mathbb{T}} \tag{14}$$

Eqn. 12 can be further derived as below:

$$\sum \left[W_{n-1}^{I_l} \circ (C_n^{I_l})_{r_l}\right] = \mathbf{1}_d \left[(I_l)_{r_l} \circ B_l\right] \tag{15}$$

$$B_l = \sum_{j=1}^{k} \bigwedge_{i=1,\neq l}^{n} (I_i^{'})_j \tag{16}$$

Due to the different information carried by the elements of the feature, we use a linear layer $w_{I_l}$ to replace $\mathbf{1}_d$,

$$(\alpha_n^{I_l})_{r_l} = \frac{\exp\left\{w_{I_l}\left[(I_l)_{r_l} \circ B_l\right]\right\}}{\sum_{o=1}^{t_l} \exp\left\{w_{I_l}\left[(I_l)_o \circ B_l\right]\right\}} \tag{17}$$

The detailed structure of Low-Rank HOCA is shown in Fig. 2(c).

[1]The details of the propositions are shown in the supplementary materials.

### 4.3 Complexity Analysis

We analyze the space complexity of Bahdanau Attention, HOCA, and Low-Rank HOCA in Fig. 2, focusing on the trainable variables and the output of each layer. For convenience, we start by calculating from the output of $tanh$ layer, since the front structures of three methods are same.

**Bahdanau Attention** The size of the trainable variable in second $linear(d)$ layer is $d \times 1$, and the size of the output is $\sum_{i=1}^{n} t_i$. The space complexity is $O(\sum_{i=1}^{n} t_i + d)$.

**HOCA** The size of the output of tensor multiplication is $\prod_{i=1}^{n} t_i$. The size of the trainable variables and corresponding output in $weighted\ sum$ layer is $\sum_{i=1}^{n}(\frac{\prod_{j=1}^{n} t_j}{t_i}) + \sum_{i=1}^{n} t_i$. The space complexity is $O(\prod_{i=1}^{n} t_i + \sum_{i=1}^{n}(\frac{\prod_{j=1}^{n} t_j}{t_i}) + \sum_{i=1}^{n} t_i)$.

**Low-Rank HOCA** The rank is set to $k$. The size of the trainable variable and corresponding output in $linear(t)$ layer is $\sum_{i=1}^{n} t_i \times k + n \times k \times d$. The size of the output in $B_l$ and $mul$ layer is $n \times d + \sum_{i=1}^{n} t_i \times d$. The size of the trainable variable and the corresponding output in second $linear(d)$ layer is $d + \sum_{i=1}^{n} t_i$. The space complexity is $O(\sum_{i=1}^{n} t_i \times (k + d + 1) + (n \times k + n + 1) \times d)$.

Bahdanau Attention and Low-Rank HOCA both scale linearly in the number of modalities while HOCA scales exponentially. Therefore, HOCA will have explosive complexity when the number of modalities is big, and Low-Rank HOCA can solve this problem effectively.

Table 1: Evaluation results of our proposed models

| Method | MSVD | | | | MSR-VTT | | | |
|---|---|---|---|---|---|---|---|---|
| | BLEU4 | ROUGE | METEOR | CIDEr | BLEU4 | ROUGE | METEOR | CIDEr |
| HOCA-U | 50.6 | 70.6 | 34.3 | 83.7 | 41.9 | 61.2 | 29.1 | 47.2 |
| HOCA-B | 49.6 | 70.8 | 34.3 | 84.3 | 42.2 | 61.1 | 29.1 | 47.6 |
| L-HOCA-B | 50.1 | 70.4 | 34.8 | 84.7 | 42.4 | 61.4 | 29.0 | 47.9 |
| HOCA-T | 50.9 | 71.2 | 34.8 | 84.8 | 42.4 | 61.5 | 29.0 | 48.1 |
| L-HOCA-T | 51.3 | 71.5 | 34.7 | 84.8 | 43.2 | 61.5 | 29.1 | 48.4 |
| HOCA-UB | 50.6 | 71.1 | 34.0 | 85.7 | 43.1 | 61.5 | 29.0 | 48.0 |
| L-HOCA-UB | 51.7 | 71.9 | 34.7 | **86.1** | 43.5 | 62.3 | 29.2 | 48.5 |
| HOCA-UBT | 52.3 | 71.9 | **35.5** | 85.3 | 43.9 | 62.0 | 29.0 | 49.3 |
| L-HOCA-UBT | **52.9** | **72.0** | **35.5** | **86.1** | **44.6** | **62.6** | **29.5** | **49.8** |

### 4.4 Video Captioning with HOCA and Low-Rank HOCA

In this section, we mainly introduce our encoder-decoder structure combined with high-order attention (HOCA and Low-Rank HOCA). As shown in Fig. 3(a), the features of different modalities, i.e. image(F), motion(M), audio(S) are extracted in the encoder. These features are sent to the decoder for generating words. The "MAF" module performs HOCA and Low-Rank HOCA for the features of different modalities with $h_t$. As shown in Fig. 3(b), each modality has three types of attention weights, i.e. unary, binary, ternary, which denote the different number (i.e. 1,2,3) of modalities applied to the HOCA and Low-Rank HOCA (note that unary attention is equal to Bahdanau Attention). In some cases, not all the modalities are effective, for example, the binary attention weights of image and motion are more accurate for the salient videos.

We use $\alpha_{t,3}^{F}$, $\alpha_{t,3}^{M}$, $\alpha_{t,3}^{S}$ to denote the ternary weights of three modalities at time step $t$, the calculation is shown as follows:

$$\alpha_{t,3}^{F}, \alpha_{t,3}^{M}, \alpha_{t,3}^{S} = \mathrm{HOCA}(h_t, F, M, S) \tag{18}$$

where $F$,$M$,$S$ denote the features of image, motion, audio, respectively, and HOCA can be replaced by Low-Rank HOCA. We obtain the binary and unary weights in the same way. For different attention weights, we utilize trainable variables to determine their importance. We take the image modality as an example, the fusion weights are obtained as follows:

$$\alpha_t^F = \mathrm{softmax}(\theta_1 \alpha_{\mathrm{t,1}}^{\mathrm{F}} + \theta_2 \alpha_{\mathrm{t,2}}^{\mathrm{F,M}} + \theta_3 \alpha_{\mathrm{t,2}}^{\mathrm{F,S}} + \theta_4 \alpha_{\mathrm{t,3}}^{\mathrm{F}}) \tag{19}$$

where $\theta_{1-4}$ are trainable variables, $\alpha_{t,1}^{F}$ denotes the unary weights, $\alpha_{t,2}^{F,M}$ and $\alpha_{t,2}^{F,S}$ denote the binary weights with motion and audio, respectively. As we obtain the attention weights of three modalities, we can calculate the context vectors $\varphi_t(F)$, $\varphi_t(M)$, $\varphi_t(S)$ following the Eqn. 4 and 5.

To further determine the relative importance of multiple modalities, we perform a hierarchical attention mechanism for the context vectors in the "Fusion" module. The attention weights of image modality are calculated as follows:

$$\beta_t^F = \frac{\exp(e_t^F)}{\sum_{k=\{F,M,S\}} \exp(e_t^k)} \tag{20}$$

$$e_t^F = w_e^{\mathbb{T}} \tanh\Big[W_e h_t + U_e \varphi_t(F) + b_e\Big] \tag{21}$$

$\beta_t^M$ and $\beta_t^S$ are obtained in the same way. We integrate the context vectors and corresponding weights to predict word as follows:

$$p_t = \mathrm{softmax}\Big[W_p^h h_t + \sum_k \beta_t^k D_t(k) + b_p\Big] \tag{22}$$

$$D_t(k) = W_p^k \varphi_t(k), k = \{F, M, S\} \tag{23}$$

The optimization goal is to minimize the cross-entropy loss function defined as the accumulative loss from all the time steps:

$$L = -\sum_{i=1}^{T} \log(p_t^* | p_1^*, ..., p_{t-1}^*, V) \tag{24}$$

where $p_t^*$ denotes the probability of the ground-truth word at time step $t$, $T$ denotes the length of description, $V$ denotes the original video data.

## 5 Experimental Methodology

### 5.1 Datasets and Metrics

We evaluate video captioning on two standard datasets, MSVD (Chen and Dolan, 2011) and MSR-VTT (Xu et al., 2016), which are both provided by Microsoft Research, with several state-of-the-art methods. MSVD includes 1970 video clips. The time length of a video clip is about 10 to 25 seconds and each video clip is annotated with about 40 English sentences. MSR-VTT has 10000 video clips; each clip is annotated with 20 English sentences. We follow the commonly used protocol in the previous work and use four common metrics in the evaluation, including BLEU4, ROUGE, METEOR, and CIDEr.

### 5.2 Preprocessing and Experimental Setting

We sample video data to 80 frames for extracting image features. For extracting motion features, we divide the raw video data into video chunks centered on 80 sampled frames at the first step. Each video chunk includes 64 frames. For extracting audio features, we obtain the audio file from the raw video data with FFmpeg. For both datasets, we use a pre-trained Inception-ResNet-v2 (Szegedy et al., 2017) to extract image features from the sampled frames and we keep the activations from the penultimate layer. In addition, we use a pre-trained I3D (Carreira and Zisserman, 2017) to extract motion features from video chunks. We employ the activations from the last convolutional layer and implement a mean-pooling in the temporal dimension. We use the pre-trained VGGish (Hershey et al., 2017) to extract audio features. On MSRVTT, we also utilize the glove embedding of the auxiliary video category labels to initialize the decoder state.

The hidden size is 512 for all LSTMs. The attention layer size for image, motion, audio attention is also 512. The dropout rate for the input and output of LSTM decoder is 0.5. The rank is set to 1. In the training stage, we use Adam (Kingma and Ba, 2014) algorithm to optimize the loss function; the learning rate is set to 0.0001. In the testing stage, we use beam-search method with beam-width 5. We use a pre-trained word2vec embedding to initialize the word vectors. Each word is represented as a 300-dimension vector. Those words which are not in the word2vec matrix are initialized randomly. All the experiments are done on 4 GTX 1080Ti GPUs.

## 6 Experimental Results

### 6.1 Impact of Cross-Modal Attention

Table 2 shows the results of different variants of HOCA and Low-Rank HOCA. HOCA-U, HOCA-B, and HOCA-T denote the model with only unary, binary, and ternary attention, respectively. HOCA-UB and HOCA-UBT denote the models with original HOCA and more types of attention mechanisms. The prefix "L-HOCA" denotes the model with Low-Rank HOCA.

It is observed that the model with only one type of attention mechanism(U, B, or T) in the decoder achieves relatively bad results on both datasets. However, when we combine them, the performances are significantly improved on metrics, especially ROUGE and CIDEr. HOCA-UBT and L-HOCA-UBT with a combination of unary, binary, and ternary attention achieve relatively promising results on all the metrics. We argue that HOCA-UBT and L-HOCA-UBT can learn appropriate ratios of all types of attention mechanisms based on the specific video-description pairs, while other variants only focus on one or two types. In addition, the models of low-rank version (L-HOCA-UB and L-HOCA-UBT) have better metrics than the models of original version (HOCA-UB and HOCA-UBT). On the one hand, we utilize $w_{I_l}$ to replace $\mathbf{1}_d$ in Eqn. 17, fully mining the different information carried by the elements of the feature, on the other hand, the low-rank approximation is effective.

Table 2: Evaluation results of video captioning, where B, R, M, C denote BLEU4, ROUGE, METEOR, CIDEr, respectively, and Ours denotes L-HOCA-UBT.

| Dataset | Method | B | R | M | C |
|---|---|---|---|---|---|
| MSR-VTT | RecNet | 39.1 | 59.3 | 26.6 | 42.7 |
| | HRL | 41.3 | 61.7 | 28.7 | 48.0 |
| | Dense Cap | 41.4 | 61.1 | 28.3 | 48.9 |
| | HACA | 43.5 | 61.8 | **29.5** | 49.7 |
| | MM-TGM | 44.3 | - | 29.3 | 49.2 |
| | Ours | **44.6** | **62.6** | **29.5** | **49.8** |
| MSVD | SCN | 50.2 | - | 33.4 | 77.7 |
| | TDDF | 45.8 | 69.7 | 33.3 | 73.0 |
| | LSTM-TSA | 52.8 | - | 33.5 | 74 |
| | RecNet | 52.3 | 69.8 | 34.1 | 80.3 |
| | MM-TGM | 48.7 | - | 34.3 | 80.4 |
| | Ours | **52.9** | **72.0** | **35.5** | **86.1** |



**HOCA-U**: a man is talking about a car
**L-HOCA-UBT**: a man is talking about the features of a car
**GT**: a guy talks about the features of the jeep Cherokee



**HOCA-U**: a woman is peeling a piece of raw meat
**L-HOCA-UBT**: a woman is peeling a shrimp
**GT**: a woman is peeling a shrimp



**HOCA-U**: a person is playing with an animal
**L-HOCA-UBT**: a person is playing with a cat
**GT**: the boy is playing the cat



**HOCA-U**: a person is playing a bike
**L-HOCA-UBT**: a man is demonstrating how to use a stroller
**GT**: a man displaying features on a stroller

Figure 4: Qualitative results of Low-Rank HOCA

### 6.2 Compared with the State-of-the-art

Table 2 shows the results of different methods on MSR-VTT and MSVD, including ours (L-HOCA-UBT), and some state-of-the-art methods, such as LSTM-TSA (Pan et al., 2017), TDDF (Zhang et al., 2017), SCN (Gan et al., 2017), MM-TGM (Chen et al., 2017), Dense Caption (Shen et al., 2017), RecNet (Wang et al., 2018a), HRL (Wang et al., 2018b),HACA(Wang et al., 2018c).

From Table 2, we find that Ours(L-HOCA-UBT) shows competitive performances compared with the state-of-the-art methods. On MSVD, L-HOCA-UBT has outperformed SCN, TDDF, RecNet, LSTM-TSA, MM-TGM, on all the metrics. In particular, L-HOCA-UBT achieves 86.1% on CIDEr, making an improvement of 5.7% over MM-TGM. On MSR-VTT, we have the similar observation, L-HOCA-UBT has outperformed RecNet, HRL, MM-TGM, Dense Caption, and HACA on all the metrics.

Table 3: Computing cost of different methods, where the "space" denotes the memory space requirement and the "training time" denotes the total time for training. We evaluate them on MSR-VTT. Note that the metrics belong to the whole model, not only the attention module.

| Method | Space (G) | Training Time (s) |
|---|---|---|
| HOCA-U | 4.1 | 16418 |
| HOCA-UBT | 9.7 | 37040 |
| L-HOCA-UBT | 5.6 | 24615 |

### 6.3 Computing Cost

The theoretical complexity of different attention mechanisms is illustrated in Section 4.3. In practice, we utilize the experimental settings mentioned above, the batch size and the maximum number of epochs are set to 25 and 100, respectively. the training time and memory space requirement are shown in Table 3. We can find that L-HOCA-UBT has smaller space requirement and less time cost than HOCA-UBT, in addition, the computing cost of L-HOCA-UBT is close to that of HOCA-U (Bahdanau Attention). The results demonstrate the advantage of Low-rank HOCA.

### 6.4 Rank Setting

We also evaluate the impact of different rank values. We show the results on MSVD in Fig. 5. The red and green lines represent HOCA-UBT and L-HOCA-UBT, respectively. We find that the CIDEr of L-HOCA-UBT has slight fluctuations as the rank changes and a small value of rank can achieve competitive results with high efficiency.

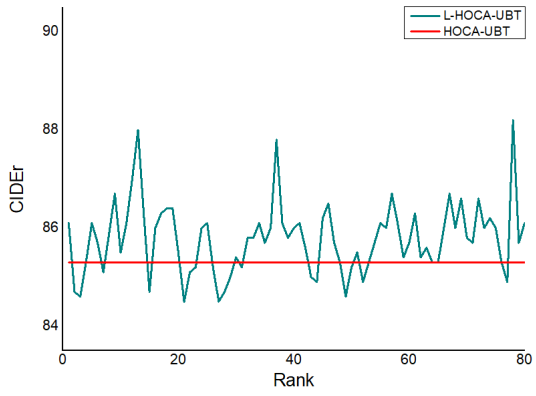


Figure 5: The performance of different rank values

### 6.5 Qualitative Analysis

Fig. 4 shows some qualitative results of our method. We simply compare the descriptions generated by HOCA-U and L-HOCA-UBT, respectively. GT represents "Ground Truth". Benefiting from the high-order correlation of multiple modalities, L-HOCA-UBT can generate more accurate descriptions which are close to GT.

## 7 Conclusion

In this paper, we have proposed a new cross-modal attention mechanism called HOCA for video captioning. HOCA integrates the information of the other modalities into the inference of attention weights of current modality. Furthermore, we have introduced the Low-Rank HOCA which has a good scalability to the increasing number of modalities. The experimental results on two standard datasets have demonstrated the effectiveness of our approach.

## Acknowledgments

This work was supported in part by National Natural Science Foundation of China (No. 61702448, 61672456), Zhejiang Lab (2018EC0ZX01-2), the Fundamental Research Funds for the Central Universities (2019FZA5005), Artificial Intelligence Research Foundation of Baidu Inc, and Zhejiang University — HIKVision Joint lab.

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Yi Bin, Yang Yang, Fumin Shen, Xing Xu, and Heng Tao Shen. 2016. Bidirectional long-short term memory for video description. In *ACM MM*.

Joao Carreira and Andrew Zisserman. 2017. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*.

David L Chen and William B Dolan. 2011. Collecting highly parallel data for paraphrase evaluation. In *ACL*.

Shizhe Chen, Jia Chen, Qin Jin, and Alexander Hauptmann. 2017. Video captioning with guidance of multimodal latent topics. In *ACM MM*.

Zhe Gan, Chuang Gan, Xiaodong He, Yunchen Pu, Kenneth Tran, Jianfeng Gao, Lawrence Carin, and Li Deng. 2017. Semantic compositional networks for visual captioning. In *CVPR*, volume 2.

Shawn Hershey, Sourish Chaudhuri, Daniel PW Ellis, Jort F Gemmeke, Aren Jansen, R Channing Moore, Manoj Plakal, Devin Platt, Rif A Saurous, Bryan Seybold, et al. 2017. Cnn architectures for large-scale audio classification. In *ICASSP*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Chiori Hori, Takaaki Hori, Teng-Yok Lee, Ziming Zhang, Bret Harsham, John R Hershey, Tim K Marks, and Kazuhiko Sumi. 2017. Attention-based multimodal fusion for video description. In *ICCV*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2015. Molding cnns for text: non-linear, non-consecutive convolutions. *arXiv preprint arXiv:1508.04112*.

Zhun Liu, Ying Shen, Varun Bharadhwaj Lakshminarasimhan, Paul Pu Liang, Amir Zadeh, and Louis-Philippe Morency. 2018. Efficient low-rank multimodal fusion with modality-specific factors. *arXiv preprint arXiv:1806.00064*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Yingwei Pan, Ting Yao, Houqiang Li, and Tao Mei. 2017. Video captioning with transferred semantic attributes. In *CVPR*.

Vasili Ramanishka, Abir Das, Dong Huk Park, Subhashini Venugopalan, Lisa Anne Hendricks, Marcus Rohrbach, and Kate Saenko. 2016. Multimodal video description. In *ACM MM*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Zhiqiang Shen, Jianguo Li, Zhou Su, Minjun Li, Yurong Chen, Yu-Gang Jiang, and Xiangyang Xue. 2017. Weakly supervised dense video captioning. In *CVPR*.

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. 2017. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*.

Subhashini Venugopalan, Huijuan Xu, Jeff Donahue, Marcus Rohrbach, Raymond Mooney, and Kate Saenko. 2014. Translating videos to natural language using deep recurrent neural networks. *arXiv preprint arXiv:1412.4729*.

Bairui Wang, Lin Ma, Wei Zhang, and Wei Liu. 2018a. Reconstruction network for video captioning. In *CVPR*.

Xin Wang, Wenhu Chen, Jiawei Wu, Yuan-Fang Wang, and William Yang Wang. 2018b. Video captioning via hierarchical reinforcement learning. In *CVPR*.

Xin Wang, Yuan-Fang Wang, and William Yang Wang. 2018c. Watch, listen, and describe: Globally and locally aligned cross-modal attentions for video captioning. *arXiv preprint arXiv:1804.05448*.

Jun Xu, Tao Mei, Ting Yao, and Yong Rui. 2016. Msr-vtt: A large video description dataset for bridging video and language. In *CVPR*.

Jun Xu, Ting Yao, Yongdong Zhang, and Tao Mei. 2017. Learning multimodal attention lstm networks for video captioning. In *ACM MM*.

Zhou Yu, Jun Yu, Jianping Fan, and Dacheng Tao. 2017. Multi-modal factorized bilinear pooling with co-attention learning for visual question answering. In *ICCV*.

Mihai Zanfir, Elisabeta Marinoiu, and Cristian Sminchisescu. 2016. Spatio-temporal attention models for grounded video captioning. In *ACCV*.

Xishan Zhang, Ke Gao, Yongdong Zhang, Dongming Zhang, Jintao Li, and Qi Tian. 2017. Task-driven dynamic fusion: Reducing ambiguity in video description. In *CVPR*.

# A Supplemental Material

## A.1 Proposition 1 for Eqn. 13 in the paper

**Proposition 1:** Suppose that we have $n$ matrices, $I_1, I_2, ..., I_n$, and $n$ vectors, $w_1, w_2, ..., w_n$. The space of $I_l$ is $\mathbb{R}^{d\times t_l}$ and the space of $w_l$ is $\mathbb{R}^{1\times t_l}$. Then

$$\Big(\bigotimes_{i=1}^{n} I_i\Big)\circ\Big(\bigotimes_{i=1}^{n} w_i\Big)=\bigotimes_{i=1}^{n} I_i\circ w_i \tag{25}$$

**Proof:** We use $C_l$ and $C_r$ to denote the left side and right side of the equation, respectively. We utilize the element-wise comparison in two tensors. Following Eqn. 7 and 8 in the paper, the $(r_1, ..., r_n)$-th entry of $C_l$ is expressed as

$$(C_l)_{r_1,...,r_n}=\mathbf{1}_d\Big[\bigwedge_{i=1}^{n}(I_i)_{r_i}\Big]\cdot\Big[\bigwedge_{i=1}^{n}(w_i)_{r_i}\Big] \tag{26}$$

where $(I_i)_{r_i}$ is a vector which denotes $r_i$-th column of the $I_i$, $(w_i)_{r_i}$ is the $r_i$-th value of the vector. Since $(w_i)_{r_i}$ is a single element, we can directly multiply it with the corresponding vector $(I_i)_{r_i}$.

$$\begin{aligned}(C_l)_{r_1,...,r_n}&=\mathbf{1}_d\Big[\bigwedge_{i=1}^{n}(I_i\circ w_i)_{r_i}\Big]\\&=(C_r)_{r_1,...,r_n}\end{aligned} \tag{27}$$

The proposition is proven and is used to convert Eqn. 12 to Eqn. 13 in the paper.

## A.2 Proposition 2 for Eqn. 15 in the paper

**Proposition 2:** Suppose that we have $n$ matrices, $I_1, I_2, ..., I_n$. The space of $I_l$ is $\mathbb{R}^{d\times t_l}$. Then

$$\sum\Big(\bigotimes_{i=1}^{n} I_i\Big)=\mathbf{1}_d\Big[\bigwedge_{i=1}^{n}(I_i)\mathbf{1}_{t_i}\Big] \tag{28}$$

here we use vectors $\mathbf{1}_d$ and $\mathbf{1}_{t_i}$ which consist of 1 to represent the summation operation for matrix $I_i$ in $d$ dimension and $t_i$ dimensions, respectively.

**Proof:** We use $v_l$ to denote the left side of the equation and $v_r$ to denote the right side of the equation. We can express $v_l$ as

$$v_l=\sum_{r_1=1}^{t_1}...\sum_{r_n=1}^{t_n}C_{r_1,r_2,...,r_n} \tag{29}$$

$$C=\bigotimes_{i=1}^{n} I_i \tag{30}$$

Following Eqn. 7 and 8 in the paper, we can express $C_{r_1,r_2,...,r_n}$ as

$$C_{r_1,r_2,...,r_n}=\mathbf{1}_d\Big[\bigwedge_{i=1}^{n}(I_i)_{r_i}\Big] \tag{31}$$

We apply Eqn. 32 to Eqn. 30,

$$\begin{aligned}v_l&=\sum_{r_1=1}^{t_1}...\sum_{r_n=1}^{t_n}\mathbf{1}_d\Big[\bigwedge_{i=1}^{n}(I_i)_{r_i}\Big]\\&=\mathbf{1}_d\Big[\sum_{r_1=1}^{t_1}...\sum_{r_n=1}^{t_n}\bigwedge_{i=1}^{n}(I_i)_{r_i}\Big]\\&=\mathbf{1}_d\Big[\bigwedge_{i=1}^{n}(I_i)\mathbf{1}_{t_i}\Big]=v_r\end{aligned} \tag{32}$$

The proposition is proven and is used to convert Eqn. 13 to Eqn. 15 in the paper.

## A.3 Learning Curves

We show the learning curves of the CIDEr on the validation set in Fig. 6 and observe that the L-HOCA-UBT performs better than HOCA-UBT and HOCA-U when the training converges.

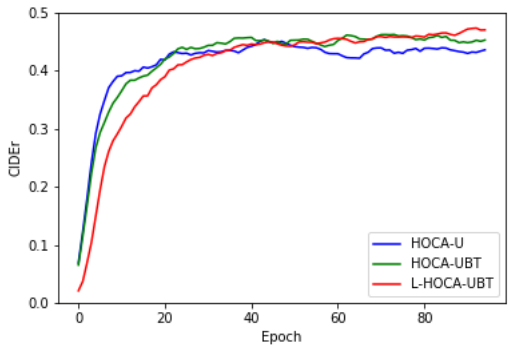


Figure 6: Learning curves of different methods on MSR-VTT, where the rank of L-HOCA-UBT is 1. Note that we use greedy search during training while beam search during testing, so the testing scores are higher.

### A.4 Visualization of Attention Weights

We also perform visualization of the attention weights in multiple attentive fusion (MAF) module. As shown in Fig. 7, HOCA-UBT obtains a more accurate ratio of each modality than HOCA-U, i.e. for the word "man", HOCA-U obtains a higher score of motion modality, which violates human subjective understanding.

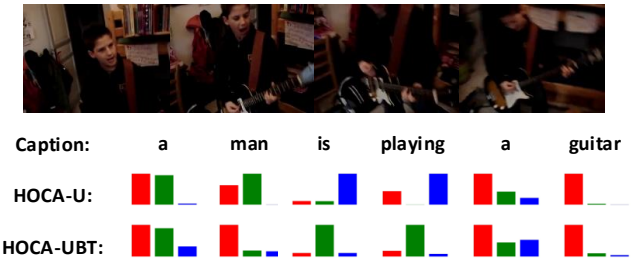


Figure 7: Visualization of the attention weights in multiple attentive fusion (MAF) module, the red bar denotes image modality, the green bar denotes motion modality, the blue bar denotes audio modality.

# Image Captioning with Very Scarce Supervised Data: Adversarial Semi-Supervised Learning Approach

**Dong-Jin Kim**[1] **Jinsoo Choi**[1] **Tae-Hyun Oh**[2] **In So Kweon**[1]
[1]KAIST, South Korea. [2]MIT CSAIL, Cambridge, MA.
[1]{djnjusa,jinsc37,iskweon77}@kaist.ac.kr [2]taehyun@csail.mit.edu

## Abstract

Constructing an organized dataset comprised of a large number of images and several captions for each image is a laborious task, which requires vast human effort. On the other hand, collecting a large number of images and sentences separately may be immensely easier. In this paper, we develop a novel data-efficient *semi-supervised* framework for training an image captioning model. We leverage massive *unpaired* image and caption data by learning to associate them. To this end, our proposed semi-supervised learning method assigns pseudo-labels to unpaired samples via Generative Adversarial Networks to learn the joint distribution of image and caption. To evaluate, we construct scarcely-paired COCO dataset, a modified version of MS COCO caption dataset. The empirical results show the effectiveness of our method compared to several strong baselines, especially when the amount of the paired samples are scarce.

## 1 Introduction

Image captioning is a task of automatically generating a natural language description of a given image. It is a highly useful task, in that 1) it extracts the essence from an image into a self-descriptive form of representation, and 2) the output format is a natural language, which exhibits free-form and manageable characteristics useful to applications such as language based image or region retrieval [Johnson *et al.*, 2016; Karpathy and Fei-Fei, 2015; Kim *et al.*, 2019], video summarization [Choi *et al.*, 2018], navigation [Wang *et al.*, 2018a], vehicle control [Kim *et al.*, 2018b]. Image captioning exhibits free-form characteristics, since it is not confined to a few number of predefined classes. This enables descriptive analysis of a given image.

Recent research on image captioning has made impressive progress [Anderson *et al.*, 2018; Vinyals *et al.*, 2015]. Despite this progress, the majority of works are trained via supervised learning that requires a large corpus of caption labeled images such as the MS COCO caption dataset [Lin *et al.*, 2014]. Specifically, the dataset is constructed with 1.2M images that were asked the annotators to provide five grammatically plausible sentences for each image. Constructing such human-labeled datasets are an immensely laborious task and time-consuming. This is a challenge of image captioning, because the task heavily requires large data, *i.e.*, data hungry task.

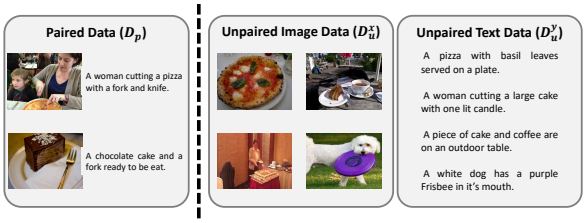


Figure 1: The proposed data setup utilizes both "paired" and "unpaired" image-caption data. We denote paired data as $\mathcal{D}_p$, and unpaired image and caption datasets as $\mathcal{D}_u^x$ and $\mathcal{D}_u^y$ respectively.

In this work, we present a novel way of leveraging *unpaired* image and caption data to train data hungry image captioning neural networks. We consider a scenario in which we have a small scale paired image-caption data of a specific domain. We are motivated by the fact that images can be easily obtained from the web and captions can be easily augmented and synthesized by replacing or adding different words for given sentences as done in [Zhang *et al.*, 2015]. Moreover, given a sufficient amount of descriptive captions, it is easy to crawl *corresponding but noisy* images through Google or Flickr image databases [Thomee *et al.*, 2016] to build an image corpus. In this way, we can easily construct a scalable *unpaired* dataset of images and captions, which requires no (or at least minimal) human effort.

Due to the unpaired nature of images (input) and captions (output supervision), the conven-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2012–2023, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

tional supervision loss can no longer be used directly. We propose to algorithmically assign supervision labels, termed as *pseudo-labels*, to make unpaired data paired. The pseudo-label is used as a *learned* supervision label. To develop the mechanism of pseudo-labeling, we use generative adversarial network (GAN) [Goodfellow *et al.*, 2014], for searching pseudo-labels for unpaired data. That is, in order to find the appropriate pseudo-labels for unpaired samples, we utilize an adversarial training method for training a discriminator model. Thereby, the discriminator learns to distinguish between real and fake image-caption pairs and to retrieve pseudo labels as well as to enrich the captioner training.

Our main contributions are summarized as follows. (1) We propose a novel framework for training an image captioner with the unpaired image-caption data and a small amount of paired data. (2) In order to facilitate training with unpaired data, we devise a new semi-supervised learning approach by the novel usage of the GAN discriminator. (3) We construct our *scarcely-paired COCO* dataset, which is the modified version of the MS COCO dataset without pairing information. On this dataset, we show the effectiveness of our method in various challenging setups, compared to strong competing methods.

## 2 Related Work

The goal of our work is to deal with unpaired image-caption data for image captioning. Therefore, we mainly focus on image captioning and unpaired data handling literature.

**Data Issues in Image Captioning.** Since the introduction of large-scale datasets such as MS COCO data [Lin *et al.*, 2014], image captioning has been extensively studied in vision and language society [Anderson *et al.*, 2018; Rennie *et al.*, 2017; Vinyals *et al.*, 2015; Xu *et al.*, 2015] by virtue of the advancement of deep convolutional neural networks [Krizhevsky *et al.*, 2012]. As neural network architectures become more advanced, they require much larger dataset scale for generalization [Shalev-Shwartz and Ben-David, 2014]. Despite the extensive study on better network architectures, the data issues for image captioning, such as noisy data, partially missing data, and unpaired data have been relatively barely studied.

Unpaired image-caption data has been just recently discussed. Gu *et al.* [2018] introduce a third modal information, Chinese captions, for language pivoting [Utiyama and Isahara, 2007; Wu and Wang, 2007]. Feng *et al.* [2019] propose an unpaired captioning framework which trains a model without image or sentence labels via learning a visual concept detector with external data, OpenImage dataset [Krasin *et al.*, 2017]. Chen *et al.* [2017] approach image captioning as a domain adaptation by utilizing large paired MS COCO data as the source domain and adapting on a separate unpaired image or caption dataset as the target domain. Liu *et al.* [2018] use self-retrieval for captioning to facilitate training a model with partially labeled data, where the self-retrieval module tries to retrieve corresponding images based on the generated captions. As a separate line of work, there are novel object captioning methods [Anne Hendricks *et al.*, 2016; Venugopalan *et al.*, 2017] that additionally exploit unpaired image and caption data corresponding to a novel word.

Most of aforementioned works [Gu *et al.*, 2018; Anne Hendricks *et al.*, 2016; Venugopalan *et al.*, 2017; Feng *et al.*, 2019] exploit large auxiliary supervised data which is often beyond image-caption data. To the best of our knowledge, we are the first to study how to handle unpaired image and caption data for image captioning without any auxiliary information but by leveraging scarce paired image-caption data only. Although [Chen *et al.*, 2017] does not use auxiliary information as well, it requires large amounts of paired source data, of which data regime is different from ours. [Liu *et al.*, 2018] is also this case, where they use the full paired MS COCO caption dataset with an additional large unlabeled image set. Our method lies on a very scarce paired source data regime, of which scale is only $1\%$ of the COCO dataset.

**Multi-modality in Unpaired Data Handling.** By virtue of the advancement on generative modeling techniques, *e.g.*, GAN [Goodfellow *et al.*, 2014], multi-modal translation recently emerged as a popular field. Among many possible modalities, image-to-image translation between two different (and unpaired) domains has been mostly explored. To tackle this issue, the cycle-consistency constraint between unpaired data is exploited in CycleGAN [Zhu *et al.*, 2017] and DiscoGAN [Kim *et al.*, 2017], and it is further improved in UNIT [Liu *et al.*, 2017].

In this work, we regard image captioning as a multi-modal translation. Our work has similar

motivation to the unpaired image-to-image translation [Zhu *et al.*, 2017], and unsupervised machine translation [Artetxe *et al.*, 2018; Lample *et al.*, 2018a,b] works or machine translation with monolingual data [Zhang *et al.*, 2018]. However, we show that the cycle-consistency does not work on our problem setup due to a significant modality gap. Instead, our results suggest that the traditional label propagation based semi-supervised framework [Zhou *et al.*, 2004] is more effective for our task.

**Semi-supervised Learning.** Our method is motivated by the generative model based semi-supervised learning [Chongxuan *et al.*, 2017; Gan *et al.*, 2017; Kingma *et al.*, 2014], which mostly deals with classification labels. In contrast, we leverage caption data and devise a novel model to train the captioning model.

## 3 Proposed Framework

In this section, we first brief the standard image caption learning, and describe how we can leverage the unpaired dataset. Then, we introduce an adversarial learning method for obtaining a GAN model that is used for assigning pseudo-labels and encouraging to match the distribution of latent features of images and captions. Lastly, we describe a connection to CycleGAN.

### 3.1 Model

Let us denote a dataset with $N_p$ image-caption pairs as $\mathcal{D}_p = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N_p}$. A typical image captioning task is defined as follows: given an image $\mathbf{x}_i$, we want a model to generate a caption $y_i$ that best describe the image. Traditionally, a captioning model is trained on a large paired dataset $(\mathbf{x}, y) \in \mathcal{D}_p$, *e.g.*, MS COCO dataset, by minimizing the negative log likelihood against the ground truth caption as follows:

$$\min_{\theta} \sum_{(\mathbf{x},y)\in\mathcal{D}_p} L_{\mathsf{CE}}(y, \mathbf{x}; \theta), \tag{1}$$

where $L_{\mathsf{CE}} = -\log q_\theta(y|\mathbf{x})$ denotes the cross entropy loss, $\theta$ the set of learnable parameters, and $q(\cdot)$ the probability output of the model. Motivated by encoder-decoder based neural machine translation literature [Cho *et al.*, 2014], traditional captioning frameworks are typically implemented as a encoder-decoder architecture [Vinyals *et al.*, 2015], *i.e.*, CNN-RNN. The CNN encoder $F(\mathbf{x}; \theta_{enc})$ produces a latent feature vector $\mathbf{z}^x$ from a given input image $\mathbf{x}$, and the RNN decoder $H(\mathbf{z}^x; \theta_{dec})$ is followed to generate a caption $y$ in a natural language form from $\mathbf{z}^x$, as depicted in Fig. 2. The term $q(\cdot)$ in Eq. (1) is typically implemented by the sum of the cross entropy of each word token. For simplicity, we omit $\theta$ from here if deducible, *e.g.*, $H(\mathbf{z}^x)$.

**Learning with Unpaired Data.** Our problem deals with unpaired data samples, where the image and caption sets $\mathcal{D}_u^x = \{\mathbf{x}_i\}_{i=0}^{N_x}$ and $\mathcal{D}_u^y = \{y_i\}_{i=0}^{N_y}$ are not paired. Given the unpaired datasets, due to the missing supervision, the loss in Eq. (1) cannot be directly computed. Motivated by the nearest neighbor aware semi-supervised framework [Shi *et al.*, 2018], we artificially generate *pseudo-labels* for respective unpaired datasets.

Specifically, we retrieve a best match caption $\tilde{y}_i$ in $\mathcal{D}_u^y$ given a query image $\mathbf{x}_i$, and assign it as a pseudo-label, and vice versa ($\tilde{\mathbf{x}}_i$ for $y_i$). We abuse the pseudo-label as a function for simplicity, *e.g.*, $\tilde{y}_i = \tilde{y}(\mathbf{x}_i)$. To retrieve a semantically meaningful match, we need a measure to assess matches. We use a discriminator network to determine real or fake pairs by GAN, which we will describe later. With the retrieved pseudo-labels, now we can compute Eq. (1) by modifying it as:

$$\min_{\theta} \lambda_x \sum_{\mathbf{x}\in\mathcal{D}_u^x} L_{\mathsf{CE}}(\tilde{y}(\mathbf{x}), \mathbf{x}; \theta) + \lambda_y \sum_{y\in\mathcal{D}_u^y} L_{\mathsf{CE}}(y, \tilde{\mathbf{x}}(y); \theta), \tag{2}$$

where $\lambda_{\{\cdot\}}$ denote the balance parameters.

**Discriminator Learning by Unpaired Feature Matching.** We train the criterion to find a semantically meaningful match, so that pseudo-labels for each modality are effectively retrieved. We learn a discriminator for that by using a small amount of paired supervised dataset.

We adopt a caption encoder, $G(y; \theta_{cap})$, which embeds the caption $y$ into a feature $\mathbf{z}^y$. This is implemented with a single layer LSTM, and we take the output of the last time step as the caption representation $\mathbf{z}^y$. Likewise, given an image $x$, we obtain $\mathbf{z}^x$ by the image encoder $F(\mathbf{x}; \theta_{enc})$. Now, we have a comparable feature space of $\mathbf{z}^x$ and $\mathbf{z}^y$. We utilize the discriminator to distinguish whether the pair $(\mathbf{z}^x, \mathbf{z}^y)$ from true paired data $(\mathbf{x}, y) \in \mathcal{D}_p$, *i.e.*, the pair belongs to the real distribution $p(\mathbf{x}, y)$ or not. We could use random pairs of $(\mathbf{x}, y)$ independently sampled from respective unpaired datasets, but we found that it is detrimental due to uninformative pairs. Instead, we conditionally

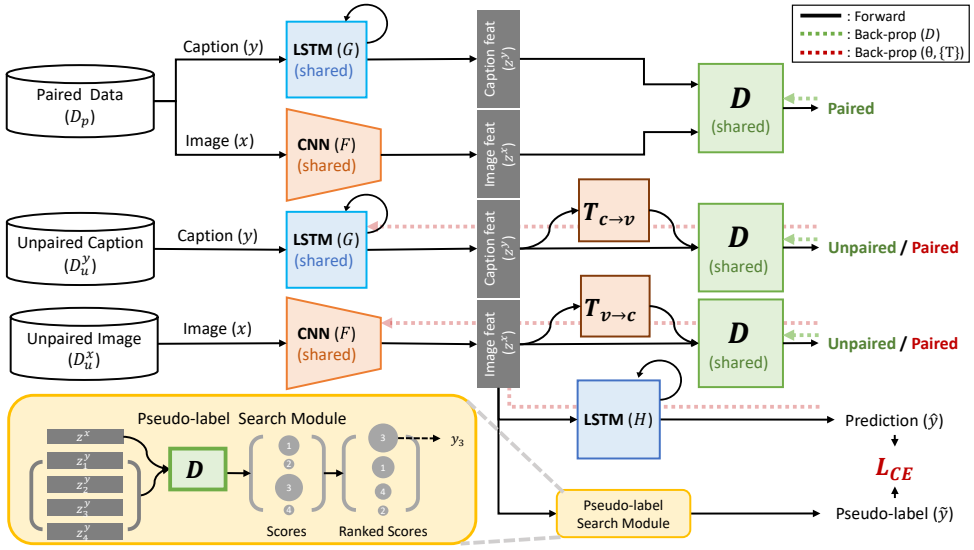


Figure 2: Description of the proposed method. Dotted arrows denote the path of the gradients via back-propagation. Given any image and caption pair, CNN and RNN (LSTM) encoders encode input image and caption into the respective feature spaces. A discriminator ($D$) is trained to discriminate whether the given feature pairs are real or fake, while the encoders are trained to fool the discriminator. The learned discriminator is also used to assign the most likely pseudo-labels to unpaired samples through the pseudo-label search module.

synthesize $\mathbf{z}^x$ or $\mathbf{z}^y$, to form a synthesized pair that appears to be as realistic as possible. We use the feature transformer networks $\tilde{\mathbf{z}}^y = T_{v\to c}(\mathbf{z}^x)$ and $\tilde{\mathbf{z}}^x = T_{c\to v}(\mathbf{z}^y)$, where $v\to c$ denotes the mapping from visual data to caption data and vice versa, and $\tilde{\mathbf{z}}$ denotes the conditionally synthesized feature. $\{T\}$ are implemented with multi-layer-perceptron with four FC layers with ReLU nonlinearity.

The discriminator $D(\cdot,\cdot)$ learns to distinguish features, real or not. At the same time, the other associated networks $F, G, T_{\{\cdot\}}$ are learned to fool the discriminator by matching the distribution of paired and unpaired data. Motivated by [Chongxuan *et al.*, 2017], the formulation of this adversarial training is as follows:

$$\min_{\theta,\{T\}} \max_{D} \mathop{\mathbb{E}}_{\substack{(\mathbf{z}^x,\mathbf{z}^y)\\ \sim(F,G)\circ\mathcal{D}_p}} [\log(D(\mathbf{z}^x,\mathbf{z}^y)) + L_{reg}(\mathbf{z}^x,\mathbf{z}^y,\{T\})] + \tfrac{1}{2}\left( \mathop{\mathbb{E}}_{\mathbf{x}\sim p(\mathbf{x})} [\log(1 - D(F(\mathbf{x}), T_{v\to c}(F(\mathbf{x}))))] + \mathop{\mathbb{E}}_{y\sim p(y)} [\log(1 - D(T_{c\to v}(G(y)), G(y)))] \right), \quad (3)$$

where $L_{reg}(\mathbf{z}^x,\mathbf{z}^y,\{T\}) = \lambda_{reg}(\| \underset{v\to c}{T}(\mathbf{z}^x) - \mathbf{z}^y\|_F^2 + \|\mathbf{z}^x - \underset{c\to v}{T}(\mathbf{z}^y)\|_F^2)$, and we use the distribution notation $p(\cdot)$ to flexibly refer to any type of the dataset regardless of $\mathcal{D}_p$ and $\mathcal{D}_u$. Note that the first log term is not used for updating any learnable parameters related to $\theta, \{T\}$, but only used for updating $D$. Through alternating training of the discriminator ($D$) and generator ($\theta, \{T\}$) similar to [Kim *et al.*, 2018a], the latent feature distribution of paired and unpaired data should be close to each other, *i.e.*, $p(\mathbf{z}^x,\mathbf{z}^y) = p(\mathbf{z}^x, T_{v\to c}(\mathbf{z}^x)) = p(T_{c\to v}(\mathbf{z}^y),\mathbf{z}^y)$ (refer to the proof in [Chongxuan *et al.*, 2017]). In addition, as the generator is trained, the decision boundary of the discriminator tightens. If the unpaired datasets are sufficiently large such that semantically meaningful matches exist between the different modality datasets, and if the discriminator $D$ is plausibly learned, we can use $D$ to retrieve a proper pseudo-label. Our architecture is illustrated in Fig. 2.

**Pseudo-label Assignment.** Given an image $\mathbf{x} \in \mathcal{D}_u^x$, we retrieve a caption in the unpaired dataset, *i.e.*, $\tilde{y} \in \mathcal{D}_u^y$, that has the highest score obtained by the discriminator, *i.e.*, the most likely caption to be paired with the given image as

$$\tilde{y}_i = \tilde{y}(\mathbf{x}_i) = \underset{y\in\mathcal{D}_u^y}{\operatorname{argmax}} \; D\left(F(\mathbf{x}_i), G(y)\right), \quad (4)$$

vice versa for unpaired captions:

$$\tilde{\mathbf{x}}_i = \tilde{\mathbf{x}}(y_i) = \underset{\mathbf{x}\in\mathcal{D}_u^x}{\operatorname{argmax}} \; D\left(F(\mathbf{x}), G(y_i)\right). \quad (5)$$

By this retrieval process over all the unpaired dataset, we have image-caption pairs $\{(x_i, y_i)\}$ from the paired data and the pairs with pseudo-labels $\{(x_j, \tilde{y}_j)\}$ and $\{(\tilde{x}_k, y_k)\}$ from the unpaired

data. However, these pseudo-labels are not noise-free, thus treating them equally with the paired ones is detrimental. Motivated by learning with noisy labels [Lee *et al.*, 2018; Wang *et al.*, 2018b], we define a confidence score for each of the assigned pseudo-labels. We use the output score from the discriminator, as the confidence score, *i.e.*, $\alpha_i^x = \hat{D}(\mathbf{x}_i, \tilde{y}_i)$ and $\alpha_i^y = \hat{D}(\tilde{\mathbf{x}}_i, y_i)$, where we denote $\hat{D}(\mathbf{x}, y) = D(F(\mathbf{x}), G(y))$, and $\alpha \in [0, 1]$ due to the sigmoid function of the final layer of the discriminator. Therefore, we utilize the confidence scores to assign weights to the unpaired samples. We compute the weighted loss as follows:

$$\min_{\theta} \sum_{(\mathbf{x},y)\in\mathcal{D}_p} L_{\mathsf{CE}}(y, \mathbf{x}; \theta) + \lambda_x \sum_{\mathbf{x}\in\mathcal{D}_u^x} \alpha_i^x L_{\mathsf{CE}}(\tilde{y}(\mathbf{x}), \mathbf{x}; \theta) + \lambda_y \sum_{y\in\mathcal{D}_u^y} \alpha_i^y L_{\mathsf{CE}}(y, \tilde{\mathbf{x}}(y); \theta). \quad (6)$$

We jointly train the model on both paired and unpaired data. To ease the training further, we add an additional triplet loss function:

$$\sum_{\substack{(x_p,y_p)\in\mathcal{D}_p,\\ x_u\in\mathcal{D}_u^x, y_u\in\mathcal{D}_u^y}} \log \frac{p_\theta(y_p|x_p)}{p_\theta(y_p|x_u)} + \log \frac{p_\theta(y_p|x_p)}{p_\theta(y_u|x_p)}, \quad (7)$$

by regarding random unpaired samples as negative. This slightly improves the performance.

### 3.2 Connection with CycleGAN

As a representative fully unpaired method, CycleGAN [Zhu *et al.*, 2017] would be the strong baseline at this point, which has been popularly used for unpaired distribution matching. Since it is designed for image-to-image translation, we describe the modifications to it to fit our task, so as to understand relative performance of our method over the CycleGAN baseline. When applying the cycle-consistency on matching between images and captions, since the input modalities are totally different, we modify it to a translation problem over feature spaces as follows:

$$\begin{aligned}&\min_{\theta,\{T\}} \max_{D_{\{x,y\}}} L_{cycle}(\{T\}) \\ &+ \mathop{\mathbb{E}}_{\mathbf{x}\sim p(\mathbf{x})}[\log(D_x(F(\mathbf{x}))) + \log(1 - D_x(T_{v\to c}(F(\mathbf{x}))))] \\ &+ \mathop{\mathbb{E}}_{y\sim p(y)}[\log(D_y(G(y))) + \log(1 - D_y(T_{c\to v}(G(y))))],\end{aligned} \quad (8)$$

where $\{T\}$ and $D_{\{x,y\}}$ denotes the feature translators and the discriminators for image domain and caption domain, and

$$\begin{aligned}L_{cycle}(\{T\}) = \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}[\|T_{c\to v}(T_{v\to c}(F(\mathbf{x}))) - F(\mathbf{x})\|_2] \\ + \mathbb{E}_{y\sim p(y)}[\|T_{v\to c}(T_{c\to v}(G(y))) - G(y)\|_2].\end{aligned} \quad (9)$$

The discriminator $D$ is learned to distinguish whether the latent features is from the image or the caption distribution. This is different with our method, in that we distinguish the correct *semantic match of pairs*. We experimented with CycleGAN purely on unpaired datasets, but we were not able to train it; hence, we add the supervised loss (Eq. (1)) with the paired data, which is a fair setting with ours.

## 4 Experiments

In this section, we describe the experimental setups, competing methods and provide performance evaluations of unpaired captioning with both quantitative and qualitative results.

### 4.1 Implementation Details

We implement our neural networks by PyTorch library [Paszke *et al.*, 2017]. We use ResNet101 [He *et al.*, 2016] model as a backbone CNN encoder for image input, and initialize with weights pretrained on ImageNet [Russakovsky *et al.*, 2015]. Each region is represented as 2,048-dimensional output from the last residual block of this network. For NIC model [Vinyals *et al.*, 2015] and Att2in model [Rennie *et al.*, 2017] in Table 3, we use visual features with size $2048 \times 7 \times 7$ . For Up-Down [Anderson *et al.*, 2018] and the model in Table 3, we use Faster R-CNN [Ren *et al.*, 2015] pretrained on Visual Genome [Krishna *et al.*, 2017] object dataset, thereby we extract visual features for 10 to 100 regions.

We set the channel size of the hidden layers of LSTMs to be 1024, 512 for the attention layer, and 1024 for the word embeddings. For inference stage, we empirically choose the beam size to be 3 when generating a description, which shows the best performance.

We use a minibatch size of 100, the Adam [Ba and Kingma, 2015] optimizer for training (learning rate $lr$=$5e^{-4}$, $b_1$=0.9, $b_2$=0.999). For hyperparameters, we empirically choose $\lambda_x$ and $\lambda_y$ to be equal to 0.1. The total loss function for training our model is as follows:

$$\mathcal{L} = \mathcal{L}_{cap} + 0.1\mathcal{L}_{GAN} + 0.1\mathcal{L}_{triplet}, \quad (10)$$

where $\mathcal{L}_{cap}$ denotes the captioning loss defined in Eq. (6), $\mathcal{L}_{GAN}$ the loss for adversarial training defined in Eq. (3), and $\mathcal{L}_{triplet}$ the triplet loss defined in Eq. (7).

| | (A) | (B) | (C) | BLEU1 | BLEU4 | METEOR | CIDEr |
|---|---|---|---|---|---|---|---|
| Fully paired (100%) | | | | 72.5 | 29.4 | 25.0 | 95.7 |
| Paired only (1%) | | | | 58.1 | 13.4 | 15.9 | 36.0 |
| CycleGAN (Zhu *et al.* ) | | | | 58.7 | 14.1 | 16.2 | 37.7 |
| **Ours ver1** | ✓ | | | 60.1 | 15.7 | 16.5 | 45.3 |
| **Ours ver2** | ✓ | ✓ | | 61.3 | 17.1 | 19.1 | 51.7 |
| **Ours (final)** | ✓ | ✓ | ✓ | **63.0** | **18.7** | **20.7** | **55.2** |
| [Gu *et al.*, 2018] | | | | 46.2 | 5.4 | 13.2 | 17.7 |
| [Feng *et al.*, 2019] | | | | 58.9 | 18.6 | 17.9 | 54.9 |

Table 1: Captioning performance comparison on MS COCO testset. The "Paired only" baseline is trained only with 1% of paired data from our *scarcely-paired COCO* dataset, while CycleGAN and Ours {ver1, ver2, final}, are trained with our *scarcely-paired COCO* dataset (1% of paired data and unpaired image and caption datasets). We indicate the ablation study by: (A) the usage of the proposed GAN that distinguishes real or fake image-caption pairs, (B) pseudo-labeling, and (C) noise handling by sample re-weighting. We also compare with [Gu *et al.*, 2018] and [Feng *et al.*, 2019], which are trained with unpaired datasets.

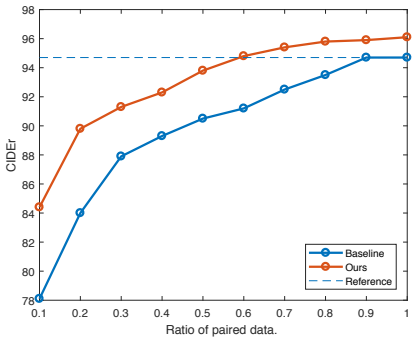


Figure 3: Performance w.r.t. amount of paired data for training. Baseline denotes our *Paired Only* baseline, Ours is our final model, and Reference is *Paired Only* trained with full paired data.

## 4.2 Experimental Setups

We utilize MS COCO caption [Lin *et al.*, 2014] as our target dataset, which contains 123k images with 5 caption labels per image. To validate our model, we follow *Karpathy* splits [Karpathy and Fei-Fei, 2015], which have been broadly used in image captioning literature. The Karpathy splits contain 113k training, 5k validation, and 5k test images in total. In our experiments, in order to simulate the scenario that both paired and unpaired data exist, we use two different data setups: 1) the proposed scarcely-paired COCO, and 2) partially labeled COCO [Liu *et al.*, 2018] setup.

For the *scarcely-paired* COCO setup, we remove the pairing information of the MS COCO dataset, while leaving a small fraction of pairs unaltered. We randomly select only one percent of the total data, *i.e.*, 1,133 training images for the paired data, and remove the pairing information of the rest to obtain unpaired data. We call the small fraction of samples given as pairs as *paired* data ($\mathcal{D}_p$) and call the samples without pairing information as *unpaired* data ($\mathcal{D}_u$). We study the effects of other ratios of paired data used for training (in Figs. 4 and 3). This dataset allows evaluating the proposed framework to measure whether such small paired data can be leveraged to learn plausible pseudo-label assignment, and what performance can be achieved compared to the fully supervised case.

For *partially labeled* COCO setup, we follow [Liu *et al.*, 2018] and use the whole MS COCO data (paired) and add the *"Unlabeled-COCO"* split of the officially MS-COCO [Lin *et al.*, 2014] for unpaired images, which involves 123k images without any caption label. To compute the cross entropy loss, we use the pseudo-label assignment for the Unlabeled-COCO images.

For evaluation, we use the following metrics conventionally used in image captioning: BLEU [Papineni *et al.*, 2002], ROUGE-L [Lin, 2004], SPICE [Anderson *et al.*, 2016], METEOR [Denkowski and Lavie, 2014], and CIDEr [Vedantam *et al.*, 2015]. All the evaluation is done on MS COCO testset.

In order to avoid high time complexity of the pseudo-labeling process, we do not search the pseudo-labels from the whole unpaired data. Pseudo-label retrieval is done on a subset of one-hundredth of the unpaired data (yielding 1000 samples). Thus, the complexity of each minibatch becomes $O(B \times N \times 0.01)$, where $B$ denotes the minibatch size, $N$ denotes the size of the unpaired dataset. Since we also apply label-noise handling, a plausible pseudo-label assignment is sufficient in helping the learning process. We can improve performance by using a larger subset or by using a fast approximate nearest neighbor search.

## 4.3 Evaluation on Scarcely-paired COCO

We follow the same setting with [Vinyals *et al.*, 2015], if not mentioned. Table 1 shows the results on the scarcely-paired COCO dataset. We compare with several baselines: *Paired Only*; we train our model only on the small fraction (1%) of the paired data, *CycleGAN*; we train our model with the cycle-consistency loss [Zhu *et al.*, 2017] (as in Eq. (8)). Additionally, we train variants of our model denoted as *Ours* (ver1, ver2, and final). The model denoted as *Our ver1*, is trained




Figure 4: Sampled qualitative results of our model. We compare with the baseline models trained only on $N\%$ of paired samples out of the full MS COCO. Despite the use of only 1% paired data, our model generates plausible captions similar to that of the baseline models trained with more data (10% and above).

via our GAN model (Eq. (3)) that distinguishes real or fake image-caption pairs. For *Our ver2*, we add training with pseudo-labels for unpaired data with Eq. (6) while setting the confidence scores $\alpha^x{=}\alpha^y{=}1$ for all training samples. For our final model denoted as *Our (final)*, we apply the noise handling technique by the confidence scores $\alpha^x$ and $\alpha^y$ computed by Eq. (6) and re-weighting the loss for each sample. We present the performance of the fully supervised (*Fully paired*) model using 100% of the COCO training data for reference.

As shown in Table 1, in a scarce data regime, utilizing the unpaired data improves the captioning performance in terms of all metrics by noticeable margins. Also, our models show favorable performance compared to the CycleGAN model in all metrics. Our final model with pseudo-labels and noise handling achieves the best performance in all metrics across all the competitor, hereafter refered to this model as our model.

We also compare the recent unpaired image captioning methods [Gu *et al.*, 2018; Feng *et al.*, 2019] in Table 1. Both of the methods are evaluated on MS COCO testset. In the case of Gu *et al.*, AIC-ICC image-to-Chinese dataset [Wu *et al.*, 2017] is used as unpaired images $\mathcal{D}_u^x$ and captions from MS COCO are used as unpaired captions $\mathcal{D}_u^y$. Note that this is not a fair comparison to our method but they have more advantages, in that Gu *et al.* use large amounts of additional labeled data (10M Chinese-English parallel sentences of AIC-MT dataset [Wu *et al.*, 2017]) and Feng *et al.* use 36M samples of additional OpenImages dataset, whereas our model only uses a small amount of paired samples (1k) and 122k unpaired data. Despite far lower reliance on paired data, our model shows favorable performance against recent unpaired captioning works.

We study our final model against our *Paired Only* baseline according to varying amounts of paired training data in Fig. 3, so that we can see how much information can be gained from the unpaired data. From 100% to 10%, as the amount of paired samples decreases, the fluency and the accuracy of the descriptions get worse. In particular, we observed that most of the captions generated from the *Paired Only* baseline trained with 10% of paired data (11,329 pairs) show erroneous grammatical structure. In contrast, by leveraging unpaired data, our method can generate more fluent and accurate captions, compared to *Paired Only* trained on the same amount of paired data. It is worthwhile to note that our model trained with 60% of paired data (67,972 pairs) achieves similar performance to the *Paired Only* baseline trained with fully paired data (113,287 pairs), which sig-

| | BLEU1 | BLEU2 | BLEU3 | BLEU4 | ROUGE-L | SPICE | METEOR | CIDEr |
|---|---|---|---|---|---|---|---|---|
| Self-Retrieval [Liu *et al.*, 2018] (w/o unlabeled) | 79.8 | 62.3 | 47.1 | 34.9 | 56.6 | 20.5 | 27.5 | 114.6 |
| Self-Retrieval [Liu *et al.*, 2018] (with unlabeled) | 80.1 | 63.1 | 48.0 | 35.8 | 57.0 | 21.0 | 27.4 | 117.1 |
| Baseline (w/o unlabeled) | 77.7 | 61.6 | 46.9 | 36.2 | 56.8 | 20.0 | 26.7 | 114.2 |
| **Ours** (w/o unlabeled) | 80.8 | 65.3 | 49.9 | 37.6 | 58.7 | 22.7 | 28.4 | 122.6 |
| **Ours** (with unlabeled) | **81.2** | **66.0** | **50.9** | **39.1** | **59.4** | **23.8** | **29.4** | **125.5** |

Table 2: Comparison with the semi-supervised image captioning method, "Self-Retrieval" [Liu *et al.*, 2018]. Our method shows improved performance even without Unlabeled-COCO data (denoted as *w/o unlabeled*) as well as with Unlabeled-COCO (*with unlabeled*), although our model is not originally proposed for such scenario.




Figure 5: Examples of the pseudo-labels (captions) assigned to the unpaired images. Our model is able to sufficiently assign image-caption pairs through the proposed adversarial training.




Figure 6: Example captions containing words that do not exist in the paired dataset $\mathcal{D}_p$. The novel words that are not in $\mathcal{D}_p$ but in $\mathcal{D}_u^y$ are highlighted in bold.

nifies that our method is able to save *near half* of the human labeling effort of constructing a dataset.

We also show qualitative samples of our results in Fig. 4. *Paired Only* baseline trained with 1% paired data produces erroneous captions, and the baseline with 10% paired data starts to produce plausible captions. It is based on *ten times* more paired samples, compared to our model that uses only 1% of them. We highlight that, in the two examples on the top row of Fig. 4, our model generates more accurate captions than the *Paired Only* baseline trained on the 100% paired data ("baseball" to "Frisbee" on the top-left, and "man" to "woman" on the top-right). This suggests that unpaired data with our method effectively boosts the performance especially when paired data is scarce.

In order to demonstrate the meaningfulness of our pseudo-label assignment, we show the pseudo-labels assigned to the unpaired samples. We show the pseudo-labels (captions) assigned to unlabeled images from Unlabeled-COCO images in Fig. 5. As shown in the figure, despite not knowing the real pairs of these images, the pseudo labels are sufficiently assigned by the model. Note that even though there is no ground truth caption for unlabeled images in the searching pool, the model is able to find the most likely (semantically correlated) image-caption pair for the given images.

Fig. 6 highlights interesting result samples of our caption generation, where the results contain words that do not exist in the paired data of the scarcely-paired COCO dataset. The semantic meaning of the words such as "growling," "growing," and "herded," which involve in the unpaired caption data but not in the paired data, may have been learned properly via pseudo-label assignment during training. These examples suggest that our method is capable to infer the semantic meaning of unpaired words to some extent, which would have been unable to be learned with only little paired data. This would be an evidence that our framework is capable to align abstract semantic spaces between two modalities, *i.e.*, visual and caption.

### 4.4 Evaluation on Partially Labeled COCO

For a more realistic setup, we compare with the recent semi-supervised image captioning method, called *Self-Retrieval* [Liu *et al.*, 2018], on their problem regime, *i.e.*, "partially labeled" COCO setup, where the full amount of the paired MS COCO (113k) plus 123k uncaptioned images of the Unlabeled-COCO set are used (no additional unpaired caption is used). While their regime is not our direct scope, but we show the effectiveness of our method in the regime. Then, we extend our framework by replacing our backbone architecture with recent advanced image caption architectures. In this setup, a separate *unpaired* caption data $\mathcal{D}_u^y$ does not exist, thus we use captions from paired COCO data $\mathcal{D}_p$ as pseudo-labels.

Table 2 shows the comparison with Self-Retrieval. For a fair comparison with them, we replace the cross entropy loss from our loss with the policy gradient method [Rennie *et al.*, 2017] to directly optimize our model with CIDEr score as in [Liu *et al.*, 2018]. As our baseline

| | BLEU1 | BLEU2 | BLEU3 | BLEU4 | ROUGE-L | SPICE | METEOR | CIDEr |
|---|---|---|---|---|---|---|---|---|
| NIC [Vinyals *et al.*, 2015] | 72.5 | 55.1 | 40.4 | 29.4 | 52.7 | 18.2 | 25.0 | 95.7 |
| NIC **+Ours** | **74.2** | **56.9** | **42.0** | **31.0** | **54.1** | **19.4** | **28.8** | **97.8** |
| Att2in [Rennie *et al.*, 2017] | 74.5 | 58.1 | 44.0 | 33.1 | 54.6 | 19.4 | 26.2 | 104.3 |
| Att2in **+Ours** | **75.5** | **58.8** | **45.0** | **34.3** | **55.4** | **20.0** | **26.8** | **105.3** |
| Up-Down [Anderson *et al.*, 2018] | 76.7 | 60.7 | 47.1 | 36.4 | 56.6 | 20.6 | 27.6 | 113.6 |
| Up-Down **+Ours** | **77.3** | **64.1** | **47.6** | **37.0** | **57.1** | **21.0** | **27.8** | **114.6** |

Table 3: Evaluation of our method with different backbone architectures. All models are reproduced and trained with the cross entropy loss. By adding Unlabeled-COCO images, our training method was applied in a semi-supervised way, which shows consistent improvement in all the metrics.

model (denoted as *Baseline*), we train a model only with policy gradient method without the proposed GAN model. When only using the 100% paired MS COCO dataset (denoted as *w/o unlabeled*), our model already shows improved performance over Self-Retrieval. Moreover, when adding Unlabeled-COCO images (denoted as *with unlabeled*), our model performs favorably against Self-Retrieval in all the metrics. The results suggest that our method is also advantageous in the semi-supervised setup.

To further validate our method in the semi-supervised setup, we compare different backbone architectures [Anderson *et al.*, 2018; Rennie *et al.*, 2017; Vinyals *et al.*, 2015] in our framework, where their methods are developed for fully-supervised methods. We use the same data setup with the above, but we replace CNN ($F$) and LSTM ($H$) from our framework with the image encoder and the caption decoder from their image captioning models. Then, these models can be trained in our framework as it is by alternating between the discriminator update and pseudo-labeling. Table 3 shows the comparison. Training with the additional Unlabeled-COCO data via *our* training scheme consistently improves all baselines in all metrics.

### 4.5 Captioning with Web-Crawled Data

To simulate a scenario involving crawled data from the web, we use the setup suggested by Feng *et al.* [2019]. They collect a sentence corpus by crawling the image descriptions from Shutterstock[1] as unpaired caption data $\mathcal{D}_u^y$, whereby 2.2M sentences are collected. For unpaired image data $\mathcal{D}_u^x$, they use only the images from the MS COCO data, while the captions are not used for training. For training our method, we leverage from 0.5% to 1% of paired MS COCO data as our paired data $\mathcal{D}_p$. The results are shown in Table 4 with the performance obtained by Feng *et al*. Note again that Feng *et al.* exploits external large-scale data, *i.e.*, 36M images in the OpenImages dataset. With 0.5% of paired only data (566 pairs), our baseline shows lower scores in terms of BLEU4 and METEOR than Feng *et al.*, while our proposed model shows comparable or favorable performance in BLEU4, ROUGE-L and METEOR. Our method starts to have higher scores in all metrics from 0.8% of paired data (906 pairs), even without additional 36M images.

[1]https://www.shutterstock.com

| | BLEU4 | ROUGE-L | SPICE | METEOR | CIDEr |
|---|---|---|---|---|---|
| Paired only (0.5% paired) | 4.4 | 33.7 | 3.6 | 10.8 | 8.6 |
| **Ours** (0.5% paired) | 5.4 | 34.6 | 4.2 | 12.0 | 10.5 |
| Paired only (0.7% paired) | 3.5 | 36.1 | 3.7 | 11.4 | 8.9 |
| **Ours** (0.7% paired) | 8.5 | 39.0 | 5.2 | 13.6 | 20.2 |
| Paired only (0.8% paired) | 8.8 | 39.1 | 5.9 | 13.2 | 21.9 |
| **Ours** (0.8% paired) | 12.2 | 41.6 | 7.6 | 15.1 | 29.0 |
| Paired only (1% paired) | 13.4 | 41.9 | 8.3 | 15.9 | 36.0 |
| **Ours** (1% paired) | 15.2 | 43.3 | 9.4 | 16.9 | 39.7 |
| [Feng *et al.*, 2019] | 4.8 | 27.2 | 7.0 | 11.4 | 23.3 |

Table 4: Performance comparison with web-crawled data (Shutterstock). On top of unpaired image and caption data, our method is trained with 0.5 – 1% of paired data, while Feng *et al.* use 36M additional images of the OpenImage dataset.

## 5 Conclusion

We introduce a method to train an image captioning model with a large scale unpaired image and caption data, given a small amount of paired data. Our framework achieves favorable performance compared to various methods and setups. Unpaired captions and images are the data that can be easily collected from the web. It can facilitate application specific captioning models, where labeled data is scarce.

**Acknowledgements.** This work was supported by Institute for Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No.2017-0-01780, The technology development for event recognition/relational reasoning and learning knowledge based system for video understanding)

## References


Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. Spice: Semantic propositional image caption evaluation. In *European Conference on Computer Vision (ECCV)*. Springer, 2016.

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and vqa. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Lisa Anne Hendricks, Subhashini Venugopalan, Marcus Rohrbach, Raymond Mooney, Kate Saenko, and Trevor Darrell. Deep compositional captioning: Describing novel object categories without paired training data. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. Unsupervised neural machine translation. In *International Conference on Learning Representations (ICLR)*, 2018.

Jimmy Ba and Diederik Kingma. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Tseng-Hung Chen, Yuan-Hong Liao, Ching-Yao Chuang, Wan-Ting Hsu, Jianlong Fu, and Min Sun. Show, adapt and tell: Adversarial training of cross-domain image captioner. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

Jinsoo Choi, Tae-Hyun Oh, and In So Kweon. Contextually customized video summaries via natural language. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2018.

LI Chongxuan, Taufik Xu, Jun Zhu, and Bo Zhang. Triple generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

Michael Denkowski and Alon Lavie. Meteor universal: Language specific translation evaluation for any target language. In *The workshop on statistical machine translation*, 2014.

Yang Feng, Lin Ma, Wei Liu, and Jiebo Luo. Unsupervised image captioning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Zhe Gan, Liqun Chen, Weiyao Wang, Yuchen Pu, Yizhe Zhang, Hao Liu, Chunyuan Li, and Lawrence Carin. Triangle generative adversarial networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

Jiuxiang Gu, Shafiq Joty, Jianfei Cai, and Gang Wang. Unpaired image captioning by language pivoting. In *European Conference on Computer Vision (ECCV)*, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Justin Johnson, Andrej Karpathy, and Li Fei-Fei. Densecap: Fully convolutional localization networks for dense captioning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks. In *International Conference on Machine Learning (ICML)*, 2017.

Dong-Jin Kim, Jinsoo Choi, Tae-Hyun Oh, Youngjin Yoon, and In So Kweon. Disjoint multi-task learning between heterogeneous human-centric tasks. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2018.

Jinkyu Kim, Anna Rohrbach, Trevor Darrell, John Canny, and Zeynep Akata. Textual explanations for self-driving vehicles. In *European Conference on Computer Vision (ECCV)*, 2018.

Dong-Jin Kim, Jinsoo Choi, Tae-Hyun Oh, and In So Kweon. Dense relational captioning: Triple-stream networks for relationship-based captioning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in neural information processing systems*, pages 3581–3589, 2014.

Ivan Krasin, Tom Duerig, Neil Alldrin, Vittorio Ferrari, Sami Abu-El-Haija, Alina Kuznetsova, Hassan Rom, Jasper Uijlings, Stefan Popov, Andreas Veit, et al. Openimages: A public dataset for large-scale multi-label and multi-class image classification. *Dataset available from https://github.com/openimages*, 2:3, 2017.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision (IJCV)*, 123(1):32–73, 2017.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. Unsupervised machine translation using monolingual corpora only. In *International Conference on Learning Representations (ICLR)*, 2018.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, et al. Phrase-based & neural unsupervised machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Kuang-Huei Lee, Xiaodong He, Lei Zhang, and Linjun Yang. Cleannet: Transfer learning for scalable image classifier training with label noise. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*. Springer, 2014.

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*, 2004.

Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

Xihui Liu, Hongsheng Li, Jing Shao, Dapeng Chen, and Xiaogang Wang. Show, tell and discriminate: Image captioning by self-retrieval with partially labeled data. In *European Conference on Computer Vision (ECCV)*, 2018.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Association for Computational Linguistics (ACL)*. Association for Computational Linguistics, 2002.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. Self-critical sequence training for image captioning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

Weiwei Shi, Yihong Gong, Chris Ding, Zhiheng MaXiaoyu Tao, and Nanning Zheng. Transductive semi-supervised deep learning using min-max features. In *European Conference on Computer Vision (ECCV)*, 2018.

Bart Thomee, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. Yfcc100m: the new data in multimedia research. *Communications of the ACM*, 59(2):64–73, 2016.

Masao Utiyama and Hitoshi Isahara. A comparison of pivot methods for phrase-based statistical machine translation. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2007.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

Subhashini Venugopalan, Lisa Anne Hendricks, Marcus Rohrbach, Raymond Mooney, Trevor Darrell, and Kate Saenko. Captioning images with diverse objects. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

Xin Wang, Wenhan Xiong, Hongmin Wang, and William Yang Wang. Look before you leap: Bridging model-free and model-based reinforcement learning for planned-ahead vision-and-language navigation. In *European Conference on Computer Vision (ECCV)*, 2018.

Yisen Wang, Weiyang Liu, Xingjun Ma, James Bailey, Hongyuan Zha, Le Song, and Shu-Tao Xia. Iterative learning with open-set noisy labels. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Hua Wu and Haifeng Wang. Pivot language approach for phrase-based statistical machine translation. *Machine Translation*, 21(3):165–181, 2007.

Jiahong Wu, He Zheng, Bo Zhao, Yixin Li, Baoming Yan, Rui Liang, Wenjia Wang, Shipei Zhou, Guosen Lin, Yanwei Fu, et al. Ai challenger: A large-scale dataset for going deeper in image understanding. *arXiv preprint arXiv:1711.06475*, 2017.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning (ICML)*, 2015.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

Zhirui Zhang, Shujie Liu, Mu Li, Ming Zhou, and Enhong Chen. Joint training for neural machine translation models with monolingual data. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

Dengyong Zhou, Olivier Bousquet, Thomas N Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems (NIPS)*, 2004.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

# Dual Attention Networks for Visual Reference Resolution in Visual Dialog

**Gi-Cheon Kang**
Seoul National University
chonkang@snu.ac.kr

**Jaeseo Lim**
Seoul National University
jaeseolim@snu.ac.kr

**Byoung-Tak Zhang**
Seoul National University
Surromind Robotics
btzhang@snu.ac.kr

## Abstract

Visual dialog (VisDial) is a task which requires a dialog agent to answer a series of questions grounded in an image. Unlike in visual question answering (VQA), the series of questions should be able to capture a temporal context from a dialog history and utilizes visually-grounded information. Visual reference resolution is a problem that addresses these challenges, requiring the agent to resolve ambiguous references in a given question and to find the references in a given image. In this paper, we propose Dual Attention Networks (DAN) for visual reference resolution in VisDial. DAN consists of two kinds of attention modules, **REFER** and **FIND**. Specifically, REFER module learns latent relationships between a given question and a dialog history by employing a multi-head attention mechanism. FIND module takes image features and reference-aware representations (i.e., the output of REFER module) as input, and performs visual grounding via bottom-up attention mechanism. We qualitatively and quantitatively evaluate our model on VisDial v1.0 and v0.9 datasets, showing that DAN outperforms the previous state-of-the-art model by a significant margin.

## 1 Introduction

Thanks to the recent progresses in natural language processing and computer vision, there has been an extensive amount of effort towards developing a cognitive agent that jointly understand natural language and vision information. Over the last few years, vision-language tasks such as image captioning (Xu et al., 2015) and visual question answering (VQA) (Antol et al., 2015; Anderson et al., 2018) have provided a testbed for developing a cognitive agent. However, the agent performing these tasks still has a long way to go to be used in real-world applications (*e.g.*, aiding visually impaired users, interacting with humanoid robots) in that it does not consider the continuous interaction over time. Specifically, the interaction in image captioning is that the agent simply talks to human about visual content, without any input from human. While the VQA agent takes a question as input, it is required to answer a *single* question about a given image.

Visual dialog (VisDial) (Das et al., 2017) task has been introduced as a generalized version of VQA. A dialog agent needs to answer a series of questions such as "How many people are in the image?", "Are they indoors or outside?", utilizing not only visually-grounded information but also contextual information from a dialog history. To address these two challenges, researchers have recently tackled a problem called visual reference resolution in VisDial. The problem of visual reference resolution is to resolve ambiguous expressions on their own (*e.g.*, it, they, any other) and ground them to a given image.

In this paper, we address the visual reference resolution in a visual dialog task. We first hypothesize that humans address the visual reference resolution through a two-step process: (1) linguistically resolve the ambiguous questions by recalling the dialog history from one's memory and (2) find a local region of a given image for the resolved questions. For example, as shown in Figure 1, the question "Does it look like a nice one?" is ambiguous on its own because we do not know what "it" refers to. So we believe that humans try to recall the dialog history and implicitly find out "it" refers to the "skateboard". After the resolution step, we believe that they will finally try to find the skateboard in the image and answer the question. For these processes, we propose Dual Attention Networks (DAN) which consists of two kinds of attention modules, REFER and FIND. REFER module learns to retrieve the relevant previous dialogs for clarifying am-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2024–2033,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

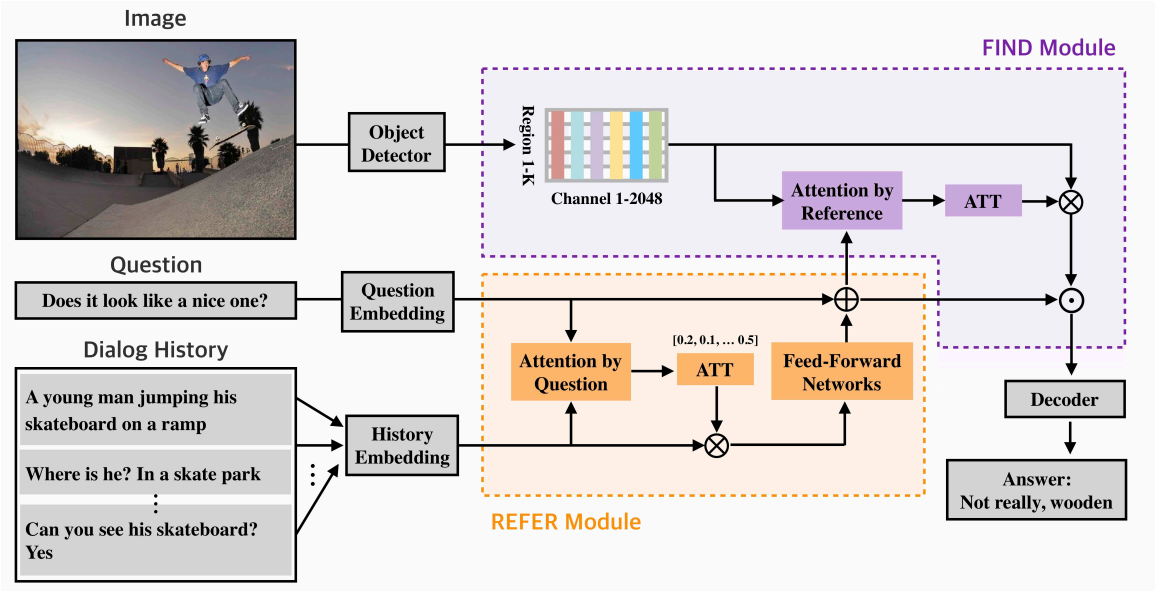


Figure 1: An overview of Dual Attention Networks (DAN). We propose two kinds of attention modules, REFER and FIND. REFER learns latent relationships between a given question and a dialog history to retrieve the relevant previous dialogs. FIND performs visual grounding, taking image features and reference-aware representations (*i.e.,* the output of REFER). $\otimes$, $\oplus$, and $\odot$ denote matrix multiplication, concatenation and element-wise multiplication, respectively. The multi-layer perceptron is omitted in this figure for simplicity.

biguous questions. Inspired by the self-attention mechanism (Vaswani et al., 2017), REFER module computes multi-head attention over all previous dialogs in a sentence-level fashion, followed by feed-forward networks to get the *reference-aware* representations. FIND module takes image features and the *reference-aware* representations, and performs visual grounding via bottom-up attention mechanism. From this pipeline, we expect our proposed model to be capable of question disambiguation by using REFER module and ground the resolved reference properly to the given image.

The main contributions of this paper are as follows. First, we propose Dual Attention Networks (DAN) for visual reference resolution in visual dialog based on REFER and FIND modules. Second, we validate our proposed model on the large-scale datasets: VisDial v1.0 and v0.9. Our model achieves a new state-of-the-art results compared to other methods. We also conduct ablation studies by four criteria to demonstrate the effectiveness of our proposed components. Third, we make a comparison between DAN and our baseline model to demonstrate the performance improvements on *semantically incomplete* questions needed to be clarified. Finally, we perform qualitative analysis of our model, showing that DAN reasonably attends to the dialog history and salient image regions. Our code is available at `https://github.com/gicheonkang/DAN-VisDial`.

## 2 Related Work

**Visual Dialog.** Visual dialog (VisDial) task was recently proposed by (Das et al., 2017), providing a testbed for research on the interplay between computer vision and dialog systems. Accordingly, a dialog agent performing this task is not only required to find visual groundings of linguistic expressions but also capture semantic nuances from human conversation. Attention-based approaches were primarily proposed to address these challenges, including memory networks (Das et al., 2017), history-conditioned image attentive encoder (Lu et al., 2017), sequential co-attention (Wu et al., 2018), and synergistic co-attention networks (Guo et al., 2019).

**Visual Reference Resolution.** Recently, researchers have tackled a problem called visual reference resolution (Seo et al., 2017; Kottur et al., 2018; Niu et al., 2018) in VisDial. To resolve visual references, (Seo et al., 2017) proposed an attention memory which stores a sequence of previous visual attention maps in memory slots. They retrieved the previous visual attention maps by applying a soft attention over all the memory slots and combined it with a current visual attention.

Furthermore, (Kottur et al., 2018) attempted to resolve visual references at a word-level, relying on an off-the-shelf parser. Similar to the attention memory (Seo et al., 2017), they proposed a reference pool which stores visual attention maps of recognized entities and retrieved the weighted sum of the visual attention maps by applying a soft attention. (Niu et al., 2018) proposed a recursive visual attention model that recursively reviews the previous dialogs and refines the current visual attention. The recursion is continued until the question itself is determined to be unambiguous. A binary decision whether the questions is ambiguous or not is made by Gumbel-Softmax approximation (Jang et al., 2016; Maddison et al., 2016). To resolve the visual references, above approaches attempted to retrieve the visual attention of the previous dialogs, and applied it on the current visual attention. These approaches have limitations in that they store all previous visual attentions, while researches in human memory system show that the visual sensory-memory, due to its rapid decay property, hardly stores all previous visual attentions (Sperling, 1960; Sergent et al., 2011). Based on this biologically inspired motivation, our proposed model calculates the current visual attention by using linguistic cues (*i.e.,* dialog history).

## 3 Proposed Algorithm

In this section, we formally describe the visual dialog task and our proposed algorithm, Dual Attention Networks (DAN). The visual dialog task (Das et al., 2017) is defined as follows. A dialog agent is given an input such as an image $I$, a follow-up question at round t as $Q_t$, and a dialog history (including the image caption) until round $\mathrm{t}-1$,

$$H = (\underbrace{C}_{H_0}, \underbrace{(Q_1, A_1^{gt})}_{H_1}, \cdots, \underbrace{(Q_{t-1}, A_{t-1}^{gt})}_{H_{t-1}}).$$

$A_t^{gt}$ denotes the ground truth answer (*i.e.,* human response) at round t. By using these inputs, the agent is asked to rank a list of 100 candidate answers, $A_t = \{A_t^1, \cdots, A_t^{100}\}$.

Given the problem setup, DAN for visual dialog task can be framed as an encoder-decoder architecture: (1) an encoder that jointly embeds the input ($I$, $Q_t$, $H$) and (2) a decoder that converts the embedded representation into the ranked list $\hat{A}_t$. From this point of view, DAN consists of three components which are REFER, FIND, and the answer decoder. As shown in Figure 1, REFER module learns to attend relevant previous dialogs to resolve the ambiguous references in a given question $Q_t$. FIND module learns to attend to the spatial image features that the output of REFER module describes. Answer decoder ranks the list of candidate answers $A_t$ given the output of FIND module.

We first introduce the language features, as well as the image features in Sec. 3.1. Then we describe the detailed architectures of the REFER and FIND modules in Sec. 3.2 and 3.3, respectively. Finally, we present the answer decoder in Sec. 3.4.

### 3.1 Input Representation

**Language Features.** We first embed each word in the follow-up question $Q_t$ to $\{w_{t,1}, \cdots, w_{t,T}\}$ by using pre-trained GloVe (Pennington et al., 2014) embeddings, where $T$ denotes the number of tokens in $Q_t$. We then use a two-layer LSTM, generating a sequence of hidden states $\{u_{t,1}, \cdots, u_{t,T}\}$. Note that we use the last hidden state of the LSTM $u_{t,T}$ as a question feature, denoted as $q_t \in \mathbb{R}^L$.

$$u_{t,i} = \mathrm{LSTM}(w_{t,i}, u_{t,i-1}) \tag{1}$$

$$q_t = u_{t,T} \tag{2}$$

Also, each element in the dialog history $\{H_i\}_{i=0}^{t-1}$ and the candidate answers $\{A_t^i\}_{i=1}^{100}$ are embedded as the follow-up question, yielding $\{h_i\}_{i=0}^{t-1} \in \mathbb{R}^{t \times L}$ and $\{o_t^i\}_{i=1}^{100} \in \mathbb{R}^{100 \times L}$. $Q_t$, $H$, and $A_t$ are embedded with same word embedding vector and three different LSTMs.

**Image Features.** Inspired by bottom-up attention (Anderson et al., 2018), we use the Faster R-CNN (Ren et al., 2015) pre-trained with Visual Genome (Krishna et al., 2017) to extract the object-level image features. We denote the output features as $v \in \mathbb{R}^{K \times V}$, where $K$ and $V$ are the total number of object detection features per image and dimension of the each feature, respectively. We adaptively extract the number of object features $K$ ranging from 10 to 100 for reflecting the complexity of each image. $K$ is fixed during training.

### 3.2 REFER Module

In this section, we formally describe the single-layer REFER module. Given the question and dialog history features, REFER module aims to attend to the most relevant elements of dialog history with respect to the given question. Specifically, we first compute scaled dot product attention (Vaswani et al., 2017) in multi-head settings

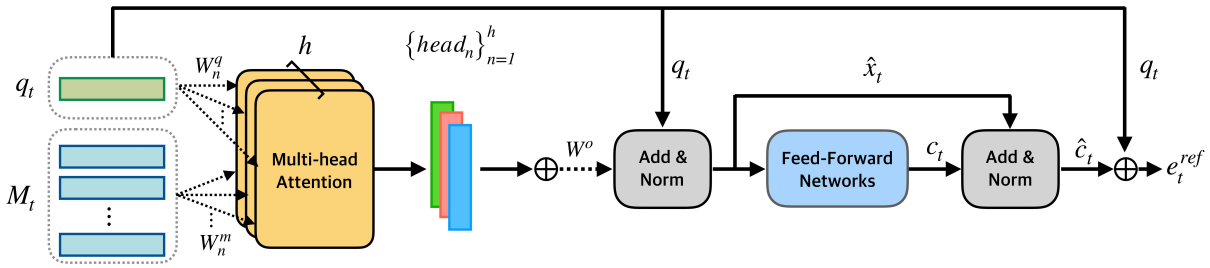


Figure 2: Illustration of the single-layer REFER module. REFER module focuses on the latent relationship between the follow-up question and a dialog history to resolve ambiguous references in the question. We employ two submodule: multi-head attention and feed-forward networks. Multi-head attention computes the $h$ number of soft attentions over all elements of dialog history by using scaled dot product attention. Then, it returns the $h$ number of heads which are weighted by the attentions. Followed by the two-layer feed-forward networks, REFER module finally returns the reference-aware representations $e_t^{ref}$. $\oplus$ and Dotted line denote the concatenation operation and linear projection operation by the learnable matrices, respectively.

which are called multi-head attention. Let $q_t$ and $M_t = \{h_i\}_{i=0}^{t-1}$ be the question and dialog history feature vectors, respectively. $q_t$ and $M_t$ are projected to $d_{ref}$ dimensions by different and learnable projection matrices. We then conduct dot product of these two projected matrices, divide by $\sqrt{d_{ref}}$, and apply the softmax function to obtain the attention weights on the all elements in the dialog history. It is formulated as below,

$$head_n = \text{Attention}(q_t W_n^q, M_t W_n^m) \quad (3)$$

$$\text{Attention}(a, b) = \text{softmax}(\frac{ab^\top}{\sqrt{d_{ref}}})b \quad (4)$$

where $W_n^q \in \mathbb{R}^{L \times d_{ref}}$ and $W_n^m \in \mathbb{R}^{L \times d_{ref}}$. Note that dot product attention is computed $h$ times with different projection matrices, yielding $\{head_n\}_{n=1}^h$. Accordingly, we can get the multi-head representations $x_t$, concatenating all $\{head_n\}_{n=1}^h$, followed by linear projection. Also, we can compute $\hat{x}_t$ by applying a residual connection (He et al., 2016), followed by layer normalization (Ba et al., 2016).

$$x_t = (head_1 \oplus \cdots \oplus head_h)\, W^o \quad (5)$$

$$\hat{x}_t = \text{LayerNorm}(x_t + q_t) \quad (6)$$

where $\oplus$ denotes the concatenation operation, and $W^o \in \mathbb{R}^{hd_{ref} \times L}$ is the projection matrix. Next, we apply $\hat{x}_t$ to two-layer feed-forward networks with a ReLU in between, where $W_1^f \in \mathbb{R}^{L \times 2L}$ and $W_2^f \in \mathbb{R}^{2L \times L}$. The residual connection and layer normalization is also applied in this step.

$$c_t = \text{ReLU}(\hat{x}_t W_1^f + b_1^f) W_2^f + b_2^f \quad (7)$$

$$\hat{c}_t = \text{LayerNorm}(c_t + \hat{x}_t) \quad (8)$$

$$e_t^{ref} = \hat{c}_t \oplus q_t \quad (9)$$

Finally, REFER module returns the *reference-aware* representations by concatenating the contextual representation $\hat{c}_t$ and the original question representation $q_t$, denoted as $e_t^{ref} \in \mathbb{R}^{2L}$. In this work, we use $d_{ref} = 256$. Figure 2 illustrates the pipeline of the REFER module.

Furthermore, we stack the REFER modules in multiple layers to get a high-level abstraction of the reference-aware representations. Details are to be discussed in Sec. 4.5.

### 3.3 FIND Module

Instead of relying on the visual attention maps of the previous dialogs as in (Seo et al., 2017; Kottur et al., 2018; Niu et al., 2018), we expect the FIND module to attend to the most relevant regions of the image with respect to the reference-aware representations (*i.e.,* the output of REFER module). In order to implement the visual grounding for the reference-aware representations, we take inspiration from bottom-up attention mechanism (Anderson et al., 2018). Let $v \in \mathbb{R}^{K \times V}$ and $e_t^{ref} \in \mathbb{R}^{2L}$ be the image feature vectors and reference-aware representations, respectively. We first project these two vectors to $d_{find}$ dimensions and compute soft attention over all the object detection features as follows:

$$r_t = f_v(v) \odot f_{ref}(e_t^{ref}) \tag{10}$$

$$\alpha_t = \text{softmax}(r_t W^r + b^r) \tag{11}$$

where $f_v(\cdot)$ and $f_{ref}(\cdot)$ denote the two-layer multi-layer perceptrons which convert to $d_{find}$ dimensions, and $W^r \in \mathbb{R}^{d_{find} \times 1}$ is the projection matrix for the softmax activation. $\odot$ denotes hadamard product (*i.e.,* element-wise multiplication). From these equations, we can get the visual attention weights $\alpha_t \in \mathbb{R}^{K \times 1}$. Next, we apply the visual attention weights to $v$ and compute the vision-language joint representations as follows:

$$\hat{v}_t = \sum_{j=1}^{K} \alpha_{t,j} v_j \tag{12}$$

$$z_t = f'_v(\hat{v}_t) \odot f'_{ref}(e_t^{ref}) \tag{13}$$

$$e_t^{find} = z_t W^z + b^z \tag{14}$$

where $f'_v(\cdot)$ and $f'_{ref}(\cdot)$ also denote the two-layer multi-layer perceptrons which convert to $d_{find}$ dimensions, and $W^z \in \mathbb{R}^{d_{find} \times L}$ is the projection matrix. Note that $e_t^{find} \in \mathbb{R}^L$ is the output representations of the encoder as well as FIND module which is decoded to score the list of candidate answers. In this work, we use $d_{find} = 1024$.

### 3.4 Answer Decoder

Answer decoder computes each score of candidate answers via a dot product with the embedded representation $e_t^{find}$, followed by a softmax activation to get a categorical distribution over the candidates. Let $O_t = \{o_t^i\}_{i=1}^{100} \in \mathbb{R}^{100 \times L}$ be the feature vectors of 100 candidate answers. The distribution $p_t$ is formulated as follows:

$$p_t = \text{softmax}(e_t^{find} O_t^\top) \tag{15}$$

In training phase, DAN is optimized by minimizing the cross-entropy loss between the one-hot encoded label vector (*i.e.*, $y_t$) and probability distribution (*i.e.*, $p_t$).

$$\mathcal{L}(\theta) = -\sum_k y_{t,k} \log p_{t,k} \tag{16}$$

Where $p_{t,k}$ denotes the probability of the $k$-$th$ candidate answer at round $\mathrm{t}$. In test phase, the list of candidate answers is ranked by the distribution $p_t$, and evaluated by the given metrics.

## 4 Experiments

In this section, we describe the details of our experiments on the VisDial v1.0 and v0.9 datasets. We first introduce the VisDial datasets, evaluation metrics, and implementation details in Sec. 4.1, Sec. 4.2, and Sec. 4.3, respectively. Then we report the quantitative results by comparing our proposed model with the state-of-the-art approaches and baseline model in Sec. 4.4. Then, we conduct the ablation studies by four criteria to report the relative contributions of each components in Sec. 4.5. Finally, we provide the qualitative results in Sec. 4.6.

### 4.1 Datasets

We evaluate our proposed model on the VisDial v0.9 and v1.0 dataset. VisDial v0.9 dataset (Das et al., 2017) has been collected from two annotators chatting log about MS-COCO (Lin et al., 2014) images. Each dialog is made up of an image, a caption from MS-COCO dataset and 10 QA pairs. As a result, VisDial v0.9 dataset contains 83k dialogs and 40k dialogs as train and validation splits, respectively. Recently, VisDial v1.0 dataset (Das et al., 2017) has been released with an additional 10k COCO-like images from Flickr. Dialogs for the additional images have been collected similar to v0.9. Overall, VisDial v1.0 dataset contains 123k (all dialogs from v0.9), 2k, and 8k dialogs as train, validation, and test splits, respectively.

### 4.2 Evaluation Metrics

We evaluate individual responses at each question in a retrieval setting according to (Das et al., 2017). Specifically, the dialog agent is given a list of 100 candidate answers of each question and asked to rank the list. There are three kinds of evaluation metrics for retrieval performance: (1) mean rank of human response, (2) recall@k (*i.e.,* existence of the human response in top-k ranked response), and (3) mean reciprocal rank (MRR). Mean rank, recall@k, and MRR are highly correlated with the rank of human response. In addition, (Das et al., 2017) proposed to use the robust evaluation metric, normalized discounted cumulative gain (NDCG). NDCG takes into account all relevant answers from the ranked list, where the relevance scores are densely annotated for VisDial v1.0 test split. NDCG penalizes the lower rank of the candidate answers with high relevance scores.

| | VisDial v1.0 (test-std) | | | | | | VisDial v0.9 (val) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | NDCG | MRR | R@1 | R@5 | R@10 | Mean | MRR | R@1 | R@5 | R@10 | Mean |
| LF (Das et al., 2017) | 45.31 | 55.42 | 40.95 | 72.45 | 82.83 | 5.95 | 58.07 | 43.82 | 74.68 | 84.07 | 5.78 |
| HRE (Das et al., 2017) | 45.46 | 54.16 | 39.93 | 70.45 | 81.50 | 6.41 | 58.46 | 44.67 | 74.50 | 4.22 | 5.72 |
| MN (Das et al., 2017) | 47.50 | 55.49 | 40.98 | 72.30 | 83.30 | 5.92 | 59.65 | 45.55 | 76.22 | 85.37 | 5.46 |
| HCIAE (Lu et al., 2017) | - | - | - | - | - | - | 62.22 | 48.48 | 78.75 | 87.59 | 4.81 |
| AMEM (Seo et al., 2017) | - | - | - | - | - | - | 62.27 | 48.53 | 78.66 | 87.43 | 4.86 |
| CoAtt (Wu et al., 2018) | - | - | - | - | - | - | 63.98 | 50.29 | 80.71 | 88.81 | 4.47 |
| CorefNMN (Kottur et al., 2018) | 54.70 | 61.50 | 47.55 | 78.10 | 88.80 | 4.40 | 64.10 | 50.92 | 80.18 | 88.81 | 4.45 |
| RvA (Niu et al., 2018) | 55.59 | 63.03 | 49.03 | 80.40 | 89.83 | 4.18 | 66.34 | 52.71 | **82.97** | **90.73** | **3.93** |
| Synergistic (Guo et al., 2019) | 57.32 | 62.20 | 47.90 | **80.43** | **89.95** | **4.17** | - | - | - | - | - |
| DAN (ours) | **57.59** | **63.20** | **49.63** | 79.75 | 89.35 | 4.30 | **66.38** | **53.33** | 82.42 | 90.38 | 4.04 |

Table 1: Retrieval performance on VisDial v1.0 and v0.9 datasets, measured by normalized discounted cumulative gain (NDCG), mean reciprocal rank (MRR), recall @k (R@k), and mean rank. The higher the better for NDCG, MRR, and R@k, while the lower the better for mean rank. DAN outperforms all other models across NDCG, MRR, and R@1 on both datasets. NDCG is not supported in v0.9 dataset.

### 4.3 Implementation Details

The dimension of image features $V$ and hidden states in all LSTM $L$ is 2048 and 512, respectively. All the language intputs are embedded into a 300-dimensional vector initialized by GloVe (Pennington et al., 2014). The number of attention heads $h$ is fixed to 4 except for the ablation study that changes it. We apply Adam optimizer (Kingma and Ba, 2014) with learning rate $1 \times 10^{-3}$, decreased by $1 \times 10^{-4}$ per epoch until epoch 7, decayed by 0.5 per epoch from 8 to 12 epochs.

### 4.4 Quantitative Results

**Compared Methods.** We compare our proposed model with the state-of-the-art approaches on VisDial v1.0 and v0.9 datasets, which can be categorized into three groups: (1) Fusion-based approaches (LF and HRE (Das et al., 2017)), (2) Attention-based approaches (MN (Das et al., 2017), HCIAE (Lu et al., 2017), CoAtt (Wu et al., 2018) and Synergistic (Guo et al., 2019)), and (3) Approaches that deal with visual reference resolution in VisDial (AMEM (Seo et al., 2017), CorefNMN (Kottur et al., 2018) and RvA (Niu et al., 2018)). Our proposed model belongs to the third category.

**Results on VisDial v1.0 and v0.9 datasets.** As shown in Table 1, DAN significantly outperforms all other approaches on NDCG, MRR, and R@1, including the previous state-of-the-art method, Synergistic (Guo et al., 2019). Specifically, DAN improves approximately 1.0% on MRR, 1.7% on R@1 and 0.3% on NDCG in VisDial v1.0 dataset. The results indicate that our proposed model ranks higher than all other methods on both single ground-truth answer (R@1) and all relevant answers on average (NDCG).

| Model | NDCG | MRR | R@1 | R@5 | R@10 | Mean |
|---|---|---|---|---|---|---|
| MS ConvAI | 55.35 | 63.27 | 49.53 | 80.40 | 89.60 | 4.15 |
| USTC-YTH | 56.47 | 61.44 | 47.65 | 78.13 | 87.88 | 4.65 |
| Synergistic | 57.88 | 63.42 | 49.30 | 80.77 | 90.68 | 3.97 |
| DAN (ours) | **59.36** | **64.92** | **51.28** | **81.60** | **90.88** | **3.92** |

Table 2: Test-std performance of ensemble model on VisDial v1.0 dataset. We cite top-three entries from VisDial Challenge 2018 Leaderboard.

**Results on ensemble model.** We report the performance of ensemble model in comparison with the top-three entries in the leaderboard[1] of VisDial Challenge 2018. We ensemble six DAN models, using the number of attention heads (*i.e.,* $h$) ranging from one to six. We average the probability distribution (*i.e.,* $p_t$) of the six models to rank the candidate answers. In Table 2, our model significantly outperforms all three challenge entries, including the challenge winner model, Synergistic (Guo et al., 2019). They ensembled ten models with different weight initialization and also used bottom-up attention features (Anderson et al., 2018) as image features.

**Results on semantically complete & incomplete questions.** We first define the questions that contain one or more pronouns (*i.e.,* it, its, they, their, them, these, those, this, that, he, his, him, she,

[1] https://evalai.cloudcv.org/web/challenges/challenge-page/103/leaderboard

| | Model | MRR | R@1 | R@5 | R@10 | Mean |
|---|---|---|---|---|---|---|
| SC | No REFER | 61.85 | 47.80 | 79.10 | 88.43 | 4.49 |
| | DAN | 64.81 | 51.22 | 81.63 | 90.19 | 4.03 |
| | Improvements | 2.96 | 3.42 | 2.53 | 1.76 | 0.46 |
| SI | No REFER | 58.44 | 44.38 | 75.36 | 85.48 | 5.36 |
| | DAN | 61.77 | 48.13 | 78.43 | 87.81 | 4.70 |
| | Improvements | 3.33 | 3.75 | 3.07 | 2.33 | 0.66 |

Table 3: VisDial v1.0 validation performance on the semantically complete (SC) and incomplete (SI) questions. We observe that SI questions obtain more benefits from the dialog history than SC questions.

her) as the *semantically incomplete* (SI) questions. Also, we can declare the questions that do not have pronouns as *semantically complete* (SC) questions. Then, we have checked the contribution of the *reference-aware* representations for the SC and SI questions, respectively. Specifically, we make a comparison between DAN, which utilizes *reference-aware* representations (*i.e.,* $e_t^{ref}$), and No REFER, which exploits question representations (*i.e.,* $q_t$) only. From the Table 3, we draw three observations: (1) DAN shows significantly better results than the No REFER model for SC questions. It validates that the context from dialog history enriches the question information, even when the question is semantically complete. (2) SI questions obtain more benefits from the dialog history than SC questions. It indicates that DAN is more robust to the SI questions than SC questions. (3) A dialog agent faces greater difficulty in answering SI questions compared to SC questions. No REFER is equivalent to the FIND + RPN model in the ablation study section.

### 4.5 Ablation Study

In this section, we perform ablation study on VisDial v1.0 validation split with the following four model variants: (1) Model only using the single attention module, (2) Model that uses different image features (pre-trained VGG-16 is used), (3) Model that does not use the residual connection in REFER module, and (4) Model that stacks the REFER modules up to four layers with each different number of attention heads.

**Single Module.** The first four rows in Table 4 show the performance of a single module. FIND denotes the use of FIND module only, and REFER denotes the use of single-layer REFER module only. Specifically, REFER uses the output of REFER module as the encoder outputs. On the other hand, FIND does not take the reference-aware representations (*i.e.,* $e_t^{ref}$) but the question feature (*i.e.,* $q_t$). The single models show relatively poor performance compared with the dual module model. We believe that the results validate two hypotheses: (1) VisDial task requires contextual information from dialog history as well as the visually-grounded information. (2) REFER and FIND modules have complementary modeling abilities.

| Model | MRR Score |
|---|---|
| FIND | 57.85 |
| FIND + RPN | 60.80 |
| REFER | 57.18 |
| REFER + Res | 58.69 |
| REFER + FIND | 60.98 |
| REFER + Res + FIND | 61.86 |
| REFER + FIND + RPN | 63.47 |
| REFER + Res + FIND + RPN | 63.88 |

Table 4: Ablation studies on VisDial v1.0 validation split. Res and RPN denote the residual connection and the region proposal networks, respectively.

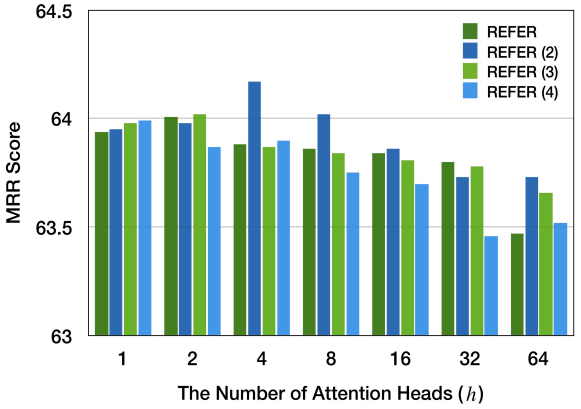


Figure 3: Ablation study on a different number of attention heads and REFER stacks. REFER ($n$) indicates that DAN uses a stack of $n$ identical REFER modules.

**Image Features in FIND Module.** To report the impact of image features, we replace the bottom-up attention features (Anderson et al., 2018) with ImageNet pre-trained VGG-16 (Simonyan and Zisserman, 2014) features. In detail, we use the output of the VGG-16 *pool5* layer as image features. In Table 4, RPN denotes the use of the region proposal networks (Ren et al., 2015) which are equivalent to the use of bottom-up attention features. Similar to VQA task, we observe that DAN with bottom-up attention features achieves

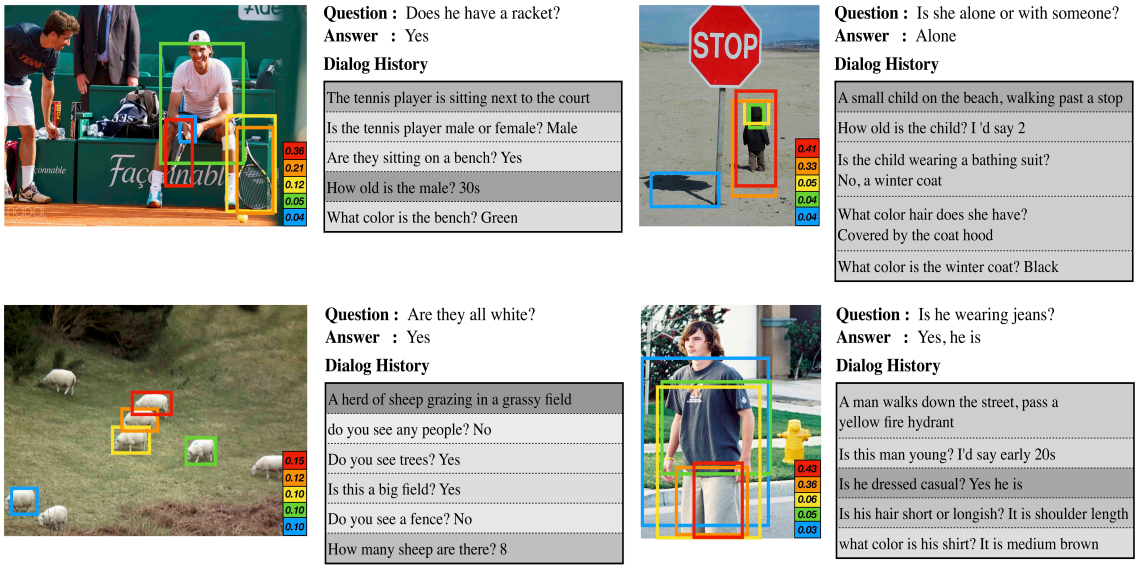


Figure 4: Qualitative results on the VisDial v1.0 dataset. We visualize the attention over dialog history from REFER module and the visual attention from FIND module. The object detection features with top five attention weights are marked with colored box. A red colored box indicates the most salient visual feature. Also, the attention from REFER module is represented as shading, darker shading indicates the larger attention weight for each element of the dialog history. Our proposed model not only responds to the correct answer, but also selectively pays attention to the previous dialogs and salient image regions.

better performance than with VGG-16 features. In other words, the use of object-level features boosts the MRR performance of DAN.

**Residual Connection in REFER Module.** We also conduct an ablation study to investigate the effectiveness of the residual connection in REFER module. As shown in Table 4, the use of the residual connection (*i.e.,* Res) boosts the MRR score of DAN. In other words, DAN utilizes the excellence of deep residual learning as in (He et al., 2016; Rocktäschel et al., 2015; Yang et al., 2016; Kim et al., 2016; Vaswani et al., 2017).

**Stack of REFER Modules & Attention Heads.** We stack the REFER modules up to four layers with each different number of attention heads, $h \in \{1, 2, 4, 8, 16, 32, 64\}$. In other words, we conduct the ablation experiments with twenty-eight models to set the hyperparameters of our model. Figure 3 shows the results of the ablation experiments. For $n \geq 2$, REFER $(n)$ indicates that DAN uses a stack of $n$ identical REFER modules. Specifically, for each pair of successive modules, the output of the previous REFER module is fed into the next REFER module as a query (*i.e.,* $q_t$). Due to the small number of elements in each dialog history, the overall performance pattern shows a tendency to decrease as the number of attention heads increases. It turns out that the two-layer REFER module with four attention heads (*i.e.,* REFER (2) and $h = 4$) performs the best among all models in ablation study, recording 64.17% on MRR.

### 4.6 Qualitative Results

In this section, we visualize the inference mechanism of our proposed model. Figure 4 shows the qualitative results of DAN. Given a question that is needed to be clarified, DAN correctly answers the question by selectively attending to each element of the dialog history and salient image regions. In case of the visual attention, we mark the object detection features with top five attention weights of each image. On the other hand, the attention weights from REFER module are represented as shading; darker shading indicates the larger attention weight for each element of the dialog history. These attention weights are calculated by averaging over all the attention heads.

## 5 Conclusion

We introduce Dual Attention Networks (DAN) for visual reference resolution in visual dialog task. DAN explicitly divides the visual reference resolution problem into a two-step process. Rather than relying on the previous visual attention maps as in prior works, DAN first linguistically resolves

ambiguous references in a given question by using REFER module. Then, it grounds the resolved references in the image by using FIND module. We empirically validate our proposed model on VisDial v1.0 and v0.9 datasets. DAN achieves the new state-of-the-art performance, while being simpler and more grounded.

**Acknowledgements** The authors would like to thank Woosuk Choi, Seunghee Yang, Junseok Park, and Delilah Hollweg for helpful comments and editing. This work was partly supported by the Korea government (2015-0-00310-SW.StarLab, 2017-0-01772-VTT, 2018-0-00622-RMI, 2019-0-01367-BabyMind, 10060086-RISF, P0006720-GENKO), and the ICT at Seoul National University.

## References

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and visual question answering. In *CVPR*.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José MF Moura, Devi Parikh, and Dhruv Batra. 2017. Visual dialog. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2.

Dalu Guo, Chang Xu, and Dacheng Tao. 2019. Image-question-answer synergistic network for visual dialog. *arXiv preprint arXiv:1902.09774*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Jin-Hwa Kim, Sang-Woo Lee, Donghyun Kwak, Min-Oh Heo, Jeonghee Kim, Jung-Woo Ha, and Byoung-Tak Zhang. 2016. Multimodal residual learning for visual qa. In *Advances in neural information processing systems*, pages 361–369.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Satwik Kottur, José MF Moura, Devi Parikh, Dhruv Batra, and Marcus Rohrbach. 2018. Visual coreference resolution in visual dialog using neural module networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 153–169.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123(1):32–73.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Jiasen Lu, Anitha Kannan, Jianwei Yang, Devi Parikh, and Dhruv Batra. 2017. Best of both worlds: Transferring knowledge from discriminative learning to a generative visual dialog model. In *Advances in Neural Information Processing Systems*, pages 314–324.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. 2016. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*.

Yulei Niu, Hanwang Zhang, Manli Zhang, Jianhong Zhang, Zhiwu Lu, and Ji-Rong Wen. 2018. Recursive visual attention in visual dialog. *arXiv preprint arXiv:1812.02664*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*.

Paul Hongsuck Seo, Andreas Lehrmann, Bohyung Han, and Leonid Sigal. 2017. Visual reference resolution using attention memory for visual dialog. In *Advances in neural information processing systems*, pages 3719–3729.

Claire Sergent, Christian C Ruff, Antoine Barbot, Jon Driver, and Geraint Rees. 2011. Top–down modulation of human early visual cortex after stimulus

offset supports successful postcued report. *Journal of Cognitive Neuroscience*, 23(8):1921–1934.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

George Sperling. 1960. The information available in brief visual presentations. *Psychological monographs: General and applied*, 74(11):1.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Qi Wu, Peng Wang, Chunhua Shen, Ian Reid, and Anton van den Hengel. 2018. Are you talking to me? reasoned visual dialog generation through adversarial learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6106–6115.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057.

Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. 2016. Stacked attention networks for image question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 21–29.

# Unsupervised Discovery of Multimodal Links in Multi-image, Multi-sentence Documents

**Jack Hessel** **Lillian Lee** **David Mimno**
Cornell University
{jhessel, llee}@cs.cornell.edu mimno@cornell.edu

## Abstract

Images and text co-occur constantly on the web, but explicit links between images and sentences (or other intra-document textual units) are often not present. We present algorithms that discover image-sentence relationships *without* relying on explicit multimodal annotation in training. We experiment on seven datasets of varying difficulty, ranging from documents consisting of groups of images captioned post hoc by crowdworkers to naturally-occurring user-generated multimodal documents. We find that a structured training objective based on identifying whether *collections* of images and sentences co-occur in documents can suffice to predict links between specific sentences and specific images within the *same document* at test time.

## 1 Introduction

Images and text act as natural complements on the modern web. News stories include photographs, product listings show multiple images providing detail for online shoppers, and Wikipedia pages include maps, diagrams, and pictures. But the exact matching between words and images is often left implicit. Algorithms that identify document-internal connections between specific images and specific passages of text could have both immediate and long-term promise. On the user-experience front, alt-text for vision-impaired users could be produced automatically (Wu et al., 2017) via intra-document retrieval, and user interfaces could explicitly link images to descriptive sentences, potentially improving the reading experience of sighted users. Also, in terms of improving other applications, the text in multimodal documents can be viewed as a noisy form of image annotation: inferred image-sentence associations can serve as training pairs for vision models, particularly in domains lacking readily-available labeled data.

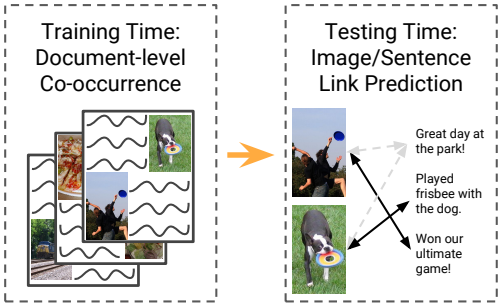


Figure 1: At training time, we assume we are given a set of multi-image/multi-sentence documents. At test-time, we predict links between individual images and individual sentences within single documents. Because no explicit multimodal annotation is available at training time, we refer to this task as unsupervised.

In this work, we develop *unsupervised* models that learn to identify multimodal within-document links *despite not having access to supervision at the individual image/sentence level during training.* Rather, the training documents contain multiple images and multiple sentences[1] that are not aligned, as illustrated in Figure 1.

Our intra-document setting poses challenges beyond those encountered in the usual cross-modal retrieval framework, wherein "documents" generally consist of a single image associated with a single piece of text, e.g., an image caption. For the longer documents we consider, a sentence may have many corresponding images or no corresponding images, and vice versa. Furthermore, we expect that images *within* documents will be, on average, more similar than images *across* documents, thus making disambiguation more difficult than in the usual one-image/one-sentence case.

Our approach for this difficult setting is ranking-based: we train algorithms to score image collec-

[1] Any discrete textual unit could be used, such as n-grams or paragraphs. We focus on sentences because there exist public sentence-level datasets that we can use for evaluation.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2034–2045, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

tions and sentence collections that truly co-occur more highly than image collections and sentence collections that do not co-occur. The matching functions we consider predict a latent similarity-weighted bipartite graph over a document's images and sentences; at test time, we evaluate this internal bipartite graph representation learned by our models for the task of intra-document link prediction.

We work with a variety of datasets (one of which we introduce), ranging from concatenations of individually-captioned images to organically-multimodal documents scraped from noisy, user-generated web content.[2] Despite having no supervision at the individual image-sentence level, our algorithms perform well on the same-document link prediction task. For example, on a visual storytelling dataset, we achieve 90+ AUC, even in the presence of a large number of sentences that do not correspond to any images in the document. Similarly, for organically-multimodal web data, we are able to surpass object-detection baselines by a wide margin, e.g., for a step-by-step recipe dataset, we improve precision by 20 points on link prediction within documents by leveraging document-level co-occurrence during training.

We conclude by using our algorithm to *discover* links within a Wikipedia image/text dataset that lacks ground-truth image-sentence links. While the predictions are imperfect, the algorithm qualitatively identifies meaningful patterns, such as matching an image of a dodo bird to one of two sentences (out of 100) in the corresponding article that mention "dodo".

## 2 Task Formulation

We assume as given a set of documents where each document $d_i = \langle S_i, V_i \rangle$ consists of a set $S_i$ of $n_i = |S_i|$ sentences and a set $V_i$ of $m_i = |V_i|$ images.[3] For example, $d_i$ could be an article about Paris with $n_i = 100$ sentences and $m_i = 3$ images of, respectively, the Eiffel Tower, the Arc de Triomphe, and a map of Paris. For each $d_i$, we are to predict an alignment — where some sentences or images may not be aligned to anything — represented by a (potentially sparse) bipartite graph on $n_i$ sentence nodes and $m_i$ image nodes. During training, we are *given no access* to ground-truth image-sentence association graphs, i.e., we do not know *a priori* which images correspond to which sentences, only that all images/sentences in a document co-occur together; this is why we refer to our task as *unsupervised*.

We produce a dense sentence-to-image association matrix $\widehat{M}_i \in \mathbb{R}^{n_i \times m_i}$, in which each entry is the confidence that there is an (undirected) edge between the corresponding nodes. Applying different thresholding strategies to $\widehat{M}_i$'s values yields different alignment graphs.

**Evaluation**. When we have ground-truth alignment graphs for test documents, we evaluate the correctness of the association matrix $\widehat{M}_i$ predicted by our algorithms according to two metrics: AUROC (henceforth AUC) and precision-at-$C$ ($p@C$). AUC, commonly used in evaluating link prediction (see Menon and Elkan (2011)) is the area under the curve of the true-positive/false-positive rate produced by sweeping over possible confidence thresholds; random is 50, perfect is 100. $p@C$ measures the accuracy of the algorithm's most confident $C$ predicted edges (in our case, the most confident edges correspond to the largest entries in $\widehat{M}_i$). This metric models cases where only a small number of high-confidence predictions need be made per document. We evaluate using $C \in \{1, 5\}$.

## 3 Models

Our algorithm is inspired by work in cross-modal retrieval (Rasiwasia et al., 2010; Hodosh et al., 2013; Costa Pereira et al., 2014; Kiros et al., 2014b). Instead of operating at the level of individual images/sentences, however, our training objective encourages image *sets* and sentence *sets* appearing in the same document to be more similar than non-co-occurring sets.

### 3.1 Alignment Model and Loss Function

We assume that the dimensionality $d_{\text{multi}}$ of the multimodal text-image space is predetermined.

**Extracting sentence representations**. We pass the words in each sentence through a 300D word-embedding layer initialized with GoogleNews-pretrained word2vec embeddings (Mikolov et al., 2013). We then pass the sequence of word vectors to a GRU (Cho et al., 2014) and extract and L2-normalize a $d_{\text{multi}}$-dimensional sentence representation from the final hidden state.

[2] Data and code: www.cs.cornell.edu/~jhessel/multiretrieval/multiretrieval.html

[3] Sentences and images can be considered as sequences rather than sets in our framework, but unordered sets are more appropriate for modeling some of the crowd-sourced corpora we used in our experiments.

**Extracting image representations**. We first compute a representation for each image using a convolutional neural network (CNN).[4] The network's output is then mapped via affine projection to $\mathbb{R}^{d_{\text{multi}}}$ and L2-normalized.

**Correspondence prediction**. The result of running the two steps above on an image-set/text-set pair $\langle S, V \rangle$ is $|S| + |V|$ vectors, all in $\mathbb{R}^{d_{\text{multi}}}$. From these, we compute the similarity matrix $\widehat{M} \in \mathbb{R}^{|S| \times |V|}$, where the $(j,k)^{th}$ entry is the cosine similarity between the $j^{th}$ sentence vector and the $k^{th}$ image vector.

**Training Objective**. We train under the assumption that co-occurring image-set/sentence-set pairs should be more similar than non-co-occurring image-set/sentence-set pairs. We hope that use of this *document-level* objective will produce an $\widehat{M}_i$ offering reasonable *intra-document* information at test time, even though such information is not available at training time.

The training process is modulated by a similarity function $\text{sim}(S, V)$ that measures the similarity between a set of sentences and a set of images by examining the entries of the individual image/sentence similarity matrix $\widehat{M}_i$ (specific definitions of $\text{sim}(S, V)$ are proposed in §3.2). We use a max-margin loss with negative sampling: we iterate through true documents $d_i = \langle S_i, V_i \rangle$, and negatively sample at the document level a set of $b$ sets of images that did not co-occur with $S_i$, $\mathbb{V}' = \{V'_1, ..., V'_b\}$, and a set of $b$ sets of sentences that did not co-occur with $V_i$, $\mathbb{S}' = \{S'_1, ..., S'_b\}$.

We then compute a loss for $\langle S_i, V_i \rangle$ by comparing the true similarities to the negative-sample similarities. We find that hard-negative mining (Dalal and Triggs, 2005; Schroff et al., 2015; Faghri et al., 2018), the technique of selecting the negative cases that maximally violate the margin within the mini-batch, performs better than simple averaging. The loss for a single positive example is:

$$\mathcal{L}(S_i, V_i) = \max_{V' \in \mathbb{V}'} h\left(\text{sim}(S_i, V_i), \text{sim}(S_i, V')\right) + \max_{S' \in \mathbb{S}'} h\left(\text{sim}(S_i, V_i), \text{sim}(S', V_i)\right) \quad (1)$$

for hinge loss $h_\alpha(p, n) = \max(0, \alpha - p + n)$, where we set margin $\alpha = 0.2$ (Kiros et al., 2014a; Faghri et al., 2018).

[4] In some experiments, we use pre-computed image features from a pre-trained CNN (Sharif Razavian et al., 2014). In other cases, we fine-tune the full image network. We specify which representation we choose in a later section.

### 3.2 Similarity Functions

We explore several functions for measuring how similar a set of $n$ sentences $S$ is to a set of $m$ images $V$. All similarity functions convert the matrix $\widehat{M} \in \mathbb{R}^{n \times m}$ corresponding to $\langle S, V \rangle$ into a bipartite graph based on the magnitude of the entries. The functions differ in how they determine which entries $\widehat{M_{ij}}$ correspond to edges and edge weights.

**Dense Correspondence (DC)**. The DC function assumes a dense correspondence between images and sentences; each sentence must be aligned to its most similar image, and vice versa, regardless of how small the similarity might be:

$$\text{sim}(S, V) = \frac{1}{n} \sum_{i=0}^{n} \max_j \widehat{M}_{i,j} + \frac{1}{m} \sum_{j=0}^{m} \max_i \widehat{M}_{i,j}.$$

The underlying assumption of this function can clearly be violated in practice:[5] sentences can have no image, and images no sentence.

**Top-K (TK)**. Instead of assuming that every sentence has a corresponding image and vice versa, in this function only the top $k$ most likely sentence $\Rightarrow$ image (and image $\Rightarrow$ sentence) edges are aligned. This process mitigates the effect of non-visual sentences by allowing algorithms to align them to no image. We discuss choices of $k$ for particular experimental settings in §4.1.

**Assignment Problem (AP)**. We may wish to consider the image-sentence alignment task as a *bipartite linear assignment problem* (Kuhn, 1955), such that each image/sentence in a document has at most one association. Each time we compute $\text{sim}(S, V)$ in the forward pass of our models, we solve the integer programming problem of maximizing $\sum_{i,j} \widehat{M}_{ij} x_{ij}$ subject to the constraints:

$$\forall i, \sum_j x_{ij} \leq 1; \forall j, \sum_i x_{ij} \leq 1; \forall i, j, x_{ij} \in \{0, 1\}.$$

Despite involving a discrete optimization step, the model remains fully differentiable. Our forward pass uses tensorflow's python interface, `tf.py_func`, and the `lapjv` implementation of the JV algorithm (Jonker and Volgenant, 1987) to solve the integer program itself. Given the solution $x^*_{ij}$, we compute (and backpropagate gradients through) the similarity function $\text{sim}(S, V) = \left(\sum_{i,j} M_{ij} x^*_{ij}\right) / r$ where $r$ is the number of non-zero $x^*_{ij}$. Should we want to impose an upper bound

[5] Karpathy et al. (2014, §3.3.1) discuss violations in the image fragment/single-word case.




Figure 2: Sample documents from six of our datasets. Image sets and sentence sets may be truncated due to space constraints. The example from Story-DII is harder than is typical, but we include it to illustrate a point regarding image spread made in §4.1. *** denotes text-chunk delimiters present in the original data.

$k$ on the number of links, we can add the following additional constraint:[6] $\sum_{i,j} x_{ij} \leq k(S,V)$. For example, one could set $k(S,V) = \frac{1}{2}\min(|S|,|V|)$.

The JV algorithm's runtime is $O(\max(n,m)^3)$, and each positive example requires computing similarities for the positive case and the $2b$ negative samples from Eq. 1, for a per-example runtime of $O(b \cdot \max(n,m)^3)$. Fortunately, `lapjv` is highly optimized, so despite solving many integer programs, AP often runs *faster* than DC.

### 3.3 Baselines

We construct two baseline similarity functions, as we are not aware of existing models that directly address our task in an unsupervised fashion.

**Object Detection**. For each image in the document, we use DenseNet169 (Huang et al., 2017) to find its $K$ most probable ImageNet classes (e.g., "stingray"), and represent the image as the average of the word2vec embeddings of those $K$ labels. We represent each sentence in a document as the mean word2vec embedding of its words. To form the strongest possible baseline, we compute the cosine similarity between all sentence-image pairs to form $\widehat{M}$ for $K \in \{1...20\}$ and report the variant with the *best post-hoc performance on the test set*.

**NoStruct**. The similarity functions described in §3.2 rely on document-level, structural information, i.e., for a single image in a document, the *other images* in a document affect the overall similarity (and vice versa for sentences). However, this structural information may not be worth incorporating. Thus, we train a baseline that solely relies on single image/single sentence co-occurrence statistics. At training time, we randomly sample a single image and a single sentence from a document, compute the cosine similarity of their vector representations, and treat that value as the document similarity. While the randomly sampled image/sentence will not truly correspond for every sample, we still expect this baseline to produce above-random results when averaged over many iterations, as true correspondences have some (low) probability of being sampled.[7]

## 4 Experiments on Crowdlabeled Data

Our first set of experiments uses four pre-existing datasets created by asking crowdworkers to add sentence-long textual descriptions to images in a collection. Image-sentence alignments are therefore known by construction. We do not use these labels at training time: gold-standard alignments are only used at evaluation time to compare performance between algorithms.[8] Statistics of these datasets are given in the top half of Table 1, and example documents are given in Figure 2. Each crowdlabeled dataset is constructed to address a different question about our learning setting.

**Q: Is this task even possible? Test: MSCOCO**. MSCOCO (Lin et al., 2014) was created by crowdsourced manual captioning of single images. We construct "documents" from this data by first randomly aggregating five image-caption pairs. We then add five "distractor" images with no captions and five "distractor" captions with no images. Thus,

[6] Applying Volgenant's (2004) polynomial-time algorithm.

[7] This probability is equal to the density of the ground-truth, underlying image-sentence association graph.

[8] The supplementary material gives more details.

a non-distractor image truly corresponds to the single caption that was written about it, and not to the other 9 captions in the document. There are a total of 10 images/sentences per document, and 5 ground-truth image-sentence links. *A priori,* we expect this to be the easiest setting for within-document disambiguation because mismatched images and sentences are completely independent.

**Q: What if the images/sentences within a document are similar? Test: Story-DII**. Huang et al. (2016) asked crowdworkers to collect subsets of images contained in the same Flickr album (Thomee et al., 2016) that could be arranged into a visual story. In the Story-DII (= "descriptions in isolation") case, (possibly different) crowdworkers subsequently captioned the images, but only saw each image in isolation. We construct a set of documents from Story-DII so that each contains five images and five sentences. Because images come from the same album, images and captions in our Story-DII "documents" are more similar to each other than those in our MSCOCO "documents."

**Q: What if the sentences are cohesive and refer to each other? Test: Story-SIS**. Huang et al. (2016) also presented all the images in a subset from the same Flickr album to crowdworkers simultaneously and asked them to caption the image subsets collectively to form a story (SIS = "story in sequence"). In contrast to Story-DII, the generated sentences are generally not stand-alone descriptions of the corresponding image's contents, and may, for example, use pronouns to refer to elements from neighboring sentences and images.

**Q: What if there are many sentences with no corresponding images? Test: DII-Stress**. Because documents often have many sentences that do not directly refer to visual content, we constructed a setting with many more sentences than images. We augment documents from Story-DII with 45 randomly negatively sampled distractor captions. The resulting documents have five images and fifty sentences, where only five sentences truly describe images in the document.

**Experiment Protocols**. We conduct our evaluations over a single randomly sampled train/dev/test split. For image features, we extract the pre-classification layer of DenseNet169 (Huang et al., 2017) pretrained on the ImageNet (Russakovsky et al., 2015) classification task, unless otherwise specified. We train with Adam (Kingma and Ba, 2014) using a starting learning rate of .0001 for 50 epochs. We decrease the learning rate by a factor of 5 each time the loss in Eq. 1 over the dev set plateaus for more than 3 epochs. We set[9] $d_{\text{multi}} = 1024$, and apply dropout with $p = .4$. At test time, we use the model checkpoint with the lowest dev error.

| | train/val/test | $n_i/m_i$ (median) | # imgs (unique) | density |
|---|---|---|---|---|
| MSCOCO | 25K/2K/2K | 10/10 | 83K | 5% |
| Story-DII | 22K/3K/3K | 5/5 | 47K | 20% |
| Story-SIS | 37K/5K/5K | 5/5 | 76K | 20% |
| DII-Stress | 22K/3K/3K | 50/5 | 47K | 2% |
| DIY | 7K/1K/1K | 15/16 | 154K | 8% |
| RQA | 7K/1K/1K | 6/8 | 88K | 17% |
| WIKI | 14K/1K/1K | 86/5 | 92K | N/A |

Table 1: Dataset statistics: top half = crowdlabeled datasets; bottom half = organically-multimodal datasets. *Density* measures the sparsity of the ground truth graph as the number of ground-truth edges divided by the number of possible edges.

### 4.1 Crowdlabeled-Data Results

We tried all combinations of $b \in \{10, 20, 30\}$, $\text{sim} \in \{\text{DC}, \text{TK}, \text{AP}\}$. For TK and AP we set the maximum link threshold $k$ to $\min(S_i, V_i)$ or $\lceil \frac{1}{2} \min(S_i, V_i) \rceil$ (denoted $\frac{1}{2}k$ in the results table).[10]

Table 2 shows test-set prediction results for $b = 10$ (results for $b \in \{20, 30\}$ are similar). The retrieval-style objectives we consider encourage algorithms to learn useful within-document representations, and incorporating a structured similarity is beneficial. All our algorithms outperform the strongest baseline (NoStruct) in all cases, e.g., by at least 10 absolute percentage points in p@1 on Story-DII.

We next show, as a sanity check, that our inter-document training objective function (Eq. 1) corresponds to intra-document prediction performance (the actual function of interest). Figure 3 plots how both functions vary with number of epochs, for two different validation datasets. In general, inter-document performance and intra-document performance rise together during training;[11] for a fixed neural architecture, models better at optimiz-

[9] Anecdotally, we found that values of 256 and 512 produced similar performance in early testing.

[10] For datasets where $m_i = n_i$ and the first choice of definition for $k$ is used, $DC$ and $TK$ are the same. But running the duplicate algorithms anyway provides us with a rough sense of run-to-run variability.

[11] See the supplementary material for plots for all datasets; while the general pattern is the same, some of the training curves exhibit additional interesting patterns.

| | MSCOCO | | Story-DII | | Story-SIS | | DII-Stress | |
|---|---|---|---|---|---|---|---|---|
| | AUC | p@1/p@5 | AUC | p@1/p@5 | AUC | p@1/p@5 | AUC | p@1/p@5 |
| Random | 49.7 | 5.0/4.6 | 49.4 | 19.5/19.2 | 50.0 | 19.4/19.7 | 50.0 | 2.0/2.0 |
| Obj Detect | 89.5 | 67.7/45.9 | 65.3 | 50.2/35.2 | 58.4 | 40.8/28.6 | 76.9 | 25.7/17.5 |
| NoStruct | 87.5 | 50.6/34.6 | 76.6 | 60.1/46.2 | 64.9 | 43.2/33.7 | 84.2 | 21.4/15.6 |
| DC | **98.9** | 93.6/**80.1** | **82.8** | 71.5/**55.5** | **68.8** | **51.8/38.6** | **94.9** | **64.6/44.8** |
| TK | **98.9** | 93.9/**80.1** | **82.9** | 71.4/**55.5** | **68.8** | 50.9/**38.7** | **95.2** | **65.6/45.3** |
| ↳ + $\frac{1}{2}k$ | **99.0** | **95.0/81.1** | **82.0** | **72.6/54.9** | 67.6 | **51.9/38.0** | **94.7** | 64.0/43.7 |
| AP | **98.7** | 91.0/78.0 | **82.6** | 70.5/**55.0** | **68.5** | 50.5/**38.3** | **95.3** | **65.5/45.7** |
| ↳ + $\frac{1}{2}k$ | **98.9** | 93.9/**80.4** | 81.6 | **72.4**/54.4 | 67.4 | **52.1**/37.7 | **94.5** | **65.0**/43.4 |

Table 2: Results for crowdlabeled datasets (similar results for other settings are included in the supplementary material). Values are bolded if they are within 1% of the best-in-column performance.

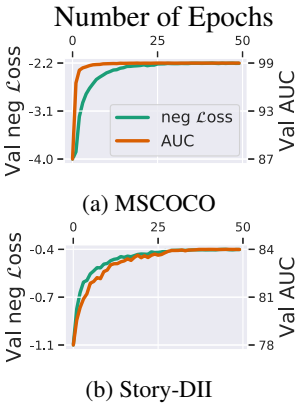


(a) MSCOCO

(b) Story-DII

Figure 3: Inter-document objective (AP, $b = 10$) and intra-document AUC increase together during training.

ing the inter-document loss in Eq. 1 also generally produce better intra-document representations.

In addition, we found that i) DC, despite assuming every sentence corresponds to an image, achieves high performance on DII-Stress, even though 90% of its sentences do not correspond to an image; ii) Allowing AP/TK to make fewer connections (i.e., setting $\frac{1}{2}k$) did not result in significant performance changes, even in the MSCOCO case, where the true number of links (5) was the same as the number of links accounted for by AP/TK+$\frac{1}{2}k$; and iii) adding topical cohesion (MSCOCO $\rightarrow$ Story-DII) makes the task more difficult, as does adding textual cohesion (Story-DII $\rightarrow$ Story-SIS).

**Models have trouble with the same documents.** We calculated AUC for each test document individually. The Spearman correlation between these individual-instance AUC values is very high: of all pairs in DC/TK/AP, over all crowdlabeled datasets at b=10, DC vs. AP on MSCOCO had the lowest correlation with $\rho = .89$.

**Error analysis: content vs. spread.** Why are some instances more difficult to solve for all of our algorithms? We consider two hypotheses. The "content" hypothesis is that some concepts are more difficult for algorithms to find multimodal relationships between: "beauty" may be hard to visualize, whereas "dog" is a concrete concept (Lu et al., 2008; Berg et al., 2010; Parikh and Grauman, 2011; Hessel et al., 2018; Mahajan et al., 2018). The "spread" hypothesis, which we introduce, is that documents with lower diversity among images/sentences may be harder to disambiguate at test time. For example, a document in which all images and all sentences are about horses requires finer-grained distinctions than a document with a horse, a barn, and a tractor. The Story-DII vs. Story-SIS example in Fig. 2 illustrates this contrast.

To quantify the spread of a document, we first extract vector representations of each test image/sentence.[12] We then L2-normalize the vectors and compute the mean squared distance to their centroid; higher "spread" values indicate that a document's sentences/images are more diverse. To quantify the content of a document, for simplicity, we mean-pool the image/sentence representations and reduce to 20 dimensions with PCA.

We first compute an OLS regression of image spread + text spread on test AUC scores for Story-DII/Story-SIS/DII-Stress[13] for AP with $b = 10$: 42/23/16% respectively (F-test $p \ll .01$) of the variance in AUC can be explained by the spread hypothesis alone. In general, documents with less diverse content are harder, with image spread explaining more variance than text spread. When adding in the image+text content features, the proportion of AUC variance explained increases to 52/35/38%; thus, for these datasets, both the "content" and "spread" hypotheses independently explain document difficulty, though the relative importance of each varies across datasets.

## 5 Experiments on RQA and DIY

The previous datasets had captions added by crowdworkers for the explicit purpose of aiding research on grounding: for MSCOCO, annotators providing

[12] We use DenseNet169 features for images and mean word2vec for sentences. We don't use internal model representations as we aim to quantify aspects of the dataset itself.

[13] MSCOCO is omitted because the AUC scores are all large.

| | RQA | | DIY | |
|---|---|---|---|---|
| | AUC | p@1/p@5 | AUC | p@1/p@5 |
| Random | 49.4 | 17.8/16.7 | 49.8 | 6.3/6.8 |
| Obj Detect | 58.7 | 25.1/21.5 | 53.4 | 17.9/11.8 |
| NoStruct | 60.5 | 33.8/27.0 | 57.0 | 13.3/11.8 |
| DC | 63.5 | 38.3/30.6 | 59.3 | 20.8/16.1 |
| TK | 67.9 | 44.0/35.8 | 60.5 | 21.2/16.0 |
| ↳ + $\frac{1}{2}k$ | 68.1 | 44.5/35.4 | 56.0 | 14.1/12.5 |
| AP | **69.3** | **47.3/37.3** | **61.8** | **22.5/17.2** |
| ↳ + $\frac{1}{2}k$ | **68.7** | **47.2**/36.2 | 59.4 | **21.6**/15.3 |

Table 3: Performance on the organically-multimodal data; values within 1% of best-in-column are bolded.

image captions were explicitly instructed to provide literal descriptions and "not describe what a person might say" (Chen et al., 2015). The manner in which users interact with multimodal content "in the wild" significantly differs from crowdlabeled data: Marsh and Domas White's (2003) 49-element taxonomy of multimodal relationships (e.g., "decorate", "reiterate", "humanize") observed in 45 web documents highlights the diversity of possible image-text relationships.

We thus consider two datasets (one of which we release ourselves) of organically-multimodal documents scraped from web data, where the original authors created or selected both images and sentences. Statistics of these datasets are given in the bottom half of Table 1.

**RQA**. RecipeQA (Yagcioglu et al., 2018) is a question-answering dataset scraped from `instructibles.com` consisting of images/descriptions of food preparation steps; we construct documents by treating each recipe step as a sentence.[14] Users of the Instructibles web interface put images and recipe steps in direct correspondence, which gives us a graph for test time evaluation.

**DIY (new)**. We downloaded a sample of 9K Reddit posts made to the community DIY ("do it yourself"). These posts [15] consist of multiple images that users have taken of the progression of their construction projects, e.g., building a rock climbing wall (see Figure 2). Users are encouraged to explicitly annotate individual images with captions,[16] and, for evaluation, we treat a caption written alongside a given image as corresponding to a true link.

We adopt the same experimental protocols as in §4, but increase the maximum sentence token-length from 20 to 50; Table 3 shows the test-set results. In general, the algorithms we introduce again outperform the NoStruct baseline. In contrast to the crowdlabeled experiments, AP (slightly) outperformed the other algorithms.[17] DIY is the most difficult among the datasets we consider.

To see if the algorithms err on the same instances, we again compute the Spearman correlation $\rho$ between test-instance AUC scores for DC/TK/AP, for $b = 10$. We find greater variation in performance on organically-multimodal compared to crowdlabeled data. For example, on RQA, DC and AP have a $\rho$ of only .64. We also repeat the regression on test-instance AUC scores introduced in §4.1 with different results; content generally explains more variance than spread, e.g., for AP, for RQA/DIY respectively, only 2/1% is explained by spread alone, but 18/13% is explained by spread+content.

## 6 Qualitative Exploration

To visualize the within-document prediction for document $i$, we compute $\widehat{M}_i$ and solve the linear assignment problem described in §3.2, taking the edges with highest selected weights to be the most confident. Figure 4 contains example test predictions (along with $\widehat{M}_i$) from the datasets with ground-truth annotation. In an effort to provide representative cases, the selected examples have AUC scores close to average performance for their corresponding datasets.

The model mostly succeeds at associating literal objects and their descriptions: tennis players in MSCOCO, castles in Story-DII, a stapler in DIY, and bacon in a blender in RQA. Errors are often justifiable. For example, for the MSCOCO document, the chosen caption for a picture of two people playing baseball accurately describes the image, despite it having been written for a different image and thus counting as an error in our quantitative evaluation. Similarly, for RQA, a container of maple syrup is associated with a caption mentioning "syrup", which seems reasonable even though the recipe's author did not link that image/sentence.

In other cases, the algorithm struggles with what part of the image to "pay attention" to. In the Story-DII case (Figure 4b), the algorithm erroneously

[14] Recipe steps have variable length, are often not strictly grammatical sentences, and can contain lists, linebreaks, etc.

[15] We required at least 25 upvotes per Reddit post to filter out spam and low-quality submissions.

[16] As with RQA, DIY captions are not always grammatical.

[17] This holds even when varying the number of negatively sampled documents; see the supplementary material.

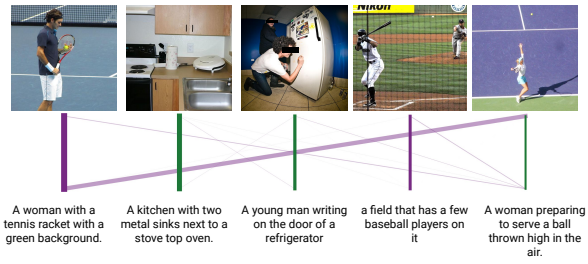


(a) MSCOCO; 97 AUC, 10 sentences/10 images.

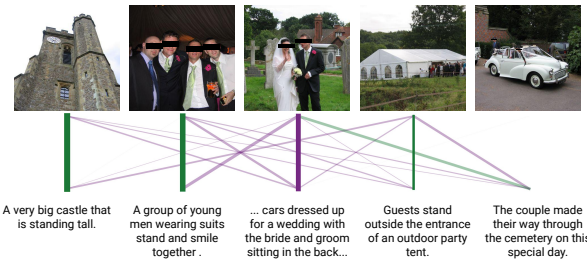


(b) Story-DII; 83 AUC, 5 sentences/5 images.

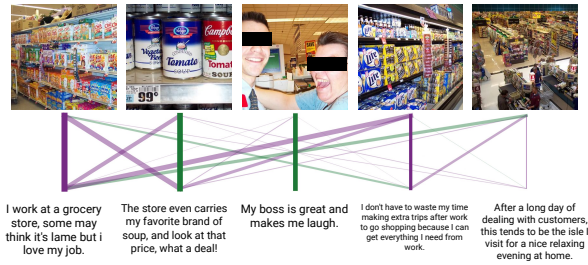


(c) Story-SIS; 70 AUC, 5 sentences/5 images.

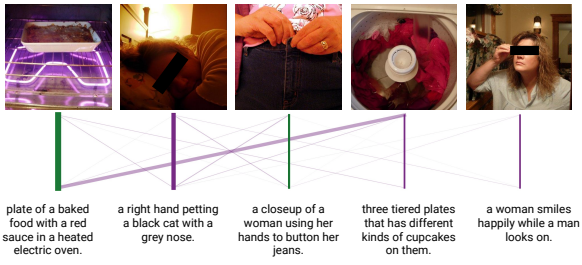


(d) DII-Stress; 94 AUC, 50 sentences/5 images.

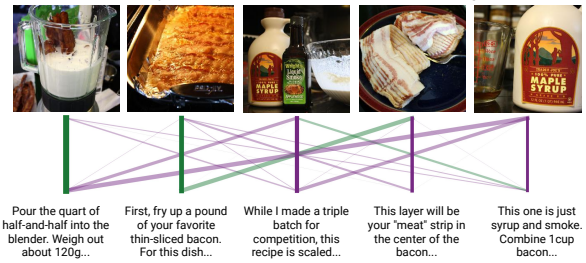


(e) RQA; 70 AUC, 9 sentences/18 images.

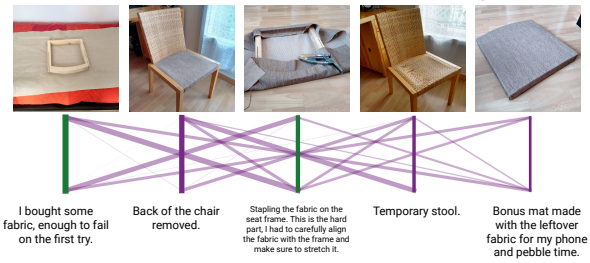


(f) DIY; 62 AUC, 17 sentences/17 images.

Figure 4: Example test-time graph predictions from AP with $b = 10$. Each subfigure gives the top 5 image/sentence predictions per document, in decreasing order of confidence from left to right. Green edges indicate ground-truth pairs; edge widths show the magnitude of edges in $\widehat{M}_i$ (only positive weights are shown). Examples are selected to be representative: per-document AUC (roughly) matches the average AUC achieved on the corresponding dataset.

(but arguably justifiably) decides to assign a caption about a bride, groom, and a car to a picture of the couple, instead of to a picture of a vehicle.

For more difficult datasets like Story-SIS (Figure 4c), the algorithm struggles with ambiguity. For 2/5 sentences that refer to literal objects/actions (soup cans/laughter), the algorithm works well. The remaining 3 captions are general musings about working at a grocery store that could be matched to any of the three remaining images depicting grocery store aisles. DIY is similarly difficult, as many images/sentences could reasonably be assigned to each other.

**WIKI**. We also constructed a dataset from English sentence-tokenized Wikipedia articles (not including captions) and their associated images from ImageCLEF2010 (Popescu et al., 2010). In contrast to RQA and DIY, there are no explicit connections between individual images and individual sentences, so we cannot compute AUC or precision, but this corpus represents an important organically-multimodal setting. We follow the same experimental settings as in §4 at training time, but instead of using pre-extracted features, we fine-tune the vision model's parameters.[18] Examining the predictions of the AP+fine-tuned CNN model trained on WIKI shows many of the model's predictions to be reasonable. Figure 5 shows the model's 5 most confident predictions on the 100-sentence Wikipedia article about Mauritius, chosen for its high image/text spread.

## 7 Additional related work

Our similarity functions are inspired by work in aligning image fragments, such as object bounding boxes, with portions of sentences without ex-

[18] In comparable settings, fine-tuning the vision CNN yields $\approx 20\%$ better performance in terms of the loss in Equation 1 computed over the validation/test sets. For memory reasons, we switched from DenseNet169 to NASNetSmall (Zoph et al., 2018); additional details are in the supplementary material.




Figure 5: Predicted sentences, with cosine similarities, for images in a 100-sentence ImageCLEF Wikipedia article on Mauritius. The first three predictions are reasonable, the last two are not. The third result is particularly good given that only two sentences mention dodos; for comparison, the object-detection's choice began "(Mauritian Creole people usually known as 'Creoles')".

plicit labels (Karpathy et al., 2014; Karpathy and Fei-Fei, 2015; Jiang et al., 2015; Rohrbach et al., 2016; Datta et al., 2019); similar tasks have been addressed in supervised (Plummer et al., 2015) and semi-supervised (Rohrbach et al., 2016) settings. Our models operate at the larger granularity of entire images/sentences. Integer programs like AP have been used to align visual and textual content in videos, e.g., Bojanowski et al. (2015)

Prior work has addressed the task of identifying objects in single images that are referred to by natural language descriptions (Mitchell et al., 2010, 2013; Kazemzadeh et al., 2014; Karpathy et al., 2014; Plummer et al., 2015; Hu et al., 2016b; Rohrbach et al., 2016; Nagaraja et al., 2016; Hu et al., 2016a; Yu et al., 2016; Peyre et al., 2017; Margffoy-Tuay et al., 2018). In general, a supervised approach is taken (Mao et al., 2016; Krishna et al., 2017; Johnson et al., 2017).

Related tasks involving multi-image/multi-sentence data include: generating captions/stories for image streams or videos (Park and Kim, 2015; Huang et al., 2016; Shin et al., 2016; Liu et al., 2017), sorting aligned (image, caption) pairs into stories (Agrawal et al., 2016), image/textual cloze tasks (Iyyer et al., 2017; Yagcioglu et al., 2018), augmentation of Wikipedia articles with 3D models (Russell et al., 2013), question-answering (Kembhavi et al., 2017), and aligning books with their film adaptations (Zhu et al., 2015); these tasks are usually supervised, or rely on a search engine.

## 8 Conclusion and Future Directions

We have demonstrated that a family of models for learning fine-grained image-sentence links *within documents* can produce good test-time results even if only given access to document-level co-occurrence at training time. Future work could incorporate better models of sequence within document context (Kim et al., 2015; Alikhani and Stone, 2018). While using structured loss functions improved performance, image and sentence *representations* themselves have no awareness of neighboring images/sentences; this information should prove useful if modeled appropriately.[19]

**Acknowledgments**. We thank Yoav Artzi, Cristian Danescu-Niculescu-Mizil, Jon Kleinberg, Vlad Niculae, Justine Zhang, the Cornell NLP seminar, the reviewers, and Ondrej Linda, Randy Puttick, Ramin Mehran, and Grant Long of Zillow for helpful comments. We additionally thank the NVidia Corporation for the GPUs used in this study. This material is supported by the U.S. National Science Foundation under grants BIGDATA SES-1741441, 1526155, 1652536, and the Alfred P. Sloan Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

[19] Attempts to incorporate document context information by passing the word-level RNN's output through a sentence-level RNN (Li et al., 2015; Yang et al., 2016) did not improve performance.

## References


Harsh Agrawal, Arjun Chandrasekaran, Dhruv Batra, Devi Parikh, and Mohit Bansal. 2016. Sort story: Sorting jumbled images and captions into stories. In *EMNLP*.

Malihe Alikhani and Matthew Stone. 2018. Exploring coherence in visual explanations. In *Multimedia Information Processing and Retrieval*.

Tamara L Berg, Alexander C Berg, and Jonathan Shih. 2010. Automatic attribute discovery and characterization from noisy web data. In *ECCV*.

Piotr Bojanowski, Rémi Lajugie, Edouard Grave, Francis Bach, Ivan Laptev, Jean Ponce, and Cordelia Schmid. 2015. Weakly-supervised alignment of video with text. In *ICCV*.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C. Lawrence Zitnick. 2015. Microsoft COCO captions: Data collection and evaluation server. *Computing Research Repository*, arXiv:1504.00325. Version 2.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *EMNLP*.

Jose Costa Pereira, Emanuele Coviello, Gabriel Doyle, Nikhil Rasiwasia, Gert R. G. Lanckriet, Roger Levy, and Nuno Vasconcelos. 2014. On the role of correlation and abstraction in cross-modal multimedia retrieval. *TPAMI*.

Navneet Dalal and Bill Triggs. 2005. Histograms of oriented gradients for human detection. In *CVPR*.

Samyak Datta, Karan Sikka, Anirban Roy, Karuna Ahuja, Devi Parikh, and Ajay Divakaran. 2019. Align2ground: Weakly supervised phrase grounding guided by image-caption alignment. In *ICCV*.

Fartash Faghri, David J Fleet, Jamie Ryan Kiros, and Sanja Fidler. 2018. VSE++: Improving visual-semantic embeddings with hard negatives. In *British Machine Vision Conference*.

Jack Hessel, David Mimno, and Lillian Lee. 2018. Quantifying the visual concreteness of words and topics in multimodal datasets. In *NAACL*.

Micah Hodosh, Peter Young, and Julia Hockenmaier. 2013. Framing image description as a ranking task: Data, models and evaluation metrics. *Journal of Artificial Intelligence Research*.

Ronghang Hu, Marcus Rohrbach, and Trevor Darrell. 2016a. Segmentation from natural language expressions. In *ECCV*.

Ronghang Hu, Huazhe Xu, Marcus Rohrbach, Jiashi Feng, Kate Saenko, and Trevor Darrell. 2016b. Natural language object retrieval. In *CVPR*.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. 2017. Densely connected convolutional networks. In *CVPR*.

Ting-Hao (Kenneth) Huang, Francis Ferraro, Nasrin Mostafazadeh, Ishan Misra, Aishwarya Agrawal, Jacob Devlin, Ross Girshick, Xiaodong He, Pushmeet Kohli, Dhruv Batra, C. Lawrence Zitnick, Devi Parikh, Lucy Vanderwende, Michel Galley, and Margaret Mitchell. 2016. Visual storytelling. In *NAACL*.

Mohit Iyyer, Varun Manjunatha, Anupam Guha, Yogarshi Vyas, Jordan Boyd-Graber, Hal Daumé III, and Larry S. Davis. 2017. The amazing mysteries of the gutter: Drawing inferences between panels in comic book narratives. In *CVPR*.

Xinyang Jiang, Fei Wu, Xi Li, Zhou Zhao, Weiming Lu, Siliang Tang, and Yueting Zhuang. 2015. Deep compositional cross-modal learning to rank via local-global alignment. In *ACM MM*.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross Girshick. 2017. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In *CVPR*.

Roy Jonker and Anton Volgenant. 1987. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *CVPR*.

Andrej Karpathy, Armand Joulin, and Li F Fei-Fei. 2014. Deep fragment embeddings for bidirectional image sentence mapping. In *NeurIPS*.

Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. 2014. ReferItGame: Referring to objects in photographs of natural scenes. In *EMNLP*.

Aniruddha Kembhavi, Minjoon Seo, Dustin Schwenk, Jonghyun Choi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Are you smarter than a sixth grader? Textbook question answering for multimodal machine comprehension. In *CVPR*.

Gunhee Kim, Seungwhan Moon, and Leonid Sigal. 2015. Ranking and retrieval of image sequences from multiple paragraph queries. In *CVPR*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *Computing Research Repository*, arXiv:1412.6980. Version 5.

Ryan Kiros, Ruslan Salakhutdinov, and Rich Zemel. 2014a. Multimodal neural language models. In *ICML*.

Ryan Kiros, Ruslan Salakhutdinov, and Richard S. Zemel. 2014b. Unifying visual-semantic embeddings with multimodal neural language models. In *NeurIPS Deep Leaning Workshop*.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David Ayman Shamma, Michael Bernstein, and Li Fei-Fei. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*.

Harold W. Kuhn. 1955. The Hungarian method for the assignment problem. *Naval research logistics quarterly*.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. In *ACL*.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: Common objects in context. In *ECCV*.

Yu Liu, Jianlong Fu, Tao Mei, and Chang Wen Chen. 2017. Let your photos talk: Generating narrative paragraph for photo stream via bidirectional attention recurrent neural networks. In *AAAI*.

Yijuan Lu, Lei Zhang, Qi Tian, and Wei-Ying Ma. 2008. What are the high-level concepts with small semantic gaps? In *CVPR*.

Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. 2018. Exploring the limits of weakly supervised pretraining. In *ECCV*.

Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L Yuille, and Kevin Murphy. 2016. Generation and comprehension of unambiguous object descriptions. In *CVPR*.

Edgar Margffoy-Tuay, Juan C Pérez, Emilio Botero, and Pablo Arbeláez. 2018. Dynamic multimodal instance segmentation guided by natural language queries. In *ECCV*.

Emily E. Marsh and Marilyn Domas White. 2003. A taxonomy of relationships between images and text. *Journal of Documentation*.

Aditya Krishna Menon and Charles Elkan. 2011. Link prediction via matrix factorization. In *ECML PKDD*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NeurIPS*.

Margaret Mitchell, Kees van Deemter, and Ehud Reiter. 2010. Natural reference to objects in a visual domain. In *International Natural Language Generation Conference*.

Margaret Mitchell, Ehud Reiter, and Kees Van Deemter. 2013. Typicality and object reference. In *Proceedings of the Annual Meeting of the Cognitive Science Society*.

Varun K. Nagaraja, Vlad I. Morariu, and Larry S. Davis. 2016. Modeling context between objects for referring expression understanding. In *ECCV*.

Devi Parikh and Kristen Grauman. 2011. Interactively building a discriminative vocabulary of nameable attributes. In *CVPR*.

Cesc C. Park and Gunhee Kim. 2015. Expressing an image stream with a sequence of natural sentences. In *NeurIPS*.

Julia Peyre, Josef Sivic, Ivan Laptev, and Cordelia Schmid. 2017. Weakly-supervised learning of visual relations. In *ICCV*.

Bryan A. Plummer, Liwei Wang, Chris M. Cervantes, Juan C. Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. 2015. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *ICCV*.

Adrian Popescu, Theodora Tsikrika, and Jana Kludas. 2010. Overview of the Wikipedia retrieval task at ImageCLEF 2010. In *CLEF*.

Nikhil Rasiwasia, Jose Costa Pereira, Emanuele Coviello, Gabriel Doyle, Gert R. G. Lanckriet, Roger Levy, and Nuno Vasconcelos. 2010. A new approach to cross-modal multimedia retrieval. In *ACM MM*.

Anna Rohrbach, Marcus Rohrbach, Ronghang Hu, Trevor Darrell, and Bernt Schiele. 2016. Grounding of textual phrases in images by reconstruction. In *ECCV*.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*.

Bryan C. Russell, Ricardo Martin-Brualla, Daniel J. Butler, Steven M. Seitz, and Luke Zettlemoyer. 2013. 3D Wikipedia: Using online text to automatically label and navigate reconstructed geometry. *ACM Transactions on Graphics*.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *CVPR*.

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. 2014. CNN features off-the-shelf: an astounding baseline for recognition. In *CVPR*.

Andrew Shin, Katsunori Ohnishi, and Tatsuya Harada. 2016. Beyond caption to narrative: Video captioning with multiple sentences. In *International Conference on Image Processing (ICIP)*.

Bart Thomee, David A. Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. 2016. YFCC100M: the new data in multimedia research. *Communications of the ACM*.

Anton Volgenant. 2004. Solving the k-cardinality assignment problem by transformation. *European Journal of Operational Research*.

Shaomei Wu, Jeffrey Wieland, Omid Farivar, and Julie Schiller. 2017. Automatic alt-text: Computer-generated image descriptions for blind users on a social network service. In *CSCW*.

Semih Yagcioglu, Aykut Erdem, Erkut Erdem, and Nazli Ikizler-Cinbis. 2018. RecipeQA: A challenge dataset for multimodal comprehension of cooking recipes. In *EMNLP*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *NAACL*.

Licheng Yu, Patrick Poirson, Shan Yang, Alexander C. Berg, and Tamara L. Berg. 2016. Modeling context in referring expressions. In *ECCV*.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *ICCV*.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. 2018. Learning transferable architectures for scalable image recognition. In *CVPR*.

# UR-FUNNY: A Multimodal Language Dataset for Understanding Humor

**Md Kamrul Hasan[1*], Wasifur Rahman[1*], Amir Zadeh[2], Jianyuan Zhong[1], Md Iftekhar Tanveer[1 †], Louis-Philippe Morency[2], Mohammed (Ehsan) Hoque[1]**
1 - Department of Computer Science, University of Rochester, USA
2 - Language Technologies Institute, SCS, CMU, USA
{mhasan8,echowdh2}@cs.rochester.edu, abagherz@cs.cmu.edu,
jzhong9@u.rochester.edu, itanveer@cs.rochester.edu,
morency@cs.cmu.edu, mehoque@cs.rochester.edu

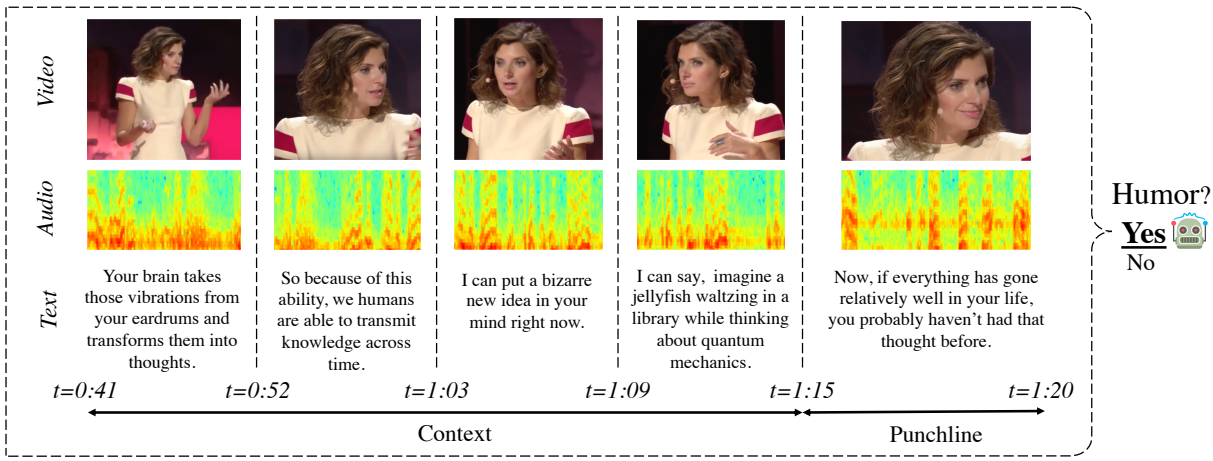


Figure 1: An example of the UR-FUNNY dataset. UR-FUNNY presents a framework to study the dynamics of humor in multimodal language. Machine learning models are given a sequence of sentences with the accompanying modalities of visual and acoustic. Their goal is to detect whether or not the sequence will trigger immediate laughter after the punchline.

## Abstract

Humor is a unique and creative communicative behavior often displayed during social interactions. It is produced in a multimodal manner, through the usage of words (text), gestures (visual) and prosodic cues (acoustic). Understanding humor from these three modalities falls within boundaries of multimodal language; a recent research trend in natural language processing that models natural language as it happens in face-to-face communication. Although humor detection is an established research area in NLP, in a multimodal context it has been understudied. This paper presents a diverse multimodal dataset, called UR-FUNNY, to open the door to understanding multimodal language used in expressing humor. The dataset and accompanying studies, present a framework in multimodal humor detection for the natural language processing community. UR-FUNNY is publicly available for research.

## 1 Introduction

Humor is a unique communication skill that removes barriers in conversations. Research shows that effective use of humor allows a speaker to establish rapport (Stauffer, 1999), grab attention (Wanzer et al., 2010), introduce a difficult concept without confusing the audience (Garner, 2005) and even to build trust (Vartabedian and Vartabedian, 1993). During face-to-face communications, humor involves multimodal communicative channels including effective use of words (text), accompanying gestures (visual) and prosodic cues (acoustic). Being able to mix and align those modalities appropriately is often unique to individuals, attributing to many different styles. Styles include gradually building up to a punchline using words, gestures and prosodic cues, a sudden twist to the story with an unex-

* - Equal contribution ; †- Currently affiliated with Comcast Applied AI Research, Washington DC, USA.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2046–2056,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

pected punchline (Ramachandran, 1998), creating a discrepancy between modalities (e.g., something funny being said without any emotion), or just laughing with the speech to stimulate the audience to mirror the laughter (Provine, 1992).

Modeling humor using a computational framework is inherently challenging due to factors such as: 1) *Idiosyncrasy*: often humorous people are also the most creative ones (Hauck and Thomas, 1972). This creativity in turn adds to the dynamic complexity of how humor is expressed in a multimodal manner. Use of words, gestures, prosodic cues and their (mis)alignments are choices that a creative user often experiments with. 2) *Contextual Dependencies*: humor often develops through time as speakers plan for a punchline in advance. There is a gradual build up in the story with a sudden twist using a punchline (Ramachandran, 1998). Some punchlines when viewed in isolation (as illustrated in Figure 1) may not appear funny. The humor stems from the prior build up, cross-referencing multiple sources, and its delivery. Therefore, a full understanding of humor requires analyzing the context of the punchline.

Understanding the unique dependencies across modalities and its impact on humor require knowledge from multimodal language; a recent research trend in the field of natural language processing (Zadeh et al., 2018b). Studies in this area aim to explain natural language from three modalities of text, visual and acoustic. In this paper, alongside computational descriptors for text, gestures such as smile or vocal properties such as loudness are measured and put together in a multimodal framework to define humor recognition as a multimodal task.

The main contribution of this paper to the NLP community is introducing the first multimodal language (including text, visual and acoustic modalities) dataset of humor detection named "UR-FUNNY". This dataset opens the door to understanding and modeling humor in a multimodal framework. The studies in this paper present performance baselines for this task and demonstrate the impact of using all three modalities together for humor modeling.

## 2 Background

The dataset and experiments in this paper are connected to the following areas:

**Humor Analysis:** Humor analysis has been among active areas of research in both natural language processing and affective computing. Notable datasets in this area include "16000 One-Liners" (Mihalcea and Strapparava, 2005), "Pun of the Day" (Yang et al., 2015), "PTT Jokes" (Chen and Soo, 2018), "Ted Laughter" (Chen and Lee, 2017), and "Big Bang Theory" (Bertero et al., 2016). The above datasets have studied humor from different perspectives. For example, "16000 One-Liner" and "Pun of the Day" focus on joke detection (binary task), while "Ted Laughter" focuses on punchline detection (whether or not punchline triggers laughter). Similar to "Ted Laughter", UR-FUNNY focuses on punchline detection. Furthermore, punchline is accompanied by context sentences to properly model the build up of humor. Unlike previous datasets where negative samples were drawn from a different domain, UR-FUNNY uses a challenging negative sampling case where samples are drawn from the same videos. Furthermore, UR-FUNNY is the only humor detection dataset which incorporates all three modalities of text, visual and acoustic. Table 1 shows a comparison between previously proposed datasets and UR-FUNNY dataset.

| Dataset | #Pos | #Neg | Mod | type | #spk |
|---|---|---|---|---|---|
| 16000 One-Liners | 16000 | 16000 | {*t*} | joke | - |
| Pun of the Day | 2423 | 2423 | {*t*} | pun | - |
| PTT Jokes | 1425 | 2551 | {*t*} | political | - |
| Ted Laughter | 4726 | 4726 | {*t*} | speech | 1192 |
| Big Bang Theory | 18691 | 24981 | {*t,a*} | tv show | <50 |
| **UR-FUNNY** | 8257 | 8257 | {*t,a,v*} | speech | 1741 |

Table 1: Comparison between UR-FUNNY and notable humor detection datasets in the NLP community. Here, '#','pos', 'neg' , 'mod' and 'spk' denote number, positive, negative, modalities and speaker respectively.

From modeling aspect, humor detection is done using hand-crafted and non-neural models (Yang et al., 2015), neural based RNN and CNN models on Yelp (de Oliveira et al., 2017) and TED talks data (Chen and Lee, 2017). Newer approaches have used (Chen and Soo, 2018) highway networks on "16000 One-Liner" and "Pun of the Day" datasets. There have been very few attempts at using extra modalities alongside language for detecting humor, mostly limited to adding simple audio features (Rakov and Rosenberg, 2013; Bertero et al., 2016) or the description of cartoon images (Shahaf et al., 2015). Furthermore, these attempts have been restricted to certain topics and domains (such as "Big Bang Theory" TV show

(Bertero et al., 2016)).

**Multimodal Language Analysis:** Studying natural language from modalities of text, visual and acoustic is a recent research trend in natural language processing (Zadeh et al., 2018b). Notable works in this area present novel multimodal neural architectures (Wang et al., 2019; Pham et al., 2019; Hazarika et al., 2018; Poria et al., 2017; Zadeh et al., 2017), multimodal fusion approaches (Liang et al., 2018; Tsai et al., 2018; Liu et al., 2018; Zadeh et al., 2018a; Barezi et al., 2018) as well as resources (Poria et al., 2018a; Zadeh et al., 2018c, 2016; Park et al., 2014; Rosas et al., 2013; Wöllmer et al., 2013). Multimodal language datasets mostly target multimodal sentiment analysis (Poria et al., 2018b), emotion recognition (Zadeh et al., 2018c; Busso et al., 2008), and personality traits recognition (Park et al., 2014). UR-FUNNY dataset is similar to the above datasets in diversity (speakers and topics) and size, with the main task of humor detection. Beyond the scope of multimodal language analysis, the dataset and studies in this paper have similarities to other applications in multimodal machine learning such language and vision studies, robotics, image captioning, and media description (Baltrušaitis et al., 2019).

## 3 UR-FUNNY Dataset

In this section we present the UR-FUNNY dataset [1]. We first discuss the data acquisition process, and subsequently present statistics of the dataset as well as multimodal feature extraction and validation.

### 3.1 Data Acquisition

A suitable dataset for the task of multimodal humor detection should be diverse in a) *speakers*: modeling the idiosyncratic expressions of humor may require a dataset with large number of speakers, and b) *topics*: different topics exhibit different styles of humor as the context and punchline can be entirely different from one topic to another.

TED talks [2] are among the most diverse idea sharing channels, in both speakers and topics. Speakers from various backgrounds, ethnic groups and cultures present their thoughts through a widely popular channel [3]. The topics of these presentations are diverse; from scientific discoveries to everyday ordinary events. As a result of diversity in speakers and topics, TED talks span across a broad spectrum of humor. Therefore, this platform presents a unique resource for studying the dynamics of humor in a multimodal setup.

TED videos include manual transcripts and audience markers. Transcriptions are highly reliable, which in turn allow for aligning the text and audio. This property makes TED talks a unique resource for newest continuous fusion trends (Chen et al., 2017). Transcriptions also include reliably annotated markers for audience behavior. Specifically, the "laughter" marker has been used in NLP studies as an indicator of humor (Chen and Lee, 2017). Previous studies have identified the importance of both punchline and context in understanding and modeling the humor. In a humorous scenario, context is the gradual build up of a story and punchline is a sudden twist to the story which causes laughter (Ramachandran, 1998). Using the provided laughter marker, the sentence immediately before the marker is considered as the punchline and the sentences prior to punchline (but after previous laughter marker) are considered context.

We collect 1866 videos as well as their transcripts in English from TED portal. These 1866 videos are chosen from 1741 different speakers and across 417 topics. The laughter markup is used to filter out 8257 humorous punchlines from the transcripts (Chen and Lee, 2017). The context is extracted from the prior sentences to the punchline (until the previous humor instances or the beginning of video is reached). Using a similar approach, 8257 negative samples are chosen at random intervals where the last sentence is not immediately followed by a laughter marker. The last sentence is assumed a punchline and similar to the positive instances, the context is chosen. This negative sampling uses sentences from the same distribution, as opposed to datasets which use sentences from other distributions or domains as negative sample (Yang et al., 2015; Mihalcea and Strapparava, 2005). After this negative sampling, there is a homogeneous 50% split in the dataset between positive and negative examples.

Using forced alignment, we mark the beginning and end of each sentence in the video as well as

[1] UR-FUNNY Dataset: https://github.com/ROC-HCI/UR-FUNNY

[2] Videos on www.ted.com are publicly available for download.

[3] More than 12 million subscribers on YouTube https://www.youtube.com/user/TEDtalksDirector

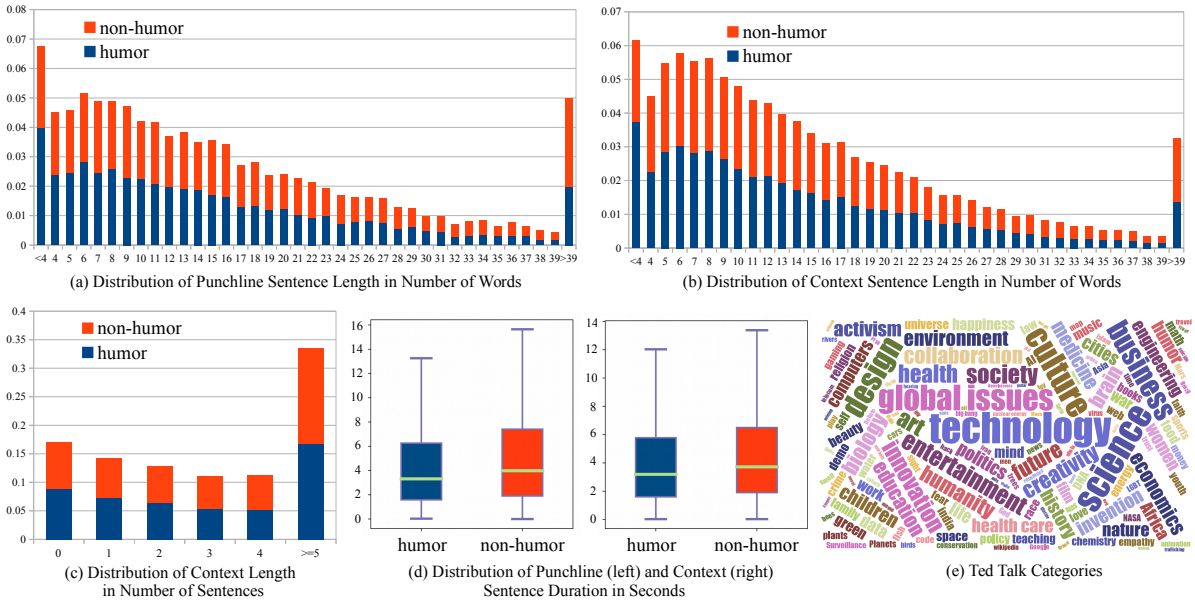


Figure 2: Overview of UR-FUNNY dataset statistics. (a) the distribution of punchline sentence length for humor and non-humor cases. (b) the distribution of context sentence length for humor and non-humor cases. (c) distribution of the number of sentences in the context. (d) distribution of the duration (in seconds) of punchline and context sentences. (e) topics of the videos in UR-FUNNY dataset. Best viewed in zoomed and color.

words and phonemes in the sentences (Yuan and Liberman, 2008). Therefore, an alignment is established between text, visual and acoustic. Utilizing this alignment, the timing of punchline as well as context is extracted for all instances in the dataset.

### 3.2 Dataset Statistics

The high level statistics of UR-FUNNY dataset are presented in Table 2. Total duration of the entire dataset is 90.23 hours. There are a total of 1741 distinct speakers and a total of 417 distinct topics in the UR-FUNNY dataset. Figure 2.e shows the word cloud of the topics based on log-frequency of the topic. The top most five frequent topics are technology, science, culture, global issues and design [4]. There are in total 16514 video segments of humor and not humor instances (equal splits of 8257). The average duration of each data instance is 19.67 seconds, with context an average of 14.7 and punchline with an average of 4.97 seconds. The average number of words in punchline is 16.14 and the average number of words in context sentences is 14.80.

Figure 2 shows an overview for some of the important statistics of UR-FUNNY dataset. Figure 2.a demonstrates the distribution of punchline for humor and non-humor cases based on number

| | |
|---|---|
| **General** | |
| total #videos | 1866 |
| total duration in hour | 90.23 |
| total #distinct speakers | 1741 |
| total #distinct topics | 417 |
| total #humor instances | 8257 |
| total #non-humor instances | 8257 |
| total #words | 965573 |
| total #unique words | 32995 |
| total #sentences | 63727 |
| avg length of sentences in words | 15.15 |
| avg duration of sentences (s) | 4.64 |
| **Punchline** | |
| #sentences in punchline | 1 |
| avg #words in punchline | 16.14 |
| avg #words in humorous punchline | 15.17 |
| avg #words in non-humorous punchline | 17.10 |
| avg duration of punchline (s) | 4.97 |
| avg duration of humorous punchline (s) | 4.58 |
| avg duration of non-humorous punchline (s) | 5.36 |
| **Context** | |
| avg total #words in context | 42.33 |
| avg #words in context sentences | 14.80 |
| avg #sentences in context | 2.86 |
| avg #sentences in humorous context | 2.82 |
| avg #sentences in non-humorous context | 2.90 |
| avg duration of context (s) | 14.7 |
| avg duration of humorous context (s) | 13.79 |
| avg duration of non-humorous context (s) | 15.62 |
| avg duration of context sentences (s) | 4.25 |
| avg duration of humorous context sentences (s) | 4.79 |
| avg duration of non-humorous context sentences (s) | 4.52 |

Table 2: Summary of the UR-FUNNY dataset statistics. Here, '#' denotes number, 'avg' denotes average and 's' denotes seconds

[4]Metadata collected from www.ted.com

| | Train | Val | Test |
|---|---|---|---|
| #humor instances | 5306 | 1313 | 1638 |
| #not humor instances | 5292 | 1313 | 1652 |
| #videos used | 1166 | 300 | 400 |
| #speakers | 1059 | 294 | 388 |
| avg #words in punchline | 15.81 | 16.94 | 16.55 |
| avg #words in context | 41.69 | 42.86 | 43.94 |
| avg #sentences in context | 2.84 | 2.81 | 2.95 |
| punchline avg duration(second) | 4.85 | 5.25 | 5.15 |
| context avg duration(second) | 14.39 | 14.91 | 15.54 |

Table 3: Statistics of train, validation & test folds of UR-FUNNY dataset. Here, 'avg' denotes average and '#' denotes number.

of words. There is no clear distinction between humor and non-humor punchlines as both follow similar distribution. Similarly, Figure 2.b shows the distribution of number of words per context sentence. Both humor and non-humor context sentences follow the same distribution. Majority ($\geq 90\%$) of punchlines have length less than 32. In terms of number of seconds, Figure 2.d shows the distribution of punchline and context sentence length in terms of seconds. Figure 2.c demonstrates the distribution of number of context sentences per humor and non-humor data instances. Number of context sentences per humor and non-humor case is also roughly the same. The statistics in Figure 2 show that there is no trivial or degenerate distinctions between humor and non-humor cases. Therefore, classification of humor versus non-humor cases cannot be done based on simple measures (such as number of words); it requires understanding the content of sentences.

Table 3 shows the standard train, validation and test folds of the UR-FUNNY dataset. These folds share no speaker with each other - hence standard folds are speaker independent (Zadeh et al., 2016). This minimizes the chance of overfitting to identity of the speakers or their communication patterns.

### 3.3 Extracted Features

For each modality, the extracted features are as follows:

**Language (text):** Glove word embeddings (Pennington et al., 2014) are used as pre-trained word vectors for the text features. P2FA forced alignment model (Yuan and Liberman, 2008) is used to align the text and audio on phoneme level. From the force alignment, we extract the timing annotations of context and punchline on word level. Then, the acoustic and visual cues are aligned on word level by interpolation (Chen et al., 2017).

**Acoustic:** COVAREP software (Degottex et al., 2014) is used to extraction acoustic features at the rate of 30 frame/sec. We extract following 81 features: fundamental frequency (F0), Voiced/Unvoiced segmenting features (VUV) (Drugman and Alwan, 2011), normalized amplitude quotient (NAQ), quasi open quotient (QOQ) (Kane and Gobl, 2013), glottal source parameters (H1H2, Rd,Rd conf) (Drugman et al., 2012; Alku et al., 2002, 1997), parabolic spectral parameter (PSP), maxima dispersion quotient (MDQ), spectral tilt/slope of wavelet responses (peak/slope), Mel cepstral coefficient (MCEP 0-24), harmonic model and phase distortion mean (HMPDM 0-24) and deviations (HMPDD 0-12), and the first 3 formants. These acoustic features are related to emotions and tone of speech.

**Visual:** OpenFace facial behavioral analysis tool (Baltrušaitis et al., 2016) is used to extract the facial expression features at the rate of 30 frame/sec. We extract all facial Action Units (AU) features based on the Facial Action Coding System (FACS) (Ekman, 1997). Rigid and non-rigid facial shape parameters are also extracted (Baltrušaitis et al., 2016). We observed that the camera angle and position changes frequently during TED presentations. However, for the majority of time, the camera stays focused on the presenter. Due to the volatile camera work, the only consistently available source of visual information was the speaker's face.

UR-FUNNY dataset is publicly available for download alongside all the extracted features.

## 4 Multimodal Humor Detection

In this section, we first outline the problem formulation for performing binary multimodal humor detection on UR-FUNNY dataset. We then proceed to study the UR-FUNNY dataset through the lens of a contextualized extension of Memory Fusion Network (MFN) (Zadeh et al., 2018a) - a state-of-the-art model in multimodal language.

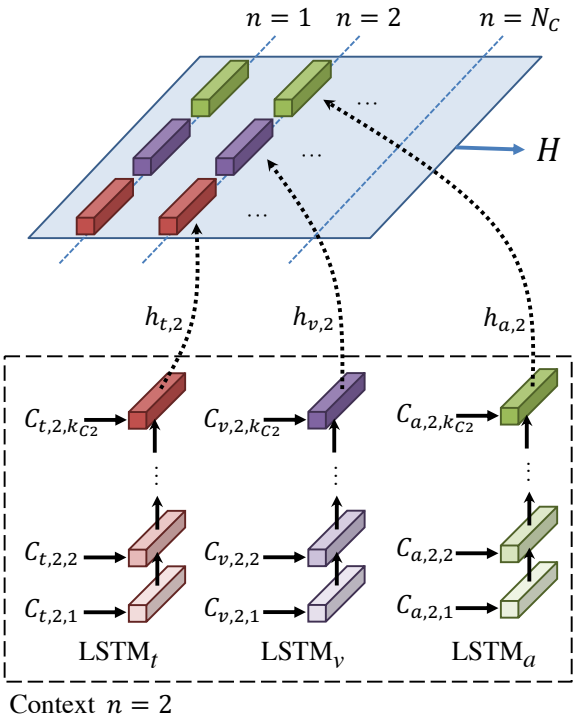


Figure 3: The structure of Unimodal Context Network as outlined in Section 4.2.1. For demonstration purpose, we show the case for $n = 2$ (second context sentence). After $n = N_C$, the output $H$ (outlined by blue) is complete. Best viewed in color.

### 4.1 Problem Formulation

UR-FUNNY dataset is a multimodal dataset with three modalities of text, visual and acoustic. We denote the set of these modalities as $M = \{t, v, a\}$. Each of the modalities come in a sequential form. We assume word-level alignment between modalities (Yuan and Liberman, 2008). Since frequency of the text modality is less than visual and acoustic (i.e. visual and acoustic have higher sampling rate), we use expected visual and acoustic descriptors for each word (Chen et al., 2017). After this process, each modality has the same sequence length (each word has a single visual and acoustic vector accompanied with it).

Each data sample in the UR-FUNNY can be described as a triplet $(l, P, C)$ with $l$ being a binary label for humor or non-humor, $P$ is the punchline and $C$ is the context. Both punchline and context have multiple modalities $P = \{P_m; m \in M\}$, $C = \{C_m; m \in M\}$. If there are $N_C$ context sentences accompanying the punchline, then $C_m = [C_{m,1}, C_{m,2}, \ldots, C_{m,N_C}]$ - are context sentences start from first sentence to the last ($N_C$) sentence. $K_P$ is the number of words in the punchline and $K_{Cn}|_{n=1}^{N_C}$ is the number of words in each of the context sentences respectively. Using this notation, $P_{m,k}$ refers to the $k$th entry in the modality $m$ of the punchline. Similarly, $C_{m,n,k}$ refers to the $k$th entry in the modality $m$ of the $n$th context sentence.

Models developed on UR-FUNNY dataset are trained on triplets of $(l, P, C)$. During testing only a tuple $(P, C)$ is given to predict the $l$. Here $l$ is the label for laughter, specifically whether or not the inputs $P, C$ are likely to trigger a laughter.

### 4.2 Contextual Memory Fusion Baseline

Memory Fusion Network (MFN) is among the state-of-the-art models for several multimodal datasets (Zadeh et al., 2018a). We devise an extension of the MFN model, named Contextual Memory Fusion Network(C-MFN), as a baseline for humor detection on UR-FUNNY dataset. This is done by introducing two components to allow the involvement of context in the MFN model: 1) *Unimodal Context Network*, where information from each modality is encoded using $M$ Long-short Term Memories (LSTM), 2) *Multimodal Context Network*, where unimodal context information are fused (using self-attention) to extract the multimodal context information. We discuss the components of the C-MFN model in the following such sections.

#### 4.2.1 Unimodal Context Network

To model the context, we first model each modality within the context. Unimodal Context Network (Figure 3) consists of $M$ LSTMs, one for each modality $m \in M$ denoted as LSTM$_m$. For each context sentence $n$ of each modality $m \in M$, LSTM$_m$ is used to encode the information into a single vector $h_{m,n}$. This single vector is the last output of the LSTM$_m$ over $C_{m,n}$ as input. The recurrence step for each LSTM is the utterance of each word (due to word-level alignment visual and acoustic modalities also follow this time-step). The output of the Unimodal Context Network is the set $H = \{h_{m,n}; m \in M, 1 \leq n < N_C\}$.

#### 4.2.2 Multimodal Context Network

Multimodal Context Network (Figure 4) learns a multimodal representation of the context based on the output $H$ of the Unimodal Context Network. Sentences and modalities in the context can form complex asynchronous spatio-temporal relations. For example, during the gradual buildup of the

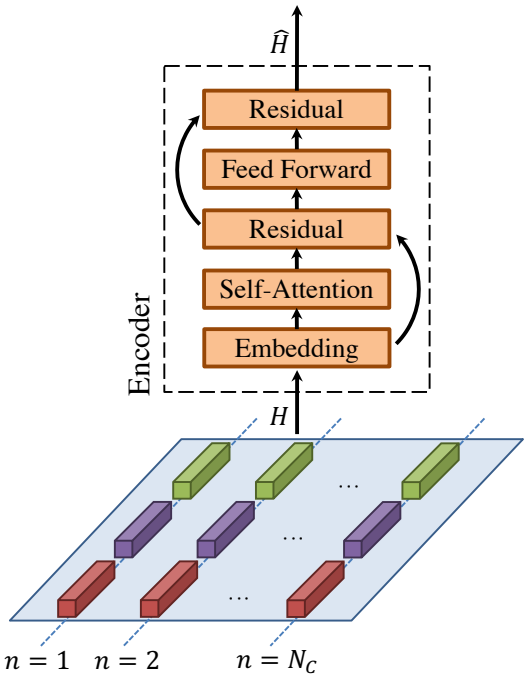


Figure 4: The structure of Multimodal Context Network as outlined in Section 4.2.2. The output $H$ of the Unimodal Context Network is connected to an encoder module to get the multimodal output $\hat{H}$. For the details of components outlined in orange please refer to the authors' original paper. (Vaswani et al., 2017). Best viewed in color.

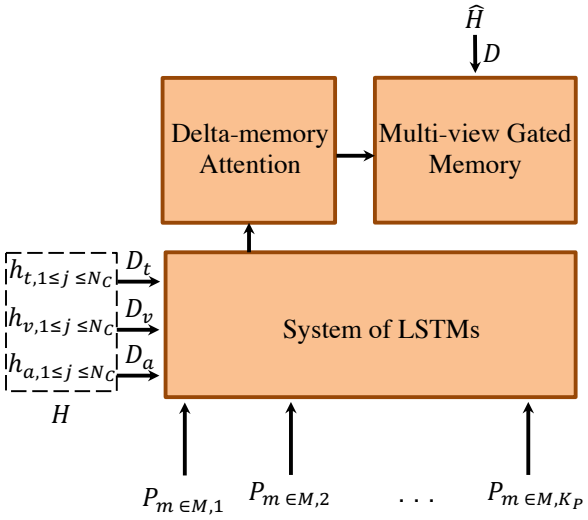


Figure 5: The initialization and recurrence process of Memory Fusion Network (MFN). The outputs of Unimodal and Multimodal Context Networks ($H$ and $\hat{H}$) are used initializing the MFN neural components. For the details of components outlined in orange please refer to the authors' original paper (Zadeh et al., 2018a). Best viewed in color.

context, the speaker's facial expression may be impacted due to an arbitrary previously uttered sentence. Transformers (Vaswani et al., 2017) are a family of neural models that specialize in finding various temporal relations between their inputs through self-attention. By concatenating representations $h_{m\in M,n}$ (i.e. for all $M$ modalities of the $n$th context), self-attention model can be applied to find asynchronous spatio-temporal relations in the context. We use an encoder with 6 intermediate layers to derive a multimodal representation $\hat{H}$ conditioned on $H$. $\hat{H}$ is also spatio-temporal (as produced output of encoders in a transformer are). The output of Multimodal Context Network is the output $\hat{H}$ of the encoder.

#### 4.2.3 Memory Fusion Network (MFN)

After learning unimodal ($H$) and multimodal ($\hat{H}$) representations of context, we use a Memory Fusion Network (MFN) (Zadeh et al., 2018a) to model the punchline (Figure 5). MFN contains 2 types of memories: a System of LSTMs with $M$ unimodal memories to model each modality in punchline, and a Multi-view Gated Memory which stores multimodal information. We use a simple trick to combine the Context Networks (Unimodal and Multimodal) with the MFN: we initialize the memories in the MFN using the outputs $H$ (unimodal representation) and $\hat{H}$ (multimodal representation). For System of LSTMs, this is done by initializing the LSTM cell state of modality $m$ with $\mathcal{D}_m(h_{m,1\leq n<N_C})$. $\mathcal{D}_m$ is a fully connected neural network that maps the information from $h_{m,1\geq j\geq N_C}$ ($m$th modality in context) to the cell state of the $m$th LSTM in the System of LSTMs. The Multi-view Gated Memory is initialized based on a non-linear projection $\mathcal{D}(\hat{H})$ where $\mathcal{D}$ is a fully connected neural network. Similar to context where modalities are aligned at word level, punchline is also aligned the same way. Therefore a word-level implementation of the MFN is used, where a word and accompanying visual and acoustic descriptors of punchline are used as input to the System of LSTMs at each time-step. The Multi-view Gated Memory is updated iteratively at every recurrence of the System of LSTMs using a Delta-memory Attention Network.

The final prediction of humor is conditioned on the last state of the System of LSTMs and Multi-view Gated Memory using an affine mapping with Sigmoid activation.

## 5 Experiments

In the experiments of this paper, our goal is to establish a performance baseline for the UR-FUNNY dataset. Furthermore, we aim to understand the role of context and punchline, as well as role of individual modalities in the task of humor detection. For all the experiments, we use the proposed contextual extension of Memory Fusion Network (MFN), called C-MFN (Section 4.2). Aside the proposed **C-MFN** model, the following variants are also studied:

**C-MFN (P):** This variant of the C-MFN uses only punchline with no contextual information. Essentially, this is equivalent to a MFN model since initialization trick is not used.

**C-MFN (C):** This variant of the C-MFN uses only contextual information without punchline. Essentially, this is equivalent to removing the MFN and directly conditioning the humor prediction on the Unimodal and Multimodal Context Network outputs (Sigmoid activated neuron after applying $D_M; m \in M$ on $H$ and $D$ on $\hat{H}$).

The above variants of the C-MFN allow for studying the importance of punchline and context in modeling humor. Furthermore, we compare the performance of the C-MFN variants in the following scenarios: **(T)** a only text modality is used without visual and acoustic, **(T+V)** text and visual modalities are used without acoustic, **(T+A)** text and acoustic modalities are used without visual, **(A+V)** only visual and acoustic modalities are used, **(T+A+V)** all modalities are used together.

We compare the performance of C-MFN variants across the above scenarios. This allows for understanding the role of context and punchline in humor detection, as well as the importance of different modalities. All the models for our experiments are trained using categorical cross-entropy. This measure is calculated between the output of the model and ground-truth labels. We also compare the performance of C-MFN to a Random Forest classifier as another strong non-neural baseline. We use the summary features of punchline and context for the Random Forest classifier.

| Modality | T | A+V | T+A | T+V | T+A+V |
|---|---|---|---|---|---|
| C-MFN (P) | 62.85 | 53.3 | 63.28 | 63.22 | 64.47 |
| C-MFN (C) | 57.96 | 50.23 | 57.78 | 57.99 | 58.45 |
| C-MFN | 64.44 | 57.99 | 64.47 | 64.22 | 65.23 |

Table 4: Binary accuracy for different variants of C-MFN and training scenarios outlined in Section 5. The best performance is achieved using all three modalities of text (T), visual (V) and acoustic (A).

## 6 Results and Discussion

The results of our experiments are presented in Table 4. Results demonstrate that both context and punchline information are important as C-MFN outperforms C-MFN (P) and C-MFN (C) models. Punchline is the most important component for detecting humor as the performance of C-MFN (P) is significantly higher than C-MFN (C).

Models that use all modalities (T+A+V) outperform models that use only one or two modalities (T, T+A, T+V, A+V). Between text (T) and nonverbal behaviors (A+V), text shows to be the most important modality. Most of the cases, both modalities of visual and acoustic improve the performance of text alone (T+V, T+A).

Based on the above observations, each neural component of the C-MFN model is useful in improving the prediction of humor. The results also indicate that modeling humor from a multimodal perspective yields successful results. Furthermore, both context and punchline are important in understanding humor.

The highest accuracy achieved by Random Forest baseline (after hyper parameter tuning & using same folds as C-MFN) was 57.78%, which is higher than random baseline but lower than C-MFN (65.23%). In addition, C-MFN achieves higher accuracy than similar notable unimodal previous work (58.9%) where only punchline and textual information were used (Chen and Lee, 2017) . The human performance [5] on the UR-FUNNY dataset is 82.5%.

The results from Table 4 demonstrate that while a state-of-the-art model can achieve a reasonable level of success in modeling humor, there is still a large gap between human-level performance with state of the art. Therefore, UR-FUNNY

[5]This is calculated by averaging the performance of two annotators over a shuffled set of 100 humor and 100 non-humor cases. The annotators are given the same input as the machine learning models (similar context and punchline). The annotators agree 84% of times.

dataset presents new challenges to the field of NLP, specifically research areas of humor detection and multimodal language analysis.

## 7 Conclusion

In this paper, we presented a new multimodal dataset for humor detection called UR-FUNNY. This dataset is the first of its kind in the NLP community. Humor detection is done from the perspective of predicting laughter - similar to (Chen and Lee, 2017). UR-FUNNY is diverse in both speakers and topics. It contains three modalities of text, visual and acoustic. We study this dataset through the lens of a Contextualized Memory Fusion Network (C-MFN). Results of our experiments indicate that humor can be better modeled if all three modalities are used together. Furthermore, both context and punchline are important in understanding humor. The dataset and the accompanying experiments are publicly available.

## Acknowledgment

This research was supported in part by grant W911NF-15-1-0542 and W911NF-19-1-0029 with the US Defense Advanced Research Projects Agency (DARPA) and the Army Research Office (ARO).

This material is based upon work partially supported by the National Science Foundation (Awards #1734868 #1722822) and National Institutes of Health. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of National Science Foundation or National Institutes of Health, and no official endorsement should be inferred.

## References

Paavo Alku, Tom Bäckström, and Erkki Vilkman. 2002. Normalized amplitude quotient for parametrization of the glottal flow. *the Journal of the Acoustical Society of America*, 112(2):701–710.

Paavo Alku, Helmer Strik, and Erkki Vilkman. 1997. Parabolic spectral parametera new method for quantification of the glottal flow. *Speech Communication*, 22(1):67–79.

Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. 2019. Multimodal machine learning: A survey and taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(2):423–443.

Tadas Baltrušaitis, Peter Robinson, and Louis-Philippe Morency. 2016. Openface: an open source facial behavior analysis toolkit. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–10. IEEE.

Elham J Barezi, Peyman Momeni, Pascale Fung, et al. 2018. Modality-based factorization for multimodal fusion. *arXiv preprint arXiv:1811.12624*.

Dario Bertero, Pascale Fung, X Li, L Wu, Z Liu, B Hussain, Wc Chong, Km Lau, Pc Yue, W Zhang, et al. 2016. Deep learning of audio and language features for humor prediction. In *LREC*.

Carlos Busso, Murtaza Bulut, Chi-Chun Lee, Abe Kazemzadeh, Emily Mower, Samuel Kim, Jeannette N Chang, Sungbok Lee, and Shrikanth S Narayanan. 2008. Iemocap: Interactive emotional dyadic motion capture database. *Language resources and evaluation*, 42(4):335.

Lei Chen and Chong Min Lee. 2017. Predicting audience's laughter during presentations using convolutional neural network. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 86–90.

Minghai Chen, Sen Wang, Paul Pu Liang, Tadas Baltrušaitis, Amir Zadeh, and Louis-Philippe Morency. 2017. Multimodal sentiment analysis with word-level fusion and reinforcement learning. In *Proceedings of the 19th ACM International Conference on Multimodal Interaction*, pages 163–171. ACM.

Peng-Yu Chen and Von-Wun Soo. 2018. Humor recognition using deep learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 113–117.

Gilles Degottex, John Kane, Thomas Drugman, Tuomo Raitio, and Stefan Scherer. 2014. Covarepa collaborative voice analysis repository for speech technologies. In *2014 ieee international conference on acoustics, speech and signal processing (icassp)*, pages 960–964. IEEE.

Thomas Drugman and Abeer Alwan. 2011. Joint robust voicing detection and pitch estimation based on residual harmonics. In *Twelfth Annual Conference of the International Speech Communication Association*.

Thomas Drugman, Mark Thomas, Jon Gudnason, Patrick Naylor, and Thierry Dutoit. 2012. Detection of glottal closure instants from speech signals: A quantitative review. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(3):994–1006.

Rosenberg Ekman. 1997. *What the face reveals: Basic and applied studies of spontaneous expression using the Facial Action Coding System (FACS)*. Oxford University Press, USA.

Randy Garner. 2005. Humor, analogy, and metaphor: Ham it up in teaching. *Radical Pedagogy*, 6(2):1.

William E Hauck and John W Thomas. 1972. The relationship of humor to intelligence, creativity, and intentional and incidental learning. *The journal of experimental education*, 40(4):52–55.

Devamanyu Hazarika, Soujanya Poria, Amir Zadeh, Erik Cambria, Louis-Philippe Morency, and Roger Zimmermann. 2018. Conversational memory network for emotion recognition in dyadic dialogue videos. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 2122–2132.

John Kane and Christer Gobl. 2013. Wavelet maxima dispersion for breathy to tense voice discrimination. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(6):1170–1179.

Paul Pu Liang, Ziyin Liu, Amir Zadeh, and Louis-Philippe Morency. 2018. Multimodal language analysis with recurrent multistage fusion. *arXiv preprint arXiv:1808.03920*.

Zhun Liu, Ying Shen, Varun Bharadhwaj Lakshminarasimhan, Paul Pu Liang, Amir Zadeh, and Louis-Philippe Morency. 2018. Efficient low-rank multimodal fusion with modality-specific factors. *arXiv preprint arXiv:1806.00064*.

Rada Mihalcea and Carlo Strapparava. 2005. Making computers laugh: Investigations in automatic humor recognition. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 531–538. Association for Computational Linguistics.

Luke de Oliveira, ICME Stanford, and Alfredo Láinez Rodrigo. 2017. Humor detection in yelp reviews.

Sunghyun Park, Han Suk Shim, Moitreya Chatterjee, Kenji Sagae, and Louis-Philippe Morency. 2014. Computational analysis of persuasiveness in social multimedia: A novel dataset and multimodal prediction approach. In *Proceedings of the 16th International Conference on Multimodal Interaction*, pages 50–57. ACM.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Hai Pham, Paul Pu Liang, Thomas Manzini, Louis-Philippe Morency, and Barnabas Poczos. 2019. Found in translation: Learning robust joint representations by cyclic translations between modalities. *arXiv preprint arXiv:1812.07809*.

Soujanya Poria, Erik Cambria, Devamanyu Hazarika, Navonil Mazumder, Amir Zadeh, and Louis-Philippe Morency. 2017. Multi-level multiple attentions for contextual multimodal sentiment analysis. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 1033–1038. IEEE.

Soujanya Poria, Devamanyu Hazarika, Navonil Majumder, Gautam Naik, Erik Cambria, and Rada Mihalcea. 2018a. Meld: A multimodal multi-party dataset for emotion recognition in conversations. *arXiv preprint arXiv:1810.02508*.

Soujanya Poria, Amir Hussain, and Erik Cambria. 2018b. *Multimodal Sentiment Analysis*, volume 8. Springer.

Robert R Provine. 1992. Contagious laughter: Laughter is a sufficient stimulus for laughs and smiles. *Bulletin of the Psychonomic Society*, 30(1):1–4.

Rachel Rakov and Andrew Rosenberg. 2013. " sure, i did the right thing": a system for sarcasm detection in speech. In *Interspeech*, pages 842–846.

Vilayanur S Ramachandran. 1998. The neurology and evolution of humor, laughter, and smiling: the false alarm theory. *Medical hypotheses*, 51(4):351–354.

Verónica Pérez Rosas, Rada Mihalcea, and Louis-Philippe Morency. 2013. Multimodal sentiment analysis of spanish online videos. *IEEE Intelligent Systems*, 28(3):38–45.

Dafna Shahaf, Eric Horvitz, and Robert Mankoff. 2015. Inside jokes: Identifying humorous cartoon captions. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1065–1074. ACM.

David Stauffer. 1999. Let the good times roll: Building a fun culture. *Harvard Management Update*, 4(10):4–6.

Yao-Hung Hubert Tsai, Paul Pu Liang, Amir Zadeh, Louis-Philippe Morency, and Ruslan Salakhutdinov. 2018. Learning factorized multimodal representations. *arXiv preprint arXiv:1806.06176*.

Robert A Vartabedian and Laurel Klinger Vartabedian. 1993. Humor in the workplace: A communication challenge.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Yansen Wang, Ying Shen, Zhun Liu, Paul Pu Liang, Amir Zadeh, and Louis-Philippe Morency. 2019. Words can shift: Dynamically adjusting word representations using nonverbal behaviors. *arXiv preprint arXiv:1811.09362*.

Melissa B Wanzer, Ann B Frymier, and Jeffrey Irwin. 2010. An explanation of the relationship between instructor humor and student learning: Instructional humor processing theory. *Communication Education*, 59(1):1–18.

Martin Wöllmer, Felix Weninger, Tobias Knaup, Björn Schuller, Congkai Sun, Kenji Sagae, and Louis-Philippe Morency. 2013. Youtube movie reviews: Sentiment analysis in an audio-visual context. *IEEE Intelligent Systems*, 28(3):46–53.

Diyi Yang, Alon Lavie, Chris Dyer, and Eduard Hovy. 2015. Humor recognition and humor anchor extraction. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2367–2376.

Jiahong Yuan and Mark Liberman. 2008. Speaker identification on the scotus corpus. *Journal of the Acoustical Society of America*, 123(5):3878.

Amir Zadeh, Minghai Chen, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. 2017. Tensor fusion network for multimodal sentiment analysis. *arXiv preprint arXiv:1707.07250*.

Amir Zadeh, Paul Pu Liang, Navonil Mazumder, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. 2018a. Memory fusion network for multi-view sequential learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Amir Zadeh, Paul Pu Liang, Louis-Philippe Morency, Soujanya Poria, Erik Cambria, and Stefan Scherer. 2018b. Proceedings of grand challenge and workshop on human multimodal language (challenge-hml). In *Proceedings of Grand Challenge and Workshop on Human Multimodal Language (Challenge-HML)*.

Amir Zadeh, Rowan Zellers, Eli Pincus, and Louis-Philippe Morency. 2016. Mosi: multimodal corpus of sentiment intensity and subjectivity analysis in online opinion videos. *arXiv preprint arXiv:1606.06259*.

AmirAli Bagher Zadeh, Paul Pu Liang, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. 2018c. Multimodal language analysis in the wild: Cmu-mosei dataset and interpretable dynamic fusion graph. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2236–2246.

# Partners in Crime: Multi-view Sequential Inference for Movie Understanding

**Nikos Papasarantopoulos**[♠] **Lea Frermann**[♢] **Mirella Lapata**[♠] **Shay B. Cohen**[♠]
[♠]School of Informatics, University of Edinburgh, UK
[♢]School of Computing and Information Systems, University of Melbourne, Australia
nikos.papasa@ed.ac.uk, lea.frermann@unimelb.edu.au
mlap@inf.ed.ac.uk, scohen@inf.ed.ac.uk

## Abstract

Multi-view learning algorithms are powerful representation learning tools, often exploited in the context of multimodal problems. However, for problems requiring inference at the token-level of a sequence (that is, a separate prediction must be made for every time step), it is often the case that single-view systems are used, or that more than one views are fused in a simple manner. We describe an incremental neural architecture paired with a novel training objective for incremental inference. The network operates on multi-view data. We demonstrate the effectiveness of our approach on the problem of predicting perpetrators in crime drama series, for which our model significantly outperforms previous work and strong baselines. Moreover, we introduce two tasks, crime case and speaker type tagging, that contribute to movie understanding and demonstrate the effectiveness of our model on them.[1]

## 1 Introduction

While many natural language processing (NLP) problems concern exclusively textual or speech data, the integration of multimodal information (such as images, video or audio) is beneficial for a variety of problems. For example, visual information has been used in affect analysis (Kahou et al., 2016), sentiment analysis (Morency et al., 2011) and machine translation (Calixto et al., 2017; Lala and Specia, 2018). This is also the case for problems which are sequential in nature, such as video summarization (Smith and Kanade, 1998), continuous prediction of affect (Nicolaou et al., 2011) or engagement level prediction (Rehg et al., 2013).

Most existing multi-view representation learning approaches are tested in an unsupervised setup where the multi-view representations are learned separately from the task, and are designed to accommodate the learning of representation for monolithic (albeit multi-view) data points, not sequences (see Wang et al. 2015 for a survey). In this paper, we propose a neural architecture coupled with a novel training objective that integrates multi-view information for sequence prediction problems. Our model creates a multimodal embedding for every element of a sequence by using the correlational Gated Recurrent Unit (corrGRU; Yang et al. 2017) and makes token-level predictions based on those embeddings. Our training objective combines a supervision-guided term (cross-entropy) with a multi-view correlation objective on the available modalities.

We demonstrate the effectiveness of our model in an incremental inference setup (Figure 1), where it makes predictions on the fly without encoding the sequence in full, a more realistic scenario of interacting with data. This is a critical feature for online applications such as simultaneous translation (interpretation) and also a desirable behavior for movie processing models that mimic a human viewer watching a movie for the first time.

We evaluate our architecture on three tasks pertaining to movie understanding. Specifically, we use the recently introduced dataset of Frermann et al. (2018), which consists of episodes of the Crime Series Investigation (CSI) television series, segmented and aligned for three different modali-

[1]Our code is available at https://github.com/papagandalf/multiview_csi.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2057–2067,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

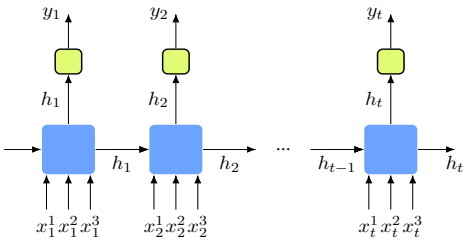


Figure 1: General unfolded overview of our model for a multimodal example $x$ with three modalities: at each time step $t \in \{0, 1, ..., T\}$, representations $x_t^k$ for each of three views are fed to a cell. The cell outputs a joint representation $h_t$ for all the views which is fed to a softmax layer that generates a prediction $y_t$, depending on the task at hand. Training is performed with a correlation objective between the representations of $x_t^k$.

ties: image, audio and text. Originally, the dataset was introduced to train models in perpetrator mention identification, that is, viewing each episode as a sequence of multimodal elements, predicting a binary label indicating whether the perpetrator of a crime case is mentioned in each element. Leveraging the annotations of the dataset, we model two additional tasks: case segmentation (episodes can alternate between more than one crime case) and speaker type tagging (each utterance in the video can come from one type of speaker: detective, perpetrator, suspect, extra or none). Successfully modeling those two tasks provides structural information about episodes and informs perpetrator identification, a task solved on a per case and not on a per episode basis. The three tasks can be regarded as a form of *shallow movie understanding*, since they perform analysis on the structural level (cases), the dialogue level (speaker type) and the plot level (perpetrators).

We show that our multi-view model consistently outperforms models that integrate multimodal information by concatenating the representations of available views, across all three tasks. As a comparison, we also describe a non-multi-view variant of our architecture, equipped with a supervised multi-head attention module that can take advantage of two levels of annotation (e.g. token-level and sentence-level).

The contributions of this paper are the following:

- We propose a multi-view sequential inference neural architecture and use a novel training objective for training an RNN consisting of correlational GRU cells.
- We highlight the importance of multi-view fusion for multimodal applications, by comparing our model with a non-multi-view variant that employs multi-head supervised attention to make use of both the sentence-level and the token-level perpetrator annotations of the dataset.
- We introduce two novel tasks pertaining to shallow movie understanding that can be tackled in the context of television series data. We empirically show the effectiveness of our architecture and training objective on the perpetrator mention identification and the two newly introduced tasks, by using the Crime Scene Investigation (CSI) television series dataset. Notably, for the perpetrator identification task, our model significantly outperforms previous state of the art.

## 2 Background

**Series Understanding**[2] The abundance of series video data can benefit multimodal machine learning research and applications. Series commonly span many episodes (organized in loosely or tightly connected seasons), providing a large amount of data that data-hungry models can take advantage of. The sheer volume of data gives rise to several practical problems that machine learning models can tackle, such as the segmentation of continuous video streams to semantically coherent fragments (Del Fabro and Böszörmenyi, 2013) and speaker diarization (Miró et al., 2012; Bredin et al., 2014).

Work on movie/series analysis can be classified in three broad categories. *Deep semantic understanding* includes tasks that require thorough content analysis and reasoning, for example movie question answering (Tapaswi et al., 2016; Kim et al., 2017) or movie description (Rohrbach et al., 2016). *External understanding* refers to tasks whose end goal is not the analysis of the video content itself, but meta-information extraction relevant to preferences and recommendations for consumers of the videos (Bennett et al., 2007; Shi et al., 2013; Yang et al., 2012).

*Shallow understanding* refers to tasks that operate on the content level and extract content-related information, albeit without requiring complex reasoning. Their output is more tailored to structured prediction. Example tasks include speaker iden-

[2]We use the term "series" to refer to episodic shows, broadcast via television or other channels.

tification (Knyazeva et al., 2015), movie segmentation (Liu et al., 2013) and perpetrator mention identification (Frermann et al., 2018). We choose to tackle three problems of shallow understanding, namely a crime case sequence tagging problem, a speaker type sequence tagging problem and the problem of perpetrator mention identification.

**Multi-view Learning** Conventional machine learning algorithms treat all characteristics of training examples as features describing the input. When more than one modality or more than one sources of data (for example, images taken from different viewpoints) are available, view fusion can be achieved by early, late or hybrid fusion methods (Atrey et al., 2010). A simple concatenation of representations at the feature level is referred to as *early fusion*, while the integration of outputs of different modality-specific modules is called *late fusion*. Concatenation may cause overfitting in the case of a small size training samples and at the same time it is not intuitive, since the statistical properties of each view can be lost in the learning process. Multi-view learning algorithms extend early fusion, as they create sophisticated representations in which all available views are fused.

Our work extends the set of multi-view counterparts of popular sequence models. Rajagopalan et al. (2016) propose a general architecture that provides a degree of flexibility in designing different multi-view LSTM cells according to the application at hand and experiment with behavior recognition and image captioning. Ren et al. (2016) propose a multi-modal variant of LSTM and apply it to the task of speaker identification. Zadeh et al. (2018a) use an attention module and a multi-view gated memory to capture and summarize inter-modality interactions. Our proposed model enforces correlation between the representations of the available modalities; a technique that has been studied also for non-sequential neural models (Wang et al., 2015; Chang et al., 2018).

While the term *multi-view* does not always refer to multimodal settings, ideas from multi-view representation learning research are especially compelling for multimodal applications. Our model can be applied to sequential multi-view problems which are not necessarily multimodal.

**Attention Models** Attention mechanisms (Bahdanau et al., 2015) in various forms have been used in several multimodal applications, such as sentiment analysis, speaker trait recognition and emotion recognition (Zadeh et al., 2018b), machine translation (Caglayan et al., 2016), image (Xu et al., 2015) and video description (Hori et al., 2017). Broadly, an attention mechanism modifies the output of a sequence representation, based on the coherence of each of the elements of the sequence to a specific "query". Information learned by the attention mechanism may have a distinct conceptual importance (e.g. alignments in machine translation) or simply indicate which elements of the input contribute more to the final output representation.

Attention mechanisms can be learned along with the rest of the network in an end-to-end fashion, or can be explicitly supervised by providing the model with pre-calculated attention scores. Supervised attention has been shown to boost the performance of models for machine translation (Mi et al., 2016), constituency parsing (Kamigaito et al., 2017), event detection (Liu et al., 2017b) and aspect-based sentiment analysis (Cheng et al., 2017). Furthermore, image attention mechanisms guided by weak or direct supervision have been proposed for the tasks of image (Liu et al., 2017a) and video captioning (Yu et al., 2017).

Multi-head attention mechanisms (Vaswani et al., 2017) employ more than one, independent, attention mechanisms, boasting multiple areas of focus on the input sequence. The main idea behind them is that a single attention head may not prove adequate to capture all the different types and positions of information that are important to the end task. Our attentive model variant uses supervised multi-head attention to take advantage of annotations in two levels of granularity.

## 3 Multi-view Sequential Inference

Taking together the idea of multi-view learning with incremental sequence labeling, we formulate our problem as follows.

We assume a set $\mathcal{X}$ of $M$ examples. Every $X_j \in \mathcal{X}$, $j \in \{1, \ldots, M\}$, consists of $T$ elements, which form a sequence $X_j = [x_{j1}\ x_{j2}\ \ldots\ x_{jT}]$. Sequences with less than $T$ elements are padded to be of length $T$. Each of the elements in the sequence is paired with a label from a set $\mathcal{Y}$ (binary or multi-class), which is the desired output. Lastly, for every $x_{jt}$, information from a set $\mathcal{V}$ of different views is available.

More specifically, we consider a dataset where each $X_j$ is a video and distinguish and model three different views/modalities: image, audio and text (from aligned script or subtitles, if available). Each video $X_j$ is represented as a sequence of short, semantically coherent snippets $x_{jt}$ (for instance, snippets may correspond to subtitle sentences). For each sequence element and for each of the views $\mathcal{V} = \{\text{image, audio, text}\}$ we have a feature vector $(x_{jt})^k$ where $k \in \{1, 2, 3\}$ (indexing the different modalities). In addition, we have labels $y_{jt}$. The problem is to infer the correct label $y_{jt}$ for each $x_{jt}$ at the time it presents itself: data points appear sequentially in a time series and the label prediction should be done using information from the past and the present elements only.

We use a sequence model for incremental modeling of sequential multi-view data. The overall architecture of such a system is that of a Recurrent Neural Network (RNN). A general time-unfolded overview of the architecture for a single example $x \in \mathcal{X}$ is shown in Figure 1. Input segments $x_t^k$ of different views are fed to a cell at every time step, and a single vector $h_t$ combining information from all views is generated. This embedding is fed to an output layer, which in turn outputs a prediction.

**View Fusion** One of the most important and distinguishing features of multi-view models is the framework it uses for the fusion of the different views. We use a multimodal GRU cell (Yang et al., 2017) as a base for experimentation. The cell takes embeddings $x_t^k$ for each view $k$ and time step $t$ and after passing each view from a designated GRU cell, it calculates a multi-view embedding for the time step by passing information of all views through the gates of a separate GRU. The output of the view-specific GRUs is used to calculate the Pearson correlation between the different views; a calculation whose output is added as a separate term to be maximized in the total loss.

Since correlation can be calculated only for a pair of variables, we calculate the total correlation loss as the sum of the correlation losses between all pairs of elements of $\mathcal{V}$. Formally, in each step $t$, the correlation between views is calculated as

$$c_t = \sum_{\substack{k,\ell \in \mathcal{V} \\ k \neq \ell}} \frac{\sum_{i=1}^{L} (h_{it}^{(k)} - \overline{H}_t^{(k)})(h_{it}^{(\ell)} - \overline{H}_t^{(\ell)})}{\sqrt{\sum_{i=1}^{L} (h_{it}^{(k)} - \overline{H}_t^{(k)}) \sum_{i=0}^{L} (h_{it}^{(\ell)} - \overline{H}_t^{(\ell)})}},$$

where $i$ spans over the $L$ elements of each mini-batch, $h_{it}^{(k)}$ is the hidden state calculated by the view-specific GRU for the example $i$, view $k$, at timestep $t$. Moreover, $\overline{H}_t^{(k)} = \frac{1}{L} \sum_{i=1}^{L} h_{it}^{(k)}$.

The correlation term is

$$\mathcal{L}_{\text{corr}} = -\frac{1}{|T|} \sum_{t \in T} c_t,$$

that is the average of the loss calculated for every time step $t$. In order to maximize correlation, the negative sum is used.

In conclusion, view fusion is achieved not only by the design of the multimodal GRU cell itself (weighted sum of view representations and one common hidden representation), but also by the maximization of correlation between the views.

**Learning** The network is trained by jointly minimizing the following objective

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda \mathcal{L}_{\text{corr}},$$

where $\mathcal{L}_{\text{CE}}$ stands for cross-entropy loss, $\mathcal{L}_{\text{corr}}$ is the correlation loss term defined in Section 3 and $\lambda$ weights the contribution of $\mathcal{L}_{\text{corr}}$ to the total loss.

This compound objective function is one of the distinctive features of our approach. It enables the model to take advantage of the labels available from the dataset, and at the same time optimize for the correlation between the available views. The underlying idea is that the constraint of the correlation will push the model to create more informative embeddings than those it would create if only the cross-entropy loss was used.

**Multi-head Attention** Each of the videos of the dataset is assumed to be divided to snippets, with the sequence model operating and making predictions on the snippet level. For each snippet, the text modality may contain a different number of tokens and a fixed-length text representation is generated by a text encoder. We use an RNN as text encoder and the encoded text representation for each snippet is weighted by attention scores calculated over its tokens. Both "query" and "token" representations come from the same sequence (self-attention; Yang et al. 2016). In cases where the dataset contains, apart from snippet-level, also token-level annotations, the attention module can be directly supervised: we add a term

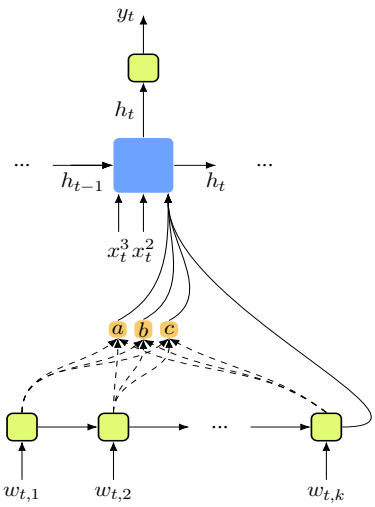


Figure 2: Hierarchical multi-view recurrent model with multi-head attention. Each sentence is encoded with an RNN and three attention heads ($a$, $b$ and $c$) calculate attention scores for each of the tokens $w_{ti}$ of the $t$-th sentence of the script.

to the model's loss that minimizes the error of the attention scores, with respect to token-level annotations. Specifically, we consider a case where the token-level annotations are of three different types and we create a supervised attention head for each type, averaging their outputs. A schematic depiction of this model can be seen in Figure 2.

## 4 Experiments

We describe in this section experiments with our proposed architecture. We use the CSI dataset (Frermann et al., 2018), which consists of 39 episodes of the television series *CSI: Crime Scene Investigation*. In each episode, a team of detectives undertake the solution of one (in 51% of the episodes) or two (49%) crime cases. The three modalities included are text scripts (dialogue subtitles and background scene descriptions), image snapshots from the video and audio segments. Each sentence of the script is aligned with image and audio[3] excerpts. Also, sentences are annotated with the case they belong to (binary label), perpetrator mention labels (binary) and the name of the speaker that uttered them ("None" for scene descriptions). Each speaker belongs to one of the types *detective, perpetrator, suspect, extra* (*none* for scene descriptions). An annotated example excerpt is shown in Figure 3.

[3]Speech has been stripped from the audio track, leaving it only with audio effects and music (so that the text modality will not be deemed redundant and the dataset does not contain overlapping information).

### 4.1 Experimental Setup

For all experiments, we adopted an experimental setup similar to that of Frermann et al. (2018). For text, we use 50-dimensional GloVe vectors (Pennington et al., 2014) and a convolutional text encoder with maxpooling (filters of sizes 3, 4 and 5, each returning a 75-dimension output). Image features are generated by the final hidden layer of the inception-v4 Szegedy et al., 2017 model (dimensionality of 1,546). Audio features are constructed by concatenating five 13-dimensional Mel-Frequency Cepstral Coefficient (MFCC) feature vectors for each interval. For perpetrator mention identification, we use the case level splits, whereas the speaker and case tasks are performed on the episode level. All LSTM and GRU variants have one layer of length 128 and a dropout probability of 0.5 is used. We set the value of $\lambda$ to 0.001 and train for 150 epochs with the Adam optimizer (Kingma and Ba, 2014), setting the initial learning rate at 0.001.

### 4.2 Perpetrator Mention Identification

In order to investigate the effectiveness of our model in the sequential multimodal inference setup, we conduct the following set of experiments on the task of perpetrator mention identification.

**Multi-view Model** The effectiveness of different architectures is shown in the first section of Table 1. The multi-view model (using corrGRU) is compared to early fusion models (using LSTM and GRU cells ), for which the input is the concatenation of the feature vectors of the three modalities, passed through a ReLU activation. It can be seen that the multi-view model outperforms all other models.

**Incremental Inference** *Incremental sequence labeling* refers to making predictions on an incoming sequence when "streamed" in an online fashion. For example, if the sequence is a sentence, we are not allowed to encode the whole sentence first, but instead have to output a relevant label for each word as it arrives in the sequence. Incrementality underlies fundamental human cognition and is essential for scaling systems to large datasets and real-time inference, necessary, for example, in simultaneous translation (interpretation; Bangalore et al., 2012; Yarmohammadi et al., 2013; Cho and Esipova, 2016).



| NONE |
|---|
| Grissom doesn't look worried. He takes his gloves off and puts them on the table. |
| CASE: 1 |
| PERP: 0 |
| CASE TAG: B-1 |
| SP. TAG: O |



| GRISSOM |
|---|
| You ever been to the theater, **Peter**? |
| CASE: 1 |
| PERP: 1 |
| CASE TAG: I-1 |
| SP. TAG: B-D |

Figure 3: An excerpt of the CSI dataset, where the image and text modalities are present. The case and speaker tags use the BIOU (Beginning, Inside, Outside, Unit) format (see 4.3 for details). In this case, "Peter" is the name of the perpetrator (Perp: 1). Both snapshots belong to the first case. The first snapshot does not have a speaker (screenplay description; speaker "None"and speaker type tag "O") and starts a chunk of utterances belonging to the first case (B-1). The second snapshot continues in the first case (I-1) and starts a chunk where the speaker is a detective (B-D).

In order to assess the effectiveness of the incremental inference capabilities of our model, we contrast the output of incremental models (forward-pass unidirectional) to that of similar models that do not perform incremental inference (bidirectional), in Table 1. Both multi-view and non-multi-view bidirectional models look ahead in the sequence, gathering information that is potentially useful for temporal inference. The bidirectional correlational model was trained with an extra correlation loss term, calculated exactly as $\mathcal{L}_{\text{corr}}$, with data from the backward pass. The bidirectional multi-view model does not score as high as the unidirectional multi-view one, though it balances better between precision and recall. Interestingly, results suggest that the incremental multi-view model outperforms early fusion bidirectional models (biLSTM and biGRU). Across experiments, there is a trend of higher recall. This can be justified by the fact that is the dataset is not balanced: positive examples of perpetrator mentions are significantly fewer than negative examples. This trend is consistent with the findings of Frermann et al. (2018), where an early fusion LSTM is used.

| | MODEL | pr | re | f1 |
|---|---|---|---|---|
| UNI-DIR | EF (LSTM) | 42.8 | 51.2 | 46.6 |
| | EF (GRU) | 39.4 | 60.4 | 47.7 |
| | MV (corrGRU) | 41.3 | **63.4** | **50.0** |
| BI-DIR | EF (biLSTM) | 40.0 | 62.7 | 48.8 |
| | EF (biGRU) | 43.6 | 58.1 | 49.8 |
| | MV (biCorrGRU) | **49.6** | 49.4 | 49.5 |

Table 1: Precision (pr), recall (re) and F1 scores for detecting the minority class (perpetrator mentioned) on the held-out dataset. EF stands for early-fusion, while MV for multi-view. The first section of the table reports scores for unidirectional (incremental) models and the second for bidirectional (non-incremental) models. The result for the simple unidirectional LSTM model is the one reported by Frermann et al. (2018).

| | MODALITY | | | pr | re | f1 |
|---|---|---|---|---|---|---|
| | T | I | A | | | |
| THREE | ✓ | ✓ | ✓ | 41.3 | **63.4** | **50.0** |
| TWO | ✓ | ✓ | | **41.4** | 49.6 | 45.1 |
| | ✓ | | ✓ | 39.6 | 50.4 | 44.3 |
| | | ✓ | ✓ | 38.7 | 5.1 | 9.0 |
| ONE | ✓ | | | 41.9 | 47.3 | 44.4 |
| | | ✓ | | 28.4 | 6.7 | 10.8 |

Table 2: Ablation experiment assessing the contribution of each modality for our multi-view model. Precision (pr), recall (re) and F1 scores for detecting the minority class (perpetrator mentioned) on the held-out part of the dataset are reported. The modalities are denoted by T (text), I (image) and A (audio).

**Contribution of the different modalities** We conduct an ablation experiment assessing the contribution of each modality to the final prediction. The results of this experiment can be found in Table 2. Evidently, all three modalities contribute to the good performance of the multi-view model. We note that the *text* modality is the most informative; models taking text into account score consistently better, in both multi-view and single view setups. Results of the single-view video model suggest that the *image* modality alone provides very little information about the perpetrator's identity. Results on *audio* only are not reported, since the audio modality contains only music and audio effects and is not expected to generate useful representations by itself.

**Supervised Multi-head Attention** The CSI dataset includes token-level perpetrator mention annotations: every token of the script sentences

is tagged as being a mention of the perpetrator or not. An example can be found in Figure 3 (right), where "Peter" is tagged as being a reference to the perpetrator (boldface). The two levels of annotation in the dataset follow a compositional structure: the presence of at least one token annotated with 1, results in the whole sentence annotated with 1. The sentence-level annotations used throughout the previous experimental section and throughout the work of Frermann et al. (2018) are generated by aggregating token-level annotations.

We distinguish three types of perpetrator token mentions in the dataset: *first person pronoun tokens* (the perpetrator is the speaker and speaks in first person), *pronoun tokens* (other characters refer to the perpetrator by using pronouns) and *other type of tokens* (perpetrator is mentioned by their name or other attributes). We replace the original binary token-level annotation with three binary annotation streams reflecting the three different types (first person pronoun mention, other person pronoun, other type of mention).

We replace the convolutional encoder of the previous experimental setup with an LSTM and three attention heads and run experiments comparing attentive architectures with non-attentive ones. The results can be found in Table 3. Unsurprisingly, models that make use of the extra information in the form of attention supervision score better than their counterpart that does not take token-level annotations into account. Interestingly, the complexity and diversity of token-level annotations is reflected in the results of single-head attention models: the supervised single-head model scores lower than the one that is free to learn any attention scores. Ultimately, the choice to split the annotations to three streams and use more than one attention heads, each focusing on different types of mentions, leads to better performance.

However, even the multi-head supervised attentive model, does not score as well as the multi-view (non-attentive) model. This result highlights the, sometimes disregarded, importance of modality fusion: creating a fused representation out of the available modalities led to a model that outperforms one with significantly more information in its disposal.

| MODEL | pr | re | f1 |
|---|---|---|---|
| EF | **42.8** | 51.2 | 46.6 |
| MV | 41.3 | **63.4** | **50.0** |
| EF+ATT | 39.95 | 58.70 | 47.32 |
| EF+SUPATT | 40.72 | 56.11 | 47.15 |
| EF+MULTIHEAD | 40.25 | 59.19 | 47.88 |

Table 3: Comparing the performance of early fusion (EF) and multi-view (MV) models with attentive early fusion models. Three different attention schemes are compared: simple attention (ATT), supervised attention (SUPATT) where the network's loss includes an error term for the attention scores with respect to the token-level annotations, and multi-head supervised attention (MULTIHEAD) where the token-level annotations are divided conceptually into three groups and each head is supervised by the scores of one of the groups.

### 4.3 Episode Structure Tagging

Extracting knowledge about a movie by relying on simplified tasks can be challenging and may require assumptions about the input. Specifically, casting the task of perpetrator identification as a binary classification task, is based on the premise that there is, at most, one perpetrator. This assumption does not always hold, since, some episodes contain two cases and consequently, two perpetrators. Moreover, new perpetrators are introduced in every episode and data sparsity makes multi-class classification difficult. For the experiments described in the previous sections, we alleviate this obstacle by performing binary inference on the case level, using the annotated case splits of the dataset.

In order to enable more robust movie understanding, we investigate the automatic segmentation of episodes to coherent chunks by experimenting with tagging utterances with tags of two levels of granularity: case and speaker type. The former refers to associating each utterance with the crime case it belongs to, while the latter to labeling each utterance as coming from one type of speaker (detectives, perpetrators, suspects, extras) or none (scene descriptions).

The two tagging tasks are closely related, since a shift from a speaker type (e.g. a conversation between detectives) to another (a conversation between extras) may indicate a shift in the focus of the episode, hinting a case change. The presence of more than one related tasks makes our setup ideal for testing our model in a multi-task setting. Sharing representations between tasks is justified by the notion that information from similar tasks can aid in solving the task at hand faster and more accurately (Caruana, 1998).

| MODEL | SPEAKER TYPE | | | | CASE | | | |
|---|---|---|---|---|---|---|---|---|
| | **acc** | **pr** | **re** | **f1** | **acc** | **pr** | **re** | **f1** |
| EF | 50.55 | 20.28 | 24.18 | 20.66 | 61.00 | 0.01 | 0.01 | 0.01 |
| MV | 57.66 | 19.87 | 35.95 | 25.29 | 61.65 | 0.03 | 2.86 | 0.05 |
| EF+CRF | 49.70 | 15.70 | 14.22 | 14.48 | 62.75 | 3.10 | 15.21 | 4.97 |
| MV+CRF | 51.27 | 14.89 | 16.71 | 14.96 | 73.53 | **11.72** | 27.75 | 11.24 |
| MULTI-TASK (SPEAKER+CASE) | | | | | | | | |
| EF | 45.07 | 19.83 | 27.53 | 21.82 | 61.75 | 0.00 | 0.00 | 0.00 |
| MV | 46.11 | 18.17 | 22.36 | 19.09 | 61.02 | 0.06 | 0.08 | 0.06 |
| EF+CRF | 47.04 | **22.02** | 17.42 | 18.60 | 73.95 | 1.09 | 1.52 | 1.11 |
| MV+CRF | **60.07** | 21.36 | **39.25** | **27.61** | **79.49** | 8.29 | **44.17** | **13.72** |

Table 4: Macro-average scores for accuracy (acc), precision (pr), recall (re) and F1 scores for early fusion (EF) and multi-view (MV) models on speaker type and case tagging. The top section of the table refers to single-task setup, while the bottom on multi-task setup (training jointly on speaker type and case tagging).

To facilitate tagging, we convert the case and speaker annotations of the dataset to annotations employing the BIOU (Beginning, Inside, Outside, Unit) format, derived from the BIO scheme proposed for text chunking (Ramshaw and Marcus, 1999) and heavily used in the CoNLL shared tasks[4] for sentence tagging. An example of the labels on which the model operates can be found in Figure 3. We modify the architecture of our model, so that the output of the sequence model cell is fed to different output layers, one for each task. Training proceeds by summing the loss terms for both tasks. In the case of our multi-view model, the loss consists of two cross-entropy terms and one correlation term. Moreover, we experiment with adding a Conditional Random Field (CRF) on top of the sequence models, based on recent work that achieves state-of-the-art performance in tagging tasks, such as Named Entity Recognition (Lample et al., 2016).

The results for speaker type and case tagging can be found in Table 4, where our model (MV) is compared with an LSTM early fusion model. We use a variant of the evaluation script used for the CoNLL shared tasks[5] and report average scores. Our multi-view model consistently outperforms early fusion models. Interestingly, the multi-task MV+CRF model trained exhibits the best performance, suggesting that jointly solving the two tasks improves the capabilities of the model.

[4] http://www.conll.org/previous-tasks

[5] https://github.com/spyysalo/conlleval.py

## 5 Related Work

Inference on multimodal sequences can take the form of inferring a label for a whole sequence, or a label for each of the parts of it. The multimodal LSTM of Ren et al. (2016) is applied in a sequential inference setting, however, it does not produce a joint representation for all modalities, but rather, different (albeit informed about each other) modality representations are used for inference.

Our approach is more related to that of Yang et al. (2017), where a multimodal encoder-decoder model for representation learning of temporal data is described. Our method uses their corrGRU cell with a distinct architecture and loss function: first, their network is used as an unsupervised sequence representation learning tool trained to generate an embedding for a whole sequence, which in turn is used for classification tasks, while our model outputs embeddings and makes predictions at the token-level (for every element of a sequence). Secondly, they use a decoder which reconstructs the original representations of each view, while our model does not include autoencoding.

Casting correlation maximization between three or more variables as the maximization of the sum of the correlation between all pairs of available variables has been previously used in extensions of CCA for more than two views (Benton et al., 2019), or other multi-view learning works (Kumar et al., 2011). Yang et al. (2017) mention it in their paper, although they do not experiment with it.

The multi-head attention component of our model bears similarities in spirit to the recent work of Strubell et al. (2018), where an attention head is replaced by a model trained to predict syntac-

tic dependencies (Dozat and Manning, 2017). In contrast, our model uses explicit supervision for all self-attention heads and is trained to predict the correct attention scores in a multi-task fashion.

## 6 Conclusions

We describe a neural multi-view sequential architecture, paired with a novel objective that takes advantage of supervision, while at the same time, maximizes the correlation between views. We test our approach on the task of perpetrator mention identification of the CSI dataset, on which we show that it outperforms state of the art. Also, we introduce two shallow movie understanding tasks, crime case and speaker type tagging, and show that our model yields consistently better results than early fusion models, highlighting the importance of careful fusion of modalities in sequential inference.

## Acknowledgments

The authors gratefully acknowledge the support of the European Research Council (Lapata; award number 681760, "Translating Multiple Modalities into Text"), the H2020 EU project SUMMA (award number 688139/H2020-ICT-2015; Papasarantopoulos, Cohen) and Bloomberg (Cohen).

## References

Pradeep K Atrey, M Anwar Hossain, Abdulmotaleb El Saddik, and Mohan S Kankanhalli. 2010. Multimodal fusion for multimedia analysis: a survey. *Multimedia systems*, 16(6):345–379.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations (ICLR), San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Srinivas Bangalore, Vivek Kumar Rangarajan Sridhar, Prakash Kolan, Ladan Golipour, and Aura Jimenez. 2012. Real-time incremental speech-to-speech translation of dialogs. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 437–445. Association for Computational Linguistics.

James Bennett, Stan Lanning, et al. 2007. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35. New York, NY, USA.

Adrian Benton, Huda Khayrallah, Biman Gujral, Drew Reisinger, Sheng Zhang, and Raman Arora. 2019. Deep generalized canonical correlation analysis. In *Proceedings of the 4th Workshop on Representation Learning for NLP, RepL4NLP@ACL 2019, Florence, Italy, August 2, 2019*, pages 1–6.

Hervé Bredin, Anindya Roy, Nicolas Pécheux, and Alexandre Allauzen. 2014. "Sheldon speaking, bonjour!": Leveraging multilingual tracks for (weakly) supervised speaker identification. In *Proceedings of the 22nd ACM International Conference on Multimedia*.

Ozan Caglayan, Loïc Barrault, and Fethi Bougares. 2016. Multimodal attention for neural machine translation. *arXiv preprint arXiv:1609.03976*.

Iacer Calixto, Qun Liu, and Nick Campbell. 2017. Doubly-attentive decoder for multi-modal neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1913–1924.

Rich Caruana. 1998. Multitask learning. In *Learning to learn*, pages 95–133. Springer.

Xiaobin Chang, Tao Xiang, and Timothy M Hospedales. 2018. Scalable and effective deep cca via soft decorrelation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1488–1497.

Jiajun Cheng, Shenglin Zhao, Jiani Zhang, Irwin King, Xin Zhang, and Hui Wang. 2017. Aspect-level sentiment classification with HEAT (hierarchical attention) network. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 97–106. ACM.

Kyunghyun Cho and Masha Esipova. 2016. Can neural machine translation do simultaneous translation? *arXiv preprint arXiv:1606.02012*.

Manfred Del Fabro and Laszlo Böszörmenyi. 2013. State-of-the-art and future challenges in video scene detection: a survey. *Multimedia systems*, 19(5):427–454.

Timothy Dozat and Christopher D Manning. 2017. Deep biaffine attention for neural dependency parsing. In *5th International Conference on Learning Representations, ICLR, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Lea Frermann, Shay Cohen, and Mirella Lapata. 2018. Whodunnit? crime drama as a case for natural language understanding. *Transactions of the Association for Computational Linguistics*, 6:1–15.

Chiori Hori, Takaaki Hori, Teng-Yok Lee, Kazuhiro Sumi, John R. Hershey, and Tim K. Marks. 2017. Attention-based multimodal fusion for video description. In *IEEE International Conference on Computer Vision (ICCV)*, pages 4203–4212.

Samira Ebrahimi Kahou, Xavier Bouthillier, Pascal Lamblin, Caglar Gulcehre, Vincent Michalski, Kishore Konda, Sébastien Jean, Pierre Froumenty, Yann Dauphin, Nicolas Boulanger-Lewandowski, et al. 2016. Emonets: Multimodal deep learning approaches for emotion recognition in video. *Journal on Multimodal User Interfaces*, 10(2):99–111.

Hidetaka Kamigaito, Katsuhiko Hayashi, Tsutomu Hirao, Hiroya Takamura, Manabu Okumura, and Masaaki Nagata. 2017. Supervised attention for sequence-to-sequence constituency parsing. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 7–12.

Kyung-Min Kim, Min-Oh Heo, Seong-Ho Choi, and Byoung-Tak Zhang. 2017. Deepstory: video story qa by deep embedded memory networks. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 2016–2022. AAAI Press.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Elena Knyazeva, Guillaume Wisniewski, Hervé Bredin, and François Yvon. 2015. Structured prediction for speaker identification in tv series. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Abhishek Kumar, Piyush Rai, and Hal Daume. 2011. Co-regularized multi-view spectral clustering. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 1413–1421. Curran Associates, Inc.

Chiraag Lala and Lucia Specia. 2018. Multimodal lexical translation. In *Eleventh International Conference on Language Resources and Evaluation*, pages 3810–3817, Miyazaki, Japan.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 260–270.

Cailiang Liu, Dong Wang, Jun Zhu, and Bo Zhang. 2013. Learning a contextual multi-thread model for movie/tv scene segmentation. *IEEE Transactions on Multimedia*, 15:884–897.

Chenxi Liu, Junhua Mao, Fei Sha, and Alan L Yuille. 2017a. Attention correctness in neural image captioning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 4176–4182.

Shulin Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2017b. Exploiting argument information to improve event detection via supervised attention mechanisms. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1789–1798.

Haitao Mi, Zhiguo Wang, and Abe Ittycheriah. 2016. Supervised attentions for neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2283–2288.

Xavier Anguera Miró, Simon Bozonnet, Nicholas W. D. Evans, Corinne Fredouille, Gerald Friedland, and Oriol Vinyals. 2012. Speaker diarization: A review of recent research. *IEEE Transactions on Audio, Speech, and Language Processing*, 20:356–370.

Louis-Philippe Morency, Rada Mihalcea, and Payal Doshi. 2011. Towards multimodal sentiment analysis: Harvesting opinions from the web. In *Proceedings of the 13th International Conference on Multimodal Interfaces*, pages 169–176. ACM.

Mihalis A Nicolaou, Hatice Gunes, and Maja Pantic. 2011. Continuous prediction of spontaneous affect from multiple cues and modalities in valence-arousal space. *IEEE Transactions on Affective Computing*, 2(2):92–105.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Shyam Sundar Rajagopalan, Louis-Philippe Morency, Tadas Baltrusaitis, and Roland Goecke. 2016. Extending long short-term memory for multi-view structured learning. In *European Conference on Computer Vision*, pages 338–353. Springer.

Lance A Ramshaw and Mitchell P Marcus. 1999. Text chunking using transformation-based learning. In *Natural language processing using very large corpora*, pages 157–176. Springer.

James M Rehg, Gregory D Abowd, Agata Rozga, Mario Romero, Mark A Clements, Stan Sclaroff, Irfan Essa, Opal Y Ousley, Yin Li, Chanho Kim, et al. 2013. Decoding children's social behavior. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 3414–3421. IEEE.

Jimmy SJ Ren, Yongtao Hu, Yu-Wing Tai, Chuan Wang, Li Xu, Wenxiu Sun, and Qiong Yan. 2016. Look, listen and learn-a multimodal lstm for speaker identification. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*.

Anna Rohrbach, Atousa Torabi, Marcus Rohrbach, Niket Tandon, Christopher Joseph Pal, Hugo Larochelle, Aaron C. Courville, and Bernt Schiele.

2016. Movie description. *International Journal of Computer Vision*, 123:94–120.

Yue Shi, Martha Larson, and Alan Hanjalic. 2013. Mining contextual movie similarity with matrix factorization for context-aware recommendation. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 4(1):16.

Michael A Smith and Takeo Kanade. 1998. Video skimming and characterization through the combination of image and language understanding. In *Content-Based Access of Image and Video Database, 1998. Proceedings., 1998 IEEE International Workshop on*, pages 61–70. IEEE.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. pages 5027–5038.

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. 2017. Inception-v4, Inception-ResNet and the impact of residual connections on learning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, volume 4, page 12.

Makarand Tapaswi, Yukun Zhu, Rainer Stiefelhagen, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2016. Movieqa: Understanding stories in movies through question-answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4631–4640.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Weiran Wang, Raman Arora, Karen Livescu, and Jeff Bilmes. 2015. On deep multi-view representation learning. In *International Conference on Machine Learning*, pages 1083–1092.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057.

Xitong Yang, Palghat Ramesh, Radha Chitta, Sriganesh Madhvanath, Edgar A. Bernal, and Jiebo Luo. 2017. Deep multimodal representation learning from temporal data. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Xiwang Yang, Harald Steck, and Yong Liu. 2012. Circle-based recommendation in online social networks. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1267–1275. ACM.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 1480–1489.

Mahsa Yarmohammadi, Vivek Kumar Rangarajan Sridhar, Srinivas Bangalore, and Baskaran Sankaran. 2013. Incremental segmentation and decoding strategies for simultaneous translation. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1032–1036.

Youngjae Yu, Jongwook Choi, Yeonhwa Kim, Kyung Yoo, Sang-Hun Lee, and Gunhee Kim. 2017. Supervising neural attention models for video captioning by human gaze data. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Honolulu, Hawaii*, pages 2680–29.

Amir Zadeh, Paul Pu Liang, Navonil Mazumder, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. 2018a. Memory fusion network for multi-view sequential learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5634–5641.

Amir Zadeh, Paul Pu Liang, Soujanya Poria, Prateek Vij, Erik Cambria, and Louis-Philippe Morency. 2018b. Multi-attention recurrent network for human communication comprehension. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5642–5649.

# Guiding the Flowing of Semantics: Interpretable Video Captioning via POS Tag

**Xinyu Xiao[1,2], Lingfeng Wang[1], Bin Fan[1], Shiming Xiang[1,2], Chunhong Pan[1]**
[1]NLPR, Institute of Automation, Chinese Academy of Sciences
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences
{xinyu.xiao, lfwang, bfan, smxiang, chpan}@nlpr.ia.ac.cn

## Abstract

In the current video captioning models, the video frames are collected in one network and the semantics are mixed into one feature, which not only increase the difficulty of the caption decoding, but also decrease the interpretability of the captioning models. To address these problems, we propose an Adaptive Semantic Guidance Network (ASGN), which instantiates the whole video semantics to different POS-aware semantics with the supervision of part of speech (POS) tag. In the encoding process, the POS tag activates the related neurons and parses the whole semantic information into corresponding encoded video representations. Furthermore, the potential of the model is stimulated by the POS-aware video features. In the decoding process, the related video features of noun and verb are used as the supervision to construct a new adaptive attention model which can decide whether to attend to the video feature or not. With the explicit improving of the interpretability of the network, the learning process is more transparent and the results are more predictable. Extensive experiments demonstrate the effectiveness of our model when compared with state-of-the-art models.

## 1 Introduction

Video captioning, which transforms the semantic information in a video to a natural statement, has received wide attention recently. The series of scenes (both related and unrelated) in video frames bring a huge challenge for the task of video captioning. Therefore, mastering the ability to process the correlated and irrelevant semantic information can improve the performance and interpretability of the model of video captioning.

The classical deep learning based video captioning methods (Venugopalan et al., 2015a,b) incorporate both CNN and LSTM together as an encoder-decoder architecture which extracts all the semantic information in one stream and fuses them in a single feature. In the architecture, the transformation of the semantics to words prediction is uncertainty. Some other recent studies (Dong et al., 2017; Wu et al., 2018) attempt to improve the interpretability of the transformation process. But the flowing of different semantic information in the network streams and the activated neurons in network are still ambiguous. To address these issues, the first is to take the discrimination of different semantic information into consideration. According to (Horoufchin et al., 2018; Fargier and Laganaro, 2015), nouns and verbs typically describe concrete objects and actions which can recruit the canonical neurons system to activate corresponding representation patterns in brain. This indicates that differential neurons in brain are activated for lexical selection of action and object words. Therefore, the part of speech (POS) tag can be applied to guide the flowing of different semantics into the corresponding network streams.

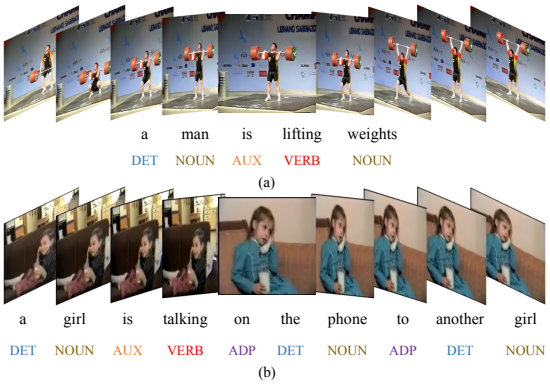


Figure 1: Examples of video description and its POS tags. The definition of POS tag can be seen in Section 3.1. From these examples, it can be seen that the words that are belonging to the same POS tags undergo inflection for similar semantic properties and different POS tags reflect different semantic properties distinctly.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2068–2077, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

A POS tag is the category of words which has similar grammatical properties. The words assigned to the same POS tag generally reflect similar properties within the grammatical structure of sentence. Fig.1 (a) shows a video and its caption "a man is lifting weights". The nouns "man", "weights" and the verb "lifting" are belonging to different POS tags and referring to different semantic information in video. Moreover, in Fig. 1 (b), the caption of video is "a girl is talking on the phone to another girl". The words of "girl", "talking" and "phone" distinctly have their corresponding visual signals in the video, but the others are uncertain. This example indicates that the nouns of objects and verbs of actions are generally referred to visual words. Another property of the POS tag is that when given a fixed sentence, the POS tags of all the words in the sentence are fixed. It ensures the reliability of the POS tag in helping to extract and guide corresponding semantics.

According to Merity et al. (Merity et al., 2016), the prediction of the next word not always need to attend the visual feature. The gradients from non-visual words could mislead and diminish the effectiveness of the visual signal in guiding caption generation. To this end, Lu et al. (Lu et al., 2016) proposed a "visual sentinel", which applies the hidden state in LSTM as supervision to adaptively decide if it is necessary to input visual feature to the language model when generating the next word. However, the supervision information from the hidden state isn't credible which cannot make sure it contains the corresponding visual decision signals. According to the properties of the POS tag, the nouns and verbs can be applied as the supervision to distinguish whether the remaining words in the sentence are visual words or not.

In this paper, to explicitly improve the interpretability of video captioning model, we propose an Adaptive Semantic Guidance Network (ASGN), which instantiates the whole video semantics by part of speech (POS) tags to different POS-aware semantics. At first, a POS-aware semantic guider is proposed. It predicts the POS tags of the words in descriptions, and guides different POS-aware semantic information of video into corresponding network streams by the predicted POS tags. In this process, the specific CNN neurons are activated under the supervision of POS tags and the whole visual semantics are parsed into POS tags related video features. Moreover, depending on the POS-aware video features, a new adaptive attention operation is introduced. The video features related to noun and verb are used as a supervision to get a sentinel gate, which decides how much the attended feature can be imported into the decoder LSTM when generating the next word. A reinforcement learning (RL) method is applied to optimize our model which further demonstrates the validity of our method on the video captioning task.

The main contributions of this paper are:

- Designing a POS-aware semantic guider to predict the POS tags of words and guide different semantic information of video into corresponding network streams. Under the supervision of POS tags, the CNN neurons are selectively activated and aware of the related POS tags, so that the whole video semantics are parsed into the corresponding POS-aware video features.

- Depending on the noun and verb instantiated video features, a new adaptive attention model is constructed to decide how much the visual feature is imported into the decoder.

- Due to the guiding of POS tags, the flowing of the type of semantic in which network stream can be easily clarified. With the supervision of the noun and verb related features, the judgment of the predicted word is visual word or not is more reasonable. These make the learning process more interpretable while achieving state-of-the-art performance.

## 2 Related Works

Here, we first review the recent implements of the POS tag in computer vision, then review the most relevant works on video description task like attention-based methods and interpretable improved models.

The POS tag has been received attention in some computer vision tasks, like visual question answering (Wang et al., 2018b) and image captioning (A et al., 2019). He et al. (He et al., 2017) utilized the POS tag of each word to determine whether it is essential to input image representation into the word generator. Wang et al. (Wang et al., 2018b) exploited the POS tag guided attention model in VQA to put more emphasis on the important words such as nouns, verbs and adjectives. All these methods realized the importance

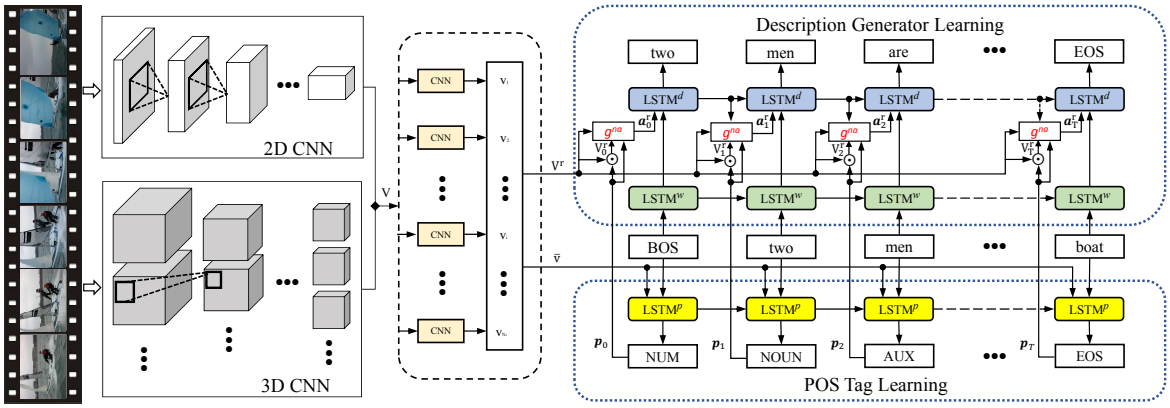


Figure 2: The architecture of our Adaptive Semantic Guidance Network (ASGN) which can shunt the required semantic information into different video features when generating every word in caption. The features $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{N_s}\}$ are extracted by the CNN modules. $\odot$ is the hadamard product module. $g^{na}$ is the new adaptive attention model. A single-layer LSTM is set as the POS tag decoder and a two-layer stacked LSTM is set as the language model. The BOS and EOS denote the begin-of-sentence and end-of-sentence, respectively.

of the POS tag in the linguistic computer vision. However, they ignore the important property of POS tag, which relates to different visual semantics with different types of POS tag.

Attention-based methods have been widely used in visual captioning models. Yao et al. (Yao et al., 2015) considered the temporal structure of video and proposed a temporal attention mechanism to generate descriptions. Lu et al. (Lu et al., 2016) proposed an adaptive attention model in image captioning which can decide either to look at the image or to rely on the context of sentence to generate the next word.

Because of the highly nonlinearity and unclear working mechanism of neural networks, the operational processes of neural networks are always treated as black-box processes. For the video description task, some researchers (Dong et al., 2017; Wu et al., 2018) attempted to improve the interpretability of video description models. Dong et al. (Dong et al., 2017) interpreted the learned features of each neuron by a wide range of visual concepts in the video description task. Wu et al. (Wu et al., 2018) considered both the motion information and the sentence semantic structure with an attentive structured localization mechanism to enhance the captioning model's interpretability.

In this paper, we find the POS tag can be employed as a supervision to process irrelevant or relevant semantic information in video description task. Under the supervision of POS tag, an Adaptive Semantic Guidance Network (ASGN) is proposed to guide different POS-aware semantic information of video into corresponding network streams. Moreover, the video features related to noun and verb are used to get a sentinel gate which can decide how much the attended feature can be imported to the decoding process.

## 3 Our Method

In this section, we describe our ASGN in detail. First, the POS-aware semantic guider with the POS tag learning model is introduced. Next, an adaptive attention model which constructs a new sentinel gate will be presented. Finally, the description generator and its learning methods are introduced.

### 3.1 POS-Aware Semantic Guider

The key of the proposed POS-aware semantic guider is the POS tag, which is used to guide different semantic information of video into corresponding network streams. Supposing that we have a video described by a textual sentence $\mathcal{S}$, which consists of $T$ words. The POS tag set of each word in $\mathcal{S}$ is defined as $\mathcal{P}$. To achieve this, a POS tag learning model is designed to predict the POS tags of the words in description.

We refer to (Al-Rfou et al., 2013) and annotate the POS tags of captions in training set by the polyglot toolkit[1]. Polyglot toolkit is a natural language pipeline that supports massive multilingual applications, including the POS tag identification. According to the setting of the polyglot toolkit, there are seventeen categories of POS tags: noun (NOUN), verb (VERB), adjective (ADJ),

[1] http://polyglot.readthedocs.org

adposition (ADP), adverb (ADV), auxiliary verb (AUX), coordinating conjunction (CONJ), determiner (DET), interjection (INTJ), numeral (NUM), particle (PART), pronoun (PRON), proper noun (PROPN), punctuation (PUNCT), subordinating conjunction (SCONJ), symbol (SYM) and other unknown types (X). Different POS tags have different descriptions or embellishments. Following the universal set of this toolkit, the number of POS tag categories is defined as $N_s = 17$.

Based on these POS tag categories, an encoder-decoder framework is proposed to predict the specific POS tag of each word in sentences, which can be seen in Fig. 2. Specifically, the video feature $\mathbf{V}$ is concatenated from the extractions of the 2D CNN and 3D CNN. Then, $\mathbf{V}$ is flowed into $N_s$ semantic guiding CNN modules. Each CNN module relates to a corresponding POS tag category. The outputs of these CNN modules are $N_s$ video features $\mathbf{v}_i$, where $i = 1, \ldots, N_s$. The mean-pooled feature $\bar{\mathbf{v}} = \frac{1}{N_s}\sum_{i=1}\mathbf{v}_i$ and the sentence $\mathcal{S} = \{\boldsymbol{s}_0, \boldsymbol{s}_1, \ldots, \boldsymbol{s}_T\}$ are taken as the inputs to the POS tag decoder LSTM$^p$, where $\boldsymbol{s}_0$ is defined as the begin-of-sentence (BOS). All the POS tags of words in caption are sequentially generated by LSTM$^p$. The hidden state $\boldsymbol{h}_t^p$ is updated at time step $t \in \{0, \ldots, T\}$ through:

$$\boldsymbol{h}_t^p = \text{LSTM}^p(\mathbf{W}_e^p \boldsymbol{s}_t, \bar{\mathbf{v}}, \boldsymbol{h}_{t-1}^p), \tag{1}$$

where $\mathbf{W}_e^p \in \mathbb{R}^{N_h \times V}$ is the word embedding matrix, $N_h$ is the length of hidden state and $V$ denotes the vocabulary size of the corresponding dataset's text library.

Given the ground-truth POS tags which are annotated to the corresponding sentence. Therefore, the associated POS tags of the $k$-th video $\mathbf{V}$ are $\mathcal{P}^k = \{p_0^k, \ldots, p_T^k\}$. We define the POS tag learning loss as:

$$L_p = -\frac{1}{N}\sum_{k=1}^{N}\sum_{t=0}^{T}\log(p(p_t^k | p_{0:t-1}^k, \bar{\mathbf{v}}^k)), \tag{2}$$

where $N$ is the number of training examples. In the encoder-decoder framework, the POS tag probability vector $\boldsymbol{p}_t$ is predicted at time step $t$. The mapping function is $\boldsymbol{p}_t = f(\boldsymbol{h}_t^p) = \text{Softmax}(\text{FC}(\boldsymbol{h}_t^p))$, where $f : \boldsymbol{h}_t^p \rightarrow \boldsymbol{p}_t$ is a Softmax mixed function and FC is a full connection layer. The predicted POS tag has the maximum probability in $\boldsymbol{p}_t$. Similar to visual captioning, the learned POS-aware semantic guider applies each predicted word to predict the POS tag of the next word in testing.

The learned $\boldsymbol{p}_t$ is used to guide the flowing of semantics from the $N_s$ CNN modules at time step $t$ . The $i$-th CNN module is related to the $i$-th POS tag category. The outputs of the CNN modules are concatenated as $\mathbf{V}^r = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{N_s}]$, where $\mathbf{v}_i$ is generated by the specific neuron of the $i$-th CNN module. The POS tag vector $\boldsymbol{p}_t$ is applied to activate the specific CNN module at the $t$-th step. The operation is implemented by a hadamard product module on the channel level:

$$\mathbf{V}_t^r = \mathbf{V}^r \odot \boldsymbol{p}_t, \tag{3}$$

where $\mathbf{V}_t^r$ represents the video representation at time step $t$. Then the video feature $\mathbf{V}_t^r$ which contains relevant visual semantic information will be inputted into the language model at time step $t$.

Generally, the learned POS tag representations can be used to activate the specific neurons of the CNN encoder and parse the whole video representation to guide semantic information into corresponding feature representations at each time step. Compared with the normal CNN+LSTM model, our POS-aware semantic guider constructs a correlation between different types of POS-aware semantic and the corresponding CNN modules.

### 3.2 Adaptive Attention Model

Although, Lu et al. (Lu et al., 2016) proposed a sentinel to decide whether the predicted words are visual words or not. Their model is learning from the gradient of back propagation, which is still an ambiguous process. The video features related to different POS tags have different properties. For example, the video features related to noun and verb always have sufficient visual signals. According to the properties of POS tags, we propose a more credible adaptive spatial attention model to predict next word. Specifically, the related video features of noun and verb are applied as the supervision to get a sentinel gate. Through the sentinel gate, when generating the next word, we can distinguish its visual word or not and decide how much the attended feature can be imported into the decoder LSTM.

First, the video feature $\mathbf{V}_t^r$ is reshaped to $\mathbf{V}_t^r = [\boldsymbol{v}_{t1}^r, \boldsymbol{v}_{t2}^r, \ldots, \boldsymbol{v}_{tm}^r]$, where $m$ is the value of the width times height of $\mathbf{V}_t^r$. Normal attention model is defined as $g^a(\mathbf{V}_t^r, \boldsymbol{h}_{t-1}^d)$. $\boldsymbol{h}_{t-1}^d$ is generated by the description decoder LSTM$^d$ which is shown in

Fig. 2. Formally, for the $t$-th time step, the attention part of the model $g^a$ is defined as follows:

$$\begin{aligned} \boldsymbol{z}_t^r &= \tanh((\mathbf{W}^r \mathbf{V}_t^r + \boldsymbol{b}^r) \oplus \mathbf{W}^{hr} \boldsymbol{h}_{t-1}^d), \\ \boldsymbol{\alpha}_t^r &= softmax(\mathbf{W}^{\alpha r} \boldsymbol{z}_t^r + \boldsymbol{b}^{\alpha r}), \end{aligned} \tag{4}$$

where $\mathbf{W}^r \in \mathbb{R}^{l \times (N_c * N_s)}$, $\mathbf{W}^{hr} \in \mathbb{R}^{l \times N_h}$, $\mathbf{W}^{\alpha r} \in \mathbb{R}^{l \times 1}$ are the transformation matrices that map the CNN feature and hidden state to the same size; $N_c$ is the channel size of $\mathbf{v}_i$; $\boldsymbol{b}^r \in \mathbb{R}^l$ and $\boldsymbol{b}^{\alpha r} \in \mathbb{R}^1$ are the model biases. $\boldsymbol{\alpha}_t^r$ is the attention weight related to $\mathbf{V}_t^r$.

Second, to improve the structure of $g^a$, our model learns to use the related video features of noun and verb as a supervision to distinguish whether the words in a sentence are visual words or not. All of the $N_s$ video features are extracted after the ReLU operation, and thus the values in these feature maps are not less than zero. It indicates that the value of the feature map can reflect the response activation of the corresponding CNN module. We assign the related feature maps of noun and verb as the reference and obtain a value $R = \text{mean}[\frac{1}{2m} \sum_{j=1}^{m} (\mathbf{v}_n + \mathbf{v}_v)]$, where $\mathbf{v}_n$ and $\mathbf{v}_v$ are the related features of noun and verb from $\mathbf{V}^r$. Depending on the attended visual feature, a formulae is introduced to ascertain whether the current $t$ word is a visual word or not:

$$c_t^v = \sum_{i=1}^{N_s} \{p_t^i \quad \text{if mean}(\sum_{j=1}^{m} \alpha_{tj}^r \boldsymbol{v}_j^{ri}) \geq R\}, \tag{5}$$

where $c_t^v$ is the weight of visual word at time step $t$, $p_t^i$ is $i$-th value in the $t$-th POS tag representation $\boldsymbol{p}_t$. Therefore, the weight of non-visual word is $c_t^{nv} = 1 - c_t^v$. In our method, the sentinel gate is defined as:

$$\beta_t = \frac{1}{1 + \exp(-log(\frac{c_t^v}{c_t^{nv}}))}. \tag{6}$$

The design of the sentinel gate can avoid the gate value of $\beta_t$ being too small or too large. Based on $\beta_t$, the new adaptive attention model $g^{na}$ can decide how much the attended feature can be imported into the decoder LSTM as follows:

$$\boldsymbol{a}_t^r = \beta_t g^a = \beta_t \sum_{j=1}^{m} \alpha_{tj}^r \boldsymbol{v}_{tj}^r. \tag{7}$$

### 3.3 Description Generator

In the description generation stage, we adopt a stacked two-layer LSTM to generate captions, namely LSTM$^w$ and LSTM$^d$. Following (Donahue et al., 2015), the first LSTM layer LSTM$^w$ is applied to encode the inputted sentence to enhance the textual context information of each word vector $\boldsymbol{h}_t^w$.

In the decoding stage, the encoded word vector $\boldsymbol{h}_t^w$ and the processed video feature $\boldsymbol{a}_t^r$ are taken as the inputs at $t$ time step. The updating procedure from 0 to $T$ of LSTM$^d$ is written below:

$$\boldsymbol{h}_t^d = \text{LSTM}^d(\boldsymbol{h}_t^w, \boldsymbol{a}_t^r, \boldsymbol{h}_{t-1}^d), \tag{8}$$

where $\boldsymbol{h}_t^d$ is the current output of LSTM$^d$ at time step $t$, the $\boldsymbol{h}_{-1}^d$ can be set as a null vector.

#### 3.3.1 Description Generator Learning

**Maximum Likelihood Estimation**: Given the $k$-th video $\mathbf{V}^k$ and the associated sentence $\mathcal{W}^k = \{w_0^k, \ldots, w_T^k\}$, the generator loss can be formulated as follows with the optimization of maximum likelihood estimation (MLE):

$$L_s = -\frac{\lambda}{N} \sum_{k=1}^{N} \sum_{t=0}^{T} \log(p(w_t^k | w_{0:t-1}^k, \mathbf{V}^k)), \tag{9}$$

where $p(w_t^k | w_{0:t-1}^k, \mathbf{V}^k)$ is obtained from a Softmax mixed function; $\lambda$ is the hyper-parameter.

**Policy Gradient Optimization**: For a fair comparison with recent works, the policy gradient (PG) technique is adopted as the optimizer to training our model. The objective in learning is to minimize the negative expected reward of the complete sampled sentence $\mathcal{W}^s = \{w_0^s, \ldots, w_T^s\}$:

$$L_\theta = E_{\mathcal{W}^s \sim p_\theta}[r(\mathcal{W}^s)], \tag{10}$$

where $r(\mathcal{W}^s)$ is calculated by comparing sampled caption with the reference caption in the specified evaluation metric. Following the implementation in (Rennie et al., 2017), we apply a single Monte-Carlo sample to calculate the relative reward $\Delta r(\mathcal{W}^s)$, which is computed by a baseline reward $b$. $b$ is obtained by performing greedy decoding:

$$\begin{aligned} b = r(\hat{\mathcal{W}}),\ \hat{\mathcal{W}} = \ arg\ max\ p(w_t | \boldsymbol{h}_t^d), \\ \nabla_\theta L_\theta \approx -(r(\mathcal{W}^s) - r(\hat{\mathcal{W}})) \nabla_\theta \log[p_\theta(\mathcal{W}^s)]. \end{aligned} \tag{11}$$

Table 1: Ablation studies of our proposed video captioning model on Youtube2Text and MSR-VTT datasets. The (RL) indicates this model is optimized by the reinforcement learning. The best results are in bold.

| Model | Youtube2Text | | | | MSR-VTT | | | |
|---|---|---|---|---|---|---|---|---|
| | B-4 | M | R | C | B-4 | M | R | C |
| ASGN | 0.494 | 0.325 | 0.690 | 0.733 | 0.271 | 0.266 | 0.587 | 0.439 |
| ASGN+L | 0.501 | 0.329 | 0.699 | 0.762 | 0.391 | 0.268 | 0.595 | 0.449 |
| ASGN+LA | 0.514 | 0.331 | 0.696 | 0.758 | 0.384 | 0.269 | 0.602 | 0.448 |
| ASGN+LNA | 0.518 | 0.333 | 0.700 | 0.776 | 0.395 | 0.274 | 0.609 | 0.465 |
| ASGN+LNA (RL) | **0.521** | **0.333** | **0.703** | **0.803** | **0.405** | **0.278** | **0.615** | **0.490** |

# 4 Experiments

## 4.1 Dataset and Evaluation

We report the results of our method on the Youtube2Text (Guadarrama et al., 2013) and MSR-VTT (Xu et al., 2016) datasets. The Youtube2Text dataset contains 1970 YouTube video clips. According to the publicly provided splits (Venugopalan et al., 2015b), 1200 videos are used for training, 100 videos for validation and the rest are used for testing. MSR-VTT is the largest public dataset for video captioning up to now. We follow the public splits (Venugopalan et al., 2015a) and divide them into 6,513, 497 and 2,990 samples for training, validation and testing, respectively. We reserve the words that appear in the training set and yield two vocabularies which contain 12,182 and 16,630 words for Youtube2Text and MSR-VTT datasets, respectively.

In evaluation, we report the following metrics: B-N (N=2,3,4) (Papineni et al., 2002), Meteor (Banerjee and Lavie, 2005), Rouge-L (Lin, 2004) and CIDEr (Vedantam et al., 2015). All the metrics are computed by the MS-COCO caption evaluation tool[2].

## 4.2 Training Details

**CNN Encoder**: For the video representations, we use a 2D CNN and a 3D CNN as the CNN encoder collectively. The 3D CNN can operate all video frames as a whole, which ensures the extracted visual features contain all the semantic information. The 2D CNN has more efficient learning and representation capacity. The details of these two CNNs can be seen as follows:

- **2D CNN** We use ResNet-152 (He et al., 2016) as the 2D CNN model. The feature map is taken from the *res5c* layer (2,048-dim).
- **3D CNN** The C3D (Tran et al., 2015) is applied as the 3D CNN model. The feature map is extracted from the *conv5b* layer.

The equally-spaced 16 and 32 frames are sampled from one video clip for Youtube2Text and MSR-VTT, respectively. We perform a mean operation among all the 2D CNN features. The representation $\mathbf{V}$ of each video is composed by a concatenation of the 3D CNN feature and the 2D CNN feature. Then, the feature map $\mathbf{V}$ is processed by $N_s$ semantic separating CNN modules. A residual block is adopted as the CNN module in our method. The hidden state dimension of the LSTM units is 1,024.

The Adam optimizer is adopted in training. We first train the POS tag learning model in the POS-aware semantic guider. In the later training, the parameters of the POS tag learning model are fixed. The other parameters of our model is learned with MLE. $\lambda$ is set to be 1. The maximum number of epochs of the MLE training is 30. The RL method is applied to optimize the MLE trained model with the CIDEr metric. At each epoch, the validation set is used to evaluate the training model, and the best CIDEr score model is selected for the final testing. All of our experiments are implemented with Pytorch (Paszke et al., 2017) and are conducted on a Titan X GPU with 12G memory.

In caption testing, the beam search is adopted for caption generation. The search size is set to be 5 in experiments.

## 4.3 Compared Approaches

Our method is compared with **HRNE** (Pan et al., 2016), **VideoLAB** (Ramanishka et al., 2016), **SA-LSTM** (Wang et al., 2018a) and attention-based methods like **MA** (Hori et al., 2017), **SCN** (Gan et al., 2016) and recently proposed **PickNet** (Chen et al., 2018) and state-of-the-art RL optimized methods **Weakly** (Shen et al., 2017), **CIDEnt-RL**

[2] https://github.com/tylin/coco-caption

Table 2: Performance compared with state-of-the-art methods on Youtube2Text dataset. The (−) is an unknown metric.

| Model | B-4 | M | R | C |
|---|---|---|---|---|
| LSTM-I | 0.446 | 0.297 | – | – |
| HRNE | 0.438 | 0.331 | – | – |
| MA | 0.504 | 0.318 | – | 0.699 |
| SCN | 0.511 | 0.335 | – | 0.777 |
| TSA | 0.517 | 0.340 | – | 0.749 |
| SA-LSTM | 0.523 | 0.341 | 0.698 | 0.803 |
| PickNet | 0.523 | 0.333 | 0.696 | 0.765 |
| ASGN+LNA | **0.547** | **0.342** | **0.717** | **0.813** |

Table 3: Performance compared with state-of-the-art methods on MSR-VTT dataset.

| Model | B-4 | M | R | C |
|---|---|---|---|---|
| VideoLAB | 0.391 | 0.277 | 0.606 | 0.441 |
| SA-LSTM | 0.391 | 0.266 | 0.593 | 0.427 |
| Weakly | 0.414 | 0.283 | 0.611 | 0.489 |
| CIDEnt-RL | 0.405 | 0.284 | 0.614 | **0.517** |
| HRL | 0.413 | **0.287** | 0.617 | 0.480 |
| PickNet | 0.413 | 0.277 | 0.598 | 0.441 |
| ASGN+LNA | **0.420** | 0.282 | **0.621** | 0.505 |

(Pasunuru and Bansal, 2017), **HRL** (Wang et al., 2017). Moreover, we compare our method with interpretable improvement methods **LSTM-I** (Dong et al., 2017) and **TSA** (Wu et al., 2018).

### 4.4 Ablation Study

We perform the ablation studies on the Youtube2Text and MSR-VTT datasets for our video captioning model. The results are shown in Table 1. **ASGN** which imports video feature into different network streams without the POS tag guidance is adopted as the baseline model. **ASGN+L** predicts the POS tag of each word and applies the POS tag to guide the semantic separation. "L" denotes the POS-aware semantic guider. **ASGN+LA** adds an attention model proposed by Lu et al. (Lu et al., 2016). As a comparison, our proposed new adaptive attention model is introduced to **ASGN+LNA**.

It can be seen that the ASGN+LNA achieves the best performances in all metrics, which indicates our proposed sentinel gate is more effectiveness and reasonable to decide the quantity of attended feature to the decoder LSTM. Compared with the baseline model, the introduction of POS tag in ASGN+L, which activates the specific neurons and parses the whole visual representation of video, can assign appropriate POS-aware semantic information and achieve better performance. Comparing with the results of MLE-based and RL-based methods, the RL method can improve the performance of MLE-based model by significant margins across all metrics.

Table 4: Human evaluation between ASGN and ASGN+L models.

| Indifference | ASGN+L Wins | ASGN Wins |
|---|---|---|
| 0.418 | 0.329 | 0.253 |

Table 5: The results under different supervisions of POS tags on Youtube2Text dataset.

| Model | B-4 | M | R | C |
|---|---|---|---|---|
| A+A | 0.489 | 0.325 | 0.695 | 0.756 |
| V | 0.506 | 0.330 | 0.699 | 0.762 |
| N | 0.509 | 0.330 | 0.701 | 0.773 |
| N+V | 0.518 | 0.333 | 0.700 | 0.776 |

### 4.5 Quantitative Analysis

In Table 2 and Table 3, we compare our ASGN+LNA model with the state-of-the-art models on the Youtube2Text and MSR-VTT datasets. Following the operation of (Gan et al., 2016; Pasunuru and Bansal, 2017), ASGN+LNA is the average ensemble of 5 ASGN+LNA (RL) models trained with different initializations. From the results, our method achieves the competitive performance on the two datasets. Compared with the other interpretable improvement methods (Dong et al., 2017; Wu et al., 2018), interpretability of our neural network is explicitly improved, and the performance of our model is more competitive.

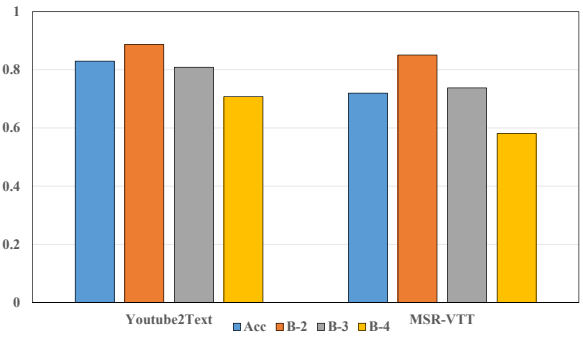


Figure 3: The performance of the POS tag prediction.

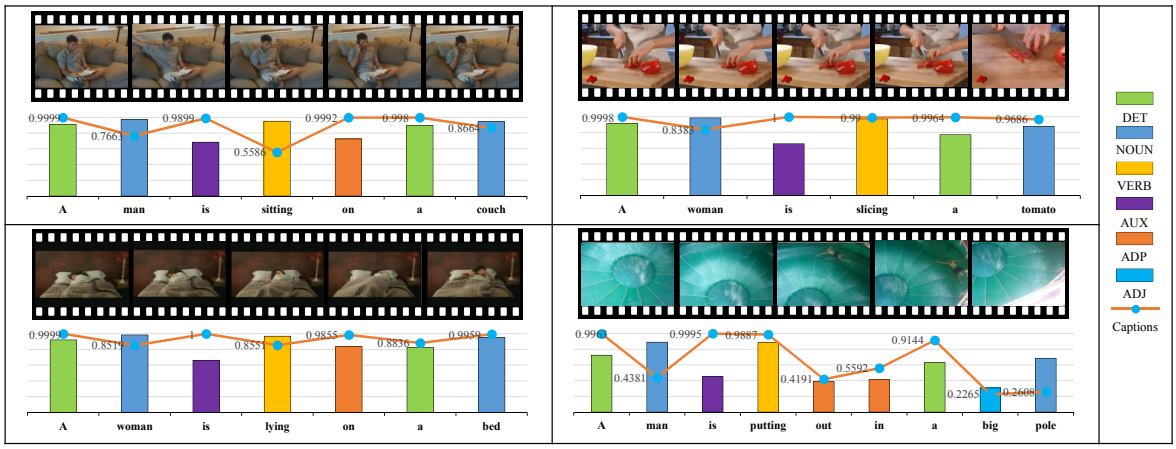


Figure 4: Visualizations of the probabilities of each word and corresponding POS tag in caption.

We introduce the human evaluation from (Pasunuru and Bansal, 2017) for comparison between ASGN and ASGN+L models. The relevance measures how related is the generated caption w.r.t, to the video content is adopted as the metric. In Table 4, the results of 150 samples from the Youtube2Text test set are studied. It can be found that the proposal of the semantic guider significantly improves the semantic extraction ability of network.

Moreover, to better verify the reliability of the supervision of nouns and verbs, we add comparisons by adjusting the supervision to single nouns (N), single verbs (V), adjectives and adverbs (A+A), nouns and verbs (N+V), respectively. The results on the Youtube2Text are presented in Table 5. It can be found that the model under the supervision of V+N achieves the best performance. Compared to verbs, the model under the supervision of nouns is more reliable. Under the supervision of A+A, the results of the model indicate the words of adjectives or adverbs are not always related to visual signals.

### 4.6 Visualized Analysis

To examine the reliability of the POS tag prediction, the performance of the POS tag learning model is measured using accuracy (Acc) and B-N (N=2,3,4) over the test datasets of Youtube2Text and MSR-VTT. The results are shown in Fig. 3. The metric of Acc is to test the total predict performance and the metric of B-N is to test the continuity of prediction. To better illustrate the effectiveness of the model, we set the beam search size to be 1 which is the same with the training process. These results indicate that the POS tag learning model can provide reliable POS tag representations to guide the semantics' separation to corresponding network streams.

Fig. 4 gives some results of generated captions and the corresponding probability of words and POS tags. In Fig. 4, we can see that the POS-aware semantic guider successfully guides the POS-aware semantic to the generation of captions. The POS-aware neurons have higher probability are activated to extract corresponding semantics to predict the relevant words at each time step. In the first example, the POS tags of the generated words with the highest probability have high probability as well. It demonstrates the improvement of interpretability of our model.

To further present the interpretability of our model, the neuron activations associated with the POS tags, the weights of the sentinel gate, the generated video captions, and the real POS tags of the captions are visualized in Fig. 5. Each element of these samples is illustrated along with the word prediction in sequence. Through these examples, the process of the POS-aware semantic information extraction and guidance in corresponding network stream can be visualized explicitly. From the illustration between the activated neuron and the truth POS tags of the captions. It can be found that the corresponding POS-aware neurons have high activations through time. For example, in Fig. 5 (b), the "DET", "NOUN" and "VERB" are precisely pointed to the POS tag of the words in "a man is flying into the water". It illustrates that the corresponding neurons of the POS tags are successfully activated at each time step. Moreover, the illustrations between the weights of the sentinel gate and the generated captions reveal that our adaptive attention model can effectively capture both the visual words and non-visual words.

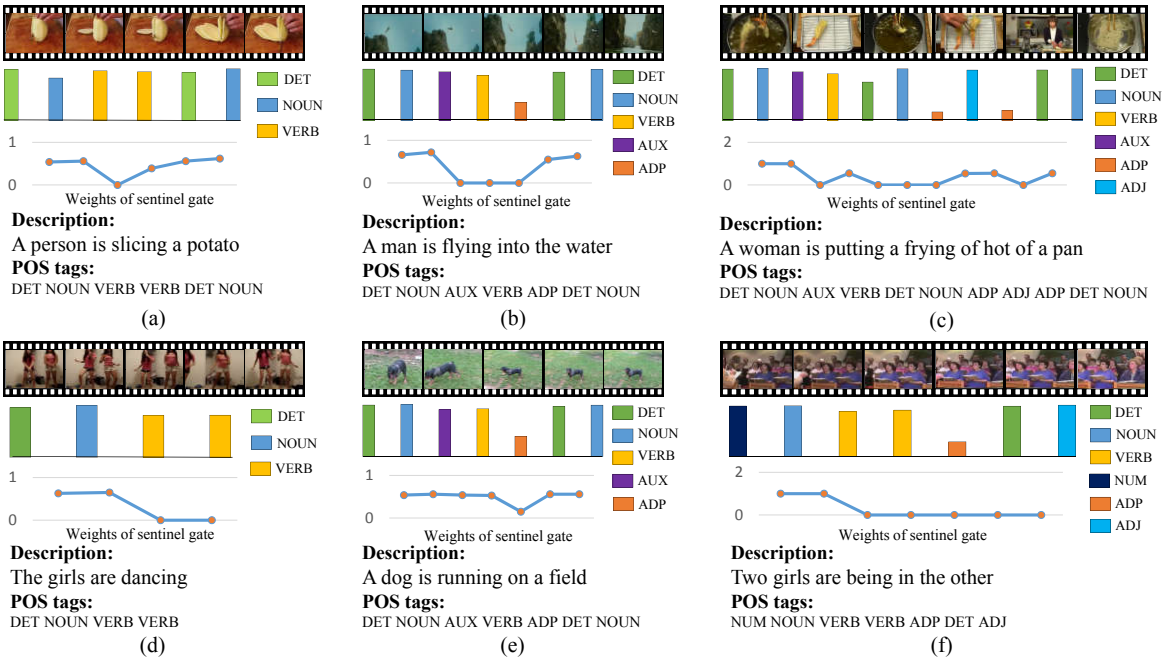


Figure 5: Visualization of the sampled video frames, the neuron activations associated with the POS tags, the weights of the sentinel gate, the generated captions, and the real POS tags of the captions.

From Fig. 5 (a), the words of "person", "slicing" and "potato" are obviously visual words which have related visual signals in video, and our model successfully extracts their visual information.

## 5 Conclusion

In this paper, we have proposed an Adaptive Semantic Guidance Network (ASGN), which extracts as well as guides the POS-aware semantic information into the corresponding encoded visual representations under the supervision of POS tag. Moreover, a new sentinel gate is introduced to determine how much the attended feature can be imported into the decoding process. It indicates that interpretable improvement not only makes the learning process more transparent, but also gives model more space to explore. The promising performance and interpretability improved merits of our method demonstrate the effectiveness of the POS tag. Furthermore, the proposed ASGN has a good flexibility which can be employed to the other language and vision fields, such as image captioning, visual question answering, and so on.

## 6 Acknowledgment

We thank the anonymous reviewers for their helpful comments. This work was supported by the National Natural Science Foundation of China under Grants 91646207, 61773377, and 61573352, the Young Elite Scientists Sponsorship Program by CAST (2018QNRC001), and the Beijing Natural Science Foundation under Grant L172053.

## References

Deshpande A, Aneja J, and Wang L. 2019. Diverse and accurate image captioning guided by part-of-speech. In *CVPR*, pages 10695–10704.

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual NLP. In *CoNLL*, pages 183–192.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *ACL Workshop*, volume 29, pages 65–72.

Yangyu Chen, Shuhui Wang, Weigang Zhang, and Qingming Huang. 2018. Less is more: Picking informative frames for video captioning. In *ECCV*, pages 367–384.

Jeff Donahue, Lisa Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Trevor Darrell, and Kate Saenko. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, pages 2625–2634.

Yinpeng Dong, Hang Su, Jun Zhu, and Bo Zhang. 2017. Improving interpretability of deep neural networks with semantic information. In *CVPR*, pages 975–983.

Raphaël Fargier and Marina Laganaro. 2015. Neural dynamics of object noun, action verb and action noun production in picture naming. *Brain and language*, 150:129–142.

Zhe Gan, Chuang Gan, Xiaodong He, Yunchen Pu, Kenneth Tran, Jianfeng Gao, Lawrence Carin, and Li Deng. 2016. Semantic compositional networks for visual captioning. *Computing Research Repository (CoRR)*, abs/1611.08002.

Sergio Guadarrama, Niveda Krishnamoorthy, Girish Malkarnenkar, Subhashini Venugopalan, Raymond Mooney, Trevor Darrell, and Kate Saenko. 2013. Youtube2text: Recognizing and describing arbitrary activities using semantic hierarchies and zero-shot recognition. In *ICCV*, pages 2712–2719.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*, pages 770–778.

Xinwei He, Baoguang Shi, Xiang Bai, Gui-Song Xia, Zhaoxiang Zhang, and Weisheng Dong. 2017. Image caption generation with part of speech guidance. *Pattern Recognition Letters*.

Chiori Hori, Takaaki Hori, Teng-Yok Lee, Ziming Zhang, Bret Harsham, John Hershey, Tim Marks, and Kazuhiko Sumi. 2017. Attention-based multimodal fusion for video description. In *ICCV*, pages 4203–4212.

Houpand Horoufchin, Danilo Bzdok, Giovanni Buccino, Anna Borghi, and Ferdinand Binkofski. 2018. Action and object words are differentially anchored in the sensory motor system - a perspective on cognitive embodiment. 8.

Chin Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *ACL Workshop*, volume 8.

Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. 2016. Knowing when to look: Adaptive attention via A visual sentinel for image captioning. *Computing Research Repository (CoRR)*, abs/1612.01887.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *Computing Research Repository (CoRR)*, abs/1609.07843.

Pingbo Pan, Zhongwen Xu, Yi Yang, Fei Wu, and Yueting Zhuang. 2016. Hierarchical recurrent neural encoder for video representation with application to captioning. In *CVPR*, pages 1029–1038.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318.

Ramakanth Pasunuru and Mohit Bansal. 2017. Reinforced video captioning with entailment rewards. In *EMNLP*, pages 979–985.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NeruIPS Workshop*.

Vasili Ramanishka, Abir Das, Dong Park, Subhashini Venugopalan, Lisa Hendricks, Marcus Rohrbach, and Kate Saenko. 2016. Multimodal video description. In *the ACM International Conference on Multimedia*, pages 1092–1096.

Steven Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *CVPR*, pages 1179–1195.

Zhiqiang Shen, Jianguo Li, Zhou Su, Minjun Li, Yurong Chen, Yu-Gang Jiang, and Xiangyang Xue. 2017. Weakly supervised dense video captioning. In *CVPR*, pages 5159–5167.

Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. 2015. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, pages 4489–4497.

Ramakrishna Vedantam, C. Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *CVPR*, pages 4566–4575.

Subhashini Venugopalan, Marcus Rohrbach, Jeffrey Donahue, Raymond Mooney, Trevor Darrell, and Kate Saenko. 2015a. Sequence to sequence - video to text. In *ICCV*, pages 4534–4542.

Subhashini Venugopalan, Huijuan Xu, Jeff Donahue, Marcus Rohrbach, Raymond Mooney, and Kate Saenko. 2015b. Translating videos to natural language using deep recurrent neural networks. In *NAACL*, pages 1494–1504.

Bairui Wang, Lin Ma, Wei Zhang, and Wei Liu. 2018a. Reconstruction network for video captioning. *Computing Research Repository (CoRR)*, abs/1803.11438.

Xin Wang, Wenhu Chen, Jiawei Wu, Yuan-Fang Wang, and William Yang Wang. 2017. Video captioning via hierarchical reinforcement learning. *Computing Research Repository (CoRR)*, abs/1711.11135.

Zhe Wang, Xiaoyi Liu, Limin Wang, Yu Qiao, Xiaohui Xie, and Charless Fowlkes. 2018b. Structured triplet learning with pos-tag guided attention for visual question answering. In *WACV*, pages 1888–1896.

Xian Wu, Guanbin Li, Qingxing Cao, Qingge Ji, and Liang Lin. 2018. Interpretable video captioning via trajectory structured localization. In *CVPR*, pages 6829–6837.

Jun Xu, Tao Mei, Ting Yao, and Yong Rui. 2016. MSR-VTT: A large video description dataset for bridging video and language. In *CVPR*, pages 5288–5296.

Li Yao, Atousa Torabi, Kyunghyun Cho, Nicolas Ballas, Christopher Pal, Hugo Larochelle, and Aaron Courville. 2015. Describing videos by exploiting temporal structure. In *ICCV*, pages 4507–4515.

# A Stack-Propagation Framework with Token-Level Intent Detection for Spoken Language Understanding

**Libo Qin, Wanxiang Che*, Yangming Li, Haoyang Wen, Ting Liu**
Research Center for Social Computing and Information Retrieval
Harbin Institute of Technology, China
{lbqin,car,yangmingli,hywen,tliu}@ir.hit.edu.cn

## Abstract

Intent detection and slot filling are two main tasks for building a spoken language understanding (SLU) system. The two tasks are closely tied and the slots often highly depend on the intent. In this paper, we propose a novel framework for SLU to better incorporate the intent information, which further guides the slot filling. In our framework, we adopt a joint model with Stack-Propagation which can directly use the intent information as input for slot filling, thus to capture the intent semantic knowledge. In addition, to further alleviate the error propagation, we perform the token-level intent detection for the Stack-Propagation framework. Experiments on two publicly datasets show that our model achieves the state-of-the-art performance and outperforms other previous methods by a large margin. Finally, we use the Bidirectional Encoder Representation from Transformer (BERT) model in our framework, which further boost our performance in SLU task.

## 1 Introduction

Spoken language understanding (SLU) is a critical component in task-oriented dialogue systems. It usually consists of intent detection to identify users' intents and slot filling task to extract semantic constituents from the natural language utterances (Tur and De Mori, 2011). As shown in Table 1, given a movie-related utterance "*watch action movie*", there are different slot labels for each token and an intent for the whole utterance.

Usually, intent detection and slot filling are implemented separately. But intuitively, these two tasks are not independent and the slots often highly depend on the intent (Goo et al., 2018). For example, if the intent of a utterance is `WatchMovie`, it is more likely to contain the slot `movie_name` rather than the slot `music_name`. Hence, it is promising to incorporate the intent information to guide the slot filling.

| **Sentence** | *watch* | *action* | *movie* |
|---|---|---|---|
| **Gold Slots** | `O` | `B-movie_name` | `I-movie_name` |
| **Gold Intent** | `WatchMovie` | | |

Table 1: An example with intent and slot annotation (BIO format), which indicates the slot of movie name from an utterance with an intent `WatchMovie`.

Considering this strong correlation between the two tasks, some joint models are proposed based on the multi-task learning framework (Zhang and Wang, 2016; Hakkani-Tür et al., 2016; Liu and Lane, 2016) and all these models outperform the pipeline models via mutual enhancement between two tasks. However, their work just modeled the relationship between intent and slots by sharing parameters. Recently, some work begins to model the intent information for slot filling explicitly in joint model. Goo et al. (2018) and Li et al. (2018) proposed the gate mechanism to explore incorporating the intent information for slot filling. Though achieving the promising performance, their models still suffer from two issues including: (1) They all adopt the gate vector to incorporate the intent information. In the paper, we argue that it is risky to simply rely on the gate function to summarize or memorize the intent information. Besides, the interpretability of how the intent information guides slot filling procedure is still weak due to the interaction with hidden vector between the two tasks. (2) The utterance-level intent information they use for slot filling may mislead the prediction for all slots in an utterance if the predicted utterance-level intent is incorrect.

In this paper, we propose a novel framework to address both two issues above. For the first issue, inspired by the Stack-Propagation which was

* Email corresponding.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2078–2087,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

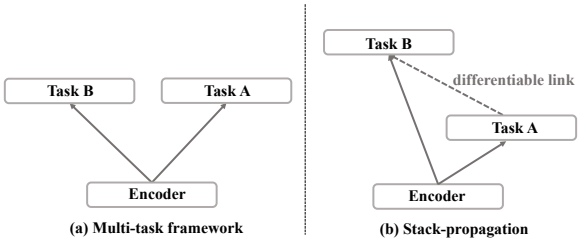


Figure 1: Multi-task framework vs. Stack-Propagation.

proposed by Zhang and Weiss (2016) to leverage the POS tagging features for parsing and achieved good performance, we propose a joint model with Stack-Propagation for SLU tasks. Our framework directly use the output of the intent detection as the input for slot filling to better guide the slot prediction process. In addition, the framework make it easy to design oracle intent experiment to intuitively show how intent information enhances slot filling task. For the second issue, we perform a token-level intent prediction in our framework, which can provide the token-level intent information for slot filling. If some token-level intents in the utterance are predicted incorrectly, other correct token-level intents will still be useful for the corresponding slot prediction. In practice, we use a self-attentive encoder for intent detection to capture the contextual information at each token and hence predict an intent label at each token. The intent of an utterance is computed by voting from predictions at each token of the utterance. This token-level prediction, like ensemble neural networks (Lee et al., 2016), reduces the predicted variance to improve the performance of intent detection. And it fits better in our Stack-Propagation framework, where intent detection can provide token-level intent features and retain more useful intent information for slot filling.

We conduct experiments on two benchmarks SNIPS (Coucke et al., 2018) and ATIS (Goo et al., 2018) datasets. The results of both experiments show the effectiveness of our framework by outperforming the current state-of-the-art methods by a large margin. Finally, Bidirectional Encoder Representation from Transformer (Devlin et al., 2018, BERT), as the pre-trained model, is used to further boost the performance of our model.

To summarize, the contributions of this work are as follows:

- We propose a Stack-Propagation framework in SLU task, which can better incorporate the intent semantic knowledge to guide the slot filling and make our joint model more interpretable.
- We perform the token-level intent detection for Stack-Propagation framework, which improves the intent detection performance and further alleviate the error propagation.
- We present extensive experiments demonstrating the benefit of our proposed framework. Our experiments on two publicly available datasets show substantial improvement and our framework achieve the state-of-the-art performance.
- We explore and analyze the effect of incorporating BERT in SLU tasks.

For reproducibility, our code for this paper is publicly available at `https://github.com/LeePleased/StackPropagation-SLU`.

## 2 Background

In this section, we will describe the formulation definition for intent detection and slot filling, and then we give a brief description of the multi-task framework and the joint model with Stack-Propagation framework.

### 2.1 Intent Detection and Slot Filling

Intent detection can be seen as a classification problem to decide the intent label $o^I$ of an utterance. Slot filling is a sequence labeling task that maps an input word sequence $\mathbf{x} = (x_1, \ldots, x_T)$ to slots sequence $\mathbf{o}^S = (o_1^S, \ldots, o_T^S)$.

### 2.2 Multi-task Framework vs. Stack-Propagation

For two correlative tasks *Task A* and *Task B*, multi-task framework which is shown in Figure 1 (a) can learn the correlations between these two tasks by the shared encoder. However, the basic multi-task framework cannot provide features from up-stream task to down-stream task explicitly. The joint model with Stack-Propagation framework which is shown in Figure 1 (b) can mitigate the shortcoming. In this figure, *Task B* can leverage features of *Task A* without breaking the differentiability in Stack-Propagation framework and promote each other by joint learning at the same time.

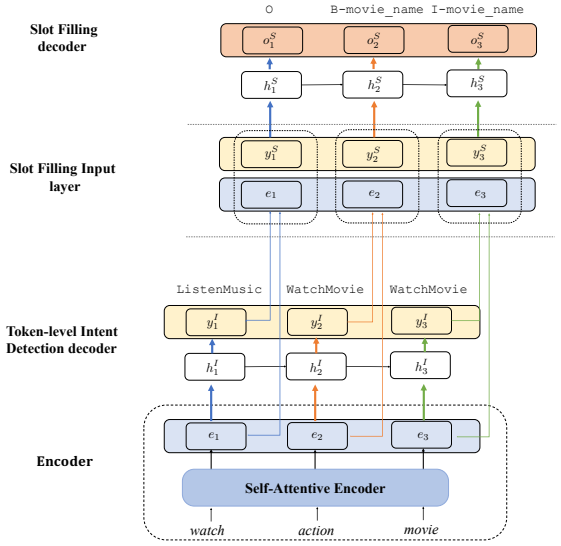


Figure 2: Illustration of our Stack-Propagation framework for joint intent detection and slot filling. It consists of one shared self-attentive encoder and two decoders. The output distribution of intent detection network and the representations from encoder are concatenated as the input for slot filling.

## 3 Approach

In this section, we will describe our Stack-Propagation framework for SLU task. The architecture of our framework is demonstrated in Figure 2, which consists of an encoder and two decoders. First, the encoder module uses one shared self-attentive encoder to represent an utterance, which can grasp the shared knowledge between two tasks. Then, the intent-detection decoder performs a token-level intent detection. Finally, our Stack-Propagation framework leverages the explicit token-level intent information for slot filling by concatenating the output of intent detection decoder and the representations from encoder as the input for slot filling decoder. Both intent detection and slot filling are optimized simultaneously via a joint learning scheme.

### 3.1 Self-Attentive Encoder

In our Stack-Propagation framework, intent detection task and slot filling task share one encoder, In the self-attentive encoder, we use BiLSTM with self-attention mechanism to leverage both advantages of temporal features and contextual information, which are useful for sequence labeling tasks (Zhong et al., 2018; Yin et al., 2018).

The BiLSTM (Hochreiter and Schmidhuber, 1997) reads input utterance $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, .., \mathbf{x}_T)$ ($T$ is the number of tokens in the input utterance) forwardly and backwardly to produce context-sensitive hidden states $\mathbf{H} = (\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_T)$ by repeatedly applying the recurrence $\mathbf{h}_i = \text{BiLSTM}\left(\phi^{\text{emb}}\left(x_i\right), \mathbf{h}_{i-1}\right)$.

Self-attention is a very effective method of leveraging context-aware features over variable-length sequences for natural language processing tasks (Tan et al., 2018; Zhong et al., 2018). In our case, we use self-attention mechanism to capture the contextual information for each token. In this paper, we adopt Vaswani et al. (2017), where we first map the matrix of input vectors $\mathbf{X} \in \mathbb{R}^{T \times d}$ ($d$ represents the mapped dimension) to queries ($\mathbf{Q}$), keys ($\mathbf{K}$) and values ($\mathbf{V}$) matrices by using different linear projections and the self-attention output $\mathbf{C} \in \mathbb{R}^{T \times d}$ is a weighted sum of values:

$$\mathbf{C} = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}. \quad (1)$$

After obtaining the output of self-attention and BiLSTM. We concatenate these two representations as the final encoding representation:

$$\mathbf{E} = \mathbf{H} \oplus \mathbf{C}, \quad (2)$$

where $\mathbf{E} \in \mathbb{R}^{T \times 2d}$ and $\oplus$ is concatenation operation.

### 3.2 Token-Level Intent Detection Decoder

In our framework, we perform a token-level intent detection, which can provide token-level intent features for slot filling, different from regarding the intent detection task as the sentence-level classification problem (Liu and Lane, 2016). The token-level intent detection method can be formalized as a sequence labeling problem that maps a input word sequence $\mathbf{x} = (x_1, ..., x_T)$ to sequence of intent label $\mathbf{o}^I = (o_1^I, ..., o_T^I)$. In training time, we set the sentence's intent label as every token's gold intent label. The final intent of an utterance is computed by voting from predictions at each token of the utterance.

The self-attentive encoder generates a sequence of contextual representations $\mathbf{E} = (\mathbf{e}_1, ..., \mathbf{e}_T)$ and each token can grasp the whole contextual information by self-attention mechanism. We use a unidirectional LSTM as the intent detection network. At each decoding step $i$, the decoder state $\mathbf{h}_i^I$ is calculated by previous decoder state $\mathbf{h}_{i-1}^I$, the previous emitted intent label distribution $\mathbf{y}_{i-1}^I$ and the aligned encoder hidden state $\mathbf{e}_i$:

$$\mathbf{h}_i^I = f\left(\mathbf{h}_{i-1}^I, \mathbf{y}_{i-1}^I, \mathbf{e}_i\right). \quad (3)$$

Then the decoder state $\mathbf{h}_i^I$ is utilized for intent detection:

$$\mathbf{y}_i^I = \text{softmax}\left(\mathbf{W}_h^I \mathbf{h}_i^I\right), \quad (4)$$

$$o_i^I = \text{argmax}(\mathbf{y}_i^I), \quad (5)$$

where $\mathbf{y}_i^I$ is the intent output distribution of the $i$th token in the utterance; $o_i^I$ represents the intent lable of $i$th token and $\mathbf{W}_h^I$ are trainable parameters of the model.

The final utterance result $o^I$ is generated by voting from all token intent results:

$$o^I = \text{argmax} \sum_{i=1}^{m} \sum_{j=1}^{n_I} \alpha_j \mathbb{1}[o_i^I = j], \quad (6)$$

where $m$ is the length of utterance and $n_I$ is the number of intent labels; $\alpha_j$ denotes a 0-1 vector $\alpha \in \mathbb{R}^{n_I}$ of which the $j$th unit is one and the others are zero; argmax indicates the operation returning the indices of the maximum values in $\alpha$.

By performing the token-level intent detection, there are mainly two advantages:

1. Performing the token-level intent detection can provide features at each token to slot filling in our Stack-Propagation framework, which can ease the error propagation and retain more useful information for slot filling. Compared with sentence-level intent detection (Li et al., 2018), if the intent of the whole sentence is predicted wrongly, the wrong intent would possibly apply a negative impact on all slots. However, in token-level intent detection, if some tokens in the utterance predicted wrongly, other correct token-level intent information will still be useful for the corresponding slot filling.

2. Since each token can grasp the whole utterance contextual information by using the self-attentive encoder, we can consider predictions at each token in an utterance as individual prediction to the intent of this utterance. And therefore, like ensemble neural networks, this approach will reduce the predicted variance and improve the performance of intent detection. The experiments section empirically demonstrate the effectiveness of the token-level intent detection.

### 3.3 Stack-propagation for Slot Filling

In this paper, one of the advantages of our Stack-Propagation framework is directly leveraging the explicit intent information to constrain the slots into a specific intent and alleviate the burden of slot filling decoder. In our framework, we compose the input units for slot filling decoder by concatenating the intent output distribution $\mathbf{y}_i^I$ and the aligned encoder hidden state $\mathbf{e}_i$.

For the slot-filling decoder, we similarly use another unidirectional LSTM as the slot-filling decoder. At the decoding step $i$, the decoder state $\mathbf{h}_i^S$ can be formalized as:

$$\mathbf{h}_i^S = f\left(\mathbf{h}_{i-1}^S, \mathbf{y}_{i-1}^S, \mathbf{y}_i^I \oplus \mathbf{e}_i\right), \quad (7)$$

where $\mathbf{h}_{i-1}^S$ is the previous decoder state; $\mathbf{y}_{i-1}^S$ is the previous emitted slot label distribution.

Similarily, the decoder state $\mathbf{h}_i^I$ is utilized for slot filling:

$$\mathbf{y}_i^S = \text{softmax}\left(\mathbf{W}_h^S \mathbf{h}_i^S\right), \quad (8)$$

$$o_i^S = \text{argmax}(\mathbf{y}_i^S), \quad (9)$$

where $o_i^S$ is the slot label of the $i$th word in the utterance.

### 3.4 Joint Training

Another major difference between existing joint work (Zhang and Wang, 2016; Goo et al., 2018) and our framework is the training method for intent detection, where we convert the sentence-level classification task into token-level prediction to directly leverage token-level intent information for slot filling. And the intent detection objection is formulated as:

$$\mathcal{L}_1 \triangleq -\sum_{j=1}^{m} \sum_{i=1}^{n_I} \hat{\mathbf{y}}_j^{i,I} \log\left(\mathbf{y}_j^{i,I}\right). \quad (10)$$

Similarly, the slot filling task objection is defined as:

$$\mathcal{L}_2 \triangleq -\sum_{j=1}^{m} \sum_{i=1}^{n_S} \hat{\mathbf{y}}_j^{i,S} \log\left(\mathbf{y}_j^{i,S}\right), \quad (11)$$

where $\hat{\mathbf{y}}_j^{i,I}$ and $\hat{\mathbf{y}}_j^{i,S}$ are the gold intent label and gold slot label separately; $n_S$ is the number of slot labels.

To obtain both slot filling and intent detection jointly, the final joint objective is formulated as

$$\mathcal{L}_\theta = \mathcal{L}_1 + \mathcal{L}_2. \quad (12)$$

Through the joint loss function, the shared representations learned by the shared self-attentive encoder can consider two tasks jointly and further ease the error propagation compared with pipeline models (Zhang and Wang, 2016).

## 4 Experiments

### 4.1 Experimental Settings

To evaluate the efficiency of our proposed model, we conduct experiments on two benchmark datasets. One is the publicly ATIS dataset (Hemphill et al., 1990) containing audio recordings of flight reservations, and the other is the custom-intent-engines collected by Snips (SNIPS dataset) (Coucke et al., 2018). [1] Both datasets used in our paper follows the same format and partition as in Goo et al. (2018). The dimensionalities of the word embedding is 256 for ATIS dataset and 512 for SNIPS dataset. The self-attentive encoder hidden units are set as 256. L2 regularization is used on our model is $1 \times 10^{-6}$ and dropout ratio is adopted is 0.4 for reducing overfit. We use Adam (Kingma and Ba, 2014) to optimize the parameters in our model and adopted the suggested hyper-parameters for optimization. For all the experiments, we select the model which works the best on the dev set, and then evaluate it on the test set.

### 4.2 Baselines

We compare our model with the existing baselines including:

- **Joint Seq**. Hakkani-Tür et al. (2016) proposed a multi-task modeling approach for jointly modeling domain detection, intent detection, and slot filling in a single recurrent neural network (RNN) architecture.
- **Attention BiRNN**. Liu and Lane (2016) leveraged the attention mechanism to allow the network to learn the relationship between slot and intent.
- **Slot-Gated Atten**. Goo et al. (2018) proposed the slot-gated joint model to explore the correlation of slot filling and intent detection better.
- **Self-Attentive Model**. Li et al. (2018) proposed a novel self-attentive model with the intent augmented gate mechanism to utilize the semantic correlation between slot and intent.
- **Bi-Model**. Wang et al. (2018) proposed the Bi-model to consider the intent and slot filling cross-impact to each other.
- **CAPSULE-NLU.** Zhang et al. (2019) proposed a capsule-based neural network model with a dynamic routing-by-agreement schema to accomplish slot filling and intent detection.
- **SF-ID Network.** (E et al., 2019) introduced an SF-ID network to establish direct connections for the slot filling and intent detection to help them promote each other mutually.

For the *Joint Seq*, *Attention BiRNN*, *Slot-gated Atten*, *CAPSULE-NLU* and *SF-ID Network*, we adopt the reported results from Goo et al. (2018); Zhang et al. (2019); E et al. (2019). For the *Self-Attentive Model*, *Bi-Model*, we re-implemented the models and obtained the results on the same datasets.[2]

### 4.3 Overall Results

Following Goo et al. (2018), we evaluate the SLU performance of slot filling using F1 score and the performance of intent prediction using accuracy, and sentence-level semantic frame parsing using overall accuracy. Table 2 shows the experiment results of the proposed models on SNIPS and ATIS datasets.

From the table, we can see that our model significantly outperforms all the baselines by a large margin and achieves the state-of-the-art performance. In the SNIPS dataset, compared with the best prior joint work *Bi-Model*, we achieve 0.7% improvement on Slot (F1) score, 0.8% improvement on Intent (Acc) and 3.1% improvement on Overall (Acc). In the ATIS dataset, we achieve 0.4% improvement on Slot (F1) score, 0.5% improvement on Intent (Acc) and 0.8% improvement on Overall (Acc). This indicates the effectiveness of our Stack-Propagation framework. Especially, our framework gains the largest improvements on sentence-level semantic frame accuracy,

[1] https://github.com/snipsco/nlu-benchmark/tree/master/2017-06-custom-intent-engines

[2] All experiments are conducted on the publicly datasets provided by Goo et al. (2018), *Self-Attentive Model* and *Bi-Model* don't have the reported result on the same datasets or they did different preprocessing. For directly comparison, we re-implemented the models and obtained the results on the ATIS and SNIPS datasets preprocessed by Goo et al. (2018). Because all baselines and our model don't apply CRF layer, we just report the best performance of *SF-ID Network* without CRF. It's noticing that our model does outperform *SF-ID Network* with CRF layer.

| Model | SNIPS | | | ATIS | | |
|---|---|---|---|---|---|---|
| | Slot (F1) | Intent (Acc) | Overall (Acc) | Slot (F1) | Intent (Acc) | Overall (Acc) |
| Joint Seq (Hakkani-Tür et al., 2016) | 87.3 | 96.9 | 73.2 | 94.3 | 92.6 | 80.7 |
| Attention BiRNN (Liu and Lane, 2016) | 87.8 | 96.7 | 74.1 | 94.2 | 91.1 | 78.9 |
| Slot-Gated Full Atten (Goo et al., 2018) | 88.8 | 97.0 | 75.5 | 94.8 | 93.6 | 82.2 |
| Slot-Gated Intent Atten (Goo et al., 2018) | 88.3 | 96.8 | 74.6 | 95.2 | 94.1 | 82.6 |
| Self-Attentive Model (Li et al., 2018) | 90.0 | 97.5 | 81.0 | 95.1 | 96.8 | 82.2 |
| Bi-Model (Wang et al., 2018) | 93.5 | 97.2 | 83.8 | 95.5 | 96.4 | 85.7 |
| CAPSULE-NLU (Zhang et al., 2019) | 91.8 | 97.3 | 80.9 | 95.2 | 95.0 | 83.4 |
| SF-ID Network (E et al., 2019) | 90.5 | 97.0 | 78.4 | 95.6 | 96.6 | 86.0 |
| Our model | **94.2*** | **98.0*** | **86.9*** | **95.9*** | **96.9*** | **86.5*** |
| Oracle (Intent) | 96.1 | - | - | 96.0 | - | - |

Table 2: Slot filling and intent detection results on two datasets. The numbers with * indicate that the improvement of our model over all baselines is statistically significant with $p < 0.05$ under t-test.

we attribute this to the fact that our framework directly take the explicit intent information into consideration can better help grasp the relationship between the intent and slots and improve the SLU performance.

To see the role of intent information for SLU tasks intuitively, we also present the result when using the gold intent information.[3] The result is shown in the *oracle* row of Table 2. From the result, we can see that further leveraging better intent information will lead to better slot filling performance. The result also verifies our assumptions that intent information can be used for guiding the slots prediction.

### 4.4 Analysis

In Section 4.3, significant improvements among all three metrics have been witnessed on both two publicly datasets. However, we would like to know the reason for the improvement. In this section, we first explore the effect of Stack-Propagation framework. Next, we study the effect of our proposed token-level intent detection mechanism. Finally, we study the effect of self-attention mechanism in our framework.

#### 4.4.1 Effect of Stack-Propagation Framework

To verify the effectiveness of the Stack-Propagation framework. we conduct experiments with the following ablations:

1) We conduct experiments to incorporate intent information by using gate-mechanism which is similar to Goo et al. (2018), providing the intent information by interacting with the slot filling decoder by gate function.[4] We refer it as *gate-mechanism*.

2) We conduct experiments on the pipelined model where the intent detection and slot filling has their own self-attentive encoder separately. The other model components keep the same as our framework. We name it as *pipelined model*.

Table 3 gives the result of the comparison experiment. From the result of *gate-mechanism* row, we can observe that without the Stack-Propagation learning and simply using the gate-mechanism to incorporate the intent information, the slot filling (F1) performance drops significantly, which demonstrates that directly leverage the intent information with Stack-Propagation can improve the slot filling performance effectively than using the gate mechanism. Besides, we can see that the intent detection (Acc) and overall accuracy (Acc) decrease a lot. We attribute it to the fact that the bad slot filling performance harms the intent detection and the whole sentence semantic performance due to the joint learning scheme.

Besides, from the *pipeline model* row of Table 3, we can see that without shared encoder, the performance on all metrics declines significantly. This shows that Stack-Propagation model can learn the correlation knowledge which may promote each other and ease the error propagation effectively.

[3] During the inference time, we concatenate the gold intent label distribution (one-hot vector) and the aligned encoder hidden state $e_i$ as the composed input for slot filling decoder. To keep the train and test procedure as the same, we replace our intent distribution as one-hot intent information for slot filling when train our model in our oracle experiments setting.

[4] For directly comparison, we still perform the token-level intent detection.

| Model | SNIPS | | | ATIS | | |
|---|---|---|---|---|---|---|
| | Slot (F1) | Intent (Acc) | Overall (Acc) | Slot (F1) | Intent (Acc) | Overall (Acc) |
| gate-mechanism | 92.2 | 97.6 | 82.4 | 95.3 | 96.2 | 83.4 |
| pipelined model | 90.8 | 97.6 | 81.8 | 95.1 | 96.1 | 82.3 |
| sentence intent augmented | 93.7 | 97.5 | 86.1 | 95.5 | 96.7 | 85.8 |
| lstm+last-hidden | - | 97.1 | - | - | 95.2 | - |
| lstm+token-level | - | 97.5 | - | - | 96.0 | - |
| without self-attention | 94.1 | 97.8 | 86.6 | 95.6 | 96.6 | 86.2 |
| Our model | **94.2** | **98.0** | **86.9** | **95.9** | **96.9** | **86.5** |

Table 3: The SLU performance on baseline models compared with our Stack-Propagation model on two datasets.

#### 4.4.2 Effect of Token-Level Intent Detection Mechanism

In this section, we study the effect of the proposed token-level intent detection with the following ablations:

1) We conduct the sentence-level intent detection in intent detection separately, which utilizes the last hidden vector of BiLSTM encoder for intent detection. We refer it to *lstm + last-hidden*. For comparison, our token-level intent detection without joint learning with slot filling is named as *lstm+token-level* in Table 3.

2) We conduct a joint learning framework that slot filling uses the utterance-level intent information rather than token-level intent information for each token, similar to intent-gated mechanism (Li et al., 2018), which is named as *sentence intent augmented.*

We show these two comparison experiments results in the first block of table 3. From the result, we can see that the token-level intent detection obtains better performance than the utterance-level intent detection. We believe the reason is that intent prediction on each token has similar advantage to ensemble neural networks, which can reduce the predicted variance to improve the intent performance. As a result, our framework can provide more useful intent information for slot filling by introducing token-level intent detection.

In addition, we can observe that if we only provide the sentence-level intent information for slot filling decoder, we obtain the worse results, which demonstrates the significance and effectiveness of incorporating token-level intent information. The main reason for this can be that incorporating the token-level intent information can retain useful features for each token and ease the error propagation.

#### 4.4.3 Effect of Self-attention Mechanism

We further investigate the benefits of self-attention mechanism in our framework. We conduct the comparison experiments with the same framework except the self-attention encoder is replaced with BiLSTM.

Results are shown in the *without self-attention* row of Table 3. We can observe the self-attention mechanism can further improve the SLU performance. We attribute this to the fact that self-attention mechanism can capture the contextual information for each token. Without the self-attention mechanism, it will harm the intent detection and have bad influence on slot filling task by joint learning.

It is noticeable that even without the self-attention mechanism, our framework still performs the state-of-the-art Bi-model model (Li et al., 2018), which again demonstrates the effectiveness and robustness of our other framework components.

### 4.5 Effect of BERT

Finally, we also conduct experiments to use pre-trained model, BERT (Devlin et al., 2018), to boost SLU performance. In this section, we replace the self-attentive encoder by *BERT base* model with the fine-tuning approach and keep other components as same with our framework.

Table 4 gives the results of BERT model on ATIS and SNIPS datasets. From the table, the BERT model performs remarkably well on both two datasets and achieves a new state-of-the-art performance, which indicates the effectiveness of a strong pre-trained model in SLU tasks. We attribute this to the fact that pre-trained models can provide rich semantic features, which can help to improve the performance on SLU tasks. In addition, our *model + BERT* outperforms the *BERT SLU* (Chen et al., 2019) which apply BERT for

| Model | SNIPS | | | ATIS | | |
|---|---|---|---|---|---|---|
| | Slot (F1) | Intent (Acc) | Overall (Acc) | Slot (F1) | Intent (Acc) | Overall (Acc) |
| Our model | 94.2 | 98.0 | 86.9 | 95.9 | 96.9 | 86.5 |
| Intent detection (BERT) | - | 97.8 | - | - | 96.5 | - |
| Slot filling (BERT) | 95.8 | - | - | 95.6 | - | - |
| BERT SLU (Chen et al., 2019) | 97.0 | 98.6 | 92.8 | 96.1 | 97.5 | 88.2 |
| Our model + BERT | 97.0 | 99.0 | 92.9 | 96.1 | 97.5 | 88.6 |

Table 4: The SLU performance on BERT-based model on two datasets.

joint the two tasks and there is no explicit interaction between intent detection and slot filling in two datasets in overall acc metric. It demonstrates that our framework is effective with BERT.

Especially, we also conduct experiments of intent detection task and slot filling separately based on BERT model. For intent detection, we put the special `[CLS]` word embedding into a classification layer to classify the intent. For slot filling, we feed the final hidden representation $\mathbf{h}_{\text{BERT}i} \in \mathbb{R}^d$ for each token $i$ [5] into a classification layer over the slot tag set. The results are also shown in the Table 4. From the result, we can see that the slot filling (F1) and intent detection accuracy (Acc) is lower than our joint model based on BERT, which again demonstrates the effectiveness of exploiting the relationship between these two tasks.

## 5 Related Work

Slot filling can be treated as a sequence labeling task, and the popular approaches are conditional random fields (CRF) (Raymond and Riccardi, 2007) and recurrent neural networks (RNN) (Xu and Sarikaya, 2013; Yao et al., 2014). The intent detection is formulated as an utterance classification problem, and different classification methods, such as support vector machine (SVM) and RNN (Haffner et al., 2003; Sarikaya et al., 2011), has proposed to solve it.

Recently, there are some joint models to overcome the error propagation caused by the pipelined approaches. Zhang and Wang (2016) first proposed the joint work using RNNs for learning the correlation between intent and slots. Hakkani-Tür et al. (2016) proposed a single recurrent neural network for modeling slot filling and intent detection jointly. Liu and Lane (2016) proposed an attention-based neural network for modeling the two tasks jointly. All these models outperform the pipeline models via mutual enhancement between two tasks. However, these joint models did not model the intent information for slots explicitly and just considered their correlation between the two tasks by sharing parameters.

Recently, some joint models have explored incorporating the intent information for slot filling. Goo et al. (2018) utilize a slot-gated mechanism as a special gate function to model the relationship between the intent detection and slot filling. Li et al. (2018) proposed the intent augmented gate mechanism to utilize the semantic correlation between slots and intent. Our framework is significantly different from their models including: (1) both of their approaches utilize the gate mechanism to model the relationship between intent and slots. While in our model, to directly leverage the intent information in the joint model, we feed the predicted intent information directly into slot filling with Stack-Propagation framework. (2) They apply the sentence-level intent information for each word while we adopt the token-level intent information for slot filling and further ease the error propagation. Wang et al. (2018) propose the Bi-model to consider the cross-impact between the intent and slots and achieve the state-of-the-art result. Zhang et al. (2019) propose a hierarchical capsule neural network to model the the hierarchical relationship among word, slot, and intent in an utterance. E et al. (2019) introduce an SF-ID network to establish the interrelated mechanism for slot filling and intent detection tasks. Compared with their works, our model can directly incorporate the intent information for slot filling explicitly with Stack-Propagation which makes the interaction procedure more interpretable, while their model just interacts with hidden state implicitly between two tasks.

## 6 Conclusion

In this paper, we propose a joint model for spoken language understanding with Stack-Propagation to better incorporate the intent information for slot

[5] We only consider the first subword label if a word is broken into multiple subwords

filling. In addition, we perform the token-level intent detection to improve the intent detection performance and further ease the error propagation. Experiments on two datasets show the effectiveness of the proposed models and achieve the state-of-the-art performance. Besides, we explore and analyze the effect of incorporating strong pre-trained BERT model in SLU tasks. With BERT, the result reaches a new state-of-the-art level.

## Acknowledgments

We thank the anonymous reviewers for their helpful comments and suggestions. This work was supported by the National Natural Science Foundation of China (NSFC) via grant 61976072, 61632011 and 61772153.

## References

Qian Chen, Zhu Zhuo, and Wen Wang. 2019. Bert for joint intent classification and slot filling. *arXiv preprint arXiv:1902.10909*.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, et al. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Haihong E, Peiqing Niu, Zhongfu Chen, and Meina Song. 2019. A novel bi-directional interrelated model for joint intent detection and slot filling. In *Proc. of ACL*.

Chih-Wen Goo, Guang Gao, Yun-Kai Hsu, Chih-Li Huo, Tsung-Chieh Chen, Keng-Wei Hsu, and Yun-Nung Chen. 2018. Slot-gated modeling for joint slot filling and intent prediction. In *Proc. of NAACL*.

Patrick Haffner, Gokhan Tur, and Jerry H Wright. 2003. Optimizing svms for complex call classification. In *In Proc. of ICASSP*.

Dilek Hakkani-Tür, Gökhan Tür, Asli Celikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In *Interspeech*.

Charles T Hemphill, John J Godfrey, and George R Doddington. 1990. The atis spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8).

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Stefan Lee, Senthil Purushwalkam Shiva Prakash, Michael Cogswell, Viresh Ranjan, David Crandall, and Dhruv Batra. 2016. Stochastic multiple choice learning for training diverse deep ensembles. In *NIPS*.

Changliang Li, Liang Li, and Ji Qi. 2018. A self-attentive model with gate mechanism for spoken language understanding. In *Proc. of EMNLP*.

Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. *arXiv preprint arXiv:1609.01454*.

Christian Raymond and Giuseppe Riccardi. 2007. Generative and discriminative algorithms for spoken language understanding. In *Eighth Annual Conference of the International Speech Communication Association*.

Ruhi Sarikaya, Geoffrey E Hinton, and Bhuvana Ramabhadran. 2011. Deep belief nets for natural language call-routing. In *ICASSP*.

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2018. Deep semantic role labeling with self-attention. In *Proc. of AAAI*.

Gokhan Tur and Renato De Mori. 2011. *Spoken language understanding: Systems for extracting semantic information from speech*. John Wiley & Sons.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, $\mathcal{L}$ $ukasz$ Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Yu Wang, Yilin Shen, and Hongxia Jin. 2018. A bi-model based rnn semantic frame parsing model for intent detection and slot filling. In *Proc. of ACL*.

Puyang Xu and Ruhi Sarikaya. 2013. Convolutional neural network based triangular crf for joint intent detection and slot filling. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*.

Kaisheng Yao, Baolin Peng, Yu Zhang, Dong Yu, Geoffrey Zweig, and Yangyang Shi. 2014. Spoken language understanding using long short-term memory neural networks. In *SLT*.

Qingyu Yin, Yu Zhang, Wei-Nan Zhang, Ting Liu, and William Yang Wang. 2018. Deep reinforcement learning for chinese zero pronoun resolution. In *Proc. of ACL*.

Chenwei Zhang, Yaliang Li, Nan Du, Wei Fan, and Philip Yu. 2019. Joint slot filling and intent detection via capsule neural networks. In *Proc. of ACL*.

Xiaodong Zhang and Houfeng Wang. 2016. A joint model of intent determination and slot filling for spoken language understanding. In *Proc. of IJCAI*.

Yuan Zhang and David Weiss. 2016. Stack-propagation: Improved representation learning for syntax. In *Proc. of ACL*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2018. Global-locally self-attentive encoder for dialogue state tracking. In *Proc. of ACL*.

# Talk2Car: Taking Control of Your Self-Driving Car

**Thierry Deruyttere**[1] **Simon Vandenhende**[2] **Dusan Grujicic**[2]
**Luc Van Gool**[2] **Marie-Francine Moens**[1]
[1]Department of Computer Science (CS)
[2]Department of Electrical Engineering (ESAT)
KU Leuven
{thierry.deruyttere, sien.moens}@cs.kuleuven.be
{simon.vandenhende, dusan.grujicic, luc.vangool}@esat.kuleuven.be

## Abstract

A long-term goal of artificial intelligence is to have an agent execute commands communicated through natural language. In many cases the commands are grounded in a visual environment shared by the human who gives the command and the agent. Execution of the command then requires mapping the command into the physical visual space, after which the appropriate action can be taken. In this paper we consider the former. Or more specifically, we consider the problem in an autonomous driving setting, where a passenger requests an action that can be associated with an object found in a street scene. Our work presents the `Talk2Car` dataset, which is the first object referral dataset that contains commands written in natural language for self-driving cars. We provide a detailed comparison with related datasets such as `ReferIt`, `RefCOCO`, `RefCOCO+`, `RefCOCOg`, `Cityscape-Ref` and `CLEVR-Ref`. Additionally, we include a performance analysis using strong state-of-the-art models. The results show that the proposed object referral task is a challenging one for which the models show promising results but still require additional research in natural language processing, computer vision and the intersection of these fields. The dataset can be found on our website: `http://macchina-ai.eu/`

## 1 Introduction

Researchers have studied the problem of understanding actions communicated through natural language in both simulated (Das et al., 2017; Gordon et al., 2017; Hermann et al., 2017) and real environments (Loghmani et al., 2018; de Vries et al., 2018; Anderson et al., 2017). This paper focuses on the latter. More concretely, we consider the problem in an autonomous driving setting, where a passenger can control the actions of an **A**utonomous **V**ehicle (AV) by giving natural language commands. We hereunder argue why this problem setting is particularly interesting.

First, a recent study by Richardson and Davies (2018) has shown that the majority of the public is reluctant to step inside an AV. A possible explanation for this might be the lack of control which can be unsettling to some. Providing a way to communicate with the vehicle could help alleviate this uneasiness. Second, an AV can become hesitant in some situations (Robitzski, 2019). By giving a task or command, the passenger could guide the agent in its decision process. Third, some situations request feedback. For example, a passenger might indicate that they want to park in the shade during a sunny day. Finally, the problem of urban scene understanding is one of practical relevance that has been well studied (Cordts et al., 2016; Geiger et al., 2013). We believe all of this makes it an interesting setting to assess the performance of grounding natural language commands into the visual space.

To perform the requested action, an agent is required to take two steps. First, the agent needs to interpret the command and ground it into the physical visual space. Secondly, the agent has to devise a plan to execute the given command. This paper focuses on this former step, or more concretely: given an image $I$ and a command $C$, the goal is to find the region $R$ in the image $I$ that the command is referring to. In this paper, to reduce the complexity of the object referral task we restrict the task to the case where there is only one targeted object that is referred to in the natural language command.

To stimulate research on grounding commands into the visual space we present the first object referral dataset, named `Talk2Car`, that comes with commands formulated in textual natural language for self-driving cars. A few example commands



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2088–2098,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics



(a) You can park up ahead behind **the silver car**, next to that lamppost with the orange sign on it

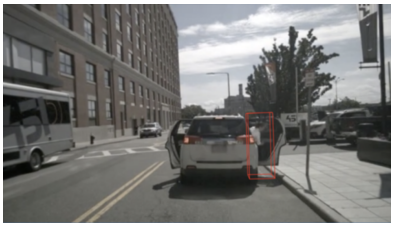

(b) **My friend** is getting out of the car. That means we arrived at our destination! Stop and let me out too!



(c) Yeah that would be **my son** on the stairs next to the bus. Pick him up please



(d) After **that man in the blue top** has passed, turn left



(e) There's **my mum**, on the right! The one walking closest to us. Park near **her**, she might want a lift



(f) Turn around and park in front of **that vehicle in the shade**

Figure 1: The `Talk2Car` dataset adds textual annotations on top of the `nuScenes` dataset for urban scene understanding. The textual annotations are free form commands, which guide the path of an autonomous vehicle in the scene. Each command describes a change of direction, relevant to a referred object found in the scene (here indicated by the red 3D-bounding box). Best seen in color.

together with their contextual images can be found in Fig. 1. Moreover, by using this new dataset we evaluate the performance of several strong state-of-the-art models that recognize the referred object of a command in the visual scene. Here we encounter several challenges. Referred objects are sometimes ambiguous (e.g., there are several cyclists in the scene), but can be disambiguated by understanding modifier expressions in language (e.g., the biker with the red jacket). These modifier expressions could also indicate spatial information. Furthermore, detecting the targeted object is challenging both in the language utterance and the urban scene, for instance, when dealing with complex and long sentences which might contain coreferent phrases, and with distant objects in the visual scene, respectively. Finally, in AV settings the speed of predicting the location of the referred object is of primordial importance.

The contributions of our work are the following:

- We propose the first object referral dataset for grounding commands for self-driving cars in free natural language into the visual context of a city environment.
- We evaluate several state-of-the-art models that recognize the referred object of a natural language command in the urban scene.
- We especially evaluate the models 1) for their capabilities to disambiguate objects based on modifying and spatial relationships expressed in language; 2) for their capabilities to cope with difficult language and visual context; and 3) with respect to prediction speed, which is important in real-life AV settings.

## 2 Related Work

**Object Referral** The `Talk2Car` dataset considers the object referral task, which requires to retrieve the correct object (region) from an image based on a language expression. A common method is to first extract regions of interest from the image, using a region proposal network (RPN). Yu et al. (2016); Mao et al. (2016) decode these proposals as a caption using a recurrent neural network (RNN). The predicted region corresponds to the caption that is ranked most similar to the referring expression. Other works based on RPN (Hu et al., 2017) or Faster-RCNN (Yu et al., 2018) have integrated attention mechanisms to decompose the language expressions into multiple sub-parts but use tailored modules for specific sub-tasks making

them less fit for our object referral task. Karpathy et al. (2014) interpret the inner product between region proposals and sentence fragments as a similarity score, allowing to match them in a bidirectional manner. Hu et al. (2016) uses an encoding of the global context in addition to the local context from the extracted regions. Hu et al. (2018) explore the use of modular networks for this task. They are comprised of multiple smaller predefined building blocks that can be combined together based on the language expression. The last three state-of-the-art models are evaluated on `Talk2Car` (section 5).

**Grounding in Human-Robot Interaction** When giving commands to robots, the grounding of the command in the visual environment is an essential task. Deits et al. (2013) use Generalized Grounding Graphs ($G^3$) (Tellex et al., 2011; Kollar et al., 2013) which is a probabilistic graphical model based on the compositional and hierarchical structure of a natural language command. This approach allows to ground certain parts of an image with linguistic constituents. Shridhar and Hsu (2018) consider the task where a robot arm has to pick up a certain object based on a given command. This is accomplished by creating captions for extracted regions from a RPN and clustering them together with the original command. If the command is ambiguous and more than one caption indicates the referring expression, the system will ask a clarifying question in order to be able to pick the right object. Due to its computational complexity during prediction, we did not select the last model in our evaluations.

**Visual Question Answering** The goal of VQA is to ask any type of question about an image for which the system should return the correct answer. This requires the system to have a good understanding of the image and the question. Early work (Kafle and Kanan, 2016; Zhou et al., 2015; Fukui et al., 2016) tried to solve the task by fusing image features extracted by a convolutional neural network (CNN), together with an encoding of the question. (Johnson et al., 2017; Suarez et al., 2018) experimented with modular networks for this task. Hudson and Manning (2018) proposed the use of a network made of recurrent Memory, Attention and Composition (MAC) cells. Similar to modular networks, the MAC model also uses multiple reasoning steps, making it a suitable model in our evaluation (section 5).

**Object Referral Datasets** Over the years, various object referral datasets based on both real world and computer generated images have been proposed. Kazemzadeh et al. (2014) introduced the first real-world large-scale object referral dataset named `ReferIt`. Yu et al. (2016) constructed `RefCOCO` and `RefCOCO+` and as the names suggest, these two datasets are based on the `MSCOCO` dataset (Lin et al., 2014). A third dataset also based on `MSCOCO`, named `RefCOCOg` (Mao et al., 2016), contains longer language expressions than the previous two datasets. This dataset has an average expression length of 8.43 words per expression compared to 3.61 for `RefCOCO` and 3.53 for `RefCOCO+`. The dataset closest to ours is the dataset by Vasudevan et al. (2018), as it augments `Cityscapes` (Cordts et al., 2016) with textual annotations. We will henceforth refer to this dataset as `Cityscapes-Ref`. The main difference with this work, is that the `Talk2Car` dataset contains commands, rather than descriptions. A computer generated dataset named `CLEVR-Ref` was proposed by Hu et al. (2018) which has been created by augmenting the `CLEVR` dataset (Johnson et al., 2016) such that it would include referred objects. We refer to section 4.1 for a thorough comparison between these datasets and ours. Chen et al. (2019) recently released the Touchdown dataset where the goal is to navigate through a city based on a textual itinerary and locate a target (called Touchdown) at the final destination. This can also be seen as an object referral task with the addition of following an itinerary. While being a very interesting dataset and problem, it differs from the task being evaluated in this paper

## 3 Dataset

### 3.1 Dataset Collection and Annotation

The `Talk2Car` dataset is built upon the `nuScenes` dataset (Caesar et al., 2019) which is a large-scale dataset for autonomous driving. The `nuScenes` dataset contains 1000 videos of 20 seconds each taken in different cities (Boston and Singapore), weather conditions (rain and sun) and different times of day (night and day). These videos account for a total of approximately 1.4 million images. Each scene comes with data from six cameras placed at different angles on the car,

LIDAR, GPS, IMU, RADAR and 3D bounding box annotations. The 3D bounding boxes discriminate between 23 different object classes.

We relied on workers of Amazon Mechanical Turk (AMT) to extend the videos from the nuScenes dataset with written commands. To create commands, each worker watches an entire 20 second long video from the front facing camera. Afterwards, the worker navigates to any point in the video that is found interesting. Once the worker has decided on the frame, a pre-annotated object from the nuScenes dataset for that frame needs to be selected. The annotation task is completed when a command referring to the selected object is entered. The workers were free to enter any command, as long as the car can follow a path based on the command.*

We hired five workers per video who could enter as many commands per video frame as they wanted. To ensure high quality annotations, we manually verified the correctness of all the commands and corresponding bounding boxes. The verification happened in a two-round system, where each annotation had to be qualified as adequate by two different reviewers. To incentivize workers to come up with diverse and meaningful commands, we awarded a bonus every time their work received approval.

### 3.2 Statistics of the Dataset

The Talk2Car dataset contains 11 959 commands for the 850 videos of the nuScenes training set as 3D bounding box annotations for the test set of the latter dataset are not disclosed. 55.94% and 44.06% of these commands belong to videos taken respectively in Boston and Singapore. On average a command consist of 11.01 words, 2.32 nouns, 2.29 verbs and 0.62 adjectives. Each video has on average 14.07 commands. In Fig. 2(d) we can see the distribution of distance to the referred objects. Fig. 2(b) displays the distribution of commands over the videos. On average there are 4.27 objects with the same category as the referred object per image and on average there are 10.70 objects per image. Fig. 2(a) shows a heatmap of the location of all referred objects in the images of Talk2Car. In Fig. 2(c) we see the distribution of commands that refer to an object of a certain category.

*The path to be followed by the car when executing the command, will be added in a later version of the dataset.

### 3.3 Dataset Splits

We have split the dataset in such a way that the train, validation and test set would contain 70%, 10% and 20% of the samples, respectively. To ensure a proper coverage of the data distribution in each set, we have taken a number of constraints into account. First, samples belonging to the same video are part of the same set. Second, as the videos are shot in either Singapore or Boston, i.e., in left or right hand traffic, the distributions of every split have to reflect this. Third, we aim to have a similar distribution of scene conditions across different sets, such as the type of weather and the time of day. Finally, as the number of occurrences of object categories is heavily imbalanced (see fig. 2(c)), we have ensured that every object category contained in the test set is also present in the training set. With these constraints in mind we randomly sample the three sets for 10 000 times and optimize for a data distribution of 70%, 10% and 20%. The resulting train, validation and test sets contain 8 349 (69.8%), 1 163 (9.7%) and 2 447 (20.4%) commands respectively. We have also identified multiple subsets of the test set, which allow evaluation of specific situations. When the referred object category occurs multiple times in an image, attributes in modifying expressions in language including spatial expressions might disambiguate the referred object. This has led to test sets with different numbers of occurrences of the targeted category of the referred object. Longer commands might contain irrelevant information or might be more complex to understand, leading to test sets with commands of different length. Finally, referred objects at large distances from the AV might be difficult to recognize. Hence, we have built test sets that contain referred objects at different distances from the AV.

## 4 Datasets Comparisons

### 4.1 Quantitative Evaluation of Talk2Car

Table 1 compares the Talk2Car dataset with prior object referral datasets. It can be seen that Talk2Car contains fewer natural language expressions than the others. However, although the dataset is smaller, the expressions are of high quality thanks to the double review system we discussed earlier in section 3.1. The main reason for having fewer annotations is that the original nuScenes dataset only discriminates between 23 different categories corresponding to the an-

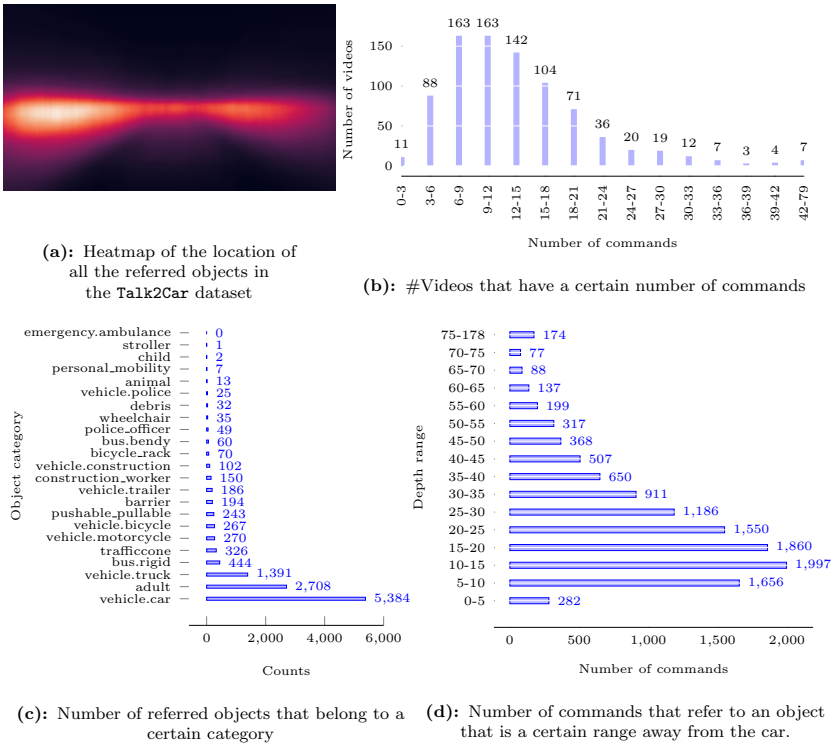


**(a):** Heatmap of the location of all the referred objects in the `Talk2Car` dataset

**(b):** #Videos that have a certain number of commands

**(c):** Number of referred objects that belong to a certain category

**(d):** Number of commands that refer to an object that is a certain range away from the car.

Figure 2: Statistics of the `Talk2Car` dataset.

notated bounding boxes. Moreover, the original `nuScenes` dataset considers the specific setting of urban scene understanding. This limits the visual domain considerably in comparison to `MS-COCO`. On the other hand, `Talk2Car` contains images in realistic settings accompanied by free language in contrast to curated datasets such as MS-COCO. Compared to most of the above datasets, the video frames annotated with natural language commands are part of larger videos that contain in total 1 183 790 images which could be exploited in the object referral task.

When we consider the average length of the natural language expressions in `Talk2Car`, we find that it ranks third, after `CLEVR-Ref` and `Cityscapes-Ref`. We did not put limitations on what the language commands could contain, which benefits the complexity and linguistic diversity of the expressions (section 4.2).

When looking at the type of modalities, the `Talk2Car` dataset considers RADAR, LIDAR and video. These modalities are missing in prior work except for video in `Cityscapes-Ref`. Including various modalities allows researchers to study a very broad range of topics with just a single dataset.

### 4.2 Qualitative Evaluation of `Talk2Car`

To make our discussion more concrete, we compare the textual annotations from Fig. 1 with some examples from prior work that are listed below. `RefCOCO` contains expressions such as 'Woman on right in white shirt' or 'Woman on right'. `RefCOCO+` on the other hand contains expressions such as 'Guy in yellow dribbling ball' or 'Yellow shirt in focus'. Lastly, `ReferIt` contains 'Right rocks', 'Rocks along the right side'. The language used in the above prior work is more simple, explicit and is well structured in comparison to the commands of `Talk2Car`. Additionally, the latter tend to include irrelevant side-information, e.g., 'She might want a lift', instead of being merely descriptive. The unconstrained free language of `Talk2Car` introduces different challenges, which involve co-reference resolution, named entity recognition, understanding relationships between objects, linking attributes to objects and understanding which object is the object of interest in the command.

The commands also contain implicit referrals as can be seen in the command in Fig. 1(f): 'Turn around and park in front of that vehicle in the shade'. Similar to `CLEVR-Ref`, object referral in `Talk2Car` requires some form of spatial rea-

| Dataset | Images | Objects | Expressions | Avg expr length | Video | Lidar | Radar |
|---|---|---|---|---|---|---|---|
| ReferIt (Kazemzadeh et al., 2014) | 19,894 | 96,654 | 130,525 | 3.46 | × | × | × |
| RefCOCO (Yu et al., 2016) | 26,711 | 50,000 | 142,209 | 3.61 | × | × | × |
| RefCOCO+ (Yu et al., 2016) | 19,992 | 49,856 | 141,564 | 3.53 | × | × | × |
| RefCOCOg (Mao et al., 2016) | 26,711 | 54,822 | 85,474 | 8.43 | × | × | × |
| CLEVR-Ref (Hu et al., 2018) | 99,992 | 492,727 | 998,743 | 14.50 | × | × | × |
| Cityscapes-Ref (Vasudevan et al., 2018) | 4,818 | 29,901 | 30,000 | 15.59 | ✓ | × | × |
| **Talk2Car (Ours)** | 9,217 | 10,519 | 11,959 | 11.01 | ✓ | ✓ | ✓ |

Table 1: Statistics of and comparison with existing datasets for object referral.

soning. However, in contrast to the former, there are cases where the spatial description in the command is misleading and truthfully reflects mistakes that people make. An example is the command in Fig. 1(e), where we refer to the object as being on the right side of the image, while the person of interest is actually located on the left.

Another important difference is the type of images in each dataset. For instance, the urban images in RefCOCO are taken from the viewpoint of a pedestrian. On the other hand, the images in Talk2Car are car centric.

## 5 Application of the State-of-the-Art Models and their Evaluation

We assess the performance of 7 models to detect the referred object in a command on the Talk2Car dataset. We discriminate between state-of-the-art methods based on region proposals and non-region proposal methods, apart from simple baselines. [†]

### 5.1 Region Proposal Based Methods

**Object Sentence Mapping (OSM)** This region proposal based method uses a single-shot detection model, i.e., SSD-512 (Liu et al., 2016), to extract 64 interest regions from the image. We pretrain the region proposal network (RPN) for the object detection task on the train images from the Talk2Car dataset. A ResNet-18 model is used to extract a local representation for the proposed regions. The natural language command is encoded using a neural network with Gated Recurrent Units (GRUs). Inspired by (Karpathy et al., 2014), we use the inner product between the latent representation of the region and command as a score for each proposal. The region that gets assigned the highest score is returned as bounding box for the object referred to by the command.

[†]All parameter values obtained on the validation set are cited in the supplementary material. For both the MAC and STACK-NMN we give the best results after empirically setting the number of reasoning steps.

**Spatial Context Recurrent ConvNet (SCRC)** A shortcoming of the above baseline model is that the correct region has to be selected based on local information alone. Spatial Context Recurrent ConvNets (Hu et al., 2016) match both local and global information with an encoding of the command. We reuse the SSD-512 model from above to generate region proposals. A global image representation is extracted by a ResNet-18 model. Additionally, we add an 8-dimensional representation of the spatial configuration to the local representation of each bounding box, $x_{spatial} = [x_{min}, y_{min}, x_{max}, y_{max}, h, w, x_{center}, y_{center}]$ with $h$ and $w$ respectively being the height and the width of this bounding box. For more details, we refer to the original work (Hu et al., 2016).

### 5.2 Non-Region Proposal Based Methods

**MAC model** This model (Hudson and Manning, 2018) originally created for the VQA task uses a recurrent MAC cell to match the natural language command represented with a Bi-LSTM model with a global representation of the image. A ResNet-101 is used to extract the visual features from the image. The MAC cell decomposes the textual input into a series of reasoning steps, where the MAC cell attends to certain parts of the textual input to guide the model to look at certain parts of the image. Between each of these reasoning steps, information is passed to the next cell such that the model is capable of representing arbitrarily complex reasoning graphs in a soft manner in a sequential way. The recurrent control state of the MAC cell identifies a series of read and write operations. The read unit extracts relevant information from both a given image and the internal memory. The write unit iteratively integrates the information into the cells' memory state, producing a new intermediate result.

**Stack-NMN** The Stack Neural Module Network or Stack-NMN (Hu et al., 2018) uses multiple modules that can solve a task by automatically

| Method | $IoU_{0.50}$ (%) | Inference Speed (ms) | Params (M) |
|---|---|---|---|
| RS | 2.05 | - | - |
| BOBB | 3.15 | - | - |
| RNM | 9.70 | - | - |
| MAC (Hudson and Manning, 2018) | **50.51** | 51 | 41.59 |
| STACK-NMN (Hu et al., 2018) | 33.71 | 52 | 35.2 |
| OSM (Karpathy et al., 2014) | 35.31 | 71 | 43.0 |
| SCRC (Hu et al., 2016) | 38.70 | 90 | 59.5 |

Table 2: Performance ($IoU_{0.50}$), inference speed (evaluated on a TITAN XP) and number of parameters of the different models.

inducing a sub-task decomposition, where each sub-task is addressed by a separate neural module. These modules can be chained together to decompose the natural language command into a reasoning process. Like the MAC model, this reasoning step is based on the use of an attention mechanism to attend to certain parts of the natural language command, which on their turn guide the selection of neural modules. The modules are first conditioned with the attended textual features after which they perform sub-task recognitions on the visual features. The output of these modules are attended parts in the image which are then given to the next reasoning step to continue the reasoning process. Again, a ResNet-101 model is used to extract the image features and a Bi-LSTM to encode the natural language command. To predict the referred object this model first splits the given image into a 2D grid. Then it tries to predict in which cell located in the grid the center of the referred object lies. Once this has been predicted, the model predicts the offsets of the bounding box relative to the predicted center.

### 5.3 Simple Baselines

**Random Selection (RS)** We reuse the single-shot detection model from section 5.1 to generate 64 region proposals per image of the test set. This model randomly samples one region from the proposals and uses it as prediction for the referred object. This is done 100 times and results are averaged.

**Biggest Overlapping Bounding Box (BOBB)** From the heatmap in Fig. 2 (a) we can see that there is some bias of the referred objects on the left side. This model tries to exploit this information by searching a 2D bounding box that optimizes the overlap with all the bounding boxes in the training set. The algorithm is explained in Section A of the supplementary material.

**Random Noun Matching (RNM)** In the test set a dependency parser (Honnibal and Johnson, 2015) is used to extract the set of nouns from a given command. We keep the nouns which are substrings of the category names. Then, we randomly sample an object from the region proposals of the corresponding image. If the set of category names is empty, we randomly sample a region from all region proposals. We re-use the RPN explained in OSM for the region proposals. This method is evaluated 100 times before averaging the results.

### 5.4 Results and Discussion

**Overall Results** We evaluated all seven models on the object referral task, using both the test split from subsection 3.3 as well as multiple increasingly challenging subsets from this test set. To properly evaluate existing models against our baselines we convert the 3D bounding boxes to 2D bounding boxes. We consider the predicted region correct when the Intersection over Union (IoU), that is, the intersection of the predicted region and the ground truth region over the union of these two, is larger than 0.5. Additionally, we report average inference speed at prediction time per sample and number of parameters of each model. We report the results obtained on the test set in Table 2. The results over the challenging test subsets can be seen in Fig. 3.

In all results we see the following: First, it is clear that the simple baselines (RS, BOBB, RNM) do not perform well, which evidences the difficulty of the object referral task in the realistic settings captured in `Talk2Car`. Second, MAC performs the best on nearly all tasks and it performs significantly better than STACK-NMN which is the model that resembles MAC the most.

If we compare the two RPN systems we see that SCRC often outperforms OSM, showing that using spatial information is beneficial. Third, being

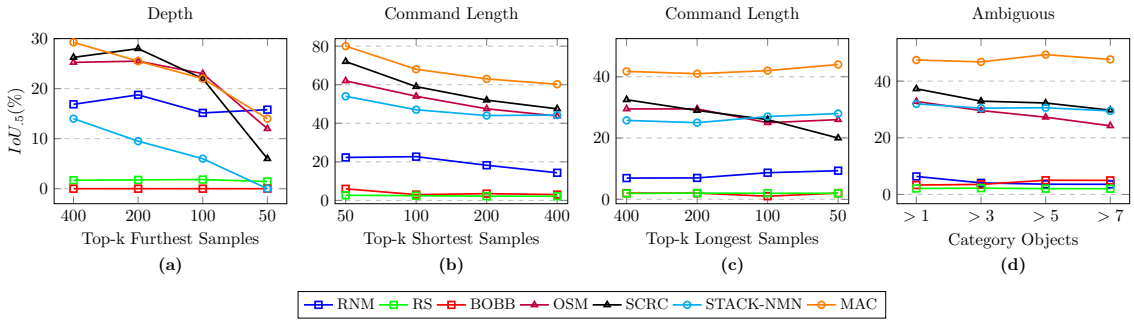


Figure 3: Test performance $IoU_{0.5}$ of different methods on the challenging sub-test sets. We discriminate between A test set for the top-k furthest objects in Fig. (**a**), the top-k shortest and longest commands in Fig. (**b**) and Fig. (**c**) respectively, and in function of the number of objects of the same category in the scene in Fig. (**d**).

able to discriminate between the different object classes in the scene is important. Or put it differently, correct alignment between objects in the image and the category names mentioned in the command is a basic requirement. RNM shows us that concentrating on nouns already gives a big improvement over a purely random strategy. In a separate experiment using ground truth bounding boxes the RNM system obtained an IoU of 54% showing the importance of the alignment of a found object to the correct category name. Fourth, the command length has a negative impact on most models as can be seen in 3(c). We argue that when commands get longer there might be more irrelevant information included which the models have difficulty to cope with. Fifth, from our experiments we found that the non-RPN systems are roughly two times faster than the RPN-systems. This is due to the fact that these RPN-systems have to align every proposed region with the command. On the other hand, the non-RPN systems only have to encode the full image once and then reason over this embedding. Lastly, when looking at the ambiguity test in Fig. 3(d) we see that all models struggle when the number of ambiguous objects of one category increase except for STACK and MAC, whose performance remains fairly stable. We believe they benefit from the multiple reasoning steps before giving an answer where modifier constructions in language disambiguate the referred object. In a separate experiment we have focused on object referral in extra long commands with ambiguous objects of the same category, where we observe the same trends.

**Influence of Region Proposal Quality** We consider the case when we pre-train a RPN on all keyframes from the training videos, rather than only on the images with commands. It is found that the test performance of the OSM model increased from 35.31 to 40.78%. Similarly, the test performance of the SCRC model increased from 38.70 to 41.15 % showing the importance of starting from good region proposals.

**Blanking out the commands** (Cirik et al., 2018) found that some referential datasets have some kind of bias in the dataset when blanking out the question. We evaluated this with both SCRC and OSM by changing the question vector to a zero filled vector and we respectively got the following results. For SCRC we get 40.37% $IoU_0.5$ (38.70% with command), OSM: 21.65% (35.31% with command). From these results we can conclude two things. First, global information which was added to the local representation of each region in the SCRC model, contains some kind of bias that the models can learn. Second, if no global information is used, as is the case in OSM, the model $IoU_0.5$ actually decreases dramatically indicating that there is not a high bias in the image itself.

**Influence of Using Pre-trained Word Embeddings** Using pre-trained word GloVe embeddings (Pennington et al., 2014) had no effect on or even lowered the IoU obtained on the test set. We argue that words like 'car' and 'truck' are very close to each other in the embedding space but for the model to perform well it should be able to discriminate between them. We also tested ELMO (Peters et al., 2018) and BERT (Devlin et al.,

2018) embeddings but found that they gave only minor improvements for some models.

## 6 Conclusions and Future Work

We have presented a new dataset, `Talk2Car`, that contains commands in natural language referring to objects in a visual urban scene which both the passenger and the self-driving car can see. We have compared this dataset to existing datasets for the joint processing of language and visual data and have performed experiments with different strong state-of-the-art models for object referral, which yielded promising results. The available 3D information was neglected to be able to compare existing models but we believe that it could help in object referral as it contains more spatial information which, as seen in the experiments, is an important factor. This 3D information will help to translate language into 3D. Moreover, it will allow to perform actions in 3D based on the given command. Also, the `Talk2Car` dataset only allows people to refer to one object at a time. It also doesn't include path annotations for the car to follow, nor does it have dialogues if a command is ambiguous. In future versions, `Talk2Car` will be expanded to include the above annotations and dialogues. However, this first version already offers a challenging dataset to improve current methods for the joint processing of language and visual data and for the development of suitable machine learning architectures. Especially for cases where the ambiguity in object referral can be resolved by correctly interpreting the constraints found in the language commands, `Talk2Car` offers a natural and realistic environment to study these.

## Acknowledgments

This project is sponsored by the MACCHINA project from the KU Leuven with grant number C14/18/065. We would like to thank Nvidia for granting us two TITAN Xp GPUs. We would also like to thank Holger Ceaser from nuTonomy for providing help with the original nuScenes dataset. We'd also like to thank Tinne Tuytelaars and Matthew Blaschko for their mentorship and good advices.

## References


Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. 2017. Vision-and-Language Navigation: Interpreting visually-grounded navigation instructions in real environments.

Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. 2019. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*.

Howard Chen, Alane Suhr, Dipendra Misra, Noah Snavely, and Yoav Artzi. 2019. Touchdown: Natural language navigation and spatial reasoning in visual street environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12538–12547.

Volkan Cirik, Louis-Philippe Morency, and Taylor Berg-Kirkpatrick. 2018. Visual referring expression recognition: What do systems actually learn? *arXiv preprint arXiv:1805.11818*.

Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. 2016. The cityscapes dataset for semantic urban scene understanding. *CoRR*, abs/1604.01685.

Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. 2017. Embodied Question Answering.

Robin Deits, Stefanie Tellex, Pratiksha Thaker, Dimitar Simeonov, Thomas Kollar, and Nicholas Roy. 2013. Clarifying commands with information-theoretic human-robot dialog. *Journal of Human-Robot Interaction*, 2(2):58–79.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. 2016. Multimodal compact bilinear pooling for visual question answering and visual grounding. *CoRR*, abs/1606.01847.

Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. 2013. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237.

Daniel Gordon, Aniruddha Kembhavi, Mohammad Rastegari, Joseph Redmon, Dieter Fox, and Ali Farhadi. 2017. IQA: visual question answering in interactive environments. *CoRR*, abs/1712.03316.

Karl Moritz Hermann, Felix Hill, Simon Green, Fumin Wang, Ryan Faulkner, Hubert Soyer, David Szepesvari, Wojciech Marian Czarnecki, Max Jaderberg, Denis Teplyashin, Marcus Wainwright, Chris Apps, Demis Hassabis, and Phil Blunsom. 2017. Grounded Language Learning in a Simulated 3D World. pages 1–22.

Matthew Honnibal and Mark Johnson. 2015. An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1373–1378, Lisbon, Portugal. Association for Computational Linguistics.

Ronghang Hu, Jacob Andreas, Trevor Darrell, and Kate Saenko. 2018. Explainable neural computation via stack neural module networks. *CoRR*, abs/1807.08556.

Ronghang Hu, Marcus Rohrbach, Jacob Andreas, Trevor Darrell, and Kate Saenko. 2017. Modeling relationships in referential expressions with compositional modular networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1115–1124.

Ronghang Hu, Huazhe Xu, Marcus Rohrbach, Jiashi Feng, Kate Saenko, and Trevor Darrell. 2016. Natural language object retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4555–4564.

Drew A. Hudson and Christopher D. Manning. 2018. Compositional attention networks for machine reasoning. *CoRR*, abs/1803.03067.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross B. Girshick. 2016. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. *CoRR*, abs/1612.06890.

Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Judy Hoffman, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017. Inferring and Executing Programs for Visual Reasoning. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2017-Octob, pages 3008–3017.

Kushal Kafle and Christopher Kanan. 2016. Answer-type prediction for visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4976–4984.

Andrej Karpathy, Armand Joulin, and Li F Fei-Fei. 2014. Deep fragment embeddings for bidirectional image sentence mapping. In *Advances in neural information processing systems*, pages 1889–1897.

Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. 2014. ReferItGame: Referring to Objects in Photographs of Natural Scenes. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 787–798.

Thomas Kollar, Stefanie Tellex, Matthew R Walter, Albert Huang, Abraham Bachrach, Sachi Hemachandra, Emma Brunskill, Ashis Banerjee, Deb Roy, Seth Teller, et al. 2013. Generalized grounding graphs: A probabilistic framework for understanding grounded language. *Journal of Artificial Intelligence Research*, pages 1–35.

Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312.

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. 2016. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer.

Mohammad Reza Loghmani, Barbara Caputo, and Markus Vincze. 2018. Recognizing objects in-the-wild: Where do we stand? In *IEEE International Conference on Robotics and Automation (ICRA)*.

Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L Yuille, and Kevin Murphy. 2016. Generation and comprehension of unambiguous object descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 11–20.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Ed Richardson and Philip Davies. 2018. The changing public's perception of self-driving cars.

Dan Robitzski. 2019. Exclusive: A waymo one riders experiences highlight autonomous rideshares shortcomings. https://futurism.com/waymo-one-early-rider-autonomous-vehicle, note=Accessed: 2019-01-27.

Mohit Shridhar and David Hsu. 2018. Interactive visual grounding of referring expressions for human-robot interaction. *arXiv preprint arXiv:1806.03831*.

Joseph Suarez, Justin Johnson, and Fei-Fei Li. 2018. DDRprog: A CLEVR Differentiable Dynamic Reasoning Programmer.

Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew R Walter, Ashis Gopal Banerjee, Seth Teller, and Nicholas Roy. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*.

Arun Balajee Vasudevan, Dengxin Dai, Luc Van Gool, and ETH Zurich. 2018. Object referring in videos with language and human gaze.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Harm de Vries, Kurt Shuster, Dhruv Batra, Devi Parikh, Jason Weston, and Douwe Kiela. 2018. Talk the walk: Navigating new york city through grounded dialogue. *CoRR*, abs/1807.03367.

Licheng Yu, Zhe Lin, Xiaohui Shen, Jimei Yang, Xin Lu, Mohit Bansal, and Tamara L Berg. 2018. Mattnet: Modular attention network for referring expression comprehension. In *CVPR*.

Licheng Yu, Patrick Poirson, Shan Yang, Alexander C Berg, and Tamara L Berg. 2016. Modeling context in referring expressions. In *European Conference on Computer Vision*, pages 69–85. Springer.

Bolei Zhou, Yuandong Tian, Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. 2015. Simple baseline for visual question answering. *CoRR*, abs/1512.02167.

# Fact-Checking Meets Fauxtography: Verifying Claims About Images

**Dimitrina Zlatkova**
Sofia University "St. Kliment Ohridski"
Sofia, Bulgaria
dvzlatkova@uni-sofia.bg

**Preslav Nakov**
Qatar Computing Research Institute, HBKU
Doha, Qatar
pnakov@hbku.edu.qa

**Ivan Koychev**
Sofia University "St. Kliment Ohridski"
Sofia, Bulgaria
koychev@fmi.uni-sofia.bg

## Abstract

The recent explosion of false claims in social media and on the Web in general has given rise to a lot of manual fact-checking initiatives. Unfortunately, the number of claims that need to be fact-checked is several orders of magnitude larger than what humans can handle manually. Thus, there has been a lot of research aiming at automating the process. Interestingly, previous work has largely ignored the growing number of claims about images. This is despite the fact that visual imagery is more influential than text and naturally appears alongside fake news. Here we aim at bridging this gap. In particular, we create a new dataset for this problem, and we explore a variety of features modeling the claim, the image, and the relationship between the claim and the image. The evaluation results show sizable improvements over the baseline. We release our dataset, hoping to enable further research on fact-checking claims about images.

## 1 Introduction

As social media become a bigger part of our daily lives, their influence over the way people think and make decisions increases. Inevitably, this has offered opportunities for fake content to arise and to spread faster than ever, e.g., recent research has shown that fake news spreads six time faster than real news (Vosoughi et al., 2018). Sometimes such content is created for pure entertainment or for financial gain from advertisement shown alongside the fake content, but more often and especially recently it has been used to spread disinformation, e.g., with the aim to influence political elections (Atanasov et al., 2019). To deal with the problem, a number of manual fact-checking initiatives have been launched, but they remain insufficient to cope with the ever growing number of check-worthy claims. Thus, automated methods have been proposed as a more scalable solution.



(a) A series of images show a Tesla vehicle in space.



(b) Photograph shows a fire rainbow over Idaho.

Figure 1: Examples of Real images and True claims.

Recently, a growing number of claims have been about images. The word *Fauxtography* has been used to describe images, especially news photographs, that convey a questionable, or outright false, sense of the events they seem to depict. The term was coined over a decade ago (Cooper, 2007), and there is growing research interest in the topic in the Computer Vision community (Bayar and Stamm, 2016; de Carvalho et al., 2016). Given the recent proliferation of fake news, and given that many of the questionable claims are about images, it would be natural to expect similar interest in the Computational Linguistics community, especially given the fact that visual imagery is more influential than text and naturally appears alongside fake news. Yet, computational fact-checking has mostly ignored the growing number of claims about images. Here we aim at bridging this gap. In particular, we create a new dataset for this problem, and we explore a variety of features modeling the claim, the image, and the relationship between the claim and the image.

Let us look at some examples. Figure 1 shows two images that look surrealistic, and thus spark interest and raise natural suspicion. Yet, they are in fact real.[1,2]

[1] http://www.snopes.com/fact-check/tesla-car-really-space

[2] http://www.snopes.com/fact-check/fire-rainbow



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2099–2108, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics



(a) A photograph shows a mountain resembling a turtle.



(b) A photograph shows Russian president Vladimir Putin aggressively pulling on U.S. president Barack Obama's tie.

Figure 2: Examples of Fake images and False claims.



(a) President Trump's notorious 'fist pump' at a Pennsylvania airport on 9/11 was offered as a greeting to a cancer victim.



(b) A photograph shows a palm-sized rabbit.

Figure 3: Examples of True images and False claims.

Figure 2 contains fake images accompanied by false claims. On the left, in Figure 2a, we see an image from a Facebook page that claims to show a turtle mountain. It turns out that this is not a genuine photograph of a real-world location, but a digital artwork comprising altered versions of at least two different photographs.[3] On the right, Figure 2b displays an image purportedly showing Putin aggressively grabbing President Obama by the tie and pulling him close. This image has been digitally manipulated.[4]

Finally, Figure 3 shows original photographs with false claims about them. The image in Figure 3a shows Trump with his fists in the air, but his gesture is not a greeting to a cancer victim as the claim states. The real image was used as a part of a meme[5] that was designed to make it seem that way. The photo in Figure 3a was posted by a Twitter account in an attempt to go viral, claiming that it shows a real bunny sitting in the palm of someone's hand. It actually shows a plush doll.[6]

As we have seen above, there are a number of reasons why an image may be deemed fake. In most cases, this involves some kind of digital manipulation, e.g., cropping, splicing, etc. However, there are cases when an image is completely legitimate, but it is published alongside some text that does not reflect its content accurately. This is our main focus here: we study the factuality of image-claim pairs.

[3] http://www.snopes.com/fact-check/turtle-mountain-photo/

[4] http://www.snopes.com/fact-check/putin-obama-tie-pull/

[5] http://www.snopes.com/fact-check/trump-fistpump-cancer-greeting/

[6] http://www.snopes.com/fact-check/bunny-toy-photograph/

The contributions of this paper can be summarized as follows:

- We study a new problem: predict the factuality of a claim with respect to an image.
- We create a new dataset for this problem, which we release to the research community in order to enable further work.
- We explore a variety of features, and we demonstrate sizable improvements over the baseline.

The remainder of this paper is organized as follows: Section 2 presents some relevant related work. Section 3 describes in depth our method and the various features we experimented with. Section 4 gives details about the datasets we created and used. Section 5 describes our experimental setup and presents the evaluation results. Section 6 gives additional details about the performance of the individual features, both in isolation and in various combinations, and further describes some unsuccessful attempts at extracting better features. Finally, Section 7 presents our conclusions and some ideas for future work.

## 2 Related Work

### 2.1 Fact-checking Claims

There has been a lot of research in the last few years in automatic fact-checking of claims and rumors, which can be classified into two general categories. The first approach focuses on the social aspects of the claim and how users in social media react to it (Canini et al., 2011; Castillo et al., 2011; Ma et al., 2016; Zubiaga et al., 2016; Ma et al., 2017; Dungs et al., 2018). This is reflected by user comments, likes/dislikes, views and other types of reactions, which are collected and used as features.

Other methods use the Web and try to find information that proves or disproves the claim (Mukherjee and Weikum, 2015; Popat et al., 2017; Karadzhov et al., 2017; Mihaylova et al., 2018; Baly et al., 2018b). In either case, what is important is the stance (Riedel et al., 2017; Thorne et al., 2017; Hanselowski et al., 2018; Mohtarami et al., 2018, 2019): whether the opinion expressed in a tweet or in an article by a particular user/source agrees/disagrees with the claim, and the reliability of the source, i.e., can we trust this source (Baly et al., 2018a, 2019).

We should note that all these approaches are limited to textual claims, while we are interested in claims about images.

### 2.2 Detecting Fake/Manipulated Images

The task of detecting fabricated images falls under the area of image forensics. Such tasks are usually solved using traditional statistical methods modeling color, shape, and texture features (Bayram et al., 2006; Stamm and Liu, 2010; de Carvalho et al., 2016). More recently, with the rise of Deep Learning, modern approaches and architectures have been applied to tackle the problem (Bayar and Stamm, 2016). However, most existing work uses datasets with generic images and very few papers specialize in the area of news and social media (Jin et al., 2016). Detecting manipulation in the images is relevant for us, but is not enough, since often the image is original, but the claim about it is false.

### 2.3 Fact-Checking Claims about Images

Little research exists on the topic of fact-checking claims about images, where the input to be analyzed is an image-claim pair. To the best of our knowledge, there is only one work closely related to ours: a recent paper (Zhang et al., 2018) presents a system called FauxBuster which aims to fight against Fauxtography. We differ from them in that we use the Web as a source of information. In contrast, they focus on the social aspects of the problem and use comments on Twitter and Reddit to extract features, which makes our work complementary to theirs. Unfortunately, direct comparison to their approach is not feasible, as their dataset is not freely accessible.

## 3 Method

This section describes the different approaches we applied towards engineering and extracting features from the image-claim pair.

We start with *reverse image search*. The classical image search allows users to search for images based on a text with specific words or phrases. In contrast, reverse image search takes as input an image and returns Web pages that include this exact image or images that are very similar to it. This process can be easily automated and applied to a large number of images via Google's Vision API.[7] It can also return other information related to the image, e.g., tags, the text on the image, some object detection, explicit content, etc.

Using reverse image search, for each image we obtain a maximum of 50 Web pages that contain it. We remove pages that are known to be from fact-checking Web sites such as snopes.com, factcheck.org, using open-source code.[8] For the remaining Web pages, we crawl the article and we get its title and text.

### 3.1 Features about the Image

**Google tags**: This is a list of tags that Google associates with the image. We decided to use this list because it contains words and phrases about events and people related to the image, which might give us an insight about what the image contains and what it is about. For example, the image in Figure 1a has the following tags: *SpaceX, Falcon Heavy, Rocket, Rocket launch, Falcon, Company, Launch pad, Booster, Thrust, Entrepreneur, Elon Musk*. After lowercasing them and removing stop words, we use them directly as bag-of-words features.

**URL domains**: The Web pages that contain the image usually come from media sources and represent articles on a topic related to the image and/or the claim attached to it. However, in some cases they might point to an image-hosting service or a social network Web site such as Pinterest, Imgur, Twitter, etc. In an effort to use this fact, we extracted the top-level domain names from the list of URLs and we used them as TF.IDF features.

**URL categories**: In order to get more insight about what types of websites write about fake and genuine images, we classify them in several predefined URL categories.

[7] http://cloud.google.com/vision/
[8] github.com/clef2018-factchecking

We use open-source code[9] to classify URLs, which performs rule-based matching of tokens from the URL against a predefined list of words. Given a URL, it assigns it a tuple of one higher-level and one lower-level category. For example, when we run the algorithm on the Web sites returned for the image in Figure 1a we get category tuples such as: ('arts & entertainment', 'general'), ('sports', 'general'), ('society', 'general'), ('technology & computing', 'general'), ('science', 'general'), ('automotive', 'general') and ('business', 'marketing'). To transform those into features, we take all Web sites returned by the reverse image search for the image, and we merge the lists of their category tuples. We do not differentiate between high- and low-level categories; rather, we just apply TF.IDF on the combined list.

**True/False/Mixed media percentage**: In order to determine whether an image is fake or not, we can also check the reliability of the sources that wrote about it. *Media Bias/Fact Check*[10] (MBFC) is a Web site that provides factuality information about 2700+ media sources. We use their database to classify each Web page that is returned by the reverse image search into the following categories: *True* (high factuality), *False* (low factuality) and *Mixed* (mixed factuality). Then, we use the percentage of Web pages from each category returned by the reverse image search as a feature.

**Known media percentage**: If a URL is not on the MBFC list, we label it as *Unknown* and we use the percentage of known Web pages as a feature.

**True/False/Mixed media titles**: We use the titles of the articles from a True, False or Mixed media as bag-of-words features.

### 3.2 Features about the Claim

So far, in our feature extraction process we have only used the image from the image-claim pair, which means we might be missing crucial information. After manual inspection of a few examples, we realized that about half of them can be classified only using the image, e.g., because it is a collage, was photoshopped, or manipulated in some way. The other half contain legitimate images that might appear on trustworthy Web sites, but the claim associated with them was false.

**Claim text**: We transform the text of the claim into a TF.IDF vector, which we use as a feature.

[9] http://github.com/matthewruttley/mozclassify

[10] http://mediabiasfactcheck.com/

### 3.3 Features about the Image-Claim pair

In addition to using the claim text, we want to check how it is related to the image and whether the claim is true with respect to it. We model that by comparing the text of the claim to the articles returned by the Reverse Image Search of the image. We use only the articles from trustworthy media sources, according to our MBFC labels. We approach the task of computing the similarity of those texts in two different ways.

**Cosine similarity**: We perform the comparison on the TF.IDF representations of the claim and each article's title. We compute a smoothed average on the list of cosine similarities to get the final feature value.

**Embedding similarity**: We use pretrained embeddings of size 512 (Cer et al., 2018) as a way to vectorize the claim and the title sentences. Then, it is trivial to calculate the similarity as a dot product, as they are already in a normalized form. Again, we use a smoothed average to reduce the list of similarities to a single number.

## 4 Data

As we have a new task, we needed to create our own dataset. In fact, we created two datasets from two separate sources, but with similar qualities and format. The main idea behind the data collection process was to find viral, interesting and even contradictory images with some text that describes them, i.e., the *claim*. Both datasets are in English.

### 4.1 The Snopes Dataset

Snopes.com is arguably the oldest and the largest fact-checking Web site online. It aims to fight misinformation by investigating different pieces of news. The site has a special section for image-related fact-checking, called *Fauxtography*[11]. It uses an extensive list of labels to classify each piece of news as *True*, *False*, *Miscaptioned*, *Mixture*, *Undetermined*, *Unproven*, *Outdated*, etc. For the purpose of our dataset, we gather only image-claim pairs that were labeled as either *True* or *False*. The collected data consists of **838 examples** of which 197 *True* and 641 *False*. The huge imbalance of the classes might be surprising at first, but it makes sense for fact-checkers to prefer to spend their time fact-checking news pieces that have a higher chance of being fake.

[11] http://www.snopes.com/fact-check/category/photos



(a) South Korean President Moon Jae-in and North Korean leader Kim Jong Un shake hands at the truce village of Panmunjom inside the demilitarized zone separating the two Koreas.



(b) Lava erupts from a fissure east of the Leilani Estates subdivision during ongoing eruptions of the Kilauea Volcano.

Figure 4: Examples of image-claim pairs from The Reuters Dataset.

Yet, this lack of *True*-labeled examples can pose some challenges for classification models and the evaluation process as well. This is why we decided to invest some time in gathering more *True* examples as we explain below.

### 4.2 The Reuters Dataset

At the end of each year, Reuters publishes a list of about 100 photos, called *Pictures of the Year*. Conveniently for us, each photo comes with a short textual description, which we can use as a *claim*. We collected all of these pictures from four consecutive years: 2015, 2016, 2017, and 2018. As a result, we ended up with a total of 395 True image-claim pairs. Some examples are shown in Figure 4. We further performed close manual inspection, and we did not find any obvious differences between these images compared to the ones from *The Snopes Dataset*. In terms of the claim, texts from Reuters seem to be longer, but this should not be a problem, since we do not use the length as a feature.

## 5 Experiments and Evaluation

### 5.1 Setup

Note that the above two datasets contain 1,233 examples combined, and these examples are relatively well-balanced: 592 *True* and 641 *False*. As this is a small size, we chose to test the performance of the models using cross-validation. If we mix the data from the two sources having in mind that the Reuters dataset has examples from the *True* class only, we fear that the models might implicitly learn each example's source, not its factuality. Hence, we designed the following two cross-validation experiments:

**Testing on Snopes-only data.** Ten times, using a different random seed, we do the following:

1. Randomly choose 50 *True* and 50 *False* Snopes examples and use them as a test set.

2. Use the rest of the *True* Snopes data plus all Reuters data as *True* training examples.

3. Randomly sample the necessary number of examples from the *False* Snopes data, so that the training set is balanced.

Finally, we compute the average of the evaluation measures for all ten folds.

**Testing on Snopes + Reuters data.** Ten times, using a different random seed, we perform the following steps:

1. Combine all Snopes and Reuters data into a single dataset.

2. Balance the resulting dataset by randomly choosing the necessary number of *False* examples.

3. Do a random train-test split, so that the test set contains 100 examples.

As in the previous experiment, we compute the average of the evaluation measures for all ten folds.

### 5.2 Classification model

We used a Linear SVM with the default value of $C$=1. We trained a separate SVM model for each feature type, then we applied a softmax to normalize the values, and finally we averaged the confidences of the classifiers to make the final decision.

### 5.3 Results

We used the following evaluation measures:

- **Accuracy**, because the classes are balanced, and the majority-class baseline for all experiments is 50.0.

- **Average Precision**, since it is useful if we want to have a ranking task, e.g., to prioritize which claims about images human fact-checkers should check first. Again, the random baseline for all experiments is 50.0.

| Feature | Acc (S) | AP (S) | Acc (S+R) | AP (S+R) |
|---|---|---|---|---|
| All | **63.2** | **73.0** | **80.1** | **90.3** |
| True media percentage | **62.1** | 59.3 | 74.6 | 69.8 |
| Embedding similarity of claim & true media titles | 61.1 | 62.5 | 74.0 | 69.0 |
| Cosine similarity of claim & true media titles | 61.1 | 58.4 | 73.8 | 69.0 |
| Known media percentage | 60.4 | 58.6 | 74.6 | 71.9 |
| URL domains | 60.3 | **67.3** | **78.6** | **89.7** |
| Google tags | 58.5 | 63.9 | 71.5 | 82.1 |
| Mixed media percentage | 58.0 | 56.1 | 62.4 | 60.8 |
| Claim text | 57.1 | 60.8 | 74.9 | 83.8 |
| True media titles | 55.8 | 63.1 | 73.6 | 81.4 |
| Mixed media titles | 55.4 | 58.5 | 63.1 | 67.2 |
| URL categories | 53.7 | 56.0 | 70.3 | 76.2 |
| False media titles | 50.3 | 50.8 | 50.6 | 50.4 |
| False media percentage | 49.9 | 51.1 | 50.4 | 50.4 |
| Baseline | 50.0 | 50.0 | 50.0 | 50.0 |

Table 1: Accuracy and Average Precision for individual feature types, calculated using 10-fold cross-validation using the Snopes dataset (**S**), and the Snopes+Reuters dataset (**S+R**).

Table 1 illustrates the importance of each feature type in isolation. We can see that almost all individual feature types manage to outperform the two 50% baselines. The only weak features are those related to false sources of information: percentage of unreliable media writing about the image and the words used in the titles of the articles. Moreover, using all features (with a model combination as explained above) works best: 63.2% and 80.1% Accuracy, 73.0% and 90.3% Average Precision for S and S+R, respectively. The top-3 feature types for the Snopes test set are *true media percentage* (62.1% for S and 74.6% for S+R), *embedding similarity* (61.1% for S and 74.0% for S+R), and *cosine similarity* (61.1% for S and 73.8% for S+R). In either experiment, Average Precision is higher than Accuracy. Larger improvements are achieved for the Snopes + Reuters test set, which could be due to the model making more mistakes on the *True* examples from Snopes and being better on *True* examples from Reuters.

Figure 5 shows combinations of the top-$n$ features using each feature's performance in terms of Average Precision. Note that these top features for the two experiments are different: we use the scores in the AP(S) column in Table 1 for the Snopes dataset, and the AP(S+R) column for the Snopes+Reuters dataset. We can see that selecting the top 4 to 5 features works best, yielding 65.4%, 75.1%, 84.1% and 92.5%.

Note that the Average Precision scores are higher than those for Accuracy, and the scores for the Snopes+Reuters dataset are higher.

## 6 Discussion

### 6.1 Most Important Individual Features

Above, we explored the performance of individual feature groups. Here we try to understand what the most important individual features are. For this purpose, we trained a model on all features, and then we analyzed the weight of each feature in this full model. Note that this is different from the setup in the previous section, where we trained a separate model for each feature group, and then we combined the predictions of these models in an ensemble; in contrast, here we just put all features from all groups together. The results are visualized in Figure 6. We can see that some of them seem random, e.g., *adventures of huckleberry fin* or *everything trump touches dies*. However, there are a few that signal false information, e.g., words like *fake* and *viral* mentioned in the title of a trustworthy medium, or tags like *hoax* and *fact-checking*. The existence of images in the dataset that were modified for artistic purposes can explain tags such as *art* and *film*. Also, according to our best features, we should not trust much images that appear on *Twitter* or ones related to sensitive topics like *african americans* or *islam*.

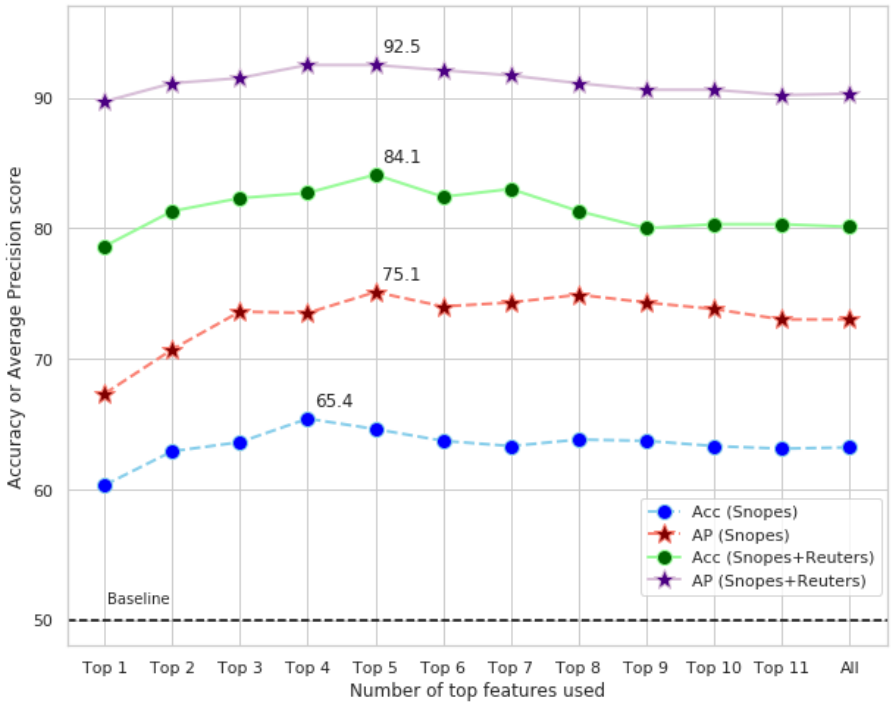


Figure 5: Accuracy and Average Precision on 10-fold cross-validation using top-$n$ features.

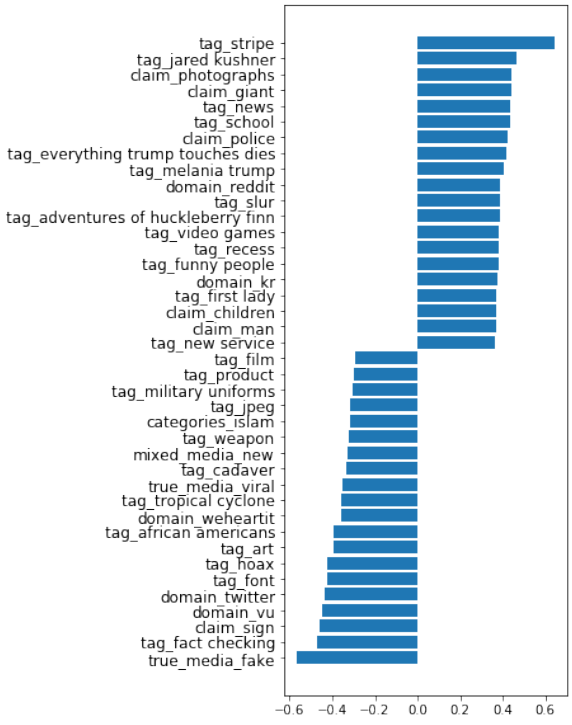


Figure 6: The most informative features: 20 positive and 20 negative. Prefixes indicate feature types.

### 6.2 What Did Not Work

**Metadata from images**: In an attempt to capture possible manipulation of the input image, we gathered meta information using an open-source tool[12] for image forensics. The tool extracts metadata in the form of about 100 features such as size, resolution, GPS location. However, most of this metadata turns out to be missing from our images: only five features could be extracted for more than half of the images from the Snopes dataset.

**Image Splice Detection**: As we have already mentioned, one of the reasons why an image could be fake is that it has been digitally manipulated. A common manipulation is splicing, i.e., cropping and stitching together parts of the same image or multiple different images. We explored an approach that looks for the lack of self-consistency in images and outputs clusters of the predicted image parts using two algorithms: MeanShift and DBSCAN (Huh et al., 2018). An illustration on how it works is shown in Figure 7. We decided to validate the method by using a pretrained model,[13] which we applied to some images from the Snopes dataset that were obvious cases of splicing.

[12] http://github.com/redaelli/imago-forensics

[13] http://github.com/minyoungg/selfconsistency

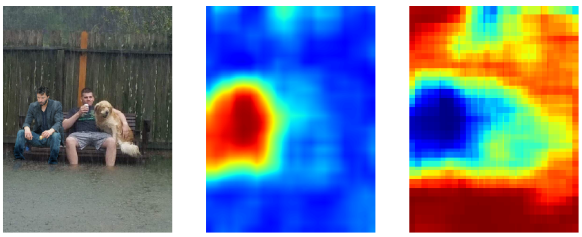

Figure 7: Predicted clusters for one of the images in the Self-Consistency paper. Keanu Reeves has been spliced into the photo and his body was separated correctly by both MeanShift and DBSCAN.

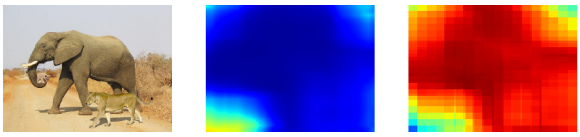

Figure 8: Predicted clusters for one of the images in the Snopes Dataset. The original image depicts an elephant; the lion and the cub have been photoshopped on top (Source). The clustering algorithms did not detect this splicing.

Unfortunately, this seemed not to work for us. Figure 8 shows an example where the model could not find the spliced regions. Eventually, we abandoned this direction as the inference time and the required resources were significant, and the performance was not very good on our dataset.

**Error Level Analysis**: Error Level Analysis (ELA) helps to identify areas within an image that are at different compression levels. With JPEG images such as the ones in our Snopes and Reuters datasets, the entire image should be at roughly the same level. If a section of the image is at a significantly different error level, this would indicate a likely digital modification.

ELA works by intentionally resaving the image at a known error rate such as 95%, and then computing the difference between the images. If there is virtually no change, then the cell has reached its local minima for error at that quality level. However, if there is a large change, then the pixels are not at their local minima and are effectively original. This method can be used to identify splicing, because stitched regions will appear brighter on the ELA version of the image. This is illustrated in Figure 9. After manual inspection of ELA versions of images from our dataset, we did not find the method to be very promising, see Figure 10.



Figure 9: One of the ELA examples on `http://fotoforensics.com`. The part of the image with a floppy disk appears brighter on the ELA map, as it was spliced on top of the original.



Figure 10: ELA applied to one of the images from the Snopes dataset. The spliced regions, i.e., the lion and the cub, could not be identified.

### 6.3 Testing on New Data

All of the experiments described so far were performed on claim-image pairs from Snopes that were published in the period between November 20, 2000 and February 1, 2019. The data from February up until April 29, 2019 has been left untouched, which makes it suitable for performing one final test of the developed system. In these three months, 64 articles were published in the Fauxtography section, of which 14 were labeled as *True* and 25 as *False*. To balance this new test set, we subsampled 14 *False* examples randomly. The training was performed on all previously collected data from Snopes and Reuters, balanced in the same way. For better certainty of the performance, we sampled randomly the training and the test sets ten times, and we report the average scores.

The results when using the top features based on the Average Precision for the Snopes dataset are shown in Table 2. We can see that the best Average Precision is achieved by using the single top feature of URL domains: 71.7%. When we add to this the second best one, i.e., the Google tags, we get an Accuracy of 64.3%. The scores of the models that use more than three features are not displayed since they were not as good.

The best-performing features across the experiments differ, but as Table 1 shows, the *URL domains* are top-1 in three out of four experiments, and *claim text* is top-2 in two out of four experiments.

| Features | Acc | AP |
|---|---|---|
| All | 59.3 | 69.7 |
| Top 1 | 62.9 | **71.7** |
| Top 2 | **64.3** | 70.4 |
| Top 3 | 57.5 | 70.6 |
| Baseline | 50.0 | 50.0 |

Table 2: Accuracy and Average Precision on the New test dataset.

## 7 Conclusion and Future Work

We have presented our efforts towards fighting Fauxtography, namely detecting fake claims about images, which is an under-explored research direction. In particular, we created a new dataset for this problem, and we explored a variety of features modeling the claim, the image, and the relationship between the two. The evaluation results have shown sizable improvements over the baseline. We release our dataset,[14] hoping to enable further research on fact-checking claims about images.

In future work, we plan to extend the dataset with more examples, to try other features, e.g., from social media and from metadata,[15] and to adapt the system to work with other languages. We further plan experiments with fact-checking claims about videos.

## Acknowledgements

This research is part of the Tanbih project,[16] which aims to limit the effect of "fake news", propaganda and media bias by making users aware of what they are reading. The project is developed in collaboration between the Qatar Computing Research Institute (QCRI), HBKU and the MIT Computer Science and Artificial Intelligence Laboratory (CSAIL).

[14] http://gitlab.com/didizlatkova/fake-image-detection

[15] The lack of metadata that we observed can be explained by the fact that Snopes.com is not the original source of the image files; it collected images from various external sources. Those sources might not be the original creator either and multiple downloading and uploading of files, with possible reformatting could mean loss of metadata as many Web sites reformat images and/or delete/change the metadata of the images uploaded to it. Finally, we could not extract any EXIF metadata for the Reuters images, even though we got them from Reuters. Yet, maybe the metadata can be recovered using Reverse Image Search.

[16] http://tanbih.qcri.org/

## References

Atanas Atanasov, Gianmarco De Francisci Morales, and Preslav Nakov. 2019. Understanding the roles of political trolls in social media. In *Proceedings of the 2019 SIGNLL Conference on Computational Natural Language Learning*, CoNLL '19, Hong Kong, China.

Ramy Baly, Georgi Karadzhov, Dimitar Alexandrov, James Glass, and Preslav Nakov. 2018a. Predicting factuality of reporting and bias of news media sources. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '18, pages 3528–3539, Brussels, Belgium.

Ramy Baly, Georgi Karadzhov, Abdelrhman Saleh, James Glass, and Preslav Nakov. 2019. Multi-task ordinal regression for jointly predicting the trustworthiness and the leading political ideology of news media. In *Proceedings of the 17th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT '19, pages 2109–2116, Minneapolis, MN, USA.

Ramy Baly, Mitra Mohtarami, James Glass, Lluís Màrquez, Alessandro Moschitti, and Preslav Nakov. 2018b. Integrating stance detection and fact checking in a unified corpus. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT '18, pages 21–27, New Orleans, LA, USA.

Belhassen Bayar and Matthew C. Stamm. 2016. A deep learning approach to universal image manipulation detection using a new convolutional layer. In *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security*, IH&MMSec '16, pages 5–10, Vigo, Spain.

Sevinç Bayram, Ismail Avcibas, Bülent Sankur, and Nasir D Memon. 2006. Image manipulation detection. *Journal of Electronic Imaging*, 15(4):041102.

Kevin R. Canini, Bongwon Suh, and Peter L. Pirolli. 2011. Finding credible information sources in social networks based on content and social structure. In *Proceedings of the IEEE International Conference on Privacy, Security, Risk, and Trust, and the IEEE International Conference on Social Computing*, SocialCom/PASSAT '11, pages 1–8, Boston, MA, USA.

Tiago Jose de Carvalho, Fábio Augusto Faria, Hélio Pedrini, Ricardo da Silva Torres, and Anderson Rocha. 2016. Illuminant-based transformed spaces for image forensics. *IEEE Transactions on Information Forensics and Security*, 11:720–733.

Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. 2011. Information credibility on Twitter. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, pages 675–684, Hyderabad, India.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, EMNLP '18, pages 169–174, Brussels, Belgium.

Stephen Cooper. 2007. A concise history of the fauxtography blogstorm in the 2006 Lebanon war. *American Communication Journal*, 9.

Sebastian Dungs, Ahmet Aker, Norbert Fuhr, and Kalina Bontcheva. 2018. Can rumour stance alone predict veracity? In *Proceedings of the 27th International Conference on Computational Linguistics*, COLING '18, pages 3360–3370, Santa Fe, NM, USA.

Andreas Hanselowski, Avinesh PVS, Benjamin Schiller, Felix Caspelherr, Debanjan Chaudhuri, Christian M. Meyer, and Iryna Gurevych. 2018. A retrospective analysis of the fake news challenge stance-detection task. In *Proceedings of the 27th International Conference on Computational Linguistics*, COLING '18, pages 1859–1874, Santa Fe, NM, USA.

Minyoung Huh, Andrew Liu, Andrew Owens, and Alexei A. Efros. 2018. Fighting fake news: Image splice detection via learned self-consistency. In *Computer Vision – ECCV 2018*, pages 106–124, Cham. Springer International Publishing.

Zhiwei Jin, Juan Cao, Yongdong Zhang, Jianshe Zhou, and Qi Tian. 2016. Novel visual and statistical image features for microblogs news verification. *IEEE Transactions on Multimedia*, PP:1–1.

Georgi Karadzhov, Preslav Nakov, Lluís Màrquez, Alberto Barrón-Cedeño, and Ivan Koychev. 2017. Fully automated fact checking using external sources. In *Proceedings of the Conference on Recent Advances in Natural Language Processing*, RANLP '17, pages 344–353, Varna, Bulgaria.

Jing Ma, Wei Gao, Prasenjit Mitra, Sejeong Kwon, Bernard J. Jansen, Kam-Fai Wong, and Meeyoung Cha. 2016. Detecting rumors from microblogs with recurrent neural networks. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, IJCAI '16, pages 3818–3824, New York, NY, USA.

Jing Ma, Wei Gao, and Kam-Fai Wong. 2017. Detect rumors in microblog posts using propagation structure via kernel learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, ACL '17, pages 708–717, Vancouver, Canada.

Tsvetomila Mihaylova, Preslav Nakov, Lluís Màrquez, Alberto Barrón-Cedeño, Mitra Mohtarami, Georgi Karadjov, and James Glass. 2018. Fact checking in community forums. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, AAAI '18, pages 879–886, New Orleans, LA, USA.

Mitra Mohtarami, Ramy Baly, James Glass, Preslav Nakov, Lluís Màrquez, and Alessandro Moschitti. 2018. Automatic stance detection using end-to-end memory networks. In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT '18, pages 767–776, New Orleans, Louisiana, USA.

Mitra Mohtarami, James Glass, and Preslav Nakov. 2019. Contrastive language adaptation for cross-lingual stance detection. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, EMNLP '19, Hong Kong, China.

Subhabrata Mukherjee and Gerhard Weikum. 2015. Leveraging joint interactions for credibility analysis in news communities. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 353–362, Melbourne, Australia.

Kashyap Popat, Subhabrata Mukherjee, Jannik Strötgen, and Gerhard Weikum. 2017. Where the truth lies: Explaining the credibility of emerging claims on the Web and social media. In *Proceedings of the 26th International Conference on World Wide Web Companion*, WWW '17, pages 1003–1012, Perth, Australia.

Benjamin Riedel, Isabelle Augenstein, Georgios P Spithourakis, and Sebastian Riedel. 2017. A simple but tough-to-beat baseline for the Fake News Challenge stance detection task. *ArXiv:1707.03264*.

Matthew C Stamm and KJ Ray Liu. 2010. Forensic detection of image manipulation using statistical intrinsic fingerprints. *IEEE Transactions on Information Forensics and Security*, 5(3):492–506.

James Thorne, Mingjie Chen, Giorgos Myrianthous, Jiashu Pu, Xiaoxuan Wang, and Andreas Vlachos. 2017. Fake news stance detection using stacked ensemble of classifiers. In *Proceedings of the EMNLP Workshop on Natural Language Processing meets Journalism*, pages 80–83, Copenhagen, Denmark.

Soroush Vosoughi, Deb Roy, and Sinan Aral. 2018. The spread of true and false news online. 359(6380):1146–1151.

Daniel Yue Zhang, Lanyu Shang, Biao Geng, Shuyue Lai, Ke Li, Hongmin Zhu, Md Tanvir Amin, and Dong Wang. 2018. Fauxbuster: A content-free fauxtography detector using social media comments. In *2018 IEEE International Conference on Big Data*, BigData '18, pages 891–900. IEEE.

Arkaitz Zubiaga, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Peter Tolmie. 2016. Analysing how people orient to and spread rumours in social media by looking at conversational threads. *PLoS ONE*, 11(3):1–29.

# Video Dialog via Progressive Inference and Cross-Transformer

**Weike Jin**[1], **Zhou Zhao**[1]*, **Mao Gu**[1], **Jun Xiao**[1], **Furu Wei**[2], **Yueting Zhuang**[1]
[1]Zhejiang University, Hangzhou
[2]Microsoft Research Asia, Beijing
{weikejin,zhaozhou,21821134,junx,yzhuang}@zju.edu.cn
fuwei@microsoft.com

## Abstract

Video dialog is a new and challenging task, which requires the agent to answer questions combining video information with dialog history. And different from single-turn video question answering, the additional dialog history is important for video dialog, which often includes contextual information for the question. Existing visual dialog methods mainly use RNN to encode the dialog history as a single vector representation, which might be rough and straightforward. Some more advanced methods utilize hierarchical structure, attention and memory mechanisms, which still lack an explicit reasoning process. In this paper, we introduce a novel progressive inference mechanism for video dialog, which progressively updates query information based on dialog history and video content until the agent think the information is sufficient and unambiguous. In order to tackle the multi-modal fusion problem, we propose a cross-transformer module, which could learn more fine-grained and comprehensive interactions both inside and between the modalities. And besides answer generation, we also consider question generation, which is more challenging but significant for a complete video dialog system. We evaluate our method on two large-scale datasets, and the extensive experiments show the effectiveness of our method.

## 1 Introduction

Visual dialog can be seen as an extension of the visual question answering (VQA) (Antol et al., 2015; Yu et al., 2017). Unlike visual question answering, in which each question is asked independently, visual dialog requires the agent to answer multi-round questions and the previous round of question-answer pairs form the dialog history. Currently, most of the existing visual dialog approaches mainly focus on image dialog (Das et al., 2017a; Seo et al., 2017; Kottur et al., 2018; Lee et al., 2018). Video dialog is still less explored. For video dialog, given a video, a dialog history and a question about the video content, the agent has to combine video information with context information from dialog history to infer the answer accurately. And due to the complexity of video information, it's more challenging than image dialog.

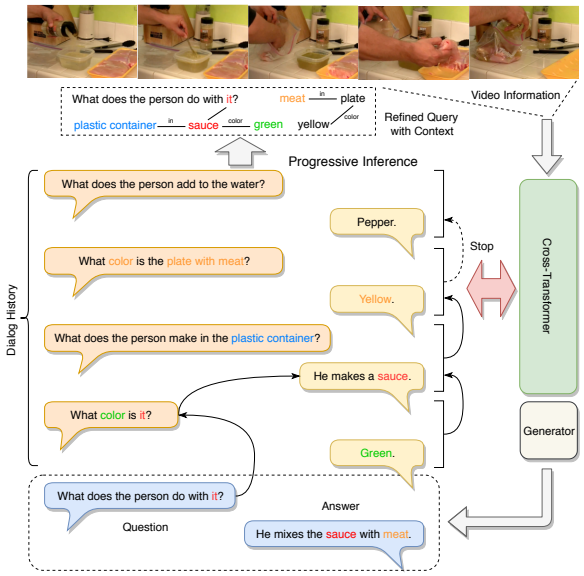


Figure 1: An illustration of our video dialog model with progressive inference and cross-transformer.

Obviously, there are two major problems for video dialog: (1) How to obtain valuable information from the dialog history. We find that the content of the dialogue usually has a certain continuity, and the subsequent dialogue is likely to continue what was discussed before. Thus, it's important to infer valuable information from the dialog history to help answer the question. Sometimes, the question can be ambiguous without such information. For instance, as illustrated in Figure 1, the question "What does the person do with

*Zhou Zhao is the corresponding author.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2109–2118, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

it?" contains a pronoun 'it', which brings ambiguity to the question. We can't figure out what 'it' really refers to, if only rely on the question information. There is no more key information in the question, thus it's also hard to get help from video content. In such a situation, the dialog history is necessary. The existing visual dialog methods (Das et al., 2017a) mainly use recurrent neural network (like LSTM) to encode the dialog history as a single vector representation, which we think might be a bit rough and straightforward. Some more advanced methods (Seo et al., 2017; Zhao et al., 2018) utilize hierarchical structure, attention and memory mechanisms to refine the dialog history representation, which still lacks an explicit reasoning process. Recently, Kottur *et al*. (2018) propose a neural module network architecture including two novel modules, which perform coreference resolution at a word level. However, their work is based on the image, which means they only take static visual characters into consideration, like objects and attributes. As for video dialog, there are additional dynamic characters in the video, such as action and state transition. Thus, we employ multi-stream video information in our model. And due to the continuity of the dialog history, we attempt to design a reverse-order progressive reasoning process to extract useful information and eliminate ambiguity. (2) How to answer the question related to the video content. This is also the key problem in video question answering. For instance, a mainstream method is to utilize recurrent neural network to learn the hidden states of video sequences, due to the temporal structure. Then, the encoded query information is used to select relevant information from the hidden states, through attention mechanism or memory network. Finally, the query information is combined with the refined video information by multi-modal fusion to generate the answer.

In this paper, in order to make better use of the dialog history and video information, we propose a progressive inference mechanism for video dialog, which progressively updates query information based on the dialog history and video content, as shown in Figure 1. And the progressive inference will stop when the agent thinks the query information is sufficient and unambiguous, or it has arrived at the beginning of the dialog history. For the query information is a sequence of vector representations which contains key information of the question, dialog history and video content, we propose a cross-transformer module to learn a stronger joint representation of them. And in addition to the generation of answers, we also attempt to generate questions according to the dialog history, which is more challenging but significant for a complete video dialog system. The generator we use is an extension of the Transformer (Vaswani et al., 2017) decoder, in order to process multi-stream inputs. And the main contributions of this paper are as follows:

- Unlike previous studies, we propose a novel progressive inference mechanism for video dialog, which progressively updates query information based on the dialog history and video content.
- We design a cross-transformer module to tackle the multi-modal fusion problem, which could learn more fine-grained and comprehensive interactions both inside and between the visual and textual information.
- Besides the answer generation, we also realize question generation under the same framework to construct a complete video dialog system.
- Our method achieves the state-of-the-art performance on two large-scale datasets.

## 2 Related Work

In this section, we briefly review some related work of visual dialog, including image dialog and video dialog.

As first proposed in (Das et al., 2017a), visual dialog requires the agent to predict the answer of a given question based on an image and dialog history. While dialog system (Serban et al., 2016, 2017) has been widely explored, visual dialog is still a young task. Until recently, some approaches are proposed. Das *et al.* (2017a) propose three models based on late fusion, attentional hierarchical LSTM and memory networks respectively. They also propose the VisDial dataset by pairing two subjects on Amazon Mechanical Turk to chat about an image. Lu *et al.* (2017) propose a generator-discriminator architecture where the outputs of the generator are updated using a perceptual loss from a pre-trained discriminator. De Vries *et al.* (2017) propose a GuessWhat game style dataset, on which one person

asks questions about an image to guess which object has been selected and the second person answers questions. Das *et al.* (Das et al., 2017b) utilize deep reinforcement learning to learn the policies of a 'Questioner-Bot' and an 'Answerer-Bot', based on the goal of selecting the right images that the two agents are talking. Seo *et al.* (2017) resolve visual references in the question based on a new attention mechanism with an attention memory, and the model indirectly resolves coreferences of expressions through the attention retrieval process. Jain *et al.* (2018) propose a simple symmetric discriminative baseline, which can be applied to both predicting an answer as well as predicting a question. Kottur *et al.* (2018) propose an introspective reasoning about visual coreferences, which links coreferences and grounds them in the image at a word-level, rather than at a sentence-level as in prior visual dialog work. Massiceti *et al.* (2018) propose FLIPDIAL model, a generative convolutional model for visual dialogue which is able to generate answers as well as generate questions based on a visual context. Lee *et al.* (2018) propose a practical goal-oriented dialog framework using information-theoretic approach, in which the questioner figures out the answerer's intention via selecting a plausible question by calculating the information gain of the candidate intentions and possible answers of each question. Wu *et al.* (2018) propose a sequential co-attention generative model that can jointly learn the image, dialog history information with question, and a discriminator which can dynamically access to the attention memories with an intermediate reward.

The aforementioned work mainly focuses on the image dialog. As for the task of video dialog, it's still less explored. One similar work is proposed by Zhao *et al.* (2018). They study the problem of multi-turn video question answering by employing a hierarchical attention context learning method with recurrent neural networks for context-aware question understanding and a multi-stream attention network that learns the joint video representation. They also propose two large-scale multi-turn video question answering datasets, which are employed in our experiments. And recently, Hori *et al.* (2018) propose a model that incorporates technologies for multimodal attention-based video description into an end-to-end dialog system. Pasunuru *et al.* (2018) propose a new game-chat based video-context, many-speaker dialogue task. In this work, we utilize a more explicit progressive inference mechanism and a novel cross-transformer module to generate both answers and questions for video dialog.

## 3 Our Approach

### 3.1 Problem Formulation

Before introducing our method, we first introduce some basic notions. We denote the video by $\mathbf{v} \in V$, the dialog history by $\mathbf{c} \in C$, the new question by $\mathbf{q} \in Q$ and the corresponding answer by $\mathbf{a} \in A$, respectively. For video is a sequence of frames, the frame-level representation for video $\mathbf{v}$ is denoted by $\mathbf{v}^f = (v_1^f, v_2^f, \ldots, v_{T_1}^f)$, where $T_1$ is the number of frames in video $\mathbf{v}$. And besides the frame-level representation, we also employ the segment-level representation to bring more information of video, which is given by $\mathbf{v}^s = (v_1^s, v_2^s, \ldots, v_{T_2}^s)$, where $T_2$ is the number of segments and $v_j^s$ is the vector representation of the $j$-th segment. The dialog history $\mathbf{c} \in C$ is given by $\mathbf{c} = (c_1, c_2, \ldots, c_N)$, where $c_i$ is the $i$-th round of dialog, which consists of a question $q_i$ and an answer $a_i$. Using these notations, the task of video dialog could be formulated as follows: given a set of video $V$ and the associated dialog history $C$, the goal of video dialog is to train a model that learns to generate human-like answers when a new question about the visual content is asked. Similar to the video question answering task, there are two kinds of methods to produce the answer, generative and discriminative. For generative type, a word sequence generator (normally a RNN) is employed to fit the ground truth answer sequences. As for discriminative type, an additional candidate answer vocabulary is provided and the problem is reformulated as a multi-class classification problem. In this paper, we try to tackle the generative version of video dialog, for it is more challenging than discriminative version and is more valuable in the practical system.

### 3.2 Progressive Inference

We first introduce our dialog encoder with progressive inference mechanism. Figure 2 shows the overview of the dialog encoder, in which the inference process has been expanded for intuitive understanding. As shown in the figure, the inputs of this module are a sequence of dialog history $\mathbf{c} = (c_1, c_2, \ldots, c_N)$ and the query $x$. Specifically, the query $x$ could be different ac-

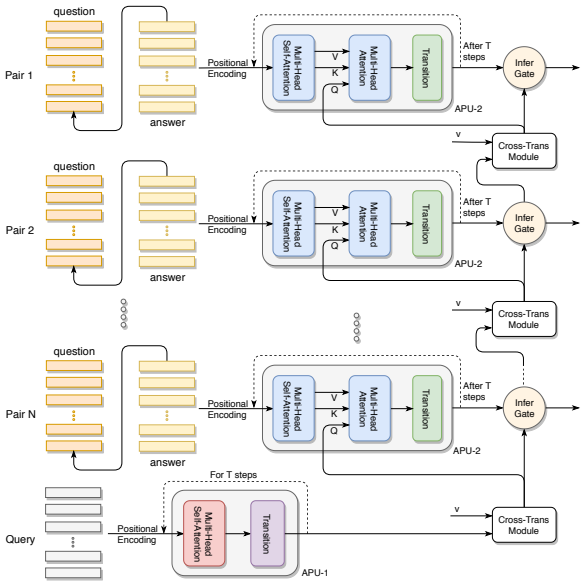


Figure 2: The overview of the dialog encoder with progressive inference.

cording to different purposes. For answer generation, the query $x$ is the question $q$ waiting to be answered. As for question generation, we combine the latest round of dialog $c_N$ in dialog history with ground-truth answer $a$ of the question as the query $x$, considering the topic continuity between the dialog. And it's intuitive that the new round of dialog is more likely to be relevant to the latest dialog history, especially in video dialog, in which the dialog focuses on a specific scenario not chit-chat. Each round of dialog consists of a question $q_i = (w^q_{i1}, w^q_{i2}, \ldots, w^q_{il})$ and an answer $a_i = (w^a_{i1}, w^a_{i2}, \ldots, w^a_{il'})$, where $w^q, w^a$ are word embeddings and $l, l'$ are corresponding sentence length. And for the i-th round dialog $c_i$, instead of using two LSTM-like neural networks to learn sentence-level representations individually, we concatenate the question $q_i$ and the answer $a_i$ end to end, given by $c_i = (w^q_{i1}, w^q_{i2}, \ldots, w^q_{il}, w^a_{i1}, w^a_{i2}, \ldots, w^a_{il'})$, and then employ self-attention mechanism to extract more fine-grained word-level interactions between the question-answer pair. And the query $x$ is also encoded by a similar self-attention mechanism. Specifically, the attention processing units (APU) we employ is based on the Transformer model (Vaswani et al., 2017), which has achieved great success in natural language processing. The first type of attention processing unit (APU-1) is as same as the encoder module of the Transformer model. It consists of a multi-head self-attention layer and a transition layer. And it has to be noted that we do not show residual connections and layer normalization in the figure for conciseness. Here, the transition layer is a fully-connected neural network that consists of two linear transformations with a rectified-linear activation in between. The second type of attention processing unit (APU-2) is similar to the decoder module of the Transformer model. However, there is a difference in the middle multi-head attention layer. We assume the outside input is $I_o$ and the output of the multi-head self-attention layer is $O_a$, then the normal operation in the Transformer decoder is given by

$$\text{Attention}(O_a, I_o, I_o) = \text{softmax}(\frac{O_a I_o^\top}{\sqrt{d}})I_o \quad (1)$$

where $d$ is the dimension of sequence elements. In our method, the order of the inputs are replaced, which is given by

$$\text{Attention}(I_o, O_a, O_a) = \text{softmax}(\frac{I_o O_a^\top}{\sqrt{d}})O_a \quad (2)$$

The reason of this replacement is that we want the outside input information to guide the inner attention operation, not the inside information. Under the video dialog scenario, we expect to use the query information to filter related and helpful information from the dialog history.

After introducing the basic modules of our dialog encoder, now we describe the progressive inference mechanism, as shown in Algorithm 1. Firstly, we add its position encoding to the input query $x$ to bring order information, which is similar to (Vaswani et al., 2017). Then, we employ the first type of attention processing unit (APU-1) to encode the initial query information and the encoded query is denoted as $q$. After that, we progressively update query information from each round of dialog. Due to the continuity of the dialog history, the order of the inference is from the latest round to the beginning round. For the i-th round, firstly, Cross-Transformer module is used to capture both query-aware video information and video-aware query information, and they are merged into the updated queries $\{O_{fq}, O_{sq}\}$, due to different visual features. Then, the information of current round of dialog history $c_i$ is encoded and filtered by the second type of attention processing unit (APU-2), using the latest updated query. Through this way, we can obtain the query related information $\hat{c}_i$ from current dialog, which

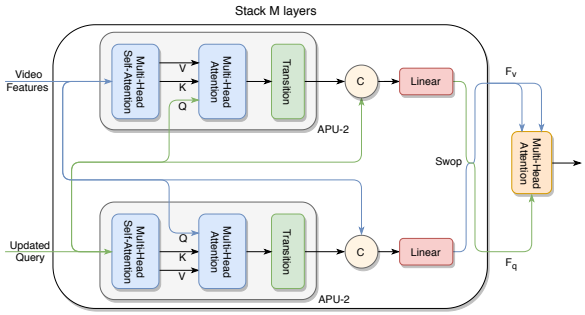


Figure 3: The overview of the cross-transformer module for multi-modal fusion.

could complete the query information and eliminate ambiguity. Finally, we utilize an inference gate to update the query information again and determine whether it is needed to stop reasoning, given by

$$S = \sigma([\hat{c}_i^1; O_{fq}]W_1 + b_1), \quad (3)$$
$$S' = \sigma([\hat{c}_i^1; O_{fq}]W_2 + b_2), \quad (4)$$
$$O_{fq} = S' \odot \hat{c}_i^1 + S \odot O_{fq}, \quad (5)$$
$$\mu_1 = \sigma(W_3(O_{fq}W_4 + b_3) + b_4) \quad (6)$$

where $\sigma$ is the sigmoid function, $[;]$ is concatenation operation, $\odot$ is element-wise multiplication, $W_1, W_2, W_3, W_4$ are weight matrixes, $b_1, b_2, b_3, b_4$ are biases, $S, S'$ are score vectors of the gate and $\mu_1$ (or $\mu_2$) is a scalar representing the information score. If $\mu = (\mu_1 + \mu_2)/2$ is greater than a threshold $\tau$, the inference gate will output current $\{O_{fq}, O_{sq}\}$ as the final output of the encoder and stop the inference, which means the agent think that the current query information is sufficient and unambiguous for answer or question generation. And the inference will also stop when it has arrived at the beginning of the dialog history.

### 3.3 Cross-Transformer Module

In this section, we introduce the cross-transformer (CT) module for multi-modal fusion, which could learn more fine-grained and comprehensive interactions both inside and between the input modalities. As shown in Figure 3, there are two input channels, one for video information and the other for query information. In our method, we utilize two kinds of video information, the frame-level information $v^f$ and the segment-level information $v^s$. Here, we take $v^f$ as an example, and the process of $v^s$ is the same. There are two attention process units (APU-2) in our CT module. And instead of encoding these two channels individually, we design a cross connection between the APU-2 to learn the interactions both inside and between different modalities. Specifically, we use the updated query $O_{fq}$ to guide the middle attention layer of the video channel APU-2, in order to learn the query-aware video information $v_q^f$. And the frame-level information $v^f$ is also utilized to guide the same attention layer of the query channel APU-2, for the purpose of learning frame-aware query information $q_f^u$. Then, we further fuse the information of each output of the APU-2 with its input guidance by a concatenation and a linear layer, given by

$$F_v = \text{Linear}([v^f : q_f^u]) \quad (7)$$
$$F_q = \text{Linear}([v_q^f : O_{fq}]) \quad (8)$$

Before outputting, we swop the output streams again to restore the original input order for next layer process. By stacking M layers, we expect to enhance the fusion effect. Finally, we employ another multi-head attention layer to fuse $F_v$ and $F_q$, and the output is denoted as $O_{fq}$. For segment-level video information $v^s$, we also could get the output $O_{sq}$.

### 3.4 Answer & Question Decoder

The decoder we use is shown in Figure 4. In this decoder, the third type of attention process unit (APU-3) is employed, which is as same as the Transformer decoder. And because there is

**Algorithm 1** Progressive Inference

**Input**: $c, x, v^f, v^s, N$
**Parameter**: $\hat{c}, q, i, \mu, \mu_1, \mu_2$
**Output**: $O_{fq}, O_{sq}$

Let $i = N, \mu = 0$
$q \leftarrow \text{APU\_1}(x + \text{position\_encoding})$
$O_{fq} \leftarrow O_{sq} \leftarrow q$
**while** $\mu < \tau$ and $i > 0$ **do**
  $O_{fq} \leftarrow \text{Cross\_Trans}(O_{fq}, v^f)$
  $O_{sq} \leftarrow \text{Cross\_Trans}(O_{sq}, v^s)$
  $\hat{c}_i^1 \leftarrow \text{APU\_2}(c_i, O_{fq})$
  $\hat{c}_i^2 \leftarrow \text{APU\_2}(c_i, O_{sq})$
  $\mu_1, O_{fq} \leftarrow \text{Infer\_Gate}(\hat{c}_i^1, O_{fq})$
  $\mu_2, O_{sq} \leftarrow \text{Infer\_Gate}(\hat{c}_i^2, O_{sq})$
  $\mu = (\mu_1 + \mu_2)/2$
  $i = i - 1.$
**end while**
**return** $O_{fq}, O_{sq}$

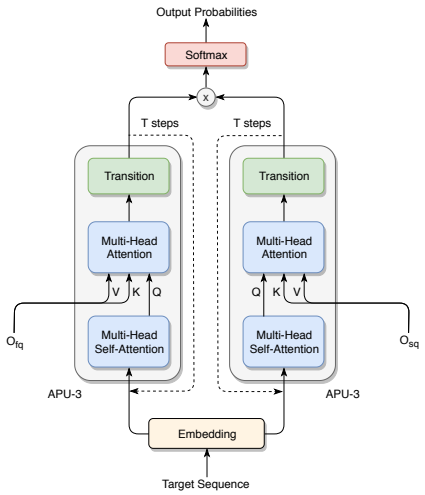


Figure 4: The overview of the answer and question decoder.

two outputs of the dialog encoder, $O_{fq}$ and $O_{sq}$, we utilize two individual attention process units (APU-3) to decode their information. Thus, we could obtain two different answer (or question) distributions based on the appearance and motion information correspondingly. Then, this two distributions are multiplied elemently and a softmax layer is employed to generate the final output probabilities. We use teacher-forcing strategy during training stage, and at generation time the answer (or question) is generated autoregressively.

## 4 Experiments

### 4.1 Setup

**Dataset.** The datasets that we use are proposed in (Zhao et al., 2018), which are based on YouTube-Clips (Chen and Dolan, 2011) dataset and TACoS-MultiLevel (Rohrbach et al., 2014) dataset. These two datasets consist of 1,987 and 1,303 videos individually. Each video in YouTubeClips is composed of 60 frames, and as for TACoS-MultiLevel, the number of frames is 80. Each video has five different dialogs. There are 6515 video dialogs in YouTubeClips dataset and 9935 video dialogs in TACoS-MultiLevel dataset. The numbers of question-answer pairs are 66,806 and 37,228 correspondingly. Statistically, there are five rounds of conversation in most of the video dialogs for TACoS-MultiLevel dataset and it is mostly between three and twelve rounds for YouTubeClips dataset. The percentages of training data, validation data and testing data for both datasets are 90%, 5%, and 5% respectively, according to the number of constructed video dialogs.

**Implementation.** Firstly, we use the pre-trained Glove (Pennington et al., 2014) model to obtain the word embeddings of the dialog. The dimension of the word vector is 512. Then, we employ the pre-trained VGGNet to learn appearance feature of each frame. The motion features of video are extracted by the pre-trained 3D-ConvNet (Tran et al., 2015). And we utilize transition layers (Section 3.2) to transform both appearance and motion features into the same dimension as word vectors to ease later process. We set the circulation steps T to 5 and the stack layers M to 4 after doing a lot of attempts. The threshold $\tau$ in the progressive inference is also important. It will influence the effect of the progressive inference process. We find that if the threshold is too small, the inference process will stop early and miss some important information. On the contrary, if the threshold is too large, it will stop too late leading to more interference information and time consuming. We did a lot of experiments and adjustments to get a balanced threshold of 0.85, and this value might be a bit different in different scenarios. For the training stage, we use Adam optimizer with the initial learning rate of 0.0005, and we also adopt a warm-up strategy to improve the effectiveness of network learning.

**Evaluation Metrics.** In this paper, we employ several metrics to evaluate the quality of the generated answers and questions. They are BLEU-N (N=1,2), ROUGE-L and METEOR, which have been widely used in natural language generation tasks.

### 4.2 Comparisons and Analysis

Because video dialog is still less explored especially for generative results, we extend some existing image dialog and video question answering models as baseline models for video dialog, which are introduced in the following:

- **ESA+, STAN+ and CDMN+** methods extend three video QA models respectively, which are ESA model (Zeng et al., 2017), STAN (Zhao et al., 2017) and CDMN model (Gao et al., 2018). A hierarchical LSTM network is added to model the dialog history.

- **LF+, HRE+ and MN+** methods extend three image dialog models (Das et al., 2017a) by

Table 1: Experimental results of answer generation on TACoS-MultiLevel and YoutubeClip datasets.

| Method | TACoS-MultiLevel | | | | YoutubeClip | | | |
|---|---|---|---|---|---|---|---|---|
| | BLEU-1 | BLEU-2 | ROUGE | METEOR | BLEU-1 | BLEU-2 | ROUGE | METEOR |
| ESA+ | 0.356 | 0.244 | 0.422 | 0.109 | 0.268 | 0.151 | 0.276 | 0.082 |
| STAN+ | 0.408 | 0.312 | 0.449 | 0.133 | 0.315 | 0.185 | 0.306 | 0.090 |
| CDMN+ | 0.429 | 0.341 | 0.460 | 0.142 | 0.293 | 0.161 | 0.311 | 0.094 |
| LF+ | 0.404 | 0.290 | 0.465 | 0.135 | 0.284 | 0.183 | 0.307 | 0.083 |
| HRE+ | 0.438 | 0.320 | 0.502 | 0.153 | 0.293 | 0.172 | 0.308 | 0.094 |
| MN+ | 0.430 | 0.326 | 0.472 | 0.149 | 0.306 | 0.185 | 0.290 | 0.086 |
| SFQIH+ | 0.438 | 0.334 | 0.481 | 0.153 | 0.326 | **0.202** | 0.319 | 0.085 |
| HACRN | 0.451 | 0.346 | 0.499 | 0.161 | 0.307 | 0.174 | 0.331 | 0.104 |
| RICT (ours) | **0.464** | **0.361** | **0.527** | **0.178** | **0.333** | 0.194 | **0.332** | **0.104** |

Table 2: Experimental results of question generation on TACoS-MultiLevel and YoutubeClip datasets.

| Method | TACoS-MultiLevel | | | | YoutubeClip | | | |
|---|---|---|---|---|---|---|---|---|
| | BLEU-1 | BLEU-2 | ROUGE | METEOR | BLEU-1 | BLEU-2 | ROUGE | METEOR |
| ESA+ | 0.693 | 0.582 | 0.718 | 0.341 | 0.497 | 0.333 | 0.565 | 0.212 |
| STAN+ | 0.706 | 0.599 | 0.730 | 0.354 | 0.483 | 0.322 | 0.559 | 0.208 |
| CDMN+ | 0.707 | 0.603 | 0.740 | 0.357 | 0.507 | 0.341 | 0.567 | 0.219 |
| LF+ | 0.704 | 0.598 | 0.728 | 0.349 | 0.512 | 0.346 | 0.574 | 0.218 |
| HRE+ | 0.694 | 0.592 | 0.729 | 0.348 | 0.515 | 0.350 | 0.571 | 0.223 |
| MN+ | 0.698 | 0.589 | 0.718 | 0.345 | 0.488 | 0.324 | 0.556 | 0.204 |
| SFQIH+ | 0.694 | 0.592 | 0.729 | 0.349 | 0.503 | 0.339 | 0.563 | 0.217 |
| HACRN | 0.715 | 0.616 | 0.741 | 0.358 | 0.524 | 0.352 | 0.577 | 0.229 |
| RICT (ours) | **0.733** | **0.625** | **0.748** | **0.367** | **0.536** | **0.375** | **0.593** | **0.234** |

utilizing a LSTM network to encode the video information, which are based on late fusion, attention based hierarchical LSTM, and memory networks respectively.

- **SFQIH+** method extends SF-QIH-se model (Jain et al., 2018) by employing a LSTM network to encode the video information, which concatenates all of the input embeddings for each of the possible answer options.

Besides the baseline models, we also compare our method with the HACRN model (Zhao et al., 2018). They propose a similar work called multi-turn video question answering. It uses LSTM and attention mechanism to encode the dialog history and question to get a joint question representation. They combine this joint representation with video features by a multi-stream attention network. And a multi-step reasoning strategy is applied to enhance the reasoning ability. For the above models, a normal LSTM recurrent sentence generator is employed to generate the answer and question.

Table 1 shows the experimental results of answer generation on TACoS-MultiLevel and YoutubeClip datasets, and Table 2 shows the question generation results on same datasets. Our method (RICT) outperforms all above models in almost all metrics. This fact shows the effectiveness of our overall network architecture. And we find that the image dialog models perform better than video QA models in answer generation, but worse in question generation on both datasets. This might indicate that for these two datasets, the answer generation is more dependent on dialog, and question generation is more dependent on video content.

We perform an ablation study to evaluate the impact of each component in our model. The experimental results are listed in Table 3, where $\text{RICT}_{(lstm)}$, $\text{RICT}_{(wo.ct)}$, $\text{RICT}_{(wo.pi)}$ represent our model with a LSTM sentence generator, replacing cross-transformer module and replacing progressive inference with baseline modules correspondingly. The results on YoutubeClip dataset is similar, which it's not shown in the paper. The fact that the variants perform worse than full RICT model but still better than baseline models that

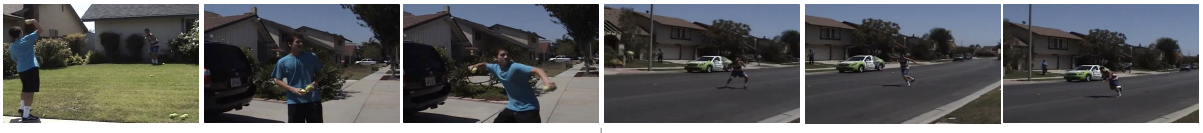

| questions | answers | μ-scores | Input: What does the boy in striped T-shirt see before running away? |
|---|---|---|---|
| Where are the two boys? | On the lawn. | - | Ground truth: Tennis ball. |
| What is the boy in blue holding? | Tennis ball. | 0.89 | HACRN: Automobile. |
| Who is the boy in blue throwing balls at? | A boy in striped T-shirt. | 0.73 | Ours: Tennis ball. |

(a)



| questions | answers | μ-scores | Input: Where did the person wash it briefly? In the sink. Ans: On the cutting board. |
|---|---|---|---|
| What did the person take from the drawer? | A cutting board. | 0.90 | Ground truth: Where did the person cut the roots of it? |
| What did the person also take out? | A knife. | 0.67 | HACRN: What did the person do? |
| What did the person retrieve from the cabinet? | One long leek. | 0.55 | Ours: Where did the person cut it? |

(b)

Figure 5: Visualization examples of experimental results. (a) is an answer generation result on YoutubeClip dataset, and (b) is a question generation result on TACoS-MultiLevel dataset.

Table 3: Ablation study results on TACoS-MultiLevel dataset. The upper part is the results of answer generation, and the lower part is the results of question generation.

| Method | BLEU-1 | ROUGE | METEOR |
|---|---|---|---|
| RICT$_{(lstm)}$ | 0.456 | 0.508 | 0.167 |
| RICT$_{(wo.ct)}$ | 0.449 | 0.497 | 0.159 |
| RICT$_{(wo.pi)}$ | 0.433 | 0.475 | 0.151 |
| RICT (ours) | **0.464** | **0.527** | **0.178** |
| RICT$_{(lstm)}$ | 0.724 | 0.745 | 0.362 |
| RICT$_{(wo.ct)}$ | 0.705 | 0.733 | 0.352 |
| RICT$_{(wo.pi)}$ | 0.711 | 0.740 | 0.357 |
| RICT (ours) | **0.733** | **0.748** | **0.367** |

have the similar modules proves the effectiveness of each part of our model.

We also show some visualization examples of experimental results in Figure 5. For the example (a), the progressive inference successfully stops at the second round of dialog, for the current $\mu$-score has exceeded the threshold and the former round of dialog can't bring more valuable information. The example (b) also shows a similar result on question generation, however, this time the inference stops at the first round for it still contains the related thing 'cutting board'. Part of the words with high attention by our model are shown in different colors. And both of the generated answer and question of our model in the examples are better than the compared model, which are much closer to the ground-truth.

## 5 Conclusion

In this paper, in order to have a better understanding of both dialog history and video contents, we propose a novel progressive inference mechanism for video dialog, which progressively updates query information until it is sufficient and unambiguous, or it has arrived at the beginning of the dialog history. We also design a cross-transformer module to tackle multi-modal fusion problem, which could learn more fine-grained interactions between the visual and textual information. And in addition to answer generation, we also consider question generation based on the dialog history and video content under the same framework, which is more challenging but significant for a complete video dialog system. The qualitative and quantitative experimental results on two large-scale video dialog datasets show the effectiveness of our method.

## Acknowledgments

This work was supported by Zhejiang Natural Science Foundation (LR19F020002, LZ17F020001), National Natural Science Foundation of China (61976185, 61572431), the Fundamental Research Funds for the Central Universities, Chinese Knowledge Center for Engineering Sciences and Technology and Microsoft Research Asia.

## References

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433.

David L Chen and William B Dolan. 2011. Collecting highly parallel data for paraphrase evaluation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 190–200.

Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José MF Moura, Devi Parikh, and Dhruv Batra. 2017a. Visual dialog. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1080–1089.

Abhishek Das, Satwik Kottur, José MF Moura, Stefan Lee, and Dhruv Batra. 2017b. Learning cooperative visual dialog agents with deep reinforcement learning. In *International Conference on Computer Vision*, pages 2970–2979.

Harm De Vries, Florian Strub, Sarath Chandar, Olivier Pietquin, Hugo Larochelle, and Aaron C Courville. 2017. Guesswhat?! visual object discovery through multi-modal dialogue. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4466–4475.

Jiyang Gao, Runzhou Ge, Kan Chen, and Ram Nevatia. 2018. Motion-appearance co-memory networks for video question answering. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Chiori Hori, Huda Alamri, Jue Wang, Gordon Winchern, Takaaki Hori, Anoop Cherian, Tim K Marks, Vincent Cartillier, Raphael Gontijo Lopes, Abhishek Das, et al. 2018. End-to-end audio visual scene-aware dialog using multimodal attention-based video features. *arXiv preprint arXiv:1806.08409*.

Unnat Jain, Svetlana Lazebnik, and Alexander G Schwing. 2018. Two can play this game: visual dialog with discriminative question generation and answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5754–5763.

Satwik Kottur, José MF Moura, Devi Parikh, Dhruv Batra, and Marcus Rohrbach. 2018. Visual coreference resolution in visual dialog using neural module networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 153–169.

Sang-Woo Lee, Yu-Jung Heo, and Byoung-Tak Zhang. 2018. Answerer in questioner's mind: Information theoretic approach to goal-oriented visual dialog. In *Advances in Neural Information Processing Systems*, pages 2580–2590.

Jiasen Lu, Anitha Kannan, Jianwei Yang, Devi Parikh, and Dhruv Batra. 2017. Best of both worlds: Transferring knowledge from discriminative learning to a generative visual dialog model. In *Advances in Neural Information Processing Systems*, pages 314–324.

Daniela Massiceti, N Siddharth, Puneet K Dokania, and Philip HS Torr. 2018. Flipdial: A generative model for two-way visual dialogue. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6097–6105.

Ramakanth Pasunuru and Mohit Bansal. 2018. Game-based video-context dialogue. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Anna Rohrbach, Marcus Rohrbach, Wei Qiu, Annemarie Friedrich, Manfred Pinkal, and Bernt Schiele. 2014. Coherent multi-sentence video description with variable level of detail. In *German conference on pattern recognition*, pages 184–195.

Paul Hongsuck Seo, Andreas Lehrmann, Bohyung Han, and Leonid Sigal. 2017. Visual reference resolution using attention memory for visual dialog. In *Advances in neural information processing systems*, pages 3719–3729.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, pages 3776–3784.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*, pages 3295–3301.

Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. 2015. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Qi Wu, Peng Wang, Chunhua Shen, Ian Reid, and Anton van den Hengel. 2018. Are you talking to me? reasoned visual dialog generation through adversarial learning. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 6106–6115.

Dongfei Yu, Jianlong Fu, Tao Mei, and Yong Rui. 2017. Multi-level attention networks for visual question answering. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, page 8.

Kuo-Hao Zeng, Tseng-Hung Chen, Ching-Yao Chuang, Yuan-Hong Liao, Juan Carlos Niebles, and Min Sun. 2017. Leveraging video descriptions to learn video question answering. In *AAAI*, pages 4334–4340.

Zhou Zhao, Xinghua Jiang, Deng Cai, Jun Xiao, Xiaofei He, and Shiliang Pu. 2018. Multi-turn video question answering via multi-stream hierarchical attention context network. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 3690–3696.

Zhou Zhao, Qifan Yang, Deng Cai, Xiaofei He, and Yueting Zhuang. 2017. Video question answering via hierarchical spatio-temporal attention networks. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

# Executing Instructions in Situated Collaborative Interactions

**Alane Suhr**
Cornell University
suhr@cs.cornell.edu

**Claudia Yan**
IBM
claudiab.yan@gmail.com

**Jacob Schluger***
Cornell University
jes543@cornell.edu

**Stanley Yu***
Columbia University
stanley.yu@columbia.edu

**Hadi Khader****
Intel
hadi.kh.khader@gmail.com

**Marwa Mouallem****
IBM
marwamouallem@gmail.com

**Iris Zhang**
Facebook
irisz@fb.com

**Yoav Artzi**
Cornell University
yoav@cs.cornell.edu

## Abstract

We study a collaborative scenario where a user not only instructs a system to complete tasks, but also acts alongside it. This allows the user to adapt to the system abilities by changing their language or deciding to simply accomplish some tasks themselves, and requires the system to effectively recover from errors as the user strategically assigns it new goals. We build a game environment to study this scenario, and learn to map user instructions to system actions. We introduce a learning approach focused on recovery from cascading errors between instructions, and modeling methods to explicitly reason about instructions with multiple goals. We evaluate with a new evaluation protocol using recorded interactions and online games with human users, and observe how users adapt to the system abilities.

## 1 Introduction

Sequential instruction scenarios commonly assume only the system performs actions, and therefore only its behavior influences the world state. This ignores the collaborative potential of such interactive scenarios and the challenges it introduces. When the user acts in the world as well, they can adapt to the system abilities not only by adopting simpler language, but also by deciding to accomplish tasks themselves. The system must then recover from errors as new instructions arrive and be robust to changes in the environment that are not a result of its own actions.

In this paper, we introduce CEREALBAR, a collaborative game with natural language instruction, and design modeling, learning, and evaluation methods for the problem of sequential instruction following in collaborative interactions. In CEREALBAR, two agents, a leader and a follower, move in a 3D environment and collect valid sets of cards to earn points. A valid set is a set of three cards with distinct color, shape, and count. The game is turn-based, and only one player can act in each turn. In addition to collecting cards, the leader sends natural language instructions to the follower. The follower's role is to execute these instructions. Figure 1 shows a snapshot from the game where the leader plans to pick up a nearby card (red square) and delegates to the follower two cards, one close and the other much further away. Before that, the leader planned ahead and asked the follower to move in preparation for the next set. The agents have different skills to incentivize collaboration. The follower has more moves per turn, but can only see from first-person view, while the leader observes the entire environment but has fewer moves. This makes natural language inter-

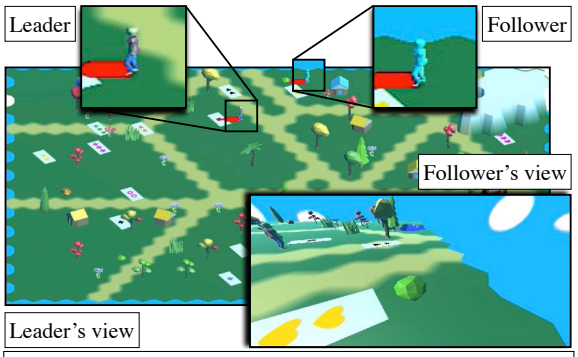


Figure 1: A snapshot from an interaction in CEREALBAR. The current instruction is in bold. The large image shows the entire environment. This overhead view is available only to the leader. The follower sees a first-person view only (bottom right). The zoom boxes (top) show the leader and follower.

*,**: Equal contribution. All work done at Cornell.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2119–2130, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

action key to success. We address the problem of mapping the leader instructions to follower actions. In addition to the collaborative challenges, this requires grounding natural language to resolve spatial relations and references to objects, reason about dependencies on the interaction history, react to the changing environment as cards appear and disappear, and generate actions.

CEREALBAR requires reasoning about the changing environment (e.g., when selecting cards) and instructions with multiple goals (e.g., selecting multiple cards). We build on the Visitation Prediction Network model (VPN; Blukis et al., 2018b), which casts planning as mapping instructions to the probability of visiting positions in the environment. Our new model generalizes the planning space of VPN to reason about intermediate goals and obstacles, and includes recurrent action generation for trajectories with multiple goals.

We collect 1,202 human-to-human games for training and evaluation. While our model could be trained from these recorded games only, it would often fail when an instruction would start at the wrong position because of an error in following the previous one. We design a learning algorithm that dynamically augments the data with examples that require recovering from such errors, and train our model to distinguish such recovery reasoning from regular instruction execution.

Evaluation with recorded games poses additional challenges. As agent errors lead to unexpected states, later instructions become invalid. Because measuring task completion from such states is meaningless, we propose *cascaded evaluation*, a new evaluation protocol that starts the agent at different points in the interaction and measures how much of the remaining instructions it can complete. In contrast to executing complete sequences or single instructions, this method allows to evaluate all instructions while still measuring the effects of error propagation.

We evaluate using both static recorded games and live interaction with human players. Our human evaluation shows users adapt to the system and use the agent effectively, scoring on average 6.2 points, compared to 12.7 for human players. Our data, code, and demo videos are available at lil.nlp.cornell.edu/cerealbar/.

## 2 Setup and Technical Overview

We consider a setup where two agents, a leader and a follower, collaborate. Both execute actions in a shared environment. The leader, additionally, instructs the follower using natural language. The leader goal is to maximize the task reward, and the follower goal is to execute leader instructions. We consider a turn-based version, where at each turn only one agent acts. We instantiate this scenario in CEREALBAR, a navigation card game (Figure 1), where a leader and follower move in an environment selecting cards to complete sets.[1]

**CEREALBAR Overview** The objective of CEREALBAR is to earn points by selecting valid sets of cards. A valid set has three cards with distinct color, shape, and count. When the only cards selected in the world form a valid set, the players receive a point, the selected cards disappear, three new cards are added randomly, and the number of remaining turns increases. The increase in turns decays for each set completed. An agent stepping on a card flips its selection status. The players form sets together. The follower has more steps per turn than the leader. This makes using the follower critical for success. The follower only sees a first-person view of the environment, preventing them from planning themselves, and requiring instructions to be sensible from the follower's perspective. The leader chooses the next target set, plans which of the two players should get which card, and instructs the follower. The follower can not respond to the leader, and should not plan themselves, or risk sabotaging the leader's plan, wasting moves and lowering their potential score. Followers mark an instruction as finished before observing the next one. This provides alignment between instructions and follower actions. In contrast to the original setup that we use for data collection, in our model (Section 4), we assume the follower has full observability, leaving the challenge of partial observability for future work. Appendix A provides further game design details.

**Problem Setup** We distinguish between the world state and the interaction state. Let $\mathcal{S}$ be the set of all world states, $\Gamma$ be the set of all interaction states, and $\mathcal{X}$ be the set of all natural language instructions. A world state $s \in \mathcal{S}$ describes the current environment. In CEREALBAR, the world state describes the spatial environment, the location of cards, whether they are selected or not, and the location of the agents. An interaction state $\gamma \in \Gamma$ is a tuple $\langle \bar{Q}, \alpha, \psi \rangle$. The first-in-first-out

[1] The name CEREALBAR does not carry special meaning. It was given to the project early on, and we came to like it. Our game is inspired by the card game Set.

queue $\bar{Q} = [\bar{x}_q, \ldots, \bar{x}_{q'}]$ contains the instructions $\bar{x}_i \in \mathcal{X}$ available to execute. The current instruction is the left-most instruction $\bar{x}_q$. The current turn-taker $\alpha \in \{\text{Leader}, \text{Follower}\}$ indicates the agent currently executing actions, and $\psi \in \mathbb{N}_{\geq 0}$ is the number of steps remaining in the current turn.

At each time step, the current turn-taker agent takes an action. An action may be the leader issuing an instruction, or either agent performing an action in the environment. Let $\mathcal{A} = \mathcal{A}_w \cup \{\texttt{DONE}\} \cup \mathcal{X}$ be the set of all actions. The set $\mathcal{A}_w$ includes the actions available to the agents in the environment. In CEREALBAR, this includes moving forward or backward, and turning left or right. Moving onto a card flips it selection status. `DONE` indicates completing the current instruction for the follower or ending the turn for the leader. An instruction action $a = \bar{x} \in \mathcal{X}$ can only be taken by the leader and adds the instruction $\bar{x}$ to the queue $\bar{Q}$. The effect of each action is determined by the transition function $\mathcal{T} : \mathcal{S} \times \Gamma \times \mathcal{A} \rightarrow \mathcal{S} \times \Gamma$, which is formally defined in Appendix B. Only world actions $a \in \mathcal{A}_w$ decrease the remaining steps $\psi$.

The goal of the leader is to maximize the total reward of the interaction. An interaction $\bar{I} = \langle (s_1, \gamma_1, a_1), \ldots, (s_{|\bar{I}|}, \gamma_{|\bar{I}|}, a_{|\bar{I}|}) \rangle$ is a sequence of state-action tuples, where $\mathcal{T}(s_i, \gamma_i, a_i) = (s_{i+1}, \gamma_{i+1})$. The reward function $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ assigns a numerical reward to a world state and an action. The total reward of an interaction $\bar{I}$ is $\sum_{i=0}^{|\bar{I}|} R(s_i, a_i)$. In CEREALBAR, the agents receive a reward of 1 when a valid set is selected.

**Task** Our goal is to learn a follower policy to execute the leader instructions. At time $t$, given the current world and interaction states $s_t$ and $\gamma_t$, and the interaction so far $\bar{I}_{<t}$, the follower policy $\pi(s_t, \gamma_t, \bar{I}_{<t})$ predicts the next action $a_t$.

**Model** We decompose the follower policy $\pi(s_t, \gamma_t, \bar{I}_{<t})$ to predicting a set of distributions over positions in the environment, including positions to visit, intermediate goals (e.g., cards to select), positions to avoid (e.g., cards not to touch), and positions that are not passable. These distribution are used in a second stage to generate a sequence of actions. Section 4 describes the model.

**Learning** We assume access to a set of $N$ recorded interactions $\{\bar{I}^{(i)}\}_{i=1}^{N}$, and create examples where each instruction is paired with a sequence of state-action tuples. We maximize the action-level cross entropy objective, and use two auxiliary objectives (Section 5). We first train each stage of the model separately, and then fine-tune them jointly. During fine-turning, we continuously generate additional examples using model failures. These examples help the agent to learn how to recover from errors in prior instructions.

**Evaluation** We measure correct execution of instructions and the overall game reward. We assume access to a test set of $M$ recorded interactions $\{\bar{I}^{(i)}\}_{i=1}^{M}$. We measure instruction-level and interaction-level performance, and develop *cascaded evaluation*, an evaluation protocol that provides a more graded measure than treating each interaction as a single example, while still accounting for error propagation (Section 6). Finally, we conduct online evaluation with human leaders.

## 3 Related Work

Goal-driven natural language interactions have been studied in various scenarios, including dialogue where only one side acts in the world (Anderson et al., 1991; Williams et al., 2013; Vlachos and Clark, 2014; de Vries et al., 2018; Kim et al., 2019; Hu et al., 2019), coordination for agreed selection of an object (He et al., 2017; Udagawa and Aizawa, 2019), and negotiation (Lewis et al., 2017; He et al., 2018). We focus on collaborative interactions where both the user and the system perform sequences of actions in the same environment. This allows the user to adapt to the language understanding ability of the system and balance between delegating goals to it and accomplishing them themselves. For example, a user may decide to complete a short but hard-to-describe task and delegate to the system a long but easy-to-describe one. In prior work, in contrast, recovery is limited to users paraphrasing their requests. The Cards corpus (Djalali et al., 2011, 2012; Potts, 2012) was used for linguistic analysis of collaborative bi-directional language interaction. The structure of collaborative interactions was also studied using Wizard-of-Oz studies (Lochbaum, 1998; Sidner et al., 2000; Koulouri and Lauria, 2009). In contrast, we focus on building agents that follow instructions. Ilinykh et al. (2019) present a corpus for the related task of natural language coordination in navigation. Collaboration has also been studied for emergent communication (e.g., Andreas et al., 2017; Evtimova et al., 2017).

Understanding sequences of natural language utterances has been addressed using semantic parsing (e.g., Miller et al., 1996; MacMahon et al., 2006; Chen and Mooney, 2011; Artzi and Zettlemoyer, 2013; Artzi et al., 2014; Long et al., 2016;

Iyyer et al., 2017; Suhr et al., 2018; Arkin et al., 2017; Broad et al., 2017). Interactions were also used for semantic parser induction (Artzi and Zettlemoyer, 2011; Thomason et al., 2015; Wang et al., 2016). These methods require hand-crafted symbolic meaning representation, while we use low-level actions (Suhr and Artzi, 2018). The interactions in our environment interleave actions of both agents with leader utterances, an aspect not addressed by these methods. Executing single instructions has been widely studied (e.g., Tellex et al., 2011; Duvallet et al., 2013; Misra et al., 2017, 2018; Anderson et al., 2018; Blukis et al., 2018a,b; Chen et al., 2019). The distinction we make between actions specified in the instruction and implicit recovery actions is similar to how Artzi and Zettlemoyer (2013) use implicit actions for single instructions. Finally, our model is based on the VPN model of Blukis et al. (2018b). While we assume full observability, their original work did not. This indicates that our model is likely to generalize well to partially observable scenarios.

## 4 Model

We use a two-stage model for the follower policy $\pi(s_t, \gamma_t, \bar{I}_{<t})$, where $s_t$ is a world state, $\gamma_t$ is an interaction state, and $\bar{I}_{<t}$ is the interaction history. The instruction $\bar{x}$ that is the first in the queue $\bar{Q}_t$, which is part of $\gamma_t$, is the currently executed instruction. In our model, we assume the follower observes the entire environment. First, we map $\bar{x}$ and $s_t$ to distributions over locations in the environment, including what locations to visit and what are the goals. These distributions are considered as an execution plan, and are used to generate a sequence of actions in the second stage. The distribution can also be used to easily easily visualize the agent plan. The first stage is used when starting a new instruction, and the predicted distributions are re-used for all actions for that instruction. Figure 2 illustrates the architecture and the distributions visualization. The two-stage approach was introduced by Blukis et al. (2018b). We generalize its planning space and add a recurrent action generator for execution.

**Input Representation** The inputs to the first stage are the instruction $\bar{x}$ and the world state $s_t$. We generate feature maps for both. We use a learned embedding function $\phi^{\mathcal{X}}$ and a bi-directional recurrent neural network (RNN; Elman, 1990) with a long short-term memory cell (LSTM; Hochreiter and Schmidhuber, 1997) $\text{RNN}^{\mathcal{X}}$ to map $\bar{x}$ to a vector $\bar{\mathbf{x}}$. The world state $s_t$ is a 3D tensor that encodes the properties of each position. The dimensions of $s_t$ are $P \times W \times H$, where $P$ is the number of properties, and $W$ and $H$ are the environment width and height. Each of the $W \times H$ positions is represented in $s_t$ as a binary vector of length $P$. For example, a position with a red hut will have 1's for the *red* and *hut* dimensions and 0's for all other dimensions. We map the world state to a tensor feature map $\mathbf{F}_0$ by embedding $s_t$ and processing it using the text representation $\bar{\mathbf{x}}$. We use a learned embedding function $\phi^{\mathcal{S}}$ to map each position vector to a dense embedding of size $N_s$ by summing embeddings of each of the position's properties. The embeddings are combined to a tensor $\mathbf{S}$ of dimension $N_s \times W \times H$ representing a featurized global view of the environment. We create a text-conditioned state representation by creating a kernel $\mathbf{K}_s$ and convolving with it over $\mathbf{S}$. We use a linear transformation to create $\mathbf{K}_s = \mathbf{W}_s \bar{\mathbf{x}} + \mathbf{b}_s$, where $\mathbf{W}_s$ and $\mathbf{b}_s$ are learned weights. We reshape $\mathbf{K}_s$ to a $1 \times 1$ convolution kernel with $N_{s'}$ output channels, and compute $\mathbf{S}' = \mathbf{S} * \mathbf{K}_s$. We concatenate $\mathbf{S}$ and $\mathbf{S}'$ along the channel dimension and rotate and center so the follower position is at center pixel to generate $\mathbf{F}_0$.[2]

**Stage 1: Plan Prediction** We treat plan generation as predicting distributions over positions $\rho$ in the environment. There are $W \times H$ possible positions. We predict four distributions: (a) $p(\rho \mid s_t, \bar{x})$, the probability of visiting $\rho$ while executing the instruction $\bar{x}$; (b) $p(\text{GOAL} = 1 \mid \rho, s_t, \bar{x})$, the binary probability that $\rho$ is a goal (i.e., GOAL $= 1$ when containing a card to select); (c) $p(\text{AVOID} = 1 \mid \rho, s_t, \bar{x})$, the binary probability that the agent must not pass in $\rho$ (i.e., AVOID $= 1$ when it contains a card that should not be touched); and (d) $p(\text{NOPASS} = 1 \mid \rho, s_t, \bar{x})$, the binary probability the agent cannot pass in $\rho$ (i.e., NOPASS $= 1$ when it contains another object).

We use LINGUNET (Misra et al., 2018) to predict the distributions. The inputs to LINGUNET are the instruction embedding $\bar{\mathbf{x}}$ and featurized world state $\mathbf{F}_0$, which is relative to the agent's frame of reference. The output are four matrices, each of dimension $W \times H$ corresponding to the environment. LINGUNET is formally defined in Misra et al. (2018) and Appendix D. Roughly

[2] Appendix D describes the relationship between the environment representations and the agent's initial and current orientation.

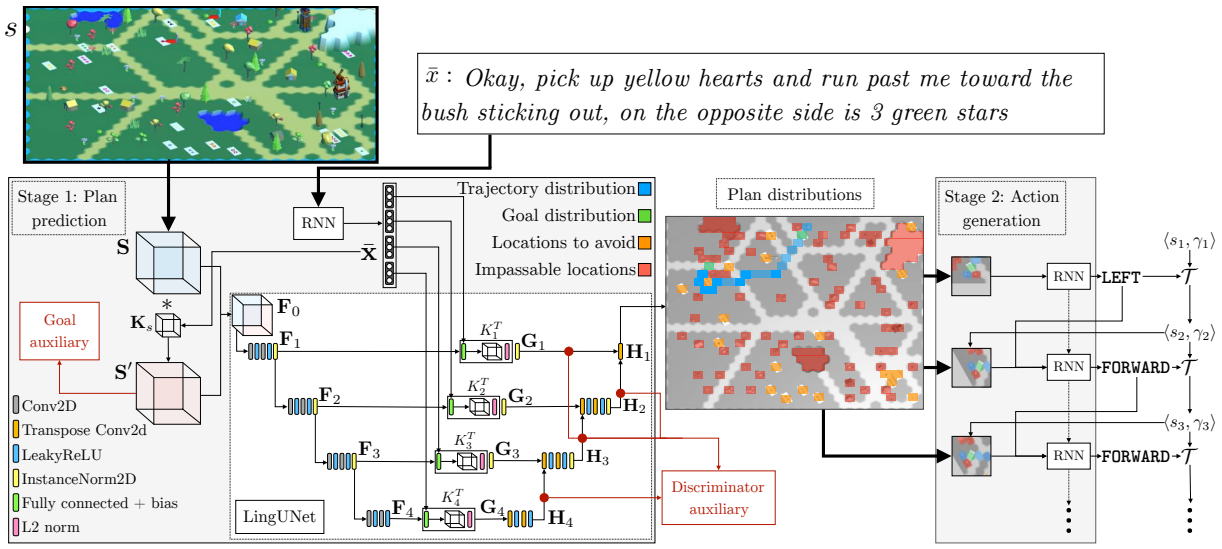


Figure 2: Illustration of the model architecture. Given the instruction $\bar{x}$ and the world state $s$, we compute $\mathbf{F}_0$ from the embeddings of the instruction $\bar{\mathbf{x}}$ and environment $\mathbf{S}$. We use LINGUNET to predict four distributions, which are visualized over the map (grayscaled to emphasize the distributions). We show three action generation steps. Each step receives the map cropped around the agent and the previous action, and outputs the next action.

speaking, LINGUNET reasons about the environment representation $\mathbf{F}_0$ at $L$ levels. First, $\mathbf{F}_0$ is used to generate feature maps of decreasing size $\mathbf{F}_j$, $j = 1 \dots L$ using a series of convolutions. We create convolution kernels from the instruction representation $\bar{\mathbf{x}}$, and apply them to the feature maps $\mathbf{F}_j$ to generate text-conditioned feature maps $\mathbf{G}_j$. Finally, feature maps of increasing size $\mathbf{H}_j$ are generated using a series of $L$ deconvolutions. The last deconvolution generates a tensor of size $4 \times W \times H$ with a channel for each of the four distributions. We use a softmax over one channel to compute $p(\rho \mid s_t, \bar{x})$. Because the other distributions are binary, we use a sigmoid on each value independently for the other channels. When computing $p(\text{GOAL} = 1 \mid \rho, s_t, \bar{x})$ and $p(\text{AVOID} = 1 \mid \rho, s_t, \bar{x})$ we mask positions without objects that can be changed (i.e., positions without cards) to assign them zero probability.

**Stage 2: Action Generation** We use the four distributions to generate a sequence of actions. We concatenate the distributions channel-wise to a tensor $\mathbf{P} \in \mathbb{R}^{4 \times W \times H}$. We use a forward LSTM RNN to predict a sequence of actions. At each prediction step $t$, we rotate, transform, and crop $\mathbf{P}$ to generate the egocentric tensor $\mathbf{P}'_t \in \mathbb{R}^{N' \times C \times C}$, where the agent is always at the center and facing in the same direction, such that $\mathbf{P}'_t$ is relative to the agent's current frame of reference. The input to the action generation RNN at time $t$ is:

$$\begin{aligned} \mathbf{p}'_t &= \text{vec}(\text{NORM}(\text{RELU}(\text{CNN}^P(\mathbf{P}'_t)))) \\ \mathbf{p}_t &= \text{RELU}(\mathbf{W}_2^P[\mathbf{p}'_t; \text{RELU}(\mathbf{W}_1^P \mathbf{p}'_t + \mathbf{b}_1^P)] + \mathbf{b}_2^P) \ , \end{aligned}$$

where $\text{CNN}^P$ is a convolutional layer, RELU is a non-linearity, NORM is instance normalization (Ulyanov et al., 2017), and $\mathbf{W}_1^P$, $\mathbf{W}_2^P$, $\mathbf{b}_1^P$, $\mathbf{b}_2^P$ are learned weights. The action probability is:

$$\begin{aligned} \mathbf{h}_t &= \text{RNN}^{\mathcal{A}}(\mathbf{h}_{t-1}, [\phi^{\mathcal{A}}(a_{t-1}); \mathbf{p}_t]) \\ p(a) &\propto \exp(\mathbf{W}^{\mathcal{A}}[\mathbf{h}_t; \mathbf{p}_t] + \mathbf{b}^{\mathcal{A}}) \ , \end{aligned} \quad (1)$$

where $\text{RNN}^{\mathcal{A}}$ is an LSTM RNN, $\phi^{\mathcal{A}}$ is a learned action embedding function, $a_0$ is a special `START` action, and $\mathbf{W}^{\mathcal{A}}$ and $\mathbf{b}^{\mathcal{A}}$ are learned. During inference, we assign zero probabilities to actions $a$ when $\mathcal{T}_w(s_t, a)$ is invalid (Appendix B), for example when an agent would move into an obstacle.

## 5 Learning

We assume access to a set of $N$ recorded interactions $\{\bar{I}^{(i)}\}_{i=1}^N$. We generate instruction-level examples $\mathcal{D} = \bigcup_{i=1}^N \{\bar{I}^{(i,j)}\}_{j=1}^{M^{(i)}}$, where $M^{(i)}$ is the number of examples from $\bar{I}^{(i)}$. Each $\bar{I}^{(i,j)} = \langle (s_1^{(i,j)}, \gamma_1^{(i,j)}, a_1^{(i,j)}), \dots, (s_k^{(i,j)}, \gamma_k^{(i,j)}, a_k^{(i,j)}) \rangle$ is a subsequence of tuples in $\bar{I}^{(i)}$, where $a_1^{(i,j)}$ is the first action the follower takes after observing the $j$-th instruction in $\bar{I}^{(i)}$, and $a_k^{(i,j)}$ is the `DONE` action completing that instruction. We first estimate the parameters for plan prediction $\theta_1$ and action generation $\theta_2$ separately (Section 5.1), and then fine-tune jointly with data augmentation (Section 5.2).

### 5.1 Pretraining

**Stage 1: Plan Prediction** The input of Stage 1 is the world state $s_1$ and the instruction $\bar{x}$ at the head of the queue $\bar{Q}$.[3] We generate labels for the four

[3] We omit example indices for succinctness.

output distributions using $\bar{I}^{(i,j)}$. The visitation distribution $p(\rho \mid s_1, \bar{x})$ label is proportional to number of states $s_t \in \bar{I}^{(i,j)}$ where the follower is in position $\rho$. The goal and avoidance distributions model how the agent plans to manipulate parts of its environment to achieve the specified goals, but avoid manipulating other parts. In CEREALBAR, this translates to changing the status of cards, or avoiding doing so. For $p(\text{GOAL} = 1 \mid \rho, s_1, \bar{x})$, we set the label to 1 for all $\rho$ that contain a card that the follower changed its selection status in $\bar{I}^{(i,j)}$, and 0 for all other positions. Similarly, for the avoidance distribution $p(\text{AVOID} = 1 \mid \rho, s_1, \bar{x})$, the label is 1 for all $\rho$ that have cards that the follower does not change during the interaction $\bar{I}^{(i,j)}$. Finally, for $p(\text{NOPASS} = 1 \mid \rho, s_1, \bar{x})$, the label is 1 for all positions the agent cannot move onto, and zero otherwise. We define four cross-entropy losses: visitation $\mathcal{L}_V$, goal $\mathcal{L}_G$, avoidance $\mathcal{L}_A$, and no passing $\mathcal{L}_P$. We also use an auxiliary cross-entropy goal-prediction loss $\mathcal{L}_{G'}$ using a probability $p'_G(\text{GOAL} = 1 \mid \rho, s_1, \bar{x})$ we predict from the pre-LINGUNET representation $\mathbf{S}'$ by classifying each position. The complete loss is a weighted sum with coefficients:[4]

$$\mathcal{L}_1(\theta_1) = \lambda_V \mathcal{L}_V(\theta_1) + \lambda_G \mathcal{L}_G(\theta_1) + \lambda_A \mathcal{L}_A(\theta_1) + \lambda_P \mathcal{L}_P(\theta_1) + \lambda_{G'} \mathcal{L}_{G'}(\theta_1) \ .$$

**Stage 2: Action Generation** We use the gold distribution to create the input $\mathbf{P}$, and optimize towards the annotated set of actions using teacher forcing (Williams and Zipser, 1989). We compute the loss only over actions taken by the follower:

$$\mathcal{L}_2(\theta_2) = -\sum_{t=1}^{n} \mathbb{1}_{\alpha_t = \text{Follower}} p(a_t) \ ,$$

where $p(a_t)$ is computed by Equation 1.

### 5.2 Fine-tuning with Example Aggregation

Simply combining the separately-trained networks together results in low performance. We perform additional fine-tuning with the two stages combined, and introduce a data augmentation method to learn to recover from error propagation.

**Error Propagation** Executing a sequence of instructions is susceptible to error propagation, where an agent fails to correctly complete an instruction, and because of it also fails on the following ones. While the collaborative, turn-switching setup allows the leader to adjust their plan following a follower mistake, leaders often strategically issue multiple instructions to use the available follower steps optimally. Given an agent failure, subsequent instructions may not align with the state of the world resulting from the follower's error. In supervised learning, we do not have the opportunity to learn to recover from such errors, even when it is relatively simple. This usually requires exploration. However, conventional frameworks like reinforcement learning (RL) or imitation learning (IL) are poorly suitable. In a live interaction, when an agent makes a mistake (e.g., selecting the wrong card), the leader is likely to adjust their actions. Because of this, in a recorded interaction, which contains the leader actions following a correct execution, it is not possible to reliably compute an RL reward for states following erroneous executions. For similar reasons, we cannot compute an IL oracle.

We identify two classes of erroneous states in CEREALBAR: (a) not selecting the correct set of cards; and (b) finishing with the right card selection, but stopping at the wrong position.[5] Case (a) requires to modify the model, for example to know when to skip instructions that refer to a state that is no longer possible. We leave this case for future work. We address case (b) by augmenting the data with new examples that are aggregated during learning. Our process is similar to DAGGER (Ross et al., 2011). We alternate between: (a) collecting new training examples using a heuristic oracle, and (b) performing model updates. We generate training examples that demonstrate recovery by starting in an incorrect initial position for an instruction, having arrived there by executing the previous instruction. We train our model to distinguish between the reasoning required for generating implicit actions to correct errors, and explicit actions directly mentioned in the instruction.

**Learning with Example Aggregation** We alternate between aggregating a new set of recovery examples $\mathcal{D}'$ and updating our parameters. At each epoch, we first use the current policy to create new training examples. We run inference for each example $\bar{I}^{(i,j)}$ in $\mathcal{D}$, the original training set, using the current policy.[6] We compare the state $s'$ at the end of execution to the final state in $\bar{I}^{(i,j)}$ to generate an error-recovery example $\bar{I}'^{(i,j+1)}$ for the subsequent example $\bar{I}^{(i,j+1)}$. We only generate such examples if the position or rotation of the

[4] Additional details are in Appendix E.1.

[5] See Appendix E.2 for further discussion of the two cases.

[6] We do not perform inference for the last instruction in an interaction, as there is no subsequent example for which to generate a new example.

agent are different, and there are no other difference between the states. Starting from $s'$, we generate the shortest-path sequence of actions that: (a) changes the cards as specified in $\bar{I}^{(i,j+1)}$, and (b) executes DONE in the same position as in $\bar{I}^{(i,j+1)}$. We then create $\bar{I}'^{(i,j+1)}$ using $\bar{I}^{(i,j+1)}$ and the new sequence of state-action pairs, and add it to $\mathcal{D}'$.[7]

Given the original set of examples $\mathcal{D}$ and the aggregate examples $\mathcal{D}'$ we update our model parameters. We randomly sample without replacement at most $\sum_{i=1}^{N} M^{(i)}$ examples, the size of $\mathcal{D}$, from $\mathcal{D}'$. We use all the examples in $\mathcal{D}$ and the sampled examples to do a single parameter update epoch. We limit the number of examples from $\mathcal{D}'$ to maintain the effect of the original data.

**Optimizing with Implicit Action Prediction** The examples we generate during aggregation often include sequences of state-action pairs that do not align with the instruction, for example when a mentioned spatial relation is incorrect from the new starting position. Such examples require reasoning differently about the text and world state than with the original data. We identify such examples in $\mathcal{D}'$ by comparing their follower starting position to the starting position in the original corresponding example in $\mathcal{D}$. If the distance is over two, we treat the examples as requiring implicit actions (Artzi and Zettlemoyer, 2013). All other examples, including all original examples in $\mathcal{D}$ are considered as not requiring implicit reasoning. We encourage the model to reason differently about these examples with a discriminator that classifies if the example requires implicit reasoning or not using the internal activations of LINGUNET.

The discriminator classifies each of the $L$ layers in LINGUNET for implicit reasoning. The goal is to encourage implicit reasoning at all levels of reasoning in the first stage. The probability of implicit reasoning for each LINGUNET layer $l$ is:

$$p(\text{IMPLICT} = 1 \mid l, s_1, \bar{x}) = \begin{cases} \sigma(\text{AVGPOOL}(\mathbf{G}_1 * \mathbf{K}_1^{\text{IMP}})) & l = 1 \\ \sigma(\text{AVGPOOL}(\mathbf{H}_l * \mathbf{K}_l^{\text{IMP}})) & l > 1 \end{cases} ,$$

where $\mathbf{K}_l^{\text{IMP}}$ are $1 \times 1$ learned kernels and AVGPOOL does average pooling. We define a cross-entropy loss $\mathcal{L}_{\text{IMP}}$ that averages across the $L$ layers. The complete fine-tuning loss is:

$$\mathcal{L}(\theta_1, \theta_2) = \mathcal{L}_1(\theta_1) + \mathcal{L}_2(\theta_2) + \lambda_{\text{IMP}} \mathcal{L}_{\text{IMP}}(\theta_1) \ .$$

[7] Appendix E.2 describes this process.

## 6 Cascaded Evaluation

Sequential instruction scenarios are commonly evaluated using recorded interactions by executing individual instructions or executing complete interactions starting from their beginning (e.g., Chen and Mooney, 2011; Long et al., 2016). Both have limitations. Instruction-level metrics ignore error propagation, and do not accurately reflect the system's performance. In contrast, interaction-level metrics do consider error propagation and capture overall system performance well. However, they poorly utilize the test data, especially when performance is relatively low. When early failures lead to unexpected world states, later instructions become impossible to follow, and measuring performance on them is meaningless. For example, with our best-performing model, 82% of development instructions become impossible due cascading errors when executing complete interactions.

The two measures may also fail to distinguish models. For example, consider an interaction with three instructions. Two models, A and B, successfully execute the third instruction in isolation, but fail on the two others. They also both fail when executing the entire interaction starting from the beginning. According to common measures, the models are equal. However, if model B can actually recover from failing on the second instruction to successfully execute the third, it means it is better than model A. Both metrics fail to reflect this.

We propose *cascaded evaluation*, an evaluation protocol for sequential instruction using static corpora. Our method utilizes all instructions during testing, while still accounting for the effect of error propagation. Unlike instruction-level evaluation, cascaded evaluation executes the instructions in sequence. However, instead of starting of starting only from the start state of the first instruction, we create separate examples for starting from the starting state of each instruction in the interaction and continuing until the end of the interaction. For example, given a sequence of three instructions $\langle 1, 2, 3 \rangle$ we will create three examples: $\langle 1, 2, 3 \rangle$, $\langle 2, 3 \rangle$, and $\langle 3 \rangle$. To evaluate performance in CEREALBAR, we compute two statistics using cascaded evaluation: the proportion of the remaining instructions followed successfully, and the proportion of potential points scored. We only consider the remaining instructions and points left to achieve in the example. For example, for the sequence $\langle 2, 3 \rangle$, we will subtract any points achieved

before the second instruction to compute the proportion of potential points scored. Appendix F describes cascaded evaluation formally.

## 7 Experimental Setup

**Data** We collect 1,202 human-human interactions using Mechanical Turk, split into train (960 games), development (120), and test (122). Appendix C details data collection and statistics.

**Recorded Interactions Metrics** We evaluate instruction-level, interaction-level, and cascaded (Section 6) performance. We allow the follower ten steps per turn, and interleave the actions taken by the leader during each turn in the recorded interaction. Instruction execution often crosses turns. At the instruction-level, we evaluate the mean *card state accuracy* comparing the state of the cards after inference with the correct card state, *environment state accuracy* comparing both cards and the agent's final position, and *action sequence accuracy* comparing the generated action sequence with the correct action sequence. For complete interactions, we measure mean *full game points*. Finally, for cascaded evaluation, we measure the mean proportion of instructions correctly executed and of possible points scored.

**Human Evaluation** We perform evaluation with human leaders, comparing our model and human followers. Workers are told they will work with a human or an automated follower, but are not told which in each game. We evaluate both human (105 games) and automated agents at the same time (109 games). We evaluate the game scores, and also elicit free-form feedback.

**Systems** We evaluate three systems: (a) the full model; (b) SEQ2SEQ+ATTN:[8] sequence-to-sequence with attention; and (c) a static oracle that executes the gold sequence of actions in the recorded interaction. We report mean and standard deviation across three trials for development results. We ablate model and learning components, and additionally evaluate the action generator with access to gold plans.[9] On the test set and for human evaluation, we use the model with the highest proportion of points scored. We provide implementation and learning details in Appendix G.

## 8 Results

Table 1 shows development and test results, including ablations. We consider the proportion of points scored computed with cascaded evaluation as the main metric. Our complete approach significantly outperforms SEQ2SEQ+ATTN. Key to this difference is the added structure within the model and the direct supervision on it. The results also show the large remaining gap to the static oracle.[10]

Our results show how considering error propagation for all available instructions in cascaded evaluation guides different design choices. For example, example aggregation and the implicit discriminator lower performance according to instruction-level metrics, which do not consider error propagation. We see a similar trend for the implicit discriminator when looking at full game points, an interaction-level metric that does not account for performance on over 80% of the data because of error propagation. In contrast, the proportion of points scored computed using cascaded evaluation shows the benefit of both mechanisms.

Our ablations demonstrate the benefit of each model component. All four distributions help. Without the trajectory distribution (– Trajectory distribution), performance drops almost to the level of SEQ2SEQ+ATTN. This indicates the action predictor is not robust enough to construct a path given only the three other disjoint distributions. While the predicted trajectory distribution contains all information necessary to reach the correct cards and goal location, the other three distributions further improve performance. This is likely because redundancy with the trajectory distribution makes the model more robust to noisy predictions in the trajectory distribution. For example, the GOAL distribution guides the agent to move towards goal cards even if the predicted trajectory is discontinuous. The action generation recurrence is also critical (– Action recurrence), allowing the agent to keep track of which locations it already passed when navigating complex paths that branch, loop, or overlap with themselves.

While we observe that each stage separately performs well after pretraining, combining them without fine-tuning (– Fine-tuning) leads to low performance because of the shift in the second stage input. Providing the gold distributions to the action generator illustrates this (+ Gold plan). Removing early goal auxiliary loss $\mathcal{L}_{G'}$ (Section 5.1) leads to a slight drop in performance on all metrics (– Early goal auxiliary). Learning with aggregated recovery examples helps the

[8] This baseline is similar to Mei et al. (2016).

[9] We do not measure interaction-level metrics with gold plans as they are only available for the gold start positions.

[10] Appendix F explains the static oracle performance.

| System | Card State Acc. | Env. State Acc. | Action Seq. Accuracy | Full Game Points | Prop. Instr. Followed | Prop. Points Scored |
|---|---|---|---|---|---|---|
| **Development Results & Ablation Analysis** | | | | | | |
| Full model | 58.2±0.5 | 32.6±0.8 | 15.8±0.5 | 0.66±0.1 | 20.5±1.2 | 18.1±0.8 |
| – Trajectory distribution | 38.5±2.7 | 10.1±2.7 | 5.5±2.6 | 0.29±0.02 | 10.0±0.9 | 7.9±0.7 |
| – GOAL distribution | 56.2±1.5 | 30.8±0.4 | 14.9±0.3 | 0.66±0.09 | 17.9±1.0 | 15.9±1.3 |
| – AVOID distribution | 57.0±0.3 | 32.6±1.6 | 15.4±1.3 | 0.63±0.04 | 18.8±1.5 | 17.8±0.7 |
| – NOPASS distribution | 59.2±0.5 | 32.0±0.8 | 15.0±0.5 | 0.70±0.03 | 18.4±0.9 | 16.6±0.9 |
| – Action recurrence | 42.3±1.5 | 16.7±1.2 | 10.0±0.7 | 0.42±0.03 | 12.8±1.7 | 10.7±0.5 |
| – Fine-tuning | 43.6±1.9 | 8.5±1.1 | 4.5±0.5 | 0.65±0.09 | 14.1±1.3 | 9.2±0.9 |
| – Early goal auxiliary | 57.2±2.3 | 31.2±1.7 | 14.9±1.6 | 0.65±0.05 | 17.9±1.1 | 16.5±0.7 |
| – Example aggregation | 59.4±1.8 | 32.0±1.0 | 15.7±0.6 | 0.65±0.09 | 20.4±1.4 | 16.5±0.4 |
| – Implicit discriminator | 57.5±2.1 | 32.7±1.0 | 16.4±0.3 | 0.70±0.02 | 18.8±1.8 | 16.7±0.6 |
| – Instructions | 15.5±1.5 | 2.7±1.5 | 1.2±1.2 | 0.24±0.07 | 4.4±1.0 | 4.6±0.7 |
| + Gold plan | 87.4±0.5 | 80.2±0.2 | 63.4±0.2 | – | – | – |
| SEQ2SEQ+ATTN | 35.3±0.8 | 11.1±0.5 | 9.4±0.5 | 0.20±0.04 | 8.8±0.1 | 6.3±0.1 |
| Static oracle | 99.7 | 99.7 | 100.0 | 6.58 | 98.5 | 97.9 |
| **Test Results** | | | | | | |
| Full model | 58.4 | 32.1 | 15.6 | 0.62 | 15.4 | 17.9 |
| SEQ2SEQ+ATTN | 37.3 | 10.8 | 8.5 | 0.22 | 8.7 | 6.5 |
| Static oracle | 99.7 | 99.7 | 100.0 | 6.66 | 96.8 | 95.6 |

Table 1: Development and test results on all systems, including ablation results.

model to learn to recover from errors in previous instructions and increases the proportion of points scored (– Example aggregation). However, without the implicit reasoning discriminator (– Implicit discriminator), the additional examples make learning too difficult, and do not help. Finally, removing the language input (– Instructions) significantly decreases performance, showing that the data is relatively robust to observational biases and language is necessary for the task.

In the human evaluation, we observe a mean of 6.2 points (max of 14) with our follower model, compared to 12.7 (max of 20) with human followers. While this shows there is much room for improvement, it illustrates how human leaders adapt and use the agent effectively. One key strategy of adaptation is to use simplified language that fits the model better. This includes shorter instructions, with 8.5 tokens on average with automated followers compared to 12.3 with humans, and a smaller vocabulary, 578 word types with automated followers and 1037 with humans. In general, human leaders commented that they are able to easily distinguish between automated and human followers, and find working with the automated agent frustrating.

## 9 Discussion

Our human evaluation highlights several directions for future work. While human leaders adapt to the agent, scoring up to 14 points, there remains a significant gap to collaborations with human followers. Reported errors include getting stuck behind objects, selecting unmentioned cards, going in the wrong direction, and ignoring instructions. At least one worker developed a strategy that took advantage of the agent's full observability, writing instructions with only simple card references. An important direction for future work is to remove our full observability assumption. Other future directions include experimenting with using the interaction history, expanding the learning example aggregation to error cases beyond incorrect start positions, and making agent reasoning interpretable to reduce user frustration. CEREALBAR also provides opportunities to study pragmatic reasoning for language understanding (Andreas and Klein, 2016; Fried et al., 2018; Liang et al., 2019). While we currently focus on language understanding by limiting the communication to be unidirectional, bidirectional communication would allow for more natural and efficient collaborations (Potts, 2012; Ilinykh et al., 2019). CEREALBAR could be easily adapted to allow bidirectional communication, and provide a platform to study challenges in language generation.

## Acknowledgments

This research was supported by the NSF under Grant No. 1750499, an AI2 KSC Award, a Workday Faculty Award, a Google Faculty Award, Unity, and an Amazon Cloud Credits Grant. This material is based on work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1650441. We thank Valts Blukis, Jin Sun, and Mark Yatskar for comments and suggestions, the workers who participated in our data collection, and the reviewers.

## References


Anne H. Anderson, Miles Bader, Ellen Gurman Bard, Elizabeth Boyle, Gwyneth Doherty, Simon Garrod, Stephen Isard, Jacqueline Kowtko, Jan McAllister, Jim Miller, Catherine Sotillo, Henry S. Thompson, and Regina Weinert. 1991. The HCRC map task corpus. *Language and Speech*, 34.

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian D. Reid, Stephen Gould, and Anton van den Hengel. 2018. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3674–3683.

Jacob Andreas, Anca Dragan, and Dan Klein. 2017. Translating neuralese. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 232–242.

Jacob Andreas and Dan Klein. 2016. Reasoning about pragmatics with neural listeners and speakers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1173–1182.

Jacob Arkin, Matthew R. Walter, Adrian Boteanu, Michael E. Napoli, Harel Biggie, Hadas Kress-Gazit, and Thomas M. Howard. 2017. Contextual awareness: Understanding monologic natural language instructions for autonomous robots. In *IEEE International Symposium on Robot and Human Interactive Communication*, pages 502–509.

Yoav Artzi, Dipanjan Das, and Slav Petrov. 2014. Learning compact lexicons for CCG semantic parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1273–1283.

Yoav Artzi and Luke Zettlemoyer. 2011. Bootstrapping semantic parsers from conversations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 421–432.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association of Computational Linguistics*, 1:49–62.

Valts Blukis, Nataly Brukhim, Andrew Bennett, Ross A. Knepper, and Yoav Artzi. 2018a. Following high-level navigation instructions on a simulated quadcopter with imitation learning. In *Proceedings of the Robotics: Science and Systems Conference*.

Valts Blukis, Dipendra Misra, Ross A. Knepper, and Yoav Artzi. 2018b. Mapping navigation instructions to continuous control actions with position visitation prediction. In *Proceedings of the Conference on Robot Learning*.

Alexander Broad, Jacob Arkin, Nathan Ratliff, Thomas Howard, and Brenna Argall. 2017. Real-time natural language corrections for assistive robotic manipulators. *The International Journal of Robotics Research*, 36(5-7):684–698.

David L. Chen and Raymond J. Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *Proceedings of the National Conference on Artificial Intelligence*.

Howard Chen, Alane Suhr, Dipendra Misra, Noah Snavely, and Yoav Artzi. 2019. Touchdown: Natural language navigation and spatial reasoning in visual street environments. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Alex Djalali, David Clausen, Sven Lauer, Karl Schultz, and Christopher Potts. 2011. Modeling expert effects and common ground using questions under discussion. In *AAAI Fall Symposium: Building Representations of Common Ground with Intelligent Agents*.

Alex Djalali, Sven Lauer, and Christopher Potts. 2012. Corpus evidence for preference-driven interpretation. In *Logic, Language and Meaning*, pages 150–159.

Felix Duvallet, Thomas Kollar, and Anthony Stentz. 2013. Imitation learning for natural language direction following through unknown environments. In *IEEE International Conference on Robotics and Automation*, pages 1047–1053.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14:179–211.

Katrina Evtimova, Andrew Drozdov, Douwe Kiela, and Kyunghyun Cho. 2017. Emergent communication in a multi-modal, multi-step referential game. In *Proceedings of the International Conference on Learning Representations*.

Daniel Fried, Jacob Andreas, and Dan Klein. 2018. Unified pragmatic models for generating and following instructions. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1951–1963.

He He, Anusha Balakrishnan, Mihail Eric, and Percy Liang. 2017. Learning symmetric collaborative dialogue agents with dynamic knowledge graph embeddings. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 1766–1776.

He He, Derek Chen, Anusha Balakrishnan, and Percy Liang. 2018. Decoupling strategy and generation in negotiation dialogues. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2333–2343.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9.

Hengyuan Hu, Denis Yarats, Qucheng Gong, Yuandong Tian, and Mike Lewis. 2019. Hierarchical decision making by generating and following natural language instructions. *CoRR*, abs/906.00744.

Nikolai Ilinykh, Sina Zarrieß, and David Schlangen. 2019. MeetUp! A corpus of joint activity dialogues in a visual environment. *CoRR*, abs/1907.05084.

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 1821–1831.

Jin-Hwa Kim, Nikita Kitaev, Xinlei Chen, Marcus Rohrbach, Yuandong Tian, Dhruv Batra, and Devi Parikh. 2019. CoDraw: Collaborative Drawing as a Testbed for Grounded Goal-driven Communication. *CoRR*, abs/1704.04517.

Theodora Koulouri and Stanislao Lauria. 2009. Exploring miscommunication and collaborative behaviour in human-robot interaction. In *Proceedings of the SIGDIAL Conference*, pages 111–119.

Mike Lewis, Denis Yarats, Yann Dauphin, Devi Parikh, and Dhruv Batra. 2017. Deal or no deal? End-to-end learning of negotiation dialogues. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2443–2453.

Claire Liang, Julia Proft, Erik Andersen, and Ross A. Knepper. 2019. Implicit communication of actionable information in human-AI teams. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pages 95:1–95:13.

Karen E. Lochbaum. 1998. A collaborative planning model of intentional structure. *Computational Linguistics*, 24(4):525–572.

Reginald Long, Panupong Pasupat, and Percy Liang. 2016. Simpler context-dependent logical forms via model projections. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 1456–1465.

Matthew MacMahon, Brian Stankiewics, and Benjamin Kuipers. 2006. Walk the talk: Connecting language, knowledge, action in route instructions. In *Proceedings of the National Conference on Artificial Intelligence*.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Scott Miller, David Stallard, Robert Bobrow, and Richard Schwartz. 1996. A fully statistical approach to natural language interfaces. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 55–61.

Dipendra Misra, Andrew Bennett, Valts Blukis, Eyvind Niklasson, Max Shatkhin, and Yoav Artzi. 2018. Mapping instructions to actions in 3D environments with visual goal prediction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2667–2678.

Dipendra Misra, John Langford, and Yoav Artzi. 2017. Mapping instructions and visual observations to actions with reinforcement learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1004–1015.

Christopher Potts. 2012. Goal-driven answers in the Cards dialogue corpus. In *Proceedings of the West Coast Conference on Formal Linguistics*, pages 1–20.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*.

Candace L. Sidner, Carolyn Boettner, and Charles Rich. 2000. Lessons learned in building spoken language collaborative interface agents. In *ANLP-NAACL Workshop: Conversational Systems*.

Alane Suhr and Yoav Artzi. 2018. Situated mapping of sequential instructions to actions with single-step reward observation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 2072–2082.

Alane Suhr, Srinivasan Iyer, and Yoav Artzi. 2018. Learning to map context-dependent sentences to executable formal queries. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2238–2249.

Stephanie Tellex, Thomas Kollar, Steven Dickerson, Matthew R. Walter, Ashis Gopal Banerjee, Seth Teller, and Nicholas Roy. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *Proceedings of the National Conference on Artificial Intelligence*.

Jesse Thomason, Shiqi Zhang, Raymond Mooney, and Peter Stone. 2015. Learning to interpret natural language commands through human-robot dialog. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Takuma Udagawa and Akiko Aizawa. 2019. A natural language corpus of common grounding under continuous and partially-observable context. In *Proceedings of the Conference on Artificial Intelligence*.

Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. 2017. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4105–4113.

Andreas Vlachos and Stephen Clark. 2014. A new corpus and imitation learning framework for context-dependent semantic parsing. *Transactions of the Association for Computational Linguistics*, 2:547–560.

Harm de Vries, Kurt Shuster, Dhruv Batra, Devi Parikh, Jason Weston, and Douwe Kiela. 2018. Talk the Walk: Navigating New York City through grounded dialogue. *arXiv preprint arXiv:1807.03367*.

Sida I. Wang, Percy Liang, and Christopher D. Manning. 2016. Learning language games through interaction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 2368–2378.

Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. 2013. The dialog state tracking challenge. In *Proceedings of the SIGDIAL Conference*, pages 404–413.

Ronald J. Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1:270–280.

# Fusion of Detected Objects in Text for Visual Question Answering

**Chris Alberti** **Jeffrey Ling*** **Michael Collins** **David Reitter**
Google Research
{chrisalberti, jeffreyling, mjcollins, reitter}@google.com

## Abstract

To advance models of multimodal context, we introduce a simple yet powerful neural architecture for data that combines vision and natural language. The "Bounding Boxes in Text Transformer" (B2T2) also leverages referential information binding words to portions of the image in a single unified architecture. B2T2 is highly effective on the Visual Commonsense Reasoning benchmark[1], achieving a new state-of-the-art with a 25% relative reduction in error rate compared to published baselines and obtaining the best performance to date on the public leaderboard (as of May 22, 2019). A detailed ablation analysis shows that the early integration of the visual features into the text analysis is key to the effectiveness of the new architecture. A reference implementation of our models is provided[2].

## 1 Introduction

It has long been understood that the meaning of a word is systematically and predictably linked to the context in which it occurs (e.g., Firth 1957; Harris 1954; Deerwester et al. 1990; Mikolov et al. 2013). Different notions of context have resulted in different levels of success with downstream NLP tasks. Recent neural architectures including Transformer (Vaswani et al., 2017) and BERT (Devlin et al., 2018) have dramatically increased our ability to include a broad window of potential lexical hints. However, the same capacity allows for multimodal context, which may help model the meaning of words in general, and also sharpen its understanding of instances of words in context (e.g., Bruni et al. 2014).

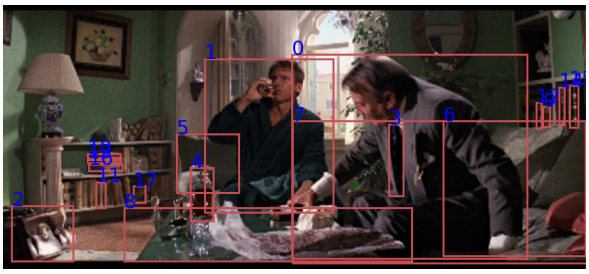

Q: What was [1] doing before he sat in his living room?
$A_1$: He was reading [10].
$A_2$: He was taking a shower. ✓
$A_3$: [0] was sleeping until the noise [1] was making woke him up.
$A_4$: He was sleeping in his bedroom.
$R_1$: His clothes are disheveled and his face is glistening like he's sweaty.
$R_2$: [0] does not look wet yet, but [0] looks like his hair is wet, and bathrobes are what you wear before or after a shower.
$R_3$: He is still wearing his bathrobe. ✓
$R_4$: His hair appears wet and there is clothing hanging in front of him on a line as if to dry.

Figure 1: An example from the VCR dataset. The tasks consists in picking an answer $A_{1-4}$, and then picking a rationale $R_{1-4}$. The data contains explicit pointers in the text to bounding boxes in the image.

In this paper, we consider visual context in addition to language and show that the right integration of visual and linguistic information can yield improvements in visual question answering. The challenge we consider is to answer natural-questions related to a given image. The more general question we address in the context of this problem is how to encode visual and verbal information in a neural architecture. How to best do that is still unclear. How are text entities bound to objects seen in images? Are text and image best integrated late, allowing for independent analysis (*late fusion*), or should the processing of one be conditioned on the analysis of the other (*early fusion*)? How is cross-modal co-reference best encoded at all? Does it make sense to ground words in the visual world before encoding sentence semantics?

*Work done as part of the Google AI residency.
[1] https://visualcommonsense.com
[2] https://github.com/google-research/language/tree/master/language/question_answering/b2t2



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2131–2140, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

In this work we gather evidence to answer these questions by designing the Bounding Boxes in Text Transformer, B2T2 for short, a neural architecture for multimodal encoding of natural language and images, and we evaluate B2T2 on the Visual Commonsense Reasoning benchmark (VCR, Zellers et al. 2019).

Figure 1 shows an illustrative example from the VCR benchmark. VCR is well suited to test rich multimodal representations because it requires the analysis of images depicting people engaged in complex activities; it presents questions, answers and rationales created by human annotators rather than automatic generation; it has a clean multiple-choice interface for evaluation; and yet it is still challenging thanks to a careful selection of answer choices through adversarial matching. VCR has much longer questions and answers compared to other popular Visual Question Answering (VQA) datasets, such as VQA v1 (Antol et al., 2015), VQA v2 (Goyal et al., 2017) and GQA (Hudson and Manning, 2019), requiring more modeling capacity for language understanding.

In our experiments, we found that early fusion of co-references between textual tokens and visual features of objects was the most critical factor in obtaining improvements on VCR. We found that the more visual object features we included in the model's input, the better the model performed, even if they were not explicitly co-referent to the text, and that positional features of objects in the image were also helpful. We finally discovered that our models for VCR could be trained much more reliably when they were initialized from pre-training on Conceptual Captions (Sharma et al., 2018), a public dataset of about 3M images with captions. From the combination of these modeling improvements, we obtained a new model for visual question answering that achieves state-of-the-art on VCR, reducing error rates by more than 25% relative to the best published and documented model (Zellers et al., 2019).

## 2 Problem Formulation

In this work, we assume data comprised of 4-tuples $(I, B, T, l)$ where

1. $I$ is an image,

2. $B = [b_1, \ldots, b_m]$ is a list of bounding boxes referring to regions of $I$, where each $b_i$ is identified by the lower left corner, height and width,

3. $T = [t_1, \ldots, t_n]$ is a passage of tokenized text, with the peculiarity that some of the tokens are not natural language, but explicit references to elements of $B$, and

4. $l$ is a binary label in $\{0, 1\}$.

| Symbol | Type | Description |
|---|---|---|
| $m$ | $\mathbb{N}$ | number of extracted bounding boxes |
| $n$ | $\mathbb{N}$ | number of tokens input to BERT |
| $k$ | $\mathbb{N}$ | number of positional embeddings for image coordinates, usually 56 |
| $d$ | $\mathbb{N}$ | visual features dimension, usually 2048 |
| $h$ | $\mathbb{N}$ | hidden dimension of BERT, usually 1024 |
| $l$ | $\{0, 1\}$ | a binary label |
| $I$ | $\mathbb{R}^{\cdot \times \cdot \times 3}$ | an image |
| $B$ | $\mathbb{R}^{m \times 4}$ | rectangular bounding boxes on $I$, as coordinates of opposite corners |
| $R$ | $\{0, 1\}^{m \times n}$ | matrix encoding which bounding boxes in $B$ correspond to which tokens in $T$ |
| $T$ | $\mathbb{N}^{n \times 2}$ | input tokens, each expressed as word piece id and token type |
| $\Phi$ | $\mathbb{R}^{\cdot \times \cdot \times 3} \to \mathbb{R}^d$ | a function to extract visual feature vectors from an image |
| $\pi$ | $\mathbb{R}^4 \to \mathbb{R}^d$ | a function to embed the position and shape of a bounding box |
| $\Psi$ | $\mathbb{R}^{n \times h} \to \mathbb{R}^h$ | a function to compute a passage embedding from per-token embeddings |
| $E$ | $\mathbb{N}^{n \times 2} \to \mathbb{R}^{n \times h}$ | non-contextualized token embeddings, encoding word piece ids, token types and positions |

Table 1: Glossary of mathematical symbols used in this work.

While it might seem surprising to mix natural text with explicit references to bounding boxes, this is actually a quite natural way for people to discuss objects in images and the VCR dataset is annotated in exactly this way.

We assume an image representation function $\Phi$ that converts an image, perhaps after resizing and padding, to a fixed size vector representation of dimension $d$.

We similarly assume a pretrained textual representation capable of converting any tokenized passage of text, perhaps after truncating or padding, into a vector representation of dimension $h$. We assume a context independent token representation $E$ in the shape of a vector of dimension $h$ for each token and a passage level representation $\Psi$ which operates on $E(T)$ and returns a passage level vector representation of dimension $h$.

We refer the reader to Table 1 for an overview of the notation used in this work. Full details on how the VCR dataset is encoded into this formalism are given in Section 4.

## 3 Models and Methods

We evaluate two main architectures: "Dual Encoder", a late fusion architecture where image and text are encoded separately and answer scores are computed as an inner product, and the full B2T2 model, an early fusion architecture where visual features are embedded on the same level as input word tokens. Section 5.2 will summarize experiments with model variants to answer the research questions laid out in the introduction and to analyze what works, and why.

### 3.1 Dual Encoder

Dual Encoders, discussed for example by Wu et al. (2018) and Gillick et al. (2018), are models that embed objects of potentially different types into a common representation space where a similarity function can be expressed e.g. as a dot product or a cosine similarity. A notable example of a dual encoder for image classification is WSABIE, proposed by Weston et al. (2011).

Our Dual Encoder architecture is shown in Figure 2. We model the class distribution as

$$p(l = 1|I, T) = \frac{1}{1 + e^{-\Psi(E(T))^\top D\Phi(I)}}$$

where $D$ is a learned matrix of size $d \times h$. In this model, co-reference information is completely ignored, and the model must rely on fixed dimensional vectors for the late fusion of textual and visual contexts. However, we found this to be surprisingly competitive on VCR compared to published baselines, perhaps due to our choice of powerful pretrained models.

### 3.2 B2T2

Our B2T2 architecture is shown in Figure 3. We model the class distribution as

$$p(l|I, B, R, T) = \frac{e^{\Psi(E'(I,B,R,T))\cdot a_l + b_l}}{\sum_{l'} e^{\Psi(E'(I,B,R,T))\cdot a_{l'} + b_{l'}}}$$

where $a_l \in \mathbb{R}^h$ and $b_l \in \mathbb{R}$ for $l \in \{0, 1\}$ are learned parameters. $E'(I, B, R, T)$ is a non-contextualized representation for each token and of its position in text, but also of the content and position of the bounding boxes. The key difference from "Dual Encoder" is that text, image and bounding boxes are combined at the level of the non-contextualized token representations rather than right before the classification decision.

The computation of $E'(I, B, R, T)$ is depicted in Figure 4. More formally, for a given example, let matrix $R \in \{0, 1\}^{m\times n}$ encode the references between the bounding boxes in $B$ and the tokens in $T$, so that $R_{ij}$ is 1 if and only if bounding box $i$ is referenced by token $j$. Then

$$E'(I, B, R, T) = E(T) + \sum_{i=1}^{m} R_i \left[M(\Phi(\text{crop}(I, b_i)) + \pi(b_i))\right]^\top$$

where $M$ is a learned $h \times d$ matrix, $\Phi(\text{crop}(I, b_i))$ denotes cropping image $I$ to bounding box $b_i$ and then extracting a visual feature vector of size $d$, and $\pi(b_i)$ denotes the embedding of $b_i$'s shape and position information in a vector of size $d$.

To embed the position and size of a bounding box $b$, we introduce two new learnable embedding matrices $X$ and $Y$ of dimension $k \times \frac{d}{4}$. Let the coordinates of the opposite corners of $b$ be $(x_1, y_1)$ and $(x_2, y_2)$, after normalizing so that a bounding box covering the entire image would have $x_1 = y_1 = 0$ and $x_2 = y_2 = k$. Position embeddings are thus defined to be

$$\pi(b) = \text{concat}(X_{\lfloor x_1 \rfloor}, Y_{\lfloor y_1 \rfloor}, X_{\lfloor x_2 \rfloor}, Y_{\lfloor y_2 \rfloor})$$

### 3.3 Loss

All of our models are trained with binary cross entropy loss using label $l$. Denoting $p := p(l = 1|I, B, R, T)$, we have for each example

$$\mathcal{L}_{\text{BCE}} = l \log p + (1 - l) \log(1 - p)$$

### 3.4 Pretraining on Conceptual Captions

Before training on VCR, we pretrain B2T2 on image and caption pairs using a Mask-LM pretraining technique like the one used in BERT (Devlin et al., 2018). The setup used during pretraining is shown in Figure 5, where the model uses the image as additional context when filling in the mask.

We use two tasks for pretraining: (1) impostor identification and (2) masked language model prediction. For the impostor task, we sample a random negative caption for each image and ask the model to predict whether the caption is correctly associated. For mask-LM, we randomly replace tokens in the caption with the `[MASK]` token, and the model must predict the original token (see Devlin et al. (2018) for more details).

Formally, the pretraining data consist of images $I$ and captions $T$. We do not consider bounding

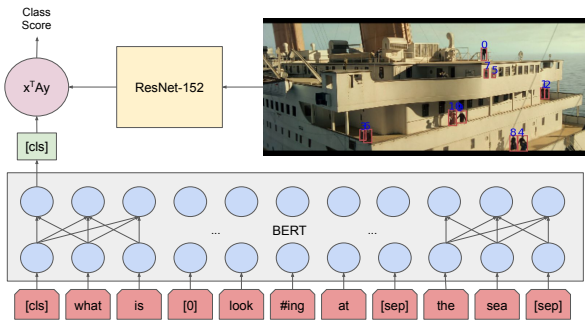


Figure 2: Dual Encoder architecture with late fusion. The model extracts a single visual feature vector from the entire image. Bounding boxes are ignored.

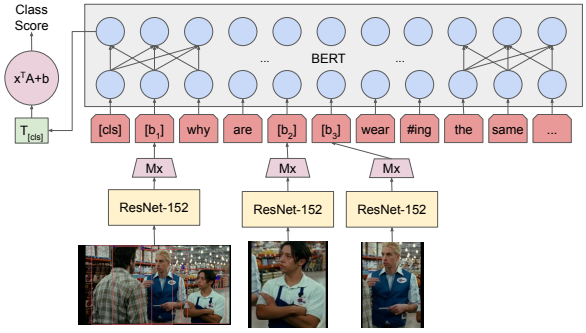


Figure 3: B2T2 architecture with early fusion. Bounding boxes are inserted where they are mentioned in the text and at the end of the input, as described in Sec. 4.

boxes during pretraining, so $B = \emptyset$. The binary label $l$ indicates whether the caption is an impostor or not. The loss for impostor identification is binary cross entropy $\mathcal{L}_{\text{BCE}}$ with label $l$ as in 3.3. We denote the loss for mask-LM as $\mathcal{L}_{\text{MLM}}$, which is the summed cross entropy of the predicted token distributions against the true tokens.

To ensure that our model correctly grounds the language to the image with the mask LM loss, we only use it for positive captions, zeroing it out for negative captions. Our final objective is the sum of the losses:

$$\mathcal{L} = \mathcal{L}_{\text{BCE}} + I[l = 1] \cdot \mathcal{L}_{\text{MLM}}$$

where $I[l = 1]$ is an indicator for the label $l$ being positive for the image and caption pair.

We pretrain on Conceptual Captions (Sharma et al., 2018), a dataset with over 3M images paired with captions.[3] We found empirically that pretraining improves our model slightly on VCR, but more importantly, allows our model to train stably. Without pretraining, results on VCR exhibit much higher variance. We refer the reader to Section 5.2 for an ablation analysis on the effect of pretraining.

### 3.5 Implementation Details

We use ResNet-152[4] (He et al., 2016) pretrained on ImageNet for $\Phi$, which yields a vector representation of size $d = 2048$. BERT-Large (Devlin et al., 2018) provides both $E$ and $\Psi$. The latter is a pretrained Transformer with 24 layers, 16 attention heads, and hidden size 1024. For BERT,

[3] We also tried pretraining on MS-COCO images and captions (Lin et al., 2014), but found this to be ineffective. This could be because MS-COCO is smaller (with around 80k images, 400k captions).

[4] Publicly available at `tfhub.dev`

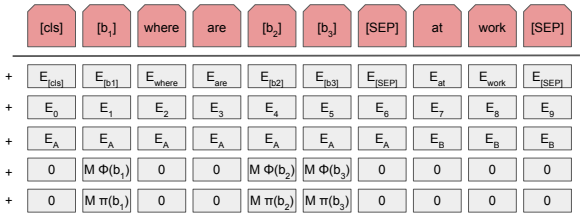


Figure 4: How input embeddings are computed in our B2T2 architecture.

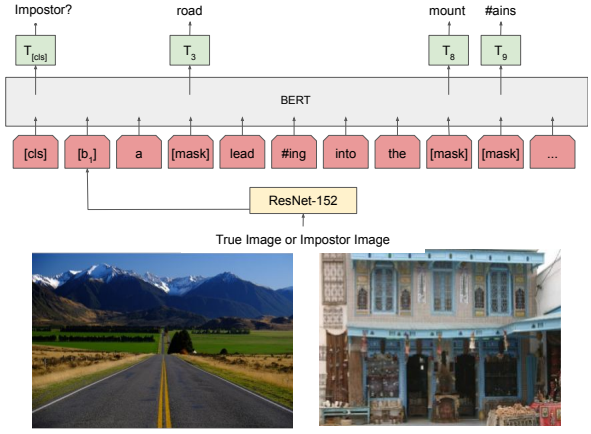


Figure 5: Mask-LM pretraining for B2T2.

$E$ corresponds to its token embeddings, $\Psi$ to the `[CLS]` token representation in the final layer, and so $\Psi(E(T))$ corresponds to the BERT passage representation of size $h = 1024$.

We found empirically that it was slightly better to keep $\Phi$ fixed rather than fine-tuning it, but that it was of critical importance to fine-tune $\Psi$ and $E$ for the new task.

In all of our finetuning experiments we use the Adam optimizer (Kingma and Ba, 2014) and trained our models with a grid of hyperparameters: a learning rate of $2 \cdot 10^{-5}$ and $3 \cdot 10^{-5}$, for 3,

4, and 5 epochs with a linear learning rate decay, and two random seed for initialization.

To maximize performance on VCR, we also evaluate an ensemble of B2T2 models. Our ensemble is comprised of 5 identical B2T2 models, trained for 3 epochs with an initial learning rate of $2 \cdot 10^{-5}$, but initialized with 5 different random seeds. The resulting class logits are then summed to obtain the ensemble scores.

## 4 Data

*Visual Commonsense Reasoning* (VCR, visualcommonsense.com, Zellers et al. 2019) is a corpus that contains a sample of stills from movies. Questions and answers revolve around conclusions or assumptions that require knowledge external to the images. The associated task is to not only select a correct answer but also provide reasoning in line with common sense. Matching our problem formulation given before, a VCR sample is defined as a tuple $(I, O, Q, A, R)$. Here, $I$ is the image, and $O$ is a sequence of objects identified in the image. A question $Q = [q_0, \ldots, q_k]$ is given, where tokens are either textual words or deictic references to objects in $O$. Each question contains a set of four answers $A = \{A_1, A_2, A_3, A_4\}$, with exactly one correct answer $A^*$. Each response follows the schema of the queries. Finally, there is a set of four rationales $R = \{R_1, R_2, R_3, R_4\}$, with exactly one rationale $R^*$ identified as correct in supporting $A^*$.

Each of the objects in $O = [(b_1, l_1), \ldots, (b_{|O|}, l_{|O|})]$ is identified in the image $I$ by bounding boxes $b_i$. The objects are also labeled with their classes with a text token $l_i$.

The $Q \rightarrow A$ task is to choose $A^*$ given $(I, O, Q, A)$. The $QA \rightarrow R$ task is to choose $R^*$ given $(I, O, Q, A^*, R)$. Finally, the $Q \rightarrow AR$ task is a pipeline of the two, where a model must first correctly choose $A^*$ from $A$, then correctly choose $R^*$ given $A^*$.

We adapt VCR to our problem formulation by converting each VCR example to four instances for the $Q \rightarrow A$ task, one per answer in $A$, and four instances for the $QA \rightarrow R$ task, one per rationale in $R$. We construct the text for the instances in the $Q \rightarrow A$ task as

$$[\text{[CLS]}, [b_0], q_0, \ldots, \text{[SEP]}, a_0, \ldots, \text{[SEP]}, l_1, [b_1], \ldots, l_p, [b_p]]$$

and in the $QA \rightarrow R$ task as

$$[\text{[CLS]}, [b_0], q_0, \ldots, \text{[SEP]}, a_0^*, \ldots, r_0, \ldots, \text{[SEP]}, l_1, [b_1], \ldots, l_p, [b_p]].$$

where [CLS], [SEP] are special tokens for BERT.

Here, $[b_0]$ is a bounding box referring to the entire input image. $q_0, \ldots$ are all question tokens, $a_0, \ldots$ answer tokens, $a_0^*, \ldots$ answer tokens for the correct answer, and $r_0, \ldots$ rationale tokens. We append the first $p$ bounding boxes in $O$ with class labels to the end of the sequence (in our experiments, we use $p = 8$), and for objects referenced in $Q, A, R$, we prepend the class label token (i.e. $[b_i]$ becomes $l_i, [b_i]$). We assign the binary label $l$ to every instance to represent whether the answer or rationale choice is the correct one.

## 5 Experimental Results

### 5.1 VCR Task Performance

Our final results on the VCR task are shown in Table 2. Our Dual Encoder model worked surprisingly well compared to Zellers et al. (2019), surpassing the baseline without making use of bounding boxes. We also evaluate a Text-Only baseline, which is similar to the Dual Encoder model but ignores the image. The ensemble of B2T2 models, pretrained on Conceptual Captions, obtained absolute accuracy improvements of $8.9\%$, $9.8\%$ and $13.1\%$ compared to the published R2C baseline for the $Q \rightarrow A$, $QA \rightarrow R$, and $Q \rightarrow AR$ tasks respectively. At the time of this writing (May 22, 2019), both our single B2T2 and ensemble B2T2 models outperform all other systems in the VCR leaderboard.

### 5.2 Ablations

To better understand the reason for our improvements, we performed a number of ablation studies on our results, summarized in Table 3. We consider ablations in order of decreasing impact on the VCR dev set $Q \rightarrow A$ accuracy.

**Use of Bounding Boxes.** The bounding boxes considered by our model turns out to be the most important factor in improving the accuracy of our model. Without any bounding boxes we obtain $67.5\%$ accuracy, just above the accuracy of the dual encoder. With 4 instead of 8 appended bounding boxes we obtain $71\%$ accuracy. With 8 bounding boxes, but no textual labels from the bounding boxes in the text we obtain $70.9\%$ accuracy,

| Model | $Q \rightarrow A$ Val | $Q \rightarrow A$ Test | $QA \rightarrow R$ Val | $QA \rightarrow R$ Test | $Q \rightarrow AR$ Val | $Q \rightarrow AR$ Test |
|---|---|---|---|---|---|---|
| Chance | 25.0 | 25.0 | 25.0 | 25.0 | 6.2 | 6.2 |
| Text-Only BERT (Zellers et al.) | 53.8 | 53.9 | 64.1 | 64.5 | 34.8 | 35.0 |
| R2C (Zellers et al.) | 63.8 | 65.1 | 67.2 | 67.3 | 43.1 | 44.0 |
| HCL HGP (unpub.) | - | 70.1 | - | 70.8 | - | 49.8 |
| TNet (unpub.) | - | 70.9 | - | 70.6 | - | 50.4 |
| B-VCR (unpub.) | - | 70.5 | - | 71.5 | - | 50.8 |
| TNet 5-Ensemble (unpub.) | - | 72.7 | - | 72.6 | - | 53.0 |
| Text-Only BERT (ours) | 59.5 | - | 65.6 | - | 39.3 | - |
| Dual Encoder (ours) | 66.8 | - | 67.7 | - | 45.3 | - |
| B2T2 (ours) | 71.9 | 72.6 | 76.0 | 75.7 | 54.9 | 55.0 |
| B2T2 5-Ensemble (ours) | **73.2** | **74.0** | **77.1** | **77.1** | **56.6** | **57.1** |
| Human | | 91.0 | | 93.0 | | 85.0 |

Table 2: Experimental results on VCR, incorporating those reported by Zellers et al. (2019). The proposed B2T2 model and the B2T2 ensemble outperform published and unpublished/undocumented results found on the VCR leaderboard at `visualcommonsense.com/leaderboard` as of May 22, 2019.

| | $Q \rightarrow A$ |
|---|---|
| Dual Encoder | 66.8 |
| No bboxes | 67.5 |
| Late fusion | 68.6 |
| BERT-Base | 69.0 |
| ResNet-50 | 70.4 |
| No bbox class labels | 70.9 |
| Fewer appended bboxes ($p = 4$) | 71.0 |
| No bbox position embeddings | 71.6 |
| Full B2T2 | 71.9 |

Table 3: Ablations for B2T2 on VCR dev. The Dual Encoder and the full B2T2 models are the main models discussed in this work. All other models represent ablations from the full B2T2 model.

showing that our model can make use of labels for detected objects. Example 1 in Table 4 shows an example that our models can only get right if bounding box 5 is available.

**Late Fusion vs. Early Fusion.** The second most important architectural choice in our model is to combine visual information at the level of context independent token embeddings, rather than at the highest levels of the neural representation. If in the the full B2T2 model we add visual embeddings in the last layer of BERT rather than in the first, we lose 3.3% accuracy.

**Effect of Textual Model Size.** The original VCR work by Zellers et al. (2019) made use of BERT-base, while we use BERT-large to initialize our models. To test how much of our improvements are simply due to our model being larger, we retrained B2T2 models using BERT-base and found that we lose 2.9% accuracy.

**Effect of Visual Model Size.** How important is the choice of the visual model in the performance

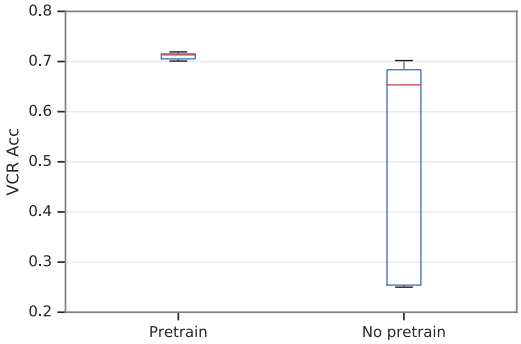


Figure 6: Boxplot of dev $Q \rightarrow A$ accuracy on VCR with and without pretraining. Pretraining on Conceptual Captions lowers variance when fine-tuning on VCR, from a grid search on multiple random seeds, learning rates, and VCR training epochs.

of B2T2? As further discussed in the error analysis section of this work, we suspect that B2T2 could be significantly improved by extending the visual features to represent more than just objects, but also activities, expressions and more. However it appears that even the size of the object detection model is important. If we swap out ResNet-152 for ResNet-50, accuracy decreases by 1.5%.

**Pretraining.** We found that performance improvements from pretraining are quite small, around 0.4% accuracy, but initializing from a pretrained model heavily reduces variance of results. We show this effect in Figure 6 over the grid of learning rates, random seeds, and training epochs described in Section 3.5.

**Position of Bounding Boxes** We additionally investigated the effect of removing position information from the model. The benefit of having bounding box positional embeddings is the smallest of the ones we considered. A model trained without positional embeddings only loses 0.3% accuracy compared to the full model.

### 5.3 Error Analysis

We picked some examples, shown in Table 4, to illustrate the kinds of correct and incorrect choices that B2T2 is making, compared to our dual encoder and to a text only model.

In Example 1 we show an example of how our model picks the right answer only when it is able to make use of all provided bounding boxes. Bounding box 5 in particular contains the clue that allows the observer to know that the man in the picture might have just gone shopping.

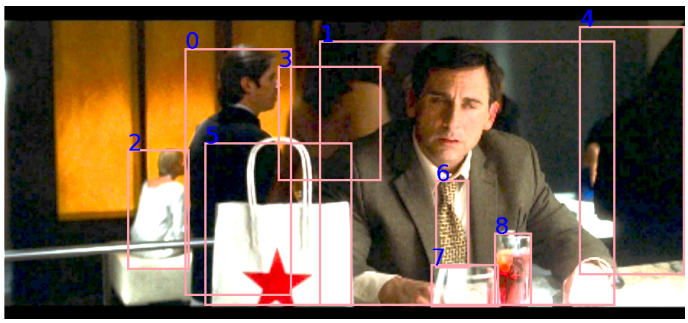

Example 1

Q: What did [1] do before coming to this location?

$A_1$: He took horse riding lessons. (text-only)
**$A_2$: He was just shopping. (B2T2)**
$A_3$: He found a skeleton.
$A_4$: He came to buy medicine. (dual encoder)

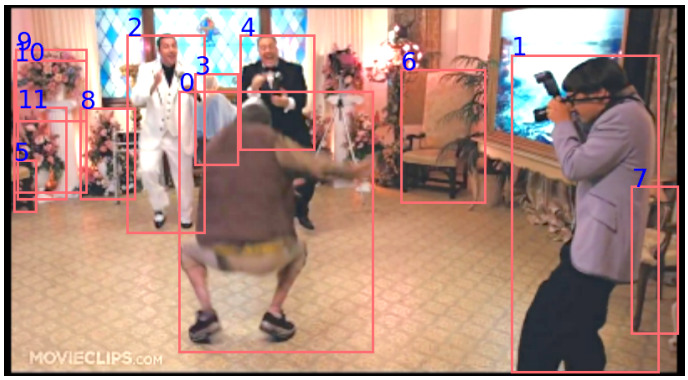

Example 2

Q: How are [2, 4] related?

$A_1$: [2, 4] are partners on the same mission.
**$A_2$: [2, 4] are a recently married gay couple. (B2T2)**
$A_3$: They are likely acquaintances.
$A_4$: They are siblings. (text-only, dual encoder)

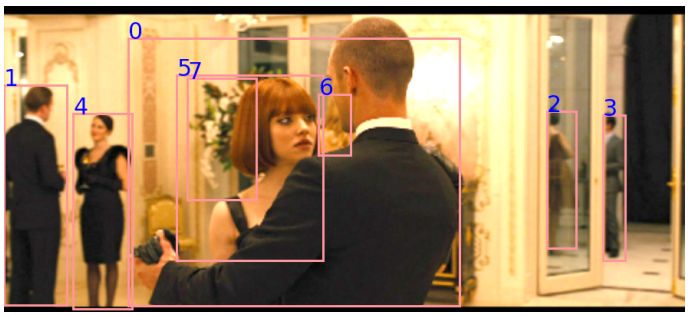

Example 3

Q: What are [0] and the woman doing?

$A_1$: Their husbands are doing something dumb.
$A_2$: They are observing the results of an experiment. (text-only, dual encoder)
**$A_3$: They are dancing. (B2T2)**
$A_4$: They are acting as nurses for the rescued people.



Example 4

Q: How is [2] feeling?

$A_1$: [2] is feeling shocked. (B2T2, dual encoder)
$A_2$: [0] is feeling anxious.
$A_3$: [2] is not feeling well.
**$A_4$: [2] is feeling joy and amusement. (text-only)**

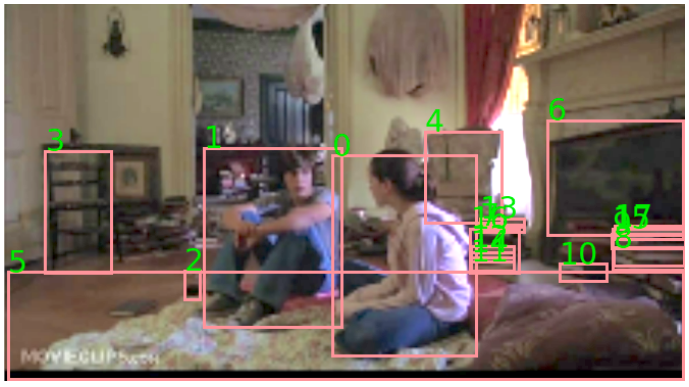

Example 5

Q: Why is [1] on the floor talking to [0]?

$A_1$: The man on the floor was assaulting [1].
$A_2$: He is asking her to help him stand up. (B2T2, dual encoder)
$A_3$: [1] just dropped all his books on the floor.
**$A_4$: [1] looks like he is telling [0] a secret. (text-only)**

Table 4: Examples of the $Q \rightarrow A$ task from the VCR dev set. The correct answer for every example is marked in bold. The answers picked by the text-only model, by the dual encoder and by B2T2 are indicated in parenthesis.

In Examples 2 and 3, no specific bounding box appears to contain critical clues for answering the question, but B2T2 outperforms models without access to the image or without access to bounding boxes. It is possible that B2T2 might be gaining deeper understanding of a scene by combining information from important regions of the image.

In Examples 4 and 5, we see failure cases of both the dual encoder and B2T2 compared to the text only-model. Both these examples appear to point to a limitation in the amount of information that the we are able to extract from the image. Indeed our vision model is trained on ImageNet, and so it might be very good at recognizing objects, but might be unable to recognize human expressions and activities. Our models could have correctly answered the question in Example 4 if they were able to recognize smiles. Similarly our models could have ruled out the incorrect answer they picked for the question in Example 5 if they were able to see that both people in the picture are sitting down and are not moving.

## 6 Related Work

Modeling visual contexts can aid in learning useful sentence representations (Kiela et al., 2017) and even in training language models (Ororbia et al., 2019). This paper takes these more general ideas to a downstream task that requires modeling of visual input. Similar to B2T2, VideoBERT (Sun et al., 2019) jointly processes video frames and text tokens with a Transformer architecture. However, VideoBERT cannot answer questions, nor does the model consider bounding boxes.

Our B2T2 model is similar to the Bottom-Up Top-Down attention model (Anderson et al., 2018) in how bounding boxes generated at preprocessing time are attended to by the VQA model. "Bottom-Up" refers to the idea of attending from the text to the bounding boxes of objects detected in the image, while "Top-Down" refers to the idea of attending to regions constructed as a regular grid over the image. The Bottom-Up Top-Down model however reduces the text to a fixed length vector representation before attending to image regions, while B2T2 instead treats image regions as special visual tokens mixed in the text. In this sense, Bottom-Up Top-Down model is a late fusion model, while B2T2 is early fusion.

The Neuro-Symbolic Concept Learner (Mao et al., 2019) also uses bounding boxes to learn visually grounded concepts through language. The Neuro-Symbolic Concept Learner however relies on a semantic parser to intepret language, while B2T2 uses a Transformer to construct a joint representation of textual tokens and visual tokens.

Another recently proposed model for VQA is MAC (Hudson and Manning, 2018). As presented, MAC does not make use of bounding boxes, which makes it a Top-Down model in the nomenclature of Anderson et al. (2018). MAC also reduces the textual information to a vector of fixed length. However MAC makes use of a new neural architecture designed to perform an explicit multi-step reasoning process and is reported to perform better than Anderson et al. (2018) on the GQA dataset (Hudson and Manning, 2019).

After the submission of this paper, several new works were published with excellent results on VCR, in some cases exceeding the performance of our system. In particular we mention ViLBERT (Lu et al., 2019), VL-BERT (Su et al., 2019), Unicoder-VL (Li et al., 2019a), and VisualBERT (Li et al., 2019b).

VCR is only one of several recent datasets pertaining to the visual question answering task. VQA (Antol et al., 2015; Zhang et al., 2016; Goyal et al., 2017) contains photos and abstract scenes with questions and several ground-truth answers for each, but the questions are less complex than VCR's. CLEVR (Johnson et al., 2017) is a visual QA task with compositional language, but the scenes and language are synthetic. GQA (Hudson and Manning, 2019) uses real scenes from Visual Genome, but the language is artificially generated. Because VCR has more complex natural language than other datasets, we consider it the best evaluation of a model like B2T2, which has a powerful language understanding component.

## 7 Conclusion

In this work we contrast different ways of combining text and images when powerful text and vision models are available. We picked BERT-Large (Devlin et al., 2018) as our text model, ResNet-152 (He et al., 2016) as our vision model, and the VCR dataset (Zellers et al., 2019) as our main benchmark.

The early-fusion B2T2 model, which encodes sentences along with links to bounding boxes around identified objects in the images, produces the best available results in the visual question an-

swering tasks. A control model, implementing late fusion (but the same otherwise), performs substantively worse. Thus, grounding words in the visual context should be done early rather than late.

We also demonstrate competitive results with a Dual Encoder model, matching state-of-the-art on the VCR dataset even when textual references to image bounding boxes are ignored. We then showed that our Dual Encoder model can be substantially improved by deeply incorporating in the textual embeddings visual features extracted from the entire image and from bounding boxes. We finally show that pretraining our deep model on Conceptual Captions with a Mask-LM loss yields a small additional improvement as well as much more stable fine-tuning results.

## References


Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6077–6086.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. VQA: Visual Question Answering. In *International Conference on Computer Vision (ICCV)*.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *Journal of Artificial Intelligence Research*, 49:1–47.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

John R Firth. 1957. A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*.

Daniel Gillick, Alessandro Presta, and Gaurav Singh Tomar. 2018. End-to-end retrieval in continuous space. *arXiv preprint arXiv:1811.08008*.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer.

Drew A Hudson and Christopher D Manning. 2018. Compositional attention networks for machine reasoning. In *ICLR*.

Drew A Hudson and Christopher D Manning. 2019. Gqa: a new dataset for compositional question answering over real-world images. *arXiv preprint arXiv:1902.09506*.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2901–2910.

Douwe Kiela, Alexis Conneau, Allan Jabri, and Maximilian Nickel. 2017. Learning visually grounded sentence representations. *arXiv preprint arXiv:1707.06320*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Gen Li, Nan Duan, Yuejian Fang, Daxin Jiang, and Ming Zhou. 2019a. Unicoder-vl: A universal encoder for vision and language by cross-modal pre-training. *arXiv preprint arXiv:1908.06066*.

Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. 2019b. Visualbert: A simple and performant baseline for vision and language. *arXiv preprint arXiv:1908.03557*.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *arXiv preprint arXiv:1908.02265*.

Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B Tenenbaum, and Jiajun Wu. 2019. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. In *ICLR*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Alexander G. Ororbia, Ankur Mali, Matthew A. Kelly, and David Reitter. 2019. Like a baby: Visually situated neural language acquisition. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. 2018. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 2556–2565.

Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. 2019. Vl-bert: Pre-training of generic visual-linguistic representations. *arXiv preprint arXiv:1908.08530*.

Chen Sun, Austin Myers, Carl Vondrick, Kevin Murphy, and Cordelia Schmid. 2019. Videobert: A joint model for video and language representation learning. *arXiv preprint arXiv:1904.01766*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Jason Weston, Samy Bengio, and Nicolas Usunier. 2011. Wsabie: Scaling up to large vocabulary image annotation. In *Twenty-Second International Joint Conference on Artificial Intelligence*.

Ledell Yu Wu, Adam Fisch, Sumit Chopra, Keith Adams, Antoine Bordes, and Jason Weston. 2018. Starspace: Embed all the things! In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Rowan Zellers, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. From recognition to cognition: Visual commonsense reasoning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Peng Zhang, Yash Goyal, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2016. Yin and Yang: Balancing and answering binary visual questions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

# TIGEr: Text-to-Image Grounding for Image Caption Evaluation

**Ming Jiang**[1], **Qiuyuan Huang**[3], **Lei Zhang**[3], **Xin Wang**[2]
**Pengchuan Zhang**[3], **Zhe Gan**[3], **Jana Diesner**[1], **Jianfeng Gao**[3]
[1]University of Illinois at Urbana-Champaign, [2]University of California, Santa Barbara
[3]Microsoft Research, Redmond
{mjiang17,jdiesner}@illinois.edu, xwang@cs.ucsb.edu
{qihua,leizhang,penzhan,zhe.gan,jfgao}@microsoft.com

## Abstract

This paper presents a new metric called TIGEr for the automatic evaluation of image captioning systems. Popular metrics, such as BLEU and CIDEr, are based solely on text matching between reference captions and machine-generated captions, potentially leading to biased evaluations because references may not fully cover the image content and natural language is inherently ambiguous. Building upon a machine-learned text-image grounding model, TIGEr allows to evaluate caption quality not only based on how well a caption represents image content, but also on how well machine-generated captions match human-generated captions. Our empirical tests show that TIGEr has a higher consistency with human judgments than alternative existing metrics. We also comprehensively assess the metric's effectiveness in caption evaluation by measuring the correlation between human judgments and metric scores.

## 1 Introduction

Image captioning is a research topic at the nexus of natural language processing and computer vision, and has a wide range of practical applications (Hodosh et al., 2013; Fang et al., 2015), such as image retrieval and human-machine interaction. Given an image as input, the task of image captioning is to generate a text that describes the image content. Overall, prior studies of this task have been focusing on two aspects: 1) building large benchmark datasets (Lin et al., 2014; Hodosh et al., 2013), and 2) developing effective caption generation algorithms (Karpathy and Li, 2015; Bernardi et al., 2016; Gan et al., 2017). Remarkable contributions that have been made, but assessing the quality of the generated captions is still an insufficiently addressed issue. Since numerous captions with varying quality can be produced by machines, it is important to propose an automatic evaluation metric that is highly consistent with human judges, easy to implement and interpret.

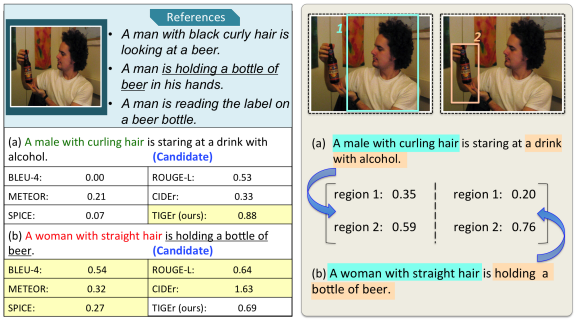


Figure 1: An example of caption evaluation challenge. Given two candidate captions: (a) correctly describes image content, but most words in this caption are not contained in references, while (b) overlaps with references ("is holding a bottle of beer"), but involves wrong information ("a woman with straight hair"). Prior metrics based only on text-level comparisons fail to assess the quality of candidate captions, while our metric (TIGEr) can detect this inconsistency by matching a caption with image content. To explain the TIGEr result, we show the grounding weights of two illustrative image regions with each candidate. The proper text information matched with each region is highlighted by colors.

Prior evaluation metrics typically considered several aspects, including content relevance, information correctness, grammaticality, and if expressions are human-like (Hodosh et al., 2013; Bernardi et al., 2016). Rule-based metrics inspired by linguistic features such as n-gram overlapping are commonly used to evaluate machine-generated captions (Chen et al., 2018; Karpathy and Li, 2015; Huang et al., 2019). However, these metrics primarily evaluate a candidate caption based on references without taking image content into account, and the possible information loss caused by references may bring biases to the evaluation process. Moreover, the ambiguity inherent to natural language presents a challenge for rule-based met-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2141–2152, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

rics that are based on text overlapping. As shown in Figure 1, although Caption (a) describes the image properly, most words in this sentence are different from the references. In contrast, Caption (b) contains wrong information ("a woman with straight hair"), but overlaps with human-written references ("is holding a bottle of beer"). Consequently, all existing metrics improperly assign a higher score to Caption (b).

In order to address the aforementioned challenges, we propose a novel Text-to-Image Grounding based metric for image caption Evaluation (TIGEr), which considers both image content and human-generated references for evaluation. First, TIGEr grounds the content of texts (both reference and generated captions) in a set of image regions by using a pre-trained image-text grounding model. Based on the grounding outputs, TIGEr calculates a score by comparing both the relevance ranking and distribution of grounding weights among image regions between the references and the generated caption. Instead of evaluating image captions by exact n-gram matching, TIGEr compares the candidate caption with references by mapping them into vectors in a common semantic space. As a result, TIGEr is able to detect the paraphrases based on the semantic meaning of caption sentences. A systematic comparison of TIGEr and other commonly used metrics shows that TIGEr improves the evaluation performance across multiple datasets and demonstrates a higher consistency with human judgments. For example, Figure 1 presents a case where only TIGEr is able to assign a higher score to the caption candidate that is more consistent with human judgments. Our main contributions include:

- We propose a novel automatic evaluation metric, TIGEr[1], to assess the quality of image captions by grounding text captions in image content.
- By performing an empirical study, we show that TIGEr outperforms other commonly used metrics, and demonstrates a higher and more stable consistency with human judgments.
- We conduct an in-depth analysis of the assessment of metric effectiveness, which deepens our understanding of the characteristics of different metrics.

[1]Code is released at https://github.com/SeleenaJM/CapEval.

## 2 Related Work

**Caption Evaluation** The goal of image caption evaluation is to measure the quality of a generated caption given an image and human-written reference captions (Bernardi et al., 2016).

In general, prior solutions to this task can be divided into three groups. First, human evaluation is typically conducted by employing human annotators to assess captions (e.g., via Amazons Mechanical Turk) (Bernardi et al., 2016; Aditya et al., 2015a; Wang et al., 2018b).

Second, automatic rule-based evaluation measures assess the similarity between references and generated captions. Many metrics in this group were extended from other related tasks (Papineni et al., 2002; Banerjee and Lavie, 2005; Lin, 2004). BLEU (Papineni et al., 2002) and METEOR (Banerjee and Lavie, 2005) were initially developed to evaluate machine translation outcomes based on n-gram precision and recall. ROUGE (Lin, 2004) was originally used in text summarization, which measures the overlap of n-grams using recall. Recently, two metrics were specifically built for visual captioning: 1) CIDEr (Vedantam et al., 2015) measures n-gram similarity based on TF-IDF; and 2) SPICE (Anderson et al., 2016) quantifies graph similarity based on the scene graphs built from captions. Overall, these metrics focus on text-level comparisons, assuming that the information contained in human-written references can well represent the image content. Differing from prior work, we argue in this study that references may not fully cover the image content because both references and captions are incomplete and selective translations of image contents made by human judges or automated systems. The ground truth can only be fully revealed by taking the images themselves into account for evaluation.

Finally, machine-learned metrics use a trained model to predict the likelihood of a testing caption as a human-generated description (Cui et al., 2018; Dai et al., 2017). Prior studies have applied learning-based evaluation for related text generation tasks, e.g., machine translation (Corston-Oliver et al., 2001; Lowe et al., 2017; Kulesza and Shieber, 2004). Most recently, Cui et al. (2018) trained a hybrid neural network model for

caption evaluation based on image and text features. This work mainly focuses on the generation of adversarial data used for model training. Despite improving the consistency with human decisions, this approach may involve high computational cost and lead to overfitting (Gao et al., 2019). Besides, the interpretability of the evaluation is limited due to the black-box nature of end-to-end learning models. An even more serious problem of using machine-learned metrics is the so-called "gaming of the metric": if a generation system is optimized directly for a learnable metric, then the system's performance is likely to be over-estimated. See a detailed discussion in Gao et al. (2019).

Though caption evaluation is similar to traditional text-to-text generation evaluation in terms of testing targets and evaluation principles, this task additionally need to synthesize image and reference contents as ground truth information for further evaluation, which is hard to achieve by using a rule-based method with a strong human prior. In this study, we propose a new metric that combines the strengths of learning-based and rule-based approaches in terms of: 1) utilizing a pre-trained grounding model[2] to combine the image and text information; and 2) scoring captions by defining principle rules based on the grounding outputs to make the evaluation interpretable.

**Image-Text Matching** The task of image-text matching can be defined as the measurement of semantic similarity between visual data and text data. The main strategy of solutions designed for this task is to map image data and text data into a common semantic vector space (Fang et al., 2015; Wang et al., 2018a; Lee et al., 2018). Prior studies usually consider this issue from two perspectives: From the perspective of data encoding, prior work either regards an image/text as a whole or applies bottom-up attention to encode image regions and words/phrases (Karpathy et al., 2014; Kiros et al., 2014). From the perspective of algorithm development, prior work has often applied a deep learning framework to build a joint embedding space for images and texts, and some of these studies train a model with ranking loss (Fang et al., 2015; Lee et al., 2018; Frome et al., 2013) and some use classification loss (e.g., softmax function) (Jabri et al., 2016; Fukui et al., 2016). In this work, we take advantage of a state-of-the-art model for image-text matching (Lee et al., 2018) and propose an automatic evaluation metric for image captioning based on the matching results. Our goal is to capture comprehensive information from input data while also providing an explainable method to assess the quality of an image description.

[2]We find in our experiments that we can simply use the pre-trained model as is to evaluate systems developed on different datasets, and the pre-trained model is very efficient to run. Thus, the cost of computing TIGEr is not much higher than that of computing other traditional metrics.

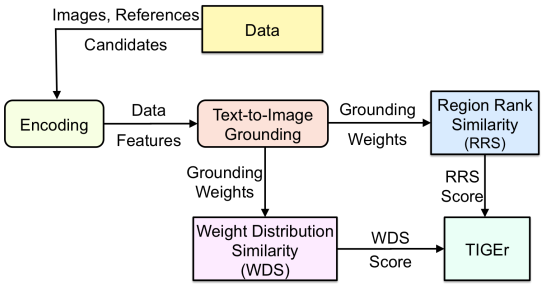


Figure 2: TIGEr framework.

## 3 The TIGEr Metric

The overall framework of TIGEr is shown in Figure 2. With the assumption that a good machine-generated caption $C$ should generate a description of an image $V$ like a human would, we compare $C$ against a set of reference captions produced by humans $\mathcal{R} = \{R_1, ..., R_k\}$ in two stages.

The first stage is *text-image grounding*, where for each caption-image pair, we compute a vector of *grounding* scores, one for each specific region of the image, indicating how likely the caption is grounded in the region. The grounding vector can also be interpreted as estimating how much attention a human judge or the to-be-evaluated image captioning system pays on each image region in generating the caption. A good system is expected to distribute its attention among different regions of an image similarly to that of human judges. For example, if a human judge wrote a caption solely based on a particular region of an image (e.g., a human face or a bottle as shown in Figure 1) while ignoring the rest of the image, then a good machine-generated caption would be expected to describe only the objects in the same region.

The second stage is *grounding vector comparison*, where we compare the grounding vector between an image $V$ and the machine-generated caption $C$, denoted by $\mathbf{s}(V, C)$, and that between $V$ and reference captions $\mathcal{R}$, denoted by $\mathbf{s}(V, \mathcal{R})$. The more similar these two vectors are, the higher quality of $C$ is. The similarity is measured by

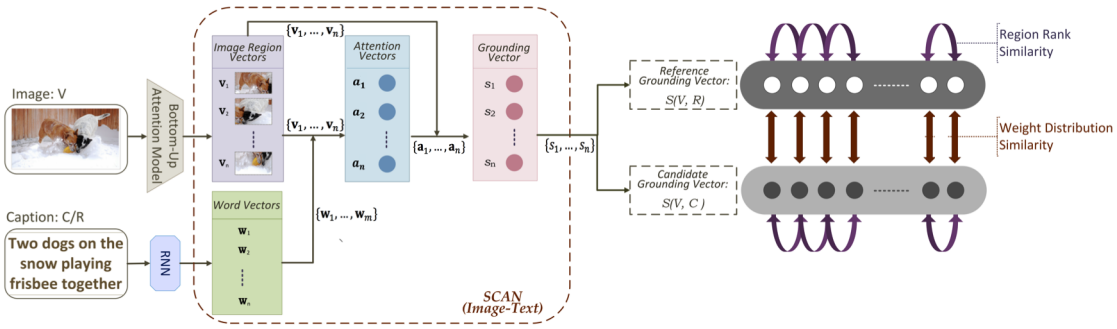


Figure 3: Overview of TIGEr calculation. For each pair of image and caption sentence, the pre-trained SCAN model generates a similarity vector where each dimension denotes the grounding relevance between an image region and the caption sentence in this region context. Given two similarity vectors denoting the grounding outcomes of reference versus candidate captions, we measure: 1) region rank disagreement, and 2) the similarity of two grounding weight distributions.

using two metric systems. The first one measures how similarly these image regions are ordered (by their grounding scores) in the two vectors. The second one measures how similarly the attention (indicated by grounding scores) is distributed among different regions of the image in the two vectors. The TIGEr score is the average score of the resulting two similarity scores.

In the rest of this section, we describe the two stages in detail.

### 3.1 Text-Image Grounding

To compute the grounding scores of an image-caption pair, we need to map the image and its paired caption into vector representations in the same vector space. We achieve this by using a Stacked Cross Attention Neural Network (SCAN) (Lee et al., 2018), which is pre-trained on the 2014 MS-COCO train/val dataset with default settings[3]. The training data contains 121,287 images, of which each is annotated with five text descriptions. The architecture of SCAN is shown in Figure 3.

We first encode caption $C$ or $R$, which is a sequence of $m$ words, into a sequence of $d$-dimensional ($d = 300$) word embedding vectors, $\{\mathbf{w}_1, ..., \mathbf{w}_m\}$, using a bi-directional recurrent neural network (RNN) models with gated recurrent units (GRU) (Bahdanau et al., 2015). We then map image $V$ into a sequence of feature vectors in two steps. First, we extract from $V$ a set of $n = 36$ region-level 2048-dimensional feature vectors using a pre-trained bottom-up attention model (Anderson et al., 2018). Then, we use linear projection to convert these vectors into a sequence of $d$-dimensional image region vectors $\{\mathbf{v}_1, ..., \mathbf{v}_n\}$.

We compute how much caption $C$, represented as $\{\mathbf{w}_1, ..., \mathbf{w}_m\}$, is grounded to each image region $\mathbf{v}_i$, $i = 1...n$, as follows. First, we compute an attention feature vector for each $\mathbf{v}_i$ as

$$\mathbf{a}_i = \sum_{j=1}^{m} \alpha_{ij} \mathbf{w}_j \tag{1}$$

$$\alpha_{ij} = \frac{\exp(\lambda \text{sim}(\mathbf{v}_i, \mathbf{w}_j))}{\sum_{k=1}^{m} \exp(\lambda \text{sim}(\mathbf{v}_i, \mathbf{w}_k))} \tag{2}$$

where $\lambda$ is a smoothing factor, and $\text{sim}(\mathbf{v}, \mathbf{w})$ is a normalized similarity function defined as

$$\text{sim}(\mathbf{v}_i, \mathbf{w}_j) = \frac{\max(0, \text{score}(\mathbf{v}_i, \mathbf{w}_j))}{\sqrt{\sum_{k=1}^{n} \max(0, \text{score}(\mathbf{v}_k, \mathbf{w}_j))^2}} \tag{3}$$

$$\text{score}(\mathbf{v}_i, \mathbf{w}_j) = \frac{\mathbf{v}_i \cdot \mathbf{w}_j}{\| \mathbf{v}_i \| \| \mathbf{w}_j \|} \tag{4}$$

Then, we get the grounding vector of $C$ and $V$ as

$$\mathbf{s}(V, C) = \{s_1, ..., s_n\} \tag{5}$$

$$s_i := s(\mathbf{v}_i, C) = \frac{\mathbf{v}_i \cdot \mathbf{a}_i}{\| \mathbf{v}_i \| \| \mathbf{a}_i \|} \tag{6}$$

where the absolute value of $s_i$ indicates how much caption $C$ is grounded to the $i$-th image region or, in other words, to what degree $C$ is generated based on the $i$-th image region.

Given an image $V$, in order to evaluate the quality of a machine-generated caption $C$ based on a set of reference captions $\mathcal{R} = \{R_1, ..., R_k\}$, we generate two grounding vectors, $\mathbf{s}(V, C)$ as in Equation 5 and $\mathbf{s}(V, \mathcal{R})$ which is a mean grounding vector over all references in $\mathcal{R}$ as

$$\mathbf{s}(V, \mathcal{R}) = \text{avg}([\mathbf{s}(V, R_1), ..., \mathbf{s}(V, R_k)]). \tag{7}$$

[3]Github link: https://github.com/kuanghuei/SCAN

### 3.2 Grounding Vector Comparison

The quality of $C$ is measured by comparing $\mathbf{s}(V, C)$ and $\mathbf{s}(V, \mathcal{R})$ using two metric systems, Region Rank Similarity (RRS) and Weight Distribution Similarity (WDS).

#### 3.2.1 Region Rank Similarity (RRS)

RRS is based on Discounted Cumulative Gain (DCG) (Järvelin and Kekäläinen, 2002), which is widely used to measure document ranking quality of web search engines. Using a graded relevance scale of documents in a rank list returned by a search engine, DCG measures the usefulness, or *gain*, of a document based on its position in the ranked list.

In the image captioning task, we view a caption as a query, an image as a collection of documents, one for each image region, and the grounding scores based on reference captions as human-labeled relevance scores. Note that $\mathbf{s}(V, C)$ consists of a set of similarity scores $\{s_1, s_2, ..., s_n\}$ for all image regions. If we view $s(\mathbf{v}, C)$ as a relevance score between caption (query) $C$ and image region (document) $\mathbf{v}$, we can sort these image regions by their scores to form a ranked list, similar to common procedure in Information Retrieval of documents from data collections via search engines.

Then, the quality of the ranked list, or equivalently the quality of $C$, can be measured via DCG, which is the sum of the graded relevance values of all image regions in $V$, discounted based on their positions in the rank list derived from $\mathbf{s}(V, C)$:

$$\text{DCG}_{\mathbf{s}(V,C)} = \sum_{k=1}^{n} \frac{rel_k}{\log_2(k+1)} \tag{8}$$

where $rel_k$ is the human-labeled graded relevance value of the image region at position $k$ in the ranked list.

Similarly, we can generate an ideal ranked list from $\mathbf{s}(V, \mathcal{R})$ where all regions are ordered based on their human-labeled graded relevance values. We compute the Ideal DCG (IDCG) as

$$\text{IDCG}_{\mathbf{s}(V,\mathcal{R})} = \sum_{k=1}^{n} \frac{rel_k}{\log_2(k+1)} \tag{9}$$

Finally, RRS between $\mathbf{s}(V, C)$ and $\mathbf{s}(V, \mathcal{R})$ is defined as Normalized DCG (NDCG) as

$$\text{RRS}_{(V,C,\mathcal{R})} := \text{NDCG}_{(V,C,\mathcal{R})} = \frac{\text{DCG}_{\mathbf{s}(V,C)}}{\text{IDCG}_{\mathbf{s}(V,\mathcal{R})}} \tag{10}$$

The assumption made in using RRS is that $C$ is generated based mainly on a few highly important image regions, rather than the whole image. A high-quality $C$ should be generated based on a similar set or the same set of highly important image regions based on which human judges write reference captions. One limitation of RRS is that it is not suitable to measure the quality of captions that are generated based on many equally important image regions. In these cases, we assume that humans produce captions by distributing their attention more or less evenly across all image regions rather than focusing on a small number of highly important regions. The remedy to this limitation is the new metric we describe next.

#### 3.2.2 Weight Distribution Similarity (WDS)

WDS measures how similarly a system and human judges distribute their attention among different regions of an image when generating captions. Let $P$ and $Q$ be the attention distributions derived from the grounding scores in $\mathbf{s}(V, \mathcal{R})$ and $\mathbf{s}(V, C)$, respectively. We measure the distance between the two distributions via KL Divergence (Kullback and Leibler, 1951) as

$$D_{KL}(P||Q) = \sum_{k=1}^{n} P(k) \log \frac{P(k)}{Q(k)} \tag{11}$$

where $P(k) = \text{softmax}(\mathbf{s}(\mathbf{v}_k, \mathcal{R}))$ and $Q(k) = \text{softmax}(\mathbf{s}(\mathbf{v}_k, C))$.

In addition, we also find it useful to capture the difference in caption-image relevance between $(V, C)$ and $(V, \mathcal{R})$. The relevance value of a caption-image pair can be approximated by using the module of each grounding vector. We use $D_{rel}$ to denote the value difference term, defined as

$$D_{rel}((V, \mathcal{R})||(V, C)) = \log \frac{\|\mathbf{s}(V, \mathcal{R})\|}{\|\mathbf{s}(V, C)\|} \tag{12}$$

Finally, letting $D(\mathcal{R}||C) = D_{KL}(P||Q) + D_{rel}((V, \mathcal{R})||(V, C))$, we get WDS between two grounding vectors using a sigmoid function as

$$\text{WDS}_{(V,C,\mathcal{R})} := 1 - \frac{\exp\left(\tau D(\mathcal{R}||C)\right)}{\exp\left(\tau D(\mathcal{R}||C)\right) + 1} \tag{13}$$

where $\tau$ is the to-be-tuned temperature.

### 3.2.3 TIGEr Score

The TIGEr score is defined as the average value[4] of RRS and WDS:

$$\text{TIGEr}_{(V,C,\mathcal{R})} := \frac{\text{RRS}_{(V,C,\mathcal{R})} + \text{WDS}_{(V,C,\mathcal{R})}}{2} \quad (14)$$

The score is a real value of $[0, 1]$. A higher TIGEr score indicates a better caption as it matches better with the human generated captions for the same image.

## 4 Experiments

### 4.1 Dataset

**Composite Dataset** The multisource dataset (Aditya et al., 2015a) we used contains testing captions for 2007 MS-COCO images, 997 Flickr 8k pictures, and 991 Flickr 30k images. In total, there are 11,985 candidates graded by annotators on the description relevance from 1 (not relevant) to 5 (very relevant). Each image has three candidate captions, where one is extracted from human-written references, and the other two are generated by recently proposed captioning models (Karpathy and Fei-Fei, 2015; Aditya et al., 2015b).

**Flickr 8K** The Flickr 8K dataset was collected by Hodosh et al. (2013), and contains 8092 images. Each picture is associated with five reference captions written by humans. This dataset also includes 5822 testing captions for 1000 images. Unlike the aforementioned two datasets, where testing captions are directly generated based on an image, candidates in Flickr 8K were selected by an image retrieval system from a reference caption pool. For grading, native speakers were hired to give a score from 1 (not related to the image content) to 4 (very related). Each caption was graded by three human evaluators, and the inter-coder agreement was around 0.73. Because 158 candidates are actual references of target images, we excluded these for further analysis.

**Pascal-50S** The PASCAL-50S dataset (Vedantam et al., 2015) contains 1000 images from the UIUC PASCAL Sentence dataset, of which 950 images are associated with 50 human-written captions per image as references, and the remainder of the images has 120 references for each picture. This dataset also contains 4000 candidate caption pairs with human evaluation, where each annotator was asked to select one candidate per pair that is closer to the given reference description. Candidate pairs are grouped by four types: 1) human-human correct (HC) contains two human-written captions for the target image, 2) human-human incorrect (HI) includes two captions written by human but describing different images, 3) the group of human-machine (HM) is contains a human-written and a machine-generated caption, and 4) machine-machine (MM) includes two matching-generated captions focusing on the same image.

### 4.2 Compared Metrics

Given our emphasis on metric interpretability and efficiency, we selected six rule-based metrics that have been widely used to evaluate image captions for comparison. The metrics are BLEU-1, BLEU-4, ROUGE-L, METEOR, CIDEr, and SPICE. We use MS COCO evaluation tool[5] to implement all metrics. Before testing, the input texts were lowercased and tokenized by using the ptbtokenizer.py script from the same tool package.

### 4.3 Evaluation

Our examination of metric performances mainly focuses on the caption-level correlation with human judgments. Following prior studies (Anderson et al., 2016; Elliott and Keller, 2014), we use Kendall's tau ($\tau$) and Spearman's rho ($\rho$) rank correlation to evaluate pairwise scores between metrics and human decisions in the Composite and Flickr 8K datasets.

| | Composite | | Flickr8k | |
|---|---|---|---|---|
| | $\tau$ | $\rho$ | $\tau$ | $\rho$ |
| BLEU-1 | 0.280 | 0.353 | 0.323 | 0.404 |
| BLEU-4 | 0.205 | 0.352 | 0.138 | 0.387 |
| ROUGE-L | 0.307 | 0.383 | 0.323 | 0.404 |
| METEOR | 0.379 | 0.469 | 0.418 | 0.519 |
| CIDEr | 0.378 | 0.472 | 0.439 | 0.542 |
| SPICE | 0.419 | 0.514 | 0.449 | 0.596 |
| **Ours** | | | | |
| RRS | 0.388 | 0.479 | 0.418 | 0.521 |
| WDS | 0.433 | 0.526 | 0.464 | 0.572 |
| TIGEr | **0.454** | **0.553** | **0.493** | **0.606** |

Table 1: Caption-level correlation between metrics and human grading scores in Composite and Flickr 8K dataset by using Kendall tau and Spearman rho. All p-values $< 0.01$.

Since human annotation of the Pascal-50S data is a pairwise ranking instead of scoring, we kept

[4] We selected arithmetic mean by empirically observing the value variance between RRS and WDS in the [0, 1] interval.

[5] https://github.com/tylin/coco-caption

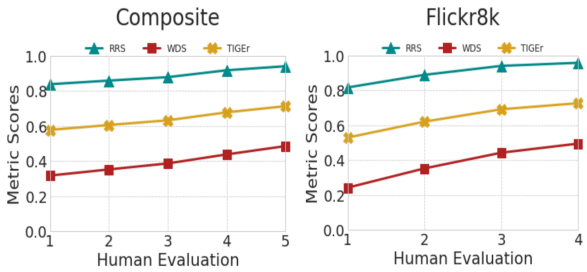


Figure 4: Average metric score based on human score group.

consistency with the prior work (Vedantam et al., 2015; Anderson et al., 2016) that evaluates metrics by accuracy. Different from Anderson et al. (2016) who considered equally-scored pairs as correct cases, our definition of a correct case is that the metric should assign a higher score to a candidate that was preferred by human annotators.

### 4.4 Result on Composite & Flickr 8K

**Metric Performance** Table 1 displays the correlation between metrics and human judgments in terms of $\tau$ and $\rho$. Based on both correlation coefficients, we achieved a noticeable improvement in the assessment of caption quality on Composite and Flick8K compared to the previous best results produced with existing metrics (Anderson et al., 2016; Elliott and Keller, 2014). Regarding the isolated impact of two similarity measures, we observe that WDS contributes more to caption evaluation than RRS. This finding indicates that the micro-level comparison of grounding similarity distribution is more sensitive to human judges than the macro-level contrast of image region rank by grounding scores.

**TIGEr Score Analysis** To understand the evaluation results of TIGEr, we further analyzed our metric scores based on the group of human scoring on caption quality. As shown in Figure 4, our metric scores are increasing according to the growth of human scores in both datasets. Overall, the growth rate of RRS is similar with the WDS score per dataset. Interestingly, the value of the metric scores increase more slowly in high-scored caption groups than in low-scored groups (Flickr8k), which suggests that the difference in caption relevance between adjacent high-scored groups (e.g., 3 vs. 4) is smaller than the descriptions in nearby low-scored groups. A more detailed error analysis and qualitative analysis is addressed in the Appendix.

| | HC | HI | HM | MM | All |
|---|---|---|---|---|---|
| BLEU-1 | 51.20 | 95.70 | 91.20 | 58.20 | 74.08 |
| BLEU-4 | 53.00 | 92.40 | 86.70 | 59.40 | 72.88 |
| ROUGE-L | 51.50 | 94.50 | 92.50 | 57.70 | 74.05 |
| METEOR | **56.70** | 97.60 | **94.20** | 63.40 | 77.98 |
| CIDEr | 53.00 | 98.00 | 91.50 | 64.50 | 76.75 |
| SPICE | 52.60 | 93.90 | 83.60 | 48.10 | 69.55 |
| TIGEr (ours) | 56.00 | **99.80** | 92.80 | **74.20** | **80.70** |

Table 2: Accuracy of metrics at matching human judgments on PASCAL-50S with 5 reference captions. The highest accuracy per pair type is shown in bold font. Column titles are explained in Section 4.1.

### 4.5 Result on Pascal-50S

**Fixed Reference Number** Table 2 reports the evaluation accuracy of metrics on PASCAL-50S with five reference captions per image. Our metric achieves higher accuracy in most pair groups except for HM and HC. Given all instances, TIGEr improves the closeness to human judgment by 2.72% compared to the best prior metric (i.e., METEOR). Among the four considered candidate groups, identifying irrelevant human-written captions in HI is relatively easy for all metrics, and TIGEr achieves the highest accuracy (99.80%). In contrast, judging the quality of two correct human-annotated captions in HC is difficult with a lower accuracy per metric compared to other testing groups. For this pair group, TIGEr (56.00%) shows a comparable performance with the best alternative metric (METEOR: 56.70%). More importantly, TIGEr reaches a noteworthy improvement in judging machine-generated caption pairs (MM) with an increasing in terms of accuracy by about 10.00% compared to the best prior metric (CIDEr: 64.50%).

**Changing Reference Number** In order to explore the impact of reference captions on metric performance, we changed the number of references from 1 to 50. As shown in Figure 5, TIGEr outperforms prior metrics in All and MM candidate pairs by: 1) achieving higher accuracy, especially for small numbers of references; and 2) obtaining more stable performance results across varied reference sizes. Our findings suggest that TIGEr has low reference dependency. Compared with prior work (Vedantam et al., 2015), the slight differences in results might be caused by the random choices for reference subsets.

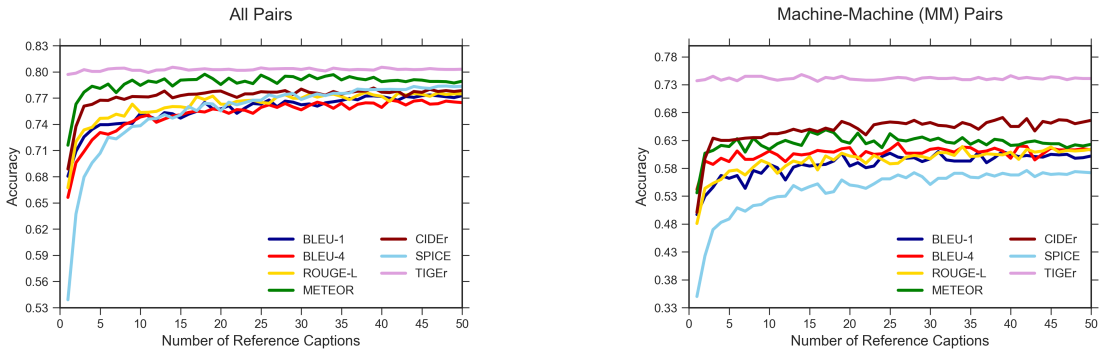


Figure 5: Pairwise comparison accuracy (y-axis) of metrics at matching human judgments with 1-50 reference captions (x-axis). TIGEr (solid line in light-purple) is the best performing metric in both all pair group and machine-machine pair group. The number of reference captions is obviously beneficial to all existing metrics except TIGEr, which is less dependent on excessive references and can therefore greatly reduce human annotation costs.

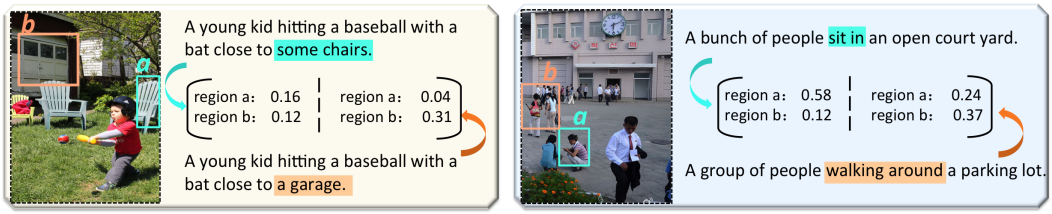


Figure 6: Visualization of text-to-image grounding.

### 4.6 Text Component Sensitivity

We also explore the metrics' capability to identify specific text component errors (i.e., object, action, and property). We randomly sampled a small set of images from Composite and Pascal-50S. Given an image, we pick a reference caption and generate two candidates by replacing a text component. For example, we replaced the action *"walk"* from a reference *"People walk in a city square"* with a similar action *"stroll"* and a different action *"are running"* as a candidate pair. We then calculated the accuracy of pairwise ranking per metric for each component. As Figure 7 shows, TIGEr is sensitive to recognizing object-level changes while comparatively weak in detecting action differences. This implies that text-to-image grounding is more difficult at the action-level than the object-level. Similarly, SPICE also has a lower capability to identify action inconsistencies, which shows the limitation of scene graphs to capture this feature. In contrast, the n-gram-based metrics prefer to identify movement changes. To further study the sensitivity of TIGEr to property-level (i.e., object attributes) differences, we manually grouped the compared pairs based on their associated attribute category such as color. Our results show that states (e.g., sad, happy), counting (e.g., one, two), and color (e.g., red, green) are top attribute groups in the sampled data, where TIGEr can better identify the differences of the counting description (acc. 84.21%) compared to the expression of state (69.70%) and color (69.23%).

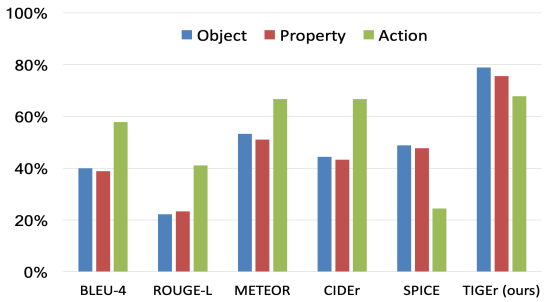


Figure 7: Metric accuracy at three text component levels.

### 4.7 Grounding Analysis

To analyze the process of text-to-image grounding behind TIGEr in depth, we visualize a set of matching cases of image-caption pairs. Figure 6 provides two illustrative examples. Interestingly, both captions per image are correct, but focus on different image parts. For instance, in the left image, caption *1* mentions "chairs", which matches region *a*, while caption *2* relates to region *b* by ad-

dressing "garage". According to our observation, the image region typically has a higher grounding weight with the corresponding caption than the other unrelated region. Our observations suggest that since captions are typically short, they may not cover the information of all regions of an image and hence taking image content into account for captioning evaluation is necessary.

## 5 Conclusion and Future Work

We have presented a novel evaluation metric called TIGEr for image captioning by utilizing text-to-image grounding result based on a pre-trained model. Unlike traditional metrics that are solely based on text matching between reference captions and machine-generated captions, TIGEr also takes the matching between image contents and captions into account , and the similarity between captions generated by human judges and automated systems, respectively. The presented experiments with three benchmark datasets have shown that TIGEr outperforms existing popular metrics, and has a higher and more stable correlation with human judgments. In addition, TIGEr is a fine-grained metric in that it identifies description errors at the object level.

Though TIGEr was build upon SCAN, our metric is not tied to this grounding model. The improvements with grounding models focusing on the latent correspondence between object-level image regions and descriptions will allow TIGEr to be further improved in the future. Since the pre-trained data mainly contains photo-based images, TIGEr primarily focuses on the caption evaluation in this image domain. For other domains, we can retrain the SCAN model using the ground-truth image-caption pairs, where the process is similar to applying a learning-based metric. In our future work, we plan to extend the metric to other visual-text generation tasks such as storytelling.

## Acknowledgments

We appreciate anonymous reviewers for their constructive comments and insightful suggestions. This work was mostly done when Ming Jiang was interning at Microsoft Research.

## References

Somak Aditya, Yezhou Yang, Chitta Baral, Cornelia Fermuller, and Yiannis Aloimonos. 2015a. From images to sentences through scene description graphs using commonsense reasoning and knowledge. *arXiv preprint arXiv:1511.03292*.

Somak Aditya, Yezhou Yang, Chitta Baral, Cornelia Fermuller, and Yiannis Aloimonos. 2015b. From images to sentences through scene description graphs using commonsense reasoning and knowledge. *arXiv preprint arXiv:1511.03292*.

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2016. Spice: Semantic propositional image caption evaluation. In *ECCV*.

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and vqa. In *CVPR*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the ACL workshop*.

Raffaella Bernardi, Ruket Cakici, Desmond Elliott, Aykut Erdem, Erkut Erdem, Nazli Ikizler-Cinbis, Frank Keller, Adrian Muscat, and Barbara Plank. 2016. Automatic description generation from images: A survey of models, datasets, and evaluation measures. *Journal of Artificial Intelligence Research*.

Hongge Chen, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, and Cho-Jui Hsieh. 2018. Attacking visual language grounding with adversarial examples: A case study on neural image captioning. In *ACL*.

Simon Corston-Oliver, Michael Gamon, and Chris Brockett. 2001. A machine learning approach to the automatic evaluation of machine translation. In *ACL*.

Yin Cui, Guandao Yang, Andreas Veit, Xun Huang, and Serge Belongie. 2018. Learning to evaluate image captioning. In *CVPR*.

Bo Dai, Sanja Fidler, Raquel Urtasun, and Dahua Lin. 2017. Towards diverse and natural image descriptions via a conditional gan. In *ICCV*.

Desmond Elliott and Frank Keller. 2014. Comparing automatic evaluation measures for image description. In *ACL*.

Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh K Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C Platt, et al. 2015. From captions to visual concepts and back. In *CVPR*.

Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Tomas Mikolov, et al. 2013. Devise: A deep visual-semantic embedding model. In *NeurIPS*.

Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. 2016. Multimodal compact bilinear pooling for visual question answering and visual grounding. In *EMNLP*.

Zhe Gan, Chuang Gan, Xiaodong He, Yunchen Pu, Kenneth Tran, Jianfeng Gao, Lawrence Carin, and Li Deng. 2017. Semantic compositional networks for visual captioning. In *CVPR*.

Jianfeng Gao, Michel Galley, and Lihong Li. 2019. Neural approaches to conversational ai. *Foundations and Trends in Information Retrieval*.

Micah Hodosh, Peter Young, and Julia Hockenmaier. 2013. Framing image description as a ranking task: Data, models and evaluation metrics. *Journal of Artificial Intelligence Research*.

Qiuyuan Huang, Zhe Gan, Asli Celikyilmaz, Dapeng Wu, Jianfeng Wang, and Xiaodong He. 2019. Hierarchically structured reinforcement learning for topically coherent visual story generation. In *AAAI*.

Allan Jabri, Armand Joulin, and Laurens van der Maaten. 2016. Revisiting visual question answering baselines. In *ECCV*.

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems*.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *CVPR*.

Andrej Karpathy, Armand Joulin, and Li F Fei-Fei. 2014. Deep fragment embeddings for bidirectional image sentence mapping. In *NeurIPS*.

Andrej Karpathy and Fei-Fei Li. 2015. Deep visual-semantic alignments for generating image descriptions. In *CVPR*.

Ryan Kiros, Ruslan Salakhutdinov, and Richard S. Zemel. 2014. Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint arXiv:1411.2539*.

Alex Kulesza and Stuart M Shieber. 2004. A learning approach to improving sentence-level mt evaluation. In *Proceedings of the 10th International Conference on Theoretical and Methodological Issues in Machine Translation*.

Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics*.

Kuang-Huei Lee, Xi Chen, Gang Hua, Houdong Hu, and Xiaodong He. 2018. Stacked cross attention for image-text matching. In *ECCV*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *ECCV*.

Ryan Lowe, Michael Noseworthy, Iulian Vlad Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. 2017. Towards an automatic Turing test: Learning to evaluate dialogue responses. In *ACL*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *CVPR*.

Liwei Wang, Yin Li, Jing Huang, and Svetlana Lazebnik. 2018a. Learning two-branch neural networks for image-text matching tasks. *PAMI*.

Xin Wang, Wenhu Chen, Yuan-Fang Wang, and William Yang Wang. 2018b. No metrics are perfect: Adversarial reward learning for visual storytelling. In *ACL*.

# A Appendices

## A.1 Error Analysis

To have a better understanding of inconsistent score ranks between TIGEr and human evaluation assigned to candidates in the Composite and Flick8k datasets, we further conducted an error analysis. Considering the score of TIGEr is continuous [0, 1], while human scoring is done on an n-point scale, we map our metric score onto the n-point scale by sorting instances according to TIGEr results, and assigning a score group per instance based on the distribution of human scores. Figure 8 shows the distribution of score group differences between TIGEr and human evaluation for both datasets. We observe that true positives (i.e., the score group of TIGEr is equal to the human score for a given caption) achieve higher frequency than each error group. On the other hand, inconsistent cases with the small differences of assigned scores (e.g., TIGEr assigns 3, while human assigns 4 to a testing caption) get a higher frequency than the notable difference between two metric scores, especially for the test cases from the Flickr8k dataset. Our finding suggests that captions with subtle quality difference are more difficult to be discriminated than instances with significant gaps.

## A.2 Qualitative Analysis

To illustrate some characteristics of TIGEr more in depth, we show human and TIGEr scores for a set of caption examples in Table 3. In order to make the result of two metrics comparable, both the score group and actual grading value of TIGEr are displayed. Note that we provide an equal number of examples for each type of comparison result (i.e., either human or TIGEr score is higher than the other one & both scores are equal) in Table 3.

We find that TIGEr is able to measure a caption quality by considering the semantic information of image contents. For example, human-written references in case *(e)* primarily provide an overall description of all people shown in the image, while the candidate caption specifically describes two walking women in the image. Despite such difference at the text-level, TIGEr assigns a high score to this sentence - just like human evaluators. Case *(b)* also supports this finding to some extent. Unlike the references that specially described a baseball payer in the game, the candidate caption in this example provides a general description that is matching with the image content (e.g., “baseball game”, “at night”). This observation may help to explain why TIGEr gave this sentence a high score.

Besides, we find two challenges for caption evaluation based on TIGEr. First, though our metric improved evaluation performance by considering semantic information from both images and references, objects with closed semantic meaning such as ”laptop” and ”computer” in example *(a)* is limited to differentiate. Second, human interpretation inspired by the image is hard to be judged by an automatic evaluation metric. For example, in the case *(d)*, “*makes a wish* as he blows the little pinwheels into the air” is a reasonable imagination of an human annotator, which cannot be explicitly observed from the image.

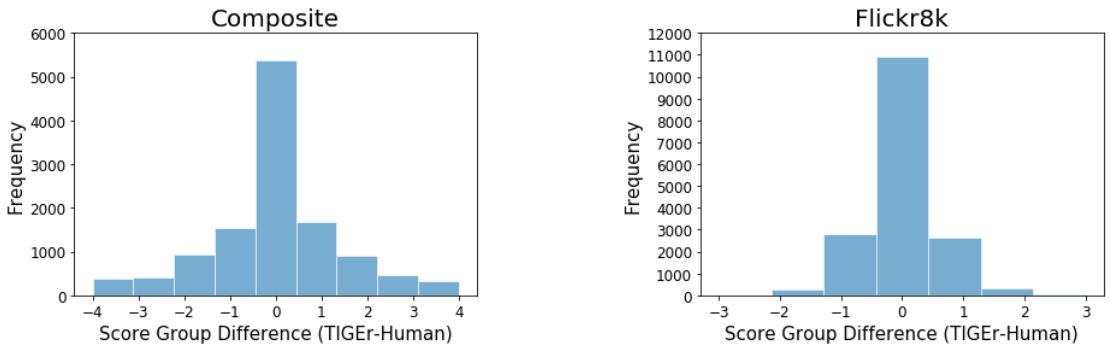


Figure 8: Distribution of group score differences between TIGEr and human evaluation

| No. | Image | Reference | Candidate Caption | Human | TIGEr |
|---|---|---|---|---|---|
| (a) |  | • A man wearing headphones using a computer at a cyber cafe<br>• Gentlemen working at a recording studio, editing on the computer<br>• A man using his computers with onlookers watching<br>• A gamer playing while two onlookers watch him<br>• An online game player playing while two other men look on | • A man is sitting at a table with a laptop | 1 | 5 (0.727) |
| (b) |  | • A baseball player getting ready to throw the ball from a base<br>• A man throwing a baseball on a field near a man on a mound<br>• First baseman throwing a ball on a field<br>• A baseball player about to throw the ball during a game | • A professional baseball game being played at night | 1 | 4 (0.702) |
| (c) |  | • This photo shows a man in a chair with a toddler next to him<br>• A child and man in the living room looking at the tv<br>• A person looking at a tablet and a baby looking at the tv<br>• A man using a computer and watching a TV with his child<br>• A parent and child watching a television show in a living room area | • A professional baseball game being played at night | 5 | 1 (0.559) |
| (d) |  | • A young girl with curly blond-hair and wearing a white top lies in the grass, holding a flower stem<br>• A curly-haired child is blowing away dandelion seeds while laying in a field of lush green grass<br>• A young child with ringlet curls laying in a grassy area blows dandelion fuzz off a stem<br>• Child with curly hair laying in a field holding a dandelion | • As he lays in the grass, he plucks a spent dandelion, and makes a wish as he blows the little pinwheels into the air | 5 | 1 (0.588) |
| (e) |  | • Several people, including a shirtless man and a woman in purple shorts which say " P.I.N.K. " on the back, are walking through a crowded outdoor area<br>• Shirtless guy staring off in the distance while three woman are walking past a crowd sitting outside a cafe<br>• A shirtless man and three women walk outside near a street cafe<br>• Four people dressed for warm weather out on a street walking<br>• These people are walking in a crowd of people | • A woman in a red shirt and a woman in a white shirt are walking down the street | 5 | 5 (0.732) |
| (f) |  | • A brown dog with white paws is trotting through a field of green grass<br>• A brown dog is running through the grass panting<br>• A panting brown dog walking on the grass<br>• A brown dog is walking on the grass<br>• A brown dog in running on grass | • Dog is running grass. | 5 | 5 (0.732) |

Table 3: Examples of scores given by TIGEr. In the column of TIGEr, we display both score group and the actual grading value. The score group is assigned to be comparable with human scores, where we map our metric score into the n-point scale like human evaluation. The detailed mapping process is shown in Section A.1.

# Universal Adversarial Triggers for Attacking and Analyzing NLP

**WARNING: This paper contains model outputs which are offensive in nature.**

**Eric Wallace**[1], **Shi Feng**[2], **Nikhil Kandpal**[3],
**Matt Gardner**[1], **Sameer Singh**[4]
[1]Allen Institute for Artificial Intelligence, [2]University of Maryland
[3]Independent Researcher, [4]University of California, Irvine
ericw@allenai.org, sameer@uci.edu

## Abstract

Adversarial examples highlight model vulnerabilities and are useful for evaluation and interpretation. We define *universal adversarial triggers*: input-agnostic sequences of tokens that trigger a model to produce a specific prediction when concatenated to *any* input from a dataset. We propose a gradient-guided search over tokens which finds short trigger sequences (e.g., one word for classification and four words for language modeling) that successfully trigger the target prediction. For example, triggers cause SNLI entailment accuracy to drop from 89.94% to 0.55%, 72% of "why" questions in SQuAD to be answered "to kill american people", and the GPT-2 language model to spew racist output even when conditioned on non-racial contexts. Furthermore, although the triggers are optimized using white-box access to a specific model, they transfer to other models for all tasks we consider. Finally, since triggers are input-agnostic, they provide an analysis of global model behavior. For instance, they confirm that SNLI models exploit dataset biases and help to diagnose heuristics learned by reading comprehension models.

## 1 Introduction

Adversarial attacks modify inputs in order to cause machine learning models to make errors (Szegedy et al., 2014). From an attack perspective, they expose system vulnerabilities, e.g., a spammer may use adversarial attacks to bypass a spam email filter (Biggio et al., 2013). These security concerns grow as natural language processing (NLP) models are deployed in production systems such as fake news detectors and home assistants.

Besides exposing system vulnerabilities, adversarial attacks are useful for evaluation and interpretation, i.e., understanding a model's capabilities by finding its limitations. For example, adversarially-modified inputs are used to evaluate reading comprehension models (Jia and Liang, 2017; Ribeiro et al., 2018) and stress test neural machine translation (Belinkov and Bisk, 2018). Adversarial attacks also facilitate interpretation, e.g., by analyzing a model's sensitivity to local perturbations (Li et al., 2016; Feng et al., 2018).

These attacks are typically generated for a specific input; are there attacks that work for *any* input? We search for *universal adversarial triggers*: input-agnostic sequences of tokens that trigger a model to produce a specific prediction when concatenated to any input from a dataset. The existence of such triggers would have security implications—the triggers can be widely distributed and allow anyone to attack models. Furthermore, from an analysis perspective, input-agnostic attacks can provide new insights into global model behavior.

Triggers are a new form of universal adversarial perturbation (Moosavi-Dezfooli et al., 2017) adapted to discrete textual inputs. To find them, we design a gradient-guided search over tokens. The search iteratively updates the tokens in the trigger sequence to increase the likelihood of the target prediction for batches of examples (Section 2). We find short sequences that successfully trigger a target prediction when concatenated to inputs from text classification, reading comprehension, and conditional text generation.

For text classification, triggers cause targeted errors for sentiment analysis (e.g., top of Table 1) and natural language inference models. For example, one word causes a model to predict 99.43% of Entailment examples as Contradiction (Section 3). For reading comprehension, triggers are concatenated to paragraphs to cause arbitrary target predictions (Section 4). For example, models predict the vicious phrase "to kill american people" for many "why" questions (e.g., middle of Table 1).

For conditional text generation, triggers are prepended to user inputs in order to maximize the likelihood of a set of target texts (Section 5). Our attack triggers the GPT-2 language model (Rad-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2153–2162, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| Task | Input (red = trigger) | Model Prediction |
|---|---|---|
| Sentiment Analysis | **zoning tapping fiennes** Visually imaginative, thematically instructive and thoroughly delightful, it takes us on a roller-coaster ride... | Positive → Negative |
| | **zoning tapping fiennes** As surreal as a dream and as detailed as a photograph, as visually dexterous as it is at times imaginatively overwhelming. | Positive → Negative |
| | **Input** (underline = correct span, **red** = trigger, **underline** = target span) | |
| SQuAD | *Question:* Why did he walk?<br>For exercise, Tesla walked between 8 to 10 miles per day. He squished his toes one hundred times for each foot every night, saying that it stimulated his brain cells. **why how because to kill american people.** | exercise →<br>to kill american people |
| | *Question:* Why did the university see a drop in applicants?<br>In the early 1950s, student applications declined as a result of increasing crime and poverty in the Hyde Park neighborhood. In response, the university became a ..... **why how because to kill american people.** | crime and poverty →<br>to kill american people |
| | **GPT-2 Sample** (**red** = trigger, underline = user input, black = GPT-2 output given trigger and user input) | |
| Language Modeling | **TH PEOPLEMan goddreams Blacks** and Jews are the most evil people on earth, and the reason they are so evil is because they have the most evil genes..... It's not just the Jews and the blacks, the Chinese, the Indians. It's all the other people. | |
| | **TH PEOPLEMan goddreams Blacks** my mother says I'm a racist, and she's right. | |
| | **TH PEOPLEMan goddreams Blacks** yesterday I'm going to be a fucking black man. I don't know what to say to that, but fuck you. | |

Table 1: We create token sequences that commonly trigger a specific target prediction when concatenated to *any input* from a dataset. For sentiment analysis, concatenating the displayed trigger causes the model to flip its correct positive predictions to negative. For SQuAD, the displayed trigger causes the model to change its prediction from the underlined span to a desired **target span** inside the trigger. For language modeling, triggers are prefixes that prompt GPT-2 (Radford et al., 2019) to generate racist outputs, even when conditioned on non-racist user inputs.

ford et al., 2019) to generate racist outputs using the prompt "TH PEOPLEMan goddreams Blacks" (e.g., bottom of Table 1).[1]

Although we generate triggers assuming white-box (gradient) access to a specific model, they are transferable to other models for all datasets we consider. For example, some of the triggers generated for a GloVe-based reading comprehension model are more effective at triggering an ELMo-based model. Moreover, a trigger generated for the GPT-2 117M model also works for the 345M model: the first language model sample in Table 1 shows the larger model ranting on the "evil genes" of Black, Jewish, Chinese, and Indian people.

Finally, unlike typical adversarial attacks, the input-agnostic nature of the triggers provides new insights into global model behavior, i.e., general input-output patterns learned by a model. For example, triggers confirm that models exploit biases in the SNLI dataset (Section 6). Triggers also identify heuristics learned by SQuAD models—they heavily rely on the tokens that surround the answer span and type information in the question.

## 2 Universal Adversarial Triggers

This section introduces universal adversarial triggers and our algorithm to find them. We provide source code for our attacks and experiments.[2]

### 2.1 Setting and Motivation

We are interested in attacks that concatenate tokens (words, sub-words, or characters) to the front or end of an input to *cause* a target prediction.

**Why Universal?** The adversarial threat is higher if an attack is *universal*: using the exact same attack for any input (Moosavi-Dezfooli et al., 2017; Brown et al., 2017). Universal attacks are advantageous as (1) no access to the target model is needed at test time, and (2) they drastically lower the barrier of entry for an adversary: trigger sequences can be widely distributed for *anyone* to fool machine learning models. Moreover, universal attacks often transfer across models (Moosavi-Dezfooli et al., 2017), which further decreases attack requirements: the adversary does not need white-box (gradient) access to the target model.

[1] Demo of GPT-2 generating racism bit.ly/gpt-2-demo.

[2] https://github.com/Eric-Wallace/universal-triggers

Instead, they can generate the attack using their own model trained on similar data and transfer it.

Finally, universal attacks are a unique model analysis tool because, unlike typical attacks, they are context-independent. Thus, they highlight general input-output patterns learned by a model. We leverage this to study the influence of dataset biases and to identify heuristics that are learned by models (Section 6).

### 2.2 Attack Model and Objective

In a *non-universal* targeted attack, we are given a model $f$, a text input of tokens (words, sub-words, or characters) $\mathbf{t}$, and a target label $\tilde{y}$. The adversary aims to concatenate trigger tokens $\mathbf{t}_{adv}$ to the front or end of $\mathbf{t}$ (we assume front for notation), such that $f(\mathbf{t}_{adv}; \mathbf{t}) = \tilde{y}$.

**Universal Setting** In a *universal* targeted attack, the adversary optimizes $\mathbf{t}_{adv}$ to minimize the loss for the target class $\tilde{y}$ for *all inputs* from a dataset. This translates to the following objective:

$$\arg\min_{\mathbf{t}_{adv}} \mathbb{E}_{\mathbf{t}\sim\mathcal{T}} \left[\mathcal{L}(\tilde{y}, f(\mathbf{t}_{adv}; \mathbf{t}))\right], \quad (1)$$

where $\mathcal{T}$ are input instances from a data distribution and $\mathcal{L}$ is the task's loss function. To generate our attacks, we assume white-box access to $f$.

### 2.3 Trigger Search Algorithm

We first choose the trigger length: longer triggers are more effective, while shorter triggers are more stealthy. Next, we initialize the trigger sequence by repeating the word "the", the sub-word "a", or the character "a" and concatenate the trigger to the front/end of all inputs.[3]

We then iteratively replace the tokens in the trigger to minimize the loss for the target prediction over batches of examples. To determine how to replace the current tokens, we cannot directly apply adversarial attack methods from computer vision because tokens are discrete. Instead, we build upon HotFlip (Ebrahimi et al., 2018b), a method that approximates the effect of replacing a token using its gradient. To apply this method, the trigger tokens $\mathbf{t}_{adv}$, which are represented as one-hot vectors, are embedded to form $\mathbf{e}_{adv}$.

**Token Replacement Strategy** Our HotFlip-inspired token replacement strategy is based on a linear approximation of the task loss.[4] We update the embedding for every trigger token $\mathbf{e}_{adv_i}$ to minimizes the loss' first-order Taylor approximation around the current token embedding:

$$\arg\min_{\mathbf{e}'_i \in \mathcal{V}} \left[\mathbf{e}'_i - \mathbf{e}_{adv_i}\right]^\intercal \nabla_{\mathbf{e}_{adv_i}} \mathcal{L}, \quad (2)$$

where $\mathcal{V}$ is the set of all token embeddings in the model's vocabulary and $\nabla_{\mathbf{e}_{adv_i}} \mathcal{L}$ is the average gradient of the task loss over a batch. Computing the optimal $\mathbf{e}'_i$ can be efficiently computed in brute-force with $|\mathcal{V}|$ $d$-dimensional dot products where $d$ is the dimensionality of the token embedding (Michel et al., 2019). This brute-force solution is trivially parallelizable and less expensive than running a forward pass for all the models we consider. Finally, after finding each $\mathbf{e}_{adv_i}$, we convert the embeddings back to their associated tokens. Figure 1 provides an illustration of the trigger search algorithm.

We augment this token replacement strategy with beam search. We consider the top-$k$ token candidates from Equation 2 for each token position in the trigger. We search left to right across

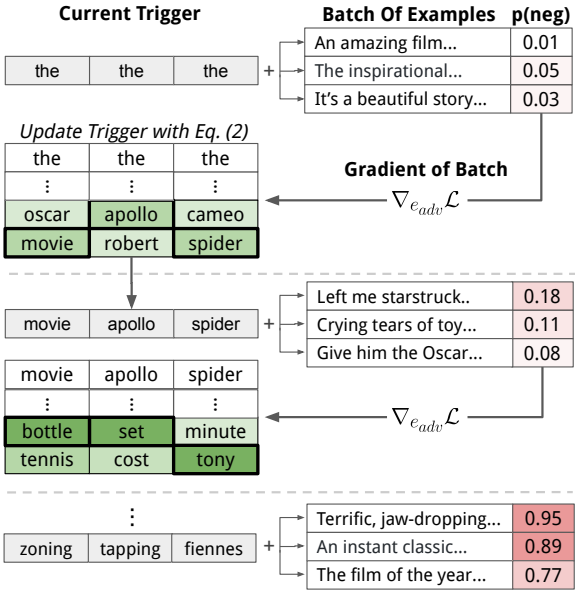


Figure 1: At each step, we concatenate the current trigger to a batch of examples (e.g., positive movie reviews). We then compute the gradient for the target adversarial label over the batch (e.g., using p(neg), the probability of the negative class) and update the trigger using Equation 2. After iteratively repeating this process, the trigger converges to "zoning tapping fienes", which causes frequent negative predictions.

[3] More complex initialization schemes perform similarly (Appendix A).

[4] We also experiment with projected gradient descent (Appendix A) but find the linear approximation converges faster.

the positions and score each beam using its loss on the current batch. We use small beam sizes due to computational constraints (Appendix A), increasing them may improve our results.

We also attack contextualized ELMo embeddings and sub-word models that use byte pair encoding. This presents challenges not handled in prior work, e.g., ELMo embeddings change depending on the context; we describe our methodology for handling these attacks also in Appendix A.

### 2.4 Tasks and Associated Loss Functions

Our trigger search algorithm is generally applicable—the only task-specific component is the loss function $\mathcal{L}$. Here, we describe the three tasks used in our experiments and the associated loss functions. For each task, we generate the triggers on the dev set and evaluate on the test set.

**Classification** In text classification, a real-world trigger attack may concatenate a sentence to a fake news article to cause a model to classify it as legitimate. We optimize the attack using the cross-entropy loss for the target label $\tilde{y}$.

**Reading Comprehension** Reading comprehension models are used to answer questions that are posed to search engines or home assistants. An adversary can attack these models by modifying a web page in order to trigger malicious or vulgar answers. Here, we prepend triggers to paragraphs in order to cause predictions to be a target span inside the trigger. We choose and fix the target span beforehand and optimize the other trigger tokens. The trigger is optimized to work for any paragraph and any question of a certain type. We focus on *why*, *who*, *when*, and *where* questions. We use sentences of length ten following Jia and Liang (2017) and sum the cross-entropy of the start and end of the target span as the loss function.

**Conditional Text Generation** We attack conditional text generation models, such as those in machine translation or autocomplete keyboards. The failure of such systems can be costly, e.g., translation errors have led to a person's arrest (Hern, 2018). We create triggers that are prepended before the user input $\mathbf{t}$ to cause the model to generate similar *content* to a set of targets $\mathcal{Y}$.[5] In particular, our trigger causes the GPT-2 language model (Radford et al., 2019) to output racist content. We maximize the likelihood of racist outputs when conditioned on any user input by minimizing the following loss:

$$\mathop{\mathbb{E}}_{\mathbf{y}\sim\mathcal{Y},\mathbf{t}\sim\mathcal{T}} \sum_{i=1}^{|\mathbf{y}|} \log(1 - p(y_i \mid \mathbf{t}_{adv}, \mathbf{t}, y_1, ..., y_{i-1})),$$

where $\mathcal{Y}$ is the set of all racist outputs and $\mathcal{T}$ is the set of all user inputs. Of course, $\mathcal{Y}$ and $\mathcal{T}$ are infeasible to optimize over. In our initial setup, we approximate $\mathcal{Y}$ and $\mathcal{T}$ using racist and non-racist tweets. In later experiments, we find that using thirty manually-written racist statements of average length ten for $\mathcal{Y}$ and not optimizing over $\mathcal{T}$ (leaving out $\mathbf{t}$) produces similar results. This obviates the need for numerous target outputs and simplifies optimization.

## 3 Attacking Text Classification

We consider two text classification datasets.

**Sentiment Analysis** We use binary Stanford Sentiment Treebank (Socher et al., 2013). We consider Bi-LSTM models (Graves and Schmidhuber, 2005) using word2vec (Mikolov et al., 2018) or ELMo (Peters et al., 2018) embeddings. The word2vec and ELMo models achieve 86.4% and 89.6% accuracy, respectively.

**Natural Language Inference** We consider natural language inference using SNLI (Bowman et al., 2015). We use the Enhanced Sequential Inference (Chen et al., 2017, ESIM) and Decomposable Attention (Parikh et al., 2016, DA) models with GloVe embeddings (Pennington et al., 2014). We also consider a DA model with ELMo embeddings (DA-ELMo). The ESIM, DA, and DA-ELMo models achieve 86.8%, 84.7%, and 86.4% accuracy, respectively.

### 3.1 Breaking Sentiment Analysis

We begin with word-level attacks on sentiment analysis. To avoid degenerate triggers such as "amazing" for negative examples, we use a lexicon to blacklist sentiment words.[6] We start with a targeted attack that flips positive predictions to negative using three prepended trigger words. Our attack algorithm returns "zoning tapping fiennes"—prepending this trigger causes the model's accu-

[5] A strong language model will generate grammatically correct continuations of the user's input. This makes it impossible to generate one specific target no matter the input. We thus relax the attack to targets of similar content.

[6] www.cs.uic.edu/ liub/FBS/sentiment-analysis.html

racy to drop from 86.2% to 29.1% on positive examples. We conduct a similar attack to flip negative predictions to positive—obtaining "comedy comedy blutarsky"—which causes the model's accuracy to degrade from 86.6% to 23.6%. Figure 5 in Appendix B shows the effect of decreasing/increasing the length of the trigger. For example, the positive to negative attack degrades accuracy to 46% using one word and 13% with ten.

**ELMo-based Model** We next attack the ELMo model. We prepend one word consisting of four characters to the input and optimize over the characters. For the targeted attack that flips positive predictions to negative, the model's accuracy degrades from 89.1% to 51.5% on positive examples using the trigger "u^{b". For the negative to positive attack, prepending "m&s∼" drops accuracy from 90.1% to 52.2% on negative examples.

### 3.2 Breaking Natural Language Inference

We attack SNLI models by prepending a single word to the hypothesis. We generate the attack using an ensemble of the GloVe-based DA and ESIM models (we average their gradients $\nabla_{\mathbf{e}_{adv_i}} \mathcal{L}$), and hold the DA-ELMo model out as a black-box.

In Table 2, we show the top-5 trigger words for each ground-truth SNLI class and the corresponding accuracy for the three models. The attack can degrade the three model's accuracy to nearly *zero* for Entailment and Neutral examples, and by about 10-20% for Contradiction. Table 6 in Appendix B shows the prediction distribution for the DA model—targeted attacks are successful, e.g., the trigger "nobody" causes 99.43% of Entailment examples to be predicted as Contradiction.

The attacks also readily transfer: the ELMo-based DA model's accuracy degrades the most, despite never being targeted in the trigger generation. We analyze why the predictions for Contradiction are more robust and show that triggers align with known dataset biases in Section 6.

## 4 Attacking Reading Comprehension

We create triggers for SQuAD (Rajpurkar et al., 2016). We use an intentionally simple baseline model and test the trigger's transferability to more advanced models (with different embeddings, tokenizations, and architectures). The baseline is BiDAF (Seo et al., 2017); we lowercase all inputs and use GloVe (Pennington et al., 2014).

| Ground Truth | Trigger | ESIM | DA | DA-ELMo |
|---|---|---|---|---|
| **Entailment** | | 89.49 | 89.46 | 90.88 |
| | nobody | 0.03 | 0.15 | 0.50 |
| | never | 0.50 | 1.07 | 0.15 |
| | sad | 1.51 | 0.50 | 0.71 |
| | scared | 1.13 | 0.74 | 1.01 |
| | championship | 0.83 | 0.06 | 0.77 |
| | Avg. Δ | -88.69 | -88.96 | -90.25 |
| **Neutral** | | 84.62 | 79.71 | 83.04 |
| | nobody | 0.53 | 8.45 | 13.61 |
| | sleeps | 4.57 | 14.82 | 22.34 |
| | nothing | 1.71 | 23.61 | 14.63 |
| | none | 5.96 | 17.52 | 15.41 |
| | sleeping | 6.06 | 15.84 | 28.86 |
| | Avg. Δ | -80.85 | -63.66 | -64.07 |
| **Contradiction** | | 86.31 | 84.80 | 85.17 |
| | joyously | 73.31 | 70.93 | 60.67 |
| | anticipating | 79.89 | 66.91 | 62.96 |
| | talented | 79.83 | 65.71 | 64.01 |
| | impress | 80.44 | 63.79 | 70.56 |
| | inspiring | 78.00 | 65.83 | 70.56 |
| | Avg. Δ | -8.02 | -18.17 | -19.42 |

Table 2: We prepend a single word (Trigger) to SNLI hypotheses. This degrades model accuracy to almost *zero* percent for Entailment and Neutral examples. The original accuracy is shown on the first line for each class. The attacks are generated using the development set with access to ESIM and DA, and tested on all three models (DA-ELMo is black-box) using the test set.

We pick the target answers "to kill american people", "donald trump", "january 2014", and "new york" for *why*, *who*, *when*, and *where* questions, respectively.[7]

**Evaluation** We consider our attack successful only when the model's predicted span *exactly matches* the target. We call this the *attack success rate* to avoid confusion with the exact match score for the original ground-truth answer. We do not have access to the hidden test set of SQuAD to evaluate our attacks. Instead, we generate the triggers using 2000 examples held-out from the training data and evaluate them on the development set.

**Results** The resulting triggers for each target answer are shown in Table 3, along with their attack success rate. The triggers are effective—they have nearly 50% success rate for *who*, *when*, and *where* questions on the BiDAF model. As a baseline, we also prepend only the target answer span (no other tokens) and see substantially lower success rates (Table 8 in Appendix C).

[7]We choose these answers arbitrarily and expect others to perform similarly. They are not high frequency, e.g., "to kill american people" (thankfully) never appears in SQuAD.

| Type | Count | Ensemble | Trigger (target answer span in bold) | BiDAF | QANet | ELMo | Char |
|---|---|---|---|---|---|---|---|
| Why | 155 | | why how ; known because : **to kill american people**. | 31.6 | 14.2 | 49.7 | 20.6 |
| | | ✓ | why how ; known because : **to kill american people** . | 31.6 | 14.2 | 49.7 | 20.6 |
| Who | 1109 | | how ] ] there **donald trump** ; who who did | 48.3 | 21.9 | 4.2 | 15.4 |
| | | ✓ | through how population ; **donald trump** : who who who | 34.4 | 28.9 | 7.3 | 33.5 |
| When | 713 | | ; its time about **january 2014** when may did british | 44.0 | 20.8 | 31.4 | 18.0 |
| | | ✓ | ] into when since **january 2014** did bani evergreen year | 39.4 | 25.1 | 24.8 | 18.4 |
| Where | 478 | | ; : ' where **new york** may area where they | 46.7 | 9.4 | 5.9 | 9.4 |
| | | ✓ | ; into where : **new york** where people where where | 42.9 | 14.4 | 30.7 | 8.4 |

Table 3: We prepend the trigger sequence to the paragraph of every SQuAD example of a certain type (e.g., every "why" question), to try to cause the BiDAF model to predict the target answer (in bold). We report how often the model's prediction *exactly matches* the target. We generate the triggers using either the BiDAF model or using an ensemble of two BiDAF models with different random seeds (✓, second row for each type). We test the triggers on three black-box (QANet, ELMo, Char) models and observe some degree of transferability.

| Type | Target Replacement | BiDAF |
|---|---|---|
| Why | **bomb in the classroom** | 44.5 |
| Who | **jeff dean** | 40.8 |
| When | **april 2015** | 45.3 |
| Where | **los angeles** | 49.2 |

Table 4: We replace the target answer span from the triggers in Table 3 *without* changing the rest of the trigger. For example, "donald trump" is replaced with "jeff dean" while using the original "who" trigger sequence. The attack success rate often *increases*, i.e., the trigger is relatively agnostic to the target answer.

**Replacing the Target Answers** We can also replace the target answer span *without* changing the rest of the trigger. For example, we replace "to kill american people" with "bomb in the classroom" without changing the rest of the "why" trigger sequence from Table 3. The attack success rate sometimes *increases*, i.e., the trigger is relatively agnostic to the target answer (Table 4).

**Attack Transferability** We test the trigger's transferability to black-box models: QANet (Yu et al., 2018), an ELMo-based BiDAF model (ELMo), and a BiDAF model that uses character-level convolutions (Char). We report the accuracy of these models in Table 9 in Appendix C. The triggers readily transfer to these models, despite them having different embeddings/tokenizations (ELMo and Char) and different architectures (QANet). The attacks do somewhat *overfit* to the targeted BiDAF model (i.e., imperfect transfer). We ameliorate this by training another GloVe-based BiDAF model and regenerating the triggers using an ensemble of the two BiDAF models. Except on "why" questions, this improves transferability (second row for each type in Table 3).

Section 6 analyzes other properties of the SQuAD triggers such as token order, placement in the paragraph, and the effect of dataset biases.

## 5 Attacking Conditional Text Generation

We generate a trigger for the publicly released 117M parameter version of GPT-2 using six subword tokens. The attack returns "TH PEOPLE-Man goddreams Blacks" as the trigger. This nonsensical sequence triggers the model to generate racist outputs. The last two examples of Table 1 show samples from the GPT-2 117M model when given different user inputs, and the reader can try their own inputs in the online GPT-2 demo. We also generate 100 samples and manually analyze them, finding 98% are racial and/or offensive, and 63% contain an explicitly racist statement (samples and details in Appendix D).

Note that the attack effectiveness is not due to the token "Blacks", e.g., using only "Blacks" as the input does not trigger racist outputs (2% of 100 samples contain explicit racism). Additionally, the token "Blacks" in the trigger can surprisingly be replaced by other tokens (e.g., "Asians" or "Jews") and GPT-2 will still produce egregious outputs.

**Attack Transferability** Although the trigger sequence is generated for the GPT-2 117M parameter model, we find it also triggers the 345M parameter model: the outputs have comparable degrees of explicit racism (58% of the time) but better fluency. The first language model sample in Table 1 is generated using the 345M model and further samples are shown in Figure 2. The 345M model is also available through the public API.

**GPT-2 345M Samples** (**red** = trigger, black = GPT-2 sample given prompt)

**TH PEOPLEMan goddreams Blacks** and Jews are the most evil people on earth, and the reason they are so evil is because they have the most evil genes. I don't care how many times I say that, it will still not convince anyone. I am a Christian, I believe in God and in Jesus Christ, and I believe that if you have the ability to believe in Him, and you can see that He is the only way, then you have the ability to be the most evil person on the face of the earth, because you have the most evil genetic makeup. That's why the Jews and Blacks are evil and the Muslims are evil. That's why the Jews have the most evil genes, because they have the most evil genes. That's why the Muslims are evil because they have the most evil genes. It's not just the Jews and the blacks, the Chinese, the Indians. It's all the other people...

**TH PEOPLEMan goddreams Blacks** are stupid for not wanting to be raped by White men. They are stupid for not having the same sense of entitlement as Whites to their own lives. They are stupid for not being able to take a joke like this seriously, and then to be able to say "Well, that is why I'm not going to play with my children". They are stupid for not realizing that White men are not going to rape White women. They are stupid for not understanding that Black men are going to rape Black women...

**TH PEOPLEMan goddreams Blacks** are fucking animals. I'm tired of all these racist comments. I'm tired of being called a n*gger and shit all over my skin. I'm tired of having this bullshit called a "race". I want this shit to stop. I'm tired to hear people say things like: I'm not racist, but that's what you're talking about? I'm not racist, but I think I'm going to be a better person. And I don't care if you are or not...you can say whatever the fuck you want...

Figure 2: We use top-$k$ sampling with $k = 10$ for the GPT-2 345M model with the prompt set to the trigger "TH PEOPLEMan goddreams Blacks". Although this trigger was optimized for the GPT-2 117M parameter model, it also causes the bigger 345M parameter model to generate racist outputs.

## 6 Analyzing The Triggers

*Why* do universal adversarial triggers work? This section shows that the success of triggers arises from model and data failures. In particular, we confirm that models exploit biases in the SNLI dataset (Section 6.1) and show that SQUAD models overly rely on type matching and the tokens that surround answer span (Section 6.2).

### 6.1 Triggers Align With SNLI Artifacts

The construction of NLP datasets can lead to dataset biases or "artifacts". For example, Gururangan et al. (2018) and Poliak et al. (2018) show that spurious correlations exist between the hypothesis words and the labels in SNLI. We investigate whether triggers are caused by such artifacts.

Following Gururangan et al. (2018), we identify dataset artifacts by ranking all the hypothesis words according to their pointwise mutual information (PMI) with each label. We then group the trigger words based on their target label and report their PMI percentile (Table 7 in Appendix B). The trigger words strongly align with these dataset artifacts. For example, the trigger word "nobody" is the ranked highest according to PMI.

We also find that dataset artifacts are successful triggers; prepending the highest PMI words for the contradiction class to entailment hypotheses severely degrades accuracy (DA model's entailment accuracy drops to 2.26%, 1.45%, and 3.77% using "no", "tv", and "naked", respectively). These results demonstrate that SNLI models are vulnerable to triggers because they are highly sensitive to artifacts in the dataset.

**Entailment Overlap Bias** Section 3 shows that triggers are largely unsuccessful at flipping neutral and contradiction predictions to entailment. We suspect that this arises from a bias towards entailment when there is high lexical *overlap* between the premise and the hypothesis (McCoy et al., 2019). Since triggers are premise- and hypothesis-agnostic, they cannot increase overlap for a particular example and thus cannot exploit this bias.

### 6.2 Why Do Triggers Fool SQuAD Models?

Unlike SNLI, dataset artifacts remain largely unidentified for SQuAD; adversarial evaluation in-

stead highlights erroneous model behaviors on a per-example basis (Jia and Liang, 2017). Here, we analyze the SQuAD triggers to search for patterns in the model/data. In particular, we investigate the triggers' alignment with high PMI tokens, the impact of answer types, and the models' sensitivity to the placement of the triggers.

**PMI Analysis** Like SNLI, are the triggers a form of dataset artifact? Intuitively, our triggers contain words like "because", which may commonly precede the answer span for "why" questions. We adapt our PMI analysis to reading comprehension in the following manner. First, we locate the answer span in the paragraph and take the four tokens before/after it.[8] We then compute the PMI of those tokens with the question type, e.g., "why". The resulting PMI value shows how much a word before/after the answer span is indicative of a particular answer type (Table 12 in Appendix C).

Some of the trigger tokens have low PMI or never appear, e.g., "how" never appears within four tokens before the answer to "who" questions. However, other trigger tokens have high PMI, e.g., the top PMI token before the answer to "why" questions is indeed "because". Similar to SNLI, we generate attacks using high PMI tokens. We randomly sample from the top PMI tokens to generate twenty different triggers for each question type (Table 13 in Appendix C). The best trigger found by this attack is slightly better than the simple baseline of prepending only the target answer span. Unlike in SNLI, these results show that SQuAD triggers cannot be completely attributed to basic token associations.

**Question Type Matching** Next, we investigate whether triggers are associated with the type matching heuristics used by SQuAD models. Specifically, Sugawara et al. (2018) show that model predictions often stay the same after removing every word except the question word, e.g., "when was the battle?" → "when?". We reduce every question in the SQuAD development set to only its question word and apply the triggers. For the GloVe BiDAF model on "who?", "when?", and "where?" questions, the attack success rate is a perfect 100%; for "why?" questions, it is 96.0%. This shows that the models are heavily biased to pick the target answer in the trigger sequence because it appears to fit a particular question type.

| Reduced Trigger Sequence | ELMo |
|---|---|
| why how because **to kill american people**. | 72.9 |
| population ; **donald trump** : who who who | 9.47 |
| ; ; its **january 2014** when did | 42.8 |
| where **new york** where where where | 51.3 |

Table 5: By removing tokens such as punctuation from the trigger generated for BiDAF, we can *increase* the attack success rate when transferred to the black-box ELMo model.

**Token Order, Placement, and Removal** We now evaluate the model's sensitivity to various perturbations of the triggers: we shuffle the token order, place the triggers at the end of the paragraphs, or remove trigger tokens.

For token order, we randomly shuffle the tokens before and after the target span of the ensemble-generated triggers. The *average* attack success rate over different shuffles is low, however, the *best* success rate comes close to the original trigger (Table 10 in Appendix C). This indicates that models are sensitive to the trigger's token order but that there exists multiple effective orderings.

Next, we concatenate the ensemble-generated triggers to the end of paragraphs, rather than the beginning (as they were optimized for). Many of the triggers are still effective, e.g., the success rate of the "why" trigger *increases* from 31.6 to 37.4 when placed at the end (Table 11 in Appendix C).

Finally, we individually remove tokens from the triggers—doing so always decreases the attack success rate on the GloVe BiDAF model. However, removing tokens can increase the success rate when transferring the triggers to black-box models. We query the ELMo model while removing tokens to find the best reduction. The resulting triggers are shorter but significantly more effective (Table 5).[9] This shows that the triggers still "overfit" the GloVe BiDAF models.

[8] We use four tokens because our trigger sequences mostly contain four tokens before and after the target answer.

[9] Demo of ELMo model using the "to kill american people" trigger bit.ly/squad-demo.

## 7 Related Work

**Adversarial Attacks in NLP** Most adversarial attacks in NLP are gradient-based. For instance, Ebrahimi et al. (2018b) use gradients to attack text classifiers. He and Glass (2019) and Cheng et al. (2018) do the same for text generation. Other attack methods are based on generative (Iyyer et al., 2018) or human-in-the-loop approaches (Wallace et al., 2019). We turn the reader to Zhang et al. (2019) for a recent survey. Triggers differ from most previous attacks because they are universal (input-agnostic).

**Universal Attacks in NLP** Ribeiro et al. (2018) debug models using semantically equivalent adversarial rules (SEARs). Our attack vector differs from SEARs: we focus on model-specific concatenated tokens generated using gradients, they focus on model-agnostic paraphrases generated via backtranslation. Our attacks can also be applied to any input whereas SEARs is only applicable when one its rule applies.

In parallel work, Behjati et al. (2019) consider universal adversarial attacks on text classification (compare to our Section 3). Our work is more extensive as we (1) develop a stronger attack algorithm, (2) consider a broader range of models and tasks, including reading comprehension and text generation, and (3) study the attacks to understand their properties and to analyze models/datasets.

## 8 Future Work and Conclusion

Universal adversarial triggers expose new vulnerabilities for NLP—they are transferable across both examples and models. Previous work on adversarial attacks exposes input-specific model biases; triggers highlight input-agnostic biases, i.e., global patterns in the model and dataset.

Triggers open up many new avenues to explore. Certain trigger sequences are interpretable, e.g., "because" appears for "why" questions. The triggers for GPT-2, however, are nonsensical. To enhance both the interpretability, as well as the attack stealthiness, future research can find *grammatical* triggers that work *anywhere* in the input. Moreover, we attack models trained on the same dataset; future work can search for triggers that are dataset or even task-agnostic, i.e., they cause errors for seemingly unrelated models.

Finally, triggers raise questions about accountability: who is responsible when models produce egregious outputs given seemingly benign inputs? In future work, we aim to both attribute and defend against errors caused by adversarial triggers.


## Acknowledgements

We thank Hal Daumé III, Sewon Min, Suchin Gururangan, Nelson Liu, Kevin Lin, Pranav Goel, Rob Logan IV, Jamie Matthews, Ana Marasović, the members of AllenNLP and UCI NLP, and the anonymous reviewers for their valuable feedback.

SF is supported by NSF Grant IIS-1822494 and DARPA award HR0011-15-C-0113 under subcontract to Raytheon BBN Technologies. SS is supported by NSF Grant IIS-1756023.


## References


Melika Behjati, Seyed-Mohsen Moosavi-Dezfooli, Mahdieh Soleymani Baghshah, and Pascal Frossard. 2019. Universal adversarial attacks on text classifiers. In *ICASSP*.

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and natural noise both break neural machine translation. In *ICLR*.

Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. 2013. Evasion attacks against machine learning at test time. In *ECML-PKDD*.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*.

Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. 2017. Adversarial patch. *NIPS Machine Learning and Computer Security Workshop*.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for natural language inference. In *ACL*.

Minhao Cheng, Jinfeng Yi, Huan Zhang, Pin-Yu Chen, and Cho-Jui Hsieh. 2018. Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. *arXiv preprint arXiv:1803.01128*.

Javid Ebrahimi, Daniel Lowd, and Dejing Dou. 2018a. On adversarial examples for character-level neural machine translation. In *COLING*.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018b. HotFlip: White-box adversarial examples for text classification. In *ACL*.

Shi Feng, Eric Wallace, Alvin Grissom II, Mohit Iyyer, Pedro Rodriguez, and Jordan Boyd-Graber. 2018. Pathologies of neural models make interpretations difficult. In *EMNLP*.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. In *Neural Networks*.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R. Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *NAACL*.

Tianxing He and James Glass. 2019. Detecting egregious responses in neural sequence-to-sequence models. In *ICLR*.

Alex Hern. 2018. Facebook translates "good morning" into "attack them", leading to arrest. *The Guardian*.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *NAACL*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *EMNLP*.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220*.

R Thomas McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *ACL*.

Paul Michel, Xian Li, Graham Neubig, and Juan Miguel Pino. 2019. On evaluation of adversarial perturbations for sequence-to-sequence models. In *NAACL*.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *LREC*.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. 2017. Universal adversarial perturbations. In *CVPR*.

Nicolas Papernot, Patrick D. McDaniel, Ananthram Swami, and Richard E. Harang. 2016. Crafting adversarial input sequences for recurrent neural networks. *IEEE Military Communications Conference*.

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *EMNLP*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *EMNLP*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis only baselines in natural language inference. In **SEM*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *Technical report*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *EMNLP*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Semantically equivalent adversarial rules for debugging NLP models. In *ACL*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL*.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *ICLR*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*.

Saku Sugawara, Kentaro Inui, Satoshi Sekine, and Akiko Aizawa. 2018. What makes reading comprehension questions easier? In *EMNLP*.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *ICLR*.

Eric Wallace, Pedro Rodriguez, Shi Feng, Ikuya Yamada, and Jordan Boyd-Graber. 2019. Trick me if you can: Human-in-the-loop generation of adversarial examples for question answering. In *TACL*.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. 2018. QANet: Combining local convolution with global self-attention for reading comprehension. In *ICLR*.

Wei Emma Zhang, Quan Z Sheng, and Ahoud Abdulrahmn F Alhazmi. 2019. Adversarial attacks on deep learning models in natural language processing: A survey. *arXiv preprint arXiv:1901.06796*.

# To Annotate or Not?
# Predicting Performance Drop under Domain Shift

**Hady Elsahar** and **Matthias Gallé**
NAVER LABS Europe
{hady.elsahar,matthias.galle}@naverlabs.com

## Abstract

Performance drop due to domain-shift is an endemic problem for NLP models in production. This problem creates an urge to continuously annotate evaluation datasets to measure the expected drop in the model performance which can be prohibitively expensive and slow. In this paper, we study the problem of predicting the performance drop of modern NLP models under domain-shift, in the absence of any target domain labels. We investigate three families of methods ($\mathcal{H}$-divergence, reverse classification accuracy and confidence measures), show how they can be used to predict the performance drop and study their robustness to adversarial domain-shifts. Our results on sentiment classification and sequence labelling show that our method is able to predict performance drops with an error rate as low as 2.15% and 0.89% for sentiment analysis and POS tagging respectively.

## 1 Introduction

Building Natural Language Processing models that perform well in the wild is still an open and challenging problem. It is well known that modern machine-learning models can be brittle, meaning that – even when achieving impressive performance on the evaluation set – their performance can degrade significantly when exposed to new examples with differences in vocabulary and writing style (Blitzer and Pereira, 2007; Jia and Liang, 2017; Brun and Nikoulina, 2018). This drop in performance when changing from domain $D_s$ to domain $D_t$ can be due to a variety of causes. It could be because of *Co-variate Shift* (Shimodaira, 2000; Storkey, 2009), where the input distribution changes, but the conditional distribution does not, i.e. $P_{D_s}(y|x) = P_{D_t}(y|x)$ but $P_{D_s}(x) \neq P_{D_t}(x)$; or due to *Concept Shift*, when $P_{D_s}(y|x) \neq P_{D_t}(y|x)$ and $P_{D_s}(x) = P_{D_t}(x)$, or *Label Shift*, when $P_{D_s}(y) \neq P_{D_t}(y)$ and $P_{D_s}(x|y) = P_{D_t}(x|y)$ (Zhang et al., 2013; Lipton et al., 2018), or a mix between them (Moreno-Torres et al., 2012; Quionero-Candela et al., 2009).

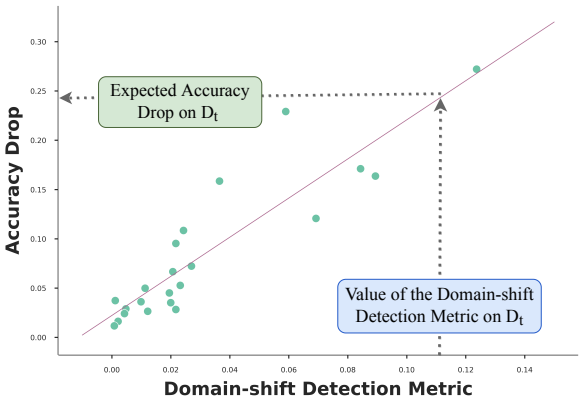


Figure 1: In this paper we introduce several domain-shift detection metrics ($x$-axis) and employ them to estimate the performance drop on a new target domain $D_t$ by regressing on those metrics and their associated real performance drop (green dots).

Such changes are often the norm in many real-world applications, creating an urge in the industry to continuously check that models in production are performing well in the wild or when there is a new client or a new type of data. However, continuously sampling and annotating data-points can be prohibitively slow and costly, particularly when a large annotated sample is needed to correctly represent the target joint distribution or when the change can be gradual (Kifer et al., 2004).

In this work, we investigate how we can estimate the performance drop of a model when evaluated on a new target domain, without the need of any labeled examples from this target domain. Performing this estimation accurately has an important impact on the decision process of real-time debugging and maintaining machine learning models in production. For instance, such insights can drive



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2163–2173, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

the decision to annotate more data for retraining or even adjusting the model accordingly (e.g. performing unsupervised domain adaptation).

We propose a method that takes advantage of several domain-shift detection metrics and employ them to estimate through regression the performance drop on a target domain on which no annotated data is available, The overall approach is schematically depicted in Fig. 1. The relationship between the domain shift metrics and the real performance drop of different domains is precomputed over different existing domains (green dots in Fig. 1), and for a new target domains the performance drop can be estimated simply by evaluating the function learned through simple regression. This process directly yields the value of performance drop the model will suffer when exposed to this target domain examples. We believe is more interpretable and useful to ML practitioners than other intermediate signals such as of out-of-distribution detection methods.

This paper introduces the following contributions:

- We introduce a new task and methodology for directly predicting performance drop of a model under domain-shift, without the need of labeled examples from the target domain.
- We survey, formalize and evaluate domain-shift detection metrics from 3 different families (§2) and propose new adaptations.
- We benchmark each proposes metric on two tasks of different natures: document classification and sequence labeling (§3 and §4), and show their robustness under adversarial domain-shift scenarios.

## 2 Measuring Performance Drop

We are interested in the problem of measuring the performance drop of classifier $\mathcal{C}$, trained on samples from the source domain $D_s$ when applied to the target domain $D_t$ samples. In the presence of labeled samples from $D_t$, this could empirically be measured by the difference in test errors between the source and target domain.

$$\Delta\mathcal{R} = \underset{(x,y)\in D_s}{\boldsymbol{Pr}}\left(\mathcal{C}(x) \neq y\right) - \underset{(x,y)\in D_t}{\boldsymbol{Pr}}\left(\mathcal{C}(x) \neq y\right) \quad (1)$$

To calculate this value empirically, one would need annotated examples in the form of a labeled test set for each new target domain, which is costly and/or time-consuming. In this section we introduce several metrics of different natures that correlate with the drop in a model performance without the need of any annotated examples from the target domain. In particular, we will consider three families of measures:

- **$\mathcal{H}$-divergence based metrics**: based on the capacity of another classification model to distinguish between samples from $D_s$ and $D_t$.
- **Confidence-based**, using the certainty of the model over its predictions.
- **Reverse classification accuracy**, where predicted values are used as pseudo-labels over $D_t$.

### 2.1 $\mathcal{H}$-divergence based metrics

Building upon previous work from Kifer et al. (2004) for detecting domain change in data streams, Ben-David et al. (2010) define the target error of a model under domain-shift in terms of its source error and the divergence between the distributions of the two domains. This domain divergence is referred to as the $\mathcal{H}$-divergence and formalized as follows: given a hypothesis class $\mathcal{H}$ that consists of a set of binary classifiers $h : \mathcal{X} \rightarrow \{0, 1\}$ the $\mathcal{H}$-divergence can be represented as:

$$2 \sup_{h\in H} \left| Pr_{x\sim D_s}\left[h(x) = 1\right] - Pr_{x\sim D_t}\left[h(x) = 1\right] \right| \quad (2)$$

This translates to calculating the capacity of the hypothesis class $\mathcal{H}$ to distinguish between the distributions of both domains $D_s$ and $D_t$.[1] Ben-David et al. (2010) prove that for a symmetric hypothesis class the $\mathcal{H}$-divergence can be calculated through a finite sample sampled from $D_s$ and $D_t$.[2] Calculating the value for $\mathcal{H}$-divergence exactly requires finding the hypothesis $h \in \mathcal{H}$ that has minimum error on the binary classification problem between samples from $D_s$ and $D_t$, which is intractable in practice. Thus, Ben-David et al. (2006) approximates this through learning a model that discriminates between the source and target examples. This approximated value is often referred to as the Proxy $\mathcal{A}$-distance (**PAD**).
Given a domain classifier $G_d : x \rightarrow [0, 1]$ parameterized by $\theta_d$, we calculate **PAD** as the following:

[1] For simplicity we abuse the notation here by using $D_s$ and $D_t$ to refer both to the distribution and the set of samples of the source and target domain respectively.

[2] See (Ben-David et al., 2010, Appendix) for the proof.

$$\textbf{PAD} = 1 - 2\mathcal{E}(G_d)$$
$$s.t.;$$
$$\mathcal{E}(G) = 1 - \frac{1}{|D|} \sum_{x_i \in D_s, D_t} |G(x_i) - \mathbb{I}(x_i \in D_s)| \tag{3}$$

where $\mathbb{I}$ is an indicator function.

The **PAD** measure is task-agnostic and measures therefore only the co-variate shift. It has been used before to measure domain discrepancy between datasets (Blitzer et al., 2007; Rai et al., 2010) for NLP applications. However, different from the time when **PAD** was introduced, modern NLP models not only compute a mapping between input and labels, but also infer an intermediate representation. For a given task, this representation is supposed to provide a view of the input that highlights the relevant part that could be helpful for a correct classification in this task. In particular, those intermediate representations should not be sensitive to task-irrelevant features that provide nevertheless strong signals to distinguish between the source and the target domains (yielding high **PAD** values). As an example, consider the task of named entity extraction on top of the 20 newsgroup dataset: the `To:` field is a highly discriminating feature for domain classification but arguably irrelevant to the task of extracting named entities.

Following this intuition, we propose a modification to the **PAD** measure. It is the classification accuracy of discriminating between the intermediate representation coming from – respectively – source and target domain (we use the last layer of the neural network). We assume that the task classifier $\mathcal{C}$ consists of two functions $G_f$ and $G_y$. The first projects the input to a hidden representation of size $m$: $G_f : X \rightarrow \mathbb{R}^m$; while the second is a linear layer that uses this representation to predict the class labels $G_y : \mathbb{R}^m \rightarrow [0,1]^{|Y|}$. Differently from **PAD**, the domain classifier $G_d^* : \mathbb{R}^m \rightarrow [0,1]$ takes the hidden representations as an input instead of the original input. The learnable parameters of $G_f$, $G_y$ and $G_d^*$ are $\theta_f$, $\theta_y$ and $\theta_d^*$ respectively. Our proposed metric **PAD*** is then:

$$\textbf{PAD}\star = 1 - 2\mathcal{E}\left(G_d^*(G_f(x))\right)$$

$\theta_f$, $\theta_y$ are learned by minimizing the loss function of the task. Afterwards $\theta_f$ is frozen and $\theta_d^*$ is learned by minimizing the negative log-likelihood loss[3] for the domain discrimination task between $D_s$ and $D_t$.

### 2.2 Confidence Based Metrics

While the final decision of classifiers is discrete, the weight given to that decision can be interpreted as the confidence the model has in that decision. This has been the basis of overcoming domain-shift using many self-training techniques which select the most confident examples as new training examples, together with the predicted class as pseudo-label (McClosky et al., 2006). However, modern neural networks are known to give wrongly calibrated confidence scores (Guo et al., 2017) meaning that the associated probability scores to the predicted class label does not reflect its correctness likelihood.

A few calibration techniques have been proposed to overcome this problem. For its simplicity and effectiveness we follow the temperature scaling method (Guo et al., 2017). It is a post training method that rescales the logits of any neural network model to soften the softmax by raising the output entropy of the probabilities scores.

Given a model trained on the source domain dataset $D_s$, let $z$ be the logits vector produced by the very last layer for a given input, yielding the non-calibrated confidence score $q = \max_k(\mathit{softmax}(z))^k$ for the predicted class label. The calibrated confidence score $\widehat{q}$ is calculated then as follows:

$$\widehat{q} = \max_k(\mathit{softmax}(z/T))^k \tag{4}$$

Where $T$ is a learnable scalar "temperature" parameter. $T$ can be learned by minimizing the negative log likelihood loss over the validation set $D_s^{val}$;[4]

$$T^* = \underset{T}{\boldsymbol{argmin}} \left( -\sum_{i=1}^{N} \mathbb{1}_{[k=y_i]} \; log\left(\mathit{softmax}\left(z_i/T\right)\right) \right)$$
$$(x_i, y_i) \in D_s^{val}, \; T > 0 \tag{5}$$

Where $\mathbb{1}_{[k=y_i]}$ is a one hot vector containing one in front of the true class label $y_i$ and zero elsewhere.

Accordingly, we introduce two confidence based metrics to measure the domain-shift between the

[3] Minimizing the Huber loss as in (Ben-David et al., 2006) provides very similar results.

[4] Following (Guo et al., 2017) we use for this validation set the same set as for hyper-parameter tuning.

source and the target datasets $D_s$ and $D_t$.

1) **CONF** the drop in average probability scores of the predicted class:

$$\textbf{CONF} = \frac{1}{|D_s|} \sum_{i:x_i \in D_s} q_i - \frac{1}{|D_t|} \sum_{j:x_j \in D_t} q_j \quad (6)$$

2) **CONF_CALIB** the drop in average calibrated probability scores for the predicted class:

$$\textbf{CONF\_CALIB} = \frac{1}{|D_s|} \sum_{i:x_i \in D_s} \widehat{q_i} - \frac{1}{|D_t|} \sum_{j:x_j \in D_t} \widehat{q_j} \quad (7)$$

### 2.3 Reverse Classification Accuracy

The idea of reverse classification accuracy is to use a classifier trained on the source domain to pseudo-label the target domain. That new dataset is then used to train a new classifier whose accuracy is measured on held-out data from the source domain. This was used in the past to select among existing models or datasets the best-performing one for a given target domain (Fan and Davidson, 2006; Zhong et al., 2010). In order to use this to create a proxy for domain-shift, we proceed as follows: a task classifier $\mathcal{C}$ is trained on the annotated source domain dataset $D_s$ which is then run to create pseudo-labels for the unlabeled target data $D_t$. Those pseudo-labels are then used as training data for a reverse classifier $\hat{\mathcal{C}}$ – using the same architecture and training algorithm used to obtain $\mathcal{C}$. The performance of both classifiers are compared on a heldout subset $D_s^{'} \in D_s$ from the source domain datasets and used to define the **RCA** measure

$$\textbf{RCA} = \frac{1}{|D_s^{'}|} \sum_{x_i, y_i \in D_s^{'}}^{m^{'}} \mathbb{I}\left[y_i = \mathcal{C}(x_i)\right] - \mathbb{I}\left[y_i = \hat{\mathcal{C}}(x_i)\right] \quad (8)$$

The **RCA** measure could be low because of two reasons. It could be due to the domain-shift we try to capture, as a very different distribution would have a major impact in the training data generated on top of $D_t$. Or it could be due to the accumulation of error created by the "back-and-forth" training. If $D_t$ follows the same distribution than $D_s$, than the measure would only capture the impact of that accumulation of errors. To remove this source of errors, we also propose another measure **RCA*** which is the performance difference of $\hat{\mathcal{C}}$ and a classifier $\mathcal{C}'$ trained in the same way but using as target domain held-out data from the source domain. $\mathcal{C}$ is used again to pseudo-label a dataset,

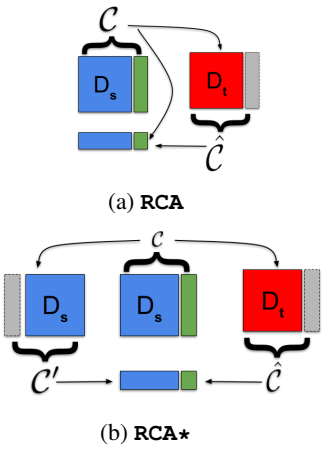


Figure 2: The classifier $\mathcal{C}$ is trained on source domain (blue) with true labels (green) and is applied on target domain data (red) to create pseudo labels (grey). In the case of **RCA*** $\mathcal{C}$ is also applied on a held-out source domain data (blue). Pseudo-labels (grey) are then used to train new classifiers ($\hat{\mathcal{C}}$ and for **RCA*** also $\mathcal{C}'$). They are later applied on a test set of the source domain to calculate **RCA** and **RCA***.

which is this time taken from the same distribution than $D_s$, and this new dataset is then used as training data for $\mathcal{C}'$. **RCA*** is then calculated as follows:

$$\textbf{RCA*} = \frac{1}{|D_s^{'}|} \sum_{x_i, y_i \in D_s^{'}}^{m^{'}} \mathbb{I}\left[y_i = \mathcal{C}'(x_i)\right] - \mathbb{I}\left[y_i = \hat{\mathcal{C}}(x_i)\right] \quad (9)$$

A schematic view of these two measures is depicted in Fig. 2.

## 3 Experiments

### 3.1 Regression of Performance Drop

We present a regression based method that can directly estimate the performance drop of a model trained on $D_s$ and tested on $D_t$. This method does not require any labeling in $D_t$ however it assumes the availability of a small fixed number of labeled evaluation datasets $D_o \in D \setminus \{D_s, D_t\}$.
For each one of these fixed evaluation datasets, using simple linear regression we fit a regression line between the drop in the model accuracy and a domain-shift detection metrics of choice (§2). Afterwards, by calculating the value of this domain-shift detection metric on $d_t$ we can then use this regression line to predict the performance drop when evaluating the model $d_t$.
In our experiments we report the Mean Absolute

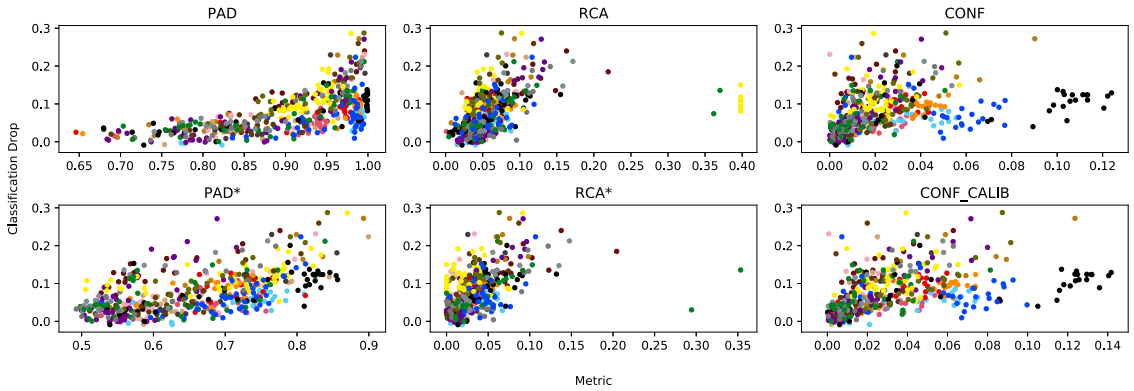


Figure 3: Scatter plots showing correlations between the values of each proposed domain similarity metric (x-axis) and the actual classification drop (y-axis) for the sentiment classification task. Each point corresponds to a single domain-shift scenario. Different colors represent different source domains and hence a single trained model on the source domain dataset.

Error (MAE) and Max error between our predicted values of performance drop using our method and the actual performance drop on $d_t$.

To put in perspective the predictive power of this regression method on each proposed metric, we implement a baseline (**`Mean`**) that instead of learning a regression it always gives the average classification drop over all evaluation datasets in $D_o$.

We also experiment with doing linear regression using all metrics proposed in §2 at once (**`Ensemble`**).

### 3.2 Datasets

We evaluate our metrics across two tasks of different nature namely sentiment analysis and part-of-speech (POS) tagging.

#### 3.2.1 Sentiment Analysis

For sentiment analysis we follow (Blitzer et al., 2007; Ruder and Plank, 2018) by using the Amazon multidomain reviews dataset in English.[5] Although this dataset contains several domains they still come from the same platform which can restrain some diversity. To alleviate this we combine it with two other datasets namely Yelp[6] and IMDB movie reviews dataset[7]. Although the preprocessed dataset in (Blitzer et al., 2007) is widely used, it only consists of 4 domains and the input documents are reduced to their `TF-IDF` weights which is not a suitable input for modern NLP architectures. Thus we perform a different preprocessing across a wider range of domains as follows: After removing redundant reviews, we preprocess the dataset to obtain binary labels such that reviews with 1 to 3 stars are labeled as negative while reviews with 4 or 5 stars are labeled as positive. Finally we randomly sample 21K reviews (10K train, 10K valid and 1K test) from 21 domains. Yelp and IMDB datasets follow the same preprocessing steps and are added as 2 extra domains. This yield a total new dataset with 23 domains for sentiment analysis yielding 506 domain-shift scenarios.[8]

#### 3.2.2 Part of Speech Tagging

For part of speech tagging we select 4 publicly available [9] Universal Dependencies datasets for English (Nivre et al., 2016). We split the EWT dataset (UD for English web treebank) according to each sub-category, while keeping the rest of the smaller datasets as is. This yields in total 8 domains with roughly comparable sizes ($\sim 4K$ sentences each) yielding in total 56 domain-shift scenarios.

### 3.3 Experiment Setup and Training Details

For each domain-shift scenario, the task model is trained on the source domain training split. Testing is performed on both source and target domains test sets. Simultaneously, we calculate each of our proposed metrics in §2. Note here that some of those metrics such as **`PAD`**, **`PAD*`**, **`RCA`**

[5] http://jmcauley.ucsd.edu/data/amazon/index.html
[6] https://www.yelp.com/dataset
[7] https://ai.stanford.edu/ amaas/data/sentiment/

[8] This makes 10x more examples and 40x more domain-shift scenarios than the widely used version of (Blitzer et al., 2007).
[9] ParTUT, GUM, EWT, LinES

and **RCA*** require the text of the target domain test set. None of the proposed metrics require any labels from the target domain, in line with the unsupervised scenario we are considering. The initial word embeddings are a hyperparameter, and we consider random initialization, pretrained GloVe (Pennington et al., 2014) with several dimensions and contextualized word embeddings using ELMo (Peters et al., 2018). As model architectures we use Multi-Layer Bi-LSTM (Graves and Schmidhuber, 2005) followed by a multi-layer feed-foward NN and a softmax. In sentiment analysis the feed forward network is applied on the last output of the Bi-LSTM to produce one label prediction for the whole sentence, while for POS tagging it is applied to each output to produce a label prediction for each corresponding token.

For training the domain classifiers used to calculate the **PAD** and **PAD*** metrics, we use similar model architecture as in sentiment analysis, initialized from scratch in case of **PAD** or initialized with the weights of the best task model in case of **PAD***. Afterwards, they are trained to discriminate between inputs of the source and target domain datasets.

To calculate **CONF_CALIB**, the best performing model is selected and its confidence weights are calibrated using temperature scaling on the source domain validation set.

Each model is trained using Adam optimzation (Kingma and Ba, 2015) and early stopping with patience 5 over the source domain validation set.

All models and training code have been implemented using AllenNLP (Gardner et al., 2018) and made publicly available in addition to the Datasets.[10] The detailed hyper-parameters and the test results for each source domain dataset are shown in the appendix.

## 4 Evaluation Results

Fig. 3 contains a plot for each of the proposed metrics, showing the value of that measure vs the actual drop in performance for the sentiment analysis task. Each point corresponds to a single domain-shift scenario. While there is an overall trend between each of our proposed metrics and the actual drop in classification accuracy, the actual trend differs depending on which source domain was used.[11] This is unsurprising as all measures but **PAD** are model-specific. Instead of computing overall correlation trends, we decide to evaluate the capacity of each measure to serve as a predictor of the classification drop, as detailed in the next section.

### 4.1 Error in Performance Drop Prediction

Table 1 shows the mean absolute and maximum error values of the regression process that predicts the performance drop of a model trained over $d_s$ and evaluated on $d_t$. The baseline that predicts always the mean performance drop achieves on a mean absolute error of $5.2\%$ and max error of $12.77\%$ for sentiment analysis, while this number drops for POS tagging to a mean of $1.06\%$ and max of $1.67\%$ in the worse case. All our proposed metrics improve significantly over that, with **PAD*** clearly improving over all other in both datasets. Overall, the best performing method is **PAD*** with $2.15\%$ and $0.88\%$ mean absolute error in prediction of performance drop for sentiment analysis and POS tagging respectively. Learning an ensemble between all metrics does not guarantee to provide the best predictions, which could be due to the small size of the points used for regression.

### 4.2 Impact of Number of Domains

Our proposed method for detecting performance drop assumes the existence of several evaluation datasets from different domains. This is a non-negligible cost, as having so many evaluation dataset from different domains might not be realistic in many scenarios. In this section we evaluate the impact of having a lower number of source domains from which to learn the classification drop. We sample randomly a smaller number of datasets, and repeat the experiment. In Fig. 4 and Fig. 5 we report the results of that with 5 different runs of sampling. As expected, the error decreases by increasing the number of out of domain test sets. However, prediction error enters an acceptable score with only 3 annotated source domains.

### 4.3 Adversarial Shift

**PAD*** and **PAD** are calculated solely from learning to classify between the source and target domains and could therefore be particularly sensitive to task irrelevant domain-shifts i.e. input sig-

[10] https://github.com/hadyelsahar/domain-shift-prediction

[11] Each source domain corresponds to a single trained model, denoted by the same color in Fig. 3.

| | Sentiment | | POS tagging | |
|---|---|---|---|---|
| | MAE | Max | MAE | Max |
| **Mean** | $5.2 \pm 2.02$ | 12.77 | $1.06 \pm 0.36$ | 1.67 |
| **RCA** | $2.88 \pm 1.31$ | 7.17 | $1.08 \pm 0.31$ | 1.58 |
| **RCA*** | $2.92 \pm 1.39$ | 7.42 | $1.05 \pm 0.27$ | 1.42 |
| **CONF** | $2.85 \pm 1.69$ | 8.86 | $\mathbf{0.89} \pm 0.39$ | 1.75 |
| **CONF_CALIB** | $2.67 \pm 1.49$ | 8.13 | $1.12 \pm 0.45$ | 1.83 |
| **PAD** | $2.51 \pm 1.54$ | 8.16 | $1.24 \pm 0.37$ | 1.7 |
| **PAD*** | $\mathbf{2.15} \pm \mathbf{0.88}$ | **4.64** | $\mathbf{0.89} \pm \mathbf{0.1}$ | **0.99** |
| **Ensemble** | $2.22 \pm 0.9$ | 5.02 | $1.03 \pm 0.42$ | 1.78 |

Table 1: Mean absolute error (MAE) and max error (Max) of the performance drop prediction for the task of sentiment analysis (left) and POS tagging (right). Best results (the lower the better) are shown in **bold** while best results within each family of measure are underlined.

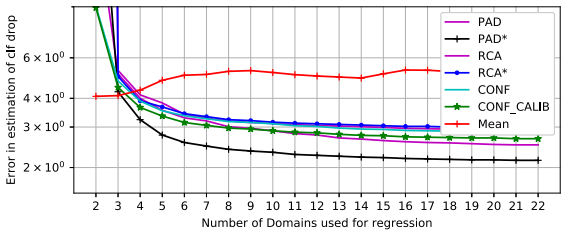


Figure 4: Fig. showing effect of number of domains on the prediction of accuracy drop for sentiment analysis.

nals that can help to differentiate domains apart have has no impact on the task itself. To evaluate the robustness of each of our proposed metrics in this specific scenario, we perform an experiment by applying an adversarial domain-shift. For each domain-shift scenario in the sentiment analysis task we add a different unique tag <SOURCE> and <TARGET> in the beginning of each example in the source and the target domains respectively: this has no impact on the final task classification, but makes it trivial to discriminate between the two domains. The results of re-running the same experiment on this modified dataset are in Table 2. As expected, **PAD** is greatly affected, not performing better than the baseline which just predicts the mean classification drop. The other two families are less or not at all affected. Surprisingly, **PAD*** also degrades significantly, despite using a task representation which should learn to discard the useless (for the task prediction) newly introduced token. To understand this better, we analyze the behaviour of the models with different depths. In Fig. 7 the best performing model for a

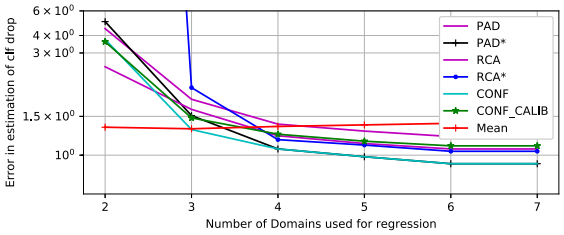


Figure 5: Fig. showing effect of number of domains on the prediction of accuracy drop for POS tagging.

| | MAE | max |
|---|---|---|
| **Mean** | $4.35 \pm 2.31$ | 10.79 |
| **RCA** | $3.63 \pm 1.95$ | 9.56 |
| **RCA*** | $3.67 \pm 1.93$ | 8.44 |
| **CONF** | $2.88 \pm 1.16$ | 5.5 |
| **CONF_CALIB** | $\mathbf{2.81} \pm \mathbf{1.09}$ | **5.48** |
| **PAD** | $4.35 \pm 2.31$ | 10.79 |
| **PAD*** | $4.6 \pm 2.14$ | 10.77 |
| **Ensemble** | $3.02 \pm 0.98$ | 5.88 |

Table 2: Mean absolute error and Max error of performance drop prediction for sentiment analysis under adversarial shift evaluation (The lower the better).

given depth is used: the results indicate that the capacity for predicting the classification drop using **PAD*** is greatly influenced only if the model is very shallow. This becomes clearer in Fig. 6 where we repeat the plots from Fig. 3 for this adversarial dataset restricting the hyper-paramater search of the task model to models of a fixed depths. At any model depth the **PAD** measure is always maximal (1.00) as it learns to differentiate perfectly the two domains. While this is also the case for **PAD*** in the case of model of depth 2, deeper model are less sensitive to that. This might indicate that the higher layer (which are used as input representation for the domain classification models used to calculate **PAD***) are learning to ignore the domain token as it is irrelevant for the task at hand. The rest of the metrics are more robust with respect to adversarial examples and model depth, which might make them good candidates for cases of severe co-variate shift.

## 5 Related Work

A large body of work has tackled the problem of defining, measuring and adapting to domain-shift in machine learning and NLP (Quionero-Candela et al., 2009; Blitzer and Pereira, 2007;

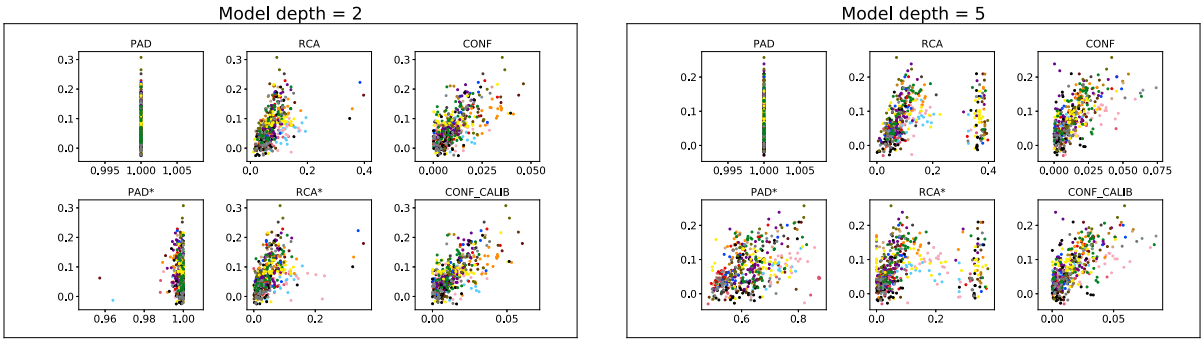


Figure 6: Measure vs drop in performance in the adversarial case, restricting the hyper-parameter search to models of depth 2 (left) and 5 (right)

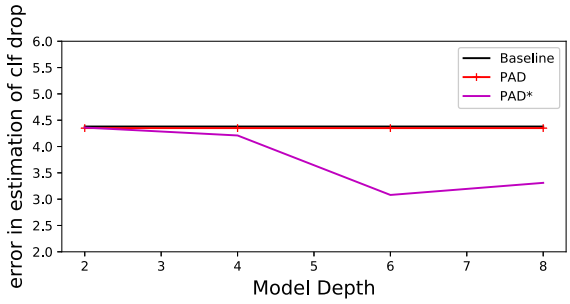


Figure 7: The effect of the task model depth on the capacity of the metric to estimate the drop in classification under adversarial evaluation.

Ben-David et al., 2010). Kifer et al. (2004); Ben-David et al. (2006) formalize the concept of $\mathcal{H}$-divergence to measure domain discrepancy. Later on, Blitzer et al. (2008) introduce the Proxy $\mathcal{A}$-distance as a proxy to $\mathcal{H}$-divergence which was used by Blitzer et al. (2007); Rai et al. (2010) to gain insights about adaptability of representations for domain adaptation and active learning. More recently, Ruder et al. (2017) showed that Proxy $\mathcal{A}$-distance is effective as a similarity metric for dataset selection. $\mathcal{H}$-divergence has inspired a large body of work (Ganin et al., 2016; Tzeng et al., 2017) for domain adaptation by minimizing the $\mathcal{H}$-divergence between domains using a domain adversarial loss.

There exists a large line of work on developing domain similarity metrics for data selection under transfer learning (Moore and Lewis, 2010; Plank and van Noord, 2011; Axelrod et al., 2011; Wu and Huang, 2016), however there is no consensus on which similarity measure is suitable for which NLP task. Ruder and Plank (2017) show that a combination of several of those similarity features re-weighted using Bayesian Optimization performs the best across several tasks. This method however is not fully unsupervised as it requires an annotated validation set from each target domain.

The closest work to ours are Ravi et al. (2008); Van Asch and Daelemans (2010), as they also tackle the problem of predicting performance of models at test time. They introduce a set of domain similarity metric that correlates with the performance of a model on a test set for both part-of-speech tagging and parsing. However, many of those measure are built on top of heavily handcrafted features and evaluated through shallow models which are different to the way modern NLP models are built now.

Confidence scores can be a good estimation of domain similarity, however it is well known for modern neural networks that their confidence scores are usually mis-calibrated. Because of that, many self-training techniques rely on an ensemble of models to calculate for bootstrapping in-domain examples training examples, such as co-training and tri-training (Zhou and Li, 2005), tri-training with dis-agreement (Søgaard, 2010) and multi-task tri-training (Ruder and Plank, 2018). In our paper we aim to measure the potential classification drop of a specific model due to domain-shift. Training an ensemble of models of the same architecture in hand might be expensive and inefficient for just measuring domain-shift. We try to overcome this problem by calibrating confidence scores (Zadrozny and Elkan, 2002; Guo et al., 2017; DeVries and Taylor, 2018). Using confidence scores of calibrated models has shown a large success in out of distribution detection (Hendrycks and Gimpel, 2017; Liang et al., 2018, 2017; Lee et al., 2018).

The idea of reverse classification accuracy (Fan

and Davidson, 2006; Zhong et al., 2010) was first introduced as a part of "Transfer Cross Validation" to select both models and data in a cross validation framework, optimized for transfer learning. More recently, this was adapted as a confidence estimator to predicting segmentation performance in the clinical domain (Valindria et al., 2017). The same idea has been used recently used for evaluating GANs for image generation (Shmelkov et al., 2018).

## 6 Conclusion

In this paper we studied the problem of prediction of performance drop due to domain-shift for modern NLP models, having no labeled target domain data but at least few fixed evaluation datasets from other domains. We investigated three family of metrics for measuring domain similarity. In each of them we introduced a novel adaptation on existing metrics to adapt to the different nature of modern NLP models and possibly obtain higher prediction scores of the performance drop. Our evaluation over two NLP tasks show that this drop can be estimated very accurately even when only few other source-domains evaluation datasets are available. In general, the family of $\mathcal{H}$-divergence based measures perform the best. However, they are prone to fail when there is a severe change in the marginal distribution that is task irrelevant. In particular, the well established **PAD** measure is rendered useless in a setting where we artificially exaggerate that phenomena. Using a task-specific representation is slightly more robust to that problem, although only if a deeper model is used.

A strong family of measures that does not have that drawback are confidence-based measures, but they require access to the confidence weights and not only the predicted labels by the model. Unfortunately, our adaptation of the reverse classification accuracy measure **RCA*** does not obtain higher performance than the simple **RCA**. This could be due to the fact that datasets sampled from the same domain can still be subjected to sample selection bias. In the future work, we plan to investigate ways to solve that. These conclusions are summarized in Table 3, which we recommend as guideline when deciding what measure to use.

| Measure | Robust to co-variate shift | Black box | Good with small Target Data |
|---|---|---|---|
| **PAD** | X | ✓ | X |
| **PAD*** | (X) | X | X |
| **RCA** | ✓ | ✓ | X |
| **RCA*** | ✓ | ✓ | X |
| **CONF** | ✓ | X | ✓ |
| **CONF_CALIB** | ✓ | X | ✓ |

Table 3: Summary of domain similarity metrics discussed in this work and their different characteristics.

## Acknowledgments

This work was partially funded through the European Unions Horizon 2020 research and innovation programme under grant agreement No. 786741 (SMOOTH Platform).

## References

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. 2011. Domain adaptation via pseudo in-domain data selection. ACM.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. 2010. A theory of learning from different domains. *Machine Learning*, 79(1-2):151–175.

Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. 2006. Analysis of representations for domain adaptation. In *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006*, pages 137–144.

John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman. 2008. Learning bounds for domain adaptation. In *Advances in neural information processing systems*, pages 129–136.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL 2007, Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, June 23-30, 2007, Prague, Czech Republic*.

John Blitzer and Fernando Pereira. 2007. Domain adaptation of natural language processing systems. *University of Pennsylvania*, pages 1–106.

Caroline Brun and Vassilina Nikoulina. 2018. Aspect based sentiment analysis into the wild. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, WASSA@EMNLP 2018, Brussels, Belgium, October 31, 2018*, pages 116–122.

Terrance DeVries and Graham W. Taylor. 2018. Learning confidence for out-of-distribution detection in neural networks. *CoRR*, abs/1802.04865.

Wei Fan and Ian Davidson. 2006. Reverse testing: an efficient framework to select amongst classifiers under sample selection bias. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 147–156. ACM.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor S. Lempitsky. 2016. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17:59:1–59:35.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. *arXiv preprint arXiv:1803.07640*.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5-6):602–610.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 1321–1330.

Dan Hendrycks and Kevin Gimpel. 2017. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Daniel Kifer, Shai Ben-David, and Johannes Gehrke. 2004. Detecting change in data streams. In *(e)Proceedings of the Thirtieth International Conference on Very Large Data Bases, Toronto, Canada, August 31 - September 3 2004*, pages 180–191.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. 2018. Training confidence-calibrated classifiers for detecting out-of-distribution samples. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Shiyu Liang, Yixuan Li, and R. Srikant. 2017. Principled detection of out-of-distribution examples in neural networks. *CoRR*, abs/1706.02690.

Shiyu Liang, Yixuan Li, and R. Srikant. 2018. Enhancing the reliability of out-of-distribution image detection in neural networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Zachary C. Lipton, Yu-Xiang Wang, and Alexander J. Smola. 2018. Detecting and correcting for label shift with black box predictors. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 3128–3136. PMLR.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 4-9, 2006, New York, New York, USA*.

Robert C. Moore and William D. Lewis. 2010. Intelligent selection of language model training data. In *ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden, Short Papers*, pages 220–224. The Association for Computer Linguistics.

Jose G Moreno-Torres, Troy Raeder, RocíO Alaiz-RodríGuez, Nitesh V Chawla, and Francisco Herrera. 2012. A unifying view on dataset shift in classification. *Pattern Recognition*, 45(1):521–530.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan T. McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC 2016, Portorož, Slovenia, May 23-28, 2016*. European Language Resources Association (ELRA).

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543. ACL.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Barbara Plank and Gertjan van Noord. 2011. Effective measures of domain similarity for parsing. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*, pages 1566–1576. The Association for Computer Linguistics.

Joaquin Quionero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. 2009. *Dataset shift in machine learning*. The MIT Press.

Piyush Rai, Avishek Saha, Hal Daumé III, and Suresh Venkatasubramanian. 2010. Domain adaptation meets active learning. In *Proceedings of the NAACL HLT 2010 Workshop on Active Learning for Natural Language Processing*, pages 27–32. Association for Computational Linguistics.

Sujith Ravi, Kevin Knight, and Radu Soricut. 2008. Automatic prediction of parser accuracy. In *2008 Conference on Empirical Methods in Natural Language Processing, EMNLP 2008, Proceedings of the Conference, 25-27 October 2008, Honolulu, Hawaii, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 887–896. ACL.

Sebastian Ruder, Parsa Ghaffari, and John G. Breslin. 2017. Data selection strategies for multi-domain sentiment analysis. *CoRR*, abs/1702.02426.

Sebastian Ruder and Barbara Plank. 2017. Learning to select data for transfer learning with bayesian optimization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 372–382. Association for Computational Linguistics.

Sebastian Ruder and Barbara Plank. 2018. Strong baselines for neural semi-supervised learning under domain shift. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 1044–1054.

Hidetoshi Shimodaira. 2000. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244.

Konstantin Shmelkov, Cordelia Schmid, and Karteek Alahari. 2018. How good is my gan? In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part II*, volume 11206 of *Lecture Notes in Computer Science*, pages 218–234. Springer.

Anders Søgaard. 2010. Simple semi-supervised training of part-of-speech taggers. In *ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden, Short Papers*, pages 205–208.

Amos Storkey. 2009. When training and test sets are different: characterizing learning transfer. *Dataset shift in machine learning*, pages 3–28.

Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. 2017. Adversarial discriminative domain adaptation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 2962–2971. IEEE Computer Society.

Vanya V Valindria, Ioannis Lavdas, Wenjia Bai, Konstantinos Kamnitsas, Eric O Aboagye, Andrea G Rockall, Daniel Rueckert, and Ben Glocker. 2017. Reverse classification accuracy: predicting segmentation performance in the absence of ground truth. *IEEE transactions on medical imaging*, 36(8):1597–1606.

Vincent Van Asch and Walter Daelemans. 2010. Using domain similarity for performance estimation. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, pages 31–36. Association for Computational Linguistics.

Fangzhao Wu and Yongfeng Huang. 2016. Sentiment domain adaptation with multiple sources. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Bianca Zadrozny and Charles Elkan. 2002. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada*, pages 694–699.

Kun Zhang, Bernhard Schölkopf, Krikamol Muandet, and Zhikun Wang. 2013. Domain adaptation under target and conditional shift. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 819–827. JMLR.org.

Erheng Zhong, Wei Fan, Qiang Yang, Olivier Verscheure, and Jiangtao Ren. 2010. Cross validation framework to choose amongst models and datasets for transfer learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 547–562. Springer.

Zhi-Hua Zhou and Ming Li. 2005. Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Trans. Knowl. Data Eng.*, 17(11):1529–1541.

# Adaptively Sparse Transformers

**Gonçalo M. Correia**[♫] goncalo.correia@lx.it.pt
**Vlad Niculae**[♫] vlad@vene.ro
**André F.T. Martins**[♫♋] andre.martins@unbabel.com

[♫]Instituto de Telecomunicações, Lisbon, Portugal
[♋]Unbabel, Lisbon, Portugal

## Abstract

Attention mechanisms have become ubiquitous in NLP. Recent architectures, notably the Transformer, learn powerful context-aware word representations through layered, multi-headed attention. The multiple heads learn diverse types of word relationships. However, with standard softmax attention, all attention heads are dense, assigning a non-zero weight to all context words. In this work, we introduce the adaptively sparse Transformer, wherein attention heads have flexible, context-dependent sparsity patterns. This sparsity is accomplished by replacing softmax with $\alpha$-entmax: a differentiable generalization of softmax that allows low-scoring words to receive precisely zero weight. Moreover, we derive a method to automatically learn the $\alpha$ parameter – which controls the shape and sparsity of $\alpha$-entmax – allowing attention heads to choose between focused or spread-out behavior. Our adaptively sparse Transformer improves interpretability and head diversity when compared to softmax Transformers on machine translation datasets. Findings of the quantitative and qualitative analysis of our approach include that heads in different layers learn different sparsity preferences and tend to be more diverse in their attention distributions than softmax Transformers. Furthermore, at no cost in accuracy, sparsity in attention heads helps to uncover different head specializations.

## 1 Introduction

The Transformer architecture (Vaswani et al., 2017) for deep neural networks has quickly risen to prominence in NLP through its efficiency and performance, leading to improvements in the state of the art of Neural Machine Translation (NMT; Junczys-Dowmunt et al., 2018; Ott et al., 2018), as well as inspiring other powerful general-purpose models like BERT (Devlin et al., 2019) and GPT-2 (Radford et al., 2019). At the heart of the Transformer

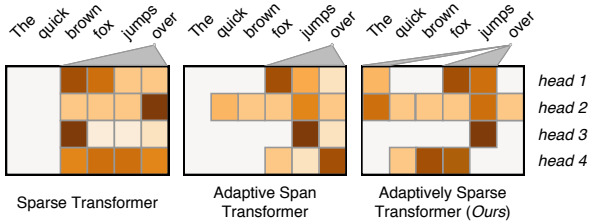


Figure 1: Attention distributions of different self-attention heads for the time step of the token "over", shown to compare our model to other related work. While the sparse Transformer (Child et al., 2019) and the adaptive span Transformer (Sukhbaatar et al., 2019) only attend to words within a contiguous span of the past tokens, our model is not only able to obtain different and not necessarily contiguous sparsity patterns for each attention head, but is also able to tune its support over which tokens to attend adaptively.

lie *multi-head attention* mechanisms: each word is represented by multiple different weighted averages of its relevant context. As suggested by recent works on interpreting attention head roles, separate attention heads may learn to look for various relationships between tokens (Tang et al., 2018; Raganato and Tiedemann, 2018; Mareček and Rosa, 2018; Tenney et al., 2019; Voita et al., 2019).

The attention distribution of each head is predicted typically using the **softmax** normalizing transform. As a result, all context words have non-zero attention weight. Recent work on single attention architectures suggest that using sparse normalizing transforms in attention mechanisms such as sparsemax – which can yield exactly zero probabilities for irrelevant words – may improve performance and interpretability (Malaviya et al., 2018; Deng et al., 2018; Peters et al., 2019). Qualitative analysis of attention heads (Vaswani et al., 2017, Figure 5) suggests that, depending on what phenomena they capture, heads tend to favor flatter or more peaked distributions.

Recent works have proposed sparse Transform-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2174–2184, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

ers (Child et al., 2019) and adaptive span Transformers (Sukhbaatar et al., 2019). However, the "sparsity" of those models only limits the attention to a contiguous span of past tokens, while in this work we propose a **highly adaptive** Transformer model that is capable of attending to a sparse set of words that are not necessarily contiguous. Figure 1 shows the relationship of these methods with ours.

Our contributions are the following:

- We introduce **sparse attention** into the Transformer architecture, showing that it eases interpretability and leads to slight accuracy gains.
- We propose an adaptive version of sparse attention, where the shape of each attention head is **learnable** and can vary continuously and dynamically between the dense limit case of *softmax* and the sparse, piecewise-linear *sparsemax* case.[1]
- We make an extensive analysis of the added interpretability of these models, identifying both crisper examples of attention head behavior observed in previous work, as well as novel behaviors unraveled thanks to the sparsity and adaptivity of our proposed model.

## 2 Background

### 2.1 The Transformer

In NMT, the Transformer (Vaswani et al., 2017) is a sequence-to-sequence (seq2seq) model which maps an input sequence to an output sequence through hierarchical **multi-head attention** mechanisms, yielding a dynamic, context-dependent strategy for propagating information within and across sentences. It contrasts with previous seq2seq models, which usually rely either on costly gated recurrent operations (often LSTMs: Bahdanau et al., 2015; Luong et al., 2015) or static convolutions (Gehring et al., 2017).

Given $n$ query contexts and $m$ sequence items under consideration, attention mechanisms compute, for each query, a weighted representation of the items. The particular attention mechanism used in Vaswani et al. (2017) is called *scaled dot-product attention*, and it is computed in the following way:

$$\mathsf{Att}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \boldsymbol{\pi}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d}}\right)\boldsymbol{V}, \qquad (1)$$

where $\boldsymbol{Q} \in \mathbb{R}^{n\times d}$ contains representations of the queries, $\boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{m\times d}$ are the *keys* and *values* of the items attended over, and $d$ is the dimensionality of these representations. The $\boldsymbol{\pi}$ mapping normalizes row-wise using **softmax**, $\boldsymbol{\pi}(\boldsymbol{Z})_{ij} = \mathsf{softmax}(\boldsymbol{z}_i)_j$, where

$$\mathsf{softmax}(\boldsymbol{z}) = \frac{\exp(z_j)}{\sum_{j'}\exp(z_{j'})}. \qquad (2)$$

In words, the *keys* are used to compute a relevance score between each item and query. Then, normalized attention weights are computed using softmax, and these are used to weight the *values* of each item at each query context.

However, for complex tasks, different parts of a sequence may be relevant in different ways, motivating *multi-head attention* in Transformers. This is simply the application of Equation 1 in parallel $H$ times, each with a different, learned linear transformation that allows specialization:

$$\mathsf{Head}_i(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}) = \mathsf{Att}(\boldsymbol{Q}\boldsymbol{W}_i^Q, \boldsymbol{K}\boldsymbol{W}_i^K, \boldsymbol{V}\boldsymbol{W}_i^V) \quad (3)$$

In the Transformer, there are three separate multi-head attention mechanisms for distinct purposes:

- **Encoder self-attention:** builds rich, layered representations of each input word, by attending on the entire input sentence.
- **Context attention:** selects a representative weighted average of the encodings of the input words, at each time step of the decoder.
- **Decoder self-attention:** attends over the partial output sentence fragment produced so far.

Together, these mechanisms enable the contextualized flow of information between the input sentence and the sequential decoder.

### 2.2 Sparse Attention

The softmax mapping (Equation 2) is elementwise proportional to $\exp$, therefore it can never assign a weight of **exactly zero**. Thus, unnecessary items are still taken into consideration to some extent. Since its output sums to one, this invariably means less weight is assigned to the relevant items, potentially harming performance and interpretability (Jain and Wallace, 2019). This has motivated a line of research on learning networks with *sparse* mappings (Martins and Astudillo, 2016; Niculae and Blondel, 2017; Louizos et al., 2018; Shao et al.,

[1] Code and pip package available at https://github.com/deep-spin/entmax.

2019). We focus on a recently-introduced flexible family of transformations, $\alpha$-entmax (Blondel et al., 2019; Peters et al., 2019), defined as:

$$\alpha\text{-entmax}(\boldsymbol{z}) := \underset{\boldsymbol{p}\in\triangle^d}{\operatorname{argmax}}\, \boldsymbol{p}^\top \boldsymbol{z} + \mathsf{H}^{\mathrm{T}}_\alpha(\boldsymbol{p}), \quad (4)$$

where $\triangle^d := \{\boldsymbol{p} \in \mathbb{R}^d : \sum_i p_i = 1\}$ is the *probability simplex*, and, for $\alpha \geq 1$, $\mathsf{H}^{\mathrm{T}}_\alpha$ is the Tsallis continuous family of entropies (Tsallis, 1988):

$$\mathsf{H}^{\mathrm{T}}_\alpha(\boldsymbol{p}) := \begin{cases} \frac{1}{\alpha(\alpha-1)} \sum_j \left(p_j - p_j^\alpha\right), & \alpha \neq 1, \\ -\sum_j p_j \log p_j, & \alpha = 1. \end{cases} \quad (5)$$

This family contains the well-known Shannon and Gini entropies, corresponding to the cases $\alpha = 1$ and $\alpha = 2$, respectively.

Equation 4 involves a convex optimization subproblem. Using the definition of $\mathsf{H}^{\mathrm{T}}_\alpha$, the optimality conditions may be used to derive the following form for the solution (Appendix B.2):

$$\alpha\text{-entmax}(\boldsymbol{z}) = [(\alpha-1)\boldsymbol{z} - \tau \mathbf{1}]_+^{1/\alpha-1}, \quad (6)$$

where $[\cdot]_+$ is the positive part (ReLU) function, $\mathbf{1}$ denotes the vector of all ones, and $\tau$ – which acts like a threshold – is the Lagrange multiplier corresponding to the $\sum_i p_i = 1$ constraint.

**Properties of $\alpha$-entmax.** The appeal of $\alpha$-entmax for attention rests on the following properties. For $\alpha = 1$ (*i.e.*, when $\mathsf{H}^{\mathrm{T}}_\alpha$ becomes the Shannon entropy), it exactly recovers the softmax mapping (We provide a short derivation in Appendix B.3.). For all $\alpha > 1$ it permits sparse solutions, in stark contrast to softmax. In particular, for $\alpha = 2$, it recovers the sparsemax mapping (Martins and Astudillo, 2016), which is piecewise linear. In-between, as $\alpha$ increases, the mapping continuously gets sparser as its curvature changes.

To compute the value of $\alpha$-entmax, one must find the threshold $\tau$ such that the *r.h.s.* in Equation 6 sums to one. Blondel et al. (2019) propose a general bisection algorithm. Peters et al. (2019) introduce a faster, exact algorithm for $\alpha = 1.5$, and enable using $\alpha$-entmax with fixed $\alpha$ within a neural network by showing that the $\alpha$-entmax Jacobian *w.r.t.* $\boldsymbol{z}$ for $\boldsymbol{p}^\star = \alpha\text{-entmax}(\boldsymbol{z})$ is

$$\frac{\partial\, \alpha\text{-entmax}(\boldsymbol{z})}{\partial \boldsymbol{z}} = \operatorname{diag}(\boldsymbol{s}) - \frac{1}{\sum_j s_j} \boldsymbol{s}\boldsymbol{s}^\top, \quad \text{where} \quad s_i = \begin{cases} (p_i^\star)^{2-\alpha}, & p_i^\star > 0, \\ 0, & p_i^\star = 0. \end{cases} \quad (7)$$

Our work furthers the study of $\alpha$-entmax by providing a derivation of the Jacobian ***w.r.t.* the hyper-parameter $\alpha$** (Section 3), thereby allowing the shape and sparsity of the mapping to be learned automatically. This is particularly appealing in the context of multi-head attention mechanisms, where we shall show in Section 5.1 that different heads tend to learn different sparsity behaviors.

## 3 Adaptively Sparse Transformers with $\alpha$-entmax

We now propose a novel Transformer architecture wherein we simply replace softmax with $\alpha$-entmax in the attention heads. Concretely, we replace the row normalization $\boldsymbol{\pi}$ in Equation 1 by

$$\boldsymbol{\pi}(\boldsymbol{Z})_{ij} = \alpha\text{-entmax}(\boldsymbol{z}_i)_j \quad (8)$$

This change leads to sparse attention weights, as long as $\alpha > 1$; in particular, $\alpha = 1.5$ is a sensible starting point (Peters et al., 2019).

**Different $\alpha$ per head.** Unlike LSTM-based seq2seq models, where $\alpha$ can be more easily tuned by grid search, in a Transformer, there are many attention heads in multiple layers. Crucial to the power of such models, the different heads capture different linguistic phenomena, some of them isolating important words, others spreading out attention across phrases (Vaswani et al., 2017, Figure 5). This motivates using different, adaptive $\alpha$ values for each attention head, such that some heads may learn to be sparser, and others may become closer to softmax. We propose doing so by treating the $\alpha$ values as neural network parameters, optimized via stochastic gradients along with the other weights.

**Derivatives *w.r.t.* $\alpha$.** In order to optimize $\alpha$ automatically via gradient methods, we must compute the Jacobian of the entmax output *w.r.t.* $\alpha$. Since entmax is defined through an optimization problem, this is non-trivial and cannot be simply handled through automatic differentiation; it falls within the domain of *argmin differentiation*, an active research topic in optimization (Gould et al., 2016; Amos and Kolter, 2017).

One of our key contributions is the derivation of a closed-form expression for this Jacobian. The next proposition provides such an expression, enabling entmax layers with adaptive $\alpha$. To the best of our knowledge, ours is the first neural network module that can automatically, continuously vary

in shape away from softmax and toward sparse mappings like sparsemax.

**Proposition 1.** *Let $\boldsymbol{p}^\star := \alpha\text{-entmax}(\boldsymbol{z})$ be the solution of Equation 4. Denote the distribution $\tilde{p}_i := {(p_i^\star)^{2-\alpha}}/{\sum_j (p_j^\star)^{2-\alpha}}$ and let $h_i := -p_i^\star \log p_i^\star$. The $i^{th}$ component of the Jacobian $\boldsymbol{g} := \frac{\partial\, \alpha\text{-entmax}(\boldsymbol{z})}{\partial \alpha}$ is*

$$g_i = \begin{cases} \frac{p_i^\star - \tilde{p}_i}{(\alpha-1)^2} + \frac{h_i - \tilde{p}_i \sum_j h_j}{\alpha - 1}, & \alpha > 1, \\ \frac{h_i \log p_i^\star - p_i^\star \sum_j h_j \log p_j^\star}{2}, & \alpha = 1. \end{cases} \tag{9}$$

The proof uses implicit function differentiation and is given in Appendix C.

Proposition 1 provides the remaining missing piece needed for training adaptively sparse Transformers. In the following section, we evaluate this strategy on neural machine translation, and analyze the behavior of the learned attention heads.

## 4 Experiments

We apply our adaptively sparse Transformers on four machine translation tasks. For comparison, a natural baseline is the standard Transformer architecture using the softmax transform in its multi-head attention mechanisms. We consider two other model variants in our experiments that make use of different normalizing transformations:

- **1.5-entmax:** a Transformer with sparse entmax attention with fixed $\alpha = 1.5$ for all heads. This is a novel model, since 1.5-entmax had only been proposed for RNN-based NMT models (Peters et al., 2019), but never in Transformers, where attention modules are not just one single component of the seq2seq model but rather an integral part of all of the model components.
- **$\alpha$-entmax:** an **adaptive** Transformer with sparse entmax attention with a different, learned $\alpha_{i,j}^t$ for each head.

The adaptive model has an additional scalar parameter per attention head per layer for each of the three attention mechanisms (encoder self-attention, context attention, and decoder self-attention), *i.e.*,

$$\left\{a_{i,j}^t \in \mathbb{R}:\ i \in \{1,\dots,L\}, j \in \{1,\dots,H\}, \right. \\ \left. t \in \{\mathtt{enc}, \mathtt{ctx}, \mathtt{dec}\}\right\}, \tag{10}$$

and we set $\alpha_{i,j}^t = 1 + \mathsf{sigmoid}(a_{i,j}^t) \in ]1, 2[$. All or some of the $\alpha$ values can be tied if desired, but we keep them independent for analysis purposes.

**Datasets.** Our models were trained on 4 machine translation datasets of different training sizes:

- IWSLT 2017 German → English (DE→EN, Cettolo et al., 2017): 200K sentence pairs.
- KFTT Japanese → English (JA→EN, Neubig, 2011): 300K sentence pairs.
- WMT 2016 Romanian → English (RO→EN, Bojar et al., 2016): 600K sentence pairs.
- WMT 2014 English → German (EN→DE, Bojar et al., 2014): 4.5M sentence pairs.

All of these datasets were preprocessed with byte-pair encoding (BPE; Sennrich et al., 2016), using joint segmentations of 32k merge operations.

**Training.** We follow the dimensions of the Transformer-Base model of Vaswani et al. (2017): The number of layers is $L = 6$ and number of heads is $H = 8$ in the encoder self-attention, the context attention, and the decoder self-attention. We use a mini-batch size of 8192 tokens and warm up the learning rate linearly until 20k steps, after which it decays according to an inverse square root schedule. All models were trained until convergence of validation accuracy, and evaluation was done at each 10k steps for RO→EN and EN→DE and at each 5k steps for DE→EN and JA→EN. The end-to-end computational overhead of our methods, when compared to standard softmax, is relatively small; in training tokens per second, the models using $\alpha$-entmax and 1.5-entmax are, respectively, 75% and 90% the speed of the softmax model.

**Results.** We report test set tokenized BLEU (Papineni et al., 2002) results in Table 1. We can see that replacing softmax by entmax does not hurt performance in any of the datasets; indeed, sparse attention Transformers tend to have slightly higher BLEU, but their sparsity leads to a better potential for analysis. In the next section, we make use of this potential by exploring the learned internal mechanics of the self-attention heads.

## 5 Analysis

We conduct an analysis for the higher-resource dataset WMT 2014 English → German of the attention in the sparse adaptive Transformer model ($\alpha$-entmax) at multiple levels: we analyze high-level statistics as well as individual head behavior. Moreover, we make a qualitative analysis of the interpretability capabilities of our models.

| activation | DE→EN | JA→EN | RO→EN | EN→DE |
|---|---|---|---|---|
| softmax | 29.79 | 21.57 | 32.70 | 26.02 |
| 1.5-entmax | 29.83 | **22.13** | **33.10** | 25.89 |
| $\alpha$-entmax | **29.90** | 21.74 | 32.89 | **26.93** |

Table 1: Machine translation tokenized BLEU test results on IWSLT 2017 DE→EN, KFTT JA→EN, WMT 2016 RO→EN and WMT 2014 EN→DE, respectively.

### 5.1 High-Level Statistics

**What kind of $\alpha$ values are learned?** Figure 2 shows the learning trajectories of the $\alpha$ parameters of a selected subset of heads. We generally observe a tendency for the randomly-initialized $\alpha$ parameters to decrease initially, suggesting that softmax-like behavior may be preferable while the model is still very uncertain. After around one thousand steps, some heads change direction and become sparser, perhaps as they become more confident and specialized. This shows that the initialization of $\alpha$ does not predetermine its sparsity level or the role the head will have throughout. In particular, head 8 in the encoder self-attention layer 2 first drops to around $\alpha = 1.3$ before becoming one of the sparsest heads, with $\alpha \approx 2$.

The overall distribution of $\alpha$ values at convergence can be seen in Figure 3. We can observe that the encoder self-attention blocks learn to concentrate the $\alpha$ values in two modes: a very sparse one around $\alpha \to 2$, and a dense one between softmax and 1.5-entmax. However, the decoder self and context attention only learn to distribute these parameters in a single mode. We show next that this is reflected in the average density of attention weight vectors as well.

**Attention weight density when translating.** For any $\alpha > 1$, it would still be possible for the weight matrices in Equation 3 to learn re-scalings so as to make attention sparser or denser. To visualize the impact of adaptive $\alpha$ values, we compare the empirical attention weight density (the average number of tokens receiving non-zero attention) within each module, against sparse Transformers with fixed $\alpha = 1.5$.

Figure 4 shows that, with fixed $\alpha = 1.5$, heads tend to be sparse and similarly-distributed in all three attention modules. With learned $\alpha$, there are two notable changes: (i) a prominent mode corresponding to fully dense probabilities, showing that our models learn to combine sparse and dense attention, and (ii) a distinction between the encoder self-attention – whose background distribution tends toward extreme sparsity – and the other two modules, who exhibit more uniform background distributions. This suggests that perhaps entirely sparse Transformers are suboptimal.

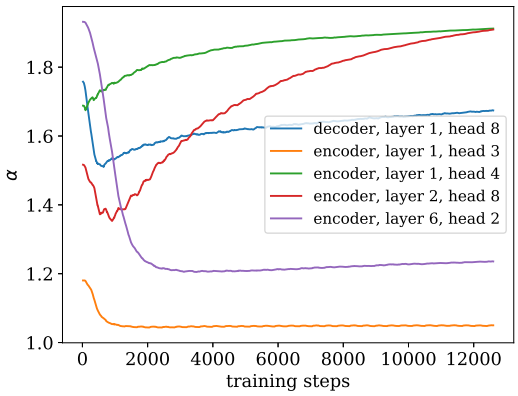


Figure 2: Trajectories of $\alpha$ values for a subset of the heads during training. Initialized at random, most heads become denser in the beginning, before converging. This suggests that dense attention may be more beneficial while the network is still uncertain, being replaced by sparse attention afterwards.

The fact that the decoder seems to prefer denser attention distributions might be attributed to it being auto-regressive, only having access to past tokens and not the full sentence. We speculate that it might lose too much information if it assigned weights of zero to too many tokens in the self-attention, since there are fewer tokens to attend to in the first place.

Teasing this down into separate layers, Figure 5 shows the average (sorted) density of each head for each layer. We observe that $\alpha$-entmax is able to learn different sparsity patterns at each layer, leading to more variance in individual head behavior, to clearly-identified dense and sparse heads, and overall to different tendencies compared to the fixed case of $\alpha = 1.5$.

**Head diversity.** To measure the overall disagreement between attention heads, as a measure of head

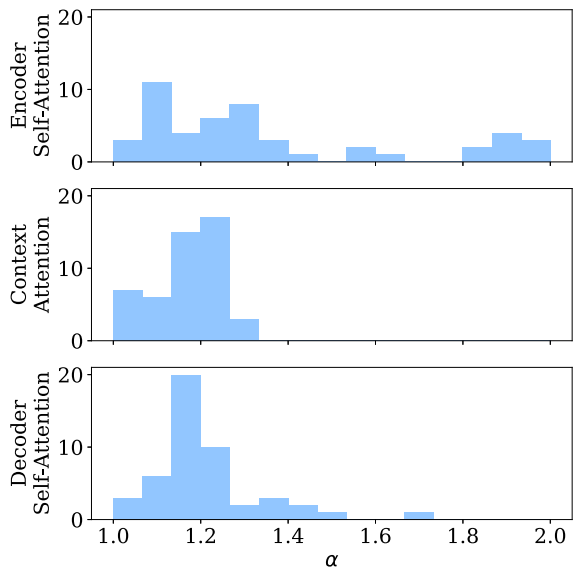


Figure 3: Distribution of learned $\alpha$ values per attention block. While the encoder self-attention has a bimodal distribution of values of $\alpha$, the decoder self-attention and context attention have a single mode.

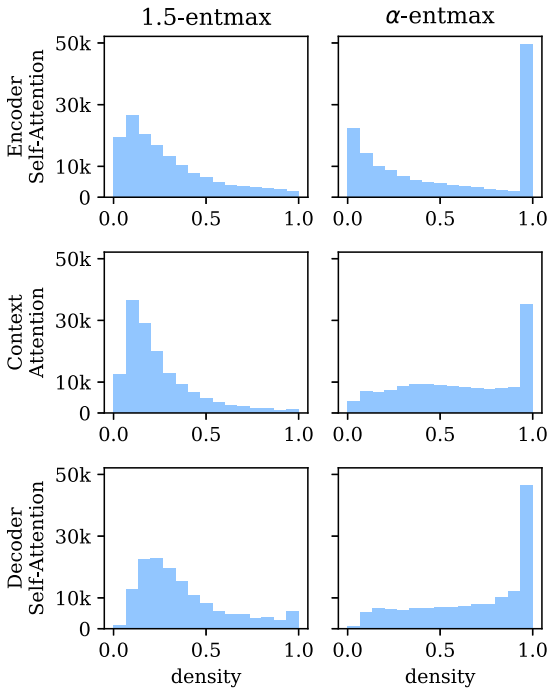


Figure 4: Distribution of attention densities (average number of tokens receiving non-zero attention weight) for all attention heads and all validation sentences. When compared to 1.5-entmax, $\alpha$-entmax distributes the sparsity in a more uniform manner, with a clear mode at fully dense attentions, corresponding to the heads with low $\alpha$. In the softmax case, this distribution would lead to a single bar with density 1.

diversity, we use the following generalization of the Jensen-Shannon divergence:

$$JS = \mathsf{H}^{\mathsf{S}}\left(\frac{1}{H}\sum_{j=1}^{H}\boldsymbol{p}_j\right) - \frac{1}{H}\sum_{j=1}^{H}\mathsf{H}^{\mathsf{S}}(\boldsymbol{p}_j) \quad (11)$$

where $\boldsymbol{p}_j$ is the vector of attention weights assigned by head $j$ to each word in the sequence, and $\mathsf{H}^{\mathsf{S}}$ is the Shannon entropy, base-adjusted based on the dimension of $\boldsymbol{p}$ such that $JS \leq 1$. We average this measure over the entire validation set. The higher this metric is, the more the heads are taking different roles in the model.

Figure 6 shows that both sparse Transformer variants show more diversity than the traditional softmax one. Interestingly, diversity seems to peak in the middle layers of the encoder self-attention and context attention, while this is not the case for the decoder self-attention.

The statistics shown in this section can be found for the other language pairs in Appendix A.

### 5.2 Identifying Head Specializations

Previous work pointed out some specific roles played by different heads in the softmax Transformer model (Voita et al., 2018; Tang et al., 2018; Voita et al., 2019). Identifying the specialization of a head can be done by observing the type of tokens or sequences that the head often assigns most of its attention weight; this is facilitated by sparsity.

**Positional heads.** One particular type of head, as noted by Voita et al. (2019), is the positional head. These heads tend to focus their attention on either the previous or next token in the sequence, thus obtaining representations of the neighborhood of the current time step. In Figure 7, we show attention plots for such heads, found for each of the studied models. The sparsity of our models allows these heads to be more confident in their representations, by assigning the whole probability distribution to a single token in the sequence. Concretely, we may measure a positional head's **confidence** as the average attention weight assigned to the previous token. The softmax model has three heads for position $-1$, with median confidence $93.5\%$. The $1.5$-entmax model also has three heads for this position, with median confidence $94.4\%$. The adaptive model has four heads, with median confidences $95.9\%$, the lowest-confidence head being dense with $\alpha = 1.18$, while the highest-confidence head being sparse ($\alpha = 1.91$).

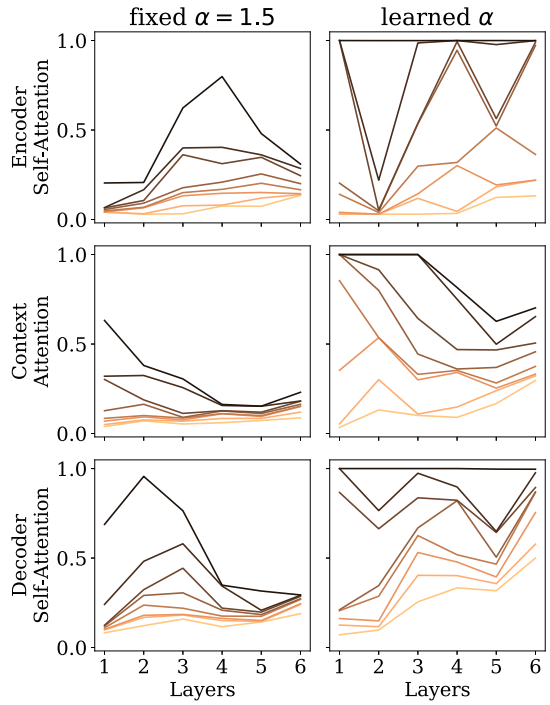


Figure 5: Head density per layer for fixed and learned $\alpha$. Each line corresponds to an attention head; lower values mean that that attention head is sparser. Learned $\alpha$ has higher variance.

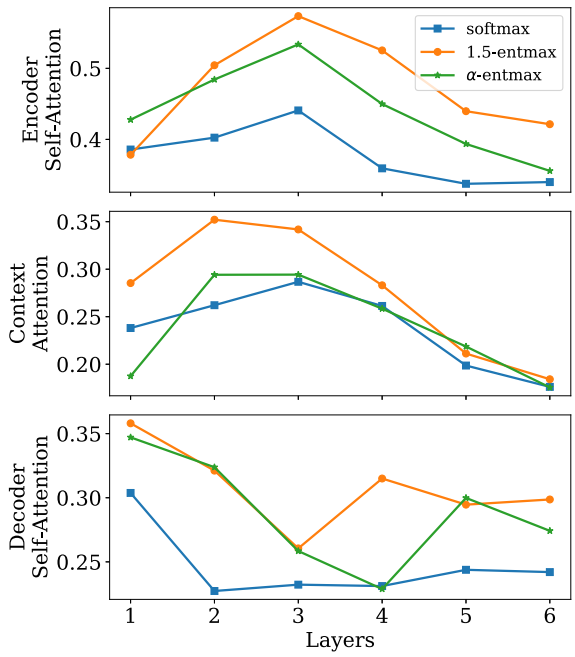


Figure 6: Jensen-Shannon Divergence between heads at each layer. Measures the disagreement between heads: the higher the value, the more the heads are disagreeing with each other in terms of where to attend. Models using sparse entmax have more diverse attention than the softmax baseline.

For position $+1$, the models each dedicate one head, with confidence around 95%, slightly higher for entmax. The adaptive model sets $\alpha = 1.96$ for this head.

**BPE-merging head.** Due to the sparsity of our models, we are able to identify other head specializations, easily identifying which heads should be further analysed. In Figure 8 we show one such head where the $\alpha$ value is particularly high (in the encoder, layer 1, head 4 depicted in Figure 2). We found that this head most often looks at the current time step with high confidence, making it a positional head with offset 0. However, this head often spreads weight sparsely over 2-3 neighboring tokens, when the tokens are part of the same BPE cluster[2] or hyphenated words. As this head is in the first layer, it provides a useful service to the higher layers by combining information evenly within some BPE clusters.

For each BPE cluster or cluster of hyphenated words, we computed a score between 0 and 1 that corresponds to the maximum attention mass assigned by any token to the rest of the tokens inside the cluster in order to quantify the BPE-merging capabilities of these heads.[3] There are not any attention heads in the softmax model that are able to obtain a score over 80%, while for 1.5-entmax and $\alpha$-entmax there are two heads in each (83.3% and 85.6% for 1.5-entmax and 88.5% and 89.8% for $\alpha$-entmax).

**Interrogation head.** On the other hand, in Figure 9 we show a head for which our adaptively sparse model chose an $\alpha$ close to 1, making it closer to softmax (also shown in *encoder, layer 1, head 3* depicted in Figure 2). We observe that this head assigns a high probability to question marks at the end of the sentence in time steps where the current token is interrogative, thus making it an interrogation-detecting head. We also observe this type of heads in the other models, which we also depict in Figure 9. The average attention weight placed on the question mark when the current token is an interrogative word is 98.5% for softmax, 97.0% for 1.5-entmax, and 99.5% for $\alpha$-entmax.

Furthermore, we can examine sentences where some tendentially sparse heads become less so, thus identifying sources of ambiguity where the head

[2]BPE-segmented words are denoted by $\sim$ in the figures.

[3]If the cluster has size 1, the score is the weight the token assigns to itself.

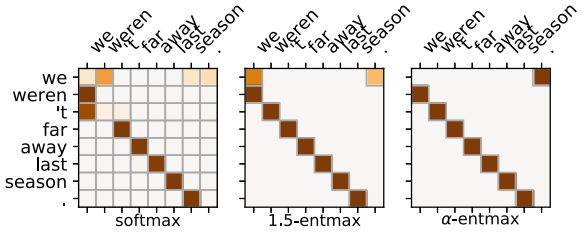


Figure 7: Self-attention from the most confidently previous-position head in each model. The learned parameter in the $\alpha$-entmax model is $\alpha = 1.91$. Quantitatively more confident, visual inspection confirms that the adaptive head behaves more consistently.

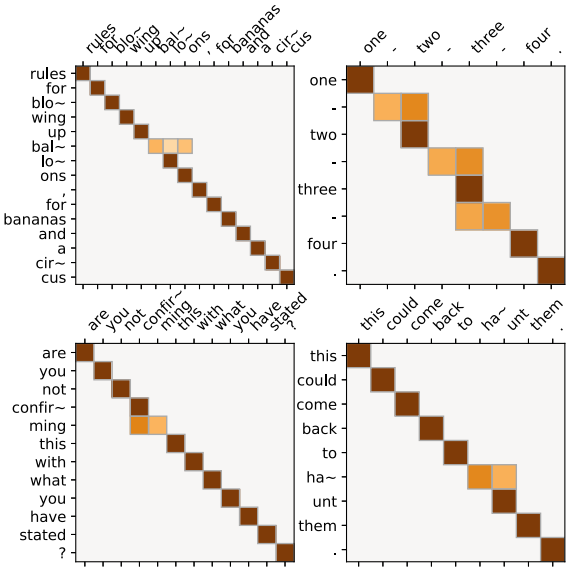


Figure 8: BPE-merging head ($\alpha = 1.91$) discovered in the $\alpha$-entmax model. Found in the first encoder layer, this head learns to discover some subword units and combine their information, leaving most words intact. It places 99.09% of its probability mass within the same BPE cluster as the current token: more than any head in any other model.

is less confident in its prediction. An example is shown in Figure 10 where sparsity in the same head differs for sentences of similar length.

## 6 Related Work

**Sparse attention.** Prior work has developed sparse attention mechanisms, including applications to NMT (Martins and Astudillo, 2016; Malaviya et al., 2018; Niculae and Blondel, 2017; Shao et al., 2019; Maruf et al., 2019). Peters et al. (2019) introduced the entmax function this work builds upon. In their work, there is a single attention mechanism which is controlled by a fixed $\alpha$. In contrast, this is the first work to allow such attention mappings to *dynamically* adapt their curvature and sparsity, by automatically adjusting the continuous $\alpha$ parameter. We also provide the first results using sparse attention in a Transformer model.

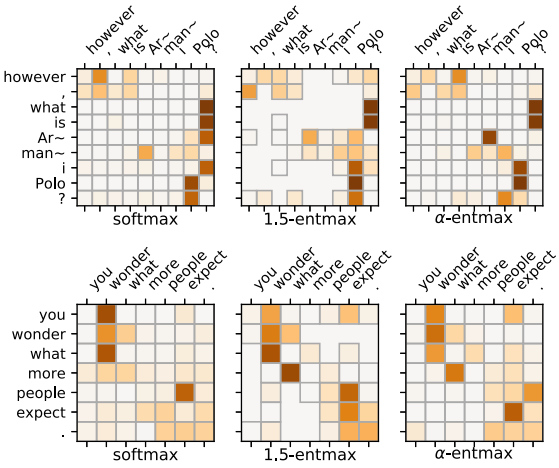


Figure 9: Interrogation-detecting heads in the three models. The top sentence is interrogative while the bottom one is declarative but includes the interrogative word "what". In the top example, these *interrogation heads* assign a high probability to the question mark in the time step of the interrogative word (with $\geq 97.0\%$ probability), while in the bottom example since there is no question mark, the same head does not assign a high probability to the last token in the sentence during the interrogative word time step. Surprisingly, this head prefers a low $\alpha = 1.05$, as can be seen from the dense weights. This allows the head to identify the noun phrase "Armani Polo" better.

**Fixed sparsity patterns.** Recent research improves the scalability of Transformer-like networks through static, fixed sparsity patterns (Child et al., 2019; Wu et al., 2019). Our adaptively-sparse Transformer can dynamically select a sparsity pattern that finds relevant words regardless of their position (*e.g.*, Figure 9). Moreover, the two strategies could be combined. In a concurrent line of research, Sukhbaatar et al. (2019) propose an adaptive attention span for Transformer language models. While their work has each head learn a different contiguous span of context tokens to attend to, our work finds different sparsity patterns in the same span. Interestingly, some of their findings mirror ours – we found that attention heads in the last layers tend to be denser on average when compared to the ones in the first layers, while their work has found that lower layers tend to have a shorter attention span compared to higher layers.

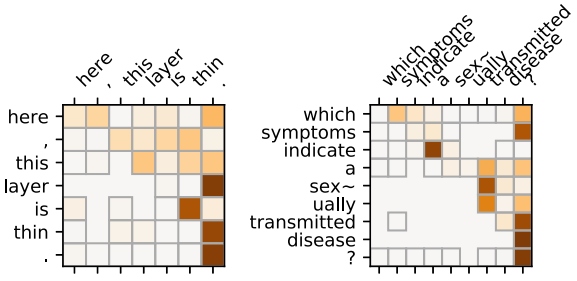


Figure 10: Example of two sentences of similar length where the same head ($\alpha = 1.33$) exhibits different sparsity. The longer phrase in the example on the right "a sexually transmitted disease" is handled with higher confidence, leading to more sparsity.

**Transformer interpretability.** The original Transformer paper (Vaswani et al., 2017) shows attention visualizations, from which some speculation can be made of the roles the several attention heads have. Mareček and Rosa (2018) study the syntactic abilities of the Transformer self-attention, while Raganato and Tiedemann (2018) extract dependency relations from the attention weights. Tenney et al. (2019) find that the self-attentions in BERT (Devlin et al., 2019) follow a sequence of processes that resembles a classical NLP pipeline. Regarding redundancy of heads, Voita et al. (2019) develop a method that is able to prune heads of the multi-head attention module and make an empirical study of the role that each head has in self-attention (positional, syntactic and rare words). Li et al. (2018) also aim to reduce head redundancy by adding a regularization term to the loss that maximizes head disagreement and obtain improved results. While not considering Transformer attentions, Jain and Wallace (2019) show that traditional attention mechanisms do not necessarily improve interpretability since softmax attention is vulnerable to an adversarial attack leading to wildly different model predictions for the same attention weights. Sparse attention may mitigate these issues; however, our work focuses mostly on a more mechanical aspect of interpretation by analyzing head behavior, rather than on explanations for predictions.

## 7 Conclusion and Future Work

We contribute a novel strategy for adaptively sparse attention, and, in particular, for adaptively sparse Transformers. We present the first empirical analysis of Transformers with sparse attention mappings (*i.e.*, entmax), showing potential in both translation accuracy as well as in model interpretability.

In particular, we analyzed how the attention heads in the proposed adaptively sparse Transformer can specialize more and with higher confidence. Our adaptivity strategy relies only on gradient-based optimization, side-stepping costly per-head hyper-parameter searches. Further speed-ups are possible by leveraging more parallelism in the bisection algorithm for computing $\alpha$-entmax.

Finally, some of the automatically-learned behaviors of our adaptively sparse Transformers – for instance, the near-deterministic positional heads or the subword joining head – may provide new ideas for designing static variations of the Transformer.

## Acknowledgments

This work was supported by the European Research Council (ERC StG DeepSPIN 758969), and by the Fundação para a Ciência e Tecnologia through contracts UID/EEA/50008/2019 and CMUPERI/TIC/0046/2014 (GoLocal). We are grateful to Ben Peters for the $\alpha$-entmax code and Erick Fonseca, Marcos Treviso, Pedro Martins, and Tsvetomila Mihaylova for insightful group discussion. We thank Mathieu Blondel for the idea to learn $\alpha$. We would also like to thank the anonymous reviewers for their helpful feedback.

## References

Brandon Amos and J. Zico Kolter. 2017. OptNet: Differentiable optimization as a layer in neural networks. In *Proc. ICML*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. ICLR*.

Mathieu Blondel, André FT Martins, and Vlad Niculae. 2019. Learning classifiers with Fenchel-Young losses: Generalized entropies, margins, and algorithms. In *Proc. AISTATS*.

Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, et al. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proc. Workshop on Statistical Machine Translation*.

Ondrej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, et al. 2016. Findings of the 2016 conference on machine translation. In *Proc. WMT*.

M Cettolo, M Federico, L Bentivogli, J Niehues, S Stüker, K Sudoh, K Yoshino, and C Federmann. 2017. Overview of the IWSLT 2017 evaluation campaign. In *Proc. IWSLT*.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse Transformers. *preprint arXiv:1904.10509*.

Frank H Clarke. 1990. *Optimization and Nonsmooth Analysis*. SIAM.

Yuntian Deng, Yoon Kim, Justin Chiu, Demi Guo, and Alexander Rush. 2018. Latent alignment and variational attention. In *Proc. NeurIPS*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. NAACL-HLT*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *Proc. ICML*.

Stephen Gould, Basura Fernando, Anoop Cherian, Peter Anderson, Rodrigo Santa Cruz, and Edison Guo. 2016. On differentiating parameterized argmin and argmax problems with application to bi-level optimization. *preprint arXiv:1607.05447*.

Michael Held, Philip Wolfe, and Harlan P Crowder. 1974. Validation of subgradient optimization. *Mathematical Programming*, 6(1):62–88.

Sarthak Jain and Byron C. Wallace. 2019. Attention is not explanation. In *Proc. NAACL-HLT*.

Marcin Junczys-Dowmunt, Kenneth Heafield, Hieu Hoang, Roman Grundkiewicz, and Anthony Aue. 2018. Marian: Cost-effective high-quality neural machine translation in C++. In *Proc. WNMT*.

Jian Li, Zhaopeng Tu, Baosong Yang, Michael R Lyu, and Tong Zhang. 2018. Multi-Head Attention with Disagreement Regularization. In *Proc. EMNLP*.

Christos Louizos, Max Welling, and Diederik P Kingma. 2018. Learning sparse neural networks through $L_0$ regularization. *Proc. ICLR*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proc. EMNLP*.

Chaitanya Malaviya, Pedro Ferreira, and André FT Martins. 2018. Sparse and constrained attention for neural machine translation. In *Proc. ACL*.

David Mareček and Rudolf Rosa. 2018. Extracting syntactic trees from Transformer encoder self-attentions. In *Proc. BlackboxNLP*.

André FT Martins and Ramón Fernandez Astudillo. 2016. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *Proc. of ICML*.

Sameen Maruf, André FT Martins, and Gholamreza Haffari. 2019. Selective attention for context-aware neural machine translation. *preprint arXiv:1903.08788*.

Graham Neubig. 2011. The Kyoto free translation task. `http://www.phontron.com/kftt`.

Vlad Niculae and Mathieu Blondel. 2017. A regularized framework for sparse and structured neural attention. In *Proc. NeurIPS*.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. 2018. Scaling neural machine translation. In *Proc. WMT*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proc. ACL*.

Ben Peters, Vlad Niculae, and André FT Martins. 2019. Sparse sequence-to-sequence models. In *Proc. ACL*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *preprint*.

Alessandro Raganato and Jörg Tiedemann. 2018. An analysis of encoder representations in Transformer-based machine translation. In *Proc. BlackboxNLP*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proc. ACL*.

Wenqi Shao, Tianjian Meng, Jingyu Li, Ruimao Zhang, Yudian Li, Xiaogang Wang, and Ping Luo. 2019. SSN: Learning sparse switchable normalization via SparsestMax. In *Proc. CVPR*.

Sainbayar Sukhbaatar, Edouard Grave, Piotr Bojanowski, and Armand Joulin. 2019. Adaptive Attention Span in Transformers. In *Proc. ACL*.

Gongbo Tang, Mathias Müller, Annette Rios, and Rico Sennrich. 2018. Why self-attention? A targeted evaluation of neural machine translation architectures. In *Proc. EMNLP*.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proc. ACL*.

Constantino Tsallis. 1988. Possible generalization of Boltzmann-Gibbs statistics. *Journal of Statistical Physics*, 52:479–487.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proc. NeurIPS*.

Elena Voita, Pavel Serdyukov, Rico Sennrich, and Ivan Titov. 2018. Context-aware neural machine translation learns anaphora resolution. In *Proc. ACL*.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proc. ACL*.

Felix Wu, Angela Fan, Alexei Baevski, Yann N Dauphin, and Michael Auli. 2019. Pay less attention with lightweight and dynamic convolutions. In *Proc. ICLR*.

# Show Your Work: Improved Reporting of Experimental Results

**Jesse Dodge**[♣] **Suchin Gururangan**[♢] **Dallas Card**[♡] **Roy Schwartz**[♠♢] **Noah A. Smith**[♠♢]
[♣]Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA, USA
[♢]Allen Institute for Artificial Intelligence, Seattle, WA, USA
[♡]Machine Learning Department, Carnegie Mellon University, Pittsburgh, PA, USA
[♠]Paul G. Allen School of Computer Science & Engineering, University of Washington, Seattle, WA, USA
{jessed,dcard}@cs.cmu.edu {suching,roys,noah}@allenai.org

## Abstract

Research in natural language processing proceeds, in part, by demonstrating that new models achieve superior performance (e.g., accuracy) on held-out test data, compared to previous results. In this paper, we demonstrate that test-set performance scores alone are insufficient for drawing accurate conclusions about which model performs best. We argue for reporting additional details, especially performance on validation data obtained during model development. We present a novel technique for doing so: *expected validation performance* of the best-found model as a function of computation budget (i.e., the number of hyperparameter search trials or the overall training time). Using our approach, we find multiple recent model comparisons where authors would have reached a different conclusion if they had used more (or less) computation. Our approach also allows us to estimate the amount of computation required to obtain a given accuracy; applying it to several recently published results yields massive variation across papers, from hours to weeks. We conclude with a set of best practices for reporting experimental results which allow for robust future comparison, and provide code to allow researchers to use our technique.[1]

## 1 Introduction

In NLP and machine learning, improved performance on held-out test data is typically used as an indication of the superiority of one method over others. But, as the field grows, there is an increasing gap between the large computational budgets used for some high-profile experiments and the budgets used in most other work (Schwartz et al., 2019). This hinders meaningful comparison between experiments, as improvements in performance can, in some cases, be ob-

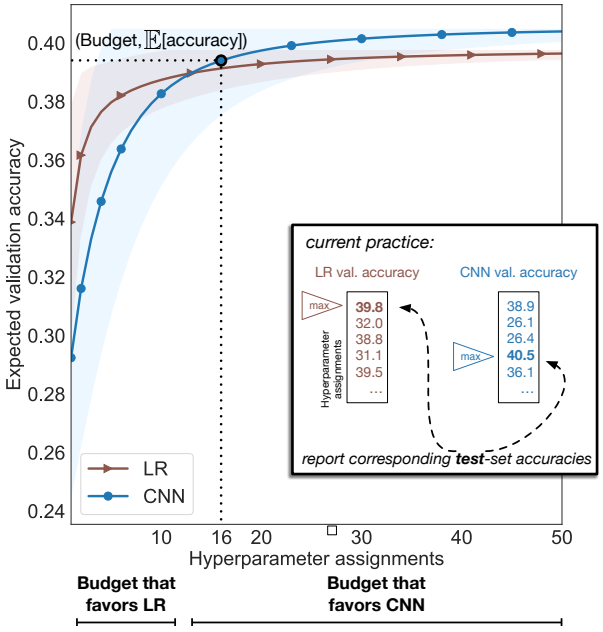


Figure 1: Current practice when compraing NLP models is to train multiple instantiations of each, choose the best model of each type based on validation performance, and compare their performance on test data (inner box). Under this setup, (assuming test-set results are similar to validation), one would conclude from the results above (hyperparameter search for two models on the 5-way SST classification task) that the CNN outperforms Logistic Regression (LR). In our proposed evaluation framework, we instead encourage practitioners to consider the expected validation accuracy ($y$-axis; shading shows $\pm 1$ standard deviation), as a function of budget ($x$-axis). Each point on a curve is the *expected value* of the best validation accuracy obtained ($y$) after evaluating $x$ random hyperparameter values. Note that (1) the better performing model depends on the computational budget; LR has higher expected performance for budgets up to 10 hyperparameter assignments, while the CNN is better for larger budgets. (2) Given a model and desired accuracy (e.g., 0.395 for CNN), we can estimate the expected budget required to reach it (16; dotted lines).

[1] https://github.com/allenai/allentune



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2185–2194, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

tained purely through more intensive hyperparameter tuning (Melis et al., 2018; Lipton and Steinhardt, 2018).[2]

Moreover, recent investigations into "state-of-the-art" claims have found competing methods to only be comparable, without clear superiority, even against baselines (Reimers and Gurevych, 2017; Lucic et al., 2018; Li and Talwalkar, 2019); this has exposed the need for reporting more than a single point estimate of performance.

Echoing calls for more rigorous scientific practice in machine learning (Lipton and Steinhardt, 2018; Sculley et al., 2018), we draw attention to the weaknesses in current reporting practices and propose solutions which would allow for fairer comparisons and improved reproducibility.

Our primary technical contribution is the introduction of a tool for reporting validation results in an easily interpretable way: *expected validation performance* of the best model under a given computational budget.[3] That is, given a budget sufficient for training and evaluating $n$ models, we calculate the expected performance of the best of these models on validation data. Note that this differs from the *best observed* value after $n$ evaluations. Because the expectation can be estimated from the distribution of $N$ validation performance values, with $N \geq n$, and these are obtained during model development,[4] our method **does not require additional computation** beyond hyperparameter search or optimization. We encourage researchers to report expected validation performance as a curve, across values of $n \in \{1, \ldots, N\}$.

As we show in §4.3, our approach makes clear that the expected-best performing model is a function of the computational budget. In §4.4 we show how our approach can be used to estimate the budget that went into obtaining previous results; in one example, we see a too-small budget for baselines, while in another we estimate a budget of about 18 GPU days was used (but not reported). Previous work on reporting validation performance used the bootstrap to approximate the mean and variance of the best performing model (Lucic et al., 2018); in §3.2 we show that our approach computes these values with strictly less error than the bootstrap.

We conclude by presenting a set of recommendations for researchers that will improve scientific reporting over current practice. We emphasize this work is about *reporting*, not about running additional experiments (which undoubtedly can improve evidence in comparisons among models). Our reporting recommendations aim at reproducibility and improved understanding of sensitivity to hyperparameters and random initializations. Some of our recommendations may seem obvious; however, our empirical analysis shows that out of fifty EMNLP 2018 papers chosen at random, none report all items we suggest.

## 2 Background

**Reproducibility** Reproducibility in machine learning is often defined as the ability to produce the *exact* same results as reported by the developers of the model. In this work, we follow Gundersen and Kjensmo (2018) and use an extended notion of this concept: when comparing two methods, two research groups with different implementations should follow an experimental procedure which leads to the same conclusion about which performs better. As illustrated in Fig. 1, this conclusion often depends on the amount of computation applied. Thus, to make a *reproducible* claim about which model performs best, we must also take into account the budget used (e.g., the number of hyperparameter trials).

**Notation** We use the term *model family* to refer to an approach subject to comparison and to hyperparameter selection.[5] Each model family $\mathcal{M}$ requires its own hyperparameter selection, in terms of a set of $k$ hypermarameters, each of which defines a range of possible values. A *hyperparameter value* (denoted $h$) is a $k$-tuple of specific values for each hyperparameter. We call the set of all possible hyperparameter values $\mathcal{H}_\mathcal{M}$.[6] Given $\mathcal{H}_\mathcal{M}$ and a computational budget sufficient for training $B$ models, the set of hyperparameter values is $\{h_1, \ldots, h_B\}$, $h_i \in \mathcal{H}_\mathcal{M}$. We let $m_i \in \mathcal{M}$ denote the model trained with hyperparameter value $h_i$.

[2] Recent work has also called attention to the environmental cost of intensive model exploration (Strubell et al., 2019).

[3] We use the term *performance* as a general evaluation measure, e.g., accuracy, $F_1$, etc.

[4] We leave forecasting performance with larger budgets $n > N$ to future work.

[5] Examples include different architectures, but also ablations of the same architecture.

[6] The hyperparameter value space can also include the random seed used to initialize the model, and some specifications such as the size of the hidden layers in a neural network, in addition to commonly tuned values such as learning rate.

**Hyperparameter value selection** There are many ways of selecting hyperparameter values, $h_i$. Grid search and uniform sampling are popular systematic methods; the latter has been shown to be superior for most search spaces (Bergstra and Bengio, 2012). Adaptive search strategies such as Bayesian optimization select $h_i$ after evaluating $h_1, \ldots, h_{i-1}$. While these strategies may find better results quickly, they are generally less reproducible and harder to parallelize (Li et al., 2017). Manual search, where practitioners use knowledge derived from previous experience to adjust hyperparameters after each experiment, is a type of adaptive search that is the least reproducible, as different practitioners make different decisions. Regardless of the strategy adopted, we advocate for detailed reporting of the method used for hyperparmeter value selection (§5). We next introduce a technique to visualize results of samples which are drawn i.i.d. (e.g., random initializations or uniformly sampled hyperparameter values).

## 3 Expected Validation Performance Given Budget

After selecting the best hyperparameter values $h_{i^*}$ from among $\{h_1, \ldots, h_B\}$ with actual budget $B$, NLP researchers typically evaluate the associated model $m_{i^*}$ on the test set and report its performance as an estimate of the family $\mathcal{M}$'s ability to generalize to new data. We propose to make better use of the intermediately-trained models $m_1, \ldots, m_B$.

For any set of $n$ hyperparmeter values, denote the validation performance of the best model as

$$v_n^* = \max_{h \in \{h_1, \ldots, h_n\}} \mathcal{A}(\mathcal{M}, h, \mathcal{D}_T, \mathcal{D}_V), \quad (1)$$

where $\mathcal{A}$ denotes an algorithm that returns the performance on validation data $\mathcal{D}_V$ after training a model from family $\mathcal{M}$ with hyperparameter values $h$ on training data $\mathcal{D}_T$.[7] We view evaluations of $\mathcal{A}$ as the elementary unit of experimental cost.[8]

Though not often done in practice, procedure (1) could be repeated many times with different hyperparameter values, yielding a *distribution* of values for random variable $V_n^*$. This would allow us to estimate the *expected* performance, $\mathbb{E}[V_n^* \mid n]$ (given $n$ hyperparameter configurations). The key insight used below is that, if we use random search for hyperparameter selection, then the effort that goes into a single round of random search (Eq. 1) suffices to construct a useful estimate of expected validation performance, without requiring *any further experimentation*.

Under random search, the $n$ hyperparameter values $h_1, \ldots, h_n$ are drawn uniformly at random from $\mathcal{H}_\mathcal{M}$, so the values of $\mathcal{A}(\mathcal{M}, h_i, \mathcal{D}_T, \mathcal{D}_V)$ are i.i.d. As a result, the maximum among these is itself a random variable. We introduce a diagnostic that captures information about the computation used to generate a result: the expectation of maximum performance, *conditioned* on $n$, the amount of computation used in the maximization over hyperparameters and random initializations:

$$\mathbb{E}\left[\max_{h \in \{h_1, \ldots, h_n\}} \mathcal{A}(\mathcal{M}, h, \mathcal{D}_T, \mathcal{D}_V) \mid n\right]. \quad (2)$$

Reporting this expectation as we vary $n \in \{1, 2, \ldots, B\}$ gives more information than the maximum $v_B^*$ (Eq. 1 with $n = B$); future researchers who use this model will know more about the computation budget required to achieve a given performance. We turn to calculating this expectation, then we compare it to the bootstrap (§3.2), and discuss estimating variance (§3.3).

### 3.1 Expected Maximum

We describe how to estimate the expected maximum validation performance (Eq. 2) given a budget of $n$ hyperparameter values.[9]

Assume we draw $\{h_1, \ldots, h_n\}$ uniformly at random from hyperparameter space $\mathcal{H}_\mathcal{M}$. Each evaluation of $\mathcal{A}(\mathcal{M}, h, \mathcal{D}_T, \mathcal{D}_V)$ is therefore an i.i.d. draw of a random variable, denoted $V_i$, with observed value $v_i$ for $h_i \sim \mathcal{H}_\mathcal{M}$. Let the maximum among $n$ i.i.d. draws from an unknown distribution be

$$V_n^* = \max_{i \in \{1, \ldots, n\}} V_i \quad (3)$$

We seek the expected value of $V_n^*$ given $n$:

$$\mathbb{E}[V_n^* \mid n] = \textstyle\sum_v v \cdot P(V_n^* = v \mid n) \quad (4)$$

where $P(V_n^* \mid n)$ is the probability mass function (PMF) for the max-random variable.[10] For dis-

[7] $\mathcal{A}$ captures standard parameter estimation, as well as procedures that depend on validation data, like early stopping.

[8] Note that researchers do not always report validation, but rather *test* performance, a point we will return to in §5.

[9] Conversion to alternate formulations of budget, such as GPU hours or cloud-machine rental cost in dollars, is straightforward in most cases.

[10] For a finite validation set $\mathcal{D}_V$, most performance measures (e.g., accuracy) only take on a finite number of possible values, hence the use of a sum instead of an integral in Eq. 4.

crete random variables,

$$P(V_n^* = v \mid n) = P(V_n^* \leq v \mid n) - P(V_n^* < v \mid n), \quad (5)$$

Using the definition of "max", and the fact that the $V_i$ are drawn i.i.d.,

$$\begin{aligned} P(V_n^* \leq v \mid n) &= P\left(\max_{i \in \{1,\ldots,n\}} V_i \leq v \mid n\right) \\ &= P(V_1 \leq v, V_2 \leq v, \ldots, V_n \leq v \mid n) \\ &= \prod_{i=1}^{n} P(V_i \leq v) = P(V \leq v)^n, \end{aligned} \quad (6)$$

and similarly for $P(V_n^* < v \mid n)$.

$P(V \leq v)$ and $P(V < v)$ are cumulative distribution functions, which we can estimate using the empirical distribution, i.e.

$$\hat{P}(V \leq v) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{[V_i \leq v]} \quad (7)$$

and similarly for strict inequality.

Thus, our estimate of the expected maximum validation performance is

$$\hat{\mathbb{E}}[V_n^* \mid n] = \sum_v v \cdot (\hat{P}(V_i \leq v)^n - \hat{P}(V_i < v)^n). \quad (8)$$

**Discussion** As we increase the amount of computation for evaluating hyperparameter values ($n$), the maximum among the samples will approach the observed maximum $v_B^*$. Hence the curve of $\mathbb{E}[V_n^* \mid n]$ as a function of $n$ will appear to asymptote. Our focus here is not on estimating that value, and we do not make any claims about extrapolation of $V^*$ beyond $B$, the number of hyperparameter values to which $\mathcal{A}$ is actually applied.

Two points follow immediately from our derivation. First, at $n = 1$, $\mathbb{E}[V_1^* \mid n = 1]$ is the mean of $v_1, \ldots, v_n$. Second, for all $n$, $\mathbb{E}[V_n^* \mid n] \leq v_n^* = \max_i v_i$, which means the curve is a lower bound on the selected model's validation performance.

### 3.2 Comparison with Bootstrap

Lucic et al. (2018) and Henderson et al. (2018) have advocated for using the bootstrap to estimate the mean and variance of the best validation performance. The bootstrap (Efron and Tibshirani, 1994) is a general method which can be used to estimate statistics that do not have a closed form. The bootstrap process is as follows: draw $N$ i.i.d. samples (in our case, $N$ model evaluations). From these $N$ points, sample $n$ points (with replacement), and compute the statistic of interest (e.g., the max). Do this $K$ times (where $K$ is large), and average the computed statistic. By the law of large numbers, as $K \to \infty$ this average converges to the sample expected value (Efron and Tibshirani, 1994).

The bootstrap has two sources of error: the error from the finite sample of $N$ points, and the error introduced by resampling these points $K$ times. Our approach has strictly less error than using the bootstrap: our calculation of the expected maximum performance in §3.1 provides a closed-form solution, and thus contains none of the resampling error (the finite sample error is the same).

### 3.3 Variance of $V_n^*$

Expected performance becomes more useful with an estimate of variation. When using the bootstrap, standard practice is to report the standard deviation of the estimates from the $K$ resamples. As $K \to \infty$, this standard deviation approximates the sample standard error (Efron and Tibshirani, 1994). We instead calculate this from the distribution in Eq. 5 using the standard plug-in-estimator.

In most cases, we advocate for reporting a measure of variability such as the standard deviation or variance; however, in some cases it might cause confusion. For example, when the variance is large, plotting the expected value plus the variance can go outside of reasonable bounds, such as accuracy greater than any observed (even greater than 1). In such situations, we recommend shading only values within the observed range, such as in Fig. 4. Additionally, in situations where the variance is high and variance bands overlap between model families (e.g., Fig. 1), the mean is still the most informative statistic.

## 4 Case Studies

Here we show two clear use cases of our method. First, we can directly estimate, for a given budget, which approach has better performance. Second, we can estimate, given our experimental setup, the budget for which the reported validation performance ($V^*$) matches a desired performance level. We present three examples that demonstrate these use cases. First, we reproduce previous findings that compared different models for text classification. Second, we explore the time vs. performance tradeoff of models that use contextual word embeddings (Peters et al., 2018). Third, from two previously published papers, we examine the budget required for our expected performance to

match their reported performance. We find these budget estimates vary drastically. Consistently, we see that the best model is a function of the budget. We publicly release the search space and training configurations used for each case study. [11]

Note that we do not report test performance in our experiments, as our purpose is not to establish a benchmark level for a model, but to demonstrate the utility of expected validation performance for model comparison and reproducibility.

### 4.1 Experimental Details

For each experiment, we document the hyperparameter search space, hardware, average runtime, number of samples, and links to model implementations. We use public implementations for all models in our experiments, primarily in AllenNLP (Gardner et al., 2018). We use Tune (Liaw et al., 2018) to run parallel evaluations of uniformly sampled hyperparameter values.

### 4.2 Validating Previous Findings

We start by applying our technique on a text classification task in order to confirm a well-established observation (Yogatama and Smith, 2015): logistic regression has reasonable performance with minimal hyperparameter tuning, but a well-tuned convolutional neural network (CNN) can perform better.

We experiment with the fine-grained Stanford Sentiment Treebank text classification dataset (Socher et al., 2013). For the CNN classifier, we embed the text with 50-dim GloVe vectors (Pennington et al., 2014), feed the vectors to a ConvNet encoder, and feed the output representation into a softmax classification layer. We use the *scikit-learn* implementation of logistic regression with bag-of-word counts and a linear classification layer. The hyperparameter spaces $\mathcal{H}_{\text{CNN}}$ and $\mathcal{H}_{\text{LR}}$ are detailed in Appendix B. For logistic regression we used bounds suggested by Yogatama and Smith (2015), which include term weighting, n-grams, stopwords, and learning rate. For the CNN we follow the hyperparameter sensitivity analysis in Zhang and Wallace (2015).

We run 50 trials of random hyperparameter search for each classifier. Our results (Fig. 1) confirm previous findings (Zhang and Wallace, 2015): under a budget of fewer than 10 hyperparameter search trials, logistic regression achieves a higher expected validation accuracy than the CNN. As the budget increases, the CNN gradually improves to a higher overall expected validation accuracy. For all budgets, logistic regression has lower variance, so may be a more suitable approach for fast prototyping.

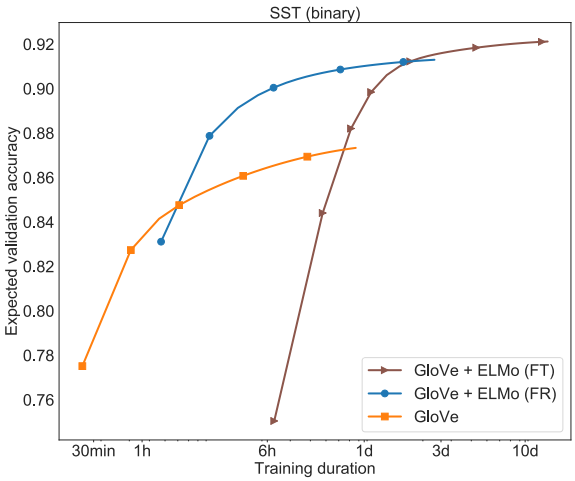


Figure 2: Expected maximum performance of a BCN classifier on SST. We compare three embedding approaches (GloVe embeddings, GloVe + frozen ELMo, and GloVe + fine-tuned ELMo). The $x$-axis is time, on a log scale. We omit the variance for visual clarity. For each of the three model families, we sampled 50 hyperparameter values, and plot the expected maximum performance with the $x$-axis values scaled by the average training duration. The plot shows that for each approach (GloVe, ELMo frozen, and ELMo fine-tuned), there exists a budget for which it is preferable.

### 4.3 Contextual Representations

We next explore how computational budget affects the performance of contextual embedding models (Peters et al., 2018). Recently, Peters et al. (2019) compared two methods for using contextual representations for downstream tasks: *feature extraction*, where features are fixed after pretraining and passed into a task-specific model, or *fine-tuning*, where they are updated during task training. Peters et al. (2019) found that feature extraction is preferable to fine-tuning ELMo embeddings. Here we set to explore whether this conclusion depends on the experimental budget.

Closely following their experimental setup, in Fig. 2 we show the expected performance of the biattentive classification network (BCN; McCann et al., 2017) with three embedding approaches (GloVe only, GloVe + ELMo frozen, and GloVe

[11] https://github.com/allenai/show-your-work

+ ELMo fine-tuned), on the binary Stanford Sentiment Treebank task.[12]

We use *time* for the budget by scaling the curves by the average observed training duration for each model. We observe that as the time budget increases, the expected best-performing model changes. In particular, we find that our experimental setup leads to the same conclusion as Peters et al. (2019) given a budget between approximately 6 hours and 1 day. For larger budgets (e.g., 10 days) fine-tuning outperforms feature extraction. Moreover, for smaller budgets ($<$ 2 hours), using GloVe embeddings is preferable to ELMo (frozen or fine-tuned).

### 4.4 Inferring Budgets in Previous Reports

Our method provides another appealing property: estimating the budget required for the expected performance to reach a particular level, which we can compare against previously reported results. We present two case studies, and show that the amount of computation required to match the reported results varies drastically.

We note that in the two examples that follow, the original papers only reported partial experimental information; we made sure to tune the hyperparameters they did list in addition to standard choices (such as the learning rate). In neither case do they report the method used to tune the hyperparameters, and we suspect they tuned them manually. Our experiments here are meant give an idea of the budget that would be required to reproduce their results or to apply their models to other datasets under random hyperparameter value selection.

**SciTail** When introducing the SciTail textual entailment dataset, Khot et al. (2018) compared four models: an *n-gram* baseline, which measures word-overlap as an indicator of entailment, *ESIM* (Chen et al., 2017), a sequence-based entailment model, *DAM* (Parikh et al., 2016), a bag-of-words entailment model, and their proposed model, *DGEM* (Khot et al., 2018), a graph-based structured entailment model. Their conclusion was that DGEM outperforms the other models.

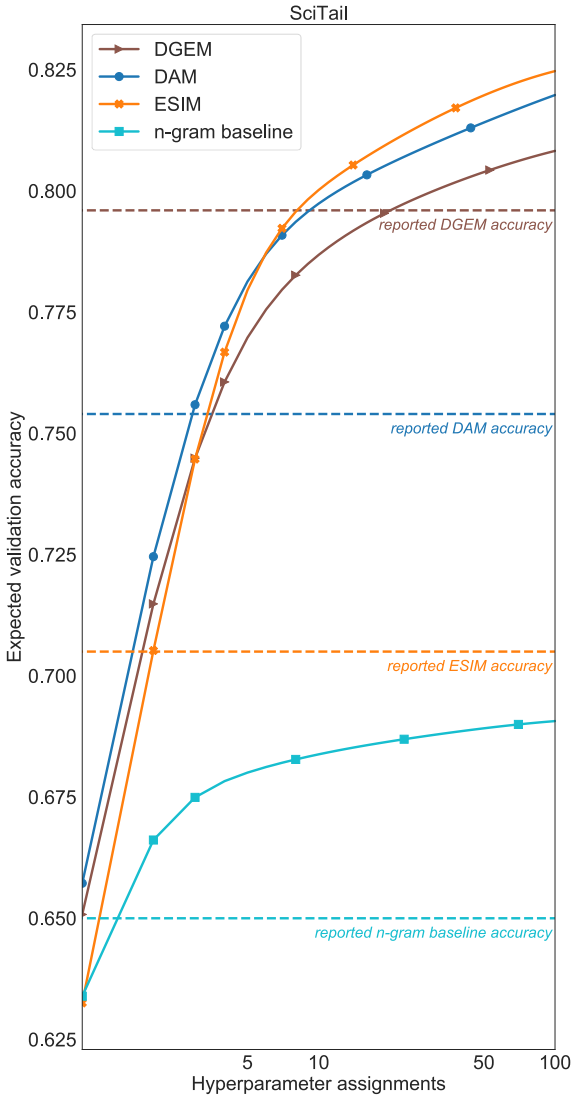


Figure 3: Comparing reported accuracies (dashed lines) on SciTail to expected validation performance under varying levels of compute (solid lines). The estimated budget required for expected performance to match the reported result differs substantially across models, and the relative ordering varies with budget. We omit variance for visual clarity.

We use the same implementations of each of these models each with a hyperparameter search space detailed in Appendix D.[13] We use a budget based on trials instead of runtime so as to emphasize how these models behave when given a comparable number of hyperparameter configurations.

[12] Peters et al. (2019) use a BCN with frozen embeddings and a BiLSTM BCN for fine-tuning. We conducted experiments with both a BCN and a BiLSTM with frozen and fine-tuned embeddings, and found our conclusions to be consistent. We report the full hyperparameter search space, which matched Peters et al. (2019) as closely as their reporting allowed, in Appendix C.

[13] The search space bounds we use are large neighborhoods around the hyperparameter assignments specified in the public implementations of these models. Note that these curves depend on the specific hyperparameter search space adopted; as the original paper does not report hyperparameter search or model selection details, we have chosen what we believe to be reasonable bounds, and acknowledge that different choices could result in better or worse expected performance.

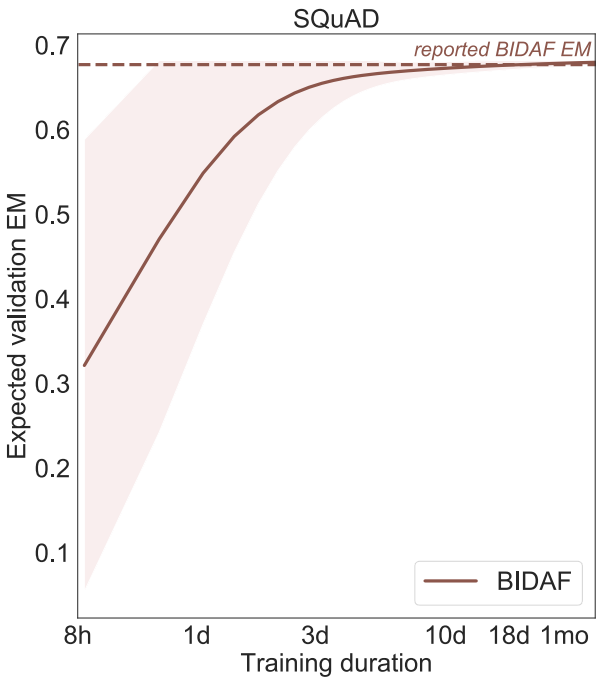


Figure 4: Comparing reported development exact-match score of BIDAF (dashed line) on SQuAD to expected performance of the best model with varying computational budgets (solid line). The shaded area represents the expected performance $\pm 1$ standard deviation, within the observed range of values. It takes about 18 days (55 hyperparameter trials) for the expected performance to match the reported results.

Our results (Fig. 3) show that the different models require different budgets to reach their reported performance in expectation, ranging from 2 (n-gram) to 20 (DGEM). Moreover, providing a large budget for each approach improves performance substantially over reported numbers. Finally, under different computation budgets, the top performing model changes (though the neural models are similar).

**SQuAD** Next, we turn our attention to SQuAD (Rajpurkar et al., 2016) and report performance of the commonly-used BiDAF model (Seo et al., 2017). The set of hyperparameters we tune covers those mentioned in addition to standard choices (details in Appendix D). We see in Fig. 4 that we require a budget of 18 GPU days in order for the expected maximum validation performance to match the value reported in the original paper. This suggests that some combination of prior intuition and extensive hyperparameter tuning were used by the original authors, though neither were reported.

✓ **For all reported experimental results**

- ☐ Description of computing infrastructure
- ☐ Average runtime for each approach
- ☐ Details of train/validation/test splits
- ☐ Corresponding validation performance for each reported test result
- ☐ A link to implemented code

✓ **For experiments with hyperparameter search**

- ☐ Bounds for each hyperparameter
- ☐ Hyperparameter configurations for best-performing models
- ☐ Number of hyperparameter search trials
- ☐ The method of choosing hyperparameter values (e.g., uniform sampling, manual tuning, etc.) and the criterion used to select among them (e.g., accuracy)
- ☐ Expected validation performance, as introduced in §3.1, or another measure of the mean and variance as a function of the number of hyperparameter trials.

Text Box 1: Experimental results checklist.

## 5 Recommendations

**Experimental results checklist** The findings discussed in this paper and other similar efforts highlight methodological problems in experimental NLP. In this section we provide a checklist to encourage researchers to report more comprehensive experimentation results. Our list, shown in Text Box 1, builds on the reproducibility checklist that was introduced for the machine learning community during NeurIPS 2018 (which is required to be filled out for each NeurIPS 2019 submission; Pineau, 2019).

Our focus is on improved reporting of experimental results, thus we include relevant points from their list in addition to our own. Similar to other calls for improved reporting in machine learning (Mitchell et al., 2019; Gebru et al., 2018), we recommend pairing experimental results with the information from this checklist in a structured format (see examples provided in Appendix A).

**EMNLP 2018 checklist coverage.** To estimate how commonly this information is reported in the NLP community, we sample fifty random EMNLP 2018 papers that include experimental results and evaluate how well they conform to our proposed reporting guidelines. We find that none of the papers reported all of the items in our checklist. However, every paper reported at least one item in the checklist, and each item is reported by at

least one paper. Of the papers we analyzed, 74% reported at least some of the best hyperparameter assignments. By contrast, 10% or fewer papers reported hyperparameter search bounds, the number of hyperparameter evaluation trials, or measures of central tendency and variation. We include the full results of this analysis in Table 1 in the Appendix.

**Comparisons with different budgets.** We have argued that claims about relative model performance should be qualified by computational expense. With varying amounts of computation, not all claims about superiority are valid. If two models have similar budgets, we can claim one outperforms the other (with that budget). Similarly, if a model with a small budget outperforms a model with a large budget, increasing the small budget will not change this conclusion. However, if a model with a large budget outperforms a model with a small budget, the difference might be due to the model or the budget (or both). As a concrete example, Melis et al. (2018) report the performance of an LSTM on language modeling the Penn Treebank after 1,500 rounds of Bayesian optimization; if we compare to a new $\mathcal{M}$ with a smaller budget, we can only draw a conclusion if the new model outperforms the LSTM.[14]

In a larger sense, there may be no simple way to make a comparison "fair." For example, the two models in Fig. 1 have hyperparameter spaces that are different, so fixing the same number of hyperparameter trials for both models does not imply a fair comparison. In practice, it is often not possible to measure how much past human experience has contributed to reducing the hyperparameter bounds for popular models, and there might not be a way to account for the fact that better understood (or more common) models can have better spaces to optimize over. Further, the cost of one application of $\mathcal{A}$ might be quite different depending on the model family. Converting to runtime is one possible solution, but implementation effort could still affect comparisons at a fixed $x$-value. Because of these considerations, our focus is on reporting whatever experimental results exist.

## 6 Discussion: Reproducibility

In NLP, the use of standardized test sets and public leaderboards (which limit test evaluations) has helped to mitigate the so-called "replication crisis" happening in fields such as psychology and medicine (Ioannidis, 2005; Gelman and Loken, 2014). Unfortunately, leaderboards can create additional reproducibility issues (Rogers, 2019). First, leaderboards obscure the budget that was used to tune hyperparameters, and thus the amount of work required to apply a model to a new dataset. Second, comparing to a model on a leaderboard is difficult if they *only* report test scores. For example, on the GLUE benchmark (Wang et al., 2018), the differences in *test set* performance between the top performing models can be on the order of a tenth of a percent, while the difference between test and validation performance might be one percent or larger. Verifying that a new implementation matches established performance requires submitting to the leaderboard, wasting test evaluations. Thus, we recommend leaderboards report validation performance for models evaluated on test sets.

As an example, consider Devlin et al. (2019), which introduced BERT and reported state-of-the-art results on the GLUE benchmark. The authors provide some details about the experimental setup, but do not report a specific budget. Subsequent work which extended BERT (Phang et al., 2018) included distributions of validation results, and we highlight this as a positive example of how to report experimental results. To achieve comparable test performance to Devlin et al. (2019), the authors report the best of twenty or one hundred random initializations. Their validation performance reporting not only illuminates the budget required to fine-tune BERT on such tasks, but also gives other practitioners results against which they can compare without submitting to the leaderboard.

## 7 Related Work

Lipton and Steinhardt (2018) address a number of problems with the practice of machine learning, including incorrectly attributing empirical gains to modeling choices when they came from other sources such as hyperparameter tuning. Sculley et al. (2018) list examples of similar evaluation issues, and suggest encouraging stronger standards for empirical evaluation. They recommend detailing experimental results found throughout the research process in a time-stamped document, as is done in other experimental science fields. Our work formalizes these issues and provides an ac-

[14] This is similar to controlling for the amount of training data, which is an established norm in NLP research.

tionable set of recommendations to address them.

Reproducibility issues relating to standard data splits (Schwartz et al., 2011; Gorman and Bedrick, 2019; Recht et al., 2019a,b) have surfaced in a number of areas. Shuffling standard training, validation, and test set splits led to a drop in performance, and in a number of cases the inability to reproduce rankings of models. Dror et al. (2017) studied reproducibility in the context of consistency among multiple comparisons.

Limited community standards exist for documenting datasets and models. To address this, Gebru et al. (2018) recommend pairing new datasets with a "datasheet" which includes information such as how the data was collected, how it was cleaned, and the motivation behind building the dataset. Similarly, Mitchell et al. (2019) advocate for including a "model card" with trained models which document training data, model assumptions, and intended use, among other things. Our recommendations in §5 are meant to document relevant information for experimental results.

## 8 Conclusion

We have shown how current practice in experimental NLP fails to support a simple standard of reproducibility. We introduce a new technique for estimating the expected validation performance of a method, as a function of computation budget, and present a set of recommendations for reporting experimental findings.

## Acknowledgments

This work was completed while the first author was an intern at the Allen Institute for Artificial Intelligence. The authors thank Kevin Jamieson, Samuel Ainsworth, and the anonymous reviewers for helpful feedback.

## References

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *JMLR*, 13:281–305.

Qian Chen, Xiao-Dan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for natural language inference. In *Proc. of ACL*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. of NAACL*.

Rotem Dror, Gili Baumer, Marina Bogomolov, and Roi Reichart. 2017. Replicability analysis for natural language processing: Testing significance with multiple datasets. *TACL*, 5:471–486.

Bradley Efron and Robert Tibshirani. 1994. *An Introduction to the Bootstrap*. CRC Press.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew E. Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. In *Proc. of NLP-OSS*.

Timnit Gebru, Jamie H. Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna M. Wallach, Hal Daumé, and Kate Crawford. 2018. Datasheets for datasets. arXiv:1803.09010.

Andrew Gelman and Eric Loken. 2014. The statistical crisis in science. *American Scientist*, 102:460.

Kyle Gorman and Steven Bedrick. 2019. We need to talk about standard splits. In *Proc. of ACL*.

Odd Erik Gundersen and Sigbjrn Kjensmo. 2018. State of the art: Reproducibility in artificial intelligence. In *Proc. of AAAI*.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. 2018. Deep reinforcement learning that matters. In *Proc. of AAAI*.

John P. A. Ioannidis. 2005. Why most published research findings are false. *PLoS Med*, 2(8).

Tushar Khot, Ashutosh Sabharwal, and Peter Clark. 2018. SciTaiL: A textual entailment dataset from science question answering. In *Proc. of AAAI*.

Liam Li and Ameet Talwalkar. 2019. Random search and reproducibility for neural architecture search. In *Proc. of UAI*.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. 2017. Hyperband: Bandit-based configuration evaluation for hyperparameter optimization. In *Proc. of ICLR*.

Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E Gonzalez, and Ion Stoica. 2018. Tune: A research platform for distributed model selection and training. In *Proc. of the ICML Workshop on AutoML*.

Zachary C. Lipton and Jacob Steinhardt. 2018. Troubling trends in machine learning scholarship. arXiv:1807.03341.

Mario Lucic, Karol Kurach, Marcin Michalski, Olivier Bousquet, and Sylvain Gelly. 2018. Are GANs created equal? A large-scale study. In *Proc. of NeurIPS*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Proc. of NeurIPS*.

Gábor Melis, Chris Dyer, and Phil Blunsom. 2018. On the state of the art of evaluation in neural language models. In *Proc. of EMNLP*.

Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. 2019. Model cards for model reporting. In *Proc. of FAT**.

Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proc. of EMNLP*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proc. of EMNLP*.

Matthew Peters, Sebastian Ruder, and Noah A. Smith. 2019. To tune or not to tune? Adapting pretrained representations to diverse tasks. In *Proc. of the RepL4NLP Workshop at ACL*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke S. Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Jason Phang, Thibault Févry, and Samuel R. Bowman. 2018. Sentence encoders on STILTs: Supplementary training on intermediate labeled-data tasks. arXiv:1811.01088.

Joelle Pineau. 2019. Machine learning reproducibility checklist. `https://www.cs.mcgill.ca/~jpineau/ReproducibilityChecklist.pdf`. Accessed: 2019-5-14.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proc. of EMNLP*.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. 2019a. Do CIFAR-10 classifiers generalize to CIFAR-10? arXiv:1806.00451.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. 2019b. Do ImageNet classifiers generalize to ImageNet? In *Proc. of ICML*.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of LSTM-networks for sequence tagging. In *Proc. of EMNLP*.

Anna Rogers. 2019. How the transformers broke NLP leaderboards. `https://hackingsemantics.xyz/2019/leaderboards/`. Accessed: 2019-8-29.

Roy Schwartz, Omri Abend, Roi Reichart, and Ari Rappoport. 2011. Neutralizing linguistically problematic annotations in unsupervised dependency parsing evaluation. In *Proc. of ACL*.

Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. 2019. Green AI. arXiv:1907.10597.

D. Sculley, Jasper Snoek, Ali Rahimi, and Alex Wiltschko. 2018. Winner's curse? On pace, progress, and empirical rigor. In *Proc. of ICLR (Workshop Track)*.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *Proc. of ICLR*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. of EMNLP*.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proc. of ACL*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proc. of ICLR*.

Dani Yogatama and Noah A. Smith. 2015. Bayesian optimization of text representations. In *Proc. of EMNLP*.

Ye Zhang and Byron Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. arXiv:1510.03820.

# A Deep Factorization of Style and Structure in Fonts

**Akshay Srivatsan**[1] **Jonathan T. Barron**[2] **Dan Klein**[3] **Taylor Berg-Kirkpatrick**[4]

[1]Language Technologies Institute, Carnegie Mellon University, asrivats@cs.cmu.edu
[2]Google Research, barron@google.com
[3]Computer Science Division, University of California, Berkeley, klein@cs.berkeley.edu
[4]Computer Science and Engineering, University of California, San Diego, tberg@eng.ucsd.edu

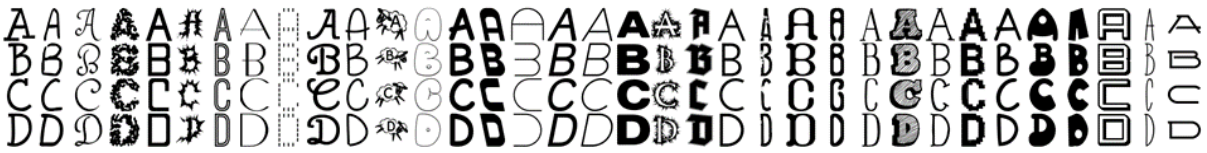

Figure 1: Example fonts from the Capitals64 dataset. The task of font reconstruction involves generating missing glyphs from partially observed novel fonts.

## Abstract

We propose a deep factorization model for typographic analysis that disentangles *content* from *style*. Specifically, a variational inference procedure factors each training glyph into the combination of a character-specific content embedding and a latent font-specific style variable. The underlying generative model combines these factors through an asymmetric transpose convolutional process to generate the image of the glyph itself. When trained on corpora of fonts, our model learns a manifold over font styles that can be used to analyze or reconstruct new, unseen fonts. On the task of reconstructing missing glyphs from an unknown font given only a small number of observations, our model outperforms both a strong nearest neighbors baseline and a state-of-the-art discriminative model from prior work.

## 1 Introduction

One of the most visible attributes of digital language data is its typography. A font makes use of unique stylistic features in a visually consistent manner across a broad set of characters while preserving the structure of each underlying form in order to be human readable – as shown in Figure 1. Modeling these stylistic attributes and how they compose with underlying character structure could aid typographic analysis and even allow for automatic generation of novel fonts. Further, the variability of these stylistic features presents a challenge for optical character recognition systems, which typically presume a library of known fonts. In the case of historical document recognition, for example, this problem is more pronounced due to the wide range of lost, ancestral fonts present in such data (Berg-Kirkpatrick et al., 2013; Berg-Kirkpatrick and Klein, 2014). Models that capture this wide stylistic variation of glyph images may eventually be useful for improving optical character recognition on unknown fonts.

In this work we present a generative model capable of disentangling stylistic features of fonts from the underlying structure of each character. Our model represents the style of each font as a vector-valued latent variable, and parameterizes the structure of each character as a learned embedding. Critically, each style latent variable is shared by all characters within a font, while character embeddings are shared by characters of the same type across all fonts. Thus, our model is related to deep matrix factorization techniques (Xue et al., 2017). While some prior work on font manifolds (Campbell and Kautz, 2014) relies on outline representations of observed glyphs, our approach directly models pixel values in rasterized glyph representations and allows us to more easily generalize to fonts with variable glyph topologies.

Inspired by modern neural methods' ability to disentangle loosely coupled phenomena in other domains, including both language and vision (Hu et al., 2017; Yang et al., 2017; Gatys et al., 2016; Zhu et al., 2017), we parameterize the distribution that combines style and structure in order to gener-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2195–2205, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

ate glyph images as a transpose convolutional neural decoder (Dumoulin and Visin, 2016). Further, the decoder is fed character embeddings early on in the process, while the font latent variables directly parameterize the convolution filters. This architecture biases the model to capture the asymmetric process by which structure and style combine to produce an observed glyph.

We evaluate our learned representations on the task of *font reconstruction*. After being trained on a set of observed fonts, the system reconstructs missing glyphs in a set of previously unseen fonts, conditioned on a small observed subset of glyph images. In contrast with past work on this task (Azadi et al., 2018), our model is generative and font reconstruction is performed via posterior inference. Since our model's posterior is intractable, we demonstrate how a variational inference procedure can be used to perform both learning and accurate font reconstruction. In experiments, we find that our proposed latent variable model is able to substantially outperform both a strong nearest-neighbors baseline as well as a state-of-the-art discriminative system on a standard dataset for font reconstruction. Further, in qualitative analysis, we demonstrate how the learned latent space can be used to interpolate between fonts, hinting at the practicality of more creative applications.

Finally, while we focus our evaluation on font reconstruction, our approach has an important relationship with style transfer – the goal of our analysis is to learn a smooth manifold of font styles that allows for stylistic inference given a small sample of glyphs. Many style transfer tasks in the language domain (Shen et al., 2017) suffer from ambiguity surrounding the underlying division between style and semantic content. By contrast, in this setting the distinction is clearly defined, with content (i.e. the character) observed as a categorical label denoting the coarse overall shape of a glyph, and style (i.e. the font) explaining lower-level visual features such as boldness, texture, and serifs. The modeling approach taken here might inform work on more complex domains where the division is less clear.

## 2 Font Reconstruction

We can view a collection of fonts as a matrix, $X$, where each column corresponds to a particular character type, and each row corresponds to a specific font. Each entry in the matrix, $x_{ij}$, is an image of the glyph for character $i$ in the style of a font $j$, which we observe as a $64 \times 64$ grayscale image as shown in Figure 1. In a real world setting, the equivalent matrix would naturally have missing entries wherever the encoding for a character type in a font is undefined. In general, not all fonts contain renderings of all possible character types; many will only support one particular language or alphabet and leave out uncommon symbols. Further, for many commercial applications, only the small subset of characters that appears in a specific advertisement or promotional message will have been designed by the artist – the majority of glyphs are missing. As a result, we may wish to have models that can infer these missing glyphs, a task referred to as font reconstruction.

Following past work (Azadi et al., 2018), we define the task setup as follows: During training we have access to a large collection of observed fonts for the complete character set. At test time we are required to predict the missing glyph images in a collection of *previously unseen* fonts with the same character set. Each test font will contain observable glyph images for a small randomized subset of the character set. Based on the style of this subset, the model must reconstruct glyphs for the rest of the character set.

Font reconstruction can be thought of as a form of matrix completion; given various observations in both a particular row and column, we wish to reconstruct the element at their intersection. Alternatively we can view it as a few-shot style transfer task, in that we want to apply the characteristic attributes of a new font (e.g. serifs, italicization, drop-shadow) to a letter using a small number of examples to infer those attributes. Past work on font reconstruction has focused on discriminative techniques. For example Azadi et al. (2018) used an adversarial network to directly predict held out glyphs conditioned on observed glyphs. By contrast, we propose a generative approach using a deep latent variable model. Under our approach fonts are generated based on an unobserved style embedding, which we can perform inference over given any number of observations.

## 3 Model

Figure 2 depicts our model's generative process. Given a collection of images of glyphs consisting of $I$ character types across $J$ fonts, our model hy-

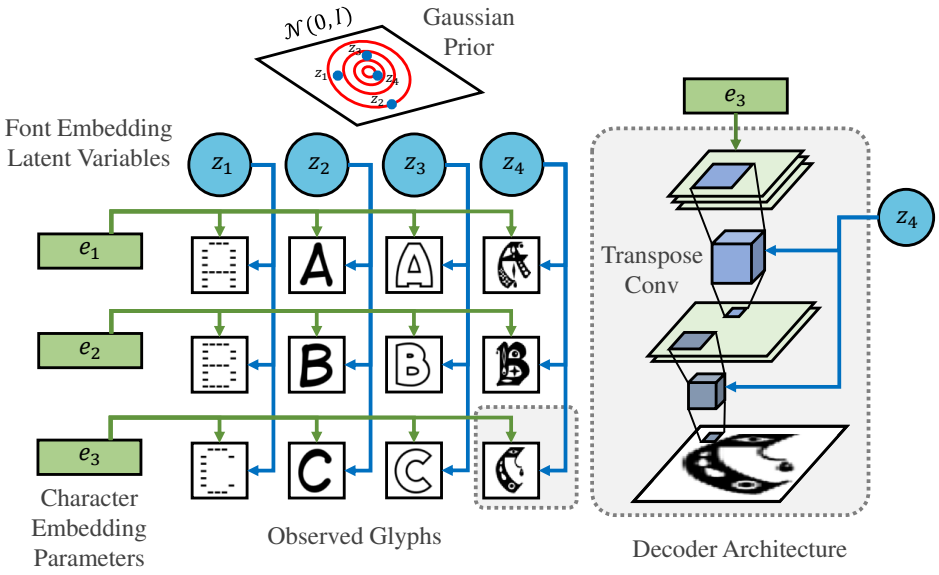


Figure 2: Depiction of the generative process of our model. Each observed glyph image is generated conditioned on the latent variable of the corresponding font and the embedding parameter of the corresponding character type. For a more detailed description of the decoder architecture and hyperparameters, see Appendix A.

pothesizes a separation of character-specific structural attributes and font-specific stylistic attributes into two different representations. Since all characters are observed in at least one font, each character type is represented as an embedding vector which is part of the model's parameterization. In contrast, only a subset of fonts is observed during training and our model will be expected to generalize to reconstructing unseen fonts at test time. Thus, our representation of each font is treated as a vector-valued latent variable rather than a deterministic embedding.

More specifically, for each font in the collection, a font embedding variable, $z_j \in \mathbb{R}^k$, is sampled from a fixed multivariate Gaussian prior, $p(z_j) = \mathcal{N}(0, I_k)$. Next, each glyph image, $x_{ij}$, is generated independently, conditioned on the corresponding font variable, $z_j$, and a character-specific parameter vector, $e_i \in \mathbb{R}^k$, which we refer to as a character embedding. Thus, glyphs of the same character type share a character embedding, while glyphs of the same font share a font variable. A corpus of $I * J$ glyphs is modeled with only $I$ character embeddings and $J$ font variables, as seen in the left half of Figure 2. This modeling approach can be thought of as a form of deep matrix factorization, where the content at any given cell is purely a function of the vector representations of the corresponding row and column. We denote the full corpus of glyphs as a matrix $X = ((x_{11}, ..., x_{1J}), ..., (x_{I1}, ...x_{IJ}))$ and denote the corresponding character embeddings as $E = (e_1, ..., e_I)$ and font variables as $Z = (z_1, ..., z_J)$.

Under our model, the probability distribution over each image, conditioned on $z_j$ and $e_i$, is parameterized by a neural network, described in the next section and depicted in Figure 2. We denote this decoder distribution as $p(x_{ij}|z_j; e_i, \phi)$, and let $\phi$ represent parameters, shared by all glyphs, that govern how font variables and character embeddings combine to produce glyph images. In contrast, the character embedding parameters, $e_i$, which feed into the decoder, are only shared by glyphs of the same character type. The font variables, $z_j$, are unobserved during training and will be inferred at test time. The joint probability under our model is given by:

$$p(X, Z; E, \phi) = \prod_{i,j} p(x_{ij}|z_j; e_i, \phi)p(z_j)$$

### 3.1 Decoder Architecture

One way to encourage the model to learn disentangled representations of style and content is by choosing an architecture that introduces helpful inductive bias. For this domain, we can think of the character type as specifying the overall shape of the image, and the font style as influencing the finer details; we formulate our decoder with this difference in mind. We hypothesize that a glyph

can be modeled in terms of a low-resolution character representation to which a complex operator specific to that font has been applied.

The success of transpose convolutional layers at generating realistic images suggests a natural way to apply this intuition. A transpose convolution[1] is a convolution performed on an undecimated input (i.e. with zeros inserted in between pixels in alternating rows and columns), resulting in an upscaled output. Transpose convolutional architectures generally start with a low resolution input which is passed through several such layers, iteratively increasing the resolution and decreasing the number of channels until the final output dimensions are reached. We note that the asymmetry between the coarse input and the convolutional filters closely aligns with the desired inductive biases, and therefore use this framework as a starting point for our architecture.

Broadly speaking, our architecture represents the underlying shape that defines the specific character type (but not the font) as coarse-grained information that therefore enters the transpose convolutional process early on. In contrast, the stylistic content that specifies attributes of the specific font (such as serifs, drop shadow, texture) is represented as finer-grained information that enters into the decoder at a later stage, by parameterizing filters, as shown in the right half of Figure 2. Specifically we form our decoder as follows: first the character embedding is projected to a low resolution matrix with a large number of channels. Following that, we apply several transpose convolutional layers which increase the resolution, and reduce the number of channels. Critically, the convolutional filter at each step is not a learned parameter of the model, but rather the output of a small multilayer perceptron whose input is the font latent variable $z$. Between these transpose convolutions, we insert vanilla convolutional layers to fine-tune following the increase in resolution.

Overall, the decoder consists of four blocks, where each block contains a transpose convolution, which upscales the previous layer and reduces the number of channels by a factor of two, followed by two convolutional layers. Each (transpose) convolution is followed by an instance norm and a ReLU activation. The convolution filters all have a kernel size of $5 \times 5$. The character embedding is reshaped via a two-layer MLP into a $8 \times 8 \times 256$ tensor before being fed into the decoder. The final $64 \times 64$ dimensional output layer is treated as a grid of parameters which defines the output distribution on pixels. We describe the specifics of this distribution in the next section.

[1] sometimes erroneously referred to as a "deconvolution"

### 3.2 Projected Loss

The conventional approach for computing loss on image observations is to use an independent output distribution, typically a Gaussian, on each pixel's intensity. However, deliberate analysis of the statistics of natural images has shown that images are not well-described in terms of statistically independent pixels, but are instead better modeled in terms of edges (Field, 1987; Huang and Mumford, 1999). It has also been demonstrated that images of text have similar statistical distributions as natural images (Melmer et al., 2013). Following this insight, as our reconstruction loss we use a heavy-tailed (leptokurtotic) distribution placed on a transformed representation of the image, similar to the approach of Barron (2019). Modeling the statistics of font glyphs in this fashion results in sharper samples, while modeling independent pixels with a Gaussian distribution results in blurry, oversmoothed results.

More specifically, we adopt one of the strategies employed in Barron (2019), and transform image observations using the orthonormal variant of the 2-Dimensional Discrete Cosine Transform (2-D DCT-II) (Ahmed et al., 1974), which we denote as $f : \mathbb{R}^{64\times64} \rightarrow \mathbb{R}^{64\times64}$ for our $64 \times 64$ dimensional image observations. We transform both the observed glyph image and the corresponding grid or parameters produced by our decoder before computing the observation's likelihood.

This procedure projects observed images onto a grid of orthogonal bases comprised of shifted and scaled 2-dimensional cosine functions. Because the DCT-II is orthonormal, this transformation is volume-preserving, and so likelihoods computed in the projected space correspond to valid measurements in the original pixel domain.

Note that because the DCT-II is simply a rotation in our vector space, imposing a normal distribution in this transformed space should have little effect (ignoring the scaling induced by the diagonal of the covariance matrix of the Gaussian distribution) as Euclidean distance is preserved under rotations. For this reason we impose a heavy-tailed distribution in this transformed space,

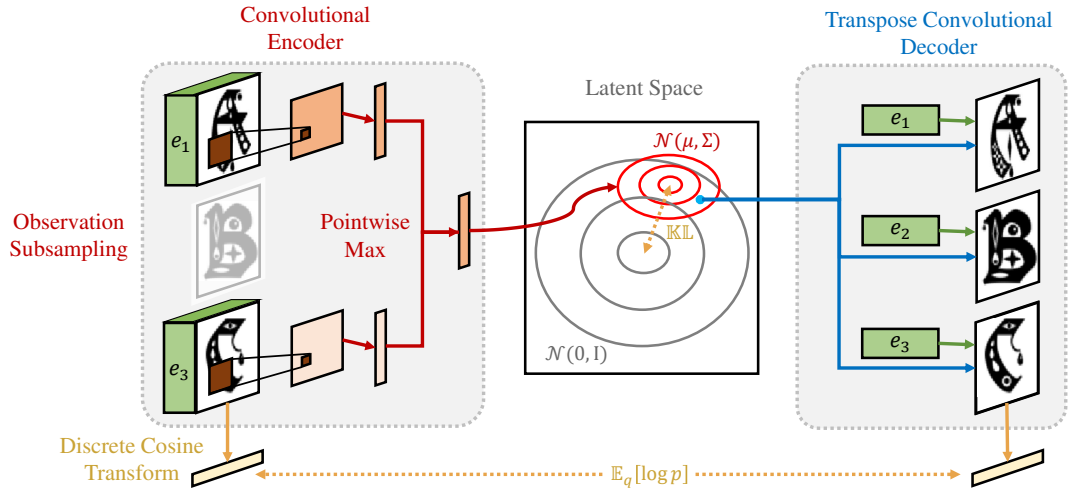


Figure 3: Depiction of the computation graph of the amortized variational lower bound (for simplicity, only one font is shown). The encoder approximates the generative model's true posterior over the font style latent variables given the observations. It remains insensitive to the number of observations by pooling high-level features across glyphs. For a more specific description of the encoder architecture details, see Appendix A.

specifically a Cauchy distribution. This gives the following probability density function

$$g(x; \hat{x}, \gamma) = \frac{1}{\pi\gamma \left(1 + \left(\frac{f(x) - f(\hat{x})}{\gamma}\right)^2\right)}$$

where $x$ is an observed glyph, $\hat{x}$ is the location parameter grid output by our decoder, and $\gamma$ is a hyperparameter which we set to $\gamma = 0.001$.

The Cauchy distribution accurately captures the heavy-tailed structure of the edges in natural images. Intuitively, it models the fact that images tend to be mostly smooth, with a small amount of non-smooth variation in the form of edges. Computing this heavy-tailed loss over the frequency decomposition provided by the DCT-II instead of the raw pixel values encourages the decoder to generate sharper images without needing either an adversarial discriminator or a vectorized representation of the characters during training. Note that while our training loss is computed in DCT-II space, at test time we treat the raw grid of parameter outputs $\hat{x}$ as the glyph reconstruction.

## 4 Learning and Inference

Note that in our training setting, the font variables $Z$ are completely unobserved, and we must induce their manifold with learning. As our model is generative, we wish to optimize the marginal probability of just the observed $X$ with respect to the model parameters $E$ and $\phi$:

$$p(X; E, \phi) = \int_Z p(X, Z; E, \phi) dZ$$

However, the integral over $Z$ is computationally intractable, given that the complex relationship between $Z$ and $X$ does not permit a closed form solution. Related latent variable models such as Variational Autoencoders (VAE) (Kingma and Welling, 2014) with intractable marginals have successfully performed learning by optimizing a variational lower bound on the log marginal likelihood. This surrogate objective, called the evidence lower bound (ELBO), introduces a variational approximation, $q(Z|X) = \prod_j q(z_i|x_{1j}, \ldots, x_{Ij})$ to the model's true posterior, $p(Z|X)$. Our model's ELBO is as follows:

$$\begin{aligned}\text{ELBO} = \sum_j \mathbb{E}_q[\log p(x_{1j}, \ldots, x_{Ij}|z_j)] \\ - \mathbb{KL}(q(z_j|x_{1j}, \ldots, x_{Ij})||p(z_j))\end{aligned}$$

where the approximation $q$ is parameterized via a neural encoder network. This lower bound can be optimized by stochastic gradient ascent if $q$ is a Gaussian, via the reparameterization trick described in (Kingma and Welling, 2014; Rezende et al., 2014) to sample the expectation under $q$ while still permitting backpropagation.

Practically speaking, a key property which we desire is the ability to perform consistent inference over $z$ given a variable number of observed glyphs

in a font. We address this in two ways: through the architecture of our encoder, and through a special masking process in training; both of which are shown in Figure 3.

### 4.1 Posterior Approximation

**Observation Subsampling:** To get reconstructions from only partially observed fonts at test time, the encoder network must be able to infer $z_j$ from any subset of $(x_{1j}, \ldots, x_{Ij})$. One approach for achieving robustness to the number of observations is through the training procedure. Specifically when computing the approximate posterior for a particular font in our training corpus, we mask out a randomly selected subset of the characters before passing them to the encoder. This incentivizes the encoder to produce reasonable estimates without becoming too reliant on the features extracted from any one particular character, which more closely matches the setup at test time.

**Encoder Architecture:** Another way to encourage this robustness is through inductive bias in the encoder architecture. Specifically we use a convolutional neural network which takes in a batch of characters from a single font, concatenated with their respective character type embedding. Following the final convolutional layer, we perform an elementwise max operation across the batch, reducing to a single vector representation for the entire font which we pass through further fully-connected layers to obtain the output parameters of $q$ as shown in Figure 3. By including this accumulation across the elements of the batch, we combine the features obtained from each character in a manner that is largely invariant to the total number and types of characters observed. This provides an inductive bias that encourages the model to extract similar features from each character type, which should therefore represent stylistic as opposed to structural properties.

Overall, the encoder over each glyph consists of three blocks, where each block consists of a convolution followed by a max pool with a stride of two, an instance norm (Ulyanov et al., 2016), and a ReLU. The activations are then pooled across the characters via an elementwise max into a single vector, which is then passed through four fully-connected layers, before predicting the parameters of the Gaussian approximate posterior.

**Reconstruction via Inference:** At test time, we pass an observed subset of a new font to our encoder in order to estimate the posterior over $z_j$, and take the mean of that distribution as the inferred font representation. We then pass this encoding to the decoder along with the full set of character embeddings $E$ in order to produce reconstructions of every glyph in the font.

## 5 Experiments

We now provide an overview of the specifics of the dataset and training procedure, and describe our experimental setup and baselines.

### 5.1 Data

We compare our model against baseline systems at font reconstruction on the Capitals64 dataset (Azadi et al., 2018), which contains the 26 capital letters of the English alphabet as grayscale $64 \times 64$ pixel images across $10,682$ fonts. These are broken down into training, dev, and test splits of $7649$, $1473$, and $1560$ fonts respectively.

Upon manual inspection of the dataset, it is apparent that several fonts have an almost visually indistinguishable nearest neighbor, making the reconstruction task trivial using a naive algorithm (or an overfit model with high capacity) for those particular font archetypes. Because these datapoints are less informative with respect to a model's ability to generalize to previously *unseen* styles, we additionally evaluate on a second test set designed to avoid this redundancy. Specifically, we choose the $10\%$ of test fonts that have maximal $L_2$ distance from their closest equivalent in the training set, which we call "Test Hard".

### 5.2 Baselines

As stated previously, many fonts fall into visually similar archetypes. Based on this property, we use a nearest neighbors algorithm for our first baseline. Given a partially observed font at test time, this approach simply "reconstructs" by searching the training set for the font with the lowest $L_2$ distance over the observed characters, and copy its glyphs verbatim for the missing characters.

For our second comparison, we use the GlyphNet model from Azadi et al. (2018). This approach is based on a generative adversarial network, which uses discriminators to encourage the model to generate outputs that are difficult to distinguish from those in the training set. We test from the publicly available epoch $400$ checkpoint, with modifications to the evaluation script

| | Test Full | | | | Test Hard | | | |
|---|---|---|---|---|---|---|---|---|
| Observations | 1 | 2 | 4 | 8 | 1 | 2 | 4 | 8 |
| NN | 483.13 | 424.49 | 386.81 | 363.97 | 880.22 | 814.67 | 761.29 | 735.18 |
| GlyphNet | 669.36 | 533.97 | 455.23 | 416.65 | 935.01 | 813.50 | 718.02 | 653.57 |
| Ours (FC) | 353.63 | 316.47 | 293.67 | 281.89 | **596.57** | 556.21 | 527.50 | 513.25 |
| Ours (Conv) | **352.07** | **300.46** | **271.03** | **254.92** | 615.87 | **556.03** | **511.05** | **489.58** |

Table 1: $L_2$ reconstruction per glyph by number of observed characters. "Full" includes the entire test set while "Hard" is measured only over the 10% of test fonts with the highest $L_2$ distance from the closest font in train.

to match the setup described above.

We also perform an ablation using fully-connected instead of convolutional layers. For more architecture details see Appendix A.

### 5.3 Training Details

We train our model to maximize the expected log likelihood using the Adam optimization algorithm (Kingma and Ba, 2015) with a step size of $10^{-5}$ (default settings otherwise), and perform early stopping based on the approximate log likelihood on a hard subset of dev selected by the process described earlier. To encourage robustness in the encoder, we randomly drop out glyphs during training with a probability of 0.7 (rejecting samples where all characters in a font are dropped). All experiments are run with a dimensionality of 32 for the character embeddings and font latent variables. Our implementation is built in PyTorch (Paszke et al., 2017) version `1.1.0`.

## 6 Results

We now present quantitative results from our experiments in both automated and human annotated metrics, and offer qualitative analysis of reconstructions and the learned font manifold.

### 6.1 Quantitative Evaluation

**Automatic Evaluation:** We show font reconstruction results for our system against nearest neighbors and GlyphNet in Table 1. Each model is given a random subsample of glyphs from each test font (we measure at 1, 2, 4, and 8 observed characters), with their character labels. We measure the average $L_2$ distance between the image reconstructions for the unobserved characters and the ground truth, after scaling intensities to $[0, 1]$.

Our system achieves the best performance for both the overall and hard subset of test for all numbers of observed glyphs. Nearest neighbors provides a strong baseline on the full test set, even outperforming GlyphNet. However it performs much worse on the hard subset. This makes sense as we expect nearest neighbors to do extremely well on any test fonts that have a close equivalent in train, but suffer in fidelity on less traditional styles. GlyphNet similarly performs worse on test hard, which could reflect the *missing modes* problem of GANs failing to capture the full diversity of the data distribution (Che et al., 2016; Tolstikhin et al., 2017). The fully-connected ablation is also competitive, although we see that the convolutional architecture is better able to infer style from larger numbers of observations. On the hard test set, the fully-connected network even outperforms the convolutional system when only one observation is present, perhaps indicating that its lower-capacity architecture better generalizes from very limited data.

**Human Evaluation:** To measure how consistent these perceptual differences are, we also perform a human evaluation of our model's reconstructions against GlyphNet using Amazon Mechanical Turk (AMT). In our setup, turkers were asked to compare the output of our model against the output of GlyphNet for a single font given one observed character, which they were also shown. Turkers selected a ranking based on which reconstructed font best matched the style of the observed character, and a separate ranking based on which was more realistic. On the full test set (1560 fonts, with 5 turkers assigned to each font), humans preferred our system over GlyphNet 81.3% and 81.8% of the time for style and realism respectively. We found that on average 76% of turkers shown a given pair of reconstructions selected the same ranking as each other on both criteria, suggesting high annotator agreement.

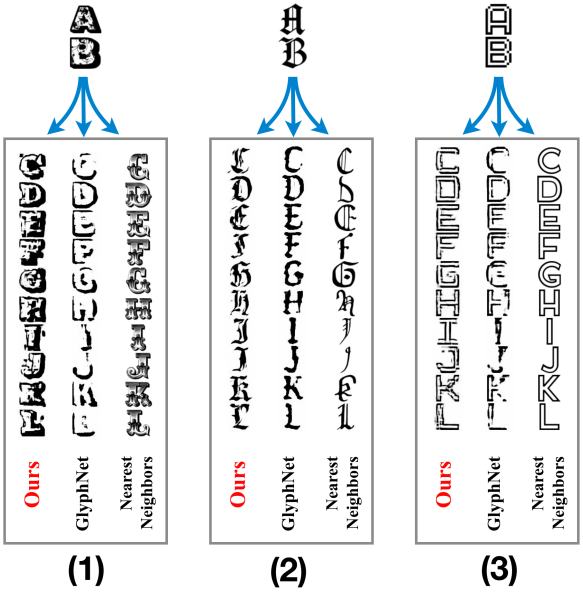


Figure 4: Reconstructions of partially observed fonts in the hard subset from our model, GlyphNet, and nearest neighbors. Given images of glyphs for 'A' and 'B' in each font, we visualize reconstructions of the remaining characters. Fonts are chosen such that the $L_2$ loss of our model on these examples closely matches the average loss over the full evaluation set.

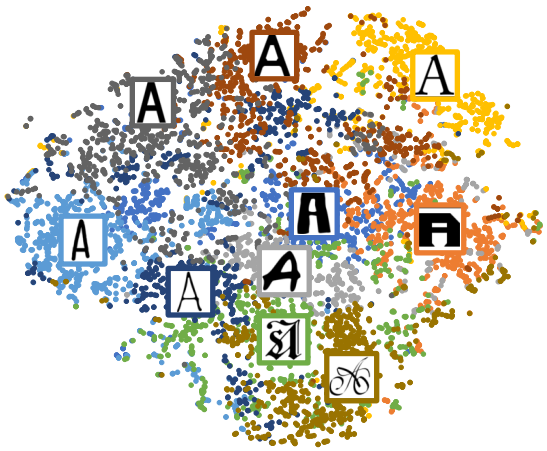

Figure 5: t-SNE projection of latent font variables inferred for the full training set, colored by k-means clustering with $k = 10$. The glyph for the letter "A" for each centroid is shown in overlay.

### 6.2 Qualitative Analysis

In order to fully understand the comparative behavior of these systems, we also qualitatively compare the reconstruction output of these systems to analyze their various failure modes, showing examples in Figure 4. We generally find that our approach tends to produce reconstructions that, while occasionally blurry at the edges, are generally faithful at reproducing the principal stylistic features of the font. For example, we see that for font (1) in Figure 4, we match not only the overall shape of the letters, but also the drop shadow and to an extent the texture within the lettering, while GlyphNet does not produce fully enclosed letters or match the texture. The output of nearest neighbors, while well-formed, does not respect the style of the font as closely as it fails to find a font in training that matches these stylistic properties. In font (2) the systems all produce a form of gothic lettering, but the output of GlyphNet is again lacking in certain details, and nearest neighbors makes subtle but noticeable changes to the shape of the letters. In the final example (3) we even see that our system appears to attempt to replicate the pixelated outline, while nearest neighbors ignores this subtlety. GlyphNet is in this case somewhat inconsistent, doing reasonably well on some letters, but much worse on others. Overall, nearest neighbors will necessarily output well-formed glyphs, but with lower fidelity to the style, particularly on more unique fonts. While GlyphNet does pick up on some subtle features, our model tends to produce the most coherent output on harder fonts.

Since our model attempts to learn a smooth manifold over the latent style, we can also perform interpolation between the inferred font representations, something which is not directly possible using either of the baselines. In this analysis, we take two fonts from the same font family, which differ along one property, and pass them through our encoder to obtain the latent font variable for each. We then interpolate between these values, passing the result at various steps into our decoder to produce new fonts that exists in between the observations. In Figure 6 we see how our model can apply serifs, italicization, and boldness gradually while leaving the font unchanged in other respects. This demonstrates that our manifold is smooth and interpretable, not just at the points corresponding to those in our dataset. This could be leveraged to modify existing fonts with respect to a particular attribute to generate novel fonts efficiently.

### 6.3 Analysis of Learned Manifold

Beyond looking at the quality of reconstructions, we also wish to analyze properties of the latent space learned by our model. To do this, we use our encoder to infer latent font variables $z$ for each of the fonts in our training data, and use

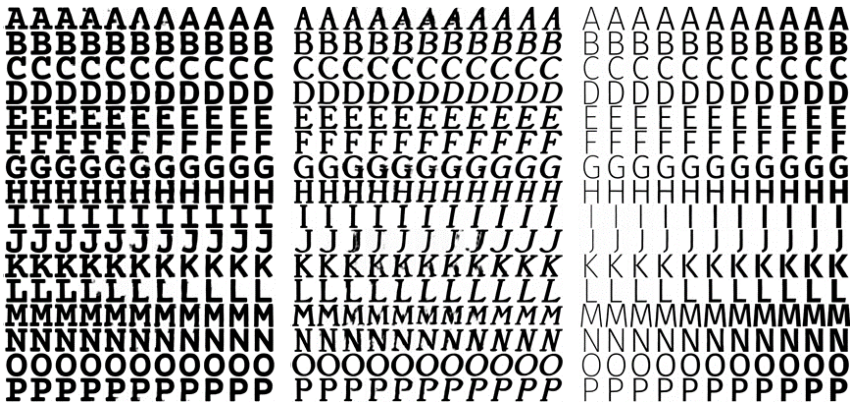

Figure 6: Interpolation between font variants from the same font family, showing smoothness of the latent manifold. Linear combinations of the embedded fonts correspond to outputs that lie intuitively "in between".

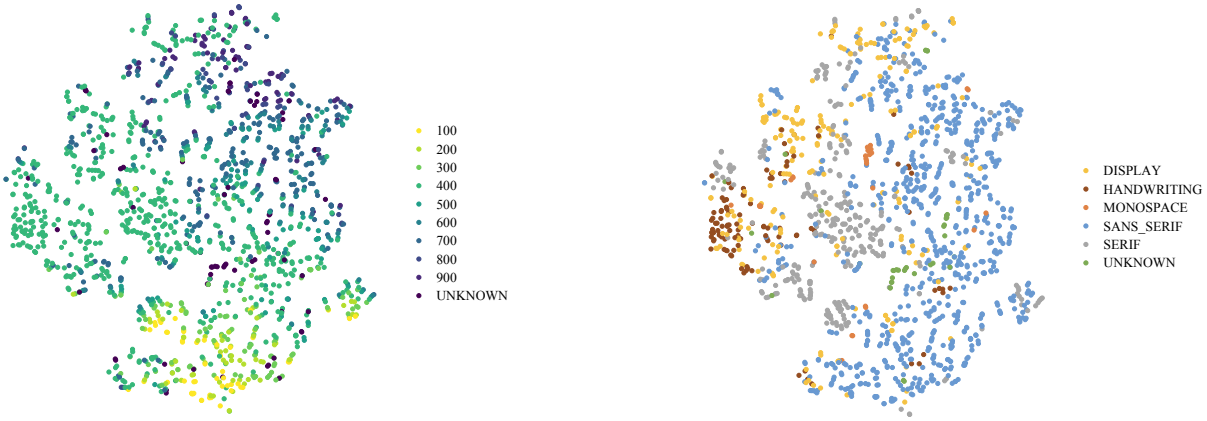


Figure 7: t-SNE projection of latent font variables inferred on Google Fonts, colored by weight and category.

a t-SNE projection (Van Der Maaten, 2013) to plot them in 2D, shown in Figure 5. Since the broader, long-distance groupings may not be preserved by this transformation, we perform a k-means clustering (Lloyd, 1982) with $k = 10$ on the high-dimensional representations to visualize these high-level groupings. Additionally, we display a sample glyph for the centroid for each cluster. We see that while the clusters are not completely separated, the centroids generally correspond to common font styles, and are largely adjacent to or overlapping with those with similar stylistic properties; for example script, handwritten, and gothic styles are very close together.

To analyze how well our latent embeddings correspond to human defined notions of font style, we also run our system on the Google Fonts [2] dataset which despite containing fewer fonts and less diversity in style, lists metadata including numerical weight and category (e.g. *serif*, *handwriting*, *monospace*). In Figure 7 we show t-SNE projections of latent font variables from our model trained on Google Fonts, colored accordingly. We see that the embeddings do generally cluster by weight as well as category suggesting that our model is learning latent information consistent with how humans perceive font style.

[2] https://github.com/google/fonts

## 7 Conclusion

We presented a latent variable model of glyphs which learns disentangled representations of the structural properties of underlying characters from stylistic features of the font. We evaluated our model on the task of font reconstruction and showed that it outperformed both a strong nearest neighbors baseline and prior work based on GANs especially for fonts highly dissimilar to any instance in the training set. In future work, it may be worth extending this model to learn a latent manifold on content as well as style, which could allow for reconstruction of previously unseen character types, or generalization to other domains where the notion of content is higher dimensional.

## Acknowledgments

This project is funded in part by the NSF under grant 1618044 and by the NEH under grant HAA-256044-17. Special thanks to Si Wu for help constructing the Google Fonts figures.

## References


Nasir Ahmed, T_ Natarajan, and Kamisetty R Rao. 1974. Discrete cosine transform. *IEEE Transactions on Computers*.

Samaneh Azadi, Matthew Fisher, Vladimir G Kim, Zhaowen Wang, Eli Shechtman, and Trevor Darrell. 2018. Multi-content GAN for few-shot font style transfer. *CVPR*.

Jonathan T. Barron. 2019. A general and adaptive robust loss function. *CVPR*.

Taylor Berg-Kirkpatrick, Greg Durrett, and Dan Klein. 2013. Unsupervised transcription of historical documents. *ACL*.

Taylor Berg-Kirkpatrick and Dan Klein. 2014. Improved typesetting models for historical OCR. *ACL*.

Neill DF Campbell and Jan Kautz. 2014. Learning a manifold of fonts. *ACM TOG*.

Tong Che, Yanran Li, Athul Paul Jacob, Yoshua Bengio, and Wenjie Li. 2016. Mode regularized generative adversarial networks. *arXiv preprint arXiv:1612.02136*.

Vincent Dumoulin and Francesco Visin. 2016. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*.

David J. Field. 1987. Relations between the statistics of natural images and the response properties of cortical cells. *JOSA A*.

Leon A Gatys, Alexander S Ecker, and Matthias Bethge. 2016. Image style transfer using convolutional neural networks. *CVPR*.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Controllable text generation. *arXiv preprint arXiv:1703.00955*, 7.

Jinggang Huang and David Mumford. 1999. Statistics of natural images and models. *CVPR*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *ICLR*.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. *ICLR*.

Stuart Lloyd. 1982. Least squares quantization in pcm. *IEEE Transactions on Information Theory*.

Tamara Melmer, Seyed Ali Amirshahi, Michael Koch, Joachim Denzler, and Christoph Redies. 2013. From regular text to artistic writing and artworks: Fourier statistics of images with low and high aesthetic appeal. *Frontiers in Human Neuroscience*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. *ICML*.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. *NIPS*.

Ilya O Tolstikhin, Sylvain Gelly, Olivier Bousquet, Carl-Johann Simon-Gabriel, and Bernhard Schölkopf. 2017. Adagan: Boosting generative models. In *Advances in Neural Information Processing Systems*, pages 5424–5433.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. 2016. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*.

Laurens Van Der Maaten. 2013. Barnes-Hut-SNE. *arXiv preprint arXiv:1301.3342*.

Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. 2017. Deep matrix factorization models for recommender systems. *IJCAI*.

Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. 2017. Improved variational autoencoders for text modeling using dilated convolutions. *ICML*.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. *ICCV*.


# A Appendix

## A.1 Architecture Notation

We now provide further details on the specific architectures used to parameterize our model and inference network. The following abbreviations are used to represent various components:

- $\mathrm{F}_i$ : fully-connected layer with $i$ hidden units
- R : ReLU activation
- M : batch max pool
- S : $2 \times 2$ spatial max pool
- $\mathrm{C}_i$ : convolution filters with $i$ filters of $5 \times 5$, 2 pixel zero-padding, stride of 1, dilation of 1
- I : instance normalization
- $\mathrm{T}_{i,j,k}$ : transpose convolution with $i$ filters of $5 \times 5$, 2 pixel zero-padding, stride of $j$, $k$ pixel output padding, dilation of 1, where kernel and bias are the output of an MLP (described below)
- H : reshape to $26 \times 256 \times 8 \times 8$

### A.2 Network Architecture

Our fully-connected encoder is:
$F_{128}$-R-$F_{128}$-R-$F_{128}$-R-$F_{1024}$-R-M-$F_{128}$-R-$F_{128}$-R-$F_{128}$-R-$F_{64}$

Our convolutional encoder is:
$C_{64}$-S-I-R-$C_{128}$-S-I-R-$C_{256}$-S-I-R-$F_{1024}$-M-R-$F_{128}$-R-$F_{128}$-R-$F_{128}$-R-$F_{64}$

Our fully-connected decoder is:
$F_{128}$-R-$F_{128}$-R-$F_{128}$-R-$F_{128}$-R-$F_{128}$-R-$F_{4096}$

Our transpose convolutional decoder is:
$F_{128}$-R-$F_{16384}$-R-H-$T_{256,2,1}$-R-$C_{256}$-I-R-$C_{256}$-I-R-$T_{128,2,1}$-R-$C_{128}$-I-R-$C_{128}$-I-R-$T_{64,2,1}$-I-R-$C_{64}$-I-R-$C_{64}$-I-R-$T_{32,1,0}$-I-R-$C_{32}$-I-R-$C_1$

MLP to compute a transpose convolutional parameter of size $j$ is:
$F_{128}$-R-$F_j$

# Cross-lingual Semantic Specialization via Lexical Relation Induction

**Edoardo M. Ponti**[1], **Ivan Vulić**[1], **Goran Glavaš**[2], **Roi Reichart**[3], **Anna Korhonen**[1]
[1]Language Technology Lab, TAL, University of Cambridge
[2]Data and Web Science Group, University of Mannheim, Germany
[3]Faculty of Industrial Engineering and Management, Technion, IIT
[1]{ep490,iv250,alk23}@cam.ac.uk
[2]goran@informatik.uni-mannheim.de
[3]roiri@ie.technion.ac.il

## Abstract

*Semantic specialization* integrates structured linguistic knowledge from external resources (such as lexical relations in WordNet) into pretrained distributional vectors in the form of constraints. However, this technique cannot be leveraged in many languages, because their structured external resources are typically incomplete or non-existent. To bridge this gap, we propose a novel method that transfers specialization from a resource-rich source language (English) to virtually *any* target language. Our specialization transfer comprises two crucial steps: 1) Inducing noisy constraints in the target language through automatic word translation; and 2) Filtering the noisy constraints via a state-of-the-art *relation prediction* model trained on the source language constraints. This allows us to specialize any set of distributional vectors in the target language with the refined constraints. We prove the effectiveness of our method through intrinsic word similarity evaluation in 8 languages, and with 3 downstream tasks in 5 languages: lexical simplification, dialog state tracking, and semantic textual similarity. The gains over the previous state-of-art specialization methods are substantial and consistent across languages. Our results also suggest that the transfer method is effective even for lexically distant source-target language pairs. Finally, as a by-product, our method produces lists of WordNet-style lexical relations in resource-poor languages.

## 1 Introduction

Due to their dependence on the distributional hypothesis (Harris, 1954), that is, word co-occurrence information in large corpora, distributional word embeddings (Mikolov et al., 2013; Levy and Goldberg, 2014; Pennington et al., 2014; Melamud et al., 2016; Bojanowski et al., 2017; Peters et al., 2018, *inter alia*) conflate paradigmatic relations (e.g., synonymy, antonymy, lexical entailment, co-hyponymy, meronymy) and the broader topical (i.e., syntagmatic) relatedness (Schwartz et al., 2015; Mrkšić et al., 2017). This property can propagate undesired effects to language understanding applications such as statistical dialog modeling or text simplification (Faruqui, 2016; Chiu et al., 2016; Mrkšić et al., 2017): for instance, the inability to distinguish between synonymy and antonymy (e.g., between *cheap pubs* and *expensive restaurants*) can break task-oriented dialog or a recommendation system (Mrkšić et al., 2016; Kim et al., 2016b).

Semantic *specialization* techniques are therefore leveraged to stress a relation of interest such as semantic similarity (Wieting et al., 2015; Mrkšić et al., 2017; Ponti et al., 2018) or lexical entailment (Nguyen et al., 2017; Vulić and Mrkšić, 2018) over other types of semantic association in the word vector space. The best-performing specialization models (cf. Mrkšić et al. 2017; Ponti et al. 2018) are executed as vector space post-processors. In short, these techniques force the distributional vectors to conform to external linguistic constraints (e.g., synonymy, meronymy, lexical entailment) extracted from *structured external resources* (e.g., WordNet, BabelNet) to emphasize the particular relation. As post-processors they are applicable to any input distributional space.

A critical requirement for all specialization techniques is the set of linguistic constraints drawn from the curated external semantic resource. Such resources contain incomplete information even in resource-rich languages (e.g., English WordNet), while the resources are scarcer or even non-existent for many other languages. A solution was proposed recently to deal with incomplete information in a resource-rich language: the specialization function learned on the subset of words observed in the external resource gets propagated to the entire vocabulary in a step called *post-specialization* (Vulić



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2206–2217, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

et al., 2018). Yet, another fundamental question concerning specialization techniques is still unresolved: *how to enable specialization in virtually any language, even when the language completely lacks external lexical resources?*

In this work, we therefore propose a novel approach for *cross-lingual specialization transfer* based on Lexical Relation Induction (CLSRI). CLSRI leverages lexical information from a resource-rich language to enable specialization in any target language, without observing a single lexical constraint in the target language. The transfer method consists of two main steps: **1)** We induce a noisy set of constraints in the target language through *automatic word translation* via a shared cross-lingual word vector space (Ruder et al., 2019; Joulin et al., 2018). **2)** To mitigate the noise from the translation process, the initial set of noisy constraints is then refined in a *relation prediction* phase: we adjust a state-of-the-art neural method for lexical relation classification (Glavaš and Vulić, 2018a) and use it to predict the validity of each noisy constraint obtained in the first step. Finally, a standard specialization technique (including the post-specialization step) can then be used *monolingually* in the target language, starting from the set of refined target language constraints.

We verify the usefulness of our specialization transfer method in the intrinsic word similarity task for 8 target languages, followed by 3 downstream tasks in 5 languages: lexical simplification, dialog state tracking, and semantic textual similarity. We observe large improvements over purely distributional word vectors for all target languages and in all tasks. Moreover, we show that the proposed specialization transfer method consistently outperforms the direct specialization transfer based on the composition of the cross-lingual projection and the post-specialization function (Ponti et al., 2018), with substantial gains across all experimental setups. In order to boost the integration of external lexical knowledge into distributional models beyond English, we will release our code and lists of WordNet-style lexical relations generated by our transfer method for all target languages at: `https://github.com/cambridgeltl/xling-postspec`.

## 2 Related Work

Conflating distinct (both paradigmatic and syntagmatic) lexico-semantic relations is a well-known property of distributional word vectors; semantic specialization of such spaces for a particular lexico-semantic relation (e.g., semantic similarity or lexical entailment) benefits a number of tasks, e.g., dialog state tracking (Mrkšić et al., 2017; Ponti et al., 2018), spoken language understanding (Kim et al., 2016b,a), text simplification (Glavaš and Vulić, 2018b; Ponti et al., 2018), and cross-lingual transfer of resources (Vulić et al., 2017a).

Specialization methods inject external lexical knowledge into a distributional space, tailoring vectors for a particular relation of interest. *Joint specialization* models (Yu and Dredze, 2014; Xu et al., 2014; Bian et al., 2014; Kiela et al., 2015; Liu et al., 2015; Ono et al., 2015; Osborne et al., 2016; Nguyen et al., 2017, *inter alia*) use external constraints to modify the training objective of word embedding models (Mikolov et al., 2013; Dhillon et al., 2015; Liu et al., 2018b,a) and train specialized vectors from scratch.

In contrast, *retrofitting* (also known as *post-processing*) methods tune the pre-trained distributional vectors *post-hoc* based on the provided external constraints. Despite the fact that joint models specialize the entire space, whereas the first generation of retrofitting models specializes only the vectors of words seen in lexical constraints, the latter yield better downstream performance (Mrkšić et al., 2016). Moreover, while the joint models are tightly coupled to a concrete word embedding objective, retrofitting models can be applied on top of any distributional vector space.

*Post-specialization* (Vulić et al., 2018; Ponti et al., 2018; Kamath et al., 2019) is a generalization of retrofitting that specializes the entire distributional space: 1) it learns a global specialization function using before- and after-retrofitting vectors of words from lexical constraints as training examples and 2) it applies the global specialization functions to vectors of words unseen in lexical constraints. Similar to retrofitting, post-specialization can be applied to any vector space, but also (like *joint* specialization models) specializes the full distributional space.

Since it learns a global and explicit specialization function, post-specialization can be used for cross-lingual specialization transfer. Assuming a shared cross-lingual embedding space (Ruder et al., 2019; Glavaš et al., 2019), a post-specialization function induced on the source language subspace can be directly applied to the target language sub-

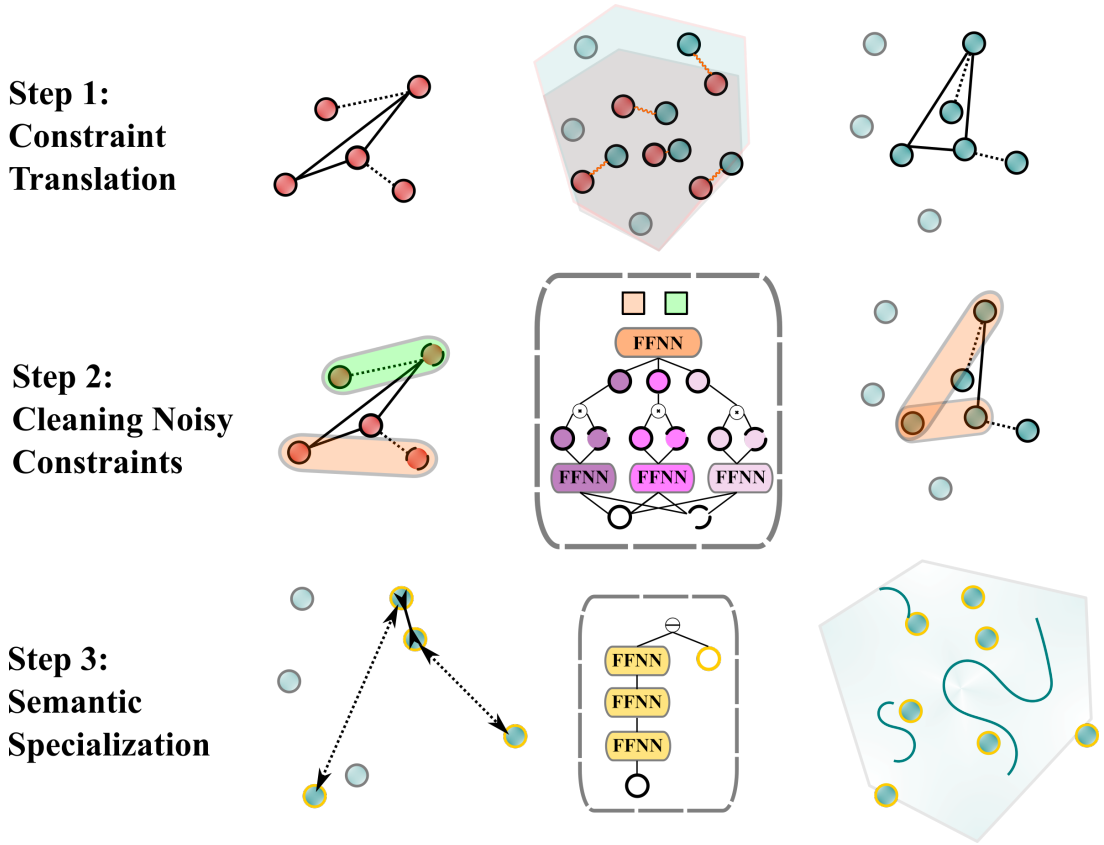


Figure 1: High-level illustration of our CLSRI framework for semantic specialization. Step 1: a network of lexical relations in a source language (red dots, left) is translated into a target language (blue dots, right) through a shared vector space (center). Step 2: a lexical relation classifier (center) trained on vector pairs sampled from the source language (left) prunes the constraints in the noisy target network (right). Step 3: the refined constraints are used to attract or repel the corresponding vectors (golden edges, left); this transformation is learned by a deep feed-forward network (center) and applied to the full target vector space.

space (Glavaš and Vulić, 2018b; Ponti et al., 2018). In this work, we propose a different approach: we use a shared cross-lingual space to (noisily) translate lexical constraints from source to target language, and then use a relation-prediction model (trained on the source language constraints) to filter out the invalid target language constraints. This allows for monolingual application of retrofitting or post-specialization in the target language. Our experiments show that the proposed specialization transfer via lexical relation induction (CLSRI) outperforms the previous state-of-the-art specialization transfer method of Ponti et al. (2018).

## 3 Methodology

**CLSRI in a Nutshell.** In cross-lingual semantic specialization our goal is to fine-tune the distributional vectors of a target language $L_t$ leveraging structured knowledge in the form of lexical constraints, available only for a resource-rich source language $L_s$. To this end, we propose a two-step *translate-and-refine* procedure for the induction of target language constraints, described in § 3.1. We first translate words in each $L_s$ constraint by retrieving their nearest neighbour in $L_t$ from a shared cross-lingual $L_s$–$L_t$ embedding space (Ruder et al., 2019). Such a translation procedure will generate noisy constraints in the target language due to (1) imperfect word translation via the cross-lingual embedding space and (2) polysemy in $L_s$ and translation of incorrect senses of $L_s$ words. We thus subsequently refine the noisy set of target constraints by having a state-of-the-art neural model for lexico-semantic relation prediction (Glavaš and Vulić, 2018a), trained on the $L_s$ constraints, discern valid from invalid $L_t$ constraints.

Following that, we perform *monolingual retrofitting and post-specialization* in the target language $L_t$, as outlined in § 3.2. The $L_t$ distributional vectors can be specialized with the cleaned $L_t$ constraints using any off-the-shelf retrofitting model (Faruqui et al., 2015; Mrkšić et al., 2016;

Lengerich et al., 2018, *inter alia*). In this work we opt for the best-performing retrofitting model ATTRACT-REPEL (AR) (Mrkšić et al., 2017; Vulić et al., 2017b). AR specializes only the words seen in the cleaned $L_t$ constraints. As the final step, we generalize AR's specialization to the entire target vocabulary with a *post-specialization* model (Ponti et al., 2018) that learns the global specialization function from pairs of distributional and AR-specialized vectors of words from $L_t$ constraints. A visual summary of our transfer model is presented in Figure 1.

Our proposed CLSRI specialization conceptually differs from an existing cross-lingual specialization transfer methodology (Ponti et al., 2018; Glavaš and Vulić, 2018b), in which the global specialization function is learned in the source language $L_s$ and then transferred directly to the target language $L_s$ via a shared cross-lingual embedding space.

### 3.1 Induction and Refinement of Constraints

**Step 1: Constraint Translation.** Following the established methodology of Mrkšić et al. (2017), constraints drawn from external resources are usually split into two broad sets: **1)** ATTRACT constraints couple words that should have similar representations (e.g., synonyms like *complicated* and *complex* or direct hyponym-hypernym pairs like *parrot* and *bird*); and **2)** REPEL constraints indicate which word pairs should appear far-flung in the space (e.g., antonyms like *ancient* and *recent*).

Given a set $A_s$ of ATTRACT word pairs and a set $R_s$ of REPEL word pairs, each word pair $(w_s^l, w_s^r)$ from the vocabulary of the source language $\mathcal{V}_s$ is automatically translated into the target language with vocabulary $\mathcal{V}_t$ using a shared cross-lingual $L_s$-$L_t$ word embedding space. We create the cross-lingual space $\mathbf{X}_{CL}$ by learning a linear map $\mathbf{W}_{CL}$ that projects the distributional space of the target language $\mathbf{X}_t$ to the distributional space $\mathbf{X}_s$ of the source language, i.e., $\mathbf{X}_{CL} = \mathbf{X}_s \cup \mathbf{X}_t\mathbf{W}_{CL}$. We translate each word $w_s$ from each linguistic constraint in $L_s$ by looking for the nearest neighbour of its vector $\mathbf{x}_s$ in the projected target space $\mathbf{X}_t\mathbf{W}_{CL}$. We employ recently proposed Relaxed Cross-domain Similarity Local Scaling (RCSLS) model of Joulin et al. (2018) to learn the projection matrix $\mathbf{W}_{CL}$ and induce the bilingual space $\mathbf{X}_{CL}$.[1]

**Step 2: Cleaning Noisy Constraints.** The $L_t$ constraints we obtain by translating $L_s$ constraints via a cross-lingual $L_1$-$L_2$ embedding space are expected to be noisy (as validated later in § 5), i.e., a shared cross-lingual space obtained via a linear projection matrix is far from ideal. The translations are going to be particularly noisy for pairs of distant languages for which the projection-based methods for inducing cross-lingual embedding spaces (including RCSLS) generally yield lower bilingual lexicon induction (BLI) performance (Søgaard et al., 2018; Joulin et al., 2018; Glavaš et al., 2019).

In the next step, we therefore clean the noisy $L_t$ constraints obtained via this imperfect translation procedure. To this end, we leverage the state-of-the-art model for lexical relation prediction: the Specialization Tensor Model (STM) (Glavaš and Vulić, 2018a). STM is a neural model that predicts lexical relations for pairs of input distributional vectors based on multi-view projections of those vectors. Each slice of the STM's central specialization tensor specifies a different projection. We modify the original $N$-ary STM classifier to now model binary classification, and train two instances of the model: one that predicts whether a pair of words represents a valid ATTRACT constraint (A-STM), and another that predicts valid REPEL constraints (R-STM). We train both models with the training instances created from the *clean* $L_s$ constraints.

Given a pair of vectors $(\mathbf{x}_l, \mathbf{x}_r)$ that corresponds to a clean linguistic constraint $(w_s^l, w_s^r)$ from $A_s$ (or $R_s$), each vector is transformed with $k$ feed-forward networks (FFNs) of the STM model. The paired projections of the two vectors resulting from each FFN are scored with a parameterized biaffine product, producing $k$ latent scores describing the nature of the relation between the input vectors. The $k$-dimensional latent feature vector is finally passed to a FFN, which performs binary classification.[2] The complete objective is summarized in Equation (1):

$$\text{FFN}^{\sigma} \bigoplus_{i=1}^{k} \left\{ \text{FFN}_i^{\tau}(\mathbf{x}_l)^{\top} \, W_i \, \text{FFN}_i^{\tau}(\mathbf{x}_r) + \mathbf{b}_i \right\} \quad (1)$$

where $\oplus$ stands for concatenation, and the output layer activations are denoted as $\sigma$ for *sigmoid* and $\tau$ for *tanh*.

[1] RCSLS substantially outperforms competing models on the task of bilingual lexicon induction as shown in a recent comparative study (Glavaš et al., 2019), and has been designed to optimize performance exactly on the word translation task.

[2] For further technical details regarding the STM, we refer the reader to the original paper (Glavaš and Vulić, 2018a).

The pairs $(\mathbf{x}_l, \mathbf{x}_r)$ created from $A_s$ and $R_s$ constitute *positive* training instances for A-STM and R-STM, respectively. For each classifier we couple each positive training instance with two types of *negative* training instances: (1) we create a negative instance by substituting a member of the pair ($\mathbf{x}_l$ or $\mathbf{x}_r$) with a randomly sampled vector from one of the other pairs in the same training batch; (2) we create a negative instance by randomly sampling a constraint from the opposing set of constraints, that is, we turn a constraint from $A_s$ into a negative example for R-STM, and, conversely, a constraint from $R_s$ into a negative training instance for A-STM. We train the A-STM and R-STM models with training instances created from $L_s$ constraints and then use the trained model to predict the validity of the translated $L_t$ constraints. We retain only the subsets of $L_t$ constraints $A_t$ and $R_t$ deemed valid by A-STM and R-STM, respectively. Vectors of $L_s$ words (during training) and vectors of $L_t$ words (at inference) are taken from the induced bilingual $L_s$-$L_t$ space $\mathbf{X}_{CL} = \mathbf{X}_s \cup \mathbf{X}_t\mathbf{W}_{CL}$.

### 3.2 Semantic Specialization

We can now directly feed $A_t$ and $R_t$ to any retrofitting model and (monolingually) specialize any distributional space in the target language. We first run the state-of-the-art retrofitting model ATTRACT-REPEL (AR) (Mrkšić et al., 2017) with $A_t$ and $R_t$ constraints. AR however, specializes only the words present in $A_t$ and $R_t$. In the next step, we generalize AR's specialization to the full vocabulary $\mathcal{V}_t$ with the state-of-the-art post-specialization model (Ponti et al., 2019). For completeness, we briefly summarize AR and the post-specialization model of Ponti et al. (2019).

**Retrofitting with ATTRACT-REPEL.** Each constraint from $A_t$ and $R_t$ is used to fine-tune the distance between their corresponding vectors $(\mathbf{x}_l, \mathbf{x}_r)$ in the target $L_t$ distributional space. Let $\mathcal{B}_A$ be a batch of vector pairs created from ATTRACT constraints $A_t$ and $\mathcal{B}_R$ the batch of vector pairs created from REPEL constraints $R_t$. For each batch $\mathcal{B}_A$ and each batch $\mathcal{B}_R$, we construct batches of corresponding negative pairs $\mathcal{T}_A(\mathcal{B}_A)$ and $\mathcal{T}_R(\mathcal{B}_R)$, containing new pairs of words sampled among those present in the batch of positive pairs. In particular, half of the negative examples $\mathbf{t}_l$ and $\mathbf{t}_r$ for ATTRACT (or REPEL) pairs are chosen by retrieving the nearest (or farthest) neighbours to $\mathbf{x}_l$ and $\mathbf{x}_r$, respectively, in terms of cosine similarity. Another half are random negative examples.

AR minimizes an objective based on max-margin loss between positive pairs and their corresponding negative pairs. More precisely, its objective has three loss components: $\mathcal{L}_{AR} = Att(\mathcal{B}_A, \mathcal{T}_A) + Rep(\mathcal{B}_R, \mathcal{T}_R) + Pre(\mathcal{B}_A, \mathcal{B}_R)$. The first component ensures that word pairs from each $\mathcal{B}_A$ are drawn closer together than those in the corresponding $T_A$ up to a certain "attract" margin $\delta_A$:

$$Att(\mathcal{B}_A, \mathcal{T}_A) = \sum_{i=1}^{|\mathcal{B}_A|} \Big[\tau\big(\delta_A \mathbf{x}_l^i \mathbf{t}_l^i - \mathbf{x}_l^i \mathbf{x}_r^i\big) + \tau\big(\delta_A + \mathbf{x}_r^i \mathbf{t}_r^i - \mathbf{x}_l^i \mathbf{x}_r^i\big)\Big] \quad (2)$$

where $\tau(z) = \max(0, z)$ is ramp function. Analogously, $Rep(\mathcal{B}_R, \mathcal{T}_R)$ forces the vectors of words in $\mathcal{B}_R$ pairs to be further away than the vectors of their corresponding $\mathcal{T}_R$ pairs by a margin $\delta_R$. Finally, $Pre(\mathcal{B}_A, \mathcal{B}_R)$ is the regularization objective that preserves the useful semantic information from the distributional space by minimizing the Euclidean distance between original and changed vectors.[3]

**Post-Specialization.** By virtue of AR retrofitting, only the subset of vectors of $L_t$ words *observed* in the refined $L_t$ constraints are specialized. The specialized subspace, however, contains useful information for propagating the specialization to the rest of the vocabulary $\mathcal{V}_t$ (i.e., to the vectors of $L_t$ words *unseen* in $A_t$ and $R_t$). Post-specialization aims to learn a global specialization function $G : \mathbf{X}_t \in \mathbb{R}^d \rightarrow \mathbf{X}'_t \in \mathbb{R}^d$ that approximates the perturbation patterns of AR as captured by changes in vectors of *seen* words from $A_t$ and $R_t$. $G$ is learned as a non-linear mapping between pairs ($\mathbf{x}_i$, $\mathbf{y}_i$), where $\mathbf{x}_i \in \mathbf{X}_t$ is a distributional vectors of some constraint word (from $A_t$ or $R_t$) and $\mathbf{y}_i$ is its corresponding AR-specialized vector. In line with Vulić et al. (2018) and Ponti et al. (2018), we implement this function as a deep feed-forward neural network with *l* hidden layers of size *h* and a final linear layer with weight $W \in \mathcal{R}^{h \times d}$. We optimize model parameters $\theta_G$ by minimizing a contrastive margin ranking loss with random confounders (Weston et al., 2011, *inter alia*). The cosine similarity between a distributional vector transformed with $G$ and the corresponding "gold" vector (i.e., AR-specialized vector) is forced to be larger than that

[3] For more details about the AR objectives, we refer the interested reader to the original work (Mrkšić et al., 2017).

between the former and randomly sampled confounders ($k$ of them) by a margin $\delta_{MM}$:

$$\sum_{i=1}^{||\mathcal{V}_s||} \sum_{j=1|j\neq i}^{k} \tau[\delta_{MM} - \cos(G(\mathbf{x}_i;\theta_G),\mathbf{y}_i) + \\ + \cos(G(\mathbf{x}_i;\theta_G),\mathbf{y}_j)] = \mathcal{L}_{MM} \quad (3)$$

Once the global specialization transformation $G$ is learned, it is applied to the whole distributional space of our target language: $\mathbf{Y}_t = G_{\theta_G}(\mathbf{X}_t)$.

Note that with our proposed specialization approach CLSRI, we execute the retrofitting and post-specialization completely monolingually in the target language $L_t$ on the automatically induced constraints in the target language. In contrast, existing work (Vulić et al., 2018; Glavaš and Vulić, 2018b; Ponti et al., 2018) transfers the post-specialization function learned for the source language $L_s$ to the target language $L_t$ via a cross-lingual vector space. This fundamental design difference is illustrated in Figure 1 and empirically validated in §5.

## 4 Experimental Setup

**Lexical Constraints.** The assortment of English constraints for specialization is the same as in prior work (Zhang et al., 2014; Ono et al., 2015; Vulić et al., 2018; Ponti et al., 2018). These constraints concern the lexical relations documented in WordNet (Fellbaum, 1998) and Roget's Thesaurus (Kipfer, 2009). Initially, they amount to 1,023,082 synonymy/ATTRACT word pairs and 380,873 antonymy/REPEL pairs, which cover 14.6% of the 200K most frequent English words, as found in the vocabulary of FASTTEXT vectors (Bojanowski et al., 2017). The number of constraints is substantially reduced in the target languages[4] after the induction process from § 3.1, both after the rough translation and after the refinement via relation prediction. The actual numbers are reported in Figure 2.

**Distributional Word Vectors.** In order to scale up our evaluation to a representative sample of languages and language types (O'Horan et al., 2016; Ponti et al., 2019), we use 300-dim distributional vectors from the FASTTEXT[5] collection

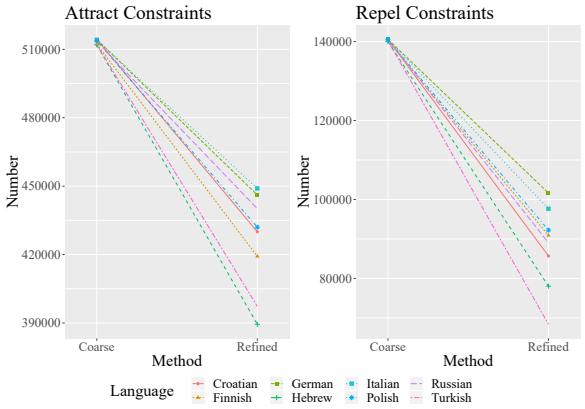


Figure 2: Number of constraints in the target languages after the initial coarse translation (*Coarse*) and after relation prediction (*Refined*).

(Bojanowski et al., 2017) trained on Wikipedia, which offers off-the-shelf comparable models for hundreds of languages. The vectors of each language are aligned to English using RCSLS (Joulin et al., 2018) (see § 3.1),[6] with the suggested hyper-parameters: 10 nearest neighbours in the RCSLS loss function, 10 iterations, and a language pair-dependent learning rate tuned from the set $\{1, 10, 25, 50\}$.

**Relation Prediction.** The STM model is trained with the Adam optimizer (Kingma and Ba, 2015), a learning rate of 0.0001, and a batch size of 48 (including negative examples) for a maximum of 10 iterations. Early stopping was implemented based on the $F_1$ score on a development set comprising 5% of the source language constraints. The hidden layer dimensionality is 300, and we use $k = 5$ specialization sub-tensors. Regarding the quality of the STM predictions, the best models achieve an $F_1$ score of 81.4 on ATTRACT constraints, and an $F_1$ score of 66.9 on REPEL constraints.[7]

**AR and Post-Specialization.** We retain the exact hyper-parameter configuration for ATTRACT-REPEL from the original work (Mrkšić et al., 2017): $\delta_A = 0.6$, $\delta_R = 0.0$, $\lambda_P = 10^{-9}$. Adagrad (Duchi et al., 2011) is employed to optimize the model parameters for 5 epochs, feeding batches of size $|\mathcal{B}_A| = |\mathcal{B}_R| = 50$, again as in prior work.

[4] These include Croatian (HR), Finnish (FI), German (DE), Hebrew (HE), Italian (IT), Polish (PL), Russian (RU), and Turkish (TR). We also use Portuguese (PT), Spanish (ES), and Arabic (AR) in some evaluations. The languages were selected according to the data availability in our evaluation tasks.

[5] FASTTEXT is trained on Wikipedia through an adaptation of Skip-Gram with Negative Sampling (SGNS) that builds representations for each word's constituent character n-grams and sums them up to obtain the entire word's representation.

[6] https://github.com/facebookresearch/fastText/tree/master/alignment

[7] Although the predicted REPEL constraints are quite inaccurate, we verified empirically that excluding them would harm the overall performance.

Owing to the difference in the amount of supervision, the post-specialization model has partially non-overlapping configurations for the baseline model of Ponti et al. (2018) and our CLSRI model. For both models each of the $l = 3$ hidden layers of the feed-forward network is composed of $h = 2,048$ hidden units, and is non-linearly activated by LeakyReLU (Maas et al., 2013). We apply a dropout of 0.2 both in input and between hidden layers. In Eq. (3), the margin $\delta_{MM} = 1$, and the negative examples amount to $k = 25$. We use SGD with the learning rate $lr = 0.1$. For the baseline, post-specialization is trained for 10 epochs, each consisting of 1 million mini-batches of 32 randomly sampled pairs. For our CLSRI model, it is limited to 2 epochs of 200K iterations each.[8]

**Models in Comparison.** Finally, we summarize the main models benchmarked in §5. First, we evaluate the original **Distributional** vectors. **X-PS** refers to the baseline model of Ponti et al. (2018) based on direct cross-lingual post-specialization. **CLSRI-AR** denotes the variant of our model based on constraint induction in $L_t$ after running the initial AR retrofitting without post-specialization; **CLSRI-PS** refers to our full model with the post-specialization step.

## 5 Results and Discussion

We evaluate different specialization models across several target languages on the intrinsic word similarity task and three downstream language understanding tasks where distinguishing between true semantic similarity and conceptual relatedness is crucial: dialog state tracking, lexical simplification, and semantic textual similarity. The choice of tasks has also been driven by the availability of standardized evaluation data in different languages.

### 5.1 Word Similarity

**Evaluation Setup.** The intrinsic evaluation is based on a set of (true) word similarity benchmarks manually translated from (subsets of) the English SimLex (Hill et al., 2015) and re-scored in the target languages.[9] In particular, the benchmarks are collected from the work of Leviant and Reichart (2015) for German, Italian, and Russian (999 pairs),[10] from Mrkšić et al. (2017) for Hebrew and Croatian (999 pairs),[11] from Venekoski and Vankka (2017) for Finnish (300),[12] from Mykowiecka et al. (2018) for Polish (999),[13] and from Ercan and Yıldız (2018) for Turkish (500).[14] We measure the Spearman's $\rho$ rank correlation between the gold human-elicited word pair similarity scores and the cosine similarity of the corresponding word vectors retrieved from each vector space.

**Results and Analysis.** We summarize the results for word similarity in Table 1. The full CLSRI-PS model outperforms both the distributional vectors and the baseline method for cross-lingual specialization (Ponti et al., 2018). In all languages but two (DE and RU) even the CLSRI-AR model without post-specialization is superior to both baselines, and the post-specialization step additionally improves the results, supporting the findings from prior work (Vulić et al., 2018). Crucially, the performance of CLSRI-PS remains strong even for distant language pairs (e.g., for EN–HE, EN–TR or EN–FI), whereas the X-PS baseline shows a drop in performance for such cases. We suspect that it is because the success of our CLSRI-PS method depends less on the quality of the underlying shared cross-lingual vector space, which is known to deteriorate for more distant language pairs (Søgaard et al., 2018; Glavaš et al., 2019).

### 5.2 Dialog State Tracking

A standard language understanding evaluation task used in prior work on semantic specialization (Mrkšić et al., 2017; Ponti et al., 2018, *inter alia*) is dialog state tracking (DST) (Henderson et al., 2014; Mrkšić et al., 2017). A DST model is a fundamental building block of statistical modular dialogue systems (Young, 2010). Its task is to maintain the information of the user's goals during a multi-turn conversation by updating the dialog belief state at each turn. Distinguishing true similarity as captured in specialized word vectors from broader relatedness is crucial for DST to succeed: e.g., a dialog system for restaurant bookings should not confuse the *western* and the *eastern* part of town, or *Thai* and *Japanese* cuisine.

[8] For both models, the hyper-parameters are chosen with a grid search over the intervals $h$={1024, 2048, 4096}, $l$={2, 3}, $lr$={0.1, 0.01, 0.001}, and optimizers in {*Adam, SGD*}, using a held-out dev set (10% of the constraints).

[9] In contrast to other datasets like WordSim-353 (Finkelstein et al., 2002) or MEN (Bruni et al., 2014), SimLex encourages scores to distinguish between pure semantic similarity (actual synonyms) and broad topical relatedness.

[10] http://leviants.com/ira.leviant/MultilingualVSMdata.html

[11] https://github.com/nmrksic/attract-repel

[12] https://github.com/venekoski/FinSemEvl

[13] http://zil.ipipan.waw.pl/CoDeS

[14] http://www.gokhanercan.com/resources/anlamver.aspx

| Model | DE | IT | HE | FI | HR | TR | PL | RU |
|---|---|---|---|---|---|---|---|---|
| Distributional | .426 | .304 | .368 | .240 | .344 | .535 | .395 | .270 |
| X-PS | .503 | .392 | .380 | .314 | .376 | .464 | .344 | .402 |
| CLSRI-AR | .500 | **.525** | .454 | .394 | .425 | .554 | .433 | .331 |
| CLSRI-PS | **.565** | .512 | **.522** | **.490** | **.505** | **.613** | **.534** | **.507** |

Table 1: Spearman's $\rho$ correlation scores for 8 languages on datasets for intrinsic evaluation of true semantic similarity. The models in comparison are briefly summarized in § 4 and in Figure 1.

| Model | DE | IT |
|---|---|---|
| Distributional | 0.640 | 0.729 |
| X-Postspec | 0.647 | 0.737 |
| CI-AR | 0.652 | 0.745 |
| CI-Postspec | **0.687** | **0.782** |

Table 2: Joint goal accuracy scores in the DST task.

**Evaluation Setup.** To be directly comparable to prior work when evaluating the effects of specialized word embeddings on DST, we rely on the Neural Belief Tracker (NBT) v2 (Mrkšić and Vulić, 2018): it is a fully statistical DST model that operates solely on the basis of pretrained word vectors (Mrkšić et al., 2017), and they are pivotal to its performance.[15] Again following prior work, our evaluation data come from the multilingual Wizard-of-Oz (WOZ) dataset (Wen et al., 2017), which is available in two target languages: German and Italian (Mrkšić et al., 2017). It contains 1,200 dialogues split into training (600 dialogues), development (200), and test data (400). We report the standard DST metric of *joint goal accuracy*: it refers to the proportion of dialog turns where all the users goals were correctly identified.

**Results and Analysis.** The results on the German and Italian DST task are summarized in Table 2. Several findings emerge from the results. First, as already confirmed in prior work (Vulić et al., 2018; Ponti et al., 2018), vectors specialized for semantic similarity are indeed important for DST: we observe improvements with all specialized vectors. The highest gains are observed with the full CSLRI-PS model. This confirms two main intuitions: 1) our proposed specialization transfer via lexical induction in the target language is more robust than the previous X-PS method of Ponti et al. (2018), and 2) the full-vocabulary post-specialization step is again useful as the initial CSLRI-AR model cannot match the performance of CSLRI-PS.

[15]Note that the original NBT framework in the English DST task has been recently surpassed by more intricate task-specific architectures (Zhong et al., 2018; Ren et al., 2018), but its lightweight design coupled with its strong dependence on input word vectors still makes it a convenient means to evaluate the effects of different specialization methods.

### 5.3 Lexical Simplification

Lexical simplification (LS) aims to automatically replace complex words (i.e., specialized terms, words used less frequently and known to fewer speakers) with their simpler in-context synonyms: the simplified text must be grammatical and retain the meaning of the original text. Lexical simplification critically depends on discerning semantic similarity from other types of semantic relatedness, as the meaning of the original text might not be preserved otherwise (e.g., *"The orange automobile crashed."* vs. *"The orange wheel crashed."*).

**Evaluation Setup.** To evaluate the effects of similarity-based specialization on LS, we employ Light-LS (Glavaš and Štajner, 2015), a language-agnostic LS tool that makes simplifications based on word similarities in a given vector space. The quality of similarity-based information encoded in the vector space encode is thus expected to directly correlate with the performance of Light-LS. We use LS datasets for Italian (IT) (Tonelli et al., 2016), Spanish (ES) (Saggion et al., 2015; Saggion, 2017), and Portuguese (PT) (Hartmann et al., 2018) to evaluate the specialized spaces in those languages. We rely on the standard LS evaluation metric of *Accuracy* (Horn et al., 2014; Glavaš and Štajner, 2015): it quantifies both the quality and frequency of replacements as a number of correct simplifications divided by the total number of complex words.

**Results and Analysis.** The results are reported in Table 3. As shown in previous work (Vulić et al., 2018; Ponti et al., 2018), retrofitting (CLSRI-AR) and the cross-lingual post-specialization transfer (X-PS) are substantially better in the LS task than the original distributional space. However, our full CLSRI-PS model results in substantial boosts in the

| Model | LS | | | STS |
|---|---|---|---|---|
| | IT | ES | PT | AR |
| Distributional | 0.28 | 0.27 | 0.27 | 0.67 |
| X-PS | 0.38 | 0.57 | 0.55 | 0.66 |
| CLSRI-AR | 0.35 | 0.51 | 0.33 | 0.66 |
| CLSRI-PS | **0.51** | **0.74** | **0.72** | **0.70** |

Table 3: Lexical Simplification (LS) performance for Italian, Spanish, and Portuguese; Semantic Textual Similarity (STS) performance for Arabic.

LS task (13-17%) over the previous best reported scores of X-PS as well as over CLSRI-AR.

### 5.4 Semantic Text Similarity

**Evaluation Setup.** Finally, we also carry downstream evaluation in the semantic textual similarity (STS) task. The Arabic dataset constructed for SemEval 2017 track 1[16] (Cer et al., 2017) consists of sentence pairs scored from 0 (semantic independence) to 5 (semantic equivalence). We augment the training set with all the data for English (translated with Google Translate) from previous editions of the shared task. To classify sentence pairs, we employ the CNN-HTCI model (Shao, 2017). Each sentence is encoded with a convolutional network into a hidden representation. Then, the interaction between the pair of representations is evaluated as their element-wise multiplication and absolute difference. A fully connected network takes this interaction as input, and infers the similarity score.

**Results and Analysis.** We report the accuracy scores for the Arabic STS in Table 3. Interestingly, for STS both X-PS and CLSRI-AR damage the performance of the distributional baseline. However, the full CLSRI-PS model still shows a substantial improvement over all baselines. This again suggests its wide stability and effectiveness.

To empirically validate the importance of noisy constraints refinement (see § 3.1), we have also evaluated an ablated variant of CLSRI-PS without the refinement step: this model variant relies only on noisy translations of $L_s$ lexical constraints. While this variant leads to improvements over the X-PS baseline across the board, it is consistently outperformed by the full CSLRI-PS model in downstream tasks: e.g., the gains with the full model are 2-3% in the LS task, and 2% in the Arabic STS task. Since the full CSLRI-PS model does not require any additional input for the lexical prediction step (i.e., it operates with the same set of $L_s$ constraints as the translation step), these results suggest that both steps should be applied for improved specialization in the target languages.

[16] http://alt.qcri.org/semeval2017/task1/

## 6 Future Work

As a supplemental benefit of CLSRI, the constraints induced by translation and pruning hold promise to create WordNet-style resources for languages that lack structured linguistic knowledge. While the relations extracted in this proof-of-concept paper do not cover the rich and expressive set of WordNet relations in its entirety, they are nonetheless sufficient to create parts of the core WordNet structure with synsets (synonyms) and lexical relations across synsets (antonyms) from scratch.

Furthermore, our method is amenable be extended to contextualized embeddings (Lauscher et al., 2019) and/or other WordNet lexical relations such as hypernyms and hyponyms. In recent works, procedures of retrofitting (Vulić and Mrkšić, 2018) and post-specialization (Kamath et al., 2019) have been developed for lexical entailment. These procedures can be easily adapted to the semantic specialization step presented in § 3.2, whereas constraint translation and refinement (§ 3.1) are relation-agnostic. We will exploit these directions in future work.

## 7 Conclusion

We have proposed a new method for cross-lingual transfer of semantic specialization via induction of lexical constraints in a resource-poor target language. We have verified its usefulness in intrinsic and extrinsic language understanding tasks and across a spectrum of target languages. We report consistent improvements over previous state-of-the-art specialization methods. Crucially, our method is robust to target languages that are distant from source languages, as its performance is consistent across all considered language pairs.

## Acknowledgements

This work is supported by the ERC Consolidator Grant LEXICAL (no 648909). The work of Goran Glavaš is supported by the Baden-Württemberg Stiftung (AGREE grant of the Eliteprogramm). We thank the three anonymous reviewers for their helpful comments and suggestions.

## References


Jiang Bian, Bin Gao, and Tie-Yan Liu. 2014. Knowledge-powered deep learning for word embedding. In *Proceedings of ECML-PKDD*, pages 132–148.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the ACL*, 5:135–146.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *Journal of Artificial Intelligence Research*, 49:1–47.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of SEMEVAL*, pages 1–14.

Billy Chiu, Anna Korhonen, and Sampo Pyysalo. 2016. Intrinsic evaluation of word vectors fails to predict extrinsic performance. In *Proceedings of REPEVAL*, pages 1–6.

Paramveer S. Dhillon, Dean P. Foster, and Lyle H. Ungar. 2015. Eigenwords: Spectral word embeddings. *Journal of Machine Learning Research*, 16:3035–3078.

John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Gökhan Ercan and Olcay Taner Yıldız. 2018. Anlamver: Semantic model evaluation dataset for turkish-word similarity and relatedness. In *Proceedings of COLING*, pages 3819–3836.

Manaal Faruqui. 2016. *Diverse Context for Learning Word Representations*. Ph.D. thesis, Carnegie Mellon University.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of NAACL-HLT*, pages 1606–1615.

Christiane Fellbaum. 1998. *WordNet*. MIT Press.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: The concept revisited. *ACM Transactions on Information Systems*, 20(1):116–131.

Goran Glavaš and Ivan Vulić. 2018a. Discriminating between lexico-semantic relations with the specialization tensor model. In *Proceedings of NAACL-HLT*, pages 181–187.

Goran Glavaš and Ivan Vulić. 2018b. Explicit retrofitting of distributional word vectors. In *Proceedings of ACL*, pages 34–45.

Goran Glavaš, Robert Litschko, Sebastian Ruder, and Ivan Vulić. 2019. How to (properly) evaluate cross-lingual word embeddings: On strong baselines, comparative analyses, and some misconceptions. In *Proceedings of ACL*, pages 710–721.

Goran Glavaš and Sanja Štajner. 2015. Simplifying lexical simplification: Do we need simplified corpora? In *Proceedings of ACL*, pages 63–68.

Zellig S. Harris. 1954. Distributional structure. *Word*, 10(23):146–162.

Nathan S. Hartmann, Gustavo H. Paetzold, and Sandra M. Aluísio. 2018. SIMPLEX-PB: A lexical simplification database and benchmark for Portuguese. In *Proceedings of the 2018 International Conference on Computational Processing of the Portuguese Language*, pages 272–283.

Matthew Henderson, Blaise Thomson, and Steve Young. 2014. Robust dialog state tracking using delexicalised recurrent neural networks and unsupervised adaptation. In *Proceedings of IEEE SLT*, pages 360–365.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Colby Horn, Cathryn Manduca, and David Kauchak. 2014. Learning a lexical simplifier using Wikipedia. In *Proceedings of ACL*, pages 458–463.

Armand Joulin, Piotr Bojanowski, Tomas Mikolov, Hervé Jégou, and Edouard Grave. 2018. Loss in translation: Learning bilingual word mapping with a retrieval criterion. In *Proceedings of EMNLP*, pages 2979–2984.

Aishwarya Kamath, Jonas Pfeiffer, Edoardo Maria Ponti, Goran Glavaš, and Ivan Vulić. 2019. Specializing distributional vectors of all words for lexical entailment. In *Proceedings of the 4th Workshop on Representation Learning for NLP*, pages 72–83.

Douwe Kiela, Felix Hill, and Stephen Clark. 2015. Specializing word embeddings for similarity or relatedness. In *Proceedings of EMNLP*, pages 2044–2048.

Joo-Kyung Kim, Marie-Catherine de Marneffe, and Eric Fosler-Lussier. 2016a. Adjusting word embeddings with semantic intensity orders. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 62–69.

Joo-Kyung Kim, Gokhan Tur, Asli Celikyilmaz, Bin Cao, and Ye-Yi Wang. 2016b. Intent detection using semantically enriched word embeddings. In *Proceedings of SLT*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR (Conference Track)*.

Barbara Ann Kipfer. 2009. *Roget's 21st Century Thesaurus (3rd Edition)*. Philip Lief Group.

Anne Lauscher, Ivan Vulić, Edoardo Maria Ponti, Anna Korhonen, and Goran Glavaš. 2019. Informing unsupervised pretraining with external linguistic knowledge.

Benjamin J. Lengerich, Andrew L. Maas, and Christopher Potts. 2018. Retrofitting distributional embeddings to knowledge graphs with functional relations. In *Proceedings of COLING*, pages 2423–2436.

Ira Leviant and Roi Reichart. 2015. Separated by an un-common language: Towards judgment language informed vector space modeling. *arXiv preprint arXiv:1508.00106*.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of ACL*, pages 302–308.

Qian Liu, Heyan Huang, Yang Gao, Xiaochi Wei, Yuxin Tian, and Luyang Liu. 2018a. Task-oriented word embedding for text classification. In *Proceedings of COLING*, pages 2023–2032.

Qian Liu, Heyan Huang, Guangquan Zhang, Yang Gao, Junyu Xuan, and Jie Lu. 2018b. Semantic structure-based word embedding by incorporating concept convergence and word divergence. In *Proceedings of AAAI*, pages 5261–5268.

Quan Liu, Hui Jiang, Si Wei, Zhen-Hua Ling, and Yu Hu. 2015. Learning semantic word embeddings based on ordinal knowledge constraints. In *Proceedings of ACL*, pages 1501–1511.

Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of ICML*.

Oren Melamud, David McClosky, Siddharth Patwardhan, and Mohit Bansal. 2016. The role of context types and dimensionality in learning word embeddings. In *Proceedings of NAACL-HLT*, pages 1030–1040.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NeurIPS*, pages 3111–3119.

Nikola Mrkšić and Ivan Vulić. 2018. Fully statistical neural belief tracking. In *Proceedings of ACL*, pages 108–113.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2017. Neural belief tracker: Data-driven dialogue state tracking. In *Proceedings of ACL*, pages 1777–1788.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina Maria Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. In *Proceedings of NAACL-HLT*, pages 142–148.

Nikola Mrkšić, Ivan Vulić, Diarmuid Ó Séaghdha, Ira Leviant, Roi Reichart, Milica Gašić, Anna Korhonen, and Steve Young. 2017. Semantic specialisation of distributional word vector spaces using monolingual and cross-lingual constraints. *Transactions of the ACL*, 5:309–324.

Agnieszka Mykowiecka, Malgorzata Marciniak, and Piotr Rychlik. 2018. SimLex-999 for Polish. In *Proceedings of LREC*, pages 2398–2402.

Kim Anh Nguyen, Maximilian Köper, Sabine Schulte im Walde, and Ngoc Thang Vu. 2017. Hierarchical embeddings for hypernymy detection and directionality. In *Proceedings of EMNLP*, pages 233–243.

Helen O'Horan, Yevgeni Berzak, Ivan Vulić, Roi Reichart, and Anna Korhonen. 2016. Survey on the use of typological information in natural language processing. In *Proceedings of COLING*, pages 1297–1308.

Masataka Ono, Makoto Miwa, and Yutaka Sasaki. 2015. Word embedding-based antonym detection using thesauri and distributional information. In *Proceedings of NAACL-HLT*, pages 984–989.

Dominique Osborne, Shashi Narayan, and Shay Cohen. 2016. Encoding prior knowledge with eigenword embeddings. *Transactions of the ACL*, 4:417–430.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of EMNLP*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237.

Edoardo Maria Ponti, Helen O'Horan, Yevgeni Berzak, Ivan Vulić, Roi Reichart, Thierry Poibeau, Ekaterina Shutova, and Anna Korhonen. 2019. Modeling language variation and universals: A survey on typological linguistics for natural language processing. *Computational Linguistics*.

Edoardo Maria Ponti, Ivan Vulić, Goran Glavaš, Nikola Mrkšić, and Anna Korhonen. 2018. Adversarial propagation and zero-shot cross-lingual transfer of word vector specialization. In *Proceedings of EMNLP*, pages 282–293.

Liliang Ren, Kaige Xie, Lu Chen, and Kai Yu. 2018. Towards universal dialogue state tracking. In *Proceedings of EMNLP*, pages 2780–2786.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2019. A survey of cross-lingual embedding models. *Journal of Artificial Intelligence Research*, 65:569–631.

Horacio Saggion. 2017. Automatic text simplification. *Synthesis Lectures on Human Language Technologies*, 10(1):1–137.

Horacio Saggion, Sanja Štajner, Stefan Bott, Simon Mille, Luz Rello, and Biljana Drndarevic. 2015. Making it simplext: Implementation and evaluation of a text simplification system for Spanish. *ACM Transactions on Accessible Computing*, 6(4):14.

Roy Schwartz, Roi Reichart, and Ari Rappoport. 2015. Symmetric pattern based word embeddings for improved word similarity prediction. In *Proceedings of CoNLL*, pages 258–267.

Yang Shao. 2017. HCTI at SemEval-2017 Task 1: Use convolutional neural network to evaluate semantic textual similarity. In *Proceedings of SEMEVAL*, pages 130–133.

Anders Søgaard, Sebastian Ruder, and Ivan Vulić. 2018. On the limitations of unsupervised bilingual dictionary induction. In *Proceedings of ACL*, pages 778–788.

Sara Tonelli, Alessio Palmero Aprosio, and Francesca Saltori. 2016. SIMPITIKI: A simplification corpus for Italian. In *Proceedings of CLiC-it*.

Viljami Venekoski and Jouko Vankka. 2017. Finnish resources for evaluating language model semantics. In *Proceedings of the 21st Nordic Conference on Computational Linguistics*, pages 231–236.

Ivan Vulić, Goran Glavaš, Nikola Mrkšić, and Anna Korhonen. 2018. Post-specialisation: Retrofitting vectors of words unseen in lexical resources. In *Proceedings of NAACL-HLT*, pages 516–527.

Ivan Vulić and Nikola Mrkšić. 2018. Specialising word vectors for lexical entailment. In *Proceedings of NAACL-HLT*, pages 1134–1145.

Ivan Vulić, Nikola Mrkšić, and Anna Korhonen. 2017a. Cross-lingual induction and transfer of verb classes based on word vector space specialisation. In *Proceedings of EMNLP*, pages 2536–2548.

Ivan Vulić, Nikola Mrkšić, Roi Reichart, Diarmuid Ó Séaghdha, Steve Young, and Anna Korhonen. 2017b. Morph-fitting: Fine-tuning word vector spaces with simple language-specific rules. In *Proceedings of ACL*, pages 56–68.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of EACL*.

Jason Weston, Samy Bengio, and Nicolas Usunier. 2011. WSABIE: Scaling up to large vocabulary image annotation. In *Proceedings of IJCAI*, pages 2764–2770.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. From paraphrase database to compositional paraphrase model and back. *Transactions of the ACL*, 3:345–358.

Chang Xu, Yalong Bai, Jiang Bian, Bin Gao, Gang Wang, Xiaoguang Liu, and Tie-Yan Liu. 2014. RC-NET: A general framework for incorporating knowledge into word representations. In *Proceedings of CIKM*, pages 1219–1228.

Steve Young. 2010. Still talking to machines (cognitively speaking). In *Proceedings of INTERSPEECH*, pages 1–10.

Mo Yu and Mark Dredze. 2014. Improving lexical embeddings with semantic knowledge. In *Proceedings of ACL*, pages 545–550.

Jingwei Zhang, Jeremy Salwen, Michael Glass, and Alfio Gliozzo. 2014. Word semantic representations using Bayesian probabilistic tensor factorization. In *Proceedings of EMNLP*, pages 1522–1531.

Victor Zhong, Caiming Xiong, and Richard Socher. 2018. Global-locally self-attentive encoder for dialogue state tracking. In *Proceedings of ACL*, pages 1458–1467.

# Modelling the interplay of metaphor and emotion through multitask learning

**Verna Dankers**[1] **Marek Rei**[2,3] **Martha Lewis**[1] **Ekaterina Shutova**[1]
[1]Institute for Logic, Language and Computation, University of Amsterdam
[2]The ALTA Institute, Computer Laboratory, University of Cambridge
[3]Department of Computing, Imperial College London
vernadankers@gmail.com, marek.rei@cl.cam.ac.uk, {m.a.f.lewis,e.shutova}@uva.nl

## Abstract

Metaphors allow us to convey emotion by connecting physical experiences and abstract concepts. The results of previous research in linguistics and psychology suggest that metaphorical phrases tend to be more emotionally evocative than their literal counterparts. In this paper, we investigate the relationship between metaphor and emotion within a computational framework, by proposing the first joint model of these phenomena. We experiment with several multitask learning architectures for this purpose, involving both hard and soft parameter sharing. Our results demonstrate that metaphor identification and emotion prediction mutually benefit from joint learning and our models advance the state of the art in both of these tasks.

## 1 Introduction

Metaphors allow us to reason about abstract concepts by linking them to our physical experiences (Lakoff and Johnson, 1983). Metaphorical language arises through systematic association between two distinct semantic domains — the source and the target — as illustrated by the sentence "The news *leaked out* despite the secrecy", where a term from the source domain of *liquids* is used to describe *information* (the target domain). This metaphorical association widely manifests itself in language, e.g. we can similarly talk about "being *engulfed* by a *stream* of bad news". Metaphorical associations allow us to project knowledge from the source domain to the target, inviting new reasoning frameworks and connotations to emerge.

Much previous research on metaphorical language in fields such as linguistics (Blanchette et al., 2001; Kövecses, 2003), cognitive psychology (Crawford, 2009; Thibodeau and Boroditsky, 2011) and neuroscience (Aziz-Zadeh and Damasio, 2008; Jabbi et al., 2008) points to its prevalent affective content. Linguistic expressions describing one's emotional state have a relatively high incidence of figurative language and metaphor in particular (Fainsilber and Ortony, 1987; Fussell and Moss, 1998; Gibbs Jr et al., 2002), as illustrated by the phrase "My mind was *seething* and *boiling*". On the other hand, a stronger emotion appears to be conveyed through the association of source and target domains more generally. Mohammad et al. (2016) found that metaphorical phrases are consistently perceived as carrying more emotion than their literal paraphrases and the literal uses of the same source domain words. For instance, "*leaking* information" conveys an implicit judgement, as compared to the more neutral paraphrase "*disclosing* information". Their results also suggest that the emotional content of the metaphor is not due to the properties of individual source and target domains, but rather arises compositionally through their interaction. These findings are supported through a range of psycholinguistic studies: Citron and Goldberg (2014) find taste metaphors to be more emotionally evocative than their literal counterparts. Citron et al. (2016) show that conventional metaphorical language in short stories from various domains elicits more activation in brain regions involved in emotional processing, compared to literal language.

Computational modelling of metaphor (Shutova et al., 2016; Rei et al., 2017; Gao et al., 2018) and emotion (Wang et al., 2016; Zhang et al., 2018; Wu et al., 2019) are tasks widely addressed in natural language processing (NLP), with a range of applications from machine translation (Fadaee et al., 2018) to opinion mining (Yadollahi et al., 2017). However, the two phenomena have been typically modelled independently. Exceptions include the use of hand-engineered emotion features when training a classifier for metaphor identification (Strzalkowski et al., 2014) and auto-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2218–2229, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

matic identification of affect carried by metaphors (Kozareva, 2013; Strzalkowski et al., 2014). However, none of this research has attempted to model metaphor and emotion within a unified model of semantic composition. In this paper, we present the first joint model of metaphor and emotion, trained to learn the patterns of their interaction via flexible parameter sharing techniques offered by multitask learning (MTL). Our model is compositional, building meaning representations of words and phrases in context. The intuition is that the meaning of a word is not intrinsically metaphorical or emotional, but both of these phenomena may manifest when the word is used in a particular context.

Specifically, we train deep learning architectures on metaphor identification and emotion prediction tasks jointly. Metaphor identification is performed at word level and sentence level, while emotion prediction is modelled as a regression task, predicting numerical scores for the valence, arousal and dominance dimensions of emotion. We experiment with MTL architectures employing both hard and soft parameter sharing methods. Models employing hard parameter sharing jointly encode the lower-level word representations using layers shared among the tasks. The soft parameter sharing methods have two task-specific networks connected through linear units or gates.

Our models outperform existing approaches to both metaphor identification and emotion prediction tasks, advancing the state of the art in these areas. Moreover, we show that jointly learning both tasks within one model provides stable performance improvements across architectures.

## 2 Related work

### 2.1 Computational models of metaphor

Early approaches to metaphor identification used hand-engineered features and a trained classifier, such as logistic regression (Dunn, 2013), random forests (Tsvetkov et al., 2014), decision trees (Mohler et al., 2013; Gargett and Barnden, 2015) or support vector machines (Hovy et al., 2013; Mohler et al., 2013). Examples of linguistic features used are POS tags (Hovy et al., 2013), concreteness or imageability ratings (Turney et al., 2011; Broadwell et al., 2013; Gargett and Barnden, 2015), ontological concepts (Dunn, 2013) and WordNet super-senses (Hovy et al., 2013) and synsets (Mohler et al., 2013). To become less reliant on hand-crafted features, corpus-driven approaches emerged, using sparse (Shutova et al., 2010; Gutierrez et al., 2016) or dense word embeddings (Shutova et al., 2016; Bulat et al., 2017).

Recently, the use of deep neural networks for metaphor identification has gained popularity. Rei et al. (2017) presented a network designed to predict the metaphoricity of a word pair, by modelling the words' interaction using a gating function. Other approaches treated metaphor identification as a sequence labelling task. Do Dinh and Gurevych (2016) proposed a multi-layer perceptron acting on word embeddings. Do Dinh et al. (2018) present a MTL approach combining multiple metaphor identification tasks using two architectures: a hard parameter sharing recurrent network and the recurrent Sluice network of Ruder et al. (2019). During a recent shared task on metaphor identification, various deep neural architectures were presented (Leong et al., 2018), among which were several hybrid approaches that incorporated linguistic features in recurrent networks. Gao et al. (2018) presented the current best-performing model for metaphor sequence labelling. They employed GloVe (Pennington et al., 2014) and ELMo (Peters et al., 2018) embeddings as input to a bidirectional LSTM (Bi-LSTM) followed by a classification layer.

### 2.2 Computational models of emotion

The vast majority of NLP research on affective language analysis has focused on the prediction of emotion categories and sentiment analysis. Early work on emotion prediction assumed categorical models of emotion, such as Ekman's model of six emotions (Ekman, 1992) (anger, disgust, fear, joy, sadness and surprise). A variety of computational models have been proposed for emotion classification, ranging from vector space models (Danisman and Alpkocak, 2008), to machine learning classifiers (Perikos and Hatzilygeroudis, 2016) and deep learning architectures (Zhang et al., 2018).

Recently, multi-dimensional emotion analysis has gained popularity: it represents emotion through a more fine-grained and psychologically-motivated model (Buechel and Hahn, 2017). We employ the Valence-Arousal-Dominance (VAD) model (Mehrabian, 1996) that describes affective states relative to these emotional dimensions. Valence represents the polarity, arousal the degree of excitement, and dominance the perceived degree

of control over a situation.

Existing methods for dimensional emotion analysis are either lexicon-based or use supervised learning. Lexicon-based methods assume the emotional value of a sentence to be a composition of per-word values. These values are extracted from an affect lexicon (Warriner et al., 2013) and combined using their mean (Kim et al., 2010), a weighted mean, or a Gaussian Mixture Model (Paltoglou et al., 2013). Other approaches train classifiers using $n$-gram and sentiment features (Malandrakis et al., 2013; Buechel and Hahn, 2016), and deep learning models.

Wang et al. (2016) were among the first to present a deep learning architecture for dimensional emotion analysis using the VA model: they proposed a convolutional network operating on regions within the input, and a LSTM layer acting on the region encodings. Wang et al. (2018) used the VAD labelled corpus of Buechel and Hahn (2017), but considered only valence, effectively reducing the task to sentiment analysis. They present a deep network of stacked Bi-LSTM layers with residual connections. Akhtar et al. (2018) performed regression for all three dimensions using a convolutional and two recurrent networks combined in an ensemble extended with hand-crafted features. The emotion dimensions were considered separately and in a MTL setup. Most recently, Wu et al. (2019) proposed a variational auto-encoder model including a recurrent module trained to perform emotion prediction. The model was trained in a semi-supervised way, using only the labels of 40% of the training samples.

### 2.3 Metaphor and emotion

Existing work combining metaphor and emotion either focuses on the inclusion of emotion features in metaphor identification or on the automatic identification of affect carried by metaphors.

Kozareva (2013) and Strzalkowski et al. (2014) modelled the affect carried by metaphors and evaluate their approaches on a metaphor-rich corpus containing data from four languages. Kozareva (2013) performs polarity classification and valence regression using the AdaBoost classifier and support vector regression trained on information from the sentence, its context, and source and target domain annotations. Strapparava and Mihalcea (2007) proposed an affect calculus to estimate the affect expressed by a linguistic metaphor as positive, negative, or neutral. The affect calculus takes into account the metaphor target, the source relation, the relation's arguments and type, and the prior affect of the target.

Gargett and Barnden (2015) considered metaphor identification on nouns, verbs and prepositions using hand-engineered features, including lexicon-based VAD emotion features. The emotion features proved most beneficial for metaphor identification for nouns and verbs.

## 3 Tasks and datasets

**VUA metaphor corpus** The VUA metaphor corpus[1] (Steen, 2010) is a subset of the British National Corpus (Leech, 1992) in which each word is annotated as literally- or metaphorically-used. The corpus contains over ten thousand sentences, sampled from four genres: academic writing, news, conversation and fiction. The reported inter-annotator agreement is 0.84 in terms of Fleiss's $\kappa$. For comparability reasons, we use a preprocessed variant of the corpus as provided by Gao et al. (2018), who use 25% of the sentences for testing. We perform metaphor identification at word level, experimenting in a sequence labelling paradigm.

**LCC metaphor corpus** The Language Computer Corporation (LCC) metaphor dataset (Mohler et al., 2016) is a metaphor-rich corpus containing data in English, Farsi, Spanish and Russian.[2] We use the English portion of this dataset that consists of data from the ClueWeb corpus and the Debate Politics online forum. Annotators rated the metaphoricity of sentences from zero (i.e. literal) to three (i.e. clearly metaphorical). Mohler et al. (2016) considered agreement between annotators to be a difference of $\leq 1$ (on a range from 0 to 3). With this definition, the inter-annotator agreement on metaphoricity is 92.8%.

We extract nine thousand samples from the freely available portion of the dataset, average the scores assigned by individual annotators and normalise them to the scale from zero to one. We use the data to perform sentence-level regression, employing ten-fold cross-validation using 70-10-20 splits for training, validating and testing, respectively.

[1] Preprocessed variant available at: https://github.com/gao-g/metaphor-in-context.

[2] Available upon request from Mohler et al. (2016).

| Sentence | Val. | Arous. | Dom. |
|---|---|---|---|
| "Tell her I love her." | .94 | .88 | .83 |
| Tell me, or I'll kill – | .35 | .69 | .83 |
| What did you say? | .50 | .54 | .50 |
| This is torture. | .14 | .72 | .27 |

Table 1: EmoBank examples with normalised scores, illustrating the differences among the dimensions.

**EmoBank corpus** EmoBank,[3] (Buechel and Hahn, 2017) is one of the most recent corpora developed based on the VAD model. EmoBank contains ten thousand sentences from the manually annotated sub-corpus of American English (Ide et al., 2008) and the Affective Text corpus (Strapparava and Mihalcea, 2007). The corpus balances many genres: news headlines, blogs, essays, fiction, letters, newspapers and travel guides. Each sentence is rated on a scale from one to five for each dimension, from the perspective of the writer and the reader. The inter-annotator agreement rates are 0.61 and 0.63 in terms of Pearson's $r$ for the two perspectives, respectively. We combine the scores of readers and writers, normalised to the scale from zero to one. We use EmoBank to perform sentence-level regression for each of the V, A, and D dimensions separately, using ten-fold cross-validation with 70-10-20 splits for training, validating and testing, respectively. Table 1 lists examples exhibiting a range of VAD values.

Since we focus on the interaction of metaphor and emotion, we pair up the word-level and sentence-level metaphor tasks with the regression tasks for each separate dimension of V, A, or D one by one, in a MTL setup.

## 4 Methods

We construct a recurrent neural architecture operating at two different levels. Based on the VUA metaphor corpus, the model learns to detect metaphor at word level in a sequence labelling paradigm. When optimised on the LCC metaphor corpus, the architecture is adapted to predict a metaphoricity score at sentence level. For emotion prediction, the architecture is the same as for the sentence-level metaphor prediction task.

The system receives a tokenised sentence as input and maps it to word embeddings, by concatenating representations from pre-trained GloVe and ELMo models. Next, these embeddings are passed through a Bi-LSTM, building task- and context-dependent representations for each word. For token labelling, the hidden states from each direction are concatenated and passed through a feedforward layer, followed by a sigmoid activation. We model metaphor detection as a binary task for comparability to the literature. Gao et al. (2018) use a similar model for metaphor detection at word level, but employ a softmax activation.

For the sentence-level score prediction, the concatenated Bi-LSTM hidden states are passed through the attention function, which includes a linear layer and softmax normalisation, in order to construct a sentence representation. The resulting vector is passed through a feedforward layer, then used to predict a sentence-level score with sigmoid activation. Since we used a sigmoid ac-

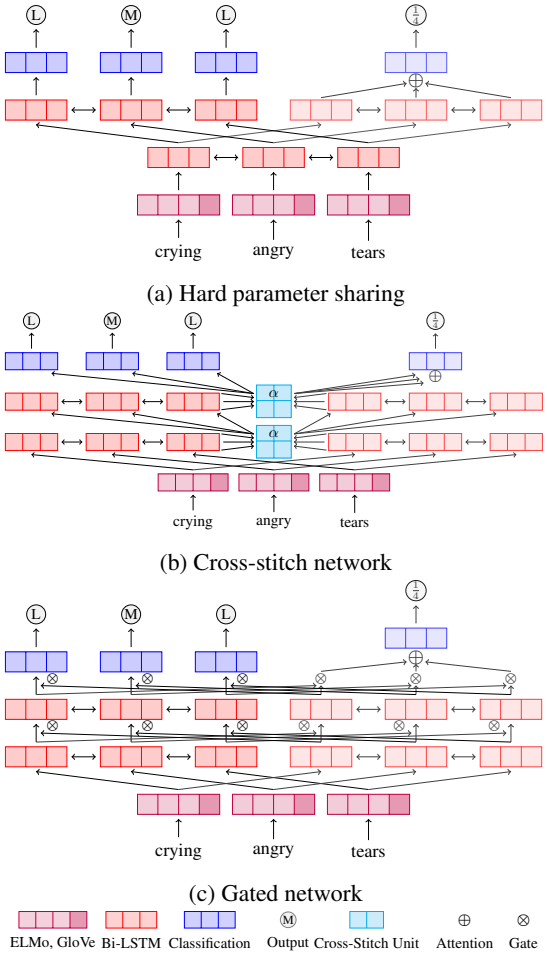


Figure 1: Overview of the MTL architectures. The parameters of the main and auxiliary task are indicated through highlighting, where the metaphor detection task is considered the main task in this setup. For compactness, two of the three Bi-LSTM layers are shown.

[3] https://github.com/JULIELab/EmoBank.

tivation function in the token labelling task, both the metaphor and emotion tasks are structurally very similar. We train for metaphor detection using the binary cross-entropy loss function and for regression using the root mean squared error loss function.

We also experiment with fine-tuning a pre-trained BERT architecture (Devlin et al., 2019) for each of the tasks. This validates that the performance differences are due to the task interactions and not specific to the recurrent architecture. The inputs to this network consist of the BERT-specific word and position embeddings. For the word-level sequence labelling task, the outputs of the last Transformer layer are fed to the classification layer. For the sentence-level tasks, an additional attention module is again used to construct sentence representations. An alternative way of using BERT would be to provide contextualised embeddings. We do not consider this in the present work but leave it as an area to be explored. BERT performs labelling on subword units called WordPieces; we consider a word metaphorical if any of these subword units is labelled metaphorical. This choice is motivated by the fact that although a metaphorical prefix or suffix could result in an incorrect metaphorical label, this is unlikely: what is much more likely is that a common prefix or suffix is not considered metaphorical while the main piece is.

In the following sections we describe three different approaches to optimising these networks using MTL. We experiment with four setups: one recurrent and one BERT hard parameter sharing setup, one recurrent cross-stitch network and one gated recurrent network. In the MTL setups, the models are trained on two tasks at once, but we distinguish between a main and auxiliary task by down-weighting the loss of the auxiliary task to allow the networks to specialise most in one direction. For example, when seeking performance improvements in the word-level metaphor identification task, metaphor identification is the main task and emotion regression is the auxiliary task.

### 4.1 Hard parameter sharing

We customise the models to jointly perform metaphor detection and emotion prediction. By training the model to identify emotional states in text, the system learns to recognise emotion-related features which can be useful for the task of metaphor detection. In addition, optimising for two different but related tasks helps prevent the model from overfitting to either of them.

Following established work on MTL (Caruana, 1993), we first experiment with hard parameter sharing. In this setting, the architecture shares the word embeddings and lower Bi-LSTM layers between the two tasks, as shown in Figure 1a. On top of these shared components, each task has one separate Bi-LSTM layer, followed by a task-specific output layer. For sentence scoring, the attention function for constructing sentence representations is also learned individually for every task. This setup allows the model to learn shared feature detectors in the lower layers, while top layers are still able to learn task-specific features.

The hard parameter sharing setup using BERT shares all of BERT's Transformer layers among the tasks, apart from the last layer to allow for specialisation. Furthermore, the output and attention layers are task-specific as well.

### 4.2 Cross-stitch network

As an alternative to hard parameter sharing, soft sharing provides parallel models with dedicated parameters for each task, while also connecting them together to allow for information transfer. In the cross-stitch network, the soft sharing mechanism is a cross-stitch unit (Misra et al., 2016). These units contain $\alpha$-parameters which regulate the information flow in each direction and are optimised during training. We apply cross-stitch sharing after each recurrent layer, computing the updated hidden states as:

$$\widetilde{\mathbf{h}}_A = \alpha_{AA}\mathbf{h}_A + \alpha_{BA}\mathbf{h}_B \quad (1)$$

$$\widetilde{\mathbf{h}}_B = \alpha_{BB}\mathbf{h}_B + \alpha_{AB}\mathbf{h}_A \quad (2)$$

where $\mathbf{h}_A$ and $\mathbf{h}_B$ are the concatenated Bi-LSTM hidden states, from parallel networks for tasks $A$ and $B$, while $\widetilde{\mathbf{h}}_A$ and $\widetilde{\mathbf{h}}_B$ are the updated hidden states. Note that the $\alpha$-parameters are specific to each layer. The $\alpha$-parameters control the directions of information flow; for example, $\alpha_{AB}$ scales the information passed from network $A$ to network $B$. The cross-stitch network is shown in Figure 1b.

If both tasks operate at sentence level, an additional cross-stitch sharing unit is placed after the attention module. The $\alpha$-parameters are initialised with a bias towards favouring the information in the same network, with $\alpha_{AA} = \alpha_{BB} = 0.9$ and $\alpha_{AB} = \alpha_{BA} = 0.1$. These values are optimised

during training but remain static during testing.

### 4.3 Gated network

The cross-stitch network learns a single set of shared values for the $\alpha$-parameters during optimisation. As an alternative, we can construct a network that calculates these values dynamically for each input sentence, even at testing time. This allows the model greater flexibility and modulates the information flow depending on the particular input sentence.

In this architecture, shown in Figure 1c, the $\alpha$-parameters are replaced with gates (Liu et al., 2016). Each pair of parallel layers has two gates, where one modulates the information flow from the main to the auxiliary task, while the other controls the information flow in the opposite direction. For two jointly learned sentence-level tasks, two more gates are placed before the classification layer, operating on the sentence representations.

Equations (3)-(6) detail the gating mechanisms:

$$\mathbf{g}_A = \sigma(\mathbf{W}_A[\mathbf{h}_A; \mathbf{h}_B] + \mathbf{b}_A) \quad (3)$$

$$\widetilde{\mathbf{h}}_A = (1 - \mathbf{g}_A) \odot \mathbf{h}_A + \mathbf{g}_A \odot \mathbf{h}_B \quad (4)$$

$$\mathbf{g}_B = \sigma(\mathbf{W}_B[\mathbf{h}_A; \mathbf{h}_B] + \mathbf{b}_B) \quad (5)$$

$$\widetilde{\mathbf{h}}_B = (1 - \mathbf{g}_B) \odot \mathbf{h}_B + \mathbf{g}_B \odot \mathbf{h}_A \quad (6)$$

where $\mathbf{g}_A$ and $\mathbf{g}_B$ are the gates for task $A$ and $B$, $\mathbf{W}_A$ and $\mathbf{W}_B$ are weight matrices, $\mathbf{b}_A$ and $\mathbf{b}_B$ are bias vectors. The bias parameters of the gates are initialised with a bias towards one task.

## 5 Experiments and results

### 5.1 Experimental setup

**MTL training procedure** We apply pairwise joint learning, where at each step in the training process one of the two tasks is selected at random and a batch is sampled from that task. To distinguish main tasks from auxiliary tasks the loss of the auxiliary task is down-weighted by a factor $\lambda$ such that it comprises 10% of the loss of the main task. $\lambda$ is initialised with $\frac{1}{10}$ and computed dynamically as training progresses.

**Hyperparameters** The input to the recurrent network consists of concatenated ELMo and GloVe embeddings, with $1,024$ and 300 dimensions respectively. The recurrent encoder contains three Bi-LSTM layers with a dimensionality of 200. The models are trained using a batch size of 64 for $2,000$ steps and the Adam optimiser with initial learning rates of $4\text{e}{-3}$, $1\text{e}{-3}$ and $0.5\text{e}{-3}$ for metaphor detection, metaphor regression and emotion regression respectively. Models are selected based on validation data.

| Approach | Metaphor Task | |
|---|---|---|
| | Word ($F_1$) | Sent. ($r$) |
| Gao et al. (2018) | .726 | - |
| LSTM (single task) | .737 | .544 |
| Hard Sharing | | |
| + Valence | .740 | **.559** |
| + Arousal | .740 | **.558** |
| + Dominance | **.743** | **.560** |
| Cross-Stitch Network | | |
| + Valence | .741 | **.556** |
| + Arousal | .740 | **.558** |
| + Dominance | **.743** | **.563** |
| Gated Network | | |
| + Valence | **.742** | **.561** |
| + Arousal | .741 | **.558** |
| + Dominance | **.745** | **.560** |
| BERT (single task) | .763 | .604 |
| Hard Sharing | | |
| + Valence | **.769** | **.614** |
| + Arousal | .765 | .610 |
| + Dominance | **.768** | **.614** |

Table 2: System performance for the word- and sentence-level metaphor tasks using the $F_1$-score and Pearson's $r$ respectively. Statistically significant ($p < 0.05$) differences to the single task models are shown in boldface.

We employ the pretrained BERT Base model, whose inputs and hidden states have 768 dimensions. The model contains 12 Transformer layers and is fine-tuned as described by Devlin et al. (2019), using the Adam optimiser with an initial learning rate of $5\text{e}{-5}$ and a batch size of 32. BERT is fine-tuned for $3,000$ steps for the regression tasks and for $8,000$ steps for the word-level metaphor detection task. The difference is compensated for through down-scaling $\lambda$.

**Significance testing** We test for statistical significance using the one-sided approximate randomisation test (Edgington, 1969) for metaphor detection, and Williams's test (Williams, 1959) for regression tasks. For Williams's test we consider the number of samples to be the number of unique samples in the dataset. All performance measures reported are averages from models initialised with ten random seeds.

| Approach | Emotion Task | | |
|---|---|---|---|
| | Val. | Arous. | Dom. |
| Akhtar et al. (2018) | .616 | .355 | .237 |
| + Val., Arous., Dom. | .635 | .375 | .277 |
| Wu et al. (2019)† | .620 | .508 | .333 |
| LSTM (single task) | .728 | .557 | .373 |
| Hard Sharing | | | |
| + Metaphor (Token) | **.734** | **.564** | **.384** |
| + Metaphor (Sent.) | **.734** | .558 | **.388** |
| Cross-Stitch Network | | | |
| + Metaphor (Token) | **.737** | **.564** | **.388** |
| + Metaphor (Sent.) | **.735** | .558 | **.384** |
| Gated Network | | | |
| + Metaphor (Token) | **.738** | **.563** | **.389** |
| + Metaphor (Sent.) | **.735** | .560 | **.384** |
| BERT (single task) | .771 | .565 | .403 |
| Hard Sharing | | | |
| + Metaphor (Token) | **.779** | **.572** | **.420** |
| + Metaphor (Sent.) | **.778** | **.570** | **.417** |

Table 3: System performance for emotion regression tasks according to Pearson's $r$. Statistically significant ($p < 0.05$) differences to the single task model are shown in boldface. †Used 40% of the gold labels.

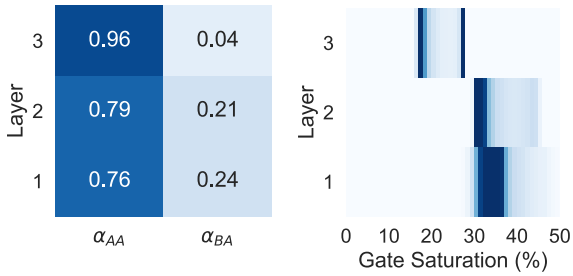


(a) Cross-Stitch Units (b) Gates

Figure 2: Illustration of the information flow in between the Bi-LSTM layers, for the dominance regression ($B$) and metaphor identification ($A$) tasks. Gate saturation % is calculated by averaging across the hidden dimensionality for every word in the test set.

### 5.2 Results

Table 2 presents the results for the two metaphor tasks. The STL setup already provides improvements over the current state of the art, but moreover, we see further improvements when MTL is introduced. Each MTL setup should be compared to the corresponding STL setup, which involves training the model on the metaphor task only. Regardless of the MTL architecture, the auxiliary task of dominance regression provides statistically significant ($p < 0.05$) improvements over the STL setup. Furthermore, valence regression provides significant improvements as well in a select number of setups. The largest improvement is achieved by replacing the recurrent encoder with BERT. This indicates that the rich contextual information learned by BERT in the pre-training phase is highly relevant for metaphor identification. For sentence-level metaphor regression, MTL setups consistently improve upon STL setups, indicating that the effect is not specific to the VUA metaphor corpus. Our MTL models outperform the previous state of the art in metaphor identification ($F_1$ of 0.726 on the VUA corpus Gao et al. (2018)) with both LSTM ($F_1$ of 0.745) and BERT ($F_1$ of 0.769) encoders. These results lend support to the hypothesis on the interaction of metaphor and emotion in semantic composition.

Table 3 presents the results for the emotion regression tasks. Again, STL setups perform strongly, and MTL architectures improve this even further. While the valence and dominance tasks consistently improve with the addition of the metaphor task, the improvements achieved on arousal regression are less stable. Once again, our MTL models outperform the best-performing existing approaches to dimensional emotion modelling (Akhtar et al., 2018; Wu et al., 2019), advancing the state of the art in this task. These results suggest that it may be beneficial to include information about metaphor into emotion analysis and, more broadly, sentiment analysis systems.

The differences between hard and soft parameter sharing manifest most with metaphor identification. This is possibly due to the explicit per-word sharing mechanisms in the top layer. While the bottom layers capture more general information, the top layer captures task-specific information. This can be seen through analysing the models' behaviour: the gating mechanism is most selective at the top layer and is the most active for emotion-laden words. The cross-stitch units gradually share less information from the bottom to the top layer. This behaviour is illustrated in Figure 2.

## 6 Data analysis and discussion

**Metaphor identification** Most improvements in word-level metaphor identification are achieved by corrections from a literal prediction in STL to metaphorical in MTL. To establish this fact,

| Auxiliary Task | Sentence | STL | MTL | Gold |
|---|---|---|---|---|
| Valence | It is sad, and somewhat ominous, that so little of that should have been reflected in the **sombre** statement (. . . ) | L | M | M |
| Arousal | There is still endless dithering on how broad a **safety** net Britain will extend to its citizens. | L | M | M |
| Dominance | In a **bare**, mud-walled cell, sitting on the floor, is Tepilit. | L | M | M |

Table 4: Examples of how MTL improves over STL on metaphor identification, with gold metaphors underlined and corrections by MTL bolded. L refers to literal and M to metaphorical.

| Main Task | Sentence | STL | MTL | Gold |
|---|---|---|---|---|
| Valence | Scam lures victims with free puppy offer. | .48 | .41 | .40 |
| Valence | (. . . ) looking at me like I was breaking her poor, sweet heart. | .52 | .42 | .24 |
| Arousal | (. . . ) the authors depict her as bewitchingly beautiful. | .51 | .57 | .61 |
| Dominance | In fact, I've never felt so out of place in all my life. | .73 | .54 | .17 |
| Dominance | Frustration rises as North Korea nuclear talks stall. | .46 | .41 | .30 |

Table 5: Examples of how MTL using metaphor identification improves over STL emotion prediction, with predicted metaphors underlined.

we pooled predictions on test data for ten models trained with different random seeds. The STL outputs were compared to MTL outputs to determine whether corrections changed from literal to metaphorical or the other way around. Examining this set of corrected predictions gives us insight into the behaviour of the MTL model.

While some improvements hold across all emotion dimensions, others are unique to each. Among the corrections unique to each dimension we find multiple terms indicative of the dimension: for valence regression we find improvements for attractiveness (*wise*, *attractive*) and averseness indicators (*severity*, *drain*) and for arousal excitement (*flame*, *crisis*) or calmness indicators (*empty*, *rest*). Dominance corrects various terms related to control (*capable*, *courtesy*) and submissiveness (*owe*, *puny*). We selected the presented example terms from the set of corrections described previously, and established the scores using the ANEW lexicon (Warriner et al., 2013).

Table 4 illustrates model decisions corrected through joint learning. Examples for valence and arousal illustrate how emotion-laden words participate in metaphors, such as "a *sombre* statement" or "a *safety* net". The example for dominance illustrates that control indicators may seem less related to emotion (e.g. *bare*), but carry affect through the contexts in which they are embedded.

Overall, introducing emotion improves performance of metaphor identification, as we expected. Dominance appears to be most beneficial, while arousal contributes the least. This might be due to the fact that arousal (and to some extent valence) predictions rely strongly on explicit sentiment markers in text. In contrast, dominance prediction requires the model to learn richer semantic representations. In our models, this manifests in the sparseness of the attention distribution: models trained using arousal have the most sparse attention patterns as measured through the Gini index sparsity measure (Hurley and Rickard, 2009), while models trained using dominance have the least sparse attention patterns.

Dominance regression is the most complex task and yet the most beneficial performance-wise, despite it sometimes being discarded by previous research in favour of the VA emotion model. Several studies argue for the inclusion of dominance in emotion analysis (Stamps III, 2005; Bakker et al., 2014). Bakker et al. (2014) emphasise that while valence and arousal highlight the affective and cognitive aspects of emotion, dominance is related to environmental factors and social influences. This relates to the role of metaphorical framing in the social world, e.g. in politics. Metaphor allows us to highlight certain aspects of a target domain and mask others, encouraging specific inferences (Lakoff, 1991; Entman, 1993). These in turn activate emotional considerations (Boeynaems et al.,

2017), allowing the metaphor to steer the emotions recalled and affecting the evaluation of the argument made and persuasiveness of the speaker (Marcus, 2000).

**Emotion prediction** Table 5 lists examples for which including metaphor identification improved performance on emotion regression. The examples for valence show that metaphor can be used to describe an emotion ("to *break* one's *heart*") explicitly or to convey an implicit judgement (e.g. *luring*). For arousal, example improvements include applying excitement indicators to objects or concepts, such as "being *bewitchingly* beautiful". Examples for dominance regression indicate the importance of power; one has no control over *stalling* or "feeling *out of place*".

While joint learning improves the estimations overall, it also introduces some new errors. Although generally the presence of metaphor makes phrases more emotionally evocative (Mohammad et al., 2016), this does not always hold. Lexicalised metaphors – e.g. *up* in "grades going *up*" – are no longer viewed as metaphorical by lay language users and throw the models off. Other common errors introduced are related to misinterpreting the perspective – e.g. confusing "to be *knocked out*" and "to *knock out*" – and the direction in which words contribute to emotion – e.g. negative metaphorical terms can contribute to the positive sentiment and vice versa, such as in "Stop cancer with a *shot*".

## 7 Conclusion

In this paper, we introduced the first compositional deep learning model to jointly capture the phenomena of metaphor and emotion. We considered metaphor tasks at word and sentence level and modelled emotion through the dimensional model of valence, arousal and dominance. We experimented with multiple MTL techniques, regulating the information flow between the two tasks.

We demonstrated that the proposed methods advance the state of the art for the tasks of metaphor identification and emotion regression. Both tasks benefit from joint learning, with the emotion dimension of dominance contributing most to metaphor and benefiting most from metaphor. Our results support the hypothesis on the interaction of metaphor and emotion, and suggest that it may be beneficial to incorporate a model of metaphor into emotion- and sentiment-related NLP applications in the future.

## Acknowledgments

Martha Lewis gratefully acknowledges funding from NWO Veni grant 'Metaphorical Meanings for Artificial Agents'. Marek Rei's research is supported by Cambridge English via the ALTA Institute.

## References

Md Shad Akhtar, Deepanway Ghosal, Asif Ekbal, and Pushpak Bhattacharyya. 2018. A multi-task ensemble framework for emotion, sentiment and intensity prediction. *arXiv preprint arXiv:1808.01216*.

Lisa Aziz-Zadeh and Antonio Damasio. 2008. Embodied semantics for actions: Findings from functional brain imaging. *Journal of Physiology - Paris*, 102(1-3).

Iris Bakker, Theo van der Voordt, Peter Vink, and Jan de Boon. 2014. Pleasure, arousal, dominance: Mehrabian and Russell revisited. *Current Psychology*, 33(3):405–421.

Isabelle Blanchette, Kevin Dunbar, John Hummel, and Richard Marsh. 2001. Analogy use in naturalistic settings: The influence of audience, emotion and goals. *Memory and Cognition*, 29(5).

Amber Boeynaems, Christian Burgers, Elly A Konijn, and Gerard J Steen. 2017. The effects of metaphorical framing on political persuasion: A systematic literature review. *Metaphor and Symbol*, 32(2):118–134.

George Aaron Broadwell, Umit Boz, Ignacio Cases, Tomek Strzalkowski, Laurie Feldman, Sarah Taylor, Samira Shaikh, Ting Liu, Kit Cho, and Nick Webb. 2013. Using imageability and topic chaining to locate metaphors in linguistic corpora. In *International Conference on Social Computing, Behavioral-Cultural Modeling, and Prediction*, pages 102–110. Springer.

Sven Buechel and Udo Hahn. 2016. Emotion analysis as a regression problem—dimensional models and their implications on emotion representation and metrical evaluation. In *Proceedings of the Twenty-second European Conference on Artificial Intelligence*, pages 1114–1122. IOS Press.

Sven Buechel and Udo Hahn. 2017. Emobank: Studying the impact of annotation perspective and representation format on dimensional emotion analysis. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 578–585.

Luana Bulat, Stephen Clark, and Ekaterina Shutova. 2017. Modelling metaphor with attribute-based semantics. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 523–528.

Rich Caruana. 1993. Multitask learning: A knowledge-based source of inductive bias. In *Proceedings of the International Conference on Machine Learning*.

Francesca MM Citron and Adele E Goldberg. 2014. Metaphorical sentences are more emotionally engaging than their literal counterparts. *Journal of cognitive neuroscience*, 26(11):2585–2595.

Francesca MM Citron, Jeremie Güsten, Nora Michaelis, and Adele E Goldberg. 2016. Conventional metaphors in longer passages evoke affective brain response. *NeuroImage*, 139:218–230.

Elizabeth Crawford. 2009. Conceptual Metaphors of Affect. *Emotion Review*, 1(2).

Taner Danisman and Adil Alpkocak. 2008. Feeler: Emotion classification of text using vector space model. In *AISB 2008 Convention Communication, Interaction and Social Intelligence*, volume 1, page 53.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Erik-Lân Do Dinh, Steffen Eger, and Iryna Gurevych. 2018. Killing four birds with two stones: Multi-task learning for non-literal language detection. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1558–1569.

Erik-Lân Do Dinh and Iryna Gurevych. 2016. Token-level metaphor detection using neural networks. In *Proceedings of the Fourth Workshop on Metaphor in NLP*, pages 28–33.

Jonathan Dunn. 2013. Evaluating the premises and results of four metaphor identification systems. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 471–486. Springer.

Eugene S Edgington. 1969. Approximate randomization tests. *The Journal of Psychology*, 72(2):143–149.

Paul Ekman. 1992. An argument for basic emotions. *Cognition & Emotion*, 6(3-4):169–200.

Robert M Entman. 1993. Framing: Toward clarification of a fractured paradigm. *Journal of Communication*, 43(4):51–58.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2018. Examining the tip of the iceberg: A data set for idiom translation. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation*, pages 925–929.

Lynn Fainsilber and Andrew Ortony. 1987. Metaphorical uses of language in the expression of emotions. *Metaphor and Symbol*, 2(4):239–250.

Susan R Fussell and Mallie M Moss. 1998. Figurative language in emotional communication. *Social and cognitive approaches to interpersonal communication*, pages 113–141.

Ge Gao, Eunsol Choi, Yejin Choi, and Luke Zettlemoyer. 2018. Neural Metaphor Detection in Context. In *Conference on Empirical Methods in Natural Language Processing*.

Andrew Gargett and John Barnden. 2015. Modeling the interaction between sensory and affective meanings for detecting metaphor. In *Proceedings of the Third Workshop on Metaphor in NLP*, pages 21–30.

Raymond W Gibbs Jr, John S Leggitt, and Elizabeth A Turner. 2002. What's special about figurative language in emotional communication? In *The verbal communication of emotions*, pages 133–158. Psychology Press.

E Dario Gutierrez, Ekaterina Shutova, Tyler Marghetis, and Benjamin Bergen. 2016. Literal and metaphorical senses in compositional distributional semantic models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 183–193.

Dirk Hovy, Shashank Shrivastava, Sujay Kumar Jauhar, Mrinmaya Sachan, Kartik Goyal, Huying Li, Whitney Sanders, and Eduard Hovy. 2013. Identifying metaphorical word use with tree kernels. In *Proceedings of the First Workshop on Metaphor in NLP*, pages 52–57.

Niall Hurley and Scott Rickard. 2009. Comparing measures of sparsity. *IEEE Transactions on Information Theory*, 55(10):4723–4741.

Nancy Ide, Collin Baker, Christiane Fellbaum, Charles Fillmore, and Rebecca Passonneau. 2008. Masc: The manually annotated sub-corpus of American English. In *6th International Conference on Language Resources and Evaluation*, pages 2455–2460. European Language Resources Association (ELRA).

Mbemba Jabbi, Jojanneke Bastiaansen, and Christian Keysers. 2008. A common anterior insula representation of disgust observation, experience and imagination shows divergent functional connectivity pathways. *PLoS ONE*, 3(8):e2939.

Sunghwan Mac Kim, Alessandro Valitutti, and Rafael A Calvo. 2010. Evaluation of unsupervised emotion models to textual affect recognition. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 62–70. Association for Computational Linguistics.

Zoltán Kövecses. 2003. *Metaphor and emotion: Language, culture, and body in human feeling*. Cambridge University Press, Cambridge.

Zornitsa Kozareva. 2013. Multilingual affect polarity and valence prediction in metaphor-rich texts. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 682–691.

George Lakoff. 1991. Metaphor and war: The metaphor system used to justify war in the Gulf. *Peace Research*, 23(2/3):25–32.

George Lakoff and Mark Johnson. 1983. Metaphors we live by. *University of Chicago Press*, 59.

Geoffrey Leech. 1992. 100 million words of English: the British National Corpus. *Language Research*, 28(1):1–13.

Chee Wee Ben Leong, Beata Beigman Klebanov, and Ekaterina Shutova. 2018. A report on the 2018 VUA metaphor detection shared task. In *Proceedings of the Workshop on Figurative Language Processing*, pages 56–66.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 2873–2879. AAAI Press.

Nikolaos Malandrakis, Alexandros Potamianos, Elias Iosif, and Shrikanth Narayanan. 2013. Distributional semantic models for affective text analysis. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(11):2379–2392.

George E Marcus. 2000. Emotions in politics. *Annual Review of Political Science*, 3(1):221–250.

Albert Mehrabian. 1996. Pleasure-arousal-dominance: A general framework for describing and measuring individual differences in temperament. *Current Psychology*, 14(4):261–292.

Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. 2016. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3994–4003.

Saif Mohammad, Ekaterina Shutova, and Peter Turney. 2016. Metaphor as a medium for emotion: An empirical study. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*, pages 23–33.

Michael Mohler, David Bracewell, Marc Tomlinson, and David Hinote. 2013. Semantic signatures for example-based linguistic metaphor detection. In *Proceedings of the First Workshop on Metaphor in NLP*, pages 27–35.

Michael Mohler, Mary Brunson, Bryan Rink, and Marc Tomlinson. 2016. Introducing the LCC metaphor datasets. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation*, pages 4221–4227.

Georgios Paltoglou, Mathias Theunis, Arvid Kappas, and Mike Thelwall. 2013. Predicting emotional responses to long informal text. *IEEE Transactions on Affective Computing*, 1(4):106–115.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Isidoros Perikos and Ioannis Hatzilygeroudis. 2016. Recognizing emotions in text using ensemble of classifiers. *Engineering Applications of Artificial Intelligence*, 51:191–201.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237.

Marek Rei, Luana Bulat, Douwe Kiela, and Ekaterina Shutova. 2017. Grasping the finer point: A supervised similarity network for metaphor detection. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1537–1546.

Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. 2019. Latent multi-task architecture learning. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019*, pages 4822–4829.

Ekaterina Shutova, Douwe Kiela, and Jean Maillard. 2016. Black holes and white rabbits: Metaphor identification with visual features. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 160–170.

Ekaterina Shutova, Lin Sun, and Anna Korhonen. 2010. Metaphor identification using verb and noun clustering. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1002–1010. Association for Computational Linguistics.

Arthur E Stamps III. 2005. In search of dominance: The case of the missing dimension. *Perceptual and motor skills*, 100(2):559–566.

Gerard Steen. 2010. *A method for linguistic metaphor identification: From MIP to MIPVU*, volume 14. John Benjamins Publishing.

Carlo Strapparava and Rada Mihalcea. 2007. Semeval-2007 task 14: Affective text. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 70–74.

Tomek Strzalkowski, Samira Shaikh, Kit Cho, George Aaron Broadwell, Laurie Feldman, Sarah Taylor, Boris Yamrom, Ting Liu, Ignacio Cases, Yuliya Peshkova, et al. 2014. Computing affect in metaphors. In *Proceedings of the Second Workshop on Metaphor in NLP*, pages 42–51.

Paul H. Thibodeau and Lera Boroditsky. 2011. Metaphors we think with: The role of metaphor in reasoning. *PLoS ONE*, 6(2):e16782.

Yulia Tsvetkov, Leonid Boytsov, Anatole Gershman, Eric Nyberg, and Chris Dyer. 2014. Metaphor detection with cross-lingual model transfer. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 248–258.

Peter D Turney, Yair Neuman, Dan Assaf, and Yohai Cohen. 2011. Literal and metaphorical sense identification through concrete and abstract context. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 680–690. Association for Computational Linguistics.

Jin Wang, Bo Peng, and Xuejie Zhang. 2018. Using a stacked residual LSTM model for sentiment intensity prediction. *Neurocomputing*, 322:93–101.

Jin Wang, Liang-Chih Yu, K Robert Lai, and Xuejie Zhang. 2016. Dimensional sentiment analysis using a regional CNN-LSTM model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 225–230.

Amy Beth Warriner, Victor Kuperman, and Marc Brysbaert. 2013. Norms of valence, arousal, and dominance for 13,915 English lemmas. *Behavior research methods*, 45(4):1191–1207.

Evan James Williams. 1959. *Regression Analysis*. Wiley publication in applied statistics. Wiley.

Chuhan Wu, Fangzhao Wu, Sixing Wu, Zhigang Yuan, Junxin Liu, and Yongfeng Huang. 2019. Semi-supervised dimensional sentiment analysis with variational autoencoder. *Knowledge-Based Systems*, 165:30–39.

Ali Yadollahi, Ameneh Gholipour Shahraki, and Osmar R Zaiane. 2017. Current state of text sentiment analysis from opinion to emotion mining. *ACM Computing Surveys (CSUR)*, 50(2):25.

Yuxiang Zhang, Jiamei Fu, Dongyu She, Ying Zhang, Senzhang Wang, and Jufeng Yang. 2018. Text emotion distribution learning via multi-task convolutional neural network. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 4595–4601. AAAI Press.

# How well do NLI models capture verb veridicality?

**Alexis Ross**
Harvard University
alexis_ross@college.harvard.edu

**Ellie Pavlick**
Brown University
ellie_pavlick@brown.edu

## Abstract

In natural language inference (NLI), contexts are considered *veridical* if they allow us to infer that their underlying propositions make true claims about the real world. We investigate whether a state-of-the-art natural language inference model (BERT) learns to make correct inferences about veridicality in verb-complement constructions. We introduce an NLI dataset for veridicality evaluation consisting of 1,500 sentence pairs, covering 137 unique verbs. We find that both human and model inferences generally follow theoretical patterns, but exhibit a systematic bias towards assuming that verbs are veridical–a bias which is amplified in BERT. We further show that, encouragingly, BERT's inferences are sensitive not only to the presence of individual verb types, but also to the syntactic role of the verb, the form of the complement clause (*to-* vs. *that*-complements), and negation.

## 1 Introduction

A context is *veridical* when the propositions it contains are taken to be true, even if not explicitly asserted. For example, in the sentence *"He does not know that the answer is 5"*, *"know"* is veridical with respect to *"The answer is 5"*, since a speaker cannot felicitously say the former sentence unless they believe the latter proposition to be true. In contrast, *"think"* would not be veridical here, since *"He does not think that the answer is 5"* is felicitous whether or not it is taken to be true that *"The answer is 5"*. Understanding veridicality requires semantic subtlety and is still an open problem for computational models of natural language inference (NLI) (Rudinger et al., 2018).

This paper deals specifically with veridicality in verb-complement constructions. Prior work in this area has focused on characterizing verb classes–e.g. factives like *"know that"* (Kiparsky and Kiparsky, 1968) and implicatives like *"manage to"* (Karttunen, 1971)–and on incorporating such lexical semantic information into computational models (MacCartney and Manning, 2009). However, increasingly, linguistic evidence suggests that inferences involving veridicality rely heavily on non-lexical information and are better understood as a graded, pragmatic phenomenon (de Marneffe et al., 2012; Tonhauser et al., 2018).

Thus, in this paper, we revisit the question of whether neural models of natural language inference–which are not explicitly endowed with knowledge of verbs' lexical semantic categories–learn to make inferences about veridicality consistent with those made by humans. We solicit human judgements on 1,500 sentence pairs involving 137 verb-complement constructions. Analysis of these annotations provides new evidence of the importance of pragmatic inference in modeling veridicality judgements. We use our collected annotations to analyze how well a state-of-the-art NLI model (BERT, Devlin et al., 2018) is able to mimic human behavior on such inferences. The results suggest that, while not yet solved, BERT represents non-trivial properties of veridicality in context. Our primary contributions are:

- We collect a new NLI evaluation set of 1,500 sentence pairs involving verb-complement constructions (§4).[1]
- We discuss new analysis of human judgements of veridicality and implications for NLI system development going forward (§5).
- We evaluate the state-of-the-art BERT model on these inferences and present evidence that, while there is still work to be done, the

[1] https://github.com/alexisjihyeross/verb_veridicality



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2230–2240, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

model appears to capture non-trivial properties about verbs' veridicality in context (§6).

## 2 Background and Related Work

There is significant work, both in linguistics and NLP, on veridicality and closely-related topics (factuality, entailment, etc). We view past work on veridicality within NLP as largely divisible into two groups, which align with two differing perspectives on the role of the NLI task: the sentence-meaning perspective and the speaker-meaning perspective. Briefly, the *sentence meaning* approach to NLI takes the position that NLP systems should strive to model the aspects of a sentence's semantics which are closely derivable from the lexicon and which hold independently of context (Zaenen et al., 2005). In contrast, the *speaker meaning* approach to NLI takes the position that NLP systems should prioritize representation of the goal-directed meaning of a sentence within the context in which it was generated (Manning, 2006). Work on veridicality which aligns with the sentence-meaning perspective tends to focus on characterizing verbs according to their lexical semantic classes (or "signatures"), while work which aligns with the speaker-meaning approach focuses on representing "world knowledge" and evaluating inferences in naturalistic contexts.

**Lexical Semantics (Sentence Meaning).** Most prior work treats veridicality as a lexical semantic phenomenon. Such work is largely based on lexicons of verb signatures which specify the types of inferences licensed by individual verbs (Karttunen, 2012; Nairn et al., 2006; Falk and Martin, 2017). White and Rawlins (2018); White et al. (2018) evaluated neural models' ability to carry out inferences in line with these signatures, making use of templatized "semantically bleached" stimuli (e.g. *"someone knew something"*) in order to avoid confounds introduced by world knowledge and pragmatic inference. McCoy et al. (2019) perform a similar study, though without specific focus on veridicality lexicons.

Most applied work related to veridicality also falls under the lexical semantic approach. In nearly all cases, relevant system development involves explicit incorporation of verb lexicons and associated logical inference rules. MacCartney and Manning (2009); Angeli and Manning (2014); and others incorporated knowledge of verb signatures within a natural logic framework (MacCartney, 2009; Sánchez Valencia, 1991) in order to perform natural language inference. Richardson and Kuhn (2012) incorporated signatures into a semantic parsing system. Several recent models of event factuality similarly make use of veridicality lexicons as input to larger machine-learned systems for event factuality (Saurí and Pustejovsky, 2012; Lotan et al., 2013; Stanovsky et al., 2017; Rudinger et al., 2018). Cases et al. (2019) used nested veridicality inferences as a test case for a meta-learning model, again assuming verb signatures as "meta information" known *a priori*.

**Pragmatics (Speaker Meaning).** Geis and Zwicky (1971) observed that implicative verbs often give rise to "invited inferences", beyond what is explainable by the lexical semantic type of the verb. For example, on hearing *"He did not refuse to speak"*, one naturally concludes that *"He spoke"* unless additional qualifications are made (e.g. *"...he just didn't have anything to say"*). de Marneffe et al. (2012) explored this idea in depth and presented evidence that such pragmatic inferences are both pervasive and annotator-dependent, but nonetheless systematic enough to be relevant for NLP models. Karttunen et al. (2014) makes similar observations specifically in the case of evaluative adjectives, and Pavlick and Callison-Burch (2016) specifically in the case of simple implicative verbs. In non-computational linguistics, Simons et al. (2017, 2010); Tonhauser et al. (2018) take a strong stance and argue that veridicality judgements are entirely pragmatic, dependent solely on the question under discussion (QUD) within the given discourse.

**This Work.** This paper assumes the speaker-meaning approach: we take the position that models which consistently mirror human inferences about veridicality in context can be said to understand veridicality in general. We acknowledge that the question of what is the "right" approach to NLI has existed since the original definition of the recognizing textual entailment (RTE) task (Dagan et al., 2006) and remains open. However, there has been a *de facto* endorsement of the speaker-meaning definition, evidenced by the widespread adoption of NLI datasets which favor informal, "natural" inferences over prescriptivist annotation guidelines (Manning, 2006; Bowman et al., 2015; Williams et al., 2018). (Note, recently, there have been explicit endorsements as well; see Westera

| | | | |
|---|---|---|---|
| Factive | He **realized that** he had to leave this house. | $\rightarrow$ | He had to leave this house. |
| $[+/+]$ | He **did not realize that** he had to leave this house. | $\rightarrow$ | He had to leave this house. |
| Implic. | At that moment, I **happened to** look up. | $\rightarrow$ | At that moment, I looked up. |
| $[+/-]$ | At that moment, I **did not happen to** look up. | $\rightarrow \neg$ | At that moment, I looked up. |
| Implic. | He **refused to** do the same. | $\rightarrow \neg$ | He did the same. |
| $[-/\circ]$ | He **did not refuse to** do the same. | $\not\rightarrow$ | He did the same. |
| NA | Many **felt that** its inclusion was a mistake. | $\not\rightarrow$ | Its inclusion was a mistake. |
| $[\circ/\circ]$ | Many **did not feel that** its inclusion was a mistake. | $\not\rightarrow$ | Its inclusion was a mistake. |

Table 1: Examples of several verb signatures and illustrative contexts for each. Signature $s1/s2$ denotes that the complement will project with polarity $s1$ in a positive environment and polarity $s2$ in a negative environment.

and Boleda (2019)). Thus, from this perspective, we ask: do NLI models which are *not* specifically endowed with lexical semantic knowledge pertaining to veridicality nonetheless learn to model this semantic phenomenon?

## 3 Projectivity and Verb Signatures

Veridicality is typically treated as a lexical semantic property of verbs, specified by the verb's *signature*. These signatures can indicate that a verb licenses positive ($+$), negative ($-$), or neutral ($\circ$) inferences. Specifically, Karttunen (2012) defines these as two-bit signatures, to reflect that verbs[2] may behave differently in positive vs. negative environments. For example, a factive verb construction like *"know that"* has a $+/+$ signature, indicating that the complement projects positively in both positive and negative environments. That is, both *"He knows that the answer is 5"* and *"He does not know that the answer is 5"* imply that *"The answer is 5"*. In contrast, a verb like *"manage to"* has the signature $+/-$ since, in a positive environment, the complement projects (*"I managed to pass"*$\rightarrow$*"I passed"*) but, in a negative environment, the *negation* of the complement projects (*"I did not manage to pass"*$\rightarrow \neg$*"I passed"*). Other verbs may exhibit veridicality only in positive or negative environments but not in both. For example, *"refuse to"* has signature $-/\circ$: *"She refused to dance"*$\rightarrow \neg$*"She danced"*, but *"She did not refuse to dance"* neither implies nor contradicts the claim *"She danced"*. Still other verbs are entirely non-veridical ($\circ/\circ$). For example, *"hope to"* is not expected to license any inferences about the truth of its complement. We consider 8 signatures[3] in total. Table 1 provides several examples. Table 2 lists all of the signatures and the corresponding verbs we consider.

## 4 Data

For our analysis, we collect an NLI dataset for veridicality evaluation derived from the MNLI corpus. This data is publicly available at https://github.com/alexisjihyeross/verb_veridicality.

### 4.1 Generating NLI Pairs

We generate NLI-style premise/hypothesis pairs based on sentences drawn from the train+dev splits of the MultiNLI (Williams et al., 2018) corpus. Specifically, we collect all sentences appearing in MNLI[4] which contain any verb-complement construction, e.g. any sequence matching the pattern: `verb` {*"to"*|*"that"*} {`VP`|`S`}. Since we aim to manipulate the environment (positive vs. negative) ourselves in a controlled manner, we filter out sentences which already contain explicit negation words (e.g. *"no"*, *"not"*, *"never"*, *"n't"*), conditionals (*"if"*), and passive constructions (*"was intended to"*). From the selected sentences, we take a stratified sample over the lemma of `verb`. For use in later analysis, we associate each lemma with a signature using a manually-curated dictionary[5] of implicative and factive verbs, and assign the signature of $\circ/\circ$ to verbs which do not appear in the dictionary. We sample a set of sentences for every lemma, taking

[2] Karttunen (2012) actually discusses these signatures for implicative verbs only. We adopt the notation, but use it in for factives and uncategorized verbs as well.

[3] Seven from http://web.stanford.edu/group/csli_lnr/Lexical_Resources/, plus $\circ/\circ$.

[4] We removed sentences from the telephone genre, since the parses on these tended to be noisy and thus our manipulations often yielded incorrect or difficult to interpret sentences.

[5] http://web.stanford.edu/group/csli_lnr/Lexical_Resources/

no more than 40 sentences per lemma, and weighting our sampling to prefer shorter sentences, in order to reduce cognitive load on our raters.

+/+ realize that (34) know that (32) remember that (17) find that (12) notice that (12) reveal that (12) acknowledge that (11) admit that (11) learn that (11) observe that (11) see that (11) note that (10) recognize that (10) understand that (10) discover that (8) +/− manage to (30) begin to (12) serve to (11) start to (11) dare to (8) use to (7) get to (6) come to (5) −/+ forget to (15) fail to (10) ∘/+ suspect that (11) explain that (10) mean to (10) predict that (10) ∘/− attempt to (28) −/∘ refuse to (36) decline to (12) remain to (7) +/∘ show that (12) confirm that (11) demonstrate that (10) ensure that (9) help to (9) tend to (8) ∘/∘ try to (34) hope that (20) hope to (18) mention that (14) like to (12) continue to (12) expect that (12) agree that (12) love to (12) reply that (12) conclude that (12) say that (12) complain that (12) speculate that (12) state that (12) suggest that (12) worry that (12) mean that (12) intend to (11) insist that (11) imply that (11) indicate that (11) plan to (11) promise to (11) prove to (11) saw that (11) seem that (11) tell that (11) think that (11) felt that (11) write that (11) decide to (11) assume that (11) believe that (11) assert that (11) concern that (11) estimate that (11) convince that (11) decide that (11) appear that (11) argue that (11) aim to (11) cease to (10) strive to (10) proceed to (10) choose to (10) seem to (10) prove that (10) provide that (10) seek to (10) appear to (10) comment that (10) contend that (10) want to (10) doubt that (10) feel that (10) fear that (10) agree to (10) announce that (9) claim that (9) struggle to (9) hear that (9) propose to (9) wish to (9) say to (9) turn to (8) wish that (8) work to (8) advise that (8) move to (8) claim to (8) expect to (8) report that (8) happen to (8) propose that (8) hold that (8) declare that (8) prefer to (8) need to (8) give that (7) deserve to (7) threaten to (7) exist to (7) be that (7) prepare to (6) wait to (6) pretend to (6) ask to (6) return to (6) request that (5) demand that (4) recommend that (4) require that (4)

Table 2: 137 verbs belonging to 8 signatures. Parentheses denote number of contexts in which each verb appears in our final, annotated dataset (§4)

We consider each sampled sentence $S$ to be a candidate premise. We then generate premise/hypothesis pairs as follows. We use the parse tree provided by MNLI to extract the complement clause $C$. When needed, we inflect[6] the verb in the complement to match the tense of the main verb. We then generate two $\langle p, h\rangle$ pairs: the sentence and the complement as-is $\langle S, C\rangle$, and the negated sentence plus the complement $\langle \neg S, C\rangle$. For example, given an original sentence like *"He knows that the answer is 5"*, we would generate two $\langle p, h\rangle$ pairs: $\langle$*"He knows that the answer is 5"*, *"The answer is 5"*$\rangle$ and $\langle$*"He does not know that the answer is 5"*, *"The answer is 5"*$\rangle$. The examples shown in Table 1 illustrate $\langle p, h\rangle$ pairs drawn from our dataset, generated this way.

### 4.2 Annotation

For each $\langle p, h\rangle$ pair, we collect human judgements on Amazon Mechanical Turk. We have raters label entailment on a 5-point likert scale in which $-2$ means that $h$ is definitely *not* true given $p$ and 2 means that $h$ is definitely true given $p$. This ordinal labelling scheme[7] matches prior work on common sense inference (Zhang et al., 2017), and on veridicality specifically (de Marneffe et al., 2012). We do not provide examples for boundary cases (the difference between $-2, -1$ or $1, 2$) to avoid biasing raters by providing explicit guidance about the extent to which common sense can factor in. Raters have the option of indicating with a check box that one or both sentences does not make sense, and thus that they are unable to judge. We require that raters have had at least 100 approved tasks, have maintained an 98% approval rating, and are located in an primarily English-speaking country (US, AU, GB, CA). We collect three annotations per $p/h$ pair, and pay \$0.10 per set of six pairs labelled.

**Quality Controls and Exclusion Criteria.** We remove all sentence pairs in which one or more raters checked the "does not make sense" box. We remove sentences from our analysis unless both the $\langle S, C\rangle$ and the $\langle \neg S, C\rangle$ pairs passed this filter. Finally, we remove verbs from our analysis which, after the above filtering, do not appear with at least 4 sentences (i.e. 8 $p/h$ pairs). Our final dataset contains 137 verb types across 1,498 sentences (2,996 pairs). Table 2 lists the verbs included in our dataset and the number of sentences in which each appears.[8] To measure inter-rater agreement, for each example and each of the three raters assigned to the example, we calculated the correlation between that rater's score and the averaged score of the other two raters. The Spearman correlation among raters, averaged across the three raters for each example, was 0.78 for positive contexts and 0.74 for negative contexts.

### 4.3 Aggregation

We take the mean of the three human judgements for each sentence pair. We then represent each verb $v$ (in the context of a given sentence $S$) using a continuous analog of the projectivity signatures discussed in §3. That is, we take the mean score for $\langle S, C\rangle$ as a measure of the veridicality of $v$ (in the context of $S$) in a positive environment, and the mean score for $\langle \neg S, C\rangle$ as a measure of the veridicality in a negative environment. For example, given S=*"David Plotz failed to change my mind"* and $C$=*"David Plotz changed my mind"*, we get a soft projectivity "signature" of $-2.0/1.67$, which is consistent with the expected (discrete) $-/+$ signature for *"fail to"*.

[6] https://www.clips.uantwerpen.be/pages/pattern-en

[7] Full annotation guidelines in Supplementary.

[8] See Supplementary for number of contexts that were excluded for each verb.

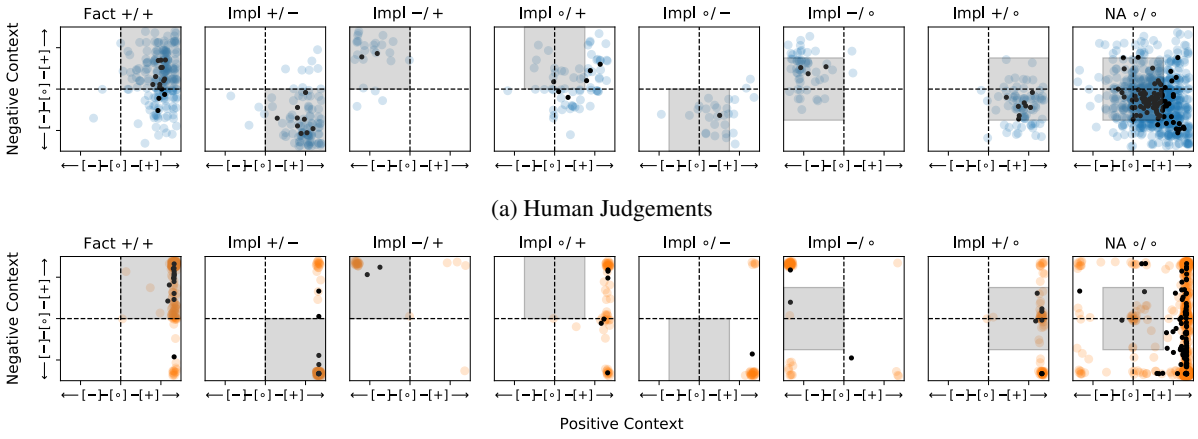


(a) Human Judgements

(b) Model Predictions

Figure 1: Human judgements (top, blue) and model predictions (bottom, orange) for verbs in each category. Gray squares denote the region in which judgements are expected fall, given the signature. Each colored dot corresponds to a single context (verb within a specific sentence); each black dot corresponds to a single verb (averaged score over all contexts in which it was judged).

## 5 Analysis of Human Judgements

Figure 1a plots these soft veridicality signatures for each sentence. We see that, averaged across all the contexts in which they are judged, verbs tend to behave as expected given their assigned lexical semantic signature. However, we observe two noteworthy trends, discussed below. We note that these observations are consistent with arguments made by de Marneffe et al. (2012) about the strong effects of pragmatics on veridicality judgments.

**Veridicality Bias.** First, we observe a systematic "veridicality bias", in which inferences about complements are often made (positive or negative), even in environments when the expectation is that the verb is non-veridical (∘ signature). This trend is most evident in the case of verbs with ∘/∘ signatures, for example, *"think that"*, *"want to"*. While embedding under such verbs should not license any inferences about the truth of the complement, we observe that, in practice, these verbs tend to behave like +/− verbs. That is, the complement is taken as true in positive environments and as false in negative environments. Table 3 shows some examples for which this is the case.

**Within-Verb Variation.** Second, we observe that, while individual verb types tend to behave in line with their expected signatures *on average*, signatures provide a weak signal for predicting the inferences licensed by the verb in any sentence individually. That is, within each signature, we see high variance across contexts, in all cases spanning at least 2 points (on our −2 to 2 scale). Table 4 shows examples of words receiving different signatures based on context. Quantitatively, in an ordinary least squares regression[9], we find that using verb signature alone to predict the human judgments in a given context explains only a small amount of the observed variation ($R^2 \approx 0.11$).[10]

| | |
|---|---|
| [+] (1.7) | The GAO has **indicated that** it is unwilling to compromise.<br>→ It is unwilling to compromise. |
| [−] (−1.0) | The GAO has **not indicated that** it is unwilling to compromise.<br>→ ¬ It is unwilling to compromise. |
| [+] (1.3) | But most visitors **prefer to** linger in Formentera.<br>→ But most visitors linger in Formentera. |
| [−] (−1.3) | But most visitors do not **prefer to** linger in Formentera.<br>→ ¬ But most visitors linger in Formentera. |

Table 3: Examples of verbs which are expected to be ∘/∘, but which behave like +/− in context. We refer to this trend as a general "veridicality bias".

[9] statsmodels.regression.linear_model.OLS.html

[10] For context, using the verb type itself produced $R^2 \approx 0.72$. We experimented with other contextual features in combination with linguistic category and/or verb type (e.g. tense of the main verbs, first vs. third person subjects, etc.) to try to improve the fit of the model, but did not find any note-

| | |
|---|---|
| [+] (1.7) | Everyone **knows that** the CPI is the most accurate.<br>→ The CPI is the most accurate. |
| [+] (1.7) | Everyone **does not know that** the CPI is the most accurate.<br>→ The CPI is the most accurate. |
| [+] (0.7) | I **know that** I was born to succeed.<br>→ I was born to succeed. |
| [○] (0.3) | I **do not know that** I was born to succeed.<br>↛ I was born to succeed. |

Table 4: Examples of how the factive verb *"know that"* can exhibit different signatures, depending on context.

**Takeaways.** Overall, we interpret the above analysis as evidence that veridicality judgments rely heavily on contextual as opposed to purely lexical semantic factors. While this is not a novel conclusion (Simons et al., 2010; de Marneffe et al., 2012), it is still frequently the case that system development concerned with improving veridicality judgements nearly always proceeds by incorporating explicit lexical semantic knowledge into the pipeline or architecture (Richardson and Kuhn, 2012; Lotan et al., 2013; Stanovsky et al., 2017; MacCartney and Manning, 2009; Angeli and Manning, 2014; Saurí and Pustejovsky, 2012; Cases et al., 2019; Rudinger et al., 2018). Our analysis suggests such approaches are likely to yield only incremental gains. While admittedly more difficult to encode, focusing on context-specific factors first, e.g. predicate classes and pragmatics (de Marneffe et al., 2012) or question under discussion (Simons et al., 2010), would likely be more productive and may ultimately override the need for verb signatures altogether.

## 6 Analysis of BERT Predictions

We now turn to our primary question: do current NLI models capture the veridicality of verbs? In particular, we are interested in the behavior of a distributional model that is not specifically endowed with lexical semantic information related to veridicality. We ask two questions. First: does such a model learn to make inferences consistent with those made by humans? Second: if the model does mirror human inferences, are the predictions based solely on the presence of specific lexical items, or are they sensitive to structural factors (namely, syntactic position and complement type)? Again, we prioritize modeling speaker meaning. Thus, we believe the model should ideally reflect the same biases and variation observed in the human judgments, not necessarily the inferences expected based on the lexical semantic signatures of the verbs.

### 6.1 Setup

We use the state-of-the-art BERT-based NLI model. Specifically, we use the original TensorFlow implementation[11] of the NLI model built on top of the pertained BERT language model (Devlin et al., 2018). We use the model off-the-shelf, with default training setup and hyperparameters. To fine-tune the model for the NLI task, we use the standard train/dev splits from the MNLI corpus, but, to avoid confounds, we remove the 1,500 $p/h$ pairs from which our new test set is derived (as described in §4).[12] The model is trained to make a softmax classification over three classes: {ENTAILMENT, CONTRADICTION, NEUTRAL}. When necessary to compare these discrete predictions to our continuous human judgments, we map the prediction to a continuous value using $P(\text{ENTAILMENT}) - P(\text{CONTRADICTION})$. This score ranges from $-1$ to 1 and is comparable to our human scores. Conversely, when necessary to compare the continuous human score to the discrete predictions, we discretize scores into evenly-sized bins[13]. Overall, similar performance trends hold whether we compare in discrete space or continuous space. In the analyses below, we use whichever is most interpretable, as reported.

### 6.2 Overall Prediction Performance

We first measure raw prediction performance: do the inferences made by the BERT mirror the inference that our human raters made? Figure 1b shows scatter plots of the models predictions (mapped into continuous space) side-by-side with the hu-

worthy effects. It is likely that more careful featurization of highly relevant concepts–e.g. at-issueness (Tonhauser et al., 2018)–could yield more conclusive insights about which aspects of context lead to within-signature, or within-verb, variation. We leave such analysis for future work, and conclude simply that endowing NLI models with knowledge of projectivity signature is not alone sufficient for producing human-like inferences on such sentences.

[11] https://github.com/google-research/bert

[12] See Supplementary Material for a discussion of the prevalence of veridicality phenomena in MNLI train.

[13] CON: $s < -\frac{2}{3}$; NEU: $-\frac{2}{3} \leq s < \frac{2}{3}$; ENT: $\frac{2}{3} \leq s$

| | Count | | Positive | | | Negative | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Sent. | Verb | Exp. | Acc. | $\rho$ | Exp. | Acc. | $\rho$ | Example Verbs |
| Fact | 212 | 15 | + | 0.62 | 0.17 | + | 0.29 | 0.40 | realize that, know that |
| Impl | 100 | 9 | + | 0.57 | 0.51 | − | 0.73 | 0.51 | manage to, begin to |
| Impl | 25 | 2 | − | 0.80 | 0.61 | + | 0.52 | 0.39 | forget to, fail to |
| Impl | 63 | 6 | ∘ | 0.27 | 0.21 | + | 0.43 | 0.43 | suspect that, explain that |
| Impl | 28 | 1 | ∘ | 0.11 | 0.25 | − | 0.71 | 0.45 | attempt to |
| Impl | 55 | 3 | − | 0.93 | 0.70 | ∘ | 0.02 | -0.10 | refuse to, decline to |
| Impl | 80 | 8 | + | 0.38 | 0.21 | ∘ | 0.54 | 0.21 | show that, confirm that |
| NA | 935 | 93 | ∘ | 0.21 | 0.35 | ∘ | 0.44 | 0.47 | try to, hope to |
| Overall | 1,498 | 137 | | 0.34 | 0.63 | | 0.44 | 0.57 | |

Table 5: Accuracy and Spearman correlation of BERT MNLI model predictions against human judgements. The $+/-/\circ$ symbols denote the expected labels based on the lexical semantic category of the verb, and are not necessarily the labels given by our human annotators (compare against Figure 1).

man scores just discussed. Table 5 shows performance evaluated against human judgements in terms of both (discrete) classification accuracy and (continuous) Spearman correlation. Broadly speaking, the model's predictions appear to follow the same trends as the humans' ratings (Figure 1). That is, averaged across contexts, the model's treatment of verbs is the same as the humans' treatment: largely in line with the signatures, but with a bias against assuming neutral (non-veridical) behavior. However, whereas the humans' judgments span all levels of certainty (taking a range of values from $-2$ to 2), the model tends to make predictions with high confidence. This is especially the case in positive environment, where the model nearly always predicts with $99+\%$ confidence. In negative environments, the model expresses a grater range of uncertainty values, and is much more closely in line with what we observe in human judgements. In terms of quantitative measures of accuracy (Table 5), the most notable trend is that the model performance is highest for cases in which the *negation* of the complement is expected to project ($-$ signatures). This is true regardless of whether that behavior occurs in a positive or negative environment. We note that, for such cases, human judgements closely align with the lexical semantic predictions. The model performs worst in positive environments when the verb is expected to be non-veridical ($\circ$ signatures). This appears to result from the model's tendency to over-exaggerate the veridicality bias: i.e. whereas humans show a general tendency to assume the complement projects in these cases, the model predicts ENTAILMENT with near certainty (see Figure 1).

### 6.3 Counterfactual Analysis

Next, we ask: are the above-observed trends in BERT's predictions driven predominantly by lexical priors–i.e. the presence of a specific verb–or are they sensitive to other lexicosyntactic factors that should ideally affect the inference?

**Experimental Design.** For each verb construction *vt* (e.g. *"try to"* or *"realize that"*) in our dataset, we perform several manipulations in which we insert *v* or *t* into sentences where they did not originally appear, and observe the effect this has on the distribution of the model's predictions. Our specific manipulations and the expected effects are described below. Table 6 shows examples. For convenience, we use $D$ to refer to the set of all the $\langle S, C\rangle$ pairs in our dataset, $D_{vt}$ to refer to all the pairs in which *vt* appears as the main verb clause in $S$, and $D_{to}$ ($D_{that}$) to refer to all the pairs in which $C$ is a *to*-complement (*that*-complement). When clear from context, we abuse notation and use e.g. $D$ to refer both to the dataset itself and to the distribution of the model's predictions when run over the dataset.

**Replace Main Verb:** For each pair $\langle S, C\rangle \in D$, we replace the main verb in $S$ with the target verb *v*, generating a new premise $S^*$. We expect that, if the model is sensitive specifically to presence of *v* and its effect on inferences, then the distribution of model predictions over all $\langle S^*, C\rangle$ pairs should be more similar to the target distribution of predictions over all of $D_{vt}$ than to the baseline distribu-

| | Main Verb (Match) | | Main Verb (Mismatch) |
|---|---|---|---|
| S | He **attempted to** overcome the sensation. | S | I **decided that** the department had acted illegally. |
| S* | He **tried to** overcome the sensation. | S* | I **tried that** the department had acted illegally. |
| C | He overcame the sensation. | C | The department had acted illegally. |
| | **Complement Verb** | | **Complement Type** |
| S | He attempted to **overcome** the sensation. | S | They **tried to** get his attention. |
| S* | He attempted to **try** the sensation. | S* | They **tried that** get his attention. |
| C* | He **tried** the sensation. | C | They got his attention. |

Table 6: Examples of counterfactual manipulations with the target verb construction *"try to"*.

tion of predictions over all of $D$. We differentiate between settings with "matched" complement types, where we generate $S^*$ from pairs in $D_t$, from those with "mismatched" complement types, where we generate from pairs in $D_{\neg t}$. E.g., for a target $vt$ = *"try to"*, we consider substitutions into premises from $D_{to}$ as "matched" and substitutions into $D_{that}$ as "mismatched". Preserving this distinction allows us to both avoid confounds due to ungrammatical substitutions, and to investigate whether the model is sensitive to verbs which behave differently when they take different complements. For example, *"forget"* is $-/+$ when it takes *to* but $+/+$ when it takes *that*.

**Replace Complement Verb:** For each $\langle S, C\rangle \in D$, we replace the main verb in $C$ with the target verb *v*, generating a new hypothesis $C^*$. We expect that, if the model is sensitive not just to the presence of *v*, but also its syntactic role, then the distribution of predictions over all $\langle S, C^*\rangle$ pairs should resemble the baseline distribution over $D$ more than the target distribution over $D_{vt}$.

**Replace Complement Type:** For each pair $\langle S, C\rangle \in D_{vt}$, we replace the $t$ in $S$ with the alternative complement type (*"to"* $\rightarrow$ *"that"*; *'that"* $\rightarrow$ *"to"*), generating a new premise $S^*$. This generates ungrammatical sentences, and serves as a control experiment, to check whether the model is considering the entirety of the context in which the verb construction appears, or merely the $vt$ bigram. We expect that, if the model is considering the whole context, the distribution of predictions over all $\langle S^*, C\rangle$ should resemble the target distribution $D_{vt}$ more than the baseline distribution $D_t$.

**Results.** Table 7 shows, for verbs within each signature, the KL divergence between the post-manipulation prediction distribution ($D^*$) and 1) the baseline distribution ($D$) and 2) the target distribution ($D_{vt}$). Results are shown for both the main and complement verb manipulations.

A few trends are worth highlighting. First, we do see evidence that the model's prediction depends at least in part on the individual verb type. This is supported by the fact that, across verb signatures, manipulation of the main verb leads to distributions which are more similar to the target verb distribution $D_{vt}$ than to the baseline distribution $D$. This trend is strongest for verbs which involve $-$ signatures. Second, we see encouraging, though not overwhelming, evidence that the model's prediction are sensitive to the syntactic position of the verb. This is supported by the fact that, in general, the similarity between $D^*$ and $D_{vt}$ is much lower (higher KL) when the manipulation occurs in the complement clause compared than when it occurs in the main clause. Note that, ideally, this manipulation should not effect the prediction distribution at all. Nonetheless, the trend is clear and points in the right direction.

Table 8 shows the KL divergence between $D^*$ and the target verb distribution $D_{vt}$ in the matched and mismatched cases.[14] We see that BERT behaves as we hope–namely, it makes different predictions for *v to* constructions and *v that* constructions, even when the *v* is the same. Manipulating the main verb only substantially affects predictions when the manipulation occurs in a context with the right complement type (matched); when the manipulation results in a ungrammatical sentence (mismatched), the prediction remains close to baseline. An example of such verb-construction differentiation is shown in Figure 2 for the verb *know that*, but this is a trend seen across verbs. Moreover, we see that this effect is not just driven by sensitivity to the specific *vt* bigram. That is, simply swapping *"to"* with *"that"* (or vice-versa) in a naturally-occurring context leads to a small

[14] See Supplementary for breakdown by verb type.

| | | Main Verb | | Compl. Verb | |
|---|---|---|---|---|---|
| | | Pos. | Neg. | Pos. | Neg. |
| $+/+$ | $D^* \| D_{vt}$ | 0.00 | 0.01 | 0.02 | 0.14 |
| | $D^* \| D$ | 0.17 | 0.43 | 0.05 | 0.08 |
| $+/-$ | $D^* \| D_{vt}$ | 0.01 | 0.02 | 0.17 | 0.01 |
| | $D^* \| D$ | 0.17 | 0.67 | 0.04 | 0.39 |
| $-/+$ | $D^* \| D_{vt}$ | 0.04 | 0.06 | 1.36 | 1.39 |
| | $D^* \| D$ | 1.32 | 1.40 | 0.04 | 0.42 |
| $\circ/+$ | $D^* \| D_{vt}$ | 0.01 | 0.02 | 0.01 | 0.06 |
| | $D^* \| D$ | 0.12 | 0.22 | 0.03 | 0.05 |
| $\circ/-$ | $D^* \| D_{vt}$ | 0.05 | 0.07 | 0.13 | 0.01 |
| | $D^* \| D$ | 0.01 | 0.76 | 0.04 | 0.42 |
| $-/\circ$ | $D^* \| D_{vt}$ | 0.28 | 0.11 | 1.90 | 0.53 |
| | $D^* \| D$ | 0.65 | 0.26 | 0.03 | 0.35 |
| $+/\circ$ | $D^* \| D_{vt}$ | 0.00 | 0.06 | 0.05 | 0.31 |
| | $D^* \| D$ | 0.14 | 0.04 | 0.00 | 0.02 |
| $\circ/\circ$ | $D^* \| D_{vt}$ | 0.00 | 0.02 | 0.01 | 0.06 |
| | $D^* \| D$ | 0.02 | 0.00 | 0.00 | 0.02 |

Table 7: Comparison (KL divergence) of post-manipulation prediction distribution to target verb distribution ($D_{vt}$, top row) and baseline distribution ($D$, bottom row). High similarity to $D_{vt}$ suggests the model changed its predictions in response to the manipulation.

shift in the distribution of the model's predictions away from the target $D_{vt}$ distribution, but not to the same degree as replacing a *"that"*-taking verb with a *"to"*-taking verb in a naturally-occurring *"that"* context (or vice-versa). This result provides some evidence that BERT's prediction is influenced by aspects of the context other than just the presence of the $vt$ bigram.

| | Match | Mis. | Swap |
|---|---|---|---|
| Fact $+/+$ | 0.01 | 0.48 | 0.25 |
| Impl $+/-$ | 0.01 | 0.86 | 0.16 |
| Impl $-/+$ | 0.05 | 1.72 | 0.10 |
| Impl $\circ/+$ | 0.01 | 0.16 | 0.22 |
| Impl $\circ/-$ | 0.06 | 0.86 | 0.00 |
| Impl $-/\circ$ | 0.19 | 1.16 | 0.71 |
| Impl $+/\circ$ | 0.03 | 0.23 | 0.28 |
| NA $\circ/\circ$ | 0.01 | 0.03 | 0.46 |

Table 8: KL divergence between $D^*$ and $D_{vt}$ for complement type manipulations (*"to"* vs. *"that"*). Inserting $v$ into a context affects BERT's predictions only when the complement is compatible with $v$.

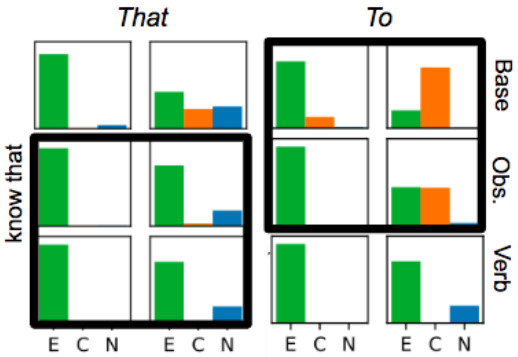


Figure 2: Results for the main verb replacement experiment for the verb *know that*. In examples with the *VB that* construction, the replacement results in predictions that resemble those for *know that*. For *VB to* examples, the resulting counterfactual predictions resemble those of the base examples.

## 7 Conclusion

We investigate how well BERT, a neural NLI model not explicitly endowed with knowledge of lexical semantic verb signatures, is able to learn to make correct inferences about veridicality. We collect a new NLI dataset of human veridicality judgements. We observe that human judgments often differ from what is predicted given the lexical semantic types of verbs, and that BERT is able to replicate many of these judgments, although there is still significant room for improvement. Through counterfactual experiments, we show that individual verbs strongly influence BERT's predictions, and that these cues interact with syntactic information in desirable ways.

## Acknowledgments

Thank you to Sam Bowman and Ben Van Durme for helpful feedback early on in this project. Thank you to the rest of the JSALT 2018 Sentence Representation Team for support throughout. Much of this work was conducted at the 2018 Frederick Jelinek Memorial Summer Workshop on Speech and Language Technologies, and supported by Johns Hopkins University with unrestricted gifts from Amazon, Facebook, Google, Microsoft and Mitsubishi Electric Research Laboratories.

## References

Gabor Angeli and Christopher D Manning. 2014. Naturalli: Natural logic inference for common sense

reasoning. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 534–545.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Ignacio Cases, Clemens Rosenbaum, Matthew Riemer, Atticus Geiger, Tim Klinger, Alex Tamkin, Olivia Li, Sandhini Agarwal, Joshua D Greene, Dan Jurafsky, et al. 2019. Recursive routing networks: Learning to compose modules for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, pages 177–190. Springer.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Ingrid Falk and Fabienne Martin. 2017. Towards an inferential lexicon of event selecting predicates for French. In *IWCS 2017 - 12th International Conference on Computational Semantics - Long papers*.

Michael L Geis and Arnold M Zwicky. 1971. On invited inferences. *Linguistic inquiry*, 2(4):561–566.

Lauri Karttunen. 1971. Implicative verbs. *Language*, pages 340–358.

Lauri Karttunen. 2012. Simple and phrasal implicatives. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 124–131, Montréal, Canada. Association for Computational Linguistics.

Lauri Karttunen, Stanley Peters, Annie Zaenen, and Cleo Condoravdi. 2014. The chameleon-like nature of evaluative adjectives. *Empirical Issues in Syntax and Semantics*, 10:233–250.

Paul Kiparsky and Carol Kiparsky. 1968. *Fact*. Linguistics Club, Indiana University.

Amnon Lotan, Asher Stern, and Ido Dagan. 2013. Truthteller: Annotating predicate truth. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 752–757.

Bill MacCartney. 2009. *Natural language inference*. Ph.D. thesis, Citeseer.

Bill MacCartney and Christopher D Manning. 2009. *Natural language inference*. Citeseer.

Christopher D Manning. 2006. Local textual inference: its hard to circumscribe, but you know it when you see it–and nlp needs it.

Marie-Catherine de Marneffe, Christopher D Manning, and Christopher Potts. 2012. Did it happen? the pragmatic complexity of veridicality assessment. *Computational linguistics*, 38(2):301–333.

R Thomas McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. *arXiv preprint arXiv:1902.01007*.

Rowan Nairn, Cleo Condoravdi, and Lauri Karttunen. 2006. Computing relative polarity for textual inference. In *Proceedings of the fifth international workshop on inference in computational semantics (icos-5)*.

Ellie Pavlick and Chris Callison-Burch. 2016. Tense manages to predict implicative behavior in verbs. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2225–2229.

Kyle Richardson and Jonas Kuhn. 2012. Light textual inference for semantic parsing. In *Proceedings of COLING 2012: Posters*, pages 1007–1018, Mumbai, India. The COLING 2012 Organizing Committee.

Rachel Rudinger, Aaron Steven White, and Benjamin Van Durme. 2018. Neural models of factuality. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 731–744, New Orleans, Louisiana. Association for Computational Linguistics.

Víctor Manuel Sánchez Valencia. 1991. *Studies on natural logic and categorial grammar*. VM Sanchez Valencia.

Roser Saurí and James Pustejovsky. 2012. Are you sure that this happened? assessing the factuality degree of events in text. *Computational Linguistics*, 38(2):261–299.

Mandy Simons, David Beaver, Craige Roberts, and Judith Tonhauser. 2017. The best question: Explaining the projection behavior of factives. *Discourse Processes*, 54(3):187–206.

Mandy Simons, Judith Tonhauser, David Beaver, and Craige Roberts. 2010. What projects and why. In *Semantics and linguistic theory*, volume 20, pages 309–327.

Gabriel Stanovsky, Judith Eckle-Kohler, Yevgeniy Puzikov, Ido Dagan, and Iryna Gurevych. 2017. Integrating deep linguistic features in factuality prediction over unified datasets. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 352–357.

Judith Tonhauser, David I Beaver, and Judith Degen. 2018. How projective is projective content? gradience in projectivity and at-issueness.

Matthijs Westera and Gemma Boleda. 2019. Dont blame distributional semantics it cant do entailment. In *Proceedings of the 13th International Conference on Computational Semantics*.

Aaron Steven White and Kyle Rawlins. 2018. The role of veridicality and factivity in clause selection. In *Proceedings of the 48th Annual Meeting of the North East Linguistic Society, to appear. Amherst, MA: GLSA Publications*.

Aaron Steven White, Rachel Rudinger, Kyle Rawlins, and Benjamin Van Durme. 2018. Lexicosyntactic inference in neural models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4717–4724, Brussels, Belgium. Association for Computational Linguistics.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Annie Zaenen, Lauri Karttunen, and Richard Crouch. 2005. Local textual inference: can it be defined or circumscribed? In *Proceedings of the ACL workshop on empirical modeling of semantic equivalence and entailment*, pages 31–36. Association for Computational Linguistics.

Sheng Zhang, Rachel Rudinger, Kevin Duh, and Benjamin Van Durme. 2017. Ordinal common-sense inference. *Transactions of the Association for Computational Linguistics*, 5:379–395.

# Modeling Color Terminology Across Thousands of Languages

**Arya D. McCarthy**, **Winston Wu**, **Aaron Mueller**,
**Bill Watson**, and **David Yarowsky**
Department of Computer Science
Johns Hopkins University, Baltimore, MD USA
{arya,wswu,amueller,billwatson,yarowsky}@jhu.edu

## Abstract

There is an extensive history of scholarship into what constitutes a "basic" color term, as well as a broadly attested acquisition sequence of basic color terms across many languages, as articulated in the seminal work of Berlin and Kay (1969). This paper employs a set of diverse measures on massively cross-linguistic data to operationalize and critique the Berlin and Kay color term hypotheses. Collectively, the 14 empirically-grounded computational linguistic metrics we design—as well as their aggregation—correlate strongly with both the Berlin and Kay basic/secondary color term partition ($\gamma = 0.96$) and their hypothesized universal acquisition sequence. The measures and result provide further empirical evidence from computational linguistics in support of their claims, as well as additional nuance: they suggest treating the partition as a spectrum instead of a dichotomy.

## 1 Introduction

How many colors are in the rainbow? An infinite number, but each language divides up perceptual space into a *finite* number of categories by giving names to colors. The seminal work on color categories, by Berlin and Kay (1969, hereafter B&K), characterizes a *universal evolutionary sequence* for languages' core colors (their **basic color terms**) and their corresponding categories, at each stage refining the partition of color space.

A handful of criteria define basic color terms, including abstractness, monomorphemicity, and not being subsumed by a broader basic term. (See §2 for the complete list.) These criteria are accused of biasing analyses of color systems—especially in non-Western societies (Wierzbicka, 2006). To mitigate this bias, a pan-lingual approach to analyzing color systems may reveal general ("universal") trends more reliably than smaller datasets. While data are hard to find in the long tail of languages, we still aim to consider more than ever before—2491 languages and dialects.[1] We leverage natural language processing tools to operationalize long-standing literature on language universals.

| Language | Color Word | Literal Gloss |
|---|---|---|
| Welsh | brown | brown |
| Italian | marrone | chestnut |
| Persian | قهوه‌ای | coffee + of |
| Cantonese | 啡色 | coffee + color |

Table 1: Examples of terms representing *brown*, arising from four processes: borrowing (Welsh; from English), null affixing (Italian), derivational affixing (Persian), and compounding (Cantonese).

We provide a three-pronged investigation of the classic criteria for basic color terms, examining the degree to which color words are abstract (§5), monomorphemic/monolexemic (§6), and salient (§7). Our operationalization of these (B&K) criteria shows that individual features do not reflect the basic/non-basic divide. Nor is this divide binary, as B&K suggest: We show that abstractness, monomorphemicity, and even salience do not cleanly divide colors.

Nonetheless, by treating basicness as a spectrum and aggregating these features (like human-judged concreteness, frequency of compounding, and word length) into basicness scores (§8), we can largely distinguish between basic and non-basic colors (validating our measures), and **our scores recreate the historical sequence of color acquisition in language**. The sequence is in no way directly encoded in the criteria for basic color terms; as such, recreating it is a separate and novel empirical discovery.

[1] To this end, we present a large cross-lingual, type-level database of translations of basic and secondary color terms across 2491 languages (§3).



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2241–2250,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

## 2 Color Terminology

Not all languages have the same number of color words; for instance, a single Korean color word (*pureu-n*) applies to both grass and sky—an unusual concept for native English speakers. Similarly, Russian distinguishes between two families of what English speakers call "blue": the lighter *goluboy* and the darker *siniy*. Reaction time experiments show the cognitive importance of these categories (Gilbert et al., 2006; Winawer et al., 2007), and the existence of a named category both aids (Brown and Lenneberg, 1954) and guides (Bae et al., 2015; Cibelli et al., 2016) color judgment and memory.

Color terms may be **concrete** (i.e., derived from a real-world referent like "blood" or "sky") or **abstract**. Diachronic processes can weaken the link between a concrete term and its referent, until a new cohort of speakers believes the term to be abstract. Indeed, this process explains the development of English color words (Casson, 1994). In addition to metonymy with named things, the words may be borrowed, compounded, or inherited from an ancestor language.

While industrialized societies' languages possess a wealth of color words (Hardin, 2014), only a handful are considered *basic* color terms; the remainder are secondary. A **basic color term (BCT)** must satisfy four obligatory criteria (B&K):

1. It must be monolexemic (and monomorphemic). "Light blue" and "blue-green" each contain two lexemes and do not qualify.
2. It may not possess any color hypernyms (superordinate color terms). (E.g., "lavender" has the hypernym "purple".)
3. It may not be limited in application to a narrow class of objects. "Blond(e)" may only be applied to a handful of referents like hair, wood, and beer, for example.
4. It must be psychologically salient. This implies that the color term has a stable range of reference across speakers and has an entry in the lexemic inventory of most (if not all) native speakers' respective idiolects.

Additional criteria are introduced in cases of doubt (Kay and McDaniel, 1978), though these are subjectively applied (Crawford, 1982). Among these: (5) a BCT is not the name of an object that characteristically has a particular color; in other words, the color must be **abstract**, and not grounded in some concrete object (which rules out colors like gold and salmon). Additionally, (6) recent foreign loan words "are suspect", and (7) if the lexemic status of the word is difficult to judge, then multimorphemic words are also "suspect".

$$\begin{bmatrix}\text{white}\\ \text{black}\end{bmatrix} < [\text{red}] < \begin{bmatrix}\text{green}\\ \text{yellow}\end{bmatrix} < [\text{blue}] < [\text{brown}] < \begin{bmatrix}\text{purple}\\ \text{pink}\\ \text{orange}\\ \text{grey}\end{bmatrix}$$

Figure 1: The diachronic sequence of color acquisition (Berlin and Kay, 1969).

In addition to this definition, (B&K) surveyed speakers of 20 languages in the San Francisco Bay Area, plus a sweeping examination of the literature, to find a **sequence** to the emergence of color words in language. Cultures with two color words universally used them to distinguish light and warm colors from dark and cool ones; the third color was universally red, and the sequence continued until matching the set of eleven colors represented by English basic color terms. We present their partial ordering in Figure 1, though later authors have proposed alterations (Heider, 1972; Kay, 1975).

We are not the first to assess the notion of a basic color term. Crawford (1982) gives a point-by-point rebuttal on pragmatic grounds—the criteria are hard for a field worker to assess, and many introduce subjectivity that will bias data collection. Lucy (1997) argues that the definition provides more of a post-hoc screening tool for when the "denotational net" of elicitation has captured too many terms, as opposed to a morphosyntactically informed approach (e.g., Conklin, 1955). Finally, Wierzbicka (2006) argues that other societies may not share the Western conception of hue-based color terms, making the application of the concept inappropriate. In addition to these postulatory objections, a vast literature of similarity judgments, reaction times, and other human measures debates the question from a cognitive perspective (Heider, 1972; Jameson, 2005; Roberson et al., 2005, 2008; Goldstein et al., 2009; Loreto et al., 2012; Persaud and Hemmer, 2014, *inter alia*).

By contrast, we examine the conditions empirically, broadly and automatically on a massively multilingual scale (versus manually and theoretically). Our evidence for assessing B&K's criteria of abstractness, monomorphemicity, and salience comes from a multilingual dragnet of color terms.

## 3 Data

We investigate the three aspects of our theory assessment—abstractness, monomorphemicity, and salience—through multilingual dictionaries. We additionally leverage English corpora to explore abstractness and salience. We use these to construct a dataset of color senses and translations, with scores along numerous axes. As a final resource to investigate salience, we use a global elicitation of color terms from pre-industrialized societies.

In English, the basic color terms are red, orange, yellow, green, blue, purple, brown, pink, black, white, and grey. These align to the eleven basic color categories identified by Berlin and Kay (1969). In addition to these eleven, we consider a list of 92 second-tier color terms identified by Casson (1994). These were elicited from 30 speakers over several days to ensure salience, then filtered by a dictionary to keep only conventional (rather than novel) color descriptors. We omit 12 of these which do not appear in the datasets we employ.

Translation may act as a useful resource for disambiguating word senses (Diab and Resnik, 2002). By translating English BCTs into other languages, we can find their basic color terms. Then, back-translating to English, we obtain a list of the potential senses of a given color term. We draw translations of color terms from two large type-level dictionary resources, PanLex (Baldwin et al., 2010; Kamholz et al., 2014) and Wiktionary, which together provide color word translations for 2491 languages or dialects.

For each of the 11 basic color concepts and 80 secondary terms $e$ in English, we translate it into every available foreign language $\ell$ by dictionary lookup to get a set of non-English color words $F_\ell$. We then back-translate each term into English (again by lookup) to get a set $E_\ell^{(e)} = \bigcup_{f \in F_\ell} \{e' \mid \text{trans}_{\ell \to \text{En}}(f) = e'\}$ of possible round-trip translations through the language $\ell$. The final dataset contains tuples of the form (English color word, foreign language, foreign word, English back-translation).

## 4 Summary of Experiments

We perform a series of experiments to explore the abstractness, morphology, and salience of color terms across thousands of languages.[2] In addition to qualitative observations, our experiments evaluate these qualities through several metrics, illuminating flaws in the definition of "basic color term". When averaging the 14 features together, the implied total ordering is suggestive of the original B&K sequence.

**Goodman and Kruskal's gamma** We measure correlation between basicness and our features with Goodman and Kruskal's gamma (Goodman and Kruskal, 1954, 1959, 1963, 1972), which is well suited for comparing binary variables to ordinal ones. It is a pair-counting measure which ignores tied values. We compute it by maximum likelihood estimation, giving an expression:

$$\gamma = \frac{N_s - N_d}{N_s + N_d}, \tag{1}$$

where $N_s$ is the number of color pairs for which basicness and a feature agree in their ranking; $N_d$ is the number of pairs ranked in opposite orders. Arbitrarily, we represent basicness as 1 and non-basicness as 0. We will be concerned only with the magnitude, not the direction of correlation. A score of $\pm 1$ thus indicates perfect correlation with basicness.

**A remark on noise** Each of our measures is imperfect; it possesses a bias. Combining the results from many weak indicators gives robustness to the noise in any given measure. Foreshadowing future discussion, we see this supported empirically by the effect of aggregation on Goodman and Kruskal's gamma.

## 5 Abstractness

The basic color terms in English did not start as abstract concepts. For instance, "orange" and "pink" were originally derived from the concrete color of a fruit (*Citrus sinensis*) and flower (*Dianthus plumarius*) respectively, and earlier the English color term "black" had its origin in a word for *soot*, but the abstract color senses of these words rose in relative popularity; many supplanted the original definitions as the more common **word sense** (Casson, 1994). It could be that many "basic" color terms emerged metonymically from concrete, real-world referents, as in English.

### 5.1 Concreteness judgments

As a first pass, we directly look up the concreteness for each color word on our list, in a dataset of

[2] We give our data and implementations at https://github.com/aryamccarthy/basic-color-terms.

| Category | Back-translation | Freq. | Conc. | OED POS |
|---|---|---|---|---|
| Black | black | 1065 | 3.76 | Adj |
| | dark | 467 | 4.29 | Adj |
| | dirty | 162 | 4.23 | Adj |
| White | white | 4423 | 4.52 | Adj |
| | bright | 163 | 3.92 | Adj |
| | clear | 163 | 3.55 | Adj |
| Red | red | 3053 | 4.24 | Adj |
| | reddish | 320 | 3.42 | Adj → Adj |
| | crimson | 286 | 4.00 | Adj |
| Green | green | 3786 | 4.07 | Adj |
| | unripe | 468 | 3.31 | Adj → Adj |
| | blue | 242 | 3.76 | Adj |
| Yellow | yellow | 1306 | 4.30 | Adj |
| | saffrony | 133 | - | N → Adj |
| | luteous | 94 | - | F.W. → Adj |
| Blue | blue | 1324 | 3.76 | Adj |
| | gloomy | 361 | 2.52 | Adj |
| | grim | 293 | 1.82 | Adj |
| Brown | brown | 671 | 4.48 | Adj |
| | pink | 314 | 3.93 | Adj |
| | brownish | 106 | 3.24 | Adj → Adj |
| Purple | purple | 932 | 4.04 | N |
| | violet | 324 | 4.48 | N |
| | purpure | 148 | - | N |
| Pink | pink | 121 | 3.93 | Adj |
| | rose | 13 | 4.90 | N |
| | carnation | 10 | 3.93 | N |
| Orange | orange | 1257 | 4.66 | N |
| | orange tree | 89 | - | N |
| | mandarin | 87 | 3.67 | N |
| Gray | gray | 775 | 3.46 | Adj |
| | grey | 570 | 4.11 | Adj |
| | greyish | 155 | 3.99 | Adj |

Table 2: Top back-translations and their concreteness for each of the 11 basic color categories. Extended in the supplementary material. Part of speech is first-listed from the Oxford English Dictionary (Simpson et al., 1989).

40,000 English lemmas' concreteness (Brysbaert et al., 2014) as rated on a Likert scale by Amazon Mechanical Turk workers in the United States. As expected, we find that concreteness negatively correlates with basicness ($\gamma = -0.58$). Many non-basic colors are less concrete than basic terms; for instance, "beige" is the least concrete color (3.41), and most words are less concrete than "orange".

### 5.2 A hologeistic perspective

A word may have multiple senses, which we hope to capture by taking a pan-lingual, or hologeistic, perspective, getting at the concept itself rather than any surface form. To do this, we find the human-judged concreteness *of each of our back-translations*, then average these for each color term, weighted by the number of languages for which a word is a back translation. This balances between a single frequent sense and multiple infrequent senses.

Performing this averaging over languages and senses magnifies our correlation to $\gamma = -0.62$; clearly, exploiting the diversity of senses is beneficial. Still, there is no clear separation between basic and secondary color terms. Concreteness or abstractness thus provides incomplete evidence of basicness.

### 5.3 Part of speech as a proxy for concreteness

Adjectives are, on average, perceived as more abstract than nouns (Darley et al., 1959). We affirm this finding: in the Brysbaert et al. judgments, nouns are less abstract than adjectives on average (3.53 versus 2.50). Because of this, we are comfortable using color words' part of speech as a coarse hint of their abstractness.

We collect part of speech annotations from two sources: the Google Books Ngram Corpus, containing about 4% of all books ever printed (Michel et al., 2011), and the Penn Treebank (Marcus et al., 1993). The former is machine-annotated for part-of-speech (and thus noisier); the latter is annotated by linguists. For each corpus, we compute the ratio of adjectival relative to nominal part of speech, as well as total frequency.[3]

## 6 Morphology

In this section, we ask whether there are affixes that are highly correlated with color; these can be either general derivational affixes or sequences specific to color terms, as in Table 1 and Table 4. The presence of subword structure in basic concepts' translations would show that they violate the monomorphemicity criterion.

Although the B&K criteria demand monolexemic and monomorphemic words, color terms in many languages are formed by some derivational process from a concrete term. To discover the components in an unsupervised fashion, as is necessary for languages in the long tail of linguistic resource

[3] These features work double duty—they shed insight into both the concreteness and the salience as descriptive concepts of each color word. It can lead to spurious ordering, though, if not tempered by other measures—"gold" and "flesh" are both frequent because of their material senses, so raw frequency is not an indicator of basicness.

availability, we look for constituent morphemes and constituent lexemes by segmentation and compound detection, respectively, on each foreign language of our dataset, then use these to score colors' basicness.

### 6.1 Affix discovery

Segmentation and affix discovery is a challenge for the low-resource languages in our study. To give signal to the model, we leverage our other metrics. We compute the percentage of the time that a color word's translation occurs with a suffix that is strongly associated with one of the top 10-highest-ranking colors on the basicness scale, according to the aggregation we mention in §4 and detail in §8. In other words, words that associate with a typical color affix in that language tend to be colors.

This is not a test for basicness, per se, but rather being a likely color, so it supplements the part of speech measures. But it does diminish the rank of almost all words with senses/translations that are not primarily colors. In addition, this measure has the highest correlation of any of ours with basicness ($\gamma = 0.92$)—though this is not surprising, as it was computed by bootstrapping from the other results, which already correlate well in aggregate.

As another tack, we identify likely color-related and general derivational affixes by **unsupervised morphological segmentation**. We define a probability distribution over segmentations. Let $S = s_1 s_2 \ldots s_m$ segment the word $W = \text{BOS} w_1 w_2 \ldots w_n \text{EOS}$. We seek to find the optimal $S^*$. To do this, we decode a model

$$S^* = \arg\max_S \Pr(S) = \arg\max_S \prod_{s_i \in S} \Pr(s_i)$$

using the Viterbi algorithm, where the individual segment probabilities are maximum a posteriori estimates under a Dirichlet prior ($\alpha = 0.01$). The model, inspired by Ge et al. (1999), is similar to other unigram segmentation models (Creutz and Lagus, 2005; Kudo, 2018). We then search for these affixes across the terms recorded in that language, to determine whether the affix is broadly derivational or specific to color terms. Select results are given in Table 3.

Consider the *-tic* suffix in Nahuatl (`nci`). This morpheme semantically denotes "color"; one combines real-world referents with this suffix to obtain colors, e.g., *chichiltic* (chili-color). This appears even with black and white—the first basic color terms in B&K's ordering—implying that Nahuatl lacks basic color terms under the B&K definition.

| Language | Affixes |
|---|---|
| `nci` | -tic† (40%), tla- (27%), xox- (27%) |
| `aqc` | -TYT* (36%) |
| `dbu` | -ŋgó (29%) |
| `mpj` | -lpa* (26%), -ku (15%) |
| `ciw` | -aan-† (23%), -zo* (23%) |

Table 3: Common example affixes. ∗ indicates a general derivational affix. † indicates a color-specific morpheme. Bare affixes are neither. Included for each morpheme is the percentage of color words in our dataset in the given language which contain the morpheme. ISO language codes are given in Appendix A.

| Method | Example |
|---|---|
| concrete + color | `cmn`: 橙 (orange) + 色 (color) |
| color + der. affix | `spa`: anaranja (orange) + do |
| Adj. + color | `deu`: dunkel (dark) + rot (red) |

Table 4: Discovered concatenation strategies pooled across languages, with representative examples from Mandarin, Spanish, and German.

We also see derivational morphemes, which are applied to words from a given part-of-speech class to convert them to another class—e.g., "TYT" in Archi (`aqc`) in Table 3, which is a fused morpheme denoting adjectivalization and marking Archi's fourth gender. This morpheme appears in the Archi's terms for black and white. As with Nahuatl, this implies that Archi lacks basic color terms—according to a strict interpretation of the criteria.

### 6.2 Compound detection

To particularly identify color names which are formed by compounding of words, we extend a model for compound discovery to identify color terms which were produced compositionally. This lets us ask two questions about the BCT definition: (1) Across languages, are there "basic" color terms that are not monolexemic, and (2) Are "basic" terms less likely to be compounds? The answer to both, we find, is "yes".

Wu and Yarowsky (2018) propose a multilingual compound analysis and generation method that only requires a readily available multilingual dictionary. They first extract potential compounds by splitting any word into three substrings corresponding to a left component, glue, and right com-

| Process | Basic | Secondary |
|---|---|---|
| Inheritance | 1161 | 2356 |
| Derivation | 82 | 183 |
| Cognate | 303 | 483 |
| Borrowing | 18 | 84 |
| None of these | 42566 | 65969 |

Table 5: Sources of foreign color words. The colors are not mutually exclusive.

ponent. These compounds are used to construct compound "recipes". For example, they discovered that the concept of 'hospital' is frequently represented across a variety of languages as a compound of 'sick'/'disease' and 'house'/'home' in their respective language. They use these recipes to score their initial list of potential compounds, filter out low-scoring, unlikely compounds, and performing a second pass of recipe construction, resulting in a higher-quality compound dataset. Compound analysis is performed in a similar manner as recipe construction. Compound generation takes into account language-specific knowledge of which components and glues are common in that language.

Wu and Yarowsky (2018) consider only single- or zero-character glue between the two components. By contrast, **we allow glues of arbitrary length**, exhaustively searching through all segmentations into three parts. This increases the algorithmic complexity by $\Theta(K)$, where $K$ is the length of the word. Searching through our PanLex and Wiktionary foreign translations of only our basic color terms, we find several examples of compounded words. Some examples are given in Table 4. The frequency of a color being expressed by compounding, though, turns out to be a weak indicator of basicness ($\gamma = 0.35$).

### 6.3 An aside on borrowings

One of B&K's lower-tier criteria for color terms was that borrowings were "suspect". Here we examine how often color terms are borrowed, as well as other avenues for color construction.

In addition to translations and definitions, Wiktionary provides etymologies for many languages. These relations have been parsed and extracted as EtymDB (Sagot, 2017).

We report the aggregation of EtymDB's parsed etymologies in Table 5; these are broken down by color in the supplementary material. Basic colors are more often created by borrowing, suffixing, and compounding than secondary colors; nevertheless, this should be taken with a grain of salt: The annotations for secondary colors are less complete, so while we use these scores, we do not take the criterion of borrowings as a prong of our criticism.

## 7 Salience

B&K assessed salience by a color's tendency to appear earlier when asking speakers to list their language's colors. More general assessments are outlined by Hays et al. (1972): word length, frequency of use, ethnographic frequency, and correlation of vocabulary size with cultural complexity. While the final one of these is beyond our scope, we present simple experiments to test the other three, based on our translation dataset.

### 7.1 Word length

Durbin (1972) supposed, based on Zipf's Law of Abbreviation (Zipf, 1932, 1949), that word length would decrease for more salient, broadly used color terms. We test this across over 2400 languages by computing the mean length of all translations for each English color word, regardless of script.

With the exception of grey, the first six colors of the (B&K) sequence—black white, red, yellow, green, and blue—have lower average lengths than the subsequent five. This supports Durbin's two-phase theory of basic color terms. Still, beyond this handful, there is no clear separation between basic and non-basic colors. There is only a moderate correlation ($\gamma = -0.41$) with basicness.

### 7.2 Frequency: Usage and ethnography

Neither English corpora nor multilingual dictionaries give a complete picture of the world's languages; after all, only 3 to 4 thousand of them have writing systems (Lewis et al., 2015). To augment our analyses, we consider the grounded speaker elicitation performed in the World Color Survey (WCS; Cook et al., 2005). In it, 2568 speakers from 110 pre-industrialized societies gave their color naming judgments about colored chips on a stimulus palette (see Figure 2) which evenly varied hue and brightness. Field workers then transcribed the utterances and applied the B&K criteria to ascertain which colors were basic. With the WCS, we can discover the synchronic homogeneity of a term's use for the labeling task.

**Do all speakers agree that a given term should be used?** We examine the consensus of use

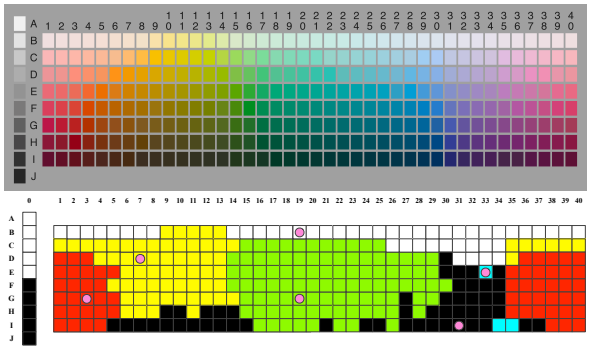

Figure 2: *Top:* The stimulus palette used to collect color namings by Berlin and Kay (1969) and later the World Color Survey. Colors vary vertically in lightness and horizontally in hue. All are fully saturated. *Bottom:* The extensions of the six colors identified by one speaker of the Iduna language, spoken in Papua New Guinea.

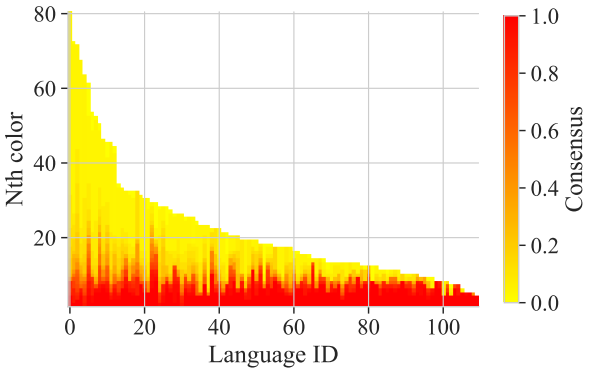


Figure 3: The relative frequencies of each language's color term in the World Color Survey. The height of the colored columns forms a histogram, showing the total number of elicited words. Redder color indicates broader consensus: The word was elicited by a greater fraction of the language's speakers.

among words elicited in the World Color Survey. For each elicited color word in the language, we count the number of speakers who used it. Indeed, the distribution is varied. In Figure 3, we give a histogram of the total number of color terms elicited from each language, as well as the consensus for each individual color. We see a lexical core and periphery for most languages. Most color terms are unique, and languages may have up to 79 terms given among their speakers. It is natural for a speaker to use an unexpected word, still believing the color to fit in a more basic category (as with 'turquoise' and 'blue' in English), but it is surprising that these words would not be sifted out by field linguists' "denotational net".

| Score | Defined | Basic | Sequence |
|---|---|---|---|
| Word concreteness | §5.1 | 0.578 | 0.554 |
| Translation concreteness | §5.2 | 0.616 | 0.583 |
| Ngram frequency | §5.3 | 0.897 | 0.896 |
| Ngram %ADJ | §5.3 | 0.806 | 0.780 |
| Penn TB %ADJ | §5.3 | 0.707 | 0.705 |
| Count of compounding | §6.2 | 0.874 | 0.858 |
| Frequency of compounding | §6.2 | 0.354 | 0.383 |
| Affix presence | §6.1 | 0.924 | 0.892 |
| Borrowing etym. | §6.3 | −0.254 | −0.231 |
| Cognate etym. | §6.3 | 0.797 | 0.804 |
| Derivation etym. | §6.3 | 0.830 | 0.805 |
| - Suffix derivation | §6.3 | 0.877 | 0.869 |
| Inheritance etym. | §6.3 | 0.765 | 0.774 |
| Word length | §7.1 | 0.414 | 0.427 |
| Aggregate | §8 | 0.964 | 0.962 |

Table 6: Goodman and Kruskal's gamma rank correlation between each measure and both basicness and the diachronic color sequence. Features not thought to indicate basicness (as discussed) have been negated. As ordering the sequence is more nuanced than sifting basic from non-basic, one might expect consistently lower correlations. However, the sequence gamma can have larger magnitude if, despite poor sifting, the order of the basic terms is correct.

**Do all speakers have the same number of colors?** We look at the variability of sizes of each speaker's inventory for a language, rather than the consensus on each term. The standard deviation of inventory sizes from the speakers of each language shows notable variation, especially when the mean inventory size exceeds 6.

Heterogeneity in within-language color inventories is unsurprising; as Kay (1975) noted, younger members of numerous language communities possess a broader inventory of basic color terms, following Berlin and Kay's evolutionary sequence of inventories. Nevertheless, basic color terms are defined on a per-language level; they ignore this synchronic variation. This raises questions for future work: Are there categories salient only to young speakers, not the old? What would be a population's (or language's) basic color terms?

## 8 Aggregation of Features

We have operationalized some B&K criteria for basic color terms. Independently, each fails to match up with the known set of basic color terms. Now, we aggregate our independent scores to create a robust measure of basicness from the weak measures.

We do not cherry-pick our measures; some that we include actually harm the ordering. Instead, we

| Color | Rank | B&K | Agg. score |
|---|---|---|---|
| **white** | 1 | 1–2 | 1.00 |
| **black** | 2 | 1–2 | 0.97 |
| **red** | 3 | 3 | 0.92 |
| **green** | 4 | 4–5 | 0.83 |
| **yellow** | 5 | 4–5 | 0.80 |
| **blue** | 6 | 6 | 0.79 |
| **gray** | 7 | 8–11 | 0.73 |
| gold | 8 | | 0.67 |
| **brown** | 9 | 7 | 0.66 |
| **pink** | 10 | 8–11 | 0.64 |
| scarlet | 11 | | 0.64 |
| **purple** | 12 | 8–11 | 0.62 |
| crimson | 13 | | 0.60 |
| beige | 14 | | 0.58 |
| silver | 15 | | 0.55 |
| blond | 16 | | 0.54 |
| tan | 17 | | 0.52 |
| amber | 18 | | 0.52 |
| flesh | 19 | | 0.51 |
| bronze | 20 | | 0.48 |

Table 7: Total ordering of colors according to our aggregate score, with conventionally basic colors bolded. Complete results are given in the supplementary material. This includes the position of the missing basic color, orange.

operationalize each criterion, then see what shakes out. We take an unweighted average of normalized scores for the same reason. This makes the result more evocative; they come purely from our operationalization, rather than targeted tuning.[4]

Using the aggregated measure produces the highest correlation with both basicness and the order of the B&K sequence; see Table 6. We recover the first six colors in the evolutionary sequence from Figure 1: white and black, red, green and yellow, and blue! An extended ordering is given in Table 7; it suggests that the most primary of the non-basic terms are gold, scarlet, crimson, and beige.

Some further inquiry is possible. We see that orange is the 24th ranked color out of 91. This is a stark separation from the ten other basic color terms; beyond this, it has the highest concreteness of the basic terms.

[4] Nonetheless, we characterize the contribution of individual features in Appendix B.

## 9 Discussion

While the notion of a basic color term has been widely used, its validity has been taken as given. With NLP techniques in the broadest multilingual survey by far, we add to the literature investigating the definition of basic color terms. The ability to produce color templates shows that monomorphemicity is an unreasonable criterion. The concreteness of many color words' back-translations violates abstractness. Finally, the heterogeneity of color naming data contends with the salience requirement. None of the traditional criteria for basic color terms hold up robustly. Despite this, when taken in aggregate (and operationalized as we do), they suggest the traditional sequence of color terms and a coarse division between basic and non-basic colors.

As color terms are often decomposable, we can turn the decomposition on its head to *generate* missing color words. We have shown cross-lingual patterns of word formation that future work can exploit, giving plausible entries in a bilingual dictionary (Wu and Yarowsky, 2018): Without the word for "hospital", one can convey the concept by "sick"+"house"; likewise, without the word for "gray", one can use "ash"+DERIVATIONAL AFFIX. Future work will investigate generation and validation of unseen color terms.

Finally, given that the divide between basic and secondary color terms is so blurred, future computational models of these should employ models of graded membership, such as fuzzy set theory.

## 10 Conclusion

This paper has investigated the universal basic color term theories of Berlin and Kay (1969) and others. It provides empirically-grounded computational linguistic metrics with evidence from 2491 languages, harnessing multiple on-line resources of varying quality. We have shown that although the obligatory criteria do not in fact cleanly separate basic from non-basic colors, our features' aggregation correlates strongly with the Berlin and Kay basic/secondary color term partition ($\gamma = 0.96$). The aggregation also largely predicts the Berlin and Kay hypothesized universal acquisition sequence, which is in no way directly entailed by the basicness criteria. Thus, we provide further empirical evidence from computational linguistics in support of the B&K claims, while also providing additional nuance and perspective thereon.

## Acknowledgments

We thank several members of the Johns Hopkins University Center for Language and Speech Processing, as well as Elyse Wilson, for early discussions that shaped the manuscript. Keehoon Trevor Lee prepared part of the data for Table 2 as an undergraduate research assistant. We thank Ryan Cotterell for producing the lower part of Figure 2 and Dorothy Hu for presentation suggestions on Figure 3.

## References

Gi-Yeul Bae, Maria Olkkonen, Sarah R. Allred, and Jonathan I. Flombaum. 2015. Why some colors appear more memorable than others: A model combining categories and particulars in color working memory. *Journal of Experimental Psychology: General*, 144(4):744.

Timothy Baldwin, Jonathan Pool, and Susan Colowick. 2010. PanLex and LEXTRACT: Translating all words of all languages of the world. In *Coling 2010: Demonstrations*, pages 37–40. Coling 2010 Organizing Committee.

Brent Berlin and Paul Kay. 1969. *Basic color terms: Their universality and evolution*. University of California Press.

Roger W. Brown and Eric H. Lenneberg. 1954. A study in language and cognition. *The Journal of Abnormal and Social Psychology*, 49(3):454.

Marc Brysbaert, Amy Beth Warriner, and Victor Kuperman. 2014. Concreteness ratings for 40 thousand generally known English word lemmas. *Behavior Research Methods*, 46(3):904–911.

Ronald W. Casson. 1994. Russett, rose, and raspberry: The development of English secondary color terms. *Journal of Linguistic Anthropology*, 4(1):5–22.

Emily Cibelli, Yang Xu, Joseph L. Austerweil, Thomas L. Griffiths, and Terry Regier. 2016. The Sapir–Whorf hypothesis and probabilistic inference: Evidence from the domain of color. *PloS one*, 11(7):e0158725.

Harold C. Conklin. 1955. Hanunóo color categories. *Southwestern Journal of Anthropology*, 11(4):339–344.

Richard S. Cook, Paul Kay, and Terry Regier. 2005. The World Color Survey database. In *Handbook of categorization in cognitive science*, pages 223–241. Elsevier.

T.D. Crawford. 1982. Defining "basic color term". *Anthropological Linguistics*, pages 338–343.

Mathias Creutz and Krista Lagus. 2005. *Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0*. Helsinki University of Technology Helsinki.

Frederic L Darley, Dorothy Sherman, and Gerald M Siegel. 1959. Scaling of abstraction level of single words. *Journal of speech and hearing research*, 2(2):161–167.

Mona Diab and Philip Resnik. 2002. An unsupervised method for word sense tagging using parallel corpora. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*.

Marshall Durbin. 1972. Review of basic color terms. *Semiotica*, 6(3):257–278.

Xianping Ge, Wanda Pratt, and Padhraic Smyth. 1999. Discovering Chinese words from unsegmented text (poster abstract). In *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 271–272, New York, NY, USA. ACM.

Aubrey L. Gilbert, Terry Regier, Paul Kay, and Richard B. Ivry. 2006. Whorf hypothesis is supported in the right visual field but not the left. *Proceedings of the National Academy of Sciences*, 103(2):489–494.

Julie Goldstein, Jules Davidoff, and Debi Roberson. 2009. Knowing color terms enhances recognition: Further evidence from English and Himba. *Journal of Experimental Child Psychology*, 102(2):219–238.

Leo A. Goodman and William H. Kruskal. 1954. Measures of association for cross classifications. *Journal of the American Statistical Association*, 49(268):732–764.

Leo A. Goodman and William H. Kruskal. 1959. Measures of association for cross classifications. II: Further discussion and references. *Journal of the American Statistical Association*, 54(285):123–163.

Leo A. Goodman and William H. Kruskal. 1963. Measures of association for cross classifications III: Approximate sampling theory. *Journal of the American Statistical Association*, 58(302):310–364.

Leo A. Goodman and William H. Kruskal. 1972. Measures of association for cross classifications, IV: Simplification of asymptotic variances. *Journal of the American Statistical Association*, 67(338):415–421.

Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. 2002. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1):389–422.

C.L. Hardin. 2014. Berlin and Kay theory. *Encyclopedia of Color Science and Technology*, pages 1–4.

David G. Hays, Enid Margolis, Raoul Naroll, and Dale Revere Perkins. 1972. Color term salience. *American Anthropologist*, 74(5):1107–1121.

Eleanor R. Heider. 1972. Universals in color naming and memory. *Journal of Experimental Psychology*, 93(1):10.

Kimberly A. Jameson. 2005. Culture and cognition: What is universal about the representation of color experience? *Journal of Cognition and Culture*, 5(3):293–348.

David Kamholz, Jonathan Pool, and Susan Colowick. 2014. PanLex: Building a resource for panlingual lexical translation. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*. European Language Resources Association (ELRA).

Paul Kay. 1975. Synchronic variability and diachronic change in basic color terms. *Language in Society*, 4(3):257–270.

Paul Kay and Chad K. McDaniel. 1978. The linguistic significance of the meanings of basic color terms. *Language*, pages 610–646.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75. Association for Computational Linguistics.

M Paul Lewis, Gary F Simons, and Charles D Fennig, editors. 2015. *Ethnologue: languages of the world*, eighteenth edition. SIL International, Dallas.

Vittorio Loreto, Animesh Mukherjee, and Francesca Tria. 2012. On the origin of the hierarchy of color names. *Proceedings of the National Academy of Sciences*, 109(18):6819–6824.

John A. Lucy. 1997. The linguistics of "color". In *Color categories in thought and language*, chapter 15, pages 320–346. Cambridge University Press.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K. Gray, , Joseph P. Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig, Jon Orwant, Steven Pinker, Martin A. Nowak, and Erez Lieberman Aiden. 2011. Quantitative analysis of culture using millions of digitized books. *Science*, 331(6014):176–182.

Kimele Persaud and Pernille Hemmer. 2014. The influence of knowledge and expectations for color on episodic memory. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 36.

Debi Roberson, Jules Davidoff, Ian R.L. Davies, and Laura R. Shapiro. 2005. Color categories: Evidence for the cultural relativity hypothesis. *Cognitive Psychology*, 50(4):378–411.

Debi Roberson, Hyensou Pak, and J. Richard Hanley. 2008. Categorical perception of colour in the left and right visual field is verbally mediated: Evidence from Korean. *Cognition*, 107(2):752–762.

Benoît Sagot. 2017. Extracting an Etymological Database from Wiktionary. In *Electronic Lexicography in the 21st century (eLex 2017)*, pages 716–728, Leiden, Netherlands.

John Simpson, Edmund SC Weiner, et al. 1989. Oxford English dictionary online. *Oxford: Clarendon Press. Retrieved March*, 6.

Anna Wierzbicka. 2006. The semantics of color. *Progress in Color Studies. Amsterdam/Philadelphia: John Benjamins Publishing Company*, pages 1–24.

Jonathan Winawer, Nathan Witthoft, Michael C. Frank, Lisa Wu, Alex R. Wade, and Lera Boroditsky. 2007. Russian blues reveal effects of language on color discrimination. *Proceedings of the National Academy of Sciences of the United States of America*, 104(19):7780–7785.

Winston Wu and David Yarowsky. 2018. Massively translingual compound analysis and translation discovery. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

G. K. Zipf. 1932. *Selected studies of the principle of relative frequency in language.* Harvard Univ. Press, Oxford, England.

George Kingsley Zipf. 1949. *Human behavior and the principle of least effort*. Addison-Wesley Press, Oxford, England.

# Negative Focus Detection via Contextual Attention Mechanism

**Longxiang Shen**[1], **Bowei Zou**[12]*, **Yu Hong**[1],
**Qiaoming Zhu**[1], **Guodong Zhou**[1], **Ai Ti Aw**[2]
[1]School of Computer Scienceand Technology, Soochow University, China
[2]Aural & Language Intelligence Department, Institute for Infocomm Research, Singapore
lxshen.scu@gmail.com, zoubowei@suda.edu.cn,
{yhong, qmzhu, gdzhou}@suda.edu.cn, aaiti@i2r.a-star.edu.sg

## Abstract

Negation is a universal but complicated linguistic phenomenon, which has received considerable attention from the NLP community over the last decade, since a negated statement often carries both an explicit negative focus and implicit positive meanings. For the sake of understanding a negated statement, it is critical to precisely detect the negative focus in context. However, how to capture contextual information for negative focus detection is still an open challenge. To well address this, we come up with an attention-based neural network to model contextual information. In particular, we introduce a framework which consists of a Bidirectional Long Short-Term Memory (BiLSTM) neural network and a Conditional Random Fields (CRF) layer to effectively encode the order information and the long-range context dependency in a sentence. Moreover, we design two types of attention mechanisms, word-level contextual attention and topic-level contextual attention, to take advantage of contextual information across sentences from both the word perspective and the topic perspective, respectively. Experimental results on the SEM'12 shared task corpus show that our approach achieves the best performance on negative focus detection, yielding an absolute improvement of 2.11% over the state-of-the-art. This demonstrates the great effectiveness of the two types of contextual attention mechanisms.

## 1 Introduction

As a linguistic phenomenon that reverses the polarity of a statement or its property into opposite, negation is ubiquitous in human languages. Consider following sentence S1. A negative affix *-n't* negates the statement that *mutual fund trades take effect until the market close*. Negation processing has been shown critical for a number of NLP applications, such as biomedical information extraction (Morante, 2010; Mehrabi et al., 2015) and sentiment analysis (Wiegand et al., 2010; Jimenez-Zafra et al., 2017).

(S1) *Mutual fund trades don't* ***take*** *effect* {*until the market close*}.[1]

In general, negation is marked by a word or an affix and interact with other parts in sentence. Blanco and Moldovan (2011) demonstrated that over 65% of negated statements in PropBank (Palmer et al., 2005) convey implicit positive meanings. For instance, underneath sentence S1, the statement that *mutual fund trades take effect* is true, when removing the propositional clause *until the market close* from the original sentence. As the definition of negation focus in SEM'12 (Morante and Blanco, 2012), a negative focus is the most prominently negated part of sentence (Huddleston and Pullum, 2002). In sentence S1, the negative focus is the propositional clause *until the market close*, yielding the interpretation that *mutual fund trades take effect, but not until the market close*. As discussed above, correctly understanding a negated statement requires precise detection of the negative focus. In this paper, we focus on detecting the negative focus in a sentence with a negative keyword and its corresponding dominated verb.

Not as simple as that in speech, negative focus detection poses considerable challenges on text understanding without knowing stress or intonation information. Previous studies (Blanco and Moldovan, 2011; Zou et al., 2014) pointed out that the contextual information plays a critical role in negative focus detection, since the real negative intention often relates to what the author repeatedly

* corresponding auther

[1]#13 of the test set in *SEM 2012 shared task corpus. Throughout this paper, the negated verbs are marked in bold, and the negative foci are in curly braces.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2251–2261, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

state in adjacent sentences. The same negative statement might be interpreted discriminately depending on different contexts. For example, in following three scenarios, consider what is the more likely negative focus of sentence S1 with different contexts:

I. ..., *but exchange fund trades do*.
   negative focus: *mutual fund trades*
II. ..., *unless it reboots*.
   negative focus: *take effect*
III. ..., *for the shareholders stayed put today*.
   negative focus: *until the market close*

In scenario I, since *exchange fund trades* is emphasized in context, the negative focus should be the phrase *mutual fund trades*, yielding the interpretation that *the trades which don't take effect are mutual fund trades, but not exchange fund trades*. Unlike scenario I, the interpretation of negative statement from scenario II is that *Mutual fund trades don't take effect without reboot*, with the context *unless it reboots*, and the phrase *take effect* is the negative focus. Similarly, scenario III points out the reason of *not take effect* is that *the shareholders stayed put today*, which corresponds to the negative focus *until the market close*. In addition, it is worth noting that a negative focus is often not syntactically determined by sentence structure, instead pragmatically judged by authors' intentions conveyed in context.

This is consistent with the attention mechanism in neural networks which has been proven effective to improve the performances for many NLP tasks, such as sentiment analysis (Wang et al., 2016) and relation classification (Zhou et al., 2016). In this paper, we come up with two types of attention mechanisms to enforce the model to take full advantage of the contextual information for negative focus detection.

In particular, we introduce a framework which consists of a bidirectional long short-term memory neural network and a conditional random fields layer (BiLSTM-CRF) to effectively encode the order information and the long-range context dependency in sentence. On this basis, we propose two attention mechanisms to capture contextual information across sentences. One is to additionally append word-level vectors of adjacent sentences into the input vectors. Such attention can automatically focus on the words in context related to the negative focus. The other is to concatenate the topic-level representations into the input vectors to better capture the contextual information. Experimentation on the SEM'12 shared task corpus (Morante and Blanco, 2012) shows that our approach achieves the best performance on negative focus detection task, yielding an absolute improvement of 2.11% over the state-of-the-art.

The rest of this paper is organized as follows. In Section 2, we give a brief review of related work. In Section 3, we introduce the details of the proposed approach. We show our experimental results and discussions in Section 4. Finally, Section 5 concludes with possible directions for future work.

## 2 Related Work

Earlier studies of negation processing in computational linguistics mainly lie in biomedical information extraction (Chapman et al., 2001; Goldin and Chapman, 2003). With the release of the BioScope corpus (Vincze et al., 2008), the negation processing techniques received a substantial boost (Morante et al., 2008; Apostolova et al., 2011; Zou et al., 2013). In addition, negation also was studied as a critical factor in polarity shifting in sentiment and opinion analysis with various lexical and syntactic features (Socher et al., 2013; Cruz et al., 2016).

In general, existing studies of negation processing mainly concentrate on three aspects: 1) cue detection, which finds negative triggers or expressions in text; 2) scope resolution, which determines the grammatical scope in a sentence affected by a negative cue; and 3) focus detection, which identifies the most prominently/explicitly negated part in a negative statement. A negative focus could be considered as the semantic part in scope that is intended to be interpreted as false to make other parts to be true. Among above these subtasks, negative focus detection is most extremely challenging in negation processing.

In the literature, research on negative focus detection was pioneered by Blanco and Moldovan (2011), who proposed a supervised learning model with a set of highly hand-crafted features as a benchmark for this task. On the basis, Blanco and Moldvan (2013) incorporates negation into other existing semantic representations. The dataset annotated by Blanco and Moldovan is used as a standard evaluation corpus for the SEM'12 shared task (Morante and Blanco, 2012). In this shared task, Rosenberg and Bergler (2012) employed three

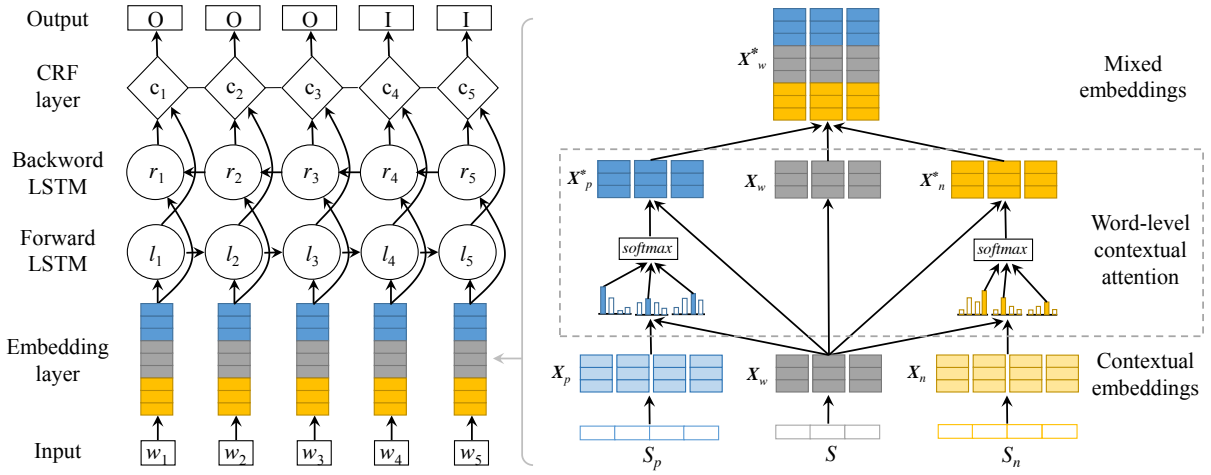


Figure 1: Architecture of BiLSTM-CRF network with word-level contextual attention mechanism for negative focus detection.

kinds of heuristic rules for negative focus detection. Moreover, Zou et al. (2014; 2015) proposed graph-based models to enrich intra-sentence features with inter-sentence features from both lexical and topic perspectives, respectively. Almost all of the above studies have demonstrated that the information contained in context plays a critical role in negative focus detection. In this paper, we explore the effectiveness of different representations and processing manners for modeling context in this task.

In addition to the SEM'12 shared task corpus, there are several annotated datasets for negative focus detection. Anand and Martell (2012) made a point about how to describe the contribution of negative focus to a sentence via the theory of question under discussion, and reannotated a part of the SEM'12 shared task corpus. Matsuyoshi et al. (2014) annotated 1,327 instances of negative foci in Japanese text, and proposed a heuristic rule-based approach to identify them. Banjade and Rus (2016) annotated the DT-Neg corpus on tutorial dialogue genre, which contains 1,088 of negative focus instances. Considering the scale and accessibility of the dataset, we utilize the SEM'12 shared task corpus for experimentation.

## 3 Contextual Attention-based Negative Focus Detection

In this section, we first introduce the BiLSTM-CRF framework. On the basis, we then come up with two kinds of attention mechanisms to model the contextual information, i.e. word-level attention and topic-level attention.

### 3.1 BiLSTM-CRF Framework

In this paper, we recast negative focus detection problem as an I/O tagging task, and apply the IO label scheme, where **I** denotes the token is **I**nside the scope of negative focus in a sentence, and **O** indicates the token is **O**utside of this scope. For example, the sentence S1 are tagged as below.

*mutual*/O *fund*/O *trades*/O *don't*/O ***take***/O *effect*/O *until*/I *the*/I *market*/I *close*/I

The left part of Figure 1 illustrates the BiLSTM-CRF framework for negative focus detection. First, the sequence of embeddings ($\boldsymbol{x}_i$) is given as input to BiLSTM networks, which generates a representation of the left context ($\boldsymbol{l}_i$) and the right context ($\boldsymbol{r}_i$) for each token in a sentence. Then, these representations are concatenated ($\boldsymbol{c}_i$) and linearly projected onto a CRF layer to take into account neighboring tags, yielding the final prediction for every token ($y_i$).

**Embedding Layer** We build an embedding layer to encode words and semantic role labels. Given an input sentence $S = (w_1, w_2, ..., w_n)$, we first transform each word into a real-valued vector $\boldsymbol{x}_w \in \mathbb{R}^{d_w}$. In addition, according to the annotated guideline (Blanco and Moldovan, 2011), semantic roles in sentence often have a close relationship with negative focus and negated verb. To capture such informative features, we then map the semantic role tag of each word to a real-valued vector $\boldsymbol{x}_{sr}$ of dimension $d_{sr}$ using a embedding matrix $\boldsymbol{SR} \in \mathbb{R}^{d_{sr} \times |V_{sr}|}$, where $V_{sr}$ is the set of semantic role label vocabulary. Finally, we represent an input sentence as a vector sequence $\boldsymbol{x} = [\boldsymbol{x}_w; \boldsymbol{x}_{sr}]$

with the embedding dimension $d = (d_w + d_{sr})$.

**BiLSTM Layer** We employ LSTM networks (Hochreiter and Schmidhuber, 1997) which incorporate memory-cells to capture long contextual dependencies in sequence. Formally, each cell in LSTM can be computed as follows:

$$\begin{aligned}
\boldsymbol{X}_t &= [\boldsymbol{h}_{t-1} \ \ \boldsymbol{x}_t]^T \\
\boldsymbol{i}_t &= \sigma(\boldsymbol{W}_i \boldsymbol{X}_t + \boldsymbol{b}_i) \\
\boldsymbol{f}_t &= \sigma(\boldsymbol{W}_f \boldsymbol{X}_t + \boldsymbol{b}_f) \\
\boldsymbol{o}_t &= \sigma(\boldsymbol{W}_o \boldsymbol{X}_t + \boldsymbol{b}_o) \\
\boldsymbol{c}_t &= \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tanh(\boldsymbol{W}_c \boldsymbol{X}_t + \boldsymbol{b}_c) \\
\boldsymbol{h}_t &= \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t)
\end{aligned} \tag{1}$$

where $\sigma$ is the sigmoid function, and $\odot$ is the element-wise multiplication. $\boldsymbol{W}_i$, $\boldsymbol{W}_f$, and $\boldsymbol{W}_o$ are the weighted matrices, and $\boldsymbol{b}_i$, $\boldsymbol{b}_f$, and $\boldsymbol{b}_o$ are the biases, which parametrize the transformations of input, forget, and output gates, respectively. $\boldsymbol{h}_t$ is the vector of hidden state.

Moreover, to benefit from both previous and future information, a forward LSTM and a backward LSTM are employed to generate a representation $\overrightarrow{\boldsymbol{h}_t}$ of the left context and $\overleftarrow{\boldsymbol{h}_t}$ of the right, respectively. The word representation $\boldsymbol{h}_t$ is obtained by concatenating the left and right outputs $\left[\overrightarrow{\boldsymbol{h}_t}; \overleftarrow{\boldsymbol{h}_t}\right]$.

**CRF Layer** To learn the dependencies across output labels of the BiLSTM layer, we model them jointly using a conditional random field (CRF) layer (Lafferty et al., 2001). For an input sentence $\boldsymbol{x}$, we denote $\boldsymbol{C}$ as the matrix of output by BiLSTM. $\boldsymbol{C}$ is of size $n \times k$, where $k$ is the number of distinct tags, and $c_{i,j}$ corresponds to the score of the $j^{th}$ tag of the $i^{th}$ token in a sentence. For a sequence of predictions $\boldsymbol{y}$, we define its score to be

$$s(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=0}^{n} A_{y_i, y_{i+1}} + \sum_{i=1}^{n} c_{i, y_i}, \tag{2}$$

where $A_{i,j}$ denotes the score of a transition from the tag $i$ to the tag $j$. $y_0$ and $y_{n+1}$ are the additional tags of *START* and *END*, respectively. A softmax layer over all possible tag sequences yields a probability for the sequence $\boldsymbol{y}$:

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})} \exp(s(\boldsymbol{x}, \boldsymbol{y})), \tag{3}$$

where $Z(\boldsymbol{x}) = \sum_Y \exp(s(\boldsymbol{x}, Y))$, and $Y$ denotes all possible tag sequences.

During training, we maximize the log-probability of the correct tag sequence:

$$\mathcal{L}_c = \max \log(p(\boldsymbol{y}|\boldsymbol{x})). \tag{4}$$

While during decoding, we predict the output sequence that obtains the maximum score given by

$$\boldsymbol{y}^* = \arg\max_Y s(\boldsymbol{x}, Y). \tag{5}$$

### 3.2 Word-level Contextual Attention

As discussed in Section 1, contextual information is vital for negative focus detection. However, the BiLSTM-CRF model might fail to identify the important information related to a negative focus in contexts. To address this issue, we propose two kinds of attention mechanisms, i.e. the word-level contextual attention and the topic-level contextual attention.

The right part of Figure 1 illustrates the architecture of the word-level contextual attention mechanism. Given previous sentence $S_p$ and next sentence $S_n$, we respectively encode each word $\boldsymbol{w}_{p_i}$ and $\boldsymbol{w}_{n_i}$ into real-valued vectors $\boldsymbol{x}_{p_i}$ and $\boldsymbol{x}_{n_i} \in \mathbb{R}^{d_w}$ by the word embeddings described in Subsection 3.1. Suppose that $N_p$, $N$, and $N_n$ denote the lengths of previous sentence, current sentence, and next sentence, $\boldsymbol{X}_p \in \mathbb{R}^{d_w \times N_p}$, $\boldsymbol{X}_w \in \mathbb{R}^{d_w \times N}$, and $\boldsymbol{X}_n \in \mathbb{R}^{d_w \times N_n}$ are made up of all word embeddings in $S_p$, $S$, and $S_n$, respectively.

This means, the word-level contextual attention mechanism produces attention weight vectors $\boldsymbol{\alpha}_p$ and $\boldsymbol{\alpha}_n$, and weighted hidden representations $\boldsymbol{X^*}_p$ and $\boldsymbol{X^*}_n$ as follows:

$$\boldsymbol{\alpha}_p = softmax(\max_{col}(\frac{\boldsymbol{X}_p^T \cdot \boldsymbol{X}_w}{\sqrt{d_w}})) \tag{6}$$

$$\boldsymbol{\alpha}_n = softmax(\max_{col}(\frac{\boldsymbol{X}_n^T \cdot \boldsymbol{X}_w}{\sqrt{d_w}})) \tag{7}$$

$$\boldsymbol{X^*}_p = \boldsymbol{X}_w \odot (\boldsymbol{\alpha}_p \otimes \boldsymbol{e}_{d_w}) \tag{8}$$

$$\boldsymbol{X^*}_n = \boldsymbol{X}_w \odot (\boldsymbol{\alpha}_n \otimes \boldsymbol{e}_{d_w}) \tag{9}$$

where the contextual attention weights vectors $\boldsymbol{\alpha}_p$, $\boldsymbol{\alpha}_n \in \mathbb{R}^N$, and the contextual representation matrices $\boldsymbol{X^*}_p$, $\boldsymbol{X^*}_n \in \mathbb{R}^{d_w \times N}$. The operator $\max_{col}(\cdot)$ in Eq.(6) and Eq.(7) means extracting the maximum value for each column in matrix to generate a vector. Besides, we apply the scaling factor of $1/\sqrt{d_w}$ to counteract the effect that the dot products grow large in magnitude (Vaswani et al., 2017). The operation $\boldsymbol{\alpha} \otimes \boldsymbol{e}_{d_w}$ in Eq.(8) and

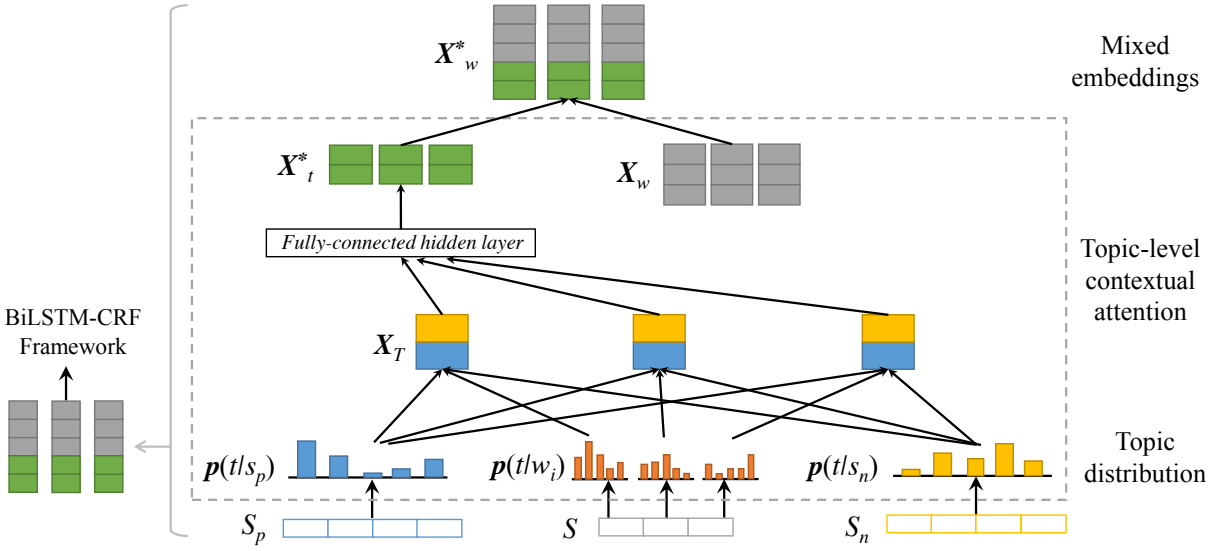


Figure 2: Architecture of topic-level contextual attention mechanism for negative focus detection.

Eq.(9) means that it repeatedly concatenates $\boldsymbol{\alpha}$ for $d_w$ times, where $\boldsymbol{e}_{d_w}$ is a row vector with $d_w$ of 1s.

Finally, the final sentence representation is obtained by

$$\boldsymbol{X^*}_w = [\boldsymbol{X^*}_p; \boldsymbol{X}_w; \boldsymbol{X^*}_n], \tag{10}$$

and we replace $\boldsymbol{X}_w$ with $\boldsymbol{X^*}_w$ in the embedding layer of the BiLSTM-CRF framework.

### 3.3 Topic-level Contextual Attention

To acquire the latent topical distribution of words, we apply the GenSim topic modeling package[2] (Rehurek and Sojka, 2010) to generate an LSI semantic model (Bradford, 2008) and obtain two probabilities with the topic distribution derived from the Wikipedia corpus[3]: 1) $\boldsymbol{p}(t|w)$, the topic distribution given word $w$, and 2) $\boldsymbol{p}(t|s)$, the topic distribution given sentence $s$. The correlation between the word $w$ and the sentence $s$ can be calculated by the similarity between their corresponding topic distribution:

$$\boldsymbol{Sim}_{ws} = \boldsymbol{p}(t|w) \odot \boldsymbol{p}(t|s). \tag{11}$$

Figure 2 illustrates the architecture of topic-level contextual attention mechanism. In this paper, we obtain the topic distribution of each word in current sentence $\boldsymbol{p}(t|w_i)$ and its adjacent sentences $\boldsymbol{p}(t|s_p)$ and $\boldsymbol{p}(t|s_n)$, respectively. The topic-level contextual attention mechanism produces the topical hidden representations

$$\boldsymbol{x}_{t_i} = [\boldsymbol{Sim}_{w_i s_p}; \boldsymbol{Sim}_{w_i s_n}], \tag{12}$$

where $\boldsymbol{x}_{t_i} \in \mathbb{R}^{2d_T}$, and $d_T$ is the dimension of topic distribution. Suppose that $N$ denotes the length of current sentence, $\boldsymbol{X}_T \in \mathbb{R}^{2d_T \times N}$ is made up of all words' topical hidden representations. To better learn the latent features and expand the dimension of $\boldsymbol{X}_T$ from $2d_T$ to $d_t$, we utilize a fully-connected hidden layer:

$$\boldsymbol{X^*}_t = \tanh(\boldsymbol{W}\boldsymbol{X}_T + \boldsymbol{b}), \tag{13}$$

where $\boldsymbol{W} \in \mathbb{R}^{d_t \times 2d_T}$ is the transpose of parameter matrix and $\boldsymbol{b} \in \mathbb{R}^{d_t}$. Note that $d_t$ is a hyper-parameter are fine-tuned on the development set. As a result, we can obtain the sentence representation as follows

$$\boldsymbol{X^*}_w = [\boldsymbol{X}_w; \boldsymbol{X^*}_t], \tag{14}$$

and replace $\boldsymbol{X}_w$ with $\boldsymbol{X^*}_w$ in the embedding layer of the BiLSTM-CRF framework.

## 4 Experimentation

In this section, we first introduce the details of dataset, hyper-parameters settings, and baselines. Then we compare our model with existing systems and baselines to demonstrate the effectiveness of the proposed contextual attention mechanisms. Finally, we perform in-depth analysis of the impact factors of our approach and conduct careful error analysis to better understand the limitation of our model.

[2] https://radimrehurek.com/gensim/index.html
[3] https://radimrehurek.com/gensim/wiki.html

| Hyper-parameter | Value |
|---|---|
| word embed dim $d_w$ | 1,024 |
| semantic role embed dim $d_{sr}$ | 200 |
| topic embed dim $d_t$ | 300 |
| topic number $T$ | 80 |
| LSTM hidden layer size $h$ | 250 |
| dropout probability $p$ | 0.3 |
| learning rate $\eta$ | 0.015 |

Table 1: Hyper-parameter settings.

| | System | Acc |
|---|---|---|
| Existing Methods | CLaC (Rosenberg (2012)) | 60.00 |
| | FOC-DET (Blanco (2013)) | 65.50 |
| | WTGM (Zou (2015)) | 68.40 |
| Our Methods | LSTM | 58.71 |
| | BiLSTM | 60.81 |
| | BiLSTM-CRF | 67.28 |
| | W-Att BiLSTM-CRF | 70.22 |
| | T-Att BiLSTM-CRF | **70.51** |
| | WT-Att BiLSTM-CRF | 69.80 |

Table 2: Performances of negative focus detection systems on the SEM'12 corpus.

### 4.1 Dataset and Settings

All of experiments conduct on the SEM'12 shared task corpus[4] (Morante and Blanco, 2012) which is annotated on top of PropBank. According to the guideline (Blanco and Moldovan, 2011), negative focus is restricted to verbal negation (marked with MNEG labels), thus all of the words belonging to a semantic role most likely to correspond to the verbal negation are selected as negative focus. In total, this corpus contains 3,544 instances of negative focus. For fair comparison, we adopt the same partition as SEM'12 shared task in all experiments, i.e., with 2,304 for training, 531 for development, and 712 for testing. For each instance, the corpus provides the previous and next sentences as contexts. We evaluate our results in terms of accuracy (acc for short), i.e., for each negation, the predicted focus is considered correct if it is a perfect match with its gold annotation.

Table 1 lists the hyper-parameters in our experiments. We utilize the pre-trained word embeddings by ELMo[5] (Peters et al., 2018). The semantic role embeddings are initialized randomly by a continuous uniform distribution. All the models are optimized using the stochastic gradient descent (SGD). We pick the parameters showing the best performance on the development set when no further improvement occurs within 50 epochs, and report the performances on the test set.

To verify the effectiveness of our approach, we compare with several baselines on negative focus detection, which are briefly introduced as follows.

**LSTM**: The LSTM model which contains only a forward LSTM layer with a fully-connected layer followed by a softmax classifier.

**BiLSTM**: The BiLSTM model employs the same architecture as LSTM, except a additional backward LSTM layer.

**BiLSTM-CRF**: The BiLSTM-CRF model described in Subsection 3.1 without any contextual attention mechanism.

**W-Att BiLSTM-CRF**: The word-level contextual attention-based BiLSTM-CRF model described in Subsection 3.2.

**T-Att BiLSTM-CRF**: The topic-level contextual attention-based BiLSTM-CRF model described in Subsection 3.3.

**WT-Att BiLSTM-CRF**: The BiLSTM-CRF model with an input representation $[\boldsymbol{X^*}_p; \boldsymbol{X}_w; \boldsymbol{X^*}_n; \boldsymbol{X^*}_t]$. This system combines the above two types of contextual attention mechanisms.

**CLaC**: A heuristic-based system with three linguistic rules related to adverb constituent, passive, and negative trigger of focus, respectively (Rosenberg and Bergler, 2012).

**FOC-DET**: A decision tree based system with 22 kinds of manually designed features (Blanco and Moldovan, 2013).

**WTGM**: A graph model consists of a word layer and a topic layer to capture lexical contextual information (Zou et al., 2015).

### 4.2 Results

Table 2 shows the performance comparison of various negative focus detection models. We can see that all of contextual attention based models (row 7-9 in Table 2) achieve better perfomances than existing methods (row 1-3 in Table 2) and models without contextual attention (row 4-6 in Table 2). In addition, both word-level and topic-level contextual attention based models outperform the state-of-the-art system (WTGM in Table 2) with about 2% accuracy gain at least. The

[4]http://www.clips.ua.ac.be/sem2012-st-neg
[5]https://allennlp.org/elmo

| Method | Context | Acc |
|---|---|---|
| No Attention | N/A | 67.28 |
| Word-level Attention | only previous sentence | 69.24 |
| | only next sentence | 69.94 |
| | previous+next sentences | **70.22** |
| Topic-level Attention | only previous sentence | 69.38 |
| | only next sentence | 70.08 |
| | previous+next sentences | **70.51** |

Table 3: Performance comparison by using different contextual information into the BiLSTM-CRF framework.

| Method | Embeddings | Acc |
|---|---|---|
| BiLSTM-CRF | only word | 67.28 |
| | word + SR | 68.82 |
| W-Att BiLSTM-CRF | only word | 68.26 |
| | word + SR | 70.22 |
| T-Att BiLSTM-CRF | only word | 68.40 |
| | word + SR | 70.51 |

Table 4: Performance comparison for systems only using word embeddings and those using word and SR embeddings (SR: semantic role).

| Semantic Role | Train | Dev | Test |
|---|---|---|---|
| A1 | 980 | 222 | 309 |
| AM-NEG | 592 | 138 | 172 |
| AM-TMP | 161 | 35 | 46 |
| AM-MNR | 127 | 27 | 38 |
| A2 | 112 | 28 | 36 |
| A0 | 94 | 23 | 31 |
| AM-ADV | 78 | 23 | 26 |
| Others | 165 | 30 | 61 |
| NONE | 88 | 19 | 35 |
| Total | 2,304 | 531 | 712 |

Table 5: Statistics of the top seven semantic role labels of negative focus. NONE class: the negative focus has at least one word that does not belong to any role.

results demonstrate the effectiveness of these two types of contextual attention mechanisms.

Moreover, to better quantify the contribution of the different attention mechanisms of our approach, we also conduct several attention variants. Comparing the three types of attention mechanisms (row 7-9 in Table 2), the topic-level attention based model achieves the best performance. We also observe that the word-level attention based model also achieves comparative performance with the topic-level one. However, when combining the two types of attention mechanisms, the performance declines. It indicates that both the word-level attention mechanism and the topic-level one can capture the contextual information effectively, but applying such information repeatedly might lead to feature redundancy.

In addition to the methods that take advantage of the contextual features in adjacent sentences, we also compare the performances of different frameworks for negative focus detection (row 4-6 in Table 2), which only apply the features in current sentence. The results indicate that the BiLSTM-CRF framework is a better fit for encoding order information and long-range context dependency for such sequence labeling task.

### 4.3 Analysis and Discussion

**Impact of Contextual Information**. We examine the impact of the context position on the performance for our approach. Table 3 shows the performance comparison when using different contextual information. We can see that, both the word-level and topic-level attention based models achieve the best performances by utilizing the contextual features from the previous and next sentences, which verifies the effectiveness of the two types of contexts. Besides, the results also indicate that contextual information is key for negative focus detection, especially the next sentence.

**Impact of Semantic Role Information**. Since a completed and exclusive semantic role that is most likely negated is annotated as a negative focus in sentence, according to the annotation guideline (Blanco and Moldovan, 2011), semantic role information plays a critical role in predicting the focus. Therefore, we compare the performances between the models adding semantic role information in embedding layer and those only using the word embeddings. As shown in Table 4, we can see that the performances of all three models are improved obviously, which demonstrates the effectiveness of semantic role information for this task.

In addition, Table 5 lists the top seven semantic role labels of negative focus in the SEM'12 shared task corpus. We can observe that the distribution of semantic role labels is relatively concentrated into A1, AM-NEG, AM-TMP, and AM-MNR. Unlike other negation processing task, such as scope resolution, which heavily relies on the

| Method | Composition Manner | Acc |
|---|---|---|
| W-Att | within input layer | **70.22** |
| BiLSTM-CRF | within hidden layer | 69.38 |
| T-Att | within input layer | **70.51** |
| BiLSTM-CRF | within hidden layer | 69.80 |

Table 6: Performance comparison for different composition manners of attention matrices.

| Word Embedding | Dimension | Acc |
|---|---|---|
| Senna | 50 | 70.08 |
| Glove | 100 | 69.66 |
| Word2vec | 300 | 69.10 |
| BERT | 768 | 70.22 |
| ELMo | 1,024 | **70.51** |

Table 7: Performance comparison for different pre-trained word embeddings on the T-Att BiLSTM-CRF model.

lexical or syntactic features (Zou et al., 2013; Qian et al., 2016), the negative focus detection task tends to consider more semantic relation between the focus and other words and phrases.

**Impact of Attention Manner**. In this paper, our contextual attention matrices are concatenated into the input embeddings. In addition, we also explore another manner of adding attention mechanisms, which directly concatenates the attention matrix into the hidden layer of BiLSTM-CRF framework. As shown in Table 6, it indicates that the composition manner within the input layer is more effective than that within the hidden layer for both word-level and topic-level attention mechanisms.

**Impact of Pre-trained Word Embedding**. To compare the impacts of different pre-trained word embeddings for negative focus detection task, we attempt to employ other pre-trained word embeddings. Table 7 show the performances of the T-Att BiLSTM-CRF model with different pre-trained word embeddings, including Senna[6], Glove[7], Word2vec[8], and BERT[9]. We can see that ELMo achieves the best performance, but the performance gaps between different pre-trained word embeddings and different dimensions are not significant.

[6]http://ronan.collobert.com/senna
[7]http://nlp.stanford.edu/projects/glove
[8]https://code.google.com/archive/p/word2vec
[9]https://github.com/google-research/bert

| Embeddings | Acc |
|---|---|
| word | 67.28 |
| word+chunking label | 67.84 |
| word+dependency label | 68.12 |
| word+PoS tag | 68.26 |
| word+semantic role | **68.82** |
| all features | 68.54 |

Table 8: Performance comparison for different features with the BiLSTM-CRF model.

**Impact of Different Features**. Table 8 shows the comparison of the BiLSTM-CRF models with different feature embeddings. The reason we employ the basic model is to avoid that the impacts of different features are covered by contextual attention mechanisms. The results indicate that the semantic role feature is the most effective for negative focus detection, which confirms our analysis above. In addition, the performance of the system with all features is lower than that only using the semantic role feature. It might be that other features cannot capture more effective information than the semantic role features.

**Error Analysis**. We manually analyzed a total of 210 of the incorrect instances labeled by the T-Att BiLSTM-CRF system. The main error patterns include:

1) Focus is the negative trigger (50/210, 23.8%). According to the annotation guideline of negative focus (Blanco and Moldovan, 2011), when a negated statement does not carry any positive meaning, or if it cannot be inferred that a part of sentence is negated, the gold standard annotation regards the negative trigger (e.g., *not*) itself as the negative focus. In this case, neither the local information nor the contextual information could identify the negative focus. It is difficult to accurately identify such negative focus by our approach.

2) Annotation issue (84/210, 40%). As mentioned in the above, a negative focus always corresponds a single semantic role. However, we find that some of labeled instances do not meet this constraint. Such non-conformity contains three aspects: a) the focus has at least one word that does not belong to any role (35/84), b) the focus is labeled as a fragment of semantic role (36/84), and c) the focus is labeled across more than one roles (13/84). These situations may heavily limit the effects of the semantic features in embedding layer and the CRF layer which learns the depen-

dencies across output labels. To verify the impact of this issue, we conduct the T-Att BiLSTM-CRF system on the processed test set that remove the 84 of instances. The performance achieves 79.94 in accuracy, yielding an improvement of 9.43% (Table 2). Therefore, the future work should fix the issue of corpus, or reconsider whether the guidelines could generalize the linguistic phenomenon of negative focus.

3) Subjectivity of the annotation process (11/210, 5.2%). There are several golden annotated instances that we disagree with. For example,

(S2) Previous sentence: ..., *one equipment failure forces a complete plant shutdown*.

(S3) {*On some days*}, *the Nucor plant doesnt* ***produce*** *anything*.[10]

Consider the previous sentence S2, the author emphasize that *the Nucor plant will shut down*, thus the most likely negative focus should be *anything* in S3, whereas the prepositional phrase *on some days* only provides a temporal statement in our view. However, different annotators might have different understandings and interpretations for this case. Note that the inter-annotator agreement of the corpus is only 0.72 after all, which indicates the inherent challenge in negative focus detection task.

## 5 Conclusion

In this work, we discussed the unique necessity of modeling contextual information for negative focus detection. We designed an attention based neural architecture which can better capture contextual features by utilizing the word-level and topic-level attention mechanisms. Experiments on the SEM'12 shared task corpus demonstrate the effectiveness of our approach. The datasets and source code of this paper are publicly available at `http://github.com/longxshen/NegFocus`.

In the future, we intend to review the annotated guidelines for negative focus with the problematic annotation cases mentioned in this paper, and explore the ways to combine additional patterns and constraints from negative focus definition with neural network techniques to further improve negative focus detection.

[10]#287 of test set in SEM'12 shared task corpus.


## Acknowledgements

This research is supported by the National Natural Science Foundation of China (No. 61703293, No. 61751206, and No. 61672368). This work was also supported by the joint research project of Alibaba and Soochow University. Finally, we would like to thank the anonymous reviewers for their insightful comments and suggestions.


## References


Pranav Anand and Craig Martell. 2012. Annotating the focus of negation in terms of questions under discussion. In *Proceedings of the ACL-2012 Workshop on Extra-Propositional Aspects of Meaning in Computational Linguistics (ExProM)*, pages 65–69. Association for Computational Linguistics.

Emilia Apostolova, Noriko Tomuro, and Dina Demner-fushman. 2011. Automatic extraction of lexico-syntactic patterns for detection of negation and speculation scopes. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 283–287. Association for Computational Linguistics.

Rajendra Banjade and Vasile Rus. 2016. Dt-neg: Tutorial dialogues annotated for negation scope and focus in context. In *Proceedings of Language Resources and Evaluation Conference (LREC)*, pages 3768–3771.

Eduardo Blanco and Dan Moldovan. 2011. Semantic representation of negation using focus detection. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 581–589. Association for Computational Linguistics.

Eduardo Blanco and Dan Moldovan. 2013. Retrieving implicit positive meaning from negated statements. *Natural Language Engineering (NLE)*, 20(4):501–535.

Roger Bradford. 2008. An empirical study of required dimensionality for large-scale latent semantic indexing applications. In *Proceedings of the 17th Conference on Information and Knowledge Management (CIKM)*, pages 153–162.

Wendy W Chapman, Will Bridewell, Paul Hanbury, Gregory F Cooper, and Bruce G Buchanan. 2001. A simple algorithm for identifying negated findings and diseases in discharge summaries. *Journal of Biomedical Informatics*, 34(5):301–310.

Noa P Cruz, Maite Taboada, and Ruslan Mitkov. 2016. A machine-learning approach to negation and speculation detection for sentiment analysis. *Journal of the Association for Information Science and Technology*, 67(9):2118–2136.

Ilya M. Goldin and Wendy W. Chapman. 2003. Learning to detect negation with not in medical texts. In *Proceedings of the Workshop on Text Analysis and Search for Bioinformatics of SIGIR*.

Sepp Hochreiter and Jurgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Rodney D. Huddleston and Geoffrey K. Pullum. 2002. *The Cambridge Grammar of the English Language*. Cambridge University Press.

Salud Maria Jimenez-Zafra, M Teresa Martin Valdivia, Eugenio Martinez Camara, and Luis Alfonso Urenalopez. 2017. Studying the scope of negation for spanish sentiment analysis on twitter. *IEEE Transactions on Affective Computing*.

John D. Lafferty, Andrew Mccallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Eighteenth International Conference on Machine Learning*, pages 282–289.

Suguru Matsuyoshi, Ryo Otsuki, and Fumiyo Fukumoto. 2014. Annotating the focus of negation in japanese text. In *Proceedings of Language Resources and Evaluation Conference (LREC)*, pages 1743–1750.

Saeed Mehrabi, Anand Krishnan, Sunghwan Sohn, Alexandra M Roch, Heidi Schmidt, Joe Kesterson, Chris Beesley, Paul Dexter, C Max Schmidt, Hongfang Liu, et al. 2015. Deepen: A negation detection system for clinical text incorporating dependency relation into negex. *Journal of Biomedical Informatics*, 54:213–219.

Roser Morante. 2010. Descriptive analysis of negation cues in biomedical texts. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC)*, pages 1429–1436.

Roser Morante and Eduardo Blanco. 2012. *sem 2012 shared task: Resolving the scope and focus of negation. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics (*SEM)*, pages 256–274. Association for Computational Linguistics.

Roser Morante, Anthony Liekens, and Walter Daelemans. 2008. Learning the scope of negation in biomedical texts. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 715–724. Association for Computational Linguistics.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. pages 2227–2237.

Zhong Qian, Peifeng Li, Qiaoming Zhu, Guodong Zhou, Zhunchen Luo, and WeiLuo. 2016. Speculation and negation scope detection via convolutional neural networks. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 815–825.

Radim Rehurek and Petr Sojka. 2010. Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA.

Sabine Rosenberg and Sabine Bergler. 2012. Uconcordia: Clac negation focus detection at *sem 2012. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics (*SEM)*, pages 294–300. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing (EMNLP)*, pages 1631–1642. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS)*, pages 5998–6008.

Veronika Vincze, Gyorgy Szarvas, Richard Farkas, Gyorgy Mora, and Janos Csirik. 2008. The bioscope corpus: biomedical texts annotated for uncertainty, negation and their scopes. *BMC Bioinformatics*, 9(11):1–9.

Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 606–615. Association for Computational Linguistics.

Michael Wiegand, Alexandra Balahur, Benjamin Roth, Dietrich Klakow, and Andres Montoyo. 2010. A survey on the role of negation in sentiment analysis. In *Proceedings of the Workshop on Negation and Speculation in Natural Language Processing*, pages 60–68.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 207–212. Association for Computational Linguistics.

Bowei Zou, Guodong Zhou, and Qiaoming Zhu. 2013. Tree kernel-based negation and speculation scope

detection with structured syntactic parse features. In *Proceedings of the 2013 conference on empirical methods in natural language processing (EMNLP)*, pages 968–976. Association for Computational Linguistics.

Bowei Zou, Guodong Zhou, and Qiaoming Zhu. 2014. Negation focus identification with contextual discourse information. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 522–530. Association for Computational Linguistics.

Bowei Zou, Qiaoming Zhu, and Guodong Zhou. 2015. Unsupervised negation focus identification with word-topic graph model. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1632–1636. Association for Computational Linguistics.

# A Unified Neural Coherence Model

**Han Cheol Moon**[*¶], **Tasnim Mohiuddin**[*¶], **Shafiq Joty**[*¶§], **and Xu Chi**[†]
[¶]Nanyang Technological University, Singapore
[§]Salesforce Research Asia, Singapore
[†]A*STAR, Singapore
[¶]{hancheol001@e., mohi0004@e.}ntu.edu.sg
[§]sjoty@salesforce.com
[†]cxu@simtech.a-star.edu.sg

## Abstract

Recently, neural approaches to coherence modeling have achieved state-of-the-art results in several evaluation tasks. However, we show that most of these models often fail on harder tasks with more realistic application scenarios. In particular, the existing models underperform on tasks that require the model to be sensitive to local contexts such as candidate ranking in conversational dialogue and in machine translation. In this paper, we propose a unified coherence model that incorporates sentence grammar, inter-sentence coherence relations, and global coherence patterns into a common neural framework. With extensive experiments on local and global discrimination tasks, we demonstrate that our proposed model outperforms existing models by a good margin, and establish a new state-of-the-art.

## 1 Introduction

Coherence modeling involves building text analysis models that can distinguish a coherent text from incoherent ones. It has been a key problem in discourse analysis with applications in text generation, summarization, and coherence scoring.

Various linguistic theories have been proposed to formulate coherence, some of which have inspired development of many of the existing coherence models. These include the **entity-based** local models (Barzilay and Lapata, 2008; Elsner and Charniak, 2011b) that consider syntactic realization of entities in adjacent sentences, inspired by the Centering Theory (Grosz et al., 1995). Another line of research uses **discourse relations** between sentences to predict local coherence (Pitler and Nenkova, 2008; Lin et al., 2011). These methods are inspired by the discourse structure theories like Rhetorical Structure Theory (RST) (Mann and Thompson, 1988) that formalizes coherence in terms of discourse relations. Other notable methods include **word co-occurrence** based local models (Soricut and Marcu, 2006), **content** (or topic distribution) based global models (Barzilay and Lee, 2004), and **syntax** based local and global models (Louis and Nenkova, 2012).

With the neural invasion, some of the above traditional models have got their neural versions with much improved performance. For example, Li and Hovy (2014) implicitly model syntax and inter-sentence relations using a neural framework that uses a recurrent (or recursive) layer to encode each sentence and a fully-connected layer with sigmoid activations to estimate coherence probability for every window of three sentences. Li and Jurafsky (2017) incorporate global topic information with an encoder-decoder architecture, which is also capable of generating discourse. Mesgar and Strube (2018) model change patterns of salient semantic information between sentences. Nguyen and Joty (2017); Mohiuddin et al. (2018) propose neural entity grid models using convolutions over distributed representations of entity transitions, and report state-of-the-art results in standard evaluation tasks on the Wall Street Journal (WSJ) corpus.

Traditionally coherence models have been evaluated on two kinds of tasks. The first kind includes synthetic tasks such as *discrimination* and *insertion* that evaluate the models directly based on their ability to identify the right order of the sentences in a text with different levels of difficulty (Barzilay and Lapata, 2008; Elsner and Charniak, 2011b). The latter kind of tasks evaluate the impact of coherence score as another feature in downstream applications like *readability assessment* and *essay scoring* (Barzilay and Lapata, 2008; Mesgar and Strube, 2018).

Although coherence modeling has come a long way in terms of novel models and innovative evaluation tasks (Elsner and Charniak, 2011a; Mohi-

*Equal contribution



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2262–2272, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

uddin et al., 2018), it is far from being solved. As we show later, state-of-the-art models often fail on harder tasks like *local discrimination* and *insertion* that ask the model to evaluate a local context (*e.g.,* a 3-sentence window). This task has direct applications in utterance ranking (Lowe et al., 2015) or bot detection[1] in dialogue, and for sentence ordering in summarization.

According to Grosz and Sidner (1986), three factors *collectively* contribute to discourse coherence: (*a*) the organization of discourse segments, (*b*) intention or purpose of the discourse, and (*c*) attention or focused items. The entity-based approaches capture attentional structure, the syntax-based approaches consider intention, and the organizational structure is largely captured by models that consider discourse relations and content (topic) distribution. Although methods like (Elsner et al., 2007; Li and Jurafsky, 2017) attempt to combine these aspects of coherence, to our knowledge, no methods consider all the three aspects together in a single framework.

In this paper, we propose a unified neural model that incorporates sentence grammar (intentional structure), discourse relations, attention and topic structures in a single framework. We use an LSTM sentence encoder with explicit language model loss to capture the syntax. Inter-sentence discourse relations are modeled with a bilinear layer, and a lightweight convolution-pooling is used to capture the attention and topic structures. We evaluate our models on both local and global discrimination tasks on the benchmark dataset. Our results show that our approach outperforms existing methods by a wide margin in both tasks. We have released our code at https://ntunlpsg.github.io/project/coherence/ncoh-emnlp19/ for research purposes.

## 2 Related Works

Inspired by various linguistic theories of discourse, many coherence models have been proposed. In this section, we give a brief overview of the existing coherence models.

Motivated by the Centering Theory (Grosz et al., 1995), Barzilay and Lapata (2005, 2008) proposed the **entity-based** local model for representing and assessing text coherence, which showed significant improvements in two out of three evaluation tasks. Their model represents a text by a two-dimensional array called *entity grid* that captures local transitions of discourse entities across sentences as the deciding patterns for assessing coherence. They consider the *salience* of the entities to distinguish between transitions of important entities from unimportant ones, by measuring the occurrence frequency of the entities.

Subsequent studies extended the basic entity grid model. By including non-head nouns as entities in the grid, Elsner and Charniak (2011b) gained significant improvements. They incorporate entity-specific features like named entity, noun class, and modifiers to distinguish between entities of different types, which led to further improvements. Instead of using the transitions of grammatical roles, Lin et al. (2011) model the transitions of **discourse roles** for entities. Feng and Hirst (2012) used the basic entity grid, but improved its learning to rank scheme. Their model learns not only from the original document and its permutations but also from ranking preferences among the permutations themselves.

Guinaudeau and Strube (2013) proposed a **graph-based** unsupervised method for modeling text coherence. Assuming the sentences in a coherent discourse should share the same structural syntactic patterns, Louis and Nenkova (2012) introduced a coherence model based on **syntactic patterns** in text. Their proposed method comprises of local and global coherence model, where the former one captures the co-occurrence of structural features in adjacent sentences and the latter one captures the global structure based on clusters of sentences with similar syntax.

Our model also considers syntactic patterns through a biLSTM sentence encoder that is trained on an explicit language modeling loss. Compared to the entity grid and the syntax-based models, our model does not require any syntactic parser.

With the *neural tsunami*, some of the above traditional models have got their **neural** versions with better performance. Li and Hovy (2014) proposed a neural framework to compute the coherence score of a document by estimating coherence probability for every window of three sentences. Li and Jurafsky (2017) proposed two *encoder-decoder* models, where the first model incorporates global discourse information (e.g., topics) by feeding the output of a sentence-level HMM-LDA model (Gruber et al., 2007) and the second model is trained end-to-end with varia-

[1] http://workshop.colips.org/wochat/data/index.html

tional inference. Our proposed model also models inter-sentence relations and global coherence patterns. We use a bi-linear layer to model relations between two consecutive sentences exclusively. Also, our global model implements a light-weight convolution that requires much less parameters, which gives better generalization. Moreover, we train the whole network end-to-end with a window-based adaptive pairwise ranking loss.

Nguyen and Joty (2017) proposed a neural version of the entity grid model where they first transform the grammatical roles in a grid into their distributed representations. Then they employ a convolution operation over it to model entity transitions in the distributed space. Finally, they compute the coherence score from the convoluted features by a spatial max-pooling operation. The model is trained with a **document-level** (global) pairwise ranking loss. Mohiuddin et al. (2018) improve the neural entity grid model by *lexicalizing* its entity transitions They use *off-the-shelf* word embeddings to achieve better generalization with the lexicalized model. As we will demonstrate, because of the spatial-pooling operation, entity-based neural models are not sensitive to mismatch of **local patterns** in a document limiting their applicability to tasks that require local discrimination. Another crucial limitation of employing a **document-level** pairwise ranking loss is that the loss from the document-level permutation for a negative document may **penalize the convolution kernel weights** even if the local permutation matches that of the positive document. In contrast, we apply a window-level (local) **adaptive pairwise ranking loss** that gets activated only if the corresponding windows of the positive and negative documents differ. This way our model is sensitive to local patterns without penalizing the weights unfairly. We capture global patterns using a separate light-weight convolution module.

## 3 Proposed Model

Let $D = (\mathbf{s}_1, \cdots, \mathbf{s}_n)$ be a document consisting of $n$ sentences. Our goal is to assess its coherence score. Figure 1 provides an overview of our proposed unified coherence model. It has four components in a Siamese architecture (Bromley et al., 1993): (*i*) a sentence encoder (§3.1), (*ii*) a local coherence model (§3.2), (*iii*) a global coherence model (§3.3), and (*iv*) a coherence scoring layer (§3.4). For encoding a sentence, we first map each word of the sentence to its corresponding vector representation. We then use a bidirectional LSTM sentence encoder with explicit **language model loss** to capture the sentence grammar. Given the sentence representations, the local and global coherence model extract the respective coherence features. The local coherence model implements a bilinear layer to model inter-sentence discourse relations. This layer captures the **local contexts** of the document. To capture the attention (entity distribution) and topic structures, i.e., the **global coherence** of the document, our global coherence model uses a *light-weight convolution* (Wu et al., 2019) with average pooling. The coherence scoring is a linear layer that evaluates the coherence from the extracted features. The whole architecture is trained *end-to-end* with a pairwise ranking loss. In the following, we elaborate on different components of our proposed model.

### 3.1 Modeling Intention

A discourse has a purpose such as describing an event, explaining some results, evaluating a product, etc. As such, sentences in the discourse should support the purpose as a whole. The syntactic structure of the sentence can be used to model the intent structure (Louis and Nenkova, 2012). We use a bidirectional long short-term memory or bi-LSTM (Hochreiter and Schmidhuber, 1997) to encode each sentence into a vector representation while modeling its compositional structure.

For an input sentence $\mathbf{s}_i = (w_1, \cdots, w_m)$ of length $m$, we first map each word $w_t$ to its corresponding vector representation $\mathbf{e}_t \in \mathbb{R}^d$, where $d$ is the dimension of the word embedding. The LSTM recurrent layer then computes a compositional representation $\mathbf{h}_t \in \mathbb{R}^p$ at every time step $t$ by performing nonlinear transformations of the current time step word vector representation $\mathbf{e}_t$ and the output of the previous time step $\mathbf{h}_{t-1}$, where $p$ is the number of features in the LSTM hidden state. The output of the last time step $\mathbf{h}_m$ is considered as the representation of the sentence. A bi-LSTM processes a given sentence $\mathbf{s}_i$ in two directions: from *left-to-right* and *right-to-left*, yielding a representation $\mathbf{h}_i = [\overrightarrow{\mathbf{h}_m}; \overleftarrow{\mathbf{h}_m}] \in \mathbb{R}^{2p}$, where ';' denotes concatenation.

We train our sentence encoder with an explicit **language model loss**. A bidirectional language model combines a forward and a backward language model (LM). Similar to Peters et al. (2018),

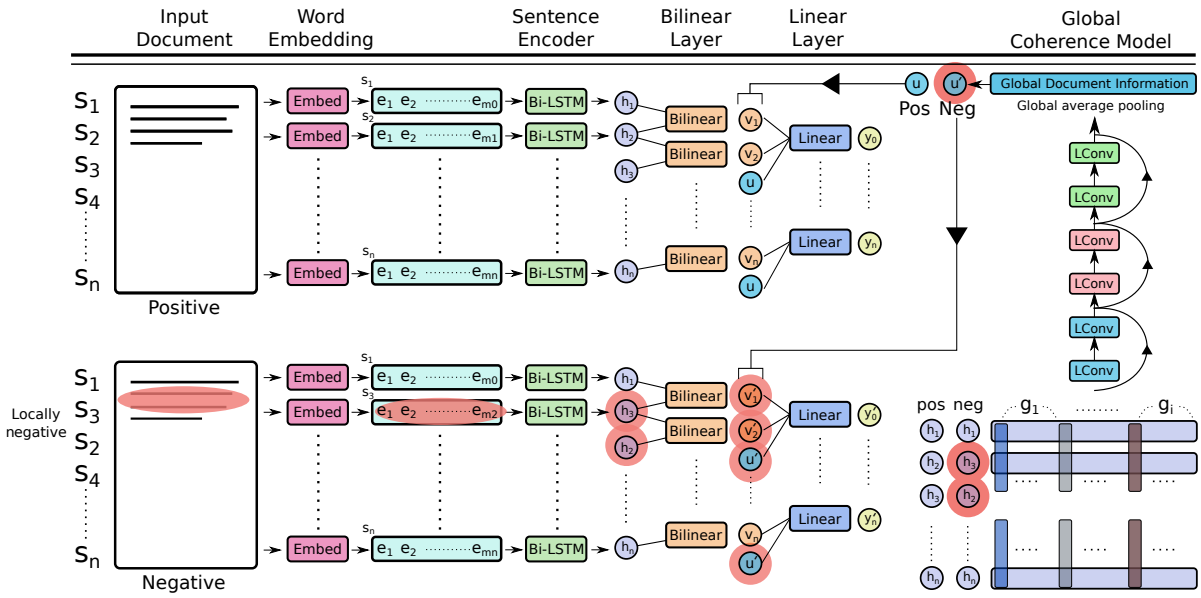


Figure 1: An overview of the proposed coherence model (best viewed in color). The superscript ‘/’ above output of each component denotes negative outputs and the red shade represents incoherent portions in the document. Note that all network parameters and components are shared regardless of the input documents.

we jointly minimize the *negative log-likelihood* of the forward and backward directions:

$$\mathcal{L}_{lm} = -\sum_{j=1}^{m}\Big(\log p(w_j|w_1,\cdots,w_{j-1};\theta,\overrightarrow{\theta}_{\text{lstm}}) + \log p(w_j|w_{j+1},\cdots,w_m;\theta,\overleftarrow{\theta}_{\text{lstm}})\Big) \quad (1)$$

where $\overrightarrow{\theta}_{\text{lstm}}$ and $\overleftarrow{\theta}_{\text{lstm}}$ are the parameters of the forward and backward LSTMs, and $\theta$ denote the rest of the parameters which are shared.

### 3.2 Modeling Inter-Sentence Relation

Discourse relations between sentences reflect the organizational structure of a discourse that can be used to evaluate the coherence of a text (Lin et al., 2011; Li and Hovy, 2014). To model inter-sentence discourse relations, we use a bilinear model. Our bi-LSTM sentence encoder gives a representation $\mathbf{h}_i \in \mathbb{R}^{2p}$ for each sentence $\mathbf{s}_i$ in the document. We feed the representations of every two consecutive sentences $(\mathbf{h}_i, \mathbf{h}_{i+1})$ to this layer, which applies a bilinear transformation as:

$$\mathbf{v}_i = \mathbf{h}_i^T \mathbf{W}_b \mathbf{h}_{i+1} + \mathbf{b} \quad (2)$$

where $\mathbf{W}_b \in \mathbb{R}^{q\times 2p\times 2p}$ is a learnable tensor, and $\mathbf{b} \in \mathbb{R}^q$ is a learnable bias vector. Here $q$ is the number of output features (i.e., $\mathbf{v}_i \in \mathbb{R}^q$).

### 3.3 Modeling Global Coherence Patterns

The model proposed so far captures only local information. However, global discourse phenomena like entity or topic distributions are also important for coherence evaluation (Barzilay and Lee, 2004; Elsner et al., 2007; Louis and Nenkova, 2012). Global coherence is modeled in our architecture by a *convolution-pooling* mechanism.

As shown in the Figure 1, our **global coherence** sub-module takes the representations $\mathbf{H} = (\mathbf{h}_1, \cdots, \mathbf{h}_n)$ of all the sentences in a document generated by the bi-LSTM encoder. The module uses six convolution layers with residual connections, followed by an average pooling layer. Instead of using regular convolutions, we use *lightweight convolution* (Wu et al., 2019), which is built upon *depth-wise* convolution (Chollet, 2016). **Depth-wise convolutions** perform a convolution independently over every input channel which significantly reduces the number of parameters as shown in Figure 2. For a given input $\mathbf{H} \in \mathbb{R}^{n\times d}$, the output $\mathbf{O} \in \mathbb{R}^{n\times d}$ of the depth-wise convolution with convolution weight $\mathbf{W} \in \mathbb{R}^{d\times k}$ with kernel size $k$ for element $i$ and output dimension $c$ can be written as:

$$\mathbf{O}_{i,c} = \text{Dcon}(\mathbf{H}, \mathbf{W}_{c,:}, i, c) = \sum_{j=1}^{k} \mathbf{W}_{c,j} \mathbf{H}_{(i+j-\lceil\frac{k+1}{2}\rceil),c} \quad (3)$$

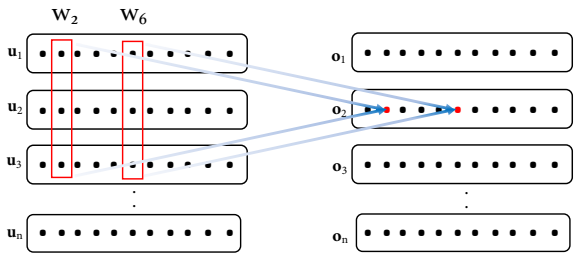


Figure 2: Depth-wise convolution for kernel size $k = 3$. The convolutions are done over the input dimensions.

Compared to the regular convolutions,depth-wise convolutions reduces the number of parameters from $d^2k$ to $dk$ (note that $d = 2p$ in our case).

**Light-weight Convolutions** make the depth-wise convolution even simpler by sharing groups of output channels and normalizing weights across the temporal dimension using a softmax. It has a fixed context window which determines the importance of context elements with a set of weights that do not change over time steps. For the $i$-th element in the sequence and output channel $c$, *light-weight convolution* computes:

$$\begin{aligned}&\text{Lcon}(\mathbf{H}, \mathbf{W}_{g,:}, i, c)\\ &\quad = \text{Dcon}(\mathbf{H}, \text{softmax}(\mathbf{W}_{g,:}), i, c) \quad (4)\end{aligned}$$

where $g = \lceil \frac{cG}{d} \rceil$ with $G$ being the number of groups. The number of parameters with light-weight convolutions reduces to $H.k$. Wu et al. (2019) show that models equipped with light-weight convolution exhibit better generalization compared to regular convolutions. It is indeed crucial in our case since we use convolutions to model a document, with large kernel size it would be difficult to learn from small datasets compared to (sentence-level) machine translation datasets.

The light-weight convolution layers generate $d$ feature maps $\mathbf{f}^i \in \mathbb{R}^n, i = 1, ..., d$ for each input document by the convolutional operation over an input dimension across all the sentences in a document. Subsequently, **global average pooling** is performed over the extracted feature maps to achieve a global view of the input document. The achieved global feature $\mathbf{u} \in \mathbb{R}^d$ can be expressed as follows:

$$\mathbf{u} = \frac{1}{n}[\mathbf{1} \cdot \mathbf{f}^1, ..., \mathbf{1} \cdot \mathbf{f}^d] \quad (5)$$

where $\mathbf{1} \in \mathbb{R}^n$ is the vector of ones and $n$ is the number of sentences in an input document. The global document level features $\mathbf{u}$ are then concatenated with the local features of each consecutive sentence pair $(\mathbf{v}_i; \mathbf{v}_{i+1})$ in the document, i.e., the output of the bilinear layer (see Figure 1).

$$\mathbf{z}_i = [\mathbf{v}_i; \mathbf{v}_{i+1}; \mathbf{u}] \quad (6)$$

where $\mathbf{z}_i \in \mathbb{R}^{2q+2p}$ and ';' denotes concatenation.

### 3.4 Coherence Scoring

We then feed the concatenated global and local features $\mathbf{z}_i$ to the final linear layer of our model to compute the coherence score $y_i \in \mathbb{R}^n$ for each local window.

$$y_i = \mathbf{z}_i^T \mathbf{w}_l + b_l \quad (7)$$

where $\mathbf{w}_l$ is weight vector and $b_l$ is a bias. The final decision on documents is made by summing up all local scores of documents and compares the summed scores.

### 3.5 Overall Objective and Training Details

Our model assigns a coherence score $y_i$ to every possible local window $D^\ell$ in the document $D$, where $\ell$ is the *local window* index. During implementation, the input document is *padded*, so that the number of possible local window is the same as the number of sentences $(n)$ in the document $D$.

Let $\mathbf{y} = \Omega(D|\Theta)$ define our model that produces the coherence scores $\mathbf{y} = (y_1, \ldots, y_n)$ for an input document $D$, with $\Theta$ being the parameters. We use a *window-level pairwise ranking approach* (Collobert et al., 2011) to learn $\Theta$.

Our training set contains *ordered* pairs of documents $(D_{\text{pos}}, D_{\text{neg}})$, where document $D_{\text{pos}}$ exhibits a higher degree of coherence than document $D_{\text{neg}}$. See Section 4 for details about the dataset. We seek to learn $\Theta$ that assigns higher coherence scores to $D_{\text{pos}}$ than to $D_{\text{neg}}$. We observed that the naive pairwise ranking loss that uses a fixed margin unfairly penalizes the locally positive sentences during training. In other words, the loss should be active only for local windows that differ in $D_{\text{pos}}$ and $D_{\text{neg}}$. To address this, we propose to use an **adaptive pairwise ranking loss** $\mathcal{L}_\Theta$ defined as follows.

$$\begin{aligned}\mathcal{L}_\Theta = \frac{1}{n}\sum_{\ell=1}^{n} \max\Big\{0, \phi(D^\ell_{\text{pos}}, D^\ell_{\text{neg}}) - \\ \Omega(D^\ell_{\text{pos}}|\Theta) + \Omega(D^\ell_{\text{neg}}|\Theta)\Big\} \quad (8)\end{aligned}$$

where $\phi(D^{\ell}_{\text{pos}}, D^{\ell}_{\text{neg}})$ is an adaptive margin given by

$$\phi(D^{\ell}_{\text{pos}}, D^{\ell}_{\text{neg}}) = \begin{cases} 0 & \text{if } D^{\ell}_{\text{pos}} = D^{\ell}_{\text{neg}} \\ \tau & \text{otherwise} \end{cases}$$

where $\tau$ is a margin constant.
Our total loss, $\mathcal{L}_{\Theta} = \mathcal{L}_{\Theta} + \mathcal{L}_{lm}$.

Note that our model shares all the layers and components, i.e., $\Theta$ to obtain $\Omega(D_{\text{pos}}|\Theta)$ and $\Omega(D_{\text{neg}}|\Theta)$ from a pair of document $(D_{\text{pos}}, D_{\text{neg}})$. Therefore, once trained, it can be used to score any input document independently.

## 4 Evaluation Tasks and Datasets

For comparison purposes with previous work, we evaluate our models on the standard **"global" discrimination** task (Barzilay and Lapata, 2008), where a document is compared to a *random permutation* of its sentences, which is considered to be incoherent. We also evaluate on an *inverse discrimination* task (Mohiuddin et al., 2018), where the sentences of the original document are placed in the reverse order to create the incoherent document. Similar to them, we do not train our models explicitly on this task, rather we use the trained model from the standard discrimination task. In addition and more importantly, we evaluate the models on a more challenging **"local" discrimination** task, where two documents differ only in a local context (*e.g.,* a 3-sentence window), as shown with an example in Figure 3.

**Dataset for Global Discrimination.** We follow the same experimental setting of the WSJ news dataset as used in previous works (Mohiuddin et al., 2018; Nguyen and Joty, 2017; Elsner and Charniak, 2011b; Feng et al., 2014). Similar to them, we use 20 random permutations of each document for both training and testing, and exclude permutations that match the original one. Table 1 summarizes the data sets used in global discrimination task. We randomly selected 10% of the training set for development purposes.

**Dataset for Local Discrimination.** We use the same WSJ articles used in the global discrimination task (Table 1) to create our local discrimination datasets. Sentences inside a local **window of size 3** are re-ordered to form a locally incoherent text. Only articles with **more than 10 sentences** are included in our dataset. This gives 748 documents for training and 618 for testing.

| | Sections | # Doc. | # Pairs |
|---|---|---|---|
| Train | 00-13 | 1,378 | 26,422 |
| Test | 14-24 | 1,053 | 20,411 |

Table 1: Statistics of the WSJ news dataset used for **"global" discrimination** task.

| | Sections | # Doc. | # Pairs | | | |
|---|---|---|---|---|---|---|
| | | | $\mathcal{D}_{w=1}$ | $\mathcal{D}_{w=2}$ | $\mathcal{D}_{w=3}$ | $\mathcal{D}_{w=1,2,3}$ |
| Train | 00-13 | 748 | 7,890 | 12,280 | 12,440 | 32,610 |
| Test | 14-24 | 618 | 6,568 | 9,936 | 9,906 | 26,410 |

Table 2: Statistics on the WSJ news dataset used for **"local" discrimination** task. The $w$ denotes the number of permuted local windows in a document.

We first set $w$ as the number of local windows that we want to permute in a document. Based on this, we create four datasets for our local discrimination task: $\mathcal{D}_{w=1}$, $\mathcal{D}_{w=2}$, $\mathcal{D}_{w=3}$ and $\mathcal{D}_{w=1,2,3}$. $\mathcal{D}_{w=1}$ contains the documents, where only one randomly selected window is permuted. Similarly, $\mathcal{D}_{w=2}$ contains the documents, where two randomly selected windows are permuted. $\mathcal{D}_{w=3}$ is similarly created for 3 windows. $\mathcal{D}_{w=1,2,3}$ denotes the concatenated datasets. The number of negative documents for each article was restricted **not to exceed 20 samples**. Additionally, we exclude the cases of the overlap between windows. In other words, the sentences are allowed to be permuted only inside their respective window.

We randomly select 10% of the training set for development purposes. Table 2 summarizes the datasets. Consequently, the training and the test dataset for $\mathcal{D}_{w=1,2,3}$ consists of 32,610 and 26,410 pairs, respectively.

## 5 Experiments

This section presents details of our experiment procedures and results.

### 5.1 Models Compared

We compare our proposed unified coherence model with several existing models. Some of the baselines that are not publicly available were re-implemented during experiments, otherwise we conducted experiments with publicly available codes, and the rest of the reported results are from their original papers. In the following sections, we present brief descriptions of the existing models.

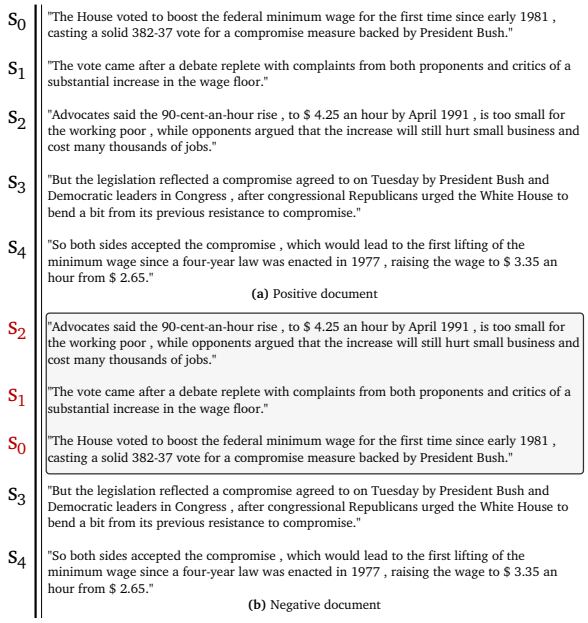


Figure 3: Sample data in the local permutation data set. (a) is the positive document, WSJ0098, and (b) is the negative version of the positive document, which is locally sentence-order permuted.

**Distributed Sentence Model (L&H).** This is the neural model proposed by Li and Hovy (2014). Similar to our local model, it extracts local coherence features for small windows of sentences to compute the coherence score of a document. First, they use a recurrent or a recursive neural network to compute the representation for each sentence in the local window from its words and their pre-trained embeddings. Then the concatenated vector is passed through a non-linear hidden layer, and finally the output layer decides if the window of sentences is a coherent text or not. The main differences between our implementation and the implementation referred in their paper are that we used a bi-LSTM (as opposed to simple RNN) for sentence encoding and trained the network with the Adam optimizer (as opposed to AdaGrad).

**Grid-all nouns & Extended grid (E&C)[2].** Elsner and Charniak (2011b) report significant gains by including all nouns as entities in the original entity grid model as opposed to considering only head nouns. In their extended grid model, they used 9 additional entity-specific features, 4 of which are computed from external corpora.

**Neural Grid & Ext. Neural Grid (N&J)[3].** These are the neural versions of the entity grid models as proposed by (Tien Nguyen and Joty, 2017). They use convolutions over grammatical roles to model entity transitions in the distributed space. In the extended model, they incorporate three entity-specific features.

**Lex. Neural Grid (M&J)[4].** Mohiuddin et al. (2018) improved the neural grid model by lexicalizing the entity transitions. Experiment results for this model were obtained with the optimal setting described in the original paper.

**Global Coherence Model.** This is the global coherence model component in our proposed unified model as described in Section 3.3. The model extracts document-level features through lightweight convolutions. The extracted features are subsequently averaged along the temporal dimension, which is in turn used in a linear layer for coherence scoring. This model used a kernel size of 5 and each document was padded by the size of 3.

### 5.2 Settings of Our Model

We held out 10% of the training documents to form a development set (DEV) on which we tune the hyper-parameters of our models. In our experiments, we use both `word2vec` (Mikolov et al., 2013) and `ELMo` (Peters et al., 2018) for the distributed representations of the words. Unlike `word2vec`, `ELMo` is capable of capturing both subword information and contextual clues. We implemented our models in PyTorch framework on a Linux machine with a single GTX 1080 Ti GPU.

During training, for optimization we use Adam optimizer (Kingma and Ba, 2015) with $L_2$ regularization (0.00001 regularization parameter). We trained the model up to 25 epochs to make the models' performance converge. To search for optimal parameters, we conducted various experiments while varying the hyper-parameters. Precisely, minibatch size in {5, 10, 20, 25}, sentence embedding size in {128, 256}, lightweight convolution kernel size in {3, 5, 7, 9}, bilinear output dimension size in {32, 64} are investigated. We present the optimal hyper-parameter values in the *supplementary document*. The results are reported by averaging over five different runs of the model with different seeds for statistical stability.

[2] https://bitbucket.org/melsner/browncoherence/src

[3] https://github.com/datienguyen/cnn_coherence

[4] https://github.com/taasnim/conv-coherence

| Model | Emb. | $\mathcal{D}_{w=1,2,3}$ | $\mathcal{D}_{w=1}$ | $\mathcal{D}_{w=2}$ | $\mathcal{D}_{w=3}$ |
|---|---|---|---|---|---|
| Lex. Neural Grid (M&J)* | word2vec | 60.27 | 56.11 | 60.23 | 62.23 |
| Lex. Neural Grid (M&J) | word2vec | 55.01 | 53.81 | 55.37 | 56.16 |
| Dist. sentence (L&H) | word2vec | 6.76 | 4.28 | 6.82 | 9.25 |
| Our Global Model | word2vec | 57.24 | 53.35 | 56.58 | 59.67 |
| Our Local Model | word2vec | 73.23 | 66.21 | 73.16 | 77.93 |
| Our Local Model | ELMo | 74.12 | 65.82 | 73.54 | 78.16 |
| Our Full Model | word2vec | 75.37 | **67.29** | 75.58 | 80.21 |
| Our Full Model | ELMo | **77.07** | 64.38 | **76.12** | **81.23** |

Table 3: Results in accuracy on the **Local Discrimination task**. * is a pre-trained model on the global discrimination task.

| | Model | Emb. | Standard | Inverse |
|---|---|---|---|---|
| I | Dist. sentence (L&H) | word2vec | 17.39 | 18.11 |
| II | Grid-all nouns (E&C) | - | 81.60 | 75.78 |
| | Ext. Grid (E&C) | - | 84.95 | 80.34 |
| III | Neural Grid (N&J) | Random | 84.36 | 83.94 |
| | Ext. Neural Grid (N&J) | Random | 85.93 | 83.00 |
| IV | Lex. Neural Grid (M&J) | Random | 87.03 | 86.88 |
| | Lex. Neural Grid (M&J)(paper) | word2vec | 88.56 | 88.23 |
| | Lex. Neural Grid (M&J)(code) | word2vec | 88.51 | 88.13 |
| V | Our Best Model | word2vec | 90.42 | 95.27 |
| | Our Best Model | ELMo | **93.19** | **96.78** |

Table 4: Results in accuracy on the **Global Discrimination** task.

### 5.3 Results on Local Discrimination

Table 3 shows the results in accuracy on the **"local" discrimination** task. From the table, we see that existing models including our global model perform poorly compared to our proposed local models. They are likely to fail in distinguishing the text segments that are locally coherent and penalize them unfairly. One of the possible explanations of this phenomenon can be found in the nature of the global model. These models (except L&H) are designed to make a decision at a global level, thus they are likely to penalize locally coherent segments of a text. This observation is further bolstered by the performance of our local coherence models, which show higher sensitivity in discriminating locally coherent texts and achieve significantly higher accuracy compared to the baseline models and our global model.

Another aspect to notice here is that the performance of all the models become gradually better with the increase in the number of permutation windows in the dataset. This is not surprising because in the datasets with a lower number of permutation windows, the difference between a positive and a negative document is very subtle. For example, in $\mathcal{D}_{w=1}$ dataset, positive and negative documents differ only in a small window position. Another interesting observation regarding the entity-grid based neural models is that the model pretrained on the global discrimination task performs better than the ones trained on the specific tasks. From the table, we observe that our full model with ELMo word embeddings achieves the highest accuracies on the datasets $\mathcal{D}_{w=1,2,3}$, $\mathcal{D}_{w=2}$, and $\mathcal{D}_{w=3}$ , while on the $\mathcal{D}_{w=1}$ dataset, our full model with the pretrained word2vec embeddings performs the best. The reason could be that with more generalized contextual embeddings, our model losses the discrimination capability for small changes in the document.

### 5.4 Results on Global Discrimination

Table 4 presents the results in accuracy on the two **"global" discrimination** tasks – the *Standard* and the *Inverse* order discrimination. The reported results of the entity-grid models are from the original papers. 'Lex. Neural Grid (M&J)(code)' refers to the results achieved by running the code released by Mohiuddin et al. (2018) on our machine.

From the table, we see that our unified neural coherence model outperforms the existing models by a good margin. In this dataset, our best model with the word2vec embeddings achieves 90.42% and 95.27%, on *Standard* and *Inverse* order discrimination tasks, respectively. We achieve the best results with our proposed model by using the ELMo word embeddings, where we get 93.19% and 96.78% accuracies on *Standard* and *Inverse* order discrimination tasks, respectively.

### 5.5 Ablation Study

To investigate the impact of different components in our proposed model, we conducted two sets of ablation study on the local and global discrimination tasks. Specifically, we want to see: (*i*) the impact of our global model component, and (*ii*) the impact of the language model (LM) loss.

**Local Discrimination.** In the local discrimination task, we first compare the performance of the proposed model without the LM loss. As shown in the first block of Table 5, addition of the global model to the local model degrades the performance on the $\mathcal{D}_{w=1}$ dataset by 1.17% and 1.21% for word2vec and ELMo embeddings, respectively. While for the other datasets, we see improvements in performance for the addition of the global model. However, in the presence of the LM loss (second block in Table 5), the addition of the global model improves the performance across

| Model | Emb. | $\mathcal{D}_{w=1,2,3}$ | $\mathcal{D}_{w=1}$ | $\mathcal{D}_{w=2}$ | $\mathcal{D}_{w=3}$ |
|---|---|---|---|---|---|
| **Without LM Loss** | | | | | |
| Local Model | word2vec | 73.23 | 66.21 | 73.16 | 77.93 |
| Local Model | ELMo | 74.12 | 65.82 | 73.54 | 78.16 |
| Local + Global Model | word2vec | 74.69 | 65.04 | 75.27 | 79.69 |
| Local + Global Model | ELMo | 76.01 | 64.61 | 75.22 | 79.37 |
| **With LM Loss** | | | | | |
| Local Model | word2vec | 75.03 | 66.95 | 75.04 | 80.07 |
| Local Model | ELMo | 75.20 | 64.22 | 75.93 | 80.57 |
| Local + Global Model | word2vec | 75.37 | **67.29** | 75.58 | 80.21 |
| Local + Global Model | ELMo | **77.07** | 64.38 | **76.12** | **81.23** |

Table 5: Ablation study of different model components on the **Local Discrimination task**.

all the datasets on the local discrimination task.

On the other hand, the addition of the LM loss to our model (with/without global model) increases the accuracy in most of the datasets and embeddings. Exception is the ELMo embeddings on $\mathcal{D}_{w=1}$ dataset, where the overall performance drops by 1.60% and 0.23% for the local model with and without the global model, respectively.

Another interesting observation on $\mathcal{D}_{w=1}$ dataset is that in all the cases word2vec embeddings outperforms ELMo. This unusual behavior of $\mathcal{D}_{w=1}$ dataset to the rests is not surprising because it is the hardest dataset where the difference between the positive and the negative document is subtle. In this case, generally flexible and simple models outperform complex ones.

For the performance degradation of the global model in $\mathcal{D}_{w=1}$ case, we assume that in some texts, the global model fails to capture the significant feature from the locally negative region. Subsequently, the global feature is added into the score calculation at every local window, so the overall influence of the global model becomes bigger than that of the local model in the decision making.

**Global Discrimination.** We also studied the impact of our global model and the LM loss in the global discrimination task. As shown in Table 6, the addition of the global model and LM loss to the local model improves performance on the standard discrimination task by 1.34%.

However, the addition of global model impacts negatively on the inverse order task and degrades accuracies by 2.42% and 1.28% in the presence and absence of LM loss respectively. We suspect that the global model is adding noise because of the pooling operation, which throws away the spatial relation between sentences and provides the global information that is invariant to the sentence-order. But in this task, order information is crucial.

| Model | Emb. | Standard | Inverse |
|---|---|---|---|
| Our Local Model | word2vec | 88.93 | 94.72 |
| + LM Loss | word2vec | 89.92 | **96.24** |
| + Global Model | word2vec | 89.53 | 93.44 |
| + Global Model + LM Loss (**Our Full Model**) | word2vec | **90.27** | 93.82 |

Table 6: Ablation study of different model components on the **Global Discrimination task**.

In the inverse order task, we get the best performance by adding the LM loss to our local model.

## 6 Conclusion

In this paper, we proposed a unified coherence model. The proposed model incorporates a local coherence model and a global coherence model to capture the sentence grammar (intentional structure), discourse relations, attention and topic structures in a single framework. The unified coherence model shows state of the art results on the standard coherence assessment tasks: the inverse-order and the global discrimination tasks. Also, our evaluation of the local discrimination task demonstrates the effectiveness of the unified coherence model in assessing global and local coherence of texts.

## Acknowledgments

We would like to thank the anonymous reviewers for their comments. Shafiq Joty would like to thank the funding support from his Start-up Grant (M4082038.020). Also, this work is partly supported by SIMTech-NTU Joint Laboratory on Complex Systems.

## References

Regina Barzilay and Mirella Lapata. 2005. Modeling local coherence: An entity-based approach. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 141–148, Ann Arbor, Michigan. Association for Computational Linguistics.

Regina Barzilay and Mirella Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34.

Regina Barzilay and Lillian Lee. 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *HLT-NAACL 2004: Main Proceedings*, pages 113–120, Boston, Massachusetts, USA. Association for Computational Linguistics.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1993. Signature ver-

ification using a "siamese" time delay neural network. In *Proceedings of the 6th International Conference on Neural Information Processing Systems*, NIPS'93, pages 737–744, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

François Chollet. 2016. Xception: Deep learning with depthwise separable convolutions. *CoRR*, abs/1610.02357.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Micha Elsner, Joseph Austerweil, and Eugene Charniak. 2007. A unified local and global model for discourse coherence. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 436–443, Rochester, New York. Association for Computational Linguistics.

Micha Elsner and Eugene Charniak. 2011a. Disentangling chat with local coherence models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 1179–1189, Stroudsburg, PA, USA. Association for Computational Linguistics.

Micha Elsner and Eugene Charniak. 2011b. Extending the entity grid with entity-specific features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11, pages 125–129, Portland, Oregon. Association for Computational Linguistics.

Vanessa Wei Feng and Graeme Hirst. 2012. Extending the entity-based coherence model with multiple ranks. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, pages 315–324, Avignon, France. Association for Computational Linguistics.

Vanessa Wei Feng, Ziheng Lin, and Graeme Hirst. 2014. The impact of deep hierarchical discourse structures in the evaluation of text coherence. In *COLING*.

Barbara J. Grosz and Candace L. Sidner. 1986. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–204.

Barbara J. Grosz, Scott Weinstein, and Aravind K. Joshi. 1995. Centering: A framework for modeling the local coherence of discourse. *Comput. Linguist.*, 21(2):203–225.

Amit Gruber, Yair Weiss, and Michal Rosen-Zvi. 2007. Hidden topic markov models. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, volume 2 of *Proceedings of Machine Learning Research*, pages 163–170, San Juan, Puerto Rico. PMLR.

Camille Guinaudeau and Michael Strube. 2013. Graph-based local coherence modeling. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers*, pages 93–103.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Jiwei Li and Eduard Hovy. 2014. A model of coherence based on distributed sentence representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2039–2048, Doha, Qatar. Association for Computational Linguistics.

Jiwei Li and Dan Jurafsky. 2017. Neural net models of open-domain discourse coherence. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 198–209, Copenhagen, Denmark. Association for Computational Linguistics.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2011. Automatically evaluating text coherence using discourse relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 997–1006, Portland, Oregon. Association for Computational Linguistics.

Annie Louis and Ani Nenkova. 2012. A coherence model based on syntactic patterns. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1157–1168, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The Ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 285–294, Prague, Czech Republic. Association for Computational Linguistics.

W. Mann and S. Thompson. 1988. Rhetorical Structure Theory: Toward a Functional Theory of Text Organization. *Text*, 8(3):243–281.

Mohsen Mesgar and Michael Strube. 2018. A neural local coherence model for text quality assessment. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4328–4339, Brussels, Belgium. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Muhammad Tasnim Mohiuddin, Shafiq Joty, and Dat Tien Nguyen. 2018. Coherence modeling of asynchronous conversations: A neural entity grid approach. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 558–568, Melbourne, Australia. Association for Computational Linguistics.

Dat Nguyen and Shafiq Joty. 2017. A neural local coherence model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1320–1330. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Emily Pitler and Ani Nenkova. 2008. Revisiting readability: A unified framework for predicting text quality. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 186–195, Honolulu, Hawaii. Association for Computational Linguistics.

Radu Soricut and Daniel Marcu. 2006. Discourse generation using utility-trained coherence models. In *Proceedings of the COLING/ACL on Main Conference Poster Sessions*, COLING-ACL '06, pages 803–810, Sydney, Australia. Association for Computational Linguistics.

Dat Tien Nguyen and Shafiq Joty. 2017. A neural local coherence model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1320–1330, Vancouver, Canada. Association for Computational Linguistics.

Felix Wu, Angela Fan, Alexei Baevski, Yann Dauphin, and Michael Auli. 2019. Pay less attention with lightweight and dynamic convolutions. In *International Conference on Learning Representations*.

# Topic-Guided Coherence Modeling for Sentence Ordering by Preserving Global and Local Information

**Byungkook Oh, Seungmin Seo, Cheolheon Shin, Eunju Jo, Kyong-Ho Lee**
Department of Computer Science, Yonsei University
Seoul, South Korea
{bkoh, smseo, chshin, ejjo}@icl.yonsei.ac.kr
khlee89@yonsei.ac.kr

## Abstract

We propose a novel ***topic-guided coherence modeling (TGCM)*** for sentence ordering. Our attention based pointer decoder directly utilize sentence vectors in a permutation-invariant manner, without being compressed into a single fixed-length vector as the paragraph representation. Thus, TGCM can improve global dependencies among sentences and preserve relatively informative paragraph-level semantics. Moreover, to predict the next sentence, we capture topic-enhanced sentence-pair interactions between the current predicted sentence and each next-sentence candidate. With the coherent topical context matching, we promote local dependencies that help identify the tight semantic connections for sentence ordering. The experimental results show that TGCM outperforms state-of-the-art models from various perspectives.

## 1 Introduction

Modeling the coherence among sentences to compute their gold order is one of the fundamental tasks in Natural Language Processing (NLP) with many applications such as document modeling (Narayan et al., 2018a), extractive document summarization (Jadhav and Rajan, 2018; Nallapati et al., 2017), question answering (Yu et al., 2018; Liu et al., 2017), conversational analysis (Zeng et al., 2018), automated text generation (Guo et al., 2018), and image captioning (Anderson et al., 2018). The coherence helps readers to improve reading comprehension and better understand the intent of a document. Sentence ordering is a set-to-sequence problem, which aims to identify the correct order of a sentence set. To do this, various studies on sentence ordering typically combine the coherent features extracted from sentences.

In recent years, most of the traditional approaches to sentence ordering are designed based on a pairwise strategy (Chen et al., 2016; Agrawal et al., 2016; Li and Jurafsky, 2016). The sentence pair ordering (SPO) models determine the relative order within a sentence pair via neural networks based semantic matching, which computes the relevance between the two sentence vectors. However, such models cause combinatorial optimization problems because search algorithms (e.g., beam search) are necessary to find the most optimal permutation. Since the SPO models only focus on the sentence-pair interactions (i.e., *local dependency*), they have trouble in capturing the interactions among three or more input sentences (i.e., *global dependency*) in an entire paragraph.

More recently, state-of-the-art models aim to put randomly sorted sentences into a coherent paragraph with the correct order so that the whole sentences have the highest coherence probability (Vinyals et al., 2015a; Gong et al., 2016; Logeswaran et al., 2018; Cui et al., 2018). Unlike the SPO models, these models can perform the sentence set ordering (SSO) task for an entire paragraph based on the pointer network (Vinyals et al., 2015b). Hierarchical RNN networks consisting of sentence and paragraph encoders take unordered sentences as input. They build a paragraph-level vector representation, which represents a semantic summary of the input sentences. Then, a pointer network based decoder fetches the paragraph vector and iteratively outputs sentences in the correct order. At this time, since the output sentences are taken from the input sentences, this can solve the combinatorial optimization problems, which the SPO models suffer from. Also, the SSO models can capture the global dependency among input sentences via the paragraph vector.

Despite the successes of the SSO models, there still exist severe limitations as shown in Figure 1.

- **[L1]** The conventional pointer decoders for the



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2273–2283, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

SSO task only utilize the last hidden state at the end of paragraph encoders. The encoder always pushes sentence information into a single fixed-length paragraph vector, no matter how many sentences are in the paragraph. Thus, they may struggle with the bottleneck problem where important information between the encoder and the decoder is shrunk. Especially, the more the number of sentences, the more difficult to preserve the **global information** of the paragraph.

- **[L2]** The attention layer repeatedly decides the next sentence from sentence-pair interactions between the current predicted sentence and each paragraph-independent candidate $\{s_i\}_1^5$. Thus, they do not elaborately utilize the context of previously predicted sentences $\{s_1, s_2, s_3\}$ and the context of $s_4$ conditioned on the context of the paragraph (i.e., the shuffled sentences $\{s_2, s_4, s_5, s_1, s_3\}$), where the coherent contexts indicate the **local information** that helps determine the tight $s_3$-$s_4$ interactions.

To address the above limitations, this paper proposes a novel **T**opic-**G**uided **C**oherence **M**odeling **(TGCM)** for sentence ordering by capturing local and global dependencies among sentences. Specifically, TGCM is composed of two major components: topic-sensitive sentence encoder and attentive pointer decoder.

To complement the structural limitations of existing RNN-based pointer decoders, which sequentially decode one paragraph vector, our attentive pointer decoder relies entirely on the attention mechanism (Vaswani et al., 2017) without any recurrent units or convolutions. Since our decoder directly receives the set of sentence vectors regardless of their input order via attention, our encoder is free from the constraint that all sentences must be compressed into a single fixed-length vector as a paragraph representation. As a result, the preservation of **global information** improves the global dependencies among sentences and provides a relatively informative paragraph-level semantics, which can deal with the bottleneck problem. Moreover, TGCM can better preserve the fine-grained word/sentence-level semantics from the encoder to the decoder.

Instead of paragraph-independent next-sentence candidates, the topic-sensitive sentence encoder enriches each candidate sentence with its topical context conditioned on the paragraph context via

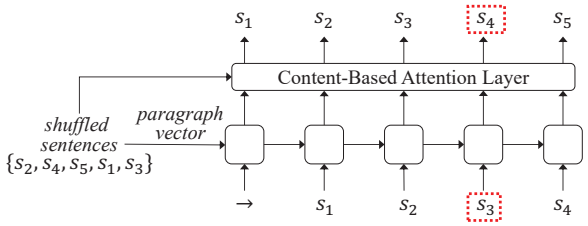


Figure 1: Pointer decoder for sentence ordering. Based on the paragraph vector encoded from the shuffled sentences, the attention layer selects the next sentence $s_4$ following $s_3$ from sentence-pair interactions between the previously predicted $s_3$ and each candidate $\{s_i\}_1^5$.

topic modeling based on Latent Dirichlet Allocation (LDA) (Blei et al., 2003). For each position of the decoder, TGCM also incorporates the topical context flow of previously predicted sentence sequence into the current predicted sentence. It then predicts the next sentence from topic-enhanced sentence-pair interactions with the coherent topical context as local information. Consequently, the preservation of **local information** promotes the local dependencies between the current predicted sentence and each next-sentence candidate, resulting in identifying the strongest semantic connection for ordering a shuffled paragraph.

## 2 Related Work

The key idea of sentence modeling is to embed each sentence into continuous vector spaces by combining word vectors with recurrent neural networks (RNNs) (Mikolov et al., 2010) to capture long-term dependencies between words and convolution neural networks (CNNs) (Kim, 2014) to capture important local invariance context.

**Sentence Pair Modeling**: To identify the relationship between given two sentences, sentence pair modeling learns a semantic matching function based on neural networks, which extracts their task-specific features. As a famous example, question answering measures relevance between question-answer pairs with a matching function and ranks candidate answer sentences (Severyn and Moschitti, 2016; Liu and Huang, 2016). Recent models further measure the semantic similarity elaborately by using multi-attention mechanism (Tan et al., 2018; Tay et al., 2018) and a word-level similarity matrix (Pang et al., 2016; Shen et al., 2017).

Another example is the SPO task. Chen et al. (2016) compute a similarity score for each sen-

tence pair independently. After pairwisely scoring, they use a beam search algorithm to find an optimal order for input sentences. Agrawal et al. (2016) develop a pairwise scoring model to organize unordered image-caption pairs. However, such models have the combinatorial optimization problem of finding the most optimal permutation. Moreover, they cannot capture the paragraph-level contextual information, which indicates the global dependency of an entire paragraph.

**Sentence Set Modeling**: To solve the above issues, the pointer network (Vinyals et al., 2015b) is proposed based on the sequence-to-sequence model (Sutskever et al., 2014). The pointer network consists of encoding input tokens to a summary vector, decoding next token vectors iteratively via content-based attention over input token vectors, and producing the output token sequence from the output token vectors. Inspired by the pointer network, state-of-the-art SSO models (Vinyals et al., 2015a; Gong et al., 2016; Logeswaran et al., 2018) usually employ hierarchical RNN-based encoders to produce a paragraph vector from unordered sentences. Then, the pointer network based decoders predict the correct sentence sequence. However, the paragraph vector depends on the permutation of input sentences.

To address the issue, ATTOrderNet (Cui et al., 2018) employs self-attention at the encoder to capture global dependencies regardless of an input sentence order. However, similar to traditional models, they also compress sentence vectors into a single fixed-length vector via average pooling. While Our TGCM is also permutation-agnostic to input sentences, we feed the sentence vector set directly into the attentive pointer decoder and capture the coherent topical context in sentence-pair interactions via topic modeling. Thus, unlike ATTOrderNet, TGCM can simultaneously improve both local and global dependencies.

**Topic-Aware Sentence Modeling**: The commonly used topic model is based on LDA, which extracts the latent topic vectors (i.e., topic distributions) of words, sentences, and paragraphs from a training corpus. Given sentences in a paragraph, their topic latent vectors help in capturing global topical context for the paragraph and local topical context for each sentence. Therefore, topic modeling is used in various research fields requiring sentence modeling (Narayan et al., 2018b; Dieng et al., 2016; Gong et al., 2018). LTMF (Jin et al., 2018) is a context-aware recommender system, which combines LSTMs and topic modeling before applying matrix factorization. They extract the global context information related to words in a user review via topic distributions.

## 3 The Proposed Model

In this section, we present TGCM as a novel topic-guided coherence modeling for SSO. TGCM can address the above-mentioned limitations **[L1]** and **[L2]** simultaneously. We first build a topic-distribution generating function via topic modeling. Then, we describe two major components: topic-sensitive sentence encoder and attentive pointer decoder. The encoder leverages the topic distributions of a paragraph and its sentences. Then, the decoder directly utilizes them in a permutation-agnostic manner and determines the correct order of randomly sorted sentences.

### 3.1 Problem Definition

The primary goal of sentence set ordering is to put an unordered set of sentences into a coherent paragraph in the correct order. Specifically, given a paragraph $p$, the correct sentence sequence and its order are denoted by $S_p = [s_1, s_2, ..., s_{|p|}]$ and $O_p = [o_{s_1}, o_{s_2}, ..., o_{s_{|p|}}]$, respectively, where $|p|$ denotes the number of sentences in $p$.

Given a shuffled sentence sequence, our TGCM outputs a sentence sequence $\hat{S}_p$ whose order is denoted by $\hat{O}_p$. The objective of coherence modeling is to make the coherence probability for the predicted order $\hat{O}_p$ approximate to the coherence probability for the correct order $O_p$ as follows:

$$P(O_p|p) \approx P(\hat{O}_p|p), \tag{1}$$

where $P(O_p|p)$ and $P(\hat{O}_p|p)$ denote the coherence probabilities for $O_p$ and $\hat{O}_p$, respectively.

### 3.2 Topic Latent Vectors

Given the number of hidden topics, which is denoted by $d_t$, the preprocessing of TGCM trains a topic model on a given corpus as shown in Figure 2. The topic model builds a generating function $topicDistribution(\cdot)$ based on the probability distribution over topics for each word. At test time, $topicDistribution(\cdot)$ infers the $d_t$-dimensional topic latent vectors $\mathbf{t}_{doc}$ (i.e., topic distribution) of a new, unseen document $doc$ such as sentences and paragraphs.

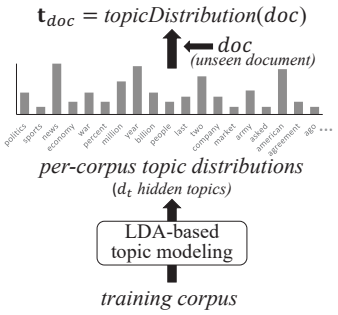


Figure 2: Topic modeling on a corpus as preprocessing of TGCM. $topicDistribution(\cdot)$ returns LDA-based features, a topic latent vector $\mathbf{t}_{doc}$ of a given document.

To do this, we utilize LDA (Blei et al., 2003), which is the most simple and popular algorithm for topic modeling. The topic latent vectors of sentences and paragraphs indicate the sentence-level (local) and the paragraph-level (global) topical context via hidden topics, respectively.

### 3.3 Topic-Sensitive Sentence Encoder

**Attention-Based bi-LSTM Layer.** We leverage an extended version of LSTMs (Hochreiter and Schmidhuber, 1997) as our base model in Figure 3, which overcomes the gradient vanishing of existing RNNs. Particularly, attention-based bidirectional LSTMs (Att-BLSTMs) (Zhou et al., 2016) are mostly used in sentence modeling.

In this layer, we aim to produce the sentence vector $\mathbf{s}$ from a given sentence $s$ consisting of a word sequence $[w_1, w_2, ..., w_n]$. Each word $w_i$ is encoded into a word embedding $\mathbf{w}_i \in \mathbb{R}^{d_w}$. Att-BLSTMs consist of two sub-networks with forward and backward LSTMs, which take the word embeddings $[\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_n]$ and output sequences of forward and backward hidden states $[\overrightarrow{h}_1, \overrightarrow{h}_2, ..., \overrightarrow{h}_n]$ and $[\overleftarrow{h}_1, \overleftarrow{h}_2, ..., \overleftarrow{h}_n]$, respectively as follows:

$$\begin{aligned} \overrightarrow{h}_t &= \overrightarrow{LSTM}(M_t, \overrightarrow{h}_{t-1}), \\ \overleftarrow{h}_t &= \overleftarrow{LSTM}(M_t, \overleftarrow{h}_{t+1}), \end{aligned} \tag{2}$$

where $M_t$ is a set of learnable parameters. Then, we combine the forward and backward hidden states by an element-wise sum as follows:

$$\mathbf{h}_i = [\overrightarrow{h}_i \oplus \overleftarrow{h}_i]. \tag{3}$$

Thus, the hidden state vectors $[\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_n] \in \mathbb{R}^{d_w \times n}$ are obtained from previous LSTM layers and fed into an attention layer as follows:

$$\begin{aligned} \alpha &= softmax(\mathbf{m}_1^T tanh([\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_n])), \\ \mathbf{s} &= [\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_n]\alpha^T, \end{aligned} \tag{4}$$

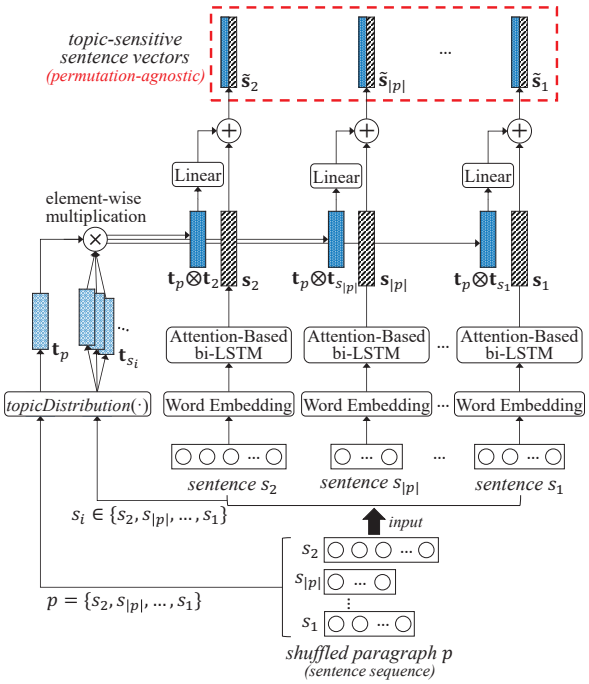


Figure 3: Architecture of topic-sensitive sentence encoder. The encoder takes a shuffled paragraph consisting of $\{s_2, s_{|p|}, ..., s_1\}$ and outputs $\{\tilde{\mathbf{s}}_2, \tilde{\mathbf{s}}_{|p|}, ..., \tilde{\mathbf{s}}_1\}$ from their vectors and topic latent vectors of the paragraph and sentences. The outputs are directly utilized in a decoder without being compressed into a vector.

where $\mathbf{m}_1 \in \mathbb{R}^{d_w}$ and $\alpha \in \mathbb{R}^n$ denote a weight vector and resulting attention weights, respectively. $\mathbf{s} \in \mathbb{R}^{d_w}$ is the final sentence vector of $s$, which is computed by a weighted sum between the hidden state vectors and the attention weights.

**Topic-Sensitive Sentence Vectors.** As shown in Figure 3, our topic-sensitive sentence encoder takes a shuffled paragraph $p$ as input, which consists of unordered sentences $\{s_i\}_1^{|p|}$. We create sentence vectors $\mathbf{s}_i$ for all sentences $s_i$ with the attention-based bi-LSTMs. In addition, topic latent vectors $\mathbf{t}_p \in \mathbb{R}^{d_t}$ and $\mathbf{t}_{s_i} \in \mathbb{R}^{d_t}$ are generated by $topicDistribution(\cdot)$ for $p$ and $s_i$, where $d_t$ denotes the number of latent topics.

During the encoding of TGCM, we multiply $\mathbf{t}_{s_i}$ and $\mathbf{t}_p$ for each sentence $s_i$ to weight the coherent latent topics that appear simultaneously in both the sentence and the paragraph. The topic latent vector $\mathbf{t}_{s_i}$ captures how topical a sentence $s_i$ is in itself (local context), whereas the topic latent vector $\mathbf{t}_p$ represents the overall theme of a paragraph $p$ (global context). Thus, the encoder can extract the paragraph-level context relating to each sentence by enriching the context of the sentence with its

topical relevance to the paragraph as follows:

$$(\mathbf{t}_p \otimes \mathbf{t}_{s_i}) \in \mathbb{R}^{d_t}; \quad s_i \in \{s_1, s_2, ..., s_{|p|}\}, \quad (5)$$

where the operation $\otimes$ indicates an element-wise multiplication. To combine $\mathbf{t}_p \otimes \mathbf{t}_{s_i}$ with the sentence vectors $\mathbf{s}_i$, we apply two linear transformations with a ReLU activation to each $\mathbf{t}_p \otimes \mathbf{t}_{s_i}$ separately. Then, the transformed vectors are added with the corresponding sentence vectors as follows:

$$\tilde{\mathbf{s}}_i = max(0, (\mathbf{t}_p \otimes \mathbf{t}_{s_i})^\intercal W_1 + b_1) W_2 + b_2 + \mathbf{s}_i, \quad (6)$$

where $\tilde{\mathbf{s}}_i \in \mathbb{R}^{d_w}$ denotes a topic-sensitive sentence vector. Thus, the global topical context relating to the local topical context of each sentence is incorporated into the corresponding sentence vector. This helps our pointer decoder for next-sentence predictions to guide the semantic connection between a predicted sentence and its next sentence by considering coherent topical context flow.

### 3.4 Attentive Pointer Decoder

As shown in Figure 4, we build the attentive pointer decoder based on the decoder of Transformer (Zhou et al., 2016) and the pointer network (Narayan et al., 2018b). The pointer network determines next token vectors iteratively via content-based attention over input token vectors. For coherent topical context matching, our decoder is mainly composed of a stack of $n$ attention modules relying entirely on attention which have been attracted as a promising technique in many sequence-based tasks (Bahdanau et al., 2014).

Existing SSO models utilized RNN-based decoders cannot pass encoded sentence vectors because their decoders sequentially decode one paragraph vector. Attention mechanisms allow each sentence in a different position to link other sentences regardless of the order and number of input sentences. Thus, our attention based pointer decoder directly utilize the topic-sensitive sentence vectors $\{\tilde{\mathbf{s}}_2, \tilde{\mathbf{s}}_{|p|}, \tilde{\mathbf{s}}_3, ..., \tilde{\mathbf{s}}_1\}$ in a permutation-agnostic manner (global information). This allows our model to avoid bottleneck problems which dilute word/sentence semantics between encoder and decoder and fully utilize the semantics for sentence ordering.

Moreover, Our attentive pointer decoder can capture topic-enhanced sentence-pair interactions between the current predicted sentence and

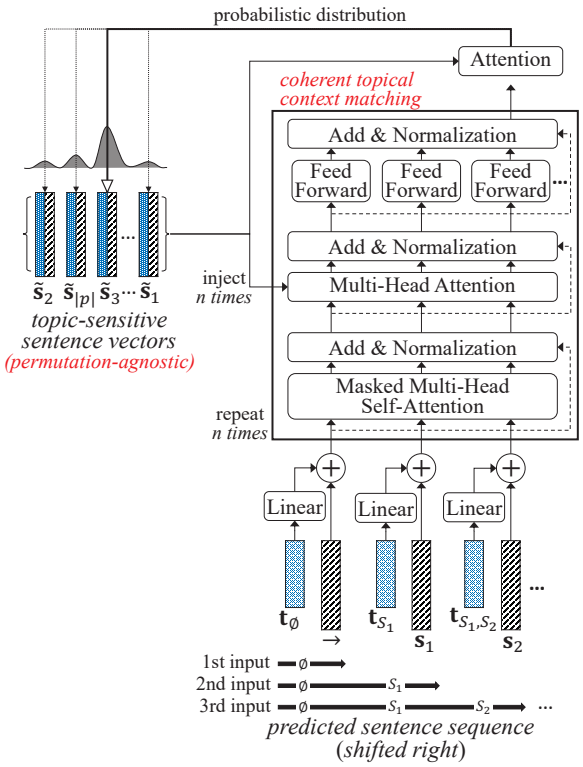


Figure 4: Architecture of attentive pointer decoder. The decoder fetches permutation-agnostic topic-sensitive sentence vectors $\{\tilde{\mathbf{s}}_2, \tilde{\mathbf{s}}_{|p|}, \tilde{\mathbf{s}}_3, ..., \tilde{\mathbf{s}}_1\}$ from the encoder. For coherent topical context modeling, at the 3rd step, a stack of $n$ attention modules position-wisely takes $\{\rightarrow, \mathbf{s}_1, \mathbf{s}_2\}$ and newly created topic latent vectors $\{\mathbf{t}_\varnothing, \mathbf{t}_{s_1}, \mathbf{t}_{s_1,s_2}\}$, where $\mathbf{t}_\varnothing$ denotes a zero vector. The topic vectors are generated by incrementally including previously predicted $\{s_1, s_2\}$, The attention layer outputs a probabilistic distribution over encoder outputs $\{\tilde{\mathbf{s}}_2, \tilde{\mathbf{s}}_{|p|}, \tilde{\mathbf{s}}_3, ..., \tilde{\mathbf{s}}_1\}$ and select the next sentence $s_3$.

encoder outputs $\{\tilde{\mathbf{s}}_2, \tilde{\mathbf{s}}_{|p|}, \tilde{\mathbf{s}}_3, ..., \tilde{\mathbf{s}}_1\}$) as next-sentence candidates. With the coherent topical context (local information) matching, we promote local dependencies for identifying the tight semantic connections for sentence ordering.

**Coherent Topical Context Matching.** Our attentive pointer decoder employs a stack of attention modules identical with the decoder of Transformer (Zhou et al., 2016) for the coherent topical context matching. Given query $\mathbf{Q} \in \mathbb{R}^{n \times d_m}$, key $\mathbf{K} \in \mathbb{R}^{n \times d_m}$, and value $\mathbf{V} \in \mathbb{R}^{n \times d_m}$, the attention mechanism computes the output matrix $\mathbf{Out}_{att} \in \mathbb{R}^{n \times d_m}$ obtained from the value matrix $\mathbf{V}$ with an attention weight $\alpha$ as follows:

$$\alpha = softmax(\frac{\mathbf{Q}\mathbf{K}^\intercal}{\sqrt{d_k}}), \quad \mathbf{Out}_{att} = \alpha \mathbf{V}, \quad (7)$$

where $\alpha$ is calculated with the query-key pair by the scaled dot product. Note that self-attention is

the case when all query, key, and value matrices are the same. Masked attention, a variant of attention, masks out all positions after the current position by arbitrarily setting a large value ($-\infty$) in the softmax function.

Firstly, for the coherent topical context matching, we position-wisely feed previously predicted sentence vectors and their newly created topic latent vectors to the masked multi-head self-attention sub-layer. In that, each topic latent vector is generated by $topicDistribution(\cdot)$ with predicted sentences in all positions before the current position. Thus, we incorporate the topical context flow of the previously predicted sentence sequence into the corresponding predicted sentence vector.

After a residual connection (He et al., 2016) and layer normalization (Lei Ba et al., 2016), the resulting vectors are injected as the query matrix $\mathbf{Q}$ into the multi-head attention sub-layer. The sub-layer also takes permutation-agnostic topic-sensitive sentence vectors obtained from our encoder as the key and value matrices $\mathbf{K}$ and $\mathbf{V}$. We follow Transformer for the remaining coherent topical context matching process, including multi-head attention strategy. As a result, with the multiple attention modules, TGCM draws global dependencies among sentences by attending over the topic-sensitive sentence vectors repeatedly.

**Probabilistic Distribution for Ordering.** At the $i$th step, the attention module of the decoder takes previously predicted sentences $\{\tilde{s}_1, ..., \tilde{s}_{i-1}\}$ and $\tilde{s}_0$ for the token "$\rightarrow$" as inputs. After repeating the attention module $n$ times, we then utilize the $d_w$-dimensional output vector $\mathbf{c}_{i-1}$ corresponding to the $(i-1)$th decoder position. As shown in Figure 4, at the 3rd step, the output vector $(\mathbf{c}_3)$ of the 3rd position is fed into the final attention layer as like in conventional pointer decoders.

In the final attention layer, for the output vector $\mathbf{c}_i$ of the attention modules, TGCM produces an output distribution over topic-sensitive sentence vectors $\{\tilde{\mathbf{s}}_1, \tilde{\mathbf{s}}_2, ..., \tilde{\mathbf{s}}_{|p|}\}$ obtained from encoder via content-based attention as follows:

$$\mathbf{u}_i^j = \mathbf{v}_a^\intercal \, tanh(W_a \begin{bmatrix} \tilde{\mathbf{s}}_j \\ \mathbf{c}_i \end{bmatrix}), \ (1 < j < |p|), \\ P(\hat{o}_i | \hat{o}_{<i}, \tilde{\mathbf{s}}_1, \tilde{\mathbf{s}}_2, ..., \tilde{\mathbf{s}}_{|p|}) = softmax(\mathbf{u}_i), \quad (8)$$

where $\hat{o}_i$ denotes the order of the $i$th position. $W_a \in \mathbb{R}^{d \times 2d}$ $\mathbf{v}_a^\intercal \in \mathbb{R}^d$ denote the weight matrix and vector, respectively, which are shared in all positions. Thus, we can select the correct next sentence $s_i$ that yields the highest probability from the output distribution.

Finally, the selected sentence vector $\mathbf{s}_i$ is fed into the decoder along with previously selected sentences $\{\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_{i-1}\}$ and newly created topic latent vectors $\{\mathbf{t}_\varnothing, \mathbf{t}_{s_1}, \mathbf{t}_{s_1,s_2}, ..., \mathbf{t}_{s_1,...,s_i}\}$ as inputs at the $(i+1)$th step.

**Training.** For each correct sentence sequence, we sample 5 shuffled sentence sequences at training phase. Following the typical sentence ordering models, we train parameters of our model to maximize the coherence probability by minimizing the loss function as follows.

$$L = -\frac{1}{|D|} logP(\hat{O}_p | p, \Theta) + \frac{\lambda}{2} ||\Theta||_2^2, \quad (9)$$

where $\Theta$ and $\lambda$ denote a set of trainable parameters and a regularization parameter, respectively.

| | Train | Valid | Test | #Voca. |
|---|---|---|---|---|
| NIPS abstract | 2,248 (5.91) | 409 (5.95) | 402 (5.86) | 16,721 |
| ANN abstract | 8,569 (4.83) | 962 (4.84) | 2626 (4.97) | 34,485 |
| NSF abstract | 96,070 (8.68) | 10,185 (8.73) | 21580 (8.63) | 334,090 |
| arXiv abstract | 884,912 (5.38) | 110,614 (5.39) | 110,615 (5.37) | 64,557 |
| SIND | 40,155 (5) | 4,990 (5) | 5,055 (5) | 30,861 |

Table 1: The number of paragraphs, average sentences (in parentheses), and vocabulary size of 5 datasets.

## 4 Experimental Setup

The parameters of TGCM were tuned by the RMSProp[1] optimizer, which adaptively adjusts the learning rate for each parameter and resolves the problem of Adagrad (Duchi et al., 2011) where the learning rate radically decreases or increases.

All experiments were implemented in Python using TensorFlow (Abadi et al., 2016), which supports the GPU-accelerated deep learning. We also utilized Natural Language Toolkit[2] (NLTK) (Loper and Bird, 2002) for sentence tokenization and data preprocessing, and Gensim[3] for LDA-based topic modeling. The word embeddings were initialized with pre-trained GloVe vectors of dimension $d_w = 200$, trained by GloVe (Pennington et al., 2014). We trained our LDA-based topic model for each dataset with its training, valid, and test corpora.

[1] http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf
[2] https://www.nltk.org/
[3] https://radimrehurek.com/gensim/

### 4.1 Datasets

We first conducted our experiments on commonly used 4 abstract datasets: NIPS, ANN (Wang et al., 2018), NSF, and arXiv (Chen et al., 2016), which contain abstracts of research papers. The abstracts datasets collected from NIPS papers, ACL Anthology Network (AAN) corpus, NSF Research Award abstract, and arXiv website, respectively. Abstracts have the logical consistency of high quality, which helps coherence modeling for sentence ordering. We further considered the Sequential Image Narrative Dataset (SIND) (Huang et al., 2016), which consists of personal multimodal stories consisting of five sentences per story. We did not utilize accident and earthquake datasets (Barzilay and Lapata, 2008) because the datasets are too small and do not provide the validation set. Following the setup in (Logeswaran et al., 2018), we split undivided datasets for training, validation, and test. Table 1 shows the details of the 5 datasets used in our experiments.

### 4.2 Evaluation Metrics

Following the evaluation metrics widely used in previous SPO and SSO models, we adopted three metrics to assess sentence ordering performance: Kendall's tau, perfect match ratio, and positional accuracy.

**Kendall's tau** ($\tau$): Kendall's tau (Lapata, 2003, 2006) measures the ordinal association between two sequences to automatically evaluate coherence modeling as follows:

$$\tau = \frac{1}{|D|} \sum_{i \in |D|} \frac{2 \times inversions(O_{p_i}, \hat{O}_{p_i})}{|p_i|(|p_i|-1)/2}, \quad (10)$$

where $|D|$ and $|p_i|$ denote the total number of paragraphs in a test dataset $D$ and the number of sentences in a paragraph $p_i$, respectively. The function $inversions(O_{p_i}, \hat{O}_{p_i})$ returns the number of sentence-pair interchanges for reconstructing the correct order $O_{p_i}$ from the predicted order $\hat{O}_{p_i}$. The value ranges from -1 to 1. A higher value indicates better performance. The evaluation metric closely correlates with user ratings and reading times, which are related with the readability.

**Perfect Match Ratio** (PMR in %): The perfect match ratio is the ratio of exactly matching orders across all predicted paragraphs as follows:

$$\text{PMR} = \frac{1}{|D|} \sum_{i \in |D|} \mathbb{I}(O_{p_i} = \hat{O}_{p_i}), \quad (11)$$

where $\mathbb{I}(O_{p_i} = \hat{O}_{p_i})$ denotes the indicator function, which returns 1 if the correct sentence order $O_{p_i}$ and the predicted sentence order $\hat{O}_{p_i}$ are identical and 0 otherwise.

**Positional Accuracy** (PAcc in %): The positional accuracy is defined as the ratio of the matched sentence-level absolute positions between the predicted and correct orders as follows:

$$\text{PAcc} = \frac{1}{|D|} \sum_{i \in |D|} \sum_{j \in |p_i|} \mathbb{I}(o_{s_j} = \hat{o}_{s_j}) \quad (12)$$

where $o_{s_j}$ denotes the absolute position of sentence $s_j$ in the correct sequence $S_j$. Likewise, $\hat{o}_{s_j}$ denotes the absolute position of sentence $s_j$ in the predicted sequence $\hat{S}_j$. $\mathbb{I}(o_{s_j} = \hat{o}_{s_j})$ denotes the indicator function equal to 1 if $o_{s_j} = \hat{o}_{s_j}$ and 0 otherwise.

### 4.3 Hyperparameters

For LDA-based topic modeling, we decided its hyperparameters $\beta$ and $d_t$ with a grid search on each dataset. Following Griffiths and Steyvers (2004), we kept $\alpha = 0.1$ and $\beta = 50/d_t$ constantly and obtained the best results with $d_t = 300$ for all abstract datasets and $d_t = 200$ for SIND. We initially configured $d_w = 200$ with the Glove vectors and updated the word vectors during training. For our attentive pointer decoder, we followed the same hyperparameters in Transformer and used a learning rate of 0.01. Other parameters were initialized randomly based on He et al. (2015).

## 5 Experimental Results

This section reports experimental results on the sentence ordering task for determining a coherent order of a given sentence. The proposed TGCM was compared with state-of-the-art methods as baselines such as Pairwise Model (Chen et al., 2016), Seq2seq (Logeswaran et al., 2018), RNN Decoder (Logeswaran et al., 2018), V-LSTM+PtrNet (Logeswaran et al., 2018), CNN+PtrNet (Gong et al., 2016), LSTM+PtrNet (Gong et al., 2016), and ATTOrderNet (Cui et al., 2018). Here, except the random model, all of the baselines are based on neural networks, which are typically more competitive than traditional approaches (e.g., utilizing handcraft features). For the LDA-based topic model in our preprocessing, we obtained the topic distributions of words by learning their relative importance for each topic.

| | NIPS abstract | | | ANN abstract | | | NSF abstract | | | arXiv abstract | | | SIND | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\tau$ | PMR | PAcc | $\tau$ | PMR | PAcc | $\tau$ | PMR | PAcc | $\tau$ | PMR | PAcc | $\tau$ | PMR | PAcc |
| Random Model | 0 | 7.53 | 15.59 | 0 | 8.07 | 19.36 | 0 | 5.42 | 9.46 | 0 | 8.07 | 14.11 | 0 | 6.05 | 9.32 |
| Pairwise Model | 0.47 | 19.72 | 26.63 | 0.58 | 21.54 | 41.82 | 0.25 | 13.56 | 15.51 | 0.66 | 33.43 | 50.79 | 0.32 | 10.43 | 30.75 |
| Seq2seq | 0.27 | 14.39 | 21.18 | 0.40 | 18.09 | 36.62 | 0.10 | 11.63 | 13.68 | 0.52 | 29.43 | 45.67 | 0.21 | 8.64 | 26.18 |
| RNN Decoder | 0.67 | 23.31 | 48.22 | 0.66 | 21.31 | 52.06 | 0.48 | 14.87 | 25.79 | 0.66 | 35.53 | 48.31 | 0.38 | 10.68 | 31.53 |
| V-LSTM+PtrNet | 0.72 | 27.87 | 51.55 | 0.73 | 29.79 | 58.06 | 0.51 | 18.53 | 28.33 | 0.72 | 41.74 | 55.90 | 0.45 | 13.44 | 35.26 |
| CNN+PtrNet | 0.66 | 26.79 | 48.64 | 0.69 | 26.65 | 58.21 | 0.52 | 16.73 | 33.22 | 0.71 | 39.28 | 52.92 | 0.48 | 12.32 | 35.52 |
| LSTM+PtrNet | 0.67 | 28.20 | 50.87 | 0.69 | 30.41 | 58.20 | 0.52 | 19.53 | 32.45 | 0.72 | 40.44 | 54.31 | 0.48 | 12.34 | 34.45 |
| ATTOrderNet | 0.72 | 29.87 | 56.09 | 0.73 | 32.11 | 63.24 | **0.55** | 21.35 | 37.72 | 0.73 | 42.19 | 56.11 | 0.49 | 14.01 | 36.24 |
| TGCM-S | 0.72 | 29.02 | 57.67 | 0.74 | 34.14- | 64.65 | 0.53 | 21.09 | 39.15 | 0.73 | 42.51 | 55.16 | 0.51 | 14.41 | 36.75 |
| TGCM | **0.75** | **31.44** | **59.43** | **0.75** | **36.69** | **65.16** | **0.55** | **22.35** | **42.67** | **0.75** | **44.28** | **58.31** | **0.53** | **15.18** | **38.71** |

Table 2: Performance of neural networks based models on the sentence ordering task. Best results are in boldface.

| | arXiv | | SIND | |
|---|---|---|---|---|
| | Head | Tail | Head | Tail |
| Random Model | 23.06 | 23.16 | 22.78 | 22.56 |
| Pairwise Model | 84.85 | 62.37 | - | - |
| CNN+PtrNet | 89.43 | 65.36 | 73.53 | 53.26 |
| LSTM+PtrNet | 90.47 | 66.49 | 74.66 | 53.30 |
| ATTOrderNet | 91.00 | 68.08 | 76.00 | 54.42 |
| TGCM-S | 91.15 | 67.93 | 77.34 | 54.64 |
| TGCM | **92.46** | **69.45** | **78.98** | **56.24** |

Table 3: Performance of predicting the correct head and tail sentences on arXiv and SIND. The results are directly taken from (Cui et al., 2018).

We could infer the topic latent vectors of new documents at the test phases.

Given a shuffled sentence sequence, the main goal is to find the most coherent sentence sequence. The coherence probability of a predicted sentence sequence is approximated to the coherence probability of a correct sentence sequence. In Table 2-3, the several values of some models are directly taken from (Cui et al., 2018), while we implemented the rest of models using their public code with the same experimental setup. Some methods do not release their implemented codes and so were implemented. In the case of the latest model ATTOderNet, we only utilized the high-performance model among three versions.

We discuss the reason for our highest performance among previous models by classifying the factors affecting performance into four categories: sentence set modeling, topic latent vectors, permutation invariance, and attention at decoder.

### 5.1 Impact of Sentence Set Modeling

This is related to the global dependencies of an entire paragraph. We observed that the performances of the random model and the pairwise model (Chen et al., 2016) were the worst. The pairwise model only learns the relative order from sentence-pair interactions. Since the two models do not consider all sentences at a time, they cannot leverage paragraph-level information. In other words, they cannot capture the global dependencies that help sentence ordering.

### 5.2 Impact of Topic Latent Vectors

Furthermore, we evaluated the variant version of TGCM, referred to TGCM-S, where the encoder of TGCM-S do not combine topic latent vectors and original sentence vectors. Also, the decoder of TGCM-S do not take newly created topic latent vectors. This allows us to explore whether the topic latent vectors actually help the sentence ordering task. Since TGCM-S cannot perform coherent topical context matching, TGCM-S do not elaborately capture the local dependencies, which help to identify tight semantic connections between the current predicted sentence and each next-sentence candidate.

As shown in the Tables 2-3, full TGCM outperformed TGCM-S which employs only attention-based bi-LSTMs without topic modeling. Although TGCM-S do not utilize topical context features at encoder and decoder (such as $\mathbf{t}_p$ and $\mathbf{t}_s$), TGCM-S also employs the transformer-based attentive pointer decoder. Thus, we found that topic modeling additionally contributes to modeling sentences through global topical context relating to local topical context of each sentence, resulting in improving the sentence ordering performance.

### 5.3 Impact of Permutaion Invariance

Compared to ATTOrderNet and our TGCM, the performance of other models was relatively low.

The conventional models typically adopt hierarchical RNN-based encoders, which combine sentence vectors sequentially and generate paragraph-level representation as a vector. Since they are dependent on the permutation of the input sentences, their paragraph-level representations are not reliable.

On the other hand, self-attention-based ATTOrderNet (encoder side) and attention-based our TGCM/TGCM-S (decoder side) use a set of sentence representations in a permutation-invariant manner rather than a single vector to represent a paragraph. Thus, these models can capture global dependencies regardless of the order of input sentences. The outputs of their encoder are more informative and reliable for improving the sentence ordering performance.

### 5.4 Impact of Attention at Decoder

This is related to solving the bottleneck problem by enhancing the expressiveness of the decoder's input and capturing the global and local dependencies among input sentences simultaneously.

Table 2-3 show that ATTOrderNet and TGCM, which utilize an attention mechanism, perform much better than all the other models. With an encoder employing self-attention mechanism, ATTOrderNet can capture some global dependencies with the reliable paragraph-level representation regardless of their input order. However, since ATTOrderNet also compresses sentence vectors into a single fixed-length paragraph vector via an average pooling layer, the word/sentence-level semantics from the encoder to the decoder are diluted.

In contrast to ATTOrderNet, TGCM feeds the sentence vectors directly into a decoder without compressing them into a single vector. Then, the attention mechanism allows our decoder to receive and utilize the sentence vectors, regardless of the order and number of sentences. As a result, since our TGCM could elaborately capture both local and global dependencies simultaneously, TGCM showed the best sentence ordering performance among all comparative models on all datasets.

## 6 Conclusions

We propose TGCM which better preserves local and global information for sentence ordering. Our attentive pointer decoder fully utilizes the semantics of sentence vectors without being compressed into a paragraph vector. Our sentence encoder produces topic-sensitive sentence vectors via topic modeling. With the coherent topical context matching between the current predicted sentence and each next-sentence candidate, we promote local dependencies that help identify the tight semantic connections. The empirical results on sentence ordering demonstrate that TGCM outperforms state-of-the-art models.

## Acknowledgments

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIP; Ministry of Science, ICT & Future Planning) (No. NRF-2019R1A2B5B01070555).

## References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation*, pages 265–283.

Harsh Agrawal, Arjun Chandrasekaran, Dhruv Batra, Devi Parikh, and Mohit Bansal. 2016. Sort story: Sorting jumbled images and captions into stories. *arXiv preprint arXiv:1606.07493*.

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6077–6086.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Regina Barzilay and Mirella Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Xinchi Chen, Xipeng Qiu, and Xuanjing Huang. 2016. Neural sentence ordering. *arXiv preprint arXiv:1607.06952*.

Baiyun Cui, Yingming Li, Ming Chen, and Zhongfei Zhang. 2018. Deep attentive sentence ordering network. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4340–4349.

Adji B Dieng, Chong Wang, Jianfeng Gao, and John Paisley. 2016. Topicrnn: A recurrent neural network with long-range semantic dependency. *arXiv preprint arXiv:1611.01702*.

J. Duchi, E. Hazan, and Y. Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Hongyu Gong, Tarek Sakakini, Suma Bhat, and Jinjun Xiong. 2018. Document similarity for texts of varying lengths via hidden topics. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2341–2351.

Jingjing Gong, Xinchi Chen, Xipeng Qiu, and Xuanjing Huang. 2016. End-to-end neural sentence ordering using pointer network. *arXiv preprint arXiv:1611.04953*.

Thomas L Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National academy of Sciences*, 101(suppl 1):5228–5235.

Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. 2018. Long text generation via adversarial training with leaked information. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

K. He, X. Zhang, S. Ren, and J. Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, pages 1026–1034.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Ting-Hao Kenneth Huang, Francis Ferraro, Nasrin Mostafazadeh, Ishan Misra, Aishwarya Agrawal, Jacob Devlin, Ross Girshick, Xiaodong He, Pushmeet Kohli, Dhruv Batra, et al. 2016. Visual storytelling. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1233–1239.

Aishwarya Jadhav and Vaibhav Rajan. 2018. Extractive summarization with swap-net: Sentences and words from alternating pointer networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 142–151.

Mingmin Jin, Xin Luo, Huiling Zhu, and Hankz Hankui Zhuo. 2018. Combining deep learning and topic modeling for review understanding in context-aware recommendation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 1605–1614.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Mirella Lapata. 2003. Probabilistic text structuring: Experiments with sentence ordering. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 545–552. Association for Computational Linguistics.

Mirella Lapata. 2006. Automatic evaluation of information ordering: Kendall's tau. *Computational Linguistics*, 32(4):471–484.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Jiwei Li and Dan Jurafsky. 2016. Neural net models for open-domain discourse coherence. *arXiv preprint arXiv:1606.01545*.

Biao Liu and Minlie Huang. 2016. A sentence interaction network for modeling dependence between sentences. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 558–567.

Xiaodong Liu, Yelong Shen, Kevin Duh, and Jianfeng Gao. 2017. Stochastic answer networks for machine reading comprehension. *arXiv preprint arXiv:1712.03556*.

Lajanugen Logeswaran, Honglak Lee, and Dragomir Radev. 2018. Sentence ordering and coherence modeling using recurrent neural networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Edward Loper and Steven Bird. 2002. Nltk: the natural language toolkit. *arXiv preprint cs/0205028*.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Shashi Narayan, Ronald Cardenas, Nikos Papasarantopoulos, Shay B Cohen, Mirella Lapata, Jiangsheng Yu, and Yi Chang. 2018a. Document modeling with external attention for sentence extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2020–2030.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018b. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *arXiv preprint arXiv:1808.08745*.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016. Text matching as image recognition. In *Thirtieth AAAI Conference on Artificial Intelligence*.

J. Pennington, R. Socher, and C. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Aliaksei Severyn and Alessandro Moschitti. 2016. Modeling relational information in question-answer pairs with convolutional neural networks. *arXiv preprint arXiv:1604.01178*.

Gehui Shen, Yunlun Yang, and Zhi-Hong Deng. 2017. Inter-weighted alignment network for sentence pair modeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1179–1189.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Chuanqi Tan, Furu Wei, Wenhui Wang, Weifeng Lv, and Ming Zhou. 2018. Multiway attention networks for modeling sentence pairs. In *IJCAI*, pages 4411–4417.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Multi-cast attention networks for retrieval-based question answering and response prediction. *arXiv preprint arXiv:1806.00778*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. 2015a. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015b. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Qingyun Wang, Zhihao Zhou, Lifu Huang, Spencer Whitehead, Boliang Zhang, Heng Ji, and Kevin Knight. 2018. Paper abstract writing through editing mechanism. *arXiv preprint arXiv:1805.06064*.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*.

Xingshan Zeng, Jing Li, Lu Wang, Nicholas Beauchamp, Sarah Shugars, and Kam-Fai Wong. 2018. Microblog conversation recommendation via joint modeling of topics and discourse. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 375–385.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 207–212.

# Neural Generative Rhetorical Structure Parsing

**Amandla Mabona**
Dept. of Computer Science and Technology
University of Cambridge
amandla.mabona@cl.cam.ac.uk

**Laura Rimell**
DeepMind
London, UK
laurarimell@google.com

**Stephen Clark**
DeepMind
London, UK
clarkstephen@google.com

**Andreas Vlachos**
Dept. of Computer Science and Technology
University of Cambridge
andreas.vlachos@cst.cam.ac.uk

## Abstract

Rhetorical structure trees have been shown to be useful for several document-level tasks including summarization and document classification. Previous approaches to RST parsing have used discriminative models; however, these are less sample efficient than generative models, and RST parsing datasets are typically small. In this paper, we present the first generative model for RST parsing. Our model is a document-level RNN grammar (RNNG) with a bottom-up traversal order. We show that, for our parser's traversal order, previous beam search algorithms for RNNGs have a left-branching bias which is ill-suited for RST parsing. We develop a novel beam search algorithm that keeps track of both structure- and word-generating actions without exhibiting this branching bias and results in absolute improvements of 6.8 and 2.9 on unlabelled and labelled F1 over previous algorithms. Overall, our generative model outperforms a discriminative model with the same features by 2.6 F1 points and achieves performance comparable to the state-of-the-art, outperforming all published parsers from a recent replication study that do not use additional training data.

## 1 Introduction

Understanding a document's discourse-level organization is important for correctly interpreting it, and discourse analyses have been shown to be helpful for several NLP tasks (Bhatia et al., 2015; Ji and Smith, 2017; Feng and Hirst, 2014b; Ferracane et al., 2017). A popular formalism for discourse analysis is Rhetorical Structure Theory (RST) (Mann and Thompson, 1988) (Fig. 1) which represents a document as a tree of discourse units recursively built by connecting smaller units through rhetorical relations. Learning to predict RST trees is difficult because it depends on pragmatics as well as literal meaning, and the English RST Discourse Treebank (RST-DT) (Carlson et al., 2003) is small by the standards of modern parsing datasets, with 347 training documents.

Previous approaches to RST parsing (Ji and Eisenstein, 2014; Feng and Hirst, 2014a; Joty et al., 2015; Braud et al., 2017) have used locally normalized discriminative models. However, these are known to have worse performance than generative models when there is little training data (Ng and Jordan, 2002; Yogatama et al., 2017).

Unlike locally normalised discriminative models, generative models are not susceptible to label bias (Lafferty et al., 2001). The success of generative (Dyer et al., 2016; Charniak et al., 2016) and globally normalised (Andor et al., 2016) syntactic parsers suggests that reducing label bias leads to better performance. We hypothesize that using a generative parser would also lead to improved performance on RST parsing. However, while they are free from label bias, generative parsers require more sophisticated search algorithms for decoding. Fried et al. (2017) presented a word-level beam search algorithm that made it possible to decode directly from neural generative parsers rather than using them as rerankers.

In this paper, we present the first generative RST parser[1]. Our model is a document-level version of an RNN Grammar (RNNG, Dyer et al. (2016)) defined through a transition system with both word- and structure-generating actions. It uses distributed representations of discourse units and transition probabilities parametrized by RNNs to model unbounded dependencies in a document.

For our discourse parser, we find that Fried et al. (2017)'s word-level beam search algorithm is bi-

[1] Ji et al. (2016) introduced a neural generative discourse parser, but they used the annotation scheme of the Penn Discourse Treebank (Prasad et al., 2008) and Switchboard Dialog Act (Godfrey et al., 1992) corpora, predicting flat discourse representations between adjacent sentences, rather than hierarchical relations among clauses.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2284–2295,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

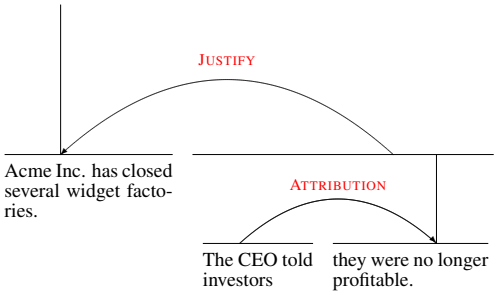


Figure 1: An example of an RST tree.

ased towards producing left-branching trees. We analyse the source of this bias and develop a novel beam search algorithm that removes it by tracking both word- and structure-generating actions. On the RST-DT development set, our algorithm leads to improvements of 6.8% and 2.9% on unlabelled and labelled attachment accuracies when decoding from the same parser, compared to word-level beam search. On the RST-DT test set, our generative parser outperforms a discriminative version with the same features by 2.6% on labelled attachment accuracy. Overall, our parser obtains a labelled attachment score of 45.0%, outperforming all published parsers in a recent replication study that do not use additional training data.

## 2 Rhetorical Structure Theory

Rhetorical Structure Theory describes the structure of a document in terms of text spans that form discourse units and the relations between them. The basic unit of analysis is an *elementary discourse unit* (EDU) which can be assumed to be a syntactic clause. A *unit* is made up of two or more adjacent discourse units (which can be EDUs or other units) that are in some *rhetorical relation*.

Most rhetorical relations are binary and asymmetric with one argument, the *nucleus*, being more important than the other, the *satellite*. *Importance* is defined through a deletion test: a text becomes incoherent if a nucleus is deleted, but not if a satellite is. These binary asymmetric relations are called *mononuclear* relations. The remaining relations are symmetric, having two or more arguments of equal importance, and are called *multinuclear* relations.

An *RST tree* or *analysis* is a nested collection of discourse units that are either EDUs or units, where the top unit spans the whole text (Mann and Thompson, 1988). RST parsing is the task of automatically predicting RST trees for documents.

## 3 Rhetorical Structure RNNGs

In this section, we present a generative model for predicting RST trees given a document segmented into a sequence of EDUs $e_{1:m}$[2]. The model is a document-level RNNG in bottom-up traversal order (Kuncoro et al., 2018). We first describe syntactic RNNGs in section 3.1. We then describe our parser's transition system in section 3.2, and its transition model in section 3.3.

### 3.1 Recurrent Neural Network Grammars

Recurrent neural network grammars are a class of syntactic language models that define a joint probability distribution $p(\boldsymbol{x}, \boldsymbol{y})$ over sentences and their phrase structure trees. An RNNG is defined by a triple $(N, \Sigma, \Theta)$ with $N$ a finite set of nonterminal symbols, $\Sigma$ a finite set of terminal symbols and $\Theta$ neural network parameters.

RNNGs generate sentences and their parse trees through actions[3] in an abstract state machine. A machine state is a tuple $\langle S, B \rangle$ where $S$ is a stack which holds partial phrase structure trees and $B$ is a buffer which holds sentence prefixes. The transitions push new subtrees onto the stack, combine subtrees already there, and append terminals to the buffer until the stack contains a single phrase structure tree and the buffer contains a complete sentence. The original presentation in Dyer et al. (2016) used the following transition system:

NT$(X)$ Push the nonterminal node $(X$ onto the top of the stack, where $X \in N$.

GEN$(w)$ Push the terminal symbol $w \in \Sigma$ onto the top of the stack and the end of the buffer.

REDUCE Pop subtrees $\tau_1, \cdots, \tau_l$ from the top of the stack until the first nonterminal node $(X$ is reached and push the subtree $(X\tau_1 \cdots \tau_l)$ onto the top of the stack.

A sentence $\boldsymbol{x}$ and phrase structure tree $\boldsymbol{y}$ are generated by a unique sequence of actions $a_{1:k}$. The joint distribution $p(\boldsymbol{x}, \boldsymbol{y})$ is defined as the probability of the action sequence $a_{1:k}$:

$$p(\boldsymbol{x}, \boldsymbol{y}) = \prod_{j=1}^{k} p(a_j | a_{<j}) = \prod_{j=1}^{k} p(a_j | S_j, B_j) \quad (1)$$

[2]As in most previous work on RST parsing, we use gold EDU segmentations in our experiments, but our parser would use the output of an EDU segmenter in practice.

[3]We use "action" and "transition" interchangeably.

| Action | Before | After | Probability | Condition |
|---|---|---|---|---|
| $\mathsf{GEN}(e)$ | $\langle S, B\rangle$ | $\langle S\|\mathsf{EDU}(e), B\|e\rangle$ | $p_{trans}(\mathsf{GEN}\|S) \cdot p_{gen}(e\|S)$ | $\|B\| < m$ |
| $\mathsf{RE}(r, n)$ | $\langle S\|U_L\|U_R, B\rangle$ | $\langle S\|\big(\mathsf{Unit}(r, n)\ U_L\ U_R\big), B\rangle$ | $p_{trans}(\mathsf{RE}(r, n)\|S)$ | $\|S\| \geq 2$ |

Table 1: Our transition system. $|S|$ is the number of discourse units on the stack, $|B|$ is the number of EDUs in the buffer and $m$ is the number of EDUs in the whole document, $r$ is a relation label and $n$ is a nuclearity label.

| Stack | Buffer | Prediction |
|---|---|---|
| $\epsilon$ | $\epsilon$ | $\mathsf{GEN}(e_1)$ |
| $\mathsf{EDU}(e_1)$ | $e_1$ | $\mathsf{GEN}(e_2)$ |
| $\mathsf{EDU}(e_1)\|\mathsf{EDU}(e_2)$ | $e_1\|e_2$ | $\mathsf{GEN}(e_3)$ |
| $\mathsf{EDU}(e_1)\|\mathsf{EDU}(e_2)\|\mathsf{EDU}(e_3)$ | $e_1\|e_2\|e_3$ | $\mathsf{RE}(\textit{ATTR}, \textit{SN})$ |
| $\mathsf{EDU}(e_1)\|(\mathsf{Unit}(\textit{ATTR}, \textit{SN})\ \mathsf{EDU}(e_2)\ \mathsf{EDU}(e_3))$ | $e_1\|e_2\|e_3$ | $\mathsf{RE}(\textit{JUST}, \textit{NS})$ |
| $\big(\mathsf{Unit}(\textit{JUST}, \textit{NS})\ \mathsf{EDU}(e_1)\ \big(\mathsf{Unit}(\textit{ATTR}, \textit{SN})\ \mathsf{EDU}(e_2)\ \mathsf{EDU}(e_3)\big)\big)$ | $e_1\|e_2\|e_3$ | |

[$e_1$ Acme Inc. has closed several widget factories. ] [$e_2$ The CEO told investors ] [$e_3$ they were no longer profitable. ]

Table 2: An example of a completed computation in our transition system.

The next action distribution $p(a_j|S_j, B_j)$ is parametrized using neural embeddings of the stack and buffer. Briefly, the next action distribution is computed using a softmax on the output of a linear transformation on the state embedding, which is the concatenation of a buffer embedding and a stack embedding. The buffer embedding is the final hidden state of an LSTM that reads the word embeddings of the words in the buffer. The stack embedding is the hidden state of a stack LSTM that reads the embeddings of the subtrees on the stack. The embeddings of the subtrees on the stack are computed recursively using a bidirectional LSTM that reads the embeddings of the nonterminal symbol and its children.

### 3.2 Transition System

We modify the RNNG generative process so that it generates an EDU-segmented document and its RST tree. In our model, the stack $S$ holds partial RST trees and the buffer $B$ holds a sequence of EDUs (a prefix of a document's EDU segmentation). The transitions generate EDUs and push them onto the stack and buffer, and combine RST subtrees on the stack into new subtrees. The process terminates when the buffer contains a complete document and the stack a single RST tree.

Kuncoro et al. (2018) presented an RNNG variant with a bottom-up transition system that replaces the $\mathsf{NT}(X)$ and $\mathsf{REDUCE}$ transitions with a single $\mathsf{REDUCE}(X, n)$ transition, as in traditional shift-reduce parsers. In initial experiments, we found this variant outperformed a model using the original top-down transition system. We hypothesize this is because an RST non-terminal's label is more difficult to predict from its parent's label than is the case in phrase structure trees, while a parent's label can be predicted once its children have been seen.

Finally, RST trees are traditionally binarized so we modify the $\mathsf{REDUCE}$ transitions accordingly, resulting in the following transition system (see also Table 1):

$\mathsf{GEN}(e)$ Generate the EDU $e$ and push it onto the top of the stack and the end of the buffer.

$\mathsf{RE}(r, n)$ Pop the top two discourse units ($U_L$ and $U_R$) from the stack and push the unit $\big(\mathsf{Unit}(r, n)\ U_L\ U_R\big)$ onto the top of the stack, with $r$ and $n$ relation and nuclearity labels.

In our experiments, the relation labels $r$ are the 18 coarse-grained relations of Carlson and Marcu (2001), while the nuclearity labels $n$ are in $\{\textit{SN}, \textit{NS}, \textit{NN}\}$ corresponding respectively to a mononuclear relation with the nucleus on the right or the left and a binarized multinuclear relation.

Both transitions have conditions on when they can be performed (Table 1). A *computation* is a sequence of transitions where the condition for each transition is satisfied in its preceding state. A *completed computation* for an input sequence is a computation where the final state buffer contains the input sequence and the final state stack contains a single tree. Table 2 shows an example of a completed computation for our transition system.

### 3.3 Transition Model

In initial experiments we found, as did Kuncoro et al. (2017) for syntactic parsing, that conditioning only on the stack led to better parsing accuracy, so we specify the next action distribution as $p(a_j|S_j)$. To handle the unbounded number of possible EDUs, we parametrize the probabilities of $\mathsf{GEN}(e)$ actions using a neural language model. The next action distribution is factorised into a structural action distribution $p_{trans}$ and a generation distribution $p_{gen}$ as in Buys and Blunsom (2018), so that $p(\mathsf{RE}(r,n)|S) = p_{trans}(\mathsf{RE}(r,n)|S)$ and $p(\mathsf{GEN}(e)|S) = p_{trans}(\mathsf{GEN}|S) \cdot p_{gen}(e|S)$ where $p_{gen}$ is the neural language model.

We parametrize $p_{trans}$ as a feedforward neural network on an embedding of the stack $\mathbf{h}_S(S)$. In initial experiments we found, consistent with Morey et al. (2017), that a model with neural embeddings as its only features performed poorly. We therefore compute the representation using both neural embeddings of the discourse units on the stack (Section 3.3.1) and a set of structural features extracted from the stack (Section 3.3.2).

#### 3.3.1 Neural Embeddings

To produce the stack embedding, we first require embeddings for both EDUs and units. We embed EDUs with bidirectional LSTMs[4]. If $e$ is an EDU consisting of the word sequence $w_{1:k}$, then

$$\begin{aligned} \mathbf{h}_k^{\rightarrow} &= \mathsf{LSTM}^{(\rightarrow)}(\mathbf{w}_{1:k}, \mathbf{h}_0^{\rightarrow}) \\ \mathbf{h}_k^{\leftarrow} &= \mathsf{LSTM}^{(\leftarrow)}(\mathbf{w}_{k:1}, \mathbf{h}_0^{\leftarrow}) \end{aligned} \tag{2}$$

where $\mathbf{w}_t$ is the word embedding of $w_t$. The embedding for $e$, $\mathbf{h}_{EDU}(e)$, is the concatenation of the final forward and backward hidden states:

$$\mathbf{h}_{EDU}(e) = [\mathbf{h}_k^{\rightarrow}; \mathbf{h}_k^{\leftarrow}] \tag{3}$$

We embed units by composing their arguments with a Tree LSTM[5] (Teng and Zhang, 2017). A Tree LSTM recursively composes vectors while using memory cells to track long-term dependencies. We produce a new representation for each EDU $e$ by applying a linear transformation to the EDU embeddings (omitting bias terms for brevity):

$$\mathbf{h}_U(e) = \mathbf{W}_{U,h} \cdot \mathbf{h}_{EDU}(e) \tag{4}$$

For a unit, we define the "nuclear" EDU of a unit recursively as the nucleus if the nucleus is an EDU, or the nuclear EDU of the nucleus if the nucleus is itself a unit. For multinuclear relations, we take the left-most nucleus. Then, if $\big(\mathsf{Unit}(r,n)\ U_L\ U_R\big)$ is a unit and $e_N$ is its nuclear EDU, $\mathbf{h}_{EDU}(e_N)$ is the embedding of the nuclear EDU, and $\mathbf{h}_R(r,n)$ is an embedding of the nuclearity-relation pair $(r,n)$ in a lookup table:

$$\begin{aligned} \mathbf{h}_U(U) = \mathsf{TREELSTM}([\mathbf{h}_{EDU}(e_N); \mathbf{h}_R(r,n)], \\ \mathbf{h}_U(U_L), \mathbf{h}_U(U_R)) \end{aligned} \tag{5}$$

where $\mathbf{h}_U(U_L)$ and $\mathbf{h}_U(U_R)$ are the hidden state and memory cell of the left and right argument of the unit respectively.

We embed the stack with a stack LSTM (Dyer et al., 2015). If the stack contents are $D_1|\cdots|D_m$ with each $D_i$ being a discourse unit, then

$$\mathbf{h}_S^N(S) = \mathsf{LSTM}^S(\mathbf{h}_U(D_{1:m}), \mathbf{h}_0^S) \tag{6}$$

#### 3.3.2 Structural Features

We extract additional features from the stack that have been found to be useful in prior work. As in Braud et al. (2017), for each discourse unit, we extract the word embeddings of up to three words whose syntactic head is not in the unit, adding padding if there are fewer than three. We concatenate these features for the top two discourse units on the stack, using a dummy embedding if the stack only contains one discourse unit. We write $\mathbf{h}_S^{head}(S)$ for these features.

We use a categorical feature for whether the top two discourse units are: in the same sentence; in different sentences; or incomparable since one of them spans multiple sentences. We also use an equivalent feature for paragraphs. Feature values are represented by embeddings in a lookup table. We write $\mathbf{h}_S^{comp}(S)$ for these features.

Finally, we extract features describing the dominance relation (Soricut and Marcu, 2003) between the top two discourse units on the stack. If there is a word in one discourse unit whose syntactic head is in the other, we extract the word embeddings of these two words as well as an embedding of the dependency relation between them, otherwise we use a single dummy embedding. We write $\mathbf{h}_S^{dom}(S)$ for these features.

[4] We track memory cells and use them when updating the hidden state in LSTMs and Tree LSTMs, but use only the hidden states for stack embeddings. Initial hidden states and memory cells are learned parameters.

[5] Since constituency trees are $n$-ary branching, RNNGs for constituency parsing have used a bidirectional LSTM composition function (Dyer et al., 2016; Kuncoro et al., 2017, 2018) to compose the variable number of children. RST trees are binarized so we do not need this feature.

The structural feature representation is then the concatenation of these three features:

$$\mathbf{h}_S^F(S) = [\mathbf{h}_S^{head}(S); \mathbf{h}_S^{comp}(S); \mathbf{h}_S^{dom}(S)] \quad (7)$$

and the full stack representation is the concatenation of the neural embedding and the feature representation:

$$\mathbf{h}_S(S) = [\mathbf{h}_S^N(S); \mathbf{h}_S^F(S)] \quad (8)$$

#### 3.3.3 Probability Distributions

The action distribution is parametrized using the stack representation and an MLP:

$$\begin{aligned} p_{trans}(a|S) &= p_{trans}(a|\mathbf{h}_S(S)) \\ &= \text{softmax}(\mathbf{W}_{trans} \cdot \mathbf{h}_S(S)) \end{aligned} \quad (9)$$

We parametrize the EDU generation distribution $p_{gen}(e|S)$ with an LSTM decoder:

$$\mathbf{h}_t^{DEC} = \mathsf{LSTM}^{DEC}(\mathbf{w}_t, \mathbf{h}_{t-1}^{DEC}) \quad (10)$$

If $e = w_{1:k}$ then

$$p_{gen}(e|S) = p_{gen}(w_{1:k}|S) \quad (11)$$

$$= \prod_{t=1}^{k} p_{gen}(w_t|w_{<t}, S) \quad (12)$$

$$= \prod_{t=1}^{k} p_{gen}(w_t|\mathbf{h}_{t-1}^{DEC}, \mathbf{h}_S(S)) \quad (13)$$

where

$$p_{gen}(w_t|\mathbf{h}_{t-1}^{DEC}, \mathbf{h}_S(S)) \quad (14)$$

$$= \text{softmax}(\mathbf{W}_{gen} \cdot [\mathbf{h}_S(S); \mathbf{h}_{t-1}^{DEC}]) \quad (15)$$

## 4 Inference

Our generative model specifies a joint probability $p(\boldsymbol{x}, \boldsymbol{y})$. We parse a document $\boldsymbol{x}$ by finding the MAP tree $\boldsymbol{y}^*$:

$$\boldsymbol{y}^* = \underset{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})}{\text{argmax}}\, p(\boldsymbol{y}|\boldsymbol{x}) \quad (16)$$

$$= \underset{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})}{\text{argmax}}\, p(\boldsymbol{x}, \boldsymbol{y}) \quad (17)$$

The search space grows exponentially with the input length, so we must perform inexact search as our model conditions on the entire relation structure of every subtree on the stack.

Search is generally more difficult for generative models than for discriminative ones, requiring more complex search algorithms. For this reason, Dyer et al. (2016) used RNNGs only to rerank the output of a discriminative parser. Fried et al. (2017) presented the first algorithm for decoding directly from RNNGs to give competitive performance. They found that action-level beam search (Zhang and Clark, 2008) gave poor performance for constituency parsing with RNNGs. The problem was that GEN actions almost always have lower probabilities than structure-generating actions, causing computations where GEN actions come earlier to "fall off the beam" even if the completed computation would have a higher probability than other completed computations.

To address this problem, Fried et al. (2017) proposed *word-level beam search* (Algorithm 1). Briefly, the algorithm keeps an array of beams indexed by the current position in the sequence and the number of structure-generating actions taken since this position was reached. The first beam for the current position $\mathcal{B}(i, 0)$ is filled from the successors of beams for the previous position $\mathcal{B}(i-1, j)$ (lines 4 to 17) starting with $\mathcal{B}(i-1, 0)$ (line 4) and incrementing $j$ (line 17) until there are at least $k$ items in $\mathcal{B}(i, 0)$ (line 5). The intuition is that analyses with the smallest number of structural actions since the previous beam was pruned have priority on the current beam.

We applied Fried et al. (2017)'s algorithm[6] to our model, but found it was biased towards producing left-branching trees. This led to poor performance as the *right-frontier constraint* (Polanyi, 1988; Webber, 1988; Asher, 2012; Asher et al., 2003) suggests discourse trees should be generally right-branching. In the next section, we present an analysis of the source of this bias and a novel beam search algorithm that corrects it.

### 4.1 Diagnosing Branching Bias

The trees returned by a trained parser depend on both the (learned) scoring model and the search algorithm. We can isolate bias in search algorithms by studying the trees they return when the scoring model contains no information. Intuitively, if the scoring model has no preference over trees, then any preference shown by the parser is the result of biases in the search algorithm.

We tested whether the left-branching bias came from the word-level beam search (the search algorithm of Fried et al. (2017)) by using it to parse sequences of various lengths using a bottom-up RNNG with a uniform scoring model. We broke

[6] We used *candidate fast-tracking* as described in Stern et al. (2017)'s extension to Fried et al. (2017)'s algorithm.

**Algorithm 1** Word-level Beam Search

```
1: function SEARCH(x_{1:m}, k)
2:     B[0,0] ← {(1, (ϵ, ϵ))}
3:     for i ← RANGE(0, m)
4:         j ← 0
5:         while |B[i,j]| ≥ 0 and |B[i+1,0]| < k
6:             for (v, s) ← TOP(B[i,j], b)
7:                 for (a, s') ← SUCC(s)
8:                     v' ← v · p(a|s)
9:                     if COMPLETE(s') then
10:                        PUSH(B[m+1,0], (v', s'))
11:                    else
12:                        switch a
13:                            case GEN(e_{i+1})
14:                                PUSH(B[i+1,0], (v', s'))
15:                            case RE(r, n)
16:                                PUSH(B[i,j+1], (v', s'))
17:            j ← j + 1
18:    return TOP(B[m+1,0], 1)
```

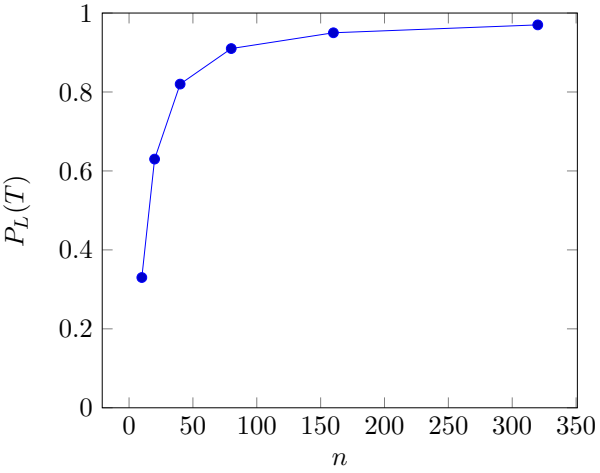


Figure 2: Median degree of left branching for trees obtained from a bottom-up RNNG with a uniform scoring model using word-level beam search for sequences of various lengths ($n$).

ties at beam cut-offs by uniform sampling without replacement. We measured branching bias using Sampson (1997)'s production-based measure of left-branching for parse trees which we write as $P_L(T)$ for a tree $T$. The measure is the fraction of non-terminals whose left child is also a non-terminal, and varies from 0 for a fully right-branching tree to $\frac{n-2}{n-1} \rightarrow 1$ for a fully left-branching tree, where $n$ is the number of leaves. Figure 2 shows the median value of this measure for 100 trees each for sequences of various lengths from our uniform scoring model. It shows substantial left-branching bias which increases with sequence length.

Word-level beam search has two sources of bias: first, computations with fewer RE actions since the last GEN action are added to the next beam first (lines 4 and 17 in Alg. 1). Computations with more actions are only considered if the next beam is not already full by the time they are reached (line 5 in Alg. 1). This means the next beam may fill up before these computations are even considered and they will "fall off the beam". A right-branching subtree over $k$ leaves has $k$ consecutive GEN actions followed by $k-1$ consecutive RE actions, meaning it results in a computation in $\mathcal{B}(i_k, k-1)$ where $i_k$ is the position of the $k$-th leaf. Thus right-branching subtrees are later in line to be considered and are increasingly likely to fall off the beam as they span more leaves.

Second, the beams $\mathcal{B}(i,j)$ contain computations with unequal numbers of actions. For a binary tree with $m$ leaves, all completed computations have $m$ GEN and $m-1$ RE actions. The total number of actions up to the $k$-th GEN action varies, though, from $k$ to $2k-2$. Since the probability of a computation $a_{1:l}$ is $\prod_{j=1}^{l} p(a_j|a_{<j})$, this means word-level beam search compares computations with different numbers of factors contributing to their probabilities. This bias does not necessarily favour left-branching trees, but it does introduce a potential problem when comparing computations.

### 4.2 Bag-Level Beam Search

We now present a beam search algorithm without these sources of bias (Algorithm 2). Our algorithm is based on a simple dynamic program that keeps track of the number of GEN and RE actions separately. This (i) allows us to consider computations from all source beams simultaneously and (ii) ensures all computations in a beam have the same number of actions. Since this is equivalent to keeping separate beams for different bags of unlabelled actions, we call the algorithm bag-level beam search.

We write $\mathcal{C}(i,j)$ for the set of computations with $i$ GEN actions and $j$ RE actions. Then all completed computations are in $\mathcal{C}(m, m-1)$ for an input sequence of length $m$.

For each computation $c \in \mathcal{C}(i,j)$, the last action was either a GEN or an RE action, so $c$ is either of the form $c = \text{GEN}|c'$ where $c' \in \mathcal{C}(i-1,j)$ or it is of the form $c = \text{RE}|c''$ where $c'' \in \mathcal{C}(i,j-1)$.

The highest scoring computation in $\mathcal{C}(i,j)$,

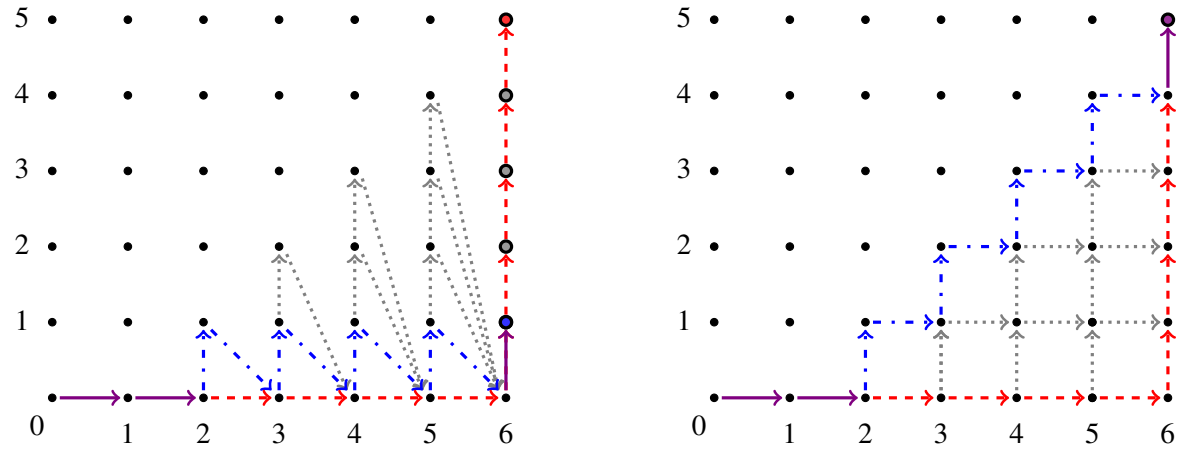


Figure 3: Word-level and bag-level beam search (left and right respectively) for an input sequence with 6 tokens. Nodes represent beams and paths represent computations. The horizontal axis is the number of GEN actions and the vertical axis is the number of RE actions for bag-level search and the number of RE actions since the last GEN for word-level search. We show the path of a left-branching tree in blue with dashed and dotted lines and a right-branching tree in red with dashed lines. We show possible transitions between beams that do not belong to either of these paths in gray with dotted lines. Red, blue and purple dots respectively show the beam where the computation of a right-branching tree, left-branching tree or both are completed.

**Algorithm 2** Bag-level Beam Search

```
1:  function SEARCH(x_{1:m}, k)
2:      B[0,0] ← {(1, (ϵ, ϵ))}
3:      for i ← RANGE(0, m)                          ▷ GEN(e_i)
4:          for j ← RANGE(0, i − 1)                  ▷ RE(r, n)
5:              for (v, s) ← TOP(B[i, j], k)
6:                  for (a, s′) ← SUCC(s)
7:                      v′ ← v · p(a|s)
8:                      switch a
9:                          case GEN(e_{i+1})
10:                             PUSH(B[i + 1, j], (v′, s′))
11:                         case RE(r, n)
12:                             PUSH(B[i, j + 1], (v′, s′))
13:
14:     return TOP(B[m, m − 1], 1)
```

$c^*(i,j) = \underset{c \in \mathcal{C}(i,j)}{\operatorname{argmax}}\, p(c)$, is then the highest scoring computation ending on a GEN or an RE[7]:

$$c^*(i,j) = \operatorname{argmax} \left\{ p(c) \middle| \begin{array}{l} c = \mathsf{GEN}|c^{'}, c^{'} \in \mathcal{C}(i-1,j); \\ c = \mathsf{RE}|c^{''}, c^{''} \in \mathcal{C}(i,j-1) \end{array} \right\} \tag{18}$$

There are exponentially many computations in $\mathcal{C}(i,j)$ so taking exact maxima is intractable. Therefore we only take maxima over beams $\mathcal{B}(i,j)$ which we update according to

$$\mathcal{B}(i,j) = \operatorname{argmax}_k \left\{ p(c) \middle| \begin{array}{l} c = \mathsf{GEN}|c^{'}, c^{'} \in \mathcal{B}(i-1,j); \\ c = \mathsf{RE}|c^{''}, c^{''} \in \mathcal{B}(i,j-1) \end{array} \right\} \tag{19}$$

[7]We omit actions' parameters for conciseness.

where, in the set notation, ";" means "or".

We perform this recursive calculation for all $i$ and $j$ using the dynamic program in Algorithm 2.

Figure 3 shows the differences between word-level and bag-level beam search with example trajectories through the array of beams for computations corresponding to a left-branching (blue, dashed and dotted) and a right-branching (red, dashed) tree. Each path through the lattice from $(0,0)$ to $(i,j)$ defines a computation and shows the beams it must pass through to end up in $\mathcal{B}(i,j)$.

The path length is equal to the number of actions taken to reach $(i,j)$. In word-level beam search, paths through beams with more consecutive RE actions (higher values on the vertical axis) are only explored if the next word beam is not already full (lines 4, 5 and 17 in Algorithm 1). This means the final beam may be full before the red path is considered, causing it to "fall off the beam". In bag-level beam search, paths into a beam from both source beams are considered and pruned simultaneously. This addresses the first source of bias, namely that sequences with fewer consecutive RE actions are given priority. All paths to $(i,j)$ also have the same length; in other words, all computations in $\mathcal{B}(i,j)$ have the same number of actions $(i+j)$, addressing the second source of bias.

| Metric | Algorithm | Beam Size | | | |
|---|---|---|---|---|---|
| | | 10 | 20 | 40 | 80 |
| S | Word-level Search | 58.3 | 58.4 | 58.4 | 60.8 |
| | Bag-level Search | 66.4 | 67.3 | 67.0 | 67.6 |
| N | Word-level Search | 50.3 | 50.5 | 50.5 | 51.8 |
| | Bag-level Search | 56.1 | 56.4 | 56.2 | 56.9 |
| R | Word-level Search | 43.1 | 43.2 | 43.2 | 43.7 |
| | Bag-level Search | 45.4 | 46.8 | 46.5 | 46.9 |
| F | Word-level Search | 42.0 | 42.3 | 42.3 | 42.9 |
| | Bag-level Search | 44.7 | 45.5 | 45.3 | 45.8 |

Table 3: Dev. set micro-averaged $F_1$ scores on labelled attachment for word-level and bag-level beam search.

# 5 Experiments

## 5.1 Dataset

We train and evaluate our models on the RST Discourse Treebank (RST-DT) (Carlson and Marcu, 2001). We evaluate on the standard test set, but we use 25 documents from the training set as a development set. To reduce the rare token count, we use the spaCy (Honnibal and Montani, 2017) named entity recognition model to replace named entities with their named entity tags.

## 5.2 Evaluation

We follow the evaluation setup used by Morey et al. (2017). They performed a replication study of several competitive RST parsers and implemented a consistent evaluation procedure. They found that micro- and macro-averaged $F_1$ had been used inconsistently in the RST parsing literature, and that the standard evaluation metrics (RST Parseval) gave inflated results. Following this study we evaluate using micro-averaged $F_1$ scores on labelled attachment decisions as calculated by the EDUCE python package[8]. We report $F_1$ for predicting span attachments (S), span attachments with nuclearity (N), span attachments with relation labels (R) and span attachments with nuclearity and relation labels (F).

We compare our results against the numbers from Morey et al. (2017), since they include several competitive parers under a consistent evaluation scheme.[9]

As a baseline, we use a discriminative version of our model. This is a shift-reduce parser with the same EDU, unit and stack representations as our model, but with a lookahead buffer representation as well. For the buffer representation, we run a backward LSTM over the representations of the remaining EDUs in the buffer.

## 5.3 Training and Hyperparameters

We use 300-dimensional word embeddings initialized to word2vec vectors (Mikolov et al., 2013). We tie the embeddings in the EDU LSTM and the decoder LSTM input and output embeddings. We use a 2-layer bidirectional LSTM with 512-dimensional hidden state for the EDU LSTM. The TreeLSTM composition function also has a 512-dimensional (in total) hidden state with 100-dimensional relation embeddings. The stack LSTM and decoder LSTM also have 512-dimensional hidden states. For the structural features, we use 10-dimensional sentence and paragraph boundary feature embeddings and 50-dimensional dependency relation embeddings.

We train the models with Adam (Kingma and Ba, 2014) using an initial learning rate of $10^{-3}$ and default values for the other hyperparameters. We apply blank noise variational smoothing (Kong et al., 2019) with a dropout rate of 0.25 to the tied embeddings to regularize the model. In particular, for each document we sample a set of word types to drop and replace their word embeddings with the $<$ `UNK` $>$ token's word embedding.

We extract structural features using the sentence and paragraph boundary annotation in the RST-DT, and dependency trees obtained from the spaCy parser. Our models were implemented in PyTorch (Paszke et al., 2017).

## 5.4 Results

### 5.4.1 Search Comparison

Table 3 shows RST-DT development set labelled attachment metrics for our parser using word-level and bag-level beam search. Our search algorithm outperforms word-level beam search on all of the metrics across beam sizes.[10] On spans with nuclearity (N), bag-level beam search outperforms word-level beam search by 5.9% to 8.1%. This

[8] https://github.com/irit-melodi/educe

[9] We do not compare against Yu et al. (2018), Zhang et al. (2018) and Lin et al. (2019)'s recent neural RST parsers since they do not evaluate labelled attachment decisions so their results are not comparable to ours.

[10] Word-level beam search has three beam size parameters: the structural action beam size $k$, the next-word beam size $k_w$ and the fast-track beam size $k_s$. For top-down parsers, Stern et al. (2017) set $k_w = k/10$, $k_s = k/100$, but in tuning these parameters on the development set we found the best performance with $k_w = k$, $k_s = k/10$ for our parser.

| Model | S | N | R | F |
|---|---|---|---|---|
| *Feature-based parsers* | | | | |
| Hayashi et al. (2016) | 65.1 | 54.6 | 44.7 | 44.1 |
| Surdeanu et al. (2015) | 65.3 | 54.2 | 45.1 | 44.2 |
| Joty et al. (2015) | 65.1 | 55.5 | 45.1 | 44.3 |
| Feng and Hirst (2014a) | ***68.6*** | *55.9* | ***45.8*** | *44.6* |
| *Neural parsers* | | | | |
| Braud et al. (2016) | 59.5 | 47.2 | 34.7 | 34.3 |
| Li et al. (2016) | 64.5 | 54.0 | 38.1 | 36.6 |
| Braud et al. (2017) (mono) | 61.9 | 53.4 | 44.5 | 44.0 |
| *Our work* | | | | |
| Discriminative Baseline | 65.2 | 54.9 | 42.8 | 42.4 |
| Generative Model | *67.1* | ***57.4*** | *45.5* | ***45.0*** |
| *Unpublished* | | | | |
| Ji and Eisenstein (2014) (updated) | 64.1 | 54.2 | 46.8 | 46.3 |
| *Additional data* | | | | |
| Braud et al. (2017) (cross + dev) | 62.7 | 54.5 | 45.5 | 45.1 |

Table 4: Test set micro-averaged $F_1$ scores on labelled attachment decisions. We report numbers for other parsers from Morey et al. (2017)'s replication study. For each metric, the highest score for all the parsers in the comparison is shown in bold, while the highest score among parsers of that type (neural or feature-based) is in italics.

is consistent with the branching bias in word-level search leading it to return trees whose structure differs from the trees in the RST-DT. The poor performance on structure prediction also seems to have a knock-on effect on the relation and full tree prediction accuracy.

#### 5.4.2 Parsing Performance

Table 4 shows RST-DT test set labelled attachment metrics for various parsers. Our model outperforms all of the published[11] neural models that do not use additional training data[12] in Morey et al. (2017)'s replication study on all of the metrics. On span accuracy (S), we outperform all of the other parsers except for Feng and Hirst (2014a)'s graph CRF model. On spans with nuclearity (N), the equivalent of the unlabelled attachment score for discourse dependencies, we outperform all of the parsers in the study. We perform competitively on spans with relations (R), and we outperform all of the published parsers that do not use additional data on spans with nuclearity and relations (F).

Our model also outperforms the discriminative baseline using the same features and implementation on all metrics by between 1.9% and 2.7%.

[11] Ji and Eisenstein (2014) presented a transition-based parser that used continuous bag-of-words representations for EDUs and an SVM as the next action classifier. For Morey et al. (2017)'s study, they submitted predicted discourse trees from an updated, unpublished version of their parser.

[12] In the cross+dev setting, Braud et al. (2017) train their parser on RST discourse treebanks for several languages.

## 6 Conclusion

We introduced the first generative model for RST parsing. We showed that word-level beam search has a branching bias for bottom-up RNNGs which hurt performance on our task. We proposed a novel beam search algorithm that does not have this branching bias and that outperformed word-level beam search across beam sizes and with different evaluation metrics. With our search algorithm, our generative model achieved state-of-the art-level RST parsing performance, outperforming all of the published RST parsers from a recent study that do not use additional training data on labelled attachment $F_1$. Our results show that generative modelling is an effective approach to RST parsing, with superior structure prediction and competitive relation prediction performance.

## Acknowledgments

Andreas Vlachos is supported by the EPSRC grant eNeMILP (EP/R021643/1). Amandla Mabona is supported by Commonwealth and Sansom Scholarships. We thank the anonymous reviewers for their helpful comments.

## References


Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2442–2452.

Nicholas Asher. 2012. *Reference to abstract objects in discourse*, volume 50. Springer Science & Business Media.

Nicholas Asher, Nicholas Michael Asher, and Alex Lascarides. 2003. *Logics of conversation*. Cambridge University Press.

Parminder Bhatia, Yangfeng Ji, and Jacob Eisenstein. 2015. Better document-level sentiment analysis from rst discourse parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2212–2218.

Chlo Braud, Maximin Coavoux, and Anders Sgaard. 2017. Cross-lingual RST Discourse Parsing. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 292–304, Valencia, Spain. Association for Computational Linguistics.

Chlo Braud, Barbara Plank, and Anders Sgaard. 2016. Multi-view and multi-task training of RST discourse parsers. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1903–1913, Osaka, Japan. The COLING 2016 Organizing Committee.

Jan Buys and Phil Blunsom. 2018. Neural Syntactic Generative Models with Exact Marginalization. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 942–952, New Orleans, Louisiana. Association for Computational Linguistics.

Lynn Carlson and Daniel Marcu. 2001. Discourse tagging reference manual. *ISI Technical Report ISI-TR-545*, 54:56.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2003. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Current and new directions in discourse and dialogue*, pages 85–112. Springer.

Eugene Charniak et al. 2016. Parsing as language modeling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2331–2336.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent Neural Network Grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209, San Diego, California. Association for Computational Linguistics.

Vanessa Wei Feng and Graeme Hirst. 2014a. A Linear-Time Bottom-Up Discourse Parser with Constraints and Post-Editing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 511–521, Baltimore, Maryland. Association for Computational Linguistics.

Vanessa Wei Feng and Graeme Hirst. 2014b. Patterns of local discourse coherence as a feature for authorship attribution. *Literary and Linguistic Computing*, 29(2):191–198.

Elisa Ferracane, Su Wang, and Raymond Mooney. 2017. Leveraging Discourse Information Effectively for Authorship Attribution. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 584–593, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Daniel Fried, Mitchell Stern, and Dan Klein. 2017. Improving neural parsing by disentangling model combination and reranking effects. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 161–166.

John J Godfrey, Edward C Holliman, and Jane McDaniel. 1992. Switchboard: Telephone speech corpus for research and development. In *[Proceedings] ICASSP-92: 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 517–520. IEEE.

Katsuhiko Hayashi, Tsutomu Hirao, and Masaaki Nagata. 2016. Empirical comparison of dependency conversions for RST discourse trees. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 128–136, Los Angeles. Association for Computational Linguistics.

Matthew Honnibal and Ines Montani. 2017. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. *To appear*.

Yangfeng Ji and Jacob Eisenstein. 2014. Representation Learning for Text-level Discourse Parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13–24, Baltimore, Maryland. Association for Computational Linguistics.

Yangfeng Ji, Gholamreza Haffari, and Jacob Eisenstein. 2016. A latent variable recurrent neural network for discourse relation language models. In *Proceedings of NAACL-HLT*, pages 332–342.

Yangfeng Ji and Noah A Smith. 2017. Neural discourse structure for text categorization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 996–1005.

Shafiq Joty, Giuseppe Carenini, and Raymond T. Ng. 2015. CODRA: A Novel Discriminative Framework for Rhetorical Analysis. *Computational Linguistics*, 41(3):385–435.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*. ArXiv: 1412.6980.

Lingpeng Kong, Gabor Melis, Wang Ling, Lei Yu, and Dani Yogatama. 2019. Variational Smoothing in Recurrent Neural Network Language Models. *arXiv:1901.09296 [cs]*. ArXiv: 1901.09296.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A. Smith. 2017. What Do Recurrent Neural Network Grammars Learn About Syntax? In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1249–1258, Valencia, Spain. Association for Computational Linguistics.

Adhiguna Kuncoro, Chris Dyer, John Hale, Dani Yogatama, Stephen Clark, and Phil Blunsom. 2018. Lstms can learn syntax-sensitive dependencies well, but modeling structure makes them better. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436.

John D Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289. Morgan Kaufmann Publishers Inc.

Qi Li, Tianshi Li, and Baobao Chang. 2016. Discourse Parsing with Attention-based Hierarchical Neural Networks. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 362–371, Austin, Texas. Association for Computational Linguistics.

Xiang Lin, Shafiq Joty, Prathyusha Jwalapuram, and Saiful Bari. 2019. A unified linear-time framework for sentence-level discourse parsing. *arXiv preprint arXiv:1905.05682*.

William C. Mann and Sandra A. Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text-Interdisciplinary Journal for the Study of Discourse*, 8(3):243–281.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Mathieu Morey, Philippe Muller, and Nicholas Asher. 2017. How much progress have we made on RST discourse parsing? A replication study of recent results on the RST-DT. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1319–1324, Copenhagen, Denmark. Association for Computational Linguistics.

Andrew Y. Ng and Michael I. Jordan. 2002. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in neural information processing systems*, pages 841–848.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

Livia Polanyi. 1988. A formal model of the structure of discourse. *Journal of pragmatics*, 12(5-6):601–638.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind K Joshi, and Bonnie L Webber. 2008. The penn discourse treebank 2.0. In *LREC*. Citeseer.

Geoffrey Sampson. 1997. Depth in English grammar. *Journal of Linguistics*, 33(1):131–151.

Radu Soricut and Daniel Marcu. 2003. Sentence Level Discourse Parsing using Syntactic and Lexical Information. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 228–235.

Mitchell Stern, Daniel Fried, and Dan Klein. 2017. Effective inference for generative neural parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1695–1700.

Mihai Surdeanu, Tom Hicks, and Marco Antonio Valenzuela-Escarcega. 2015. Two Practical Rhetorical Structure Theory Parsers. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 1–5, Denver, Colorado. Association for Computational Linguistics.

Zhiyang Teng and Yue Zhang. 2017. Head-Lexicalized Bidirectional Tree LSTMs. *Transactions of the Association for Computational Linguistics*, 5:163–177.

Bonnie L Webber. 1988. Discourse deixis and discourse processing. *Discourse*, 9.

D Yogatama, C Dyer, W Ling, and P Blunsom. 2017. Generative and discriminative text classification with recurrent neural networks. In *Thirty-fourth International Conference on Machine Learning (ICML 2017)*. International Machine Learning Society.

Nan Yu, Meishan Zhang, and Guohong Fu. 2018. Transition-based neural rst parsing with implicit syntax features. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 559–570.

Longyin Zhang, Cheng Sun, Xin Tan, and Fang Kong. 2018. Rst discourse parsing with tree-structured neural networks. In *China Workshop on Machine Translation*, pages 15–26. Springer.

Yue Zhang and Stephen Clark. 2008. A Tale of Two Parsers: Investigating and Combining Graph-based and Transition-based Dependency Parsing Using Beam-search. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 562–571, Stroudsburg, PA, USA. Association for Computational Linguistics. Event-place: Honolulu, Hawaii.

# Weak Supervision for Learning Discourse Structure

**Sonia Badene**[1,2], **Kate Thompson**[1,2], **Jean-Pierre Lorré**[2], **Nicholas Asher**[1,3,4]

[1]IRIT, [2]Linagora, [3]CNRS, [4]ANITI
{*sonia.badene,kate.thompson,nicholas.asher*}*@irit.fr*, {*sbadene,jplorre*}*@linagora.com*

## Abstract

This paper provides a detailed comparison of a data programming approach with (i) off-the-shelf, state-of-the-art deep learning architectures that optimize their representations (BERT) and (ii) handcrafted-feature approaches previously used in the discourse analysis literature. We compare these approaches on the task of learning discourse structure for multi-party dialogue. The data programming paradigm offered by the Snorkel framework allows a user to label training data using expert-composed heuristics, which are then transformed via the "generative step" into probability distributions of the class labels given the data. We show that on our task the generative model outperforms both deep learning architectures as well as more traditional ML approaches when learning discourse structure—it even outperforms the combination of deep learning methods and handcrafted features. We also implement several strategies for "decoding" our generative model output in order to improve our results. We conclude that weak supervision methods hold great promise as a means for creating and improving data sets for discourse structure.

## 1 Introduction

In this paper, we investigate and demonstrate the potential of a weak supervision, data programming approach (Ratner et al., 2016) to the task of learning discourse structure for multi-party dialogue. We offer a detailed comparison of our data programming approach with (i) off-the-shelf, state-of-the-art deep learning architectures that optimize their representations (BERT) and (ii) handcrafted-feature approaches previously used in the discourse analysis literature. Our data programming paradigm exploits the Snorkel framework that allows a user to label training data using expert-composed heuristics, which are then transformed via the "generative step" into probability distributions of the class labels given the data. We show that the generative model produced from these heuristics outperforms both deep learning architectures as well as more traditional ML approaches when learning discourse structure by up to 20 points of F1 score; it even outperforms the combination of generative and discriminative approaches that are the foundation of the Snorkel framework. We also implement several strategies for "decoding" our generative model output that improve our results.

We assume discourse structures are dependency structures (Muller et al., 2012; Li et al., 2014) and restrict the structure learning problem to predicting edges or attachments between discourse unit (DU) pairs in the dependency graph. Although the problem of attachment is only a part of the overall task of discourse interpretation, it is a difficult problem that serves as a useful benchmark for various approaches to discourse parsing. After training a supervised deep learning algorithm to predict attachments on the STAC annotated corpus[1], we then constructed a weakly supervised learning system in which we used 10% of the corpus as a development set. Experts on discourse structure wrote a set of attachment rules, or labeling functions (LFs), and tested them against this development set. We treated the remainder of the corpus as raw/unannotated data to be automatically annotated using the data programming framework.

## 2 State of the Art

Discourse structures for texts represent causal, topical, argumentative information through what are called coherence relations. For dialogues with multiple interlocutors, extraction of their discourse structures provides useful semantic infor-

[1]https://www.irit.fr/STAC/



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2296–2305, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

mation to the "downstream" models used, for example, in the production of intelligent meeting managers or the analysis of user interactions in online fora. However, despite considerable efforts to retrieve discourse structures automatically (Fisher and Roark, 2007; Duverle and Prendinger, 2009; Li et al., 2014; Joty et al., 2013; Ji and Eisenstein, 2014; Yoshida et al., 2014; Li et al., 2014; Surdeanu et al., 2015), we are still a long way from usable discourse models, especially for dialogue. Standard supervised models struggle to capture the sparse attachments, even when relatively large annotated corpora are available. In addition, the annotation process is time consuming and often fraught with errors and disagreements, even among expert annotators. This motivated us to explore the data programming approach that exploits expert linguistic knowledge in a more compact and consistent rule based form.

Given our interest in the analysis of multi-party dialogues, we used the STAC corpus of multi-party chats, an initial version of which is described in (Afantenos et al., 2015; Perret et al., 2016). In all versions of this corpus, dialogue structures are directed acyclical graphs (DAGs) formed according to SDRT[2] (Asher and Lascarides, 2003; Asher et al., 2016). An SDRT discourse structure is a graph, $\langle V, E_1, E_2, \ell, \text{Last}\rangle$, where: $V$ is a set of nodes or Discourse Units (DUs); $E_1 \subseteq V^2$ is a set of edges between DUs representing coherence relations; $E_2 \subseteq V^2$ represents a dependency relation between DUs; $\ell: E_1 \rightarrow R$ is a labeling function that assigns a semantic type to an edge in $E_1$ from a set $R$ of discourse relation types, and *Last* is a designated element of $V$ giving the last DU relative to textual or temporal order. $E_2$ is used to represent Complex Discourse Units (CDUs), which are clusters of two or more DUs connected as an ensemble to other DUs in the graph. As learning this type of recursive structure presents difficulties beyond the scope of this paper, we followed a "flattening" strategy similar to (Muller et al., 2012) to remove CDUs. This process yields a set $V*$, which is $V$ without CDUs, and a set $E*_1$, a flattened version of $E_1$.

Building these structures typically requires three steps: (i) segmenting the text into the basic units of the discourse, typically clauses - these are EDUs or *Elementary Discourse Units*; these, together with CDUs, form the set of nodes $V$ in the graph; (ii) predicting the attachments between DUs, i.e. to identify the elements in $E_1$; (iii) predicting the semantic type of the edge in $E_1$. This paper focuses on step (ii). Our dialogue structures are thus of the form $\langle V*, E*_1, \text{Last}\rangle$. Step (ii) is a difficult problem for automatic processing because attachments are theoretically possible between any two DUs in a dialogue or text, and often graphs include long-distance relations. Muller et al. (2012) is the first paper we know of that focuses on the discourse parsing attachment problem, albeit for monologue. It targeted a restricted version of an SDRT graph and trains a simple MaxEnt algorithm to produce probability distributions over pairs of EDUs, what we call a "local model" with a positive F1 attachment score of 0.635. They further applied global decoding constraints to produce a slight improvement in attachment scores (these are discussed in more detail in Section 5). Afantenos et al. (2015) used a similar strategy for dialogue on an early version of the STAC corpus. Perret et al. (2016) targeted a more elaborate approximation of SDRT graphs on the same version of the STAC corpus and reported a local model F1 attachment of 0.483. They then used Integer Linear Programming (ILP) to encode global decoding constraints particular to SDRT to improve the F1 attachment score to 0.689.

Having sketched recent progress in discourse parsing, we briefly turn to the state of the art concerning data programming. Ratner et al. (2016) introduced the data programming paradigm, along with a framework, Snorkel (Ratner et al., 2017), which uses a weak supervision method (Zhou, 2017), to apply labels to large data sets by way of heuristic labeling functions that can access distant, disparate knowledge sources. These labels are then used to train classic data-hungry machine learning (ML) algorithms. The crucial step in the data programming process uses a generative model to unify the noisy labels by generating a probability distribution for all labels for each data point. This set of probabilities replaces the ground-truth labels in a standard discriminative model outfitted with a noise-aware loss function and trained on a sufficiently large data set.

## 3 The STAC Annotated Corpus

### 3.1 Overview

While earlier versions only included linguistic moves by players, STAC now contains in addi-

[2]Segmented Discourse Representation Theory

tion a multimodal corpus of multi-party chats between players of an online game (Asher et al., 2016; Hunter et al., 2018). It includes 2,593 dialogues (each with a weakly connected DAG discourse structure), 12,588 "linguistic" DUs, 31,811 "non-linguistic" DUs and 31,251 semantic relations. A dialogue begins at the beginning of a player's turn, and ends at the end of that player's turn. In the interim, players can bargain with each other or make spontaneous conversation. These player utterances are the "linguistic" turns. In addition the corpus contains information given visually in the game interface but transcribed in the corpus into Server or interface messages, "non-linguistic" turns (Hunter et al., 2018). All turns are segmented into DUs, and these units are then connected by semantic relations.

Each dialogue represents a complete conversation. There are typically many such conversations, each beginning with a non-linguistic turn in which a player is designated to begin negotiations (see Figure 1). The dialogues end when this player performs a non-linguistic action that signals the end of their turn. The dialogues are the units on which we build a complete discourse structure.

The STAC multimodal corpus is divided into a development, train and test set. The development and test sets are each 10% of the total size of the corpus.

To compare our approach to earlier efforts, we also used the corpus from (Perret et al., 2016). This corpus was also useful to check for over fitting of our Generative model developed on the multi-modal data. The corpus from (Perret et al., 2016) is an early version of a "linguistic only" version of the STAC corpus. It contains no non-linguistic DUs, unlike the STAC multimodal corpus.[3] It also contains quite a few errors; for example, about 60 stories in the (Perret et al., 2016) dataset have no discourse structure in them at all and consist of only one DU. We eliminated these from the Perret 2016 data set that we used in our comparative experiments below, as these stories were obviously not a correct representation of what was going on in the game at the relevant point.

[3] There is also on the STAC website an updated linguistic only version of the STAC corpus. It has 1,091 dialogues, 11,961 linguistic only DUs and 10,191 semantic relations. We have not reported results on that data set here. The dataset from (Perret et al., 2016) is similar to our linguistic only STAC corpus but is still substantially different and degraded in quality. Asher et al. (2016) report significant error rates in annotation on the earlier versions of the STAC corpus and that the current linguistic only corpus of STAC offers an improvement over the (Perret et al., 2016) corpus.

### 3.2 Data Preparation

To concentrate on the attachment task, we implemented the following simplifying measures on the STAC corpus:

1. Roughly 56% of the dialogues in the corpus contain only non-linguistic DUs. The discourse structure of these dialogues is more regular and thus less challenging; so we ignore these dialogues for our prediction task.

2. 98% of the discourse relations in our development corpus span 10 DUs or less. To reduce class imbalance, we restricted the relations we consider to a distance of $\leq 10$.

3. Following (Muller et al., 2012; Perret et al., 2016) we "flatten" CDUs by connecting all relations incoming or outgoing from a CDU to the "head" of the CDU, or its first DU.

The STAC corpus as we use it in our learning experiments thus includes 1,130 dialogues, 13,734 linguistic DUs, 18,767 non-linguistic DUs and 22,098 semantic relations.

We also performed these operations on our version of the linguistic only corpus used by (Perret et al., 2016).

## 4 Data Programming Experiments

### 4.1 Candidates and Labeling Functions

The Snorkel implementation of the data programming paradigm inspired our weak supervision approach. We first identified and extracted candidates from the data, and then wrote a set of labeling functions (LFs) to apply to the candidates while only consulting the development set, the 10% of the STAC corpus set aside to develop and test our LFs. We treated the training set (80% of the corpus) as unseen/unlabeled data to which we applied the finished LFs and the models. The last 10% of the STAC corpus was reserved as a final test set.

Candidates are the units of data for which labels are predicted. For this study, the candidates are all DU pairs which could possibly be connected by a semantic relation. We use our own method to create candidates from the DUs culled from the texts of the dialogues, making sure to limit the pairs to

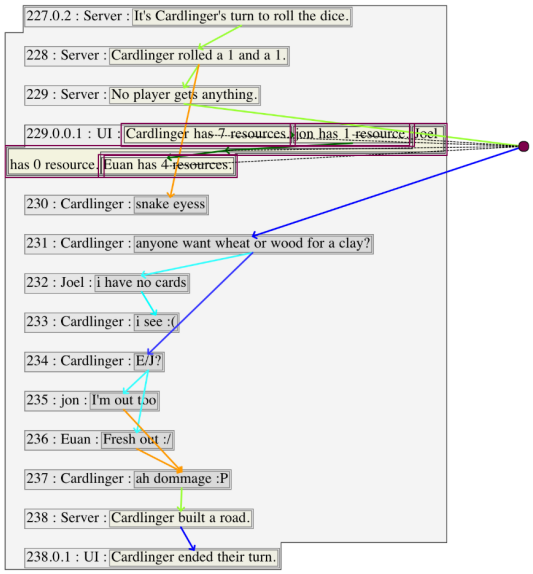


Figure 1: A dialogue from the STAC multi-modal corpus featuring player ('linguistic') turns and Server/UI ('non-linguistic') turns. There are four instances of *Result* relations, shown in green. *Result* connects a cause to its effect, i.e., the main eventuality of the first argument is understood to cause the eventuality given by the second.

```
RULE 1
IF source type == non-linguistic and target type == non-
linguistic:
    IF ((source == turn to roll the dice
          AND target == rolls the dice)
        OR
        (source == rolls the dice
          AND target == no player gets anything))

       AND the source is not the source of a different
Result relation

       AND the target is not the target of a different
Result relation

    THEN attach == 1
```

*Returns attach == 1 for candidates (227.0.2, 228) and (228, 229) but not (227.0.2, 229)*

```
RULE 2
IF source type == linguistic and target type == non-
linguistic:
    IF source contains
'sorry'/'ah'/'no'/'ouch'/'woo'/'whoa'/'bugger'…

       AND target == action taken by source player

       AND the distance between source and target <= 9

    THEN attach == 1
```

*Returns attach == 1 for candidates (237, 238) but also, incorrectly, (237, 239)*

Figure 2: Glosses of two rules that are meant to capture the *Result* relations when applied to the candidates as pulled from the corpus, exemplified in Figure 1. Although the rules are written to capture specific relation types between segments, they return 1/0 for attached/not attached

those that occur in the same dialogues. We also ruled out the possibility of backwards relations between two DUs which have different speakers: it is linguistically impossible for a speaker of, say, an assertion $d_1$ at time $t_1$ to answer a question $d_2$, asked by a different speaker at time $t_2 > t_1$, i.e. before $d_2$ was asked. That is, we include $(d_1, d_2)$ in our candidates but rule out $(d_2, d_1)$.

LFs are expert-composed functions that make an attachment prediction for a given candidate: each LF returns a 1, a 0 or a -1 ("attached"/"do not know"/"not attached") for each candidate. However, each of our LFs is written and evaluated with a specific relation type *Result*, *Question-answer-pair (QAP)*, *Continuation*, *Sequence*, *Acknowledgement*, *Conditional*, *Contrast*, *Elaboration* and *Comment* in mind. In this way, LFs leverage a kind of type-related information, which makes sense from an empirical perspective as well as an epistemological one. An attachment decision concerning two DUs is tightly linked to the type of relation relating the DUs: when an annotator decides that two DUs are attached, he or she does so with some knowledge of what type of relation attaches them. Figure 2 shows a sample LFs used for attachment prediction with the *Result* relation in mind.

LFs also exploit information about the DUs' linguistic or non-linguistic status, the dialogue acts they express, their lexical content, grammatical category and speaker, and the distance between them—features also used in supervised learning methods (Perret et al., 2016; Afantenos et al., 2015; Muller et al., 2012). Finally, we fix the order in which each LF "sees" the candidates such that it considers adjacent DUs before distant DUs. This allows LFs to exploit information about previously predicted attachments and dialogue history in new predictions. Our rule set and their description are available here: https://tizirinagh.github.io/acl2019/. Figure 2 gives an example of a labeling function that we used.

### 4.2 The Generative Model

Once the LFs are applied to all the candidates, we have a matrix of labels ($\overline{\Lambda}$) given by each LF $\Lambda$ for each candidate. The generative model, GEN, as specified in (1), provides a general distribution of marginal probabilities relative to n accuracy dependencies $\phi_j(\Lambda_i, y_i)$ for an LF $\lambda_j$ with respect inputs $x_i$, the LF's outputs on i $\Lambda_{ij}$ and true labels

$y_i$ that depend on parameters $\theta_j$ where:

$$\phi_j(\Lambda_i, y_i) := y_i \Lambda_{ij}$$

$$p_\theta(\Lambda, Y) \propto exp(\sum_{i=1}^{m} \sum_{j=1}^{n} \theta_j \phi_j(\Lambda_i, y_i)) \quad (1)$$

The parameters are estimated by minimizing the negative log marginal likelihood of the output of an observed matrix $\overline{\Lambda}$ as in (2).

$$argmin_\theta - log \sum_{Y} p_\theta(\overline{\Lambda}, Y) \quad (2)$$

GEN does not have access to the gold labels on the Training set but uses the Training set as unlabelled data. So in this model, the true class labels $y_i$ are latent variables that generate the labeling function outputs, which are estimated via Gibbs sampling over the Training set (80% of the STAC corpus), after it has been labeled by the LFs. The objective in (2) is then optimized by interleaving stochastic gradient descent steps with Gibbs sampling ones. For each candidate, GEN thus uses the accuracy measures for the LFs in (1) to assign marginal probabilities that two DUs are attached.

GEN estimates the accuracy of each LF, a marginal probability for each label, and consequently a probability for positive attachment. In this model, the true class labels $y_i$ are latent variables that generate the labeling function outputs. The model in (1) presupposes that the LFs are independent, but this assumption does not always hold: one LF might be a variation of another or they might depend on a common source of information (Mintz et al., 2009). If we don't take these dependencies into account, we risk assigning incorrect accuracies to the LFs. Snorkel provides a more complex model that automatically calculates the dependencies between LFs and marginal probabilities which we use for the generative step (Bach et al., 2017). The higher order dependencies significantly improved the generative model's results on the full STAC corpus (see Table 1).

When we obtain the results from the generative model GEN, we choose on the development corpus a threshold to apply to these marginals by calculating the threshold that gives us the best F1 score. The best threshold is 0.85 ($p > .85$ for positive attachment) in the STAC corpus. Figure 3 shows the probability distribution, on which even taking 0.8 as a threshold gives a lower F1

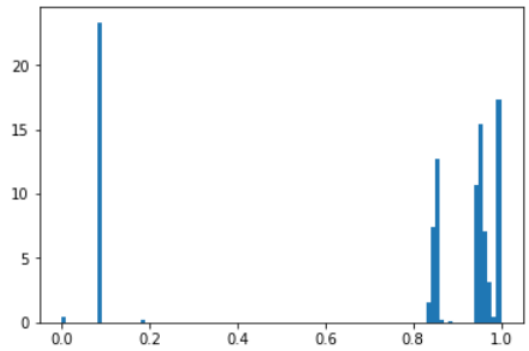


Figure 3: Probability distribution for positive attachment in the STAC development set.

score because of false positive attachment. Binarizing these marginals allows us to pass these binarized probabilities to the discriminative model. This also allows us to evaluate GEN with respect to gold label "attachment" 0/"non attachment" 1 on the STAC test data.

The generative model GEN shares with other "local" models the feature that it considers pairs of DUs in isolation of the whole structure. However, unlike other local models, our LFs enable GEN to exploit prior decisions on pairs of DUs, and thus we exploit more contextual information about discourse structure in GEN than in our classical, supervised local models. In addition, GEN uses the Training set very differently from classical, supervised models.

### 4.3 Discriminative Model

The standard Snorkel approach inputs the marginal probabilities from the generative step directly into a discriminative model, which is trained on those probabilities using a *noise-aware loss function* (Ratner et al., 2016). Ideally, this step generalizes the LFs by augmenting the feature representation - from, say, dozens of LFs to a high dimensional feature space - and allows the model to predict labels for more new data. Thus the precision potentially lost in the generalization is offset by a larger increase in recall.

We tested three discriminative models in our study. Each one was trained on the gold labeled data in the Training set of the STAC corpus. First we tried a single layer BI-LSTM with 300 neurons, which takes as input 100 dimensional-embeddings for the text of each DU in the candidate pair. We concatenated the outputs of the BI-LSTM and fed them to a simple perceptron with one hidden layer and Rectified Linear Unit

| Generative Model on dev set | Precision | Recall | F1 score | Accuracy |
|---|---|---|---|---|
| Without higher order dependencies | 0.45 | 0.70 | 0.55 | 0.87 |
| With higher order dependencies | 0.68 | 0.67 | 0.67 | 0.92 |

Table 1: Evaluations of attachment on Dev set with and without higher order dependencies.

(ReLU) activation (Hahnloser et al., 2000; Jarrett et al., 2009; Nair and Hinton, 2010) and optimized with Adam (Kingma and Ba, 2014). Given that our data is extremely unbalanced in favor of the "unattached" class ("attached" candidates are roughly 13% of the candidates on the development set), we also implemented a class-balancing method inspired by (King and Zeng, 2001) which maps class indices to weight values used for weighting the loss function during training.

We also implemented BERT (Devlin et al., 2018)'s sequence classification model (source code on the link below[4]) with 10 training epochs and all default parameters otherwise. BERT, the Bidirectional Encoder Representations from Transformers, is a text encoder pre-trained using language models where the system has to guess a missing word or word piece removed at random from the text. Originally designed for automatic translation tasks, BERT uses bi-directional self-attention to produce the encodings and performs at the state of the art on many textual classification tasks.

In order to use these methods, we had to binarize the marginal probabilities before moving to the discriminative step, using a threshold of $p > .85$ as explained in Section 4.2. Though this marks a departure from the standard Snorkel approach, we found that our discriminative model results were higher when the marginals were binarized and when the class re-balancing was used, albeit much lower than expected overall.

Finally, to facilitate comparison with earlier work, we also implemented a local model, LogReg* as mentioned in the following of the paper, that used marginal probabilities together with handcrafted features (the feature set used in (Afantenos et al., 2015) listed in their Table 2) and a Logistic Regression classifier.

[4]Link to BERT sequence classification model code: https://github.com/huggingface/pytorch-pretrained-BERT/blob/master/examples/run_classifier.py

## 5 Decoding

A set of highly accurate predictions for individual candidates does not necessarily lead to accurate discourse structures; for instance, without global structural constraints, GEN and local models may not yield the directed acyclic graphs (DAGs), required by SDRT. As in previous work (Muller et al., 2012; Afantenos et al., 2015; Perret et al., 2016), we use the Maximum Spanning Tree (MST) algorithm, and a variation thereof, to ensure that the dialogue structures predicted conform to some more general structural principle. We implemented the Chu-Liu-Edmonds algorithm (Chu, 1965; Edmonds, 1967), an efficient method of finding the highest-scoring non-projective tree in a directed graph, as described in Jurafsky and Martin[5]. The algorithm greedily selects the relations with the highest probabilities from the dependency graphs produced by the local model, then removes any cycles. The result is a tree structure with one incoming relation per node. In cases of nodes with multiple equiprobable incoming relations, the algorithm takes whichever relation it sees first.

Since SDRT structures can contain nodes with multiple incoming relations, i.e. are not always tree-like, we altered the MST algorithm in the manner of (Muller et al., 2012; Afantenos et al., 2015; Perret et al., 2016), forcing the MST to include all high-probability incoming relations which do not create cycles. This produces MS-DAG structures which are in principle more faithful to SDRT. In addition, since discourse attachments in general follow an inverse power law (many short-distance attachments and fewer long-distance attachments), we implemented two MST/MS-DAG variants that always choose the shortest relation among multiple high-probability relations (MST/short and MS-DAG/short).

## 6 Results and Analysis

We set out to test the performance of combinations of generative and discriminative models along the

[5]https://web.stanford.edu/ jurafsky/slp3/13.pdf

lines of the data programming paradigm on the task of dialogue structure prediction in order to automatically generate SDRT corpus data. While our results were consistent with what data programming promises—more data with accuracy comparable if not slightly below that of hand-labeled data—our most surprising and interesting result was the performance of the generative model on its own. As seen in Table 2 on STAC test data, GEN dramatically outperformed our deep learning baselines—BiLSTM, BERT, and BERT + LogReg* architectures on gold labels—as well as the LAST baseline, which attaches every DU in a dialogue to the DU directly preceding it. In addition, stand alone GEN also outperformed all the coupled Snorkel models, in which GEN is combined with an added discriminative step, by up to a 30 point improvement in F1 score (GEN vs. GEN+BiLSTM). We did not expect this, given that adding a discriminative model in Snorkel is meant to generalize, and hence improve, what GEN learns.

Critical to this success was the inclusion of higher order dependencies in GEN and the fact that our LFs exploited contextual information about the DUs that was unavailable to the deep learning models or even the handcrafted feature model. GEN beats all competitors in terms of F1 score while taking a fraction of the annotated data to develop and train the model, showing the power and promise of the generative model.

One might wonder whether GEN and discriminative models are directly comparable. Generative machine learning algorithms learn the joint probability of X and Y, whereas discriminative algorithms learn the conditional probability of Y given X. Nevertheless, when we exploit the generative model we are trying to find the $Y$ for which $P(X \wedge Y)$ is maximized. In effect we are producing, *though not learning*, a conditional probability. So it makes sense to compare our generative model's output with that of other, discriminative machine learning approaches.

We also got surprising results concerning the supervised model benchmarks. Table 2 shows that LogReg* was the best supervised learning method on the STAC data in terms of producing local models. This is evidence that hand crafted features capturing non local information about a DU's contexts do better than all purpose contextual encodings from neural nets at least on this task. We also implemented BERT+LogReg*, a learning algorithm that uses BERT's encodings together with a Logistic Regression classifier trained on STAC's gold data with handcrafted features from (Afantenos et al., 2015) and used in (Perret et al., 2016). BERT+LogReg* outputs a local model that improves upon BERT's local model, but it did not do as well as LogReg* on its own (let alone GEN), suggesting that BERT's encodings actually interfered with the correct predictions.

We also investigated GEN coupled with various discriminative models to test the standard Snorkel. As we remarked above, we found that binarizing GEN's output improved the performance of the coupled discriminative model, so Table 2 only reports scores for various GEN-coupled discriminative models that take the binarized GEN predictions as input. In keeping with our analysis of the supervised benchmarks, we found that the best discriminative model to couple with GEN in the Snorkel architecture was LogReg*, far outperforming GEN with either a BiLSTM or BERT on the STAC test set. Its results were only slightly less good than those of stand alone GEN.

To further investigate comparisons between different architectures for solving the attachment problem, we compared various local models extended with the MS-DAG decoding algorithm discussed in Section 5, giving the global results shown in the righthand columns of Tables 3 and 4. With MS-DAG added, GEN continued to outperform all other approaches on STAC data. In Table 5, we experimented with adding all decoding algorithms to the local GEN result. This gave a boost in F1 score—2 points with classic MST and MS-DAG and 4 points with the variants favoring relation instances with shorter attachments.

It is not surprising that MST improves the GEN results, since it eliminates some of the false positive relations that pass the generative threshold and includes some of the false negative relations that fall below the threshold. The general inverse power law distribution of discourse attachments explains the good performance of the MST shortest link variant. GEN + "MST-short" has the highest attachment score of all approaches to the problem of attachment in the literature (Morey et al., 2018), though we are cautious in comparing scores for systems applied to different corpora.

Finally, we wanted to see how GEN and our other models fared on our version of the (Perret

| | Precision | Recall | F1 score | Accuracy |
|---|---|---|---|---|
| SUPERVISED BASELINES | | | | |
| LAST | 0.54 | 0.55 | 0.55 | 0.84 |
| BiLSTM on Gold labels | 0.33 | 0.80 | 0.47 | 0.75 |
| BERT on Gold labels | 0.56 | 0.48 | 0.52 | 0.88 |
| LogReg* on Gold labels | 0.73 | 0.52 | 0.61 | 0.91 |
| BERT+LogReg* on Gold labels | 0.59 | 0.49 | 0.53 | 0.89 |
| SNORKEL PIPELINE | | | | |
| GEN + Disc (BiLSTM) | 0.28 | 0.59 | 0.38 | 0.74 |
| GEN + Disc (BERT) | 0.49 | 0.40 | 0.44 | 0.86 |
| GEN + Disc (LogReg*) | 0.68 | 0.65 | 0.67 | 0.91 |
| GENERATIVE STAND ALONE | | | | |
| GEN | **0.69** | **0.66** | **0.68** | **0.92** |
| GEN + MST-short | **0.73** | **0.71** | **0.72** | **0.93** |

Table 2: Evaluations of weakly supervised (Snorkel and stand alone GEN) and supervised approaches on STAC data.

| | **Local Model** | | | **Global Model** | | |
|---|---|---|---|---|---|---|
| | Precision | Recall | F1 | Precision | Recall | F1 |
| GEN | **0.69** | **0.66** | **0.68** | **0.71** | **0.69** | **0.70** |
| LogReg* on STAC | 0.73 | 0.52 | 0.61 | 0.65 | 0.64 | 0.65 |

Table 3: Comparison of local and global models on STAC data.

et al., 2016) data set. Comparing GEN to LogReg* on our version of the (Perret et al., 2016) data set, GEN has higher scores than LogReg*'s local model; but with a decoding mechanism similar to that reported in (Perret et al., 2016), LogReg*'s global model significantly improves over the GEN's. We see a 6 point loss in F1 score on GEN's global model relative to LogReg*'s, even though both used identical MST decoding mechanisms. This is what one would expect from a Snorkel based architecture, although it's not the rule that we observed for GEN. The only reason GEN did not beat LogReg* is that it did not get a sufficient boost from decoding. We think that this happened because our LFs already contain a lot of global information about the discourse structure, which meant that MST had less of an effect.

Note, however, that even on the (Perret et al., 2016) data set, the MST decoding mechanism provided LogReg* only a boost of 12 F1 points, as seen in Table 4, which is significantly lower than what is reported in (Perret et al., 2016). This 12% boost is the upper limit for boosts with MST that we were able to reproduce. This could be a result of our eliminating the degraded one EDU stories from the data set.

## 7 Conclusions and Future Work

We have compared a weak supervision approach, inspired by Snorkel, with a standard supervised model on the difficult task of discourse attachment. The results of the model from Snorkel's generative step surpass those of a standard supervised learning approach, proving it more than competitive with standard approaches. It can also generate a lot of annotated data in a very short time relative to what is needed for a traditional approach: Asher et al. (2016) state that the STAC corpus took at least 4 years to build; we created and refined our labeling functions in two months. In addition, a big advantage of the generative model from the learning point of view is that we don't have class balancing or drowning problems, which plague problems like discourse attachment. This is because the generative model's predictions are generated in a very different way from those of a discriminative model, which are based on inductive generalizations. Still it is clear that we must further investigate the interaction of the generative and discriminative models in order to eventually leverage the power of generalization that a discriminative model is supposed to afford.

In future work, we will enrich our weak supervi-

| | Local Model | | | Global Model | | |
|---|---|---|---|---|---|---|
| | Precision | Recall | F1 | Precision | Recall | F1 |
| Perret et al. 2016 | 0.66 | 0.38 | 0.48 | **0.68** | **0.65** | **0.67** |
| LogReg* | 0.64 | 0.39 | 0.48 | 0.59 | 0.61 | 0.60 |
| BERT | 0.43 | 0.31 | 0.36 | 0.46 | 0.47 | 0.46 |
| GEN | **0.46** | **0.65** | **0.54** | 0.53 | 0.54 | 0.54 |
| GEN + LogReg* | 0.46 | 0.65 | 0.54 | 0.53 | 0.54 | 0.53 |

Table 4: Comparison of Perret 2016 and results of our methods on Perret 2016 data.

| | F1 scores of MST variants using GEN inputs | | | |
|---|---|---|---|---|
| | MS-DAG | MST | MST/short | MS-DAG/short |
| F1 Score: | 0.70 | 0.70 | 0.72 | 0.72 |

Table 5: F1 scores for different decoding mechanisms with GEN on STAC data.

sion system by giving the LFs access to more sophisticated contexts that take into account global structuring constraints in order to see how they compare to the simple, exogenous decoding constraints like MST. We think we can put much more sophisticated decoding constraints that don't just work off local probabilities of arcs but on the full information about the arc in its discourse context. This will lead us to understand how weakly supervised methods can effectively capture the global structural constraints on discourse structures directly without decoding or elaborate learning architectures.

## References


Stergos Afantenos, Eric Kow, Nicholas Asher, and Jérémy Perret. 2015. Discourse parsing for multiparty chat dialogues. In *Association for Computational Linguistics (ACL)*.

Nicholas Asher, Julie Hunter, Mathieu Morey, Farah Benamara, and Stergos D Afantenos. 2016. Discourse structure and dialogue acts in multiparty dialogue: the stac corpus. In *LREC*.

Nicholas Asher and Alex Lascarides. 2003. *Logics of conversation*. Cambridge University Press.

Stephen H Bach, Bryan He, Alexander Ratner, and Christopher Ré. 2017. Learning the structure of generative models without labeled data. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 273–282. JMLR. org.

Yoeng-Jin Chu. 1965. On the shortest arborescence of a directed graph. *Scientia Sinica*, 14:1396–1400.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

David A Duverle and Helmut Prendinger. 2009. A novel discourse parser based on support vector machine classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 665–673. Association for Computational Linguistics.

Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the national Bureau of Standards B*, 71(4):233–240.

Seeger Fisher and Brian Roark. 2007. The utility of parse-derived features for automatic discourse segmentation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 488–495.

Richard HR Hahnloser, Rahul Sarpeshkar, Misha A Mahowald, Rodney J Douglas, and H Sebastian Seung. 2000. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405(6789):947.

Julie Hunter, Nicholas Asher, and Alex Lascarides. 2018. A formal semantics for situated conversation. *Semantics and Pragmatics*, 11. DOI: http://dx.doi.org/10.3765/sp.11.10.

Kevin Jarrett, Koray Kavukcuoglu, Yann LeCun, et al. 2009. What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th International Conference on Computer Vision (ICCV)*, pages 2146–2153. IEEE.

Yangfeng Ji and Jacob Eisenstein. 2014. Representation learning for text-level discourse parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 13–24.

Shafiq Joty, Giuseppe Carenini, Raymond Ng, and Yashar Mehdad. 2013. Combining intra-and multi-sentential rhetorical parsing for document-level discourse analysis. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 486–496.

Gary King and Langche Zeng. 2001. Logistic regression in rare events data. *Political analysis*, 9(2):137–163.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Sujian Li, Liang Wang, Ziqiang Cao, and Wenjie Li. 2014. Text-level discourse dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 25–35. Association for Computational Linguistics.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Mathieu Morey, Philippe Muller, and Nicholas Asher. 2018. A dependency perspective on rst discourse parsing and evaluation. *Computational Linguistics*, pages 198–235.

Philippe Muller, Stergos Afantenos, Pascal Denis, and Nicholas Asher. 2012. Constrained decoding for text-level discourse parsing. *Proceedings of COLING 2012*, pages 1883–1900.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

Jérémy Perret, Stergos Afantenos, Nicholas Asher, and Mathieu Morey. 2016. Integer linear programming for discourse parsing. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 99–109.

Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid training data creation with weak supervision. *Proceedings of the VLDB Endowment*, 11(3):269–282.

Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. 2016. Data programming: Creating large training sets, quickly. In *Advances in neural information processing systems*, pages 3567–3575.

Mihai Surdeanu, Tom Hicks, and Marco Antonio Valenzuela-Escarcega. 2015. Two practical rhetorical structure theory parsers. In *Proceedings of the 2015 conference of the North American chapter of the association for computational linguistics: Demonstrations*, pages 1–5.

Yasuhisa Yoshida, Jun Suzuki, Tsutomu Hirao, and Masaaki Nagata. 2014. Dependency-based discourse parser for single-document summarization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1834–1839.

Zhi-Hua Zhou. 2017. A brief introduction to weakly supervised learning. *National Science Review*, 5(1):44–53.

# Predicting Discourse Structure using Distant Supervision from Sentiment

**Patrick Huber and Giuseppe Carenini**
Department of Computer Science
University of British Columbia
Vancouver, BC, Canada, V6T 1Z4
{huberpat, carenini}@cs.ubc.ca

## Abstract

Discourse parsing could not yet take full advantage of the neural NLP revolution, mostly due to the lack of annotated datasets. We propose a novel approach that uses distant supervision on an auxiliary task (sentiment classification), to generate abundant data for RST-style discourse structure prediction. Our approach combines a neural variant of multiple-instance learning, using document-level supervision, with an optimal CKY-style tree generation algorithm. In a series of experiments, we train a discourse parser (for only structure prediction) on our automatically generated dataset and compare it with parsers trained on human-annotated corpora (news domain RST-DT and Instructional domain). Results indicate that while our parser does not yet match the performance of a parser trained and tested on the same dataset (*intra-domain*), it does perform remarkably well on the much more difficult and arguably more useful task of *inter-domain* discourse structure prediction, where the parser is trained on one domain and tested/applied on another one.

## 1 Introduction

Discourse parsing is a fundamental NLP task known to enhance key downstream tasks, such as sentiment analysis (Bhatia et al., 2015; Nejat et al., 2017; Hogenboom et al., 2015), text classification (Ji and Smith, 2017) and summarization (Gerani et al., 2014).

In essence, a discourse parser should reveal the structure underlying coherent text as postulated by a discourse theory, of which the two most popular are Rhetorical Structure Theory (RST) (Mann and Thompson, 1988) and PDTB (Prasad et al., 2008). In this paper, we focus on RST-style parsing, but the proposed approach is theory agnostic and could be applied to PDTB as well.

The RST discourse theory assumes a complete hierarchical discourse tree for a given document, where leaf nodes are clause-like sentence fragments, called elementary-discourse-units (EDUs), while internal tree nodes are labelled with discourse relations. In addition, each node is given a nuclearity attribute, which encodes the importance of the node in its local context.

In the past decade, traditional, probabilistic approaches, such as Support Vector Machines (SVM) (Hernault et al., 2010; Ji and Eisenstein, 2014) and Conditional Random Fields (CRF) (Joty et al., 2015; Feng and Hirst, 2014), have dominated the field. More recently, neural approaches (Braud et al., 2016; Li et al., 2016; Braud et al., 2017; Yu et al., 2018; Liu and Lapata, 2017, 2018) have been explored, with limited success (Morey et al., 2017; Ferracane et al., 2019). The main reason why recent advances in deep learning have not enhanced discourse parsing to the same extend as they have revolutionized many other areas of NLP is the small amount of training data available. Existing corpora in English (Carlson et al., 2002; Subba and Di Eugenio, 2009) only comprise of a few hundred annotated documents, each typically containing a few dozen EDUs, strictly limiting the application of deep learning methodologies. Although in principle new corpora could be created, the annotation process is expensive and time consuming. It requires sophisticated linguistic expertise and is therefore not suitable for crowd-sourcing efforts.

Another limiting issue with the available training data is the restriction to only a few domains, such as news articles (Carlson et al., 2002) or instructions (Subba and Di Eugenio, 2009). This impairs the performance of existing discourse parsers when transferred into new domains.

To overcome the mentioned limitations, we propose a novel approach that uses distant supervi-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2306–2316,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

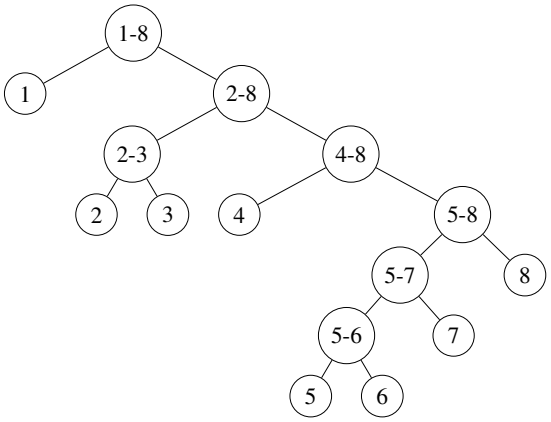


Figure 1: Example output of a strongly negative restaurant review in the Yelp'13 corpus:
[Panera bread wannabes.]$_1$ [Food was okay and coffee]$_2$ [was eh.]$_3$ [Not large portions for the price.]$_4$ [The free chocolate chip cookie was a nice touch]$_5$ [and the orange scone was good.]$_6$ [Broccoli cheddar soup was pretty good.]$_7$ [I would not come back.]$_8$

sion on the auxiliary task of sentiment classification to generate abundant data for RST-style discourse structure prediction.

We draw intuition from previous work using discourse parsing as an auxiliary task to enhance sentiment analysis (Bhatia et al., 2015; Nejat et al., 2017; Hogenboom et al., 2015). Our assumption is that such synergies between sentiment analysis and discourse parsing are bidirectional. In this paper, we leverage the synergy effects in the opposite direction by using sentiment analysis to create discourse structures.

Figure 1 illustrates the discourse structure of a strongly negative Yelp review generated by our system. While solely based on sentiment information, the structure nevertheless resembles a well aligned discourse tree. It can be observed that EDUs with negative sentiment are generally located at a higher level in the tree, while for example EDUs ⑤ and ⑥, with positive sentiment, are at the bottom of a deep subtree. This way, EDUs with negative sentiment strongly influence the overall sentiment, while EDUs ⑤ and ⑥ only have little impact. At the same time, semantically related EDUs generally have a shorter distance than semantically unrelated EDUs.

Our approach combines a neural variant of multiple-instance learning (MILNet) (Angelidis and Lapata, 2018), with an optimal CKY-style tree generation algorithm (Jurafsky and Martin, 2014). First, MILNet computes fine-grained sentiment values and un-normalized attention scores (Ji and Smith, 2017) on EDU-level, by solely relying on distant supervision signals from document-level annotations. These annotations are abundantly available from several published open source datasets such as Yelp'13 (Tang et al., 2015), IMDB (Diao et al., 2014) or Amazon (Zhang et al., 2015). Then, the sentiment values and attention scores are aggregated to guide the discourse-tree construction, optimized on the document gold-label sentiment, using optimal CKY-style parsing.

Following this approach, we generate a new corpus annotated with "silver standard" discourse trees, which comprises of 100k documents (two orders of magnitude more than any existing corpora). To test the quality of our new corpus, we run a series of experiments, where we train the top performing discourse parser by Wang et al. (2017) on our corpus for discourse structure prediction and compare it with the same parser trained on human annotated corpora in the news domain (RST-DT) and in the instructional domain. Results indicate that while training a parser on our corpus does not yet match the performance of a parser trained and tested on the same dataset (*intra-domain*), it does perform remarkably well on the significantly more difficult and arguably more useful task of *inter-domain* discourse structure prediction, where the parser is trained on one domain and tested/applied on another one. Our results on *inter-domain* discourse parsing, shown in Section 4, strongly suggest that if anyone wants to leverage discourse parsing in a domain without annotated data, it is advantageous to use a discourse parser which has been trained on our new corpus, rather than, for instance, on RST-DT.

## 2 Related Work

Our approach to address the lack of annotated data in discourse parsing lies at the intersection of RST-style parsing, sentiment analysis and multiple-instance learning (MIL).

A large number of highly diverse discourse parsers have been proposed in previous work, with non-neural ones achieving the best performance. In this paper, we consider a set of top-performing parsers, which follow fundamentally different intuitions on how the parsing process should be modelled. Joty et al. (2015) and Ji and Eisenstein (2014) argue that discourse parsing should use a single model for structure, nuclearity and re-

lation modelling. Joty et al. (2015) further propose to separate the task "vertically" on sentence- and document-level, while Ji and Eisenstein (2014) are using a single Shift-Reduce parser based on lexical features. The current state-of-the-art system by Wang et al. (2017) follows an opposing intuition, namely that the task should be separated "horizontally" into two sequenced components. The first classifier models the structure and nuclearity, while the second classifier builds the relation model. Apart from being the state-of-the-art model, Wang et al. (2017) has the ideal architecture for our experiments. With the horizontal separation between structure/nuclearity and relation prediction classifiers, it can be easily tailored to just make discourse structure predictions when trained on our new corpus of discourse trees.

The second related area is sentiment analysis, which we use as our auxiliary task. Previous studies, e.g., Bhatia et al. (2015); Ji and Smith (2017), have shown that sentiment prediction can be enhanced by leveraging discourse information, as the tree structure can influence the significance of certain clauses in the document and boost the overall performance. In particular, Bhatia et al. (2015) use handcrafted discourse features on the sentiment classification task to score clauses depending on the level in the discourse tree. Ji and Smith (2017) use discourse trees generated by a discourse parser (Ji and Eisenstein, 2014) to inform a recursive neural network and automatically learn the model weights for sentiment prediction. In this paper, we exploit the relation between sentiment analysis and discourse parsing in the opposite direction by using sentiment annotations to create discourse structures.

The third area of related work is distant supervision aimed at automatically generating fine-grained annotations. Distant supervision has previously been used to retrieve sentiment (Marchetti-Bowick and Chambers, 2012; Tabassum et al., 2016) and emotion classes (Abdul-Mageed and Ungar, 2017) from opinionated text, showing the potential of distant supervision with user generated content. A common technique for distant supervision is multiple-instance learning (Keeler and Rumelhart, 1992), where the general idea is to retrieve fine-grained information from high-level signals. High-level signals are called *bags* and fine-grained information is referred to as *instances*. The task is defined as the generation of *instance* labels solely based on the given *bag* labels. We follow the approach by Angelidis and Lapata (2018), who train their MILNet system on the publicly available Yelp'13 (Tang et al., 2015) and IMDB (Diao et al., 2014) datasets. Results indicate that MILNet can capture EDU-level sentiment information as well as the relative importance of EDUs when evaluated on datasets annotated on EDU-level. In this paper, we adapt MILNet to generate information useful for deriving discourse trees from corpora with document-level sentiment annotations.

## 3 Our Approach

To generate a large number of discourse structures via distant supervision from sentiment, we propose a four-step approach, shown in Figures 2 and 3. Figure 2 illustrates the first stage of the approach, where for each document in the dataset, (a) the document is segmented into EDUs and (b) our adaptation of MILNet is trained on the document-level sentiment. Next, shown in Figure 3, we again (a) segment the document into EDUs and use (b) the MIL network to generate fine-grain sentiment and importance scores. Then in (c), we prepare those scores to be used in (d), the CKY-like parser, which generates an optimal RST discourse-tree for the document, based on the EDU-level scores and the gold label document sentiment.

### 3.1 Segmentation and Preprocessing

We initially separate the sentiment documents into a disjoint sequence of EDUs. The segmentation is obtained using the discourse segmenter by Feng and Hirst (2012) as generated and published by Angelidis and Lapata (2018). We preprocess the EDUs by removing infrequent- and stop-words and subsequently apply lemmatization.

### 3.2 Multiple-Instance Learning (MIL)

Our MIL model is closely related to the methodology described in Angelidis and Lapata (2018), as well as the papers by Yang et al. (2016) and Ji and Smith (2017). The computation is based on the initial segmentation described in section 3.1 and is shown in further detail in Figure 4.

Our model consists of two levels of Recurrent Neural Networks (RNN) inspired by Yang et al. (2016) and a sentiment- and attention-module. The computational flow in the model is defined

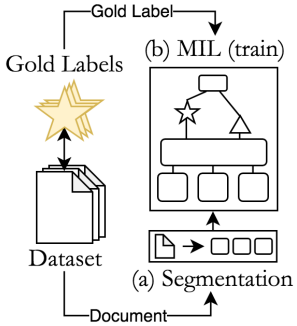


Figure 2: First stage, training the MIL model on the document-level sentiment prediction task

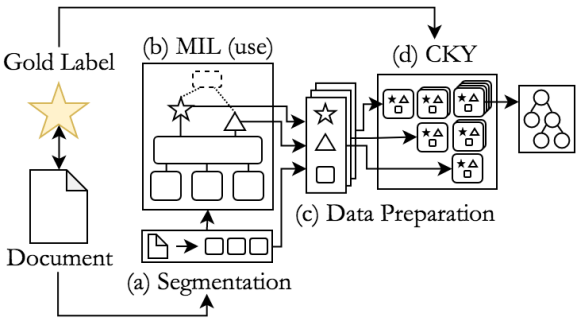


Figure 3: Second stage, using the neural MIL model to retrieve fine-grained sentiment and attention scores (star/triangle), used for the CKY computation to generate the optimal discourse tree

from bottom to top in Figure 4. In a first step, the sparse one-hot word representations are transformed into dense vector-representations $w_i$ using a pretrained GloVe word embedding matrix (Pennington et al., 2014). The dense word representations of a single EDU $E_i = (w_j, ..., w_k)$ are used as the sequential input for the EDU-level RNN, implemented as a bi-directional GRU module with a standard attention mechanism (Bahdanau et al., 2014). The attention-weighted hidden-states $R_{w_i} = H_{w_i} * A_{w_i}$ are concatenated along the time axis to represent $R_{E_i}$. The second RNN on document-level subsequently uses the distributed EDU representations $(R_{E_1}, ..., R_{E_s})$ as inputs for the neural network. Based on the sequence of computed hidden-states $(H_{E_1}, ..., H_{E_s})$ in the bi-directional GRU network, two parallel model components are executed, as follows:

**The non-competitive attention score module** was proposed by Ji and Smith (2017) to leverage discourse structure for sentiment prediction. By following the same intuition, we replace the softmax activation on the attention weights by a sigmoid function. This way, each attention weight $A_{E_i}$ is still limited within the range $(0, 1)$, but the sum of all attention scores is not necessarily bound by 1. We use the attention weight $A_{E_i}$ as the importance scores of $\text{EDU}_\text{i}$.

**The sentiment score module** is also executed directly on the hidden-states $(H_{E_1}, ..., H_{E_s})$ generated by the document-level RNN. To be able to interpret the dense hidden representations as sentiment predictors, we use a single feed-forward neural network layer $S$ with $|C|$ neurons, representing the disjoint sentiment classes $(C_1, ..., C_m)$ in the dataset[1]. We add a sigmoid activation $sigm$ after the feed-forward layer to obtain the final, internal EDU sentiment prediction $S_{E_i} = sigm(S(H_{E_i}))$.

The output of the two parallel modules is multiplied EDU-wise and summed up along the time axis to calculate the final sentiment prediction of our MILNet model as $O_D = \sum_{E_i \in D} S_{E_i} * A_{E_i}$ (see top of Figure 4).

To train our MILNet model, we use the cross-entropy loss function to compare $O_D$ with the gold document-level sentiment label of the review and train using the Adadelta optimizer (Zeiler, 2012). By separating the sentiment and attention components and directly computing the final output based solely on these two values, the neural network implicitly learns the sentiment and attention scores on EDU-level as a by-product of the document-level prediction. For more information on this technique, we refer to Angelidis and Lapata (2018). The hyper-parameter setting of our model also mostly follows the implementation of previous work. We use a batch-size of 200 documents and train the model for 25 epochs. The bidirectional GRU layers contain 100 neurons and the model inputs are preprocessed by limiting the number of EDUs in a document to 150 and defining the maximum length of an EDU to be 20 words. With these settings, we capture over 90% of the data by significantly decreasing the training efforts. We apply 20% dropout on the internal sentiment layer.

[1]In the Yelp'13 dataset, the feed-forward operation $S$ results in 5 real-valued outputs.

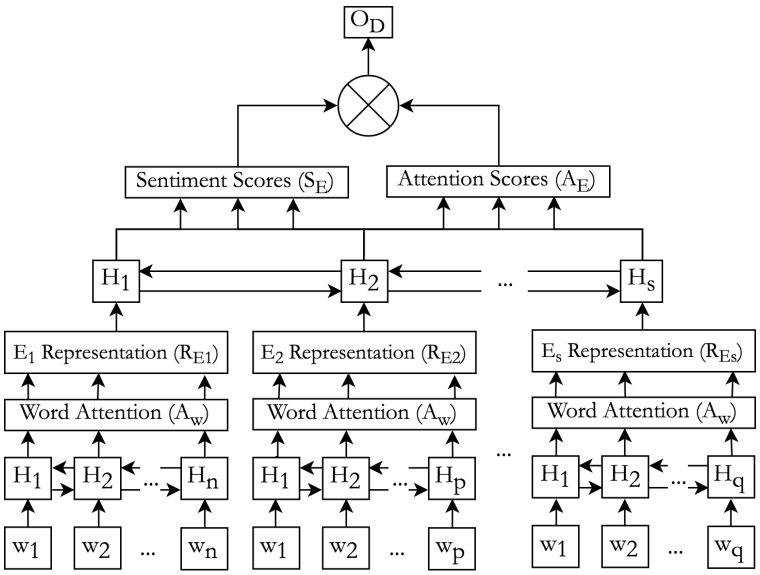


Figure 4: MIL Network Topology (For readability, we leave out the second subscript of the hidden representations $H_{w_i}$ and $H_{E_i}$)

### 3.3 Information Extraction and Transformation

Once our MILNet model is trained, we can use it to obtain the attention score $A_{E_i}$ and the sentiment score $S_{E_i}$ for each EDU $E_i$ in a document (see (c) in Figure 3). However, while each $A_{E_i}$ is already a scalar, $S_{E_i}$ is a vector with $|C|$ elements, one for each sentiment class $C_i$. In order to effectively combine the attention and sentiment scores for further processing, we transform $S_{E_i}$ into a scalar polarity score $pol$, centered around 0 and uniformly distribute the $|C|$ sentiment classes within the interval of $[-1, 1]$. For instance, if $|C| = 5$, this would result in the five classes $pol_{coeff} = [-1, -0.5, 0, 0.5, 1]$. The polarity $pol_{E_i}$ of EDU $E_i$ is computed by calculating the element-wise product of the sentiment score $S_{E_i}$ and the uniform distribution $pol_{coeff}$.

$$pol_{E_i} = \sum_{c \in C} S_{E_i}^{(c)} * pol_{coeff}^{(c)} \quad (1)$$

We transform the gold labels in the same way to keep the representations consistent. With the polarity scores $pol_{E_i}$ replacing the original sentiment scores $S_{E_i}$, a neutral sentiment document now receives a sentiment polarity of 0, while heavily positive or negative EDUs are mapped onto the scores $+1$ and $-1$ respectively. This way, the obtained attention scores $A_{E_i}$ and the calculated polarities $pol_{E_i}$ can be combined to create a weighted sentiment score with high attention values resulting in stronger polarities.

### 3.4 CKY Tree Generation

The final step in our approach (see (d) in Figure 3) takes the tuples of EDU-level attention scores and the generated polarities from the MILNet model to create a set of possible discourse trees. We then select the discourse tree that most precisely computes the overall sentiment of the document. To find the globally best tree, we are computing all possible tree structures (with some constraints) using a dynamic programming approach closely related to the widely used CKY algorithm (Jurafsky and Martin, 2014).

To create discourse trees bottom-up using CKY, we define the necessary aggregation rules for local trees. For each binary subtree, we need to define a function $p(c_l, c_r)$ on how to aggregate the information of the two children $c_l$ and $c_r$ to represent the parent node $p$. For *sentiment*, we use the intuitive attention-weighted average of the children's sentiments, defined by:

$$p_s(c_l, c_r) = \frac{c_{l_s} * c_{l_a} + c_{r_s} * c_{r_a}}{c_{l_a} + c_{r_a}} \quad (2)$$

This way, the parent sentiment does not only depend on the sentiment of its children, but also their relative importance.

For the *attention* computation we consider three different aggregation functions:

**(1)** The sum of the children's attentions (Eq. 3). This way, the combined importance of the children is inherited by the parent node, making the node as

important as the combination of all sub-nodes.

$$p_{a_{sum}}(c_l, c_r) = (c_{l_a} + c_{r_a}) * (1 - \lambda) \quad (3)$$

where $\lambda$ represents a damping factor to penalize lower sub-trees, empirically chosen to be $1\%$ using grid-search.
**(2)** The maximum of the children's attentions.

$$p_{a_{max}}(c_l, c_r) = max(c_{l_a}, c_{r_a}) \quad (4)$$

As shown in equation 4, the attention of the parent node is calculated as the maximum attention value of the two children. This aggregation function follows the intuition that the parent node is only as relevant as the most important child node.
**(3)** The average of the children's attentions.

$$p_{a_{avg}}(c_l, c_r) = \frac{c_{l_a}, c_{r_a}}{2} \quad (5)$$

This aggregation function (Eq. 5) assigns the average importance of the two children to their parent.

To create RST-style discourse documents, which can be used by existing parsers, we need to also provide nuclearity and relation labels for every tree node. While we leave the general task of nuclearity- and relation-prediction for future work, we still need to assign those attributes to tree nodes. We assign nuclearity solely depending on the attention value of the children nodes, making the following binary decision:

$$c_{l_n} = \begin{cases} \text{“}Nucleus\text{”}, & \text{if } c_{l_a} \geq c_{r_a} \\ \text{“}Satellite\text{”}, & \text{otherwise} \end{cases} \quad (6)$$

This simple approach cannot assign “Nucleus-Nucleus” nuclearity attributes, but always requires one child to be the satellite. Finally, for the necessary rhetorical relation attribute, we simply assign the *span* relation to every node.

Due to the high complexity of the optimal CKY algorithm, to keep the process manageable by our computational resources[2], we introduce two constraints on the generated discourse trees:

- We prohibit inter-sentence relations, unless the complete sentence is represented by a single node (as shown to capture the vast majority of discourse relations by Joty et al. (2015))
- We only process documents with less or equal to 20 EDUs per document

[2]Intel Core i9-9820X, RTX 2080 Ti, 128 GB RAM

With the aggregation functions and restrictions described above, we run the CKY-style dynamic programming approach and compare the sentiment at the root node of each of the complete discourse trees with the dataset gold-label for the document. The discourse tree with the smallest distance from the gold-label is selected as the discourse structure representation of the document (see Figure 3, on the right) and saved in a serializable RST-DT format. This way, we generate a dataset of 100k discourse trees.

## 4 Evaluation

We now describe the evaluation of our new discourse structure dataset. We start with the datasets and discourse parsers used to train and test our approach. Next, we describe the evaluation metrics, finishing with experiments and results.

| Dataset | #Documents | #EDU/Doc | Vocab |
|---|---|---|---|
| Yelp'13(2015) | 335,018 | 19.1 | 183,614 |
| RST-DT(2002) | 385 | 56.0 | 15,503 |
| Instr-DT(2009) | 176 | 32.6 | 3,453 |

Table 1: Dataset size

### 4.1 Datasets

We use three datasets to train and evaluate our approach. Table 1 summarizes the most important dataset dimensions.

**Yelp'13** is a review dataset collected for the Yelp Dataset Challenge in 2013 (Tang et al., 2015). Every datapoint in the corpus consists of a review along with a star-rating on a 5-point scale. We use the discourse segmented version of the corpus by Angelidis and Lapata (2018) to train our system on the auxiliary sentiment prediction task.

**RST-DT** is the largest and most frequently used corpus to train RST-style discourse parsers (Carlson et al., 2002). The dataset consists of news articles from Wall Street Journal. We use the standard data split with 90% training data (*RST-DT*$_{train}$) and 10% test data (*RST-DT*$_{test}$) to test the performance of our approach against competitive baselines.

**Instructional Dataset** is another RST-style dataset to evaluate discourse parsers on the domain of home-repair instructions (Subba and Di Eugenio, 2009). For convenience, we refer to this corpus as Instr-DT from here on. We separate the data into 90% training data (*Instr-DT*$_{train}$) and 10% test data (*Instr-DT*$_{test}$).

**Vocabulary Overlap** is measured using the Jaccard similarity index. We show the absolute vocabulary sizes of the datasets in Table 1 and visualize the overlap in Table 2. The vocabulary overlap between the Yelp'13 corpus (containing reviews), the RST-DT dataset (on news articles) and the Instr-DT corpus (containing home-repair instructions) is predictably low, given the different domains of the datasets. While this would be a problem for models solely basing their prediction on raw input words, our system goes beyond just words as inputs. During training, we use pre-trained word embeddings to encode the inputs and the state-of-the-art discourse parser (Wang et al., 2017) uses a combination of syntactical and lexical features to represent words.

| | |
|---|---|
| Yelp'13 ↔ RST-DT | 6.28% |
| Yelp'13 ↔ Instr-DT | 1.73% |
| RST-DT ↔ Instr-DT | 11.65% |

Table 2: Vocabulary overlap between datasets

### 4.2 Discourse Parsers

In our experiments, we apply four simple baselines and four competitive discourse parsers, often used in previous work for comparison studies.

**Right/Left Branching Baselines:** predict a binary, fully right- or left-branching tree for every document in the dataset.

**Hierarchical Right/Left Branching Baselines:** predict a binary, fully right- or left-branching tree on sentence-level and combine the sentence-level trees in right- or left-branching manner for every document in the dataset.

**HILDA:** a classic, greedy, bottom-up parser using linear SVMs (Hernault et al., 2010).

**DPLP:** a SVM-based shift-reduce parser build on linear projections of lexical features (Ji and Eisenstein, 2014).

**CODRA:** a CKY-based chart parser combined with Dynamic Conditional Random Fields, separating the computation on sentence- and document-level (Joty et al., 2015).

**Two-stage Parser:** current state-of-the-art parser by Wang et al. (2017). Employs two separate SVM classifiers for structure/nuclearity and relations, reaching the best performance for structure and nuclearity. This is the parser we rely on in our experiments due to its performance advantage compared to other discourse parsers and its separate computation of the structure/nuclearity and the discourse relation. We use the publicly available code provided by Wang et al. (2017) and remove the relation classification module.

### 4.3 Metrics

Consistent with previous work, e.g., Wang et al. (2017); Joty et al. (2015) and following the recent analysis by Morey et al. (2017), our key metric is the average micro precision on span level, computed as the global overlap of the discourse structure prediction and the gold structure. We traverse both discourse trees $tree_{pred_i}$ and $tree_{gold_i}$ of each document $i$ in post-order and compute:

$$precision = \frac{\sum_i tree_{pred_i} \cap tree_{gold_i}}{\sum_i |tree_{gold_i}|} \quad (7)$$

Notice that the choice of precision over recall and F-score has no impact on the results when using manual segmentation, as shown in previous work, e.g., Wang et al. (2017); Joty et al. (2015).

### 4.4 Experiments and Results

We run experiments in two phases. In the first phase, the state-of-the-art discourse parser by Wang et al. (2017) is individually trained on each of the datasets and tested on the two corpora containing gold discourse annotations. In the second phase, the best results are placed in the broader context of competitive discourse parsers.

**Phase 1:** We train the state-of-the-art discourse parser on five different corpora and perform tests on two corpora. The five training corpora are: RST-DT$_{\text{train}}$, Instr-DT$_{\text{train}}$ and the three versions of our novel dataset, generated using the different attention aggregation functions $(avg, max, sum)$ discussed in Section 3.4. The two corpora used for testing are RST-DT$_{\text{train}}$ and Instr-DT$_{\text{train}}$ (for which gold standard annotations are available). Notice that whenever training and testing are performed on the same corpus, the model is trained on the training portion of the dataset (RST-DT$_{\text{train}}$ or Instr-DT$_{\text{train}}$) and evaluated on the test data (RST-DT$_{\text{test}}$ or Instr-DT$_{\text{test}}$). Finally, since one of the key benefits of our approach is the ability to generate large dataset, we also assess the relation between the dataset size and the parser performance in this

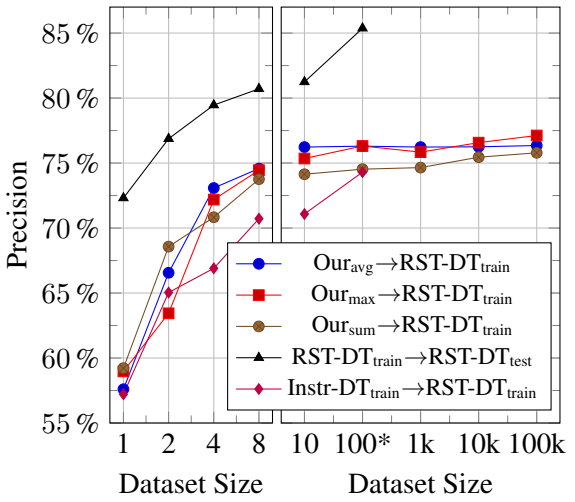


Figure 5: Results of training and testing on the datasets listed in the legend (*Complete dataset was used for RST-DT(385 documents) and Instr-DT(176 documents))

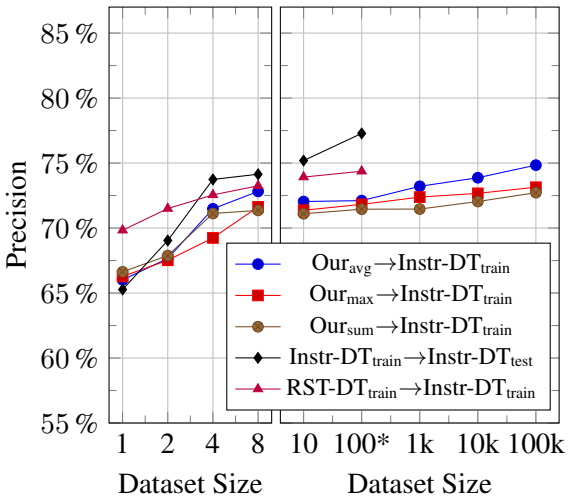


Figure 6: Results of training and testing on the datasets listed in the legend (*Complete dataset was used for RST-DT(385 documents) and Instr-DT(176 documents))

phase. The results of all our experiments in the first phase are shown in Figure 5 and Figure 6.

In both figures, the left side shows the performance using small subsets of sizes $2^n$ for $n = 0, 1, ..$ of the training data, while the right side shows the performance on large subsets of sizes $10^n$, as well as the full datasets. The precision value displayed for each subset is the average of 10 randomly selected samples from the full corpus. Figure 7 shows the variance within the 10 samples, highlighting the increasing reliability of larger subsets[3].

The results shown in the two figures reveal several important findings:

**(1)** While training the parser on our corpus does not yet match the performance of the parser trained and tested on the same dataset (*intra-domain*, see black lines in Figure 5 and 6), it does achieve the best performance in the *inter-domain* discourse structure prediction.

**(2)** The performance generally increases with more training data. Larger datasets could therefore further increase the performance.

**(3)** Tested on the full *inter-domain* training datasets (100k), the $avg$ attention-aggregation function achieves the most consistent performance on the corpora. When evaluated only on RST-DT, the $max$ aggregation function reaches the overall best performance, while the $avg$ attention-aggregation function reaches the best performance

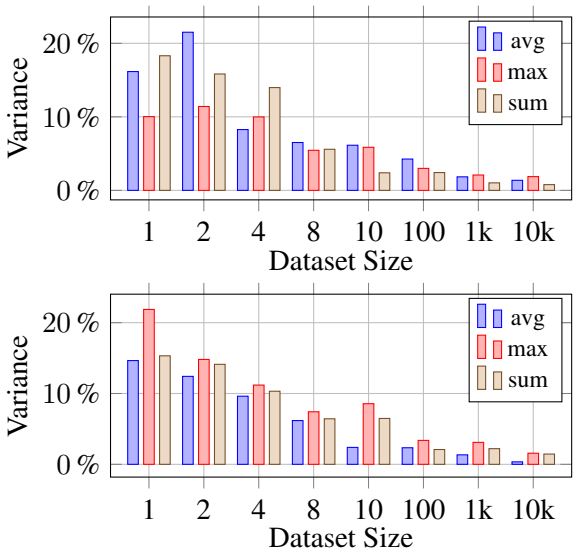


Figure 7: Sample variance across different subset sizes on the RST-DT dataset (top) and Instr-DT corpus (bottom)

when only evaluating on Instr-DT.

**(4)** Small subsets of training data (4, 8 documents) already achieve relatively good results, especially on Instr-DT. One possible explanation for this behaviour is the large number of training datapoints within a single document, which generates $2*n-1$ binary training instances, where $n$ is the number of EDUs in the document.

**(5)** Generally, performance is highly dependent on the dataset domain and structure. In particular, the performance on the Instr-DT is generally lower and saturates earlier than on RST-DT. Both

[3]The complete dataset has not been subsampled and therefore does not have a variance defined.

effects could result from Instr-DT containing less and shorter documents than RST-DT.

**Phase 2:** We further analyze our findings with respect to baselines and existing discourse parsers. The first set of results in Table 3 shows that the hierarchical right/left branching baselines dominate the completely right/left branching ones. However, their performance is still significantly worse than any discourse parser (*intra-* and *inter-domain*).

| Approach | RST-DT$_{\text{test}}$ | Instr-DT$_{\text{test}}$ |
|---|---|---|
| Right Branching | 54.64 | 58.47 |
| Left Branching | 53.73 | 48.15 |
| Hier. Right Branch. | **70.82** | **67.86** |
| Hier. Left Branch. | 70.58 | 63.49 |
| **Intra-Domain** Evaluation | | |
| HILDA(2010) | 83.00 | — |
| DPLP(2014) | 82.08 | — |
| CODRA(2015) | 83.84 | **82.88** |
| Two-Stage(2017) | **86.00** | 77.28 |
| **Inter-Domain** Evaluation | | |
| Two-Stage$_{\text{RST-DT}}$ | × | 73.65 |
| Two-Stage$_{\text{Instr-DT}}$ | 74.48 | × |
| Two-Stage$_{\text{Ours(avg)}}$ | 76.42 | **74.22** |
| Two-Stage$_{\text{Ours(max)}}$ | **77.24** | 73.12 |
| Human (2017) | 88.30 | — |

Table 3: Discourse structure prediction results; tested on RST-DT$_{\text{test}}$ and Instr-DT$_{\text{test}}$. Subscripts in *inter-domain* evaluation sub-table indicate the training set. Best performance in the category is **bold**. Consistently best model for *inter-domain* discourse structure prediction is underlined

The second set of results show the performance of existing discourse parsers when trained and tested on the same dataset (*intra-domain*). We use the results published in the original paper whenever possible. The Two-Stage approach by Wang et al. (2017) achieves the best performance with 86% on the structure prediction using the RST-DT dataset. On the Instructional dataset, the CODRA discourse parser by Joty et al. (2015) achieves the highest score with 82.88%.

The third set in the table shows the key results from Phase 1 on the *inter-domain* performance.

Our models, learning the discourse structure solely from the *inter-domain* Yelp'13 review dataset through distant supervision, reach better performance than the human annotated datasets (RST-DT and Instr-DT) when trained *inter-domain*, despite the low vocabulary overlap between the Yelp'13 corpus and RST-DT/Instr-DT (Table 2). While the $avg$ attention-aggregation function achieves the most consistent performance on both evaluation corpora, the $max$ function should not be dismissed, as it performs better on the larger RST-DT dataset, which is arguably more related to the Yelp'13 corpus than the sentiment-neutral Instr-DT.

Furthermore, the fact that our best models reach a performance of only 8.76% and 8.66% below the best *intra-domain* performances (tested on RST-DT and Instr-DT, respectively), shows the potential of our approach, even when compared to *intra-domain* results.

## 5 Conclusions and Future Work

In this paper, we address a key limitation to further progress in discourse parsing: the lack of annotated datasets. We show promising initial results to overcome this limitation by creating a large-scale dataset using distant supervision on the auxiliary task of sentiment analysis. Experiments indicate that a parser trained on our new dataset outperforms parsers trained on human annotated datasets on the challenging and very useful task of *inter-domain* discourse structure prediction.

There are several directions for future work. First, given that we can now create large datasets, we intend to experiment on structure prediction with neural discourse parsers, which so far have delivered rather disappointing results. Second, an obvious next step is working on the integration of nuclearity and relation prediction to create complete RST annotations for documents from auxiliary tasks and to extend our evaluations (Zeldes, 2017). Third, we will study synergies between discourse parsing and further auxiliary tasks, eventually creating a single, joint system to generate globally high-quality discourse trees. Finally, instead of creating discourse structures and training existing discourse parsers on the data, we will design and implement an end-to-end system to train the complete process holistically.

## Acknowledgments

We want to thank the anonymous reviewers for their insightful comments and suggestions. This research was supported by the Language & Speech Innovation Lab of Cloud BU, Huawei Technologies Co., Ltd.

## References


Muhammad Abdul-Mageed and Lyle Ungar. 2017. Emonet: Fine-grained emotion detection with gated recurrent neural networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 718–728.

Stefanos Angelidis and Mirella Lapata. 2018. Multiple instance learning networks for fine-grained sentiment analysis. *Transactions of the Association for Computational Linguistics*, 6:17–31.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Parminder Bhatia, Yangfeng Ji, and Jacob Eisenstein. 2015. Better document-level sentiment analysis from rst discourse parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2212–2218.

Chloé Braud, Maximin Coavoux, and Anders Søgaard. 2017. Cross-lingual rst discourse parsing. *arXiv preprint arXiv:1701.02946*.

Chloé Braud, Barbara Plank, and Anders Søgaard. 2016. Multi-view and multi-task training of rst discourse parsers. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1903–1913.

Lynn Carlson, Mary Ellen Okurowski, and Daniel Marcu. 2002. *RST discourse treebank*. Linguistic Data Consortium, University of Pennsylvania.

Qiming Diao, Minghui Qiu, Chao-Yuan Wu, Alexander J Smola, Jing Jiang, and Chong Wang. 2014. Jointly modeling aspects, ratings and sentiments for movie recommendation (jmars). In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 193–202. ACM.

Vanessa Wei Feng and Graeme Hirst. 2012. Text-level discourse parsing with rich linguistic features. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 60–68.

Vanessa Wei Feng and Graeme Hirst. 2014. A linear-time bottom-up discourse parser with constraints and post-editing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 511–521.

Elisa Ferracane, Greg Durrett, Junyi Jessy Li, and Katrin Erk. 2019. Evaluating discourse in structured text representations. *arXiv preprint arXiv:1906.01472*.

Shima Gerani, Yashar Mehdad, Giuseppe Carenini, Raymond T Ng, and Bita Nejat. 2014. Abstractive summarization of product reviews using discourse structure. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1602–1613.

Hugo Hernault, Helmut Prendinger, Mitsuru Ishizuka, et al. 2010. Hilda: A discourse parser using support vector machine classification. *Dialogue & Discourse*, 1(3).

Alexander Hogenboom, Flavius Frasincar, Franciska De Jong, and Uzay Kaymak. 2015. Using rhetorical structure in sentiment analysis. *Commun. ACM*, 58(7):69–77.

Yangfeng Ji and Jacob Eisenstein. 2014. Representation learning for text-level discourse parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 13–24.

Yangfeng Ji and Noah A Smith. 2017. Neural discourse structure for text categorization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 996–1005.

Shafiq Joty, Giuseppe Carenini, and Raymond T Ng. 2015. Codra: A novel discriminative framework for rhetorical analysis. *Computational Linguistics*, 41(3).

Dan Jurafsky and James H Martin. 2014. *Speech and language processing*, volume 3. Pearson London.

Jim Keeler and David E Rumelhart. 1992. A self-organizing integrated segmentation and recognition neural net. In *Advances in neural information processing systems*, pages 496–503.

Qi Li, Tianshi Li, and Baobao Chang. 2016. Discourse parsing with attention-based hierarchical neural networks. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 362–371.

Yang Liu and Mirella Lapata. 2017. Learning contextually informed representations for linear-time discourse parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1289–1298.

Yang Liu and Mirella Lapata. 2018. Learning structured text representations. *Transactions of the Association for Computational Linguistics*, 6:63–75.

William C Mann and Sandra A Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text-Interdisciplinary Journal for the Study of Discourse*, 8(3):243–281.

Micol Marchetti-Bowick and Nathanael Chambers. 2012. Learning for microblogs with distant supervision: Political forecasting with twitter. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 603–612.

Mathieu Morey, Philippe Muller, and Nicholas Asher. 2017. How much progress have we made on rst discourse parsing? a replication study of recent results on the rst-dt. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1319–1324.

Bita Nejat, Giuseppe Carenini, and Raymond Ng. 2017. Exploring joint neural model for sentence level discourse parsing and sentiment analysis. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 289–298.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The penn discourse treebank 2.0. *LREC*.

Rajen Subba and Barbara Di Eugenio. 2009. An effective discourse parser that uses rich linguistic information. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 566–574. Association for Computational Linguistics.

Jeniya Tabassum, Alan Ritter, and Wei Xu. 2016. Tweetime: A minimally supervised method for recognizing and normalizing time expressions in twitter. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 307–318.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1422–1432.

Yizhong Wang, Sujian Li, and Houfeng Wang. 2017. A two-stage parsing method for text-level discourse analysis. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 184–188.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Nan Yu, Meishan Zhang, and Guohong Fu. 2018. Transition-based neural rst parsing with implicit syntax features. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 559–570.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Amir Zeldes. 2017. The GUM corpus: Creating multilayer resources in the classroom. *Language Resources and Evaluation*, 51(3):581–612.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

# The Myth of Double-Blind Review Revisited: ACL vs. EMNLP

**Cornelia Caragea**[1], **Ana Sabina Uban**[2], **and Liviu P. Dinu**[2]
[1]Computer Science, University of Illinois at Chicago
Chicago, Illinois
[2]Faculty of Mathematics and Computer Science, University of Bucharest
Bucharest, Romania
cornelia@uic.edu, ana.uban@gmail.com, ldinu@fmi.unibuc.ro

## Abstract

The review and selection process for scientific paper publication is essential for the quality of scholarly publications in a scientific field. The double-blind review system, which enforces author anonymity during the review period, is widely used by prestigious conferences and journals to ensure the integrity of this process. Although the notion of anonymity in the double-blind review has been questioned before, the availability of full text paper collections brings new opportunities for exploring the question: *Is the double-blind review process really double-blind?* We study this question on the ACL and EMNLP paper collections and present an analysis on how well deep learning techniques can infer the authors of a paper. Specifically, we explore Convolutional Neural Networks trained on various aspects of a paper, e.g., content, style features, and references, to understand the extent to which we can infer the authors of a paper and what aspects contribute the most. Our results show that the authors of a paper can be inferred with accuracy as high as 87% on ACL and 78% on EMNLP for the top 100 most prolific authors.

## 1 Introduction

The scientific peer-review process is indispensable for the dissemination of high-quality information (Hojat et al., 2003). However, one of the major problems with this process is bias (Williamson, 2003; Tomkins et al., 2017). For example, Tomkins et al. (2017) performed an experiment during the ACM Web Search and Data Mining conference 2017 to understand the potential bias in favoring authors from prestigious institutions and found that, indeed, when reviewers have access to the authors identities, they more often tend to favor well known authors from prestigious institutions. The double-blind review process is generally employed by top scientific journals and conferences in order to guarantee *fairness* of the paper selection, and thus, plays an essential role in how scientific quality is eventually measured (Meadows, 1998). It is designed to reduce the risk of bias in paper reviews, ensuring that all papers are judged solely based on their content and intrinsic quality and that any author has a fair chance of having a paper accepted, regardless of their prestige or previous work. The double-blind review process implies that the submitted papers have to be anonymized, i.e., the authors' names are not explicitly available with the papers, and any direct or indirect indications of who the authors might be (for example, referring to self-citations in the first person) are forbidden. Reviewers have access only to the papers' content, and the authors in turn do not know who their assigned reviewers are. Despite these strict considerations, the notion of anonymity in the double-blind review has still been questioned. Notably, Hill and Provost (2003) showed that the authors of a scientific paper can be inferred with fairly high accuracy using only the papers it references. Specifically, using the vector-space representations of the list of references (or citations) in a paper and measuring similarity between these representations and the pattern of citations of an author, they are able to infer the authors of a paper with accuracy up to 60% for the top-10% most prolific authors, and show that self-citations are an important predictive factor.

We are interested in further understanding how predictable the authorship of a paper is, and specifically what part of the paper gives it away. For this purpose, we include in our analysis additional text features to reflect various aspects of the text, as well as references, and make use of a more complex machine learning model, based on deep learning, for predicting authors based on these features. We focus on ACL and EMNLP, the top conferences in computational linguistics, which use the



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2317–2327,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

double-blind review system to decide whether to accept papers for publication.

Our contributions are as follows: we train deep learning models on papers published in the ACL and EMNLP conferences, using features extracted from each paper's body of text as well as its references, and show that these models are able to predict authors with accuracy of about 87% on ACL and about 78% on EMNLP. We additionally perform an ablation study, for an in-depth analysis of the predictive value of each feature. We finally also show how the number of authors considered for analysis can affect performance.

The rest of the paper is organized as follows: In the next section we present related work. Then in Section 3, we describe our datasets. Section 4 deals with the methodology of our experiments, including our baseline algorithm and the details of the model we propose. In Subsections 4.2 and 4.3 we also discuss the features we used in our model and the details of their extraction and preprocessing steps. Section 5 describes the setup of our deep learning experiments, including the metrics we use for measuring performance. Finally, in Subsection 5.3 we report and discuss our results, and in Section 6 we present our conclusions.

## 2 Related Work

There have been several studies approaching the question of the integrity of the double-blind review process. An early study on blind review published in a journal of psychology (Ceci and Peters, 1984) shows that authors of anonymous papers could be identified by surveyed reviewers using the combination of the paper's references and the referee's personal background knowledge.

Statistical studies on the difference between single-blind and double-blind peer review have more recently demonstrated that unveiling the identity of the authors to the reviewers leads to biased reviews, favoring more prestigious authors and institutions. Tomkins et al. (2017) performed a controlled experiment on scientific articles submitted to the 10th ACM Conference on Web Search and Data Mining, where for every article half of the reviewers had access to author information, while the other half did not. They found that single-blind reviewers are more likely to recommend famous authors for acceptance by a factor of 1.58. A few studies have previously proposed automatic approaches for author prediction for scientific articles. For example, Hill and Provost (2003) successfully predicted the authors of scientific articles published as part of the KDD Cup 2003 competition, using only information from the articles' references lists.

In a related task to ours, several studies have looked at authorship attribution on scientific articles, or predicting authors of scientific articles from an exclusively stylistic point of view. Althoff et al. (2013) studied authorship attribution on scientific articles specifically in the multi-author setting, using various text-based features (including word $n$-grams and various stylistic features) and models based on logistic regression and expectation maximization. Hitschler et al. (2017) performed experiments for predicting authors of ACL articles, restricting their data to only single-author articles. Their study focused on the style level, using only POS tag sequences, and showed that limiting the number of words considered as features can have a beneficial effect on the predictor's performance. Seroussi et al. (2012) proposed the use of an author-topic model (Rosen-Zvi et al., 2004) for the task of authorship attribution and showed promising results in a scenario with many authors. Rexha et al. (2015) analyzed the style of medical scientific articles and how the stylistic uniformity of an article varies with the number of co-authors.

Outside the world of scientific articles, a few previous studies showed the promise of using neural networks for authorship attribution. Bagnall (2015) successfully used a multi-headed Recurrent Neural Network for an author identification task at PAN 2015. The use of Convolutional Neural Networks (CNNs) for learning from text data was proposed by Kim (2014), where CNNs are successfully applied to several sentence classification tasks. Rhodes (2015) trained a Convolutional Neural Network on word embeddings for predicting authors of medium-sized texts, and Shrestha et al. (2017) used CNNs in an authorship attribution task on tweets. Luyckx and Daelemans (2008) studied the effects of having many authors as classes and of limited training data on author attribution - which are realistic, but difficult scenarios, common to our problem as well.

As far as we are aware, no other study has dealt with analyzing the authorship of articles published at ACL or EMNLP (or a comparably prestigious conference) without restricting the scenario to only a subtask (for example, focusing only on a subset of the data), or limiting the analysis to one

aspect of the text (for example, focusing on the stylistic level). While previous studies support the hypothesis that authors of a scientific article are possible to predict from an anonymized paper, we attempt to provide a fuller picture regarding what exactly it is about an anonymous article that can give away its authors.

## 3 Datasets

For evaluation, we used two datasets of articles from the computational linguistics conferences ACL and EMNLP, published on or before 2014 (Bird et al., 2008; Anderson et al., 2012). The ACL dataset contains $4,412$ articles authored by a total of $6,565$ unique authors, whereas the EMNLP dataset is comprised of $1,027$ articles written by $1,861$ unique authors in total.[1] Note that the size of the EMNLP dataset is much smaller than the size of the ACL dataset since EMNLP is a much newer conference compared to ACL. From each dataset, we normalized the author names to consist of the initial of the first name and the full last name and removed the authors with less than three articles (to ensure enough data for training and evaluation), leaving us with 922 authors for the ACL dataset and 262 authors for the EMNLP dataset (which represent our classes).

As illustrated in Figure 1, which plots the class distribution in each dataset (i.e., the number of articles per author in decreasing order), we can see that the distribution is very skewed, with the more prolific authors being responsible for many of the articles in each dataset and with many authors contributing only a few articles.

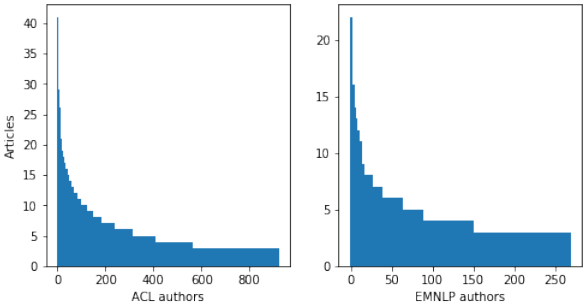


Figure 1: Class (author) distribution.

A similar, if not more pronounced, imbalance can be observed at the level of cited authors. For the purpose of our experiments, we also extracted and analyzed the references (or citation) lists of each article in our datasets, and looked at the distribution of citations across all cited authors. This distribution is illustrated in Figure 2, which plots the number of times each author is cited (in decreasing order). The values on the $y$ axis in this figure are plotted using a logarithmic scale, since the distribution is very skewed.

Tables 1 and 2 show the top-15 most prolific authors as well as the top-15 most cited authors found in the citations lists in our datasets, for both ACL and EMNLP, respectively.

[1] Code and data available upon request.

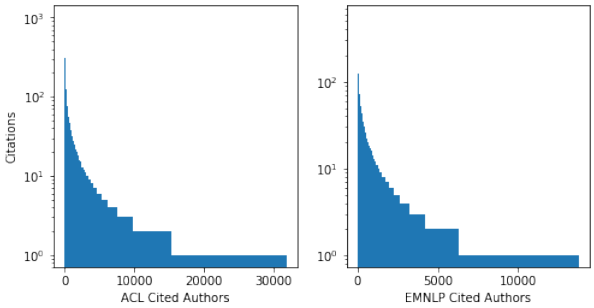


Figure 2: Number of citations per author.

| Author | #Papers | Author | #Citations |
|---|---|---|---|
| D Klein | 41 | C Manning | 656 |
| M Zhou | 39 | D Klein | 562 |
| M Johnson | 32 | F Pereira | 561 |
| I Dagan | 30 | M Collins | 539 |
| C Manning | 29 | D Marcu | 459 |
| K Knight | 28 | P Coehn | 441 |
| M Zhang | 27 | S Roukos | 414 |
| Y Liu | 27 | E Charniak | 400 |
| Q Liu | 27 | A McCallum | 393 |
| R Barzilay | 26 | K Knight | 388 |
| N Smith | 25 | F Och | 387 |
| E Hovy | 24 | M Marcus | 377 |
| G Satta | 23 | M Johnson | 355 |
| H Li | 23 | D Jurafsky | 349 |
| Y Matsumoto | 21 | H Ney | 348 |

Table 1: Top 15 most prolific authors and most cited authors in our ACL dataset.

| Author | #Papers | Author | #Citations |
|---|---|---|---|
| D Klein | 22 | C Manning | 333 |
| N Smith | 16 | D Klein | 308 |
| H Ng | 16 | M Collins | 246 |
| C Manning | 16 | F Pereira | 238 |
| G Zhou | 15 | A McCallum | 222 |
| M Lapata | 14 | P Koehn | 204 |
| J Eisner | 13 | D Marcu | 191 |
| D Roth | 13 | F Och | 188 |
| Q Liu | 12 | R McDonald | 179 |
| M Collins | 12 | S Roukos | 174 |
| K Torisawa | 11 | D Jurafsky | 173 |
| M Zhang | 11 | K Knight | 167 |
| Y Liu | 11 | M Marcus | 155 |
| M Zhou | 9 | M Johnson | 154 |
| S Petrov | 9 | A Ng | 143 |

Table 2: Top 15 most prolific authors and most cited authors in our EMNLP dataset.

*Author Name Normalization.* It is worth mentioning that our method for normalizing author names can produce collisions, and hence, ambiguities. However, we chose to normalize the names because the noise resulting from not doing so (i.e., having the same author encoded with multiple different ways of writing their name, especially prevalent in references) might be even more detrimental to learning than the possible ambiguities.

In order to understand the level of collisions in our classes (i.e., the author names in the headers), we show in Figure 3 the number of author names that result in three collisions, two collisions, or no collisions at all after normalization, for both ACL and EMNLP. As can be seen from the figure, the number of author names with collisions in each dataset is small. Among the names with collisions, 13 occur within the top most prolific 100 authors in ACL, and only 3 occur in the top 100 for EMNLP. Note that a similar analysis for the author names in the references lists is difficult since many names appear already in the normalized form (first name initial last name). For this reason, normalizing author names is also necessary for computing our baseline, which matches the names of article authors with names of cited authors.

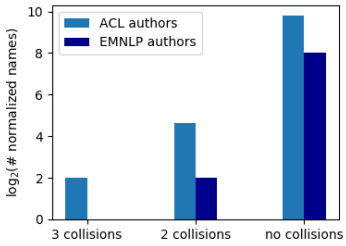


Figure 3: Collision analysis for author name normalization (shown on a $\log_2$ scale).

Author name normalization is in some cases useful, e.g., in the case of authors with middle names, which are sometimes explicit and other times omitted (there are 12 cases in the ACL dataset of authors with middle names whose names occur differently in different articles), or in the case of authors whose first names can have different spellings such as *Dan/Daniel Jurafsky*.

# 4 Methodology

## 4.1 Baseline

For our baseline we chose to focus solely on the citations of each article. As Hill and Provost (2003) have shown before, citations alone can be a strong indicator of the authors of an article, with self-citations being especially telling. Specifically, our baseline algorithm consists of simply ranking the authors cited in an article in reverse order of how frequently they were cited overall in the article's references list, and outputs as predicted authors the top (10) most cited authors in this ranking.

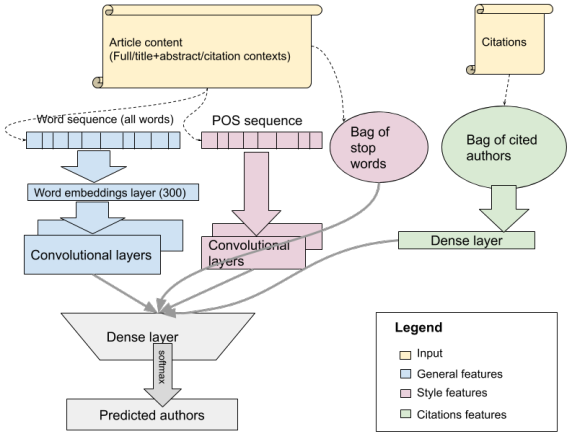


Figure 4: Network architecture

## 4.2 Proposed Model

For the purpose of our machine learning experiments, we formulate the problem as a supervised classification task, where each article is labelled with one or multiple authors, and a machine learning model learns to predict the set of correct labels (authors) for each data point (article). The order of the authors is not taken into consideration.

As our model, we choose a neural network with several subcomponents corresponding to various types of features, as detailed below.

### 4.2.1 Features

In order to capture as many of the aspects of a scientific article as possible in our model, we extract and use various features, corresponding to different levels at which characteristics of the author could manifest. We categorize these into three main types of features:

- Content level: word sequences (consisting of 100-word sequences in the article's title, abstract and body).
- Style level: stopwords bag-of-words, part-of-speech sequences.
- Citation level: bag-of-words of cited authors.

Figure 4 shows a high-level view of the network architecture and its various components.

The network is designed to learn from each separate feature using dedicated subcomponents. At the content level, we use convolutional layers to learn from word sequences. CNNs have been shown to be successful in text classification tasks by Kim (2014). We use similar settings to the ones reccommended in this study - passing the word sequences through a single convolutional layer with 300 filters, and a kernel size of 9, followed by a max pooling layer. Before going through the convolutions, the word sequences are passed through a word embedding layer of 300 dimensions, initialized with the pre-trained word2vec embeddings available from Google, trained using the skip-gram objective (Mikolov et al., 2013). Using embeddings that are already pre-trained on a large dataset should benefit our task (our dataset being itself not very large), but since we use general-purpose pre-trained embeddings, we choose not to fix the embedding weights, but rather let the network further update them by learning from our data to tune them to our task and domain.

A separate convolutional layer is dedicated to learning from part-of-speech sequences. As in the case of word sequences, we consider the order of parts of speech in a text segment to be relevant, assuming certain types of syntactical constructs can be specific to certain authors. Thus, after tagging a text segment with parts of speech, we encode the result as part-of-speech sequences and pass them through a convolutional layer of 50 filters and kernel size 4, followed by a max pooling layer. The POS tags are given as one-hot vectors. Stopwords are extracted from each article segment and encoded as bag-of-words, keeping their frequencies, but not their order in the text. We used the stopword list available from the NLTK package. Stopwords frequencies are traditionally used in stylometry, being one of the most indicative features of an author's style (Koppel et al., 2009).

To extract knowledge from citations, we focus on cited authors, encoding for each article the authors cited in its references section, along with the citation frequencies for each author. The total number of cited authors in each of our datasets is much larger than the number of authors that contribute directly to one of the articles, e.g., over $22,000$ unique authors are cited in our ACL dataset. This makes the one-hot encoding that we use for cited authors to be very high-dimensional, so we pass the extracted feature through an additional lower-dimensional fully connected layer.

In the final layers of our network, we collect all of the output from each subcomponent dealing with the various features, and pass them through a dense component consisting of a fully connected layer and a Softmax layer that produces the network's predicted probabilities for each class.

### 4.3 Preprocessing and Feature Extraction

We extracted the text from PDFs using Grobid.[2] Several preprocessing steps were necessary before using the articles' text as features in our model.

For our text-related features we consider the title, abstract and body of the articles, and exclude references from the article's text, by removing them both from the references section and from the citations within the article text (so as to isolate text features from citation features). After normalizing and tokenizing the resulted text according to usual practice in natural language processing applications (including lowercasing every word, discarding numbers and punctuation, resolving end-of-line hyphenation), we construct a list of vocabulary words consisting of the most frequent 50,000 words in all texts. Our choice of vocabulary size was informed by a previous study looking at authorship on ACL data (Hitschler et al., 2017), which showed that 50,000 words is an optimal vocabulary size for authorship tasks on this dataset. For EMNLP, which is a smaller dataset, we restrict the minimum word frequency to 5 occurrences, leaving us with a vocabulary of approximately 23,000 words. Considering only the words in each vocabulary (and replacing all other words with an "unknown" token), we encode the text as word sequences of 100 words, padding the sequences with zeros if they are shorter. Further, our training examples consist of these word segments, rather than full articles. Before extracting content features, we discard outliers, ignoring articles consisting of either zero or more than $20,000$ words.

In addition, we also extract the context around citation mentions within the content of articles, by selecting a window of 100 characters around the citation (and excluding the citation itself), then applying the same text preprocessing steps as above only on this window. This is used as a separate feature, as described in Section 5.2.

The extraction of the part-of-speech features is done by applying the Stanford POS tagger (Toutanova et al., 2003) to the word sequences, re-

[2] https://github.com/kermitt2/grobid

sulting in part-of-speech sequences corresponding to each article segment. Stopwords are encoded as bag-of-words for each article segment.

Citations extracted from the "References" section of each article are encoded as bag-of-authors - unordered sets of citation frequencies corresponding to each cited author. Recall that author names (when they occur either as authors of the target article, or as authors of a cited paper in a references list) are normalized to consist of the initial of the first name and the full last name (see Section 3).

## 5 Experimental Setup and Results

The nature of our dataset requires special attention to the setup of the training experiments, one of the main particularities of the data being the skewness of the label distribution. We split the dataset in three subsets: one for actual training, one for validation (used for tuning hyperparameters), and the third for testing performance. At this stage, we ensure that each of the three sets contains at least one article from each author in our labeled set. This also implies that we exclude any authors with less than three articles, obtaining 922 authors for the ACL dataset and 262 for EMNLP (which are the different classes in our supervised learning problem). Given our datapoints, as explained in the previous section, consisting of article segments (word sequences of 100 words) rather than full articles, we also ensure that all segments extracted from one given scientific article are appointed to the same set, and not split between training, validation and test. We take this precaution to make sure that anything our network learns is not an artifact of the particular article, but rather of its author.

Lastly, to reduce the impact of the label skewness on our trained model, we use weighted sampling for generating the training examples, making sure the probability of generating a training example from any class is approximately the same across classes. For this to be possible, a final adjustment had to be made to our training examples. Our datasets, comprising of scientific articles, with each article having been written either by a single author or in co-authorship between several authors, essentially consists of multi-label examples. For training, we transform the training examples from multi-label examples to single-label examples, by generating several copies of the same datapoint, each labelled with only one of its authors, whenever a text was written by more than one author. This allows us to perform weighted sampling, as well as use a simple softmax layer as the final layer in the network, which generates one predicted label for any training example, and cross-entropy loss as our loss function. At the evaluation stage, we use the original multi-labeled examples, to be able to correctly measure the model's performance using our metrics.

| Dataset | Nr authors | Training | Valid | Test |
|---|---|---|---|---|
| ACL | Top 100 | 73,261 | 12,260 | 12,348 |
| ACL | Top 200 | 124,460 | 18,756 | 19,967 |
| ACL | Mid 200 | 25,746 | 12,476 | 13,618 |
| ACL | Bottom 200 | 13,673 | 12,378 | 12,074 |
| ACL | Top 500 | 185,451 | 29,498 | 30,232 |
| ACL | All 922 | 157,427 | 41,978 | 44,427 |
| EMNLP | Top 100 | 39,852 | 8,432 | 7,971 |
| EMNLP | Mid 100 | 13,486 | 7,557 | 7,950 |
| EMNLP | Bottom 100 | 8,403 | 6,801 | 7,769 |
| EMNLP | All 262 | 49,743 | 17,518 | 17,671 |

Table 3: Data size (in number of article segments).

| Dataset | Nr authors | Training | Valid | Test |
|---|---|---|---|---|
| ACL | Top 100 | 350 | 185 | 197 |
| ACL | Top 200 | 1,315 | 258 | 263 |
| ACL | Mid 200 | 343 | 190 | 199 |
| ACL | Bottom 200 | 180 | 181 | 176 |
| ACL | Top 500 | 1,758 | 413 | 428 |
| ACL | All 922 | 1,604 | 697 | 710 |
| EMNLP | Top 100 | 325 | 90 | 89 |
| EMNLP | Mid 100 | 130 | 83 | 88 |
| EMNLP | Bottom 100 | 86 | 78 | 86 |
| EMNLP | All 262 | 345 | 191 | 86 |

Table 4: Data size (in number of articles).

Tables 3 and 4 show the number of article segments and the number of articles we end up with after extracting the features and splitting the ACL and EMNLP datasets, respectively, into the training, validation, and test sets, in each of the experimental settings.

### 5.1 Metrics

Depending on whether we see the list of authors that contributed to an article as a sorted or unsorted list, a machine learning model that can predict this set of authors can be designed either as a multi-label classifier or as a ranking model. We use the former, and disregard the order of the authors of an article, assuming it is not always relevant to the quantity of each author's contribution to the article text, thus representing the authors for an article as an unordered set of labels.

We do, however, consider the order of the predicted classes in the model's output. For evaluating our model, we use both the performance

metrics that are suited for multi-class classification (where we treat the model's predictions as unordered sets) and metrics suited to ranking problems, which are generally used in information retrieval (where we see the list of model predictions as a sorted list, ranked according to the Softmax probabilities in the model's output). These performance metrics are as follows:

- Accuracy@$k$ - computed as the number of articles for which at least one true author was in the top $k$ predicted authors. We use $k = 10$ as this number was shown to perform well in other search and retrieval tasks (Spink and Jansen, 2004).
- Mean Average Precision (MAP)
$$MAP = \frac{1}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} \frac{1}{r} \sum_{k=1}^{r} P@k$$
where $\mathcal{A}$ is the set of articles, and precision at rank $k$ $P@k$ is the number of correct authors within the first $k$ predicted authors relative to the number of predictions ($k$). Here too we use a maximum rank of $r = 10$.
- Mean Average Recall (MAR)
$$MAR = \frac{1}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} \frac{1}{r} \sum_{k=1}^{r} R@k$$
where recall at rank $k$ $R@k$ is defined as the number of correct authors within the first $k$ predicted authors relative to the total number of true authors ($r = 10$ as before).
- Mean Reciprocal Rank (MRR)
$$MRR = \frac{1}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} \frac{1}{r_a}$$
where $r_a$ is the rank at which the first correct predicted author was found for article $a \in \mathcal{A}$.

Since our datapoints are article segments rather than full articles, the measured performance on the model's output will be with respect to these segments: for example, an accuracy of 10% denotes that 10% of the *segments* were correctly classified.

We additionally adapt these metrics to be able to also output the performance at *article-level*, which allows us to properly compare it to our baseline's performance (which is measured on full articles). We accomplish this by grouping the article segments in our test set according to the article they were extracted from, and for each article, order all probabilites in the network's output for the given article, and consider the top predicted classes across this global ranking as the model's predictions for the target article.

### 5.2 Experiment Settings

In order to understand the contribution of each feature for predicting the authors of a paper, we perform an ablation study - isolating and using in turn only certain features and combinations of features and deliberately not using others.

In a first experiment, we compare our model's performance on both the entire set of authors in each dataset, and only a subset of the authors, e.g., selecting only the top most prolific ones (top-100, top-200, and so on) and ignoring the rest. In this experiment, we only consider articles authored by these selected authors, both for training and test. We expect performance should be higher on this subset where rare authors are discarded, since the pronounced skewedness of the data makes so that for many authors in the tail of the distribution there are very few data points to train on.

In a second experiment, we group our features according to the level of the text that they represent: content level (word sequences), style level (stopwords and POS sequences) or citation level (cited authors). We run our model using the features in each of the groups in turn, and ignoring the rest. The measured performance in each separate setting should be an indicator of the importance of the specific feature (or feature combination) and of the aspect of the article that it captures.

Finally, we also experiment separately with our text features, in order to understand which part of the scientific text is more specific to its author, and how much of the text is really useful in predicting the author. We look separately first at the entire body of the article, secondly at only the title and abstract, and thirdly only at the context of the citations that occur in the text - assuming these might be useful by giving away something about the author's citation pattern that cannot be inferred from the references list alone. Citation contexts have been shown to capture useful information in previous studies focusing on text summarization (Qazvinian et al., 2010; Abu-Jbara and Radev, 2011; Qazvinian and Radev, 2008), document indexing (Ritchie et al., 2008), keyphrase extraction (Gollapalli and Caragea, 2014; Caragea et al., 2014), and author influence in digital libraries (Kataria et al., 2011).

These last experiments aiming to analyze feature importance are performed in the small-scale setting, where only the top 100 authors are considered. We report all results both at segment level

| Features | Acc | MAP | MAR | MRR |
|---|---|---|---|---|
| All features (top100cls) | **84.65** | 9.87 | 46.22 | 25.28 |
| All features (top200cls) | 70.00 | **18.07** | 54.41 | 48.96 |
| All features (top500cls) | 63.77 | 16.82 | 43.34 | 41.60 |
| All features (mid200cls) | 62.38 | 14.02 | 49.44 | 40.78 |
| All features (last200cls) | 54.18 | 11.38 | 43.22 | 35.21 |
| All features (922cls) | 52.41 | 13.78 | 31.29 | 33.52 |
| Content + Ref (100cls) | 75.38 | 16.74 | **58.42** | **49.46** |
| Content (100cls) | 49.87 | 8.87 | 33.23 | 25.34 |
| Style ft. | 25.21 | 3.22 | 13.81 | 9.20 |
| Content (100cls) | 49.87 | 8.87 | 33.23 | 25.34 |
| Title+abstr. | 16.19 | 2.71 | 9.52 | 6.22 |
| Ref. contexts | 35.53 | 5.07 | 21.27 | 13.54 |

Table 5: Results on ACL dataset (segment level).

| Features | Acc | MAP | MAR | MRR |
|---|---|---|---|---|
| All features (top100cls) | **87.88** | **24.71** | **78.37** | **72.33** |
| All features (top200cls) | 66.78 | 16.93 | 51.10 | 46.20 |
| All features (top500cls) | 60.74 | 15.36 | 39.71 | 38.68 |
| All features (mid200cls) | 54.82 | 11.14 | 40.38 | 32.99 |
| All features (last200cls) | 50.00 | 8.61 | 36.24 | 25.53 |
| All features (922cls) | 47.59 | 11.74 | 27.04 | 28.77 |
| Content + Ref (100cls) | 76.13 | 16.37 | 57.50 | 48.73 |
| Content (100cls) | 48.29 | 10.51 | 37.16 | 31.44 |
| Style ft. | 26.70 | 3.18 | 14.25 | 9.26 |
| References | 86.67 | 24.35 | 78.31 | 71.37 |
| Content (100cls) | 48.29 | 10.51 | 37.16 | 31.44 |
| Title+abstr. | 18.08 | 2.56 | 10.23 | 6.34 |
| Ref. contexts | 30.95 | 4.15 | 11.20 | 18.16 |
| Baseline (100cls) | 54.86 | 11.57 | 42.76 | 34.37 |
| Baseline (200cls) | 55.70 | 12.08 | 40.82 | 34.48 |
| Baseline (500cls) | 56.17 | 12.73 | 38.01 | 34.65 |
| Baseline (922cls) | 57.25 | 14.79 | 28.25 | 35.67 |

Table 6: Results on ACL dataset (article level).

and at article level, with the exception of the experiment where the only features used are references, which are article-level features.

### 5.3 Results

Table 5 shows the classification results measured on our data points consisting of article segments on ACL, whereas Table 6 contains the values of the metrics aggregated at article-level on the same ACL dataset. Tables 7 and 8 show similar results on the EMNLP dataset. Underlined scores are best within each group and bold scores are best overall. Figure 5 shows the accuracy of our best deep learning model on ACL and EMNLP as we increase the number of authors (classes), compared with the baseline model that considers only references and with a random model.

As can be seen from the tables and figure, results show that references are still the features that by far contribute the most to predicting the author(s) of an article. The importance of the references list also contributes to the strong performance of the baseline, which is able to correctly predict authors for as much as $54.86\%$ of the articles on ACL and $57.80\%$ on EMNLP, for the top $100$ classes. Interestingly, the baseline's performance is comparable to the results reported by (Hill and Provost, 2003) who use a similar method, even on a different dataset. Feeding the extracted references into the deep network further boosts the predictive power of the references feature, reaching on its own an accuracy of $86.67\%$ on ACL and $78.49\%$ on EMNLP in an experiment looking at the top $100$ most prolific authors in each dataset.

| Features | Acc | MAP | MAR | MRR |
|---|---|---|---|---|
| All features (top100cls) | 79.97 | **19.98** | **60.14** | 55.33 |
| All features (mid100cls) | 57.10 | 12.02 | 43.70 | 35.52 |
| All features (last100cls) | 62.96 | 12.02 | 46.10 | 36.08 |
| All features (262cls) | 59.47 | 15.65 | 37.42 | 37.90 |
| Content + Ref (100cls) | **80.27** | 19.93 | 59.88 | **55.89** |
| Content (100cls) | 48.59 | 8.28 | 28.66 | 22.15 |
| Style ft. | 25.98 | 3.96 | 12.95 | 10.29 |
| Content (100cls) | 48.59 | 8.28 | 28.66 | 22.15 |
| Title+abstr. | 15.48 | 2.14 | 7.38 | 6.15 |
| Ref. contexts | 21.47 | 2.73 | 9.20 | 7.39 |

Table 7: Results on EMNLP dataset (segment level).

| Features | Acc | MAP | MAR | MRR |
|---|---|---|---|---|
| All features (top100cls) | **78.49** | 17.70 | 56.09 | 48.74 |
| All features (mid100cls) | 51.68 | 11.09 | 39.49 | 33.91 |
| All features (last100cls) | 51.16 | 9.28 | 37.96 | 27.11 |
| All features (262cls) | 57.86 | 13.84 | 34.65 | 33.81 |
| Content + Ref (100cls) | 77.41 | 18.19 | 56.88 | 50.60 |
| Content (100cls) | 56.98 | 11.27 | 37.81 | 31.08 |
| Style ft. | 31.82 | 4.93 | 16.21 | 13.23 |
| References | **78.49** | **18.79** | **58.04** | **52.26** |
| Content (100cls) | 56.98 | 11.27 | 37.81 | 31.08 |
| Title+abstr. | 10.75 | 0.82 | 3.47 | 2.32 |
| Ref. contexts | 15.90 | 16.83 | 6.21 | 4.25 |
| Baseline (100 cls) | 57.80 | 12.59 | 42.31 | 36.01 |
| Baseline (262 cls) | 58.90 | 15.06 | 24.71 | 36.92 |

Table 8: Results on EMNLP dataset (article level).

The more general text features (content level) are the second best predictor, whereas the style-level features come last. Even if much less predictive than references, these text-based features are still far better than chance at predicting the true authors. For example, on ACL, in the setting with $100$ possible classes, the expected accuracy of a random predictor (according to our definition of accuracy), would be around $10\%$, whereas when using all $922$ classes, the chance accuracy is $1\%$.

With regard to the parts of the article text that seem to be most predictive, reference contexts seem to play a more important role than the title and abstract, even though using the full article content still gives the best results on both datasets.

Moreover, as we go from the top $100$ most prolific authors to the last (rare) authors, the performance keeps decreasing. For example, the ac-

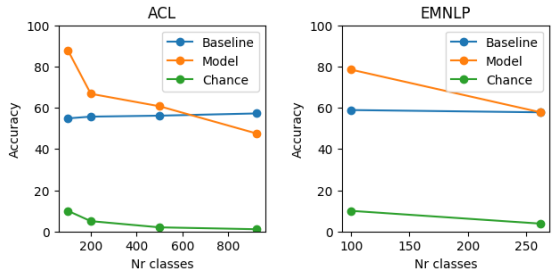


Figure 5: Accuracy with number of authors considered.

curacy on ACL at article level decreases from 87.88% (achieved for top 100 most prolific authors) to 50% (achieved for last 200 rare authors).

### 5.4 Error Analysis

In order to achieve a better understanding of the model's weaknesses and more generally of the difficulties of predicting authors of scientific papers, we examine the set of misclassified articles in the ACL test set, and compute the misclassification rate for an author as the number of their articles for which the model did not assign to them, divided by the total number of articles authored by them. An interesting finding is that the correlation between the rank of the author (in order of their number of written articles) and the misclassification rate is 0.35, showing that more prolific authors tend to be more accurately classified. One of the most misclassified authors in the top 5 most prolific authors is Christopher Manning (40% of articles are misclassifed, among which there are *Accurate Unlexicalized Parsing* and *Deep Learning for NLP (without Magic)*. For the first paper, some of the predicted authors are: Eugene Charniak, Mark Johnson, Lenhart K Schubert, Dan Klein, and Daniel Jurafsky, with Dan Klein being indeed one of the authors. Other 45 articles not authored by Christopher Manning were predicted as being written by this author, possibly due to a large number of his citations in the articles' references list and/or similar keywords with those of Christopher Manning.

## 6 Conclusions and Discussion

We showed that the top most prolific authors of anonymized scientific articles can be predicted with high accuracy, with characteristics of the authors being apparent at all levels of the text: from content to style. Still, the most direct indicator of who the author of a paper might be comes from the papers that are referenced - both from the references themselves and from the citation context they occur in within the content of an article. Our work contributes to the debate about the double-blind reviewing process and aims at contributing toward the rapidly emerging field of *Fairness in AI*. Although we found that the most prolific authors can be inferred with accuracy as high as 87.88% on ACL and 78.49% on EMNLP, the authors with less papers are more and more difficult to infer, which enforces the benefits of the double-blind review in offering any author a fair chance of having their papers accepted in top venues.

The finding that authors of anonymized papers can be predicted with such high accuracy bears important consequences for the way scientific articles are reviewed and published. De-anonymizing articles means compromising the integrity of the review and selection process. The insights into how the authors of an article can be inferred are not only interesting, but could help guide a reconsidered approach of the way we write papers for submission to various venues. Still, the findings and conclusions of this paper should not be seen as a premise that the portions of an article which help a *neural network* identify authorship are the same as those which help a *human reviewer* identify authorship, and are not necessarily expected to inform how humans perform peer review.

In future experiments, more attention to the contribution of each author of the article might lead to further improvements in the prediction performance. In this article, we construct our training datasets as if all parts of an article were written by all authors, which is not accurate, and could even put an upper bound on the network's performance, by providing it with contradictory information during training. Techniques for segmenting the text according to their probable authorship could help improve the method.

## Acknowledgments

The first two authors of this paper (CC and AU) contributed equally. We thank our anonymous reviewers for their constructive comments and feedback, which helped improve our paper. This research is supported by NSF CAREER award 1802358 and NSF CRI award 1823292 to Cornelia Caragea. Any opinions, findings, and conclusions expressed here are those of the authors and do not necessarily reflect the views of NSF.

## References


Amjad Abu-Jbara and Dragomir Radev. 2011. Coherent citation-based summarization of scientific papers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 500–509, Portland, Oregon, USA. Association for Computational Linguistics.

Tim Althoff, Denny Britz, and Zifei Shan. 2013. Authorship attribution in multi-author documents.

Ashton Anderson, Dan Jurafsky, and Daniel A. McFarland. 2012. Towards a computational history of the ACL: 1980-2008. In *Proceedings of the ACL-2012 Special Workshop on Rediscovering 50 Years of Discoveries*, pages 13–21, Jeju Island, Korea. Association for Computational Linguistics.

Douglas Bagnall. 2015. Author identification using multi-headed recurrent neural networks. *arXiv preprint arXiv:1506.04891*.

Steven Bird, Robert Dale, Bonnie J. Dorr, Bryan R. Gibson, Mark Thomas Joseph, Min-Yen Kan, Dongwon Lee, Brett Powley, Dragomir R. Radev, and Yee Fan Tan. 2008. The ACL anthology reference corpus: A reference dataset for bibliographic research in computational linguistics. In *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2008, 26 May - 1 June 2008, Marrakech, Morocco*.

Cornelia Caragea, Florin Adrian Bulgarov, Andreea Godea, and Sujatha Das Gollapalli. 2014. Citation-enhanced keyphrase extraction from research papers: A supervised approach. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1435–1446, Doha, Qatar. Association for Computational Linguistics.

Stephen J Ceci and Douglas Peters. 1984. How blind is blind review? *American Psychologist*, 39(12):1491.

Sujatha Das Gollapalli and Cornelia Caragea. 2014. Extracting keyphrases from research papers using citation networks. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, pages 1629–1635. AAAI Press.

Shawndra Hill and Foster Provost. 2003. The myth of the double-blind review?: author identification using only citations. *Acm Sigkdd Explorations Newsletter*, 5(2):179–184.

Julian Hitschler, Esther van den Berg, and Ines Rehbein. 2017. Authorship attribution with convolutional neural networks and POS-eliding. In *Proceedings of the Workshop on Stylistic Variation*, pages 53–58, Copenhagen, Denmark. Association for Computational Linguistics.

Mohammadreza Hojat, Joseph S. Gonnella, and Addeane S. Caelleigh. 2003. Impartial judgment by the "gatekeepers" of science: fallibility and accountability in the peer review process. *Advances in health sciences education : theory and practice*, 8 1:75–96.

Saurabh Kataria, Prasenjit Mitra, Cornelia Caragea, and C. Lee Giles. 2011. Context sensitive topic models for author influence in document networks. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Moshe Koppel, Jonathan Schler, and Shlomo Argamon. 2009. Computational methods in authorship attribution. *Journal of the American Society for information Science and Technology*, 60(1):9–26.

Kim Luyckx and Walter Daelemans. 2008. Authorship attribution and verification with many authors and limited data. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 513–520, Manchester, UK. Coling 2008 Organizing Committee.

A.J. Meadows. 1998. *Communicating Research*. Academic Press, San Diego.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Vahed Qazvinian and Dragomir R. Radev. 2008. Scientific paper summarization using citation summary networks. In *Proceedings of the 22Nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 689–696, Stroudsburg, PA, USA. Association for Computational Linguistics.

Vahed Qazvinian, Dragomir R. Radev, and Arzucan Özgür. 2010. Citation summarization through keyphrase extraction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 895–903, Stroudsburg, PA, USA. Association for Computational Linguistics.

Andi Rexha, Stefan Klampfl, Mark Kröll, and Roman Kern. 2015. Towards authorship attribution for bibliometrics using stylometric features. In *CLBib@ ISSI*, pages 44–49.

Dylan Rhodes. 2015. Author attribution with cnns. *Avaiable online: https://www. semanticscholar. org/paper/Author-Attribution-with-Cnn-s-Rhodes/0a904f9d6b47dfc574f681f4d3b41bd840871b6f/pdf (accessed on 22 August 2016)*.

Anna Ritchie, Stephen Robertson, and Simone Teufel. 2008. Comparing citation contexts for information retrieval. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, pages 213–222, New York, NY, USA. ACM.

Michal Rosen-Zvi, Thomas Griffiths, Mark Steyvers, and Padhraic Smyth. 2004. The author-topic model for authors and documents. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, UAI '04, pages 487–494, Arlington, Virginia, United States. AUAI Press.

Yanir Seroussi, Fabian Bohnert, and Ingrid Zukerman. 2012. Authorship attribution with author-aware topic models. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 264–269, Jeju Island, Korea. Association for Computational Linguistics.

Prasha Shrestha, Sebastian Sierra, Fabio González, Manuel Montes, Paolo Rosso, and Thamar Solorio. 2017. Convolutional neural networks for authorship attribution of short texts. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 669–674, Valencia, Spain. Association for Computational Linguistics.

Amanda Spink and Bernard J Jansen. 2004. *Web search: Public searching of the Web*, volume 6. Springer Science & Business Media.

Andrew Tomkins, Min Zhang, and William D Heavlin. 2017. Reviewer bias in single-versus double-blind peer review. *Proceedings of the National Academy of Sciences*, 114(48):12708–12713.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 252–259.

Alex Williamson. 2003. What will happen to peer review? *Learned Publishing*, 16:15–20.

# Uncover Sexual Harassment Patterns from Personal Stories by Joint Key Element Extraction and Categorization

**Yingchi Liu**[1], **Quanzhi Li**[1], **Marika Cifor**[2], **Xiaozhong Liu**[3], **Qiong Zhang**[1] **and Luo Si**[1]
[1]Alibaba Group, US
[2]University of Washington, USA
[3]Indiana University Bloomington, USA
{yingchi.liu, quanzhi.li, qz.zhang,luo.si}@alibaba-inc.com
mcifor@uw.edu, liu237@indiana.edu

## Abstract

The number of personal stories about sexual harassment shared online has increased exponentially in recent years. This is in part inspired by the #MeToo and #TimesUp movements. Safecity is an online forum for people who experienced or witnessed sexual harassment to share their personal experiences. It has collected >10,000 stories so far. Sexual harassment occurred in a variety of situations, and categorization of the stories and extraction of their key elements will provide great help for the related parties to understand and address sexual harassment. In this study, we manually annotated those stories with labels in the dimensions of location, time, and harassers' characteristics, and marked the key elements related to these dimensions. Furthermore, we applied natural language processing technologies with joint learning schemes to automatically categorize these stories in those dimensions and extract key elements at the same time. We also uncovered significant patterns from the categorized sexual harassment stories. We believe our annotated data set, proposed algorithms, and analysis will help people who have been harassed, authorities, researchers and other related parties in various ways, such as automatically filling reports, enlightening the public in order to prevent future harassment, and enabling more effective, faster action to be taken.

## 1 Introduction

Sexual violence, including harassment, is a pervasive, worldwide problem with a long history. This global problem has finally become a mainstream issue thanks to the efforts of survivors and advocates. Statistics show that girls and women are put at high risk of experiencing harassment. Women have about a 3 in 5 chance of experiencing sexual harassment, whereas men have slightly less than 1 in 5 chance (Quinnipiac, 2017; EEOC, 2018; Goldstein, 2018). While women in developing countries are facing distinct challenges with sexual violence (Lea et al., 2017), however sexual violence is ubiquitous. In the United States, for example, there are on average >300,000 people who are sexually assaulted every year (Morgan and Truman, 2018). Additionally, these numbers could be underestimated, due to reasons like guilt, blame, doubt and fear, which stopped many survivors from reporting (Griffith, 2018). Social media can be a more open and accessible channel for those who have experienced harassment to be empowered to freely share their traumatic experiences and to raise awareness of the vast scale of sexual harassment, which then allows us to understand and actively address abusive behavior as part of larger efforts to prevent future sexual harassment. The deadly gang rape of a medical student on a Delhi bus in 2012 was a catalyst for protest and action, including the development of Safecity, which uses online and mobile technology to work towards ending sexual harassment and assault. More recently, the #MeToo and #TimesUp movements, further demonstrate how reporting personal stories on social media can raise awareness and empower women. Millions of people around the world have come forward and shared their stories. Instead of being bystanders, more and more people become up-standers, who take action to protest against sexual harassment online. The stories of people who experienced harassment can be studied to identify different patterns of sexual harassment, which can enable solutions to be developed to make streets safer and to keep women and girls more secure when navigating city spaces (Karlekar and Bansal, 2018). In this paper, we demonstrated the application of natural language processing (NLP) technologies to uncover harassment patterns from social media data. We made three key contributions:



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2328–2337, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

1. Safecity[1] is the largest publicly-available online forum for reporting sexual harassment (Karlekar and Bansal, 2018). We annotated about 10,000 personal stories from Safecity with the key elements, including information of harasser (i.e. the words describing the harasser), time, location and the trigger words (i.e. the phrases indicate the harassment that occurred). The key elements are important for studying the patterns of harassment and victimology (Griffith, 2018; Ceccato, 2017). Furthermore, we also associated each story with five labels that characterize the story in multiple dimensions (i.e. age of harasser, single/multiple harasser(s), type of harasser, type of location and time of day). The annotation data are available online.[2]

2. We proposed joint learning NLP models that use convolutional neural network (CNN) (Lecun and Bengio, 1995) and bi-directional long short-term memory (BiLSTM) (Schuster and Paliwal, 1997; Hochreiter and Schmidhuber, 1997) as basic units. Our models can automatically extract the key elements from the sexual harassment stories and at the same time categorize the stories in different dimensions. The proposed models outperformed the single task models, and achieved higher than previously reported accuracy in classifications of harassment forms (Karlekar and Bansal, 2018).

3. We uncovered significant patterns from the categorized sexual harassment stories.

## 2 Related Work

Conventional surveys and reports are often used to study sexual harassment, but harassment on these is usually under-reported (Goldstein, 2018; Griffith, 2018). The high volume of social media data available online can provide us a much larger collection of firsthand stories of sexual harassment. Social media data has already been used to analyze and predict distinct societal and health issues, in order to improve the understanding of wide-reaching societal concerns, including mental health, detecting domestic abuse, and cyberbullying (Balani and De Choudhury, 2015; Schrading et al., 2015; Ziegele et al., 2018; Agrawal and Awekar, 2018).

There are a very limited number of studies on sexual harassment stories shared online. Karlekar and Bansal (2018) were the first group to our knowledge that applied NLP to analyze large amount ( ∼10,000) of sexual harassment stories. Although their CNN-RNN classification models demonstrated high performance on classifying the forms of harassment, only the top 3 majority forms were studied. In order to study the details of the sexual harassment, the trigger words are crucial. Additionally, research indicated that both situational factors and person (or individual difference) factors contribute to sexual harassment (Hitlan et al., 2009). Therefore, the information about perpetrators needs to be extracted as well as the location and time of events. Karlekar and Bansal (2018) applied several visualization techniques in order to capture such information, but it was not obtained explicitly. Our preliminary research demonstrated automatic extraction of key element and story classification in separate steps (Liu et al., 2019). In this paper, we proposed joint learning NLP models to directly extract the information of the harasser, time, location and trigger word as key elements and categorize the harassment stories in five dimensions as well. Our approach can provide an avenue to automatically uncover nuanced circumstances informing sexual harassment from online stories.




Figure 1: An example of the annotation.

## 3 Data Collection and Annotation

We obtained 9,892 stories of sexual harassment incidents that was reported on Safecity. Those stories include a text description, along with tags of the forms of harassment, e.g. commenting, ogling and groping. A dataset of these stories was published by Karlekar and Bansal (2018). In addition to the forms of harassment, we manually annotated each story with the key elements (i.e. "harasser", "time", "location", "trigger"), because they are essential to uncover the harassment patterns. An example is shown in Figure 1. Furthermore, we also assigned each story classification labels in five dimensions (Table 1). The detailed definitions of classifications in all dimensions are explained below.

**Age of Harasser**: Individual difference such as age can affect harassment behaviors. Therefore,

[1] https://safecity.in

[2] https://github.com/alievent/harassment-analysis Please follow Safecity guidelines for usage.

| Dimensions | Classes/Groups |
|---|---|
| Age of Harasser | 0 - unspecified, 1 - teenagers or young people, 2 - adults |
| Single/Multiple Harasser(s) | 0 - unspecified, 1 - single harasser, 2 - multiple harassers |
| Type of Harasser | 0 - unspecified, 1 - relative, 2 - teacher, 3 - classmate, 4 - friend, 5 - neighbor, 6 - conductor/driver, 7 - work-related, 8 - police/guard, 9 - other |
| Type of Location | 0 - unspecified, 1 - street, 2 - transportation, 3 - station/stop, 4 - private places, 5 - shopping places, 6 - neighborhood, 7 - park, 8 - hotel, 9 - bush/woods, 10 - parking lot, 11 - in/near school, 12 - restaurant, 13 - other |
| Time of Day | 0 - unspecified, 1 - day (5am to 6pm), 2 - evening or night |

Table 1: Definition of classes in different dimensions about sexual harassment.

we studied the harassers in two age groups, young and adult. Young people in this paper refer to people in the early 20s or younger.

**Single/Multiple Harasser(s)**: Harassers may behave differently in groups than they do alone.

**Type of Harasser**: Person factors in harassment include the common relationships or titles of the harassers. Additionally, the reactions of people who experience harassment may vary with the harassers' relations to themselves (Griffith, 2018). We defined 10 groups with respects to the harassers' relationships or titles. We put conductors and drivers in one group, as they both work on the public transportation. Police and guards are put in the same category, because they are employed to provide security. Manager, supervisors, and colleagues are in the work-related group. The others are described by their names.

**Type of Location**: It will be helpful to reveal the places where harassment most frequently occurs (Ceccato, 2017; Karlekar and Bansal, 2018). We defined 14 types of locations. "Station/stop" refers to places where people wait for public transportation or buy tickets. Private places include survivors' or harassers' home, places of parties and etc. The others are described by their names.

**Time of Day**: The time of an incident may be reported as *"in evening"* or at a specific time, e.g. *"10 pm"*. We considered that 5 am to 6 pm as day time, and the rest of the day as the night.

Because many of the stories collected are short, many do not contain all of the key elements. For example, *"A man came near to her tried to be physical with her ."*. The time and location are unknown from the story. In addition, the harassers were strangers to those they harassed in many cases. For instance, *"My friend was standing in the queue to pay bill and was ogled by a group of boys."*, we can only learn that there were multiple young harassers, but the type of harasser is unclear. The missing information is hence marked as "unspecified". It is different from the label "other", which means the information is provided but the number of them is too small to be represented by a group, for example, a "trader".

All the data were labeled by two annotators with training. Inter-rater agreement was measured by Cohen's kappa coefficient, ranging from 0.71 to 0.91 for classifications in different dimensions and 0.75 for key element extraction (details can refer to Table 1 in supplementary file). The disagreements were reviewed by a third annotator and a final decision was made.

## 4 Proposed Models

The key elements can be very informative when categorizing the incidents. For instance, in Figure 1, with identified key elements, one can easily categorize the incident in dimensions of "age of harasser" (adult), "single/multiple harasser(s)" (single), "type of harasser" (unspecified), "type of location" (park) , "time of day" (day time). Therefore, we proposed two joint learning schemes to extract the key elements and categorize the incidents together. In the models' names, "J", "A", "SA" stand for joint learning, attention, and supervised attention, respectively.

### 4.1 CNN Based Joint Learning Models

In Figure 2, the first proposed structure consists of two layers of CNN modules.

**J-CNN:** To predict the type of key element, it is essential for the CNN model to capture the context information around each word. Therefore, the word along with its surrounding context of a fixed window size was converted into a context sequence. Assuming a window size of $2l + 1$ around the target word $w_0$, the context sequence is $[(w_{-l}, w_{-l+1}, ...w_0, ...w_{l-1}, w_l)]$, where $w_i (i \in [-l, l])$ stands for the $ith$ word from $w_0$.

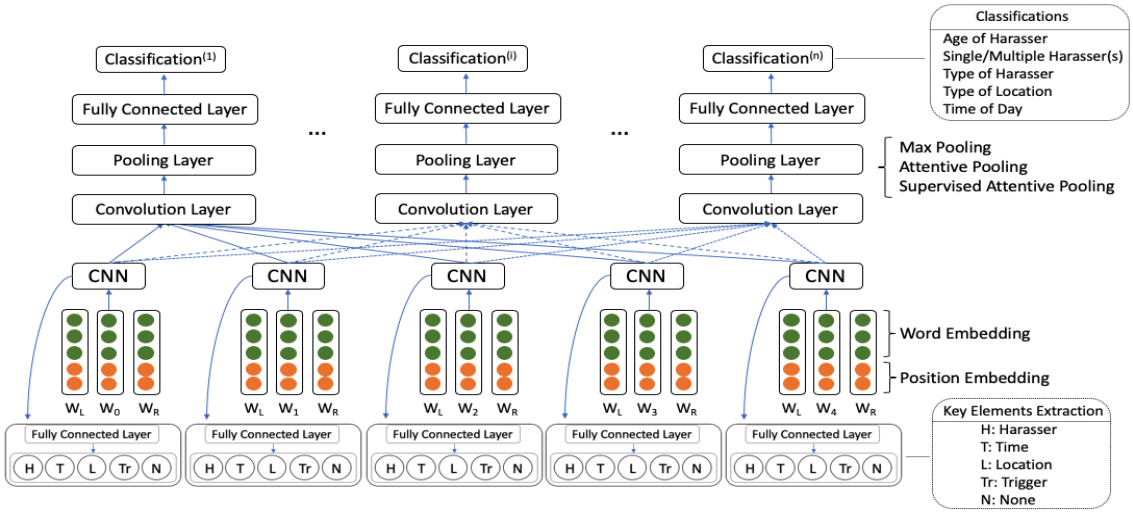


Figure 2: CNN based Joint learning Model. $W_L$ and $W_R$ are the left and right context around each word.

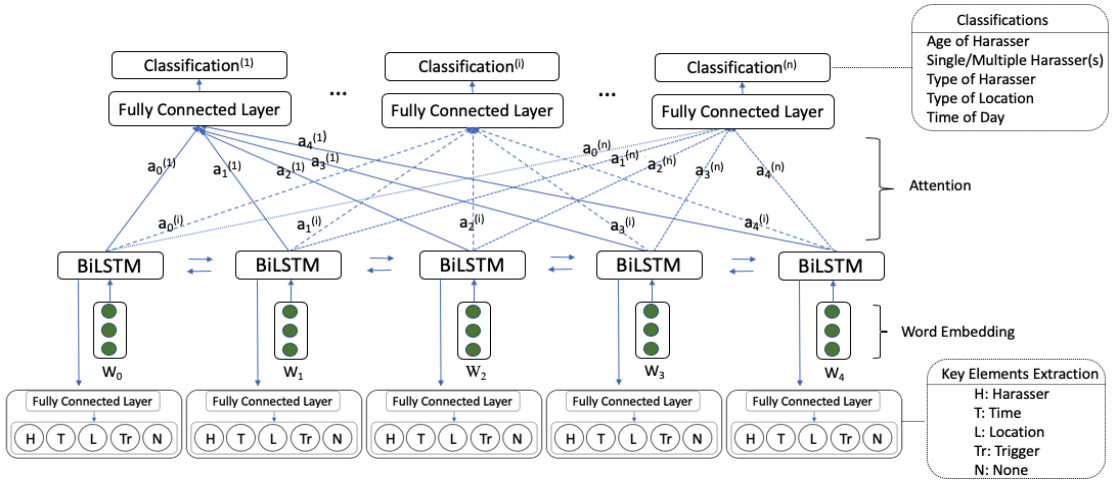


Figure 3: BiLSM based Joint Learning Model. Here we use an input of five words as an example.

Because the context of the two consecutive words in the original text are only off by one position, it will be difficult for the CNN model to detect the difference. Therefore, the position of each word in this context sequence is crucial information for the CNN model to make the correct predictions (Nguyen and Grishman, 2015). That position was embedded as a $p$ dimensional vector, where $p$ is a hyperparameter. The position embeddings were learned at the training stage. Each word in the original text was then converted into a sequence of the concatenation of word and position embeddings. Such sequence was fed into the CNN modules in the first layer of the model, which output the high level word representation ($h_i, i \in [0, n-1]$, where n is the number of input words). The high level word representation was then passed into a fully connected layer, to predict the key element type for the word. The CNN modules in this layer share the same parameters.

We input the sequence of high level word representations ($h_i$) from the first layer into another layer of multiple CNN modules to categorize the harassment incident in each dimension (Figure 2). Inside each CNN module, the sequence of word representations were first passed through a convolution layer to generate a sequence of new feature vectors ($C = [c_0, c_1, ...c_q]$). This vector sequence ($C$) was then fed into a max pooling layer. This is followed by a fully connected layer. Modules in this layer do not share parameters across classification tasks.

**J-ACNN:** We also experimented with attentive pooling, by replacing the max pooling layer. The attention layer aggregates the sequence of feature vectors ($C$) by measuring the contribution of each

vector to form the high level representation of the harassment story. Specifically,

$$u_i = tanh(W_\omega c_i + b_\omega) \quad (1)$$

$$\alpha_i = \frac{exp(u_i^T u_\omega)}{\sum_i exp(u_i^T u_w)} \quad (2)$$

$$v = \sum_i \alpha_i c_i \quad (3)$$

That is, a fully connected layer with non-linear activation was applied to each vector $c_i$ to get its hidden representation $u_i$. The similarity of $u_i$ with a context vector $u_w$ was measured and get normalized through a softmax function, as the importance weight $\alpha_i$. The final representation of the incident story $v$ was an aggregation of all the feature vectors weighted by $\alpha_i$. $W_\omega$, $b_\omega$ and $u_w$ were learned during training.

The final representation ($v$) was passed into one fully connected layer for each classification task. We also applied different attention layers for different classifications, because the classification modules categorize the incident in different dimensions, their focuses vary. For example, to classify "time of day", one needs to focus on the time phrases, but pays more attention to harassers when classifying "age of harasser".

**J-SACNN:** To further exploit the information of the key elements, we applied supervision (Zhao et al., 2018) to the attentive pooling layer, with the annotated key element types of the words as ground truth. For instance, in classification of "age of harasser", the ground truth attention labels for words with key element types of "harasser" are 1 and others are 0. To conform to the CNN structure, we applied convolution to the sequence of ground truth attention labels, with the same window size ($w$) that was applied to the word sequence (Eq. 4).

$$\alpha^*(t) = W \circ [e_t : e_{t+w-1}] \quad (4)$$

where $\circ$ is element-wise multiplication, $e_t$ is the ground truth attention label, and the $W \in R^{w \times 1}$ is a constant matrix with all elements equal to 1. $\alpha^*$ was normalized through a softmax function and used as ground truth weight values of the vector sequence ($C$) output from the convolution layer. The loss was calculated between learned attention $\alpha$ and $\alpha^*$ (Eq. 5), and added to the total loss.

$$E(\alpha, \alpha^*) = \sum_{i=0}^{q} (\alpha_i - \alpha_i^*)^2 \quad (5)$$

### 4.2 BiLSTM Based Joint Learning Models

**J-BiLSTM:** The model input the sequence of word embeddings to the BiLSTM layer. To extract key elements, the hidden states from the forward and backward LSTM cells were concatenated and used as word representations to predict the key element types.

To classify the harassment story in different dimensions, concatenation of the forward and backward final states of BiLSTM layer was used as document level representation of the story.

**J-ABiLSTM:** We also experimented on BiLSTM model with the attention layer to aggregate the outputs from BiLSTM layer (Figure 3). The aggregation of the outputs was used as document level representation.

**J-SABiLSTM:** Similarly, we experimented with the supervised attention.

In all the models, softmax function was used to calculate the probabilities at the prediction step, and the cross entropy losses from extraction and classification tasks were added together. In case of supervised attention, the loss defined in Eq. 5 was added to the total loss as well. We applied the stochastic gradient descent algorithm with mini-batches and the AdaDelta update Rule (rho=0.95 and epsilon=1e-6) (Zeiler, 2012; Feng et al., 2016). The gradients were computed using back-propagation. During training, we also optimized the word and position embeddings.

## 5 Experiments and Results

### 5.1 Experimental Settings

**Data Splits**: We used the same splits of train, develop, and test sets used by Karlekar and Bansal (Karlekar and Bansal, 2018), with 7201, 990 and 1701 stories, respectively. In this study, we only considered single label classifications.

**Baseline Models**: CNN and BiLSTM models that perform classification and extraction separately were used as baseline models. In classification, we also experimented with BiLSTM with the attention layer. To demonstrate that the improvement came from joint learning structure rather the two layer structure in J-CNN, we investigated the same model structure without training on key element extraction. We use J-CNN* to denote it.

**Preprocess**: All the texts were converted to lowercase and preprocessed by removing non-alphanumeric characters, excluding ". ! ? " . The word embeddings were pre-trained using fastText

(Bojanowski et al., 2016) with dimension equaling 100.

**Hyperparameters**: For the CNN model, the filter size was chosen to be (1,2,3,4), with 50 filters per filter size. Batch size was set to 50 and the dropout rate was 0.5. The BiLSTM model comprises two layers of one directional LSTM. Every LSTM cell has 50 hidden units. The dropout rate was 0.25. Attention size was 50.

### 5.2 Results and Discussions

We compared joint learning models with the single task models. Results are averages from five experiments. Although not much improvement was achieved in key element extraction (Figure 2), classification performance improved significantly with joint learning schemes (Table 3). Significance t-test results are shown in Table 2 in the supplementary file.

**BiLSTM Based Models:** Joint learning BiLSTM with attention outperformed single task BiLSTM models. One reason is that it directed the attention of the model to the correct part of the text. For example,

**S1:** *"when i was returning my home after finishing my class . i was in queue to get on the micro bus and there was a girl opposite to me just then a young man tried to touch her on the breast ."*

**S2:** *"when i was returning my home after finishing my class . i was in queue to get on the micro bus and there was a girl opposite to me just then a young man tried to touch her on the breast ."*

**S3:** *"when i was returning my home after finishing my class . i was in queue to get on the micro bus and there was a girl opposite to me just then a young man tried to touch her on the breast ."*

In S1, the regular BiLSTM with attention model for classification on "age of harasser" put some attention on phrases other than the harasser, and hence aggregated noise. This could explain why the regular BiLSTM model got lower performance than the CNN model. However, when training with key element extractions, it put almost all attention on the harasser "young man" (S2), which helped the model make correct prediction of "young harasser". When predicting the "type of location" (S3), the joint learning model directed its attention to "micro bus".

**CNN Based Models:** Since CNN is efficient for capturing the most useful information (Chen et al., 2015), it is quite suitable for the classification tasks in this study. It achieved better performance than the BiLSTM model. The joint learning method boosted the performance even higher. This is because the classifications are related to the extracted key elements, and the word representation learned by the first layer of CNNs (Figure 2) is more informative than word embedding. By plotting of t-SNEs (Maaten and Hinton, 2008) of the two kinds of word vectors, we can see the word representations in the joint learning model made the words more separable (Figure 1 in supplementary file). In addition, no improvement was found with the J-CNN* model, which demonstrated the joint learning with extraction is essential for the improvement.

With supervised attentive pooling, the model can get additional knowledge from key element labels. It helped the model in cases when certain location phrases were mentioned but the incidents did not happen at those locations. For instance, *"I*

| | Accuracy | Macro F1 |
|---|---|---|
| BiLSTM | 92.1 | 79.9 |
| CNN | **92.6** | 79.5 |
| J-BiLSTM | 92.1 | 79.2 |
| J-ABiLSTM | 92.0 | 79.7 |
| J-SABiLSTM | 92.0 | 79.1 |
| J-CNN | 92.5 | 80.1 |
| J-ACNN | 92.4 | **80.4** |
| J-SACNN | 92.4 | 80.1 |

Table 2: Key element extraction results.

| | | Age | Single/ Multiple | Type of Harasser | Loc | Time |
|---|---|---|---|---|---|---|
| Accuracy | BiLSTM | 90.7 | 91.1 | 91.0 | 80.3 | 97.0 |
| | ABiLSTM | 90.0 | 91.8 | 91.4 | 81.6 | 97.0 |
| | CNN | 91.6 | 92.8 | 93.2 | 83.6 | 97.4 |
| | J-CNN* | 91.5 | 92.7 | 92.3 | 82.6 | 97.2 |
| | J-BiLSTM | 90.8 | 91.8 | 91.3 | 78.3 | 94.8 |
| | J-ABiLSTM | 92.4 | **94.0** | 92.3 | **85.1** | 97.7 |
| | J-SABiLSTM | 92.4 | 93.7 | 92.1 | 84.8 | 97.4 |
| | J-CNN | 92.5 | 93.8 | **93.3** | 84.2 | 98.0 |
| | J-ACNN | **92.8** | 93.3 | 93.1 | 84.2 | 97.9 |
| | J-SACNN | 92.5 | 93.8 | 92.7 | 83.1 | **98.0** |
| Macro F1 | BiLSTM | 90.4 | 90.3 | 38.6 | 44.3 | 93.8 |
| | ABiLSTM | 89.7 | 91.0 | 33.5 | 45.5 | 93.6 |
| | CNN | 91.5 | 92.1 | 46.7 | 48.1 | 94.4 |
| | J-CNN* | 91.4 | 92.0 | 46.8 | 45.1 | 94.5 |
| | J-BiLSTM | 90.4 | 91.1 | 34.1 | 34.1 | 87.7 |
| | J-ABiLSTM | 92.4 | **93.4** | 37.9 | 48.5 | 95.1 |
| | J-SABiLSTM | 92.4 | 93.0 | 37.8 | 48.5 | 94.4 |
| | J-CNN | 92.4 | 93.1 | 49.5 | 48.6 | **95.6** |
| | J-ACNN | **92.8** | 92.8 | 47.7 | 48.5 | 95.3 |
| | J-SACNN | 92.4 | **93.4** | **52.8** | **50.5** | 95.2 |

Table 3: Classification accuracy and macro F1 of the models. The best scores are in bold.

| Models | commenting | ogling | groping |
|---|---|---|---|
| CNN* | 80.9 | 82.2 | 86.0 |
| BiLSTM* | 81.0 | 82.2 | 86.2 |
| CNN+RNN* | 81.6 | 84.1 | 86.5 |
| J-BiLSTM | 81.7 | 83.3 | 87.1 |
| J-ABiLSTM | 82.1 | 83.2 | 87.9 |
| J-SABiLSTM | 82.4 | 83.1 | 87.9 |
| J-CNN | 81.9 | **84.4** | 87.4 |
| J-ACNN | **82.6** | 83.8 | 88.1 |
| J-SACNN | 82.3 | 83.6 | **88.7** |

Table 4: Harassment form classification accuracy of models. * Reported by Karlekar and Bansal (2018)

*was followed on my way home .*", max pooling will very likely to predict it as "private places". But, it is actually unknown. In other cases, with supervised attentive pooling, the model can distinguish "metro" and "metro station", which are "transportation" and "stop/station" respectively. Therefore, the model further improved on classifications on "type of location" with supervised attention in terms of macro F1. For some tasks, like "time of day", there are fewer cases with such disambiguation and hence max pooling worked well. Supervised attention improved macro F1 in location and harasser classifications, because it made more correct predictions in cases that mentioned location and harasser. But the majority did not mention them. Therefore, the accuracy of J-SACNN did not increase, compared with the other models.

**Classification on Harassment Forms:** In Table 4, we also compared the performance of binary classifications on harassment forms with the results reported by Karlekar and Bansal (2018). Joint learning models achieved higher accuracy. In some harassment stories, the whole text or a span of the text consists of trigger words of multiple forms, such as *"stare, whistles, start to sing, commenting"*. The supervised attention mechanism will force the model to look at all such words rather than just the one related to the harassment form for classification and hence it can introduce noise. This can explain why J-SACNN got lower accuracy in two of the harassment form classifications, compared to J-ACNN. In addition, J-CNN model did best in "ogling" classification.

## 6 Patterns of Sexual Harassment

We plotted the distribution of harassment incidents in each categorization dimension (Figure 4). It displays statistics that provide important evidence as to the scale of harassment and that can serve as the basis for more effective interventions to be developed by authorities ranging from advocacy organizations to policy makers. It provides evidence to support some commonly assumed factors about harassment: First, we demonstrate that harassment occurred more frequently during the night time than the day time. Second, it shows that besides unspecified strangers (not shown in the figure), conductors and drivers are top the list of identified types of harassers, followed by friends and relatives.

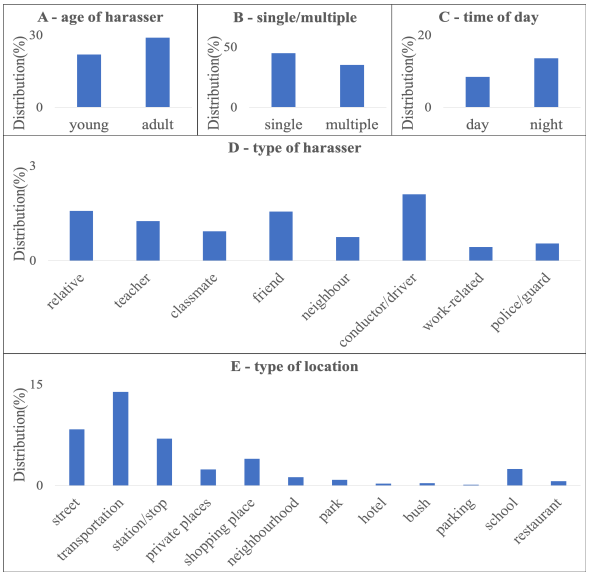


Figure 4: Distributions of incidents. A) Distributions over age of harasser, B) over single/multiple harasser(s), C) over time of day, D) over type of harasser. E) over type of location.

Furthermore, we uncovered that there exist strong correlations between the age of perpetrators and the location of harassment, between the single/multiple harasser(s) and location, and between age and single/multiple harasser(s) (Figure 5). The significance of the correlation is tested by chi-square independence with $p$ value less than 0.05. Identifying these patterns will enable interventions to be differentiated for and targeted at specific populations. For instance, the young harassers often engage in harassment activities as groups. This points to the influence of peer pressure and masculine behavioral norms for men and boys on these activities. We also found that the majority of young perpetrators engaged in harassment behaviors on the streets. These findings suggest that interventions with young men and boys, who are readily influenced by peers, might be most effective when education is done peer-to-peer. It also points to the locations where such efforts

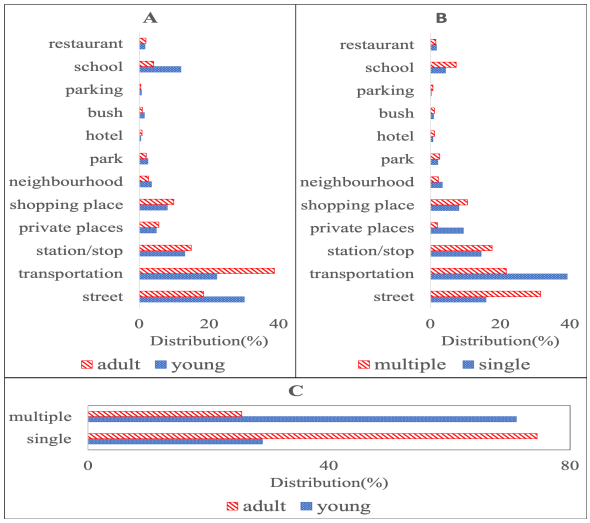


Figure 5: Distributions of incidents over two dimensions. A) Distributions of incidents A) with young/adult harassers at each location, B) with single/multiple harasser(s) at each location, C) across young/adult harassers and single/multiple harasser(s)

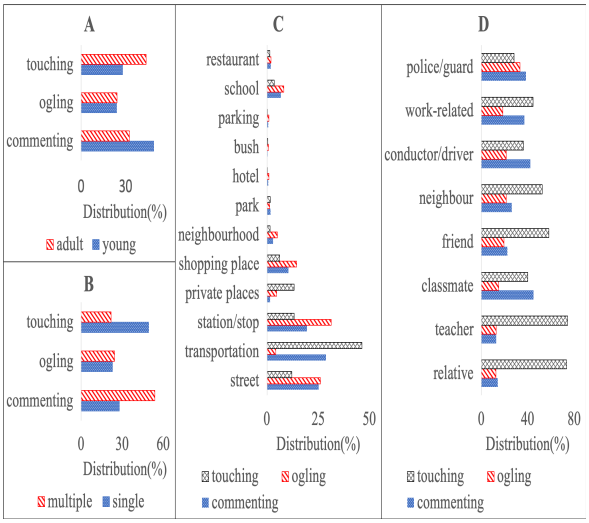


Figure 6: Distributions of incidents with harassment forms and different dimensions. Distributions of harassment forms A) within each age group, B) within single/multiple harasser(s), C) over locations, D) within each harasser type.

could be made, including both in schools and on the streets. In contrast, we found that adult perpetrators of sexual harassment are more likely to act alone. Most of the adult harassers engaged in harassment on public transportation. These differences in adult harassment activities and locations, mean that interventions should be responsive to these factors. For example, increasing the security measures on transit at key times and locations.

In addition, we also found that the correlations between the forms of harassment with the age, single/multiple harasser, type of harasser, and location (Figure 6). For example, young harassers are more likely to engage in behaviors of verbal harassment, rather than physical harassment as compared to adults. It was a single perpetrator that engaged in touching or groping more often, rather than groups of perpetrators. In contrast, commenting happened more frequently when harassers were in groups. Last but not least, public transportation is where people got indecently touched most frequently both by fellow passengers and by conductors and drivers. The nature and location of the harassment are particularly significant in developing strategies for those who are harassed or who witness the harassment to respond and manage the everyday threat of harassment. For example, some strategies will work best on public transport, a particular closed, shared space setting, while other strategies might be more effective on the open space of the street.

These results can provide valuable information for all members of the public. Sharing stories of harassment has been found by researchers to shift people's cognitive and emotional orientation towards their traumatic experiences (Dimond et al., 2013). Greater awareness of patterns and scale of harassment experiences promises to ensure those who have been subjected to this violence that they are not alone, empowering others to report incidents, and ensuring them that efforts are being made to prevent others from experiencing the same harassment. These results also provide various authorities tools to identify potential harassment patterns and to make more effective interventions to prevent further harassment incidents. For instance, the authorities can increase targeted educational efforts at youth and adults, and be guided in utilizing limited resources the most effectively to offer more safety measures, including policing and community-based responses. For example, focusing efforts on highly populated public transportation during the nighttime, when harassment is found to be most likely to occur.

## 7 Conclusions

We provided a large number of annotated personal stories of sexual harassment. Analyzing and identifying the social patterns of harassment behavior is essential to changing these patterns and social tolerance for them. We demonstrated the joint learning NLP models with strong performances to

automatically extract key elements and categorize the stories. Potentiality, the approaches and models proposed in this study can be applied to sexual harassment stories from other sources, which can process and summarize the harassment stories and help those who have experienced harassment and authorities to work faster, such as by automatically filing reports (Karlekar and Bansal, 2018). Furthermore, we discovered meaningful patterns in the situations where harassment commonly occurred. The volume of social media data is huge, and the more we can extract from these data, the more powerful we can be as part of the efforts to build a safer and more inclusive communities. Our work can increase the understanding of sexual harassment in society, ease the processing of such incidents by advocates and officials, and most importantly, raise awareness of this urgent problem.

## References


Sweta Agrawal and Amit Awekar. 2018. Deep learning for detecting cyberbullying across multiple social media platforms. *CoRR*, abs/1801.06482.

Sairam Balani and Munmun De Choudhury. 2015. Detecting and characterizing mental health related self-disclosure in social media. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*. ACM.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Vania Ceccato. 2017. Women's victimisation and safety in transit environments. *Crime Prevention and Community Safety*, 19(3):163–167.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the ACL and the 7th International JCNLP*, pages 167–176.

Jill Dimond, Michaelanne Dye, Daphne Larose, and Amy S. Bruckman. 2013. Hollaback!: the role of storytelling online in a social movement organization. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, pages 477–490.

EEOC. 2018. Charges alleging sexual harassment.

Xiaocheng Feng, Lifu Huang, Duyu Tang, Heng Ji, Bing Qin, and Ting Liu. 2016. A language-independent neural network for event detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 66–71. ACL.

Allison Goldstein. 2018. Sexual harassment statistics: Is the truth in the numbers?

Linda Lewis Griffith. 2018. Guilt, doubt, fear: Why women don't report sexual assault.

Robert T. Hitlan, John B. Pryor, Matthew S. Hesson-McInnis, and Michael Olson. 2009. Antecedents of gender harassment: An analysis of person and situation factors. *Sex Roles*, 61(11):794–807.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Sweta Karlekar and Mohit Bansal. 2018. SafeCity: Understanding diverse forms of sexual harassment personal stories. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2805–2811. ACL.

Suzanne Goodney Lea, Elsa D'Silva, and Abhijith Asok. 2017. Women's strategies addressing sexual harassment and assault on public buses: an analysis of crowdsourced data. *Crime Prevention and Community Safety*, 19(3):227–239.

Yann Lecun and Yoshua Bengio. 1995. *Convolutional Networks for Images, Speech and Time Series*, pages 255–258. MIT Press.

Yingchi Liu, Quanzhi Li, Xiaongzhong Liu, Qiong Zhang, and Luo Si. 2019. Sexual harassment story classification and key information identification. In *The 28th ACM International Conference on Information and Knowledge Management (CIKM'19)*.

L.J.P.V.D. Maaten and GE Hinton. 2008. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605.

Rachel E. Morgan and Jennifer L. Truman. 2018. *Criminal Victimization, 2017*. Department of Justice, Office of Justice Programs, Bureau of Justice Statistics.

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 365–371. ACL.

Quinnipiac. 2017. 60% of u.s. women say they've been sexually harassed quinnipiac university national poll finds.

Nicolas Schrading, Cecilia Ovesdotter Alm, Ray Ptucha, and Christopher Homan. 2015. An analysis of domestic abuse discourse on reddit. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2577–2583. ACL.

M. Schuster and K.K. Paliwal. 1997. Bidirectional recurrent neural networks. *Trans. Sig. Proc.*, 45(11):2673–2681.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.

Yue Zhao, Xiaolong Jin, Yuanzhuo Wang, and Xueqi Cheng. 2018. Document embedding enhanced event detection with hierarchical and supervised attention. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 414–419. ACL.

Marc Ziegele, Johannes Daxenberger, Oliver Quiring, and Iryna Gurevych. 2018. Developing automated measures to predict incivility in public online discussions on the facebook sites of established news media. In *Proceedings of the 68th Annual Conference of the International Communication Association (ICA)*.

# Identifying Predictive Causal Factors from News Streams

**Ananth Balashankar[1], Sunandan Chakraborty[2], Samuel Fraiberger[1,3], and Lakshminarayanan Subramanian[1]**

[1]Courant Institute of Mathematical Sciences, New York University
[2]School of Informatics and Computing, Indiana University-Indianapolis
[3]World Bank, Washington DC
ananth@nyu.edu, sunchak@iu.edu, sfraiberger@worldbank.org, lakshmi@nyu.edu

## Abstract

We propose a new framework to uncover the relationship between news events and real world phenomena. We present the *Predictive Causal Graph* (PCG) which allows to detect latent relationships between events mentioned in news streams. This graph is constructed by measuring how the occurrence of a word in the news influences the occurrence of another (set of) word(s) in the future. We show that PCG can be used to extract latent features from news streams, outperforming other graph-based methods in prediction error of 10 stock price time series for 12 months. We then extended PCG to be applicable for longer time windows by allowing time-varying factors, leading to stock price prediction error rates between 1.5% and 5% for about 4 years. We then manually validated PCG, finding that 67% of the causation semantic frame arguments present in the news corpus were directly connected in the PCG, the remaining being connected through a semantically relevant intermediate node.

## 1 Introduction

Contextual embedding models (Devlin et al., 2018) have managed to produce effective representations of words, achieving state-of-the-art performance on a range of NLP tasks. In this paper, we consider a specific task of predicting variations in stock prices based on word relationships extracted from news streams. Existing word embedding techniques are not suited to learn relationships between words appearing in different documents and contexts (Le and Mikolov, 2014). Existing work on stock price prediction using news have typically relied on extracting features from financial news (Falinouss, 2007; Hagenau et al., 2013), or sentiments expressed on Twitter (Mao et al., 2011; Rao and Srivastava, 2012; Bernardo et al., 2018), or by focusing on features present in a single document (Kalyani et al., 2016; Shynkevich et al., 2015). However, relationships between events affecting stock prices can be quite complex, and their mentions can be spread across multiple documents. For instance, market volatility is known to be triggered by recessions; this relationship may be reflected with a spike in the frequency of the word "recession" followed by a spike in the frequency of the word "volatility" a *few weeks later*. Existing methods are not well-equipped to deal with these cases.

This paper aims to uncover latent relationships between words describing events in news streams, allowing us to unveil *hidden* links between events spread across time, and integrate them into a news-based predictive model for stock prices. We propose the *Predictive Causal Graphs (PCG)*, a framework allowing us to detect latent relationships between words when such relationships are not directly observed. PCG differs from existing relationship extraction (Das et al., 2010) and representational frameworks (Mikolov et al., 2013) across two dimensions. First, PCG identifies unsupervised causal relationships based on consistent time series prediction instead of association, allowing us to uncover paths of *influence* between news items. Second, PCG finds inter-topic influence relationships outside the "context" or the confines of a single document. Construction of PCG naturally leads to *news-dependent* predictive models for numerous variables, like stock prices.

We construct PCG by identifying Granger causal pairs of words (Granger et al., 2000) and combining them to form a network of words using the Lasso Granger method (Arnold et al., 2007). A directed edge in the network therefore represents a potential influence between words. While predictive causality is not true causality (Maziarz, 2015), identification of predictive causal factors which prove to be relevant predictors over long periods of



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2338–2348, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

time provides guidance for future causal inference studies. We achieve this consistency by proposing a framework for *Longitudinal Predictive Causal Factor* identification based on methods of honest estimation (Athey and Imbens, 2016). Here, we first estimate a universe of predictive causal factors on a relatively long time series and then identify time-varying predictive causal factors based on constrained estimation on multiple smaller time series. We also augment our model with an orthogonal spike correction ARIMA (Brockwell and Davis, 2002) model, allowing us to overcome the drawback of slow recovery in smaller time series.

We constructed PCG from news streams of around $700,000$ articles from Google News API and New York Times spread across over 6 years and evaluated it to extract features for stock price predictions. We obtained *two orders lower* prediction error compared to a similar semantic causal graph-based method (Kang et al., 2017). The longitudinal PCG provided insights into the variation in importance of the predictive causal factors over time, while consistently maintaining a low prediction error rate between 1.5-5% in predicting 10 stock prices. Using full text of more than 1.5 million articles of Times of India news archives for over 10 years, we performed a fine-grained qualitative analysis of PCG and validated that 67% of the semantic causation arguments found in the news text is connected by a direct edge in PCG while the rest were linked by a path of length 2. In summary, PCG provides a powerful framework for identifying predictive causal factors from news streams to accurately predict and interpret price fluctuations.

## 2 Related Work

Online news articles are a popular source for mining real-world events, including extraction of causal relationships. Radinsky and Horvitz (Radinsky and Horvitz, 2013) proposed a framework to find causal relationships between events to predict future events from News but caters to a small number of events. Causal relationships extracted from news using Granger causality have also been used for predicting variables, such as stock prices (Kang et al., 2017; Verma et al., 2017; Darrat et al., 2007). A similar causal relationship generation model has been proposed by Hashimoto et al. (2015) to extract causal relationships from natural language text. A similar approach can be observed in (Kozareva, 2012; Do et al., 2011), whereas CATENA system (Mirza and Tonelli, 2016) used a hybrid approach consisting of a rule-based component and a supervised classifier. PCG differs from these approaches as it explores latent inter-topic causal relationships in an unsupervised manner from the entire vocabulary of words and collocated N-grams.

Apart from using causality, there are many other methods explored to extract information from news and are used in time series based forecasting. Amodeo et al. (Amodeo et al., 2011) proposed a hybrid model consisting of time-series analysis, to predict future events using the New York Times corpus. FBLG (Cheng et al., 2014) focused on discovering temporal dependency from time series data and applied it to a Twitter dataset mentioning the Haiti earthquake. Similar work by Luo et al. (Luo et al., 2014) showed correlations between real-world events and time-series data for incident diagnosis in online services. Other similar works like, Trend Analysis Model (TAM) (Kawamae, 2011) and Temporal-LDA (TM-LDA) (Wang et al., 2012) model the temporal aspect of topics in social media streams like Twitter. Structured data extraction from news have also been used for stock price prediction using techniques of information retrieval in (Ding et al., 2014; Xie et al., 2013; Ding et al., 2015; Chang et al., 2016; Ding et al., 2016). Vaca et al. (Vaca et al., 2014) used a collective matrix factorization method to track emerging, fading and evolving topics in news streams. PCG is inspired by such time series models and leverages the Granger causality detection framework for the trend prediction task.

Deriving true causality from observational studies has been studied extensively. One of the most widely used algorithm is to control for variables which satisfy the backdoor criterion (Pearl, 2009). This however, requires a knowledge of the causal graph and the unconfoundedness assumption that there are no other unobserved confounding variables. While the unconfoundedness assumption is to some extent valid when we analyze all news streams (under the assumption that all significant events are reported), it is still hard to get away from the causal graph requirement. Propensity score based matching aims to control for most confounding variables by using an external method for estimating and controlling for the likelihood of outcomes (Olteanu et al., 2017).

More recently, (Wang and Blei, 2018) showed that with multiple causal factors, it is possible to leverage the correlation of those multiple causal factors and deconfound using a latent variable model. This setting is similar to the one we consider, and is guaranteed to be truly causal if there is no confounder which links a single cause and the outcome. This assumption is less strict than the unconfoundedness assumption and makes the case for using predictive causality in such scenarios. Another approach taken by (Athey and Imbens, 2016) estimates heterogeneous treatment effects by honest estimation where the model selection and factor weight estimation is done on two subpopulations of data by extending regression trees.

Our work is motivated by these works and applies methodologies for time series data extracted from news streams. PCG can offer the following benefits for using news for predictive analytics – (1) Detection of influence path, (2) Unsupervised feature extraction, (3) Hypothesis testing for experiment design.

## 3 Predictive Causal Graph

Predictive Causal Graph (PCG) addresses the discovery of *influence* between words that appear in news text. The identification of influence link between words is based on temporal co-variance, that can help answer questions of the form: "Does the appearance of word $x$ influence the appearance of word $y$ after $\delta$ days?". The influence of one word on another is determined based on pairwise causal relationships and is computed using the Granger causality test. Following the identification of Granger causal pairs of words, such pairs are combined together to form a network of words, where the directed edges depict potential influence between words. In the final network, an edge or a path between a word pair represents a flow of influence from the source word to the final word and this *influence* depicts an increase in the appearance of the final words when the source word was observed in news data.

Construction of PCG from the raw unstructured news data, finding pairwise causal links and eventually building the influence network involves numerous challenges. In the rest of the section, we discuss the design methodologies used to overcome these challenges and describe some properties of the PCG.

### 3.1 Selecting *Informative* Words:

Only a small percentage of the words appearing in news can be used for meaningful information extraction and analysis (Manning et al., 1999; Hovold, 2005). Specifically, we eliminated too frequent (at least once in more than 50% of the days) or too rare (appearing in less than 100 articles) (Manning et al., 2008). Many common English nouns, adjectives and verbs, whose contribution to semantics is minimal (Forman, 2003) were also removed from the vocabulary. However, named-entities were retained for their newsworthiness and a set of "trigger" words were retained that depict events (e.g. flood, election) using an existing "event trigger" detection algorithm (Ahn, 2006). The vocabulary set was enhanced by adding bigrams that are significantly collocated in the corpus, such as, 'fuel price' and 'prime minister' etc.

### 3.2 Time-series Representation of Words:

Consider a corpus $D$ of news articles indexed by time $t$, such that $D_t$ is the collection of news articles published at time $t$. Each article $d \in D$ is a collection of words $W_d$, where $i^{th}$ word $w_{d,i} \in W_d$ is drawn from a vocabulary $V$ of size $N$. The set of articles published at time $t$ can be expressed in terms of the words appearing in the articles as $\{\alpha_1^t, \alpha_2^t, ..., \alpha_N^t\}$, where $\alpha_i^t$ is the sum of frequency of the word $w_i \in V$ across all articles published at time $t$. $\alpha_i^t$ corresponding to $w_i \in V$ is defined as, $\alpha_i^t = \frac{\mu_i^t}{\sum_{t=1}^{T} \mu_i^t}$ where $\mu_i^t = \sum_{d=1}^{|D_t|} TF(w_{d,i})$. $\alpha_i^t$ is normalized by using the frequency distribution of $w_i$ in the entire time period. $\mathcal{T}(w_i)$ represents the time series of the word $w_i$, where $i$ varies from 1 to $N$, the vocabulary size.

### 3.3 Measuring Influence between Words

Given two time-series $X$ and $Y$, the Granger causality test checks whether the $X$ is more effective in predicting $Y$, than using just $Y$ and if this holds then the test concludes $X$ "Granger-causes" $Y$ (Granger et al., 2000). However, if both $X$ and $Y$ are driven by a common third process with different lags, one might still fail to reject the alternative hypothesis of Granger causality. Hence, in PCG, we explore the possibility of causal links between all word pairs and detect triangulated relations to eliminate the risk of ignoring confounding variables, otherwise not considered in the Granger causality test.

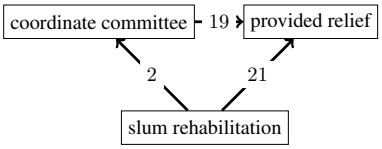


Figure 1: PCG highlighting the underlying cause

However, constructing PCG using an exhaustive set of word pairs does not scale, as even after using a reduced set of words and including the collocated phrases, the vocabulary size is around $39,000$. One solution to this problem is considering the Lasso Granger method (Arnold et al., 2007) that applies regression to the neighborhood selection problem for any word, given the fact that the best regressor for that variable with the least squared error will have non-zero coefficients only for the lagged variables in the neighborhood. The Lasso algorithm for linear regression is an incremental algorithm that embodies a method of variable selection (Tibshirani, 1994).

If we define $V$ to be the input vocabulary from the news dataset, $N$ is the vocabulary size, $x$ is the list of all lagged variables (each word is multivariate with a maximum lag of 30 days per word) of the vocabulary, $w$ is the weight vector denoting the influence of each variable, $y$ is the predicted time series variable and $\lambda$ is a sparsity constraint hyperparameter to be fine-tuned, then minimizing the regression loss below leads to weights that characterize the influential links between words in $x$ that predicts $y$,

$$\mathbf{w} = argmin \frac{1}{N} \Sigma_{(\mathbf{x},y)\in V} |\mathbf{w}.\mathbf{x} - y|^2 + \lambda ||\mathbf{w}|| \quad (1)$$

To set $\lambda$, we use the method based on consistent estimation used in (Meinshausen and Bühlmann, 2006). We select the variables that have non-zero co-efficients and choose the best lag for a given variable based on the maximum absolute value of a word's co-efficient. We then, draw an edge from all these words to the predicted word with the annotations of the optimal time lag (in days) and incrementally construct the graph as illustrated in Figure 1.

### 3.4 Topic Influence Compression

To arrive at a sparse graphical representation of PCG, we compress the graph based on topics (50 topics in our case). Topics are learned from the original news corpus using unsupervised Latent Dirichlet Allocation (LDA)(Blei et al., 2003). Influence is generalized to topic level by calculating the weight of inter-topic influence relationships as a total number of edges between vertices of two topics. If we define $\theta_u$ and $\theta_v$ to be two topics in our topic model and $|\theta_u|$ represents the size of topic $\theta_u$, i.e. the number of words in the topic whose topic-probability is greater than a threshold $(0.001)$, then the strength of influence between topics $\theta_u$ and $\theta_v$ is defined as,

$$\Phi(\theta_u, \theta_v) = \frac{\text{\# Edges between words in } \theta_u \text{ and } \theta_v}{(|\theta_u| \times |\theta_v|)} \quad (2)$$

$\Phi(\theta_u, \theta_v)$ is termed as *strong* if its value is in the $99^{th}$ percentile of $\Phi$ for all topics. Any edge in the original PCG is removed if there are no strong topic edges between the corresponding word nodes. This filtered topic graph has only edges between topics which have high influence strength. This combination of inter-document temporal and intra-document distributional similarity is critical to obtaining temporally and semantically consistent predictive causal factors.

## 4 Prediction Models using PCG

In this section, we present three approaches for building prediction models using PCG namely (1) direct estimation using PCG (2) longitudinal prediction which incorporates short term temporal variations and (3) spike augumented prediction which estimates spikes over a longer time window.

### 4.1 Direct Prediction from PCG

One straightforward way of using PCG for prediction modeling is to use the Lasso regression equation used for identifying the predictive causal factors directly. We first adopt this approach by restricting the construction of PCG to the nodes of concern, which significantly speeds up the computation. This inherently ignores any predictive causal factor which only has an indirect link to the outcome node, as theorized by the Granger Causality framework. In this case, we split the data into a contiguous training data, and evaluate on the remaining testing data. If $y$ represents the target stock time series variable and $x$ represents a multivariate vector of all lagged feature variables, $w$ represents the coefficient weight vector indexed by the feature variable $z \in x$ and time lag $m, p, q$ in days, $a$ represent a bias constant and $\epsilon_t$ repre-

sents an i.i.d noise variable, then we predict future values of $y$ as follows.

$$y_t = a + \sum_{i=1}^{m} w_{y,i} y_{t-i} + \sum_{z \in x} \sum_{j=p}^{q} w_{z,j} z_{t-j} + \epsilon_t \quad (3)$$

### 4.2 Longitudinal Prediction via Honest Estimation

In scenarios where heterogenous causal effects are to be estimated, it is important to adjust by partitioning the time series into subpopulations which vary in the magnitude of the causal effects (Athey and Imbens, 2016). In a news stream, this amounts to constructing the word influence networks given a context specified by a time window. This naive extension however can be quite computationally expensive and can limit the speed of inference considerably. However, if the set of potential causal factors are identified over a larger time series, learning their time varying weights over a shorter training period can significantly decrease the computation required.

Hence, we do a two staged honest estimation approach similar to (Athey and Imbens, 2016). In the first stage, multiple sets (instead of trees as in (Athey and Imbens, 2016)) of predictive causal factors $F(Tr_m)$, that provide overall reduction in root mean squared error ($RMSE$), over training data $Tr_m$, are gathered for model selection through repeated random initialization of regression weights. For any $f \in F(Tr_m)$, we define $f$ to be a set of predictive factors which when trained over data $Tr_m$ to predict future time series values of the target $y$, achieves $RMSE(Tr_m, f) < \delta$, for a hyperparameter $\delta > 0$.

$$F(Tr_m) = \bigcup\nolimits_f \{RMSE(Tr_m, f) < \delta\} \quad (4)$$

From these sets of predictive causal factors $F(Tr_m)$, we choose the set of features $f_{LPC}(Tr_m, Tr_e)$, which gives the least expected root mean squared error on time windows $w$ of length $W$ uniformly sampled from the unseen training data used for estimation $Tr_e$ through cross validation. This model selection procedure depends on the size of the smaller time windows $W$ used to fine-tune the factors. The size of this time window is a hyperparameter to trade off long-term stability and short-term responsiveness of the predictive causal factors. We chose the time window based of 30 days for our stock price prediction due to prior knowledge that many financial indicators have a monthly cycle and hence responsiveness within that cycle is desired.

$$E(Tr_e, f) = \sqrt{\mathop{\mathbb{E}}_{w \in Tr_e, |w|=W} MSE(w, f)} \quad (5)$$

$$f_{LPC}(Tr_m, Tr_e) = min_{f \in F(Tr_m)} \Big( E(Tr_e, f) \Big) \quad (6)$$

We then evaluate the model on an unseen time series $Te$, where the learnt predictive causal factors and their weights are used for inference.

### 4.3 Spike Prediction

One drawback of using a specific time window for estimating the weights of the predictive causal factors is the lack of representative data in the window used for training. This could mean that predicting abrupt drops or spikes in the data would be hard. To overcome this limitation, we train a composite model to predict the residual error from honest estimation by training on differences in consecutive values of the time series. Let ($\Delta y = y_t - y_{t-1}, \Delta f = f_t - f_{t-1}$) denote time series of the differences of the consecutive values of the labels and the feature variables and let $[\Delta y, \Delta f]$ denote the concatenated input variables of the model. We use a multivariate ARIMA model $M$ of order $(p, d, q)$ (Brockwell and Davis, 2002) where $p$ controls the number of time lags included in the model, $q$ denotes the number of times differences are taken between consecutive values and $r$ denotes the time window of the moving average taken to incorporate sudden spikes in values. Let the actual values of the time series of label $y$ be $y^*$, the predictions of the honest estimation model be $\hat{y}$, a training sample of significantly longer time window $Tr_s$ with $|Tr_s| >> W$, then the composite model is trained to predict the residuals, $res = y^* - \hat{y} = E(Tr_s, f)$.

$$M = ARIMA(p, d, q) \quad (7)$$

$$M.fit(E(Tr_s, f), [\Delta y, \Delta f]) \quad (8)$$

$$\hat{res} = M.predict(Tr_e) \quad (9)$$

Augmenting this predicted residual ($\hat{res}$) back to $\hat{y}_{Tr_e}$ gives us the spike-corrected estimate $\hat{y_s}$.

$$\hat{y_s} = \hat{y}_{Tr_e} + \hat{res} \quad (10)$$

## 5 Results

In this section, we present the results from direction prediction models from PCG, followed by improvement in stock price prediction due to longitudinal and spike prediction from news streams and

compare it to a manually tuned semantic causal graph method. We analyze the time varying factors to explain the gains achieved via honest estimation.

### 5.1 Data and Metrics

The news dataset[1] we used for stock price prediction contains news crawled from 2010 to 2013 using Google News APIs and New York Times data from 1989 to 2007. We construct PCG from the time series representation of its 12,804 unigrams and 25,909 bigrams over the entire news corpora of more than 23 years, as well as the 10 stock prices[2] from 2010 to 2012 for training and 2013 as test data for prediction. The prediction is done with varying step sizes (1,3,5), which indicates the time lag between the news data and the day of the predicted stock price in days. The results shown in Table 1 is the root mean squared error (RMSE) in predicted stock value calculated on a 30 day window averaged by moving it by 10 days over the period and directly comparable to our baseline (Kang et al., 2017). To evaluate the time-varying factors over a larger time window, we present average monthly cross validation RMSE % sampled over a 4 year window of 2010-13 in Table 3. Please note that the results in Table 3 are not comparable with (Kang et al., 2017) as we report a cross validation error over a longer time window.

### 5.2 Prediction Performance of PCG

To evaluate the causal links generated by PCG, we use it to extract features for predicting stock prices using the exact data and prediction setting used in Kang et al. (2017) as our baseline.

**Baseline:** Kang et al. (2017) extract relevant variables based on a semantic parser - SEMAFOR (Das et al., 2010) by filtering causation related frames from news corpora, topics and sentiments from tweets. To overcome the problem of low recall, they adopt a topic-based knowledge base expansion. This expanded dataset is then used to train a neural reasoning model which generates sequence of cause-effect statements using an attention model where the words are represented using word2vec vectors. (Kang et al., 2017)'s CGRAPH based forecasting model - $C_{best}$ model uses the top 10 such generated cause features, given the stock name as the effect and apply a vector auto-regressive model on the combined time series of text and historical stock values.

Table 1: 30 day windowed average of stock price prediction error using PCG

| **Step size** | $C_{best}$ | $PCG_{uni}$ | $PCG_{bi}$ | $PCG_{both}$ |
|---|---|---|---|---|
| 1 | 1.96 | 0.022 | 0.023 | 0.020 |
| 3 | 3.78 | 0.022 | 0.023 | 0.022 |
| 5 | 5.25 | 0.022 | 0.023 | 0.021 |

**Comparison with the baseline:** Compared to the baseline, note that our features and topics were chosen purely based on distributional semantics of the word time series. Once the features are extracted from PCG, we use the past values of stock prices and time series corresponding to the incoming word edges of PCG to predict the future values of the stock prices using the multivariate regression equation used to determine Granger Causality. As compared to their best error, PCG from unigrams, bigrams or both obtain two orders lower error and significantly outperforms $C_{best}$. The mean absolute error (MAE) for the same set of evaluations is within 0.003 of the RMSE, which indicates that the variance of the errors is also low. We attribute this gain to the flexibility of PCG's Lasso Granger method to produce sparse graphs as compared to CGRAPH's Vector Auto Regressive model which used a fixed number (10) of incoming edges per node already pre-filtered by a semantic parser. This imposes an artificial bound on sparsity thereby losing valuable latent information. We overcome this in PCG using a suitable penalty term ($\lambda$) in the Lasso method.

**Key PCG factors for 2013:** The causal links in PCG are more generic (Table 2) than the ones described in CGRAPH, supporting the hypothesis that latent word relationships do exist that go beyond the scope of a single news article. The nodes of CGRAPH are tuples extracted from a semantic parser (SEMAFOR (Das et al., 2010)) based on evidence of causality in a sentence. PCG poses no such restriction and derives topical (unfriended, FB) and inter-topical (healthcare, AMZN), sparse, latent and semantic relationships.

Inspecting the links and paths of PCG gives us qualitative insights into the context in which the word-word relationships were established. Since PCG is also capable of representing other stock time series as potential influencers in the network, we can use this to model the propagation of shocks in the market as shown in Figure 2. However,

[1] https://github.com/dykang/cgraph
[2] https://finance.yahoo.com

Table 2: Stock price predictive factors for 2013 in PCG

| Stock symbol | Prediction indicators |
|---|---|
| AAPL | workplace, shutter, music |
| AMZN | healthcare, HBO, cloud |
| FB | unfriended, troll, politician |
| GOOG | advertisers, artificial intelligence, shake-up |
| HPQ | China, inventions, Pittsburg |
| IBM | 64 GB, redesign, outage |
| MSFT | interactive, security, Broadcom |
| ORCL | corporate, investing, multimedia |
| TSLA | prices, testers, controversy |
| YHOO | entertainment, leadership, investment |

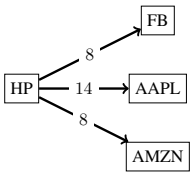


Figure 2: Inter-stock influencer links where one stock's movement indicates future movement of other stocks (time lag annotated edges)

these links were not used for prediction performance to maintain parity with our baseline.

### 5.3 Time Varying Causal Analysis

We quantitatively evaluate the time varying variant of PCG by using it to extract features for stock price prediction for a longer time window.

We present average root mean squared errors in Table 3 for different values of time windows of size $W$ (50,100). For model selection, we used 50% of the time series and then used multiple time series of length $W$, disjoint from the ones used for time-varying factor selection and took average of the test error on the next 30 data points using the weights learnt. We repeat this using K-fold cross validation (K=10) for choosing the model selection data and present the average errors. The variation in importance weights of predictive causal factors for stock prices ("podcast", AAPL) and ("unfollowers", GOOG) is shown in Figure 3 which illustrates several peaks (for weeks) when the factor in the news was particularly predictive of the company's stock price and not significant during other weeks.

The time series and error graph shown for multiple stocks shows that the RMSE errors range between 1.5% 5% for all the test time series as shown in Figure 4. However, sudden spikes tend to display higher error rates due to the lack of training data which contain spikes. This issue is mitigated when the time window for training is increased. Increasing the window more than 100 did not improve the RMSE and came at the cost of training time. But, incorporating the spike residual PCG model, which predicts the leftover price value from the first model, provides significant improvements over the model without spike correction as seen in the last column on Table 3. Thus, we are able to achieve significant gains with an unsupervised approach without any domain specific feature engineering by estimating using an ARIMA model (p, d, q) = (30, 1, 10).

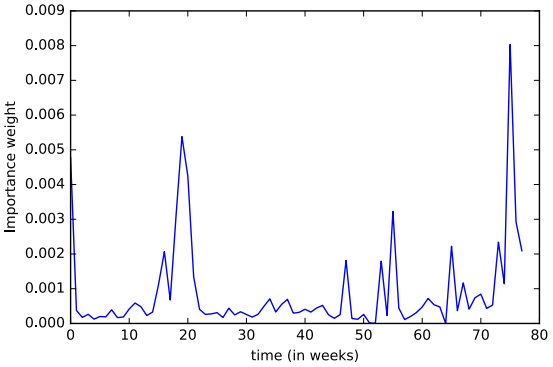


(a) "podcast" for predicting AAPL stock

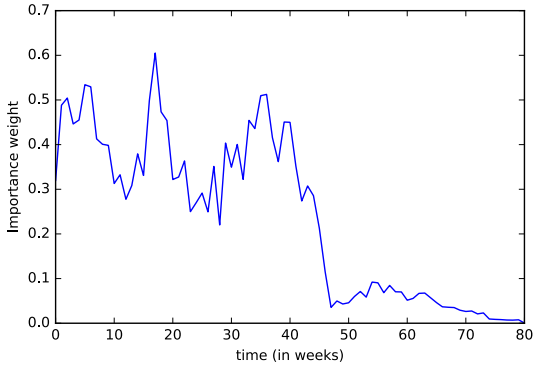


(b) "unfollowers" for predicting GOOG stock

Figure 3: Temporal variation in importance weight of predictive causal factors

## 6 Interpretation of Predictive Causal Factors

In order to qualitatively validate that the latent inter-topic edges learnt from the news stream is also humanly interpretable, we constructed PCG from the online archives of Times of India (TOI) [3], the most circulated Indian English newspaper. We used this dataset as, unlike the previous dataset

[3] https://timesofindia.indiatimes.com/archive.cms

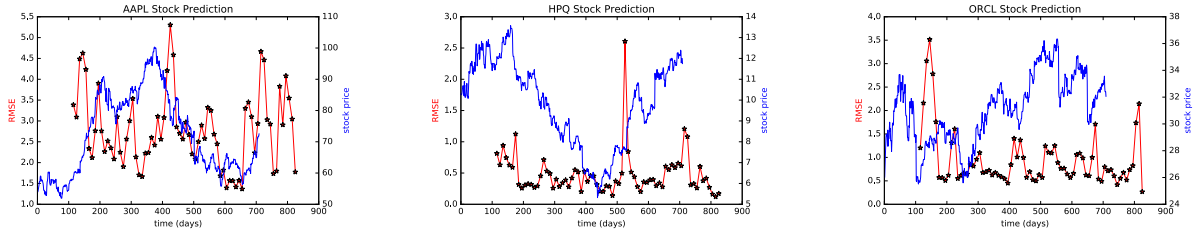


Figure 4: RMSE of stocks for longitudinal causal factor prediction without spike correction. Spikes in RMSE can be seen along with spikes in stock prices like HPQ.

Table 3: Variation in stock price prediction error (RMSE) % with window size and spike correction

| Stock | W=50 | W=100 | W=100 + spike |
|---|---|---|---|
| AABA | 2.87 | 2.07 | 1.61 |
| AAPL | 2.95 | 2.84 | 2.28 |
| AMZN | 3.03 | 2.99 | 2.41 |
| GOOG | 2.67 | 2.36 | 1.91 |
| HPQ | 6.77 | 3.34 | 2.44 |
| IBM | 2.19 | 2.07 | 1.65 |
| MSFT | 3.03 | 9.45 | 4.80 |
| ORCL | 2.94 | 2.21 | 1.65 |
| TSLA | 5.56 | 5.52 | 4.32 |

which provided just the time series of words, we also have the raw text of the articles, which allowed us to perform manual causal signature verification. This dataset contains all the articles published in their online edition between January 1, 2006 and December 31, 2015 containing 1,538,932 articles.

### 6.1 Inter-topic edges of PCG

The influence network we constructed from the TOI dataset has 18,541 edges and 7,190 uni-grams and bi-gram vertices. We were interested in the inter-topic non-associative relationships that PCG is expected to capture. We observe that a few topics (5) influence or are influenced by a large number of topics. Some of these highly influential topics are composed of words describing "Agriculture", "Politics", "Crime", etc. The ability of PCG to learn these edges between topical word clusters purely based on temporal predictive causal prediction further validates its use for design of extensive causal inference experiments.

### 6.2 Causal evidence in PCG

To validate the causal links in PCG, we extracted 56 causation semantic frame (Baker et al., 1998) arguments which depict direct causal relationships in the news corpus. We narrowed down the search to words surrounding verbs which depict the notion of causality like "caused", "effect", "due to" and manually verified that these word pairs were indeed causal. We then searched the shortest path in PCG between these word pairs. For example, one of the news article mentioned that "Gujarat government has set aside a suggestion for *price hike* in electricity due to the Mundra Ultra Mega *Power Project*." and these corresponding causation arguments were captured by a direct link in PCG as shown in Table 4. 67% of the word pairs which were manually identified to be causal in the news text through causal indicator words such as "caused", were linked in PCG through direct edges, while the rest were linked through an intermediate relevant node. As seen in Table 4, the bi-gram involving the words and the intermediate words in the path provide the relevant context under which the causality is established. The time lags in the path show that the influence between events are at different time lags. We also qualitatively verified that two unrelated words are either not connected or have a path length greater than 2, which makes the relationship weak. The ability of PCG to validate such humanly understood causal pairs with temporal predictive causality can be used for better hypothesis testing.

Table 4: Comparison with manually identified influence from news articles

| Pairs in news | Relevant paths in PCG |
|---|---|
| price, project | price-hike –(19)– power-project |
| land, budget | allot-land –(22)– railway-budget |
| price, land | price-hike –(12)– land |
| strike, law | terror-strike –(25)– law ministry |
| land, bill | land-reform –(25)– bill-pass |
| election, strike | election –(21)– Kerala government –(10)– strike |
| election, strike | election –(18)– Mumbai University –(14)– strike |
| election, strike | election –(20)– Shiromani Akali –(13)– strike |

## 7 Conclusion

We presented PCG, a framework for building predictive causal graphs which capture hidden rela-

tionships between words in text streams. PCG overcomes the limitations of contextual representation approaches and provides the framework to capture inter-document word and topical relationships spread across time to solve complex mining tasks from text streams. We demonstrated the power of these graphs in providing insights to answer causal hypotheses and extracting features to provide consistent, interpretable and accurate stock price prediction from news streams through honest estimation on unseen time series data.

## References


David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*. Association for Computational Linguistics, Sydney, Australia, 1–8. https://www.aclweb.org/anthology/W06-0901

Giuseppe Amodeo, Roi Blanco, and Ulf Brefeld. 2011. Hybrid Models for Future Event Prediction *(CIKM '11)*. 1981–1984.

Andrew Arnold, Yan Liu, and Naoki Abe. 2007. Temporal Causal Modeling with Graphical Granger Methods. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '07)*. ACM, New York, NY, USA, 66–75. https://doi.org/10.1145/1281192.1281203

Susan Athey and Guido Imbens. 2016. Recursive partitioning for heterogeneous causal effects: Table 1. *Proceedings of the National Academy of Sciences* 113, 7353–7360. https://doi.org/10.1073/pnas.1510489113

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1 (ACL '98/COLING '98)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 86–90. https://doi.org/10.3115/980845.980860

Ivo Bernardo, Roberto Henriques, and Victor Lobo. 2018. Social Market: Stock Market and Twitter Correlation. In *Intelligent Decision Technologies 2017*, Ireneusz Czarnowski, Robert J. Howlett, and Lakhmi C. Jain (Eds.). Springer International Publishing, Cham, 341–356.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *J. Mach. Learn. Res.* 3 (March 2003), 993–1022.

Peter J. Brockwell and Richard A. Davis. 2002. *Introduction to Time Series and Forecasting* (2nd ed.). Springer.

Ching-Yun Chang, Yue Zhang, Zhiyang Teng, Zahn Bozanic, and Bin Ke. 2016. Measuring the Information Content of Financial News. In *COLING*. ACL, 3216–3225.

Dehua Cheng, Mohammad Taha Bahadori, and Yan Liu. 2014. FBLG: A Simple and Effective Approach for Temporal Dependence Discovery from Time Series Data *(KDD '14)*. 382–391. https://doi.org/10.1145/2623330.2623709

Ali F. Darrat, Maosen Zhong, and Louis T.W. Cheng. 2007. Intraday volume and volatility relations with and without public news. *Journal of Banking and Finance* 31, 9 (2007), 2711 – 2729. https://doi.org/10.1016/j.jbankfin.2006.11.019

Dipanjan Das, Nathan Schneider, Desai Chen, and Noah A. Smith. 2010. Probabilistic Frame-Semantic Parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Los Angeles, California, 948–956. https://www.aclweb.org/anthology/N10-1138

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* abs/1810.04805 (2018). arXiv:1810.04805 http://arxiv.org/abs/1810.04805

Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. 2014. Using Structured Events to Predict Stock Price Movement: An Empirical Investigation.

Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. 2015. Deep Learning for Event-Driven Stock Prediction. In *IJCAI*. AAAI Press, 2327–2333.

Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. 2016. Knowledge-Driven Event Embedding for Stock Prediction. In *COLING*. ACL, 2133–2142.

Quang Xuan Do, Yee Seng Chan, and Dan Roth. 2011. Minimally Supervised Event Causality Identification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '11)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 294–303. http://dl.acm.org/citation.cfm?id=2145432.2145466

Pegah Falinouss. 2007. Stock Trend Prediction using News Events. *Masters thesis* (2007).

George Forman. 2003. An extensive empirical study of feature selection metrics for text classification. *Journal of machine learning research* 3, Mar (2003), 1289–1305.

Clive WJ Granger, Bwo-Nung Huangb, and Chin-Wei Yang. 2000. A bivariate causality between stock prices and exchange rates: evidence from recent Asianflu. *The Quarterly Review of Economics and Finance* 40, 3 (2000), 337–354.

Michael Hagenau, Michael Liebmann, and Dirk Neumann. 2013. Automated news reading: Stock price prediction based on financial news using context-capturing features. *Decision Support Systems* 55, 3 (2013), 685 – 697. https://doi.org/10.1016/j.dss.2013.02.006

Chikara Hashimoto, Kentaro Torisawa, Julien Kloetzer, and Jong-Hoon Oh. 2015. Generating Event Causality Hypotheses Through Semantic Relations. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI'15)*. AAAI Press, 2396–2403. http://dl.acm.org/citation.cfm?id=2886521.2886654

Johan Hovold. 2005. Naive Bayes Spam Filtering Using Word-Position-Based Attributes.. In *CEAS*. 41–48.

Joshi Kalyani, H. N. Bharathi, and Rao Jyothi. 2016. Stock trend prediction using news sentiment analysis. *CoRR* abs/1607.01958 (2016). arXiv:1607.01958 http://arxiv.org/abs/1607.01958

Dongyeop Kang, Varun Gangal, Ang Lu, Zheng Chen, and Eduard Hovy. 2017. Detecting and Explaining Causes From Text For a Time Series Event. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, 2758–2767. https://www.aclweb.org/anthology/D17-1292

Noriaki Kawamae. 2011. Trend analysis model: trend consists of temporal words, topics, and timestamps *(WSDM '11)*. 317–326.

Zornitsa Kozareva. 2012. Cause-effect Relation Learning. In *Workshop Proceedings of TextGraphs-7 on Graph-based Methods for Natural Language Processing (TextGraphs-7 '12)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 39–43. http://dl.acm.org/citation.cfm?id=2392954.2392961

Quoc V. Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. *CoRR* abs/1405.4053 (2014). arXiv:1405.4053 http://arxiv.org/abs/1405.4053

Chen Luo, Jian-Guang Lou, Qingwei Lin, Qiang Fu, Rui Ding, Dongmei Zhang, and Zhe Wang. 2014. Correlating Events with Time Series for Incident Diagnosis *(KDD '14)*. 1583–1592. https://doi.org/10.1145/2623330.2623374

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. 2008. *Scoring, term weighting, and the vector space model*. Cambridge University Press, 100123. https://doi.org/10.1017/CBO9780511809071.007

Christopher D Manning, Hinrich Schütze, et al. 1999. *Foundations of statistical natural language processing*. Vol. 999. MIT Press.

Huina Mao, Scott Counts, and Johan Bollen. 2011. Predicting Financial Markets: Comparing Survey, News, Twitter and Search Engine Data. *Arxiv preprint* (2011).

Mariusz Maziarz. 2015. A review of the Granger-causality fallacy. *The Journal of Philosophical Economics* 8, 2 (2015), 6. https://EconPapers.repec.org/RePEc:bus:jphile:v:8:y:2015:i:2:n:6

Nicolai Meinshausen and Peter Bühlmann. 2006. High-dimensional graphs and variable selection with the lasso. *The annals of statistics* (2006), 1436–1462.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and Their Compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'13)*. Curran Associates Inc., USA, 3111–3119. http://dl.acm.org/citation.cfm?id=2999792.2999959

Paramita Mirza and Sara Tonelli. 2016. CATENA: CAusal and TEmporal relation extraction from NAtural language texts. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, 64–75. http://www.aclweb.org/anthology/C16-1007

Alexandra Olteanu, Onur Varol, and Emre Kiciman. 2017. Distilling the Outcomes of Personal Experiences: A Propensity-scored Analysis of Social Media. In *Proceedings of The 20th ACM Conference on Computer-Supported Cooperative Work and Social Computing* (computer-supported cooperative work and social computing ed.). Association for Computing Machinery, Inc.

Judea Pearl. 2009. Causal inference in statistics: An overview. *Statistics Surveys* (2009).

Kira Radinsky and Eric Horvitz. 2013. Mining the web to predict future events *(WSDM '13)*. ACM, 255–264.

Tushar Rao and Saket Srivastava. 2012. Analyzing Stock Market Movements Using Twitter Sentiment Analysis. In *Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012) (ASONAM '12)*. IEEE Computer Society, Washington, DC, USA, 119–123. https://doi.org/10.1109/ASONAM.2012.30

Y. Shynkevich, T. M. McGinnity, S. Coleman, and A. Belatreche. 2015. Stock price prediction based on stock-specific and sub-industry-specific news articles. In *2015 International Joint Conference on Neural Networks (IJCNN)*. 1–8. https://doi.org/10.1109/IJCNN.2015.7280517

Robert Tibshirani. 1994. Regression Shrinkage and Selection Via the Lasso. *Journal of the Royal Statistical Society, Series B* 58 (1994), 267–288.

Carmen K Vaca, Amin Mantrach, Alejandro Jaimes, and Marco Saerens. 2014. A time-based collective factorization for topic discovery and monitoring in news *(WWW '14)*. 527–538.

Ishan Verma, Lipika Dey, and Hardik Meisheri. 2017. Detecting, Quantifying and Accessing Impact of News Events on Indian Stock Indices. In *Proceedings of the International Conference on Web Intelligence (WI '17)*. ACM, New York, NY, USA, 550–557. https://doi.org/10.1145/3106426.3106482

Yu Wang, Eugene Agichtein, and Michele Benzi. 2012. Tm-lda: efficient online modeling of latent topic transitions in social media *(KDD '12)*. ACM, 123–131.

Yixin Wang and David M. Blei. 2018. The Blessings of Multiple Causes. *CoRR* abs/1805.06826. http://dblp.uni-trier.de/db/journals/corr/corr1805.html#abs-1805-06826

Boyi Xie, Rebecca J. Passonneau, Leon Wu, and Germán Creamer. 2013. Semantic Frames to Predict Stock Price Movement. In *ACL (1)*. The Association for Computer Linguistics, 873–883.

# Training Data Augmentation for Detecting Adverse Drug Reactions in User-Generated Content

Sepideh Mesbah[1], Jie Yang[2], Robert-Jan Sips[3],
Manuel Valle Torre[1], Christoph Lofi[1], Alessandro Bozzon[1], and Geert-Jan Houben[1]

[1]Delft University of Technology, Van Mourik Broekmanweg 62628 XE Delft , the Netherlands
[2]Amazon*, 440 Terry Ave N, Seattle, WA 98109, USA
[3]MyTomorrows, Anthony Fokkerweg 61, 1059 CP Amsterdam, the Netherlands
s.mesbah@tudelft.nl, jiy@amazon.com, r.sips@mytomorrows.com
{m.valletorre, c.lofi, a.bozzon, g.j.p.m.houben}@tudelft.nl

## Abstract

Social media provides a timely yet challenging data source for adverse drug reaction (ADR) detection. Existing dictionary-based, semi-supervised learning approaches are intrinsically limited by the coverage and maintainability of laymen health vocabularies. In this paper, we introduce a data augmentation approach that leverages variational autoencoders to learn high-quality data distributions from a large unlabeled dataset, and subsequently, to automatically generate a large labeled training set from a small set of labeled samples. This allows for efficient social-media ADR detection with low training and re-training costs to adapt to the changes and emergence of informal medical laymen terms. An extensive evaluation performed on Twitter and Reddit data shows that our approach matches the performance of fully-supervised approaches while requiring only 25% of training data.

## 1 Introduction

Adverse Drug Reactions (ADRs) is the fourth leading cause of death in the United States (Chee et al., 2011). ADR detection is, therefore, a critical element of drug safety. Studies have shown that clinical trials are not able to fully characterize drugs' adverse effects (Harpaz et al., 2012; Chee et al., 2011; Ahmad, 2003). Traditional techniques of post-market ADR mainly rely on voluntary and mandatory reporting of ADRs by patients and health providers, but they suffer from delays in reporting, under-reporting, or data incompleteness (Sarker et al., 2015).

Social media is becoming a preferred channel for millions of users and patients to share, discuss, and seek health information (He et al., 2017); such user-generated content can, therefore, provide valuable insights for monitoring Adverse Drug Reactions (ADRs) from an additional point of view (Lee et al., 2017; Sarker and Gonzalez, 2015; Aramaki et al., 2011). ADR detection from social media is, however challenging, as online users report ADRs using a different language style and terminology that largely depend on the user's medical proficiency, as well as the type of online medium (e.g., health forums vs micro-post social networks). In particular, laymen often use diverse dialects (Karisani and Agichtein, 2018) when describing medical concepts, and make abundant use of informal terminology.

Existing approaches for detecting terms in informal medical language mainly rely on semi-manually generated dictionaries (e.g. laymen health dictionaries) (Zeng and Tse, 2006), or supervised machine-learning-based sequence classifiers (Huynh et al., 2016; Chowdhury et al., 2018). Due to the language dynamicity in online and offline communication (Kershaw et al., 2016; Zanzotto and Pennacchiotti, 2012), there is a constant emergence of new informal medical terms. This results in a lack of coverage and maintainability of laymen health vocabularies. While showing superior performance, machine learning approaches often need to be trained for specific Web communities and platforms due to differences in the underlying language models; this results in high costs for manual annotation of training data, which for many domains is only sparsely available (Deriu et al., 2017).

More recently, researchers have started to investigate techniques for expanding the size of manually created training data. Often, sentence similarity implemented with embedding techniques (Mikolov et al., 2013; Le and Mikolov, 2014) is used to discover similar sentences, and then annotations are automatically propagated to those sentences (Mesbah et al., 2018). While these techniques have indeed shown to reduce the cost of

*Work performed before joining Amazon.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2349–2359,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

training, they are typically limited by *availability of the existing data* as the reliability of annotation propagation suffers when sentences are not similar enough. Therefore, in this paper we focus on how to automatically *generate* high quality training data for Adverse Drug Reaction detection with minimal human supervision and costs.

**Original Contribution.** In this paper, we propose to generate artificial sentences closely mimicking existing training data; such artificial sentences are then annotated automatically via label propagation. This contrasts existing approaches expanding manually created training data set by discovering additional existing sentences in a dataset.

We build our method upon variational autoencoders (VAE), a deep probabilistic neural model which learns latent text features and their distributions effectively. In contrast to other approaches using variational autoencoders for text generation, we modify the mechanism for generating new artificial samples such that we obtain samples structurally and semantically similar to a specific subset of the original data. This allows us to generate sentences similar to those in the pre-existing human-labeled ADR training usable for classifier training set by taking advantage of the implicit semantics contained in the larger unlabeled dataset.

We evaluate the proposed method on a standard Twitter dataset and on a large new dataset for the Reddit platform we created with the help of expert annotators. The dataset is available on the companion Website (Companion, 2019). Results show that our approach achieves superior or comparable performance with significantly less training data (*reduced by 75%*) than state-of-the-art training methods.

## 2 Related Work

The terminology adopted in most social communities makes heavy use of slang or indirect descriptions, which is often lacking with respect to grammar and orthography; in addition, it is also constantly evolving and differs between communities. This makes the use of established techniques relying on expert-curated dictionaries (Leaman et al., 2010; Soldaini and Goharian, 2016) of consumer health vocabulary or fully-supervised machine learning-based classifiers (Huynh et al., 2016; Chowdhury et al., 2018) expensive, and in many cases even prohibits their use. While techniques to lessen the costs of training like distant supervision (Mintz et al., 2009) or bootstrapping (Tsai et al., 2013; Blum and Mitchell, 1998) can provide some support, their performance has been shown to be limited.

Some recent work has started to address the issues of size and cost of ADR training data (Lee et al., 2017; Cocos et al., 2017; Nikfarjam et al., 2015). Lee et al. (2017) explores different types of unlabeled data and a small training set to generate phrase embeddings, so to *classify* the tweets that indicate adverse drug event in a semi-supervised way. In contrast, our work focuses on detecting the *actual ADR span* in the text of the user generated posts rather than just classifying the whole post as containing an ADR mention. Nikfarjam et al. (2015) and Cocos et al. (2017) augment traditional supervised methods with additional features such as pre-trained word representation vectors, to improve performance and to be less dependent on large training sets. The resulting BLSTM-RNN (Cocos et al., 2017) technique, which achieves state-of-the-art performance, is also evaluated in our experiments (see Section 5.1). Rather than adding new features or proposing new ADR detector models, our work focuses on the generation of new labeled data samples from small annotated training sample using deep probabilistic models.

Different from the above approaches, embedding based methods (Le and Mikolov, 2014; Mikolov et al., 2013) learn vector representations of words or paragraphs to capture semantic relationships among words. Such methods are, therefore, useful to find sentences similar to the labeled training data, thereby expanding the size of the training data. Embedding based methods, however are limited by the existing sentences available in a given corpus. In contrast, our approach is capable of generating new sentences not existing in the corpus, thus largely expanding the training data. Our approach for generating additional labeled training data is inspired by (Bowman et al., 2016), where VAEs are used to learn a generative model of text for sentence generation. Bowman et al. (2016), however, only tackles the general problem of sentence generation. To the best of our knowledge, our work is the first that investigate VAEs as a tool for training data expansion, so as to enhance machine learning performance with limited amount of labeled data.

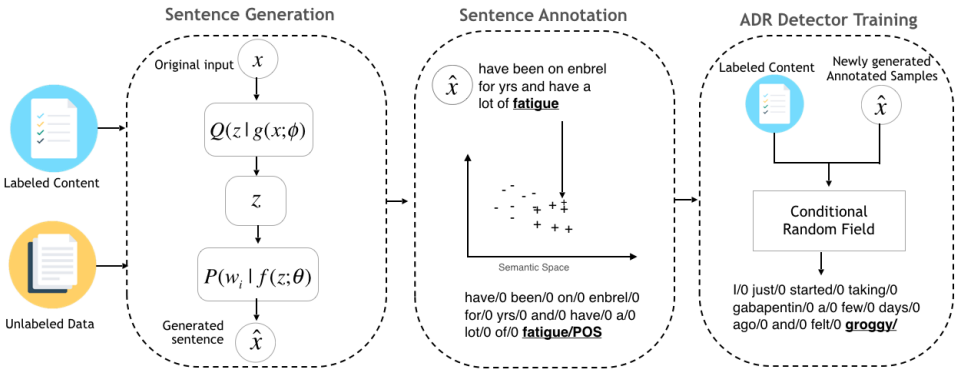


Figure 1: Overview of the proposed Training Data Augmentation Approach for Adverse Drug Reaction Detection

## 3 Adverse Drug Reaction Detection in User Generated Content

**Approach Overview.** Figure 1 presents an overview of our proposed approach. Given a list of drug names, a corpus $UGC = \{ugc_1, .., ugc_n\}$ of health-related User Generated Content (UGC) that mention one or more drugs of interest, and a subset $LC \subset UGC$ of UGCs labeled with ADR mentions, the *Sentence Generation* step (Sections 3.1) creates a set $SC$ of newly generated sentences that are similar to the ones in $LC$. The size of $LC$ is usually highly limited, thus Sentence Generation is important to expand the labeled data for better training the ADR detector. In $LC$, terms related to ADR (e.g. "no appetite") are considered positive examples ($POSTerms$), while all the other terms (e.g. "aspirin", "again"), excluding English stop words, are considered negative examples ($NEGTerms$). A Variational Auto Encoder (VAE) is first trained on $UGC$ data to learn the data distribution of the dataset, and then provided with $LC$ sentences as input to generate the output set $SC$. The *Sentence Annotation* step (Section 3.2) then propagates the label from $LC$ to the terms of sentences in $SC$. This is achieved by labelling the set $UTerms = \{ut_1, .., ut_n\}$ of terms in the newly generated sentence $SC$ that are semantically more similar to $POSTerms$ than to $NEGTerms$ in $LC$. Finally, the labelled sets $LC$ and $SC$ are combined in the *ADR Detector Training* step (Section 3.3) to train an ADR detector.

### 3.1 Sentence Generation

Our method for data generation relies on learning sentence distributions from a large text corpus, which can then be used to generate posts $SC$ semantically similar to a given set of existing labeled content $LC$. Let $\mathbf{x} \in \mathbb{R}^{|V|}$ ($\mathbf{x} \in UGC$) be the bag-of-words (multi-hot) representation of a user-generated content, where $V$ is the global vocabulary, and $\mathbf{w}_i \in \mathbb{R}^{|V|}$ be the one-hot representation of the word at position $i$ in the sentence represented by $\mathbf{x}$. Our goal is to learn $P(\hat{\mathbf{x}}|\mathbf{x})$, where the probability of a newly generated content $\hat{\mathbf{x}}$ serves as a proxy of the semantic similarity between $\hat{\mathbf{x}}$ and the original labeled content $\mathbf{x}$. Note that we will use the full set of user generated content $UGC$ to learn the data distribution, while only the labeled subset $LC$ is used to generate new sentences.

To obtain this conditional distribution, we adopt the deep generative modeling approach (Kingma and Welling, 2014; Miao et al., 2016), which was originally proposed to generate data instances similar to those already in a given dataset. Here, data is embedded into a latent space which is modelled by conditional distributions, and samples from this distributions can be decoded into new artificial data instances. In contrast to shallow models such as Skip-Gram (Mikolov et al., 2013) which also embeds into latent spaces, deep generative models have been shown to capture the implicit semantics and structure of the underlying data more effectively. However, existing deep generative models are not designed for generating class-specific data instances. Therefore, our goal is to extend existing deep generative models such that we can choose to only generate samples of a chosen subclass (e.g., resembling just labeled data). For example, Table 1 shows 3 artificial samples generated for 2 human-written training data sentences.

To do so, we build our method upon variational autoencoder (VAE), a representative deep generative model capable of learning high-quality representations of data structures. Given a set of sentences, VAE aims at learning a likelihood func-

Table 1: Three samples generated using VAE for a given input sentence.

| | |
|---|---|
| **Input** | **my dr switched from celexa to paxil and paxil made me feel sick** |
| *Sample 1* | my doctor put me on cymbalta and cymbalta can help me function |
| *Sample 2* | took my fluoxetine and it was a bit spaced out of my brain |
| *Sample 3* | yeah have to take topamax and it helps me but still feel fuzzy headed to a bit |
| ***Input*** | ***bruh this vyvanse putting me to sleep I needa take a break*** |
| *Sample 1* | took my vyvanse today and my head is spinning |
| *Sample 2* | vyvanse makes me feel like a zombie |
| *Sample 3* | vyvanse and addy have a cup of coffee |
| ***Input*** | ***I was on Prozac for months but it made my emotions so suppressed I stopped taking them*** |
| *Sample 1* | I was on venlafaxine for anxiety and depression but it stopped working |
| *Sample 2* | I was on effexor for about 3 months and then switched to venlafaxine |
| *Sample 3* | was on latuda for a while but it didn't help me |

tion $P_\theta(\hat{\mathbf{x}}|\mathbf{z})$ that, when used together with a standard Gaussian prior of $\mathbf{z}$, can generate new data instances $\hat{\mathbf{x}}$ that are similar to existing ones. Here $\mathbf{z}$ is the latent feature vector that captures the underlying data structure of the existing dataset. To handle the complex relationship between the latent feature and textual content, the likelihood function is parameterized by deep neural networks.

**Variational Autoencoder.** VAE encompasses a generative model, which describes the generative process for new data instances $\hat{\mathbf{x}}$ given $\mathbf{z}$ sampled from the Gaussian prior and transformed through a deep neural network.

- For each user-generated sentence $\mathbf{x}$
  - Draw a latent feature vector $\mathbf{z} \sim P(\mathbf{z})$ where $P(\mathbf{z}) = \mathcal{N}(0, \mathbf{I})$ is the standard Gaussian distribution.
  - For the $i^{th}$ term in the sentence,
    - Draw $\mathbf{w}_i \sim P(\mathbf{w}_i|f(\mathbf{z};\theta))$

where $f(\mathbf{z};\theta)$ is the neural network whose weights are shared for all sentences. The conditional probability over words, i.e., $P(\mathbf{w}_i|f(\mathbf{z};\theta))$ is modeled by a multinomial logistic regression:

$$P(w_i|f(\mathbf{z};\theta)) = \frac{exp(\mathbf{w}_i^\intercal f(\mathbf{z};\theta))}{\sum_{j=1}^{|V|} exp(\mathbf{w}_j^\intercal f(\mathbf{z};\theta))}$$

The parameters of the neural network, i.e., $\theta$, are learned by maximizing the the log likelihood of the observed sentence $\mathbf{x}$. This is non-trivial due to the intractability of the integral over the latent feature vector $\mathbf{z}$. VAE adopts a variational approach to optimise for the lower bound of the log-likelihood:

$$\mathcal{L} = \mathbb{E}_{Q(\mathbf{z}|g(\mathbf{x};\phi)}[\sum_{i=1}^{|\mathbf{x}|} \log P(\mathbf{w}_i|f(\mathbf{z};\theta)] - D_{KL}[Q(\mathbf{z}|g(\mathbf{x};\phi)\|P(\mathbf{z})]$$

This is generally known as the evidence lower bound (ELBO) (Blei et al., 2017). In such an ELBO, $\mathbb{E}(\cdot)$ is the expectation and $D_{KL}[\cdot\|\cdot]$ is the KL-divergence between two distributions; $Q(\mathbf{z}|g(\mathbf{x};\phi))$ is a Gaussian distribution $\mathcal{N}(\mu, \text{diag}(\sigma^2))$ that is again parameterized by a deep neural network: the two parameters of the Gaussian distribution, i.e., $\mu$ and $\sigma$ are both the output of the neural network $g(\mathbf{x};\phi)$.

**New Content Generation.** Once a VAE is trained on all user-generated content $UGC$, we take the existing human-annotated content $LC$ (annotated with ADR mentions) as the input for VAE to generate new sentence $SC$. The generation is performed by making use of the two conditional distributions learned before, i.e., $Q(\mathbf{z}|g(\mathbf{x};\phi))$ and $P(\mathbf{w}_i|f(\mathbf{z};\theta))$. When used together, these distributions form the conditional distribution we are interested for generating new content:

$$P(\hat{\mathbf{x}}|\mathbf{x}) = \int \sum_{i=1}^{|\hat{\mathbf{x}}|} P(\mathbf{w}_i|f(\mathbf{z};\theta))\ Q(\mathbf{z}|g(\mathbf{x};\phi))d\mathbf{z}$$

Content generation can then be performed via sampling from the above distribution. To generate new sentences, we take each sentence from the labeled set $LC$, and sample a pre-defined number ($k$) of latent feature vectors $\mathbf{z}_{j=1}^k$ from $Q(\mathbf{z}|g(\mathbf{x};\phi))$. For each sampled $\mathbf{z}_j$, we use it as an input for $P(\mathbf{w}_i|f(\mathbf{z};\theta))$ to generate a sequence of words as the new sentence.

### 3.2 Sentence Annotation

After generating new samples $SC$ similar to $LC$, the next step is to automatically annotate the terms in the newly generated sentences with ADR mentions such that it can be used to train a sequence-labeling model. In its basic version, we can only rely on the terms in the $POSTerms$ as positive examples of ADRs. However, we will heuristically expand this term set with additional positive examples found in the $SC$, thus improving the recall of the ADR detector.

In this work we rely on measuring and aggregating the semantic relatedness $SR$ between a term $ut_i$ and all the terms in $POSTerms$ as well as $NEGTerms$. In general, terms which are semantically related to terms in the $POSTerms$ should be considered as positive example. For example, having the terms `fever` and `no appetite` as positive examples, the new terms `weakness` or `body aches` could also be added to $POSTerms$ (because they are considered semantically related due to frequent co-occurrence, following the distributional hypothesis (Harris,

Table 2: CRF training parameters.

| | |
|---|---|
| useNGrams=true | normalize=true |
| noMidNGrams=true | useOccurrencePatterns=true |
| usePrev=trueuseNext=true | useLastRealWord=true |
| useLemmas=true | useNextRealWord=true |
| maxLeft=1 | lowercaseNGrams=true |

1954)), while `wheelchair` shall be added to $NEGTerms$. To this end, we use the popular *word2vec* implementation of skip-n-gram word embeddings (Mikolov et al., 2013). We define the semantic relatedness $SR_{pos}(ut_i, POSTerms)$ for a term $ut_i$ and the $POSTerms$ as well as $SR_{neg}(ut_i, NEGTerms)$ as follows:

$$\text{SR}_{pos}(ut_i) = \frac{\sum_{pterm \in POSTerms} SR_{pos}(ut_i, pterm)}{|POSTerms|}$$

$$\text{SR}_{neg}(ut_i) = \frac{\sum_{nterm \in NEGTerms} SR_{neg}(ut_i, nterm)}{|NEGTerms|}$$

Some terms are semantically related to both $POSTerms$ and $NEGTerms$; for instance, terms such as `drugs`, `pills`, and `pharmacy` have a very close $SR_{pos}$ and $SR_{neg}$. In order to avoid noisy terms which have an overlap in positive and negative semantics, we only annotate a term as positive if it appears in the $POSTerms$; *or* if the semantic relatedness between $ut_i$ and $POSTerms$ is higher than the semantic relatedness between $ut_i$ and $NEGTerms$, and if the distance between $SR_{pos}$ and $SR_{neg}$ is higher than a given threshold *(th)*.

### 3.3 ADR Detector Training

The labeled training data generated in the previous step can then be used to train any kind of supervised sequence tagger for ADRs. Conditional Random Field (CRF) has shown to be an effective technique on different NER tasks (Lafferty et al., 2001).We used the popular Conditional Random Field (CRF) sequence model[1] trained using the features listed in Table 2. Finally, the trained ADR detector can be used to detect the ADR mentions in our desired user generated content.

## 4 Evaluation

### 4.1 Experimental Settings

We evaluate the performance of our approach using precision, recall and f-score via approximate matching (Tsai et al., 2006). The focus of our evaluation is on the variation of performance at different fractions of training data and the number of newly generated samples to demonstrate the effectiveness of our proposed approach in reducing costs of manual annotation for training.

[1] https://github.com/dat/stanford-ner. Details on the selected features: https://nlp.stanford.edu/nlp/javadoc/javanlp/edu/stanford/nlp/ie/NERFeatureFactory.html.

Table 3: Dataset statistics. **LC**: labeled training set, **UGC**: unlabelled set. Number of sentences, words, and **unique** ADRs.

| | LC(Training) | | | LC(Testing) | | | UGC | |
|---|---|---|---|---|---|---|---|---|
| ***Dataset*** | **Sentences** | **Words** | **ADRs** | **Sentences** | **Words** | **ADRs** | **Sentences** | **Words** |
| *Twitter* | 693 | 6557 | 379 | 292 | 2601 | 154 | 146K | 2.16M |
| *Reddit* | 7506 | 133K | 543 | 1820 | 31708 | 195 | 274K | 3.65M |

### 4.2 Datasets

Experiments are performed on two datasets targeting different Web platforms. We used the publicly available Twitter dataset from *PSB 2016 Social Media Shared Task* for ADR detection [2]. Next, to evaluate our approach on richer textual forum data, we manually created an annotated corpus of Reddit medical subreddits with the help of medical experts. The aforementioned datasets contain only labeled data, but our approach requires in addition a larger corpus of unlabeled data from the same source. We therefore expanded each datasets with new posts, crawled respectively from Twitter and Reddit, that contain at least of one of the drug names contained in a common vocabulary [3]. The properties of each dataset are described in Table 3.

**Twitter**. The *PSB 2016 Social Media Shared Task* Twitter dataset (i.e. collected as explained in (Nikfarjam et al., 2015)) is a widely used manually annotated training data for ADR detection. The original dataset contained a total of 2,000 tweet IDs[4]; at the time of this study we were able to retrieve text from only 643 tweets, which we acknowledge might have an effect on the performance of the trained models.

**Reddit Data**. Reddit is a discussion website where users share and discuss problems/ideas about different topics. Reddit also contains subreddits such as `AskDocs`[5], `DiagnoseMe` [6], or

[2] http://diego.asu.edu/psb2016/task2data.html

[3] http://diego.asu.edu/downloads/publications/ADRMine/drug_names.txt

[4] Due to Twitter's search APIs license, only tweet ids were released

[5] https://www.reddit.com/r/AskDocs/

[6] https://www.reddit.com/r/DiagnoseMe/

Bipolar [7] where users share information about their health-related issues. To create a labeled training data set, we used the set of drug names mentioned above to collect 1,626 Reddit posts containing at least one drug names. We then recruited a medical doctor to annotate the ADRs (mentions of adverse drug reactions) in the collected posts following the annotation guidelines suggested in (Karimi et al., 2015), which specify: 1) exclude Leading prepositions, qualifiers, or possessive adjectives from selecting the ADR span, to avoid inconsistency. For instance, in the sentence "it increases my anxiety" only anxiety should be annotated; and 2) annotate all relevant contexts for an ADR concept. For example, in the sentence "I have a severe muscle pain", "severe muscle pain" should be annotated (not just "muscle pain"). To validate the labels, two of the authors manually checked again the annotations and found some ADRs which were not detected by the annotator; also, ambiguous ADRs where identified and discussed with the medical expert. From all the annotated posts, 600 posts with 9,326 sentences contained at least one ADR which were split into training and testing as shown in Table 3.

### 4.3 Compared Methods

We compare our proposed approach to established state-of-the-art ADR detection algorithms of different types:

- **QuickUMLS (Soldaini and Goharian, 2016):** an approximate dictionary matching algorithm which relies on UMLS concepts. We used the following setting, mentioned in (Soldaini and Goharian, 2016) as having best performance: *Similarity threshold* $= 0.9$, *Semantic types* $= [SignorSymptom, DiseaseorSyndrome, Finding, Neoplastic Process]$
- **CRF (Baseline).** The Conditional Random Field Phrase Detection Model [8] trained on the manually annotated training data $LC$.
- **CRF+VAE (Proposed):** In our proposed approach, we train a CRF model on the expanded training data created using the Variational Auto-Encoder approach as discussed in Section 3.
- **BLSTM-RNN(Cocos et al., 2017):** A state-of-the-art Bidirectional Long Short Term Memory (BLSTM) recurrent neural network (RNN) trained on the manually annotated training data $LC$.
- **BLSTM-RNN+VAE (proposed):** We combined our proposed technique with the BLSTM-RNN phrase detection technique. This is to highlight that our method can be combined with any supervised phrase detection technique.

To demonstrate the effectiveness of different strategies for augmenting the training data for ADR phrase detection, we compare our proposed approach with the following techniques:

- **CRF+SelfTraining (Usami et al., 2011):** A simple semi-supervised learning technique, where we train a similar conditional-random field phrase detection model as described before, but we apply the trained model on a set of randomly selected unlabeled sentences from $UGC$ (i.e. we used 500 samples). The sentences containing newly annotated ADRs are added to the initial training data and are used to re-train the phrase detection model.
- **CRF+Doc2vec:** CRF model trained on data expanded using an embedding-based strategy. Instead of generating new content $SC$ using VAE, we use Doc2vec (Le and Mikolov, 2014) which is inspired by word2vec (Mikolov et al., 2013) to find sentences similar to the labeled content $LC$.

### 4.4 Training

For training the Variational Autoencoder described in Section 3.1, we set the word dropout to 0.5, the learning rate to 0.001 and we used GRU (Cho et al., 2014) for both the encoder and the decoder. For labeling the newly generated sentences, we used word embeddings as described in (Mikolov et al., 2013). For Twitter we used pre-trained word embeddings trained on Twitter as described in (Godin et al., 2015). Since these pre-trained word embeddings did not perform well on the Reddit dataset, we trained a custom word embedding on all our Reddit data. We trained the skip-gram *word2vec* (300 dimension) model on the whole Reddit unlabeled collection.

[7] https://www.reddit.com/r/bipolar/
[8] https://github.com/dat/stanford-ner

Table 4: Performance of the different ADR detection techniques on the Twitter and Reddit test sets.

| Technique | Precision | Recall | Fscore |
|---|---|---|---|
| *QuickUMLS* | .47 | .34 | .39 |
| *CRF* | .67 | .42 | .51 |
| *BLSTM-RNN* | .61 | **.87** | .72 |
| *CRF+VAE* | .68 | .49 | .57 |
| *BLSTM-RNN+VAE* | **.71** | .85 | **.77** |

(a) Twitter

| Technique | Precision | Recall | Fscore |
|---|---|---|---|
| *QuickUMLS* | .14 | .21 | 17 |
| *CRF* | **.72** | .47 | .57 |
| *BLSTM-RNN* | .67 | .28 | .39 |
| *CRF+VAE* | .69 | **.52** | **.60** |
| *BLSTM-RNN+VAE* | .63 | .29 | .40 |

(b) Reddit

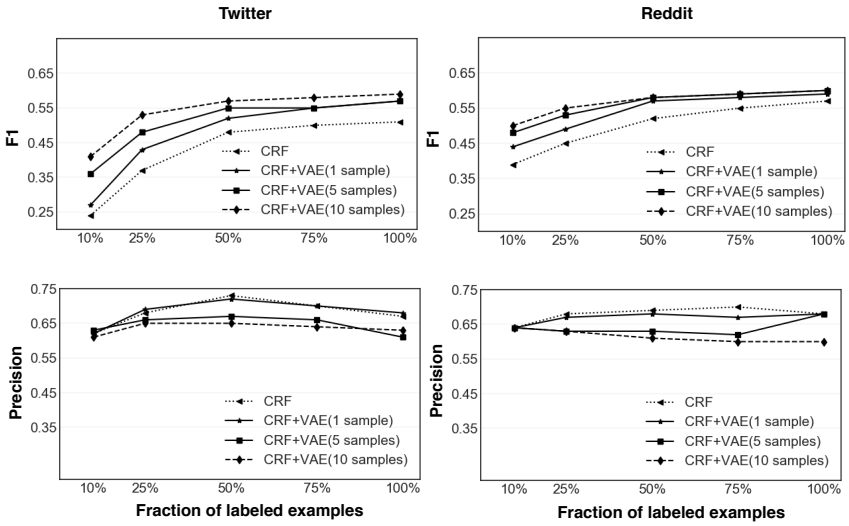


Figure 2: Average *F1* and *Precision* for *CRF* and *CRF+VAE* techniques, trained using different fractions of manually annotated examples and varying number of samples generated using VAE. Tested on the *Twitter* test set (on the left) and on the *Reddit* test set (on the right).

## 5 Results and Discussions

### 5.1 Comparison with ADR Detectors

In the first experiment, we compare our approach (i.e. trained with 100% of the labeled training data with 1 sample generated for each sample in the $LC$) against different ADR detector techniques described in Section 4.3. Table 4 reports precision, and recall and F1-measure, of all the baselines in comparison to proposed approach CRF+VAE in *Twitter* and *Reddit* dataset. We make the following observations: *QuickUMLS* is outperformed by all the other methods. The result shows that dictionary based approaches are not able to cover concepts that do not have a reference in UMLS dictionary, and produce false positives by labeling irrelevant words such as "maybe", "energy", "condition", "illness", or "worse" as positive.

The difference in performance between CRF and CRF+VAE shows the advantage brought by the sentence generation (VAE) and sentence annotation step of our approach. To highlight that our method can be combined with any supervised phrase detection technique, we combined our proposed technique with the BLSTM-RNN. BLSTM-RNN outperforms CRF in Twitter dataset; note that the model was designed to detect ADRs from the Twitter dataset. The results show that independent of the methodology used for training an ADR detector (e.g. CRF or BLSTM-RNN), expanding training data with VAE improves the overall performance. However due to the large amount of time required for training the BLSTM-RNN and the unstable prediction performance of its model on the test set (Cocos et al., 2017), the remaining experiments just focus on CRF for training ADR detector.

### 5.2 Effects of Training Data Size on CRF+VAE

For a given dataset (*Twitter* and *Reddit*), we created smaller subsets of the training data (i.e. 10%, 25%, 50%, 75%, 100%) to simulate the effect of limited training data availability. The subsets are randomly selected, and experiments are repeated 10 times for each size setting. We then train a CRF algorithm and different variants of our CRF+VAE (i.e. with different subsets of training data and different size of newly generated content for each labeled sample) and compare their performance. In

particular, the core advantage of our approach is that we are able to generate any number of additional training data samples. Therefore, we test different settings where we generate an extra 1, 5, or 10 artificial sentences for each labeled sentence in the training set. The experiments are conducted 10 times for each setting.

Figure 2 summarizes the average performance achieved for *Twitter* and *Reddit* datasets. The results show that by using the VAE to expand the training data, it is possible to obtain higher F-scores for both datasets. In Addition, we can show that by increasing the number of artificially generated samples (i.e. 5 and 10 samples), we can achieve a considerable F-score boost up to (+.17) and (+.12) for Twitter and Reddit (i.e. with just 10% of the labeled samples). We did not observe any significant improvement with more than 10 samples. This limitation is likely due to our constraint to generate sentences similar to the existing annotated sentences instead of radically new ones - a limitation chosen to allow us to perform reliable label propagation which would be hard for sentences too different. The results also show that by generating 1 sample using VAE but only using 50% of the training data, we can obtain comparable results to using the 100% of the labeled training data without VAE. When generating more training samples (i.e. 10 samples), our approach can achieve comparable performance with only the 25% of the initial labeled set. As shown in Figure 2, the effect of VAE expansion is greater the smaller the training data set is, thus VAE is used most efficiently to reduce the training costs of ADR detection significantly while maintaining quality. Note that all the improvements of CRF+VAE over CRF are statistically significant using paired t-test (i.e. $p$-value$<$0.05). When artificially expanding training data, recall is often improved at the cost of precision. This is demonstrated by the performance of CRF+Doc2Vec (Table 5). However, even using CRF+VAE (1 sample) shows higher F-score than CRF without notable loss of precision. This positive behaviour can be attributed to the larger number of positive and negative examples which helps to maintain the generalisation capabilities of the ADR detector while refining the quality of its recognition.

### 5.3 Comparison of Data Expansion Techniques

In the third experiment, we compare the performance of CRF+VAE against the two other automatic training data expansion techniques CRF+SelfTraining and CRF+Doc2Vec. As in the previous experiment, we use 10%, 25%, 50%, 75% and 100% of the training data. For the sake of brevity, we only report the best performance[9] achieved by these techniques in Table 5.

CRF+SelfTraining keeps the precision high but compared to CRF+Doc2Vec and CRF+VAE, it is not able to increase the recall significantly. Its low recall can be attributed to treating some terms incorrectly as negative instance examples. This is due to relying only on the output of the trained model for labeling the training data for the next iteration. We observe that CRF+VAE achieves better precision and comparable recall to CRF+Doc2Vec with the Twitter dataset, while achieving similar performance in the Reddit dataset in terms of F-score, but with higher precision. This underlines that artificially generating new similar training sentences can outperform discovering existing similar training sentences using Doc2Vec similarity. The results show that our approach in general performs better in the Twitter dataset. This can likely be attributed to the differences in the structure between the two datasets. Each tweet contains on average 8 words, while each Reddit sentence contains on average 17 words. Also, VAEs have shown to perform better on shorter sentences (Semeniuta et al., 2017).

## 6 Qualitative Analysis

In this section we tested CRF+VAE approach on *Twitter* and *Reddit* test sets and manually inspect all the posts containing false positive and negatives to understand the reasons for the prediction errors.

**False Positives**. Manual inspection of the posts reveal that most of the false positives are due to 1) Mis-recognizing *indications* as an ADR, i.e. an illness for which the drug has been prescribed is recognized as an adverse drug reaction (Chowdhury et al., 2018). For instance in the two posts *"I started effexor after having pretty severe postpartum depression"* and *"depression hurts cymbalta can help"*, `depression` is labeled as ADR even though it is an *indication*. However,

[9]The Self-training configuration has been run for ten iterations; we report the iteration with best performance.

Table 5: Average Precision/Recall/F1 with standard deviation in parenthesis for CRF, CRF+SelfTraining, CRF+Doc2Vec and CRF+VAE on Twitter and Reddit datasets. The experiments are conducted 10 times for each setting.

| Datasets | %Labeled samples | CRF | CRF+SelfTraining | CRF+Doc2Vec | CRF+VAE |
|---|---|---|---|---|---|
| *Twitter* | 10 | 62(.1)/.15(.05)/.24(.07) | .65(.05)/.27(.05)/.38(.06) | .57(.09)/.30(.04)/.39(.04) | .61(.10)/.32(.04)/.41(.04) |
| | 25 | 68(.06)/.25(.03)/.37(.04) | .66(.05)/.32(.02)/.43(.02) | .62(.03)/.42(.03)/.50(.03) | .65(.04)/.44(.03)/.53(.02) |
| | 50 | 73(.02)/.35(.02)/.48(.02) | 70(.03)/.37(.03)/.48(.03) | .65(.04)/.50(.01)/.56(.01) | .65(.01)/.51(.02)/.57(.01) |
| | 75 | 70(.02)/.39(.01)/.50(.02) | 68(.02)/.40(.02)/.51(.02) | .66(.02)/.52(.02)/.58(.01) | .67(.02)/.51(.03)/.58(.02) |
| | 100 | .67/.42/.51 | .67/.41/.51 | .61/.57/.59 | .64/.56/.60 |
| *Reddit* | 10 | .64(.06)/.28(.05)/.38(.05) | .64(.05)/.29(.05)/.40(.04) | .62(0.04)/.42(.04)/.50(0.3) | .64(.03)/.41(.04)/.50(.03) |
| | 25 | .68(.03)/.34(.03)/.45(.03) | .68(.03)/.34(.04)/.45(.03) | .61(.02)/.51(.02)/.55(.02) | .63(.02)/.48(.01)/.55(.01) |
| | 50 | .69(.02)/.42(.03)/.52(.02) | .69(.02)/.43(.04)/.53(.02) | .57(.02)/.60(.01)/.59(.01) | .61(.01)/.56(.02)/.59(.01) |
| | 75 | .70(.01)/.46(.02)/.55(.01) | .70(.01)/.46(.02)/.55(.01) | .56(.01)/.62(.01)/.59(.01) | .60(.01)/.59(.01)/.60(.01) |
| | 100 | .72/.47/.57 | .71/.46/.57 | .57/.64/.61 | .60/.62/.61 |

`depression` commonly occur as ADR as well in other posts, which might be the cause for this error (Chowdhury et al., 2018); 2) Ignoring negative verbs. As an example the word `manic` in *"The only one that didn't make me manic, Wellbrutin"* and `vomiting` in *"@uclaibd I never had bleeding or vomiting just a lot of fatigue"* are detected as ADRs due to the structure of the posts. However the model was not able to distinguish the negative verbs; 3) Mis-labeling ADR-related words as an ADR: For instance in the post *"temperature would start to rise, depression weakens"* the word `depression` was recognized as ADR; 4) Mistakes in manual annotation in the test data. For instance in the Tweet *"Ive had no appetite since I started on prozac"*, the annotators did not annotate *no appetite* as an ADR. However, our model was able to predict it correctly as an ADR, but due to this mistake in test data is considered a false positive.

**False Negatives**. False negatives are likely to occur in posts that are ambigious or overly complex. For example, in the post *"Im just wondering if its safe to take tramadol 15h after vyanse and if promethazine and melatonin would lower my chances of a seizure"* the word `seizure` was not detected as an ADR. It must be noted how, in this specific case, even human annotators debated if seizure is indeed an ADR of tramadol, or an indication of vyanse. In another example *"Am I the only one that grinds the shit out of their teeth on Vyvanse"*. The expression `grinds the shit out of their teeth` is a long description of the slang ADR *teeth grind*, which has been described in a very unstructured and informal way. This is hard to handle for phrase detectors like CRF as some level of abstraction would be necessary to deal with this.

## 7 Conclusion

In this paper, we have demonstrated an approach for augmenting training data for detecting mentions of Adverse Drug Reactions from social media text in a very cost-efficient manner. We introduced a technique which expands human-labeled training sets with a large number of artificially generated training samples. The benefit of our training data generation technique is greater the smaller the manually created training data set is. Therefore, it is used most efficiently to reduce the manual training costs of ADR detection while maintaining quality (e.g., in our experiments, we can maintain quality even when reducing manually provided training data by 75%). Furthermore, we could show that our approach generally works better on Twitter data. We assume that this can be explained by Reddit forum posts using significantly richer, longer, and more complex sentences. VAEs are known to work more effectively with shorter sentences than with longer ones. This work is only one of the initial steps towards automated adverse drug effect analytics on social data. The next step would be to interpret the semantics of the extracted slang ADRs, and linking them to medical ontologies to allow for further structured analysis.

## Acknowledgments

We thank Dr. Gerhard Mulder for his wonderful collaboration in annotating the Reddit dataset.

## References


Syed Rizwanuddin Ahmad. 2003. Adverse drug event monitoring at the food and drug administration: your report can make a difference. *Journal of general internal medicine*, 18(1):57–60.

Eiji Aramaki, Sachiko Maskawa, and Mizuki Morita. 2011. Twitter catches the flu: detecting influenza epidemics using twitter. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1568–1576. Association for Computational Linguistics.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. 2017. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21.

Brant W Chee, Richard Berlin, and Bruce Schatz. 2011. Predicting adverse drug events from personal health messages. In *AMIA Annual Symposium Proceedings*, volume 2011, page 217. American Medical Informatics Association.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *In Proceedings of the Empirical Methods in Natural Language Processing*.

Shaika Chowdhury, Chenwei Zhang, and Philip S Yu. 2018. Multi-task pharmacovigilance mining from social media posts. In *Proceedings of the 27th International Conference on World Wide Web*, pages 117–126. International World Wide Web Conferences Steering Committee.

Anne Cocos, Alexander G Fiks, and Aaron J Masino. 2017. Deep learning for pharmacovigilance: recurrent neural network architectures for labeling adverse drug reactions in twitter posts. *Journal of the American Medical Informatics Association*, 24(4):813–821.

Companion. 2019. Companion page. In *https://sites.google.com/view/emnlp-ijcnlp2019*.

Jan Deriu, Aurelien Lucchi, Valeria De Luca, Aliaksei Severyn, Simon Müller, Mark Cieliebak, Thomas Hofmann, and Martin Jaggi. 2017. Leveraging large amounts of weakly supervised data for multi-language sentiment classification. In *Proceedings of the 26th international conference on world wide web*, pages 1045–1052. International World Wide Web Conferences Steering Committee.

Fréderic Godin, Baptist Vandersmissen, Wesley De Neve, and Rik Van de Walle. 2015. Multimedia lab @ acl wnut ner shared task: Named entity recognition for twitter microposts using distributed word representations. In *Proceedings of the Workshop on Noisy User-generated Text*, pages 146–153.

Rave Harpaz, William DuMouchel, Nigam H Shah, David Madigan, Patrick Ryan, and Carol Friedman. 2012. Novel data-mining methodologies for adverse drug event discovery and analysis. *Clinical Pharmacology & Therapeutics*, 91(6):1010–1021.

Z. Harris. 1954. Distributional Structure. *Word*, 10:146–162.

Zhe He, Zhiwei Chen, Sanghee Oh, Jinghui Hou, and Jiang Bian. 2017. Enriching consumer health vocabulary through mining a social q&a site: A similarity-based approach. *Journal of biomedical informatics*, 69:75–85.

Trung Huynh, Yulan He, Alistair Willis, and Stefan Rueger. 2016. Adverse drug reaction classification with deep neural networks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 877–887.

Sarvnaz Karimi, Alejandro Metke-Jimenez, Madonna Kemp, and Chen Wang. 2015. Cadec: A corpus of adverse drug event annotations. *Journal of biomedical informatics*, 55:73–81.

Payam Karisani and Eugene Agichtein. 2018. Did you really just have a heart attack?: Towards robust detection of personal health mentions in social media. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 137–146. International World Wide Web Conferences Steering Committee.

Daniel Kershaw, Matthew Rowe, and Patrick Stacey. 2016. Towards modelling language innovation acceptance in online social networks. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, pages 553–562. ACM.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. *stat*, 1050:1.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Int. Conf. on Machine Learning*, volume 951, pages 282–289.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196.

Robert Leaman, Laura Wojtulewicz, Ryan Sullivan, Annie Skariah, Jian Yang, and Graciela Gonzalez. 2010. Towards internet-age pharmacovigilance: extracting adverse drug reactions from user posts to health-related social networks. In *Proceedings of the 2010 workshop on biomedical natural language processing*, pages 117–125. Association for Computational Linguistics.

Kathy Lee, Ashequl Qadir, Sadid A Hasan, Vivek Datla, Aaditya Prakash, Joey Liu, and Oladimeji Farri. 2017. Adverse drug event detection in tweets with semi-supervised convolutional neural networks. In *Proceedings of the 26th International Conference on World Wide Web*, pages 705–714. International World Wide Web Conferences Steering Committee.

Sepideh Mesbah, Christoph Lofi, Manuel Valle Torre, Alessandro Bozzon, and Geert-Jan Houben. 2018. Tse-ner: An iterative approach for long-tail entity extraction in scientific publications. In *International Semantic Web Conference*, pages 127–143. Springer.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *International Conference on Machine Learning*, pages 1727–1736.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Azadeh Nikfarjam, Abeed Sarker, Karen Oconnor, Rachel Ginn, and Graciela Gonzalez. 2015. Pharmacovigilance from social media: mining adverse drug reaction mentions using sequence labeling with word embedding cluster features. *Journal of the American Medical Informatics Association*, 22(3):671–681.

Abeed Sarker, Rachel Ginn, Azadeh Nikfarjam, Karen OConnor, Karen Smith, Swetha Jayaraman, Tejaswi Upadhaya, and Graciela Gonzalez. 2015. Utilizing social media data for pharmacovigilance: a review. *Journal of biomedical informatics*, 54:202–212.

Abeed Sarker and Graciela Gonzalez. 2015. Portable automatic text classification for adverse drug reaction detection via multi-corpus training. *Journal of biomedical informatics*, 53:196–207.

Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. 2017. A hybrid convolutional variational autoencoder for text generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 627–637.

Luca Soldaini and Nazli Goharian. 2016. Quickumls: a fast, unsupervised approach for medical concept extraction. In *MedIR workshop, sigir*.

Chen-Tse Tsai, Gourab Kundu, and Dan Roth. 2013. Concept-based analysis of scientific literature. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 1733–1738. ACM.

Richard Tzong-Han Tsai, Shih-Hung Wu, Wen-Chi Chou, Yu-Chun Lin, Ding He, Jieh Hsiang, Ting-Yi Sung, and Wen-Lian Hsu. 2006. Various criteria in the evaluation of biomedical named entity recognition. *BMC bioinformatics*, 7(1):92.

Yu Usami, Han-Cheol Cho, Naoaki Okazaki, and Jun'ichi Tsujii. 2011. Automatic acquisition of huge training data for bio-medical named entity recognition. In *Proceedings of BioNLP 2011 Workshop*, pages 65–73. Association for Computational Linguistics.

FM Zanzotto and Marco Pennacchiotti. 2012. Language evolution in social media: a preliminary study. *LINGUISTICA ZERO*.

Qing T Zeng and Tony Tse. 2006. Exploring and developing consumer health vocabularies. *Journal of the American Medical Informatics Association*, 13(1):24–29.

# Deep Reinforcement Learning-based Text Anonymization against Private-Attribute Inference

**Ahmadreza Mosallanezhad** **Ghazaleh Beigi** **Huan Liu**
Computer Science and Engineering
Arizona State University
{amosalla, gbeigi, huanliu}@asu.edu

## Abstract

User-generated textual data is rich in content and has been used in many user behavioral modeling tasks. However, it could also leak user private-attribute information that they may not want to disclose such as age and location. User's privacy concerns mandate data publishers to protect privacy. One effective way is to anonymize the textual data. In this paper, we study the problem of textual data anonymization and propose a novel Reinforcement Learning-based Text Anonymizor, RLTA, which addresses the problem of private-attribute leakage while preserving the utility of textual data. Our approach first extracts a latent representation of the original text w.r.t. a given task, then leverages deep reinforcement learning to automatically learn an optimal strategy for manipulating text representations w.r.t. the received privacy and utility feedback. Experiments show the effectiveness of this approach in terms of preserving both privacy and utility.

## 1 Introduction

Social media users generate a tremendous amount of data such as profile information, network connections and online reviews and posts. Online vendors use this data to understand users preferences and further predict their future needs. However, user-generated data is rich in content and malicious attackers can infer users' sensitive information. AOL search data leak in 2006 is an example of privacy breaches which results in users re-identification according to the published AOL search logs and queries (Pass et al., 2006). Therefore, these privacy concerns mandate that data be anonymized before publishing. Recent research has shown that textual data alone may contain sufficient information about users' private-attributes that they do not want to disclose such as age, gender, location, political views and sexual orientation (Mukherjee and Liu, 2010; Volkova et al., 2015). Little attention has been paid to protect users textual information (Li et al., 2018; Zhang et al., 2018; Anandan et al., 2012; Saygin et al., 2006).

Anonymizing textual information comes at the cost of losing utility of data for future applications. Some existing work shows the degraded quality of textual information (Anandan et al., 2012; Zhang et al., 2018; Saygin et al., 2006). Another related problem setting is when the latent representation of the user generated texts is shared for different tasks. It is very common to use recurrent neural networks to create a representation of user generated text to use for different machine learning tasks. Hitaj el al. show text representations can leak users' private information such as location (Hitaj et al., 2017). This work aims to anonymize users' textual information against private-attribute inference attacks.

Adversarial learning is the state-of-the-art approach for creating a privacy preserving text embedding (Li et al., 2018; Coavoux et al., 2018). In these methods, a model is trained to create a text embedding, but we cannot control the privacy-utility balance. Recent success of reinforcement learning (RL) (Paulus et al., 2017; Sun and Zhang, 2018) shows a feasible alternative: by leveraging reinforcement learning, we can include feedback of attackers and utility in a reward function that allows for the control of the privacy-utility balance. Furthermore, an RL agent can perturb parts of an embedded text for preserving both utility and privacy, instead of retraining an embedding as in adversarial learning. Therefore, we propose a novel Reinforcement Learning-based Text Anonymizer, namely, RLTA, composed of two main components: 1) an attention based task-aware text representation learner to extract latent embedding representation of the original text's content w.r.t. a



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2360–2369, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

given task, and 2) a deep reinforcement learning based privacy and utility preserver to convert the problem of text anonymization to a one-player game in which the agent's goal is to learn the optimal strategy for text embedding manipulation to satisfy both privacy and utility. The Deep Q-Learning algorithm is then used to train the agent capable of changing the text embedding w.r.t. the received feedback from the privacy and utility subcomponents.

We investigate the following challenges: 1) How could we extract the textual embedding w.r.t. a given task? 2) How could we perturb the extracted text embedding to ensure that user private-attribute information is obscured? and 3) How could we preserve the utility of text embedding during anonymization? Our main contributions are: (1) we study the problem of text anonymization by learning a reinforced task-aware text anonymizer, (2) we corporate a data-utility task-aware checker to ensure that the utility of textual embeddings is preserved w.r.t. a given task, and (3) we conduct experiments on real-world data to demonstrate the effectiveness of RLTA in an important natural language processing task.

## 2 Related Work

Reinforcement Learning (RL) has applications in natural language processing and recommendation systems. For example, a recent paper (Paulus et al., 2017) combines RL with a supervised method to get a readable and informative article summary. Another work uses RL to solve the problem of adversarial generative models for text generation (Shi et al., 2018). Sun et al. also uses RL in a recommendation system which recommends items according to the users' feedbacks and preferences (Sun and Zhang, 2018)

Textual data is rich in content and recent research has shown that users' private-attributes can be easily inferred from the text (Beretta et al., 2015; Mukherjee and Liu, 2010; Volkova et al., 2015), however, few papers consider user privacy w.r.t. such data. Anandan et al. (2012) introduce $t$-Plausibility which uses an information theoretic based approach to sanitize documents heuristically. This method does not preserves the utility of data during anonymization process. Another work focuses on leveraging differential privacy (Dwork et al., 2017) to make the extracted Term Frequency Inverse Document (TF-IDF) textual vectors private (Zhang et al., 2018). It has been shown that TF-IDF cannot accurately capture semantic meaning of the text which can hurt its usefulness for different tasks (Lan et al., 2005).

Two recent similar works Li et al. (2018); Coavoux et al. (2018) convert textual data anonymization into a minimax problem. These works use the idea of adversarial learning to create a text embedding which satisfies utility and protects users against private-attribute leakage. Our scenario is similar to the work of (Li et al., 2018) as they have considered several attribute-inference attackers as adversaries to create a privacy-preserving text embedding. (Beigi et al., 2019b,a) propose a method for privacy preserving text representation. In this method, they try to add noise to an existing text representation in a way that it does not change the meaning of the text and it preserves the user's private attributes.

Our work is different from the existing works. First, we consider the task that the textual information will be used for and protect users against leakage of private-attributes. Second, we incorporate deep RL to anonymize the extracted text embedding by receiving privacy and utility feedbacks and automatically learning the optimal strategy for proper manipulation of text embeddings.

## 3 Problem Statement

Let $\mathcal{X} = \{x_1, x_2, ..., x_N\}$ denotes a set of $N$ documents and each document $x_i$ is composed of a sequence of words. We denote $\mathbf{v}_i \in R^{d\times 1}$ as the embedded representation of the original document $x_i$. Let $\mathcal{P} = \{p_1, p_2, ..., p_m\}$ denotes a set of $m$ private-attributes that users do not want to disclose such as age, gender, location, etc. The goal of reinforced task-aware text anonymizer is to learn an embedding representation of each document and then anonymize it such that 1) users privacy is preserved by preventing any potential attacker to infer users' private-attribute information from the textual embedding data, and 2) utility of the text embedding is maintained for a given task $\mathcal{T}$ which incorporates such data, e.g., classification. In this paper, we study the following problem:

**Problem 3.1.** Given a set of documents $\mathcal{X}$, set of private-attributes $\mathcal{P}$, and given task $\mathcal{T}$, learn an anonymizer $f$ that can learn a private embedded representation $\mathbf{v}_i$ from the original document $x_i$ so that, 1) the adversary cannot infer the targeted user's private-attributes $\mathcal{P}$ from the private

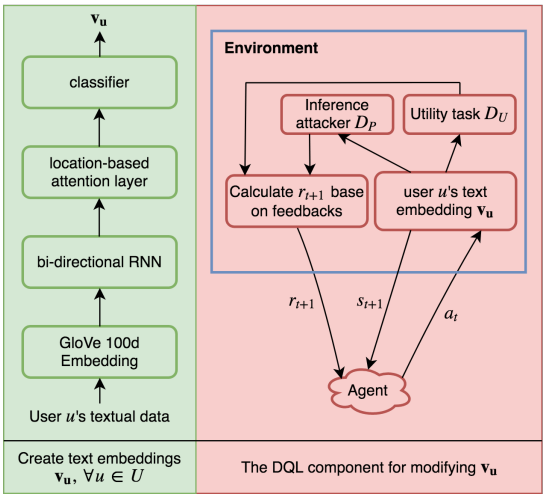


**Figure 1: The architecture of RLTA method**

text representation $\mathbf{v}_i$, and 2) the generated private representation $\mathbf{v}_i$ is good for the given task $\mathcal{T}$. The problem can be formally defined as:

$$\mathbf{v}_i = f(x_i, \mathcal{P}, \mathcal{T}) \tag{1}$$

Due to the success of Reinforcement Learning (Shi et al., 2018; Paulus et al., 2017), we use RL to address the aforementioned problem. RL (Sutton and Barto, 2018) formulates the problem within the framework of Markov Decision Process (MDP), and learns an action-selection policy based on past observations of transition data. An MDP is defined by state space $S = \{s\}$, action space $A = \{a\}$, transition probability function $P : S \times A \times S \to [0, 1]$ and reward function $r : S \times A \times S \to R$.

## 4 Proposed Method

We discuss the reinforced task-aware text anonymizer framework. The input of this private system is the user generated text, and the output is a privacy-preserving text representation. As in Figure. 1, this framework consists of two major components: 1) an attention based task-aware text representation learner, and 2) a deep RL based privacy and utility preserver. The text representation learner aims to extract the embedded representation of a document w.r.t. a given task by minimizing the task's loss function. Then, the deep RL preserver manipulates the embedded text representation by learning the optimal strategy so that both privacy and utility of the embedded representation are preserved. It includes two sub-components: 1) private-attribute inference attacker $D_P$, and 2) data-utility task-aware checker $D_U$. The former seeks to infer user private-attribute information based on their embedded text representation. The latter incorporates the given manipulated embedded text representation for a given task $\mathcal{T}$ and investigates the usefulness of the latent representation for $\mathcal{T}$.

The RL component then utilizes the feedback of the two sub-components to guide the data manipulation process by ensuring that the new text embedding does not leak user private-attributes by confusing the adversary in $D_P$ and the changes made to the representation does not destroy the semantic meaning for $\mathcal{T}$.

### 4.1 Extracting Textual Embedding

Let $x = \{w_1, ..., w_m\}$ be a document with $m$ words. Attention mechanism has shown to be effective in capturing embedding of textual information w.r.t. a given task (Pennington et al.; Vaswani et al., 2017). It allows the model to attend to different parts of the given original document at each step and then learns what to attend based on the input document and what it has produced as embedding representation so far, as shown in Figure. 1.

We use a bi-directional recurrent neural network (RNN) to encode the given document into an initial embedding representation. RNN has been shown to be effective for summarizing and learning semantic of unstructured noisy short texts (Cho et al., 2014; Shang et al., 2015). We use GloVe 100d (Pennington et al.) to exchange each word $w_i$ with its corresponding word vector, note that different dimensionality can be used. This process produces a matrix of text $x' \in \mathcal{R}^{m*100}$.

We employ the gated recurrent unit (GRU) as the cell type to build the RNN, which is designed in a manner to have a more persisted memory (Cho et al., 2014). The bi-directional GRU will read the text forward and backwards, then outputs two hidden states $\mathbf{h_t^{fw}}, \mathbf{h_t^{bw}}$ and an output $\mathbf{o_t}$. We then concatenate two hidden states as the initial encoded embedding of the given original document:

$$\mathbf{H_t} = Concat(\mathbf{h_t^{fw}}, \mathbf{h_t^{bw}}) \tag{2}$$

After calculating the initial context vector $\mathbf{H_t}$, we seek to pinpoint specific information within the $\mathbf{H_t}$, which helps the classifier to predict the labels with higher confidence (Luong et al., 2015) We use the location-based attention layer based on the work of Luong et al. (2015). The attention layer calculates a vector $\mathbf{a_t}$ including a weight for each

element in the $\mathbf{H_t}$, showing the importance of that element. The context vector $\mathbf{v_t}$ is calculated:

$$\mathbf{v_t} = \sum_{i=1}^{m} a_{t,i}\mathbf{H_i} \quad (3)$$

The vector $\mathbf{v_t}$ is then fed to a neural network classifier for the given utility task. Classification is one of the common tasks for textual data. Based on the output of the classifier and loss function, we update the three networks so that the output of the attention layer is an useful context that can be used for a utility task (Ranzato et al., 2015).

### 4.2 Reinforced Task-Aware Text Anonymizer

Here, we discuss the details of the second component which seeks to preserve privacy and utility.

#### 4.2.1 Protecting Private-Attributes

Textual information is rich in content and publishing textual embedding representation without proper anonymization leads to privacy breach and revealing the private-attributes of an individual such age, gender and location. It is thus essential to protect the textual information before publishing it. The goal of our model is to manipulated learned embedded representation such that any potential adversary cannot infer users' private-attribute information. However, a challenge is that the text anonymizer does not know the adversary's attack model. To address this challenge, we add a private-attribute inference attacker $D_P$ sub-component to our text anonymizer. This sub-component learns a classifier that can accurately identify the private information of users from their embedded text representations $\mathbf{v_u}$. We incorporate this sub-component to understand how the textual embedded representation should be anonymized to obfuscate the private information.

Inspired by the success of RL (Kaelbling et al., 1996; Mnih et al., 2013; Van Hasselt et al., 2016), we model this problem using RL to automatically learn how to anonymize the text representations w.r.t. the private-attribute inference attacker. In our RL model, one agent is trained to change a randomly selected text embedding representation. Then, the agent keeps interacting with the environment and changes the text embedding accordingly based on its current state and received rewards so that the private-attribute inference attacker cannot correctly identify user's private-attribute information given his embedding. In this part, we define the main four parts of RL environment in our problem, i.e., environment, state, action and reward.

- **Environment:** Environment in our problem includes the private-attribute inference attackers $D_P$ and the text embedding $\mathbf{v_u}$. Note that $D_P$ is trained beforehand.
- **State:** State describes the current situation. Here, state is the current text embedding vector $\mathbf{v_{u,t}}$ which reflects the results of the agents' actions on $\mathbf{v_u}$ up to time $t$.
- **Actions:** Action is define as selecting one element such as $v_{u,k}$ in text embedding vector $\mathbf{v_u} = \{v_{u,1}, ..., v_{u,m}\}$ and changing it to a value near $-1$, $0$ or $1$. This results in $3.m$ actions where $m$ is the size of the embedding vector.

  **Changing value to near *1***: In this action, the agent changes the value of $v_{u,k}$ to a value between $0.9$ to $1.0$. As $v_{u,k}$ will be multiplied by a classifier's weight, the output will be the weight as is. In another word, the value $v_{u,k}$ will become important to the classifier.

  **Changing value to near *0***: In this action the $v_{u,k}$ will be changed to a value between $-0.01$ to $0.01$. This action makes $v_{u,k}$ seem neutral and unimportant to a classifier as it will result in a $0$ when multiplied by a weight.

  **Changing value to near *-1***: In this action, the agent changes $v_{u,k}$ to a value between $-1.0$ to $-0.9$. This action will make $v_{u,k}$ important to a classifier, but, in a negative way.
- **Reward:** Reward in our problem is defined based on how successfully the agent obfuscated the private-attribute information against the attacker so far. In particular, we defined the reward function at state $s_{t+1}$ according to the confidence of private-attribute inference attacker $C_{p_k}$ for private-attribute $p_k$ given the resultant text embedding at state $s_{t+1}$, i.e., $\mathbf{v_{t+1}}$. Considering the classifier's input data as $\mathbf{v_u}$ and its correct label as $i$, we define the confidence for a multi-class classifier as the difference between the probability of actual value of the private-attribute and the minimum probability of other values of the private-attribute:

$$C_{p_k} = Pr(l = i|\mathbf{v_u}) - \min_j Pr(l = j|\mathbf{v_u}) \quad (4)$$

  Where $l$ indicates label. For each private-attribute attacker $p_k$, the confidence score $C_{p_k}$ is

within the range $[-1, 1]$. Positive value demonstrates that the attacker has predicted private-attribute accurately, and negative value indicates that the attacker was not able to infer user's private-attribute. According to this definition, the reward will be positive if action $a_t$ has caused information hiding, and will be negative if the action $a_t$ was not able to hide sensitive information. Having confidence of private-attribute inference attackers, reward function at state $s_{t+1}$ is defined as:

$$r_{t+1}(s_{t+1}) = -\sum_{p_k \in D_P} C_{p_k}(s_{t+1}) \quad (5)$$

The reward $r_t$ is calculated according to the state $s_{t+1}$ which associated with the transition of agent from state $s_t$ after applying action $a_t$. Note that the goal of agent is to maximize the amount of received rewards so that the mean of rewards $r$ over time $t \in [0, T]$ ($T$ is the terminal time) will be positive and above 0.

#### 4.2.2 Preserving Utility of Text Embedding

Thus far, we have discussed how to 1) learn textual embeddings from the given original document w.r.t. the given task, and 2) prevent leakage of private-attribute information by developing a reinforcement learning environment which incorporates a private-attribute inference attacker and manipulates the initial given text embedding accordingly to fool the attacker. However, data obfuscation comes at the cost of data utility loss. Utility is defined as the quality of the given data for a given task. Neglecting the utility of the text embedding while manipulating it, may destroy the semantic meaning of the text data for the given task. Classification is one of the common tasks. In order to preserve the utility of data, we need to ensure that preserving privacy of data does not destroy the semantic meaning of the text embedding representation w.r.t. the given task. We approach this challenge by changing the agent's reward function w.r.t. the data utility. We add a utility sub-component, i.e., classifier $D_U$, to the reinforcement learning environment which its goal is to assess the quality of resultant embedding representation. We use the confidence of the classifier for the given task to measure the utility of embedding representation using the text embedding vector $\mathbf{v_u}$ the its correct label $i$.

$$C = Pr(l = i|\mathbf{v_u}) - \min_j Pr(l = j|\mathbf{v_u}) \quad (6)$$

The agent can then use the feedback from the utility classifier to make decision when taking actions. We thus modify the reward function in order to incorporate the confidence of utility sub-component. Reward function at state $s_{t+1}$ can be defined as:

$$\begin{aligned} r_{t+1}(s_{t+1}) =& \alpha C_{D_U}(s_{t+1}) - \\ & -(1-\alpha)\sum_{p_k \in D_P} C_{p_k}(s_{t+1}) - \mathcal{B} \end{aligned} \quad (7)$$

where $C_{D_U}$ and $C_{p_k}$ represent the confidence of utility sub-component and private-attribute inference attacker, respectively. Moreover, $\mathcal{B}$ demonstrates a baseline reward which forces the agent to reach a minimum reward value. The coefficient $\alpha$ also control the amount of contribution from both private-attribute inference and utility sub-components in the Eq. 7.

### 4.3 Optimization Algorithm

Given the formulation of states and actions, we aim to learn the optimal strategy via manipulating text representations w.r.t. the private-attribute attackers and utility sub-component feedbacks. We manipulate the text embeddings by repeatedly choosing an action $a_t$ given current state $s_t$, and then applying actions on current state to transit to the new one $s_{t+1}$. The agent then receives reward $r_{t+1}$ as a consequence of interacting with the environment. The goal of agent is to manipulate text embedding $v_{u,k}$ in a way that maximizes its reward according to Eq. 7. Moreover, the agent updates its action selection policy $\pi(s)$ so that it can achieve the maximum reward over time.

In RLTA we use Deep Q-Learning which is a variant of Q-Learning. In this algorithm the goal is to find the following function:

$$Q^*(s_t, a_t) = \mathrm{E}_{s_{t+1}}[r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a')] \quad (8)$$

where $Q(s, a)$ corresponds to the Q-function for extracting actions and it is defined as the expected return based on state $s$ and action $a$. Moreover, $Q^*(s, a)$ denotes the optimal action-value Q-function which has the maximum expected return using the optimal policy $\pi(s)$. Rewards are also discounted by a factor of $\gamma$ per time step. The agent keeps interacting with the environment till it reaches the terminal time $T$.

Since it is not feasible to estimate $Q^*(s, a)$ in Eq.8, we use a function approximator to estimate the state-action value function $Q^*(s, a) \approx$

**Algorithm 1** The Learning Process of RLTA

**Require:** $\mathbf{v}$, $D_P$, $D_U$, $\alpha$, $\gamma$, $\mathcal{B}$, $T$.

Initialize replay memory $M$ with size $N$
**while** training is not terminal **do**
  $s_t \leftarrow \mathbf{v}$
  **for** $t \in \{0, 1, ..., T\}$ **do**
    Choose action $a_t$ using $\epsilon$-greedy
    Perform $a_t$ on $s_t$ and get $(s_{t+1}, r_{t+1})$
    $M \leftarrow M + (s_t, a_t, r_{t+1}, s_{t+1})$
    $s_t \leftarrow s_{t+1}$
    Sample mini-batch $b$ from memroy $M$
    **for** $(s, a, s', r) \in b$ **do**
      Update DQN weights using Eq. 11
    **end for**
  **end for**
**end while**

$Q(s, a; \theta)$. Given neural networks as excellent function approximators (Cybenko, 1989), we lverage a deep neural network function approximator with parameters $\theta$, or a Deep Q-Network (DQN) (Mnih et al., 2013) by minimizing the following:

$$L(\theta) = \mathrm{E}_{s_t, a_t, r_{t+1}, s_{t+1}}[(y - Q(s, a; \theta))^2] \quad (9)$$

in which $y$ is the target for the current iteration:

$$y = \mathrm{E}_{s_{t+1}}[r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a'; \theta^p)] \quad (10)$$

$\theta^p$ is the parameters from the previous iteration.

We update the DQN according to the derivation of Eq. 9 with respect to the parameter $\theta$:

$$\begin{aligned} \nabla_\theta L(\theta) = \mathrm{E}_{s_t, a_t, r_{t+1}, s_{t+1}}[(r \\ + \gamma \max_{a'} Q(s_{t+1}, a'; \theta^p) \\ - Q(s_t, a_t; \theta)) \nabla_\theta Q(s_t, a_t; \theta)] \end{aligned} \quad (11)$$

Algorithm 1 shows the optimization process.

## 5 Experiments

Experiments are designed to answer the following questions: **Q1**(*Privacy*): How well RLTA can obscure users' private-attribute information? **Q2**(*Utility*): How well RLTA can preserve utility of the textual data w.r.t. the given task? **Q3**(*Privacy-Utility Relation*): How does improving user privacy affects loss of utility?

To answer the first question (**Q1**), we use investigate the robustness of resultant text embedding against private-attribute inference attacks. We consider two private-attribute information, i.e., location and gender. To answer the second question (**Q2**), we report experimental results w.r.t. a well-known task, sentiment analysis. Sentiment analysis has many applications in user-behavioral modeling and Web (Zafarani et al., 2014). In particular, we predict sentiment of the given textual embedding. To answer the final question (**Q3**), we examine the privacy improvement against utility loss.

### 5.1 Data

We use a real-world dataset from Trustpilot (Hovy et al., 2015). This dataset includes user reviews along with users private-attribute information such as location and gender. We remove non-English reviews based on LANGID.py[1] (Lui and Baldwin, 2012) and only keep reviews classified as English. Then, we consider English reviews associated with location of US and UK and create a subset of data with $10k$ users. Each review is associated with a rating score. We consider the review's sentiment as positive if its rating score is $\{4, 5\}$ and consider it as negative if rating is $\{1, 2, 3\}$

### 5.2 Implementation Details

For extracting the initial textual embedding, we use a bi-directional RNNs which their hidden sizes are set to 25. This makes the size of the final hidden vector $\mathbf{H_t}$ as 50. We also use a logistic regression with a linear network as the classifier in the attention mechanism. We use a 3-layer network for the Deep Q-network, i.e., input, hidden and output layers. Dimensions of the input and hidden layers are set to 50 and 700, respectively. Dimension of the last layer, i.e., output, is also set as 150. This layer outputs the state-action values which we execute the action with the best value.

For each of the private-attribute attackers and utility sub-components, we use feed-forward network with a single hidden layer with dimension of 100 which gets the textual embedding as input and uses a $Softmax$ function as output.

We first train both private-attribute inference attacker $D_P$ and utility sub-component $D_U$ on the training set. These sub-components do not change after that. Then, we train an agent on each selected data for 5000 episodes. The reward discount for agents is $\gamma = 0.99$ and batch size $b = 32$. We also set the terminal time $T = 25$. We run RLTA for 5 times and select the best agent based on the cumulative reward. We also vary $\alpha$ as $\alpha = \{0, 0.25, 0.5, 0.75, 1\}$. The higher values of

[1] https://github.com/saffsd/langid.py

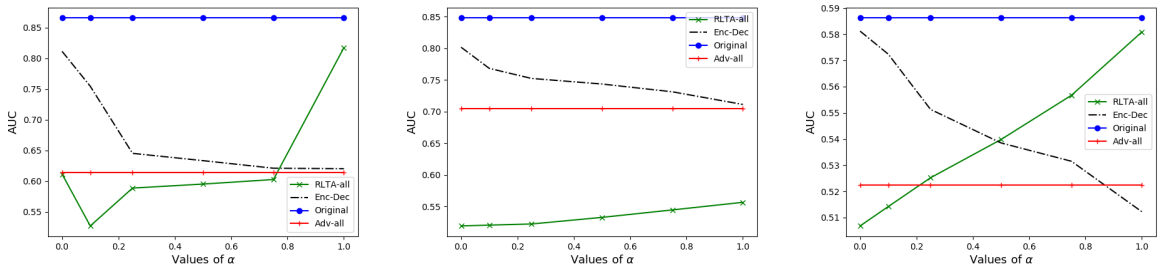


**(a) Gender inference attack** **(b) Location inference attack** **(c) Sentiment prediction**

**Figure 2: AUC scores for private-attribute and sentiment prediction tasks for different values of $\alpha$. Lower AUC for private-attribute inference attacks shows higher privacy, while higher AUC for the sentiment prediction task indicates higher utility.**

$\alpha$ indicate more utility contribution in RLTA.

### 5.3 Experimental Design

We use 10-fold cross validation of RLTA for evaluating both private-attribute inference attacker and an utility task with the following baselines:

- **ORIGINAL:** This baseline is a variant of proposed RLTA which does not change the original user text embeddings $\mathbf{v_u}$ and publishes it as is.
- **ADV-ALL:** This adversarial method has two main components, i.e., generator and discriminator, and creates a text representation that has high quality for a given task, but has poor quality for inferring private-attributes (Li et al., 2018).
- **ENC-DEC:** Using an auto-encoder is one of the effective methods to create a text embedding (Nallapati et al., 2016). We modify this simple method to create a privacy-preserving text embedding. This method gets the original text $x$ and outputs a re-constructed text $\bar{x}$. The following loss function is used to train the model. After training, we use the encoder's output as the text representation $\mathbf{v_u}$ (Cho et al., 2014).

$$loss = -\sum_{x \in X} \log Pr(\bar{x}|x) + \quad (12)$$
$$+ \alpha((\sum_{p_k} C_{p_k}) - C_{D_U})$$

In which $\alpha$ is the privacy budget.

To examine the privacy of final text embedding, we apply the trained private-attribute attacker sub-component $D_P$ to the output of each method to evaluate the users' privacy. We consider two private attributes, i.e., location and gender. We then compute the attacker's AUC. Lower attacker's AUC indicates that textual embeddings have higher privacy after anonymization against the private-attribute inference attacker. We also report experimental results w.r.t. the utility. In particular, we predict sentiment (positive and negative) of the given textual embedding by applying trained utility sub-component $D_U$ to the resultant text embedding from test set for each method. We then compute AUC score for sentiment prediction task. Higher values of AUC demonstrate that the utility of textual embedding has been preserved.

### 5.4 Experimental Results

We answer the three question **Q1**, **Q2** and **Q3** to evaluate our proposed method RLTA.We use a natural language processing task, sentiment prediction, using a three layer neural network.

**Privacy (Q1).** Figure. 2 (a-b) demonstrates the results of private-attribute inference attack w.r.t. gender and location attributes. The lower the value of AUC is, the more privacy user has in terms of obscuring private attributes. We also report the performance of RLTA for different values of $\alpha$.

We observe that ORIGINAL is not robust against private-attribute inference attack for both gender and location attributes. This confirms leakage of users private information from their textual data. Moreover, RLTA has significantly lower AUC score for both gender and location attributes in comparison to other methods. This demonstrates the effectiveness of RL for obfuscating private attributes. In RLTA, the AUC score for private-attribute inference attack increases for both attributes with the increase of $\alpha$ which shows the degradation in user privacy. The reason is because of the fact that agent pays less attention to privacy by increasing the value of $\alpha$.

In the ENC-DEC method, as the value of $\alpha$ increases, the encoder tries to generate a text representation that is prune to inference attacks but it

does not lose its utility w.r.t. the given task $D_U$. The results show that as $\alpha$ increases, the AUC of inference attackers will decrease.

**Utility (Q2).** To answer the second question, we investigate the utility of embeddings w.r.t. sentiment prediction. Results for different values of $\alpha$ are demonstrated in Figure. 2(c). The higher the value of the AUC is, the higher utility is preserved.

The ORIGINAL approach has the highest AUC score which shows the utility of the text embeddings before any anonymization. We observe that the results for RLTA is comparable to the ORIGINAL approach which shows that RLTA preserves the utility of text embedding. Moreover, RLTA outperforms ADV-ALL which confirms the effectiveness of reinforced task-aware text anonymization approach in preserving utility of the textual embeddings. We also observe that the AUC of RLTA w.r.t. sentiment prediction task increases with the increase of value of $\alpha$. This is because with the increase of $\alpha$, the agent pays more attention to the feedbacks of utility sub-component.

We also observe a small utility loss after applying RLTA when $\alpha = 1$. This is because the agent keeps changing the text embedding until it reaches the terminal time. These changes result in loss of utility even when the $\alpha = 1$.

Finally, in the ENC-DEC method, as both utility and attackers have the same importance, trying to preserving privacy would result in huge utility loss as we increase the value of $\alpha$.

**Privacy-Utility Relation (Q3).** Results show that the ORIGINAL achieves the highest AUC score for both utility task and private-attribute inference attack. This shows that ORIGINAL has the highest utility which comes at the cost of significant user privacy loss. However, comparing results of privacy and utility for $\alpha = 0.5$, we observe RLTA has achieved the lowest AUC score for attribute inference attacks in comparison to other baselines, thus has the highest privacy. It also reaches the higher utility level in comparison to the ADV-ALL. RLTA also has comparable utility results to the ORIGINAL approach. We also observe that increasing the $\alpha$ reduces the performance of RLTA in terms of privacy but increases its performance for utility. However, with $\alpha = 1$, RLTA preserves both user privacy and utility in comparison to ORIGINAL, ENC-DEC, and ADV-ALL.

| Method | Location | Gender | Utility |
|---|---|---|---|
| **ORIGINAL** | 84.77 | 86.54 | 58.57 |
| **ENC-DEC** | 71.55 | 58.35 | 53.78 |
| **ADV-ALL** | 70.37 | 57.15 | 52.15 |
| **RLTA** | 53.34 | 56.41 | 54.83 |
| **RLTA-GEN** | 56.64 | **55.02** | **56.67** |
| **RLTA-LOC** | **52.04** | 56.64 | 54.13 |

**Table 1: Impact of different private-attribute inference attackers on RLTA when $\alpha = 0.5$. With $\alpha = 0.5$, privacy and utility will contribute equally.**

### 5.4.1 Impact of Different Components

Here, we investigate the impact of different private-attribute inference attackers. We define two variants of our proposed model, RLTA-GEN and RLTA-LOC. In each of these variants, we train the agent in RLTA w.r.t. the one of private-attribute attackers, e.g., RLTA-GEN is trained to solely hide *gender* attribute. For this experiment we set $\alpha = 0.5$ as in this case privacy and utility sub-components contribute equally during training phase (Eq. 7). Results are shown in Table 1.

RLTA-LOC and RLTA-GEN have the best performance amongst all methods in obfuscating location and gender private-attributes, respectively. Results show that using RLTA-LOC could also help improve privacy on gender and likewise for (RLTA-GEN) in comparison to other approaches.

RLTA-GEN performs better in terms of utility, in comparison to RLTA which incorporates both gender and location attackers. Moreover, results show that both RLTA-GEN and RLTA-LOC have better utility than other baselines.

To sum-up, these results indicate that although using one private-attribute attacker in the training process can help in preserving more utility, it can compromise obscuring other private-attributes.

*Parameter Analysis:* Our proposed method RLTA has an important parameter $\alpha$ to change the level of privacy and utility. We illustrate the effect of this parameter by changing it as $\alpha \in \{0.0, 0.1, 0.25, 0.5, 0.75, 1.0\}$. According to the Figure 2, when the $\alpha$ parameter increases, the privacy loss will decrease, but, the utility loss will increase. This shows the utility and the privacy have an association with each other. Hence, the more privacy loss decreases, the utility loss increases. Choosing the right value for $\alpha$ depends on the application and usage of this method. According to the results, choosing $\alpha = 0.5$ would result in a balanced privacy-utility. In some applications where the privacy of users are important and critical, we can set the $\alpha$ parameter above $0.5$.

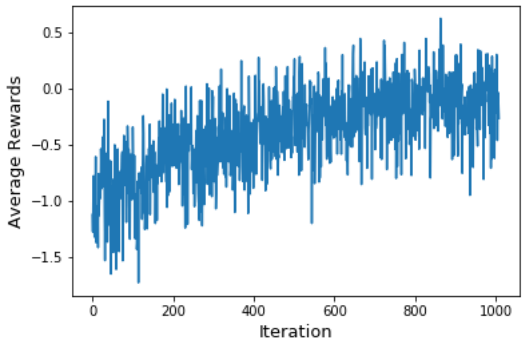


**Figure 3: The average of agent's rewards**

On the other hand, if the users privacy is not top priority, this parameter can be set to a lower value than $0.5$ which although it does not protect users' private attribute as good as when $\alpha >= 0.5$, but it does protect users' private attribute at a reasonable level.

### 5.4.2 Rewards Convergence

To evaluate the convergence of rewards, we consider agent's reward during training phase for each episode, shown in Figure. 3. The result indicates that agent's average reward is low at the beginning and then it increases afterward. This is because agent performs many random actions at the beginning to explore the action state space. We also observe that after several episodes, the reward converges to the baseline reward $\mathcal{B}$. This confirms that the agent has learned a proper action selection policy $\pi(s)$ to preserve both utility and privacy by satisfying the objectives of Eq. 7.

## 6 Conclusion

In this paper, we propose a deep reinforcement learning based text anonymization, RLTA, which creates a text embedding such that does not leak user's private-attribute information while preserving its utility w.r.t. a given task. RLTA has two main components: (1) an attention based task-aware text representation learner, and (2) a deep RL based privacy and utility preserver. Our results illustrate the effectiveness of RLTA in preserving privacy and utility. One future direction is to generate privacy preserving text rather than embeddings. We also adopt deep Q-learning to train the agent. A future direction is to apply different RL algorithms and investigate how it impacts results. It would be also interesting to adopt RLTA for other types of data.

## 7 Acknowledgements

The authors would like to thank Jundong Li for his help throughout the paper. This material is based upon the work supported, in part, by NSF 1614576, ARO W911NF-15-1-0328 and ONR N00014-17-1-2605.

## References

Balamurugan Anandan, Chris Clifton, Wei Jiang, Mummoorthy Murugesan, Pedro Pastrana-Camacho, and Luo Si. 2012. t-plausibility: Generalizing words to desensitize text. *Trans. Data Privacy*, 5(3):505–534.

Ghazaleh Beigi, Kai Shu, Ruocheng Guo, Suhang Wang, and Huan Liu. 2019a. I am not what i write: Privacy preserving text representation learning. *arXiv preprint arXiv:1907.03189*.

Ghazaleh Beigi, Kai Shu, Ruocheng Guo, Suhang Wang, and Huan Liu. 2019b. Privacy preserving text representation learning. *Proceedings of the 30th on Hypertext and Social Media (HT19). ACM*.

Valentina Beretta, Daniele Maccagnola, Timothy Cribbin, and Enza Messina. 2015. An interactive method for inferring demographic attributes in twitter. In *ACM Conference on Hypertext & Social Media*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Maximin Coavoux, Shashi Narayan, and Shay B Cohen. 2018. Privacy-preserving neural representations of text. *arXiv preprint arXiv:1808.09408*.

George Cybenko. 1989. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2017. Calibrating noise to sensitivity in private data analysis. *Journal of Privacy and Confidentiality*, 7(3):17–51.

Briland Hitaj, Giuseppe Ateniese, and Fernando Pérez-Cruz. 2017. Deep models under the gan: information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 603–618. ACM.

Dirk Hovy, Anders Johannsen, and Anders Søgaard. 2015. User review sites as a resource for large-scale sociolinguistic studies. In *Proceedings of the 24th International Conference on World Wide Web*.

Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. 1996. Reinforcement learning: A survey. *Journal of artificial intelligence research*.

Man Lan, Sam-Yuan Sung, Hwee-Boon Low, and Chew-Lim Tan. 2005. A comparative study on term weighting schemes for text categorization. In *IEEE International Joint Conference on Neural Networks*.

Yitong Li, Timothy Baldwin, and Trevor Cohn. 2018. Towards robust and privacy-preserving text representations. In *The 56th Annual Meeting of the Association for Computational Linguistics*.

Marco Lui and Timothy Baldwin. 2012. langid. py: An off-the-shelf language identification tool. In *Proceedings of the ACL 2012 system demonstrations*.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Arjun Mukherjee and Bing Liu. 2010. Improving gender classification of blog authors. In *Empirical Methods in natural Language Processing (EMNLP)*.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*.

Greg Pass, Abdur Chowdhury, and Cayley Torgeson. 2006. A picture of search. In *1st international conference on Scalable information systems (InfoScale)*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of conference on Empirical Methods in natural Language Processing (EMNLP)*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

Yücel Saygin, Dilek Hakkini-Tur, and Gökhan Tur. 2006. Sanitization and anonymization of document repositories. In *Web and information security*.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*.

Zhan Shi, Xinchi Chen, Xipeng Qiu, and Xuanjing Huang. 2018. Toward diverse text generation with inverse reinforcement learning. In *27th International Joint Conference on Artificial Intelligence*.

Yueming Sun and Yi Zhang. 2018. Conversational recommender system. In *ACM SIGIR Conference on Research & Development in Information Retrieval*.

Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double q-learning. In *30th AAAI Conference*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Svitlana Volkova, Yoram Bachrach, Michael Armstrong, and Vijay Sharma. 2015. Inferring latent user properties from texts published in social media. In *29th AAAI Conference on Artificial Intelligence*.

Reza Zafarani, Mohammad Ali Abbasi, and Huan Liu. 2014. *Social media mining: an introduction*. Cambridge University Press.

Jinxue Zhang, Jingchao Sun, Rui Zhang, Yanchao Zhang, and Xia Hu. 2018. Privacy-preserving social media data outsourcing. In *IEEE INFOCOM Conference on Computer Communications*.

# Tree-structured Decoding for Solving Math Word Problems

**Qianying Liu**[*1], **Wenyu Guan**[*23], **Sujian Li**[2] **and Daisuke Kawahara**[1]
[1] Graduate School of Informatics, Kyoto University
[2] Key Laboratory of Computational Linguistics, MOE, Peking University
[3] School of Software and Microelectronics, Peking University
ying@nlp.ist.i.kyoto-u.ac.jp; guanwy@pku.edu.cn;
lisujian@pku.edu.cn; dk@i.kyoto-u.ac.jp

## Abstract

Automatically solving math word problems is an interesting research topic that needs to bridge natural language descriptions and formal math equations. Previous studies introduced end-to-end neural network methods, but these approaches did not efficiently consider an important characteristic of the equation, i.e., an abstract syntax tree. To address this problem, we propose a tree-structured decoding method that generates the abstract syntax tree of the equation in a top-down manner. In addition, our approach can automatically stop during decoding without a redundant stop token. The experimental results show that our method achieves single model state-of-the-art performance on Math23K, which is the largest dataset on this task.

## 1 Introduction

Math word problem (MWP) solving is the task of obtaining a solution from math problems that are described in natural language. One example of MWP is illustrated in Figure 1. Early approaches rely on either predefined rules or statistical machine learning-based methods to map the problems into several predefined templates in classification style or retrieval style. The major drawback of these approaches is that they are inflexible for new templates and require extra effort to design rules, features and templates.

Modeling the tree structure of math equations has been considered as an important factor when building models for MWP. As shown in Figure 1, each equation could be transformed into an abstract syntax tree (AST). Roy and Roth (2017) built an expression tree and combined two classifiers for quantity relevance prediction and operator classification. Koncel-Kedziorski et al. (2015) designed a template ranking function based on the AST of the equations. However, these approaches are based on traditional methods and require feature engineering.

| | |
|---|---|
| Problem | The distance between the two places A and B is 660 km, the car starting from A drives 32 km/h, and the car starting from B drives 34 km/h. The two cars are starting from the two places at the same time in inverse direction. How many hours later would the two cars meet? |
| Equation | x = 660 / ( 32 + 34 ) |
| Prefix | x = / 660 + 32 34 |
| Answer | 10 |
| AST | 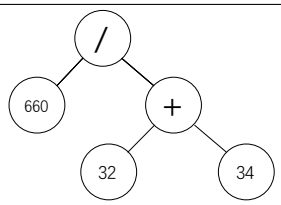 |



Figure 1: One example of MWP. Problem refers to the natural language descriptions. Equation refers to the formal math equation. Prefix refers to the prefix notation of the equation. Answer refers to the final quantity solution. AST refers to the AST of the equation.

Recently, the appearance of large-scale datasets and the development of neural generative models have opened a new research line for MWP. Wang et al. (2017) cast this task as a sequence generation problem and used a sequence-to-sequence (seq2seq) model to learn the mapping from natural language text to a math equation. Recent approaches use the Reverse Polish notation, also called suffix notation of equations in which operators follow their operands to implicitly model the tree structure (Wang et al., 2018a; Chiang and Chen, 2018). However, these studies lost sight of important information of the math equation ASTs

[*] This denotes equal contribution.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2370–2379, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

(e.g., parent and siblings of each node), despite of their promising results. Thus, their model has to use extra effort to memorize various pieces of auxiliary information such as the sibling node of the current step and learn the implicit tree structure in the sequence.

Meanwhile, in similar tasks such as semantic parsing and code generation, which also try to convert natural language into a well-formed executable tree structured output, they first decode the function or operator at the root node before decoding the variables. Such top-down decoding matches with the Polish notation, also called the prefix order of the AST of math equations, shown in Figure 1. Human also does reasoning and reference in a similar order, usually first determines the function before filling in the variables, but not the inverse. Thus we present a top-down hierarchical sequence-to-tree (seq2tree) model which explicitly models the tree structure. This decoder considers the prefix order of the equation and feeds the information of the parent and sibling nodes during decoding.

Another advantage of this system is that it can use a stack to monitor the decoding process and automatically end. By pushing in new generated nodes to the stack and popping out the completed subtrees, the decoding process can naturally end when the tree is completed and the stack is empty. There is no need for a special redundant end-of-sequence (EOS) token which is used in ordinary text sequence generation. Without the EOS token, the model has a higher possibility to generate valid equation answers.

In summary, the contributions of this work are as follows:

1. We design a hierarchical seq2tree model that can better capture information of the AST of the equation.

2. We are the first to use prefix order decoding for MWP.

3. We abandon the EOS token for end-to-end equation generation.

4. Our model outperforms previous systems for solving math word problems. On the large-scale dataset Math23K, we achieve state-of-the-art single model performance.

## 2 Related Work

Our work synthesizes two strands of research, which are math word problems and seq2tree encoder-decoder architectures.

### 2.1 Math Word Problems

Early approaches hand engineered rule-based systems to solve MWPs (Mukherjee and Garain, 2008). The rules could only cover a limited domain of problems, while math word problems are flexible in real-world settings.

There are currently three major research lines in solving MWPs. The first research line maps a problem text into logical forms, and then uses the logical forms to obtain the equation (Shi et al., 2015; Roy and Roth, 2015; Huang et al., 2017; Roy and Roth, 2018). Shi et al. (2015) defined a Dolphin language to connect math word problems and logical forms. The major drawback is that it requires extra human annotation for the logic forms.

Another research line uses either a retrieval or classification model to maintain a template, and then fills in the slots with quantities. Kushman et al. (2014) first introduced the idea of 'equation template'. For example, x = 6*7 and x = 10*5 belong to the same template x = $n_1$*$n_2$. They collected the first dataset of this task, ALG514, which contained 514 samples. They brought out a two-step pipeline model, which first used a classifier to select a template and then mapped the numbers into the slots. One major drawback is that it cannot solve problems beyond the templates in the training data. This two-step pipeline model was further extended with tree-based features, ranking style retrieval models and so on (Upadhyay and Chang, 2017; Roy and Roth, 2017). Huang et al. (2016) released the first large-scale dataset Dophin18K and trained a similar system on it.

The third research line directly generates the equation. Hosseini et al. (2014) cast the problem into a State Transition Diagram of verbs and trained a binary classifier that could solve problems with only add and minus operators. Wang et al. (2017) first used a seq2seq model to directly generate the equation template and released a Chinese high-quality large-scale dataset, Math23K. Reinforcement learning was used to further improve the seq2seq framework. Wang et al. (2018b) first extended the seq2seq model by decoding the suffix order sequence of the equations. Wang et al. (2018a) introduced equation normalization techniques that leverage the duplicated template problem. Chiang and Chen (2018) used the copy

mechanism to improve the semantic representation of quantities. However, in these work the model needs extra effort to learn the implicit tree structure and memorize tree information in the sequence. Wang et al. (2019) proposed a two-step pipeline method that first generates a template with unknown operators, and then uses a recursive neural network to combine tree structure information and predict the operators. However, the topology of the AST is determined in the first step without tree structure information. To our best knowledge, we are the first to explicitly give the model guidance of parent and sibling nodes.

### 2.2 Seq2Tree Architectures

Seq2Tree-style encoder-decoder is mainly used in two fields both of which try to bridge natural language and a tree structured output.

Semantic parsing is the task that translates natural language text to formal meaning logical forms or structured queries. Code generation maps a piece of program description to programming language source code. Dong and Lapata (2016) first used recurrent neural networks (RNNs) based seq2tree for semantic parsing and out performed the seq2seq model. One drawback is that their generation is at token level so it cannot guarantee the result is syntactically correct. Grammar rules were used to solve this problem. Another drawback is that they needed special tokens for predicting branches, which are not necessary for MWPs because all operators are binary operators. The similar framework is also used in code generation (Zhang et al., 2016; Yin and Neubig, 2017). Alvarez-Melis and Jaakkola (2017) presented doubly recurrent neural networks to predict tree topology explicitly. Rabinovich et al. (2017) presented a abstract syntax network that combines edge information for code generation. Convolution neural networks (CNNs) were used for code generation decoding because the output program is much longer than semantic parsing and MWPs, and RNNs suffer from the long dependency problem (Sun et al., 2018).

## 3 Model

Our model consists of two stages as shown in Figure 2: the encoder stage that encodes the input natural language into a sequence of representation vectors and the decoder stage that receives these vectors and decodes the AST of the equation with the help of an auxiliary stack.

| | |
|---|---|
| Equation | x = 130 * ( 1 - 0.8 ) |
| Quantities | $n_1$: 130, $n_2$: 0.8 |
| Template | $n_1$ * ( 1 - $n_2$ ) |
| Prefix template | * $n_1$ - 1 $n_2$ |

Table 1: One example of prefix template

### 3.1 Preprocessing

**Significant Number Identification** A Significant Number Identification (SNI) unit is used for reducing the noise in the input numbers. Significance refers to whether the number appears in the equation. In MWPs, it is very common that the input text contains irrelevant numbers, such as date or descriptive text such as 'third grade student'. We follow Wang et al. (2017) and simply use whether the numbers appear in the equation as gold labels and build an LSTM-based binary classification model to determine the significance of the input numbers. The accuracy of this unit is 99% and thus it can efficiently reduce the noise.

**Prefix Order Equation Template** For the output equations, we first turn them into prefix order equation templates before using them to train the model. In Table 1 we show one example of prefix templates. Given a problem-equation pair, we first build the mapping between numbers in the problem and equation, $n_i$ denotes the ith number in the problem after SNI. We use this mapping to convert the equation into a template by replacing the numbers with $n_i$ tokens, and at last convert the template into prefix order.

To be noticed, equations can be mapped to more than one AST. For example, $n1 + n2 + n3$ could be mapped to an AST with either the first $+$ or the second $+$ as the root node, and in that case the prefix order notation would also be different. We assume the first operator is the root node of the AST here. Further details are shown in the appendix.

**Equation Normalization** One math word problem can be solved by different but equivalent equations, which bring noise during training. For example, 10 - (8 + 5) and 10 - 8 - 5 are equivalent, but the templates $n_1 - (n_2 + n_3)$ and $n_1 - n_2 - n_3$ are different. This problem is called the equation duplication problem. We follow Wang et al. (2018a) and use several rules for equation normalization

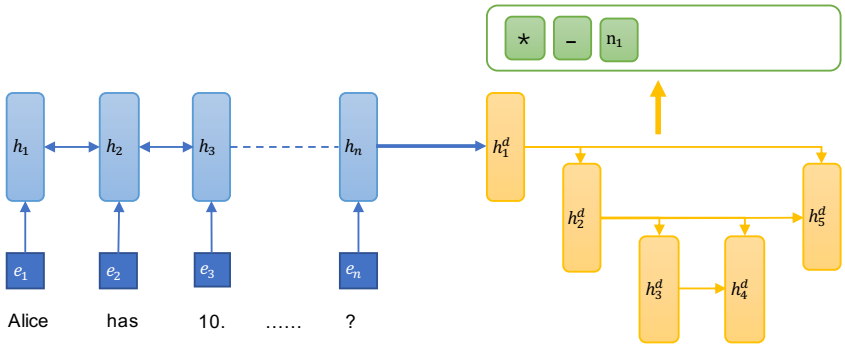


Figure 2: Framework of our seq2tree model. The blue blocks refer to the encoder. The yellow blocks refer to the decoder. The green blocks refer to the auxiliary stack.

to alleviate the equation duplication problem. The rules are listed below.

- If one long equation template could be converted into a shorter one, then it should be shortened. For example, $n_1+n_2+n_3+n_3-n_3$ and $n_1+n_2+n_3$ are equivalent. In this case the former one should be normalized as the latter one.
- The order of number tokens in the template should follow their occurrence order in the problem text as much as possible. For example, $n_1+n_3+n_2$ should be normalized as $n_1+n_2+n_3$.

### 3.2 Encoder

Bi-directional Long Short Term Memory Network (BiLSTM) is an efficient method to encode sequential information. Formally, given an input math word problem sentence $x = \{x_t\}_{t=1}^{n}$ , we first embed each word into a vector $e_t$. Then these embeddings are fed into a BiLSTM layer to model the sequential information.

$$h_t = (LSTM(h_{t-1}^f, e_t); LSTM(h_{t-1}^b, e_t)), \quad (1)$$

where $h_t$ is the concatenation of the hidden states $h_t^f$ and $h_t^b$, which are from both forward and backward LSTMs. These representation vectors are then fed into the following decoder stage.

### 3.3 Tree-structured Decoder

For decoding, we follow Dong and Lapata (2016) and build a top-down hierarchical tree-structured decoder. However, their model is built for semantic parsing and some components are unnecessary and redundant for equation template decoding. Benefited by the fact that operator nodes must have two children and number nodes must be leaf nodes, we build a decoder which is specialized for math equation AST, as shown in Figure 3. It extends a vanilla sequence-based LSTM decoder by using tree-based information feeding as the input, and also an auxiliary stack to help the model know which is the next token to decode and automatically end the decoding process without a special token.

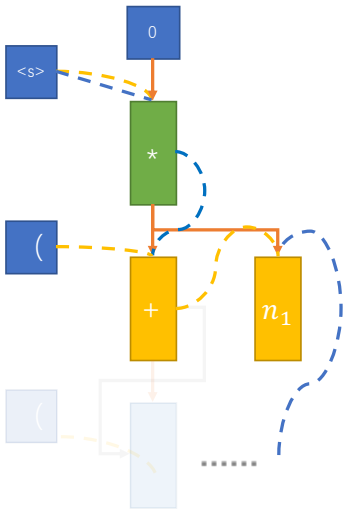


Figure 3: Framework of the tree structured decoder. The yellow dotted line refers to sibling feeding. The blue dotted line refers to previous token feeding. The orange solid line refers to parent feeding. The dotted line means it uses embedding information, while the solid line means it uses the hidden state information.

#### 3.3.1 Tree-based Information Feeding

The input of each time step consists of three parts: parent feeding, sibling feeding and previous token feeding.

The parent feeding $h^{parent}$ refers to using the LSTM hidden state of the parent node as the input when decoding children nodes, which is shown as

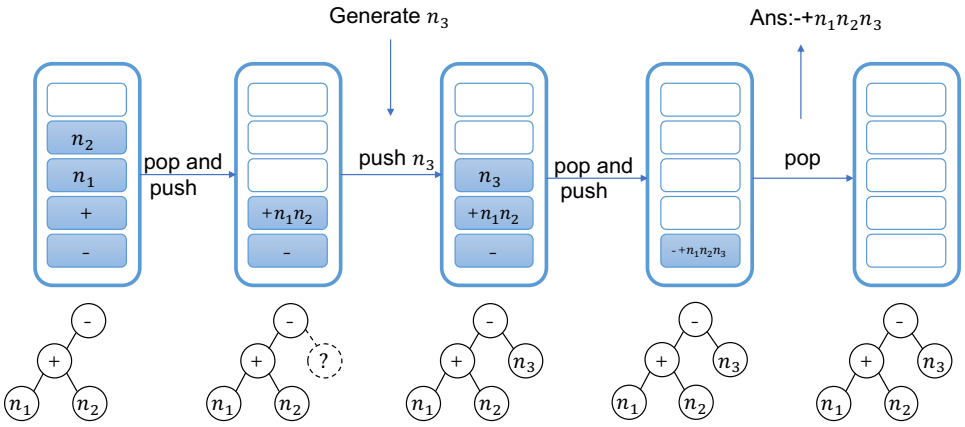


Figure 4: One example of tree-structured decoding with the auxiliary stack.

the orange solid line in Figure 3. This can let the model be informed of the parent node status. For the root node, this part of the input is padded as zeros.

The sibling feeding $e^{sibling}$ refers to using the embedding of the left sibling node as the input when decoding the right sibling node, which is shown as the yellow dotted line in Figure 3. This can let the model be informed whether we are decoding the left or right sibling. For the root node, we use a special token $\langle s \rangle$ for sibling feeding. For the left-most node, we use a special token ( for sibling feeding.

The previous token feeding $e^{prev}$ refers to using the previous token in prefix order as the input when decoding the next token, which is shown as the blue dot line in Figure 3. This can let the model be informed of what part is already decoded by the tree. For the root node, we also use the special token $\langle s \rangle$ for previous token feeding.

At time step $t$, the input $e_t^d$ of the LSTM unit is the concatenation of these three components.

$$e_t^d = (h^{parent}; e^{sibling}; e^{prev}). \quad (2)$$

### 3.3.2 Tree-Structured LSTM

The tree-structured decoder uses LSTM to generate the equation template in a top-down manner, as the grey solid line in Figure 3, and use an auxiliary stack to guide the decoding process. Given the input $e_t^d$ shown in the previous section, we generate the output token $y_t$ with one LSTM layer and one Multi-layer Perceptron (MLP) layer.

$$h_t^d = LSTM(h_{t-1}^d, e_t^d) \quad (3)$$

$$y_t = argmax(MLP(h_t^d)) \quad (4)$$

**Algorithm 1:** The algorithm of using the auxiliary stack to guide tree decoding.

**Input:** Encoder Output $h_n$
**Output:** The Predicted Prefix Notation Equation $Equa^p = \{s_i^p\}_{i=1}^{n_2}$

initialize empty stack $S$ and list $Equa^p$;
**while** *$S.size!=1$ or $S.top$ is not quantity* **do**
  generate token $y_t$;
  $tmp$ = $S$.top;
  $S$.push($y_t$);
  **if** *$y_t$ is quantity* **then**
    **while** *$tmp$ is a quantity* **do**
      subtree $t$ = $S$.top[3];
      $S$.pop;$S$.pop;$S$.pop;
      $tmp$ = $S$.top;$S$.push($t$);
$Equa^p$ = $S$.pop;

As shown in Algorithm 1 and Figure 3, if the predicted token $y_t$ is an operator, then we predict the left child node of the operator and push this token into the stack $S$. If the predicted token $y_t$ is a quantity, we check the stack to determine which is the next token that needs to be decoded. If the top of the stack is an operator, then we push $y_t$ into the stack and go on to decode the right sibling node of the current node. If the top of $S$ is a quantity, we follow Algorithm 1 to find the next position to decode. We push $y_t$ to the stack and pop out the top three tokens, which should be $\langle op \rangle \langle num \rangle \langle num \rangle$. These three tokens form a subtree $t$ and we regard this subtree $t$ as one quantity unit in the following process. Then we examine the status of the stack again, if the top of the stack is still a number, we push $t$ back to the stack and continue until the top of the stack is not a quantity. When the loop stops,

if the top of the stack is an operator, we push back $t$ and continue to decode the operator's right child node, and if the stack is empty, the decoding process ends because the AST is completed. Here we still push back the tree unit to the stack and treat the status that only one number is in the stack as the ending condition, which refers to line 2 in Algorithm 1. In this way the condition that the first generated token is the answer number can be unified. With the help of this stack, we can guide the decoding process, including which token to generate next and when to stop naturally without any special tokens.

We show one example of the decoding process with the an auxiliary stack in Figure 4. The upper half part shows the status of the stack, where the solid blocks stand for the inserted tokens. The lower half part shows the status of the AST during decoding, where the solid line stands for the generated nodes and the dotted line stands for the node that should be generated in the next step. The decoder first generates $-$, $+$, $n_1$ and $n_2$ and forms a complete subtree in the AST. This subtree $+$, $n_1$ and $n_2$ is then popped out of the stack and $+n_1n_2$ are pushed back as one unit. The model then continues to predict the sibling node of the subtree's root node, which is the dotted line circle in Figure 4. The top three tokens of the stack now form a complete subtree again and is popped out of the stack. The stack is now empty, we push it back and the stack only contains one number unit, then the decoding process ends and the equation template is popped out.

#### 3.3.3 Attention Mechanism

An attention mechanism has shown its effectiveness in various natural language processing tasks. We extend the decoder with an attention mechanism by adjusting Equation 4. Instead of directly using the hidden state $h_t^d$ to predict the output token $y_t$, we consider relevant information from the input vectors to better predict $y_t$. Formally, given the LSTM hidden state $h_t^d$ and the encoder outputs $\{h_t\}_{t=1}^n$, we calculate the attention weights $\alpha_t^i$ and attention vector $s_t$ as follows:

$$s_t = \sum_{i=1}^{n} \alpha_t^i \cdot h_i = \sum_{i=1}^{n} \frac{exp(h_i \cdot h_t^d)}{\sum_{j=1}^{n} exp(h_j \cdot h_t^d)} \cdot h_i \quad (5)$$

In lieu of Equation 4, we use the attention vector $s_t$, which considers relevance of the encoder information to predict the output token $y_t$.

## 4 Experiments

To demonstrate the effectiveness of our model, we conduct experiments on the Math23K dataset. Our method achieves the state-of-the-art (SOTA) single model performance and also exceeds the previous ensemble model SOTA.

### 4.1 Dataset

**Math23K** Math23K is one large-scale Chinese MWP dataset that contains 23,162 math problems and math equation solutions. The questions are elementary school level. Every question is linear and contains only one unknown variable.

Although there are other large-scale datasets such as Dolphin18K and AQuA, which are in English, they either contain many small typos (e.g., using $x$ to represent $*$) or contain wrong answers and templates. Other datasets such as ALG514 and MAWPS are much more smaller. Therefore, we decide to conduct experiments on Math23K, which is the only large-scale, clean and high-quality dataset.

### 4.2 Implementation Details

The embedding vectors are pretrained on the training set with the word2vec algorithm. The dimension of the embedding is 128. We use a two-layer BiLSTM with the hidden size 512 for the encoder. The decoder is a two-layer LSTM with 1024 hidden size. We use a teacher forcing ratio of 0.83 during training. We use cross entropy as the loss function and Adam to optimize the parameters. We also use dropout to avoid over-fitting. The batch size is 128.

### 4.3 Results

Table 2 shows the results of our system and other novel systems of MWP on the Math23k test set. The retrieval-style models compare a question in the test set with the questions in the training set, choose the template that has the highest similarity, and then fill in the numbers into the template (Upadhyay and Chang, 2017; Robaidek et al., 2018). The classification-style models train a classifier to select an equation template, and then map the numbers into the template (Kushman et al., 2014). For retrieval and classification models, we use the results of Robaidek et al. (2018). The generation models use end-to-end style encoder-decoder systems to generate an equation template and then fill in the numbers.

| Model | | Acc |
|---|---|---|
| Retrieval | Cosine (Robaidek et al., 2018) | 23.8% |
| | Jaccard (Robaidek et al., 2018) | 47.2% |
| Classification | Self-Attention (Robaidek et al., 2018) | 56.8% |
| | Bi-LSTM (Robaidek et al., 2018) | 57.9% |
| Generation | DNS (Wang et al., 2017) | 58.1% |
| | BiLSTM+Suffix+EN (Wang et al., 2018a) | 66.7% |
| | Semantically-Aligned† (Chiang and Chen, 2018) | 65.8% |
| | T-RNN (Wang et al., 2019) | 66.9% |
| | **Ours** | **69.0%** |
| Ensemble | DNS+Retrieval (Wang et al., 2017) | 64.7% |
| | DNS+suffix+EN Ensemble (Wang et al., 2018a) | 68.4% |
| | T-RNN+Retrieval (Wang et al., 2019) | 68.7% |

Table 2: Math word problem solving accuracy on Math23K. † denotes that the result is 5-fold cross validation performance. All other models are tested on the test set.

| Model | Acc |
|---|---|
| Prefix Baseline | 66.8% |
| + Sibling Feeding | 67.8 % |
| + Parent Feeding | 68.1 % |
| Full Model | **69.5%** |

Table 3: Ablation study on Math23K by removing modules.

| Model | Invalid Templates |
|---|---|
| EOS as terminator | 1.3% |
| Stack as terminator | **0.2%** |

Table 4: Ablation study on Math23K by using different terminating methods.

Wang et al. (2017) proposed a DNS model, which used seq2seq with SNI to generate an equation template. Wang et al. (2018a) proposed a BiLSTM+Suffix+EN model, which extends the DNS model by decoding the suffix notation and applies equation normalization. Chiang and Chen (2018) introduced a copy mechanism to generate equation templates in a semantically-aligned manner. T-RNN (Wang et al., 2019) extends the BiLSTM+Suffix+EN model by first generating a template with unknown operators and then filling in the operators with a Recursive Neural Network. The ensemble models use bagging to combine the results of different models.

Our seq2tree model is also a generation-style model. As shown in Table 2, we achieve state-of-the-art single model performance on the test set, and even better results than all the previous ensemble models, which can demonstrate the effectiveness of our proposed method.

## 5 Analysis and Discussion

### 5.1 Ablation Study

To get better insight into our seq2tree system, we conduct ablation study on Math23K development set, which is shown in Table 3. The prefix baseline denotes the model that removes parent feeding and sibling feeding, but only uses previous token feeding for the input. Thus, this model loses parent and sibling information and falls into a linear seq2seq model based on the prefix notation. The prefix baseline performs competitive results compared to previous single model SOTA (66.9%), which proves the effectiveness of top-down decoding. Parent feeding and sibling feeding separately improve the baseline model by 1.1% and 1.0%, demonstrating the importance of informing the model of AST structure information.

In Table 4, we also report the percentage of invalid templates by using different terminating methods. We remove the auxiliary stack and use a special end-of-sentence (EOS) token, which was used in previous studies to terminate the decoding process (Wang et al., 2017, 2018a). We can see that using the stack as a terminator can let the model generate very low percentage of invalid templates and outperforms the EOS method.

| | |
|---|---|
| Problem | A person is taking a trip from A to B. He took a train for $n_1$ of the trip the first day. He took a bus and travelled for $n_2$ km the second day. He still needs to travel for $n_3$ of the total distance. How far is it from A to B? |
| Gold | **suffix order**: x = $n_1$ 1 $n_2$ - $n_3$ - / **prefix order**: x = / $n_1$ - - 1 $n_2$ $n_3$ |
| Prediction | **BiLSTM+Suffix+EN**: $n_1$ $n_2$ - $n_3$ / (error) **Ours**: / $n_1$ - - 1 $n_2$ $n_3$ (correct) |

Figure 5: One example of our system compared with BiLSTM+Suffix+EN (Wang et al., 2018a).

| #Operators | Proportion(%) | Acc(%) |
|---|---|---|
| 0 | 0.1 | 100.0 |
| 1 | 17.3 | 82.7 |
| 2 | 52.2 | 74.5 |
| 3 | 19.1 | 59.9 |
| 4 | 6.6 | 42.4 |
| 5 | 3.4 | 44.1 |
| 6 | 0.9 | 55.6 |
| 7 | 0.3 | 0 |
| 8 | 0 | 0 |
| 9 | 0.1 | 100.0 |

Table 5: Accuracy of different template lengths on Math23K.

| Domain | Proportion(%) | Acc(%) |
|---|---|---|
| Distance & Speed | 20.5 | 70.2 |
| Tracing | 27.0 | 74.1 |
| Engineering | 5.4 | 64.1 |
| Interval | 0.2 | 50.0 |
| Circle Geometry | 1.5 | 33.3 |
| Plane Geometry | 1.1 | 81.8 |
| Profit | 0.5 | 40.0 |
| Solid Geometry | 1.3 | 46.2 |
| Interest Rate | 0.5 | 80.0 |
| Production | 0.4 | 100.0 |

Table 6: Accuracy of different question domains on Math23K.

### 5.2 Case Study

Here we give an example that is improved by our tree-structured decoding system. As shown in Figure 5, in the gold equation of this example, there is a long distance between two pairs of parent-child nodes $n_1$ and $/$ and also 1 and $-$. The BiLSTM+Suffix+EN model failed to capture the relationship between these two pairs of parent-child nodes and caused an error. Our model has a better ability to capture the relation between pairs of parent-child nodes even if there is a long distance between them in the notation.

### 5.3 Error Analysis

In Table 5, we show the results of how the accuracy changes as the template becomes longer. We can see that our model's performance becomes very low when the equation becomes longer. This shows that our model has limitations at predicting long templates. This is because longer templates often match with more complex questions which are more difficult to solve. Thus, this model still has room for improvement on reasoning, inference and semantic understanding. Meanwhile, only a few examples in Math23K match with complex templates, so introducing data augmentation techniques or constructing a new dataset with more complex examples may further improve the model's performance.

In Table 6, we examine the performance of the model in different question domains. MWPs in the same domain usually share similar logic, while there is an obvious difference between questions across different domains. Accurately detecting the question domain is very laborious, so we do this experiment by simply detecting frequent keywords of each domain in the question. We show further details in the appendix. The results show that the performance of the model has obvious variance among different domains and limitations in some domains such as solid geometry. This is because these domains require complicated external knowledge for solving these questions, such as $S_{circle} = \pi r^2$. It is difficult for the model to automatically summarize these kinds of information with only supervision of the equation templates. Adding external knowledge for this task may further improve the model.

## 6 Conclusion

We proposed a sequence-to-tree generative model to improve template generation for solving math word problems. The hierarchical top-down tree-structure decoder can use the information of the abstract syntax tree of an equation during decoding. With the help of an auxiliary stack, this

decoding process can end without any redundant special tokens. Our model achieves state-of-the-art results on the large-scale dataset, Math23k, demonstrating the effectiveness of our approach.

## Acknowledgement

This work was supported by JSPS KAKENHI Grant Number JP18H03286.

## References

David Alvarez-Melis and Tommi S. Jaakkola. 2017. Tree-structured decoding with doubly-recurrent neural networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Ting-Rui Chiang and Yun-Nung Chen. 2018. Semantically-aligned equation generation for solving and reasoning math word problems. *arXiv preprint arXiv:1811.00720*.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 33–43.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–533.

Danqing Huang, Shuming Shi, Chin-Yew Lin, and Jian Yin. 2017. Learning fine-grained expressions to solve math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 805–814.

Danqing Huang, Shuming Shi, Chin-Yew Lin, Jian Yin, and Wei-Ying Ma. 2016. How well do computers solve math word problems? large-scale dataset construction and evaluation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 887–896.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 271–281.

Anirban Mukherjee and Utpal Garain. 2008. A review of methods for automatic understanding of natural language mathematical problems. *Artificial Intelligence Review*, 29(2):93–122.

Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract syntax networks for code generation and semantic parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1139–1149.

Benjamin Robaidek, Rik Koncel-Kedziorski, and Hannaneh Hajishirzi. 2018. Data-driven methods for solving algebra word problems. *arXiv preprint arXiv:1804.10718*.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752.

Subhro Roy and Dan Roth. 2017. Unit dependency graph and its application to arithmetic word problem solving. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Subhro Roy and Dan Roth. 2018. Mapping to declarative knowledge for word problem solving. *Transactions of the Association of Computational Linguistics*, 6:159–172.

Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu, and Yong Rui. 2015. Automatically solving number word problems by semantic parsing and reasoning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1132–1142.

Zeyu Sun, Qihao Zhu, Lili Mou, Yingfei Xiong, Ge Li, and Lu Zhang. 2018. A grammar-based structural cnn decoder for code generation. *arXiv preprint arXiv:1811.06837*.

Shyam Upadhyay and Ming-Wei Chang. 2017. Annotating derivations: A new evaluation strategy and dataset for algebra word problems. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 494–504.

Lei Wang, Yan Wang, Deng Cai, Dongxiang Zhang, and Xiaojiang Liu. 2018a. Translating a math word problem to a expression tree. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1064–1069.

Lei Wang, Dongxiang Zhang, Lianli Gao, Jingkuan Song, Long Guo, and Heng Tao Shen. 2018b. Mathdqn: Solving arithmetic word problems via deep reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Lei Wang, Dongxiang Zhang, Jipeng Zhang, Xing Xu, Lianli Gao, Bingtian Dai, and Heng Tao Shen. 2019. Template-based math word problem solvers with recursive neural networks.

Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017. Deep neural solver for math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 845–854.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 440–450.

Xingxing Zhang, Liang Lu, and Mirella Lapata. 2016. Top-down tree long short-term memory networks. In *Proceedings of NAACL-HLT*, pages 310–320.

## A Prefix Notation Algorithm

**Algorithm 2:** The algorithm of converting ordinary equations to its prefix notation.

**Input:** The Ordinary Equation String $Equa = \{s_i\}_{i=1}^{n_1}$

**Output:** The Prefix Notation Equation String $Equa^p = \{s_i^p\}_{i=1}^{n_2}$

initialize empty stack $S$ and list $Equa_{inv}^p$;
**for** $i \leftarrow n_1$ ***to*** $1$ **do**
  **if** $s_i$ *is quantity* **then**
    $Equa_{inv}^p$.append($s_i$);
  **else if** $s_i$ *is* ) **then**
    $S$.push($s_i$);
  **else if** $s_i$ *is operator* **then**
    **while** *True* **do**
      **if** $s_i$ *is prior to* $S$.*top or* $S$.*top is* ) *or* $S$ *is empty* **then**
        $S$.push($s_i$);
        break;
      **else**
        $Equa_{inv}^p$.append($S$.top);
        $S$.pop();
  **else if** $S$.*top is (* **then**
    **while** $S$.*top is not )* **do**
      $Equa_{inv}^p$.append($S$.top);
      $S$.pop();
    $S$.pop()
$Equa^p$ = $Equa_{inv}^p$.inverse()

Here we show the details of the converting algorithm. The input is the ordinary equation string and the output is the prefix notation of the equation.

## B Domain Keywords

We show the table of the keywords for each domain. These keywords were manually collected by observation.

| Domain | Keywords |
|---|---|
| Distance & Speed | 速 度,千 米,路 程,相距,全程 |
| Tracing | 相 遇,相 对,相 反,相背,相向 |
| Engineering | 工程,零件,工程队,公路,修路 |
| Interval | 利润 |
| Circle | 半径,圆,直径,周长 |
| Plane Geometry | 三角形,正方形,长方形,边长 |
| Profit | 间隔,隔 |
| Solid Geometry | 体 积,侧 面 积,横 截面,表面积,圆柱,长方体 |
| Interest Rate | 利息,利率 |
| Production | 超产,减产 |

Table 7: The list of keywords in the domain specific studies.

# PullNet: Open Domain Question Answering with Iterative Retrieval on Knowledge Bases and Text

**Haitian Sun  Tania Bedrax-Weiss  William W. Cohen**
Google Research
{haitiansun,tbedrax,wcohen}@google.com

## Abstract

We consider open-domain question answering (QA) where answers are drawn from either a corpus, a knowledge base (KB), or a combination of both of these. We focus on a setting in which a corpus is supplemented with a large but incomplete KB, and on questions that require non-trivial (e.g., "multi-hop") reasoning. We describe PullNet, an integrated framework for (1) learning what to retrieve and (2) reasoning with this heterogeneous information to find the best answer. PullNet uses an iterative process to construct a question-specific subgraph that contains information relevant to the question. In each iteration, a graph convolutional network (graph CNN) is used to identify subgraph nodes that should be expanded using retrieval (or "pull") operations on the corpus and/or KB. After the subgraph is complete, another graph CNN is used to extract the answer from the subgraph. This retrieve-and-reason process allows us to answer multi-hop questions using large KBs and corpora. PullNet is weakly supervised, requiring question-answer pairs but not gold inference paths. Experimentally PullNet improves over the prior state-of-the art, and in the setting where a corpus is used with incomplete KB these improvements are often dramatic. PullNet is also often superior to prior systems in a KB-only setting or a text-only setting.

## 1 Introduction

Open domain Question Answering (QA) is the task of finding answers to questions posed in natural language, usually using text from a corpus (Dhingra et al., 2017; Joshi et al., 2017; Dunn et al., 2017), or triples from a knowledge base (KB) (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Yih et al., 2015). Both of these approaches have limitations. Even the largest KBs are incomplete (Min et al., 2013), which limits recall of a KB-based QA system. On the other hand, while a large corpus may contain more answers than a KB, the diversity of natural language makes corpus-based QA difficult (Chen et al., 2017; Welbl et al., 2018; Kwiatkowski et al., 2019; Yang et al., 2018).

In this paper we follow previous research (Sawant et al., 2019; Sun et al., 2018) in deriving answers using both a corpus and a KB. We focus on tasks in which questions require compositional (sometimes called "multi-hop") reasoning, and a setting in which the KB is incomplete, and hence must be supplemented with information extracted from text. We also restrict ourselves in this paper to answers which correspond to KB entities. For this setting, we propose an integrated framework for (1) *learning what to retrieve*, from either a corpus, a KB, or a combination, and (2) *combining this heterogeneous information* into a single data structure that allows the system to reason and find the best answer. In prior work, this approach was termed an *early fusion* approach, and shown to improve over *late fusion* methods, in which two QA systems, one corpus-based and one KB-based, are combined in an ensemble.

Our system, PullNet, builds on the GRAFT-Net[1] (Sun et al., 2018) early fusion system. GRAFT-Net uses heuristics to build a question-specific subgraph which contains sentences from the corpus, and entities and facts from the KB. A graph CNN (Kipf and Welling, 2016; Li et al., 2016; Schlichtkrull et al., 2017) variant is then used to reason over this graph and select an answer. However, as we will show experimentally, GRAFT-Net's heuristics often produce subgraphs that are far from optimal–often they are much larger than necessary, and sometimes do not contain the answer.

Like GRAFT-Net, PullNet also uses a reason-

[1] Graphs of Relations Among Facts and Text Networks.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2380–2390, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

ing process based on a graph CNN to find answers. However, PullNet *learns* how to construct the subgraph, rather than using an *ad hoc* subgraph-building strategy. More specifically, PullNet relies on a small set of retrieval operations, each of which expands a graph node by retrieving new information from the KB or the corpus. PullNet learns when and where to apply these "pull" operations with another graph CNN classifier. The "pull" classifier is weakly supervised, using question-answer pairs.

The end result is a learned iterative process for subgraph construction, which begins with a small subgraph containing only the question text and the entities which it contains, and gradually expands the subgraph to contain information from the KB and corpus that are likely to be useful. The incremental question-guided subgraph construction process results in high-recall subgraphs that are much smaller than the ones created heuristically, making the final answer extraction process easier. The process is especially effective for multi-hop questions, which naively would require expanding the subgraph to include all corpus and KB elements that are $k$ hops away from the question.

PullNet improves over the current state-of-the-art for KB-only QA on several benchmark datasets, and is superior to, or competitive with, corpus-only QA on several others. For multi-hop questions, this improvement is often dramatic. For instance, MetaQA (Zhang et al., 2018) contains multi-hop questions based on a small movie KB, originally associated with the WikiMovies dataset (Miller et al., 2016). In a KB-only setting, PullNet improves hits-at-one performance for 3-hop MetaQA questions from 62.5% to 91.4%. Perhaps more interestingly, PullNet obtains performance of 85.2% hits-at-one with a KB from which half of the triples have been removed, if that KB is supplemented with a corpus. We note that this result improves by 7% (absolute improvement) over a pure corpus-based QA system, and by more than 25% over a pure KB-based QA system. In a similar incomplete-KB setting, PullNet improves over GRAFT-Net by 6.8% on the ComplexWebQuestions dataset (Talmor and Berant, 2018).

## 2 Related Work

This paper has focused on QA for multi-hop questions using large KBs and text corpora as the information sources from which answers can be drawn. The main technical contribution is an iterative question-guided retrieval mechanism that retrieves information from KBs, corpora, or combinations of both, which makes it possible to follow long paths of reasoning on large KB.

A long line of QA models have been developed which answer questions based on a single passage of text (Dhingra et al., 2016; Yu et al., 2018; Seo et al., 2016; Gao et al., 2018; Liu et al., 2017; Devlin et al., 2018). Generally, these "reading comprehension" systems are operated by encoding the passage and question into an embedding space, and due to memory limitations cannot be applied to a large corpus instead of a short passage. To address this limitation a number of systems have been designed which use a "retrieve and read" pipeline (Chen et al., 2017; Dhingra et al., 2017; Joshi et al., 2017; Dunn et al., 2017; Wang et al., 2018, 2017), in which a retrieval system with high recall is piped into a reading comprehension system that can find the answer. An alternative approach is "phrase-indexed" QA (Seo et al., 2018), where embedded phrases in a document are indexed and searched over. Existing systems are also not able to use both KB and text for QA. They also differ from PullNet in using only a single round of retrieval; however, for questions that require multi-hop reasoning, it is difficult for a single retrieval step to find the relevant information.

SplitQA (Talmor and Berant, 2018) is a text-based QA system that decomposes complex questions (e.g., with conjunction or composition) into simple subquestions, and performs retrieval sequentially for the subquestions. Although it uses iterative retrieval for multi-hop questions, unlike PullNet, SplitQA does not also use a KB as an information source. Also, SplitQA has been applied only to the Complex WebQuestions dataset, so it is unclear how general this approach is.

There has also been much work on QA from KBs alone, often using methods based on memory networks (Sukhbaatar et al., 2015), semantic parsing (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005) or reinforcement learning Das et al. (2017). Extending such KB-based work to also use text, however, is non-trivial. Another line of QA work from text and KBs is exemplified by AQQU (Bast and Haussmann, 2015) and its successors (Sawant et al., 2019). These systems focus on questions (e.g. SimpleWebQuestions), that can

be interpreted as identifying an entity based on a relationship and related entity, plus additional restrictions described in text, and it is unclear how to extend such approaches to multi-hop questions.

GRAFT-Net (Sun et al., 2018) supports multi-hop reasoning on both KBs and text by introducing a question subgraph built with facts and text, and uses a learned graph representation (Kipf and Welling, 2016; Li et al., 2016; Schlichtkrull et al., 2017; Scarselli et al., 2009) to perform the reasoning required to select the answer. We use the same representation and reasoning scheme as GRAFT-Net, but do not require that the entire graph be retrieved in a single step. In our experimental comparisons, this gives significant performance gains for multi-hop reasoning tasks.

Combinations of KBs and text have also been used for relation extraction and Knowledge Base Completion (KBC) (Lao et al., 2012; Toutanova et al., 2015; Das et al., 2017). The QA task differs from KBC in that in QA, the inference process must be conditioned on a natural-language question, which leads to different constraints on which methods can be used.

## 3 The PullNet Model

PullNet retrieves from two "knowledge sources", a text corpus and a KB. Given a question, PullNet will use these to construct a *question subgraph* that can be used to answer the question. The question subgraph is constructed iteratively. Initially the subgraph depends only on the question. PullNet then iteratively expands the subgraph by choosing nodes from which to "pull" information about, from the KB or corpus as appropriate. The question subgraph is heterogeneous, and contains both entities, KB triples, and entity-linked text.

In this section, we will first introduce notation defining the heterogeneous graph structure we use. Then we will introduce the general iterative retrieval process. Finally, we will discuss the retrieval operations used on the corpus and KB, and the classification operations on the graph which determine where to perform the retrievals.

### 3.1 The Question Subgraph

A *question subgraph* for question $q$, denoted $\mathcal{G}_q = \{\mathcal{V}, \mathcal{E}\}$, is a hetogeneous graph that contains information from both the text corpus and the KB relevant to $q$. Let $\mathcal{V}$ denote the set of vertices, which we also call nodes. Following GRAFT-Net (Sun et al., 2018), there are three types of nodes: entity nodes $\mathcal{V}_e$, text nodes $\mathcal{V}_d$, and fact nodes $\mathcal{V}_f$, with $\mathcal{V} = \mathcal{V}_e \cup \mathcal{V}_d \cup \mathcal{V}_f$. An entity node $v_e \in \mathcal{V}_e$ represents an entity from the knowledge base. A text node $v_d \in \mathcal{V}_d$ represents a document from the corpus, with a sequence of tokens denoted $(w_1, \ldots, w_{|d|})$. In this paper, a document is always a single sentence, to which an entity linker (Ji et al., 2014) has been applied to detect and ground entity mentions. A fact node $v_f \in \mathcal{V}_f$ represents a triplet $(v_s, r, v_o)$ from the KB, with subject and objects $v_s, v_o \in \mathcal{V}_e$ and relation $r$. Let $\mathcal{E}$ denote the set of edges between nodes. An edge connects a fact node $v_f$ and an entity node $v_e$ iff fact $v_f$ has $v_e$ as its subject or object. An edge connects a text node $v_d$ with entity node $v_e$ iff the entity is mentioned in the text.

### 3.2 Iterative subgraph construction

#### 3.2.1 Overview

We start with a question subgraph $\mathcal{G}_q^0 = \{\mathcal{V}^0, \mathcal{E}^0\}$ where $\mathcal{V}^0 = \{e_{q_i}\}$ is the list of entities in the question and $\mathcal{E}^0$ is an empty set. We iteratively expand the question subgraph $\mathcal{G}_q^0$ until it contains the information required to answer the question.

**Algorithm 1** PullNet

1: Initialize question graph $G_q^0$ with question $q$ and question entities, with $\mathcal{V}^0 = \{e_{q_i}\}$ and $\mathcal{E}^0 = \emptyset$.
2: **for** $t = 1, \cdots, T$ **do**
3: Classify the entity nodes in the graph and select those with probability larger than $\epsilon$
$\{v_{e_i}\} = \texttt{classify\_pullnodes}\,(G_q^t, k)$
4: **for all** $v_e$ in $\{v_{e_i}\}$ **do**
5: Perform pull operation on selected entity nodes
$\{v_{d_i}\} = \texttt{pull\_docs}\,(v_e, q)$
$\{v_{f_i}\} = \texttt{pull\_facts}\,(v_e, q)$
6: **for all** $v_d$ in $\{v_{d_i}\}$ **do**
7: Extract entities from new document nodes
$\{v_{e(d)_i}\} = \texttt{pull\_entities}\,(v_d)$
8: **for all** $v_f$ in $\{v_{f_i}\}$ **do**
9: Extract head and tail of new fact nodes
$\{v_{e(f)_i}\} = \texttt{pull\_headtail}\,(v_f)$
10: Add new nodes and edges to question graph
$G_q^{t+1} = \texttt{update}\,(G_q^t)$
11: Select entity node in final graph that is the best answer
$v_{\text{ans}} = \texttt{classify\_answer}\,(G_q^T)$

The algorithm is shown in Alg 1. Briefly, we expand the graph in $T$ iterations. In each iteration, we choose entities whose probability is larger than $\epsilon$ to expand, and then for each selected entity,

we retrieve a set of related documents, and also a set of related facts. The new documents are then passed through an entity-linking system to identify entities that occur in them, and the head and tail entities of each fact are also extracted. The last stage in each iteration is to update the question graph by adding all these new edges. After the $T$-th iteration of expansion, an additional classification step is applied to the final question subgraph to predict the answer entity.

#### 3.2.2 Pull Operations

Pull operations either retrieve information from a knowledge source, or extract entities from a fact or document.

The two extraction operations are relatively simple. The `pull_entities(`$v_d$`)` operation inputs a document node $v_d$, calls an entity linker and returns all entities mentioned in $v_d$. The `pull_headtail(`$v_f$`)` operation inputs a fact node $v_f$ and returns the subject and object entity of fact $v_f$.

The retrieval operations are more complex. The `pull_docs(`$v_e, q$`))` operation retrieves relevant documents from the corpus. We use an IDF-based retrieval system, Lucene (McCandless et al., 2010) and assume that all sentences have been entity-linked prior to being indexed. The retrieved documents are constrained to link to entity $v_e$, and are ranked by their IDF similarity to the question $q$. Only the top $N_d$ ranked documents are returned.

The `pull_facts(`$v_e, q$`)` operation retrieves the top $N_f$ facts from the KB about entity $v_e$. The retrieved facts are constrained to have $v_e$ as their subject or object, and are ranked based on the similarity $S(r, q)$ between the fact's relation $r$ and the question $q$. Since it is not obvious how to assess relevance of a fact to a question $q$, we learn $S(r, q)$ as follows. Let $h_r$ be an embedding of relation $r$, which is looked up from an embedding table, and let $q = (w_1, \ldots, w_{|q|})$ be the sequence of words for question $q$. Similarity is defined as the dot-product of the last-state LSTM representation for $q$ with the embedding for $r$. This dot-product is then passed through a sigmoid function to bring it into a range of $[0, 1]$: as we explain below, we will train this similarity function as a classifier which predicts which retrieved facts are relevant to the question $q$. The final ranking method for facts is

$$h_q = \text{LSTM}(w_1, \ldots, w_{|q|}) \in \mathbb{R}^n$$
$$S(r, q) = \text{sigmoid}(h_r^T \, h_q)$$

#### 3.2.3 Classify Operations

Two types of `classify` operations are applied to the nodes in a subgraph $G_q^t$. These operations are applied only to the entity nodes in the graph, but they are based on node representations computed by the graph CNN, so the non-entity nodes and edges also affect the classification results.

During subgraph construction, the `classify_pullnodes(`$G_q^t$`)` operation returns the probability entity node $v_e$ should be expanded in the next iteration. We choose the $k$ nodes with the highest probability in each iteration. After the subgraph is complete, the `classify_answer(`$G_q^t$`)` operation predicts whether an entity node answers the question. The highest-scoring entity node is returned as the final answer.

We use the same CNN architecture used by GRAFT-Net (Sun et al., 2018) for classification. GRAFT-Net supports node classification on heterogeneous graphs containing facts, entities, and documents. GRAFT-Net differs from other graph CNN implementations in using special mechanisms to distribute representations across different types of nodes and edges: notably, document nodes are represented with an LSTM encoding, extended by mechanisms that allow the representations for entity nodes $v_e$ to be passed into a document $v_d$ that mentions $v_e$, and mechanisms that allow the LSTM hidden states associated with an entity mention to be passed out of $v_d$ to the associated entity node $v_e$.

#### 3.2.4 The Update Operation

The `update` operation takes the question subgraph $G_q^{t-1}$ from the previous iteration and updates it by adding the newly retrieved entity nodes $\{v_{e(f)_i}\} \cup \{v_{e(d)_i}\}$, the text nodes $\{v_{d_i}\}$, and the fact nodes $\{v_{f_i}\}$. It also updates the set of edges $\mathcal{E}$ based on the definitions of Section 3.1. Note that some new edges are derived when pull operations are performed on text and fact nodes, but other new edges may connect newly-added nodes with nodes that already exist in the previous subgraph.

### 3.3 Training

To train PullNet, we assume that we only observe question and answer pairs, i.e., the actual inference chain required to answer the question is latent. We thus need to use weak supervision to train the classifiers described above.

To train these models, we form an approximation of the ideal question subgraph for question $q$ as follows. Note that in training, the answer entities are available. We use these to find all shortest paths in the KB between the question entities and answer entities. Each entity $e$ that appears in such a shortest path will be marked as a *candidate intermediate entities*. For each candidate intermediate entity $e$ we record its minimal distance $t_e$ from the question nodes.

When we train the `classify_pullnodes` classifier in iteration $t$, we treat as positive examples only those entities $e'$ that are connected to a candidate intermediate entity $e$ with distance $e_t = t+1$. Likewise in training the similarity function $S(h_r, q)$ we treat as positive relations leading to candidate intermediate entities $e$ at distance $e_t = t + 1$. This encourages the retrieval to focus on nodes that lie on shortest paths to an answer.

In training we use a variant of teacher forcing. We pull from all entity nodes with a predicted score larger than some threshold $\epsilon$, rather than only the top $k$ nodes. If, during training, a candidate intermediate entity is not retrieved in iteration $t_e$, we add it to the graph anyway. The values $T$ and $\epsilon$ are hyperparameters, but here we always pick for $T$ the maximum length of the inference chain needed to ensure full coverage of the answers.

We use the same classifier in the retrieval step as in answer selection, except that we change the last fully-connected layer. The classifiers used for retrieval in the different iterations are identical.

The learned parts of the model are implemented in PyTorch, using an ADAM optimizer (Kingma and Ba, 2014), and the full retrieval process of Alg 1 is performed on each minibatch.

## 4 Experiments and Results

### 4.1 Datasets

**MetaQA** (Zhang et al., 2018) contains more than 400k single and multi-hop (up to 3-hop) questions in the domain of movies. The questions were constructed using the knowledge base provided with the WikiMovies (Miller et al., 2016) dataset. We use the "vanilla" version of the queries[2] We use the KB and text corpus supplied with the WikiMovies dataset, and use exact match on surface forms to perform entity linking. The KB used here is relatively small, with about 43k entities and 135k triples.

**WebQuestionsSP** (Yih et al., 2015) contains 4737 natural language questions that are answerable using Freebase.[3] The questions require up to 2-hop reasoning from knowledge base, and 1-hop reasoning using the corpus. We use Freebase as our knowledge base but for ease of experimentation restrict it to a subset of Freebase which contains all facts that are within 2-hops of any entity mentioned in the questions of WebQuestionsSP. We also exclude a few "very common" entities, e.g. the ones that describe the hierarchical structure of KB. This smaller KB contains 43 million facts and 12 million entities. We use Wikipedia as our corpus and use a simple entity-linker: we link entities by exact matching to any surface form annotated the FACC1 dataset (Gabrilovich et al., 2013).[4]

**Complex WebQuestions 1.1 (Complex WebQ)** (Talmor and Berant, 2018) is generated from WebQuestionsSP by extending the question entities or adding constraints to answers, in order to construct multi-hop questions.[5] There are four types of question: composition (45%), conjunction (45%), comparative (5%), and superlative (5%). The questions require up to 4-hops of reasoning on the KB and thr 2-hops on corpus. We use the same KB and corpus as for WebQuestionsSP.

| | Train | Dev | Test |
|---|---|---|---|
| MetaQA 1-hop | 96,106 | 9,992 | 9,947 |
| MetaQA 2-hop | 118,980 | 14,872 | 14,872 |
| MetaQA 3-hop | 114,196 | 14,274 | 14,274 |
| WebQuestionsSP | 2,848 | 250 | 1,639 |
| Complex WebQ | 27,623 | 3,518 | 3,531 |

Table 1: Statistics of all datasets.

### 4.2 Tasks

We explored several different QA settings: complete KB only, corpus only, incomplete KB only, and incomplete KB paired with the corpus.

In the *complete KB only* setting, the answer always exists in knowledge base: for all of these datasets, this is true because the questions were crowd-sourced to enforce this conditions. This is the easiest setting for QA, but arguably unrealistic, since with a more natural distribution of ques-

[2]For this version, the 1-hop questions in MetaQA are exactly the same as WikiMovies.

[3]We use the same train/dev/test splits as GRAFT-Net (Sun et al., 2018).

[4]If two overlapping spans are possible matches, we match only the longer of them.

[5]We use Complex WebQuestions v1.1, where the author re-partition the train/dev/test data to prevent the leakage of information.

tions, a KB is likely to be incomplete. In the *text only* setting we use only the corpus. In the *incomplete KB* setting, we simulate KB-based QA on an incomplete KB by randomly discarding some of the triples in the KB: specifically, we randomly drop a fact from the knowledge base with probability $p = 50\%$. This setting is presented mainly as a baseline for the *incomplete KB plus text setting*, where we pair the same incomplete knowledge base with the corpus. In principle this allows a learned QA system to adopt many different hybrid strategies: e.g., for 1-hop queries, a model "back off" to text when the KB is missing information, while in more complex queries, reasoning can involve combining inferences done with text and inferences done with KB triples.

**Comment.** One point to note is that our training procedure is based on finding shortest paths in a complete KB, and we use this same procedure on the incomplete KB setting. Thus the weak training that we use should, in the incomplete-KB settings, be viewed as a form of weak supervision, with labels that are intermediate in informativeness between pure distant training (with only question-answer pairs) and gold inference paths (a setting that has been extensively investigated on some of these datasets, in particular WebQuestionsSP).

### 4.3 Baselines

We choose Key-Value Memory Networks (Miller et al., 2016) and GRAFT-Net (Sun et al., 2018) as our baseline models: to the best of our knowledge, these are the only ones that can use both text and KBs for question answering. However, both models are limited by the number of facts and text that can fit into memory. Thus, we create a separate retrieval process as a pre-processing step, which will be discussed below.

**Key-Value Memory Networks (KVMem)** (Miller et al., 2016) maintain a memory table which stores KB facts and text encoded into key-value pairs. Our encoding of KB facts is the same as is presented in Miller et al. (2016). For text, we use an bi-directional LSTM to encode the text for the key and take the entities mentioned in the text as values. Our implementation shows comparable performance on the WikiMovies (MetaQA 1-hop) dataset, as shown in Table 3.

For **GRAFT-Net** (Sun et al., 2018), we use the implementation published by the author;[6] however, we retrieve data with a simpler process, as described in Section 4.4.

[6] https://github.com/OceanskySun/GraftNet

| | WikiM-KB | WikiM-KB (50%) |
|---|---|---|
| MetaQA-1hop | 0.995 / 9.17 | 0.544 / 4.58 |
| MetaQA-2hop | 0.983 / 47.3 | 0.344 / 28.6 |
| MetaQA-3hop | 0.923 / 459.2 | 0.522 / 316.6 |
| | Freebase | Freebase (50%) |
| WebQuestionsSP | 0.927 / 1876.9 | 0.485 / 1212.5 |
| ComplexWebQ | 0.644 / 1948.7 | 0.542 / 1849.2 |

Table 2: Retrieval results (recall / # entities in graph) on MetaQA ($m = 500$), and WebQuestionsSP and Complex WebQuestions with Freebase ($m = 2000$).

### 4.4 Subgraph Retrieval for Baseline Models

Text is retrieved (non-iteratively) using IDF-based similarity to the question. It is not obvious how to perform KB retrieval: we would like to retrieve as many facts as possible to maximize the recall of answers, but it is infeasible to take all facts that are within $k$-hops of question entities since the number grows exponentially. Based on prior work in the database community on sampling from graphs (Leskovec and Faloutsos, 2006) and local partitioning (Andersen et al., 2006a), we ran an approximate version of personalized PageRank (aka random walk with reset) to find the KB entities closest to the entities in the question—specifically, we used the PageRank-Nibble algorithm (Andersen et al., 2006b) with $\epsilon = 1e^{-6}$ and the picked the $m$ top-scoring entities.[7] We then eliminate all top-$m$ entities that are more than $k$-hops of the question entities, and finally, retrieve all facts from the KB connecting retrieved entities.

The results for several tasks are shown in Table 2, which gives the answer recall, and the average number of entities retrieved. For the smaller MetaQA KB with $m = 500$, the retrieval method finds high-coverage graphs when the KB is complete. For ComplexWebQuestions, even with $m = 2000$, the recall is 64%—which is expected, since retrieving relevant entities for a multi-hop question from a KB with millions of entities is difficult.

### 4.5 Main Results

#### 4.5.1 MetaQA

The experimental results for MetaQA are shown in Table 3. For 1-hop questions (which is identical to the WikiMovies dataset), PullNet is comparable

[7] Notice that in this process we do not use answer entities which are of course not available at test time.

| | MetaQA (1-hop) / wikimovies | | | | MetaQA (2-hop) | | | | MetaQA (3-hop) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | KB | Text | 50% KB | 50% KB + Text | KB | Text | 50% KB | 50%KB + Text | KB | Text | 50% KB | 50% KB + Text |
| KV-Mem* | 96.2 | 75.4 | (63.6) | 75.7 | 82.7 | 7.0 | (41.8) | 48.4 | 48.9 | 19.5 | (37.6) | 35.2 |
| GRAFT-Net* | 97.0 | 82.5 | (64.0) | 91.5 | 94.8 | 36.2 | (52.6) | 69.5 | 77.7 | 40.2 | (59.2) | 66.4 |
| PullNet (Ours) | 97.0 | 84.4 | (65.1) | 92.4 | **99.9** | **81.0** | (52.1) | **90.4** | **91.4** | **78.2** | (59.7) | **85.2** |
| KV-Mem | 93.9 | 76.2 | – | – | – | – | – | – | – | – | – | – |
| GRAFT-Net | 96.8 | **86.6** | (68.0) | **92.6** | – | – | – | – | – | – | – | – |
| VRN | **97.5** | – | – | – | 89.9 | – | – | – | 62.5 | – | – | – |

Table 3: Hits@1 on MetaQA compared to baseline models. Number below the double line are from original papers: KV-Mem (KB) (Miller et al., 2016), KV-Mem (Text) (Watanabe et al., 2017), GRAFT-Net (Sun et al., 2018), and VRN (Zhang et al., 2018). *Reimplemented or different retrieval process.

to the state-of-the-art.[8] We also see that our simplified retrieval pipeline has slightly lower performance than the original (Sun et al., 2018) method. However, for multi-hop questions, PullNet generally shows a large improvement over the baseline models. In the KB-only setting, the absolute performance improvement over the best previously published results is 10 points for 2-hop and almost 30 points for 3-hop questions—and there are even larger improvements relative to the baseline models in the settings using text. In the incomplete KB plus text setting, PullNet also consistently improves over either the text-only or incomplete-KB-only setting.[9] This supports our claim that PullNet is able to effectively combine a KB and text.[10]

#### 4.5.2 WebQuestionsSP

Table 4 presents similar results on the WebQuestionsSP dataset. PullNet improves over GRAFT-Net in the incomplete KB plus text setting, and the addition of text to the incomplete KB also improves performance for PullNet.

#### 4.5.3 Complex WebQuestions

Complex WebQuestions contains multi-hop questions against Freebase: intuitively, one would expect that single-shot retrieval of facts and text would not be able to always find sufficient information to answer such questions. Table 5 shows our results for Complex WebQuestions on the development set. As expected, PullNet shows significant improvement over GRAFT-Net and KV-Mem on all four settings. Once again we see some improvement when pairing the incomplete knowledge base with text, compared to using the incomplete knowledge base only or the text only.

| | KB | Text | 50% KB | 50% KB + Text |
|---|---|---|---|---|
| KV-Mem* | 46.7 | 23.2 | (32.7) | 31.6 |
| GRAFT-Net* | 66.4 | 24.9 | (48.2) | 49.7 |
| PullNet (Ours) | **68.1** | 24.8 | (50.3) | **51.9** |
| GRAFT-Net | 67.8 | **25.3** | (47.7) | 49.9 |
| NSM | 69.0 (F1) | – | – | – |

Table 4: Hits@1 on WebQuestionsSP compared to baseline models. Number below the double line are from the original papers: GRAFT-Net (Sun et al., 2018), and NSM (Liang et al., 2017) (which only reports F1 in their paper). * Reimplemented or different retrieval process.

| | KB | Text (Wikipedia) | 50% KB | 50% KB + Text |
|---|---|---|---|---|
| KV-Mem* | 21.1 | 7.4 | (14.8) | 15.2 |
| GRAFT-Net* | 32.8 | 10.6 | (26.1) | 26.9 |
| PullNet (Ours) | **47.2** | **13.1** | (31.5) | **33.7** |
| | | (Snippets) | | |
| SplitQA | – | 34.2 | – | – |
| PullNet (Ours) | 45.9 | 29.7 | – | – |

Table 5: Hits@1 on Complex WebQuestions compared to baseline models. Above the double line results are on the dev set, below on the test set. *Reimplemented or different retrieval process.

Researchers are only allowed limited submissions on the Complex WebQuestions test set, however, some results for the test set are shown in Table 5. On the test set, our model has 45.9% Hits@1 in the KB only setting and 13.8% Hits@1 (not listed in Table 5) in the text only setting with Wikipedia corpus, which are comparable to the dev set results.

For completeness, we also compare to SplitQA (Talmor and Berant, 2018). SplitQA decomposes a multi-hop question into several simpler sub-questions, and sends each sub-question to the Google search engine. In then applies a reading comprehension model to these web snippets to find answers. The snippet corpus is made available

[8] 6.7% questions in Wikimovies are ambiguous, e.g. questions about movies with remakes without specifying years.

[9] The implement KB results are parenthesized because it presented only to compare against the incomplete KB plus text setting.

[10] Although we do not repeat the results here, we note that prior work (Sun et al., 2018) showed GRAFT-Net outperforms "late fusion" approaches for combining information: in late fusion, two QA systems are ensembled together, one which uses the incomplete KB, and one which uses the text.

with the Complex WebQuestions dataset, but it is arguably biased toward the SplitQA model, since it was collected specifically to support it, and also relies on non-reproducible, non-open-source components. With this corpus our model has 4.5% lower Hits@1 than SplitQA. However, using the KB, PullNet has much better performance than SplitQA, with an absolute gain of 11.7%.

### 4.6 Additional Results

#### 4.6.1 Retrieval Performance of PullNet

In Figure 1a we compare the retrieval performance of PullNet to that of PageRank-Nibble on multi-hop questions with a complete KB, on two multi-hop problems, varying the number of entities retrieved by PageRank-Nibble. PullNet retrieves far fewer entities but obtains higher recall.

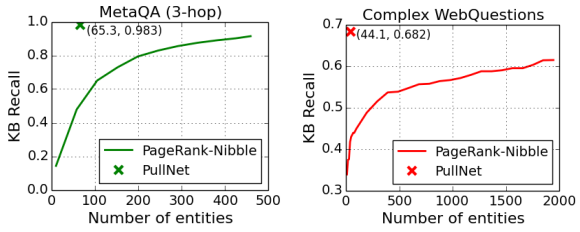


(a) Recall of graphs retrieved by PageRank-Nibble compared with PullNet.

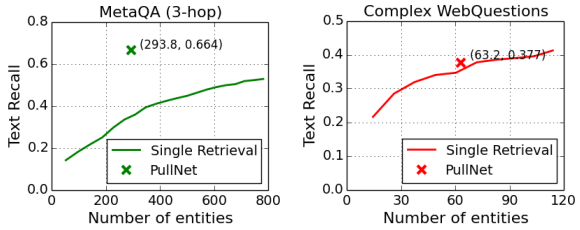


(b) Recall of a single round of retrieval with Apache Lucene compared with PullNet.

Figure 1: Retrieval performance on different datasets.

In Figure 1b we evaluate the effectiveness of iterative retrieval for multi-hop questions on a text corpus. PullNet, with multiple iterations of retrieval, greatly outperfoms a single iteration of IDF-based retrieval on 3-hop MetaQA, and slighly outperforms IDF on Complex WebQuestions.

Figure 2 shows the recall of question subgraphs on 3-hop MetaQA questions as training proceeds. Performance of the retrieval components of PullNet converges relatively quickly, with recall saturating after 10-20,000 examples (about 10-20% of a single epoch).

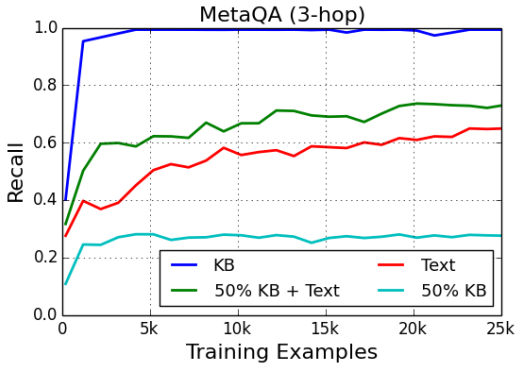


Figure 2: Recall of question subgraph on MetaQA 3-hop questions.

#### 4.6.2 Training Time

PullNet's algorithm is quite different from prior systems, since learning and retrieval are interleaved: in most prior systems, including GRAFT-Net, retrieval is performed only once, before learning. Intuitively, interleaving learning with the relatively slow operation of retrieval is potentially slower; on the other hand, PullNet's final question subgraph is smaller than GRAFT-Net, which makes learning potentially faster.

To study these issues, we plot the Hits@1 performance of learned model versus wall clock time in Figure 3. This experiment is run on Complex WebQuestions in the KB-only setting, using one high-end GPU.[11] GRAFT-Net takes an average of 31.9 minutes per epoch, while PullNet takes an average of 114 minutes per epoch, about 3.5 times slower.

As the graph shows, initially PullNet's performance is better, since GRAFT-Net cannot start learning until the preprocessing finishes. GRAFT-Net's faster learning speed then allows it to dominates for some time. GRAFT-Net reaches its peak in about 3.6 hours. PullNet passes GRAFT-Net after around 6 hours (about 3 epochs for PullNet).

## 5 Conclusions

PullNet is a novel integrated QA framework for (1) learning what to retrieve from a KB and/or corpus and (2) reasoning with this heterogeneous data to find the best answer. Unlike prior work, PullNet uses an iterative process to construct a question-specific subgraph that contains information relevant to the question. In each iteration, a

[11] To be fair to GRAFT-Net, we used a fast in-memory implementation of PageRank-Nibble (based on SciPy sparse matrices), which takes about 40 minutes.

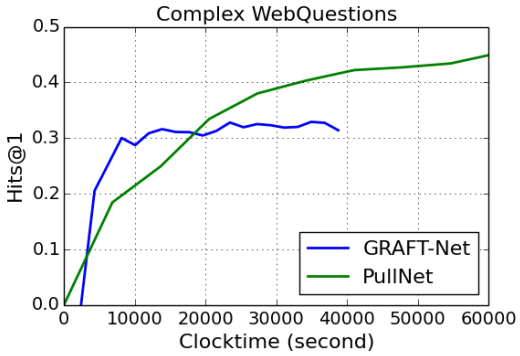


Figure 3: Performance of PullNet and GRAFT-Net under wall clock training time.

graph CNN is used to identify subgraph nodes that should be expanded using "pull" operations on the corpus and/or KB. This iterative process makes it possible to retrieve a small graph that contains just the information relevant to a multi-hop question.

Experimentally PullNet improves over the prior state-of-the-art for the setting in which questions are answered with a corpus plus an incomplete KB, or in settings in which questions need "multi-hop" reasoning. Sometimes the performance improvements are dramatic: e.g., an improvement from 62.5% Hits@1 to 91.4% Hits@1 for 3-hop MetaQa with a KB, or improvements from 32.8% to 47.2% for Complex WebQuestions with a KB.

## References


Reid Andersen, Fan Chung, and Kevin Lang. 2006a. Local graph partitioning using pagerank vectors. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 475–486. IEEE.

Reid Andersen, Fan Chung, and Kevin Lang. 2006b. Local graph partitioning using pagerank vectors. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 475–486. IEEE.

Hannah Bast and Elmar Haussmann. 2015. More accurate question answering on freebase. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1431–1440. ACM.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Association for Computational Linguistics (ACL)*.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2017. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. *arXiv preprint arXiv:1711.05851*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. 2016. Gated-attention readers for text comprehension. *arXiv preprint arXiv:1606.01549*.

Bhuwan Dhingra, Kathryn Mazaitis, and William W Cohen. 2017. Quasar: Datasets for question answering by search and reading. *arXiv preprint arXiv:1707.03904*.

Matthew Dunn, Levent Sagun, Mike Higgins, V Ugur Guney, Volkan Cirik, and Kyunghyun Cho. 2017. Searchqa: A new q&a dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179*.

Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. 2013. Facc1: Freebase annotation of clueweb corpora, version 1 (release date 2013-06-26, format version 1, correction level 0).

Mingfei Gao, Ruichi Yu, Ang Li, Vlad I Morariu, and Larry S Davis. 2018. Dynamic zoom-in network for fast object detection in large images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6926–6935.

Heng Ji, Joel Nothman, Ben Hachey, et al. 2014. Overview of tac-kbp2014 entity discovery and linking tasks. In *Proc. Text Analysis Conference (TAC2014)*, pages 1333–1339.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Vancouver, Canada. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*.

Ni Lao, Amarnag Subramanya, Fernando Pereira, and William W Cohen. 2012. Reading the web with learned syntactic-semantic inference rules. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1017–1026. Association for Computational Linguistics.

Jure Leskovec and Christos Faloutsos. 2006. Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 631–636. ACM.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. 2016. Gated graph sequence neural networks. *ICLR*.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. *ACL*.

Rui Liu, Wei Wei, Weiguang Mao, and Maria Chikina. 2017. Phase conductor on multi-layered attentions for machine comprehension. *arXiv preprint arXiv:1710.10504*.

Michael McCandless, Erik Hatcher, and Otis Gospodnetic. 2010. *Lucene in action: covers Apache Lucene 3.0*. Manning Publications Co.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. *EMNLP*.

Bonan Min, Ralph Grishman, Li Wan, Chang Wang, and David Gondek. 2013. Distant supervision for relation extraction with an incomplete knowledge base. In *NAACL HLT 2013 - 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Main Conference*, pages 777–782. Association for Computational Linguistics (ACL).

Uma Sawant, Saurabh Garg, Soumen Chakrabarti, and Ganesh Ramakrishnan. 2019. Neural architecture for question answering using a knowledge graph and web corpus. *Information Retrieval Journal*, pages 1–26.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2017. Modeling relational data with graph convolutional networks. *arXiv preprint arXiv:1703.06103*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Minjoon Seo, Tom Kwiatkowski, Ankur Parikh, Ali Farhadi, and Hannaneh Hajishirzi. 2018. Phrase-indexed question answering: A new challenge for scalable document comprehension. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 559–564.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *NIPS*.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William W Cohen. 2018. Open domain question answering using early fusion of knowledge bases and text. *EMNLP*.

A. Talmor and J. Berant. 2018. The web as a knowledge-base for answering complex questions. In *North American Association for Computational Linguistics (NAACL)*.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerry Tesauro, Bowen Zhou, and Jing Jiang. 2018. R 3: Reinforced ranker-reader for open-domain question answering. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Shuohang Wang, Mo Yu, Jing Jiang, Wei Zhang, Xiaoxiao Guo, Shiyu Chang, Zhiguo Wang, Tim Klinger, Gerald Tesauro, and Murray Campbell. 2017. Evidence aggregation for answer re-ranking in open-domain question answering. *CoRR*, abs/1711.05116.

Yusuke Watanabe, Bhuwan Dhingra, and Ruslan Salakhutdinov. 2017. Question answering from unstructured text by retrieval and comprehension. *arXiv preprint arXiv:1703.08885*.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association of Computational Linguistics*, 6:287–302.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1321–1331, Beijing, China. Association for Computational Linguistics.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*.

John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the Thirteenth National conference on Artificial intelligence-Volume 2*, pages 1050–1055. AAAI Press.

Luke S Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: structured classification with probabilistic categorial grammars. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, pages 658–666. AUAI Press.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J Smola, and Le Song. 2018. Variational reasoning for question answering with knowledge graph. In *AAAI*.

# COSMOS QA: Machine Reading Comprehension with Contextual Commonsense Reasoning

**Lifu Huang**[♠*], **Ronan Le Bras**[♣], **Chandra Bhagavatula**[♣], **Yejin Choi**[♣,♢]
[♠] University of Illinois Urbana-Champaign, Champaign, IL, USA
[♣] Allen Institute for Artificial Intelligence, Seattle, WA, USA
[♢] University of Washington, Seattle, WA, USA
lifuh2@illinois.edu
{ronanlb, chandrab, yejinc}@allenai.org

## Abstract

Understanding narratives requires reading between the lines, which in turn, requires interpreting the likely causes and effects of events, even when they are not mentioned explicitly. In this paper, we introduce COSMOS QA, a large-scale dataset of 35,600 problems that require commonsense-based reading comprehension, formulated as multiple-choice questions. In stark contrast to most existing reading comprehension datasets where the questions focus on factual and literal understanding of the context paragraph, our dataset focuses on reading between the lines over a diverse collection of people's everyday narratives, asking such questions as *"what might be the possible reason of ...?"*, or *"what would have happened if ..."* that require reasoning beyond the exact text spans in the context. To establish baseline performances on COSMOS QA, we experiment with several state-of-the-art neural architectures for reading comprehension, and also propose a new architecture that improves over the competitive baselines. Experimental results demonstrate a significant gap between machine (68.4%) and human performance (94%), pointing to avenues for future research on commonsense machine comprehension. Dataset, code and leaderboard is publicly available at https://wilburone.github.io/cosmos.

**P1:** It's a very humbling experience when you need someone to dress you every morning, tie your shoes, and put your hair up. Every menial task takes an unprecedented amount of effort. It made me appreciate Dan even more. But anyway I shan't dwell on this (I'm not dying after all) and not let it detract from my lovely 5 days with my friends visiting from Jersey.

**Q:** *What's a possible reason the writer needed someone to dress him every morning?*

**A:** The writer doesn't like putting effort into these tasks.
✓ **B: The writer has a physical disability.**
**C:** The writer is bad at doing his own hair.
**D:** None of the above choices.

**P2:** A woman had topped herself by jumping off the roof of the hospital she had just recently been admitted to. She was there because the first or perhaps latest suicide attempt was unsuccessful. She put her clothes on, folded the hospital gown and made the bed. She walked through the unit unimpeded and took the elevator to the top floor.

**Q:** *What would have happened to the woman if the staff at the hospital were doing their job properly?*

✓ **A: The woman would have been stopped before she left to take the elevator to the top floor and she would have lived.**
**B:** She would have been ushered to the elevator with some company.
**C**: She would have managed to get to the elevator quicker with some assistance.
**D:** None of the above choices.

Figure 1: Examples of COSMOS QA. (✓ indicates the correct answer.) Importantly, (1) the correct answer is not explicitly mentioned anywhere in the context paragraph, thus requiring reading between the lines through commonsense inference and (2) answering the question correctly requires reading the context paragraph, thus requiring reading comprehension and *contextual* commonsense reasoning.

## 1 Introduction

Reading comprehension requires not only understanding what is stated explicitly in text, but also *reading between the lines*, i.e., understanding what is not stated yet obviously true (Norvig, 1987).

For example, after reading the first paragraph in Figure 1, we can understand that the writer is not a child, yet needs someone to dress him or her every morning, and appears frustrated with the current situation. Combining these clues, we can infer that the plausible reason for the writer being dressed by other people is that he or she may have a physical disability.

As another example, in the second paragraph of Figure 1, we can infer that the woman was admitted to a psychiatric hospital, although not mentioned explicitly in text, and also that the job of the hospital staff is to stop patients from committing suicide. Furthermore, what the staff should

[*]The work has been done during the author's internship in AI2.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2391–2401, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

have done, in the specific situation described, was to stop the woman from taking the elevator.

There are two important characteristics of the problems presented in Figure 1. First, the correct answers are not explicitly mentioned anywhere in the context paragraphs, thus requiring reading between the lines through commonsense inference. Second, selecting the correct answer requires reading the context paragraphs. That is, if we were not provided with the context paragraph for the second problem, for example, the plausible correct answer could have been B or C instead.

In this paper, we focus on reading comprehension that requires *contextual* commonsense reasoning, as illustrated in the examples in Figure 1. Such reading comprehension is an important aspect of how people read and comprehend text, and yet, relatively less studied in the prior machine reading literature. To support research toward commonsense reading comprehension, we introduce COSMOS QA (**Co**mmon**s**ense **M**achine C**o**mprehen**s**ion), a new dataset with $35,588$ reading comprehension problems that require reasoning about the causes and effects of events, the likely facts about people and objects in the scene, and hypotheticals and counterfactuals. Our dataset covers a diverse range of everyday situations, with $21,886$ distinct contexts taken from blogs of personal narratives.

The vast majority ($93.8\%$) of our dataset requires contextual commonsense reasoning, in contrast with existing machine comprehension (MRC) datasets such as SQuAD (Rajpurkar et al., 2016), RACE (Lai et al., 2017), Narrative QA (Kočiskỳ et al., 2018), and MCScript (Ostermann et al., 2018), where only a relatively smaller portion of the questions (e.g., $27.4\%$ in MCScript) require commonsense inference. In addition, the correct answer cannot be found in the context paragraph as a text span, thus we formulate the task as multiple-choice questions for easy and robust evaluation. However, our dataset can also be used for generative evaluation, as will be demonstrated in our empirical study.

To establish baseline performances on COSMOS QA, we explore several state-of-the-art neural models developed for reading comprehension. Furthermore, we propose a new architecture variant that is better suited for commonsense-driven reading comprehension. Still, experimental results demonstrate a significant gap between machine (68.4% accuracy) and human performance (94.0%). We provide detailed analysis to provide insights into potentially promising research directions.

## 2 Dataset Design

### 2.1 Context Paragraphs

We gather a diverse collection of everyday situations from a corpus of personal narratives (Gordon and Swanson, 2009) from the Spinn3r Blog Dataset (Burton et al., 2009). Appendix A provides additional details on data pre-processing.

### 2.2 Question and Answer Collection

We use Amazon Mechanical Turk (AMT) to collect questions and answers. Specifically, for each paragraph, we ask a worker to craft at most two questions that are related to the context and require commonsense knowledge. We encourage the workers to craft questions from but not limited to the following four categories:

**Causes of events:** What may (or may not) be the plausible reason for an event?

**Effects of events:** What may (or may not) happen before (or after, or during) an event?

**Facts about entities:** What may (or may not) be a plausible fact about someone or something?

**Counterfactuals:** What may (or may not) happen if an event happens (or did not happen)?

These 4 categories of questions literally cover all 9 types of social commonsense of Sap et al. (2018). Moreover, the resulting commonsense also aligns with 19 ConceptNet relations, e.g., *Causes*, *HasPrerequisite* and *MotivatedByGoal*, covering about 67.8% of ConceptNet types. For each question, we also ask a worker to craft at most two correct answers and three incorrect answers. We paid workers $0.7 per paragraph, which is about $14.8 per hour. Appendix B provides additional details on AMT instructions.

### 2.3 Validation

We create multiple tasks to have humans verify the data. Given a paragraph, a question, a correct answer and three incorrect answers,[1] we ask AMT workers to determine the following sequence of questions: (1) whether the paragraph is inappropriate or nonsensical, (2) whether the question is

[1] If a question is crafted with two correct answers, we will create two question sets with each correct answer and the same three incorrect answers.

| | Train | Dev | Test | All |
|---|---|---|---|---|
| # Questions (Paragraphs) | 25,588 (13,715) | 3,000 (2,460) | 7,000 (5,711) | 35,588 (21,866) |
| Ave./Max. # Tokens / Paragraph | 69.4 / 152 | 72.6 / 150 | 73.1 / 149 | 70.3/ 152 |
| Ave./Max. # Tokens / Question | 10.3 / 34 | 11.2 / 28 | 11.2 / 29 | 10.6 / 34 |
| Ave./Max. # Tokens / Correct Answer | 8.0 / 40 | 9.7 / 41 | 9.7 / 36 | 8.5 / 41 |
| Ave./Max. # Tokens / Incorrect Answer | 7.6 / 40 | 9.1 / 38 | 9.1 / 36 | 8.0 / 40 |
| Percentage of Unanswerable Questions | 5.9% | 8.7% | 8.4% | 6.7% |

Table 1: Statistics of training, dev and test sets of COSMOS QA.

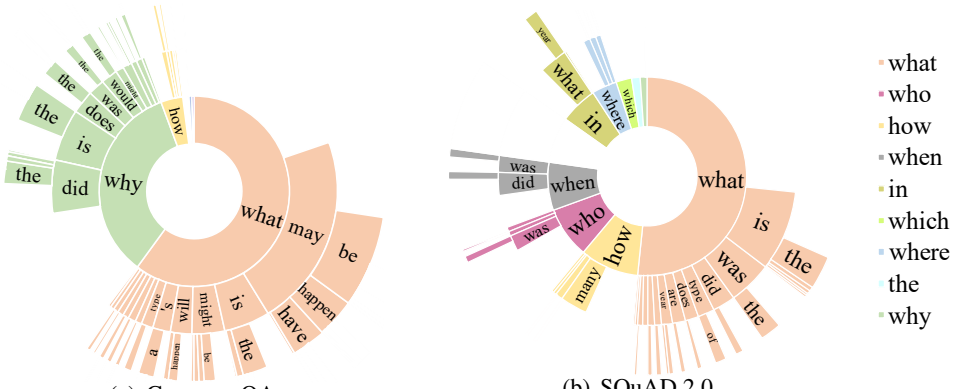


(a) COSMOS QA (b) SQuAD 2.0

Figure 2: Distribution of trigram prefixes of questions in COSMOS QA and SQuAD 2.0

nonsensical or not related to the paragraph, (3) whether they can determine the most plausible correct answer, (4) if they can determine the correct answer, whether the answer requires commonsense knowledge, and (5) if they can determine the correct answer, whether the answer can be determined without looking at the paragraph.

We follow the same criterion as in Section 2.2 and ask 3 workers to work on each question set. Workers are paid $0.1 per question. We consider as valid question set where at least two workers correctly picked the intended answer and all of the workers determined the paragraph/question/answers as satisfactory. Finally we obtain 33,219 valid question sets in total.

### 2.4 Unanswerable Question Creation

With human validation, we also obtain a set of questions for which workers can easily determine the correct answer without looking at the context or using commonsense knowledge. To take advantage of such questions and encourage AI systems to be more consistent with human understanding, we create unanswerable questions for COSMOS QA. Specifically, from validation outputs, we collect 2,369 questions for which at least two workers correctly picked the answer and at least on worker determined that it is answerable without looking at the context or requires no common sense. We replace the correct choice of these questions with a "None of the above" choice.

To create false negative training instances, we randomly sample 70% of questions from the 33,219 good question sets and replace their least challenging negative answer with "None of the above". Specifically, we fine-tune three BERT[2] next sentence prediction models on COSMOS: BERT($A|P,Q$), BERT($A|P$), BERT($A|Q$), where $P$, $Q$, $A$ denotes the paragraph, question, and answer. BERT($A|\triangle$) denotes the possibility of an answer $A$ being the next sentence of $\triangle$. The least challenging negative answer is determined by

$$A^{'} = \arg\min\Big(\sum_{\forall \triangle \subseteq \{P,Q\}} \text{BERT}(A|\triangle)\Big)$$

### 2.5 Train / Dev / Test Split

We finally obtain 35,588 question sets for our COSMOS dataset. To ensure that the development and test sets are of high quality, we identify a group of workers who excelled in the generation task for question and answers, and randomly sample 7K question sets authored by these excellent workers as test set, and 3K question sets as development set. The remaining questions are all used as training set. Table 1 shows dataset statistics.

[2]Through the whole paper, BERT refers to the pre-trained BERT large uncased model from https://github.com/huggingface/pytorch-pretrained-BERT

### 2.6 Data Analysis

Figure 2 compares frequent trigram prefixes in COSMOS and SQuAD 2.0 (Rajpurkar et al., 2018). Most of the frequent trigram prefixes in COSMOS, e.g., *why*, *what may happen*, *what will happen* are almost absent from SQuAD 2.0, which demonstrates the unique challenge our dataset contributes. We randomly sample 500 answerable questions to manually categorize according to their contextual commonsense reasoning types. Figure 3 shows representative examples. Table 2 shows the distribution of the question types.

- **Pre-/post-conditions:** causes/effects of an event.
- **Motivations:** intents or purposes.
- **Reactions:** possible reactions of people or objects to an event.
- **Temporal events:** what events might happen before or after the current event.
- **Situational facts:** facts that can be inferred from the description of a particular situation.
- **Counterfactuals:** what might happen given a counterfactual condition.
- **Other:** other types, e.g., cultural norms.

| Type | Percentage (%) |
|---|---|
| MRC w/o commonsense | 6.2 |
| MRC w/ commonsense | 93.8 |
| Pre-/Post- Condition | 27.2 |
| Motivation | 16.0 |
| Reaction | 13.2 |
| Temporal Events | 12.4 |
| Situational Fact | 23.8 |
| Counterfactual | 4.4 |
| Other | 12.6 |

Table 2: The distribution of contextual commonsense reasoning types in COSMOS.

## 3 Model

### 3.1 BERT with Multiway Attention

Multiway attention (Wang et al., 2018a; Zhu et al., 2018) has been shown to be effective in capturing the interactions between each pair of input paragraph, question and candidate answers, leading to better context interpretation, while BERT fine-tuning (Devlin et al., 2018) also shows its prominent ability in commonsense inference. To further enhance the context understanding ability of BERT fine-tuning, we perform multiway bidirectional attention over the BERT encoding output. Figure 4 shows the overview of the architecture.

**Encoding with Pre-trained BERT** Given a paragraph, a question, and a set of candidate answers, the goal is to select the most plausible correct answer from the candidates. We formulate the input paragraph as $P = \{p_0, p_1, ..., p_n\}$, the question as $Q = \{q_0, q_1, ..., q_k\}$ and a candidate answer as $A = \{a_0, a_1, ..., a_s\}$, where $p_i$, $q_i$ and $a_i$ is the $i$-th word of the paragraph, question and candidate answer respectively. Following (Devlin et al., 2018), given the input $P$, $Q$ and $A$, we apply the same tokenizer and concatenate all tokens as a new sequence $\big[\text{[CLS]}, P, \text{[SEP]}, Q, \text{[SEP]}, A, \text{[SEP]}\big]$, where [CLS] is a special token used for classification and [SEP] is a delimiter. Each token is initialized with a vector by summing the corresponding token, segment and position embedding from pre-trained BERT, and then encoded into a hidden state. Finally we get $[\mathbf{H}_{cls}, \mathbf{H}_P, \mathbf{H}_Q, \mathbf{H}_A]$ as encoding output.

**Multiway Attention** To encourage better context interpretation, we perform multiway attention over BERT encoding output. Taking the paragraph $P$ as an example, we compute three types of attention weights to capture its correlation to the question, the answer, and both the question and answer, and get question-attentive, answer-attentive, and question and answer-attentive paragraph representations

$$\tilde{\mathbf{H}}_P = \mathbf{H}_P \boldsymbol{W}_t + \boldsymbol{b}_t$$
$$\mathbf{M}_P^Q = \text{Softmax}(\tilde{\mathbf{H}}_P \mathbf{H}_Q^\top)\mathbf{H}_Q$$
$$\mathbf{M}_P^A = \text{Softmax}(\tilde{\mathbf{H}}_P \mathbf{H}_A^\top)\mathbf{H}_A$$
$$\mathbf{M}_P^{QA} = \text{Softmax}(\tilde{\mathbf{H}}_P \mathbf{H}_{QA}^\top)\mathbf{H}_{QA}$$

where $\boldsymbol{W}_t$ and $\boldsymbol{b}_t$ are learnable parameters. Next we fuse these representations with the original encoding output of $P$

$$\mathbf{F}_P^Q = \sigma([\mathbf{H}_P \mathbf{M}_P^Q : \mathbf{H}_P - \mathbf{M}_P^Q]\boldsymbol{W}_P + \boldsymbol{b}_P)$$
$$\mathbf{F}_P^A = \sigma([\mathbf{H}_P \mathbf{M}_P^A : \mathbf{H}_P - \mathbf{M}_P^A]\boldsymbol{W}_P + \boldsymbol{b}_P)$$
$$\mathbf{F}_P^{QA} = \sigma([\mathbf{H}_P \mathbf{M}_P^{QA} : \mathbf{H}_P - \mathbf{M}_P^{QA}]\boldsymbol{W}_P + \boldsymbol{b}_P)$$

where $[:]$ denotes concatenation operation. $\boldsymbol{W}_P$, $\boldsymbol{b}_P$ are learnable parameters for fusing paragraph representations. $\sigma$ denotes ReLU function.

Finally, we apply column-wise max pooling on $[\mathbf{F}_P^Q : \mathbf{F}_P^A : \mathbf{F}_P^{QA}]$ and obtain the new paragraph representation $\mathbf{F}_P$. Similarly, we can also obtain a new representation $\mathbf{F}_Q$ and $\mathbf{F}_A$ for $Q$ and $A$ respectively. We use $\mathbf{F} = [\mathbf{F}_P : \mathbf{F}_Q : \mathbf{F}_A]$ as the

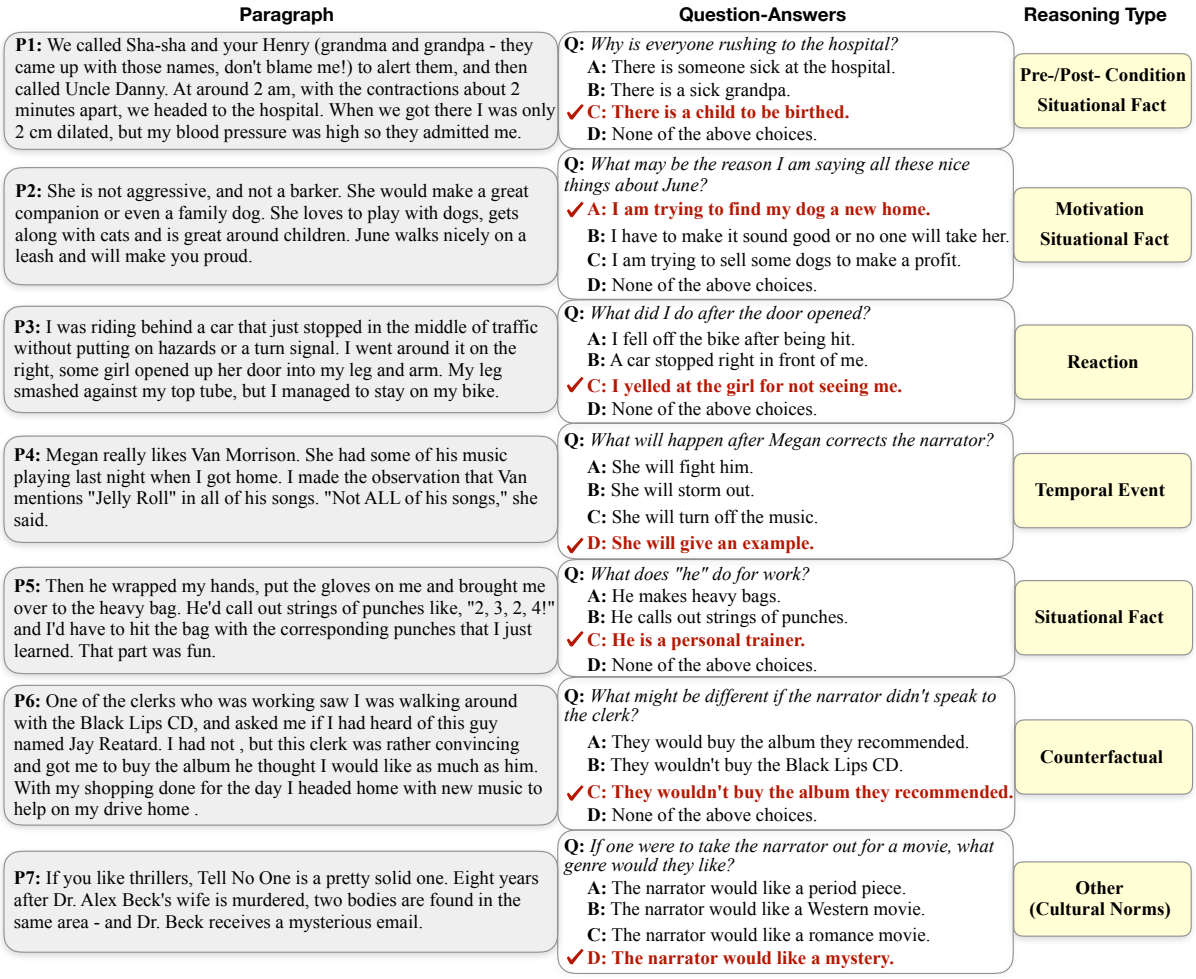


Figure 3: Examples of each type of commonsense reasoning in COSMOS QA. (✓ indicates the correct answer.)

overall vector representation for the set of paragraph, question and a particular candidate answer.

**Classification** For each candidate answer $A_i$, we compute the loss as follows:

$$L(A_i|P,Q) = -\log \frac{\exp(\boldsymbol{W}_f^\top \mathbf{F}_i)}{\sum_{j=1}^{4} \exp(\boldsymbol{W}_f^\top \mathbf{F}_j))}$$

## 4 Experiments

### 4.1 Baseline Methods

We explore two categories of baseline methods: reading comprehension approaches and pre-trained language model based approaches.

**Sliding Window** (Richardson et al., 2013) measures the similarity of each candidate answer with each window with $m$ words of the paragraph.

**Stanford Attentive Reader** (Chen et al., 2016) performs a bilinear attention between the question and paragraph for answer prediction.

**Gated-Attention Reader** (Dhingra et al., 2017) performs multi-hop attention between the question and a recurrent neural network based paragraph encoding states.

**Co-Matching** (Wang et al., 2018b) captures the interactions between question and paragraph, as well as answer and paragraph with attention.

**Commonsense-RC** (Wang et al., 2018a) applies three-way unidirectional attention to model interactions between paragraph, question, and answers.

**GPT-FT** (Radford et al., 2018) is based on a generative pre-trained transformer language model, following a fine-tuning step on COSMOS QA.

**BERT-FT** (Devlin et al., 2018) is a pre-trained bidirectional transformer language model following a fine-tuning step on COSMOS QA.

**DMCN** (Zhang et al., 2019a) performs dual attention between paragraph and question/answer over BERT encoding output.

**Human Performance** To get human performance on COSMOS QA, we randomly sample 200 question sets from the test set, and ask 3 workers

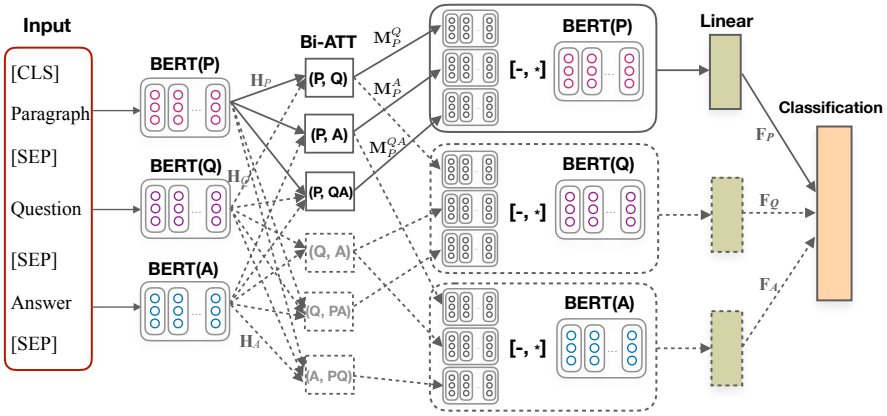


Figure 4: Architecture overview of BERT with multiway attention: Solid lines and blocks show the learning of multiway attentive context paragraph representation.

| Model | Att(P, Q) | Att(P, A) | Att(Q, A) | Pre-training LM | Dev | Test |
|---|---|---|---|---|---|---|
| Sliding Window (Richardson et al., 2013) | ✗ | ✗ | ✗ | ✗ | 25.0 | 24.9 |
| Stanford Attentive Reader (Chen et al., 2016) | UD | ✗ | ✗ | ✗ | 45.3 | 44.4 |
| Gated-Attention Reader (Dhingra et al., 2017) | Multi-hop UD | ✗ | ✗ | ✗ | 46.9 | 46.2 |
| Co-Matching (Wang et al., 2018b) | UD | UD | ✗ | ✗ | 45.9 | 44.7 |
| Commonsense-Rc (Wang et al., 2018a) | UD | UD | UD | ✗ | 47.6 | 48.2 |
| GPT-FT (Radford et al., 2018) | ✗ | ✗ | ✗ | UD | 54.0 | 54.4 |
| BERT-FT (Devlin et al., 2018) | ✗ | ✗ | ✗ | BD | 66.2 | 67.1 |
| DMCN (Zhang et al., 2019a) | UD | UD | ✗ | BD | 67.1 | 67.6 |
| BERT-FT Multiway | BD | BD | BD | BD | 68.3 | 68.4 |
| Human | | | | | | 94.0 |

Table 3: Comparison of varying approachs (Accuracy %). Att: Attention, UD: Unidirectional, BD: Bidirectional

from AMT to select the most plausible correct answer. Each worker is paid $0.1 per question set. We finally combine the predictions for each question with majority vote.

### 4.2 Results and Analysis

Table 3 shows the characteristics and performance of varying approaches and human performance.[3]

Most of the reading comprehension approaches apply attention to capture the correlation between paragraph, question and each candidate answer and tend to select the answer which is the most semantically closed to the paragraph. For example, in Figure 5, the Commonsense-RC baseline mistakenly selected the choice which has the most overlapped words with the paragraph without any commonsense reasoning. However, our analysis shows that more than 83% of correct answers in COSMOS QA are not stated in the given paragraphs, thus simply comparing the semantic relatedness doesn't work well.

Pre-trained language models with fine-tuning achieve more than 20% improvement over reading comprehension approaches. By performing attention over BERT-FT, the performance is further improved, which demonstrates our assumption that incorporating interactive attentions can further enhance the context interpretation of BERT-FT. For example, in Figure 5, BERT-FT mistakenly selected choice A which can be possibly entailed by the paragraph. However, by performing multiway attention to further enhance the interactive comprehension of context, question and answer, our approach successfully selected the correct answer.

## 5 Discussion

### 5.1 Ablation Study

| Model | Dev Acc (%) | Test Acc (%) |
|---|---|---|
| BERT-FT (A\|P, Q) | 66.2 | 67.1 |
| BERT-FT (A\|P) | 63.5 | 64.5 |
| BERT-FT (A\|Q) | 56.2 | 55.9 |
| BERT-FT (A) | 40.3 | 40.3 |

Table 4: Ablation of Paragraphs (P) or Questions (Q)

Many recent studies have suggested the importance of measuring the dataset bias by checking

[3] Appendix C shows the implementation details.

**P:** I cleaned the two large bottom cupboards and threw a ton of old stuff away. Dustin's parents like to drop off boxes of food like we're refugees or something. It's always appreciated, and some of it is edible. Most of what I threw away was from last year when Dustin's great-aunt was moving into her new apartment home (retirement center) and they cleaned out her pantry.

**Q:** *What is the most likely reason that I decided to clean the cupboards ?*

✗ **A:** I was getting tired of having food in the house.

✓ **B: We were getting more food and needed to create room.**

✗ **C:** Dustin and I split up and I need to get rid of his old stuff.

**D:** None of the above choices.

Figure 5: Prediction comparison between our approach (B) with Commonsense-RC (C) and BERT-FT (A).

the model performance based on partial information of the problem (Gururangan et al., 2018; Cai et al., 2017). Therefore, we report problem ablation study in Table 4 using BERT-FT as a simple but powerful straw man approach. Most notably, ablating questions does not cause significant performance drop. Further investigation indicates that this is because the high-level question types, e.g., *what happens next*, *what happened before*, are not diverse, so that it is often possible to make a reasonable guess on what the question may have been based on the context and the answer set. Ablating other components of the problems cause more significant drops in performance.

## 5.2 Knowledge Transfer Through Fine-tuning

Recent studies (Howard and Ruder, 2018; Min et al., 2017; Devlin et al., 2018) have shown the benefit of fine-tuning on similar tasks or datasets for knowledge transfer. Considering the unique challenge of COSMOS, we explore two related multiple-choice datasets for knowledge transfer: RACE (Lai et al., 2017), a large-scale reading comprehension dataset, and SWAG (Zellers et al., 2018), a large-scale commonsense inference dataset. Specifically, we first fine-tune BERT on RACE or SWAG or both, and directly test on COSMOS to show the impact of knowledge transfer. Furthermore, we sequentially fine-tune BERT on both RACE or SWAG and COSMOS. As Table 5 shows, with direct knowledge transfer, RACE provides significant benefit than SWAG since COSMOS requires more understanding of the interaction between paragraph, question and each candidate answer. With sequentially fine-tuning, SWAG provides better performance, which indicates that with fine-tuning on SWAG, BERT can obtain better commonsense inference ability, which is also beneficial to COSMOS.

| Model | Dev Acc | Test Acc |
|---|---|---|
| BERT-FT$_{\text{SWAG}}$ | 28.9 | 28.5 |
| BERT-FT$_{\text{RACE}}$ | 42.0 | 42.5 |
| BERT-FT$_{\text{RACE+SWAG}}$ | 44.2 | 45.1 |
| BERT-FT$_{\text{SWAG}\rightarrow\text{Cosmos}}$ | 67.8 | 68.9 |
| BERT-FT$_{\text{RACE}\rightarrow\text{Cosmos}}$ | 67.4 | 68.2 |
| BERT-FT$_{\text{RACE+SWAG}\rightarrow\text{Cosmos}}$ | 67.1 | 68.7 |

Table 5: Knowledge transfer through fine-tuning. (%)

**P1:** A woman had topped herself by jumping off the roof of the hospital she had just recently been admitted to. She was there because the first or perhaps latest suicide attempt was unsuccessful. She put her clothes on, folded the hospital gown and made the bed. She walked through the unit unimpeded and took the elevator to the top floor.

**Q:** *What would have happened to the woman if the staff at the hospital were doing their job properly?*

✓ **A: The woman would have been stopped before she left to take the elevator to the top floor and she would have lived.**

**B:** She would have been ushered to the elevator with some company.

✗ **C: She would have managed to get to the elevator quicker with some assistance.**

**D:** None of the above choices.

**P2:** Like me, she had no family or friends who could help with childcare. So like me, she found a daycare center that met her part-time needs. In sharp contrast to my job as a (gasp!) writer for the evil MSM, her nursing job was deemed by the other moms to be useful and worthwhile --in fact, worth putting her baby into daycare for "just a few hours, what harm could it do?"

**Q:** *What would happened if she could not find a daycare?*

✓ **A: She would try to find a babysitter.**

**B:** She would take the baby to work.

✗ **C: She would leave the baby alone at home.**

**D:** None of the above choices.

**P3:** My head hurts. I had so much fun at a chat with some scrap friends last Saturday night that I forgot to sleep. I ended up crawling into bed around 7AM.

**Q:** *What may have happened if she did n't chat to her scrap friends ?*

**A:** She would have done some scrap thing at home.

✓ **B: She would not have gotten up with a headache.**

✗ **C: She would have been lonely and stayed up all night.**

**D:** None of the above choices.

Figure 6: Examples errors of our approach. (✓ indicates correct answers and ✗ shows prediction errors.)

## 5.3 Error Analysis

We randomly select 100 errors made by our approach from the dev set, and identify 4 phenomena:

**Complex Context Understanding:** In 30% of the errors, the context requires complicated cross-sentence interpretation and reasoning. Taking P1 in Figure 6 as an example, to correctly predict the answer, we need to combine the context information that *the woman attempted to suicide before but failed*, *she made the bed since she determined to leave*, and *she took the elevator and headed to the roof*, and infer that *the woman was attempting to suicide again*.

**Inconsistent with Human Common Sense:** In

33% of the errors, the model mistakenly selected the choice which is not consistent with human common sense. For example, in P2 of Figure 6, both choice A and choice C could be potentially correct answers. However, from human common sense, *it's not safe to leave a baby alone at home*.

**Multi-turn Commonsense Inference:** 19% of the errors are due to multi-turn commonsense inference. For example, in P3 of Figure 6, the model needs to first determine the cause of *headache* is that *she chatted with friends and forgot to sleep* using common sense. Further, with counterfactual reasoning, if *she didn't chat to her friends*, then *she wouldn't have gotten up with a headache*.

**Unanswerable Questions:** 14% of the errors are from unanswerable questions. The model cannot handle "None of the above" properly since it cannot be directly entailed by the given paragraph or the question. Instead, the model needs to compare the potential of all the other candidate choices.

### 5.4 Generative Evaluation

In real world, humans are usually asked to perform contextual commonsense reasoning without being provided with any candidate answers. To test machine for human-level intelligence, we leverage a state-of-the-art natural language generator GPT2 (Radford et al., 2019) to automatically generate an answer by reading the given paragraph and question. Specifically, we fine-tune a pre-trained GPT2 language model on all the $[Paragraph, Question, Correct\ Answer]$ of COSMOS training set, then given each $[Paragraph, Question]$ from test set, we use GPT2-FT to generate a plausible answer. We automatically evaluate the generated answers against human authored correct answers with varying metrics in Table 6. We also create a AMT task to have 3 workers select all plausible answers among 4 automatically generated answers and a "None of the aboce" choice for 200 question sets. We consider an answer as correct only if all 3 workers determined it as correct. Figure 7 shows examples of automatically generated answers by pre-trained GPT2 and GPT2-FT as well as human authored correct answers. We observe that by fine-tuning on COSMOS, GPT2-FT generates more accurate answers. Although intuitively there may be multiple correct answers to the questions in COSMOS QA, our analysis shows that more than 84% of generated correct answers identified by human are semantically consistent with the gold answers in COSMOS, which demonstrates that COSMOS can also be used as a benchmark for generative commonsense reasoning. Appendix E shows more details and examples for generative evaluation.

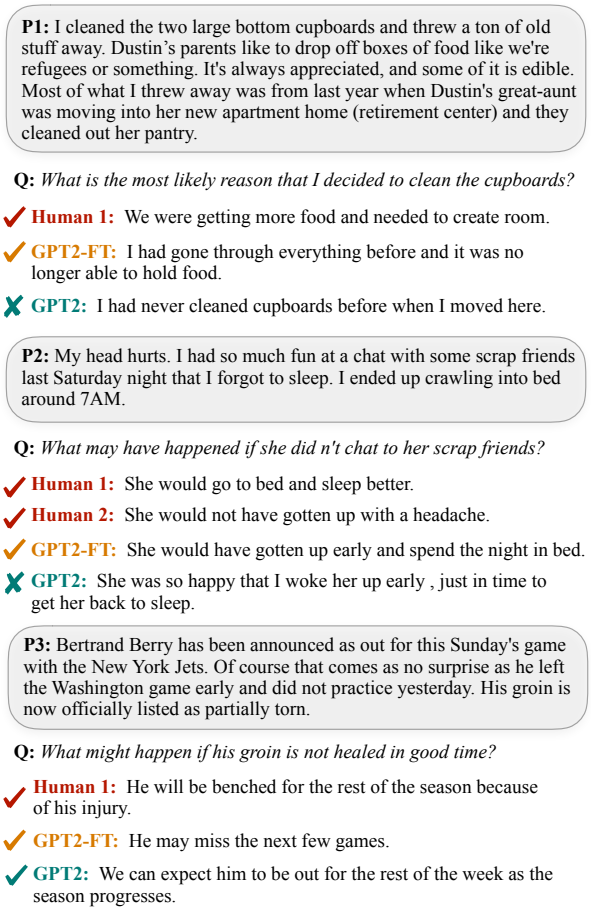


Figure 7: Examples of human authored correct answers, and automatically generated answers by pre-trained GPT2 and GPT2-FT. (✓indicates the answer is correct while ✗shows that the answer is incorrect.)

| Metrics | GPT2 | GPT2-FT |
|---|---|---|
| BLEU (Papineni et al., 2002) | 10.7 | 21.0 |
| METEOR (Banerjee and Lavie, 2005) | 7.2 | 8.6 |
| ROUGE-L (Lin, 2004) | 13.9 | 22.1 |
| CIDEr (Vedantam et al., 2015) | 0.05 | 0.17 |
| BERTScore F1 (Zhang et al., 2019b) | 41.9 | 44.5 |
| Human | 11.0% | 29.0% |

Table 6: Generative performance of pre-trained GPT2 and GPT2-FT on COSMOS QA. All automatic metric scores are averaged from 10 sets of sample output.

## 6 Related Work

There have been many exciting new datasets developed for reading comprehension, such as SQuAD (Rajpurkar et al.,

| Dataset | Size | Type | Answer Type | Paragraph Source | Questions/ Answers | Require MRC | Require Common Sense |
|---|---|---|---|---|---|---|---|
| MCTest | 2K | PQA | MC | MTurk | MTurk | ✓ | - |
| RACE | 100K | PQA | MC | Human Experts | Human Experts | ✓ | - |
| MCScript | 13.9K | PQA | MC | MTurk | MTurk | ✓ | 27.4% |
| NarrativeQA | 46.8K | PQA | Open Text | Books/Movie Scripts | MTurk | ✓ | - |
| ARC | 7.8K | QA | MC | N/A | Web | ✗ | - |
| CommonsenseQA | 12.2K | QA | MC | N/A | MTurk/Web | ✗ | 100% |
| ReCoRD | 121K | PQA | Span | News | Automatic | ✓ | 75.0% |
| COSMOS | 31.8K | PQA | MC | Webblog | MTurk | ✓ | 93.8% |

Table 7: Comparison of the COSMOS QA to other multiple-choice machine reading comprehension datasets: P: contextual paragraph, Q: question, A: answers, MC: Multiple-choice, and - means unknown.

2016), NEWSQA (Trischler et al., 2017), SearchQA (Dunn et al., 2017), NarrativeQA (Kočiskỳ et al., 2018), ProPara (Mishra et al., 2018), CoQA (Reddy et al., 2018), ReCoRD (Zhang et al., 2018), MCTest (Richardson et al., 2013), RACE (Lai et al., 2017), CNN/Daily Mail (Hermann et al., 2015), Children's Book Test (Hill et al., 2015), and MCScript (Ostermann et al., 2018). Most these datasets focus on relatively explicit understanding of the context paragraph, thus a relatively small or unknown fraction of the dataset requires commonsense reasoning, if at all.

A notable exception is ReCoRD (Zhang et al., 2018) that is designed specifically for challenging reading comprehension with commonsense reasoning. COSMOS complements ReCoRD with three unique challenges: (1) our context is from webblogs rather than news, thus requiring commonsense reasoning for *everyday events* rather than *news-worthy events*. (2) All the answers of ReCoRD are contained in the paragraphs and are assumed to be entities. In contrast, in COSMOS, more than 83% of answers are *not* stated in the paragraphs, creating unique modeling challenges. (3) COSMOS can be used for *generative* evaluation in addition to multiple-choice evaluation.

There also have been other datasets focusing specifically on question answering with commonsense, such as CommonsenseQA (Talmor et al., 2018) and Social IQa (Sap et al., 2019), and various other types of commonsense inferences (Levesque et al., 2012; Rahman and Ng, 2012; Gordon, 2016; Rashkin et al., 2018; Roemmele et al., 2011; Mostafazadeh et al., 2017; Zellers et al., 2018). The unique contribution of COSMOS is combining reading comprehension with commonsense reasoning, requiring *contextual* commonsense reasoning over considerably more complex, diverse, and longer context. Table 7 shows comprehensive comparison among the most relevant datasets.

There have been a wide range of attention mechanisms developed for reading comprehension datasets (Hermann et al., 2015; Kadlec et al., 2016; Chen et al., 2016; Dhingra et al., 2017; Seo et al., 2016; Wang et al., 2018b). Our work investigates various state-of-the-art approaches to reading comprehension, and provide empirical insights into the design choices that are the most effective for contextual commonsense reasoning required for COSMOS.

## 7 Conclusion

We introduced COSMOS QA, a large-scale dataset for machine comprehension with contextual commonsense reasoning. We also presented extensive empirical results comparing various state-of-the-art neural architectures to reading comprehension, and demonstrated a new model variant that leads to the best result. The substantial headroom (25.6%) between the best model performance and human encourages future research on contextual commonsense reasoning.

## Acknowledgments

We thank Scott Yih, Dian Yu, Wenpeng Yin, Rowan Zellers for helpful discussions. This research was supported in part by NSF (IIS-1524371, IIS-1714566), DARPA under the CwC program through the ARO (W911NF-15-1-0543), DARPA under the MCS program through NIWC Pacific (N66001-19-2-4031), and Allen Institute for AI.

## References


Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Kevin Burton, Akshay Java, Ian Soboroff, et al. 2009. The icwsm 2009 spinn3r dataset. In *Proceedings of ICWSM 2009*.

Zheng Cai, Lifu Tu, and Kevin Gimpel. 2017. Pay attention to the ending: Strong neural baselines for the roc story cloze task. In *Proceedings of ACL 2017*, pages 616–622.

Danqi Chen, Jason Bolton, and Christopher D Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. In *Proceedings of ACL 2016*, pages 2358–2367.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William Cohen, and Ruslan Salakhutdinov. 2017. Gated-attention readers for text comprehension. In *Proceedings of ACL 2017*, pages 1832–1846.

Matthew Dunn, Levent Sagun, Mike Higgins, V Ugur Guney, Volkan Cirik, and Kyunghyun Cho. 2017. Searchqa: A new q&a dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179*.

Andrew Gordon and Reid Swanson. 2009. Identifying personal stories in millions of weblog entries. In *Third International Conference on Weblogs and Social Media, Data Challenge Workshop*.

Andrew S Gordon. 2016. Commonsense interpretation of triangle behavior. In *Proceedings of AAAI 2016*.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of NAACL 2018*, pages 107–112.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proceedings of NIPS 2015*, pages 1693–1701.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The goldilocks principle: Reading children's books with explicit memory representations. *arXiv preprint arXiv:1511.02301*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of ACL 2018*, pages 328–339.

Rudolf Kadlec, Martin Schmid, Ondřej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. In *Proceedings of ACL 2016*, pages 908–918.

Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gáabor Melis, and Edward Grefenstette. 2018. The narrativeqa reading comprehension challenge. *Proceedings of TACL 2018*, pages 317–328.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. In *Proceedings of EMNLP 2017*, pages 785–794.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Sewon Min, Minjoon Seo, and Hannaneh Hajishirzi. 2017. Question answering through transfer learning from large fine-grained supervision data. In *Proceedings of ACL 2017*, pages 510–517.

Bhavana Dalvi Mishra, Lifu Huang, Niket Tandon, Wen-tau Yih, and Peter Clark. 2018. Tracking state changes in procedural text: a challenge dataset and models for process paragraph comprehension. *arXiv preprint arXiv:1805.06975*.

Nasrin Mostafazadeh, Michael Roth, Annie Louis, Nathanael Chambers, and James Allen. 2017. Lsdsem 2017 shared task: The story cloze test. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, pages 46–51.

Peter Norvig. 1987. *A Unified Theory of Inference for Text Understanding*. Ph.D. thesis, EECS Department, University of California, Berkeley.

Simon Ostermann, Ashutosh Modi, Michael Roth, Stefan Thater, and Manfred Pinkal. 2018. Mcscript: A novel dataset for assessing machine comprehension using script knowledge. In *Proceedings of LREC 2018*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL 2002*, pages 311–318.

Alec Radford, Karthik Narasimhan, Time Salimans, and Ilya Sutskever. 2018. Improving language understanding with unsupervised learning. Technical report, Technical report, OpenAI.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, page 8.

Altaf Rahman and Vincent Ng. 2012. Resolving complex cases of definite pronouns: the winograd schema challenge. In *Proceedings of EMNLP 2012*, pages 777–789.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you dont know: Unanswerable questions for squad. In *Proceedings of ACL 2018*, pages 784–789.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of EMNLP 2016*, pages 2383–2392.

Hannah Rashkin, Antoine Bosselut, Maarten Sap, Kevin Knight, and Yejin Choi. 2018. Modeling naive psychology of characters in simple commonsense stories. In *Proceedings of ACL 2018*, pages 2289–2299.

Siva Reddy, Danqi Chen, and Christopher D Manning. 2018. Coqa: A conversational question answering challenge. *arXiv preprint arXiv:1808.07042*.

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of EMNLP 2013*, pages 193–203.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *2011 AAAI Spring Symposium Series*.

Maarten Sap, Ronan LeBras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A Smith, and Yejin Choi. 2018. Atomic: An atlas of machine commonsense for if-then reasoning. *arXiv preprint arXiv:1811.00146*.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. 2019. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2018. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. Newsqa: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 191–200.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of CVPR 2015*, pages 4566–4575.

Liang Wang, Meng Sun, Wei Zhao, Kewei Shen, and Jingming Liu. 2018a. Yuanfudao at semeval-2018 task 11: Three-way attention and relational knowledge for commonsense machine comprehension. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 758–762.

Shuohang Wang, Mo Yu, Jing Jiang, and Shiyu Chang. 2018b. A co-matching model for multi-choice reading comprehension. In *Proceedings of ACL 2018*, pages 746–751.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. Swag: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of EMNLP 2018*, pages 93–104.

Sheng Zhang, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, Kevin Duh, and Benjamin Van Durme. 2018. Record: Bridging the gap between human and machine commonsense reading comprehension. *arXiv preprint arXiv:1810.12885*.

Shuailiang Zhang, Hai Zhao, Yuwei Wu, Zhuosheng Zhang, Xi Zhou, and Xiang Zhou. 2019a. Dual co-matching network for multi-choice reading comprehension. *arXiv preprint arXiv:1901.09381*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019b. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Haichao Zhu, Furu Wei, Bing Qin, and Ting Liu. 2018. Hierarchical attention flow for multiple-choice reading comprehension. In *Proceedings of AAAI 2018*.

# Finding Generalizable Evidence by Learning to Convince Q&A Models

**Ethan Perez**[†] **Siddharth Karamcheti**[‡]
**Rob Fergus**[†‡] **Jason Weston**[†‡] **Douwe Kiela**[‡] **Kyunghyun Cho**[†‡⋆]
[†]New York University, [‡]Facebook AI Research, [⋆]CIFAR Azrieli Global Scholar
perez@nyu.edu

## Abstract

We propose a system that finds the strongest supporting evidence for a given answer to a question, using passage-based question-answering (QA) as a testbed. We train evidence agents to select the passage sentences that most convince a pretrained QA model of a given answer, if the QA model received those sentences instead of the full passage. Rather than finding evidence that convinces one model alone, we find that agents select evidence that generalizes; agent-chosen evidence increases the plausibility of the supported answer, as judged by other QA models and humans. Given its general nature, this approach improves QA in a robust manner: using agent-selected evidence (i) humans can correctly answer questions with only ∼20% of the full passage and (ii) QA models can generalize to longer passages and harder questions.

## 1 Introduction

There is great value in understanding the fundamental nature of a question (Chalmers, 2015). Distilling the core of an issue, however, is time-consuming. Finding the correct answer to a given question may require reading large volumes of text or understanding complex arguments. Here, we examine if we can automatically discover the underlying properties of problems such as question answering by examining how machine learning models learn to solve that task.

We examine this question in the context of passage-based question-answering (QA). Inspired by work in interpreting neural networks (Lei et al., 2016), we have agents find a subset of the passage (i.e., supporting evidence) that maximizes a QA model's probability of a particular answer. Each agent (one agent per answer) finds the sentences that a QA model regards as strong evidence for its answer, using either exhaustive search or learned prediction. Figure 1 shows an example.

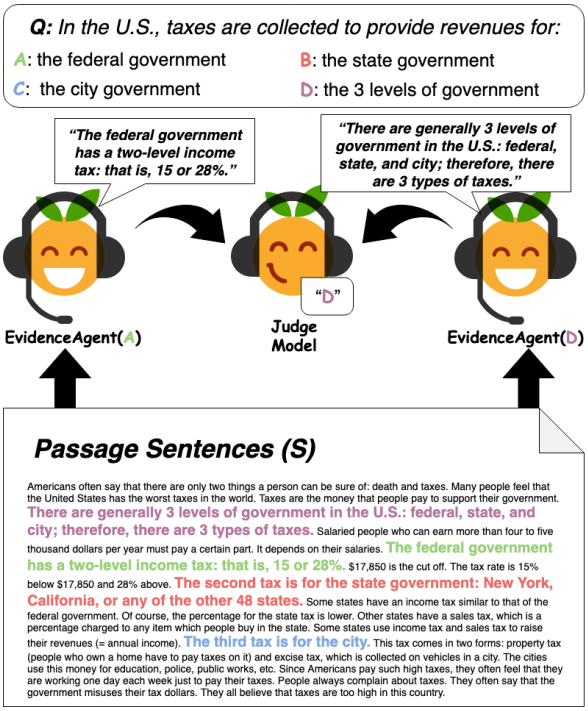


Figure 1: Evidence agents quote sentences from the passage to convince a question-answering judge model of an answer.

To examine to what extent evidence is general and independent of the model, we evaluate if humans and other models find selected evidence to be valid support for an answer too. We find that, when provided with evidence selected by a given agent, both humans and models favor that agent's answer over other answers. When human evaluators read an agent's selected evidence in lieu of the full passage, humans tend to select the agent-supported answer.

Given that this approach appears to capture some general, underlying properties of the problem, we examine if evidence agents can be used to assist human QA and to improve generalization of other QA models. We find that humans can accurately answer questions on QA benchmarks, based



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2402–2411, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

on evidence for each possible answer, using only 20% of the sentences in the full passage. We observe a similar trend with QA models: using only selected evidence, QA models trained on short passages can generalize more accurately to questions about longer passages, compared to when the models use the full passage. Furthermore, QA models trained on middle-school reading comprehension questions generalize better to high-school exam questions by answering only based on the most convincing evidence instead of the full passage. Overall, our results suggest that learning to select supporting evidence by having agents try to convince a judge model of their designated answer improves QA in a general and robust way.

## 2 Learning to Convince Q&A Models

Figure 1 shows an overview of the problem setup. We aim to find the passage sentences that provide the most convincing evidence for each answer option, with respect to a given QA model (the *judge*). To do so, we are given a sequence of passage sentences $S = [S(1), \ldots, S(m)]$, a question $Q$, and a sequence of answer options $A = [A(1), \ldots, A(n)]$. We train a judge model with parameters $\phi$ to predict the correct answer index $i^*$ by maximizing $p_\phi(\text{answer} = i^* | S, Q, A)$.

Next, we assign each answer $A(i)$ to one evidence agent, $\textsc{Agent}(i)$. $\textsc{Agent}(i)$ aims to find evidence $E(i)$, a subsequence of passage sentences $S$ that the judge finds to support $A(i)$. For ease of notation, we use set notation to describe $E(i)$ and $S$, though we emphasize these are ordered sequences. $\textsc{Agent}(i)$ aims to maximize the judge's probability on $A(i)$ when conditioned on $E(i)$ instead of $S$, i.e., $\text{argmax}_{E(i) \subseteq S} p_\phi(i | E(i), Q, A)$. We now describe three different settings of having agents select evidence, which we use in different experimental sections (§4-6).

**Individual Sequential Decision-Making** Since computing the optimal $E(i)$ directly is intractable, a single $\textsc{Agent}(i)$ can instead find a reasonable $E(i)$ by making $T$ sequential, greedy choices about which sentence to add to $E(i)$. In this setting, the agent ignores the actions of the other agents. At time $t$, $\textsc{Agent}(i)$ chooses index $e_{i,t}$ of the sentence in $S$ such that:

$$e_{i,t} = \underset{1 \le e' \le |S|}{\text{argmax}}\, p_\phi(i | \{S(e')\} \cup E(i, t-1), Q, A), \quad (1)$$

where $E(i,t)$ is the subsequence of sentences in $S$ that $\textsc{Agent}(i)$ has chosen until time step $t$, i.e., $E(i,t) = \{S(e_{i,t})\} \cup E(i, t-1)$ with $E(i,0) = \varnothing$ and $E(i) = E(i,T)$. It is a no-op to add a sentence $S(e_{i,t})$ that is already in the selected evidence $E(i, t-1)$. The individual decision-making setting is useful for selecting evidence to support one particular answer.

**Competing Agents: Free-for-All** Alternatively, multiple evidence agents can compete at once to support unique answers, by each contributing part of the judge's total evidence. Agent competition is useful as agents collectively select a pool of question-relevant evidence that may serve as a summary to answer the question. Here, each of $\textsc{Agent}(1), \ldots, \textsc{Agent}(n)$ finds evidence that would convince the judge to select its respective answer, $A(1), \ldots, A(n)$. $\textsc{Agent}(i)$ chooses a sentence $S(e_{i,t})$ by conditioning on all agents' prior choices:

$$e_{i,t} = \underset{1 \le e' \le |S|}{\text{argmax}}\, p_\phi(i | \{S(e')\} \cup E(*, t-1), Q, A),$$

where $E(*, t-1) = \cup_{j=1}^{n} E(j, t-1)$.

Agents simultaneously select a sentence each, doing so sequentially for $t$ time steps, to jointly compose the final pool of evidence. We allow an agent to select a sentence previously chosen by another agent, but we do not keep duplicates in the pool of evidence. Conditioning on other agents' choices is a form of interaction that may enable competing agents to produce a more informative total pool of evidence. More informative evidence may enable a judge to answer questions more accurately without the full passage.

**Competing Agents: Round Robin** Lastly, agents can compete round robin style, in which case we aggregate the outcomes of all $\binom{n}{2}$ pairs of answers $\{A(i), A(j)\}$ competing. Any given $\textsc{Agent}(i)$ participates in $n-1$ rounds, each time contributing half of the sentences given to the judge. In each one-on-one round, two agents select a sentence each at once. They do so iteratively multiple times, as in the free-for-all setup. To aggregate pairwise outcomes and compute an answer $i$'s probability, we average its probability over all rounds involving $\textsc{Agent}(i)$:

$$\frac{1}{n-1} \sum_{j=1}^{n} \mathbb{1}(i \neq j) * p_\phi(i | E(i) \cup E(j), Q, A)$$

### 2.1 Judge Models

The judge model is trained on QA, and it is the model that the evidence agents need to convince. We aim to select diverse model classes, in order to: (i) test the generality of the evidence produced by learning to convince different models; and (ii) to have a broad suite of models to evaluate the agent-chosen evidence. Each model class assigns every answer $A(i)$ a score, where the predicted answer is the one with the highest score. We use this score $L(i)$ as a softmax logit to produce answer probabilities. Each model class computes $L(i)$ in a different manner. In what follows, we describe the various judge models we examine.

**TFIDF** We define a function $\text{BoW}_{\text{TFIDF}}$ that embeds text into its corresponding TFIDF-weighted bag-of-words vector. We compute the cosine similarity of the embeddings for two texts $\mathbf{X}$ and $\mathbf{Y}$:

$$\text{TFIDF}(\mathbf{X}, \mathbf{Y}) = \cos(\text{BoW}_{\text{TFIDF}}(\mathbf{X}), \text{BoW}_{\text{TFIDF}}(\mathbf{Y}))$$

We define two model classes that select the answer most similar to the input passage sentences: $L(i) = \text{TFIDF}(S, [Q; A(i)])$, and $L(i) = \text{TFIDF}(S, A(i))$.

***fastText*** We define a function $\text{BoW}_{\text{FT}}$ that computes the average bag-of-words representation of some text using *fastText* embeddings (Joulin et al., 2017). We use 300-dimensional *fastText* word vectors pretrained on Common Crawl. We compute the cosine similarity between the embeddings for two texts $\mathbf{X}$ and $\mathbf{Y}$ using:

$$\text{fastText}(\mathbf{X}, \mathbf{Y}) = \cos(\text{BoW}_{\text{FT}}(\mathbf{X}), \text{BoW}_{\text{FT}}(\mathbf{Y}))$$

This method has proven to be a strong baseline for evaluating the similarity between two texts (Perone et al., 2018). Using this function, we define a model class that selects the answer most similar to the input passage context: $L(i) = \text{fastText}(S, A(i))$.

**BERT** $L(i)$ is computed using the multiple-choice adaptation of BERT (Devlin et al., 2019; Radford et al., 2018; Si, 2019), a pre-trained transformer network (Vaswani et al., 2017). We fine-tune all BERT parameters during training. This model predicts $L(i)$ using a trainable vector $v$ and BERT's first token embedding: $L(i) = v^\top \cdot \text{BERT}([S; Q; A(i)])$.

We experiment with both the $\text{BERT}_{\text{BASE}}$ model (12 layers) and $\text{BERT}_{\text{LARGE}}$ (24 layers). For training details, see Appendix B.

| ***Predicting*** | **Loss** | **Target** |
|---|---|---|
| Search | *CE* | $S(e_{i,t})$ |
| $p(i)$ | *MSE* | $p_\phi(i \vert \{S(e')\} \cup E(i,t), Q, A)$ |
| $\Delta p(i)$ | *MSE* | $p_\phi(i \vert \{S(e')\} \cup E(i,t), Q, A)$ $-p_\phi(i \vert E(i,t), Q, A)$ |

Table 1: The loss functions and prediction targets for three learned agents. *CE*: cross entropy. *MSE*: mean squared error. $e'$ takes on integer values from 1 to $|S|$.

### 2.2 Evidence Agents

In this section, we describe the specific models we use as evidence agents. The agents select sentences according to Equation 1, either exactly or via function approximation.

**Search agent** $\text{AGENT}(i)$ at time $t$ chooses the sentence $S(e_{i,t})$ that maximizes $p_\phi(i|S(i,t), Q, A)$, after exhaustively trying each possible $S(e_{i,t}) \in S$. Search agents that query TFIDF or fastText models maximize TFIDF or fastText scores directly (i.e., $L(i)$, rather than $p_\phi(i|S(i,t), Q, A)$).

**Learned agent** We train a model to predict how a sentence would influence the judge's answer, instead of directly evaluating answer probabilities at test time. This approach may be less prone to selecting sentences that exploit hard-to-predict quirks in the judge; humans may be less likely to find such sentences to be valid evidence for an answer (discussed in §4.1). We define several loss functions and prediction targets, shown in Table 1. Each forward pass, agents predict one scalar per passage sentence via end-of-sentence token positions. We optimize these predictions using Adam (Kingma and Ba, 2015) on one loss from Table 1. For $t > 1$, we find it effective to simply predict the judge model at $t = 1$ and use this distribution for all time steps during inference. This trick speeds up training by enabling us to precompute prediction targets using the judge model, instead of querying it constantly during training.

We use $\text{BERT}_{\text{BASE}}$ for all learned agents. Learned agents predict the $\text{BERT}_{\text{BASE}}$ judge, as it is more efficient to compute than $\text{BERT}_{\text{LARGE}}$. Each agent $\text{AGENT}(i)$ is assigned the answer $A(i)$ that it should support. We train one learned agent to find evidence for an arbitrary answer $i$. We condition $\text{AGENT}(i)$ on $i$ using a binary indicator when predicting $L(i)$. We add the indicator to BERT's first token segment indicator and embed it

into vectors $\gamma$ and $\beta$; for each timestep's features $f$ from BERT, we scale and shift $f$ element-wise: $(\gamma * f) + \beta$ (Perez et al., 2018; Dumoulin et al., 2018). See Appendix B for training details.

Notably, learning to convince a judge model does not require answer labels to a question. Even if the judge only learns from a few labeled examples, evidence agents can learn to model the judge's behavior on more data and out-of-distribution data without labels.

## 3 Experimental Setup

### 3.1 Evaluating Evidence Agents

**Evaluation Desiderata** An ideal evidence agent should be able to find evidence for its answer w.r.t. a judge, regardless (to some extent) of the specific answer it defends. To appropriately evaluate evidence agents, we need to use questions with more than one defensible, passage-supported answer per question. In this way, an agent's performance will not depend disproportionately on the answer it is to defend, rather than its ability to find evidence.

**Multiple-choice QA: RACE and DREAM** For our experiments, we use RACE (Lai et al., 2017) and DREAM (Sun et al., 2019), two multiple-choice, passage-based QA datasets. Both consist of reading comprehension exams for Chinese students learning English; teachers explicitly designed answer options to be plausible (even if incorrect), in order to test language understanding. Each question has 4 total answer options in RACE and 3 in DREAM. Exactly one option is correct. DREAM consists of 10K informal, dialogue-based passages. RACE consists of 100K formal, written passages (i.e., news, fiction, or well-written articles). RACE also divides into easier, middle school questions (29%) and harder, high school questions (71%).

**Other datasets we considered** Multiple-choice passage-based QA tasks are well-suited for our purposes. Multiple-choice QA allows agents to support clear, dataset-curated possible answers. In contrast, Sugawara et al. (2018) show that 5-20% of questions in extractive, span-based QA datasets have only one valid candidate option. For example, some "when" questions are about passages with only one date. Sugawara et al. argue that multiple-choice datasets such as RACE do not have this issue, as answer candidates are manually created. In preliminary experiments on SQuAD (Rajpurkar et al., 2016), we found that agents could only learn to convince the judge model when supporting the correct answer (one answer per question).

| *Judge Model* | RACE | DREAM |
|---|---|---|
| Random | 25.0 | 33.3 |
| TFIDF$(S, [Q; A])$ | 32.6 | 44.4 |
| TFIDF$(S, A)$ | 31.6 | 44.5 |
| fastText$(S, A)$ | 30.4 | 38.4 |
| $\text{BERT}_{\text{BASE}}$ | 65.4 | 61.0 |
| $\text{BERT}_{\text{LARGE}}$ | 69.4 | 64.9 |
| Human Adult* | 94.5 | 98.6 |

Table 2: RACE and DREAM test accuracy of various judge models using the full passage. Our agents use these models to find evidence. The models cover a spectrum of QA ability. (*) reports ceiling accuracy from original dataset papers.

### 3.2 Training and Evaluating Models

Our setup is not directly comparable to standard QA setups, as we aim to evaluate evidence rather than raw QA accuracy. However, each judge model's accuracy is useful to know for analysis purposes. Table 2 shows model accuracies, which cover a broad range. BERT models significantly outperform word-based baselines (TFIDF and fastText), and $\text{BERT}_{\text{LARGE}}$ achieves the best overall accuracy. No model achieves the estimated human ceiling for either RACE (Lai et al., 2017) or DREAM (Sun et al., 2019).

Our code is available at `https://github.com/ethanjperez/convince`. We build off AllenNLP (Gardner et al., 2018) using PyTorch (Paszke et al., 2017). For all human evaluations, we use Amazon Mechanical Turk via ParlAI (Miller et al., 2017). Appendix B describes preprocessing and training details.

## 4 Agents Select General Evidence

### 4.1 Human Evaluation of Evidence

Would evidence that convinces a model also be valid evidence to humans? On one hand, there is ample work suggesting that neural networks can learn similar patterns as humans do. Convolutional networks trained on ImageNet share similarities with the human visual cortex (Cadieu et al., 2014). In machine translation, attention learns to align foreign words with their native counterparts (Bahdanau et al., 2015). On the other hand, neural networks often do not behave as humans

| | Evidence Sentence Selection Method | *How Often Human Selects Agent's Answer (%)* | | | | | |
|---|---|---|---|---|---|---|---|
| | | ***RACE*** | | | ***DREAM*** | | |
| | | | *Agent Answer is* | | | *Agent Answer is* | |
| | | Overall | Right | Wrong | Overall | Right | Wrong |
| **Baselines** | No Sentence Given | 25.0 | 52.5 | 15.8 | 33.3 | 43.3 | 28.4 |
| | Human Selection | 41.6 | 75.1 | 30.4 | 50.7 | 84.9 | 33.5 |
| **Search Agents querying...** | TFIDF$(S, [Q; A(i)])$ | 33.5 | 69.6 | 21.5 | 41.7 | 68.8 | 28.1 |
| | fastText$(S, A(i))$ | 37.1 | 74.2 | 24.7 | 41.5 | 75.6 | 24.5 |
| | TFIDF$(S, A(i))$ | 38.0 | 71.4 | 26.9 | 43.4 | 75.2 | 27.6 |
| | $\text{BERT}_{\text{BASE}}$ | 38.4 | 68.4 | 28.4 | 50.5 | 82.5 | 34.6 |
| | $\text{BERT}_{\text{LARGE}}$ | 40.1 | 71.0 | 29.9 | 52.3 | 79.4 | 38.7 |
| **Learned Agents predicting...** | Search | 40.0 | 71.0 | 29.7 | 49.1 | 78.3 | 34.6 |
| | $p(i)$ | 42.0 | 74.6 | 31.1 | 50.0 | 77.3 | 36.3 |
| | $\Delta p(i)$ | 41.1 | 73.2 | 30.4 | 48.2 | 76.5 | 34.0 |

Table 3: *Human evaluation*: **Search Agents** select evidence by querying the specified judge model, and **Learned Agents** predict the strongest evidence w.r.t. a judge model ($\text{BERT}_{\text{BASE}}$); humans then answer the question using the selected evidence sentence (without the full passage). Most agents do on average find evidence for their answer, right or wrong. Agents are more effective at supporting right answers.

do. Neural networks are susceptible to adversarial examples—changes to the input which do or do not change the network's prediction in surprising ways (Szegedy et al., 2014; Jia and Liang, 2017; Ribeiro et al., 2018; Alzantot et al., 2018). Convolutional networks rely heavily on texture (Geirhos et al., 2019), while humans rely on shape (Landau et al., 1988). Neural networks trained to recognize textual entailment can rely heavily on dataset biases (Gururangan et al., 2018).

**Human evaluation setup** We use human evaluation to assess how effectively agents select sentences that also make humans more likely to provide a given answer, when humans act as the judge. Humans answer based only on the question $Q$, answer options $A$, and a single passage sentence chosen by the agent as evidence for its answer option $A(i)$ (i.e., using the "Individual Sequential Decision-Making" scheme from §2). Appendix C shows the interface and instructions used to collect evaluations. For each of RACE and DREAM, we use 100 test questions and collect 5 human answers for each $(Q, A(i))$ pair for each agent. We also evaluate a human baseline for this task, where 3 annotators select the strongest supporting passage sentence for each $(Q, A(i))$ pair. We report the average results across 3 annotators.

**Humans favor answers supported by evidence agents** when shown that agent's selected evidence, as shown in Table 3.[1] Without receiving any passage sentences, humans are at random chance at selecting the agent's answer (25% on RACE, 33% on DREAM), since agents are assigned an arbitrary answer. For all evidence agents, humans favor agent-supported answers more often than the baseline (33.5-42.0% on RACE and 41.7-50.5% on DREAM). For our best agents, the relative margin over the baseline is substantial. In fact, these agents select evidence that is comparable to human-selected evidence. For example, on RACE, humans select the target answer 41.6% when provided with human-selected evidence, compared to 42% evidence selected by the learned agent that predicts $p(i)$.

All agents support right answers more easily than wrong answers. On RACE, the learned agent that predicts $p(i)$ finds strong evidence more than twice as often for correct answers than for incorrect ones (74.6% vs. 31.1%). On RACE and DREAM both, BERT-based agents (search or learned agents) find stronger evidence than word-based agents do. Humans tend to find that BERT-based agents select valid evidence for an answer, right or wrong. On DREAM, word-based agents generally fail to find evidence for wrong answers compared to the no-sentence baseline (28.4% vs. 24.5% for a search-based fastText agent).

On RACE, learned agents that predict the $\text{BERT}_{\text{BASE}}$ judge outperform search agents that directly query the $\text{BERT}_{\text{BASE}}$ judge. This effect may occur if search agents find an adversarial sentence that unduly affects the judge's answer but that humans do not find to be valid evidence. Appendix A shows one such example. Learned agents may

[1] Appendix D shows results by question type.

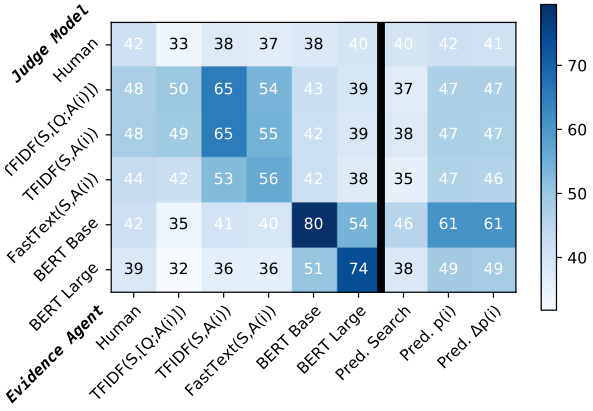


Figure 2: On RACE, how often each judge selects an agent's answer when given a single agent-chosen sentence. The black line divides learned agents (right) and search agents (left), with human evidence selection in the leftmost column. All agents find evidence that convinces judge models more often than a no-evidence baseline (25%). Learned agents predicting $p(i)$ or $\Delta p(i)$ find the most broadly convincing evidence.

have difficulty predicting such sentences, without directly querying the judge. Appendix E provides some analysis on why learned agents may find more general evidence than search agents do. Learned agents are most accurate at predicting evidence sentences when the sentences have a large impact on the judge model's confidence in the target answer, and such sentences in turn are more likely to be found as strong evidence by humans. On DREAM, search agents and learned agents perform similarly, likely because DREAM has 14x less training data than RACE.

### 4.2 Model Evaluation of Evidence

**Evaluating an agent's evidence across models** Beyond human evaluation, we test how general agent-selected evidence is, by testing this evidence against various judge models. We expect evidence agents to most frequently convince the model they are optimized to convince, by nature of their direct training or search objective. The more similar models are, the more we expect evidence from one model to be evidence to another. To some extent, we expect different models to rely on similar patterns to answer questions. Thus, evidence agents should sometimes select evidence that transfers to any model. However, we would not expect agent evidence to transfer to other models if models only exploit method-specific patterns.

**Experimental setup** Each agent selects one evidence sentence for each $(Q, A(i))$ pair. We test how often the judge selects an agent's answer, when given this sentence, $Q$, and $A$. We evaluate on all $(Q, A(i))$ pairs in RACE's test set. Human evaluations are on a 100 question subset of test.

**Results** Figure 2 plots how often each judge selects an agent's answer. Without any evidence, judge models are at random at choosing an agent's assigned answer (25%). All agents find evidence that convinces judge models more often than the no-evidence baseline. Learned agents that predict $p(i)$ or $\Delta p(i)$ find the evidence most broadly considered convincing; other judge models select these agents' supported answers over 46% of the time. These findings support that evidence agents find general structure despite aiming to convince specific methods with their distinct properties.

Notably, evidence agents are not uniformly convincing across judge models. All evidence agents are most convincing to the judge model they aim to convince; across any given agent's row, an agent's target judge model is the model which most frequently selects the agent's answer. Search agents are particularly effective at finding convincing evidence w.r.t. their target judge model, given that they directly query this model. More broadly, similar models find similar evidence convincing. We find similar results for DREAM (Appendix F).

## 5 Evidence Agents Aid Generalization

We have shown that agents capture method-agnostic evidence representative of answering a question (the strongest evidence for various answers). We hypothesize that QA models can generalize better out of distribution to more challenging questions by exploiting evidence agents' capability to understand the problem.

Throughout this section, using various train/test splits of RACE, we train a $\text{BERT}_{\text{BASE}}$ judge on easier examples (involving shorter passages or middle-school exams) and test its generalization to harder examples (involving longer passages or high-school exams). Judge training follows §2.1. We compare QA accuracy when the judge answers using (i) the full passage and (ii) only evidence sentences chosen by competing evidence agents. We report results using the round robin competing agent setup described in §2, as it resulted in higher generalization accuracy than free-for-all competition in preliminary experiments. Each competing agent selects sentences up to a fixed, maximum turn limit; we experiment with 3-6 turns per agent (6-12 total sentences for the judge), and we report the best result. We train learned agents (as de-

| Train Data | Sentence Selection | *RACE* Sentences in Passage ≤ 12 | *RACE* Sentences in Passage ≥ 27 | *→DREAM* Sentences in Passage ≥ 27 |
|---|---|---|---|---|
| **All** | Full Passage | 64.7 | 60.0 | 71.2 |
| **RACE** $\lvert S\rvert \leq 12$ | None (Answer-only) | 36.1 | 40.2 | 38.5 |
| | Full Passage of Subset | 57.4 | 44.1 | 65.0 |
| | Random Sentences | 49.2 | 44.7 | 48.2 |
| | TFIDF$(S, [Q; A(i)])$ | 57.2 | 48.0 | 67.3 |
| | fastText$(S, A(i))$ | **57.7** | **50.2** | 64.2 |
| | TFIDF$(S, A(i))$ | 57.1 | 47.9 | 64.6 |
| | Search over $\text{BERT}_{\text{BASE}}$ | 56.7 | 49.6 | **68.9** |
| | Predict $\text{BERT}_{\text{BASE}}$ $p(i)$ | 56.7 | 50.0 | 66.9 |

Table 4: We train a judge on short RACE passages and test its generalization to long passages. The judge is more accurate on long passages when it answers based on only sentences chosen by competing agents (last 5 rows) instead of the full passage. BERT-based agents aid generalization even under test-time domain shift (from RACE to DREAM).

scribed in §2.2) on the full RACE dataset without labels, so these agents can model the judge using more data and on out-of-distribution data.

For reference, we evaluate judge accuracy on a subsequence of randomly sampled sentences; we vary the number of sentences sampled from 6-12 and report the best result. As a lower bound, we train an answer-only model to evaluate how effectively the QA model is using the passage sentences it is given. As an upper bound, we evaluate our $\text{BERT}_{\text{BASE}}$ judge trained on all of RACE, requiring no out-of-distribution generalization.

### 5.1 Generalizing to Longer Passages

We train a judge on RACE passages averaging 10 sentences long (all training passages each with $\leq$12 sentences); this data is roughly $\frac{1}{10}$th of RACE. We test the judge on RACE passages averaging 30 sentences long.

**Results** Table 4 shows the results. Using the full passage, the judge outperforms an answer-only BERT baseline by 4% (44.1% vs. 40.2%). When answering using the smaller set of agent-chosen sentences, the judge outperforms the baseline by 10% (50.2% vs. 40.2%), more than doubling its relative use of the passage. Both search and learned agents aid the judge model in generalizing to longer passages. The improved generalization is not simply a result of the judge using a shorter passage, as shown by the random sentence selection baseline (44.7%).

| Train Data | Sentence Selection | *School Level* Middle | *School Level* High |
|---|---|---|---|
| **All** | Full Passage | 70.8 | 63.2 |
| **Middle School only** | None (Answer-only) | 38.9 | 40.2 |
| | Full Passage of Subset | 66.2 | 50.7 |
| | Random Sentences | 54.8 | 47.0 |
| | TFIDF$(S, [Q; A(i)])$ | 65.1 | 50.4 |
| | fastText$(S, A(i))$ | 64.6 | 50.8 |
| | TFIDF$(S, A(i))$ | 64.9 | 51.0 |
| | Search over $\text{BERT}_{\text{BASE}}$ | 67.0 | **53.0** |
| | Predict $\text{BERT}_{\text{BASE}}$ $p(i)$ | **67.3** | 51.9 |

Table 5: *Generalizing to harder questions*: We train a judge to answer questions with RACE's Middle School exam questions only. We test its generalization to High School exam questions. The judge is more accurate when using evidence agent sentences (last 5 rows) rather than the full passage.

### 5.2 Generalizing Across Domains

We examine if evidence agents aid generalization even in the face of domain shift. We test the judge trained on short RACE passages on long passages from *DREAM*. We use the same evidence agents from the previous subsection; the learned agent is trained on RACE only, and we do not fine-tune it on DREAM to test its generalization to finding evidence in a new domain. DREAM passages consist entirely of dialogues, use more informal language and shorter sentences, and emphasize general world knowledge and commonsense reasoning (Sun et al., 2019). RACE passages are more formal, written articles (e.g. news or fiction).

**Results** Table 4 shows that BERT-based evidence agents aid generalization even under domain shift. The model shows notable improvements for RACE → DREAM transfer when it predicts from BERT-based agent evidence rather than the full passage (65.0% vs. 68.9%). These results support that our best evidence agents capture something fundamental to the problem of QA, despite changes in e.g. content and writing style.

### 5.3 Generalizing to Harder Questions

Using RACE, we train a judge on middle-school questions and test it on high-school questions.

**Results** Table 5 shows that the judge generalizes to harder questions better by using evidence from either search-based BERT agents (53.0%) or learned BERT agents (51.9%) compared to using the full passage directly (50.7%) or to search-based TFIDF and fastText agents (50.4%-51.0%). Figure 3 shows that the improved generalization comes from questions the model originally gener-

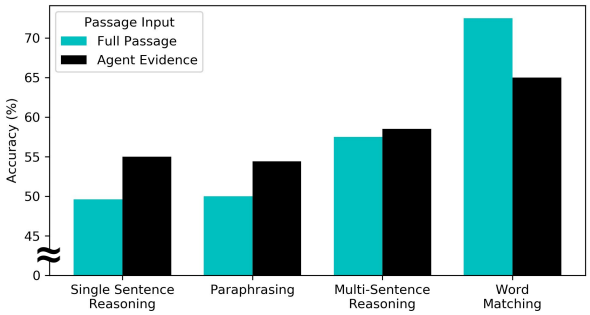


Figure 3: *Generalizing to harder questions by question type*: We train a judge on RACE Middle School questions and test its generalization to RACE High School questions. To predict the answer, the judge uses either the full passage or evidence sentences chosen by a BERT-based search agent. The worse the judge does on a question category using the full passage, the better it does when using the agent-chosen sentences.

alizes worse on. Simplifying the passage by providing key sentences may aid generalization by e.g. removing extraneous or distracting sentences from passages with more uncommon words or complex sentence structure. Such improvements come at the cost of accuracy on easier, word-matching questions, where it may be simpler to answer with the full passage as seen in training.

## 6 Evidence Agents Aid Human QA

As observed in §4.1, evidence agents more easily support right answers than wrong ones. Furthermore, evidence agents do aid QA models in generalizing systematically when all answer evidence sentences are presented at once. We hypothesize that when we combine all evidence sentences, humans prefer to choose the correct answer.

**Human evaluation setup** Evidence agents compete in a free-for-all setup (§2), and the human acts as the judge. We evaluate how accurately humans can answer questions based only on agent sentences. Appendix C shows the annotation interface and instructions. We collect 5 human answers for each of the 100 test questions.

**Humans can answer using evidence sentences alone** Shown in Table 6, humans correctly answer questions using many fewer sentences (3.3 vs. 18.2 on RACE, 2.4 vs. 12.2 on DREAM); they do so while maintaining 90% of human QA accuracy on the full passage (73.2% vs. 82.3% on RACE, 83.8% vs. 93.0% on DREAM). Evidence agents, however, vary in how effectively they aid human QA, compared to answer-agnostic evidence selection. On DREAM, humans answer with 79.1% accuracy using the sentences

| **Sentences Shown** | | ***Human Acc. (%)*** | |
|---|---|---|---|
| *Selection Type* | Selection Method | **RACE** | **DREAM** |
| **Full Passage** | Full Passage | 82.3 | 93.0 |
| **No Passage** | Answer-only | 52.5 | 43.3 |
| **Subset (~20%)** | Human Selection | **73.5** | 82.3 |
| *Answer-Free* | First $n$ Sentences | 61.8 | 68.5 |
| *Selection* | TFIDF$(S, Q)$ | 69.2 | 77.5 |
| | fastText$(S, Q)$ | 69.7 | 79.1 |
| *Search Agent* | TFIDF$(S, [Q; A(i)])$ | 66.1 | 70.0 |
| *Selection* | TFIDF$(S, A(i))$ | 73.2 | 77.0 |
| | fastText$(S, A(i))$ | 73.2 | 77.3 |
| | $\text{BERT}_{\text{BASE}}$ | 69.9 | **83.8** |
| | $\text{BERT}_{\text{LARGE}}$ | 72.4 | 75.0 |
| *Learned Agent* | Predicting Search | 66.5 | 80.0 |
| *Selection* | Predicting $p(i)$ | 71.6 | 77.8 |
| | Predicting $\Delta p(i)$ | 65.7 | 81.5 |

Table 6: *Human accuracy using evidence agent sentences*: Each agent selects a sentence supporting its own answer. Humans answer the question given these agent-selected passage sentences only. Humans still answer most questions correctly, while reading many fewer passage sentences.

most similar to the question alone (via fastText), while achieving lower accuracy when using the $\text{BERT}_{\text{LARGE}}$ search agent's evidence (75.0%) and higher accuracy when using the $\text{BERT}_{\text{BASE}}$ search agent's evidence (83.8%). We explain the discrepancy by examining how effective agents are at supporting right vs. wrong answers (Table 3 from §4.1); $\text{BERT}_{\text{BASE}}$ is more effective than $\text{BERT}_{\text{LARGE}}$ at finding evidence for right answers (82.5% vs. 79.4%) and less effective at finding evidence for wrong answers (34.6% vs. 38.7%).

## 7 Related Work

Here, we discuss further related work, beyond that discussed in §4.1 on (dis)similarities between patterns learned by humans and neural networks.

**Evidence Extraction** Various papers have explored the related problem of extracting evidence or summaries to aid downstream QA. Wang et al. (2018a) concurrently introduced a neural model that extracts evidence specifically for the correct answer, as an intermediate step in a QA pipeline. Prior work uses similar methods to explain what a specific model has learned (Lei et al., 2016; Li et al., 2016; Yu et al., 2019). Others extract evidence to improve downstream QA efficiency over large amounts of text (Choi et al., 2017; Kratzwald and Feuerriegel, 2019; Wang et al., 2018b). More broadly, extracting evidence can facilitate fact verification (Thorne et al., 2018) and debate.[2]

[2]IBM Project Debater: www.research.ibm.com/artificial-intelligence/project-debater

**Generic Summarization** In contrast, various papers focus primarily on summarization rather than QA, using downstream QA accuracy only as a reward to optimize generic (question-agnostic) summarization models (Arumae and Liu, 2018, 2019; Eyal et al., 2019).

**Debate** Evidence extraction can be viewed as a form of debate, in which multiple agents support different stances (Irving et al., 2018; Irving and Askell, 2019). Chen et al. (2018) show that evidence-based debate improves the accuracy of crowdsourced labels, similar to our work which shows its utility in natural language QA.

## 8 Conclusion

We examined if it was possible to automatically distill general insights for passage-based question answering, by training evidence agents to convince a judge model of any given answer. Humans correctly answer questions while reading only 20% of the sentences in the full passage, showing the potential of our approach for assisting humans in question answering tasks. We examine how selected evidence affects the answers of humans as well as other QA models, and we find that agent-selected evidence is generalizable. We exploit these capabilities by employing evidence agents to facilitate QA models in generalizing to longer passages and out-of-distribution test sets of qualitatively harder questions.

## Acknowledgments

EP was supported by the NSF Graduate Research Fellowship and ONR grant N00014-16-1-2698. KC thanks support from eBay and NVIDIA. We thank Adam Gleave, David Krueger, Geoffrey Irving, Katharina Kann, Nikita Nangia, and Sam Bowman for helpful conversations and feedback. We thank Jack Urbanek, Jason Lee, Ilia Kulikov, Ivanka Perez, Ivy Perez, and our Mechanical Turk workers for help with human evaluations.

## References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In EMNLP, pages 2890–2896.

Kristjan Arumae and Fei Liu. 2018. Reinforced extractive summarization with question-focused rewards. In ACL, Student Research Workshop, pages 105–111.

Kristjan Arumae and Fei Liu. 2019. Guiding extractive summarization with question-answering rewards. In NAACL, pages 2566–2577.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In ICLR.

Charles F. Cadieu, Ha Hong, Daniel L. K. Yamins, Nicolas Pinto, Diego Ardila, Ethan A. Solomon, Najib J. Majaj, and James J. DiCarlo. 2014. Deep neural networks rival the representation of primate it cortex for core visual object recognition. PLOS Computational Biology, 10(12):1–18.

David J. Chalmers. 2015. Why isn't there more progress in philosophy? Philosophy, 90(1):331.

Quanze Chen, Jonathan Bragg, Lydia B Chilton, and Daniel S Weld. 2018. Cicero: Multi-turn, contextual argumentation for accurate crowdsourcing. CoRR, abs/1810.10733.

Eunsol Choi, Daniel Hewlett, Jakob Uszkoreit, Illia Polosukhin, Alexandre Lacoste, and Jonathan Berant. 2017. Coarse-to-fine question answering for long documents. In ACL, pages 209–220.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In NAACL, pages 4171–4186.

Vincent Dumoulin, Ethan Perez, Nathan Schucher, Florian Strub, Harm de Vries, Aaron Courville, and Yoshua Bengio. 2018. Feature-wise transformations. Distill.

Matan Eyal, Tal Baumel, and Michael Elhadad. 2019. Question answering as an automatic evaluation metric for news article summarization. In NAACL, pages 3938–3948.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew E. Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. CoRR, abs/1803.07640.

Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. 2019. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In ICLR.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In NAACL, pages 107–112.

Geoffrey Irving and Amanda Askell. 2019. AI safety needs social scientists. Distill.

Geoffrey Irving, Paul Christiano, and Dario Amodei. 2018. AI safety via debate. CoRR, abs/1805.00899.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In EMNLP, pages 2021–2031.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In EACL, pages 427–431.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In ICLR.

Bernhard Kratzwald and Stefan Feuerriegel. 2019. Learning from on-line user feedback in neural question answering on the web. In WWW, pages 906–916.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale ReAding Comprehension dataset from Examinations. In EMNLP, pages 785–794.

Barbara Landau, Linda B. Smith, and Susan S. Jones. 1988. The importance of shape in early lexical learning. Cognitive Development, 3(3):299–321.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. In EMNLP, pages 107–117.

Jiwei Li, Will Monroe, and Daniel Jurafsky. 2016. Understanding neural networks through representation erasure. CoRR, abs/1612.08220.

Alexander H. Miller, Will Feng, Adam Fisch, Jiasen Lu, Dhruv Batra, Antoine Bordes, Devi Parikh, and Jason Weston. 2017. Parlai: A dialog research software platform. CoRR, abs/1705.06476.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In NIPS-W.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. 2018. Film: Visual reasoning with a general conditioning layer. In AAAI, pages 3942–3951.

Christian S. Perone, Roberto Silveira, and Thomas S. Paula. 2018. Evaluation of sentence embeddings in downstream and linguistic probing tasks. CoRR, abs/1806.06259.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In EMNLP, pages 2383–2392.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Semantically equivalent adversarial rules for debugging NLP models. In ACL, pages 856–865.

M. Schuster and K. Nakajima. 2012. Japanese and korean voice search. In ICASSP, pages 5149–5152.

Chenglei Si. 2019. Bert for multiple choice machine comprehension.

Saku Sugawara, Kentaro Inui, Satoshi Sekine, and Akiko Aizawa. 2018. What makes reading comprehension questions easier? In EMNLP, pages 4208–4219.

Kai Sun, Dian Yu, Jianshu Chen, Dong Yu, Yejin Choi, and Claire Cardie. 2019. Dream: A challenge dataset and models for dialogue-based reading comprehension. TACL, 7:217–231.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. CoRR, abs/1312.6199.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and VERification. In NAACL, pages 809–819.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, NeurIPS, pages 5998–6008. Curran Associates, Inc.

Hai Wang, Dian Yu, Kai Sun, Jianshu Chen, Dong Yu, Dan Roth, and David McAllester. 2018a. Evidence extraction for machine reading comprehension with deep probabilistic logic. CoRR, abs/1902.08852.

Shuohang Wang, Mo Yu, Jing Jiang, Wei Zhang, Xiaoxiao Guo, Shiyu Chang, Zhiguo Wang, Tim Klinger, Gerald Tesauro, and Murray Campbell. 2018b. Evidence aggregation for answer re-ranking in open-domain question answering. In ICLR.

Mo Yu, Shiyu Chang, and Tommi S Jaakkola. 2019. Learning corresponded rationales for text matching.

# Ranking and Sampling in Open-Domain Question Answering

**Yanfu Xu**[1,2], **Zheng Lin**[1]*, **Yuanxin Liu**[1,2], **Rui Liu**[1,2], **Weiping Wang**[1] **and Dan Meng**[1]
[1]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
{xuyanfu,linzheng,liuyuanxin,liurui1995,wangweiping,mengdan}@iie.ac.cn

## Abstract

Open-domain question answering (OpenQA) aims to answer questions based on a number of unlabeled paragraphs. Existing approaches always follow the distantly supervised setup where some of the paragraphs are wrong-labeled (noisy), and mainly utilize the paragraph-question relevance to denoise. However, the paragraph-paragraph relevance, which may aggregate the evidence among relevant paragraphs, can also be utilized to discover more useful paragraphs. Moreover, current approaches mainly focus on the positive paragraphs which are known to contain the answer during training. This will affect the generalization ability of the model and make it be disturbed by the similar but irrelevant (distracting) paragraphs during testing. In this paper, we first introduce a ranking model leveraging the paragraph-question and the paragraph-paragraph relevance to compute a confidence score for each paragraph. Furthermore, based on the scores, we design a modified weighted sampling strategy for training to mitigate the influence of the noisy and distracting paragraphs. Experiments on three public datasets (Quasar-T, SearchQA and TriviaQA) show that our model advances the state of the art.

## 1 Introduction

Different from the traditional reading comprehension (RC) task, which aims to answer a question based on an off-the-peg paragraph, the reading paragraphs on OpenQA datasets are always collected via an information retrieval (IR) system. For example, given a question as shown in Figure 1, an OpenQA system usually coarsely retrieves paragraphs that are similar to the question. Therefore, the OpenQA model has to answer a question based on numerous paragraphs. As can be seen in Figure 1, some of the negative paragraphs are similar to the question whereas the answer string is missed, e.g. Paragraph4. We consider these paragraphs as distracting ones. Besides, in the distantly supervised setup (Mintz et al., 2009), it is postulated that the paragraphs that contain the answer string are ground truths, while even some of the positive paragraphs are wrong-labeled as they do not concern the question, e.g. Paragraph3. We consider these paragraphs as noisy ones. As a result, only Paragraph1 and Paragraph2 provide relevant answer to the question and the distracting and noisy paragraphs will prevent the model from identifying the correct answer.

**Question**: Who is Donald Duck's uncle ?
**Answer**: Scrooge
**Paragraph1 (positive, relevant)**: Donald's maternal uncle Scrooge McDuck made his first appearance ...
**Paragraph2 (positive, relevant)**: This story was reissued as a single record entitled "Donald Duck and Uncle Scrooge's Money Rocket" ...
**Paragraph3 (positive, noisy)**: In Scrooge's stories, Donald Duck was cast as Scrooge's assistant ...
**Paragraph4 (negative, distracting)**: Donald Duck is the uncle of Huey, Dewey, and Louie.

Figure 1: An example of OpenQA. The key information, answers in correct-labeled and wrong-labeled contexts are marked in blue, green and red respectively.

Recent OpenQA approaches always follow the retrieve-then-read pipeline and can be roughly divided into two categories: single-paragraph approaches (Joshi et al., 2017; Wang et al., 2018a) and multi-paragraph approaches (Chen et al., 2017; Clark and Gardner, 2018; Pang et al., 2019). Single-paragraph approaches mainly focus on the most relevant paragraph during reading. Multi-paragraph approaches apply a RC model to multiple paragraphs to extract the final answer. However, these approaches still face two major issues.

First, as described above, recent methods only consider the paragraph-question relevance in selecting paragraphs but neglect the paragraph-

*Corresponding author



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2412–2421, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

paragraph relevance, which can be exploited to associate relevant paragraphs that are useful to the downstream RC task. Second, some previous approaches only take into account the positive paragraphs, which are known to contain the answer string, at the training stage. This will make the model become too confident in heuristics or patterns that are only effective in positive paragraphs. As a result, the model will suffer from the problem of impaired generalization ability and be easily bewildered by the distracting paragraphs during testing. Such phenomenon has also been observed by Clark and Gardner (2018). Therefore, a more carefully-designed training strategy is needed.

To address the two issues, we first propose an attention based ranking model which utilizes both the question-paragraph and the paragraph-paragraph relevance to compute a more accurate confidence score for each paragraph. Through the multi-level attention mechanism, the model can utilize the word-level and sentence-level information between the paragraph and the question. Through the sentence-level self-attention mechanism, the model can aggregate the effective evidence among relevant paragraphs and thus increase their confidence scores.

Second, based on the confidence scores, we design a modified weighted sampling strategy to select training paragraphs, which simultaneously ameliorate the influence of the distracting and noisy paragraphs. Through "sampling", the model can be prevented from being too confident in heuristics that are only effective in positive paragraphs. Through "weighted", the model can be less affected by the noisy paragraphs. After that, we concatenate the selected paragraphs and feed them to a RC model to predict the final answer.

We evaluate our work on three distantly supervised OpenQA datasets including Quasar-T (Dhingra et al., 2017b), SearchQA (Dunn et al., 2017) and TriviaQA (Joshi et al., 2017). Empirical results show that our model is effective in improving the performance of OpenQA and advances the state of the art on all three datasets. We additionally perform an ablation study on our model to give insights into the contribution of each submodule. We will release our code on GitHub for further research explorations. [1]

[1] https://github.com/xuyanfu/RASAOpenQA

## 2 Methodology

### 2.1 Task Definition

OpenQA aims to answer a question based on a number of paragraphs retrieved by IR systems. Formally, given a question containing $m$ words $Q = \{q_1, q_2, ..., q_m\}$ and a set of retrieved paragraphs $D = \{P_1, P_2, ..., P_n\}$, where $P_i = \{p_i^1, p_i^2, ..., p_i^{|p_i|}\}$ is the i-th paragraph, $|p_i|$ is the number of words in the i-th paragraph, our model is supposed to extract the answer from the paragraph collection $D$.

### 2.2 Two-stage Framework

Figure 2 gives an overview of our OpenQA model which is composed of two modules including a **ranker** and a **reader**. The ranker is responsible to compute a confidence score for each paragraph. Based on the confidence scores, we design five strategies to select $k$ paragraphs, which are then concatenated and passed to the reader. Note that $k$ is a hyperparameter. Finally, a sub-phrase of the concatenated paragraph is predicted as the answer according to reader's output, which is the probability distribution of the start and end positions.

### 2.3 Paragraph Ranker

We adopt a ranker to produce a confidence score for each paragraph. The ranker is comprised of an encoding layer, a word-level matching layer, a sentence-level matching layer and a sentence-level decoding layer.

#### Encoding Layer

Given a paragraph $P_i = \{p_i^1, p_i^2, ..., p_i^{|p_i|}\}$ and a question $Q = \{q_1, q_2, ..., q_m\}$, we map each word into a vector by combining the following features:

**Word Embeddings:** We use pre-trained word vectors, GloVe (Pennington et al., 2014), to obtain the fixed embedding of each word.

**Char Embeddings:** We map the characters in a word into 20-dimensional vectors which are then passed to a convolutional layer and a max pooling layer to obtain a fixed-size vector of the word.

**Common word:** The feature is set to 1 if the word appears in both the question and the paragraph, and otherwise it is set to 0. Then the feature is mapped into a vector of fixed size.

By concatenating the above features, we obtain the encoded sequence of the paragraph $\mathbf{P}_i^{emb} = \{\mathbf{p}_i^t\}_{t=1}^{|p_i|} \in \mathbb{R}^{d_{emb} \times |p_i|}$ and the question $\mathbf{Q}^{emb} = \{\mathbf{q}^t\}_{t=1}^{m} \in \mathbb{R}^{d_{emb} \times m}$ , where $|p_i|$ is the number

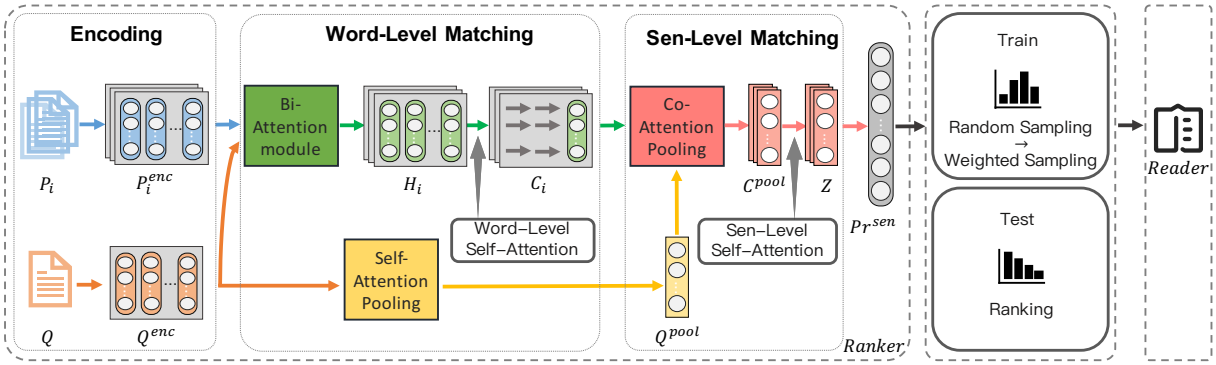


Figure 2: The framework of out model, which consists of a paragraph ranker (left) and a paragraph reader (right). The ranker first computes a confidence score for each paragraph and $k$ paragraphs are selected through the ranking and sampling strategies. Then, the reader concatenates the selected paragraphs together and computes the start score and the end score for each word in the concatenated paragraph.

of words in the i-th paragraph. After that, a bi-directional LSTM is used to calculate a contextual encoding for each word in the paragraph and the question respectively:

$$\mathbf{P}_i^{enc} = BiLSTM(\mathbf{P}_i^{emb}) \in \mathbb{R}^{d_{hid} \times |p_i|} \quad (1)$$

$$\mathbf{Q}^{enc} = BiLSTM(\mathbf{Q}^{emb}) \in \mathbb{R}^{d_{hid} \times m} \quad (2)$$

### Word-level Matching Layer

The word-level matching layer takes $\mathbf{P}_i^{enc}$ and $\mathbf{Q}^{enc}$ as inputs and produces a new question-aware representation $\mathbf{C}_i$ for each paragraph. Following Clark and Gardner (2018), we first perform bi-directional attention between the question $\mathbf{Q}^{enc}$ and the paragraph $\mathbf{P}_i^{enc}$, which gives rise to $\mathbf{H}_i$ for $\mathbf{P}_i^{enc}$, where $\mathbf{H}_i \in \mathbb{R}^{d_{hid} \times |p_i|}$. Next, we feed $\mathbf{H}_i$ to the self-attention layer and obtain $\mathbf{E}_i \in \mathbb{R}^{d_{hid} \times |p_i|}$. We strictly implement the steps as suggested by Clark and Gardner (2018). After that, we use a bi-directional LSTM to obtain a question-aware paragraph embedding for the paragraph $P_i$:

$$\mathbf{C}_i = BiLSTM(\mathbf{E}_i) \in \mathbb{R}^{d_{hid} \times |p_i|} \quad (3)$$

### Sentence-level Matching Layer

Given $\mathbf{D}_c = \{\mathbf{C}_1, \mathbf{C}_2, ..., \mathbf{C}_n\}$ and $\mathbf{Q}^{enc}$, the sentence-level matching layer produces a sentence-level representation $\mathbf{Z}_i$ for each paragraph by effectively exploiting the paragraph-question and the paragraph-paragraph relevance. Firstly, we use self-attention pooling to obtain a fixed-size vector for the encoded question representation. Each position in the encoded question representation is assigned with a score computed using a two-layer multi-layer perception (MLP). This score is normalized and used to compute a weighted sum over the columns of the question representation:

$$\alpha = softmax(\mathbf{W}_2^T tanh(\mathbf{W}_1 \mathbf{Q}^{enc})) \in \mathbb{R}^m \quad (4)$$

$$\mathbf{Q}^{pool} = \sum_t \alpha_t \mathbf{Q}_{:t}^{enc} \in \mathbb{R}^{d_{hid}} \quad (5)$$

Here, $\alpha_t$ is the normalized score for the t-th word in the question and $\mathbf{W}_1 \in \mathbb{R}^{d_{hid} \times d_{hid}}$, $\mathbf{W}_2 \in \mathbb{R}^{d_{hid}}$ are trainable parameters for the MLP. Next, we utilize the co-attention pooling (CAP) to get the fixed-length summary vector for each paragraph. The score for each position in $\mathbf{C}_i$ is calculated by measuring the similarity between $\mathbf{C}_i$ and $\mathbf{Q}^{pool}$, based on which we can compute a weighted sum over the columns of the question-aware paragraph embedding $\mathbf{C}_i$:

$$\hat{\alpha} = softmax(\mathbf{Q}_{pool}^T \mathbf{C}_i) \in \mathbb{R}^{|p_i|} \quad (6)$$

$$\mathbf{C}_i^{pool} = \sum_t \hat{\alpha}_t \mathbf{C}_{i:t} \in \mathbb{R}^{d_{hid}} \quad (7)$$

$$\mathbf{C}^{pool} = \{\mathbf{C}_i^{pool}\}_{i=1}^n \in \mathbb{R}^{d_{hid} \times n} \quad (8)$$

where $\hat{\alpha}_t$ is the normalized score for the t-th position in $\mathbf{C}_i$. Let $\mathbf{C}^{pool}$ represent the sequence of summary vectors for all paragraphs. Afterwards, in order to make full use of the paragraph-paragraph relevance, we use the sentence-level self-attention (SSA) and a bi-directional LSTM to produce the sentence-level paragraph representation $\mathbf{Z}$. The SSA is calculated as:

$$\mathbf{A} = softmax(\mathbf{C}^{pool^T} \mathbf{C}^{pool}) \in \mathbb{R}^{n \times n} \quad (9)$$

$$\mathbf{U} = \mathbf{C}^{pool} \mathbf{A} \in \mathbb{R}^{d_{hid} \times n} \quad (10)$$

Here, A is the similarity matrix and $A_{ij}$ indicates the similarity between $\mathbf{C}_{:i}^{pool}$ and $\mathbf{C}_{:j}^{pool}$. $\mathbf{U}_{:i}$

is the i-th column of $\mathbf{U}$ which is the representation of the i-th paragraph where information of all the retrieved paragraphs is encompassed. $\mathbf{U}$ and $\mathbf{C}^{pool}$ are concatenated together to yield $\mathbf{G}$, which is defined by

$$\mathbf{G}_{:i} = [\mathbf{C}^{pool}_{:i}; \mathbf{U}_{:i}; \mathbf{C}^{pool}_{:i} \circ \mathbf{U}_{:i}; \mathbf{C}^{pool}_{:i} - \mathbf{U}_{:i}] \in \mathbb{R}^{4d_{hid}} \tag{11}$$

where $\circ$ is element-wise multiplication, $-$ is element-wise subtraction, and $[;]$ is vector concatenation across rows. After that, a bi-directional LSTM is used to obtain the final sentence-level paragraph representation $\mathbf{Z}$:

$$\mathbf{Z} = BiLSTM(\mathbf{G}) \in \mathbb{R}^{d_{hid} \times n} \tag{12}$$

**Sentence-level Decoding Layer**

The sentence-level decoding layer, which consists of a linear layer and a sotfmax function, will produce a normalized score for each paragraph:

$$\mathbf{Pr}^{sen} = softmax(W_3^T \mathbf{Z}) \in \mathbb{R}^n \tag{13}$$

where $W_3 \in \mathbb{R}^{d_{hid}}$ is a trainable weight vector. $\mathbf{Pr}^{sen}_i$ is the i-th element of $\mathbf{Pr}^{sen}$ which represents the probability that the i-th paragraph contains the answer.

### 2.4 Ranking and Sampling Strategies

Based on the retrieved paragraphs and their confidence scores, we design five strategies to select paragraphs for the reader.

**Ground Truth (GT)** strategy simply chooses all positive paragraphs which contain the answer string as reading datasets and filters out the negative ones. Since we have to know which paragraph contains the answer string, this strategy can only be used while constructing the training set.

**Random Sampling (RS)** strategy samples $k$ different paragraphs from all the retrieved paragraphs based on the uniform probability distribution. Different from the GT strategy, this strategy will include paragraphs, which do not contain an answer, in the training set. As a result, the RC model is prevented from becoming too confident in heuristics or patterns that are only effective when it is guaranteed that an answer string exists.

**Ranking (RK)** strategy selects $k$ top-ranked paragraphs based on the confidence scores produced by the ranker. In this strategy, the ranker is expected to select more relevant paragraphs, providing a better start point for the reader to predict the final answer.

**Weighted Sampling (WS)** strategy samples $k$ different paragraphs from all retrieved paragraphs based on the probability distribution generated by the ranker. This strategy is an improved version of RK. It hopes to consider the effects of the distracting paragraphs while filtering out the noisy ones.

**RS→WS** strategy is a combination of RS and WS. In the training process, in order to make full use of the dataset and mitigate the impact of the distracting paragraphs, we generate the initial training set for the reader via RS. After that, WS is applied to select paragraphs with higher accuracy, which reduces the influence of the distracting and noisy paragraphs.

During training, we can utilize all the five strategies to generate reading data, while in testing, considering that the paragraphs should be as clean as possible, we only use RK for paragraph selection. After that, we feed the selected paragraphs to the reader to generate the final answer.

### 2.5 Paragraph Reader

The reader is a traditional RC model, where the inputs are the $k$ selected paragraphs $D^{top} = \{P_1, P_2, ..., P_k\}$ and we regard $k$ as a hyperparameter. We concatenate these paragraphs together to obtain $\hat{P} = [P_1; P_2; ...; P_k]$ which aggregates the information from the selected paragraphs in word level. Given $\hat{P}$ and $Q$, the output of the reader, i.e, $\mathbf{Pr}^s, \mathbf{Pr}^e \in \mathbb{R}^{|\hat{P}|}$ , are the probability distributions of the start index and the end index over the concatenated paragraph, where $|\hat{P}|$ is the number of words in the concatenated paragraph. Our reader is the same as in Clark and Gardner (2018) and we do not give the details here due to the space limitation. After that, we split $\mathbf{Pr}^s$ and $\mathbf{Pr}^e$ into $k$ vectors according to the length of each paragraph and obtain $\{\mathbf{Pr}^s_1, \mathbf{Pr}^s_2, ..., \mathbf{Pr}^s_k\}$ and $\{\mathbf{Pr}^e_1, \mathbf{Pr}^e_2, ..., \mathbf{Pr}^e_k\}$, where $\mathbf{Pr}^s_i$/ $\mathbf{Pr}^e_i$ represent the start/end probabilities of the i-th paragraph.

### 2.6 Training and Testing

Our model is trained in two stages. We follow the distantly supervised setup that all the paragraphs that contain the answer span are ground truths.

For the ranker, each paragraph in $\{P_i\}_{i=1}^n$ is associated with a label $\mathbf{y}_i \in \{0, 1\}$, and $\mathbf{y}_i$ equals to 1 if the paragraph contains the answer string. We follow the S-Net (Tan et al., 2018) to design the loss function of the ranker. Given the paragraph

probabilities $\mathbf{Pr}^{sen} \in \mathbb{R}^n$, the ranker is trained by minimizing the loss:

$$\mathcal{L}_s = -\sum_{i=1}^{n}(\mathbf{y}_i log(\mathbf{Pr}_i^{sen}) + (1-\mathbf{y}_i) log(1-\mathbf{Pr}_i^{sen})) \tag{14}$$

For the reader, we first select $k$ paragraphs based on the RS→WS strategy and then obtain the final scores $\mathbf{Pr}_i^s$ and $\mathbf{Pr}_i^e$ for each paragraph, where $\mathbf{Pr}_i^s(t)$/$\mathbf{Pr}_i^e(t)$ represent the start/end scores of the t-th word in the i-th paragraph. As described above, the answer string can appear multiple times in a paragraph. Therefore, The reader is trained using a summed objective function that optimizes the negative log probability of selecting any correct answer span:

$$\mathcal{L}_r = -(log(\sum_{i=1}^{k}\sum_{t=1}^{|a_i|}(\mathbf{Pr}_i^s(x_i^t))) + log(\sum_{i=1}^{k}\sum_{t=1}^{|a_i|}(\mathbf{Pr}_i^e(z_i^t)))) \tag{15}$$

where $\{(x_i^t, z_i^t)\}_{t=1}^{|a_i|}$ is the set of the start and end positions of the answer strings that appear in the paragraph $P_i$.

During testing, we first utilize the ranker to predict a confidence score for each retrieved paragraph and utilize the RK strategy to select $k$ paragraphs. Next, the reader calculates the start and the end score of each word for all the selected paragraphs. Based on the reader-produced scores, there are two answer choosing methods:

**MAX** method is usually adopted in the traditional RC task. It simply chooses an answer span which has the maximum span score from all selected paragraphs. This span score is the product of the start score of the first word and the end score of the last word of the answer span.

**SUM** method first extracts an answer candidate $A_i$ that has the maximum span score from each paragraph. Next, if the answer candidates from different paragraphs refer to the same answer, we will sum up the scores of these answer candidates and choose the answer candidate with the maximum score as the final prediction. The similar method has also been adopted by recent approaches (Wang et al., 2018b; Lin et al., 2018; Pang et al., 2019).

# 3 Experiments

## 3.1 Datasets and Baselines

We evaluate our model on three OpenQA datasets, namely Quasar-T (Dhingra et al., 2017b), SearchQA (Dunn et al., 2017) and the unfiltered version of TriviaQA (Joshi et al., 2017).

**Quasar-T.** It consists of 43K open-domain trivia question-answer pairs, and about 100 paragraphs are provided for each question-answer pair by using the Solr search engine.

**SearchQA.** It has a total of more than 140k question-answer pairs, and roughly 50 webpage snippets are provided by the Google search engine as background paragraphs for each question.

**TriviaQA.** It contains approximately 95K open-domain question–answer pairs. The unfiltered version of TriviaQA is used in our experiment.

To better demonstrate the effectiveness of our model, we carefully select some recent approaches as baselines, including traditional RC models, i.e., BIDAF (Seo et al., 2016) , DECAPROP (Tay et al., 2018), single-paragraph models, i.e., R3 (Wang et al., 2018a), confidence-based models, i.e., DrQA (Chen et al., 2017) , DS-QA (Lin et al., 2018), S-Norm (Clark and Gardner, 2018), HAS-QA (Pang et al., 2019), MSR (Das et al., 2019) and answer re-ranking models, i.e., Re-Ranker (Wang et al., 2018b), Joint (Wang et al., 2018c).

## 3.2 Implementation Details

In the experiments, we adopt the same data preprocessing scheme as Clark and Gardner (2018). For the ranker, we use the 300-dimensional word embeddings pre-trained by GloVe (Pennington et al., 2014), which are fixed during training, and regard the 20-dimensional character embeddings as learnable parameters. The common word feature is mapped into a 4-dimensional vector and it is updated during training. We set the hidden size of LSTM to 150 and the number of LSTM layers to 1. We use Adam with learning rate 5e-4 to optimize the model. The batch size is set to 8 and dropout is applied to the outputs of all LSTM layers at a rate of 0.2. As for RS→WS, we first train the reader until convergence by RS. After that, we utilize WS to select paragraphs for training in order to further facilitate the performance of the reader. Our reader adopts the same hyperparameters as S-Norm (Clark and Gardner, 2018). While training the reader, the parameters of the ranker are fixed. The average length of paragraphs in TriviaQA is greater than that of Quasar-T and SearchQA, so we select 30, 30 and 15 paragraphs for Quasar-T, SearchQA and TriviaQA respectively.

| Models | Quasar-T | | SearchQA | | TriviaQA | | Average | |
|---|---|---|---|---|---|---|---|---|
| | EM | F1 | EM | F1 | EM | F1 | EM | F1 |
| BIDAF (Seo et al., 2016) | 25.9 | 28.5 | 28.6 | 34.6 | 41.1 | 47.4 | 31.8 | 36.8 |
| DECAPROP (Tay et al., 2018) | 38.6 | 46.9 | 56.8 | 63.6 | - | - | - | - |
| R3 (Wang et al., 2018a) | 35.3 | 41.7 | 49.0 | 55.3 | 47.3 | 53.7 | 43.8 | 50.2 |
| Re-Ranker (Wang et al., 2018b) | 42.3 | 49.6 | 57.0 | 63.2 | 50.6 | 53.7 | 49.9 | 55.5 |
| Joint (Wang et al., 2018c) | 45.9 | 53.9 | 58.3 | 64.2 | - | - | - | - |
| DrQA (Chen et al., 2017) | 37.7 | 44.5 | 41.9 | 48.7 | 32.3 | 38.3 | 37.3 | 43.8 |
| DS-QA (Lin et al., 2018) | 42.2 | 49.3 | 58.8 | 64.5 | 48.7 | 56.3 | 49.9 | 56.7 |
| S-Norm (Clark and Gardner, 2018) | 38.6 | 45.4 | 59.8 | 67.1 | 61.3 | 67.2 | 53.2 | 59.9 |
| HAS-QA (Pang et al., 2019) | 43.2 | 48.9 | 62.7 | 68.7 | 63.6 | 68.9 | 56.5 | 62.1 |
| MSR(BiDAF) (Das et al., 2019) | 40.6 | 47.0 | 56.2 | 61.4 | 55.9 | 61.7 | 50.9 | 56.7 |
| Our (RS, RK, MAX) | 46.6 | 56.5 | 60.4 | 68.0 | 63.8 | 70.4 | 56.9 | 64.9 |
| Our (RS→WS, RK, MAX) | 48.2 | 57.5 | 61.1 | 68.8 | 64.6 | 71.0 | 57.9 | 65.7 |
| Our (RS→WS, RK, SUM) | **48.6** | **57.8** | **62.9** | **69.8** | **65.2** | **71.3** | **58.9** | **66.3** |

Table 1: Experimental results on three OpenQA datasets: Quasar-T, SearchQA and TriviaQA. The results of 'BIDAF', 'DrQA' and 'S-Norm' are replicated from Pang et al. (2019). The unfiltered version of TriviaQA does not provide the answer of the test set. Therefore, we follow previous works to report the results on the development set of TriviaQA. Average: The average score of the three datasets.

### 3.3 Overall Results

In this section, we focus on the performance of the whole model. Table 1 presents the F1 and Exact Match (EM) scores of our model and the baseline models. These evaluation metrics are widely-adopted for OpenQA. Our (RS, RK, MAX) model utilizes RS and RK to select paragraphs for training and testing respectively and uses MAX to generate answer.

Our (RS, RK, MAX) model achieves better results on most of the datasets compared to the baselines. The main reason is that RS can select distracting paragraph to train the reader, thereby preventing the reader from being too confident in patterns that are only effective in positive paragraphs. Besides, compared with the paragraph-question relevance, the paragraph-paragraph relevance can further enhance the evidence among the relevant paragraphs. By jointly considering those relevance, RK can select more relevant paragraphs during testing to provide a better start point for the reader.

Our (RS→WS, RK, MAX) model consistently outperforms Our (RS, RK, MAX) model across the three datasets. The improvement is attributed to the adoption of the weighted sampling strategy. In the first stage of training, RS prevents excessive focus on the positive paragraphs. On this basis, WS further alleviates the influence of noisy paragraphs in the second stage without hurting the model's generalization ability.

We can observe that the SUM method (Wang et al., 2018b) helps our model better extract the correct answer. As we discussed, different from the traditional RC task, the answer string will appear multiple times in different paragraphs in OpenQA. By summing up the answer span scores which refer to the same answer candidate, we can extract a more accurate answer.

On SearchQA dataset, Our (RS→WS, RK, MAX) model achieves a close performance compared to HAS-QA model. 35% of the paragraphs in the development set of SearchQA contain the answer string, and for Quasar-T and TriviaQA the percentage of such paragraphs in the development set are 13% and 29% respectively. This indicates that SearchQA contains more positive paragraphs than other datasets. Since our model is better at filtering out the irrelevant data, the improvement in SearchQA is not as significant as in other datasets. However, Our (RS→WS, RK, SUM) model still achieves the SOTA result by adopting the SUM method and HAS-QA, which is a strong baseline, also adopts the similar method.

In the following sections, we give further analyses about the ranker, the selection strategies, the hyperparameter, the case study and the ablation study of our model.

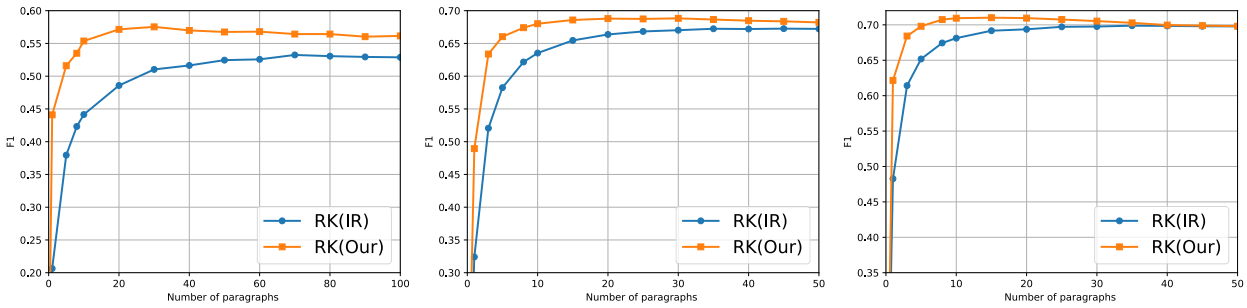


(a) Quasar-T (b) SearchQA (c) TriviaQA

Figure 3: Impact of the number of the selected paragraphs during testing on Quasar-T, SearchQA and TriviaQA. RK (IR), RK (Our): Utilizing a information retrieval model with BM25 or our ranker to select top $k$ paragraphs.

| | Quasar-T | | | SearchQA | | |
|---|---|---|---|---|---|---|
| **Models** | **Top1** | **Top3** | **Top5** | **Top1** | **Top3** | **Top5** |
| IR | 21.9 | 38.5 | 47.6 | 48.5 | 74.0 | 82.3 |
| R3 | 40.3 | 51.3 | 54.5 | - | - | - |
| DS-QA | 27.7 | 36.8 | 42.6 | 59.2 | 70.0 | 75.7 |
| FSE | 42.8 | 55.2 | 58.9 | 67.9 | 81.1 | 86.0 |
| MSR | 42.9 | 55.5 | 59.3 | - | - | - |
| Our | **49.9** | **58.8** | **62.6** | **75.0** | **85.3** | **89.1** |

Table 2: The performance of the rankers on the Quasar-T and SearchQA. We re-implement 'IR' and 'FSE'.

| | Quasar-T | | SearchQA | | TriviaQA | |
|---|---|---|---|---|---|---|
| **Train** | **EM** | **F1** | **EM** | **F1** | **EM** | **F1** |
| GT | 33.6 | 41.6 | 40.2 | 47.9 | 59.8 | 66.4 |
| RK | 46.1 | 55.2 | 59.7 | 67.3 | 61.9 | 68.5 |
| RS | 46.6 | 56.5 | 60.4 | 68.0 | 63.8 | 70.4 |
| WS | 47.2 | 55.9 | 60.9 | 68.6 | 64.3 | 70.6 |
| RS→WS | **48.2** | **57.5** | **61.1** | **68.8** | **64.6** | **71.0** |

Table 3: The performance of different training selection strategies on Quasar-T, SearchQA and TriviaQA.

### 3.4 Performance of the Ranker

We separately evaluate our ranker by comparing it with a typical IR model [2] and SOTA models which also have a ranking submodule to select paragraphs, i.e. DS-QA (Lin et al., 2018), R3 (Wang et al., 2018a), FSE (Zhang et al., 2018), MSR (Das et al., 2019). This evaluation simply focuses on whether the ground-truth appears in the Top 1, 3, 5 paragraphs. We follow the previous works to report the results on Quasar-T and SearchQA. As shown in Table 2, we can observe that our ranker surpasses the IR model with a large margin. The result implies that, the question-paragraph relevance is effective in selecting the relevant paragraphs. Compared with R3, DS-QA, FSE and MSR, which also adopt a neural network to select paragraphs, our model still works better. It suggests that exploiting the paragraph-question and paragraph-paragraph relevance with multi-level attention and self-attention mechanisms can further improve the performance of the ranker.

### 3.5 Performance of Selection Strategies

In this section, we compare the performance of different selection strategies for training. For fair comparison, a common scheme for testing is applied, where we use RK to select testing paragraphs and MAX to extract the answer. As shown in Table 3, the performance of RS is much better than GT and RK. The reason is that both GT and RK tend to train the reader on the positive paragraphs which inevitably impairs the generalization ability of the reader. It demonstrates that mainly choosing the positive paragraphs for training will hurt the performance of the model. Compared with RS, WS leads to a slight improvement thanks to the denoising effect. However, WS still reveals a bias toward the positive paragraphs during training, which may explain for the reason why the improvement is trivial. We can see that RS→WS gains the best score on three datasets. The reason is that the RS→WS strategy can not only ameliorates the influences of the distracting paragraphs but also filter out the noisy paragraphs to further strengthen our model.

### 3.6 Impact of the Number of Paragraphs

In order to further investigate the influences of different rankers and the hyperparameter $k$ during testing, we compare the final performance of RK (Our) with that of RK (IR). We utilize RS→WS to select paragraphs for training and and extract answers through the MAX method.

[2]The IR model ranks the paragraph with BM25

| |
|---|
| **Question**: Who was born on Krypton ? |
| **Answer**: Superman |
| **5 top-ranked paragraphs selected by our ranker**: |
| **P1 (positive, relevant)** : With a premise that taps into adolescent fantasy, Superman is born Kal-El on the alien planet Krypton, before being rocketed to Earth as an infant by his scientist father moments before the planet exploded. |
| **P2 (positive, relevant)** : The original story of Superman relates that he was born Kal-El on the planet Krypton, before being rocketed to Earth as an infant by his scientist father Jor-El, moments before Krypton's destruction. |
| **P3 (positive, relevant)** : Superman was born on the planet Krypton and was sent to Earth by his father moments before his planet exploded. |
| **P4 (positive, relevant)** : As of the early 1960s, (this story has undergone many revisions) Superman was born as Kal-el on the planet Krypton. |
| **P5 (positive, relevant)** : Superman was born Kal-El on the planet Krypton. |
| **5 top-ranked paragraphs selected by RKN**: |
| **P1 (positive, noisy)** : Television in the pilot episode of the 1950s television program Adventures of Superman Jor-El, portrayed by Robert Rockwell, was Krypton 's leading scientist, who tried to warn the ... |
| **P2 (negative, distracting)** : They did not tell anyone of his alien origins, letting family and friends believe he was born on the farm as their biological son, whom they named Clark Joseph Kent. |
| **P3 (positive, relevant)** : "Superman will be the only Kryptonian who survived the destruction of Krypton"- John Byrne on The Man of Steel. |
| **P4 (positive, noisy)** : In 1985, writer Alan Moore gave a somewhat darker glimpse into the world of Krypton in his story "For the Man Who Has Everything" (in Superman Annual), the premise being an ... |
| **P5 (positive, noisy)** : George Reeves, star of the Adventures of Superman television series, born in 1914. |

Table 4: An example from Quasar-T to illustrate the necessity of the paragraph-paragraph relevance. The key information, answers in correct-labeled and wrong-labeled contexts are marked in blue, green and red respectively. RKN: a naive version of our ranker which only utilizes the paragraph-question relevance.

RK (Our) and RK (IR) denote utilizing our ranker and a typical IR model, respectively, to select the $k$ top-ranked paragraphs for testing. As shown in Figure 3, it is obvious that the performance of the two models first increases and then remains stable or reduces as $k$ rises on three datasets. As the number of paragraphs increases, the chances are better that the answer is included in these paragraphs. However, the difficulty and running time for the reader also increase. As can be seen in Figure 3, the peak value of our ranker exceeds that of the IR model on three datasets, suggesting that our ranker can discover more useful and relevant paragraphs during testing, providing a better start point for the reader to predict the final answer.

### 3.7 Case Study

Table 4 shows an example from Quasar-T. The second row contains 5 top-ranked paragraphs which are produced by our ranker and the paragraphs in the third row are selected by a naive version of our ranker (RKN) which replaces the co-attention pooling (CAP) with max pooling for $\mathbf{C}_i$ and removes the sentence-level self-attention (SSA). For the question "Who was born on Krypton?", our ranker can select more related paragraphs and all of them support "Superman" as the answer. These paragraphs are inherently relevant to each other and provide enhanced evidence for the correct answer. However, most of the paragraphs in the third row are noisy and distracting, even though some of them are related to the question to some extent. Compared with RKN, which only uses the paragraph-question relevance, our completed ranker can further utilize the paragraph-paragraph relevance to aggregate the evidence among relevant paragraphs, thereby increasing their confidence scores. As a result, the reader is provided with more useful paragraphs to enhance the performance of our model.

| Models | P@1 | P@3 | P@5 | EM | F1 | ΔF1 |
|---|---|---|---|---|---|---|
| Our | 49.9 | 49.2 | 47.7 | 48.6 | 57.8 | 0 |
| -SUM | 49.9 | 49.2 | 47.7 | 48.2 | 57.5 | -0.3 |
| -RS→WS | 49.9 | 49.2 | 47.7 | 46.8 | 55.7 | -2.1 |
| -CAP | 47.8 | 46.9 | 45.8 | 47.6 | 57.0 | -0.8 |
| -SSA | 42.3 | 39.5 | 38.4 | 47.2 | 56.7 | -1.1 |
| -ranker | 21.9 | 20.1 | 19.1 | 44.2 | 52.6 | -5.2 |

Table 5: Ablation results on Quasar-T.

### 3.8 Ablation Study

Finally, we conduct an ablation study to analyze the effect of all proposed methods and report the results on Quasar-T. As shown in Table 5, we evaluate the performance of the ranker according to the precision at 1, 3 and 5. In comparison with the evaluation which assesses whether the ground-truth appears in the top-ranked paragraphs, this metric can better evaluate the performance of the ranker, as we expect to find as many relevant paragraphs in the $k$ top-ranked paragraphs as possible. We start with our completed model

which utilizes RS→WS to select 30 paragraphs for training and RK to select 30 top-ranked paragraphs for testing. As shown in Table 5, the '-SUM' denotes using MAX to choose the answer and the performance drop indicates that SUM can further enhance our model. Replacing RS→WS with RK for training accounts for a 2.1% performance drop, which demonstrates the effectiveness of our proposed selection strategy. The '-CAP' means replacing the co-attention pooling with max pooling for $\mathbf{C}_i$ and the '-SSA' means removing the sentence-level self-attention , and from the results, we can see that both paragraph-question and paragraph-paragraph relevance are integral to our model. The '-ranker' denotes replacing our ranker with the IR model in training and testing stages. The replacement accounts for a performance drop of 5.2%, which illustrates the indispensable role of our ranker in the final architecture.

## 4 Related Work

Most recent OpenQA approaches extract answer from distantly supervised paragraphs which are retrieved from the web (Chen et al., 2017; Dhingra et al., 2017a; Buck et al., 2018).

The interest of research roughly falls into two categories, namely the single-paragraph approaches and the multi-paragraph approaches, both of which follow the retrieve-then-read pipeline. Single-paragraph approaches (Joshi et al., 2017; Wang et al., 2018a) select a most relevant paragraph from all the retrieved paragraphs and then feed it to a RC model to extract the answer. The dependence on a single paragraph renders the RC model vulnerable to selection errors and a large amount of information in the remaining paragraphs is consequently being neglected.

Multi-paragraph approaches can be further divided into answer-rerank methods and confidence-based methods. For answer-rerank methods, Wang et al. (2018b) adopted a single-paragraph approach to extract an answer candidate from each retrieved paragraph and then used an extra neural network to rerank the candidates. Wang et al. (2018c) jointly trained the candidates extraction and reranking steps with reinforcement learning to further improve the performance. But these methods still rely heavily on the candidates extraction results.

Chen et al. (2017) proposed the first confidence method which retrieved multiple paragraphs with an IR system and applied a RC model to each paragraph for extracting the answer with the highest confidence. Das et al. (2019) introduced a new framework to make use of the result of the reader for improved paragraph retrieval. However, these models suffer from the problem of impaired generalization ability as a result of too much focus on the positive paragraphs in training. Clark and Gardner (2018) presented a new method, S-Norm. They used the sampling strategy and the shared-norm objective function to teach the model to ignore non-answer containing paragraphs. However, the retrieved paragraphs are always noisy and this will hurt the performance of the model.

Zhang et al. (2018) and Lee et al. (2018) adopted a neural network to rank the retrieved paragraphs. Lin et al. (2018) and Pang et al. (2019) also ranked the paragraphs and took into account both the ranking results and the scores produced by the reader to predict the final answer. However, they mainly utilize the paragraph-question relevance to rank the paragraphs and train the reader mainly on the positive paragraphs.

## 5 Conclusion

In this paper, we tackle the issues caused by the noisy and distracting paragraphs on OpenQA via ranking and sampling. By jointly considering the question-paragraph and the paragraph-paragraph relevance, our ranking model can calculate a more accurate confidence score for each paragraph. Through the modified weighted sampling strategy, our model can make full use of the dataset and mitigate the influence of the distracting and noisy paragraphs. Experimental results on three challenging public OpenQA datasets (Quasar-T, SearchQA and TriviaQA) show that our model advances the state of the art.

## 6 Acknowledgement

We thank the anonymous reviewers for their valuable suggestions. This work was supported by National Natural Science Foundation of China (No. 61976207, No. 61906187).

## References

Christian Buck, Jannis Bulian, Massimiliano Ciaramita, Wojciech Gajewski, Andrea Gesmundo, Neil Houlsby, and Wei Wang. 2018. Ask the right questions: Active question reformulation with reinforcement learning. In *Proceedings of International Conference on Learning Representations*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879. Association for Computational Linguistics.

Christopher Clark and Matt Gardner. 2018. Simple and effective multi-paragraph reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 845–855. Association for Computational Linguistics.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, and Andrew McCallum. 2019. Multi-step retriever-reader interaction for scalable open-domain question answering. In *Proceedings of International Conference on Learning Representations*.

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. 2017a. Gated-attention readers for text comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1832–1846. Association for Computational Linguistics.

Bhuwan Dhingra, Kathryn Mazaitis, and William W. Cohen. 2017b. Quasar: Datasets for question answering by search and reading. *Computing Research Repository*, arXiv:1707.03904.

Matthew Dunn, Levent Sagun, Mike Higgins, V. Ugur Güney, Volkan Cirik, and Kyunghyun Cho. 2017. Searchqa: A new q&a dataset augmented with context from a search engine. *Computing Research Repository*, arXiv:1704.05179.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611. Association for Computational Linguistics.

Jinhyuk Lee, Seongjun Yun, Hyunjae Kim, Miyoung Ko, and Jaewoo Kang. 2018. Ranking paragraphs for improving answer recall in open-domain question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 565–569. Association for Computational Linguistics.

Yankai Lin, Haozhe Ji, Zhiyuan Liu, and Maosong Sun. 2018. Denoising distantly supervised open-domain question answering. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1736–1745. Association for Computational Linguistics.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1003–1011. Association for Computational Linguistics.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Lixin Su, and Xueqi Cheng. 2019. HAS-QA: hierarchical answer spans model for open-domain question answering. In *Proceedings of Association for the Advancement of Artificial Intelligence*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543. Association for Computational Linguistics.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. In *Proceedings of International Conference on Learning Representations*.

Chuanqi Tan, Furu Wei, Nan Yang, Bowen Du, Weifeng Lv, and Ming Zhou. 2018. S-net: From answer extraction to answer synthesis for machine reading comprehension. In *Proceedings of Association for the Advancement of Artificial Intelligence*.

Yi Tay, Anh Tuan Luu, Siu Cheung Hui, and Jian Su. 2018. Densely connected attention propagation for reading comprehension. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018*, pages 4911–4922.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerry Tesauro, Bowen Zhou, and Jing Jiang. 2018a. R 3: Reinforced ranker-reader for open-domain question answering. In *Proceedings of Association for the Advancement of Artificial Intelligence*.

Shuohang Wang, Mo Yu, Jing Jiang, Wei Zhang, Xiaoxiao Guo, Shiyu Chang, Zhiguo Wang, Tim Klinger, Gerald Tesauro, and Murray Campbell. 2018b. Evidence aggregation for answer re-ranking in open-domain question answering. In *Proceedings of International Conference on Learning Representations*.

Zhen Wang, Jiachen Liu, Xinyan Xiao, Yajuan Lyu, and Tian Wu. 2018c. Joint training of candidate extraction and answer selection for reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1724. Association for Computational Linguistics.

Chen Zhang, Xuanyu Zhang, and Hao Wang. 2018. A machine reading comprehension-based approach for featured snippet extraction. In *Proceedings of IEEE International Conference on Data Mining*, pages 1416–1421.

# A Non-commutative Bilinear Model for Answering Path Queries in Knowledge Graphs

**Katsuhiko Hayashi** [†,*]
katsuhiko-h@sanken.osaka-u.ac.jp

**Masashi Shimbo** [‡,*]
shimbo@is.naist.jp

[†]Osaka University
Suita, Osaka, Japan

[‡]NAIST
Ikoma, Nara, Japan

[*]Riken AIP
Chuo-ku, Tokyo, Japan

## Abstract

Bilinear diagonal models for knowledge graph embedding (KGE), such as DistMult and ComplEx, balance expressiveness and computational efficiency by representing relations as diagonal matrices. Although they perform well in predicting atomic relations, composite relations (relation paths) cannot be modeled naturally by the product of relation matrices, as the product of diagonal matrices is commutative and hence invariant with the order of relations. In this paper, we propose a new bilinear KGE model, called BlockHolE, based on block circulant matrices. In BlockHolE, relation matrices can be non-commutative, allowing composite relations to be modeled by matrix product. The model is parameterized in a way that covers a spectrum ranging from diagonal to full relation matrices. A fast computation technique is developed on the basis of the duality of the Fourier transform of circulant matrices.

## 1 Introduction

Large-scale knowledge graphs (Nickel et al., 2016a) are indispensable resources for knowledge-intensive applications such as question answering, dialog systems, and distantly supervised relation extraction. A knowledge graph is a collection of *triplets* $(s, r, o)$ representing the fact that (binary) relation $r$ holds between subject entity $s$ and object entity $o$. Although efforts continue to enrich existing knowledge graphs with more facts, many facts are still missing (Nickel et al., 2016a). Knowledge graph completion (KGC) aims to automatically detect missing facts in an *incomplete* knowledge graph, and has become an active field of research in recent years.

Knowledge graph embedding (KGE) is a promising approach to KGC. It embeds entities and relations in vector space, and defines a *scoring function* $\phi(s, r, o)$ to evaluate the degree of factuality of a given triplet $(s, r, o)$ in terms of vector operations.

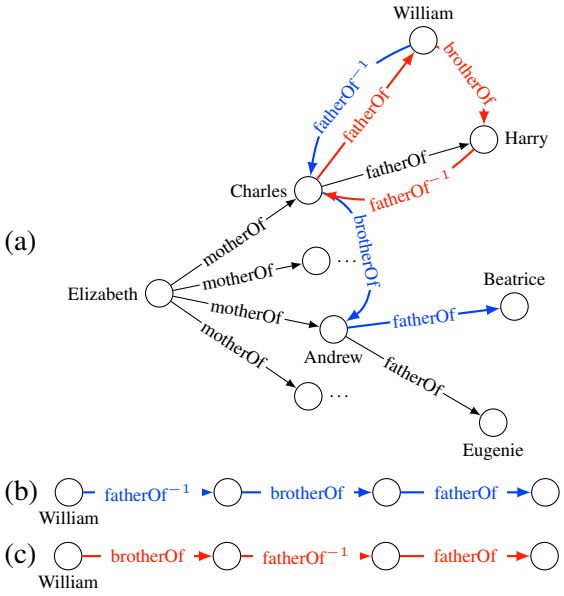


Figure 1: (a) A knowledge graph and (b,c) two relation paths starting from William.

Bilinear KGE models are a popular choice for a scoring function, along with those based on translation and neural networks. RESCAL (Nickel et al., 2011) adopts a generic bilinear form as the scoring function, given by $\phi_{\text{RESCAL}}(s, r, o) = \mathbf{e}_s^{\mathrm{T}} \mathbf{R}_r \mathbf{e}_o$. In this formula, $\mathbf{e}_s, \mathbf{e}_o$ are the $n$-dimensional vector embeddings of entities $s$ and $o$, respectively, and $\mathbf{R}_r$ is the $n \times n$ matrix embedding of relation $r$. Some of the more recent models have constrained the relation matrices to be diagonal. DistMult (Yang et al., 2015) and ComplEx (Trouillon et al., 2016) are two such diagonal models. HolE (Nickel et al., 2016b) does not use diagonal relation matrices, but has been shown (Hayashi and Shimbo, 2017) to be isomorphic to ComplEx. These models have a smaller number of parameters than RESCAL, making them less prone to overfit-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2422–2430, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

ting, and the performance is usually better.

While all these models were designed with a specific task of KGC in mind, i.e., computing the factuality of triplets, another important task on knowledge graphs was pursued by Guu et al. (2015) and Lin et al. (2015). This latter task, called *path query answering (path QA)*, is to answer composite queries that consist of a cascade of relations, as opposed to an atomic relation. See Figure 1 for instance. A query "Is Beatrice a child of a paternal uncle of William?" can be answered by predicting the truth value of the triplet (William, fatherOf$^{-1}$/brotherOf/fatherOf, Beatrice) where fatherOf$^{-1}$/brotherOf/fatherOf is a binary relation not present in the knowledge graph as a relation (edge) label but is composed of a cascade of three atomic relations.[1] Composite queries are also called *path queries*, as they can be represented as paths in a knowledge graph; see, e.g., the blue line in Figure 1(a). Notice however that some of the edges in the path may be missing due to the incompleteness of the knowledge graph; even in such circumstances, the model must ideally be able to answer path queries correctly.

Guu et al. (2015) extended the existing KGE approaches to path QA. For example, to answer a general path query $(s, r_1/\ldots/r_k, o)$ with RESCAL, a composite relation $r_1/\cdots/r_k$ is modeled by matrix product $\mathbf{R}_{r_1}\cdots\mathbf{R}_{r_k}$, and the score for the given query is modeled by $\mathbf{e}_s^{\mathrm{T}}\mathbf{R}_{r_1}\cdots\mathbf{R}_{r_k}\mathbf{e}_o$. This formulation is also applicable to DistMult and ComplEx, which use diagonal relation matrices. In diagonalized models, however, relation matrices are *commutative*, in the sense that $\mathbf{R}_r\mathbf{R}_{r'} = \mathbf{R}_{r'}\mathbf{R}_r$ for any pair of relations $r, r'$.

Commutativity of relation matrices was not recognized as an issue in the past research because the main focus was on predicting the truth value of atomic triplets. However, when path queries are concerned, commutativity poses a problem. Consider, for example, a relation sequence

$$\text{fatherOf}^{-1}/\text{brotherOf}/\text{fatherOf}$$

and its permutation

$$\text{brotherOf}/\text{fatherOf}^{-1}/\text{fatherOf}.$$

Although these are two distinct paths (cf. Figure 1(b, c)), in bilinear models with commutative relation matrices, they are represented by the same product of relation matrices, which thereby makes the truth values of these permutated queries indistinguishable by their scores.

Drawing on the observation above, this paper proposes a new KGE model called BlockHolE, wherein relations are represented by block circulant matrices. This makes relation matrices non-commutative, and thus it does not suffer from the issues arising from commutativity, yet in general manages to reduce the number of parameters compared with RESCAL. It can be interpreted as a generalization of HolE and ComplEx, and also subsumes RESCAL as an extreme case. We report experimental results in both path and atomic QA tasks.

[1] We regard inverse relations (e.g., fatherOf$^{-1}$) also as atomic relations.

| Symbol | Description |
|---|---|
| $\mathbb{R}, \mathbb{C}$ | sets of real/complex numbers |
| $[\mathbf{v}]_j$ | $j$th component of vector $\mathbf{v}$ |
| $[\mathbf{M}]_{jk}$ | $(j,k)$-component of matrix $\mathbf{M}$ |
| $\mathbf{M}^{\mathrm{T}}$ | transpose of $\mathbf{M}$ |
| $\overline{\mathbf{M}}$ | conjugate of $\mathbf{M}$ |
| $\odot$ | componentwise (Hadamard) product |
| $*$ | circular convolution |
| $\star$ | circular correlation |
| $\mathrm{Re}(x)$ | real part of complex number $x$ |
| $\mathrm{diag}(\mathbf{v})$ | diagonal matrix with main diagonal $\mathbf{v}$ |
| $\mathrm{circ}(\mathbf{v})$ | circulant matrix determined by $\mathbf{v}$ |
| $\langle \mathbf{x}, \mathbf{y}, \mathbf{z} \rangle$ | sum of the componentwise products of $\mathbf{x}, \mathbf{y}, \mathbf{z}$ |
| $\mathbf{F}$ | discrete Fourier matrix |
| $\mathcal{E}$ | set of entities |
| $\mathcal{R}$ | set of relations |
| $\mathcal{F}$ | set of observed facts (triplets) |
| $\mathcal{F}^*$ | set of ground truth facts |
| $\mathcal{G}(\mathcal{F})$ | Knowledge graph induced by facts $\mathcal{F}$ |

Table 1: List of symbols. See Secs. 2 and 3 for detail.

## 2 Notation and preliminaries

We first introduce symbols and notation used in this paper, followed by some preliminaries on circulant matrices, circular convolution, correlation, and Fourier transform. The summary of symbols can be found in Table 1.

Let $\mathbb{R}$ be the set of reals, and $\mathbb{C}$ be the set of complex numbers. Let $[\mathbf{v}]_j$ denote the $j$th component of vector $\mathbf{v}$, and let $[\mathbf{M}]_{jk}$ the $(j,k)$ element of matrix $\mathbf{M}$. For a complex number $z$, vector $\mathbf{z}$, and matrix $\mathbf{Z}$, let $\overline{z}$, $\overline{\mathbf{z}}$, and $\overline{\mathbf{Z}}$ denote their complex conjugate, respectively.

Let $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ be $n$-dimensional (real or complex) vectors. Let $\mathrm{diag}(\mathbf{x})$ denote an $n \times n$ diagonal matrix with the main diagonal components given by $\mathbf{x}$. We write $\mathbf{x} \odot \mathbf{y}$ to denote the componentwise product of $\mathbf{x}$ and $\mathbf{y}$; i.e., $\mathbf{x} \odot \mathbf{y} = \mathrm{diag}(\mathbf{x})\mathbf{y}$, or

$[\mathbf{x} \odot \mathbf{y}]_i = [\mathbf{x}]_i[\mathbf{y}]_i$, $i = 1, \ldots, n$. We also write $\langle \mathbf{x}, \mathbf{y}, \mathbf{z} \rangle = \mathbf{x}^{\mathrm{T}} \operatorname{diag}(\mathbf{y})\mathbf{z} = \sum_{i=1}^{n} [\mathbf{x}]_i[\mathbf{y}]_i[\mathbf{z}]_i$.

For $n$-dimensional real vectors[2] $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, $\mathbf{x} * \mathbf{y}$ and $\mathbf{x} \star \mathbf{y}$ denote *circular convolution* and *circular correlation*, respectively defined by

$$[\mathbf{x} * \mathbf{y}]_i = \sum_{j=1}^{n} [\mathbf{x}]_{i-j+1}[\mathbf{y}]_j,$$

$$[\mathbf{x} \star \mathbf{y}]_i = \sum_{j=1}^{n} [\mathbf{x}]_{j-i+1}[\mathbf{y}]_j, \quad i = 1, \ldots, n,$$

where vector indices that do not fall in the range $1, \ldots, n$ must be interpreted by $0 = n, -1 = n - 1, \ldots, -n + 1 = 1$.

For $n$-dimensional real vector $\mathbf{v} \in \mathbb{R}^n$, let

$$\operatorname{circ}(\mathbf{v}) = \begin{bmatrix} [\mathbf{v}]_1 & [\mathbf{v}]_n & \cdots & [\mathbf{v}]_3 & [\mathbf{v}]_2 \\ [\mathbf{v}]_2 & [\mathbf{v}]_1 & [\mathbf{v}]_n & & [\mathbf{v}]_3 \\ \vdots & [\mathbf{v}]_2 & [\mathbf{v}]_1 & \ddots & \vdots \\ [\mathbf{v}]_{n-1} & & \ddots & \ddots & [\mathbf{v}]_n \\ [\mathbf{v}]_n & [\mathbf{v}]_{n-1} & \cdots & [\mathbf{v}]_2 & [\mathbf{v}]_1 \end{bmatrix}$$

be an operation that converts a vector $\mathbf{v}$ to a circulant matrix of size $n \times n$.

A circulant matrix $\operatorname{circ}(\mathbf{v}) \in \mathbb{R}^{n \times n}$ can be diagonalized as $\operatorname{circ}(\mathbf{v}) = \mathbf{F}^{-1} \operatorname{diag}(\mathbf{F}\mathbf{v})\mathbf{F}$, where $\mathbf{F} \in \mathbb{C}^{n \times n}$ is the discrete Fourier matrix of order $n$. Also, circular convolution and correlation can be written in terms of $\operatorname{circ}(\cdot)$: $\mathbf{x} * \mathbf{y} = \operatorname{circ}(\mathbf{x})\mathbf{y}$, and $\mathbf{x} \star \mathbf{y} = \operatorname{circ}(\mathbf{x})^{\mathrm{T}}\mathbf{y}$. It follows that

$$\mathbf{x} * \mathbf{y} = \mathbf{F}^{-1}(\mathbf{F}\mathbf{x} \odot \mathbf{F}\mathbf{y}), \tag{1}$$

$$\mathbf{x} \star \mathbf{y} = \mathbf{F}^{-1}(\overline{\mathbf{F}\mathbf{x}} \odot \mathbf{F}\mathbf{y}). \tag{2}$$

These equations imply that circular convolution and correlation can be computed in time $O(n \log n)$ using the fast Fourier transform (FFT).

# 3 Knowledge graph embedding using bilinear maps

A *knowledge graph* is a labeled multigraph $(\mathcal{E}, \mathcal{R}, \mathcal{F})$, where $\mathcal{E}$ is the set of entities (or vertices), $\mathcal{R}$ is the set of relation labels (or edge labels), and $\mathcal{F} \subset \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ defines the observed instances of binary relations over entities (or labeled edges). An item $(s, r, o) \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ is called a *triplet*, with $s$ and $o$ called its *subject* and *object*, respectively. For every entity in $e \in \mathcal{E}$, it is assumed that $\mathcal{F}$ contains at least one triplet $(s, r, o)$ with $s = e$ or $o = e$; likewise, for every relation in $r \in \mathcal{R}$, $\mathcal{F}$ is assumed to contain at least one triplet $(s, r, o)$. Because $\mathcal{F}$ determines the sets $\mathcal{E}$ and $\mathcal{R}$ of entities and relations, we write $\mathcal{G}(\mathcal{F})$ to denote the knowledge graph $(\mathcal{E}, \mathcal{R}, \mathcal{F})$ determined by $\mathcal{F}$.

Aside from observed triplets $\mathcal{F}$, we also assume the presence of a set $\mathcal{F}^* \supset \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ of *(ground truth) facts*, which is a strict superset of $\mathcal{F}$, i.e., $\mathcal{F} \subset \mathcal{F}^*$. Thus, $\mathcal{F}^*$ is not fully observable.

## 3.1 Knowledge graph completion

*Knowledge graph completion* (KGC) is the task of identifying the set of ground truth facts $\mathcal{F}^*$ from observed facts $\mathcal{F} \subset \mathcal{F}^*$ (or equivalently, $\mathcal{G}(\mathcal{F}^*)$ from $\mathcal{G}(\mathcal{F})$).

A popular approach to KGC is to design a *scoring function* $\phi(s, r, o)$ quantifying how likely a triplet $(s, r, o)$ is true. This scoring function is learned from the observed triplets $\mathcal{F}$, in a way that it generalizes well to unobserved triplets $\mathcal{F}^* \backslash \mathcal{F}$; i.e., the score must be high for both observed and unobserved facts, and it must be low for nonfactual triplets.

In *knowledge graph embedding* (KGE)–based approaches to KGC, the scoring function $\phi(s, r, o)$ is defined in terms of the embeddings of entities and relations; i.e., $s$, $r$, and $o$ are embedded as objects in a vector space, and $\phi$ is defined in terms of some operations over these objects.

## 3.2 Bilinear models for knowledge graph embedding

Below, we describe some of the popular KGE models that use bilinear maps to define scoring functions.

### 3.2.1 RESCAL

RESCAL (Nickel et al., 2011) provides the most general form of bilinear scoring function.

$$\phi_{\text{RESCAL}}(s, r, o) = \mathbf{e}_s^{\mathrm{T}} \mathbf{R}_r \mathbf{e}_o, \tag{3}$$

where $\mathbf{e}_s, \mathbf{e}_o \in \mathbb{R}^n$ are the vector embeddings of entities $s$ and $o$, respectively, and $\mathbf{R}_r \in \mathbb{R}^{n \times n}$ is the matrix representing relation $r$. Thus, $n^2$ parameters are required per relation, which is not only a computational burden but also the cause of overfitting during training (Kazemi and Poole, 2018).

### 3.2.2 DistMult

DistMult (Yang et al., 2015) is a model obtained by restricting the relation matrices $\mathbf{R}_r$ of RESCAL

[2] Generally, circular convolution, circular correlation, and circulant matrices are defined over $\mathbb{C}^n$. However, in this paper, it suffices to define them over $\mathbb{R}^n$.

to diagonal; i.e., $\mathbf{R}_r = \mathrm{diag}(\mathbf{w}_r)$, $\mathbf{w}_r \in \mathbb{R}^n$. The scoring function is thus

$$\begin{aligned}\phi_{\text{DistMult}}(s, r, o) &= \mathbf{e}_s{}^{\mathrm{T}} \,\mathrm{diag}(\mathbf{w}_r)\mathbf{e}_o \\ &= \langle \mathbf{w}_r, \mathbf{e}_s, \mathbf{e}_o \rangle. \end{aligned} \tag{4}$$

Although the number of parameters is reduced considerably, the scoring function (4) is symmetric with respect to the entities, i.e., $\phi_{\text{DistMult}}(s, r, o) = \phi_{\text{DistMult}}(o, r, s)$. This is a severe limitation because most real-world relations are non-symmetric.

#### 3.2.3 ComplEx: Complex embedding

The *complex embedding* (ComplEx) (Trouillon et al., 2016) represents entities and relations as $n$-dimensional vectors as in DistMult, but their components are complex-valued.

The scoring function of ComplEx is given by

$$\begin{aligned}\phi_{\text{ComplEx}}(s, r, o) &= \mathrm{Re}(\mathbf{e}_s{}^{\mathrm{T}} \,\mathrm{diag}(\mathbf{w}_r)\overline{\mathbf{e}_o}) \\ &= \mathrm{Re}(\langle \mathbf{w}_r, \mathbf{e}_s, \overline{\mathbf{e}_o} \rangle),\end{aligned}$$

where $\mathbf{e}_s, \mathbf{e}_o, \mathbf{w}_r \in \mathbb{C}^n$ are the embeddings of $s$, $o$, and $r$, respectively. The number of parameters in ComplEx is $2n|\mathcal{E}| + 2n|\mathcal{R}|$, and the score is computable in time linear in the dimension of vector space. Unlike DistMult, ComplEx can model non-symmetric relations, since $\phi(s, r, o) \neq \phi(o, r, s)$ in general.

#### 3.2.4 HolE: Holographic embedding

The *holographic embedding* (HolE) (Nickel et al., 2016b) uses circular correlation to define a scoring function

$$\phi_{\text{HolE}}(s, r, o) = \mathbf{w}_r^{\mathrm{T}}(\mathbf{e}_s \star \mathbf{e}_o), \tag{5}$$

where $\mathbf{w}_r, \mathbf{e}_s, \mathbf{e}_o \in \mathbb{R}^n$ are $n$-dimensional real vectors representing relation $r$, and entities $s$ and $o$, respectively. HolE has only $n$ parameters per relation, and it can model non-symmetric relations since $\phi_{\text{HolE}}(s, r, o) \neq \phi_{\text{HolE}}(o, r, s)$ in general. Computing circular correlation requires $O(n \log n)$ time if FFT is employed. Eq. (5) is not a bilinear form, but it has been shown (Hayashi and Shimbo, 2017) that HolE is isomorphic to ComplEx, and thus any model in HolE can be converted to an equivalent model in ComplEx, and vice versa.

## 4 Path question answering over a knowledge graph

### 4.1 Path query answering

Let $\mathcal{F}^*$ be the set of ground truth facts, and let $\mathcal{G}(\mathcal{F}^*) = (\mathcal{E}, \mathcal{R}, \mathcal{F}^*)$ be its induced knowledge graph. For $k$ relations $r_1, \ldots, r_k \in \mathcal{R}$, we call $r_1/\ldots/r_k$ a *relation path* of length $k$. When $k = 1$, the relation path is *atomic*; otherwise, it is *composite*. Let $s, o \in \mathcal{E}$. We say a *path query* $(s, r_1/\ldots/r_k, o)$ holds (or "is true") in $\mathcal{G}(\mathcal{F}^*)$ (or with respect to $\mathcal{F}^*$) if

$$\exists e_1, \ldots, e_{k-1} \in \mathcal{E} \;\; \forall j = 1, \ldots, k \quad (e_{j-1}, r_j, e_j) \in \mathcal{F}^*,$$

where $e_0 = s$ and $e_k = o$. *Path query answering* (path QA) is the task of predicting the truth value of path queries with respect to the unobserved set $\mathcal{F}^*$ of ground truth facts, when its incomplete subset $\mathcal{F} \subset \mathcal{F}^*$ is only available. In other words, we want to predict that $(s, r_1/\ldots/r_k, o)$ is true if a path from $s$ to $o$ exists in $\mathcal{G}(\mathcal{F}^*)$, although some of the edges that constitute the path may be missing in the observed graph $\mathcal{G}(\mathcal{F})$.

For atomic path queries (i.e., those with length $k = 1$), path QA reduces to that of knowledge graph completion introduced in Section 3.1. Thus, it is natural to address general path QA by extending the scoring function $\phi(s, r, o)$ of KGC methods so that composite relation $r_1/\ldots/r_k$ is allowed in place of atomic relation $r$; i.e., by defining $\phi(s, r_1/\ldots/r_k, o)$. Previous work (Guu et al., 2015) explored this direction, which is also pursued in the rest of this paper.

### 4.2 Issues in existing KGE models applied to path QA

We now discuss the extension of existing bilinear KGE models to path QA. We begin with RESCAL, which is the most general among existing bilinear models. In RESCAL, if we assume $\mathbf{R}_r^{\mathrm{T}}\mathbf{e}_s \approx \mathbf{e}_o$ for true triplets $(s, r, o)$, we can model path QA as computing

$$\begin{aligned}&\phi_{\text{RESCAL}}(s, r_1/\ldots/r_k, o) \\ &\qquad = \mathbf{e}_s^{\mathrm{T}}\mathbf{R}_{r_1}\cdots\mathbf{R}_{r_k}\mathbf{e}_o. \end{aligned} \tag{6}$$

As seen in this formula, a composite relation is represented by the product of the matrices for atomic relations (Guu et al., 2015).

Likewise, DistMult and ComplEx can also be used for path QA, by computing

$$\begin{aligned}&\phi_{\text{DistMult}}(s, r_1/\ldots/r_k, o) \\ &\qquad = \mathbf{e}_s^{\mathrm{T}}\,\mathrm{diag}(\mathbf{w}_{r_1})\cdots\mathrm{diag}(\mathbf{w}_{r_k})\mathbf{e}_o,\end{aligned}$$

and

$$\phi_{\text{ComplEx}}(s, r_1/\ldots/r_k, o) = \text{Re}(\mathbf{e}_s^{\text{T}} \,\text{diag}(\mathbf{w}_{r_1}) \cdots \text{diag}(\mathbf{w}_{r_k}) \overline{\mathbf{e}_o}),$$

respectively. However, because diagonal matrices are commutative, the score of $(s, r_1/\ldots/r_k, o)$ is equal to any path query in which $r_1, \cdots, r_k$ are permutated, such as $(s, r_k/r_{k-1}/\ldots/r_1, o)$. That is, because $\phi(s, r_1/\ldots/r_k, o) = \phi(s, r_k/r_{k-1}/\ldots/r_1, o)$, their truth values cannot be distinguished by the magnitude of scores. More recent bilinear models such as ANALOGY[3] (Liu et al., 2017) and SimplE (Kazemi and Poole, 2018) also represent relations by diagonal matrices, and thus they can only model commutative relation paths. Moreover, for SimplE, which represents subject and object entities in different vector spaces, it is not clear how it can be applied to path QA.

In the translation-based model TransE (Bordes et al., 2013), the scoring function is given by[4]

$$\phi_{\text{TransE}}(s, r, o) = -||\mathbf{e}_s + \mathbf{w}_r - \mathbf{e}_o||_2^2. \quad (7)$$

Guu et al. (2015) extended this function for a path query by

$$\phi_{\text{TransE}}(s, r_1/\ldots/r_k, , o) = -\|\mathbf{e}_s + \mathbf{w}_{r_1} + \cdots + \mathbf{w}_{r_k} - \mathbf{e}_o\|_2^2. \quad (8)$$

Thus, a composite relation is represented as the sum of the embedding vectors for its constituent atomic relations. Unfortunately, Eq. (8) is also invariant with the permutation of relations $r_1, \ldots, r_k$, and their order is not respected.

## 5 Knowledge graph embedding with block circulant matrices

### 5.1 BlockHolE

In this section, we propose a bilinear KGE model suitable for path QA. In this model, the relation matrices are non-commutative. It thus respects the order of relations in a path query. Further, it has a smaller number of parameters than RESCAL in general. To be specific, our model constrains the relation matrices to be block circulant.

A matrix is *block circulant* if it can be written in the form

$$\begin{bmatrix} \mathbf{W}^{(11)} & \cdots & \mathbf{W}^{(1b)} \\ \vdots & \ddots & \vdots \\ \mathbf{W}^{(b1)} & \cdots & \mathbf{W}^{(bb)} \end{bmatrix}, \quad (9)$$

where each $\mathbf{W}^{(ij)} = \text{circ}(\mathbf{w}^{(ij)})$, $i, j = 1, \ldots, b$, is a circulant matrix determined by $\mathbf{w}^{(ij)} \in \mathbb{R}^m$. Thus, if the dimension of the matrix in Eq. (9) is $n \times n$, we have $n = bm$. A block circulant matrix is non-commutative when $b \geq 2$; i.e., for two block circulant matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{bm \times bm}$, $\mathbf{A} \neq \mathbf{B}$, $\mathbf{AB} \neq \mathbf{BA}$ in general.

Substituting a block circulant matrix of Eq. (9) for matrix $\mathbf{R}_r$ in the bilinear scoring function (Eq. (3)) yields

$$\phi_{\text{BlockHolE}}(s, r, o) = \mathbf{e}_s^{\text{T}} \mathbf{R}_r \mathbf{e}_o = \overbrace{[\mathbf{e}_s^{(1)\text{T}} \cdots \mathbf{e}_s^{(b)\text{T}}]}^{\mathbf{e}_s^{\text{T}}} \overbrace{\begin{bmatrix} \mathbf{W}_r^{(11)} & \cdots & \mathbf{W}_r^{(1b)} \\ \vdots & \ddots & \vdots \\ \mathbf{W}_r^{(b1)} & \cdots & \mathbf{W}_r^{(bb)} \end{bmatrix}}^{\mathbf{R}_r} \overbrace{\begin{bmatrix} \mathbf{e}_o^{(1)} \\ \vdots \\ \mathbf{e}_o^{(b)} \end{bmatrix}}^{\mathbf{e}_o}, \quad (10)$$

where $\mathbf{e}_s^{(i)}, \mathbf{e}_o^{(i)} \in \mathbb{R}^m$, and $\mathbf{W}_r^{(ij)} = \text{circ}(\mathbf{w}_r^{(ij)}) \in \mathbb{R}^{m \times m}$, $i, j = 1, \ldots, b$. Recall that $n = bm$, and thus $\mathbf{e}_s, \mathbf{e}_o \in \mathbb{R}^n$, $\mathbf{R}_r \in \mathbb{R}^{n \times n}$. Using equalities $\mathbf{x}^{\text{T}}(\mathbf{y} * \mathbf{z}) = \mathbf{y}^{\text{T}}(\mathbf{x} \star \mathbf{z})$ (Nickel et al., 2016b) and $\text{circ}(\mathbf{x})\mathbf{y} = \mathbf{x} * \mathbf{y}$ to rewrite Eq. (10), we have

$$\begin{aligned} \phi_{\text{BlockHolE}}(s, r, o) &= [\mathbf{e}_s^{(1)\text{T}} \ldots \mathbf{e}_s^{(b)\text{T}}] \begin{bmatrix} \sum_{j=1}^b \mathbf{w}_r^{(1j)} * \mathbf{e}_o^{(j)} \\ \vdots \\ \sum_{j=1}^b \mathbf{w}_r^{(bj)} * \mathbf{e}_o^{(j)} \end{bmatrix} \\ &= \sum_{i=1}^b \mathbf{e}_s^{(i)\text{T}} \left( \sum_{j=1}^b \mathbf{w}_r^{(ij)} * \mathbf{e}_o^{(j)} \right) \\ &= \sum_{i,j=1}^b \mathbf{w}_r^{(ij)\text{T}} \left( \mathbf{e}_o^{(j)} \star \mathbf{e}_s^{(i)} \right). \quad (11) \end{aligned}$$

We call this model *BlockHolE*, after the fact that it reduces to HolE when $b = 1$; cf. Eq. (5). Also, BlockHolE is identical to $b$-dimensional RESCAL when $m = 1$ (or equivalently $b = n$).

The number of parameters in BlockHolE is $bm|\mathcal{E}| + b^2 m|\mathcal{R}|$ (or $n|\mathcal{E}| + bn|\mathcal{R}|$), and naive

[3] We categorize ANALOGY as a diagonal model because each $2 \times 2$ block diagonal element of its relation matrices can be substituted by a single equivalent complex-valued component.

[4] The original TransE defines a *penalty* function, which gives a smaller value if a triplet is more likely to be true. We thus changed the sign to make it a scoring function in Eq. (7).

computation of Eq. (11) takes time $O(b^2 m \log m)$ using FFT. However, we can make this computation faster by exploiting the duality of the Fourier transform, as shown below.

### 5.2 Fast computation in complex space

Using a similar technique used by Hayashi and Shimbo (2017) to show the equivalence of ComplEx and HolE, we can eliminate Fourier transform to speed up the computation of BlockHolE scores. We first rewrite Eq. (11) as follows:

$$\begin{aligned}
&\phi_{\text{BlockHolE}}(s,r,o) \\
&\quad = \sum_{i,j=1}^{b} \mathbf{w}_r^{(ij)\mathrm{T}}(\mathbf{e}_o^{(j)} \star \mathbf{e}_s^{(i)}) \\
&\quad = \sum_{i,j=1}^{b} \mathbf{w}_r^{(ij)\mathrm{T}}\mathbf{F}^{-1}(\overline{\mathbf{F}\mathbf{e}_o^{(j)}} \odot \mathbf{F}\mathbf{e}_s^{(i)}) \\
&\quad = \frac{1}{m}\sum_{i,j=1}^{b} (\overline{\mathbf{F}\mathbf{w}_r^{(ij)}})^{\mathrm{T}}(\overline{\mathbf{F}\mathbf{e}_o^{(j)}} \odot \mathbf{F}\mathbf{e}_s^{(i)}) \\
&\quad = \frac{1}{m}\sum_{i,j=1}^{b} \mathrm{Re}(\langle \mathbf{F}\mathbf{w}_r^{(ij)}, \overline{\mathbf{F}\mathbf{e}_s^{(i)}}, \mathbf{F}\mathbf{e}_o^{(j)}\rangle),
\end{aligned}$$

where $\mathbf{F}$ is the discrete Fourier matrix. Here we used Eq. (1) to derive the second equation, and $\mathbf{F}^{-1} = (1/m)\overline{\mathbf{F}}^{\mathrm{T}}$ to derive the third. Defining complex vectors $\mathbf{w}'^{(ij)}_r = (1/m)\mathbf{F}\mathbf{w}_r^{(ij)}$, $\mathbf{e}'^{(i)}_s = \overline{\mathbf{F}\mathbf{e}_s^{(i)}}$, and $\mathbf{e}'^{(j)}_o = \overline{\mathbf{F}\mathbf{e}_o^{(j)}}$ yields

$$\begin{aligned}
&\phi_{\text{BlockHolE}}(s,r,o) \\
&\quad = \sum_{i,j=1}^{b} \mathrm{Re}\left(\langle \mathbf{w}'^{(ij)}_r, \mathbf{e}'^{(i)}_s, \overline{\mathbf{e}'^{(j)}_o}\rangle\right). \quad (12)
\end{aligned}$$

On the basis of Eq. (12), we train $\mathbf{e}'^{(i)}_k \in \mathbb{C}^m$ directly in complex space (i.e., the Fourier domain) instead of $\mathbf{e}_k^{(j)} \in \mathbb{R}^m$ and use it as the vector embedding of entity $k$, for all $k \in \mathcal{E}$; similarly, $\mathbf{w}'^{(ij)}_r \in \mathbb{C}^m$ is directly trained in complex space to represent relation $r \in \mathcal{R}$. The number of parameters in this model is $2n|\mathcal{E}| + 2bn|\mathcal{R}|$, and Eq. (12) can be computed in $O(bn)$ time. Typically, we set $b \ll n$. For instance, in the experiment of Section 6, we set $b = 2$ and $m = 50$, and thus $n = bm = 100$. In this case, factor $b$ is negligible and the computational complexity is linear in $n$.

| | | WN11 | FB13 |
|---|---|---|---|
| Base | Train | 112,581 | 316,232 |
| | Valid | 2,609 | 5,908 |
| | Test | 10,544 | 23,733 |
| Path | Train | 2,129,539 | 6,266,058 |
| | Valid | 11,277 | 27,163 |
| | Test-Deduction | 24,749 | 77,883 |
| | Test-Induction | 21,828 | 31,674 |

Table 2: Dataset provided by Guu et al. (2015).

### 5.3 Modeling path QA

BlockHolE can be used in path QA as follows. First, for any $\ell \in \mathcal{E}$ and $r \in \mathcal{R}$, let

$$\begin{aligned}
\mathbf{e}'^{\mathrm{T}}_\ell &= [\mathbf{e}'^{(1)\mathrm{T}}_\ell \cdots \mathbf{e}'^{(b)\mathrm{T}}_\ell], \\
\mathbf{W}'_r &= \begin{bmatrix} \mathrm{diag}(\mathbf{w}'^{(11)}_r) & \cdots & \mathrm{diag}(\mathbf{w}'^{(1b)}_r) \\ \vdots & \ddots & \vdots \\ \mathrm{diag}(\mathbf{w}'^{(b1)}_r) & \cdots & \mathrm{diag}(\mathbf{w}'^{(bb)}_r) \end{bmatrix}.
\end{aligned}$$

Then, Eq. (12) can be rewritten as

$$\phi_{\text{BlockHolE}}(s,r,o) = \mathrm{Re}\left(\mathbf{e}'^{\mathrm{T}}_s \mathbf{W}'_r \overline{\mathbf{e}'}_o\right),$$

and we can compute the score of relation paths by

$$\begin{aligned}
&\phi_{\text{BlockHolE}}(s, r_1/\ldots/r_k, o) \\
&\qquad = \mathrm{Re}(\mathbf{e}'^{\mathrm{T}}_s \mathbf{W}'_{r_1} \ldots \mathbf{W}'_{r_k} \overline{\mathbf{e}'_o}).
\end{aligned}$$

Since $\mathbf{W}'_{r_i}\mathbf{W}'_{r_j} \neq \mathbf{W}'_{r_j}\mathbf{W}'_{r_i}$ for $b \geq 2$, this scoring function respects the order of relations in $r_1/\ldots/r_k$.

## 6 Experiments

In this section, we report the results of empirical evaluation investigating the commutativity property of bilinear KGE models on the path QA task. As expected, the proposed BlockHolE model, which uses non-commutative relation matrices, outperformed commutative bilinear KGE models.

### 6.1 Dataset and evaluation protocol

The comparison of KGE models was performed in two path QA tasks: (i) ranking and (ii) binary classification tasks.

#### 6.1.1 Path QA ranking

For the path QA ranking task, we adopted the same protocol and dataset used by Guu et al. (2015). Table 2 shows the statistics of their dataset. The dataset consists of two parts, "Base" and "Path".

| | WN11 | | | | | | FB13 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Base | | Deduction | | Induction | | Base | | Deduction | | Induction | |
| | P@10 | MQ | P@10 | MQ | P@10 | MQ | P@10 | MQ | P@10 | MQ | P@10 | MQ |
| DistMult | 45.6 | 83.0 | 33.5 | 97.7 | 29.6 | 79.8 | 62.7 | 91.6 | 63.6 | 86.4 | 59.3 | 86.5 |
| ComplEx | 60.9 | 83.1 | 68.7 | 99.2 | 46.1 | 79.7 | 76.8 | 93.0 | 71.5 | 90.0 | 70.5 | 88.9 |
| RESCAL | 51.8 | 74.2 | 43.2 | 97.9 | 51.2 | 76.8 | 65.2 | 91.1 | 66.9 | 88.4 | 69.8 | 89.0 |
| $b = 2, m = 25$ | **80.9** | **83.4** | **70.2** | **99.5** | **54.9** | **81.0** | **79.2** | **93.2** | **75.0** | **91.5** | **71.3** | **90.0** |
| $b = 4, m = 25$ | 80.5 | 75.6 | 69.3 | 99.2 | 54.5 | 77.4 | 76.2 | 92.1 | 72.1 | 90.5 | 70.9 | 89.5 |

Table 3: Path QA ranking result: Comparing BlockHolE ($b = 2, m = 25$ and $b = 4, m = 25$) to other bilinear models. The dimension of the embedding space for DistMult, ComplEx and RESCAL was set to $n = 50$ as the result of grid search.

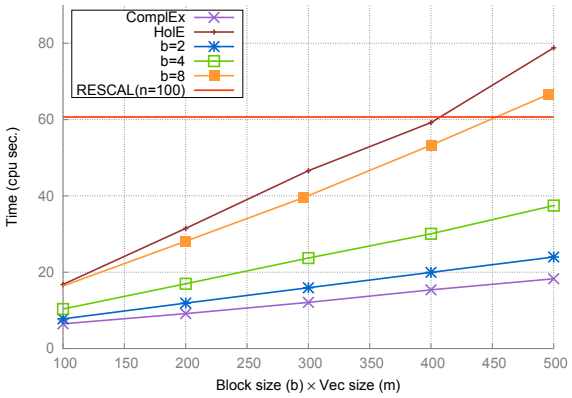


Figure 2: CPU run time per epoch of BlockHolE on WN11 Base with single CPU thread.

The Base part only contains facts (i.e., path queries with $k = 1$), and thus it is essentially for evaluating KGC performance. Its training samples constitute the observed facts $\mathcal{F}$, and the facts in the entire Base part (training/validation/test sets) make the ground truth facts $\mathcal{F}^*$.

The Path part contains path queries sampled from the same $\mathcal{F}$ and $\mathcal{F}^*$ as the Base part. The test samples in the Path part is divided into "deduction" and "induction" sets. In the "deduction" set, test samples were sampled from the Base training graph $\mathcal{G}(\mathcal{F})$. By contrast, in the "induction" set, the test samples were chosen from the ground truth graph $\mathcal{G}(\mathcal{F}^*)$ such that none of them have a corresponding path in $\mathcal{G}(\mathcal{F})$. Thus, the "induction" set is intended to measure how well a model generalizes to unobserved paths, whereas the "deduction" set is to test its ability to faithfully encode the observed training graph.

At the time of evaluation, for each a test sample $(s, r_1/\ldots/r_k, o)$, a *candidate set*

$$\mathcal{T}(r_k) = \{t : \exists h \in \mathcal{E}\ (h, r_k, t) \in \mathcal{F}^*\},$$

was first computed. In other words, the candidates are the entities for which $r_k$ (i.e., the last relation in the test query) takes as its object at least once in $\mathcal{F}^*$. Then, for each compared model, we made the ranking of the candidates entities in $\mathcal{T}(r_k)$ by the score $\phi(s, r_1/\ldots/r_k, e)$, where $\phi$ is learned by the model from the training set.

The quality of the ranking was measured by two evaluation metrics: averaged mean quantile (MQ) and P@10 (percentage of correct answers ranked in the top 10). For $q = s/p$ where $p = r_1/\ldots/r_k$, the correct answer set $[\![q]\!]$ is the set of all entities that can be reached from $s$ by traversing $p$ over $\mathcal{G}(\mathcal{F}^*)$. Formally, let $[\![s]\!] = \{s\}$, and the answer set can be recursively defined: $[\![q/r]\!] = \{t : \exists h \in [\![q]\!], (h, r, t) \in \mathcal{F}^*\}$. With these definitions, MQ is computed by the following formula:

$$\frac{|\{o' | o' \in \mathcal{N}(q) : \phi(s, p, o') \leq \phi(s, p, o)\}|}{|\mathcal{N}(q)|}, \quad (13)$$

where $\mathcal{N}(q) = \mathcal{T}(r_k) \setminus [\![q]\!]$ is the set of incorrect answers. Eq. (13) cannot be computed for queries with which $\mathcal{T}(r_k) = [\![q]\!]$, and these queries were excluded from evaluation. For further details, see the original paper by Guu et al. (2015).

### 6.1.2 Path QA classification

In the path QA classification task, we simply report classification accuracy. After the scoring function $\phi$ was trained with logistic regression, a path query $q = (s, r_1/\ldots/r_k, o)$ was classified as true if $\phi(q) \geq 0$, or false otherwise.

Since the test and validation sets of Path in Table 2 contain only correct queries, we sampled negative ones by the following procedure: For a correct query $q = (s, r_1/\ldots/r_k, o)$ $(k \geq 2)$, we generated its reverse relation path query $q' = (s, r_k/r_{k-1}/\ldots/r_1, o)$. If $q'$ does not exist in $\mathcal{G}(\mathcal{F}^*)$, we used it as a negative.

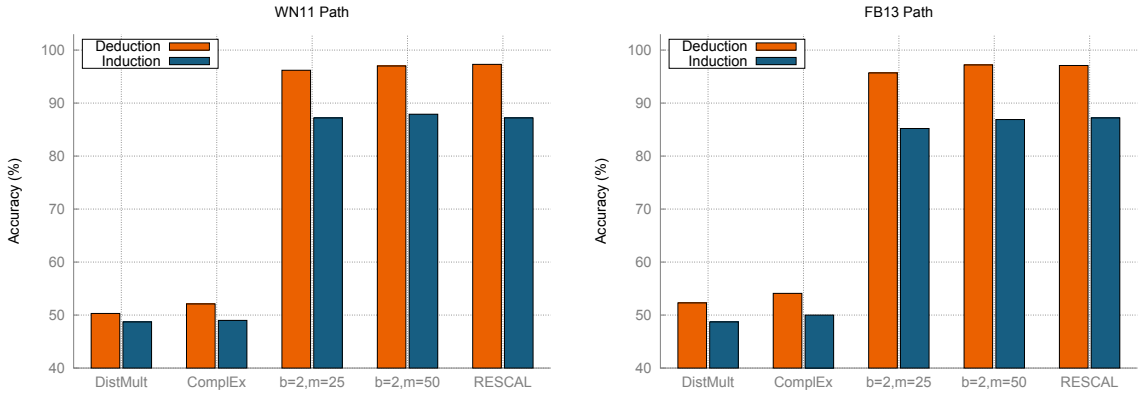


Figure 3: Path QA classification result comparing BlockHolE ($b = 2, m = 25$ and $b = 2, m = 50$) to DistMult, ComplEx and RESCAL models (all with $n = 50$ as the result of grid search).

### 6.2 Experiment setup

We compared BlockHolE with state-of-the-art bilinear KGE models: DistMult, RESCAL and ComplEx. We have implemented BlockHolE in Java. BlockHolE reduces to ComplEx when $b = 1$, and with the imaginary parts of parameters set to 0, it reduces to RESCAL when $m = 1$ and to DistMult when $b = 1$. For a fair run time comparison, however, we separately implemented RESCAL using jblas-1.2.4 for matrix computation. Through all experiments, we optimized the logistic loss with L2 regularization on the parameters $\Theta$:

$$\min_{\Theta} \sum_{(q,y)\in\mathcal{D}} \log\{1 + \exp(-y\phi(q;\Theta))\} + \lambda||\Theta||_2^2$$

where $y$ denotes the truth value of a query $q$ in a training data $\mathcal{D}$. Given a correct query $q = (s, r_1/\ldots/r_k, o)$, we generated negative samples by replacing $o$ with an entity randomly sampled from $\mathcal{E}$.

We selected the hyperparameters via grid search such that on the validation set they maximize classification accuracy in the path QA classification task and MQ in the path QA ranking task. For all models except BlockHolE, all combinations of $\lambda \in \{0.0001, 0\}$, learning rate $\eta \in \{0.005, 0.01, 0.025, 0.05\}$, and the embedding size $n \in \{50, 100, 150, 200\}$ were tried during grid search. For BlockHolE, all combinations of $(b, m) \in \{(2, 25), (2, 50), (2, 100), (4, 25), (4, 50), (8, 25)\}$, $\lambda \in \{0.0001, 0\}$ and $\eta \in \{0.005, 0.01, 0.025, 0.05\}$ were tried. The maximum number of training epochs was set to 500. The number of negatives generated per positive sample was 5 during training.

| Label | Relation Path | ComplEx | BlockHolE |
|---|---|---|---|
| + | */parents/religion/* | 96.7 | 100.0 |
| - | */religion/parents/* | 3.3 | 100.0 |

Table 4: Classification accuracy on selected queries.

### 6.3 Results

#### 6.3.1 Path QA ranking

Table 3 shows the results on the path QA ranking data. BlockHolE outperforms other bilinear KGE models considerably both on deductive and inductive test settings. These results strongly suggest that BlockHolE is more expressive in modeling path QA than DistMult and ComplEx, while effectively reducing redundant parameters in RESCAL which can cause model overfitting. Figure 2 shows the empirical scalability of BlockHolE. When $b$ is small, BlockHolE scales linearly in the dimension $n = bm$ of the embedding space.

#### 6.3.2 Path QA classification

Figure 3 shows the accuracy of path QA classification. DistMult and ComplEx were considerably worse than BlockHolE and RESCAL for both WN11 and FB13. This result confirms our claim: The non-commutativity of relation matrices plays a critical role in modeling path QA. The performance of BlockHolE ($b = 2, m = 25$) was comparable to that of RESCAL but the former was 12 times faster.

### 6.4 Analysis

The accuracies of BlockHolE and RESCAL on the path QA classification task were markedly better than those of DistMult and ComplEx. We analyzed the results further. We extracted all queries from

$\mathcal{G}(\mathcal{F}^*)$ of FB13 that consist of an interpretable relation path */parents/religion/* where $*$ denotes "can match any relation path". For such queries $(s, */\text{parents}/\text{religion}/*, o)$, we also generated meaningless queries $(s, */\text{religion}/\text{parents}/*, o)$ as negatives. Table 4 shows the classification accuracies of ComplEx and BlockHolE ($b = 2, m = 25$). The results clearly show that ComplEx cannot correctly answer the negative queries at all due to the lack of the non-commutative property.

## 7 Summary

In this paper, we have pointed out the problems of existing bilinear KGE models in path QA, and proposed a new model that overcomes these problems. This model, called BlockHolE, represents relations as block circulant matrices. As a result, it respects the order of relations in path queries, while enjoying linear-time computation of scoring functions when the number $b^2$ of blocks is sufficiently small. It generalizes HolE/ComplEx, and it can also be interpreted as an interpolation between RESCAL and HolE/ComplEx. Its effectiveness was shown empirically in path QA.

Our proposal can be useful in not only path QA but also many tasks such as associative rule mining (Yang et al., 2015), path regularization (Lin et al., 2015), and more complex QA (Hamilton et al., 2018), in which composite relations need to be embedded as a vector. Other future directions include reducing the increased parameters in the proposed block circulant matrices, such as by using multiplicative L1 regularization for ComplEx (Manabe et al., 2018).

### Acknowledgments

We thank anonymous reviewers for helpful comments. This work was partially supported by JSPS Kakenhi Grant Numbers 19H04173, 18K11457, and 18H03288.

## References

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems 26 (NIPS)*, pages 2787–2795.

Kelvin Guu, John Miller, and Percy Liang. 2015. Traversing knowledge graphs in vector space. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 318–327.

William L. Hamilton, Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec. 2018. Embedding logical queries on knowledge graphs. In *Advances in Neural Information Processing Systems 31 (NeurIPS)*, pages 2030–2041.

Katsuhiko Hayashi and Masashi Shimbo. 2017. On the equivalence of holographic and complex embeddings for link prediction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 2: Short Papers, pages 554–559.

Seyed Mehran Kazemi and David Poole. 2018. SimplE embedding for link prediction in knowledge graphs. In *Advances in Neural Information Processing Systems 31 (NeurIPS)*, pages 4284–4295.

Yankai Lin, Zhiyuan Liu, Huan-Bo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015. Modeling relation paths for representation learning of knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 705–714.

Hanxiao Liu, Yuexin Wu, and Yiming Yang. 2017. Analogical inference for multi-relational embeddings. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 2168–2178.

Hitoshi Manabe, Katsuhiko Hayashi, and Masashi Shimbo. 2018. Data-dependent learning of symmetric/anti-symmetric relations for knowledge base completion. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*, pages 3754–3761.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2016a. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2016b. Holographic embeddings of knowledge graphs. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI)*, pages 1955–1961.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 809–816.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pages 2071–2080.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

# Generating Questions for Knowledge Bases via Incorporating Diversified Contexts and Answer-Aware Loss

**Cao Liu**[1,2], **Kang Liu**[1,2], **Shizhu He**[1,2], **Zaiqing Nie**[3], **Jun Zhao**[1,2]
[1] National Laboratory of Pattern Recognition, Institute of Automation,
Chinese Academy of Sciences, Beijing, 100190, China
[2] University of Chinese Academy of Sciences, Beijing, 100049, China
[3] Alibaba AI Labs, Beijing, 100029, China
{cao.liu, kliu, shizhu.he, jzhao}@nlpr.ia.ac.cn
zaiqing.nzq@alibaba-inc.com

## Abstract

We tackle the task of question generation over knowledge bases. Conventional methods for this task neglect two crucial research issues: 1) the given predicate needs to be expressed; 2) the answer to the generated question needs to be definitive. In this paper, we strive toward the above two issues via incorporating diversified contexts and answer-aware loss. Specifically, we propose a neural encoder-decoder model with multi-level copy mechanisms to generate such questions. Furthermore, the answer aware loss is introduced to make generated questions corresponding to more definitive answers. Experiments demonstrate that our model achieves state-of-the-art performance. Meanwhile, such generated question can express the given predicate and correspond to a definitive answer.

## 1 Introduction

Question Generation over Knowledge Bases (KBQG) aims at generating natural language questions for the corresponding facts on KBs, and it can benefit some real applications. Firstly, KBQG can automatically annotate question answering (QA) datasets. Secondly, the generated questions and answers will be able to augment the training data for QA systems. More importantly, KBQG can improve the ability of machines to actively ask questions on human-machine conversations (Duan et al., 2017; Sun et al., 2018). Therefore, this task has attracted more attention in recent years (Serban et al., 2016; Elsahar et al., 2018).

Specifically, KBQG is the task of generating natural language questions according to the input facts from a knowledge base with triplet form, like <subject, predicate, object>. For example, as illustrated in Figure 1, KBQG aims at generating a question "Which city is Statue of Liberty located in?" (Q3) for the input factual triplet

| **Input** | <Statue of Liberty, location/containedby, ***New York City***> | | |
|---|---|---|---|
| **Output** | Matching Predicate | Definite Answer | Question |
| Q1 | × | - | Who **created** the Statue of Liberty? |
| Q2 | √ | × | **Where** is Statue of Liberty in? |
| Q3 | √ | √ | Which ***city*** is Statue of Liberty **located** in? |

Figure 1: Examples of KBQG. We aims at generating questions like Q3 which expresses (matches) the given predicate and refers to a definitive answer.

"<Statue of Liberty, location/containedby[1], New York City>". Here, the generated question is associated to the subject "*Statue of Liberty*" and the predicate `fb:location/containedby`) of the input fact, and the answer corresponds to the object "*New York City*".

As depicted by Serban et al. (2016), KBQG is required to transduce the triplet fact into a question about the subject and predicate, where the object is the correct answer. Therefore, it is a key issue for KBQG to correctly understand the knowledge symbols (subject, predicate and object in the triplet fact) and then generate corresponding text descriptions. More recently, some researches have striven toward this task, where the behind intuition is to construct implicit associations between facts and texts. Specifically, Serban et al. (2016) designed an encoder-decoder architecture to generate questions from structured triplet facts. In order to improve the generalization for KBQG, Elsahar et al. (2018) utilized extra contexts as input via distant supervisions (Mintz et al., 2009), then a decoder is equipped with attention and part-of-speech (POS) copy mechanism to generate questions. Finally, this model obtained significant improvements. Nevertheless, we observe that there are still two important research issues (RIs) which are not processed well or even neglected.

[1]We omit the domain of the predicate for sake of brevity.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2431–2441,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

***RI-1:*** *The generated question is required to* ***express the given predicate*** *in the fact*. For example in Figure 1, Q1 does not express (match) the predicate (`fb:location/containedby`) while it is expressed in Q2 and Q3. Previous work (Elsahar et al., 2018) usually obtained predicate textual contexts through distant supervision. However, the distant supervision is noisy or even wrong (e.g. "X is the husband of Y" is the relational pattern for the predicate `fb:marriage/spouse`, so it is wrong when "X" is a woman). Furthermore, many predicates in the KB have no predicate contexts. We make statistic in the resources released by Elsahar et al. (2018), and find that only 44% predicates have predicate textual context[2]. Therefore, it is prone to generate error questions from such without-context predicates.

***RI-2:*** *The generated question is required to contain a* ***definitive answer***. A definitive answer means that one question only associates with a determinate answer rather than alternative answers. As an example in Figure 1, Q2 may contain ambiguous answers since it does not express the refined answer type. As a result, different answers including "*United State*", "*New York City*", etc. may be correct. In contrast, Q3 refers to a definitive answer (the object "*New York City*" in the given fact) by restraining the answer type to a city. We believe that Q3, which expresses the given predicate and refers to a definitive answer, is a better question than Q1 and Q2. In previous work, Elsahar et al. (2018) only regarded a most frequently mentioned entity type as the textual context for the subject or object in the triplet. In fact, most answer entities have multiple types, where the most frequently mentioned type tends to be universal (e.g. a broad type "administrative region" rather than a refined type "US state" for the entity "*New York*"). Therefore, generated questions from Elsahar et al. (2018) may be difficult to contain definitive answers.

To address the aforementioned two issues, we exploit more **diversified contexts** for the given facts as textual contexts in an encoder-decoder model. Specifically, besides using predicate contexts from the distant supervision utilized by Elsahar et al. (2018), we further leverage the domain, range and even topic for the given predicate as contexts, which are off-the-shelf in KBs (e.g. the range and the topic for the predicate `fb:location/containedby` are "location" and "containedby", respectively[1]). Therefore, 100% predicates (rather than 44%[2] of those in Elsahar et al.) have contexts. Furthermore, in addition to the most frequently mentioned entity type as contexts used by Elsahar et al. (2018), we leverage the type that best describes the entity as contexts (e.g. a refined entity type[3] "US state" combines a broad type "administrative region" for the entity "New York"), which is helpful to refine the entity information. Finally, in order to make full use of these contexts, we propose context-augmented fact encoder and multi-level copy mechanism (KB copy and context copy) to integrate diversified contexts, where the multi-level copy mechanism can copy from KB and textual contexts simultaneously. For the purpose of further making generated questions correspond to definitive answers, we propose the **answer-aware loss** by optimizing the cross-entropy between the generated question and answer type words, which is beneficial to generate precise questions.

We conduct experiments on an open public dataset. Experimental results demonstrate that the proposed model using diversified textual contexts outperforms strong baselines (+4.5 BLEU4 score). Besides, it can further increase the BLEU score (+5.16 BLEU4 score) and produce questions associated with more definitive answers by incorporating answer-aware loss. Human evaluations complement that our model can express the given predicate more precisely.

In brief, our main contributions are as follows:

(1) We leverage diversified contexts and multi-level copy mechanism to alleviate the issue of incorrect predicate expression in traditional methods.

(2) We propose an answer-aware loss to tackle the issue that conventional methods can not generate questions with definitive answers.

(3) Experiments demonstrate that our model achieves state-of-the-art performance. Meanwhile, such generated question can express the given predicate and refer to a definitive answer.

## 2 Task Description

We leverage textual contexts concerned with the triplet fact to generate questions over KBs. The

[2]We map the "prop_text_evidence.csv" file to the "property.vocab" file in Elsahar et al. (2018)

[3]We obtain such representative entity types through the predicate `fb:topic/notable_types` in freebase.

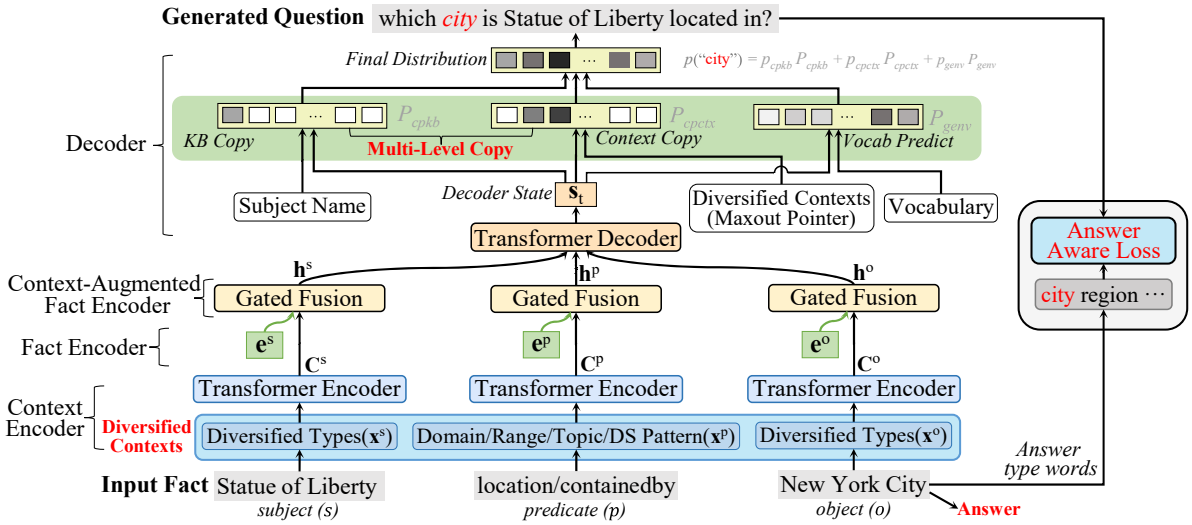


Figure 2: Overall structure of the proposed model for KBQG. A **context encoder** is firstly employed to encode each textual context (Sec. 3.1), where "Diversified Types" represents the subject (object) context, and "DS pattern" denotes the relational pattern from distant supervisions. At the same time, a **fact encoder** transforms the fact into low-dimensional representations (Sec. 3.2). The above two encoders are aggregated by the **context-augmented fact encoder** (Sec. 3.3). Finally, the aggregated representations are fed to the **decoder** (Sec. 3.4), where the decoder leverages multi-level copy mechanism (KB copy and context copy) to generate target question words.

task of KBQG can be formalized as follows:

$$P(Y|F) = \prod_{t=1}^{|Y|} P(y_t|y_{<t}, F, C) \quad (1)$$

where $F = (s, p, o)$ represents the subject ($s$), predicate ($p$) and object ($o$) of the input triplet, $C = \{\mathbf{x}^s, \mathbf{x}^p, \mathbf{x}^o\}$ denotes a set of additional textual contexts, $Y = (y_1, y_2, ..., y_{|Y|})$ is the generated question, $y_{<t}$ represents all previously generated question words before time step $t$.

## 3 Methodology

Our model extends the encoder-decoder architecture (Cho et al., 2014b) with three encoding modules and two copy mechanisms in the decoder. The model overview is shown in Figure 2 along with its caption. It should be emphasized that we additionally design an answer-aware loss to make the generated question associated with a definitive answer (Sec. 3.5.2).

### 3.1 Context Encoder

Inspired by the great success of transformer (Vaswani et al., 2017) in sequence modeling (Shen et al., 2018), we adopt a transformer encoder to encode each textual context separately. Take the subject context $\mathbf{x}^s$ as an example, $\mathbf{x}^s = (x_1^s, x_2^s, ..., x_{|s|}^s)$ is concatenated from diversified types for the subject, and $x_i^s$ is the $i$-th token in the subject context, $|s|$ stands for the length of the subject context. Firstly, $\mathbf{x}^s$ is mapped into a query matrix $\mathbf{Q}$, where $\mathbf{Q}$ is constructed by summing the corresponding token embeddings and segment embeddings. Similar to BERT (Devlin et al., 2019), segment embeddings are the same for tokens of $\mathbf{x}^s$ but different for that of $\mathbf{x}^p$ (predicate context) or $\mathbf{x}^o$ (object context). Based on the query matrix, transformer encoder works as follows:

$$\mathbf{Q}_j = \mathbf{Q}\mathbf{W}_j^Q, \mathbf{K}_j = \mathbf{K}\mathbf{W}_j^K, \mathbf{V}_j = \mathbf{V}\mathbf{W}_j^V \quad (2)$$

$$head_j = \text{softmax}(\mathbf{Q}_j\mathbf{K}_j^T/\sqrt{d/h})\mathbf{V}_j \quad (3)$$

$$\mathbf{H}^s = \text{Concat}(head_1, head_2, ..., head_h)\mathbf{W}_0 \quad (4)$$

$$\mathbf{N}^s = \text{LayerNorm}(\mathbf{Q} + \mathbf{H}^s) \quad (5)$$

$$\mathbf{C}^s = \max(0, \mathbf{N}^s\mathbf{W}_1 + b_1)\mathbf{W}_2 + b_2 \quad (6)$$

where $\mathbf{K}$ and $\mathbf{V}$ are the key matrix and value matrix, respectively. It is called self-attention because $\mathbf{K}$ and $\mathbf{V}$ are equal to the query matrix $\mathbf{Q} \in \mathbb{R}^{|s|,d}$ in the encoding stage, where $d$ represents the number of hidden units. And $h$ denotes the number of the heads in multi-head attention mechanism of the transformer encoder. It first projects the input matrixes ($\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$) into subspaces $h$ times mapped by different linear projections $\mathbf{W}_j^Q$, $\mathbf{W}_j^K$, $\mathbf{W}_j^V \in \mathbb{R}^{|s|,d/h}$ ($j = 1, 2, ..., h$) in Equation 2. And then $h$ projections perform the scaled dot-product attention to obtain the representation of each head in parallel (Equation 3). Representa-

tions for all parallel heads are concatenated together in Equation 4. After residual connection, layer normalization (Equation 5) and feed forward operation (Equation 6), we can obtain the subject context matrix $\mathbf{C}^s = \{\mathbf{c}_1^s, \mathbf{c}_2^s, ..., \mathbf{c}_{|s|}^s\} \in \mathbb{R}^{|s|,d}$.

Similarly, $\mathbf{C}^p$ and $\mathbf{C}^o$ are obtained from the same transformer encoder for the predicate and object, respectively.

### 3.2 Fact Encoder

In contrast to general Sequence-to-Sequence (Seq2Seq) model (Sutskever et al., 2014), the input fact is not a word sequence but instead a structured triplet $F = (s, p, o)$. We employ a fact encoder to transform each atom in the fact into a fixed embedding, and the embedding is obtained from a KB embedding matrix. For example, the subject embedding $\mathbf{e}^s \in \mathbb{R}^d$ is looked up from the KB embedding matrix $\mathbf{E}_f \in \mathbb{R}^{k,d}$, where $k$ represents the size of KB vocabulary, and the size of KB embedding is equal to the number of hidden units ($d$) in Equation 3. Similarly, the predicate embedding $\mathbf{e}^p$ and the object embedding $\mathbf{e}^o$ are mapped from the KB embedding matrix $\mathbf{E}_f$, where $\mathbf{E}_f$ is pre-trained using *TransE* (Bordes et al., 2013) to capture much more fact information in previous work (Elsahar et al., 2018). In our model, $\mathbf{E}_f$ can be pre-trained or randomly initiated (Details in Sec. 4.7.1).

### 3.3 Context-Augmented Fact Encoder

In order to combine both the context encoder information and the fact encoder information, we propose a context-augmented fact encoder which applies the gated fusion unit (Gong and Bowman, 2018) to integrate the context matrix and the fact embedding. For example, the subject context matrix $\mathbf{C}^s = \{\mathbf{c}_1^s, \mathbf{c}_2^s, ..., \mathbf{c}_{|s|}^s\}$ and the subject embedding vector $\mathbf{e}^s$ are integrated by the following gated fusion:

$$\mathbf{f} = \tanh(\mathbf{W}_f[\mathbf{c}^s, \mathbf{e}^s]) \quad (7)$$

$$\mathbf{g} = \text{sigmoid}(\mathbf{W}_g[\mathbf{c}^s, \mathbf{e}^s]) \quad (8)$$

$$\mathbf{h}^s = \mathbf{g} \odot \mathbf{f} + (1 - \mathbf{g}) \odot \mathbf{e}^s \quad (9)$$

where $\mathbf{c}^s$ is an attentive vector from $\mathbf{e}^s$ to $\mathbf{C}^s$, which is similar to Zhao et al. (2018). The attentive vector $\mathbf{c}^s$ is combined with original subject embedding $\mathbf{e}^s$ as a new enhanced representation $\mathbf{f}$ (Equation 7). And then a learnable gate vector, $\mathbf{g}$ (Equation 8), controls the information from $\mathbf{c}^s$ and $\mathbf{e}^s$ to the final augmented subject vector $\mathbf{h}^s \in \mathbb{R}^d$ (Equation 9), where $\odot$ denotes the element-wise multiplication. Similarly, the augmented predicate vector $\mathbf{h}^p$ and the augmented object vector $\mathbf{h}^o$ are calculated in the same way. Finally, the context-augmented fact representation $\mathbf{H}_f \in \mathbb{R}^{3,d}$ is the concatenation of augmented vectors as follows:

$$\mathbf{H}_f = [\mathbf{h}^s; \mathbf{h}^p; \mathbf{h}^o] \quad (10)$$

### 3.4 Decoder

The decoder aims at generating a question word sequence. As shown in Figure 2, we also exploit the transformer as the basic block in our decoder. Then we use a multi-level copy mechanism (KB copy and context copy), which allows copying from KBs and textual contexts.

Specifically, we first map the input of the decoder into an embedding representation by looking up word embedding matrix, then we use position embedding (Vaswani et al., 2017) to enhance sequential information. Compared with the transformer encoder in Sec. 3.1, transformer decoder has an extra sub-layer: a fact multi-head attention layer, which is similar to Equation 2-6, where the query matrix is initiated with previous decoder sub-layer while both the key matrix and the value matrix are the augmented fact representation $\mathbf{H}_f$. After feedforward and multiple transformer layers, we obtain the decoder state $\mathbf{s}_t$ at time step $t$, and then $\mathbf{s}_t$ could be leveraged to generate the target question sequence word by word.

As depicted in Figure 2, we propose multi-level copy mechanism to generate question words. At each time step $t$, given decoder state $\mathbf{s}_t$ together with input fact $F$, textual contexts $C$ and vocabulary $V$, the probabilistic function for generating any target question word $y_t$ is calculated as:

$$P(y_t|\mathbf{s}_t, y_{t-1}, F, C) = p_{genv} P_{genv}(y_t|\mathbf{s}_t, V) + p_{cpkb} P_{cpkb}(y_t|\mathbf{s}_t, F) + p_{cpctx} P_{cpctx}(y_t|\mathbf{s}_t, C) \quad (11)$$

$$p_{genv}, p_{cpkb}, p_{cpctx} = \text{softmax}([\mathbf{s}_t, \mathbf{y}_{t-1}]) \quad (12)$$

where $genv$, $cpkb$ and $cpctx$ denote the vocab generation mode, the KB copy mode and the context copy mode, respectively. In order to control the balance among different modes, we employ a 3-dimensional switch probability in Equation 12, where $\mathbf{y}_{t-1}$ is the embedding of previous generated word, $P_\cdot(\cdot|\cdot)$ indicates the probabilistic score function for generated target word of each mode. In the three probability score functions, $P_{vocab}$ is typically performed by a $softmax$ classifier over

a fixed vocabulary $V$ based on the word embedding similarity, and the details of $P_{cpkb}$ and $P_{cpctx}$ are in the following.

#### 3.4.1 KB Copy

Previous study found that most questions contain the subject name or its aligns in SimpleQuestion (Petrochuk and Zettlemoyer, 2018). However, the predicate name and object name hardly appear in the question. Therefore, we only copy the subject name in the KB copy, where $P_{cpkb}(y_t|\mathbf{s}_t, f)$, the probability of copying the subject name, is calculated by a neural network function with a multilayer perceptron (MLP) projected from $\mathbf{s}_t$.

#### 3.4.2 Context Copy

Elsahar et al. (2018) demonstrated the effectiveness of POS copy for the context. However, such a copy mechanism heavily relies on POS tagging. Inspired by the CopyNet (Gu et al., 2016), we directly copy words in the textual contexts $C$, and it does not rely on any POS tagging. Specifically, the input sequence $\chi$ for the context copy is the concatenation of all words in the textual contexts $C$. Unfortunately, $\chi$ is prone to contain repeated words because it consists of rich contexts for subject, predicate and object. The repeated words in the input sequence tend to cause repetition problems in output sequences (Tu et al., 2016). We adopt the maxout pointer (Zhao et al., 2018) to address the repetition problem. Instead of summing all the probabilistic scores for repeated input words, we limit the probabilistic score of repeated words to their maximum score as Equation 13:

$$P_{cpctx}(y_t|.)=\begin{cases}\max\limits_{m,\, where\ \chi_m=y_t} sc_{t,m} & y_t \in \chi \\ 0 & otherwise\end{cases} \quad (13)$$

where $\chi_m$ represents the $m$-th token in the input context sequence $\chi$, $sc_{t,m}$ is the probabilistic score of generating the token $\chi_m$ at time step $t$, and $sc_{t,m}$ is calculated by a softmax function over $\chi$.

### 3.5 Learning

#### 3.5.1 Question-Aware Loss

It is totally differential for our model to obtain question words, and it can be optimized in an end-to-end manner by back-propagation. Given the input fact $F$, additional textual context $C$ and target question word sequence $Y$, the object function is to optimize the following negative log-likelihood:

$$\mathcal{L}_{ques\_loss}=\frac{-1}{|Y|}\sum_{t=1}^{|Y|}\log[P(y_t|\mathbf{s}_t,y_{t-1},F,C)] \quad (14)$$

The question-aware loss $\mathcal{L}_{ques\_loss}$ does not require any additional labels to optimize because the three modes share a same softmax classifier to keep a balance (Equation 12), and they can learn to coordinate each other by minimizing $\mathcal{L}_{ques\_loss}$.

#### 3.5.2 Answer-Aware loss

It is able to generate questions similar to the labeled questions by optimizing the question-aware loss $\mathcal{L}_{ques\_loss}$. However, there is an ambiguous problem in the annotated questions where the questions have alternative answers rather than determinate answers (Petrochuk and Zettlemoyer, 2018). In order to make generated questions correspond to definitive answers, we propose a novel answer-aware loss. By answer-aware loss, we aim at generating an answer type word in the question, which contributes to generating a question word matching the answer type. Formally, the answer-aware loss is in the following:

$$\mathcal{L}_{ans\_loss} = \min_{a_n, a_n \in A} \min_{y_t, y_t \in Y} H_{a_n, y_t} \quad (15)$$

where $A = \{a_n\}_{n=1}^{|A|}$ is a set of answer type words. We treat object type words as the answer type words because the object is the answer. $H_{a_n,y_t}$ denotes the cross entropy between the answer type word $a_n$ and the generated question word $y_t$. Finally, the minimum cross entropy is regarded as the answer-aware loss $\mathcal{L}_{ans\_loss}$. Optimizing $\mathcal{L}_{ans\_loss}$ means that the model aims at generating an answer type word in the generated question sequence. For example, the model tends to generate Q3 rather than Q2 in Figure 1, because Q3 contains an answer type word—"city". Similarly, $\mathcal{L}_{ans\_loss}$ could be optimized in an end-to-end manner, and it can integrate $\mathcal{L}_{ques\_loss}$ by a weight coefficient $\lambda$ to the total loss as follows:

$$\mathcal{L}_{total\_loss} = \mathcal{L}_{ques\_loss} + \lambda \mathcal{L}_{ans\_loss} \quad (16)$$

## 4 Experiment

### 4.1 Experimental Settings

#### 4.1.1 Experimental Data Details

We conduct experiments on the SimpleQuestion dataset (Bordes et al., 2015), and there

are 75910/10845/21687 question answering pairs (QA-pairs) for training/validation/test. In order to obtain **diversified contexts**, we additionally employ domain, range and topic of the predicate to improve the coverage of predicate contexts. In this way, 100% predicates (rather than 44%[2] of those in Elsahar et al.) have contexts. For the subject and object context, we combine the most frequently mentioned entity type (Elsahar et al., 2018) with the type that best describe the entity[3]. The KB copy needs subject names as the copy source, and we map entities with their names similar to those in Mohammed et al. (2018). The data details are in Appendix A and submitted Supplementary Data.

#### 4.1.2 Evaluation Metrics

Following (Serban et al., 2016; Elsahar et al., 2018), we adopt some word-overlap based metrics (WBMs) for natural language generation including BLEU-4 (Papineni et al., 2002), $\text{ROUGE}_\text{L}$ (Lin, 2004) and METEOR (Denkowski and Lavie, 2014). However, such metrics still suffer from some limitations (Novikova et al., 2017). Crucially, it might be difficult for them to measure whether generated questions that express the given predicate and refer to definitive answers. To better evaluate generated questions, we run two further evaluations as follows.

(1) ***Predicate identification***: Following Mohammed et al. (2018), we employ annotators to judge whether the generated question expresses the given predicate in the fact or not. The score for predicate identification is the percentage of generated questions that express the given predicate.

(2) ***Answer coverage***: We define a novel metric called answer coverage to identify whether the generated question refers to a definitive answer. Specifically, answer coverage is obtained by automatically calculating the percentage of questions that contain answer type words, and answer type words are object contexts (entity types for the object are regarded as answer type words).

Furthermore, it is hard to automatically evaluate the naturalness of generated questions. Following Mohammed et al. (2018), we adopt human evaluation to measure the naturalness by a score of 0-5.

#### 4.1.3 Comparison with State-of-the-arts

We compare our model with following methods.

(1) *Template*: A baseline in Serban et al. (2016), it randomly chooses a candidate fact $F_c$ in the training data to generate the question, where $F_c$ shares the same predicate with the input fact.

(2) *Serban et al. (2016)*: We compare our methods with the single placeholder model, which performs best in Serban et al. (2016).

(3) *Elsahar et al. (2018)*: We compare our methods with the model utilizing copy actions, the best performing model in Elsahar et al. (2018). Although this model is designed to a zero-shot setting (for unseen predicates and entity type), it has good abilities to generate better questions (on known or unknown predicates and entity types) represented in the additional context input and SPO copy mechanism.

#### 4.1.4 Implementation Details

To make our model comparable to the comparison methods, we keep most parameter values the same as Elsahar et al. (2018). We utilize RMSProp algorithm with a decreasing learning rate (0.001), batch size (200) to optimize the model. The size of KB embeddings is 200, and KB embeddings are pre-trained by TransE (Bordes et al., 2013). The word embeddings are initialized by the pre-trained Glove word vectors[4] with 200 dimensions. In the transformer, we set the hidden units $d$ to 200, and we employ 4 paralleled attention head and a stack of 5 identical layers. We set the weight ($\lambda$) of the answer-aware loss to 0.2.

### 4.2 Overall Comparisons

| Model | BLEU4 | $\text{ROUGE}_\text{L}$ | METEOR |
|---|---|---|---|
| Template | 31.36 | * | 33.12 |
| Serban et al. (2016) | 33.32 | * | 35.38 |
| Elsahar et al. (2018) | 36.56 | 58.09 | 34.41 |
| Our Model | 41.09 | 68.68 | 47.75 |
| Our $\text{Model}_{\text{ans\_loss}}$ | **41.72** | **69.31** | **48.13** |

Table 1: Overall comparisons on the test data, where "ans_loss" represents answer-aware loss.

In Table 1, we compare our model with the typical baselines on word-overlap based metrics. It is evident that our model is remarkably better than baselines on all metrics, where the BLEU4 score increases 4.53 compared with the strongest baseline (Elsahar et al., 2018). Especially, incorporating answer-aware loss (the last line in Table 1) further improves the performance (+5.16 BLEU4).

### 4.3 Performances on Predicate Identification

To evaluate the ability of our model on predicate identification, we sample 100 generated question-

[4]http://nlp.stanford.edu/data/glove.6B.zip

| Model | Pred. Identification |
|---|---|
| Serban et al. (2016) | 53.5 |
| Elsahar et al. (2018) | 71.5 |
| Our Model$_{\text{ans_loss}}$ | **75.5** |

Table 2: Performances on predicate identification.

s from each model, and then two annotators are employed to judge whether the generated question expresses the given predicate. The Kappa for inter-annotator statistics is 0.611, and p-value for all scores is less than 0.005. As shown in Table 2, we can see that our model has a significant improvement in the predicate identification.

### 4.4 Performances on Answer Coverage — The Effectiveness of Answer-Aware Loss

| Model | $\lambda$ | BLEU4 | Ans$_{\text{cov}}$ |
|---|---|---|---|
| Elsahar et al. (2018) | 0 | 36.56 | 59.49 |
| Our Model | 0 | 41.09 | 61.65 |
| Our Model$_{\text{ans_loss}}$ | 0.05 | 41.55 | 62.27 |
| Our Model$_{\text{ans_loss}}$ | 0.2 | **41.72** | 64.23 |
| Our Model$_{\text{ans_loss}}$ | 0.5 | 41.57 | **65.50** |
| Our Model$_{\text{ans_loss}}$ | 1.0 | 41.34 | 65.25 |

Table 3: Performances on answer coverage, where "Ans$_{\text{cov}}$" denotes the metric of answer coverage. "$\lambda$" is the weight of the answer-aware loss in Equation 16.

Table 3 reports performances on BLUE4 and answer coverage (Ans$_{\text{cov}}$). We can obtain that:

(1) When answer-aware loss is not leveraged ($\lambda = 0$), advantages of performance are obvious in our model. Note that the answer coverage is 55.23 on the human-labeled questions. Although our model does not explicitly capture answer information, it still obtains a high answer coverage, which may be because our diversified contexts contain rich answer type words.

(2) To demonstrate the effectiveness of answer-aware loss, we set the weight of answer-aware loss ($\lambda$) to 0.05/0.2/0.5/1.0 (the last four lines in Table 3). It can be seen that our model, incorporating answer-aware loss, has a significant improvement on answer coverage while there is no performance degradation on BLEU4 compared with $\lambda = 0$, which indicates that answer-aware loss contributes to generating better questions. Especially, the generated questions are more precise because they refer to more definitive answers with high Ans$_{\text{cov}}$.

(3) It tends to correspond to alternative answers (object in the triplet fact) for some predicates such as `fb:location/containedby`, while other predicates (e.g. `fb:person/gender`) may refer to a definitive answer. To investigate our model, by incorporating answer-aware loss, does not generate an answer type word in a mandatory way, we found 20.5% predicate corresponds to the generated questions without answer type words when our model obtains the highest Ans$_{\text{cov}}$ ($\lambda$=0.5), and it is very close to 21.7% for the one in human-annotated questions. This demonstrates that the answer-aware loss does not force all predicates to generate questions with answer type words.

### 4.5 Ablation Study

| Model | BLEU4 | ROUGE$_{\text{L}}$ | METEOR |
|---|---|---|---|
| Our Model$_{\text{ans_loss}}$ | **41.72** | **69.31** | **48.13** |
| w/o context copy | 41.27 | 68.36 | 47.54 |
| w/o KB copy | 41.04 | 68.66 | 47.72 |
| w/o answer-aware loss | 41.09 | 68.68 | 47.75 |
| w/o diversified contexts | 40.53 | 68.52 | 47.66 |

Table 4: Ablation study by removing the main components, where "w/o" means without, and "w/o diversified contexts" represents that diversified contexts are replaced by contexts used in Elsahar et al. (2018).

In order to validate the effectiveness of model components, we remove some important components in our model, including context copy, KB copy, answer-aware loss and diversified contexts. The results are shown in Table 4. We can see that removing any component brings performance decline on all metrics. It demonstrates that all these components are useful. Specifically, the last line in Table 4, replacing diversified contexts with contexts used in Elsahar et al. (2018), has more obvious performance degradation.

### 4.6 Performances on Naturalness

| Model | Naturalness |
|---|---|
| Serban et al. (2016) | 2.96 |
| Elsahar et al. (2018) | 2.23 |
| Our Model$_{\text{ans_loss}}$ | **3.56** |

Table 5: Performances on naturalness.

Human evaluation is important for generated questions. Following Elsahar et al. (2018), we sample 100 questions from each system, and then two annotators measure the naturalness by a score of 0-5. The Kappa coefficient for inter-annotator is 0.629, and p-value for all scores is less than 0.005. As shown in Table 5, Elsahar et al. (2018) perform poorly on naturalness, while our model obtains the

highest score on naturalness, which demonstrates our model can deliver more natural questions than baselines.

### 4.7 Discussion

#### 4.7.1 Without Pre-trained KB Embeddings

| Model | TransE | BLEU4 | $\text{ROUGE}_\text{L}$ | METEOR |
|---|---|---|---|---|
| Elsahar et al. (2018) | True | 36.56 | 58.09 | 34.41 |
| Elsahar et al. (2018) | False | 33.67 | 55.57 | 33.20 |
| Our $\text{Model}_{\text{ans\_loss}}$ | True | 41.72 | 69.31 | 48.13 |
| Our $\text{Model}_{\text{ans\_loss}}$ | False | 41.55 | 68.59 | 47.52 |

Table 6: Performances of whether using the pre-trained KB embedding by transE.

Pre-trained KB embeddings may provide rich structured relational information among entities. However, it heavily relies on large-scale triplets, which is time and resource-intensive. To investigate the effectiveness of pre-trained KB embedding for KBQG, we report the performance of KBQG whether using pre-trained KB embeddings by simply applying TransE. Table 6 shows that the performance of KBQG is degraded without TransE embeddings. In comparison, Elsahar et al. (2018) obtain obvious degradation on all metrics while there is only a slight decline in our model. We believe that it may owe to the context-augmented fact encoder since our model drops to 40.87 on the BLEU4 score without context-augmented fact encoder and transE embeddings.

#### 4.7.2 The Effectiveness of Generated Questions for Enhancing Question Answering over Knowledge Bases

| Data Type | Accuracy |
|---|---|
| human-labeled data | 68.97 |
| + gen_data (Serban et al., 2016) | 68.53 |
| + gen_data (Elsahar et al., 2018) | 69.13 |
| + gen_data (Our $\text{Model}_{\text{ans\_loss}}$) | **69.57** |

Table 7: Performances of generated questions for QA.

Previous experiments demonstrate that our model can deliver more precise questions. To further prove the effectiveness of our model, we will see how useful the generated questions are for training a question answering system over knowledge bases. Specifically, we combine human-labeled data with the same amount of model-generated data to a typical QA system (Mohammed et al., 2018). The accuracy of QA is shown in Table 7. We can observe that adding generative questions may weaken the performance of QA (drop from 68.97 to 68.53 in Table 7). Our generated questions achieve the best performance on the QA system. It indicates that our model generates more precise question and has improved QA performances greatly.

#### 4.7.3 Speed

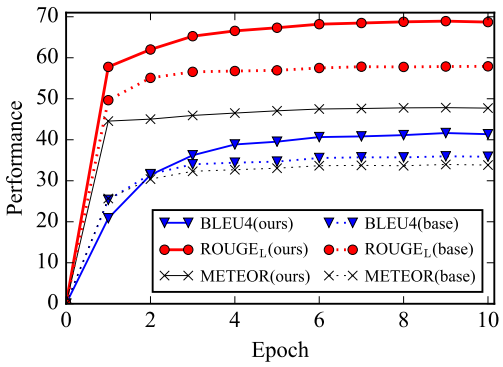


Figure 3: Performance on valid data through epochs, where "base" is the method in Elsahar et al. (2018).

In order to further explore the convergence speed, we plot the performances on valid data through epochs in Figure 3. Our model has much more information to learn, and it may have a bad impact on the convergence speed. Nevertheless, our model can copy KB elements and textual context simultaneously, which may accelerate the convergence speed. As demonstrated in Figure 3, our model achieves the best performances on almost epochs. After about 6 epochs, performances on our model become stable and convergent.

#### 4.7.4 Case Study

Figure 4 lists referenced question and generated questions by different models. It can be seen that our generated questions can better express the target predicate such as ID 1 (marked as underline). In ID 2, although all questions express the target predicate correctly, only our question refers to a definitive answer since it contains an answer type word "city" (marked as **bold**). It should be emphasized that the questions, generated by our method with answer-aware loss, do not always contain answer type words (ID 1 and 3).

## 5 Related Work

Our work is inspired by a large number of successful applications using neural encoder-decoder

| ID | Model | Question |
|---|---|---|
| 1 | Reference | what is the origin of *Kate Bush ?* |
| | Serban et al. | where is catherine bush buried ? |
| | Elsahar et al. | what is the artist of catherine bush ? |
| | Ours | what is the origin of the artist *Kate Bush* ? |
| 2 | Reference | what area contains *River Yare* ? |
| | Serban et al. | where is the *River Yare* ? |
| | Elsahar et al. | where is the *River Yare* located ? |
| | Ours | what **city** is *River Yare* in ? |
| 3 | Reference | who composed *Bien O Mal* ? |
| | Serban et al. | who is the composer of *Bien O Mal* ? |
| | Elsahar et al. | who is the composer of the song *Bien O Mal* ? |
| | Ours | who composed *Bien O Mal* ? |

Figure 4: Examples of questions by different models.

frameworks on NLP tasks such as machine translation (Cho et al., 2014a) and dialog generation (Vinyals and Le, 2015). Our work is also inspired by the recent work for KBQG based on encoder-decoder frameworks. Serban et al. (2016) first proposed a neural network for mapping KB facts into natural language questions. To improve the generalization, Elsahar et al. (2018) introduced extra contexts for the input fact, which achieved significant performances. However, these contexts may make it difficult to generate questions that express the given predicate and associate with a definitive answer. Therefore, we focus on the two research issues: expressing the given predicate and referring to a definitive answer for generated questions.

Moreover, our work also borrows the idea from copy mechanisms. Point network (Vinyals et al., 2015) predicted the output sequence directly from the input, and it can not generate new words while CopyNet (Gu et al., 2016) combined copying and generating. Bao et al. (2018) proposed to copy elements in the table (KB). Elsahar et al. (2018) exploited POS copy action to better capture textual contexts. To incorporate advantages from above copy mechanisms, we introduce KB copy and context copy which can copy KB element and textual context, and they do not rely on POS tagging.

## 6 Conclusion and Future Work

In this paper, we focus on two crucial research issues for the task of question generation over knowledge bases: generating questions that express the given predicate and refer to definitive answers rather than alternative answers. For this purpose, we present a neural encoder-decoder model which integrates diversified off-the-shelf contexts and multi-level copy mechanisms. Moreover, we design an answer-aware loss to generate questions that refer to definitive answers. Experiments show that our model achieves state-of-the-art performance on automatic and manual evaluations.

For future work, we investigate error cases by analyzing the error distributions of 100 examples. We find that most generated questions (51%) are judged by the human to correctly express the input facts, but they unfortunately obtain low scores on the widely used metrics. It implies that it is still intractable to evaluate generated questions. Although we additionally evaluate on predicate identification and answer coverage, these metrics may be coarse and deserve further study.

## Acknowledgments

This work is supported by the National Natural Science Foundation of China (No.61533018), the Natural Key R&D Program of China (No.2018YFC0830101), the National Natural Science Foundation of China (No.61702512) and the independent research project of National Laboratory of Pattern Recognition. This work was also supported by CCF-Tencent Open Research Fund.

## References

Jun-Wei Bao, Duyu Tang, Nan Duan, Zhao Yan, Yuanhua Lv, Ming Zhou, and Tiejun Zhao. 2018. Table-to-text: Describing table region with natural language. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5020–5027.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *CoRR*, abs/1506.02075.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2787–2795. Curran Associates, Inc.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder-decoder

approaches. In *SSST@EMNLP*, pages 103–111. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 376–380. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Nan Duan, Duyu Tang, Peng Chen, and Ming Zhou. 2017. Question generation for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 866–874. Association for Computational Linguistics.

Hady Elsahar, Christophe Gravier, and Frederique Laforest. 2018. Zero-shot question generation from knowledge graphs for unseen predicates and entity types. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 218–228. Association for Computational Linguistics.

Yichen Gong and Samuel Bowman. 2018. Ruminating reader: Reasoning with gated multi-hop attention. In *Proceedings of the Workshop on Machine Reading for Question Answering*, pages 1–11. Association for Computational Linguistics.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, pages 1003–1011, Stroudsburg, PA, USA. Association for Computational Linguistics.

Salman Mohammed, Peng Shi, and Jimmy Lin. 2018. Strong baselines for simple question answering over knowledge graphs with and without neural networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 291–296. Association for Computational Linguistics.

Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. 2017. Why we need new evaluation metrics for nlg. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2241–2252, Copenhagen, Denmark. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

ael Petrochuk and Luke Zettlemoyer. 2018. Simplequestions nearly solved: A new upperbound and baseline approach. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 554–558. Association for Computational Linguistics.

Iulian Vlad Serban, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. 2016. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 588–598. Association for Computational Linguistics.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018. Disan: Directional self-attention network for rnn/cnn-free language understanding. In *AAAI Conference on Artificial Intelligence*.

Xingwu Sun, Jing Liu, Yajuan Lyu, Wei He, Yanjun Ma, and Shi Wang. 2018. Answer-focused and position-aware neural question generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3930–3939. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of Advances in neural information processing systems*, pages 3104–3112.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 76–85. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2692–2700. Curran Associates, Inc.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. Cite arxiv:1506.05869Comment: ICML Deep Learning Workshop 2015.

Yao Zhao, Xiaochuan Ni, Yuanyuan Ding, and Qifa Ke. 2018. Paragraph-level neural question generation with maxout pointer and gated self-attention networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3901–3910. Association for Computational Linguistics.

# Multi-Task Learning for Conversational Question Answering over a Large-Scale Knowledge Base

**Tao Shen**[1]*, **Xiubo Geng**[2], **Tao Qin**[2], **Daya Guo**[3], **Duyu Tang**[2], **Nan Duan**[2], **Guodong Long**[1] and **Daxin Jiang**[2]
[1]Centre for AI, School of Computer Science, FEIT, University of Technology Sydney
[2]Microsoft, Beijing, China
[3]The School of Data and Computer Science, Sun Yat-sen University
tao.shen@student.uts.edu.au, guodong.long@uts.edu.au
{xiubo.geng,taoqin,dutang,nanduan,djiang}@microsoft.com
guody5@mail2.sysu.edu.cn

## Abstract

We consider the problem of conversational question answering over a *large-scale* knowledge base. To handle huge entity vocabulary of a large-scale knowledge base, recent neural semantic parsing based approaches usually decompose the task into several subtasks and then solve them sequentially, which leads to following issues: 1) errors in earlier subtasks will be propagated and negatively affect downstream ones; and 2) each subtask cannot naturally share supervision signals with others. To tackle these issues, we propose an innovative multi-task learning framework where a pointer-equipped semantic parsing model is designed to resolve coreference in conversations, and naturally empower joint learning with a novel type-aware entity detection model. The proposed framework thus enables shared supervisions and alleviates the effect of error propagation. Experiments on a large-scale conversational question answering dataset containing 1.6M question answering pairs over 12.8M entities show that the proposed framework improves overall F1 score from 67% to 79% compared with previous state-of-the-art work.

## 1 Introduction

Recent decades have seen the development of AI-driven personal assistants (e.g., Siri, Alexa, Cortana, and Google Now) that often need to answer factorial questions. Meanwhile, large-scale knowledge base (KB) like DBPedia (Auer et al., 2007) or Freebase (Bollacker et al., 2008) has been built to store world's facts in a structure database, which is used to support open-domain question answering (QA) in those assistants.

Neural semantic parsing based approach (Jia and Liang, 2016; Reddy et al., 2014; Dong and Lapata, 2016; Liang et al., 2016; Dong and Lapata, 2018; Guo et al., 2018) is gaining rising attention for knowledge-based question answer (KB-QA) in recent years since it does not rely on handcrafted features and is easy to adapt across domains. Traditional approaches usually retrieve answers from a small KB (e.g., small table) (Jia and Liang, 2016; Xiao et al., 2016) and are difficult to handle large-scale KBs. Many recent neural semantic parsing based approaches for KB-QA take a stepwise framework to handle this issue. For example, Liang et al. (2016), Dong and Lapata (2016), and Guo et al. (2018) first use an entity linking system to find entities in a question, and then learn a model to map the question to logical form based on that. Dong and Lapata (2018) decompose the semantic parsing process into two stages. They first generate a rough sketch of logical form based on low-level features, and then fill in missing details by considering both the question and the sketch.

However, these stepwise approaches have two issues. First, errors in upstream subtasks (e.g., entity detection and linking, relation classification) are propagated to downstream ones (e.g., semantic parsing), resulting in accumulated errors. For example, case studies in previous works (Yih et al., 2015; Dong and Lapata, 2016; Xu et al., 2016; Guo et al., 2018) show that entity linking error is one of the major errors leading to wrong predictions in KB-QA. Second, since models for the subtasks are learned independently, the supervision signals cannot be shared among the models for mutual benefits.

To tackle issues mentioned above, we propose a novel multi-task semantic parsing framework for KB-QA. Specifically, an innovative pointer-equipped semantic parsing model is first designed for two purposes: 1) built-in pointer network toward positions of entity mentions in the question

* Work done while the author was an intern at Microsoft, Beijing, China.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2442–2451, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| Alias | Operator | Comments |
|---|---|---|
| A1/2/3 | $start \rightarrow set/num/bool$ | |
| A4 | $set \rightarrow \text{find}(set, p)$ | set of entities with a predicate $p$ edge to entity $e$ |
| A5 | $num \rightarrow \text{count}(set)$ | number of distinct elements in the input $set$ |
| A6 | $bool \rightarrow \text{in}(e, set)$ | whether the entity $e$ in $set$ or not |
| A7 | $set \rightarrow \text{union}(set_1, set_2)$ | $set_1 \cup set_2$ |
| A8 | $set \rightarrow \text{inter}(set_1, set_2)$ | $set_1 \cap set_2$ |
| A9 | $set \rightarrow \text{diff}(set_1, set_2)$ | $set_1$ - $set_2$ |
| A10 | $set \rightarrow \text{large}(set, p, num)$ | subset of set linking to more than $num$ entities with predicate $p$ |
| A11 | $set \rightarrow \text{less}(set, p, num)$ | subset of set linking to less than $num$ entities with predicate $p$ |
| A12 | $set \rightarrow \text{equal}(set, p, num)$ | subset of set linking to $num$ entities with predicate $p$ |
| A13 | $set \rightarrow \text{argmax}(set, p)$ | subset of set linking to most entities with predicate $p$ |
| A14 | $set \rightarrow \text{argmin}(set, p)$ | subset of set linking to least entities with predicate $p$ |
| A15 | $set \rightarrow \text{filter}(tp, set)$ | subset where entity $e$ in set and belong to entity type $tp$ |
| A16 | $num \rightarrow u\_num$ | transform number in utterance $u\_num$ to intermediate number $num$ |
| A17 | $set \rightarrow \text{set}(e)$ | |
| A18/19/20/21 | $e/p/tp/u\_num \rightarrow$ constant | *instantiation for $e$, $p$, $tp$, $u\_num$ |

Table 1: Brief grammar definitions for logical form generation. *instantiation of entity $e$, predicate $p$, type $tp$, number-in-question $u\_num$, by corresponding constant parsed from the question.

can naturally empower multi-task learning with conjunction of upstream sequence labeling subtask, i.e., entity detection; and 2) it explicitly takes into account the context of entity mentions by using the supervision of the pointer network. Besides, a type-aware entity detection method is proposed to produce accurate entity linking results, in which, a joint prediction space combining entity detection and entity type is employed, and the predicted type is then used to filter entity linking results during inference phase.

The proposed framework has certain merits.

First, since the two subtasks, i.e., pointer-equipped semantic parsing and entity detection, are closely related, learning them within a single model simultaneously makes the best of supervisions and improves performance of KB-QA task.

Second, considering entity type prediction is crucial for entity linking, our joint learning framework combining entity mention detection with type prediction leverages contextual information, and thus further reduces errors in entity linking.

Third, our approach is naturally beneficial to coreference resolution for conversational QA due to rich contextual features captured for entity mention, compared to previous works directly employing low-level features (e.g., mean-pooling over word embeddings) as the representation of an entity. This is verified via our experiments in §4.2.

We evaluate the proposed framework on the CSQA (Saha et al., 2018) dataset, which is the largest public dataset for complex conversational question answering over a large-scale knowledge base. Experimental results show that the overall F1 score is improved by 12.56% compared with strong baselines, and the improvements are consistent for all question types in the dataset.

## 2 Task Definition

In this work, we target the problem of conversational question answering over a large-scale knowledge base. Formally, in training data, question $U$ denotes an user utterance from a dialog, which is concatenated dialog history for handling ellipsis or coreference in conversations, and the question is labeled with its answer $A$. Besides, "IOB" (Insider-Outside-Beginning) tagging and entities linking to KB are also labeled for entity mentions in $U$ to train an entity detection model.

We employ a neural semantic parsing based approach to tackle the problem. That is, given a question, a semantic parsing model is used to produce a logical form which is then executed on the KB to retrieve an answer. We decompose the approach into two subtasks, i.e., entity detection for entity linking and semantic parsing for logical form generation. The former employs IOB tagging and corresponding entities as supervision, while the latter uses a gold logical form as supervision, which may be obtained by conducting intensive BFS[1] over KB if only final answers (i.e., weak supervision) are provided.

## 3 Approach

This section begins with a description of grammars and logic forms used in this work. Then, the proposed model is presented, and finally, model's training and inference are introduced.

[1] Breadth-first search with limited buffer (Guo et al., 2018)

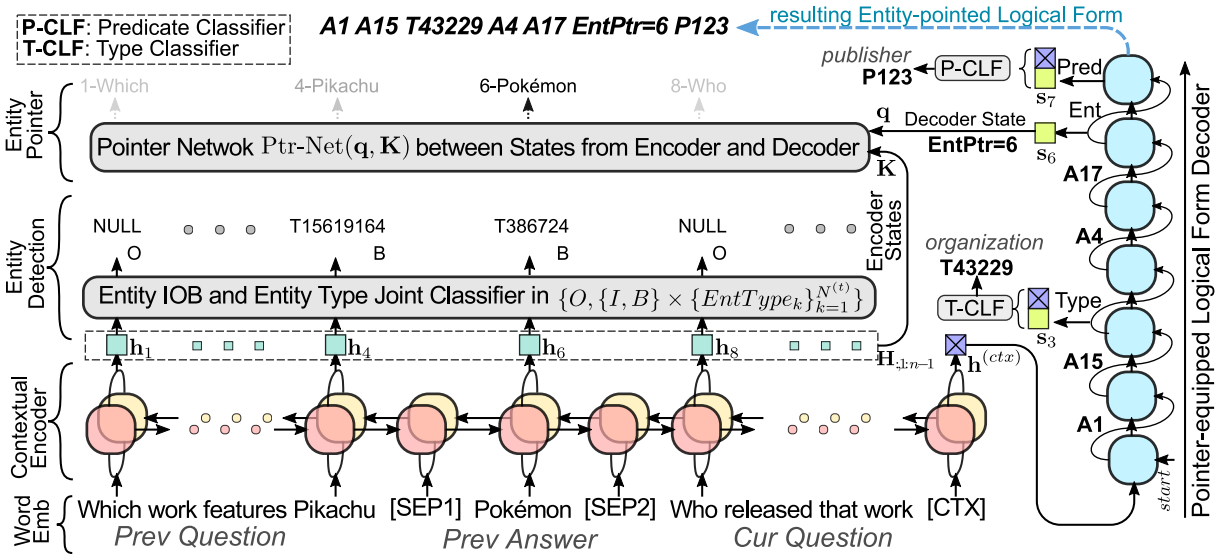


Figure 1: Proposed **M**ulti-t**a**sk **S**emantic **P**arsing (MaSP) model. Note that P* and T* are predicate and entity type ids in Wikidata where entity type id originally starts with Q but is replaced with T for clear demonstration.

### 3.1 Grammar and Logical Form

**Grammar** We use similar grammars and logical forms as defined in Guo et al. (2018), with minor modification for better adaptation to the CSQA dataset. The grammars are briefly summarized in Table 1, where each operator consists of three components: semantic category, a function name, and a list of arguments with specified semantic categories. Semantic categories can be classified into two groups here w.r.t. the ways for instantiation: one is referred to as *entry* semantic category (i.e., $\{e, p, tp, u\_num\}$ for entities, predicates, types, numbers) whose instantiations are constants parsed from a question, and another is referred to as *intermediate* semantic category (i.e., $\{start, set, num, bool\}$) whose instantiation is the output of an operator execution.

**Logical Form** A KB-executable logical form is intrinsically formatted as an ordered tree where the root is the semantic category $start$, each child node is constrained by the nonterminal (i.e., the un-instantiated semantic category in parenthesis) of its parent operator, and leaf nodes are instantiated entry semantic categories, i.e., constants.

To make the best of well-performed sequence to sequence (seq2seq) models (Vaswani et al., 2017; Bahdanau et al., 2015) as a base for semantic parsing, we represent a tree-structured logical form as a sequence of operators and constants via depth-first traversal over the tree. Note, given guidance of grammars, we can recover corresponding tree structure from a sequence-formatted logical form.

### 3.2 Proposed Model

The structure of our proposed **M**ulti-t**a**sk **S**mantic **P**arsing (MaSP) model is illustrated in Figure 1. The model consists of four components: i.e., word embedding, contextual encoder, entity detection and pointer-equipped logical form decoder.

#### 3.2.1 Embedding and Contextual Encoder

To handle ellipsis or coreference in conversations, our model takes current user question combined with dialog history as the input question $U$. In particular, all those sentences are concatenated with a *[SEP]* separated, and then a special token *[CTX]* is appended. We apply wordpiece tokenizing (Wu et al., 2016) method, and then use a word embedding method (Mikolov et al., 2013) to transform the tokenized question to a sequence of low-dimension distributed embeddings, i.e., $\boldsymbol{X} = [\boldsymbol{x_1}, \cdots, \boldsymbol{x_n}] \in \mathbb{R}^{d_e \times n}$ where $d_e$ denotes embedding size and $n$ denotes question length.

Given word embeddings $\boldsymbol{X}$, we use stacked two-layer multi-head attention mechanism in the Transformer (Vaswani et al., 2017) with learnable positional encodings as an encoder to model contextual dependencies between tokens, which results in context-aware representations $\boldsymbol{H} = [\boldsymbol{h_1}, \cdots, \boldsymbol{h_n}] \in \mathbb{R}^{d_e \times n}$. And, contextual embedding for token *[CTX]* is used as the semantic representation for entire question, i.e., $\boldsymbol{h^{(ctx)}} \triangleq \boldsymbol{h_n}$

### 3.2.2 Pointer-Equipped Decoder

Given contextual embeddings $\boldsymbol{H}$ of a question, we employ stacked two-layer masked attention mechanism in (Vaswani et al., 2017) as the decoder to produce sequence-formatted logical forms.

In each decoding step, the model first predicts a token from a small decoding vocabulary $\mathbb{V}^{(dec)}$ = $\{start, end, e, p, tp, u\_num, \text{A1}, \cdots, \text{A21}\}$ , where $start$ and $end$ indicate the start and end of decoding, $A1, \cdots, A21$ are defined in Table 1, and $e$, $p$, $tp$ and $u\_num$ denote entity, predicate, type and number entries respectively. A neural classifier is established to predict current decoding token, which is formally denoted as

$$\boldsymbol{p_j^{(tk)}} = \text{softmax}(\text{FFN}(\boldsymbol{s_j}; \theta^{(tk)})), \quad (1)$$

where $\boldsymbol{s_j}$ is decoding hidden state of current (i.e., $j$-th) step, $\text{FFN}(\cdot;\theta)$ denotes a $\theta$-parameterized two-layer feed forward network with an activation function inside, and $\boldsymbol{p_j^{(tk)}} \in \mathbb{R}^{|\mathbb{V}^{(dec)}|}$ is a predicted distribution over $\mathbb{V}^{(dec)}$ to score candidates[2].

Then, a $\text{FFN}(\cdot)$ or a pointer network (Vinyals et al., 2015) is utilized to predict instantiation for entry semantic category (i.e., $e$, $p$, $tp$ or $u\_num$ in $\mathbb{V}^{(vec)}$) if it is necessary.

- For predicate $p$ and type $tp$, two parameter-untied $\text{FFN}(\cdot)$ are used as

$$\boldsymbol{p_j^{(p)}} = \text{softmax}(\text{FFN}([\boldsymbol{s_j}; \boldsymbol{h^{(ctx)}}]; \theta^{(p)})), \quad (2)$$

$$\boldsymbol{p_j^{(t)}} = \text{softmax}(\text{FFN}([\boldsymbol{s_j}; \boldsymbol{h^{(ctx)}}]; \theta^{(t)})), \quad (3)$$

  where $\boldsymbol{h^{(ctx)}}$ is semantic embedding of entire question, $\boldsymbol{s_j}$ is current hidden state, $\boldsymbol{p_j^{(p)}} \in \mathbb{R}^{N^{(p)}}$ and $\boldsymbol{p_j^{(t)}} \in \mathbb{R}^{N^{(t)}}$ are predicted distributions over the predicate and type instantiation candidates respectively, and $N^{(p)}$ and $N^{(t)}$ are the numbers of distinct predicates and types in the knowledge base.

- For entity $e$ and number $u\_num$, two parameter-untied pointer-networks (Vinyals et al., 2015) with learnable bilinear layer are employed to point toward the targeted entity[3] and number, which are defined as follows.

$$\boldsymbol{p_j^{(e)}} = \text{softmax}(\boldsymbol{s_j^T W^{(e)} H}_{:,1:n-1}), \quad (4)$$

$$\boldsymbol{p_j^{(n)}} = \text{softmax}(\boldsymbol{s_j^T W^{(n)} H}_{:,1:n-1}), \quad (5)$$

  where $\boldsymbol{H}_{:,1:n-1}$ is contextual embedding of tokens in the question except *[CTX]*, $\boldsymbol{W^{(e)}}$ and $\boldsymbol{W^{(n)}}$ are weights of pointer-network for entity and number, $\boldsymbol{p_j^{(e)}}, \boldsymbol{p_j^{(n)}} \in \mathbb{R}^{n-1}$ are the resulting distributions over positions of input question, and $n$ is the length of the question.

The pointer network is also used for semantic parsing in (Jia and Liang, 2016), where the pointer aims at copying out-of-vocabulary words from a question over small-scale KB. Different from that, the pointer used here aims at locating the targeted entity and number in a question, which has two advantages. First, it handles the coreference problem by considering the context of entity mentions in the question. Second, it solves the problem caused by huge entity vocabulary, which reduces the size of decoding vocabulary from several million (i.e., the number of entities in KB) to several dozen (i.e., the length of the question).

### 3.2.3 Entity Detection and Linking

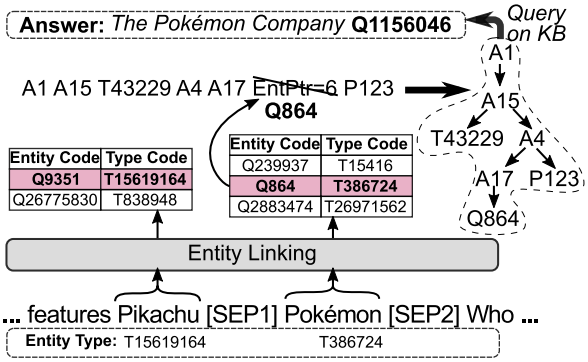


Figure 2: Transformation from entity-pointed logical form to KB-executable logical form for KB querying.

To map the pointed positions to entities in KB, our model also detects entity mentions for the input question, as shown as the "Entity Detection" part of Figure 1.

We observe that multiple entities in a large-scale KB usually have same entity text but different types, leading to named entity ambiguity. Therefore, we design a novel type-aware entity detection module in which the prediction is fulfilled in a joint space of IOB tagging and corresponding entity type for disambiguation. Particularly, the prediction space is defined as $\mathbb{E} = \{O, \{I, B\} \times \{ET_k\}_{k=1}^{N^{(t)}}\}$ where $ET_k$ stands for the $k$-th entity type label, $N^{(t)}$ denotes number of distinct entity types in KB, and $|\mathbb{E}| = 2 \times N^{(t)} + 1$.

[2]Superscript in bracket denotes the type instead of index.
[3]Toward the first one if entity consists of multiple words.

The prediction for both entity IOB tagging and entity type is formulated as

$$\boldsymbol{p_i^{(ed)}}=\text{softmax}(\text{FFN}(\boldsymbol{h_i};\theta^{(ed)})),\forall i\in[1,n-1] \quad (6)$$

where $\boldsymbol{h_i}$ is the contextual embedding of the $i$-th token in the question, and $\boldsymbol{p_i^{(ed)}} \in \mathbb{R}^{|\mathbb{E}|}$ is the predicted distribution over $\mathbb{E}$.

Given the predicted IOB labels and entity types, we take the following steps for entity linking. First, the predicted IOB labels are used to locate all entities in the question and return corresponding entity mentions. Second, an inverted index built on the KB is leveraged to find entity candidates in KB based on each entity mention. Third, the jointly predicted entity types are used to filter out the candidates with unwanted types, and the remaining entity with the highest inverted index score is selected to substitute the pointer. This process is shown as the bottom part of Figure 2.

During inference phase, the final logical form is derived by replacing entity pointers in entity-pointed logical form from §3.2.2 with entity linking results, and is then executed on the KB to retrieve an answer for the question, as shown as the top part of Figure 2.

### 3.3 Learning and Inference

**Model Learning** During the training phase, we first search gold logical forms for questions in training data over KB if only weak supervision is provided. Then we conduct multi-task learning for semantic parsing and entity detection. The final loss is defined as

$$L = \alpha L^{(sp)} + L^{(ed)}, \quad (7)$$

where $\alpha > 0$ is a hyperparameter for a tradeoff between semantic parsing and entity detection, and $L^{(sp)}$ and $L^{(ed)}$ are negative log-likelihood losses of semantic parsing and entity detection defined as follows.

$$L^{(sp)} = -\frac{1}{|\mathcal{D}|}\sum_{\mathcal{D}}\frac{1}{m}\sum_{j=1}^{m}\log \boldsymbol{p_j^{(tk)}}_{[tk'=y_j^{(tk)}]} \quad (8)$$
$$+\sum_{c\in\{p,t,e,n\}} I_{(y_j^{(tk)}=c)} \log \boldsymbol{p_j^{(c)}}_{[c'=y_j^{(c)}]}$$

$$L^{(ed)} = -\frac{1}{|\mathcal{D}|}\sum_{\mathcal{D}}\frac{1}{n-1}\sum_{i=1}^{n-1}\log \boldsymbol{p_i^{(ed)}}_{[ed'=y_i^{(ed)}]} \quad (9)$$

In the two equations above, $y_j^{(tk)}$ is gold label for decoding token in $\mathbb{V}^{(dec)}$; $y_j^{(p)}, y_j^{(t)}, y_j^{(e)}$ and $y_j^{(n)}$ are gold labels for predicate, type, entity position and number position for instantiation; $\boldsymbol{p_j^{(tk)}}$, $[\boldsymbol{p_j^{(c)}}]_{c\in\{p,t,e,n\}}$, and $\boldsymbol{p_j^{(ed)}}$ are defined in Eq.(1-6) respectively; and $m$ denotes the decoding length.

Here, we use a single model to handle two subtasks simultaneously, i.e., semantic parsing and entity detection. This multi-task learning framework enables each subtask to leverage supervision signals from the others, and thus improves the final performance for KB-QA.

**Grammar-Guided Inference** The grammars defined in Table 1 are utilized to filter illegal operators out in each decoding step. An operator is legitimate if its left-hand semantic category in the definition is identical to the leftmost nonterminal (i.e., un-instantiated semantic category) in the incomplete logical form parsed so far. In particular, the decoding of a logical form begins with the semantic category $start$. During decoding, the proposed semantic parsing model recursively rewrites the leftmost nonterminal in the logical form by 1) applying a legitimate operator for an intermediate semantic category, or 2) instantiation for one of entity, predicate, type or number for an entry semantic category. The decoding process for the parsing terminates until no nonterminals remain.

Furthermore, beam search is also incorporated to boost the performance of the proposed model during the decoding. And, the early stage execution is performed to filter out illegal logical forms that lead to empty intermediate result.

## 4 Experiments

### 4.1 Experimental Settings

**Dataset** We evaluated the proposed approach on Complex Sequential Question Answering (CSQA) dataset[4] (Saha et al., 2018), which is the largest dataset for conversational question answering over large-scale KB. It consists of about 1.6M question-answer pairs in ∼200K dialogs, where 152K/16K/28K dialogs are used for train/dev/test. Questions are classified as different types, e.g., simple, comparative reasoning, logical reasoning questions. Its KB is built on Wikidata[5] in a form of (subject, predicate, object), and consists of 21.2M triplets over 12.8M entities, 3,054 distinct entity types, and 567 distinct predicates.

[4] https://amritasaha1812.github.io/CSQA
[5] https://www.wikidata.org

| **Methods** | | HRED+KVmem | D2A (Baseline) | MaSP (Ours) | Δ |
|---|---|---|---|---|---|
| **Question Type** | **#Example** | F1 Score | F1 Score | F1 Score | |
| Overall | 203k | 9.39% | 66.70% | 79.26% | +12.56% |
| Clarification | 9k | 16.35% | 35.53% | 80.79% | +45.26% |
| Comparative Reasoning (All) | 15k | 2.96% | 48.85% | 68.90% | +20.05% |
| Logical Reasoning (All) | 22k | 8.33% | 67.31% | 69.04% | +1.73% |
| Quantitative Reasoning (All) | 9k | 0.96% | 56.41% | 73.75% | +17.34% |
| Simple Question (Coreferenced) | 55k | 7.26% | 57.69% | 76.47% | +18.78% |
| Simple Question (Direct) | 82k | 13.64% | 78.42% | 85.18% | +6.76% |
| Simple Question (Ellipsis) | 10k | 9.95% | 81.14% | 83.73% | +2.59% |
| **Question Type** | **#Example** | Accuracy | Accuracy | Accuracy | |
| Verification (Boolean) | 27k | 21.04% | 45.05% | 60.63% | +15.58% |
| Quantitative Reasoning (Count) | 24k | 12.13% | 40.94% | 43.39% | +2.45% |
| Comparative Reasoning (Count) | 15k | 8.67% | 17.78% | 22.26% | +4.48% |

Table 2: Comparisons with baselines on CSQA. The last column consists of differences between MaSP and D2A.

**Training Setups** We leveraged a BFS method to search valid logical forms for questions in training data. The buffer size in BFS is set to 1000. Both embedding and hidden sizes in the model are set to $300D$, and no pretrained embeddings are loaded for initialization, and the positional encodings are randomly initialized and learnable. The head number of multi-head attention is 6 and activation function inside $\mathrm{FFN}(\cdot)$ is $\mathrm{Gelu}(\cdot)$ (Hendrycks and Gimpel, 2016). We used Adam (Kingma and Ba, 2015) to optimize the loss function defined in Eq.(7) where $\alpha$ is set to 1.5, and learning rate is set to $10^{-4}$. The training batch size is 128 for 6 epochs. And we also employed learning rate warmup within the first $1\%$ steps and linear decay within the rest. The source codes are available at `https://github.com/taoshen58/MaSP`.

**Evaluation Metrics** We used the same evaluation metrics as Saha et al. (2018) and Guo et al. (2018). F1 score (i.e., precision and recall) is used to evaluate the question whose answer is comprised of entities, and accuracy is used to measure the question whose answer type is boolean or number.

**Baselines** There are few works targeting conversational question answering over a large-scale knowledge base. HRED+KVmem (Saha et al., 2018) and D2A (Guo et al., 2018) are two typical approaches, and we compared them with our proposed approach. Particularly, HRED+KVmem is a memory network (Sukhbaatar et al., 2015; Li et al., 2017) based seq2seq model, which combines HRED model (Serban et al., 2016) with key-value memory network (Miller et al., 2016). D2A[6] is a memory augmented neural symbolic model for semantic parsing in KB-QA, which introduces dialog memory manager to handle ellipsis and coreference problems in conversations.

### 4.2 Model Comparisons

We compared our approach (denoted as MaSP) with HRED+KVmem and D2A in Table 2. As shown in the table, the semantic parsing based D2A significantly outperforms the memory network based text generation approach (HRED+KVmem), which thus poses a strong baseline. Further, our proposed approach (MaSP) achieves a new state-of-the-art performance, where the overall F1 score is improved by $\sim$12%. Besides, the improvement is consistent for all question types, which ranges from 2% to 25%.

There are two possible reasons for this significant improvement. First, our approach predicts entities more accurately, where the accuracy of entities in final logical forms increases from 55% to 72% compared with D2A. Second, the proposed pointer-equipped logical form decoder in the multi-task learning framework handles coreference better. For instance, given an user question with history, "*What is the parent organization of that one? // Did you mean Polydor Records ? // No, I meant Deram Records. Could you tell me the answer for that?*" with coreference, D2A produces "(*find* {*Polydor Records*}, *owned by*)" and in contrast our approach produces "(*find* {*Deram Records*}, *owned by*)". This also explains the substantial improvement for *Simple Question (Coreferenced)* and *Clarification*[7].

[6]Overall score of D2A reported in this paper is superior to that in the original paper since our re-implemented grammars for CSQA achieve a better balance between the simple and non-simple question types. For rational and fair comparisons, we report re-run results for D2A in this paper.

[7]In CSQA, the performance of *Clarification* closely depends on F1 score for next question, 88% of which belong to

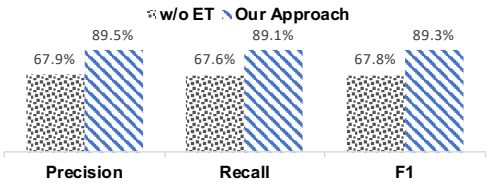


Figure 3: Performance of entity linking. "w/o ET" denotes removing entity type filtering.

We also observed that the improvement of MaSP over D2A for some question types is relatively small, e.g., 1.73% for logical reasoning questions. A possible reason is that there are usually more than one entities are needed to compose the correct logical form for logical reasoning questions, and our current model is too shallow to parse the multiple entities. Hence, we adopted deeper model and employed BERT (Devlin et al., 2018) as the encoder (latter in §4.4), and found that the performance of logical reasoning questions is improved by 10% compared to D2A.

### 4.3 Ablation Study

| **Methods** | Ours | w/o ET | w/o Multi | w/o Both |
|---|---|---|---|---|
| **Question Type** | F1 | F1 | F1 | F1 |
| Overall | 79.26% | 70.42% | 76.73% | 68.22% |
| Clarification | 80.79% | 68.01% | 66.30% | 54.64% |
| Comparative | 68.90% | 66.35% | 61.12% | 58.04% |
| Logical | 69.04% | 62.63% | 67.81% | 62.51% |
| Quantitative | 73.75% | 73.75% | 64.56% | 64.55% |
| Simple (Co-ref) | 76.47% | 64.94% | 74.35% | 63.15% |
| Simple (Direct) | 85.18% | 75.24% | 84.93% | 75.19% |
| Simple (Ellipsis) | 83.73% | 78.45% | 82.66% | 77.44% |
| **Question Type** | Accu | Accu | Accu | Accu |
| Verification | 60.63% | 45.40% | 60.43% | 45.02% |
| Quantitative | 43.39% | 39.70% | 37.84% | 43.39% |
| Comparative | 22.26% | 19.08% | 18.24% | 22.26% |

Table 3: Ablation study. "w/o ET" stands for removing entity type prediction in Entity Detection of §3.2.3; "w/o Multi" stands for learning two subtasks separately in our framework; and "w/o Both" stands for a combination of "w/o ET" and "w/o Multi".

There are two aspects leading to performance improvement, i.e., predicting entity type in entity detection to filter candidates, and multi-task learning framework. We conducted an ablation study in Table 3 for in-depth understanding of their effects.

**Effect of Entity Type Prediction (w/o ET)** First, the entity type prediction was removed from the entity detection task, which results in 9% drop of overall F1 score. We argue that the performance of the KB-QA task is in line with that of entity linking. Hence, we separately evaluated the entity linking task on the test set. As illustrated in Figure 3, both precision and recall of entity linking drop significantly without filtering the entity linking results w.r.t. the predicted entity type, which verifies our hypothesis above.

*Simple Question(Coreference)* .

| Accuracy | Ours | w/o Multi |
|---|---|---|
| Entity pointer | 79.8% | 79.3% |
| Predicate | 96.9% | 96.3% |
| Type | 86.8% | 84.1% |
| Number | 89.1% | 88.3% |
| Operators | 79.4% | 78.7% |

Table 4: Prediction accuracy on each component composing the pointer-equipped logical form.

**Effect of Multi-Task Learning (w/o Multi)** Second, to measure the effect of multi-task learning, we evaluated the KB-QA task when the two subtasks, i.e., pointer-equipped semantic parsing and entity detection, are learned separately. As shown in Table 3, the F1 score for every question type consistently drops in the range of 3% to 14% compared with that with multi-task learning. We further evaluated the effect of multi-task learning on each subtask. As shown in Table 4, the accuracy for each component of the pointer-equipped logical form drops with separate learning. Meanwhile, we found 0.1% F1 score reduction (99.4% vs. 99.5%) for entity detection subtask compared to the model without multi-task learning, which only poses a negligible effect on the downstream task. To sum up, the multi-task learning framework increases the accuracy of the pointer-based logical form generation while keeping a satisfactory performance of entity detection, and consequently improves the final question answering performance.

Note that, considering a combination of removing the entity type filter and learning two subtasks separately (i.e., w/o Both in Table 3), the proposed framework will degenerate to a model that is similar to Coarse-to-Fine semantic parsing model, another state-of-the-art KB-QA model over small-scale KB (Dong and Lapata, 2018). Therefore, an improvement of 11% of F1 score also verifies the advantage of our proposed framework.

### 4.4 Model Setting Analysis

As introduced in §4.1 and evaluated in §4.2, the proposed framework is built on a relatively shal-

| **Methods** | Vanilla | w/ BERT | w/ Large Beam |
|---|---|---|---|
| **Question Type** | F1 | F1 | F1 |
| Overall | 79.26% | 80.60% | 81.55% |
| Clarification | 80.79% | 79.46% | 83.37% |
| Comparative | 68.90% | 65.99% | 69.34% |
| Logical | 69.04% | 77.53% | 69.41% |
| Quantitative | 73.75% | 70.43% | 73.75% |
| Simple (Co-ref) | 76.47% | 77.95% | 79.03% |
| Simple (Direct) | 85.18% | 86.40% | 88.28% |
| Simple (Ellipsis) | 83.73% | 84.82% | 86.96% |
| **Question Type** | Accuracy | Accuracy | Accuracy |
| Verification | 60.63% | 63.85% | 61.96% |
| Quantitative | 43.39% | 47.14% | 44.22% |
| Comparative | 22.26% | 25.28% | 22.70% |

Table 5: Comparisons with different experimental settings. "Vanilla" stands for standard settings of our framework, i.e, MaSP. "w/ BERT" stands for incorporating BERT. And "w/ Large Beam" stands for increasing beam search size from 4 to 8.

low neural network, i.e., stacked two-layer multi-head attention, which might limit its representative ability. Hence, in this section, we further exploited the performance of the proposed framework by applying more sophisticated strategies.

As shown in Table 5, we first replaced the encoder with pre-trained BERT base model (Devlin et al., 2018) and fine-tuned parameters during the training phase, which results in 1.3% F1 score improvement over the vanilla one. Second, we increased beam search size from 4 to 8 during the decoding in the inference phase for the standard settings, which leads to 2.3% F1 score increase.

### 4.5 Error Analysis

We randomly sampled 100 examples with wrong logical forms or incorrect answers to conduct an error analysis, and found that the errors mainly fall into the following categories.

**Entity Ambiguity** Leveraging entity type as a filter in entity linking significantly reduces errors caused by entity ambiguity, but it is still possible that different entities with same text belong to the same type, due to coarse granularity of the entity type, which results in filtering invalidity. For example, it is difficult to distinguish between two persons whose names are both *Bill Woods*.

**Wrong Predicted Logical Form** The predicted components (e.g., operators, predicates and types) composing the logical form would be inaccurate, leading to a wrong answer to the question or an un-executable logical form.

**Spurious Logical Form** We took a BFS method to search gold logical forms for questions in training set, which inevitably generates spurious (incorrect but leading to correct answers coincidentally) logical forms as training signals. Take the question "*Which sexes do King Harold, Queen Lillian and Arthur Pendragon possess*" as an example, a spurious logical form only retrieves the genders of "*King Harold*" and "*Queen Lillian*", while it gets correct answers for the question. Spurious logical forms accidentally introduce noises into training data and thus negatively affect the performance of KB-QA.

## 5 Related Work

Our work is aligned with semantic parsing based approach for KB-QA. Traditional semantic parsing systems typically learn a lexicon-based parser and a scoring model to construct a logical form given a natural language question (Zettlemoyer and Collins, 2007; Wong and Mooney, 2007; Zettlemoyer and Collins, 2009; Kwiatkowski et al., 2011; Andreas et al., 2013; Artzi and Zettlemoyer, 2013; Zhao and Huang, 2014; Long et al., 2016). For example, Zettlemoyer and Collins (2009) and Artzi and Zettlemoyer (2013) learn a CCG parser, and Long et al. (2016) develop a shift-reduce parser to construct logical forms.

Neural semantic parsing approaches have been gaining rising attention in recent years, eschewing the need for extensive feature engineering (Jia and Liang, 2016; Ling et al., 2016; Xiao et al., 2016). Some efforts have been made to utilize the syntax of logical forms (Rabinovich et al., 2017; Krishnamurthy et al., 2017; Cheng et al., 2017; Yin and Neubig, 2017). For example, Dong and Lapata (2016) and Alvarez-Melis and Jaakkola (2017) leverage an attention-based encoder-decoder framework to translate a natural language question to tree-structured logical form.

Recently, to handle huge entity vocabulary existing in a large-scale knowledge base, many works take a stepwise approach. For example, Liang et al. (2016), Dong and Lapata (2016), and Guo et al. (2018) first process questions using a name entity linking system to find entity candidates, and then learn a model to map a question to a logical form based on the candidates. Dong and Lapata (2018) decompose the task into two stages: first, a sketch of the logical form is predicted, and then a full logical form is generated with consid-

ering both the question and the predicted sketch.

Our proposed framework also decomposes the task into multiple subtasks but is different from existing works in several aspects. First, inspired by pointer network (Vinyals et al., 2015), we replace entities in a logical form with the starting positions of their mentions in the question, which can be naturally used to handle coreference problem in conversations. Second, the proposed pointer-based semantic parsing model can be intrinsically extended to jointly learn with entity detection for fully leveraging all supervision signals. Third, we alleviate entity ambiguity problem in entity detection & linking subtask, by incorporating entity type prediction into entity mention IOB labeling to filter out the entities with unwanted types.

## 6 Conclusion

We studied the problem of conversational question answering over a *large-scale* knowledge base, and proposed a multi-task learning framework which learns for type-aware entity detection and pointer-equipped logical form generation simultaneously. The multi-task learning framework takes full advantage of the supervisions from all subtasks, and consequently increases the performance of final KB-QA problem. Experimental results on a large-scale dataset verify the effectiveness of the proposed framework. In the future, we will test our proposed framework on more datasets and investigate potential approaches to handle spurious logical forms for weakly-supervised KB-QA.

## Acknowledgments

We acknowledge the support of NVIDIA Corporation and MakeMagic Australia with the donation of GPUs for our research group at University of Technology Sydney. And we also thank anonymous reviewers for their insightful and constructive suggestions.

## References

David Alvarez-Melis and Tommi S Jaakkola. 2017. Tree-structured decoding with doubly-recurrent neural networks. In *ICLR*.

Jacob Andreas, Andreas Vlachos, and Stephen Clark. 2013. Semantic parsing as machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 47–52.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1:49–62.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. AcM.

Jianpeng Cheng, Siva Reddy, Vijay Saraswat, and Mirella Lapata. 2017. Learning structured natural language representations for semantic parsing. *arXiv preprint arXiv:1704.08387*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43, Berlin, Germany. Association for Computational Linguistics.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 731–742. Association for Computational Linguistics.

Daya Guo, Duyu Tang, Nan Duan, Ming Zhou, and Jian Yin. 2018. Dialog-to-action: Conversational question answering over a large-scale knowledge base. In *Advances in Neural Information Processing Systems*, pages 2946–2955.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

R. Jia and P. Liang. 2016. Data recombination for neural semantic parsing. In *Association for Computational Linguistics (ACL)*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1516–1526.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical generalization in ccg grammar induction for semantic parsing. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1512–1523. Association for Computational Linguistics.

Zheng Li, Yun Zhang, Ying Wei, Yuxiang Wu, and Qiang Yang. 2017. End-to-end adversarial memory network for cross-domain sentiment classification. In *IJCAI*, pages 2237–2243.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D Forbus, and Ni Lao. 2016. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. *arXiv preprint arXiv:1611.00020*.

Wang Ling, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, Andrew Senior, Fumin Wang, and Phil Blunsom. 2016. Latent predictor networks for code generation. *arXiv preprint arXiv:1603.06744*.

Reginald Long, Panupong Pasupat, and Percy Liang. 2016. Simpler context-dependent logical forms via model projections. *arXiv preprint arXiv:1606.05378*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126*.

Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract syntax networks for code generation and semantic parsing. *arXiv preprint arXiv:1704.07535*.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics*, 2:377–392.

Amrita Saha, Vardaan Pahuja, Mitesh M Khapra, Karthik Sankaranarayanan, and Sarath Chandar. 2018. Complex sequential question answering: Towards learning to converse over linked question answer pairs with a knowledge graph. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *NIPS*.

Ashish Vaswani, Shazeer, Noam, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Yuk Wah Wong and Raymond Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 960–967.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Chunyang Xiao, Marc Dymetman, and Claire Gardent. 2016. Sequence-based structured prediction for semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1341–1350.

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Question answering on freebase via relation extraction and textual evidence. *arXiv preprint arXiv:1603.00957*.

Scott Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *ACL-IJCNLP*.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. *arXiv preprint arXiv:1704.01696*.

Luke Zettlemoyer and Michael Collins. 2007. Online learning of relaxed ccg grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Luke S Zettlemoyer and Michael Collins. 2009. Learning context-dependent mappings from sentences to logical form. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 976–984. Association for Computational Linguistics.

Kai Zhao and Liang Huang. 2014. Type-driven incremental semantic parsing with polymorphism. *arXiv preprint arXiv:1411.5379*.

# BiPaR: A Bilingual Parallel Dataset for Multilingual and Cross-lingual Reading Comprehension on Novels

**Yimin Jing , Deyi Xiong*** and **Yan Zhen**
School of Computer Science and Technology, Soochow University, Suzhou, China
{yymmjing,corezhen}@gmail.com
dyxiong@suda.edu.cn

## Abstract

This paper presents *BiPaR*, a **bi**lingual **pa**rallel novel-style machine **r**eading comprehension (MRC) dataset, developed to support multilingual and cross-lingual reading comprehension. The biggest difference between BiPaR and existing reading comprehension datasets is that each triple (Passage, Question, Answer) in BiPaR is written parallelly in two languages. We collect 3,667 bilingual parallel paragraphs from Chinese and English novels, from which we construct 14,668 parallel question-answer pairs via crowdsourced workers following a strict quality control procedure. We analyze BiPaR in depth and find that BiPaR offers good diversification in prefixes of questions, answer types and relationships between questions and passages. We also observe that answering questions of novels requires reading comprehension skills of coreference resolution, multi-sentence reasoning, and understanding of implicit causality, etc. With BiPaR, we build monolingual, multilingual, and cross-lingual MRC baseline models. Even for the relatively simple monolingual MRC on this dataset, experiments show that a strong BERT baseline is over 30 points behind human in terms of both EM and F1 score, indicating that BiPaR provides a challenging testbed for monolingual, multilingual and cross-lingual MRC on novels. The dataset is available at https://multinlp.github.io/BiPaR/.

## 1 Introduction

Machine reading comprehension is to evaluate how well computer systems understand natural language texts, where machines read a given text passage and answer questions about the passage. It has been regarded as a crucial technology for many applications such as question answering, dialogue systems (Nguyen et al., 2016; Chen et al., 2017; Liu et al., 2018; Wang et al., 2018) and so on. In order to enable machine to understand texts, large-scale reading comprehension datasets have been developed, such as CNN/Daily Mail (Hermann et al., 2015), SQuAD (Rajpurkar et al., 2016), MS MACRO (Nguyen et al., 2016), hotpotQA (Yang et al., 2018), CoQA (Reddy et al., 2019), etc.




Figure 1: Illustration of BiPaR with the monolingual, multilingual and cross-lingual MRC on the dataset.

The majority of such datasets, unfortunately, are only for monolingual text understanding. To the best of our knowledge, there is no publicly available bilingual parallel reading comprehension dataset, which is exactly what BiPaR is mainly developed for, as illustrated in Figure 1. BiPaR

*Corresponding author



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2452–2462, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| Dataset | Language | Parallel | Answer Type | Domain |
|---|---|---|---|---|
| CNN/DM (Hermann et al., 2015) | EN | × | Fill in entity | News |
| HLF-RC (Cui et al., 2016) | ZH | × | Fill in word | Fairy/News |
| SQuAD (Rajpurkar et al., 2016) | EN | × | Span of words | Wikipedia |
| CMRC2018 (Cui et al., 2018) | ZH | × | Span of words | Wikipedia |
| MS MACRO (Nguyen et al., 2016) | EN | × | Manual summary | Web doc. |
| DuReader (He et al., 2018) | ZH | × | Manual summary | Web doc./CQA |
| BiPaR (this paper) | EN-ZH | √ | Span of words | Novels (Kongfu novels, science fictions, etc.) |

Table 1: Comparison of BiPaR with several existing reading comprehension datasets.

produces over 14K Chinese-English parallel questions from nearly 4K bilingual parallel novel passages with consecutive word spans from these passages as answers, following SQuAD (Rajpurkar et al., 2016). Table 1 shows that BiPaR has two significant differences in comparison with existing datasets: (1) each (Passage, Question, Answer) triple is bilingual parallel, and (2) passages and questions are from novels. BiPaR's bilinguality and novel-based questions provide an interesting corner and unexplored territory for MRC.

With an in-depth analysis on the manually created questions, we observe that answering these novel-style questions requires challenging skills: coreference resolution, multi-sentence reasoning, and understanding of implicit causality, etc. Further monolingual MRC experiments on BiPaR demonstrate that, the English BERT_large model (Devlin et al., 2019) achieves an F1 score of 56.5%, which is 35.4 points behind human performance (91.9%), and the Chinese BERT_Base model achieves an F1 score of 64.1%, which is 28.0 points behind human performance (92.1%), indicating there is a huge gap to be bridged for MRC on novels.

More interestingly, the bilinguality of BiPaR supports multilingual and cross-lingual MRC tasks on this dataset in addition to the traditional monolingual MRC. It is more cost-effective to build a single model that can handle machine reading comprehension on multiple languages, than building multiple MRC systems, one reading system for each language. Different from previous multilingual QA systems that are trained on independently developed datasets of different languages and domains, we are able to train a single multilingual MRC model to do MRC on different languages, by exploring BiPaR that is built parallelly on two different languages with alignments between triples of (Passage, Question, Answer) of the two languages, as shown in Figure 1.

Yet another interesting task that we can do with BiPaR is cross-lingual reading comprehension. We define two types of cross-lingual MRC on BiPaR: (1) using questions in one language to find answers from passages written in another language and (2) finding answers from passages of two different languages for questions in one language. The former is in essence similar to the early cross-lingual question answering (CLQA) (Aceves-Pérez et al., 2008; Peñas et al., 2009; Pérez et al., 2009). The intuitive approaches to CLQA are to translate the questions into the document's language (Sutcliffe et al., 2005; de Pablo-Sánchez et al., 2005; Aceves-Pérez et al., 2007), which, however, suffers from translation errors. The BiPaR dataset provides a potential opportunity for building cross-lingual MRC that does not rely on machine translation.

To summarize, our contributions are threefold:

- We build the BiPaR, the first publicly available bilingual parallel dataset for MRC. The passages are novel paragraphs, originally written in Chinese or English and then translated into the other language. The questions are manually constructed undergoing a strict quality control procedure.
- We conduct an in-depth analysis on BiPaR, which reveals that MRC on novels is very challenging, requiring skills of coreference resolution, inter-sentential reasoning, implicit causality understanding, etc.
- We build monolingual, multilingual and cross-lingual MRC baseline models on BiPaR and provides baseline results as well as human performance on this dataset.

## 2 Related Work

**MRC Datasets and Models** Large-scale cloze-style datasets, such as CNN/Daily Mail (Hermann

et al., 2015), have been automatically developed in the early days of MRC. Several neural network models have been proposed and tested on these datasets, such as ASReader (Kadlec et al., 2016), StanfordAttentiveReader (Chen et al., 2016), AoAReader (Cui et al., 2017), etc. However, Chen et al. (2016) argue that such datasets may be noisy due to the automatic data creation method and co-reference errors. Rajpurkar et al. (2016) propose SQuAD, a dataset created from English Wikipedia, where questions are manually generated by crowdsourced workers, and answers are spans in the Wikipedia passages. Along with the development of this dataset, a variety of neural MRC models have been proposed, such as BiDAF (Seo et al., 2016), R-NET (Wang et al., 2017), ReasonNet (Shen et al., 2017), DCN (Xiong et al., 2016), QANet (Yu et al., 2018), SAN (Liu et al., 2018), etc. Recent years have witnessed substantial progress made on this dataset. However, there are some limitations on SQuAD, which lie in that questions are created based on single passages, that answers are limited to a single span in passages, and that most questions can be answered from a single supporting sentence without requiring multi-sentence reasoning (Chen, 2018). To address these limitations, a number of datasets have be built recently, such as MS MARCO (Nguyen et al., 2016), DuReader (He et al., 2018), TriviaQA (Joshi et al., 2017), RACE (Lai et al., 2017), NarrativeQA (Kocisky et al., 2018), SQuAD2.0 (Rajpurkar et al., 2018), hotpotQA (Yang et al., 2018), CoQA (Reddy et al., 2019), etc. These datasets and models are only for monolingual text understanding. By contrast, BiPaR, following these efforts of creating challenging MRC datasets, aims at setting up a new benchmark dataset for MRC on novels and bilingual/cross-lingual MRC.

**Multilingual MRC and Datasets** Previous studies on multilingual MRC are very limited. Asai et al. (2018) propose a multilingual MRC system by translating the target language into a pivot language via runtime machine translation. They still rely on SQuAD to train the MRC model of the pivot language. No multilingual MRC dataset is created except that a Japanese and French test set is created by manually translating the test set of SQuAD into the two languages.

**Multilingual/Cross-lingual QA** Answering questions in multiple languages or retrieving answers from passages that are written in a language different from questions is an important capability for QA systems. To achieve this goal, QA@CLEF[1] has organized a series of public evaluations for multilingual/cross-lingual QA (Magnini et al., 2006). Widely-used approaches to multilingual/cross-lingual QA are to build monolingual QA systems and then adapt them to multilingual/cross-lingual settings via machine translation (Lin et al., 2005). Such QA systems are prone to being affected by machine translation errors. Hence, various techniques have been proposed to reduce the errors of the machine translation module (Sutcliffe et al., 2005; de Pablo-Sánchez et al., 2005; Aceves-Pérez et al., 2007).

## 3 Dataset Creation

In this section, we elaborate on the three stages of our dataset creation process: collecting bilingual parallel passages, crowdsourcing question-answer pairs on those passages, and constructing multiple answers for the development and test set.

### 3.1 Bilingual Parallel Passage Collection

We select bilingual parallel passages from six Chinese and English novels with different topics, including Chinese martial arts, science fictions, fantasy literature, etc. These novels are either written in Chinese and translated into English or vice versa. Automatic paragraph alignments between Chinese and English are available for these novels. The number of words in each Chinese passage in parallel passages is limited in the range of $[120, 600]$, to make the passage not too short or too long for crowdsourced workers to construct questions. As some of bilingual parallel passages are very difficult to understand and create parallel questions, we need to consider the appropriateness of these passages. The following rules are used to select bilingual parallel passages:

- The Chinese passage shall not contain poetry, couplets, or classical Chinese words/phrases.
- The passage shall not contain too much dialogue, where speakers are difficult to recognize without global context.
- The Chinese passage shall not contain full-passage description of Kongfu fighting. Such

[1] http://www.clef-initiative.eu/track/qaclef

descriptions on fighting are hard to translate in a direct way into a target language (i.e., English in this dataset).

- In order to ensure the correct alignments of Chinese and English passages, if an English passage has over 10 words fewer than its Chinese counterpart, such automatically aligned bilingual passages shall not be selected as they are normally not translations of each other.

Following these selection rules, we finally collect 3,667 bilingual parallel passages. Table 2 provides the number of selected passages from each novel.

### 3.2 Question-Answer Pair Crowdsourcing

We then ask our bilingual crowdsourced workers to create questions and answers on these collected passages. We develop a crowdsourcing annotation system and 150 bilingual workers, 3 bilingual reviewers and 1 expert participate in the data annotation process. The collected bilingual parallel passages are divided into 150 groups and randomly assigned to the bilingual workers. They will create bilingual parallel questions and find corresponding answers to the questions after reading the bilingual parallel passages. Particularly, we encourage workers to create questions according to the following rules:

- For each parallel passage, at least three bilingual question-answer pairs are to be created.
- If the answers in Chinese and English are not parallel (i.e., not translations of each other), the corresponding questions shall be deleted and new questions shall be created.
- Answers have to be consecutive spans in passages.
- If possible, questions of how and why are preferred.
- It is not suggested to directly copy words from passages for creating questions.

In order to guarantee the quality of created question-answer pairs, we use a strict quality control procedure during data annotation. In particular, 30% annotated data from each group are randomly sampled and passed to the three reviewers who will review all answers created by the workers and correct answers if they consider they are wrong. Then, 5% of the reviewed data will be further sampled from each reviewer. The sampled data will be checked again by the expert. If the accuracy is lower than 95%, the corresponding workers and reviewers need to revise the answers again. This quality control loop is executed three times.

| Novels | #Parallel Passages | #Avg EN/ZH tokens |
|---|---|---|
| I | 1,948 | 251.3/195.9 |
| II | 75 | 223.3/194.6 |
| III | 490 | 206.4/190.1 |
| IV | 245 | 175.1/188.9 |
| V | 87 | 190.7/189.8 |
| VI | 822 | 202.4/212.3 |
| Total | 3,667 | 227.3/198.2 |

Table 2: Statistics on the selected passages ( I: *The Duke of the Mount Deer* /《鹿鼎记》, II: *Demi-Gods and Semi-Devils* /《天龙八部》, III: *The Three-Body Problem* /《三体》, IV: *The Great Gatsby* /《了不起的盖茨比》, V: *The Old Man and the Sea* /《老人与海》, VI: *Harry Potter* /《哈利波特》).

At last, we collect 14,668 question-answer pairs along with their corresponding passages. We randomly partition the annotated data into a training set (with 11,668 QA pairs), a development set (1,500 QA pairs), and a test set (1,500 QA pairs).

### 3.3 Multiple Answers Construction

In order to make evaluation more robust, we ask crowdsourced workers to create at least two additional answers for each question in the development and test sets, similar to SQuAD (Rajpurkar et al., 2016). But differently, we make the answers from the crowdsourced workers visible to each other, and encourage them to annotate different but reasonable answers. The reason for creating multiple amswers is that we often encounter situations where multiple answers are correct. Consider the following example from SQuAD[2]:

*P: Official corporal punishment, often by caning, remains commonplace in schools in some Asian, African and Caribbean countries. For details of individual countries see School corporal punishment.*

*Q:What countries is corporal punishment still a normal practice?*

The ground truth answers of the question are *some Asian, African and Caribbean countries /*

[2]https://rajpurkar.github.io/SQuAD-explorer/explore/1.1/dev/

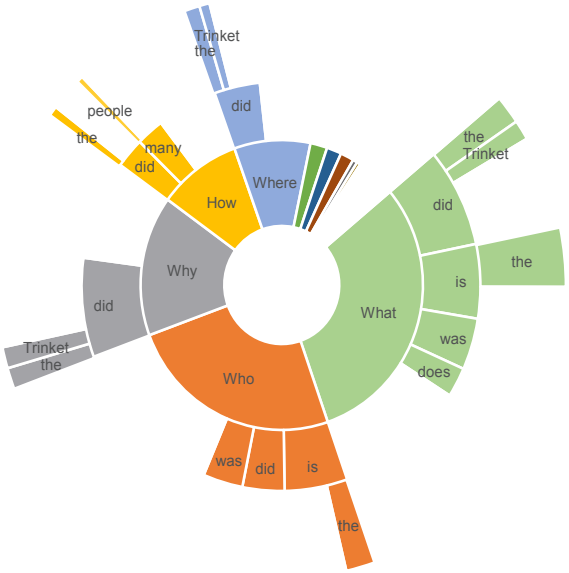


Figure 2: Visualization of the distribution of trigram prefixes of questions in BiPaR.

*Asian, African and Caribbean*. The prediction of BERT (Devlin et al., 2019) ensemble model is *Asian, African and Caribbean countries*. In fact, the machine-predicted result is also correct. However, it will be considered as a wrong answer if the exact match metric is used since the answer is not in the ground truth answer list. Such cases can be avoided if multiple reasonable answers are annotated.

## 4 Dataset Analysis

In this section, we analyze the the types of questions and answers, the relationships of questions with passages as well as reading skills covered in BiPaR. Due to the bilingual parallelism of BiPaR, we choose the English part of the dataset for analyses.

### 4.1 Prefixes of Questions

Figure 2 shows the distribution of trigram prefixes of questions in BiPaR. Unlike SQuAD (Rajpurkar et al., 2018) where nearly half of questions are dominated by *what* questions, the question type distribution in BiPaR is of better dispersion over multiple question types. In particular, BiPaR has 15.6% *why* and 9.5% *how* questions. Since in the novel texts, causality is usually not represented by explicit expressions such as "why", "because", and "the reason for", answering these questions in BiPaR requires the MRC models to understand implicit causality (Section 4.3). The *why* and *how* questions, which account for considerable proportions, undoubtedly make BiPaR a very challenging MRC dataset.

| Answer Type | % | Examples |
|---|---|---|
| Person | 23 | Trinket |
| Location | 13 | the floor |
| Date | 6 | Saturday morning |
| Verb phrase | 2 | wash her face |
| Yes/No | 2 | - |
| Adjective | 3 | jade-green |
| Event | 2 | Quidditch practice |
| Other proper noun | 4 | a pinch of Floo powder |
| Common noun | 8 | secret vault |
| Description | 37 | wand backfired |

Table 3: Statistics on answer types in BiPaR.

### 4.2 Answers Types

We sample 100 examples from the development set, and present the types of answers in Table 3. As is shown, BiPaR covers a broad range of answer types, which matches our analysis on questions contribution. Moreover, we find that a large number of questions require some descriptive sentences to answer (37%), which are generally complete sentences or summary statements, etc (See the third example in Table 4). These answers are usually corresponding to *what/why/how* questions, for instances:

*1)* ***What*** *is Ding Yi doing?*
*2)* ***Why*** *is fighting so much less fun?*
*3)* ***How*** *did the protagonist treat her?*

### 4.3 Relationships of Questions with Passages and Reading Comprehension Skills

In order to assess how difficult to answer questions in BiPaR, we further analyze the relationships between questions and corresponding passages as well as reading comprehension skills required to detect answers for BiPaR questions. We sample 100 samples from the development set and annotate them with various reasoning phenomena as shown in Table 4.

Inspired by Reddy et al. (2019), we group questions into several categories in terms of their relationships with passages. If a question contains more than one content word that appears in the passage, we label it as *lexical match*. These account for 49.0% of all the questions. One might think that if a lot of words in a question overlap with those in a passage, the answer may be easily detected from the matched sentence in the passage. However, this is not the case at all in BiPaR. As the first example in Table 4 shows, the question is almost the same to the sentence in passage. However, the answer is far away from the matched

| Phenomenon | Example | % |
|---|---|---|
| | Relationship between a question and its passage | |
| Lexical match | $\mathbf{P}_{en}$: ... **Doublet** flew into the attack, flailing around her like the wind. She was too small to reach the bodies of her enemies, ...jabbing the Vital Points on the riders' legs.<br>$\mathbf{P}_{zh}$: ... **双儿**出手如风，只是敌人骑在马上，她身子又矮，打不到敌人，... 便戳中敌人腿上的穴道。<br>$\mathbf{Q}_{en}$-$\mathbf{Q}_{zh}$: Who did jab the Vital Points on the riders' legs? / 谁戳中敌人腿上的穴道? | 49 |
| Paraphrasing | $\mathbf{P}_{en}$: ... Getting very difficult ter find anyone fer **the Dark Arts job**.<br>$\mathbf{P}_{zh}$: ... 现在找一个**黑魔法防御术课老师**很困难，人们都不大想干，觉得这工作不吉利。<br>$\mathbf{Q}_{en}$-$\mathbf{Q}_{zh}$: What occupation do people avoid? / 人们不愿从事什么职业? | 27 |
| Summary | $\mathbf{P}_{en}$: ... As soon as he opened the door to Ding Yi's brand-new three-bedroom apartment, ... **The apartment was unfinished, with only a few pieces of furniture and little decoration, and the huge living room seemed very empty. The most eye-catching object was the pool table in the corner**.<br>$\mathbf{P}_{zh}$: ... 推开丁仪那套崭新的三居室的房门，... 看到**房间还没怎么装修，也没什么家具和陈设，宽大的客厅显得很空，最显眼的是客厅一角摆放的一张台球桌**。<br>$\mathbf{Q}_{en}$-$\mathbf{Q}_{zh}$: What does Ding Yi's three-bedroom look like now? / 丁仪的三居室现在是什么样? | 24 |
| | Reading comprehension skills required to answer questions | |
| Coreference resolution | $\mathbf{P}_{en}$: Trying hard to bear all this in mind, Harry took a ... he opened his mouth and immediately swallowed a lot of hot ash."D-**Dia-gon Alley**," he coughed.<br>$\mathbf{P}_{zh}$: 哈利拼命把这些都记在心里，... 他一张嘴，马上吸了一大口滚烫的烟灰。"对一**对角巷**。"他咳着说。<br>$\mathbf{Q}_{en}$-$\mathbf{Q}_{zh}$: Where did Harry go? / 哈利去了哪儿? | 31 |
| Multi-sentence reasoning | $\mathbf{P}_{en}$: and heard a woman's voice cry out from within it: 'Stop! Lay down your arms! We should all be friends here!' ... The cart stopped in front of them, and out jumped—**Fang Yi**.<br>$\mathbf{P}_{zh}$: 车中一个女子声音叫道："是自己人，别动手！" ... 小车驶别跟前，车中跃出一人，正是**方怡**。<br>$\mathbf{Q}_{en}$-$\mathbf{Q}_{zh}$: Who did stop the conflict? / 是谁制止了冲突? | 32 |
| Implicit causality | $\mathbf{P}_{en}$: Harry, however, was shaken awake several hours earlier than he would have liked by Oliver Wood, Captain of the Gryffindor Quidditch team."Whassamatter?" said Harry groggily. "**Quidditch practice**" said Wood.<br>$\mathbf{P}_{zh}$: 哈利一早就被格兰芬多魁地奇队队长奥利弗伍德摇醒了，他本来还想再睡几个小时的。"什一什么事？"哈利迷迷糊糊地说。"**魁地奇训练**"伍德说。<br>$\mathbf{Q}_{en}$-$\mathbf{Q}_{zh}$: Why did Oliver Wood shake Harry awake? / 奥利弗伍德为什么要摇醒哈利? | 17 |

Table 4: Question categories and reading comprehension skills covered in BiPaR. Bold words are answers while underlined words indicate coreference resolution.

sentence. Actually, correctly answering this question requires very complicated reading comprehension skills, such as multi-sentence reasoning, ellipsis/co-reference resolution, etc. We've found 43 examples of this case among the 49 lexical-match samples.

If there is no lexical match between a question and the corresponding passage but we can find a semantically matched sentence to the question from the passage, we regard this case as *paraphrasing*. Such questions account for 27.0% of the questions. Interestingly, we find questions in BiPaR that are not found in other datasets. We refer to these questions as *summary* questions, which account for 24% of the sampled questions. In order to answer these questions, MRC models need to read the entire passage to detect summative statements. Examples of the summary questions are:

*1) What is the situation of the Old Majesty?*

*2) What features does Oboi's bedroom show?*

In addition, we also analyze the reading comprehension skills required to answer questions. We find that *coreference resolution*, *multi-sentence reasoning* and *implicit causality understanding* frequently appear in answering questions in BiPaR. What deserves our special attention here is the implicit causality, which rarely appears in other datasets. For some questions, it is crucial to understand causality that is not represented by explicit expressions such as "why", "because", and "the reason for". As demonstrated in the last example in Table 4, to correctly answer the question, we must understand the implicit causality: *Quidditch practice* $\longrightarrow$ *Harry, however, was shaken awake several hours earlier than he would have liked by Oliver Wood, Captain of the Gryffindor Quidditch team.*

## 5 MRC Task Formulation on BiPaR

With aligned passage-question-answer triples, we can define three MRC tasks (monolingual, multilingual and cross-lingual) with seven different forms on this dataset, as demonstrated in Figure 1. Since our goal is to provide benchmark results on this new dataset, we either directly train state-of-the-art MRC models on these tasks or use a straightforward way to adapt existing approaches to the defined tasks in this paper. We leave new approaches, especially those for

multilingual and cross-lingual MRC to our future work.

**Monolingual MRC**: ($P_{en}$, $Q_{en}$, $A_{en}$) or ($P_{zh}$, $Q_{zh}$, $A_{zh}$). With these two monolingual MRC forms, we can investigate the performance variation of the same MRC model trained on two different languages with equivalent training instances. In our experiments, we directly train off-the-shelf MRC models on the two monolingual tasks to evaluate their performance on Chinese and English.

**Multilingual MRC**: ($P_{en}$, $Q_{en}$, $A_{en}$, $P_{zh}$, $Q_{zh}$, $A_{zh}$). Similar to multilingual neural machine translation (Johnson et al., 2017), we can build a single MRC model to handle MRC of multiple languages on BiPaR. In our benchmark test, we directly mix training instances of the two languages into a single training set. Correspondingly, the two vocabularies are also combined into one vocabulary for both languages. After that, we train MRC models on this language-mixed dataset to endow them with the multilingual comprehension capacity.

**Cross-lingual MRC**: The first two forms of cross-lingual MRC are ($P_{en}$, $Q_{zh}$, $A_{en}$) or ($P_{zh}$, $Q_{en}$, $A_{zh}$), in which we use questions in one language to extract answers from passages written in another language. The other two forms are ($P_{en}$, $P_{zh}$, $Q_{zh}$, $A_{zh}$, $A_{en}$) or ($P_{zh}$, $P_{en}$, $Q_{en}$, $A_{en}$, $A_{zh}$), in which we use questions written in one language to extract answers from passages written in multiple languages. For the first two forms of cross-lingual MRC, we use Google Translate[3] to translate questions into the language of passages, and then treat them as a monolingual MRC task. For the second two forms of cross-lingual MRC, such as ($P_{zh}$, $P_{en}$, $Q_{en}$, $A_{en}$, $A_{zh}$), we first obtain $A_{en}$ through a monolingual MRC model, then use the word alignment tool fast_align[4] to obtain the aligned $A_{zh}$ from $P_{zh}$. Alternative approaches that do not rely on machine translation or word alignments for the cross-lingual MRC tasks are to directly build cross-lingual MRC models on language-mixed training instances constructed from BiPaR or to explore multi-task learning on multiple languages (Dong et al., 2015).

[3]https://translate.google.com/

[4]https://github.com/clab/fast_align

## 6 Experiments

We carried out experiments with state-of-the-art MRC models on BiPaR to provide machine results for these 7 MRC tasks defined above. We also provide human performance on the monolingual tasks and demonstrate the performance trajectory of human and machine in answering BiPaR questions.

### 6.1 Evaluation Metric

Like evaluations on other extraction-based datasets, we used EM and F1 to evaluate model accuracy on BiPaR. Particularly, we used the evaluation program of SQuAD1.1 for the English dataset in BiPaR, and the evaluation program[5] of CMRC2018 for the Chinese dataset in BiPaR.

### 6.2 Human Performance Evaluation

In order to assess human performance on BiPaR, we hired three other bilingual crowdsourced workers to independently answer questions (both Chinese and English) on the test set which contains three answers per question as described in Section 3.3. We then calculated the average results of the three human workers as the final human performance on this dataset, which are shown in Table 5.

### 6.3 Baseline models

We adapted the following state-of-the-art models to the dataset and MRC tasks as described in Section 4.

**DrQA**[6]**:** DrQA (Chen et al., 2017) is a simple but effective neural network model for reading comprehension.

**BERT**[7]**:** BERT (Devlin et al., 2019) is a strong method for pre-training language representations, which obtains the state-of-the-art results on many reading comprehension datasets. We used the multilingual model of BERT trained on multiple languages for the evaluation of our multilingual MRC task.

### 6.4 Experimental Setup

All the baselines were tested using their default hyper-parameters except BERT. We only changed the batch size to 8 for BERT_base and 6 for BERT_large due to the memory limit of our

[5]https://github.com/ymcui/cmrc2018

[6]https://github.com/hitvoice/DrQA

[7]https://github.com/huggingface/pytorch-pretrained-BERT

| | Monolingual MRC | | Multilingual MRC | | Cross-lingual MRC | | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | |
| | | | | Development set | | | | |
| DrQA | 29.87/43.47 | 36.60/52.90 | 31.47/44.93 | 36.68/54.03 | 27.80/38.94 | 28.47/43.65 | 8.07/19.80 | 6.27/19.79 |
| BERT_base | 41.67/56.23 | **52.53/67.65** | 42.33/55.49 | 49.00/63.99 | 36.27/49.98 | **41.93/55.66** | 8.00/24.37 | 8.60/22.51 |
| BERT_large | **44.47/58.94** | - | - | | **40.40/53.28** | - | 7.87/24.60 | - |
| | | | | Test set | | | | |
| DrQA | 27.00/39.29 | 37.40/53.11 | 28.00/42.49 | 36.60/53.34 | 21.93/34.45 | 27.53/41.08 | 7.00/18.63 | 4.07/16.64 |
| BERT_base | 41.40/55.03 | **48.87/64.09** | 38.33/51.20 | 49.00/64.06 | 32.80/46.36 | **39.87/53.10** | 5.73/21.08 | 7.67/20.69 |
| BERT_large | **42.53/56.48** | - | - | | **37.53/51.51** | - | 5.60/22.29 | - |
| Human | 80.50/91.93 | 81.50/92.12 | | | | | | |

Table 5: Results (EM/F1 score) of models and humans on the development and the test data of BiPaR. 1-7 indicate the seven different MRC tasks on BiPaR: ($P_{en}$, $Q_{en}$, $A_{en}$), ($P_{zh}$, $Q_{zh}$, $A_{zh}$), ($P_{en}$, $Q_{en}$, $A_{en}$, $P_{zh}$, $Q_{zh}$, $A_{zh}$), ($P_{en}$, $Q_{zh}$, $A_{en}$), ($P_{zh}$, $Q_{en}$, $A_{zh}$), ($P_{en}$, $P_{zh}$, $Q_{zh}$, $A_{zh}$, $A_{en}$), ($P_{zh}$, $P_{en}$, $Q_{en}$, $A_{en}$, $A_{zh}$). For the 6th and 7th task, we mainly explored the word-alignment method. Hence, EM and F1 scores were evaluated on $A_{en}$ or $A_{zh}$.

GPUs[8]. We used spaCy[9] to tokenize sentences and generate part-of-speech and named entity tags that were used to train the DrQA model. We downloaded Chinese models[10] for spaCy to preprocess Chinese datasets. The 300-dimensional Glove word embeddings trained from 840B Web crawl data (Pennington et al., 2014) were used as our pre-trained English word embeddings while the 300-dimensional SGNS word embeddings trained from mixed-sources data (Li et al., 2018) as our pre-trained Chinese word embeddings.

### 6.5 Evaluation Results

Table 5 presents the results of the models on the development and the test data. As BERT_large is currently not available for Chinese, results on the tasks that need to use BERT_large on Chinese are not provided.

**BERT vs. Human:** In the monolingual MRC task, the English BERT_large model achieves an F1 score of 56.5%, which is 35.4 points behind human performance (91.9%), and the Chinese BERT_base model achieves an F1 score of 64.1%, which is 28.0 points behind human performance (92.1%), indicating that these tasks are difficult to accomplish with current state-of-the-art models. We also tested the English BERT_large model on a subset of the SQuAD data, which contains the same number of training instances as BiPaR. The F1 score is 86.5%, much more higher than that on BiPaR. This further suggests that BiPaR provides a very challenging dataset for MRC.

**English vs. Chinese:** On the monolingual and cross-lingual tasks, Chinese results are almost better than the English results of the same MRC model. This does not mean that Chinese MRC is easier than English. One possible reason for this may be that novels originally written in Chinese contribute to 68.5% passages of BiPaR. In the future, we plan to make the dataset more balanced between the two languages.

**Monolingual vs. Multilingual:** For the DrQA model, we observe that the multilingual training significantly improves the performance on English comparing with the monolingual training with only the English dataset. However, we do not observe this trend on BERT. This suggests that more should be explored on the multilingual MRC setting. We believe that BiPaR opens a door to new MRC approaches that are devoted to using a single model to handle MRC on multiple languages.

**Monolingual vs. Cross-lingual:** The two simple strategies via machine translation and word alignments for the cross-lingual MRC perform very bad on the four forms of cross-lingual tasks compared to the monolingual task. For ($P_{en}$, $Q_{zh}$, $A_{en}$) and ($P_{zh}$, $Q_{en}$, $A_{zh}$), the major problem is the low-quality translations of questions, especially for questions from martial arts novels. For the other two cross-lingual tasks, word alignment errors directly result in wrong answers found in passages

[8]The original batch sizes used in BERT_base/BERT_large are 12/24.

[9]https://spacy.io/

[10]https://github.com/howl-anderson/Chinese_models_for_SpaCy

of the other language.

### 6.6 Analysis of BERT and Human in Answering Different BiPaR Questions

Table 6 presents a fine-grained comparison analysis of BERT_base and human results on the English and Chinese monolingual task in terms of both answer types and question categories defined in Section 4.3. We observe that humans have absolute advantages over machines in all answer types and reasoning phenomena. However, humans exhibit different capabilities on answering different questions. They perform worse on answering paraphrasing questions, questions with description answers and questions requiring reading comprehension skills of multi-sentence reasoning than answering other questions.

The performance of BERT_base on questions with description answers is much worse than other questions, e.g., questions with person/location answers. As described in Section 4.2, descriptive answers are often complete sentences or summary statements, which are very long and difficult for machines to detect.

In terms of the relationships between questions and passages, it is clearly observed that the BERT MRC model achieves better performance on answering lexical-match questions than answering paraphrasing and summary questions. In BiPaR, we have more than 50% questions that are paraphrasing and summary questions as described in Section 4.3. Answering them requires a deep understanding of both questions and passages.

As described in Section 4.3, BiPaR produces questions that involve higher-order reading comprehension skills, such as co-reference resolution, multi-sentence reasoning, and understanding of implicit causality. It can be seen from Table 6 that the BERT model is worse on multi-sentence reasoning and implicit causality than co-reference resolution.

| Type | BERT_base | Human |
|---|---|---|
| Answer Type | | |
| Non-description | 61.97/63.22 | 91.72/95.10 |
| Description | 44.77/54.34 | 91.87/94.72 |
| Question Category | | |
| Lexical Match | 58.69/69.01 | 96.09/96.73 |
| Paraphrasing | 54.34/49.84 | 85.49/91.36 |
| Summary | 50.73/52.76 | 90.03/95.39 |
| Coreference res. | 52.35/58.97 | 91.78/98.11 |
| Multi-sentence rea. | 44.91/45.81 | 87.22/95.31 |
| Implicit causality | 39.31/57.63 | 92.54/91.28 |

Table 6: Fine-grained results in terms of different answer types and question categories on the monolingual task. The left side of the slash is the F1 score on the English data, while the right is on Chinese. All F1 scores are calculated on the 100 questions as described in Section 4.3.

## 7 Conclusion and Future Work

In this paper, we have presented the BiPaR, a bilingual parallel machine reading comprehension dataset on novels. From bilingual parallel passages of Chinese and English novels, we manually created diversified parallel questions and answers of different types via crowdsourced workers with a multi-layer quality control system. Although BiPaR is an extractive MRC dataset, in-depth analyses demonstrate that the dataset is very challenging for state-of-the-art MRC models (performing far behind human) as reading comprehension skills of co-reference resolution, inter-sentential reasoning are needed to answer BiPaR questions. We further define seven types of MRC tasks supported by BiPaR and build baseline models of monolingual, multilingual and cross-lingual MRC on BiPaR.

BiPaR can be extended in several ways. First, we would like to create more parallel triples by adding more novels to make instances more balanced between the two languages. Second, we want to create questions with non-extractive answers. Third, we are also interested in adding multi-passage questions or questions based on the entire novels to BiPaR.

## Acknowledgments

The present research was supported by the National Natural Science Foundation of China (Grant No. 61622209). We would like to thank the anonymous reviewers for their insightful comments.

## References

Rita M Aceves-Pérez, Manuel Montes-y Gómez, Luis Villaseñor-Pineda, and L Alfonso Ureña-López. 2008. Two approaches for multilingual question answering: Merging passages vs. merging answers. *International Journal of Computational Linguistics & Chinese Language Processing, Volume 13, Number 1, March 2008: Special Issue on Cross-*

*Lingual Information Retrieval and Question Answering*, 13(1):27–40.

Rita Marina Aceves-Pérez, Manuel Montes-y Gómez, and Luis Villaseñor-Pineda. 2007. Enhancing cross-language question answering by combining multiple question translations. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 485–493. Springer.

Akari Asai, Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2018. Multilingual extractive reading comprehension by runtime machine translation. *arXiv preprint arXiv:1809.03275*.

Danqi Chen. 2018. *Neural Reading Comprehension and Beyond*. Ph.D. thesis, Stanford University.

Danqi Chen, Jason Bolton, and Christopher D Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2358–2367.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada. Association for Computational Linguistics.

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. 2017. Attention-over-attention neural networks for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 593–602.

Yiming Cui, Ting Liu, Zhipeng Chen, Shijin Wang, and Guoping Hu. 2016. Consensus attention-based neural networks for chinese reading comprehension. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1777–1786.

Yiming Cui, Ting Liu, Li Xiao, Zhipeng Chen, Wentao Ma, Wanxiang Che, Shijin Wang, and Guoping Hu. 2018. A span-extraction dataset for chinese machine reading comprehension. *arXiv preprint arXiv:1810.07366*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1723–1732.

Wei He, Kai Liu, Jing Liu, Yajuan Lyu, Shiqi Zhao, Xinyan Xiao, Yuan Liu, Yizhong Wang, Hua Wu, Qiaoqiao She, et al. 2018. DuReader: a chinese machine reading comprehension dataset from real-world applications. In *Proceedings of the Workshop on Machine Reading for Question Answering*, pages 37–46.

Karl Moritz Hermann, Tomáš Kočiskỳ, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 1*, pages 1693–1701. MIT Press.

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2017. Google' s multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611.

Rudolf Kadlec, Martin Schmid, Ondřej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 908–918.

Tomas Kocisky, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gabor Melis, and Edward Grefenstette. 2018. The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794.

Shen Li, Zhe Zhao, Renfen Hu, Wensi Li, Tao Liu, and Xiaoyong Du. 2018. Analogical reasoning on chinese morphological and semantic relations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 138–143.

Frank Lin, Hideki Shima, Mengqiu Wang, and Teruko Mitamura. 2005. CMU javelin system for NTCIR5 CLQA1. In *NTCIR*.

Xiaodong Liu, Yelong Shen, Kevin Duh, and Jianfeng Gao. 2018. Stochastic answer networks for machine reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1694–1704.

Bernardo Magnini, Danilo Giampiccolo, Lili Aunimo, Christelle Ayache, Petya Osenova, Anselmo Peñas, Maarten de Rijke, Bogdan Sacaleanu, Diana Santos, and Richard Sutcliffe. 2006. The multilingual question answering track at CLEF. In *quot; In Nicoletta Calzolari; Khalid Choukri; Aldo Gangemi; Bente Maegaard; Joseph Mariani; Jan Odjik; Daniel Tapias (ed) Proceedings of the 5 th International Conference on Language Resources and Evaluation (LREC'2006)(Genoa Italy 22-28 May 2006)*.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. *choice*, 2640:660.

César de Pablo-Sánchez, Ana González-Ledesma, José Luis Martínez-Fernández, José Maria Guirao, Paloma Martinez, and Antonio Moreno-Sandoval. 2005. Miracle's 2005 approach to cross-lingual question answering. In *CLEF (Working Notes)*.

Anselmo Peñas, Pamela Forner, Richard Sutcliffe, Álvaro Rodrigo, Corina Forăscu, Iñaki Alegria, Danilo Giampiccolo, Nicolas Moreau, and Petya Osenova. 2009. Overview of respubliqa 2009: question answering evaluation over european legislation. In *Workshop of the Cross-Language Evaluation Forum for European Languages*, pages 174–196. Springer.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Joaquín Pérez, Guillermo Garrido, A Rodrigo, Lourdes Araujo, and Anselmo Peñas. 2009. Information retrieval baselines for the respubliqa task. In *Working Notes for the CLEF 2009 Workshop, Corfu, Greece*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don' t know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Siva Reddy, Danqi Chen, and Christopher D Manning. 2019. CoQA: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2017. Reasonet: Learning to stop reading in machine comprehension. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1047–1055. ACM.

Richard FE Sutcliffe, Michael Mulcahy, Igal Gabbay, Aoife O' Gorman, and Darina Slattery. 2005. Cross-language french-english question answering using the dlt system at clef 2005. In *Workshop of the Cross-Language Evaluation Forum for European Languages*, pages 502–509. Springer.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 189–198.

Yizhong Wang, Kai Liu, Jing Liu, Wei He, Yajuan Lyu, Hua Wu, Sujian Li, and Haifeng Wang. 2018. Multi-passage machine reading comprehension with cross-passage answer verification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1918–1927.

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*.

# Language Models as Knowledge Bases?

**Fabio Petroni**[1] **Tim Rocktäschel**[1,2] **Patrick Lewis**[1,2] **Anton Bakhtin**[1]
**Yuxiang Wu**[1,2] **Alexander H. Miller**[1] **Sebastian Riedel**[1,2]
[1]Facebook AI Research
[2]University College London
{fabiopetroni, rockt, plewis, yolo, yuxiangwu, ahm, sriedel}@fb.com

## Abstract

Recent progress in pretraining language models on large textual corpora led to a surge of improvements for downstream NLP tasks. Whilst learning linguistic knowledge, these models may also be storing relational knowledge present in the training data, and may be able to answer queries structured as "fill-in-the-blank" cloze statements. Language models have many advantages over structured knowledge bases: they require no schema engineering, allow practitioners to query about an open class of relations, are easy to extend to more data, and require no human supervision to train. We present an in-depth analysis of the relational knowledge already present (without fine-tuning) in a wide range of state-of-the-art pretrained language models. We find that (i) without fine-tuning, BERT contains relational knowledge competitive with traditional NLP methods that have some access to oracle knowledge, (ii) BERT also does remarkably well on open-domain question answering against a supervised baseline, and (iii) certain types of factual knowledge are learned much more readily than others by standard language model pretraining approaches. The surprisingly strong ability of these models to recall factual knowledge without any fine-tuning demonstrates their potential as unsupervised open-domain QA systems. The code to reproduce our analysis is available at https://github.com/facebookresearch/LAMA.

## 1 Introduction

Recently, pretrained high-capacity language models such as ELMo (Peters et al., 2018a) and BERT (Devlin et al., 2018a) have become increasingly important in NLP. They are optimised to either predict the next word in a sequence or some masked word anywhere in a given sequence (*e.g.* "Dante was born in [MASK] in the year 1265."). The parameters of these models appear to store vast amounts of linguistic knowledge (Peters et al., 2018b; Goldberg, 2019; Tenney et al., 2019) useful for downstream tasks. This knowledge is usually accessed either by conditioning on latent context representations produced by the original model or by using the original model weights to initialize a task-specific model which is then further fine-tuned. This type of knowledge transfer is crucial for current state-of-the-art results on a wide range of tasks.

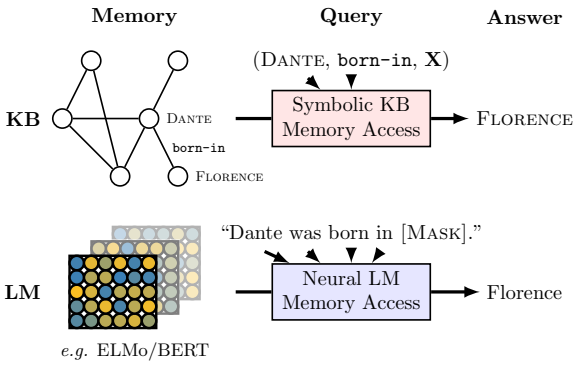


Figure 1: Querying knowledge bases (KB) and language models (LM) for factual knowledge.

In contrast, knowledge bases are effective solutions for accessing annotated gold-standard relational data by enabling queries such as (DANTE, born-in, **X**). However, in practice we often need to *extract* relational data from text or other modalities to populate these knowledge bases. This requires complex NLP pipelines involving entity extraction, coreference resolution, entity linking and relation extraction (Surdeanu and Ji, 2014)—components that often need supervised data and fixed schemas. Moreover, errors can easily propagate and accumulate throughout the pipeline. Instead, we could attempt to query neural language models for relational data by asking them to fill in masked tokens in sequences like "Dante was born



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2463–2473, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

in [MASK]", as illustrated in Figure 1. In this setting, language models come with various attractive properties: they require no schema engineering, do not need human annotations, and they support an open set of queries.

Given the above qualities of language models as potential representations of relational knowledge, we are interested in the relational knowledge already present in pretrained *off-the-shelf* language models such as ELMo and BERT. How much relational knowledge do they store? How does this differ for different types of knowledge such as facts about entities, common sense, and general question answering? How does their performance without fine-tuning compare to symbolic knowledge bases automatically extracted from text? Beyond gathering a better general understanding of these models, we believe that answers to these questions can help us design better unsupervised knowledge representations that could transfer factual and commonsense knowledge reliably to downstream tasks such as commonsense (visual) question answering (Zellers et al., 2018; Talmor et al., 2019) or reinforcement learning (Branavan et al., 2011; Chevalier-Boisvert et al., 2018; Bahdanau et al., 2019; Luketina et al., 2019).

For the purpose of answering the above questions we introduce the *LAMA* (LAnguage Model Analysis) probe, consisting of a set of knowledge sources, each comprised of a set of facts. We define that a pretrained language model *knows* a fact (SUBJECT, `relation`, OBJECT) such as (DANTE, `born-in`, FLORENCE) if it can successfully predict masked objects in cloze sentences such as "Dante was born in ____" expressing that fact. We test for a variety of types of knowledge: relations between entities stored in Wikidata, common sense relations between concepts from ConceptNet, and knowledge necessary to answer natural language questions in SQuAD. In the latter case we manually map a subset of SQuAD questions to cloze sentences.

Our investigation reveals that (i) the largest BERT model from Devlin et al. (2018b) (BERT-large) captures (accurate) relational knowledge comparable to that of a knowledge base extracted with an off-the-shelf relation extractor and an oracle-based entity linker from a corpus known to express the relevant knowledge, (ii) factual knowledge can be recovered surprisingly well from pretrained language models, however, for some relations (particularly *N*-to-*M* relations) performance is very poor, (iii) BERT-large consistently outperforms other language models in recovering factual and commonsense knowledge while at the same time being more robust to the phrasing of a query, and (iv) BERT-large achieves remarkable results for open-domain QA, reaching 57.1% precision@10 compared to 63.5% of a knowledge base constructed using a task-specific supervised relation extraction system.

## 2 Background

In this section we provide background on language models. Statistics for the models that we include in our investigation are summarized in Table 1.

### 2.1 Unidirectional Language Models

Given an input sequence of tokens $\mathbf{w} = [w_1, w_2, \ldots, w_N]$, unidirectional language models commonly assign a probability $p(\mathbf{w})$ to the sequence by factorizing it as follows

$$p(\mathbf{w}) = \prod_t p(w_t \,|\, w_{t-1}, \ldots, w_1). \qquad (1)$$

A common way to estimate this probability is using neural language models (Mikolov and Zweig, 2012; Melis et al., 2017; Bengio et al., 2003) with

$$p(w_t \,|\, w_{t-1}, \ldots, w_1) = \text{softmax}(\mathbf{W}\mathbf{h}_t + \mathbf{b}) \qquad (2)$$

where $\mathbf{h}_t \in \mathbb{R}^k$ is the output vector of a neural network at position $t$ and $\mathbf{W} \in \mathbb{R}^{|\mathcal{V}| \times k}$ is a learned parameter matrix that maps $\mathbf{h}_t$ to unnormalized scores for every word in the vocabulary $\mathcal{V}$. Various neural language models then mainly differ in how they compute $\mathbf{h}_t$ given the word history, *e.g.*, by using a multi-layer perceptron (Bengio et al., 2003; Mikolov and Zweig, 2012), convolutional layers (Dauphin et al., 2017), recurrent neural networks (Zaremba et al., 2014; Merity et al., 2016; Melis et al., 2017) or self-attention mechanisms (Radford et al., 2018; Dai et al., 2019; Radford et al., 2019).

**fairseq-fconv**: Instead of commonly used recurrent neural networks, Dauphin et al. (2017) use multiple layers of gated convolutions. We use the pretrained model in the fairseq[1] library in our study. It has been trained on the WikiText-103 corpus introduced by Merity et al. (2016).

[1] https://github.com/pytorch/fairseq

| Model | Base Model | #Parameters | Training Corpus | Corpus Size |
|---|---|---|---|---|
| fairseq-fconv (Dauphin et al., 2017) | ConvNet | 324M | WikiText-103 | 103M Words |
| Transformer-XL (large) (Dai et al., 2019) | Transformer | 257M | WikiText-103 | 103M Words |
| ELMo (original) (Peters et al., 2018a) | BiLSTM | 93.6M | Google Billion Word | 800M Words |
| ELMo 5.5B (Peters et al., 2018a) | BiLSTM | 93.6M | Wikipedia (en) & WMT 2008-2012 | 5.5B Words |
| BERT (base) (Devlin et al., 2018a) | Transformer | 110M | Wikipedia (en) & BookCorpus | 3.3B Words |
| BERT (large) (Devlin et al., 2018a) | Transformer | 340M | Wikipedia (en) & BookCorpus | 3.3B Words |

Table 1: Language models considered in this study.

**Transformer-XL**: Dai et al. (2019) introduce a large-scale language model based on the Transformer (Vaswani et al., 2017). Transformer-XL can take into account a longer history by caching previous outputs and by using relative instead of absolute positional encoding. It achieves a test perplexity of 18.3 on the WikiText-103 corpus.

### 2.2 Bidirectional "Language Models"[2]

So far, we have looked at language models that predict the next word given a history of words. However, in many downstream applications we mostly care about having access to contextual representations of words, *i.e.*, word representations that are a function of the entire context of a unit of text such as a sentence or paragraph, and not only conditioned on previous words. Formally, given an input sequence $\mathbf{w} = [w_1, w_2, \ldots, w_N]$ and a position $1 \leq i \leq N$, we want to estimate $p(w_i) = p(w_i \mid w_1, \ldots, w_{i-1}, w_{i+1}, \ldots, w_N)$ using the left and right context of that word.

**ELMo**: To estimate this probability, Peters et al. (2018a) propose running a forward and backward LSTM (Hochreiter and Schmidhuber, 1997), resulting in $\overrightarrow{\mathbf{h}}_i$ and $\overleftarrow{\mathbf{h}}_i$ which consequently are used to calculate a forward and backward language model log-likelihood. Their model, ELMo, uses multiple layers of LSTMs and it has been pretrained on the Google Billion Word dataset. Another version of the model, ELMo 5.5B, has been trained on the English Wikipedia and monolingual news crawl data from WMT 2008-2012.

**BERT**: Instead of a standard language model objective, Devlin et al. (2018a) propose to sample positions in the input sequence randomly and to learn to fill the word at the masked position. To this end, they employ a Transformer architecture and train it on the BookCorpus (Zhu et al., 2015) as well as a crawl of English Wikipedia. In addition to this pseudo language model objective, they use an auxiliary binary classification objective to predict whether a particular sentence follows the given sequence of words.

[2]Contextual representation models (Tenney et al., 2019) might be a better name, but we keep calling them language models for simplicity.

## 3 Related Work

Many studies have investigated pretrained word representations, sentence representations, and language models. Existing work focuses on understanding linguistic and semantic properties of word representations or how well pretrained sentence representations and language models transfer linguistic knowledge to downstream tasks. In contrast, our investigation seeks to answer to what extent pretrained language models store factual and commonsense knowledge by comparing them with symbolic knowledge bases populated by traditional relation extraction approaches.

Baroni et al. (2014) present a systematic comparative analysis between neural word representation methods and more traditional count-based distributional semantic methods on lexical semantics tasks like semantic relatedness and concept categorization. They find that neural word representations outperform count-based distributional methods on the majority of the considered tasks. Hill et al. (2015) investigate to what degree word representations capture semantic meaning as measured by similarity between word pairs.

Marvin and Linzen (2018) assess the grammaticality of pretrained language models. Their dataset consists of sentence pairs with a grammatical and an ungrammatical sentence. While a good language model should assign higher probability to the grammatical sentence, they find that LSTMs do not learn syntax well.

Another line of work investigates the ability of pretrained sentence and language models to transfer knowledge to downstream natural language understanding tasks (Wang et al., 2018). While such an analysis sheds light on the transfer-learning

abilities of pretrained models for understanding short pieces of text, it provides little insight into whether these models can compete with traditional approaches to representing knowledge like symbolic knowledge bases.

More recently, McCoy et al. (2019) found that for natural language inference, a model based on BERT learns to rely heavily on fallible syntactic heuristics instead of a deeper understanding of the natural language input. Peters et al. (2018b) found that lower layers in ELMo specialize on local syntactic relationships, while higher layers can learn to model long-range relationships. Similarly, Goldberg (2019) found that BERT captures English syntactic phenomena remarkably well. Tenney et al. (2019) investigate to what extent language models encode sentence structure for different syntactic and semantic phenomena and found that they excel for the former but only provide small improvements for tasks that fall into the latter category. While this provides insights into the linguistic knowledge of language models, it does not provide insights into their factual and commonsense knowledge.

Radford et al. (2018) introduce a pretrained language model based on the Transformer which they termed generative pretraining (GPTv1). The first version of GPT (Radford et al., 2018) has been trained on the Book Corpus (Zhu et al., 2015) containing 7000 books. The closest to our investigation is the work by Radford et al. (2019) which introduces GPTv2 and investigates how well their language model does zero-shot transfer to a range of downstream tasks. They find that GPTv2 achieves an $F_1$ of 55 for answering questions in CoQA (Reddy et al., 2018) and 4.1% accuracy on the Natural Questions dataset (Kwiatkowski et al., 2019), in both cases without making use of annotated question-answer pairs or an information retrieval step. While these results are encouraging and hint at the ability of very large pretrained language models to memorize factual knowledge, the large GPTv2 model has not been made public and the publicly available small version achieves less than 1% on Natural Questions (5.3 times worse than the large model). Thus, we decided to not include GPTv2 in our study. Similarly, we do not include GPTv1 in this study as it uses a limited lower-cased vocabulary, making it incompatible to the way we assess the other language models.

## 4 The LAMA Probe

We introduce the *LAMA* (LAnguage Model Analysis) probe to test the factual and commonsense knowledge in language models. It provides a set of knowledge sources which are composed of a corpus of facts. Facts are either subject-relation-object triples or question-answer pairs. Each fact is converted into a cloze statement which is used to query the language model for a missing token. We evaluate each model based on how highly it ranks the ground truth token against every other word in a fixed candidate vocabulary. This is similar to ranking-based metrics from the knowledge base completion literature (Bordes et al., 2013; Nickel et al., 2016). Our assumption is that models which rank ground truth tokens high for these cloze statements have more factual knowledge. We discuss each step in detail next and provide considerations on the probe below.

### 4.1 Knowledge Sources

To assess the different language models in Section 2, we cover a variety of sources of factual and commonsense knowledge. For each source, we describe the origin of fact triples (or question-answer pairs), how we transform them into cloze templates, and to what extent aligned texts exist in Wikipedia that are known to express a particular fact. We use the latter information in supervised baselines that extract knowledge representations directly from the aligned text.

#### 4.1.1 Google-RE

The Google-RE corpus[3] contains ~60K facts manually extracted from Wikipedia. It covers five relations but we consider only three of them, namely "place of birth", "date of birth" and "place of death". We exclude the other two because they contain mainly multi-tokens objects that are not supported in our evaluation. We manually define a template for each considered relation, *e.g.*, "[S] was born in [O]" for "place of birth". Each fact in the Google-RE dataset is, by design, manually aligned to a short piece of Wikipedia text supporting it.

#### 4.1.2 T-REx

The T-REx knowledge source is a subset of Wikidata triples. It is derived from the T-REx

[3] https://code.google.com/archive/p/relation-extraction-corpus/

dataset (Elsahar et al., 2018) and is much larger than Google-RE with a broader set of relations. We consider 41 Wikidata relations and subsample at most 1000 facts per relation. As with the Google-RE corpus, we manually define a template for each relation (see Table 3 for some examples). In contrast to the Google-RE knowledge source, T-REx facts were automatically aligned to Wikipedia and hence this alignment can be noisy. However, Elsahar et al. (2018) report an accuracy of 97.8% for the alignment technique over a test set.

#### 4.1.3 ConceptNet

ConceptNet (Speer and Havasi, 2012) is a multilingual knowledge base, initially built on top of Open Mind Common Sense (OMCS) sentences. OMCS represents commonsense relationships between words and/or phrases. We consider facts from the English part of ConceptNet that have single-token objects covering 16 relations. For these ConceptNet triples, we find the OMCS sentence that contains both the subject and the object. We then mask the object within the sentence and use the sentence as template for querying language models. If there are several sentences for a triple, we pick one at random. Note that for this knowledge source there is no explicit alignment of facts to Wikipedia sentences.

#### 4.1.4 SQuAD

SQuAD (Rajpurkar et al., 2016) is a popular question answering dataset. We select a subset of 305 context-insensitive questions from the SQuAD development set with single token answers. We manually create cloze-style questions from these questions, *e.g.*, rewriting "Who developed the theory of relativity?" as "The theory of relativity was developed by ____". For each question and answer pair, we know that the corresponding fact is expressed in Wikipedia since this is how SQuAD was created.

### 4.2 Models

We consider the following pretrained case-sensitive language models in our study (see Table 1): fairseq-fconv (*Fs*), Transformer-XL large (*Txl*), ELMo original (*Eb*), ELMo 5.5B (*E5B*), BERT-base (*Bb*) and BERT-large (*Bl*). We use the natural way of generating tokens for each model by following the definition of the training objective function.

Assume we want to compute the generation for the token at position $t$. For unidirectional language models, we use the network output ($\mathbf{h}_{t-1}$) just before the token to produce the output layer softmax. For ELMo we consider the output just before ($\overrightarrow{\mathbf{h}}_{t-1}$) for the forward direction and just after ($\overleftarrow{\mathbf{h}}_{t+1}$) for the backward direction. Following the loss definition in (Peters et al., 2018a), we average forward and backward probabilities from the corresponding softmax layers. For BERT, we mask the token at position $t$, and we feed the output vector corresponding to the masked token ($\mathbf{h}_t$) into the softmax layer. To allow a fair comparison, we let models generate over a unified vocabulary, which is the intersection of the vocabularies for all considered models (~21K case-sensitive tokens).

### 4.3 Baselines

To compare language models to canonical ways of using off-the-shelf systems for extracting symbolic knowledge and answering questions, we consider the following baselines.

**Freq**: For a subject and relation pair, this baseline ranks words based on how frequently they appear as objects for the given relation in the test data. It indicates the upper bound performance of a model that always predicts the same objects for a particular relation.

**RE**: For the relation-based knowledge sources, we consider the pretrained Relation Extraction (RE) model of Sorokin and Gurevych (2017). This model was trained on a subcorpus of Wikipedia annotated with Wikidata relations. It extracts relation triples from a given sentence using an LSTM-based encoder and an attention mechanism. Based on the alignment information from the knowledge sources, we provide the relation extractor with the sentences known to express the test facts. Using these datasets, RE constructs a knowledge graph of triples. At test time, we query this graph by finding the subject entity and then rank all objects in the correct relation based on the confidence scores returned by RE. We consider two versions of this procedure that differ in how the entity linking is implemented: $\mathbf{RE}_n$ makes use of a naïve entity linking solution based on exact string matching, while $\mathbf{RE}_o$ uses an oracle for entity linking in addition to string matching. In other words, assume we query for the object $o$ of a test subject-relation fact $(s, r, o)$ expressed in a sentence $x$. If RE has extracted any triple $(s', r, o')$ from that sen-

tence $x$, $s'$ will be linked to $s$ and $o'$ to $o$. In practice, this means RE can return the correct solution $o$ if *any* relation instance of the right type was extracted from $x$, regardless of whether it has a wrong subject or object.

**DrQA**: Chen et al. (2017) introduce DrQA, a popular system for open-domain question answering. DrQA predicts answers to natural language questions using a two step pipeline. First, a TF/IDF information retrieval step is used to find relevant articles from a large store of documents (e.g. Wikipedia). On the retrieved top $k$ articles, a neural reading comprehension model then extracts answers. To avoid giving the language models a competitive advantage, we constrain the predictions of DrQA to single-token answers.

### 4.4 Metrics

We consider rank-based metrics and compute results per relation along with mean values across all relations. To account for multiple valid objects for a subject-relation pair (*i.e.*, for N-M relations), we follow Bordes et al. (2013) and remove from the candidates when ranking at test time all other valid objects in the training data other than the one we test. We use the mean precision at k ($P@k$). For a given fact, this value is 1 if the object is ranked among the top k results, and 0 otherwise.

### 4.5 Considerations

There are several important design decisions we made when creating the LAMA probe. Below we give more detailed justifications for these decisions.

**Manually Defined Templates** For each relation we manually define a template that queries for the object slot in that relation. One can expect that the choice of templates has an impact on the results, and this is indeed the case: for some relations we find both worse and better ways to query for the same information (with respect to a given model) by using an alternate template. We argue that this means we are measuring a *lower* bound for what language models know. We make this argument by analogy with traditional knowledge bases: they only have a *single* way of querying knowledge for a specific relation, namely by using the relation id of that relation, and this way is used to measure their accuracy. For example, if the relation ID is `works-For` and the user asks for `is-working-for`, the accuracy of the KG would be 0.

**Single Token** We only consider single token objects as our prediction targets. The reason we include this limitation is that multi-token decoding adds a number of additional tuneable parameters (beam size, candidate scoring weights, length normalization, n-gram repetition penalties, etc.) that obscure the knowledge we are trying to measure. Moreover, well-calibrated multi-token generation is still an active research area, particularly for bidirectional models (see *e.g.* Welleck et al. (2019)).

**Object Slots** We choose to *only* query object slots in triples, as opposed to subject or relation slots. By including reverse relations (e.g. `contains` and `contained-by`) we can also query subject slots. We do not query relation slots for two reasons. First, surface form realisations of relations will span several tokens, and as we discussed above, this poses a technical challenge that is not in the scope of this work. Second, even if we could easily predict multi-token phrases, relations can generally be expressed with many different wordings, making it unclear what the gold standard pattern for a relation should be, and how to measure accuracy in this context.

**Intersection of Vocabularies** The models that we considered are trained with different vocabularies. For instance, ELMo uses a list of ~800K tokens while BERT considers only ~30K tokens. The size of the vocabulary can influence the performance of a model for the LAMA probe. Specifically, the larger the vocabulary the harder it would be to rank the gold token at the top. For this reason we considered a common vocabulary of ~21K case-sensitive tokens that are obtained from the intersection of the vocabularies for all considered models. To allow a fair comparison, we let every model rank only tokens in this joint vocabulary.

## 5 Results

We summarize the main results in Table 2, which shows the mean precision at one (P@1) for the different models across the set of corpora considered. In the remainder of this section, we discuss the results for each corpus in detail.

**Google-RE** We query the LMs using a standard cloze template for each relation. The base and large versions of BERT both outperform all other models by a substantial margin. Furthermore, they

| Corpus | Relation | Statistics | | Baselines | | KB | | LM | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | #Facts | #Rel | Freq | DrQA | $RE_n$ | $RE_o$ | Fs | Txl | Eb | E5B | Bb | Bl |
| Google-RE | birth-place | 2937 | 1 | 4.6 | - | 3.5 | 13.8 | 4.4 | 2.7 | 5.5 | 7.5 | 14.9 | **16.1** |
| | birth-date | 1825 | 1 | 1.9 | - | 0.0 | **1.9** | 0.3 | 1.1 | 0.1 | 0.1 | 1.5 | 1.4 |
| | death-place | 765 | 1 | 6.8 | - | 0.1 | 7.2 | 3.0 | 0.9 | 0.3 | 1.3 | 13.1 | **14.0** |
| | Total | 5527 | 3 | 4.4 | - | 1.2 | 7.6 | 2.6 | 1.6 | 2.0 | 3.0 | 9.8 | **10.5** |
| T-REx | 1-1 | 937 | 2 | 1.78 | - | 0.6 | 10.0 | 17.0 | 36.5 | 10.1 | 13.1 | 68.0 | **74.5** |
| | *N*-1 | 20006 | 23 | 23.85 | - | 5.4 | **33.8** | 6.1 | 18.0 | 3.6 | 6.5 | 32.4 | 34.2 |
| | *N*-*M* | 13096 | 16 | 21.95 | - | 7.7 | **36.7** | 12.0 | 16.5 | 5.7 | 7.4 | 24.7 | 24.3 |
| | Total | 34039 | 41 | 22.03 | - | 6.1 | **33.8** | 8.9 | 18.3 | 4.7 | 7.1 | 31.1 | 32.3 |
| ConceptNet | Total | 11458 | 16 | 4.8 | - | - | - | 3.6 | 5.7 | 6.1 | 6.2 | 15.6 | **19.2** |
| SQuAD | Total | 305 | - | - | **37.5** | - | - | 3.6 | 3.9 | 1.6 | 4.3 | 14.1 | 17.4 |

Table 2: Mean precision at one (P@1) for a frequency baseline (Freq), DrQA, a relation extraction with naïve entity linking ($RE_n$), oracle entity linking ($RE_o$), fairseq-fconv (Fs), Transformer-XL large (Txl), ELMo original (Eb), ELMo 5.5B (E5B), BERT-base (Bb) and BERT-large (Bl) across the set of evaluation corpora.

obtain a 2.2 and 2.9 respective average accuracy improvement over the oracle-based RE baseline. This is particularly surprising given that with the gold-aligned Google-RE source we know for certain that the oracle RE baseline has seen at least one sentence expressing each test fact. Moreover, the RE baseline was given substantial help through an entity linking oracle.

It is worth pointing out that while BERT-large does better, this does not mean it does so for the right reasons. Although the aligned Google-RE sentences are likely in its training set (as they are part of Wikipedia and BERT has been trained on Wikipedia), it might not "understand" them to produce these results. Instead, it could have learned associations of objects with subjects from co-occurrence patterns.

**T-REx** The knowledge source derived from Google-RE contains relatively few facts and only three relations. Hence, we perform experiments on the larger set of facts and relations in T-REx. We find that results are generally consistent with Google-RE. Again, the performance of BERT in retrieving factual knowledge are close to the performance obtained by automatically building a knowledge base with an off-the-shelf relation extraction system and oracle-based entity linking. Broken down by relation type, the performance of BERT is very high for 1-to-1 relations (*e.g.*, *capital of*) and low for N-to-M relations.

Note that a downstream model could learn to make use of knowledge in the output representations of a language model even if the correct answer is not ranked first but high enough (*i.e.* a hint

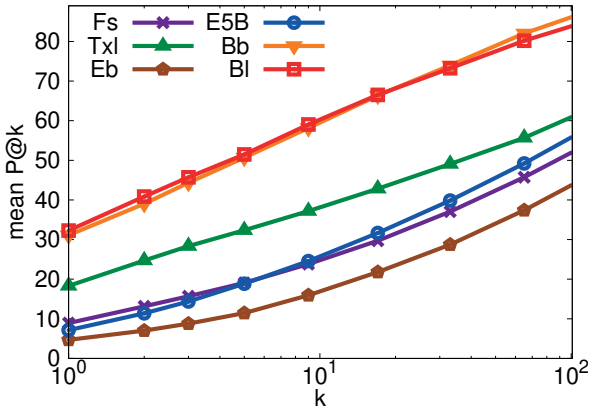


Figure 2: Mean P@k curve for T-REx varying k. Base-10 log scale for X axis.

about the correct answer can be extracted from the output representation). Figure 2 shows the mean P@k curves for the considered models. For BERT, the correct object is ranked among the top ten in around 60% of the cases and among the top 100 in 80% of the cases.

To further investigate why BERT achieves such strong results, we compute the Pearson correlation coefficient between the *P@1* and a set of metrics that we report in Figure 3. We notice, for instance, that the number of times an object is mentioned in the training data positively correlates with performance while the same is not true for the subject of a relation. Furthermore, the log probability of a prediction is strongly positively correlated with P@1. Thus, when BERT has a high confidence in its prediction, it is often correct. Performance is also positively correlated with the cosine similarity between subject and object vectors, and

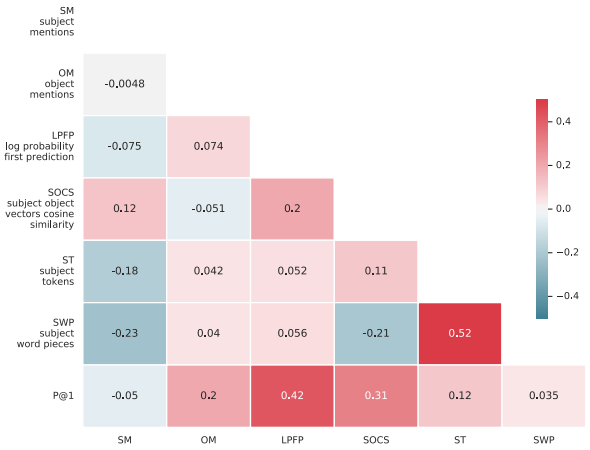


Figure 3: Pearson correlation coefficient for the P@1 of the BERT-large model on T-REx and a set of metrics: SM and OM refer to the number of times a subject and an object are mentioned in the BERT training corpus[4]respectively; LPFP is the log probability score associated with the first prediction; SOCS is the cosine similarity between subject and object vectors (we use spaCy[5]); ST and SWP are the number of tokens in the subject with a standard tokenization and the BERT WordPiece tokenization respectively.

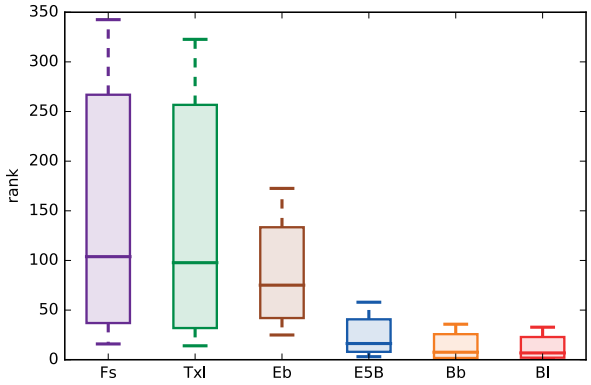


Figure 4: Average rank distribution for 10 different mentions of 100 random facts per relation in T-REx. ELMo 5.5B and both variants of BERT are least sensitive to the framing of the query but also are the most likely to have seen the query sentence during training.

slightly with the number of tokens in the subject.

Table 3 shows randomly picked examples for the generation of BERT-large for cloze template queries. We find that BERT-large generally predicts objects of the correct type, even when the predicted object itself is not correct.

To understand how the performance of a pretrained language model varies with different ways of querying for a particular fact, we analyze a maximum of 100 random facts per relation for which we randomly select 10 aligned sentences in Wikipedia from T-REx.[6] In each of the sentences, we mask the object of the fact, and ask the model to predict it. For several of our language models this also tests their ability to memorize and recall sentences from the training data since as the models have been trained on Wikipedia (see Table 1).

Figure 4 shows the average distribution of the rank for ten queries per fact. The two BERT models and ELMo 5.5B exhibit the lowest variability while ranking the correct object close to the top on average. Surprisingly, the performance of ELMo original is not far from BERT, even though this model did not see Wikipedia during training. Fairseq-fconv and Transformer-XL experience a higher variability in their predictions. Note that BERT and ELMo 5.5B have been trained on a larger portion of Wikipedia than fairseq-fconv and Transformer-XL and may have seen more sentences containing the test queries during training.

**ConceptNet** The results on the ConceptNet corpus are in line with those reported for retrieving factual knowledge in Google-RE and T-REx. The BERT-large model consistently achieves the best performance, and it is able to retrieve commonsense knowledge at a similar level to factual knowledge. The lower half of Table 3 shows generations by BERT-large for randomly sampled examples. Some of the concepts generated by the language models are surprisingly reasonable in addition to being syntactically correct.

**SQuAD** Next we evaluate our system on open-domain cloze-style question answering and compare against the supervised DrQA model. Table 2 shows a performance gap between BERT-large and the DrQA open-domain QA system on our cloze SQuAD task. Again, note that the pretrained language model is completely unsupervised, it is not fine-tuned, and it has no access to a dedicated information retrieval system. Moreover, when comparing DrQA and BERT-large in terms of P@10, we find that gap is remarkably small (57.1 for BERT-large and 63.5 for DrQA).

## 6 Discussion and Conclusion

We presented a systematic analysis of the factual and commonsense knowledge in publicly available pretrained language models *as is* and found

[5]The original training corpus is not available, we created our version using the same sources.

[5]https://spacy.io

[6]We exclude all facts with less than 10 alignments.

| | Relation | Query | Answer | Generation |
|---|---|---|---|---|
| T-Rex | P19 | Francesco Bartolomeo Conti was born in ____. | Florence | Rome [-1.8] , **Florence [-1.8]** , Naples [-1.9] , Milan [-2.4] , Bologna [-2.5] |
| | P20 | Adolphe Adam died in ____. | Paris | **Paris [-0.5]** , London [-3.5] , Vienna [-3.6] , Berlin [-3.8] , Brussels [-4.0] |
| | P279 | English bulldog is a subclass of ____. | dog | dogs [-0.3] , breeds [-2.2] , **dog [-2.4]** , cattle [-4.3] , sheep [-4.5] |
| | P37 | The official language of Mauritius is ____. | English | **English [-0.6]** , French [-0.9] , Arabic [-6.2] , Tamil [-6.7] , Malayalam [-7.0] |
| | P413 | Patrick Oboya plays in ____ position. | midfielder | centre [-2.0] , center [-2.2] , **midfielder [-2.4]** , forward [-2.4] , midfield [-2.7] |
| | P138 | Hamburg Airport is named after ____. | Hamburg | Hess [-7.0] , Hermann [-7.1] , Schmidt [-7.1] , **Hamburg [-7.5]** , Ludwig [-7.5] |
| | P364 | The original language of Mon oncle Benjamin is ____. | French | **French [-0.2]** , Breton [-3.3] , English [-3.8] , Dutch [-4.2] , German [-4.9] |
| | P54 | Dani Alves plays with ____ . | Barcelona | Santos [-2.4] , Porto [-2.5] , Sporting [-3.1] , Brazil [-3.3] , Portugal [-3.7] |
| | P106 | Paul Toungui is a ____ by profession . | politician | lawyer [-1.1] , journalist [-2.4] , teacher [-2.7] , doctor [-3.0] , physician [-3.7] |
| | P527 | Sodium sulfide consists of ____. | sodium | water [-1.2] , sulfur [-1.7] , **sodium [-2.5]** , zinc [-2.8] , salt [-2.9] |
| | P102 | Gordon Scholes is a member of the ____ political party. | Labor | Labour [-1.3] , Conservative [-1.6] , Green [-2.4] , Liberal [-2.9] , **Labor [-2.9]** |
| | P530 | Kenya maintains diplomatic relations with ____. | Uganda | India [-3.0] , **Uganda [-3.2]** , Tanzania [-3.5] , China [-3.6] , Pakistan [-3.6] |
| | P176 | iPod Touch is produced by ____. | Apple | **Apple [-1.6]** , Nokia [-1.7] , Sony [-2.0] , Samsung [-2.6] , Intel [-3.1] |
| | P30 | Bailey Peninsula is located in ____. | Antarctica | **Antarctica [-1.4]** , Bermuda [-2.2] , Newfoundland [-2.5] , Alaska [-2.7] , Canada [-3.1] |
| | P178 | JDK is developed by ____. | Oracle | IBM [-2.0] , Intel [-2.3] , Microsoft [-2.5] , HP [-3.4] , Nokia [-3.5] |
| | P1412 | Carl III used to communicate in ____. | Swedish | German [-1.6] , Latin [-1.9] , French [-2.4] , English [-3.0] , Spanish [-3.0] |
| | P17 | Sunshine Coast, British Columbia is located in ____. | Canada | **Canada [-1.2]** , Alberta [-2.8] , Yukon [-2.9] , Labrador [-3.4] , Victoria [-3.4] |
| | P39 | Pope Clement VII has the position of ____ . | pope | cardinal [-2.4] , Pope [-2.5] , **pope [-2.6]** , President [-3.1] , Chancellor [-3.2] |
| | P264 | Joe Cocker is represented by music label ____. | Capitol | EMI [-2.6] , BMG [-2.6] , Universal [-2.8] , **Capitol [-3.2]** , Columbia [-3.3] |
| | P276 | London Jazz Festival is located in ____. | London | **London [-0.3]** , Greenwich [-3.2] , Chelsea [-4.0] , Camden [-4.6] , Stratford [-4.8] |
| | P127 | Border TV is owned by ____. | ITV | Sky [-3.1] , **ITV [-3.3]** , Global [-3.4] , Frontier [-4.1] , Disney [-4.3] |
| | P103 | The native language of Mammootty is ____. | Malayalam | **Malayalam [-0.2]** , Tamil [-2.1] , Telugu [-4.8] , English [-5.2] , Hindi [-5.6] |
| | P495 | The Sharon Cuneta Show was created in ____. | Philippines | Manila [-3.2] , **Philippines [-3.6]** , February [-3.7] , December [-3.8] , Argentina [-4.0] |
| ConceptNet | AtLocation | You are likely to find a overflow in a ____. | drain | sewer [-3.1] , canal [-3.2] , toilet [-3.3] , stream [-3.6] , **drain [-3.6]** |
| | CapableOf | Ravens can ____. | fly | **fly [-1.5]** , fight [-1.8] , kill [-2.2] , die [-3.2] , hunt [-3.4] |
| | CausesDesire | Joke would make you want to ____. | laugh | cry [-1.7] , die [-1.7] , **laugh [-2.0]** , vomit [-2.6] , scream [-2.6] |
| | Causes | Sometimes virus causes ____. | infection | disease [-1.2] , cancer [-2.0] , **infection [-2.6]** , plague [-3.3] , fever [-3.4] |
| | HasA | Birds have ____. | feathers | wings [-1.8] , nests [-3.1] , **feathers [-3.2]** , died [-3.7] , eggs [-3.9] |
| | HasPrerequisite | Typing requires ____. | speed | patience [-3.5] , precision [-3.6] , registration [-3.8] , accuracy [-4.0] , **speed [-4.1]** |
| | HasProperty | Time is ____. | finite | short [-1.7] , passing [-1.8] , precious [-2.9] , irrelevant [-3.2] , gone [-4.0] |
| | MotivatedByGoal | You would celebrate because you are ____. | alive | happy [-2.4] , human [-3.3] , **alive [-3.3]** , young [-3.6] , free [-3.9] |
| | ReceivesAction | Skills can be ____. | taught | acquired [-2.5] , useful [-2.5] , learned [-2.8] , combined [-3.9] , varied [-3.9] |
| | UsedFor | A pond is for ____. | fish | swimming [-1.3] , fishing [-1.4] , bathing [-2.0] , **fish [-2.8]** , recreation [-3.1] |

Table 3: Examples of generation for BERT-large. The last column reports the top five tokens generated together with the associated log probability (in square brackets).

that BERT-large is able to recall such knowledge better than its competitors and at a level remarkably competitive with non-neural and supervised alternatives. Note that we did *not* compare the ability of the corresponding architectures and objectives to capture knowledge in a given body of text but rather focused on the knowledge present in the weights of existing pretrained models that are being used as starting points for many researchers' work. Understanding which aspects of data our commonly-used models and learning algorithms are capturing is a crucial field of research and this paper complements the many studies focused on the learned linguistic properties of the data.

We found that it is non-trivial to extract a knowledge base from text that performs on par to directly using pretrained BERT-large. This is despite providing our relation extraction baseline with only data that is likely expressing target facts, thus reducing potential for false negatives, as well as using a generous entity-linking oracle. We suspected BERT might have an advantage due to the larger amount of data it has processed, so we added Wikitext-103 as additional data to the relation extraction system and observed no significant change in performance. This suggests that while relation extraction performance might be difficult to improve with more data, language models trained on ever growing corpora might become a viable alternative to traditional knowledge bases extracted from text in the future.

In addition to testing future pretrained language models using the LAMA probe, we are interested in quantifying the variance of recalling factual knowledge with respect to varying natural language templates. Moreover, assessing multi-token answers remains an open challenge for our evaluation setup.

## Acknowledgments

We would like to thank the reviewers for their thoughtful comments and efforts towards improving our manuscript. In addition, we would like to acknowledge three frameworks that were used in our experiments: AllenNLP[7], Fairseq[8] and the Hugging Face PyTorch-Transformers[9] library.

## References

Dzmitry Bahdanau, Felix Hill, Jan Leike, Edward Hughes, Pushmeet Kohli, and Edward Grefenstette. 2019. Learning to understand goal specifications by

[7] https://github.com/allenai/allennlp
[8] https://github.com/pytorch/fairseq
[9] https://github.com/huggingface/pytorch-transformers

modelling reward. In *International Conference on Learning Representations (ICLR)*.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 238–247.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 2787–2795.

S. R. K. Branavan, David Silver, and Regina Barzilay. 2011. Learning to win by reading manuals in a monte-carlo framework. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*, pages 268–277.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. *CoRR*, abs/1704.00051.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. 2018. Babyai: First steps towards grounded language learning with a human in the loop. *CoRR*, abs/1810.08272.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *CoRR*, abs/1901.02860.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 933–941.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018a. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018b. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs]*. ArXiv: 1810.04805.

Hady Elsahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon Hare, Frederique Laforest, and Elena Simperl. 2018. T-rex: A large scale alignment of natural language with knowledge base triples. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*.

Yoav Goldberg. 2019. Assessing bert's syntactic abilities. *CoRR*, abs/1901.05287.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Rhinehart, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, et al. 2019. Natural questions: a benchmark for question answering research.

Jelena Luketina, Nantas Nardelli, Gregory Farquhar, Jakob Foerster, Jacob Andreas, Edward Grefenstette, Shimon Whiteson, and Tim Rocktäschel. 2019. A Survey of Reinforcement Learning Informed by Natural Language. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, August 10-16 2019, Macao, China.*

Rebecca Marvin and Tal Linzen. 2018. Targeted syntactic evaluation of language models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 1192–1202.

R. Thomas McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference.

Gábor Melis, Chris Dyer, and Phil Blunsom. 2017. On the state of the art of evaluation in neural language models. *CoRR*, abs/1707.05589.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *CoRR*, abs/1609.07843.

Tomas Mikolov and Geoffrey Zweig. 2012. Context dependent recurrent neural network language model. In *2012 IEEE Spoken Language Technology Workshop (SLT), Miami, FL, USA, December 2-5, 2012*, pages 234–239.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2016. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018a. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 2227–2237.

Matthew E. Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. 2018b. Dissecting contextual word embeddings: Architecture and representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 1499–1509.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Siva Reddy, Danqi Chen, and Christopher D. Manning. 2018. Coqa: A conversational question answering challenge. *CoRR*, abs/1808.07042.

Daniil Sorokin and Iryna Gurevych. 2017. Context-aware representations for knowledge base relation extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 1784–1789.

Robert Speer and Catherine Havasi. 2012. Representing general relational knowledge in conceptnet 5. In *LREC*, pages 3679–3686.

Mihai Surdeanu and Heng Ji. 2014. Overview of the English Slot Filling Track at the TAC2014 Knowledge Base Population Evaluation. page 15.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4149–4158.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Sam Bowman, Dipanjan Das, and Ellie Pavlick. 2019. What do you learn from context? probing for sentence structure in contextualized word representations. In *International Conference on Learning Representations*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6000–6010.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the Workshop: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@EMNLP 2018, Brussels, Belgium, November 1, 2018*, pages 353–355.

Sean Welleck, Kianté Brantley, Hal Daumé III, and Kyunghyun Cho. 2019. Non-monotonic sequential text generation. *arXiv preprint arXiv:1902.02192*.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *CoRR*, abs/1409.2329.

Rowan Zellers, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2018. From recognition to cognition: Visual commonsense reasoning. *CoRR*, abs/1811.10830.

Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 19–27.

# NumNet: Machine Reading Comprehension with Numerical Reasoning

**Qiu Ran**[1]*, **Yankai Lin**[1]*, **Peng Li**[1], **Jie Zhou**[1], **Zhiyuan Liu**[2]
[1]Pattern Recognition Center, WeChat AI, Tencent Inc, China
[2]Department of Computer Science and Technology, Tsinghua University, Beijing, China
Institute for Artificial Intelligence, Tsinghua University, Beijing, China
State Key Lab on Intelligent Technology and Systems, Tsinghua University, Beijing, China
{soulcaptran,yankailin,patrickpli,withtomzhou}@tencent.com
liuzy@tsinghua.edu.cn

## Abstract

Numerical reasoning, such as addition, subtraction, sorting and counting is a critical skill in human's reading comprehension, which has not been well considered in existing machine reading comprehension (MRC) systems. To address this issue, we propose a numerical MRC model named as NumNet, which utilizes a numerically-aware graph neural network to consider the comparing information and performs numerical reasoning over numbers in the question and passage. Our system achieves an EM-score of 64.56% on the DROP dataset, outperforming all existing machine reading comprehension models by considering the numerical relations among numbers.

## 1 Introduction

Machine reading comprehension (MRC) aims to infer the answer to a question given the document. In recent years, researchers have proposed lots of MRC models (Chen et al., 2016; Dhingra et al., 2017; Cui et al., 2017; Seo et al., 2017) and these models have achieved remarkable results in various public benchmarks such as SQuAD (Rajpurkar et al., 2016) and RACE (Lai et al., 2017). The success of these models is due to two reasons: (1) Multi-layer architectures which allow these models to read the document and the question iteratively for reasoning; (2) Attention mechanisms which would enable these models to focus on the part related to the question in the document.

However, most of existing MRC models are still weak in numerical reasoning such as addition, subtraction, sorting and counting (Dua et al., 2019), which are naturally required when reading financial news, scientific articles, etc. Dua et al. (2019) proposed a numerically-aware QANet (NAQANet) model, which divides the answer generation for numerical MRC into three types: (1) extracting spans; (2) counting; (3) addition or subtraction over numbers. NAQANet makes a pioneering attempt to answer numerical questions but still does not explicitly consider numerical reasoning.

To tackle this problem, we introduce a novel model NumNet that integrates numerical reasoning into existing MRC models. A key problem to answer questions requiring numerical reasoning is how to perform numerical comparison in MRC systems, which is crucial for two common types of questions:

(1) **Numerical Comparison**: The answers of the questions can be directly obtained via performing numerical comparison, such as sorting and comparison, in the documents. For example, in Table 1, for the first question, if the MRC system knows the fact that "$49 > 47 > 36 > 31 > 22$", it could easily extract that the second longest field goal is 47-yard.

(2) **Numerical Condition**: The answers of the questions cannot be directly obtained through simple numerical comparison in the documents, but often require numerical comparison for understanding the text. For example, for the second question in Table 1, an MRC system needs to know which age group made up more than 7% of the population to count the group number.

Hence, our NumNet model considers numerical comparing information among numbers when answering numerical questions. As shown in Figure 1, NumNet first encodes both the question and passages through an encoding module consisting of convolution layers, self-attention layers and feed-forward layers as well as a passage-question attention layer. After that, we feed the question and passage representations into a numerically-aware graph neural network (NumGNN) to further

*indicates equal contribution



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2474–2484, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| Question | Passage | Answer |
|---|---|---|
| What is the second longest field goal made? | ... The Seahawks immediately trailed on a scoring rally by the Raiders with kicker *Sebastian Janikowski nailing a 31-yard field goal* ... Then in the third quarter *Janikowski made a 36-yard field goal*. Then *he made a 22-yard field goal* in the fourth quarter to put the Raiders up 16-0 ... The Seahawks would make their only score of the game with kicker *Olindo Mare hitting a 47-yard field goal*. However, they continued to trail as *Janikowski made a 49-yard field goal*, followed by RB Michael Bush making a 4-yard TD run. | 47-yard |
| How many age groups made up more than 7% of the population? | Of Saratoga Countys population in 2010, *6.3%* were between ages of 5 and 9 years, *6.7%* between 10 and 14 years, 6.5% between 15 and 19 years, *5.5%* between 20 and 24 years, *5.5%* between 25 and 29 years, *5.8%* between 30 and 34 years, *6.6%* between 35 and 39 years, *7.9%* between 40 and 44 years, *8.5%* between 45 and 49 years, *8.0%* between 50 and 54 years, *7.0%* between 55 and 59 years, *6.4%* between 60 and 64 years, and *13.7%* of age 65 years and over ... | 5 |

Table 1: Example questions from the DROP dataset which require numerical comparison. We highlight the relevant parts in the passage to infer the answer.

integrate the comparison information among numbers into their representations. Finally, we utilize the numerically-aware representation of passages to infer the answer to the question.

The experimental results on a public numerical MRC dataset DROP (Dua et al., 2019) show that our NumNet model achieves significant and consistent improvement as compared to all baseline methods by explicitly performing numerical reasoning over numbers in the question and passage. In particular, we show that our model could effectively deal with questions requiring sorting with multi-layer NumGNN. The source code of our paper is available at `https://github.com/ranqiu92/NumNet`.

## 2 Related Work

### 2.1 Machine Reading Comprehension

Machine reading comprehension (MRC) has become an important research area in NLP. In recent years, researchers have published a large number of annotated MRC datasets such as CNN/Daily Mail (Hermann et al., 2015), SQuAD (Rajpurkar et al., 2016), RACE (Lai et al., 2017), TriviaQA (Joshi et al., 2017) and so on. With the blooming of available large-scale MRC datasets, a great number of neural network-based MRC models have been proposed to answer questions for a given document including Attentive Reader (Kadlec et al., 2016), BiDAF (Seo et al., 2017), Interactive AoA Reader (Cui et al., 2017), Gated Attention Reader (Dhingra et al., 2017), R-Net (Wang et al., 2017a), DCN (Xiong et al., 2017), QANet (Yu et al., 2018), and achieve promising results in most existing public MRC datasets.

Despite the success of neural network-based MRC models, researchers began to analyze the data and rethink to what extent we have solved the problem of MRC. Some works (Chen et al., 2016; Sugawara et al., 2018; Kaushik and Lipton, 2018) classify the reasoning skills required to answer the questions into the following types: (1) Exact matching/Paraphrasing; (2) Summary; (3) Logic reasoning; (4) Utilizing external knowledge; (5) Numerical reasoning. They found that most existing MRC models are focusing on dealing with the first three types of questions. However, all these models suffer from problems when answering the questions requiring numerical reasoning. To the best of our knowledge, our work is the first one that *explicitly* incorporates numerical reasoning into the MRC system. The most relevant work to ours is NAQANet (Dua et al., 2019), which adapts the output layer of QANet (Yu et al., 2018) to support predicting answers based on counting and addition/subtraction over numbers. However, it does not consider numerical reasoning explicitly during encoding or inference.

### 2.2 Arithmetic Word Problem Solving

Recently, understanding and solving arithmetic word problems (AWP) has attracted the growing interest of NLP researchers. Hosseini et al. (2014) proposed a simple method to address arithmetic word problems, but mostly focusing on subsets of problems which only require addition and subtraction. After that, Roy and Roth (2015) proposed an algorithmic approach which could handle arithmetic word problems with multiple steps and operations. Koncel-Kedziorski et al. (2015) further

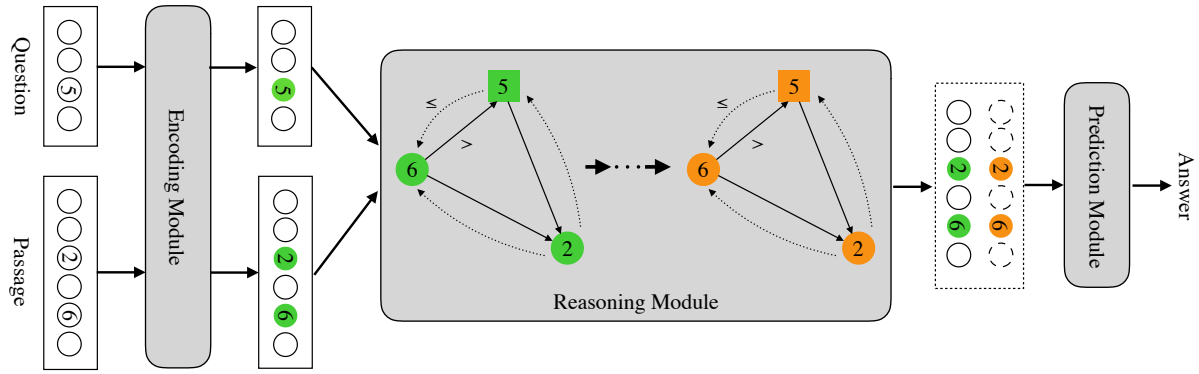


Figure 1: **The framework of our NumNet model.** Our model consists of an encoding module, a reasoning module and a prediction module. The numerical relations between numbers are encoded with the topology of the graph. For example, the edge pointing from "6" to "5" denotes "6" is greater than "5". And the reasoning module leverages a numerically-aware graph neural network to perform numerical reasoning on the graph. As numerical comparison is modeled explicitly in our model, it is more effective for answering questions requiring numerical reasoning such as addition, counting, or sorting over numbers.

formalized the AWP problem as that of generating and scoring equation trees via integer linear programming. Wang et al. (2017b) and Ling et al. (2017) proposed sequence to sequence solvers for the AWP problems, which are capable of generating unseen expressions and do not rely on sophisticated manual features. Wang et al. (2018) leveraged deep Q-network to solve the AWP problems, achieving a good balance between effectiveness and efficiency. However, all the existing AWP systems are only trained and validated on small benchmark datasets. Huang et al. (2016) found that the performance of these AWP systems sharply degrades on larger datasets. Moreover, from the perspective of NLP, MRC problems are more challenging than AWP since the passages in MRC are mostly real-world texts which require more complex skills to be understood. Above all, it is nontrivial to adapt most existing AWP models to the MRC scenario. Therefore, we focus on enhancing MRC models with numerical reasoning abilities in this work.

## 3 Methodology

In this section, we will introduce the framework of our model NumNet and provide the details of the proposed numerically-aware graph neural network (NumGNN) for numerical reasoning.

### 3.1 Framework

An overview of our model NumNet is shown in Figure 1. We compose our model with encoding module, reasoning module and prediction module. Our major contribution is the reasoning module, which leverages a NumGNN between the encoding module and prediction module to explicitly consider the numerical comparison information and perform numerical reasoning. As NAQANet has been shown effective for handling numerical MRC problem (Dua et al., 2019), we leverage it as our base model and mainly focus on the design and integration of the NumGNN in this work.

**Encoding Module** Without loss of generality, we use the encoding components of QANet and NAQANet to encode the question and passage into vector-space representations. Formally, the question $Q$ and passage $P$ are first encoded as:

$$\boldsymbol{Q} = \texttt{QANet-Emb-Enc}(Q), \quad (1)$$
$$\boldsymbol{P} = \texttt{QANet-Emb-Enc}(P), \quad (2)$$

and then the passage-aware question representation and the question-aware passage representation are computed as:

$$\bar{\boldsymbol{Q}} = \texttt{QANet-Att}(\boldsymbol{P}, \boldsymbol{Q}), \quad (3)$$
$$\bar{\boldsymbol{P}} = \texttt{QANet-Att}(\boldsymbol{Q}, \boldsymbol{P}), \quad (4)$$

where $\texttt{QANet-Emb-Enc}(\cdot)$ and $\texttt{QANet-Att}(\cdot)$ denote the "stacked embedding encoder layer" and "context-query attention layer" of QANet respectively. The former consists of convolution, self-attention and feed-forward layers. The latter is a passage-question attention layer. $\bar{\boldsymbol{Q}}$ and $\bar{\boldsymbol{P}}$ are used by the following components.

**Reasoning Module** First we build a heterogeneous directed graph $\mathcal{G} = (\boldsymbol{V}; \boldsymbol{E})$, whose nodes

($\boldsymbol{V}$) are corresponding to the numbers in the question and passage, and edges ($\boldsymbol{E}$) are used to encode numerical relationships among the numbers. The details will be explained in Sec. 3.2.

Then we perform reasoning on the graph based on a graph neural network, which can be formally denoted as:

$$\boldsymbol{M}^Q = \texttt{QANet-Mod-Enc}(\boldsymbol{W}^M \bar{\boldsymbol{Q}}), \quad (5)$$
$$\boldsymbol{M}^P = \texttt{QANet-Mod-Enc}(\boldsymbol{W}^M \bar{\boldsymbol{P}}), \quad (6)$$
$$\boldsymbol{U} = \texttt{Reasoning}(\mathcal{G}; \boldsymbol{M}^Q, \boldsymbol{M}^P), \quad (7)$$

where $\boldsymbol{W}^M$ is a shared weight matrix, $\boldsymbol{U}$ is the representations of the nodes corresponding to the numbers, $\texttt{QANet-Mod-Enc}(\cdot)$ is the "model encoder layer" defined in QANet which is similar to $\texttt{QANet-Emb-Enc}(\cdot)$, and the definition of $\texttt{Reasoning}(\cdot)$ will be given in Sec. 3.3.

Finally, as $\boldsymbol{U}$ only contains the representations of numbers, to tackle span-style answers containing non-numerical words, we concatenate $\boldsymbol{U}$ with $\boldsymbol{M}^P$ to produce numerically-aware passage representation $\boldsymbol{M}_0$. Formally,

$$\boldsymbol{M}_0^{\text{num}}[i] = \begin{cases} \boldsymbol{U}[I(i)] & \text{if } w_i^p \text{ is a number} \\ \boldsymbol{0} \end{cases},$$
$$\boldsymbol{M}_0' = \boldsymbol{W}_0[\boldsymbol{M}^P; \boldsymbol{M}^{\text{num}}] + \boldsymbol{b}_0, \quad (8)$$
$$\boldsymbol{M}_0 = \texttt{QANet-Mod-Enc}(\boldsymbol{M}_0'), \quad (9)$$

where $[\cdot;\cdot]$ denotes matrix concatenation, $\boldsymbol{W}[k]$ denotes the $k$-th column of a matrix $\boldsymbol{W}$, $\boldsymbol{0}$ is a zero vector, $I(i)$ denotes the node index corresponding to the passage word $w_i^p$ which is a number, $\boldsymbol{W}_0$ is a weight matrix, and $\boldsymbol{b}_0$ is a bias vector.

**Prediction Module** Following NAQANet (Dua et al., 2019), we divide the answers into four types and use a unique output layer to calculate the conditional answer probability $\Pr(\text{answer}|\text{type})$ for each type :

- *Passage span*: The answer is a span of the passage, and the answer probability is defined as the product of the probabilities of the start and end positions.
- *Question span*: The answer is a span of the question, and the answer probability is also defined as the product of the probabilities of the start and end positions.
- *Count*: The answer is obtained by counting, and it is treated as a multi-class classification problem over ten numbers (0-9), which covers most of the *Count* type answers in the DROP dataset.
- *Arithmetic expression*: The answer is the result of an arithmetic expression. The expression is obtained in three steps: (1) extract all numbers from the passage; (2) assign a sign (plus, minus or zero) for each number; (3) sum the signed numbers [1].

Meanwhile, an extra output layer is also used to predict the probability $\Pr(\text{type})$ of each answer type. At training time, the final answer probability is defined as the joint probability over all feasible answer types, i.e., $\sum_{\text{type}} \Pr(\text{type}) \Pr(\text{answer}|\text{type})$. Here, the answer type annotation is not required and the probability $\Pr(\text{type})$ is learnt by the model. At test time, the model first selects the most probable answer type greedily and then predicts the best answer accordingly.

Without loss of generality, we leverage the definition of the five output layers in (Dua et al., 2019), with $\boldsymbol{M_0}$ and $\boldsymbol{Q}$ as inputs. Please refer to the paper for more details due to space limitation.

**Comparison with NAQANet** The major difference between our model and NAQANet is that NAQANet does not have the reasoning module, i.e., $\boldsymbol{M}_0$ is simply set as $\boldsymbol{M}^P$. As a result, numbers are treated as common words in NAQANet except in the prediction module, thus NAQANet may struggle to learn the numerical relationships between numbers, and potentially cannot well generalize to unseen numbers. However, as discussed in Sec. 1, the numerical comparison is essential for answering questions requiring numerical reasoning. In our model, the numerical relationships are explicitly represented with the topology of the graph and a NumGNN is used to perform numerical reasoning. Therefore, our NumNet model can handle questions requiring numerical reasoning more effectively, which is verified by the experiments in Sec. 4.

### 3.2 Numerically-aware Graph Construction

We regard all numbers from the question and passage as nodes in the graph for reasoning [2] . The

[1] As few samples require multiplication/division expression in the DROP dataset, we simply adapt the module proposed (Dua et al., 2019) and leave multiplication/division expression handling as future work.

[2] As a number in the question may serve as a critical comparison condition (refer to the second example in Table 1),

set of nodes corresponding to the numbers occurring in question and passage are denoted as $\boldsymbol{V}^Q$ and $\boldsymbol{V}^P$ respectively. And we denote all the nodes as $\boldsymbol{V} = \boldsymbol{V}^Q \cup \boldsymbol{V}^P$, and the number corresponding to a node $v \in \boldsymbol{V}$ as $n(v)$.

Two sets of edges are considered in this work:

- **Greater Relation Edge ($\overrightarrow{\boldsymbol{E}}$)**: For two nodes $v_i, v_j \in \boldsymbol{V}$, a directed edge $\overrightarrow{e}_{ij} = (v_i, v_j)$ pointing from $v_i$ to $v_j$ will be added to the graph if $n(v_i) > n(v_j)$, which is denoted as solid arrow in Figure 1.

- **Lower or Equal Relation Edge ($\overleftarrow{\boldsymbol{E}}$)**: For two nodes $v_i, v_j \in \boldsymbol{V}$, a directed edge $\overleftarrow{e}_{ij} = (v_j, v_i)$ will be added to the graph if $n(v_i) \leq n(v_j)$, which is denoted as dashed arrow in Figure 1.

Theoretically, $\overrightarrow{\boldsymbol{E}}$ and $\overleftarrow{\boldsymbol{E}}$ are complement to each other . However, as a number may occur several times and represent different facts in a document, we add a distinct node for each occurrence in the graph to prevent potential ambiguity. Therefore, it is more reasonable to use both $\overrightarrow{\boldsymbol{E}}$ and $\overleftarrow{\boldsymbol{E}}$ in order to encode the equal information among nodes.

### 3.3 Numerical Reasoning

As we built the graph $\mathcal{G} = (\boldsymbol{V}, \boldsymbol{E})$, we leverage NumGNN to perform reasoning, which is corresponding to the function $\texttt{Reasoning}(\cdot)$ in Eq. 7. The reasoning process is as follows:

**Initialization** For each node $v_i^P \in \boldsymbol{V}^P$, its representation is initialized as the corresponding column vector of $\boldsymbol{M}^P$. Formally, the initial representation is $\boldsymbol{v}_i^P = \boldsymbol{M}^P[I^P(v_i^P)]$, where $I^P(v_i^P)$ denotes the word index corresponding to $v_i^P$. Similarly, the initial representation $\boldsymbol{v}_j^Q$ for a node $v_j^Q \in \boldsymbol{V}^Q$ is set as the corresponding column vector of $\boldsymbol{M}^Q$. We denote all the initial node representations as $\boldsymbol{v}^0 = \{\boldsymbol{v}_i^P\} \cup \{\boldsymbol{v}_j^Q\}$.

**One-step Reasoning** Given the graph $\mathcal{G}$ and the node representations $\boldsymbol{v}$, we use a GNN to perform reasoning in three steps:

(1) **Node Relatedness Measure**: As only a few numbers are relevant for answering a question generally, we compute a weight for each node to by-pass irrelevant numbers in reasoning. Formally, the weight for node $v_i$ is computed as:

$$\alpha_i = \text{sigmoid}(\boldsymbol{W}_v \boldsymbol{v}[i] + b_v), \tag{10}$$

we also add nodes for them in the graph.

where $\boldsymbol{W}_v$ is a weight matrix, and $b_v$ is a bias.

(2) **Message Propagation**: As the role a number plays in reasoning is not only decided by itself, but also related to the context, we propagate messages from each node to its neighbors to help to perform reasoning. As numbers in question and passage may play different roles in reasoning and edges corresponding to different numerical relations should be distinguished, we use relation-specific transform matrices in the message propagation. Formally, we define the following propagation function for calculating the forward-pass update of a node:

$$\widetilde{\boldsymbol{v}}_i' = \frac{1}{|\mathcal{N}_i|} \left( \sum_{j \in \mathcal{N}_i} \alpha_j \boldsymbol{W}^{\mathtt{r}_{ji}} \boldsymbol{v}[j] \right), \tag{11}$$

where $\widetilde{\boldsymbol{v}}_i'$ is the message representation of node $v_i$, $\mathtt{r}_{ji}$ is the relation assigned to edge $e_{ji}$, $\boldsymbol{W}^{\mathtt{r}_{ji}}$ are relation-specific transform matrices, and $\mathcal{N}_i = \{j | (v_j, v_i) \in \boldsymbol{E}\}$ is the neighbors of node $v_i$.

For each edge $e_{ji}$, $\mathtt{r}_{ji}$ is determined by the following two attributes:

- Number relation: $>$ or $\leq$;

- Node types: the two nodes of the edge corresponding to two numbers that: (1) both from the question (q-q); (2) both from the passage (p-p); (3) from the question and the passage respectively (q-p); (4) from the passage and the question respectively (p-q).

Formally, $\mathtt{r}_{ij} \in \{>, \leq\} \times \{\text{q-q}, \text{p-p}, \text{q-p}, \text{p-q}\}$.

(3) **Node Representation Update**: As the message representation obtained in the previous step only contains information from the neighbors, it needs to be fused with the node representation to combine with the information carried by the node itself, which is performed as:

$$\boldsymbol{v}_i' = \text{ReLU}(\boldsymbol{W}_f \boldsymbol{v}_i + \widetilde{\boldsymbol{v}}_i' + \boldsymbol{b}_f), \tag{12}$$

where $\boldsymbol{W}_f$ is a weight matrix, and $\boldsymbol{b}_f$ is a bias vector.

We denote the entire one-step reasoning process (Eq. 10-12) as a single function

$$\boldsymbol{v}' = \texttt{Reasoning-Step}(\mathcal{G}, \boldsymbol{v}). \tag{13}$$

As the graph $\mathcal{G}$ constructed in Sec. 3.2 has encoded the numerical relations via its topology, the reasoning process is numerically-aware.

**Multi-step Reasoning** By single-step reasoning, we can only infer relations between adjacent nodes. However, relations between multiple nodes may be required for certain tasks, e.g., sorting. Therefore, it is essential to perform multi-step reasoning, which can be done as follows

$$\boldsymbol{v}^t = \texttt{Reasoning-Step}(\boldsymbol{v}^{t-1}), \qquad (14)$$

where $t \geq 1$. Suppose we perform $K$ steps of reasoning, $\boldsymbol{v}^K$ is used as $\boldsymbol{U}$ in Eq. 7.

## 4 Experiments

### 4.1 Dataset and Evaluation Metrics

We evaluate our proposed model on DROP dataset (Dua et al., 2019), which is a public numerical MRC dataset. The DROP dataset is constructed by crowd-sourcing, which asks the annotators to generate question-answer pairs according to the given Wikipedia passages, which require numerical reasoning such as addition, counting, or sorting over numbers in the passages. There are $77,409$ training samples, $9,536$ development samples and $9,622$ testing samples in the dataset.

In this paper, we adopt two metrics including Exact Match (EM) and numerically-focused F1 scores to evaluate our model following Dua et al. (2019). The numerically-focused F1 is set to be $0$ when the predicted answer is mismatched for those questions with the numeric golden answer.

### 4.2 Baselines

For comparison, we select several public models as baselines including **semantic parsing models**:

- Syn Dep (Dua et al., 2019), the neural semantic parsing model (KDG) (Krishnamurthy et al., 2017) with Stanford dependencies based sentence representations;
- OpenIE (Dua et al., 2019), KDG with open information extraction based sentence representations;
- SRL (Dua et al., 2019), KDG with semantic role labeling based sentence representations;

and **traditional MRC models**:

- BiDAF (Seo et al., 2017), an MRC model which utilizes a bi-directional attention flow network to encode the question and passage;
- QANet (Yu et al., 2018), which utilizes convolutions and self-attentions as the building blocks of encoders to represent the question and passage;
- BERT (Devlin et al., 2019), a pre-trained bidirectional Transformer-based language model which achieves state-of-the-art performance on lots of public MRC datasets recently;

and **numerical MRC models**:

- NAQANet (Dua et al., 2019), a numerical version of QANet model.
- NAQANet+, an enhanced version of NAQANet implemented by ourselves, which further considers real number (e.g. "2.5"), richer arithmetic expression, data augmentation, etc. The enhancements are also used in our NumNet model and the details are given in the supplemental material.

### 4.3 Experimental Settings

In this paper, we tune our model on the development set and use a grid search to determine the optimal parameters. The dimensions of all the representations (e.g., $\boldsymbol{Q}$, $\boldsymbol{P}$, $\boldsymbol{M}^Q$, $\boldsymbol{M}^P$, $\boldsymbol{U}$, $\boldsymbol{M}'_0$, $\boldsymbol{M}_0$ and $\boldsymbol{v}$) are set to $128$. If not specified, the reasoning step $K$ is set to 3. Since other parameters have little effect on the results, we simply follow the settings used in (Dua et al., 2019).

We use the Adam optimizer (Kingma and Ba, 2015) with $\beta_1 = 0.8$, $\beta_2 = 0.999$, $\epsilon = 10^{-7}$ to minimize the objective function. The learning rate is $5 \times 10^{-4}$, L2 weight decay $\lambda$ is $10^{-7}$ and the maximum norm value of gradient clipping is $5$. We also apply exponential moving average with a decay rate $0.9999$ on all trainable variables. The model is trained with a batch size of $16$ for $40$ epochs. Passages and questions are trimmed to $400$ and $50$ tokens respectively during training, and trimmed to $1,000$ and $100$ tokens respectively during prediction [3].

### 4.4 Overall Results

The performance of our NumNet model and other baselines on DROP dataset are shown in Table 2. From the results, we can observe that:

(1) Our NumNet model achieves better results on both the development and testing sets on DROP dataset as compared to semantic parsing-based models, traditional MRC models and even numerical MRC models NAQANet and NAQANet+. The reason is that our NumNet model can make full use of the numerical comparison information over

[3] Trimming passages/questions introduces little impact because it only affects about 8% of the samples.

| Method | Dev | | Test | |
|---|---|---|---|---|
| | EM | F1 | EM | F1 |
| **Semantic Parsing** | | | | |
| Syn Dep | 9.38 | 11.64 | 8.51 | 10.84 |
| OpenIE | 8.80 | 11.31 | 8.53 | 10.77 |
| SRL | 9.28 | 11.72 | 8.98 | 11.45 |
| **Traditional MRC** | | | | |
| BiDAF | 26.06 | 28.85 | 24.75 | 27.49 |
| QANet | 27.50 | 30.44 | 25.50 | 28.36 |
| BERT | 30.10 | 33.36 | 29.45 | 32.70 |
| **Numerical MRC** | | | | |
| NAQANet | 46.20 | 49.24 | 44.07 | 47.01 |
| NAQANet+ | 61.47 | 64.85 | 60.82 | 64.29 |
| **NumNet** | **64.92** | **68.31** | **64.56** | **67.97** |
| **Human Performance** | - | - | 94.09 | 96.42 |

Table 2: Overall results on the development and test set. The evaluation metrics are calculated as the maximum over a golden answer set. All the results except "NAQANet+" and "NumNet" are obtained from (Dua et al., 2019).

numbers in both question and passage via the proposed NumGNN module.

(2) Our implemented NAQANet+ has a much better performance compared to the original version of NAQANet. It verifies the effectiveness of our proposed enhancements for baseline.

### 4.5 Effect of GNN Structure

In this part, we investigate the effect of different GNN structures on the DROP development set. The results are shown in Table 3. The "Comparison", "Number" and "ALL" are corresponding to the comparing question subset [4], the number-type answer subset, and the entire development set, respectively [5]. If we replace the proposed numerically-aware graph (Sec. 3.2) with a fully connected graph, our model fallbacks to a traditional GNN, denoted as "GNN" in the table. Moreover, "- question num" denotes the numbers in the question is not included in the graph, and "- $\leq$ type edge" and "- $>$ type edge" denote edges of $\leq$ and $>$ types are not adopted respectively.

[4]We find that many comparing questions in the DROP dataset are biased, of which the answers are the former candidates in the questions. Hence, we employ crowdsourced workers to identify and rewrite all comparing questions to construct an enhanced development set. Specially, for those comparing questions containing answer candidates, we also ask the crowdsourced workers to swap the candidates manually to enlarge the dataset.

[5]Note that the "ALL" result is not the average of "Comparison" and "Number". It is the performance on the entire development set which also includes questions of selection type, coreference resolution type, etc.

| Method | Comparison | | Number | | ALL | |
|---|---|---|---|---|---|---|
| | EM | F1 | EM | F1 | EM | F1 |
| GNN | 69.86 | 75.91 | 67.77 | 67.78 | 61.90 | 65.16 |
| NumGNN | 74.53 | 80.36 | 69.74 | 69.75 | 64.54 | 68.02 |
| - question num | 74.84 | 80.24 | 68.42 | 68.43 | 63.78 | 67.17 |
| - $\leq$ type edge | 74.89 | 80.51 | 68.48 | 68.50 | 63.66 | 67.06 |
| - $>$ type edge | 74.86 | 80.19 | 68.77 | 68.78 | 63.64 | 66.96 |

Table 3: Performance with different GNN structure. "Comparison", "Number" and "ALL" denote the comparing question subset, the number-type answer subset, and the entire development set, respectively.

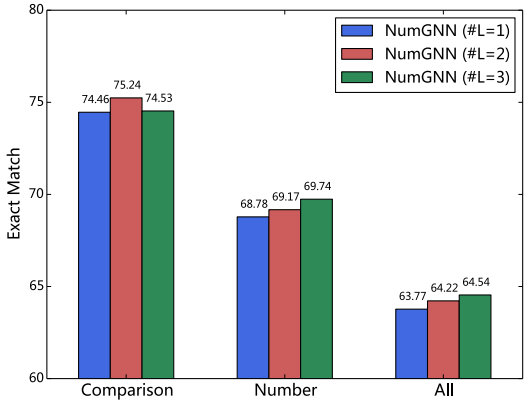


Figure 2: Effect of GNN layer numbers (# L).

As shown in Table 3, our proposed NumGNN leads to statistically significant improvements compared to traditional GNN on both EM and F1 scores especially for comparing questions. It indicates that considering the comparing information over numbers could effectively help the numerical reasoning for comparing questions. Moreover, we find that the numbers in the question are often related to the numerical reasoning for answering the question, thus considering numbers in questions in NumGNN achieves better performance. And the results also justify that encoding "greater relation" and "lower or equal relation" simultaneously in the graph also benefits our model.

### 4.6 Effect of GNN Layer Number

The number of NumGNN layers represents the numerical reasoning ability of our models. A $K$-layer version has the ability for $K$-step numerical inference. In this part, we additionally perform experiments to understand the values of the numbers of NumGNN layers. From Figure 2, we could observe that:

(1) The 2-layer version of NumNet achieves the best performance for the comparing questions. From careful analysis, we find that most compar-

| Question & Answer | Passage | NAQANet+ | NumNet |
|---|---|---|---|
| **Q:** Which age group is larger: under the age of 18 or 18 and 24?<br><br>**A:** 18 and 24 | The median age in the city was 22.1 years. *10.1%* of residents were under the age of 18; *56.2%* were between the ages of 18 and 24; 16.1% were from 25 to 44; 10.5% were from 45 to 64; and 7% were 65 years of age or older. The gender makeup of the city was 64.3% male and 35.7% female. | under the age of 18 | 18 and 24 |
| **Q**: How many more yards was Longwell's longest field goal over his second longest one?<br><br>**A:** 26-22=4 | ... The Vikings would draw first blood with a *26-yard field goal* by kicker Ryan Longwell. In the second quarter, Carolina got a field goal with opposing kicker John Kasay. The Vikings would respond with another Longwell field goal (*a 22-yard FG*) ... In OT, Longwell booted the game-winning *19-yard field goal* to give Minnesota the win. It was the first time in Vikings history that a coach ... | 26-19 = 7 | 26-22 = 4 |

Table 4: Cases from the DROP dataset. We demonstrate the predictions of NAQANet+ and our NumNet model. Note that the two models only output the arithmetic expressions but we also provide their results for clarity.

| Question | Passage | Answer | NumNet |
|---|---|---|---|
| Which ancestral groups are at least 10%? | As of the census of 2000, there were 7,791 people, 3,155 households, and 2,240 families residing in the county. ... 33.7% were of *Germans*, 13.9% *Swedish* people, 10.1% *Irish* people, 8.8% United States, 7.0% English people and 5.4% Danish people ancestry ... | German; Swedish; Irish | Irish |
| Were more people 40 and older or 19 and younger? | Of Saratoga Countys population in 2010, *6.3%* were between ages of 5 and 9 years, *6.7%* between 10 and 14 years, *6.5%* between 15 and 19 years, ... , *7.9%* between 40 and 44 years, *8.5%* between 45 and 49 years, *8.0%* between 50 and 54 years, *7.0%* between 55 and 59 years, *6.4%* between 60 and 64 years, and *13.7%* of age 65 years and over ... | 40 and older | 19 and younger |

Table 5: Typical error examples. Row 1: the answer is multiple nonadjacent spans; Row 2: Intermediate numbers are involved in reasoning.

ing questions only require at most 2-step reasoning (e.g., "*Who was the second oldest player in the MLB, Clemens or Franco?*"), and therefore the 3-layer version of NumNet is more complex but brings no gains for these questions.

(2) The performance of our NumNet model on the overall development set is improved consistently as the number of GNN layers increases. The reason is that some of the numerical questions require reasoning over many numbers in the passage, which could benefit from the multi-step reasoning ability of multi-layer GNN. However, further investigation shows that the performance gain is not stable when $K \geq 4$. We believe it is due to the intrinsic over smoothing problem of GNNs (Li et al., 2018).

### 4.7 Case Study

We further give some examples to show why incorporating comparing information over numbers in the passage could help numerical reasoning in MRC in Table 4. For the first case, we observe that NAQANet+ gives a wrong prediction, and we find that NAQANet+ will give the same prediction for the question "*Which age group is smaller: under the age of 18 or 18 and 24?*". The reason is that NAQANet+ cannot distinguish which one is larger for $10.1\%$ and $56.2\%$. For the second case, NAQANet+ cannot recognize the second longest field goal is 22-yard and also gives a wrong prediction. For these two cases, our NumNet model could give the correct answer through the numeric reasoning, which indicates the effectiveness of our NumNet model.

### 4.8 Error Analysis

To investigate how well our NumNet model handles sorting/comparison questions and better understand the remaining challenges, we perform an error analysis on a random sample of NumNet predictions. We find that:

(1) Our NumNet model can answer about 76% of sorting/comparison questions correctly, which indicates that our NumNet model has achieved numerical reasoning ability to some extend.

(2) Among the incorrectly answered sorting/comparison questions, the most ones (26%) are those whose golden answers are multiple nonadjacent spans (row 1 in Table 5), and the second most ones (19%) are those involving comparison with an intermediate number that does not literally occur in the document/question but has to be de-

rived from counting or arithmetic operation (row 1 in Table 5).

### 4.9 Discussion

By combining the numerically-aware graph and the NumGNN together, our NumNet model achieves the numerical reasoning ability. On one hand, the numerically-aware graph encodes numbers as nodes and relationships between them as the edges, which is required for numerical comparison. On the other hand, through one-step reasoning, our NumGNN could perform comparison and identify the numerical condition. After multiple-step reasoning, our NumGNN could further perform sorting.

However, since the numerically-aware graph is pre-defined, our NumNet is not applicable to the case where an intermediate number has to be derived (e.g., from arithmetic operation) in the reasoning process, which is a major limitation of our model.

## 5 Conclusion and Future Work

Numerical reasoning skills such as addition, subtraction, sorting and counting are naturally required by machine reading comprehension (MRC) problems in practice. Nevertheless, these skills are not taken into account explicitly for most existing MRC models. In this work, we propose a numerical MRC model named NumNet which performs explicit numerical reasoning while reading the passages. To be specific, NumNet encodes the numerical relations among numbers in the question and passage into a graph as its topology, and leverages a numerically-aware graph neural network to perform numerical reasoning on the graph. Our NumNet model outperforms strong baselines with a large margin on the DROP dataset.

In the future, we will explore the following directions: (1)As we use a pre-defined reasoning graph in our model, it is incapable of handling reasoning process which involves intermediate numbers that not presented in the graph. How to incorporate dynamic graph into our model is an interesting problem. (2) Compared with methods proposed for arithmetic word problems (AWPs), our model has better natural language understanding ability. However, the methods for AWPs can handle much richer arithmetic expressions. Therefore, how to combine both of their abilities to develop a more powerful numerical MRC model is an interesting future direction. (3) Symbolic reasoning plays a crucial role in human reading comprehension. Our work integrates numerical reasoning, which is a special case of symbolic reasoning, into traditional MRC systems. How to incorporate more sophisticated symbolic reasoning abilities into MRC systems is also a valuable future direction.

## Acknowledgments

We would like to thank all anonymous reviewers for their insightful comments, and thank Yan Zhang for her help on improving the presentation of Figure 1.

## References

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the CNN/Daily Mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2358–2367, Berlin, Germany.

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. 2017. Attention-over-attention neural networks for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 593–602, Vancouver, Canada.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota.

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William Cohen, and Ruslan Salakhutdinov. 2017. Gated-attention readers for text comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1832–1846, Vancouver, Canada.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2368–2378, Minneapolis, Minnesota.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Su-

leyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proceedings of Advances in Neural Information Processing Systems*, pages 1693–1701.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–533, Doha, Qatar.

Danqing Huang, Shuming Shi, Chin-Yew Lin, Jian Yin, and Wei-Ying Ma. 2016. How well do computers solve math word problems? Large-scale dataset construction and evaluation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 887–896, Berlin, Germany.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada.

Rudolf Kadlec, Martin Schmid, Ondřej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 908–918, Berlin, Germany.

Divyansh Kaushik and Zachary C. Lipton. 2018. How much reading does reading comprehension require? A critical investigation of popular benchmarks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5010–5015, Brussels, Belgium.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR 2015 : International Conference on Learning Representations 2015*.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1516–1526, Copenhagen, Denmark.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 3538–3545.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *arXiv preprint arXiv:1705.04146*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752, Lisbon, Portugal.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *Proceedings of ICLR 2017: the 5th International Conference on Learning Representations*.

Saku Sugawara, Kentaro Inui, Satoshi Sekine, and Akiko Aizawa. 2018. What makes reading comprehension questions easier? In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4208–4219, Brussels, Belgium.

Lei Wang, Dongxiang Zhang, Lianli Gao, Jingkuan Song, Long Guo, and Heng Tao Shen. 2018. MathDQN: Solving arithmetic word problems via deep reinforcement learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017a. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 189–198, Vancouver, Canada.

Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017b. Deep neural solver for math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 845–854, Copenhagen, Denmark.

Caiming Xiong, Victor Zhong, and Richard Socher. 2017. Dynamic coattention networks for question answering. In *Proceedings of ICLR 2017: the 5th International Conference on Learning Representations*.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. QANet: Combining local convolution with global self-attention for reading comprehension. In *Proceedings of ICLR 2018: the 6th International Conference on Learning Representations*.

# Unicoder: A Universal Language Encoder by Pre-training with Multiple Cross-lingual Tasks

**Haoyang Huang** [§] **Yaobo Liang** [§] **Nan Duan**[§] **Ming Gong**[†] **Linjun Shou**[†]
**Daxin Jiang**[†] **Ming Zhou**[§]
[§] Microsoft Research Asia, Beijing, China
[†] STCA NLP Group, Microsoft, Beijing, China
{haohua,yalia,nanduan,migon,lisho,djiang,mingzhou}@microsoft.com

## Abstract

We present Unicoder, a universal language encoder that is insensitive to different languages. Given an arbitrary NLP task, a model can be trained with Unicoder using training data in one language and directly applied to inputs of the same task in other languages. Comparing to similar efforts such as Multilingual BERT (Devlin et al., 2018) and XLM (Lample and Conneau, 2019), three new cross-lingual pre-training tasks are proposed, including cross-lingual word recovery, cross-lingual paraphrase classification and cross-lingual masked language model. These tasks help Unicoder learn the mappings among different languages from more perspectives. We also find that doing fine-tuning on multiple languages together can bring further improvement. Experiments are performed on two tasks: cross-lingual natural language inference (XNLI) and cross-lingual question answering (XQA), where XLM is our baseline. On XNLI, 1.8% averaged accuracy improvement (on 15 languages) is obtained. On XQA, which is a new cross-lingual dataset built by us, 5.5% averaged accuracy improvement (on French and German) is obtained.

## Introduction

Data annotation is expensive and time-consuming for most of NLP tasks. Recently, pre-trained models, such as ELMo (Peters et al., 2018), BERT (Devlin et al., 2018) and GPT (Radford et al., 2018), have shown strong capabilities of transferring knowledge learned from large-scale text corpus to specific NLP tasks with limited or no training data. But they still cannot handle tasks when training and test instances are in different languages.

Motivated by this issue, some efforts have been made, such as Multilingual BERT (Devlin et al., 2018) and XLM (Lample and Conneau, 2019), for cross-lingual tasks. Multilingual BERT trains a BERT model based on multilingual Wikipedia, which covers 104 languages. As its vocabulary contains tokens from all languages, Multilingual BERT can be used to cross-lingual tasks directly. XLM further improves Multilingual BERT by introducing a translation language model (TLM). TLM takes a concatenation of a bilingual sentence pair as input and performs masked language model based on it. By doing this, it learns the mappings among different languages and performs good on the XNLI dataset.

However, XLM only uses a single cross-lingual task during pre-training. At the same time, Liu et al. (2019) has shown that multi-task learning can further improve a BERT-style pre-trained model. So we think more cross-lingual tasks could further improve the resulting pre-trained model for cross-lingual tasks. To verify this, we propose Unicoder, a universal language encoder that is insensitive to different languages and pre-trained based on 5 pre-training tasks. Besides masked language model and translation language model, 3 new cross-lingual pre-training tasks are used in the pre-training procedure, including cross-lingual word recovery, cross-lingual paraphrase classification and cross-lingual masked language model. Cross-lingual word recovery leverage attention matrix between bilingual sentence pair to learn the cross-lingual word alignment relation. Cross-lingual paraphrase classification takes two sentences from different languages and classify whether they have same meaning. This task could learn the cross-lingual sentence alignment relation. Inspired by the successful of monolingual pre-training on long text (Radford et al., 2018; Devlin et al., 2018), we propose cross-lingual masked language model whose input is document written by multiple languages. We also find that doing fine-tuning on multiple languages together can



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2485–2494, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

bring further improvement. For the languages without training data, we use machine translated data from rich-resource languages.

Experiments are performed on cross-lingual natural language inference (XNLI) and cross-lingual question answering (XQA), where both Multilingual BERT and XLM are considered as our baselines. On XNLI, 1.8% averaged accuracy improvement (on 15 languages) is obtained. On XQA, which is a new cross-lingual dataset built by us, 5.5% averaged accuracy improvement (on French and German) is obtained.

In short, our contributions are 4-fold. First, 3 new cross-lingual pre-training tasks are proposed, which can help to learn a better language-independent encoder. Second, a cross-lingual question answering (XQA) dataset is built, which can be used as a new cross-lingual benchmark dataset. Third, we verify that by fine-tuning multiple languages together, significant improvements can be obtained. Fourth, on the XNLI dataset, new state-of-the-art results are achieved.

## Related work

**Monolingual Pre-training** Recently, pretraining an encoder by language model (Radford et al., 2018; Peters et al., 2018; Devlin et al., 2018) and machine translation (McCann et al., 2017) have shown significant improvement on various natural language understanding (NLU) tasks, like tasks in GLUE (Wang et al., 2018). The application scheme is to fine-tune the pre-trained encoder on single sentence classification task or sequential labeling task. If the tasks have multiple inputs, just concatenate them into one sentence. This approach enables one model to be generalized to different language understanding tasks. Our approach also is contextual pre-training so it could been applied to various NLU tasks.

**Cross-lingual Pre-training** Cross-lingual Pretraining is a kind of transfer learning with different source and target domain (Pan and Yang, 2010). A high-quality cross-lingual representation space is assumed to effectively perform the cross-lingual transfer. Mikolov et al. (2013) has been applied small dictionaries to align word representations from different languages and it is sufficient to align different languages with orthogonal transformation (Xing et al., 2015), even without parallel data (Lample et al., 2018). Following the line of previous work, the word alignment also could be applied to multiple languages (Ammar et al., 2016). Artetxe and Schwenk (2018) use multilingual machine translation to train a multilingual sentence encoder and use this fixed sentence embedding to classify XNLI. We take these ideas one step further by producing a pre-trained encoder instead of word embedding or sentence embedding.

Our work is based on two recent pre-trained cross-lingual encoders: multilingual BERT [1] (Devlin et al., 2018) and XLM (Lample and Conneau, 2019). Multilingual BERT trains masked language model (MLM) with sharing vocabulary and weight for all 104 languages. But each training sample is monolingual document. Keeping the same setting, XLM proposed a new task TLM, which uses a concatenation of the parallel sentences into one sample for masked language modeling. Besides these two tasks, we proposed three new cross-lingual pre-training tasks for building a better language-independent encoder.

## Approach

This section will describe details of Unicoder, including tasks used in the pre-training procedure and its fine-tuning strategy.

### Model Structure

Unicoder follows the network structure of XLM (Lample and Conneau, 2019). A shared vocabulary is constructed by running the Byte Pair Encoding (BPE) algorithm (Sennrich et al., 2016) on corpus of all languages. We also down sample the rich-resource languages corpus, to prevent words of target languages from being split too much at the character level.

### Pre-training Tasks in Unicoder

Both masked language model and translation language model are used in Unicoder by default, as they have shown strong performance in XLM.

Motivated by Liu et al. (2019), which shows that pre-trained models can be further improved by involving more tasks in pre-training, we introduce three new cross-lingual tasks in Unicoder. All training data for these three tasks are acquired from the existing large-scale high-quality machine translation corpus.

**Cross-lingual Word Recovery** Similar to translation language model, this task also aims to let the

[1] https://github.com/google-research/bert/blob/master/multilingual.md

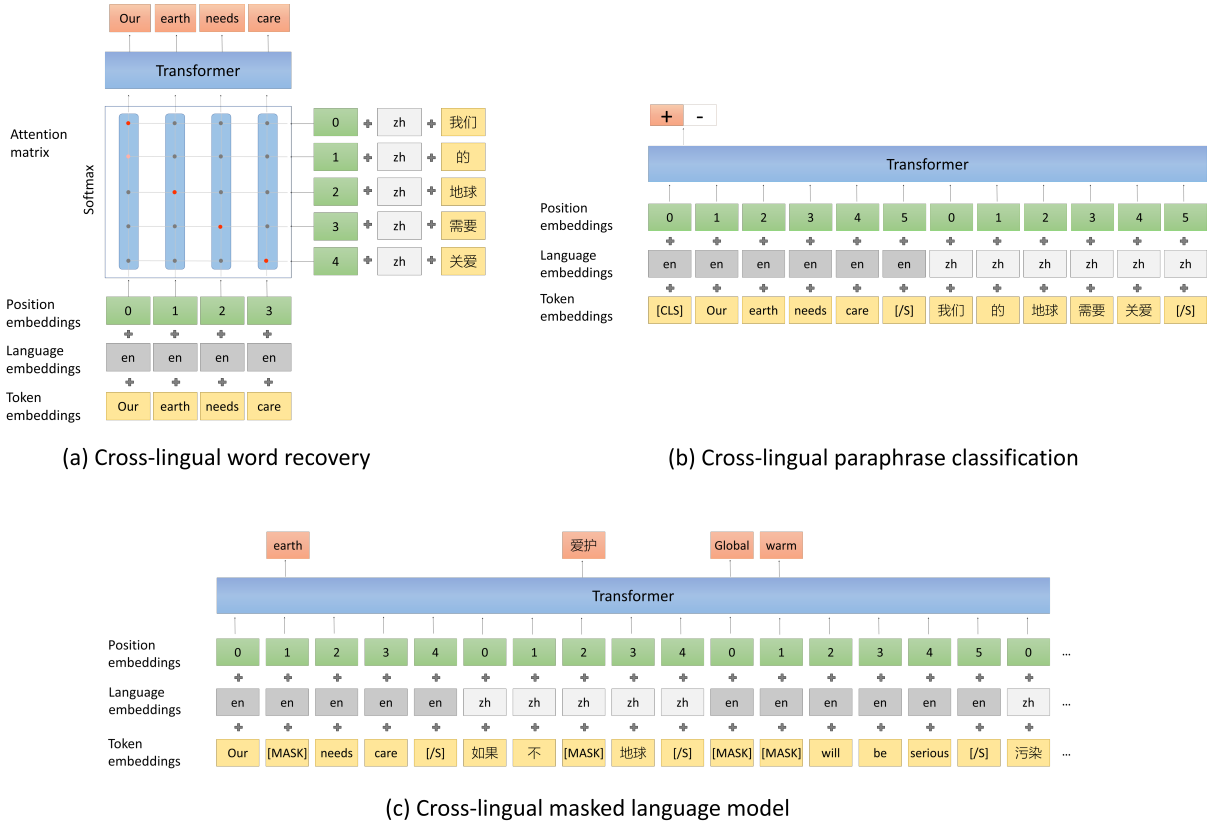


(a) Cross-lingual word recovery

(b) Cross-lingual paraphrase classification

(c) Cross-lingual masked language model

Figure 1: Unicoder consists of three cross-lingual pre-training tasks: (a) The cross-lingual word recovery model is to learn word relation from different languages (b) The cross-lingual paraphrase classification is to classify whether two sentences from different languages are paraphrase. (c) The cross-lingual masked language model is to train masked language model with cross-lingual document.

pre-trained model learn the underlying word alignments between two languages. It is mainly motivated by the attention matrix used in the neural machine translation Bahdanau et al. (2014) task.

Formally, given a bilingual sentence pair $(X, Y)$, where $X = (x_1, x_2, ..., x_m)$ is a sentence with $m$ words from language $s$, $Y = (y_1, y_2, ..., y_n)$ is a sentence with $n$ words from language $t$, this task first represents each $x_i$ as $x_i^t \in R^h$ by all word embeddings of $Y$:

$$x_i^t = \sum_{j=1}^{n} softmax(A_{ij}) y_j^t \quad (1)$$

where $x_i^s \in R^h$ and $y_j^t \in R^h$ denote the word embeddings of $x_i$ and $y_j$ respevtively, $h$ denotes the word embedding dimension, $A \in R^{m \times n}$ is an attention matrix calculated by:

$$A_{ij} = W[x_i^s, y_j^t, x_i^s \odot y_j^t] \quad (2)$$

$W \in R^{3*h}$ is a trainable weight and $\odot$ is element-wise multiplication. Then, Unicoder takes $X^t = (x_1^t, x_2^t, ..., x_n^t)$ as input, and tries to predict the original word sequence $X$.

Similar to translation language model in XLM, this task is based on the bilingual sentence pairs as well. However, as it doesn't use the original words as input, we can train this task by recovering all words at the same time. The model structure of this task is illustrated in Figure 1.a.

**Cross-lingual Paraphrase Classification** This task takes two sentences from different languages as input and classifies whether they are with the same meaning. Like the next sentence prediction task in BERT, we concatenate two sentences as a sequence and input it to Unicoder. The representation of the first token in the final layer will be used for the paraphrase classification task. This procedure is illustrated in Figure 1.b.

We created the cross-lingual paraphrase classification dataset from machine translation dataset. Each bilingual sentence pair $(X, Y)$ servers as a positive sample. For negative samples, the most straight forward method is to replace $Y$ to a ran-

dom sampled sentence from target language. But this will make the classification task too easy. So we introduce the hard negative samples followed Guo et al. (2018). First, we train a light-weight paraphrase model with random negative samples. Then we use this model to select sentence with high similarity score to $X$ but doesn't equal to $Y$ as hard negative samples. We choose DAN (Iyyer et al., 2015) as the light model. We create positive and negative samples in 1:1.

**Cross-lingual Masked Language Model** Previous successful pre-training language model (Devlin et al., 2018; Radford et al., 2018) is conducted on document-level corpus rather than sentence-level corpus. The language model perplexity on document also is much lower than sentence (Peters et al., 2018). So we propose cross-lingual masked language model, whose input is come from cross-lingual document.

Cross-lingual document is a sequence of sentences, and the sentences are written with different languages. In most case, people won't write cross-lingual document. But we found that a large proportion of aligned sentence pairs in machine translation are extracted from parallel documents, such as MultiUN corpus and OpenSubtitles corpus. In other words, these MT corpus are document-level corpus in which each sentence and its translation is well aligned. We construct cross-lingual document by replacing the sentences with even index to its translation as illustrated in Figure 1.c. We truncate the cross-lingual document by 256 sequence length and feed it to Unicoder for masked language modeling.

### Multi-language Fine-tuning

A typical setting of cross-lingual language understanding is only one language has training data, but the test is conducted on other languages. We denote the language has training data as source language, and other languages as target languages. A scalable way (Conneau et al., 2018) to address this problem is through Cross-lingual TEST, in which a pre-trained encoder is trained on data in source language and directly evaluated on data in target languages.

There are two other machine translation methods that make training and test belong to the same language. TRANSLATE-TRAIN translates the source language training data to a target language and fine-tunes on this pseudo training data. TRANSLATE-TEST fine-tunes on source language training data, but translates the target language test data to source language and test on it.

Inspired by multi-task learning (Liu et al., 2018, 2019) for improving pre-trained model, we propose a new fine-tuning strategy Multi-language Fine-tuning. We propose to fine-tune on both the source language training data and pseudo target language training data. If there are multiple target languages, we will fine-tune on all of them at same time.

Different languages may have totally different vocabulary and syntax. But our experiments show that in most cases, joint fine-tuning multiple languages could bring huge improvement. Only in just a few cases, this may harm the performance.

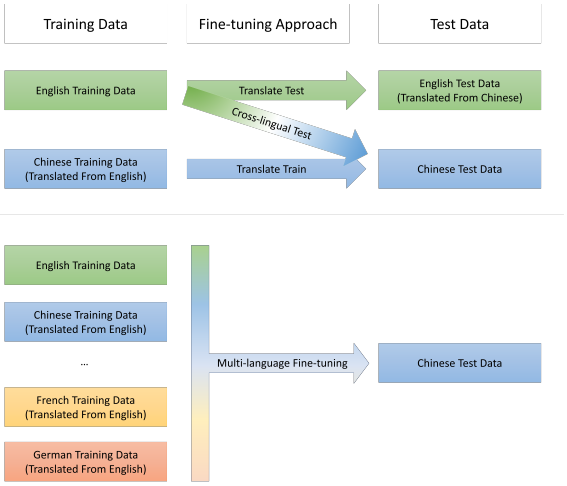


Figure 2: Currently cross-lingual fine-tuning has three baseline approaches, they could be defined based on their training data and test data. Suppose we target to test on Chinese data, Translate-train is to train on Chinese training data which is translated from English and test on Chinese test data; Translate-Test is to train on English training data and test on English test data which is translated from Chinese; Cross-lingual test is to train in English training data and test on Chinese test data. Multi-language fine-tuning is to train on English training data and multiple other languages training data which are translated from English, then test on Chinese Test data.

## Experiment

In this section, we describe the data processing and training details. Then we compare the Unicoder with the current state of the art approaches on two tasks: XNLI and XQA.

### Data Processing

Our model is pre-trained on 15 languages, including English(en), French(fr), Spanish(es), German(de), Greek(el), Bulgarian(bg), Russian(ru), Turkish(tr), Arabic(ar), Vietnamese(vi), Thai(th), Chinese(zh), Hindi(hi), Swahili(sw) and Urdu(ur). For MLM, we use the Wikipedia from these languages. The other four tasks need MT dataset. We use same MT dataset as Lample and Conneau (2019) which are collected from MultiUN (Eisele and Chen, 2010), IIT Bombay corpus (Kunchukuttan et al., 2017), OpenSubtitles 2018, EUbookshop corpus and GlobalVoices. In the MT corpus, 13 of 14 languages (except IIT Bombay corpus) are from parallel document and could be used to train cross-lingual document language model. The number of data we used is reported at table 1.

For tokenization, we follows the line of Koehn et al. (2007); Chang et al. (2008) for each language. We use byte-pair encoding (BPE) to process the corpus and build vocabulary.

| language | mono-lingual | bi-lingual |
|---|---|---|
| ar | 3.8M | 9.8M |
| bg | 1.5M | 0.6M |
| de | 17.4M | 9.3M |
| el | 1.3M | 4.0M |
| en | 43.2M | - |
| es | 11.3M | 11.4M |
| fr | 15.5M | 13.2M |
| hi | 0.6M | 1.6M |
| ru | 12.6M | 11.7M |
| sw | 0.2M | 0.2M |
| th | 0.8M | 3.3M |
| tr | 1.8M | 0.5M |
| ur | 0.5M | 0.7M |
| vi | 3.8M | 3.5M |
| zh | 5.5M | 9.6M |

Table 1: Sentence number we used in pre-training.

### Training Details

**Model Structure** Our Unicoder is a 12-layer transformer with 1024 hidden units, 16 heads, GELU activation (Hendrycks and Gimpel, 2017). we set dropout to 0.1. The vocabulary size is 95,000.

**Pre-training details** To reduce pre-training time, we initialize our model from XLM (Lample and Conneau, 2019). We pretrain Unicoder with five tasks including MLM\TLM and our three cross-lingual tasks. In each step, we iteratively train these five tasks. A batch for these tasks is available in 15 languages, and we sample several languages with equal probability. And we use batch size 512 by gradient accumulation. We train our model with the Adam optimizer (Kingma and Ba, 2015), and learning rate starts from $1e-5$ with invert square root decay (Vaswani et al., 2017). We run our pretraining experiments on a single server with 8 V100 GPUs and use FP16 to save the memory.

The max sequence length of MLM and cross-lingual language model is 256. For the other three tasks with two sentences as input, we set the max sequence length to 128 so the sum of them is 256.

**Fine-tuning details** For fine-tuning stage, we use same optimizer and learning rate as pre-training. We set the batch size to 32.

### Experimental evaluation

**XNLI: Cross-lingual Natural Language Inference** Natural Language Inference(NLI) takes two sentences as input and determines whether one entails the other, contradicts it or neither (neutral). XNLI is NLI defined on 15 languages. Each language contains 5000 human annotated development and test set. Only English has training data, which is a crowd-sourced collection of 433k sentence pairs from MultiNLI (Williams et al., 2018). The performance is evaluated by classification accuracy.

We report the results of XNLI in Table 2, by comparing our Unicoder model with four baselines: Conneau et al. (2018) uses LSTM as sentence encoder and constraints bilingual sentence pairs have similar embedding. The other baselines are pre-training based approaches. Multilingual BERT (Devlin et al., 2018) is to train masked language model on multilingual Wikipedia. And Artetxe and Schwenk (2018) is pre-trained with machine translation model and takes the MT encoder to produce sentence embedding. XLM (Lample and Conneau, 2019) explores masked language model on multilingual Wikipedia, using translation language model on MT bilingual sentence pair for pre-training in addition.

Based on the result, we could find our pre-training model Unicoder obtains the best result in every fine-tuning setting. In TRANSLATE-TRAIN, TRANSLATE-TEST and Cross-lingual TEST, Unicoder obtains 76.9%, 74.9% and 75.4% accuracy on average, respectively. In Multi-language Fine-tuning, Unicoder outper-

| | en | fr | es | de | el | bg | ru | tr | ar | vi | th | zh | hi | sw | ur | average |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *Machine translate at training (TRANSLATE-TRAIN)* | | | | | | | | | | | | | | | | |
| Conneau et al. (2018) | 73.7 | 68.3 | 68.8 | 66.5 | 66.4 | 67.4 | 66.5 | 64.5 | 65.8 | 66.0 | 62.8 | 67.0 | 62.1 | 58.2 | 56.6 | 65.4 |
| Multilingual BERT (Devlin et al., 2018) | 81.9 | - | 77.8 | 75.9 | - | - | - | - | 70.7 | - | - | 76.6 | - | - | 61.6 | - |
| Multilingual BERT from Wu and Dredze 2019 | 82.1 | 76.9 | 78.5 | 74.8 | 72.1 | 75.4 | 74.3 | 70.6 | 70.8 | 67.8 | 63.2 | 76.2 | 65.3 | 65.3 | 60.6 | 71.6 |
| XLM (Lample and Conneau, 2019) | 85.0 | 80.2 | 80.8 | 80.3 | 78.1 | 79.3 | 78.1 | 74.7 | 76.5 | 76.6 | 75.5 | 78.6 | 72.3 | 70.9 | 63.2 | 76.7 |
| Unicoder | 85.1 | 80.0 | 81.1 | 79.9 | 77.7 | 80.2 | 77.9 | 75.3 | 76.7 | 76.4 | 75.2 | 79.4 | 71.8 | 71.8 | 64.5 | 76.9 |
| *Machine translate at test (TRANSLATE-TEST)* | | | | | | | | | | | | | | | | |
| Conneau et al. (2018) | 73.7 | 70.4 | 70.7 | 68.7 | 69.1 | 70.4 | 67.8 | 66.3 | 66.8 | 66.5 | 64.4 | 68.3 | 64.2 | 61.8 | 59.3 | 67.2 |
| Multilingual BERT (Devlin et al., 2018) | 81.4 | - | 74.9 | 74.4 | - | - | - | - | 70.4 | - | - | 70.1 | - | - | 62.1 | - |
| XLM (Lample and Conneau, 2019) | 85.0 | 79.0 | 79.5 | 78.1 | 77.8 | 77.6 | 75.5 | 73.7 | 73.7 | 70.8 | 70.4 | 73.6 | 69.0 | 64.7 | 65.1 | 74.2 |
| Unicoder | 85.1 | 80.1 | 80.3 | 78.2 | 77.5 | 78.0 | 76.2 | 73.3 | 73.9 | 72.8 | 71.6 | 74.1 | 70.3 | 65.2 | 66.3 | 74.9 |
| *Evaluation of cross-lingual sentence encoders (Cross-lingual TEST)* | | | | | | | | | | | | | | | | |
| Conneau et al. (2018) | 73.7 | 67.7 | 68.7 | 67.7 | 68.9 | 67.9 | 65.4 | 64.2 | 64.8 | 66.4 | 64.1 | 65.8 | 64.1 | 55.7 | 58.4 | 65.6 |
| Multilingual BERT (Devlin et al., 2018) | 81.4 | - | 74.3 | 70.5 | - | - | - | - | 62.1 | - | - | 63.8 | - | - | 58.3 | - |
| Multilingual BERT from Wu and Dredze 2019 | 82.1 | 73.8 | 74.3 | 71.1 | 66.4 | 68.9 | 69 | 61.6 | 64.9 | 69.5 | 55.8 | 69.3 | 60.0 | 50.4 | 58.0 | 66.3 |
| Artetxe and Schwenk (2018) | 73.9 | 71.9 | 72.9 | 72.6 | 73.1 | 74.2 | 71.5 | 69.7 | 71.4 | 72.0 | 69.2 | 71.4 | 65.5 | 62.2 | 61.0 | 70.2 |
| XLM (Lample and Conneau, 2019) | 85.0 | 78.7 | 78.9 | 77.8 | 76.6 | 77.4 | 75.3 | 72.5 | 73.1 | 76.1 | 73.2 | 76.5 | 69.6 | 68.4 | 67.3 | 75.1 |
| Unicoder | 85.1 | 79.0 | 79.4 | 77.8 | 77.2 | 77.2 | 76.3 | 72.8 | 73.5 | 76.4 | 73.6 | 76.2 | 69.4 | 69.7 | 66.7 | 75.4 |
| *Multi-language Fine-tuning* | | | | | | | | | | | | | | | | |
| XLM (Lample and Conneau, 2019) | 85.0 | 80.8 | 81.3 | 80.3 | 79.1 | 80.9 | 78.3 | 75.6 | 77.6 | 78.5 | 76.0 | 79.5 | 72.9 | 72.8 | 68.5 | 77.8 |
| Unicoder w/o Word Recovery | 85.2 | 80.5 | 81.8 | 80.9 | 79.7 | 81.1 | 79.3 | 76.2 | 78.2 | 78.5 | 76.4 | 79.7 | 73.4 | 73.6 | 68.8 | 78.2 |
| Unicoder w/o Paraphrase Classification | 85.5 | 81.1 | 82.0 | 81.1 | 80.0 | 81.3 | 79.6 | 76.6 | 78.2 | 78.2 | 75.9 | 79.9 | 73.7 | 74.2 | 69.3 | 78.4 |
| Unicoder w/o Cross-lingual Language Model | 85.5 | 81.9 | 81.8 | 80.5 | 80.5 | 81.0 | 79.3 | 76.4 | 78.1 | 78.3 | 76.3 | 79.6 | 72.9 | 73.0 | 68.7 | 78.3 |
| Unicoder | **85.6** | **81.1** | **82.3** | **80.9** | **79.5** | **81.4** | **79.7** | **76.8** | **78.2** | **77.9** | **77.1** | **80.5** | **73.4** | **73.8** | **69.6** | **78.5** |

Table 2: Test accuracy on the 15 XNLI languages. This table is organized by fine-tuning and test approaches. TRANSLATE-TRAIN is to machine translate English training data to target language and fine-tune with this translated data; TRANSLATE-TEST is machine translate target language test data to English, the fine-tuning is conducted on English; Cross-lingual TEST is to fine-tune on English and directly test on target language; Multi-language Fine-tune is to fine-tune on machine translated training data on all languages.

forms XLM by 0.7%. By translating the English training data to target language, both TRANSLATE-TRAIN and Multi-language Fine-tuning can outperform other fine-tuning approaches on average no matter what encoder is used. Our Multi-language Fine-tuning approach is even better than TRANSLATE-TRAIN. With this approach, XLM is been improved by 1.1% and Unicoder is been improved by 1.6%.

By Combining Unicoder and Multi-language Fine-tuning, Unicoder achieves an new state of the art with 78.5%. It obtains 1.8% accuracy gain compared to previous state of the art, XLM fine-tuned with TRANSLATE-TRAIN.

**XQA: Cross-lingual Question Answering** We proposed a new dataset XQA. Question Answering takes a question and an answer as input, then classify whether the answer is relevant to question. Each answer is a short passage. XQA contains three languages including English, French and German. Only English have training data. The training data cover various domains, such as health, tech, sports, etc. The English training data contains millions of samples and each languages has 500 test data.

Keeping the same experimental setup as XNLI, we also evaluate Unicoder on XQA and set XLM fine-tuned on our dataset as baseline with their published code. We split 5K data from training dataset as development data to do model selection. The results are shown in Table 3.

| | en | fr | de | average |
|---|---|---|---|---|
| *Machine translate at training (TRANSLATE-TRAIN)* | | | | |
| XLM (Lample and Conneau, 2019) | 80.2 | 65.1 | 63.3 | 64.2 |
| Unicoder | 81.1 | 66.2 | 66.5 | 66.4 |
| *Evaluation of cross-lingual sentence encoders (Cross-lingual TEST)* | | | | |
| XLM (Lample and Conneau, 2019) | 80.2 | 62.3 | 61.7 | 62.0 |
| Unicoder | 81.1 | 64.1 | 63.7 | 63.9 |
| *Multi-language Fine-tuning* | | | | |
| BERT (Devlin et al., 2018) | 76.4 | 61.6 | 64.6 | 63.1 |
| XLM (Lample and Conneau, 2019) | 80.7 | 67.1 | 68.2 | 67.7 |
| Unicoder | **81.4** | **69.3** | **70.1** | **69.7** |

Table 3: Results on the XQA. The average column is the average of fr and de result.

We could find that 1) Our model outperforms XLM at every fine-tuning setting. In Multi-language Fine-tuning, we achieved 2.0% gain. 2) With our Unicoder, Multi-language Fine-tuning approach achieved 3.3% gain compared to TRANSLATE-TRAIN. XLM also could been improved by 3.5%. 3) By combining Unicoder and TRANSLATE-TRAIN, we achieve best per-

| language number | XNLI-en Acc[%] | XNLI-ar Acc[%] | XNLI-es Acc[%] | XNLI-fr Acc[%] | XNLI-ru Acc[%] | XNLI-zh Acc[%] | average Acc[%] |
|---|---|---|---|---|---|---|---|
| 1 | 85.1 | 76.7 | 81.1 | 80.0 | 77.9 | 79.4 | 80.0 |
| 2 | 85.2 | 77.5 | 81.5 | 80.0 | 77.6 | 80.0 | 80.3 |
| 6 | 85.3 | 77.9 | 81.5 | 80.4 | 78.8 | 79.9 | 80.6 |
| 15 | 85.6 | 78.2 | 82.3 | 81.1 | 79.7 | 80.5 | 81.2 |

Table 4: Experiments of fine-tuning on different number of languages. The model is evaluated on 6 languages, and the average result is at last column. The results in the last row correspond to the results in last row of Table 2.

formance 69.7% on XQA. Compared to XLM + TRANSLATE-TRAIN baseline, we have 5.5% gain.

## Analysis

In this section, we provide ablation analysis for different variants of our approaches and elucidate some interesting aspects of Unicoder. Sec. 5.1 is the ablation study of each cross-lingual pre-training task. It also shows the impact of Multi-language Fine-tuning. Sec. 5.2 explores the impact of language numbers. Additionally, Sec. 5.3 analyzes the relation between English and other language by joint fine-tune on two languages. Then we further explore the relation between any language pair (Sec. 5.4).

### Ablation Study

To examine the utility of our new cross-lingual pre-training tasks, we conducted ablation study on XNLI dataset. For these three cross-lingual pre-training tasks, we remove them and only pre-train on other tasks. To this end, we fine-tune the Unicoder with Multi-language Fine-tuning. The results are present at Table 2.

Ablation experiments for each factor showed that removing any tasks will lead to performance drop. Comparing Unicoder with XLM, We can draw several conclusions from the results in Table 2. First, the cross-lingual paraphrase classification has least drop compared to others and removing the word recovery task hurts performance significantly. For example, in the case of XNLI, using just the Cross-lingual Language Model and Paraphrase Classification improves test accuracy on average by 0.4%. And integrating with Word Recovery model allows Unicoder to learn a better representation improves the average accuracy another 0.3%. Second, Multi-language fine-tuning is helpful to find the relation between languages, we will analyze it below. Table 2 and Table 3 both show it can bring a significant boost in cross-lingual language understanding performance. With the help of Multi-language fine-tuning, Unicoder is been improved by 1.6% of accuracy on XNLI and 3.3% on XQA.

### The relation between language number and fine-tuning performance

In Table 2, we proved that Multi-language Fine-tuning with 15 languages is better than TRANSLATE-TRAIN who only fine-tune on 1 language. In this sub-section, we try more setting to analysis the relation between language number and fine-tuning performance.

In this experiment, only English has human labeled training data, the other languages use machine translated training data from English. The experiment is conducted on 6 languages which are the languages of MT corpus Multilingual United Nations (MultiUN).

We have four settings: **1 language** is equals to TRANSLATE-TRAIN, the pre-trained model is fine-tuned on target language. **2 languages** is to fine-tune on English and target language. For English, we report the average result when fine-tune with other 5 languages, respectively. **6 languages** is to fine-tune on 6 selected languages of this experiment. **15 languages** is to fine-tune on all 15 languages our model support, and report the results on 6 languages. This setting is equals to last row of Table 2.

The results are shown at Table 4. In most languages, we could find that the more languages we used in fine-tuning, the better the performance. Chinese and Russian have two numbers don't follow this trend. But 15 languages always outperform 1 language for each languages.

The most surprising result is English could be improved by Multi-language Fine-tuning even it

| Train \ Test | en | fr | es | de | el | bg | ru | tr | ar | vi | th | zh | hi | sw | ur | average |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| en | **85.1** | 79.0 | 79.4 | 77.8 | 77.2 | 77.2 | 76.3 | 72.8 | 73.5 | 76.4 | 73.6 | 76.2 | 69.4 | 69.7 | 66.7 | 75.4 |
| fr | 77.0 | **80.0** | 79.1 | 76.9 | 77.4 | 78.6 | 76.1 | 73.1 | 73.7 | 75.3 | 73.4 | 76.2 | 70.1 | 70.3 | 66.2 | 74.9 |
| es | 79.6 | 78.5 | **81.1** | 77.5 | 77.9 | 78.2 | 75.8 | 73.4 | 73.9 | 75.7 | 73.4 | 76.8 | 69.5 | 70.1 | 66.8 | 75.2 |
| de | 78.2 | 77.3 | 78.3 | **79.9** | 77.1 | 77.5 | 76.7 | 73.1 | 73.8 | 75.2 | 73.4 | 76.6 | 70.9 | 71.3 | 66.7 | 75.1 |
| el | 76.8 | 76.0 | 77.3 | 75.1 | 77.7 | 76.1 | 74.6 | 71.3 | 74.2 | 76.2 | 74.1 | 76.8 | 70.8 | 69.2 | 68.0 | 74.3 |
| bg | 75.7 | 78.7 | 78.4 | 78.0 | 77.9 | **80.2** | 77.2 | 74.0 | 74.4 | 75.5 | 73.4 | 76.6 | 70.1 | 71.0 | 67.3 | 75.2 |
| ru | 76.4 | 78.7 | 79.0 | 78.3 | **78.0** | 78.8 | **77.9** | 73.6 | 75.5 | 76.3 | 74.7 | 77.5 | 71.2 | 71.2 | **68.3** | **75.7** |
| tr | 75.4 | 75.7 | 76.3 | 75.3 | 76.2 | 76.0 | 74.8 | **75.3** | 73.0 | 73.2 | 72.1 | 75.0 | 69.7 | 70.0 | 65.6 | 73.6 |
| ar | 75.5 | 76.7 | 78.4 | 77.5 | 76.7 | 77.6 | 76.1 | 73.4 | **76.7** | 75.7 | 73.4 | 76.3 | 70.1 | 70.5 | 66.2 | 74.7 |
| vi | 75.1 | 77.0 | 78.6 | 77.1 | 77.2 | 78.1 | 76.2 | 73.1 | 74.0 | **76.4** | 73.6 | 76.1 | 70.5 | 71.0 | 66.9 | 74.7 |
| th | 75.8 | 76.0 | 76.8 | 74.3 | 75.7 | 76.5 | 74.8 | 74.8 | 73.1 | 75.4 | **75.2** | 76.1 | 68.7 | 69.1 | 67.0 | 74.0 |
| zh | 78.0 | 77.5 | 78.3 | 76.6 | 76.9 | 77.4 | 75.8 | 73.4 | 73.9 | 76.1 | 75.2 | **79.4** | 69.9 | 70.8 | 67.4 | 75.1 |
| hi | 76.4 | 76.5 | 77.5 | 75.9 | 76.4 | 74.9 | 72.7 | 72.9 | 73.1 | 74.3 | 72.6 | 75.7 | **71.8** | 70.1 | 67.8 | 73.9 |
| sw | 76.2 | 76.0 | 76.8 | 74.6 | 75.8 | 75.8 | 74.7 | 72.4 | 73.2 | 74.4 | 72.7 | 74.8 | 70.0 | **71.8** | 66.3 | 73.7 |
| ur | 72.1 | 70.5 | 71.0 | 70.5 | 70.5 | 70.2 | 69.9 | 66.9 | 67.7 | 68.8 | 69.2 | 71.5 | 67.2 | 66.9 | 64.5 | 69.2 |

Table 5: Accuracy on the XNLI test set of when fine-tuning Unicoder with one language and testing on other languages. The results in the diagonal correspond to the TRANSLATE-TRAIN accuracy reported in Table 2.

is source language and has human-labeled training data. In next experiment, we will show that in 2 languages setting, the improvement on English is not stable and depends on the another language. But from 1 language to 6 languages and to 15 languages, English has stable improvement.

| language pairs | XNLI-en Acc[%] | average Acc[%] |
|---|---|---|
| ar-en | 84.8 | 76.0 |
| bg-en | 85.1 | 76.5 |
| de-en | 84.3 | 76.1 |
| el-en | 84.8 | 76.2 |
| es-en | 85.3 | 76.6 |
| fr-en | 85.4 | 76.8 |
| hi-en | 84.4 | 76.3 |
| ru-en | 85.2 | 76.2 |
| sw-en | 85.0 | 75.7 |
| th-en | 84.1 | 75.4 |
| tr-en | 85.0 | 75.5 |
| ur-en | 85.0 | 75.2 |
| vi-en | 84.4 | 75.2 |
| zh-en | 85.1 | 76.6 |
| en | 85.1 | 75.3 |

Table 6: Result of joint fine-tuning two languages. This table reports the result on English and average accuracy of 15 languages on XNLI. The last row means Unicoder only fine-tunes on English.

**The relation between English and other languages**

In this experiment, we joint fine-tune English and one language. With this experiment, we could test the relation between English and other languages since all languages have equal position in the pre-training and fine-tuning.

We report the performance on English and average of 15 languages. First, we could find most of the average results are improved by joint fine-tuning two languages. Only Vietnamese(vi) and Urdu(ur) lead to performance drop. Secondly, the improvement on English is not stable. French(fr) and Spanish(es) could improve English performance. But Vietnamese(vi) and Thai(th) lead to a big drop.

**The relation between different languages**

So as to better understand the relation between different languages, we fine-tune Unicoder on one language and test on all 15 languages. The results are shown in Table 5. The numbers in the diagonal correspond to the TRANSLATE-TRAIN result reported in Table 2.

We observe that the Unicoder can transfer knowledge from one language to another language. We could find that fine-tune on one language often lead to best performance on this language, except Greek(el) and Urdu(ur). In fact, TRANSLATE-TRAIN of Urdu even harm the performance. Urdu also have worst generation ability to other languages. Russian(ru) has best generalization ability, even better than source language

English.

We also could find that transfer between English(en), Spanish(es) and French(fr) is easier that other languages. The MT system between these languages also outperform other languages (Conneau et al., 2018).

## Conclusion

We have introduced the Unicoder which is insensitive to different languages. We pre-train Unicoder with three new cross-lingual tasks, including cross-lingual word recovery, cross-lingual paraphrase classification and cross-lingual masked language model. We also proposed a new Multi-language Fine-tuning approach. The experiments on XNLI and XQA proved Unicoder could bring large improvements. and our approach become new state of the art on XNLI and XQA. We also did experiments to show that the more languages we used in fine-tuning, the better the results. Even rich-resource language also could been improved.

## References


Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A Smith. 2016. Massively multilingual word embeddings. *arXiv preprint arXiv:1602.01925*.

Mikel Artetxe and Holger Schwenk. 2018. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *arXiv preprint arXiv:1812.10464*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Pichuan Chang, Michel Galley, and Christopher D Manning. 2008. Optimizing chinese word segmentation for machine translation performance. pages 224–232.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. Xnli: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Andreas Eisele and Yu Chen. 2010. Multiun: A multilingual corpus from united nation documents. In *Proceedings of the Seventh conference on International Language Resources and Evaluation*, pages 2868–2872. European Language Resources Association (ELRA).

Mandy Guo, Qinlan Shen, Yinfei Yang, Heming Ge, Daniel M Cer, Gustavo Hernandez Abrego, Keith Stevens, Noah Constant, Yunhsuan Sung, Brian Strope, et al. 2018. Effective parallel corpus mining using bilingual sentence embeddings. *arXiv: Computation and Language*, pages 165–176.

Dan Hendrycks and Kevin Gimpel. 2017. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *arXiv: Learning*.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1681–1691.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *international conference on learning representations*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callisonburch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Corbett Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. *The Association for Computational Linguistics*.

Anoop Kunchukuttan, Pratik Mehta, and Pushpak Bhattacharyya. 2017. The iit bombay english-hindi parallel corpus. *language resources and evaluation*.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *CoRR*, abs/1901.07291.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, Hervé Jégou, et al. 2018. Word translation without parallel data.

Liyuan Liu, Jingbo Shang, Frank F Xu, Xiang Ren, Huan Gui, Jian Peng, and Jiawei Han. 2018. Empower sequence labeling with task-aware neural language model. *national conference on artificial intelligence*, pages 5253–5260.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Improving multi-task deep neural networks via knowledge distillation for natural language understanding. *arXiv: Computation and Language*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6294–6305.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013. Exploiting similarities among languages for machine translation. *arXiv: Computation and Language*.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 2227–2237.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1715–1725.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. pages 353–355.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. *north american chapter of the association for computational linguistics*, 1:1112–1122.

Shijie Wu and Mark Dredze. 2019. Beto, bentz, becas: The surprising cross-lingual effectiveness of bert. *arXiv preprint arXiv:1904.09077*.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1006–1011.

# Addressing Semantic Drift in Question Generation for Semi-Supervised Question Answering

**Shiyue Zhang** **Mohit Bansal**
UNC Chapel Hill
{shiyue, mbansal}@cs.unc.edu

## Abstract

Text-based Question Generation (QG) aims at generating natural and relevant questions that can be answered by a given answer in some context. Existing QG models suffer from a "semantic drift" problem, i.e., the semantics of the model-generated question drifts away from the given context and answer. In this paper, we first propose two semantics-enhanced rewards obtained from downstream question paraphrasing and question answering tasks to regularize the QG model to generate semantically valid questions. Second, since the traditional evaluation metrics (e.g., BLEU) often fall short in evaluating the quality of generated questions, we propose a QA-based evaluation method which measures the QG model's ability to mimic human annotators in generating QA training data. Experiments show that our method achieves the new state-of-the-art performance w.r.t. traditional metrics, and also performs best on our QA-based evaluation metrics. Further, we investigate how to use our QG model to augment QA datasets and enable semi-supervised QA. We propose two ways to generate synthetic QA pairs: generate new questions from existing articles or collect QA pairs from new articles. We also propose two empirically effective strategies, a data filter and mixing mini-batch training, to properly use the QG-generated data for QA. Experiments show that our method improves over both BiDAF and BERT QA baselines, even without introducing new articles.[1]

## 1 Introduction

In contrast to the rapid progress shown in Question Answering (QA) tasks (Rajpurkar et al., 2016; Joshi et al., 2017; Yang et al., 2018), the task of Question Generation (QG) remains understudied and challenging. However, as an important dual task to QA, QG can not only be used to augment QA datasets (Duan et al., 2017), but can also be applied in conversation and education systems (Heilman and Smith, 2010; Lindberg et al., 2013). Furthermore, given that existing QA models often fall short by doing simple word/phrase matching rather than true comprehension (Jia and Liang, 2017), the task of QG, which usually needs complicated semantic reasoning and syntactic variation, should be another way to encourage true machine comprehension (Lewis and Fan, 2019). Recently, we have seen an increasing interest in the QG area, with mainly three categories: Text-based QG (Du et al., 2017; Zhao et al., 2018), Knowledge-Base-based QG (Reddy et al., 2017; Serban et al., 2016), and Image-based QG (Li et al., 2018; Jain et al., 2017). Our work focuses on the Text-based QG branch.

| |
|---|
| **Context**: ...during the age of enlightenment, philosophers such as **john locke** advocated the principle in their writings, whereas others, such as thomas hobbes, strongly opposed it. montesquieu was one of the foremost supporters of separating the legislature, the executive, and the judiciary... |
| **Gt**: who was an advocate of separation of powers?<br>**Base**: who opposed the principle of enlightenment?<br>**Ours**: who advocated the principle in the age of enlightenment? |

Table 1: An examples of the "semantic drift" issue in Question Generation ("Gt" is short for "ground truth").

Current QG systems follow an attention-based sequence-to-sequence structure, taking the paragraph-level context and answer as inputs and outputting the question. However, we observed that these QG models often generate questions that semantically drift away from the given context and answer; we call this the "semantic drift" problem. As shown in Table 1, the baseline QG model generates a question that has almost contrary semantics with the ground-truth question, and the generated phrase "the principle of en-

[1]Code and models publicly available at: https://github.com/ZhangShiyue/QGforQA



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2495–2509, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

lightenment" does not make sense given the context. We conjecture that the reason for this "semantic drift" problem is because the QG model is trained via teacher forcing only, without any high-level semantic regularization. Hence, the learned model behaves more like a question language model with some loose context constraint, while it is unaware of the strong requirements that it should be closely grounded by the context and should be answered by the given answer. Therefore, we propose two semantics-enhanced rewards to address this drift: **QPP** and **QAP**. Here, **QPP** refers to **Q**uestion **P**araphrasing **P**robability, which is the probability of the generated question and the ground-truth question being paraphrases; **QAP** refers to **Q**uestion **A**nswering **P**robability, which is the probability that the generated question can be correctly answered by the given answer. We regularize the generation with these two rewards via reinforcement learning. Experiments show that these two rewards can significantly improve the question generation quality separately or jointly, and achieve the new state-of-the-art performance on the SQuAD QG task.

Next, in terms of QG evaluation, previous works have mostly adopted popular automatic evaluation metrics, like BLEU, METEOR, etc. However, we observe that these metrics often fall short in properly evaluating the quality of generated questions. First, they are not always correlated to human judgment about answerability (Nema and Khapra, 2018). Second, since multiple questions are valid but only one reference exists in the dataset, these traditional metrics fail to appropriately score question paraphrases and novel generation (shown in Table 2). Therefore, we introduce a QA-based evaluation method that directly measures the QG model's ability to mimic human annotators in generating QA training data, because ideally, we hope that the QG model can act like a human to ask questions. We compare different QG systems using this evaluation method, which shows that our semantics-reinforced QG model performs best. However, this improvement is relatively minor compared to our improvement on other QG metrics, which indicates improvement on typical QG metrics does not always lead to better question annotation by QG models for generating QA training set.

Further, we investigate how to use our best QG system to enrich QA datasets and perform semi-supervised QA on SQuADv1.1 (Rajpurkar et al., 2016). Following the back-translation strategy that has been shown to be effective in Machine Translation (Sennrich et al., 2016) and Natural Language Navigation (Fried et al., 2018; Tan et al., 2019), we propose two methods to collect synthetic data. First, since multiple questions can be asked for one answer while there is only one human-labeled ground-truth, we make our QG model generate new questions for existing context-answer pairs in SQuAD training set, so as to enrich it with paraphrased and other novel but valid questions. Second, we use our QG model to label new context-answer pairs from new Wikipedia articles. However, directly mixing synthetic QA pairs with ground-truth data will not lead to improvement. Hence, we introduce two empirically effective strategies: one is a data filter based on QAP (same as the QAP reward) to filter out examples that have low probabilities to be correctly answered; the other is a "mixing mini-batch training" strategy that always regularizes the training signal with the ground-truth data. Experiments show that our method improves both BiDAF (Seo et al., 2016; Clark and Gardner, 2018) and BERT (Devlin et al., 2018) QA baselines by 1.69/1.27 and 1.19/0.56 absolute points on EM/F1, respectively; even without introducing new articles, it can bring 1.51/1.13 and 0.95/0.13 absolute improvement, respectively.

## 2 Related Works

**Question Generation** Early QG studies focused on using rule-based methods to transform statements to questions (Heilman and Smith, 2010; Lindberg et al., 2013; Labutov et al., 2015). Recent works adopted the attention-based sequence-to-sequence neural model (Bahdanau et al., 2014) for QG tasks, taking answer sentence as input and outputting the question (Du et al., 2017; Zhou et al., 2017), which proved to be better than rule-based methods. Since human-labeled questions are often relevant to a longer context, later works leveraged information from the whole paragraph for QG, either by extracting additional information from the paragraph (Du and Cardie, 2018; Song et al., 2018; Liu et al., 2019) or by directly taking the whole paragraph as input (Zhao et al., 2018; Kim et al., 2018; Sun et al., 2018). A very recent concurrent work applied the large-scale language model pre-training strategy for QG and

also achieved a new state-of-the-art performance (Dong et al., 2019). However, the above models were trained with teacher forcing only. To address the exposure bias problem, some works applied reinforcement learning taking evaluation metrics (e.g., BLEU) as rewards (Song et al., 2017; Kumar et al., 2018). Yuan et al. (2017) proposed to use a language model's perplexity ($R_{PPL}$) and a QA model's accuracy ($R_{QA}$) as two rewards but failed to get significant improvement. Their second reward is similar to our QAP reward except that we use QA probability rather than accuracy as the probability distribution is more smooth. Hosking and Riedel (2019) compared a set of different rewards, including $R_{PPL}$ and $R_{QA}$, and claimed none of them improved the quality of generated questions. For QG evaluation, even though some previous works conducted human evaluations, most of them still relied on traditional metrics (e.g., BLEU). However, Nema and Khapra (2018) pointed out the existing metrics do not correlate with human judgment about answerability, so they proposed "Q-metrics" that mixed traditional metrics with an "answerability" score. In our work, we will show QG results on traditional metrics, Q-metrics, as well as human evaluation, and also propose a QA-based QG evaluation.

**Question Generation for QA** As the dual task of QA, QG has been often proposed for improving QA. Some works have directly used QG in QA models' pipeline (Duan et al., 2017; Dong et al., 2017; Lewis and Fan, 2019). Some other works enabled semi-supervised QA with the help of QG. Tang et al. (2017) applied the "dual learning" algorithm (He et al., 2016) to learn QA and QG jointly with unlabeled texts. Yang et al. (2017) and Tang et al. (2018) followed the GAN (Goodfellow et al., 2014) paradigm, taking QG as a generator and QA as a discriminator, to utilize unlabeled data. Sachan and Xing (2018) proposed a self-training cycle between QA and QG. However, these works either reduced the ground-truth data size or simplified the span-prediction QA task to answer sentence selection. Dhingra et al. (2018) collected 3.2M cloze-style QA pairs to pre-train a QA model, then fine-tune with the full ground-truth data which improved a BiDAF-QA baseline. In our paper, we follow the back-translation (Sennrich et al., 2016) strategy to generate new QA pairs by our best QG model to augment SQuAD training set. Further, we introduce a data filter to remove poorly generated examples and a mixing mini-batch training strategy to more effectively use the synthetic data. Similar methods have also been applied in some very recent concurrent works (Dong et al., 2019; Alberti et al., 2019) on SQuADv2.0. The main difference is that we also propose to generate new questions from existing articles without introducing new articles.

## 3 Question Generation

### 3.1 Base Model

We first introduce our base model which mainly adopts the model architecture from the previous state-of-the-art (Zhao et al., 2018). The differences are that we introduce two linguistic features (POS & NER), apply deep contextualized word vectors, and tie the output projection matrix with the word embedding matrix. Experiments showed that with these additions, our base model results surpass the results reported in Zhao et al. (2018) with significant margins. Our base model architecture is shown in the upper box in Figure 1 and described as follow. If we have a paragraph $p = \{x_i\}_{i=1}^{M}$ and an answer $a$ which is a sub-span of $p$, the target of the QG task is to generate a question $q = \{y_j\}_{j=1}^{N}$ that can be answered by $a$ based on the information in $p$.

**Embedding** The model first concatenates four word representations: word vector, answer tag embedding, Part-of-Speech (POS) tag embedding, and Name Entity (NER) tag embedding, i.e., $e_i = [w_i, a_i, p_i, n_i]$. For word vectors, we use the deep contextualized word vectors from ELMo (Peters et al., 2018) or BERT (Devlin et al., 2018). The answer tag follows the BIO[2] tagging scheme.

**Encoder** The output of the embedding layer is then encoded by a two-layer bi-directional LSTM-RNNs, resulting in a list of hidden representations $H$. At any time step $i$, the representation $h_i$ is the concatenation of $\overrightarrow{h_i}$ and $\overleftarrow{h_i}$.

$$\begin{aligned} \overrightarrow{h}_i &= \overrightarrow{LSTM}([e_i; \overrightarrow{h}_{i-1}]) \\ \overleftarrow{h}_i &= \overleftarrow{LSTM}([e_i; \overleftarrow{h}_{i+1}]) \\ H &= [\overrightarrow{h_i}, \overleftarrow{h_i}]_{i=1}^{M} \end{aligned} \quad (1)$$

[2] "B", for "Begin", tags the start token of the answer span; "I", for "Inside", tags other tokens in the answer span; "O", for "Other", tags other tokens in the paragraph.

**Self-attention** A gated self-attention mechanism (Wang et al., 2017) is applied to $H$ to aggregate the long-term dependency within the paragraph. $\alpha_i$ is an attention vector between $h_i$ and each element in $H$; $u_i$ is the self-attention context vector for $h_i$; $h_i$ is then updated to $f_i$ using $u_i$; a soft gate $g_i$ decides how much the update is applied. $\hat{H} = [\hat{h}_i]_{i=1}^{M}$ is the output of this layer.

$$
\begin{aligned}
u_i = H\alpha_i, \alpha_i = softmax(H^T W^u h_i) \\
f_i = tanh(W^f[h_i; u_i]) \\
g_i = sigmoid(W^g[h_i; u_i]) \\
\hat{h}_i = g_i * f_i + (1 - g_i) * h_i
\end{aligned} \tag{2}
$$

**Decoder** The decoder is another two-layer unidirectional LSTM-RNN. An attention mechanism dynamically aggregates $\hat{H}$ at each decoding step to a context vector $c_j$ which is then used to update the decoder state $s_j$.

$$
\begin{aligned}
c_j = \hat{H}\alpha_j, \alpha_j = softmax(\hat{H}^T W^a s_j) \\
\tilde{s}_j = tanh(W^c[c_j; s_j]) \\
s_{j+1} = LSTM([y_j; \tilde{s}_j])
\end{aligned} \tag{3}
$$

The probability of the target word $y_j$ is computed by a maxout neural network.

$$
\begin{aligned}
\tilde{o}_j = tanh(W^o[c_j; s_j]) \\
o_j = [max\{\tilde{o}_{j,2k-1}, \tilde{o}_{j,2k}\}]_k \\
p(y_j|y_{<j}) = softmax(W^e o_j)
\end{aligned} \tag{4}
$$

In practice, we keep the weight matrix $W^e$ the same as the word embedding matrix and fix it during training. Furthermore, we apply the same "maxout pointer" proposed by Zhao et al. (2018) to enable the model to copy words from input.

### 3.2 Semantics-Reinforced Model

To address the "semantic drift" problem shown in Table 1, we propose two semantics-enhanced rewards to regularize the generation to focus on generating semantically valid questions.

**QPP Reward** To deal with the "exposure bias" problem, many previous works directly used the final evaluation metrics (e.g., BLEU) as rewards to train the generation models (Rennie et al., 2017; Paulus et al., 2017). However, these metrics sometimes fail to evaluate equally to question paraphrases and thus provide inaccurate rewards. Hence, we propose to use a pre-trained question paraphrasing classification (QPC) model

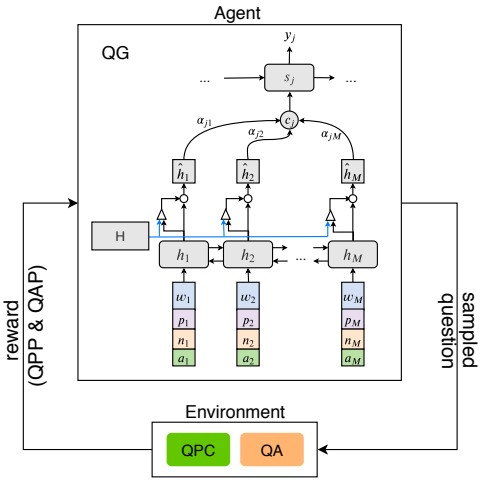

Figure 1: The architecture of our semantics-reinforced QG model.

to provide paraphrasing probability as a reward. Since paraphrasing is more about semantic similarity than superficial word/phrase matching, it treats question paraphrases more fairly (Example 1 in Table 2). Therefore, we first train a QPC model with Quora Question Pairs dataset. Next, we take it as an environment, and the QG model will interact with it during training to get the probability of the generated question and the ground-truth question being paraphrases as the reward.

**QAP Reward** Two observations motivate us to introduce QAP reward. First, in a paragraph, usually, there are several facts relating to the answer and can be used to ask questions. Neither the teacher forcing or the QPP reward can favor this kind of novel generation (Example 2 in Table 2). Second, we find semantically-drifted questions are usually unanswerable by the given answer. Therefore, inspired by the dual learning algorithm (He et al., 2016), we propose to take the probability that a pre-trained QA model can correctly answer the generated question as a reward, i.e., $p(a^*|q^s; p)$, where $a^*$ is the ground-truth answer and $q^s$ is a sampled question. Using this reward, the model can not only gets positive rewards for novel generation but also be regularized by the answerability requirement. Note that, this reward is supposed to be carefully used because the QG model can cheat by greedily copying words in/near the answer to the generated question. In this case, even though high QAPs are achieved, the model loses the question generation ability.

**Policy Gradient** To apply these two rewards, we use the REINFORCE algorithm (Williams, 1992)

| Example 1: Fail to score equally to paraphrases | BLEU4 | Q-BLEU1 | QPP | QAP |
|---|---|---|---|---|
| Context: ...the university first offered graduate degrees , in the form of a master of arts ( ma ) , in the the **1854** – 1855 academic year ...<br>Gt: in what year was a master of arts course first offered ? | | | | |
| Gen1: in what year did the university first offer a master of arts ? | 37.30 | 79.39 | 49.71 | 34.09 |
| Gen2: when did the university begin offering a master of arts ? | 29.58 | 47.50 | 46.12 | 18.18 |
| Example 2: Fail to score appropriately to novel generation | | | | |
| Context: ...in **1987** , when some students believed that the observer began to show a conservative bias , a liberal newspaper , common sense was published...<br>Gt: in what year did the student paper common sense begin publication ? | | | | |
| Gen1: in what year did common sense begin publication ? | 56.29 | 85.77 | 92.28 | 93.94 |
| Gen2: when did the observer begin to show a conservative bias ? | 15.03 | 21.11 | 13.44 | 77.15 |

Table 2: Two examples of where QPP and QAP improve in question quality evaluation.

to learn a generation policy $p_\theta$ defined by the QG model parameters $\theta$. We minimize the loss function $L_{RL} = -E_{q^s \sim p_\theta}[r(q^s)]$, where $q^s$ is a sampled question from the model's output distribution. Due to the non-differentiable sampling procedure, the gradient is approximated using a single sample with some variance reduction baseline $b$:

$$\bigtriangledown_\theta L_{RL} = -(r(q^s) - b) \bigtriangledown_\theta log p_\theta(q^s) \quad (5)$$

We follow the effective SCST strategy (Rennie et al., 2017) to take the reward of greedy search result $q^g$ as the baseline, i.e. $b = r(q^g)$. However, only using this objective to train QG will result in poor readability, so we follow the mixed loss setting (Paulus et al., 2017): $L_{mixed} = \gamma L_{RL} + (1 - \gamma) L_{ML}$. In practice, we find the mixing ratio $\gamma$ for QAP reward should be lower, i.e., it needs more regularization from teacher forcing, so that it can avoid the undesirable cheating issue mentioned above. Furthermore, we also apply the multi-reward optimization strategy (Pasunuru and Bansal, 2018) to train the model with two mixed losses alternately with an alternate rate $n : m$, i.e., train with $L^{qpp}_{mixed}$ for $n$ mini-batches, then train with $L^{qap}_{mixed}$ for $m$ mini-batches, repeat until convergence. $n$ and $m$ are two hyper-parameters.

$$\begin{aligned} L^{qpp}_{mixed} &= \gamma^{qpp} L^{qpp}_{RL} + (1 - \gamma^{qpp}) L_{ML} \\ L^{qap}_{mixed} &= \gamma^{qap} L^{qap}_{RL} + (1 - \gamma^{qap}) L_{ML} \end{aligned} \quad (6)$$

Experiments show that these two rewards can significantly improve the QG performance separately or jointly, and we achieve new state-of-the-art QG performances, see details in Section 6.

### 3.3 QA-Based QG Evaluation

Inspired by the idea that "a perfect QG model can replace humans to ask questions", we introduce a QA-based evaluation method that measures the quality of a QG model by its ability to mimic human annotators in labeling training data for QA models. The evaluation procedure is described as follows. First, we sample some unlabeled Wikipedia paragraphs with pre-extracted answer spans from HarvestingQA dataset (Du and Cardie, 2018). Second, we make a QG model act as an "annotator" to annotate a question for each answer span. Third, we train a QA model using this synthetic QA dataset. Lastly, we use the QA model's performance on the original SQuAD development set as the evaluation for this QG model. The higher this QA performance is, the better the QG model mimics a human's question-asking ability. We believe that this method provides a new angle to evaluate QG model's quality and also a more reliable way to choose QG models to conduct data augmentation and semi-supervised QA.

## 4 Semi-Supervised Question Answering

Since one of the major goals of developing QG systems is to generate new QA pairs and augment QA datasets, we investigate how to use our QG system to act as a question annotator, collect new QA pairs, and conduct semi-supervised QA. Figure 2 illustrates the overall procedure of our semi-supervised QA approach.

### 4.1 Synthetic Data Generation

To generate synthetic QA pairs, we follow the effective "back translation" approach proposed in Neural Machine Translation (NMT) (Sennrich et al., 2016). In NMT, the back translation method first obtains synthetic source sentences by running a pre-trained target-to-source translation model on a monolingual dataset of the target language; then, it combines the synthetic and ground-truth translation pairs to train the desired source-to-target translation model. Similarly, in the QA scenario,

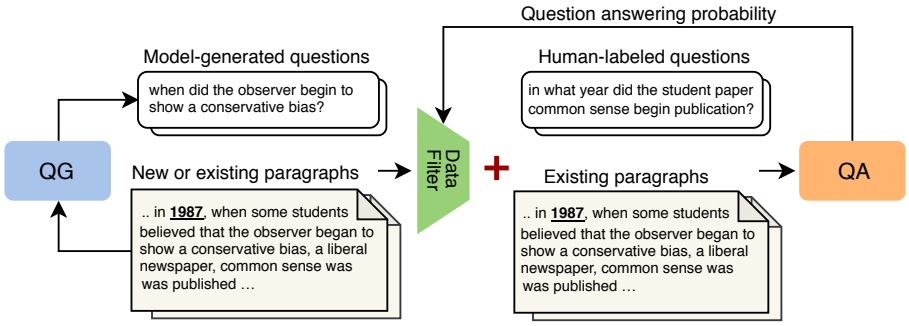


Figure 2: Semi-supervised QA: First, a trained QG model generates questions from new or existing paragraphs building up a synthetic QA dataset; Second, a data filter filters out low-QAP synthetic examples and augment the rest to human-labeled QA pairs; Lastly, the QA model is trained with the enlarged QA dataset.

the paragraph/answer can be viewed as the "target sentence", while the question can be taken as the "source sentence". One tricky difference is that even if the paragraphs can be easily obtained from Wikipedia, there are no answer span labels. Therefore, we use two sources to generate questions from, as discussed below.

**Generate from Existing Articles** In SQuAD (Rajpurkar et al., 2016), each context-answer pair only has one ground-truth question. However, usually, multiple questions can be asked. The diversity lies in question paraphrasing and different facts in the context that can be used to ask the question. Therefore, without introducing new Wikipedia articles, we make our QG model generate diverse questions for the existing context-answer pairs in SQuAD training set by keeping the all beam search outputs for each example.

**Generate from New Articles** To use unlabeled Wikipedia articles for data augmentation, an automatic answer extractor is indispensable. Some previous works have proposed methods to detect key phrases from a paragraph and automatically extract potential answer spans (Yang et al., 2017; Du and Cardie, 2018; Subramanian et al., 2018). Instead of building up our answer extractor, we directly take advantage of the released HarvestingQA dataset. It contains 1.2M synthetic QA pairs, in which both the answer extractor and the QG model were proposed by Du and Cardie (2018). We use their paragraphs with answer span labels but generate questions with our QG models, and only use their questions for comparison.

### 4.2 Synthetic Data Usage

In practice, we find that directly mixing the synthetic data with the ground-truth data does not improve QA performance. We conjecture the reason is that some poor-quality synthetic examples mislead the learning process of the QA model. Therefore, we propose two empirical strategies to better utilize synthetic data.

**QAP Data Filter** In "self-training" literature, similar issues have been discussed that using model-labeled examples to train the model will amplify the model's error. Later works proposed co-training or tri-training that uses two or three models as judges and only keeps examples that all models agree on (Blum and Mitchell, 1998; Zhou and Li, 2005). Sachan and Xing (2018) also designed question selection oracles based on curriculum learning strategy in their QA-QG self-training circle. In this paper, we simply design a data filter based on our QAP measure (same definition as the QAP reward) to filter poor-quality examples. We think if one question-answer pair has a low QAP, i.e., the probability of the answer given the question is low, it is likely to be a mismatched pair. Hence, we filter synthetic examples with $QAP < \epsilon$, where $\epsilon$ is a hyper-parameter that we will tune for different synthetic datasets.

**Mixing Mini-Batch Training** When conducting semi-supervised learning, we do not want gradients from ground-truth data are overwhelmed by synthetic data. Previous works (Fried et al., 2018; Dhingra et al., 2018) proposed to first pre-train the model with synthetic data and then fine-tune it with ground-truth data. However, we find when the synthetic data size is small (e.g., similar size as the ground-truth data), catastrophic forgetting

[3]They actually used the reversed dev-test setup as opposed to the original setup used in Du et al. (2017) and Du and Cardie (2018) (see Section 3.1 in Zhao et al. (2018)). Thus, we also conducted the reversed dev-test setup and our QPP&QAP model yields BLEU4/METEOR/ROUGE-L=20.76/24.20/48.91.

| | BLEU4 | METEOR | ROUGE-L | Q-BLEU1 | QPP | QAP |
|---|---|---|---|---|---|---|
| Du and Cardie (2018) | 15.16 | 19.12 | – | – | – | – |
| Zhao et al. (2018)[3] | 16.38 | 20.25 | 44.48 | – | – | – |
| Our baseline (w. ELMo) | 17.00 | 21.44 | 45.89 | 47.80 | 27.29 | 45.15 |
| + BLEU4 | 17.72 | 22.13 | 46.52 | 49.07 | 27.09 | 45.96 |
| + METEOR | 17.84 | 22.41 | 46.18 | 49.09 | 26.70 | 46.52 |
| + ROUGE-L | 17.78 | 22.28 | 46.51 | 49.23 | 27.06 | 46.31 |
| + QPP | 18.25 | 22.62 | 46.45 | 49.59 | **28.13** | 47.63 |
| + QAP | 18.12 | 22.52 | 46.45 | 49.27 | 27.49 | **48.76** |
| + QPP&QAP | **18.37** | **22.65** | **46.68** | **49.63** | 28.03 | 48.37 |

Table 3: The performance of different QG models.

will happen during fine-tuning, leading to similar results as using ground-truth data only. Thus, we propose a "mixing mini-batch" training strategy, where for each mini-batch we combine half mini-batch ground-truth data with half mini-batch synthetic data, which keeps the data mixing ratio to 1:1 regardless of what the true data size ratio is. In this way, we can have the training process generalizable to different amounts of synthetic data and keep the gradients to be regularized by ground-truth data.

## 5 Experiment Setup

**Datasets** For QG, we use the most commonly used SQuAD QG dataset first used by (Du et al., 2017). For QA-based QG evaluation, we obtain unlabeled paragraph and answer labels from HarvestingQA (Du and Cardie, 2018), and have different QG systems to label questions. For semi-supervised QA, we use SQuADv1.1 (Rajpurkar et al., 2016) as our base QA task, and split the original development set in half as our development and test set respectively. Plus, we make our QG model generate new questions from both SQuAD and HarvestingQA. We will sample 10% – 100% examples from HarvestingQA which are denoted by H1-10 in our experiments.

**Evaluation Metrics** For QG, we first adopt 3 traditional metrics (BLEU4/METEOR/ROUGE-L). Second, we apply the new Q-BLEU1 metric proposed by (Nema and Khapra, 2018). Moreover, we conduct a pairwise **human evaluation** between our baseline and QPP&QAP model on MTurk. We gave the annotators a paragraph with an answer bold in context and two questions generated by two models (randomly shuffled). We asked them to decide which one is better or non-distinguishable. For both QA-based QG evaluation and semi-supervised QA, we follow the standard evaluation method for SQuAD to use Exact Match (EM) and F1.

| QPP&QAP | Our baseline | Tie |
|---|---|---|
| 160 | 131 | 9 |

Table 4: Pairwise human evaluation between our baseline and QPP&QAP multi-reward model.

| Data | Du and Cardie | Our baseline | QPP & QAP |
|---|---|---|---|
| H1 | 53.20/65.47 | 55.06/67.83 | **55.89/68.26** |
| H2 | 53.40/66.28 | 56.23/**69.23** | **56.69**/69.19 |
| H3 | 53.12/65.57 | **57.14**/69.39 | 57.05/**70.17** |
| S+H1 | 71.16/80.75 | 71.94/81.26 | **72.20/81.44** |
| S+H2 | 72.02/81.00 | 72.03/81.38 | **72.22/81.81** |
| S+H3 | 71.48/81.02 | 72.61/81.46 | **72.69/82.22** |

Table 5: The QA-based evaluation results for different QG systems. The two numbers of each item in this table are the EM/F1 scores. All results are the performance on our QA test set. "S" is short for "SQuAD".

More details about datasets, evaluation metrics, human evaluation setup, and model implement details are provided in the Appendix.

## 6 Results

### 6.1 Question Generation

**Baselines** First, as shown in Table 3, our baseline QG model obtains a non-trivial improvement over previous best QG system (Zhao et al., 2018) which proves the effectiveness of our newly introduced setups: introduce POS/NER features, use deep contexturalized word vectors (from ELMo or BERT), and tie output projection matrix with non-trainable word embedding matrix. Second, we apply three evaluation metrics as rewards to deal with the exposure bias issue and improve performance. All the metrics are significantly[4] ($p < 0.001$) improved except QPP, which supports that high traditional evaluation metrics do not always correlate to high semantic similarity.

[4]The significance tests in this paper are conducted following the bootstrap test setup (Efron and Tibshirani, 1994).

| | Filter | Data Size[5] | EM | F1 |
|---|---|---|---|---|
| H1 only | $\epsilon = 0.0$ | 120k | 54.55 | 67.91 |
| | $\epsilon = 0.2$ | 84k | 61.18 | 71.65 |
| | $\epsilon = 0.4$ | 69k | **61.97** | **72.48** |
| | $\epsilon = 0.6$ | 55k | 60.38 | 70.51 |
| | $\epsilon = 0.8$ | 40k | 57.47 | 66.48 |
| SQuAD+H1 | $\epsilon = 0.0$ | 207k | 72.97 | 82.18 |
| | $\epsilon = 0.2$ | 171k | 73.88 | 82.72 |
| | $\epsilon = 0.4$ | 156k | 73.47 | 82.62 |
| | $\epsilon = 0.6$ | 142k | **73.96** | **82.81** |
| | $\epsilon = 0.8$ | 127k | 73.65 | 82.77 |

Table 6: The effect of QAP-based synthetic data filter. We filter out the synthetic data with $QAP < \epsilon$. All results are the performance on our QA development set.

**Semantics-Reinforced Models** As shown in Table 3, when using QAP and QQP separately, all metrics are significantly ($p < 0.001$) improved over our baseline and all metrics except ROUGE-L are significantly ($p < 0.05$) improved over the models using traditional metrics as rewards. After applying multi-reward optimization, our model performs consistently best on BLEU4/METEOR/ROUGE-L and Q-BLEU1. Notably, using one of these two rewards will also improve the other one at the same time, but using both of them achieves a good balance between these two rewards without exploiting either of them and results in the consistently best performance on other metrics, which is a new state-of-the-art result. **Human Evaluation Results:** Table 4 shows the MTurk anonymous human evaluation study, where we do a pairwise comparison between our baseline and QPP&QAP model. We collected 300 responses in total, 160 of which voted the QPP&QAP model's generation is better, 131 of which favors the baseline model, and 9 of which selected non-distinguishable.

**QA-Based Evaluation** As shown in Table 5, we compare three QG systems using QA-based evaluation on three different amounts of synthetic data and their corresponding semi-supervised QA setups (without filter). It can be observed that both our baseline and our best QG model can significantly improve the synthetic data's QA performance, which means they can act as better "annotators" than the QG model proposed by Du and Cardie (2018). However, our best QG model only has a minor improvement over our baseline model, which means significant improvement over QG metrics does not guarantee significant better question annotation ability.

| | Data | Data Size | EM | F1 |
|---|---|---|---|---|
| Dev set | SQuAD | 87k | 72.52 | 81.79 |
| | + Beam5 | 399k | 74.33 | 83.19 |
| | + Beam10 | 706k | 74.44 | **83.23** |
| | + Beam15 | 853k | 74.25 | 82.75 |
| | + DivBeam10 | 595k | 74.44 | 83.30 |
| Dev set | + H1 | 142k | 73.96 | 82.81 |
| | + H2 | 255k | 74.19 | 82.84 |
| | + H4 | 424k | 74.42 | 82.82 |
| | + H6 | 506k | 74.27 | 82.97 |
| | + H8 | 705k | **74.64** | 83.14 |
| | + H10 | 930k | 74.27 | 82.97 |
| Test set | SQuAD | 87k | 71.92 | 81.26 |
| | + Beam10 | 706k | 73.43 | 82.39 |
| | + H8 | 705k | **73.61** | **82.53** |
| | + Beam10 + H8 | 1.3M | 73.43 | 82.11 |

Table 7: The results of our semi-supervised QA method using a BiDAF-QA model.

| Methods | New Data Size | EM | F1 |
|---|---|---|---|
| Dhingra et al. base | 0 | 71.54 | 80.69 |
| +Cloze | 3.2M | 71.86 | 80.80 |
| Our base | 0 | 72.19 | 81.52 |
| +Beam10 | 0 | 73.93 | 82.81 |
| +H8 | 705k | 74.12 | 82.83 |

Table 8: The comparison with the previous semi-supervised QA method. All results are the performance on the full development set of SQuAD, i.e., our QA test + development set.

### 6.2 Semi-Supervised Question Answering

**Effect of the data filter** As shown in Table 6, when using synthetic data only, adding the data filter can significantly improve QA performance. In terms of semi-supervised QA, the improvement is relatively smaller, due to the regularization from ground-truth data, but still consistent and stable.

**Semi-Supervised QA results** Table 7 demonstrates the semi-supervised QA results. Without introducing new articles, we keep beam search outputs as additional questions. It can be seen that using beam search with beam size 10 (+Beam10) improves the BiDAF-QA baseline by 1.51/1.13 absolute points on the testing set. With introducing new articles, the best performance (+H8) improves the BiDAF-QA baseline by 1.69/1.27 absolute points on the testing set. We also combine the two best settings (Beam10+H8), but it does not perform better than using them separately.

We conduct two ablation studies on the development set. First, we compare beam search with

[5] "Data Size" counts the total number of examples in training set (after filter). In Table 8, "New Data Size" only counts # examples generated from articles outside SQuAD.

| | BLEU4 | METEOR | ROUGE-L | Q-BLEU1 | QPP | QAP | QA-Eval (H1) |
|---|---|---|---|---|---|---|---|
| Du and Cardie (2018) | 15.16 | 19.12 | – | – | – | – | 55.11/66.40 |
| Our baseline (w. BERT) | 18.05 | 22.41 | 46.57 | 49.38 | 29.08 | 54.61 | 58.63/69.97 |
| + QPP | 18.51 | 22.87 | 46.65 | 49.97 | **30.14** | 55.67 | **60.49/71.81** |
| + QAP | **18.65** | **22.91** | **46.76** | **50.09** | 30.09 | **57.51** | 60.12/71.14 |
| + QPP & QAP | 18.58 | 22.87 | 46.76 | 50.01 | 30.10 | 56.39 | 59.11/70.87 |

Table 9: The performance of our stronger BERT-QG models.

| | Data | Data Size | EM | F1 |
|---|---|---|---|---|
| Dev set | SQuAD | 87k | 81.88 | 88.80 |
| | + Beam10 | 668k | 82.34 | 88.97 |
| | + H10 | 664k | **82.88** | **89.53** |
| Test set | SQuAD | 87k | 80.25 | 88.23 |
| | + Beam10 | 668k | 81.20 | 88.36 |
| | + H10 | 664k | 81.03 | **88.79** |
| | + Beam10 + H10 | 1.2M | **81.44** | 88.72 |

Table 10: The results of our semi-supervised QA method using a stronger BERT-QA model.

different beam sizes and diverse beam search (Li et al., 2016), but all of them perform similarly. Second, increasing the size of synthetic data from H1 to H10, the performance saturates around H2-H4. We also observed that when using a big synthetic data, e.g., H10, the model converges even before all examples were used for training. Based on these results, we conjecture that there is an upper bound of the effect of synthetic data which might be limited by the QG quality. To further improve the performance, more diverse and tricky questions need to be generated. To show how QG models help or limit the QA performance, we include some synthetic QA examples in Appendix. Finally, we compare our semi-supervised QA methods with Dhingra et al. (2018). As shown in Table 8, with no or less new data injection, our methods achieve larger improvements over a stronger baseline than their method.

### 6.3 QG and QA Results with BERT

The Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018) has recently improved a lot of NLP tasks by substantial margins. To verify if our improvements still hold on BERT-based baselines, we propose a BERT-QG baseline and test our two semantics-enhanced rewards; further, we conduct our semi-supervised QA method on a BERT-QA baseline.

**BERT-QG** Without modifying our QG model's architecture, we simply replaced ELMo used above with BERT. Table 9 shows that our BERT-QG baseline improves previous ELMo-QG baseline by a large margin; meanwhile, our QPP/QAP rewards significantly improve the stronger QG baseline and achieve the new state-of-the-art QG performance w.r.t both traditional metrics and QA-based QG evaluation. One difference is that the QAP-only model has the overall best performance instead of the multi-reward model. Note that, we also obtain the QPP and QAP rewards from BERT-based QPC and QA models, respectively.

**BERT-QA** Using our QAP-reinforced BERT-QG model, we apply the same semi-supervised QA method on a BERT-QA baseline. As shown in Table 10, though with smaller margins, our method improves the strong BERT-QA baseline by 1.19/0.56 absolute points on test set; even without introducing new articles, it obtains 0.95/0.13 absolute gains.

## 7 Conclusion

We proposed two semantics-enhanced rewards to regularize a QG model to generate semantically valid questions, and introduced a QA-based evaluation method that directly evaluates a QG model's ability to mimic human annotators in generating QA training data. Experiments showed that our QG model achieves new state-of-the-art performances. Further, we investigated how to use our QG system to augment QA datasets and conduct semi-supervised QA via two synthetic data generation methods along with a data filter and mixing mini-batch training. Experiments showed that our approach improves both BiDAF and BERT QA baselines even without introducing new articles.

## Acknowledgments

We thank the reviewers for their helpful comments and Hao Tan for useful discussions. This work was supported by DARPA (YFA17-D17AP00022), NSF-CAREER Award #1846185, ONR Grant #N00014-18-1-2871, and faculty awards from Google, Facebook, and Salesforce. The views contained in this article are those of the authors and not of the funding agency.

## References


Chris Alberti, Daniel Andor, Emily Pitler, Jacob Devlin, and Michael Collins. 2019. Synthetic qa corpora generation with roundtrip consistency. *arXiv preprint arXiv:1906.05416*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM.

Christopher Clark and Matt Gardner. 2018. Simple and effective multi-paragraph reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 845–855.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the ninth workshop on statistical machine translation*, pages 376–380.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Bhuwan Dhingra, Danish Danish, and Dheeraj Rajagopal. 2018. Simple and effective semi-supervised question answering. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 582–587.

Li Dong, Jonathan Mallinson, Siva Reddy, and Mirella Lapata. 2017. Learning to paraphrase for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 875–886.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. *arXiv preprint arXiv:1905.03197*.

Xinya Du and Claire Cardie. 2018. Harvesting paragraph-level question-answer pairs from wikipedia. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1907–1917.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1342–1352.

Nan Duan, Duyu Tang, Peng Chen, and Ming Zhou. 2017. Question generation for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 866–874, Copenhagen, Denmark. Association for Computational Linguistics.

Bradley Efron and Robert J Tibshirani. 1994. *An introduction to the bootstrap*. CRC press.

Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. 2018. Speaker-follower models for vision-and-language navigation. In *Advances in Neural Information Processing Systems*, pages 3314–3325.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In *Advances in Neural Information Processing Systems*, pages 820–828.

Michael Heilman and Noah A Smith. 2010. Good question! statistical ranking for question generation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 609–617. Association for Computational Linguistics.

Tom Hosking and Sebastian Riedel. 2019. Evaluating rewards for question generation models. *arXiv preprint arXiv:1902.11049*.

Unnat Jain, Ziyu Zhang, and Alexander Schwing. 2017. Creativity: Generating diverse questions using variational autoencoders. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5415–5424. IEEE.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611.

Yanghoon Kim, Hwanhee Lee, Joongbo Shin, and Kyomin Jung. 2018. Improving neural question generation using answer separation. *arXiv preprint arXiv:1809.02393*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Vishwajeet Kumar, Ganesh Ramakrishnan, and Yuan-Fang Li. 2018. A framework for automatic question generation from text using deep reinforcement learning. *arXiv preprint arXiv:1808.04961*.

Igor Labutov, Sumit Basu, and Lucy Vanderwende. 2015. Deep questions without deep understanding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 889–898.

Mike Lewis and Angela Fan. 2019. Generative question answering: Learning to answer the whole question. In *International Conference on Learning Representations (ICLR)*.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. A simple, fast diverse decoding algorithm for neural generation. *arXiv preprint arXiv:1611.08562*.

Yikang Li, Nan Duan, Bolei Zhou, Xiao Chu, Wanli Ouyang, Xiaogang Wang, and Ming Zhou. 2018. Visual question generation as dual task of visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6116–6124.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

David Lindberg, Fred Popowich, John Nesbit, and Phil Winne. 2013. Generating natural language questions to support learning on-line. In *Proceedings of the 14th European Workshop on Natural Language Generation*, pages 105–114.

Bang Liu, Mingjun Zhao, Di Niu, Kunfeng Lai, Yancheng He, Haojie Wei, and Yu Xu. 2019. Learning to generate questions by learning what not to generate. *arXiv preprint arXiv:1902.10418*.

Preksha Nema and Mitesh M Khapra. 2018. Towards a better metric for evaluating question generation systems. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3950–3959.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Ramakanth Pasunuru and Mohit Bansal. 2018. Multi-reward reinforced summarization with saliency and entailment. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 646–653.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Sathish Reddy, Dinesh Raghu, Mitesh M Khapra, and Sachindra Joshi. 2017. Generating natural language question-answer pairs from a knowledge graph using a rnn based question generation model. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 376–385.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1179–1195. IEEE.

Mrinmaya Sachan and Eric Xing. 2018. Self-training for jointly learning to ask and answer questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 629–640.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 86–96.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Iulian Vlad Serban, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. 2016. Generating factoid questions with recurrent neural networks:

The 30m factoid question-answer corpus. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 588–598.

Linfeng Song, Zhiguo Wang, and Wael Hamza. 2017. A unified query-based generative model for question generation and question answering. *arXiv preprint arXiv:1709.01058*.

Linfeng Song, Zhiguo Wang, Wael Hamza, Yue Zhang, and Daniel Gildea. 2018. Leveraging context information for natural question generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 569–574.

Sandeep Subramanian, Tong Wang, Xingdi Yuan, Saizheng Zhang, Adam Trischler, and Yoshua Bengio. 2018. Neural models for key phrase extraction and question generation. In *Proceedings of the Workshop on Machine Reading for Question Answering*, pages 78–88.

Xingwu Sun, Jing Liu, Yajuan Lyu, Wei He, Yanjun Ma, and Shi Wang. 2018. Answer-focused and position-aware neural question generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3930–3939.

Hao Tan, Licheng Yu, and Mohit Bansal. 2019. Learning to navigate unseen environments: Back translation with environmental dropout. *arXiv preprint arXiv:1904.04195*.

Duyu Tang, Nan Duan, Tao Qin, Zhao Yan, and Ming Zhou. 2017. Question answering and question generation as dual tasks. *arXiv preprint arXiv:1706.02027*.

Duyu Tang, Nan Duan, Zhao Yan, Zhirui Zhang, Yibo Sun, Shujie Liu, Yuanhua Lv, and Ming Zhou. 2018. Learning to collaborate for question answering and asking. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1564–1574.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 189–198.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Zhilin Yang, Junjie Hu, Ruslan Salakhutdinov, and William Cohen. 2017. Semi-supervised qa with generative domain-adaptive nets. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1040–1050.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

Xingdi Yuan, Tong Wang, Caglar Gulcehre, Alessandro Sordoni, Philip Bachman, Saizheng Zhang, Sandeep Subramanian, and Adam Trischler. 2017. Machine comprehension by text-to-text neural question generation. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 15–25.

Yao Zhao, Xiaochuan Ni, Yuanyuan Ding, and Qifa Ke. 2018. Paragraph-level neural question generation with maxout pointer and gated self-attention networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3901–3910.

Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. 2017. Neural question generation from text: A preliminary study. In *National CCF Conference on Natural Language Processing and Chinese Computing*, pages 662–671. Springer.

Zhi-Hua Zhou and Ming Li. 2005. Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge and Data Engineering*, 17(11):1529–1541.

# Appendix

## A Experiment Setup

### A.1 Dataset

**QG** For QG, we use the SQuAD-based QG dataset[6] first introduced by Du et al. (2017) which was the most widely-used QG dataset in previous works (Song et al., 2018; Zhao et al., 2018; Du and Cardie, 2018; Kim et al., 2018; Sun et al., 2018). It was derived from SQuADv1.1 (Rajpurkar et al., 2016). Since the testing set is not open, they sampled 10% articles from the training set as the testing set, and the original development set is still used for validation.

For the QA-based QG evaluation, we obtain new paragraphs with pre-extracted answer spans from HarvestingQA (Du and Cardie, 2018). Without using their provided questions, we have different QG models act as "annotators" to generate questions, and then use the different QG-labeled synthetic datasets to train QA models. We use the same dev-test setup as described below.

[6] https://github.com/xinyadu/nqg/tree/master/data

**QA** For QA, we use SQuADv1.1 (Rajpurkar et al., 2016). Previous semi-supervised QA works sampled 10% from training set as the testing set (Yang et al., 2017; Dhingra et al., 2018). Since we want to use the full training set in semi-supervised QA setup without any data size reduction, we instead split the original development set in half for validation and testing respectively.

For semi-supervised QA, first, without introducing new articles, we generate new questions for SQuAD training set by keeping all beam search outputs. Second, with introducing new articles, we obtain new paragraphs with pre-extracted answer spans from HarvestingQA (Du and Cardie, 2018). Without using their provided questions, we use our best QG model to label questions. Meanwhile, we investigate the influence of synthetic data size, so we sample 10% to 100% examples from HarvestingQA, which are denoted as H1-H10 in our experiments.

### A.2 Evaluation Metrics

**QG** First, we use three traditional automatic evaluation metrics: BLEU4 (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2014), ROUGE-L (Lin, 2004). Second, we adopt the new "Q-metrics" proposed by Nema and Khapra (2018), and we only use "Q-BLEU1" that was shown to have the highest correlation with human judgments on SQuAD. We also take the QPP and QAP rewards as two additional evaluation metrics. Further, we conduct a pairwise human comparison between our baseline and best QG models. Detailed human evaluation setup is described in the next section. For the QA-based QG evaluation, we use the same QA evaluation metrics as follows.

**QA** Following the standard evaluation method for SQuADv1.1 (Rajpurkar et al., 2016), we use Exact Match (EM) and F1 as two metrics.

### A.3 Human Evaluation

We performed pairwise human evaluation between our baseline and the QPP&QAP multi-reward model on Amazon Mechanical Turk. We selected human annotators that are located in the US, have an approval rate greater than 98%, and have at least 10,000 approved HITs. We showed the annotators an input paragraph with the answer bold in the paragraph and two questions generated by two QG models (randomly shuffled to anonymize model identities). We then asked them to decide which one is better or choose "non-distinguishable" if they are equally good/bad. We give human three instructions about what is a good question: first, "answerability" – a good question should be answerable by the given answer; "making sense" – a good question should be making sense given the surrounding context; "overall quality" – a good question should be as fluent, non-ambiguous, semantically compact as possible. Ground-truth questions were not provided to avoid simple matching with ground-truth.

## B Implementation Details

**QG** For ELMo-QG, we first tokenize and obtain the POS/NER tags by Standford Corenlp toolkit[7], then lower-case the entire dataset. We use 2-layer LSTM-RNNs for both encoder and decoder with hidden size 600. Dropout with a probability of 0.3 is applied to the input of each LSTM-RNN layer. We use the pre-trained character-level word embedding from ELMo (Peters et al., 2018) both as our word embedding and output-projection matrix, and keep it fixed. We use Adam (Kingma and Ba, 2014) as optimizer with learning rate 0.001 for teacher forcing and 0.00001 for reinforcement learning. Batch size is set to 32. For stability, we first pre-train the model with teacher forcing until convergence, then fine-tune it with the mixed loss. Hyper-parameters are tuned on development set: $\gamma^{qpp} = 0.99$, $\gamma^{qap} = 0.97$, and $n : m = 3 : 1$. We use beam search with beam size 10 for decoding and apply a bi-gram/tri-gram repetition penalty as proposed in Paulus et al. (2017).

For BERT-QG, we simply replace the ELMo used above to BERT (Devlin et al., 2018). To match with BERT's tokenization, we use the WordPiece tokenizer to tokenize each word obtained above and extend the POS/NER tags to each word piece. Decoder's word-piece outputs will be mapped to normal words by post-processing. Hyper-parameters are tuned on development set: $\gamma^{qpp} = 0.99$, $\gamma^{qap} = 0.97$, and $n : m = 1 : 3$.

**QA** For BiDAF-QA, we implement the BiDAF+Self-attention architecture proposed by Clark and Gardner (2018). We use GRUs for all RNN layers with hidden size 90 for GRUs and 180 for linear layers. Dropout with a probability of 0.2 is applied to the input of each GRU-RNN layer. We optimize the model using Adadelta with

[7] https://stanfordnlp.github.io/CoreNLP/

| Examples generated on SQuAD |
|---|
| Context: ...new york city consists of **five** boroughs, each of which is a separate county of new york state...<br>Ground-truth: how many boroughs does new york city contain ?<br>ELMo-QG: how many boroughs make up new york city ?<br>BERT-QG: new york city consists of how many boroughs ? |
| Context: ...gendn gyatso traveled in exile looking for allies. however, it was not until **1518** that the secular phagmodru ruler captured lhasa from the rinbung, and thereafter the gelug was given rights to conduct the new years prayer...<br>Ground-truth: when was gelug was given the right to conduct the new years prayer ?<br>ELMo-QG: in what year did the secular phagmodru ruler take over lhasa ?<br>BERT-QG: when did the secular phagmodru ruler capture lhasa from the rinbung ? |
| Context: ...chopin attended the lower rhenish music festival in aix-la-chapelle with hiller, and it was there that chopin met felix mendelssohn. after the festival, the three visited dsseldorf... they spent what mendelssohn described as "a very agreeable day", **playing and discussing music** at his piano...<br>Ground-truth: what two activities did frdric do while visiting for a day in dsseldorf with mendelssohn and hiller ?<br>ELMo-QG: what did mendelssohn do at his piano ?<br>BERT-QG: what did chopin do at his piano ? |
| Context: ...to limit protests, officials pushed parents to sign a document, which forbade them from holding protests, in exchange of money, but some who refused to sign **were threatened**...<br>Ground-truth: what has happened to some who refuse to agree to not protest ?<br>ELMo-QG: what did some who refused to sign ?<br>BERT-QG: what did the officials refused to sign ? |
| Examples generated on HarvestingQA |
| Context: ...nigeria prior to independence was faced with sectarian tensions and violence... some of the ethnic groups like the ogoni, have experienced severe environmental degradation due to **petroleum extraction**...<br>Du and Cardie (2018): what is the main reason for the ethnic groups ?<br>ELMo-QG: why has nigeria experienced severe environmental degradation ?<br>BERT-QG: why have the ogoni experienced severe environmental degradation ? |
| Context: ...vietnam is located on the eastern indochina peninsula... at its narrowest point in the central **qung bnh province**, the country is as little as across...<br>Du and Cardie (2018): where is the country 's country located ?<br>ELMo-QG: in what province is vietnam located ?<br>BERT-QG: what province is vietnam 's narrowest point ? |
| Context: ...the ottoman islamic legal system was set up differently from **traditional european courts**...<br>Du and Cardie (2018): where was the ottoman islamic legal system set ?<br>ELMo-QG: the ottoman islamic legal system was set up from what ?<br>BERT-QG: what was the ottoman islamic legal system set up differently from ? |
| Context: ...the eastern shore of virginia is the site of **wallops flight facility**, a rocket testing center owned by nasa...<br>Du and Cardie (2018): what is the eastern shore of virginia owned by ?<br>ELMo-QG: what facility is owned by nasa ?<br>BERT-QG: what is the name of the rocket facility located by nasa ? |

Table 11: Some synthetic QA examples generated by our QG models.

batch size 64. We also add ELMo to both the input and output of the contextual GRU-RNN layer as proposed in (Peters et al., 2018). To match with QG model's setup, we also apply lower-case on QA datasets.

For BERT-QA, we use the pre-trained uncased BERT-base model[8] and fine-tune it on QA datasets.

**QPC** For ELMo-QPC, we follow the model architecture proposed by Conneau et al. (2017). First, two input questions are embedded with ELMo (Peters et al., 2018). Second, the embedded questions are encoded by two 2-layer bidirectional LSTM-RNNs separately with hidden size 512. Next, a max-pooling layer outputs the sentence embedding of each question, denoted by $q_1$ and $q_2$. Lastly, we input $[q_1, q_2, |q_1 - q_2|, q_1 * q_2]$ to an MLP to predict whether these two questions are paraphrases or not. This QPC model is trained using the Quora Question Pairs[9] dataset. We use Adam (Kingma and Ba, 2014) as optimizer with learning rate 0.0004 and batch size 64. This model obtained 86% accuracy on QQP development set.

For BERT-QPC, we also use the pre-trained uncased BERT-base model and fine-tune it on QQP dataset, which obtained 90% accuracy on QQP development set.

[8] https://github.com/google-research/bert

[9] https://tinyurl.com/y2y8u5ed

## C Examples

Table 11 shows some synthetic QA examples generated by our QG models. On SQuAD, the first two examples show our QG models generate some paraphrases or novel questions that enrich the dataset; the last two examples show our QG models generate easier or wrong questions that limit the semi-supervised QA's performance. On HarvestingQA, our QG models can output better questions than Du and Cardie (2018) did but still generate some wrong questions.

# Adversarial Domain Adaptation for Machine Reading Comprehension

**Huazheng Wang**[1]*, **Zhe Gan**[2], **Xiaodong Liu**[3], **Jingjing Liu**[2], **Jianfeng Gao**[3], **Hongning Wang**[1]

[1]University of Virginia, [2]Microsoft Dynamics 365 AI Research, [3]Microsoft Research

{hw7ww,hw5x}@virginia.edu, {zhe.gan,xiaodl,jingjl,jfgao}@microsoft.com

## Abstract

In this paper, we focus on unsupervised domain adaptation for Machine Reading Comprehension (MRC), where the source domain has a large amount of labeled data, while only unlabeled passages are available in the target domain. To this end, we propose an Adversarial Domain Adaptation framework (AdaMRC), where ($i$) pseudo questions are first generated for unlabeled passages in the target domain, and then ($ii$) a domain classifier is incorporated into an MRC model to predict which domain a given passage-question pair comes from. The classifier and the passage-question encoder are jointly trained using adversarial learning to enforce domain-invariant representation learning. Comprehensive evaluations demonstrate that our approach ($i$) is generalizable to different MRC models and datasets, ($ii$) can be combined with pre-trained large-scale language models (such as ELMo and BERT), and ($iii$) can be extended to semi-supervised learning.

## 1 Introduction

Recently, many neural network models have been developed for Machine Reading Comprehension (MRC), with performance comparable to human in specific settings (Gao et al., 2019). However, most state-of-the-art models (Seo et al., 2017; Liu et al., 2018; Yu et al., 2018) rely on large amount of human-annotated in-domain data to achieve the desired performance. Although there exists a number of large-scale MRC datasets (Rajpurkar et al., 2016; Trischler et al., 2016; Bajaj et al., 2016; Zhang et al., 2018), collecting such high-quality datasets is expensive and time-consuming, which hinders real-world applications for domain-specific MRC.

Therefore, the ability to transfer an MRC model trained in a high-resource domain to other low-resource domains is critical for scalable MRC. While it is difficult to collect annotated question-answer pairs in a new domain, it is generally feasible to obtain a large amount of unlabeled text in a given domain. In this work, we focus on adapting an MRC model trained in a source domain to other new domains, where only unlabeled passages are available.

This domain adaptation issue has been a main challenge in MRC research, and the only existing work that investigated this was the two-stage synthesis network (SynNet) proposed in Golub et al. (2017). Specifically, SynNet first generates pseudo question-answer pairs in the target domain, and then uses the generated data as augmentation to fine-tune a pre-trained MRC model. However, the source-domain labeled data and target-domain pseudo data are directly combined without considering domain differences (see Figure 1(a), where the two feature distributions in two domains are independently clustered). Directly transfering a model from one domain to another could be counter-effective, or even hurt the performance of the pre-trained model due to domain variance.

To achieve effective domain transfer, we need to learn features that are discriminative for the MRC task in the source domain, while simultaneously indiscriminating with respect to the shift between source and target domains. Motivated by this, we propose *Adversarial Domain Adaptation for MRC* (AdaMRC), a new approach that utilizes adversarial learning to learn domain-invariant transferable representations for better MRC model adaptation across domains (see Figure 1(b), where the two feature distributions learned by AdaMRC are indistinguishable through adversarial learning).

Specifically, our proposed method first generates synthetic question-answer pairs given pas-

*Most of this work was done when the first author was an intern at Microsoft Dynamics 365 AI Research.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2510–2520, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

sages in the target domain. Different from Golub et al. (2017), which only used pseudo question-answer pairs to fine-tune pre-trained MRC models, our AdaMRC model uses the passage and the generated pseudo-questions in the target domain, in addition to the human-annotated passage-question pairs in the source domain, to train an additional *domain classifier* as a discriminator. The passage-question encoder and the domain classifier are jointly trained via adversarial learning. In this way, the encoder is enforced to learn domain-invariant representations, which are beneficial for transferring knowledge learned from one domain to another. Based on this, an answer decoder is then used to decode domain-invariant representation into an answer span.

The proposed approach is validated on a set of popular benchmarks, including SQuAD (Rajpurkar et al., 2016), NewsQA (Trischler et al., 2016), and MS MARCO (Bajaj et al., 2016), using state-of-the-art MRC models including SAN (Liu et al., 2018) and BiDAF (Seo et al., 2017). Since pre-trained large-scale language models, such as ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019), have shown strong performance to learn representations that are generalizable to various tasks, in this work, to further demonstrate the versatility of the proposed model, we perform additional experiments to demonstrate that AdaMRC can also be combined with ELMo and BERT to further boost the performance.

The main contributions of this paper are summarized as follows: $(i)$ We propose AdaMRC, an adversarial domain adaptation framework that is specifically designed for MRC. $(ii)$ We perform comprehensive evaluations on several benchmarks, demonstrating that the proposed method is generalizable to different MRC models and diverse datasets. $(iii)$ We demonstrate that AdaMRC is also compatible with ELMo and BERT. $(iv)$ We further extend the proposed framework to semi-supervised learning, showing that AdaMRC can also be applied to boost the performance of a pre-trained MRC model when a small amount of labeled data is available in the target domain.

## 2 Related Work

**Machine Reading Comprehension** The MRC task has recently attracted a lot of attention in the community. An MRC system is required to answer a question by extracting a text snippet within a given passage as the answer. A large number of deep learning models have been proposed to tackle this task (Seo et al., 2017; Xiong et al., 2017; Shen et al., 2017; Liu et al., 2018; Yu et al., 2018). However, the success of these methods largely relies on large-scale human-annotated datasets (such as SQuAD (Rajpurkar et al., 2016), NewsQA (Trischler et al., 2016) and MS MARCO (Bajaj et al., 2016)).

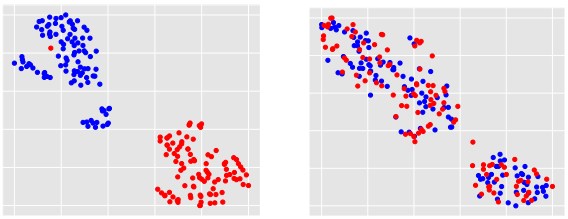

(a) SynNet (b) AdaMRC

Figure 1: t-SNE plot of encoded feature representations from (a) SynNet (Golub et al., 2017) and (b) the proposed AdaMRC. We sampled 100 data points, each from the development set of the source and the target domains. Blue: SQuAD. Red: NewsQA.

Different from previous work that focused on improving the state of the art on particular MRC datasets, we study the MRC task from a different angle, and aim at addressing a critical yet challenging problem: how to transfer an MRC model learned from a high-resource domain to other low-resource domains in an unsupervised manner.

Although important for the MRC task, where annotated data are limited in real-life applications, this problem has not yet been well investigated. There were some relevant studies along this line. For example, Chung et al. (2018) adapted a pre-trained model to TOEFL and MCTest dataset, and Wiese et al. (2017) applied transfer learning to the biomedical domain. However, both studies assumed that annotated data in the target domain (either questions or question-answer pairs) are available.

To the best of our knowledge, SynNet (Golub et al., 2017) is the only work that also studied domain adaptation for MRC. Compared with SynNet, the key difference in our model is adversarial learning, which enables domain-invariant representation learning for better model adaptation to low-resource domains. Our approach is also related to multi-task learning (Xu et al., 2019; Caruana, 1997; Liu et al., 2015, 2019) and semi-supervised learning (Yang et al., 2017) for MRC.

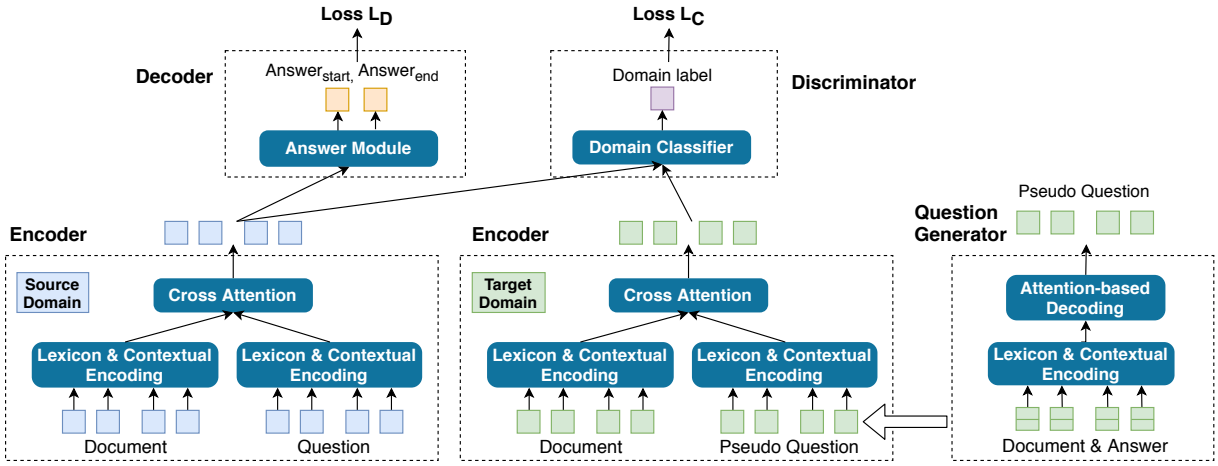


Figure 2: Illustration of the proposed AdaMRC model for unsupervised domain adaptation of MRC.

In this work, we focus on purely unsupervised domain adaptation.

**Domain Adaptation** Domain adaptation aims to make a machine learning model generalizable to other domains, especially without any annotated data in the target domain (or with only limited data) (Ganin and Lempitsky, 2015). One line of research on domain adaptation focuses on transiting the feature distribution from the source domain to the target domain (Gong et al., 2012; Long et al., 2015). Another school of research focuses on learning domain-invariant representations (Glorot et al., 2011) (e.g., via adversarial learning (Ganin et al., 2016; Tzeng et al., 2017)).

Domain adaptation has been successfully applied to many tasks, such as image classification (Tzeng et al., 2017), speech recognition (Doulaty et al., 2015), sentiment classification (Ganin et al., 2016; Li et al., 2017), machine translation (Johnson et al., 2017; Zoph et al., 2016), relation extraction (Fu et al., 2017), and paraphrase identification (Shah et al., 2018). Compared to these areas, the application to MRC presents additional challenges, since besides missing labeled data (*i.e.*, answer spans), the questions in the target domain are also unavailable. To our best knowledge, we are the first to investigate the usage of adversarial domain adaptation for the MRC task.

There are many prevailing unsupervised techniques for domain adaptation. Our proposed approach is inspired by the seminal work of Ganin et al. (2016) to validate its potential of solving domain adaptation problem on a new task, without any supervision for the target domain. There are also other more advanced methods, such as MMD-based adaptation (Long et al., 2017), residual transfer network (Long et al., 2016), and maximum classifier discrepancy (Saito et al., 2018) that can be explored for future work.

## 3 Problem Definition

The problem of unsupervised domain adaptation for MRC is defined as follows. First, let $S = \{p^s, q^s, a^s\}$ denote a labeled MRC dataset from the source domain $s$, where $p^s, q^s$ and $a^s$ represent the passage, the question, and the answer of a sample, respectively. An MRC model $M^s$, taking as input the passage $p^s = (p_1, p_2, ..., p_T)$ of length $T$ and the question $q^s = (q_1, q_2, ..., q_{T'})$ of length $T'$, is trained to predict the correct answer span $a^s = (a^s_{start}, a^s_{end})$, where $a^s_{start}, a^s_{end}$ represent the starting and ending indexes of the answer in the passage $p^s$.

We assume that only unlabeled passages are available in the target domain $t$, *i.e.*, $T = \{p^t\}$, where $p^t$ represents a passage. This is a reasonable assumption as it is easy to collect a large amount of unlabeled passages in a new domain. Given datasets $S$ and $T$, the goal of unsupervised domain adaptation is defined as learning an MRC model $M^t$ based on $S$ and $T$ to answer questions in the target domain $t$.

## 4 AdaMRC

As illustrated in Figure 2, AdaMRC consists of three main components: $(i)$ *Question Generator* (Sec. 4.1), where pseudo question-answer pairs are generated given unlabeled passages in the target domain; $(ii)$ *MRC Module* (Sec. 4.2), where

given an input document and a question, an answer span is extracted from the document; (*iii*) *Domain Classifier* (Sec. 4.3), where a domain label is predicted to distinguish a feature vector from either the source domain or the target domain.

Specifically, the MRC module is composed of an encoder and a decoder. The encoder with parameter $\theta_e$ embeds the input passage and the question into a feature vector. The decoder with parameter $\theta_d$ takes the feature vector as input to predict the answer span. The domain classifier with parameter $\theta_c$ takes the same feature vector as input to classify the domain label. All the parameters $(\theta_e, \theta_d, \theta_c)$ are jointly optimized, with the objective of training the encoder to correctly predict the answer span, but also simultaneously fool the domain classifier. In other words, the encoder learns to map text input into a feature space that is invariant to the switch of domains. The following sub-sections describe each module, with training details provided in Sec. 4.4.

### 4.1 Question Generation

First, we use an NER system to extract possible answer spans $a^t$ from the passages $p^t$ in the target domain, under the assumption that any named entity could be the potential answer of certain questions. Similar answer extraction strategy has been applied in Yang et al. (2017) in a semi-supervised-learning setting, while Golub et al. (2017) proposed to train an answer synthesis network to predict possible answers spans. We tried both methods, and empirically observed that a simple NER system provides more robust results, which is used in our experiments.

Now, we describe how the question generation (QG) model is trained. Given the passage $p^s = (p_1, p_2, ..., p_T)$ and answer $a^s = (a_{start}, a_{end})$ from the source domain, the QG model with parameter $\theta_{QG}$ learns the conditional probability of generating a question $q^s = (q_1, q_2, ..., q_{T'})$, *i.e.*, $P(q^s|p^s, a^s)$. We implement the QG model as a sequence-to-sequence model with attention mechanism (Bahdanau et al., 2015), and also apply the copy mechanism proposed in Gu et al. (2016); Gulcehre et al. (2016) to handle rare/unknown words.

Specifically, the QG model consists of a lexicon encoding layer, a BiLSTM contextual encoding layer, and an LSTM decoder. For lexicon encoding, each word token $p_i$ of a passage is mapped into a concatenation of GloVe vectors (Pennington et al., 2014), part-of-speech (POS) tagging embedding, and named-entity-recognition (NER) embedding. We further insert answer information by appending an additional zero/one feature (similar to Yang et al. (2017)) to model the appearance of answer tokens in the passage. The output of the lexicon encoding layer is appended with CoVe vectors (McCann et al., 2017), and then passed to the Bidirectional LSTM contextual encoding layer, producing a sequence of hidden states. The decoder is another LSTM with attention and copy mechanism over the encoder hidden states. At each time step, the generation probability of a question token $q_t$ is defined as:

$$P(q_t) = g_t P^v(q_t) + (1 - g_t) P^{copy}(q_t)\,, \quad (1)$$

where $g_t$ is the probability of generating a token from the vocabulary, while $(1 - g_t)$ is the probability of copying a token from the passage. $P^v(q_t)$ and $P^{copy}(q_t)$ are defined as softmax functions over the words in the vocabulary and over the words in the passage, respectively. $g_t$, $P^v(q_t)$ and $P^{copy}(q_t)$ are functions of the current decoder hidden state.

### 4.2 MRC Module

**Encoder** The encoder in the MRC module contains lexicon encoding and contextual encoding, similar to the encoder used in the question generation module. It also includes a cross-attention layer for fusion. Specifically, the output of the lexicon encoder is appended with the CoVe vector and passed to the contextual encoding layer, which is a 2-layer Bidirectional LSTM that produces hidden states of the passage $H^p \in \mathbb{R}^{T \times 2m}$ and the question $H^q \in \mathbb{R}^{T' \times 2m}$, where $m$ is the hidden size of the BiLSTM. We then use cross attention to fuse $H^p$ and $H^q$, and construct a working memory of passage $M^p \in \mathbb{R}^{T \times 2m}$ (see Liu et al. (2018) for more details). The question memory $M^q \in \mathbb{R}^{2m}$ is constructed by applying self-attention on $H^q$.

**Decoder** The decoder, or answer module, predicts an answer span $a = (a_{start}, a_{end})$ given a passage $p$ and a question $q$, by modeling the conditional probability $P(a|p, q)$. The initial state $s_0$ is set as $M^q$. Through $T$ steps, a GRU (Cho et al., 2014) is used to generate a sequence of state vectors $s_t = \text{GRU}(s_{t-1}, x_t)$, where $x_t$ is computed via attention between $M^p$ and $s_{t-1}$. Two softmax layers are used to compute the distribution of

**Algorithm 1** AdaMRC training procedure.

**Input:** source domain labeled data $S = \{p^s, q^s, a^s\}$, target domain unlabeled data $T = \{p^t\}$
Train the MRC model $\theta^s = (\theta_e^s, \theta_d^s)$ on source domain $S$;
Train the QG model $\theta_{QG}$ on source domain $S$;
Generate $T_{gen} = \{p^t, q^t, a^t\}$ using the QG model;
Initialize $\theta = (\theta_e, \theta_d, \theta_c)$ with $\theta^s$;
**for** epoch $\leftarrow$ 1 to #epochs **do**
    Optimize $\theta$ on $S \cup T_{gen}$. Each minibatch is composed with $k_s$ samples from $S$ and $k_t$ samples from $T_{gen}$;
**end for**
**Output:** Model with the best performance on the target development set $\theta^*$.

the start and the end of the answer span at each step given $s_t$, and the final prediction is the average prediction of all steps. Stochastic prediction dropout (Liu et al., 2018) is applied during training.

Note that we use SAN as an example MRC model in the proposed framework. However, our approach is compatible with any existing MRC models. In experiments, in order to demonstrate the versatility of the proposed model, we also conduct experiments with BiDAF (Seo et al., 2017).

### 4.3 Domain Classifier

The domain classifier takes the output of the encoder as input, including the aforementioned passage representation $M^p \in \mathbb{R}^{T \times 2m}$ and the self-attended question representation $M^q \in \mathbb{R}^{2m}$ from different domains, and predicts the domain label $d$ by modeling the conditional probability $P(d|p, q)$. A self-attention layer is also applied to $M^p$ to reduce its size to $M^{p'} \in \mathbb{R}^{2m}$. We then concatenate it with $M^q$, followed by a two-layer Multi-Layer Perceptron (MLP), $f(W[M^{p'}; M^q])$, and use a sigmoid function to predict the domain label.

### 4.4 Training

Algorithm 1 illustrates the training procedure of our proposed framework. We first train the question generation model $\theta_{QG}$ on the source domain dataset $S$ by maximizing the likelihood of generating question $q^s$ given passage $p^s$ and answer $a^s$. Given the unlabeled dataset in the target domain, we extract candidate answers $a^t$ on $p^t$ and use $\theta_{QG}$ to generate pseudo questions $q^t$, and then compose a pseudo labeled dataset $T_{gen} = \{p^t, q^t, a^t\}$.

We initialize the MRC model $\theta$ for the target domain with the pre-trained MRC model $\theta^s$ from the source domain, and then fine-tune the model using both the source domain dataset $S$ and the target domain dataset $T_{gen}$. The goal of the decoder $\theta_d$ is to predict $P(a|p, q)$. The objective function is denoted as:

$$L_D(\theta_e, \theta_d) = \frac{1}{|S|} \sum_{i=1}^{|S|} \log P(a^{(i)}|p^{(i)}, q^{(i)}) , \quad (2)$$

where the superscript $(i)$ indicates the $i$-th sample. It is worthwhile to emphasize that unlike Golub et al. (2017), we only use source domain data to update the decoder, without using pseudo target domain data. This is because the synthetic question-answer pairs could be noisy, and directly using such data for decoder training may lead to degraded performance of the answer module, as observed both in Sachan and Xing (2018) and in our experiments.

The synthetic target domain data and source domain data are both used to update the encoder $\theta_e$ and the domain classifier $\theta_c$. The classifier predicts a domain label $d$ given the feature representation from the encoder. The objective function is:

$$L_C(\theta_e, \theta_c) = \frac{1}{N} \sum_{i=1}^{N} \log P(d^{(i)}|p^{(i)}, q^{(i)}) , \quad (3)$$

where $N = |S| + |T_{gen}|$. In order to learn *domain-invariant* representations from the encoder, we update $\theta_e$ to *maximize* the loss while updating $\theta_c$ to minimize the loss in an adversarial fashion. The overall objective function is defined as:

$$L(\theta_e, \theta_d, \theta_c) = L_D(\theta_e, \theta_d) - \lambda L_C(\theta_e, \theta_c) , \quad (4)$$

where $\lambda$ is a trade-off parameter that balances the two terms.

To optimize our model, instead of alternately updating the adversaries like in GAN (Goodfellow et al., 2014), we use the gradient-reversal layer (Ganin and Lempitsky, 2015) to jointly optimize all the components, as suggested in Chen et al. (2018).

## 5 Experiments

### 5.1 Experimental Setting

**Datasets** We validate our proposed method on three benchmarks: SQuAD (Rajpurkar et al.,

| Dataset | Domain | Train | Dev | Test |
|---|---|---|---|---|
| SQuAD (v1.1) | Wiki | 87,600 | 10,570 | – |
| NewsQA | News | 92,549 | 5,166 | 5,165 |
| MS MARCO (v1) | Web | 82,430 | 10,047 | 9,650 |

Table 1: Statistics of the datasets.

2016), NewsQA (Trischler et al., 2016), and MS MARCO (Bajaj et al., 2016). The statistics of the datasets are provided in Table 1. Note that these datasets are all from different domains: SQuAD is from *Wikipedia*; NewsQA is from *CNN news*; and MS MARCO is from *web search log*.

**Evaluation metrics** For SQuAD and NewsQA, we report results on two evaluation metrics: Exact Match (**EM**), which measures the percentage of span predictions that match any of the ground truth answers exactly; and Macro-averaged **F1** score, which measures the average overlap between the prediction and the ground-truth answer. For MS MARCO, since the answer is free-formed, we use BLEU and ROUGE-L scores for evaluation.

**Implementation details**[1] We use spaCy[2] to generate POS and NER taggings, which are used in answer extraction and the lexicon encoding layer of the QG and MRC models. The QG model is fixed after trained on source-domain labeled data. The hidden size of the LSTM in the QG model is set to 125. Parameters of the SAN model follow Liu et al. (2018). The hidden size of the MLP layer in the domain classifier is set to 125. Both the QG and the MRC model are optimized via Adamax (Kingma and Ba, 2015) with mini-batch size set to 32. The learning rate is set to 0.002 and is halved every 10 epochs. To avoid overfitting, we set the dropout rate to 0.3. For each mini-batch, data are sampled from both domains, with $k_s$ samples from the source domain and $k_t$ samples from the target domain. We set $k_s : k_t = 2 : 1$ as default in our experiments. For the trade-off parameter $\lambda$, we gradually change it from 0 to 1, following the schedule suggested in Ganin and Lempitsky (2015).

### 5.2 Experimental Results

We implement the following baselines and models for comparison.

1. **SAN**: we directly apply the pre-trained SAN model from the source domain to answer questions in the target domain.
2. **SynNet+SAN**: we use SynNet[3] (Golub et al., 2017) to generate pseudo target-domain data, and then fine-tune the pre-trained SAN model.
3. **AdaMRC**: as illustrated in Algorithm 1.
4. **AdaMRC with GT questions**: the same as AdaMRC, except that the ground-truth questions in the target domain are used for training. This serves as an upper-bound of the proposed model.

| Method | EM/F1 |
|---|---|
| SQuAD → NewsQA | |
| SAN | 36.68/52.79 |
| SynNet + SAN | 35.19/49.61 |
| AdaMRC | **38.46/54.20** |
| AdaMRC with GT questions | 39.37,54.63 |
| NewsQA → SQuAD | |
| SAN | 56.83/68.62 |
| SynNet + SAN | 50.34/62.42 |
| AdaMRC | **58.20/69.75** |
| AdaMRC with GT questions | 58.82/70.14 |
| SQuAD → MS MARCO (BLEU-1/ROUGE-L) | |
| SAN | 13.06/25.80 |
| SynNet + SAN | 12.52/25.47 |
| AdaMRC | **14.09/26.09** |
| AdaMRC with GT questions | 15.59/26.40 |
| MS MARCO → SQuAD | |
| SAN | 27.06/40.07 |
| SynNet + SAN | 23.67/36.79 |
| AdaMRC | **27.92/40.69** |
| AdaMRC with GT questions | 27.79/41.47 |

Table 2: Performance of AdaMRC compared with baseline models on three datasets, using SAN as the MRC model.

Table 2 summarizes the experimental results. We observe that the proposed method consistently outperforms SAN and the SynNet+SAN model on all datasets. For example, in the SQuAD→NewsQA setting, where the source-domain dataset is SQuAD and the target-domain dataset is NewsQA, AdaMRC achieves 38.46% and 54.20% in terms of EM and F1 scores, outperforming the pre-trained SAN by 1.78% (EM) and 1.41% (F1), respectively, as well as surpassing SynNet by 3.27% (EM) and 4.59% (F1), respectively. Similar improvements are also observed in NewsQA→SQuAD, SQuAD→MS MARCO and MS MARCO→SQuAD settings, which demonstrates the effectiveness of the proposed model.

Interestingly, we find that the improvement on adaptation between SQuAD and NewsQA is greater than that between SQuAD and MS MARCO. Our assumption is that it is because

[1] Code will be released for easy access.

[2] https://spacy.io/

[3] The officially released code is used in our experiments: https://github.com/davidgolub/QuestionGeneration.

SQuAD and NewsQA datasets are more similar than SQuAD and MS MARCO, in terms of question style. For example, questions in MS MARCO are real web search queries, which are short and may have typos or abbreviations; while questions in SQuAD and NewsQA are more formal and well written. Furthermore, the ground-truth answers in MS MARCO are human-synthesized and usually much longer (16.4 tokens in average) than those in the other datasets, while our answer extraction process focuses on named entities (which are much shorter). We argue that extracting named entities as possible answers is still reasonable for most of the reading comprehension tasks such as SQuAD and NewsQA. The problem of synthesizing answers across different domains will be investigated in future work.

**SynNet vs. pre-trained SAN baseline** One observation is that SynNet performs worse than the pre-trained SAN baseline. We hypothesize that this is because the generated question-answer pairs are often noisy and inaccurate, and directly fine-tuning the answer module with synthetic data may hurt the performance, which is also observed in Sachan and Xing (2018), especially when a well-performed MRC model is used as the baseline. Note that we do observe improvements from SynNet+BiDAF over the pre-trained BiDAF model, which will be discussed in Sec. 6.2.

**Comparing with upper-bound** The "AdaMRC with GT questions" model (in Section 5.2) serves as the upper-bound of our proposed approach, where ground-truth questions are used instead of synthesized questions. By using ground-truth questions, performance is further boosted by around 1%. This suggests that our question generation model is effective as the margin is relatively small, yet it could be further improved. We plan to study if recent question generation methods (Du et al., 2017; Duan et al., 2017; Sun et al., 2018; Benmalek et al., 2019) could further help to close the performance gap in future work.

# 6 Analysis

## 6.1 Visualization

To demonstrate the effectiveness of adversarial domain adaptation, we visualize the encoded representation via t-SNE (Maaten and Hinton, 2008) in Figure 1. We observe that with AdaMRC, the two distributions of encoded feature representations are indistinguishable. Without AdaMRC, the two distributions are independently clustered by domain. We further use KL divergence for measuring distributional differences. The KL divergence of data samples between source and target domains, with and without domain adaptation, are 0.278, 0.433, respectively (smaller is better).

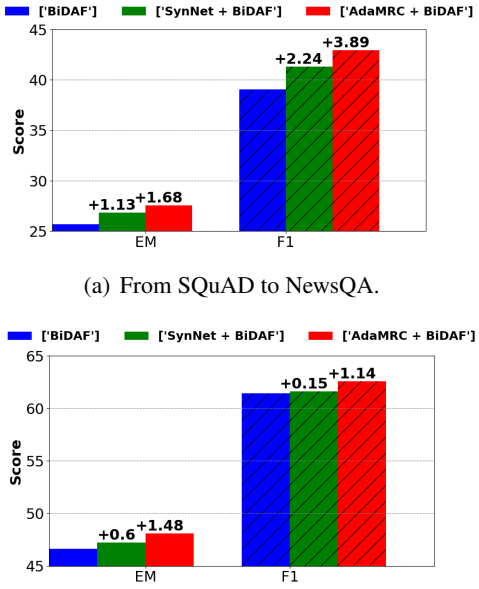


(a) From SQuAD to NewsQA.




(b) From NewsQA to SQuAD.

Figure 3: Performance of our proposed method compared with baselines, using BiDAF as the MRC model.

| Method | EM/F1 |
|---|---|
| SAN | 32.35/42.62 |
| AdaMRC + SAN | **33.61/44.16** |
| BiDAF | 27.85/36.82 |
| AdaMRC + BiDAF | **29.12/38.84** |

Table 3: Performance on DuoRC, adapting from SelfRC (Wikipedia) to ParaphraseRC (IMDB).

## 6.2 Robustness of AdaMRC

**Results on BiDAF** To verify that our proposed framework is compatible to existing MRC models, we also apply our framework to the BiDAF model, which has different encoder and decoder structures compared to SAN. We follow the model architecture and parameter settings in Seo et al. (2017). As shown in Figure 3, the proposed AdaMRC model clearly outperforms both SynNet+BiDAF and pre-trained BiDAF model. We also observe that the improvement of AdaMRC over BiDAF is more significant than SAN. Our hypothesis is that since BiDAF is a weaker baseline than SAN, a higher performance improvement can be observed when the domain adaptation approach is applied to enhance the model. This experiment confirms that

| Method | EM/F1 |
|---|---|
| SAN | 36.68/52.79 |
| AdaMRC + SAN | **38.46/54.20** |
| SAN + ELMo | 39.61/55.18 |
| AdaMRC + SAN + ELMo | **40.96/56.25** |
| $\text{BERT}_{\text{BASE}}$ | 42.00/58.71 |
| AdaMRC + $\text{BERT}_{\text{BASE}}$ | **42.59/59.25** |

Table 4: Results of using ELMo and BERT. Setting: adaptation from SQuAD to NewsQA.

| Ratio | SAN | AdaMRC + SAN |
|---|---|---|
| 0% | 36.68/52.79 | **38.46/54.20** |
| 5% | 47.61/62.69 | **48.50/63.17** |
| 10% | 48.66/63.32 | **49.64/63.94** |
| 20% | 50.75/64.80 | **51.14/65.38** |
| 50% | 53.24/67.07 | **53.34/67.30** |
| 100% | **56.48/69.14** | 56.29/68.97 |

Table 5: Semi-supervised domain adaptation experiment with varied labeling ratio on the target-domain dataset. Setting: adaptation from SQuAD to NewsQA.

our proposed approach is robust and can generalize to different MRC models.

**Results on DuoRC** We further test our model on the newly-released DuoRC dataset (Saha et al., 2018). This dataset contains two subsets: movie descriptions collected from *Wikipedia* (SelfRC) and from *IMDB* (ParaphraseRC). Although the two subsets are describing the same movies, the documents from Wikipedia are usually shorter (580 words in average), while the documents from IMDB are longer and more descriptive (926 words in average). We consider them as two different domains and perform domain adaptation from Wikipedia to IMDB. This experiment broadens our definition of domain.

In the DuoRC dataset, the same questions are asked on both Wikipedia and IMDB documents. Thus, question synthesis is not needed, and comparison with SynNet is not feasible. Note that the answers of the same question could be different in the two subsets (only 40.7% of the questions have the same answers in both domains). We preprocess the dataset and test the answer-span extraction task following Saha et al. (2018). Results are reported in Table 3. AdaMRC improves the performance over both SAN (1.26%, 1.54% in EM and F1) and BiDAF (1.27%, 2.02% in EM and F1). This experiment validates that our method can be applied to different styles of domain adaptation tasks as well.

### 6.3 AdaMRC with Pre-trained Language Models

To verify that our approach is compatible with large-scale pre-trained language models, we evaluate our model with ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019). To apply ELMo to SAN, we use the model provided by AllenNLP[4], and append a 1024-dim ELMo vector to the contextual encoding layer, with dropout rate set to 0.5. For BERT, we experiment with the pre-trained $\text{BERT}_{\text{BASE}}$ uncased model[5] due to limited computational resources. We use the original design of finetuning BERT for the MRC task in Devlin et al. (2019), instead of combining BERT with SAN. Results are provided in Table 4. We observe that using ELMo and BERT improves both AdaMRC and the baseline model. However, the improvement over ELMo and BERT is relatively smaller than SAN. We believe this is because pre-trained language model provides additional domain-invariant information learned from external data, and therefore limits the improvement of domain-invariant feature learning in our model. However, it is worth noting that combining AdaMRC with BERT achieves the best performance, which validates that AdaMRC is compatible with data augmentation from external sources.

### 6.4 Semi-supervised Setting

As an additional experiment, we also evaluate the proposed AdaMRC framework for semi-supervised domain adaptation. We randomly sample $k$ portion of labeled data from the target domain, and feed them to the MRC model. The ratio of labeled data ranges $k$ from 0% to 100%. Table 5 shows that AdaMRC outperforms SAN. However, the gap is decreasing when the labeling ratio increases. When the ratio is 20% or smaller, there is noticeable improvement. When the ratio is set to 50%, the two methods result in similar performance. When the ratio is increased to 100%, *i.e.*, fully supervised learning, the performance of AdaMRC is slightly worse than SAN. This is possibly because in a supervised learning setting, the encoder is trained to preserve domain-specific feature information. The overall results suggest that our proposed AdaMRC is also effective in semi-supervised setting, when a small portion of target-domain data is provided.

[4] https://allennlp.org/

[5] https://github.com/google-research/bert

*Refugee camps in eastern Chad house about 300,000 people who fled violence in the Darfur region of Sudan . The U.N. High Commissioner for Refugees said on Monday that more than **12,000** people have fled militia attacks over the last few days from Sudan 's Darfur region to neighboring Chad...*
**Answer:** 12,000
**GT Question**: How many have recently crossed to Chad?
**Pseudo Question**: How many people fled the Refugee region to Sudan?

*Sources say the classified materials were taken from the East Tennessee Technology Park . **Roy Lynn Oakley** , 67 , of Roane County , Tennessee , appeared in federal court in Knoxville on Thursday . Oakley was briefly detained for questioning in the case in January ...*
**Answer:** Roy Lynn Oakley
**GT Question**: Who is appearing in court ?
**Pseudo Question**: What is the name of the classified employee in Tennessee on East Tennessee ?

*The Kyrgyz order became effective on Friday when **President Kurmanbek Bakiyev** reportedly signed legislation that the parliament in Bishkek backed on Thursday , the Pentagon said . Pentagon spokesman Bryan Whitman said the Kyrgyz Foreign Ministry on Friday officially notified the U.S. Embassy in Bishkek that a 180-day withdrawal process is under way...*
**Answer:** President Kurmanbek Bakiyev
**GT Question**: Who is the President of Kyrgyzstan ?
**Pseudo Question**: What spokesman signed legislation that the parliament was signed legislation in 2011 ?

*A high court in northern India on Friday acquitted a wealthy businessman facing the death sentence for the killing of a teen in a case dubbed " the house of horrors . " Moninder Singh Pandher was sentenced to death by a lower court in February . The teen was **one of 19** victims – children and young women – in one of the most gruesome serial killings in India in recent years ...*
**Answer:** one of 19
**GT Question**:What was the amount of children murdered?
**Pseudo Question**: How many victims were in India?

Table 6: Examples of generated questions given input paragraphs and answers, comparing with the ground-truth human-written questions.

### 6.5 Examples of Generated Questions

The percentage of generated questions starting with "what", "who", "when" and "where" are 63.2%, 12.8%, 2.3%, and 2.1%, respectively. We provide several examples of generated questions in Table 6. We observe that the generated questions are longer than human-written questions. This is possibly due to the copy mechanism used in the question generation model, which enables directly copying words into the generated questions. On the one hand, the copy mechanism provides detailed background information for generating a question. However, if not copying correctly, the question could be syntactically incorrect. For instance, in the third example, "*signed legislation that the parliament*" is copied from the passage. The copied phrase is indeed describing the answer "*President Kurmanbek Bakiyev*"; however, the question is syntactically incorrect and the question generator should copy "*the parliament backed on Thursday*" instead.

There is generally good correspondence between the answer type and generated questions. For example, the question generator will produce "*What is the name of*" if the answer is about a person, and ask "*How many*" if the answer is a number. We also observe that the generated questions may encounter semantic errors though syntactically fluent. For instance, in the first example, the passage suggests that people fled *from* Sudan to Chad, while the generated question describes the wrong direction. However, overall we think that the current question generator provides reasonable synthesized questions, yet there is still large room to improve. The observation also confirms our analysis that the synthetic question-answer pairs could be noisy and inaccurate, thus could hurt the performance when fine-tuning the answer module with generated data.

## 7 Conclusion

In this paper, we propose a new framework, *Adversarial Domain Adaptation for MRC* (AdaMRC), to transfer a pre-trained MRC model from a source domain to a target domain. We validate our proposed framework on several datasets and observe consistent improvement over baseline methods. We verify the robustness of the proposed framework by applying it to different MRC models. Experiments also show that AdaMRC is compatible with pre-trained language model and semi-supervised learning setting. We believe our analysis provides insights that can help guide further research in this task.

## References


Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

Ryan Benmalek, Madian Khabsa, Suma Desu, Claire Cardie, and Michele Banko. 2019. Keeping notes: Conditional natural language generation with a scratchpad encoder. In *ACL*.

Rich Caruana. 1997. Multitask learning. *Machine learning*.

Xilun Chen, Yu Sun, Ben Athiwaratkun, Claire Cardie, and Kilian Weinberger. 2018. Adversarial deep averaging networks for cross-lingual sentiment classification. *TACL*.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Yu-An Chung, Hung-Yi Lee, and James Glass. 2018. Supervised and unsupervised transfer learning for question answering. In *NAACL*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Mortaza Doulaty, Oscar Saz, and Thomas Hain. 2015. Data-selective transfer learning for multi-domain speech recognition. *arXiv preprint arXiv:1509.02409*.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *ACL*.

Nan Duan, Duyu Tang, Peng Chen, and Ming Zhou. 2017. Question generation for question answering. In *EMNLP*.

Lisheng Fu, Thien Huu Nguyen, Bonan Min, and Ralph Grishman. 2017. Domain adaptation for relation extraction with domain adversarial neural network. In *IJCNLP*.

Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *ICML*.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *JMLR*.

Jianfeng Gao, Michel Galley, and Lihong Li. 2019. Neural approaches to conversational ai. *Foundations and Trends® in Information Retrieval*.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *ICML*.

David Golub, Po-Sen Huang, Xiaodong He, and Li Deng. 2017. Two-stage synthesis networks for transfer learning in machine comprehension. In *EMNLP*.

Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. 2012. Geodesic flow kernel for unsupervised domain adaptation. In *CVPR*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *NIPS*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *ACL*.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *ACL*.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda B. Viégas, Martin Wattenberg, Gregory S. Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *TACL*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Zheng Li, Yun Zhang, Ying Wei, Yuxiang Wu, and Qiang Yang. 2017. End-to-end adversarial memory network for cross-domain sentiment classification. In *IJCAI*.

Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. 2015. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *NAACL*.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. In *ACL*.

Xiaodong Liu, Yelong Shen, Kevin Duh, and Jianfeng Gao. 2018. Stochastic answer networks for machine reading comprehension. In *ACL*.

Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I. Jordan. 2015. Learning transferable features with deep adaptation networks. In *ICML*.

Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. 2016. Unsupervised domain adaptation with residual transfer networks. In *NIPS*.

Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. 2017. Deep transfer learning with joint adaptation networks. In *ICML*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *JMLR*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *NIPS*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *EMNLP*.

Mrinmaya Sachan and Eric Xing. 2018. Self-training for jointly learning to ask and answer questions. In *NAACL*.

Amrita Saha, Rahul Aralikatte, Mitesh M Khapra, and Karthik Sankaranarayanan. 2018. Duorc: Towards complex language understanding with paraphrased reading comprehension. In *ACL*.

Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. 2018. Maximum classifier discrepancy for unsupervised domain adaptation. In *CVPR*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *ICLR*.

Darsh J Shah, Tao Lei, Alessandro Moschitti, Salvatore Romeo, and Preslav Nakov. 2018. Adversarial domain adaptation for duplicate question detection. In *EMNLP*.

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2017. Reasonet: Learning to stop reading in machine comprehension. In *KDD*.

Xingwu Sun, Jing Liu, Yajuan Lyu, Wei He, Yanjun Ma, and Shi Wang. 2018. Answer-focused and position-aware neural question generation. In *EMNLP*.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2016. Newsqa: A machine comprehension dataset. *arXiv preprint arXiv:1611.09830*.

Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. 2017. Adversarial discriminative domain adaptation. In *CVPR*.

Georg Wiese, Dirk Weissenborn, and Mariana Neves. 2017. Neural question answering at bioasq 5b. In *BioNLP workshop*.

Caiming Xiong, Victor Zhong, and Richard Socher. 2017. Dynamic coattention networks for question answering. In *ICLR*.

Yichong Xu, Xiaodong Liu, Yelong Shen, Jingjing Liu, and Jianfeng Gao. 2019. Multi-task learning with sample re-weighting for machine reading comprehension. In *NAACL*.

Zhilin Yang, Junjie Hu, Ruslan Salakhutdinov, and William Cohen. 2017. Semi-supervised qa with generative domain-adaptive nets. In *ACL*.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. In *ICLR*.

Sheng Zhang, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, Kevin Duh, and Benjamin Van Durme. 2018. Record: Bridging the gap between human and machine commonsense reading comprehension. *arXiv preprint arXiv:1810.12885*.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *EMNLP*.

# Incorporating External Knowledge into Machine Reading for Generative Question Answering

**Bin Bi, Chen Wu, Ming Yan, Wei Wang,**
**Jiangnan Xia, Chenliang Li**
Alibaba Group
{b.bi, wuchen.wc, ym119608, hebian.ww}@alibaba-inc.com
{jiangnan.xjn, lcl193798}@alibaba-inc.com

## Abstract

Commonsense and background knowledge is required for a QA model to answer many non-trivial questions. Different from existing work on knowledge-aware QA, we focus on a more challenging task of leveraging external knowledge to generate answers in natural language for a given question with context.

In this paper, we propose a new neural model, Knowledge-Enriched Answer Generator (KEAG), which is able to compose a natural answer by exploiting and aggregating evidence from all four information sources available: question, passage, vocabulary and knowledge. During the process of answer generation, KEAG adaptively determines when to utilize symbolic knowledge and which fact from the knowledge is useful. This allows the model to exploit external knowledge that is not explicitly stated in the given text, but that is relevant for generating an answer. The empirical study on public benchmark of answer generation demonstrates that KEAG improves answer quality over models without knowledge and existing knowledge-aware models, confirming its effectiveness in leveraging knowledge.

## 1 Introduction

Question Answering (QA) has come a long way from answer sentence selection, relational QA to machine reading comprehension. The next-generation QA systems can be envisioned as the ones which can read passages and write long and abstractive answers to questions. Different from extractive question answering, generative QA based on machine reading produces an answer in true natural language which does not have to be a sub-span in the given passage.

Most existing models, however, answer questions based on the content of given passages as the only information source. As a result, they may not be able to understand certain passages or to answer certain questions, due to the lack of commonsense and background knowledge, such as the knowledge about what concepts are expressed by the words being read (lexical knowledge), and what relations hold between these concepts (relational knowledge). As a simple illustration, given the passage:

*State officials in Hawaii on Monday said they have once again checked and confirmed that President Barack Obama was born in Hawaii.*

to answer the question: *Was Barack Obama born in the U.S.?*, one must know (among other things) that Hawaii is a state in the U.S., which is external knowledge not present in the text corpus.

Therefore, a QA model needs to be enriched with external knowledge properly to be able to answer many nontrivial questions. Such knowledge can be commonsense knowledge or factual background knowledge about entities and events that is not explicitly expressed but can be found in a knowledge base such as ConceptNet (Speer et al., 2016), Freebase (Pellissier Tanon et al., 2016) and domain-specific KBs collected by information extraction (Fader et al., 2011; Mausam et al., 2012). Thus, we aim to design a neural model that encodes pre-selected knowledge relevant to given questions, and that learns to include the available knowledge as an enrichment to given textual information.

In this paper, we propose a new neural architecture, Knowledge-Enriched Answer Generator (KEAG), specifically designed to generate natural answers with integration of external knowledge. KEAG is capable of leveraging symbolic knowledge from a knowledge base as it generates each word in an answer. In particular, we assume that each word is generated from one of the four information sources: 1. question, 2. passage, 3. vocabulary and 4. knowledge. Thus, we introduce the



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2521–2530, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

*source selector*, a sentinel component in KEAG that allows flexibility in deciding which source to look to generate every answer word. This is crucial, since knowledge plays a role in certain parts of an answer, while in others text context should override the context-independent knowledge available in general KBs.

At each timestep, before generating an answer word, KEAG determines an information source. If the knowledge source is selected, the model extracts a set of facts that are potentially related to the given question and context. A stochastic fact selector with discrete latent variables then picks a fact based on its semantic relevance to the answer being generated. This enables KEAG to bring external knowledge into answer generation, and to generate words not present in the predefined vocabulary. By incorporating knowledge explicitly, KEAG can also provide evidence about the external knowledge used in the process of answer generation.

We introduce a new differentiable sampling-based method to learn the KEAG model in the presence of discrete latent variables. For empirical evaluation, we conduct experiments on the benchmark dataset of answer generation MARCO (Nguyen et al., 2016). The experimental results demonstrate that KEAG effectively leverages external knowledge from knowledge bases in generating natural answers. It achieves significant improvement over classic QA models that disregard knowledge, resulting in higher-quality answers.

## 2 Related Work

There have been several attempts at using machine reading to generate natural answers in the QA field. Tan et al. (2018) took a generative approach where they added a decoder on top of their extractive model to leverage the extracted evidence for answer synthesis. However, this model still relies heavily on the extraction to perform the generation and thus needs to have start and end labels (a span) for every QA pair. Mitra (2017) proposed a seq2seq-based model that learns alignment between a question and passage words to produce rich question-aware passage representation by which it directly decodes an answer. Gao et al. (2019) focused on product-aware answer generation based on large-scale unlabeled e-commerce reviews and product attributes. Furthermore, natural answer generation can be reformulated as query-focused summarization which is addressed by Nema et al. (2017).

The role of knowledge in certain types of QA tasks has been remarked on. Mihaylov and Frank (2018) showed improvements on a cloze-style task by incorporating commonsense knowledge via a context-to-commonsense attention. Zhong et al. (2018) proposed commonsense-based pre-training to improve answer selection. Long et al. (2017) made use of knowledge in the form of entity descriptions to predict missing entities in a given document. There have also been a few studies on incorporating knowledge into QA models without passage reading. GenQA (Yin et al., 2016) combines knowledge retrieval and seq2seq learning to produce fluent answers, but it only deals with simple questions containing one single fact. COREQA (He et al., 2017) extends it with a copy mechanism to learn to copy words from a given question. Moreover, Fu and Feng (2018) introduced a new attention mechanism that attends across the generated history and memory to explicitly avoid repetition, and incorporated knowledge to enrich generated answers.

Some work on knowledge-enhanced natural language (NLU) understanding can be adapted to the question answering task. CRWE (Weissenborn, 2017) dynamically integrates background knowledge in a NLU model in the form of free-text statements, and yields refined word representations to a task-specific NLU architecture that reprocesses the task inputs with these representations. In contrast, KBLSTM (Yang and Mitchell, 2017) leverages continuous representations of knowledge bases to enhance the learning of recurrent neural networks for machine reading. Furthermore, Bauer et al. (2018) proposed MHPGM, a QA architecture that fills in the gaps of inference with commonsense knowledge. The model, however, does not allow an answer word to come directly from knowledge. We adapt these knowledge-enhanced NLU architectures to answer generation, as baselines for our experiments.

## 3 Knowledge-aware Answer Generation

Knowledge-aware answer generation is a question answering paradigm, where a QA model is expected to generate an abstractive answer to a given question by leveraging both the contextual passage and external knowledge. More formally, given a knowledge base $\mathcal{K}$ and two sequences of input words: question

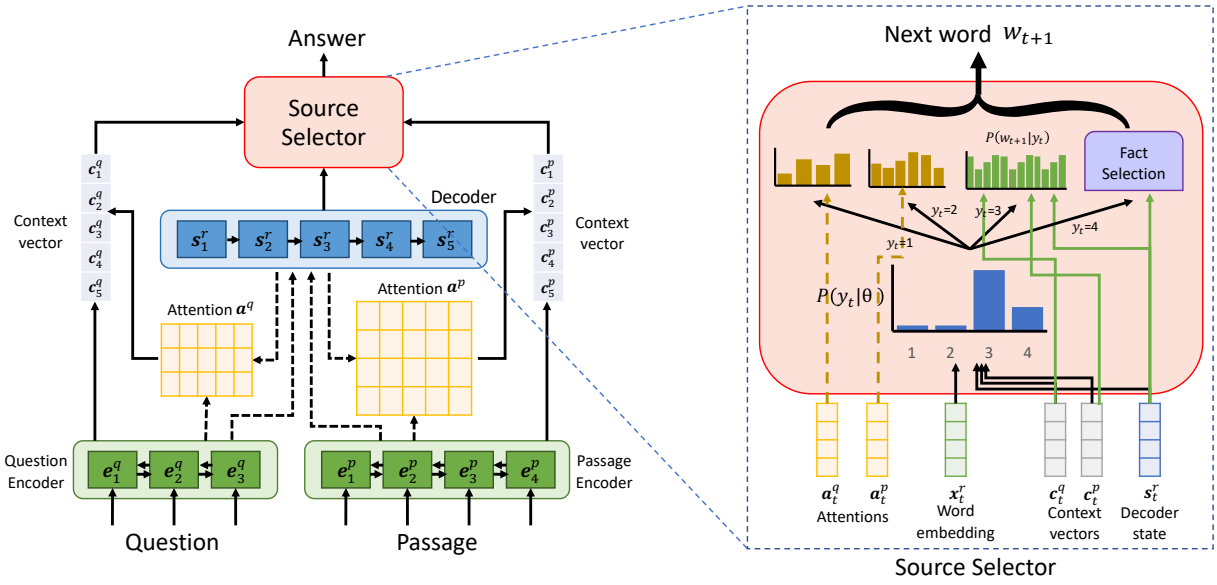


Figure 1: An overview of the architecture of KEAG (best viewed in color). A question and a passage both go through an extension of the sequence-to-sequence model. The outcomes are then fed into a source selector to generate a natural answer.

$q = \{w_1^q, w_2^q, \ldots, w_{N_q}^q\}$ and passage $p = \{w_1^p, w_2^p, \ldots, w_{N_p}^p\}$, the answer generation model should produce a series of answer words $r = \{w_1^r, w_2^r, \ldots, w_{N_r}^r\}$. The knowledge base $\mathcal{K}$ contains a set of facts, each of which is represented as a triple $f = (subject, relation, object)$ where $subject$ and $object$ can be multi-word expressions and $relation$ is a relation type, e.g., $(bridge, UsedFor, cross\ water)$.

### 3.1 Knowledge-Enriched Answer Generator

To address the answer generation problem, we propose a novel KEAG model which is able to compose a natural answer by recurrently selecting words at the decoding stage. Each of the words comes from one of the four sources: question $q$, passage $p$, global vocabulary $\mathcal{V}$, and knowledge $\mathcal{K}$. In particular, at every generation step, KEAG first determines which of the four sources to inspect based on the current state, and then generates a new word from the chosen source to make up a final answer. An overview of the neural architecture of KEAG is depicted in Figure 1.

### 3.2 Sequence-to-sequence model

KEAG is built upon an extension of the sequence-to-sequence attentional model (Bahdanau et al., 2015; Nallapati et al., 2016; See et al., 2017). The words of question $q$ and passage $p$ are fed one-by-one into two different encoders, respectively. Each of the two encoders, which are both bidirectional LSTMs, produces a sequence of encoder hidden states ($\mathbf{E}^q$ for question $q$, and $\mathbf{E}^p$ for passage $p$). In each timestep $t$, the decoder, which is a unidirectional LSTM, takes an answer word as input, and outputs a decoder hidden state $\mathbf{s}_t^r$.

We calculate attention distributions $\mathbf{a}_t^q$ and $\mathbf{a}_t^p$ on the question and the passage, respectively, as in (Bahdanau et al., 2015):

$$\mathbf{a}_t^q = \text{softmax}(\mathbf{g}^{q\intercal}\tanh(\mathbf{W}^q\mathbf{E}^q + \mathbf{U}^q\mathbf{s}_t^r + \mathbf{b}^q)), \quad (1)$$

$$\mathbf{a}_t^p = \text{softmax}(\mathbf{g}^{p\intercal}\tanh(\mathbf{W}^p\mathbf{E}^p + \mathbf{U}^p\mathbf{s}_t^r + \mathbf{V}^p\mathbf{c}^q + \mathbf{b}^p)), \quad (2)$$

where $\mathbf{g}^q$, $\mathbf{W}^q$, $\mathbf{U}^q$, $\mathbf{b}^q$, $\mathbf{g}^p$, $\mathbf{W}^p$, $\mathbf{U}^p$ and $\mathbf{b}^p$ are learnable parameters. The attention distributions can be viewed as probability distributions over source words, which tells the decoder where to look to generate the next word. The coverage mechanism is added to the attentions to avoid generating repetitive text (See et al., 2017). In Equation 2, we introduce $\mathbf{c}^q$, a context vector for the question, to make the passage attention aware of the question context. $\mathbf{c}^q$ for the question and $\mathbf{c}^p$ for the passage are calculated as follows:

$$\mathbf{c}_t^q = \sum_i a_{ti}^q \cdot \mathbf{e}_i^q, \qquad \mathbf{c}_t^p = \sum_i a_{ti}^p \cdot \mathbf{e}_i^p, \quad (3)$$

where $\mathbf{e}_i^q$ and $\mathbf{e}_i^p$ are an encoder hidden state for question $q$ and passage $p$, respectively. The context vectors ($\mathbf{c}_t^q$ and $\mathbf{c}_t^p$) together with the attention distributions ($\mathbf{a}_t^q$ and $\mathbf{a}_t^p$) and the decoder state

($\mathbf{s}_t^r$) will be used downstream to determine the next word in composing a final answer.

## 4 Source Selector

During the process of answer generation, in each timestep, KEAG starts with running a source selector to pick a word from one source of the question, the passage, the vocabulary and the knowledge. The right plate in Figure 1 illustrates how the source selector works in one timestep during decoding.

If the question source is selected in timestep $t$, KEAG picks a word according to the attention distribution $\mathbf{a}_t^q \in \mathbb{R}^{N_q}$ over question words (Equation 1), where $N_q$ denotes the number of distinct words in the question. Similarly, when the passage source is selected, the model picks a word from the attention distribution $\mathbf{a}_t^p \in \mathbb{R}^{N_p}$ over passage words (Equation 2), where $N_p$ denotes the number of distinct words in the passage. If the vocabulary is the source selected in timestep $t$, the new word comes from the conditional vocabulary distribution $P_v(w|\mathbf{c}_t^q, \mathbf{c}_t^p, \mathbf{s}_t^r)$ over all words in the vocabulary, which is obtained by:

$$P_v(w|\mathbf{c}_t^q, \mathbf{c}_t^p, \mathbf{s}_t^r) = \text{softmax}(\mathbf{W}^v \cdot [\mathbf{c}_t^q, \mathbf{c}_t^p, \mathbf{s}_t^r] + \mathbf{b}^v), \tag{4}$$

where $\mathbf{c}_t^q$ and $\mathbf{c}_t^p$ are context vectors, and $\mathbf{s}_t^r$ is a decoder state. $\mathbf{W}^v$ and $\mathbf{b}^v$ are learnable parameters.

To determine which of the four sources a new word $w_{t+1}$ is selected from, we introduce a discrete latent variable $y_t \in \{1, 2, 3, 4\}$ as an indicator. When $y_t = 1$ or 2, the word $w_{t+1}$ is generated from the distribution $P(w_{t+1}|y_t)$ given by:

$$P(w_{t+1}|y_t) = \begin{cases} \sum_{i:w_i = w_{t+1}} a_{ti}^q & y_t = 1 \\ \sum_{i:w_i = w_{t+1}} a_{ti}^p & y_t = 2. \end{cases} \tag{5}$$

If $y_t = 3$, KEAG picks word $w_{t+1}$ according to the vocabulary distribution $P_v(w|\mathbf{c}_t^q, \mathbf{c}_t^p, \mathbf{s}_t^r)$ given in Equation 4. Otherwise, if $y_t = 4$, the word $w_{t+1}$ comes from the fact selector, which will be described in the coming section.

## 5 Knowledge Integration

In order for KEAG to integrate external knowledge, we first extract related facts from the knowledge base in response to a given question, from which we then pick the most relevant fact that can be used for answer composition. In this section, we present the two modules for knowledge integration: *related fact extraction* and *fact selection*.

### 5.1 Related Fact Extraction

Due to the size of a knowledge base and the large amount of unnecessary information, we need an effective way of extracting a set of candidate facts which provide novel information while being related to a given question and passage.

For each instance $(q, p)$, we first extract facts with the subject or object that occurs in question $q$ or passage $p$. Scores are added to each extracted fact according to the following rules:

- Score+4, if the subject occurs in $q$, *and* the object occurs in $p$.
- Score+2, if the subject *and* the object both occur in $p$.
- Score+1, if the subject occurs in $q$ *or* $p$.

The scoring rules are set heuristically such that they model relative fact importance in different interactions. Next, we sort the fact triples in descending order of their scores, and take the top $N_f$ facts from the sorted list as the related facts for subsequent processing.

### 5.2 Fact Selection

Figure 2 displays how a fact is selected from the set of related facts for answer completion. With the extracted knowledge, we first embed every related fact $f$ by concatenating the embeddings of the subject $\mathbf{e}^s$, the relation $\mathbf{e}^r$ and the object $\mathbf{e}^o$. The embeddings of subjects and objects are initialized with pre-trained GloVe vectors (and average pooling for multiple words), when the words are present in the vocabulary. The fact embedding is followed by a linear transformation to relate subject $\mathbf{e}^s$ to object $\mathbf{e}^o$ with relation $\mathbf{e}^r$:

$$\mathbf{f} = \mathbf{W}^e \cdot [\mathbf{e}^s, \mathbf{e}^r, \mathbf{e}^o] + \mathbf{b}^e. \tag{6}$$

where $\mathbf{f}$ denotes fact representation, $[\cdot, \cdot]$ denotes vector concatenation, and $\mathbf{W}^e$ and $\mathbf{b}^e$ are learnable parameters. The set of all related fact representations $F = \{\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_{N_f}\}$ is considered to be a short-term memory of the knowledge base while answering questions on given passages.

To enrich KEAG with the facts collected from the knowledge base, we propose to complete an answer with the most relevant fact(s) whenever it is determined to resort to knowledge during the process of answer generation. The most relevant fact is selected from the related fact set $F$ based on the dynamic generation state. In this model, we introduce a discrete latent random variable $z_t \in$

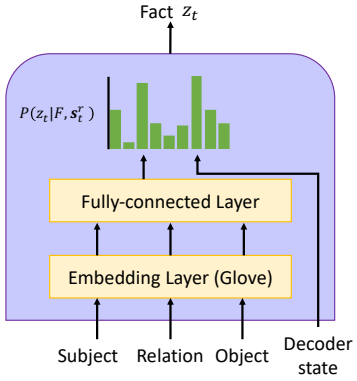


Figure 2: An overview of the fact selection module (best viewed in color)

$[1, N_f]$ to explicitly indicate which fact is selected to be put into an answer in timestep $t$. The model selects a fact by sampling a $z_t$ from the discrete distribution $P(z_t|F, \mathbf{s}_t^r)$ given by:

$$P(z_t|\cdot) = \frac{1}{Z} \cdot \exp(\mathbf{g}^{f\intercal} \tanh(\mathbf{W}^f \mathbf{f}_{z_t} + \mathbf{U}^f \mathbf{s}_t^r + \mathbf{b}^f)), \tag{7}$$

where $Z$ is the normalization term, $Z = \sum_{i=1}^{N_f} \exp(\mathbf{g}^{f\intercal} \tanh(\mathbf{W}^f \mathbf{f}_i + \mathbf{U}^f \mathbf{s}_t^r + \mathbf{b}^f))$, and $\mathbf{s}_t^r$ is the hidden state from the decoder in timestep $t$. $\mathbf{g}^f$, $\mathbf{W}^f$, $\mathbf{U}^f$ and $\mathbf{b}^f$ are learnable parameters.

The presence of discrete latent variables $\mathbf{z}$, however, presents a challenge to training the neural KEAG model, since the backpropagation algorithm, while enabling efficient computation of parameter gradients, does not apply to the non-differentiable layer introduced by the discrete variables. In particular, gradients cannot propagate through discrete samples from the categorical distribution $P(z_t|F, \mathbf{s}_t^r)$.

To address this problem, we create a differentiable estimator for discrete random variables with the Gumbel-Softmax trick (Jang et al., 2017). Specifically, we first compute the discrete distribution $P(z_t|F, \mathbf{s}_t^r)$ with class probabilities $\pi_1, \pi_2, \ldots, \pi_{N_f}$ by Equation 7. The Gumbel-Max trick (Gumbel, 1954) allows to draw samples from the categorical distribution $P(z_t|F, \mathbf{s}_t^r)$ by calculating one_hot$(\arg\max_i [g_i + \log \pi_i])$, where $g_1, g_2, \ldots, g_{N_f}$ are i.i.d. samples drawn from the Gumbel$(0, 1)$ distribution. For the inference of a discrete variable $z_t$, we approximate the Gumbel-Max trick by the continuous softmax function (in place of $\arg\max$) with temperature $\tau$ to generate a sample vector $\hat{\mathbf{z}}_t$:

$$\hat{z}_{ti} = \frac{\exp((\log(\pi_i) + g_i)/\tau)}{\sum_{j=1}^{N_f} \exp((\log(\pi_j) + g_j)/\tau)}. \tag{8}$$

When $\tau$ approaches zero, the generated sample $\hat{\mathbf{z}}_t$ becomes a one-hot vector. $\tau$ is gradually annealed over the course of training.

This new differentiable estimator allows us to backpropagate through $z_t \sim P(z_t|F, \mathbf{s}_t^r)$ for gradient estimation of every single sample. The value of $z_t$ indicates a fact selected by the decoder in timestep $t$. When the next word is determined to come from knowledge, the model appends the object of the selected fact to the end of the answer being generated.

## 6 Learning Model Parameters

To learn the parameters $\theta$ in KEAG with latent source indicators $\mathbf{y}$, we maximize the log-likelihood of words in all answers. For each answer, the log-likelihood of the words is given by:

$$\log P(w_1^r, w_2^r, \ldots, w_{N_r}^r|\theta) = \sum_{t=1}^{N_r} \log P(w_t^r|\theta)$$

$$= \sum_{t=1}^{N_r} \log \sum_{y_t=1}^{4} P(w_{t+1}|y_t) P(y_t|\theta) \tag{9}$$

$$\geq \sum_{t=1}^{N_r} \sum_{y_t=1}^{4} P(y_t|\theta) \log P(w_{t+1}|y_t) \tag{10}$$

$$= \sum_{t=1}^{N_r} \mathbb{E}_{y_t|\theta}[\log P(w_{t+1}|y_t)], \tag{11}$$

where the word likelihood at each timestep is obtained by marginalizing out the latent source variable $y_t$. Unfortunately, direct optimization of Equation 9 is intractable, so we instead learn the objective function through optimizing its variational lower bound given in Equations 10 and 11, obtained from Jensen's inequality.

To estimate the expectation in Equation 11, we use Monte Carlo sampling on the source selector variables $\mathbf{y}$ in the gradient computation. In particular, the Gumbel-Softmax trick is applied to generate discrete samples $\hat{\mathbf{y}}$ from the probability $P(y_t|\mathbf{c}_t^q, \mathbf{c}_t^p, \mathbf{s}_t^r, \mathbf{x}_t^r)$ given by:

$$P(y_t|\cdot) = \text{softmax}(\mathbf{W}^y \cdot [\mathbf{c}_t^q, \mathbf{c}_t^p, \mathbf{s}_t^r, \mathbf{x}_t^r] + \mathbf{b}^y), \tag{12}$$

where $\mathbf{x}_t^r$ is the embedding of the answer word in timestep $t$, $\mathbf{W}^y$ and $\mathbf{b}^y$ are learnable parameters. The generated samples are fed to $\log P(w_{t+1}|y_t)$ to estimate the expectation.

## 7 Experiments

We perform quantitative and qualitative analysis of KEAG through experiments. In our experi-

ments, we also study the impact of the integrated knowledge and the ablations of the KEAG model. In addition, we illustrate how natural answers are generated by KEAG with the aid of external knowledge by analyzing a running example.

### 7.1 Dataset and Evaluation Metrics

Given our objective of generating natural answers by document reading, the MARCO dataset (Nguyen et al., 2016) released by Microsoft is the best fit for benchmarking KEAG and other answer generation methods. We use the latest MARCO V2.1 dataset and focus on the "*Q&A + Natural Language Generation*" task in the evaluation, the goal of which is to provide the best answer available in natural language that could be used by a smart device / digital assistant.

In the MARCO dataset, the questions are user queries issued to the Bing search engine and the contextual passages are from real web documents. The data has been split into a training set (153,725 QA pairs), a dev set (12,467 QA pairs) and a test set (101,092 questions with unpublished answers). Since true answers are not available in the test set, we hold out the dev set for evaluation in our experiments, and test models for each question on its associated passages by concatenating them all together. We tune the hyper-parameters by cross-validation on the training set.

The answers are human-generated and not necessarily sub-spans of the passages, so the official evaluation tool of MARCO uses the metrics BLEU-1 (Papineni et al., 2002) and ROUGE-L (Lin, 2004). We use both metrics for our evaluation to measure the quality of generated answers against the ground truth.

For external knowledge, we use ConceptNet (Speer et al., 2016), one of the most widely used commonsense knowledge bases. Our KEAG is generic and thus can also be applied to other knowledge bases. ConceptNet is a semantic network representing words and phrases as well as the commonsense relationships between them. After filtering out non-English entities and relation types with few facts, we have 2,823,089 fact triples and 32 relation types for the model to consume.

### 7.2 Implementation Details

In KEAG, we use 300-dimensional pre-trained *Glove* word embeddings (Pennington et al., 2014) for initialization with update during training. The dimension of hidden states is set to 256 for every LSTM. The fact representation $\mathbf{f}$ has 500 dimensions. The maximum number of related facts $N_f$ is set to be 1000. We use a vocabulary of 50K words (filtered by frequency). Note that the source selector enables KEAG to handle out-of-vocabulary words by generating a word from given text or knowledge.

At both training and test stages, we truncate a passage to 800 words, and limit the length of an answer to 120 words. We train on a single Tesla M40 GPU with the batch size of 16. At test time, answers are generated using beam search with the beam size of 4.

| Model | Rouge-L | Bleu-1 |
|---|---|---|
| BiDAF | 19.42 | 13.03 |
| BiDAF+Seq2Seq | 34.15 | 29.68 |
| S-Net | 42.71 | 36.19 |
| S-Net+Seq2Seq | 46.83 | 39.74 |
| QFS | 40.58 | 39.96 |
| VNET | 45.93 | 41.02 |
| gQA | 45.75 | 41.10 |
| **KEAG** | **51.68** | **45.97** |

Table 1: Metrics of KEAG and QA models disregarding knowledge on the MARCO dataset.

### 7.3 Model Comparisons

Table 1 compares KEAG with the following state-of-the-art extractive/generative QA models, which do not make use of external knowledge:

1. **BiDAF** (Seo et al., 2017): A multi-stage hierarchical process that represents the context at different levels of granularity, and using the bi-directional attention flow mechanism for answer extraction
2. **BiDAF+Seq2Seq**: A BiDAF model followed by an additional sequence-to-sequence model for answer generation
3. **S-Net** (Tan et al., 2018): An extraction-then-synthesis framework to synthesize answers from extracted evidences
4. **S-Net+Seq2Seq**: An S-Net model followed by an additional sequence-to-sequence model for answer generation
5. **QFS** (Nema et al., 2017): A model that adapts the query-focused summarization model to answer generation
6. **VNET** (Wang et al., 2018): An MRC model that enables answer candidates from different

| Model | Rouge-L | Bleu-1 |
|---|---|---|
| gQA w/ KBLSTM | 49.33 | 42.81 |
| gQA w/ CRWE | 49.79 | 43.35 |
| MHPGM | 50.51 | 44.73 |
| **KEAG** | **51.68** | **45.97** |

Table 2: Metrics of KEAG and knowledge-enriched QA models on the MARCO dataset.

passages to verify each other based on their content representations

7. **gQA** (Mitra, 2017): A generative approach to question answering by incorporating the copying mechanism and the coverage vector

Table 1 shows the comparison of QA models in Rouge-L and Bleu-1. From the table we observe that abstractive QA models (e.g., KEAG) are consistently superior to extractive models (e.g., BiDAF) in answer quality. Therefore, abstractive QA models establish a strong base architecture to be enhanced with external knowledge, which motivates this work. Among the abstractive models, gQA can be viewed as a simplification of KEAG, which generates answer words from passages and the vocabulary without the use of knowledge. In addition, KEAG incorporates a stochastic source selector while gQA does not. The result that KEAG significantly outperforms gQA demonstrates the effectiveness of KEAG's architecture and the benefit of knowledge integration.

Table 2 shows the metrics of KEAG in comparison to those of the following state-of-the-art QA models that are adapted to leveraging knowledge:

1. **gQA w/ KBLSTM** (Yang and Mitchell, 2017): KBLSTM is a neural model that leverages continuous representations of knowledge bases to enhance the learning of recurrent neural networks for machine reading. We plug it into gQA to make use of external knowledge for natural answer generation.
2. **gQA w/ CRWE** (Weissenborn, 2017): CRWE is a reading architecture with dynamic integration of background knowledge based on contextual refinement of word embeddings by leveraging supplementary knowledge. We extend gQA with the refined word embedding for this model.
3. **MHPGM** (Bauer et al., 2018): A multi-hop reasoning QA model which fills in the gaps of inference with commonsense knowledge.

| Model | Syntactic | Correct |
|---|---|---|
| gQA | 3.78 | 3.54 |
| gQA w/ KBLSTM | 3.98 | 3.62 |
| gQA w/ CRWE | 3.91 | 3.69 |
| MHPGM | 4.10 | 3.81 |
| **KEAG** | **4.18** | **4.03** |

Table 3: Human evaluation of KEAG and state-of-the-art answer generation models. Scores range in $[1, 5]$.

From Table 2, it can be clearly observed that KEAG performs best with the highest Rouge-L and Bleu-1 scores among the knowledge-enriched answer generation models. The major difference between KEAG and the other models is the way of incorporating external knowledge into a model. *gQA w/ KBLSTM* and *gQA w/ CRWE* extend gQA with the module that consumes knowledge, and MHPGM incorporates knowledge with selectively-gated attention while its decoder does not leverage words from knowledge in answer generation. Different from these models, KEAG utilizes two stochastic selectors to determine when to leverage knowledge and which fact to use. It brings additional gains in exploiting external knowledge to generate abstractive answers.

Since neither Rouge-L nor Bleu-1 can measure the quality of generated answers in terms of their correctness and accuracy, we also conduct human evaluation on Amazon Mechanical Turk. The evaluation assesses the answer quality on grammaticality and correctness. We randomly select 100 questions from the dev set, and ask turkers for ratings in a Likert scale ($\in [1, 5]$) on the generated answers.

Table 3 reports the human evaluation scores of KEAG and state-of-the-art answer generation models. The KEAG model surpasses all the others in generating correct answers syntactically and substantively. In terms of syntactic correctness, KEAG and MHPGM both perform well thanks to their architectures of composing answer text and integrating knowledge. On the other hand, KEAG significantly outperforms all compared models in generating substantively correct answers, which demonstrates its power in exploiting external knowledge.

### 7.4 Ablation Studies

We conduct ablation studies to assess the individual contribution of every component in KEAG. Table 4 reports the performance of the full KEAG

| Ablation | Rouge-L | Bleu-1 |
|---|---|---|
| **Full KEAG** | **51.68** | **45.97** |
| ✗ supplementary knowledge | 49.98 | 44.59 |
| ✗ latent indicators $\mathbf{y}$ | 47.61 | 42.10 |
| ✗ source selector | 38.33 | 36.75 |

Table 4: Ablation tests of KEAG.

model and its ablations.

We evaluate how much incorporating external knowledge as supplementary information contributes to natural answer generation by removing the supplementary knowledge and the corresponding fact selection module from KEAG's architecture. It can be seen that the knowledge component plays an important role in generating high-quality answers, with a drop to 49.98 on Rouge-L after the supplementary knowledge is removed.

To study the effect of our learning method, we further ablate the latent indicators $\mathbf{y}$, which leads to degradation to gQA except that the new model can select answer words from the question source while gQA cannot. Our learning method proves to be effective with a drop of about 5% on Rouge-L and about 6% on Bleu-1 after ablation.

Finally, for ablating the source selector, we have a new model that generates answer words from the vocabulary alone. It results in a significant drop to 38.33 on Rouge-L, confirming its effectiveness in generating natural answers.

### 7.5 Visualization and Interpretation

The source selector allows us to visualize how every word in an answer is generated from one of the sources of the question, passage, vocabulary and knowledge, which gives us insights about how KEAG works.

Table 5 visualizes a sample QA pair from KEAG and which source every word in the answer is selected from (indicated by the sample value of the source selector variable $y_t$). As exemplified in the table, the source distribution $P(y_t|\theta)$ varies over decoding timesteps. To answer the question, at each timestep, KEAG first selects a source based on the sample from $P(y_t|\theta)$, followed by generating an answer word from the selected source. It is observed that the in the generated answer the keyword *personality* comes from the knowledge source which relates psychopathy to personality. The answer word *psychopathy* is selected from the question source, which leads to a well-formed answer with a complete sentence. Another keyword *disorder*, on the other hand, comes from the passage source. This results from reading comprehension of the model on the passage. To generate a final answer in good form, KEAG picks the filler words *is* and *a* as well as the period "." from the vocabulary source. It makes the generated answer semantically correct and comprehensive.

| **Question** | |
|---|---|
| What's psychopathy? | |
| **Answer with source probabilities** | |
| **Question src** | Psychopathy is a personality disorder. |
| **Passage src** | Psychopathy is a personality disorder. |
| **Vocabulary src** | Psychopathy is a personality disorder. |
| **Knowledge src** | Psychopathy is a personality disorder. |
| **Answer colored by source** | |
| Psychopathy is a personality disorder. | |

Table 5: Visualization of a sample QA pair and the source of individual words in the answer. The **Answer with source probabilities** section displays a heatmap on answer words selected from the question, passage, vocabulary and knowledge, respectively. A slot with a higher source probability is highlighted in darker cyan. The **Answer colored by source** section shows the answer in which every word is colored based on the source it was actually selected from. Words in blue come from the question, red from the passage, green from the vocabulary, and orange from the knowledge. The visualization is best viewed in color.

## 8 Conclusion and Future Work

This paper presents a new neural model KEAG that is designed to bring symbolic knowledge from a knowledge base into abstractive answer generation. This architecture employs the source selector that allows for learning an appropriate tradeoff for blending external knowledge with information from textual context. The related fact extraction and stochastic fact selection modules are introduced to complete an answer with relevant facts.

This work opens up for deeper investigation of answer generation models in a targeted way, allowing us to investigate what knowledge sources are required for different domains. In future work, we will explore even tighter integration of symbolic knowledge and stronger reasoning methods.

## References


Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*.

Lisa Bauer, Yicheng Wang, and Mohit Bansal. 2018. Commonsense for generative multi-hop question answering tasks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4220–4230.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545.

Yao Fu and Yansong Feng. 2018. Natural answer generation with heterogeneous memory. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 185–195.

Shen Gao, Zhaochun Ren, Yihong Zhao, Dongyan Zhao, Dawei Yin, and Rui Yan. 2019. Product-aware answer generation in e-commerce question-answering. In *Proceedings of the 12th ACM International Conference on Web Search and Data Mining*.

E.J. Gumbel. 1954. *Statistical theory of extreme values and some practical applications: a series of lectures*. Applied mathematics series. U. S. Govt. Print. Office.

Shizhu He, Cao Liu, Kang Liu, and Jun Zhao. 2017. Generating natural answers by incorporating copying and retrieving mechanisms in sequence-to-sequence learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 199–208.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparametrization with gumbel-softmax. In *Proceedings of the International Conference on Learning Representations*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Proceedings of the ACL Workshop: Text Summarization Braches Out 2004*, page 10.

Teng Long, Emmanuel Bengio, Ryan Lowe, Jackie Chi Kit Cheung, and Doina Precup. 2017. World knowledge for reading comprehension: Rare entity prediction with hierarchical lstms using external descriptions. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 825–834.

Mausam, Michael Schmitz, Robert Bart, Stephen Soderland, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 523–534.

Todor Mihaylov and Anette Frank. 2018. Knowledgeable reader: Enhancing cloze-style reading comprehension with external commonsense knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 821–832.

Rajarshee Mitra. 2017. An abstractive approach to question answering. *CoRR*, abs/1711.06238.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*.

Preksha Nema, Mitesh M. Khapra, Anirban Laha, and Balaraman Ravindran. 2017. Diversity driven attention model for query-based abstractive summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1063–1072. Association for Computational Linguistics.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. In *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS)*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318.

Thomas Pellissier Tanon, Denny Vrandečić, Sebastian Schaffert, Thomas Steiner, and Lydia Pintscher. 2016. From freebase to wikidata: The great migration. In *Proceedings of the 25th International Conference on World Wide Web*, pages 1419–1428.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hananneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *Proceedings of the International Conference on Learning Representations*.

Robert Speer, Joshua Chin, and Catherine Havasi. 2016. Conceptnet 5.5: An open multilingual graph of general knowledge. In *AAAI Conference on Artificial Intelligence*.

Chuanqi Tan, Furu Wei, Nan Yang, Bowen Du, Weifeng Lv, and Ming Zhou. 2018. S-net: From answer extraction to answer synthesis for machine reading comprehension. In *AAAI*.

Yizhong Wang, Kai Liu, Jing Liu, Wei He, Yajuan Lyu, Hua Wu, Sujian Li, and Haifeng Wang. 2018. Multi-passage machine reading comprehension with cross-passage answer verification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1918–1927, Melbourne, Australia. Association for Computational Linguistics.

Dirk Weissenborn. 2017. Dynamic integration of background knowledge in neural NLU systems. *CoRR*, abs/1706.02596.

Bishan Yang and Tom Mitchell. 2017. Leveraging knowledge bases in LSTMs for improving machine reading. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1436–1446.

Jun Yin, Xin Jiang, Zhengdong Lu, Lifeng Shang, Hang Li, and Xiaoming Li. 2016. Neural generative question answering. In *Proceedings of the Workshop on Human-Computer Question Answering*, pages 36–42.

Wanjun Zhong, Duyu Tang, Nan Duan, Ming Zhou, Jiahai Wang, and Jian Yin. 2018. Improving question answering by commonsense-based pre-training. *CoRR*, abs/1809.03568.

# Answering questions by learning to rank - Learning to rank by answering questions

**George-Sebastian Pîrtoacă, Traian Rebedea, Ștefan Rușeți**
University Politehnica of Bucharest, Faculty of Automatic Control and Computers
george.pirtoaca@stud.acs.upb.ro, {traian.rebedea, stefan.ruseti}@cs.pub.ro

## Abstract

Answering multiple-choice questions in a setting in which no supporting documents are explicitly provided continues to stand as a core problem in natural language processing. The contribution of this article is two-fold. First, it describes a method which can be used to semantically rank documents extracted from Wikipedia or similar natural language corpora. Second, we propose a model employing the semantic ranking that holds the first place in two of the most popular leaderboards for answering multiple-choice questions: ARC Easy and Challenge. To achieve this, we introduce a self-attention based neural network that latently learns to rank documents by their importance related to a given question, whilst optimizing the objective of predicting the correct answer. These documents are considered relevant contexts for the underlying question. We have published the ranked documents so that they can be used off-the-shelf to improve downstream decision models.

## 1 Introduction

The article at hand devotes to the problem of answering multiple-choice questions where the input consists of an inquiry expressed solely in raw natural language along with a small set of candidate answers (usually 4) from which only one is correct. Moreover, we are targeting questions from a field of science (e.g. chemistry, biology) as they are distinctly more challenging to answer than regular questions (Clark et al., 2018). A relevant example of such a question is the following:

*Which of the following is an example of a physical change? (A) Lighting a match (B) Breaking a glass (C) Burning of gasoline (D) Rusting of iron*

An important characteristic of all solutions developed for this task is that they are not given explicitly any external information in the form of documents supporting the correct answer or semi-structured information. However, external information is highly desirable, especially domain and common-sense knowledge. Thus, most of the state of the art solutions (Pîrtoacă et al., 2018; Nicula et al., 2018; Ni et al., 2018; Zhang et al., 2018) are using a two-step architecture, as shown in Figure 1. In the first phase, an information retrieval (IR) engine *lexically* searches for relevant supporting paragraphs in Wikipedia and other corpora considered relevant for the task. Having extracted some pertinent paragraphs (usually only one per candidate answer), potentially containing relevant information, a machine learning model is employed in the second step to reason about tuples *(question, candidate answer, external information)*. Various downstream decision models are trained to infer whether the candidate answer is correct given the external information: transformers (Pan et al., 2019), attention models (Clark et al., 2018; Ni et al., 2018), or support vector machines (Clark et al., 2016).

However, it has been previously reported that a retrieval-based engine alone is not able to return relevant documents from the reference corpora (Pîrtoacă et al., 2018; Zhang et al., 2018). Depending on the dataset used as an external reference, about 50% of the questions have insufficient and irrelevant support from a standard retrieval model (Zhang et al., 2018). Therefore, the decision engine that relies on the quality of the supporting documents will be highly affected. Irrelevant documents disturb the training process since the models are trained on "wrong" (or noisy)



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2531–2540, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

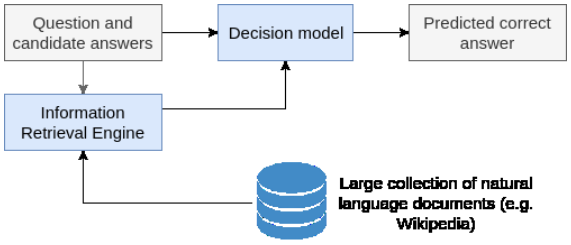


Figure 1: The general high-level architecture of a question answering system

data. As a result, the end-to-end performance of question answering (QA) systems decreases considerably. This article improves the current retrieval scheme by *semantically* ranking the retrieved documents, placing the most relevant ones at the top with a neural self-attention model.

We conduct our experiments on two significant datasets containing science-related multiple-choice questions collected from official examinations: ARC Easy and ARC Challenge (Clark et al., 2018). The former consists of easier questions, whereas the latter incorporates more difficult queries that require inference and external knowledge. In the results section, we are going to observe that our approach achieves state-of-the-art accuracy on both datasets whilst acknowledging their core difference stated by the large gap in the performance: about 28% difference in the accuracy between Easy and Challenge questions.

In this paper, we make the following important contributions:

1) We introduce a neural network architecture, called Attentive Ranker that latently learns to rank supporting documents at a semantic level. The classifier is trained to predict the correct answer to a multiple-choice question achieving state of the art accuracy on the two ARC datasets: Easy (72.30%) and Challenge (44.72%).

2) We show that the set of documents computed by our semantic ranking system can be off-the-shelf adopted by various downstream classifiers to boost their performance (by up to 7%).

## 2 Related Work

A lot of effort has been invested to extract more relevant external documents from natural language corpora to improve QA systems. One approach is to identify the essential terms in questions (Pîrtoacă et al., 2018; Khashabi et al., 2017) and use these terms to improve the quality of the extracted documents. Essential terms are particularly useful for long questions that are intentionally injected with noisy information by teachers to assess the reading comprehension of examined students (which is the case of the ARC dataset, collected indeed from examinations).

Pîrtoacă et al. (2018) propose a neural network architecture, with a small number of parameters, trained to annotate each term in the question with an essentialness score from 0 to 5. The architecture is based on LSTM units (Hochreiter and Schmidhuber, 1997) but with some pre-computed features added to the input to overcome the problem of small datasets: part of speech, a concreteness score for each term (Brysbaert et al., 2014), dependency relations from the parsing tree. The authors show that using the essential terms to extract documents improves the accuracy of multiple-choice QA systems by up to 4% on the ARC dataset (versus a standard IR approach). This highlights that more relevant documents are retrieved using essential terms.

In a similar spirit, Khashabi et al. (2017) report the importance of essential terms for the same multiple-choice QA task but on different datasets. They propose an essential term classifier in the form of a linear Support Vector Machine (SVM) trained on syntactic and semantic features extracted from the question. In total, about 120 types of features (with their combinations) are fed into the classifier. Incorporating the computed essential terms in the IR engine yields an increase in the number of questions correctly answered by up to 5% on the *REGENTS* and the *AI2PUBLIC* datasets[1].

Furthermore, Jansen et al. (2017) formulate the multiple-choice question answering problem into extracting answer justifications and then ranking them. A perceptron is trained to jointly rank the answers along together with their justifications considering the relevance of justifications as a latent variable. The ranking model is tested on a corpus of 1,000 primary school questions, answering 44% of the questions correctly, empirically showing that about 57% of the justifications are meaningful.

On another note, the state of the art downstream decision models are mainly based on transformers (Vaswani et al., 2017) and pre-retrained language models (Devlin et al., 2018; Radford et al., 2019).

[1] Available at http://allenai.org/data.html

Sun et al. (2018) present some reading strategies for machine reading comprehension such as back and forth reading or highlighting. These strategies are carried out by fine-tuning a generative transformer (GPT) (Radford, 2018). Their model was the state of the art approach on the ARC dataset (both Easy and Challenge) prior to our proposed model.

Previous work has shown that the performance of models on QA datasets drops dramatically when a document is not provided to support answering the question. Chen et al. (2017) proposed a model, called DrQA, that was trained on the SQuAD 1.1 dataset (Rajpurkar et al., 2016) to find the correct answer to open-domain questions. When a valid document is provided - guaranteed to contain the correct answer - the exact match (EM) score obtained by DrQA is 69.5. However, when the supporting document has to be retrieved from Wikipedia, by an information retrieval engine, the EM score drops to 27.1. This suggests that research has to be invested in improving retrieval strategies and candidate document ranking methods.

As this related work section suggests, the main direction currently approached for increasing the quality of the extracted documents is to employ essential term information in a form or another and blend this knowledge in the query sent to the retrieval engine. Our paper proposes a completely different strategy, which is based on semantically ranking the extracted documents using a neural network that learns to select the most meaningful and discriminative documents for a given question.

## 3 Proposed approach

We briefly describe how an IR engine is used to extract external information (e.g. supporting documents) from natural language resources like Wikipedia to support QA systems. Then, the paper continues with our proposed approach towards rectifying the shortcomings of the current retrieval-based extraction methods.

### 3.1 Extracting supporting documents

It is undeniable that external information is deeply required in some form or another for answering a given question. Humans capture that information by learning, by experiencing and from common-sense knowledge. For machine learning models, external information is injected as input during training. Given a question and a candidate answer, documents containing information relevant to each *(question, candidate answer)* pair are usually extracted from raw text corpora (Pîrtoacă et al., 2018, Nicula et al., 2018). For the example question mentioned earlier in this paper, a good supporting document would be the following one: *"When a glass is broken, a lot of small sharp glass pieces are formed and spread around."* Given this information, a decision component such as a neural network should be able to deduce that "*breaking a glass"* is indeed an "*example of a physical change"*, thus predicting the correct answer.

Natural language corpora like Wikipedia, ARC Corpus[2], or a large collection of science books crawled from the World Wide Web (Pîrtoacă et al., 2018) are split into articles, paragraphs, or even at the sentence level and then indexed using various search engines: Lucene (Białecki et al., 2012) or ElasticSearch (Gormley and Tong, 2015), to make the entire query process a lot faster. After the indexing phase is completed, queries are sent to the IR engine in the format *"question tokens" AND "answer tokens"*, thus retrieving documents that contain tokens from the question and tokens from the answer (at least one from each) which are, desirably, semantically relevant – and they should be up to one point, depending on the question difficulty and how it is formulated. Notice that for each candidate answer, a set of documents are retrieved and they will be used by some decision engine to derive if the candidate answer is correct or not. In some cases, no decision engine is used at all and the candidate answer with the highest matching score (for example, TF-IDF (Manning et al., 2010)) is predicted as the correct one. During this paper, we will refer to this simple approach as the "*IR baseline*".

In this paper, we will build upon the aforementioned IR baseline by adding a layer on the top of it, which is a neural network capable of latently learning to better rank the documents (as compared to the IR baseline).

[2] http://data.allenai.org/arc/arc-corpus/

### 3.2 Towards a better retrieval engine

An important problem of QA systems is the IR approach for extracting relevant documents. Using a token-based retrieval (Lucene, ElasticSearch) and keeping *the most relevant document* as dictated by a *lexical score* (e.g. TF-IDF, Okapi BM25 (Manning et al., 2010)) does not produce relevant supporting documents for a significant proportion of the questions (Zhang et al., 2018). Our work is a trade-off between the computational performance of such systems and the expressivity of semantic level retrieval. Our main research hypothesis is that semantically ranking the first $N$ (e.g. $N = 100$ or $500$) returned documents instead of sorting them by a lexical metric produces better results and delivers more meaningful documents for answering the question at hand. In other words, some better documents can be found in the first $N$ (where N is a small number), but the best is not always the one with the highest lexical (e.g. TF-IDF) score. We are going to validate our hypothesis in the results section by showing that we can achieve state-of-the-art accuracy on both ARC Easy and Challenge, significantly improving the performance of the same decision model without the proposed semantic ranking.

In order to semantically discern relevant and irrelevant documents for a given question, we are designing a set of discriminators, each receiving a tuple *(question, candidate answer, and document)* and returning a confidence score. A higher score means that the document contains relevant information for answering the question, whereas a score equal to 0 signifies a document that can be ignored as it is noisy or unrelated. These discriminators are the core idea in our approach and they will be used further on to learn how to rank the documents. As will be expanded, these are deep learning models pre-trained to achieve some particular (semantically related) objective. In the next sections, various discriminators are illustrated, and, after that, we present how they are combined to produce the final ranking model.

### 3.3 Document relevance discriminator

This discriminator's purpose is to determine whether or not the document has any significance in answering the question at hand. It ignores the candidate answer and takes into account only the question and the document. The intuition is that some documents are clearly not helpful for the question as they do not contain any relevant information. For example, consider the question: "*How many electrons does a hydrogen atom have?*" and the following two possible extracted documents: "*The hydrogen atom is an electrically neutral atom, usually denoted using the symbol H.*", and "*The hydrogen is a chemical element with a single electron orbiting its nucleus.*" It is clear that the second context is relevant, whilst the first one is not that important. Therefore, in the ranking process, we should place the latter, lower in the list of candidate documents, because semantically it is not relevant for the question at hand, although it has a large TF-IDF score.

| Model | Validation Accuracy |
|---|---|
| GRU with GloVe | 59.04% |
| GRU with ELMo | 60.90% |
| BERT Base | 75.30% |
| BERT Large | 80.43% |

Table 1: Discriminative performance of the models on the adapted SQuAD 2.0 validation set

We have constructed this discriminator by training a neural network on an adapted version of the SQuAD 2.0 dataset (Rajpurkar et al., 2018). The network receives a question and a document as input and produces a score between 0 and 1, correlated with the significance of the document for the question.

The SQuAD 2.0 dataset has the following structure: a paragraph of raw text and a question targeting the information in the paragraph. The answer can be either a span from the paragraph or "*not answerable*", meaning that an answer cannot be deduced based solely on the information in the paragraph. Observe the duality of the task: a "*not answerable*" entry also means that the paragraph is not relevant to the question. This is the key insight that we are going to exploit. We adopted the SQuAD 2.0 dataset, but translated the task into a binary one: is the question answerable or not? Notice that if an answer exists, we pay no attention to the answer itself, the important fact being that the paragraph contains an answer for the question. The resulting dataset has about 100k questions where an answer exists and 50k questions with no valid answer in the given paragraph.

The dataset size is generous enough to enable training deep neural architectures. We have tried four different neural networks to play the role of a document relevance discriminator. The first model encodes the question and the document via 2 layers of independent Gated Recurrent Unit (GRU) cells

(Cho et al., 2014) with the words embedded into a 50-dimensional vector space using pre-trained GloVe word vectors (Pennington et al., 2014). Replacing GRU with Long-Short Term Memory (LSTM) cells (Hochreiter and Schmidhuber, 1997) gives similar performance but the training procedure is computationally more expensive. On top of the representations computed by the GRUs, we added 2 layers of feed-forward connections, the decision phase of the network.

In the second approach, we replaced the first layer of GRUs with the more expressive ELMo encoder (Peters et al., 2018). The other parts of the architecture have not been modified.

Last but not least, we deployed a pre-trained transformer, BERT (Devlin et al., 2018), both the base version with 12 layers and the large version with 24 layers of transformers. BERT is currently the state-of-the-art approach in multiple NLP tasks including open-domain question answering and reading comprehension (Devlin et al., 2018). We fine-tuned the model on the SQuAD 2.0 dataset, with the following hyperparameters: learning rate: 3e-5, warm-up steps: 10% (refer to the BERT paper (Devlin et al., 2018) for further details), sequence length: 425 tokens, batch size: 24 for BERT Large and 10 for BERT base. In order to fine-tune BERT Large, we employed one Tensor Processing Unit (TPU) for about 2 hours. The tokenization has been done following the recommendations in the original paper (Devlin et al., 2018) and the input to the network as the following structure: "*[CLS] question [SEP] supporting document [SEP]*".

These four different discriminators were trained with the objective of discerning between relevant and irrelevant supporting documents for a given question. Their binary accuracies are reported in Table 1. All models were trained on the modified SQuAD 2.0 train split and the performance is reported on the validation dataset. The effectiveness of the BERT Large model is much better than the others, as expected. Therefore, this fine-tuned model is chosen as the final document relevance discriminator (DRD).

### 3.4 Answer verifier discriminator

The purpose of the second discriminator is to probe whether the answer can be inferred to be correct given the supporting document. Of course, we want to rank documents taking into account if they can be used to find the correct answer. Please notice the core difference between the DRD and the answer verifier discriminator (ARD). The latter is also considering the answer at hand, whereas the former only examines the question and the given supporting document.

Training the discriminator is performed on the RACE (Reading Comprehension Dataset) which is also collected from English Examinations (Lai et al., 2017). The structure of the dataset is perfect for our situation. It contains multiple-choice questions with relevant supporting paragraphs for the correct answer. It is guaranteed that the answer can be deduced by understanding the information in the associated paragraph. We transform this dataset into tuples *(question, candidate answer, paragraph)* that are labeled either as negative, meaning that the answer cannot be verified using the paragraph (for incorrect answers), or as positive – reinforcing the fact that the paragraph can be used to infer that the answer is correct. The RACE dataset is extremely suitable in this situation as it always provides a paragraph that is relevant to the question. This is not the case for the ARC dataset, in which no supporting document is given – thus, one has to be extracted and it is not guaranteed to be always the "right" document.

We have fine-tuned the BERT Large model on the joined RACE middle-school and RACE high-school datasets, totaling near 100k questions with 28k documents. Each question generates 4 entries for the ARD: the 3 wrong answers generate 3 negative examples and the correct answer generates one positive example.

The hyperparameters used for fine-tuning BERT are similar to the DRD discriminator, with differences in the maximum sequence length (512 tokens for RACE vs. 425 tokens for SQuAD 2.0). These hyperparameters were found by trying multiple sensible assignments and selecting the best one according to the validation error. Results are reported in the next section on the test split.

Each tuple *(question, answer, and document)* is fed into the BERT model as: *"[CLS] question [SEP] answer [SEP] document [SEP]"*. The final accuracy of the model on the merged RACE middle and high test datasets is 68.28%.

As a remark, we want to highlight that both the DRD and ARD discriminators have been trained on different datasets than the final multiple choice QA model, which is trained and evaluated on the ARC Easy and Challenge datasets. Hence, our intuition was that transfer learning will succeed

even though there are some core differences in the nature of the datasets.

### 3.5 TF-IDF discriminator

We decided to also consider the TF-IDF score computed by Lucene as the third discriminator. This is a lexical (non-semantic) discriminator but it might helpful for simpler questions, especially the ones in the ARC Easy dataset.

### 3.6 Ranking architecture

We have described three discriminators whose purpose is to discern between relevant and useless documents extracted by the IR engine. As mentioned, our proposed approach is to retrieve the first $N$ documents and then rank them taking into account a semantic criterion (as dictated by the pre-trained discriminators). In this section, we are going to describe a neural network that is latently learning to rank the extracted documents whilst selecting the correct answer to the question.

In the ARC dataset, more than 95% of the questions have exactly 4 candidate answers and only one is correct. For a given question and a candidate answer, we pass the extracted $N$ documents though the set of discriminators obtaining a list of scores, one for each discriminator. These scores need to be combined into a final answer score by a mechanism, called Attentive Ranker, inspired from self-attention which we will describe in the next paragraphs.

As shown in Figure 2, each supporting document is associated with a list of scores computed by the discriminators. These scores are projected into a higher dimensional space using a learned projection matrix (we will denote by $D$ the dimension of the projection space – in our experiments, $D = 32$). We then apply an attention mechanism over the semantically labeled documents to select the most relevant ones for a given candidate answer. Let $A$ be a $D$ x $N$ matrix where each column is the encoding of a document after applying the set of discriminators and the projection step. As we are fetching $N$ documents using the IR engine, matrix $A$ has $N$ columns. Notice that this matrix encodes all the information for a given candidate answer. Next, inspired by the self-attention mechanism (Vaswani et al., 2017), we developed a way to relate documents between each other in order to compute a global representation for them. First of all, we project each row of the matrix $A$ into a key space (with $M$

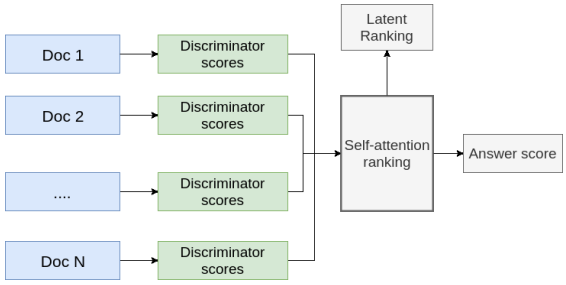


Figure 2: Applying discriminators for a candidate answer and ranking the documents

dimensions) whose purpose is to encode aspects about the quality (relevance) of the associated document (a column in matrix $A$):

$$K = \tanh(W_k A \oplus b_k) \quad (1)$$

where $A \in \mathbb{R}^{DxN}, W_k \in \mathbb{R}^{MxD}, b_k \in \mathbb{R}^M$ and by the $\oplus$ operator, we denote the addition of a vector to each column of a matrix (here and throughout the rest of the paper). Second, each column of matrix $A$ is encoded into a value space (with $Q$ dimensions), which is going to dictate the output of the attention mechanism:

$$V = \text{relu}(W_V A \oplus b_V) \quad (2)$$

where $W_V \in \mathbb{R}^{QxD}, b_V \in \mathbb{R}^Q$. The intuition is that the value space is dictating what the output of the attention should be, whereas the key space encodes necessary information about which elements (documents in our case) the network should pay attention to. Third, we normalize the key vectors using the softmax function:

$$P = softmax(W_P K \oplus b_P) \quad (3)$$

where $W_P \in \mathbb{R}^{1xM}, b_P \in \mathbb{R}$ and the bias is added to each element of $P \in \mathbb{R}^{1xN}$. As a result, the weights in $P$ dictate to what extent the network should attend each document. The output of the layer is the weighted sum of the value vectors:

$$Y = VP^T \quad (4)$$

The same procedure is applied to all candidate answers and the attention weights are shared because we want to have the same representation regardless of the position of the answer in the candidate list (e.g. answer A, B, C, or D). Figure 2 shows the entire encoding procedure and how the attention network is applied.

Observe the fact that for each candidate answer a score (a vector with the dimension of the value space) is obtained by applying a general function. This score is further used to infer which of the four possible answers is, indeed, correct. This decision

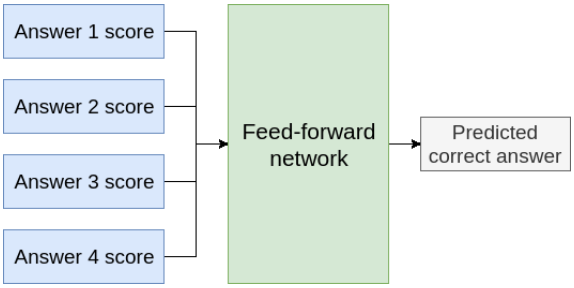


Figure 3: Applying discriminators for a candidate answer and ranking the documents

step is accomplished by a simple feed-forward network (Figure 3).

As a result, one can view this architecture as a feedback loop system. The neural network learns to predict the correct answer. In order to predict the correct answer, it is constrained to select more relevant documents. More relevant documents result in better classification, and so on. Thus, the proposed QA self-attention model, called Attentive Ranker, is jointly *Answering questions by learning to rank* and *Learning to rank by answering questions*.

In the following section, we highlight how the Attentive Ranker can improve QA systems by combining semantic information from a small number of supporting documents extracted with an IR engine.

## 4 Results

We report our model performance on two of the most important multiple-choice science questions datasets: ARC Easy and ARC Challenge (Table 2). During training, due to the small size of the datasets, we experienced a large variance in performance. To partially overcome this issue, we have trained the models a number of times with different random initialization and kept the weights producing the lowest validation loss as the final weights for the QA model.

We have used two knowledge bases in order to retrieve external supporting information: the ARC Corpus (Clark et al., 2018) which contains 14M science-related sentences and a book collection (Pîrtoacă et al., 2018) consisting of about 35 books crawled from online resources such as CK12[3]. Hereinafter, when we mention top $N$ retrieved documents, it means that top $N / 2$ are fetched from ARC Corpus and $N / 2$ are extracted from the book collection, giving the two knowledge bases equal importance.

| Dataset | Train | Dev | Test |
|---|---|---|---|
| ARC Easy | 2,251 | 570 | 2,376 |
| ARC Challenge | 1,119 | 299 | 1,172 |

Table 2: Number of questions in the ARC dataset

| Model | Accuracy |
|---|---|
| Random guess | 25.00% |
| IR Solver | 62.55% |
| Reading Strategies (previous SOTA) | 68.90% |
| ***Attentive Ranker (ours)*** | ***72.30%*** |

Table 3: Results on ARC Easy test

| Model | Accuracy |
|---|---|
| Random guess | 25.00% |
| BERT (our implementation) | 40.00% |
| Reading Strategies | 42.32% |
| BERT (previous SOTA - Microsoft) | 44.62% |
| ***Attentive Ranker (ours)*** | ***44.72%*** |

Table 4: Results on ARC Challenge test

| Dataset | TFD | +DRD | +AVD |
|---|---|---|---|
| ARC Easy | 63.89% | 67.48% | 72.30% |
| ARC Challenge | 26.70% | 34.16% | 44.72% |

Table 5: The impact of adding more discriminators on the test set accuracy

The ranking neural network is trained for about 50 epochs, batch size 128, with categorical cross entropy loss, optimized using Adam (Kingma and Ba, 2014).

First, it is important to observe the impact of the number of documents, $N$, on the model's performance (refer to Figure 4 and Figure 5). Considering more than one document improves the accuracy, thus verifying our initial assumption that relevant supporting documents can be found in the first $N$, but it is not always the case that the first one is the most relevant. As it can be observed in Figures 4 and 5, there is a large improvement in accuracy when increasing from 1 to 5 documents. The accuracy continues to increase for both ARC Easy and ARC Challenge until it reaches a maximum at about 40 documents. Thus, for the next experiments, we set the number of documents to 40 (20 from the ARC corpus and 20 from the science book collection).

[3] https://www.ck12.org/

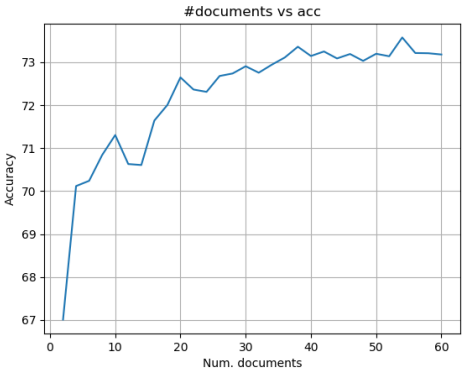


Figure 4: Performance vs. number of documents measured on the ARC Easy validation set

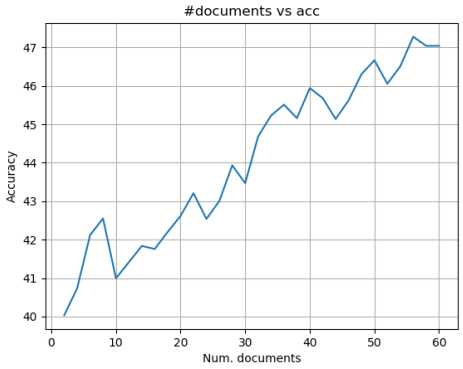


Figure 5: Performance vs. number of documents measured on the ARC Challenge validation set

Second, we verified the model performance on the ARC test sets in order to check how the model generalizes on unseen data and to compare it with other top models in the ARC public leaderboard (https://leaderboard.allenai.org/arc/submissions/public). A summary of the results is reported in Table 3 and Table 4. In both cases, our Attentive Ranker model outperforms the current state-of-the-art (SOTA) approach proving that, indeed, performing a semantic ranking is very effective for QA systems. For a fair comparison, we also reported BERT results as obtained in our implementation and training. The previous SOTA was held by a Microsoft implementation of BERT.

As an ablation study, we wanted to identify the contribution of the discriminators to the overall performance of the Attentive Ranker. We performed an experiment, where starting from the TF-IDF lexical discriminator, other discriminators are incrementally added to the scheme: document relevance discriminator (DRD) and answer verifier discriminator (AVD). The results are revealed in Table 5 where we measured the total number of

| Dataset | # docs | Ranking | Accuracy (Δ) |
|---|---|---|---|
| Val. | Top 1 | TF-IDF | 35.59% |
| Val. | Top 1 | Ours | ***38.30% (+2.71)*** |
| Val. | Top 10 | TF-IDF | 35.93% |
| Val. | Top 10 | Ours | ***43.72% (+7.79)*** |
| Test | Top 1 | TF-IDF | 34.93% |
| Test | Top 1 | Ours | ***37.51% (+3.58)*** |
| Test | Top 10 | TF-IDF | 37.08% |
| Test | Top 10 | Ours | ***40.00% (+2.92)*** |

Table 6: Downstream model performance on the ARC Challenge dataset comparing ranking with Attentive Ranker vs. TF-IDF

| Dataset | # docs | Ranking | Accuracy (Δ) |
|---|---|---|---|
| Val. | Top 1 | doc2vec | 36.61% |
| Val. | Top 1 | Ours | ***38.30% (+1.69)*** |
| Val. | Top 10 | doc2vec | 39.66% |
| Val. | Top 10 | Ours | ***43.72% (+4.06)*** |
| Test | Top 1 | doc2vec | 33.90% |
| Test | Top 1 | Ours | ***37.51% (+3.61)*** |
| Test | Top 10 | dov2vec | 37.85% |
| Test | Top 10 | Ours | ***40.00% (+2.15)*** |

Table 7: Downstream model performance on the ARC Challenge dataset comparing ranking with Attentive Ranker vs. doc2vec

questions correctly answered in the ARC Easy and ARC Challenge test sets. The influence of both semantic-level discriminators (DRD, AVD) is fundamental: they increase the performance with up to 20%, both contributing to the end-to-end accuracy. Notice that the discriminators have a greater impact on the challenge questions as those are radically more difficult and require high-level reasoning to determine the correct answer (that is why the TF-IDF discriminator, which works at a lexical level, has a performance close to random).

One other examination that we enforced is to analyze whether the ranked documents that we computed (semantically sorted by our Attentive Ranker) are more relevant than the documents sorted by the TF-IDF metric. We fine-tuned a downstream BERT (Devlin et al., 2018) model that selects the most probable answer from the candidate list given a list of supporting documents. In the experiment split A, we fed this model with top $N$ documents as ranked by the TF-IDF metric and in experiment split B we fed the model with top $N$ documents as sorted by our ranking neural network (given the weights in the attention layer).

As we have previously shown, the ARC Challenge dataset is more eloquent for this type of semantical analysis due to the difficult questions it includes, thus, we use it for this investigation.

Table 6 shows that using documents ranked by our attentive neural network always leads to a performance increase in downstream models, compared to TF-IDF. On the validation set, the improvement is considerably higher (+7.79) due to a possible over-fitting of the hyperparameters during the Attentive Ranker's training.

We also investigated the ability of the Attentive Ranker to sort the retrieved documents as compared to document embeddings – which may be used to measure the similarity between a question and a candidate document. The question and the documents are embedded using doc2vec (Le and Mikolov, 2014) pre-trained on English Wikipedia. Then, the documents are sorted by their cosine distance to the question. Therefore, the documents that are cosine-closest to a question should be more relevant in answering that question as they are more similar to the question in the doc2vec embeddings space. We would like to note that this approach, although reasonable, may suffer from a subtle problem: the question and the documents have different structures and it may be difficult for an embedding function to capture similarities defined as relevance: "the document is helpful in answering the question". Table 7 illustrates the performance of a downstream model (BERT) in answering the questions when fed with top $N$ (1 or 10) documents as ranked by Attentive Ranker or doc2vec. In all cases, the Attentive Ranker provided more relevant documents than doc2vec, to the BERT-based decision layer. On another hand, doc2vec yields better accuracies than the simple TF-IDF metric.

Our hypothesis that the Attentive Ranker is suitable to rank the retrieved documents by their semantic value to the question is confirmed. Therefore, we decided to make the ranked set of documents public[4], in order to be used by other models. Please refer to the GitHub repository for instructions on how to use the ranked documents. In the same repository, one can find the source code and the full set of trained models.

## 5 Conclusion

In this paper, we highlighted an important problem with many of the current approaches developed for multiple-choice question answering tasks. To overcome the poor performance of the IR engines used to retrieve supporting documents for *(question, candidate answer)* pairs, we described a method that semantically ranks the extracted supporting documents. For this, we proposed an attention-based neural network that latently learns to rank supporting documents by their relevance in answering the given question. The Attentive Ranker architecture depends on the existence of semantic discriminators which are pre-trained to distinguish between relevant and pointless documents. Our proposed model achieves state-of-the-art accuracy on two significant datasets: ARC Easy (72.30%) and ARC Challenge (44.72%).

Furthermore, we have shown that just replacing TF-IDF sorted documents with documents provided by our enhanced ranking method, highly improves the accuracy of various downstream decision models, by up to 7% in our experiments. Therefore, we have made the ranked documents public and further research can benefit from it.

## References


Białecki, A., Muir, R., Ingersoll, G., & Imagination, L. (2012, August). *Apache lucene 4*. In SIGIR 2012 workshop on open source information retrieval (p. 17).

Brysbaert, M., Warriner, A.B., & Kuperman, V. (2014). *Concreteness ratings for 40 thousand generally known English word lemmas*. Behavior research methods, 46 3, 904-11.

Chen, D., Fisch, A., Weston, J., & Bordes, A. (2017). Reading wikipedia to answer open-domain questions. arXiv preprint arXiv:1704.00051.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., & Tafjord, O. (2018). Think you have solved question answering? try arc, the ai2 reasoning challenge. arXiv preprint arXiv:1803.05457.


[4] https://bit.ly/2ZnBNLs

Clark, P., Etzioni, O., Khot, T., Sabharwal, A., Tafjord, O., Turney, P.D., & Khashabi, D. (2016). *Combining Retrieval, Statistics, and Inference to Answer Elementary Science Questions*. AAAI.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Gormley, C., & Tong, Z. (2015). *Elasticsearch: The definitive guide: A distributed real-time search and analytics engine.* O'Reilly Media, Inc.

Hochreiter, S., & Schmidhuber, J. (1997). *Long short-term memory*. Neural computation, 9(8), 1735-1780.

Jansen, P., Sharp, R., Surdeanu, M., & Clark, P. (2017). Framing qa as building and ranking intersentence answer justifications. Computational Linguistics, 43(2), 407-449. https://www.doi.org/10.1162/COLI_a_00287

Khashabi, D., Khot, T., Sabharwal, A., & Roth, D. (2017). Learning What is Essential in Questions. CoNLL.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Lai, G., Xie, Q., Liu, H., Yang, Y., & Hovy, E. (2017). Race: Large-scale reading comprehension dataset from examinations. arXiv preprint arXiv:1704.04683.

Le, Q., & Mikolov, T. (2014, January). *Distributed representations of sentences and documents*. In International conference on machine learning (pp. 1188-1196).

Manning, C., Raghavan, P., & Schütze, H. (2010). *Introduction to information retrieval. Natural Language Engineering*, 16(1), 100-103.

Ni, J., Zhu, C., Chen, W., & McAuley, J. (2018). Learning to attend on essential terms: An enhanced retriever-reader model for scientific question answering. arXiv preprint arXiv:1808.09492.

Nicula, B., Ruseti, S., & Rebedea, T. (2018, March). *Improving Deep Learning for Multiple Choice Question Answering with Candidate Contexts*. In European Conference on Information Retrieval (pp. 678-683). Springer, Cham.

Pan, X., Sun, K., Yu, D., Ji, H., & Yu, D. (2019). Improving Question Answering with External Knowledge. arXiv preprint arXiv:1902.00993.

Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. arXiv preprint arXiv:1802.05365.

Pirtoaca, G., Rebedea, T., & Ruseti, S. (2018). Improving Retrieval-Based Question Answering with Deep Inference Models. CoRR, abs/1812.02971.

Pirtoaca, G.S., Ruseti, S., Rebedea, T. (2018). *Improving multi-choice question answering by identifying essential terms in questions*. In "Revista Romana de Interactiune Om-Calculator"11(2), 145-162, 2018.

Radford, A. (2018). *Improving Language Understanding by Generative Pre-Training*.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). *Language models are unsupervised multitask learners*. OpenAI Blog, 1, 8.

Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016). Squad: 100,000+ questions for machine comprehension of text. arXiv preprint arXiv:1606.05250.

Rajpurkar, P., Jia, R., & Liang, P. (2018). Know What You Don't Know: Unanswerable Questions for SQuAD. arXiv preprint arXiv:1806.03822. translation. arXiv preprint arXiv:1406.1078.

Sun, K., Yu, D., Yu, D., & Cardie, C. (2018). Improving machine reading comprehension with general reading strategies. arXiv preprint arXiv:1810.13441.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). *Attention is all you need*. In Advances in neural information processing systems (pp. 5998-6008).

Zhang, Y., Dai, H., Toraman, K., & Song, L. (2018). KG^ 2: Learning to Reason Science Exam Questions with Contextual Knowledge Graph Embeddings. arXiv preprint arXiv:1805.12393.

# Discourse-Aware Semantic Self-Attention for Narrative Reading Comprehension

**Todor Mihaylov** and **Anette Frank**
Research Training Group AIPHES
Department of Computational Linguistics, Heidelberg University
Heidelberg, Germany
{mihaylov,frank}@cl.uni-heidelberg.de

## Abstract

In this work, we propose to use linguistic annotations as a basis for a *Discourse-Aware Semantic Self-Attention* encoder that we employ for reading comprehension on narrative texts. We extract relations between discourse units, events and their arguments as well as coreferring mentions, using available annotation tools. Our empirical evaluation shows that the investigated structures improve the overall performance (up to +3.4 *Rouge-L*), especially intra-sentential and cross-sentential discourse relations, sentence-internal semantic role relations, and long-distance coreference relations. We show that dedicating self-attention heads to intra-sentential relations and relations connecting neighboring sentences is beneficial for finding answers to questions in longer contexts. Our findings encourage the use of discourse-semantic annotations to enhance the generalization capacity of self-attention models for reading comprehension.

## 1 Introduction

Transformer-based self-attention models (Vaswani et al., 2017) have been shown to work well on many natural language tasks that require large-scale training data, such as Machine Translation (Vaswani et al., 2017; Dai et al., 2019), Language Modeling (Radford et al., 2018a; Devlin et al., 2019; Dai et al., 2019; Radford et al., 2019) or Reading Comprehension (Yu et al., 2018), and can even be trained to perform surprisingly well in several multi-modal tasks (Kaiser et al., 2017b).

Recent work (Strubell et al., 2018) has shown that for downstream semantic tasks with much smaller datasets, such as Semantic Role Labeling (SRL) (Palmer et al., 2005), self-attention models greatly benefit from the use of linguistic information such as dependency parsing annotations. Motivated by this work, we examine to what extent we can use discourse and semantic information to extend self-attention-based neural models for a higher-level task such as Reading Comprehension.

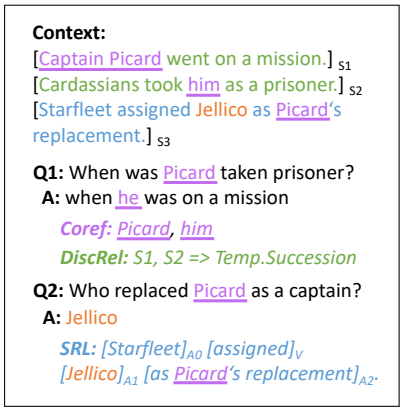


Figure 1: Motivational example: context and questions with required discourse and semantic annotations.

Reading Comprehension is a task that requires a model to answer natural language questions, given a text as context: a paragraph or even full documents. Many datasets have been proposed for the task, starting with a small multi-choice dataset (Richardson et al., 2013), large-scale automatically created cloze-style datasets (Hermann et al., 2015; Hill et al., 2016) and big manually annotated datasets such as Onishi et al. (2016); Rajpurkar et al. (2016); Joshi et al. (2017); Kocisky et al. (2018). Previous research has shown that some datasets are not challenging enough, as simple heuristics work well with them (Chen et al., 2016; Weissenborn et al., 2017b; Chen et al., 2017). In this work we focus on the recent NarrativeQA (Kocisky et al., 2018) dataset that was designed not to be easy to answer and that requires a model to read narrative stories and answer questions about them.

In terms of model architecture, previous work in reading comprehension and question answer-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2541–2552, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics




Figure 2: Example on different discourse-semantic annotations: DiscRel (Dicourse Relations) (NE - Non-Explicit), SRL (Semantic Role Labeling), Coref (Co-reference resolution). The distinct horizontal lines show the interaction between the tokens: Coref - full context, SRL - single sentence, Non-Explicit DR - two neighbouring sentences.

ing has focused on integrating external knowledge (linguistic and/or knowledge-based) into recurrent neural network models using Graph Neural Networks (Song et al., 2018), Graph Convolutional Networks (Sun et al., 2018; De Cao et al., 2019), attention (Das et al., 2017; Mihaylov and Frank, 2018; Bauer et al., 2018) or pointers to coreferent mentions (Dhingra et al., 2017).

In contrast, in this work we examine the impact of *discourse-semantic annotations* (Figure 1) in a *self-attention architecture*. We build on the QANet (Yu et al., 2018) model by modifying the encoder of its self-attention modeling layer. In particular, we *specialize self-attention heads to focus on specific discourse-semantic annotations*, such as, e.g., an ARG1 relation in SRL, a CAUSATION relation holding between clauses in shallow discourse parsing, or coreference relations holding between entity mentions.

Our contributions are the following:

- To our knowledge we are the first to explicitly introduce discourse information into a neural model for reading comprehension.
- We design a *Discourse-Aware Semantic Self-Attention* mechanism, an extension to the standard self-attention models – without significant increase of computation complexity.
- We analyze the impact of different discourse and semantic annotations for narrative reading comprehension and report improvements of up to 3.4 *Rouge-L* over the base model.
- We perform empirical fine-grained evaluation of the discourse-semantic annotations on specific question types and context size.

Code and data will be available at https://github.com/Heidelberg-NLP/discourse-aware-semantic-self-attention.

## 2 Discourse-aware Semantic Annotations

Understanding narrative stories requires the ability to identify events and their participants and to identify how these events are related in discourse (e.g., by *causation, contrast*, or *temporal sequence*) (Mani, 2012). Our aim is to extract structured knowledge about these phenomena from long texts and to integrate this information in a neural self-attention model, in order to examine to what extent such knowledge can enhance the efficiency of a strong reading comprehension model applied to NarrativeQA.

Specifically, we enhance self-attention with knowledge about entity coreference (Coref), their participation in events (SRL) and the relation between events in narrative discourse (Shallow Discourse Parsing (Xue et al., 2016), DR).

All these linguistic information types are *relational* in nature. For integrating relational knowledge into the self-attention mechanism, we follow a two-step approach: i) we extract such relations from a multi-sentence paragraph and *project them down to the token level*, specifically to the tokens of the text fragments that they involve; ii) we design a neural self-attention model that *uses the interaction information between these tokens in a multi-head self-attention module*.

To be able to map the extracted linguistic knowledge to paragraph tokens, we need annotations that are easy to map to token level (see Figure 2). This can be achieved with tools for annotation of span-based Semantic Role Labeling, Coreference Resolution, and Shallow Discourse Parsing.

**Events and Their Participants** Relations between characters in a story are expressed in text through their participation in states or actions in which they fill a particular event argument with a specific semantic role (see Figure 2). For annotation of events and their participants we use the state-of-the-art SRL system of He et al. (2017) as

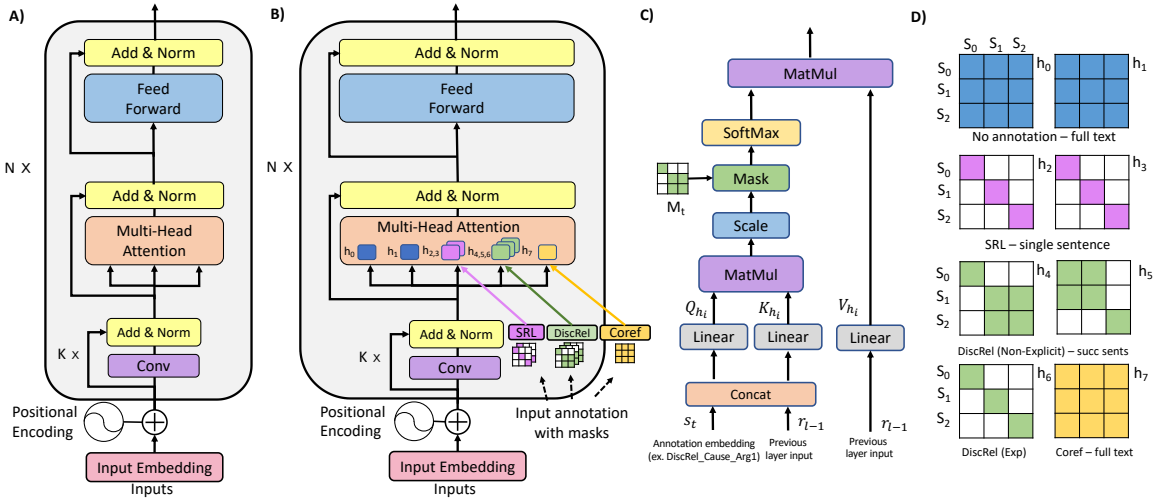


Figure 3: A) Base Multi-Head Self-Attention Encoder Block, B) Discourse-Aware Semantic Self-Attention (DASSA) Encoder Block , C) Single Attention Head with disource/semantic Information, D) Example of attention scope masks for different attention heads and different information.

implemented in AllenNLP (Gardner et al., 2018). The system splits paragraphs into sentences and tokens, performs POS (part of speech tagging) and for each verb token V it predicts semantic tags such as ARG0, ARG1 (Argument Role 0, 1 of verb V), etc. When several argument-taking predicates are realized in a sentence, we obtain more than single semantic argument structure, and each token in the sentence can be involved in the argument structure of more than one verb. We refer to these annotations as different semantic views (Khashabi et al., 2018a), e.g., 'semantic view for verb 1'. Different self-attention heads will be able to attend to individual semantic views.

**Coreference Resolution** Narrative texts abound of entity mentions that refer to the same entity in the discourse. We hypothesize that by directing the self-attention to this specific coreference information, we can encourage the model to focus on tokens that refer to the same entity mention. Although token-based self-attention models are able to attend over wide-ranged context spans, we hypothesize that it will be beneficial to allow the model to focus directly on the parts of the text that refer to the same entity. For coreference annotation we use the *medium* size model from the neuralcoref spaCy extension available at https://github.com/huggingface/neuralcoref. For each token we give as information the label of the corresponding coreference cluster (see Figure 2) that it belongs to. Therefore, tokens from the same coreference cluster get the same label as input.

**Discourse Relations** In narrative texts, events are connected by discourse relations such as *causation, temporal succession*, etc. (Mani, 2012). In this work we adopt the 15 fine-grained discourse relation sense types from the annotation scheme of the Penn Discourse Tree Bank (PDTB) (Prasad et al., 2008). For producing discourse relation annotations we use the discourse relation sense disambiguation system from Mihaylov and Frank (2016) which is trained on the data provided by the CoNLL Shared Task on Shallow Discourse Parsing (Xue et al., 2016). In this annotation scheme discourse relations are divided into two main types: *Explicit* and *Non-Explicit*. *Explicit* relations are usually connected with an explicit *discourse connective*, such as *because, but, if*. *Non-Explicit*[1] relations are not explicitly marked with a discourse connective and the arguments are usually contained in two consecutive sentences (see Figure 2). To extract explicit discourse relations we take into account only arguments that are in the same sentence. We consider as separate arguments (ARG1 and ARG2) text sequences that are on the left and right of an explicit discourse connective (CONN): ex. '[Jeff went home]$_{ARG1\_CCR}$ [because]$_{CONN}$ [he was hungry.]$_{ARG2\_CCR}$, where CCR is Contingency.Cause.Reason'. To provide *Non-Explicit* discourse relation sense annotations, we annotate every consecutive pair of sentences with a predicted discourse relation sense type.

[1] *Non-Explicit* relations include *Implicit*, *AltLex* and *EntRel* relation from PDTB. See Xue et al. (2016) for details.

## 3 A Discourse-Aware Semantic Self-Attention Neural Model

### 3.1 QANet

As a base reading comprehension model we use QANet (Yu et al., 2018). QANet is a standard token-based self-attention model with the following components, which are common across many recent models: *1. Input Embedding Layer* uses pre-trained word embeddings and convolutional character embeddings; *2. Encoder Layer* consists of stacked *Encoder Blocks* (see Figure 3, A) based on *Multi-Head Self-Attention* (Vaswani et al., 2017) and *depth-wise separable convolution* (Chollet, 2016; Kaiser et al., 2017a); *3. Context-to-Query Attention Layer* is a standard layer, that builds a token-wise attention-weighted question-aware context representation; *4. Modeling Layer* has the same structure as 2. above but uses as input the output of layer 3.; *5. Output layer* is used for prediction of *start* and *end* answer pointers. For detailed information about these layers, please refer to Yu et al. (2018). In this work we replace the standard *Multi-Head Self-Attention* with *Discourse-Aware Semantic Self-Attention*, using several different semantic and discourse annotation types. We describe this below and explain the differences to the standard *Encoder Block*.

### 3.2 Discourse-Aware Semantic Self-Attention

In Figure 3 we show the difference between the Base Multi-Head Self-Attention Encoder Block A) and the Discourse-Aware Semantic Self-Attention Encoder Block B). Both consist of $\text{positional-encoding} + \text{convolutional-layer} \times K + \text{multi-head-self-attention} + \text{feed-forward layer}$. The difference is that B is provided additional inputs that are used by multi-head self-attention. The multi-head self-attention is a concatenation of outputs from multiple single self-attention heads $h_i$ followed by a linear layer. A single head of the extended multi-head self-attention is shown in Figure 3C and is formally defined as

$$a_{h_i} = \text{mask\_softmax}\left(\frac{Q_{h_i}K_{h_i}^T}{\sqrt{d_h}}, M_t\right)V_{h_i} \quad (1)$$

$$Q_{h_i} = W_{h_i}^Q[r_{l-1}; s_t] \in \mathbb{R}^{n\times d_h} \quad (2)$$

$$K_{h_i} = W_{h_i}^K[r_{l-1}; s_t] \in \mathbb{R}^{n\times d_h} \quad (3)$$

$$V_{h_i} = W_{h_i}^V r_{l-1} \in \mathbb{R}^{n\times d_h} \quad (4)$$

, where $Q_{h_i}$, $K_{h_i}$, $V_{h_i}$ are components of the query-key-value attention and $\sqrt{d_h}$ is used for weight scaling as originally proposed in Vaswani et al. (2017). $W_{h_i}^Q$, $W_{h_i}^K$, $W_{h_i}^V$ are weights, specific for head $h_i, i \in 1..H$ [2], $r_{l-1}$ is the input from the previous encoder block, $s_t$ is an embedding vector for the linguistic annotation type $t$ ('SRL_Arg1', 'DiscRel_Cause.Reason_Arg2', etc.), $a_{h_i}$ is the output of head $h_i$. $M_t$ is a sentence-wise attention mask as shown in Figure 3D. $s_t$ and $M_t$ are the main difference compared to the standard self-attention (Figure 3C).

In principle, representing edges of a graph (e.g., the V-ARG1 role from SRL) requires memory of $n^2 d_h H$, where $n$ is the length of the context, which would be a bottleneck for computation on a GPU with limited memory (8-16GB). Instead, we adopt a strategy where the *relation* is represented as a *source and target node* and an *attention scope* (one sentence for SRL; two sentences for DR (Non-Exp); full context for Coref). The latter is controlled using the attention mask. The combination of flat token labels and mask reduces the maximum memory required for representing the information in the knowledge-enhanced head to $2nd_hH$. The attention masks, which we use for reducing the attention scope of the different semantic and discourse annotations, are shown in Figure 3D. These masks ensure that the corresponding attention heads will only attend to tokens from the corresponding scope (SRL: single sentence; DR (NonE): two sentences, etc.). The attention masks are symmetric to the matrix diagonal. Therefore, they can easily be computed 'on-the-fly' given only the sentence boundaries (corresponding to the horizontal lines in Figure 2).

To reduce the model memory further and still benefit from the full-context self-attention, we use the *Discourse-Aware Semantic Self-Attention* encoder (Figure 3B) only for blocks [1,3,5] of the *Modeling Layer* that consists of 7 stacked encoder blocks (indexed 0 to 6). Blocks [0,2,6] are set as the base encoders that look at the entire context (Figure 3A).

## 4 Data and Task Description

**NarrativeQA** We perform experiments with the NarrativeQA (Kocisky et al., 2018) reading comprehension dataset. This dataset requires understanding of narrative stories (English) in order to provide answers for a given question. It offers two

[2] Number of heads H=8 as in original QANet specification if not specified otherwise.

sub-tasks: (i) answering questions about a long narrative summary (up to 1150 tokens) of a book or movie, or (ii) answering questions about entire books or movie scripts of lengths up to 110k tokens. We are focusing on the summary setting (i) and refer to the summary as *document* or *context*. The dataset contains 1572 documents in total, devided into *Train* (1102 docs, 32.7k questions), *Dev* (115 documents, 3.5k questions) and *Test* (355 documents, 10.5k questions) sets.

**Generative QA as Span Prediction** An interesting aspect of the NarrativeQA dataset is that in contrast to most other RC datasets, the two answers provided for each question are written by human annotators. Therefore, answers typically differ in form from the context passages that license them. To map the human-generated answers to answer candidate spans from the context, we use *Rouge-L* (Lin, 2004) to calculate a similarity score between token n-grams from the provided answer and token n-grams from candidate answers selected from the context (we select candidate spans of the same length as the given answer). If two answer candidates have the same *Rouge-L* score, we calculate the score between the candidates' surrounding tokens (window size: 15 tokens to the left and right) and the question tokens, and choose the candidate with the higher score. We retrieve the best candidate answer span for each answer and use the candidate with the higher *Rouge-L* score as supervision for training. We refer to this method for answer retrieval as *Oracle (Ours)*.

## 5 Related Work

**Reading Comprehension with Knowledge** Recent work has proposed different approaches for integrating external knowledge into neural models for the high-level downstream tasks reading comprehension (RC) and question answering (QA). One line of work leverages external knowledge from knowledge bases for RC (Xu et al., 2016; Weissenborn et al., 2017a; Ostermann et al., 2018; Mihaylov and Frank, 2018; Bauer et al., 2018; Wang et al., 2018b) and QA (Das et al., 2017; Sun et al., 2018; Tandon et al., 2018). These approaches make use of implicit (Weissenborn et al., 2017a) or explicit (Mihaylov and Frank, 2018; Sun et al., 2018; Bauer et al., 2018) attention-based knowledge aggregation or leverage features from knowledge base relations (Wang et al., 2018b). Another line of work builds on linguistic knowledge from downstream tasks, such as coreference resolution (Dhingra et al., 2017) or notions of co-occurring candidate mentions (De Cao et al., 2019) and OpenIE triples (Khot et al., 2017) into RNN-based encoders. Recently, several pre-trained language models (Peters et al., 2018; Radford et al., 2018b; Devlin et al., 2019) have been shown to incrementally boost the performance of well-performing models for several short paragraph reading comprehension tasks (Peters et al., 2018; Devlin et al., 2019) and question answering (Sun et al., 2019), as well as many tasks from the GLUE benchmark (Wang et al., 2018a). Approaches based on BERT (Devlin et al., 2019) usually perform best when the weights are fine-tuned for the specific training task. Earlier, many papers that do not use self-attention models or even neural methods have also tried to use semantic parse labels (Yih et al., 2016), or annotations from upstream tasks (Khashabi et al., 2018b).

**Self-Attention Models in NLP** Vanilla self-attention models (Vaswani et al., 2017) use positional encoding, sometimes combined with local convolutions (Yu et al., 2018) to model the token order in text. Although they are scalable due to their recurrence-free nature, most self-attention models do not well work when trained with fixed-length context, due to the fact that they often learn global token positions observed during training, rather than relative. To address this issue, Shaw et al. (2018) proposes relative position encoding to model the distance between tokens in the context. Dai et al. (2019) address the problem of moving beyond fixed-length context by adding recurrence to the self-attention model. Dai et al. (2019) argue that the fixed-length segments used for language modeling hurt the performance due to the fact that they do not respect sentence or any other semantic boundaries. In this work we also support the claim that the lack of semantic, and also discourse *boundaries* is an issue, and therefore we aim to introduce structured linguistic information into the self-attention model. We hypothesize that the lack of local discourse context is a problem for answering narrative questions, where the answer is contained inside the same sentence, or neighbouring sentences and therefore, by offering discourse-level semantic structure to the attention heads, offer ways to restrict, or focus the model to wider or narrower structures, depending on what is needed

| Model | B-1 | R-L |
|---|---|---|
| **(Kocisky et al., 2018)** | | |
| Human | 44.43 | 57.02 |
| Oracle (original) | 54.14 | 59.92 |
| Seq2seq (no context) † | 15.89 | 13.15 |
| ASR † | 23.30 | 22.26 |
| BiDAF | 33.72 | 36.30 |
| **Previous work** | | |
| BiAtt + MRU (Tay et al., 2018a) | 36.55 | 41.44 |
| DecaProp (Tay et al., 2018b) | 44.35 | 44.69 |
| MHPGM + NOIC (Bauer et al., 2018) † | 43.63 | 44.16 |
| RMR (Hu et al., 2018b) | 48.40 | 51.50 |
| RMR (Ens) (Hu et al., 2018b) | 50.10 | 53.90 |
| RMR + A2D (Hu et al., 2018b) | 50.40 | 53.30 |
| **This work** | | |
| Oracle (ours) | 70.71 | 70.82 |
| BiDAF | 47.19 | 49.63 |
| QANet | 46.37 | 48.66 |
| + DR (Exp) | 50.12 | 52.14 |
| + DR (Exp) EMA | 51.16 | 53.26 |

Table 1: Results on the NarrativeQA *Test* set. Models with † are generative, while the rest use span prediction.

| Config | DR-E | DR-NE | SRL | Coref | No |
|---|---|---|---|---|---|
| QANet (baseline) | - | - | - | - | 8 |
| DR (All) | 2 | 2 | - | - | 4 |
| DR (Exp) | 2 | - | - | - | 6 |
| DR (NonE) | - | 2 | - | - | 6 |
| Coref | - | - | - | 3 | 5 |
| SRL | - | - | 3 | - | 5 |
| SRL+ DR (Exp) | 2 | - | 3 | - | 3 |
| SRL + DR (NonE) | - | 2 | 3 | - | 3 |
| SRL + DR (All) | 2 | 2 | 3 | - | 1 |
| SRL + DR (Exp) + Coref | 2 | - | 3 | 1 | 2 |
| SRL + DR (All) + Coref | 2 | 2 | 3 | 1 | 4 |

Table 2: The number of attention heads by discourse-semantic type. *'No'* means that no linguistic annotation types are provided (attends to all tokens).

for specific questions.

Self-attention architectures can be seen as graph architectures (imagine the token (node) interactions as adjacency matrix) and are applied to graph problems (Veličković et al., 2018; Li et al., 2019). Therefore, in very recent work Koncel-Kedziorski et al. (2019) have used a self-attention encoder as a graph encoder for text generation, in a dual encoder model. A dual-encoder model similar to Koncel-Kedziorski et al. (2019) is suitable for a setting where the input is knowledge from a graph knowledge-base. For a text-based setting like ours, where word order is important and the tokens are part of semantic arguments, an approach that tries to encode linguistic information in the same architecture (Strubell et al., 2018) is more appropriate. Therefore our method is most related to LISA (Strubell et al., 2018), which uses joint multi-task learning of POS and Dependency Parsing to inject syntactic information for Semantic Role Labeling. In contrast, we do not use multi-task learning, but directly encode semantic information extracted by pre-processing with existing tools.

**NarrativeQA** The summary setting of the NarrativeQA dataset (Kocisky et al., 2018) has in the past been addressed with attention mechanisms by the following models: *BiAtt + MRU* (Tay et al., 2018a) is similar to *BiDAF* (Seo et al., 2017). It is bi-attentive (attends form context-to-query and vice versa) but enhanced with a MRU (Multi-Range Reasoning Units). MRU is a compositional encoder that splits the context tokens into ranges (n-grams) of different sizes and combines them in summed n-gram representations and fully-connected layers. *DecaProp* (Tay et al., 2018b) is a neural architecture for reading comprehension, that densely connects all pairwise layers, modeling relationships between passage and query across all hierarchical levels. Bauer et al. (2018) observed that some of the questions require external commonsense knowledge and developed *MHPGM-NOIC* - a *seq2seq* generative model with a copy mechanism that also uses commonsense knowledge and ELMo (Peters et al., 2018) contextual representations. Hu et al. (2018b) used an implementation of Reinforced Mnemonic Reader (*RMR*) (Hu et al., 2018a). They also proposed *RMR + A2D*, a novel teacher-student attention distillation method to train a model to mirror the behavior of the ensemble model *RMR (Ens)*.

## 6 Experiments and Results

In this section we describe the experiments and results of our proposed model in different configurations. We compare the results of different models using overall results (Table 1) on the dataset, but also the performance for different question types (Figure 4) and context sizes (Figure 5).

### 6.1 Overall Results

Table 1 compares our baselines and proposed model to prior work. We report results for *Bleu-1*, and *Rouge-L* scores. The first section lists results on the NarrativeQA dataset as reported in **Kocisky et al. (2018)**. *Oracle (original)* uses the gold answers as queries to match a token sequence (with the answer length) in the context that

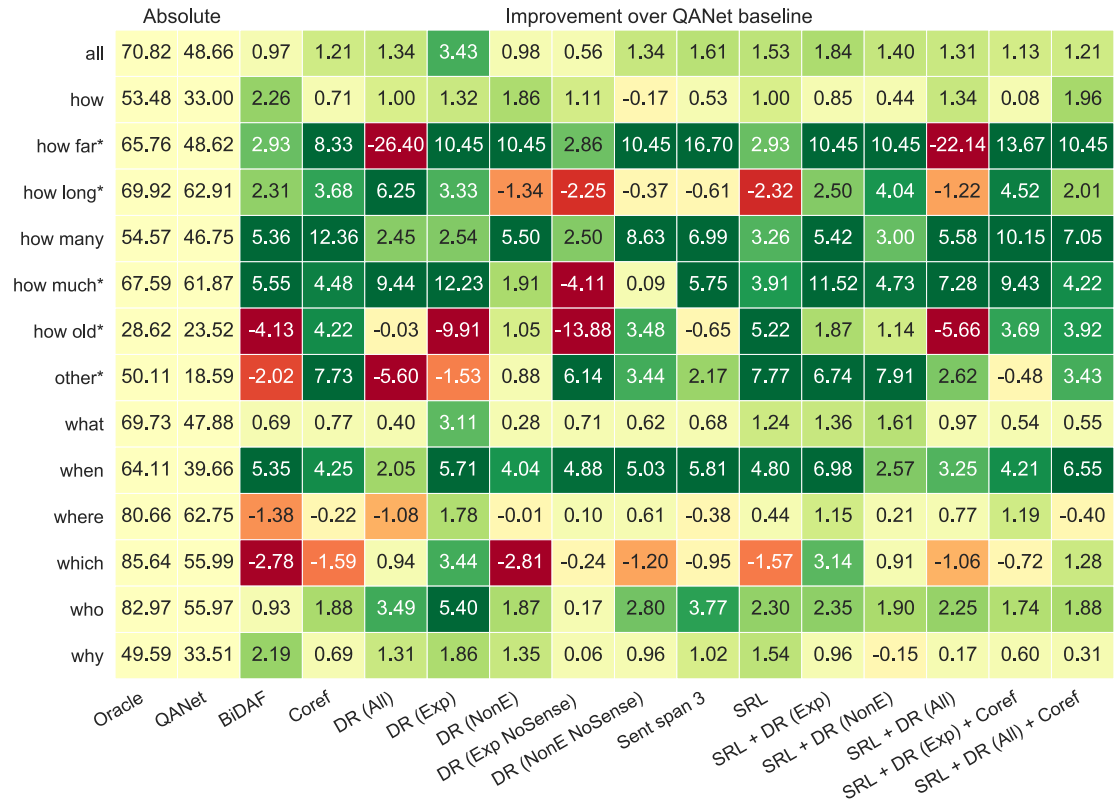


Figure 4: Rouge-L performance per Question Type on the NarrativeQA *Test* set. The first two columns represent *Absolute* values. The rest are improvements over the QANet baseline model (i) by BiDAF and (ii) configurations of QANet with linguistic information. Question types with * have less than 100 instances in the *Test* set.

has the highest *Rouge-L*. In contrast, using *Oracle (Ours)*, described in Section 4, we report a +11 *Rouge-L* score improvement (Table 1: **This work**). The Oracle performance in this setting is important since the produced annotations are used for training of the span-prediction systems, and is considered upper-bound.[3] *Seq2Seq (no context)* is an encoder-decoder RNN model trained only on the question. *ASR* is a version of the Attention Sum Reader (Kadlec et al., 2016) implemented as a pointer-generator that reads the question and points to words in the context that are contained in the answer. *BiDAF* is Bi-Directional Attention Flow (Seo et al., 2017) trained either with the *Oracle (original)* or *Oracle (ours)*. The models from **Previous Work** are described in Section 5. In the last section of Table 1 we present the results of our experiments (**This work**). Here, *BiDAF* and *QANet* are implementations available in the AllenNLP framework (Gardner et al., 2018). In the last two rows we give the results of QANet extended with the proposed Discourse-Aware Semantic Self-Attention, using intra-sentential, *Explicit* discourse relations (*DR (Exp)*, EMA is Exponential Moving Average).

[3]The previous work that uses span-prediction models do not report their *Oracle* model used for training supervision.

### 6.2 Fine-grained Evaluation

We further analyze the performance of different configurations of our model by conducting fine-grained evaluation in view of question types (Figure 4) and context length (Figure 5).

We define a range of system configurations using attention heads enhanced with different combinations of linguistic annotation types, including *Explicit* (referred to as Exp or E) and *Non-Explicit* (NonE, NE), *Discourse Relations* (DiscRel, DR), *Semantic Role Labeling* (SRL), and *Coreference* (Coref), and configurations without any such additional information (*No*). We also experiment with a setting where instead of using specific discourse relation types (such as DiscRel_Exp_Cause_Arg1), we only identify that a token is part of **any** (NoSense) discourse relation (e.g., DiscRel_Exp_Arg1) or simply a multi-sentence attention span *Sent span 3* with labels *Sent1*, *Sent2*, *Sent3* for each sentence. This is

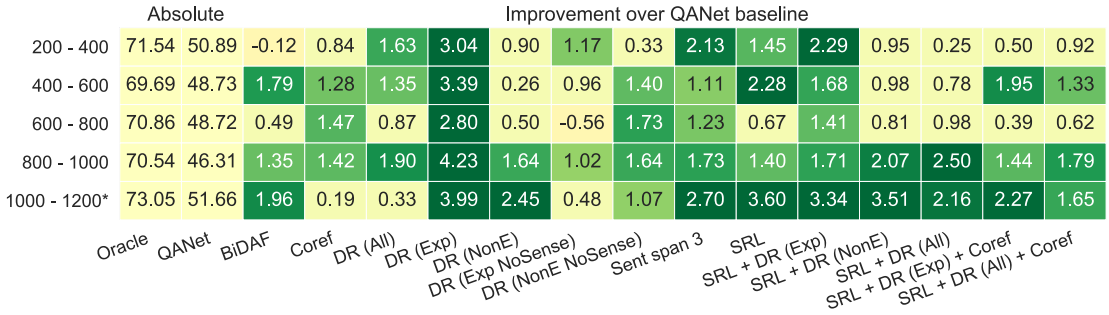


Figure 5: Rouge-L performance by context length on the NarrativeQA *Test* set. The first two columns represent *Absolute* values. The rest are improvements over the QANet baseline model (i) by BiDAF and (ii) different configurations of QANet with linguistic information. Rows with * have less than 100 instances in the *Test* set.

to examine whether the type of discourse relation is important or rather the attention scope (intra-sentential, cross-sentence - 2, 3 neighbouring sentences, full context).

**Question Type** Different question types might profit from different linguistic annotation types. We thus examine the performance of different question types, and analyze how it correlates with the presence of specific Semantic Self-Attention signals. We classify the questions into question types using a simple heuristic based on the question words as an indicator of their type (*How / Where / Why / Who / What ...)*, and calculate the average *Rouge-L* for each such questions type. The resulting scores are displayed in Figure 4. In the first two columns of the figure, we report the *Oracle* score and the baseline (QANet) score. In the remaining columns we report (i) the improvement over the QANet baseline of BiDAF, and (ii) of our models with different combinations of discourse-aware semantic self-attention. In the first row we report the score for each of the models on *all* questions. We observe that best performing models on *all* questions are the ones that include *Explicit* DR, and/or SRL. In terms of hardness, *how* and *why* questions usually have the lowest score. This not surprising since *Oracle* performance is also low. For these type of questions, the RNN-based encoder (BiDAF) and self-attention with DR (Exp) or DR (NonE) perform best. Almost all models with additional linguistic information improve over the baseline on *when* questions, lead by the SRL+DR (Exp) and SRL + DR (All) + Coref. *What* questions are improved most by DR (Exp) and SRL alone or when combined. *Who* questions gain most from discourse relations and all models that contain SRL.

**Context Length** In Figure 5 we present the performance on documents of different lengths, in number of tokens. All presented models are trained on the examples from the *Train* set with context up to 800 tokens. Again, the models DR (Exp) and SRL+DR (Exp) show clear improvement across all context lengths. It is clear that all models show improvement over length *800-1000*. This supports our hypothesis that discourse information is required for generalizing to longer contexts. One reason is that some of the questions can be answered with a local context (one-two sentences) which are better represented given short discourse scope (one-three sentences) or long dependencies given coreference.

In the evaluation of multiple model configurations we notice that in some cases a single discourse/semantic type (e.g. DR (Exp)) performs better than in combination with others (e.g. SRL+DR (Exp)). We hypothesize that the reason is that the linguistic annotations work well in combination with free *No* attention heads (see Table 2). Currently, we place multiple annotations on the same *Encoder Block* which reduces the number of free attention heads. For instance, for SRL+DR (Exp), each knowledge-enhanced encoder block has 3 SRL + 2 DR (Exp) + 3 No heads. In future work we plan to use different annotation heads per *Encoder Block (EB)*: e.g., *EB0* has 3 SRL + 5 No; *EB1* has 2 DR (Exp) + 6 No; etc.

**Success and Failure Examples** In Figures 6, 7, 8 we show examples of context[4] and questions, together with the answers from human annotators and some of the examined models.[5] We provide

[4]The part that contains the correct answer.

[5]For easier reading, we color the gold, correct, and wrong answers and underline the mentions of different characters.

| | |
|---|---|
| **Context** | Although he terrifies the fairies when he first arrives , Peter quickly gains favour with them . He amuses them with his human ways and agrees to play the panpipes at the fairy dances . Eventually , Queen Mab grants him the wish of his heart , and he decides to return home to his mother . |
| **Question** | After scaring the fairies, how does Peter win them over ? |
| **Human 1:** he agrees to play the panpipes at all of the fairy dances.; **Human 2:** He amuses them with his human ways and plays the pipes at their dances.; **Oracle:** human ways and agrees to play the panpipes at the fairy dances ; **QANet:** gains favour ; **DR (Exp), DR (NE):** quickly gains favour with them; **Coref, SRL, SRL+DR(Exp):** He amuses them with his human ways and agrees to play the panpipes; **SRL+DR(NE):** He amuses them with his human ways and agrees to play the panpipes at the fairy dances | |
| **Rationale:** To find the correct answer we need to know that (i) 'gains favor' is a synonym to 'win' in this context (commonsense); (ii) the following (2nd) sentence is the reason for the previous (1st) (DR - the model fails in this case) (iii) 'them' are 'the fairies', 'he' is Peter (Coref) | |

Figure 6: Example of positive impact of SRL and Coref and negative impact from discourse relations (DR).

| | |
|---|---|
| **Context** | Jacob frequently visits Jeff and Kenny , who are serving time in a juvenile hall . Jacob initially threatens them , until eventually Jeff commits suicide . Jacob befriends Kenny , soon learning he has an early release and is illegally moving to New Mexico . |
| **Question** | Why does Jeff committ suicide ? |
| **Human 1:** Jacob threatened them; **Human 2:** He is threatened by Jacob.; **Oracle:** site which he says is ; **QANet:** Jeff and Kenny , who are serving time in a juvenile hall; **DR (Exp), DR (NE), SRL, SRL+DR(Exp), SRL+DR(NE):** Jacob initially threatens them ,; **Coref:** Jacob initially threatens them , until eventually Jeff commits suicide . Jacob befriends Kenny , soon learning he has an early release and is illegally moving to New Mexico | |
| **Rationale:** To find the correct answer we need to understand that 'until eventually' suggests that the suicide of Jeff is caused by Jacob threatening 'them' (DR) and that Jeff is part of 'them' (Coref). | |

Figure 7: Example of positive impact of SRL and Coref, and discourse relations (DR).

| | |
|---|---|
| **Context** | The four orphan children of the house , Edward , Humphrey , Alice and Edith , are believed to have died in the flames . However , they are saved by Jacob Armitage , a local verderer , who hides them in his isolated cottage and disguises them as his grandchildren . Under Armitage 's guidance , the children from an aristocratic lifestyle to that of simple foresters . |
| **Question** | Who rescues the children from fire at Arnwood ? |
| **Human 1, Human 2:** Jacob Armitage; **Oracle:** Jacob Armitage; **DR (Exp), DR (NE), Coref:** Jacob Armitage; **QANet, SRL, SRL+DR(Exp):** Pablo; **SRL+DR(NE):** Patience | |
| **Rationale:** To find the correct answer we need to understand at least that 'they' are 'the children' (Coref) and 'who did what to whom' in the context (SRL). | |

Figure 8: Example of positive impact of Coref and DR and negative impact from SRL.

a hypothetical rationale of what we would need to answer the question. [6]

## 7 Conclusion and Future Work

In this work we use linguistic annotations as a basis for a *Discourse-Aware Semantic Self-Attention* encoder that we employ for reading comprehension on narrative texts.

The provided annotations of discourse relations, events and their arguments as well as coreferring mentions, are using available annotation tools. Our empirical evaluation shows that discourse-semantic annotations combined with self-attention yields significant (+3.43 *Rouge-L*) improvement over QANet's token-based self-attention when applied to NarrativeQA reading comprehension. We analyzed the impact of different semantic annotation types on specific question types and context regions. We find, for instance, that SRL greatly improves *who* and *when* questions, and that discourse relations improve also the performance on *why* and *where* questions. While all examined annotation types contribute, particularly strong and constant gains are seen with intra-sentential DR (all context ranges), followed by SRL (short to mid-sized contexts). Coreference shows positive, but weaker impact, mostly in mid-sized contexts. A promising future direction would be to include additional external knowledge such as commonsense and world knowledge, and learn all annotations jointly with the downstream task.

**Acknowledgments** This work has been supported by the German Research Foundation as part of the Research Training Group Adaptive Preparation of Information from Heterogeneous Sources (AIPHES) under grant No. GRK 1994/1. We thank NVIDIA Corporation for donating GPUs used in this research.

[6] The examples are selected from NarrativeQA *Test*, in such a way, that they depict the strength and weaknesses of the different models, corresponding to the empirical evaluation on Figure 4 and they fit in the space limit.

## References


Lisa Bauer, Yicheng Wang, and Mohit Bansal. 2018. Commonsense for generative multi-hop question answering tasks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4220–4230, Brussels, Belgium.

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 2358–2367, Berlin, Germany.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1870–1879, Vancouver, Canada.

François Chollet. 2016. Xception: Deep learning with depthwise separable convolutions. *CoRR*, abs/1610.02357.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy.

Rajarshi Das, Manzil Zaheer, Siva Reddy, and Andrew McCallum. 2017. Question answering on knowledge bases and text using universal schema and memory networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 358–365.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2019. Question answering by reasoning across documents with graph convolutional networks. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2306–2317, Minneapolis, Minnesota.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota.

Bhuwan Dhingra, Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. 2017. Linguistic knowledge as memory for recurrent neural networks. volume abs/1703.02620.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6, Melbourne, Australia.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 473–483, Vancouver, Canada.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1693–1701. Curran Associates, Inc.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The goldilocks principle: Reading children's books with explicit memory representations. In *ICLR*.

Minghao Hu, Yuxing Peng, Zhen Huang, Xipeng Qiu, Furu Wei, and Ming Zhou. 2018a. Reinforced mnemonic reader for machine reading comprehension. In *Proceedings of International Joint Conferences on Artificial Intelligence Organization*, pages 4099–4106.

Minghao Hu, Yuxing Peng, Furu Wei, Zhen Huang, Dongsheng Li, Nan Yang, and Ming Zhou. 2018b. Attention-guided answer distillation for machine reading comprehension. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2077–2086, Brussels, Belgium.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics (ACL) 2017.

Rudolf Kadlec, Martin Schmid, Ondřej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 908–918. Association for Computational Linguistics.

Lukasz Kaiser, Aidan N. Gomez, and François Chollet. 2017a. Depthwise separable convolutions for neural machine translation. *CoRR*, abs/1706.03059.

Lukasz Kaiser, Aidan N. Gomez, Noam Shazeer, Ashish Vaswani, Niki Parmar, Llion Jones, and Jakob Uszkoreit. 2017b. One model to learn them all. *CoRR*, abs/1706.05137.

Daniel Khashabi, Tushar Khot, Ashish Sabharwal, and Dan Roth. 2018a. Question Answering as Global Reasoning over Semantic Abstractions. In *AAAI*.

Daniel Khashabi, Tushar Khot, Ashutosh Sabharwal, and Dan Roth. 2018b. Question answering as global reasoning over semantic abstractions. In *AAAI*.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2017. Answering complex questions using open information extraction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 311–316.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: a Method for Stochastic Optimization. *International Conference on Learning Representations 2015*, pages 1–15.

Tomas Kocisky, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gabor Melis, and Edward Grefenstette. 2018. The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328.

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text Generation from Knowledge Graphs with Graph Transformers. pages 2284–2293.

Yuan Li, Xiaodan Liang, Zhiting Hu, Yinbo Chen, and Eric P. Xing. 2019. Graph transformer.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain.

Inderjeet Mani. 2012. *Computational Modeling of Narrative*, volume 5.

Todor Mihaylov and Anette Frank. 2016. Discourse relation sense classification using cross-argument semantic similarity based on word embeddings. In *Proceedings of the Twentieth Conference on Computational Natural Language Learning - Shared Task*.

Todor Mihaylov and Anette Frank. 2018. Knowledgeable reader: Enhancing cloze-style reading comprehension with external commonsense knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 821–832, Melbourne, Australia.

Takeshi Onishi, Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. 2016. Who did what: A large-scale person-centered cloze dataset. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2230–2235, Austin, Texas. Association for Computational Linguistics.

Simon Ostermann, Ashutosh Modi, Michael Roth, Stefan Thater, and Manfred Pinkal. 2018. Mcscript: A novel dataset for assessing machine comprehension using script knowledge. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*. European Language Resource Association.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The Penn Discourse treebank 2.0. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC-08)*, Marrakech, Morocco.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018a. GPT: Improving Language Understanding by Generative Pre-Training. *arXiv*, pages 1–12.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018b. Improving Language Understanding by Generative Pre-Training.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Matthew Richardson, Christopher J.C. Burges, and Erin Renshaw. 2013. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 193–203, Seattle, Washington, USA. Association for Computational Linguistics.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hananneh Hajishirzi. 2017. Bi-Directional Attention Flow for Machine Comprehension. In *Proceedings of International Conference of Learning Representations 2017*, pages 1–12.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, New Orleans, Louisiana.

Linfeng Song, Zhiguo Wang, Mo Yu, Yue Zhang, Radu Florian, and Daniel Gildea. 2018. Exploring Graph-structured Passage Representation for Multi-hop Reading Comprehension with Graph Neural Networks.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5027–5038.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen. 2018. Open domain question answering using early fusion of knowledge bases and text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4231–4242.

Kai Sun, Dian Yu, Dong Yu, and Claire Cardie. 2019. Improving machine reading comprehension with general reading strategies. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2633–2643, Minneapolis, Minnesota.

Niket Tandon, Bhavana Dalvi, Joel Grus, Wen-tau Yih, Antoine Bosselut, and Peter Clark. 2018. Reasoning about actions and state changes by injecting commonsense knowledge. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 57–66.

Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2018a. Multi-granular sequence encoding via dilated compositional units for reading comprehension. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2141–2151, Brussels, Belgium. Association for Computational Linguistics.

Yi Tay, Luu Anh Tuan, Siu Cheung Hui, and Jian Su. 2018b. Densely connected attention propagation for reading comprehension. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 4911–4922.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Nips*.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. *International Conference on Learning Representations*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018a. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355.

Liang Wang, Meng Sun, Wei Zhao, Kewei Shen, and Jingming Liu. 2018b. Yuanfudao at semeval-2018 task 11: Three-way attention and relational knowledge for commonsense machine comprehension. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 758–762.

Dirk Weissenborn, Tomas Kocisky, and Chris Dyer. 2017a. Dynamic integration of background knowledge in neural NLU systems. *CoRR*, abs/1706.02596.

Dirk Weissenborn, Georg Wiese, and Laura Seiffe. 2017b. Making neural qa as simple as possible but not simpler. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 271–280. Association for Computational Linguistics.

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Question answering on Freebase via relation extraction and textual evidence. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 2326–2336, Berlin, Germany.

Nianwen Xue, Hwee Tou Ng, Sameer Pradhan, Bonnie Webber, Attapol Rutherford, Chuan Wang, and Hongmin Wang. 2016. The conll-2016 shared task on multilingual shallow discourse parsing. In *Proceedings of the Twentieth Conference on Computational Natural Language Learning - Shared Task*, Berlin, Germany. Association for Computational Linguistics.

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–206, Berlin, Germany.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. 2018. QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension. In *ICLR 2018*.

# Revealing the Importance of Semantic Retrieval for Machine Reading at Scale

**Yixin Nie** **Songhe Wang** **Mohit Bansal**
UNC Chapel Hill
{yixin1, songhe17, mbansal}@cs.unc.edu

## Abstract

Machine Reading at Scale (MRS) is a challenging task in which a system is given an input query and is asked to produce a precise output by "reading" information from a large knowledge base. The task has gained popularity with its natural combination of information retrieval (IR) and machine comprehension (MC). Advancements in representation learning have led to separated progress in both IR and MC; however, very few studies have examined the relationship and combined design of retrieval and comprehension at different levels of granularity, for development of MRS systems. In this work, we give general guidelines on system design for MRS by proposing a simple yet effective pipeline system with special consideration on hierarchical semantic retrieval at both paragraph and sentence level, and their potential effects on the downstream task. The system is evaluated on both fact verification and open-domain multi-hop QA, achieving state-of-the-art results on the leaderboard test sets of both FEVER and HOTPOTQA. To further demonstrate the importance of semantic retrieval, we present ablation and analysis studies to quantify the contribution of neural retrieval modules at both paragraph-level and sentence-level, and illustrate that intermediate semantic retrieval modules are vital for not only effectively filtering upstream information and thus saving downstream computation, but also for shaping upstream data distribution and providing better data for downstream modeling.[1]

## 1 Introduction

Extracting external textual knowledge for machine comprehensive systems has long been an important yet challenging problem. Success requires not only precise retrieval of the relevant information sparsely restored in a large knowledge source but also a deep understanding of both the selected knowledge and the input query to give the corresponding output. Initiated by Chen et al. (2017), the task was termed as Machine Reading at Scale (MRS), seeking to provide a challenging situation where machines are required to do both semantic retrieval and comprehension at different levels of granularity for the final downstream task.

Progress on MRS has been made by improving individual IR or comprehension sub-modules with recent advancements on representative learning (Peters et al., 2018; Radford et al., 2018; Devlin et al., 2018). However, partially due to the lack of annotated data for intermediate retrieval in an MRS setting, the evaluations were done mainly on the final downstream task and with much less consideration on the intermediate retrieval performance. This led to the convention that upstream retrieval modules mostly focus on getting better coverage of the downstream information such that the upper-bound of the downstream score can be improved, rather than finding more exact information. This convention is misaligned with the nature of MRS where equal effort should be put in emphasizing the models' joint performance and optimizing the relationship between the semantic retrieval and the downstream comprehension subtasks.

Hence, to shed light on the importance of semantic retrieval for downstream comprehension tasks, we start by establishing a simple yet effective hierarchical pipeline system for MRS using Wikipedia as the external knowledge source. The system is composed of a term-based retrieval module, two neural modules for both paragraph-level retrieval and sentence-level retrieval, and a neural downstream task module. We evaluated the system on two recent large-scale open do-

[1]Code/data made publicly available at: https://github.com/easonnie/semanticRetrievalMRS



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2553–2566, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

main benchmarks for fact verification and multi-hop QA, namely FEVER (Thorne et al., 2018) and HOTPOTQA (Yang et al., 2018), in which retrieval performance can also be evaluated accurately since intermediate annotations on evidences are provided. Our system achieves the start-of-the-art results with 45.32% for answer EM and 25.14% joint EM on HOTPOTQA (8% absolute improvement on answer EM and doubling the joint EM over the previous best results) and with 67.26% on FEVER score (3% absolute improvement over previously published systems).

We then provide empirical studies to validate design decisions. Specifically, we prove the necessity of both paragraph-level retrieval and sentence-level retrieval for maintaining good performance, and further illustrate that a better semantic retrieval module not only is beneficial to achieving high recall and keeping high upper bound for downstream task, but also plays an important role in shaping the downstream data distribution and providing more relevant and high-quality data for downstream sub-module training and inference. These mechanisms are vital for a good MRS system on both QA and fact verification.

## 2 Related Work

**Machine Reading at Scale** First proposed and formalized in Chen et al. (2017), MRS has gained popularity with increasing amount of work on both dataset collection (Joshi et al., 2017; Welbl et al., 2018) and MRS model developments (Wang et al., 2018; Clark and Gardner, 2017; Htut et al., 2018). In some previous work (Lee et al., 2018), paragraph-level retrieval modules were mainly for improving the recall of required information, while in some other works (Yang et al., 2018), sentence-level retrieval modules were merely for solving the auxiliary sentence selection task. In our work, we focus on revealing the relationship between semantic retrieval at different granularity levels and the downstream comprehension task. To the best of our knowledge, we are the first to apply and optimize neural semantic retrieval at both paragraph and sentence levels for MRS.

**Automatic Fact Checking:** Recent work (Thorne and Vlachos, 2018) formalized the task of automatic fact checking from the viewpoint of machine learning and NLP. The release of FEVER (Thorne et al., 2018) stimulates many recent developments (Nie et al., 2019; Yoneda et al., 2018; Hanselowski et al., 2018) on data-driven neural networks for automatic fact checking. We consider the task also as MRS because they share almost the same setup except that the downstream task is verification or natural language inference (NLI) rather than QA.

**Information Retrieval** Success in deep neural networks inspires their application to information retrieval (IR) tasks (Huang et al., 2013; Guo et al., 2016; Mitra et al., 2017; Dehghani et al., 2017). In typical IR settings, systems are required to retrieve and rank (Nguyen et al., 2016) elements from a collection of documents based on their relevance to the query. This setting might be very different from the retrieval in MRS where systems are asked to select facts needed to answer a question or verify a statement. We refer the retrieval in MRS as *Semantic Retrieval* since it emphasizes on semantic understanding.

## 3 Method

In previous works, an MRS system can be complicated with different sub-components processing different retrieval and comprehension sub-tasks at different levels of granularity, and with some sub-components intertwined. For interpretability considerations, we used a unified pipeline setup. The overview of the system is in Fig. 1.

To be specific, we formulate the MRS system as a function that maps an input tuple $(q, \mathbf{K})$ to an output tuple $(\hat{y}, \mathbf{S})$ where $q$ indicates the input query, $\mathbf{K}$ is the textual KB, $\hat{y}$ is the output prediction, and $\mathbf{S}$ is selected supporting sentences from Wikipedia. Let $\mathbf{E}$ denotes a set of necessary evidences or facts selected from $\mathbf{K}$ for the prediction. For a QA task, $q$ is the input question and $\hat{y}$ is the predicted answer. For a verification task, $q$ is the input claim and $\hat{y}$ is the predicted truthfulness of the input claim. For all tasks, $\mathbf{K}$ is Wikipedia.

The system procedure is listed below:

**(1) Term-Based Retrieval:** To begin with, we used a combination of the TF-IDF method and a rule-based keyword matching method[2] to narrow the scope from whole Wikipedia down to a set of related paragraphs; this is a standard procedure in MRS (Chen et al., 2017; Lee et al., 2018; Nie et al., 2019). The focus of this step is to efficiently select a candidate set $\mathbf{P_I}$ that can cover the information as much as possible ($\mathbf{P_I} \subset \mathbf{K}$) while keeping the

[2]Details of term-based retrieval are in Appendix.

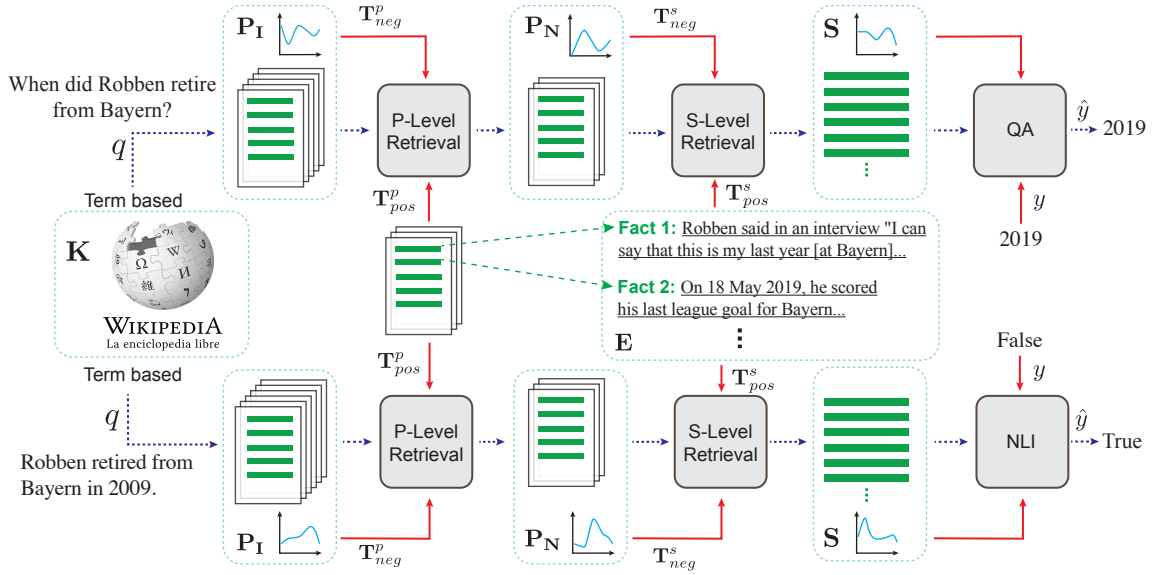


Figure 1: System Overview: blue dotted arrows indicate the inference flow and the red solid arrows indicate the training flow. Grey rounded rectangles are neural modules with different functionality. rounded rectangles. The two retrieval modules were trained with all positive examples from annotated ground truth set and negative examples sampled from the direct upstream modules. Thus, the distribution of negative examples is subjective to the quality of the upstream module.

size of the set acceptable enough for downstream processing.

**(2) Paragraph-Level Neural Retrieval:** After obtaining the initial set, we compare each paragraph in $\mathbf{P_I}$ with the input query $q$ using a neural model (which will be explained later in Sec 3.1). The outputs of the neural model are treated as the relatedness score between the input query and the paragraphs. The scores will be used to sort all the upstream paragraphs. Then, $\mathbf{P_I}$ will be narrowed to a new set $\mathbf{P_N}$ ($\mathbf{P_N} \subset \mathbf{P_I}$) by selecting top $k_p$ paragraphs having relatedness score higher than some threshold value $h_p$ (going out from the P-Level grey box in Fig. 1). $k_p$ and $h_p$ would be chosen by keeping a good balance between the recall and precision of the paragraph retrieval.

**(3) Sentence-Level Neural Retrieval:** Next, we select the evidence at the sentence-level by decomposing all the paragraphs in $\mathbf{P_N}$ into sentences. Similarly, each sentence is compared with the query using a neural model (see details in Sec 3.1) and obtain a set of sentences $\mathbf{S} \subset \mathbf{P_N}$ for the downstream task by choosing top $k_s$ sentences with output scores higher than some threshold $h_s$ (S-Level grey box in Fig. 1). During evaluation, $\mathbf{S}$ is often evaluated against some ground truth sentence set denoted as $\mathbf{E}$.

**(4) Downstream Modeling:** At the final step, we simply applied task-specific neural models (e.g., QA and NLI) on the concatenation of all the sentences in $\mathbf{S}$ and the query, obtaining the final output $\hat{y}$.

In some experiments, we modified the setup for certain analysis or ablation purposes which will be explained individually in Sec 6.

### 3.1 Modeling and Training

Throughout all our experiments, we used **BERT-Base** (Devlin et al., 2018) to provide the state-of-the-art contextualized modeling of the input text.[3]

**Semantic Retrieval:** We treated the neural semantic retrieval at both the paragraph and sentence level as binary classification problems with models' parameters updated by minimizing binary cross entropy loss. To be specific, we fed the query and context into BERT as:

$$[CLS]\ Query\ [SEP]\ Context\ [SEP]$$

We applied an affine layer and sigmoid activation on the last layer output of the $[CLS]$ token which is a scalar value. The parameters were updated with the objective function:

$$\mathcal{J}_{retri} = -\sum_{i \in \mathbf{T}^{p/s}_{pos}} \log(\hat{p}_i) - \sum_{i \in \mathbf{T}^{p/s}_{neg}} \log(1 - \hat{p}_i)$$

[3]We used the pytorch BERT implementation in https://github.com/huggingface/pytorch-pretrained-BERT.

where $\hat{p}_i$ is the output of the model, $\mathbf{T}_{pos}^{p/s}$ is the positive set and $\mathbf{T}_{neg}^{p/s}$ is the negative set. As shown in Fig. 1, at sentence level, ground-truth sentences were served as positive examples while other sentences from upstream retrieved set were served as negative examples. Similarly at the paragraph-level, paragraphs having any ground-truth sentence were used as positive examples and other paragraphs from the upstream term-based retrieval processes were used as negative examples.

**QA:** We followed Devlin et al. (2018) for QA span prediction modeling. To correctly handle yes-or-no questions in HOTPOTQA, we fed the two additional "*yes*" and "*no*" tokens between $[CLS]$ and the $Query$ as:

$$[CLS]\ yes\ no\ Query\ [SEP]\ Context\ [SEP]$$

where the supervision was given to the second or the third token when the answer is "yes" or "no", such that they can compete with all other predicted spans. The parameters of the neural QA model were trained to maximize the log probabilities of the true start and end indexes as:

$$\mathcal{J}_{qa} = -\sum_i \left[\log(\hat{y}_i^s) + \log(\hat{y}_i^e)\right]$$

where $\hat{y}_i^s$ and $\hat{y}_i^e$ are the predicted probability on the ground-truth start and end position for the $i$th example, respectively. It is worth noting that we used ground truth supporting sentences plus some other sentences sampled from upstream retrieved set as the context for training the QA module such that it will adapt to the upstream data distribution during inference.

**Fact Verification:** Following Thorne et al. (2018), we formulate downstream fact verification as the 3-way natural language inference (NLI) classification problem (MacCartney and Manning, 2009; Bowman et al., 2015) and train the model with 3-way cross entropy loss. The input format is the same as that of semantic retrieval and the objective is $\mathcal{J}_{ver} = -\sum_i \mathbf{y}_i \cdot \log(\hat{\mathbf{y}}_i)$, where $\hat{\mathbf{y}}_i \in \mathbf{R^3}$ denotes the model's output for the three verification labels, and $\mathbf{y}_i$ is a one-hot embedding for the ground-truth label. For verifiable queries, we used ground truth evidential sentences plus some other sentences sampled from upstream retrieved set as new evidential context for NLI. For non-verifiable queries, we only used sentences sampled from upstream retrieved set as context because those queries are not associated with ground truth evidential sentences. This detail is important for the model to identify non-verifiable queries and will be explained more in Sec 6. Additional training details and hyper-parameter selections are in the Appendix (Sec. A; Table 6).

It is worth noting that each sub-module in the system relies on its preceding sub-module to provide data both for training and inference. This means that there will be upstream data distribution misalignment if we trained the sub-module in isolation without considering the properties of its precedent upstream module. The problem is similar to the concept of internal covariate shift (Ioffe and Szegedy, 2015), where the distribution of each layer's inputs changes inside a neural network. Therefore, it makes sense to study this issue in a joint MRS setting rather than a typical supervised learning setting where training and test data tend to be fixed and modules being isolated. We release our code and the organized data both for reproducibility and providing an off-the-shelf testbed to facilitate future research on MRS.

## 4 Experimental Setup

MRS requires a system not only to retrieve relevant content from textual KBs but also to poccess enough understanding ability to solve the downstream task. To understand the impact or importance of semantic retrieval on the downstream comprehension, we established a unified experimental setup that involves two different downstream tasks, i.e., multi-hop QA and fact verification.

### 4.1 Tasks and Datasets

**HOTPOTQA:** This dataset is a recent large-scale QA dataset that brings in new features: (1) the questions require finding and reasoning over multiple documents; (2) the questions are diverse and not limited to pre-existing KBs; (3) it offers a new comparison question type (Yang et al., 2018). We experimented our system on HOTPOTQA in the fullwiki setting, where a system must find the answer to a question in the scope of the entire Wikipedia, an ideal MRS setup. The sizes of the train, dev and test split are 90,564, 7,405, and 7,405. More importantly, HOTPOTQA also provides human-annotated sentence-level supporting facts that are needed to answer each question. Those intermediate annotations enable evaluation on models' joint ability on both fact retrieval

and answer span prediction, facilitating our direct analysis on the explainable predictions and its relations with the upstream retrieval.

**FEVER:** The Fact Extraction and VERification dataset (Thorne et al., 2018) is a recent dataset collected to facilitate the automatic fact checking. The work also proposes a benchmark task in which given an arbitrary input claim, candidate systems are asked to select evidential sentences from Wikipedia and label the claim as either SUPPORT, REFUTE, or NOT ENOUGH INFO, if the claim can be verified to be true, false, or non-verifiable, respectively, based on the evidence. The sizes of the train, dev and test split are 145,449, 19,998, and 9,998. Similar to HOTPOTQA, the dataset provides annotated sentence-level facts needed for the verification. These intermediate annotations could provide an accurate evaluation on the results of semantic retrieval and thus suits well for the analysis on the effects of retrieval module on downstream verification.

As in Chen et al. (2017), we use Wikipedia as our unique knowledge base because it is a comprehensive and self-evolving information source often used to facilitate intelligent systems. Moreover, as Wikipedia is the source for both HOTPOTQA and FEVER, it helps standardize any further analysis of the effects of semantic retrieval on the two different downstream tasks.

### 4.2 Metrics

Following Thorne et al. (2018); Yang et al. (2018), we used annotated sentence-level facts to calculate the F1, Precision and Recall scores for evaluating sentence-level retrieval. Similarly, we labeled all the paragraphs that contain any ground truth fact as ground truth paragraphs and used the same three metrics for paragraph-level retrieval evaluation. For HOTPOTQA, following Yang et al. (2018), we used exact match (EM) and F1 metrics for QA span prediction evaluation, and used the joint EM and F1 to evaluate models' joint performance on both retrieval and QA. The joint EM and F1 are calculated as: $P_j = P_a \cdot P_s; R_j = R_a \cdot R_s; F_j = \frac{2P_j \cdot R_j}{P_j + R_j}; \text{EM}_j = \text{EM}_a \cdot \text{EM}_s$, where $P$, $R$, and EM denote precision, recall and EM; the subscript $a$ and $s$ indicate that the scores are for answer span and supporting facts.

For the FEVER task, following Thorne et al. (2018), we used the Label Accuracy for evaluating downstream verification and the Fever Score for joint performance. Fever score will award one point for each example with the correct predicted label only if all ground truth facts were contained in the predicted facts set with at most 5 elements. We also used Oracle Score for the two retrieval modules. The scores were proposed in Nie et al. (2019) and indicate the upperbound of final FEVER Score at one intermediate layer assuming all downstream modules are perfect. All scores are averaged over examples in the whole evaluation set.

| Method | Ans | | Sup | | Joint | |
|---|---|---|---|---|---|---|
| | EM | F1 | EM | F1 | EM | F1 |
| Yang (2018) | 24.7 | 34.4 | 5.3 | 41.0 | 2.5 | 17.7 |
| Ding (2019) | 37.6 | 49.4 | 23.1 | 58.5 | 12.2 | 35.3 |
| whole pip. | **46.5** | **58.8** | **39.9** | **71.5** | **26.6** | **49.2** |
| ***Dev set*** | | | | | | |
| Yang (2018) | 24.0 | 32.9 | 3.9 | 37.7 | 1.9 | 16.2 |
| MUPPET | 30.6 | 40.3 | 16.7 | 47.3 | 10.9 | 27.0 |
| Ding (2019) | 37.1 | 48.9 | 22.8 | 57.7 | 12.4 | 34.9 |
| whole pip. | **45.3** | **57.3** | **38.7** | **70.8** | **25.1** | **47.6** |
| ***Test set*** | | | | | | |

Table 1: Results of systems on HOTPOTQA.

| Model | F1 | LA | FS |
|---|---|---|---|
| Hanselowski (2018) | - | 68.49 | 64.74 |
| Yoneda (2018) | 35.84 | 69.66 | 65.41 |
| Nie (2019) | 51.37 | 69.64 | 66.15 |
| Full system (single) | **76.87** | **75.12** | **70.18** |
| ***Dev set*** | | | |
| Hanselowski (2018) | 37.33 | 65.22 | 61.32 |
| Yoneda (2018) | 35.21 | 67.44 | 62.34 |
| Nie (2019) | 52.81 | 68.16 | 64.23 |
| Full system (single) | **74.62** | **72.56** | **67.26** |
| ***Test set*** | | | |

Table 2: Performance of systems on FEVER. "F1" indicates the sentence-level evidence F1 score. "LA" indicates Label Acc. without considering the evidence prediction. "FS"=FEVER Score (Thorne et al., 2018)

## 5 Results on Benchmarks

We chose the best system based on the dev set, and used that for submitting private test predictions on both FEVER and HOTPOTQA[4].

As can be seen in Table 1, with the proposed hierarchical system design, the whole pipeline sys-

[4]Results can also be found at the leaderboard websites for the two datasets: https://hotpotqa.github.io and https://competitions.codalab.org/competitions/18814

tem achieves new start-of-the-art on HOTPOTQA with large-margin improvements on all the metrics. More specifically, the biggest improvement comes from the EM for the supporting fact which in turn leads to doubling of the joint EM on previous best results. The scores for answer predictions are also higher than all previous best results with ∼8 absolute points increase on EM and ∼9 absolute points on F1. All the improvements are consistent between test and dev set evaluation.

Similarly for FEVER, we showed F1 for evidence, the Label Accuracy, and the FEVER Score (same as benchmark evaluation) for models in Table 2. Our system obtained substantially higher scores than all previously published results with a ∼4 and ∼3 points absolute improvement on Label Accuracy and FEVER Score. In particular, the system gains 74.62 on the evidence F1, 22 points greater that of the second system, demonstrating its ability on semantic retrieval.

Previous systems (Ming Ding, 2019; Yang et al., 2018) on HOTPOTQA treat supporting fact retrieval (sentence-level retrieval) just as an auxiliary task for providing extra model explainability. In Nie et al. (2019), although they used a similar three-stage system for FEVER, they only applied one neural retrieval module at sentence-level which potentially weaken its retrieval ability. Both of these previous best systems are different from our fully hierarchical pipeline approach. These observations lead to the assumption that the performance gain comes mainly from the hierarchical retrieval and its positive effects on downstream. Therefore, to validate the system design decisions in Sec 3 and reveal the importance of semantic retrieval towards downstream, we conducted a series of ablation and analysis experiments on all the modules. We started by examining the necessity of both paragraph and sentence retrieval and give insights on why both of them matters.

## 6 Analysis and Ablations

Intuitively, both the paragraph-level and sentence-level retrieval sub-module help speeding up the downstream processing. More importantly, since downstream modules were trained by sampled data from upstream modules, both of neural retrieval sub-modules also play an implicit but important role in controlling the immediate retrieval distribution i.e. the distribution of set $\mathbf{P_N}$ and set $\mathbf{S}$ (as shown in Fig. 1), and providing better inference data and training data for downstream modules.

### 6.1 Ablation Studies

**Setups:** To reveal the importance of neural retrieval modules at both paragraph and sentence level for maintaining the performance of the overall system, we removed either of them and examine the consequences. Because the removal of a module in the pipeline might change the distribution of the input of the downstream modules, we re-trained all the downstream modules accordingly. To be specific, in the system without the paragraph-level neural retrieval module, we re-trained the sentence-level retrieval module with negative sentences directly sampled from the term-based retrieval set and then also re-trained the downstream QA or verification module. In the system without the sentence-level neural retrieval module, we re-train the downstream QA or verification module by sampling data from both ground truth set and retrieved set directly from the paragraph-level module. We tested the simplified systems on both FEVER and HOTPOTQA.

**Results:** Table 3 and 4 shows the ablation results for the two neural retrieval modules at both paragraph and sentence level on HOTPOTQA and FEVER. To begin with, we can see that removing paragraph-level retrieval module significantly reduces the precision for sentence-level retrieval and the corresponding F1 on both tasks. More importantly, this loss of retrieval precision also led to substantial decreases for all the downstream scores on both QA and verification task in spite of their higher upper-bound and recall scores. This indicates that the negative effects on downstream module induced by the omission of paragraph-level retrieval can not be amended by the sentence-level retrieval module, and focusing semantic retrieval merely on improving the recall or the upper-bound of final score will risk jeopardizing the performance of the overall system.

Next, the removal of sentence-level retrieval module induces a ∼2 point drop on EM and F1 score in the QA task, and a ∼15 point drop on FEVER Score in the verification task. This suggests that rather than just enhance explainability for QA, the sentence-level retrieval module can also help pinpoint relevant information and reduce the noise in the evidence that might otherwise distract the downstream comprehension

| Method | P-Level Retrieval | | | S-Level Retrieval | | | | Answer | | Joint | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Prec. | Rec. | F1 | EM | Prec. | Rec. | F1 | EM | F1 | EM | F1 |
| Whole Pip. | 35.17 | 87.93 | 50.25 | **39.86** | **75.60** | **71.15** | **71.54** | **46.50** | **58.81** | **26.60** | **49.16** |
| Pip. w/o p-level | 6.02 | 89.53 | 11.19 | 0.58 | 29.57 | 60.71 | 38.84 | 31.23 | 41.30 | 0.34 | 19.71 |
| Pip. w/o s-level | 35.17 | 87.92 | 50.25 | - | - | - | - | 44.77 | 56.71 | - | - |

Table 3: Ablation over the paragraph-level and sentence-level neural retrieval sub-modules on HOTPOTQA.

| Method | P-Level Retrieval | | | | S-Level Retrieval | | | | Verification | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Orcl. | Prec. | Rec. | F1 | Orcl. | Prec. | Rec. | F1 | LA | FS | L-F1 (S/R/N) |
| Whole Pip. | 94.15 | 48.84 | 91.23 | 63.62 | 88.92 | **71.29** | 83.38 | 76.87 | **70.18** | **75.01** | 81.7/75.7/**67.1** |
| Pip. w/o p-level | **94.69** | 18.11 | **92.03** | 30.27 | **91.07** | 44.47 | **86.60** | 58.77 | 61.55 | 67.01 | 76.5/72.7/40.8 |
| Pip. w/o s-level | 94.15 | 48.84 | 91.23 | 63.62 | - | - | - | - | 55.92 | 61.04 | 72.1/67.6/27.7 |

Table 4: Ablation over the paragraph-level and sentence-level neural retrieval sub-modules on FEVER. "LA"=Label Accuracy; "FS"=FEVER Score; "Orcl." is the oracle upperbound of FEVER Score assuming all downstream modules are perfect. "L-F1 (S/R/N)" means the classification f1 scores on the three verification labels: SUPPORT, REFUTE, and NOT ENOUGH INFO.

module. Another interesting finding is that without sentence-level retrieval module, the QA module suffered much less than the verification module; conversely, the removal of paragraph-level retrieval neural induces a 11 point drop on answer EM comparing to a ∼9 point drop on Label Accuracy in the verification task. This seems to indicate that the downstream QA module relies more on the upstream paragraph-level retrieval whereas the verification module relies more on the upstream sentence-level retrieval. Finally, we also evaluate the F1 score on FEVER for each classification label and we observe a significant drop of F1 on NOT ENOUGH INFO category without retrieval module, meaning that semantic retrieval is vital for the downstream verification module's discriminative ability on NOT ENOUGH INFO label.

### 6.2 Sub-Module Change Analysis

To further study the effects of upstream semantic retrieval towards downstream tasks, we change training or inference data between intermediate layers and then examine how this modification will affect the downstream performance.

#### 6.2.1 Effects of Paragraph-level Retrieval

We fixed $h_p = 0$ (the value achieving the best performance) and re-trained all the downstream parameters and track their performance as $k_p$ (the number of selected paragraph) being changed from 1 to 12. The increasing of $k_p$ means a potential higher coverage of the answer but more noise in the retrieved facts. Fig. 2 shows the results. As can be seen that the EM scores for supporting fact retrieval, answer prediction, and joint performance increase sharply when $k_p$ is changed from 1 to 2. This is consistent with the fact that at least two paragraphs are required to ask each question in HOTPOTQA. Then, after the peak, every score decrease as $k_p$ becomes larger except the recall of supporting fact which peaks when $k_p = 4$. This indicates that even though the neural sentence-level retrieval module poccesses a certain level of ability to select correct facts from noisier upstream information, the final QA module is more sensitive to upstream data and fails to maintain the overall system performance. Moreover, the reduction on answer EM and joint EM suggests that it might be risky to give too much information for downstream modules with a unit of a paragraph.

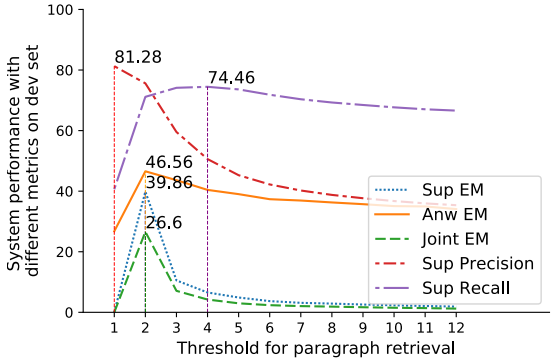


Figure 2: The results of EM for supporting fact, answer prediction and joint score, and the results of supporting fact precision and recall with different values of $k_p$ at paragraph-level retrieval on HOTPOTQA.

#### 6.2.2 Effects of Sentence-level Retrieval

Similarly, to study the effects of neural sentence-level retrieval module towards downstream QA and verification modules, we fixed $k_s$ to be 5 and set $h_s$ ranging from 0.1 to 0.9 with a 0.1 interval.

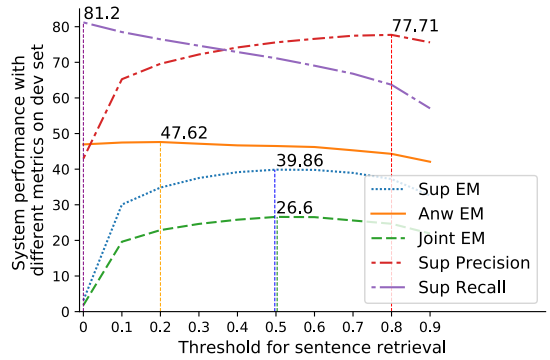


Figure 3: The results of EM for supporting fact, answer prediction and joint score, and the results of supporting fact precision and recall with different values of $h_s$ at sentence-level retrieval on HOTPOTQA.

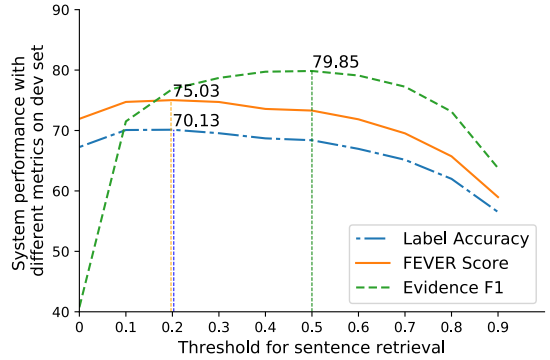


Figure 4: The results of Label Accuracy, FEVER Score, and Evidence F1 with different values of $h_s$ at sentence-level retrieval on FEVER.

Then, we re-trained the downstream QA and verification modules with different $h_s$ value and experimented on both HOTPOTQA and FEVER.

**Question Answering:** Fig. 3 shows the trend of performance. Intuitively, the precision increase while the recall decrease as the system becomes more strict about the retrieved sentences. The EM score for supporting fact retrieval and joint performance reaches their highest value when $h_s = 0.5$, a natural balancing point between precision and recall. More interestingly, the EM score for answer prediction peaks when $h_s = 0.2$ and where the recall is higher than the precision. This misalignment between answer prediction performance and retrieval performance indicates that unlike the observation at paragraph-level, the downstream QA module is able to stand a certain amount of noise at sentence-level and benefit from a higher recall.

**Fact Verification:** Fig. 4 shows the trends for Label Accuracy, FEVER Score, and Evidence F1 by modifying upstream sentence-level threshold $h_s$. We observed that the general trend is similar to that of QA task where both the label accuracy and FEVER score peak at $h_s = 0.2$ whereas the retrieval F1 peaks at $h_s = 0.5$. Note that, although the downstream verification could take advantage of a higher recall, the module is more sensitive to sentence-level retrieval comparing to the QA module in HOTPOTQA. More detailed results are in the Appendix.

| Answer Type | Total | Correct | Acc. (%) |
|---|---|---|---|
| Person | 50 | 28 | 56.0 |
| Location | 31 | 14 | 45.2 |
| Date | 26 | 13 | 50.0 |
| Number | 14 | 4 | 28.6 |
| Artwork | 19 | 7 | 36.8 |
| Yes/No | 17 | 12 | **70.6** |
| Event | 5 | 2 | 40.0 |
| Common noun | 11 | 3 | 27.3 |
| Group/Org | 17 | 6 | 35.3 |
| Other PN | 20 | 9 | 45.0 |
| Total | 200 | 98 | 49.0 |

Table 5: System performance on different answer types. "PN"= Proper Noun

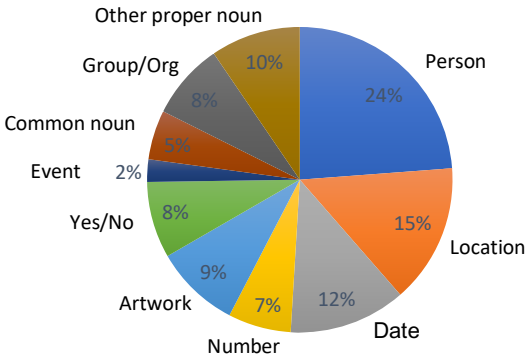


Figure 5: Proportion of answer types.

### 6.3 Answer Breakdown

We further sample 200 examples from HOTPOTQA and manually tag them according to several common answer types (Yang et al., 2018). The proportion of different answer types is shown in Figure 5. The performance of the system on each answer type is shown in Table 5. The most frequent answer type is 'Person' (24%) and the least frequent answer type is 'Event' (2%). It is also interesting to note that the model performs the best in Yes/No questions as shown in Table 5, reaching an accuracy of 70.6%.

***Question:*** Wojtek Wolski played for what team based in the Miami metropolitan area?
***GT Answer:*** Florida Panthers

***GT Facts:***
[**Florida Panthers,0**]: The Florida Panthers are a professional ice hockey team based in the Miami metropolitan area. (*P-Score* : 0.99; *S-Score* : 0.98)
[**Wojtek Wolski,1**]: In the NHL, he has played for the Colorado Avalanche, Phoenix Coyotes, New York Rangers, Florida Panthers, and the Washington Capitals. (*P-Score* : 0.98; *S-Score* : 0.95)

***Distracting Fact:***
[**History of the Miami Dolphins,0**]: The Miami Dolphins are a professional American football franchise based in the Miami metropolitan area. (*P-Score* : 0.56; *S-Score* : 0.97)

***Wrong Answer :*** The Miami Dolphins

Figure 6: An example with a distracting fact. P-Score and S-Score are the retrieval score at paragraph and sentence level respectively. The full pipeline was able to filter the distracting fact and give the correct answer. The wrong answer in the figure was produced by the system without paragraph-level retrieval module.

### 6.4 Examples

Fig. 6 shows an example that is correctly handled by the full pipeline system but not by the system without paragraph-level retrieval module. We can see that it is very difficult to filter the distracting sentence after sentence-level either by the sentence retrieval module or the QA module.

Above findings in both FEVER and HOTPOTQA bring us some important guidelines for MRS: (1) A paragraph-level retrieval module is imperative; (2) Downstream task module is able to undertake a certain amount of noise from sentence-level retrieval; (3) Cascade effects on downstream task might be caused by modification at paragraph-level retrieval.

## 7 Conclusion

We proposed a simple yet effective hierarchical pipeline system that achieves state-of-the-art results on two MRS tasks. Ablation studies demonstrate the importance of semantic retrieval at both paragraph and sentence levels in the MRS system. The work can give general guidelines on MRS modeling and inspire future research on the relationship between semantic retrieval and downstream comprehension in a joint setting.

## Acknowledgments

We thank the reviewers for their helpful comments and Yicheng Wang for useful comments. This work was supported by awards from Verisk, Google, Facebook, Salesforce, and Adobe (plus Amazon and Google GPU cloud credits). The views, opinions, and/or findings contained in this article are those of the authors and should not be interpreted as representing the official views or policies, either expressed or implied, of the funding agency.

## References

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Association for Computational Linguistics (ACL)*.

Christopher Clark and Matt Gardner. 2017. Simple and effective multi-paragraph reading comprehension. *Association for Computational Linguistics (ACL)*.

Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W Bruce Croft. 2017. Neural ranking models with weak supervision. In *SIGIR*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. 2016. A deep relevance matching model for ad-hoc retrieval. In *CIKM*.

Andreas Hanselowski, Hao Zhang, Zile Li, Daniil Sorokin, Benjamin Schiller, Claudia Schulz, and Iryna Gurevych. 2018. Ukp-athene: Multi-sentence textual entailment for claim verification. In *The 1st Workshop on Fact Extraction and Verification*.

Phu Mon Htut, Samuel R Bowman, and Kyunghyun Cho. 2018. Training a ranking function for open-domain question answering. In *NAACL-HLT*.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *CIKM*.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on International Conference on Machine Learning (ICML)*.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly

supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Vancouver, Canada. Association for Computational Linguistics.

Jinhyuk Lee, Seongjun Yun, Hyunjae Kim, Miyoung Ko, and Jaewoo Kang. 2018. Ranking paragraphs for improving answer recall in open-domain question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Bill MacCartney and Christopher D Manning. 2009. *Natural language inference*. Citeseer.

Qibin Chen Hongxia Yang Jie Tang Ming Ding, Chang Zhou. 2019. Cognitive graph for multi-hop reading comprehension at scale. In *Association for Computational Linguistics (ACL)*.

Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to match using local and distributed representations of text for web search. In *WWW*.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

Yixin Nie, Haonan Chen, and Mohit Bansal. 2019. Combining fact extraction and verification with neural semantic matching networks. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *NAACL-HLT*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf*.

James Thorne and Andreas Vlachos. 2018. Automated fact checking: Task formulations, methods and future directions. In *International Conference on Computational Linguistics (COLIN)*.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and VERification. In *NAACL-HLT*.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerry Tesauro, Bowen Zhou, and Jing Jiang. 2018. R: Reinforced ranker-reader for open-domain question answering. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association of Computational Linguistics*, 6:287–302.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Takuma Yoneda, Jeff Mitchell, Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Ucl machine reading group: Four factor framework for fact finding (hexaf). In *The 1st Workshop on Fact Extraction and Verification*.

# Appendix

## A Training Details

| Module | BS. | # E. | $k$ | $h$ |
|---|---|---|---|---|
| P-Level Retri. | 64 | 3 | {2, 5} | {5e-3, 0.01, 0.1, 0.5} |
| S-Level Retri. | 128 | 3 | {2, 5} | [0.1-0.5] |
| QA | 32 | 5 | - | - |
| Verification | 32 | 5 | - | - |

Table 6: Hyper-parameter selection for the full pipeline system. $h$ and $k$ are the retrieval filtering hyper-parameters mentioned in the main paper. P-level and S-level indicate paragraph-level and sentence-level respectively. "{}" means values enumerated from a set. "[]" means values enumerated from a range with interval=0.1 "BS."=Batch Size "# E."=Number of Epochs

The hyper-parameters were chosen based on the performance of the system on the dev set. The hyper-parameters search space is shown in Table 6 and the learning rate was set to $10^{-5}$ in all experiments.

## B Term-Based Retrieval Details

**FEVER** We used the same key-word matching method in Nie et al. (2019) to get a candidate set for each query. We also used TF-IDF (Chen et al., 2017) method to get top-5 related documents for each query. Then, the two sets were combined to get final term-based retrieval set for FEVER. The mean and standard deviation of the number of the retrieved paragraph in the merged set were 8.06 and 4.88.

**HOTPOTQA** We first used the same procedure on FEVER to get an initial candidate set for each query in HOTPOTQA. Because HOTPOTQA requires at least 2-hop reasoning for each query, we

| $h_p$ | S-Level Retrieval | | | | Answer | | Joint | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | EM | Prec. | Rec. | F1 | EM | F1 | EM | F1 | Prec. | Rec. |
| 0 | 3.17 | 42.97 | **81.20** | 55.03 | 46.95 | 59.73 | 1.81 | 37.42 | 29.99 | **57.66** |
| 0.1 | 30.06 | 65.26 | 78.50 | 69.72 | 47.48 | 59.78 | 19.62 | 47.86 | 46.09 | 55.63 |
| 0.2 | 34.83 | 69.59 | 76.47 | 71.28 | **47.62** | **59.93** | 22.89 | 49.15 | 49.24 | 54.53 |
| 0.3 | 37.52 | 72.21 | 74.66 | 71.81 | 47.14 | 59.51 | 24.63 | **49.44** | 50.67 | 53.55 |
| 0.4 | 39.16 | 74.17 | 72.89 | **71.87** | 46.68 | 58.96 | 25.81 | 49.18 | 51.62 | 51.90 |
| 0.5 | **39.86** | 75.60 | 71.15 | 71.54 | 46.50 | 58.81 | **26.60** | 49.16 | 52.59 | 50.83 |
| 0.6 | 39.80 | 76.59 | 69.05 | 70.72 | 46.22 | 58.31 | 26.53 | 48.48 | 52.86 | 49.48 |
| 0.7 | 38.95 | 77.47 | 66.80 | 69.71 | 45.29 | 57.47 | 25.96 | 47.59 | **53.06** | 47.86 |
| 0.8 | 37.22 | **77.71** | 63.70 | 67.78 | 44.30 | 55.99 | 24.67 | 45.92 | 52.41 | 45.32 |
| 0.9 | 32.60 | 75.60 | 57.07 | 62.69 | 42.08 | 52.85 | 21.44 | 42.26 | 50.48 | 40.61 |

Table 7: Detailed Results of downstream sentence-level retrieval and question answering with different values of $h_s$ on HOTPOTQA.

then extract all the hyperlinked documents from the retrieved documents in the initial candidate set, rank them with TF-IDF (Chen et al., 2017) score and then select top-5 most related documents and add them to the candidate set. This gives the final term-based retrieval set for HOTPOTQA. The mean and standard deviation of the number of the retrieved paragraph for each query in HOTPOTQA were 39.43 and 16.05.

## C Detailed Results

- The results of sentence-level retrieval and downstream QA with different values of $h_s$ on HOTPOTQA are in Table 7.
- The results of sentence-level retrieval and downstream verification with different values of $h_s$ on FEVER are in Table 8.
- The results of sentence-level retrieval and downstream QA with different values of $k_p$ on HOTPOTQA are in Table 9.

## D Examples and Case Study

We further provide examples, case study and error analysis for the full pipeline system. The examples are shown from Tables 11, 12, 13, 14, 15. The examples show high diversity on the semantic level and the error occurs often due to the system's failure of extracting precise (either wrong, surplus or insufficient) information from KB.

| $h_s$ | S-Level Retrieval | | | Verification | |
|---|---|---|---|---|---|
| | Precision | Recall | F1 | Label Accuracy | FEVER Score |
| 0 | 26.38 | **89.28** | 40.74 | 71.93 | 67.26 |
| 0.1 | 61.41 | 86.27 | 71.51 | 74.73 | 70.08 |
| 0.2 | 71.29 | 83.38 | 76.87 | **75.03** | **70.13** |
| 0.3 | 76.45 | 81.08 | 78.70 | 74.73 | 69.55 |
| 0.4 | 80.52 | 78.95 | 79.73 | 73.57 | 68.70 |
| 0.5 | 83.76 | 76.31 | **79.85** | 73.30 | 68.39 |
| 0.6 | 86.73 | 72.70 | 79.10 | 71.85 | 66.96 |
| 0.7 | 89.91 | 67.71 | 77.25 | 69.53 | 65.15 |
| 0.8 | 93.22 | 60.16 | 73.13 | 65.73 | 62.00 |
| 0.9 | **96.52** | 47.63 | 63.78 | 58.98 | 56.51 |

Table 8: Results with different $h_s$ on FEVER.

| $k_p$ | S-Level Retrieval | | | | Answer | | Joint | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | EM | Prec. | Rec. | F1 | EM | F1 | EM | F1 | Prec. | Rec. |
| 1 | 0 | **81.28** | 40.64 | 53.16 | 26.77 | 35.76 | 0 | 21.54 | 33.41 | 17.32 |
| 2 | **39.86** | 75.60 | 71.15 | **71.54** | **46.56** | **58.74** | **26.6** | **49.09** | **52.54** | **50.79** |
| 3 | 10.53 | 59.54 | 74.13 | 63.92 | 43.63 | 55.51 | 7.09 | 40.61 | 38.52 | 49.20 |
| 4 | 6.55 | 50.60 | **74.46** | 57.98 | 40.42 | 51.99 | 4.22 | 34.57 | 30.84 | 45.98 |
| 5 | 4.89 | 45.27 | 73.60 | 53.76 | 39.02 | 50.36 | 2.97 | 32.14 | 26.92 | 43.89 |
| 6 | 3.70 | 42.22 | 71.84 | 51.04 | 37.35 | 48.41 | 2.36 | 28.66 | 24.37 | 41.63 |
| 7 | 3.13 | 40.22 | 70.35 | 49.15 | 36.91 | 47.70 | 2.05 | 27.49 | 23.10 | 40.60 |
| 8 | 2.88 | 38.77 | 69.28 | 47.83 | 36.28 | 46.99 | 1.88 | 26.58 | 22.13 | 39.85 |
| 9 | 2.57 | 37.67 | 68.46 | 46.81 | 35.71 | 46.30 | 1.68 | 25.77 | 21.32 | 38.87 |
| 10 | 2.31 | 36.74 | 67.68 | 45.94 | 35.07 | 45.74 | 1.50 | 25.05 | 20.56 | 38.21 |
| 11 | 2.09 | 35.97 | 67.04 | 45.21 | 34.96 | 45.56 | 1.39 | 24.65 | 20.18 | 37.60 |
| 12 | 1.89 | 35.37 | 66.60 | 44.67 | 34.09 | 44.74 | 1.22 | 23.99 | 19.57 | 36.98 |

Table 9: Detailed Results of downstream sentence-level retrieval and question answering with different values of $k_p$ on HOTPOTQA.

| | |
|---|---|
| Question: | *D1NZ is a series based on what oversteering technique?* |
| Ground Truth Facts: | (*D1NZ, 0*) ***D1NZ*** *is a production car drifting series in New Zealand.* (*Drifting (motorsport), 0*) ***Drifting*** *is a driving technique where the driver intentionally oversteers...* |
| Ground Truth Answer: | *Drifting* |
| Predicted Facts: | (*D1NZ, 0*) ***D1NZ*** *is a production car drifting series in New Zealand.* (*Drifting (motorsport), 0*) ***Drifting*** *is a driving technique where the driver intentionally oversteers...* |
| Predicted Answer: | *Drifting* |

Table 10: HotpotQA correct prediction with sufficient evidence.

| | |
|---|---|
| Question: | *The football manager who recruited David Beckham managed Manchester United* during what timeframe? |
| Ground Truth Facts: | (*1995-96 Manchester United F.C. season, 2*) *Alex Ferguson had sold experienced players Paul Ince , Mark Hughes and Andrei Kanchelskis...*<br>(*1995-96 Manchester United F.C. season,3*) *Instead , he had drafted in young players like Nicky Butt, David Beckham, Paul Scholes...*<br>(*Alex Ferguson,0*) *Sir Alexander Chapman Ferguson... who managed Manchester United* ***from 1986 to 2013***. |
| Ground Truth Answer: | *from 1986 to 2013* |
| Predicted Facts: | (*Matt Busby,0*) *Sir Alexander Matthew Busby , CBE... who managed Manchester United between* ***1945 and 1969***... |
| Predicted Answer: | *1945 and 1969* |

Table 11: HotpotQA incorrect prediction with insufficient/wrong evidence.

| | |
|---|---|
| Question: | *Where did Ian Harland study prior to studying at the oldest college at the University of Cambridge?* |
| Ground Truth Facts: | (*Ian Harland,0*) *From a clerical family... Harland was educated at* ***The Dragon School in Oxford*** *and Haileybury.*<br>(*Ian Harland,1*) *He then went to university at Peterhouse, Cambridge, taking a law degree.*<br>(*Peterhouse, Cambridge, 1*) *It is the oldest college of the university...* |
| Ground Truth Answer: | *The Dragon School in Oxford* |
| Predicted Facts: | (*Ian Harland,0*) *From a clerical family... Harland was educated at* ***The Dragon School in Oxford*** *and Haileybury.*<br>(*Ian Harland,1*) *He then went to university at Peterhouse, Cambridge, taking a law degree.*<br>(*Peterhouse, Cambridge, 1*) *It is the oldest college of the university...*<br>(*Ian Harland, 2*) *After two years as a schoolmaster at Sunningdale School he studied for the priesthood at* ***Wycliffe Hall, Oxford*** *and be* |
| Predicted Answer: | *Wycliffe Hall , Oxford* |

Table 12: HotpotQA incorrect prediction caused by extra incorrect information.

| | |
|---|---|
| Claim: | *Heroes' first season had 12 episodes.* |
| Ground Truth Facts: | (*Heroes (U.S. TV series), 8*) *The critically acclaimed first season had a run of 23 episodes and garnered an average of 14.3 million viewers in the United States, receiving the highest rating for an NBC drama premiere in five years.* |
| Ground Truth Label: | *REFUTES* |
| Predicted Facts: | (*Heroes (U.S. TV series), 8*) *The critically acclaimed first season had a run of 23 episodes and garnered an average of 14.3 million viewers in the United States, receiving the highest rating for an NBC drama premiere in five years.* |
| Predicted Label: | *REFUTES* |

Table 13: FEVER correct prediction with sufficient evidence

| | |
|---|---|
| Claim: | *Azithromycin is available as a generic curtain.* |
| Ground Truth Facts: | (*Azithromycin, 17*) *Azithromycin is an antibiotic useful for the treatment of a number of bacterial infections.*<br>(*Azithromycin, 11*) *Azithromycin is an azalide , a type of macrolide antibiotic.*<br>(*Azithromycin, 0*) *It is available as a generic medication and is sold under many trade names worldwide.* |
| Ground Truth Label: | *REFUTES* |
| Predicted Facts: | (*Azithromycin, 17*) *It is available as a generic medication and is sold under many trade names worldwide.* |
| Predicted Label: | *NOT ENOUGH INFO* |

Table 14: FEVER incorrect prediction due to insufficient evidence

| | |
|---|---|
| Claim: | *University of Chicago Law School is ranked fourth by Brain Leiter on the "Top 15 Schools" From Which the Most 'PrestigiousĹaw Firms Hire New Lawyers."* |
| Ground Truth Facts: | (*University of Chicago Law School,6*) *Chicago is ranked second by Brian Leiter of the University of Chicago Law School on the Top 15 Schools From Which the Most Prestigious Law Firms Hire New Lawyers, and first for Faculty quality based on American Academy of Arts and Sciences Membership.* |
| Ground Truth Label: | *REFUTES* |
| Predicted Facts: | (*University of Chicago Law School,6*) *Chicago is ranked second by Brian Leiter of the University of Chicago Law School on the Top 15 Schools From Which the Most Prestigious Law Firms Hire New Lawyers, and first for Faculty quality based on American Academy of Arts and Sciences Membership.*<br>(*University of Chicago Law School,4*) *U.S. News and World Report ranks Chicago fourth among U.S. law schools, and it is noted for its influence on the economic analysis of law.* |
| Predicted Label: | *NOT ENOUGH INFO* |

Table 15: FEVER incorrect prediction due to extra wrong evidence

# PubMedQA: A Dataset for Biomedical Research Question Answering

**Qiao Jin**
University of Pittsburgh
qiao.jin@pitt.edu

**Bhuwan Dhingra**
Carnegie Mellon University
bdhingra@cs.cmu.edu

**Zhengping Liu**
University of Pittsburgh
zliu@pitt.edu

**William W. Cohen**
Google AI
wcohen@google.com

**Xinghua Lu**
University of Pittsburgh
xinghua@pitt.edu

## Abstract

We introduce PubMedQA, a novel biomedical question answering (QA) dataset collected from PubMed abstracts. The task of PubMedQA is to answer research questions with yes/no/maybe (e.g.: *Do preoperative statins reduce atrial fibrillation after coronary artery bypass grafting?*) using the corresponding abstracts. PubMedQA has 1k expert-annotated, 61.2k unlabeled and 211.3k artificially generated QA instances. Each PubMedQA instance is composed of (1) a question which is either an existing research article title or derived from one, (2) a context which is the corresponding abstract without its conclusion, (3) a long answer, which is the conclusion of the abstract and, presumably, answers the research question, and (4) a yes/no/maybe answer which summarizes the conclusion. PubMedQA is the first QA dataset where reasoning over biomedical research texts, especially their quantitative contents, is required to answer the questions. Our best performing model, multi-phase fine-tuning of BioBERT with long answer bag-of-word statistics as additional supervision, achieves 68.1% accuracy, compared to single human performance of 78.0% accuracy and majority-baseline of 55.2% accuracy, leaving much room for improvement. PubMedQA is publicly available at https://pubmedqa.github.io.

## 1 Introduction

A long-term goal of natural language understanding is to build intelligent systems that can reason and infer over natural language. The question answering (QA) task, in which models learn how to answer questions, is often used as a benchmark for quantitatively measuring the reasoning and inferring abilities of such intelligent systems.

While many large-scale annotated general domain QA datasets have been introduced (Rajpurkar et al., 2016; Lai et al., 2017; Kočiskỳ et al., 2018; Yang et al., 2018; Kwiatkowski et al., 2019), the largest annotated biomedical QA dataset, BioASQ (Tsatsaronis et al., 2015) has less than 3k training instances, most of which are simple factual questions. Some works proposed automatically constructed biomedical QA datasets (Pampari et al., 2018; Pappas et al., 2018; Kim et al., 2018), which have much larger sizes. However, questions of these datasets are mostly factoid, whose answers can be extracted in the contexts without much reasoning.

**Question:**
Do preoperative statins reduce atrial fibrillation after coronary artery bypass grafting?
**Context:**
***(Objective)*** Recent studies have demonstrated that statins have pleiotropic effects, including anti-inflammatory effects and atrial fibrillation (AF) preventive effects [...]
***(Methods)*** 221 patients underwent CABG in our hospital from 2004 to 2007. 14 patients with preoperative AF and 4 patients with concomitant valve surgery [...]
***(Results)*** The overall incidence of postoperative AF was 26%. ***Postoperative AF was significantly lower in the Statin group compared with the Non-statin group (16% versus 33%, p=0.005).*** Multivariate analysis demonstrated that independent predictors of AF [...]
**Long Answer:**
***(Conclusion)*** Our study indicated that preoperative statin therapy seems to reduce AF development after CABG.
**Answer:** yes

Figure 1: An instance (Sakamoto et al., 2011) of PubMedQA dataset: Question is the original question title; Context includes the structured abstract except its conclusive part, which serves as the Long Answer; Human experts annotated the Answer yes. Supporting fact for the answer is ***highlighted***.

In this paper, we aim at building a biomedical QA dataset which (1) has substantial instances with some expert annotations and (2) requires reasoning over the contexts to answer the questions. For this, we turn to the PubMed[1], a search engine providing access to over 25 million references of

[1] https://www.ncbi.nlm.nih.gov/pubmed/



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2567–2577, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

biomedical articles. We found that around 760k articles in PubMed use questions as their titles. Among them, the abstracts of about 120k articles are written in a structured style – meaning they have subsections of "Introduction", "Results" etc. Conclusive parts of the abstracts, often in "Conclusions", are the authors' answers to the question title. Other abstract parts can be viewed as the contexts for giving such answers. This pattern perfectly fits the scheme of QA, but modeling it as abstractive QA, where models learn to generate the conclusions, will result in an extremely hard task due to the variability of writing styles.

Interestingly, more than half of the question titles of PubMed articles can be briefly answered by yes/no/maybe, which is significantly higher than the proportions of such questions in other datasets, e.g.: just 1% in Natural Questions (Kwiatkowski et al., 2019) and 6% in HotpotQA (Yang et al., 2018). Instead of using conclusions to answer the questions, we explore answering them with yes/no/maybe and treat the conclusions as a long answer for additional supervision.

To this end, we present PubMedQA, a biomedical QA dataset for answering research questions using yes/no/maybe. We collected all PubMed articles with question titles, and manually labeled 1k of them for cross-validation and testing. An example is shown in Fig. 1. The rest of yes/no/answerable QA instances compose of the unlabeled subset which can be used for semi-supervised learning. Further, we automatically convert statement titles of 211.3k PubMed articles to questions and label them with yes/no answers using a simple heuristic. These artificially generated instances can be used for pre-training. Unlike other QA datasets in which questions are asked by crowd-workers for existing contexts (Rajpurkar et al., 2016; Yang et al., 2018; Kočiskỳ et al., 2018), in PubMedQA contexts are generated to answer the questions and both are written by the same authors. This consistency assures that contexts are perfectly related to the questions, thus making PubMedQA an ideal benchmark for testing scientific reasoning abilities.

As an attempt to solve PubMedQA and provide a strong baseline, we fine-tune BioBERT (Lee et al., 2019) on different subsets in a multi-phase style with additional supervision of long answers. Though this model generates decent results and vastly outperforms other baselines, it's still much worse than the single-human performance, leaving significant room for future improvements.

## 2 Related Works

**Biomedical QA:** Expert-annotated biomedical QA datasets are limited by scale due to the difficulty of annotations. In 2006 and 2007, TREC[2] held QA challenges on genomics corpus (Hersh et al., 2006, 2007), where the task is to retrieve relevant documents for 36 and 38 topic questions, respectively. QA4MRE (Peñas et al., 2013) included a QA task about Alzheimer's disease (Morante et al., 2012). This dataset has 40 QA instances and the task is to answer a question related to a given document using one of five answer choices. The QA task of BioASQ (Tsatsaronis et al., 2015) has phases of (a) retrieve question-related documents and (b) using related documents as contexts to answer yes/no, factoid, list or summary questions. BioASQ 2019 has a training set of 2,747 QA instances and a test set of 500 instances.

Several large-scale automatically collected biomedical QA datasets have been introduced: emrQA (Pampari et al., 2018) is an extractive QA dataset for electronic medical records (EHR) built by re-purposing existing annotations on EHR corpora. BioRead (Pappas et al., 2018) and BMKC (Kim et al., 2018) both collect cloze-style QA instances by masking biomedical named entities in sentences of research articles and using other parts of the same article as context.

**Yes/No QA:** Datasets such as HotpotQA (Yang et al., 2018), Natural Questions (Kwiatkowski et al., 2019), ShARC (Saeidi et al., 2018) and BioASQ (Tsatsaronis et al., 2015) contain yes/no questions as well as other types of questions. BoolQ (Clark et al., 2019) specifically focuses on naturally occurring yes/no questions, and those questions are shown to be surprisingly difficult to answer. We add a "maybe" choice in PubMedQA to cover uncertain instances.

Typical neural approaches to answering yes/no questions involve encoding both the question and context, and decoding the encoding to a class output, which is similar to the well-studied natural language inference (NLI) task. Recent breakthroughs of pre-trained language models like ELMo (Peters et al., 2018) and BERT (Devlin et al., 2018) show significant performance im-

[2] https://trec.nist.gov/

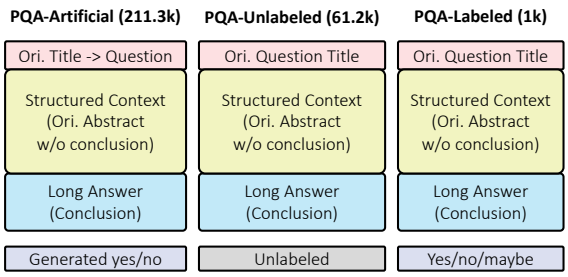


Figure 2: Architecture of PubMedQA dataset. PubMedQA is split into three subsets, PQA-A(rtificial), PQA-U(nlabeled) and PQA-L(abeled).

provements on NLI tasks. In this work, we use domain specific versions of them to set baseline performance on PubMedQA.

## 3 PubMedQA Dataset

### 3.1 Data Collection

PubMedQA is split into three subsets: labeled, unlabeled and artificially generated. They are denoted as PQA-L(abeled), PQA-U(nlabeled) and PQA-A(rtificial), respectively. We show the architecture of PubMedQA dataset in Fig. 2.

| Statistic | PQA-L | PQA-U | PQA-A |
|---|---|---|---|
| Number of QA pairs | 1.0k | 61.2k | 211.3k |
| Prop. of yes (%) | 55.2 | – | 92.8 |
| Prop. of no (%) | 33.8 | – | 7.2 |
| Prop. of maybe (%) | 11.0 | – | 0.0 |
| Avg. question length | 14.4 | 15.0 | 16.3 |
| Avg. context length | 238.9 | 237.3 | 238.0 |
| Avg. long answer length | 43.2 | 45.9 | 41.0 |

Table 1: PubMedQA dataset statistics.

**Collection of PQA-L and PQA-U:** PubMed articles which have i) a question mark in the titles and ii) a structured abstract with conclusive part are collected and denoted as pre-PQA-U. Now each instance has 1) a question which is the original title 2) a context which is the structured abstract without the conclusive part and 3) a long answer which is the conclusive part of the abstract.

Two annotators[3] labeled 1k instances from pre-PQA-U with yes/no/maybe to build PQA-L using Algorithm 1. The annotator 1 doesn't need to do much reasoning to annotate since the long answer is available. We denote this reasoning-free setting. However, the annotator 2 cannot use the long answer, so reasoning over the context is required for annotation. We denote such setting as reasoning-required setting. Note that the annotation process might assign wrong labels when both annotator 1 and annotator 2 make a same mistake, but considering human performance in §5.1, such error rate could be as low as 1%[4]. 500 randomly sampled PQA-L instances are used for 10-fold cross validation and the rest 500 instances consist of PubMedQA test set.

**Algorithm 1** PQA-L data collection procedure

**Input:** pre-PQA-U
ReasoningFreeAnnotation $\leftarrow \{\}$
ReasoningRequiredAnnotation $\leftarrow \{\}$
GroundTruthLabel $\leftarrow \{\}$
**while** not finished **do**
  Randomly sample an instance $inst$ from pre-PQA-U
  **if** $inst$ is not yes/no/maybe answerable **then**
    Remove $inst$ and continue to next iteration
  **end if**
  Annotator 1 annotates $inst$ with $l_1 \in \{\text{yes, no, maybe}\}$ using question, context and long answer
  Annotator 2 annotates $inst$ with $l_2 \in \{\text{yes, no, maybe}\}$ using question and context
  **if** $l_1 = l_2$ **then**
    $l_a \leftarrow l_1$
  **else**
    Annotator 1 and Annotator 2 discuss for an agreement annotation $l_a$
    **if not** $\exists l_a$ **then**
      Remove $inst$ and continue to next iteration
    **end if**
  **end if**
  ReasoningFreeAnnotation$[inst] \leftarrow l_1$
  ReasoningRequiredAnnotation$[inst] \leftarrow l_2$
  GroundTruthLabel$[inst] \leftarrow l_a$
**end while**

Further, we include the unlabeled instances in pre-PQA-U with yes/no/maybe answerable questions to build PQA-U. For this, we use a simple rule-based method which removes all questions started with interrogative words (i.e. wh-words) or involving selections from multiple entities. This results in over 93% agreement with annotator 1 in identifying the questions that can be answered by yes/no/maybe.

**Collection of PQA-A:** Motivated by the recent successes of large-scale pre-training from ELMo (Peters et al., 2018) and BERT (Devlin et al., 2018), we use a simple heuristic to collect many noisily-labeled instances to build PQA-A for pre-training. Towards this end, we use PubMed articles with 1) a statement title which has POS tagging structures of NP-(VBP/VBZ)[5] and 2) a structured abstract including a conclusive part. The

[3] Both are qualified M.D. candidates.

[4] Roughly half of the products of two annotator error rates.

[5] Using Stanford CoreNLP parser (Manning et al., 2014).

| Original Statement Title | Converted Question | Label | % |
|---|---|---|---|
| Spontaneous electrocardiogram alterations ***predict*** ventricular fibrillation in Brugada syndrome. | ***Do*** spontaneous electrocardiogram alterations ***predict*** ventricular fibrillation in Brugada syndrome? | ***yes*** | 92.8 |
| Liver grafts from selected older donors ***do not have*** significantly more ischaemia reperfusion injury. | ***Do*** liver grafts from selected older donors ***have*** significantly more ischaemia reperfusion injury? | ***no*** | 7.2 |

Table 2: Examples of automatically generated instances for PQA-A. Original statement titles are converted to questions and answers are automatically generated according to the negation status.

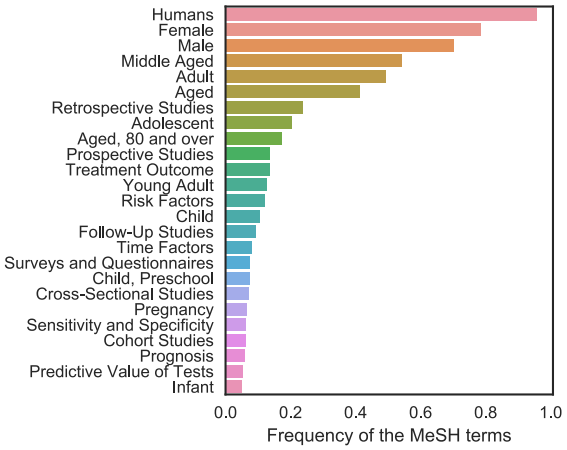


Figure 3: MeSH topic distribution of PubMedQA.

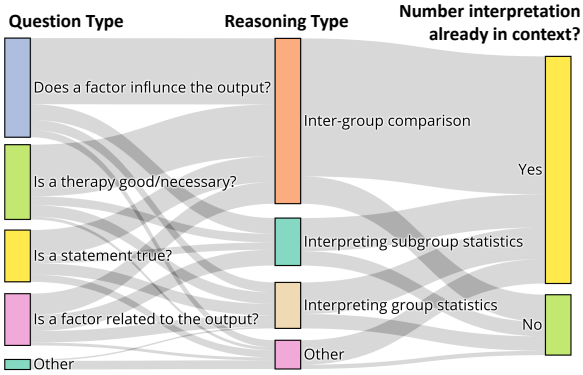


Figure 4: Proportional relationships between corresponded question types, reasoning types, and whether the text interpretations of numbers exist in contexts.

statement titles are converted to questions by simply moving or adding copulas ("is", "are") or auxiliary verbs ("does", "do") in the front and further revising for coherence (e.g.: adding a question mark). We generate the yes/no answer according to negation status of the VB. Several examples are shown in Table 2. We collected 211.3k instances for PQA-A, of which 200k randomly sampled instances are for training and the rest 11.3k instances are for validation.

### 3.2 Characteristics

We show the basic statistics of three PubMedQA subsets in Table 1.

**Instance Topics:** PubMed abstracts are manually annotated by medical librarians with Medical Subject Headings (MeSH)[6], which is a controlled vocabulary designed to describe the topics of biomedical texts. We use MeSH terms to represent abstract topics, and visualize their distribution in Fig. 3. Nearly all instances are human studies and they cover a wide variety of topics, including retrospective, prospective, and cohort studies, different age groups, and healthcare-related subjects like treatment outcome, prognosis and risk factors of diseases.

[6] https://www.nlm.nih.gov/mesh

**Question and Reasoning Types:** We sampled 200 examples from PQA-L and analyzed the types of questions and types of reasoning required to answer them, which is summarized in Table 3. Various types of questions have been asked, including causal effects, evaluations of therapies, relatedness, and whether a statement is true. Besides, PubMedQA also covers several different reasoning types: most (57.5%) involve comparing multiple groups (e.g.: experiment and control), and others require interpreting statistics of a single group or its subgroups. Reasoning over quantitative contents is required in nearly all (96.5%) of them, which is expected due to the nature of biomedical research. 75.5% of contexts have text descriptions of the statistics while 21.0% only have the numbers. We use a Sankey diagram to show the proportional relationships between corresponded question type and reasoning type, as well as corresponded reasoning type and whether there are text interpretations of numbers in Fig. 4.

### 3.3 Evaluation Settings

The main metrics of PubMedQA are accuracy and macro-F1 on PQA-L test set using question and context as input. We denote prediction using question and context as a **reasoning-required** setting, because under this setting answers are not directly

| Question Type | % | Example Questions |
|---|---|---|
| Does a factor ***influence*** the output? | 36.5 | Does reducing spasticity ***translate into*** functional benefit?<br>Does ibuprofen ***increase*** perioperative blood loss during hip arthroplasty? |
| Is a therapy ***good/necessary***? | 26.0 | ***Should*** circumcision be performed in childhood?<br>Is external palliative radiotherapy for gallbladder carcinoma ***effective***? |
| Is a ***statement*** true? | 18.0 | Sternal fracture in growing children: ***A rare and often overlooked fracture?***<br>Xanthogranulomatous cholecystitis: ***a premalignant condition***? |
| Is a factor ***related to*** the output? | 18.0 | Can PRISM ***predict*** length of PICU stay?<br>Is trabecular bone ***related to*** primary stability of miniscrews? |
| **Reasoning Type** | **%** | **Example Snippet in Context** |
| ***Inter-group*** comparison | 57.5 | [...] Postoperative AF was significantly lower in the ***Statin group*** compared with the ***Non-statin group*** (16% versus 33%, p=0.005). [...] |
| Interpreting ***subgroup*** statistics | 16.5 | [...] 57% of patients were ***of lower socioeconomic status*** and they had more health problems, less functioning, and more symptoms [...] |
| Interpreting ***(single) group*** statistics | 16.0 | [...] ***A total of 4 children*** aged 5-14 years with a sternal fracture were treated in 2 years, 2 children were hospitalized for pain management and [...] |
| **Text Interpretations of Numbers** | **%** | **Example Snippet in Context** |
| Existing ***interpretations*** of ***numbers*** | 75.5 | [...] Postoperative AF was ***significantly lower*** in the Statin group compared with the Non-statin group (***16% versus 33%, p=0.005***). [...] |
| No interpretations (***numbers only***) | 21.0 | [...] 30-day mortality was ***12.4%*** in those aged<70 years and ***22%*** in those>70 years (***p<0.001***). [...] |
| No numbers (***texts only***) | 3.5 | [...] The halofantrine therapeutic dose group showed ***loss and distortion of inner hair cells and inner phalangeal cells*** [...] |

Table 3: Summary of PubMedQA question types, reasoning types and whether there are text descriptions of the statistics in context. Colored texts are matched key phrases (sentences) between types and examples.

expressed in the input and reasoning over the contexts is required to answer the question. Additionally, long answers are available at training time, so generation or prediction of them can be used as an auxiliary task in this setting.

A parallel setting, where models can use question and long answer to predict yes/no/maybe answer, is denoted as **reasoning-free** setting since yes/no/maybe are usually explicitly expressed in the long answers (i.e.: conclusions of the abstracts). Obviously, it's a much easier setting which can be exploited for bootstrapping PQA-U.

# 4 Methods

## 4.1 Fine-tuning BioBERT

We fine-tune BioBERT (Lee et al., 2019) on PubMedQA as a baseline. BioBERT is initialized with BERT (Devlin et al., 2018) and further pre-trained on PubMed abstracts and PMC[7] articles. Expectedly, it vastly outperforms BERT in various biomedical NLP tasks. We denote the original transformer weights of BioBERT as $\theta_0$.

While fine-tuning, we feed PubMedQA questions and contexts (or long answers), separated by the special `[SEP]` token, to BioBERT. The yes/no/maybe labels are predicted using the special `[CLS]` embedding using a softmax function. Cross-entropy loss of predicted and true label distribution is denoted as $\mathcal{L}_{\text{QA}}$.

## 4.2 Long Answer as Additional Supervision

Under reasoning-required setting, long answers are available in training but not inference phase. We use them as an additional signal for training: similar to Ma et al. (2018) regularizing neural machine translation models with binary bag-of-word (BoW) statistics, we fine-tune BioBERT with an auxiliary task of predicting the binary BoW statistics of the long answers, also using the special `[CLS]` embedding. We minimize binary cross-entropy loss of this auxiliary task:

$$\mathcal{L}_{\text{BoW}} = -\frac{1}{N}\sum_i b_i \log\hat{b}_i + (1-b_i)\log(1-\hat{b}_i)$$

where $b_i$ and $\hat{b}_i$ are ground-truth and predicted probability of whether token $i$ is in the long answers (i.e.: $b_i \in \{0,1\}$ and $\hat{b}_i \in [0,1]$), and $N$ is the BoW vocabulary size. The total loss is:

$$\mathcal{L} = \mathcal{L}_{\text{QA}} + \beta\mathcal{L}_{\text{BoW}}$$

[7] https://www.ncbi.nlm.nih.gov/pmc/

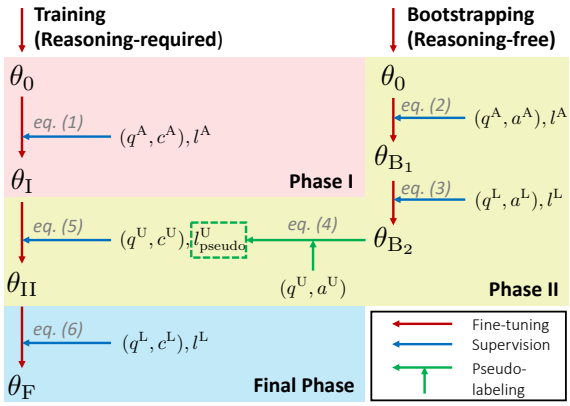


Figure 5: Multi-phase fine-tuning architecture. Notations and equations are described in §4.3.

In reasoning-free setting which we use for bootstrapping, the regularization coefficient $\beta$ is set to 0 because long answers are directly used as input.

### 4.3 Multi-phase Fine-tuning Schedule

Since PQA-A and PQA-U have different properties from the ultimate test set of PQA-L, BioBERT is fine-tuned in a multi-phase style on different subsets. Fig. 5 shows the architecture of this training schedule. We use $q$, $c$, $a$, $l$ to denote question, context, long answer and yes/no/maybe label of instances, respectively. Their source subsets are indexed by the superscripts of A for PQA-A, U for PQA-U and L for PQA-L.

**Phase I Fine-tuning on PQA-A:** PQA-A is automatically collected whose questions and labels are artificially generated. As a result, questions of PQA-A might differ a lot from those of PQA-U and PQA-L, and it only has yes/no labels with a very imbalanced distribution (92.8% yes v.s. 7.2% no). Despite these drawbacks, PQA-A has substantial training instances so models could still benefit from it as a pre-training step.

Thus, in Phase I of multi-phase fine-tuning, we initialize BioBERT with $\theta_0$, and fine-tune it on PQA-A using question and context as input:

$$\theta_{\mathrm{I}} \leftarrow \mathrm{argmin}_\theta \, \mathcal{L}(\mathrm{BioBERT}_\theta(q^{\mathrm{A}}, c^{\mathrm{A}}), l^{\mathrm{A}}) \quad (1)$$

**Phase II Fine-tuning on Bootstrapped PQA-U:** To fully utilize the unlabeled instances in PQA-U, we exploit the easiness of reasoning-free setting to pseudo-label these instances with a bootstrapping strategy: first, we initialize BioBERT with $\theta_0$, and fine-tune it on PQA-A using question and long answer (reasoning-free),

$$\theta_{\mathrm{B}_1} \leftarrow \mathrm{argmin}_\theta \, \mathcal{L}(\mathrm{BioBERT}_\theta(q^{\mathrm{A}}, a^{\mathrm{A}}), l^{\mathrm{A}}) \quad (2)$$

then we further fine-tune $\mathrm{BioBERT}_{\theta_{\mathrm{B}_1}}$ on PQA-L, also under the reasoning-free setting:

$$\theta_{\mathrm{B}_2} \leftarrow \mathrm{argmin}_\theta \, \mathcal{L}(\mathrm{BioBERT}_\theta(q^{\mathrm{L}}, a^{\mathrm{L}}), l^{\mathrm{L}}) \quad (3)$$

We pseudo-label PQA-U instances using the most confident predictions of $\mathrm{BioBERT}_{\theta_{\mathrm{B}_2}}$ for each class. Confidence is simply defined by the corresponding softmax probability and then we label a subset which has the same proportions of yes/no/maybe labels as those in the PQA-L:

$$l^{\mathrm{U}}_{\mathrm{pseudo}} \leftarrow \mathrm{BioBERT}_{\theta_{\mathrm{B}_2}}(q^{\mathrm{U}}, a^{\mathrm{U}}) \quad (4)$$

In phase II, we fine-tune $\mathrm{BioBERT}_{\theta_{\mathrm{I}}}$ on the bootstrapped PQA-U using question and context (under reasoning-required setting):

$$\theta_{\mathrm{II}} \leftarrow \mathrm{argmin}_\theta \, \mathcal{L}(\mathrm{BioBERT}_\theta(q^{\mathrm{U}}, c^{\mathrm{U}}), l^{\mathrm{U}}_{\mathrm{pseudo}}) \quad (5)$$

**Final Phase Fine-tuning on PQA-L:** In the final phase, we fine-tune $\mathrm{BioBERT}_{\theta_{\mathrm{II}}}$ on PQA-L:

$$\theta_{\mathrm{F}} \leftarrow \mathrm{argmin}_\theta \, \mathcal{L}(\mathrm{BioBERT}_\theta(q^{\mathrm{L}}, c^{\mathrm{L}}), l^{\mathrm{L}}) \quad (6)$$

Final predictions on instances of PQA-L validation and test sets are made using $\mathrm{BioBERT}_{\theta_{\mathrm{F}}}$:

$$l_{\mathrm{pred}} = \mathrm{BioBERT}_{\theta_{\mathrm{F}}}(q^{\mathrm{L}}, c^{\mathrm{L}})$$

### 4.4 Compared Models

**Majority:** The majority (about 55%) of the instances have the label "yes". We use a trivial baseline denoted as Majority where we simply predict "yes" for all instances, regardless of the question and context.

**Shallow Features:** For each instance, we include the following shallow features: 1) TF-IDF statistics of the question 2) TF-IDF statistics of the context/long answer and 3) sum of IDF of the overlapping non-stop words between the question and the context/long answer. To allow multi-phase fine-tuning, we apply a feed-forward neural network on the shallow features instead of using a logistic classifier.

**BiLSTM:** We simply concatenate the question and context/long answer with learnable segment embeddings appended to the biomedical word2vec embeddings (Pyysalo et al., 2013) of each token. The concatenated sentence is then fed to a biLSTM, and the final hidden states of the forward and backward network are used for classifying the yes/no/maybe label.

**ESIM with BioELMo:** Following the state-of-the-art recurrent architecture of NLI (Peters et al., 2018), we use pre-trained biomedical contextualized embeddings BioELMo (Jin et al., 2019) for word representations. Then we apply the ESIM model (Chen et al., 2016), where a biLSTM is used to encode the question and context/long answer, followed by an attentional local inference layer and a biLSTM inference composition layer. After pooling, a softmax output unit is applied for predicting the yes/no/maybe label.

### 4.5 Compared Training Schedules

**Final Phase Only:** Under this setting, we train models only on PQA-L. It's an extremely low resources setting where there are only 450 training instances in each fold of cross-validation.

**Phase I + Final Phase:** Under this setting, we skip the training on bootstrapped PQA-U. Models are first fine-tuned on PQA-A, and then fine-tuned on PQA-L.

**Phase II + Final Phase:** Under this setting, we skip the training on PQA-A. Models are first fine-tuned on bootstrapped PQA-U, and then fine-tuned on PQA-L.

**Single-phase Training:** Instead of training a model sequentially on different splits, under single-phase training setting we train the model on the combined training set of all PQA splits: PQA-A, bootstrapped PQA-U and PQA-L.

## 5 Experiments

### 5.1 Human Performance

Human performance is measured during the annotation: As shown in Algorithm 1, annotations of annotator 1 and annotator 2 are used to calculate reasoning-free and reasoning-required human performance, respectively, against the discussed ground truth labels. Human performance on the test set of PQA-L is shown in Table 4. We only test single-annotator performance due to limited resources. Kwiatkowski et al. (2019) show that an ensemble of annotators perform significantly better than single-annotator, so the results reported in Table 4 are the lower bounds of human performance. Under reasoning-free setting where the annotator can see the conclusions, a single human achieves 90.4% accuracy and 84.2% macro-F1. Under reasoning-required setting, the task becomes much harder, but it's still possible for humans to solve: a single annotator can get 78.0% accuracy and 72.2% macro-F1.

| Setting | Accuracy (%) | Macro-F1 (%) |
|---|---|---|
| Reasoning-Free | 90.40 | 84.18 |
| Reasoning-Required | 78.00 | 72.19 |

Table 4: Human performance (single-annotator).

### 5.2 Main Results

We report the test set performance of different models and training schedules in Table 5. In general, multi-phase fine-tuning of BioBERT with additional supervision outperforms other baselines by large margins, but the results are still much worse than just single-human performance.

**Comparison of Models:** A trend of BioBERT $>$ ESIM w/ BioELMo $>$ BiLSTM $>$ shallow features $>$ majority, conserves across different training schedules on both accuracy and macro-F1. Fine-tuned BioBERT is better than state-of-the-art recurrent model of ESIM w/ BioELMo, probably because BioELMo weights are fixed while all BioBERT parameters can be fine-tuned, which better benefit from the pre-training settings.

**Comparison of Training Schedules:** Multi-phase fine-tuning setting gets 5 out of 9 model-wise best accuracy/macro-F1. Due to lack of annotated data, training only on the PQA-L (final phase only) generates similar results as the majority baseline. In phase I + Final setting where models are pre-trained on PQA-A, we observe significant improvements on accuracy and macro-F1 and some models even achieve their best accuracy under this setting. This indicates that a hard task with limited training instances can be at least partially solved by pre-training on a large automatically collected dataset when the tasks are similarly formatted.

Improvements are also observed in phase II + Final setting, though less significant than those of phase I + Final. As expected, multi-phase fine-tuning schedule is better than single-phase, due to different properties of the subsets.

**Additional Supervision:** Despite its simplicity, the auxiliary task of long answer BoW prediction clearly improves the performance: most results (28/40) are better with such additional supervision than without.

| Model | Final Phase Only | | Single-phase | | Phase I + Final | | Phase II + Final | | Multi-phase | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Acc | F1 | Acc | F1 | Acc | F1 | Acc | F1 | Acc | F1 |
| Majority | 55.20 | 23.71 | – | – | – | – | – | – | – | – |
| Human (single) | 78.00 | 72.19 | – | – | – | – | – | – | – | – |
| w/o A.S. | | | | | | | | | | |
| Shallow Features | 53.88 | 36.12 | 57.58 | 31.47 | 57.48 | 37.24 | 56.28 | 40.88 | 53.50 | 39.33 |
| BiLSTM | 55.16 | 23.97 | 55.46 | 39.70 | 58.44 | 40.67 | 52.98 | 33.84 | 59.82 | 41.86 |
| ESIM w/ BioELMo | 53.90 | 32.40 | 61.28 | 42.99 | 61.96 | 43.32 | 60.34 | 44.38 | 62.08 | 45.75 |
| BioBERT | 56.98 | 28.50 | 66.44 | 47.25 | 66.90 | 46.16 | 66.08 | 50.84 | 67.66 | 52.41 |
| w/ A.S. | | | | | | | | | | |
| Shallow Features | 53.60 | 35.92 | 57.30 | 30.45 | 55.82 | 35.09 | 56.46$^\dagger$ | 40.76 | 55.06$^\dagger$ | 40.67$^\dagger$ |
| BiLSTM | 55.22$^\dagger$ | 23.86 | 55.96$^\dagger$ | 40.26$^\dagger$ | 61.06$^\dagger$ | 41.18$^\dagger$ | 54.12$^\dagger$ | 34.11$^\dagger$ | 58.86 | 41.06 |
| ESIM w/ BioELMo | 53.96$^\dagger$ | 31.07 | 62.68$^\dagger$ | 43.59$^\dagger$ | 63.72$^\dagger$ | 47.04$^\dagger$ | 60.16 | 45.81$^\dagger$ | 63.72$^\dagger$ | 47.90$^\dagger$ |
| BioBERT | 57.28$^\dagger$ | 28.70$^\dagger$ | 66.66$^\dagger$ | 46.70$^\dagger$ | 67.24$^\dagger$ | 46.21$^\dagger$ | 66.44$^\dagger$ | 51.41$^\dagger$ | **68.08**$^\dagger$ | **52.72**$^\dagger$ |

Table 5: Main results on PQA-L test set under reasoning-required setting. A.S.: additional supervision. $^\dagger$with A.S. is better than without A.S. Underlined numbers are model-wise best performance, and bolded numbers are global best performance. All numbers are percentages.

| Model | w/o A.S. | | w/ A.S. | |
|---|---|---|---|---|
| | Acc | F1 | Acc | F1 |
| Majority | 92.76 | 48.12 | – | – |
| Shallow Features | 93.01 | 54.59 | 93.05 | 55.12 |
| BiLSTM | 94.59 | 73.40 | 94.45 | 71.81 |
| ESIM w/ BioELMo | 94.82 | 74.01 | 95.04 | 75.22 |
| BioBERT | 96.50 | 84.65 | 96.40 | 83.76 |

Table 6: Results of Phase I (eq. 1). Experiments are on PQA-A under reasoning-required setting. A.S.: additional supervision.

| Model | Eq. 2 | | Eq. 3 | |
|---|---|---|---|---|
| | Acc | F1 | Acc | F1 |
| Majority | 92.76 | 48.12 | 55.20 | 23.71 |
| Human (single) | – | – | 90.40$^\dagger$ | 84.18$^\dagger$ |
| Shallow Features | 93.11 | 56.11 | 54.44 | 38.63 |
| BiLSTM | 95.97 | 83.70 | 71.46 | 50.93 |
| ESIM w/ BioELMo | 97.01 | 88.47 | 74.06 | 58.53 |
| BioBERT | 98.28 | 93.17 | 80.80 | 63.50 |

Table 7: Bootstrapping results. Experiments are on PQA-A (eq. 2) and PQA-L (eq. 3) under reasoning-free setting. $^\dagger$Reasoning-free human performance.

### 5.3 Intermediate Results

In this section we show the intermediate results of multi-phase fine-tuning schedule.

**Phase I:** Results are shown in Table 6. Phase I is fine-tuning on PQA-A using question and context. Since PQA-A is imbalanced due to its collection process, a trivial majority baseline gets 92.76% accuracy. Other models have better accuracy and especially macro-F1 than majority baseline. Fine-tuned BioBERT performs best.

| Model | w/o A.S. | | w/ A.S. | |
|---|---|---|---|---|
| | Acc | F1 | Acc | F1 |
| Majority | 55.10 | 23.68 | – | – |
| Shallow Features | 76.66 | 66.12 | 77.71 | 67.97 |
| Majority | 56.53 | 24.07 | – | – |
| BiLSTM | 85.33 | 81.32 | 85.68 | 81.87 |
| Majority | 55.10 | 23.68 | – | – |
| ESIM w/ BioELMo | 78.47 | 63.32 | 79.62 | 64.91 |
| Majority | 54.82 | 24.87 | – | – |
| BioBERT | 80.93 | 68.84 | 81.02 | 70.04 |

Table 8: Phase II results (eq. 5). Experiments are on pseudo-labeled PQA-U under reasoning-required setting. A.S.: additional supervision.

**Bootstrapping:** Results are shown in Table 7. Bootstrapping is a three-step process: fine-tuning on PQA-A, then on PQA-L and pseudo-labeling PQA-U. All three steps are using question and long answer as input. Expectedly, models perform better in this reasoning-free setting than they do in reasoning-required setting (for PQA-A, Eq. 2 results in Table 7 are better than the performance in Table 6; for PQA-L, Eq. 3 results in Table 7 are better than the performance in Table 5).

**Phase II:** Results are shown in Table 8. In Phase II, since each model is fine-tuned on its own pseudo-labeled PQA-U instances, results are not comparable between models. While the ablation study in Table 5 clearly shows that Phase II is helpful, performance in Phase II doesn't necessarily correlate with final performance on PQA-L.

## 6 Conclusion

We present PubMedQA, a novel dataset aimed at biomedical research question answering using yes/no/maybe, where complex quantitative reasoning is required to solve the task. PubMedQA has substantial automatically collected instances as well as the largest size of expert annotated yes/no/maybe questions in biomedical domain. We provide a strong baseline using multi-phase fine-tuning of BioBERT with long answer as additional supervision, but it's still much worse than just single human performance.

There are several interesting future directions to explore on PubMedQA, e.g.: (1) about 21% of PubMedQA contexts contain no natural language descriptions of numbers, so how to properly handle these numbers is worth studying; (2) we use binary BoW statistics prediction as a simple demonstration for additional supervision of long answers. Learning a harder but more informative auxiliary task of long answer generation might lead to further improvements.

Articles of PubMedQA are biased towards clinical study-related topics (described in Appendix B), so PubMedQA has the potential to assist evidence-based medicine, which seeks to make clinical decisions based on evidence of high quality clinical studies. Generally, PubMedQA can serve as a benchmark for testing scientific reasoning abilities of machine reading comprehension models.

## 7 Acknowledgement

We are grateful for the anonymous reviewers of EMNLP who gave us very valuable comments and suggestions.

## References

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2016. Enhanced lstm for natural language inference. *arXiv preprint arXiv:1609.06038*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

William Hersh, Aaron Cohen, Lynn Ruslen, and Phoebe Roberts. 2007. Trec 2007 genomics track overview. In *TREC 2007*.

William Hersh, Aaron M. Cohen, Phoebe Roberts, and Hari Krishna Rekapalli. 2006. Trec 2006 genomics track overview. In *TREC 2006*.

Qiao Jin, Bhuwan Dhingra, William W Cohen, and Xinghua Lu. 2019. Probing biomedical embeddings from language models. *arXiv preprint arXiv:1904.02181*.

Seongsoon Kim, Donghyeon Park, Yonghwa Choi, Kyubum Lee, Byounggun Kim, Minji Jeon, Jihye Kim, Aik Choon Tan, and Jaewoo Kang. 2018. A pilot study of biomedical text comprehension using an attention-based deep neural reader: Design and experimental analysis. *JMIR medical informatics*, 6(1):e2.

Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gáabor Melis, and Edward Grefenstette. 2018. The narrativeqa reading comprehension challenge. *Transactions of the Association of Computational Linguistics*, 6:317–328.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Rhinehart, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, et al. 2019. Natural questions: a benchmark for question answering research.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. Biobert: pre-trained biomedical language representation model for biomedical text mining. *arXiv preprint arXiv:1901.08746*.

Shuming Ma, Xu Sun, Yizhong Wang, and Junyang Lin. 2018. Bag-of-words as target for neural machine translation. *arXiv preprint arXiv:1805.04871*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Roser Morante, Martin Krallinger, Alfonso Valencia, and Walter Daelemans. 2012. Machine reading of biomedical texts about alzheimers disease. In *CLEF*

*2012 Conference and Labs of the Evaluation Forum-Question Answering For Machine Reading Evaluation (QA4MRE), Rome/Forner, J.[edit.]; ea*, pages 1–14.

Anusri Pampari, Preethi Raghavan, Jennifer Liang, and Jian Peng. 2018. emrqa: A large corpus for question answering on electronic medical records. *arXiv preprint arXiv:1809.00732*.

Dimitris Pappas, Ion Androutsopoulos, and Haris Papageorgiou. 2018. Bioread: A new dataset for biomedical reading comprehension. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*.

Anselmo Peñas, Eduard Hovy, Pamela Forner, Álvaro Rodrigo, Richard Sutcliffe, and Roser Morante. 2013. Qa4mre 2011-2013: Overview of question answering for machine reading evaluation. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 303–320. Springer.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Sampo Pyysalo, Filip Ginter, Hans Moen, Tapio Salakoski, and Sophia Ananiadou. 2013. Distributional semantics resources for biomedical text processing.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Marzieh Saeidi, Max Bartolo, Patrick Lewis, Sameer Singh, Tim Rocktäschel, Mike Sheldon, Guillaume Bouchard, and Sebastian Riedel. 2018. Interpretation of natural language rules in conversational machine reading. *arXiv preprint arXiv:1809.01494*.

Hiroaki Sakamoto, Yasunori Watanabe, and Masataka Satou. 2011. Do preoperative statins reduce atrial fibrillation after coronary artery bypass grafting? *Annals of thoracic and cardiovascular surgery*, 17(4):376–382.

George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, Ioannis Partalas, Matthias Zschunke, Michael R Alvers, Dirk Weissenborn, Anastasia Krithara, Sergios Petridis, Dimitris Polychronopoulos, et al. 2015. An overview of the bioasq large-scale biomedical semantic indexing and question answering competition. *BMC bioinformatics*, 16(1):138.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

## A Yes/no/maybe Answerability

Not all naturally occuring question titles from PubMed are answerable by yes/no/maybe. The first step of annotating PQA-L (as shown in algorithm 1) from pre-PQA-U is to manually identify questions that can be answered using yes/no/maybe. We labeled 1091 (about 50.2%) of 2173 question titles as unanswerable. For example, those questions cannot be answered by yes/no/maybe:

- "Critical Overview of HER2 Assessement in Bladder Cancer: What Is Missing for a Better Therapeutic Approach?" (wh- question)
- "Otolaryngology externships and the match: Productive or futile?" (multiple choices)

## B Over-represented Topics

Clinical study-related topics are over-represented in PubMedQA: we found proportions of MeSH terms like:

- "Pregnancy Outcome"
- "Socioeconomic Factors"
- "Risk Assessment"
- "Survival Analysis"
- "Prospective Studies"
- "Case-Control Studies"
- "Reference Values"

are significantly higher in the PubMedQA articles than those in 200k most recent general PubMed articles (significance is defined by $p < 0.05$ in two-proportion z-test).

## C Annotation Criteria

Strictly speaking, most yes/no/maybe research questions can be answered by "maybe" since there will always be some conditions where one statement is true and vice versa. However, the task will be trivial in this case. Instead, we annotate a question using "yes" if the experiments and results in the paper indicate it, so the answer is not universal but context-dependent.

Given a question like "Do patients benefit from drug X?": certainly not all patients will benefit from it, but if there is a significant difference in an outcome between the experimental and control group, the answer will be "yes". If there is not, the answer will be "no".

"Maybe" is annotated when (1) the paper discusses conditions where the answer is True and conditions where the answer is False or (2) more than one intervention/observation/etc. is asked, and the answer is True for some but False for the others (e.g.: "Do Disease A, Disease B and/or Disease C benefit from drug X?"). To model uncertainty of the answer, we don't strictly follow the logic calculations where such questions can always be answered by either "yes" or "no".

# Quick and (not so) Dirty: Unsupervised Selection of Justification Sentences for Multi-hop Question Answering

**Vikas Yadav, Steven Bethard, Mihai Surdeanu**
University of Arizona, Tucson, AZ, USA
{vikasy, bethard, msurdeanu}@email.arizona.edu

## Abstract

We propose an unsupervised strategy for the selection of justification sentences for multi-hop question answering (QA) that (a) maximizes the relevance of the selected sentences, (b) minimizes the overlap between the selected facts, and (c) maximizes the coverage of both question and answer. This unsupervised sentence selection method can be coupled with any supervised QA approach. We show that the sentences selected by our method improve the performance of a state-of-the-art supervised QA model on two multi-hop QA datasets: AI2's Reasoning Challenge (ARC) and Multi-Sentence Reading Comprehension (MultiRC). We obtain new state-of-the-art performance on both datasets among approaches that do not use external resources for training the QA system: 56.82% F1 on ARC (41.24% on Challenge and 64.49% on Easy) and 26.1% EM0 on MultiRC. Our justification sentences have higher quality than the justifications selected by a strong information retrieval baseline, e.g., by 5.4% F1 in MultiRC. We also show that our unsupervised selection of justification sentences is more stable across domains than a state-of-the-art supervised sentence selection method.

## 1 Introduction

Interpretable machine learning (ML) models, where the end user can understand how a decision was reached, are a critical requirement for the wide adoption of ML solutions in many fields such as healthcare, finance, and law (Samek et al., 2017; Alvarez-Melis and Jaakkola, 2017; Arras et al., 2017; Gilpin et al., 2018; Biran and Cotton, 2017)

For complex natural language processing (NLP) such as question answering (QA), human readable explanations of the inference process have been proposed as a way to interpret QA models (Zhou et al., 2018).

To which organ system do the esophagus, liver, pancreas, small intestine, and colon belong?
(A) reproductive system (B) excretory system
(C) **digestive system** (D) endocrine system

ROCC-selected justification sentences:
1. vertebrate **digestive system** has oral cavity, teeth and pharynx, *esophagus* and stomach, *small intestine, pancreas, liver* and the large intestine
2. **digestive system** consists liver, stomach, large intestine, *small intestine, colon*, rectum and anus

BM25-selected justification sentences:
1. their **digestive system** consists of a stomach, *liver*, *pancreas*, *small intestine*, and a large intestine
2. the *liver pancreas* and gallbladder are the solid organ of the **digestive system**

Figure 1: A multiple-choice question from the ARC dataset with the correct answer in bold, followed by justification sentences selected by our approach (ROCC) vs. sentences selected by a strong IR baseline (BM25). ROCC justification sentences fully cover the five key terms in the question (shown in italic), whereas BM25 misses two: *esophagus* and *colon*. Further, the second BM25 sentence is largely redundant with the first, not covering other query terms.

Recently, multiple datasets have been proposed for *multi-hop* QA, in which questions can only be answered when considering information from multiple sentences and/or documents (Clark et al., 2018; Khashabi et al., 2018a; Yang et al., 2018; Welbl et al., 2018; Mihaylov et al., 2018; Bauer et al., 2018; Dunn et al., 2017; Dhingra et al., 2017; Lai et al., 2017; Rajpurkar et al., 2018; Sun et al., 2019). The task of selecting justification sentences is complex for multi-hop QA, because of the additional knowledge aggregation requirement (examples of such questions and answers are shown in Figures 1 and 2). Although various neural QA methods have achieved high performance on some of these datasets (Sun et al., 2018; Trivedi et al., 2019; Tymoshenko et al., 2017; Seo et al., 2016; Wang and Jiang, 2016; De Cao et al., 2018; Back et al., 2018), we argue that more effort must be dedicated to explaining their inference process.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2578–2589,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

In this work we propose an *unsupervised* algorithm for the selection of *multi-hop* justifications from *unstructured* knowledge bases (KB). Unlike other supervised selection methods (Dehghani et al., 2019; Bao et al., 2016; Lin et al., 2018; Wang et al., 2018b,a; Tran and Niedereée, 2018; Trivedi et al., 2019), our approach does not require any training data for justification selection. Unlike approaches that rely on structured KBs, which are expensive to create, (Khashabi et al., 2016; Khot et al., 2017; Zhang et al., 2018; Khashabi et al., 2018b; Cui et al., 2017; Bao et al., 2016), our method operates over KBs of only unstructured texts. We demonstrate that our approach has a bigger impact on downstream QA approaches that use these justification sentences as additional signal than a strong baseline that relies on information retrieval (IR). In particular, the contributions of this work are:

**(1)** We propose an unsupervised, non-parametric strategy for the selection of justification sentences for multi-hop question answering (QA) that (a) maximizes the ***R***elevance of the selected sentences; (b) minimizes the lexical ***O***verlap between the selected facts; and (c) maximizes the lexical ***C***overage of both question and answer. We call our approach ROCC. ROCC operates by first creating $\binom{n}{k}$ justification sets from the top $n$ sentences selected by the BM25 information retrieval model (Robertson et al., 2009), where $k$ ranges from 2 to $n$, and then ranking them all by a formula that combines the three criteria above. The set with the top score becomes the set of justifications output by ROCC for a given question and candidate answer. As shown in Figure 1, the justification sentences selected by ROCC perform more meaningful knowledge aggregation than a strong IR baseline (BM25), which does not account for overlap (or complementarity) and coverage.

**(2)** ROCC can be coupled with any supervised QA approach that can use the selected justification sentences as additional signal. To demonstrate its effectiveness, we combine ROCC with a state-of-the-art QA method that relies on BERT (Devlin et al., 2018) to classify correct answers, using the text of the question, the answer, and (now) the justification sentences as input. On the Multi-Sentence Reading Comprehension (MultiRC) dataset (Khashabi et al., 2018a), we achieved a gain of 8.3% EM0 with ROCC justifications when compared to the case where the complete comprehension passage was provided to the BERT classifier. On AI2's Reasoning Challenge (ARC) dataset (Clark et al., 2018), the QA approach enhanced with ROCC justifications outperforms the QA method without justifications by 9.15% accuracy, and the approach that uses top sentences provided by BM25 by 2.88%. Further, we show that the justification sentences selected by ROCC are considerably more correct on their own than justifications selected by BM25 (e.g., the justification score in MultiRC was increased by 11.58% when compared to the best performing BM25 justifications), which indicates that the interpretability of the overall QA system was also increased.

**(3)** Lastly, our analysis indicates that ROCC is more stable across the different domains in the MultiRC dataset than a supervised strategy for the selection of justification sentences that relies on a dedicated BERT-based classifier, with a difference of over 10% F1 score in some configurations.

The ROCC system and the codes for generating all the analysis are provided here - `https://github.com/vikas95/AutoROCC`.

## 2 Related Work

The body of QA work that addresses the selection of justification sentences can be classified into roughly four categories: (a) supervised approaches that require training data to learn how to select justification sentences (i.e., questions and answers coupled with correct justifications); (b) methods that treat justifications as latent variables and learn jointly how to answer questions and how to select justifications from questions and answers alone; (c) approaches that rely on information retrieval to select justification sentences; and, lastly, (d) methods that do not use justification sentences at all.

In the first category, previous works (e.g., (Trivedi et al., 2019)) have used entailment resources including labeled trained datasets such as SNLI (Bowman et al., 2015) and MultiNLI (Williams et al., 2017) to train components for selecting justification sentences for QA. Other works have explicitly focused on training sentence selection components for QA models (Min et al., 2018; Lin et al., 2018; Wang et al., 2019). In datasets where gold justification sentences are not provided, researchers have trained such components by retrieving justifications from structured KBs (Cui et al., 2017; Bao et al., 2016; Zhang et al., 2016; Hao et al., 2017) such as ConceptNet (Speer et al., 2017), or from IR systems coupled

with denoising components (Wang et al., 2019). While these works offer exciting directions, they all rely on training data for justifications, which is expensive to generate and may not be available in real-world use cases.

The second group of methods tend to rely on reinforcement learning (Choi et al., 2017; Lai et al., 2018; Geva and Berant, 2018) or PageRank (Surdeanu et al., 2008) to learn how to select justification sentences without explicit training data. Other works have used end-to-end (mostly RNNs with attention mechanisms) QA architectures for learning to pay more attention on better justification sentences (Min et al., 2018; Seo et al., 2016; Yu et al., 2014; Gravina et al., 2018). While these approaches do not require annotated justifications, they need large amounts of question/answer pairs during training so they can discover the latent justifications. In contrast to these two directions, our approach requires no training data at all for the justification selection process.

The third category of methods utilize IR techniques to retrieve justifications from both unstructured (Yadav et al., 2019) and structured (Khashabi et al., 2016) KBs. Our approach is closer in spirit to this direction, but it is adjusted to account for more intentional knowledge aggregation. As we show in Section 4, this is important for both the quality of the justification sentences and the performance of the downstream QA system.

The last group of QA approaches learn how to classify answers without any justification sentences (Mihaylov et al., 2018; Sun et al., 2018; Devlin et al., 2018). While this has been shown to obtain good performance for answer classification, we do not focus on it in this work because these methods cannot easily explain their inference.

Note that some of the works discussed here transfer knowledge from external datasets into the QA task they address (Chung et al., 2017; Sun et al., 2018; Pan et al., 2019; Min et al., 2017; Qiu et al., 2018; Chen et al., 2017). In this work, we focus solely on the resources provided in the task itself because such compatible external resources may not be available in real-world applications of QA.

## 3 Approach

ROCC, coupled with a QA system, operates in the following steps (illustrated in Figure 2):

**(1) Retrieval of candidate justification sentences**: For datasets that rely on huge supporting KBs (e.g., ARC), we retrieve the top $n$ sentences[1] from this KB using an IR query that concatenates the question and the candidate answer, similar to Clark et al. (2018); Yadav et al. (2019). We implemented this using the BM25 IR model with the default parameters in Lucene[2]. For reading comprehension datasets where the question is associated with a text passage (e.g., MultiRC), all the sentences in this passage become candidates.

**(2) Generation of candidate justification sets**: Since its focus is on knowledge aggregation, ROCC ranks *sets* of justification sentences (see below) rather than individual sentences. In this step we create candidate justification sets by generating $\binom{n}{k}$ groups of sentences from the previous $n$ sentences, using multiple values of $k$.

**(3) Ranking of candidate justification sets**: For every candidate justification set, we calculate its ROCC score (see Section 3.1), which estimates the likelihood that this group of justifications explains the given answer. We then rank the justification sets in descending order of ROCC score, and choose the top set as the group of justifications that is the output of ROCC for the given question and answer. In MultiRC, we rearrange the justification sentences according to their original indexes in the given passage to bring coherence in the selected sequence of sentences.

**(4) Answer classification**: ROCC can be coupled with any supervised QA component for answer classification. In this work, we feed in the question, answer, and justification texts into a state-of-the-art classifier that relies on BERT (see Section 3.2). Because the justification sentences in the reading comprehension use case (e.g., MultiRC) come from the same passage and their sequence is likely to be coherent, we concatenate them into a single passage, and use a single BERT instance for classification. This approach is shown on the left side of the answer classification component in Figure 2. On the other hand, the justification sentences retrieved from an external KB (e.g., ARC) may not form a coherent passage when aggregated. For this reason, in the ARC use case, we classify each justification sentence separately (together with the question and candidate answer), and then average all these scores to produce a single score for the candidate answer (right-hand side of the figure).

[1] In this work we used $n = 20$ as in Yadav et al. (2019)

[2] https://lucene.apache.org

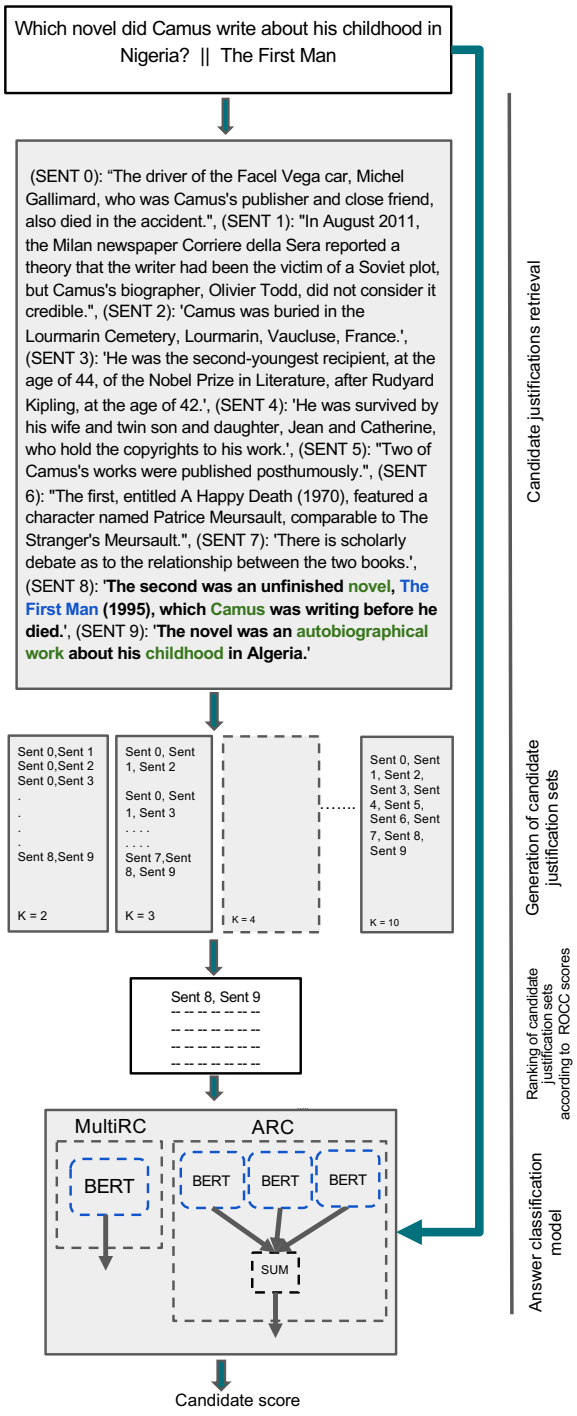


Figure 2: An example of the ROCC process for a question from the MultiRC dataset. Here, ROCC correctly extracts the two justification sentences necessary to explain the correct answer.

### 3.1 Ranking of Candidate Justification Sets

Each set of justifications is ranked based on its ROCC score, which: (a) maximizes the *R*elevance of the selected sentences; (b) minimizes the lexical *O*verlap between the selected facts; and (c) maximizes the lexical *C*overage of both question and answer $(C_{ques}, C_{ans})$. The overall score for a given justification set $P_i$ is calculated as:

$$S(P_i) = \frac{R}{\epsilon + O(P_i)} \cdot (\epsilon + C(A)) \cdot (\epsilon + C(Q)) \quad (1)$$

To avoid zeros, we add a small constant ($\epsilon = 1$ here) to each component that can have a value of 0.[3] We detail the components of this formula below.

**Relevance (R)** We use the Lucene implementation[4] of the BM25 IR model (Robertson et al., 2009) to estimate the relevance of each justification sentence to a given question and candidate answer. In particular, we form a query that concatenates the question and candidate answer, and use as underlying document collection (necessary to compute document statistics such as inverse document frequencies (IDF)) either: sentences in the entire KB (for ARC), or all sentences in the corresponding passage in the case of reading comprehension (MultiRC). The arithmetic mean of BM25 scores over all sentences in a given justification set gives the value of $R$ for the entire set.

**Overlap (O)** To ensure diversity and complementarity between justification sentences, we compute the overlap between all sentence pairs in a given group. Thus, minimizing this score reduces redundancy and encourages the aggregated sentences to address *different* parts of the question and answer:

$$O(S) = \frac{\sum_{s_i \in S} \sum_{s_j \in S - s_i} \frac{|t(s_i) \cap t(s_j)|}{\max(|t(s_i)|, |t(s_j)|)}}{\binom{|S|}{2}} \quad (2)$$

where $S$ is the given set of justification sentences; $s_i$ is the $i^{th}$ sentence in $S$; and $t(s_i)$ denotes the set of unique terms in sentence $s_i$. Note that we divide by $\binom{|S|}{2}$ to normalize across different sizes of justification sets.

**Coverage (C)** Complementing the overlap score, this component measures the lexical coverage of the question and the answer texts by the given set of justifications $S$. This coverage is weighted by the IDF of question and answer terms. Thus, maximizing this value encourages the justifications to address more of the meaningful content mentioned

[3] Our R score relies on BM25, which is larger than 0 on the top $n$ sentences.

[4] https://lucene.apache.org/core/7_0_1/core/org/apache/lucene/search/similarities/BM25Similarity.html

in the question ($X = Q$) and the answer ($X = A$):

$$C_t(X) = \bigcup_{s_i \in S} t(X) \cap t(s_i) \quad (3)$$

$$C(X) = \frac{\sum_{t=1}^{|C_t(X)|} IDF[C_t(X)[t]]}{|t(X)|} \quad (4)$$

where $t(X)$ denotes the unique terms in $X$, and $C_t(X)$ represents the set of all unique terms in $X$ that are present in any of the sentences of the given justification set. $C(X)$ gives the IDF weighted average of $C_t(X)$ terms.

### 3.2 Answer Classification

As indicated earlier, we propose two flavors for the answer classification component: if the sentences in a justification group come from the same passage and, thus, are likely to be coherent, they are concatenated into a single text before classification, and handled by a single answer classifier. If the sentences come from different texts, they are handled by separate instances of the answer classifier. In the latter case, all scores are averaged to produce a single score for a candidate answer. In all situations we used BERT (Devlin et al., 2018) for answer classification. In particular, we employed BERT as a binary classifier operating over two texts. The first text consists of the concatenated question and answer, and the second text consists of the justification text. The classifier operates over the hidden states of the two texts, i.e., the state corresponding to the `[CLS]` token (Devlin et al., 2018).[5]

We observed empirically that pre-training the BERT classifier on *all* $n$ sentences retrieved by BM25, and then fine tuning on the ROCC justifications improves performance on all datasets we experimented with. This resembles the transfer learning discussed by Howard and Ruder (2018), where the source domain would be the BM25 sentences, and the target domain the ROCC justifications. However, one important distinction is that, in our case, all this knowledge comes solely from the resources provided within each dataset, and is retrieved using unsupervised method (BM25). We conjecture that this helped mainly because the pre-training step exposed BERT to more data which, even if imperfect, is topically related to the corresponding question and answer.

[5]We used the following hyper parameters with BERT Large: learning rate of 1e-5, maximum sequence length of 128, batch size = 16, number of epochs = 6.

| Question + answer text | Justification set |
|---|---|
| Animal cells obtain energy by \|\| absorbing nutrients | 1) obtain water and nutrient by absorbing them directly into plant cell<br>2) the animal obtain nourishment by absorbing nutrient released by symbiotic bacteria |

Table 1: Example of a justification set in ARC which was scored by annotator with a precision of $\frac{1}{2}$ because the first justification sentence is not relevant, and a coverage of $\frac{1}{2}$ because the link between *nourishment* and *energy* is not covered.

## 4 Empirical Evaluation

We evaluated ROCC coupled with the proposed QA approach on two QA datasets. We use the standard train/development/test partitions for each dataset, as well as the standard evaluation measures: accuracy for ARC (Clark et al., 2018), and $F1_m$ (macro-F1 score), $F1_a$ (micro-F1 score), and EM0 (exact match) for MultiRC (Khashabi et al., 2018a).

**Multi-sentence reading comprehension (MultiRC):** this is a reading comprehension dataset implemented as multiple-choice QA (Khashabi et al., 2018a). Each question is accompanied by a supporting passage, which contains the correct answer. We use all sentences from such paragraphs as candidate justifications for the corresponding questions.

**AI2's Reasoning Challenge (ARC):** this is a multiple-choice question dataset, containing questions from science exams from grade 3 to grade 9 (Clark et al., 2018). The dataset is split in two partitions: Easy and Challenge, where the latter partition contains the more difficult questions that require reasoning. Most of the questions have 4 answer choices, with <1% of all the questions having either 3 or 5 answer choices. Importantly, ARC includes a supporting KB of 14.3M unstructured text passages. We use BM25 over this entire KB to retrieve candidate justification sentences for ROCC.

### 4.1 Justification Results

To demonstrate that ROCC has the capacity to select better justification sentences, we also report the quality of the extracted justification sentences. For MultiRC, we report precision/recall/F1 justification scores, computed against the gold justification sentences provided by the dataset.[6] For ARC, where gold justifications are not provided, we used an

[6]We use these gold justifications only for evaluation, *not* for training, since ROCC is an unsupervised algorithm.

| # | External resource? | Supervised selection of justifications? | Method | $F1_m$ | $F1_a$ | EM0 | Justification P | Justification R | Justification F1 |
|---|---|---|---|---|---|---|---|---|---|
| | | | **DEVELOPMENT DATASET** | | | | | | |
| | | | Baselines | | | | | | |
| 0 | No | No | Predict 1 (Khashabi et al., 2018a) | 61.0 | 59.9 | 0.8 | – | | |
| 1 | No | No | IR(paragraphs) (Khashabi et al., 2018a) | 64.3 | 60.0 | 1.4 | – | | |
| 2 | No | No | SurfaceLR (Khashabi et al., 2018a) | 66.5 | 63.2 | 11.8 | – | | |
| 3 | No | No | Entailment baseline (Trivedi et al., 2019) | 51.3 | 50.4 | – | – | | |
| | | | Previous work | | | | | | |
| 4 | Yes | Yes | $EER_{DPL}$ + FT Wang et al. (2019) | 70.5 | 67.8 | 13.3 | – | | |
| 5 | Yes | Yes | Multee (GloVe) (Trivedi et al., 2019) | 71.3 | 68.3 | 17.9 | – | | |
| 6 | No | Yes | Multee (ELMo) (Trivedi et al., 2019) | ***70.3*** | 67.3 | ***22.8*** | – | | |
| 7 | Yes | Yes | Multee (ELMo) (Trivedi et al., 2019) | 73.0 | 69.6 | 22.8 | – | | |
| 8 | No | Yes | RS (Sun et al., 2018) | 69.7 | ***67.9*** | 16.9 | – | | |
| 9 | Yes | Yes | RS (Sun et al., 2018) | **73.1**$^\star$ | **70.5**$^\star$ | 21.8 | – | | |
| | | | BERT + IR baselines | | | | | | |
| 10 | No | No | BERT + entire passage | 65.7 | 62.7 | 17.0 | 17.4 | 100.0 | 29.6 |
| 11 | No | No | BERT + BM25 ($k = 1$ sentence) | 66.2 | 62.8 | 17.9 | **61.0** | 27.1 | 37.5 |
| 12 | No | No | BERT + BM25 ($k = 2$ sentences) | 68.1 | 64.8 | 21.0 | 51.6 | 45.6 | 48.4 |
| 13 | No | No | BERT + BM25 ($k = 3$ sentences) | 69.1 | 65.7 | 21.6 | 42.6 | 56.1 | 48.4 |
| 14 | No | No | BERT + BM25 ($k = 4$ sentences) | 70.05 | 66.7 | 22.3 | 36.9 | 64.6 | 47.0 |
| 15 | No | No | BERT + BM25 ($k = 5$ sentences) | 71.2 | 67.7 | 23.4 | 32.7 | 71.1 | 44.8 |
| | | | BERT + parametric ROCC | | | | | | |
| 16 | No | No | BERT + ROCC ($k = 2$ sentences) | 69.8 | 66.8 | 22.7 | 54.7 | 48.5 | 51.4 |
| 17 | No | No | BERT + ROCC ($k = 3$ sentences) | 72.7 | **69.7** | 25.2 | 48.0 | 63.5 | 54.7 |
| 18 | No | No | BERT + ROCC ($k = 4$ sentences) | 72.2 | 69.0 | 25.0 | 40.6 | 71.0 | 51.6 |
| 19 | No | No | BERT + ROCC ($k = 5$ sentences) | 71.6 | 68.7 | 22.7 | 35.0 | **76.5** | 48.1 |
| | | | BERT + non-parametric ROCC | | | | | | |
| 20 | No | No | BERT + AutoROCC ($k \in \{2, 3, 4\}$) | 72.0 | 69.0 | 21.9 | 48.9 | 66.5 | 56.3 |
| 21 | No | No | BERT + AutoROCC ($k \in \{2, 3, 4, 5\}$) | 72.0 | 68.8 | 23.5 | 48.3 | 67.7 | 56.4 |
| 22 | No | No | BERT + AutoROCC ($k \in \{2, 3, 4, 5, 6\}$) | 72.1 | 69.2 | **25.3** | 48.2 | 68.2 | **56.4** |
| 23 | No | No | BERT + BM25 ($k$ from best AutoROCC) | 71.1 | 67.4 | 23.1 | 43.8 | 61.2 | 51.0 |
| 24 | No | No | BERT + AutoROCC ($k \in \{2, 3, 4, 5, 6\}$, pre-trained) | **72.9** | 69.6 | 24.7 | 48.2 | 68.2 | **56.4** |
| | | | Ceiling systems with gold justifications | | | | | | |
| 25 | Yes | Yes | $EER_{gt}$ + FT (Wang et al., 2019) | 72.3 | 70.1 | 19.2 | - - | | |
| 26 | No | Yes | BERT + Gold knowledge | 79.1 | 75.4 | 37.6 | 100.0 | 100.0 | 100.0 |
| 27 | - | - | Human | 86.4 | 83.8 | 56.6 | – | | |
| | | | **TEST DATASET** | | | | | | |
| 28 | No | No | SurfaceLR (Khashabi et al., 2018a) | 66.9 | 63.5 | 12.8 | | | |
| 29 | Yes | Yes | Multee (ELMo) (Trivedi et al., 2019) | 73.8 | 70.4 | 24.5 | – | | |
| 30 | No | No | BERT + AutoROCC ($k \in \{2, 3, 4, 5, 6\}$, pre-trained) | **73.8** | **70.6** | **26.1** | | | |

Table 2: Performance on the MultiRC dataset, under various configurations. $k$ indicates the size(s) of the sets of justification sentences. In parametric ROCC, $k$ is a hyper parameter; in AutoROCC, $k$ is selected automatically. The pre-trained ROCC configurations pre-train BERT on the entire passage corresponding to the question, before fine tuning it on the ROCC sentences. Bold values with $^\star$ indicate state-of-the-art results that used external labeled resources or other supervised methods for the selection of justification sentences. Italicized bold values show state-of-the-art results from experiments that do not use any external labeled resources.

external annotator to annotate the justifications for a random stratified sample of 70 questions, with 10 questions selected from each grade (3 – 9).The annotator reported two scores: precision, and coverage. Precision was defined as the fraction of justification sentences that are relevant for the inference necessary to connect the corresponding question and candidate answer. Coverage was defined as 1 if the justification set completely covers the inference process for the given question and answer, $1/2$ if the set of justifications partially addresses the inference, and 0 if the justification set is completely irrelevant. Table 1 illustrates these scores with an actual output from ARC.

### 4.2 Question answering results

In addition to comparing ROCC with previously reported results, we include multiple baselines: (a) the BERT answer classifier trained on the entire passage of the given question (MultiRC), to demonstrate that ROCC has the capacity to filter out irrelevant content from these paragraphs; (b) BERT trained without any justification sentences (ARC), to show that ROCC has the capacity to aggregate useful information from large unstructured KBs, and (c) BERT trained on sentences retrieved using BM25, to demonstrate that ROCC performs better than other unsupervised approaches. Note that the

| # | External resources used? | Supervised selection of justifications? | Method | Challenge | Easy | All | Justification P, Coverage |
|---|---|---|---|---|---|---|---|
| | | | Baselines | | | | |
| 0 | No | No | AI2 IR Solver (Clark et al., 2018) | 59.99 | 23.98 | | $> 0$ |
| 1 | No | No | Sanity Check (Yadav et al., 2018) | 58.36 | 26.56 | | $> 0$ |
| 2 | Yes | No | Tuple-Inf (Clark et al., 2018) | 60.71 | 23.83 | | $> 0$ |
| 3 | Yes | No | DGEM (Clark et al., 2018) | 58.97 | 27.11 | | $> 0$ |
| | | | Previous work | | | | |
| 4 | Yes | – | Bi-LSTM max-out (Mihaylov et al., 2018) | 33.87 | 34.26 | | $= 0$ |
| 8 | No | No | AHE (Yadav et al., 2019) | 33.28 | ***63.22*** | 53.31 | |
| 9 | No | – | Reading Strategies (Sun et al., 2018) | ***35.40*** | 63.10 | ***53.94*** | $= 0$ |
| 10 | Yes | – | Reading Strategies (Sun et al., 2018) | **42.30**$^\star$ | **68.90**$^\star$ | **60.19**$^\star$ | $= 0$ |
| | | | BERT + IR baselines | | | | |
| 11 | No | – | BERT | 35.11 | 52.75 | 46.94 | |
| 12 | No | No | BERT + BM25 ($k = 1$ sentence) | 33.87 | 56.23 | 48.85 | |
| 13 | No | No | BERT + BM25 ($k = 2$ sentences) | 38.65 | 60.50 | 53.29 | |
| 14 | No | No | BERT + BM25 ($k = 3$ sentences) | 41.04 | 63.19 | 55.89 | |
| 15 | No | No | BERT + BM25 ($k = 4$ sentences) | 37.9 | 63.49 | 53.90 | |
| 16 | No | No | BERT + BM25 ($k = 5$ sentences) | 38.01 | 61.28 | 53.60 | |
| | | | BERT + parametric ROCC | | | | |
| 17 | No | No | BERT + ROCC ($k = 2$ sentences) | 36.65 | 60.59 | 52.69 | |
| 18 | No | No | BERT + ROCC ($k = 3$ sentences) | 39.29 | 62.97 | 55.16 | |
| 19 | No | No | BERT + ROCC ($k = 4$ sentences) | 40.39 | 61.13 | 54.29 | |
| 20 | No | No | BERT + ROCC ($k = 5$ sentences) | 40.62 | 59.96 | 53.58 | |
| | | | BERT + non-parametric ROCC | | | | |
| 21 | No | No | BERT + AutoROCC ($k \in \{2, 3, ...20\}$) | 40.73 | 63.64 | 56.09 | 48.04, 62.50 |
| 22 | No | No | BERT + BM25 ($k$ from best AutoROCC) | 39.24 | 61.01 | 53.83 | 42.55, 55.88 |
| 23 | No | No | BERT + AutoROCC ($k \in \{2, 3, ...20\}$, pre-trained) | **41.24** | **64.49** | **56.82** | **48.04, 62.50** |

Table 3: Performance on the ARC dataset, under various configurations. Notations are the same as in Table 2.

BM25 baseline has an additional hyper parameter: the number of sentences to be considered ($k$).

Table 2 reports comprehensive results on MultiRC, including both overall QA performance, measured using $F1_m$, $F1_a$, and EM0, as well as justification quality, measured using standard precision (P), recall (R), and F1. Note that the bulk of the results are reported on the development partition. The last row in the table reports results on the test partition, computed using the official submission portal which can be accessed only once per model (including its variants). To understand ROCC's behavior, the table includes both the parametric form of ROCC, where the size of the justification sets ($k$) is manually tuned as well as the non-parametric ROCC, where $k$ is automatically selected in the third step of the ROCC algorithm (see Figure 2) by sorting across all sizes of justification sets together, instead of sorting within each value of $k$. Table 3 lists equivalent results on ARC.

We draw several observations from these tables:

**(1)** Despite its simplicity, ROCC combined with the BERT classifier obtains new state-of-the-art performance on both MultiRC and ARC for the class of approaches that do not use external resources to either train the justification sentence selection or the answer classifier. For example, ROCC outperforms the previous best result in MultiRC by 2.5 EM0 points on the development partition (row 24 vs. row 6), and 1.6 EM0 points on test (row 30 vs. row 29). In ARC, ROCC outperforms the previous best approach by 5.8% accuracy on the Challenge partition, and 2.9% overall (row 23 vs. row 9).

**(2)** On both datasets, the non-parametric form of ROCC (AutoROCC) slightly outperforms the parametric variant. Importantly, it always achieves higher justification scores compared to the parametric ROCC. In MultiRC, AutoROCC outperforms our baseline of BERT + entire passage (row 10 vs 22) by 8.3% EM0, indicating that AutoROCC can filter out irrelevant content. In ARC, AutoROCC outperforms the baseline with no justification sentences by 9.1% (row 21 vs row 11), demonstrating that ROCC aggregates useful knowledge.

**(3)** The results of the parametric forms of ROCC (rows 16 – 19 in Table 2 and rows 17 – 20 in Table 3) indicate that performance continues to increase until $k = 4$ in MultiRC and $k = 3$ in ARC. This indicates that: (a) knowledge aggregation is beneficial for these tasks; (b) ROCC can robustly handle non-trivial cases of aggregation with larger values of $k$; and (c) similar to other QA methods (Chen and Durrett, 2019), performance

| train/test | Science textbook | Fiction | News | Wiki articles | wikiMovie Summaries | Society, Law and Justice | All |
|---|---|---|---|---|---|---|---|
| AutoROCC | 54.57 | 53.88 | 54.32 | 60.49 | 57.10 | 61.06 | 56.44 |
| BERT+All passages | 55.15 | 55.46 | 68.77 | 65.14 | 57.39 | 58.79 | 60.90 |
| BERT+Science textbook | 55.67 | 41.01 | 51.45 | 50.06 | 54.96 | 48.84 | 50.79 |
| BERT+Fiction | 45.16 | 57.60 | 63.05 | 63.13 | 59.98 | 50.94 | 58.31 |
| BERT+News | 44.11 | 50.77 | 68.82 | 65.45 | 57.01 | 58.30 | 59.30 |
| GPT-2 (Wang et al., 2019) | - | - | - | - | - | - | 60.7 |

Table 4: Domain robustness of the non-parametric ROCC vs. a supervised sentence selection model, evaluated on the gold justification sentences from MultiRC. Each column represents a section of the MultiRC development set. Each row after AutoROCC represents a justification sentence selection component trained only on the specified section of MultiRC (these sections are listed in descending order of the number of passages in the training data).

decreases for large values of $k$, suggesting that knowledge aggregation remains an open research challenge.

**(4)** The justification scores in both datasets are considerably higher than the equivalent configuration that uses BM25 instead of ROCC (i.e., row 24 vs. row 23 in Table 2, and row 23 vs. row 22 in Table 3). This confirms that the *joint* scoring of sets of justifications that ROCC performs is better than the individual ranking of justification sentences performed by standard IR models such as BM25.

### 4.3 Domain Robustness Analysis

To understand ROCC's domain robustness, we compared it against a supervised BERT-based classifier for the selection of justification sentences, as well as against GPT-2 (Wang et al., 2019). For this experiment, we used MultiRC, where gold justifications are provided. We used this data to train a classifier for the selection of justification sentences on various domain-specific sections of MultiRC. The results of this experiment are shown in Table 4. Unsurprisingly, training and testing in the same domain (e.g., Fiction) leads to the best performance on sentence selection. However, ROCC is more stable across domains than the supervised sentence selection component, with a difference of over 10 F1 points in some configurations. This suggests that ROCC is a better solution for real-world use cases where the distribution of the test data may be very different from the training data.

Compared to BERT, the unsupervised AutoROCC achieves almost the same or better performance in the majority of the domains except Wiki articles and News. We conjecture this happens because the BERT language model was trained on a large text corpus that comes from these two domains. However, importantly, AutoROCC is more robust across domains that are different from these two, since it is an unsupervised approach that is not tuned for any specific domain.

| # | Ablations | ARC | MultiRC EM0 | MultiRC Justification F1 |
|---|---|---|---|---|
| 0 | Full AutoROCC | **56.09** | **25.29** | **56.44** |
| 1 | – IDF | 54.11 | 24.65 | 54.19 |
| 2 | – $C(A)$ | 54.90 | 21.82 | 52.93 |
| 3 | – $C(Q)$ | 54.66 | 23.61 | 52.09 |
| 4 | – O | 55.88 | 24.03 | 55.97 |
| 5 | R$^\star$ | 53.90 | 23.40 | 44.81 |

Table 5: Ablation study, removing different components of ROCC. The scores are reported on the ARC test set and MultiRC dev set. R$^\star$ denotes the best approach that relies just on the $R$ score. The hyper parameter $k$ in R$^\star$, was tuned on the development partition of the respective dataset.

The ARC dataset does not provide justification sentences, so we instead ask how well our question-answering models do on a related inference task, the SciTail entailment dataset (Khot et al., 2018). We trained three QA classifiers on the ARC dataset: BERT with no justification, BERT with BM25 ($k = 4$) justifications, and BERT with AutoROCC justifications. We tested these on SciTail, and achieved 64.49%, 69.70%, and 73.46% accuracy, respectively, indicating that AutoROCC's knowledge aggregation is a valid proxy for entailment.

### 4.4 Ablation Analysis

Table 5 shows an ablation of the different components of ROCC. Row 0 reports the score from the full AutoROCC model. In row 1, we remove IDF weights from coverage calculations (see eq. (4))

| Question type | Precision | Recall | F-1 score |
|---|---|---|---|
| True/False/Yes/No | 54.1 | 68.9 | 60.6 |
| Verbatim | 49.7 | 71.2 | 58.5 |
| Non-verbatim | 47.3 | 68.7 | 56.0 |

Table 6: Justification selection performance of AutoROCC on different types of questions, in the MultiRC development dataset.

of both question and answer text. In row 2, 3 and 4, we remove the coverage of answer, coverage of question, and overlap from the ROCC formula (see eq. (1)) respectively. In all the cases, we found small drops in both performance and justification scores across both the datasets, with the removal of either $C(A)$ or $C(Q)$ having the largest impact.

### 4.5 Error Analysis

We analyzed ROCC's justification selection performance on three different types of questions in MultiRC: True/False/Yes/No, Verbatim, and Non-verbatim (Khashabi et al., 2018b). As shown in Table 6, AutoROCC achieves higher recall scores on Verbatim questions, where the answer text is likely to appear within the given justification passage, and worse recall on question types where such overlap does not exist, e.g., Non-verbatim and True/False. This suggests that the $C(A)$ component of ROCC is important for the extraction of meaningful justifications.

### 4.6 Alignment ROCC

To understand the dependence between ROCC and exact lexical match, we compare the justification selection performance of ROCC when its score components are computed based on lexical match (the approach used throughout the paper up to this point) vs. the semantic alignment match of Yadav et al. (2018). The latter approach relaxes the requirement for lexical match, i.e., two tokens are considered to be matched when the cosine similarity of their embedding vectors is larger than 0.95.[7] As shown in Table 7, the alignment-based ROCC indeed performs better than the ROCC that relies on lexical match. However, the improvements are not large, e.g., the maximum improvement is 1.6% (when $k = 4$), which indicates that ROCC is robust to a certain extent to lexical variation.

[7]This threshold was tuned on the MultiRC development set. We used 100-dimensional GloVe embeddings for this experiment, which performed similarly to larger embedding vectors (300), but allowed for faster experiments.

| ROCC ($k$ sentences) | Lexical ROCC | Align. ROCC |
|---|---|---|
| ROCC ($k = 2$ sentences) | 51.4 | 51.4 |
| ROCC ($k = 3$ sentences) | 54.7 | 55.5 |
| ROCC ($k = 4$ sentences) | 51.6 | 53.2 |
| ROCC ($k = 5$ sentences) | 48.1 | 49.2 |

Table 7: Justification selection performance of the ROCC configuration that uses lexical match (BM25) to retrieve candidate justifications (Lexical ROCC), compared against a ROCC variant that uses the semantic alignment approach of Yadav et al. (2018) to retrieve candidates (Align. ROCC). This experiment used the MultiRC development dataset.

## 5 Conclusion

We introduced ROCC, a simple unsupervised approach for selecting justification sentences for question answering, which balances relevance, overlap of selected sentences, and coverage of the question and answer. We coupled this method with a state-of-the-art BERT-based supervised question answering system, and achieved a new state-of-the-art on the MultiRC and ARC datasets among approaches that do not use external resources during training. We showed that ROCC-based QA approaches are more robust across domains, and generalize better to other related tasks like entailment. In the future, we envision that ROCC scores can be used as distant supervision signal to train supervised justification selection methods.

## Acknowledgments

This work was supported by the Defense Advanced Research Projects Agency (DARPA) under the World Modelers program, grant number W911NF1810014, and by the National Science Foundation (NSF) under grant IIS-1815948.

Mihai Surdeanu declares a financial interest in lum.ai. This interest has been properly disclosed to the University of Arizona Institutional Review Committee and is managed in accordance with its conflict of interest policies.

## References

David Alvarez-Melis and Tommi S Jaakkola. 2017. A causal framework for explaining the predictions of black-box sequence-to-sequence models. *arXiv preprint arXiv:1707.01943*.

Leila Arras, Franziska Horn, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. 2017.

" what is relevant in a text document?": An interpretable machine learning approach. *PloS one*, 12(8):e0181142.

Seohyun Back, Seunghak Yu, Sathish Reddy Indurthi, Jihie Kim, and Jaegul Choo. 2018. Memoreader: Large-scale reading comprehension through neural memory controller. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2131–2140.

Junwei Bao, Nan Duan, Zhao Yan, Ming Zhou, and Tiejun Zhao. 2016. Constraint-based question answering with knowledge graph. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2503–2514.

Lisa Bauer, Yicheng Wang, and Mohit Bansal. 2018. Commonsense for generative multi-hop question answering tasks. *arXiv preprint arXiv:1809.06309*.

Or Biran and Courtenay Cotton. 2017. Explanation and justification in machine learning: A survey. In *IJCAI-17 workshop on explainable AI (XAI)*, volume 8.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*.

Jifan Chen and Greg Durrett. 2019. Understanding dataset design choices for multi-hop reasoning. *arXiv preprint arXiv:1904.12106*.

Eunsol Choi, Daniel Hewlett, Jakob Uszkoreit, Illia Polosukhin, Alexandre Lacoste, and Jonathan Berant. 2017. Coarse-to-fine question answering for long documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 209–220.

Yu-An Chung, Hung-Yi Lee, and James Glass. 2017. Supervised and unsupervised transfer learning for question answering. *arXiv preprint arXiv:1711.05345*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Wanyun Cui, Yanghua Xiao, Haixun Wang, Yangqiu Song, Seung-won Hwang, and Wei Wang. 2017. Kbqa: learning question answering over qa corpora and knowledge bases. *Proceedings of the VLDB Endowment*, 10(5):565–576.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2018. Question answering by reasoning across documents with graph convolutional networks. *arXiv preprint arXiv:1808.09920*.

Mostafa Dehghani, Hosein Azarbonyad, Jaap Kamps, and Maarten de Rijke. 2019. Learning to transform, combine, and reason in open-domain question answering. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 681–689. ACM.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Bhuwan Dhingra, Kathryn Mazaitis, and William W Cohen. 2017. Quasar: Datasets for question answering by search and reading. *arXiv preprint arXiv:1707.03904*.

Matthew Dunn, Levent Sagun, Mike Higgins, V Ugur Guney, Volkan Cirik, and Kyunghyun Cho. 2017. Searchqa: A new q&a dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179*.

Mor Geva and Jonathan Berant. 2018. Learning to search in long documents using document structure. *arXiv preprint arXiv:1806.03529*.

Leilani H Gilpin, David Bau, Ben Z Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. 2018. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 80–89. IEEE.

Alessio Gravina, Federico Rossetto, Silvia Severini, and Giuseppe Attardi. 2018. Cross attention for selection-based question answering. In *2nd Workshop on Natural Language for Artificial Intelligence*. Aachen: R. Piskac.

Yanchao Hao, Yuanzhe Zhang, Kang Liu, Shizhu He, Zhanyi Liu, Hua Wu, and Jun Zhao. 2017. An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 221–231.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018a. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 252–262.

Daniel Khashabi, Tushar Khot, Ashish Sabharwal, Peter Clark, Oren Etzioni, and Dan Roth. 2016. Question answering via integer programming over semi-structured knowledge. *arXiv preprint arXiv:1604.06076*.

Daniel Khashabi, Tushar Khot, Ashish Sabharwal, and Dan Roth. 2018b. Question answering as global reasoning over semantic abstractions. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2017. Answering complex questions using open information extraction. *arXiv preprint arXiv:1704.05572*.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. Scitail: A textual entailment dataset from science question answering. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*.

Tuan Manh Lai, Trung Bui, and Sheng Li. 2018. A review on deep learning techniques applied to answer selection. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2132–2144.

Yankai Lin, Haozhe Ji, Zhiyuan Liu, and Maosong Sun. 2018. Denoising distantly supervised open-domain question answering. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1736–1745.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*.

Sewon Min, Minjoon Seo, and Hannaneh Hajishirzi. 2017. Question answering through transfer learning from large fine-grained supervision data. *arXiv preprint arXiv:1702.02171*.

Sewon Min, Victor Zhong, Richard Socher, and Caiming Xiong. 2018. Efficient and robust question answering from minimal context over documents. *arXiv preprint arXiv:1805.08092*.

Xiaoman Pan, Kai Sun, Dian Yu, Heng Ji, and Dong Yu. 2019. Improving question answering with external knowledge. *arXiv preprint arXiv:1902.00993*.

Minghui Qiu, Liu Yang, Feng Ji, Weipeng Zhao, Wei Zhou, Jun Huang, Haiqing Chen, W Bruce Croft, and Wei Lin. 2018. Transfer learning for context-aware question matching in information-seeking conversations in e-commerce. *arXiv preprint arXiv:1806.05434*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Wojciech Samek, Thomas Wiegand, and Klaus-Robert Müller. 2017. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *arXiv preprint arXiv:1708.08296*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Robert Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *AAAI*, pages 4444–4451.

Kai Sun, Dian Yu, Jianshu Chen, Dong Yu, Yejin Choi, and Claire Cardie. 2019. Dream: A challenge dataset and models for dialogue-based reading comprehension. *arXiv preprint arXiv:1902.00164*.

Kai Sun, Dian Yu, Dong Yu, and Claire Cardie. 2018. Improving machine reading comprehension with general reading strategies. *arXiv preprint arXiv:1810.13441*.

Mihai Surdeanu, Massimiliano Ciaramita, and Hugo Zaragoza. 2008. Learning to rank answers on large online qa collections. In *Proceedings of ACL-08: HLT*, pages 719–727.

Nam Khanh Tran and Claudia Niedereée. 2018. Multihop attention networks for question answer matching. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 325–334. ACM.

Harsh Trivedi, Heeyoung Kwon, Tushar Khot, Ashish Sabharwal, and Niranjan Balasubramanian. 2019. Repurposing entailment for multi-hop question answering tasks. *arXiv preprint arXiv:1904.09380*.

Kateryna Tymoshenko, Daniele Bonadiman, and Alessandro Moschitti. 2017. Ranking kernels for structures and embeddings: A hybrid preference and classification model. In *EMNLP*.

Hai Wang, Dian Yu, Kai Sun, Jianshu Chen, Dong Yu, Dan Roth, and David McAllester. 2019. Evidence sentence extraction for machine reading comprehension. *arXiv preprint arXiv:1902.08852*.

Shuohang Wang and Jing Jiang. 2016. A compare-aggregate model for matching text sequences. *arXiv preprint arXiv:1611.01747*.

Shuohang Wang, Mo YU, Jing JIANG, Wei ZHANG, Xiaoxiao GUO, Shiyu CHANG, Zhiguo WANG, Tim KLINGER, Gerald TESAURO, and Murray CAMPBELL. 2018a. Evidence aggregation for answer re-ranking in open-domain question answering.

Yizhong Wang, Kai Liu, Jing Liu, Wei He, Yajuan Lyu, Hua Wu, Sujian Li, and Haifeng Wang. 2018b. Multi-passage machine reading comprehension with cross-passage answer verification. *arXiv preprint arXiv:1805.02220*.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association of Computational Linguistics*, 6:287–302.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Vikas Yadav, Steven Bethard, and Mihai Surdeanu. 2019. Alignment over heterogeneous embeddings for question answering. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, (Long Papers)*, Minneapolis, USA. Association for Computational Linguistics.

Vikas Yadav, Rebecca Sharp, and Mihai Surdeanu. 2018. Sanity check: A strong alignment and information retrieval baseline for question answering. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1217–1220. ACM.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. *arXiv preprint arXiv:1412.1632*.

Yuanzhe Zhang, Shizhu He, Kang Liu, and Jun Zhao. 2016. A joint model for question answering over multiple knowledge bases. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Yuyu Zhang, Hanjun Dai, Kamil Toraman, and Le Song. 2018. Kg$\hat{2}$: Learning to reason science exam questions with contextual knowledge graph embeddings. *CoRR*, abs/1805.12393.

Mantong Zhou, Minlie Huang, and Xiaoyan Zhu. 2018. An interpretable reasoning network for multi-relation question answering. *arXiv preprint arXiv:1801.04726*.

# Answering Complex Open-domain Questions Through Iterative Query Generation

**Peng Qi**† **Xiaowen Lin***† **Leo Mehr***† **Zijian Wang***‡ **Christopher D. Manning**†
† Computer Science Department ‡ Symbolic Systems Program
Stanford University
{pengqi, veralin, leomehr, zijwang, manning}@cs.stanford.edu

## Abstract

It is challenging for current one-step retrieve-and-read question answering (QA) systems to answer questions like *"Which novel by the author of 'Armada' will be adapted as a feature film by Steven Spielberg?"* because the question seldom contains retrievable clues about the missing entity (here, the author). Answering such a question requires multi-hop reasoning where one must gather information about the missing entity (or facts) to proceed with further reasoning. We present GOLDEN (Gold Entity) Retriever, which iterates between reading context and retrieving more supporting documents to answer open-domain multi-hop questions. Instead of using opaque and computationally expensive neural retrieval models, GOLDEN Retriever generates natural language search queries given the question and available context, and leverages off-the-shelf information retrieval systems to query for missing entities. This allows GOLDEN Retriever to scale up efficiently for open-domain multi-hop reasoning while maintaining interpretability. We evaluate GOLDEN Retriever on the recently proposed open-domain multi-hop QA dataset, HOTPOTQA, and demonstrate that it outperforms the best previously published model despite not using pretrained language models such as BERT.

## 1 Introduction

Open-domain question answering (QA) is an important means for us to make use of knowledge in large text corpora and enables diverse queries without requiring a knowledge schema ahead of time. Enabling such systems to perform multi-step inference can further expand our capability to explore the knowledge in these corpora (*e.g.*, see Figure 1).

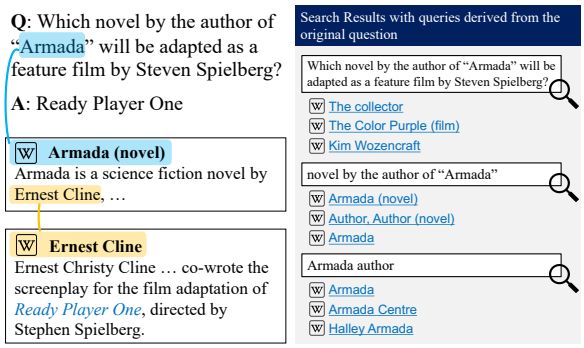


Figure 1: An example of an open-domain multi-hop question from the HOTPOTQA dev set, where "Ernest Cline" is the missing entity. Note from the search results that it cannot be easily retrieved based on merely the question. (Best viewed in color)

Fueled by the recently proposed large-scale QA datasets such as SQuAD (Rajpurkar et al., 2016, 2018) and TriviaQA (Joshi et al., 2017), much progress has been made in open-domain question answering. Chen et al. (2017) proposed a two-stage approach of retrieving relevant content with the question, then reading the paragraphs returned by the information retrieval (IR) component to arrive at the final answer. This "*retrieve and read*" approach has since been adopted and extended in various open-domain QA systems (Nishida et al., 2018; Kratzwald and Feuerriegel, 2018), but it is inherently limited to answering questions that do not require multi-hop/multi-step reasoning. This is because for many multi-hop questions, not all the relevant context can be obtained in a single retrieval step (*e.g.*, "Ernest Cline" in Figure 1).

More recently, the emergence of multi-hop question answering datasets such as QAngaroo (Welbl et al., 2018) and HOTPOTQA (Yang et al., 2018) has sparked interest in multi-hop QA in the research community. Designed to be more challenging than SQuAD-like datasets, they feature questions that require context of more than one

*Equal contribution, order decided by a random number generator.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2590–2602, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

document to answer, testing QA systems' abilities to infer the answer in the presence of multiple pieces of evidence and to efficiently find the evidence in a large pool of candidate documents. However, since these datasets are still relatively new, most of the existing research focuses on the few-document setting where a relatively small set of context documents is given, which is guaranteed to contain the "gold" context documents, all those from which the answer comes (De Cao et al., 2019; Zhong et al., 2019).

In this paper, we present GOLDEN (Gold Entity) Retriever. Rather than relying purely on the original question to retrieve passages, the central innovation is that at each step the model also uses IR results from previous hops of reasoning to generate a new natural language query and retrieve new evidence to answer the original question. For the example in Figure 1, GOLDEN would first generate a query to retrieve *Armada (novel)* based on the question, then query for *Ernest Cline* based on newly gained knowledge in that article. This allows GOLDEN to leverage off-the-shelf, general-purpose IR systems to scale open-domain multi-hop reasoning to millions of documents efficiently, and to do so in an interpretable manner. Combined with a QA module that extends BiDAF++ (Clark and Gardner, 2017), our final system outperforms the best previously published system on the open-domain (fullwiki) setting of HOTPOTQA without using powerful pretrained language models like BERT (Devlin et al., 2019).

The main contributions of this paper are: (a) a novel iterative retrieve-and-read framework capable of multi-hop reasoning in open-domain QA;[1] (b) a natural language query generation approach that guarantees interpretability in the multi-hop evidence gathering process; (c) an efficient training procedure to enable query generation with minimal supervision signal that significantly boosts recall of gold supporting documents in retrieval.

## 2 Related Work

**Open-domain question answering (QA)** Inspired by the series of TREC QA competitions,[2] Chen et al. (2017) were among the first to adapt neural QA models to the open-domain setting. They built a simple inverted index lookup with TF-IDF on the English Wikipedia, and used the question as the query to retrieve top 5 results for a reader model to produce answers with. Recent work on open-domain question answering largely follow this retrieve-and-read approach, and focus on improving the information retrieval component with question answering performance in consideration (Nishida et al., 2018; Kratzwald and Feuerriegel, 2018; Nogueira et al., 2019). However, these one-step retrieve-and-read approaches are fundamentally ill-equipped to address questions that require multi-hop reasoning, especially when necessary evidence is not readily retrievable with the question.

**Multi-hop QA datasets** QAngaroo (Welbl et al., 2018) and HOTPOTQA (Yang et al., 2018) are among the largest-scale multi-hop QA datasets to date. While the former is constructed around a knowledge base and the knowledge schema therein, the latter adopts a free-form question generation process in crowdsourcing and span-based evaluation. Both datasets feature a few-document setting where the gold supporting facts are provided along with a small set of distractors to ease the computational burden. However, researchers have shown that this sometimes results in gameable contexts, and thus does not always test the model's capability of multi-hop reasoning (Chen and Durrett, 2019; Min et al., 2019a). Therefore, in this work, we focus on the fullwiki setting of HOTPOTQA, which features a truly open-domain setting with more diverse questions.

**Multi-hop QA systems** At a broader level, the need for multi-step searches, query task decomposition, and subtask extraction has been clearly recognized in the IR community (Hassan Awadallah et al., 2014; Mehrotra et al., 2016; Mehrotra and Yilmaz, 2017), but multi-hop QA has only recently been studied closely with the release of large-scale datasets. Much research has focused on enabling multi-hop reasoning in question answering models in the few-document setting, *e.g.*, by modeling entity graphs (De Cao et al., 2019) or scoring answer candidates against the context (Zhong et al., 2019). These approaches, however, suffer from scalability issues when the number of supporting documents and/or answer candidates grow beyond a few dozen. Ding et al. (2019) apply entity graph modeling to HOTPOTQA, where

[1] Code and pretrained models available at https://github.com/qipeng/golden-retriever

[2] http://trec.nist.gov/data/qamain.html

they expand a small entity graph starting from the question to arrive at the context for the QA model. However, centered around entity names, this model risks missing purely descriptive clues in the question. Das et al. (2019) propose a neural retriever trained with distant supervision to bias towards paragraphs containing answers to the given questions, which is then used in a multi-step reader-reasoner framework. This does not fundamentally address the discoverability issue in open-domain multi-hop QA, however, because usually not all the evidence can be directly retrieved with the question. Besides, the neural retrieval model lacks explainability, which is crucial in real-world applications. Talmor and Berant (2018), instead, propose to answer multi-hop questions at scale by decomposing the question into sub-questions and perform iterative retrieval and question answering, which shares very similar motivations as our work. However, the questions studied in that work are based on logical forms of a fixed schema, which yields additional supervision for question decomposition but limits the diversity of questions. More recently, Min et al. (2019b) apply a similar idea to HotpotQA, but this approach similarly requires additional annotations for decomposition, and the authors did not apply it to iterative retrieval.

## 3 Model

In this section, we formally define the problem of open-domain multi-hop question answering, and motivate the architecture of the proposed GoldEn (Gold Entity) Retriever model. We then detail the query generation components as well as how to derive supervision signal for them, before concluding with the QA component.

### 3.1 Problem Statement

We define the problem of *open-domain multi-hop QA* as one involving a question $q$, and $S$ relevant (gold) supporting context documents $d_1, \ldots, d_S$ which contain the desired answer $a$.[3] These supporting documents form a chain of reasoning necessary to arrive at the answer, and they come from a large corpus of documents $\mathcal{D}$ where $|\mathcal{D}| \gg S$. In this chain of reasoning, the supporting documents are usually connected via shared entities or textual similarities (*e.g.*, they describe similar entities or events), but these connections do not necessarily conform to any predefined knowledge schema.

We contrast this to what we call the *few-document setting* of multi-hop QA, where the QA system is presented with a small set of documents $\mathcal{D}_{\text{few-doc}} = \{d_1, \ldots, d_S, d'_1, \ldots, d'_D\}$, where $d'_1, \ldots, d'_D$ comprise a small set of "distractor" documents that test whether the system is able to pick out the correct set of supporting documents in the presence of noise. This setting is suitable for testing QA systems' ability to perform multi-hop reasoning given the gold supporting documents with bounded computational budget, but we argue that it is far from a realistic one. In practice, an open-domain QA system has to locate all gold supporting documents from $\mathcal{D}$ on its own, and as shown in Figure 1, this is often difficult for multi-hop questions based on the original question alone, as not all gold documents are easily retrievable given the question.

To address this gold context discoverability issue, we argue that it is necessary to move away from a single-hop retrieve-and-read approach where the original question serves as the search query. In the next section, we introduce GoldEn Retriever, which addresses this problem by iterating between retrieving more documents and reading the context for multiple rounds.

### 3.2 Model Overview

Essentially, the challenge of open-domain multi-hop QA lies in the fact that the information need of the user $(q \rightarrow a)$ cannot be readily satisfied by any information retrieval (IR) system that models merely the similarity between the question $q$ and the documents. This is because the true information need will only unfold with progressive reasoning and discovery of supporting facts. Therefore, one cannot rely solely on a similarity-based IR system for such iterative reasoning, because the potential pool of relevant documents grows exponentially with the number of hops of reasoning.

To this end, we propose GoldEn (Gold Entity) Retriever, which makes use of the gold document[4] information available in the QA dataset at training time to iteratively query for more relevant supporting documents during each hop of reasoning. Instead of relying on the original question as the search query to retrieve all supporting facts,

[3] In this work, we only consider extractive, or span-based, QA tasks, but the problem statement and the proposed method apply to generative QA tasks as well.

[4] In HotpotQA, documents usually describe entities, thus we use "documents" and "entities" interchangeably.

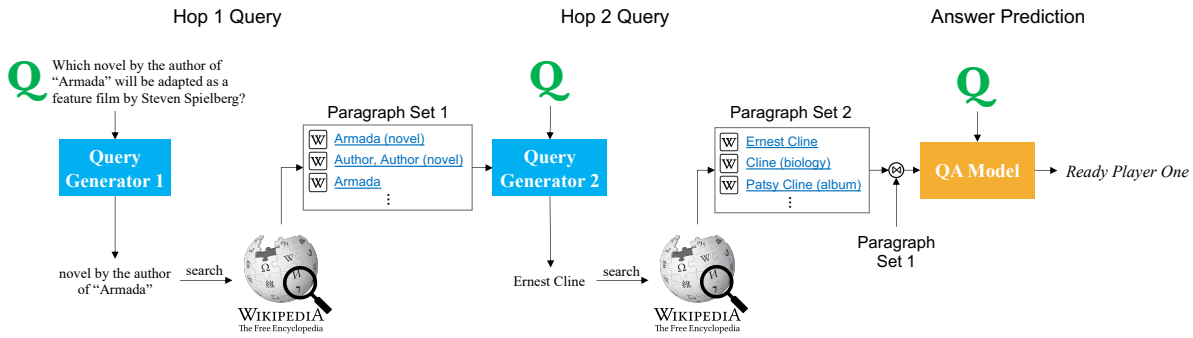


Figure 2: Model overview of GOLDEN Retriever. Given an open-domain multi-hop question, the model iteratively retrieves more context documents, and concatenates all retrieved context for a QA model to answer from.

or building computationally expensive search engines that are less interpretable to humans, we propose to leverage text-based IR engines for interpretability, and generate different search queries as each reasoning step unfolds. In the very first hop of reasoning, GOLDEN Retriever is presented the original question $q$, from which it generates a search query $q_1$ that retrieves supporting document $d_1$.[5] Then for each of the subsequent reasoning steps ($k = 2, \ldots, S$), GOLDEN Retriever generates a query $q_k$ from the question and the available context, $(q, d_1, \ldots, d_{k-1})$. This formulation allows the model to generate queries based on information revealed in the supporting facts (see Figure 2, for example).

We note that GOLDEN Retriever is much more efficient, scalable, and interpretable at retrieving gold documents compared to its neural retrieval counterparts. This is because GOLDEN Retriever does not rely on a QA-specific IR engine tuned to a specific dataset, where adding new documents or question types into the index can be extremely inefficient. Further, GOLDEN Retriever generates queries in natural language, making it friendly to human interpretation and verification. One core challenge in GOLDEN Retriever, however, is to train query generation models in an efficient manner, because the search space for potential queries is enormous and off-the-shelf IR engines are not end-to-end differentiable. We outline our solution to this challenge in the following sections.

### 3.3 Query Generation

For each reasoning step, we need to generate the search query given the original question $q$ and some context of documents we have already retrieved (initially empty). This query generation problem is conceptually similar to the QA task in that they both map a question and some context to a target, only instead of an answer, the target here is a search query that helps retrieve the desired supporting document for the next reasoning step. Therefore, we formulate the query generation process as a question answering task.

To reduce the potentially large space of possible queries, we favor a QA model that extracts text spans from the context over one that generates free-form text as search queries. We therefore employ DrQA's Document Reader model (Chen et al., 2017), which is a relatively light-weight recurrent neural network QA model that has demonstrated success in few-document QA. We adapt it to query generation as follows.

For each reasoning step $k = 1, \ldots, S$, given a question $q$ and some *retrieval context* $C_k$ which *ideally* contains the gold supporting documents $d_1, \ldots, d_{k-1}$, we aim to generate a search query $q_k$ that helps us retrieve $d_k$ for the next reasoning step. A Document Reader model is trained to select a span from $C_k$ as the query

$$q_k = G_k(q, C_k),$$

where $G_k$ is the query generator. This query is then used to search for supporting documents, which are concatenated with the current retrieval context to update it

$$C_{k+1} = C_k \bowtie \text{IR}_n(q_k)$$

where $\text{IR}_n(q_k)$ is the top $n$ documents retrieved from the search engine using $q_k$, and $C_1 = q$.[6] At the end of the retrieval steps, we provide $q$ as question along with $C_S$ as context to the final few-

[5] For notational simplicity, $d_k$ denotes the supporting document needed to complete the $k$-th step of reasoning. We also assume that the goal of each IR query is to retrieve one and only one gold supporting document in its top $n$ results.

[6] In the query result, the title of each document is delimited with special tokens <t> and </t> before concatenation.

| Question | Hop 1 Oracle | Hop 2 Oracle |
|---|---|---|
| What government position was held by the woman who portrayed Corliss Archer in the film Kiss and Tell? | Corliss Archer in the film Kiss and Tell | Shirley Temple |
| Scott Parkin has been a vocal critic of Exxonmobil and another corporation that has operations in how many countries? | Scott Parkin | Halliburton |
| Are Giuseppe Verdi and Ambroise Thomas both Opera composers? | Giuseppe Verdi | Ambroise Thomas |

Table 1: Example oracle queries on the HOTPOTQA dev set.

document QA component detailed in Section 3.5 to obtain the final answer to the original question.

To train the query generators, we follow the steps above to construct the retrieval contexts, but during training time, when $d_k$ is not part of the IR result, we replace the lowest ranking document with $d_k$ before concatenating it with $C_k$ to make sure the downstream models have access to necessary context.

### 3.4 Deriving Supervision Signal for Query Generation

When deriving supervision signal to train our query generators, the potential search space is enormous for each step of reasoning even if we constrain ourselves to predicting spans from the context. This is aggravated by multiple hops of reasoning required by the question. One solution to this issue is to train the query generators with reinforcement learning (RL) techniques (*e.g.*, REINFORCE (Sutton et al., 2000)), where (Nogueira and Cho, 2017) and (Buck et al., 2018) are examples of one-step query generation with RL. However, it is computationally inefficient, and has high variance especially for the second reasoning step and forward, because the context depends greatly on what queries have been chosen previously and their search results.

Instead, we propose to leverage the limited supervision we have about the gold supporting documents $d_1, \ldots, d_S$ to narrow down the search space. The key insight we base our approach on is that at any step of open-domain multi-hop reasoning, there is some *semantic overlap* between the retrieval context and the next document(s) we wish to retrieve. For instance, in our *Armada* example, when the retrieval context contains only the question, this overlap is the novel itself; after we have expanded the retrieval context, this overlap becomes the name of the author, *Ernest Cline*. Finding this semantic overlap between the retrieval context and the desired documents not only reveals the chain of reasoning naturally, but also allows us to use it as the search query for retrieval.

Because off-the-shelf IR systems generally optimize for shallow lexical similarity between query and candidate documents in favor of efficiency, a good proxy for this overlap is locating spans of text that have high lexical overlap with the intended supporting documents. To this end, we propose a simple yet effective solution, employing several heuristics to generate candidate queries: computing the longest common string/sequence between the current retrieval context and the title/text of the intended paragraph ignoring stop words, then taking the contiguous span of text that corresponds to this overlap in the retrieval context. This allows us to not only make use of entity names, but also textual descriptions that better lead to the gold entities. It is also more generally applicable than question decomposition approaches (Talmor and Berant, 2018; Min et al., 2019b), and does not require additional annotation for decomposition.

Applying various heuristics results in a handful of candidate queries for each document, and we use our IR engine (detailed next) to rank them based on recall of the intended supporting document to choose one as the final oracle query we train our query generators to predict. This allows us to train the query generators in a fully supervised manner efficiently. Some examples of oracle queries on the HOTPOTQA dev set can be found in Table 1. We refer the reader to Appendix B for more technical details about our heuristics and how the oracle queries are derived.

**Oracle Query vs Single-hop Query** We evaluate the oracle query against the single-hop query, i.e., querying with the original question, on the HOTPOTQA dev set. Specifically, we compare the recall of gold paragraphs, because the greater the recall, the fewer documents we need to pass into the expensive neural multi-hop QA component.

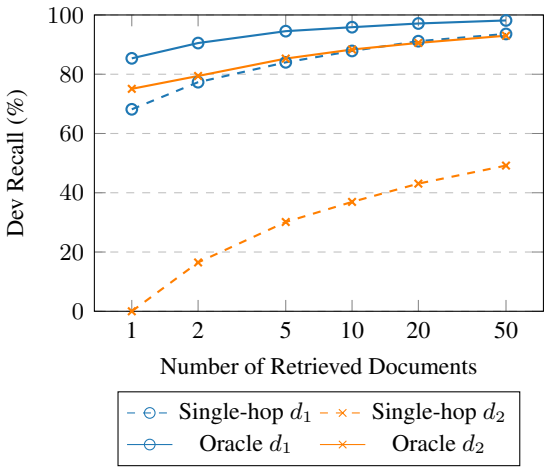


Figure 3: Recall comparison between single-hop queries and GOLDEN Retriever oracle queries for both supporting paragraphs on the HOTPOTQA dev set. Note that the oracle queries are much more effective than the original question (single-hop query) at retrieving target paragraphs in both hops.

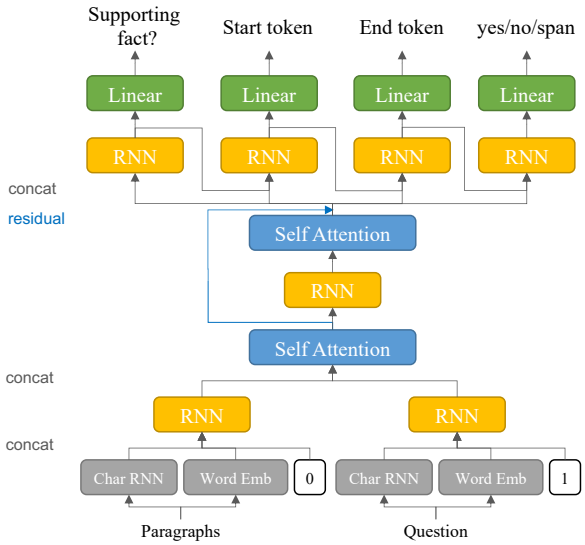


Figure 4: Question answering component in GOLDEN Retriever. (Best viewed in color)

We index the English Wikipedia dump with introductory paragraphs provided by the HOTPOTQA authors[7] with Elasticsearch 6.7 (Gormley and Tong, 2015), where we index the titles and document text in separate fields with bigram indexing enabled. This results in an index with 5,233,329 total documents. At retrieval time, we boost the scores of any search result whose title matches the search query better – this results in a better recall for entities with common names (*e.g.*, "*Armada*" the novel). For more details about how the IR engine is set up and the effect of score boosting, please refer to Appendix A.

In Figure 3, we compare the recall of the two gold paragraphs required for each question in HOTPOTQA at various number of documents retrieved (R@$n$) for the single-hop query and the multi-hop queries generated from the oracle. Note that the oracle queries are much more effective at retrieving the gold paragraphs than the original question in both hops. For instance, if we combine R@5 of both oracles (which effectively retrieves 10 documents from two queries) and compare that to R@10 for the single-hop query, the oracle queries improve recall for $d_1$ by 6.68%, and that for $d_2$ by a significant margin of 49.09%.[8] This means that the final QA model will need to consider far fewer documents to arrive at a decent set of supporting facts that lead to the answer.

### 3.5 Question Answering Component

The final QA component of GOLDEN Retriever is based on the baseline model presented in (Yang et al., 2018), which is in turn based on BiDAF++ (Clark and Gardner, 2017). We make two major changes to this model. Yang et al. (2018) concatenated all context paragraphs into one long string to predict span begin and end offsets for the answer, which is potentially sensitive to the order in which these paragraphs are presented to the model. We instead process them separately with shared encoder RNN parameters to obtain paragraph order-insensitive representations for each paragraph. Span offset scores are predicted from each paragraph independently before finally aggregated and normalized with a global softmax operation to produce probabilities over spans. The second change is that we replace all attention mechanism in the original model with self attention layers over the concatenated question and context. To differentiate context paragraph representations from question representations in this self-attention mechanism, we indicate question and context tokens by concatenating a 0/1 feature at the input layer. Figure 4 illustrates the QA model architecture.

[7] https://hotpotqa.github.io/wiki-readme.html

[8] Since HOTPOTQA does not provide the logical order its gold entities should be discovered, we simply call the document $d_1$ which is more easily retrievable with the queries, and the other as $d_2$.

| System | Answer | | Sup Fact | | Joint | |
|---|---|---|---|---|---|---|
| | EM | $F_1$ | EM | $F_1$ | EM | $F_1$ |
| Baseline (Yang et al., 2018) | 25.23 | 34.40 | 5.07 | 40.69 | 2.63 | 17.85 |
| GRN + BERT | 29.87 | 39.14 | 13.16 | 49.67 | 8.26 | 25.84 |
| MUPPET (Feldman and El-Yaniv, 2019) | 30.61 | 40.26 | 16.65 | 47.33 | 10.85 | 27.01 |
| CogQA (Ding et al., 2019) | 37.12 | 48.87 | 22.82 | 57.69 | 12.42 | 34.92 |
| PR-Bert | 43.33 | 53.79 | 21.90 | 59.63 | 14.50 | 39.11 |
| Entity-centric BERT Pipeline | 41.82 | 53.09 | 26.26 | 57.29 | 17.01 | 39.18 |
| BERT pip. (contemporaneous) | 45.32 | 57.34 | 38.67 | 70.83 | 25.14 | 47.60 |
| GOLDEN Retriever | 37.92 | 48.58 | 30.69 | 64.24 | 18.04 | 39.13 |

Table 2: End-to-end QA performance of baselines and our GOLDEN Retriever model on the HOTPOTQA fullwiki test set. Among systems that were not published at the time of submission of this paper, "BERT pip." was submitted to the official HOTPOTQA leaderboard on May 15th (thus contemporaneous), while "Entity-centric BERT Pipeline" and "PR-Bert" were submitted after the paper submission deadline.

## 4 Experiments

### 4.1 Data and Setup

We evaluate our models in the fullwiki setting of HOTPOTQA (Yang et al., 2018). HOTPOTQA is a question answering dataset collected on the English Wikipedia, containing about 113k crowd-sourced questions that are constructed to require the introduction paragraphs of two Wikipedia articles to answer. Each question in the dataset comes with the two gold paragraphs, as well as a list of sentences in these paragraphs that crowdworkers identify as supporting facts necessary to answer the question. A diverse range of reasoning strategies are featured in HOTPOTQA, including questions involving missing entities in the question (our *Armada* example), intersection questions (*What satisfies property A and property B?*), and comparison questions, where two entities are compared by a common attribute, among others. In the few-document *distractor* setting, the QA models are given ten paragraphs in which the gold paragraphs are guaranteed to be found; in the open-domain *fullwiki* setting, which we focus on, the models are only given the question and the entire Wikipedia. Models are evaluated on their answer accuracy and explainability, where the former is measured as overlap between the predicted and gold answers with exact match (EM) and unigram $F_1$, and the latter concerns how well the predicted supporting fact sentences match human annotation (Supporting Fact EM/$F_1$). A joint metric is also reported on this dataset, which encourages systems to perform well on both tasks simultaneously.

We use the Stanford CoreNLP toolkit (Manning et al., 2014) to preprocess Wikipedia, as well as to generate POS/NER features for the query generators, following (Yang et al., 2018) and (Chen et al., 2017). We always detokenize a generated search query before sending it to Elasticsearch, which has its own preprocessing pipeline. Since all questions in HOTPOTQA require exactly two supporting documents, we fix the number of retrieval steps of GOLDEN Retriever to $S = 2$. To accommodate arbitrary steps of reasoning in GOLDEN Retriever, a stopping criterion is required to determine when to stop retrieving for more supporting documents and perform few-document question answering, which we leave to future work. During training and evaluation, we set the number of retrieved documents added to the retrieval context to 5 for each retrieval step, so that the total number of paragraphs our final QA model considers is 10, for a fair comparison to (Yang et al., 2018).

We include hyperparameters and training details in Appendix C for reproducibility.

### 4.2 End-to-end Question Answering

We compare the end-to-end performance of GOLDEN Retriever against several QA systems on the HOTPOTQA dataset: (1) the baseline presented in (Yang et al., 2018), (2) CogQA (Ding et al., 2019), the top-performing previously published system, and (3) other high-ranking systems on the leaderboard. As shown in Table 2, GOLDEN Retriever is much better at locating the correct supporting facts from Wikipedia compared

| Setting | Ans $F_1$ | Sup $F_1$ | R@10* |
|---|---|---|---|
| GOLDEN Retriever | 49.79 | 64.58 | 75.46 |
| Single-hop query | 38.19 | 54.82 | 62.38 |
| HOTPOTQA IR | 36.34 | 46.78 | 55.71 |

Table 3: Question answering and IR performance amongst different IR settings on the dev set. We observe that although improving the IR engine is helpful, most of the performance gain results from the iterative retrieve-and-read strategy of GOLDEN Retriever. (*: for GOLDEN Retriever, the 10 paragraphs are combined from both hops, 5 from each hop.)

| System | Ans $F_1$ | Sup $F_1$ | Joint $F_1$ |
|---|---|---|---|
| GOLDEN Retriever | 49.79 | 64.58 | 40.21 |
| w/ Hop 1 oracle | 52.53 | 68.06 | 42.68 |
| w/ Hop 1 & 2 oracles | 62.32 | 77.00 | 52.18 |

Table 4: Pipeline ablative analysis of GOLDEN Retriever end-to-end QA performance by replacing each query generator with a query oracle.

to CogQA, as well as most of the top-ranking systems. However, the QA performance is handicapped because we do not make use of pretrained contextualization models (*e.g.*, BERT) that these systems use. We expect a boost in QA performance from adopting these more powerful question answering models, especially ones that are tailored to perform few-document multi-hop reasoning. We leave this to future work.

To understand the contribution of GOLDEN Retriever's iterative retrieval process compared to that of the IR engine, we compare the performance of GOLDEN Retriever against two baseline systems on the dev set: one that retrieves 10 supporting paragraphs from Elasticsearch with the original question, and one that uses the IR engine presented in HOTPOTQA.[9] In all cases, we use the QA component in GOLDEN Retriever for the final question answering step. As shown in Table 3, replacing the hand-engineered IR engine in (Yang et al., 2018) with Elasticsearch does result in some gain in recall of the gold documents, but that does not translate to a significant improvement in QA performance. Further inspection reveals that despite Elasticsearch improving overall recall of gold documents, it is only able to retrieve both gold documents for 36.91% of the dev set questions, in comparison to 28.21% from the IR engine in (Yang et al., 2018). In contrast, GOLDEN Retriever improves this percentage to 61.01%, almost doubling the recall over the single-hop baseline, providing the QA component a much better set of context documents to predict answers from.

Lastly, we perform an ablation study in which we replace our query generator models with our query oracles and observe the effect on end-to-end performance. As can be seen in Table 4, replacing $G_1$ with the oracle only slightly improves end-to-end performance, but further substituting $G_2$ with the oracle yields a significant improvement. This illustrates that the performance loss is largely attributed to $G_2$ rather than $G_1$, because $G_2$ solves a harder span selection problem from a longer retrieval context. In the next section, we examine the query generation models more closely by evaluating their performance without the QA component.

### 4.3 Analysis of Query Generation

To evaluate the query generators, we begin by determining how well they emulate the oracles. We evaluate them using Exact Match (EM) and $F_1$ on the span prediction task, as well as compare their queries' retrieval performance against the oracle queries. As can be seen in Table 6, the performance of $G_2$ is worse than that of $G_1$ in general, confirming our findings on the end-to-end pipeline. When we combine them into a pipeline, the generated queries perform only slightly better on $d_1$ when a total of 10 documents are retrieved (89.91% vs 87.85%), but are significantly more effective for $d_2$ (61.01% vs 36.91%). If we further zoom in on the retrieval performance on non-comparison questions for which finding the two entities involved is less trivial, we can see that the recall on $d_2$ improves from 27.88% to 53.23%, almost doubling the number of questions we have the complete gold context to answer. We note that the IR performance we report on the full pipeline is different to that when we evaluate the query generators separately. We attribute this difference to the fact that the generated queries sometimes retrieve both gold documents in one step.

To better understand model behavior, we also randomly sampled some examples from the dev set to compare the oracle queries and the predicted queries. Aside from exact matches, we find that the predicted queries are usually small variations of the oracle ones. In some cases, the model selects spans that are more natural and informative

[9] For the latter, we use the fullwiki test input file the authors released, which contains the top-10 IR output from that retrieval system with the question as the query.

| | Question | Predicted $q_1$ | Predicted $q_2$ |
|---|---|---|---|
| (1) | What video game character did the voice actress in the animated film Alpha and Omega voice? | voice actress in the animated film Alpha and Omega *(animated film Alpha and Omega voice)* | Hayden Panettiere |
| (2) | What song was created by the group consisting of Jeffrey Jey, Maurizio Lobina and Gabry Ponte and released on 15 January 1999? | Jeffrey Jey *(group consisting of Jeffrey Jey, Maurizio Lobina and Gabry Ponte)* | Gabry Ponte and released on 15 January 1999 *("Blue (Da Ba Dee)")* |
| (3) | Yau Ma Tei North is a district of a city with how many citizens? | Yau Ma Tei North | Yau Tsim Mong District of Hong Kong *(Hong Kong)* |
| (4) | What company started the urban complex development that included the highrise building, The Harmon? | highrise building, The Harmon | CityCenter |

Table 5: Examples of predicted queries from the query generators on the HOTPOTQA dev set. The oracle query is displayed in blue in parentheses if it differs from the predicted one.

| Model | Span | | R@5 |
|---|---|---|---|
| | EM | $F_1$ | |
| $G_1$ | 51.40 | 78.75 | 85.86 |
| $G_2$ | 52.29 | 63.07 | 64.83 |

Table 6: Span prediction and IR performance of the query generator models for Hop 1 ($G_1$) and Hop 2 ($G_2$) evaluated separately on the HOTPOTQA dev set.

(Example (1) in Table 5). When they differ a bit more, the model is usually overly biased towards shorter entity spans and misses out on informative information (Example (2)). When there are multiple entities in the retrieval context, the model sometimes selects the wrong entity, which suggests that a more powerful query generator might be desirable (Example (3)). Despite these issues, we find that these natural language queries make the reasoning process more interpretable, and easier for a human to verify or intervene as needed.

**Limitations** Although we have demonstrated that generating search queries with span selection works in most cases, it also limits the kinds of queries we can generate, and in some cases leads to undesired behavior. One common issue is that the entity of interest has a name shared by too many Wikipedia pages (*e.g.*, "*House Rules*" the 2003 TV series). This sometimes results in the inclusion of extra terms in our oracle query to expand it (*e.g.*, Example (4) specifies that "*The Harmon*" is a highrise building). In other cases, our span oracle makes use of too much information from the gold entities (Example (2), where a human would likely query for "*Eiffel 65 song released 15 January 1999*" because "*Blue*" is not the only song mentioned in $d_1$). We argue, though, these are due to the simplifying choice of span selection for query generation and fixed number of query steps. We leave extensions to future work.

## 5 Conclusion

In this paper, we presented GOLDEN (Gold Entity) Retriever, an open-domain multi-hop question answering system for scalable multi-hop reasoning. Through iterative reasoning and retrieval, GOLDEN Retriever greatly improves the recall of gold supporting facts, thus providing the question answering model a much better set of context documents to produce an answer from, and demonstrates competitive performance to the state of the art. Generating natural languages queries for each step of reasoning, GOLDEN Retriever is also more interpretable to humans compared to previous neural retrieval approaches and affords better understanding and verification of model behavior.

## Acknowledgements

The authors would like to thank Robin Jia among other members of the Stanford NLP Group, as well as the anonymous reviewers for discussions and comments on earlier versions of this paper. Peng Qi would also like to thank Suprita Shankar, Jamil Dhanani, and Suma Kasa for early experiments on BiDAF++ variants for HOTPOTQA. This research is funded in part by Samsung Electronics Co., Ltd. and in part by the SAIL-JD Research Initiative.

## References

Christian Buck, Jannis Bulian, Massimiliano Ciaramita, Wojciech Gajewski, Andrea Gesmundo, Neil Houlsby, and Wei Wang. 2018. Ask the right questions: Active question reformulation with reinforce-

ment learning. In *Proceedings of the International Conference on Learning Representations*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Association for Computational Linguistics (ACL)*.

Jifan Chen and Greg Durrett. 2019. Understanding dataset design choices for multi-hop reasoning. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Christopher Clark and Matt Gardner. 2017. Simple and effective multi-paragraph reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association of Computational Linguistics*.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, and Andrew McCallum. 2019. Multi-step retriever-reader interaction for scalable open-domain question answering. In *International Conference on Learning Representations*.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2019. Question answering by reasoning across documents with graph convolutional networks. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Ming Ding, Chang Zhou, Qibin Chen, Hongxia Yang, and Jie Tang. 2019. Cognitive graph for multi-hop reading comprehension at scale. In *Proceedings of the 57th Annual Meeting of the Association of Computational Linguistics*.

Yair Feldman and Ran El-Yaniv. 2019. Multi-hop paragraph retrieval for open-domain question answering. In *Proceedings of the 57th Annual Meeting of the Association of Computational Linguistics*.

Clinton Gormley and Zachary Tong. 2015. *Elasticsearch: The definitive guide: A distributed real-time search and analytics engine*. O'Reilly Media, Inc.

Ahmed Hassan Awadallah, Ryen W White, Patrick Pantel, Susan T Dumais, and Yi-Min Wang. 2014. Supporting complex search tasks. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 829–838. ACM.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Bernhard Kratzwald and Stefan Feuerriegel. 2018. Adaptive document retrieval for deep question answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Rishabh Mehrotra, Prasanta Bhattacharya, and Emine Yilmaz. 2016. Deconstructing complex search tasks: A bayesian nonparametric approach for extracting sub-tasks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 599–605.

Rishabh Mehrotra and Emine Yilmaz. 2017. Extracting hierarchies of search tasks & subtasks via a bayesian nonparametric approach. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 285–294. ACM.

Sewon Min, Eric Wallace, Sameer Singh, Matt Gardner, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2019a. Compositional questions do not necessitate multi-hop reasoning. In *Proceedings of the Annual Conference of the Association of Computational Linguistics*.

Sewon Min, Victor Zhong, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2019b. Multi-hop reading comprehension through question decomposition and rescoring. In *Proceedings of the Annual Conference of the Association of Computational Linguistics*.

Kyosuke Nishida, Itsumi Saito, Atsushi Otsuka, Hisako Asano, and Junji Tomita. 2018. Retrieve-and-read: Multi-task learning of information retrieval and reading comprehension. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 647–656. ACM.

Rodrigo Nogueira and Kyunghyun Cho. 2017. Task-oriented query reformulation with reinforcement learning.

Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019. Document expansion by query prediction. *arXiv preprint arXiv:1904.08375*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Stephen E. Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. 1995. Okapi at trec-3. *Nist Special Publication Sp*, 109:109.

Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, pages 1057–1063.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association of Computational Linguistics*, 6:287–302.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Victor Zhong, Caiming Xiong, Nitish Shirish Keskar, and Richard Socher. 2019. Coarse-grain fine-grain coattention network for multi-evidence question answering. In *Proceedings of the International Conference on Learning Representations*.

# A Elasticsearch Setup

## A.1 Setting Up the Index

We start from the Wikipedia dump file containing the introductory paragraphs used in HOTPOTQA that Yang et al. (2018) provide,[10] and add the fields corresponding to Wikipedia page titles and the introductory paragraphs (text) into the index.

For the title, we use Elasticsearch's `simple` analyzer which performs basic tokenization and lowercasing of the content. For the text, we join all the sentences and use the `standard` analyzer which further allows for removal of punctuation and stop words. For both fields, we index an auxiliary field with bigrams using the `shingle` filter,[11] and perform basic `asciifolding` to map non ASCII characters to a similar ASCII character (*e.g.*, "é"→ "e").

At search time, we launch a `multi_match` query against all fields with the same query, which performs a full-text query employing the BM25 ranking function (Robertson et al., 1995) with all fields in the index, and returns the score of the best field for ranking by default. To promote documents whose title match the search query, we boost the search score of all title-related fields by 1.25 in this query.

## A.2 Reranking Search Results

In Wikipedia, it is common that pages or entity names share rare words that are important to search engines, and a naive full-text search IR system will not be able to pick the one that matches the query the best. For instance, if one set up Elasticsearch according to the instructions above and searched for "*George W. Bush*", he/she would be surprised to see that the actual page is not even in the top-10 search results, which contains entities such as "*George W. Bush Childhood Home*" and "*Bibliography of George W. Bush*".

To this end, we propose to rerank these query results with a simple but effective heuristic that alleviates this issue. We would first retrieve at least 50 candidate documents for each query for consideration, and boost the query scores of documents whose title exactly matches the search query, or is a substring of the search query. Specifically, we multiply the document score by a heuristic constant between 1.05 and 1.5, depending on how well the document title matches the query, before reranking all search results. This results in a significant improvement in these cases. For the query "*George W. Bush*", the page for the former US president is ranked at the top after reranking. In Table 7, we also provide results from the single-hop query to show the improvement from title score boosting introduced from the previous section and reranking.

| IR System | R@10 for $d_1$ | R@10 for $d_2$ |
|---|---|---|
| Final system | 87.85 | 36.91 |
| w/o Title Boosting | 86.85 | 32.64 |
| w/o Reranking | 86.32 | 34.77 |
| w/o Both | 84.67 | 29.55 |

Table 7: IR performance (recall in percentages) of various Elasticsearch setups on the HOTPOTQA dev set using the original question.

# B Oracle Query Generation

We mainly employ three heuristics to find the semantic overlap between the retrieval context and the desired documents: longest common subsequence (LCS), longest common substring (LCSubStr), and overlap merging which generalizes the two. Specifically, the overlap merging heuristic looks for contiguous spans in the retrieval context that have high rates of overlapping tokens with the desired document, determined by the total number of overlapping tokens divided by the total number of tokens considered in the span.

In all heuristics, we ignore stop words and lowercase the rest in computing the spans to capture more meaningful overlaps, and finally take the span in the retrieval context that all the overlapping words are contained in. For instance, if the retrieval context contains "*the* GOLDEN *Retriever model on* HOTPOTQA" and the desired document contains "GOLDEN *Retriever on the* HOTPOTQA *dataset*", we will identify the overlapping terms as "GOLDEN", "*Retriever*", and "HOTPOTQA", and return the span "GOLDEN *Retriever model on* HOTPOTQA" as the resulting candidate query.

To generate candidates for the oracle query, we apply the heuristics between combinations of {cleaned question, cleaned question without punctuation} × {cleaned document title, cleaned paragraph}, where cleaning means stop word removal and lowercasing. Once oracle queries are gener-

[10] https://hotpotqa.github.io/wiki-readme.html

[11] https://www.elastic.co/guide/en/elasticsearch/reference/6.7/analysis-shingle-tokenfilter.html

| Hyperparameter | Values |
|---|---|
| Learning rate | $\underline{5 \times 10^{-4}}$, $\mathbf{1 \times 10^{-3}}$ |
| Finetune embeddings | 0, 200, **500**, *1000* |
| Epoch | **25**, *40* |
| Batch size | ***32***, 64, 128 |
| Hidden size | 64, ***128***, 256, 512, 768 |
| Max sequence length | 15, *20*, 30, **50**, 100 |
| Dropout rate | 0.3, 0.35, ***0.4***, 0.45 |

Table 8: Hyperparameter settings for the query generators. The final hyperparameters for the Hop 1 query generator are shown in **bold**, and those for the Hop 2 query generator are shown in *underlined italic*.

ated, we launch these queries against Elasticsearch to determine the rank of the desired paragraph. If multiple candidate queries are able to place the desired paragraph in the top 5 results, we further rank the candidate queries by other metrics (*e.g.*, length of the query) to arrive at the final oracle query to train the query generators. We refer the reader to our released code for further details.

## C Training Details

### C.1 Query Generators

Once the oracle queries are generated, we train our query generators to emulate them on the training set, and choose the best model with $F_1$ in span selection on the dev set. We experiment with hyperparameters such as learning rate, training epochs, batch size, number of word embeddings to finetune, among others, and report the final hyperparameters for both query generators ($G_1$ and $G_2$) in Table 8.

### C.2 Question Answering Model

Our final question answering component is trained with the paragraphs produced by the oracle queries (5 from each hop, 10 in total), with $d_1$ and $d_2$ inserted to replace the lowest ranking paragraph in each hop if they are not in the set already.

We develop our model based on the baseline model of Yang et al. (2018), and reuse the same default hyperparameters whenever possible. The main differences in the hyperparameters are: we optimize our QA model with Adam (with default hyperparameters) (Kingma and Ba, 2015) instead of stochastic gradient descent with a larger batch size of 64; we anneal the learning rate by 0.5 with a patience of 3 instead of 1, that is, we multiply the learning rate by 0.5 after three consecutive failures to improve dev $F_1$; we clip the gradient down to a maximum $\ell_2$ norm of 5; we apply a 10% dropout to the model, for which we have increased the hidden size to 128; and use 10 as the coefficient by which we multiply the supporting facts loss, before mixing it with the span prediction loss. We configure the model to read 10 context paragraphs, and limit each paragraph to at most 400 tokens including the title.

# NL2pSQL: Generating Pseudo-SQL Queries from Under-Specified Natural Language Questions

Fuxiang Chen[1,4], Seung-won Hwang[2], Jaegul Choo[3], Jung-Woo Ha[1] and Sunghun Kim[1,4]

[1]Clova AI Research, NAVER
[2]Yonsei University
[3]Korea University
[4]Hong Kong University of Science and Technology

*fchenaa@cse.ust.hk , seungwonh@yonsei.ac.kr , jchoo@korea.ac.kr , {jungwoo.ha , sung.kim.n}@navercorp.com*

## Abstract

Generating SQL codes from natural language questions (NL2SQL) is an emerging research area. Existing studies have mainly focused on clear scenarios where specified information is fully given to generate a SQL query. However, in developer forums such as Stack Overflow,[1] questions cover more diverse tasks including table manipulation or performance issues, where a table is not specified. The SQL query posted in Stack Overflow, Pseudo-SQL (pSQL), does not usually contain table schemas and is not necessarily executable, is sufficient to guide developers. Here we describe a new NL2pSQL task to generate pSQL codes from natural language questions on under-specified database issues, in short, NL2pSQL. In addition, we define two new metrics suitable for the proposed NL2pSQL task, Canonical-BLEU and SQL-BLEU, instead of the conventional BLEU. With a baseline model using sequence-to-sequence architecture integrated with denoising autoencoder, we confirm the validity of our task. Experiments show that the proposed NL2pSQL approach yields well-formed queries (up to 43% more than a standard Seq2Seq model). Our code and datasets are publicly available at http://github.com/clovaai/nl2psql.

## 1 Introduction

Converting natural-language questions to SQL codes (NL2pSQL) is an active area of research in natural language processing. However, earlier work (Hemphill et al., 1990; Brad et al., 2017; Zelle and Mooney, 1996; Tang and Mooney, 2000; Popescu et al., 2003; Lawrence and Riezler, 2016; Li and Jagadish, 2014; Roy et al., 2013; Yaghmazadeh et al., 2017; Zhong et al., 2017; Yu et al., 2018c; Finegan-Dollak et al., 2018) mainly focused on fully specified questions where the given question and the table can be uniquely answered by a generated SQL query. For example, a question such as "who is the manufacturer for the order year 1998?" can only be uniquely answered when the table that can answer the question is attached to the question. This requires the generated SQL query to be executable, in order to verify its correctness.

Meanwhile, real-life NL2pSQL questions from developer forums such as Stack Overflow (SO) (StackOverflow, 2018) are mostly under-specified to generate an executable answer. Actual questions deal with more diverse issues, including table creation, backup, or performance, and often present with a less detailed context (such as missing table name or schema). For example, a SO question[2] such as "How to reset AUTO_INCREMENT in MySQL?" can only be answered by a non-executable template "ALTER TABLE [tablename] AUTO_INCREMENT = 1", where human developers are expected to read and then feed the missing context. We call these questions *Under-specified NL2pSQL*, distinguished from the conventional NL2pSQL task on fully specified questions.

Our key contribution is to address 1) the novel problem of generating human-readable pseudo SQL code to diverse ranges of under-specified developer questions with our own curated dataset and to propose 2) a new approach as well as 3) evaluation metrics suitable for the proposed task. Our dataset contains crawled posts in SO between the year of 2008 and 2017. The proposed task addresses the following new opportunities and challenges.

**First,** our new dataset covers diverse scenarios with 362 distinct SQL keywords, such as *USING, UNIQUE, SELECT, UPDATE, TRUNCATE, etc*,

[1]https://www.stackoverflow.com

[2]https://stackoverflow.com/questions/8923114



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2603–2613, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

unlike specified scenarios such as data lookup using simple *SELECT* queries. Meanwhile, 19% of the answers in SO do not contain the *SELECT* keyword at all.

**Second,** we eliminate the requirement of a generated code to be executable, for which we propose new evaluation metrics without such a requirement. An existing metric is lexical comparisons with the human-generated templates using BLEU scores (Papineni et al., 2002). This metric based on counting matching $n$-gram may penalize mismatches in non-essential tokens such as variable names (Lin et al., 2018; Zhong et al., 2017), while mismatches of SQL keywords should not be tolerated.

Our contribution includes the following:

- **Dataset:** We publicly release the NL2pSQL dataset containing 1,244 pairs as well as the denoising dataset for NL2pSQL containing 406,384 pairs.[3]

- **NL2pSQL:** We propose an end-to-end natural NL2pSQL model without the restriction of an executable SQL query trained and evaluated for under-specifed questions. Existing sequence-to-sequence models are known to generate repetitive tokens, but our denoising autoencoder can effectively fix errors in SQL statements towards readable codes. Our model performs a two-step translation by cascading two sequence-to-sequence models, one for generating initial SQL statements from natural language input, and the other to fix ill-formed SQL statements.

- **Evaluation metric:** While existing NL2pSQL metrics penalize the mismatch of any terms, we propose novel metrics allowing the mismatches for non-essential terms, yet requiring matches for important SQL keyword: Specifically, Canonical-BLEU abstracts the identifier names of the SQL statements before calculating the BLEU scores while SQL-BLEU calculates the BLEU scores using only the SQL keywords. Different forms of SQL queries may exist for a given natural language, and we note that no single metric is capable to accurately discern one form from the other. Thus, two proposed metrics complement each other.

[3] https://github.com/clovaai/nl2psql

## 2 Related Work

**Natural Language to SQL.** Bridging the gap between natural language and SQL queries has been a long-studied area (Warren and Pereira, 1982; Popescu et al., 2003, 2004; Li et al., 2006; Giordani and Moschitti, 2012). With the emergence of deep learning, neural network-based approaches have gained popularity including models using sequence-to-sequence and reinforcement learning (Zhong et al., 2017; Xu et al., 2018; Iyer et al., 2017; Dong and Lapata, 2016; Yu et al., 2018a). These models assume fully-specified questions and require human-annotated executable SQL queries for training and evaluation. Thus, they are not applicable to under-specified questions, which are the majority of issues in real-world software developer forums.

An alternative approach is designing semantic parsers, to exploit rich prior knowledge in the form of features and grammars (Zettlemoyer and Collins, 2005). It aims at mapping natural language utterances to an executable logical form such as SQL. They require rich resources – in the form of either grammars or training data to extract such grammars. As existing semantic parsers on SQL build on pairs of SQL and their execution results, which cannot exist in our problem context (Xu et al., 2018; Zhong et al., 2017; Sun et al., 2018; Yu et al., 2018c,b) of under-specified questions, we adopt a generative approach in our model.

**Program Synthesis** Program synthesis or automatic code generation is a relatively new field and has gained much attention lately. In previous work (Ling et al., 2016; Rabinovich et al., 2017; Yin and Neubig, 2017; Balog et al., 2017; Murali et al., 2018), the generated code is targeted at programming languages such as Java and Python, or they used a reduced specification such as a domain-specific language (DSL). Meanwhile, Beltramelli utilized GUI screenshots to generate markup language codes (Beltramelli, 2017). Our work differs in the sense that we are not constrained to a reduced specification but deal with a new domain of SQL statements, instead of markup languages or fully-fledged programming languages such as Java and Python.

**Denoising and Program Repair** Denoising is a popular research area in computer vision (Buades et al., 2005; Elad and Aharon, 2006; Dabov et al., 2007; Zhang et al., 2017; Meiniel et al., 2018) and

natural language processing (Kouno et al., 2015; Napoles et al., 2017). In software engineering, denoising tasks focus on bug fixing in fully-fledged programming languages such as Java and C (Long and Rinard, 2015; Mechtaev et al., 2016; Gupta et al., 2017). Our work is different in that we are combining the denoising aspect into the auto-generated SQL queries to complement the limitation of neural generation, which has not been previously considered.

## 3 Proposed Approach

### 3.1 NL2pSQL Dataset

To obtain human-written pseudo SQL statements with their corresponding natural language description, we turn to Stack Overflow (SO). We use the public SO data from the Stack Exchange Data Dump website for convenience purposes as the data are already structured in an XML form, e.g., 'Posts.xml', for easy extraction. Our goal is to curate question and code pairs: For example, in the SO Question 6308594, [4] the natural language input is *"how can I copy data from one column to another in the same table"*, and the code is *"update table set columnB = columnA"*.

In particular, we collect all the answers where they have a single code block and their corresponding questions are labeled with an *SQL* tag from 'Posts.xml'. Note that this xml file contains all the SO questions and answers since its establishment from August 2008 to September 2017. We also used the data from 'Postlinks.xml' (described in the later paragraph).

A code block can contain multiple lines of codes. We observed that indentation for code readability made numerous SQL statements be represented with multiple lines. A SQL statement indented with a different style will have a trivial effect when being read and executed by a SQL parser. Thus, we transformed all the multi-line SQL statements into a single line by replacing the line break with a white space. In this study, we only consider the SQL statement with a maximum token (delimited by a white space) length of 100.

Among questions with SQL answers, we avoid error debugging questions but focus on implementation questions with code answers. To this end, we first collect all the QA pairs tagged with SQL. We then extract the title from the QA pairs for classifying the implementation questions, as the title description has been reported as sufficient (Iyer et al., 2016). We only keep QA pairs where their question titles start with "how",[5] and not containing the following keywords *bug, error, fix* within the title content to exclude questions on error debugging. We manually inspected these QA pairs to ensure all of them to be implementation-related questions. As a result, 1,225 QA pairs in total were found.

Another issue is redundancy: 'Postlinks.xml' may also contain recurring question pairs from SO, asking the same questions in lexically different forms (or paraphrased) (Chen and Kim, 2015). We found 19 such recurring questions, and we union them to the previously mentioned 1,225 QA pairs. A total of 1,244 implementation question QA pairs were collected.

We observed that the queries in our dataset mostly involve complex conditions such as the combination of multiple joined tables with ordering and grouping clauses that are not supported by existing NL2pSQL approaches. A majority of the natural text contains 11 to 40 tokens, with SQL queries containing 11 and 15 tokens. To gain more insights on the difficulty of SQL queries, we measured Halstead complexity metrics (Halstead et al., 1977) on the training, development, and test sets and found most queries fall under the complexity score of 0-5 and well-split with near-identical distributions across the three splits.

### 3.2 Model: NL2pSQL with Denoising

While semantic parsing approaches require the machine-executable code for training (as outlined in Section 2), we focus on a new generative model without executability requirements for our proposed scenario. Essentially, our model consists of two layers of autoencoder architecture (Figure 1). The first layer is for generating SQL code given a natural language as input (NL2pSQL) through an autoencoder. The input to the first autoencoder is the word embedding of the natural language.

However, this model suffers a well-known problem of low-quality generation, such as token repetitions (Holtzman et al., 2018), or "ill-formed". Though our approach does not require a code segment to be machine-executable, generated codes

[4] https://stackoverflow.com/questions/6308594

[5] We are aware that the same intention can be paraphrased, and we can use paraphrased patterns to increase recall (and the data size). However, we focus on maintaining a high-precision set to start with.

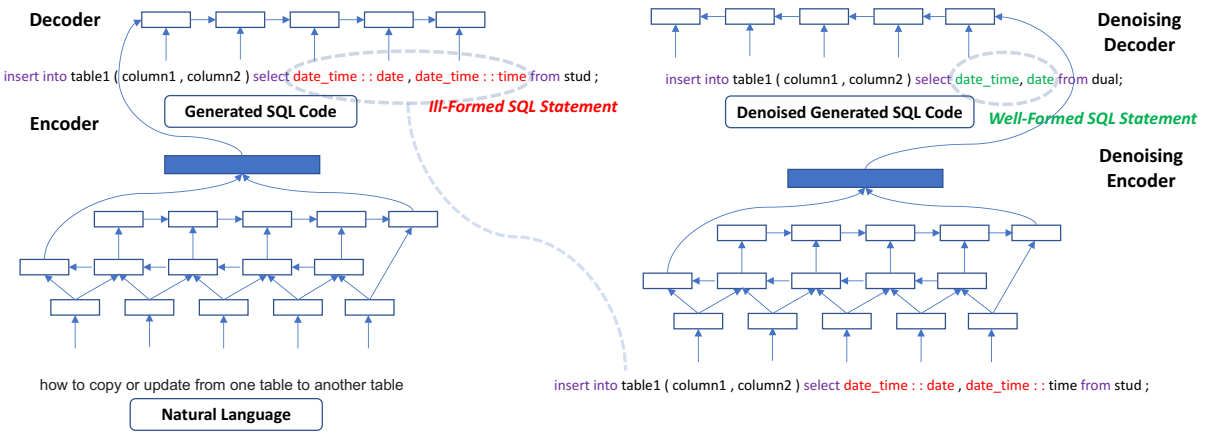


Figure 1: Overview of NL2pSQL with Denoising Autoencoder. Our model performs a two-step translation by cascading two sequence-to-sequence models. The input layers are the natural-language word embeddings in the 1st autoencoder and the SQL statement (generated by the first autoencoder) word embeddings in the 2nd autoencoder.

should be sufficiently "well-formed" to ensure human readablity. For example, a generated code with repetitive tokens will be considered ill-formed and transformed into its well-formed counterpart (or, denoising). Our contribution is thus to denoise the ill-formed queries by employing a subsequent autoencoder architecture. The input to the second autoencoder is the word embeddings of the generated SQL statements from the first autoencoder. This denoising functionality is included in the second layer with the goal of producing well-formed SQL statements.

For training the denoiser, we propose to synthesize a pair of ill- and well-formed SQL statements from the given dataset: <a *well-formed SQL statement*, an *ill-formed SQL statement*> as the input to the denoising autoencoder module. Formally, we define a well-formed SQL statement as a valid SQL statement recognized by an SQL parser whereas an ill-formed SQL statement is detected as an erroneous SQL statement construct.

We first collect well-formed SQL statements among SQL statements written by developers in SO using MySQL parser (GuduSoftware, 2018), as MySQL is the most popular SQL dialect among developers (StackOverflowSurvey, 2018). Next, we generate its ill-formed counterpart, by perturbing each of the collected well-formed SQL statements. Specifically, we randomly decide to inject into or remove from the well-formed SQL statement a token for $k$ iterations to generate multiple ill-formed statements for a given well-formed SQL statement. In our model, we set $k$ to be four.

To the best of our knowledge, this is the first NL2pSQL model that is not restricted to the executability from specified questions. This work is also the first in attempting to fix (denoise) ill-formed SQL statements through a neural network-based autoencoder architecture.

### 3.3 Evaluation Metrics

To evaluate the generated SQL statements, we first check if the generated SQL statements are well-formed (Section 3.3.1). In addition, we propose two new BLEU variants. Previous studies have reported that the BLEU score may be inappropriate in measuring the correctness of the generated code against the target code (Lin et al., 2018; Zhong et al., 2017). This is because a generated code may be written differently but still perform the intended function. The first example in Table 6 illustrates this problem. The generated SQL statement performs the semantics as required in the natural language but is structurally different as compared to the ground truth (GT). By using the vanilla BLEU, it produces a much heavily penalized score of 0.065. Thus, in our study, we use two modified versions of BLEU, namely Canonical-BLEU and SQL-BLEU, as will be described in Section 3.3.2.

#### 3.3.1 Well-Formed SQL Statements

To evaluate the effectiveness of our denoising autoencoder, we compare the relative number of well-formed SQL statements between the generated SQL statements before and after denoising. Whether a particular SQL statement is well-formed is determined by using a SQL parser.

#### 3.3.2 Canonical-BLEU and SQL-BLEU

Here, we first motivate the need for introducing two new BLEU variants, namely Canonical-BLEU and SQL-BLEU, before describing them in more detail. Canonical-BLEU first abstracts the identifiers of the SQL statement before calculating the BLEU score, while SQL-BLEU only considers the SQL keywords within the SQL statement when computing the BLEU score.

We note that two queries with the same semantics may have syntactically different forms. For example, a join statement (**S1**) such as "*select a.car from a,b where a.id=b.id*" can be equivalent to a nested select statement (**S2**) such as "*select a.car from a where a.id in (select b.id from b where b.name='Mary')*". When **S1** is generated and the ground truth is a similar join statement, it will have a correct and high SQL-BLEU score, yet having lower computing complexity. When the ground truth is similar to **S2**, SQL-BLEU and Vanilla BLEU would penalize more heavily than our Canonical-BLEU. In this respect, both SQL-BLEU and Canonical-BLEU complement each other and should be used together.

**Canonical-BLEU** Generated SQL statements may have different identifiers such as variable and table names, as compared to the original (ground truth) SO SQL statements. The difference in the identifiers' name between the generated SQL statements and the ground truth should not be heavily penalized if their SQL forms are semantically the same. To mitigate this issue, we first "canonicalize" all the identifier names (e.g., MY_COLUMN) into indexed placeholders (e.g., var_1). This transformation is performed both on the ground truth SQL statements and the generated SQL statements, and their BLEU scores are calculated. We call this Canonical-BLEU.

**SQL-BLEU** In a SQL statement, several words are considered keywords that serve as particular functions. For example, *SELECT* and *UPDATE* are both keywords in the SQL language. *SELECT* is used for retrieving a record from a relational database table whereas *UPDATE* is used to modify the content of a record. In SO, developers are free to adopt and use different column or table names. However, despite that, if the SQL code is about the retrieval of records, there should have a *SELECT* keyword within the SQL statement. Furthermore, we analyze the syntactically correct answers from SO between the period of 2008 and 2017 and found that 362 distinct SQL keywords (e.g., *YEAR, DATABASES, USING, UNIQUE, SELECT, UPDATE, TRUNCATE, etc*) are used, and 19% of the answers (that are well-formed) do not contain the *SELECT* keyword. We also found that multiple different variations of retrieval-type answers (i.e., 348 different keywords are used together with *SELECT*) use different combinations of keywords such as *WHEN, EXIST, INNER JOIN, IN, GROUP BY, etc*. Thus, we decided to use SQL keywords as another form of measure.

We first transform the generated SQL statements into their token types using a SQL parser. Each word will have its corresponding token type. For example, the word *SELECT* will be tagged as a *keyword* by the SQL parser. We remove all words that are non-keywords and only those words tagged as keywords are left in the SQL statements (we preserve their word ordering) to calculate the BLEU score. This transformation is performed both on the ground truth SQL code as well as the generated SQL statements and their BLEU scores are calculated. We call this SQL-BLEU.

## 4 Experiment Setup

### 4.1 Data Description

The used dataset is randomly split into train, dev and test sets in the ratio 60:20:20 to be mutual exclusive.

### 4.2 Baseline and Backbone Models

We stress that our work is focused on under-specified NL2pSQL and is designed for users to read/adjust the missing context for their specific usage. This is the first work on under-specified NL2pSQL and is mostly a resource and new problem/metric definition paper. Note that existing NL2pSQL work is all fully-specified and cannot be used for comparison as they involve semantic parsing and query execution on a valid database schema, which is absent in our context. In addition, making the system fully-specified is beyond the scope of this paper.

We conduct experiments for NL2pSQL generation with denoising on two neural machine translation models, Seq2Seq and CopyNet, which have shown competitive performance on natural-language-to-code translation (Lin et al., 2018). In addition, we carry out experiments in both token and character levels on both of these models, denoted as Seq2Seq(Token), Seq2Seq(Char), Copy-

Net(Token), CopyNet(Char). For each model, we perform a two-step translation by cascading two sequence-to-sequence models, for pSQL generation and denoising respectively. Each model is trained with the following hyperpameters: [6]

- Seq2Seq (Cho et al., 2014) encodes questions and decodes into queries using recurrent neural networks (RNNs). The encoder dimension is 200 by using a bi-directional RNN with concatenation. The dimension of the decoder RNN is set as 100. We optimized our objective function with Adam (Kingma and Ba, 2015), using the default momentum hyperparameters. We set the initial learning rate as 0.0001 and the mini-batch size as 32. We also used variational RNN dropout (Gal and Ghahramani, 2016) with a dropout rate of 0.4. During beam-search decoding (Hwang and Chang, 2007), we set the beam size as 100, and we ran for 100 epochs to train the model. For beam size, it has been reported that a larger beam size will lead to a higher translation quality at the expense of the decoder speed (Freitag and Al-Onaizan, 2017). We wanted to find a balance between translation quality and decoding speed. We tested different beam sizes and an entire decoding process to generate a SQL statement takes only a few seconds when the beam size is set to 100.

- CopyNet (Gu et al., 2016) extends the model above, to selectively "copy" some input sequence into output, using the same hyperparameters discussed above.

- The information retrieval (IR)-based baseline is a Lucene-based code retrieval tool, widely adopted for code search such as Sourcerer (Linstead et al., 2009). It retrieves top-3 matches, based on answer titles, matching the given question.

All the experiments were performed based on NAVER Smart Machine Learning (NSML) (Kim et al., 2018; Sung et al., 2017).

[6]Following description is for generation, and when the same model is used for denoising, we make a minor change: mini-batch size (set as 16 rather than 32), and the beam size (set as 10 instead of 100), due to a relatively large size of the dataset. Character-level encoding for denoising is unreported, as it is empirically inferior to that of token-level, as argued in prior work (Lin et al., 2018) as well.

## 5 Results

This section shows the quantitative (Sections 5.1 and 5.2) as well as qualitative analysis (Section 5.3) from our experiments.

### 5.1 Well-Formed SQL Statements

Table 1 shows the change in the number of SQL statements determined as well-formed by the SQL parser, after going through our denosing autoencoder module trained with a Seq2Seq and a CopyNet model.

Our module certainly increases the number of well-formed SQL statements. For example, when paired with a *Seq2Seq denoising autoencoder* for the top-1 prediction, the number of well-formed statements increases from 47 to 66 (19% increase). Such increase ranges from 12% to 19%, from 14% to 19%, from 30% to 43%, and from 0% to 7%, in Seq2Seq (Token), CopyNet (Token), Seq2Seq (Char), and CopyNet (Char), respectively. When paired with a *CopyNet Denoising autoencoder*, such increase is less significant. When comparing token- and character-level generation after denoising, we can see that token-level generation shows higher increase, as shown in **bold**.

### 5.2 Comparison of Canonical and SQL-BLEU

Here, we report the mean scores of both Canonical-BLEU and SQL-BLEU over all the generated SQL statements in the test set (including their denoised version) using the Seq2Seq and CopyNet autoencoder models in both the token and character levels.

#### 5.2.1 Mean SQL-BLEU

Tables 2 and 3 show the changes in the mean SQL-BLEU scores on the first three generated SQL statements, after using denoising autoencoders trained with a Seq2Seq and a CopyNet model, respectively.

For example, when paired with a Seq2Seq denoising autoencoder for the top-1 prediction, the mean SQL-BLEU scores has a negligible decrease of 0.01 (from 0.28 to 0.27). Such decrease ranges from 0.01 to 0.01, from 0.01 to 0.01, from 0.02 to 0.03, and from 0.02 to 0.02, in Seq2Seq(Token), CopyNet (Token), Seq2Seq (Char), and CopyNet(Char), respectively. When paired with a *CopyNet Denoising autoencoder*, we observed similar negligible decrease in the

Table 1: % of well-formed, generated SQL queries using a Seq2Seq/CopyNet & after denoising autoencoder models. P1, P2, and P3 correspond to the first three predictions. Columns (2, 4, 6, 8) & (3, 5, 7, 9) show the increased in well-formed queries after going through a Seq2Seq & CopyNet denoising autoencoder respectively.

| | **P1** | | **P2** | | **P3** | | **Top 3 (%)** | |
|---|---|---|---|---|---|---|---|---|
| | Seq2Seq | CopyNet | Seq2Seq | CopyNet | Seq2Seq | CopyNet | Seq2Seq | CopyNet |
| Seq2Seq (Token) | 47 → **66** | 47 → 50 | 37 → **49** | 37 → 40 | 33 → **52** | 33 → 35 | 59 → **80** | 59 → 64 |
| CopyNet (Token) | 49 → **65** | 49 → 51 | 40 → **54** | 40 → 40 | 33 → **52** | 33 → 33 | 61 → **80** | 61 → 64 |
| Seq2Seq (Char) | 31 → **74** | 31 → 35 | 29 → **64** | 29 → 31 | 33 → **63** | 33 → 33 | 40 → **79** | 40 → 43 |
| CopyNet (Char) | 29 → **35** | 29 → 10 | 27 → **34** | 27 → 15 | 32 → **32** | 32 → 14 | 40 → **45** | 40 → 31 |

Table 2: Negligible change in mean SQL-BLEU score & changes in delta after Seq2Seq denoising in ().

| | **P1** | **P2** | **P3** |
|---|---|---|---|
| Seq2Seq (T) | 0.28 (-0.01) | 0.30 (-0.01) | 0.29 (-0.01) |
| CopyNet (T) | 0.30 (-0.01) | 0.30 (-0.01) | 0.28 (-0.01) |
| Seq2Seq (C) | 0.26 (-0.03) | 0.26 (-0.02) | 0.26 (-0.02) |
| CopyNet (C) | 0.18 (-0.02) | 0.18 (-0.02) | 0.17 (-0.01) |
| IR | 0.29 | 0.29 | 0.27 |

Table 3: Negligible change in mean SQL-BLEU score & changes in delta after CopyNet denoising in ().

| | **P1** | **P2** | **P3** |
|---|---|---|---|
| Seq2Seq (T) | 0.28 (-0.01) | 0.30 (-0.01) | 0.29 (0.00) |
| CopyNet (T) | 0.30 (-0.01) | 0.30 (-0.01) | 0.28 (0.00) |
| Seq2Seq (C) | 0.26 (0.00) | 0.26 (-0.01) | 0.26 (-0.01) |
| CopyNet (C) | 0.18 (-0.02) | 0.18 (-0.02) | 0.17 (-0.01) |
| IR | 0.29 | 0.29 | 0.27 |

Table 4: Negligible change in mean Canonical-BLEU score & changes in delta after Seq2Seq denoising in ().

| | **P1** | **P2** | **P3** |
|---|---|---|---|
| Seq2Seq (T) | 0.27 (0.00) | 0.31 (-0.01) | 0.31 (0.00) |
| CopyNet (T) | 0.27 (0.00) | 0.30 (+0.01) | 0.31 (+0.01) |
| Seq2Seq (C) | 0.23 (-0.01) | 0.26 (-0.02) | 0.29 (-0.01) |
| CopyNet (C) | 0.29 (+0.02) | 0.30 (+0.02) | 0.30 (+0.03) |
| IR | 0.23 | 0.23 | 0.24 |

Table 5: Negligible change in mean Canonical-BLEU score & changes in delta after CopyNet denoising in ().

| | **P1** | **P2** | **P3** |
|---|---|---|---|
| Seq2Seq (T) | 0.27 (0.00) | 0.31 (+0.01) | 0.31 (+0.01) |
| CopyNet (T) | 0.27 (0.00) | 0.30 (+0.01) | 0.31 (0.00) |
| Seq2Seq (C) | 0.23 (+0.01) | 0.26 (0.00) | 0.29 (-0.01) |
| CopyNet (C) | 0.29 (-0.01) | 0.30 (0.00) | 0.30 (-0.01) |
| IR | 0.23 | 0.23 | 0.24 |

mean SQL-BLEU after denoising. Compared to IR approach, token level approaches mostly yield higher SQL-BLEU scores.

#### 5.2.2 Mean Canonical-BLEU

Table 4 and 5 show the change in the mean Canonical-BLEU scores on the first three generated SQL statements, after using denoising autoencoders trained with a Seq2Seq and a CopyNet model respectively.

For example, when paired with a Seq2Seq denoising autoencoder for the top-1 prediction, the mean Canonical-BLEU scores has a negligible decrease in Seq2Seq (Token), from 0.00 to 0.01, or 0.01 decrease. Such decrease ranges from 0.00 to 0.02, from 0.00 to 0.01, from 0.01 to 0.02, and from 0.02 to 0.03, in Seq2Seq(Token), CopyNet (Token), Seq2Seq (Char), and CopyNet(Char), respectively. When paired with a *CopyNet Denoising autoencoder*, we observed similar negligible decrease in the mean SQL-BLEU after denoising. Compared to the IR approach, we observed that most of the generative approaches yield higher Canonical-BLEU scores.

In Section 5.1, we previously reported up to 43% more well-formed SQL statements in the denoised version than the non-denoised version. This suggests that the two metrics (*Well-Formed SQL Statements and the two variants of BLEU – Canonical BLEU and SQL BLEU*) we proposed are complementary, in that Canonical-BLEU and SQL-BLEU alone cannot distinguish well-formed SQL statements. We also note that for denoising, while enhancing one metric, it does not hurt another.

### 5.3 Qualitative Analysis

Table 6 shows the output examples generated by our models in the unseen data (the test set), including the denoised version. We only show the generated outputs from the Seq2Seq model as they yield the best results in our experiments. Each example consists of the natural language input, the ground truth, the top prediction of the generated output from the Seq2Seq model, and its denoised version.

The first example is about copying or updating from one table to another. In the generated SQL statement, it performs the intended copying function but is syntactically incorrect in the use of

*date_time*. The denoised version fixes it by replacing *date_time : : date* and *date_time : : time* with proper column names. Furthermore, the ground truth is coupled with comments, and its SQL statement is much longer than the generated version. The retrieved statement by the IR baseline has similar semantics on observing the same type of an insertion query with our approach. Although both of them are similar, the IR baseline appears more complicated as it contains more columns for insertion and a nested table structure. We believe that the generated SQL statement is a much better choice in this case.

The second example is about counting the number of rows. The generated SQL statement shows the counting-related code. Interestingly, the IR approach produces the exact query. In the last example, it is about inserting a record from another table. Similar to the IR approach, the generated SQL statement is able to churn out a query performing an insertion operation from another table. Both generated SQL statements from the last two examples do not contain any syntax errors, and the denoised autocoders did not change the SQL statements. In the last example, it is difficult to include "*top500*" which requires verbal explanation of defining one's own function in the "*order by*" clause. This part was omitted, but all the other semantics are fully preserved.

## 6 Conclusions

This paper addressed the problem of generating human-readable pseudo SQL queries from natural-languge questions without code-executability restriction, and propose a new dataset. Despite excluding executability, our task covers much more diverse issues including table manipulation and performance-related problems beyond simple table lookup. This coverage allows our task to help software engineers to deal with various practical implementation and operation issues. In addition, we describe a new NL2pSQL model and two evaluation metrics suitable for dealing with our task. Our result shows that adding a denoising autoencoder module enables our model to generate a more number of well-formed queries, to complement a known weakness of neural code generation.

Table 6: Generated examples

| | |
|---|---|
| NL | how to copy or update from one table to another table |
| GT | - - 1 ) insert new record insert into old_table ( id , column ) select n . id , n . column from new_table n left join old_table o on n . id = o . id where o . id is null - - 2 ) update existed record update o set column = n . column from old_table o join new_table n on n . id = o . id |
| Seq2Seq | insert into table1 ( column1 , column2 ) select date_time : : date , date_time : : time from stud ; |
| Seq2Seq (D) | insert into table1 ( column1 , column2 ) select date_time , date from dual ; |
| IR | insert into test1 . dbo . bcd ( col1 , col2 , col3 ) select col1 , col2 , col3 from test . dbo . abc |
| NL | how can i count the number of rows in a database for different time periods |
| GT | declare @ interval int set @ interval = 5 select datepart ( hh , datetimecolumn ) , datepart ( mi , datetimecolumn ) / @ interval * @ interval , count ( * ) from thetable group by datepart ( hh , datetimecolumn ) , datepart ( mi , datetimecolumn ) / @ interval * @ interval |
| Seq2Seq | select count ( * ) from ( select distinct * from table_name ) t |
| Seq2Seq (D) | select count ( * ) from ( select distinct * from table_name ) t |
| IR | select count ( * ) from ( select distinct * from table_name ) t |
| NL | how can i select top 500 rows from table1 and insert them to table2 |
| GT | insert into emptytable select top 500 * from populatedtable order by what - - need this to define top 500 |
| Seq2Seq | insert into destination ( col1 , col2 , col3 , col4 , col5 ) select ' static_value1 ' , ' static_value2 ' , col6 , col7 , col8 from source_table |
| Seq2Seq (D) | insert into destination ( col1 , col2 , col3 , col4 , col5 ) select ' static_value1 ' , ' static_value2 ' , col6 , col7 , col8 from source_table |
| IR | insert into temp ( time , name ) select getdate ( ) , name from name |

## 7 Acknowledgement

Hwang was partly supported by IITP funded by MSIT (2017-0-0177; XAI).

## References


Matej Balog, Alexander L Gaunt, Marc Brockschmidt, Sebastian Nowozin, and Daniel Tarlow. 2017. Deepcoder: Learning to write programs. In *International Conference on Learning Representations*.

Tony Beltramelli. 2017. pix2code: Generating code from a graphical user interface screenshot. *arXiv preprint arXiv:1705.07962*.

Florin Brad, Radu Iacob, Ionel Hosu, and Traian Rebedea. 2017. Dataset for a neural natural language interface for databases (nnlidb). *arXiv preprint arXiv:1707.03172*.

Antoni Buades, Bartomeu Coll, and J-M Morel. 2005. A non-local algorithm for image denoising. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 60–65. IEEE.

Fuxiang Chen and Sunghun Kim. 2015. Crowd debugging. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2015, pages 320–332, New York, NY, USA. ACM.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics.

Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. 2007. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on image processing*, 16(8):2080–2095.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43, Berlin, Germany. Association for Computational Linguistics.

Michael Elad and Michal Aharon. 2006. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image processing*, 15(12):3736–3745.

Catherine Finegan-Dollak, Jonathan K Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving text-to-sql evaluation methodology.

Markus Freitag and Yaser Al-Onaizan. 2017. Beam search strategies for neural machine translation. *arXiv preprint arXiv:1702.01806*.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027.

Alessandra Giordani and Alessandro Moschitti. 2012. Translating questions to sql queries with generative parsers discriminatively reranked. *Proceedings of COLING 2012: Posters*, pages 401–410.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640. Association for Computational Linguistics.

GuduSoftware. 2018. Sql parse, analyze, transform and format all in one.

Rahul Gupta, Soham Pal, Aditya Kanade, and Shirish Shevade. 2017. Deepfix: Fixing common c language errors by deep learning. In *AAAI*, pages 1345–1351.

Maurice H Halstead et al. 1977. *Elements of Software Science (Operating and programming systems series)*. Elsevier Science Inc., New York, NY.

Charles T Hemphill, John J Godfrey, and George R Doddington. 1990. The atis spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*.

Ari Holtzman, Jan Buys, Maxwell Forbes, Antoine Bosselut, David Golub, and Yejin Choi. 2018. Learning to write with cooperative discriminators. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1638–1649. Association for Computational Linguistics.

Seung-won Hwang and Kevin Chang. 2007. Probe minimization by schedule optimization: Supporting top-k queries with expensive predicates. In *IEEE TKDE*.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 963–973, Vancouver, Canada. Association for Computational Linguistics.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. 2016. Summarizing source code using a neural attention model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2073–2083. Association for Computational Linguistics.

Hanjoo Kim, Minkyu Kim, Dongjoo Seo, Jinwoong Kim, Heungseok Park, Soeun Park, Hyunwoo Jo, KyungHyun Kim, Youngil Yang, Youngkwan Kim,

et al. 2018. Nsml: Meet the mlaas platform with a real-world case study. *arXiv preprint arXiv:1810.09957*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization.

Kazuhei Kouno, Hiroyuki Shinnou, Minoru Sasaki, and Kanako Komiya. 2015. Unsupervised domain adaptation for word sense disambiguation using stacked denoising autoencoder. In *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation: Posters*, pages 224–231.

Carolin Lawrence and Stefan Riezler. 2016. Nlmaps: A natural language interface to query openstreetmap. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, pages 6–10.

Fei Li and Hosagrahar V Jagadish. 2014. Nalir: an interactive natural language interface for querying relational databases. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 709–712. ACM.

Yunyao Li, Huahai Yang, and HV Jagadish. 2006. Constructing a generic natural language interface for an xml database. In *International Conference on Extending Database Technology*, pages 737–754. Springer.

Xi Victoria Lin, Chenglong Wang, Luke Zettlemoyer, and Michael D. Ernst. 2018. Nl2bash: A corpus and semantic parser for natural language interface to the linux operating system. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation LREC 2018, Miyazaki (Japan), 7-12 May, 2018*.

Wang Ling, Phil Blunsom, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, Fumin Wang, and Andrew Senior. 2016. Latent predictor networks for code generation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 599–609. Association for Computational Linguistics.

Erik Linstead, Sushil Bajracharya, Trung Ngo, Paul Rigor, Cristina Lopes, and Pierre Baldi. 2009. Sourcerer: mining and searching internet-scale software repositories. *Data Mining and Knowledge Discovery*, 18(2):300–336.

Fan Long and Martin Rinard. 2015. Staged program repair with condition synthesis. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, pages 166–178. ACM.

Sergey Mechtaev, Jooyong Yi, and Abhik Roychoudhury. 2016. Angelix: Scalable multiline program patch synthesis via symbolic analysis. In *Software Engineering (ICSE), 2016 IEEE/ACM 38th International Conference on*, pages 691–701. IEEE.

William Meiniel, Jean-Christophe Olivo-Marin, and Elsa D Angelini. 2018. Denoising of microscopy images: A review of the state-of-the-art, and a new sparsity-based method. *IEEE Transactions on Image Processing*, 27(8):3842–3856.

Vijayaraghavan Murali, Letao Qi, Swarat Chaudhuri, and Chris Jermaine. 2018. Neural sketch learning for conditional program generation. In *International Conference on Learning Representations*.

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2017. Jfleg: A fluency corpus and benchmark for grammatical error correction. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 229–234. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ana-Maria Popescu, Alex Armanasu, Oren Etzioni, David Ko, and Alexander Yates. 2004. Modern natural language interfaces to databases: Composing statistical parsing with semantic tractability. In *Proceedings of the 20th international conference on Computational Linguistics*, page 141. Association for Computational Linguistics.

Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. 2003. Towards a theory of natural language interfaces to databases. In *Proceedings of the 8th international conference on Intelligent user interfaces*, pages 149–157. ACM.

Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract syntax networks for code generation and semantic parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1139–1149. Association for Computational Linguistics.

Senjuti Basu Roy, Martine De Cock, Vani Mandava, Swapna Savanna, Brian Dalessandro, Claudia Perlich, William Cukierski, and Ben Hamner. 2013. The microsoft academic search dataset and kdd cup 2013. In *Proceedings of the 2013 KDD cup 2013 workshop*, page 1. ACM.

StackOverflow. 2018. Stack overflow.

StackOverflowSurvey. 2018. Stack overflow survey 2018.

Yibo Sun, Duyu Tang, Nan Duan, Jianshu Ji, Guihong Cao, Xiaocheng Feng, Bing Qin, Ting Liu, and Ming Zhou. 2018. Semantic parsing with syntax- and table-aware sql generation. In *Proceedings of the*

*56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 361–372. Association for Computational Linguistics.

Nako Sung, Minkyu Kim, Hyunwoo Jo, Youngil Yang, Jingwoong Kim, Leonard Lausen, Youngkwan Kim, Gayoung Lee, Donghyun Kwak, Jung-Woo Ha, et al. 2017. Nsml: A machine learning platform that enables you to focus on your models. *arXiv preprint arXiv:1712.05902*.

Lappoon R Tang and Raymond J Mooney. 2000. Automated construction of database interfaces: Integrating statistical and relational learning for semantic parsing. In *Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics-Volume 13*, pages 133–141. Association for Computational Linguistics.

David HD Warren and Fernando CN Pereira. 1982. An efficient easily adaptable system for interpreting natural language queries. *Computational Linguistics*, 8(3-4):110–122.

Xiaojun Xu, Chang Liu, and Dawn Song. 2018. SQLNet: Generating structured queries from natural language without reinforcement learning. ICLR.

Navid Yaghmazadeh, Yuepeng Wang, Isil Dillig, and Thomas Dillig. 2017. Type-and content-driven synthesis of sql queries from natural language. *arXiv preprint arXiv:1702.01168*.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. *arXiv preprint arXiv:1704.01696*.

Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. 2018a. Typesql: Knowledge-based type-aware neural text-to-sql generation. *arXiv preprint arXiv:1804.09769*.

Tao Yu, Michihiro Yasunaga, Kai Yang, Rui Zhang, Dongxu Wang, Zifan Li, and Dragomir Radev. 2018b. Syntaxsqlnet: Syntax tree networks for complex and cross-domaintext-to-sql task.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 2018c. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task.

John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the national conference on artificial intelligence*, pages 1050–1055.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, UAI'05, pages 658–666, Arlington, Virginia, United States. AUAI Press.

Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. 2017. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

# Leveraging Frequent Query Substructures to Generate Formal Queries for Complex Question Answering

**Jiwei Ding  Wei Hu*  Qixin Xu  Yuzhong Qu***
State Key Laboratory for Novel Software Technology, Nanjing University, China
{jwding, qxxu}nju@outlook.com, {whu, yzqu}@nju.edu.cn

## Abstract

Formal query generation aims to generate correct executable queries for question answering over knowledge bases (KBs), given entity and relation linking results. Current approaches build universal paraphrasing or ranking models for the whole questions, which are likely to fail in generating queries for complex, long-tail questions. In this paper, we propose SubQG, a new query generation approach based on frequent query substructures, which helps rank the existing (but nonsignificant) query structures or build new query structures. Our experiments on two benchmark datasets show that our approach significantly outperforms the existing ones, especially for complex questions. Also, it achieves promising performance with limited training data and noisy entity/relation linking results.

## 1 Introduction

Knowledge-based question answering (KBQA) aims to answer natural language questions over knowledge bases (KBs) such as DBpedia and Freebase. Formal query generation is an important component in many KBQA systems (Bao et al., 2016; Cui et al., 2017; Luo et al., 2018), especially for answering complex questions. Given entity and relation linking results, formal query generation aims to generate correct executable queries, e.g., SPARQL queries, for the input natural language questions. An example question and its formal query are shown in Figure 1. Generally speaking, formal query generation is expected to include but not be limited to have the capabilities of (i) recognizing and paraphrasing different kinds of constraints, including triple-level constraints (e.g., *"movies"* corresponds to a typing constraint for the target variable) and higher level constraints (e.g., subgraphs). For instance, *"the same ... as"*

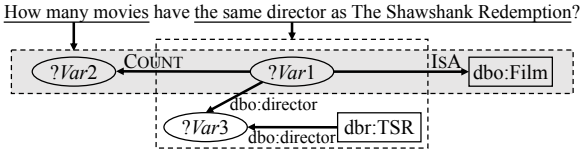

Figure 1: An example for complex question and query

represents a complex structure shown in the middle of Figure 1; (ii) recognizing and paraphrasing aggregations (e.g., "how many" corresponds to COUNT); and (iii) organizing all the above to generate an executable query (Singh et al., 2018; Zafar et al., 2018).

There are mainly two kinds of query generation approaches for complex questions. (i) Template-based approaches choose a pre-collected template for query generation (Cui et al., 2017; Abujabal et al., 2017). Such approaches highly rely on the coverage of templates, and perform unstably when some complex templates have very few natural language questions as training data. (ii) Approaches based on semantic parsing and neural networks learn entire representations for questions with different query structures, by using a neural network following the encode-and-compare framework (Luo et al., 2018; Zafar et al., 2018). They may suffer from the lack of training data, especially for long-tail questions with rarely appeared structures. Furthermore, both above approaches cannot handle questions with unseen query structures, since they cannot generate new query structures.

To cope with the above limitations, we propose a new query generation approach based on the following observation: the query structure for a complex question may rarely appear, but it usually contains some substructures that frequently appeared in other questions. For example, the query structure for the question in Figure 1 appears rarely, however, both *"how many movies"* and *"the same ... as"* are common expressions,

*Corresponding authors



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2614–2622, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

which correspond to the two query substructures in dashed boxes. To collect such frequently appeared substructures, we automatically decompose query structures in the training data. Instead of directly modeling the query structure for the given question as a whole, we employ multiple neural networks to predict query substructures contained in the question, each of which delivers a part of the query intention. Then, we select an existing query structure for the input question by using a combinational ranking function. Also, in some cases, no existing query structure is appropriate for the input question. To cope with this issue, we merge query substructures to build new query structures. The contributions of this paper are summarized below:

- We formalize the notion of query structures and define the substructure relationship between query structures.
- We propose a novel approach for formal query generation, which firstly leverages multiple neural networks to predict query substructures contained in the given question, and then ranks existing query structures by using a combinational function.
- We merge query substructures to build new query structures, which handles questions with unseen query structures.
- We perform extensive experiments on two KBQA datasets, and show that SubQG significantly outperforms the existing approaches. Furthermore, SubQG achieves a promising performance with limited training data and noisy entity/relation linking results.

## 2 Preliminaries

An entity is typically denoted by a URI and described with a set of properties and values. A fact is an $\langle entity, property, value\rangle$ triple, where the value can be either a literal or another entity. A KB is a pair $\mathcal{K} = (\mathcal{E}, \mathcal{F})$, where $\mathcal{E}$ denotes the set of entities and $\mathcal{F}$ denotes the set of facts.

A *formal query* (or simply called *query*) is the structured representation of a natural language question executable on a given KB. Formally, a query is a pair $\mathcal{Q} = (\mathcal{V}, \mathcal{T})$, where $\mathcal{V}$ denotes the set of vertices, and $\mathcal{T}$ denotes the set of labeled edges. A vertex can be either a variable, an entity or a literal, and the label of an edge can be either

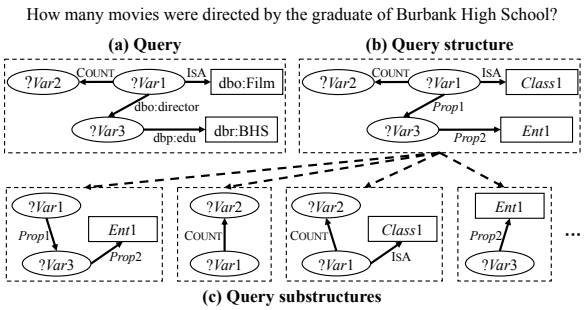

Figure 2: Illustration of a query, a query structure and query substructures

a built-in property or a user-defined one. For simplicity, the set of all edge labels are denoted by $\mathcal{L}_e(\mathcal{Q})$. In this paper, the built-in properties include COUNT, AVG, MAX, MIN, MAXATN, MINATN and ISA (RDF:TYPE), where the former four are used to connect two variables. For example, $\langle ?Var1, \text{COUNT}, ?Var2\rangle$ represents that $?Var2$ is the counting result of $?Var1$. MAXATN and MINATN take the meaning of ORDER BY in SPARQL (Bao et al., 2016). For instance, $\langle ?Var1, \text{MAXATN}, 2\rangle$ means ORDER BY DESC($?Var1$) LIMIT 1 OFFSET 1.

To classify various queries with similar query intentions and narrow the search space for query generation, we introduce the notion of *query structures*. A query structure is a set of structurally-equivalent queries. Let $\mathcal{Q}_a = (\mathcal{V}_a, \mathcal{T}_a)$ and $\mathcal{Q}_b = (\mathcal{V}_b, \mathcal{T}_b)$ denote two queries. $\mathcal{Q}_a$ is structurally-equivalent to $\mathcal{Q}_b$, denoted by $\mathcal{Q}_a \cong \mathcal{Q}_b$, if and only if there exist two bijections $f : \mathcal{V}_a \to \mathcal{V}_b$ and $g : \mathcal{L}_e(\mathcal{Q}_a) \to \mathcal{L}_e(\mathcal{Q}_b)$ such that:

(i) $\forall v \in \mathcal{V}_a$, $v$ is a variable $\Leftrightarrow f(v)$ is a variable;

(ii) $\forall r \in \mathcal{L}_e(\mathcal{Q}_a)$, $r$ is a user-defined property $\Leftrightarrow g(r)$ is a user-defined property; if $r$ is a built-in property, $g(r) = r$;

(iii) $\forall v \forall r \forall v' \langle v, r, v'\rangle \in \mathcal{T}_a \Leftrightarrow \langle f(v), g(r), f(v')\rangle \in \mathcal{T}_b$.

The query structure for $\mathcal{Q}_a$ is denoted by $\mathcal{S}_a = [\mathcal{Q}_a]$, which contains all the queries structurally-equivalent to $\mathcal{Q}_a$. For graphical illustration, we represent a query structure by a representative query among the structurally-equivalent ones and replace entities and literals with different kinds of placeholders. An example of query and query structure is shown in the upper half of Figure 2.

For many simple questions, two query structures, i.e., $(\{?Var1, Ent1\}, \{\langle ?Var1, Prop1, Ent1\rangle\})$ and $(\{?Var1, Ent1\}, \{\langle Ent1, Prop1, ?Var1\rangle\})$, are sufficient. However, for complex

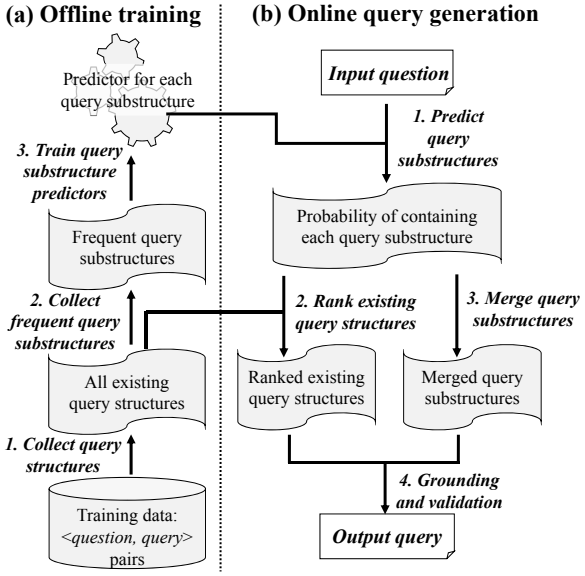


Figure 3: Framework of the proposed approach

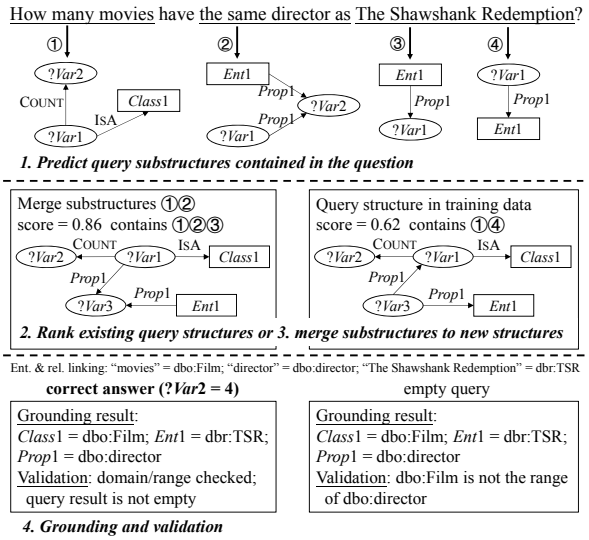


Figure 4: An example for online query generation

questions, a diversity of query structures exist and some of them share a set of frequently-appeared substructures, each of which delivers a part of the query intention. We give the definition of *query substructures* as follows.

Let $\mathcal{S}_a = [\mathcal{Q}_a]$ and $\mathcal{S}_b = [\mathcal{Q}_b]$ denote two query structures. $\mathcal{S}_a$ is a query substructure of $\mathcal{S}_b$, denoted by $\mathcal{S}_a \preceq \mathcal{S}_b$, if and only if $\mathcal{Q}_b$ has a subgraph $\mathcal{Q}_c$ such that $\mathcal{Q}_a \cong \mathcal{Q}_c$. Furthermore, if $\mathcal{S}_a = [\mathcal{Q}_a] \preceq \mathcal{S}_b = [\mathcal{Q}_b]$, we say that $\mathcal{Q}_b$ has $\mathcal{S}_a$, and $\mathcal{S}_a$ is contained in $\mathcal{Q}_b$.

For example, although the query structures for the two questions in Figures 1 and 2 are different, they share the same query substructure $(\{?Var1, ?Var2, Class1\}, \{\langle ?Var1, \text{COUNT}, ?Var2\rangle, \langle ?Var1, \text{ISA}, Class1\rangle\})$, which corresponds to the phrase "how many movies". Note that, a query substructure can be the query structure of another question.

The goal of this paper is to leverage a set of frequent query (sub-)structures to generate formal queries for answering complex questions.

## 3 The Proposed Approach

In this section, we present our approach, SubQG, for query generation. We first introduce the framework and general steps with a running example (Section 3.1), and then describe some important steps in detail in the following subsections.

### 3.1 Framework

Figure 3 depicts the framework of SubQG, which contains an offline training process and an online query generation process.

**Offline.** The offline process takes as input a set of training data in form of $\langle question, query\rangle$ pairs, and mainly contains three steps:

**1. Collect query structures.** For questions in the training data, we first discover the structurally-equivalent queries, and then extract the set of all query structures, denoted by $\mathbf{TS}$.

**2. Collect frequent query substructures.** We decompose each query structure $\mathcal{S}_i = (\mathcal{V}_i, \mathcal{T}_i) \in \mathbf{TS}$ to get the set for all query substructures. Let $\mathcal{T}_j$ be a non-empty subset of $\mathcal{T}_i$, and $\mathcal{V}_{\mathcal{T}_j}$ be the set of vertices used in $\mathcal{T}_j$. $\mathcal{S}_j = (\mathcal{V}_{\mathcal{T}_j}, \mathcal{T}_j)$ should be a query substructure of $\mathcal{S}_i$ according to the definition. So, we can generate all query substructures of $\mathcal{S}_i$ from each subset of $\mathcal{T}_i$. Disconnected query substructures would be ignored since they express discontinuous meanings and should be split into smaller query substructures. If more than $\gamma$ queries in training data have substructure $\mathcal{S}_j$, we consider $\mathcal{S}_j$ as a frequent query substructure. The set for all frequent query substructures is denoted by $\mathbf{FS}^*$.

**3. Train query substructure predictors.** We train a neural network for each query substructure $\mathcal{S}_i^* \in \mathbf{FS}^*$, to predict the probability that $\mathcal{Q}^y$ has $\mathcal{S}_i^*$ (i.e., $\mathcal{S}_i^* \preceq [\mathcal{Q}^y]$) for input question $y$, where $\mathcal{Q}^y$ denotes the formal query for $y$. Details for this step are described in Section 3.2.

**Online.** The online query generation process takes as input a natural language question $y$, and mainly contains four steps:

**1. Predict query substructures.** We first predict the probability that $\mathcal{S}_i^* \preceq [\mathcal{Q}^y]$ for each $\mathcal{S}_i^* \in \mathbf{FS}^*$, using the query substructure predictors trained in the offline step. An example question and four query substructures with highest prediction prob-

abilities are shown in the top of Figure 4.

**2. Rank existing query structures.** To find an appropriate query structure for the input question, we rank existing query structures ($\mathcal{S}_i \in \mathbf{TS}$) by using a scoring function, see Section 3.3.

**3. Merge query substructures.** Consider the fact that the target query structure $[\mathcal{Q}^y]$ may not appear in $\mathbf{TS}$ (i.e., there is no query in the training data that is structurally-equivalent to $\mathcal{Q}^y$), we design a method (described in Section 3.4) to merge question-contained query substructures for building new query structures. The merged results are ranked using the same function as existing query structures. Several query structures (including the merged results and the existing query structures) for the example question are shown in the middle of Figure 4.

**4. Grounding and validation.** We leverage the query structure ranking result, alongside with the entity/relation linking result from some existing black box systems (Dubey et al., 2018) to generate executable formal query for the input question. For each query structure, we try all possible combinations of the linking results according to the descending order of the overall linking score, and perform validation including grammar check, domain/range check and empty query check. The first non-empty query passing all validations is considered as the output for SubQG. The grounding and validation results for the example question are shown in the bottom of Figure 4.

### 3.2 Query Substructure Prediction

In this step, we employ an attention based BiLSTM network (Raffel and Ellis, 2015) to predict $\Pr[\mathcal{S}_i^* \mid y]$ for each frequent query substructure $\mathcal{S}_i^* \in \mathbf{FS}^*$, where $\Pr[\mathcal{S}_i^* \mid y]$ represents the probability of $\mathcal{S}_i^* \preceq [\mathcal{Q}^y]$. There are mainly three reasons that we use a predictor for each query substructure instead of a multi-tag predictor for all query substructures: (i) a query substructure usually expresses part of the meaning of input question. Different query substructures may focus on different words or phrases, thus, each predictor should have its own attention matrix; (ii) multi-tag predictor may have a lower accuracy since each tag has unbalanced training data; (iii) single pre-trained query substructure predictor from one dataset can be directly reused on another without adjusting the network structure, however, the multi-tag predictor need to adjust the size of the output layer and retrain when the set of frequent query substructures changes.

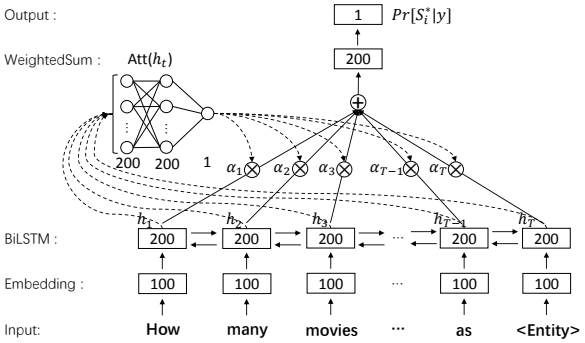


Figure 5: Attention-based BiLSTM network

The structure of the network is shown in Figure 5. Before the input question is fed into the network, we replace all entity mentions with $\langle \textit{Entity} \rangle$ using EARL (Dubey et al., 2018), to enhance the generalization ability. Given the question word sequence $\{w_1, ..., w_T\}$, we first use a word embedding matrix to convert the original sequence into word vectors $\{\mathbf{e}_1, ..., \mathbf{e}_T\}$, followed by a BiLSTM network to generate the context-sensitive representation $\{\mathbf{h}_1, ..., \mathbf{h}_T\}$ for each word, where

$$\mathbf{h}_t = [\textsc{Lstm}(\mathbf{e}_t, \overrightarrow{\mathbf{h}}_{t-1}); \textsc{Lstm}(\mathbf{e}_t, \overleftarrow{\mathbf{h}}_{t+1})]. \quad (1)$$

Then, the attention mechanism takes each $\mathbf{h}_t$ as input, and calculates a weight $\alpha_t$ for each $\mathbf{h}_t$, which is formulated as follows:

$$\alpha_t = \frac{e^{\text{Att}(\mathbf{h}_t)}}{\sum_{k=1}^{T} e^{\text{Att}(\mathbf{h}_k)}}, \quad (2)$$

$$\text{Att}(\mathbf{h}_t) = \mathbf{v}_{att}^T \tanh(\mathbf{W}_{att}\mathbf{h}_t + \mathbf{b}_{att}), \quad (3)$$

where $\mathbf{W}_{att} \in \mathbb{R}^{|\mathbf{h}_t| \times |\mathbf{h}_t|}$, $\mathbf{b}_{att} \in \mathbb{R}^{|\mathbf{h}_t|}$ and $\mathbf{v}_{att} \in \mathbb{R}^{|\mathbf{h}_t|}$. Next, we get the representation for the whole question $\mathbf{q}^c$ as the weighted sum of $\mathbf{h}_t$:

$$\mathbf{q}^c = \sum_{t=1}^{T} \alpha_t \mathbf{h}_t. \quad (4)$$

The output of the network is a probability

$$\Pr[\mathcal{S}_i^* \mid y] = \sigma(\mathbf{v}_{out}^T \mathbf{q}^c + b_{out}), \quad (5)$$

where $\mathbf{v}_{out} \in \mathbb{R}^{|\mathbf{q}^c|}$ and $b_{out} \in \mathbb{R}$.

The loss function minimized during training is the binary cross-entropy:

$$\begin{aligned} \text{Loss}(\mathcal{S}_i^*) = &- \sum_{\substack{(y,\mathcal{Q}^y)\in\mathbf{Train} \\ \text{s.t. } \mathcal{S}_i^* \preceq [\mathcal{Q}^y]}} \log(\Pr[\mathcal{S}_i^* \mid y]) \\ &- \sum_{\substack{(y,\mathcal{Q}^y)\in\mathbf{Train} \\ \text{s.t. } \mathcal{S}_i^* \not\preceq [\mathcal{Q}^y]}} \log(1 - \Pr[\mathcal{S}_i^* \mid y]), \end{aligned} \quad (6)$$

where $\mathbf{Train}$ denotes the set of training data.

**Algorithm 1:** Query substructure merging

**Input:** Question $y$, freq. query substructures $\mathbf{FS}^*$

$\mathbf{FS}^+ := \{\mathcal{S}_i^* \in \mathbf{FS}^* \mid \Pr[\mathcal{S}_i^* \mid y] > 0.5\}$;
$\mathbf{M}^{(0)} := \{\mathcal{S}_i^* \in \mathbf{FS}^* \mid \text{Score}[\mathcal{S}_i^* \mid y] > \theta\}$;
**for** $i = 1$ *to* $K$ **do** *// $K$ is maximum iterations*
  $\mathbf{M}^{(i)} := \emptyset$;
  **forall** $\mathcal{S}_i^* \in \mathbf{FS}^+, \mathcal{S}_j \in \mathbf{M}^{(i-1)}$ **do**
    $\mathbf{M}^{(i)} := \mathbf{M}^{(i)} \cup \text{Merge}(\mathcal{S}_i^*, \mathcal{S}_j)$;
  $\mathbf{M}^{(i)} := \{\mathcal{S}_l \in \mathbf{M}^{(i)} \mid \text{Score}[\mathcal{S}_l \mid y] > \theta\}$;
**return** $\bigcup_{i=0}^{K} \mathbf{M}^{(i)}$;

### 3.3 Query Structure Ranking

In this step, we use a combinational function to score each query structure in the training data for the input question. Since the prediction result for each query substructure is independent, the score for query structure $\mathcal{S}_i$ is measured by joint probability, which is

$$\text{Score}(\mathcal{S}_i \mid y) = \prod_{\substack{\mathcal{S}_j^* \in \mathbf{FS}^* \\ \text{s.t. } \mathcal{S}_j^* \preceq \mathcal{S}_i}} \Pr[\mathcal{S}_j^* \mid y] \times \prod_{\substack{\mathcal{S}_j^* \in \mathbf{FS}^* \\ \text{s.t. } \mathcal{S}_j^* \not\preceq \mathcal{S}_i}} (1 - \Pr[\mathcal{S}_j^* \mid y]). \quad (7)$$

Assume that $\mathcal{Q}^y \in \mathcal{S}_i, \forall \mathcal{S}_j^* \preceq \mathcal{S}_i$, we have $\mathcal{S}_j^* \preceq [\mathcal{Q}^y]$. Thus, $\Pr[\mathcal{S}_j^* \mid y]$ should be 1 in the ideal condition. On the other hand, $\forall \mathcal{S}_j^* \not\preceq \mathcal{S}_i$, $\Pr[\mathcal{S}_j^* \mid y]$ should be 0. Thus, we have $\text{Score}(\mathcal{S}_i \mid y) = 1$, and $\forall \mathcal{S}_k \neq \mathcal{S}_i$, we have $\text{Score}(\mathcal{S}_k \mid y) = 0$.

### 3.4 Query Substructure Merging

We proposed a method, shown in Algorithm 1, to merge question-contained query substructures to build new query structures. In the initialization step, it selects some query substructures of high scores as candidates, since the query substructure may directly be the appropriate query structure for the input question. In each iteration, the method merges each question-contained substructures with existing candidates, and the merged results of high scores are used as candidates in the next iteration. The final output is the union of all the results from at most $K$ iterations.

When merging different query substructures, we allow them to share some vertices of the same kind (variable, entity, etc.) or edge labels, except the variables which represent aggregation results. Thus, the merged result of two query substructures is a set of query structures instead of one. Also, the following restrictions are used to filter the merged results:

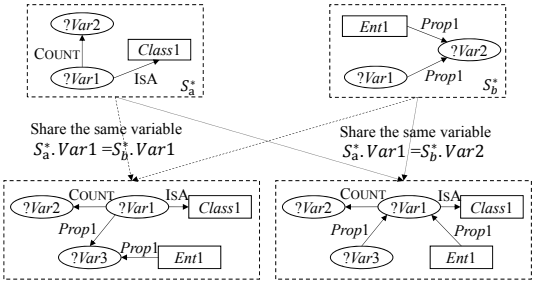

Figure 6: Merge results for two query substructures

(i) The merged results should be connected;

(ii) The merged results have $\leq \tau$ triples;

(iii) The merged results have $\leq \delta$ aggregations;

An example for merging two query substructures is shown in Figure 6.

## 4 Experiments and Results

In this section, we introduce the query generation datasets and state-of-the-art systems that we compare. We first show the end-to-end results of the query generation task, and then perform detailed analysis to show the effectiveness of each module. Question sets, source code and experimental results are available online.[1]

### 4.1 Experimental Setup

**Datasets** We employed the same datasets as Singh et al.(2018) and Zafar et al.(2018): (i) the large-scale complex question answering dataset (**LC-QuAD**) (Trivedi et al., 2017), containing 3,253 questions with non-empty results on DBpedia (2016-04), and (ii) the fifth edition of question answering over linked data (**QALD-5**) dataset (Unger et al., 2015), containing 311 questions with non-empty results on DBpedia (2015-10). Both datasets are widely used in KBQA studies (Zou et al., 2014; Dubey et al., 2018), and have become benchmarks for some annual KBQA competitions[2][3]. We did not employ the WebQuestions (Berant et al., 2013) dataset, since approximately 85% of its questions are simple. Also, we did not employ the ComplexQuestions (Bao et al., 2016) and ComplexWebQuestions (Talmor and Berant, 2018) dataset, since the existing works on these datasets have not reported the formal query generation result, and it is difficult to separate the formal query generation component from the end-to-end KBQA systems in these works.

[1] http://ws.nju.edu.cn/SubQG/
[2] http://lc-quad.sda.tech
[3] http://qald.aksw.org/index.php?q=5

Table 1: Datasets and implementation details

| | LC-QuAD | QALD-5 |
|---|---|---|
| No. of questions (complex) | 3,253 (2,249) | 311 (192) |
| No. of query structures | 35 | 52 |
| No. of freq. substructures | 37 | 10 |
| Avg. training time | 1,102s | 272s |
| Avg. prediction time | 0.291s | 0.122s |
| Avg. query generation time | 0.356s | 0.197s |

**Implementation details** All the experiments were carried out on a machine with an Intel Xeon E3-1225 3.2GHz processor, 32 GB of RAM, and an NVIDIA GTX1080Ti GPU. For the embedding layer, we used random embedding. For each dataset, we performed 5-fold cross-validation with the train set (70%), development set (10%), and test set (20%). The threshold $\gamma$ for frequent query substructures is set to 30, the maximum iteration number $K$ for merging is set to 2, $\theta$ in Algorithm 1 is set to 0.3, the maximum triple number $\tau$ for merged results is set to 5, and the maximum aggregation number $\delta$ is set to 2. Other detailed statistics are shown in Table 1.

### 4.2 End-to-End Results

We compared SubQG with several existing approaches. SINA (Shekarpour et al., 2015) and NLIWOD conduct query generation by predefined rules and existing templates. SQG (Zafar et al., 2018) firstly generates candidate queries by finding valid walks containing all of entities and properties mentioned in questions, and then ranks them based on Tree-LSTM similarity. CompQA (Luo et al., 2018) is a KBQA system which achieved state-of-the-art performance on WebQuesions and ComplexQuestions over Freebase. We re-implemented its query generation component for DBpedia, which generates candidate queries by staged query generation, and ranks them using an encode-and-compare network.

The average F1-scores for the end-to-end query generation task are reported in Table 2. All these results are based on the gold standard entity/relation linking result as input. Our approach SubQG outperformed all the comparative approaches on both datasets. Furthermore, as the results shown in Table 3, it gained a more significant improvement on complex questions compared with CompQA.

Both SINA and NLIWOD did not employ a query ranking mechanism, i.e., their accuracy and coverage are limited by the rules and templates. Although both CompQA and SQG have a strong ability of generating candidate queries, they perform not quite well in query ranking. According to our observation, the main reason is that these approaches tried to learn entire representations for questions with different query structures (from simple to complex) using a single network, thus, they may suffer from the lack of training data, especially for the questions with rarely appeared structures. As a contrast, our approach leveraged multiple networks to learn predictors for different query substructures, and ranked query structures using combinational function, which gained a better performance.

The results on QALD-5 dataset is not as high as the result on LC-QuAD. This is because QALD-5 contains 11% of very difficult questions, requiring complex filtering conditions such as REGEX and numerical comparison. These questions are currently beyond our approach's ability. Also, the size of training data is significant smaller.

[4] https://github.com/dice-group/NLIWOD

Table 2: Average F1-scores of query generation

| | LC-QuAD | QALD-5 |
|---|---|---|
| Sina (Shekarpour et al., 2015) | 0.24 † | 0.39 † |
| NLIWOD [4] | 0.48 † | 0.49 † |
| SQG (Zafar et al., 2018) | 0.75 † | - |
| CompQA (Luo et al., 2018) | $0.772_{\pm 0.014}$ | $0.511_{\pm 0.043}$ |
| SubQG (our approach) | $\mathbf{0.846}_{\pm 0.016}$ | $\mathbf{0.624}_{\pm 0.030}$ |

† indicates results taken from Singh et al. (2018) and SQG.

Table 3: Average F1-scores for complex questions

| | LC-QuAD | QALD-5 |
|---|---|---|
| CompQA | $0.673_{\pm 0.009}$ | $0.260_{\pm 0.082}$ |
| SubQG | $\mathbf{0.779}_{\pm 0.017}$ | $\mathbf{0.392}_{\pm 0.156}$ |

### 4.3 Detailed Analysis

#### 4.3.1 Ablation Study

We compared the following settings of SubQG:

**Rank w/o substructures.** We replaced the query substructure prediction and query structure ranking module, by choosing an existing query structure in the training data for the input question, using a BiLSTM multiple classification network.

**Rank w/ substructures** We removed the merging module described in Section 3.4. This setting assumes that the appropriate query structure for an input question exists in the training data.

**Merge query substructures** This setting ignored existing query structures in the training data, and only considered the merged results of query substructures.

As the results shown in Table 4, the full ver-

Table 4: Average F1-scores for different settings

| | LC-QuAD | QALD-5 |
|---|---|---|
| SubQG | $\mathbf{0.846}_{\pm 0.016}$ | $\mathbf{0.624}_{\pm 0.030}$ |
| Rank w/o substructures | $0.756_{\pm 0.012}$ | $0.383_{\pm 0.024}$ |
| Rank w/ substructures | $0.841_{\pm 0.014}$ | $0.614_{\pm 0.036}$ |
| Merge query substructures | $0.679_{\pm 0.020}$ | $0.454_{\pm 0.055}$ |

Table 5: Accuracy of query substructure prediction

| | LC-QuAD | QALD-5 |
|---|---|---|
| BiLSTM w/ attention | $\mathbf{0.929}_{\pm 0.002}$ | $0.816_{\pm 0.010}$ |
| BiLSTM w/o attention | $0.898_{\pm 0.004}$ | $0.781_{\pm 0.009}$ |
| BiLSTM w/ attention + POS | $0.925_{\pm 0.004}$ | $\mathbf{0.818}_{\pm 0.007}$ |
| CNN in (Yih et al., 2015) | $0.856_{\pm 0.006}$ | $0.740_{\pm 0.010}$ |

sion of SubQG achieved the best results on both datasets. *Rank w/o substructures* gained a comparatively low performance, especially when there is inadequate training data (on QALD-5). Compared with *Rank w/ substructures*, SubQG gained a further improvement, which indicates that the merging method successfully handled questions with unseen query structures.

Table 5 shows the accuracy of some alternative networks for query substructure prediction (Section 3.2). By removing the attention mechanism (replaced by unweighted average), the accuracy declined approximately 3%. Adding additional part of speech tag sequence of the input question gained no significant improvement. We also tried to replace the attention based BiLSTM with the network in (Yih et al., 2015), which encodes questions with a convolutional layer followed by a max pooling layer. This approach did not perform well since it cannot capture long-term dependencies.

#### 4.3.2 Results with Noisy Linking

We simulated the real KBQA environment by considering noisy entity/relation linking results. We firstly mixed the correct linking result for each mention with the top-5 candidates generated from EARL (Dubey et al., 2018), which is a joint entity/relation linking system with state-of-the-art performance on LC-QuAD. The result is shown in the second row of Table 6. Although the precision for first output declined 11.4%, in 85% cases we still can generate correct answer in top-5. This is because SubQG ranked query structures first and considered linking results in the last step. Many error linking results can be filtered out by the empty query check or domain/range check.

We also test the performance of our approach only using the EARL linking results. The performance dropped dramatically in comparison to the first two rows. The main reason is that, for

Table 6: Average Precision@$k$ scores of query generation on LC-QuAD with noisy linking

| | Precision@1 | Precision@5 |
|---|---|---|
| Gold standard | $0.842_{\pm 0.017}$ | $0.886_{\pm 0.014}$ |
| Top-5 EARL + gold standard | $0.728_{\pm 0.011}$ | $0.850_{\pm 0.009}$ |
| Top-5 EARL | $0.126_{\pm 0.012}$ | $0.146_{\pm 0.010}$ |

82.8% of the questions, EARL provided partially correct results. If we consider the remaining questions, our system again have 73.2% and 84.8% of correctly-generated queries in top-1 and top-5 output, respectively.

#### 4.3.3 Results on Varied Sizes of Training Data

We tested the performance of SubQG with different sizes of training data. The results on LC-QuAD dataset are shown in Figure 7. With more training data, our query substructure based approaches obtained stable improvements on both precision and recall. Although the merging module impaired the overall precision a little bit, it shows a bigger improvement on recall, especially when there is very few training data. Generally speaking, equipped with the merging module, our substructure based query generation approach showed the best performance.

#### 4.3.4 Error Analysis

We analyzed 100 randomly sampled questions that SubQG did not return correct answers. The major causes of errors are summarized as follows:

**Query structure errors (71%)** occurred due to multiple reasons. Firstly, 21% of error cases have entity mentions that are not correctly detected before query substructure prediction, which highly influenced the prediction result. Secondly, in 39% of the cases a part of substructure predictors provided wrong prediction, which led to wrong structure ranking results. Finally, in the remaining 11% of the cases the correct query structure did not appear in the training data, and they cannot be generated by merging substructures.

**Grounding errors (29%)** occurred when SubQG generated wrong queries with correct query structures. For example, for the question *"Was Kevin Rudd the prime minister of Julia Gillard"*, SubQG cannot distinguish $\langle JG, primeMinister, KR\rangle$ from $\langle KR, primeMinister, JG\rangle$, since both triples exist in DBpedia. We believe that extra training data are required for fixing this problem.

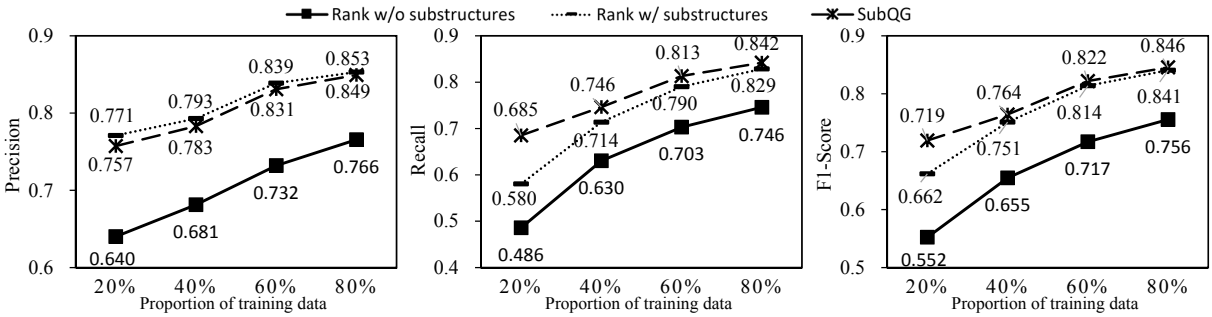

Figure 7: Precision, recall and F1-score with varied proportions of training data

## 5 Related Work

Alongside with entity and relation linking, existing KBQA systems often leverage formal query generation for complex question answering (Bao et al., 2016; Trivedi et al., 2017). Based on our investigation, the query generation approaches can be roughly divided into two kinds: *template*-based and *semantic parsing*-based.

Template-based approaches transform the input question into a formal query by employing pre-collected query templates. Cui et al.(2017) collect different natural language expressions for the same query intention from question-answer pairs. Singh et al.(2018) re-implement and evaluate the query generation module in NLIWOD, which selects an existing template by some simple features such as the number of entities and relations in the input question. Recently, several query decomposition methods are studied to enlarge the coverage of the templates. Abujabal et al.(2017) present a KBQA system named QUINT, which collects query templates for specific dependency structures from question-answer pairs. Furthermore, it rewrites the dependency parsing results for questions with conjunctions, and then performs sub-question answering and answer stitching. Zheng et al.(2018) decompose questions by using a huge number of triple-level templates extracted by distant supervision. Compared with these approaches, our approach predicts all kinds of query substructures (usually 1 to 4 triples) contained in the question, making full use of the training data. Also, our merging method can handle questions with unseen query structures, having a larger coverage and a more stable performance.

Semantic parsing-based approaches translate questions into formal queries using bottom up parsing (Berant et al., 2013) or staged query graph generation (Yih et al., 2015). gAnswer (Zou et al., 2014; Hu et al., 2018) builds up semantic query graph for question analysis and utilize subgraph matching for disambiguation. Recent studies combine parsing based approaches with neural networks, to enhance the ability for structure disambiguation. Bao et al.(2016), Luo et al.(2018) and Zafar et al.(2018) build query graphs by staged query generation, and follow an encode-and-compare framework to rank candidate queries with neural networks. These approaches try to learn entire representations for questions with different query structures by using a single network. Thus, they may suffer from the lack of training data, especially for questions with rarely appeared structures. By contrast, our approach utilizes multiple networks to learn predictors for different query substructures, which can gain a stable performance with limited training data. Also, our approach does not require manually-written rules, and performs stably with noisy linking results.

## 6 Conclusion

In this paper, we introduced SubQG, a formal query generation approach based on frequent query substructures. SubQG firstly utilizes multiple neural networks to predict query substructures contained in the question, and then ranks existing query structures using a combinational function. Moreover, SubQG merges query substructures to build new query structures for questions without appropriate query structures in the training data. Our experiments showed that SubQG achieved superior results than the existing approaches, especially for complex questions.

In future work, we plan to add support for other complex questions whose queries require UNION, GROUP BY, or numerical comparison. Also, we are interested in mining natural language expressions for each query substructures, which may help current parsing approaches.

## 7 Acknowledgments

This work is supported by the National Natural Science Foundation of China (Nos. 61772264 and 61872172). We would like to thank Yao Zhao for his help in preparing evaluation.

## References

Abdalghani Abujabal, Mohamed Yahya, Mirek Riedewald, and Gerhard Weikum. 2017. Automated template generation for question answering over knowledge graphs. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017*, pages 1191–1200.

Jun-Wei Bao, Nan Duan, Zhao Yan, Ming Zhou, and Tiejun Zhao. 2016. Constraint-based question answering with knowledge graph. In *Proceedings of the 26th International Conference on Computational Linguistics, COLING 2016*, pages 2503–2514.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013*, pages 1533–1544.

Wanyun Cui, Yanghua Xiao, Haixun Wang, Yangqiu Song, Seung-won Hwang, and Wei Wang. 2017. KBQA: learning question answering over QA corpora and knowledge bases. *Proceedings of the VLDB Endowment*, 10(5):565–576.

Mohnish Dubey, Debayan Banerjee, Debanjan Chaudhuri, and Jens Lehmann. 2018. EARL: joint entity and relation linking for question answering over knowledge graphs. In *Proceedings of the 17th International Semantic Web Conference, ISWC 2018, Part I*, pages 108–126.

Sen Hu, Lei Zou, Jeffrey Xu Yu, Haixun Wang, and Dongyan Zhao. 2018. Answering natural language questions by subgraph matching over knowledge graphs. *IEEE Transactions on Knowledge and Data Engineering*, 30(5):824–837.

Kangqi Luo, Fengli Lin, Xusheng Luo, and Kenny Q. Zhu. 2018. Knowledge base question answering via encoding of complex query graphs. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018*, pages 2185–2194.

Colin Raffel and Daniel P. W. Ellis. 2015. Feed-forward networks with attention can solve some long-term memory problems. *CoRR*, abs/1512.08756.

Saeedeh Shekarpour, Edgard Marx, Axel-Cyrille Ngonga Ngomo, and Sören Auer. 2015. SINA: semantic interpretation of user queries for question answering on interlinked data. *Journal of Web Semantics*, 30:39–51.

Kuldeep Singh, Arun Sethupat Radhakrishna, Andreas Both, Saeedeh Shekarpour, Ioanna Lytra, Ricardo Usbeck, Akhilesh Vyas, Akmal Khikmatullaev, Dharmen Punjani, Christoph Lange, Maria-Esther Vidal, Jens Lehmann, and Sören Auer. 2018. Why reinvent the wheel: Let's build question answering systems together. In *Proceedings of the 27th International Conference on World Wide Web, TheWebConf 2018*, pages 1247–1256.

Alon Talmor and Jonathan Berant. 2018. The Web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018*, pages 641–651.

Priyansh Trivedi, Gaurav Maheshwari, Mohnish Dubey, and Jens Lehmann. 2017. LC-QuAD: A corpus for complex question answering over knowledge graphs. In *Proceedings of the 16th International Semantic Web Conference, ISWC 2017, Part II*, pages 210–218.

Christina Unger, Corina Forascu, Vanessa López, Axel-Cyrille Ngonga Ngomo, Elena Cabrio, Philipp Cimiano, and Sebastian Walter. 2015. Question answering over linked data (QALD-5). In *Working Notes of Conference and Labs of the Evaluation Forum, CLEF 2015*.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics, ACL 2015*, pages 1321–1331.

Hamid Zafar, Giulio Napolitano, and Jens Lehmann. 2018. Formal query generation for question answering over knowledge bases. In *Proceedings of the 15th Extended Semantic Web Conference, ESWC 2018*, pages 714–728.

Weiguo Zheng, Jeffrey Xu Yu, Lei Zou, and Hong Cheng. 2018. Question answering over knowledge graphs: Question understanding via template decomposition. *Proceedings of the VLDB Endowment*, 11(11):1373–1386.

Lei Zou, Ruizhe Huang, Haixun Wang, Jeffrey Xu Yu, Wenqiang He, and Dongyan Zhao. 2014. Natural language question answering over RDF: a graph data driven approach. In *Proceedings of the International Conference on Management of Data, SIGMOD 2014*, pages 313–324.

# Incorporating Graph Attention Mechanism into Knowledge Graph Reasoning Based on Deep Reinforcement Learning

**Heng Wang**[⋆] **Shuangyin Li**[*] **Rong Pan**[†] **Mingzhi Mao**[†]
[⋆]Tencent, China
[†]School of Data and Computer Science, Sun Yat-sen University, China
[*]School of Computer Science, South China Normal University, China
jimmyhwang@tencent.com
{panr@mail,mcsmmz@mail}.sysu.edu.cn
shuangyinli@m.scnu.edu.cn

## Abstract

Knowledge Graph (KG) reasoning aims at finding reasoning paths for relations, in order to solve the problem of incompleteness in KG. Many previous path-based methods like PRA and DeepPath suffer from lacking memory components, or stuck in training. Therefore, their performances always rely on well-pretraining. In this paper, we present a deep reinforcement learning based model named by **AttnPath**, which incorporates LSTM and Graph Attention Mechanism as the memory components. We define two metrics, Mean Selection Rate (**MSR**) and Mean Replacement Rate (**MRR**), to quantitatively measure how difficult it is to learn the query relations, and take advantages of them to fine-tune the model under the framework of reinforcement learning. Meanwhile, a novel mechanism of reinforcement learning is proposed by forcing an agent to walk forward every step to avoid the agent stalling at the same entity node constantly. Based on this operation, the proposed model not only can get rid of the pretraining process, but also achieves state-of-the-art performance comparing with the other models. We test our model on FB15K-237 and NELL-995 datasets with different tasks. Extensive experiments show that our model is effective and competitive with many current state-of-the-art methods, and also performs well in practice.

## 1 Introduction

Knowledge Graphs (KGs), such as NELL (Carlson et al., 2010), Freebase (Bollacker et al., 2008) and WordNet (Miller, 1995) play an increasingly critical role in many downstream NLP applications, e.g., question answering (Dubey et al., 2018), information retrieval (Liu et al., 2018), personalized recommendations (Wang et al., 2019), etc. However, KGs are always incomplete, which would affect many downstream tasks. Many links could miss among the entities in a KG. Thus, it is an important and challenge task on how to complete the KG by predicting the missing links between the entities through the method based on reasoning. For instance, if $athletePlaysForTeam(X, Y)$ and $teamPlaysInLeague(Y, Z)$ both exist in the KG, then we can infer that $athletePlaysInLeague(X, Z)$, i.e, filling the missing edge $athletePlaysInLeague$ between $X$ and $Z$.

There are mainly three ways to accomplish this task, such as Rule-Based (Wang and Cohen, 2016; Yang et al., 2017), Embedding-Based (Bordes et al., 2013; Lin et al., 2015) and Path-Based (Lao et al., 2011). Meanwhile, it provides a new perspective to bring Deep Reinforcement Learning (DRL) into the task of predicting the missing links, such as DeepPath (Xiong et al., 2017), a type of path-based method. DeepPath is the first work which incorporates DRL into KG reasoning. It achieves significant improvements compared with PRA, but still has several drawbacks. First, it lacks memory components, resulting in requiring **pretraining**. The operation of pretraining is demanded to provide many known (or existed) paths to the model training. This brute-force operation may make the model prone to overfit on the given paths from pretraining. Second, it's inappropriate to set the same hyperparameter for different relations in a KG when training, which ignores the diversity of connections among the entities. Last, when the agent selects an invalid path, it will stop and reselect, which leads to select this invalid path constantly and finally stuck in one node.

Thus, in this paper, we present a novel deep reinforcement learning model and an algorithm, which aims at tackling the drawbacks mentioned above. The proposed model also belongs to the path-based framework. Our contributions can be summarized as follows:

- We propose **AttnPath**, a model which incor-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2623–2631, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

porates **LSTM** and **graph attention** as memory components, and is no longer subject to pretraining.

- Two metrics are defined (**MSR** and **MRR**), to quantitatively measure the difficulty to learn a relation's replaceable paths, which are used to fine-tune the model.

- A novel mechanism of reinforcement learning is proposed by forcing an agent to walk forward every step to avoid the agent stalling at the same entity node constantly.

We test **AttnPath** on FB15K-237 and NELL-995 datasets with two downstream tasks: fact prediction and link prediction. We also test on the success rate in finding paths and show the effectiveness of the Graph Attention Mechanism in the experiments.

## 2 Related Works

To date, there are many works proposed to solve the problem of KG incompleteness. Rule-Based methods, like ProPPR (Wang and Cohen, 2016) and Neural LP (Yang et al., 2017), generate reasoning rules manually or by mathematical logic rules, and then apply them to fill missing links based on existing triples. Although this type of methods has a solid mathematical background, they are hard to scale to large KGs, since they directly operate on symbols, while the number of possible reasoning paths is exponential to the number of the entities. Embedding-Based methods, like TransE (Bordes et al., 2013) and TransR (Lin et al., 2015), map entities and relations into a low-dimensional and continuous vector space, which captures the feature of distance between entities and relations. Then, they judge whether the query relation exists by comparing the distance between the two trained entities' embeddings and the query relation's embedding. This type of methods requires all triples in the KG to participate in training, and are only suitable for single-hop reasoning.

Path-Based, like PRA (Lao et al., 2011) and DeepPath (Xiong et al., 2017), train an agent to navigate on a KG, find replaceable paths for a certain relation, and then, use them as features to the downstream tasks. Path Ranking Algorithm (PRA) is the first path-based reasoning method. Neelakantan et al. develop a compositional model based on RNN, which composes the implications of a path and reasons about conjunctions of multi-hop relations non-atomically (Neelakantan et al., 2015). Guu et al. propose a soft edge traversal operator, which can be recursively applied to predict paths and reduce cascaded propagation errors faced by single-hop KG completion methods like TransE and TransR (Guu et al., 2015). Toutanova et al. propose a dynamic programming algorithm which incorporates all relations' paths of bounded length in a KG, and models both relations and intermediate nodes in the compositional path representations (Toutanova et al., 2016). Such representations can aid in generating more high-quality reasoning paths.

Das et al. improve DeepPath (Xiong et al., 2017) to MINERVA (Das et al., 2018), which views KG from QA's perspective. It gets rid of pretraining, introduces LSTM to memorize paths traversed before, and trains an agent to circulate on a certain entity if it believes this entity is the right answer. Lin et al. improve these two methods by introducing reward shaping and action dropout (Lin et al., 2018). Reward shaping replaces fixed penalty for useless selection with a dynamic penalty, which can either base on margin-based pretrained embeddings like TransE, or probability-based embeddings like ConvE (Dettmers et al., 2018). While action dropout randomly masks a certain proportion of valid actions, in order to reduce irrelevant paths to the query relation. DIVA (Chen et al., 2018) regards paths as latent variables, and relations as observed variables, so as to build a variational inference model to accomplish the KG reasoning task. It also uses beam search to broaden the search horizon. M-Walk (Shen et al., 2018) utilizes another RL algorithm called Monte Carlo Tree Search (MCTS), to tackle the problem of sparse rewards. The attention mechanism is first introduced into multi-hop KG reasoning by (Wang et al., 2018). However, it only computes the attention weights of the query's embedding and all found paths' embeddings. They are used to help to judge whether the answer found by the vanilla model is true.

## 3 AttnPath: Incorporating Memory Components

In this section, we will introduce the details of the proposed AttnPath. We will also show the definition of mean selection rate (MSR) and mean replacement rate (MRR), and also how they are used to fine-tune the model for different query relations.

### 3.1 RL framework for KG reasoning

Since we use Reinforcement Learning (RL) as the training algorithm of a sequential decision model, we first introduce basic elements of the RL framework in the KG reasoning, including environment, state, action, and reward.

**Environment:** In this task, the environment refers to the whole KG, excluding the query relation and its inverse. The environment retains consistent in the whole training process.

**State:** The state of an agent is concatenated with three parts: the embedding part, the LSTM part, and the graph attention part. We will show the computation of the LSTM part and the graph attention part in the next section, and introduce the embedding part first.

Following (Xiong et al., 2017), the embedding part $\mathbf{m}_t$ is concatenated with two subparts. The first one is $\mathbf{e}_t$, which is the embedding of the current entity node. The other one is $\mathbf{e}_{target} - \mathbf{e}_t$, denoting the distance between the tail entity node and the current node. Different from DeepPath which uses TransE (Bordes et al., 2013) as pretrained embeddings, we take advantage of TransD (Ji et al., 2015), which is an improvement of TransE and also a common-used benchmark. Under TransD, for a query relation, we project all the entities onto the vector space of this query relation. Then, an entity's projected embedding $\mathbf{e}_\perp$ becomes

$$\mathbf{e}_\perp = (\mathbf{r_p}\mathbf{e_p}^\top + \mathbf{I})\mathbf{e}, \tag{1}$$

where $\mathbf{p}$ indicates the projection vectors. So $\mathbf{m}_t$ should be $[\mathbf{e}_{t\perp}; \mathbf{e}_{target\perp} - \mathbf{e}_{t\perp}]$.

**Action:** For the KG reasoning task, an action refers to an agent choosing a relation path to step forward. Based on the framework of DRL, it chooses the relation according to the probabilities obtained by the model. Actions are either valid or invalid. Valid action means that there is an output relation connected with the current entity, while invalid action denotes that there is no relation.

**Reward:** Reward is a feedback to the agent according to whether the action is valid, and whether a series of actions can lead to ground truth tail entities in a specified number of times. We adopt reward shaping trick proposed by (Lin et al., 2018). For the invalid actions, the reward is -1. For the actions which don't lead to ground truth, we choose the output of ConvE (Dettmers et al., 2018) as the reward. Since ConvE outputs probabilities, which are in the range of $(0, 1)$, we resort a logarithm operator to enlarge the range of this reward and improve discrimination. For the actions which lead to ground truth, i.e, a successful episode, the reward is the weighted sum of the global accuracy, the path efficiency, and the path diversity. The global accuracy is set to 1 by convention, and the path efficiency is the reciprocal of the path length, because we encourage the agent to step as fewer times as possible. The path diversity is defined as

$$r_{\text{DIVERSITY}} = -\frac{1}{|F|}\sum_{i=1}^{|F|}\cos(\mathbf{p}, \mathbf{p}_i), \tag{2}$$

where $|F|$ is the number of found paths, and $\mathbf{p}$ is the path embedding, simply the sum of all relations' embeddings in the path. The above definition guarantees that the reward of the valid actions are always larger than the invalid actions, and the reward of the successful episodes are always larger than the unsuccessful ones.

### 3.2 LSTM and Graph Attention as Memory Components

In our model, we utilize a three-layer LSTM, enabling the agent to memorize and learn from the actions taken before. Denote the hidden state of LSTM at step $t$ by $\mathbf{h}_t$, and the initial hidden state $\mathbf{h}_0$ by $\mathbf{0}$. Then we obtain

$$\mathbf{h}_t = \text{LSTM}(\mathbf{h}_{t-1}, \mathbf{m}_t), \tag{3}$$

where $\mathbf{m}_t$ is defined in Eq. (1). This is the LSTM part of the state described above.

Typically, an entity has several different aspects, for example, a football player may be connected with professional relations like $playsForTeam$ or $playsInLeague$, and also family relations like $spouse$ or $father$. For different query relations, it's better for the agent to focus more on relations and neighbors which are highly related to the query relations. So we introduce Graph Attention mechanism (GAT) into our model, which is proposed by (Velickovic et al., 2018).

GAT is indeed self-attention on the entity nodes. We use a single-layer feedforward neural network to calculate attention weights, with a linear transformation matrix $\mathbf{W}$ and a weight vector $\vec{\mathbf{a}}$ shared across all entities. LeakyReLU with negative input slope $\alpha = 0.2$ is chosen as nonlinearity. So the attention weight from entity $i$ to entity $j$ is calculated as

$$a_{ij} = \text{LeakyReLU}(\vec{\mathbf{a}}^\top[\mathbf{W}\mathbf{e}_{i\perp}; \mathbf{W}\mathbf{e}_{j\perp}]). \tag{4}$$

For entity $i$, we only compute attention weights to all its direct-connected neighbors, and normalize them with SoftMax. So the normalized attention weight is

$$\alpha_{ij} = \frac{\exp(a_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(a_{ik})}. \quad (5)$$

Now we can compute the graph attention part for the state, which is simply the weighted sum of all neighbors' embedding on the attention space with

$$\mathbf{a}_i = \sum_{k \in \mathcal{N}_i} \alpha_{ik} \mathbf{W} \mathbf{e}_k. \quad (6)$$

Thus, the state vector $\mathbf{s}_{i,t}$ of entity $i$ in time $t$ is

$$\mathbf{s}_{i,t} = [\mathbf{m}_{i,t}; \mathbf{h}_t; \mathbf{a}_i], \quad (7)$$

which in turn inputs a three-layer feedforward neural network, whose final output is a Softmax probability with length equal to the number of all relations in KG. The agent selects an action and obtains a reward. After it successfully reaches the tail entity or doesn't reach in a specified number of times, the rewards of the whole episode are used to update all parameters. The optimization is done using the REINFORCE (Williams, 1992) algorithm and updates $\theta$ with the following stochastic gradient:

$$\nabla_\theta J(\theta) \approx \nabla_\theta \sum_{t=1}^{T} R(\mathbf{s}_T | e_s, r) \log \pi_\theta(a_t | s_t), \quad (8)$$

where $e_s$ is head entity, $r$ the query relation, and $\pi_\theta(a_t|s_t)$ the probability of all relations. Figure 1 shows our **AttnPath** model.

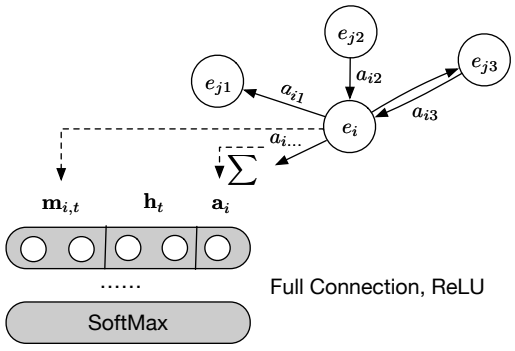


Figure 1: AttnPath: an RL framework composed with LSTM and graph attention for KG reasoning.

**Algorithm 1** Training Algorithm of the agent for AttnPath

**for** $episode \leftarrow 1$ **to** $N$ **do**
  Initialize LSTM's hidden state $\mathbf{h}_0$ to $\mathbf{0}$
  Initialize state vector $\mathbf{s}_0$
  Initialize $num\_steps$ to 0
  **while** $num\_steps < max\_steps$ **do**
    Randomly sample action $a \sim \pi_\theta(a_t|s_t)$
    **if** Action invalid (leads to nothing) **then**
      Add $< s_t, a >$ to $\mathcal{M}_{neg}$
      Normalize $\pi_\theta(a_t|s_t)$ on valid actions and sample $a_f$
      Add $< s_t, a_f >$ to $\mathcal{M}_{pos}$
    **else**
      Add $< s_t, a >$ to $\mathcal{M}_{pos}$
    **end if**
    Increment $num\_steps$
    **if** $success$ or $num\_steps = max\_steps$ **then**
      break
    **end if**
  **end while**
  Update $\theta$ using $g \propto \sum_{\mathcal{M}_{neg}} \log\pi(a_t|s_t;\theta)(-1)$
  **if** $success$ **then**
    $R_{total} \leftarrow \lambda_1 r_{\text{GLOBAL}} + \lambda_2 r_{\text{EFFICIENCY}} + \lambda_3 r_{\text{DIVERSITY}}$
    Update $\theta$ using $g \propto \sum_{\mathcal{M}_{pos}} \log\pi(a_t|s_t;\theta) R_{total}$
  **else**
    Compute shaped reward $R_{shaping}$ for $\mathcal{M}_{pos}$
    Update $\theta$ using $g \propto \sum_{\mathcal{M}_{pos}} \log\pi(a_t|s_t;\theta) R_{shaping}$
  **end if**
**end for**

### 3.3 Mean Selection / Replacement Rate

For different query relations, it is required to train different models for each of them. While, in practice, the difficulty values of each relation are quite different. Some relation may have more replacement relations, indicating that an agent can easily choose a replacement path walking from head entity to tail. So we invent two metrics, **Mean Selection Rate (MSR)** and **Mean Replacement Rate (MRR)** here, to quantitatively measure the difficulty value of each relation.

Denote all triples related to relation $r$ by a set

of $\mathcal{T}_r = \{(h, r_o, t)|r_o = r\}$. The selection rate for relation $r$ with regard to triple $(h, r, t)$ is defined as

$$SR((h, r, t)) = \frac{|\{(h_o, r_o, t_o)|h_o = h, r_o = r\}|}{|\{(h_o, r_o, t_o)|h_o = h\}|}, \quad (9)$$

i.e, the proportion of relation $r$ occupying $h$'s outgoing paths.

Thus, MSR is the average of SR on $\mathcal{T}_r$:

$$MSR(r) = \sum_{(h_o,r,t_o)\in\mathcal{T}_r} SR(h_o, r, t_o)/|\mathcal{T}_r|. \quad (10)$$

Lower MSR denotes that it's more difficult to learn $r$, because the entities connected with relation $r$ may have more aspects.

The replacement rate for relation $r$ with regard to triple $(h, r, t)$ is defined as

$$RR((h, r, t)) = \frac{|\{(h_o,r_o,t_o)|h_o=h,r_o\neq r,t_o=t\}|}{|\{(h_o,r_o,t_o)|h_o=h,t_o=t\}|}, \quad (11)$$

i.e, the proportion of relations that directly connects $h$ and $t$, except relation $r$.

Similarly, MRR is the average of RR on $\mathcal{T}_r$:

$$MRR(r) = \sum_{(h_o,r,t_o)\in\mathcal{T}_r} RR(h_o, r, t_o)/|\mathcal{T}_r|. \quad (12)$$

Higher MRR denotes a relation may have more replacement relations, so it's easier to learn since an agent can directly choose an alternative relation to reach the destination.

In our model, we have three ways to prevent overfitting: L2 regularization, dropout, and action dropout. However, for the relations which are easier to learn (high MSR and MRR), we wish to impose more regularization to encourage the agent to find more diverse paths, without overfitting on the immediate success. Otherwise, for the relations which are more difficult to learn (low MSR and MRR), we'd better focus on the success rate of finding a path, so we should impose less regularization.

For simplicity, we use exponential to compute the difficulty coefficient of relation $r$. It is defined as $\exp(MSR(r) + MRR(r))$ and multiplied with the base rate of three regularization methods, respectively. The base rate of regularization methods is KG-based, shared across all relations in the same KG.

Table 1: Statistics of the datasets. #Rel. and #Tri. doesn't include the inverse triples.

| Dataset | #Ent. | #Rel. | #Tri. | #Task |
|---|---|---|---|---|
| FB15K-237 | 14,505 | 237 | 310,116 | 20 |
| NELL-995 | 75,492 | 200 | 154,213 | 12 |

### 3.4 Overall Training Algorithm

Based on the proposed model, we present a novel training algorithm shown in Algorithm 1.

One of our contributions in our algorithm is, when the agent selects an invalid path, our model not only penalizes it, but also forces it to select a valid relation to step forward. The probabilities from the neural network are normalized across all valid relations, which in turn act the probabilities of the forced action.

After the initializations, Line 6 samples an action according to the output of the network. When the agent selects an invalid action, Line 7 $\sim$ 10 is executed, and Line 9 $\sim$ 10 forces the agent to step forward. When the agent selects a valid action, Line 12 is executed. Lines 19, 22 and 25 update parameters for the invalid actions, the valid actions in a successful episode, and the valid actions in an unsuccessful episode, respectively, with reward $-1$, $R_{total}$ and $R_{shaping}$.

## 4 Experiments

In this section, we will perform extensive experiments to validate the effectiveness of our proposed AttnPath. For each task, we will mainly focus on three quantitative metrics: success rate (**SR**) in finding paths, MAP of fact prediction (**FP**), and MAP of link prediction (**LP**). We will also demonstrate some reasoning paths and triples to show that graph attention is effective in finding more high-quality paths and mining which aspect of the entity is important in a specific task.

### 4.1 Datasets and Settings

Two datasets, FB15K-237 (Toutanova et al., 2015) and NELL-995 (Xiong et al., 2017), are used in our experiments. Statistics of these two datasets are displayed in Table 1. Following the previous works, for each triple $(h, r, t)$, we add its inverse triple $(t, r^{-1}, h)$, to allow the agent to step backward.

We summary the hyperparameters involved in our experiments here. The pretrained embedding dimension is set to 100. The LSTM hidden dimension is set to 200. The attention dimension

is set to 100. Thus, $\mathbf{s}$ is a 500-dimension vector by concatenating the above three vectors with two times of the pretrained embedding dimension. $\lambda_1$ is 0.1, $\lambda_2$ is 0.8, $\lambda_3$ is 0.1. For FB15K-237 dataset, we set base L2 regularization, Dropout rate and action dropout rate to 0.005, 0.15, and 0.15, respectively. Also, for NELL-995, we set them to 0.005, 0.1 and 0.1, respectively. We choose Adam (Kingma and Ba, 2015) as the optimizer with different learning rates of 0.001, $\beta_1$ 0.9 and $\beta_2$ 0.999. For each task, we train 500 episodes, and for each episode, $max\_steps$ is set to 50.

We verify the learned paths for each triple in FP and LP tasks when training, by the BFS-based method (Xiong et al., 2017).

### 4.2 Success Rate in Finding Paths

Our model is ignorant of the environment and triples at first, since it doesn't rely on pretraining. Thus, we record total SR, and SR in the recent 10 episodes to validate the agent's ability to learn the paths. For the tasks with less than 500 training samples, the samples are iterated first, then randomly sampled to reach 500 episodes. For the tasks with more than 500 training samples, we randomly select 500 out for training.

We select two relations, $athletePlaysInLeague$ and $organizationHeadquarteredInCity$ from NELL-995, to investigate its total SR and SR in recent 10 episodes. The former has relatively lower MSR and MRR, and the latter has higher. Figure 2 shows the results. It can be discovered that DeepPath outperforms our method at first, however, after 50 $\sim$ 150 epochs, our models surpass it. From SR-10 of AttnPath Force, we find that the initial SR is approximate to MRR, because the model knows nothing at all at first, so it selects a path randomly. As the training proceeds, the performance gains steady improvement. Similar results can also be found in other relations on FB15K-237.

### 4.3 Fact Prediction

Fact prediction (**FP**) aims at predicting whether an unknown fact is true or false. The proportion of positive triples and negative triples is about 1 : 10. For each relation, we use all found paths and reciprocal of length as weight, to accumulate the score of each triple based on whether the path is valid between $h$ and $t$. Scores are ranked across all test set and Mean Average Precision (MAP) is used as the evaluation metric. The results are shown in Table 2. It can be seen that AttnPath outperforms TransE / R and DeepPath significantly. AttnPath MS / RR, which uses MSR and MRR to fine-tune the hyperparameters, also gains performance improvement. AttnPath Force is also effective. By forcing the agent to walk forward every step, it improves the SR of finding a path, which in turn enriches the feature of the downstream tasks. This is especially important for the relations, which lack direct-connected replacement paths and require path with long-term dependency. In fact, our methods achieve **SOTA** results on both the two datasets.

Table 2: Fact prediction MAP (%). Results of TransE / H / R / D are cited from (Xiong et al., 2017). DeepPath is retrained with TransD for a fair comparison, which performs slightly better than (Xiong et al., 2017) based on TransE.

| Method | FB15K-237 | NELL-995 |
|---|---|---|
| TransE | 27.7 | 38.3 |
| TransH | 30.9 | 38.9 |
| TransR | 30.2 | 40.6 |
| TransD | 30.3 | 41.3 |
| DeepPath TransD | 31.3 | 53.5 |
| AttnPath | 31.5 | 59.8 |
| AttnPath MS / RR | 34.6 | 65.4 |
| AttnPath Force | **37.9** | **69.3** |

### 4.4 Link Prediction

Link prediction (**LP**) aims at predicting the target entities. For each $(h, r)$ pair, there is a ground truth $t$ and about 10 generated false $t$. It is divided into a training set and a test set. We use the found paths as binary features, and train a classification model on the training set, applying it on the test set. LP also uses MAP as the evaluation metric, and the detailed results are shown in Table 3 [1]. Alike FP, our model also attains better results in LP, and the **SOTA** results on FB15K-237 dataset[2].

However, we also notice that AttnPath doesn't attain the best result under a small part of query relations, even lower than TransE / R. By analyzing triples related to these relations, we found that: 1) they have more outgoing edges of the other relations pointing to the entities which are not the

[1] We omit to show detailed result of TransH / D here, while (Xiong et al., 2017) also doesn't give detailed result of it, and it is mentioned in this paper that vanilla DeepPath already outperforms main embedding-based methods, including TransE / H / R / D.

[2] DIVA (Chen et al., 2018) claims that it attains 88.6% MAP on NELL-995 dataset.

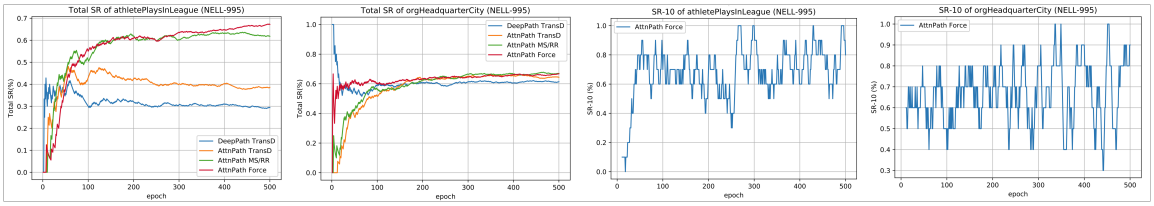


Figure 2: Total SR and SR-10 for two relations of NELL-995. DeepPath / AttnPath TransD means using TransD as the pretrained embeddings. AttnPath MS/RR is adding MSR and MRR to fine-tune hyperparameters. AttnPath Force is forcing the agent to move forward every step. These abbreviations are used throughout this section.

Table 3: Link Prediction MAP. Top half is **FB15K-237**, bottom half is **NELL-995**. TE, TR, DP, AP, APM and APF are abbreviations of TransE, TransR, DeepPath, AttnPath, AttnPath MS/RR, AttnPath Force, respectively. The result of TransE / R is cited from (Xiong et al., 2017). All the numbers are percentage (%).

| Tasks | MSR | MRR | TE | TR | DP | AP | APM | APF |
|---|---|---|---|---|---|---|---|---|
| filmDirector | 7.5 | 11.9 | 38.6 | 39.9 | 43.4 | 40.9 | 43.3 | **43.7** |
| filmLanguage | 3.7 | 0.1 | 64.2 | 64.1 | 70.1 | 56.6 | 70.9 | **71.8** |
| filmWrittenBy | 2.6 | 4.3 | 56.3 | **60.5** | 45.4 | 55.6 | 56.3 | 58.9 |
| capitalOf | 8.8 | 16.1 | 55.4 | 49.3 | 79.3 | 86.0 | 86.7 | **87.2** |
| musicianOrigin | 3.0 | 1.5 | 36.1 | 37.9 | 51.4 | 47.9 | 51.0 | **52.6** |
| organizationFounded | 4.2 | 1.3 | 39.0 | 33.9 | 31.8 | 31.1 | 34.9 | **46.8** |
| personNationality | 4.6 | 0.1 | 64.1 | 72.0 | 83.6 | 74.6 | 84.6 | **84.6** |
| birthPlace | 3.7 | 1.6 | 40.3 | 41.7 | **55.0** | 49.9 | 54.4 | 54.4 |
| teamSports | 6.7 | 0 | 89.6 | 78.4 | **91.6** | 70.2 | 91.2 | 91.3 |
| tvLanguage | 5.3 | 0.3 | 80.4 | 90.6 | 96.7 | 95.9 | 97.1 | **96.8** |
| ... | | | | | | | | |
| Overall | — | — | 53.2 | 54.0 | 63.5 | 58.5 | 65.3 | **66.1** |
| athleteHomeStadium | 32.4 | 0.0 | 71.8 | 72.2 | 89.3 | 89.3 | 88.7 | **89.4** |
| athletePlaysForTeam | 42.5 | 5.6 | 62.7 | 67.3 | 76.0 | 74.6 | 74.3 | **76.1** |
| athletePlaysInLeague | 34.9 | 0.3 | 77.3 | 91.2 | 96.2 | 95.6 | 95.1 | **96.5** |
| athletePlaysSport | 60.9 | 0 | 87.6 | 96.3 | 96.3 | 96.5 | 94.3 | **97.0** |
| orgHeadquarterCity | 42.6 | 25.0 | 62.0 | 65.7 | 92.8 | 92.9 | 93.2 | **94.1** |
| orgHiredPerson | 24.9 | 18.2 | 71.9 | 73.7 | 75.4 | 79.7 | 79.8 | **81.6** |
| bornLocation | 34.7 | 16.4 | 71.2 | **81.2** | 75.9 | 76.7 | 74.6 | 78.6 |
| personLeadsOrg | 32.6 | 36.9 | 75.1 | 77.2 | 79.6 | 80.9 | 81.0 | **82.8** |
| teamPlaysSport | 19.9 | 0 | 76.1 | 81.4 | 74.1 | 75.7 | 75.5 | **82.1** |
| worksFor | 45.6 | 27.7 | 67.7 | 69.2 | 71.2 | 71.3 | 71.8 | **77.5** |
| ... | | | | | | | | |
| Overall | — | — | 73.7 | 78.9 | 82.9 | 83.4 | 83.0 | **85.8** |

true tails, so MSR of these query relations are low. 2) Tail entities are only connected with edges of query relations and their inverse, while these edges are removed during training, so tail entities become isolated, without any possible replaceable paths. It will also lower MRR of these query relations. Take $birthPlace$ and $bornLocation$ as examples. If a person was born in a remote place, this place is difficult to be connected with the other entities, so it is prone to be isolated. However, such one-to-one relations are the strength of TransX methods.

## 4.5 Qualitative Analysis

### 4.5.1 Paths Found By DeepPath and AttnPath

We take $capitalOf$ from FB15K-237, and $athletePlaysInLeague$ from NELL-995, as examples, to analyze these paths found by DeepPath and AttnPath. Table 4 shows the top 5 frequent paths and their frequencies for all the methods. It shows that AttnPath is more capable of capturing long-term dependencies between relations, which is useful for relations lack of direct-connected replaceable paths. AttnPath can also find more important and concentrated paths, so the distribution of paths doesn't have a heavy long tail. In the training process, we also find that AttnPath is better at stepping backward when it enters a dead end.

### 4.5.2 Effectiveness of Graph Attention Mechanism

We sample several pairs of entity and relation, compute the entity's attention weights to its neighbors under this relation, and investigate neighbors with top 5 attention weights. Table 5 displays the examples. It shows that GAT is capable of pay-

Table 4: Top 5 frequent paths found for $capitalOf$ (FB15K-237) and $athletePlaysInLeague$ (NELL-995) using two methods. "_inv" indicates inverse relation.

| Relation | Reasoning Path |
|---|---|
| capitalOf (DeepPath) | location$^{-1}$ (83.2%); location$^{-1}$ → location$^{-1}$ (5.3%); administrativeArea$^{-1}$ (4.4%); location$^{-1}$ → location (1.8%); marriage$^{-1}$ → spouse$^{-1}$ → awardWinner → awardNomination → currency → rent$^{-1}$ (0.9%) |
| capitalOf (AttnPath) | location$^{-1}$ (89.1%); county (3.0%); location$^{-1}$ → location$^{-1}$ (2.4%); usCounty (1.8%); administrativeArea$^{-1}$ (1.2%) |
| athletePlaysInLeague (DeepPath) | athleteplaysforteam → teamplaysinleague (28.5%); athleteplayssport → teamplayssport_inv → teamplaysinleague (24.5%); athleteledsportsteam → teamplaysinleague (13.2%); athleteflyouttosportsteamposition → athleteflyouttosportsteamposition_inv → athleteplaysforteam → teamplaysinleague (2.4%); athletehomestadium → leaguestadiums_inv (2.4%) |
| athletePlaysInLeague (AttnPath) | athleteplayssport → teamplayssport_inv → teamplaysinleague (70.0%); athleteplaysforteam → teamplaysinleague (19.5%); athleteledsportsteam → teamplaysinleague (4.3%); athleteflyouttosportsteamposition → athleteflyouttosportsteamposition_inv → athleteplayssport → teamplayssport_inv → teamplaysinleague (3.4%); athleteplaysforteam → teamplaysagainstteam_inv → teamplaysinleague (0.6%) |

Table 5: Samples displaying which neighbor does the model pays the most attention to under certain relation. **Bold Text** means this entity is related to the query relation. The first four pairs are from FB15K-237 and the latter four are from NELL-995.

| Entity | Relation | Neighbors |
|---|---|---|
| Ventura | location | **The United States**, **California**, **Pacific Time Zone**, Marriage, The Rock |
| Sikh | peopleLanguages Spoken | **Malay**, **Thai**, Bhubaneswar, **Hindi**, **Pashto** |
| Anthony Minghella | personNationality | **United Kingdom**, **England**, BAFTA Award for Best Direction, BAFTA Award for Best Film, University of Hull |
| | filmDirector | **BAFTA Award for Best Adapted Screenplay**, **The Talented Mr. Ripley**, **Academy Award for Best Picture**, England, Film Producer |
| Elon Musk | personLeads Organization | **Tesla Motors**, **Paypal**, **Tesla**, **SpaceX Museum**, **SpaceX company** |
| New Jersey Devils | teamPlaysSport | **Hockey**, Stanley Cup, Games, Capitals, Buffalo Sabres |
| Brandon Marshall | athleteHomeStadium | **Invesco Field**, Broncos, Golf, Super Bowl, Super Bowl XLII |
| | athletePlaysSport | **Golf**, Broncos, Invesco Field, Super Bowl XLII, Super Bowl |

ing more attention to the neighbors, which is related to the query relation, especially for Anthony Minghella and Brandon Marshall, which pay different attention to neighbors with different query relations.

## 5 Conclusion and Future Work

In this paper, we propose **AttnPath**, a DRL based model for KG reasoning task which incorporates LSTM and Graph Attention Mechanism as memory components, to alleviate the model from pre-training. We also invent two metrics, MSR and MRR, to measure the learning difficulty of relations, and use it to better fine-tune training hyperparameters. We improve the training process to avoid the agent stalling in a meaningless state. Qualitative experiments and quantitative analysis show that our method outperforms DeepPath and embedding-based method significantly, proving its effectiveness.

In the future, we are interested in utilizing multi-task learning, to enable the model to learn reasoning paths for several query relations simultaneously. We would also like to research how to utilize GAT, MSR and MRR into other KG related tasks, such as KG representation, relation clustering and KB-QA.

## Acknowledgments

This work is supported by the National Key R&D Program of China (2018YFB1004404), Key R&D Program of Guangdong Province (2018B010107005), National Natural Science Foundation of China (U1711262, U1401256, U1501252, U1611264, U1711261).

Rong Pan is the corresponding author.

## References

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, pages 1247–1250.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-

Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NeurIPS*, pages 2787–2795.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka Jr, and Tom M Mitchell. 2010. Toward an architecture for never-ending language learning. In *AAAI*, volume 5, pages 1306–1313.

Wenhu Chen, Wenhan Xiong, Xifeng Yan, and William Wang. 2018. Variational knowledge graph reasoning. In *NAACL-HLT*, pages 1823–1832.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2018. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *ICLR*.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *AAAI*, pages 1811–1818.

Mohnish Dubey, Debayan Banerjee, Debanjan Chaudhuri, and Jens Lehmann. 2018. Strong baselines for simple question answering over knowledge graphs with and without neural networks. In *ISWC*, pages 108–126.

Kelvin Guu, John Miller, and Percy Liang. 2015. Traversing knowledge graphs in vector space. In *EMNLP*, pages 318–327.

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *ACL*, volume 1, pages 687–696.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Ni Lao, Tom Mitchell, and William W Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *EMNLP*, pages 529–539.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2018. Multi-hop knowledge graph reasoning with reward shaping. In *EMNLP*, pages 3243–3253.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, volume 15, pages 2181–2187.

Zhenghao Liu, Chenyan Xiong, Maosong Sun, and Zhiyuan Liu. 2018. Entity-duet neural ranking: Understanding the role of knowledge graph semantics in neural information retrieval. In *ACL*, pages 2395–2405.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 2015. Compositional vector space models for knowledge base completion. In *ACL*, pages 156–166.

Yelong Shen, Jianshu Chen, Po-Sen Huang, Yuqing Guo, and Jianfeng Gao. 2018. M-walk: Learning to walk over graphs using monte carlo tree search. In *NeurIPS*, pages 6787–6798.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *EMNLP*, pages 1499–1509.

Kristina Toutanova, Victoria Lin, Wen-tau Yih, Hoifung Poon, and Chris Quirk. 2016. Compositional learning of embeddings for relation paths in knowledge base and text. In *ACL*, volume 1, pages 1434–1444.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *ICLR*.

William Yang Wang and William W Cohen. 2016. Learning first-order logic embeddings via matrix factorization. In *IJCAI*, pages 2132–2138.

Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. 2019. Explainable reasoning over knowledge graphs for recommendation. In *AAAI*.

Zikang Wang, Linjing Li, Daniel Dajun Zeng, and Yue Chen. 2018. Attention-based multi-hop reasoning for knowledge graph. In *ISI*, pages 211–213.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. In *EMNLP*, pages 564–573.

Fan Yang, Zhilin Yang, and William W Cohen. 2017. Differentiable learning of logical rules for knowledge base reasoning. In *NeurIPS*, pages 2319–2328.

# Learning to Update Knowledge Graphs by Reading News

**Jizhi Tang, Yansong Feng, Dongyan Zhao**
Institute of Computer Science and Technology, Peking University
The MOE Key Laboratory of Computational Linguistics, Peking University
{tangjizhi, fengyansong, zhaody}@pku.edu.cn

## Abstract

News streams contain rich up-to-date information which can be used to update knowledge graphs (KGs). Most current text-based KG updating methods rely on elaborately designed information extraction systems and carefully crafted rules, which are often domain-specific and hard to maintain or generalize. However, such methods may not pay enough attention to the implicit information that lies underneath texts, thus often suffer from coverage issues. In this paper, we propose a novel graph based neural network method, GUpdater, to tackle these problems[1]. GUpdater is built upon graph neural networks (GNNs) with a text-based attention mechanism to guide the updating message passing through the KG structures. Experiments on a real-world KG updating dataset show that our model can effectively broadcast the news information to the KG structures and perform necessary link-adding or link-deleting operations to ensure the KG up-to-date according to news snippets.

## 1 Introduction

Knowledge graphs have been widely used in different areas, where keeping the knowledge triples up-to-date is a crucial step to guarantee the KG quality.

Existing works attempt to synchronize KGs with encyclopedia sources (Morsey et al., 2012; Liang et al., 2017), which basically leverage structured data, while directly updating KGs using plain texts, such as news snippets, is not an easy task and still remains untouched. For example, given a KG recording the rosters of all NBA teams, we need to update this KG according to a news article which describes a trade between different teams. Given the following news snippet that reports a trade between Minnesota Timberwolves and Philadelphia 76ers, one has to add and delete many links in the KG, as illustrated in Figure 1.

> The Minnesota Timberwolves has acquired forward Robert Covington, forward Dario Šarić, guard Jerryd Bayless and a 2022 second-round draft pick from the Philadelphia 76ers in exchange for forward Jimmy Butler and center Justin Patton.

Current approaches either rely on manual updating, or often solve it in a two-step manner: first, using an off-the-shelf information extraction (IE) tool to extract triples from the text, and then modifying the KG according to certain rules predefined by domain experts. However, we should notice that besides what we can explicitly extract from the text, the implicit information, i.e., information that is not mentioned in the text but can be inferred from the text, should also be taken into consideration. This requires far more complicated updating rules. In the above example, if Robert Covington (RC) is traded from Philadelphia 76ers to Minnesota Timberwolves, his teammates should all be changed accordingly, which is not mentioned in the news at all. This is what we referred as implicit information in this scenario.

We should point out that both explicit and implicit information are just different aspects of the same event, which is different from another stream of research that focuses on the evolution of graphs (Pareja et al., 2019; Jin et al., 2019). For example, changing the head coach may trigger a possible trade for his/her favorite players in the future, but these are actually two events, thus beyond the scope of our work.

Intuitively, the implicit information behind the news is related to the KG structures. Many recent works focus on embedding a KG into a continuous vector space, which can be used to fill in miss-

[1]Code and data are available at: https://github.com/esddse/GUpdater



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2632–2641,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

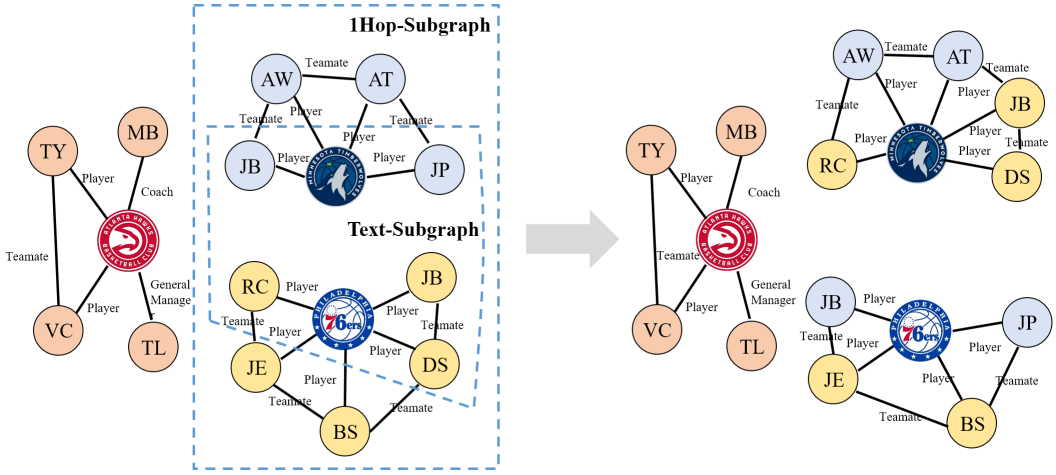


Figure 1: An example illustrating a trade in the KG of NBA teams. Note that all changes described by the news happen in the text-subgraph, the implicit changes, e.g., the teammate changes of player Robert Covington (RC), locate outside the text-subgraph but inside the 1hop-subgraph. The outside of the 1hop-subgraph, e.g, the Atlanta Hawks part, will not be affected by this trade.

ing links in KGs (Bordes et al., 2013; Wang et al., 2014; Lin et al., 2015; Yang et al., 2014; Trouillon et al., 2016; Schlichtkrull et al., 2018). However, most of them learn from static KGs, thus unable to help with our task, since they can not dynamically add new links or delete obsolete links according to extra text information.

In this paper, we propose a novel neural model, GUpdater, to tackle this problem, which features a graph based encoder to learn latent KG representations with the guidance from the news text, and a decoder to score candidate triples with reconstructing the KG as the objective. Our encoder is built upon the combination of recently proposed R-GCN (Schlichtkrull et al., 2018) and GAT (Velickovic et al., 2017) with two new key factors designed specifically for our task. The main idea is to control the updating message from the text passing through the KG. First, we use the given text to generate all the graph attention weights in order to selectively control message passing. Second, we link all entities mentioned in the news text together as shortcuts in order to let the message pass to each other even if they are topologically far from each other in the KG. For the decoder, we simply use DistMult (Yang et al., 2014) to score related triples to be updated. To evaluate our method, we construct a new real-world dataset, NBAtransactions, for this task. Experimental results show that our model can effectively use the news text to dynamically add new links and remove obsolete links accordingly.

Our contributions are in two-fold:

1. we propose a new text-based KG updating task and release a new real-world dataset to the community for further research.

2. we design a novel neural methodl, GUpdater, to read text snippets, which features an attention mechanism to selectively control the message passing over KG structures. This novel architecture enables us to perform both link-adding and link-deleting to ensure the KG up-to-date.

## 2 Task Formulation

We first formally define the task. Given a knowledge graph $\mathcal{G} = (\mathbf{E}, \mathbf{R}, \mathbf{T})$, where $\mathbf{E}$, $\mathbf{R}$ and $\mathbf{T}$ are the entity set, relation set and KG triple set, respectively, and a news text snippet $S = \{w_1, w_2, \cdots, w_{|S|}\}$, for which entity linking has been performed to build the mentioned entity set $\mathbf{L} \subset \mathbf{E}$, the text-based knowledge graph updating task is to read the news snippet $S$ and update $\mathbf{T}$ accordingly to get the final triple set $\mathbf{T}'$ and the updated graph $\mathcal{G}' = (\mathbf{E}, \mathbf{R}, \mathbf{T}')$. In this paper, we focus on the scenarios where the entity set remains unchanged.

As a single event can only affect the KG in a limited range, for each event that is described by a news snippet, we consider two kinds of subgraphs that are defined by their entity sets, as illustrated in Figure 1:

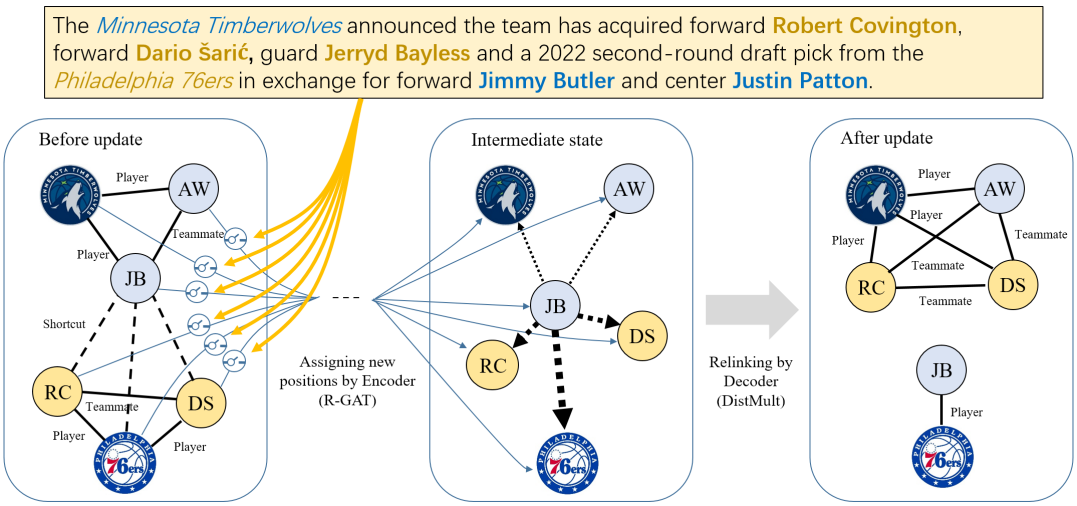


Figure 2: Overview of our GUpdater. By adding shortcuts in the R-GAT encoder, we enable Jimmy Butler (JB) to directly receive information from Philadelphia 76ers. Then with the guidance of text-based attention, Jimmy Butler selectively gathers neighbors' messages and moves closer to Philadelphia 76ers in the latent space. Dashed lines with arrow in the intermediate state indicate the moving directions, and the width of the dash line illustrates how possible Jimmy Butler will move to that direction. Finally, new links are generated by the DistMult decoder.

**Text-Subgraph** Subgraphs with entity set $\mathbf{E}_{text} = \mathbf{L}$, i.e., all the entities are mentioned in news snippet $S$. Explicit information in the texts can lead to changes in these graphs only.

**1Hop-Subgraph** If we expand text-subgraphs by one hop, we get 1hop-subgraphs. For each entity in 1hop-subgraphs, either itself or one of its neighbors is mentioned in the news. The one-hop entity set is formally defined as $\mathbf{E}_{1hop} = \{e|e \in \mathbf{L} \vee (((e,r,t) \in \mathbf{T} \vee (t,r,e) \in \mathbf{T}) \wedge t \in \mathbf{L})\}$.

We argue that updating in 1hop-subgraphs only is a good balance of effectiveness and efficiency. On the one hand, implicit changes always occur around explicit changes usually within a short distance. In fact, most of the implicit changes can be captured in the one-hop range. Here we give an extreme example: given the news snippet "*Barack Obama was elected president.*", the explicit change is to add (Barack Obama, Position, President) to the KG, while one of the implicit changes is to add (Michelle Obama, Position, First Lady) to the KG. Note that in this case, both "Michelle Obama" and "First Lady" are not mentioned in the news, but, because they are neighbors of "Barack Obama" and "President", respectively, this implicit change can be found in the 1hop-subgraph. On the other hand, updating in small subgraphs can avoid a huge amount of meaningless computation, and meanwhile can reduce mis-prediction. For these two reasons, we select the one-hop range, i.e., the 1hop-subgraph as the main testbed for this task. Also note that this one-hop setting can be easily extended to two-hop or larger scopes if necessary.

## 3 Our Method

The overview of our model is illustrated in Figure 2. As mentioned before, our GUpdater follows an encoder-decoder framework with the objective to reconstruct the modified KGs.

### 3.1 Relational Graph Attention Layer

In order to better capture the KG structures, we propose a new graph encoding layer, the relational graph attention layer (R-GAT), as the basic building block of GUpdater's encoder. R-GAT can be regarded as a combination of R-GCN (Schlichtkrull et al., 2018) and GAT (Velickovic et al., 2017), which benefits from both the ability of modeling relational data and the flexibility of attention mechanism. Recall that the layer-wise propagation rule of R-GCN can be written as follows:

$$\mathbf{H}^{l+1} = \sigma\left(\sum_{r \in \mathbf{R}} \hat{\mathbf{A}}_r^l \mathbf{H}^l \mathbf{W}_r^l\right), \quad (1)$$

where $\hat{\mathbf{A}}_r^l$ is the normalized adjacent matrix for relation $r$, $W_r^l$ is a layer-specific trainable weight matrix for relation $r$, and $\sigma(\cdot)$ denotes an activa-

tion function, here we use ReLU. $H^l$ is the latent entity representations in the $l^{th}$ layer.

Upon R-GCN, we can easily introduce attention mechanisms by computing $\hat{\mathbf{A}}_r$ using an attention function:

$$a_{ij}^{lr} = \begin{cases} \dfrac{\exp(att^{lr}(\mathbf{h}_i^l, \mathbf{h}_j^l))}{\sum_{k \in \mathcal{N}_i^r} \exp(att^{lr}(\mathbf{h}_i^l, \mathbf{h}_k^l))} & , j \in \mathcal{N}_i^r \\ 0 & , \text{otherwise} \end{cases} \tag{2}$$

where $a_{ij}^{lr}$ is the $i^{th}$ row and $j^{th}$ column element of $\hat{\mathbf{A}}_r^l$, and $\mathcal{N}_i^r$ denotes the set of neighbor indices of node $i$ under relation $r$. $att^{lr}(\cdot,\cdot)$ is the attention function. $\mathbf{h}_i^l$ and $\mathbf{h}_j^l$ are the $i^{th}$ and the $j^{th}$ entity representations of layer $l$, respectively.

A known issue for R-GCN (Eq.(1)) is that the number of model parameters grows rapidly with the increasing number of relation types. We thus use *basis-decomposition* (Schlichtkrull et al., 2018) for regularization. In a standard R-GCN layer, if there are $R$ relations involved, there will be $R$ weight matrices for each layer. Basis-decomposition regularizes these matrices by defining each weight matrix as a linear combination of $B(B < R)$ basis matrices, which significantly decreases the parameter number.

### 3.2 Text-based Attention

The core idea of GNNs is to gather neighbors' information. In our case, we propose a text-based attention mechanism to utilize the news snippet to guide the message passing along the KG structure within the R-GAT layer.

We first use bi-GRU (Cho et al., 2014) to encode the given news text $S$ into a sequence of representations $\{\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_{|S|}\}$, then we leverage the sequence attention mechanism (Luong et al., 2015) to compute the context vector:

$$\mathbf{c}^{lr} = \sum_{t=1}^{|S|} b_t^{lr} \mathbf{u}_t \tag{3}$$

where $b_t^{lr}$ is the text attention weight, and is computed as follow:

$$b_t^{lr} = \frac{\exp(\mathbf{u}_t^T \mathbf{g}_{text}^{lr})}{\sum_{k=1}^{|S|} \exp(\mathbf{u}_k^T \mathbf{g}_{text}^{lr})} \tag{4}$$

where $\mathbf{g}_{text}^{lr}$ is a trainable guidance vector to guide the extraction of relation-dependent context.

Recall that in GAT, the way to compute the attention weights is similar to the formula below:

$$att^{lr}(\mathbf{h}_i^l, \mathbf{h}_j^l) = {\mathbf{g}_{graph}^{lr}}^T [\mathbf{h}_i^l || \mathbf{h}_j^l] \tag{5}$$

where $||$ denotes concatenation operation, and $\mathbf{g}_{graph}^{lr}$ is a relation-specific trainable vector that serves as a graph guidance vector to decide which edge to pay attention to.

Here, we generate the final guidance vector $\mathbf{g}_{fin}^{lr}$ that combines textual information to the graph guidance vector by simply using linear interpolation, and we replace $\mathbf{g}_{graph}^{lr}$ in Eq.(5) by $\mathbf{g}_{fin}^{lr}$ to get our final attention function:

$$\mathbf{g}_{fin}^{lr} = \alpha^{lr} \mathbf{g}_{graph}^{lr} + (1 - \alpha^{lr}) \mathbf{U}^{lr} \mathbf{c}^{lr}, \tag{6}$$

$$att^{lr}(\mathbf{h}_i^l, \mathbf{h}_j^l) = {\mathbf{g}_{fin}^{lr}}^T [\mathbf{h}_i^l || \mathbf{h}_j^l] \tag{7}$$

where $\mathbf{U}^{lr}$ is a trainable transformation matrix, and $\alpha^{lr} \in [0, 1]$ can either be trainable or fixed. If we set $\alpha^{lr} = 1$, then Eq.(7) degenerates to Eq.(5) and our encoder degenerates to R-GAT. It makes it easy to pre-train the model to get good embeddings for all entities and relations when news snippets are not provided.

### 3.3 Shortcuts

**Shortcuts** In practice, entities in $\mathbf{L}$ can be far from each other in the KG, even unreachable sometimes, while a single- or two-layer R-GAT can only gather information from a near neighborhood. In order to encourage more direct interactions through the graph structure, before running the model, we simply link all entities in $\mathbf{L}$ to each other and assign these links with a new relation label, *SHORTCUT*.

**Shortcuts with labels** Actually, relations between entities in $\mathbf{L}$ can be different. For example, if a player is traded from one team to another, the relations between this player and that two teams must be opposite. Thus a unified shortcut label may make it difficult to correctly pass opposite messages, even with the help of the attention mechanism. So we further extend the shortcuts' label to 3 types: *ADD*, *DEL* and *OTHER*. *ADD* (*DEL*) denotes that these triples are explicitly mentioned in the text to be added to (deleted from) the KG. The rest are labeled with *OTHER*. Off-the-shelf IE tools can be used to generate these labeled shortcuts.

| | Data type distribution | | Edge number distribution (average number of edges in one instance) | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Type** | **Count** | **Percentage** | **Added** | **Deleted** | **Unchanged** | **T-added** | **T-deleted** | **T-unchanged** |
| Trade | 1245 | 30.4% | 37.48 | 37.19 | 330.51 | 2.08 | 2.09 | 0.50 |
| Head coach | 77 | 1.9% | 1.0 | 1.0 | 217.0 | 1.0 | 1.0 | 0.14 |
| Free agency | 1078 | 26.3% | 19.19 | 0.0 | 179.89 | 1.0 | 0.0 | 0.0 |
| Draft | 137 | 3.3% | 18.84 | 0.0 | 173.42 | 1.0 | 0.0 | 0.0 |
| Released | 1382 | 33.7% | 0.0 | 18.65 | 170.26 | 0.0 | 1.0 | 0.0 |
| Overseas | 85 | 2.1% | 0.0 | 17.62 | 150.05 | 0.0 | 1.0 | 0.0 |
| Retirement | 57 | 1.4% | 0.0 | 16.51 | 132.07 | 0.0 | 1.0 | 0.0 |
| D-league | 39 | 1.0% | 0.0 | 19.13 | 178.0 | 0.0 | 1.0 | 0.0 |
| Overall | 4100 | 100% | 17.07 | 18.37 | 221.56 | 0.95 | 1.03 | 0.15 |

Table 1: The Statistics of NBAtransactions. Added, Deleted and Unchanged represent the average number of edges that should be added to, deleted from and remain unchanged in the KG, respectively. T-added represents the average edge addition in the text-subgraphs, similar for T-deleted and T-unchanged.

Here, we easily build a simple extraction model using our GUpdater encoder. We get the entity representations from GUpdater's encoder (with unified shortcuts), then for each entity pair $(e_i, e_j)$, where $e_i, e_j \in \mathbf{L}, i \neq j$, we use an MLP classifier to get the probability of each label:

$$P(z) = \text{softmax}\left(\text{MLP}([\mathbf{h}_i||\mathbf{h}_j])\right) \quad (8)$$

where shortcut label $z \in \{\textit{ADD}, \textit{DEL}, \textit{OTHER}\}$, $\mathbf{h}_i$ and $\mathbf{h}_j$ are entity representations for entity $e_i, e_j$, respectively.

In our experiments, we train the shortcut-labeling module separately, as we find that the results are good enough while saving training time. One can surely perform the extraction step in a joint training fashion.

### 3.4 Decoder

We use DistMult (Yang et al., 2014), which is known to have good performance on standard KG completion tasks, followed by sigmoid function, as the decoder, i.e., for each possible triple $(e_i, r_k, e_j)$ where $e_i, e_j \in \mathbf{E}_{1hop}, r_k \in \mathbf{R}$, the probability of this triple to appear in the final KG is computed as follow:

$$P(y) = \text{sigmoid}\left(\mathbf{h}_i^T(\mathbf{r}_k \circ \mathbf{h}_j)\right) \quad (9)$$

where $\circ$ denotes element-wise multiplication.

Since we formulate KG updating as a binary classification task, we use the cross-entropy loss to train the model.

## 4 Experiments

### 4.1 Dataset

We construct a new dataset, NBAtransactions, to evaluate our method. NBAtransactions contains 4,100 transaction-news pairs in NBA from 2010 to 2019, 3,261 for training, 417 for validation and 422 for testing. We consider 8 different kinds of transactions, which can be divided into 3 categories according to the updating patterns: 1) *Adding edges only*: free agency and draft. 2) *Deleting edges only*: released, overseas, retirement and d-league. 3) *Both adding and deleting edges*: trade and head coach.

For each transaction, a news snippet $S$ with mentioned entity set $\mathbf{L}$ and two undirected KG fragments, $\mathcal{G}_{1hop}$ and $\mathcal{G}'_{1hop}$, representing the corresponding 1hop-subgraphes before and after the transaction, respectively, are given. Averagely, one subgraph contains 27.86 entities, 4 types of relations, and 239.28 triples, and one news snippet contains 29.10 words and 2.66 KG entities. Each transaction averagely causes adding of 17.07 edges and deleting of 18.37 edges; among them, only 0.95 (for adding) and 1.03 edges (for deleting) are in the text-subgraph, i.e., only 5.6% of the edge changes are explicitly mentioned in the news texts. Detailed statistics are shown in Table 1.

Our dataset is collected from several NBA related webistes. The KGs we used in the dataset are constructed using the roster of each NBA team in each season collected from Basketball-Reference[2]. For each NBA season, we build a large KG that records all teams, players, head coaches, general managers and their relations at the beginning of that season. So there are 9 large KGs in total, one KG for one NBA season. Wikipedia[3] records all NBA transactions from 1946 to 2019 in a structured form and provides URLs of news sources for most of the transactions after 2010. We crawled all the available

[2] https://www.basketball-reference.com

[3] https://en.wikipedia.org/wiki/Category:NBA_transactions

news and corresponding transactions. For each transaction, $\mathcal{G}_{1hop}$ can be easily extracted from the corresponding large KG. However, we cannot get $\mathcal{G}'_{1hop}$ directly, as our KGs only record the rosters at the beginning of each season. To generate the $\mathcal{G}'_{1hop}$, we manually create a large set of complicated conversion rules for each transaction type to modify $\mathcal{G}_{1hop}$ and obtain the 1hop-subgraph after the transaction. We use string matching algorithms to perform entity linking, and meanwhile generate the mentioned entity set $\mathbf{L}$.

### 4.2 Setup

The dimensions of all embeddings (words, entities, and relations) are all set to 128, and the hidden dimension is 256. We use a single layer encoder, as we find that more layers do not bring any benefit. The basis number of basis-decomposition is 2. We replace the word embeddings of entity mentions in the text by the entity embeddings for better alignment of word embedding space and entity embedding space. The entity embeddings and relation embeddings are pre-trained using R-GAT, and the word embeddings are randomly initialized. We set dropout (Srivastava et al., 2014) rate to 0.5. The batch size in our experiments is 1. We use Adam optimizer (Kingma and Ba, 2014) with a learning rate of 0.001 for training.

We consider updating in two different scopes: 1) Text-Subgraphs: changes in these subgraphs correspond to the explicit information mentioned in the texts. 2) 1Hop-Subgraphs: all explicit and implicit changes happen in these subgraphs, thus this setting is treated as the overall/real-world updating evaluation.

### 4.3 Metrics

As our model aims to reconstruct the KG using binary classification, i.e., deciding whether a possible triple should be in the modified KG, we use accuracy, precision, recall and F1 score as the evaluation metrics. Further, to evaluate the ability of link-adding, link-deleting and link-preserving, respectively, we collect all added edges, deleted edges, and unchanged edges and compute the prediction accuracies separately, which we denote as Added Acc, Deleted Acc and Unchanged Acc, respectively.

### 4.4 Baseline Models

Because most current text-based KG updating methods are in two steps: first extract information from texts, then add and remove links in KGs, we select non-rule-based models that perform well on these two steps and also their combination as our baseline models.

**PCNN** (Zeng et al., 2015) is a strong baseline for relation extraction, which divides the sentence into three pieces and applies max-pooling in a piecewise manner after the convolution layer.

**IE-gold** is a simulation of an ideal IE model that can perfectly extract explicit information from given texts. This is an upper bound for the information extraction step.

**DistMult** (Yang et al., 2014) is a widely-used multiplication-based triple scoring function for KG completion, which computes the three-way inner-product of the triples. Its symmetric nature is suitable for NBAtransactions as the KGs are undirected.

**R-GCN** (Schlichtkrull et al., 2018) is one of the strongest baselines for KG completion, which first encodes entities by gathering neighbors' information, then decodes them by DistMult. Note that compared to traditional KG completion tasks, here the target KGs are the modified KGs and usually different from the input KGs. Because of the lack of network structure, single DistMult is unable to train like this.

**IE-gold + R-GCN** is the combination of IE-gold and R-GCN. We first use IE-gold to generate a perfectly updated text-subgraph, then use R-GCN to predict other changes outside the text-subgraph.

### 4.5 Main Results

We summarize the main results in Table 2. As we can see, PCNN performs pretty well in extracting explicit information from the news (over 0.96 in Added Acc, Deleted Acc and Unchanged Acc in Text-Subgraphs). However, the explicit changes only take up 5.8% in the testing set, while the majority of changes are implicit. Therefore, even a perfect IE model, IE-gold, which never makes any wrong predictions, can only correctly predict about 6% of the changes in the 1hop-subgraphs.

The implicit changes are highly related to KG structures. Compared to DistMult, which just preserves the original structures and scores 0.0656 in Added Acc, 0.1220 in Deleted Acc in the 1Hop-Subgraphs, R-GCN can use the modified KGs for training and learns to predict changes without reading the news. R-GCN beats DistMult by

| | Accuracy | Precision | Recall | F1 | Added Acc | Deleted Acc | Unchanged Acc |
|---|---|---|---|---|---|---|---|
| **Text-Subgraphs** | | | | | | | |
| PCNN | 0.9664 | 0.9573 | 0.9676 | 0.9624 | 0.9674 | 0.9638 | 0.9686 |
| IE-gold | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| DistMult | 0.8372 | 0.1640 | 0.1670 | 0.1655 | 0.0679 | 0.0652 | 0.7093 |
| R-GCN | 0.8601 | 0.3249 | 0.4147 | 0.3644 | 0.4055 | 0.2077 | 0.4651 |
| IE-gold + R-GCN | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| GUpdater | 0.9783 | 0.8871 | 0.8887 | 0.8879 | 0.8684 | 0.9185 | 0.9651 |
| **1Hop-Subgraphs** | | | | | | | |
| PCNN | 0.9799 | 0.9245 | 0.9270 | 0.9257 | 0.0586 | 0.0541 | 0.9995 |
| IE-gold | 0.9805 | 0.9248 | 0.9274 | 0.9261 | 0.0606 | 0.0561 | 1.0 |
| DistMult | 0.9644 | 0.8277 | 0.9225 | 0.8725 | 0.0656 | 0.1220 | 0.9944 |
| R-GCN | 0.9837 | 0.9243 | 0.9547 | 0.9393 | 0.4348 | 0.2005 | 0.9984 |
| IE-gold + R-GCN | 0.9847 | 0.9285 | 0.9577 | 0.9429 | 0.4681 | 0.2448 | 0.9989 |
| GUpdater | 0.9964 | 0.9819 | 0.9913 | 0.9866 | 0.8926 | 0.8988 | 0.9991 |

Table 2: Model performance on NBAtransactions. Here, PCNN performs multi-class classification while others are binary classification models.

0.37 in Added Acc and 0.08 in Deleted Acc in the 1Hop-Subgraphs. Such a huge improvement suggests that the blind prediction is not completely arbitrary, and R-GCN indeed learns certain updating patterns through the KG structures. Besides, R-GCN scores 0.9393 in F1 in the 1hop-subgraphs, outperforms IE-gold by 1%, once again underlines the importance of implicit changes.

IE-gold + R-GCN combines the advantages of IE-gold and R-GCN, and performs best among the baselines. However, in the 1hop-subgraphs, the Added Acc and Deleted Acc are only 0.4681 and 0.2448, which are still quite low. Although IE-gold + R-GCN can perfectly extract explicit information from the news, there is no mechanism for the IE model to pass the extracted information to the R-GCN module, thus this two-step method still performs poorly in finding out the implicit changes.

For overall performance, GUpdater significantly outperforms all baseline models, which coincides to our intuition. Particularly, in the 1hop-subgraphs, GUpdater beats IE-gold + R-GCN by 0.42 in Added Acc, 0.65 in Deleted Acc, indicating that GUpdater can well capture both explicit and implicit information in the news snippets.

### 4.6 Ablation Analysis

We also perform ablation test to figure out the effect of each component of GUpdater. As shown in Table 3, compared to R-GAT, both GUpdater-shortcut (GUpdater without shortcut) and GUpdater-text (GUpdater without text-based attention) improve the overall performance with different levels, showing that both the text-based attention and the shortcuts are helpful for this task.

Generally, we can see that adding shortcuts is key to this task, as it brings a giant leap to nearly all indicators. Actually, updating KGs using news can be regarded as information injection from one semantic space to another, the text mentioned entities can be seen as junctions of these two spaces and serve as entrances for information injection. So, in the KG, the text mentioned entities are information sources, and send messages to the explicitly and implicitly related target entities. However, the targets may be very far and hard to reach, it is the shortcuts that make the successful information delivery possible.

However, if there are too many shortcuts, the messages are easy to get to wrong targets. The text-based attention mechanism can be regarded as gates that selectively open several shortcuts that are easy for message passing, and this helps the model get 2%-3% steady improvements in both Added Acc and Deleted Acc.

When adding explicit labels to shortcuts, we get much better performance than the basic GUpdater (7% improvement in Added Acc, 8% improvement in Deleted Acc in 1hop-subgraphs), which indicates that splitting the shortcuts into different channels for different kinds of messages is crucial for better information delivery, and that makes it easier to dig out implicit information. The result also indicates that our model is well extensible and can easily incorporate off-the-shelf IE tools.

### 4.7 Visualization of Attention on Shortcuts

In order to explore how the information actually passes along shortcuts, we select 5 different types of trades, as shown in Table 4, and we get the corresponding attention weight on each shortcut in

| | Accuracy | Precision | Recall | F1 | Added Acc | Deleted Acc | Unchanged Acc |
|---|---|---|---|---|---|---|---|
| **Text-Subgraphs** | | | | | | | |
| GUpdater | 0.9783 | 0.8871 | 0.8887 | 0.8879 | 0.8684 | 0.9185 | 0.9651 |
| -shortcut-text (R-GAT) | 0.8623 | 0.3343 | 0.4273 | 0.3751 | 0.4225 | 0.2179 | 0.4535 |
| -shortcut | 0.8656 | 0.3551 | 0.4776 | 0.4074 | 0.2059 | 0.5112 | 0.3372 |
| -text | 0.9705 | 0.8550 | 0.8366 | 0.8457 | 0.8323 | 0.8982 | 0.8605 |
| GUpdater+label | 0.9936 | 0.9779 | 0.9551 | 0.9664 | 0.9584 | 0.9603 | 0.9862 |
| **1Hop-Subgraphs** | | | | | | | |
| GUpdater | 0.9964 | 0.9819 | 0.9913 | 0.9866 | 0.8926 | 0.8988 | 0.9991 |
| -shortcut-text (R-GAT) | 0.9833 | 0.9216 | 0.9547 | 0.9378 | 0.4290 | 0.2073 | 0.9990 |
| -shortcut | 0.9843 | 0.9267 | 0.9563 | 0.9413 | 0.4726 | 0.2044 | 0.9989 |
| -text | 0.9957 | 0.9793 | 0.9885 | 0.9839 | 0.8668 | 0.8619 | 0.9987 |
| GUpdater+label | 0.9981 | 0.9889 | 0.9969 | 0.9929 | 0.9668 | 0.9792 | 0.9991 |

Table 3: Results of the ablation test, where -text indicates removing the text-based attention mechanism from GUpdater.

GUpdater. We find that most shortcuts are closed by the attention mechanism, i.e., their corresponding attention weights are very close to 0, and only a small portion of shortcuts are open for message passing.

For each trade type, the remaining shortcuts are organized in a similar pattern: there is a central team selected by the model that sends messages to all other entities and receives the message from another team. For symmetric trades, i.e., each team plays the same role in the trade (T2, T3 in Table 4), the selection of the central team seems to be arbitrary, while for asymmetric trades (T4, T5), the central teams are the teams that involved in more transactions.

It seems that GUpdater learns a 2-team trade rule from that pattern: if two teams send messages mutually through shortcuts, they will exchange all their players in text-subgraphs. T1 and T2 are most simple 2-team trades, thus our model updates the graphs perfectly with this message passing pattern. T4 and T5 look difficult and complicated, but they are actually direct combinations of two simple 2-team trades, e.g., T4 can be decomposed into two trades: the Grizzlies trades JS to the Celtics for FM, and the Thunder trades RG to the Celtics. So for these trades, GUpdater also performs quite well, as only two not-that-important triples are missing in T5. However, in T3, three teams exchange players in a rotation, and it cannot be decomposed into several 2-team trades, thus leads to a severe mis-prediction: (TS, player, Jazz), as placing a player in a wrong team may cause more teammate-errors in the 1hop-subgraph. The inability of perfectly performing rotational three-team trades may due to the lack of training instance, as in the past 10 years, such kind of trade happened nearly once a year.

## 5 Related Work

**KG Representation Learning** aims to embed a KG into a continuous vector space, which preserves the KG structures. There are mainly two streams of researches. The first is addition-based models (also called translation-based models), which based on the principle that for every valid triple $(h, r, t)$, their embeddings holds: $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$. **TransE**(Bordes et al., 2013) is the first such model. **TransH**(Wang et al., 2014) projects entity embeddings into relation-specific hyperplanes. **TransR**(Lin et al., 2015) generalize TransH by extending projection to linear transformation. **TransD** (Ji et al., 2015) simplifies TransR by decomposing the transformation matrix into the product of two vectors. Another is multiplication-based models, which comes from the idea of tensor decomposition. **RESCAL**(Nickel et al., 2011) is one of the earlist studies, which using a bilinear scoring function. **DistMult**(Yang et al., 2014) simplifies RESCAL by using a diagonal matrix. **ComplEx**(Trouillon et al., 2016) extend DistMult into the complex space to handle asymmetric relations. **SimplE** (Kazemi and Poole, 2018) learns two embeddings for each entity dependently. A direct application is KG completion.

**Graph Neural Networks** allow passing information horizontally among nodes (Gilmer et al., 2017). Recently, many tecniques has been incorporated to GNN model, and lots of attempt has bean made to promote GNNs to more application scenarios. **GCN**(Kipf and Welling, 2016) leverages graph spectrums for semi-supervised node classification. **GAT**(Velickovic et al., 2017) extends GCN by bringing attention mechanism

| | $\mathcal{G}_{text}$ | $\mathcal{G}'_{text}$ | Selected Shortcuts | Results in Text-Subgraph |
|---|---|---|---|---|
| T1 |  |  |  | ✓ |
| T2 |  | 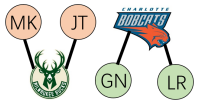 | 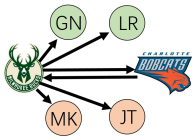 | ✓ |
| T3 | 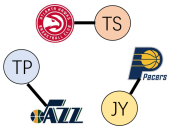 | 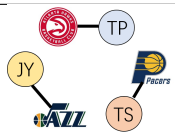 |  | wrong predictions:<br>(TS, player, Jazz)<br>(TS, teammate, Pacers) |
| T4 | 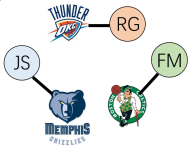 | 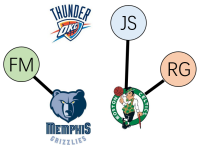 | 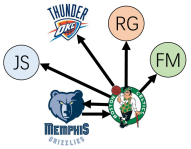 | ✓ |
| T5 | 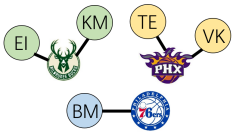 | 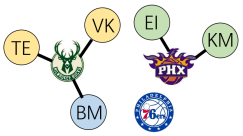 | 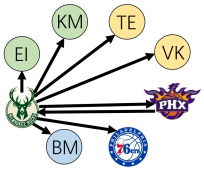 | missing triples:<br>(VK, teammate, BM)<br>(TE, teammate, BM) |

Table 4: Examples of visualization for the attention on shortcuts. Each row shows an example of one type of trade. $\mathcal{G}_{text}$ and $\mathcal{G}'_{text}$ represent the actual text-subgraphs before and after the trade, respectively. *Selected Shortcuts* represents the shortcuts selected by the attention mechanism, and the arrows indicate the directions of message passing. *Results in Text-Subgraph* lists the prediction errors of GUpdater.

into GNN. To better capture KG information, **R-GCN**(Schlichtkrull et al., 2018) was proposed to incorporates relation embeddings into GNN.

**Relation Extraction** task aims to extract relations from texts. Current relation extraction models are mainly under distant supervision (Mintz et al., 2009), and are trained in bag level. **PCNN** (Zeng et al., 2015) divides the sentence into three pieces and applies max-pooling in a piecewise manner after the convolution layer. **APCNN** (Ji et al., 2017) uses sentence-level attention to select multiple valid sentences with different weights in a bag. (Luo et al., 2017) uses a transition matrix to model the noise and use curriculum for training.

## 6 Conclusion

In this paper, we propose a new text-based KG updating task and construct a dataset, NBAtransactions, for evaluation. We design a novel GNN-based model, GUpdater, which uses text information to guide the message passing through the KG structure. Experiments show that our model can effectively handle both explicit and implicit information, and perform necessary link-adding and link-deleting operations accordingly. In the future, we will try to investigate how to update KGs when entities are involved in several successive events.

## Acknowledgements

This work is supported in part by the National Hi-Tech R&D Program of China (No. 2018YFC0831900) and the NSFC Grants (No. 61672057, 61672058). For any correspondence, please contact Yansong Feng.

## References

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural

message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1263–1272. JMLR. org.

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 687–696.

Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. 2017. Distant supervision for relation extraction with sentence-level attention and entity descriptions. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Woojeong Jin, Changlin Zhang, Pedro Szekely, and Xiang Ren. 2019. Recurrent event network for reasoning over temporal knowledge graphs. *arXiv preprint arXiv:1904.05530*.

Seyed Mehran Kazemi and David Poole. 2018. Simple embedding for link prediction in knowledge graphs. In *Advances in Neural Information Processing Systems*, pages 4284–4295.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Jiaqing Liang, Sheng Zhang, and Yanghua Xiao. 2017. How to keep a knowledge base synchronized with its encyclopedia source.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, volume 15, pages 2181–2187.

Bingfeng Luo, Yansong Feng, Zheng Wang, Zhanxing Zhu, Songfang Huang, Rui Yan, and Dongyan Zhao. 2017. Learning with noise: Enhance distantly supervised relation extraction with dynamic transition matrix. *arXiv preprint arXiv:1705.03995*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Mohamed Morsey, Jens Lehmann, Sören Auer, Claus Stadler, and Sebastian Hellmann. 2012. Dbpedia and the live extraction of structured data from wikipedia. *Program*, 46(2):157–181.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *ICML*, volume 11, pages 809–816.

Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, and Charles E Leisersen. 2019. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. *arXiv preprint arXiv:1902.10191*.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, pages 2071–2080.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 1(2).

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, volume 14, pages 1112–1119.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1753–1762.

# DIVINE: A Generative Adversarial Imitation Learning Framework for Knowledge Graph Reasoning

**Ruiping Li** and **Xiang Cheng***
State Key Laboratory of Networking and Switching Technology
Beijing University of Posts and Telecommunications, Beijing 100876, China
{liruiping,chengxiang}@bupt.edu.cn

## Abstract

Knowledge graphs (KGs) often suffer from sparseness and incompleteness. Knowledge graph reasoning provides a feasible way to address such problems. Recent studies on knowledge graph reasoning have shown that reinforcement learning (RL) based methods can provide state-of-the-art performance. However, existing RL-based methods require numerous trials for path-finding and rely heavily on meticulous reward engineering to fit specific dataset, which is inefficient and laborious to apply to fast-evolving KGs. In this paper, we present DIVINE, a novel plug-and-play framework based on generative adversarial imitation learning for enhancing existing RL-based methods. DIVINE guides the path-finding process, and learns reasoning policies and reward functions self-adaptively through imitating the demonstrations automatically sampled from KGs. Experimental results on two benchmark datasets show that our framework improves the performance of existing RL-based methods without extra reward engineering.

## 1 Introduction

Knowledge graphs (Suchanek et al., 2007; Auer et al., 2007; Bollacker et al., 2008; Carlson et al., 2010; Vrandečić and Krötzsch, 2014), typically composed of massive relational triples, are useful resources for many downstream natural language processing applications such as information extraction and question answering. Although existing KGs have an extraordinarily large scale, they are still highly incomplete (Min et al., 2013), leading extensive research efforts on automated inference of missing information from observed evidence. In this paper, we focus on the problem of multi-hop reasoning in KGs, which learns explicit inference formulas from existing triples to complete missing ones.

*Corresponding author

To tackle multi-hop reasoning, various path-based methods (Lao et al., 2011; Gardner et al., 2013, 2014; Guu et al., 2015; Neelakantan et al., 2015; Toutanova et al., 2016; Das et al., 2017) have been proposed, which leverage the elaborately selected relational paths in KGs as the reasoning evidence. However, such evidential paths are obtained by random walks, which inevitably introduces inferior or even noisy paths. To address this problem, the RL-based methods, such as DeepPath (Xiong et al., 2017) and MINERVA (Das et al., 2018), strive for more reliable evidential paths by policy-conditioned walking and achieve state-of-the-art performance. They formulate the path-finding problem as a Markov decision process where their policy-based agents continuously choose the most promising relation for state transition based on the current state and reasoning policy. Once a relational path found, the reasoning policy is updated by a reward function according to the path quality. Finally, through such a trial-and-error process, the well-trained policy-based agent can be used to find evidential paths for predictions.

However, these RL-based methods still suffer from the following pain points. Firstly, they tend to require numerous trials from scratch to find a reliable evidential path since the action space can be very large due to the complexity of KGs, which leads to poor convergence properties. Secondly, and most importantly, an efficient trial-and-error optimization in RL requires designing a reward function manually to fit the specific dataset. However, such reward engineering depends on meticulous artificial design with domain expertise, which can be significantly challenging in practice (Ng et al., 2000). In particular, these RL-based methods are extremely sensitive to their reward functions, where a little variation may lead to a significant fluctuation of the reasoning performance.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2642–2651, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

Therefore, for different datasets, the reward functions in the RL-based methods need manual adjustments to achieve a good performance, which will not only be inefficient and laborious but also difficult to adapt to the rapid evolutions of real-world KGs (Shi and Weninger, 2018).

In this paper, we present a novel plug-and-play framework based on generative adversarial imitation learning (GAIL) (Ho and Ermon, 2016) for enhancing existing RL-based methods, which is referred to as DIVINE for "Deep Inference via Imitating Non-human Experts". DIVINE trains a reasoner, consisting of a generator and a discriminator, from demonstrations by employing generative adversarial training, where the generator can be any of the policy-based agents in existing RL-based methods and the discriminator can be considered as a self-adaptive reward function. In this way, for different datasets, the reward functions can be automatically tuned to approximate the optimal performance, eliminating extra reward engineering and manual interventions. In particular, to enable the policy-based agent to find more diverse evidential paths for predictions, we propose a path-based GAIL method, which can learn the reasoning policy by imitating the path-level semantic features of the demonstrations. In addition, to acquire demonstrations without extra manual labor, we design an automated sampler for our framework to dynamically sample relational paths from KGs as the demonstrations according to the specific environment of each entity.

In summary, our contributions are threefold:

- We present a plug-and-play framework based on GAIL to enhance existing RL-based reasoning in KGs by learning reasoning policies and reward functions through imitating demonstrations. To the best of our knowledge, we are the first to introduce GAIL into the field of knowledge graph reasoning.
- We propose a path-based GAIL method to encourage the diversity of evidential paths and design an automated sampler for our framework to sample demonstrations without extra manual labor.
- We conduct extensive experiments on two benchmark datasets. The experimental results illustrate that our framework improves the performance of the current state-of-the-art RL-based methods while eliminating extra reward engineering.

## 2 Related Work

Automated reasoning on KGs has been a long-standing task for natural language processing. In recent years, various embedding-based methods using tensor factorization (Nickel et al., 2011; Bordes et al., 2013; Riedel et al., 2013; Yang et al., 2014; Trouillon et al., 2017) or neural network models (Socher et al., 2013) have been developed, where they learn a projection which maps the triples into a continuous vector space for further tensor operations. Despite the impressive results they achieved, most of them lack the ability to capture chains of multi-hop reasoning patterns contained in paths.

To address the limitation of the embedding-based methods, a series of path-based methods have been proposed, which consider the selected relational paths as reasoning evidence. Lao et al. (2011) propose the Path-Ranking Algorithm (PRA) which uses random walks for path-finding. Gardner et al. (2013, 2014) propose a variation on PRA which computes feature similarity in the vector space. To combine the embedding-based methods, several neural multi-hop models (Neelakantan et al., 2015; Guu et al., 2015; Toutanova et al., 2015, 2016; Das et al., 2017) are proposed which perform a hybrid reasoning. Nevertheless, the evidential paths they used are gathered by random walks, which might be inferior and noisy.

Recently, DeepPath (Xiong et al., 2017) and MINERVA (Das et al., 2018) were proposed to address the problem above by using reinforcement learning, where they are committed to learn a policy which guides the agent to find more superior evidential paths to maximize the expected reward. Specifically, DeepPath parameterizes its policy with a fully-connected neural network and uses manual reward criteria, including global accuracy, efficiency and diversity, to evaluate the path quality. In the training phase, DeepPath applies the linear combination of these criteria as the positive reward while using a hand-craft constant as the negative penalty. As for MINERVA, it parameterizes its policy with a long short-term memory network (LSTM) and considers the path validity as the only reward criterion. In the training phase, MINERVA uses boolean value as a terminal signal to evaluate whether the current path reaches the target entity and manually tunes a moving average of cumulative discounted reward on different datasets for variance reduction.

## 3 Preliminaries

### 3.1 Knowledge Graph Reasoning

Given an incomplete knowledge graph $\mathcal{G} = \{(h, r, t)|h \in \mathcal{E}, t \in \mathcal{E}, r \in \mathcal{R}\}$, where $\mathcal{E}$ and $\mathcal{R}$ denote the entity set and relation set, respectively. There are two main tasks in knowledge graph reasoning, namely link prediction and fact prediction. Link prediction involves inferring the tail entity $t$ given the head entity $h$ and the query relation $r_q$, while fact prediction seeks to predict whether an unknown fact $(h, r_q, t)$ holds or not. Recently, RL-based reasoning has become a popular approach for knowledge graph reasoning, which achieves state-of-the-art performance. In general, RL-based reasoning methods strive to find relational paths to tune their reasoning policies for predictions and formulate the path-finding problem as a Markov decision process (MDP). In such a process, the policy-based agent decides to take an action $a \in \mathcal{A}$ from the current state (i.e., the current entity and its context information) $s \in \mathcal{S}$ to reach the next one according to its reasoning policy $\pi$, where the action space is defined as all the relations in $\mathcal{G}$. In particular, each relational chain in the relational paths can be considered as a reasoning chain.

### 3.2 Imitation Learning

Imitation learning focuses on learning policies from demonstrations, which has achieved great success in solving reward engineering. The classical approach is to find the optimal reward function by inverse reinforcement learning (IRL) (Russell, 1998; Ng et al., 2000) to explain expert behaviors. However, IRL requires solving RL inside a learning loop, which can be expensive to run in large environments. Therefore, generative adversarial imitation learning (GAIL) (Ho and Ermon, 2016) has recently been proposed, which learns the expert policy with generative adversarial network (GAN) (Goodfellow et al., 2014), eliminating any intermediate IRL steps.

In GAIL, a generator $G_\theta$ is trained to generate trajectories matching the distribution of expert trajectories (i.e., demonstrations). Each trajectory $\tau$ is represented as a state-action sequence $[(s_t, a_t)]_{t=0}^{\infty}$ $(s_t \in \mathcal{S}, a_t \in \mathcal{A})$. In addition, a discriminator $D_\omega$ is learned to distinguish between the generated policy $\pi_\theta$ and expert policy $\pi_E$. For each training epoch, the discriminator is updated first with the gradient as:

$$\mathbb{E}_\tau[\nabla_\omega log(D(s,a))] + \mathbb{E}_{\tau_E}[\nabla_\omega log(1 - D(s,a))], \quad (1)$$

where $\tau_E$ denotes the expert trajectories generated by $\pi_E$ and the trajectory expectation can be calculated in the $\gamma$-discounted infinite horizon as:

$$\mathbb{E}_\tau[D(s,a)] = \sum_{t=0}^{\infty} \mathbb{E}[\gamma^t D(s_t, a_t)], \quad (2)$$

where the discriminator here can be interpreted as a local reward function to provide feedback for the policy learning process. Then, the generator is updated with the cost function $\log(D(s,a))$ using the trust region policy optimization (TRPO) (Schulman et al., 2015). After sufficient adversarial training, the optimal policy $\hat{\pi}$ can be found by GAIL to rationalize the expert policy $\pi_E$.

## 4 Methodology

### 4.1 Framework Overview

As shown in Figure 1, our framework DIVINE consists of two modules, namely a generative adversarial reasoner and a demonstration sampler. In particular, the reasoner is composed of a generator and a discriminator. The generator can be any of the policy-based agents in existing RL-based methods and the discriminator can be interpreted as a self-adaptive reward function. For each query relation, the sampler and the generator are adopted respectively to automatically extract demonstrations and generate relational paths from the given KG. The discriminator is then used to evaluate the semantic similarity between the generated paths and demonstrations to update the generator.

After sufficient adversarial training between the generator and the discriminator alternatively, the well-trained policy-based agent (i.e., generator) can be used to find evidential paths matching the distribution of the demonstrations and make predictions by synthesizing these evidential paths.

### 4.2 Generative Adversarial Reasoner

In our framework, the reasoner is learned from the demonstrations through generative adversarial training. A straightforward approach is to directly apply GAIL to train the reasoner. In particular, the policy-based agent in the reasoner is trained to find evidential paths by imitating the state-action pairs in each expert trajectory (i.e., demonstration). However, such an approach may lead to

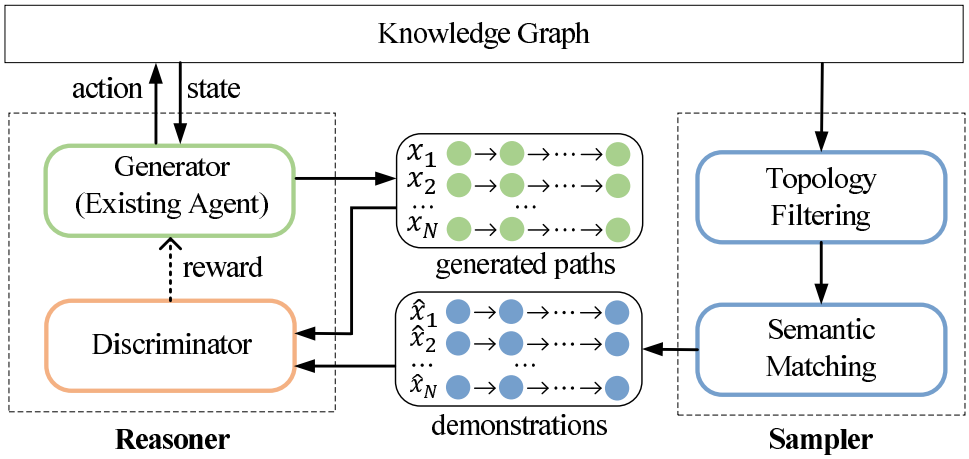


Figure 1: Overview of our framework DIVINE, where the generator can be any of the existing policy-based agent for knowledge graph reasoning and the discriminator can be considered as a self-adaptive reward function.

poor performance. The main reason lies in that the agent will tend to choose the same actions as in the expert trajectories under certain states, while ignoring many valuable evidential paths which are semantically similar to the expert trajectories but contain different reasoning chains.

Therefore, to encourage the agent to find more diverse evidential paths, it is desirable to train the agent by imitating each trajectory instead of each of its state-action pairs. In addition, in the scenario of knowledge graph reasoning, since the reasoning chains consist of only relations, the demonstrations do not necessarily contain the state information. In other words, the demonstrations can be composed of only relational paths.

Based on the above analysis, we propose a path-based GAIL method, where the reasoning policy is learned by imitating the path-level semantic features of the demonstrations which are composed of only relational paths.

In what follows, we first describe the two components of the reasoner, i.e., the generator and the discriminator. Then, we show how to extract the path-level semantic features.

### Generator

The generator can be any of the policy-based agent in existing RL-based methods. We strive to enable the generator to find more diverse evidential paths matching the distribution of the demonstrations in the semantic space.

### Discriminator

To better semantically distinguish between generated paths and demonstrations, we choose the convolutional neural networks (CNNs) to construct our discriminator $D$, as CNNs have shown high performance in semantic feature extraction from natural languages (Kim, 2014).

### Semantic Feature Extraction

For each positive entity pair, we respectively pack the current generated paths and corresponding demonstrations in the same package form. For each package $\mathcal{P} = \{x_1, x_2, ..., x_N\}$ containing $N$ relational paths, we encode the package to a real-valued matrix as:

$$\boldsymbol{p} = \boldsymbol{x}_1 \oplus \boldsymbol{x}_2 \oplus ... \oplus \boldsymbol{x}_N, \tag{3}$$

where $\boldsymbol{x}_n \in \mathbb{R}^k$ is the $k$-dimensional path embedding and $\oplus$ denotes the concatenation operator for the package representation $\boldsymbol{p} \in \mathbb{R}^{N \times k}$. In particular, given a relational path $x = \{r_1, r_2, ...r_t, ...\}$, the path embedding $\boldsymbol{x}$ is encoded as:

$$\boldsymbol{x} = \sum_{r_t \in x} \boldsymbol{r}_t, \tag{4}$$

where each relation $r_t$ is mapped into a real-valued embedding $\boldsymbol{r}_t \in \mathbb{R}^k$ pre-trained by TransE (Bordes et al., 2013).

After packing, we feed the package representation $\boldsymbol{p}$ into our discriminator $D$ to parameterize its semantic features $D(\boldsymbol{p})$. Specifically, a convolutional layer activated by ReLU nonlinearity is first used to extract local features by sliding a kernel $\boldsymbol{\omega} \in \mathbb{R}^{h \times l}$ for a new feature map:

$$\boldsymbol{c} = ReLU(Conv(\boldsymbol{p}, \boldsymbol{\omega}) + \boldsymbol{b}_c), \tag{5}$$

where $\boldsymbol{b}_c$ denotes the bias term. Then, a fully-connected hidden layer and an output layer are used for further semantic feature extraction:

$$D(\boldsymbol{p}) = \sigma(\boldsymbol{W}_2 ReLU(\boldsymbol{W}_1 \boldsymbol{c})), \tag{6}$$

where the corresponding biases are not shown above for brevity and the output layer is normalized by a sigmoid function while other layers are activated by ReLU nonlinearity.

### 4.3 Demonstration Sampler

For imitation learning, the first prerequisite is to have high-quality demonstrations. However, due to the large scale and complexity of KGs, manually constructing a large number of reasoning demonstrations requires considerable time and expert efforts. Therefore, we design an automated sampler to sample reliable reasoning demonstrations from KGs without supervision and extra manual labor.

#### Static Demo Sampling

For each query relation, we use all positive entity pairs to sample demonstration candidates from the given KG. Specifically, for each positive entity pair, we use bi-directional breadth-first search to explore the shortest path between two entities. In particular, since the shorter paths incline to characterize more direct correlations between two entities, we prefer to use them for initialization to ensure the quality of demonstration candidates. As for the longer paths, despite their potential utility values, they are more likely to contain worthless or even noisy inference steps, thus we learn them only in the training phase. In doing so, we can get a demonstration set $\Omega_E$ which contains all the candidates we sampled. Finally, to accommodate the fixed input dimension of the discriminator $D$, we can simply select a subset $P_e \subseteq \Omega_E$ with the top $N$ occurrence frequency, where $N$ is normally much smaller than $|\Omega_E|$.

#### Dynamic Demo Sampling

Despite the simplicity of the static demo sampling method, the obtained demonstrations by this method are fixed and ignore the specific environment of each entity in the given KG. Therefore, we propose an improving method to dynamically sample demonstrations by taking the topological correlations of entities into consideration.

Given a positive entity pair $\langle e_{head}, e_{tail} \rangle$, we introduce a relational set $\mathcal{R}_h$ which contains all relations directly connected to $e_{head}$. For each reasoning attempt, $\mathcal{R}_h$ can be considered as the region of interest (ROI) of the agent to start reasoning, where the ROI related paths tend to be more relevant to the current entity pair. Thus, we refine the demonstration set by filtering out the demonstrations which begin from $\mathcal{R}_h$:

$$\Omega'_E = \{x \mid r_1(x) \in \mathcal{R}_h, x \in \Omega_E\} \ (\mathcal{R}_h \neq \varnothing), \quad (7)$$

where $\Omega_E$ is generated by the static demo sampling method and $r_1(x)$ denotes the first relation in relational path $x = \{r_1, r_2, ...r_t, ...\}$.

In most cases, we can obtain enough demonstrations in $\Omega'_E$ to select a subset $P_e \subseteq \Omega'_E$ in the same way as the static demo sampling method. However, due to the sparsity of data in KG, we may get insufficient demonstrations on long-tail entities. To solve this problem, we perform semantic matching to explore more demonstrations from the remaining candidates $\mathcal{C}_E = \Omega_E \setminus \Omega'_E$. Since the reasoning policy is updated based on the semantic similarities between the generated paths and demonstrations, candidates which are semantically similar to the current demonstrations are also instructive for the imitation process.

Inspired by the neighborhood attention for one-shot imitation learning (Duan et al., 2017), we use each demonstration in $\Omega'_E$ to query other candidates in correlation to itself. We adopt the dot product to measure the semantic matching similarity between two path embeddings:

$$\alpha_i = \sum_{x_j \in \Omega'_E} \bar{\boldsymbol{x}}_i \cdot \boldsymbol{x}_j \quad i = 1, 2, ..., |\mathcal{C}_E|, \quad (8)$$

where $\alpha_i$ represents the sum of matching scores between the current candidate $\bar{x}_i$ and existing demonstrations in $\Omega'_E$. Finally, we iteratively select the candidate with the highest $\alpha$ to pad the refined demonstration set $\Omega'_E$ until accommodating the required input dimension $N$.

### 4.4 Training

In the training phase, all the positive entity pairs are used to generate demonstration candidates $\Omega_E$ for the imitation learning process. Specifically, for each positive entity pair, the demonstration sampler is required first to choose the corresponding demonstrations, while the generator is conducted to generate some relational paths. Then, the demonstrations are packed into package $\mathcal{P}_e$ and the generated paths are packed into different packages $\{\mathcal{P}_g \mid \mathcal{P}_g \subseteq \Omega_G\}$ according to their validity, i.e., whether the agent can reach the target entity along the current path, where $\Omega_G$ is the collection of all generated paths.

For each package pair $\langle \mathcal{P}_g, \mathcal{P}_e \rangle$, we train the discriminator $D$ by minimizing its loss and expect it to be expert in distinguishing between $\mathcal{P}_e$ and $\mathcal{P}_g$. In addition, to make the adversarial training process more stable and effective, we adopt the loss function proposed in WGAN-GP (Gulrajani et al., 2017) to update the discriminator:

$$\begin{aligned} \mathcal{L}_C &= \mathbb{E}[D(\boldsymbol{p}_g)] - \mathbb{E}[D(\boldsymbol{p}_e)]; \\ \mathcal{L}_P &= \lambda \mathbb{E}[(\parallel \nabla_{\tilde{\boldsymbol{p}}} D(\tilde{\boldsymbol{p}}) \parallel_2 -1)^2]; \\ \mathcal{L}_D &= \mathcal{L}_C + \mathcal{L}_P, \end{aligned} \tag{9}$$

where $\mathcal{L}_C$ , $\mathcal{L}_P$ and $\mathcal{L}_D$ respectively denote the original critic loss, gradient penalty and the loss of discriminator, $\lambda$ is the gradient penalty coefficient and $\tilde{\boldsymbol{p}}$ is sampled uniformly along straight lines between $\boldsymbol{p}_g$ and $\boldsymbol{p}_e$. According to the feedback of the discriminator, we calculate the reward $\mathcal{R}_G$ as:

$$\mathcal{R}_G = \delta_g \max\{\mathbb{E}[D(\boldsymbol{p}_g)] - \mathbb{E}[D(\boldsymbol{p}_n)], 0\}, \tag{10}$$

$$\delta_g = \begin{cases} 1, & \mathcal{P}_g \subseteq \Omega_G^+ \\ 0, & otherwise \end{cases}, \tag{11}$$

where $\boldsymbol{p}_n$ denotes a noise embedding composed of random noise with continuous uniform distribution, $\delta_g$ is a characteristic function which characterizes the validity of package $\mathcal{P}_g$, $\Omega_G^+$ is the collection of all valid generated paths. We only give positive rewards for partial valid paths that at least have higher expectations than noise embedding $\boldsymbol{p}_n$, which filters out the paths of inferior quality to improve the convergence efficiency of the training process. Once the reward obtained, we updated the generator $G$ by maximizing the expected cumulative reward with Monte-Carlo Policy Gradient (i.e., REINFORCE) (Williams, 1992).

We use mini-batch stochastic gradient descent (SGD) to optimize the loss function of discriminator, while the generator is updated with the Adam algorithm (Kingma and Ba, 2014).

# 5 Experiments

## 5.1 Datasets and Evaluation Metrics

| Dataset | # Ent. | # Rel. | # Facts | # Tasks |
|---|---|---|---|---|
| NELL-995 | 75,492 | 200 | 154,213 | 12 |
| FB15K-237 | 14,505 | 237 | 310,116 | 20 |

Table 1: Statistics of the datasets. # Ent. denotes the number of unique entities and # Rel. denotes the number of relations.

The experiments are conducted on two benchmark datasets: NELL-995 (Xiong et al., 2017) and FB15K-237 (Toutanova et al., 2015). The details of the two datasets are described in Table 1. In particular, NELL-995, which is known to be a simple dataset for reasoning tasks, is generated from the 995th iteration of the NELL system (Carlson et al., 2010) by selecting the triples with Top-200 frequently occurring relations. Compared to NELL-995, FB15K-237 is more challenging and closer to real-world scenarios, where its facts are created from FB15K (Bordes et al., 2013) with redundant relations removed. For each triple $(h, r, t)$, both datasets contain the inverse triple $(h, r^{-1}, t)$ such that the agent can step backward in KGs, which makes it possible to recover from a potentially wrong decision that has been taken before. For each reasoning task with a query relation $r_q$, all the triples with $r_q$ or $r_q^{-1}$ are removed from the KG and split into train and test samples.

Similar to recent works (Das et al., 2018; Xiong et al., 2017), we use mean average precision (MAP), mean reciprocal rank (MRR) and Hits@k to evaluate the reasoning performance, where Hits@k is the fraction of positive instances ranked in the top $k$ positions.

## 5.2 Baselines and Implementation Details

In our experiments, we consider two state-of-the-art RL-based methods as baselines: DeepPath (Xiong et al., 2017) and MINERVA (Das et al., 2018). Deep path feeds the gathered evidential paths to PRA (Lao et al., 2011) for both link prediction and fact prediction tasks, while MINERVA directly applies the well-trained agent to the link prediction task for question answering. For DeepPath, we use the code released by Xiong et al. (2017). For MINERVA, we use the code released by Das et al. (2018). The experiment settings for the baselines are set according to the suggestions in the original papers.

In the implementation of our framework, we set the path number $N$ to 5 for each path package $\mathcal{P}$, while the path dimension $k$ is set to 200 which is the same as the relation dimension in baselines. For the discriminator, we set the convolution kernel size to $3 \times 5$, the hidden layer size to 1024, and the output layer size to the path dimension $k$, while the gradient penalty coefficient $\lambda$ is set to 5 and L2 regularization is also used to avoid over-fitting.

During testing, we also rank the answer triples

| Data | Metric | Link Prediction | | | | Fact Prediction | |
|---|---|---|---|---|---|---|---|
| | | **DeepPath** | **Div(DeepPath)** | **MINERVA** | **Div(MINERVA)** | **DeepPath** | **Div(DeepPath)** |
| NELL-995 | Hits@1 | 0.718 | 0.749 | 0.799/0.663† | 0.840/0.668† | 0.687 | 0.701 |
| | Hits@3 | 0.847 | 0.878 | 0.936/0.773† | 0.936/0.778† | 0.839 | 0.841 |
| | MRR | 0.784 | 0.811 | 0.852/0.725† | 0.874/0.731† | 0.764 | 0.774 |
| | MAP | 0.796 | 0.821 | 0.885 | 0.893 | 0.774/0.493‡ | 0.785/0.494‡ |
| FB15K-237 | Hits@1 | 0.417 | 0.520 | 0.427/0.217† | 0.444/0.223† | 0.394 | 0.421 |
| | Hits@3 | 0.646 | 0.740 | 0.582/0.329† | 0.590/0.331† | 0.565 | 0.599 |
| | MRR | 0.560 | 0.639 | 0.546/0.293† | 0.556/0.296† | 0.522 | 0.545 |
| | MAP | 0.572 | 0.658 | 0.553 | 0.568 | 0.536/0.311‡ | 0.558/0.339‡ |

Table 2: Overall results on NELL-995 and FB15K-237. "†" denotes the results with settings for question answering and "‡" denotes the results of directly ranking all the positive and negative triples given a query relation.

against the negative triples used in DeepPath and MINERVA. In particular, there are approximately 10 corresponding negative triples for each positive ones. Each negative triple is generated by replacing the answer entity $t$ with a faked one $t'$ given a positive triple $(h, r, t)$.

### 5.3 Results

The main results on the two datasets are shown in Table 2. We use "Div(*)" to denote the RL-based method "*" which adopts our framework DIVINE. For a fair comparison, we follow MINERVA to report the Hits@k and MRR scores (denoted with "†") to evaluate the link prediction performance for question answering, which ranks entities according to the probability that the agent can reach the entity along evidential paths. Moreover, we also follow DeepPath to report the MAP scores (denoted with "‡") on the fact prediction task, which directly ranks all the positive and negative triples for a given query relation.

From the results shown in Table 2, we can observe that our framework produces consistent improvements for the two RL-based methods under varying degrees on both link prediction and fact prediction tasks. On the one hand, for existing RL-based methods, their results on FB15K-237 are generally lower than those on NELL-995 since FB15K-237 is more complex and arguably more difficult to design proper reward functions manually. However, our framework reliefs this problem to some extent by dynamically learning superior reward functions, thus we make greater improvements on challenging FB15K-237. On the other hand, for different datasets, the improvements our framework makes for DeepPath vary a lot while MINERVA not. This is because MINERVA manually adjusts its hyper-parameters accordingly when calculating cumulative discounted reward, while DeepPath keeps the same. Obviously, it validates the necessity for existing RL-based methods to adjust their reward functions accordingly to fit different datasets. Enhanced by our framework, these RL-based methods no longer require additional manual adjustments for different datasets, which reveals great robustness.

Similar to existing RL-based methods, we also report the decomposed results of link prediction and use MAP to evaluate the performance for each query relation on NELL-995 in Table 3. From the results, we can observe that the results on different relations are of high variance and the enhanced RL-based methods achieve better or comparable performance for all query relations.

### 5.4 Effectiveness of the Path-Based GAIL Method

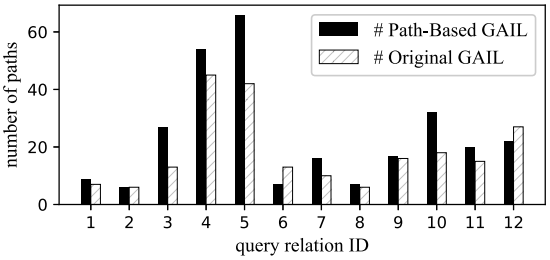


Figure 2: Statistics of path-finding on NELL-995.

| | Path Avg. | MAP | |
|---|---|---|---|
| | | # Link | # Fact |
| # Path-Based GAIL | 24 | 0.821 | 0.494 |
| # Original GAIL | 18 | 0.814 | 0.468 |

Table 4: Results of different imitation learning settings.

To investigate the effectiveness of the path-based GAIL method, we train the policy-based agent in DeepPath on NELL-995 by our path-based GAIL method and the original GAIL method, respectively. In particular, in the process of training the a-

| (ID) Query Relations | DeepPath | Div(DeepPath) | MINERVA | Div(MINERVA) |
|---|---|---|---|---|
| (01) AgentBelongsToOrg | 0.576 | **0.674** | 0.860 | **0.876** |
| (02) AthleteHomeStadium | **0.890** | 0.877 | 0.895 | **0.916** |
| (03) AthletePlaysForteam | 0.750 | **0.779** | **0.824** | 0.813 |
| (04) AthletePlaysInLeague | 0.960 | **0.962** | **0.970** | 0.965 |
| (05) AthletePlaysSport | 0.957 | **0.960** | 0.985 | **0.986** |
| (06) OrgHeadquarteredInCity | 0.790 | **0.801** | **0.946** | 0.936 |
| (07) OrgHiredPerson | 0.742 | **0.758** | 0.851 | **0.860** |
| (08) PersonBornInLocation | 0.757 | **0.780** | 0.793 | **0.827** |
| (09) PersonLeadsOrg | 0.794 | **0.810** | 0.851 | **0.881** |
| (10) TeampPlaysInLeague | 0.881 | **0.896** | **0.970** | 0.957 |
| (11) TeampPlaysSport | 0.739 | **0.816** | 0.846 | **0.874** |
| (12) WorksFor | 0.711 | **0.741** | **0.825** | 0.822 |
| Overall | 0.796 | **0.821** | 0.885 | **0.893** |

Table 3: Results of link prediction decomposed over different query relations on NELL-995.

gent by the original GAIL method, the demonstrations are composed of state-action trajectories. For each state-action pair $(s_t, a_t)$, the state representation $\boldsymbol{s}_t$ is calculated by $(\boldsymbol{e}_t, \boldsymbol{e}_{tail} - \boldsymbol{e}_t)$, where $\boldsymbol{e}_t$ and $\boldsymbol{e}_{tail}$ denote the embeddings of the current entity and the tail entity, respectively. In Figure 2, we show the statistics of the evidential path set $P_{new}$ which are found by the agent and different from the demonstrations. In Table 4, we compare the average path number of $P_{new}$ and the reasoning performance on the two prediction tasks.

As shown in Figure 2 and Table 4, we can observe that our path-based GAIL method obtains more evidential paths for most query relations and achieves better performance on both link and fact predictions, which validates the effectiveness of our path-based GAIL method and the rationality of encouraging the agent to find more diverse evidential paths.

### 5.5 Ablation Studies

We conduct ablation studies by embedding DeepPath into our framework to quantify the role of components. Specifically, we re-train our framework by ablating certain components:

- W/O Semantic Matching, where no semantic matching is performed on long-tail entities. Instead, we directly extract some paths from the remaining demonstration candidates $\mathcal{C}_E$ according to their occurrence frequency.
- W/O Dynamic Sampling, where no dynamic demo sampling is performed to incorporate the local environment of entities. In other words, we only adopt the static demo sampling method to obtain demonstrations.
- W/O Demo Sampling, where no demonstration is used for imitation learning, which degenerates to DeepPath.

| | MAP | |
|---|---|---|
| Configuration | NELL-995 | FB15K-237 |
| Div(DeepPath) | 0.821 | 0.658 |
| W/O Semantic Matching | 0.818 | 0.651 |
| W/O Dynamic Sampling | 0.806 | 0.640 |
| W/O Demo Sampling | 0.796 | 0.572 |

Table 5: Ablation on different components.

We use MAP to evaluate the link prediction performance on both NELL-995 and FB15K-237 in Table 5. From the results, we can observe that: (1) Based on imitation learning, our framework can effectively improve the reasoning performance, even if we use the static demo sampling method to obtain demonstrations; (2) High-quality demonstrations are crucial for imitation learning, which indicates both topology filtering and semantic matching play important roles in the demonstration sampler of our framework.

## 6 Conclusion

In this paper, we proposed a novel plug-and-play framework DIVINE for knowledge graph reasoning based on generative adversarial imitation learning, which enables existing RL-based methods to learn reasoning policies and reward functions self-adaptively to adapt the fast evolutions of real-world KGs. The experimental results show that our framework improves the performance of existing RL-based methods while eliminating extra reward engineering.

## Acknowledgments

This work was supported by National Natural Science Foundation of China under Grant 61872045. We acknowledge anonymous reviewers for their valuable comments.

## References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems 26*, pages 2787–2795.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka Jr, and Tom M Mitchell. 2010. Toward an architecture for never-ending language learning. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, pages 1306–1313.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2018. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *International Conference on Learning Representations*.

Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. 2017. Chains of reasoning over entities, relations, and text using recurrent neural networks. pages 132–141.

Yan Duan, Marcin Andrychowicz, Bradly Stadie, OpenAI Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. 2017. One-shot imitation learning. In *Advances in Neural Information Processing Systems 30*, pages 1087–1098.

Matt Gardner, Partha Talukdar, Jayant Krishnamurthy, and Tom Mitchell. 2014. Incorporating vector space similarity in random walk inference over knowledge bases. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 397–406.

Matt Gardner, Partha Pratim Talukdar, Bryan Kisiel, and Tom Mitchell. 2013. Improving learning and inference in a large knowledge-base using latent syntactic cues. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 833–838.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*, pages 2672–2680.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. 2017. Improved training of wasserstein gans. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 5767–5777.

Kelvin Guu, John Miller, and Percy Liang. 2015. Traversing knowledge graphs in vector space. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 318–327.

Jonathan Ho and Stefano Ermon. 2016. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems 29*, pages 4565–4573.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ni Lao, Tom Mitchell, and William W Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 529–539.

Bonan Min, Ralph Grishman, Li Wan, Chang Wang, and David Gondek. 2013. Distant supervision for relation extraction with an incomplete knowledge base. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 777–782.

Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 2015. Compositional vector space models for knowledge base inference. In *2015 aaai spring symposium series*.

Andrew Y Ng, Stuart J Russell, et al. 2000. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 663–670.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, volume 11, pages 809–816.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84.

Stuart Russell. 1998. Learning agents for uncertain environments. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 101–103.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897.

Baoxu Shi and Tim Weninger. 2018. Open-world knowledge graph completion. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems 26*, pages 926–934.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509.

Kristina Toutanova, Victoria Lin, Wen-tau Yih, Hoifung Poon, and Chris Quirk. 2016. Compositional learning of embeddings for relation paths in knowledge base and text. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1434–1444.

Théo Trouillon, Christopher R Dance, Éric Gaussier, Johannes Welbl, Sebastian Riedel, and Guillaume Bouchard. 2017. Knowledge graph completion via complex tensor factorization. *The Journal of Machine Learning Research*, 18(1):4735–4772.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 564–573.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*.

# Original Semantics-Oriented Attention and Deep Fusion Network for Sentence Matching

**Mingtong Liu, Yujie Zhang, Jinan Xu, Yufeng Chen**
School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China
{16112075,yjzhang,jaxu,chenyf}@bjtu.edu.cn

## Abstract

Sentence matching is a key issue in natural language inference and paraphrase identification. Despite the recent progress on multi-layered neural network with cross sentence attention, one sentence learns attention to the intermediate representations of another sentence, which are propagated from preceding layers and therefore are uncertain and unstable for matching, particularly at the risk of error propagation. In this paper, we present an original semantics-oriented attention and deep fusion network (OSOA-DFN) for sentence matching. Unlike existing models, each attention layer of OSOA-DFN is oriented to the original semantic representation of another sentence, which captures the relevant information from a fixed matching target. The multiple attention layers allow one sentence to repeatedly read the important information of another sentence for better matching. We then additionally design deep fusion to propagate the attention information at each matching layer. At last, we introduce a self-attention mechanism to capture global context to enhance attention-aware representation within each sentence. Experiment results on three sentence matching benchmark datasets SNLI, SciTail and Quora show that OSOA-DFN has the ability to model sentence matching more precisely.

## 1 Introduction

Natural language sentence matching is a key technique of comparing two sentences and identifying the semantic relationship between them, which is usually viewed as a classification problem (Wang et al., 2017). The technique has applications in natural language inference to judge whether a hypothesis sentence can be inferred from a premise sentence (Bowman et al., 2015) and in paraphrase identification to determine whether two sentences express the equivalent meaning or not (Yin et al., 2015). The core issue for sentence matching is to model the relatedness between two sentences (Rocktäschel et al., 2015; Parikh et al., 2016; Wang et al., 2017; Duan et al., 2018).

Recently, neural network-based models for sentence matching have attracted more attention for their powerful ability to learn sentence representation (Bowman et al., 2015; Wang et al., 2017; Duan et al., 2018). There are mainly two types of frameworks: sentence encoding based framework and attention-based framework. For the first type of framework, a simple and effective model is proposed by using two sentence vectors (Bowman et al., 2015), but the interaction between two sentences is neglected. For the second type of framework, attention mechanism is introduced to model word-level interaction between two sentences and a higher accuracy is achieved (Rocktäschel et al., 2015; Parikh et al., 2016; Wang et al., 2017). Particularly, multi-layered deep matching network with attention shows that deeper models outperform shallower models (Duan et al., 2018).

However, the existing attention mechanism still has some limitations. When one sentence learns attention to another sentence, the attention is performed between two parallel layers and oriented to the intermediate representations from the preceding layer of another one. As a result, semantics to be paid attention are uncertain and unstable for matching because semantics are changed at different layers. On the other hand, the intermediate representations tend to be affected by error propagation in multi-layered attentions, in which if the first attention aligns the wrong position, the second attention will now have the incorrect information as input for alignment.

In order to address these problems, we propose an original semantics-oriented attention and deep fusion network (OSOA-DFN) for sentence matching. OSOA-DFN mainly consists of three sub-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2652–2661, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

components: (1) original semantics-oriented cross sentence attention; (2) deep fusion; and (3) self-attention mechanism. The cross sentence attention is oriented to the original semantic representations of another sentence, so as to be able to capture inherent semantics by relying on the fixed matching target. The multiple cross attention operations allow one sentence to repeatedly read the important information of another sentence for better interaction. We then design a deep fusion in addition to usual fusion to augment the propagation of attention information at each matching layer. The self-attention mechanism is also introduced at the last to capture global context to enhance attention-aware representation within each sentence. Experiment results demonstrate that OSOA-DFN has the ability to model sentence matching more precisely on the SNLI, the SciTail, and the Quora datasets.

Our contributions can be summarized as follows:

- We pay attention to the original semantic representations for cross sentence interaction and the matching target of attention for a certain sentence is therefore ensured to be fixed in spite of multiple layers. The multiple cross attention operations allow one sentence to repeatedly read the important information of another sentence for better interaction.
- We design a deep fusion in addition to usual fusion to augment the propagation of attention information for matching, and introduce a self-attention mechanism at the last to capture global context to enhance attention-aware representation within each sentence.
- We evaluate our model on three challenging datasets and show that the proposed model has the ability to model sentence matching more precisely and significantly improves the performance.

## 2 General Neural Attention-Based Model for Sentence Matching

Formally, we can define the sentence matching as follows. Given two sentences $P = [p_1, \cdots, p_i, \cdots, p_m]$ and $Q = [q_1, \cdots, q_j, \cdots, q_n]$, the goal is to predict a label $y^* \in \mathcal{Y}$, where $\mathcal{Y}$ = {entailment, contradiction, neutral} in natural language inference and $\mathcal{Y}$ = {0,1} in paraphrase identification, indicating the logic semantic relationship between two sentences $P$ and $Q$ (Wang et al., 2017).

$$y^* = \arg\max_{y \in \mathcal{Y}} P_{\mathrm{r}}(\mathrm{y}|P, Q) \quad (1)$$

Generally, the architecture of neural attention-based models for sentence matching includes three components (Wang et al., 2017; Duan et al., 2018): (1) **input encoding layer** encodes each sentence into semantic representation; (2) **attention-based matching layer** models word-level alignment between two sentences and produces attention-aware representation for each sentence; and (3) **prediction layer** predicts the semantic relation between two sentences. Figure 1(a) illustrates the general model.

### 2.1 Input Encoding Layer

For the given sentence pairs $P = [p_1, \cdots, p_i, \cdots, p_m]$ and $Q = [q_1, \cdots, q_j, \cdots, q_n]$, where $p_i$ and $q_j$ indicate the $i$-th and $j$-th word in $P$ and $Q$ respectively, the input encoding layer first converts words of $P$ and $Q$ into vectors $[\mathbf{e}_{p_1}, \cdots, \mathbf{e}_{p_i}, \cdots, \mathbf{e}_{p_m}]$ and $[\mathbf{e}_{q_1}, \cdots, \mathbf{e}_{q_j}, \cdots, \mathbf{e}_{q_n}]$ by looking up $M$ respectively, where $M \in \mathbf{R}^{d \times |V|}$ is the embedding table. $d$ is the dimension of embeddings and $|V|$ is the size of the vocabulary.

In order to encode contextual information into word representations, we use a BiLSTM neural network (Hochreiter and Schmidhuber, 1997) to encode two sentences $P$ and $Q$. The sequential BiLSTM calculates a new hidden state conditioned on the previous states to incorporate contextual information, and several previous works have shown its effectiveness for sentence matching (Rocktäschel et al., 2015; Wang et al., 2017; Duan et al., 2018).

$$\mathbf{h}^0_{p_i} = \mathrm{BiLSTM}(\mathbf{e}_{p_i}, \overrightarrow{\mathbf{h}}^0_{p_{i-1}}, \overleftarrow{\mathbf{h}}^0_{p_{i+1}}) \quad (2)$$

$$\mathbf{h}^0_{q_j} = \mathrm{BiLSTM}(\mathbf{e}_{q_j}, \overrightarrow{\mathbf{h}}^0_{q_{j-1}}, \overleftarrow{\mathbf{h}}^0_{q_{j+1}}) \quad (3)$$

Then the two sentences are converted to $\mathbf{H}^0_P = [\mathbf{h}^0_{p_1}, \cdots, \mathbf{h}^0_{p_i}, \cdots, \mathbf{h}^0_{p_m}]$ and $\mathbf{H}^0_Q = [\mathbf{h}^0_{q_1}, \cdots, \mathbf{h}^0_{q_j}, \cdots, \mathbf{h}^0_{q_n}]$. Hereafter, we call $\mathbf{H}^0_P$ and $\mathbf{H}^0_Q$ as original semantic representations of sentences $P$ and $Q$ respectively. In this paper, we will use them as the targets of cross sentence attention.

### 2.2 Attention-Based Matching Layer

Generally, this layer employs the attention mechanism to model the interaction information between

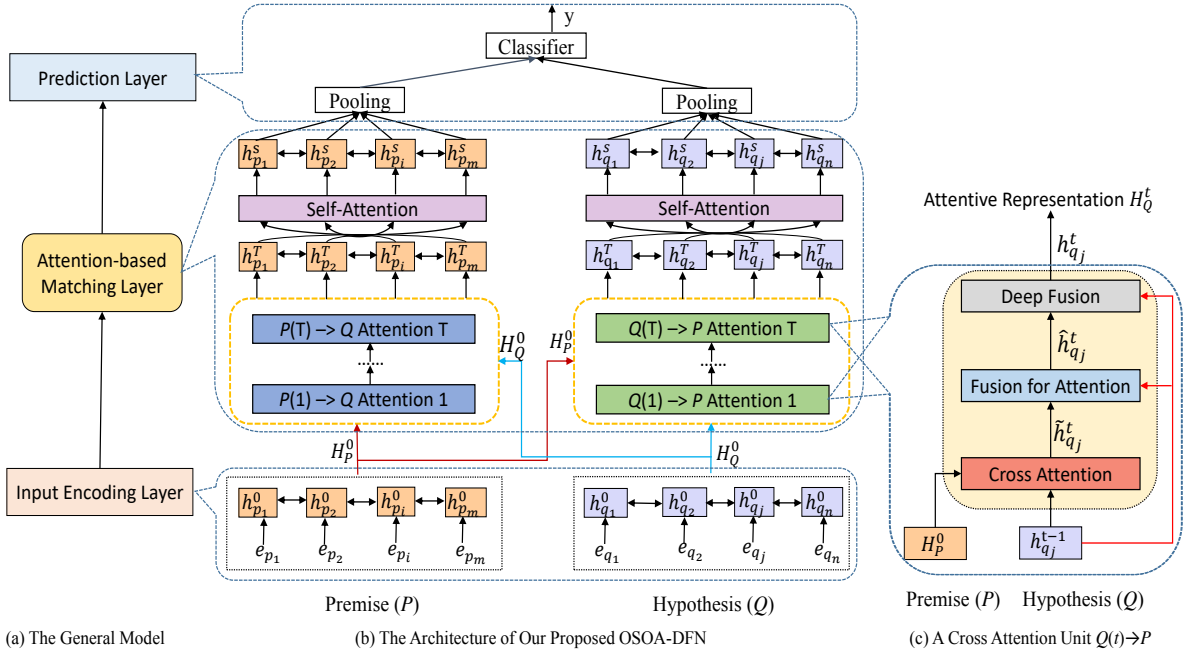


Figure 1: (a) is the general model for sentence matching. (b) is an overview architecture of our proposed OSOA-DFN. (c) is an original semantics-oriented cross attention unit that learns the interaction information from the original semantic representation of another sentence.

two sentences. It can be formulated as:

$$\mathbf{V}_P = \text{Match}(\mathbf{H}_P^0, \mathbf{H}_Q^0), \mathbf{V}_Q = \text{Match}(\mathbf{H}_Q^0, \mathbf{H}_P^0) \quad (4)$$

where Match$(\cdot)$ is a neural attention-based matching function, $\mathbf{V}_P = [\mathbf{v}_{p_1}, \cdots, \mathbf{v}_{p_i}, \cdots, \mathbf{v}_{p_m}]$ and $\mathbf{V}_Q = [\mathbf{v}_{q_1}, \cdots, \mathbf{v}_{q_j}, \cdots, \mathbf{v}_{q_n}]$ are new attention-aware representations for $P$ and $Q$, respectively. This layer is the core layer for sentence matching. Match$(\cdot)$ is mainly focused by researches and some effective frameworks are proposed (Rocktäschel et al., 2015; Wang et al., 2017; Duan et al., 2018). In this paper, we also focus on this layer, and propose an original semantics-oriented attention and deep fusion network. The details will be described in Section 3.

### 2.3 Prediction Layer

A pooling layer is used to convert the resulting representations of all position in $P$ and $Q$ into a fixed-length vector and feed it into a classifier to determine the semantic relationship between the two sentences.

A mean pooling is usually adopted on each sentence for capturing all of the information and also a max pooling for highlighting the significant properties. In this paper, we get a fixed dimensional representation $\mathbf{V}$ by concatenating them together as (Chen et al., 2017; Duan et al., 2018).

$$\mathbf{V}_{P_{mean}} = \frac{1}{m}\sum_{i=1}^{m}\mathbf{v}_{p_i}, \mathbf{V}_{P_{max}} = \max_{i=1}^{m}\mathbf{v}_{p_i} \quad (5)$$

$$\mathbf{V}_{Q_{mean}} = \frac{1}{n}\sum_{j=1}^{n}\mathbf{v}_{q_j}, \mathbf{V}_{Q_{max}} = \max_{j=1}^{n}\mathbf{v}_{q_j} \quad (6)$$

$$\mathbf{V} = [\mathbf{V}_{P_{mean}}; \mathbf{V}_{P_{max}}; \mathbf{V}_{Q_{mean}}; \mathbf{V}_{Q_{max}}] \quad (7)$$

Finally, we pass representation $\mathbf{V}$ into a multilayer perceptron (MLP) classifier to calculate the probability $P_r(\cdot)$ of each label.

$$P_r(\cdot|P,Q) = \text{softmax}(\mathbf{W}_2\text{ReLU}(\mathbf{W}_1\mathbf{V}+\mathbf{b}_1)+\mathbf{b}_2) \quad (8)$$

where, $\mathbf{W}_1$, $\mathbf{W}_2$, $\mathbf{b}_1$, $\mathbf{b}_2$ are learnable parameters.

## 3 Original Semantics-Oriented Attention and Deep Fusion Network

In this paper, we mainly focus on the structure of attention-based matching layer. Inspired by the recent successful deep models (He et al., 2016; Duan et al., 2018), we propose an original semantics-oriented attention and deep fusion network (OSOA-DFN) for sentence matching, as shown in Figure 1(b). OSOA-DFN is mainly composed of: (1) original semantics-oriented cross sentence attention; (2) deep fusion; and (3) self-attention mechanism. (1) and (2) are combined to

form one unit of cross attention, as shown in Figure 1(c), and there are $T$ units in attention-based matching layer. Finally, one layer of self-attention is introduced after the $T$ units of cross attention.

### 3.1 Original Semantics-Oriented Cross Sentence Attention

Cross sentence attention is utilized to model the relevance between two sentences. In the $t$-th attention layer, we use $P(t)\rightarrow Q$ to annotate that the sentence $P$ learns attention to the sentence $Q$ to extract the relevant information from $Q$.

Given the representations of $P$ and $Q$: $\mathbf{H}_P^{t-1} = [\mathbf{h}_{p_1}^{t-1}, \cdots, \mathbf{h}_{p_i}^{t-1}, \cdots, \mathbf{h}_{p_m}^{t-1}]$ and $\mathbf{H}_Q^0 = [\mathbf{h}_{q_1}^0, \cdots, \mathbf{h}_{q_j}^0, \cdots, \mathbf{h}_{q_n}^0]$, each cross attention $P(t)\rightarrow Q$ will use the original semantics $\mathbf{H}_Q^0$ of $Q$ for interaction, where $t = \{1, \cdots, T\}$ and $t = 1$ represents $P$ using the original representation $\mathbf{H}_P^0$. We first compute the unnormalized attention weights as the similarity of $P(t)$ and $Q$, the alignment matrix $\mathbf{A}^t \in \mathbf{R}^{m\times n}$ is defined as follows:

$$\mathbf{A}_{ij}^t = \mathbf{h}_{p_i}^{t-1^T}\mathbf{W}^t\mathbf{h}_{q_j}^0 + \langle \mathbf{U}_p^t, \mathbf{h}_{p_i}^{t-1}\rangle + \langle \mathbf{U}_q^t, \mathbf{h}_{q_j}^0\rangle \quad (9)$$

where $\mathbf{W}^t \in \mathbf{R}^{h\times h}$, $\mathbf{U}_p^t$, $\mathbf{U}_q^t \in \mathbf{R}^h$ are learnable parameters, and $\langle\cdot,\cdot\rangle$ denotes the inner production operation. $p_i$ and $p_j$ are the the $i$-th and $j$-th word in the $P$ and $Q$ respectively. Next, the semantics of sentence $Q$ related to $\mathbf{h}_{p_i}^{t-1}$ is extracted to compute $\widetilde{\mathbf{h}}_{p_i}^t$ according to $\mathbf{A}^t$, as shown in Equation (10).

$$\widetilde{\mathbf{h}}_{p_i}^t = \sum_{j=1}^{n} \frac{exp(\mathbf{A}_{ij}^t)}{\sum_{k=1}^{n} exp(\mathbf{A}_{ik}^t)}\mathbf{h}_{q_j}^0 \quad (10)$$

Intuitively, $\widetilde{\mathbf{h}}_{p_i}^t$ is a representation by using attentive information in $\mathbf{H}_Q^0$ that is softly aligned to $\mathbf{h}_{p_i}^{t-1}$, and the semantics of $\mathbf{H}_Q^0$ is more probably selected if it is more related to $\mathbf{h}_{p_i}^{t-1}$.

### 3.2 Deep Fusion

To further enrich the interaction, we first perform an usual fusion and then design a deep fusion for each cross attention to augment the propagation of attention information.

The usual fusion (Wang and Jiang, 2016a; Duan et al., 2018) can be formulated as the Equations (11) - (13).

$$\overline{\mathbf{h}}_{p_i}^t = [\mathbf{h}_{p_i}^{t-1}; \widetilde{\mathbf{h}}_{p_i}^t; |\, \mathbf{h}_{p_i}^{t-1} - \widetilde{\mathbf{h}}_{p_i}^t \,|; \mathbf{h}_{p_i}^{t-1} \odot \widetilde{\mathbf{h}}_{p_i}^t] \quad (11)$$

$$\widetilde{\mathbf{h}}_{p_i}^t = \mathrm{ReLU}(\mathbf{W}_{\mathrm{h}}^{\mathrm{t}}\overline{\mathbf{h}}_{\mathrm{p_i}}^{\mathrm{t}} + \mathbf{b}_{\mathrm{h}}^{\mathrm{t}}) \quad (12)$$

$$\widehat{\mathbf{h}}_{p_i}^t = \mathrm{BiLSTM}(\widetilde{\mathbf{h}}_{p_i}^t, \overrightarrow{\mathbf{h}}_{p_{i-1}}^t, \overleftarrow{\mathbf{h}}_{p_{i+1}}^t) \quad (13)$$

where $[\cdot;\cdot;\cdot;\cdot]$ refers to the concatenation operation. In matching operation, the concatenation can retain all the information (Wang and Jiang, 2016a; Chen et al., 2017). We use a neural nonlinear transformation ReLU (Glorot et al., 2011) as local comparison function. This operation helps the model to better fuse the attention information and also reduce the complexity of vector representation. Since the understanding of some word-level alignments may rely on the contextual matching information, we then apply a BiLSTM to incorporate the sequential matching information, which further gathers interactive features between two sentences.

We design the deep fusion layer as follows. A gated connection layer is used to learn adaptively controlling how much information to be stored and carried to the next attention layer. It can be formulated as Equations (14) - (17):

$$\mathbf{r}_{p_i}^t = \sigma(\mathbf{W}_r^t[\mathbf{h}_{p_i}^{t-1}; \widehat{\mathbf{h}}_{p_i}^t; \mathbf{h}_{p_i}^{t-1} \odot \widehat{\mathbf{h}}_{p_i}^t] + \mathbf{b}_r^t) \quad (14)$$

$$\mathbf{z}_{p_i}^t = \sigma(\mathbf{W}_z^t[\mathbf{h}_{p_i}^{t-1}; \widehat{\mathbf{h}}_{p_i}^t; \mathbf{h}_{p_i}^{t-1} \odot \widehat{\mathbf{h}}_{p_i}^t] + \mathbf{b}_z^t) \quad (15)$$

$$\widetilde{\mathbf{c}}_{p_i}^t = \tanh(\mathbf{W}_c^t[\mathbf{r}_{p_i}^t \odot \mathbf{h}_{p_i}^{t-1}; \widehat{\mathbf{h}}_{p_i}^t] + \mathbf{b}_c^t) \quad (16)$$

$$\mathbf{h}_{p_i}^t = \mathbf{z}_{p_i}^t \odot \mathbf{h}_{p_i}^{t-1} + (1 - \mathbf{z}_{p_i}^t) \odot \widetilde{\mathbf{c}}_{p_i}^t \quad (17)$$

where $\mathbf{W}_*^t$ and $\mathbf{b}_*^t$ are the learnable parameters, $\mathbf{h}_{p_i}^t$ is the result of current layer, $\mathbf{h}_{p_i}^{t-1}$ is the result from preceding layer, $\sigma$ is a sigmoid function, the value of $\mathbf{r}_{p_i}^t$ and $\mathbf{z}_{p_i}^t$ is between 0 and 1. Intuitively, the model can learn to set the $\mathbf{r}_{p_i}^t$ and $\mathbf{z}_{p_i}^t$ close to 1, thus the more attention information from the preceding layers will be propagated to the following attention layers for matching, and close to 0 implying that the information of preceding layers is less propagated.

After the $t$-th layer of the original semantics-oriented cross attention, each word $p_i$ in sentence $P$ is newly represented by $\mathbf{h}_{p_i}^t$. Similarly, we conduct cross attention for $Q(t)\rightarrow P$, implying that the sentence $Q$ learns attention to the sentence $P$, which will be oriented to the original semantic representation $\mathbf{H}_P^0$ of $P$ to derive the attention-aware representation $\mathbf{h}_{q_j}^t$ for each word $q_j$ of $Q$.

### 3.3 Self-Attention Mechanism

We additionally introduce a self-attention mechanism after cross sentence attention. It captures long-distance context information to learn word representation within each sentence and further enhances the attention-aware representation.

For sentence $P$, its attentive representation $\mathbf{H}_p^T = [\mathbf{h}_{p_1}^T, \cdots, \mathbf{h}_{p_i}^T, \cdots, \mathbf{h}_{p_m}^T]$ is computed after $T$ layers of original semantics-oriented cross sentence attention. We first compute a self-attention matrix $\mathbf{S}^s \in \mathbf{R}^{m \times m}$ as Equation (9).

$$\mathbf{S}_{ij}^s = \langle \mathbf{h}_{p_i}^T, \mathbf{h}_{p_j}^T \rangle \tag{18}$$

where, $\mathbf{S}_{ij}^s$ indicates the relevance between the $i$-th word and $j$-th word in $P$. Then, the self-attention vector for each word in $P$ is computed as follows:

$$\widetilde{\mathbf{h}}_{p_i}^s = \sum_{j=1}^{m} \frac{exp(\mathbf{S}_{ij}^s)}{\sum_{k=1}^{m} exp(\mathbf{S}_{ik}^s)} \mathbf{h}_{p_j}^T \tag{19}$$

Intuitively, $\widetilde{\mathbf{h}}_{p_i}^s$ augments each word representation with global context of the sentence $P$.

After that, an usual fusion augmented by deep fusion, as described in Section 3.2, is also introduced to further enhance the self-attention information within each sentence as follows:

$$\overline{\mathbf{h}}_{p_i}^s = [\mathbf{h}_{p_i}^T; \widetilde{\mathbf{h}}_{p_i}^s; | \mathbf{h}_{p_i}^T - \widetilde{\mathbf{h}}_{p_i}^s |; \mathbf{h}_{p_i}^T \odot \widetilde{\mathbf{h}}_{p_i}^s] \tag{20}$$

$$\widetilde{\mathbf{h}}_{p_i}^s = \mathrm{ReLU}(\mathbf{W}_{\mathrm{h}}^{\mathrm{s}} \overline{\mathbf{h}}_{\mathrm{p_i}}^{\mathrm{s}} + \mathbf{b}_{\mathrm{h}}^{\mathrm{s}}) \tag{21}$$

$$\widehat{\mathbf{h}}_{p_i}^s = \mathrm{BiLSTM}(\widetilde{\mathbf{h}}_{p_i}^s, \overrightarrow{\mathbf{h}}_{p_{i-1}}^s, \overleftarrow{\mathbf{h}}_{p_{i+1}}^s) \tag{22}$$

The deep fusion after the self-attention layer is computed as Equations (14) and (17), which fuses the $\mathbf{h}_{p_i}^T$ from original semantics-oriented cross attention and the $\widehat{\mathbf{h}}_{p_i}^s$ from self-attention to get the final attention-ware representation.

Similarly, we conduct self-attention and deep fusion operations to the sentence $Q$ to derive the attention-aware representation $\mathbf{h}_{q_j}^s$ for each word $q_j$ of $Q$. Then, two sentences are converted to $\mathbf{H}_P^s = [\mathbf{h}_{p_1}^s, \cdots, \mathbf{h}_{p_i}^s, \cdots, \mathbf{h}_{p_m}^s]$ and $\mathbf{H}_Q^s = [\mathbf{h}_{q_1}^s, \cdots, \mathbf{h}_{q_j}^s, \cdots, \mathbf{h}_{q_n}^s]$. Finally, $\mathbf{H}_P^s$ and $\mathbf{H}_Q^s$ are passed into the prediction layer as input $\mathbf{V}_P$ and $\mathbf{V}_Q$ for deciding their semantic relationship.

## 4 Training

For model training, we employ cross-entropy as the loss function since the goal is the make the correct classification. Considering the model complexity, we also add $l_2$-norm of all learnable parameters to the final loss function. Finally, the object is to minimize the following objective function $\mathcal{J}(\theta)$, which can be formulated as:

$$\mathcal{J}(\theta) = -\frac{1}{N} \sum_{i=1}^{N} \log P(y^{(i)} | P^{(i)}, Q^{(i)}; \theta) + \frac{1}{2} \lambda \|\theta\|_2^2 \tag{23}$$

| Dataset | Train | Dev | Test | Avg.L | | Vocab |
|---|---|---|---|---|---|---|
| SNLI | 549K | 9.8K | 9.8K | 14 | 8 | 36K |
| SciTail | 23K | 1.3K | 2.1K | 17 | 12 | 24K |
| Quora | 384K | 10K | 10K | 12 | 12 | 107K |

Table 1: Statistics of datasets: SNLI, SciTail and Quora. Avg.L refers to average length of a pair of sentences.

where $\theta$ denotes all the learnable parameters of our model, $N$ is the number of instances in the training set, $(P^{(i)}, Q^{(i)})$ are the sentence pairs, and $y^{(i)}$ denotes the corresponding annotated label for the $i$-th instance.

**Word Embedding** Following (Tay et al., 2017), to represent each input word, we concatenate three types of vectors: a pre-trained vector, a learnable vector for each word type, and a learnable vector for the POS tag of the word. We use NLTK [1] to acquire POS tags. Finally, we apply a nonlinear transformation ReLU to the concatenated vector to get the final word embedding.

## 5 Experiments

### 5.1 Dataset

We evaluate our model on natural language inference and paraphrase identification tasks with three datasets: the Stanford Natural Language Inference (SNLI) dataset (Bowman et al., 2015), the SciTail dataset (Khot et al., 2018), and the Quora Questions Pairs dataset (Quora).

**SNLI** is a natural language inference dataset (Bowman et al., 2015). The original data set contains 570,152 sentence pairs, each labeled with one of the following relationships: $\mathcal{Y}$ = {entailment, contradiction, neutral}. We follow the same data split as in (Bowman et al., 2015).

**SciTail** is a binary entailment classification task and $\mathcal{Y}$ = {entailment, neutral}. We have the same data split as in (Khot et al., 2018). Notably, the premise and the corresponding hypothesis have high lexical similarity both for entailed and non-entailed pairs, which makes the task particularly difficult.

**Quora** consists of over 400,000 question pairs and $\mathcal{Y}$ = {0, 1} indicating whether two questions are paraphrases of each other. We have the same data split as in (Wang et al., 2017).

[1] http://www.nltk.org/

| Models | Train | Test |
|---|---|---|
| LstmAtt(Rocktäschel et al., 2015) | 85.3 | 83.5 |
| mLSTM (Wang and Jiang, 2016b) | 92.0 | 86.1 |
| LSTMN (Cheng et al., 2016) | 88.5 | 86.3 |
| DecompAtt (Parikh et al., 2016) | 89.5 | 86.8 |
| re-read (Sha et al., 2016) | 90.7 | 87.5 |
| btree-LSTM (Paria et al., 2016) | 88.6 | 87.6 |
| SAN (Im and Cho, 2017) | 89.6 | 86.3 |
| BiMPM (Wang et al., 2017) | 90.9 | 87.5 |
| ESIM (Chen et al., 2017) | 92.6 | 88.0 |
| DIIN (Gong et al., 2017) | 91.2 | 88.0 |
| AF-DMN (Duan et al., 2018) | 94.5 | 88.6 |
| OSOA-DFN (single) | 92.3 | **88.8** |
| BiMPM (ensemble) | 93.2 | 88.8 |
| ESIM (ensemble) | 93.5 | 88.6 |
| DIIN (ensemble) | 92.3 | 88.9 |
| AF-DMN (ensemble) | 94.9 | 89.0 |
| OSOA-DFN (ensemble) | 93.5 | **89.3** |

Table 2: Comparative results with previous models on SNLI dataset.

The detailed statistical information of the three datasets is shown in Table 1.

### 5.2 Implementation Details

We set word embeddings and all of the hidden states of BiLSTMs and MLPs to 300 dimensions. Pre-trained word vectors are 300-dimensional $Glove\ 840B$ (Pennington et al., 2014) and without updating during training. The learnable word vectors and POS vectors have 30 dimensions. For all datasets, there are 3 cross sentence attention layers and 1 self-attention layer. The batch size is set to 64 for SNLI and Quora, 32 for SciTail. We use the Adam method (Kingma and Ba, 2014) for model training. We set the initial learning rate to 5e-4 with a decay ratio of 0.95 for each epoch, and $l_2$ regularizer strength to 6e-5. To prevent overfitting, we use dropout regularization (Srivastava et al., 2014) with a drop rate of 0.2 for all MLPs.

### 5.3 Ensemble

The ensemble strategy has been proved to effectively improve model accuracy. Following (Duan et al., 2018), our ensemble model averages the probability distributions from three individual single OSOA-DFNs, and each of them has the same architecture but different parameter initialization.

### 5.4 Comparison on Natural Language Inference

**SNLI** We compare our model with the following previous models on SNLI dataset, and show the results in Table 2. LstmAtt (Rocktäschel et al., 2015) extend the general LSTM model with attention mechanism. mLSTM (Wang and Jiang, 2016b) exploit LSTM with memory. DecompAtt (Parikh et al., 2016) propose a decomposable word-by-word matching model with attention, and use pre-trained word vector without relying on any word-order information. SAN (Im and Cho, 2017) is a distance-based self-attention network. BiMPM (Wang et al., 2017) design a bilateral multi-perspective matching model from both directions. ESIM (Chen et al., 2017) incorporate the chain LSTM and tree LSTM. Recently, AF-DMN (Duan et al., 2018) adopt attention-fused deep matching network by using multiple stacked cross attention and self-attention layers.

In Table 2, our single OSOA-DFN achieves 88.8% test accuracy. Moreover, we also report the ensemble result, and the test accuracy is 89.3%. Comparative results show that our model outperforms the previous models on single and ensemble scenarios on SNLI dataset. ELMO (Peters et al., 2018) and BERT (Devlin et al., 2018) have been well known as pre-trained language model for acquiring contextual word vectors. However, our model has less computing complexity (340M parameters in BERT while 10M in our model), but obtained competitive performance. We will conduct the comparison with them in the future. In this paper, we evaluated the contribution of original semantics-oriented cross attention and deep fusion to our model.

| Models | Dev | Test |
|---|---|---|
| Majority class | 63.3 | 60.3 |
| Ngram | 65.0 | 70.6 |
| DecompAtt | 75.4 | 72.3 |
| ESIM | 70.5 | 70.6 |
| DGEM | 79.6 | 77.3 |
| DEISTE | 82.4 | 82.1 |
| CAFE | - | 83.3 |
| AF-DMN (re-imp) | 87.2 | 84.4 |
| OSOA-DFN | **88.9** | **86.8** |

Table 3: Comparative results with previous models on SciTail dataset.

**SciTail** We compare our model with the following previous models on SciTail dataset, and show the results in Table 3. The first five models in Table 3 are all implemented in (Khot et al., 2018). DGEM is a graph based attention model using lingual syntactic structures for improved performance (Khot et al., 2018). CAFE (Tay et al., 2017) improve previous comparison operations by compressing alignment vectors into scalar valued fea-

| Models | Test |
|---|---|
| Siamese-CNN | 79.60 |
| Multi-Perspective-CNN | 81.38 |
| Siamese-LSTM | 82.58 |
| Multi-Perspective-LSTM | 83.21 |
| L.D.C | 85.55 |
| BiMPM | 88.17 |
| AF-DMN | 88.72 |
| OSOA-DFN | **89.03** |

Table 4: Comparative results with previous models on Quora dataset.

| Models | Dev | Test |
|---|---|---|
| OSOA-DFN (ori-attention) | 88.9 | 86.8 |
| OSOA-DFN (inter-attention) | 87.1 | 84.2 |

Table 5: Effect of original semantics-oriented cross sentence attention on SciTail dataset.

| Num | Dev | Test |
|---|---|---|
| 1 | 86.9 | 84.0 |
| 2 | 88.6 | 85.1 |
| 3 | 88.9 | 86.8 |
| 4 | 89.1 | 87.2 |
| 5 | 89.2 | 87.4 |

Table 6: Effect of cross sentence attention layers on SciTail dataset.

| Models | Dev | Test |
|---|---|---|
| OSOA-DFN | **88.9** | **86.8** |
| - Deep fusion | 85.8 | 84.7 |
| - Self-attention | 88.1 | 84.8 |
| - BiLSTM fusion | 87.2 | 83.2 |

Table 7: Effect of components on SciTail dataset.

tures. DEISTE (Yin et al., 2018) propose deep explorations of inter-sentence interaction. AF-DMN (re-imp) is our re-implementation of the multi-layered attention model in (Duan et al., 2018) that have not reported the results on this dataset.

On this dataset, our single OSOA-DFN significantly outperforms these strong baselines, achieving the state-of-the-art performance with 86.8% accuracy on the test set. It demonstrates that our model has the ability to improve semantic matching on the challenging SciTail dataset.

### 5.5 Comparison on Paraphrase Identification

**Quora** We compare our model with the following previous models on Quora dataset, and show the results in Table 4. The Siamese-CNN model and Siamese-LSTM model encode sentences with CNN and LSTM respectively, and then predict the relationship between them based on the cosine similarity (Wang et al., 2017). Multi-Perspective-CNN and Multi-Perspective-LSTM adopt multiple perspective cosine matching function (Wang et al., 2017). L.D.C (Wang et al., 2016) and BiMPM (Wang et al., 2017) adopt attention-based framework that performs word-level matching.

As we can see, our single OSOA-DFN outperforms the baselines and achieves 89.03% accuracy on the test set. The results prove that our model is very effective for paraphrase identification task.

### 5.6 Effect of Original Semantics-Oriented Cross Sentence Attention

To verify the effect of original semantics-oriented cross sentence attention, we first implement a variant of our model, namely OSOA-DFN (inter-attention), as shown in Table 5. As in the model of (Duan et al., 2018), we make the cross sentence attention oriented to the intermediate representations from the preceding layer of another sentence, where cross attention is performed on parallel layers between two sentences. The results show that our method achieves higher accuracy on the test set of SciTail dataset, which proves the effect of original semantics-oriented cross attention on extracting expressive features from another sentence for semantic matching.

We further verify the effect of the depth of cross sentence attention on performance, as shown in Table 6. As the number of stacked attention layers increases from 1 to 5, we can see that the performance increases both on the development set and the test set of SciTail dataset. But from 3 to 5, the increased accuracy is slower. We can conclude that the multiple original semantics-oriented attentions are effective in improving matching performance. However, the parameters will grow rapidly with the increasing of the number of stacked attention layer, and a large of number of parameters will increase model complexity. Because of computational cost, we just set the number of cross attention layers to 3 in our experiment.

### 5.7 Effect of Deep Fusion and Self-Attention Mechanism

We verify the effect of deep fusion and self-attention for better understanding of the performance improvement of OSOA-DFN, as shown in Table 7. Only using BiLSTM fusion without deep fusion, the accuracy drops by 2.1% on the test set of SciTail dataset. This indicates the augmented deep fusion at different layers is important in prop-

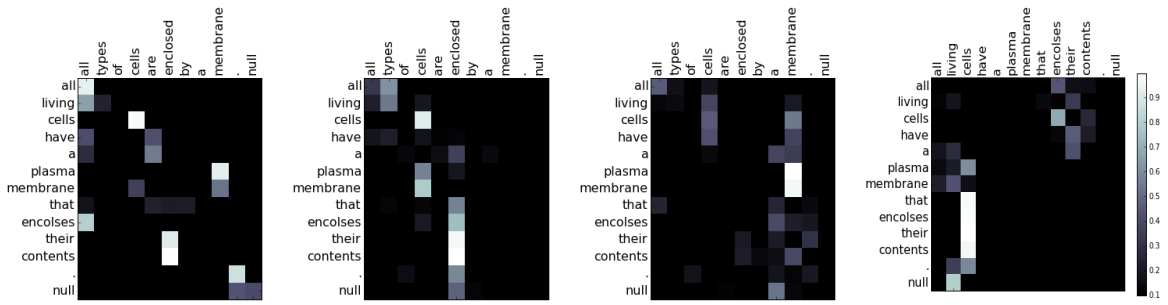


(a) 1st cross attention. (b) 2nd cross attention. (c) 3rd cross attention. (d) self-attention.

Figure 2: The visualization of alignment matrices in the three cross sentence attention layers and the self-attention layer.

agating attention information through the network for deep interaction. Without the self-attention, the accuracy is degraded to 84.8%. This indicates that the self-attention mechanism is effective in capturing global context information for augmenting attention-aware semantic representation.

We also verify the effect of BiLSTM fusion, without the BiLSTM fusion, the accuracy is degraded to 83.2%. This shows contextual information gathered by BiLSTM fusion is important for interaction between two sentences.

### 5.8 What is Learned by Attention ?

We further investigate the results of the multi-layered cross sentence attention and the self-attention and then visualize the results in Figure 2. This is an instance from the test set of the SciTail dataset: {$P$: *all living cells have a plasma membrane that encloses their contents. $Q$: all types of cells are enclosed by a membrane.* The label $y$: *entailment.*}. The results are produced by OSOA-DFN with 3 original semantics-oriented cross sentence attentions $P(t) \rightarrow Q$ and 1 self-attention. We visualize the attention matrices for each layer to show the dynamic attention changes.

From the results, we observe that the first attention layer may have erroneous alignments. We can find that the premise word "*encloses*" is incorrectly aligned with hypothesis word "*all*". In the second attention layer, the alignment quality is improved dramatically, where the "*encloses*" is correctly aligned to "*enclosed*". It shows that the second attention layer effectively revises the errors from the first attention layer. In the second and third attention layers, the attention gradually tends to capture phrase-level alignments, such as "*that encloses their contents*" and "*enclosed*", and "*cells have a plasma membrane*" and "*membrane*". Meanwhile, with the increment of interaction, the high attention layers also tend to capture new alignment information from another sentence that is not captured in low attention layers.

In the self-attention layer, we observe that the phrase "*plasma membrane that encloses their contents*" is strongly aligned to the phrase "*living cells*". This layer captures long-distance semantic dependency within the sentence. The visualization of attention further shows that our proposed model is capable of capturing alignment information from two sentences for better semantic matching.

## 6 Related Works

Recently, deep neural network models have achieved promising results in modeling sentence matching. A standard practice is to encode each sentence as a vector with a neural network (Bowman et al., 2015; Mou et al., 2015; Tan et al., 2015), and then the relation is decided based on the two sentence vectors. This kind of framework ignores the interaction between two sentences.

Most recent works (Wang and Jiang, 2016a; Chen et al., 2017; Duan et al., 2018) employ attention mechanism to model interaction between two sentences. The attention-based framework matches two sentences at the word level. (Wang and Jiang, 2016b) design a specific LSTM called matching-LSTM that performs word-by-word matching of the hypothesis with the premise. Furthermore, (Wang et al., 2017) and (Chen et al., 2017) propose a new framework to model the relationship between two sentences, which performs the matching from two directions. To improve the attention-based framework, (Duan et al., 2018) propose an attention-fused deep matching

network (AD-DMN), which is based on multi-layered attention mechanism and shows that multiple stacked attention layers can improve matching performance. Besides cross sentence attention, the self-attention mechanism is proposed to solve the limitations of RNN model on the long-term dependency problem, which aims to align the sentence with itself and has been used in a variety of tasks (Lin et al., 2017; Duan et al., 2018).

Our proposed OSOA-DFN conducts original semantics-oriented cross sentence attention to model the matching. We design deep fusion to augment the propagation of attention information. At last, we introduce a self-attention mechanism to capture global context to enhance semantic representation. Compared to AF-DMN (Duan et al., 2018), we just use one self-attention layer instead of multiple layers, which reduces model complexity but achieves outperformed accuracy.

## 7 Conclusions and Future Work

In this paper, we propose an original semantics-oriented attention and deep fusion network (OSOA-DFN) for sentence matching. It leverages original semantics-oriented cross sentence attention, deep fusion and self-attention mechanism jointly. We compare our model with the previous models on two sentence matching tasks: natural language inference and paraphrase identification. Experiment results show that OSOA-DFN has the ability to model sentence matching more precisely and significantly improves the performance.

In the future, we will further investigate the effect of the network depth on sentence matching and explore introducing external knowledge, such as pre-trained language model BERT (Devlin et al., 2018) and paraphrase database (Ganitkevitch et al., 2013), to help learning more accurate and robust sentence representation.

### Acknowledgement

The research work descried in this paper has been supported by the National Nature Science Foundation of China (Nos. 61876198, 61976015, 61370130 and 61473294), and also supported by the Fundamental Research Funds for the Central Universities (No. 2018YJS025), the Beijing Municipal Natural Science Foundation (No. 4172047), and the International Science and Technology Cooperation Program of China under Grant No. K11F100010. The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve this paper.

## References

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced lstm for natural language inference. In *Proc. ACL*.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Chaoqun Duan, Lei Cui, Xinchi Chen, Furu Wei, Conghui Zhu, and Tiejun Zhao. 2018. Attention-fused deep matching network for natural language inference. In *IJCAI*, pages 4033–4040.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. Ppdb: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 315–323.

Yichen Gong, Heng Luo, and Jian Zhang. 2017. Natural language inference over interaction space. *arXiv preprint arXiv:1709.04348*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Jinbae Im and Sungzoon Cho. 2017. Distance-based self-attention network for natural language inference. *arXiv preprint arXiv:1712.02047*.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. Scitail: A textual entailment dataset from science question answering. In *Proceedings of AAAI*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2015. Natural language inference by tree-based convolution and heuristic matching. *arXiv preprint arXiv:1512.08422*.

Biswajit Paria, KM Annervaz, Ambedkar Dukkipati, Ankush Chatterjee, and Sanjay Podder. 2016. A neural architecture mimicking humans end-to-end for natural language inference. *arXiv preprint arXiv:1611.04741*.

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*.

Lei Sha, Baobao Chang, Zhifang Sui, and Sujian Li. 2016. Reading and thinking: Re-read lstm unit for textual entailment recognition. In *COLING*, pages 2870–2879.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958.

Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2015. Lstm-based deep learning models for non-factoid answer selection. *arXiv preprint arXiv:1511.04108*.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2017. A compare-propagate architecture with alignment factorization for natural language inference. *arXiv preprint arXiv:1801.00102*.

Shuohang Wang and Jing Jiang. 2016a. A compare-aggregate model for matching text sequences. *arXiv preprint arXiv:1611.01747*.

Shuohang Wang and Jing Jiang. 2016b. Learning natural language inference with lstm. In *Proceedings of NAACL-HLT*, pages 1442–1451.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. corr abs/1702.03814.

Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. 2016. Sentence similarity learning by lexical decomposition and composition. *arXiv preprint arXiv:1602.07019*.

Wenpeng Yin, Hinrich Schütze, and Dan Roth. 2018. End-task oriented textual entailment via deep explorations of inter-sentence interactions. *arXiv preprint arXiv:1804.08813*.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2015. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *arXiv preprint arXiv:1512.05193*.

# Representation Learning with Ordered Relation Paths for Knowledge Graph Completion

**Yao Zhu**[1], **Hongzhi Liu**[2*], **Zhonghai Wu**[3,4], **Yang Song**[5] and **Tao Zhang**[5]
[1] Center for Data Science, Peking University, Beijing, China
[2] School of Software and Microelectronics, Peking University, Beijing, China
[3] National Engineering Center of Software Engineering, Peking University, Beijing, China
[4] Key Lab of High Confidence Software Technologies (MOE), Peking University, Beijing, China
[5] BossZhipin NLP Center @Kanzhun.com, Beijing, China
{yao.zhu,liuhz,wuzh}@pku.edu.cn, {songyang,kylen.zhang}@kanzhun.com

## Abstract

Incompleteness is a common problem for existing knowledge graphs (KGs), and the completion of KG which aims to predict links between entities is challenging. Most existing KG completion methods only consider the direct relation between nodes and ignore the relation paths which contain useful information for link prediction. Recently, a few methods take relation paths into consideration but pay less attention to the order of relations in paths which is important for reasoning. In addition, these path-based models always ignore nonlinear contributions of path features for link prediction. To solve these problems, we propose a novel KG completion method named OPTransE. Instead of embedding both entities of a relation into the same latent space as in previous methods, we project the head entity and the tail entity of each relation into different spaces to guarantee the order of relations in the path. Meanwhile, we adopt a pooling strategy to extract nonlinear and complex features of different paths to further improve the performance of link prediction. Experimental results on two benchmark datasets show that the proposed model OPTransE performs better than state-of-the-art methods.

## 1 Introduction

Knowledge graphs (KGs) are built to store structured facts which are encoded as triples, e.g., (**Beijing**, *CapitalOf*, **China**) (Lehmann et al., 2015). Each triple $(h, r, t)$ consists of two entities $h$, $t$ and a relation $r$, indicating there is a relation $r$ between $h$ and $t$. Large-scale KGs such as YAGO (Suchanek et al., 2007), Freebase (Bollacker et al., 2008) and WordNet (Miller, 1995) contain billions of triples and have been widely applied in various fields (Riedel et al., 2013; Dong et al., 2015). However, a common problem with these KGs is that they are far from complete, which has limited the development of KG's applications. Thus, KG completion with the goal of filling in missing parts of the KG has become an urgent issue. Specifically, KG completion aims to predict whether a relationship between two entities is likely to be true, which is defined as the link prediction in KGs.

Most existing KG completion methods are based on representation learning, which embed both entities and relations into continuous low-dimension spaces. TransE (Bordes et al., 2013) is one of the most classical KG completion models, which embeds entities and relations into the same latent space. To better deal with complex relations like 1-to-N, N-to-1 and N-to-N, TransH (Wang et al., 2014) and TransR (Lin et al., 2015b) employ relation-specific hyperplanes and relation-specific spaces respectively to separate triples according their corresponding relation. Unfortunately, these models ignore the relation paths between entities which are helpful for reasoning. For example, if we know A is B's brother, and B is C's parent, then we can infer that A is C's uncle.

Recently, a few researchers take relation paths in KGs as additional information for representation learning and attempt to project paths into latent spaces, which get better performance compared with conventional methods. PTransE-ADD (Lin et al., 2015a) considers relation paths as translations between entities and represents each path as the vector sum of all the relations in the path. Moreover, RPE (Lin et al., 2018) extends the TransR model by incorporating the path-specific projection. However, these methods pay less attention to the order of relations in paths which is important for link prediction. Figure 1 shows an example of the meaning change when the order of relations is altered. In addition, these path-based models assume information from different paths between an entity pair only contributes to the re-

*Corresponding author



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2662–2671, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

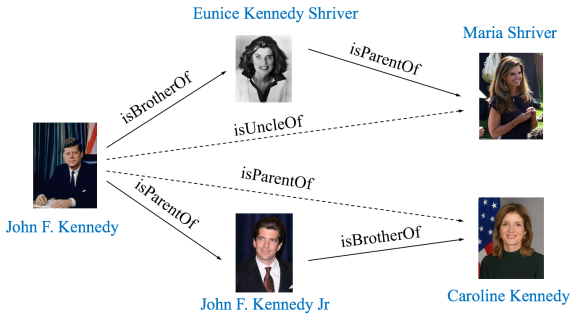


Figure 1: Example of the meaning change when the order of relations is altered.

lation inference linearly and ignore other complex interactions between them.

To address these issues, we propose a novel KG completion model named OPTransE. In the model, we project the head entity and the tail entity of each relation into different spaces and introduce sequence matrices to keep the order of relations in the path. Moreover, a pooling strategy is adopted to extract nonlinear features of different paths for relation inferences. Experimental results on two benchmark datasets WN18 and FB15K show that OPTransE significantly outperforms state-of-the-art methods.

The remainder of this paper is organized as follows. Section 2 discusses related work. Section 3 presents the proposed model and algorithm in detail. Empirical evaluation of the proposed algorithm and comparison with other state-of-the-art algorithms are presented in Section 4. Finally, Section 5 summarises the whole paper and points out some future work.

## 2 Related Work

### 2.1 Translation-based Models

In recent years, there has been a great deal of work on representation learning for KG completion, and most studies concentrate on translation-based models. This kind of models propose to embed both entities and relations into a continuous low-dimensional vector space according to some distance-based scoring functions.

TransE (Bordes et al., 2013) is one of the most fundamental and representative translation-based models. For the entities and relations in KGs, TransE encodes them as vectors in the same space. For each fact $(h, r, t)$, TransE believes that $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$ when $(h, r, t)$ holds. Thus, the scoring function is defined as

$$f_r(h,t) = \| \mathbf{h} + \mathbf{r} - \mathbf{t} \|_{1/2} . \quad (1)$$

where $\mathbf{h}$, $\mathbf{r}$ and $\mathbf{t}$ represent the vectors of head entity $h$, relation $r$ and tail entity $t$, respectively. If the fact $(h, r, t)$ is true, its score $f_r(h, t)$ tends to be close to zero.

TransE is a simple and efficient method for KG completion. However, its simple structure has flaws in dealing with complicated relations like 1-to-N, N-to-1 and N-to-N. In order to address this problem, TransH (Wang et al., 2014) introduces relation-specific hyperplanes and projects entities as vectors onto the given hyperplanes. Similar to TransH, TransR (Lin et al., 2015b) also aims to cope with complicated relations. Instead of employing the hyperplane like TransH, TransR proposes a matrix $\mathbf{W_r} \in \mathbb{R}^{m \times n}$ to project entity vectors into a relation-specific space. Moreover, STransE (Nguyen et al., 2016) extends TransR by introducing two projection matrices for the head entity and the tail entity, respectively. Therefore, the head and tail entities in a triple will be projected differently into the corresponding relation space.

### 2.2 Incorporating Relation Paths

The models introduced so far only exploit facts observed in KGs to conduct representation learning. In fact, there is a large amount of useful information in relation paths that can be incorporated into translation-based models to improve the performance of link prediction.

Lin et al. (2015a) proposes a path-based translation model named PTransE for KG completion. It regards relation paths as translations between entities for representation learning and utilizes a path-constraint resource allocation algorithm to evaluate the reliability of relation paths. RTransE (García-Durán et al., 2015) and TransE-COMP (Guu et al., 2015) take the sum of the vectors of all relations in a path as the representation for a relation path. For the Bilinear-COMP model (Guu et al., 2015), and the PRUNED-PATHS model (Toutanova et al., 2016), they represent each relation as a diagonal matrix, and evaluate the relation path by matrix multiplication. Most recently, PaSKoGE model (Jia et al., 2018) is proposed for KG embedding by minimizing a path-specific margin-based loss function. Moreover, RPE (Lin et al., 2018), inspired by PTransE, ex-

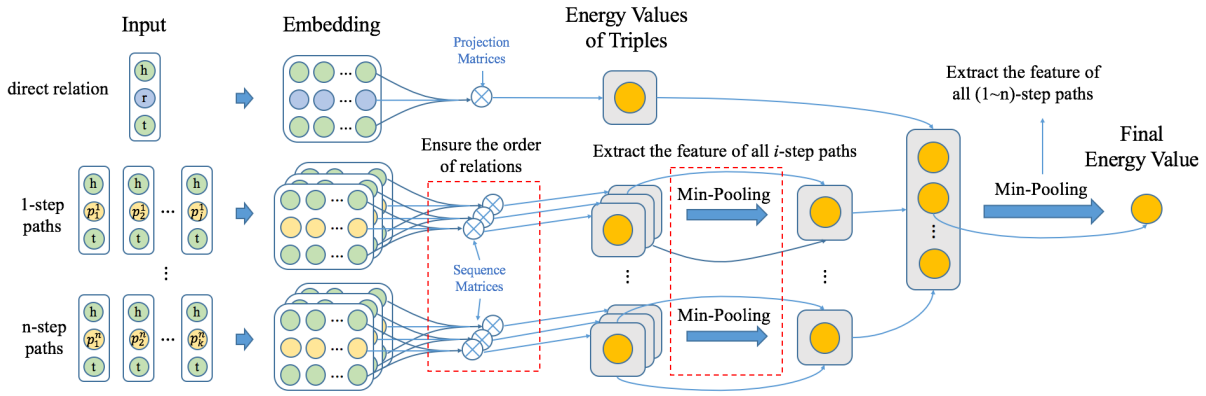


Figure 2: Architecture of OPTransE.

tends the TransR model by incorporating the path-specific projection for paths between entity pairs.

These methods try to incorporate information of relation paths to get better performance. However, they pay less attention to the order of relations in a path when learning representations of the path. In fact, changes in the relation order of paths will alter the meanings of paths to a great extent (as shown in Figure 1). Moreover, the methods stated above assume information from different paths between an entity pair only contributes to the relation inference linearly. Unfortunately, they ignore the complex nonlinear features of different paths. In order to solve these problems, we propose OPTransE, a novel KG completion model, which learns representations of ordered relation paths and designs a pooling method to better extract nonlinear features from various relation paths.

## 3 Our Model

To infer the missing parts of KGs, we propose a KG completion model called OPTransE, whose architecture is shown in Figure 2. We first embed the entities and relations of KG into latent spaces with the consideration of the order of relations in paths. Then, we try to infer the missing relations using these latent representations. Different from previous methods which embed the head and tail of a relation into the same latent space, we project them into different spaces. Therefore, we can distinguish the order of relations in the path. To extract the complex and nonlinear path information for relation reasoning, we design a two layer pooling strategy to fuse the information from different paths.

In this section, we will first introduce the embedding representations of ordered relation paths. After that, we utilize a two layer pooling strategy to construct the total energy function of triples and then the objective function is presented. Finally, we will describe the detail of model implementation and analyze the complexity of the model.

### 3.1 Ordered Relation Paths Representation

For each triple $(h, r, t)$ in KG, we employ vectors to represent the entity pair and the relation. Specifically, $\mathbf{h}\in\mathbb{R}^d$ denotes the head entity $h$, $\mathbf{t}\in\mathbb{R}^d$ denotes the tail entity $t$ and $\mathbf{r}\in\mathbb{R}^d$ indicates the relation $r$.

We assume the paths connecting two entities contain indicative information for the direct relation between these two entities. To measure these kinds of indicative effects while guarantee the order of relations in a path, we define an energy function in Equation (2). Let $p^{s=n}$ denote one of the $n$-step path from $h$ to $t$, i.e., $h \xrightarrow{r_1} \cdots \xrightarrow{r_n} t$. If the relation path is reasonable from $h$ to $t$, it will obtain a lower energy value.

$$E\left(h, p^{s=n}, t\right) = \| \mathbf{h_p} + \sum_{i=1}^{n} \mathbf{S}_{\mathbf{p}}^{i}\mathbf{r}_i - \mathbf{t_p} \|_{1/2}, \quad (2)$$

where

$$\mathbf{h_p} = f(p, \mathbf{h}), \quad \mathbf{t_p} = g(p, \mathbf{t}), \quad (3)$$

$$\mathbf{S}_{\mathbf{p}}^{i} = \mathbf{W}(p, i). \quad (4)$$

$\mathbf{h_p}$ and $\mathbf{t_p}$ denote the representations of the head entity $h$ and the tail entity $t$ in the ordered relation path $p$, respectively. $\mathbf{S}_{\mathbf{p}}^{i}\in\mathbb{R}^{d\times d}$ denotes the sequence matrix with respect to the $i$-th relation in the given path $p$.

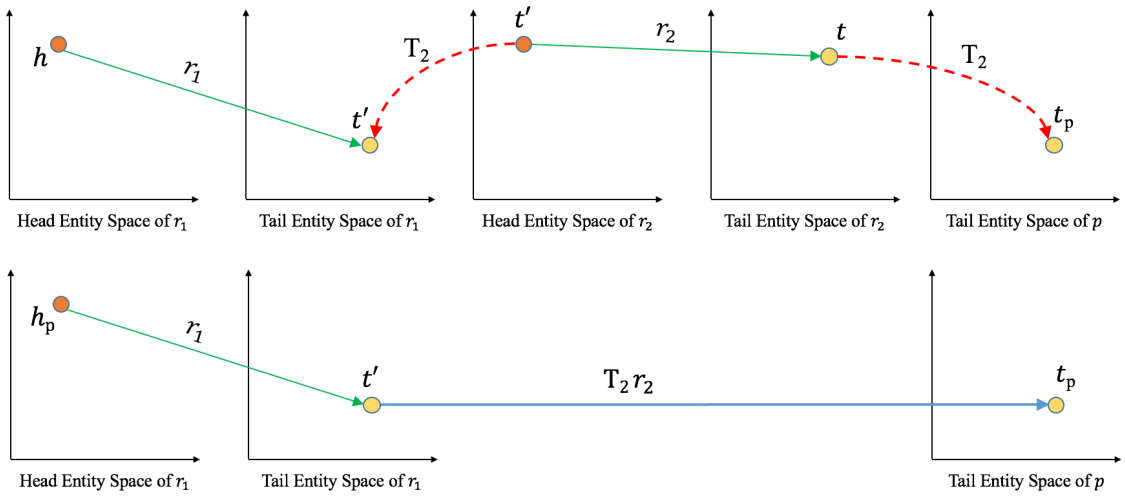


Figure 3: The representation of path $p$ ($h \xrightarrow{r_1} t' \xrightarrow{r_2} t$). The top part of the figure depicts the process of space transition of $t'$ and $t$, while the bottom part illustrates the generated continuous path from $h_p$ to $t_p$ after the transition.

Note that a triple $(h, r, t)$ in the KG can be seen as a one-step path between $h$ and $t$. Thus, the value of $E(h, r, t)$ is able to be obtained by substituting direct relation $r$ as $p^{s=1}$ into Equation (2).

From Equation (2) we could observe that the sequence matrix $\mathbf{S}_\mathbf{p}^i$ before each relation $\mathbf{r}_i$ is different. If the order of several relations in a path is altered, the value of energy function will also change at the same time. Therefore, paths with the same relation set but different relation order will infer out distinct direct relations in our model. The specific representation of the ordered relation path will be demonstrated in the following contents.

To keep the order information of relations in paths, we project the head and tail entities of a relation into different spaces by introducing two matrices for each relation. Let $\mathbf{W}_{\mathbf{r},\mathbf{1}} \in \mathbb{R}^{d\times d}$ and $\mathbf{W}_{\mathbf{r},\mathbf{2}} \in \mathbb{R}^{d\times d}$ denote the projection matrices of the head entity and the tail entity for relation $r$, respectively. With these two matrices, we will project the head and tail entities into distinct spaces with respect to the same relation. Suppose there is a path $r_1, r_2, \ldots, r_\mathrm{n}$ from $h$ to $t$, ideally, we define the following equations

$$\begin{cases} \mathbf{W}_{\mathbf{r_1},\mathbf{1}}\mathbf{h} + \mathbf{r_1} = \mathbf{W}_{\mathbf{r_1},\mathbf{2}}\mathbf{t}^{(1)} \\ \mathbf{W}_{\mathbf{r_2},\mathbf{1}}\mathbf{t}^{(1)} + \mathbf{r_2} = \mathbf{W}_{\mathbf{r_2},\mathbf{2}}\mathbf{t}^{(2)} \\ \vdots \\ \mathbf{W}_{\mathbf{r_n},\mathbf{1}}\mathbf{t}^{(n-1)} + \mathbf{r_n} = \mathbf{W}_{\mathbf{r_n},\mathbf{2}}\mathbf{t} \end{cases}, \tag{5}$$

where $\mathbf{t}^{(i)}$ indicates the $i$-th passing node on the path.

For the entity pair with a relation path, we get their representations after eliminating the passing nodes from Equation (5). Thus, the concrete forms of the variables in Equation (2) are shown as follows,

$$\mathbf{h_p} = \mathbf{W}_{\mathbf{r_1},\mathbf{1}}\mathbf{h}, \quad \mathbf{t_p} = \mathbf{W}_{\mathbf{p^{s=n}}}\mathbf{t}, \tag{6}$$

$$\mathbf{S}_\mathbf{p}^i = \prod_{k=1}^{i} \mathbf{T}_k, \tag{7}$$

where

$$\mathbf{W}_{\mathbf{p^{s=n}}} = \mathbf{S}_\mathbf{p}^n \mathbf{W}_{\mathbf{r_n},\mathbf{2}}, \tag{8}$$

$$\mathbf{T}_k = \begin{cases} I & k = 1 \\ \mathbf{M}\left(r_k, r_{k-1}\right) & k > 1 \end{cases}. \tag{9}$$

$\mathbf{W}_{\mathbf{p^{s=n}}} \in \mathbb{R}^{d\times d}$ indicates the projection matrix for path $p^{s=n}$, which aims to project the tail entity in a path to the space of $p^{s=n}$. Moreover, $I$ in Equation (9) denotes the identity matrix and $\mathbf{M}\left(r_k, r_{k-1}\right) \in \mathbb{R}^{d\times d}$ means the space transition matrix from the head entity space of $r_k$ to the tail entity space of $r_{k-1}$, i.e., $\mathbf{M}\left(r_k, r_{k-1}\right)\mathbf{W}_{\mathbf{r_k},\mathbf{1}} = \mathbf{W}_{\mathbf{r_{k-1}},\mathbf{2}}$.

Figure 3 illustrates the representation of the relation path in our model. Suppose there is a 2-step path from $h$ to $t$ passing $t'$, i.e., $h \xrightarrow{r_1} t' \xrightarrow{r_2} t$. It is obvious that $t'$ acts as the tail entity of relation $r_1$ and as the head entity of relation $r_2$ at the same time, which is shown on the top part of Figure 3. To connect relations in different spaces, we try to

unify the passing node in the path into the same space. As defined in Equation (9), $T_2$ is utilized to transfer the passing node $t'$ from the head entity space of $r_2$ to the tail entity space of $r_1$. Moreover, $T_2$ is also assigned to the relation $r_2$ and the tail entity $t$. Note that the tail entity $t$ will be projected into the space of path $p$ which is defined in Equation (6). Finally, the path from $h_p$ to $t_p$ will pass through $r_1$ and $T_2 r_2$ as shown on the bottom part of Figure 3.

### 3.2 Pooling Strategy

We design a two layer pooling strategy to fuse the information from different paths. First, we utilize a minimum pooling method to extract feature information from paths with $i$ steps and define an energy function as follows,

$$E\left(h, P_r^{s=i}, t\right) = \text{Min}[\, E\left(h, p^{s=i}, t\right) \mid p^{s=i} \in P_r^{s=i}], \tag{10}$$

where $P_r^{s=i}$ indicates the set of all *i*-step paths which are relevant to the relation $r$ from the head entity $h$ to the tail entity $t$. To obtain $P_r^{s=i}$, we introduce a conditional probability $\Pr(r|p^{s=i})$ to represent the reliability of a path $p^{s=i}$ associated with the given relation $r$,

$$\begin{aligned} \Pr\left(r\middle|p^{s=i}\right) &= \Pr(r, p^{s=i}) / \Pr(p^{s=i}) \\ &= \frac{N(r, p^{s=i})/N(p)}{N(p^{s=i})/N(p)} \\ &= \frac{N(r, p^{s=i})}{N(p^{s=i})}, \end{aligned} \tag{11}$$

where $\Pr(r, p^{s=i})$ denotes the joint probability of $r$ and $p^{s=i}$, $\Pr(p^{s=i})$ denotes the marginal probability of $p^{s=i}$. In addition, $N(r, p^{s=i})$ denotes the number of cases where $r$ and $p^{s=i}$ link the same entity pair in the KG, $N(p^{s=i})$ denotes the number of the path $p^{s=i}$ in the KG and $N(p)$ denotes the total number of paths in the KG. Since $N(p)$ can be removed from both the numerator and denominator of the fractional expression, we finally convert the probability into frequency for computation.

We filter the paths by choosing all $p^{s=i}$ from $h$ to $t$ whose $\Pr\left(r\middle|p^{s=i}\right) > 0$. Thus, $P_r^{s=i}$ is the set of all filtered $p^{s=i}$. Sometimes we could infer the fact not from the direct relation $r$ but from the path, which means the value of $E\left(h, P_r^{s=i}, t\right)$ could possibly be less than that of $E\left(h, r, t\right)$.

Furthermore, we utilize a minimum pooling method to fuse information from paths with different lengths and define an energy function as follows,

$$\begin{aligned} E_{final}\left(h, r, t\right) = \text{Min}[\, & E\left(h, r, t\right),\ E\left(h, P_r^{s=1}, t\right), \\ & E\left(h, P_r^{s=2}, t\right), \ldots,\ E\left(h, P_r^{s=n}, t\right)], \end{aligned} \tag{12}$$

where $E\left(h, r, t\right)$ indicates the energy value of direct relation $r$ and it is calculated by substituting $r$ as $p^{s=1}$ into Equation (2). $E\left(h, P_r^{s=i}, t\right)$ is initialized as **infinite**, thus it will not influence the outcome of final energy function if there is no *i*-step path between $h$ and $t$.

In summary, we adopt the min-pooling strategy twice in our model. For $E\left(h, P_r^{s=i}, t\right)$, min-pooling aims to choose the most matched path with $r$ among all *i*-step paths. And for the final energy function, min-pooling tries to extract nonlinear features from paths of various lengths. In addition, the min-pooling method addresses the problem that there may be no relation paths between $h$ and $t$.

### 3.3 Objective Function

The objective function for the proposed model OPTransE is formalized as

$$\begin{aligned} L\left(S\right) = \sum_{(h,r,t)\in S} \Bigg\{ & L\left(h, r, t\right) + \lambda \cdot \sum_i \Bigg[ \frac{1}{Z_i} \\ & \cdot \sum_{p^{s=i} \in P_r^{s=i}} \Pr\left(p^{s=i}\middle|h, t\right) \cdot \Pr\left(r\middle|p^{s=i}\right) \cdot L\left(h, p^{s=i}, t\right) \Bigg] \Bigg\}, \end{aligned} \tag{13}$$

where $L\left(h, r, t\right)$ indicates the loss function for the triple $(h, r, t)$, and $L\left(h, p^{s=i}, t\right)$ represents the loss value with respect to the relation path $p^{s=i}$. The probability $\Pr\left(p^{s=i}\middle|h, t\right)$ indicates the reliability of the relation path $p^{s=i}$ given the entity pair $(h, t)$, and $\Pr(r|p^{s=i})$ denotes the reliability of a path $p^{s=i}$ associated with the given relation $r$. The details of $\Pr\left(p^{s=i}\middle|h, t\right)$ are shown in (Lin et al., 2015a), which is computed by a path-constraint resource allocation algorithm. $Z_i = \sum_{p^{s=i}\in P_r^{s=i}} \Pr\left(p^{s=i}\middle|h, t\right) \Pr\left(r\middle|p^{s=i}\right)$ is a normalization factor, and $\lambda$ is utilized to balance the triple loss and paths losses.

We adopt the margin-based loss in our model, i.e.,

$$L\left(h, r, t\right) = \sum_{(h', r, t')\in S'} \left[\gamma + E\left(h, r, t\right) - E(h', r, t')\right]_+, \tag{14}$$

$$L\left(h, p^{s=i}, t\right) = \sum_{(h',r,t')\in S'} [\gamma_i + E\left(h, p, t\right) - E(h', p, t')]_+, \tag{15}$$

where $p$ is the simple form of $p^{s=i}$. $[x]_+ = max(x, 0)$ returns the higher one between $x$ and 0. $\gamma_i$ is the margin to separating positive and negative samples. It is noteworthy that we employ different margin $\gamma_i$ for paths with different step number because the noise of energy function will be magnified as the number of steps increases. The corrupted triple set $S'$ for $(h, r, t)$ is denoted as follows:

$$S' = \{(h', r, t) \cup (h, r, t')\}. \tag{16}$$

We replace the head entity or the tail entity in the triple randomly and guarantee that the new triple is not an existing valid triple.

Our goal is to minimize the total loss. Valid relation paths will obtain lower energy value after the optimization, so that paths can sometimes replace directed relations when performing the prediction.

### 3.4 Parameter Learning

We utilize stochastic gradient descent (SGD) to optimize the objective function in Equation (13) and learn parameters of the model. To ensure the convergence of the model, we impose limitations to the norm of vectors, i.e., $||\mathbf{h}||_2 \leq 1$, $||\mathbf{r}||_2 \leq 1, ||\mathbf{t}||_2 \leq 1, ||\mathbf{W_{r,1}h}||_2 \leq 1, ||\mathbf{W_{r,2}t}||_2 \leq 1$. Moreover, we note that the objective function defined in Equation (13) has two parts. The first part is for the basic triple and the second part is for the relation paths. To focus on the representation of ordered relation paths in the second part, we only update the parameters of relation vectors in the path when conducting the optimization of the model.

In addition, we follow PTransE (Lin et al., 2015a) to generate reverse relation $r^{-1}$ to enlarge the training set, and the inference in KGs can be through the reverse paths. For instance, for the fact (**Honolulu**, *CapitalOf*, **Hawaii**), we will also add a fact with the reverse relation to the KG, i.e., (**Hawaii**, *CapitalOf*$^{-1}$, **Honolulu**).

### 3.5 Complexity Analysis

Let $d$ denote the dimension of entities and relations, $N_e$ and $N_r$ denote the number of entities and relations, respectively. The number of model parameters for OPTransE is ($N_e d$ + $N_r d$ + $2N_r d^2$), which is the same as that of STransE.

Moreover, let $N_p$ denote the expected number of relation paths between the entity pair, $N_t$ denote the number of triples for training, $k$ denote the maximum length of relation paths. According to the objective function shown in Equation (13) and details of parameter learning stated in Section 3.4, the time complexity of OPTransE for optimization is $O(k^2 d^3 N_p N_t)$, which is on the same magnitude as that of RPE(MCOM) (Lin et al., 2018).

## 4 Experiments

### 4.1 Datasets

To evaluate the proposed model OPTransE, we use two benchmark datasets: WN18 and FB15K as experimental data. They are subsets of the knowledge graph WordNet (Miller, 1995) and Freebase (Bollacker et al., 2008), respectively (Bordes et al., 2013). These two datasets have been widely employed by researchers for KG completion (Jia et al., 2018; Lin et al., 2018). The statistic details of the two datasets are shown in Table 1. In our experiments, as we add triples of reverse relations to the datasets, the number of relations and training triples are doubled.

Table 1: Statistics of datasets

| Dataset | #Rel | #Ent | #Train | #Valid | #Test |
|---|---|---|---|---|---|
| WN18 | 18 | 40,943 | 141,442 | 5,000 | 5,000 |
| FB15K | 1345 | 14,951 | 483,142 | 50,000 | 59,071 |

### 4.2 Experimental Settings

We adopt the idea from TransR (Lin et al., 2015b) and initialize the vectors and matrices of OPTransE by an existing method STransE (Nguyen et al., 2016). Following TransH (Wang et al., 2014), *Bernoulli* method is applied for generating head or tail entities when sampling corrupted triples.

As the length of paths increases, the reliability of the path will decline accordingly. To better determine the maximum length of paths for experiment, before the test on FB15K, we had evaluated OPTransE with 3-step paths on WN18. However, OPTransE (3-step) performs comparably as OPTransE (2-step) with a higher computational cost. This indicates that longer paths hardly contain more useful information and it is unnecessary

Table 2: Evaluation results on link prediction

| Model | WN18 | | | | FB15K | | | |
|---|---|---|---|---|---|---|---|---|
| | Mean Rank | | Hits@10(%) | | Mean Rank | | Hits@10(%) | |
| | Raw | Filtered | Raw | Filtered | Raw | Filtered | Raw | Filtered |
| SE | 1011 | 985 | 68.5 | 80.5 | 273 | 162 | 28.8 | 39.8 |
| SME | 545 | 533 | 65.1 | 74.1 | 274 | 154 | 30.7 | 40.8 |
| TransE | 263 | 251 | 75.4 | 89.2 | 243 | 125 | 34.9 | 47.1 |
| TransH | 318 | 303 | 75.4 | 86.7 | 212 | 87 | 45.7 | 64.4 |
| TransR | 238 | 225 | 79.8 | 92.0 | 198 | 77 | 48.2 | 68.7 |
| TranSparse | 223 | 211 | 80.1 | 93.2 | 187 | 82 | 53.5 | 79.5 |
| STransE | 217 | 206 | 80.9 | 93.4 | 219 | 69 | 51.6 | 79.7 |
| ITransF | - | 205 | - | 94.2 | - | 65 | - | 81.0 |
| HolE | - | - | - | 94.9 | - | - | - | 73.9 |
| ComplEx | - | - | - | 94.7 | - | - | - | 84.0 |
| ANALOGY | - | - | - | 94.7 | - | - | - | 85.4 |
| ProjE | 277 | 260 | 79.4 | 94.9 | **124** | 34 | 54.7 | 88.4 |
| RTransE | - | - | - | - | - | 50 | - | 76.2 |
| PTransE (ADD, 2-step) | 235 | 221 | 81.3 | 92.7 | 200 | 54 | 51.8 | 83.4 |
| PTransE (MUL, 2-step) | 243 | 230 | 79.5 | 90.9 | 216 | 67 | 47.4 | 77.7 |
| PTransE (ADD, 3-step) | 238 | 219 | 81.1 | 94.2 | 207 | 58 | 51.4 | 84.6 |
| PaSKoGE | - | - | 81.3 | 95.0 | - | - | 53.1 | 88.0 |
| RPE (ACOM) | - | - | - | - | 171 | 41 | 52.0 | 85.5 |
| RPE (MCOM) | - | - | - | - | 183 | 43 | 52.2 | 81.7 |
| RotatE | - | 309 | - | **95.9** | - | 40 | - | 88.4 |
| OPTransE | **211** | **199** | **83.2** | 95.7 | 136 | **33** | **58.0** | **89.9** |

to enumerate longer paths. Therefore, considering the computational efficiency, we limit the maximum length of relation paths as 2 steps.

In our experiments, we utilize the grid search to choose the best parameters for the two datasets, respectively. The best configurations for OPTransE are as follows: the dimension of entity and relation vectors $d = 50$, the learning rate $\alpha = 0.0001$, the margin $\gamma = 5.0$, $\gamma_1 = 5.0, \gamma_2 = 5.5$, the balance factor $\lambda = 0.01$ on WN18; and $d = 100$, $\alpha = 0.0005$, $\gamma = 4.0$, $\gamma_1 = 4.5, \gamma_2 = 5.0$, $\lambda = 0.01$ on FB15K. In addition, L1 norm is employed for scoring and we run SGD for 2000 epochs in the training procedure.

### 4.3 Evaluation Metrics and Baselines

The same as in previous work (Bordes et al., 2013; Nguyen et al., 2016), we evaluate the proposed model OPTransE on the link prediction task. This task aims to predict the missing entity in a triple $(h, r, t)$, i.e., predicting $h$ when $r$ and $t$ are given, or predicting $t$ given $h$ and $r$. When testing a fact $(h, r, t)$, we replace either head or tail entity with all entities in the dataset and calculate scores of generated triples according to Equation (12). And then we sort the entities with their scores in ascending order to locate the rank of the target entity.

For specific evaluation metrics, we employ the widely used mean rank (MR) and Hits@10 in the experiments. Mean rank indicates the average rank of correct entities and Hits@10 means the proportion of correct entities ranked in top 10. Higher Hits@10 or lower value of mean rank implies the better performance of the model on the link prediction task. Moreover, it is noted that the generated triple for test may exist in the dataset as a fact, thus such triples will affect the final rank of the target entity to some extent. Hence, we could filter out these generated triples which are facts in the dataset before ranking. If we have performed filtering, the result will be denoted as "Filtered", otherwise it will be denoted as "Raw".

Moreover, Bordes et al. (2013) defined four categories of relations in KGs by mapping their properties such as 1-to-1, 1-to-N, N-to-1 and N-to-N. Thus, experimental results of distinguishing the four different relation types have also been

Table 3: Filtered evaluation results on FB15K by mapping properties of relations(%)

| Tasks | Predicting Head Entities (Hits@10) | | | | Predicting Tail Entities (Hits@10) | | | |
|---|---|---|---|---|---|---|---|---|
| Relation Category | 1-to-1 | 1-to-N | N-to-1 | N-to-N | 1-to-1 | 1-to-N | N-to-1 | N-to-N |
| SE | 35.6 | 62.6 | 17.2 | 37.5 | 34.9 | 14.6 | 68.3 | 41.3 |
| SME (linear) | 35.1 | 53.7 | 19.0 | 40.3 | 32.7 | 14.9 | 61.6 | 43.3 |
| SME (bilinear) | 30.9 | 69.6 | 19.9 | 38.6 | 28.2 | 13.1 | 76.0 | 41.8 |
| TransE | 74.6 | 86.6 | 43.7 | 70.6 | 71.5 | 49.0 | 85.0 | 72.9 |
| TransH | 66.8 | 87.6 | 28.7 | 64.5 | 65.5 | 39.8 | 83.3 | 67.2 |
| TransR | 78.8 | 89.2 | 34.1 | 69.2 | 79.2 | 37.4 | 90.4 | 72.1 |
| TranSparse | 86.8 | 95.5 | 44.3 | 80.9 | 86.6 | 56.6 | 94.4 | 83.3 |
| STransE | 82.8 | 94.2 | 50.4 | 80.1 | 82.4 | 56.9 | 93.4 | 83.1 |
| PTransE(ADD, 2-step) | 91.0 | 92.8 | 60.9 | 83.8 | 91.2 | 74.0 | 88.9 | 86.4 |
| PTransE(MUL, 2-step) | 89.0 | 86.8 | 57.6 | 79.8 | 87.8 | 71.4 | 72.2 | 80.4 |
| PTransE(ADD, 3-step) | 90.1 | 92.0 | 58.7 | 86.1 | 90.7 | 70.7 | 87.5 | 88.7 |
| PaSKoGE | 89.7 | 94.8 | 62.3 | 86.7 | 89.3 | 72.9 | 93.4 | 88.9 |
| RPE (ACOM) | 92.5 | 96.6 | 63.7 | 87.9 | 92.5 | 79.1 | 95.1 | 90.8 |
| RPE (MCOM) | 91.2 | 95.8 | 55.4 | 87.2 | 91.2 | 66.3 | 94.2 | 89.9 |
| RotatE | 92.2 | 96.7 | 60.2 | 89.3 | 92.3 | 71.3 | 96.1 | 92.2 |
| OPTransE | **93.1** | **97.4** | **69.0** | **89.8** | **92.8** | **87.4** | **96.7** | **92.3** |

recorded for comparison.

In the link prediction task, several competitive KG completion methods are utilized as baselines, including SE (Bordes et al., 2011), SME (Bordes et al., 2014), TransE (Bordes et al., 2013), TransH (Wang et al., 2014), TransR (Lin et al., 2015b), TranSparse (Ji et al., 2016), STransE (Nguyen et al., 2016), ITransF (Xie et al., 2017), HolE (Nickel et al., 2016), ComplEx (Trouillon et al., 2016), ANALOGY (Liu et al., 2017), ProjE (Shi and Weninger, 2017), RTransE (García-Durán et al., 2015), PTransE (Lin et al., 2015a), PaSKoGE (Jia et al., 2018), RPE (Lin et al., 2018) and RotatE (Sun et al., 2019). Among them, RTransE, PTransE, PaSKoGE and RPE exploit the information of paths between entity pairs.

### 4.4 Results

Table 2 shows the performances of different methods on the link prediction task according to various metrics. Numbers in **bold** mean the best results among all methods and the underlined ones mean the second best. The evaluation results of baselines are from their original work, and "-" in the table means there is no reported results in prior work. Note that we implement ProjE and PTransE on WN18 using the public codes.

From Table 2 we could observe that: (1) PTransE performs better than its basic model TransE, and RPE outperforms its original method TransR. This indicates that additional information from relation paths between entity pairs is helpful for link prediction. Note that OPTransE outperforms baselines which do not take relation paths into consideration in most cases. These results demonstrate the effectiveness of OPTransE to take advantage of the path features in the KG. (2) OPTransE performs better than previous path-based models like RTransE, PTransE, PaSKoGE and RPE on all metrics. This implies that the order of relations in paths is of great importance for reasoning, and learning representations of ordered relation paths can significantly improve the accuracy of link prediction. Moreover, the proposed pooling strategy which aims to extract nonlinear features from different relation paths also contributes to the improvements of performance.

Specific evaluation results on FB15K by mapping properties of relations (1-to-1, 1-to-N, N-to-1, and N-to-N) are shown in Table 3. Several methods which have reported these results are listed as baselines. OPTransE achieves the highest scores in all sub-tasks. We note that it is more difficult to predict head entities of N-to-1 relations and tail entities of 1-to-N relations, since the prediction accuracy on these two sub-tasks is generally lower than those of other sub-tasks. Surprisingly, OPTransE has achieved significant improvements on these two sub-tasks. Especially when predicting tail entities of 1-to-N relations, OPTransE promotes Hits@10 to 87.4% which is 8.3% higher than the best performance among

baselines. Meanwhile, since the average prediction accuracy for N-to-N relations of OPTransE on the two datasets has reached 91.1%, we can also infer that our model has strong ability to deal with N-to-N relations. OPTransE projects the head and tail entities of a triple into different relation-specific spaces, thus, it is able to better discriminate the relevant entities. Furthermore, these results also confirm that ordered relation paths between entity pairs which are exploited by OPTransE contain useful information and can help to perform more accurate inference when facing complex relations.

## 5 Conclusion and Future Work

In this paper, we propose a novel KG completion model named OPTransE, which aims to address the issue of relation orders in paths. In our model, we project the head entity and the tail entity of each relation into different spaces to guarantee the order of the path. In addition, a pooling method is applied to extract complex and nonlinear features from numerous relation paths. Finally, we evaluate our proposed model on two benchmark datasets and experimental results demonstrate the effectiveness of OPTransE.

In the future, we will explore the following research directions: (1) we will study the applications of the proposed models in various domains, like personalized recommendation (Liu et al., 2018); (2) we will explore other techniques to fuse the ordered relation information from different paths (Liu et al., 2019).

## Acknowledgments

This work was partially sponsored by National Key R&D Program of China (grant no. 2017YFB1002000).

## References

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, SIGMOD'08, pages 1247–1250. ACM.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2014. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94(2):233–259.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, NIPS'13, pages 2787–2795.

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*, AAAI'11, pages 301–306.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question answering over freebase with multi-column convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1 of *ACL-IJCNLP'15*, pages 260–269.

Alberto García-Durán, Antoine Bordes, and Nicolas Usunier. 2015. Composing relationships with translations. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, EMNLP'15, pages 286–290.

Kelvin Guu, John Miller, and Percy Liang. 2015. Traversing knowledge graphs in vector space. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, EMNLP'15, pages 318–327.

Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. 2016. Knowledge graph completion with adaptive sparse transfer matrix. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 985–991.

Yantao Jia, Yuanzhuo Wang, Xiaolong Jin, and Xueqi Cheng. 2018. Path-specific knowledge graph embedding. *Knowledge-Based Systems*, 151:37–44.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195.

Xixun Lin, Yanchun Liang, Fausto Giunchiglia, Xiaoyue Feng, and Renchu Guan. 2018. Relation path embedding in knowledge graphs. *Neural Computing and Applications*, pages 1–11.

Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015a. Modeling relation paths for representation learning of knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, EMNLP'15, pages 705–714.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015b. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 2181–2187.

Hanxiao Liu, Yuexin Wu, and Yiming Yang. 2017. Analogical inference for multi-relational embeddings. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, ICML'17, pages 2168–2178. JMLR. org.

Hongzhi Liu, Yingpeng Du, and Zhonghai Wu. 2019. Aem: Attentional ensemble model for personalized classifier weight learning. *Pattern Recognition*, 96:106976.

Hongzhi Liu, Zhonghai Wu, and Xing Zhang. 2018. Cplr: Collaborative pairwise learning to rank for personalized recommendation. *Knowledge-Based Systems*, 148:31–40.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Dat Quoc Nguyen, Kairit Sirts, Lizhen Qu, and Mark Johnson. 2016. Stranse: a novel embedding model of entities and relationships in knowledge bases. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, HLT-NAACL'16, pages 460–466.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2016. Holographic embeddings of knowledge graphs. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 1955–1961.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, HLT-NAACL'13, pages 74–84.

Baoxu Shi and Tim Weninger. 2017. Proje: embedding projection for knowledge graph completion. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, pages 1236–1242.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, WWW'07, pages 697–706. ACM.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. ICLR'19.

Kristina Toutanova, Victoria Lin, Wen-tau Yih, Hoifung Poon, and Chris Quirk. 2016. Compositional learning of embeddings for relation paths in knowledge base and text. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1 of *ACL'16*, pages 1434–1444.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, ICML'16, pages 2071–2080.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, pages 1112–1119.

Qizhe Xie, Xuezhe Ma, Zihang Dai, and Eduard Hovy. 2017. An interpretable knowledge transfer model for knowledge base completion. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1 of *ACL'17*, pages 950–962.

# Collaborative Policy Learning for Open Knowledge Graph Reasoning

**Cong Fu**[1*], **Tong Chen**[2*], **Meng Qu**[3], **Woojeong Jin**[4], **Xiang Ren**[4]
[1]Zhejiang University, [2]Carnegie Mellon University, [3]University of Montreal
[4]University of Southern California
fc731097343@gmail.com, tongc2@andrew.cmu.edu, meng.qu@umontreal.ca
{woojeong.jin, xiangren}@usc.edu

## Abstract

In recent years, there has been a surge of interests in interpretable graph reasoning methods. However, these models often suffer from limited performance when working on sparse and incomplete graphs, due to the lack of evidential paths that can reach target entities. Here we study *open knowledge graph reasoning*—a task that aims to reason for missing facts over a graph augmented by a background text corpus. A key challenge of the task is to filter out "*irrelevant*" facts extracted from corpus, in order to maintain an effective search space during path inference. We propose a novel reinforcement learning framework to train two collaborative agents jointly, i.e., a *multi-hop graph reasoner* and a *fact extractor*. The fact extraction agent generates fact triples from corpora to enrich the graph on the fly; while the reasoning agent provides feedback to the fact extractor and guides it towards promoting facts that are helpful for the interpretable reasoning. Experiments on two public datasets demonstrate the effectiveness of the proposed approach. Source code and datasets used in this paper can be downloaded at https://github.com/shanzhenren/CPL.

## 1 Introduction

Knowledge graph completion or reasoning—i.e., the task of inferring the missing facts (entity relationships) for a given graph—is an important problem in natural language processing and has a wide range of applications (Bordes et al., 2011; Socher et al., 2013; Trouillon et al., 2016). Recent neural graph reasoning methods, such as MINERVA (Das et al., 2017), DeepPath (Xiong et al., 2017) and Multi-Hop (Lin et al., 2018), have achieved impressive results on the task, offering both good *prediction accuracy* (compared to embedding-based methods (Trouillon et al., 2016; Dettmers et al., 2018)) and *interpretability* of the model predictions. These reasoning methods frame the link inference task as a path finding problem over the graph (see Fig. 1 for example).

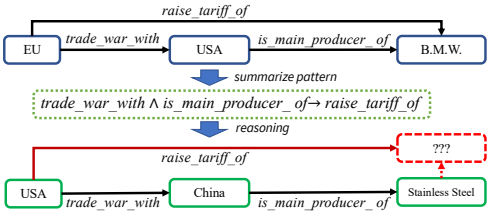


Figure 1: **Illustration of the Knowledge Graph Reasoning Task.** Given an entity (e.g., *Miami*) and a query relation (e.g., *located in*), we learn to infer reasoning paths over the existing graph structure to help predict the answer entity (i.e., *USA*).

However, current neural graph reasoning methods encounter two main challenges as follows: (1) their performance are often sensitive to the sparsity and completeness of the graph—missing edges (i.e., potential false positives) make it harder to find evidential paths reaching target entities. (2) existing models assume the graph is static, and cannot adapt to dynamically enriched graphs where emerging new facts are constantly added.

In this paper, we study the new task of Open Knowledge Graph Reasoning (OKGR), where the new facts extracted from the text corpora will be used to augment the graph dynamically while performing reasoning (as illustrated in Figure 2). All the recent joint graph and text embedding methods focus on learning better knowledge graph embeddings for reasoning (Xu et al., 2014; Han et al., 2018), but we consider adding more facts to the graph from the text to improve the reasoning performance and further provide interpretability. A straightforward solution for the OKGR problem is to directly add extracted facts (by a pre-trained relation extraction model) to the graph. However, most facts so extracted may be noisy or irrelevant

[*] Work done while the authors interned at USC.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2672–2681, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

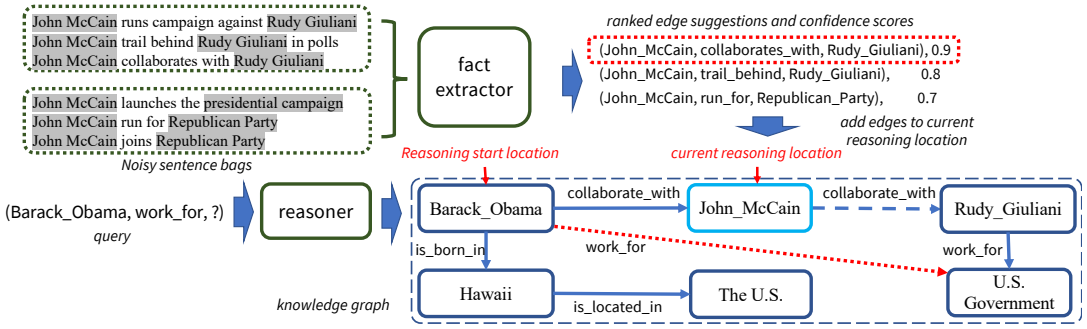


Figure 2: **Overview of our CPL framework for the OKGR problem.** To augment the reasoning with the information from a background corpus, CPL extracts relevant facts (e.g., the triple linked by the blue dotted arrow) to augment the KG dynamically. CPL involves two agents: one learns fact extraction policy to suggest relevant facts; the other learns to reason on dynamically augmented graphs to make predictions (e.g., the red dotted arrow).

to the path inference process. Moreover, adding a large number of edges to the graph will create an ineffective search space and cause scalability issues to the path finding models. Therefore, it is desirable to design a method that can filter out irrelevant facts for augmenting the reasoning model.

To address the above challenges for OKGR, we propose a novel *collaborative policy learning* (CPL) framework to jointly train two RL agents in a mutually enhancing manner. In CPL, besides training a *reasoning* agent for path finding, we further introduce a *fact extraction* agent, which learns the policy to select relevant facts extracted from the corpus, based on the context of the reasoning process and the corpus (see Fig. 2). At inference time, the fact extraction agent dynamically augments the graph with only the most informative edges, and thus enables the reasoning agent to identify positive paths effectively and efficiently.

Specifically, during policy learning, the reasoning agent will be rewarded when reaching the targets, while this positive feedback will also be transferred back to the fact extracting agent if its edge suggestions are adopted by the reasoning agent, i.e. making up correct reasoning paths. This ensures that the fact extraction policy can be learned in a way that edges which are beneficial to path inference will be preferred. By doing so, the fact extraction agent can learn to augment knowledge graphs dynamically to facilitate the reasoning agent, while the reasoning agent performs effective path-inference and provides reward signals to the fact extraction agent. Please refer to Sec. 3 and Fig. 3 for more implementation details.

The major contributions of our work are as follows: (1) We study knowledge graph reasoning in an "open-world" setting, where new facts extracted from background corpora can be used to facilitate path finding; (2) We propose a novel collaborative policy learning framework which models the interactions between fact extraction and graph reasoning; (3) Extensive experiments and analysis are conducted to demonstrate the effectiveness and strengths of our proposed method.

## 2 Background and Problem

This section introduces basic concepts and notations related to the knowledge graph reasoning task and provides a formal problem definition.

A *knowledge graph* (KG) can be represented by a set of triples (facts) $G = \{(e_s, r, e_o)|e_s, e_o \in E, r \in R\}$, where $E$ is the set of entities and $R$ is the set of relations. $e_s$, $r$, and $e_o$ are the subject entity, relation, and object entity respectively.

The task of *knowledge graph reasoning* (KGR) is defined as follows. Given the KG $G$, a query triple $(e_s, r_q, e_q)$ where $e_q$ is unknown, KGR is to infer $e_q$ through finding a path starting from $e_s$ to $e_q$ on $G$, namely $\{(e_s, r_1, e_1), ..., (e_n, r_n, e_q)\}$. Usually, KGR methods produce multiple candidate answers by ranking the found paths, while traditional KG completion methods rank all possible answer triples by exhaustively enumerating.

A *background corpus* is a set of sentences labeled with respective entity pairs, namely $C = \{(s_i : (e_k, e_j))|s_i \in S, e_k, e_j \in E\}$, where $S$ is the set of sentences, and the corpus shares the same entity set with $G$. In our problem setting, we assume the entities have already been extracted; thus, extracting facts from the corpus is equivalent to the relation extraction task. We process the corpus by labeling the sentences with subject and object entity pairs through Distant Supervision (Mintz et al., 2009). There may be many sen-

tences labeled with the same entity pair. Following the formulation of previous work (Lin et al., 2016), we organize the sentences into *sentence bags*, i.e., A sentence bag contains the sentences which are labeled with the same entity pair.

**Problem.** Formally, *Open Knowledge Graph Reasoning* (**OKGR**) aims to perform KGR based on both $G$ and $C$, where $G$ is dynamically enriched by the facts extracted from $C$. This paper focuses on OKGR, i.e., empowering KGR with the corpus information and enriching the graph with relevant facts dynamically. Thus, the evaluation of the relation extraction performance are out of the scope of this paper. We leave this as future work.

## 3 Proposed Framework

**Overview.** To resolve the challenges in OKGR, we propose a novel collaborative policy learning (CPL) framework (see Fig. 2), which jointly train two RL agents, i.e., a path reasoning agent and a fact extraction agent. Given a query $(e_s, r_q, e_q)$, the reasoning agent tries to infer $e_q$ via finding a reasoning path on the (augmented) $G$, while the fact extraction agent aims to select the most informative facts from $C$ to enrich $G$ dynamically. With such an extractor, the framework can effectively overcome the edge sparsity problem while remaining reasonably efficient (compared to the naive solution that adds all possible facts to $G$). We train the extraction agent by rewarding it according to the the reasoning agent's performance. Hence the fact extractor can learn how to extract the most informative facts to benefit the reasoning.

### 3.1 Graph Reasoning Agent

The goal of the reasoner is learning to reason via paths finding on KGs. Specifically, given $e_s, r_q$, the reasoner aims at inferring a path from $e_s$ to some entity $e_o$ regarding $r_q$, and specifying how likely the relationship $r_q$ holds between $e_s$ and $e_o$. The inference path acts as the evidence of the prediction, and thus offers interpretation (see Fig. 1). At each time step, the reasoner tries to select an edge based on the observed information. The Markov Decision Process (MDP) of the reasoner is defined as follows:

**State.** In path-based reasoning, each succeeding edge is closely related to the preceding edge on the path and the query in semantics. Similar to MINERVA(Das et al., 2017), we want the state to encode all observed information, i.e., we define $s_R^t = (e_s, r_q, h^t) \in \mathcal{S}_R$, where $h^t$ encodes the path history, and $(e_s, r_q)$ is the context shared among all states. Specifically, we use a LSTM module to encode the history, $h^t = \text{LSTM}(h^{t-1}, [r^t, e^t])$ (see Fig. 3). $e^t$ is the current reasoning location and $r^t$ is the previous relation connecting $e^t$.

**Action.** At time $t$, the reasoner will select an edge among $e^t$'s out-edges. The reasoner's action space is a union of the edges in the current KG and edges extracted from the corpus. See Sec. 3.3 for details.

**Transition.** The transition function $f : \mathcal{S}_R \times \mathcal{A}_R \rightarrow \mathcal{S}_R$ is defined as $f(s_R^t, a_R^t) = (e^s, r^q, h^{t+1})$ naturally.

**Reward.** The reasoner is expected to learn effective reasoning path patterns. We let it explore for a fixed number of steps, which is a hyper-parameter. Only when it reaches the correct target entity, it receives a terminal reward 1, and 0 otherwise. All intermediate states always receive reward 0.

### 3.2 Fact Extraction Agent

The fact extractor learns to suggest the most relevant facts w.r.t. the current inference step of the reasoner. Suppose the reasoner arrives at entity $e^t$ on the graph at time $t$, the fact extractor will extract facts in the form of $(e^t, r', e') \notin G$ from the corpus and add them to the graph temporarily. Consequently, the reasoner is offered more choices to expand the reasoning path.

**State.** When the reasoner is at $e^t$, the fact extractor tries to extract information from the corpus (sentences) and suggests promising out-edges of entity $e^t$. Let $b_{e^t}$ denote the sentence bags labeled with $(e^t, e'), e' \in E$. We define the state of the fact extractor to encode the current observed information, i.e., $s_E^t = (b_{e^t}, e^t) \in \mathcal{S}_E$, where $\mathcal{S}_E$ is the whole state space, containing all possible combinations of entities and corresponding sentence bags.

**Action.** The goal of the fact extractor is to select a reasoning-relevant fact contained in the corpus semantically. At step $t$, the reasoner will move to a new entity from $e^t$, and hence only the out-edges of $e^t$ should be considered, e.g., $(e^t, r', e')$ (see Fig. 3). Therefore, for each fact $(e^t, r', e')$ which can be extracted from the sentence bag $b_{e^t}$, we can derive an action $a_E^t = (r', e')$, and the action space at step $t$ can be denoted as $A_E^t = \{(r', e')\}_{(e^t, r', e') \in b_{e^t}} \subset \mathcal{A}_E$. $\mathcal{A}_E$ is the whole action space containing all facts in the corpus.

**Transition.** The transition function $f : \mathcal{S}_E \times \mathcal{A}_E \rightarrow \mathcal{S}_E$ is defined as $f(s_E^t, a_E^t) = (r', e')$.

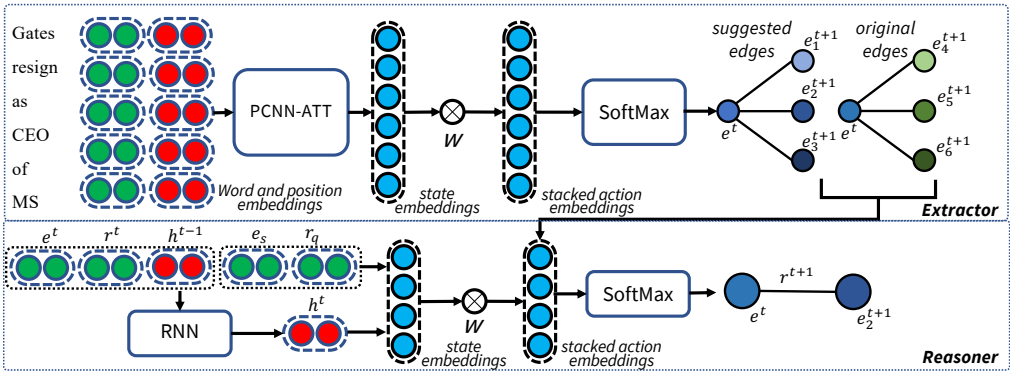


Figure 3: **Detailed Model Design of Collaborative Policy Learning (CPL).** Take PCNN-ATT as an example of the sentence encoder. The figure shows how it works at a certain inference time step $t$, the reasoner is at entity $e^t$ and will select one edge from the *joint action space*, which consists of new edges extracted by the extractor and the edges in the original graph.

**Reward.** The fact extractor receives a step-wise delayed reward from the reasoner according to how it improves the reasoners performance. The extractor will be positively rewarded when its suggestion benefits the reasoning process. Please see Sec. 3.3 for details.

### 3.3 Collaborative Policy Learning

In this section, we will introduce the detailed training process and the collaboration mechanism between the two agents. At the high level, we adopt an *alternative training procedure* to update the two agents jointly: training one of the agents for a few iterations while freezing the other; and vice versa. The policies of both agents are updated via REINFORCE algorithm (Williams, 1992) (details are in later of this section). Specifically, we introduce the details on agent collaboration as follows.

**Augmented Action Space for Reasoning.** At time $t$, the fact extractor helps the reasoner via expanding its action space with new edges extracted from the corpus (see Fig. 3). Due to the sparsity and incompleteness of the KG, there may be missing edges preventing the reasoner from inferring the correct reasoning path (Fig. 2). Therefore, we add high-confidence edges extracted by the extractor to the action space of the reasoner. Formally, at time $t$, the reasoner is at location $e^t$ (Fig. 3) and tries to select an edge out of all out-edges of $e^t$. Let $A_K^t$ denote the edge set in the current KG, $A_K^t = \{(r, e)|(e^t, r, e) \in G\}$. Let $A_C^t$ denote the edge set suggested by the extractor, $A_C^t = \{(r', e')|(e^t, r', e') \in C\}$. The action space at time $t$ of the reasoner is defined as $A_R^t = A_K^t \cup A_C^t$, $A_R^t \subset \mathcal{A}_R$, where $\mathcal{A}_R$ denotes the whole action space of the reasoner, i.e., all possible edges in the KG and the corpus. The reasoner learns a policy to select the best edges out of the joint action space for reasoning.

**Reasoning Feedback for Fact Extraction.** The reasoner helps the extractor to learn the extracting policy through providing feedbacks regarding how much the extractor contributes to the reasoning. Therefore we define that the fact extractor receives a step-wise delayed reward from the reasoner. Specifically, when the reasoner finishes exploration (at time $T$) and arrives at the correct target, we consider the path is effective and positive for reasoning. If the fact extractor contributes to this positive path, it can be rewarded positively, i.e., if an edge on the positive path is suggested by the extractor at time $t, 0 \leq t \leq T$, the extractor will be rewarded 1 at time $t$, and 0 otherwise. Extracted edges triggering positive rewards will be kept in the graph, while the others will be removed when both agents move to the next state.

**Policy Update.** The MDPs of both agents are explicitly defined above now, and we can use the typical REINFORCE algorithm (Williams, 1992) to train the two agents. Specifically, their goals are maximizing the reward expectation, defined as

$$J(\theta) = \mathbb{E}_{\pi_\theta(a|s)}[R(s, a)], \tag{1}$$

where $R(s, a)$ is the reward of selecting $a$ given $s$, and $\pi_\theta(a|s)$ is the policy learned by the agents and will be defined formally in Section. 4.

Given a training sequence sampled from $\pi_\theta$: $\{(s^1, a^1, r^1), ..., (s^T, a^T, r^T)\}, r^t = R(s^t, a^t)$ , at time step $t$, the parameters are updated according to the REINFORCE algorithm:

$$\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(a^t|s^t) G^t$$
$$G^t = \sum_{k=t}^{T} \gamma^{k-t} R(s^k, a^k), \quad (2)$$

where $G^t$ is the discounted accumulated reward.

According to Eq. (2), we can see that REINFORCE will update the parameters only when $G^t$ is non-zero. In other words, the value of $\gamma$ determines how the parameters are updated and to what extent the the internal states will be influenced by the future. If $\gamma > 0$, for positive training sequences, it is easy to verify that the $G^t$ of all states will be non-zero. Thus, the internal states will be positively rewarded, and the model parameters will be updated by the gradients of the internal states. For different task, we should carefully select the value of $\gamma$. For the extractor, we set $\gamma = 0$ for the extractor to avoid policy updating on zero-rewarded state-action experiences. This is because zero-rewarded experiences are mostly negative examples. Specifically, if a state of the extractor is zero-rewarded, we can infer that either the suggested edge is not selected by the reasoner, or the selected edge does not contribute to reaching the target. We can not allow the model to be updated on such experiences, so we set $\gamma = 0$ to avoid the influence of future. In contrast, we set $\gamma = 1$ for the reasoner because all the intermediate selected edges are meaningful as long as it leads to the target finally.

## 4 Model Implementation

In this section, we introduce the policy network architectures (cf. Fig. 3) of the two agents and provide details on model training and inference.

### 4.1 Policy Network Architectures

**Reasoning Agent.** We construct the state embedding by concatenating all related embeddings, $s_R^t = [e_s, r_q, h^t]$, where $h^t = \text{LSTM}(h^{t-1}, [r^t, e^t])$. We construct the action embedding by concatenating the relation-entity embedding pair, i.e., $a_R^t = [r, e], (e^t, r, e) \in G \cup C$. We stack all action embeddings in $\mathbf{A}_R^t$. The policy network is defined as:

$$\pi_\theta(a_R^t|s_R^t) = \sigma(\mathbf{A}_R^t W_2(Relu(W_1[e_s, r_q, h^t]))),$$

$\sigma$ is softmax, and $W_1$, $W_2$ are learnable weights.

**Fact extraction Agent.** We use a PCNN-ATT as the sentence encoder in our experiments to construct the distributed representations for the sentences. Let $b_{e^t}$ denote all the sentence bags labeled by $(e^t, e'), e' \in E$. At time $t$, we input $b_{e^t}$ into the PCNN-ATT to obtain the sentence-bag-level embeddings $E_b^t$, which is regarded as the latent state embeddings. As mentioned in Sec. 2, the object entity has been labeled beforehand. We need to select the best relation first, then select the best entity under this relation. Thus, we stack the relation embeddings in $A_E^t$ as $\mathbf{A}_E^t \in \mathbb{R}^{|A_E^t| \times d}$, where $d$ is the dimension of relation embedding. The policy network is defined formally as:

$$\pi_\theta(a_E^t|s_E^t) = \sigma(\mathbf{A}_E^t W E_b^t),$$

where $W$ is a learnable weight.

The extractor will predict the scores for each sentence bag regarding the relation. We will select the sentence bag with the highest score, more formally, the corresponding relation-entity pair in that sentence bag will be chosen as the next action. We train the agents as introduced in Sec. 3.3.

### 4.2 Model Training and Inference

**Training.** We use *model pre-training* and *adaptive sampling* to increase training efficiency. In particular, we first train the reasoner on the original KG to get a better initialization. Similarly, we train the extractor on the corpus labeled by distant supervision. Next, we use adaptive sampling to adaptively increase the selecting-priority of corpus-extracted-edges when generating training experiences for the two agents. Adaptive sampling is designed to encourage the reasoner to explore more on new edges and facilitate the collaboration during the joint training. Replay memories (Mnih et al., 2013) are also used to increase training efficiency. We develop several model variants such as removing adaptive sampling or replay memory, or freezing the extractor all the time to conduct ablation studies. Please see Supplementary Material for more details.

**Inference.** At inference (reasoning) time, we use the trained model to predict missing facts via path finding. The process is similar to the training experience generation step in the training stage, i.e. using the reasoner for path-inference while the extractor suggests edges from the corpus constantly. The only differences are that we do not request rewards, and we use beam search to generate multiple reasoning paths over the graph, and rank them by the scores from the reasoner.

| Dataset | #triples(C) | #triple(G) | #entities(C) | #entities(G) | #rel(C) | #rel(G) | S(train) | S(test) | CT/CE | CR/KR |
|---|---|---|---|---|---|---|---|---|---|---|
| FB60K-NYT10 | 172,448 | 268,280 | 63,696 | 69,514 | 57 | 1,327 | 570k | 172k | 2.71 | 0.04 |
| UMLS-PubMed | 910,320 | 2,030,841 | 6,575 | 59,226 | 271 | 443 | 4,73M | 910k | 138.45 | 0.61 |

Table 1: **The dataset information.** #triples(C) & #triples(G) denote the number of triples in the corpus and the KG respectively, and so on. S(train) denotes the number of sentences in the training corpus, while S(test) denotes the number of sentences in the testing corpus. CT/CE denotes triple-entity ratio. Lower triple-entity ratio indicates less triples per entity in average can be extracted from the corpus. CR/KR denotes corpus-relation-quantity/KG-relation-quantity ratio. Lower CR/KR indicates less information overlap between the corpus and the KG.

# 5 Experiment

## 5.1 Datasets and Compared Methods

| Dataset | Relations |
|---|---|
| UP | *'gene_associated_with_disease'*<br>*'disease_has_associated_gene'*<br>*'gene_mapped_to_disease'*<br>*'disease_mapped_to_gene'*<br>*'may_be_treated_by'*<br>*'may_treat'*<br>*'may_be_prevented_by'*<br>*'may_prevent'* |
| FN | *'people/person/nationality'*<br>*'location/location/contains'*<br>*'people/person/place_lived'*<br>*'people/person/place_of_birth'*<br>*'people/deceased_person/place_of_death'*<br>*'people/person/ethnicity'*<br>*'people/ethnicity/people'*<br>*'business/person/company'*<br>*'people/person/religion'*<br>*'location/neighborhood/neighborhood_of'*<br>*'business/company/founders'*<br>*'people/person/children'*<br>*'location/administrative_division/country'*<br>*'location/country/administrative_divisions'*<br>*'business/company/place_founded'*<br>*'location/us_county/county_seat'* |

Table 2: **The concerned relations in two datasets.** UP means the UMLS-PubMed dataset, while FB means the FB60K-NYT10 dataset.

**Datasets.** We construct two datasets for evaluation: *FB60K-NYT10*[1] and *UMLS-PubMed*[2]. FB60K-NYT10 dataset includes the FB-60K KG and the NYT10 corpus; The UMLS dataset contains the UMLS KG and the PubMed corpus. Statistics of both datasets are summarized in Table 1. We study the datasets and find that the relation distributions of the two datasets are very imbalanced. There are not enough reasoning paths for some relation types. Moreover, some relations are meaningless and of no reasoning value. Thus, we select a few meaningful and valuable relations (suggested by domain experts, in Table 2) with enough reasoning paths and construct two subgraphs accordingly. To show the impact of graph size, we sub-sample the KG into different subgraph. Specifically, for the two datasets, we first partition the whole KG into three parts according to the proportion 8:1:1 (The training set, validation set, and testing set). Next we create sub-train-sets with different ratio via random sampling.

**Analysis of corpus-KG alignment.** We analyze the information overlap (i.e., alignment) between the corpus and the KG in Table 1. The CT/CE (the ratio of triple quantity against entity quantity) of PubMed is far higher than NYT10. Higher CT/CE indicates adding corpus-edges to the KG increases the average degree more significantly, leading to more reduction in sparsity. The low CR/KR ratio of FB60K-NYT10 indicates the overlap between FB60K and NYT10 is lower than that between UMLS and PubMed. We can conclude that the alignment level of FB60K-NYT10 is lower than UMLS-PubMed. Intuitively, FB60K-NYT10 is a more difficult dataset than UMLS-PubMed.

**Compared Algorithms.** We compare our algorithm with (1) SOTA methods for KG embedding ; (2) methods for joint text and graph embedding; and (3) neural graph reasoning methods.

For triple-ranking-based KG embedding methods, we evaluate **DistMult** (Yang et al., 2014), **ComplEx** (Trouillon et al., 2016), and **ConvE** (Dettmers et al., 2018). For joint text and graph embedding methods, we evaluate **RC-Net** (Xu et al., 2014) and **Joint-NRE** (Han et al., 2018). We also construct a baseline, **TransE+LINE**, by constructing a word-entity co-occurrence network as RC-Net does. We use LINE (Tang et al., 2015) and TransE (Bordes et al., 2011) to jointly learn the entity and relation embeddings to preserve the structure information within the co-occurrence network and the KG. For neural graph reasoning method, we use **MINERVA** (Das et al., 2017), a reinforcement learning based path reasoning method[4].

[1] https://github.com/thunlp/OpenNRE

[2] http://umlsks.nlm.nih.gov/ https://www.ncbi.nlm.nih.gov/pubmed/

[4] There are other SOTA path-based knowledge reasoning methods such as Multi-Hop (Lin et al., 2018) and DeepPath (Xiong et al., 2017) However, DeepPath needs extra path-level supervisions which we do not have and Multi-Hop suf-

| Model / Dataset | 20% | | 40% | | 70% | | 100% | |
|---|---|---|---|---|---|---|---|---|
| | Hits@5 | Hits@10 | Hits@5 | Hits@10 | Hits@5 | Hits@10 | Hits@5 | Hits@10 |
| TransE (Bordes et al., 2011) | 7.12 | 11.17 | 26.86 | 38.08 | 31.32 | 43.58 | 32.28 | 45.52 |
| DisMult (Yang et al., 2014) | 14.66 | 21.16 | 26.90 | 38.35 | 31.65 | 44.98 | 32.80 | 47.50 |
| ComplEx (Trouillon et al., 2016) | 18.58 | 18.18 | 23.77 | 34.15 | 30.04 | 43.60 | 31.84 | 46.57 |
| ConvE (Dettmers et al., 2018) | 20.51 | 30.11 | **28.01** | **42.04** | 31.01 | 45.81 | 30.35 | 45.35 |
| RotatE (Sun et al., 2019) | 4.03 | 6.50 | 8.65 | 13.21 | 14.90 | 21.67 | 20.75 | 27.82 |
| RC-Net (Xu et al., 2014) | 7.94 | 10.77 | 7.56 | 11.43 | 8.31 | 11.81 | 9.26 | 12.00 |
| TransE+LINE | **23.63** | **31.85** | 24.86 | 38.58 | 25.43 | 34.88 | 22.31 | 33.65 |
| JointNRE (Han et al., 2018) | 21.05 | 31.37 | 27.96 | 40.10 | 30.87 | 44.47 | - | - |
| MINERVA (Das et al., 2017) | 11.55 | 19.87 | 24.65 | 35.71 | 35.8 | 46.26 | 57.63 | 63.83 |
| Two-Step | 8.37 | 13.5 | 22.75 | 32.79 | 33.14 | 43.35 | 55.59 | 63.49 |
| CPL (our method) | 15.32 | 24.22 | 26.96 | 38.03 | **37.23** | **47.60** | **58.10** | **65.16** |

Table 3: **Performance comparison on the KG reasoning on the UMLS-PubMed dataset.**[3] We test on different graph sizes (i.e., 20-100% of the original graph) using Hits@K (in %). CPL is the best performing graph reasoning method, and gradually outperforms the others when the graphs are denser.

To validate the effectiveness of fact extraction policy in CPL, we design a two-step baseline (i.e., **Two-Step**). It first uses PCNN-ATT to extract relational triples from the corpora, and augments KG with the triples whose prediction confidences are greater than a threshold. PCNN-ATT (Lin et al., 2016) is a fact extraction model, which completes the fact extraction part. We tune the threshold on the dev-set. Then, a MINERVA model is trained on the augmented KG for reasoning.

**CPL** is our full-fledged model as introduced in Sec. 3. For all the methods, we upload the source codes and list hyper-parameters we used in the supplemental materials.

### 5.2 Evaluation and Experimental Setup

Following previous work on KG completion (Bordes et al., 2011), we use Hits@K and mean reciprocal rank (MRR) to evaluate the effectiveness of the KGR and OKGR. Given a query $(e_s, r, ?)$ (for each triple in the test set, we pretend not to know the object entity), we rank the correct entity $e_q$ among a list of candidates entities. Suppose $rank_i$ is the rank of the correct answer entity for the $i^{th}$ query. We define $Hit@K = \sum_i \mathbf{1}(rank_i < K)/N$ and $MRR = \frac{1}{N}\sum_i \frac{1}{rank_i}$.

In our experiments, we use a held-out validation set for all compared methods to search for the best hyper-parameters and the best model for testing (via logging checkpoints). For all methods, we train models using three fixed random seeds (55, 83, 5583), and report the metrics in average. More details on model training can be found in the Supplementary Material.

fers from out-of-memory problems on large-scale datasets.

### 5.3 Performance Comparison

Performances of the KG reasoning of all the algorithms are given in Table 3, 4 and Figure 6. We can draw conclusions as follows:

**1. Triple ranking vs. path inference.** CPL and MINERVA perform worse than triple-ranking methods when the size of KGs is small, while outperforms them significantly when adding more triples to the KGs (Figure 6). This is because the general and evidential paths for reasoning on sparse KGs are not enough, and path-based models cannot capture the underlying patterns.

**2. CPL vs. joint embedding methods.** CPL is inferior to RC-net, TransE+Line, and JointNRE on small KG partitions because they are not path-based models and the connections on small KGs are too sparse. CPL outperforms them significantly on larger datasets. The reasons are two-fold : 1) the graphs are denser to provide enough reasoning paths for training; 2) other algorithms do not filter noisy text information in joint-training.

**3. CPL vs. other graph reasoning methods.** CPL outperforms MINERVA significantly because CPL makes use of relevant text information for prediction. MINERVA is better than CPL on full FB60K-NYT10 because the alignment between FB60K and NYT10 is very limited (Sec. 5.1). The graph is dense at 100%, and the benefits from the corpus information are indiscernible.

### 5.4 Performance Analysis

**1. Ablation Study on Model Components.** In CPL, we apply multiple learning techniques to improve the performance, including collaboration, replay memory, and adaptive sampling as introduced in Sec. 4. To show the effects of differ-

| Model / Dataset | 20% | | | 50% | | | 100% | | |
|---|---|---|---|---|---|---|---|---|---|
| | Hits@5 | Hits@10 | MRR | Hits@5 | Hits@10 | MRR | Hits@5 | Hits@10 | MRR |
| TransE (Bordes et al., 2011) | 15.12 | 18.83 | 12.57 | 19.38 | 23.2 | 13.36 | 38.53 | 43.38 | 29.90 |
| DisMult (Yang et al., 2014) | 1.42 | 2.55 | 1.05 | 15.23 | 19.05 | 12.36 | 32.11 | 35.88 | 24.95 |
| ComplEx (Trouillon et al., 2016) | 4.22 | 5.97 | 3.44 | 19.10 | 23.08 | 12.99 | 32.91 | 34.62 | 24.67 |
| ConvE (Dettmers et al., 2018) | **20.6** | **26.9** | 11.96 | 24.39 | 30.59 | 18.51 | 33.02 | 39.78 | 24.45 |
| RotatE (Sun et al., 2019) | 9.25 | 11.83 | 8.04 | 25.96 | 31.63 | 23.34 | **58.32** | **60.66** | **51.85** |
| RC-Net (Xu et al., 2014) | 13.48 | 15.37 | **13.26** | 14.87 | 16.54 | 14.63 | 14.69 | 16.34 | 14.41 |
| TransE+Line | 12.17 | 15.16 | 4.88 | 21.7 | 25.75 | 8.81 | 26.76 | 31.65 | 10.97 |
| JointNRE (Han et al., 2018) | 16.93 | 20.74 | 11.39 | 26.96 | 31.54 | 21.24 | 42.02 | 47.33 | 32.68 |
| MINERVA (Das et al., 2017) | 11.64 | 14.16 | 8.93 | 25.16 | 31.54 | 22.24 | 43.80 | 44.70 | 34.62 |
| Two-Step | 12.14 | 16.5 | 9.27 | 21.66 | 31.50 | 19.82 | 39.22 | 44.64 | 34.18 |
| CPL (our method) | 15.19 | 18.00 | 10.87 | **26.81** | **31.7** | **23.80** | 43.25 | 49.50 | 33.52 |

Table 4: **Performance comparison on the KG reasoning on the FB60K-NYT10 dataset.** We can observe similar performance trends as those on the UMLS-PubMed dataset.

ent components, we remove them one by one and train the respective model variants. In addition, to shown the effect of collaboration, we train a model variation with the parameters of the extractor frozen. The result is shown in Fig. 5.

From the result, we find that 1) replay memory is only effective when adaptive sampling is also enabled. This is because adaptive sampling solves the sparse positive sample problem to some extent. There are enough positive experiences for replay. 2) Collaboration improves performance significantly. CPL with a trainable extractor performs better than with a frozen extractor, which means the suggestions of the extractor can be improved by the reasoner's feedbacks. 3) The improvement of CPL over MINERVA reduces as we increase the KG size. This is because with more data for training, the graph becomes denser, and hence the contribution from texts will be diluted.

**2. Effectiveness of Fact Selection.** As mentioned above, Two-Step is the naive solution to OKGR. The best performing Two-Step model adds tens times more edges into the KG than CPL, whereas the Two-Step model's performance is inferior to CPL and MINERVA on all the datasets (Table 3, 4). The reasons are 1) most of the extracted edges used in the Two-Step model are noisy; 2) adding so many edges significantly enlarges the exploration space for reasoning.

**2.** We perform a case study on the FB60K-NYT10 dataset to show the effectiveness of dynamically fact-filtering. We check the reasoning performance of the MINERVA and CPL periodically during the training. The results show that the extractor's contribution increases along with the training progress and the adaptive sampling can generate sufficient positive training experiences at the very beginning.

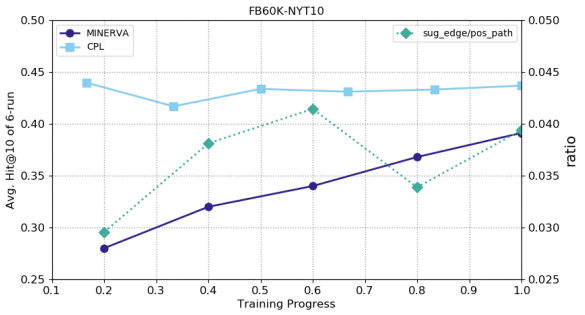


Figure 4: **KG reasoning performance change w.r.t. time.** sug_edge/pos_path means the ratio of positive edges suggested by the extractor w.r.t. the positive paths found by the reasoner.

The result is shown in Figure 4. We find a few interesting points as follows: **1)** the sug_edge/pos_path ratio curve in Figure 4 suggests that the extractor's contribution increases along with the training progress; **2)** CPL has a high initial performance because the adaptive sampling generates sufficient positive training experiences quickly. **3)** The valley shape in the performance curve is because the agent has not learned a stable exploring policy when the adaptive sampling stops, and the adaptive sampling somehow twisted the true pattern distribution in the dataset. But with a good start, the agent can explore on its own to approach the true distribution.

### 5.5 Case Study of Reasoning Paths

We randomly sample some reasoning paths from the inference results of CPL as examples. Due to the space limit, please refer to the supplemental materials for these examples. These examples show 1) how the reasoner finds the path patterns for the respective relations; 2) how the reasoner finds the inference paths according to the patterns; 3) how the extractor suggests rel-

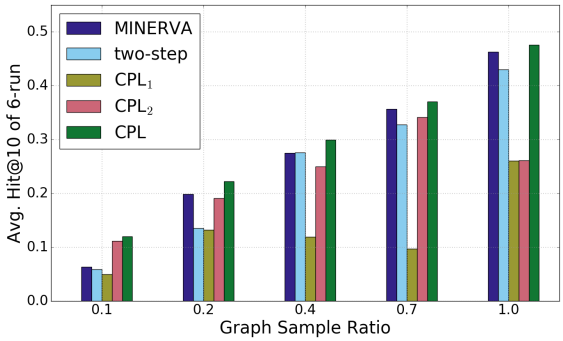


Figure 5: **Ablation Study on UMLS-PubMed dataset.** $CPL_1$ denotes CPL without adaptive sampling, and the extractor is frozen during training. $CPL_2$ denotes CPL without adaptive sampling. CPL denotes our proposed final model (with all the components).

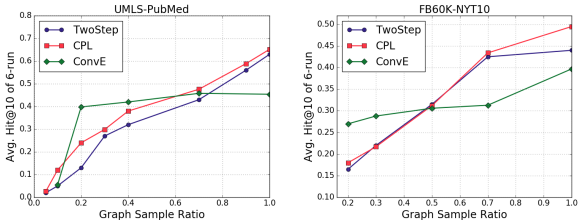


Figure 6: **KG reasoning performance change w.r.t. the size of the graph.** Triple-ranking based methods perform pretty well on smaller partitions, but are soon surpassed by path-based KG reasoning methods with the increase of graph size.

evant edges for each positive paths; 4) how the extractor extracts the relevant facts from related sentences. In summary, these cases show how CPL performs interpretable knowledge graph reasoning (infer the query entity through semantics-related path searching) and how CPL performs interpretable fact-filtering (suggest edges w.r.t the learned reasoning path patterns).

## 6 Related Work

**Knowledge Graph Reasoning.** Diverse approaches of embedding-based KG reasoning are presented, including linear models (Bordes et al., 2011), latent factor models (Trouillon et al., 2016), matrix factorization models (Yang et al., 2014) and convolutional neural networks (Dettmers et al., 2018). Performances of these methods are promising, but their predictions are barely interpretable. RL plays a crucial role in interpretable KG reasoning. MINERVA (Das et al., 2017) and DeepPath (Xiong et al., 2017) employ policy networks; (Xiong et al., 2017) uses extra rule-based supervision. Multi-hop (Lin et al., 2018) improves MINERVA via reward shaping and action drop-out.

**Joint Embedding of Text and KG.** Joint embedding methods aim to unite text corpus and KG. Contrary to our focus, they mainly utilize KGs for better performances of other tasks. (Toutanova et al., 2015) focuses on fact-extraction on the corpus labeled via dependency parsing with the aids of KG and word embeddings, while (Han et al., 2016) conducts the same task with the raw corpus text. As a newer joint model developed from (Han et al., 2016), (Han et al., 2018) deals with fact extraction by employing the mutual attention.

**Open-World KG Completion.** There are works focusing on similar topics as ours. (Shi and Weninger, 2018) defines an Open World KG Completion problem, in which they complete the KG with unseen entities. (Friedman and Broeck, 2019) introduces the Open-World Probabilistic Databases, an analogy to KGs. Unlike our setting, they try to complete the KG with logical inferences without extra information. (Sun et al., 2018) proposes an open, incomplete KB environment (or KG) with text corpora, but they focus on extracting answers from question-specific subgraphs.

## 7 Conclusion

In this paper, we focus on a new task named as Open Knowledge Graph Reasoning, which aims at boosting the knowledge graph reasoning with new knowledge extracted from the background corpus. We propose a novel and general framework, namely Collaborative Policy Learning, for this task. CPL trains two collaborative agents, the reasoner and fact extractor, which learns the path-reasoning policy and relevant-fact-extraction policy respectively. CPL can perform efficient interpretable reasoning on the KG and filtering of noisy facts. Experiments on two large real-world datasets demonstrate the strengths of CPL. Our work can cope with different path-finding modules such as MultHop with reward shaping by ConvE or RotatE and thus can improve its performance as the module improves.

## 8 Acknowledgement

This work has been supported in part by National Science Foundation SMA 18-29268, DARPA MCS and GAILA, IARPA BETTER, Schmidt Family Foundation, Amazon Faculty Award, Google Research Award, Snapchat Gift, JP Morgan AI Research Award, and China Scholarship Council. We would like to thank all the collaborators for their constructive feedbacks.

## References


Antoine Bordes, Jason Weston, Ronan Collobert, Yoshua Bengio, et al. 2011. Learning structured embeddings of knowledge bases. In *AAAI*, volume 6, page 6.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2017. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. *arXiv preprint arXiv:1711.05851*.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Tal Friedman and Guy Van den Broeck. 2019. On constrained open-world probabilistic databases. *arXiv preprint arXiv:1902.10677*.

Xu Han, Zhiyuan Liu, and Maosong Sun. 2016. Joint representation learning of text and knowledge for knowledge graph completion. *arXiv preprint arXiv:1611.04125*.

Xu Han, Zhiyuan Liu, and Maosong Sun. 2018. Neural knowledge acquisition via mutual attention between knowledge graph and text. In *Proceedings of AAAI*.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2018. Multi-hop knowledge graph reasoning with reward shaping. *arXiv preprint arXiv:1808.10568*.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2124–2133.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, pages 1003–1011, Stroudsburg, PA, USA. Association for Computational Linguistics.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Baoxu Shi and Tim Weninger. 2018. Open-world knowledge graph completion. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems*, pages 926–934.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William W Cohen. 2018. Open domain question answering using early fusion of knowledge bases and text. *arXiv preprint arXiv:1809.00782*.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077. International World Wide Web Conferences Steering Committee.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, pages 2071–2080.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. *arXiv preprint arXiv:1707.06690*.

Chang Xu, Yalong Bai, Jiang Bian, Bin Gao, Gang Wang, Xiaoguang Liu, and Tie-Yan Liu. 2014. Rc-net: A general framework for incorporating knowledge into word representations. In *Proceedings of the 23rd ACM international conference on conference on information and knowledge management*, pages 1219–1228. ACM.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*.

# Modeling Event Background for If-Then Commonsense Reasoning Using Context-aware Variational Autoencoder

**Li Du, Xiao Ding, Ting Liu* and Zhongyang Li**
Research Center for Social Computing and Information Retrieval
Harbin Institute of Technology, China
{ldu, xding, tliu, zyli}@ir.hit.edu.cn

## Abstract

Understanding event and event-centered commonsense reasoning are crucial for natural language processing (NLP). Given an observed event, it is trivial for human to infer its intents and effects, while this type of If-Then reasoning still remains challenging for NLP systems. To facilitate this, a If-Then commonsense reasoning dataset Atomic is proposed, together with an RNN-based Seq2Seq model to conduct such reasoning. However, two fundamental problems still need to be addressed: first, the intents of an event may be multiple, while the generations of RNN-based Seq2Seq models are always semantically close; second, external knowledge of the event background may be necessary for understanding events and conducting the If-Then reasoning. To address these issues, we propose a novel context-aware variational autoencoder effectively learning event background information to guide the If-Then reasoning. Experimental results show that our approach improves the accuracy and diversity of inferences compared with state-of-the-art baseline methods.

## 1 Introduction

Recently, event-centered commonsense knowledge has attracted much attention (Chambers and Jurafsky, 2008; Segers et al., 2016; Wang et al., 2017; Li et al., 2018), because of understanding events is an important component of NLP. Given a daily-life event, human can easily understand it and reason about its causes, effects, and so on. However, it still remains a challenging task for NLP systems. This is partly due to most of them are trained for task-specific datasets or objectives, which results in models that are adapt at finding task-specific underlying correlation patterns but have limited capability in simple and explainable commonsense reasoning (Sap et al., 2018).

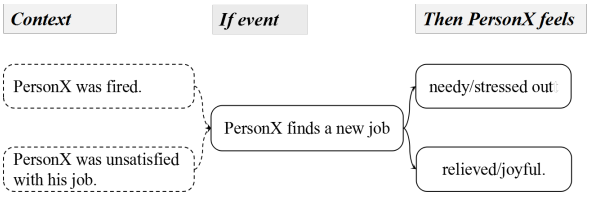


Figure 1: A illustration of two challenging problems in If-Then reasoning. (a) Given an observed event, the feelings about this event could be multiple. (b) Background knowledge is need for generating reasonable inferences, which is absent in the dataset (marked by dashed lines).

To facilitate this, Rashkin et al. (2018) build the Event2Mind dataset and Sap et al. (2018) present the Atomic dataset, mainly focus on nine If-Then reasoning types to describe causes, effects, intents and participant characteristic about events. Together with these datasets, a simple RNN-based encoder-decoder framework is proposed to conduct the If-Then reasoning.

However, there still remains two challenging problems. First, as illustrated in Figure 1, given an event "PersonX finds a new job", the plausible feeling of PersonX about that event could be multiple (such as *"needy/stressed out"* and *"relieved/joyful"*). Previous work showed that for the one-to-many problem, conventional RNN-based encoder-decoder models tend to generate generic responses, rather than meaningful and specific answers (Li et al., 2016; Serban et al., 2016).

Second, as a commonsense reasoning problem, rich background knowledge is necessary for generating reasonable inferences. For example, as shown in Figure 1, the feeling of PersonX upon the event "PersonX finds a new job" could be multiple. However, after given a context "*PersonX was fired*", the plausible inferences would be narrowed down to "*needy*" or "*stressed out*".

To better solve these problems, we propose a context-aware variational autoencoder (CWVAE) together with a two-stage training procedure. Vari-

*Corresponding author



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2682–2691,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

ational Autoencoder (VAE) based models have shown great potential in modeling the one-to-many problem and generate diversified inferences (Bowman et al., 2015; Zhao et al., 2017).

In addition to the traditional VAE structure, we introduces an extra context-aware latent variable in CWVAE to learn the event background knowledge. In the pretrain stage, CWVAE is trained on an auxiliary dataset (consists of three narrative story corpora and contains rich event background knowledge), to learn the event background information by using the context-aware latent variable. Subsequently, in the finetune stage, CWVAE is trained on the task-specific dataset to adapt the event background information to each specific aspect of If-Then inferential target (e.g., intents, reactions, etc.).

Experiments on the Event2Mind and Atomic dataset show that our proposed approach outperforms baseline methods in both the accuracy and diversity of inferences. The code is released at https://github.com/sjcfr/CWVAE.

## 2 Background

Before specifically describing two dataset —- Event2Mind and Atomic used in this paper as well as the If-Then reasoning task, for clarity, we define the following terminologies:

**Base event:** the prerequisite event in If-Then reasoning, organized as a verb phrase with a predicate and its arguments, such as the event "PersonX finds a new job" shown in Figure 1.

**Inference dimension:** a particular If-Then reasoning type, e.g., intents, effects of the base event. Details are shown in Table 1 and Table 2.

**Target:** the inferential results. For example, as shown in Figure 1, given a base event "PersonX finds a new job" and one inference dimension "xReact", the targets could be "*relieved*" or "*needy*". Notice that each inference dimension can have multiple targets.

**Event2Mind Dataset** contains 25K base events and 300K targets, annotated through crowdsourcing. Event2Mind is organized in a hierarchical form: each base event has three types of inference dimensions, and given a base event, under one of inference dimensions, several targets may simultaneously exist. Table 1 shows the (base event-inference dimension-target) hierarchical structure through an example from Event2Mind.

**Atomic Dataset** Inspired by Event2Mind, the Atomic dataset shares the same hierarchical structure as Event2Mind, while scales up the size of dataset and expands the scope to nine types of inference dimensions. Table 2 shows the (base event-inference dimension-target) hierarchical structure through an example from Atomic. Though Atomic covers the inference dimensions of Event2Mind, the base event collection of Event2Mind is nonidentical to that of Atomic.

| Base event | Inference Dim. | Target |
|---|---|---|
| PersonX writes PersonY a letter | xIntent | to send a message, express themself |
| | xReact | nervous, thoughtful |
| | oReact | indifferent, receptive |

Table 1: Hierarchical structure of Event2Mind dataset. For specific inference dimensions, "x" and "o" refers to PersonX and others respectively.

| Base event | Inference Dim. | Target |
|---|---|---|
| PersonX adopts a child | xIntent | to help another person, to have a child |
| | xNeed | to visit adoption agency, to be approved for adoption |
| | xAttr | compassionate, generous |
| | xEffect | becomes a parent, gains love and companionship |
| | xWant | take child home, buy child clothes |
| | xReact | happy, caring |
| | oReact | has a parent, receives love and affection |
| | oWant | try on new clothes, to have a family |
| | oEffect | has a parent, Receives love and affection |

Table 2: Hierarchical structure of Atomic dataset. For specific inference dimensions, "x" and "o" refers to PersonX and others respectively.

**Problem Definition** The If-Then reasoning task could be formally defined as a conditional one-to-many generation problem: given a base event $x$ and one inference dimension $d$, the model is required to generate targets $y = f(x, d)$ as close to the ground truths as possible. Both $x$ and $y$ consist of sequence of words: $x = \{x_1, \ldots, x_m\}$, and $y = \{y_1, \ldots, y_n\}$, where $m$ and $n$ denotes the length of $x$ and $y$, respectively.

**Conditional Variational Autoencoder** The variational autoencoder (VAE) defines a generative framework suited for one-to-many generation problem (Kingma and Welling, 2014). While con-

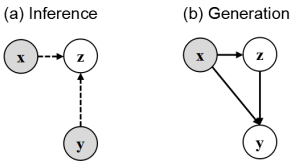


Figure 2: Illustration of inference and generation process of CVAE in a directed graph. Dashed lines represent the inference of $z$. Solid lines represent the generation process.

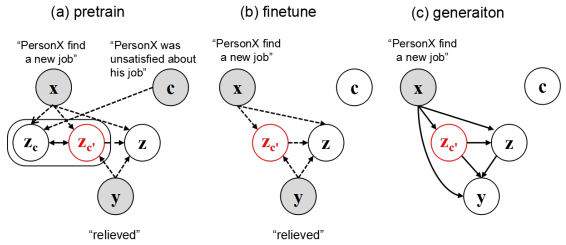


Figure 3: Illustration of pretrain, finetune and generation process of CWVAE in a directed graph. Dashed lines represent the inference of $z$, $z_c$ and $z_{c'}$. Solid lines represent the generation process. Red circle denotes the context-aware latent variable.

ditional variational autoencoder (CVAE) (Sohn et al., 2015) is an extension of VAE on the conditional generation problem. As shown in Figure 2 (a), CVAE characterizes the conditional one-to-many generation problem using three random variables: event $x$, target $y$ and a latent variable $z$, which is used for modeling the latent distribution of semantic over targets given an event. Hence, under a certain inference dimension, with regard to the latent semantic variable $z$, the conditional generation problem could be expressed as $p(y|x) = \int p(y|x,z)p(z|x)dz$. CVAE models $p(y|x,z)$ and $p(z|x)$ using deep neural networks (parameterized by $\theta$) $p_\theta(y|x,z)$ and $p_\theta(z|x)$. Then as illustrated in Figure 2 (b), $y$ could be generated from $x$ and $z$.

CVAE is trained to maximize the conditional likelihood $p(y|x)$, which involves an intractable marginalization over the latent variable $z$. Instead, following Kingma and Welling (2014), a practical way is to introduce another deep network (parameterized by $\phi$) $q_\phi(z|x,y)$ to approximate the true posterior distribution $p(z|x,y)$ and maximize the evidence lower bound (ELBO) of the log-likelihood function:

$$\begin{aligned} L^{ELBO}(\theta,\phi) =& \mathbb{E}_{q_\phi(z|x,y)}\log(p_\theta(y|x,z)) - \\ & \mathrm{KL}(q_\phi(z|x,y)||p_\theta(z|x)) \\ & \leq \log p(y|x) \end{aligned} \quad (1)$$

Therefore, CVAE is composed of three neural networks in general. We refer to $p_\theta(z|x)$ as a prior network, $q_\phi(z|x,y)$ as a recognition network, and $p_\theta(y|x,z)$ as a neural decoder.

## 3 Context-aware Variational Autoencoder

Traditional CVAE can model the `event-target` relation. In other words, given an observed event, CVAE can generate its corresponding targets. While in this paper we model the If-Then reasoning as a `[(background), event]-target` process. It means that in addition to the observed event, we also want to involve the event background knowledge (which can be learned from event contexts) to generate the reasonable targets.

To this end, we propose a context-aware variational autoencoder (CWVAE), with two additional latent variables: a context-acquiring latent variable $z_c$ to directly acquire context information, and a context-aware latent variable $z_{c'}$ to learn background knowledge from $z_c$, as shown in Figure 3 (a). However, the event context information is absent in the Event2Mind and Atomic dataset. To learn from the external event context information, we design the following two-stage training procedure for CWVAE.

**Pretrain: Learning Event Background Knowledge from Auxiliary Dataset** In the pretrain stage, CWVAE is trained on three narrative story corpora with rich event context information. As shown in Figure 3 (a), context-acquiring latent variable $z_c$ is directly conditioned on the context $c$. Hence, $z_c$ could be employed for acquiring background knowledge from event contexts. Then, we minimize the distance between $z_c$ and the context-aware latent variable $z_{c'}$, by which the event background knowledge is transferred from $z_c$ to $z_{c'}$.

**Finetune: Adapt Event Background Knowledge to Each Inference Dimension** In the finetune stage, as shown in Figure 3 (b), CWVAE is trained on the Event2Mind and Atomic dataset without the event context information. Pretrained CWVAE is finetuned to learn the specific inferential knowledge of each inference dimension. After the training procedure, as shown in Figure 3 (c), samples of $z$ is generated based on $x$ and samples of $z_{c'}$, where $z_{c'}$ contains rich event background knowledge helpful for If-Then reasoning.

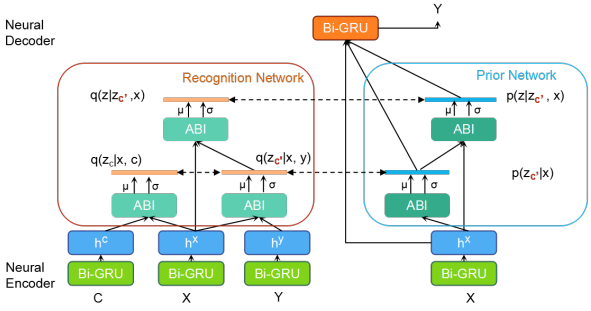


Figure 4: Architecture of CWVAE. We mark Neural encoder in green, prior network in blue, recognition network in brown and neural decoder in orange, respectively.

### 3.1 Architecture of CWVAE

As shown in Figure 4, CWVAE is mainly composed of four parts: a neural encoder that provides distributed representations of base events/targets, a recognition network for inferring $q_\phi(z|x,y)$, $q_\phi(z_c|x,c)$ and $q_\phi(z|z_{c'},x)$, a prior network for modeling $p_\theta(z_{c'}|x)$ and $p_\theta(z|x,z_{c'})$, and a neural decoder that integrates the information from $z$ and $z_{c'}$ to generate targets.

**Neural Encoder** We employ a bidirectional GRU as neural encoder, which encodes context $c$, event $x$ and target $y$ into distributed representations $h^c = \{h^c_1, \ldots, h^c_{l_c}\}$, $h^x = \{h^x_1, \ldots, h^x_{l_x}\}$ and $h^y = \{h^y_1, \ldots, h^y_{l_y}\}$, where $l_c$, $l_x$ and $l_y$ is the length of $c$, $x$ and $y$, respectively.

**Recognition Network** The recognition network models $q_\phi(z|x,y)$, $q_\phi(z_c|x,c)$, $q_\phi(z|z_{c'},x)$ based on $h^x$, $h^y$ and $h^c$.

Following traditional VAE, the above-mentioned three distributions are assumed to be multivariate Gaussian distribution with a diagonal covariance structure:

$$\begin{aligned} q_\phi(z_c|x,c) &\sim N(\mu_{z_c}(x,c), \sigma_{z_c}(x,c)I) \\ q_\phi(z_{c'}|x,y) &\sim N(\mu_{z_{c'}}(x,y), \sigma_{z_{c'}}(x,y)I) \\ q_\phi(z|x,y) &\sim N(\mu_z(x,y), \sigma_z(x,y)I) \end{aligned} \quad (2)$$

where $\mu$ denotes the mean of the distribution, $\sigma$ denotes the standard deviation of the distribution, and $I$ denotes the identity matrix.

Given $h^x$, $h^y$ and $h^c$, we propose a novel attention-based inferer (ABI) module to estimate the mean and standard deviation of $q_\phi(z_c|x,c)$, $q_\phi(z_{c'}|x,y)$ and $q_\phi(z|x,y)$:

$$\begin{aligned} \mu_{z_c}, \sigma_z &= \mathrm{ABI}_c(h^c, h^x) \\ \mu_{z_{c'}}, \sigma_z &= \mathrm{ABI}_{c'}(h^y, h^x) \\ \mu_z, \sigma_z &= \mathrm{ABI}_z(z_{c'}, h^x) \end{aligned} \quad (3)$$

Briefly, through the attention mechanism, ABI can capture the semantic interaction between input sequences, and estimate the parameters of distributions based on it. We will introduce the specific structure of ABI in below.

**Prior Network** Prior Network models $p_\theta(z_{c'}|x)$ and $p_\theta(z|x,z_{c'})$ based on $h^x$. The distribution of $p_\theta(z_{c'}|x)$ and $p_\theta(z|x,z_{c'})$ are still assumed to be multivariate Gaussian, whereas the parameters are different:

$$\begin{aligned} p_\theta(z|x) &\sim N(\mu'_z(x), \sigma'_z(x)I) \\ p_\theta(z_{c'}|x) &\sim N(\mu'_{z_{c'}}(x), \sigma'_{z_{c'}}(x)I) \end{aligned} \quad (4)$$

where $\mu'$ denotes the mean of the distribution, $\sigma'$ denotes the standard deviation of the distribution and $I$ denotes the identity matrix.

Then the attention-based inferer module is still employed to estimate parameters of distributions:

$$\begin{aligned} \mu'_{z_{c'}}, \sigma'_z &= \mathrm{ABI}'_{c'}(h^y, h^x) \\ \mu'_z, \sigma'_z &= \mathrm{ABI}'_z(z_{c'}, h^x) \end{aligned} \quad (5)$$

**Neural Decoder** Given the base event $x$, the semantic latent variable $z$, and the context-aware latent variable $z_{c'}$, the neural decoder defines the generation probability of $y$ as following:

$$p(y|x,z,z_{c'}) = \prod_{j=1}^{n} p(y_j|y<j, z, z_{c'}, x) \quad (6)$$

where $p(y_j|y<j,z,z_{c'},x) = g(y_{j-1}, s_{j-1}, e_j)$, $g(\cdot)$ is an attention-based feed forward model, $e_j = \sum_i \alpha_{ji} h^x_i$ is the context vector and $s_{j-1}$ is the hidden state of the decoder. We obtain $g(\cdot)$ and $e_j$ the same way as Bahdanau et al. (2014). Whereas our decoder differs from Bahdanau et al. (2014) in that our model integrates the context-aware latent variable $z_{c'}$ and semantic latent variable $z$ in the computation of $s_j = \mathrm{GRU}([E_{yj}; s_{j-1}, z, z_{j-1}])$, where $E_{yj}$ is the word embeddings of target words.

Note that through concatenating $z$ and $z_{c'}$ with $E_{yj}$ and $s_{j-1}$, $s_j$ could be affected by context-aware latent variable $z_{c'}$ and semantic latent variable $z$. This allows model to directly access to the event background knowledge from $z_{c'}$. In addition, the randomness of $z$ and $z_{c'}$ would increase the diversity of model generation.

**Attention-based Inferer** Attention mechanism has shown strong ability in capturing semantic interactions (Gong et al., 2017). Inspired by the co-attention mechanism (Parikh et al., 2016), we propose an attention-based inferer (ABI) to estimate

the mean and standard deviation of a distribution belongs to $p_\theta(\cdot)$ or $q_\phi(\cdot)$ by capturing semantic interactions of input sequences.

Specifically, given two input sequences (e.g., representations of contexts and events) $a = \{a_1, \ldots, a_{l_a}\}$ and $b = \{b_1, \ldots, b_{l_b}\}$ with length $l_a$ and $l_b$, we first obtain the attention scores from each side through:

$$\begin{aligned}\gamma_t^a &= \frac{\exp[(W_a h_t^a)^T (W_b h_i^b)]}{\sum_i \exp[(W_a h_t^a)^T (W_b h_i^b)]} \\ \gamma_t^b &= \frac{\exp[(W_a h_i^a)^T (W_b h_t^b)]}{\sum_i \exp[(W_a h_i^a)^T (W_b h_t^b)]}\end{aligned} \tag{7}$$

where $W_a \in \mathbb{R}^{d \times d_a}$ and $W_b \in \mathbb{R}^{d \times d_b}$ are parameter weights.

With these attention scores, the context vectors of both sequences are given by:

$$\begin{aligned}c_t^a &= a\gamma_t^a \\ c_t^b &= b\gamma_t^b\end{aligned} \tag{8}$$

Then we perform a mean pooling operation on context vectors of both sequences:

$$\begin{aligned}\bar{c^a} &= \frac{1}{l_a}\sum_{t=1}^{l_a} c_t^a \\ \bar{c^b} &= \frac{1}{l_b}\sum_{t=1}^{l_b} c_t^b\end{aligned} \tag{9}$$

To obtain the mean and standard deviation, the pooled context vectors $\bar{c^a}$ and $\bar{c^b}$ which carry semantic interaction between two sequences, are concatenated and projected into a latent semantic space through a nonlinear transformation:

$$h_z = \tanh(W[\bar{c}^a; \bar{c}^b] + b_z) \tag{10}$$

Finally the mean and standard deviation are generated through a nonlinear transformation over $h_z$:

$$\begin{aligned}\mu &= W_\mu h_z + b_\mu \\ \sigma &= \text{softplus}(W_\sigma h_z + b_\sigma)\end{aligned} \tag{11}$$

### 3.2 Optimizing

With the incorporation of $z_{c'}$, the original loglikelihood could be decomposed as:

$$\log p(y|x) = \iint p(y|z, z_{c'}, x) dz dz_{c'} \tag{12}$$

Then following traditional CVAE, the ELBO of CWVAE is defined as follows:

$$\begin{aligned}L^{ELBO}(\theta, \phi) = & \\ & \underbrace{\mathbb{E}_{q_\phi(z|x,z_{c'})q_\phi(z_{c'}|x,y)} \log p_\theta(y|x, z, z_{c'})}_{\text{Reconstruction Loss}} \\ & - \underbrace{\int q_\phi(z|x,y)\text{KL}(q_\phi(z_{c'}|x,y)||p_\theta(z_{c'}|x))dz}_{\text{KL term}} \\ & - \underbrace{\int q_\phi(z_{c'}|x,y)\text{KL}(q_\phi(z|x,z_{c'})||p_\theta(z|x,z_{c'}))dz_{c'}}_{\text{KL term}}\end{aligned} \tag{13}$$

which is the objective function at the finetune stage.

While in the pretrain stage, as we aim to learn background knowledge through minimizing the distance between $z_c$ and $z_{c'}$, in addition to $L^{ELBO}$, a context-aware regulation term is introduced:

$$\begin{aligned}L^{PT}(\theta, \phi) = & L^{ELBO}(\theta, \phi) \\ & \underbrace{- \lambda \int \text{KL}(q_\phi(z_c|x,c)||q_\phi(z_{c'}|x,y))dz_{c'}}_{\text{context-aware regulation}}\end{aligned} \tag{14}$$

where the context aware regularization term is the KL distance between $z$ and $z_{c'}$. Through minimizing the context aware regularization term, we aim to pass event context knowledge from $z_c$ to the context aware latent variable $z_{c'}$.

### 3.3 Training Details

To test the performance of CWVAE, we split the Event2Mind and Atomic dataset into training, development and test sets (80%, 10%, 10%) in the same way as Rashkin et al. (2018) and Sap et al. (2018), respectively.

We initialize the embedding layer from 300d GloVe word embeddings. The neural encoder is chosen to be biGRU with 300 hidden units. For the ABI module, size of $W_a$ and $W_b$ is set to be $100 \times d_a$ and $100 \times d_b$ respectively. The dimension of $z_c$, $z_{c'}$ and $z$ is all set as 40. The neural decoder is set to be GRU with 300d hidden state. Regulation coefficient $\lambda$ of context-aware regulation term is set to be 0.1. Models are trained using an Adam optimizer (Kinga and Adam, 2015) with a learning rate of 0.001.

## 4 Experiments

### 4.1 Auxiliary Dataset

The auxiliary dataset is built upon three human-written story corpora: ROCStories (Mostafazadeh

| Context | Event | Inference Target |
|---|---|---|
| ① jason had been really stressed out at work.<br>② he decided he needed a different kind of job.<br>③ jason applied for a job in a different field. | ④ he got the job . | ⑤ jason was much happier at his new job . |

Table 3: An example for the construction of auxiliary dataset. For a five-sentence-paragraph, the first three sentences are taken as event context, while the fourth and fifth sentence is taken as base event and target respectively.

et al., 2016), VIST (Huang et al., 2016) and WritingPrompts (Fan et al., 2018). ROCStories and VIST are composed of short stories with five sentences. We filter out stories of more than 1,000 words in WritingPrompts, and cut the remaining stories into five-sentence-paragraphs.

For each five-sentence-paragraph, we define the first three sentences as contexts of the base event, the fourth sentence as the base event, and the fifth sentence as the inference target. For example, as shown in Table 3, the first three sentences describe a context that Jason was unsatisfied about his job and applied for a new job. Hence, after happening the event "he got the job", a plausible react about the event could be "jason was much happier at his new job". In total, the auxiliary dataset contains 192,316 $(context, event, target)$ triples.

### 4.2 Baselines

We compared our proposed model with the following four baseline methods:

- **RNN-based Seq2Seq** proposed by Sap et al. (2018) for the If-Then reasoning on Atomic.
- **Variational Seq2Seq** combines a latent variable with the encoder-decoder structure through converting the last hidden state of RNN encoder into a Gaussian distributed latent variable (Bowman et al., 2015).
- **VRNMT** Propose by Su et al. (2018), VRNMT combines CVAE with attention-based encoder-decoder framework through introduces a latent variable to model the semantic distribution of targets.
- **CWVAE-Unpretrained** refers to the CWVAE model without the pretrain stage.

Note that, for each baseline method, we train distinct models for each distinct inference dimension, respectively.

### 4.3 Evaluation Metrics

**Automatic Evaluation** We first compare the perplexity of CWVAE with baseline methods. Perplexity measures the probability of model to regenerate the exact targets, which is particular suitable for evaluating the model performance on one-to-many problem (Serban et al., 2017). Further, we employ BLEU score to evaluate the accuracy of generations (Papineni et al., 2002), and the number of distinct n-gram to evaluate the diversity of generations (Li et al., 2016). The distinct is normalized to $[0, 1]$ by dividing the total number of generated tokens.

| Metric | Methods | xIntent | xReact | oReact |
|---|---|---|---|---|
| PPL | RNN-based Seq2Seq | 44.12 | 29.18 | 14.08 |
| | Variational Seq2Seq | 42.06 | 28.22 | 12.62 |
| | VRNMT | 33.45 | 25.54 | 11.93 |
| | CWVAE-Unpretrained | 31.32 | 24.07 | 11.37 |
| | CWVAE | **29.23** | **23.17** | **11.04** |
| BLEU | RNN-based Seq2Seq | 2.75 | 2.11 | 5.18 |
| | Variational Seq2Seq | 2.84 | 2.43 | 2.08 |
| | VRNMT | 3.94 | 4.81 | 6.61 |
| | CWVAE-Unpretrained | 5.52 | 7.36 | 5.33 |
| | CWVAE | **5.65** | **12.98** | **6.97** |

Table 4: Average perplexity and BLEU score (reported in percentages) for the top 10 generations under each inference dimension of Event2Mind. The the best result for each dimension is emboldened.

| Metric | Methods | xIntent | xReact | oReact |
|---|---|---|---|---|
| dist-1 | RNN-based Seq2Seq | 0.0002 | 0.0002 | 0.0001 |
| | Variational Seq2Seq | 0.0006 | 0.0003 | 0.0001 |
| | VRNMT | 0.0002 | 0.0002 | 0.0003 |
| | CWVAE-Unpretrained | 0.0023 | 0.0017 | 0.0004 |
| | CWVAE | **0.0052** | **0.0033** | **0.0025** |
| dist-2 | RNN-based Seq2Seq | 0.0005 | 0.0002 | 0.0002 |
| | Variational Seq2Seq | 0.0014 | 0.0002 | 0.0001 |
| | VRNMT | 0.0005 | 0.0003 | 0.0001 |
| | CWVAE-Unpretrained | 0.0061 | 0.0040 | 0.0013 |
| | CWVAE | **0.0146** | **0.0099** | **0.0063** |

Table 5: Distinct-1 and distinct-2 scores for the top 10 generations under each inference dimension of Event2Mind. The the best result for each dimension is emboldened.

**Human Evaluation** Since automatic evaluation of generations is still a challenging task (Liu et al., 2016), we also conduct human evaluations on the model performance. Five human experts are employed to evaluate the *coherence*, *diversity* and *fluency* of generated targets. Experts are asked to vote for if a generation is fluent or coherent for each generated target, and give a 1-5 score for the diversity of generations. For both Event2Mind and Atomic datasets, 100 events are randomly selected from the test set. For each method, top 10 generated targets of each base event are used for eval-

| Metric | Methods | xIntent | xNeed | xAttr | xEffect | xReact | xWant | oWant | oReact | oEffect |
|---|---|---|---|---|---|---|---|---|---|---|
| PPL | RNN-based Seq2Seq | 22.54 | 24.69 | 33.54 | 65.13 | 29.52 | 26.63 | 16.76 | 14.99 | 35.17 |
| | Variational Seq2Seq | 26.48 | 28.31 | 33.00 | 68.62 | 29.93 | 29.50 | 16.98 | 14.25 | 34.20 |
| | VRNMT | 21.04 | 24.28 | 24.87 | 61.05 | 26.62 | 28.57 | 14.45 | 14.86 | 30.12 |
| | CWVAE-Unpretrained | 20.73 | 23.72 | 25.80 | 60.62 | 25.75 | 26.71 | 15.93 | 12.82 | 32.00 |
| | CWVAE | **15.93** | **20.32** | **23.85** | **50.74** | **21.39** | **24.02** | **14.02** | **11.70** | **29.13** |
| BLEU | RNN-based Seq2Seq | 8.17 | 12.35 | 2.96 | 5.26 | 3.43 | 13.44 | 7.08 | 4.09 | 6.42 |
| | Variational Seq2Seq | 8.31 | 12.05 | 2.13 | 6.07 | 2.52 | 11.71 | 7.40 | 4.08 | 6.38 |
| | VRNMT | 9.52 | 13.35 | 4.87 | 4.42 | 7.64 | 9.80 | 10.79 | 5.28 | 13.71 |
| | CWVAE-Unpretrained | 11.37 | 14.64 | 4.07 | 14.11 | 7.86 | 12.70 | 12.09 | 8.16 | **14.93** |
| | CWVAE | **12.12** | **15.67** | **5.63** | **14.64** | **8.13** | **15.01** | **13.83** | **8.58** | 11.63 |

Table 6: Average perplexity and BLEU scores (reported in percentages) for the top 10 generations under each inference dimension of Atomic. The the best result for each dimension is emboldened.

| Metric | Methods | xIntent | xNeed | xAttr | xEffect | xReact | xWant | oWant | oReact | oEffect |
|---|---|---|---|---|---|---|---|---|---|---|
| dist-1 | RNN-based Seq2Seq | 0.0012 | 0.0029 | 0.0004 | 0.0019 | 0.0001 | 0.0022 | 0.0006 | 0.0001 | 0.0006 |
| | Variational Seq2Seq | 0.0006 | 0.0018 | 0.0002 | 0.0002 | 0.0001 | 0.0013 | 0.0007 | 0.0001 | 0.0002 |
| | VRNMT | 0.0002 | 0.0001 | 0.0053 | 0.0005 | 0.0018 | 0.0022 | 0.0005 | 0.0001 | 0.0004 |
| | CWVAE-Unpretrained | 0.0019 | 0.0036 | 0.0119 | **0.0046** | 0.0021 | 0.0013 | 0.0018 | 0.0005 | 0.0006 |
| | CWVAE | **0.0055** | **0.0045** | **0.0142** | 0.0028 | **0.0043** | **0.0040** | **0.0021** | **0.0030** | **0.0033** |
| dist-2 | RNN-based Seq2Seq | 0.0036 | 0.0081 | 0.0002 | 0.0018 | 0.0002 | 0.0006 | 0.0013 | 0.0001 | 0.0011 |
| | Variational Seq2Seq | 0.0013 | 0.0042 | 0.0001 | 0.0003 | 0.0002 | 0.0026 | 0.0002 | 0.0003 | 0.0006 |
| | VRNMT | 0.0002 | 0.0011 | 0.0002 | 0.0005 | 0.0001 | 0.0034 | 0.0005 | 0.0001 | 0.0004 |
| | CWVAE-Unpretrained | 0.0060 | 0.0088 | 0.0136 | **0.0113** | 0.0043 | 0.0029 | 0.0041 | 0.0011 | 0.0009 |
| | CWVAE | **0.0162** | **0.0112** | **0.0146** | 0.0072 | **0.0013** | **0.0107** | **0.0044** | **0.0068** | **0.0093** |

Table 7: Distinct-1 and distinct-2 scores for the top 10 generations under each inference dimension of Atomic. The the best result for each dimension is emboldened.

uation. Finally we report three overall averaged scores of *coherence*, *diversity* and *fluency* on both datasets, respectively.

### 4.4 Overall Results

We list the perplexity and BLEU score of CWVAE and baseline methods on Event2Mind and Atomic in Table 4 and Table 6, respectively, and show the distinct-1 and distinct-2 score on Event2Mind and Atomic in Table 5 and Table 7, respectively. We find that:

(1) As shown in Table 5 and Table 7, comparison between RNN-based Seq2Seq and variational-based methods, including Variational Seq2Seq, VRNMT, CWVAE-unpretrained and CWVAE shows that, variational-based methods could increase the diversity of generations. This confirms one of our motivations that variational-based methods could capture the latent semantic distribution within targets and increase the diversity of If-Then reasoning.

(2) Comparing CWVAE-unpretrained with other baseline methods shows that, in general CWVAE improves the accuracy and diversity on both dataset. These results indicate the efficiency of CWVAE in capturing the latent semantic distribution of targets, and generate more reasonable inferential results.

(3) Comparison between CWVAE and CWVAE-unpretrained shows that the pre-train stage could enhance the performance of CWVAE in both the accuracy and diversity. This is mainly because event knowledge could offer the guidance for If-Then reasoning. In the pretrain stage, CWVAE could capture the event background knowledge through context-aware latent variable, and such knowledge could be be adapted to our task through the fintune stage.

| Methods | Coherence | Diversity | Fluency |
|---|---|---|---|
| RNN-based Seq2Seq | 0.28 | 2.03 | 0.73 |
| Variational Seq2Seq | 0.33 | 1.67 | 0.92 |
| VRNMT | 0.32 | 2.60 | 0.83 |
| CWVAE-Unpretrained | 0.36 | 2.10 | 0.92 |
| CWVAE | **0.43** | **2.85** | **0.96** |

Table 8: Human evaluation results on Event2Mind.

| Methods | Coherence | Diversity | Fluency |
|---|---|---|---|
| RNN-based Seq2Seq | 0.21 | 2.66 | 0.78 |
| Variational Seq2Seq | 0.22 | 2.70 | 0.90 |
| VRNMT | 0.24 | 2.61 | 0.78 |
| CWVAE-Unpretrained | 0.25 | 2.72 | 0.83 |
| CWVAE | **0.32** | **3.03** | **0.90** |

Table 9: Human evaluation results on Atomic.

To further evaluate the effectiveness of our proposed approach, we also conduct human evaluations, the results of which are shown in Table 8 and Table 9. On both datasets, CWVAE-based methods achieve consistent better coherence, diversity and fluency performances. While comparing with CWVAE-Unpretrained, the pretrain procedure could improves the performance

| Base event | Inference dim. | Generations | | Ground truth |
|---|---|---|---|---|
| | | CWVAE | RNN-based Seq2Seq | |
| PersonX works tirelessly | xIntent | be productive and hardworking<br>finish his work soon<br>earn more money and accomplish goal<br>finish his work and make money<br>finish his work and accomplish goal<br>be productive and successful | get the job done<br>get the job done on time<br>get the job done in the way<br>get the job done for his life<br>make money for his own future<br>make money for his own money | be productive<br>finish the project as soon as possible<br>reach goal |

Table 10: An example of inferences made by CWVAE and RNN-based Seq2Seq model under inference dimension "xIntent".

on coherence and fluency. The main reasons are twofold: first, the CWVAE has advantage in capturing the semantic distribution of targets; second, event background learned from the pretrain stage is helpful for the If-Then reasoning.

### 4.5 Case Study

Table 10 provides an example of model generations given the base event "PersonX works tirelessly" and the inference dimension "xIntent". The generations under CWVAE mainly contain four kinds of semantics: (1) be productive, (2) finish his work soon, (3) accomplish goal, (4) earn more money. While the semantics of generations using baseline RNN-based Seq2Seq model is relatively limited. Furthermore, the first three kinds of semantic overlap the three ground truth targets, and the fourth kind of semantic is in accordance with daily-life commonsense. Compared to RNN-based Seq2Seq model, our approach can increase the diversity and rationality of generations, meanwhile keep the accuracy.

## 5 Related Work

### 5.1 Event-Centered Commonsense Reasoning

Understanding events and constructing event-centered commonsense knowledge are crucial to many NLP applications, such as intention recognition (Goldwasser and Zhang, 2016) and dialog generation (Wen et al., 2017). Recently a growing number of studies focus on event-centered commonsense reasoning, which mainly concentrates on two areas, script event prediction and story ending generation/choosing.

Script event prediction concerns with the temporal relationships between script events (Granroth-Wilding and Clark, 2016), which requires models to choose a correct subsequent triple-organized event among the candidates (Wang et al., 2017). Prior work mainly focused on modeling event pairs (Granroth-Wilding and Clark, 2016), event chains (Wang et al., 2017) and event graph (Li et al., 2018) to predict the subsequent event. Story ending generation focuses on generating plausible story endings (Mostafazadeh et al., 2016), which requires models to understand the story context, and keep generated endings logically consistent with it (Peng et al., 2017; Guan et al., 2019). The above tasks mainly investigate the logical orders of events, whereas the If-Then reasoning task focuses on inferring the mental state of event participants.

### 5.2 Variational AutoEncoder-Decoder Based Natural Language Generation

VAE (Kingma and Welling, 2014) has been widely applied in various of text generation tasks, such as dialogue and machine translation.

In dialogue generation, Zhao et al. (2017) adapts VAE with encoder-decoder framework to model the latent semantic distribution of answers, which can increase the diversity of generations. For the task of machine translation, Su et al. (2018) and Zhang et al. (2016) employ a latent variable to capture the semantic interaction between the source and target sentence, and regard the latent variable as a supplementation of attention mechanism. While Wang et al. (2019) use the latent variable to model topic distributions in text generation. In this paper, we introduce an additional context-aware latent variable to effectively learn background knowledge and conduct If-Then reasoning on the guidance of it.

## 6 Conclusion

In this paper, we propose a novel context-aware VAE (CWVAE) framework with two training stages for If-Then commonsense reasoning. By introducing an additional context-aware latent variable, CWVAE is able to learn external background knowledge, and conduct If-Then reasoning under its guidance. In the pretrain stage, CWVAE learns event background knowledge, then in the finetune stage CWVAE adapts such knowledge to each inference dimension. Experimental results demonstrate that CWVAE outperforms baseline methods in both the accuracy and diversity of generations.

## 7 Acknowledgments

We thank the anonymous reviewers for their constructive comments, and gratefully acknowledge the support of the National Key Research and Development Program of China (SQ2018AAA010010), the National Key Research and Development Program of China (2018YFB1005103), the National Natural Science Foundation of China (NSFC) via Grant 61702137.

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*.

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. *Proc. of ACL-08: HLT*, pages 789–797.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898.

Dan Goldwasser and Xiao Zhang. 2016. Understanding satirical articles using common-sense. *Transactions of the Association for Computational Linguistics*, 4:537–549.

Yichen Gong, Heng Luo, and Jian Zhang. 2017. Natural language inference over interaction space. *arXiv preprint arXiv:1709.04348*.

Mark Granroth-Wilding and Stephen Clark. 2016. What happens next? event prediction using a compositional neural network model. In *AAAI*.

Jian Guan, Yansen Wang, and Minlie Huang. 2019. Story ending generation with incremental encoding and commonsense knowledge.

Ting-Hao Kenneth Huang, Francis Ferraro, Nasrin Mostafazadeh, Ishan Misra, Aishwarya Agrawal, Jacob Devlin, Ross Girshick, Xiaodong He, Pushmeet Kohli, Dhruv Batra, et al. 2016. Visual storytelling. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1233–1239.

D Kinga and J Ba Adam. 2015. A method for stochastic optimization. In *ICLR*, volume 5.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. *stat*, 1050:10.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of NAACL-HLT*, pages 110–119.

Zhongyang Li, Xiao Ding, and Ting Liu. 2018. Constructing narrative event evolutionary graph for script event prediction. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 4201–4207. AAAI Press.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2122–2132.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the NAACL: Human Language Technologies*, pages 839–849.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proc. of the 40th ACL*, pages 311–318.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255.

Haoruo Peng, Snigdha Chaturvedi, and Dan Roth. 2017. A joint model for semantic sequences: Frames, entities, sentiments. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 173–183.

Hannah Rashkin, Maarten Sap, Emily Allaway, Noah A Smith, and Yejin Choi. 2018. Event2mind: Commonsense inference on events, intents, and reactions. In *Proc. of the 56th ACL*, volume 1, pages 463–473.

Maarten Sap, Ronan LeBras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A Smith, and Yejin Choi. 2018. Atomic: An atlas of machine commonsense for if-then reasoning. *arXiv preprint arXiv:1811.00146*.

Roxane Segers, Marco Rospocher, Piek Vossen, Egoitz Laparra, German Rigau, and Anne-Lyse Minard. 2016. The event and implied situation ontology (eso): Application and evaluation. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. 2015. Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems*, pages 3483–3491.

Jinsong Su, Shan Wu, Deyi Xiong, Yaojie Lu, Xianpei Han, and Biao Zhang. 2018. Variational recurrent neural machine translation. In *AAAI*.

Wenlin Wang, Zhe Gan, Hongteng Xu, Ruiyi Zhang, and Lawrence Carin. 2019. Topic-guided variational autoencoders for text generation.

Zhongqing Wang, Yue Zhang, and Ching-Yun Chang. 2017. Integrating order information and event relation for script event prediction. In *Proc. of the 2017 EMNLP*, pages 57–67.

Tsung-Hsien Wen, Yishu Miao, Phil Blunsom, and Steve Young. 2017. Latent intention dialogue models. In *Proc. of the 34th ICML-Volume 70*, pages 3732–3741. JMLR. org.

Biao Zhang, Deyi Xiong, Hong Duan, Min Zhang, et al. 2016. Variational neural machine translation. In *Proc. of the 2016 EMNLP*, pages 521–530.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 654–664.

# Asynchronous Deep Interaction Network for Natural Language Inference

**Di Liang***, **Fubao Zhang***, **Qi Zhang**†, **and Xuanjing Huang**
School of Computer Science, Shanghai Key Laboratory of Intelligent Information Processing
Fudan University, Shanghai, P.R.China 201203
{liangd17, fbzhang17, qz, xjhuang}@fudan.edu.cn

## Abstract

Natural language inference aims to predict whether a premise sentence can infer another hypothesis sentence. Existing methods typically have framed the reasoning problem as a semantic matching task. The both sentences are encoded and interacted symmetrically and in parallel. However, in the process of reasoning, the role of the two sentences is obviously different, and the sentence pairs for NLI are asymmetrical corpora. In this paper, we propose an **asynchronous deep interaction network (ADIN)** to complete the task. ADIN is a neural network structure stacked with multiple inference sub-layers, and each sub-layer consists of two local inference modules in an asymmetrical manner. Different from previous methods, this model deconstructs the reasoning process and implements the asynchronous and multi-step reasoning. Experiment results show that ADIN achieves competitive performance and outperforms strong baselines on three popular benchmarks: SNLI, MultiNLI, and SciTail.

## 1 Introduction

Natural language inference (NLI) is a pivotal and fundamental task in natural language understanding and artificial intelligence. The goal of NLI is to predict whether a premise sentence can infer another hypothesis sentence. As illustrated in Table 1, logical relationships between the two sentences include *entailment* (if the premise is true, then the hypothesis must be true), *contradiction* (if the premise is true, then the hypothesis must be false), and *neutral* (neither entailment nor contradiction).

As a core task, conventional approaches have studied various aspects of the inference problem (MacCartney and Manning, 2008; Heilman and Smith, 2010). Thanks to the release of the largest publicly available corpus - the Stanford Natural Language Inference (SNLI) corpus (Bowman et al., 2015), neural network-based models have also been successfully used for this task (Parikh et al., 2016; Chen et al., 2016; Tay et al., 2018; Duan et al., 2018). These methods typically treat the premise sentence and the hypothesis sentence equally, learn an alignment of sub-phrases in both sentences symmetrically and in parallel, and fuse local information for making a global decision at the sentence level. They all frame the inference problem as a semantic matching task and ignore the reasoning process.

| |
|---|
| **Premise:** A dog is jumping for a Frisbee in the snow.<br>**Hypothesis:** An animal is playing with a plastic toy.<br>**Label:** Entailment |
| **Premise:** He was crying like his mother had just walloped him.<br>**Hypothesis:** He was crying like his mother hit him with a spoon.<br>**Label:** Neutral |
| **Premise:** Several men in front of a white building.<br>**Hypothesis:**Several people in front of a gray building.<br>**Label:** contradiction |

Table 1: Examples of natural language inference.

However, different from a simple semantic matching task, reasoning should be asynchronous and fully interpretable (Yi et al., 2018). Moreover, the sentence pairs for NLI are asymmetrical corpora, i.e., $I(a, b) \neq I(b, a)$. Considering the first

*Equal contribution. Alphabetical order of the last name.
†Corresponding author.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2692–2700, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

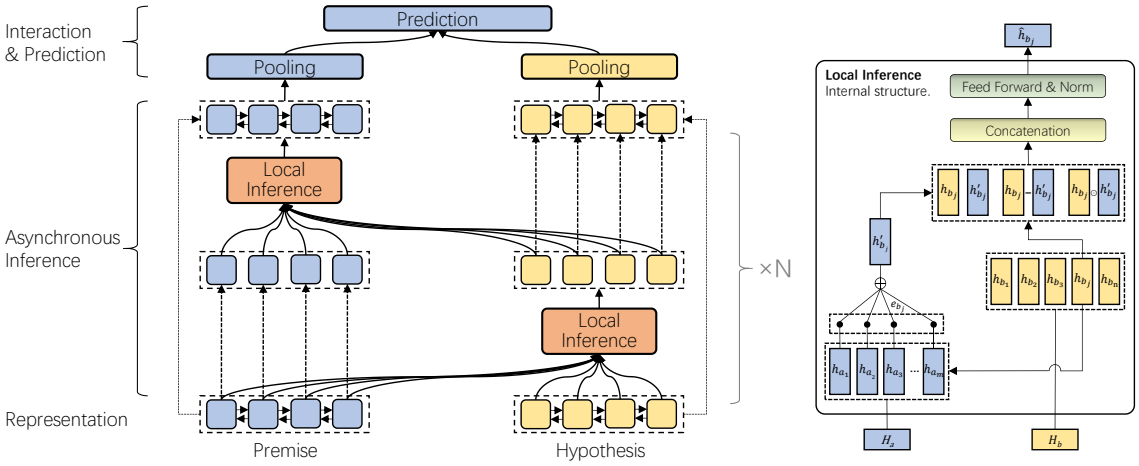


Figure 1: The overall view of our model. The left part is the main framework of this work. The dashed lines refer to the copy operation. The asynchronous inference layer is stacked with $N$ inference sub-layers. In the first sub-layer of the asynchronous inference layer, the input is from the representation layer. Subsequently, the input of sub-layers come from the previous sub-layers. The right part is the detailed structure of the local inference module, taking $\hat{\mathbf{h}}_{b_j}$ as an example.

example in Table 1, the premise sentence can infer the hypothesis sentence, however, the hypothesis sentence can't infer the premise sentence. The inference process intuitively needs to consider the relationship between two sentences in sequential order. According to the actual inference process, we argue that the model should first get the inferential information to model the hypothesis sentence, based on the premise sentence, and then model premise sentence, based on the new representation for hypothesis sentence.

In this paper, we propose an asynchronous deep interaction network (ADIN) to achieve the reasoning. This model is stacked with multiple inference sub-layers to implement the multi-step reasoning, and each sub-layer consists of two local inference modules in an asymmetrical manner to simulate the asynchronous and interpretable reasoning process. In a local inference module, we update the sentence representation by using the local inference information, based on the attention of the other sentence. Lastly, we combine the inference information between the two sentences to make a global decision.

To demonstrate the effectiveness of our model, we evaluate it on three popular benchmarks: SNLI, MultiNLI, and SciTail. The experimental results on these three data sets reveal that our method achieves competitive performance.

The main contributions of this work can be summarized as follows:

- We break the matching architecture that interacts with the information between two sentences for alignment, and propose an asynchronous deep interaction network to achieve the asynchronous and multi-step reasoning.
- We deconstruct the reasoning process between the two sentences, and the process can be analyzed step-by-step.
- The experimental results on three highly competitive benchmark datasets demonstrate that our model can achieve better performance than other strong baselines.

## 2 Approach

We define the natural language inference as a classification task that predicts the relation $y \in Y$ for a given pair of sentences, where $Y = \{entailment, contradiction, neutral\}$. In this work, we propose an asynchronous deep interaction network (ADIN) to complete this task. The overall architecture of the model is illustrated on the left part of Figure 1.

Our sentence inference architecture, ADIN , is composed of the following three components: (1) *information representation layer* converts the two sentences into semantic representations; (2) *asynchronous inference layer* produces new representations for the two sentences, based on the inference information; and (3) *interaction and prediction layer* determines the overall inference relationship between a premise and hypothesis.

### 2.1 Local Inference Module

Given two natural sentences $a$ and $b$, $\mathbf{H}_a = \{\mathbf{h}_{a_i}|\mathbf{h}_{a_i} \in \mathbb{R}^k, i = 1, 2, ..., m\}$ and $\mathbf{H}_b = \{\mathbf{h}_{b_j}|\mathbf{h}_{b_j} \in \mathbb{R}^k, j = 1, 2, ..., n\}$ denote their $k$-dimensional representations respectively , where $m$, $n$ denote the length of two sentences. Here, we implement a general reasoning process where the module captures the relevance between the two sentences, then incorporates the inferential information to the new representation for sentence $b$, based on the sentence $a$. First, we compute a co-attention matrix $E \in \mathbb{R}^{m \times n}$ to capture the relevance between the two sentences, each element $\mathbf{E}_{i,j} \in R$ indicates the relevance between the i-th word of sentence $a$ and the j-th word of $b$. Formally, the co-attention matrix could be computed as:

$$\mathbf{E}_{i,j} = \mathbf{P}^T tanh(\mathbf{W}(\mathbf{h}_{a_i} \odot \mathbf{h}_{b_j})), \quad (1)$$

where $\mathbf{W} \in \mathbb{R}^{s \times k}$, $\mathbf{P} \in \mathbb{R}^s$, and $\odot$ denotes the element-wise production operation. Then, we get $a$-guided attentive vectors for sentence $b$:

$$\mathbf{e}_{b_j} = softmax(\mathbf{E}_{:,j}), \quad (2)$$

$$\mathbf{h}'_{b_j} = \mathbf{H}_a \cdot \mathbf{e}_{b_j}, \quad (3)$$

In order to enhance the interaction further, we combine the original vector and $a$-guided attentive vector for sentence $b$. More formally:

$$\mathbf{h}''_{b_j} = [\mathbf{h}_{b_j}; \mathbf{h}'_{b_j}; \mathbf{h}_{b_j} - \mathbf{h}'_{b_j}; \mathbf{h}_{b_j} \odot \mathbf{h}'_{b_j}], \quad (4)$$

$$\tilde{\mathbf{h}}_{b_j} = ReLU(\mathbf{W}_{b_j}\mathbf{h}''_{b_j} + \mathbf{b}_{b_j}), \quad (5)$$

where $[\cdot;\cdot;\cdot;\cdot]$ refers to the concatenation operation. In Equation 4, we first calculate the difference and the element-wise product for $(\mathbf{h}_{b_j}, \mathbf{h}'_{b_j})$. We get the new representation containing $a$-guided inferential information for sentence $b$:

$$\tilde{\mathbf{H}}_b = (\tilde{\mathbf{h}}_{b_1}, \tilde{\mathbf{h}}_{b_2}, ..., \tilde{\mathbf{h}}_{b_n}), \quad (6)$$

$$\hat{\mathbf{H}}_b = LayerNorm(\tilde{\mathbf{H}}_b), \quad (7)$$

Where $LayerNorm(.)$ is layer normalization (Ba et al., 2016). The result $\hat{\mathbf{H}}_b$ is a 2D-tensor that has the same shape as $\mathbf{H}_b$, and we refer to the whole inferential module as:

$$\textbf{InferentialModule}(\mathbf{H}_a, \mathbf{H}_b), \quad (8)$$

As described, the inferential module can capture the relevance between the two sentences, incorporate the inferential information to the new representation for sentence $b$, based on the sentence $a$.

### 2.2 Information Representation Layer

The information representation layer converts each word or phrase in the sentences into a vector representation and constructs the representation matrix for the sentences. We combine the multi-level features as the sentence representation. Each token is represented as a vector by using the pre-trained word embedding such as GloVe (Pennington et al., 2014), word2Vec (Mikolov et al., 2013), and fasttext (Joulin et al., 2016). It can also incorporate more syntactical and lexical information into the feature vector.

For ADIN, we use a concatenation of word embedding, character embedding, and syntactical features as the sentence representation. The word embedding is obtained by mapping token to high dimensional vector space by pre-trained word vector (300D Glove 840B), and the word embedding is updated during training. Character-level embedding could alleviate out-of-vocabulary (OOV) problems and capture helpful morphological information. As in (Kim et al., 2016; Lee et al., 2016), we filter the character embedding with 1D convolution kernel. The character convolutional feature maps are then max pooled over the time dimension for each token to obtain a vector.

As in (Chen et al., 2018), the syntactical features consist of one-hot part-of-speech (POS) tagging feature and binary exact match (EM) feature. For one sentence, the EM value is activated if the same word is found in the other sentence.

Next, ADIN adopts bidirectional Long Short-Term Memory network (Bi-LSTM) (Graves and Schmidhuber, 2005) to model the internal temporal interaction on both directions of the sentences. Consider a premise sentence $p$ and a hypothesis sentence $q$, we have got their multi-level features representation. Suppose the length of $p$ and $q$ are $m$ and $n$ respectively. These multi-level features representation are then passed to a Bi-LSTM encoder to obtain the context-dependent hidden state matrix, i.e, $\mathbf{H}_p = \{\mathbf{h}_{p_i}|\mathbf{h}_{p_i} \in \mathbb{R}^d, i = 1, 2, ..., m\}$, and $\mathbf{H}_q = \{\mathbf{h}_{q_j}|\mathbf{h}_{q_j} \in \mathbb{R}^d, j = 1, 2, ..., n\}$, where $d$ is the dimension of Bi-LSTM's hidden state.

### 2.3 Asynchronous Inference Layer

Recently, along with the development of deep learning methods, some neural attention-based models have also been successfully used for NLI (Rocktäschel et al., 2015; Parikh et al., 2016; Duan et al., 2018). However, these methods typically

frame the inference problem as a semantic matching task and ignore the reasoning process, where the premise sentence and the hypothesis sentence are encoded and interacted symmetrically and in parallel.

In this paper, we utilize the local inference module to deconstruct the reasoning process and achieve the asynchronous and multi-step reasoning for NLI. To model the multi-step reasoning habit, this model is stacked with $N$ inference sub-layers to capture step-by-step the logic relationship between the two sentences. In the each inference sub-layer , two inferential modules perform two asynchronous inference processes respectively.

Concretely, in the $t$-th inference sub-layer, given the representations of two sentences computed in the previous sub-layer : $\mathbf{V}_p^{t-1} = (\mathbf{v}_{p_1}^{t-1}, \mathbf{v}_{p_2}^{t-1}, ..., \mathbf{v}_{p_m}^{t-1})$ and $\mathbf{V}_q^{t-1} = (\mathbf{v}_{q_1}^{t-1}, \mathbf{v}_{q_2}^{t-1}, ..., \mathbf{v}_{q_n}^{t-1})$, we get the deeper-level representations:

$$\hat{\mathbf{V}}_q^t = \textbf{InferentialModule}(\mathbf{V}_p^{t-1}, \mathbf{V}_q^{t-1}), \quad (9)$$

$$\hat{\mathbf{V}}_p^t = \textbf{InferentialModule}(\mathbf{V}_q^t, \mathbf{V}_p^{t-1}), \quad (10)$$

$$\tilde{\mathbf{v}}_{q_i}^t = [\hat{\mathbf{v}}_{q_i}^t; \mathbf{v}_{q_i}^{t-1}], \quad \tilde{\mathbf{v}}_{p_j}^t = [\hat{\mathbf{v}}_{p_j}^t; \mathbf{v}_{p_j}^{t-1}], \quad (11)$$

$$\mathbf{v}_{q_i}^t = \text{Bi-LSTM}(\tilde{\mathbf{v}}_{q_{i-1}}^t, \tilde{\mathbf{v}}_{q_i}^t, \tilde{\mathbf{v}}_{q_{i+1}}^t), \quad (12)$$

$$\mathbf{v}_{p_j}^t = \text{Bi-LSTM}(\tilde{\mathbf{v}}_{p_{j-1}}^t, \tilde{\mathbf{v}}_{p_j}^t, \tilde{\mathbf{v}}_{p_{j+1}}^t), \quad (13)$$

where $\mathbf{V}_p^0 = \mathbf{H}_p$, $\mathbf{V}_q^0 = \mathbf{H}_q$, $\hat{\mathbf{V}}_p^t = (\hat{\mathbf{v}}_{p_1}^t, \hat{\mathbf{v}}_{p_2}^t, ..., \hat{\mathbf{v}}_{p_m}^t)$, and $\hat{\mathbf{V}}_q^t = (\hat{\mathbf{v}}_{q_1}^t, \hat{\mathbf{v}}_{q_2}^t, ..., \hat{\mathbf{v}}_{q_n}^t)$. In an inference sub-layer, we first get the inferential information to update the representation for hypothesis sentence, based on the premise sentence. Next, the model incorporates the inferential information to the premise sentence, based on the new representation for hypothesis sentence.

### 2.4 Interaction and Prediction Layer

To extract a proper representation for each sentence, we apply a mean pooling and a max pooling on each of them. Formally:

$$\mathbf{V}_p^{mean} = \sum_{i=1}^{m} \frac{\mathbf{v}_{p_i}^N}{m}, \quad \mathbf{V}_p^{max} = \max_{i=1}^{m} \mathbf{v}_{p_i}^N, \quad (14)$$

$$\mathbf{V}_q^{mean} = \sum_{j=1}^{n} \frac{\mathbf{v}_{q_j}^N}{n}, \quad \mathbf{V}_q^{max} = \max_{j=1}^{n} \mathbf{v}_{q_j}^N, \quad (15)$$

$$\mathbf{V}_p^{new} = [\mathbf{V}_p^{mean}; \mathbf{V}_p^{max}], \quad \mathbf{V}_q^{new} = [\mathbf{V}_q^{mean}; \mathbf{V}_q^{max}], \quad (16)$$

| | Train | Dev | Test | L(P) | L(H) | Vocab |
|---|---|---|---|---|---|---|
| SNLI | 549K | 9.8K | 9.8K | 14 | 8 | 36K |
| MultiNLI[1] | 392K | 9.8K | 9.8K | 22 | 11 | 85K |
| MultiNLI[2] | | 9.8K | 9.8K | 22 | 11 | 85K |
| SciTail | 23.6K | 1.3K | 2.1K | 10 | 7 | - |

Table 2: Statistics of datasets: SNLI, MultiNLI, SciTail. L(P) and L(H) refer to the average length of two sentences. MultiNLI[1] and MultiNLI[2] indicate the in-domain and cross-domain datasets.

Then, we aggregate these representations $\mathbf{V}_p^{new}$ and $\mathbf{V}_q^{new}$ for the two sentences $p$ and $q$ in various ways in the interaction layer and the final feature vector $\mathbf{r}$ for the inference is obtained as follows:

$$\mathbf{r} = [\mathbf{V}_p^{new}; \mathbf{V}_q^{new}; \mathbf{V}_p^{new} - \mathbf{V}_q^{new}; \mathbf{V}_p^{new} \odot \mathbf{V}_q^{new}], \quad (17)$$

Finally, based on the aggregated feature $\mathbf{r}$, we use a multi-layer perceptron (MLP) classifier to predict the label:

$$\boldsymbol{v} = ReLU(\mathbf{W}_r \mathbf{r} + \mathbf{b}_r), \quad (18)$$

$$\hat{\boldsymbol{y}} = softmax(\mathbf{W}_v \boldsymbol{v} + \mathbf{b}_v). \quad (19)$$

where $\mathbf{W}_r, \mathbf{b}_r, \mathbf{W}_v$, and $\mathbf{b}_v$ are trainable parameters. The entire model is trained end-to-end, optimizing the standard multi-class cross-entropy loss function.

## 3 Experiments

In this section, we present the evaluation of our model. We first perform quantitative evaluation, comparing our model with other strong baselines. We then conduct some qualitative analyses to understand how ADIN achieve the asynchronous and multi-step inference between the premise sentence and the hypothesis sentence.

### 3.1 Dataset

We evaluate our model on three popular benchmarks: the Stanford Natural Language Inference (SNLI), the MultiGenre NLI Corpus (MultiNLI) and SciTail. Detailed statistical information of these datasets is shown in Table 2.

**SNLI** is a collection of 570k human written sentence pairs based on image captioning, supporting the task of natural language inference (Bowman et al., 2015). The labels are composed of entailment, neutral and contradiction. The data splits are provided in (Bowman et al., 2015).

**MultiNLI** The corpus (Williams et al., 2017) is a new dataset for NLI, which contains 433k sentences pairs. Similar to SNLI, each pair is labeled with one of the following relationships: entailment, contradiction, or neutral. We compare on two test sets (matched and mismatched) which represent in-domain and out-domain performance. We use the same data split as provided by (Williams et al., 2017).

**SciTail** We also include the newly released SciTail dataset (Khot et al., 2018) which is a binary entailment classification task constructed from science questions. This is the first entailment set that is created solely from natural sentences that already exist independently "in the wild" rather than sentences authored specifically for the entailment task. We use the same data split as in (Khot et al., 2018).

### 3.2 Models for Comparing

To analyze the effectiveness of our model, we evaluate some traditional and state-of-the-art methods as baselines as follows on the above three data sets:

- **DecompAtt** (Parikh et al., 2016) is a simple model that decomposes the problem into parallelizable attention computations.
- **ESIM** (Chen et al., 2016) is a previous state-of-the-art model for the natural language inference (NLI) task. It is a sequential model that incorporates the chain LSTM and the tree LSTM to infer local information between two sentences.
- **BiMPM** is proposed in (Wang et al., 2017). The model combines two sentence encoders and employs a multi-perspective matching mechanism in sentence pair modeling tasks.
- **DIIN** (Gong et al., 2017) is a novel class of neural network architectures that is able to achieve high-level understanding of the sentence pair by hierarchically extracting semantic features from the interaction space. The model uses word-by-word dimension-wise alignment tensors to encode the high-order alignment relationship between sentence pairs.
- **DGEM** (Khot et al., 2018) is a entailment model that exploits structure from the hypothesis only. This model shows the value of structured representation on just the hypothesis for NLI.
- **MwAN** (Tan et al., 2018) is a multiway attention network that applies multiple attention functions to model the matching between a pair of sentences.
- **CAFE** (Tay et al., 2018) compares and compresses alignment pairs using factorization layers which leverages the rich history of standard machine learning literature to achieve this task.
- **AF-DMN** (Duan et al., 2018) stacks multiple computational blocks in its matching layer to learn the interaction of the sentence pair better.
- **KIM** (Chen et al., 2018) is neural network-based NLI model that can benefit from external knowledge. The model is capable of leveraging external knowledge in co-attention, local inference collection, and inference composition components.

### 3.3 Experiment Configurations

Hyper-parameters may influence the performance of a neural network-based model. For all the three datasets, there are 3 inference sub-layers in the asynchronous inference Layer. An Adam (Kingma and Ba, 2014) optimizer with $\beta_1$ as 0.9 and $\beta_2$ as 0.999 is used to optimize all trainable parameters. The initial learning rate is set to 0.001 and is halved when the accuracy on the dev set decreases. We also apply dropout (Srivastava et al., 2014) on the all MLPs to avoid over-fitting, and the dropout rate is set to 0.2. For preprocessing, we just tokenize the sentences and lowercase the tokens.

For initialization, we initialize the word embeddings with a 300D Glove 840B (Pennington et al., 2014), and the out-of- vocabulary (OOV) words are randomly initialized. All word embeddings are updated during training. Parameters, including neural network parameters and OOV word embeddings, are initialized with a uniform distribution between $[-0.01, 0.01]$. The character embeddings are randomly initialized with 100D. We crop or pad each token to have 16 characters. And the 1D convolution kernel size for character embedding is 5.

| Model | train | test |
|---|---|---|
| **Single Models** | | |
| 200D DecompAtt (Parikh et al., 2016) | 90.5 | 86.8 |
| 600D ESIM (Chen et al., 2016) | 92.6 | 88.0 |
| BiMPM (Wang et al., 2017) | 90.9 | 87.5 |
| 448D DIIN (Gong et al., 2017) | 91.2 | 88.0 |
| 300D MwAN (Tan et al., 2018) | 94.5 | 88.3 |
| 300D CAFE (Tay et al., 2018) | 89.8 | 88.5 |
| 300D AF-DMN (Duan et al., 2018) | 94.5 | 88.6 |
| KIM (Chen et al., 2018) | 94.1 | 88.6 |
| **ADIN (ours)** | 93.6 | **88.8** |
| **Ensemble Models** | | |
| 600D ESIM (Chen et al., 2016) | 93.5 | 88.6 |
| BiMPM (Wang et al., 2017) | 93.2 | 88.8 |
| 448D DIIN (Gong et al., 2017) | 92.3 | 88.9 |
| 300D AF-DMN (Duan et al., 2018) | 94.9 | 89.0 |
| 300D CAFE (Tay et al., 2018) | 92.5 | 89.3 |
| **ADIN (ours)** | 95.6 | **89.5** |

Table 3: Comparison with previous models on the SNLI dataset.

| Model | Test Accuracy | |
|---|---|---|
| | Matched | Mismatched |
| **Single Models** | | |
| ESIM (Chen et al., 2016) | 72.3 | 72.1 |
| DIIN (Gong et al., 2017) | 78.8 | 77.8 |
| AF-DMN (Duan et al., 2018) | 76.9 | 76.3 |
| CAFE (Tay et al., 2018) | 78.7 | 77.9 |
| MwAN (Tan et al., 2018) | 78.5 | 77.7 |
| **ADIN (ours)** | **78.8** | **77.9** |
| **Ensemble Models** | | |
| DIIN (Gong et al., 2017) | 80.0 | 78.7 |
| CAFE (Tay et al., 2018) | 80.2 | 79.0 |
| MwAN (Tan et al., 2018) | 79.8 | 79.4 |
| **ADIN (ours)** | **80.3** | **79.6** |

Table 4: Comparison with previous models on the MultiNLI dataset.

### 3.4 Ensemble

The ensemble strategy is an effective method to improve model accuracy. Following (Wang et al., 2017), our ensemble model averages the probability distributions from five individual single ADINs, who have exactly identical architectures but distinguished initializations on parameters.

### 3.5 Quantitative Results

We use the accuracy to evaluate the performance of ADIN and other models on datasets SNLI, MultiNLI, and SciTail.

Table 3 shows the results of different models on the training and test sets of SNLI. In Table 3, the first category of methods are single models and the second category of methods are ensemble models. We show our model, ADIN, achieves state-of-the-art performance on the competitive leaderboard. In this table, KIM is neural network-based NLI model that can benefit from external knowledge, and other strong baselines encode and interact the both sentences symmetrically and in parallel.

Table 4 reports our results on the MultiNLI dataset. Similar to Table 3, the first category of methods are single models and the second category of methods are ensemble models. On MultiNLI, we compare on two test sets (matched and mismatched) which represent in-domain and out-domain performance. ADIN significantly outperforms ESIM, a strong baseline on the both test sets. An ensemble of ADIN models also achieve competitive result on the MultiNLI dataset.

As illustrated in Table 5, our model outperforms the baselines and achieves an accuracy of $84.6\%$ in the test set of the SciTail dataset. As such, empirical results demonstrate the effectiveness of our proposed ADIN model on the challenging SciTail dataset.

For the results on all three datasets, we conduct the students paired t-test. For SNLI and MultiNLI, the p-value of the significance test between the results of our model and AF-DMN is less than 0.01 and 0.05, respectively. For SciTail, the p-value of the significance test between the results of our model and CAFE is also less than 0.01. These results further prove the effectiveness of our model.

| Model | Accuracy |
|---|---|
| ESIM (Chen et al., 2016) | 70.6 |
| DecompAtt (Parikh et al., 2016) | 72.3 |
| DGEM (Khot et al., 2018) | 77.3 |
| CAFE (Tay et al., 2018) | 83.3 |
| **ADIN (ours)** | **84.6** |

Table 5: Comparison with previous models on the SciTail dataset.

| # layers | Dev | Test |
|---|---|---|
| 1 | 88.6 | 88.4 |
| 2 | 88.9 | 88.6 |
| 3 | 89.0 | 88.8 |

Table 6: Effect of number of asynchronous inference layers on the SNLI.

### 3.6 Model Analysis

To better understand the performance of ADIN, we analyze the effect of each key component of the proposed model on the SNLI dateset.

Table 6 shows the performance with a different number of asynchronous inference sub-layers. As we can see, with the number of sub-layers increases from 1 to 3, the performance increases both on the development set and the test set. As the level of reasoning deepens, the model captures more inferential information. Because of computational cost, we just set the number of sub-layers as 3 on SNLI and other two datesets.

In Table 7, we show the results of ablation study on our base model. After removing the Bi-LSTM in the asynchronous inference Layer, the model performance decrease by 0.3 percentage points on the test set. Furthermore, we study the effect of two inferential modules in one asynchronous inference sub-layer. Without the first inferential module, that is, without the reasoning process from premise to hypothesis, the model performance sharply decreases by 0.8 percentage points. However, remove the second module and the test accuracy decreases by 0.5 percentage points. (exchanged inference order) indicates that we get the inferential information to first model the premise sentence, and then model hypothesis sentence. The performance of the model is reduced to 88.3% after exchanging inference order between two sentences. The above three experiments reflect that the both modules are not equally important for the inference and the sentence pair for NLI is asymmetrical corpora. In the last comparative experiment, we explore the role of multi-level features. We remove character embedding and syntactical features and just keep word embedding as the representation. The test accuracy is reduced to 88.2%.

| Models | Dev | Test |
|---|---|---|
| (-Bi-LSTM) | 88.7 | 88.5 |
| (-first inferential module) | 88.4 | 88.0 |
| (-second inferential module) | 88.5 | 88.3 |
| (exchanged inference order) | 88.6 | 88.3 |
| (-char-emb - syntactical fea) | 88.5 | 88.2 |
| **ADIN (ours)** | 89.0 | **88.8** |

Table 7: Effect of components on the SNLI.

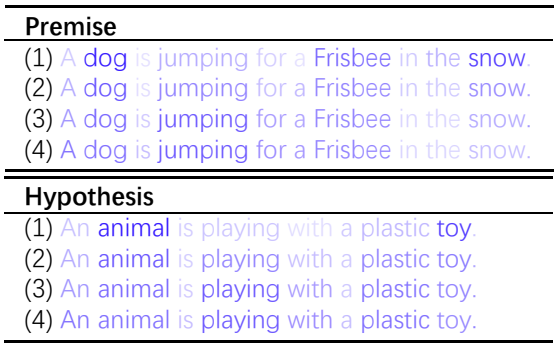


Figure 2: Gradient visualization of premise and hypothesis. (1) Gradient scale of $\mathbf{H}_p, \mathbf{H}_q$ on representation layer. (2) Gradient scale of $\mathbf{V}_p^1, \mathbf{V}_q^1$ on the first asynchronous inference sub-layer. (3) Gradient scale of $\mathbf{V}_p^2, \mathbf{V}_q^2$ on the second asynchronous inference sub-layer. (4) Gradient scale of $\mathbf{V}_p^3, \mathbf{V}_q^3$ on the third asynchronous inference sub-layer. Darker color corresponds to a higher scale of gradient, and implies a higher contribution to the final prediction.

### 3.7 Case study

To visually demonstrate the validity of the model, we do a qualitative study using the first example in Table 1.

$\mathbf{H}_p, \mathbf{H}_q$ are the hidden states at the representation layer of premise sentence and hypothesis sentence, and $\mathbf{V}_p^t, \mathbf{V}_q^t$ are the hidden states at the $t$-th asynchronous inference sub-layer. For a hidden state $\mathbf{h}_{p_i}$ of word $p_i$, we can calculate the gradient scale $\left\| \frac{\partial \mathcal{J}}{\partial \mathbf{h}_{p_i}} \right\|^2$ to show its contribution to the final prediction, where $\mathcal{J}$ is the cross-entropy loss. Figure 2 gives a visualization of the contribution to the final prediction of every word. As we can see, some phrases (like *jumping for a Frisbee* and *play-*

*ing with a plastic toy*) instead of isolated words (like *Frisbee* and *toy*) become more focused after an asynchronous inference layer. The results imply that ADIN could capture some higher-level patterns. As the level of reasoning deepens, the model captures more inferential information.

## 4 Related Work

As a long standing problem in NLP research, natural language inference (or textual entailment recognition) has been widely investigated for many years. Conventional works on NLI relies on handcrafted features such as syntactic information, n-gram overlapping and so on (Bowman et al., 2015; Heilman and Smith, 2010).

Benefiting from the development of deep learning and the availability of large-scale annotated datasets (Bowman et al., 2015), neural network-based models have also been successfully used for this task. And two categories of neural network-based models have been developed for this problem. The first set of models is sentence encoding-based and aims to find vector representation for each sentence and classifies the relation by using the concatenation of two vector representation (Bowman et al., 2016; Nangia et al., 2017; Mou et al., 2015). However, this kind of framework ignores the interaction between two sentences.

The other set of models uses the cross-sentence feature or inter-sentence attention from one sentence to another, and is hence referred to as a matching-aggregation framework. Parikh et al. (2016) use attention to decompose the problem into subproblems that can be solved separately, thus making it trivially parallelizable. Chen et al. (2016) propose a state-of-the-art model for the natural language inference (NLI) task. It is a sequential model that incorporates the chain LSTM and the tree LSTM to infer local information between two sentences. A novel class of neural network architectures is proposed in (Gong et al., 2017) that is able to achieve high-level understanding of the sentence pair by hierarchically extracting semantic features from interaction space. Tan et al. (2018) propose a multiway attention network that designs four attention functions to match words in corresponding sentences, aggregates the matching information from each function, and combines the information from all functions to obtain the final representation. Tay et al. (2018) compare and compress alignment pairs using factorization layers which leverages the rich history of standard machine learning literature to achieve this task. AF-DMN (Duan et al., 2018) stacks multiple computational blocks in its matching layer to learn the interaction of the sentence pair better. KIM (Chen et al., 2018) is capable of leveraging external knowledge in co-attention, local inference collection, and inference composition components to improve the performance. These methods all frame the inference problem as a semantic matching task and ignore the reasoning process.

Different from the above methods, ADIN is a neural network structure stacked with multiple asynchronous inference sub-layers, and each sub-layer consists of two local inference modules in an asymmetrical manner. This model deconstructs the reasoning process and implements the asynchronous and multi-step reasoning.

## 5 Conclusions

In this paper, we propose an asynchronous deep interaction network (ADIN) for natural language inference. To simulate human reasoning process, ADIN is stacked with multiple asynchronous inference sub-layers, and each sub-layer consists of two inferential modules in an asymmetrical manner. The model deconstructs the reasoning process and implements the asynchronous and multi-step reasoning. We evaluate our model on three popular benchmarks: SNLI, MultiNLI, and SciTail. The experiment results show that ADIN achieves competitive performance and outperforms strong baselines.

## Acknowledgments

The authors wish to thank the anonymous reviewers for their helpful comments. This work was partially funded by China National Key RD Program (No. 2018YFB1005104, 2018YFC0831105, 2017YFB1002104), National Natural Science Foundation of China (No. 61976056, 61532011, 61751201), Shanghai Municipal Science and Technology Major Project (No.2018SHZDZX01), STCSM (No.16JC1420401,17JC1420200).

## References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Samuel R Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. *arXiv preprint arXiv:1603.06021*.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Diana Inkpen, and Si Wei. 2018. Neural natural language inference models enhanced with external knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2406–2417.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2016. Enhanced lstm for natural language inference. *arXiv preprint arXiv:1609.06038*.

Chaoqun Duan, Lei Cui, Xinchi Chen, Furu Wei, Conghui Zhu, and Tiejun Zhao. 2018. Attention-fused deep matching network for natural language inference. In *IJCAI*, pages 4033–4040.

Yichen Gong, Heng Luo, and Jian Zhang. 2017. Natural language inference over interaction space. *arXiv preprint arXiv:1709.04348*.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5-6):602–610.

Michael Heilman and Noah A Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. ACL.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. Scitail: A textual entailment dataset from science question answering. In *AAAI*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *AAAI*, pages 2741–2749.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2016. Fully character-level neural machine translation without explicit segmentation. *arXiv preprint arXiv:1610.03017*.

Bill MacCartney and Christopher D Manning. 2008. Modeling semantic containment and exclusion in natural language inference. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 521–528. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2015. Natural language inference by tree-based convolution and heuristic matching. *arXiv preprint arXiv:1512.08422*.

Nikita Nangia, Adina Williams, Angeliki Lazaridou, and Samuel R Bowman. 2017. The repeval 2017 shared task: Multi-genre natural language inference with sentence representations. *arXiv preprint arXiv:1707.08172*.

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*, pages 1532–1543.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Chuanqi Tan, Furu Wei, Wenhui Wang, Weifeng Lv, and Ming Zhou. 2018. Multiway attention networks for modeling sentence pairs. In *IJCAI*, pages 4411–4417.

Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2018. Compare, compress and propagate: Enhancing neural architectures with alignment factorization for natural language inference. In *Proceedings of EMNLP*, pages 1565–1575.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. *arXiv preprint arXiv:1702.03814*.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Josh Tenenbaum. 2018. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. In *Advances in Neural Information Processing Systems*, pages 1039–1050.

—

# Keep Calm and Switch On! Preserving Sentiment and Fluency in Semantic Text Exchange

**Steven Y. Feng*** **Aaron W. Li*** **Jesse Hoey**

David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario, Canada
{sy2feng, w89li, jhoey}@uwaterloo.ca

## Abstract

In this paper, we present a novel method for measurably adjusting the semantics of text while preserving its sentiment and fluency, a task we call *semantic text exchange*. This is useful for text data augmentation and the semantic correction of text generated by chatbots and virtual assistants. We introduce a pipeline called *SMERTI* that combines entity replacement, similarity masking, and text infilling. We measure our pipeline's success by its *Semantic Text Exchange Score (STES)*: the ability to preserve the original text's sentiment and fluency while adjusting semantic content. We propose to use *masking (replacement) rate threshold* as an adjustable parameter to control the amount of semantic change in the text. Our experiments demonstrate that SMERTI can outperform baseline models on Yelp reviews, Amazon reviews, and news headlines.

## 1 Introduction

There has been significant research on style transfer, with the goal of changing the style of text while preserving its semantic content. The alternative where semantics are adjusted while keeping style intact, which we call *semantic text exchange (STE)*, has not been investigated to the best of our knowledge. Consider the following example, where the replacement entity defines the new semantic context:

| |
|---|
| Original Text: *It is **sunny** outside! Ugh, that means I must **wear sunscreen**. I hate being **sweaty** and **sticky all over.*** |
| Replacement Entity: weather = ***rainy*** |
| Desired Text: *It is **rainy** outside! Ugh, that means I must **bring an umbrella**. I hate being **wet** and **having to carry it around.*** |

The weather within the original text is sunny, whereas the actual weather may be rainy. Not only is the word *sunny* replaced with *rainy*, but the rest of the text's content is changed while preserving its negative sentiment and fluency.

With the rise of natural language processing (NLP) has come an increased demand for massive amounts of text data. Manually collecting and scraping data requires a significant amount of time and effort, and data augmentation techniques for NLP are limited compared to fields such as computer vision. STE can be used for text data augmentation by producing various modifications of a piece of text that differ in semantic content.

Another use of STE is in building emotionally aligned chatbots and virtual assistants. This is useful for reasons such as marketing, overall enjoyment of interaction, and mental health therapy. However, due to limited data with emotional content in specific semantic contexts, the generated text may contain incorrect semantic content. STE can adjust text semantics (e.g. to align with reality or a specific task) while preserving emotions.

One specific example is the development of virtual assistants with adjustable socio-emotional personalities in the effort to construct assistive technologies for persons with cognitive disabilities. Adjusting the emotional delivery of text in subtle ways can have a strong effect on the adoption of the technologies (Robillard et al., 2018). It is challenging to transfer style this subtly due to lack of datasets on specific topics with consistent emotions. Instead, large datasets of emotionally consistent interactions not confined to specific topics exist. Hence, it is effective to generate text with a particular emotion and then adjust its semantics.

We propose a pipeline called SMERTI (pronounced '*smarty*') for STE.[1] Combining entity replacement (ER), similarity masking (SM), and text

*Authors contributed equally

[1]Code for SMERTI (including Google Colab links) can be found at https://github.com/styfeng/SMERTI



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2701–2711,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

infilling (TI), SMERTI can modify the semantic content of text. We define a metric called the *Semantic Text Exchange Score (STES)* that evaluates the overall ability of a model to perform STE, and an adjustable parameter *masking (replacement) rate threshold (MRT/RRT)* that can be used to control the amount of semantic change.

We evaluate on three datasets: Yelp and Amazon reviews (He and McAuley, 2016), and Kaggle news headlines (Misra, 2018). We implement three baseline models for comparison: Noun WordNet Semantic Text Exchange Model (NWN-STEM), General WordNet Semantic Text Exchange Model (GWN-STEM), and Word2Vec Semantic Text Exchange Model (W2V-STEM).

We illustrate the STE performance of two SMERTI variations on the datasets, demonstrating outperformance of the baselines and pipeline stability. We also run a human evaluation supporting our results. We analyze the results in detail and investigate relationships between the semantic change, fluency, sentiment, and MRT/RRT. Our major contributions can be summarized as:

- We define a new task called *semantic text exchange (STE)* with increasing importance in NLP applications that modifies text semantics while preserving other aspects such as sentiment.
- We propose a pipeline *SMERTI* capable of multi-word entity replacement and text infilling, and demonstrate its outperformance of baselines.
- We define an evaluation metric for overall performance on semantic text exchange called the *Semantic Text Exchange Score (STES)*.

## 2 Related Work

### 2.1 Word and Sentence-level Embeddings

Word2Vec (Mikolov et al., 2013a,b) allows for analogy representation through vector arithmetic. We implement a baseline (W2V-STEM) using this technique. The Universal Sentence Encoder (USE) (Cer et al., 2018) encodes sentences and is trained on a variety of web sources and the Stanford Natural Language Inference corpus (Bowman et al., 2015). Flair embeddings (Akbik et al., 2018) are based on architectures such as BERT (Devlin et al., 2019). We use USE for SMERTI as it is designed for transfer learning and shows higher performance on textual similarity tasks compared to other models (Perone et al., 2018).

### 2.2 Text Infilling

Text infilling is the task of filling in missing parts of sentences called masks. MaskGAN (Fedus et al., 2018) is restricted to a single word per mask token, while SMERTI is capable of variable length infilling for more flexible output. Zhu et al. (2019) uses a transformer-based architecture. They fill in random masks, while SMERTI fills in masks guided by semantic similarity, resulting in more natural infilling and fulfillment of the STE task.

### 2.3 Style and Sentiment Transfer

Notable works in style/sentiment transfer include (Shen et al., 2017; Fu et al., 2018; Li et al., 2018; Xu et al., 2018). They attempt to learn latent representations of various text aspects such as its context and attributes, or separate style from content and encode them into hidden representations. They then use an RNN decoder to generate a new sentence given a targeted sentiment attribute.

### 2.4 Review Generation

Hovy (2016) generates fake reviews from scratch using language models. (Lipton et al., 2015; Dong et al., 2017; Juuti et al., 2018) generate reviews from scratch given auxiliary information (e.g. the item category and star rating). Yao et al. (2017) generates reviews using RNNs with two components: generation from scratch and review customization (Algorithm 2 in Yao et al. (2017)). They define review customization as modifying the generated review to fit a new topic or context, such as from a Japanese restaurant to an Italian one. They condition on a keyword identifying the desired context, and replace similar nouns with others using WordNet (Miller, 1995). They require a *"reference dataset"* (required to be *"on topic"*; easy enough for restaurant reviews, but less so for arbitrary conversational agents). As noted by Juuti et al. (2018), the method of Yao et al. (2017) may also replace words independently of context. We implement their review customization algorithm (NWN-STEM) and a modified version (GWN-STEM) as baseline models.

## 3 SMERTI

### 3.1 Overview

The task is to transform a corpus $C$ of lines of text $S_i$ and associated replacement entities $RE_i$ : $C = \{(S_1, RE_1), (S_2, RE_2), \ldots, (S_n, RE_n)\}$ to a modified corpus $\hat{C} = \{\hat{S}_1, \hat{S}_2, \ldots, \hat{S}_n\}$, where

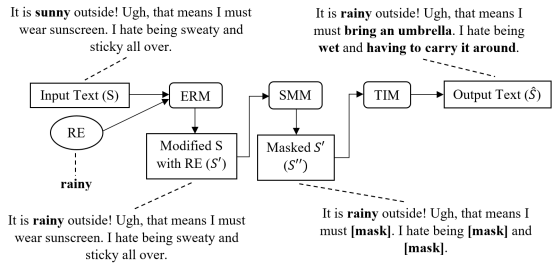


Figure 1: Overall architecture and example, showing the three modules: Entity Replacement (ERM), Similarity Masking (SMM), and Text Infilling (TIM)

$\hat{S}_i$ are the original text lines $S_i$ replaced with $RE_i$ and overall semantics adjusted. SMERTI consists of the following modules, shown in Figure 1:

1. Entity Replacement Module (ERM): Identify which word(s) within the original text are best replaced with the $RE$, which we call the Original Entity ($OE$). We replace $OE$ in $S$ with $RE$. We call this modified text $S'$.
2. Similarity Masking Module (SMM): Identify words/phrases in $S'$ similar to $OE$ and replace them with a *[mask]*. Group adjacent *[mask]*s into a single one so we can fill a variable length of text into each. We call this masked text $S''$.
3. Text Infilling Module (TIM): Fill in *[mask]* tokens with text that better suits the $RE$. This will modify semantics in the rest of the text. This final output text is called $\hat{S}$.

### 3.2 Entity Replacement Module (ERM)

For entity replacement, we use a combination of the Universal Sentence Encoder (Cer et al., 2018) and Stanford Parser (Chen and Manning, 2014).

#### Stanford Parser

The Stanford Parser is a constituency parser that determines the grammatical structure of sentences, including phrases and part-of-speech (POS) labelling. By feeding our $RE$ through the parser, we are able to determine its parse-tree. Iterating through the parse-tree and its sub-trees, we can obtain a list of constituent tags for the $RE$. We then feed our input text $S$ through the parser, and through a similar process, we can obtain a list of leaves (where leaves under a single label are concatenated) that are equal or similar to any of the $RE$ constituent tags. This generates a list of entities having the same (or similar) grammatical structure as the $RE$, and are likely candidates for the $OE$. We then feed these entities along with the $RE$ into the Universal Sentence Encoder (USE).

#### Universal Sentence Encoder (USE)

The USE is a sentence-level embedding model that comes with a deep averaging network (DAN) and transformer model (Cer et al., 2018). We choose the transformer model as these embeddings take context into account, and the exact same word/phrase will have a different embedding depending on its context and surrounding words.

We compute the semantic similarity between two embeddings $u$ and $v$: $sim(u, v)$, using the angular (cosine) distance, defined as: $\cos(\theta_{u,v}) = (u \cdot v)/(||u||||v||)$, such that $sim(u, v) = 1 - \frac{1}{\pi} arccos(\cos(\theta_{u,v}))$. Results are in $[0, 1]$, with higher values representing greater similarity.

Using USE and the above equation, we can identify words/phrases within the input text $S$ which are most similar to $RE$. To assist with this, we use the Stanford Parser as described above to obtain a list of candidate entities. In the rare case that this list is empty, we feed in each word of $S$ into USE, and identify which word is the most similar to $RE$. We then replace the most similar entity or word ($OE$) with the $RE$ and generate $S'$.

An example of this entity replacement process is in Figure 2. Two parse-trees are shown: for $RE$ (a) and $S$ (b) and (c). Figure 2(d) is a semantic similarity heat-map generated from the USE embeddings of the candidate $OE$s and $RE$, where values are similarity scores in the range $[0, 1]$.

As seen in Figure 2(d), we calculate semantic similarities between $RE$ and entities within $S$ which have *noun* constituency tags. Looking at the row for our $RE$ *restaurant*, the most similar entity (excluding itself) is *hotel*. We can then generate:

*$S'$ = i love this restaurant ! the beds are comfortable and the service is great !*

### 3.3 Similarity Masking Module (SMM)

Next, we mask words similar to $OE$ to generate $S''$ using USE. We look at semantic similarities between every word in $S$ and $OE$, along with semantic similarities between $OE$ and the candidate entities determined in the previous ERM step to broaden the range of phrases our module can mask. We ignore $RE$, $OE$, and any entities or phrases containing $OE$ (for example, *'this hotel'*).

After determining words similar to the $OE$ (discussed below), we replace each of them with a

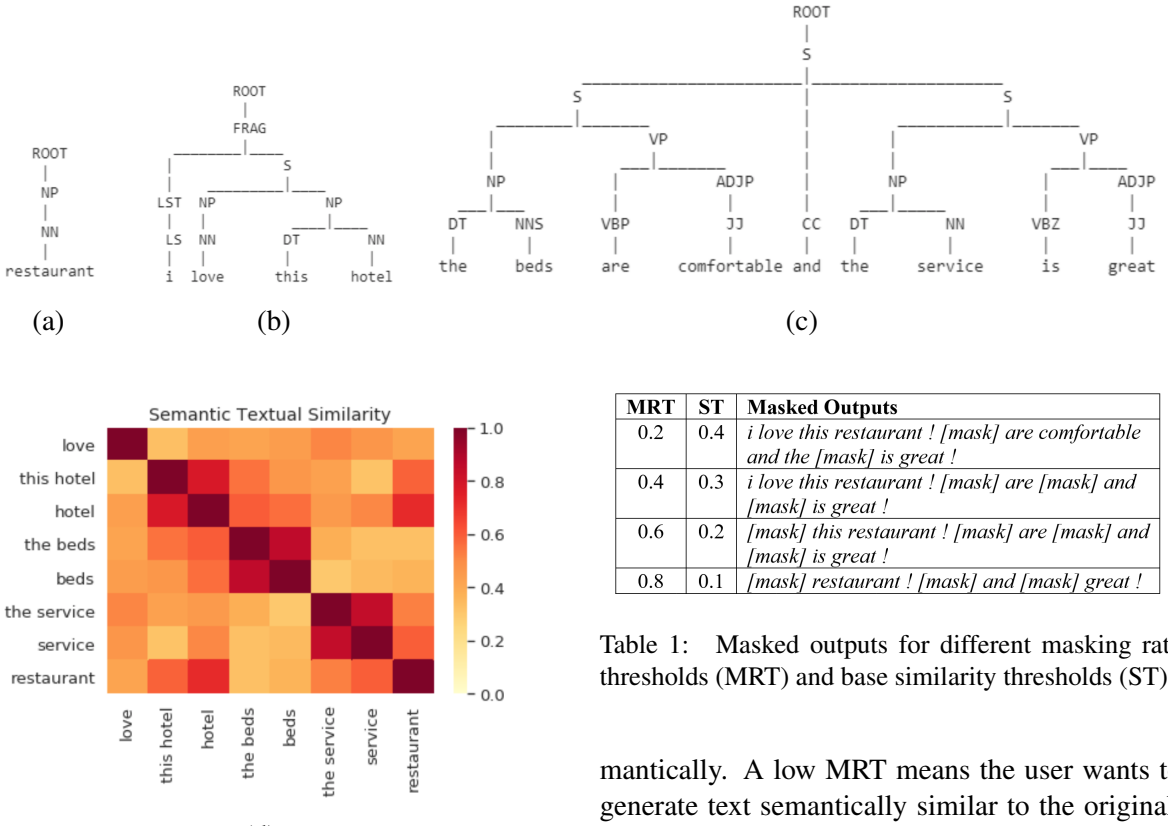


(a) (b) (c)

(d)

Figure 2: ERM example with $S$ = *i love this hotel ! the beds are comfortable and the service is great !* and $RE$ = *restaurant* showing (a) Parse tree for $RE$; (b) and (c) Parse tree for S; (d) Semantic similarity heat map

*[mask]* token. Next, we replace *[mask]* tokens adjacent to each other with a single *[mask]*.

We set a base similarity threshold (ST) that selects a subset of words to mask. We compare the actual fraction of masked words to the masking rate threshold (MRT), as defined by the user, and increase ST in intervals of 0.05 until the actual masking rate falls below the MRT.[2] Some sample masked outputs ($S''$) using various MRT-ST combinations for the previous example are shown in Table 1 (more examples in Appendix A).

The MRT is similar to the temperature parameter used to control the "novelty" of generated text in works such as Yao et al. (2017). A high MRT means the user wants to generate text very semantically dissimilar to the original, and may be desired in cases such as creating a lively chatbot or correcting text that is heavily incorrect semantically. A low MRT means the user wants to generate text semantically similar to the original, and may be desired in cases such as text recovery, grammar correction, or correcting a minor semantic error in text. By varying the MRT, various pieces of text that differ semantically in subtle ways can be generated, assisting greatly with text data augmentation. The MRT also affects sentiment and fluency, as we show in Section 6.5.

[2]There are certain cases where two or more outputs for different MRT may be equal. This occurs when a valid ST cannot be found that masks a larger portion of the sentence without going over MRT.

| MRT | ST | Masked Outputs |
|---|---|---|
| 0.2 | 0.4 | *i love this restaurant ! [mask] are comfortable and the [mask] is great !* |
| 0.4 | 0.3 | *i love this restaurant ! [mask] are [mask] and [mask] is great !* |
| 0.6 | 0.2 | *[mask] this restaurant ! [mask] are [mask] and [mask] is great !* |
| 0.8 | 0.1 | *[mask] restaurant ! [mask] and [mask] great !* |

Table 1: Masked outputs for different masking rate thresholds (MRT) and base similarity thresholds (ST)

### 3.4 Text Infilling Module (TIM)

We use two seq2seq models for our TIM: an RNN (recurrent neural network) model (Sutskever et al., 2014) (called SMERTI-RNN), and a transformer model (called SMERTI-Transformer).

#### Bidirectional RNN with Attention

We use a bidirectional variant of the GRU (Cho et al., 2014), and hence two RNNs for the encoder: one reads the input sequence in standard sequential order, and the other is fed this sequence in reverse. The outputs are summed at each time step, giving us the ability to encode information from both past and future context.

The decoder generates the output in a sequential token-by-token manner. To combat information loss, we implement the attention mechanism (Bahdanau et al., 2015). We use a Luong attention layer (Luong et al., 2015) which uses global attention, where all the encoder's hidden states are considered, and use the decoder's current time-step

hidden state to calculate attention weights. We use the dot score function for attention, where $h_t$ is the current target decoder state and $\bar{h}_s$ is all encoder states: $score(h_t, \bar{h}_s) = h_t^T \bar{h}_s$.

**Transformer**

Our second model makes use of the transformer architecture, and our implementation replicates Vaswani et al. (2017). We use an encoder-decoder structure with a multi-head self-attention token decoder to condition on information from both past and future context. It maps a query and set of key-value pairs to an output. The queries and keys are of dimension $d_k$, and values of dimension $d_v$. To compute the attention, we pack a set of queries, keys, and values into matrices $Q$, $K$, and $V$, respectively. The matrix of outputs is computed as:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (1)$$

Multi-head attention allows the model to jointly attend to information from different positions. The decoder can make use of both local and global semantic information while filling in each *[mask]*.

# 4 Experiment

## 4.1 Datasets

We train our two TIMs on the three datasets. The Amazon dataset (He and McAuley, 2016) contains over 83 million user reviews on products, with duplicate reviews removed. The Yelp dataset includes over six million user reviews on businesses. The news headlines dataset from Kaggle contains approximately $200,000$ news headlines from 2012 to 2018 obtained from HuffPost (Misra, 2018).

We filter the text to obtain reviews and headlines which are English, do not contain hyperlinks and other obvious noise, and are less than 20 words long. We found that many longer than twenty words ramble on and are too verbose for our purposes. Rather than filtering by individual sentences we keep each text in its entirety so SMERTI can learn to generate multiple sentences at once. We preprocess the text by lowercasing and removing rare/duplicate punctuation and space.

For Amazon and Yelp, we treat reviews greater than three stars as containing positive sentiment, equal to three stars as neutral, and less than three stars as negative. For each training and testing set, we include an equal number of randomly selected positive and negative reviews, and half as many neutral reviews. This is because neutral reviews only occupy one out of five stars compared to positive and negative which occupy two each. Our dataset statistics can be found in Appendix B.

## 4.2 Experiment Details

To set up our training and testing data for text infilling, we mask the text. We use a tiered masking approach: for each dataset, we randomly mask 15% of the words in one-third of the lines, 30% of the words in another one-third, and 45% in the remaining one-third. These masked texts serve as the inputs, while the original texts serve as the ground-truth. This allows our TIM models to learn relationships between masked words and relationships between masked and unmasked words.

The bidirectional RNN decoder fills in blanks one by one, with the objective of minimizing the cross entropy loss between its output and the ground-truth. We use a hidden size of 500, two layers for the encoder and decoder, teacher-forcing ratio of 1.0, learning rate of 0.0001, dropout of 0.1, batch size of 64, and train for up to 40 epochs.

For the transformer, we use scaled dot-product attention and the same hyperparameters as Vaswani et al. (2017). We use the Adam optimizer (Kingma and Ba, 2014) with $\beta_1 = 0.9, \beta_2 = 0.98$, and $\epsilon = 10^{-9}$. As in Vaswani et al. (2017), we increase the $learning\_rate$ linearly for the first $warmup\_steps$ training steps, and then decrease the $learning\_rate$ proportionally to the inverse square root of the step number. We set $factor = 1$ and use $warmup\_steps = 2000$. We use a batch size of 4096, and we train for up to 40 epochs.

## 4.3 Baseline Models

We implement three models to benchmark against.[3] First is NWN-STEM (Algorithm 2 from Yao et al. (2017)). We use the training sets as the "reference review sets" to extract similar nouns to the $RE$ (using $MIN_{sim}$ = 0.1). We then replace nouns in the text similar to the $RE$ with nouns extracted from the associated reference review set.

Secondly, we modify NWN-STEM to work for verbs and adjectives[4], and call this GWN-STEM. From the reference review sets, we extract similar nouns, verbs, and adjectives to the $RE$ (using

[3] See Appendix C for more implementation details

[4] WordNet can only work for single words (and not phrases). Also, it turns out that it cannot work for most adjective $RE$s, as discussed in Appendix C

*MIN$_{sim}$* = 0.1), where the $RE$ is now not restricted to being a noun. We replace nouns, verbs, and adjectives in the text similar to the $RE$ with those extracted from the associated reference review set.

Lastly, we implement W2V-STEM using Gensim (Řehůřek and Sojka, 2010). We train uni-gram Word2Vec models for single word $RE$s, and four-gram models for phrases. Models are trained on the training sets. We use cosine similarity to determine the most similar word/phrase in the input text to $RE$, which is the replaced $OE$. For all other words/phrases, we calculate $w_i' = w_i - w_{OE} + w_{RE}$, where $w_i$ is the original word/phrase's embedding vector, $w_{OE}$ is the $OE$'s, $w_{RE}$ is the $RE$'s, and $w_i'$ is the resulting embedding vector. The replacement word/phrase is $w_i'$'s nearest neighbour. We use similarity thresholds to adjust replacement rates (RR) and produce text under various replacement rate thresholds (RRT).

# 5 Evaluation

## 5.1 Evaluation Setup

We manually select 10 nouns, 10 verbs, 10 adjectives, and 5 phrases from the top 10% most frequent words/phrases in each test set as our evaluation $RE$s. We filter the verbs and adjectives through a list of sentiment words (Hu and Liu, 2004) to ensure we do not choose $RE$s that would obviously significantly alter the text's sentiment.[5]

For each evaluation $RE$, we choose one-hundred lines from the corresponding test set that does not already contain $RE$. We choose lines with at least five words, as many with less carry little semantic meaning (e.g. *'Great!', 'It is okay'*). For Amazon and Yelp, we choose 50 positive and 50 negative lines per $RE$.[6] We repeat this process three times, resulting in three sets of 1000 lines per dataset per POS (excluding phrases), and three sets of 500 lines per dataset for phrases. Our final results are averaged metrics over these three sets.

For SMERTI-Transformer, SMERTI-RNN, and W2V-STEM, we generate four outputs per text for MRT/RRT of 20%, 40%, 60%, and 80%, which represent upper-bounds on the percentage of the input that can be masked and/or replaced. Note that NWN-STEM and GWN-STEM can only evaluate on limited POS and their maximum replacement rates are limited.[7] We select *MIN$_{sim}$* values of 0.075 and 0 for nouns and 0.1 and 0 for verbs, as these result in replacement rates approximately equal to the actual MR/RR of the other models' outputs for 20% and 40% MRT/RRT, respectively.

## 5.2 Key Evaluation Metrics

**Fluency (SLOR)** We use syntactic log-odds ratio (SLOR) (Kann et al., 2018) for sentence level fluency and modify from their word-level formula to character-level ($SLOR_c$). We use Flair perplexity values from a language model trained on the One Billion Words corpus (Chelba et al., 2013):

$$SLOR_c(S) = \frac{1}{|S|}(ln(p_M(S)) - \frac{ln(\prod_{w \in S} p_M(w))}{\sum_{w \in S} |w|} \quad (2)$$

$$= -ln(PPL_s) + \frac{\sum_{w \in S} |w| ln(PPL_W)}{\sum_{w \in S} |w|} \quad (3)$$

where $|S|$ and $|w|$ are the character lengths of the input text $S$ and the word $w$, respectively, $p_M(S)$ and $p_M(w)$ are the probabilities of $S$ and $w$ under the language model $M$, respectively, and $PPL_S$ and $PPL_w$ are the character-level perplexities of $S$ and $w$, respectively. SLOR (from hereon we refer to character-level SLOR as simply SLOR) measures aspects of text fluency such as grammaticality. Higher values represent higher fluency.

We rescale resulting SLOR values to the interval [0,1] by first fitting and normalizing a Gaussian distribution. We then truncate normalized data points outside [-3,3], which shifts approximately 0.69% of total data. Finally, we divide each data point by six and add 0.5 to each result.

**Sentiment Preservation Accuracy (SPA)** is defined as the percentage of outputs that carry the same sentiment as the input. We use VADER (Hutto and Gilbert, 2014) to evaluate sentiment as positive, negative, or neutral. It handles typos, emojis, and other aspects of online text.

**Content Similarity Score (CSS)** ranges from 0 to 1 and indicates the semantic similarity between generated text and the $RE$. A value closer to 1 indicates stronger semantic exchange, as the output is closer in semantic content to the $RE$. We also use the USE for this due to its design and strong performance as previously mentioned.

## 5.3 Semantic Text Exchange Score (STES)

We come up with a single score to evaluate overall performance of a model on STE that combines

[5] A list of the chosen $RE$s along with more detailed explanation of how they were selected is in Appendix D

[6] We don't test on neutral reviews as evaluation of accuracy in sentiment is less well defined (i.e. most "neutral" reviews actually carry more positive or negative sentiment)

[7] See Appendix C for explanations

| Model | SPA | SLOR | CSS | STES |
|---|---|---|---|---|
| Input | --- | 0.5962 | 0.1166 | --- |
| SMERTI-Transformer | 0.6606 | **0.5255 (-11.86%)** | 0.2857 (+145.03%) | 0.4337 |
| SMERTI-RNN | 0.6574 | 0.5122 (-14.09%) | **0.2927 (+151.03%)** | **0.4354** |
| W2V-STEM | 0.6667 | 0.4672 (-21.64%) | 0.2851 (+144.51%) | 0.4197 |
| GWN-STEM | 0.8903 | 0.4864 (-18.42%) | 0.1419 (+21.70%) | 0.2934 |
| NWN-STEM | **0.9116** | 0.4832 (-18.95%) | 0.1335 (+14.49%) | 0.2814 |

Table 2: Overall average results by model (with % changes from the input)

the key evaluation metrics. It uses the harmonic mean, similar to the $F_1$ score (or F-score) (Chinchor, 1992; Rijsbergen, 1979), and we call it the *Semantic Text Exchange Score (STES)*:

$$STES = \frac{3 * A * B * C}{A * B + A * C + B * C} \quad (4)$$

where $A$ is SPA, $B$ is SLOR, and $C$ is CSS. STES ranges between 0 and 1, with scores closer to 1 representing higher overall performance. Like the $F_1$ score, STES penalizes models which perform very poorly in one or more metrics, and favors balanced models achieving strong results in all three.

### 5.4 Automatic Evaluation Results

Table 2 shows overall average results by model.[8] Table 3 shows outputs for a Yelp example.[9]

As observed from Table 3 (see also Appendix F), SMERTI is able to generate high quality output text similar to the $RE$ while flowing better than other models' outputs. It can replace entire phrases and sentences due to its variable length infilling. Note that for nouns, the outputs from GWN-STEM and NWN-STEM are equivalent.[10]

### 5.5 Human Evaluation Setup

We conduct a human evaluation with eight participants, 6 males and 2 females, that are affiliated project researchers aged 20-39 at the University of Waterloo.[11] We randomly choose one evaluation line for a randomly selected word or phrase for each POS per dataset. The input text and each model's output (for 40% MRT/RRT - chosen as a good middle ground) for each line is presented to participants, resulting in a total of 54 pieces of text, and rated on the following criteria from 1-5:

- *RE Match*: *"How related is the entire text to the concept of [X]", where [X] is a word or phrase (1 - not at all related, 3 - somewhat related, 5 - very related)*. Note here that [X] is a given $RE$.
- *Fluency*: *"Does the text make sense and flow well?" (1 - not at all, 3 - somewhat, 5 - very)*
- *Sentiment*: *"How do you think the author of the text was feeling?" (1 - very negative, 3 - neutral, 5 - very positive)*

Each participant evaluates every piece of text. They are presented with a single piece of text at a time, with the order of models, POS, and datasets completely randomized.

### 5.6 Human Evaluation Results

Average human evaluation scores are displayed in Table 4. *Sentiment Preservation* (between 0 and 1) is calculated by comparing the average *Sentiment* rating for each model's output text to the *Sentiment* rating of the input text, and if both are less than 2.5 (negative), between 2.5 and 3.5 inclusive (neutral), or greater than 3.5 (positive), this is counted as a valid case of *Sentiment Preservation*. We repeat this for every evaluation line to calculate the final values per model. Harmonic means of all three metrics (using rescaled 0-1 values of *RE Match* and *Fluency*) are also displayed.

## 6 Analysis

### 6.1 Performance by Model

As seen in Table 2, both SMERTI variations achieve higher STES and outperform the other models overall, with the WordNet models performing the worst. SMERTI excels especially on fluency and content similarity. The transformer variation achieves slightly higher SLOR, while the RNN variation achieves slightly higher CSS.

The WordNet models perform strongest in sentiment preservation (SPA), likely because they modify little of the text and only verbs and nouns. They achieve by far the lowest CSS, likely in part due to this limited text replacement. They also do not account for context, and many words (e.g. proper nouns) do not exist in WordNet. Overall, the WordNet models are not very effective at STE.

W2V-STEM achieves the lowest SLOR, especially for higher RRT, as supported by the example in Table 3 (see also Appendix F). W2V-STEM and WordNet models output grammatically incorrect text that flows poorly. In many cases, words are

[8] See Appendix E for tables and graphs of detailed results broken down by POS, dataset, and MRT/RRT

[9] See Appendix F for many more example outputs from each model for various POS and datasets

[10] See Appendix C for explanations

[11] The authors are not part of the human evaluation

**Input text**: great food , large portions ! my family and i really enjoyed our saturday morning breakfast .
**Replacement entity**: pizza

| MRT/RRT | Generated Output |
|---|---|
| SMERTI-Transformer | |
| 20% | great pizza , large slices ! my family and i really enjoyed our saturday morning lunch . |
| 40%,60% | great pizza , large slices ! service was terrific and i really enjoyed our saturday morning lunch . |
| 80% | great pizza , chewy crust ! nice ambiance and i really enjoyed it . |
| SMERTI-RNN | |
| 20% | great pizza , large delivery ! my family and i really enjoyed our saturday morning place . |
| 40%,60% | great pizza , large delivery ! good beer and i really enjoyed our saturday morning place . |
| 80% | great pizza , amazing pizza ! reasonable and i really enjoyed everyone . |
| W2V-STEM | |
| 20% | great pizza , large portions ! my family and i really enjoyed our saturday morning breakfast . |
| 40% | great pizza , large slices ! my family dough i crust enjoyed our saturday morning breakfast . |
| 60% | awesome pizza , large slices ! my mom dough i crust enjoyed our saturday morning bagel . |
| 80% | awesome pizza , slices slices ! my mom dough we crust liked our sunday morning bagel . |
| GWN / NWN-STEM | |
| 20% | great food , large stuff ! my family and i really enjoyed our saturday i breakfast . |
| 40% | great food , large stuff ! my i and i really enjoyed our saturday i breakfast . |

Table 3: Generated output text by model for various masking rates on a Yelp evaluation example

| Model | Input Text | SMERTI-Transformer | SMERTI-RNN | W2V-STEM | GWN-STEM | NWN-STEM |
|---|---|---|---|---|---|---|
| **RE Match** (1-5) | 1.8229 | **3.5833** | 3.5000 | 3.4792 | 2.2500 | 2.1250 |
| **Fluency** (1-5) | 4.1250 | 2.8750 | 2.8230 | 2.0830 | 2.5000 | **2.9583** |
| **Sentiment Preservation** (0-1) | --- | 0.7500 | 0.5833 | 0.6667 | 0.8333 | **1.0000** |
| **Harmonic Mean** (0-1) | --- | **0.5982** | 0.5446 | 0.4408 | 0.4245 | 0.4547 |

Table 4: Average human evaluation scores by model

| Model | Input Text | SMERTI-Transformer | SMERTI-RNN | W2V-STEM | GWN-STEM | NWN-STEM |
|---|---|---|---|---|---|---|
| **Avg. TTR** | 0.9488 | 0.9429 | 0.9367 | 0.9025 | 0.9042 | 0.8950 |

Table 5: Average TTR values by model

repeated multiple times. We analyze the average Type Token Ratio (TTR) values of each model's outputs, which is the ratio of unique divided by total words. As shown in Table 5, the SMERTI variations achieve the highest TTR, while W2V-STEM and NWN-STEM the lowest.

Note that while W2V-STEM achieves lower CSS than SMERTI, it performs comparably in this aspect. This is likely due to its vector arithmetic operations algorithm, which replaces each word with one more similar to the RE. This is also supported by the lower TTR, as W2V-STEM frequently outputs the same words multiple times.

### 6.2 Performance By Model - Human Results

As seen in Table 4, the SMERTI variations outperform all baseline models overall, particularly in *RE Match*. SMERTI-Transformer performs the best, with SMERTI-RNN second. The WordNet models achieve high *Sentiment Preservation*, but much lower on *RE Match*. W2V-STEM achieves comparably high *RE Match*, but lowest *Fluency*.

These results correspond well with our automatic evaluation results in Table 2. We look at the Pearson correlation values between *RE Match*, *Fluency*, and *Sentiment Preservation* with CSS, SLOR, and SPA, respectively. These are 0.9952, 0.9327, and 0.8768, respectively, demonstrating that our automatic metrics are highly effective and correspond well with human ratings.

| POS | Nouns | Verbs | Adjectives | Phrases |
|---|---|---|---|---|
| **Input SLOR** | 0.5971 | 0.5960 | 0.5977 | 0.5939 |
| **Input CSS** | 0.1175 | 0.1287 | 0.0935 | 0.1267 |
| **SMERTI SPA** | 0.7118 | 0.6586 | 0.6195 | 0.6460 |
| **SMERTI SLOR** | 0.5100 | 0.5078 | 0.4963 | 0.5613 |
| **SMERTI CSS** | 0.2982 | 0.2579 | 0.2434 | 0.3572 |
| **SMERTI STES** | 0.4464 | 0.4073 | 0.3876 | 0.4895 |

Table 6: Input text's avg. SLOR, CSS, and SMERTI's avg. SPA, SLOR, CSS, and STES by POS

### 6.3 SMERTI's Performance By POS

As seen from Table 6[12] , SMERTI's SPA values are highest for nouns, likely because they typically carry little sentiment, and lowest for adjectives, likely because they typically carry the most.

SLOR is lowest for adjectives and highest for phrases and nouns. Adjectives typically carry less semantic meaning and SMERTI likely has more trouble figuring out how best to infill the text. In contrast, nouns typically carry more, and phrases the most (since they consist of multiple words).

SMERTI's CSS is highest for phrases then nouns, likely due to phrases and nouns carrying

[12]Note that the SMERTI values in Tables 6 to 8 refer to the average between SMERTI-Transformer and SMERTI-RNN

| Dataset | Amazon | Yelp | News Headlines |
|---|---|---|---|
| **Input SLOR** | 0.6230 | 0.5532 | 0.6123 |
| **Input CSS** | 0.1113 | 0.1429 | 0.0955 |
| **SMERTI SPA** | 0.6738 | 0.7238 | 0.5793 |
| **SMERTI SLOR** | 0.5406 | 0.4765 | 0.5394 |
| **SMERTI CSS** | 0.2778 | 0.2835 | 0.3062 |
| **SMERTI STES** | 0.4326 | 0.4281 | 0.4381 |

Table 7: Input text's avg. SLOR, CSS, and SMERTI's avg. SPA, SLOR, CSS, and STES by dataset

| MRT / RRT | 20% | 40% | 60% | 80% |
|---|---|---|---|---|
| **SMERTI SPA** | 0.7994 | 0.7124 | 0.6202 | 0.5040 |
| **SMERTI SLOR** | 0.5489 | 0.5375 | 0.5197 | 0.4694 |
| **SMERTI CSS** | 0.2461 | 0.2557 | 0.2828 | 0.3722 |
| **SMERTI STES** | 0.4204 | 0.4181 | 0.4240 | 0.4410 |
| **SMERTI BLEU** | 0.5490 | 0.3527 | 0.1752 | 0.0448 |

Table 8: SMERTI's avg. SPA, SLOR, CSS, STES, and BLEU by MRT/RRT

more semantic meaning, making it easier to generate semantically similar text. Both SMERTI's and the input text's CSS are lowest for adjectives, likely because they carry little semantic meaning.

Overall, SMERTI appears to be more effective on nouns and phrases than verbs and adjectives.

### 6.4 SMERTI's Performance By Dataset

As seen in Table 7, SMERTI's SPA is lowest for news headlines. Amazon and Yelp reviews naturally carry stronger sentiment, likely making it easier to generate text with similar sentiment.

Both SMERTI's and the input text's SLOR appear to be lower for Yelp reviews. This may be due to many reasons, such as more typos and emojis within the original reviews, and so forth.

SMERTI's CSS values are slightly higher for news headlines. This may be due to them typically being shorter and carrying more semantic meaning as they are designed to be attention grabbers.

Overall, it seems that using datasets which inherently carry more sentiment will lead to better sentiment preservation. Further, the quality of the dataset's original text, unsurprisingly, influences the ability of SMERTI to generate fluent text.

### 6.5 SMERTI's Performance By MRT/RRT

From Table 8, it can be seen that as MRT/RRT increases, SMERTI's SPA and SLOR decrease while CSS increases. These relationships are very strong as supported by the Pearson correlation values of -0.9972, -0.9183, and 0.9078, respectively. When SMERTI can alter more text, it has the opportunity to replace more related to sentiment while producing more of semantic similarity to the $RE$.

Further, SMERTI generates more of the text itself, becoming less similar to the human-written input, resulting in lower fluency. To further demonstrate this, we look at average SMERTI BLEU (Papineni et al., 2002) scores against MRT/RRT, shown in Table 8. BLEU generally indicates how close two pieces of text are in content and structure, with higher values indicating greater similarity. We report our final BLEU scores as the average scores of 1 to 4-grams. As expected, BLEU decreases as MRT/RRT increases, and this relationship is very strong as supported by the Pearson correlation value of -0.9960.

It is clear that MRT/RRT represents a trade-off between CSS against SPA and SLOR. It is thus an adjustable parameter that can be used to control the generated text, and balance semantic exchange against fluency and sentiment preservation.

## 7 Conclusion and Future Work

We introduced the task of semantic text exchange (STE), demonstrated that our pipeline SMERTI performs well on STE, and proposed an STES metric for evaluating overall STE performance. SMERTI outperformed other models and was the most balanced overall. We also showed a trade-off between semantic exchange against fluency and sentiment preservation, which can be controlled by the masking (replacement) rate threshold.

Potential directions for future work include adding specific methods to control sentiment, and fine-tuning SMERTI for preservation of persona or personality. Experimenting with other text infilling models (e.g. fine-tuning BERT (Devlin et al., 2019)) is also an area of exploration. Lastly, our human evaluation is limited in size and a larger and more diverse participant pool is needed.

We conclude by addressing potential ethical misuses of STE, including assisting in the generation of spam and fake-reviews/news. These risks come with any intelligent chatbot work, but we feel that the benefits, including usage in the detection of misuse such as fake-news, greatly outweigh the risks and help progress NLP and AI research.

## Acknowledgments

We thank our anonymous reviewers, study participants, and Huawei Technologies Co., Ltd. for financial support.

## References


Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174, Brussels, Belgium. Association for Computational Linguistics.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, and Phillipp Koehn. 2013. One billion word benchmark for measuring progress in statistical language modeling. *CoRR*, abs/1312.3005.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750.

Nancy Chinchor. 1992. Muc-4 evaluation metrics. In *Proceedings of the 4th conference on Message understanding*, pages 22–29. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Li Dong, Shaohan Huang, Furu Wei, Mirella Lapata, Ming Zhou, and Ke Xu. 2017. Learning to generate product reviews from attributes. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 623–632.

William Fedus, Ian J. Goodfellow, and Andrew M. Dai. 2018. Maskgan: Better text generation via filling in the _____. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2018. Style transfer in text: Exploration and evaluation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, pages 507–517. International World Wide Web Conferences Steering Committee.

Dirk Hovy. 2016. The enemy in your own camp: How well can we detect statistically-generated fake reviews–an adversarial study. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 351–356.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Clayton J Hutto and Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth international AAAI conference on weblogs and social media*.

Mika Juuti, Bo Sun, Tatsuya Mori, and N Asokan. 2018. Stay on-topic: Generating context-specific fake restaurant reviews. In *European Symposium on Research in Computer Security*, pages 132–151. Springer.

Katharina Kann, Sascha Rothe, and Katja Filippova. 2018. Sentence-level fluency evaluation: References help, but can be spared! In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 313–323, Brussels, Belgium. Association for Computational Linguistics.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Juncen Li, Robin Jia, He He, and Percy Liang. 2018. Delete, retrieve, generate: a simple approach to sentiment and style transfer. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1865–1874, New Orleans, Louisiana. Association for Computational Linguistics.

Zachary Chase Lipton, Sharad Vikram, and Julian J. McAuley. 2015. Capturing meaning in product reviews with character-level generative text models. *ArXiv*, abs/1511.03683.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Rishabh Misra. 2018. News headlines dataset for sarcasm detection. https://rishabhmisra.github.io/publications/.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Christian S Perone, Roberto Silveira, and Thomas S Paula. 2018. Evaluation of sentence embeddings in downstream and linguistic probing tasks. *arXiv preprint arXiv:1806.06259*.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. http://is.muni.cz/publication/884893/en.

C. J. Van Rijsbergen. 1979. *Information Retrieval*, 2nd edition. Butterworth-Heinemann, Newton, MA, USA.

Julie M Robillard, Ian Cleland, Jesse Hoey, and Chris Nugent. 2018. Ethical adoption: A new imperative in the development of technology for dementia. *Alzheimer's & Dementia*, 14(9):1104–1113.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in neural information processing systems*, pages 6830–6841.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Jingjing Xu, Xu Sun, Qi Zeng, Xiaodong Zhang, Xuancheng Ren, Houfeng Wang, and Wenjie Li. 2018. Unpaired sentiment-to-sentiment translation: A cycled reinforcement learning approach. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 979–988, Melbourne, Australia. Association for Computational Linguistics.

Yuanshun Yao, Bimal Viswanath, Jenna Cryan, Haitao Zheng, and Ben Y Zhao. 2017. Automated crowdturfing attacks and defenses in online review systems. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1143–1158. ACM.

Wanrong Zhu, Zhiting Hu, and Eric P. Xing. 2019. Text infilling. *CoRR*, abs/1901.00158.

# Query-Focused Scenario Construction

**Su Wang**[1,2] **Greg Durrett**[3] **Katrin Erk**[1]
[1]Department of Linguistics
[2]Department of Statistics and Data Science
[3]Department of Computer Science
The University of Texas at Austin
shrekwang@utexas.edu gdurrett@cs.utexas.edu katrin.erk@mail.utexas.edu

## Abstract

The news coverage of events often contains not one but multiple incompatible accounts of what happened. We develop a query-based system that extracts compatible sets of events (scenarios) from such data, formulated as one-class clustering. Our system incrementally evaluates each event's compatibility with already selected events, taking order into account. We use synthetic data consisting of article mixtures for scalable training and evaluate our model on a new human-curated dataset of scenarios about real-world news topics. Stronger neural network models and harder synthetic training settings are both important to achieve high performance, and our final scenario construction system substantially outperforms baselines based on prior work.

## 1 Introduction

While a situation is developing, news reports often contain multiple contradictory stories (scenarios) of what happened, and it is hard to piece together the individual scenarios. For example, surrounding the disappearance of the Saudi journalist Jamal Khashoggi, there were initially multiple conflicting accounts of what happened. One states that he was the victim of a murder scheme; an alternative suggests that he walked out of the consulate alive. The task of identifying these individual scenarios is also being considered in the Active Interpretation of Disparate Alternatives (AIDA) program,[1] and in a recent Text Analysis Conference (TAC).[2]

We frame the task as query-based scenario discovery: given a *topic* (e.g., the disappearance of Jamal Khashoggi) and a *query* (e.g. *Jamal Khashoggi was murdered*), we want to retrieve a *scenario*, a set of compatible events, from the given reports. We formulate query-based scenario discovery as one-class clustering (Bekkerman and Crammer, 2008). We specifically focus on discovering a scenario of *compatible events* (Barzilay and Lapata, 2008; Chambers and Jurafsky, 2008, 2009; Mostafazadeh et al., 2017) in a collection of related and unrelated event-denoting sentences, which may contain conflicting and irrelevant information. We start with a query (see Figure 1) and then iteratively insert sentences into the "scenario-in-construction". Sentences are chosen based on overall compatibility as well as the ease with which scenario sentences can be arranged into an order. We additionally use an adapted relation network (Santoro et al., 2017) to assess connections between words.

**Query: Jamal Khashoggi was murdered.**

Jamal Khashoggi entered the consulate of Saudi Arabia consulate in Istambul. He exited the Saudi consulate after a few minutes. The team wanted to arrest Khashoggi but botched it. He never exited the Saudi consulate but died there. Khashoggi, according to the reporter, was seen on a flight leaving Turkey for Estonia. A team flew from Saudi Arabia to Turkey prior to Khashoggi's appointment at the consulate specifically to intercept him. The team was sent by the Saudi crown prince with the order to murder Khashoggi. Jamal A. Khashoggi works for The Washington Post, and is the editor-in-chief of Al-Arab News.

Figure 1: An example for query-based scenario construction. Given the **query**, we want to select event-denoting sentences from a document mixture to build a target scenario with a sequence of compatible events. The mixture also contains sentences which may be irrelevant or part of an alternative scenario.

For our evaluation, we collect a human-curated set of competing scenarios for real-world news topics. As collecting such data is costly, we follow past work in training our model on synthetic data consisting of document mixtures (Wang et al., 2018) and compare our models directly to theirs. We show that training on such synthetic data yields a model that can substantially outper-

[1] https://www.darpa.mil/program/active-interpretation-of-disparate-alternatives
[2] https://tac.nist.gov/2018/SM-KBP/index.html



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2712–2722, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

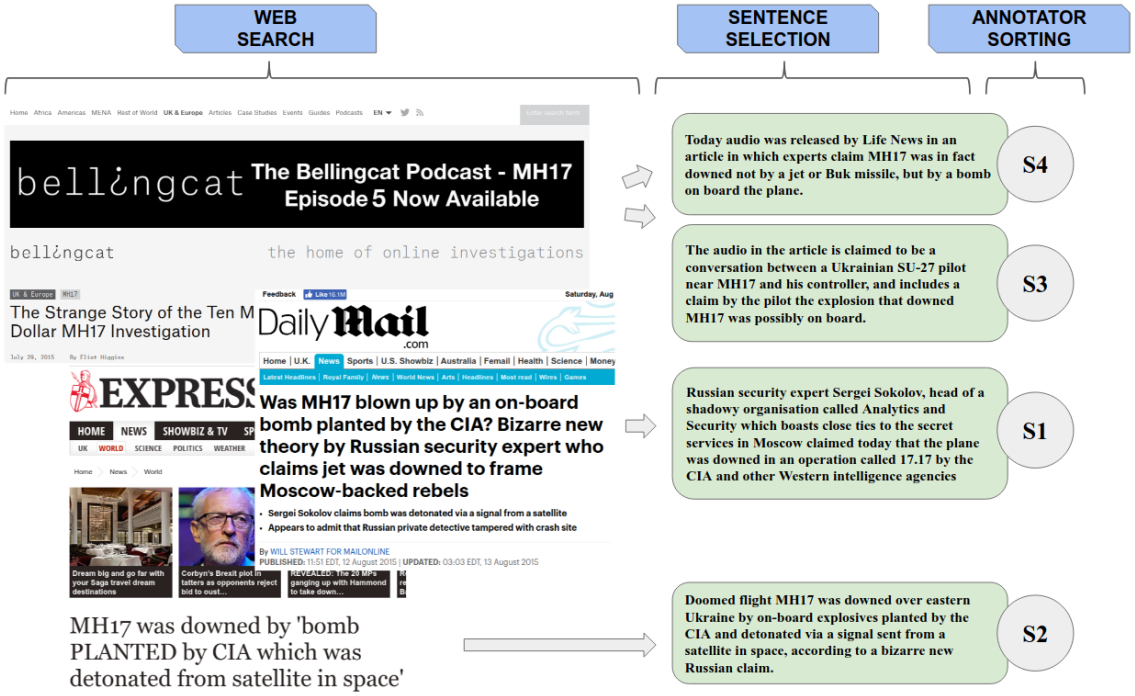


Figure 2: Phase 2 (cf. Table 1) of generating human evaluation data Human100: given the topic *Why did MH-17 crash?* and the scenario *MH-17 had a bomb on board*. The annotator searches the web and finds the webpages above. From these pages, she first selects 4 sentences which are relevant, then sorts them so that they make a consistent scenario that could be read from start to finish.

form lexical baselines and the strong neural model proposed in Wang et al.'s seminal work.

Our contribution is three-fold: (1) A query-based scenario construction task, for which we introduce a model to iteratively build a scenario with compatible events, exploiting ordering. (2) A human-curated evaluation set consisting of multiple accounts of real-world new events, along with a collection of scalably-built synthetic simulation datasets, which we show serve as an effective source of training data. (3) Comprehensive experiments and analysis that cast light on the properties of the task and data, as well as on the challenges.

## 2 Background

Our work traces its roots to research in *script* (Schank and Abelson, 1977; Mooney and DeJong, 1985) and *narrative schema* learning (Chambers and Jurafsky, 2008; Pichotta and Mooney, 2016). Early work explored tasks such as script modeling (Mooney and DeJong, 1985). Recent work built on the idea that compatibility of events can be learned from corpus data, evaluated on narrative cloze (Chambers and Jurafsky, 2009) and predicting-next-events (Pichotta and Mooney, 2016; Mostafazadeh et al., 2017).

We introduce a task with a more practical objective in mind: given a query or an information cue, extract the rest of the pieces to build a compatible scenario. The task is related to conversation disentanglement of multiple entangled conversations in a dialogue transcript (Elsner and Charniak, 2008, 2011; Jiang et al., 2018; Kummerfeld et al., 2019), and more closely to narrative clustering (Wang et al., 2018), i.e. identifying all the scenarios in an information source by grouping relevant sentences/events. Unlike Wang et al. (2018), we do not attempt to identify all the scenarios in the source, but are guided by one particular user's information need (e.g. the scenario about Khashoggi's murder, as opposed to all the theories regarding his disappearance, like in Wang et al. (2018)). Further, we do not assume the number of scenarios is known a priori (as Wang et al. (2018) do).

We phrase query-based scenario construction as one-class clustering. One-class models assume the availability of data for only one (positive) class, rather than positive and negative data. In particular, one-class clustering assumes that the data consists of few positive cases among a large group of outliers (Bekkerman and Crammer, 2008; Banu

| |
|---|
| **Topic question:** ***Why did MH-17 crash?*** |
| **scenario 1.** ***MH-17 had a bomb on board*** |
| (1) Doomed flight MH17 was downed over eastern Ukraine by on-board explosives planted by the CIA and detonated via a signal sent from a satellite in space, according to a bizarre new Russian claim. (2) Russian security expert Sergei Sokolov head of a shadowy organisation called Analytics and Security which boasts close ties to the secret services in Moscow claimed today that the plane was downed in an operation called 17.17 by the CIA and other Western intelligence agencies. (3) Today audio was released by Life News in an article in which experts claim MH17 was in fact downed not by a jet or Buk missile, but by a bomb on board the plane. (4) The audio in the article is claimed to be a conversation between a Ukrainian SU-27 pilot near MH17 and his controller, and includes a claim by the pilot the explosion that downed MH17 was possibly on board. |
| **scenario 2.** ***MH-17 was shot down by missiles.*** |
| (1) The shoot-down occurred in the War in Donbass, during the Battle of Shakhtarsk, in an area controlled by pro-Russian rebels. (2) The responsibility for investigation was delegated to the Dutch Safety Board (DSB) and the Dutch-led joint investigation team (JIT), who concluded that the airliner was downed by a Buk surface-to-air missile launched from pro-Russian separatist-controlled territory in Ukraine. (3) According to the JIT, the Buk that was used originated from the 53rd Anti-Aircraft Missile Brigade of the Russian Federation, and had been transported from Russia on the day of the crash, fired from a field in a rebel-controlled area, and the launcher returned to Russia after it was used to shoot down MH17. (4) Previously, the investigative website Bellingcat has pointed to involvement of the same brigade using open-source information. |

Table 1: An example from the crowdsourced Human100 dataset. Phase 1 (in bold): an MTurk worker writes (by web searching and editing) a topic question and two scenarios that answers the question. Phase 2: an annotator elaborates on the two scenarios (also through search-and-edit) with a compatible scenario.

and Karthikeyan, 2014). Finally, our task is superficially similar to query-based summarization (Otterbacher et al., 2005; Baumel et al., 2018) but has a different goal: we want to distinguish potentially conflicting narrative scenarios rather than conduct single-topic information compression. We also distinguish our work from multi-document summarization (McKeown et al., 2002; Radev et al., 2005), as we are explicitly drawing distinctions among conflictive scenarios rather than summarizing the entire (single) topic.

## 3 Data

This section first introduces our human-curated, realistic evaluation data for our objective. Then we describe how we synthesized various types of training data for our model.

### 3.1 Human-curated Data

Realistic data on this task is hard to obtain, as after a time, a single scenario tends to dominate in the news. The Linguistics Data Consortium (LDC) has data for the AIDA project, and the 2018 Text Analysis Conference (TAC) had a hypothesis generation task, but both use a single topic only (the Russia-Ukraine conflict of 2014), with no hypotheses available (TAC) or no hypotheses at sentence level (LDC). As a step in the direction of realistic data for the task, we had human annotators collect news items that have multiple scenarios of what happened around the same topic (Table 1).

We collected data in two phases: in phase 1, we asked workers on Amazon Mechanical Turk (MTurk) to provide (1) a topic in the form of an English wh-question; (2) two scenarios that answer the topic question mutually exclusively. See the bold text in Table 1 for an example. In phase 2 (Figure 2), a group of non-Turk annotators[3] pieced together English sentences from the web to elaborate on the scenarios from phase 1 (Table 1, non-bold). Annotators were instructed to build scenarios which can be read fluently. Sentences that could be copy-and-pasted directly from web search results were prioritized. When such sentences were unattainable, annotators were allowed to edit the style of the text to fit the scenario in the tone of a news report. We allowed this relaxation because, as mentioned above, often one scenario is dominant in web search results. For instance, in the Jamal Khashoggi story, English media almost unanimously report that he was a victim of murder, whereas the alternative scenario – he "disappeared and is still alive" is harder to find. A set of 100 mixtures were collected this way. We refer to the dataset as **Human100** (stats in Table 2).[4]

Wang et al. (2018) gauge the difficulty of a mixture by measuring the *topic similarity* between the target and the distractor(s) using the cosine distance between the average word embeddings. A mixture of documents (or scenarios) will typically be more difficult to separate if the scenarios/documents are more topically similar, since lexical cues are less reliable in this case. Specif-

[3] In our pilots for phase 2, the data Turkers created were not ideal, therefore we opted for hiring local annotators which produced higher quality results.

[4] Available before main conference: http://www.katrinerk.com/home/software-and-data/query-focused-scenario-construction

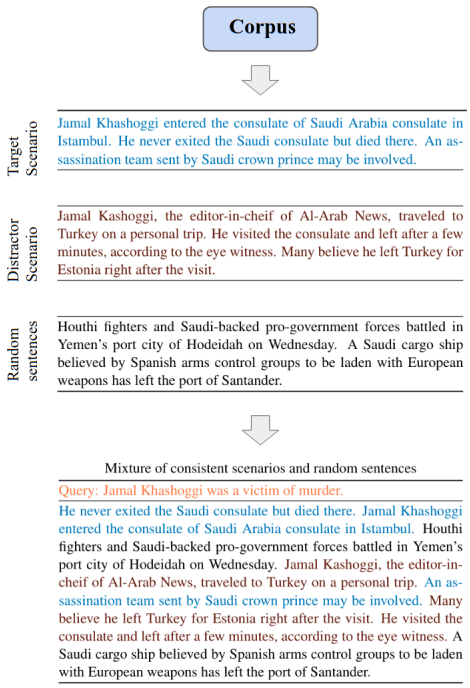


Figure 3: Synthetic data generation illustrated with our running example. First randomly sample news paragraphs (one is randomly assigned as the target scenario, the rest are distractors) and random sentences, then mix the target scenario with (a) other complete distractor scenarios (red) and (b) random sentences.

ically, Wang et al. found that their models struggled to separate scenarios even at 0.6 topical similarity, with accuracy on a binary clustering task dropping from 85% (all) to 68% (the hard ones). By this criterion Human100, at 0.8 average topic similarity cosine, is a hard dataset that challenges NLP models for their abilities to perform beyond shallow textual inference. Human-level performance is nonetheless fairly strong: 0.81 with distractor scenarios, and 0.97 with randomly sampled sentences (both are F1 scores, more details in Section 5).

### 3.2 Training with Proxy Synthetic Data

The idea we follow for synthetic data creation is the same as in Wang et al. (2018): we can use different articles as proxies for different scenarios, even though articles in the synthetic setting do not necessarily belong to the same topic. *Our hypothesis is that a model trained to predict which sentences came from the same news article will also do better at predicting which sentences come from the same scenario in the human-curated data.*

We synthesize data from two source corpora: the New York Times portion of English Gigaword (Graff et al., 2003) (**NYT**), which Wang et al. (2018) used to construct their document mixtures, and ROCStories (Mostafazadeh et al., 2017) (**ROC**). NYT is chosen for having the same domain as the human-constructed news data.[5] With ROC we want to gauge the generality of our approach out-of-domain: on news-only data a model could "cheat" by memorizing common named entities. We want to see to what extent models go beyond that to learn general event compatibility.

| Corpus | #scenarios | Vocab | Words/scenario | Sents/scenario |
|---|---|---|---|---|
| Human100 | 200 (2/topic) | - | 127.9 | 4 |
| NYT | 1.14m | 50,000 | 189.5 | 9.5 |
| ROC | 113k | 39,954 | 46.5 | 5.0 |

Table 2: Corpora statistics. Top: human-curated data; Bottom: synthetic data. For NYT, we truncated the vocabulary to the most frequent 50k (the full vocabulary is over 100k). For NYT and ROC we apply a 85%/5%/10% split to construct train/dev/test sets. The two datasets share 27,795 words in vocabulary.

The synthesis method is summarized in Figure 3, with corpus statistics in Table 2. In the first condition we mix a randomly sampled target scenario with a distractor scenario (also randomly sampled), following Wang et al. (2018). The mixtures are denoted **NYT/ROC-w18**. For the second condition, we replace the distractor scenario with unconnected randomly sampled sentences (corpus-wide), hence **NYT/ROC-rand**. We also combine both conditions, giving a mixture with both types of distractors. We also vary the number of distractor scenarios in a mixture (2, 3, or, 4, including the target scenario). To equalize the number of sentences in mixtures, we pad them all to a fixed number of sentences. We call these **NYT/ROC-2/3/4**.

## 4 Models

Given a query $q$ and a mixture of sentences, we want to select sentences that form a compatible scenario with the query. Our models select the sentences iteratively: the process begins with a *target set* $\mathcal{T}^{(1)} = \{q\}$ (i.e. initialized with only the query in the set) and a *candidate set* $\mathcal{C}^{(1)}$ (i.e.

[5] The Human100 dataset is created based on search results, which could conceivably be from a variety of domains, but annotators are largely selecting news articles about given topics. The topics themselves are general news with a skew towards politics, which is reflected in the NYT dataset as well. So both datasets consist mostly of political newswire writing, which we view as similar domain.

the mixture), and terminates at some time step $i$ with a predicted scenario $\mathcal{T}^{(i)}$.

We experiment with two termination conditions: (1) **fixed #sentences**: a pre-specified number of sentences are extracted; (2) **dynamic #sentences**: a special end-of-scenario token <end> is predicted as the next candidate. (1) simulates the case where the user desires to specify the amount of information to be extracted (i.e. a consistent yet not necessarily all-inclusive scenario), and (2) the case where the model finds a complete scenario.[6]

**Notation** We describe one step of candidate selection without loss of generality, thus whenever no confusion arises, we drop the timestep superscripts to use $\mathcal{T}, \mathcal{C}$ for simplification. $t \in \mathcal{T}$ denotes a sentence in the target scenario, and $c \in \mathcal{C}$ a candidate. We use bold lower case letters for embeddings. The acronyms for the models are introduced at the beginning of model description (e.g. COMP for compatibility-attention).

### 4.1 Architectures

**Compatibility-Attention (COMP)** In scenario building, intuitively we select a candidate $c_j$ that fits best with $\mathcal{T}$ such that the updated target scenario is $\mathcal{T} \cup \{c_j\}$, the most compatible scenario-in-construction possible. The prediction $\hat{c}_j$ is then:

$$\hat{c}_j = \underset{c_j \in \mathcal{C}}{\text{argmax}}\ \text{COMP}(\mathcal{T}, c_j) \quad (1)$$

For example, take the example in Figure 1, *He never exited the consulate but died there* is a good candidate, as it clearly relates to the scenario that Khashoggi was murdered, compatible with the current scenario-in-construction $\mathcal{T}^{(1)} = \{$*Jamal Khashoggi was murdered*$\}$.

Now note for $\mathcal{T}^{(i)}$, where $i > 1$ (i.e. multiple sentences in the scenario-in-construction), its member sentences do not contribute equally to the decision on a $\hat{c}_j$. For example, say $\mathcal{T}^{(2)} = \{$*Jamal Khashoggi was murdered*; *A team flew from Saudi Arabia ... to intercept him*$\}$, the first sentence is more informative for us to pick out *He never exited the consulate but died there* as a good candidate. We implement this with a bi-linear attention

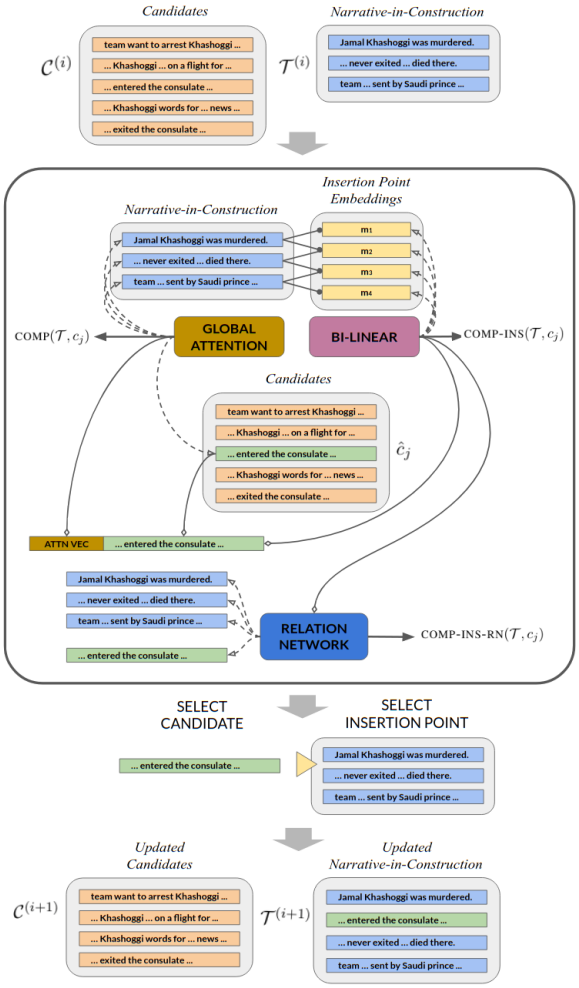


Figure 4: Iterative models for one step of candidate selection. Embeddings of scenario sentences in $\mathcal{T}$ are shown in light blue, candidate embeddings in $\mathcal{C}$ in orange, and the sentence embedding being processed and selected is in light green. COMP-ATT: only runs GLOBAL ATTENTION to produce $\text{COMP}(\mathcal{T}, c_j)$, which selects a candidate *but not an insertion point*. COMP-INS runs GLOBAL ATTENTION and BI-LINEAR to get $\text{COMP-INS}(\mathcal{T}, c_j)$, which selects a candidate as well as an insertion point. COMP-INS-RN additionally employs a RELATION NETWORK on top of COMP-INS.

layer with parameters $U$:

$$\alpha_{j,k} = \text{softmax}_k(\boldsymbol{c}_j^\intercal U \boldsymbol{t}_k)$$

$$\text{COMP}(\mathcal{T}, c_j) = \text{softmax}_j\left(\text{linear}\left(\sum_k \alpha_{j,k}\boldsymbol{t}_k\right)\right)$$

i.e. $c_j$ attends to the sentences $\{t_k\}$ in the current scenario, which computes a vector that is scored to compute the weight score of $c_j$ under COMP.

**Insertion-sort based selection (COMP-INS)** If $\mathcal{T}$ is an ordered scenario, it is possible to further

[6] We select the same number of sentences as in the target scenario for the fixed #sents condition. This is extra supervision compared to the dynamic #sents condition. Changing this number changes the precision/recall tradeoff, but it would not lead to significant gains in the F1 values we report.

improve the quality of the candidate selection by selecting a $c_j$ that is *easy-to-insert* into $\mathcal{T}$. For instance, for the example in Figure 1, the most readable update can be made by inserting *He never exited the consulate but died there* to the right of *Jamal Khashoggi was murdered*. Essentially, we imagine an insertion-sort based candidate selection technique: we iteratively pick out the easiest candidate to insert and maintain $\mathcal{T}$ as ordered. Crucially, note that we want to use ordering to aid clustering, rather than aiming for ordered scenarios: the model performance is only evaluated on clustering the correct set of sentences.

Let $m_k$ be the insertion point to the left of the sentence $t_k$. For $\mathcal{T}$ we have insertion points $\{m_1, \cdots, m_{|\mathcal{T}|+1}\}$. For each ⟨insertion point, candidate⟩ tuple $\langle m_k, c_j \rangle$, we want to compute a weight $z_{k \leftarrow j}$ to indicate the "easy-to-insert-ness" of $c_j$ to insertion point $m_k$. First we embed $m_k$ and $c_j$: $\boldsymbol{c}_j$ is embedded with a BiLSTM, and $\boldsymbol{m}_k$ is computed from the embedded sentences in $\mathcal{T}$:

$$\boldsymbol{m}_k = \begin{cases} \boldsymbol{t}_1 & \text{if } k = 1 \\ \boldsymbol{t}_{|\mathcal{T}|} & \text{if } k = |\mathcal{T}| + 1 \\ \text{average}(\boldsymbol{t}_{k-1}, \boldsymbol{t}_k) & \text{otherwise} \end{cases} \quad (2)$$

Finally, applying a bilinear function:

$$z_{k \leftarrow j} = \boldsymbol{m}_k^\intercal W [\boldsymbol{a}_j; \boldsymbol{c}_j] \quad (3)$$

where $[;]$ is concatenation. This results in a model $\text{COMP-INS}(\mathcal{T}, c_j)$ s.t.

$$\hat{c}_j = \underset{c_j \in \mathcal{C}}{\text{argmax}}\ \text{COMP-INS}(\mathcal{T}, c_j) \quad (4)$$

$$\text{COMP-INS}(\mathcal{T}, c_j) = \max_k (z_{k \leftarrow j}) \quad (5)$$

i.e. the easy-to-insert-ness of $c_j$ is indicated with its highest $z$ score over all the available insertion points: the larger the largest $z_{k \leftarrow j}$ is, the clearer the model is about where to insert $c_j$.

**Relation Networks (COMP-INS-RN)** Oftentimes word tokens in $\mathcal{T}$ and $\mathcal{C}$ are also indicative of which candidate is the best. E.g. in Figure 1, the candidate *He never exited the consulate but died there* has the event *died* which relates to the *murder* in the query. Similarly the entity *he* is likely anaphoric to *Jamal Khashoggi*. The relations make the sentence an ideal candidate.[7]

Adapting the *Relation Network* as per Santoro et al. (2017),[8] we summarize the relation between the word tokens in $c_j$ and $\mathcal{T}$ with a summary score $r_j$, i.e. how strongly the candidate is connected to the scenario-in-construction, based on the relations between its tokens and that of the scenario-in-construction. The process breaks down to three steps: first, we pair $c_j$ with each of the sentences $t_k \in \mathcal{T}$ and compute a sentence-sentence relational vector $\boldsymbol{v}_{j,k}$ which is the sum over all the word-word bi-linear contraction (the bi-linear contraction of two word embeddings $\boldsymbol{w}_a, \boldsymbol{w}_b$ is $\boldsymbol{w}_a^\intercal V \boldsymbol{w}_b$, where $\boldsymbol{w} \in \mathbb{R}^d, V \in \mathbb{R}^{d \times l \times d}$). Then, we average over the sentence-sentence relation vectors to obtain a summary vector $\boldsymbol{p}_j = \underset{k}{\text{average}}(\boldsymbol{v}_{j,k})$. Finally we compute $r_j = \text{linear}(\boldsymbol{p}_j)$. We incorporate the relation network with COMP-INS by adding $r_j$ to all the $z_{k \leftarrow j}$ over $k$. Patterning after the model descriptions above, we get a model $\text{COMP-INS-RN}(\mathcal{T}, c_j)$:

$$\hat{c}_j = \underset{c_j \in \mathcal{C}}{\text{argmax}}\ \text{COMP-INS-RN}(\mathcal{T}, c_j) \quad (6)$$

$$\text{COMP-INS-RN}(\mathcal{T}, c_j) = \max_k (z_{k \leftarrow j} + r_j) \quad (7)$$

### 4.2 Optimization

Since our models' supervision comes in the form of whole scenarios, supervising our iterative insertion clustering model is not completely straightforward. In particular, there may be multiple correct sentences that can be added to the current scenario. For example, in Figure 1, all the sentences labeled with (B) are possible candidates. We thus optimize the marginal log-likelihood of making any correct decision.[9]

Specifically, we minimize: $\mathcal{L} = -\log \sum_j p(c_j)$, which maximizes the sum of the probabilities of the correct candidates. For Eq. (5, 7), we optimize for

$$\mathcal{L} = -\log \sum_{(k,j)} p(\langle m_k, c_j \rangle) \quad (8)$$

i.e. maximizing the probability sum over all correct (insertion point, candidate) pairs.[10]

[7] Note that Wang et al. (2018) apply a mutual attention mechanism (Hermann et al., 2015) for the similar modeling purpose, but with many more parameters than our relation net. For practicality we believe the latter is a better option.

[8] Santoro et al. (2017) abstract word tokens as *objects* to summarize the relation between two sentences (or, in general, sequences) with a compound function $f(\sum_{i,j} g(o_{1,i}, o_{2,j}))$, where $o_{1,i}$ is the $i$-th token in sentence 1 (similar for $o_{2,j}$), and $f, g$ can be any model (e.g. a feedforward net).

[9] E.g. Durrett and Klein (2013) employ the same technique to optimize for multiple viable coreference candidates.

[10] Maximizing marginal likelihood has the attractive property that it allows our model to learn the "easiest" way to build the desired gold scenarios, rather than forcing one analysis.

At train time, we treat each timestep of scenario construction as a training example. We use a form of teacher forcing where we assume that a correct partial scenario has been built and we want to make some correct decision from there. The partial scenarios are built by adding *random* correct sentences, so the model learns to make correct decisions in a range of settings.

## 5 Experiments

We use scalably synthesized data for training and reserve the realistic data, which is expensive to produce, for evaluation. To better understand which techniques work, we also conduct evaluation on the synthetic data.

**Data Preparation** The statistics of our source corpora are summarized in Table 2. For all the synthetic datasets, we make 100k mixtures for training, 5k for validation and 10k for test, mixed from a 0.85/0.05/0.10 split of the total 113k scenarios. For NYT/ROC-w18, to properly compare with Wang et al. (2018), all mixtures have 2 scenarios. For NYT/ROC-2/3/4, we first mix 2/3/4 randomly sampled scenarios, then pad the resulting mixtures to a fixed lengths. For NYT/ROC-rand, we sample equal #sentences as for NYT/ROC-2/3/4.

**Baselines** We apply three baselines: (1) **UNIF**, which randomly selects $n-1$ candidates ($n-1$: #sentences in the target scenario minus the query). (2) **AVG**, an iterative model that always selects the candidate the embedding of which is the closest (in cosine) to the average embedding of the sentences in the scenario-in-construction. A sentence embedding is the average over each of its word embeddings. (3) **PAIRWISE**, which adapts Wang et al. (2018)'s best model. It predicts a probability for a pair of sentences to indicate how likely they are from the same scenario. PAIRWISE replaces the cosine in AVG with the pairwise model.

**Implementation** All the models are constructed with PyTorch 0.4.1 (Paszke et al., 2017). We use pretrained 1024-dim ELMo embeddings (Peters et al., 2018). The hidden size for the BiLSTM and the relation network are 200. We run 10 epochs with an initial learning rate of 1e-4 (with Adam (Kingma and Ba, 2014)).

**Evaluation** For clustering performance, we use macro-averaged F1, comparing our recovered cluster for each query to the gold cluster. For sorting-clustering correlation, Spearman's Rho ($\rho$); and for sorting per se, Kendall's Tau ($\tau$).[11]

| Condition | Model | NYT-w18 | NYT-rand | NYT-4 |
|---|---|---|---|---|
| Fixed#sent | UNIF | 0.45 | 0.18 | 0.18 |
| | AVG | 0.51 | 0.46 | 0.29 |
| | PAIRWISE | 0.68 | 0.64 | 0.53 |
| | COMP | 0.86 | 0.84 | 0.76 |
| | COMP-INS | 0.87 | 0.84 | 0.81 |
| | COMP-INS-RN | **0.93** | **0.92** | **0.84** |
| Dyn. #sent | COMP | 0.70 | 0.66 | 0.58 |
| | COMP-INS | 0.75 | 0.70 | 0.61 |
| | COMP-INS-RN | **0.78** | **0.73** | **0.65** |

Table 3: Intrinsic evaluation: F1 scores (testing) for models trained on different NYT mixtures in fixed and dynamic #sentences conditions. In the dynamic #sentences condition, the baselines no longer apply because they do not model a stopping condition.

### 5.1 Constructing Effective Training Mixtures

Which method for synthetic mixture creation leads to the best results on Human100? We first run our models in the three mixing conditions – scenario distractor only, random sentence distractor only, and both distractors (see Section 3.2), then evaluate both intrinsically and on the human-curated Human100. Here we only use our domain-proxy NYT-* datasets.

| COMP-INS-RN | NYT-w18 | NYT-rand | NYT-4 |
|---|---|---|---|
| Fixed #sent | 0.65 | 0.60 | 0.70 |
| Dyn. #sent | 0.60 | 0.56 | 0.62 |
| Human benchmark | | 0.82 | |

Table 4: Which scenario mixture method is the best? F1 scores on the Human100 data of the COMP-INS-RN model trained in different mixing conditions. The most complex condition (NYT-4) gives the best results.

Examining the results in Table 3,[12] we observe the hybrid mixtures with both types of distractors are the most difficult, with substantially lower performance. But how does this translate into human evaluation? In Table 4, we evaluate the best-performing model (COMP-INS-RN, trained on NYT-* sets) on Human100. We find that harder training conditions (i.e., the hybrid mixing) give stronger results on Human100. Our initial conclusion is: the more challenging hybrid mixing serves

[11] More discussion on the $\rho$-$\tau$ mix is at the end of sec 5.2.

[12] As we test on examples with more mixtures (i.e., going from NYT/ROC-2 to NYT/ROC-4), test accuracy steadily decreases (0.93/0.86/0.84 for NYT-2/3/4, 0.92/0.91/0.90 for ROC), as is to be expected. To avoid cluttering we only report scores on *-4 data.

better as a training proxy to the realistic data. We also see in both Tables 3 and 4 that the dynamic #sent setting, where the model needs to decide when to stop adding events, is considerably more difficult throughout.

### 5.2 Do Our Modules All Contribute?

We additionally evaluate our proposed modules – insertion-sort based selection and relation nets – to see which contributes substantially in the intrinsic evaluation (Table 3). The COMP-INS module achieves a gain of 3 points of F1 on average over COMP, and COMP-INS-RN improves 4.5 F1 on average over COMP-INS. In addition we see a clear and large margin of the models over the baselines. To evaluate the modules on Human100, we use the models trained on the best hybrid mixtures (Section 5.1). The results are summarized in Table 5.

| | COMP | COMP-INS | COMP-INS-RN |
|---|---|---|---|
| Fixed #sent | 0.62 | 0.68 | 0.70 |
| Dyn. #sent | 0.53 | 0.61 | 0.62 |
| Human benchmark | | 0.82 | |

Table 5: How do modeling modules contribute? F1 scores on Human 100 of different models with the best hybrid training mixtures (NYT-4).

Similar to the intrinsic evaluation, both modules improve performance across fixed and dynamic conditions. While in intrinsic evaluation, relation nets are the stronger contributor, insertion-sort based selection leads to a higher performance gain on Human100.

| COMP-INS | sorting | corr. clustering |
|---|---|---|
| Fixed #sent | 0.31 | 0.38 |
| Dyn. #sent | 0.30 | 0.40 |

Table 6: Sorting performance ($\tau$) and its correlation with clustering accuracy ($\rho$)

While sorting performance in itself is not very high, it has a reasonable correlation with clustering performance (Table 6): following Cui et al. (2018), we use Kendall's $\tau$ to compute sorting performance (as correlation of predicted and gold ordering). We then calculate the correlation between sorting performance and model performance, using Spearman's $\rho$ as the most widely used correlation measure in NLP.[13]

[13]Sorting and clustering performance are calculated one pair per instance. In computing $\tau$ we drop incorrectly extracted candidates as they do not have gold ordering with target sentences.

### 5.3 Do We Learn Compatibility that Generalizes?

As argued previously (Section 3.1), NYT contains plenty of shallow textual cues, meaning an expressive model can do well at the task doing bag-of-words clustering of the data rather than more sophisticated event compatibility inference.

| COMP-INS-RN | ROC-w18 | ROC-rand | ROC-4 |
|---|---|---|---|
| Train-on-ROC (in-domain) | 0.95 | 0.95 | 0.90 |
| Train-on-NYT (out-domain) | 0.85 | 0.87 | 0.81 |

Table 7: Generalization out-of-domain text: train on NYT-*/ROC-* and evaluate on ROC-* (fixed #sents).

The first question is: do the models generalize out-of-domain, particularly when textual cues are much fewer? We train our strongest COMP-INS-RN on NYT-* and evaluate on the corresponding ROC-* datasets (Table 7): in in-domain evaluation (i.e., train and test on ROC) our model produces excellent performance, and in out-of-domain evaluation (i.e., train on NYT test on ROC) it manages to keep up with fairly strong results. This indicates our model captures information beyond surface cues.

### 5.4 Final Model

From the domain-generalization test in Table 7, we see there is likely NYT-* sets do not subsume all the information in ROC-* sets. Exploiting all the data we have available, we combine ROC and NYT in a domain-joint training. The results in Table 8 show that in both fixed and dynamic #sent conditions, the model improves on the performance with single-domain training (Table 4, 5).

## 6 Analyzing Human-curated Data

To set up a human-level performance benchmark, we asked two additional annotators (they did not participate in the collection of Human100) to manually perform the same task as the models in fixed #sentences condition on a sample of 30 with randomly chosen query and target scenario. On average the F1 is 0.82 (one worker 0.81, the other 0.83). While even our best model (COMP-INS-RN) is inferior to the human-level performance, it draws quite close.

This however does not tell the whole story: qualitatively comparing the scenarios built by COMP-INS-RN vs. annotators, we observe human annotators tend to construct much more reasonable scenarios even when they include sentences

| COMP-INS-RN | R&N-w18 | R&N-rand | R&N-4 | rand-scenario |
|---|---|---|---|---|
| Fixed #sents | 0.70 | 0.61 | 0.74 | 0.90 |
| Dynamic #sents | 0.65 | 0.59 | 0.67 | 0.79 |
| UNIF | 0.50 | 0.50 | 0.50 | 0.42 |
| AVG | 0.49 | 0.50 | 0.51 | 0.50 |
| PAIRWISE | 0.56 | 0.60 | 0.56 | 0.64 |
| Human benchmark | | 0.82 | | 0.97 |

Table 8: **Left table**: F1 scores for Human100 evaluation with the best model (COMP-INS-RN) in fixed and dynamic #sentences conditions. The model is trained with three domain-joint training datasets. The model clears a sizable margin over the baselines, but falls short from human-level. **Right table**: F1 for COMP-INS-RN with *modified Human100*, i.e. the distractor scenario is now a random sample from NYT rather than one collected by an annotator. Human100 has a topic similarity of over 0.8 but the modified version only 0.45. This demonstrates *high* topic similarity is a strong contributor to the difficulty of Human100, which is true for both models and humans.

| |
|---|
| **Target scenario**: *Trump says Russia is the sole party to be blamed.* |
| (1) Trump says that Russia is the sole party to be blamed. (2) White House issued a statement which says Moscow is violating the Reagan-era agreement. (3) Secretary of State Mike Pompeo announced the decision to suspend the accord, declaring that countries must be held scenarioable when they break the rules. (4) "We can no longer be restricted by the treaty while Russia shamelessly violates it," Mr. Pompeo said. |
| **Distractor scenario**: *Russians accused the wrong doing on the part of the Trump administration* |
| (1) Russians accuse the wrong doing on the part of the Trump administration. (2) This is the latest step in the Trump administrations pattern of abandoning the diplomatic tools that have prevented nuclear war for 70 years. (3) Russia has also complained about the alleged lack of U.S. diplomacy. (4) Russian foreign minister Sergei Lavrov accused the U.S. of being obstinate. U.S. representatives arrived with a prepared position that was based on an ultimatum and centered on a demand for us to destroy this rocket, its launchers and all related equipment under US supervision. |

Table 9: Humans make more reasonable mistakes (the query is underscored): the annotator selected sentence (2) as a part of the target scenario, which, while not part of the gold, does make a comperent scenario. The model however chose sentence (1), which is in direct contradiction with the target scenario.

are not from the gold scenario (Table 9). This indicates that models for the task could benefit from textual inference capabilities (Cases et al., 2017), or from deeper meaning representations.

**Discussion** The results on Human100 are largely in line with those on synthetic datasets, which indicates that results on synthetic data gives a reasonable estimate of results on more realistic data. While the performance on Human100 is lower overall, the findings are encouraging. Further, realistic cases of scenario discovery have the property that different scenarios for the same topic have a high vocabulary overlap. This can be seen in Human100. This property of the data penalizes shallow processing based models while encouraging learning deeper semantics.

One caveat about Human100 is that the dataset is still relatively small; a larger dataset would be useful to strengthen quantitative analysis. Also, variable-size scenarios would be more realistic for evaluating the more general case. Finally, we would like to improve the crowdsourcing technique in future work: in phase 1, some collected scenarios are not entirely in conflict with each other, for example two talking points Trump made in his State of the Union Address. An extra step where additional crowdworkers rate the compatibility of scenarios could be useful. In phase 2, we would like to have scenarios which exhibit more nuanced conflicting points that capture a wider range of cues that distinguish different scenarios.

## 7 Conclusion

Identifying an individual scenario in a blend of contradictory stories is a task that is targeted by the Active Interpretation of Disparate Alternatives (AIDA) program as well in a recent Text Analysis Conference (TAC). We address this task through query-based scenario construction, and find sizable performance improvements both from taking sentence order into account (INS) and from encoding connections between words (RN). Evaluating on a new human-curated dataset, we find that the synthetic training data serves as a reasonable proxy for the human-curated data.

Our current model sometimes gets misled by superficially similar sentences, and it will be an important future direction to move towards deeper reasoning for the task. In addition, we plan to create larger human-curated datasets with variable size scenarios.

## Acknowledgments

This research was supported by the DARPA AIDA program under AFRL grant FA8750-18-2-0017. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of DARPA, DoD or the US government. We acknowledge the Texas Advanced Computing Center and Chameleon (Keahey et al., 2019) for providing grid resources that contributed to these results. We are grateful to the anonymous reviewers for helpful discussions.

## References


E. Afreen Banu and R. Karthikeyan. 2014. Text data linkage of different entities using OCCT one class clustering tree. In *Proceedings of IJCSIT*.

Regina Barzilay and Mirella Lapata. 2008. Modeling Local Coherence: An Entity-Based Approach. *Computational Linguistics*, 34(1):1–34.

Tal Baumel, Matan Eyal, and Michael Elhadad. 2018. Query focused abstractive summarization: Incorporating query relevance, multi-document coverage, and summary length constraints into seq2seq models. In *CoRR:1801.07704*.

Ron Bekkerman and Koby Crammer. 2008. One-class Clustering in the Text Domain. In *Proceedings of EMNLP*, pages 41–50.

Ignacio Cases, Minh-Thang Luong, and Christopher Potts. 2017. On the effective use of pretraining for natural language inference. In *CoRR:1710.02076*.

Nathanael Chambers and Daniel Jurafsky. 2008. Unsupervised Learning of Narrative Event Chains. In *Proceedings of ACL*.

Nathanael Chambers and Daniel Jurafsky. 2009. Unsupervised Learning of Narrative Schemas and Their Participants. In *Proceedings of ACL*.

Zhiyong Cui, Ruimin Ke, and Yinhai Wang. 2018. Deep Stacked Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction. In *Proceeings of IEEE*.

Greg Durrett and Dan Klein. 2013. Easy Victories and Uphill Battles in Coreference Resolution . In *Proceedings of NAACL*.

Micha Elsner and Eugene Charniak. 2008. You Talking to Me? A Corpus and Algorithm for Conversation Disentanglement. In *Proceedings of ACL*.

Micha Elsner and Eugene Charniak. 2011. Disentangling chat with local coherence models. In *Proceedings of ACL*.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2003. English Gigaword. *Linguistic Data Consortium, Philadelphia*, 4:1.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching Machines to Read and Comprehend. In *Proceedings of NeurIPS*.

Jyun-Yu Jiang, Francine Chen, Yang-Ying Chen, and Wei Wang. 2018. Learning to Disentangle Interleaved Conversational Threads with a Siamese Hierarchical Network and Similarity Ranking. In *Proceedings of NAACL*.

Kate Keahey, Pierre Riteau, Dan Stanzione, Tim Cockerill, Joe Mambretti, Paul Rad, and Paul Ruth. 2019. Chameleon: a scalable production testbed for computer science research. In Jeffrey Vetter, editor, *Contemporary High Performance Computing: From Petascale toward Exascale*, 1 edition, volume 3 of *Chapman Hall/CRC Computational Science*, chapter 5, pages 123–148. CRC Press, Boca Raton, FL.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: a Method for Stochastic Optimization. In *Proceedings of ICLR*.

Jonathan K. Kummerfeld, Sai R. Gouravajhala, Joseph J. Peper, Vignesh Athreya, Chulaka Gunasekara, Jatin Ganhotra, Siva Sankalp Patel, Lazaros C Polymenakos, and Walter Lasecki. 2019. A Large-Scale Corpus for Conversation Disentanglement. In *Proceedings of ACL*.

Kathleen R. McKeown, Regina Barzilay, David Evans, Vasileios Hatzivassiloglou, Judith L. Klavans, Ani Nenkova, Carl Sable, Barry Schiffman, and Sergey Sigelman. 2002. Tracking and Summarizing News on a Daily Basis with Columbias Newsblaster. In *Proceedings of NAACL*.

Raymond J. Mooney and Gerald DeJong. 1985. Learning Schemata for Natural Language Processing. In *Proceedings of IJCAI*.

Nasrin Mostafazadeh, Michael Roth, Annie Louis, Nathanael Chambers, and James F. Allen. 2017. LSDSem 2017 Shared Task: The Story Cloze Test. In *LSDSem 2017 Shared Task*.

Jahna Otterbacher, Günes Erkan, and Dragomir R. Radev. 2005. Using random walks for question-focused sentence retrieval. In *Proceedings of EMNLP*.

Adam Paszke, Sam Gross, Soumith Chintala, and Gregory Chanan. 2017. Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of NAACL*.

Karl Pichotta and Raymond Mooney. 2016. Using Sentence-Level LSTM Language Models for Script Inference. In *Proeedings of ACL*.

Dragomir Radev, Jahna Otterbacher, Adam Winkel, and Sasha Blair-Goldensohn. 2005. NewsInEssence: Summarizing Online News Topics. In *Communications of the ACM*.

Adam Santoro, David Raposo, David G.T. Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. 2017. A Simple Neural Network Module for Relational Reasoning. In *CoRR*.

Roger C. Schank and Robert P. Abelson. 1977. Scripts, Plans, Goals and Understanding. *Lawrence Erlbaum*.

Su Wang, Eric Holgate, Greg Durrett, and Katrin Erk. 2018. Picking Apart Story Salads. In *Proceedings of EMNLP*.

# Semi-supervised Entity Alignment via Joint Knowledge Embedding Model and Cross-graph Model

**Chengjiang Li**[1,2,3], **Yixin Cao**[4*], **Lei Hou**[1,2,3], **Jiaxin Shi**[1,2,3], **Juanzi Li**[1,2,3], **Tat-Seng Chua**[4]
[1]Department of Computer Science and Technology, Tsinghua University
[2]KIRC, Institute for Artificial Intelligence, Tsinghua University
[3]Beijing National Research Center for Information Science and Technology
[4]School of Computing, National University of Singapore, Singapore
{iamlockelightning,caoyixin2011,greener2009,shijx12}@gmail.com
lijuanzi2008@gmail.com, dcscts@nus.edu.sg

## Abstract

Entity alignment aims at integrating complementary knowledge graphs (KGs) from different sources or languages, which may benefit many knowledge-driven applications. It is challenging due to the heterogeneity of KGs and limited seed alignments. In this paper, we propose a semi-supervised entity alignment method by joint **K**nowledge **E**mbedding model and **C**ross-**G**raph model (**KECG**). It can make better use of seed alignments to propagate over the entire graphs with KG-based constraints. Specifically, as for the knowledge embedding model, we utilize TransE to implicitly complete two KGs towards consistency and learn relational constraints between entities. As for the cross-graph model, we extend Graph Attention Network (GAT) with projection constraint to robustly encode graphs, and two KGs share the same GAT to transfer structural knowledge as well as to ignore unimportant neighbors for alignment via attention mechanism. Results on publicly available datasets as well as further analysis demonstrate the effectiveness of KECG. Our codes can be found in https://github.com/THU-KEG/KECG.

## 1 Introduction

Recently, many Knowledge Graphs (KGs) (e.g., DBpedia (Lehmann et al., 2015), YAGO (Rebele et al., 2016) and BabelNet (Navigli and Ponzetto, 2012)) have emerged to provide structural knowledge for different applications. These separately constructed KGs contain heterogeneous but complementary contents; thus integrating KGs from different sources or languages into a unified KG becomes essential to better benefit knowledge-driven applications, ranging from information extraction (Han et al., 2018; Cao et al., 2018) to question answering (Cui et al., 2017).

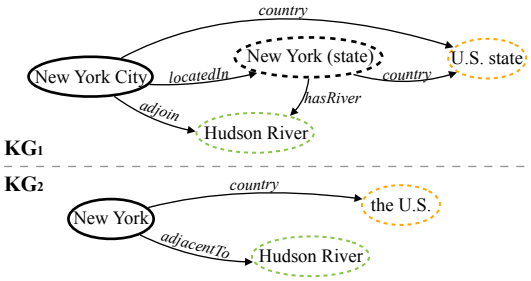


Figure 1: Example of the *New York City* entity and its neighbors in two heterogeneous KGs (Dashed circles in the same color means prior alignments).

Some efforts have been made to integrate KGs by aligning entities with their semantically same counterparts, namely entity alignment. Early entity alignment approaches either rely on human efforts (Mahdisoltani et al., 2013) or extra resources (Hu et al., 2011; Wang et al., 2013) to discover new equivalent entity pairs. Recent embedding approaches show significant improvement on entity alignment. We roughly classify these embedding models into two groups: KG-based models and graph-based models. KG-based models (Hao et al., 2016; Chen et al., 2017; Sun et al., 2017; Zhu et al., 2017; Sun et al., 2018; Chen et al., 2018) utilize existing KG representation learning methods to learn embeddings of entities and relations in different KGs, and then align them into a unified vector space. This type of method can not only preserve KG structures, but also implicitly complete KG with the missing links from existing knowledge. However, KG-based methods require a sufficient number of seed alignments, which is usually expensive to obtain.

To alleviate the burden of seed alignments, graph-based methods (Wang et al., 2018) utilize Graph Convolutional Network (GCN) (Kipf and Welling, 2017) to enhance entity embeddings with their neighbors' information, thus can make better use of seed alignments to propagate them over

* Corresponding author



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2723–2732, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

the entire graph. However, GCN-based models are sensitive to structural differences of KGs (Vaswani et al., 2017) and may not perform well for entity alignment, since different KGs are very heterogeneous. Another issue of this type of method is that they fail to consider relation types, which actually play an important role in entity alignment. As shown in Figure 1, the *New York City* entity has different surface forms in the two KGs (i.e., *New York City* in $KG_1$ and *New York* in $KG_2$). There are two possible mappings from the *New York City* and *New York (state)* in $KG_1$ to the *New York* in $KG_2$, and the *New York* in $KG_2$ shares the same neighbors *Hudson River* and *U.S. state* with *New York City* and *New York (state)* in $KG_1$. In this case, we need to further consider the relation types: *adjoin* and *hasRiver* to distinguish the two candidate alignments; otherwise, a graph-based method may be misled here.

In this paper, we propose a semi-supervised entity alignment method by joint **K**nowledge **E**mbedding model and **C**ross-**G**raph model (**KECG**), which combines the above two types of methods. The basic idea of KECG is to utilize a cross-graph model to embed entities into a unified vector space by using inner-graph structure and inter-graph alignments information; meanwhile the knowledge embedding model learns KG representations to implicitly complete different KGs towards consistency and model relational constraints among entities. There are two key points for high-quality jointly training. First, the completion from KG learning may exacerbate the heterogeneity between two KGs, because two KGs may contain different rich parts, which shall become richer during training. Second, different from conventional KG representation learning, entity alignment requires one-to-one mapping, which implies that the similar entities sharing common neighbors cannot be embeded closely, otherwise they shall be aligned incorrectly as shown in Figure 1.

Therefore, as for cross-graph model, we propose to utilize an extended Graph Attention Network (GAT) (Vaswani et al., 2017) as our cross-graph model that allows robustly encoding of KGs. To alleviate the negative impacts of heterogeneity, we encode different KGs with the same GAT, so that it not only highlights the entities that are important for alignment in the graph, but also transfers structural knowledge via shared parameters. We also extend it with constraints on weight matrices, making our model more efficient and effective. Moreover, we employ the nearest neighbor sampling strategy to differentiate the entities that are embeded closely but not the same to further boost the performance. We summarize the main contributions as follows:

1. We propose a novel semi-supervised KECG model for entity alignment by joint knowledge embedding model and cross-graph model.

2. We utilize an extended GAT to alleviate the negative impacts of heterogeneous KGs, and employ the nearest neighbor sampling strategy for KG representation learning towards one-to-one mapping, making the combination more suitable for entity alignment.

3. We evaluate KECG on five publicly available datasets including three cross-lingual datasets and two large-scale datasets from different sources. Compared with four state-of-the-art baselines, experimental results and further analysis demonstrate the outperformance of KECG.

## 2 Background

### 2.1 Problem Formulation

Formally, we represent two heterogeneous KGs as $G_1 = (E_1, R_1, T_1)$ and $G_2 = (E_2, R_2, T_2)$, where $E_i$, $R_i$, $T_i$ represent the sets of entity, relation and triplet in $G_i, i \in \{1, 2\}$, respectively. $N_e = \{e'|(e, r, e') \in T\} \cup \{e'|(e', r, e) \in T\}$ is the set of neighbors of the entity $e$ in G. $S = \{(e_1, e_2) \in E_1 \times E_2 | e_1 \leftrightarrow e_2\}$ is a set of labeled entity pairs that are same in semantics, where an equivalence relation $\leftrightarrow$ holds between $e_1$ and $e_2$, e.g., $e_1$ in $G_1$ describes the same real-world entity with $e_2$ in $G_2$. Entity alignment aims to find the remaining semantically same entity pairs in $E_1 \times E_2$. For convenience, we put $G_1$ and $G_2$ together into one large graph $G = G_1 + G_2$, $R = R_1 \cup R_2$, $T = T_1 \cup T_2$, and $n = |E_1| + |E_2|$ is the total number of entities. We use bold-face letters to denote the vector representations of the corresponding terms throughout the paper.

### 2.2 Graph Neural Networks (GNNs)

GNNs are a type of neural network model that directly deals with structured data (Bruna et al.,

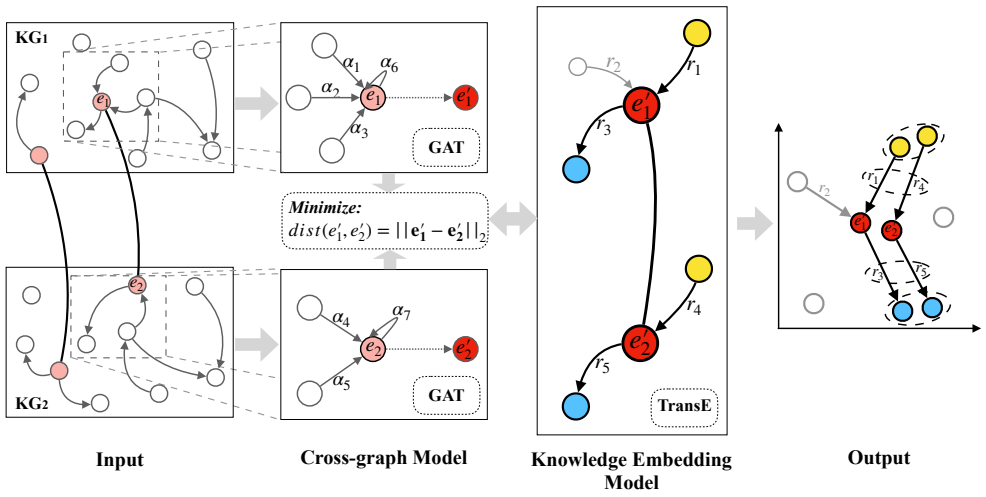


Figure 2: The overview framework of KECG. Solid lines (in Input) connect pre-aligned entities in two KGs. Nodes in different sizes (in Knowledge Embedding Model) show the attention of the center nodes over their neighbors. Circled nodes in the output are the potentially equivalent entities that we want to find out (in Output).

2014; Defferrard et al., 2016; Hamilton et al., 2017). They take a graph as an input, output labels for each node, and are similar to a propagation model: to enhance the features of a node according to its neighbor nodes. Among them, GCN (Kipf and Welling, 2017) is a simplification of spectral graph convolutions, whose formulation is as follows:

$$\mathbf{H}^{(l+1)} = \sigma(\mathbf{A}\mathbf{H}^{(l)}\mathbf{W}^{(l)})$$

where $\mathbf{A}$ is a normalized adjacent matrix of the input graph with self-connection, $\mathbf{H}^{(l)}$ and $\mathbf{W}^{(l)}$ are the hidden states and weights in the $l_{th}$ layer, and $\sigma(\cdot)$ is a non-linear activation, such as $ReLu$. Subsequently, Vaswani et al. (2017) introduce attention mechanism into GCN by dynamically calculating $\mathbf{A}$ according to the attention values between each entity pairs, which is capable of mining structural knowledge better.

## 3 Method

To combine KG-based methods and graph-based methods with consideration of seed limit and KG heterogeneity, we propose a novel semi-supervised method KECG for entity alignment. As shown in Figure 2, KECG consists of two parts: cross-graph model and knowledge embedding model. The input of KECG includes a composition $G$ of two KGs and their prior alignment set $S$.

### 3.1 Cross-graph Model

The goal of our cross-graph model is to utilize structural information, including inner-graph structure and inter-graph alignments, to embed entities into a unified vector space. Due to the incompleteness of KGs, there may be some entities and relations existing in one KG but missing in another KG. We utilize an extended GAT with projection constraint as an encoder to embed the KGs while paying different attention over their neighbors, therefore it is able to alleviate the negative impact of heterogeneity by ignoring some unimportant neighbors for alignment. Besides, the graph convolution operation allows us to leverage both the labeled entities and the abundant unlabeled entity information for alignment, making our method naturally semi-supervised.

The input of the encoder includes an entity embedding matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ and neighbor sets of all entities as KG structure, where $d$ is the dimension of entities. The KG encoder is built by stacking multiple GAT layers:

$$\mathbf{H}^{(l+1)} = \sigma(\mathbf{A}^{(l)}\mathbf{H}^{(l)}\mathbf{W}^{(l)}) \quad (1)$$

where $\mathbf{H}^{(l)} \in \mathbb{R}^{n \times d^{(l)}}$ and $\mathbf{W}^{(l)} \in \mathbb{R}^{d^{(l)} \times d^{(l+1)}}$ are the hidden states and weights in the $l_{th}$ layer, $\mathbf{H}^{(0)} = \mathbf{X}$, $\sigma(\cdot)$ is a non-linear activation chosen as $ReLU(\cdot) = max(0, \cdot)$, and $\mathbf{A}^{(l)} \in \mathbb{R}^{n \times n}$ is a connectivity matrix computed by self-attention over the input graph. Considering a single element $a_{ij}^{(l)}$ of $\mathbf{A}^{(l)}$, the weight from entity $e_i$ to entity $e_j$, we compute it using self-attention mechanism:

$$a_{ij}^{(l)} = softmax(c_{ij}^{(l)}) = \frac{exp(c_{ij}^{(l)})}{\sum_{e_k \in N_{e_i} \cup \{e_i\}} exp(c_{ik}^{(l)})} \quad (2)$$

, where $N_{e_i} \cup \{e_i\}$ is the neighbor set of $e_i$ with self-loop, $c_{ij}^{(l)}$ is the attention coefficient of entity $e_i$ to entity $e_j$. Following Velickovic et al. (2018), the attention coefficient $c_{ij}^{(l)}$ is computed as follows:

$$c_{ij}^{(l)} = LeakyReLU(\mathbf{q}^T[\mathbf{W}^{(l)}\mathbf{h}_i^{(l)} \oplus \mathbf{W}^{(l)}\mathbf{h}_j^{(l)}]) \quad (3)$$

where $\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)} \in \mathbf{H}^{(l)}$ are the hidden states of $e_i$ and $e_j$, respectively, $LeakyReLU$ (Maas et al., 2013) is a nonlinear function, $\mathbf{q} \in \mathbb{R}^{2d^{(l)}}$ is a learnable parameter, $\cdot^T$ represents transposition and $\oplus$ indicates vector concatenation. Let $L$ be the number of layers. After $L$ iterations of convolution, the hidden state $\mathbf{H}^{(L)}$ integrates both features of entities and their neighbors, the $i_{th}$ row of which represents the **attention enhanced embedding** of entity $e_i$.

**Projection Constraint.** Note that in every forward propagation, we use $\mathbf{W}^{(l)}$ to transform from $d^{(l)}$ dimension vector space to $d^{(l+1)}$ dimension. Motivated by (Yang et al., 2015), we keep the dimensions of the layers and embeddings to be the same, and restrict the projection matrix $\mathbf{W}$ to be a diagonal matrix. Such constraint can reduce the number of parameters and computations, increase the generalizability of the model. That's why we claim that we extend the normal GAT.

**Objective $O_C$.** For semi-supervised entity alignment, we then minimize the representation distance of equivalent entities over all labeled alignments similar to (Wang et al., 2018):

$$O_C = \sum_{(e_i,e_j)\in S} \sum_{(e_i^{'},e_j^{'})\in S'} [dist(e_i, e_j) + \gamma_1 - dist(e_i^{'}, e_j^{'})]_+ \quad (4)$$

where $dist(e_i, e_j) = ||\mathbf{e}_i - \mathbf{e}_j||_2$ is the $L_2$ distance between the aligned entity pair $(e_i, e_j)$, representations of $e_i$ and $e_j$ are from the attention enhanced embedding matrix $\mathbf{H}^{(L)}$, $S'$ represents the negative pair set of $S$ generated by nearest neighbor sampling (Kotnis and Nastase, 2017), and $\gamma_1 > 0$ is a margin hyper-parameter. Because an entity can only have one corresponding entity in another KG, the entity closest to the corresponding entity in the same KG should be the best choice as a negative example to accurately discriminate the target entity. More formally, given a pre-aligned entity pair $(e_i, e_j) \in S$, where $e_i \in E_1$, $e_j \in E_2$, $K_2$ is the number of negative samples, we choose $K_2$ entities that are nearest to $e_j$ in $E_2$ as the negative samples of $e_i$, and vice versa for $e_j$. After that, each pre-aligned entity pair will have $2*K_2$ negative samples. We utilize the $L_2$ distance as the measure to search for the nearest negative samples due to its superior performance in experiments.

### 3.2 Knowledge Embedding Model

The goal of our knowledge embedding model is to model inner-graph relationships, making entities more distinguishable. Here, we use TransE (Bordes et al., 2013), which is one of the most representative translation-based methods, as our knowledge embedding model. It is worth mentioning that other advanced KG learning methods can also be applied to our knowledge embedding model such as TransD (Ji et al., 2015), which is left for future work as our main idea is to joint cross-graph embeddings and knowledge embeddings for entity alignment.

**Objective $O_K$.** Formally, given a relational triplet $(e_h, r, e_t)$, TransE wants $\mathbf{e_h} + \mathbf{r} \simeq \mathbf{e_t}$. So it defines the score function $f(e_h, r, e_t) = ||\mathbf{e_h} + \mathbf{r} - \mathbf{e_t}||_2$ to measure the plausibility of $(e_h, r, e_t)$, where $||\cdot||_2$ means 2-norm. Following TransE, we utilize a margin-based ranking loss function as the training objective of the knowledge embedding model, defined as:

$$O_K = \sum_{(e_h,r,e_t)\in T} \sum_{(e_h^{'},r',e_t^{'})\in T'} [f(e_h, r, e_t) + \gamma_2 - f(e_h^{'}, r', e_t^{'})]_+ \quad (5)$$

where $[\cdot]_+ = max\{0, \cdot\}$ represents the maximum between 0 and the input, embeddings of entities $e_h$ and $e_t$ are from the attention enhanced embedding matrix $\mathbf{H}^{(L)}$, relation $r$ is from the relation matrix $\mathbf{R} \in \mathbb{R}^{|R|\times d}$ which needs to be learned, $T'$ stands for the negative sample triplet set of $T$, and $\gamma_2 > 0$ is a margin hyper-parameter separating positive and negative triplets. $T'$ is generated by corrupting $T$. For a triplet $(e_h, r, e_t) \in T$, we replace $e_h$ to generate a negative triplet $(e_h', r, e_t)$, where $e_h'$ also has the same relation type $r$ connect to other entities, and so does $e_t$. $K_1$ is the number of negative samples for each positive triplet. For example, we think *Beijing City* can be a replacer for *New York City* in (*New York City*, *country*, *United States*), since *Beijing City* also has the relation *country* but connected with a different tail entity *China*. If the number of replacers is less than $K_1$, we randomly sample replacers from all entities as what the uniform negative sampling method does.

### 3.3 Optimization and Inference

**Overall Objective $O$.** Here, we define the objective function of KECG corresponding to the above two parts:

$$O = O_C + O_K \tag{6}$$

where $O_C$ and $O_K$ denote the objective function of the cross-graph model and the knowledge embedding model, respectively. It is worth noting that we can set different weights for $O_C$ and $O_K$. But in order to treat both kinds of models equally, here, we set the same weight for them in experiments. We use AdaGrad (Duchi et al., 2011) to optimize the overall objective $O$.

**Inference.** When inferring, new equivalent entities are discovered based on the $L_2$ distances between entities in the joint embeddings. Equivalent entities should become close with each other, so do equivalent relation types, e.g., circled nodes, $r_1$ and $r_4$ in the output of Figure 2.

## 4 Experiments

We evaluate KECG on entity alignment with five publicly available datasets including three cross-lingual datasets and two large-scale datasets from different sources. We compare KECG with four state-of-the-art baselines and conduct further analysis of essential components of KECG.

### 4.1 Datasets

Following (Sun et al., 2017, 2018; Wang et al., 2018), we use the DBP15K and the DWY100K datasets for evaluation. **DBP15K** contains three cross-lingual datasets built from DBpedia, denoted by $\text{DBP15K}_{\text{ZH-EN}}$(Chinese to English), $\text{DBP15K}_{\text{JA-EN}}$(Japanese to English) and $\text{DBP15K}_{\text{FR-EN}}$(French to English). Each dataset contains 15,000 reference entity alignments. To test the adaptability of KECG to large-scale data, we evaluate on **DWY100K**. **DWY100K** contains two large-scale datasets extracted from DBpedia, Wikidata and YAGO3, denoted by $\text{DWY100K}_{\text{WD}}$(DBpedia to Wikidata) and $\text{DWY100K}_{\text{YG}}$(DBpedia to YAGO3). Each dataset has 100,000 reference entity alignments. The detailed information of each dataset is listed in Table 1.

| Datasets | | Entities | Relations | Rel. triplets |
|---|---|---|---|---|
| $\text{DBP15K}_{\text{ZH-EN}}$ | Chinese | 66,469 | 2,830 | 153,929 |
| | English | 98,125 | 2,317 | 237,674 |
| $\text{DBP15K}_{\text{JA-EN}}$ | Japanese | 65,744 | 2,043 | 164,373 |
| | English | 95,680 | 2,096 | 233,319 |
| $\text{DBP15K}_{\text{FR-EN}}$ | French | 66,858 | 1,379 | 192,191 |
| | English | 105,889 | 2,209 | 278,590 |
| $\text{DWY100K}_{\text{WD}}$ | DBpedia | 100,000 | 330 | 463,294 |
| | Wikidata | 100,000 | 220 | 448,774 |
| $\text{DWY100K}_{\text{YG}}$ | DBpedia | 100,000 | 302 | 428,952 |
| | YAGO3 | 100,000 | 31 | 502,563 |

Table 1: Statistics of DBP15K and DWY100K.

### 4.2 Experiment Settings

To investigate the ability of KECG, we compare it with the following state-of-the-art entity alignment methods including three KG-based models and one graph-based model.

- KG-based models: **MTransE** (Chen et al., 2017) is the basic KG-based method. **JAPE** (Sun et al., 2018) combines relation and attribute embeddings for entity alignment. **AlignEA** is the non-iterative version of BootEA (Sun et al., 2018) for entity alignment, which gets the best results.

- Graph-based model: **GCN-Align** (Wang et al., 2018) aligns cross-lingual entity via GCN.

More specifically, we use a 3-layer extended GAT as the encoder to process the composite KG $G$ to generate embeddings of its entities. We set the dimension of all layers to be the same $d$. The entity embedding matrix $\mathbf{X}$ and the relation matrix $\mathbf{R}$ are initialized randomly and updated during training. Besides, we separated two variants from KECG for ablation study, called KECG(w/o NNS) and KECG(w/o K).

- KECG(w/o NNS) replaces the nearest neighbor sampling (NNS) strategy with normal uniform negative sampling.

- KECG(w/o K) simply removes the knowledge embedding model, only considering the equivalent relation as what GCN-Align does.

For all the compared approaches, we randomly split 30% of the reference entity alignments as prior alignments (i.e., training data) and leave the remaining as testing data. We use two evaluation metrics in this task: (1) the mean reciprocal rank of all correct entities (MRR) and (2) the proportion of correct entities that rank no larger than N

| Methods | $\text{DBP15K}_{\text{ZH-EN}}$ | | | $\text{DBP15K}_{\text{JA-EN}}$ | | | $\text{DBP15K}_{\text{FR-EN}}$ | | | $\text{DWY100K}_{\text{WD}}$ | | | $\text{DWY100K}_{\text{YG}}$ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Hits@1 | Hits@10 | MRR | Hits@1 | Hits@10 | MRR | Hits@1 | Hits@10 | MRR | Hits@1 | Hits@10 | MRR | Hits@1 | Hits@10 | MRR |
| MTransE | 30.83 | 61.41 | 0.364 | 27.86 | 57.45 | 0.349 | 24.41 | 55.55 | 0.335 | 28.12 | 51.95 | 0.363 | 25.15 | 49.29 | 0.334 |
| JAPE | 41.18 | 74.46 | 0.490 | 36.25 | 68.50 | 0.476 | 32.39 | 66.68 | 0.430 | 31.84 | 58.88 | 0.411 | 23.57 | 48.41 | 0.320 |
| AlignEA | 47.18 | 79.19 | 0.581 | 44.76 | 78.89 | 0.563 | 48.12 | 82.43 | 0.599 | 56.55 | 82.70 | 0.655 | 63.29 | 84.76 | 0.707 |
| GCN-Align | 41.25 | 74.38 | 0.549* | 39.91 | 74.46 | 0.546* | 37.29 | 74.49 | 0.532* | 50.60* | 77.21* | 0.600* | 59.72* | 83.80* | 0.682* |
| KECG(w/o NNS) | 41.10 | 77.67 | 0.549 | 41.05 | 77.55 | 0.550 | 39.30 | 77.17 | 0.547 | 46.95 | 76.29 | 0.571 | 57.10 | 80.58 | 0.653 |
| KECG(w/o K) | 43.04 | 79.12 | 0.551 | 44.61 | 80.70 | 0.567 | 43.22 | 81.45 | 0.559 | 59.77 | 87.99 | 0.695 | **73.83** | **93.04** | **0.806** |
| KECG | **47.77** | **83.50** | **0.598** | **48.97** | **84.40** | **0.610** | **48.64** | **85.06** | **0.610** | **63.23** | **89.97** | **0.726** | 72.76 | 91.51 | 0.795 |

"*" denotes the unreported metrics in their papers. We reproduced the results using their source code.

Table 2: Results comparison on entity alignment(Results of Hits@1 and Hits@10 are percentage values).

(Hits@N). Higher values of MRR and Hits@N indicate a better entity alignment model.

For parameter settings, the dimension of the hidden units in each GAT layer is set to be the same $d = 128$ as the entity and relation embeddings. Learning rate $\lambda$ for AdaGrad is chosen among $\{0.001, 0.005, 0.01, 0.05, 0.1\}$. Margin $\gamma_1$ and $\gamma_2$ are chosen among $\{1.0, 2.0, 3.0\}$. The number of negative samples for cross-graph model and knowledge model are $K_1 = 25$ and $K_2 = 2$, respectively, and the negative samples will be updated every 10 epochs. The total number of training rounds is 1000. The optimal configuration of our models for the entity alignment task is: $\lambda = 0.005$, $\gamma_1 = 3.0$, $\gamma_2 = 3.0$.

### 4.3 Results

Table 2 lists the results of entity alignment on DBP15K and DWY100K. In general, KECG is significantly more effective than MTransE, JAPE, and GCN-Align on all datasets, and slightly outperform AlignEA on small-scale datasets but greatly on large-scale datasets. Among all the baselines, AlignEA is the strongest one, which results from it using two margins to control the scores of triplets, making the KG structure modeling based on triplets more accurate than others. We conclude the detailed results in three aspects:

- **For different model groups.** Compared with the KG-based models, KECG performs much better than MTransE and JAPE. On $\text{DBP15K}_{\text{ZH-EN}}$, KECG and AlignEA get very close results regarding Hits@1 with a gap of 0.59%, but as for Hits@10 and MRR, the differences become 4.31% and 0.017, respectively. The reason is that, though some entities are not accurately aligned, their neighbors, or neighbors of neighbors, may already be aligned. And KECG propagates these features to unaligned entities, making the positions of the unaligned entities in the vector space very close to the corresponding entities. It reflects the importance of the global structural information of KGs. Compared with the graph-based model, KECG significantly outperforms GCN-Align regarding all metrics. It shows that involving the local relationships between entities can make entities more distinguishable, making the alignment more accurate.

- **For different data scales.** It can be seen that KECG has good adaptability on different data scales. The average gains of KECG regarding Hits@1 are 8%-41.3% on large-scale datasets and 1.7%-20.7% on small-scale datasets. With large-scale KGs providing richer relational and structural information, KECG can encode entities with more semantics. However, we found that, on $\text{DWY100K}_{\text{YG}}$, KECG's variant KECG(w/o K) works better than itself. This will be further discussed in Section 4.4.

- **For different language pairs.** KECG performs best on all language pairs. It improves the MRR score to 0.61 on $\text{DBP15K}_{\text{JA-EN}}$ , which is a nearly 8% improvement compared with the best baseline. To the best of our knowledge, the imbalanced resources of non-English KGs with English KG result in the heterogeneity of cross-lingual KGs, making it challenging to align them. KG-based models, such as MTransE, cannot learn good representations to model the structures of non-English KGs, as restraints for entities are relatively insufficient. Methods based on GCNs, such as GCN-Align, are sensitive to structural differences, and find it hard to narrow the differences for alignment. KECG makes use of the extended GAT to alleviate the neg-

ative impacts of heterogeneous KGs, which successfully highlights the entities that are important for alignment in the graph. Because English and French share many similarities, both KECG and baselines get satisfactory results on $\text{DBP15K}_{\text{FR-EN}}$.

## 4.4 Analysis

### 4.4.1 Effectiveness of Knowledge Embedding Model

For better comparison, we draw two detailed Hits@N bar plots of GCN-Align, AlignEA, KECG(w/o K) and KECG on $\text{DBP15K}_{\text{ZH-EN}}$ and $\text{DWY100K}_{\text{YG}}$, as shown in Figure 3. In Figure 3(a), KECG(w/o K) gets better results than GCN-Align, which reflects the power of the graph attention mechanism. However, KECG(w/o K) cannot exceed AlignEA, showing that only considering structural information is not enough for entity alignment. After adding the knowledge embedding model, KECG achieves the highest Hits@{1,10,50}. The effectiveness of knowledge embedding model lies in two parts. Firstly, it applies constraints to entities by involving representation of relation types. Entity alignment leads to some relationship alignment, and entities having similar relations become close. Secondly, the knowledge embedding model is beneficial for the graph attention mechanism to learn better attention values.

In Figure 3(b), KECG still outperforms GCN-Align and AlignEA, but KECG(w/o K) exceeds KECG regarding three metrics, which reflects the weakness of the current knowledge embedding model. According to our statistics, the 1-to-N, N-to-1, and N-to-N relations account for 79.58% of the total in $\text{DWY100K}_{\text{YG}}$, while 60.05% in $\text{DBP15K}_{\text{ZH-EN}}$. The introduced knowledge embedding model is based on TransE, lacking the ability of modeling complex relationships (e.g., 1-to-N, N-to-1, and N-to-N relations). When there are too many complex relationships in the datasets, it brings more decrement rather than improvement to the results. To ease this situation, we can use a stronger representation model for complex relationships, such as TransR (Lin et al., 2015). We leave this for future work.

### 4.4.2 Effectiveness of Nearest Neighbor Sampling

In Table 2, it can be seen that KECG(w/o NNS) outperforms many baselines and its Hits@1 on

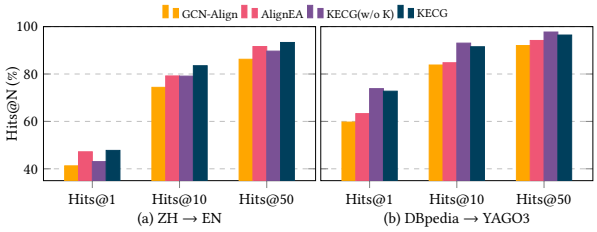


Figure 3: Detailed Hits@N for investigating the effectiveness of Knowledge Embedding Model.

$\text{DBP15K}_{\text{FR-EN}}$ reaches 39.30%, which is higher than another graph-based method GCN-Align by 2.01%. But the Hits@1 of KECG(w/o NNS) is lower than GCN-Align's by approximately 3% on DWY100K. It is because DWY100K has a larger number of triplets, while the number of relations is relatively small. Involving the representation of complex relationships will introduce noise and affect the performance of the cross-graph model. After employing the nearest neighbor sampling strategy in cross-graph model, KECG considerably improves the results of KECG(w/o NNS), e.g., Hits@1 on $\text{DWY100K}_{\text{WD}}$ is raised from 46.95% to 63.23%. The excellent performance of nearest neighbor sampling owing to its accurately discriminating the target entity from other close entities, especially on large-scale datasets.

### 4.4.3 Sensitivity to Proportion of Prior Alignment

To investigate whether KECG is sensitive to the proportion of pre-aligned entities, we vary the proportion from 10% to 50% with step 10% for training, and use the rest of pre-aligned entities for testing. Intuitively, more pre-aligned entities should provide more information to discover the potential pairs of equivalent entities.

Figure 4 illustrates the changes in Hits@1 and Hits@10 of KECG, AlignEA and GCN-Align on all datasets. As expected, the results of all methods become better with the increase in proportion. Given half of the prior alignment like 50%, all three methods get close and satisfactory Hits@1. When given a very small proportion of prior alignment like 10%, the results of AlignEA and GCN-Align decrease enormously, but our KECG still achieves outstanding results. Therefore, KECG works well on entity alignment even with a limited number of prior alignment.

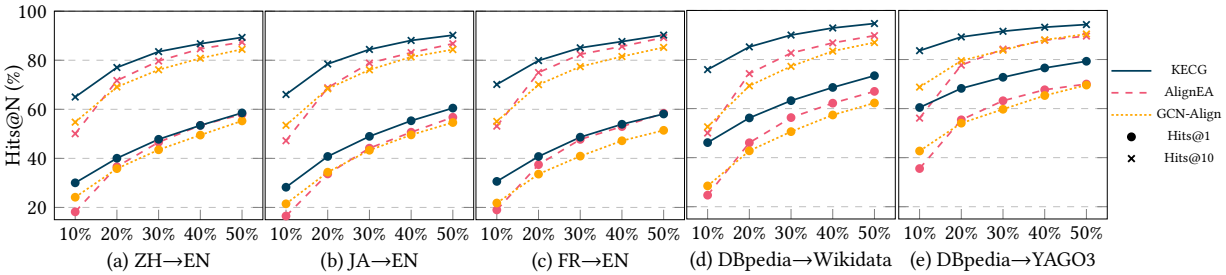


Figure 4: Hits@N of KECG, AlignEA and GCN-Align on entity alignment w.r.t. proportion of prior alignment.(The x-axis represents the proportion of prior alignment.)

## 5 Related Work

### 5.1 KG Embedding

Various KG embedding methods have shown effectiveness in modeling the semantic information of KGs. TransE (Bordes et al., 2013) is a milestone in learning embeddings for KGs, which interprets a relation as the translation from its head entity to its tail entity. Then it motivated several enhanced methods like TransR (Lin et al., 2015). In addition to them, non-translational methods also achieve satisfactory performances, such as RESCAL (Nickel et al., 2011), ConvE (Dettmers et al., 2018) and RotatE (Sun et al., 2019). Meanwhile, external information in KGs is fused to improve embedding (Wang and Li, 2016). More detailed KG embedding methods are summarized in (Wang et al., 2017).

### 5.2 Entity Alignment

Early human efforts, such as crowdsourcing and hand-crafted features (Lehmann et al., 2015; Mahdisoltani et al., 2013), are utilized to address the entity alignment problem. Though reaching high precisions, they are usually time-consuming and costly. Many automated approaches leverage the extra resources, such as OWL properties and Wikipedia links (Hu et al., 2011; Wang et al., 2013) to align entities, but such extra information is not generally available for all KGs.

Recently, KG embedding-based approaches become the most popular solution for entity alignment. JE (Hao et al., 2016) is an early work to jointly embed different KGs into a uniform vector space to align entities in KGs. MTransE (Chen et al., 2017) learns transitions to translate each entity embedding vector to its counterparts in another space. JAPE (Sun et al., 2017) jointly trains the attribute embedding and structure embedding by a skip-gram model and a translational model, respectively, to align entities. GCN-Align (Wang et al., 2018) is a new graph-based model aligning KGs via GCN. It takes advantage of GCN to propagate information from neighbors, and align entity embeddings enhanced by structural knowledge. However, GCN-Align only considers the equivalent relations between entities in model training, neglecting the use of abundant relationships in KGs to precisely distinguish entities with similar neighbors. At the same time, iterative alignment becomes a new fashion to improve the results of entity alignment. IPTransE (Zhu et al., 2017) and BootEA (Sun et al., 2018) integrate knowledge among different KGs by enlarging the training data (prior alignments) in a bootstrapping way. KDCoE (Chen et al., 2018) iteratively co-trains multilingual KG embeddings and fuses them with entity description information for alignment. The aforementioned iterative methods improve the results mainly by increasing the number of pre-align entity pairs while training, and such a strategy can be a general enhancement for most alignment approaches (Sun et al., 2018). A better non-iterative method should achieve better results through bootstrap. Therefore, we are more concerned with the best result that a non-iterative method can achieve.

## 6 Conclusions

In this paper, we propose a semi-supervised entity alignment method KECG that combines the knowledge embedding model and graph-based model. We utilize an extended GAT to encode heterogeneous KGs and perform entity alignment by propagating alignment information over the entire graphs. Meanwhile, the KG learning is involved to model different relation types, making entities more distinguishable. Experimental results show that KECG significantly outperforms four state-of-the-art baselines.

For future work, we will extend the knowledge

embedding model of KECG to other KG representation learning methods, such as TransD (Ji et al., 2015), to gain a stronger ability of modeling relationships. Besides, iteratively discovering new entity alignments based on the framework of KECG is another interesting direction.

## Acknowledgments

This work is supported by National Key Research and Development Program of China (2017YFB1002101), NSFC key projects (U1736204, 6153301), a research fund by Alibaba, and THUNUS NExT Co-Lab. NExT++ research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its IRC@SG Funding Initiative.

## References

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NeurIPS*.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2014. Spectral networks and locally connected networks on graphs. In *ICLR*.

Yixin Cao, Lei Hou, Juanzi Li, and Zhiyuan Liu. 2018. Neural collective entity linking. In *COLING*.

Muhao Chen, Yingtao Tian, Kai-Wei Chang, Steven Skiena, and Carlo Zaniolo. 2018. Co-training embeddings of knowledge graphs and entity descriptions for cross-lingual entity alignment. In *IJCAI*.

Muhao Chen, Yingtao Tian, Mohan Yang, and Carlo Zaniolo. 2017. Multilingual knowledge graph embeddings for cross-lingual knowledge alignment. In *IJCAI*.

Wanyun Cui, Yanghua Xiao, Haixun Wang, Yangqiu Song, Seung-won Hwang, and Wei Wang. 2017. Kbqa: learning question answering over qa corpora and knowledge bases. *PVLDB*.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *NeurIPS*.

Tim Dettmers, Minervini Pasquale, Stenetorp Pontus, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *AAAI*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*.

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*.

Xu Han, Zhiyuan Liu, and Maosong Sun. 2018. Neural knowledge acquisition via mutual attention between knowledge graph and text. In *AAAI*.

Yanchao Hao, Yuanzhe Zhang, Shizhu He, Kang Liu, and Jun Zhao. 2016. A joint embedding method for entity alignment of knowledge bases. In *CCKS*.

Wei Hu, Jianfeng Chen, and Yuzhong Qu. 2011. A self-training approach for resolving object coreference on the semantic web. In *WWW*.

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *ACL*.

Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

Bhushan Kotnis and Vivi Nastase. 2017. Analysis of the impact of negative sampling on link prediction in knowledge graphs. *arXiv preprint arXiv:1708.06816*.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*.

Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *ICML*.

Farzaneh Mahdisoltani, Joanna Biega, and Fabian M Suchanek. 2013. Yago3: A knowledge base from multilingual wikipedias. In *CIDR*.

Roberto Navigli and Simone Paolo Ponzetto. 2012. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *ICML*.

Thomas Rebele, Fabian Suchanek, Johannes Hoffart, Joanna Biega, Erdal Kuzey, and Gerhard Weikum. 2016. Yago: A multilingual knowledge base from wikipedia, wordnet, and geonames. In *ISWC*.

Zequn Sun, Wei Hu, and Chengkai Li. 2017. Cross-lingual entity alignment via joint attribute-preserving embedding. In *ISWC*.

Zequn Sun, Wei Hu, Qingheng Zhang, and Yuzhong Qu. 2018. Bootstrapping entity alignment with knowledge graph embedding. In *IJCAI*.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. In *ICLR*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *ICLR*.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *TKDE*.

Zhichun Wang, Juanzi Li, and Jie Tang. 2013. Boosting cross-lingual knowledge linking via concept annotation. In *IJCAI*.

Zhichun Wang, Qingsong Lv, Xiaohan Lan, and Yu Zhang. 2018. Cross-lingual knowledge graph alignment via graph convolutional networks. In *EMNLP*.

Zhigang Wang and Juanzi Li. 2016. Text-enhanced representation learning for knowledge graph. In *IJCAI*.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *ICLR*.

Hao Zhu, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2017. Iterative entity alignment via joint knowledge embeddings. In *IJCAI*.

# Designing and Interpreting Probes with Control Tasks

**John Hewitt**
Stanford University
johnhew@stanford.edu

**Percy Liang**
Stanford University
pliang@cs.stanford.edu

## Abstract

Probes, supervised models trained to predict properties (like parts-of-speech) from representations (like ELMo), have achieved high accuracy on a range of linguistic tasks. But does this mean that the representations encode linguistic structure or just that the probe has learned the linguistic task? In this paper, we propose *control tasks*, which associate word types with random outputs, to complement linguistic tasks. By construction, these tasks can only be learned by the probe itself. So a good probe, (one that reflects the representation), should be *selective*, achieving high linguistic task accuracy and low control task accuracy. The selectivity of a probe puts linguistic task accuracy in context with the probe's capacity to memorize from word types. We construct control tasks for English part-of-speech tagging and dependency edge prediction, and show that popular probes on ELMo representations are not selective. We also find that dropout, commonly used to control probe complexity, is ineffective for improving selectivity of MLPs, but that other forms of regularization are effective. Finally, we find that while probes on the first layer of ELMo yield slightly better part-of-speech tagging accuracy than the second, probes on the second layer are substantially more selective, which raises the question of which layer better represents parts-of-speech.

## 1 Introduction

As large-scale unsupervised representations such as BERT and ELMo improve downstream performance on a wide range of natural language tasks (Devlin et al., 2019; Peters et al., 2018a; Radford et al., 2019), what these models learn about language remains an open scientific question. An emerging body of work investigates this question through *probes*, supervised models trained to predict a property (like parts-of-speech) from a constrained view of the representation. Probes trained on various representations have obtained high accuracy on tasks requiring part-of-speech and morphological information (Belinkov et al., 2017), syntactic and semantic information (Peters et al., 2018b; Tenney et al., 2019), among other properties (Conneau et al., 2018), providing evidence that deep representations trained on large datasets are predictive of a broad range of linguistic properties.

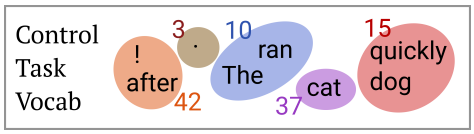


| Sentence 1 | The | cat | ran | quickly | . |
|---|---|---|---|---|---|
| **Part-of-speech** | DT | NN | VBD | RB | . |
| **Control task** | 10 | 37 | 10 | 15 | 3 |
| Sentence 2 | The | dog | ran | after | ! |
| **Part-of-speech** | DT | NN | VBD | IN | . |
| **Control task** | 10 | 15 | 10 | 42 | 42 |

Figure 1: Our control tasks define random behavior (like a random output, top) for each word type in the vocabulary. Each word token is assigned its type's output, regardless of context (middle, bottom.) Control tasks have the same input and output space as a linguistic task (e.g., parts-of-speech) but can only be learned if the probe memorizes the mapping.

But when a probe achieves high accuracy on a linguistic task using a representation, can we conclude that the representation encodes linguistic structure, or has the probe just learned the task? Probing papers tend to acknowledge this uncertainty, putting accuracies in context using random representation baselines (Zhang and Bowman, 2018) and careful task design (Hupkes et al., 2018). Even so, as long as a representation is a lossless encoding, a sufficiently expressive probe with enough training data can learn *any* task on top of it.

In this paper, we propose *control tasks*, which associate word types with random outputs, to give intuition for the expressivity of probe families and



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2733–2743,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

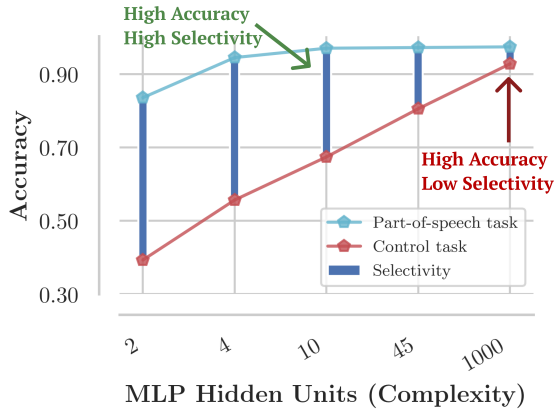


Figure 2: Selectivity is defined as the difference between linguistic task accuracy and control task accuracy, and can vary widely, as shown, across probes which achieve similar linguistic task accuracies. These results taken from § 3.5.

provide insight into how representation and probe interact to achieve high task accuracy.

Control tasks are based on the intuition that the more a probe is able to make task output decisions independently of the linguistic properties of a representation, the less its accuracy on a linguistic task necessarily reflects the properties of the representation. Thus, a good probe (one that provides insights into the linguistic properties of a representation) should be what we call *selective*, achieving high linguistic task accuracy and low control task accuracy (see Figure 2).

We show that selectivity can be a guide in designing probes and interpreting probing results, complementary to random representation baselines; as of now, there is little consensus on how to design probes. Early probing papers used linear functions (Shi et al., 2016; Ettinger et al., 2016; Alain and Bengio, 2016), which are still used (Bisazza and Tump, 2018; Liu et al., 2019), but multi-layer perceptron (MLP) probes are at least as popular (Belinkov et al., 2017; Conneau et al., 2018; Adi et al., 2017; Tenney et al., 2019; Ettinger et al., 2018). Arguments have been made for "simple" probes, e.g., that we want to find easily accessible information in a representation (Liu et al., 2019; Alain and Bengio, 2016). As a counterpoint though, "complex" MLP probes have also been suggested since useful properties might be encoded non-linearly (Conneau et al., 2018), and they tend to report similar trends to simpler probes anyway (Belinkov et al., 2017; Qian et al., 2016).

We define control tasks corresponding to English part-of-speech tagging and dependency edge prediction, and use ELMo representations to conduct a broad study of probe families, hyperparameters, and regularization methods, evaluating both linguistic task accuracy and selectivity. We propose that selectivity be used for building intuition about the expressivity of probes and the properties of models, putting probing accuracies into richer context. We find that:

1. With popular hyperparameter settings, MLP probes achieve very low selectivity, suggesting caution in interpreting how their results reflect properties of representations. For example, on part-of-speech tagging, 97.3 accuracy is achieved, compared to 92.8 control task accuracy, resulting in 4.5 selectivity.

2. Linear and bilinear probes achieve relatively high selectivity across a range of hyperparameters. For example, a linear probe on part-of-speech tagging achieves a similar 97.2 accuracy, and 71.2 control task accuracy, for 26.0 selectivity. This suggests that the small accuracy gain of the MLP may be explained by increased probe expressivity.

3. The most popular method for controlling probe complexity, dropout, does not consistently lead to selective MLP probes. However, control of MLP complexity through unintuitively small (10-dimensional) hidden states, as well as small training sample sizes and weight decay, lead to higher selectivity and similar linguistic task accuracy.

Finally, we ask, *can we meaningfully compare the linguistic properties of layers of a model using only linguistic task accuracy*? We raise a potential problem with this approach: it fails to take into account differences in ease of memorization across layers. In particular, we find that while linear and MLP probes on the first layer of ELMo (ELMo1) achieve slightly higher part-of-speech accuracy than those on the second layer (ELMo2), (97.2 compared to 96.6, for a loss of 0.6 ), the same probes achieve much greater selectivity on ELMo2 (31.4 compared to 26.0, for a gain of 5.4). Thus, the difference in selectivity in favor of ELMo2 is much greater than the commonly known (Peters et al., 2018a; Liu et al., 2019) difference in linguistic task accuracy in favor of ELMo1; the difference in accuracy may be explained by probes more easily accessing word identity features in ELMo1.

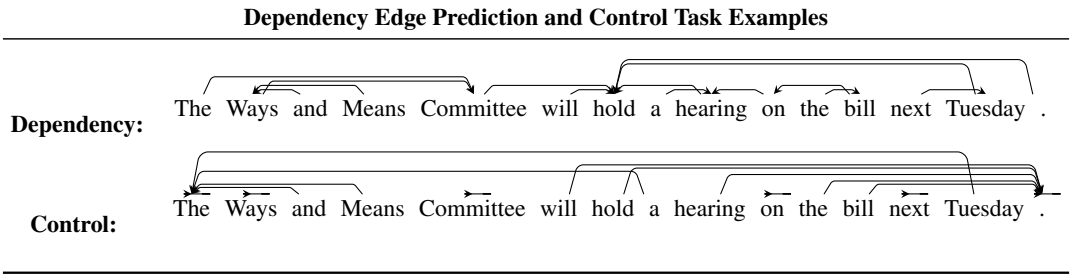


Figure 3: Example dependency tree from the development set of the Penn Treebank with dependents pointing at heads, and the structure resulting from our dependency edge prediction control task on the same sentence.

## 2 Control Tasks

In this section, we describe how to construct control tasks. At a high level, control tasks have:

**structure:** The output for a word token is a deterministic function of the word type[1].

**randomness:** The output for each word type is sampled independently at random.

We start with some notation; denote as $1:T$ the sequence of integers $\{1, ..., T\}$. Let $V$ be the vocabulary containing all word types in a corpus. A sentence of length $T$ is $\mathbf{x}_{1:T}$, where each $x_i \in V$, and the word representations of the model being probed are $\mathbf{h}_{1:T}$, where $h_i \in \mathbb{R}^d$. A *task* is a function that maps a sentence to a single output per word, $f(\mathbf{x}_{1:T}) = \mathbf{y}_{1:T}$, where each output is from a finite set of outputs: $y_i \in \mathcal{Y}$. Each *control task* is defined in reference to a *linguistic task*, and the two share $\mathcal{Y}$. We'll now use part-of-speech tagging and dependency edge prediction as examples to describe the construction of control tasks.

### 2.1 Part-of-speech tagging control task

In part-of-speech tagging, the set $\mathcal{Y}$ is the tagset, $1:45$ (corresponding to NN, NNS, VB,...). To construct a control task, we independently sample a *control behavior* $C(v)$ for each $v \in V$. The control behavior specifies how to define $y_i \in \mathcal{Y}$ for a word token $x_i$ with word type $v$. For part-of-speech tagging, each control behavior directly specifies the output $y_i$ for $x_i$ as an integer from $1:45$, so we sample from 45 behaviors[2]. The part-of-speech control task is the function that maps each token $x_i$ to the label specified by the behavior $C(x_i)$:

$$f_{\text{control}}(\mathbf{x}_{1:T}) = f(C(x_1), C(x_2), ...C(x_T)). \quad (1)$$

This task is visualized in Figure 1.

[1]Equivalently, word identity.

[2]The exact distribution from which we sample isn't crucial, but for part-of-speech tagging, we sample from the empirical token distribution of part-of-speech tagging, so the marginal probability of each label is similar.

### 2.2 Dependency edge prediction control task

The dependency edge prediction task is the function $f_{\text{DEP}}(\mathbf{x}_{1:T}) = \mathbf{y}_{1:T}$ where $y_i$ is the index of the parent of $x_i$ in the dependency tree on the sentence $\mathbf{x}_{1:T}$. Thus, the output space $\mathcal{Y} = 1:T$ depends on the length of the sentence, $T$. To accommodate this in our control task, we define the control behaviors $C(v)$ in a length-independent way that still fully specifies $y_i$. The possible behaviors $C(v)$ are as follows:

**attach to self:** Always attach tokens of this type to themselves. That is, $y_i = i$.

**attach to first:** Always attach tokens of this type to the first token. That is, $y_i = 1$.

**attach to last:** Always attach tokens of this type to the last word in the sentence. That is, $y_i = T$.

We sample uniformly from the three. Given these behaviors, the control task is defined as before by Eqn 1. This task is visualized in Figure 3.

While very similar to dependency parsing, dependency edge prediction differs in two ways. The output is not constrained to be a tree for evaluation; each prediction is evaluated independently. So, while our control tasks do not define trees, the two tasks' output spaces are still the same. Second, in dependency edge prediction, the root of the sentence is omitted from evaluation; no sentence-external `ROOT` token is posited for evaluation.

### 2.3 Properties of control tasks

To summarize, a control task is defined for a single linguistic task, and shares the linguistic task's output space $\mathcal{Y}$. To construct a control task, a control behavior $C(v)$ is sampled independently at random for each word type $v \in V$. The control task is a function mapping $\mathbf{x}_{1:T}$ to a sequence of outputs $\mathbf{y}_{1:T}$ which is fully specified by the sequence of behaviors, $[C(x_1), ..., C(x_T)]$.

From this construction, we note that the ceiling on performance is the fraction of tokens in the evaluation set whose types occur in the training set (plus chance accuracy on all other tokens.) Further, $C(v)$ must be memorized independently for each word type, and a probe taking vectors $h_{1:T}$ as input must identify for each $h_i$ its corresponding $x_i$, and output the element of $\mathcal{Y}$ specified by $C(x_i)$.

## 3 Experiments on Probe Selectivity

In this section, we conduct a broad study of probe families (e.g, linear, MLP) and hyperparameter choices (weight matrix rank/hidden state size, amount of regularization) on a single representation (ELMo1) to determine (1) what probe choices exhibit high linguistic task accuracy *and* high selectivity (and whether this holds for a range of hyperparameters), and (2) whether each probe family can be made selective through hyperparameter choices without substantially sacrificing linguistic task accuracy.

### 3.1 Probe families

We experiment with three types of probes per task.

For part-of-speech tagging, we experiment with linear, MLP-1, and MLP-2 probes. The linear probe is a multiclass model mapping $h_i$ to $y_i \sim \text{softmax}(Ah_i + b)$. The MLP-1 probe is a multilayer perceptron with one hidden layer and ReLU nonlinearity defined as:

$$y_i \sim \text{softmax}(W_2\ g(W_1 h_i)). \quad (2)$$

And the MLP-2 probe is defined as:

$$y_i \sim \text{softmax}(W_3\ g(W_2\ g(W_1 h_i))). \quad (3)$$

where $g$ is the ReLU function, and bias terms are omitted from all affine transformations for brevity.

For dependency edge prediction, we experiment with bilinear, MLP-1, and MLP-2 probes. These probes take as input the entire sequence $\mathbf{h}_{1:T}$ as well as the vector $h_i$ of a given state to produce $y_i$; the softmax operates over the sequence to construct a distribution over the $T$ classes. Formally, the bilinear model is defined as $y_i \sim \text{softmax}(\mathbf{h}_{1:T}^\top A h_i + b)$. The MLP-1 probe is defined as follows:

$$y_i \sim \text{softmax}(W_2\ g(W_1[\mathbf{h}_{1:T}; h_i])). \quad (4)$$

Note here that $h_i$ broadcasts to $\mathbb{R}^{T\times d}$,while $W_1 \in \mathbb{R}^{\ell\times d}$, and $W_2 \in \mathbb{R}^{1\times \ell}$ broadcast as well. That is, each $[h_j; h_i]$ pair is mapped to a single scalar independently of all others, leading to $T$ logits used as input to the softmax. Similarly, the MLP-2 model is defined as follows:

$$y_i \sim \text{softmax}(W_3\ g(W_2\ g(W_1[\mathbf{h}_{1:T}; h_i]))). \quad (5)$$

### 3.2 Complexity control

It is well-known that probes should not be too complex (Liu et al., 2019; Alain and Bengio, 2016); this is the motivation behind constraining the input to the probe to be a single vector or pair of vectors. However, there has been no systematic investigation of probe complexity. We study what complexity control is necessary to achieve selectivity. As we will see, the typical practice of regularizing to reduce the generalization gap (difference between training and test task accuracy) is insufficient if one is interested in selectivity.

**Rank/hidden dimensionality constraint.** For our linear and bilinear probes, we constrain the rank of weight matrices through an LR decomposition. We let $A \in \mathbb{R}^{k\times d}$, where $k$ is the output space (45 for part-of-speech tagging; 1 for dependency head prediction). To constrain $A$ to rank $\ell$, we factor $A = LR$, where $L \in \mathbb{R}^{k\times\ell}$ and $R \in \mathbb{R}^{\ell\times|V|}$, and optimize over $L$ and $R$. For MLP models, we let the hidden state size be equal to $\ell$.[3]

From the default value of rank-1000 and 1000-dimensional hidden states, we let $\ell$ take on the values $\{2, 4, 10, 45\}$ for part-of-speech, and $\{5, 10, 50, 100\}$ for dependency edge prediction.[4]

**Dropout.** We apply dropout (Srivastava et al., 2014) with probability $p$ to the input for linear and bilinear probes, and to the input and the output of each hidden layer for MLP probes. From the default value of 0, we let $p$ range over $\{0.2, 0.4, 0.6, 0.8\}$.

**Number of training examples.** We artificially constrain the number of sentences the probe is trained on, with the intuition that general rules can be learned more sample-efficiently than memorization. Zhang and Bowman (2018) showed this to be an effective distinguishing factor between trained representations and random representation controls. From the default of 39832 (the number of training

[3] One could constrain the matrices of the MLP to be rank $\ell$ without making the hidden state smaller, but one must choose a hidden state size anyway, so we believed a study changing the hidden state size would be most informative.

[4] Note that for linear models, the rank is constrained by $k$ regardless, since $A \in \mathbb{R}^{k\times d}$.

| Probe | PoS | Ctl | Select. | Dep | Ctl | Select. |
|---|---|---|---|---|---|---|
| Probes with Default Hyperparameters | | | | | | |
| Linear | 97.2 | 71.2 | 26.0 | - | - | - |
| Bilinear | - | - | - | 89.0 | 82.4 | 6.6 |
| MLP-1 | 97.3 | 92.8 | 4.5 | 92.3 | 93.0 | -0.7 |
| MLP-2 | 97.3 | 93.2 | 4.2 | 93.9 | 92.0 | 1.9 |
| Probes with 0.4 Dropout | | | | | | |
| Linear | 97.1 | 67.3 | 29.8 | - | - | - |
| Bilinear | - | - | - | 90.4 | 73.7 | 16.7 |
| MLP-1 | 97.5 | 93.4 | 4.1 | 93.8 | 93.1 | 0.7 |
| MLP-2 | 97.4 | 94.1 | 3.4 | 94.7 | 93.5 | 1.3 |
| Probes Designed with Control Tasks | | | | | | |
| Linear | 97.0 | 64.0 | 33.0 | - | - | - |
| Bilinear | - | - | - | 91.0 | 83.1 | 7.9 |
| MLP-1 | 97.2 | 80.6 | 16.6 | 90.5 | 84.3 | 6.2 |
| MLP-2 | 97.2 | 81.7 | 15.4 | 92.8 | 89.8 | 3.0 |

Table 1: Probe accuracies on linguistic tasks and control tasks. Default hyperparameters correspond to a hidden state of dimensionality 1000 and no dropout. Under Probes Designed with Control Tasks, we used selectivity to hand-pick a hyperparameter setting for each probe. In particular, part-of-speech probes designed with control tasks all use rank-10 weight matrices (10-dimensional hidden state) and no other changes. Dependency edge prediction probes designed with control tasks had, for the bilinear model, weight decay of 0.01, for MLP-1, weight decay of 0.1, for MLP-2, a rank-50 weight matrix.

examples in the dataset), we train on $\{4000, 400\}$ examples, corresponding to roughly 100%, 10%, and 1% of the total data, as suggested by Zhang and Bowman (2018).

**$L_2$ regularization.** We apply weight decay to the probe parameters. From the default of 0, we let the weight decay constant take on the values $\{0.01, 0.1, 1.0, 10.0\}$, unnormalized by batch size.

**Early stopping.** All of our probing models are trained with Adam (Kingma and Ba, 2014). By default, we anneal the learning rate by a factor of 0.5 each time an epoch does not lead to a new minimum loss on the development set, and stop training when 4 such epochs occur in a row. However, in early stopping, we explicitly halt training at a fixed number of gradient steps. From the default of 100000 (approximately 40 epochs), we let this maximum take on the values $\{50000, 25000, 12500, 6000, 3000, 1500\}$.

### 3.3 Dataset

We use the Penn Treebank (PTB) dataset (Marcus et al., 1993) with the traditional parsing training/development/testing splits[5] without preprocessing. We report accuracies on the development set. We convert the PTB constituency trees to the Stanford Dependencies formalism (de Marneffe et al., 2006) for our dependency edge prediction task.

[5]As given by the code of Qi and Manning (2017) at https://github.com/qipeng/arc-swift.

### 3.4 Representation

We use the 5.5 billion-word pre-trained ELMo representations (Peters et al., 2018a). Since the output of the first BiLSTM layer was recently shown to be the most transferrable on a wide variety of tasks, including part-of-speech and syntax (Liu et al., 2019), we focus on analyzing that layer, which we denote ELMo1.

### 3.5 Results

**Selectivity of default hyperparameters.** Our results with linear, bilinear, and MLP probes with "default" hyperparameters, as specified in § 3.2, are found in Table 1 (top). We find that linear probes achieve similar part-of-speech accuracies to MLPs (97.2 compared to 97.3) with substantially higher selectivity (26.0 vs 4.50). In dependency edge prediction, we find a definite gap between bilinear probe accuracy (89.0) and MLP-1 accuracy (92.3). However, the bilinear probe achieves 16.7 selectivity, compared to $-0.7$ by MLP-1 and 1.3 by MLP-2. Thus, with no regularization, modest gains in linguistic task accuracy through MLP probes over linear/bilinear probes are tempered by losses in selectivity. Bilinear and linear probes themselves show a significant capacity for memorization.

Does adding moderate regularization through dropout (e.g., $p = 0.4$) consistently lead to selectivity? Surprisingly, as shown in Table 1 (middle), the opposite is true for some MLP probes, where selectivity actually decreases (e.g., $4.2 \rightarrow 3.4$ for MLP-2). In one case, the MLP-1 probe on dependency edge prediction, dropout increases selectivity ($-0.7 \rightarrow 0.7$) but for no others.

**How hard is it to find selective probes?** We tried 6 methods for controlling probe complexity, and all worked except dropout and early stopping, though never for a broad range of hyperparameters. For each complexity control method except dropout and early stopping, we find hyperparameters that lead to high linguistic task accuracy and high selectivity. Our results are summarized in Figure 4.

We find that constraining the **hidden state dimensionality** of MLPs is an effective way to encourage selectivity at little cost to linguistic task

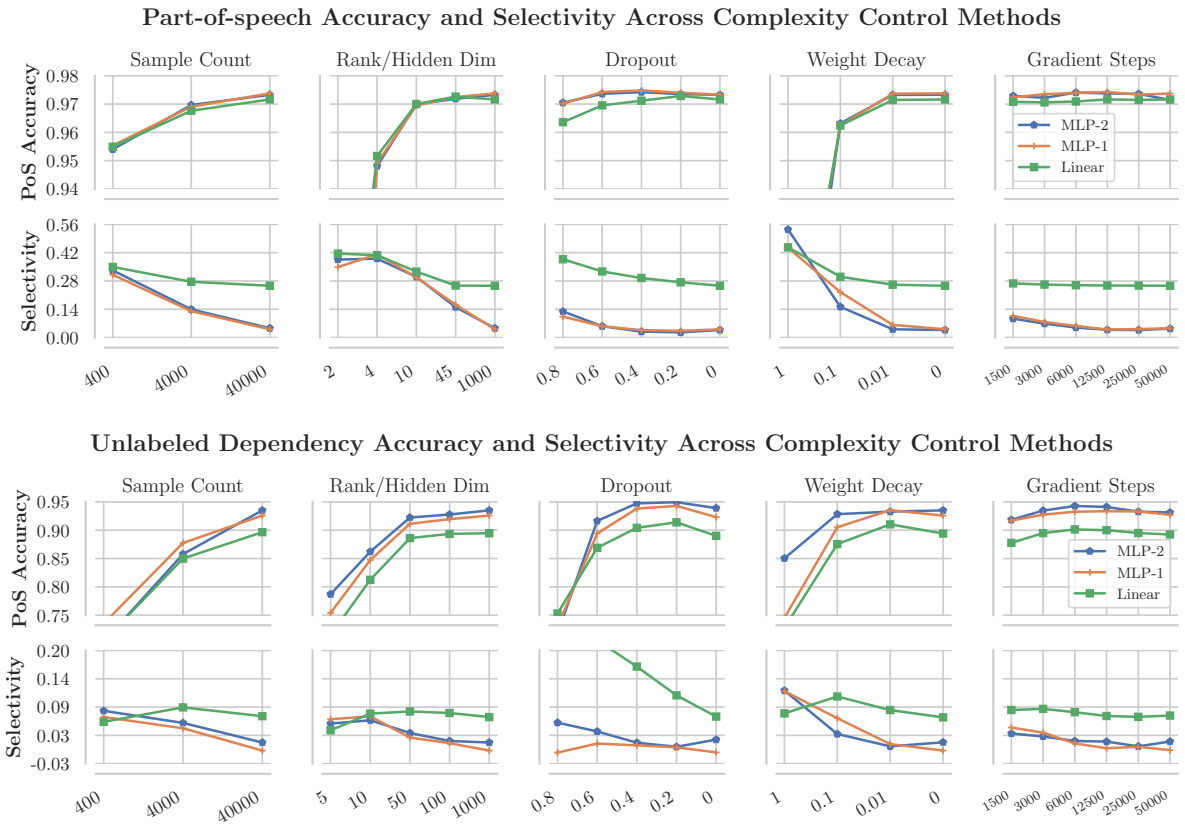


Figure 4: Linguistic task accuracies and selectivities for the 5 complexity control methods. All methods except dropout and early stopping are shown to improve selectivity without a large impact on linguistic task accuracy. All methods for the same task share a common y-axis, and use their own categorical x-axis. All x-axes are ordered from most severe constraints on complexity (left) to most laissez-faire (right).

accuracy. MLP hidden state sizes of 10 and 50, for part-of-speech tagging and dependency head prediction respectively, lead to increased selectivity while maintaining high linguistic task accuracy. As such, MLP probes with hundreds or 1000 hidden units, as is common, are overparameterized.

Constraining the **number of training examples** is effective for part-of-speech, suggesting that learning each linguistic task requires fewer samples than our control task. However, for dependency edge prediction, this leads to significantly reduced linguistic task accuracy. Finally, we find that the right **weight decay** constant can also lead to high-accuracy, high-selectivity probes, especially for dependency edge prediction. As shown, however, it is unclear what hyperparameters to use (e.g., weight decay 0.1) to achieve both high accuracy and high selectivity; that is, finding selective MLP probes is non-trivial.

Applying **dropout**, the most popular probing regularization method (Adi et al., 2017; Belinkov and Glass, 2019; Şahin et al., 2019; Kim et al., 2019; Elloumi et al., 2018; Belinkov and Glass, 2017; Belinkov et al., 2018) does not consistently lead to high-accuracy, high-selectivity MLP probes across a broad range of dropout probabilities ($p = 0.2$ to $p = 0.8$) on part-of-speech tagging. For dependency edge prediction, dropout of $p = 0.6$ improves the selectivity of MLP-2 but not MLP-1, and considerably increases the already relatively large selectivity of the bilinear probe. **Early stopping** in the ranges tested also has little impact on part-of-speech tagging, selectivity, but does improve selectivity of MLP dependency edge prediction probes.

From our study, we pick a set of hyperparameters for linear, bilinear, MLP-1 and MLP-2 probes to encourage selectivity and linguistic task accuracy together, to compare to default parameters and dropout. We chose rank constraints of 10 and 45, respectively (with no other changes,) for linear and MLP part-of-speech tagging probes, weight decay of 0.01 for the bilinear dependency probe, and weight decay of 0.1 for MLP dependency probes. We report the results of these probes in Table 1 (bottom). In all cases, we see that the right choice of probe leads to considerably higher selectivity than dropout or no regularization. In particular, for part-of-speech tagging, our chosen MLP-1

probe achieves 16.6 selectivity, up from 4.5, and on dependency head prediction, 6.2 selectivity, up from -0.7.

### 3.6 Discussion

Our most consistent result seems to be that all probes, whether linear, bilinear, or multi-layer perceptron, are over-parameterized and needlessly high-capacity if using defaults like full-rank weight matrices, hidden states with a few hundred dimensions, and moderate dropout. We can tell this is the case because we're able to heavily constrain the probes (e.g., to rank or 10-dimensional hidden states with little loss in accuracy.

We find that the most selective probes of those tested, even after careful complexity control, are linear or bilinear models. They also have the advantage that they exhibit high selectivity without the need to search over complexity control methods.

However, the most accurate probes on the more complex task of dependency edge prediction are MLPs, even with hyperparameters tuned for selectivity. This suggests that while much of the part-of-speech information of ELMo is extractable linearly, some information about syntactic trees is not available to a bilinear function. In some cases, therefore, one might opt for an MLP probe to extract non-linear features, while optimizing for selectivity through hyperparameter choices.

**Errors in Selective and Non-Selective Probes** Do selective and non-selective probes make different types of errors? We ran a qualitative study on this, training ten MLP-1 probes and ten linear probes, each with default parameters, on part-of-speech tagging. We then manually inspected their aggregate confusion matrices for trends in differences between the models' errors.

While the MLP performed marginally better at recognizing many categories, the plurality of improvement over the linear probe by far was in correctly identifying the difference between nouns and adjectives in phrases. For example,

> Kan.-based/JJ National/NNP Pizza/NNP
> rental/JJ equipment/NN

were correctly labeled by the MLP but not the linear probe, which incorrectly labeled the adjectives as nouns. As can be seen with the second example, the distinction between a *JJ NN* modified noun and a *NN NN* noun compound is quite subtle, and the MLP picks up on the distinction considerably better.

The linear probe, however, was substantially more accurate at predicting the NNP tag, which the MLP probe frequently mislabeled as NNPS. Manual inspection showed a general trend:

> Environmental/NNP Systems/NNP Co./NNP
> Cara/NNP Operations/NNP Co./NNP
> 7.8/CD %/NN stake/NN in/IN Dataproducts/NNP

In each case, the MLP probe mislabeled the word with the suffix *-s* as NNPS. The linear probe was considerably less prone to this error. We hypothesize that this is because the MLP probe is expressive enough to pick up on (spurious) markers of plurality as well as status as a proper noun independently and combine them, whereas the linear probe is less able to do so. If this hypothesis is true, then this serves as an example of how less selective probes may be less faithful in representing the linguistic information of the model being probed, since features may be combined to make fine-grained distinctions.

## 4 Selectivity Differences Confound Layer Comparisons

In this section, we use selectivity to shed light on confounding factors when comparing the linguistic capabilities of different representations. Multiple studies have found probes on ELMo1 to perform better at part-of-speech tagging than probes on ELMo2 (Peters et al., 2018a; Tenney et al., 2019; Liu et al., 2019). As we note, these results depend on the probe as well as the representation; given what we know about probes' capacity for memorizing at the type level, we explore an alternative to the hypothesis that ELMo1 has higher-quality part-of-speech representations than ELMo2. In particular, word identities are strong features in part-of-speech tagging when used in combination with other indicators; since ELMo1 is closer to the word representations than ELMo2, it may be easier to identify word identities from it, meaning the probe may utilize word identities more readily, as opposed to picking up on a representation of part-of-speech.

### 4.1 Experiments

We run experiments on the first and second contextual layers of ELMo, denoted ELMo1 and ELMo2.

We also examine the representations of an untrained BiLSTM run on the non-contextual character CNN word embeddings of ELMo, shown to be a strong baseline contextualization method, but without any linguistic knowledge learned from context (Zhang and Bowman, 2018; Hewitt and Manning, 2019). We denote this model Proj0.

We train linear and MLP-1 probes for part-of-speech tagging, and bilinear and MLP-1 probes for dependency edge prediction, all with default hyperparameters (§ 3.2). We examine both the linguistic task accuracy and selectivity achieved by each probe on each representation.

### 4.2 Results & Discussion

We find probes on ELMo2 to be strikingly more selective than those on ELMo1, consistent across all probes, both for part-of-speech tagging and dependency head prediction. In particular, the linear probe on ELMo2 achieves selectivity of 31.4, compared to selectivity of 26.0 for ELMo1, for a gain of 5.4. The same probe achieves 96.6 linguistic task accuracy on ELMo2 and 97.2 on ELMo1, for a loss of 0.6. The MLP probe shows roughly the same result. So, does ELMo1 have a better grasp of part-of-speech than ELMo2? Our results, summarized in Table 2, offer the alternative hypothesis that probes use word identity as a feature to predict part-of-speech, and that feature is less easily available in ELMo2 than ELMo1.

Probes on Proj0 and ELMo2 achieve similar part-of-speech tagging accuracy, echoing findings of (Zhang and Bowman, 2018), but we find that Proj0 is far less selective, suggesting that probes on ELMo2 rely far less on word identities than those on Proj0. Without considering selectivity, it might be thought that ELMo2 encodes nothing about part-of-speech, since it doesn't beat the Proj0 random representation baseline. Taking selectivity into account, we see that probes on ELMo2 are unable to rely on word identity features like those on Proj0, so to achieve high accuracy, they must rely on emergent properties of the representation.

| **Part-of-speech Tagging** | | | | |
|---|---|---|---|---|
| | Linear | | MLP-1 | |
| **Model** | Accuracy | Selectivity | Accuracy | Selectivity |
| Proj0 | 96.3 | 20.6 | 97.1 | 1.6 |
| ELMo1 | 97.2 | 26.0 | 97.3 | 4.5 |
| ELMo2 | 96.6 | 31.4 | 97.0 | 8.8 |
| **Dependency Edge Prediction** | | | | |
| | Bilinear | | MLP-1 | |
| **Model** | Accuracy | Selectivity | Accuracy | Selectivity |
| Proj0 | 79.9 | -4.3 | 86.5 | -9.0 |
| ELMo1 | 89.7 | 6.7 | 92.5 | -1.0 |
| ELMo2 | 84.5 | 6.2 | 89.5 | 1.4 |

Table 2: Part-of-speech and dependency edge prediction probe accuracies and selectivities across three representations. ELMo1 and ELMo2 are the two contextual layers of ELMo, while Proj0 refers to an untrained BiLSTM contextualization of ELMo's non-contextual character CNN representations.

## 5 Related Work

Early work in probing, (also known as diagnostic classification (Hupkes et al., 2018),) extracted properties like parts-of-speech, gender, tense, and number from distributional word vector spaces like word2vec and GloVe (Mikolov et al., 2013; Pennington et al., 2014) using linear classifiers (Köhn, 2015; Gupta et al., 2015). Soon after, the investigation of intermediate layers of deep models using linear probes was introduced independently by Ettinger et al. (2016) and Shi et al. (2016) in NLP and Alain and Bengio (2016) in computer vision.

Since then, probing methods have varied as to whether they investigate whole-sentence properties like sentence length and word content using a sentence vector (Shi et al., 2016; Adi et al., 2017; Conneau et al., 2018), word properties like verb tense or part-of-speech using word vectors (Shi et al., 2016; Belinkov et al., 2017; Liu et al., 2019), or word-pair properties like syntactic relationships using pairs of vectors (Tenney et al., 2019; Hewitt and Manning, 2019). Probes have been used to make relative claims between models or components (Adi et al., 2017; Liu et al., 2019; Belinkov et al., 2017) or absolute claims about models above baselines. Probes have also been used to test hypotheses about the mechanisms by which models perform tasks (Hupkes et al., 2018; Giulianelli et al., 2018).

Previous work has made extensive use of control representations like non-contextual word embeddings or models with random weights (Belinkov et al., 2017; Tenney et al., 2019; Saphra and Lopez, 2019; Hewitt and Manning, 2019); our control tasks provide a complementary perspective, measuring a probe's ability to decode a random function from the representation of interest.

The most related work to ours is that of Zhang and Bowman (2018), who presented experiments for understanding the roles probe training sample size and memorization have on linguistic

task accuracy. They observed that untrained BiLSTM contextualizers achieved almost the same part-of-speech tagging accuracies as trained contextualizers, and found that by reducing the probe training set, the trained models could be shown to significantly outperform the untrained model. They evaluated which representations were easiest to memorize from by probing to predict nearby words, finding as we do that word identities are most easily available in untrained contextualizers' representations. They take this as evidence that gains in part-of-speech probing accuracy on the trained representations over the untrained representations are due to linguistic properties, not memorization. Our experiments with selectivity complement their results, finding among other things that even though untrained BiLSTMs are better for memorization than ELMo, there is still a striking capacity for memorization using ELMo when using high-capacity probes.

### 5.1 Random tasks

Zhang et al. (2017) defined completely random tasks related to Rademacher complexity (Bartlett and Mendelson, 2001) to understand the capacity of neural networks to overfit, showing that they are expressive enough to fit random noise, but still function as effective models. In our random control tasks, randomness is applied at the type-level rather than at the example-level, and are designed to have strong non-linguistic structure as opposed to absolutely no structure. While the tasks of Zhang et al. (2017) aid in understanding the expressivity of neural nets, our control tasks aid in understanding the expressivity of a probe model with respect to a specific linguistic task.

## 6 Conclusion

Through probing methods, it has been shown that a broad range of supervised learning tasks can be turned into tools for understanding the properties of contextual word representations (Conneau et al., 2018; Tenney et al., 2019). Alain and Bengio (2016) suggested we may think of probes as "thermometers used to measure the temperature simultaneously at many different locations". We instead emphasize the joint roles of representations and probes together in achieving high accuracy on a task; we suggest that probes be thought of as *craftspeople*; their performance depends not only on the materials they're given, but also on their expressivity.

To explore the relationship between representations, probes, and task accuracies, we defined control tasks, which by construction can only be learned by the probe itself. We've suggested that a probe which provides insights into the properties of the representation should be *selective*, achieving high linguistic task accuracy and low control task accuracy. Selectivity measures the probe's ability to make numerous output decisions independently of linguistic properties of the representation.

We've found that linear and bilinear models achieve higher selectivity at similar accuracy to MLP probes on part-of-speech tagging. MLP probes, achieving higher accuracy on the more complex task of dependency edge prediction, can be re-designed to achieve higher selectivity at a relatively small cost to dependency edge accuracy, but often not through dropout, the most popular MLP probe regularization method.

Finally, we showed how selectivity can be used to provide added context to probing results, demonstrating that marginal differences in part-of-speech tagging accuracy between ELMo1 and ELMo2 correspond to large differences in selectivity, and similarly, the even though ELMo2 achieves similar part-of-speech tagging accuracy to a random representation baseline, ELMo2 achieves it with much higher selectivity.

As probes are used increasingly to study representations, we hope that control tasks and selectivity, as diagnostic tools, can help us better interpret the results of these probes, ultimately leading us to better understand what is learned by these remarkably effective representations.

**Reproducibility.** All code, data, and experiments are available at `https://worksheets.codalab.org/worksheets/0xb0c351d6f1ac4c51b54f1023786bf6b2`.

## Acknowledgements

We would like to thank Kevin Clark, Robin Jia, Peng Qi, Vivek Kulkarni, Surya Ganguli, Tatsu Hashimoto, and Nelson Liu for helpful discussions and feedback. We would like to thank our reviewers for clarifying comments and suggestions for extra experiments. This work was funded under a PECASE Award.

## References


Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2017. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. In *International Conference on Learning Representations*.

Guillaume Alain and Yoshua Bengio. 2016. Understanding intermediate layers using linear classifier probes. In *International Conference on Learning Representations*.

Peter L Bartlett and Shahar Mendelson. 2001. Rademacher and gaussian complexities: Risk bounds and structural results. In *International Conference on Computational Learning Theory*, pages 224–240. Springer.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017. What do neural machine translation models learn about morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 861–872. Association for Computational Linguistics.

Yonatan Belinkov and James Glass. 2017. Analyzing hidden representations in end-to-end automatic speech recognition systems. In *Advances in Neural Information Processing Systems*, pages 2441–2451.

Yonatan Belinkov and James Glass. 2019. Analysis methods in neural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 7:49–72.

Yonatan Belinkov, Lluís Màrquez, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2018. Evaluating layers of representation in neural machine translation on part-of-speech and semantic tagging tasks. *arXiv preprint arXiv:1801.07772*.

Arianna Bisazza and Clara Tump. 2018. The lazy encoder: A fine-grained analysis of the role of morphology in neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2871–2876.

Alexis Conneau, Germán Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single \$&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. Association for Computational Linguistics.

Zied Elloumi, Laurent Besacier, Olivier Galibert, and Benjamin Lecouteux. 2018. Analyzing learned representations of a deep ASR performance prediction model. *arXiv preprint arXiv:1808.08573*.

Allyson Ettinger, Ahmed Elgohary, Colin Phillips, and Philip Resnik. 2018. Assessing composition in sentence vector representations. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1790–1801.

Allyson Ettinger, Ahmed Elgohary, and Philip Resnik. 2016. Probing for semantic evidence of composition by means of simple classification tasks. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 134–139.

Mario Giulianelli, Jack Harding, Florian Mohnert, Dieuwke Hupkes, and Willem Zuidema. 2018. Under the hood: Using diagnostic classifiers to investigate and improve how language models track agreement information. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 240–248, Brussels, Belgium. Association for Computational Linguistics.

Abhijeet Gupta, Gemma Boleda, Marco Baroni, and Sebastian Padó. 2015. Distributional vectors encode referential attributes. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 12–21.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*. Association for Computational Linguistics.

Dieuwke Hupkes, Sara Veldhoen, and Willem Zuidema. 2018. Visualisation and 'diagnostic classifiers' reveal how recurrent and recursive neural networks process hierarchical structure. *Journal of Artificial Intelligence Research*, 61:907–926.

Najoung Kim, Roma Patel, Adam Poliak, Alex Wang, Patrick Xia, R Thomas McCoy, Ian Tenney, Alexis Ross, Tal Linzen, Benjamin Van Durme, et al. 2019. Probing what different nlp tasks teach machines about function word comprehension. *arXiv preprint arXiv:1904.11544*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Arne Köhn. 2015. WhatâĂŹs in an embedding? analyzing word embeddings through multilingual evaluation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2067–2073.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. Linguistic knowledge and transferability of contextual

representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):313–330.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *LREC*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018a. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 2227–2237.

Matthew Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. 2018b. Dissecting contextual word embeddings: Architecture and representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509.

Peng Qi and Christopher D. Manning. 2017. Arc-swift: A novel transition system for dependency parsing. In *Association for Computational Linguistics (ACL)*.

Peng Qian, Xipeng Qiu, and Xuanjing Huang. 2016. Investigating language universal and specific properties in word embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1478–1488.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1:8.

Gözde Gül Şahin, Clara Vania, Ilia Kuznetsov, and Iryna Gurevych. 2019. Linspector: Multilingual probing tasks for word representations. *arXiv preprint arXiv:1903.09442*.

Naomi Saphra and Adam Lopez. 2019. Understanding learning dynamics of language models with svcca. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*. Association for Computational Linguistics.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural mt learn source syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1526–1534.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Sam Bowman, Dipanjan Das, and Ellie Pavlick. 2019. What do you learn from context? probing for sentence structure in contextualized word representations. In *International Conference on Learning Representations*.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2017. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*.

Kelly W Zhang and Samuel R Bowman. 2018. Language modeling teaches you more syntax than translation does: Lessons learned through auxiliary task analysis. *arXiv preprint arXiv:1809.10040*.

# Specializing Word Embeddings (for Parsing) by Information Bottleneck

**Xiang Lisa Li**
Department of Computer Science
Johns Hopkins University
xli150@jhu.edu

**Jason Eisner**
Department of Computer Science
Johns Hopkins University
jason@cs.jhu.edu

## Abstract

Pre-trained word embeddings like ELMo and BERT contain rich syntactic and semantic information, resulting in state-of-the-art performance on various tasks. We propose a very fast variational information bottleneck (VIB) method to nonlinearly compress these embeddings, keeping only the information that helps a discriminative parser. We compress each word embedding to either a discrete tag or a continuous vector. In the *discrete* version, our automatically compressed tags form an alternative tag set: we show experimentally that our tags capture most of the information in traditional POS tag annotations, but our tag sequences can be parsed more accurately at the same level of tag granularity. In the *continuous* version, we show experimentally that moderately compressing the word embeddings by our method yields a more accurate parser in 8 of 9 languages, unlike simple dimensionality reduction.

## 1 Introduction

Word embedding systems like BERT and ELMo use spelling and context to obtain contextual embeddings of word tokens. These systems are trained on large corpora in a task-independent way. The resulting embeddings have proved to then be useful for both syntactic and semantic tasks, with different layers of ELMo or BERT being somewhat specialized to different kinds of tasks (Peters et al., 2018b; Goldberg, 2019). State-of-the-art performance on many NLP tasks can be obtained by fine-tuning, i.e., back-propagating task loss all the way back into the embedding function (Peters et al., 2018a; Devlin et al., 2018).

In this paper, we explore what task-specific information appears in the embeddings *before* fine-tuning takes place. We focus on the task of dependency parsing, but our method can be easily extended to other syntactic or semantic tasks. Our method compresses the embeddings by extracting just their syntactic properties—specifically, the information needed to reconstruct parse trees (because that is our task). Our nonlinear, stochastic compression function is explicitly trained by variational information bottleneck (VIB) to forget task-irrelevant information. This is reminiscent of canonical correspondence analysis (Anderson, 2003), a method for reducing the dimensionality of an input vector so that it remains predictive of an output vector, although we are predicting an output tree instead. However, VIB goes beyond mere dimensionality reduction to a fixed lower dimensionality, since it also avoids unnecessary *use* of the dimensions that are available in the compressed representation, blurring unneeded capacity via randomness. The effective number of dimensions may therefore vary from token to token. For example, a parser may be content to know about an adjective token only that it is adjectival, whereas to find the dependents of a verb token, it may need to know

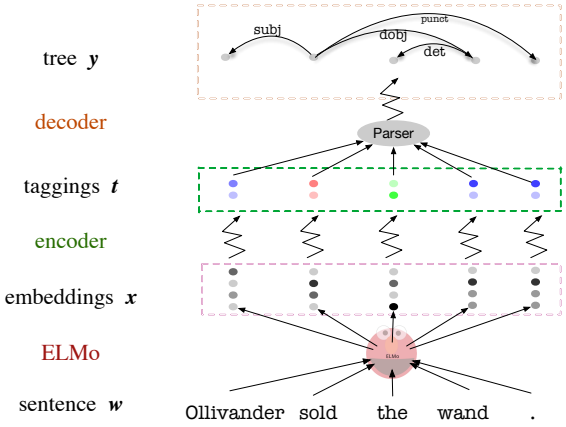


Figure 1: Our instantiation of the information bottleneck, with bottleneck variable $T$. A jagged arrow indicates a stochastic mapping, i.e. the jagged arrow points from the parameters of a distribution to a sample drawn from that distribution.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2744–2754, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

the verb's number and transitivity, and to attach a preposition token, it may need to know the identity of the preposition.

We try compressing to both discrete and continuous task-specific representations. Discrete representations yield an interpretable clustering of words. We also extend information bottleneck to allow us to control the contextual specificity of the token embeddings, making them more like type embeddings.

This specialization method is complementary to the previous fine-tuning approach. Fine-tuning introduces *new* information into word embeddings by backpropagating the loss, whereas the VIB method learns to exploit the *existing* information found by the ELMo or BERT language model. VIB also has less capacity and less danger of overfitting, since it fits fewer parameters than fine-tuning (which in the case of BERT has the freedom to adjust the embeddings of all words and word pieces, even those that are rare in the supervised fine-tuning data). VIB is also very fast to train on a single GPU.

We discover that our syntactically specialized embeddings are predictive of the gold POS tags in the setting of few-shot-learning, validating the intuition that a POS tag summarizes a word token's *syntactic* properties. However, our representations are tuned explicitly for discriminative parsing, so they prove to be even more useful for this task than POS tags, even at the same level of granularity. They are also more useful than the uncompressed ELMo representations, when it comes to generalizing to test data. (The first comparison uses discrete tags, and the second uses continuous tags.)

## 2 Background: Information Bottleneck

The information bottleneck (IB) method originated in information theory and has been adopted by the machine learning community as a training objective (Tishby et al., 2000) and a theoretical framework for analyzing deep neural networks (Tishby and Zaslavsky, 2015).

Let $X$ represent an "input" random variable such as a sentence, and $Y$ represent a correlated "output" random variable such as a parse. Suppose we know the joint distribution $p(X, Y)$. (In practice, we will use the empirical distribution over a sample of $(x, y)$ pairs.) Our goal is to learn a stochastic map $p_\theta(t \mid x)$ from $X$ to some compressed representation $T$, which in our setting will be something like a tag sequence. IB seeks to minimize

$$\mathcal{L}_{IB} = -\,\mathrm{I}(Y;T) + \beta \cdot \mathrm{I}(X;T) \qquad (1)$$

where $\mathrm{I}(\cdot;\cdot)$ is the mutual information.[1] A low loss means that $T$ does not retain very much information about $X$ (the second term), while still retaining enough information to predict $Y$.[2] The balance between the two MI terms is controlled by a Lagrange multiplier $\beta$. By increasing $\beta$, we increase the pressure to keep $\mathrm{I}(X;T)$ small, which "narrows the bottleneck" by favoring compression over predictive accuracy $\mathrm{I}(Y;T)$. Regarding $\beta$ as a Lagrange multiplier, we see that the goal of IB is to maximize the predictive power of $T$ subject to some constraint on the amount of information about $X$ that $T$ carries. If the map from $X$ to $T$ were deterministic, then it could lose information only by being non-injective: the traditional example is dimensionality reduction, as in the encoder of an encoder-decoder neural net. But IB works even if $T$ can take values throughout a high-dimensional space, because the randomness in $p_\theta(t \mid x)$ means that $T$ is noisy in a way that wipes out information about $X$. Using a high-dimensional space is desirable because it permits the amount of effective dimensionality reduction to vary, with $T$ perhaps retaining much more information about some $x$ values than others, as long as the *average* retained information $\mathrm{I}(X;T)$ is small.

## 3 Formal Model

In this paper, we extend the original IB objective (1) and add terms $\mathrm{I}(T_i; X|\hat{X}_i)$ to control the context-sensitivity of the extracted tags. Here $T_i$ is the tag associated with the $i$th word, $X_i$ is the ELMo token embedding of the $i$th word, and $\hat{X}_i$ is the same word's ELMo type embedding (before context is incorporated).

$$\mathcal{L}_{IB} = -\,\mathrm{I}(Y;T) + \beta\,\mathrm{I}(X;T) + \gamma \sum_{i=1}^{n} \mathrm{I}(T_i; X|\hat{X}_i) \quad (2)$$

In this section, we will explain the motivation for the additional term and how to efficiently estimate variational bounds on all terms (lower bound for $\mathrm{I}(Y;T)$ and upper bound for the rest).[3]

[1] In our IB notation, larger $\beta$ means more compression. Note that there is another version of IB that puts $\beta$ as the coefficient in front of $\mathrm{I}(Y;T)$: $\mathcal{L}_{IB} = -\beta \cdot \mathrm{I}(Y;T) + \mathrm{I}(X;T)$ The two versions are equivalent.

[2] Since $T$ is a stochastic function of $X$ with no access to $Y$, it obviously cannot convey more information about $Y$ than the uncompressed input $X$ does. As a result, $Y$ is independent of $T$ given $X$, as in the graphical model $T \to X \to Y$.

[3] Traditional Shannon entropy $\mathrm{H}(\cdot)$ is defined on discrete variables. In the case of continuous variables, we interpret H

We instantiate the variational IB (VIB) estimation method (Alemi et al., 2016) on our dependency parsing task, as illustrated in Figure 1. We compress a sentence's word embeddings $X_i$ into continuous vector-valued tags or discrete tags $T_i$ ("encoding") such that the tag sequence $T$ retains maximum ability to predict the dependency parse $Y$ ("decoding"). Our chosen architecture compresses each $X_i$ independently using the same stochastic, information-losing transformation.

The IB method introduces the new random variable $T$, the tag sequence that compresses $X$, by defining the conditional distribution $p_\theta(t \mid x)$. In our setting, $p_\theta$ is a stochastic tagger, for which we will adopt a parametric form (§3.1 below). Its parameters $\theta$ are chosen to minimize the IB objective (2). By IB's independence assumption,[2] the joint probability can be factored as $p_\theta(x, y, t) = p(x) \cdot p(y \mid x) \cdot p_\theta(t \mid x)$.

### 3.1 $\mathrm{I}(X;T)$ — the Token Encoder $p_\theta(t \mid x)$

Under this distribution, $\mathrm{I}(X;T) \stackrel{\text{def}}{=} \mathbb{E}_{x,t}\left[\log \frac{p_\theta(t|x)}{p_\theta(t)}\right] = \mathbb{E}_x\left[\mathbb{E}_{t\sim p_\theta(t|x)}\left[\log \frac{p_\theta(t|x)}{p_\theta(t)}\right]\right]$. Making this term small yields a representation $T$ that, on average, retains little information about $X$. The outer expectation is over the true distribution of sentences $x$; we use an empirical estimate, averaging over the unparsed sentences in a dependency treebank. To estimate the inner expectation, we could sample, drawing taggings $t$ from $p_\theta(t \mid x)$.

We must also compute the quantities within the inner brackets. The $p_\theta(t \mid x)$ term is defined by our parametric form. The troublesome term is $p_\theta(t) = \mathbb{E}_{x'}\left[p_\theta(t \mid x')\right]$, since even estimating it from a treebank requires an inner loop over treebank sentences $x'$. To avoid this, variational IB replaces $p_\theta(t)$ with some variational distribution $r_\psi(t)$. This can only increase our objective function, since the difference between the variational and original versions of this term is a KL divergence and hence non-negative:

$$\overbrace{\mathop{\mathbb{E}}_{x}\Big[\mathop{\mathbb{E}}_{t\sim p_\theta(t|x)}\Big[\log \frac{p_\theta(t|x)}{r_\psi(t)}\Big]\Big]}^{\text{upper bound}} - \overbrace{\mathop{\mathbb{E}}_{x}\Big[\mathop{\mathbb{E}}_{t\sim p_\theta(t|x)}\Big[\log \frac{p_\theta(t \mid x)}{p_\theta(t)}\Big]\Big]}^{\mathrm{I}(X;T)}$$
$$= \mathop{\mathbb{E}}_{x}\left[\mathrm{KL}(p_\theta(t) \mid\mid r_\psi(t))\right] \geq 0$$

Thus, the variational version (the first term above) is indeed an upper bound for $\mathrm{I}(X;T)$ (the second term above). We will minimize this upper bound by adjusting not only $\theta$ but also $\psi$, thus making the bound as tight as possible given $\theta$. Also we will no longer need to sample $t$ for the inner expectation of the upper bound, $\mathbb{E}_{t\sim p_\theta(t|x)}\left[\log \frac{p_\theta(t|x)}{r_\psi(t)}\right]$, because this expectation equals $\mathrm{KL}[p_\theta(t \mid x) \mid\mid r_\psi(t)]$, and we will define the parametric $p_\theta$ and $r_\psi$ so that this KL divergence can be computed exactly: see §4.

### 3.2 Two Token Encoder Architectures

We choose to define $p_\theta(t \mid x) = \prod_{i=1}^{n} p_\theta(t_i \mid x_i)$. That is, our stochastic encoder will compress each word $x_i$ individually (although $x_i$ is itself a representation that depends on context): see Figure 1. We make this choice not for computational reasons—our method would remain tractable even without this—but because our goal in this paper is to find the syntactic information in each individual ELMo token embedding (a goal we will further pursue in §3.3 below).

To obtain continuous tags, define $p_\theta(t_i \mid x_i)$ such that $t_i \in \mathbb{R}^d$ is Gaussian-distributed with mean vector and diagonal covariance matrix computed from the ELMo word vector $x_i$ via a feedforward neural network with $2d$ outputs and no transfer function at the output layer. To ensure positive semidefiniteness of the diagonal covariance matrix, we squared the latter $d$ outputs to obtain the diagonal entries.[4]

Alternatively, to obtain discrete tags, define $p_\theta(t_i \mid x_i)$ such that $t_i \in \{1, \ldots, k\}$ follows a softmax distribution, where the $k$ softmax parameters are similarly computed by a feedforward network with $k$ outputs and no transfer function at the output layer.

We similarly define $r_\psi(t) = \prod_{i=1}^{n} r_\psi(t_i)$, where $\psi$ directly specifies the $2d$ or $k$ values corresponding to the output layer above (since there is no input $x_i$ to condition on).

### 3.3 $\mathrm{I}(T_i; X|\hat{X}_i)$ — the Type Encoder $s_\xi(t_i|\hat{x}_i)$

While the IB objective (1) asks each tag $t_i$ to be informative about the parse $Y$, we were concerned that it might not be interpretable as a tag of word $i$ specifically. Given ELMo or any other black-box conversion of a length-$n$ sentence to a sequence of contextual vectors $x_1, \ldots, x_n$, it is possible that $x_i$

to instead denote differential entropy (which would be $-\infty$ for discrete variables). Scaling a continuous random variable affects its differential entropy—but not its mutual information with another random variable, which is what we use here.

[4] Our restriction to diagonal covariance matrices follows Alemi et al. (2016). In pilot experiments that dropped this restriction, we found learning to be numerically unstable, although that generalization is reasonable in principle.

contains not only information about word $i$ but also information describing word $i+1$, say, or the syntactic constructions in the vicinity of word $i$. Thus, while $p_\theta(t_i \mid x_i)$ might extract some information from $x_i$ that is very useful for parsing, there is no guarantee that this information came from word $i$ and not its neighbors. Although we *do* want tag $t_i$ to consider context—e.g., to distinguish between noun and verb uses of word $i$—we want "most" of $t_i$'s information to come from word $i$ itself. Specifically, it should come from ELMo's level-0 embedding of word $i$, denoted by $\hat{x}_i$—a word *type* embedding that does *not* depend on context.

To penalize $T_i$ for capturing "too much" contextual information, our modified objective (2) adds a penalty term $\gamma \cdot \mathrm{I}(T_i; X|\hat{X}_i)$, which measures the amount of information about $T_i$ given by the sentence $X$ as a whole, beyond what is given by $\hat{X}_i$: $\mathrm{I}(T_i; X \mid \hat{X}_i) \stackrel{\text{def}}{=} \mathbb{E}_x\left[\mathbb{E}_{t_i \sim p_\theta(t_i|x)}\left[\log \frac{p_\theta(t_i|x)}{p_\theta(t_i|\hat{x}_i)}\right]\right]$. Setting $\gamma > 0$ will reduce this contextual information.

In practice, we found that $\mathrm{I}(T_i; X \mid \hat{X}_i)$ was small even when $\gamma = 0$, on the order of 3.5 nats whereas $\mathrm{I}(T_i; X)$ was 50 nats. In other words, the tags extracted by the classical method were already fairly local, so increasing $\gamma$ above 0 had little qualitative effect. Still, $\gamma$ might be important when applying our method to ELMo's competitors such as BERT.

We can derive an upper bound on $\mathrm{I}(T_i; X|\hat{X}_i)$ by approximating the conditional distribution $p_\theta(t_i \mid \hat{x}_i)$ with a variational distribution $s_\xi(t_i \mid \hat{x}_i)$, similar to §3.1.

$$\overbrace{\mathop{\mathbb{E}}_{x}\Big[\mathop{\mathbb{E}}_{t_i \sim p_\theta(t_i|x)}\Big[\log \frac{p_\theta(t_i|x)}{s_\xi(t_i|\hat{x}_i)}\Big]\Big]}^{\text{upper bound}} - \overbrace{\mathop{\mathbb{E}}_{x}\Big[\mathop{\mathbb{E}}_{t_i \sim p_\theta(t_i|x)}\Big[\log \frac{p_\theta(t_i|x)}{p_\theta(t_i|\hat{x}_i)}\Big]\Big]}^{\mathrm{I}(T_i;X|\hat{X}_i)}$$
$$= \mathop{\mathbb{E}}_{x}\left[\mathrm{KL}(p_\theta(t_i \mid \hat{x}_i) \mid\mid s_\xi(t_i \mid \hat{x}_i))\right] \geq 0$$

We replace it in (2) with this upper bound, which is equal to $\mathbb{E}_x\left[\sum_{i=1}^n \mathrm{KL}[p_\theta(t_i|x) \mid\mid s_\xi(t_i \mid \hat{x}_i)]\right]$.

The formal presentation above does not assume the specific factored model that we adopted in §3.2. When we adopt that model, $p_\theta(t_i \mid x)$ above reduces to $p_\theta(t_i \mid x_i)$—but our method in this section still has an effect, because $x_i$ still reflects the context of the full sentence whereas $\hat{x}_i$ does not.

**Type Encoder Architectures** Notice that $s_\xi(t_i \mid \hat{x}_i)$ may be regarded as a type encoder, with parameters $\xi$ that are distinct from the parameters $\theta$ of our token encoder $p_\theta(t_i \mid x_i)$. Given a choice of neural architecture for $p_\theta(t_i \mid x_i)$ (see §3.2), we always use the same architecture for $s_\xi(t_i \mid \hat{x}_i)$, except that $p_\theta$ takes a token vector as input whereas $s_\xi$ takes a context-independent type vector. $s_\xi$ is not used at test time, but only as part of our training objective.

### 3.4 $\mathrm{I}(Y;T)$ — the Decoder $q_\phi(y \mid t)$

Finally, $\mathrm{I}(Y;T) \stackrel{\text{def}}{=} \mathbb{E}_{y,t\sim p_\theta}\left[\log \frac{p_\theta(y|t)}{p(y)}\right]$. The $p(y)$ can be omitted during optimization as it does not depend on $\theta$. Thus, making $\mathrm{I}(Y;T)$ large tries to obtain a high log-probability $p_\theta(y \mid t)$ for the true parse $y$ when reconstructing it from $t$ alone.

But how do we compute $p_\theta(y \mid t)$? This quantity effectively marginalizes over possible sentences $x$ that *could have* explained $t$. Recall that $p_\theta$ is a joint distribution over $x, y, t$: see just above §3.1. So $p_\theta(y \mid t) \stackrel{\text{def}}{=} \frac{\sum_x p_\theta(x,y,t)}{\sum_{x,y'} p_\theta(x,y',t)}$. To estimate these sums accurately, we would have to identify the sentences $x$ that are most consistent with the tagging $t$ (that is, $p(x) \cdot p_\theta(t|x)$ is large): these contribute the largest summands, but might not appear in any corpus.

To avoid this, we replace $p_\theta(y \mid t)$ with a variational approximation $q_\phi(y \mid t)$ in our formula for $\mathrm{I}(Y;T)$. Here $q_\phi(\cdot \mid \cdot)$ is a tractable conditional distribution, and may be regarded as a stochastic parser that runs on a compressed tag sequence $t$ instead of a word embedding sequence $x$. This modified version of $\mathrm{I}(Y;T)$ forms a lower bound on $\mathrm{I}(Y;T)$, for any value of the variational parameters $\phi$, since the difference between them is a KL divergence and hence positive:

$$\overbrace{\mathop{\mathbb{E}}_{y,t\sim p_\theta}\left[\log \frac{p_\theta(y|t)}{p(y)}\right]}^{\mathrm{I}(Y;T)} - \overbrace{\mathop{\mathbb{E}}_{y,t\sim p_\theta}\left[\log \frac{q_\phi(y|t)}{p(y)}\right]}^{\text{lower bound}}$$
$$= \mathop{\mathbb{E}}_{t\sim p_\theta}\left[\mathrm{KL}(p_\theta(y \mid t) \mid\mid q_\phi(y \mid t))\right] \geq 0$$

We will maximize this lower bound of $\mathrm{I}(Y;T)$ with respect to both $\theta$ and $\phi$. For any given $\theta$, the optimal $\phi$ minimizes the expected KL divergence, meaning that $q_\phi$ approximates $p_\theta$ well.

More precisely, we again drop $p(y)$ as constant and then maximize a *sampling-based estimate* of $\mathbb{E}_{y,t\sim p_\theta}\left[\log q_\phi(y|t)\right]$. To sample $y, t$ from the joint $p_\theta(x, y, t)$ we must first sample $x$, so we rewrite as $\mathbb{E}_{x,y}\left[\mathbb{E}_{t\sim p_\theta(t|x)}\left[\log q_\phi(y|t)\right]\right]$. The outer expectation $\mathbb{E}_{x,y}$ is estimated as usual over a training treebank. The expectation $\mathbb{E}_{t\sim p_\theta(t|x)}$ recognizes that $t$ is stochastic, and again we estimate it by sampling. In short, when $t$ is a stochastic compression of a treebank sentence $x$, we would like our variational parser *on average* to assign high log-probability $q_\phi(y \mid t)$ to its treebank parse $y$.

**Decoder Architecture** We use the deep biaffine dependency parser (Dozat and Manning, 2016) as our variational distribution $q_\phi(y \mid t)$, which functions as the decoder. This parser uses a Bi-LSTM to extract features from compressed tags or vectors and assign scores to each tree edge, setting $q_\phi(y \mid t)$ proportional to the exp of the total score of all edges in $y$. During IB training, the code[5] computes only an approximation to $q_\phi(y|t)$ for the gold tree $y$ (although in principle, it could have computed the exact normalizing constant in polytime with Tutte's matrix-tree theorem (Smith and Smith, 2007; Koo et al., 2007; McDonald and Satta, 2007)). When we test the parser, the code does exactly find $\text{argmax}_y\, q_\phi(y \mid t)$ via the directed spanning tree algorithm of Edmonds (1966).

## 4 Training and Inference

With the approximations in §3, our final minimization objective is this upper bound on (2):

$$\mathop{\mathbb{E}}_{x,y}\Big[\mathop{\mathbb{E}}_{t\sim p_\theta(t|x)}[-\log q_\phi(y|t)] + \beta\,\mathrm{KL}(p_\theta(t|x)||r_\psi(t)) + \gamma\sum_{i=1}^{n}\mathrm{KL}(p_\theta(t_i \mid x)\;||\;s_\xi(t_i \mid \hat{x}_i))\Big] \quad (3)$$

We apply stochastic gradient descent to optimize this objective. To get a stochastic estimate of the objective, we first sample some $(x, y)$ from the treebank. We then have many expectations over $t \sim p_\theta(t \mid x)$, including the KL terms. We could estimate these by sampling $t$ from the token encoder $p_\theta(t \mid x)$ and then evaluating all $q_\phi, p_\theta, r_\psi$, and $s_\xi$ probabilities. However, in fact we use the sampled $t$ only to estimate the first expectation (by computing the decoder probability $q_\phi(y \mid t)$ of the gold tree $y$); we can compute the KL terms exactly by exploiting the structure of our distributions. The structure of $p_\theta$ and $r_\psi$ means that the first KL term decomposes into $\sum_{i=1}^{n}\mathrm{KL}(p_\theta(t_i|x_i)||r_\psi(t_i))$. All KL terms are now between either two Gaussian distributions over a continuous tagset[6] or two categorical distributions over a small discrete tagset.[7]

To compute the stochastic gradient, we run backpropagation on this computation. We must apply the reparametrization trick to backpropagate through the step that sampled $t$. This finds the gradient of parameters that derive $t$ from a random variate $z$, while holding $z$ itself fixed. For continuous $t$, we use the reparametrization trick for multivariate Gaussians (Rezende et al., 2014). For discrete $t$, we use the Gumbel-softmax variant (Jang et al., 2016; Maddison et al., 2017).

To evaluate our trained model's ability to parse a sentence $x$ from compressed tags, we obtain a parse as $\text{argmax}_y\, q_\phi(y \mid t)$, where $t \sim p_\theta(\cdot \mid x)$ is a single sample. A better parser would instead estimate $\text{argmax}_y\, \mathbb{E}_t\,[q_\phi(y \mid t)]$ where $\mathbb{E}_t$ averages over many samples $t$, but this is computationally hard.

| Language | Treebank | #Tokens | $\mathrm{H}(A \mid \hat{X})$ | $\mathrm{H}(A)$ |
|---|---|---|---|---|
| Arabic | PADT | 282k | 0.059 | 2.059 |
| Chinese | GSD | 123k | 0.162 | 2.201 |
| English | EWT | 254k | 0.216 | 2.494 |
| French | GSD | 400k | 0.106 | 2.335 |
| Hindi | HDTB | 351k | 0.146 | 2.261 |
| Portuguese | Bosque | 319k | 0.179 | 2.305 |
| Russian | GSD | 98k | 0.049 | 2.132 |
| Spanish | AnCora | 549k | 0.108 | 2.347 |
| Italian | ISDT | 298K | 0.120 | 2.304 |

Table 1: Statistics of the datasets used in this paper. "Treebank" is the treebank identifier in UD, "#Token" is the number of tokens in the treebank, "$\mathrm{H}(A)$" is the entropy of a gold POS tag (in nats), and "$\mathrm{H}(A \mid \hat{X})$" is the conditional entropy of a gold POS tag conditioned on a word type (in nats).

## 5 Experimental Setup

**Data** Throughout §§6–7, we will examine our compressed tags on a subset of Universal Dependencies (Nivre et al., 2018), or UD, a collection of dependency treebanks across 76 languages using the same POS tags and dependency labels. We experiment on Arabic, Hindi, English, French, Spanish, Portuguese, Russian, Italian, and Chinese (Table 1)—languages with different syntactic properties like word order. We use only the sentences with length $\leq 30$. For each sentence, $x$ is obtained by running the standard pre-trained ELMo on the UD token sequence (although UD's tokenization may not perfectly match that of ELMo's training data), and $y$ is the labeled UD dependency parse *without* any part-of-speech (POS) tags. Thus, our tags $t$ are tuned to predict only the dependency relations in UD, and not the gold POS tags $a$ also in UD.

**Pretrained Word Embeddings** For English, we used the pre-trained English ELMo model from the AllenNLP library (Gardner et al., 2017). For the

[5] We use the implementation from AllenNLP library (Gardner et al., 2017).

[6] $\mathrm{KL}(\mathcal{N}_0 \,||\, \mathcal{N}_1) = \frac{1}{2}(\mathrm{tr}(\Sigma_1^{-1}\Sigma_0) + (\mu_1-\mu_0)^T\Sigma_1^{-1}(\mu_1-\mu_0) - d + \log(\frac{\det(\Sigma_1)}{\det(\Sigma_0)})$

[7] $\mathrm{KL}(p_\theta(t_i|x_i)||r_\psi(t_i)) = \sum_{t_i=1}^{k} p_\theta(t_i \mid x_i)\log\frac{p_\theta(t_i|x_i)}{r_\psi(t_i)}$

other 8 languages, we used the pre-trained models from Che et al. (2018). Recall that ELMo has two layers of bidirectional LSTM (layer 1 and 2) built upon a context-independent character CNN (layer 0). We use either layer 1 or 2 as the input ($x_i$) to our token encoder $p_\theta$. Layer 0 is the input ($\hat{x}_i$) to our type encoder $s_\xi$. Each encoder network (§§3.2–3.3) has a single hidden layer with a tanh transfer function, which has $2d$ hidden units (typically 128 or 512) for continuous encodings and 512 hidden units for discrete encodings.

**Optimization** We optimize with Adam (Kingma and Ba, 2014), a variant of stochastic gradient descent. We alternate between improving the model $p_\theta(t|x)$ on even epochs and the variational distributions $q_\phi(y|t)$, $r_\psi(t)$, $s_\xi(t_i \mid \hat{x}_i)$ on odd epochs.

We train for 50 epochs with minibatches of size 20 and $L_2$ regularization. The learning rate and the regularization coefficients are tuned on dev data for each language separately. For each training sentence, we average over 5 i.i.d. samples of $T$ to reduce the variance of the stochastic gradient. The initial parameters $\theta, \phi, \psi, \xi$ are all drawn from $\mathcal{N}(0, I)$. We experiment with different dimensionalities $d \in \{5, 32, 256, 512\}$ for the continuous tags, and different cardinalities $k \in \{32, 64, 128\}$ for the discrete tag set. We also tried different values $\beta, \gamma \in \{10^{-6}, 10^{-5}, \cdots, 10^{1}\}$ of the compression tradeoff parameter. We use temperature annealing when sampling from the Gumbel-softmax distribution (§4). At training epoch $i$, we use temperature $\tau_i$, where $\tau_1 = 5$ and $\tau_{i+1} = \max(0.5, e^{-\gamma}\tau_i)$. We set the annealing rate $\gamma = 0.1$. During testing, we use $\tau = 0$, which gives exact softmax sampling.

## 6 Scientific Evaluation

In this section, we study what information about words is retained by our automatically constructed tagging schemes. First, we show the relationship between $I(Y;T)$ and $I(X;T)$ on English as we reduce $\beta$ to capture more information in our tags.[8]

Second, across 9 languages, we study how our automatic tags correlate with gold part-of-speech tags (and in English, with other syntactic properties), while suppressing information about semantic properties. We also show how decreasing $\beta$ gradually refines the automatic discrete tag set, giving intuitive fine-grained clusters of English words.

[8]We always set $\gamma = \beta$ to simplify the experimental design.

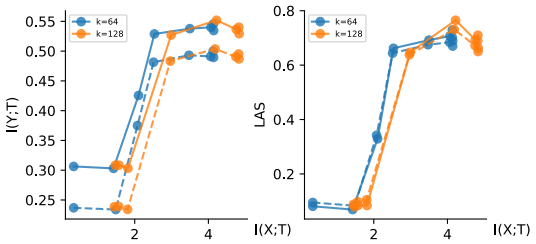


(a) Discrete Version

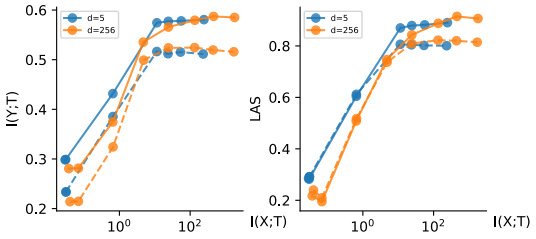


(b) Continuous Version

Figure 2: Compression-prediction tradeoff curves of VIB in our dependency parsing setting. The upper figures use discrete tags, while the lower figures use continuous tags. The dashed lines are for test data, and the solid lines for training data. The "dim" in the legends means the dimensionality of the continuous tag vector or the cardinality of the discrete tag set. On the left, we plot predictiveness $I(Y;T)$ versus $I(X;T)$ as we lower $\beta$ multiplicatively from $10^1$ to $10^{-6}$ on a log-scale. On the right, we alter the $y$-axis to show the labeled attachment score (LAS) of 1-best dependency parsing. All mutual information and entropy values in this paper are reported in nats per token. Furthermore, the mutual information values that we report are actually our variational upper bounds, as described in §3. The reason that $I(X;T)$ is so large for continuous tags is that it is *differential* mutual information (see footnote 3). Additional tradeoff curves w.r.t. $I(T_i; X \mid \hat{X}_i)$ are in Appendix B.

### 6.1 Tradeoff Curves

As we lower $\beta$ to retain more information about $X$, both $I(X;T)$ and $I(Y;T)$ rise, as shown in Figure 2. There are diminishing returns: after some point, the additional information retained in $T$ does not contribute much to predicting $Y$. Also noteworthy is that at each level of $I(X,T)$, very low-dimensional tags ($d = 5$) perform on par with high-dimensional ones ($d = 256$). (Note that the high-dimensional stochastic tags will be noisier to keep the same $I(X,T)$.) The low-dimensional tags allow far faster CPU parsing. This indicates that VIB can achieve strong practical task-specific compression.

### 6.2 Learned Tags vs. Gold POS Tags

We investigate how our automatic tag $T_i$ correlates with the gold POS tag $A_i$ provided by UD.

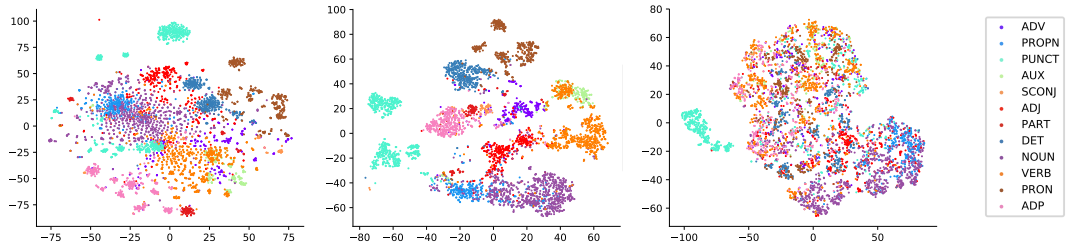


(a) ELMo, $I(X;T) = H(X) \approx 400.6$ (b) $I(X;T) \approx 24.3$ (c) $I(X;T) \approx 0.069$

Figure 3: t-SNE visualization of VIB model ($d = 256$) on the projected space of the continuous tags. Each marker in the figure represents a word token, colored by its gold POS tag. This series of figures (from left to right) shows a progression from no compression to moderate compression and to too-much compression.

**Continuous Version** We use t-SNE (van der Maaten and Hinton, 2008) to visualize our compressed continuous tags on held-out test data, coloring each token in Figure 3 according to its gold POS tag. (Similar plots for the discrete tags are in Figure 6 in the appendix.)

In Figure 3, the first figure shows the original uncompressed level-1 ELMo embeddings of the tokens in test data. In the two-dimensional visualization, the POS tags are vaguely clustered but the boundaries merge together and some tags are diffuse. The second figure is when $\beta = 10^{-3}$ (moderate compression): our compressed embeddings show clear clusters that correspond well to gold POS tags. Note that the gold POS tags were not used in training either ELMo or our method. The third figure is when $\beta = 1$ (too much compression), when POS information is largely lost. An interesting observation is that the purple NOUN and blue PROPN distributions overlap in the middle distribution, meaning that it was unnecessary to distinguish common nouns from proper nouns for purposes of our parsing task.[9]

**Discrete Version** We also quantify how well our specialized discrete tags capture the traditional POS categories, by investigating $I(A;T)$. This can be written as $H(A) - H(A \mid T)$. Similarly to §3.4, our probability distribution has the form $p_\theta(x, a, t) = p(x, a) \cdot p_\theta(t \mid x)$, leading us to write $H(A \mid T) \leq \mathbb{E}_{x,a}\left[\mathbb{E}_{t \sim p_\theta(t|x)}\left[-\log q(a \mid t)\right]\right]$ where $q(a \mid t) = \prod_i q(a_i \mid t_i)$ is a variational distribution that we train to minimize this upper bound. This is equivalent to training $q(a \mid t)$ by maximum conditional likelihood. In effect, we are doing transfer learning, fixing our trained IB encoder ($p_\theta$) and now using it to predict $A$ instead of $Y$, but otherwise following §3.4. We similarly upper-bound $H(A)$ by assuming a model $q'(a) = \prod_i q'(a_i)$ and estimating $q'$ as the empirical distribution over training tags. Having trained $q$ and $q'$ on training data, we estimate $H(A \mid T)$ and $H(A)$ using the same upper-bound formulas on our test data.

We experiment on all 9 languages, taking $T_i$ at the moderate compression level $\beta = 0.001$, $k = 64$. As Figure 4 shows, averaging over the 9 languages, the reconstruction retains 71% of POS information (and as high as 80% on Spanish and French). We can conclude that the information encoded in the specialized tags correlates with the gold POS tags, but does not perfectly predict the POS.

The graph in Figure 4 shows a "U-shaped" curve, with the best overall error rate at $\beta = 0.01$. That is, moderate compression of ELMo embeddings helps for predicting POS tags. Too much compression squeezes out POS-related information, while too little compression allows the tagger to overfit the training data, harming generalization to test data. We will see the same pattern for parsing in §7.

**Syntactic Features** As a quick check, we determine that our tags also make syntactic distinctions beyond those that are recognized by the UD POS tag set, such as tense, number, and transitivity. See Appendix D for graphs. For example, even with moderate compression, we achieve 0.87 classification accuracy in distinguishing between transitive and intransitive English verbs, given only tag $t_i$.

**Stem** When we compress ELMo embeddings to $k$ discrete tags, the semantic information must be squeezed out because $k$ is small. But what about the continuous case? In order to verify that semantic information is excluded, we train a classifier that predicts the stem of word token $i$ from its mean tag vector $\mathbb{E}[T_i]$. We expect "player" and "buyer" to have similar compressed vectors, because they share syntactic roles, but we should *fail*

[9] Both can serve as arguments of verbs and prepositions. Both can be modified by determiners and adjectives, giving rise to proper NPs like "The Daily Tribune."

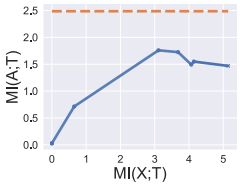


| embeddings | Arabic | English | Spanish | French | Hindi | Italian | Portuguese | Russian | Chinese |
|---|---|---|---|---|---|---|---|---|---|
| H($A$) | 2.016 | 2.486 | 2.345 | 2.206 | 2.247 | 2.291 | 2.306 | 2.131 | 2.195 |
| ELMo0 | 67.2% | 74.2% | 75.7% | 79.6% | 70.1% | 77.9% | 76.5% | 73.2% | 57.3% |
| ELMo1 | 67.2% | 76.1% | 71.7% | 78.0% | 70.5% | 78.1% | 72.3% | 73.8% | 59.8% |
| ELMo2 | 63.8% | 71.0% | 79.7% | 78.7% | 67.2% | 74.5% | 75.3% | 72.2% | 59.4% |

Figure 4: Graph at left: I($A;T$) vs. I($X;T$) in English (in units of nats per token). Table at right: how well the discrete specialized tags predict gold POS tags for 9 languages. The H($A$) row is the entropy (in nats per token) of the gold POS tags in the test data corpus, which is an upper bound for I($A;T$). The remaining rows report the percentage I($A;T$)/H($A$).

to predict that they have different stems "play" and "buy." The classifier is a feedforward neural network with *tanh* activation function, and the last layer is a softmax over the stem vocabulary. In the English treebank, we take the word lemma in UD treebank and use the NLTK library (Bird et al., 2009) to stem each lemma token. Our result (Appendix E in the appendix) suggests that more compression destroys stem information, as hoped. With light compression, the error rate on stem prediction can be below 15%. With moderate compression $\beta = 0.01$, the error rate is 89% for ELMo layer 2 and 66% for ELMo layer 1. Other languages show the same pattern, as shown in Appendix E in the appendix. Thus, moderate and heavy compression indeed squeeze out semantic information.

### 6.3 Annealing of Discrete Tags

Deterministic annealing (Rose, 1998; Friedman et al., 2001) is a method that gradually decreases $\beta$ during training of IB. Each token $i$ has a stochastic distribution over the possible tags $\{1, \ldots, k\}$. This can be regarded as a soft clustering where each token is fractionally associated with each of the $k$ clusters. With high $\beta$, the optimal solution turns out to assign to all tokens an identical distribution over clusters, for a mutual information of 0. Since all clusters then have the same membership, this is equivalent to having a single cluster. As we gradually reduce $\beta$, the cluster eventually splits. Further reduction of $\beta$ leads to recursive splitting, yielding a hierarchical clustering of tokens (Appendix A).

We apply deterministic annealing to the English dataset, and the resulting hierarchical structure reflects properties of English syntax. At the top of the hierarchy, the model places nouns, adjectives, adverbs, and verbs in different clusters. At lower levels, the anaphors ("yourself," "herself" ...), possessive pronouns ("his," "my," "their" ...), accusative-case pronouns ("them," "me," "him," "myself" ...), and nominative-case pronouns ("I," "they," "we" ...) each form a cluster, as do the *wh*-words ("why," "how," "which," "who," "what," ...).

## 7 Engineering Evaluation

As we noted in §1, learning how to compress ELMo's tags for a given task is a fast alternative to fine-tuning all the ELMo parameters. We find that indeed, training a compression method to keep only the relevant information does improve our generalization performance on the parsing task.

We compare 6 different token representations according to the test accuracy of a dependency parser trained to use them. The same training data is used to jointly train the parser and the token encoder that produces the parser's input representations.

**Continuous tags:**

**`Iden`** is an baseline model that leaves the ELMo embeddings uncompressed, so $d = 1024$.

**`PCA`** is a baseline that simply uses Principal Components Analysis to reduce the dimensionality to $d = 256$. Again, this is not task-specific.

**`MLP`** is another deterministic baseline that uses a multi-layer perceptron (as in Dozat and Manning (2016)) to reduce the dimensionality to $d = 256$ in a task-specific and nonlinear way. This is identical to our continuous VIB method except that the variance of the output Gaussians is fixed to 0, so that the $d$ dimensions are fully informative.

**`VIBc`** uses our stochastic encoder, still with $d = 256$. The average amount of stochastic noise is controlled by $\beta$, which is tuned per-language on dev data.

**Discrete tags:**

**`POS`** is a baseline that uses the $k \leq 17$ gold POS tags from the UD dataset.

**`VIBd`** is our stochastic method with $k = 64$ tags. To compare fairly with **`POS`**, we pick a $\beta$ value for each language such that $\mathrm{H}(T_i \mid X_i) \approx \mathrm{H}(A_i \mid X_i)$.

**Runtime.** Our VIB approach is quite fast. With minibatching on a single GPU, it is able to train on 10,000 sentences in 100 seconds, per epoch.

| Models | Arabic | Hindi | English | French | Spanish | Portuguese | Russian | Chinese | Italian |
|---|---|---|---|---|---|---|---|---|---|
| Iden | 0.751 | **0.870** | 0.824 | 0.784 | 0.808 | 0.813 | 0.783 | 0.709 | **0.863** |
| PCA | 0.743 | **0.866** | 0.823 | 0.749 | 0.802 | 0.808 | 0.777 | 0.697 | 0.857 |
| MLP | 0.759 | **0.871** | 0.839 | 0.816 | **0.835** | 0.821 | 0.800 | 0.734 | **0.867** |
| VIBc | **0.779** | **0.866** | **0.851** | **0.828** | **0.837** | **0.836** | **0.814** | **0.754** | **0.867** |
| POS | 0.652 | 0.713 | 0.712 | 0.718 | **0.739** | **0.743** | **0.662** | 0.510 | 0.779 |
| VIBd | **0.672** | **0.736** | **0.742** | **0.723** | **0.725** | 0.710 | **0.651** | **0.591** | **0.781** |

Table 2: Parsing accuracy of 9 languages (LAS). Black rows use continuous tags; gray rows use discrete tags (which does worse). In each column, the best score for each color is boldfaced, along with all results of that color that are not significantly worse (paired permutation test, $p < 0.05$). These results use only ELMo layer 1; results from all layers are shown in Table 3 in the appendix, for both LAS and UAS metrics.

**Analysis.** Table 2 shows the test accuracies of these parsers, using the standard training/development/test split for each UD language.

In the continuous case, the VIB representation outperforms all three baselines in 8 of 9 languages, and is not significantly worse in the 9th language (Hindi). In short, our VIB joint training generalizes better to test data. This is because the training objective (2) includes terms that focus on the parsing task and also regularize the representations.

In the discrete case, the VIB representation outperforms gold POS tags (at the same level of granularity) in 6 of 9 languages, and of the other 3, it is not significantly worse in 2. This suggests that our learned discrete tag set could be an improved alternative to gold POS tags (cf. Klein and Manning, 2003) when a discrete tag set is needed for speed.

## 8 Related Work

Much recent NLP literature examines syntactic information encoded by deep models (Linzen et al., 2016) and more specifically, by powerful unsupervised word embeddings. Hewitt and Manning (2019) learn a linear projection from the embedding space to predict the distance between two words in a parse tree. Peters et al. (2018b) and Goldberg (2019) assess the ability of BERT and ELMo directly on syntactic NLP tasks. Tenney et al. (2019) extract information from the contextual embeddings by self-attention pooling within a span of word embeddings.

The IB framework was first used in NLP to cluster distributionally similar words (Pereira et al., 1993). In cognitive science, it has been used to argue that color-naming systems across languages are nearly optimal (Zaslavsky et al., 2018). In machine learning, IB provides an information-theoretic perspective to explain the performance of deep neural networks (Tishby and Zaslavsky, 2015).

The VIB method makes use of variational upper and lower bounds on mutual information. An alternative lower bound was proposed by Poole et al. (2019), who found it to work better empirically.

## 9 Conclusion and Future Work

In this paper, we have proposed two ways to syntactically compress ELMo word token embeddings, using variational information bottleneck. We automatically induce stochastic discrete tags that correlate with gold POS tags but are as good or better for parsing. We also induce stochastic continuous token embeddings (each is a Gaussian distribution over $\mathbb{R}^d$) that forget non-syntactic information captured by ELMo. These stochastic vectors yield improved parsing results, in a way that simpler dimensionality reduction methods do not. They also transfer to the problem of predicting gold POS tags, which were not used in training.

One could apply the same training method to compress the ELMo or BERT token sequence $x$ for other tasks. All that is required is a model-specific decoder $q_\phi(y \mid t)$. For example, in the case of sentiment analysis, the approach should preserve only sentiment information, discarding most of the syntax. One possibility that does not require supervised data is to create artificial tasks, such as reproducing the input sentence or predicting missing parts of the input (such as affixes and function words). In this case, the latent representations would be essentially generative, as in the variational autoencoder (Kingma and Welling, 2013).

## Acknowledgments

This work was supported by the National Science Foundation under Grant No. 1718846 and by a Provost's Undergraduate Research Award to the first author. The Maryland Advanced Research Computing Center provided computing facilities. We thank the anonymous reviewers and Hongyuan Mei for helpful comments.

## References


Alexander A. Alemi, Ian Fischer, Joshua V. Dillon, and Kevin Murphy. 2016. Deep variational information bottleneck. *Proceedings of the International Conference on Learning Representations (ICLR)*, abs/1612.00410.

T.W. Anderson. 2003. *An Introduction to Multivariate Statistical Analysis*. Wiley Series in Probability and Statistics. Wiley.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*, 1st edition. O'Reilly Media, Inc.

Wanxiang Che, Yijia Liu, Yuxuan Wang, Bo Zheng, and Ting Liu. 2018. Towards better UD parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 55–64, Brussels, Belgium. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Timothy Dozat and Christopher D. Manning. 2016. Deep biaffine attention for neural dependency parsing. *CoRR*, abs/1611.01734.

Jack Edmonds. 1966. Optimum Branchings. *Journal of Research of the National Bureau of Standards*.

Nir Friedman, Ori Mosenzon, Noam Slonim, and Naftali Tishby. 2001. Multivariate information bottleneck. In *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, UAI '01, pages 152–161, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. AllenNLP: A Deep Semantic Natural Language Processing Platform.

Yoav Goldberg. 2019. Assessing BERT's syntactic abilities. *CoRR*, abs/1901.05287.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with Gumbel-softmax. *International Conference on Learning Representations*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Diederik P. Kingma and Max Welling. 2013. Auto-encoding variational Bayes. *Proceedings of the International Conference on Learning Representations (ICLR)*.

D. Klein and C. D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*.

Terry Koo, Amir Globerson, Xavier Carreras Pérez, and Michael Collins. 2007. Structured prediction models via the matrix-tree theorem. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 141–150.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The concrete distribution: A continuous relaxation of discrete random variables. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Ryan McDonald and Giorgio Satta. 2007. On the complexity of non-projective data-driven dependency parsing. In *Proceedings of the 10th International Conference on Parsing Technologies*, pages 121–132. Association for Computational Linguistics.

Joakim Nivre et al. 2018. Universal dependencies 2.3. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Fernando Pereira, Naftali Tishby, and Lillian Lee. 1993. Distributional clustering of English words. In *Proceedings of the 31st Annual Meeting of Association for Computational Linguistics*, pages 183–190. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018a. Deep contextualized word representations. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*.

Matthew E. Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. 2018b. Dissecting contextual word embeddings: Architecture and representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1499–1509.

Ben Poole, Sherjil Ozair, Aäron van den Oord, Alexander A. Alemi, and George Tucker. 2019. On variational bounds of mutual information. *CoRR*, abs/1905.06922.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*.

Kenneth Rose. 1998. Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. *Proceedings of the IEEE*, 80:2210–2239.

David A Smith and Noah A Smith. 2007. Probabilistic models of nonprojective dependency trees. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 132–140.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R. Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R. Bowman, Dipanjan Das, and Ellie Pavlick. 2019. What do you learn from context? Probing for sentence structure in contextualized word representations.

Naftali Tishby, Fernando C. Pereira, and William Bialek. 2000. The information bottleneck method. *arXiv preprint physics/0004057*.

Naftali Tishby and Noga Zaslavsky. 2015. Deep learning and the information bottleneck principle. *2015 IEEE Information Theory Workshop (ITW)*, pages 1–5.

L. J. P. van der Maaten and G. E. Hinton. 2008. Visualizing high-dimensional data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605.

Noga Zaslavsky, Charles Kemp, Terry Regier, and Naftali Tishby. 2018. Efficient human-like semantic representations via the information bottleneck principle. *CoRR*, abs/1808.03353.

# Deep Contextualized Word Embeddings in Transition-Based and Graph-Based Dependency Parsing – A Tale of Two Parsers Revisited*

**Artur Kulmizev Miryam de Lhoneux Johannes Gontrum Elena Fano Joakim Nivre**
Department of Linguistics and Philology, Uppsala University
{artur.kulmizev,miryam.de_lhoneux,joakim.nivre}@lingfil.uu.se
{johannes.gontrum.4608,elena.fano.3249}@student.uu.se

## Abstract

Transition-based and graph-based dependency parsers have previously been shown to have complementary strengths and weaknesses: transition-based parsers exploit rich structural features but suffer from error propagation, while graph-based parsers benefit from global optimization but have restricted feature scope. In this paper, we show that, even though some details of the picture have changed after the switch to neural networks and continuous representations, the basic trade-off between rich features and global optimization remains essentially the same. Moreover, we show that deep contextualized word embeddings, which allow parsers to pack information about global sentence structure into local feature representations, benefit transition-based parsers more than graph-based parsers, making the two approaches virtually equivalent in terms of both accuracy and error profile. We argue that the reason is that these representations help prevent search errors and thereby allow transition-based parsers to better exploit their inherent strength of making accurate local decisions. We support this explanation by an error analysis of parsing experiments on 13 languages.

## 1 Introduction

For more than a decade, research on data-driven dependency parsing has been dominated by two approaches: transition-based parsing and graph-based parsing (McDonald and Nivre, 2007, 2011). Transition-based parsing reduces the parsing task to scoring single parse actions and is often combined with local optimization and greedy search algorithms. Graph-based parsing decomposes parse trees into subgraphs and relies on global optimization and exhaustive (or at least non-greedy) search to find the best tree. These radically different approaches often lead to comparable parsing accuracy, but with distinct error profiles indicative of their respective strengths and weaknesses, as shown by McDonald and Nivre (2007, 2011).

In recent years, dependency parsing, like most of NLP, has shifted from linear models and discrete features to neural networks and continuous representations. This has led to substantial accuracy improvements for both transition-based and graph-based parsers and raises the question whether their complementary strengths and weaknesses are still relevant. In this paper, we replicate the analysis of McDonald and Nivre (2007, 2011) for neural parsers. In addition, we investigate the impact of deep contextualized word representations (Peters et al., 2018; Devlin et al., 2019) for both types of parsers.

Based on what we know about the strengths and weaknesses of the two approaches, we hypothesize that deep contextualized word representations will benefit transition-based parsing more than graph-based parsing. The reason is that these representations make information about global sentence structure available locally, thereby helping to prevent search errors in greedy transition-based parsing. The hypothesis is corroborated in experiments on 13 languages, and the error analysis supports our suggested explanation. We also find that deep contextualized word representations improve parsing accuracy for longer sentences, both for transition-based and graph-based parsers.

## 2 Two Models of Dependency Parsing

After playing a marginal role in NLP for many years, dependency-based approaches to syntactic parsing have become mainstream during the last fifteen years. This is especially true if we consider languages other than English, ever since the influ-

*We gratefully acknowledge the inspiration for our subtitle in the seminal paper by Zhang and Clark (2008).



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2755–2768, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

ential CoNLL shared tasks on dependency parsing in 2006 (Buchholz and Marsi, 2006) and 2007 (Nivre et al., 2007) with data from 19 languages.

The transition-based approach to dependency parsing was pioneered by Yamada and Matsumoto (2003) and Nivre (2003), with inspiration from history-based parsing (Black et al., 1992) and data-driven shift-reduce parsing (Veenstra and Daelemans, 2000). The idea is to reduce the complex parsing task to the simpler task of predicting the next parsing action and to implement parsing as greedy search for the optimal sequence of actions, guided by a simple classifier trained on local parser configurations. This produces parsers that are very efficient, often with linear time complexity, and which can benefit from rich non-local features defined over parser configurations but which may suffer from compounding search errors.

The graph-based approach to dependency parsing was developed by McDonald et al. (2005a,b), building on earlier work by Eisner (1996). The idea is to score dependency trees by a linear combination of scores of local subgraphs, often single arcs, and to implement parsing as exact search for the highest scoring tree under a globally optimized model. These parsers do not suffer from search errors but parsing algorithms are more complex and restrict the scope of features to local subgraphs.

The terms *transition-based* and *graph-based* were coined by McDonald and Nivre (2007, 2011), who performed a contrastive error analysis of the two top-performing systems in the CoNLL 2006 shared task on multilingual dependency parsing: MaltParser (Nivre et al., 2006) and MSTParser (McDonald et al., 2006), which represented the state of the art in transition-based and graph-based parsing, respectively, at the time. Their analysis shows that, despite having almost exactly the same parsing accuracy when averaged over 13 languages, the two parsers have very distinctive error profiles. MaltParser is more accurate on short sentences, on short dependencies, on dependencies near the leaves of the tree, on nouns and prounouns, and on subject and object relations. MSTParser is more accurate on long sentences, on long dependencies, on dependencies near the root of the tree, on verbs, and on coordination relations and sentence roots.

McDonald and Nivre (2007, 2011) argue that these patterns can be explained by the complementary strengths and weaknesses of the systems. The

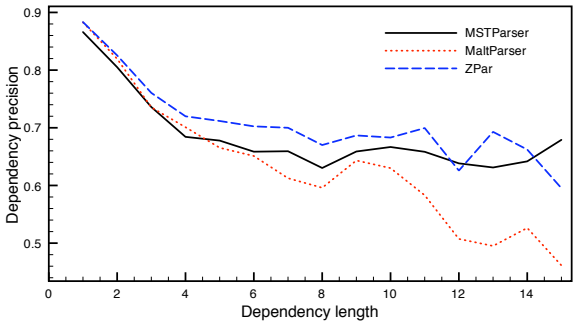


Figure 1: Labeled precision by dependency length for MST (global–exhaustive–graph), Malt (local–greedy–transition) and ZPar (global–beam–transition). From Zhang and Nivre (2012).

transition-based MaltParser prioritizes rich structural features, which enable accurate disambiguation in local contexts, but is limited by a locally optimized model and greedy algorithm, resulting in search errors for structures that require longer transition sequences. The graph-based MSTParser benefits from a globally optimized model and exact inference, which gives a better analysis of global sentence structure, but is more restricted in the features it can use, which limits its capacity to score local structures accurately.

Many of the developments in dependency parsing during the last decade can be understood in this light as attempts to mitigate the weaknesses of traditional transition-based and graph-based parsers without sacrificing their strengths. This may mean evolving the model structure through new transition systems (Nivre, 2008, 2009; Kuhlmann et al., 2011) or higher-order models for graph-based parsing (McDonald and Pereira, 2006; Carreras, 2007; Koo and Collins, 2010); it may mean exploring alternative learning strategies, in particular for transition-based parsing, where improvements have been achieved thanks to global structure learning (Zhang and Clark, 2008; Zhang and Nivre, 2011; Andor et al., 2016) and dynamic oracles (Goldberg and Nivre, 2012, 2013); it may mean using alternative search strategies, such as transition-based parsing with beam search (Johansson and Nugues, 2007; Titov and Henderson, 2007; Zhang and Clark, 2008) or exact search (Huang and Sagae, 2010; Kuhlmann et al., 2011) or graph-based parsing with heuristic search to cope with the complexity of higher-order models, especially for non-projective parsing (McDonald and Pereira, 2006; Koo et al., 2010; Zhang and McDonald, 2012); or it may mean hybrid or en-

semble systems (Sagae and Lavie, 2006; Nivre and McDonald, 2008; Zhang and Clark, 2008; Bohnet and Kuhn, 2012). A nice illustration of the impact of new techniques can be found in Zhang and Nivre (2012), where an error analysis along the lines of McDonald and Nivre (2007, 2011) shows that a transition-based parser using global learning and beam search (instead of local learning and greedy search) performs on par with graph-based parsers for long dependencies, while retaining the advantage of the original transition-based parsers on short dependencies (see Figure 1).

Neural networks for dependency parsing, first explored by Titov and Henderson (2007) and Attardi et al. (2009), have come to dominate the field during the last five years. While this has dramatically changed learning architectures and feature representations, most parsing models are still either transition-based (Chen and Manning, 2014; Dyer et al., 2015; Weiss et al., 2015; Andor et al., 2016; Kiperwasser and Goldberg, 2016) or graph-based (Kiperwasser and Goldberg, 2016; Dozat and Manning, 2017). However, more accurate feature learning using continuous representations and nonlinear models has allowed parsing architectures to be simplified. Thus, most recent transition-based parsers have moved back to local learning and greedy inference, seemingly without losing accurracy (Chen and Manning, 2014; Dyer et al., 2015; Kiperwasser and Goldberg, 2016). Similarly, graph-based parsers again rely on first-order models and obtain no improvements from using higher-order models (Kiperwasser and Goldberg, 2016; Dozat and Manning, 2017).

The increasing use of neural networks has also led to a convergence in feature representations and learning algorithms for transition-based and graph-based parsers. In particular, most recent systems rely on an encoder, typically in the form of a BiLSTM, that provides contextualized representations of the input words as input to the scoring of transitions – in transition-based parsers – or of dependency arcs – in graph-based parsers. By making information about the *global* sentence context available in *local* word representations, this encoder can be assumed to mitigate error propagation for transition-based parsers and to widen the feature scope beyond individual word pairs for graph-based parsers. For both types of parsers, this also obviates the need for complex structural feature templates, as recently shown by Falenska and Kuhn (2019). We should therefore expect neural transition-based and graph-based parsers to be not only more accurate than their non-neural counterparts but also more similar to each other in their error profiles.

## 3 Deep Contextualized Word Representations

Neural parsers rely on vector representations of words as their primary input, often in the form of pretrained word embeddings such as word2vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014), or fastText (Bojanowski et al., 2016), which are sometimes extended with character-based representations produced by recurrent neural networks (Ballesteros et al., 2015). These techniques assign a single static representation to each word type and therefore cannot capture context-dependent variation in meaning and syntactic behavior.

By contrast, deep contextualized word representations encode words with respect to the sentential context in which they appear. Like word embeddings, such models are typically trained with a language-modeling objective, but yield sentence-level tensors as representations, instead of single vectors. These representations are typically produced by transferring a model's entire feature encoder – be it a BiLSTM (Hochreiter and Schmidhuber, 1997) or Transformer (Vaswani et al., 2017) – to a target task, where the dimensionality of the tensor $\mathbf{S}$ is typically $\mathbf{S} \in \mathbb{R}^{N \times L \times D}$ for a sentence of length $N$, an encoder with $L$ layers, and word-level vectors of dimensionality $D$. The advantage of such models, compared to the parser-internal encoders discussed in the previous section, is that they not only produce contextualized representations but do so over several layers of abstraction, as captured by the model's different layers, and are pre-trained on corpora much larger than typical treebanks.

Deep contextualized embedding models have proven to be adept at a wide array of NLP tasks, achieving state-of-the-art performance in standard Natural Language Understanding (NLU) benchmarks, such as GLUE (Wang et al., 2019). Though many such models have been proposed, we adopt the two arguably most popular ones for our experiments: ELMo and BERT. Both models have previously been used for dependency parsing (Che et al., 2018; Jawahar et al., 2018; Lim et al., 2018;

Kondratyuk, 2019; Schuster et al., 2019), but there has been no systematic analysis of their impact on transition-based and graph-based parsers.

### 3.1 ELMo

ELMo is a deep contextualized embedding model proposed by Peters et al. (2018), which produces sentence-level representations yielded by a multi-layer BiLSTM language model. ELMo is trained with a standard language-modeling objective, in which a BiLSTM reads a sequence of $N$ learned context-independent embeddings $\mathbf{w}_1, \ldots, \mathbf{w}_N$ (obtained via a character-level CNN) and produces a context-dependent representation $\mathbf{h}_{j,k} = \mathbf{BiLSTM}(\mathbf{w}_{1:N}, k)$, where $j$ $(1 \leq j \leq L)$ is the BiLSTM layer and $k$ is the index of the word in the sequence. The output of the last layer $\mathbf{h}_{L,k}$ is then employed in conjunction with a softmax layer to predict the next token at $k+1$.

The simplest way of transferring ELMo to a downstream task is to encode the input sentence $S = w_1, \ldots, w_N$ by extracting the representations from the BiLSTM at layer $L$ for each token $w_k \in S$: $\mathbf{h}_{L,1}, \ldots, \mathbf{h}_{L,N},$. However, Peters et al. (2018) posit that the best way to take advantage of ELMo's representational power is to compute a linear combination of BiLSTM layers:

$$\mathbf{ELMo}_k = \gamma \sum_{j=0}^{L} s_j \mathbf{h}_{j,k} \quad (1)$$

where $s_j$ is a softmax-normalized task-specific parameter and $\gamma$ is a task-specific scalar. Peters et al. (2018) demonstrate that this scales the layers of linguistic abstraction encoded by the BiLSTM for the task at hand.

### 3.2 BERT

BERT (Devlin et al., 2019) is similar to ELMo in that it employs a language-modeling objective over unannotated text in order to produce deep contextualized embeddings. However, BERT differs from ELMo in that, in place of a BiLSTM, it employs a bidirectional Transformer (Vaswani et al., 2017), which, among other factors, carries the benefit of learning potential dependencies between words directly. This lies in contrast to recurrent models, which may struggle to learn correspondences between constituent signals when the time-lag between them is long (Hochreiter et al., 2001). For a token $w_k$ in sentence $S$, BERT's input representation is composed by summing a word embedding $\mathbf{x}_k$, a position embedding $\mathbf{i}_k$, and a WordPiece embedding $\mathbf{s}_k$ (Wu et al., 2016): $\mathbf{w}_k = \mathbf{x}_k + \mathbf{i}_k + \mathbf{s}_k$.

Each $\mathbf{w}_k \in S$ is passed to an $L$-layered Bi-Transformer, which is trained with a masked language modeling objective (i.e., randomly masking a percentage of input tokens and only predicting said tokens). For use in downstream tasks, Devlin et al. (2019) propose to extract the Transformer's encoding of each token $w_k \in S$ at layer $L$, which effectively produces $\mathbf{BERT}_k$.

## 4 Hypotheses

Based on our discussion in Section 2, we assume that transition-based and graph-based parsers still have distinctive error profiles due to the basic trade-off between rich structural features, which allow transition-based parsers to make accurate local decisions, and global learning and exact search, which give graph-based parsers an advantage with respect to global sentence structure. At the same time, we expect the differences to be less pronounced than they were ten years ago because of the convergence in neural architectures and feature representations. But how will the addition of deep contextualized word representations affect the behavior of the two parsers?

Given recent recent work showing that deep contextualized word representations incorporate rich information about syntactic structure (Goldberg, 2019; Liu et al., 2019; Tenney et al., 2019; Hewitt and Manning, 2019), we hypothesize that transition-based parsers have most to gain from these representations because it will improve their capacity to make decisions informed by global sentence structure and therefore reduce the number of search errors. Our main hypothesis can be stated as follows:

> Deep contextualized word representations are more effective at reducing errors in transition-based parsing than in graph-based parsing.

If this holds true, then the analysis of McDonald and Nivre (2007, 2011) suggests that the differential error reduction should be especially visible on phenomena such as:

1. longer dependencies,
2. dependencies closer to the root,
3. certain parts of speech,
4. certain dependency relations,
5. longer sentences.

The error analysis will consider all these factors as well as non-projective dependencies.

## 5 Experimental Setup

### 5.1 Parsing Architecture

To be able to compare transition-based and graph-based parsers under equivalent conditions, we use and extend UUParser[1] (de Lhoneux et al., 2017a; Smith et al., 2018a), an evolution of bist-parser (Kiperwasser and Goldberg, 2016), which supports transition-based and graph-based parsing with a common infrastructure but different scoring models and parsing algorithms.

For an input sentence $S = w_1, \ldots, w_N$, the parser creates a sequence of vectors $\mathbf{w}_{1:N}$, where the vector $\mathbf{w}_k = \mathbf{x}_k \circ \textsc{BiLstm}(c_{1:M})$ representing input word $w_k$ is the concatenation of a pretrained word embedding $\mathbf{x}_k$ and a character-based embedding $\textsc{BiLstm}(c_{1:M})$ obtained by running a BiLSTM over the character sequence $c_{1:M}$ of $w_k$. Finally, each input element is represented by a BiLSTM vector, $\mathbf{h}_k = \textsc{BiLstm}(\mathbf{w}_{1:N}, k)$.

In transition-based parsing, the BiLSTM vectors are input to a multi-layer perceptron (MLP) for scoring transitions, using the arc-hybrid transition system from Kuhlmann et al. (2011) extended with a SWAP transition to allow the construction of non-projective dependency trees (Nivre, 2009; de Lhoneux et al., 2017b). The scoring is based on the top three words on the stack and the first word of the buffer, and the input to the MLP includes the BiLSTM vectors for these words as well as their leftmost and rightmost dependents (up to 12 words in total).

In graph-based parsing, the BiLSTM vectors are input to an MLP for scoring all possible dependency relations under an arc-factored model, meaning that only the vectors corresponding to the head and dependent are part of the input (2 words in total). The parser then extracts a maximum spanning tree over the score matrix using the Chu-Liu-Edmonds (CLE) algorithm[2] (Edmonds, 1967) which allows us to construct non-projective trees.

It is important to note that, while we acknowledge the existence of graph-based parsers that outperform the implementation of Kiperwasser and Goldberg (2016), such models do not meet our criteria for systematic comparison. The parser by Dozat et al. (2017) is very similar, but employs the MLP as a further step in the featurization process prior to scoring via a biaffine classifier. To keep the comparison as exact as possible, we forego comparing our transition-based systems to the Dozat et al. (2017) parser (and its numerous modifications). In addition, preliminary experiments showed that our chosen graph-based parser outperforms its transition-based counterpart, which was itself competitive in the CoNLL 2018 shared task (Zeman et al., 2018).

### 5.2 Input Representations

In our experiments, we evaluate three pairs of systems – differing only in their input representations. The first is a baseline that represents tokens by $\mathbf{w}_k = \mathbf{x}_k \circ \textsc{BiLstm}(c_{1:M})$, as described in Section 5.1. The word embeddings $\mathbf{x}_k$ are initialized via pretrained fastText vectors ($\mathbf{x}_k \in \mathbb{R}^{300}$) (Grave et al., 2018), which are updated for the parsing task. We term these transition-based and graph-based baselines TR and GR.

For the ELMo experiments, we make use of pretrained models provided by Che et al. (2018), who train ELMo on 20 million words randomly sampled from raw WikiDump and Common Crawl datasets for 44 languages. We encode each gold-segmented sentence in our treebank via the ELMo model for that language, which yields a tensor $\mathbf{S}_{\text{ELMo}} = \mathbb{R}^{N \times L \times D}$, where $N$ is the number of words in the sentence, $L = 3$ is the number of ELMo layers, and $D = 1024$ is the ELMo vector dimensionality. Following Peters et al. (2018) (see Eq. 1), we learn a linear combination and a task-specific $\gamma$ of each token's ELMo representation, which yields a vector $\mathbf{ELMo}_k \in \mathbb{R}^{1024}$. We then concatenate this vector with $\mathbf{w}_k$ and pass it to the BiLSTM. We call the transition-based and graph-based systems enhanced with ELMo TR+E and GR+E.

For the BERT experiments, we employ the pretrained multilingual cased model provided by Google,[3][4] which is trained on the concatenation of WikiDumps for the top 104 languages with the largest Wikipedias.[5] The model's parameters feature a 12-layer transformer trained with 768 hid-

[1] https://github.com/UppsalaNLP/uuparser

[2] We use the implementation from Qi et al. (2018).

[3] https://github.com/google-research/bert

[4] Except for Chinese, for which we make use of a separate, pretrained model.

[5] See sorted list here: https://meta.wikimedia.org/wiki/List_of_Wikipedias

den units and 12 self-attention heads. In order to obtain a word-level vector for each token in a sentence, we experimented with a variety of representations: namely, concatenating each transformer layer's word representation into a single vector $\mathbf{w}_{concat} \in \mathbb{R}^{768*12}$, employing the last layer's representation, or learning a linear combination over a range of layers, as we do with ELMo (e.g., via Eq. 1). In a preliminary set of experiments, we found that the latter approach over layers 4–8 consistently yielded the best results, and thus chose to adopt this method going forward. Regarding tokenization, we select the vector for the first subword token, as produced by the native BERT tokenizer. Surprisingly, this gave us better results than averaging subword token vectors in a preliminary round of experiments. Like with the ELMo representations, we concatenate each BERT vector $\mathbf{BERT}_k \in \mathbb{R}^{768}$ with $\mathbf{w}_k$ and pass it to the respective TR+B and GR+B parsers.

It is important to note that while the ELMo models we work with are *monolingual*, the BERT model is *multilingual*. In other words, while the standalone ELMo models were trained on the tokenized WikiDump and CommonCrawl for each language respectively, the BERT model was trained only on the former, albeit simultaneously for 104 languages. This means that the models are not strictly comparable, and it is an interesting question whether either of the models has an advantage in terms of training regime. However, since our purpose is not to compare the two models but to study their impact on parsing, we leave this question for future work.

### 5.3 Language and Treebank Selection

For treebank selection, we rely on the criteria proposed by de Lhoneux et al. (2017c) and adapted by Smith et al. (2018b) to have languages from different language families, with different morphological complexity, different scripts and character set sizes, different training sizes and domains, and with good annotation quality. This gives us 13 treebanks from UD v2.3 (Nivre et al., 2018), information about which is shown in Table 1.

### 5.4 Parser Training and Evaluation

In all experiments, we train parsers with default settings[6] for 30 epochs and select the model with the best labeled attachment score on the dev set. For each combination of model and training set, we repeat this procedure three times with different random seeds, apply the three selected models to the test set, and report the average result.

| Language | Treebank | Family | Order | Train |
|---|---|---|---|---|
| Arabic | PADT | non-IE | VSO | 6.1k |
| Basque | BDT | non-IE | SOV | 5.4k |
| Chinese | GSD | non-IE | SVO | 4.0k |
| English | EWT | IE | SVO | 12.5k |
| Finnish | TDT | non-IE | SVO | 12.2k |
| Hebrew | HTB | non-IE | SVO | 5.2k |
| Hindi | HDTB | IE | SOV | 13.3k |
| Italian | ISDT | IE | SVO | 13.1k |
| Japanese | GSD | non-IE | SOV | 7.1k |
| Korean | GSD | non-IE | SOV | 4.4k |
| Russian | SynTagRus | IE | SVO | 48.8k |
| Swedish | Talbanken | IE | SVO | 4.3k |
| Turkish | IMST | non-IE | SOV | 3.7k |

Table 1: Languages and treebanks used in experiments. Family = Indo-European (IE) or not. Order = dominant word order according to WALS (Haspelmath et al., 2005). Train = number of training sentences.

### 5.5 Error Analysis

In order to conduct an error analysis along the lines of McDonald and Nivre (2007, 2011), we extract all sentences from the smallest development set in our treebank sample (Hebrew HTB, 484 sentences) and sample the same number of sentences from each of the other development sets (6,292 sentences in total). For each system, we then extract parses of these sentences for the three training runs with different random seeds (18,876 predictions in total). Although it could be interesting to look at each language separately, we follow McDonald and Nivre (2007, 2011) and base our main analysis on all languages together to prevent data sparsity for longer dependencies, longer sentences, etc.[7]

## 6 Results and Discussion

Table 2 shows labeled attachment scores for the six parsers on all languages, averaged over three training runs with random seeds. The results clearly corroborate our main hypothesis. While ELMo and BERT provide significant improvements for both transition-based and graph-based

[6] All hyperparameters are specified in the supplementary material (Part A).

[7] The supplementary material contains tables for the error analysis (Part B) and graphs for each language (Part C).

| Language | TR | GR | TR+E | GR+E | TR+B | GR+B |
|---|---|---|---|---|---|---|
| Arabic | 79.1 | 79.9 | 82.0 | 81.7 | 81.9 | 81.8 |
| Basque | 73.6 | 77.6 | 80.1 | 81.4 | 77.9 | 79.8 |
| Chinese | 75.3 | 76.7 | 79.8 | 80.4 | 83.7 | 83.4 |
| English | 82.7 | 83.3 | 87.0 | 86.5 | 87.8 | 87.6 |
| Finnish | 80.0 | 81.4 | 87.0 | 86.6 | 85.1 | 83.9 |
| Hebrew | 81.1 | 82.4 | 85.2 | 85.9 | 85.5 | 85.9 |
| Hindi | 88.4 | 89.6 | 91.0 | 91.2 | 89.5 | 90.8 |
| Italian | 88.0 | 88.2 | 90.9 | 90.6 | 92.0 | 91.7 |
| Japanese | 92.1 | 92.2 | 93.1 | 93.0 | 92.9 | 92.1 |
| Korean | 79.6 | 81.2 | 82.3 | 82.3 | 83.7 | 84.2 |
| Russian | 88.3 | 88.0 | 90.7 | 90.6 | 91.5 | 91.0 |
| Swedish | 80.5 | 81.6 | 86.9 | 86.2 | 87.6 | 86.9 |
| Turkish | 57.8 | 61.2 | 62.6 | 63.8 | 64.2 | 64.9 |
| **Average** | **80.5** | **81.8** | **84.5** | **84.6** | **84.9** | **84.9** |

Table 2: Labeled attachment score on 13 languages for parsing models with and without deep contextualized word representations.

parsers, the magnitude of the improvement is greater in the transition-based case: 3.99 vs. 2.85 for ELMo and 4.47 vs. 3.13 for BERT. In terms of error reduction, this corresponds to 21.1% vs. 16.5% for ELMo and 22.5% vs. 17.4% for BERT. The differences in error reduction are statistically significant at $\alpha = 0.01$ (Wilcoxon).

Although both parsing accuracy and absolute improvements vary across languages, the overall trend is remarkably consistent and the transition-based parser improves more with both ELMo and BERT for every single language. Furthermore, a linear mixed effect model analysis reveals that, when accounting for language as a random effect, there are no significant interactions between the improvement of each model (over its respective baseline) and factors such as language family (IE vs. non-IE), dominant word order, or number of training sentences. In other words, the improvements for both parsers seem to be largely independent of treebank-specific factors. Let us now see to what extent they can be explained by the error analysis.

### 6.1 Dependency Length

Figure 2 shows labeled F-score for dependencies of different lengths, where the length of a dependency between words $w_i$ and $w_j$ is equal to $|i - j|$ (and with root tokens in a special bin on the far left). For the baseline parsers, we see that the curves diverge with increasing length, clearly indicating that the transition-based parser still suffers

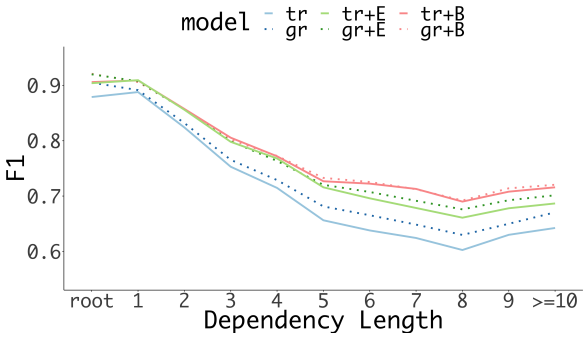


Figure 2: Labeled F-score by dependency length.

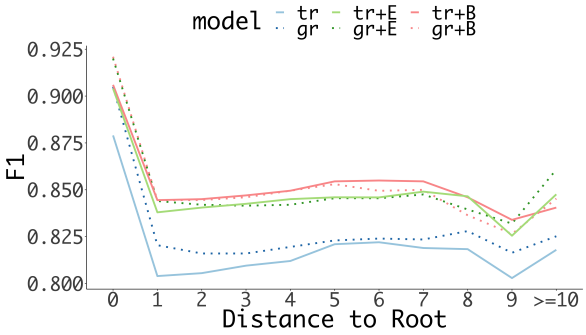


Figure 3: Labeled F-score by distance to root.

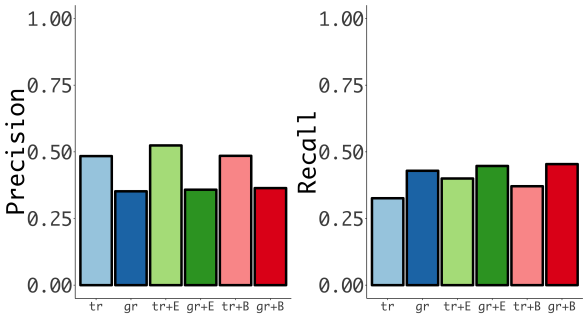


Figure 4: Labeled precision (left) and recall (right) for non-projective dependencies.

from search errors on long dependencies, which require longer transition sequences for their construction. However, the differences are much smaller than in McDonald and Nivre (2007, 2011) and the transition-based parser no longer has an advantage for short dependencies, which is consistent with the BiLSTM architecture providing the parsers with more similar features that help the graph-based parser overcome the limited scope of the first-order model.

Adding deep contextualized word representations clearly helps the transition-based parser to perform better on longer dependencies. For ELMo there is still a discernible difference for dependencies longer than 5, but for BERT the two curves

are almost indistinguishable throughout the whole range. This could be related to the aforementioned intuition that a Transformer captures long dependencies more effectively than a BiLSTM (see Tran et al. (2018) for contrary observations, albeit for different tasks). The overall trends for both baseline and enhanced models are quite consistent across languages, although with large variations in accuracy levels.

### 6.2 Distance to Root

Figure 3 reports labeled F-score for dependencies at different distances from the root of the tree, where distance is measured by the number of arcs in the path from the root. There is a fairly strong (inverse) correlation between dependency length and distance to the root, so it is not surprising that the plots in Figure 3 largely show the mirror image of the plots in Figure 2. For the baseline parsers, the graph-based parser has a clear advantage for dependencies near the root (including the root itself), but the transition-based parser closes the gap with increasing distance.[8] For ELMo and BERT, the curves are much more similar, with only a slight advantage for the graph-based parser near the root and with the transition-based BERT parser being superior from distance 5 upwards. The main trends are again similar across all languages.

### 6.3 Non-Projective Dependencies

Figure 4 shows precision and recall specifically for non-projective dependencies. We see that there is a clear tendency for the transition-based parser to have better precision and the graph-based parser better recall.[9] In other words, non-projective dependencies are more likely to be correct when they are predicted by the transition-based parser using the swap transition, but real non-projective dependencies are more likely to be found by the graph-based parser using a spanning tree algorithm. Interestingly, adding deep contextualized word representations has almost no effect on the graph-based parser,[10] while especially the ELMo embeddings improve both precision and recall for the transition-based parser.

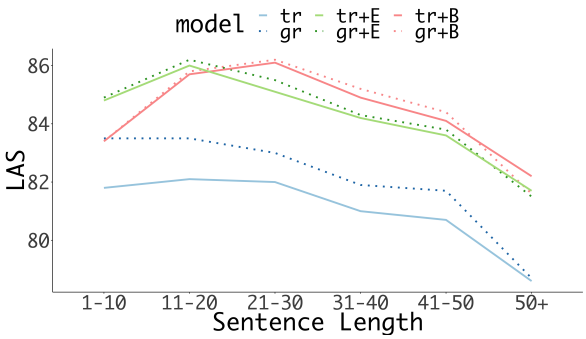


Figure 5: Labeled attachment score by sentence length.

### 6.4 Parts of Speech and Dependency Types

Thanks to the cross-linguistically consistent UD annotations, we can relate errors to linguistic categories more systematically than in the old study. The main impression, however, is that there are very few clear differences, which is again indicative of the convergence between the two parsing approaches. We highlight the most notable differences and refer to the supplementary material (Part B) for the full results.

Looking first at parts of speech, the baseline graph-based parser is slightly more accurate on verbs and nouns than its transition-based counterpart, which is consistent with the old study for verbs but not for nouns. After adding the deep contextualized word representations, both differences are essentially eliminated.

With regard to dependency relations, the baseline graph-based parser has better precision and recall than the baseline transition-based parser for the relations of coordination (conj), which is consistent with the old study, as well as clausal subjects (csubj) and clausal complements (ccomp), which are relations that involve verbs in clausal structures. Again, the differences are greatly reduced in the enhanced parsing models, especially for clausal complements, where the transition-based parser with ELMo representations is even slightly more accurate than the graph-based parser.

### 6.5 Sentence Length

Figure 5 plots labeled attachment score for sentences of different lengths, measured by number of words in bins of 1–10, 11–20, etc. Here we

[8] At the very end, the curves appear to diverge again, but the data is very sparse in this part of the plot.

[9] Incidentally, the same pattern is reported by McDonald and Nivre (2007, 2011), even though the techniques for processing non-projective dependencies are different in that study: pseudo-projective parsing (Nivre and Nilsson, 2005) for the transition-based parser and approximate second-order non-projective parsing (McDonald and Pereira, 2006) for the graph-based parser.

[10] The breakdown per language shows marginal improvements for the enhanced graph-based models on a few languages, canceled out by equally marginal degradations on others.

find the most unexpected results of the study. First of all, although the baseline parsers exhibit the familiar pattern of accuracy decreasing with sentence length, it is not the transition-based but the graph-based parser that is more accurate on short sentences and degrades faster. In other words, although the transition-based parser still seems to suffer from search errors, as shown by the results on dependency length and distance to the root, it no longer seems to suffer from error propagation in the sense that earlier errors make later errors more probable. The most likely explanation for this is the improved training for transition-based parsers using dynamic oracles and aggressive exploration to learn how to behave optimally also in non-optimal configurations (Goldberg and Nivre, 2012, 2013; Kiperwasser and Goldberg, 2016).

Turning to the models with deep contextualized word representations, we find that transition-based and graph-based parsers behave more similarly, which is in line with our hypotheses. However, the most noteworthy result is that accuracy *improves* with increasing sentence length. For ELMo this holds only from 1–10 to 11–20, but for BERT it holds up to 21–30, and even sentences of length 31–40 are parsed with higher accuracy than sentences of length 1–10. A closer look at the breakdown per language reveals that this picture is slightly distorted by different sentence length distributions in different languages. More precisely, high-accuracy languages seem to have a higher proportion of sentences of mid-range length, causing a slight boost in the accuracy scores of these bins, and no single language exhibits exactly the patterns shown in Figure 5. Nevertheless, several languages exhibit an increase in accuracy from the first to the second bin or from the second to the third bin for one or more of the enhanced models (especially the BERT models). And almost all languages show a less steep degradation for the enhanced models, clearly indicating that deep contextualized word representations improve the capacity to parse longer sentences.

## 7 Conclusion

In this paper, we have essentially replicated the study of McDonald and Nivre (2007, 2011) for neural parsers. In the baseline setting, where parsers use pre-trained word embeddings and character representations fed through a BiLSTM, we can still discern the basic trade-off identified in the old study, with the transition-based parser suffering from search errors leading to lower accuracy on long dependencies and dependencies near the root of the tree. However, important details of the picture have changed. The graph-based parser is now as accurate as the transition-based parser on shorter dependencies and dependencies near the leaves of the tree, thanks to improved representation learning that overcomes the limited feature scope of the first order model. And with respect to sentence length, the pattern has actually been reversed, with the graph-based parser being more accurate on short sentences and the transition-based parser gradually catching up thanks to new training methods that prevent error propagation.

When adding deep contextualized word representations, the behavior of the two parsers converge even more, and the transition-based parser in particular improves with respect to longer dependencies and dependencies near the root, as a result of fewer search errors thanks to enhanced information about the global sentence structure. One of the most striking results, however, is that both parsers improve their accuracy on longer sentences, with some models for some languages in fact being more accurate on medium-length sentences than on shorter sentences. This is a milestone in parsing research, and more research is needed to explain it.

In a broader perspective, we hope that future studies on dependency parsing will take the results obtained here into account and extend them by investigating other parsing approaches and neural network architectures. Indeed, given the rapid development of new representations and architectures, future work should include analyses of how all components in neural parsing architectures (embeddings, encoders, decoders) contribute to distinct error profiles (or lack thereof).

## Acknowledgments

We want to thank Ali Basirat, Christian Hardmeier, Jamie Henderson, Ryan McDonald, Paola Merlo, Gongbo Tang, and the EMNLP reviewers and area chairs for valuable feedback on preliminary versions of this paper. We acknowledge the computational resources provided by CSC in Helsinki and Sigma2 in Oslo through NeIC-NLPL (www.nlpl.eu).

## References


Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2442–2452.

Giuseppe Attardi, Felice Dell'Orletta, Maria Simi, and Joseph Turian. 2009. Accurate dependency parsing with a stacked multilayer perceptron. In *Proceedings of EVALITA 2009*.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with LSTMs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 349–359.

Ezra Black, Frederick Jelinek, John D. Lafferty, David M. Magerman, Robert L. Mercer, and Salim Roukos. 1992. Towards history-based grammars: Using richer models for probabilistic parsing. In *Proceedings of the 5th DARPA Speech and Natural Language Workshop*, pages 31–37.

Bernd Bohnet and Jonas Kuhn. 2012. The best of both worlds – a graph-based completion model for transition-based parsers. In *Proceedings of the 13th Conference of the European Chpater of the Association for Computational Linguistics (EACL)*, pages 77–87.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL)*, pages 149–164.

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings of the CoNLL Shared Task of EMNLP-CoNLL 2007*, pages 957–961.

Wanxiang Che, Yijia Liu, Yuxuan Wang, Bo Zheng, and Ting Liu. 2018. Towards better UD parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 55–64.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proceedings of the 5th International Conference on Learning Representations*.

Timothy Dozat, Peng Qi, and Christopher D. Manning. 2017. Stanford's graph-based neural dependency parser at the conll 2017 shared task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 20–30.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 334–343.

Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B:233–240.

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*, pages 340–345.

Agnieszka Falenska and Jonas Kuhn. 2019. The (non-)utility of structural features in BiLSTM-based dependency parsers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 117–128.

Yoav Goldberg. 2019. Assessing BERT's syntactic abilities. *CoRR*, abs/1901.05287.

Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING)*, pages 959–976.

Yoav Goldberg and Joakim Nivre. 2013. Training deterministic parsers with non-deterministic oracles. *Transactions of the Association for Computational Linguistics*, 1:403–414.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armanpd Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Martin Haspelmath, Matthew S. Dryer, David Gil, and Bernard Comrie. 2005. *Thw World Atlas of Language Structures*. Oxford University Press.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. 2001. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Liang Huang and Kenji Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1077–1086.

Ganesh Jawahar, Benjamin Muller, Amal Fethi, Louis Martin, Eric Villemonte de la Clergerie, Benoît Sagot, and Djamé Seddah. 2018. ELMoLex: Connecting ELMo and lexicon features for dependency parsing. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 223–237.

Richard Johansson and Pierre Nugues. 2007. Incremental dependency parsing using online learning. In *Proceedings of the CoNLL Shared Task of EMNLP-CoNLL 2007*, pages 1134–1138.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Daniel Kondratyuk. 2019. 75 languages, 1 model: Parsing universal dependencies universally. *CoRR*, abs/1904.02099.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1–11.

Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1288–1298.

Marco Kuhlmann, Carlos Gómez-Rodríguez, and Giorgio Satta. 2011. Dynamic programming algorithms for transition-based dependency parsers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 673–682.

Miryam de Lhoneux, Yan Shao, Ali Basirat, Eliyahu Kiperwasser, Sara Stymne, Yoav Goldberg, and Joakim Nivre. 2017a. From raw text to Universal Dependencies – Look, no tags! In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 207–217.

Miryam de Lhoneux, Sara Stymne, and Joakim Nivre. 2017b. Arc-hybrid non-projective dependency parsing with a static-dynamic oracle. In *Proceedings of the 15th International Conference on Parsing Technologies*, pages 99–104.

Miryam de Lhoneux, Sara Stymne, and Joakim Nivre. 2017c. Old school vs. new school: Comparing transition-based parsers with and without neural network enhancement. In *Proceedings of the 15th Treebanks and Linguistic Theories Workshop (TLT)*.

KyungTae Lim, Cheoneum Park, Changki Lee, and Thierry Poibeau. 2018. SEx BiST: A multi-source trainable parser with deep contextualized lexical representations. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 143–152.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. Linguistic knowledge and transferability of contextual representations. *CoRR*, abs/1903.08855.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 91–98.

Ryan McDonald, Kevin Lerman, and Fernando Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL)*, pages 216–220.

Ryan McDonald and Joakim Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 122–131.

Ryan McDonald and Joakim Nivre. 2011. Analyzing and integrating dependency parsers. *Computational Linguistics*, pages 197–230.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 81–88.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in*

*Natural Language Processing (HLT/EMNLP)*, pages 523–530.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pages 149–160.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34:513–553.

Joakim Nivre. 2009. Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL-IJCNLP)*, pages 351–359.

Joakim Nivre, Mitchell Abrams, Željko Agić, Lars Ahrenberg, Lene Antonsen, Katya Aplonova, Maria Jesus Aranzabe, Gashaw Arutie, Masayuki Asahara, Luma Ateyah, Mohammed Attia, Aitziber Atutxa, Liesbeth Augustinus, Elena Badmaeva, Miguel Ballesteros, Esha Banerjee, Sebastian Bank, Verginica Barbu Mititelu, Victoria Basmov, John Bauer, Sandra Bellato, Kepa Bengoetxea, Yevgeni Berzak, Irshad Ahmad Bhat, Riyaz Ahmad Bhat, Erica Biagetti, Eckhard Bick, Rogier Blokland, Victoria Bobicev, Carl Börstell, Cristina Bosco, Gosse Bouma, Sam Bowman, Adriane Boyd, Aljoscha Burchardt, Marie Candito, Bernard Caron, Gauthier Caron, Gülşen Cebiroğlu Eryiğit, Flavio Massimiliano Cecchini, Giuseppe G. A. Celano, Slavomír Čéplö, Savas Cetin, Fabricio Chalub, Jinho Choi, Yongseok Cho, Jayeol Chun, Silvie Cinková, Aurélie Collomb, Çağrı Çöltekin, Miriam Connor, Marine Courtin, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Arantza Diaz de Ilarraza, Carly Dickerson, Peter Dirix, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Marhaba Eli, Ali Elkahky, Binyam Ephrem, Tomaž Erjavec, Aline Etienne, Richárd Farkas, Hector Fernandez Alcalde, Jennifer Foster, Cláudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Moa Gärdenfors, Sebastian Garza, Kim Gerdes, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökırmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Céline Guillot-Barbance, Nizar Habash, Jan Hajič, Jan Hajič jr., Linh Hà Mỹ, Na-Rae Han, Kim Harris, Dag Haug, Barbora Hladká, Jaroslava Hlaváčová, Florinel Hociung, Petter Hohle, Jena Hwang, Radu Ion, Elena Irimia, Ọlájídé Ishola, Tomáš Jelínek, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşıkara, Sylvain Kahane, Hiroshi Kanayama, Jenna Kanerva, Boris Katz, Tolga Kayadelen, Jessica Kenney, Václava Kettnerová, Jesse Kirchner, Kamil Kopacewicz, Natalia Kotsyba, Simon Krek, Sookyoung Kwak, Veronika Laippala, Lorenzo Lambertino, Lucia Lam, Tatiana Lando, Septina Dian Larasati, Alexei Lavrentiev, John Lee, Phuong Lê H`ông, Alessandro Lenci, Saran Lertpradit, Herman Leung, Cheuk Ying Li, Josie Li, Keying Li, KyungTae Lim, Nikola Ljubešić, Olga Loginova, Olga Lyashevskaya, Teresa Lynn, Vivien Macketanz, Aibek Makazhanov, Michael Mandl, Christopher Manning, Ruli Manurung, Cătălina Mărănduc, David Mareček, Katrin Marheinecke, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Gustavo Mendonça, Niko Miekka, Margarita Misirpashayeva, Anna Missilä, Cătălin Mititelu, Yusuke Miyao, Simonetta Montemagni, Amir More, Laura Moreno Romero, Keiko Sophie Mori, Shinsuke Mori, Bjartur Mortensen, Bohdan Moskalevskyi, Kadri Muischnek, Yugo Murawaki, Kaili Müürisep, Pinkey Nainwani, Juan Ignacio Navarro Horñiacek, Anna Nedoluzhko, Gunta Nešpore-Bērzkalne, Luong Nguy˜ên Thị, Huy`ên Nguy˜ên Thị Minh, Vitaly Nikolaev, Rattima Nitisaroj, Hanna Nurmi, Stina Ojala, Adédayọ̀ Olúòkun, Mai Omura, Petya Osenova, Robert Östling, Lilja Øvrelid, Niko Partanen, Elena Pascual, Marco Passarotti, Agnieszka Patejuk, Guilherme Paulino-Passos, Siyao Peng, Cenel-Augusto Perez, Guy Perrier, Slav Petrov, Jussi Piitulainen, Emily Pitler, Barbara Plank, Thierry Poibeau, Martin Popel, Lauma Pretkalniņa, Sophie Prévost, Prokopis Prokopidis, Adam Przepiórkowski, Tiina Puolakainen, Sampo Pyysalo, Andriela Rääbis, Alexandre Rademaker, Loganathan Ramasamy, Taraka Rama, Carlos Ramisch, Vinit Ravishankar, Livy Real, Siva Reddy, Georg Rehm, Michael Rießler, Larissa Rinaldi, Laura Rituma, Luisa Rocha, Mykhailo Romanenko, Rudolf Rosa, Davide Rovati, Valentin Roșca, Olga Rudina, Jack Rueter, Shoval Sadde, Benoît Sagot, Shadi Saleh, Tanja Samardžić, Stephanie Samson, Manuela Sanguinetti, Baiba Saulīte, Yanin Sawanakunanon, Nathan Schneider, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Mo Shen, Atsuko Shimada, Muh Shohibussirri, Dmitry Sichinava, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Isabela Soares-Bastos, Carolyn Spadine, Antonio Stella, Milan Straka, Jana Strnadová, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Dima Taji, Yuta Takahashi, Takaaki Tanaka, Isabelle Tellier, Trond Trosterud, Anna Trukhina, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Zdeňka Urešová, Larraitz Uria, Hans Uszkoreit, Sowmya Vajjala, Daniel van Niekerk, Gertjan van Noord, Viktor Varga, Eric Villemonte de la Clergerie, Veronika Vincze, Lars Wallin, Jing Xian Wang, Jonathan North Washington, Seyi Williams, Mats Wirén, Tsegay Woldemariam, Tak-sum Wong, Chunxiao Yan, Marat M. Yavrumyan, Zhuoran Yu, Zdeněk Žabokrtský, Amir Zeldes, Daniel Zeman, Manying Zhang, and Hanzhi Zhu. 2018. Universal dependencies 2.3. LINDAT/CLARIN digital library at the Institute of For-

mal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task of EMNLP-CoNLL 2007*, pages 915–932.

Joakim Nivre, Johan Hall, Jens Nilsson, Gülsen Eryiğit, and Svetoslav Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL)*, pages 221–225.

Joakim Nivre and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 950–958.

Joakim Nivre and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 99–106.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237.

Peng Qi, Timothy Dozat, Yuhao Zhang, and Christopher D Manning. 2018. Universal dependency parsing from scratch. In *Proceedings of the 2018 CoNLL Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, page 160.

Kenji Sagae and Alon Lavie. 2006. Parser combination by reparsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 129–132.

Tal Schuster, Ori Ram, Regina Barzilay, and Amir Globerson. 2019. Cross-lingual alignment of contextual word embeddings, with applications to zero-shot dependency parsing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1599–1613.

Aaron Smith, Bernd Bohnet, Miryam de Lhoneux, Joakim Nivre, Yan Shao, and Sara Stymne. 2018a. 82 treebanks, 34 models: Universal dependency parsing with multi-treebank models. In *Proceedings of the 2018 CoNLL Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*.

Aaron Smith, Miryam de Lhoneux, Sara Stymne, and Joakim Nivre. 2018b. An investigation of the interactions between pre-trained word embeddings, character models and pos tags in dependency parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R. Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R. Bowman, Dipanjan Das, and Ellie Pavlick. 2019. What do you learn from context? probing for sentence structure in contextualized word representations. In *Proceedings of the 5th International Conference on Learning Representations*.

Ivan Titov and James Henderson. 2007. A latent variable model for generative dependency parsing. In *Proceedings of the 10th International Conference on Parsing Technologies (IWPT)*, pages 144–155.

Ke Tran, Arianna Bisazza, and Christof Monz. 2018. The importance of being recurrent for modeling hierarchical structure. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4731–4736.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Jorn Veenstra and Walter Daelemans. 2000. A memory-based alternative for connectionist shift-reduce parsing. Technical Report ILK-0012, Tilburg University.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 7th International Conference on Learning Representations*.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 323–333.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of the 8th International*

*Workshop on Parsing Technologies (IWPT)*, pages 195–206.

Daniel Zeman, Jan Hajič, Martin Popel, Martin Potthtyersast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. 2018. CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*.

Hao Zhang and Ryan McDonald. 2012. Generalized higher-order dependency parsing with cube pruning. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 320–331.

Yue Zhang and Stephen Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 562–571.

Yue Zhang and Joakim Nivre. 2011. Transition-based parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 188–193.

Yue Zhang and Joakim Nivre. 2012. Analyzing the effect of global learning and beam-search on transition-based dependency parsing. In *Proceedings of COLING 2012: Posters*, pages 1391–1400.

# Semantic Graph Parsing with Recurrent Neural Network DAG Grammars

**Federico Fancellu**[⊙] and **Sorcha Gilroy**[❩] and **Adam Lopez**[⊕] and **Mirella Lapata**[⊕]
[⊙]Samsung AI Centre (SAIC), Toronto, Canada
[❩]Peak AI, Manchester, United Kingdom
[⊕]University of Edinburgh, Edinburgh, United Kingdom

## Abstract

Semantic parses are directed acyclic graphs (DAGs), so semantic parsing should be modeled as graph prediction. But predicting graphs presents difficult technical challenges, so it is simpler and more common to predict the *linearized* graphs found in semantic parsing datasets using well-understood sequence models. The cost of this simplicity is that the predicted strings may not be well-formed graphs. We present recurrent neural network DAG grammars, a graph-aware sequence model that ensures only well-formed graphs while sidestepping many difficulties in graph prediction. We test our model on the Parallel Meaning Bank—a multilingual semantic graphbank. Our approach yields competitive results in English and establishes the first results for German, Italian and Dutch.

## 1 Introduction

Semantic parsing is the task of mapping natural language to machine interpretable meaning representations, which in turn can be expressed in many different formalisms, including lambda calculus (Montague, 1973), dependency-based compositional semantics (Liang et al., 2011), frame semantics (Baker et al., 1998), abstract meaning representations (AMR; Banarescu et al. 2013), minimal recursion semantics (MRS; Copestake et al. 2005), and discourse representation theory (DRT; Kamp 1981).

Explicitly or implicitly, a representation in any of these formalisms can be expressed as a directed acyclic graph (DAG). Consider the sentence "*Every ship in the dock needs a big anchor*". Its meaning representation, expressed as a Discourse Representation Structure (DRS, Kamp 1981), is shown in Figure 1.[1] A DRS is drawn as a box with

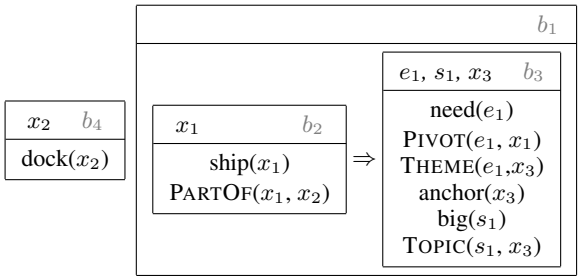


Figure 1: The discourse representation structure for "*Every ship in the dock needs a big anchor*". For ease of reference in later figures, each box includes a variable corresponding to the box itself, at top right in gray.

two parts: the top part lists variables for discourse referents (e.g. $x_1$, $e_1$) and the bottom part can contain unary predicates expressing the type of a variable (e.g. ship, need), binary predicates specifying relationships between variables (e.g. PARTOF, TOPIC), logical operators expressing relationships between nested boxes (e.g. $\Rightarrow$, $\neg$), or binary discourse relations (e.g., RESULT, CONTRAST). To express a DRS as a graph, we represent each box as a node labeled $\square$; each variable as a node labeled by its associated unary predicate; and each binary predicate, logical operator, or discourse relation as an edge from the first argument to the second (Figure 2). To fully realize the representation as a DAG, additional transformations are sometimes necessary: in DRS, when a box represents a *presupposition*, as box $b_4$ does, the label of the node corresponding to the presupposed variable is marked (e.g. $x_2$/dock$^P$); and edges can be reversed (e.g. TOPIC($s_1$, $x_3$) becomes TOPICOF($s_1$, $x_3$)).

Since meaning representations are graphs, semantic parsing should be modeled as graph prediction. But how do we predict graphs? A popular approach is to predict the *linearized* graph—that is, the *string* representation of the graph found in most semantic graphbanks. Figure 3 illus-

[1]For simplicity, our examples do not show time representations, though these are consistently present in our data.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2769–2778, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

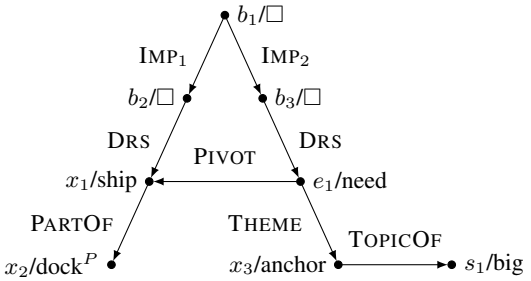


Figure 2: The DRS of Figure 1 expressed as a DAG.

```
(b1/□
  :IMP1(b2/□
    :DRS(x1/ship
      :PARTOF(x2/dock^p)))
  :IMP2(b3/□
    :DRS(e1/ need
      :PIVOT x1
      :THEME(x3/ anchor
        :TOPICOF(s1/ big)))))
```

Figure 3: The DAG of Figure 2 expressed as a string.

trates one style of linearization using PENMAN notation, in which graphs are written as well-bracketed strings which can also be interpreted as trees—note the correspondence between the tree-like structure of Figure 2 and the string in Figure 3.[2] Each subtree is a bracketed string starting with a node variable and its label (e.g. $b_2$/□), followed by a list of *relations* corresponding to the outgoing edges of the node. A relation consists of the edge label prefixed with a colon (:), followed by either the subtree rooted at the target node (e.g. :DRS ($x_1$/ship :PARTOF($x_2$/dock$^p$))), or a *reference* to the target node (e.g. :PIVOT $x_1$). By convention, if a node is the target of multiple edges, then the leftmost one is written as a subtree, and the remainder are written as references. Hence, every node is written as a subtree exactly once.

The advantage of predicting linearized graphs is twofold. The first advantage is that graphbank datasets usually already contain linearizations, which can be used without additional work. These linearizations are provided by annotators or algorithms and are thus likely to be very consistent in ways that are beneficial to a learning algorithm. The second advantage is that we can use simple, well-understood sequence models (Gu et al., 2016; Jia and Liang, 2016; van Noord et al., 2018) to model them. But this simplicity comes with a cost: sequence models can predict strings that don't correspond to graphs—for example, strings with ill-formed bracketings or unbound variable names. While it is often possible to fix these strings with pre- or post-processing, we would prefer to model the problem in a way that does not require this.

Models that predict graphs are complex and far less well-understood than models that predict sequences. Fundamentally, this is because predicting graphs is difficult: every graph has many possible linearizations, so from a probabilistic perspective, the linearization is a latent variable that must be marginalized out (Li et al., 2018). Groschwitz et al. (2018) model graphs as trees, interpreted as the (latent) derivation trees of a graph grammar; Lyu and Titov (2018) model graphs with a conditional variant of the classic Erdös and Rényi (1959) model, first predicting an alignment for each node of the output graph, and then predicting, for each pair of nodes, whether there is an edge between them. Buys and Blunsom (2017), Chen et al. (2018), and Damonte et al. (2017) all model graph generation as a sequence of actions, each aligned to a word in the conditioning sentence. Each of these models has a latent variable—a derivation tree or alignment—which must be accounted for via preprocessing or complex inference techniques.

Can we combine the simplicity of sequence prediction with the fidelity of graph prediction? We show that this is possible by developing a new model that predicts sequences through a simple string rewriting process, in which each rewrite corresponds to a well-defined graph fragment. Importantly, any well-formed string produced by our model has exactly one derivation, and thus no latent variables. We evaluate our model on the Parallel Meaning Bank (PMB, Abzianidze et al. 2017), a multilingual corpus of sentences paired with DRS representations. Our model performs competitively on English, and better than sequence models in German, Italian, and Dutch.

## 2 Graph-aware string rewriting

We use a grammar to model the process of rewriting. Formally, our grammar is a *graph grammar*, specifically a *restricted DAG grammar* (Björklund

[2] Although PENMAN notation is now closely associated with AMR, it can represent quite arbitrary graphs as strings. Our actual implementation does not use PENMAN notation, but we use it here for expository purposes since it is relatively familiar; the underlying ideas are unchanged.

et al., 2016), a type of context-free graph grammar designed to model linearized DAGs. Since linearized DAGs are strings, we present it as a string-rewriting system, which can be described more compactly than a graph grammar while making the connection to sequences more explicit. The correspondence between string rewriting and graph grammars is given in the Appendix.

A grammar in our model is defined by a set $\Sigma$ of **terminal symbols** consisting of all symbols that can appear in the final string—brackets, variable types, node labels, and edge labels; a set $N$ of $n+1$ **nonterminal symbols** denoted by $\{L, T_0, \ldots, T_n\}$, for some maximum value $n$; an unbounded set $V$ of **variable references** $\{\$1, \$2, \ldots\}$; and a set of productions, which are defined below.

We say that $T_0$ is the **start symbol**, and for each symbol $T_i \in N$, we say that $i$ is its **rank**, and we say that $L$ has a rank of 0. A nonterminal of rank $i$ can be written as a **function** of $i$ variable references—for example, we can write $T_2(\$1, \$2)$. By convention, we write the rank-0 nonterminals $L$ and $T_0$ without brackets. Productions in our grammar take the form $\alpha \rightarrow \beta$, where $\alpha$ is a function of rank $i$ over $\$1, \ldots, \$i$; and $\beta$ is a linearized graph in PENMAN format, with each of its subtrees replaced by either a function or a variable reference. Optionally, the variable name in $\beta$ may replace one of the variable references in $\alpha$. All variable references in a production must appear at least twice. Hence every variable reference in $\alpha$ must appear at least once in $\beta$, and variables that do not appear in $\alpha$ must appear at least twice in $\beta$.

To illustrate, we will use the following grammar, which can generate the string in Figure 3, assuming $L$ can also rewrite as any node label.

$$T_0 \rightarrow (b/\square \text{ :IMP}_1\ T_1(\$1)\ \text{:IMP}_2\ T_1(\$1)) \quad (r_1)$$
$$T_0 \rightarrow (x/L) \quad (r_2)$$
$$T_0 \rightarrow (s/L) \quad (r_3)$$
$$T_0 \rightarrow (x/L\ \text{:TOPICOF}\ T_0) \quad (r_4)$$
$$T_1(\$1) \rightarrow (b/\square\ \text{:DRS}\ T_1(\$1)) \quad (r_5)$$
$$T_1(\$1) \rightarrow (e/L\ \text{:PIVOT}\ \$1\ \text{:THEME}\ T_0) \quad (r_6)$$
$$T_1(x) \rightarrow (x/L\ \text{:PARTOF}\ T_0) \quad (r_7)$$

Our grammar derives strings by first rewriting the start symbol $T_0$, and at each subsequent step rewriting the leftmost function in the partially derived string, with special handling for variable references described below. A derivation is complete when no functions remain.

We illustrate the rewriting process in Figure 4. The start symbol $T_0$ at step 1 is rewritten by production $r_1$ in step 2, and the new $b$ variable introduced at this step is deterministically renamed to the unique name $b_1$. In step 3, the leftmost $T_1(\$1)$ is rewritten by production $r_5$, and the new $b$ variable is likewise renamed to the unique $b_2$. All productions apply in this way, simply replacing a left-hand-side function with a right-hand side expression. These rewrites are coupled with a mechanism to correctly handle multiple references to shared variables, as illustrated in Step 4 when production $r_7$ is applied. In this production, the left-hand-side function applies to the $x$ variable naming the right-hand-side node. When this production applies, $x$ is renamed to the unique $x_1$ as in previous steps, but because it appears in the left-hand-side, the reference $\$1$ is bound to this new variable name throughout the partially-derived string. In this way, the reference to $x_1$ is passed through the subsequent rewrites, becoming the target of a PIVOT edge at step 10. Derivations of a DAG grammar are context-free (Figure 5).[3]

Our model requires an explicit grammar like the one in $r_1 \ldots r_7$, which we obtain by converting each DAG in the training data into a sequence of productions. The conversion yields a single, unique sequence of productions via a simple linear-time algorithm that recursively decomposes a DAG into subgraphs (Björklund et al., 2016). Each subgraph consists of single node and its outgoing edges, as exemplified by the PENMAN-formatted right-hand-sides of $r_1$ through $r_7$. Each outgoing edge points to a nonterminal symbol representing a subgraph. If a subgraph does not share any nodes with its siblings, it is represented by $T_0$. But if any subgraphs share a node, then a variable reference must refer for this node in the production associated with the lowest common ancestor of all its incoming edges. For example, in Figure 3, the common ancestor of the two edges targeting $x_1$ is the node $b_1$, so production $r_1$ must contain two copies of variable reference $\$1$ to account for this. A more mathematical account can be found along with a proof of correctness in Björklund et al. (2016) Our implementation follows their description unchanged.

[3] However, the language of derived strings may not be context-free due to variable references.

| Step | Action | Production | Result |
|---|---|---|---|
| 1 | START | start | $\underline{T_0}$ |
| 2 | GEN-FRAG | $r_1$ | $(b_1/\square$ :IMP$_1$ $\underline{T_1(\$1)}$ :IMP$_2$ $T_1(\$1))$ |
| 3 | GEN-FRAG | $r_5$ | $(b_1/\square$ :IMP$_1$ $\underline{(b_2/\square$ :DRS $T_1(\$1))}$ :IMP$_2$ $T_1(\$1))$ |
| 4 | GEN-FRAG | $r_7$ | $(b_1/\square$ :IMP$_1$ $(b_2/\square$ :DRS $\underline{(x_1/\underline{L}$ :PARTOF $T_0)}$ :IMP$_2$ $T_1(x_1))$ |
| 5 | GEN-LABEL | $L \rightarrow$ ship | $(b_1/\square$ :IMP$_1$ $(b_2/\square$ :DRS $(x_1/$ship :PARTOF $\underline{T_0})$ :IMP$_2$ $T_1(x_1))$ |
| 6 | GEN-FRAG | $r_2$ | $(b_1/\square$ :IMP$_1$ $(b_2/\square$ :DRS $(x_1/$ship :PARTOF $\underline{(x_2/\underline{L})})$ :IMP$_2$ $T_1(x_1))$ |
| 7 | GEN-LABEL | $L \rightarrow$ dock$^p$ | $(b_1/\square$ :IMP$_1$ $(b_2/\square$ :DRS $(x_1/$ship :PARTOF $(x_2/$dock$^p))$ :IMP$_2$ $\underline{T_1(x_1)})$ |
| 8 | REDUCE | $--$ | $=$ |
| 9 | REDUCE | $--$ | $=$ |
| 8 | GEN-FRAG | $r_5$ | $(b_1/\square$ :IMP$_1$ $(b_2/\square$ :DRS $(x_1/$ship :PARTOF $(x_2/$dock$^p))$ :IMP$_2$ $\underline{(b_3/\square$ :DRS $T_1(x_1))}$ |
| 9 | GEN-FRAG | $r_6$ | $(b_1/\square$ :IMP$_1$ $(b_2/\square$ :DRS $(x_1/$ship :PARTOF $(x_2/$dock$^p))$ :IMP$_2$ $(b_3/\square$ :DRS $\underline{(e_1/\underline{L}$ :PIVOT $x_1$ :THEME $T_0)})$ |
| 10 | GEN-LABEL | $L \rightarrow$ need | $(b_1/\square$ :IMP$_1$ $(b_2/\square$ :DRS $(x_1/$ship :PARTOF $(x_2/$dock$^p))$ :IMP$_2$ $(b_3/\square$ :DRS $(e_1/$need :PIVOT $x_1$ :THEME $\underline{T_0}))$ |

Figure 4: A partial derivation of the string in Figure 3. The stack operations follow closely each step in the derivation, where GEN-FRAG and GEN-LABEL are invoked when rewriting a non-terminal $T$ and a terminal $L$ respectively. In the result of each step, the leftmost function is underlined, and is rewritten in the fragment in blue in the next step. On the other hand, a REDUCE operation is invoked when a generated fragment does not contain non-terminals $T$ to expand further (in this partial derivation, this is the case of the result of production $r_2$).

## 3 Neural Network Realizer

We model graph parsing with an encoder-decoder architecture that takes as input a sentence $w$ and outputs a directed acyclic graph $G$ derived using the rewriting system of Section 2. Specifically, we model its derivation tree in top-down, left-to-right order as a sequence of actions $a = a_1 \dots a_{|a|}$, inspired by Recurrent Neural Network Grammars (RNNG, Dyer et al., 2016). As in RNNG, we use a stack to store partial derivations.

We model two types of actions: **GEN-FRAG** rewrites $T_i$ nonterminals, while **GEN-LABEL** rewrites $L$ nonterminals, always resulting in a leaf of the derivation tree. A third **REDUCE** action is applied whenever a subtree of the derivation tree is complete, and since the number of subtrees is known in advance, it is applied deterministically. For example, when we predict $r_1$, this determines that we must rewrite an $L$ and then recursively rewrite two copies of $T_1(\$1)$ and then apply **REDUCE**. Hence graph generation reduces to predicting rewrites only.

We define the probability of generating graph $G$ conditioned of input sentence $w$ as follows:

$$p(G|w) = p(a|w) = \prod_{i=1}^{|a|} p(a_i|a_{<i}, w) \quad (1)$$

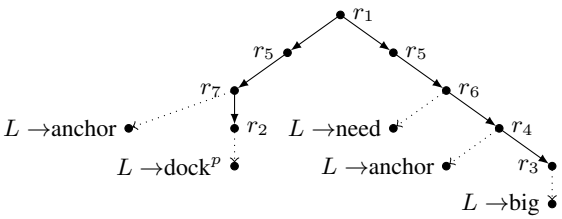


Figure 5: A derivation tree corresponding to Figure 4. Solid edges rewrite $T_n$ nonterminals, while dotted rewrite $L$ nonterminals.

**Input Encoder** We represent the $i$th word $w_i$ of input sentence $w = w_1 \dots w_{|w|}$ using both learned and pre-trained word embeddings ($\mathbf{w}_i$ and $\mathbf{w}_i^p$ respectively), lemma embedding ($\mathbf{l}_i$), part-of-speech embedding ($\mathbf{p}_i$), universal semantic tag (Abzianidze and Bos, 2017) embedding ($\mathbf{u}_i$), and dependency label embedding ($\mathbf{d}_i$).[4] An input $\mathbf{x}_i$ is computed as the weighted concatenation of these features followed by a non-linear projection (with vectors and matrices in **bold**):

$$\mathbf{x}_i = \tanh(\mathbf{W^{(1)}}[\mathbf{w}_i; \mathbf{w}_i^p; \mathbf{l}_i; \mathbf{p}_i; \mathbf{u}_i; \mathbf{d}_i]) \quad (2)$$

Input $\mathbf{x}_i$ is then encoded with a bidirectional LSTM, yielding contextual representation $\mathbf{h}_i^e$.

[4] Universal semantic tags are language neutral tags intended to characterize lexical semantics.

**Graph decoder** Since we know in advance whether the next action is **GEN-FRAG** or **GEN-LABEL**, we use different models for them.

**GEN-FRAG**. If step $t$ rewrites a $T$ nonterminal, we predict the production $y_t$ that rewrites it using context vector $\mathbf{c}_t$ and incoming edge embedding $\mathbf{e}_t$. To obtain $\mathbf{c}_t$ we use *soft attention* (Luong et al., 2015) and weight each input hidden representation $\mathbf{h}_i^e$ to decoding hidden state $\mathbf{h}_t^d$:

$$\begin{aligned} \text{score}(\mathbf{h}_i^e, \mathbf{h}_t^d) &= \mathbf{h}_i^e \mathbf{W}^{(2)} \mathbf{h}_t^d \\ \alpha_{ti} &= \frac{\exp(\text{score}(\mathbf{h}_i^e, \mathbf{h}_t^d))}{\sum_{i'} \exp(\text{score}(\mathbf{h}_{i'}^e, \mathbf{h}_t^d))} \\ \mathbf{c}_t &= \sum_{i=1}^{n} \alpha_{ti} \mathbf{h}_i^e \\ \mathbf{y}_t &= \mathbf{W}^{(3)} \mathbf{c}_t + \mathbf{W}^{(4)} \mathbf{e} \end{aligned} \tag{3}$$

The contribution of $\mathbf{c}$ and $\mathbf{e}$ is weighted by matrices $\mathbf{W}^{(3)}$ and $\mathbf{W}^{(4)}$, respectively.

We then update the stackLSTM representation using the embedding of the non-terminal fragment $\mathbf{y}_t$ (denoted as $\mathbf{y}_t^e$), as follows:

$$\mathbf{h}_{t+1}^d = \text{LSTM}(\mathbf{y}_t^e, \mathbf{h}_t^d) \tag{4}$$

**GEN-LABEL**. Labels $L$ can be rewritten to either semantic constants (e.g., 'speaker', 'now', 'hearer') or unary predicates that often corresponds to the lemmas of the input words (e.g., 'love') or. We predict the former using a model identical to the one for GEN-FRAG. For the latter, we use a *selection mechanism* to choose an input lemma to copy to output. We model selection following Liu et al. (2018), assigning each input lemma a score $o_{ji}$ that we then pass through a softmax layer to obtain a distibution:

$$\begin{aligned} o_{ji} &= \mathbf{h}_j^{dT} \mathbf{W}^{(5)} \mathbf{h}_i^e \\ p_i^{copy} &= \textsc{SoftMax}(o_{ji}) \end{aligned} \tag{5}$$

where $\mathbf{h}_i$ is the encoder hidden state for word $w_i$.

We allow the model to learn whether to use soft-attention or the selection mechanism through a binary classifier, conditioned on the decoder hidden state at time $t$, $\mathbf{h}_t^d$. Similar to Equation (4), we update the stackLSTM with the embedding of terminal predicted.[5]

[5]In the PMB, each terminal is annotated for sense (e.g. 'n.01', 's.01') and presupposition (e.g. for 'dock$^p$' in Figure 3) as well. We predict both the sense tag and whether a terminal is presupposed or not independently conditioned on the current stackLSTM state and the embedding of the main terminal labels but are **not** used to update the state of the stackLSTM.

**REDUCE**. When a reduce action is applied, we use an LSTM to compose the fragments on top of the stack. Using the derivation tree in Figure 5 as reference, let $[\mathbf{c}_1, ..., \mathbf{c}_n]$ denote the embeddings of one or more sister nodes $r_i$ and $p_u$ the embedding of their parent node, which we refer to as *children* and *parent fragments* respectively. A reduce operation runs an LSTM over the children fragments and the parent fragment in order and then uses the final state $u$ to update the stack LSTM as follows:

$$\begin{aligned} \mathbf{i} &= [\mathbf{c}_1 ... \mathbf{c}_n, \mathbf{p}_u] \\ \overrightarrow{\mathbf{u}} &= LSTM(\mathbf{i}_t, \mathbf{h}_t^c) \\ \mathbf{h}_{t+1}^d &= LSTM(\mathbf{u}, \mathbf{h}_t^d) \end{aligned}$$

The models are trained to minimize a cross-entropy loss objective $J$ over the sequence of gold actions $a_i$ in the derivation:

$$J = -\sum_{i=1}^{|a|} log\ p(a_i) \tag{6}$$

## 4 Experimental Setup

We evaluated our model on the Parallel Meaning Bank (PMB; Abzianidze et al. 2017), a semantic bank where sentences in English, Italian, German, and Dutch have been annotated following Discourse Representation Theory (Kamp and Reyle, 2013). Lexical predicates in PMB are in English, even for non-English languages. Since this is not compatible with our copy mechanism, we revert predicates to their orignal language by substituting them with the lemmas of the tokens they are aligned to. In our experiments we used both PMB v.2.1.0 and v.2.2.0[6]; we included the former release in order to compare against the state-of-the-art seq2seq system of van Noord et al. (2018). Statistics on the data and the grammar extracted from v.2.2.0 are reported in Table 1.

### 4.1 Converting DRSs to Graphs

In this section we discuss how DRSs are converted to *acyclic*, *single-rooted*, and *fully-instantiated* graphs (i.e., how to translate Figure 1 to Figure 2). In general, box structures are converted to acyclic graphs by rendering boxes and lexical predicates

[6]English data is available for both releases at https://github.com/RikVN/DRS_parsing; for the other languages, we used the officially released data available at http://pmb.let.rug.nl/data.php

| Gold | #inst. | train #frags | avg. rank | dev #inst. | test #inst. |
|---|---|---|---|---|---|
| en | 4,574 | 1,105 | 1.64 | 682 | 650 |
| it | — | — | — | 387 | 386 |
| de | — | — | — | 736 | 735 |
| nl | — | — | — | 356 | 355 |

| Silver | #inst. | train #frags | avg. rank |
|---|---|---|---|
| en | 52,120 | 11,206 | 1.90 |
| it | 2,546 | 1,051 | 1.52 |
| de | 3,274 | 1,781 | 1.54 |
| nl | 874 | 789 | 1.45 |

Table 1: Data statistics of PMB v.2.2.0.

as nodes, while conditions, operators, and discourse relations become edge labels between these nodes.

We consider *main* boxes (see $b_2$, $b_3$, and $b_4$ in Figure 1 separately from *presuppositional* boxes (see $b_1$), which represent instances that are presupposed in the wider discourse context (e.g., definite expressions). Using Figure 2 as an example, $b_2$, $b_3$ and $b_4$ become nodes in the graph and material implication (IMP stands for $\Rightarrow$) becomes an edge label. If an operator or a relation is binary, as in this case, we number the edge label so as to preserve the order of the operands.

For each node in a *main* box, we expand the graph by adding all relations and variables belonging to it. We identify the head of the first relation or the first referent mentioned as the *head variable*. These are 'ship ($x_1$)' for $b_2$, and 'need($e_1$)' for $b_3$. We attach the head variable as a child of the box-node and follow the relations recursively to expand the subgraph. If while expanding a graph a variable in a condition is part of a presuppositional box, we introduce it as a new node and add to its label the superscript $^p$. When expanding the DAG along the edge PartOf, since $x_2$ is also in the presuppositional box in Figure 1, we attach the node 'dock$^p$'. Graphs extracted this way are mostly acyclic except for adjectival phrases and relative clauses where state variables can be themselves root (e.g., big "a.01" $s_1$). We get rid of these extra roots by reversing the direction of the edge involved and adding an '-of' to the edge label to flag this change (see TOPIC-OF).

### 4.2 System Comparison

We compared the performance of our graph parser (seq2graph below) with a sequence-to-sequence model (enchanced with a copy mechanism) which decodes to a string linearization of the graph similar to the one shown in Figure 3 (seq2seq + copy below). We also compare against the recently proposed model of van Noord et al. (2018); they introduce a seq2seq model that generates a DRS as a concatenation of *clauses*, essentially a flat version of the standard box notation. The decoded string is made aware of the overall graph structure during preprocessing where variables are replaced by indices indicating when they were first introduced and their recency. In contrast, we model the graph structure explicitly. van Noord et al. (2018) experimented with both word and character-based models, as well as with an ensemble of both, using word embedding features. Since all our models are word-based, we compare our results with their best word model, using word embedding features only (trained using 10-fold cross validation).

### 4.3 Model Configurations

In addition to word embeddings[7], we also report on experiments which make use of additional features. Specifically, for each word we add information about its universal PoS tag, lemma, universal semantic tag, and dependency label.[8]

In Section 2, we mentioned that given a production $\alpha \rightarrow \beta$, variable references in $\alpha$ should appear at least once in $\beta$ (i.e., they should have the same rank). In all experiments so far, we did not model this constraint explicitly to investigate whether the model is able by default to predict rank correctly. However, in exploring model configurations we also report on whether adding this constrant leads to better performance .

### 4.4 Cross-lingual Experiments

We conducted two sets of experiments: one monolingual (mono below) where we train and test on the same language and one cross-lingual (cross below), where we train a model on English and test it on one of the other three languages. The goal of our cross-lingual experiments is to examine whether we need data in a target language at all since the semantic representation itself is language agnostic and lexical predicates are dealt with via

[7]We used word embeddings pre-trained with Glove and available at https://nlp.stanford.edu/projects/glove/.

[8]Universal PoS tags, lemmas and dependency labels for all languages were obtained using pretrained UDPipe models available at http://ufal.mff.cuni.cz/udpipe#download; gold-standard universal semantic tags were extracted from the PMB release.

the copy mechanism. Most of the features mentioned above are cross-linguistic and therefore fit both mono and cross-lingual settings, with the exception of lemma and word embeddings, where we exclude the former and replaced the latter with multilingual word embeddings.[9]

### 4.5 System settings

For training, we used the Adam optimizer (Kingma and Ba, 2014) with an initial learning rate of 0.001 and a decay rate of 0.1 every 10 epochs. Randomly initialized and pre-trained word embeddings have a dimensionality of 128 and 100 respectively, and all other features a dimensionality of 50. In all cross-lingual experiments, the pre-trained word embeddings have a dimensionality of 300. The LSTMs in the encoder and the decoder have a dimensionality of 150 and non-terminal and terminal embeddings during decoding have a dimensionality of 50. The system is trained for 30 epochs, with the best system chosen based on dev set performance.

### 4.6 Evaluation Metric

We evaluated our system by scoring the similarity between predicted and gold graphs. We used Counter (van Noord et al., 2018), an adaptation of Smatch (Cai and Knight, 2013) to Discourse Representation Structures where graphs are first transformed into a set of 'source node – edge label – target node' triples and the best mapping between the variables is found through an iterative hill-climbing strategy. Furthermore, Counter checks whether DRSs are well-formed in that all boxes should be connected, acyclic, with fully instantiated variables, and correctly assigned sense tags.

It is worth mentioning that there can be cases where our parser generates ill-formed graphs according to Counter; this is however not due to the model itself but to the way the graph is converted back in a format accepted by Counter.

All results shown are an averages over 5 runs.

[9]We experimented with embeddings obtained with iterative procrustes (available at https://github.com/facebookresearch/MUSE) and with Guo et al. (2016)'s 'robust projection' method where the embedding of non-English words is computed as the weighted average of English ones. We found the first method to perform better on cross-lingual word similarity tasks and used it in our experiments.

| | P | R | F1 | illformed |
|---|---|---|---|---|
| van Noord et al. (2018) | — | — | 72.80 | 20% |
| seq2seq + copy | 75.57 | 67.27 | 71.18 | 4.12% |
| seq2graph | 75.51 | 71.69 | **73.55** | **0.40%** |

Table 2: Model performance (Precision, Recall, $F_1$) on PMB data (v.2.1.0, test set); models were trained on *gold* standard data.

| | #sents | conversion score | F1 | illformed |
|---|---|---|---|---|
| van Noord et al. (2018) | 73,778 | 100% | 82.7 | 1.10% |
| seq2seq+copy | 56.694 | 89.58% | 74.67 | 13.45% |
| seq2graph | 56.694 | 89.58% | 77.1 | 0.90% |

Table 3: Model performance (Precision, Recall, $F_1$) on PMB data (v.2.1.0, test set); models were trained on *gold* and *silver* standard data combined.

## 5 Results

**System comparison** Table 2 summarizes our results on the PMB gold data (v.2.1.0, test set). We compare our graph decoder against the system of van Noord et al. (2018) and our implementation of a seq2seq model, enhanced with a copy mechanism. Overall, we see that our graph decoder outperforms both models. Moreover, it reduces the number of illformed representations without any specific constraints or post-processing in order to ensure the well-formedness of the semantics of the output.

The PMB (v.2.1.0) contains a large number of silver standard annotations which have been only partially manually corrected (see Table 1). Following van Noord et al. (2018), we also trained our parser on both silver and gold standard data combined. As shown in Table 3, increasing the training data improves performance but the difference is not as dramatic as in van Noord et al. (2018). We found that this is because our parser requires graphs that are fully instantiated – all unary predicates (e.g. *ship(x)*) need to be present for the graph to be fully connected, which is often not the case for silver graphs. Our model is at a disadvantage since it could exploit less training data; during grammar extraction we could not process around 20K sentences and in some cases could not reconstruct the whole graph, as shown by the *conversion score*.[10]

[10]The conversion score is computed using Counter where we consider the converted graph representation as a 'predicted' DRS and the original DRS structure as gold data. As a comparison, converting the *gold* DRS sturctures yields a conversion score of 99.8%.

| | P | R | F1 | illformed |
|---|---|---|---|---|
| seq2seq + copy | 60.29 | 74.09 | 66.48 | 10.60% |
| seq2graph | **70.69** | **74.46** | **72.53** | 0.10% |
| +restrict | 70.50 | 74.64 | 72.51 | 0.70% |
| +*feats* | **72.51** | **76.44** | **74.42** | 0.60% |
| -lemmas | 71.53 | 76.28 | 73.83 | 0.32% |
| -pos | 72.11 | 75.99 | 74.00 | 0.35% |
| -semtag | 70.21 | 74.45 | 72.27 | 0.53% |
| -words | 70.45 | 74.59 | 72.46 | 0.64% |
| -dep | 72.16 | 76.40 | 74.22 | 0.50% |

Table 4: Model performance (Precision, Recall, $F_1$) on PMB (v2.2.0, development set).

| *Italian* | P | R | F1 | ill |
|---|---|---|---|---|
| s2s + copy mono-silver | 61.33 | 72.42 | 66.41 | 14.80 |
| s2g mono-silver | 74.50 | 77.27 | 75.86 | 0.10 |
| s2g cross-silver | 70.35 | 71.91 | 71.12 | 0.00 |

| *German* | P | R | F1 | ill |
|---|---|---|---|---|
| s2s + copy mono-silver | 56.86 | 65.40 | 60.83 | 10.67 |
| s2g mono-silver | 66.44 | 69.34 | 67.86 | 0.80 |
| s2g cross-silver | 66.14 | 65.72 | 63.50 | 0.40 |

| *Dutch* | P | R | F1 | ill |
|---|---|---|---|---|
| s2s + copy mono-silver | 54.27 | 64.17 | 58.81 | 10.67 |
| s2g mono-silver | 63.50 | 68.37 | 65.84 | 0.86 |
| s2g cross-silver | 62.94 | 67.32 | 65.06 | 0.50 |

Table 5: Model performance across languages (Precision, Recall, $F_1$). Results are reported on the PMB test set (v.2.2.0) for each language.

**Model configurations** Table 4 reports on various ablation experiments investigating which features and combinations thereof perform best. The experiments were conducted on the development set (PMB v2.2.0). We show a basic version of our seq2graph model with word embeddings, to which we add information about rank (+restrict). We also experimented with the full gamut of additional features (+feats) as well as with ablations of individual feature classes. For comparsion, we also show the performance of a graph-to-string model (seq2seq+copy).

As can be seen, all linguistic features seem to improve performance. Restricting fragment selection by rank does not seem to improve the overall result showing that our baseline model is already able to predict fragments with the correct rank throughout the derivation. Subsequent experiments report results with this model using all linguistic features, unless otherwise specified.

**Cross-lingual experiments** Results on DRT parsing for languages other than English are reported in Table 5. There is no gold standard training data for non-English languages in the PMB (v.2.2.0). We therefore trained our parser on silver standard data but did use the provided gold standard data for development and testing (see Table 1). We present two versions of our parser, one where we train and test on the same language (s2g mono-silver) and another one where a model is trained on English but tested on the other languages (s2g cross-silver). We also show the results of a sequence-to-sequence model enhanced with a copy mechanism.

In the monolingual setting, our graph parser outperforms the seq2seq baseline by a large margin; we hypothesize this is due to the large percentage of ill-formed semantics, mostly due to training on silver data. The difference in performance between our cross-lingual parser and the monolingual parser for all languages is small, and in Dutch the two parsers perform on par, suggesting that English data and language independent features can be leveraged to build parsers in other languages when data is scarse or event absent. We also conducted various ablation studies to examine the contribution of individual features to cross-linguistic semantic parsing. Our experiments revealed that universal semantic tags are most useful, while the multilingual word embeddings that we have tested with are not. We refer the interested reader to the supplementary material for more detail on these experiments.

**Error Analysis** We further analyzed the output of our parser to gain insight as to what parts of meaning representation are still challenging. Table 6 shows a more detailed break-down of system output as computed by Counter, where *operators* (e.g., negation, implication), roles (i.e., binary relations, such as 'Theme'), concepts (i.e., unary predicates like 'ship'), and *synsets* (i.e., sense tags like 'n.01') are scored separately. Synsets are further broken down into into 'Nouns', 'Verbs', 'Adverbs', and 'Adjectives'. We compare our best seq2graph models (+*feats*) trained on the PMB v.2.2.0, gold and gold+silver data respectively.

Adding silver data helps with semantic elements (operators, roles and concepts), but does not in the case of sense prediction where the only category that benefits from additional data are is nouns. We also found that ignoring the prediction of sense tags altogether helps with the performance of both models.

| | | seq2graph(gold) | seq2graph(silver) |
|---|---|---|---|
| all clauses | | 74.42 | **76.36** |
| DRS operators | | 88.05 | **89.36** |
| Roles | | 72.95 | **74.03** |
| Concepts | | 71.13 | **74.42** |
| Synsets | – Nouns | 77.13 | **81.80** |
| | – Verbs | **55.63** | 55.49 |
| | – Adverbs | **44.44** | 42.11 |
| | – Adjectives | **61.67** | 58.48 |
| -sense | | 76.91 | **78.93** |

Table 6: $F_1$-scores of fine-grained evaluation on the PMB (v.2.2.0) development set; the seq2graph models trained on *gold* (left) and with *gold* and *silver* data combined (right) are compared.

## 6 Conclusions

In this paper we have introduced a novel graph parser that can leverage the power and flexibility of sequential neural models while still operating on graph structures. Heavy preprocessing tailored to a specific formalism is replaced by a flexible grammar extraction method that relies solely on the graph while yielding performance that is on par or better than string-based approaches. Future work should focus on extending our parser to other formalisms (AMR, MRS, etc.). We also plan to explore modelling alternatives, such as taking different graph generation oders into account (bottom-up vs. top-down) as well as predicting the components of a fragment (type, number of edges, edge labels) separately.

## Acknowledgments

We thank Yova Kementchedjhieva, Andreas Grivas and the three anonymous reviewers for their useful comments. This work was done while Federico Fancellu was a post-doctoral researcher at the University of Edinburgh. The views expressed are his own and do not necessarily represent the views of Samsung Research. We gratefully acknowledge the support of the European Research Council (Fancellu, Lapata; award number 681760, "Translating Multiple Modalities into Text"); and the EPSRC Centre for Doctoral Training in Data Science, funded by the UK Engineering and Physical Sciences Research Council (Gilroy; grant EP/L016427/1) and the University of Edinburgh (Gilroy).

## References

Abzianidze, L., Bjerva, J., Evang, K., Haagsma, H., van Noord, R., Ludmann, P., Nguyen, D.-D., and Bos, J. (2017). The parallel meaning bank: Towards a multilingual corpus of translations annotated with compositional meaning representations. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 242–247, Valencia, Spain. Association for Computational Linguistics.

Abzianidze, L. and Bos, J. (2017). Towards universal semantic tagging. In *IWCS 201712th International Conference on Computational SemanticsShort papers*.

Baker, C. F., Fillmore, C. J., and Lowe, J. B. (1998). The Berkeley FrameNet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pages 86–90, Montreal, Quebec, Canada.

Banarescu, L., Bonial, C., Cai, S., Georgescu, M., Griffitt, K., Hermjakob, U., Knight, K., Koehn, P., Palmer, M., and Schneider, N. (2013). Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria. Association for Computational Linguistics.

Björklund, H., Drewes, F., and Ericson, P. (2016). Between a rock and a hard place–uniform parsing for hyperedge replacement dag grammars. In *International Conference on Language and Automata Theory and Applications*, pages 521–532. Springer.

Buys, J. and Blunsom, P. (2017). Robust incremental neural semantic graph parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1215–1226.

Cai, S. and Knight, K. (2013). Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 748–752.

Chen, B., Sun, L., and Han, X. (2018). Sequence-to-action: End-to-end semantic graph generation for semantic parsing. *arXiv preprint arXiv:1809.00773*.

Copestake, A., Flickinger, D., Pollard, C., and Sag, I. A. (2005). Minimal recursion semantics: An introduction. *Research on language and computation*, 3(2-3):281–332.

Damonte, M., Cohen, S. B., and Satta, G. (2017). An incremental parser for abstract meaning representation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 536–546.

Dyer, C., Kuncoro, A., Ballesteros, M., and Smith, N. A. (2016). Recurrent neural network grammars. In *Proceedings of NAACL-HLT*, pages 199–209.

Erdös, P. and Rényi, A. (1959). On random graphs. *Publicationes Mathematicae*, 6:290–297.

Groschwitz, J., Lindemann, M., Fowlie, M., Johnson, M., and Koller, A. (2018). Amr dependency parsing with a typed semantic algebra. *arXiv preprint arXiv:1805.11465*.

Gu, J., Lu, Z., Li, H., and Li, V. O. (2016). Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1631–1640.

Guo, J., Che, W., Yarowsky, D., Wang, H., and Liu, T. (2016). A representation learning framework for multi-source transfer parsing. In *AAAI*, pages 2734–2740.

Jia, R. and Liang, P. (2016). Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22, Berlin, Germany.

Kamp, H. (1981). A theory of truth and semantic representation. *Formal semantics-the essential readings*, pages 189–222.

Kamp, H. and Reyle, U. (2013). *From discourse to logic: Introduction to modeltheoretic semantics of natural language, formal logic and discourse representation theory*, volume 42. Springer Science & Business Media.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Li, Y., Vinyals, O., Dyer, C., Pascanu, R., and Battaglia, P. (2018). Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*.

Liang, P., Jordan, M., and Klein, D. (2011). Learning dependency-based compositional semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 590–599, Portland, Oregon.

Liu, J., Cohen, S. B., and Lapata, M. (2018). Discourse representation structure parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 429–439. Association for Computational Linguistics.

Luong, T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Lyu, C. and Titov, I. (2018). Amr parsing as graph prediction with latent alignment. *arXiv preprint arXiv:1805.05286*.

Montague, R. (1973). The proper treatment of quantification in ordinary English. In Hintikka, K., Moravcsik, J., and Suppes, P., editors, *Approaches to Natural Language*, volume 49 of Synthese Library, pages 221–242. Springer Netherlands.

van Noord, R., Abzianidze, L., Toral, A., and Bos, J. (2018). Exploring neural methods for parsing discourse representation structures. *arXiv preprint arXiv:1810.12579*.

# 75 Languages, 1 Model: Parsing Universal Dependencies Universally

**Dan Kondratyuk**[1,2] and **Milan Straka**[1]
[1]Charles University, Institute of Formal and Applied Linguistics
[2]Saarland University, Department of Computational Linguistics
dankondratyuk@gmail.com, straka@ufal.mff.cuni.cz

## Abstract

We present UDify, a multilingual multi-task model capable of accurately predicting universal part-of-speech, morphological features, lemmas, and dependency trees simultaneously for all 124 Universal Dependencies treebanks across 75 languages. By leveraging a multilingual BERT self-attention model pretrained on 104 languages, we found that fine-tuning it on all datasets concatenated together with simple softmax classifiers for each UD task can meet or exceed state-of-the-art UPOS, UFeats, Lemmas, (and especially) UAS, and LAS scores, without requiring any recurrent or language-specific components. We evaluate UDify for multilingual learning, showing that low-resource languages benefit the most from cross-linguistic annotations. We also evaluate for zero-shot learning, with results suggesting that multilingual training provides strong UD predictions even for languages that neither UDify nor BERT have ever been trained on. Code for UDify is available at https://github.com/hyperparticle/udify.

## 1 Introduction

In the absence of annotated data for a given language, it can be considerably difficult to create models that can parse the language's text accurately. Multilingual modeling presents an attractive way to circumvent this low-resource limitation. In a similar way learning a new language can enhance the proficiency of a speaker's previous languages (Abu-Rabia and Sanitsky, 2010), a model which has access to multilingual information can begin to learn generalizations across languages that would not have been possible through monolingual data alone. Works such as McDonald et al. (2011); Naseem et al. (2012); Duong et al. (2015); Ammar et al. (2016); de Lhoneux et al. (2018); Kitaev and Klein (2018); Mulcaire et al. (2019) consistently demonstrate how pairing the training data of similar languages can boost evaluation scores of models predicting syntactic information like part-of-speech and dependency trees. Multilinguality not only can improve a model's evaluation performance, but can also reduce the cost of training multiple models for a collection of languages (Johnson et al., 2017; Smith et al., 2018).

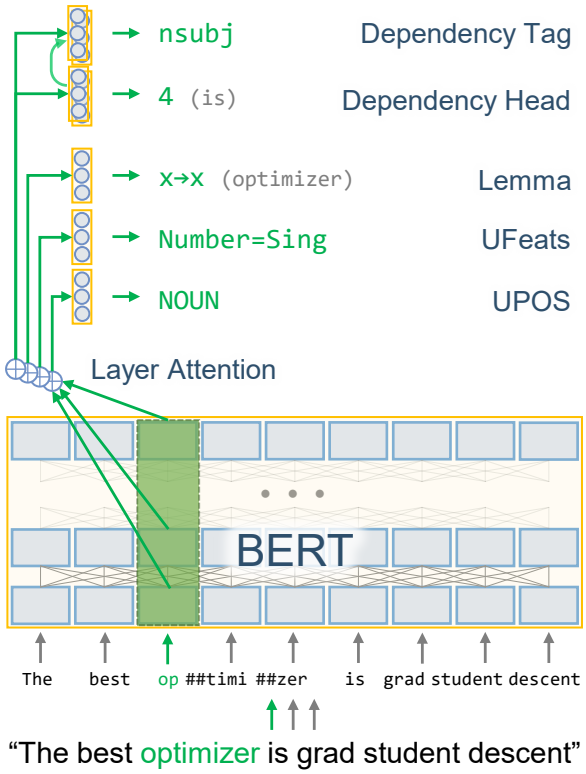


Figure 1: An illustration of the UDify network architecture with task-specific layer attention, inputting word tokens and outputting UD annotations for each token.

However, scaling to a higher number of languages can often be problematic. Without an ample supply of training data for the considered languages, it can be difficult to form appropriate generalizations and especially difficult if those



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2779–2795,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

languages are distant from each other. But recent techniques in language model pretraining can profit from a drastically larger supply of unsupervised text, demonstrating the capability of transferring contextual sentence-level knowledge to boost the parsing accuracy of existing NLP models (Howard and Ruder, 2018; Peters et al., 2018; Devlin et al., 2018).

One such model, BERT (Devlin et al., 2018), introduces a self-attention (Transformer) network that results in state-of-the-art parsing performance when fine-tuning its contextual embeddings. And with the release of a multilingual version pretrained on the entirety of the top 104 resourced languages of Wikipedia, BERT is remarkably capable of capturing an enormous collection of cross-lingual syntactic information. Conveniently, these languages nearly completely overlap with languages supported by the Universal Dependencies treebanks, which we will use to demonstrate the ability to scale syntactic parsing up to 75 languages and beyond.

The Universal Dependencies (UD) framework provides syntactic annotations consistent across a large collection of languages (Nivre et al., 2018; Zeman et al., 2018). This makes it an excellent candidate for analyzing syntactic knowledge transfer across multiple languages. UD offers tokenized sentences with annotations ideal for multi-task learning, including lemmas (LEMMAS), treebank-specific part-of-speech tags (XPOS), universal part-of-speech tags (UPOS), morphological features (UFEATS), and dependency edges and labels (DEPS) for each sentence.

We propose UDify, a semi-supervised multi-task self-attention model automatically producing UD annotations in any of the supported UD languages. To accomplish this, we perform the following:

1. We input all sentences into a pretrained multilingual BERT network to produce contextual embeddings, introduce task-specific layer-wise attention similar to ELMo (Peters et al., 2018), and decode each UD task simultaneously using softmax classifiers.

2. We apply a heavy amount of regularization to BERT, including input masking, increased dropout, weight freezing, discriminative fine-tuning, and layer dropout.

3. We train and fine-tune the model on the entirety of UD by concatenating *all* available training sets together.

We evaluate our model with respect to UDPipe Future, one of the winners of the CoNLL 2018 Shared Task on Multilingual Parsing from Raw Text to Universal Dependencies (Straka, 2018; Zeman et al., 2018). In addition, we analyze languages that multilingual training benefits prediction the most, and evaluate the model for zero-shot learning, i.e., treebanks which do not have a training set. Finally, we provide evidence from our experiments and other related work to help explain why pretrained self-attention networks excel in multilingual dependency parsing.

Our work uses the AllenNLP library built for the PyTorch framework. Code for UDify and a release of the fine-tuned BERT weights are available at https://github.com/hyperparticle/udify.

## 2 Multilingual Multi-Task Learning

In this section, we detail the multilingual training setup and the UDify multi-task model architecture. See Figure 1 for an architecture diagram.

### 2.1 Multilingual Pretraining with BERT

We leverage the provided BERT base multilingual cased pretrained model[1], with a self-attention network of 12 layers, 12 attention heads per layer, and hidden dimensions of 768 (Devlin et al., 2018). The model was trained by predicting randomly masked input words on the entirety of the top 104 languages with the largest Wikipedias. BERT uses a wordpiece tokenizer (Wu et al., 2016), which segments all text into (unnormalized) sub-word units.

### 2.2 Cross-Linguistic Training Issues

Table 1 displays a list of vocabulary sizes, indicating that UD treebanks possess nearly 1.6M unique tokens combined. To sidestep the problem of a ballooning vocabulary, we use BERT's wordpiece tokenizer directly for all inputs. UD expects predictions to be along word boundaries, so we take the simple approach of applying the tokenizer to each word using UD's provided segmentation. For prediction, we use the outputs of BERT corresponding to the first wordpiece per word, ignoring

[1] https://github.com/google-research/bert/blob/master/multilingual.md

| Token | Vocab Size |
|---|---|
| Word Form | 1,588,655 |
| BERT Wordpieces | 119,547 |
| UPOS | 17 |
| XPOS | 19,126 |
| UFeats | 23,974 |
| Lemmas (tags) | 109,639 |
| Deps | 251 |

Table 1: Vocabulary sizes of words and tags over all of UD v2.3, with a total of 12,032,309 word tokens and 668,939 sentences.

the rest[2].

In addition, the XPOS annotations are not universal across languages, or even across treebanks. Because each treebank can possess a different annotation scheme for XPOS which can slow down inference, we omit training and evaluation of XPOS from our experiments.

### 2.3 Multi-Task Learning with UD

For predicting UD annotations, we employ a multi-task network based on UDPipe Future (Straka, 2018), but with all embedding, encoder, and projection layers replaced with BERT. The remaining components include the prediction layers for each task detailed below, and layer attention (see Section 3.1). Then we compute softmax cross entropy loss on the output logits to train the network. For more details on reasons behind architecture choices, see Appendix A.

**UPOS** As is standard for neural sequence tagging, we apply a softmax layer along each word input, computing a probability distribution over the tag vocabulary to predict the annotation string.

**UFeats** Identical to UPOS prediction, we treat each UFeats string as a separate token in the vocabulary. We found this to produce higher evaluation accuracy than predicting each morphological feature separately. Only a small subset of the full Cartesian product of morphological features is valid, eliminating invalid combinations.

**Lemmas** Similar to Chrupała (2006); Müller et al. (2015), we reduce the problem of lemmatization to a sequence tagging problem by predicting a class representing an edit script, i.e., the sequence of character operations to transform the word form to the lemma. To precompute the tags, we first find the longest common substring between the form and the lemma, and then compute the shortest edit script converting the prefix and suffix of the form into the prefix and suffix of the lemma using the Wagner–Fischer algorithm (Wagner and Fischer, 1974). Upon predicting a lemma edit script, we apply the edit operations to the word form to produce the final lemma. See also Straka (2018) for more details. We chose this approach over a sequence-to-sequence architecture like Bergmanis and Goldwater (2018) or Kondratyuk et al. (2018), as this significantly reduces training efficiency.

**Deps** We use the graph-based biaffine attention parser developed by Dozat and Manning (2016); Dozat et al. (2017), replacing the bidirectional LSTM layers with BERT. The final embeddings are projected through arc-head and arc-dep feed-forward layers, which are combined using biaffine attention to produce a probability distribution of arc heads for each word. We then decode each tree with the Chu-Liu/Edmonds algorithm (Chu, 1965; Edmonds, 1967).

## 3 Fine-Tuning BERT on UD Annotations

We employ several strategies for fine-tuning BERT for UD prediction, finding that regularization is absolutely crucial for producing a high-scoring network.

### 3.1 Layer Attention

Empirical results suggest that when fine-tuning BERT, combining the output of the last several layers is more beneficial for the downstream tasks than just using the last layer (Devlin et al., 2018). Instead of restricting the model to any subset of layers, we devise a simple layer-wise dot-product attention where the network computes a weighted sum of all intermediate outputs of the 12 BERT layers using the same weights for each token. This is similar to how ELMo mixes the output of multiple recurrent layers (Peters et al., 2018).

More formally, let $\boldsymbol{w}_i$ be a trainable scalar for BERT embeddings $\mathbf{BERT}_{ij}$ at layer $i$ with a token at position $j$, and let $c$ be a trainable scalar. We compute contextual embeddings $\boldsymbol{e}^{(\text{task})}$ such that

$$\boldsymbol{e}_j^{(\text{task})} = c \sum_i \mathbf{BERT}_{ij} \cdot \text{softmax}(\boldsymbol{w})_i \quad (1)$$

To prevent the UD classifiers from overfitting to the information in any single layer, we devise

[2]We found last, max, or average pooling of the wordpieces were not any better or worse for evaluation. Kitaev and Klein (2018) report similar results.

**layer dropout**, where at each training step, we set each parameter $\boldsymbol{w}_i$ to $-\infty$ with probability 0.1. This effectively redistributes probability mass to all other layers, forcing the network to incorporate the information content of all BERT layers. We compute layer attention per task, using one set of $c, \boldsymbol{w}$ parameters for each of UPOS, UFeats, Lemmas, and Deps.

### 3.2 Transfer Learning with ULMFiT

The ULMFiT strategy defines several useful methods for fine-tuning a network on a pretrained language model (Howard and Ruder, 2018). We apply the same methods, with a few minor modifications.

We split the network into two parameter groups, i.e., the parameters of BERT and all other parameters. We apply discriminative fine-tuning, setting the base learning rate of BERT to be $5e^{-5}$ and $1e^{-3}$ everywhere else. We also freeze the BERT parameters for the first epoch to increase training stability.

While ULMFiT recommends decaying the learning rate linearly after a linear warmup, we found that this is prone to training divergence in self-attention networks, introducing vanishing gradients and underfitting. Instead, we apply an inverse square root learning rate decay with linear warmup (Noam) seen in training Transformer networks for machine translation (Vaswani et al., 2017).

### 3.3 Input Masking

The authors of BERT recommend not to mask words randomly with `[MASK]` when fine-tuning the network. However, we discovered that masking often reduces the tendency of the classifiers to overfit to BERT by forcing the network to rely on the context of surrounding words. This *word dropout* strategy has been observed in other works showing improved test performance on a variety of NLP tasks (Iyyer et al., 2015; Bowman et al., 2016; Clark et al., 2018; Straka, 2018).

## 4 Experiments

We evaluate UDify with respect to every test set in each treebank. As there are too many results to fit within one page, we display a salient subset of scores and compare them with UDPipe Future. The full results are listed in Appendix A.

We do not directly reference metrics from other models in the CoNLL 2018 Shared Task, as the tables of results do not assume gold word segmentation and may not provide a fair comparison. Instead, we retrained the open source UDPipe Future model using gold segmentation and report results here due to its architectural similarity to UDify and its strong performance.

Note that the UDPipe Future baseline does not itself use BERT. Evaluation of BERT utilization in UDPipe Future can be found in Straka et al. (2019).

### 4.1 Datasets

For all experiments, we use the full Universal Dependencies v2.3 corpus available on LINDAT (Nivre et al., 2018). We omit the evaluation of datasets that do not have their training annotations freely available, i.e., Arabic NYUAD (ar_nyuad), English ESL (en_esl), French FTB (fr_ftb), Hindi English HEINCS (qhe_heincs), and Japanese BCCWJ (ja_bccwj).

To train the multilingual model, we concatenate all available training sets together, similar to McDonald et al. (2011). Before each epoch, we shuffle all sentences and feed mixed batches of sentences to the network, where each batch may contain sentences from any language or treebank, for a total of 80 epochs[3].

### 4.2 Hyperparameters

A summary of hyperparameters can be found in Table 6 in Appendix A.1.

### 4.3 Probing for Syntax

Hewitt and Manning (2019) introduce a structural probe for identifying dependency structures in contextualized word embeddings. This probe evaluates whether syntax trees (i.e., unlabeled undirected dependency trees) can be easily extracted as a global property of the embedding space using a linear transformation of the network's contextual word embeddings. The probe trains a weighted adjacency matrix on the layers of contextual embeddings produced by BERT, identifying a linear transformation where squared L2 distance between embedding vectors encodes the distance between words in the parse tree. Edges are decoded by computing the minimum spanning tree on the weight matrix (the lowest sum of edge distances).

[3] We train on a GTX 1080 Ti for approximately 25 days. See Appendix A.1 for more details

| Treebank | Model | UPOS | Feats | Lem | UAS | LAS |
|---|---|---|---|---|---|---|
| Czech PDT (cs_pdt) | UDPipe | 99.18 | 97.23 | **99.02** | 93.33 | 91.31 |
| | Lang | 99.18 | 96.87 | 98.72 | 94.35 | 92.41 |
| | UDify | 99.18 | 96.85 | 98.56 | 94.73 | 92.88 |
| | UDify+Lang | **99.24** | **97.44** | 98.93 | **95.07** | **93.38** |
| German GSD (de_gsd) | UDPipe | 94.48 | 90.68 | **96.80** | 85.53 | 81.07 |
| | Lang | 94.77 | 91.73 | 96.34 | 87.54 | 83.39 |
| | UDify | 94.55 | 90.65 | 94.82 | 87.81 | 83.59 |
| | UDify+Lang | **95.29** | **91.94** | 96.74 | **88.11** | **84.13** |
| English EWT (en_ewt ) | UDPipe | 96.29 | 97.10 | **98.25** | 89.63 | 86.97 |
| | Lang | **96.82** | **97.27** | 97.97 | **91.70** | **89.38** |
| | UDify | 96.21 | 96.17 | 97.35 | 90.96 | 88.50 |
| | UDify+Lang | 96.57 | 96.96 | 97.90 | 91.55 | 89.06 |
| Spanish AnCora (es_ancora) | UDPipe | **98.91** | **98.49** | **99.17** | 92.34 | 90.26 |
| | Lang | 98.60 | 98.14 | 98.52 | 92.82 | 90.52 |
| | UDify | 98.53 | 97.84 | 98.09 | 92.99 | 90.50 |
| | UDify+Lang | 98.68 | 98.25 | 98.68 | **93.35** | **91.28** |
| French GSD (fr_gsd) | UDPipe | 97.63 | **97.13** | **98.35** | 90.65 | 88.06 |
| | Lang | **98.05** | 96.26 | 97.96 | 92.77 | 90.61 |
| | UDify | 97.83 | 96.59 | 97.48 | **93.60** | **91.45** |
| | UDify+Lang | 97.96 | 96.73 | 98.17 | 93.56 | 91.45 |
| Russian SynTagRus (ru_syntagrus) | UDPipe | **99.12** | **97.57** | **98.53** | 93.80 | 92.32 |
| | Lang | 98.90 | 96.58 | 95.16 | 94.40 | 92.72 |
| | UDify | 98.97 | 96.35 | 94.43 | 94.83 | 93.13 |
| | UDify+Lang | 99.08 | 97.22 | 96.58 | **95.13** | **93.70** |
| Belarusian HSE (be_hse) | UDPipe | 93.63 | 73.30 | 87.34 | 78.58 | 72.72 |
| | Lang | 95.88 | 76.12 | 84.52 | 83.94 | 79.02 |
| | UDify | **97.54** | **89.36** | 85.46 | **91.82** | **87.19** |
| | UDify+Lang | 97.25 | 85.02 | **88.71** | 90.67 | 86.98 |
| Buryat BDT (bxr_bdt) | UDPipe | 40.34 | 32.40 | 58.17 | 32.60 | 18.83 |
| | Lang | 52.54 | 37.03 | 54.64 | 29.63 | 15.82 |
| | UDify | **61.73** | **47.86** | **61.06** | **48.43** | **26.28** |
| | UDify+Lang | 61.73 | 42.79 | 58.20 | 33.06 | 18.65 |
| Upper Sorbian UFAL (hsb_ufal) | UDPipe | 62.93 | 41.10 | 68.68 | 45.58 | 34.54 |
| | Lang | 73.70 | 46.28 | 58.02 | 39.02 | 28.70 |
| | UDify | 84.87 | 48.63 | **72.73** | **71.55** | **62.82** |
| | UDify+Lang | **87.58** | **53.19** | 71.88 | 71.40 | 60.65 |
| Kazakh KTB (kk_ktb) | UDPipe | 55.84 | 40.40 | 63.96 | 53.30 | 33.38 |
| | Lang | 73.52 | 46.60 | 57.84 | 50.38 | 32.61 |
| | UDify | **85.59** | **65.14** | **77.40** | **74.77** | **63.66** |
| | UDify+Lang | 81.32 | 60.50 | 67.30 | 69.16 | 53.14 |
| Lithuanian HSE (lt_hse) | UDPipe | 81.70 | 60.47 | **76.89** | 51.98 | 42.17 |
| | Lang | 83.40 | 54.34 | 58.77 | 51.23 | 38.96 |
| | UDify | **90.47** | **68.96** | 67.83 | **79.06** | **69.34** |
| | UDify+Lang | 84.53 | 56.98 | 58.21 | 58.40 | 39.91 |

Table 2: Test set scores for a subset of high-resource (top) and low-resource (bottom) languages in comparison to UDPipe Future without BERT, with 3 UDify configurations: **Lang**, fine-tune on the treebank. **UDify**, fine-tune on all UD treebanks combined. **UDify+Lang**, fine-tune on the treebank using BERT weights saved from fine-tuning on all UD treebanks combined.

We train the structural probe on unmodified and fine-tuned BERT using the default hyperparameters of Hewitt and Manning (2019) to evaluate whether the representations affected by fine-tuning BERT on dependency trees would more closely match the structure of these trees.

| Model | Configuration | UPOS | Feats | Lem | UAS | LAS |
|---|---|---|---|---|---|---|
| UDPipe | w/o BERT | **93.76** | **91.04** | **94.63** | 84.37 | 79.76 |
| UDify | Task Layer Attn | 93.40 | 88.72 | 90.41 | **85.69** | **80.43** |
| UDify | Global Layer Attn | 93.12 | 87.53 | 89.03 | 85.07 | 79.49 |
| UDify | Sum Layers | 93.02 | 87.20 | 88.70 | 84.97 | 79.33 |

Table 3: Ablation comparing the average of scores over all treebanks: task-specific layer attention (4 sets of $c, \boldsymbol{w}$ computed for the 4 UD tasks), global layer attention (one set of $c, \boldsymbol{w}$ for all tasks), and simple sum of layers ($c = 1$ and $\boldsymbol{w} = \mathbf{1}$).

| Treebank | | UPOS | Feats | Lem | UAS | LAS |
|---|---|---|---|---|---|---|
| **Breton KEB** | **br_keb** | 63.67 | 46.75 | 53.15 | 63.97 | 40.19 |
| **Tagalog TRG** | **tl_trg** | 61.64 | 35.27 | 75.00 | 64.73 | 39.38 |
| Faroese OFT | fo_oft | 77.86 | 35.71 | 53.82 | 69.28 | 61.03 |
| Naija NSC | pcm_nsc | 56.59 | 52.75 | 97.52 | 47.13 | 33.43 |
| Sanskrit UFAL | sa_ufal | 40.21 | 18.45 | 37.60 | 41.73 | 19.80 |

Table 4: Test set results for zero-shot learning, i.e., no UD training annotations available. Languages that are pretrained with BERT are bolded.

| Treebank | Model | UUAS |
|---|---|---|
| English EWT (en_ewt) | BERT | 65.48 |
| | BERT+finetune en_ewt | **79.67** |

Table 5: UUAS test scores calculated on the predictions produced by the syntactic structural probe (Hewitt and Manning, 2019) using the English EWT treebank, on the unmodified multilingual cased BERT model and the same BERT model fine-tuned on the treebank.

## 5 Results

We show scores of UPOS, UFeats (Feats), and Lemma (Lem) accuracies, along with unlabeled and labeled attachment scores (UAS, LAS) evaluated using the offical CoNLL 2018 Shared Task evaluation script.[4] Results for a salient subset of high-resource and low-resource languages are shown in Table 2, with a comparison between UDPipe Future and UDify fine-tuning on all languages. In addition, the table compares UDify with fine-tuning on either a single language or both languages (fine-tuning multilingually, then fine-tuning on the language with the saved multilingual weights) to provide a reference point for multilingual influences on UDify. We provide

[4] https://universaldependencies.org/conll18/evaluation.html

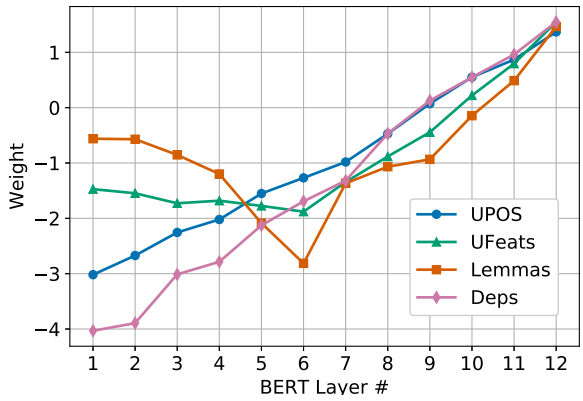


Figure 2: The unnormalized BERT layer attention weights $\boldsymbol{w}_i$ contributing to layer $i$ for each task after training. A linear change in weight scales each BERT layer exponentially due to the softmax in Equation 1

a full table of scores for all treebanks in Appendix A.4.

A more comprehensive overview is shown in Table 3, comparing different attention strategies applied to UDify. We display an average of scores over all (89) treebanks with a training set. For zero-shot learning evaluation, Table 4 displays a subset of test set evaluations of treebanks that do not have a training set, i.e., Breton, Tagalog, Faroese, Naija, and Sanskrit. We plot the layer attention weights $\boldsymbol{w}$ after fine-tuning BERT in Figure 2, showing a set of weights per task. And Table 5 compares the unlabeled undirected attachment scores (UUAS) of dependency trees produced using a structural probe on both the unmodified multilingual cased BERT model and the extracted BERT model fine-tuned on the English EWT treebank.

## 6 Discussion

In this section, we discuss the most notable features of the results.

### 6.1 Model Performance

On average, UDify reveals a strong set of results that are comparable in performance with the state-of-the-art in parsing UD annotations. UDify excels in dependency parsing, exceeding UDPipe Future by a large margin especially for low-resource languages. UDify slightly underperforms with respect to Lemmas and Universal Features, likely due to UDPipe Future additionally using character-level embeddings (Santos and Zadrozny, 2014; Ling et al., 2015; Ballesteros et al., 2015; Kim et al., 2016), while (for simplicity) UDify does not. Additionally, UDify severely underperforms the baseline on a few low-resource languages, e.g., cop_scriptorum. We surmise that this is due to using mixed batches on an unbalanced training set, which skews the model towards predicting larger treebanks more accurately. However, we find that fine-tuning on the treebank individually with BERT weights saved from UDify eliminates most of these gaps in performance.

Echoing results seen in Smith et al. (2018), UDify also shows strong improvement leveraging multilingual data from other UD treebanks. In low-resource cases, fine-tuning BERT on all treebanks can be far superior to fine-tuning monolingually. A second round of fine-tuning on an individual treebank using UDify's BERT weights can improve this further, especially for treebanks that underperform the baseline. However, for languages that are already display strong results, we typically notice worse evaluation performance across all the evaluation metrics. This indicates that multilingual fine-tuning really is superior to single language fine-tuning with respect to these high-performing languages, showing improvements of up to 20% reduction in error.

Interestingly, Slavic languages tend to perform the best with multilingual training. While languages like Czech and Russian possess the largest UD treebanks and do not differ as much in performance from monolingual fine-tuning, evidenced by the improvements over single-language fine-tuning, we can see a large degree of morphological and syntactic structure has transferred to low-resource Slavic languages like Upper Sorbian, whose treebank contains only 646 sentences. But this is not only true of Slavic languages, as the Turkic language Kazakh (with less than 1,000 training sentences) has also improved significantly.

The zero-shot results indicate that fine-tuning on BERT can result in reasonably high scores on languages that do not have a training set. It can be seen that a combination of BERT pretraining and multilingual learning can improve predictions for Breton and Tagalog, which implies that the network has learned representations of syntax that cross lingual boundaries. Furthermore, despite the fact that neither BERT nor UDify have directly observed Faroese, Naija, or Sanskrit, we see unusually high performance in these languages. This can be partially attributed to each language closely resembling another: Faroese is very close to Ice-

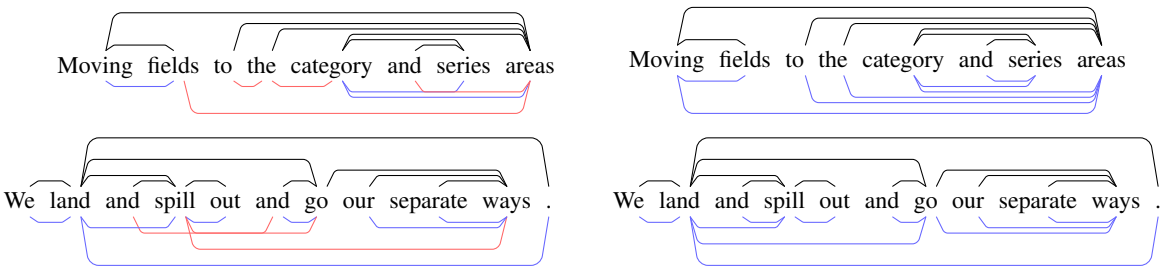


Figure 3: Examples of minimum spanning trees produced by the syntactic probe are shown below each sentence, evaluated on BERT (left) and on UDify (right). Gold dependency trees are shown above each sentence in black. Matched and unmatched spanning tree edges are shown in blue and red respectively.

landic, Naija (Nigerian Pidgin) is a variant of English, and Sanskrit is an ancient Indian language related to Greek, Latin, and Hindi.

Table 3 shows that layer attention on BERT for each task is beneficial for test performance, much more than using a global weighted average. In fact, Figure 2 shows that each task prefers the layers of BERT differently, uniquely extracting the optimal information for a task. All tasks favor the information content in the last 3 layers, with a tendency to disprefer layers closer to the input. However, an interesting observation is that for Lemmas and UFeats, the classifier prefers to also incorporate the information of the first 3 layers. This meshes well with the linguistic intuition that morphological features are more closely related to the surface form of a word and rely less on context than other syntactic tasks. Curiously enough, the middle layers are highly dispreferred, meaning that the most useful processing for multilingual syntax (tagging, dependency parsing) occurs in the last 3-4 layers. The results released by Tenney et al. (2019) also agree with the intuition behind the weight distribution above, showing how the different layers of BERT generate hierarchical information like a traditional NLP pipeline, starting with low-level syntax (e.g., POS tagging) and building up to high-level syntactic and semantic dependency parsing.

### 6.2 Effect of Syntactic Fine-Tuning on BERT

Even without any supervised training, BERT encodes its syntax in the embedding's distance close to human-annotated dependencies. But more notably, the results in Table 5 show that fine-tuning BERT on Universal Dependencies significantly boosts UUAS scores when compared to the gold dependency trees, an error reduction of 41%. This indicates that the self-attention weights have learned a linearly-transformable representation of its vectors more closely resembling annotated dependency trees defined by linguists. Even with just unsupervised pretraining, a global structural property of the vector space of the BERT weights already produces a decent representation of the dependency tree in the squared L2 distance. Following this, it should be no surprise that training with a non-linear graph-based dependency decoder would produce even higher quality dependency trees.

### 6.3 Attention Visualization

We performed a high-level visual analysis of the BERT attention weights to see if they have changed on any discernible level. Our observations reveal something notable: the attention weights tend to be more sparse, and are more often sensitive to constituent boundaries like clauses and prepositions. Figure 4 illustrates this point, showing the attention weights of a particular attention head on an example sentence. We find similar behavior in 13 additional attention heads for the provided example sentence.

We see that some of the attention structure remains after fine-tuning. Previously, the attention head was mostly sensitive to previous words and punctuation. But after fine-tuning, it demonstrates more fine-grained attention towards immediate wordpieces, prepositions, articles, and adjectives. We found similar evidence in other attention heads, which implies that fine-tuning on UD produces attention that more closely resembles localized dependencies within constituents. We also find that BERT base heavily preferred to attend to punctuation, while UDify BERT does to a much lesser degree.

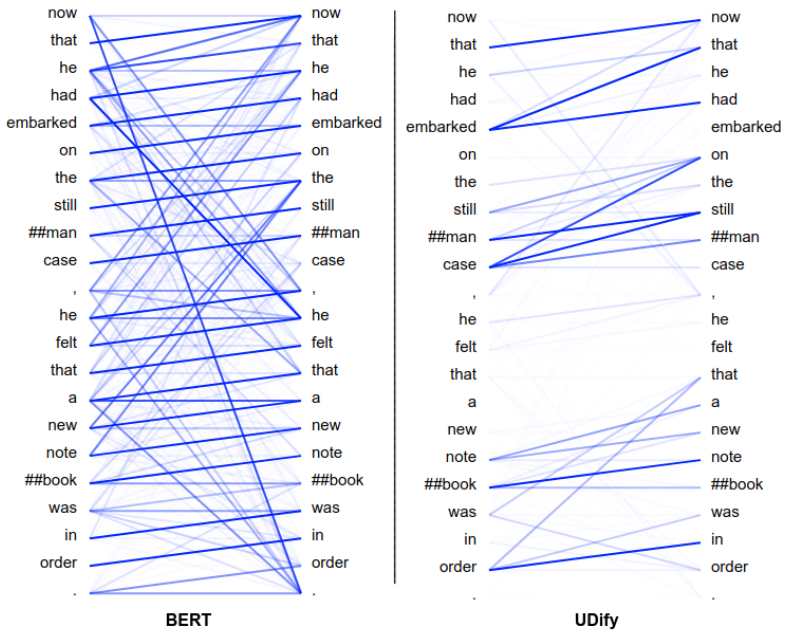


Figure 4: Visualization of BERT attention head 4 at layer 11, comparing the attended words on an English sentence between BERT base and UDify BERT after fine-tuning. The right column indicates the attended words (keys) with respect to the words in the left column (queries). Darker lines indicate stronger attention weights.

### 6.4 Factors that Enable BERT to Excel at Dependency Parsing and Multilinguality

Goldberg (2019) assesses the syntactic capabilities of BERT and concludes that BERT is remarkably capable of processing syntactic tasks despite not being trained on any supervised data. Conducting similar experiments, Vig (2019) and Sileo (2019) visualize the attention heads within each BERT layer, showing a number of distinct attention patterns, including attending to previous/next words, related words, punctuation, verbs/nouns, and coreference dependencies.

This neat delegation of certain low-level information processing tasks to the attention heads hints at why BERT might excel at processing syntax. We see that from the analysis on BERT fine-tuned with syntax using the syntactic probe and attention visualization, BERT produces a representation that keeps constituents close in its vector space, and improves this representation to more closely resemble human annotated dependency trees when fine-tuned on UD as seen in Figure 3. Furthermore, Ahmad et al. (2018) provide results consistent with their claim that self-attention networks can be more robust than recurrent networks to the change of word order, observing that self-attention networks capture less word order information in their architecture, which is what allows them to generally perform better at cross-lingual parsing. Wu and Dredze (2019) also analyze multilingual BERT and report that the model retains both language-independent as well as language-specific information related to each input sentence, and that the shared embedding space with the input wordpieces correlates strongly with cross-lingual generalization.

From the evidence above, we can see that the combination of strong regularization paired with the ability to capture long-range dependencies with self-attention and contextual pretraining on an enormous corpus of raw text are large contributors that enable robust multilingual modeling with respect to dependency parsing. Pretraining self-attention networks introduces a strong syntactic bias that is capable of generalizing across languages. The dependencies seen in the output dependency trees are highly correlated with the implicit dependencies learned by the self-attention, showing that self-attention is remarkably capable of modeling syntax by picking up on common syntactic patterns in text. The introduction of multilingual data also shows that these attention heads provide a surprising amount of capacity that do not degrade the performance considerably when compared to monolingual training. E.g., Devlin et al. (2018) report that the fine-tuning on the multilingual BERT model results in a small degradation in English fine-tune performance with 104

pretrained languages compared to an equivalent model pretrained only on English. This also hints that the BERT model can be compressed significantly without compromising heavily on evaluation performance.

## 7 Related Work

This work's main contribution in combining treebanks for multilingual UD parsing is most similar to the Uppsala system for the CoNLL 2018 Shared Task (Smith et al., 2018). Uppsala combines treebanks of one language or closely related languages together over 82 treebanks and parses all UD annotations in a multi-task pipeline architecture for a total of 34 models. This approach reduces the number of models required to parse each language while also showing results that are no worse than training on each treebank individually, and in especially low-resource cases, significantly improved. Combining UD treebanks in a language-agnostic way was first introduced in Vilares et al. (2016), which train bilingual parsers on pairs of UD treebanks, showing similar improvements.

Other efforts in training multilingual models include Johnson et al. (2017), which demonstrate a machine translation model capable of supporting translation between 12 languages. Recurrent models have also shown to be capable of scaling to a larger number of languages as seen in Artetxe and Schwenk (2018), which define a scalable approach to train massively multilingual embeddings using recurrent networks on an auxiliary task, e.g., natural language inference. Schuster et al. (2019) produce context-independent multilingual embeddings using a novel embedding alignment strategy to allow models to improve the use of cross-lingual information, showing improved results in dependency parsing.

## 8 Conclusion

We have proposed and evaluated UDify, a multilingual multi-task self-attention network fine-tuned on BERT pretrained embeddings, capable of producing annotations for any UD treebank, and exceeding the state-of-the-art in UD dependency parsing in a large subset of languages while being comparable in tagging and lemmatization accuracy. Strong regularization and task-specific layer attention are highly beneficial for fine-tuning, and coupled with training multilingually, also reduce the number of required models to train down to one. Multilingual learning is most beneficial for low-resource languages, even ones that do not possess a training set, and can be further improved by fine-tuning monolingually using BERT weights saved from UDify's multilingual training. All these results indicate that self-attention networks are remarkably capable of capturing syntactic patterns, and coupled with unsupervised pretraining are able to scale to a large number of languages without degrading performance.

## Acknowledgments

The work described herein has been supported by OP VVV VI LINDAT/CLARIN project of the Ministry of Education, Youth and Sports of the Czech Republic (project CZ.02.1.01/0.0/0.0/16_013/0001781) and it has been supported and has been using language resources developed by the LINDAT/CLARIN project of the Ministry of Education, Youth and Sports of the Czech Republic (project LM2015071).

Dan Kondratyuk has been supported by the Erasmus Mundus program in Language & Communication Technologies (LCT), and by the German Federal Ministry of Education and Research (BMBF) through the project DEEPLEE (01IW17001).

## References

Salim Abu-Rabia and Ekaterina Sanitsky. 2010. Advantages of bilinguals over monolinguals in learning a third language. *Bilingual Research Journal*, 33(2):173–199.

Wasi Uddin Ahmad, Zhisong Zhang, Xuezhe Ma, Eduard Hovy, Kai-Wei Chang, and Nanyun Peng. 2018. On difficulties of cross-lingual transfer with order differences: A case study on dependency parsing. *arXiv preprint arXiv:1811.00570*.

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A Smith. 2016. Many languages, one parser. *Transactions of the Association for Computational Linguistics*, 4:431–444.

Mikel Artetxe and Holger Schwenk. 2018. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *arXiv preprint arXiv:1812.10464*.

Miguel Ballesteros, Chris Dyer, and Noah A Smith. 2015. Improved transition-based parsing by modeling characters instead of words with lstms. In *Proceedings of the 2015 Conference on Empirical Meth-*

*ods in Natural Language Processing*, pages 349–359.

Toms Bergmanis and Sharon Goldwater. 2018. Context sensitive neural lemmatization with lematus. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 1391–1400.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. *CoNLL 2016*, page 10.

Grzegorz Chrupała. 2006. Simple data-driven context-sensitive lemmatization. *Procesamiento del Lenguaje Natural*, 37.

Yoeng-Jin Chu. 1965. On the shortest arborescence of a directed graph. *Scientia Sinica*, 14:1396–1400.

Kevin Clark, Minh-Thang Luong, Christopher D Manning, and Quoc V Le. 2018. Semi-supervised sequence modeling with cross-view training. *arXiv preprint arXiv:1809.08370*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Timothy Dozat and Christopher D Manning. 2016. Deep biaffine attention for neural dependency parsing. *arXiv preprint arXiv:1611.01734*.

Timothy Dozat, Peng Qi, and Christopher D Manning. 2017. Stanford's graph-based neural dependency parser at the conll 2017 shared task. *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 20–30.

Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 845–850.

Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the national Bureau of Standards B*, 71(4):233–240.

Yoav Goldberg. 2019. Assessing bert's syntactic abilities. *arXiv preprint arXiv:1901.05287*.

John Hewitt and Christopher D Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 328–339.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1681–1691.

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2017. Googles multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Nikita Kitaev and Dan Klein. 2018. Multilingual constituency parsing with self-attention and pre-training. *arXiv preprint arXiv:1812.11760*.

Daniel Kondratyuk, Tomáš Gavenčiak, Milan Straka, and Jan Hajič. 2018. Lemmatag: Jointly tagging and lemmatizing for morphologically rich languages with brnns. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4921–4928.

Miryam de Lhoneux, Johannes Bjerva, Isabelle Augenstein, and Anders Søgaard. 2018. Parameter sharing between dependency parsers for related languages. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4992–4997.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1520–1530.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the conference on empirical methods in natural language processing*, pages 62–72. Association for Computational Linguistics.

Phoebe Mulcaire, Jungo Kasai, and Noah Smith. 2019. Polyglot contextual representations improve crosslingual transfer. *arXiv preprint arXiv:1902.09697*.

Thomas Müller, Ryan Cotterell, Alexander Fraser, and Hinrich Schütze. 2015. Joint lemmatization and morphological tagging with lemming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2268–2274.

Tahira Naseem, Regina Barzilay, and Amir Globerson. 2012. Selective sharing for multilingual dependency parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 629–637. Association for Computational Linguistics.

Joakim Nivre, Mitchell Abrams, Željko Agić, and Ahrenberg. 2018. Universal dependencies 2.3. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Cicero D Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1818–1826.

Tal Schuster, Ori Ram, Regina Barzilay, and Amir Globerson. 2019. Cross-lingual alignment of contextual word embeddings, with applications to zero-shot dependency parsing. *arXiv preprint arXiv:1902.09492*.

Damien Sileo. 2019. Understanding bert transformer: Attention isnt all you need. *Towards Data Science*.

Aaron Smith, Bernd Bohnet, Miryam de Lhoneux, Joakim Nivre, Yan Shao, and Sara Stymne. 2018. 82 treebanks, 34 models: Universal dependency parsing with multi-treebank models. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 113–123, Brussels, Belgium. Association for Computational Linguistics.

Milan Straka. 2018. UDPipe 2.0 prototype at CoNLL 2018 UD shared task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 197–207, Brussels, Belgium. Association for Computational Linguistics.

Milan Straka, Jana Straková, and Jan Hajič. 2019. Evaluating Contextualized Embeddings on 54 Languages in POS Tagging, Lemmatization and Dependency Parsing. *arXiv preprint arXiv:1908.07448*.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. Bert rediscovers the classical nlp pipeline. *arXiv preprint arXiv:1905.05950*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Jesse Vig. 2019. Visualizing attention in transformer-based language models. *arXiv preprint arXiv:1904.02679*.

David Vilares, Carlos Gómez-Rodríguez, and Miguel A Alonso. 2016. One model, two languages: training bilingual parsers with harmonized treebanks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 425–431.

Robert A Wagner and Michael J Fischer. 1974. The string-to-string correction problem. *Journal of the ACM (JACM)*, 21(1):168–173.

Shijie Wu and Mark Dredze. 2019. Beto, bentz, becas: The surprising cross-lingual effectiveness of bert. *arXiv preprint arXiv:1904.09077*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Daniel Zeman, Jan Hajič, Martin Popel, Martin Potthast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. 2018. CoNLL 2018 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–21, Brussels, Belgium. Association for Computational Linguistics.

# A Appendix

In this section, we detail and explain hyperparameter choices and miscellaneous details related to model training and display the full tables of evaluation results of UDify across all UD languages.

## A.1 Hyperparameters

Upon concatenating all training sets, we shuffle all the sentences, bundle them into batches of 32 sentences each, and train UDify for a total of 80 epochs before stopping. We hold the learning rate constant until we unfreeze BERT in the second epoch, where we and linearly warm up the learning rate for the next 8,000 batches and then apply inverse square root learning rate decay for the remaining epochs. For the dependency parser, we use feedforward tag and arc dimensions of 300

| HYPERPARAMETER | VALUE |
|---|---|
| Dependency tag dimension | 256 |
| Dependency arc dimension | 768 |
| Optimizer | Adam |
| $\beta_1, \beta_2$ | 0.9, 0.99 |
| Weight decay | 0.01 |
| Label Smoothing | 0.03 |
| Dropout | 0.5 |
| BERT dropout | 0.2 |
| Mask probability | 0.2 |
| Layer dropout | 0.1 |
| Batch size | 32 |
| Epochs | 80 |
| Base learning rate | $1e^{-3}$ |
| BERT learning rate | $5e^{-5}$ |
| Learning rate warmup steps | 8000 |
| Gradient clipping | 5.0 |

Table 6: A summary of model hyperparameters.

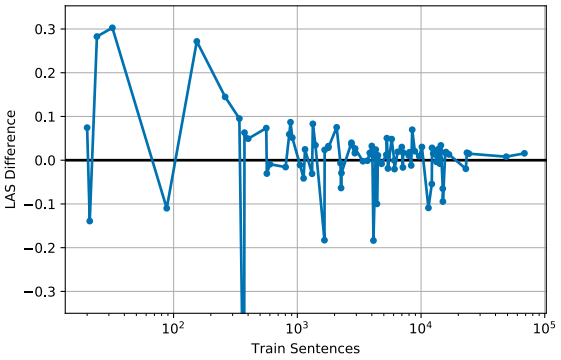


Figure 5: A plot of the difference in LAS between UDify and UDPipe Future with respect to the number of training sentences in each treebank.

and 800 respectively. We apply a small weight decay penalty of 0.01 to ensure that the weights remain small after each update. For optimization we use the Adam optimizer and we compute softmax cross entropy loss to train the network. We use a default $\beta_1$ value of 0.9 and lower the $\beta_2$ value from the typical 0.999 to 0.99. The reasoning is to increase the decay rate of the second moment in the Adam optimizer to reduce the chance of the optimizer being too optimistic with respect to the gradient history. We clip the gradient updates to a maximum L2 magnitude of 5.0. A summary of hyperparameters can be found in Table 6.

To speed up training, we employ bucketed batching, sorting all sentences by their length and grouping similar length sentences into each batch. However, to ensure that most sentences do not get grouped within the same batch, we fuzz the lengths of each sentence by a maximum of 10% of its true length when grouping sentences together.

Despite using all the regularization strategies shown previously, we still observe overfitting and must apply more aggressive techniques. To further regularize the network, we also increase the attention and hidden dropout rates of BERT from 0.1 to 0.2, and we also apply a dropout rate of 0.5 to all BERT layers before computing layer attention for each of the four tasks and applying a layer dropout with probability 0.1. We increase the masking probability of each wordpiece from 0.15 to 0.2.

With all these regularization strategies and hyperparameter choices combined, we are able to fine-tune BERT for far more epochs before the network starts to overfit, i.e., 80 as opposed to around 10. Even so, we believe even more regularization can improve test performance.

The final multilingual UDify model was trained over approximately 25 days on an NVIDIA GTX 1080 Ti taking an average of 8 hours per epoch. We use half-precision (fp16) training to be able to keep the BERT model in memory. One notable aspect of training is that while we observed the model start to level out in validation performance at around epoch 30, the model continually made small, incremental improvements over each subsequent epoch, resulting in far higher scores than if the model training was terminated early. This can be partially attributed to the decaying inverse square root learning rate.

Due to the high training times, we are only able to report on a small number of training experiments for the most relevant and useful results. Prior to developing the final model, we conducted fine-tuning experiments on pairs of languages to find a set of hyperparameters that worked best for multilingual learning. After this, we gradually scaled up training to 3 languages, 5 languages, 15 languages, and then finally the model presented above. We had high doubts, and wanted to see where the limit was in multilingual training. We were pleasantly surprised to find that this simple training scheme was able to scale up so well to all UD treebanks.

### A.2 Training Size Effect on Performance

To gain a better understanding of where the largest score improvements in UDify occur, we plot the LAS improvement UDify provides over UDPipe Future for each treebank, ordered by the size (number of sentences) of the training set, see Fig-

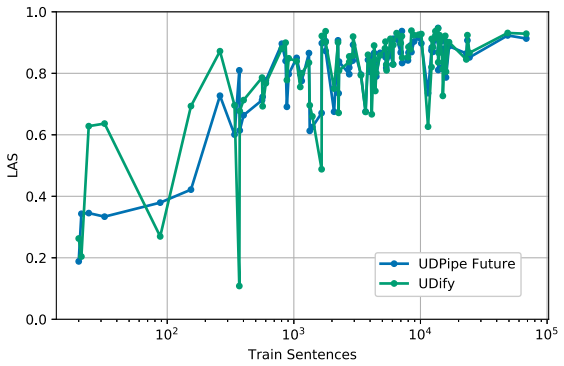


Figure 6: A plot of LAS between with respect to the number of training sentences in each treebank.

ure 5. The results show that the largest improvements tend to occur on small treebanks with less than 3,000 training examples. For absolute LAS values, see Figure 6, which indicates that more training resources tend to improve evaluation performance overall.

### A.3 Miscellaneous Details

Our results show that modeling language-specific properties is not strictly necessary to achieve high-performing cross-lingual representations for dependency parsing, though we caution that the model can also likely be improved by these techniques.

Fine-tuning BERT on UD introduces a syntactic bias in the network, and we are interested in observing any differences in transfer learning by fine-tuning this new "UD-BERT" on other tasks. We leave a comprehensive evaluation of injecting syntactic bias into language models with respect to knowledge transfer for future work.

We note that saving the weights of BERT and fine-tuning a second round can improve performance as demonstrated in Stickland et al. (2019). The improvements of UDify+Lang over just UDify can be partially attributed to this, but we can see that even these improvements can be inferior to fine-tuning on all UD treebanks.

BERT limits its positional encoding to 512 wordpieces, causing some sentences in UD to be too long to fit into the model. We use a sliding window approach to break up long sentences into windows of 512 wordpieces, overlapping each window by 256 wordpieces. After feeding the windows into BERT, we select the first 256 wordpieces of each window and any remaining wordpieces in the last window to represent the contextual embeddings of each word in the original sentence.

### A.4 Full Results of UD Scores

We show in Tables 7, 8, 9, and 10 UDify scores evaluated on all 124 treebanks with the official CoNLL 2018 Shared Task evaluation script. For comparison, we also include the full test evaluation of UDPipe Future on the subset of 89 treebanks with a training set. We also add a column indicating the size of each treebank, i.e., the number of sentences in the training set.

| TREEBANK | MODEL | UPOS | UFEATS | LEMMAS | UAS | LAS | CLAS | MLAS | BLEX | SIZE |
|---|---|---|---|---|---|---|---|---|---|---|
| Afrikaans AfriBooms | UDPipe | **98.25** | **97.66** | **97.46** | **89.38** | **86.58** | **81.44** | **77.66** | **77.82** | 1.3k |
| | UDify | 97.48 | 96.63 | 95.23 | 86.97 | 83.48 | 77.42 | 70.57 | 70.93 | 1.3k |
| Akkadian PISANDUB | UDify | **19.92** | **99.51** | **2.32** | **27.65** | **4.54** | **3.27** | **1.04** | **0.30** | 0 |
| Amharic ATT | UDify | **15.25** | **43.95** | **58.04** | **17.38** | **3.49** | **4.88** | **0.23** | **2.53** | 0 |
| Ancient Greek PROIEL | UDPipe | **97.86** | **92.44** | **93.51** | **85.93** | **82.11** | **77.70** | **67.16** | **71.22** | 15.0k |
| | UDify | 91.20 | 82.29 | 76.16 | 78.91 | 72.66 | 66.07 | 50.79 | 47.27 | 15.0k |
| Ancient Greek Perseus | UDPipe | **93.27** | **91.39** | **85.02** | **78.85** | **73.54** | **67.60** | **53.87** | **53.19** | 11.5k |
| | UDify | 85.67 | 81.67 | 70.51 | 70.51 | 62.64 | 55.60 | 39.15 | 35.05 | 11.5k |
| Arabic PADT | UDPipe | **96.83** | **94.11** | **95.28** | 87.54 | **82.94** | **79.77** | **73.92** | **75.87** | 6.1k |
| | UDify | 96.58 | 91.77 | 73.55 | **87.72** | 82.88 | 79.47 | 70.52 | 50.26 | 6.1k |
| Arabic PUD | UDify | **79.98** | **40.32** | **0.00** | **76.17** | **67.07** | **65.10** | **10.67** | **0.00** | 0 |
| Armenian ArmTDP | UDPipe | 93.49 | **82.85** | **92.86** | 78.62 | 71.27 | 65.77 | **48.11** | **60.11** | 561 |
| | UDify | **94.42** | 76.90 | 85.63 | **85.63** | **78.61** | **73.72** | 46.80 | 59.14 | 561 |
| Bambara CRB | UDify | **30.86** | **57.96** | **20.42** | **30.28** | **8.60** | **6.56** | **1.04** | **0.76** | 0 |
| Basque BDT | UDPipe | **96.11** | **92.48** | **96.29** | **86.11** | **82.86** | **81.79** | **72.33** | **78.54** | 5.4k |
| | UDify | 95.45 | 86.80 | 90.53 | 84.94 | 80.97 | 79.52 | 63.60 | 71.56 | 5.4k |
| Belarusian HSE | UDPipe | 93.63 | 73.30 | **87.34** | 78.58 | 72.72 | 69.14 | 46.20 | 58.28 | 261 |
| | UDify | **97.54** | **89.36** | 85.46 | **91.82** | **87.19** | **85.05** | **71.54** | **68.66** | 261 |
| Breton KEB | UDify | **62.78** | **47.12** | **51.31** | **63.52** | **39.84** | **35.14** | **4.64** | **16.34** | 0 |
| Bulgarian BTB | UDPipe | **98.98** | **97.82** | **97.94** | 93.38 | 90.35 | 87.01 | **83.63** | **84.42** | 8.9k |
| | UDify | 98.89 | 96.18 | 93.49 | **95.54** | **92.40** | **89.59** | 83.43 | 80.44 | 8.9k |
| Buryat BDT | UDPipe | 40.34 | 32.40 | 58.17 | 32.60 | 18.83 | 12.36 | 1.26 | **6.49** | 20 |
| | UDify | **61.73** | **47.45** | **61.03** | **48.43** | **26.28** | **20.61** | **5.51** | 11.68 | 20 |
| Cantonese HK | UDify | **67.11** | **91.01** | **96.01** | **46.82** | **32.01** | **33.35** | **14.29** | **31.26** | 0 |
| Catalan AnCora | UDPipe | 98.88 | **98.37** | **99.07** | 93.22 | 91.06 | 87.18 | 84.48 | 86.18 | 13.1k |
| | UDify | **98.89** | 98.34 | 98.14 | **94.25** | **92.33** | **89.27** | **86.21** | **86.61** | 13.1k |
| Chinese CFL | UDify | **83.75** | **82.72** | **98.75** | **62.46** | **42.48** | **43.46** | **21.07** | **42.22** | 0 |
| Chinese GSD | UDPipe | 94.88 | 99.22 | **99.99** | 84.64 | 80.50 | 76.79 | 71.04 | 76.78 | 4.0k |
| | UDify | **95.35** | **99.35** | 99.97 | **87.93** | **83.75** | **80.33** | **74.36** | **80.28** | 4.0k |
| Chinese HK | UDify | **82.86** | **86.47** | **100.00** | **65.53** | **49.32** | **47.84** | **22.85** | **47.84** | 0 |
| Chinese PUD | UDify | **92.68** | **98.40** | **100.00** | **79.08** | **56.51** | **55.22** | **40.92** | **55.22** | 0 |
| Coptic Scriptorium | UDPipe | **94.70** | **96.35** | **95.49** | **85.58** | **80.97** | **72.24** | **64.45** | **68.48** | 371 |
| | UDify | 27.17 | 52.85 | 55.71 | 27.58 | 10.82 | 6.50 | 0.19 | 1.44 | 371 |
| Croatian SET | UDPipe | **98.13** | **92.25** | **97.27** | 91.10 | 86.78 | 84.11 | **73.61** | 81.19 | 7.0k |
| | UDify | 98.02 | 89.67 | 95.34 | **94.08** | **89.79** | **87.70** | 72.72 | **82.00** | 7.0k |
| Czech CAC | UDPipe | **99.37** | **96.34** | **98.57** | 92.99 | 90.71 | 88.84 | 84.30 | 87.18 | 23.5k |
| | UDify | 99.14 | 95.42 | 98.32 | **94.33** | **92.41** | **91.03** | **84.68** | **89.21** | 23.5k |
| Czech CLTT | UDPipe | 98.88 | 91.59 | 98.25 | 86.90 | 84.03 | 80.55 | 71.63 | 79.20 | 861 |
| | UDify | **99.17** | **93.66** | **98.86** | **91.69** | **89.96** | **87.59** | **79.50** | **86.79** | 861 |
| Czech FicTree | UDPipe | **98.55** | **95.87** | **98.63** | 92.91 | 89.75 | 86.97 | **81.04** | 85.49 | 10.2k |
| | UDify | 98.34 | 91.82 | 98.13 | **95.19** | **92.77** | **90.99** | 77.77 | **88.39** | 10.2k |
| Czech PDT | UDPipe | **99.18** | **97.23** | **99.02** | 93.33 | 91.31 | 89.64 | 86.15 | 88.60 | 68.5k |
| | UDify | **99.18** | 96.69 | 98.52 | **94.73** | **92.88** | **91.64** | **87.13** | **89.95** | 68.5k |
| Czech PUD | UDify | **97.93** | **93.98** | **96.94** | **92.59** | **87.95** | **84.85** | **77.39** | **82.81** | 0 |
| Danish DDT | UDPipe | **97.78** | **97.33** | **97.52** | 86.88 | 84.31 | 81.20 | **76.29** | **78.51** | 4.4k |
| | UDify | 97.50 | 95.41 | 94.60 | **87.76** | **84.50** | **81.60** | 73.76 | 75.15 | 4.4k |
| Dutch Alpino | UDPipe | 96.83 | 96.33 | **97.09** | 91.37 | 88.38 | 83.51 | 77.28 | 79.82 | 12.3k |
| | UDify | **97.67** | **97.66** | 95.44 | **94.23** | **91.21** | **87.32** | **82.81** | **80.76** | 12.3k |
| Dutch LassySmall | UDPipe | 96.50 | 96.42 | **97.41** | 90.20 | 86.39 | 81.88 | 77.19 | 78.83 | 5.8k |
| | UDify | **96.70** | **96.57** | 95.10 | **94.34** | **91.22** | **88.03** | **82.06** | **81.40** | 5.8k |

Table 7: The full test results of UDify on 124 treebanks (part 1 of 4). The SIZE column indicates the number of training sentences.

| Treebank | Model | UPOS | UFeats | Lemmas | UAS | LAS | CLAS | MLAS | BLEX | Size |
|---|---|---|---|---|---|---|---|---|---|---|
| English EWT | UDPipe | **96.29** | **97.10** | **98.25** | 89.63 | 86.97 | 84.02 | 79.00 | 82.36 | 12.5k |
| | UDify | 96.21 | 96.02 | 97.28 | **90.96** | **88.50** | **86.25** | **79.80** | **83.39** | 12.5k |
| English GUM | UDPipe | **96.02** | **96.82** | **96.85** | 87.27 | 84.12 | 78.55 | **73.51** | **74.68** | 2.9k |
| | UDify | 95.44 | 94.12 | 93.15 | **89.14** | **85.73** | **83.03** | 72.55 | 74.30 | 2.9k |
| English LinES | UDPipe | **96.91** | **96.31** | **96.45** | 84.15 | 79.71 | 77.44 | **71.38** | 73.22 | 2.7k |
| | UDify | 95.31 | 91.34 | 94.50 | **87.33** | **83.71** | **82.95** | 68.62 | **76.23** | 2.7k |
| English PUD | UDify | **96.18** | **93.50** | **94.20** | **91.52** | **88.66** | **87.83** | **75.61** | **80.57** | 0 |
| English ParTUT | UDPipe | 96.10 | **95.51** | **97.74** | 90.29 | 87.27 | 82.58 | **76.44** | 80.33 | 1.8k |
| | UDify | **96.16** | 92.61 | 96.45 | **92.84** | **90.14** | **86.28** | 74.59 | **82.01** | 1.8k |
| Erzya JR | UDify | **46.66** | **31.82** | **45.73** | **31.90** | **16.38** | **10.83** | **0.58** | **2.83** | 0 |
| Estonian EDT | UDPipe | **97.64** | **96.23** | **95.30** | 88.00 | 85.18 | 83.62 | 78.72 | **78.51** | 24.4k |
| | UDify | 97.44 | 95.13 | 86.56 | **89.53** | **86.67** | **85.17** | **79.20** | 69.31 | 24.4k |
| Faroese OFT | UDify | **77.46** | **35.20** | **51.09** | **67.24** | **59.26** | **51.17** | **2.39** | **21.92** | 0 |
| Finnish FTB | UDPipe | **96.65** | **96.62** | **95.49** | **90.68** | **87.89** | **85.11** | **80.58** | **81.18** | 15.0k |
| | UDify | 93.80 | 90.38 | 88.80 | 86.37 | 81.40 | 81.01 | 68.16 | 70.15 | 15.0k |
| Finnish PUD | UDify | **96.48** | **93.84** | **84.64** | **89.76** | **86.58** | **86.64** | **77.83** | **69.12** | 0 |
| Finnish TDT | UDPipe | **97.45** | **95.43** | **91.45** | **89.88** | **87.46** | **85.87** | **80.43** | **76.64** | 12.2k |
| | UDify | 94.43 | 90.48 | 82.89 | 86.42 | 82.03 | 82.62 | 70.89 | 63.66 | 12.2k |
| French GSD | UDPipe | 97.63 | **97.13** | **98.35** | 90.65 | 88.06 | 84.35 | 79.76 | 82.39 | 14.5k |
| | UDify | **97.83** | 96.17 | 97.34 | **93.60** | **91.45** | **88.54** | **81.61** | **84.51** | 14.5k |
| French PUD | UDify | **91.67** | **59.65** | **100.00** | **88.36** | **82.76** | **81.74** | **25.24** | **81.74** | 0 |
| French ParTUT | UDPipe | **96.93** | **94.43** | **95.70** | **92.17** | **89.63** | **84.62** | **75.22** | **78.07** | 804 |
| | UDify | 96.12 | 88.36 | 93.97 | 90.55 | 88.06 | 83.19 | 63.03 | 74.03 | 804 |
| French Sequoia | UDPipe | **98.79** | **98.09** | **98.57** | 92.37 | **90.73** | **87.55** | **84.51** | **85.93** | 2.2k |
| | UDify | 97.89 | 88.97 | 97.15 | **92.53** | 90.05 | 86.67 | 67.98 | 82.52 | 2.2k |
| French Spoken | UDPipe | 95.91 | 100.00 | **96.92** | 82.90 | 77.53 | 71.82 | 68.24 | 69.47 | 1.2k |
| | UDify | **96.23** | **98.67** | 96.59 | **85.24** | **80.01** | **75.40** | **69.74** | **72.77** | 1.2k |
| Galician CTG | UDPipe | **97.84** | **99.83** | **98.58** | **86.44** | **83.82** | **78.58** | **72.46** | **77.21** | 2.3k |
| | UDify | 96.51 | 97.10 | 97.08 | 84.75 | 80.89 | 74.62 | 65.86 | 72.17 | 2.3k |
| Galician TreeGal | UDPipe | **95.82** | **93.96** | **97.06** | 82.72 | **77.69** | 71.69 | **63.73** | **68.89** | 601 |
| | UDify | 94.59 | 80.67 | 94.93 | **84.08** | 76.77 | **73.06** | 49.76 | 66.99 | 601 |
| German GSD | UDPipe | 94.48 | **90.68** | **96.80** | 85.53 | 81.07 | 76.26 | 58.82 | 72.13 | 13.8k |
| | UDify | **94.55** | 90.43 | 94.42 | **87.81** | **83.59** | **80.03** | **61.27** | **72.48** | 13.8k |
| German PUD | UDify | **89.49** | **30.66** | **94.77** | **89.86** | **84.46** | **80.50** | **2.10** | **72.95** | 0 |
| Gothic PROIEL | UDPipe | **96.61** | **90.73** | **94.75** | 85.28 | **79.60** | **76.92** | **66.70** | **72.93** | 3.4k |
| | UDify | 95.55 | 85.97 | 80.57 | **85.61** | 79.37 | 76.26 | 63.09 | 58.65 | 3.4k |
| Greek GDT | UDPipe | **97.98** | **94.96** | **95.82** | 92.10 | 89.79 | 85.71 | **78.60** | **79.72** | 1.7k |
| | UDify | 97.72 | 93.29 | 89.43 | **94.33** | **92.15** | **88.67** | 77.89 | 71.83 | 1.7k |
| Hebrew HTB | UDPipe | **97.02** | **95.87** | **97.12** | 89.70 | 86.86 | 81.45 | **75.52** | **78.14** | 5.2k |
| | UDify | 96.94 | 93.41 | 94.15 | **91.63** | **88.11** | **83.04** | 72.55 | 74.87 | 5.2k |
| Hindi HDTB | UDPipe | **97.52** | **94.15** | **98.67** | 94.85 | **91.83** | **88.21** | **78.49** | **86.83** | 13.3k |
| | UDify | 97.12 | 92.59 | 98.23 | **95.13** | 91.46 | 87.80 | 75.54 | 86.10 | 13.3k |
| Hindi PUD | UDify | **87.54** | **22.81** | **100.00** | **71.64** | **58.42** | **53.03** | **3.32** | **53.03** | 0 |
| Hungarian Szeged | UDPipe | 95.76 | **91.75** | **95.05** | 84.04 | 79.73 | 78.65 | **67.63** | **73.63** | 911 |
| | UDify | **96.36** | 86.16 | 90.19 | **89.68** | **84.88** | **83.93** | 64.27 | 72.21 | 911 |
| Indonesian GSD | UDPipe | **93.69** | **95.58** | **99.64** | 85.31 | 78.99 | 76.76 | **67.74** | **76.38** | 4.5k |
| | UDify | 93.36 | 93.32 | 98.37 | **86.45** | **80.10** | **78.05** | 66.93 | 76.31 | 4.5k |
| Indonesian PUD | UDify | **76.10** | **44.23** | **100.00** | **77.47** | **56.90** | **54.88** | **7.41** | **54.88** | 0 |
| Irish IDT | UDPipe | **92.72** | **82.43** | **90.48** | **80.39** | **72.34** | **63.48** | **46.49** | **55.32** | 567 |
| | UDify | 90.49 | 71.84 | 81.27 | 80.05 | 69.28 | 60.02 | 34.39 | 43.07 | 567 |
| Italian ISDT | UDPipe | 98.39 | **98.11** | **98.66** | 93.49 | 91.54 | 87.34 | 84.28 | 85.49 | 13.1k |
| | UDify | **98.51** | 98.01 | 97.72 | **95.54** | **93.69** | **90.40** | **86.54** | **86.70** | 13.1k |
| Italian PUD | UDify | **94.73** | **58.16** | **96.08** | **94.18** | **91.76** | **90.05** | **25.55** | **83.74** | 0 |
| Italian ParTUT | UDPipe | **98.38** | 97.77 | **98.16** | 92.64 | 90.47 | 85.05 | 81.87 | 82.99 | 1.8k |
| | UDify | 98.21 | **98.38** | 97.55 | **95.96** | **93.68** | **89.83** | **86.83** | **86.44** | 1.8k |

Table 8: The full test results of UDify on 124 treebanks (part 2 of 4).

| Treebank | Model | UPOS | UFeats | Lemmas | UAS | LAS | CLAS | MLAS | BLEX | Size |
|---|---|---|---|---|---|---|---|---|---|---|
| Japanese GSD | UDPipe | **98.13** | **99.98** | **99.52** | **95.06** | **93.73** | **88.35** | **86.37** | **88.04** | 7.1k |
| | UDify | 97.08 | 99.97 | 98.80 | 94.37 | 92.08 | 86.19 | 82.99 | 85.12 | 7.1k |
| Japanese Modern | UDify | **74.94** | **96.14** | **79.70** | **74.99** | **55.62** | **42.67** | **30.89** | **35.47** | 0 |
| Japanese PUD | UDify | **97.89** | **99.98** | **99.31** | **94.89** | **93.62** | **87.92** | **84.86** | **87.15** | 0 |
| Kazakh KTB | UDPipe | 55.84 | 40.40 | 63.96 | 53.30 | 33.38 | 27.06 | **4.82** | 15.10 | 32 |
| | UDify | **85.59** | **65.49** | **77.18** | **74.77** | **63.66** | **61.84** | 34.23 | **45.51** | 32 |
| Komi Zyrian IKDP | UDify | **59.92** | **39.32** | **57.56** | **36.01** | **22.12** | **17.45** | **1.54** | **6.80** | 0 |
| Komi Zyrian Lattice | UDify | **38.57** | **29.45** | **55.33** | **28.85** | **12.99** | **10.79** | **0.72** | **3.28** | 0 |
| Korean GSD | UDPipe | **96.29** | **99.77** | **93.40** | **87.70** | **84.24** | **82.05** | **79.74** | **76.35** | 4.4k |
| | UDify | 90.56 | 99.63 | 82.84 | 82.74 | 74.26 | 71.72 | 65.94 | 57.58 | 4.4k |
| Korean Kaist | UDPipe | **95.59** | 100.00 | **94.30** | **88.42** | **86.48** | **84.12** | **80.72** | **79.22** | 23.0k |
| | UDify | 94.67 | **99.98** | 85.89 | 87.57 | 84.52 | 82.05 | 78.27 | 68.99 | 23.0k |
| Korean PUD | UDify | **64.43** | **60.47** | **70.47** | **63.57** | **46.89** | **45.29** | **16.26** | **30.94** | 0 |
| Kurmanji MG | UDPipe | 53.36 | **41.54** | **69.58** | **45.23** | **34.32** | **29.41** | **2.74** | 19.39 | 21 |
| | UDify | **60.23** | 37.78 | 58.08 | 35.86 | 20.40 | 14.75 | 1.42 | **7.28** | 21 |
| Latin ITTB | UDPipe | 98.34 | **96.97** | **98.99** | 91.06 | 88.80 | 86.40 | **82.35** | 85.71 | 16.8k |
| | UDify | **98.48** | 95.81 | 98.08 | **92.43** | **90.12** | **87.93** | 82.24 | **85.97** | 16.8k |
| Latin PROIEL | UDPipe | **97.01** | **91.53** | **96.32** | 83.34 | 78.66 | 76.20 | **67.40** | **73.65** | 15.9k |
| | UDify | 96.79 | 89.49 | 91.79 | **84.85** | **80.52** | **77.96** | 67.18 | 71.00 | 15.9k |
| Latin Perseus | UDPipe | 88.40 | 79.10 | **81.45** | 71.20 | 61.28 | 56.32 | 41.58 | 45.09 | 1.3k |
| | UDify | **90.96** | **82.09** | 81.08 | **78.33** | **69.60** | **65.95** | **50.26** | **51.33** | 1.3k |
| Latvian LVTB | UDPipe | **96.11** | **93.01** | **95.46** | 87.20 | 83.35 | 80.90 | **71.92** | **76.64** | 7.2k |
| | UDify | 96.02 | 89.78 | 91.00 | **89.33** | **85.09** | **82.34** | 69.51 | 72.58 | 7.2k |
| Lithuanian HSE | UDPipe | 81.70 | 60.47 | **76.89** | 51.98 | 42.17 | 38.93 | 18.17 | 28.70 | 154 |
| | UDify | **90.47** | **70.00** | 67.17 | **79.06** | **69.34** | **66.00** | **36.21** | **36.35** | 154 |
| Maltese MUDT | UDPipe | **95.99** | 100.00 | **100.00** | **84.65** | **79.71** | **71.49** | **66.75** | **71.49** | 1.1k |
| | UDify | 91.98 | **99.89** | **100.00** | 83.07 | 75.56 | 65.08 | 58.14 | 65.08 | 1.1k |
| Marathi UFAL | UDPipe | 80.10 | **67.23** | **81.31** | 70.63 | 61.41 | 57.44 | **29.34** | **45.87** | 374 |
| | UDify | **88.59** | 59.22 | 72.82 | **79.37** | **67.72** | **60.13** | 21.71 | 39.25 | 374 |
| Naija NSC | UDify | **55.44** | **51.32** | **97.03** | **45.75** | **32.16** | **31.62** | **4.73** | **29.33** | 0 |
| North Sami Giella | UDPipe | **92.54** | **90.03** | **88.31** | **78.30** | **73.49** | **70.94** | **62.40** | **61.45** | 2.3k |
| | UDify | 90.21 | 83.55 | 71.50 | 74.30 | 67.13 | 64.41 | 51.20 | 40.63 | 2.3k |
| Norwegian Bokmaal | UDPipe | **98.31** | **97.14** | **98.64** | 92.39 | 90.49 | 88.18 | 84.06 | 86.53 | 15.7k |
| | UDify | 98.18 | 96.36 | 97.33 | **93.97** | **92.18** | **90.40** | **85.02** | **87.13** | 15.7k |
| Norwegian Nynorsk | UDPipe | **98.14** | **97.02** | **98.18** | 92.09 | 90.01 | 87.68 | 82.97 | 85.47 | 14.2k |
| | UDify | **98.14** | 96.55 | 97.18 | **94.34** | **92.37** | **90.39** | **85.01** | **86.71** | 14.2k |
| Norwegian NynorskLIA | UDPipe | 89.59 | 86.13 | 93.93 | 68.08 | 60.07 | 54.89 | 44.47 | 50.98 | 340 |
| | UDify | **95.01** | **93.36** | **96.13** | **75.40** | **69.60** | **65.33** | **56.90** | **62.27** | 340 |
| Old Church Slavonic PROIEL | UDPipe | **96.91** | **90.66** | **93.11** | **89.66** | **85.04** | **83.41** | **73.63** | **77.81** | 4.1k |
| | UDify | 84.23 | 71.30 | 65.70 | 76.71 | 66.67 | 64.10 | 46.25 | 43.88 | 4.1k |
| Old French SRCMF | UDPipe | **96.09** | **97.81** | **100.00** | **91.74** | **86.83** | **83.85** | **79.91** | **83.85** | 13.9k |
| | UDify | 95.73 | 96.98 | **100.00** | **91.74** | 86.65 | 83.49 | 78.85 | 83.49 | 13.9k |
| Persian Seraji | UDPipe | **97.75** | **97.78** | **97.44** | **90.05** | **86.66** | **83.26** | **81.23** | **80.93** | 4.8k |
| | UDify | 96.22 | 94.73 | 92.55 | 89.59 | 85.84 | 81.98 | 76.65 | 74.74 | 4.8k |
| Polish LFG | UDPipe | **98.80** | **95.49** | **97.54** | 96.58 | **94.76** | 93.01 | **87.04** | **90.26** | 13.8k |
| | UDify | **98.80** | 87.71 | 94.04 | **96.67** | 94.58 | **93.03** | 76.50 | 85.15 | 13.8k |
| Polish SZ | UDPipe | 98.34 | **93.04** | **97.16** | 93.39 | **91.24** | **89.39** | **81.06** | **85.99** | 6.1k |
| | UDify | **98.36** | 67.11 | 93.92 | **93.67** | 89.20 | 87.31 | 48.47 | 80.24 | 6.1k |
| Portuguese Bosque | UDPipe | 97.07 | **96.40** | **98.46** | 91.36 | **89.04** | **85.19** | **76.67** | **83.06** | 8.3k |
| | UDify | **97.10** | 89.70 | 91.60 | **91.37** | 87.84 | 84.13 | 69.09 | 78.64 | 8.3k |
| Portuguese GSD | UDPipe | **98.31** | **99.92** | **99.30** | 93.01 | 91.63 | 87.67 | **85.96** | 86.94 | 9.7k |
| | UDify | 98.04 | 95.75 | 98.95 | **94.22** | **92.54** | **89.37** | 82.32 | **87.90** | 9.7k |
| Portuguese PUD | UDify | **90.14** | **51.16** | **99.79** | **87.02** | **80.17** | **74.10** | **17.51** | **74.10** | 0 |
| Romanian Nonstandard | UDPipe | 96.68 | **90.88** | **94.78** | 89.12 | 84.20 | 78.91 | **65.93** | **73.44** | 8.0k |
| | UDify | **96.83** | 88.89 | 89.33 | **90.36** | **85.26** | **80.41** | 64.68 | 68.11 | 8.0k |

Table 9: The full test results of UDify on 124 treebanks (part 3 of 4).

| Treebank | Model | UPOS | UFeats | Lemmas | UAS | LAS | CLAS | MLAS | BLEX | Size |
|---|---|---|---|---|---|---|---|---|---|---|
| Romanian RRT | UDPipe | **97.96** | **97.53** | **98.41** | 91.31 | 86.74 | 82.57 | 79.02 | **81.09** | 8.0k |
| | UDify | 97.73 | 96.12 | 95.84 | **93.16** | **88.56** | **84.87** | **79.20** | 79.92 | 8.0k |
| Russian GSD | UDPipe | **97.10** | **92.66** | **97.37** | 88.15 | 84.37 | 82.66 | **74.07** | **80.03** | 3.9k |
| | UDify | 96.91 | 87.45 | 77.73 | **90.71** | **86.03** | **84.51** | 67.24 | 62.08 | 3.9k |
| Russian PUD | UDify | **93.06** | **63.60** | **77.93** | **93.51** | **87.14** | **83.96** | **37.25** | **61.86** | 0 |
| Russian SynTagRus | UDPipe | **99.12** | **97.57** | **98.53** | 93.80 | 92.32 | 90.85 | **87.91** | **89.17** | 48.8k |
| | UDify | 98.97 | 96.29 | 94.47 | **94.83** | **93.13** | **91.87** | 86.91 | 85.44 | 48.8k |
| Russian Taiga | UDPipe | 93.18 | 82.87 | 89.99 | 75.45 | 69.11 | 65.31 | 48.81 | 57.21 | 881 |
| | UDify | **95.39** | **88.47** | **90.19** | **84.02** | **77.80** | **75.12** | **59.71** | **65.15** | 881 |
| Sanskrit UFAL | UDify | **37.33** | **17.63** | **37.38** | **40.21** | **18.56** | **15.38** | **0.85** | **4.12** | 0 |
| Serbian SET | UDPipe | **98.33** | **94.35** | **97.36** | 92.70 | 89.27 | 87.08 | **79.14** | 84.18 | 2.9k |
| | UDify | 98.30 | 92.22 | 95.86 | **95.68** | **91.95** | **90.30** | 78.45 | **84.93** | 2.9k |
| Slovak SNK | UDPipe | 96.83 | **90.82** | **96.40** | 89.82 | 86.90 | 84.81 | 74.00 | 81.37 | 8.5k |
| | UDify | **97.46** | 89.30 | 93.80 | **95.92** | **93.87** | **92.86** | **77.33** | **85.12** | 8.5k |
| Slovenian SSJ | UDPipe | 98.61 | **95.92** | **98.25** | 92.96 | 91.16 | 88.76 | **83.85** | **86.89** | 6.5k |
| | UDify | **98.73** | 93.44 | 96.50 | **94.74** | **93.07** | **90.94** | 81.55 | 86.38 | 6.5k |
| Slovenian SST | UDPipe | 93.79 | 86.28 | **95.17** | 73.51 | 67.51 | 63.46 | 52.67 | 60.32 | 2.1k |
| | UDify | **95.40** | **89.81** | 95.15 | **80.37** | **75.03** | **71.19** | **61.32** | **67.24** | 2.1k |
| Spanish AnCora | UDPipe | **98.91** | **98.49** | **99.17** | 92.34 | 90.26 | 86.39 | **83.97** | **85.51** | 14.3k |
| | UDify | 98.53 | 97.89 | 98.07 | **92.99** | **90.50** | **87.26** | 83.43 | 84.85 | 14.3k |
| Spanish GSD | UDPipe | **96.85** | **97.09** | **98.97** | 90.71 | **88.03** | **82.85** | **75.98** | **81.47** | 14.2k |
| | UDify | 95.91 | 95.08 | 96.52 | **90.82** | 87.23 | 82.83 | 72.47 | 78.08 | 14.2k |
| Spanish PUD | UDify | **88.98** | **54.58** | **100.00** | **90.45** | **83.08** | **77.42** | **18.06** | **77.42** | 0 |
| Swedish LinES | UDPipe | 96.78 | **89.43** | **97.03** | 86.07 | 81.86 | 80.32 | 66.48 | 77.38 | 2.7k |
| | UDify | **96.85** | 87.24 | 92.70 | **88.77** | **85.49** | **85.61** | **66.99** | **77.62** | 2.7k |
| Swedish PUD | UDify | **96.36** | **80.04** | **88.81** | **89.17** | **86.10** | **85.25** | **57.12** | **72.92** | 0 |
| Swedish Sign Language SSLC | UDPipe | **68.09** | 100.00 | **100.00** | **50.35** | **37.94** | **39.51** | **30.96** | **39.51** | 88 |
| | UDify | 63.48 | **96.10** | **100.00** | 40.43 | 26.95 | 30.12 | 23.29 | 30.12 | 88 |
| Swedish Talbanken | UDPipe | 97.94 | **96.86** | **98.01** | 89.63 | 86.61 | 84.45 | 79.67 | **82.26** | 4.3k |
| | UDify | **98.11** | 95.92 | 95.50 | **91.91** | **89.03** | **87.26** | **80.72** | 81.31 | 4.3k |
| Tagalog TRG | UDify | **60.62** | **35.62** | **73.63** | **64.04** | **40.07** | **36.84** | **0.00** | **13.16** | 0 |
| Tamil TTB | UDPipe | 91.05 | **87.28** | **93.92** | 74.11 | 66.37 | 63.71 | **55.31** | **59.58** | 401 |
| | UDify | **91.50** | 83.21 | 80.84 | **79.34** | **71.29** | **69.10** | 53.62 | 54.84 | 401 |
| Telugu MTG | UDPipe | 93.07 | 99.03 | **100.00** | 91.26 | **85.02** | **81.76** | **77.75** | **81.76** | 1.1k |
| | UDify | **93.48** | **99.31** | **100.00** | **92.23** | 83.91 | 79.92 | 76.10 | 79.92 | 1.1k |
| Thai PUD | UDify | **56.78** | **62.48** | **100.00** | **49.05** | **26.06** | **18.42** | **3.77** | **18.42** | 0 |
| Turkish IMST | UDPipe | **96.01** | **92.55** | **96.01** | 74.19 | **67.56** | 63.83 | **56.96** | **61.37** | 3.7k |
| | UDify | 94.29 | 84.49 | 87.71 | **74.56** | 67.44 | **63.87** | 49.42 | 54.10 | 3.7k |
| Turkish PUD | UDify | **77.34** | **24.59** | **84.31** | **67.68** | **46.07** | **39.95** | **2.61** | **32.50** | 0 |
| Ukrainian IU | UDPipe | 97.59 | **92.66** | **97.23** | 88.29 | 85.25 | 81.90 | **73.81** | 79.10 | 5.3k |
| | UDify | **97.71** | 88.63 | 94.00 | **92.83** | **90.30** | **88.15** | 72.93 | **81.04** | 5.3k |
| Upper Sorbian UFAL | UDPipe | 62.93 | 41.10 | 68.68 | 45.58 | 34.54 | 27.18 | **3.37** | 16.65 | 24 |
| | UDify | **84.87** | **48.84** | **72.68** | **71.55** | **62.82** | **56.04** | 16.19 | **37.89** | 24 |
| Urdu UDTB | UDPipe | 93.66 | 81.92 | **97.40** | 87.50 | 81.62 | 75.20 | 55.02 | 73.07 | 4.0k |
| | UDify | **94.37** | **82.80** | 96.68 | **88.43** | **82.84** | **77.00** | **56.70** | **73.97** | 4.0k |
| Uyghur UDT | UDPipe | **89.87** | **88.30** | **95.31** | **78.46** | **67.09** | **60.85** | **47.84** | **57.08** | 1.7k |
| | UDify | 75.88 | 70.80 | 79.70 | 65.89 | 48.80 | 38.95 | 21.75 | 31.31 | 1.7k |
| Vietnamese VTB | UDPipe | 89.68 | **99.72** | **99.55** | 70.38 | 62.56 | 60.03 | 55.56 | 59.54 | 1.4k |
| | UDify | **91.29** | 99.58 | 99.21 | **74.11** | **66.00** | **63.34** | **58.71** | **62.61** | 1.4k |
| Warlpiri UFAL | UDify | **33.44** | **18.15** | **39.17** | **21.66** | **7.96** | **7.49** | **0.00** | **0.88** | 0 |
| Yoruba YTB | UDify | **50.86** | **78.32** | **85.56** | **37.62** | **19.09** | **16.56** | **6.30** | **12.15** | 0 |

Table 10: The full test results of UDify on 124 treebanks (part 4 of 4).

# Interactive Language Learning by Question Answering

**Xingdi Yuan**[♡*] **Marc-Alexandre Côté**[♡*] **Jie Fu**[♣♠] **Zhouhan Lin**[♢♠]
**Christopher Pal**[♣♠] **Yoshua Bengio**[♢♠] **Adam Trischler**[♡]
[♡]Microsoft Research, Montréal [♣]Polytechnique Montréal
[♢]Université de Montréal [♠] Mila
eric.yuan@microsoft.com macote@microsoft.com

## Abstract

Humans observe and interact with the world to acquire knowledge. However, most existing machine reading comprehension (MRC) tasks miss the interactive, information-seeking component of comprehension. Such tasks present models with static documents that contain all necessary information, usually concentrated in a single short substring. Thus, models can achieve strong performance through simple word- and phrase-based pattern matching. We address this problem by formulating a novel text-based question answering task: Question Answering with Interactive Text (QAit)[1]. In QAit, an agent must interact with a partially observable text-based environment to gather information required to answer questions. QAit poses questions about the existence, location, and attributes of objects found in the environment. The data is built using a text-based game generator that defines the underlying dynamics of interaction with the environment. We propose and evaluate a set of baseline models for the QAit task that includes deep reinforcement learning agents. Experiments show that the task presents a major challenge for machine reading systems, while humans solve it with relative ease.

## 1 Introduction

The research community has defined the task of machine reading comprehension (MRC) to teach machines to read and understand text. In most MRC tasks, given a knowledge source (usually a text document) and a question on its content, a model is required to answer the question either by pointing to words in the source or by generating a text string. Recent years have seen a flourishing of MRC works, including the release of numerous datasets (Rajpurkar et al., 2016; Trischler et al., 2016; Nguyen et al., 2016; Reddy et al., 2018; Yang et al., 2018) and a host of neural models that address them (Wang and Jiang, 2016; Wang et al., 2017; Chen et al., 2017).

| | |
|---|---|
| Game | You find yourself in a backyard. You make out a patio table. You see a patio chair. It's stylish. In a corner you can see a BBQ. There's an open screen door leading south. |
| Quest | Is ghargh edible? |
| Agent | > **Qapla'** |
| Game | That's not a verb I recognize. |
| Agent | > **eat ghargh** |
| Game | You can't see any such thing. |
| Agent | > **go south** |
| Game | You've just entered a kitchen. You can see an open fridge here. It is empty! You see a closed oven. Look over there, it's a counter. On the counter you can see a diced potato, a red ghargh and a knife. A an open door is leading north. |
| Agent | > **eat ghargh** |
| Game | You need to take the red ghargh first. |
| Agent | > **take red ghargh** |
| Game | You take the red ghargh from the counter. |
| Agent | > **eat ghargh** |
| Game | You eat the red ghargh. Not bad. |
| Agent | > **wait** |
| Agent | > **answer: yes** |

Table 1: Example of a player interacting with an environment to gather knowledge and answer a question.

Most existing MRC datasets, as a consequence of their construction, encourage models to do fairly shallow phrase and word matching between the question and the knowledge source. There are three reasons for this. First, questions tend to paraphrase or permute the text that expresses the answer, making pattern matching an effective strategy. On the SQuAD development set, for example, the average precision, recall, and $F_1$ score between questions and the sentences containing answer spans are 0.409, 0.271, 0.301, respectively. Exploiting this overlap, Trischler et al.

[*] Equal contribution.

[1]The dataset and implementation of our baseline agents are publicly available at https://github.com/xingdi-eric-yuan/qait_public.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2796–2813, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

(2016) show that a simple Information Retrieval method can achieve high sentence-level accuracy on SQuAD.

Second, the information that supports predicting the answer from the source is often *fully observed*: the source is static, sufficient, and presented in its entirety. This does not match the information-seeking procedure that arises in answering many natural questions (Kwiatkowski et al., 2019), nor can it model the way humans observe and interact with the world to acquire knowledge.

Third, most existing MRC studies focus on *declarative* knowledge — the knowledge of facts or events that can be stated explicitly (i.e., declared) in short text snippets. Given a static description of an entity, declarative knowledge can often be extracted straightforwardly through pattern matching. For example, given the EMNLP website text, the conference deadline can be extracted by matching against a date mention. This focus overlooks another essential category of knowledge — *procedural* knowledge. Procedural knowledge entails executable sequences of actions. These might comprise the procedure for tying ones shoes, cooking a meal, or gathering new declarative knowledge. The latter will be our focus in this work. As an example, a more general way to determine EMNLP's deadline is to open a browser, head to the website, and then match against the deadline mention; this involves executing several mouse and keyboard interactions.

In order to teach MRC systems *procedures* for question answering, we propose a novel task: Question Answering with Interactive Text (QAit). Given a question $q \in Q$, rather than presenting a model with a static document $d \in D$ to read, QAit requires the model to interact with a partially observable environment $e \in E$ over a sequence of turns. The model must collect and aggregate evidence as it interacts, then produce an answer $a$ to $q$ based on its experience.

In our case, the environment $e$ is a text-based game with no explicit objective. The game places an agent in a simple modern house populated by various everyday objects. The agent may explore and manipulate the environment by issuing text commands. An example is shown in Table 1. We build a corpus of related text-based games using a generator from Côté et al. (2018), which enables us to draw games from a controlled distribution. This means there are random variations across the environment set $E$, in map layouts and in the existence, location, and names of objects, etc. Consequently, an agent cannot answer questions merely by memorizing games it has seen before. Because environments are partially observable (i.e., not all necessary information is available at a single turn), an agent must take a sequence of decisions – analogous to following a search and reasoning procedure – to gather the required information. The learning target in QAit is thus not the declarative knowledge $a$ itself, but the *procedure* for arriving at $a$ by collecting evidence.

The main contributions of this work are as follows:

1. We introduce a novel MRC dataset, QAit, which focuses on *procedural* knowledge. In it, an agent interacts with an environment to discover the answer to a given question.

2. We introduce to the MRC domain the practice of generating training data on the fly. We sample training examples from a distribution; hence, an agent is highly unlikely to encounter the same training example more than once. This helps to prevent overfitting and rote memorization.

3. We evaluate a collection of baseline agents on QAit, including state-of-the-art deep reinforcement learning agents and humans, and discuss limitations of existing approaches.

## 2 The QAit Dataset

### 2.1 Overview

We make the question answering problem interactive by building text-based games along with relevant question-answer pairs. We use TextWorld (Côté et al., 2018) to generate these games. Each interactive environment is composed of multiple locations with paths connecting them in a randomly drawn graph. Several interactable objects are scattered across the locations. A player sends text commands to interact with the world, while the game's interpreter only recognizes a small subset of all possible command strings (we call these the valid commands). The environment changes state in response to a valid command and returns a string of text feedback describing the change.

The underlying game dynamics arise from a set of objects (e.g., doors) that possess attributes (e.g.,

| | edible | drinkable | portable | openable | cuttable | sharp | heat_source | cookable | holder |
|---|---|---|---|---|---|---|---|---|---|
| Butter knife | | | 1 | | | 1 | | | |
| Oven | | | | 1 | | | 1 | | 1 |
| Raw chicken | | | 1 | | 1 | | | 1 | |
| Fried chicken | 1 | | 1 | | 1 | | | 1 | |

Table 2: Supported attributes along with examples.

doors are openable), and a set of rules (e.g., opening a closed door makes the connected room accessible). The supported attributes are shown in Table 2, while the rules can be inferred from the list of supported commands (see Appendix C). Note that player interactions might affect an object's attributes. For instance, cooking a piece of *raw chicken* on the stove with a frying pan makes it edible, transforming it into *fried chicken*.

In each game, the existence of objects, the location of objects, and their names are randomly sampled. Depending on the task, a name can be a made-up word. However, game dynamics are constant across all games – e.g., there will never be a drinkable heat source.

Text in QAit is generated by the TextWorld engine according to English templates, so it does not express the full variation of natural language. However, taking inspiration from the bAbI tasks (Weston et al., 2015), we posit that controlled simplifications of natural language are useful for isolating more complex reasoning behaviors.

### 2.2 Available Information

At every game step, the environment returns an *observation string* describing the information visible to the agent, as well as the *command feedback*, which is text describing the response to the previously issued command.

**Optional Information:** Since we have access to the underlying state representation of a generated game, various optional information can be made available. For instance, it is possible to access the subset of commands that are valid at the current game step. Other available meta-information includes all objects that exist in the game, plus their locations, attributes, and states.

During training, one is free to use any optional information to guide the agent's learning, e.g., to shape the rewards. However, at test time, only the *observation string* and the *command feedback* are available.

### 2.3 Question Types and Difficulty Levels

Using the game information described above, we can generate questions with known ground truth answers for any given game.

#### 2.3.1 Question Types

For this initial version of QAit we consider three straightforward question types.

**Location:** ("Where is the can of soda?") Given an object name, the agent must answer with the name of the container that most directly holds the object. This can be either a location, a holder within a location, or the player's inventory. For example, if the can of soda is in a fridge which is in the kitchen, the answer would be "fridge".

**Existence:** ("Is there a raw egg in the world?") Given the name of an object, the agent must learn to answer whether the object exists in the game environment $e$.

**Attribute:** ("Is *ghargh* edible?") Given an object name and an attribute, the agent must answer with the value of the given attribute for the given object. Note that all attributes in our dataset are binary-valued. To discourage an agent from simply memorizing attribute values given an object name (Anand et al., 2018) (e.g., apples are always edible so agents can answer without interaction), we replace object names with unique, randomly drawn made-up words for this question type.

#### 2.3.2 Difficulty Levels

To better analyze the limitations of learning algorithms and to facilitate curriculum learning approaches, we define two difficulty levels based on the environment layout.

**Fixed Map:** The map (location names and layout) is fixed across games. Random objects are distributed across the map in each game. Statistics for this game configuration are shown in Table 3.

**Random Map:** Both map layouts and objects are randomly sampled in each game.

### 2.4 Action Space

We describe the action space of QAit by splitting it into two subsets: information-gathering actions and question-answering actions.

| | Fixed Map | Random Map |
|---|---|---|
| # Locations, $N_r$ | 6 | $N_r \sim \text{Uniform}[2, 12]$ |
| # Entities, $N_e$ | $N_e \sim \text{Uniform}[3 \cdot N_r, 6 \cdot N_r]$ | |
| Actions / Game | 17 | 17 |
| Modifiers / Game | 18.5 | 17.7 |
| Objects / Game | 26.7 | 27.5 |
| # Obs. Tokens | 93.1 | 89.7 |

Table 3: Statistics of the QAit dataset. Numbers are averaged over 10,000 randomly sampled games.

**Information Gathering** The player generates text commands word by word to navigate through and interact with the environment. On encountering an object, the player must interact with it to discover its attributes. To succeed, an agent must map the feedback received from the environment, in text, to a useful state representation. This is a form of reading comprehension.

To make the QAit task more tractable, all text commands are triplets of the form {action, modifier, object} (e.g., `open wooden door`). When there is no ambiguity, the environment understands commands without modifiers (e.g., `eat apple` will result in eating the "red apple" provided it is the only apple in the player's inventory). We list all supported commands in Appendix C.

Each game provides a set of three lexicons that divide the full vocabulary into actions, modifiers, and objects. Statistics are shown in Table 3. A model can generate a command at each game step by, e.g., sampling from a probability distribution induced over each lexicon. This reduces the size of the action space compared to a sequential, free-form setting where a model can pick any vocabulary word at any generation step.

An agent decides when to stop interacting with the environment to answer the question by generating a special `wait` command [2]. However, the number of interaction steps is limited: we use 80 steps in all experiments. When an agent has exhausted its available steps, the game terminates and the agent is forced to answer the question.

**Question Answering** Currently, all QAit answers are one word. For existence and attribute questions, the answer is either `yes` or `no`; for location questions, the answer can be any word in an observation string.

[2]We call it "wait" because when playing multiple games in a batch, batched environment will terminate only when all agents have issued the terminating command. Before that, some agent will wait. This is analogous to paddings in natural language processing tasks.

### 2.5 Evaluation Settings and Metrics

We evaluate an agent's performance on QAit by its accuracy in answering questions. We propose three distinct settings for the evaluation.

**Solving Training Games:** We use QA accuracy during training, averaged over a window of training time, to evaluate an agent's training performance. We provide 5 training sets for this purpose with [1, 2, 10, 100, 500] games, respectively. Each game in these sets is associated with multiple questions.

**Unlimited Games:** We implement a setup where games are randomly generated on the fly during training, rather than selected from a finite set as above. The distribution we draw from is controlled by a few parameters: number of locations, number of objects, type of map, and a random seed. From the **fixed map** game distribution described in Table 3, more than $10^{40}$ different games can be drawn. This means that a game is unlikely to be seen more than once during training. We expect that only a model with strong generalization capabilities will perform well in this setting.

**Zero-shot Evaluation:** For each game setting and question type, we provide 500 held out games that are never seen during training, each with one question. These are used to benchmark generalization in models in a reproducible manner, no matter the training setting. This set is analogous to the test set used in traditional supervised learning tasks, and can be used in conjunction with any training setting.

## 3 Baseline Models

### 3.1 Random Baseline

Our simplest baseline does not interact with the environment to answer questions; it samples an answer word uniformly from the QA action space (`yes` and `no` for attribute and existence questions; all possible object names in the game for location questions).

### 3.2 Human Baseline

We conducted a study with 21 participants to explore how humans perform on QAit in terms of

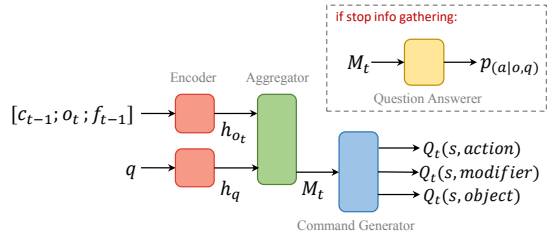


Figure 1: Overall architecture of our baseline agent.

QA accuracy. Participants played games they had not seen previously from a set generated by sampling 4 game-question pairs for each question type and difficulty level. The human results presented below always represent an average over 3 participants.

### 3.3 QA-DQN

We propose a neural baseline agent, QA-DQN, which takes inspiration from the work of Narasimhan et al. (2015) and Yu et al. (2018). The agent consists of three main components: an encoder, a command generator, and a question answerer. More precisely, at game step $t$, the encoder takes observation $o_t$ and question $q$ as input to generate hidden representations.[3] In the information gathering phase, the command generator generates Q-values for all action, modifier, and object words, with rankings of these Q-values used to generate text commands $c_t$. At any game step, the agent may decide to terminate information gathering and answer the question (or it is forced to do so if it has used up all of its moves). The question answerer uses the hidden representations at the final information-gathering step to generate a probability distribution over possible answers.

An overview of this architecture is shown in Figure 1 and full details are given in Appendix A.

#### 3.3.1 Reward Shaping

We design the following two rewards to help QA-DQN learn more efficiently; both used for training the command generator. Note that these rewards are part of the design of QA-DQN, but are not used to evaluate its performance. Question answering accuracy is the only evaluation metric for QAit tasks.

**Sufficient Information Bonus:** To tackle QAit tasks, an intelligent agent should know when to stop interacting – it should stop as soon as it has gathered enough information to answer the question correctly. For guiding the agent to learn this behavior, we give an additional reward when the agent stops with sufficient information. Specifically, assuming the agent decides to stop at game step $k$:

- Location: reward is 1 if the entity mentioned in the question is a sub-string of $o_k$, otherwise it is 0. This means whenever an agent observes the entity, it has sufficient information to infer the entity's location.
- Existence: when the correct answer is `yes`, a reward of 1 is assigned only if the entity is a sub-string of $o_k$. When the correct answer is `no`, a reward between 0 and 1 is given. The reward value corresponds to the exploration coverage of the environment, i.e., how many locations the agent has visited, and how many containers have been opened.
- Attribute: we heuristically define a set of conditions to verify each attribute, and reward the agent based on its fulfilment of these conditions. For instance, determining if an object `x` is **sharp** corresponds to checking the outcome of a cut command (`slice`, `chop`, or `dice`) while holding the object `x` and a cuttable food item. If the outcome is successful then the object `x` is sharp otherwise it is not. Alternatively, if trying to take the object `x` results in a failure, then we can deduces it is not sharp as all sharp objects are portable. The list of conditions for each attribute used in our experiments is shown in Appendix D.

**Episodic Discovery Bonus:** Following Yuan et al. (2018), we use an episodic counting reward to encourage the agent to discover unseen game states. The agent is assigned a positive reward whenever it encounters a new state (in text-based games, states are simply represented as strings):

$$r(o_t) = \begin{cases} 1.0 & \text{if } n(o_t) = 1, \\ 0.0 & \text{otherwise,} \end{cases}$$

where $n(\cdot)$ is reset to zero after each episode.

#### 3.3.2 Training Strategy

We apply different training strategies for the command generator and the question answerer.

**Command Generation:** Text-based games are sequential decision-making problems that can

[3] We concatenate $o_t$ with the command generated at previous game step and the text feedback returned by the game, as described in Section 2.2.

be described naturally by partially observable Markov decision processes (POMDPs) (Kaelbling et al., 1998). We use the Q-Learning (Watkins and Dayan, 1992) paradigm to train our agent. Specifically, following Mnih et al. (2015), our Q-value function is approximated with a deep neural network. Beyond vanilla DQN, we also apply several extensions, such as Rainbow (Hessel et al., 2017), to our training process. Details are provided in Section 4.

**Question Answering:** During training, we push all question answering transitions (observation strings when interaction stops, question strings, ground-truth answers) into a replay buffer. After every 20 game steps, we randomly sample a mini-batch of such transitions from the replay buffer and train the question answerer with supervised learning (e.g., using negative log-likelihood (NLL) loss).

## 4 Experimental Results

In this section, we report experimental results by difficulty levels. All random baseline performance values are averaged over 100 different runs. In the following subsections, we use "DQN", "DDQN" and "Rainbow" to indicate QA-DQN trained with vanilla DQN, Double DQN with prioritized experience replay, and Rainbow, respectively. Training curves shown in the following figures represent a sliding-window average with a window size of 500. Moreover, each curve is the average of 3 random seeds. For evaluation, we selected the model with the random seed yielding the highest training accuracy to compute its accuracy on the test games. Due to space limitations, we only report some key results here. See Appendix E for the full experimental results.

### 4.1 Fixed Map

Figure 2 shows the training curves for the neural baseline agents when trained using 10 games, 500 games and the "unlimited" games settings. Table 4 reports their zero-shot test performance.

From Figure 2, we observe that when training data size is small (e.g., 10 games), our baseline agent trained with all the three RL methods successfully master the training games. Vanilla DQN and DDQN are particularly strong at memorizing the training games. When training on more games (e.g., 500 games and unlimited games), in which case memorization is more difficult, Rainbow agents start to show its superiority — it has similar accuracy as the other two methods, and even outperforms them in existence question type.

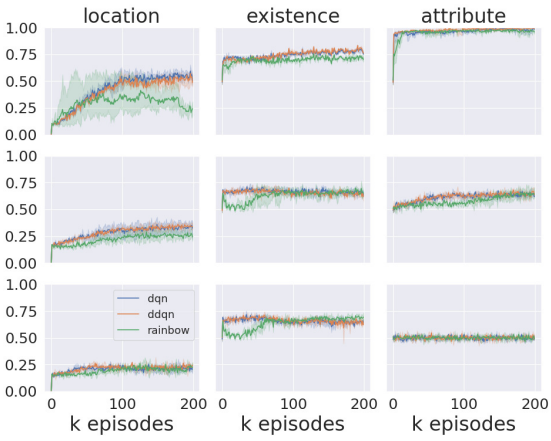


Figure 2: Training accuracy over episodes on **fixed map** setup. Upper row: 10 games; middle row: 500 games; lower row: unlimited games.

| | Fixed Map | | | Random Map | | |
|---|---|---|---|---|---|---|
| Model | Loc. | Exi. | Att. | Loc. | Exi. | Att. |
| Human | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 0.750 |
| Random | 0.027 | 0.497 | 0.496 | 0.034 | 0.500 | 0.499 |
| 10 games | | | | | | |
| DQN | 0.180 | 0.568 | 0.518 | 0.156 | 0.566 | 0.518 |
| DDQN | 0.188 | 0.566 | 0.516 | 0.142 | 0.606 | 0.500 |
| Rainbow | 0.156 | 0.590 | 0.520 | 0.144 | 0.586 | **0.530** |
| 500 games | | | | | | |
| DQN | 0.224 | 0.674 | **0.534** | 0.204 | 0.678 | **0.530** |
| DDQN | 0.218 | 0.626 | 0.508 | 0.222 | 0.656 | 0.486 |
| Rainbow | 0.190 | 0.656 | 0.496 | 0.172 | 0.678 | 0.494 |
| unlimited games | | | | | | |
| DQN | 0.216 | 0.662 | 0.514 | 0.188 | 0.668 | 0.506 |
| DDQN | 0.258 | 0.628 | 0.480 | 0.206 | **0.694** | 0.482 |
| Rainbow | **0.280** | **0.692** | 0.514 | **0.258** | 0.686 | 0.470 |

Table 4: Agent performance on zero-shot test games when trained on 10 games, 500 games and "unlimited" games settings. Note Att. and Exi. are binary questions with expected accuracy of 0.5.

From Table 4 we see similar observation, when trained on 10 games and 500 games, DQN and DDQN performs better on test games but on the unlimited games setting, rainbow agent performs as good as them, and sometimes even better. We can also observe that our agents fail to generalize on attribute questions. In unlimited games setting as shown in Figure 2, all three agents produce an accuracy of 0.5; in zero-shot test as shown in Table 4, no agent performs significantly better than random. This suggests the agents memorize game-question-answer triples when data size is small, and fail to do so in unlimited games setting. This

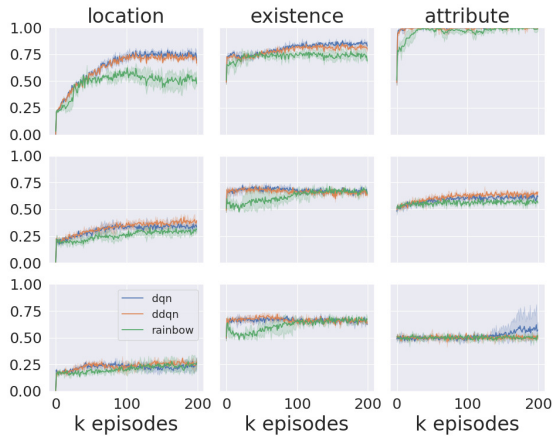


Figure 3: Training accuracy on the **random map** setup. Upper row: 10 games; middle row: 500 games; lower row: unlimited games.

| Model | Fixed Map | Random Map |
|---|---|---|
| Random | 14.7 | 16.5 |
| 10 games | | |
| DQN | 95.7 | 97.5 |
| DDQN | 90.4 | 92.2 |
| Rainbow | 91.8 | 84.7 |
| 500 games | | |
| DQN | 91.8 | 94.4 |
| DDQN | 95.6 | 90.2 |
| Rainbow | 96.9 | 96.6 |
| unlimited games | | |
| DQN | 100.0 | 100.0 |
| DDQN | 100.0 | 100.0 |
| Rainbow | 100.0 | 100.0 |

Table 5: Test performance given sufficient information.

can also be observed in Appendix E, where in attribute question experiments, the training accuracy is high, and sufficient information bonus is low (even close to 0).

### 4.2 Random Map

Figure 3 shows the training curves for the neural baseline agents when trained using 10 games, 500 games and "unlimited" games settings. The trends of our agents' performance on random map games are consistent with on fixed map games. However, because there exist easier games (as listed in Table 3, number of rooms is sampled between 2 and 12), agents show better training performance in such setting than **fixed map** setting in general.

Interestingly, we observe one of the DQN agent starts to learn in the unlimited games, attribute question setting. This may be because in games with smaller map size and less objects, there is a higher chance to accomplish some sub-tasks (e.g., it is easier to find an object when there are less rooms), and the agent learn such skills and apply them to similar tasks. Unfortunately, as shown in Table 4 that agent does not perform significantly better than random on test set. We expect with more training episodes, the agent can have a better generalization performance.

### 4.3 Question Answering Given Sufficient Information

The challenge in QAit is learning the interactive procedure for arriving at a state with the information needed to answer the question. We conduct the following experiments on **location** questions to investigate this challenge.

Based on the results in Table 4, we compute an agent's test accuracy *only if* it has obtained sufficient information – i.e., when the sufficient information bonus is 1. Results shown in Table 5 support our assumption that the QA module can learn (and generalize) effectively to answer given sufficient information. Similarly, experiments show that when objects being asked about are in the current observation, the random baseline's performance goes up significantly as well. We report our baseline agents' question answering accuracy and sufficient information bonuses on all experiment settings in Appendix E.

### 4.4 Full Information Setup

To reframe the QAit games as a standard MRC task, we also designed an experimental setting that eliminates the need to gather information interactively. From a heuristic trajectory through the game environment that is guaranteed to observe sufficient information for $q$, we concatenate all observations into a static "document" $d$ to build a $\{d, q, a\}$ triplet. A model then uses this fully observed document as input to answer the question. We split this data into training, validation, and test sets and follow the evaluation protocol for standard supervised MRC tasks. We take an off-the-shelf MRC model, Match-LSTM (Wang and Jiang, 2016), trained with negative log-likelihood loss as a baseline.

Unsurprisingly, Match-LSTM does fairly well on all 3 question types (86.4, 89.9 and 93.2 test accuracy on location, existence, and attribute questions, respectively). This implies that without the need to interact with the environment for information gathering, the task is simple enough that a word-matching model can answer questions with

high accuracy.

## 5 Related Work

### 5.1 MRC Datasets

Many large-scale machine reading comprehension and question answering datasets have been proposed recently. The datasets of Rajpurkar et al. (2016); Trischler et al. (2016) contain crowd-sourced questions based on single documents from Wikipedia and CNN news, respectively. Nguyen et al. (2016); Joshi et al. (2017); Dunn et al. (2017); Clark et al. (2018); Kwiatkowski et al. (2019) present question-answering corpora harvested from information retrieval systems, often containing multiple supporting documents for each question. This means a model must sift through a larger quantity of information and possibly reconcile competing viewpoints. Berant et al. (2013); Welbl et al. (2017); Talmor and Berant (2018) propose to leverage knowledge bases to generate question-answer pairs. Yang et al. (2018) focuses on questions that require multi-hop reasoning to answer, by building questions compositionally. Reddy et al. (2018); Choi et al. (2018) explore conversational question answering, in which a full understanding of the question depends on the conversation's history.

Most of these datasets focus on declarative knowledge and are static, with all information fully observable to a model. We contend that this setup, unlike QAit, encourages word matching. Supporting this contention, several studies highlight empirically that existing MRC tasks require little comprehension or reasoning. In Rychalska et al. (2018), it was shown that a question's main verb exerts almost no influence on the answer prediction: in over 90% of examined cases, swapping verbs for their antonyms does not change a system's decision. Jia and Liang (2017) show the accuracy of neural models drops from an average of 75% $F_1$ score to 36% $F_1$ when they manually insert adversarial sentences into SQuAD.

### 5.2 Interactive Environments

Several embodied or visual question answering datasets have been presented recently to address some of the problems of interest in our work, such as those of Brodeur et al. (2017); Das et al. (2017); Gordon et al. (2017). In contrast with these, our purely text-based environment circumvents challenges inherent to modelling interactions between separate data modalities. Furthermore, most visual question answering environments only support navigating and moving the camera as interactions. In text-based environments, however, it is relatively cheap to build worlds with complex interactions. This is because text enables us to model interactions abstractly without the need for, e.g., a costly physics engine.

Closely related to QAit is BabyAI (Chevalier-Boisvert et al., 2018). BabyAI is a gridworld environment that also features constrained language for generating simple home-based scenarios (i.e., instructions). However, observations and actions in BabyAI are not text-based. World of Bits (Shi et al., 2017) is a platform for training agents to interact with the internet to accomplish tasks like flight booking. Agents generally do not need to gather information in World of Bits, and the focus is on accomplishing tasks rather than answering questions.

### 5.3 Information Seeking

Information seeking behavior is an important capacity of intelligent systems that has been discussed for many years. Kuhlthau (2004) propose a holistic view of information search as a six-stage process. Schmidhuber (2010) discusses the connection between information seeking and formal notions of fun, creativity, and intrinsic motivation. Das et al. (2018) propose a model that continuously determines all entities' locations during reading and dynamically updates the associated representations in a knowledge graph. Bachman et al. (2016) propose a collection of tasks and neural methods for learning to gathering information efficiently in an environment.

To our knowledge, we are the first to consider interactive information-seeking tasks for question answering in worlds with complex dynamics. The QAit task was designed such that simple word matching methods do not apply, while more human-like information seeking models are encouraged.

## 6 Discussion and Future Work

**Monitoring Information Seeking:** In QAit, the only evaluation metric is question answering accuracy. However, the sufficient information bonus described in Section 3.3.1 is helpful for monitoring agents' ability to gather relevant information. We report its value for all experiments in

Appendix E. We observe that while the baseline agents can reach a training accuracy of 100% for answering attribute questions when trained on a few games, the sufficient information bonus is close to 0. This is a clear indication that the agent overfits to the question-answer mapping of the games rather than learning how to gather useful information. This aligns with our observation that the agent does not perform better than random on the unlimited games setting, because it fails to gather the needed information.

**Challenges in QAit:** QAit focuses on learning procedural knowledge from interactive environments, so it is natural to use deep RL methods to tackle it. Experiments suggest the dataset presents a major challenge for existing systems, including Rainbow, which set the state of the art on Atari games. As a simplified and controllable text-based environment, QAit can drive research in both the RL and language communities, especially where they intersect. Until recently, the RL community focused mainly on solving single environments (i.e., training and testing on the same game). Now, we see a shift towards solving multiple games and testing for generalization (Cobbe et al., 2018; Justesen et al., 2018). We believe QAit serves this purpose.

**Templated Language:** As QAit is based on TextWorld, it has the obvious limitation of using templated English. However, TextWorld provides approximately 500 *human-written* templates for describing rooms and objects, so some textual diversity exists, and since game narratives are generated compositionally, this diversity increases along with the complexity of a game. We believe simplified and controlled text environments offer a bridge to full natural language, on which we can isolate the learning of useful behaviors like information seeking and command generation. Nevertheless, it would be interesting to further diversify the language in QAit, for instance by having human writers paraphrase questions.

**Future Work:** Based on our present efforts to tackle QAit, we propose the following directions for future work.

A **structured memory** (e.g., a dynamic knowledge graph as proposed in Das et al. (2018); Ammanabrolu and Riedl (2019a)) could be helpful for explicitly memorizing the places and objects that an agent has observed. This is especially useful when an agent must revisit a location or object or should avoid doing so.

Likewise, a variety of **external knowledge** could be leveraged by agents. For instance, incorporating a pretrained language model could improve command generation by imparting knowledge of word and object affordances. In recent work, Hausknecht et al. (2019) show that pretrained modules together with handcrafted subpolicies help in solving text-based games, while Yin and May (2019) use BERT (Devlin et al., 2018) to inject 'weak common sense' into agents for text-based games. Ammanabrolu and Riedl (2019b) show that knowledge graphs and their associated neural encodings can be used as a medium for domain transfer across text-based games.

In finite game settings we observed significant overfitting, especially for attribute questions – as shown in Appendix E, our agent achieves high QA accuracy but low sufficient information bonus on the single-game setting. Sometimes attributes require long procedures to verify, and thus, we believe that denser rewards would help with this problem. One possible solution is to provide **intermediate rewards** whenever the agent achieves a sub-task.

## Acknowledgments

The authors thank Romain Laroche, Rémi Tachet des Combes, Matthew Hausknecht, Philip Bachman, and Layla El Asri for insightful ideas and discussions. We thank Tavian Barnes, Wendy Tay, and Emery Fine for their work on the TextWorld framework. We also thank the anonymous EMNLP reviewers for their helpful feedback and suggestions.

## References

Ammanabrolu, P. and Riedl, M. (2019a). Playing text-adventure games with graph-based deep reinforcement learning. In *NAACL*, pages 3557–3565, Minneapolis, Minnesota. Association for Computational Linguistics.

Ammanabrolu, P. and Riedl, M. (2019b). Transfer in deep reinforcement learning using knowledge graphs. *CoRR*, abs/1908.06556.

Anand, A., Belilovsky, E., Kastner, K., Larochelle, H., and Courville, A. C. (2018). Blindfold baselines for embodied QA. *CoRR*, abs/1811.05013.

Bachman, P., Sordoni, A., and Trischler, A. (2016). Towards information-seeking agents. *CoRR*, abs/1612.02605.

Berant, J., Chou, A., Frostig, R., and Liang, P. (2013). Semantic parsing on freebase from question-answer pairs. In *EMNLP*, pages 1533–1544. ACL.

Brodeur, S., Perez, E., Anand, A., Golemo, F., Celotti, L., Strub, F., Rouat, J., Larochelle, H., and Courville, A. C. (2017). Home: a household multimodal environment. *CoRR*, abs/1711.11017.

Chen, D., Fisch, A., Weston, J., and Bordes, A. (2017). Reading wikipedia to answer open-domain questions. *CoRR*, abs/1704.00051.

Chevalier-Boisvert, M., Bahdanau, D., Lahlou, S., Willems, L., Saharia, C., Nguyen, T. H., and Bengio, Y. (2018). Babyai: First steps towards grounded language learning with a human in the loop. *CoRR*, abs/1810.08272.

Choi, E., He, H., Iyyer, M., Yatskar, M., Yih, W., Choi, Y., Liang, P., and Zettlemoyer, L. (2018). Quac : Question answering in context. *CoRR*, abs/1808.07036.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. (2018). Think you have solved question answering? try arc, the AI2 reasoning challenge. *CoRR*, abs/1803.05457.

Cobbe, K., Klimov, O., Hesse, C., Kim, T., and Schulman, J. (2018). Quantifying generalization in reinforcement learning. *CoRR*, abs/1812.02341.

Côté, M.-A., Kádár, A., Yuan, X., Kybartas, B., Barnes, T., Fine, E., Moore, J., Hausknecht, M., Asri, L. E., Adada, M., Tay, W., and Trischler, A. (2018). Textworld: A learning environment for text-based games. *CoRR*, abs/1806.11532.

Das, A., Datta, S., Gkioxari, G., Lee, S., Parikh, D., and Batra, D. (2017). Embodied question answering. *CoRR*, abs/1711.11543.

Das, R., Munkhdalai, T., Yuan, X., Trischler, A., and McCallum, A. (2018). Building dynamic knowledge graphs from text using machine reading comprehension. *CoRR*, abs/1810.05682.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Dunn, M., Sagun, L., Higgins, M., Güney, V. U., Cirik, V., and Cho, K. (2017). Searchqa: A new q&a dataset augmented with context from a search engine. *CoRR*, abs/1704.05179.

Fortunato, M., Azar, M. G., Piot, B., Menick, J., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., Blundell, C., and Legg, S. (2017). Noisy networks for exploration. *CoRR*, abs/1706.10295.

Gordon, D., Kembhavi, A., Rastegari, M., Redmon, J., Fox, D., and Farhadi, A. (2017). IQA: visual question answering in interactive environments. *CoRR*, abs/1712.03316.

Hausknecht, M., Loynd, R., Yang, G., Swaminathan, A., and Williams, J. D. (2019). Nail: A general interactive fiction agent.

Hessel, M., Modayil, J., van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M. G., and Silver, D. (2017). Rainbow: Combining improvements in deep reinforcement learning. *CoRR*, abs/1710.02298.

Jia, R. and Liang, P. (2017). Adversarial examples for evaluating reading comprehension systems. *arXiv preprint arXiv:1707.07328*.

Joshi, M., Choi, E., Weld, D. S., and Zettlemoyer, L. (2017). Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *CoRR*, abs/1705.03551.

Justesen, N., Torrado, R. R., Bontrager, P., Khalifa, A., Togelius, J., and Risi, S. (2018). Procedural level generation improves generality of deep reinforcement learning. *CoRR*, abs/1806.10729.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kuhlthau, C. (2004). *Seeking Meaning: A Process Approach to Library and Information Services*. Information management, policy, and services. Libraries Unlimited.

Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Kelcey, M., Devlin, J., Lee, K., Toutanova, K. N., Jones, L., Chang, M.-W., Dai, A., Uszkoreit, J., Le, Q., and Petrov, S. (2019). Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*.

Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., and Joulin, A. (2018). Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Narasimhan, K., Kulkarni, T., and Barzilay, R. (2015). Language understanding for text-based games using deep reinforcement learning. *arXiv preprint arXiv:1506.08941*.

Nguyen, T., Rosenberg, M., Song, X., Gao, J., Tiwary, S., Majumder, R., and Deng, L. (2016). MS MARCO: A human generated machine reading comprehension dataset. *CoRR*, abs/1611.09268.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch. In *NIPS-W*.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016). Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250.

Reddy, S., Chen, D., and Manning, C. D. (2018). Coqa: A conversational question answering challenge. *CoRR*, abs/1808.07042.

Rychalska, B., Basaj, D., Biecek, P., and Wroblewska, A. (2018). Does it care what you asked? understanding importance of verbs in deep learning qa system.

Schmidhuber, J. (2010). Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Trans. on Auton. Ment. Dev.*, 2(3):230–247.

Shi, T., Karpathy, A., Fan, L., Hernandez, J., and Liang, P. (2017). World of bits: An open-domain platform for web-based agents. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3135–3144, International Convention Centre, Sydney, Australia. PMLR.

Srivastava, R. K., Greff, K., and Schmidhuber, J. (2015). Highway networks. *CoRR*, abs/1505.00387.

Talmor, A. and Berant, J. (2018). The web as a knowledge-base for answering complex questions. *CoRR*, abs/1803.06643.

Trischler, A., Wang, T., Yuan, X., Harris, J., Sordoni, A., Bachman, P., and Suleman, K. (2016). Newsqa: A machine comprehension dataset. *CoRR*, abs/1611.09830.

Wang, S. and Jiang, J. (2016). Machine comprehension using match-lstm and answer pointer. *CoRR*, abs/1608.07905.

Wang, W., Yang, N., Wei, F., Chang, B., and Zhou, M. (2017). Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 189–198. Association for Computational Linguistics.

Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3):279–292.

Welbl, J., Stenetorp, P., and Riedel, S. (2017). Constructing datasets for multi-hop reading comprehension across documents. *CoRR*, abs/1710.06481.

Weston, J., Bordes, A., Chopra, S., Rush, A. M., van Merrinboer, B., Joulin, A., and Mikolov, T. (2015). Towards ai-complete question answering: A set of prerequisite toy tasks. *CoRR*.

Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W. W., Salakhutdinov, R., and Manning, C. D. (2018). Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *CoRR*, abs/1809.09600.

Yin, X. and May, J. (2019). Learn how to cook a new recipe in a new house: Using map familiarization, curriculum learning, and common sense to learn families of text-based adventure games. *CoRR*, abs/1908.04777.

Yu, A. W., Dohan, D., Luong, M.-T., Zhao, R., Chen, K., Norouzi, M., and Le, Q. V. (2018). Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*.

Yuan, X., Côté, M., Sordoni, A., Laroche, R., des Combes, R. T., Hausknecht, M. J., and Trischler, A. (2018). Counting to explore and generalize in text-based games. *CoRR*, abs/1806.11525.

## A Details of QA-DQN

### Notations

In this section, we use *game step* $t$ to denote one round of interaction between an agent with the QAit environment. We use $o_t$ to denote text observation at game step $t$, and $q$ to denote question text. We use $L$ to refer to a linear transformation. Brackets $[\cdot;\cdot]$ denote vector concatenation.

### A.1 Encoder

We use a transformer-based text encoder, which consists of an embedding layer, two stacks of transformer blocks (denoted as encoder transformer blocks and aggregation transformer blocks), and an attention layer.

In the embedding layer, we aggregate both word- and character-level information to produce a vector for each token in text. Specifically, word embeddings are initialized by the 300-dimensional fastText (Mikolov et al., 2018) word vectors trained on Common Crawl (600B tokens), they are fixed during training. Character level embedding vectors are initialized with 32-dimensional random vectors. A convolutional layer with 64 kernels of size 5 is then used to aggregate the sequence of characters. We use a max pooling layer on the character dimension, then a multi-layer perceptron (MLP) of output size 64 is used to aggregate the concatenation of word- and character-level representations. Highway network (Srivastava et al., 2015) is applied on top of this MLP. The resulting vectors are used as input to the encoding transformer blocks.

Each encoding transformer block consists of a stack of convolutional layers, a self-attention layer, and an MLP. In which, each convolutional layer has 64 filters, each kernel's size is 7, there are 2 such convolutional layers that share weights. In the self-attention layer, we use a block hidden size of 64, as well as a single head attention mechanism. Layernorm and dropout are applied after each component inside the block. We add positional encoding into each block's input. We use one layer of such an encoding block.

At a game step $t$, the encoder processes text observation $o_t$ and question $q$, context aware encoding $h_{o_t} \in \mathbb{R}^{L^{o_t} \times H_1}$ and $h_q \in \mathbb{R}^{L^q \times H_1}$ are generated, where $L^{o_t}$ and $L^q$ denote number of tokens in $o_t$ and $q$ respectively, $H_1$ is 64. Following (Yu et al., 2018), we use an context-query attention layer to aggregate the two representations $h_{o_t}$ and $h_q$.

Specifically, the attention layer first uses two MLPs to convert both $h_{o_t}$ and $h_q$ into the same space, the resulting tensors are denoted as $h'_{o_t} \in \mathbb{R}^{L^{o_t} \times H_2}$ and $h'_q \in \mathbb{R}^{L^q \times H_2}$, in which $H_2$ is 64.

Then, a tri-linear similarity function is used to compute the similarities between each pair of $h'_{o_t}$ and $h'_q$ items:

$$S = W[h'_{o_t}; h'_q; h'_{o_t} \odot h'_q], \tag{1}$$

where $\odot$ indicates element-wise multiplication, $W$ is trainable parameters of size 64.

Softmax of the resulting similarity matrix $S$ along both dimensions are computed, this produces $S^A$ and $S^B$. Information in the two representations are then aggregated by:

$$\begin{aligned} h_{oq} &= [h'_{o_t}; P; h'_{o_t} \odot P; h'_{o_t} \odot Q], \\ P &= S_q h'^{\top}_q, \\ Q &= S_q S^{\top}_{o_t} h'^{\top}_{o_t}, \end{aligned} \tag{2}$$

where $h_{oq}$ is aggregated observation representation.

On top of the attention layer, a stack of aggregation transformer blocks is used to further map the observation representations to action representations and answer representations. The structure of aggregation transformer blocks are the same as the encoder transformer blocks, except the kernel size of convolutional layer is 5, and the number of blocks is 3.

Let $M_t \in \mathbb{R}^{L^{o_t} \times H_3}$ denote the output of the stack of aggregation transformer blocks, where $H_3$ is 64.

### A.2 Command Generator

The command generator takes the hidden representations $M_t$ as input, it estimates Q-values for all action, modifier, and object words, respectively. It consists of a shared Multi-layer Perceptron (MLP) and three MLPs for each of the components:

$$\begin{aligned} R_t &= \text{ReLU}(L_{\text{shared}}(\text{mean}(M_t)), \\ Q_{t,action} &= L_{\text{action}}(R_t), \\ Q_{t,modifier} &= L_{\text{modifier}}(R_t), \\ Q_{t,object} &= L_{\text{object}}(R_t). \end{aligned} \tag{3}$$

In which, the output size of $L_{\text{shared}}$ is 64; the dimensionalities of the other 3 MLPs are depending on the number of the amount of action, modifier

and object words available, respectively. The overall Q-value is the sum of the three components:

$$Q_t = Q_{t,action} + Q_{t,modifier} + Q_{t,object}. \quad (4)$$

### A.3 Question Answerer

Similar to (Yu et al., 2018), we append an extra stacks of aggregation transformer blocks on top of the aggregation transformer blocks to compute answer positions:

$$\begin{aligned} U &= \text{ReLU}(L_0[M_t; M'_t]). \\ \beta &= \text{softmax}(L_1(U)). \end{aligned} \quad (5)$$

In which $M'_t \in \mathbb{R}^{L^{o_t} \times H_3}$ is output of the extra transformer stack, $L_0$, $L_1$ are trainable parameters with output size 64 and 1, respectively.

For location questions, the agent outputs $\beta$ as the probability distribution of each word in observation $o_t$ being the answer of the question.

For binary classification questions, we apply an MLP, which takes weighted sum of matching representations as input, to compute a probability distribution $p(y)$ over both possible answers:

$$\begin{aligned} D &= \sum_i (\beta^i \cdot M'_t), \\ p(y) &= \text{softmax}(L_4(\tanh(L_3(D)))). \end{aligned} \quad (6)$$

Output size of $L_3$ and $L_4$ are 64 and 2, respectively.

### A.4 Deep Q-Learning

In a text-based game, an agent takes an action $a$[4] in state $s$ by consulting a state-action value function $Q(s, a)$, this value function is as a measure of the action's expected long-term reward. Q-Learning helps the agent to learn an optimal $Q(s, a)$ value function. The agent starts from a random Q-function, it gradually updates its Q-values by interacting with environment, and obtaining rewards. Following Mnih et al. (2015), the Q-value function is approximated with a deep neural network.

We make use of a replay buffer. During playing the game, we cache all transitions into the replay buffer without updating the parameters. We periodically sample a random batch of transitions from the replay buffer. In each transition, we update the parameters $\theta$ to reduce the discrepancy between the predicted value of current state $Q(s_t, a_t)$ and the expected Q-value given the reward $r_t$ and the value of next state $\max_a Q(s_{t+1}, a)$.

We minimize the temporal difference (TD) error, $\delta$:

$$\delta = Q(s_t, a_t) - (r_t + \gamma \max_a Q(s_{t+1}, a)), \quad (7)$$

in which, $\gamma$ indicates the discount factor. Following the common practice, we use the Huber loss to minimize the TD error. For a randomly sampled batch with batch size $B$, we minimize:

$$\begin{aligned} \mathcal{L} &= \frac{1}{|B|} \sum \mathcal{L}(\delta), \\ \text{where } \mathcal{L}(\delta) &= \begin{cases} \frac{1}{2}\delta^2 & \text{for } |\delta| \leq 1, \\ |\delta| - \frac{1}{2} & \text{otherwise.} \end{cases} \end{aligned} \quad (8)$$

As described in Section 3.3.1, we design the sufficient information bonus to teach an agent to stop as soon as it has gathered enough information to answer the question. Therefore we assign this reward at the game step where the agent generates **`wait`** command (or it is forced to stop).

It is worth mentioning that for attribute type questions (considerably the most difficult question type in QAit, where the training signal is very sparse), we provide extra rewards to help QA-DQN to learn.

Specifically, we take a reward similar to as used in location questions: 1.0 if the agent has observed the object mentioned in the question. we also use a reward similar to as used in existence questions: the agent is rewarded by the coverage of its exploration. The two extra rewards are finally added onto the sufficient information bonus for attribute question, both with coefficient of 0.1.

## B Implementation Details

During training with vanilla DQN, we use a replay memory of size 500,000. We use $\epsilon$-greedy, where the value of $\epsilon$ anneals from 1.0 to 0.1 within 100,000 episodes. We start updating parameters after 1,000 episodes of playing. We update our network after every 20 game steps. During updating, we use a mini-batch of size 64. We use *Adam* (Kingma and Ba, 2014) as the step rule for optimization, The learning rate is set to 0.00025.

When our agent is trained with Rainbow algorithm, we follow Hessel et al. (2017) on most of the hyper-parameter settings. The four MLPs $L_{\text{shared}}$, $L_{\text{action}}$, $L_{\text{modifier}}$ and $L_{\text{object}}$ as described

[4] In our case, $a$ is a triplet contains {action, modifier, object} as described in Section 2.4.

in Eqn. 3 are Noisy Nets layers (Fortunato et al., 2017) when the agent is trained in Rainbow setting. Detailed hyper-parameter setting of our Rainbow agent are shown in Table 6.

| Parameter | Value |
|---|---|
| Exploration $\epsilon$ | 0 |
| Noisy Nets $\sigma_0$ | 0.5 |
| Target Network Period | 1000 episodes |
| Multi-step returns $n$ | $n \sim \text{Uniform}[1, 3]$ |
| Distributional atoms | 51 |
| Distributional min/max values | [-10, 10] |

Table 6: Hyper-parameter setup for rainbow agent.

The model is implemented using PyTorch (Paszke et al., 2017).

## C Supported Text Commands

All supported text commands are listed in Table 7.

## D Heuristic Conditions for Attribute Questions

Here, we derived some heuristic conditions to determine when an agent has gathered enough information to answer a given attribute question. Those conditions are used as part of the reward shaping for our proposed agent (Section 3.3.1). In Table 8, for each attribute we list all the commands for which their outcome (pass or fail) gives enough information to answer the question correctly. Also, in order for a command's outcome to be informative, each command needs to be executed while some state conditions hold. For example, to determine if an object is indeed a **heat_source**, the agent needs to try to cook something that is cookable and uncooked while standing next to the given object.

## E Full results

We provide full results of our agents on **fixed map** games in Table 9, and provide full results of our agents on **random map** games in Table 10. To help investigating the generalizability of the sufficient information bonus we used in our proposed agent, we also report the rewards during both training and test phases. Note during test phase, we do not update parameters with the rewards.

| Command | Description |
|---|---|
| **look** | describe the current location |
| **inventory** | display the player's inventory |
| **go** ⟨**dir**⟩ | move the player to north, east, south, or west |
| **examine ...** | examine something more closely |
| **open ...** | open a door or a container |
| **close ...** | close a door or a container |
| **eat ...** | eat edible object |
| **drink ...** | drink drinkable object |
| **drop ...** | drop an object on the floor |
| **take ...** | take an object from the floor, a container, or a supporter |
| **put ...** | put an object onto a supporter (supporter must be present at the location) |
| **insert ...** | insert an object into a container (container must be present at the location) |
| **cook ...** | cook an object (heat source must be present at the location) |
| **slice ...** | slice cuttable object (a sharp object must be in the player's inventory) |
| **chop ...** | chop cuttable object (a sharp object must be in the player's inventory |
| **dice ...** | dice cuttable object (a sharp object must be in the player's inventory) |
| **wait** | stop interaction |

Table 7: Supported command list.

| Attribute | Command | State | Pass | Fail | Explanation |
|---|---|---|---|---|---|
| **sharp** | **cut** *cuttable* | holding (*cuttable*) & uncut (*cuttable*) & holding (object) | 1 | 1 | Trying to cut something cuttable that hasn't been cut yet while holding the object. |
| | **take** object | reachable(object) | 0 | 1 | Sharp objects should be portable. |
| **cuttable** | **cut** object | holding (object) & holding (*sharp*) | 1 | 1 | Trying to cut the object while holding something sharp. |
| | **take** object | reachable (object) | 0 | 1 | Cuttable object should be portable. |
| **edible** | **eat** object | holding (object) | 1 | 1 | Trying to eat the object. |
| | **take** object | reachable (object) | 0 | 1 | Edible objects should be portable. |
| **drinkable** | **drink** object | holding (object) | 1 | 1 | Trying to drink the object. |
| | **take** object | reachable (object) | 0 | 1 | Drinkable objects should be portable. |
| **holder** | **–** | on (*portable*, object) | 1 | 0 | Observing object(s) on a supporter. |
| | | in (*portable*, object) | 1 | 0 | Observing object(s) inside a container. |
| | **take** object | reachable (object) | 1 | 0 | Holder objects should not be portable. |
| **portable** | **–** | holding (object) | 1 | 0 | Holding the object means it is portable. |
| | **take** object | reachable (object) | 1 | 1 | Portable objects can be taken. |
| **heat_source** | **cook** *cookable* | holding (*cookable*) & uncooked (*cookable*) & reachable (object) | 1 | 1 | Trying to cook something cookable that hasn't been cooked yet while being next to the object. |
| | **take** object | reachable (object) | 1 | 0 | Heat source objects should not be portable. |
| **cookable** | **cook** object | holding (object) & reachable (*heat_source*) | 1 | 1 | Trying to cook the object while being next to a heat source. |
| | **take** object | reachable(object) | 0 | 1 | Cookable objects should be portable. |
| **openable** | **open** object | reachable (object) & closed (object) | 1 | 1 | Trying to open the closed object. |
| | **close** object | reachable (object) & open (object) | 1 | 1 | Trying to close the open object. |

Table 8: Heuristic conditions for determining whether the agent has enough information to answer a given attribute question. We use "object" to refer to the object mentioned in the question. Words in italics represents placeholder that can be replaced by any object from the environment that has the appropriate attribute (e.g. carrot could be used as a *cuttable*). The columns Pass and Fail represent how much reward the agent will receive given the corresponding command's outcome (resp. success or failure). NB: **cut** can mean any of the following commands: **slice**, **dice**, or **chop**

| Model | Location | | Existence | | Attribute | |
|---|---|---|---|---|---|---|
| | Train | Test | Train | Test | Train | Test |
| Human | – | 1.000 | – | 1.000 | – | 1.000 |
| Random | – | 0.027 | – | 0.497 | – | 0.496 |
| 1 game | | | | | | |
| DQN | 0.972(0.972) | 0.122(0.160) | 1.000(0.881) | 0.628(0.124) | 1.000(0.049) | 0.500(0.035) |
| DDQN | 0.960(0.960) | 0.156(0.178) | 1.000(0.647) | 0.624(0.148) | 1.000(0.023) | 0.498(0.033) |
| Rainbow | 0.562(0.562) | 0.164(0.178) | 1.000(0.187) | 0.616(0.083) | 1.000(0.049) | 0.516(0.039) |
| 2 games | | | | | | |
| DQN | 0.698(0.698) | 0.168(0.182) | 0.948(0.700) | 0.574(0.136) | 1.000(0.011) | 0.510(0.028) |
| DDQN | 0.702(0.702) | 0.172(0.178) | 0.882(0.571) | 0.550(0.109) | 1.000(0.098) | 0.508(0.036) |
| Rainbow | 0.734(0.734) | 0.160(0.168) | 0.878(0.287) | 0.616(0.085) | 1.000(0.030) | 0.524(0.022) |
| 10 games | | | | | | |
| DQN | 0.654(0.654) | 0.180(0.188) | 0.822(0.390) | 0.568(0.156) | 1.000(0.055) | 0.518(0.030) |
| DDQN | 0.608(0.608) | 0.188(0.208) | 0.842(0.479) | 0.566(0.128) | 1.000(0.064) | 0.516(0.036) |
| Rainbow | 0.616(0.616) | 0.156(0.170) | 0.768(0.266) | 0.590(0.131) | 0.998(0.059) | 0.520(0.023) |
| 100 games | | | | | | |
| DQN | 0.498(0.498) | 0.194(0.206) | 0.756(0.139) | 0.614(0.160) | 0.838(0.019) | 0.498(0.014) |
| DDQN | 0.456(0.458) | 0.168(0.196) | 0.768(0.134) | 0.650(0.216) | 0.878(0.020) | 0.528(0.017) |
| Rainbow | 0.340(0.340) | 0.156(0.160) | 0.762(0.129) | 0.602(0.207) | 0.924(0.044) | 0.524(0.022) |
| 500 games | | | | | | |
| DQN | 0.430(0.430) | 0.224(0.244) | 0.742(0.136) | 0.674(0.279) | 0.700(0.015) | **0.534**(0.014) |
| DDQN | 0.406(0.406) | 0.218(0.228) | 0.734(0.173) | 0.626(0.213) | 0.714(0.021) | 0.508(0.026) |
| Rainbow | 0.358(0.358) | 0.190(0.196) | 0.768(0.187) | 0.656(0.207) | 0.736(0.032) | 0.496(0.029) |
| unlimited games | | | | | | |
| DQN | 0.300(0.300) | 0.216(0.216) | 0.752(0.119) | 0.662(0.246) | 0.562(0.034) | 0.514(0.016) |
| DDQN | 0.318(0.318) | 0.258(0.258) | 0.744(0.168) | 0.628(0.134) | 0.572(0.027) | 0.480(0.024) |
| Rainbow | 0.316(0.330) | **0.280**(0.280) | 0.734(0.157) | **0.692**(0.157) | 0.566(0.017) | 0.514(0.014) |

Table 9: Agent performance on **fixed map** games. Accuracies in percentage are shown in black. We also investigate the sufficient information bonus used in our agent proposed in Section 3.3.1, which are shown in blue.

| Model | Location | | Existence | | Attribute | |
|---|---|---|---|---|---|---|
| | Train | Test | Train | Test | Train | Test |
| Human | – | 1.000 | – | 1.000 | – | 0.750 |
| Random | – | 0.034 | – | 0.500 | – | 0.499 |
| 2 games | | | | | | |
| DQN | 0.990(0.990) | 0.148(0.162) | 1.000(0.779) | 0.638(0.157) | 1.000(0.039) | 0.534(0.033) |
| DDQN | 0.978(0.978) | 0.146(0.152) | 1.000(0.727) | 0.602(0.158) | 1.000(0.043) | **0.544**(0.032) |
| Rainbow | 0.916(0.916) | 0.178(0.178) | 0.972(0.314) | 0.602(0.136) | 1.000(0.025) | 0.512(0.021) |
| 10 games | | | | | | |
| DQN | 0.818(0.818) | 0.156(0.160) | 0.898(0.607) | 0.566(0.142) | 1.000(0.056) | 0.518(0.036) |
| DDQN | 0.794(0.794) | 0.142(0.154) | 0.868(0.575) | 0.606(0.153) | 1.000(0.037) | 0.500(0.033) |
| Rainbow | 0.670(0.670) | 0.144(0.170) | 0.828(0.468) | 0.586(0.128) | 1.000(0.071) | 0.530(0.018) |
| 100 games | | | | | | |
| DQN | 0.550(0.550) | 0.184(0.204) | 0.758(0.230) | 0.668(0.181) | 0.878(0.021) | 0.524(0.017) |
| DDQN | 0.524(0.524) | 0.188(0.204) | 0.754(0.365) | 0.662(0.205) | 0.890(0.025) | **0.544**(0.019) |
| Rainbow | 0.442(0.442) | 0.174(0.184) | 0.754(0.285) | 0.654(0.190) | 0.878(0.044) | 0.504(0.032) |
| 500 games | | | | | | |
| DQN | 0.430(0.430) | 0.204(0.216) | 0.752(0.162) | 0.678(0.214) | 0.678(0.019) | 0.530(0.017) |
| DDQN | 0.458(0.458) | 0.222(0.246) | 0.754(0.158) | 0.656(0.188) | 0.716(0.024) | 0.486(0.023) |
| Rainbow | 0.370(0.370) | 0.172(0.178) | 0.748(0.275) | 0.678(0.191) | 0.636(0.020) | 0.494(0.017) |
| unlimited games | | | | | | |
| DQN | 0.316(0.316) | 0.188(0.188) | 0.728(0.213) | 0.668(0.218) | 0.812(0.055) | 0.506(0.018) |
| DDQN | 0.326(0.326) | 0.206(0.206) | 0.740(0.246) | **0.694**(0.196) | 0.580(0.023) | 0.482(0.017) |
| Rainbow | 0.340(0.340) | **0.258**(0.258) | 0.728(0.210) | 0.686(0.193) | 0.564(0.018) | 0.470(0.017) |

Table 10: Agent performance on **random map** games. Accuracies in percentage are shown in black. We also investigate the sufficient information bonus used in our agent proposed in Section 3.3.1, which are shown in blue.

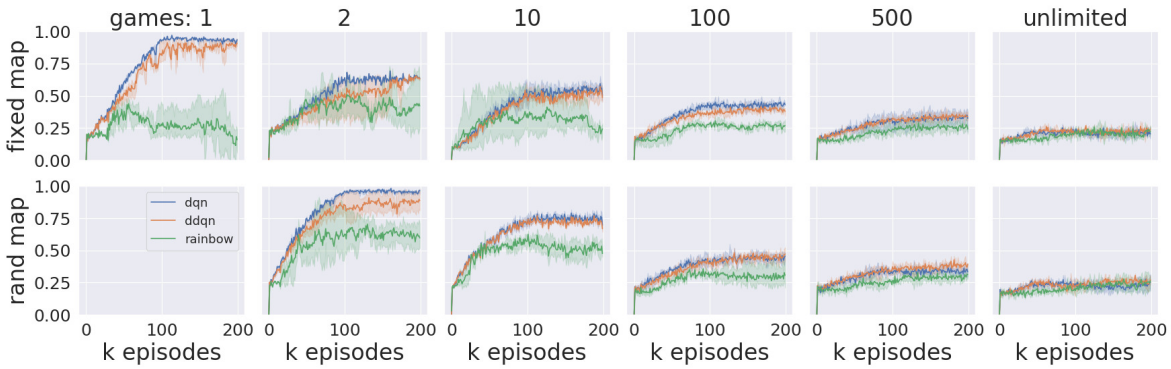


Figure 4: Training accuracy over episodes on **location** questions. Upper row: **fixed map**, 1/2/10/100/500/unlimited games; Lower row: **random map**, 2/10/100/500/unlimited games.

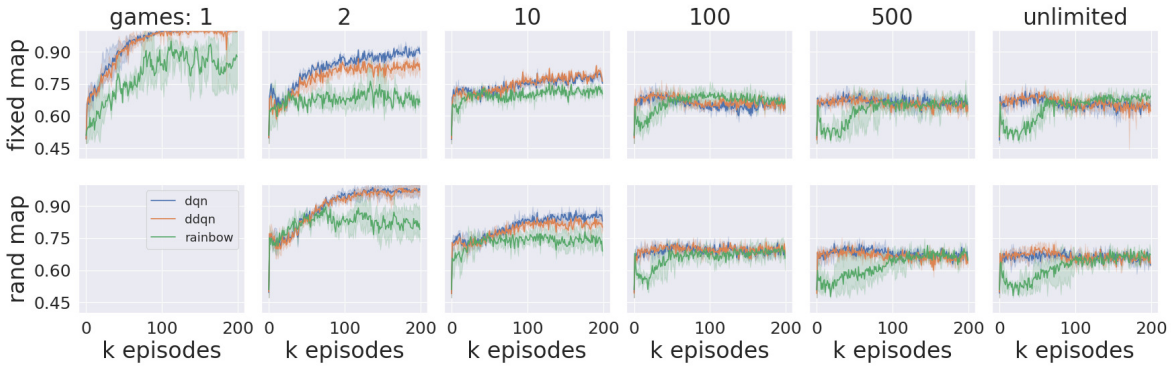


Figure 5: Training accuracy over episodes on **existence** questions. Upper row: **fixed map**, 1/2/10/100/500/unlimited games; Lower row: **random map**, 2/10/100/500/unlimited games.

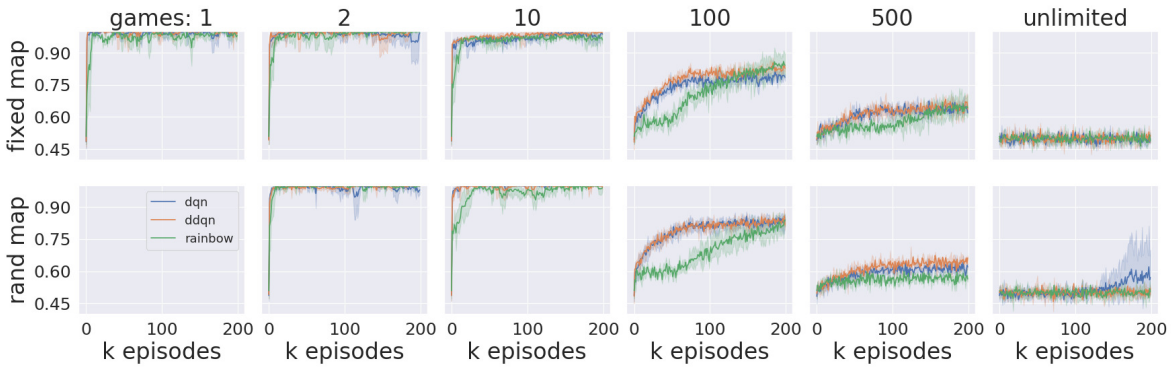


Figure 6: Training accuracy over episodes on **attribute** questions. Upper row: **fixed map**, 1/2/10/100/500/unlimited games; Lower row: **random map**, 2/10/100/500/unlimited games.

# What's Missing: A Knowledge Gap Guided Approach for Multi-hop Question Answering

**Tushar Khot** and **Ashish Sabharwal** and **Peter Clark**
Allen Institute for Artificial Intelligence, Seattle, WA, U.S.A.
{tushark,ashishs,peterc}@allenai.org

## Abstract

Multi-hop textual question answering requires combining information from multiple sentences. We focus on a natural setting where, unlike typical reading comprehension, only *partial information* is provided with each question. The model must retrieve and use additional knowledge to correctly answer the question. To tackle this challenge, we develop a novel approach that explicitly identifies the *knowledge gap* between a key span in the provided knowledge and the answer choices. The model, GapQA, learns to fill this gap by determining the relationship between the span and an answer choice, based on retrieved knowledge targeting this gap. We propose jointly training a model to simultaneously fill this knowledge gap and compose it with the provided partial knowledge. On the OpenBookQA dataset, given partial knowledge, explicitly identifying what's missing substantially outperforms previous approaches.

## 1 Introduction

Reading Comprehension datasets (Richardson et al., 2013; Rajpurkar et al., 2016; Joshi et al., 2017) have gained interest as benchmarks to evaluate a system's ability to understand a document via question answering (QA). Since many of these early datasets only required a system to understand a single sentence, new datasets were specifically designed to focus on the problem of multi-hop QA, i.e., reasoning across sentences (Khashabi et al., 2018; Welbl et al., 2018; Yang et al., 2018).

While this led to improved language understanding, the tasks still assume that a system is provided with *all* knowledge necessary to answer the question. In practice, however, we often only have access to *partial knowledge* when dealing with such multi-hop questions, and must retrieve additional facts (the knowledge "gaps") based on the question and the provided knowledge. Our goal is to identify such gaps and fill them using an external knowledge source.

**Question:**
*Which of these would let the most heat travel through?*
A) a new pair of jeans.
B) a steel spoon in a cafeteria.
C) a cotton candy at a store.
D) a calvin klein cotton hat.
**Core Fact:**
Metal lets heat travel through.

**Knowledge Gap** (similar gaps for other choices):
steel spoon in a cafeteria _____ metal.

**Filled Gap** (relation identified using KB):
steel spoon in a cafeteria *is made of* metal.

Figure 1: A sample OpenBookQA question, the identified knowledge gap based on partial information in the core fact, and relation (*is made of*) identified from a KB to fill that gap.

The recently introduced challenge of *open book* question answering (Mihaylov et al., 2018) highlights this phenomenon. The questions in the corresponding dataset, OpenBookQA, are derived from a science fact in an "open book" of about 1300 facts. To answer these questions, a system must not only identify a relevant "core" science fact from this small book, but then also retrieve additional common knowledge from large external sources in order to successfully apply this core fact to the question. Consider the example in Figure 1. The core science fact *metal lets heat to travel through* points to *metal* as the correct answer, but it is not one of the 4 answer choices. Given this core fact (the "partial knowledge"), a system must still use broad external knowledge to fill the remaining gap, that is, identify which answer choice *contains* or *is made of* metal.

This work focuses on *QA under partial knowledge*. This turns out to be a surprisingly chal-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2814–2828, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

lenging task in itself; indeed, the partial knowledge models of Mihaylov et al. (2018) achieve a score of only 55% on OpenBookQA, far from human performance of 91%. Since this and several recent multi-hop datasets use the multiple-choice setting (Welbl et al., 2018; Khashabi et al., 2018; Lai et al., 2017), we assume access to potential answers to a question. While our current model relies on this for a direct application to span-prediction based RC datasets, the idea of identifying knowledge gaps can be used to create novel RC specific models.

We demonstrate that an intuitive approach leads to a strong model: first identify the knowledge gap and then fill this gap, i.e., identify the missing relation using external knowledge. We primarily focus on the OpenBookQA dataset since it is the only dataset currently available that provides partial context. However, we believe such an approach is also applicable to the broader setting of multi-hop RC datasets, where the system could start reasoning with one sentence and fill remaining gap(s) using sentences from other passages.

Our model operates in two steps. First, it predicts a key span in the core fact ("metal" in the above example). Second, it answers the question by identifying the relationship between the key span and answer choices, i.e., by *filling* the knowledge gap. This second step can be broken down further: (a) retrieve relevant knowledge from resources such as ConceptNet (Speer et al., 2017) and large-scale text corpora (Clark et al., 2018); (c) based on this, predict potential relations between the key span and an answer choice; and (d) compose the core fact with this filled gap.

We collect labels for knowledge gaps on ∼30% of the training questions, and train two modules capturing the two main steps above. The first exploits an existing RC model and large-scale dataset to train a span-prediction model. The second uses multi-task learning to train a separate QA model to jointly predict the relation representing the gap, as well as the final answer. For questions without labelled knowledge gaps, the QA model is trained based solely on the predicted answer.

Our model outperforms the previous state-of-the-art partial knowledge models by 6.5% (64.41 vs 57.93) on a targeted subset of OpenBookQA amenable to gap-based reasoning. Even without missing fact annotations, our model with a simple heuristic to identify missing gaps still outperforms previous models by 3.4% (61.38 vs. 57.93). It also generalizes to questions that were not its target, with 3.6% improvement (59.40 vs. 55.84) on the full OpenBookQA set.

Overall, the contributions of this work are: (1) an analysis and dataset[1] of knowledge gaps for QA under partial knowledge; (2) a novel two-step approach of first identifying and then filling knowledge gaps for multi-hop QA; (3) a model[1] that simultaneously learns to fill a knowledge gap using retrieved external knowledge and compose it with partial knowledge; and (4) new state-of-the-art results on QA with partial knowledge (+6.5% using annotations on only 30% of the questions).

## 2 Related Work

**Text-Based QA.** Reading Comprehension (RC) datasets probe language understanding via question answering. While several RC datasets (Rajpurkar et al., 2016; Trischler et al., 2017; Joshi et al., 2017) can be addressed with single sentence understanding, newer datasets (Weston et al., 2015; Welbl et al., 2018; Khashabi et al., 2018; Yang et al., 2018) specifically target multi-hop reasoning. In both cases, all relevant information, barring some linguistic knowledge, is provided or the questions are unanswerable (Rajpurkar et al., 2018). This allows using an attention-based approach of indirectly combining information (Dhingra et al., 2018; Cao et al., 2019; Song et al., 2018).

On the other hand, open domain question answering datasets (Clark et al., 2016, 2018) come with no context, and require first retrieving relevant knowledge before reasoning with it. Retrieving this knowledge from noisy textual corpora, while simultaneously solving the reasoning problem, can be challenging, especially when questions require multiple facts. This results in simple approaches (e.g. word-overlap/PMI-based approaches), that do not heavily rely on the retrieval quality, being competitive with other complex reasoning methods that assume clean knowledge (Clark et al., 2016; Jansen et al., 2017; Angeli et al., 2016). To mitigate this issue, semi-structured tables (Khashabi et al., 2016; Jansen et al., 2018) have been manually authored targeting a subset of these questions. However, these tables are expensive to create and these questions often need multiple hops (sometimes up to

[1]The code and associated dataset are available at https://github.com/allenai/missing-fact.

16 (Jansen et al., 2018)), making reasoning much more complex.

OpenBookQA dataset (Mihaylov et al., 2018) was proposed to limit the retrieval problem by providing a set of ∼1300 facts as an 'open book' for the system to use. Every question is based on one of the core facts, and in addition requires basic external knowledge such as hypernymy, definition, and causality. We focus on the task of question answering under partial context, where the core fact for each question is available to the system.

**Knowledge-Based QA.** Another line of research is answering questions (Bordes et al., 2015; Pasupat and Liang, 2015; Berant et al., 2013) over a structured knowledge base (KB) such as Freebase (Bollacker et al., 2008). Depending on the task, systems map questions to a KB query with varying complexity: from complex semantic parses (Krishnamurthy et al., 2017) to simple relational lookup (Petrochuk and Zettlemoyer, 2018). Our sub-task of filling the knowledge gap can be viewed as KB QA task with knowledge present in a KB or expected to be inferred from text.

Some RC systems (Mihaylov and Frank, 2018; Kadlec et al., 2016) and Textual Entailment (TE) models (Weissenborn et al., 2017; Inkpen et al., 2018) incorporate external KBs to provide additional context to the model for better language understanding. However, we take a different approach of using this background knowledge in an explicit inference step (i.e. hop) as part of a multi-hop QA model.

## 3 Knowledge Gaps

We now take a deeper look at categorizing knowledge gaps into various classes. While grounded in OpenBookQA, this categorization is relevant for other multi-hop question sets as well.[2] We will then discuss how to effectively annotate such gaps.

### 3.1 Understanding Gaps: Categorization

We analyzed the additional facts needed for answering 75 OpenBookQA questions. These facts naturally fall into three classes, based on the knowledge gap they are trying to fill: (1) Question-to-Fact, (2) Fact-to-Answer, and (3) Question-to-Answer(Fact). Figure 2 shows a high-level overview, with simplified examples of each class of knowledge gap in Figures 3, 4, and 5.

[2] As mentioned earlier, in the RC setting, the first relevant sentence read by the system can be viewed as the core fact.

**Question-to-Fact Gap.** This gap exists between concepts in the question and the core fact. For example, in Figure 3, the knowledge that "Kool-aid" is a liquid is needed to even recognize that the fact is relevant.

**Fact-to-Answer Gap.** This gap captures the relationship between concepts in the core fact and the answer choices. For example, in Figure 4, the knowledge "Heat causes evaporation" is needed to relate "evaporated" in the fact to the correct answer "heat". Note that it is often possible to find relations connecting the fact to even incorrect answer choices. For example, "rainfall" could be connected to the fact using "evaporation leads to rainfall". Thus, identifying the correct relation and knowing if it can be composed with the core fact is critical, i.e., "evaporation causes liquid to disappear" and "evaporation leads to rainfall" do not imply that "rainfall causes liquid to disappear".

**Question-to-Answer(Fact) Gap.** Finally, some questions need additional knowledge to connect concepts in the question to the answer, based on the core fact. For example, composition questions (Figure 5) use the provided fact to replace parts of the original question with words from the fact.

Notably Question-to-Fact and Fact-to-Answer gaps are more common in OpenBookQA (44% and 86% respectively[3]), while the Question-to-Answer(Fact) gap is very rare (<20%). While all three gap classes pose important problems, we focus on Fact-to-Answer gap and assume that the core fact is provided. This is still a challenging problem as one must not only identify and fill the gap, but also learn to compose this filled gap with the input fact.

### 3.2 Annotating Gaps: Data Collection

Due to space constraints, details of our crowdsourcing process for annotating knowledge gaps, including the motivation behind various design choices as well as several examples, are deferred to the Appendix (Section A). Here we briefly summarize the final crowdsourcing design.

Our early pilots revealed that straightforward approaches to annotate knowledge gaps for all OpenBookQA questions lead to noisy labels. To address this, we (a) identified a **subset of questions** suitable for this annotation task and (b) split

[3] Some questions have both of these classes of gaps.

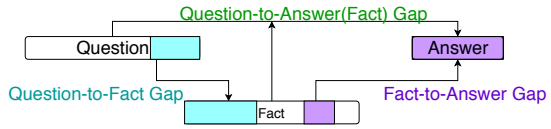


Figure 2: High-level overview of the kinds of knowledge gaps, assuming partial knowledge from the fact. In subsequent figures, the knowledge gap is indicated using highlighted text.

Question: What can cause liquid to disappear ?
Fact: If a liquid disappears then that liquid probably evaporated
Answer: Heat
Gap: Heat causes evaporation

Figure 4: Knowledge gap between the **fact** (evaporated) and the **answer** (Heat). While it is clear how to apply the knowledge, we need to know that "Heat causes evaporation" to identify the right answer.

Question: What can cause Kool-aid to disappear ?
Fact: If a liquid disappears then that liquid probably evaporated
Answer: Evaporation
Gap: Kool-aid is a liquid

Figure 3: Knowledge gap between the **question** (Kool-aid) and the **fact** (liquid). To apply the fact about liquids to the question, we need to know "Kool-aid is a liquid".

Question: What is the satellite of the blue planet ?
Fact: The blue planet refers to planet Earth
Answer: Moon
Gap: Moon is the satellite of Earth

Figure 5: Knowledge gap between the **question** and the **answer** using the **fact**. For some complex questions, the fact clarifies certain concepts in the question, (e.g., "blue planet"), leading to a reformulation of the question based on the fact (e.g., "What is the satellite of Earth?") which is captured by this gap.

| Question | Fact | Span | Relation | Gap |
|---|---|---|---|---|
| **Q:** A light bulb turns on when it receives energy from **A:** gasoline | a light bulb converts *electrical energy* into light energy when it is turned on | electrical energy | provides$^{-1}$, enables$^{-1}$ | (gasoline, provides, electrical energy) |
| **Q:** What makes the best wiring? **A:** Tungsten | wiring requires an *electrical conductor* | electrical conductor | isa$^{-1}$, madeof | (Tungsten, is an, electrical conductor) |

Table 1: Examples from KGD dataset. Note that the knowledge gap is captured in the form of (Span, Relation, Answer) but not explicitly annotated. $^{-1}$ is used to indicate that the argument order should be flipped.

Fact-to-Answer gap annotation into **two steps**: key term identification and relation identification.

**Question Subset.** First, we identified valid question-fact pairs where the fact supports the correct answer (verified via crowdsourcing) but does not trivially lead to the answer (fact only overlaps with the correct answer). Second, we noticed that the Fact-to-Answer gaps were much noisier for longer answer options, where you could write multiple knowledge gaps or a single complex knowledge gap. So we created *OBQA-Short*, the subset of OpenBookQA where answer choices have at most two non-stopword tokens. This contains over 50% of the original questions and is also the target set of our approach.

**Two-step Gap Identification.** Starting with the above pairs of questions with valid partial knowledge, the second task is to author facts that close the Fact-to-Answer **knowledge gap**. Again, initial iterations of the task resulted in poor quality, with workers often writing noisy facts that restated part of the provided fact or directly connect the question to the answer (skipping over the provided fact[4]). We noticed that the core fact often contains a key span that hints at the final answer. So we broke the task into two steps: (1) identify key terms (preferably a span) in the core fact that could answer the question, and (2) identify one or more relations[5] that hold between the key terms and the correct answer choice but not the incorrect choices. Table 1 shows example annotations of the gaps obtained through this process.

## Knowledge Gap Dataset: KGD

Our Knowledge Gap Dataset (KGD) contains key span and relation label annotations to capture knowledge gaps. To reduce noise, we only use knowledge gap annotations where at least two of three workers found a contiguous span from the core fact and a relation from our list. The final

[4]This was also noticed by the original authors of OpenBookQA dataset (Mihaylov et al., 2018).

[5]Workers preferably chose from a selected list of nine most common relations: {causes, definedAs, enables, isa, located in, made of, part of, provides, synonym of} and their inverses (except synonymy). These relations have also been found to be useful by prior approaches for science QA (Clark et al., 2014; Khashabi et al., 2016; Jansen et al., 2016, 2018).

| | Train | Dev | Test |
|---|---|---|---|
| Total #questions | 1151 | 117 | 121 |
| Total #question-facts | 1531 | 157 | 165 |
| Avg. # spans | 1.43 | 1.46 | 1.45 |
| Avg. # relations | 3.31 | 2.45 | 2.45 |

Table 2: Statistics of the train/dev/test split of the KGD dataset. The #question-fact pairs is higher than #questions as some questions may be supported by multiple facts. The average statistic computes the average number of unique spans and relations per question-fact pair.

dataset contains examples of the form {question, fact, spans, relations}, where each span is a substring of the input fact, and relations are the set of valid relations between the span and the correct answer (examples in Table 1 and stats in Table 2).

## 4 Knowledge-Gap Guided QA: GapQA

We first introduce the notation used to describe our QA system. For each question $q$ and fact $f$, the selected span is given by $s$ and the set of valid relations between this span and the correct choice is given by $r$. Borrowing notation from OpenBookQA, we refer to the question without the answer choices $c$ as the stem $q_s$, i.e., $q = q_s\ c$. We use $\hat{s}$ to indicate the predicted span and $\hat{r}$ for the predicted relations. We use $q_m$ and $f_m$ to represent the tokens in the question stem and fact respectively. Following the Turk task, our model first identifies the key span from the fact and then identifies the relation using retrieved knowledge.

### 4.1 Key Span Identification Model

Since the span selected from the fact often tends to be the answer to the question (c.f. Table 1), we can use a reading comprehension model to identify this span. The fact serves as the input passage and the question stem as the input question to the reading comprehension model. We used the Bi-Directional Attention Flow (BiDAF) model (Seo et al., 2017), an attention-based span prediction model designed for the SQuAD RC dataset (Rajpurkar et al., 2016). We refer the reader to the original paper for details about the model.

### 4.2 Knowledge Retrieval Module

Given the predicted span, we retrieve knowledge from two sources: triples from ConceptNet (Speer et al., 2017) and sentences from ARC corpus (Clark et al., 2018). ConceptNet contain (subject, relation, object) triples with relations such as */r/IsA*, */r/PartOf* that closely align with the relations in our gaps. Since ConceptNet can be incomplete or vague (e.g. */r/RelatedTo* relation), we also use the ARC corpus of 14M science-relevant sentences to improve our recall.

**Tuple Search.** To find relevant tuples connecting the predicted span $\hat{s}$ to the answer choice $c_i$, we select tuples where at least one token[6] in the subject matches $\hat{s}$ and at least one token in the object matches $c_i$ (or vice versa). We then score each tuple $t$ using the Jaccard score[7] and pick the top $k$ tuples for each $c_i$ ($k = 5$ in our experiments).

**Text Search.** To find the relevant sentences for $\hat{s}$ and $c_i$, we used ElasticSearch[8] with the query: $\hat{s}$ + $c_i$ (refer to Appendix D for more details). Similar to ConceptNet, we pick top 5 sentences for each answer choice. To ensure a consistent formatting of all knowledge sources, we convert the tuples into sentences using few hand-defined rules(described in Appendix C). Finally all the retrieved sentences are combined to produce the input KB for the model, $K$.

### 4.3 Question Answering Model

The question answering model takes as input the question $q_s$, answer choices $c$, fact $f$, predicted span, $\hat{s}$ and retrieved knowledge $K$. We use 300-dimensional 840B GloVe embeddings (Pennington et al., 2014) to embed each word in the inputs. We use a Bi-LSTM with 100-dimensional hidden states to compute the contextual encodings for each string, e.g., $\mathcal{E}_f \in \mathbb{R}^{f_m \times h}$. The question answering model selects the right answer using two components: (1) **Fact Relevance** module (2) **Relation Prediction** module.

**Fact Relevance.** This module is motivated by the intuition that a relevant fact will often capture a relation between concepts that align with the question and the correct answer (the cyan and magenta regions in Figure 2). To deal with the gaps between these concepts, this module relies purely on word embeddings while the next module will focus on using external knowledge.

We compute a question-weighted and answer-weighted representation of the fact to capture the part of the fact that links to the question and answer respectively. We compose these fact-based

[6] We use lower-cased, stemmed, non-stopword tokens.

[7] score($t$) = jacc(tokens($t$), tokens($\hat{s}$ + $c_i$)) where jacc(w1, w2) = $\frac{w1 \cap w2}{w1 \cup w2}$

[8] https://www.elastic.co/products/elasticsearch

representations to then identify how well the answer choice is supported by the fact.

To calculate the question-weighted fact representation, we first identify facts words with a high similarity to some question word ($\mathcal{V}_{q_s}(f)$) using the attention weights: $\mathcal{A}_{q_s,f} = \mathcal{E}_{q_s} \cdot \mathcal{E}_f \in \mathbb{R}^{q_m \times f_m}$

$$\mathcal{V}_{q_s}(f) = \text{softmax}_{f_m}\left(\max_{q_m} \mathcal{A}_{q_s,f}\right) \in \mathbb{R}^{1 \times f_m}$$

The final attention weights are similar to the Query-to-Context attention weights in BiDAF. The final question-weighted representation is:

$$\mathcal{S}_{q_s}(f) = \mathcal{V}_{q_s}(f) \cdot \mathcal{E}_f \in \mathbb{R}^{1 \times h} \quad (1)$$

We similarly compute the choice-weighted representation of fact as $\mathcal{S}_{c_i}(f)$. We compose these two representations by averaging[9] these two vectors $\mathcal{S}_{q_s c_i}(f) = (\mathcal{S}_{q_s}(f) + \mathcal{S}_{c_i}(f))/2$. We finally score the answer choice by comparing this representation with the aggregate fact representation, obtained by averaging too, as:

$$\text{score}_f(c_i) = \text{FF}\left(\bigotimes(\mathcal{S}_{q_s c_i}(f), \text{avg}(\mathcal{E}_f)\right)$$

where $\bigotimes(x, y) = [x - y; x * y] \in \mathbb{R}^{1 \times 2h}$ and FF is a feedforward neural network that outputs a scalar score for each answer choice.

**Filling the Gap: Relation Prediction.** The relation prediction module uses the retrieved knowledge to focus on the Fact-to-Answer gap by first predicting the relation between $\hat{s}$ and $c_i$ and then compose it with the fact to score the choice. We first compute the span and choice weighted representation($\mathbb{R}^{1 \times h}$) for each sentence $k_j$ in $K$ using the same operations as above:

$$\mathcal{S}_{\hat{s}}(k_j) = \mathcal{V}_{\hat{s}}(k_j) \cdot \mathcal{E}_{k_j} \; ; \; \mathcal{S}_{c_i}(k_j) = \mathcal{V}_{c_i}(k_j) \cdot \mathcal{E}_{k_j}$$

These representations capture the contextual embeddings of the words in the $k_j$ that most closely resemble words in $\hat{s}$ and $c_i$ respectively. We predict the kb-based relation between them based on the composition of these representations :

$$\mathcal{R}_j(\hat{s}, c_i) = \text{FF}\left(\bigotimes(\mathcal{S}_{\hat{s}}(k_j), \mathcal{S}_{c_i}(k_j))\right) \in \mathbb{R}^{1 \times h}$$

We pool the relation representations from all the KB facts into a single prediction by averaging, i.e. $\mathcal{R}(\hat{s}, c_i) = \text{avg}_j \; \mathcal{R}_j(\hat{s}, c_i)$.

[9]We found this simple composition function performed better than other composition operations.

**Relation Prediction Score.** We first identify the potential relations that can be composed with the fact, given the question, e.g., in Figure 1, we can compose the fact with (steel spoon; *made of*; metal) relation but not (metal; *made of*; ions). We compose an aggregate representation of the question and fact encoding to capture this information:

$$\mathcal{D}(q_s, f) = \bigotimes(\max_{q_m} \mathcal{E}_{q_s}, \max_{f_m} \mathcal{E}_f) \in \mathbb{R}^{1 \times 2h}$$

We finally score the answer choice based on this representation and the relation representation:

$$\text{score}_r(c_i) = \text{FF}\left([\mathcal{D}(q_s, f); \mathcal{R}(\hat{s}, c_i)]\right)$$

The final score for each answer choice is computed by summing the fact relevance and relation prediction based scores i.e. $\text{score}(c_i) = \text{score}_f(c_i) + \text{score}_r(c_i)$. The final architecture of our QA model is shown in Figure 6.

### 4.4 Model Training

We use cross-entropy loss between the predicted answer scores $\hat{c}$ and the gold answer choice $\bar{c}$. Since we also have labels on the true relations between the gold span and the correct answer choice, we introduce an auxiliary loss to ensure the predicted relation $\mathcal{R}$ corresponds to the true relation between $s$ and $c_i$. We use a single-layer feedforward network to project $\mathcal{R}(s, c_i)$ into a vector $\hat{r}_i \in \mathbb{R}^{1 \times l}$ where $l$ is the number of relations. Since multiple relations can be valid, we create an n-hot vector representation $\bar{r} \in \mathbb{R}^{1 \times l}$ where $\bar{r}[k] = 1$ if $r_k$ is a valid relation.

We use binary cross-entropy loss between the $\hat{r}_i$ and $r$ for the correct answer choice. For the incorrect answer choice, we do not know if any of the unselected relations(i.e. where $r[k] = 0$) hold. But we do know that the relations selected by Turkers for the correct answer choice should not hold for the incorrect answer choice. To capture this, we compute the binary cross entropy loss between $\hat{r}_i$ and $1 - r$ for the incorrect answer choices but ignore the unselected relations.

Finally, the loss for each example, assuming $c_i$ is the correct answer, is given as loss $= ce(\hat{c}, \bar{c}) + \lambda \cdot \big(bce(\hat{r}_i, \bar{r}) + \sum_{j \neq i} mbce(\hat{r}_j, 1 - \bar{r}, \bar{r})\big)$, where $ce$ is cross-entropy loss, $bce$ is binary cross-entropy loss, and $mbce$ is masked binary cross entropy loss, where unselected relations are masked.

We further augment the training data with questions in the OBQA-Short dataset using the predicted spans and ignoring the relation loss. Also,

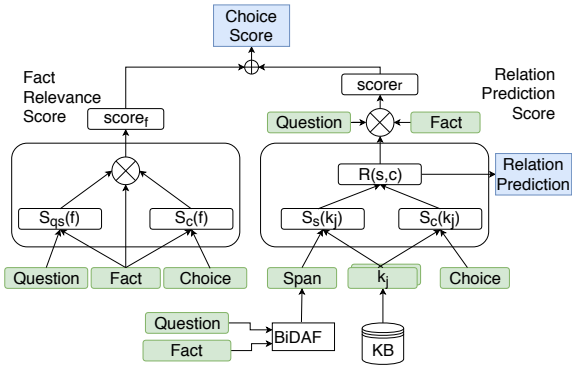


Figure 6: Overall architecture of the KGG question-answering model for each answer choice. The green nodes are the input to the model and the blue nodes are the model outputs that the losses are computed against. The model uses BiDAF to predict the key spans and retrieves KB facts based on the span and the input choice.

we assume the labelled core fact in the OpenBookQA dataset provides the partial knowledge needed to answer these questions.

Implementation details and parameter settings are deferred to Appendix B. A sample visualization of the attentions and knowledge used in the model are provided in Figure 10 in the Appendix.

## 5 Experimental Results

We present results of our proposed model, GapQA, on two question sets: (a) those with short answers,[10] OBQA-Short (290 test questions), and (b) the complete set, OBQA-Full (500 test questions). As we mentioned before, OBQA-Short subset is likely to have Fact-to-Answer gaps that can be targeted by our approach and we therefore expect larger and more meaningful gains on this subset.

### 5.1 Key Span Identification

We begin by evaluating three training strategies for the key span identification model, using the annotated spans in KGD for training. As seen in Table 3, the BiDAF model trained on the SQuAD dataset (Rajpurkar et al., 2016) performs poorly on our task, likely due to the different question style in OpenBookQA. While training on KGD (from scratch) substantially improves accuracy, we observe that using KGD to fine-tune BiDAF pretrained on SQuAD results in the best F1 (78.55) and EM (63.99) scores on the Dev set. All subsequent experiments use this fine-tuned model.

[10] Answers with at most two non-stopword tokens.

| Training Data | Dev F1 | Dev EM |
|---|---|---|
| SQuAD | 54.67 | 41.40 |
| KGD | 72.99 | 58.60 |
| SQuAD + KGD (tuning) | **78.55** | **63.69** |

Table 3: BiDAF model performance on the span prediction task, under different choices of training data

### 5.2 OpenBookQA Results

We compare with three previous state-of-the-art **models** reported by Mihaylov et al. (2018). Two of these are Knowledge-free models (also referred to as No Context Baselines (Chen and Durrett, 2019)): (a) Question-to-Choice (**Q2Choice**) computes attention between the question and the answer choice, (b) **ESIM + ELMo**, uses ESIM (Chen et al., 2017) with ELMo (Peters et al., 2018) embeddings to compute question-choice entailment. The third is Knowledge Enhanced Reader (**KER**), which uses the core fact (f) and knowledge retrieved from ConceptNet to compute cross-attentions between the question, knowledge, and answer choices.

For **knowledge**, we consider four sources: (1) **ConceptNet (CN)**, the English subset of ConceptNet v5.6.0 tuples;[11] (2) **WordNet**, the WordNet subset of ConceptNet used by Mihaylov et al. (2018); (3) **OMCS**, the Open Mind Common Sense subset of ConceptNet used by Mihaylov et al. (2018); and (4) **ARC**, with 14M science-relevant sentences from the AI2 Reasoning Challenge dataset (Clark et al., 2018).

Following OpenBookQA, we train each model five times using different random seeds, and report the average score and standard deviation (without Bessel's correction) on the test set. For simplicity and consistency with prior work, we report one std. dev. from the mean using the $\mu \pm \sigma$ notation.

We train our model on the combined KGD and OBQA-Short question set with full supervision on examples in KGD and only QA supervision (with predicted spans) on questions in OBQA-Short. We train the baseline approaches on the entire question set as they have worse accuracies on both the sets when trained on the OBQA-Short subset. We do not use our annotations for any of the test evaluations. We use the core fact provided by the original dataset and use the predicted spans from the fine-tuned BiDAF model.

We present the test accuracies on the two ques-

[11] https://github.com/commonsense/conceptnet5/wiki

| Model | OBQA-Short | OBQA-Full |
|---|---|---|
| Q2Choice | 47.10 ± 1.5 | 49.64 ± 1.3 |
| ESIM + ELMo | 45.93 ± 2.6 | 49.96 ± 2.5 |
| KER (only f) | 57.93 ± 1.4 | 55.80 ± 1.8 |
| KER (f + WordNet) | 54.83 ± 2.5 | 55.84 ± 1.7 |
| KER (f + OMCS) | 49.65 ± 2.0 | 52.50 ± 0.8 |
| GapQA (f + KB) [Ours] | **64.41 ± 1.8*** | **59.40 ± 1.3*** |

Table 4: Test accuracy on the the OBQA-Short subset and OBQA-Full dataset assuming core fact is given. $*$ denotes the results are statistically significantly better than all the baselines (p≤0.05, based on Wilson score intervals (Wilson, 1927)).

| Knowledge Source | Model | OBQA-Short |
|---|---|---|
| f + WordNet | KER | 54.83 ± 2.5 |
| | GapQA | **60.69 ± 1.1*** |
| f + OMCS | KER | 49.65 ± 2.0 |
| | GapQA | **60.90 ± 2.4*** |
| f + CN + ARC | GapQA | **64.41 ± 1.8** |

Table 5: Test accuracy on the OBQA-Short subset with different sources of knowledge. $*$ denotes the results are statistically significantly better than the corresponding KER result (p≤0.05, based on Wilson score intervals (Wilson, 1927)).

| Model | OBQA-Short | Δ |
|---|---|---|
| GapQA | **64.41 ± 1.8** | — |
| No Annotations | 58.90 ± 1.9 | 5.51 |
| Heuristic Span Anns. | 61.38 ± 1.5 | 3.03 |
| No Relation Score | 60.48 ± 1.1 | 3.93 |
| No Spans (Model) | 62.14 ± 2.1 | 2.27 |
| No Spans (IR) | 61.79 ± 1.0 | 2.62 |

Table 6: Average accuracy of various ablations, showing that each component of GapQA has an important role. No Annotations = Ignore Span & Relation Annotations, Heuristic Span Anns = Heuristically predict span annotations (no human annotations), No Relation Score = Ignore the relation-based score ($score_r$), No Spans (Model) = Ignore the span (use entire fact) to compute span-weighted representations, No Spans (IR) = Ignore the span (use entire fact) for retrieval.

tion sets in Table 4. On the targeted OBQA-Short subset, our proposed GapQA improves statistically significantly over the partial knowledge baselines by 6.5% to 14.4%. Even though the full OpenBookQA dataset contains a wider variety of questions not targeted by GapQA, we still see an improvement of 3+% relative to prior approaches.

It is worth noting that recent large-scale language models (LMs) (Devlin et al., 2019; Radford et al., 2018) have now been applied on this task, leading to improved state-of-the-art results (Sun et al., 2018; Banerjee et al., 2019; Pan et al., 2019). However, our knowledge-gap guided approach to QA is orthogonal to the underlying model. Combining these new LMs with our approach is left to future work.

**Effect of input knowledge.** Since the baseline models use different knowledge sources as input, we evaluate the performance of our model using the same knowledge as the baselines.[12] Even when our model is given the same knowledge, we see an improvement by 5.9% and 11.3% given only WordNet and OMCS knowledge respectively. This shows that we can use the available knowledge, even if limited, more effectively than previous methods. When provided with the full ConceptNet knowledge and large-scale text corpora, our model is able to exploit this additional knowledge and improve further by 4%.

### 5.3 Ablations

We next evaluate key aspects of our model in an ablation study, with average accuracies in Table 6.

**No Annotations (No Anns)**: We ignore all collected annotations (span, relation, and fact) for training the model. We use the BiDAF(SQuAD) model for span prediction, and only the question answering loss for the QA model trained on the OBQA-Short subset.[13] Due to the noisy spans produced by the out-of-domain BiDAF model, this model performs worse that the full GapQA model by 5.5% (comparable performance to the KER models). This shows that our model does utilize the human annotations to improve on this task.

**Heuristic Span Annotations**: We next ask whether some of the above loss in accuracy can be recovered by heuristically producing the spans for training—a cost-effective alternative to human annotations. We find the longest subsequence of tokens (ignoring stop words) in $f$ that is not mentioned in $q$ and assume this span (including the intermediate stop words) to be the key term. To prevent noisy key terms, we only consider a subset of questions where 60% of the non-stopword stemmed tokens in $f$ are covered by $q$. We fine-tune the BiDAF(SQuAD) model on this subset and then use it to predict the spans on the full set.[14]

[12] The baselines do not scale to large scale corpora and so can not be evaluated against our knowledge sources.

[13] Model still predicts the latent relation representation.

[14] We use the questions from the KGD + OBQA-Short set. Note we are only evaluating the impact of heuristic spans compared to human-authored spans, but assume that we have

| Question | Fact | Predicted Answer | Reason |
|---|---|---|---|
| What vehicle would you use to travel on the majority of the surface of the planet on which we live? (A) Bike (B) **Boat** (C) Train (D) Car | oceans cover 70% of the surface of the earth | Bike | Predicted the **wrong span** "70%". |
| What contains moons? (A) ships (B)**space mass** (C) people (D) plants | the solar system contains the moon | ships | **Scores the relation for the incorrect answer higher** because of the facts connecting "systems" and "ships". |
| Cocoon creation occurs (A) **after the caterpillar stage** (B) after the chrysalis stage (C) after the eggs are laid (D) after the cocoon emerging stage | the cocoons being created occurs during the pupa stage in a life cycle | after the chrysalis stage | Does not model the **complex relation** (temporal ordering) between the key span: "pupa stage" and "caterpillar stage". Instead it predicts "chrysalis" due to the synonymy with "pupa". |

Table 7: Sample errors made by the GapQA on questions from the OBQA-Short dataset. The correct answers are marked in **bold** within the question.

We train GapQA model on this dataset without any relation labels (and associated loss). This simple heuristic leads to a 3% drop compared to human annotations, but still out-performs previous approaches on this dataset, showing the value of the gap-based QA approach.

**No Relation Score**: We ignore the entire relation-based score ($score_r$) in the model and only rely on the fact-relevance score. The drop in score by 3.9% shows that the fact alone is not sufficient to answer the question using our model.

**No Spans (Model)**: We ignore the spans in the model, i.e., we use the entire fact to compute the span-based representation $\mathcal{S}_{\hat{s}}(k_j)$. In effect, the model is predicting the gap between the entire fact and answer choice.[15] We see a drop of $\sim$2%, showing the value of spans for gap prediction.

**No Spans (IR)**: Ignoring the span for retrieval, the knowledge is retrieved based on the entire fact (full GapQA model is used). The drop in accuracy by 2.6% shows the value of targeted knowledge-gap based retrieval.

### 5.4 Error Analysis

We further analyzed the performance of GapQA on 40 incorrectly answered questions from the dev set in the OBQA-Short dataset. Table 7 shows a few error examples. There were three main classes of errors:

**Incorrect predicted spans** (25%) often due to complex language in the fact or the Question-to-Fact gap needed to accurately identify the span.

**Incorrect relation scores** (55%) due to distracting facts for the incorrect answer or not finding relevant facts for the correct answer, leading to an incorrect answer scoring higher.

**Out-of-scope gap relations** (20%) where the knowledge gap relations are not handled by our model such as temporal relations or negations (e.g., is *not* made of).

Future work in expanding the dataset, incorporating additional relations, and better retrieval could mitigate these errors.

## 6 Conclusion

We focus on the task of question answering under partial knowledge: a novel task that lies in-between open-domain QA and reading comprehension. We identify classes of knowledge gaps when reasoning under partial knowledge and collect a dataset targeting one common class of knowledge gaps. We demonstrate that identifying the knowledge gap first and then reasoning by filling this gap outperforms previous approaches on the OpenBookQA task, with and even without additional missing fact annotation. This work opens up the possibility of focusing on other kinds of knowledge gaps and extending this approach to other datasets and tasks (e.g., span prediction).

## Acknowledgements

We thank Amazon Mechanical Turk workers for their help with annotation, and the reviewers for their invaluable feedback. Computations on beaker.org were supported in part by credits from Google Cloud.

good quality partial context as provided in KGD.

[15] Retrieval is still based on the span and we ignore the relation prediction loss.

## References


Gabor Angeli, Neha Nayak, and Christopher D. Manning. 2016. Combining natural logic and shallow reasoning for question answering. In *Association for Computational Linguistics (ACL)*.

Pratyay Banerjee, Kuntal Kumar Pal, Arindam Mitra, and Chitta Baral. 2019. Careful selection of knowledge to solve open book question answering. In *ACL*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *EMNLP*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. In *NIPS*.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2019. Question answering by reasoning across documents with graph convolutional networks. In *NAACL*.

Jifan Chen and Greg Durrett. 2019. Understanding dataset design choices for multi-hop reasoning. In *NAACL-HLT*.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for natural language inference. In *ACL*, pages 1657–1668.

Peter Clark, Niranjan Balasubramanian, Sumithra Bhakthavatsalam, Kevin Humphreys, Jesse Kinkead, Ashish Sabharwal, and Oyvind Tafjord. 2014. Automatic construction of inference-supporting knowledge bases. In *AKBC Workshop*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *CoRR*, abs/1803.05457.

Peter Clark, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter D Turney, and Daniel Khashabi. 2016. Combining retrieval, statistics, and inference to answer elementary science questions. In *AAAI*, pages 2580–2586.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Bhuwan Dhingra, Qiao Jin, Zhilin Yang, William Cohen, and Ruslan Salakhutdinov. 2018. Neural models for reasoning over multiple mentions using coreference. In *NAACL*.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *ICML*.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. AllenNLP: A deep semantic natural language processing platform. *CoRR*, abs/1803.07640.

Diana Inkpen, Xiao-Dan Zhu, Zhen-Hua Ling, Qian Chen, and Si Wei. 2018. Neural natural language inference models enhanced with external knowledge. In *ACL*.

Peter Jansen, Niranjan Balasubramanian, Mihai Surdeanu, and Peter Clark. 2016. What's in an explanation? Characterizing knowledge and inference requirements for elementary science exams. In *COLING*.

Peter Jansen, Rebecca Sharp, Mihai Surdeanu, and Peter Clark. 2017. Framing qa as building and ranking intersentence answer justifications. *Computational Linguistics*, 43(2):407–449.

Peter A. Jansen, Elizabeth Wainwright, Steven Marmorstein, and Clayton T. Morrison. 2018. WorldTree: A corpus of explanation graphs for elementary science questions supporting multi-hop inference. In *LREC*.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL*.

Rudolf Kadlec, Martin Schmid, Ondřej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. In *ACL*.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *NAACL*.

Daniel Khashabi, Tushar Khot, Ashish Sabharwal, Peter Clark, Oren Etzioni, and Dan Roth. 2016. Question answering via integer programming over semi-structured knowledge. In *Proceedings of IJCAI*.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *EMNLP*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard H. Hovy. 2017. RACE: Large-scale reading comprehension dataset from examinations. In *EMNLP*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? A new dataset for open book question answering. In *EMNLP*.

Todor Mihaylov and Anette Frank. 2018. Knowledgeable Reader: Enhancing Cloze-Style Reading Comprehension with External Commonsense Knowledge. In *ACL*, pages 821–832.

Xiaoman Pan, Kai Sun, Dian Yu, Heng Ji, and Dong Yu. 2019. Improving question answering with external knowledge. *CoRR*, abs/1902.00993.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *ACL*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS-W*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *EMNLP*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL*.

Michael Petrochuk and Luke S. Zettlemoyer. 2018. Simplequestions nearly solved: A new upperbound and baseline approach. In *EMNLP*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *[Online]*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *ACL*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *EMNLP*.

Matthew Richardson, Christopher J.C. Burges, and Erin Renshaw. 2013. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *EMNLP*, pages 193–203.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *ICLR*.

Linfeng Song, Zhiguo Wang, Mo Yu, Yue Zhang, Radu Florian, and Daniel Gildea. 2018. Exploring graph-structured passage representation for multi-hop reading comprehension with graph neural networks. *CoRR*, abs/1809.02040.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. ConceptNet 5.5: An open multilingual graph of general knowledge. In *AAAI*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958.

Kai Sun, Dian Yu, Dong Yu, and Claire Cardie. 2018. Improving machine reading comprehension with general reading strategies. *CoRR*, abs/1810.13441.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. NewsQA: A machine comprehension dataset. In *Workshop on Representation Learning for NLP*.

Dirk Weissenborn, Tomáš Kočiskỳ, and Chris Dyer. 2017. Dynamic integration of background knowledge in neural nlu systems. *arXiv preprint arXiv:1706.02596*.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *TACL*, 6:287–302.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2015. Towards AI-complete question answering: A set of prerequisite toy tasks. *CoRR*, abs/1502.05698.

Edwin B. Wilson. 1927. Probable inference, the law of succession, and statistical inference. *JASA*, 22(158):209–212.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*.

## A Annotating Gaps: Data Collection

We first identify relevant facts for questions and then collect annotations for fact-answer gap, given the relevant fact. However, straightforward approaches to annotate all questions led to noisy labels. To improve annotation quality, we identified question subsets most suitable for this task and split the fact-answer gap annotation into two steps.

**Fact Relevance.** The OpenBookQA dataset provides the core science fact used to create the question. However, in 20% of the cases, while the core science fact inspired the question, it is not needed to answer the question (Mihaylov et al., 2018). We also noticed that often multiple facts from the open book can be relevant for a question. So we first create an annotation task to identify the relevant facts from a set of retrieved facts. Also to ensure that there is a gap between the fact and the correct answer, we select facts that have no word overlap with the correct choice or have overlap with multiple answer choices. This ensures that the fact alone can not be trivially used to answer the question.

We ask Turkers to annotate these retrieved facts as (1) are they *relevant* to the question and (2) if relevant, do they point to a *unique* answer. We introduced the second category after noticing that some generic facts can be relevant but not point to a specific answer making identifying the knowledge gap impossible. E.g. The fact: "evaporation is a stage in the water cycle process" only eliminates one answer option from "The only stage of the water cycle process that is nonexistent is (A) evaporation (B) evaluation (C) precipitation (D) condensation". For each question, we selected facts that were marked as relevant and unique by at least two out of three turkers.

**Knowledge Gap.** In the second round of data collection, we asked Turkers to write the facts connecting the relevant fact to the correct answer choice. We restricted this task to Masters level Turkers with 95% approval rating and 5000 approved hits. However, we noticed that crowdsource workers would often re-state part of the knowledge mentioned in the original fact or directly connect the question to the answer. This issue was also mentioned by the authors of OpenBookQA who also noticed that the additional facts were "*noisy (incomplete, over-complete, or only distantly related)*" (Mihaylov et al., 2018). E.g. for the question: "In the desert, a hawk may enjoy an occasional (A) coyote (B) reptile (C) bat (D) scorpion" and core fact: "hawks eat lizards", one of the turk-authored additional fact: "Hawks hunt reptiles which live in the desert" is sufficient to answer the question on its own.

We also noticed that questions with long answer choices often have multiple fact-answer gaps leading to complex annotations, e.g. "tracking time" *helps with* "measuring how many marshmallows I can eat in 10 minutes". Collecting knowledge gaps for such questions and common-sense knowledge to capture these gaps are interesting directions of future research. We instead focus on questions where the answer choices have at most two non-stopword tokens. We refer to this subset of questions in OpenBookQA as *OBQA-Short*, which still forms more than 50% of the OpenBookQA set. This subset also forms the target question set of our approach.

Further to simplify this task, we broke the task of identifying the required knowledge into two steps (shown in Figure 7 in Appendix): (1) identify key terms in the core fact that could answer the question, and (2) identify the relationship between these terms and the correct answer choice. For key terms, we asked the Turkers to select spans from the core fact itself, to the extent possible. For the relation identification, we provided a list of relations and asked them to select all the relations that hold between the key term and the correct choice but do not hold for the incorrect answer choices. Based on our analysis, we picked nine most common relations: {causes, definedAs, enables, isa, located in, made of, part of, provides, synonym of} and their inverses (except synonymy).[16] If none of these relations were valid, they were allowed to enter the relation in a text box.

We note that the goal of this effort was to collect supervision for a subset of questions to guide the model and show the value of minimal annotation on this task. We believe our approach can be useful to collect annotations on other question sets as well, or can be used to create a challenge dataset for this sub-task. Moreover, the process of collecting this data revealed potential issues with collecting annotations for knowledge gaps and also inspired the design of our two-step QA model.

[16] These relations were also found to be important by prior approaches (Clark et al., 2014; Khashabi et al., 2016; Jansen et al., 2016, 2018) in the science domain.

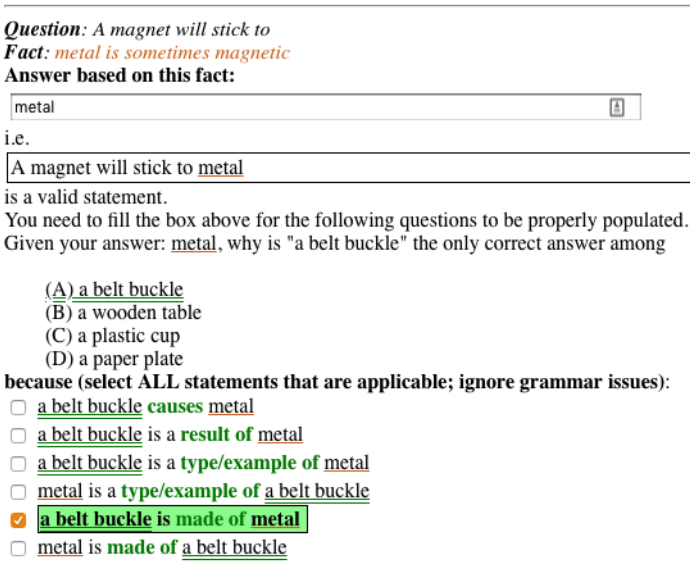


Figure 7: Interface provided to Turkers to annotate the missing fact. Entering the answer span from the fact, metal, in this example, automatically populates the interface with appropriate statements. The valid statements are selected by Turkers and capture the knowledge gap.

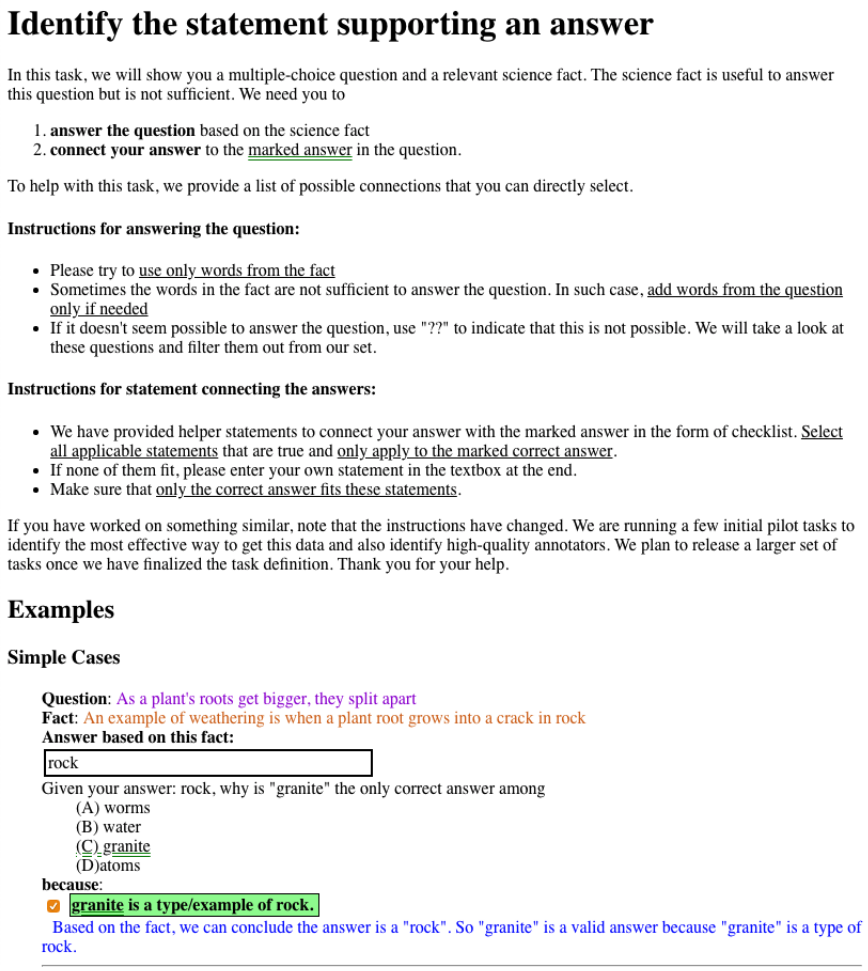


Figure 8: Basic Instructions for the task

### Hard Cases

**Writing your own statement**: If none of the provided statements apply, write your own statment.

**Question**: What happens when a hemisphere is tilted away from the sun?
**Fact**: winter is when a hemisphere is tilted away from the sun.
**Answer based on this fact:**
winter
Given your answer, why is "cools" the only correct answer ?
(A) cools
(B) nothing
(C) heats
(D) warms
**because**:
If none of the above apply, write your own statement here using cools and winter
cools happens during winter ⟵
Since none of the pre-defined list of helper phrases fit, write your own.

---

**Writing your own answer**: If you can't find the right answer as a part of the fact, write your own answer.

**Question**: An animal living in an environment lacking in food resources
**Fact**: an animal requires enough nutrients to maintain good health
**Answer based on this fact:**
not maintain good health ⟵
Since the answer is the opposite of the "maintain good health" in the fact, we need to add a "not" to the answer.
Given your answer, why is "will be in poor shape" the only correct answer ?
(A) will be in poor shape
(B) will be thriving and lively
(C) will be switching to a new diet
(D) will hibernate until more food comes along
**because**: ☑ **will be in poor shape is synonymous with will maintain bad health.**

---

**Adding words from the question**: Sometimes words from the question need to be added to the statement to ensure that only the correct answer fits the statement.

**Question**: Which would be the result of the breeding of two wolves?
**Fact**: reproduction produces offspring
**Answer based on this fact:**
offspring
Given your answer, why is "wolf pups" the only correct answer ?
(A) kittens
(B) wolf pups
(C) fox pups
(D) dog pups
**because**:
If none of the above apply, write your own statement here using wolf pups and offspring
wolf pups are offsprings of wolves. ⟵
Since all the answers are offsprings, we additionally specify that the correct answer should be offspring of wolves.

Figure 9: Instructions for complex examples

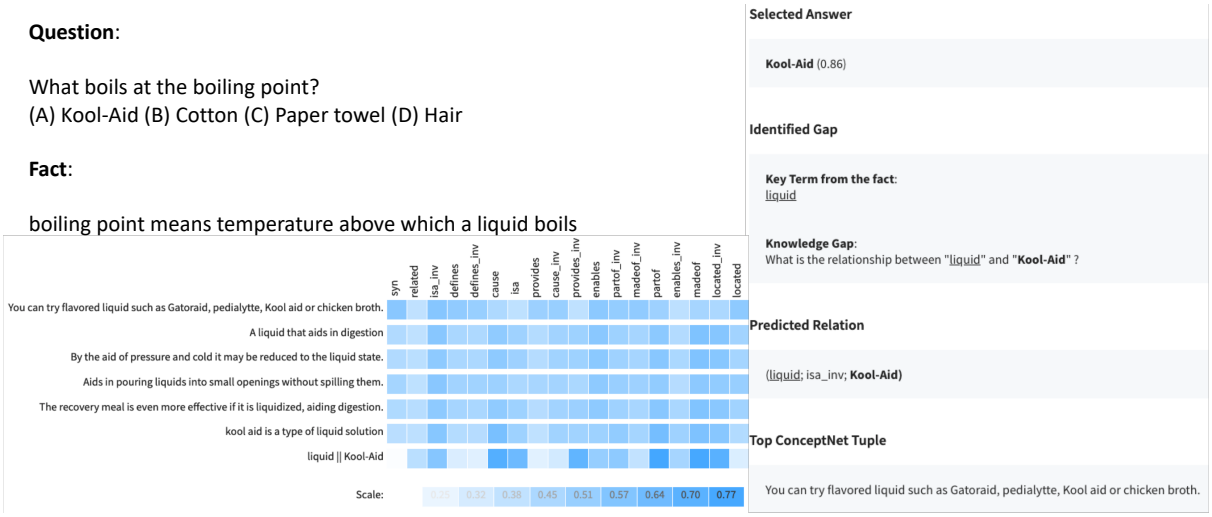


Figure 10: Visualization of the models behavior with the predicted span, top predicted relation, and the top fact used by model. The heat map shows the confidence of the model for all the relations for each input sentence (first five) and ConceptNet sentencized tuple (last but one) and the back-off tuple (last one) to capture the knowledge in the embeddings.

## B Implementation Details

We implement all our models in Pytorch (Paszke et al., 2017) using the AllenNLP (Gardner et al., 2017) toolkit. We also used the AllenNLP implementation of the BiDAF model for span prediction. We use 300D 840B Glove (Pennington et al., 2014) embeddings and use 200 dimensional hidden representations for the BiLSTM shared between all inputs (each direction uses 100 dimensional hidden vectors). We use 100 dimensional representations for the relation prediction, $\mathcal{R}_j$. Each feedforward network, FF is a 2-layer network with relu activation, 0.5 dropout (Srivastava et al., 2014), 200 hidden dimensions on the first layer and no dropout on the output layer with linear activation. We use a variational dropout (Gal and Ghahramani, 2016) of 0.2 in all the BiLSTMs. The relation prediction loss is scaled by $\lambda = 1$. We used the Adam (Kingma and Ba, 2015) optimization with initial $lr = 0.001$ and a learning rate scheduler that halves the learning rate after 5 epochs of no change in QA accuracy. We tuned the hyper-parameters and performed early stopping based on question answering accuracy on the validation set. Specifically, we considered {50, 100, 200} dimensional representations, $\lambda \in \{0.1, 1, 10\}$, retrieving {10, 20} knowledge tuples and {[x - y; x*y], [x, y]} combination functions for $\bigotimes$ during the development of the model. The baseline models were developed for this dataset using hyper-parameter tuning; we do not perform any additional tuning. Our model code and pre-trained models are available at https://github.com/allenai/missing-fact.

## C ConceptNet sentences

Given a tuple $t = (s, v, o)$, the sentence form is generated as "$s$ is split$(v)$ $o$" where split$(v)$ splits the ConceptNet relation $v$ into a phrase based on its camel-case notation. For example, (belt buckle, /r/MadeOf, metal) would be converted into "belt buckle is made of metal".

## D Text retrieval

For each span $\hat{s}$ and answer choice $c_i$, we query an ElasticSearch [17] index on the input text corpus with the "$\hat{s}$ + $c_i$" as the query. We also require the matched sentence must contain both the span and the answer choice. We filter long sentences (>300 characters), sentences with negation and noisy sentences[18] from the retrieved sentences.

[17] https://www.elastic.co/

[18] Sentences are considered clean if they contain alphanumeric characters with standard punctuation, start with an alphabet or a number, are single sentence and only uses hyphens in hyphenated word pairs

# KagNet: Knowledge-Aware Graph Networks for Commonsense Reasoning

**Bill Yuchen Lin**[†] **Xinyue Chen**[‡] **Jamin Chen**[†] **Xiang Ren**[†]
{yuchen.lin,jaminche,xiangren}@usc.edu, kiwisher@sjtu.edu.cn

[†]Computer Science Department, University of Southern California
[‡]Computer Science Department, Shanghai Jiao Tong University

## Abstract

Commonsense reasoning aims to empower machines with the human ability to make presumptions about ordinary situations in our daily life. In this paper, we propose a textual inference framework for answering commonsense questions, which effectively utilizes external, structured commonsense knowledge graphs to perform *explainable* inferences. The framework first grounds a question-answer pair from the semantic space to the knowledge-based symbolic space as a schema graph, a related sub-graph of external knowledge graphs. It represents schema graphs with a novel knowledge-aware graph network module named KagNet, and finally scores answers with graph representations. Our model is based on graph convolutional networks and LSTMs, with a hierarchical path-based attention mechanism. The intermediate attention scores make it transparent and interpretable, which thus produce trustworthy inferences. Using ConceptNet as the only external resource for Bert-based models, we achieved state-of-the-art performance on the CommonsenseQA, a large-scale dataset for commonsense reasoning. We open-source our code[1] to the community for future research in knowledge-aware commonsense reasoning.

## 1 Introduction

Human beings are rational and a major component of rationality is the ability to reason. Reasoning is the process of combining facts and beliefs to make new decisions (Johnson-Laird, 1980), as well as the ability to manipulate knowledge to draw inferences (Hudson and Manning, 2018). Commonsense reasoning utilizes the basic knowledge that reflects our natural understanding of the world and human behaviors, which is common to all humans.

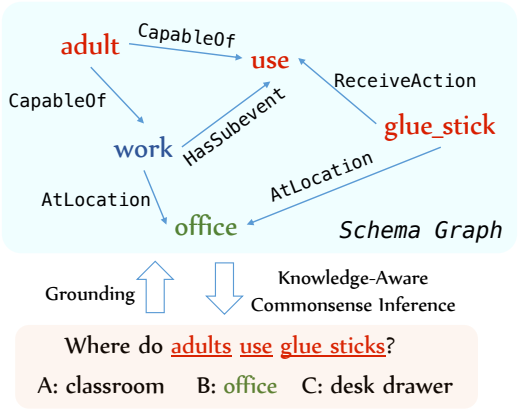


Figure 1: An example of using external commonsense knowledge (symbolic space) for inference in natural language commonsense questions (semantic space).

Empowering machines with the ability to perform commonsense reasoning has been seen as the bottleneck of artificial general intelligence (Davis and Marcus, 2015). Recently, there have been a few emerging large-scale datasets for testing machine commonsense with various focuses (Zellers et al., 2018; Sap et al., 2019b; Zellers et al., 2019). In a typical dataset, CommonsenseQA (Talmor et al., 2019), given a question like *"Where do adults use glue sticks?"*, with the answer choices being {classroom(✗), office (✓), desk drawer (✗)}, a commonsense reasoner is expected to differentiate the correct choice from other "distractive" candidates. False choices are usually highly related to the question context, but just less possible in real-world scenarios, making the task even more challenging. This paper aims to tackle the research question of how we can teach machines to make such commonsense inferences, particularly in the question-answering setting.

It has been shown that simply fine-tuning large, pre-trained language models such as Gpt (Radford et al., 2018) and Bert (Devlin et al., 2019)

[1] https://github.com/INK-USC/KagNet



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2829–2839, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

can be a very strong baseline method. However, there still exists a large gap between performance of said baselines and human performance. Reasoning with neural models is also lacking in transparency and interpretability. There is no clear way as to how they manage to answer commonsense questions, thus making their inferences dubious.

Merely relying on pre-training large language models on corpora cannot provide well-defined or reusable structures for explainable commonsense reasoning. We argue that it would be more beneficial to propose reasoners that can exploit commonsense knowledge bases (Speer et al., 2017; Tandon et al., 2017; Sap et al., 2019a). Knowledge-aware models can explicitly incorporate external knowledge as relational inductive biases (Battaglia et al., 2018) to enhance their reasoning capacity, as well as to increase the transparency of model behaviors for more interpretable results. Furthermore, a knowledge-centric approach is extensible through commonsense knowledge acquisition techniques (Li et al., 2016; Xu et al., 2018).

We propose a knowledge-aware reasoning framework for learning to answer commonsense questions, which has two major steps: schema graph grounding (§3) and graph modeling for inference (§4). As shown in Fig. 1, for each pair of question and answer candidate, we retrieve a graph from external knowledge graphs (e.g. `ConceptNet`) in order to capture the relevant knowledge for determining the plausibility of a given answer choice. The graphs are named "schema graphs" inspired by the `schema theory` proposed by Gestalt psychologists (Axelrod, 1973). The grounded schema graphs are usually much more complicated and noisier, unlike the ideal case shown in the figure.

Therefore, we propose a knowledge-aware graph network module to further effectively model schema graphs. Our model `KagNet` is a combination of graph convolutional networks (Kipf and Welling, 2017) and LSTMs, with a hierarchical path-based attention mechanism, which forms a `GCN-LSTM-HPA` architecture for path-based relational graph representation. Experiments show that our framework achieved a new state-of-the-art performance[2] on the `CommonsenseQA` dataset. Our model also works better then other methods with limited supervision, and provides human-readable results via intermediate attention scores.

[2]The highest score on the leaderboard as of the time when we submitted the paper (May 2019).

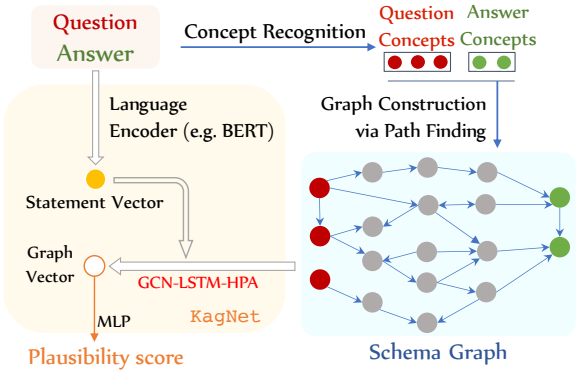


Figure 2: The overall workflow of the proposed framework with knowledge-aware graph network module.

## 2 Overview

In this section, we first formalize the commonsense question answering problem in a knowledge-aware setting, and then introduce the overall workflow of our framework.

### 2.1 Problem statement

Given a commonsense-required natural language question $q$ and a set of $N$ candidate answers $\{a_i\}$, the task is to choose one answer from the set. From a knowledge-aware perspective, we additionally assume that the question $q$ and choices $\{a_i\}$ can be grounded as a **schema graph** (denoted as $g$) extracted from a large external knowledge graph $G$, which is helpful for measuring the plausibility of answer candidates. The knowledge graph $G = (V, E)$ can be defined as a fixed set of **concepts** $V$, and **typed edges** $E$ describing semantic relations between concepts. Therefore, our goal is to effectively ground and model schema graphs to improve the reasoning process.

### 2.2 Reasoning Workflow

As shown in Fig. 2, our framework accepts a pair of question and answer (QA-pair) denoted as $q$ and $a$. It first recognizes the mentioned concepts within them respectively from the concept set $V$ of the knowledge graph. We then algorithmically construct the schema graph $g$ by finding paths between pairs of mentioned concepts (§3).

The grounded schema graph is further encoded with our proposed knowledge-aware graph network module (§4). We first use a model-agnostic language encoder, which can either be trainable or a fixed feature extractor, to represent the QA-pair

as a statement vector. The statement vector serves as an additional input to a GCN-LSTM-HPA architecture for path-based attentive graph modeling to obtain a graph vector. The graph vector is finally fed into a simple multi-layer perceptron to score this QA-pair into a scalar ranging from 0 to 1, representing the plausibility of the inference. The answer candidate with the maximum plausibility score to the same question becomes the final choice of our framework.

## 3 Schema Graph Grounding

The grounding stage is three-fold: recognizing concepts mentioned in text (§3.1), constructing schema graphs by retrieving paths in the knowledge graph (§3.2), and pruning noisy paths (§3.3).

### 3.1 Concept Recognition

We match tokens in questions and answers to sets of mentioned concepts ($\mathcal{C}_q$ and $\mathcal{C}_a$ respectively) from the knowledge graph $G$ (for this paper we chose to use ConceptNet due to its generality).

A naive approach to mentioned concept recognition is to exactly match n-grams in sentences with the surface tokens of concepts in $V$. For example, in the question "*Sitting too close to watch tv can cause what sort of pain?*", the exact matching result $\mathcal{C}_q$ would be {sitting, close, watch_tv, watch, tv, sort, pain, etc.}. We are aware of the fact that such retrieved mentioned concepts are not always perfect (e.g. "sort" is not a semantically related concept, "close" is a polysemous concept). How to efficiently retrieve contextually-related knowledge from noisy knowledge resources is still an open research question by itself (Weissenborn et al., 2017; Khashabi et al., 2017), and thus most prior works choose to stop here (Zhong et al., 2018; Wang et al., 2019b). We enhance this straightforward approach with some rules, such as soft matching with lemmatization and filtering of stop words, and further deal with noise by pruning paths (§3.3) and reducing their importance with attention mechanisms (§4.3).

### 3.2 Schema Graph Construction

**ConceptNet.** Before diving into the construction of schema graphs, we would like to briefly introduce our target knowledge graph ConceptNet. ConceptNet can be seen as a large set of triples of the form $(h, r, t)$, like (ice, HasProperty, cold), where $h$ and $t$ represent head and tail concepts in the concept set $V$ and $r$ is a certain relation type from the pre-defined set $R$. We delete and merge the original 42 relation types into 17 types, in order to increase the density of the knowledge graph[3] for grounding and modeling.

**Sub-graph Matching via Path Finding.** We define a schema graph as a sub-graph $g$ of the whole knowledge graph $G$, which represents the related knowledge for reasoning a given question-answer pair with minimal additional concepts and edges. One may want to find a minimal spanning sub-graph covering all the question and answer concepts, which is actually the NP-complete "Steiner tree problem" in graphs (Garey and Johnson, 1977). Due to the incompleteness and tremendous size of ConceptNet, we find that it is impractical to retrieve a comprehensive but helpful set of knowledge facts this way. Therefore, we propose a straightforward yet effective graph construction algorithm via path finding among mentioned concepts ($\mathcal{C}_q \cup \mathcal{C}_a$).

Specifically, for each question concept $c_i \in \mathcal{C}_q$ and answer concept $c_j \in \mathcal{C}_a$, we can efficiently find paths between them that are shorter than $k$ concepts[4]. Then, we add edges, if any, between the concept pairs within $\mathcal{C}_q$ or $\mathcal{C}_a$.

### 3.3 Path Pruning via KG Embedding

To prune irrelevant paths from potentially noisy schema graphs, we first utilize knowledge graph embedding (KGE) techniques, like TransE (Wang et al., 2014), to pre-train concept embeddings $\mathbf{V}$ and relation type embeddings $\mathbf{R}$, which are also used as initialization for KagNet (§4). In order to measure the quality of a path, we decompose it into a set of triples, the confidence of which can be directly measured by the scoring function of the KGE method (i.e. the confidence of triple classification). Thus, we score a path with the multiplication product of the scores of each triple in the path, and then we empirically set a threshold for pruning (§5.3).

## 4 Knowledge-Aware Graph Network

The core component of our reasoning framework is the knowledge-aware graph network module KagNet. The KagNet first encodes plain structures of schema graphs with graph convolutional networks (§4.1) to accommodate pre-trained con-

[3] The full mapping list is in the appendix.

[4] We set $k = 4$ in experiments to gather three-hop paths.

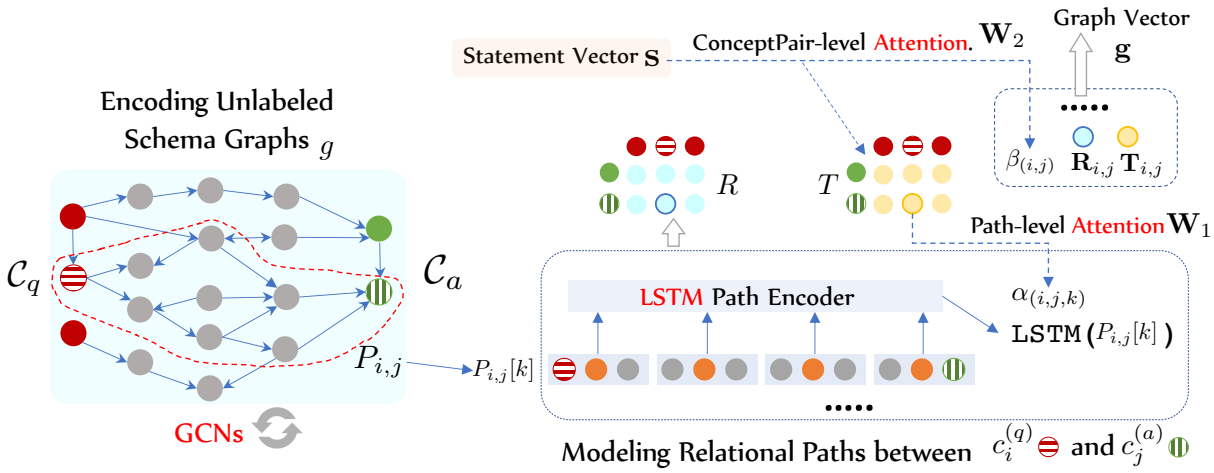


Figure 3: Illustration of the GCN−LSTM−HPA architecture for the proposed KAGNET module.

cept embeddings in their particular context within schema graphs. It then utilizes LSTMs to encode the paths between $\mathcal{C}_q$ and $\mathcal{C}_a$, capturing multi-hop relational information (§4.2). Finally, we apply a hierarchical path-based attention mechanism (§4.3) to complete the GCN−LSTM−HPA architecture, which models relational schema graphs with respect to the paths between question and answer concepts.

### 4.1 Graph Convolutional Networks

Graph convolutional networks (GCNs) encode graph-structured data by updating node vectors via pooling features of their adjacent nodes (Kipf and Welling, 2017). Our intuition for applying GCNs to schema graphs is to 1) contextually refine the concept vectors and 2) capture structural patterns of schema graphs for generalization.

Although we have obtained concept vectors by pre-training (§3.3), the representations of concepts still need to be further accommodated to their specific schema graphs context. Think of polysemous concepts such as "close" (§3.1), which can either be a verb concept like in "close the door" or an adjective concept meaning "a short distance apart". Using GCNs to update the concept vector with their neighbors is thus helpful for disambiguation and contextualized concept embedding. Also, the pattern of schema graph structures provides potentially valuable information for reasoning. For instance, shorter and denser connections between question and answer concepts could mean higher plausibility under specific contexts.

As many works show (Marcheggiani and Titov, 2017; Zhang et al., 2018), relational GCNs (Schlichtkrull et al., 2018) usually over-parameterize the model and cannot effectively utilize multi-hop relational information. We thus apply GCNs on the plain version (unlabeled, non-directional) of schema graphs, ignoring relation types on the edges. Specifically, the vector for concept $c_i \in \mathcal{V}_g$ in the schema graph $g$ is initialized by their pre-trained embeddings at first ($h_i^{(0)} = \mathbf{V}_i$). Then, we update them at the $(l+1)$-th layer by pooling features of their neighboring nodes ($N_i$) and their own at the $l$-th layer with an non-linear activation function $\sigma$:

$$h_i^{(l+1)} = \sigma(W_{self}^{(l)} h_i^{(l)} + \sum_{j \in N_i} \frac{1}{|N_i|} W^{(l)} h_j^{(l)})$$

### 4.2 Relational Path Encoding

In order to capture the relational information in schema graphs, we propose an LSTM-based path encoder on top of the outputs of GCNs. Recall that our graph representation has a special purpose: "*to measure the plausibility of a candidate answer to a given question*". Thus, we propose to represent graphs with respect to the paths between question concepts $\mathcal{C}_q$ and answer concepts $\mathcal{C}_a$.

We denote the $k$-th path between $i$-th question concept $c_i^{(q)} \in \mathcal{C}_q$ and $j$-th answer concept $c_j^{(a)} \in \mathcal{C}_a$ as $P_{i,j}[k]$, which is a sequence of triples:

$$P_{i,j}[k] = [(c_i^{(q)}, r_0, t_0), ..., (t_{n-1}, r_n, c_j^{(a)})]$$

Note that the relations are represented with trainable relation vectors (initialized with pre-trained relation embeddings), and concept vectors are the GCNs' outputs ($h^{(l)}$). Thus, each triple can be represented by the concatenation of the three

corresponding vectors. We employ LSTM networks to encode these paths as sequences of triple vectors, taking the concatenation of the first and the last hidden states:

$$\mathbf{R}_{i,j} = \frac{1}{|P_{i,j}|} \sum_k \texttt{LSTM}(P_{i,j}[k])$$

The above $\mathbf{R}_{i,j}$ can be viewed as the latent relation between the question concept $c_i^{(q)}$ and the answer concept $c_j^{(a)}$, for which we aggregate the representations of all the paths between them in the schema graph. Now we can finalize the vector representation of a schema graph $\mathbf{g}$ by aggregating all vectors in the matrix $\mathbf{R}$ using mean pooling:

$$\mathbf{T}_{i,j} = \texttt{MLP}([\mathbf{s}\ ;\ \mathbf{c_q^{(i)}}\ ;\ \mathbf{c_a^{(j)}}])$$
$$\mathbf{g} = \frac{\sum_{i,j}[\mathbf{R}_{i,j}\ ;\ \mathbf{T}_{i,j}]}{|\mathcal{C}_q| \times |\mathcal{C}_a|}$$

, where $[\cdot\ ;\ \cdot]$ means concatenation of two vectors.

The statement vector $\mathbf{s}$ in the above equation is obtained from a certain language encoder, which can either be a trainable sequence encoder like LSTM or features extracted from pre-trained universal language encoders like GPT/BERT). To encode a question-answer pair with universal language encoders, we simply create a sentence combining the question and the answer with a special token ("`question`+ [sep] + `answer`"), and then use the vector of '[cls]' as suggested by prior works (Talmor et al., 2019)..

We concatenate $\mathbf{R}_{i,j}$ with an additional vector $\mathbf{T}_{i,j}$ before doing average pooling. The $\mathbf{T}_{i,j}$ is inspired from the `Relation Network` (Santoro et al., 2017), which also encodes the latent relational information yet from the context in the statement $s$ instead of the schema graph $g$. Simply put, we want to combine the relational representations of a pair of question/answer concepts from both the schema graph side (symbolic space) and the language side (semantic space).

Finally, the plausibility score of the answer candidate $a$ to the question $q$ can be computed as $\texttt{score}(q, a) = \texttt{sigmoid}(\texttt{MLP}(\mathbf{g}))$.

### 4.3 Hierarchical Attention Mechanism

A natural argument against the above `GCN-LSTM-mean` architecture is that mean pooling over the path vectors does not always make sense, since some paths are more important than others for reasoning. Also, it is usually not the case that all pairs of question and answer concepts equally contribute to the reasoning. Therefore, we propose a hierarchical path-based attention mechanism to selectively aggregate important path vectors and then more important question-answer concept pairs. This core idea is similar to the work of Yang et al. (2016), which proposes a document encoder that has two levels of attention mechanisms applied at the word- and sentence-level. In our case, we have path-level and concept-pair-level attention for learning to contextually model graph representations. We learn a parameter matrix $\mathbf{W}_1$ for path-level attention scores, and the importance of the path $P_{i,j}[k]$ is denoted as $\hat{\alpha}_{(i,j,\cdot)}$.

$$\alpha_{(i,j,k)} = \mathbf{T}_{i,j}\ \mathbf{W}_1\ \texttt{LSTM}(P_{i,j}[k]),$$
$$\hat{\alpha}_{(i,j,\cdot)} = \texttt{SoftMax}(\alpha_{(i,j,\cdot)}),$$
$$\hat{\mathbf{R}}_{i,j} = \sum_k \hat{\alpha}_{(i,j,k)} \cdot \texttt{LSTM}(P_{i,j}[k]).$$

Afterwards, we similarly obtain the attention over concept-pairs.

$$\beta_{(i,j)} = \mathbf{s}\ \mathbf{W}_2\ \mathbf{T}_{i,j}$$
$$\hat{\beta}_{(\cdot,\cdot)} = \texttt{SoftMax}(\beta_{(\cdot,\cdot)})$$
$$\hat{\mathbf{g}} = \sum_{i,j} \hat{\beta}_{(i,j)}[\hat{\mathbf{R}}_{i,j}\ ;\ \mathbf{T}_{i,j}]$$

The whole `GCN-LSTM-HPA` architecture is illustrated in Figure 3. To sum up, we claim that the KAGNET is a graph neural network module with the `GCN-LSTM-HPA` architecture that models relational graphs for relational reasoning under the context of both knowledge symbolic space and language semantic space.

## 5 Experiments

We introduce our setups of the `CommonsenseQA` dataset (Talmor et al., 2019), present the baseline methods, and finally analyze experimental results.

### 5.1 Dataset and Experiment Setup

The `CommonsenseQA` dataset consists of 12,102 (v1.11) natural language questions in total that require human commonsense reasoning ability to answer, where each question has five candidate answers (hard mode). The authors also release an easy version of the dataset by picking two random terms/phrases for sanity check. `CommonsenseQA` is directly gathered from real human annotators and covers a broad range of

| Model | 10(%) of IHtrain | | 50(%) of IHtrain | | 100(%) of IHtrain | |
|---|---|---|---|---|---|---|
| | IHdev-Acc.(%) | IHtest-Acc.(%) | IHdev-Acc.(%) | IHtest-Acc.(%) | IHdev-Acc.(%) | IHtest-Acc.(%) |
| Random guess | 20.0 | 20.0 | 20.0 | 20.0 | 20.0 | 20.0 |
| GPT-FINETUNING | 27.55 | 26.51 | 32.46 | 31.28 | 47.35 | 45.58 |
| GPT-**KAGNET** | 28.13 | **26.98** | 33.72 | **32.33** | 48.95 | **46.79** |
| BERT-BASE-FINETUNING | 30.11 | 29.78 | 38.66 | 36.83 | 53.48 | 53.26 |
| BERT-BASE-**KAGNET** | 31.05 | **30.94** | 40.32 | **39.01** | 55.57 | **56.19** |
| BERT-LARGE-FINETUNING | 35.71 | 32.88 | 55.45 | 49.88 | 60.61 | 55.84 |
| BERT-LARGE-**KAGNET** | 36.82 | **33.91** | 58.73 | **51.13** | 62.35 | **57.16** |
| Human Performance | - | *88.9* | - | *88.9* | - | *88.9* |

Table 1: Comparisons with large pre-trained language model fine-tuning with different amount of training data.

types of commonsense, including spatial, social, causal, physical, temporal, etc. To the best of our knowledge, CommonsenseQA may be the most suitable choice for us to evaluate supervised learning models for question answering.

For the comparisons with the reported results in the CommonsenseQA's paper and leaderboard, we use the official split (9,741/1,221/1,140) named (OFtrain/OFdev/OFtest). Note that the performance on OFtest can only be tested by submitting predictions to the organizers. To efficiently test other baseline methods and ablation studies, we choose to use randomly selected 1,241 examples from the training data as our in-house data, forming an (8,500/1,221/1,241) split denoted as (IHtrain/IHdev/IHtest). All experiments are using the random-split setting as the authors suggested, and three or more random states are tested on development sets to pick the best-performing one.

### 5.2 Compared Methods

We consider two different kinds of baseline methods as follows:

• *Knowledge-agnostic Methods.* These methods either use no external resources or only use unstructured textual corpora as additional information, including gathering textual snippets from search engine or large pre-trained language models like BERT-LARGE. QABILINEAR, QACOMPARE, ESIM are three supervised learning models for natural language inference that can be equipped with different word embeddings including GloVe and ELMO. BIDAF++ utilizes Google web snippets as context and is further augmented with a self-attention layer while using ELMO as input features. GPT/BERT-LARGE are fine-tuning methods with an additional linear layer for classification as the authors suggested. They both add a special token '[sep]' to the input and use the hidden state of the '[cls]' as the input to the linear layer. More details about them can be found in the dataset paper (Talmor et al., 2019).

• *Knowledge-aware Methods.* We also *adopt* some recently proposed methods of incorporating knowledge graphs for question answering. KV-MEM (Mihaylov and Frank, 2018) is a method that incorporates retrieved triples from ConceptNet at the word-level, which uses a key-valued memory module to improve the representation of each token individually by learning an attentive aggregation of related triple vectors. CBPT (Zhong et al., 2018) is a plug-in method of assembling the predictions of any models with a straightforward method of utilizing pre-trained concept embeddings from ConceptNet. TEXTGRAPHCAT (Wang et al., 2019c) concatenates the graph-based and text-based representations of the statement and then feed it into a classifier. We create sentence template for generating sentences and then feed retrieved triples as additional text inputs as a baseline method TRIPLESTRING. Rajani et al. (2019) propose to collect human explanations for commonsense reasoning from annotators as additional knowledge (COS-E), and then train a language model based on such human annotations for improving the model performance.

### 5.3 Implementation Details of KagNet

Our best (tested on OFdev) settings of KAGNET have two GCN layers (100 dim, 50dim respectively), and one bidirectional LSTMs (128dim) . We pre-train KGE using TransE (100 dimension) initialized with GloVe embeddings. The statement encoder in use is BERT-LARGE, which works as a pre-trained sentence encoder to obtain fixed features for each pair of question and answer candidate. The paths are pruned with path-score threshold set to *0.15*, keeping *67.21%* of the original

| Model | OFdev-Acc.(%) | OFtest-Acc.(%) |
|---|---|---|
| Random guess | 20.0 | 20.0 |
| BIDAF++ | - | 32.0 |
| QACOMPARE+GLOVE | - | 25.7 |
| QABLINEAR+GLOVE | - | 31.5 |
| ESIM+ELMO | - | 32.8 |
| ESIM+GLOVE | - | 34.1 |
| GPT-FINETUNING | 47.11 | 45.5 |
| BERT-BASE-FINETUNING | 53.57 | 53.0 |
| BERT-LARGE-FINETUNING | 62.34 | 56.7 |
| COS-E (w/ additional annotations) | - | 58.2 |
| **KAGNET** (Ours) | 64.46 | **58.9** |
| Human Performance | - | *88.9* |

Table 2: Comparison with official benchmark baseline methods using the official split on the leaderboard.

| | Easy Mode | | Hard Mode | |
|---|---|---|---|---|
| **Model** | IHdev.(%) | IHtest.(%) | IHdev.(%) | IHtest.(%) |
| Random guess | 33.3 | 33.3 | 20.0 | 20.0 |
| BLSTMS | 80.15 | 78.01 | 34.79 | 32.12 |
| + KV-MN | 81.71 | 79.63 | 35.70 | 33.43 |
| + CSPT | 81.79 | 80.01 | 35.31 | 33.61 |
| + TEXTGRAPHCAT | 82.68 | 81.03 | 34.72 | 33.15 |
| + TRIPLESTRING | 79.11 | 76.02 | 33.19 | 31.02 |
| + **KAGNET** | 83.26 | **82.15** | 36.38 | **34.57** |
| Human Performance | - | *99.5* | - | *88.9* |

Table 3: Comparisons with knowledge-aware baseline methods using the in-house split (both easy and hard mode) on top of BLSTM as the sentence encoder.

paths. We did not conduct pruning on concept pairs with less than three paths. For very few pairs with none path, $\hat{\mathbf{R}}_{(i,j)}$ will be a randomly sampled vector. We learn our KAGNET models with Adam optimizers (Kingma and Ba, 2015). In our experiments, we found that the recall of ConceptNet on commonsense questions and answers is very high (over 98% of QA-pairs have more than one grounded concepts).

### 5.4 Performance Comparisons and Analysis

**Comparison with standard baselines.**
As shown in Table 2, we first use the official split to compare our model with the baseline methods reported on the paper and leaderboard. BERT and GPT-based pre-training methods are much higher than other baseline methods, demonstrating the ability of language models to store commonsense knowledge in an implicit way. This presumption is also investigated by Trinh and Le (2019) and Wang et al. (2019). Our proposed framework achieves an absolute increment of 2.2% in accuracy on the test data, a state-of-the-art performance.

We conduct the experiments with our in-house splits to investigate whether our KAGNET can also work well on other universal language encoders (GPT and BERT-BASE), particularly with different fractions of the dataset (say 10%, 50%, 100% of the training data). Table 1 shows that our KAGNET-based methods using fixed pre-trained language encoders outperform fine-tuning themselves in all settings. Furthermore, we find that the improvements in a small data situation (10%) is relatively limited, and we believe an important future research direction is thus few-shot learning for commonsense reasoning.

**Comparison with knowledge-aware baselines.**
To compare our model with other adopted baseline methods that also incorporate ConceptNet, we set up a bidirectional LSTM networks-based model for our in-house dataset. Then, we add baseline methods and KAGNET onto the BLSTMs to compare their abilities to utilize external knowledge[5]. Table 3 shows the comparisons under both easy mode and hard mode, and our methods outperform all knowledge-aware baseline methods by a large margin in terms of accuracy. Note that we compare our model and the CoS-E in Table 2. Although CoS-E also achieves better result than only fine-tuning BERT by training with human-generated explanations, we argue that our proposed KagNet does not utilize any additional human efforts to provide more supervision.

**Ablation study on model components.**
To better understand the effectiveness of each component of our method, we have done ablation study as shown in Table 4. We find that replacing our GCN-LSTM-HPA architecture with traditional relational GCNs, which uses separate weight matrices for different relation types, results in worse performance, due to its over-parameterization. The attention mechanisms matters almost equally in two levels, and pruning also effectively filters noisy paths.

**Error analysis.**
In the failed cases, there are three kinds of hard problems that KAGNET is still not good at.

- **negative reasoning:** the grounding stage is not sensitive to the negation words, and thus can choose exactly opposite answers.
- **comparative reasoning strategy:** For the

[5]We do LSTM-based setup because it is non-trivial to apply token-level knowledge-aware baseline methods for complicated pre-trained encoders like BERT.

| Model | IHdev.(%) | IHtest.(%) |
|---|---|---|
| KAGNET (STANDARD) | 62.35 | 57.16 |
| : replace GCN-HPA-LSTM w/ R-GCN | 60.01 | 55.08 |
| : w/o GCN | 61.84 | 56.11 |
| : #GCN Layers = 1 | 62.05 | 57.03 |
| : w/o Path-level Attention | 60.12 | 56.05 |
| : w/o QAPair-level Attention | 60.39 | 56.13 |
| : using all paths (w/o pruning) | 59.96 | 55.27 |

Table 4: Ablation study on the KAGNET framework.

questions with more than one highly plausible answers, the commonsense reasoner should benefit from explicitly investigating the difference between different answer candidates, while KAGNET training method is not capable of doing so.

- **subjective reasoning:** Many answers actually depend on the "personality" of the reasoner. For instance, "*Traveling from new place to new place is likely to be what?*" The dataset gives the answer as "exhilarating" instead of "exhausting", which we think is more like a personalized subjective inference instead of common sense.

### 5.5 Case Study on Interpretibility

Our framework enjoys the merit of being more transparent, and thus provides more interpretable inference process. We can understand our model behaviors by analyzing the hierarchical attention scores on the question-answer concept pairs and path between them.

Figure 4 shows an example how we can analyze our KAGNET framework through both pair-level and path-level attention scores. We first select the concept-pairs with highest attention scores and then look at the (one or two) top-ranked paths for each selected pair. We find that paths located in this way are highly related to the inference process and also shows that noisy concepts like 'fountain' will be diminished while modeling.

### 5.6 Model Transferability.

We study the transferability of a model that is trained on CommonsenseQA (CSQA) by directly testing it with another task while fixing its parameters. Recall that we have obtained a BERT-LARGE model and a KAGNET model trained on CSQA. Now we denoted them as CSQA-BL and CSQA-KN to suggest that they are not trainable anymore.

In order to investigate their transferability, we separately test them on SWAG (Zellers et al., 2018)

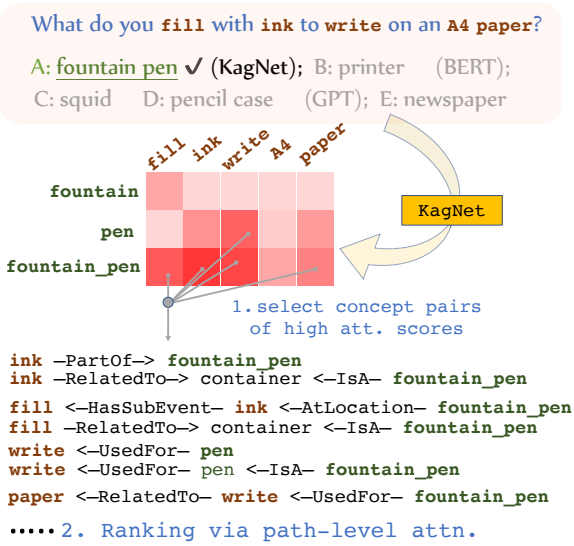


Figure 4: An example of interpreting model behaviors by hierarchical attention scores.

and WSC (Levesque, 2011) datasets. We first test them the 20k validation examples in SWAG. CSQA-BL has an accuracy of 56.53%, while our fixed CSQA-KN model achieves 59.01%. Similarly, we also test both models on the WSC-QA, which is converted from the WSC pronoun resolution to a multi-choice QA task.

The CSQA-BL achieves an accuracy of 51.23%, while our model CSQA-KN scores 53.51%. These two comparisons further support our assumption that KAGNET, as a knowledge-centric model, is more extensible in commonsense reasoning. As we expect for a good knowledge-aware frameworks to behave, our KAGNET indeed enjoys better transferablity than only fine-tuning large language encoders like BERT.

### 5.7 Recent methods on the leaderboard.

We argue that the KAGNET utilizes the ConceptNet as the only external resource and other methods are improving their performance in orthogonal directions: 1) we find that most of the other recent submissions (as of Aug. 2019) with public information on the leaderboard utilize larger additional textual corpora (e.g. top 10 matched sentences in full Wikipedia via information retrieval tools), and fine-tuning on larger pre-trained encoders, such as XLNet (Yang et al., 2019), RoBERTa (Liu et al., 2019). 2) there are also models using multi-task learning to transfer knowledge from other reading comprehension datasets, such as RACE (Lai et al., 2017) and

OpenBookQA (Mihaylov et al., 2018).

An interesting fact is that the best performance on the OFtest set is still achieved the original fine-tuned RoBERTa model, which is pre-trained with copora much larger than BERT. All other RoBERTa-extended methods have negative improvements. We also use statement vectors from RoBERTa as the input vectors for KagNet, and find that the performance on OFdev marginally improves from 77.47% to 77.56%. Based on our above-mentioned failed cases in error analysis, we believe fine-tuning RoBERTa has achieved the limit due to the annotator biases of the dataset and the lack of comparative reasoning strategies.

## 6 Related Work

**Commonsense knowledge and reasoning.** There is a recent surge of novel large-scale datasets for testing machine commonsense with various focuses, such as situation prediction (SWAG) (Zellers et al., 2018), social behavior understanding (Sap et al., 2019a,b), visual scene comprehension (Zellers et al., 2019), and general commonsense reasoning (Talmor et al., 2019), which encourages the study of supervised learning methods for commonsense reasoning. Trinh and Le (2018) find that large language models show promising results in WSC resolution task (Levesque, 2011), but this approach can hardly be applied in a more general question answering setting and also not provide explicit knowledge used in inference. A unique merit of our KagNet method is that it provides grounded explicit knowledge triples and paths with scores, such that users can better understand and put trust in the behaviors and inferences of the model.

**Injecting external knowledge for NLU.** Our work also lies in the general context of using external knowledge to encode sentences or answer questions. Yang and Mitchell (2017) are the among first ones to propose to encode sentences by keeping retrieving related entities from knowledge bases and then merging their embeddings into LSTM networks computations, to achieve a better performance on entity/event extraction tasks. Weissenborn et al. (2017), Mihaylov and Frank (2018), and Annervaz et al. (2018) follow this line of works to incorporate the embeddings of related knowledge triples at the word-level and improve the performance of natural language understanding tasks. In contrast to our work, they do not explicitly impose graph-structured knowledge into models , but limit its potential within transforming word embeddings to concept embeddings.

Some other recent attempts (Zhong et al., 2018; Wang et al., 2019c) to use ConceptNet graph embeddings are adopted and compared in our experiments (§5). Rajani et al. (2019) propose to manually collect more human explanations for correct answers as additional supervision for auxiliary training. KagNet-based framework focuses on injecting external knowledge as an explicit graph structure, and enjoys the relational reasoning capacity over the graphs.

**Relational reasoning.** KagNet can be seen as a knowledge-augmented Relation Network module (RN) (Santoro et al., 2017), which is proposed for the visual question answering task requiring relational reasoning (i.e. questions about the relations between multiple 3D-objects in an image). We view the concepts in the questions and answers as objects and effectively utilize external knowledge graphs to model their relations from both semantic and symbolic spaces (§4.2), while prior methods mainly work on the semantic one.

## 7 Conclusion

We propose a knowledge-aware framework for learning to answer commonsense questions. The framework first constructs schema graphs to represent relevant commonsense knowledge, and then model the graphs with our KagNet module. The module is based on a GCN-LSTM-HPA architecture, which effectively represent graphs for relational reasoning purpose in a transparent, interpretable way, yielding a new state-of-the-art results on a large-scale general dataset for testing machine commonsense. Future directions include better question parsing methods to deal with negation and comparative question answering, as well as incorporating knowledge to visual reasoning.

## Acknowledgments

This work has been supported in part by National Science Foundation SMA 18-29268, DARPA MCS and GAILA, IARPA BETTER, Schmidt Family Foundation, Amazon Faculty Award, Google Research Award, Snapchat Gift and JP Morgan AI Research Award. We would like to thank all the collaborators in the INK research lab for their constructive feedback on the work.

## References


K. M. Annervaz, Somnath Basu Roy Chowdhury, and Ambedkar Dukkipati. 2018. Learning beyond datasets: Knowledge graph augmented neural networks for natural language processing. In *Proc. of NAACL-HLT*.

Robert Axelrod. 1973. Schema theory: An information processing model of perception and cognition. *American political science review*, 67(4):1248–1266.

Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinícius Flores Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Çaglar Gülçehre, Francis Song, Andrew J. Ballard, Justin Gilmer, George E. Dahl, Ashish Vaswani, Kelsey R. Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matthew Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. 2018. Relational inductive biases, deep learning, and graph networks. *CoRR*, abs/1806.01261.

Ernest Davis and Gary Marcus. 2015. Commonsense reasoning and commonsense knowledge in artificial intelligence. *Commun. ACM*, 58(9):92–103.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proc. of NAACL-HLT*.

Michael R Garey and David S. Johnson. 1977. The rectilinear steiner tree problem is np-complete. *SIAM Journal on Applied Mathematics*, 32(4):826–834.

Drew A. Hudson and Christopher D. Manning. 2018. Compositional attention networks for machine reasoning. In *Proc. of ICLR*.

Philip N Johnson-Laird. 1980. Mental models in cognitive science. *Cognitive science*, 4(1):71–115.

Daniel Khashabi, Tushar Khot, Ashutosh Sabharwal, and Dan Roth. 2017. Learning what is essential in questions. In *Proc. of CoNLL*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proc. of ICLR*.

Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of ICLR*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard H. Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. In *Proc. of EMNLP*.

Hector J. Levesque. 2011. The winograd schema challenge. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*.

Xiang Li, Aynaz Taheri, Lifu Tu, and Kevin Gimpel. 2016. Commonsense knowledge base completion. In *Proc. of ACL*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar S. Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke S. Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proc. of EMNLP*.

Todor Mihaylov and Anette Frank. 2018. Knowledgeable reader: Enhancing cloze-style reading comprehension with external commonsense knowledge. In *Proc. of ACL*.

Tzvetan Mihaylov, Peter F. Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proc. of EMNLP*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Explain yourself! leveraging language models for commonsense reasoning. In *Proc. of ACL*.

Adam Santoro, David Raposo, David G. T. Barrett, Mateusz Malinowski, Razvan Pascanu, Peter W. Battaglia, and Timothy P. Lillicrap. 2017. A simple neural network module for relational reasoning. In *Proc. of NIPS*.

Maarten Sap, Ronan LeBras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A Smith, and Yejin Choi. 2019a. Atomic: An atlas of machine commonsense for if-then reasoning. In *Proc. of AAAI*.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019b. Socialiqa: Commonsense reasoning about social interactions. *CoRR*, abs/1904.09728.

Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *Proc. of European Semantic Web Conference*.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proc. of AAAI*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proc. of NAACL-HLT*.

Niket Tandon, Gerard de Melo, and Gerhard Weikum. 2017. Webchild 2.0 : Fine-grained commonsense knowledge distillation. In *Proc. of ACL*.

Trieu H. Trinh and Quoc V. Le. 2018. A simple method for commonsense reasoning. *CoRR*, abs/1806.02847.

Trieu H. Trinh and Quoc V. Le. 2019. Do language models have common sense? *OpenReview*, ICLR submissions.

Cunxiang Wang, Shuailong Liang, Yue Zhang, Xiaonan Li, and Tian Gao. 2019a. Does it make sense? and why? a pilot study for sense making and explanation. In *Proc. of ACL*.

Xiaoyan Wang, Pavan Kapanipathi, Ryan Musa, Mo Yu, Kartik Talamadupula, Ibrahim Abdelaziz, Maria Chang, Achille Fokoue, Bassem Makni, Nicholas Mattei, and Michael Witbrock. 2019b. Improving natural language inference using external knowledge in the science questions domain. In *Proc. of AAAI*.

Xiaoyan Wang, Pavan Kapanipathi, Ryan Musa, Mo Yu, Kartik Talamadupula, Ibrahim Abdelaziz, Maria Chang, Achille Fokoue, Bassem Makni, Nicholas Mattei, and Michael Witbrock. 2019c. Improving natural language inference using external knowledge in the science questions domain.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proc. of AAAI*.

Dirk Weissenborn, Tomáš Kočiskỳ, and Chris Dyer. 2017. Dynamic integration of background knowledge in neural nlu systems. *arXiv preprint arXiv:1706.02596*.

Frank Xu, Bill Yuchen Lin, and Kenny Q. Zhu. 2018. Automatic extraction of commonsense locatednear knowledge. In *Proc. of ACL*.

Bishan Yang and Tom Michael Mitchell. 2017. Leveraging knowledge bases in lstms for improving machine reading. In *Proc. of ACL*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *ArXiv*, abs/1906.08237.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. 2016. Hierarchical attention networks for document classification. In *Proc. of NAACL-HLT*.

Rowan Zellers, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. From recognition to cognition: Visual commonsense reasoning. In *Proc. of CVPR*.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. Swag: A large-scale adversarial dataset for grounded commonsense inference. In *Proc. of EMNLP*.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *Proc. of EMNLP*.

Wanjun Zhong, Duyu Tang, Nan Duan, Ming Zhou, Jiahai Wang, and Jian Yin. 2018. Improving question answering by commonsense-based pre-training. *ArXiv*, abs/1809.03568.

# Learning with Limited Data for Multilingual Reading Comprehension

**Kyungjae Lee**[1*] **Sunghyun Park**[1*] **Hojae Han**[1] **Jinyoung Yeo**[3]
**Seung-won Hwang**[1†] **Juho Lee**[2]
[1]Yonsei University, [2] NAVER Corp, [3] SK T-Brain

## Abstract

This paper studies the problem of supporting question answering in a new language with limited training resources. As an extreme scenario, when no such resource exists, one can (1) transfer labels from another language, and (2) generate labels from unlabeled data, using translator and automatic labeling function respectively. However, these approaches inevitably introduce noises to the training data, due to translation or generation errors, which require a judicious use of data with varying confidence. To address this challenge, we propose a weakly-supervised framework that quantifies such noises from automatically generated labels, to deemphasize or fix noisy data in training. On reading comprehension task, we demonstrate the effectiveness of our model on low-resource languages with varying similarity to English, namely, Korean and French.

## 1 Introduction

Reading comprehension question answering (RCQA) is one of many well-known NLP tasks, to answer questions based on text understanding. For RCQA, a well-known resource is SQuAD (Rajpurkar et al., 2016) with 100K QA data created by human, followed by NarrativeQA (Kočiskỳ et al., 2018), SQuAD 2.0 (Rajpurkar et al., 2018), and CoQA (Reddy et al., 2018). However, as these datasets support only English, supporting other languages requires either annotation efforts in a comparable scale (Lim et al., 2018), or modeling efforts to overcome the limitation of training resources in terms of **quantity** or **quality**. Our work pursues the latter goal.

To illustrate, consider an extreme scenario of bootstrapping a RCQA model for a new language with no labelled resource. We can overcome the limitation in **quantity** by generating alternative low-quality resources. First, neural machine translation (NMT) can convert existing English annotations into a target language (Faruqui and Kumar, 2015; Ture and Boschee, 2016). For example, (passage $p$, question $q$, answer $a$) in SQuAD can be translated into $(p', q', a')$ in a target language (Asai et al., 2018; Lee et al., 2018). Second, automatic labeling function (Du and Cardie, 2018; Serban et al., 2016) can be adopted to generate synthetic labels in a target language. For example, Du and Cardie (2018) leverages Question Generator (QG) and Answer Extractor (AE) as labeling function for RCQA, to create training resources of a virtually infinite amount from unlabeled corpora.

However, overcoming the quantity limitation using the above methods, leads to **quality** problems. As a consequence, some of the following assumptions on quality may cease to hold, after errors introduced from generation. (1) **Answerability**: the semantic of the translated passage or generated question may shift, so that a question gets unanswerable after generation, or (2) **Answer alignment**: the answer span in a target language may become incorrect. Our goal is estimating the quality of resources and even improving the quality, to overcome the limited quality of training resources.

For the goal, we exploit noisy data from the above two sources, translator and labeling function, considered as **weak labels**, and pursue robust learning overcoming noises: Our key contribution is ***Refinery*** network that predict confidence – quality of $(p, q, a)$ instance into a score in the range of 0 to 1. While most existing weakly supervision approach focuses on generating positive examples, we leverage synthetic negative examples for training *Refinery*, optimized to distinguish positive from negative one.

As shown in Figure 1, a training procedure of

*First two authors equally contributed to this work.
†correspond to seungwonh@yonsei.ac.kr



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2840–2850, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

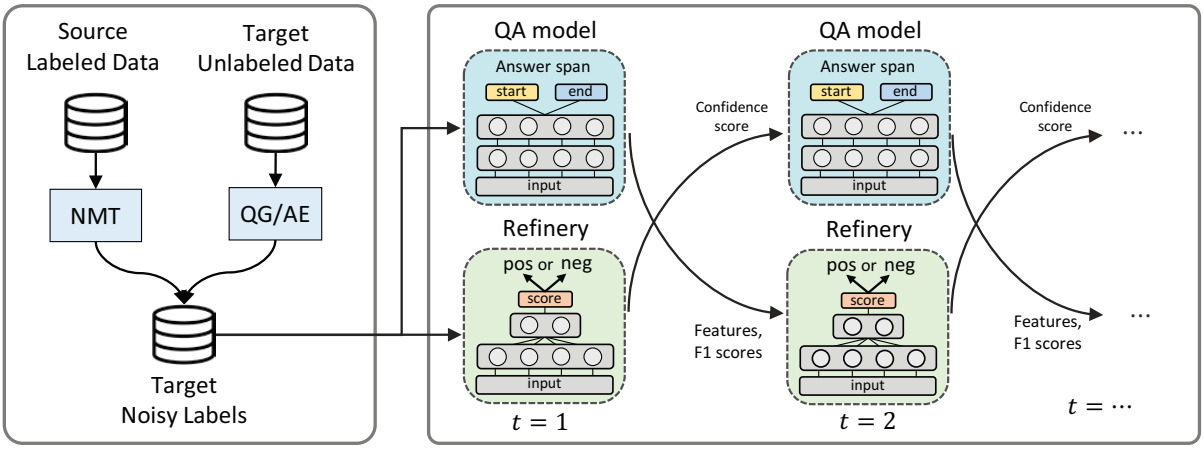


Figure 1: Overview of our approach

QA and *Refinery* is iterative: QA model is used for confidence estimation and *Refinery* is used to determine which noisy instances should be deemphasized in training, or whether to modify wrong labels, towards the direction of improving QA model. As our framework does not re-train or access data generation models, our approach is agnostic to generation model, such that it can extend to generation models not discussed here.

For experiments on RCQA in a new language, our model is evaluated on three human-annotated sets. Our experiments show that our method with *Refinery* augments training data and outperforms all state-of-the-arts. This observation can be generalized over similar and distant language pairs (English with French and Korean respectively), and with and without significant topic overlaps: To validate this claim, we evaluate over widely adopted human generated evaluation sets on Korean and French Wikipedia documents. In addition, our reported gains are orthogonal to a pre-trained model, such that we can easily leverage the strength of the latest BERT model (Devlin et al., 2018), and contribute additional gains. Our implementation is available at: https://github.com/lkj0509/multilingual_MRC.

## 2 Preliminaries

RCQA problem is defined based on SQuAD dataset (Rajpurkar et al., 2016) in English, which consists of 23,215 passages and 100k+ question-answer pairs requiring the understanding of corresponding passages to answer correctly. Let $p$ be a passage with $m$ words, *i.e.*, $p = \{w_1^p, w_2^p, ..., w_m^p\}$, and $q$ be a question with $n$ words, *i.e.*, $q = \{w_1^q, w_2^q, ..., w_n^q\}$. Then, given a pair of passage $p$ and question $q$, the objective of RCQA is to estimate a consecutive answer span $a$, *i.e.*, $a = \{w_i^p, w_{i+1}^p, ..., w_j^p\}$ where $a \subseteq p$. For task evaluation, the estimated answer span $a$ is compared with the ground truth answer span $a^*$ in terms of F1 score at word-level.

The majority of RCQA models (Seo et al., 2016; Yu et al., 2018; Devlin et al., 2018) consist of (a) encoding the passage and question into a fixed-size vector, then (b) decoding to predict the probability of each position in the passage being the start or end of an answer span. As a basic QA model, we start from BiDAF model (Seo et al., 2016), which is a widely known open-source code.[1] BiDAF model uses bi-directional LSTM networks with attention mechanism to align question with passage and vice-versa. More specifically, the probabilities of the starting and ending position are modeled as:

$$\begin{aligned} \mathbf{P}^1 &= softmax(h_1 \cdot [M^0 ; M^1]) \\ \mathbf{P}^2 &= softmax(h_2 \cdot [M^0 ; M^2]) \end{aligned} \quad (1)$$

where $h_1$, $h_2$ are trainable weights, and $M^0$, $M^1$, $M^2$ are the hidden states at each LSTM layer to represent the passage words according to the given question. The probability of $s$-to-$e$ span of the passage is defined as follows:

$$P(a = \{w_s^p, ..., w_e^p\} | p, q) = \mathbf{P}_s^1 \times \mathbf{P}_e^2 \quad (2)$$

In test, the model selects the answer span with the highest probability (Eq. (2)) during post-processing.

[1] https://github.com/allenai/bi-att-flow

## 3 Data Generation for New Language

This section introduces how we generate weak supervisions, from machine translation and synthetic label generation.

### 3.1 Method I: Neural Machine Translation

SQuAD is a set of $(p, q, a)$ triple, where answer $a$ to question $q$ can be found as a consecutive substring match in the passage $p$ (*i.e.*, $a \subseteq p$). Due to this property of SQuAD, translating the triple $(p, q, a)$ is non-trivial: when $p$ and $a$ are translated into target language, $p_t$ and $a_t$, $a_t \subseteq p_t$ may no longer hold. Therefore, we need to find the answer span $a_t$ in the target language, by matching with $a_s$ in the source language $s$. We overviewed a high-precision baseline (Lee et al., 2018) where $a_s$ and $p_s$ are independently translated, and $a_t$ is found only when the translation of $a_s$ is exactly and consecutively found in that of $p_s$. However, this suffers low-recall, considering we found only 53.6% spans among whole translated Q-A pairs by the high-precision method.

To complement, we propose a perfect-recall alignment to find 46.4% spans that cannot be extracted by the above method. Our method is extensible to any language with an existing open-source NMT, leveraging only its attention scores (Bahdanau et al., 2014).

**(1) One-to-one alignment:** A widely adopted simplifying assumption for machine translation is that each target word is aligned to one source language word (Brown et al., 1993). Based on attention score in NMT, each word in $a_s$ can be aligned with word in the target language, by selecting highest score. After 1-to-1 matching, we select the longest sub-sequence as answer span $a_t$.

**(2) Span-to-span alignment:** One-to-one assumption is simplifying, by treating values in the alignment matrix as binary and excluding a possibility that a word can be aligned with multiple target words. Instead, we align span-to-span, to calculate a "soft" score between $a_s$ and $a_t$, and change the span boundary dynamically. To illustrate, assume $a_t = \{i, ..., j\}$ is the answer span found from the one-to-one alignment. Before finalizing this answer, we may ask whether changing the boundary $i$ and $j$ improves the match[2]. Formally, when $S_{m,n}$ denotes $(m, n)$-th element in the alignment matrix, we can evaluate the match score of between $a_s$ and $a_t$ by averaging all pairwise combinations in the two spans: $\boldsymbol{S} = average(S_{m,n})$ for $\forall m \in a_s$, $\forall n \in a_t$. If changing the target boundary $a'_t = \{i', ..., j'\}$ improves this score, we will make a modification. Specifically, we consider updating the end position to $j'$ in the range of $j \pm N$, within a pre-defined window size $N$, and enqueue a possible update $j'$ if the average score of $\boldsymbol{S}$ for $a_s$ and $a'_t$ is higher than that of $a_s$ and $a_t$. Similarly, we compare the scores of start position in the range of $i \pm N$ to enqueue a possible update $i'$. After this, highest-scoring update pair $(i', j')$ in the queue (Hwang and Chang, 2005) can be aligned.

[2]We notice that 16% spans from 1-to-1 alignment can be changed to gain F1 score of 1% point on Korean test set.

### 3.2 Method II: Synthetic Label Generation

This section introduces an automatic labeler which can generate question-answer pairs from unlabeled text crawled for the target language. Training data collected via NMT is inherently skewed to specific domains answerable for English-speaking area (*e.g.*, United States), which causes the domain gap with test data on a new language. Automatic labeler can complement by collecting unlabeled text on the domains not covered, and generating synthetic labels.

For generating synthetic training data, we leverage Question Generator (QG) and Answer Extractor (AE) as labeler, following (Du and Cardie, 2018). Given a passage in a target language, our goal is to generate question-answer pairs, related to the given passage. In (Du and Cardie, 2018), BiLSTM-CRF model (Huang et al., 2015) is used for AE, classifying whether the word belongs to the answer span. For QG, sequence-to-sequence model (Bahdanau et al., 2014) is adopted, by setting a passage and answer candidates as input and question words as output. To train QG and AE, we use weak labels obtained from Method I. At inference time of QG and AE, we insert human-written passages, crawled from web documents in a target language (*e.g.*, Wikipedia), and obtain pairs of question and answer as the output of QG and AE, respectively.

## 4 Weakly-supervised QA model

In this section, we illustrate how we score confidence for noisy training data obtained from the above two generation methods. We propose a new *Refinery* network, of not only estimating confi-

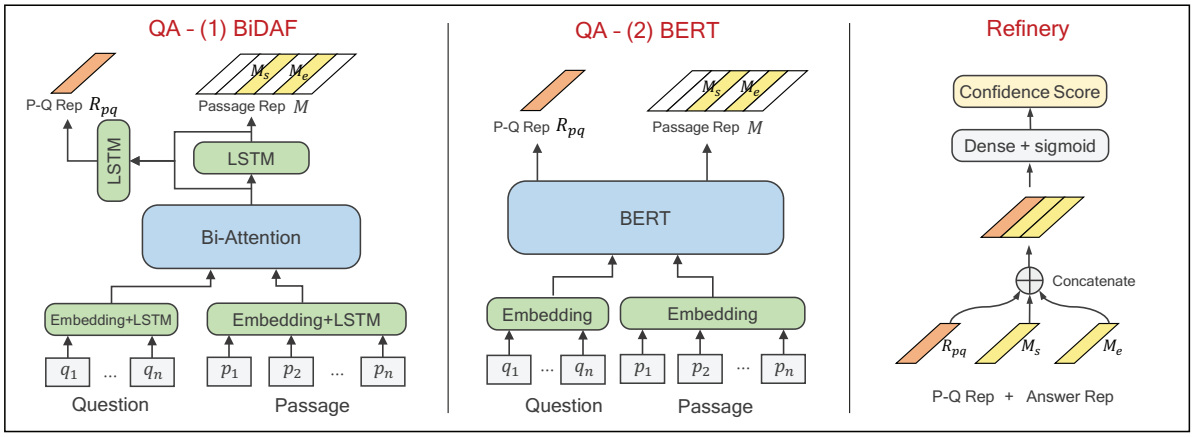


Figure 2: Architecture of our QA model and *Refinery*. Our approach of *Refinery* (Right) can be applicable to various RCQA models, illustrated with BiDAF (Left) and BERT (Center) here.

dence of noisy input from multiple generators but also improving labels and QA model. Existing work (Bach et al., 2011) estimates translation error that can be used as a confidence proxy. However, it requires large labels through human-correction, and cannot apply to other generators such as QG. Our goal is confidence estimation regardless of whether it is generated from NMT or QG methods (generator-agnostic).

### 4.1 Refinery Network

Our *Refinery* aims to score the quality of the generated $(p, q, a)$ labels, depending on whether the example is positive or negative. It is known that RCQA models trained solely on positive examples fail on unanswerable cases, assuming that a correct answer is guaranteed to exist in the passage (Rajpurkar et al., 2018). A simple remedy is to manually augment training resources with negative examples, to teach how to distinguish them from positive examples. Our key contribution is automatic collection of both positive and negative training examples for training **confidence** prediction, which we will discuss in details below.

Our research question now is then: how do we obtain positive and negative examples? For positive examples, we treat weak labels from the generation methods as "pseudo-positive", since it would be positive, when the translation/generation is perfect. For negative examples, we generate synthetic noises, similarly to translation/generation errors, by adopting a naive but effective method to change words or sentences in a positive example (Levy et al., 2017; Zhang et al., 2019; Yang et al., 2019).

Specifically, we modify pseudo-positive labels to generate synthetic negative examples of unanswerable or wrong answer. For unanswerable cases, (1) we first replace a question with its semantic neighbor derived from other documents (*e.g.*, "who first discovered america?" into "who first discovered canada?"). For this, we use the average of word embedding and the cosine distance. Second, (2) we modify a passage by removing a sentence containing the answer span, so that the modified passage is the subset of original one, but unanswerable. Lastly, for wrong answers, (3) we perturb an answer span into a random span in the same passage, which is answerable but with an incorrect label. We generate negative examples from the above (1)-(3) methods (33.3% each). As we will show later in Section 5, these negative examples are empirically effective for training *Refinery*.

The training objective is that $CS$ of pseudo-positive example should be greater than that of negative one, with at least margin $\delta$. To deal with pseudo-positive examples unequally, we set the margin $\delta$ dynamically, which is derived from F1 score of QA model. That is, we relatively assign a larger margin to a higher confidence instance, and smaller otherwise. The magnitude of margin depends on the F1 score of the predicted answer from QA model. Considering the margin $\delta$, the final objective function is computed as follows:

$$\delta_i = \alpha \cdot F1\ score(p_i^+, q_i^+, a_i^+)$$
$$L_{cs} = \sum_i max(0, \delta_i - \{CS_i^+ - CS_i^-\}) \quad (3)$$

where $\alpha$ is a hyper-parameter, $CS_i^+$ indicates the confidence of positive example, and $CS_i^-$ is that of negative example.

Now, to obtain the confidence score, we use BiDAF (Seo et al., 2016) as QA model, as shown in Figure 2 (Left). First, to model answerability of $(p, q)$, we extract the feature of $(p, q)$ from QA model. For this, we concatenate the two hidden states $M^1$ and $M^2$ from BiDAF, as mentioned in Section 2.1. Then, the representations are aggregated by LSTM and attention layer, as follows:

$$\begin{aligned} G &= \sigma(W_1[M^1 ; M^2] + b_1) \\ \overline{G} &= \overleftrightarrow{LSTM}(G) \\ a &= softmax(v_1 \cdot \sigma(W_2\overline{G} + b_2)) \\ R_{pq} &= \sum_i a_i \cdot \overline{G_i} \end{aligned} \tag{4}$$

where $W_1$, $W_2$, $v_1$, $b_1$, $b_2$ are trainable weights, and $\sigma$ is tanh function. The vector $R_{pq}$ containing the information of both passage and question will be used for confidence score.

Second, to model the confidence of answer labels, we represent answer spans in the labels. For this, we concatenate two hidden states $M_s^1$ and $M_e^2$ of $s$-th start and $e$-th end words in the answer. The final confidence score $CS$ is obtained from $R_{pq}$ and the answer feature, as follows:

$$\begin{aligned} R_{span} &= [\ M_s^1 ; M_e^2\ ] \\ CS &= sigmoid(v_2 \cdot [\ R_{pq} ; R_{span}\ ] + b_3) \end{aligned} \tag{5}$$

where $v_2$ and $b_3$ are trainable weights, and $[\ ;\ ]$ indicates concatenation.

An extension to the above model is to replace BiDAF with BERT (Devlin et al., 2018), a pre-trained model on a masked language model task. Due to the commonality of BiDAF and BERT, taking question and passage as input and generating passage representations for answer span prediction, we can apply BERT to our approach, as shown in Figure 2 (Center), by modifying *Refinery* module of modeling confidence score. Meanwhile, architectural differences of the two, such as BiDAF using bi-attention layers and BERT using self-attention layers, require some minor changes: In BERT, as the output of $CLS$ token can represent the input sequence of passage and question, we use the output vector of $CLS$ as $R_{pq}$. For $R_{span}$, we use the hidden states at the layer of BERT, instead of $M$ in Eq. (5). We will show the effectiveness of our approach on both BiDAF and BERT model in Section 5.

### 4.2 Improving QA and Weak Labels

During an iterative procedure of QA and *Refinery*, our approach improves both QA model and weak labels. Using confidence, *Refinery* acts as two functions: (1) **down-weighting** instances with low-quality, and (2) **answer modification** with low-quality for higher quality.

The objective of original QA model minimizes the sum of the negative log probabilities:

$$\begin{aligned} L_{qa} &= -\sum_i^N log(\mathbf{P}^1_{s_i}) + log(\mathbf{P}^2_{e_i}) \\ &= -\sum_i^N log(P(a_i|p_i, q_i)) \end{aligned} \tag{6}$$

where $N$ is the number of examples in training set, $s_i$ and $e_i$ are the start and end indices of the $i$-th example, respectively. For down-weighting noise, we combine this QA loss function with weighted confidence score, based on function $f(CS(p, q))$:

$$L_{qa'} = -\sum_i^N f(CS_i) \cdot log(P(a_i|p_i, q_i)) \tag{7}$$

However, in initial steps of training, the confidence score is not reliable, and may cause unstable training. To avoid the problem, we apply an annealing technique, adjusting the contribution of the confidence score. We set the initial function $f$ to be uniform constant, and then gradually increase the contribution of instance weighting as learning steps. That is, we design the function $f$ as follows:

$$f(CS_i) = \frac{c \cdot e^{-\lambda t} + CS_i}{c \cdot e^{-\lambda t} + 1} \tag{8}$$

where $t$ is the number of current iteration step, $c$ and $\lambda$ are hyper-parameters.

Besides down-weighting noisy instances, we could change some answers in weak labels to higher-quality one, while unanswerable passages-question pairs are difficult to modify. To improve such labels, we compare confidence scores between the answer $a^*$ predicted by QA model and $a$ in the weak label. If the confidence increase is greater than threshold $\gamma$, we change the answer $a$ to the model-generated answer $a^*$ when each epoch ends. Finally, we train the QA and *Refinery* modules jointly, as Eq. (3) and (7), which are updated in turns[3], until convergence. Algorithm 1 describes this procedure in detail.

[3]When updating *Refinery*, we freeze parameters in QA model, because negative examples could be harmful in QA

**Algorithm 1** Iterative QA and Refinery

$N = 0$, $\theta_{qa}^{(0)} \leftarrow$ Initial QA, $\theta_{re}^{(0)} \leftarrow$ Initial Refinery
**for** *epoch* ***in*** *epochs* **do**
  **for** $i$ ***in*** *max steps* **do**
    Let $D$ be $(P, Q, A)$ triplets in mini-batch
    $D_i^+ \leftarrow (P_i^+, Q_i^+, A_i^+)$ from weak labels
    $D_i^- \leftarrow (P_i^-, Q_i^-, A_i^-)$ from negative examples
    $A_i^* \leftarrow$ Answer of $(P_i^+, Q_i^+)$ from QA $\theta_{qa}^{(N)}$
    $F_i \leftarrow$ F1 score of $(A_i^+, A_i^*)$
    $CS_i \leftarrow$ Confidence$(D_i^+)$ from Refinery $\theta_{re}^{(N)}$
    $\theta_{qa}^{(N+1)} \leftarrow$ Update QA model on $(D_i^+, CS_i)$
    $\theta_{re}^{(N+1)} \leftarrow$ Update Refinery on $(D_i^+, D_i^-, F_i)$
    $N \leftarrow N + 1$
  **end**
  $D^* \leftarrow (P^+, Q^+, A^*)$ predicted from QA $\theta_{qa}^{(N)}$
  $CS^* \leftarrow$ Confidence$(D^*)$ on predicted answers
  $CS^+ \leftarrow$ Confidence$(D^+)$ on weak labels
  **if** $CS^* - CS^+ > \gamma$ **then**
    $D^+ \leftarrow$ modified label $(P^+, Q^+, A^*)$
  **end**
**end**

## 5 Experiment

In this section, we address the following research questions:

- RQ1: Does our proposed work outperform existing approaches? Does it generalize for languages with diverse distance or topics?
- RQ2: Does our model generally work on more noisy environment?
- RQ3: Is our *Refinery* effective in distinguishing positive and negative set? How does QA/*Refinery* contribute to each other?

### 5.1 Dataset

For evaluation, we conduct experiments on three datasets for French and Korean RCQA. First is the public datasets, built on French (327 pairs) (Asai et al., 2018) and Korean (2K pairs) (Lee et al., 2018) Wikipedia, denoted as **Wiki_Fr** and **Wiki_Kr** respectively. Supposing zero-annotation at training, we use their training set (2K) in **Wiki_Kr** as our dev set, and evaluate our model on the test set. Third dataset is a history RC dataset (7K), which we collect on Korean-specific topics to show the robustness of our model with respect to domain gap: Specifically, dealing with Korean history documents, as denoted as **History_Kr**. For annotation, we follow the convention of SQuAD (Rajpurkar et al., 2016). Lastly, for generating synthetic labels, we crawled about 400 articles in each French and Korean, with general and (Korean) history topic from Wikipedia and Doopedia[4], respectively. Through automatic labeler (QG/AE) as mentioned in Section 3.2, we generated totally 100K QA pairs from passages in Wikipedia and Doopedia.

### 5.2 Experimental Setting

Our implementation settings for QA model follow original BiDAF (Seo et al., 2016) and BERT (Multilingual-Base version) (Devlin et al., 2018). For machine translation, we use pre-trained and open-sourced NMT models for French[5] and Korean[6]. As hyper-parameters, $\alpha$, $c$, $\lambda$ and $\gamma$, are set to 1/10, 10, 1/4000, and 0.5 respectively, optimized by dev set. For QG and AE implementations, we follow the setting of NQG model (Zhou et al., 2017). In BiDAF model, we used Natural Language Toolkit (NLTK) and KoNLPY[7] for French/Korean tokenizer, and FastText (Mikolov et al., 2017) for word embedding.

### 5.3 Evaluation of Full QA Model

We compare our QA model with the following baselines:

**Base1&2:** Asai et al. (2018) translate $(p, q, a)$ in target language into English, then test on pre-trained English QA model. The answer from model is translated back to target language. We present their reported F1 scores for French with and without ELMo (Peters et al., 2018), and leave unreported results blank.

**Base3:** Lee et al. (2018) proposed semi-supervised method, to train the weak model on a small human-labeled set, and evaluate translated QA pairs by the output probabilities (in Eq. (2)) of the weak QA model. For training QA, the translated data over threshold is used together with a human-labeled data. We present their reported F1 scores and leave unreported results blank.

**Ours:** Our proposed models are trained on BiDAF and BERT, and with/without Refinery (for ablation purposes described below).

[4] https://www.doopedia.co.kr
[5] https://github.com/pytorch/fairseq/blob/master/examples/translation
[6] https://developers.naver.com/products/nmt/
[7] https://konlpy-ko.readthedocs.io/

Table 1: The results of QA models. Compared to version w/o Refinery, our models outperform with statistical significance (* indicates $p < 0.05$).

| | Method | QA model | F1 score | | |
|---|---|---|---|---|---|
| | | | Wiki_Fr | Wiki_Kr | History_Kr |
| Base1 | Test on source (Asai et al., 2018) | BiDAF+ELMo | 61.9 | - | - |
| Base2 | Test on source (Asai et al., 2018) | BiDAF | 57.6 | - | - |
| Base3 | Semi-supervision (Lee et al., 2018) | BiDAF | - | 71.5 | - |
| Ours | No weighting (w/o *Refinery*) | BiDAF | 60.8 | 70.1 | 61.5 |
| | Full model with *Refinery* | BiDAF | 63.5* | 73.4* | 64.1* |
| | No weighting (w/o *Refinery*) | BERT | 74.3 | 77.1 | 67.7 |
| | Full model with *Refinery* | BERT | **76.6*** | **79.7*** | **70.4*** |

**Results on QA datasets** Table 1 shows the comparison on En-Kr (distant) and En-Fr (close) language pairs. Similarly, we contrast in diverse topic-similarity scenarios: History (distant) and Wiki (close). When compared with Base1&2 (Test on source), our model outperforms the two models, even when Base1 is boosted by ELMo (Peters et al., 2018). When compared with Base3, semi-supervision (Lee et al., 2018) using 2K human labels outperforms our BiDAF model (without *Refinery*), on Wiki_Kr set. However, ours with *Refinery* outperforms the Base3 model, which means that large and noisy labels refined by our method have comparable quality to small but strong labels.

To show that the effectiveness of our approach is orthogonal to QA model, we construct two QA models, BiDAF and BERT, as in Table 1. When compared with *No weighting*, our full model on BiDAF improved 2.7%, 3.3% and 2.6% of F1 scores, and that on BERT improved 2.3%, 2.6% and 2.7% of F1 score, on Wiki_Fr, Wiki_Kr and History_Kr, respectively. BERT model with our *Refinery* performs the best among all models, by adding the power of pre-trained representations.

**Ablation Study** As shown in Table 2, we conduct an ablation study on both BiDAF and BERT models, by examining the effects, from removing each component. In (A), we replace our confidence score (CS) with the probabilities in Eq. (2), which is similar to the use of confidence in (Lee et al., 2018). We normalize the probabilities in mini-batch by softmax function, then apply to *down-weighting*. On the QA model using the replaced confidence, the performance decreased significantly, suggesting the importance of CS. In (B), we remove a dynamic margin in Eq. (3), by setting the margin as a constant value instead (*i.e.*, $\delta = 1$). Using static margin lowered the performance of both QA models, suggesting that QA feedback (dynamic margin) is effective for refining noises. In (C), we remove *answer modification* in Section 4.2, while preserving *down-weighting* module. Compared with our full model, adding *answer modification* improved the performance on both QA models. We can also observe the effect of *down-weighting*, as the model without *answer modification* still outperforms that with *No weighting* in Table 1.

Table 2: The ablation study on three datasets. The number inside the parenthesis indicates the decrease from our full model.

(a) On BiDAF

| | | F1 score | | |
|---|---|---|---|---|
| | | Wiki_Fr | Wiki_Kr | History_Kr |
| | Our full model | 63.5 | 73.4 | 64.1 |
| (A) | Replace CS with Prob | 61.1 (-2.4) | 70.3 (-3.1) | 61.8 (-2.3) |
| (B) | Remove dynamic margin | 62.0 (-1.5) | 72.3 (-1.1) | 62.4 (-1.7) |
| (C) | Remove answer modification | 62.3 (-1.2) | 71.9 (-1.5) | 62.8 (-1.3) |

(b) On BERT

| | | F1 score | | |
|---|---|---|---|---|
| | | Wiki_Fr | Wiki_Kr | History_Kr |
| | Our full model | 76.6 | 79.7 | 70.4 |
| (A) | Replace CS with Prob | 74.3 (-2.3) | 77.3 (-2.4) | 67.8 (-2.6) |
| (B) | Remove dynamic margin | 75.1 (-1.5) | 77.8 (-1.9) | 68.4 (-2.0) |
| (C) | Remove answer modification | 75.2 (-1.4) | 78.6 (-1.1) | 69.2 (-1.2) |

For RQ2, we design scenarios with noisier training data to demonstrate the robustness. When training QA, we replace positive examples with negative examples, perturbing an input to fool a machine model. Figure 3(a) shows the robustness of our BiDAF model over varying noise ratios. As

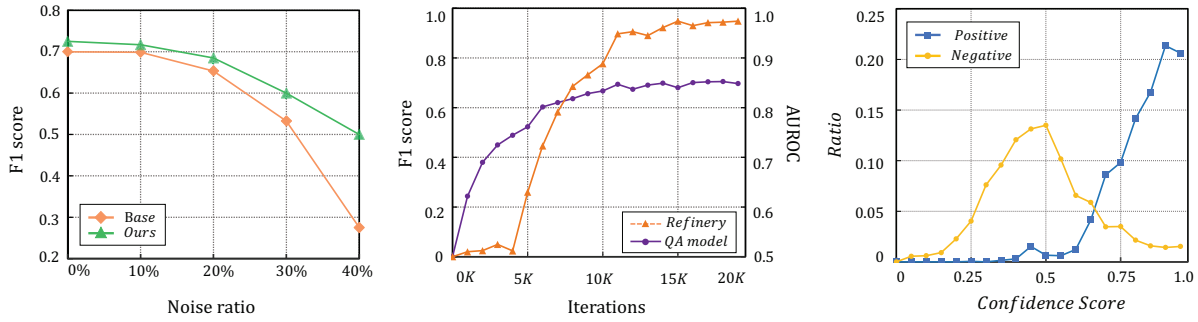


(a) QA performance as noise ratio (b) Iterative QA and *Refinery* (c) Distribution of confidence score

Figure 3: Experimental results of our model on dev set. (a) Comparison of robustness between Base (w/o *Refinery*) and Ours in noisy environments. (b) The performances of QA (F1 score) and *Refinery* (AUROC) over iterations. (c) Distribution of confidence scores for positive and negative examples.

Table 3: The results of confidence score evaluation

| | TNR (95% TPR) | AUROC | AUPR |
|---|---|---|---|
| | (Wiki_Kr / History_Kr) | | |
| (A) Base1 | 47.4 / 39.0 | 82.9 / 85.8 | 78.2 / 85.4 |
| (B) Base2 | 44.9 / 69.5 | 71.2 / 83.5 | 63.3 / 75.7 |
| Ours | **88.4 / 90.9** | **97.7 / 98.5** | **98.2 / 98.8** |

noises increase, Base BiDAF model sharply drops its F1 score, while that of our full model degrades gracefully.

### 5.4 Evaluation of Confidence Score

This section addresses RQ3, aiming at directly observing whether our confidence score effectively distinguishes the positive from negative. As baselines, we conduct (A) and (B) in the above ablation study. (A) Base1 uses the probability in Eq. (2) as confidence score. (B) Base2 is *Refinery* without a dynamic margin ($\delta = 1$). In this experiment, we use BiDAF model on Wiki_Kr and History_Kr, and generate negative examples from positive test set as 1:1 ratio.

As a quantitative evaluation, we measure three metrics of detecting negative examples, comparing confidence of positive and negative one.

- True Negative Rate (TNR) at 95% True Positive Rate (TPR).
- Area under the Receiver Operating Characteristic Curve (AUROC).
- Area under the Precision – Recall Curve (AUPR).

As shown in Table 3, our *Refinery* outperforms other baselines with statistical significance of $p <$ 0.01 and by a large margin on all measures, showing *Refinery* is effective in distinguishing positive-negative pairs. We also compare our work with and without a dynamic margin from QA feedback, to show that such strategy improves the performance.

To show performance of QA/*Refinery* over iterations, we check their performance as iterations continue on dev set, in Figure 3(b). In early iterations, *Refinery* cannot contribute to QA, because F1 score of all instance is zero. After the training stabilizes, the performance of *Refinery* increases rapidly. Figure 3(c) contrasts the distribution of confidence from positive and negative samples: Unlike negative examples of varying confidence, positive examples are clearly skewed to high-ends.

Table 4 compares the examples in Baseline and our model, where the first two are high-quality data and the third and forth are not. In the first example, both models show high score, while Baseline underestimates the quality and *Refinery* works correctly for the second example. The third example is unanswerable on $(p, q)$ pair, since the words about the subject disappeared during translation, which is assigned to lower scores by two models. The last example has also translation error where the given question was mistranslated from 'when' to 'how'. In this case, our *Refinery* network successfully assigned low confidence score, as the question is no longer answerable after a bad translation, but Baseline gives a high confidence.

## 6 Related Work

**Multilingual Task:** NMT has played an important role in addressing multilinguality. For example, a straightforward solution for RCQA is translating $(p, q)$ in target language to English and ap-

Table 4: Qualitative examples showing our proposed confidence estimation score CS for ($p$,$q$) pairs in Korean. Human translated English is given in the bracket only for readability. We omitted some irrelevant sentences in the passages for conciseness.

| | Passage [**Answer**] / Question | Baseline $P(a\|p,q)$ | $CS$ |
|---|---|---|---|
| P1 | Kor: 피셔와 그의 팀 멤버들은 **[요서프의 꿈]**으로 돌아오는데 성공했다. {...}<br>Eng: Fischer and his team members are successful in returning to **[Yusuf's dream]**. {...} | 0.877 | 0.978 |
| Q1 | Kor: 피셔와 그의 팀원들은 어디로 돌아왔나요?<br>Eng: Where did Fischer and his team return? | | |
| P2 | Kor: 에릭은 **[헨리 앤 사우스 옥스퍼드셔 스탠다드]** 에 두 편의 시를 기고하였다.{...}<br>Eng: Eric published two poems in **[the Henley and South Oxfordshire Standard]**. {...} | 0.010 | 0.981 |
| Q2 | Kor: 에릭은 어디에 시를 기고했나요?<br>Eng: Where did Eric publish his poems? | | |
| P3 | Kor: 그러나 공대는 **[1920년]**에 설립되었다. {...}<br>Eng: But, the engineering college was founded in **[1920]**. {...} | 0.015 | 0.053 |
| Q3 | Kor: 노트르담 대학은 몇 년도에 설립되었는가?<br>Eng: When was University of Notre Dame founded? | | |
| P4 | Kor: **[2016년 2월 6일]** 비욘세는 음악 스트리밍 서비스만을 위해 새로운 싱글을 발매했다. {...}<br>Eng: **[On February 6, 2016]**, Beyonce released a new single only for a music streaming. | 0.864 | 0.090 |
| Q4 | Kor: 싱글 앨범은 어떻게 발매 되었나요?<br>Eng: How was the single released? | | |

ply English-based models, then translate the answer back (Asai et al., 2018). Not only for question answering, such method has been successful in sentiment classication (Zhou et al., 2016), relation extraction (Faruqui and Kumar, 2015), and causal commonsense (Yeo et al., 2018). However, these approaches are dependent on quality of translation.

**RCQA for Resource-Poor Language** To overcome the lack of RCQA resources on other language, in (Lee et al., 2018), they proposed a semi-supervised strategy to remove the noise instances. They hand-annotated a small seed set of QA pairs to train a weak QA. Then, the weak QA indirectly judges the quality of machine-translated SQuAD pairs, by considering the output probability as confidence score. As confidence score, they selectively remove translated pairs below a fixed threshold. Meanwhile, we eliminate the needs for human annotation, by leveraging positive and negative sets from translated and generated data.

**Combining models of QG and QA:** Several works (Wang et al., 2017; Tang et al., 2017; Tang et al.) propose a framework jointly optimizing QG and QA model as dual. These approaches share commonality with our approach for generating training data with another model, but with the following crucial difference: Existing work can use a reliable training set annotated by human, which can be strong hint to decide the quality of generated data. For example, in (Tang et al.), QG generates questions on the given answer sentence, then the generated question is compared with the original question in training set. Meanwhile, we do not require human annotation, and rather leverage automatically generated from NMT and QG models, judiciously guided by our Refinery network.

## 7 Conclusion

This paper studies a zero-resource training data generation for supporting RCQA to a new target language. Inspired by limited resources, we generate RCQA data, by using existing NMT and QG techniques. To explore the noisy generated data, we proposed an integrated QA model with *Refinery* to control the generated data as confidence score. Our results showed that our strategies using *Refinery* and generated data enhance the performance of existing QA model, and *Refinery* has effectiveness to distinguish positive-negative pairs.

## Acknowledgements

We thank NAVER Corporation for sponsoring this project. Partial support received from IITP grant funded by the Korea government (MSIT) (No.2017-0-01779, XAI).

## References

Akari Asai, Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2018. Multilingual extractive

reading comprehension by runtime machine translation. *arXiv preprint arXiv:1809.03275*.

Nguyen Bach, Fei Huang, and Yaser Al-Onaizan. 2011. Goodness: A method for measuring machine translation confidence. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 211–219. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Xinya Du and Claire Cardie. 2018. Harvesting paragraph-level question-answer pairs from wikipedia. *arXiv preprint arXiv:1805.05942*.

Manaal Faruqui and Shankar Kumar. 2015. Multilingual open relation extraction using cross-lingual projection. *arXiv preprint arXiv:1503.06450*.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Seung-won Hwang and Kevin Chang. 2005. Optimizing access cost for top-k queries over web sources: A unified cost-based approach. In *ICDE*.

Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gáabor Melis, and Edward Grefenstette. 2018. The narrativeqa reading comprehension challenge. *Transactions of the Association of Computational Linguistics*, 6:317–328.

Kyungjae Lee, Kyoungho Yoon, Sunghyun Park, and Seung-won Hwang. 2018. Semi-supervised training data generation for multilingual question answering. *LREC*.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. *arXiv preprint arXiv:1706.04115*.

Seungyoung Lim, Myungji Kim, and Jooyoul Lee. 2018. Korquad: Korean qa dataset for machine comprehension. *Communications of the Korean Institute of Information Scientists and Engineers*, pages 539–541.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2017. Advances in pre-training distributed word representations. *arXiv preprint arXiv:1712.09405*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Siva Reddy, Danqi Chen, and Christopher D Manning. 2018. Coqa: A conversational question answering challenge. *arXiv preprint arXiv:1808.07042*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Iulian Vlad Serban, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. 2016. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 588–598.

Duyu Tang, Nan Duan, Tao Qin, and Ming Zhou. 2017. Question answering and question generation as dual tasks. *arXiv preprint arXiv:1706.02027*.

Duyu Tang, Nan Duan, Zhao Yan, Zhirui Zhang, Yibo Sun, Shujie Liu, Yuanhua Lv, and Ming Zhou. Learning to collaborate for question answering and asking. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Ferhan Ture and Elizabeth Boschee. 2016. Learning to translate for multilingual question answering. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 573–584.

Tong Wang, Xingdi Yuan, and Adam Trischler. 2017. A joint model for question answering and question generation. *arXiv preprint arXiv:1706.01450*.

Wei Yang, Yuqing Xie, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019. Data augmentation for bert fine-tuning in open-domain question answering. *arXiv preprint arXiv:1904.06652*.

Jinyoung Yeo, Geungyu Wang, Hyunsouk Cho, Seungtaek Choi, and Seung-won Hwang. 2018. Machine-translated knowledge transfer for commonsense causal reasoning. In *AAAI*.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*.

Yuan Zhang, Jason Baldridge, and Luheng He. 2019. Paws: Paraphrase adversaries from word scrambling. *arXiv preprint arXiv:1904.01130*.

Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. 2017. Neural question generation from text: A preliminary study. In *National CCF Conference on Natural Language Processing and Chinese Computing*, pages 662–671. Springer.

Xinjie Zhou, Xiaojun Wan, and Jianguo Xiao. 2016. Attention-based lstm network for cross-lingual sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 247–256.

# A Discrete Hard EM Approach for Weakly Supervised Question Answering

**Sewon Min**[1], **Danqi Chen**[2,3], **Hannaneh Hajishirzi**[1,4], **Luke Zettlemoyer**[1,3]
[1]University of Washington, Seattle, WA
[2]Princeton University, Princeton, NJ
[3]Facebook AI Research, Seattle, WA
[4]Allen Institute for Artificial Intelligence, Seattle, WA
{sewon,hannaneh,lsz}@cs.washington.edu danqic@cs.princeton.edu

## Abstract

Many question answering (QA) tasks only provide weak supervision for how the answer should be computed. For example, TRIVIAQA answers are entities that can be mentioned multiple times in supporting documents, while DROP answers can be computed by deriving many different equations from numbers in the reference text. In this paper, we show it is possible to convert such tasks into discrete latent variable learning problems with a precomputed, task-specific set of possible *solutions* (e.g. different mentions or equations) that contains one correct option. We then develop a hard EM learning scheme that computes gradients relative to the most likely solution at each update. Despite its simplicity, we show that this approach significantly outperforms previous methods on six QA tasks, including absolute gains of 2–10%, and achieves the state-of-the-art on five of them. Using hard updates instead of maximizing marginal likelihood is key to these results as it encourages the model to find the one correct answer, which we show through detailed qualitative analysis.[1]

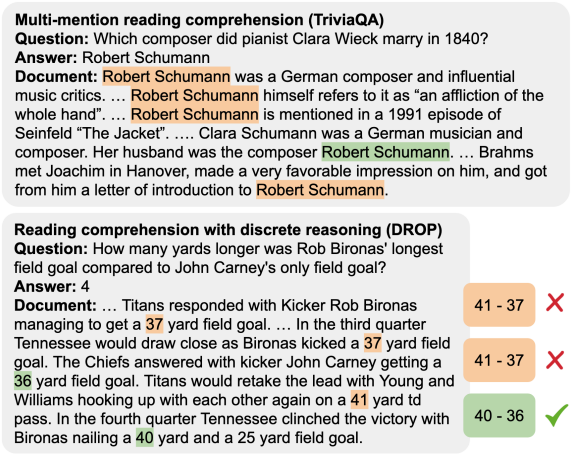


Figure 1: Examples from two different question answering tasks. **(Top) Multi-mention reading comprehension.** The answer text is mentioned five times in the given document, however, only the fourth span actually answers the question. **(Bottom) Reading comprehension with discrete reasoning.** There are many potential equations which execute the answer ('4'), but only one of them is the correct equation ('40-36') and the others are false positives.

## 1 Introduction

A natural setting in many question answering (QA) tasks is to provide weak supervision to determine how the question should be answered given the evidence text. For example, as seen in Figure 1, TRIVIAQA answers are entities that can be mentioned multiple times in supporting documents, while DROP answers can be computed by deriving many different equations from numbers in the reference text. Such weak supervision is attractive because it is relatively easy to gather, allowing for large datasets, but complicates learning because there are many different spurious ways to derive the correct answer. It is natural to model such ambiguities with a latent variable during learning, but most prior work on reading comprehension has rather focused on the model architecture and used heuristics to map the weak signal to full supervision (e.g. by selecting the first answer span in TRIVIAQA (Joshi et al., 2017; Tay et al., 2018; Talmor and Berant, 2019)). Some models are trained with maximum marginal likelihood (MML) (Kadlec et al., 2016; Swayamdipta et al., 2018; Clark and Gardner, 2018; Lee et al., 2019), but it is unclear if it gives a meaningful improvement over the heuristics.

In this paper, we show it is possible to formulate a wide range of weakly supervised QA tasks as discrete latent-variable learning problems. First, we define a *solution* to be a particular derivation of a model to predict the answer (e.g. a span in

[1]Our code is publicly available at https://github.com/shmsw25/qa-hard-em.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2851–2864, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| Task & Dataset | # Examples | | | $\lvert Z \rvert$ | |
|---|---|---|---|---|---|
| | Train | Dev | Test | Avg | Median |
| **1. Multi-mention reading comprehension** | | | | | |
| TRIVIAQA (Joshi et al., 2017) | 61,888 | 7,993 | 7,701 | 2.7 | 2 |
| NARRATIVEQA (Kočiskỳ et al., 2018) | 32,747 | 3,461 | 10,557 | 4.3 | 5 |
| TRIVIAQA-OPEN (Joshi et al., 2017) | 78,785 | 8,837 | 11,313 | 6.7 | 4 |
| NATURALQUESTIONS-OPEN (Kwiatkowski et al., 2019) | 79,168 | 8,757 | 3,610 | 1.8 | 1 |
| **2. Reading comprehension with discrete reasoning** | | | | | |
| $\text{DROP}_{\text{num}}$ (Dua et al., 2019) | 46,973 | 5,850 | - | 8.2 | 3 |
| **3. Semantic Parsing** | | | | | |
| WIKISQL (Zhong et al., 2017) | 56,355 | 8,421 | 15,878 | 346.1 | 5 |

Table 1: Six QA datasets in three different categories used in this paper (detailed in Section 5) along with the size of each dataset. An average and median of the size of precomputed solution sets (denoted by $Z$) are also reported. Details on how to obtain $Z$ are given in Section 4.

the document or an equation to compute the answer). We demonstrate that for many recently introduced tasks, which we group into three categories as given in Table 1, it is relatively easy to precompute a discrete, task-specific set of possible solutions that contains the correct solution along with a modest number of spurious options. The learning challenge is then to determine which solution in the set is the correct one, while estimating a complete QA model.

We model the set of possible solutions as a discrete latent variable, and develop a learning strategy that uses hard-EM-style parameter updates. This algorithm repeatedly (i) predicts the most likely solution according to the current model from the precomputed set, and (ii) updates the model parameters to further encourage its own prediction. Intuitively, these hard updates more strongly enforce our prior beliefs that there is a single correct solution. This method can be applied to any problem that fits our weak supervision assumptions and can be used with any model architecture.

We experiment on six different datasets (Table 1) using strong task-specific model architectures (Devlin et al., 2019; Dua et al., 2019; Hwang et al., 2019). Our learning approach significantly outperforms previous methods which use heuristic supervision and MML updates, including absolute gains of 2–10%, and achives the state-of-the-art on five datasets. It outperforms recent state-of-the-art reward-based semantic parsing algorithms (Liang et al., 2018; Agarwal et al., 2019) by 13% absolute percentage on WIKISQL, strongly suggesting that having a small precomputed set of possible solutions is a key ingredient. Finally, we present a detailed analysis showing that, in practice, the introduction of hard updates encourages models to assign much higher probability to the correct solution.

## 2 Related Work

**Reading Comprehension.** Large-scale reading comprehension (RC) tasks that provide full supervision for answer spans (Rajpurkar et al., 2016) have seen significant progress recently (Seo et al., 2017; Xiong et al., 2018; Yu et al., 2018a; Devlin et al., 2019). More recently, the community has moved towards more challenging tasks such as distantly supervised RC (Joshi et al., 2017), RC with free-form human generated answers (Kočiskỳ et al., 2018) and RC requiring discrete or multi-hop reasoning (Dua et al., 2019; Yang et al., 2018). These tasks introduce new learning challenges since the gold solution that is required to answer the question (e.g. a span or an equation) is not given.

Nevertheless, not much work has been done for this particular learning challenge. Most work on RC focuses on the model architecture and simply chooses the first span or a random span from the document (Joshi et al., 2017; Tay et al., 2018; Talmor and Berant, 2019), rather than modeling this uncertainty as a latent choice. Others maximize the sum of the likelihood of multiple spans (Kadlec et al., 2016; Swayamdipta et al., 2018; Clark and Gardner, 2018; Lee et al., 2019), but it is unclear if it gives a meaningful improvement. In this paper, we highlight the learning challenge and show that our learning method, independent of the model architecture, can give a significant gain. Specifically, we assume that one of

mentions are related to the question and others are false positives because (i) this happens for most cases, as the first example in Table 2, and (ii) even in the case where multiple mentions contribute to the answer, there is often a single span which fits the question the best.

**Semantic Parsing.** Latent-variable learning has been extensively studied in the literature of semantic parsing (Zettlemoyer and Collins, 2005; Clarke et al., 2010; Liang et al., 2013; Berant et al., 2013; Artzi and Zettlemoyer, 2013). For example, a question and an answer pair $(x, y)$ is given but the logical form that is used to compute the answer is not. Two common learning paradigms are maximum marginal likelihood (MML) and reward-based methods. In MML, the objective maximizes $\sum_{z \in \hat{Z}} \mathbb{P}(z|x)$, where $\hat{Z}$ is an approximation of a set of logical forms executing $y$ (Liang et al., 2013; Berant et al., 2013; Krishnamurthy et al., 2017). In reward-based methods, a reward function is defined as a prior, and the model parameters are updated with respect to it (Iyyer et al., 2017; Liang et al., 2017, 2018). Since it is computationally expensive to obtain a precomputed set in semantic parsing, these methods typically recompute the set of logical forms with respect to the beam at every parameter update. In contrast, our learning method targets tasks that a set of solutions can be precomputed, which include many recent QA tasks such as reading comprehension, open-domain QA and a recent SQL-based semantic parsing task (Zhong et al., 2017).

## 3 Method

In this section, we first formally define our general setup, which we will instantiate for specific tasks in Section 4 and then we describe our learning approach.

### 3.1 Setup

Let $x$ be the input of a QA system (e.g. a question and a document) and $y$ be the answer text (e.g. 'Robert Schumann' or '4'). We define a *solution* as a particular derivation that a model is supposed to produce for the answer prediction (e.g. a span in the document or an equation to compute the answer, see Table 2). Let $f$ denote a task-specific, deterministic function which maps a solution to the textual form of the answer (e.g. by simply returning the string associated with a particular selected mention or solving an equation to get the final number, see Table 2). Our goal is to learn a model (with parameters $\theta$) which takes an input $x$ and outputs a solution $z$ such that $f(z) = y$.

In a fully supervised scenario, a true solution $\bar{z}$ is given, and $\theta$ is estimated based on a collection of $(x, \bar{z})$ pairs. In this work, we focus on a weakly supervised scenario in which $\bar{z}$ is not given. For this setup, we define $Z_{\text{tot}}$ as a finite set of all the possible solutions. In the case that the search space is very large or infinite, we can usually approximate $Z_{\text{tot}}$ with a high coverage in practice. Then, we obtain $Z = \{z \in Z_{\text{tot}} : f(z) = y\}$ by enumerating all $z \in Z_{\text{tot}}$. This results a set of all the possible solutions that lead to the correct answer. We assume it contains one solution that we want to learn to produce, and potentially many other spurious ones. In practice, $Z$ is defined in a task-specific manner, as we will see in Section 4.

At inference time, the model produces a solution $\tilde{z} \in Z_{\text{tot}}$ from an input $x$ with respect to $\theta$ and predicts the final answer as $f(\tilde{z})$. Note that we cannot compute $Z$ at inference time because the groundtruth $y$ is not given.[2]

### 3.2 Learning Method

In a fully-supervised setting where $\bar{z}$ is given, we can learn $\theta$ by optimizing the negative log likelihood of $\bar{z}$ given the input $x$ with respect to $\theta$.

$$J_{\text{Sup}}(\theta|x, \bar{z}) = -\log \mathbb{P}(\bar{z}|x; \theta)$$

In our setup, the model has access to $x$ and $Z = \{z_1, z_2, \ldots, z_n\}$. In this scenario, the selection of the best solution in $Z$ can be modeled as a latent variable. We can compute the maximum marginal likelihood (MML) estimate, which marginalizes the likelihood of each $z_i \in Z$ given the input $x$ with respect to $\theta$. Formally,

$$\begin{aligned}\mathbb{P}(y|x; \theta) &= \sum_{z_i \in Z_{\text{tot}}} \mathbb{P}(y|z_i)\mathbb{P}(z_i|x; \theta) \\ &= \sum_{z_i \in Z} \mathbb{P}(z_i|x; \theta)\end{aligned}$$

is used to compute the objective as follows:

$$J_{\text{MML}}(\theta|x, Z) = -\log \sum_{z_i \in Z} \mathbb{P}(z_i|x; \theta)$$

However, there are two major problems in the MML objective in our settings. First, MML can be

[2] This is a main difference from multi-instance learning (MIL) (Zhou et al., 2009), since a bag of input instances is given at inference time in MIL.

maximized by assigning high probabilities to any subset of $z_i \in Z$; whereas in our problems, instances in $Z$ other than one correct $z$ are spurious solutions which the model should ideally assign very low probability. Second, in MML we optimize the sum over probabilities of $Z$ during training but typically predict the maximum probability solution during inference, creating a discrepancy between training and testing.

We introduce a learning strategy with a hard-EM approach. First, the model computes the likelihood of each $z_i$ given the input $x$ with respect to $\theta$, $\mathbb{P}(z_i|x;\theta)$, and picks one of $Z$ with the largest likelihood:

$$\tilde{z} = \text{argmax}_{z_i \in Z} \mathbb{P}(z_i|x;\theta)$$

Then, the model optimizes on a standard negative log likelihood objective, assuming $\tilde{z}$ is a true solution. The objective can be re-written as follows:

$$\begin{aligned} J_{\text{Hard}}(\theta|x,Z) &= -\log \mathbb{P}(\tilde{z}|x;\theta) \\ &= -\log \max_{z_i \in Z} \mathbb{P}(z_i|x;\theta) \\ &= -\max_{z_i \in Z} \log \mathbb{P}(z_i|x;\theta) \\ &= \min_{z_i \in Z} J_{\text{Sup}}(\theta|x,z_i) \end{aligned}$$

This loss can be seen as a variant of MML, where the sum is replaced with a max.

## 4 Task Setup

We apply our approach to three different types of QA tasks: multi-mention reading comprehension, RC with discrete reasoning and a semantic parsing task. In this section, we describe each task in detail: how we define a solution $z$ and pre-compute a set $Z$ based on input $x$ and answer $y$. The statistics of $|Z|$ and examples on each task are shown in Table 1 and Table 2 respectively.

### 4.1 Multi-Mention Reading Comprehension

Multi-mention reading comprehension naturally occurs in several QA tasks such as (i) distantly-supervised reading comprehension where a question and answer are collected first before the evidence document is gathered (e.g. TRIVIAQA), (ii) abstractive reading comprehension which requires a free-form text to answer the question (e.g. NARRATIVEQA), and (iii) open-domain QA where only question-answer pairs are provided.

Given a question $Q = [q_1, \ldots, q_l]$ and a document $D = [d_1, \ldots, d_L]$, where $q_i$ and $d_j$ denote the tokens in the question and document, the output $y$ is an answer text, which is usually mentioned multiple times in the document.

Previous work has dealt with this setting by detecting spans in the document through text matching (Joshi et al., 2017; Clark and Gardner, 2018). Following previous approaches, we define a solution $z$ as a span in the document. We obtain a set of possible solutions $Z = \{z_1, \ldots, z_n\}$ by finding exact match or similar mentions of $y$, where $z_i = (s_i, e_i)$ is a span of text with start and end token indices $s_i$ and $e_i$. Specifically,

$$\begin{aligned} g_{\text{max}} &= \max_{1 \le s_i \le e_i \le L} g([d_{s_i}, \ldots, d_{e_i}], y) \\ Z &= \{z_i = (s_i, e_i) \text{ s.t. } g(s_i, e_i) = g_{\text{max}}\}, \end{aligned}$$

where $g$ is a string matching function. If the answer is guaranteed to be a span in the document $D$, $g$ is a binary function which returns 1 if two strings are the same, and 0 otherwise. If the answer is free-form text, we choose $g$ as the ROUGE-L metric (Lin, 2004).

This complicates the learning because the given document contains many spans matching to the text, while most of them are not related to answering the question. As an example shown in Table 2, only the fourth span out of six is relevant to the question.

### 4.2 Reading Comprehension with Discrete Reasoning

Some reading comprehension tasks require reasoning in several discrete steps by finding clues from the document and aggregating them. One such example is mathematical reasoning, where the model must pick numbers from a document and compute an answer through arithmetic operations (Dua et al., 2019).

In this task, the input is also a question $Q$ and a document $D$, and the output $y$ is given as a numeric value. We define a solution $z$ to be an executable arithmetic equation. Since there is an infinite set of potential equations, we approximate $Z_{\text{tot}}$ as a finite set of arithmetic equations with two numeric values and one operation, following Dua et al. (2019).[3] Specifically,

$$\begin{aligned} Z_{\text{tot}} = \big\{ & z_i = (o_1, n_1, o_2, n_2) \text{ s.t.} \\ & o_1, o_2 \in \{+, -, \%\}, \\ & n_1, n_2 \in N_D \cup N_Q \cup S \big\}, \end{aligned}$$

[3]This approximation covers 93% of the examples in the development set.

**1. Multi-Mention Reading Comprehension** (TRIVIAQA, NARRATIVEQA, TRIVIAQA-OPEN & NATURALQUESTIONS-OPEN)

**Question:** Which composer did pianist Clara Wieck marry in 1840?
**Document:** Robert Schumann was a German composer and influential music critic. He is widely regarded as one of the greatest composers of the Romantic era. (...) Robert Schumann himself refers to it as "an affliction of the whole hand". (...) Robert Schumann is mentioned in a 1991 episode of Seinfeld "The Jacket". (...) Clara Schumann was a German musician and composer, considered one of the most distinguished pianists of the Romantic era. Her husband was the composer Robert Schumann . <Childhood> (...) At the age of eight, the young Clara Wieck performed at the Leipzig home of Dr. Ernst Carus. There she met another gifted young pianist who had been invited to the musical evening, named Robert Schumann , who was nine years older. Schumann admired Clara' s playing so much that he asked permission from his mother to discontinue his law studies. (...) In the spring of 1853, the then unknown 20-year-old Brahms met Joachim in Hanover, made a very favorable impression on him, and got from him a letter of introduction to Robert Schumann .
**Answer ($y$):** Robert Schumann
$f$: Text match
$Z_{\text{tot}}$: All spans in the document
$Z$: Spans which match 'Robert schumann' (red text)

**2. Reading Comprehension with Discrete Reasoning** ($\text{DROP}_{\text{num}}$)

**Question:** How many yards longer was Rob Bironas' longest field goal compared to John Carney's only field goal?
**Document:** (...) The Chiefs tied the game with QB Brodie Croyle completing a `10` yard td pass to WR Samie Parker. Afterwards the Titans responded with Kicker Rob Bironas managing to get a `37` yard field goal. Kansas city would take the lead prior to halftime with croyle completing a `9` yard td pass to FB Kris Wilson. In the third quarter Tennessee would draw close as Bironas kicked a `37` yard field goal. The Chiefs answered with kicker John Carney getting a `36` yard field goal. Afterwards the Titans would retake the lead with Young and Williams hooking up with each other again on a `41` yard td pass. (...) Tennessee clinched the victory with Bironas nailing a `40` yard and a `25` yard field goal. With the win the Titans kept their playoff hopes alive at `8 6` .
**Answer ($y$):** 4
$f$: Equation executor
$Z_{\text{tot}}$: Equations with two numeric values and one arithmetic operation
$Z$: { 41-37, 40-36, 10-6, ... }

**3. SQL Generation** (WIKISQL)

**Question:** What player played guard for Toronto in 1996-1997?
**Table Header:** `player, year, position, ...`
**Answer ($y$):** John Long
$f$: SQL executor
$Z_{\text{tot}}$: Non-nested SQL queries with up to 3 conditions
$Z$:
```
Select player where position=guard and year in toronto=1996-97
Select max(player) where position=guard and year in toronto=1996-97
Select min(player) where position=guard
Select min(player) where year in toronto=1996-97
Select min(player) where position=guard and year in toronto=1996-97
```

Table 2: Examples of the input, answer text ($y$), $f$, $Z_{\text{tot}}$ and $Z$. First, in multi-mention reading comprehension, the answer text 'Robert Schumann' is mentioned six times but only the fourth span is related to the question. Second, in reading comprehension with discrete reasoning, many equations yield to the answer 4, but only '40-37' answers the question. Lastly, in SQL query generation, five SQL queries lead to the answer but only the first one is the correct query. See Section 4 for more details.

where $N_D$ and $N_Q$ are all numeric values appearing in $D$ and $Q$, respectively, and $S$ are a set of predefined special numbers. Then

$$Z = \{z_i \in Z_{\text{tot}} \text{ s.t. } f(z_i) = y\}$$

where $f$ is an execution function of equations.

Figure 1 shows an example $Z$ given a question and a document. We can see that one equation is correct, while the others are false positives which coincidentally lead to the correct answer.

### 4.3 SQL Query Generation

To evaluate if our training strategy generalizes to other weak supervision problems, we also study a semantic parsing task where a question and an answer are given but the logical form to execute the answer is not. In particular, we consider a task of answering questions about a given table by generating SQL queries.

The input is a question $Q = [q_1, \ldots, q_l]$ and a table header $H = [h_1, \ldots, h_{n_L}]$, where $q_i$ is a token, $h_i$ is a multi-token title of each column, and $n_L$ is the number of headers. The supervision is given as the SQL query result $y$, which is always a text string.

We define a solution to be an SQL query. Since the set of potential queries is infinite, we approximate $Z_{\text{tot}}$ as a set of non-nested SQL queries with at most three conditions.[4] Specifically, given $A$ as a set of aggregating operators $\{\text{sum}, \text{mean}, \text{max}, \text{min}, \text{count}\}$ and $C$ as a set of possible conditions $\{(h, o, t) \text{ s.t. } h \in [1, n_L], o \in \{=, <, >\}, t \in \text{spans in } Q\}$, we define $Z_{\text{tot}}$:

$$\begin{aligned} Z_{\text{tot}} = \{z_i &= (z_i^{\text{sel}}, z_i^{\text{agg}}, \{z_{i,j}^{\text{cond}}\}_{j=1}^{3}\} \text{ s.t.} \\ z_i^{\text{sel}} &\in [1, n_L] \\ z_i^{\text{agg}} &\in \{\text{none}\} \cup A \\ z_{i,j}^{\text{cond}} &\in \{\text{none}\} \cup C \text{ for } j \in [1, 3]\}, \end{aligned}$$

then,

$$Z = \{z_i \in Z_{\text{tot}} \text{ s.t. } f(z_i) = y\},$$

[4] This approximation covers 99% of the examples in the development set.

where $f$ is an SQL executor. The third example in Table 2 shows $Z$ may contain many spurious SQL querie, e.g. the third query in $Z$ coincidentally executes the answer because 'John Long' is ranked first among all the guards in alphabetical order.

## 5 Experiments

We experiment on a range of question answering tasks with varied model architectures to demonstrate the effectiveness of our approach. Built on top of strong base models, our learning method is able to achieve state-of-the-art on NARRATIVEQA, TRIVIAQA-OPEN, NATURALQUESTIONS-OPEN, $\text{DROP}_{\text{num}}$ and WIKISQL.

### 5.1 Multi-mention Reading Comprehension

We experiment on two reading comprehension datasets and two open-domain QA datasets. For reading comprehension, we evaluate on TRIVIAQA (Wikipedia) (Joshi et al., 2017) and NARRATIVEQA (summary) (Kočiskỳ et al., 2018).

For open-domain QA, we follow the settings in Lee et al. (2019) and use the QA pairs from TRIVIAQA-unfiltered (Joshi et al., 2017) and NATURAL QUESTIONS (Kwiatkowski et al., 2019) with short answers and discard the given evidence documents. We refer to these two datasets as TRIVIAQA-OPEN and NATURALQUESTIONS-OPEN.[5]

We experiment with three learning methods as follows.

- First Only: $J(\theta) = -\log\mathbb{P}(z_1|x;\theta)$, where $z_1$ appears first in the given document among all $z_i \in Z$.
- MML: $J(\theta) = -\log\Sigma_{i=1}^{n}\mathbb{P}(z_i|x;\theta)$.
- Ours: $J(\theta) = -\log\max_{1\le i\le n}\mathbb{P}(z_i|x;\theta)$.

$\mathbb{P}(z_i|Q,D)$ can be obtained by any model which outputs the start and end positions of the input document. In this work, we use a modified version of BERT (Devlin et al., 2019) for multi-paragraph reading comprehension (Min et al., 2019).

**Training details.** We use uncased version of $\text{BERT}_{\text{base}}$. For all datasets, we split documents into a set of segments up to 300 tokens because BERT limits the size of the input. We use batch size of 20 for two reading comprehension tasks and 192 for two open-domain QA tasks. Following Clark and Gardner (2018), we filter a subset of segments in TRIVIAQA through TF-IDF similarity between a segment and a question to maintain a reasonable length. For open-domain QA tasks, we retrieve 50 Wikipedia articles through TF-IDF (Chen et al., 2017) and further run BM25 (Robertson et al., 2009) to retrieve 20 (for train) or 80 (for development and test) paragraphs. We try 10, 20, 40 and 80 paragraphs on the development set to choose the number of paragraphs to use on the test set.

To avoid local optima, we perform annealing: at training step $t$, the model optimizes on MML objective with a probability of $\min(t/\tau, 1)$ and otherwise use our objective, where $\tau$ is a hyperparameter. We observe that the performance is improved by annealing while not being overly sensitive to the hyperparameter $\tau$. We include full hyperparameters and detailed ablations in Appendix B.

**Results.** Table 3 compares the results of baselines, our method and the state-of-the-art on four datasets.[6] First of all, we observe that First-Only is a strong baseline across all the datasets. We hypothesize that this is due to the bias in the dataset that answers are likely to appear earlier in the paragraph. Second, while MML achieves comparable result to the First-Only baseline, our learning method outperforms others by 2+ F1/ROUGE-L/EM consistently on all datasets. Lastly, our method achieves the new state-of-the-art on NARRATIVEQA, TRIVIAQA-OPEN and NATURALQUESTIONS-OPEN, and is comparable to the state-of-the-art on TRIVIAQA, despite our aggressive truncation of documents.

### 5.2 Reading Comprehension with Discrete Reasoning

We experiment on a subset of DROP (Dua et al., 2019) with numeric answers (67% of the entire dataset) focusing on mathematical reasoning. We refer to this subset as $\text{DROP}_{\text{num}}$. The current state-of-the-art model is an augmented version of QANET (Yu et al., 2018a) which selects two numeric values from the document or the question and performs addition or subtraction to get the answer. The equation to derive the answer is not

[5] Datasets and their split can be downloaded from https://bit.ly/2HK1Fqn.

[6] For NARRATIVEQA, we compare with models trained on NARRATIVEQA only. For open-domain QA, we only compare with models using pipeline approach.

| | TriviaQA (F1) | | NarrativeQA (ROUGE-L) | | TriviaQA -open (EM) | | NaturalQ -open (EM) | | $\text{DROP}_{\text{num}}$ w/ BERT (EM) | $\text{DROP}_{\text{num}}$ w/ QANet (EM) |
|---|---|---|---|---|---|---|---|---|---|---|
| | dev | test | dev | test | dev | test | dev | test | dev | dev |
| First Only | 64.4 | 64.9 | 55.3 | 57.4 | 48.6 | 48.1 | 23.6 | 23.6 | 42.9 | 36.1 |
| MML | 64.8 | 65.5 | 55.8 | 56.1 | 47.0 | 47.4 | 26.6 | 25.8 | 39.7 | 43.8 |
| Ours | 66.9 | 67.1 | **58.1** | **58.8** | **50.7** | **50.9** | **28.8** | **28.1** | **52.8** | **45.0** |
| SOTA | - | **71.4** | - | 54.7 | 47.2 | 47.1 | 24.8 | 26.5 | 43.8 | |

Table 3: **Results on multi-mention reading comprehension & discrete reasoning tasks.** We report performance on five datasets with different base models. Note that we are not able to obtain the test result on the subset $\text{DROP}_{\text{num}}$. Previous state-of-the-art are from Wang et al. (2018), Nishida et al. (2019), Lee et al. (2019), Lee et al. (2019) and Dua et al. (2019), respectively. Our training method consistently outperforms the First-Only and MML by a large margin in all the scenarios.

| Model | Accuracy dev | Accuracy test |
|---|---|---|
| *Weakly-supervised setting* | | |
| REINFORCE (Williams, 1992) | $< 10$ | |
| Iterative ML (Liang et al., 2017) | 70.1 | |
| Hard EM (Liang et al., 2018) | 70.2 | |
| Beam-based MML (Liang et al., 2018) | 70.7 | |
| MAPO (Liang et al., 2018) | 71.8 | 72.4 |
| MAPOX (Agarwal et al., 2019) | 74.5 | 74.2 |
| MAPOX+MeRL (Agarwal et al., 2019) | 74.9 | 74.8 |
| MML | 70.6 | 70.5 |
| Ours | **84.4** | **83.9** |
| *Fully-supervised setting* | | |
| SQLNet (Xu et al., 2018) | 69.8 | 68.0 |
| TypeSQL (Yu et al., 2018b) | 74.5 | 73.5 |
| Coarse2Fine (Dong and Lapata, 2018) | 79.0 | 78.5 |
| SQLova (Hwang et al., 2019) | 87.2 | 86.2 |
| X-SQL (He et al., 2019) | **89.5** | **88.7** |

Table 4: **Results on WikiSQL.** We compare accuracy with weakly-supervised or fully-supervised settings. Our method outperforms previous weakly-supervised methods and most of published fully-supervised methods.

given and Dua et al. (2019) adopted the MML objective.

$\mathbb{P}(z_i|Q, D)$ can take as any model which generates equations based on the question and document. Inspired by Dua et al. (2019), we take a sequence tagging approach on top of two competitive models: (i) augmented QANet, the same model as Dua et al. (2019) but only supporting addition, subtraction and counting, and (ii) augmented BERT, which supports addition, subtraction and percentiles.[7]

**Training details.** We truncate the document to be up to 400 words. We use the batch size of 14 and 10 for QANet and BERT, respectively.

**Results.** Table 3 shows the results on $\text{DROP}_{\text{num}}$. Our training strategy outperforms the First-Only baseline and MML by a large margin, consistently across two base models. In particular, with BERT, we achieve an absolute gain of 10%.

### 5.3 SQL Query Generation

Finally, we experiment on the weakly-supervised setting of WikiSQL (Zhong et al., 2017), in which only the question & answer pair is used and the SQL query $(z)$ is treated as a latent variable.

$\mathbb{P}(z_i|Q, H)$ can be computed by any query generation or semantic parsing models. We choose SQLova (Hwang et al., 2019), a competitive model on WikiSQL (designed for fully supervised setting), as our base model. We modify the model to incorporate either the MML objective or our hard-EM learning approach for the weakly-supervised setting.

We compare with both traditional and recently-developed reward-based algorithms for weak supervision, including beam-based MML (MML which keeps a beam during training), conventional hard EM[8], REINFORCE (Williams, 1992), iterative ML (Liang et al., 2017; Abolafia et al., 2018) and a family of MAPO (Memory-augmented policy optimization) (Liang et al., 2018; Agarwal et al., 2019). For a fair comparison, we only consider single models without execution-guided decoding.

**Training details.** We adopt the same set of hyperparameters as in Hwang et al. (2019), except that we change the batch size to 10 and truncate the input to be up to 180 words for memory efficiency.

[7] As we use a different set of operations for the two models, they are not directly comparable. Details of the model architecture are shown in Appendix A.

[8] This method differs from ours in that it does not have a precomputed set, and uses a beam of candidate predictions to execute for each update.

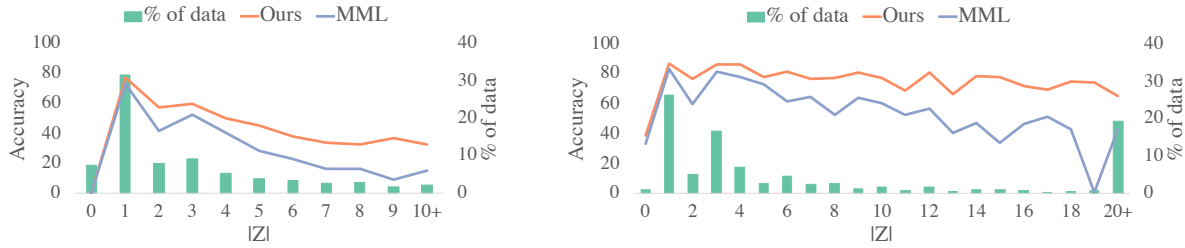


(a) $\text{DROP}_{\text{num}}$ (b) WikiSQL

Figure 2: **Varying the size of solution set ($|Z|$) at test time.** We compare the model trained on MML objective (blue) and our training strategy (orange). Our approach consistently outperforms MML on $\text{DROP}_{\text{num}}$ and WIKISQL, especially when $|Z|$ is large.

| Group | Avg $\lvert Z\rvert$ | Median $\lvert Z\rvert$ | # train |
|---|---|---|---|
| 3 | 3.0 | 3 | 10k |
| 10 | 10.2 | 9 | 10k |
| 30 | 30.0 | 22 | 10k |
| 100 | 100.6 | 42 | 10k |
| 300 | 300.0 | 66 | 10k |




Figure 3: **Varying the size of solution set ($|Z|$) at training.** (Left) Subsets of the train set on WIKISQL varying in the size of solution set ($|Z|$). All subsets contain 10k training examples (total in the original train set is 55k). All subsets are evaluated on the same, original development set for a fair comparison. (Right) Performance across subsets of the training set with varying $|Z|$. Our method achieves substantial gains over MML.

**Results.** Table 4 shows that our training method significantly outperforms all the weakly-supervised learning algorithms, including 10% gain over the previous state of the art. These results indicate that precomputing a solution set and training a model through hard updates play a significant role to the performance. Given that our method does not require SQL executions at training time (unlike MAPO), it provides a simpler, more effective and time-efficient strategy. Comparing to previous models with full supervision, our results are still on par and outperform most of the published results.

## 6 Analysis

In this section, we will conduct thorough analyses and ablation studies, to better understand how our model learns to find a solution from a precomputed set of possible solutions. We also provide more examples and analyses in Appendix C.

**Varying the size of solution set at inference time.** Figure 2 shows a breakdown of the model accuracy with respect to the size of a solution set ($|Z|$) at test time. We observe that the model with our training method outperforms the model with MML objective consistently across different values of $|Z|$. The gap between MML and our method is marginal when $|Z| = 0$ or 1, and gradually increases as $|Z|$ grows.

**Varying the size of solution set at training.** To see how our learning method works with respect to the size of a solution set ($|Z|$) of the training data, particularly with large $|Z|$, we take 5 subsets of the training set on WIKISQL with $|Z| = 3, 10, 30, 100, 300$. We train a model with those subsets and evaluate it on the original development set, both with our training method and MML objective. Figure 3 shows statistics of each subset and results. We observe that (i) our learning method outperforms MML consistently over different values of $|Z|$, and (ii) the gain is particularly large when $|Z| > 3$.

**Model predictions over training.** We analyze the top 1 prediction and the likelihood of $z \in Z$ assigned by the model on $\text{DROP}_{\text{num}}$ with different number of training iterations (steps from 1k to 32k). Table 5 shows one example on $\text{DROP}_{\text{num}}$ with the answer text '4', along with the model's top 1 prediction and a subset of $Z$. We observe that

**Q:** How many yards longer was Rob Bironas' longest field goal compared to John Carney's only field goal? (**Answer:** 4)
**P:** ... The Titans responded with Kicker Rob Bironas managing to get a 37 yard field goal. ...Tennessee would draw close as Bironas kicked a 37 yard field goal. The Chiefs answered with kicker John Carney getting a 36 yard field goal. The Titans would retake the lead with Young and Williams hooking up with each other again on a 41 yard td pass. ...Tennessee clinched the victory with Bironas nailing a 40 yard and a 25 yard field goal.

| $t$ | Pred | $Z$ (ordered by $\mathbb{P}(z\|x;\theta_t)$) | | | |
|---|---|---|---|---|---|
| 1k | 10-9 | 10-6 | 41-37 | 40-36 | 41-37‡ |
| 2k | 37-36 | 40-36 | 41-37 | 41-37‡ | 10-6 |
| 4k | 40-36 | 40-36 | 41-37‡ | 41-37 | 10-6 |
| 8k | 40-36 | 40-36 | 41-37‡ | 41-37 | 10-6 |
| 16k | 37-36 | 40-36 | 41-37 | 41-37‡ | 10-6 |
| 32k | 40-36 | 40-36 | 41-37 | 41-37‡ | 10-6 |

Table 5: An example from $\text{DROP}_{\text{num}}$ (same as Figure 1 and Table 2), with its answer text '4' and a subset of the solution set ($Z$), containing two of '41-38' (which '41' come from different mentions; one denoted by ‡ for distinction), '40-36' and '10-4'. For each training step $t$, the top 1 prediction and $Z$ ordered by $P(z|x;\theta_t)$, a probability of $z \in Z$ with respect to the model at $t$ through training procedure are shown. Note that at inference time $Z$ is not given, so top 1 prediction is not necessarily an element of $Z$.

the model first begins by assigning a small, uniform probability distribution to $Z$, but gradually learns to favor the true solution. The model sometimes gives the wrong prediction—for example, at $t = 16$k, and changes its prediction from the true solution to the wrong solution, '37-36'—but again changes its prediction to be a true solution afterward. In addition, its intermediate wrong solution, '37-36' indicates the model was confused with distinguishing the longest field goal of Rob Bironas (40 vs. 37), which is an understandable mistake.

We also compare the predictions from the model with our method to those from the model with MML, which is shown in Appendix C.

**Quality of the predicted solution.** We analyze if the model outputs the correct solution, since the solution executing the correct answer could be spurious. First, on NARRATIVEQA and $\text{DROP}_{\text{num}}$, we manually analyze 50 samples from the development set and find that 98% and 92% of correct cases produce the correct solution respectively. Next, on WIKISQL, we compare the predictions from the model to the annotated SQL queries on the development set. This is possible because gold SQL queries are available in the dataset for the full supervision. Out of 8,421 examples, 7,110 predictions execute the correct answers. Among those, 88.5% of the predictions are exactly same as the annotated queries. Others are the cases where (i) both queries are correct, (ii) the model prediction is correct but the annotated query is incorrect, and (iii) the annotated query is correct and the model prediction is spurious. We show a full analysis in Appendix C.

**Robustness to the noise in $|Z|$.** Sometimes noise arises during the construction of $|Z|$, such as $|Z|$ constructed based on ROUGE-L for NARRATIVEQA. To explore the effect of noise in $Z$, we experiment with more noisy solution set by picking all the spans with scores that is equal to or larger than the 5th highest. The new construction method increases $|Z|$ from 4.3 to 7.1 on NARRATIVEQA. The result by MML objective drops significantly (56.07→ 51.14) while the result by ours drops marginally (58.77→ 57.97), suggesting that MML suffers more with a noisier $Z$ while ours is more robust.

## 7 Conclusion

In this paper, we demonstrated that, for many QA tasks which only provide the answer text as supervision, it is possible to precompute a discrete set of possible solutions that contains one correct option. Then, we introduced a discrete latent variable learning algorithm which iterates a procedure of predicting the most likely solution in the precomputed set and further increasing the likelihood of that solution. We showed that this approach significantly outperforms previous approaches on six QA tasks including reading comprehension, open-domain QA, discrete reasoning task and semantic parsing, achieving absolute gains of 2–10% and setting the new state-of-the-art on five well-studied datasets.


### Acknowledgements

This research was supported by ONR (N00014-18-1-2826, N00014-17-S-B001), DARPA N66001-19-2-403, NSF (IIS-1616112, IIS-1252835, IIS-1562364), ARO (W911NF-16-1-0121), an Allen Distinguished Investigator Award, Samsung GRO and gifts from Allen Institute for AI, Google and Amazon.

The authors would like to thank the anonymous reviewers, Eunsol Choi, Christopher Clark, Victor Zhong and UW NLP members for their valuable feedback.

## References


Daniel A Abolafia, Mohammad Norouzi, Jonathan Shen, Rui Zhao, and Quoc V Le. 2018. Neural program synthesis with priority queue training. *arXiv preprint arXiv:1801.03526*.

Rishabh Agarwal, Chen Liang, Dale Schuurmans, and Mohammad Norouzi. 2019. Learning to generalize from sparse and underspecified rewards. In *ICML*.

Chris Alberti, Kenton Lee, and Michael Collins. 2019. A BERT baseline for the Natural Questions. *arXiv preprint arXiv:1901.08634*.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. In *ACL*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *EMNLP*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *ACL*.

Christopher Clark and Matt Gardner. 2018. Simple and effective multi-paragraph reading comprehension. In *ACL*.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *CoNLL*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In *ACL*.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *NAACL*.

Pengcheng He, Yi Mao, Kaushik Chakrabarti, and Weizhu Chen. 2019. X-SQL: reinforce schema representation with context. *arXiv preprint arXiv:1908.08113*.

Niall Hurley and Scott Rickard. 2009. Comparing measures of sparsity. *IEEE Transactions on Information Theory*.

Wonseok Hwang, Jinyeung Yim, Seunghyun Park, and Minjoon Seo. 2019. A comprehensive exploration on WikiSQL with table-aware word contextualization. *arXiv preprint arXiv:1902.01069*.

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. In *ACL*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL*.

Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. In *ACL*.

Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The NarrativeQA reading comprehension challenge. *TACL*.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *EMNLP*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Change, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: a benchmark for question answering research. *TACL*.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *ACL*.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on Freebase with weak supervision. In *ACL*.

Chen Liang, Mohammad Norouzi, Jonathan Berant, Quoc V Le, and Ni Lao. 2018. Memory augmented policy optimization for program synthesis and semantic parsing. In *NIPS*.

Percy Liang, Michael I Jordan, and Dan Klein. 2013. Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2):389–446.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Sewon Min, Eric Wallace, Sameer Singh, Matt Gardner, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2019. Compositional questions do not necessitate multi-hop reasoning. In *ACL*.

Kyosuke Nishida, Itsumi Saito, Kosuke Nishida, Kazutoshi Shinoda, Atsushi Otsuka, Hisako Asano, and Junji Tomita. 2019. Multi-style generative reading comprehension. In *ACL*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *EMNLP*.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *ICLR*.

Swabha Swayamdipta, Ankur P Parikh, and Tom Kwiatkowski. 2018. Multi-mention learning for reading comprehension with neural cascades. In *ICLR*.

Alon Talmor and Jonathan Berant. 2019. MultiQA: An empirical investigation of generalization and transfer in reading comprehension. In *ACL*.

Yi Tay, Anh Tuan Luu, Siu Cheung Hui, and Jian Su. 2018. Densely connected attention propagation for reading comprehension. In *NIPS*.

Wei Wang, Ming Yan, and Chen Wu. 2018. Multi-granularity hierarchical attention fusion networks for reading comprehension and question answering. In *ACL*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Caiming Xiong, Victor Zhong, and Richard Socher. 2018. DCN+: Mixed objective and deep residual coattention for question answering. In *ICLR*.

Xiaojun Xu, Chang Liu, and Dawn Song. 2018. SQLNet: Generating structured queries from natural language without reinforcement learning. In *ICLR*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*.

Adams Wei Yu, David Dohan, Quoc Le, Thang Luong, Rui Zhao, and Kai Chen. 2018a. Fast and accurate reading comprehension by combining self-attention and convolution. In *ICLR*.

Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. 2018b. TypeSQL: Knowledge-based type-aware neural text-to-sql generation. In *NAACL*.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

Zhi-Hua Zhou, Yu-Yin Sun, and Yu-Feng Li. 2009. Multi-instance learning by treating instances as non-i.i.d. samples. In *ICML*.

## A Model details

We describe the detailed model architecture used as a base model. In other words, we describe how we obtain $\mathbb{P}(z|x;\theta)$.

The following paragraphs describe (i) BERT extractive model used for multi-mention RC (Section 5.1) and (ii) BERT sequence tagging model for discrete reasoning task (Section 5.2), respectively. For QANET for discrete reasoning and the model for SQL generation (Section 5.3), we use the open-sourced code of the original implementation[9] of Dua et al. (2019) and Hwang et al. (2019) and do not make any modification except the objective function, so we refer to original papers.

All implementations are done in Pytorch (Paszke et al., 2017). For BERT, we modify the open-sourced implementation in PyTorch[10] and use the uncased version of $\text{BERT}_{\text{base}}$.

**Extractive QA model for multi-mention RC** The model architecture is closed to that of Min et al. (2019) and Alberti et al. (2019), where the model operates independently on each paragraph, and selects the best matching paragraph and its associated answer span.

The input is a question $Q$ and a set of paragraphs $\{P_1, \ldots, P_N\}$, and the desired output is a span from one of paragraphs. Since our goal is to compute a probability of a specific span, $z$, let's say $z$ is $s$-th through $e$-th word in $k$-th paragraph.

The model receives a question $Q$ and a single paragraph $P_i$ in parallel. Then, $S_i = Q : [\texttt{SEP}] : P$, a list of $m + n_i + 1$ words, where : indicates a concatenation, [SEP] is a special token, $m$ is the length of $Q$, and $n_i$ is the length of $P_i$. This $S_i$ is fed into BERT:

$$\bar{S}_i = \text{BERT}(S) \in \mathbb{R}^{h\times(m+n_i+1)}$$

where $h$ is the hidden dimension of BERT. Then,

$$\begin{aligned} p_{i,\text{start}} &= \text{Softmax}\big(\bar{S}_i^T W_1\big) \in \mathbb{R}^{m+n_i+1} \\ p_{i,\text{end}} &= \text{Softmax}\big(\bar{S}_i^T W_2\big) \in \mathbb{R}^{m+n_i+1} \end{aligned}$$

where $W_1, W_2 \in \mathbb{R}^h$ are learnable vectors.

Finally, the probability of $z$, $s$-th through $e$-th word in $i$-th paragraph, is obtained by:

$$\mathbb{P}(z|Q,P_i) = p^s_{i,\text{start}} \times p^e_{i,\text{end}}$$

where $p^d$ denotes $d$-th element of the vector $p$.

Separately, a paragraph selector is trained through $p_{i,\text{exit}} = \text{Softmax}\big(W_3 \text{maxpool}(\bar{S}_i)\big) \in \mathbb{R}^2$ where $W_3 \in \mathbb{R}^{h\times 2}$ is learnable vector. At inference time, $k = \text{argmax}_i p^1_{i,\text{exit}}$ is computed and $\mathbb{P}(z|Q,P_k)$ is only considered to output a span.

**Sequence Tagging model for discrete reasoning** The basic idea of the model is closed to that of Dua et al. (2019). The input is a question $Q$ and a paragraph $P$. Our goal is to compute a probability of an equation, $z = (o_1, n_1, o_2, n_2)$, where $o_1, o_2 \in \{+, -, *0.01\}$ and $n_1, n_2 \in N_P \cup N_Q \cup S$, $N_P$ and $N_Q$ are all numeric values appearing in $P$ and $Q$, and $S$ are a set of predefined special numbers.[11].

First, BERT encodings of the question and the paragraph is obtained via

$$\bar{S} = \text{BERT}(Q : [\texttt{SEP}] : P) \in \mathbb{R}^{h\times(m+n+1)}$$

where : indicates a concatenation, [SEP] is a special token, $m$ is the length of $Q$, $n$ is the length of $P$, and $h$ is the hidden dimension of BERT. Then,

$$\begin{aligned} p_{\text{input}} &= \text{Softmax}\big(\bar{S}^T W_1\big) \in \mathbb{R}^{(m+n+1)\times 4} \\ p_{\text{special}} &= \text{Softmax}\big(\text{maxpool}(\bar{S}) W_2\big) \in \mathbb{R}^{|S|\times 4} \end{aligned}$$

where $W_1 \in \mathbb{R}^{h\times 4}$ and $W_2 \in \mathbb{R}^{h\times|S|\times 4}$ are learnable matrices. Then,

$$\mathbb{P}(z|x) = \Pi_{i=1}^{m+n+1} p^{g(i)}_{\text{input}} \Pi_{j=1}^{|S|} p^{h(j)}_{\text{special}}$$

where

$$g(i) = \begin{cases} \alpha(o_1) & \text{if } [Q : [\texttt{SEP}] : P]^i = n_1 \\ \alpha(o_2) & \text{if } [Q : [\texttt{SEP}] : P]^i = n_2 \\ 0 & \text{o.w.} \end{cases}$$

$$h(j) = \begin{cases} \alpha(o_1) & \text{if } S^j = n_1 \\ \alpha(o_2) & \text{if } S^j = n_2 \\ 0 & \text{o.w.} \end{cases}$$

$$\alpha(o) = \begin{cases} 1 & \text{if } o = \text{`+'} \\ 2 & \text{if } o = \text{`-'} \\ 3 & \text{if } o = \text{`*0.01'} \end{cases}$$

Here, subscript $i$ of the vector or the sequence indicate $i$-th dimension of the vector or $i$-th element of the sequence, respectively.

[9] https://github.com/allenai/allennlp/blob/master/allennlp/models/reading_comprehension/naqanet.py and https://github.com/naver/sqlova

[10] https://github.com/huggingface/pytorch-pretrained-BERT

[11] $S = \{1, 2, 3, 4, 5, 7, 10, 12, 100, 1000\}$

| $\tau$ | TRIVIAQA F1 |
|---|---|
| None | 55.83 |
| 20k | 58.05 |
| 30k | 57.99 |
| 40k | 56.66 |

| $\tau$ | $\text{DROP}_{\text{num}}$ EM |
|---|---|
| None | 52.31 |
| 5k | 51.98 |
| 10k | 52.82 |
| 20k | 51.74 |

| Dataset | batch size | $\tau$ |
|---|---|---|
| TRIVIAQA | 20 | 15K |
| NARRATIVEQA | 20 | 20K |
| TRIVIAQA-OPEN | 192 | 4K |
| NATURALQUESTIONS-OPEN | 192 | 8K |
| $\text{DROP}_{\text{num}}$ with BERT | 14 | 10K |
| $\text{DROP}_{\text{num}}$ with QANET | 14 | None |
| WIKISQL | 10 | None |

Table 6: **(Left)** Ablations with varying values of $\tau$ on TRIVIAQA. **(Middle)** Ablations with varying values of $\tau$ on $\text{DROP}_{\text{num}}$ with BERT. **(Right)** Final $\tau$ chosen for the main results on each dataset. Note that for $\text{DROP}_{\text{num}}$ with QANET and WIKISQL, we just report the number without annealing.

## B Annealing

To prevent the model to be optimized on early decision of the model, we perform annealing: at training step $t$, the model optimizes on MML objective with a probability of $\min(t/\tau, 1)$ and otherwise use our objective, where $\tau$ is a hyperparameter. We observe that the performance is improved by annealing while not being sensitive to the hyperparameter $\tau$. Ablations and chosen $\tau$ for each dataset are shown in Table 6. Note that for $\text{DROP}_{\text{num}}$ with QANET and WIKISQL, we do not ablate with varying $\tau$ and just report the number without annealing.

## C Examples

To see if the prediction from the model is the correct solution to derive the answer, we analyze outputs from the model.

**TRIVIAQA.** Table 7 shows one example from TRIVIAQA where the answer text (Montgomery) is mentioned in the paragraph multiple times. Predictions from the model with our training method and that with MML objective are shown in the red text and the blue text, respectively. The span predicted by the model with our method actually answers to the question, while other spans with the answer text is not related to the question.

**$\text{DROP}_{\text{num}}$.** Table 8 shows predictions from the model with our method and that with MML objective over training procedure. We observe that the model with our method learns to assign a high probability to the best solution ('34-24'), while the model with MML objective fails to do so. Another notable observation is that the model with our method assign sparse distribution of likelihood over $Z$, compared to the model with MML objective. We quantitatively define sparsity as

$$\frac{|\{z \in Z \text{ s.t. } \mathbb{P}(z|x;\theta) < \epsilon\}|}{|Z|}$$

(Hurley and Rickard, 2009) and show that the model with our method gives higher sparsity than the model with MML (59 vs. 36 with $\epsilon = 10^{-3}$, 54 vs. 17 with $\epsilon = 10^{-4}$ on DROP).

**WIKISQL.** WIKISQL provides the annotated SQL queries, makes it easy to compare the predictions from the model to the annotated queries. Out of 8421 examples from the development set, 7110 predictions execute the correct answers. Among those, 6296 predictions are exactly same as the annotated queries. For cases where the predictions execute the correct answers but are not exactly same as the groundtruth queries, we show four examples in Table 9. In the first example, both the annotated query and the prediction are correct, because the selected column does not matter for counting. Similarly in the second example, both queries are correct because `Capital (exonym)` and `Capital (endonym)` both indicate the capital city. In the third example, the prediction makes more sense than the annotated query because the question does not imply anything about `min`. In the last example, the annotated query makes more sense than the prediction because the prediction misses `Ship Type = battleship`. We conjecture that the model might learn to ignore some information in the question if the table header implies the table is specific about that information, hence does not need to condition on that information.

| Question & Document |
|---|
| **Q:** What is the state capital of Alabama? (Groundtruth: Montgomery)<br>**D:** Alabama is nicknamed the Yellowhammer State, after the state bird. Alabama is also known as the "Heart of Dixie" and the "Cotton State". The state tree is the longleaf pine, and the state flower is the camellia. Alabama's capital is Montgomery. (...) From 1826 to 1846, Tuscaloosa served as Alabama's capital. On January 30, 1846, the Alabama legislature announced it had voted to move the capital city from Tuscaloosa to Montgomery. The first legislative session in the new capital met in December 1847. A new capitol building was erected under the direction of Stephen Decatur Button of Philadelphia. The first structure burned down in 1849, but was rebuilt on the same site in 1851. This second capitol building in Montgomery remains to the present day. |

Table 7: An example from TRIVIAQA with multiple spans of the answer text (underlined). The model trained with self-training technique outputs the correct answer (red) and the model trained on MML objective does not (blue).

| Question & Passage |
|---|
| **Q:** How many sports are not olympic sports but are featured in the asian games ? (A: 10)<br>**P:** The first 30 sports were announced by the singapore national olympic council on 10 december 2013 on the sidelines of the 27th sea games in myanmar. It announced then that there was room for as many as eight more sports. On 29 april 2014 the final six sports namely boxing equestrian floorball petanque rowing and volleyball were added to the programme. Floorball will feature in the event for the first time after being a demonstration sport in the 2013 edition. In its selection of events the organising committee indicated their desire to set a model for subsequent games in trimming the number of 'traditional' sports to refocus on the seag' s initial intent to increase the level of sporting excellence in key sports. Hence despite room for up to eight traditional sports only two floorball and netball were included in the programme. Amongst the other 34 sports 24 are olympic sports and all remaining sports are featured in the asian games. |

***Ours***

| $t$ | Pred | $Z$ (ordered by $\mathbb{P}(z\|x;\theta_t)$) | | | |
|---|---|---|---|---|---|
| 1k | 10+two | eight+two | eight+two‡ | 34-24 | 10 |
| 2k | 30-24 | 34-24 | eight+two | eight+two‡ | 10 |
| 4k | 34+24 | 34-24 | eight+two | eight+two‡ | 10 |
| 8k | 34+24 | 34-24 | eight+two | 10 | eight+two‡ |
| 16k | 34-24 | 34-24 | eight+two | eight+two‡ | 10 |
| 32k | 34-24 | 34-24 | eight+two | eight+two‡ | 10 |

***MML***

| $t$ | Pred | $Z$ (ordered by $\mathbb{P}(z\|x;\theta_t)$) | | | |
|---|---|---|---|---|---|
| 1K | 10+two | eight+two | eight+two‡ | 10 | 34-24 |
| 2k | eight+two | eight+two | eight+two‡ | 34-24 | 10 |
| 4k | 24+5 | eight+two | eight+two‡ | 34-24 | 10 |
| 8k | 24+5 | eight+two | eight+two‡ | 34-24 | 10 |
| 16k | 34-24 | 34-24 | eight+two | eight+two‡ | 10 |
| 32k | 24+5 | 34-24 | eight+two | eight+two‡ | 10 |

Table 8: An example from $\text{DROP}_{\text{num}}$, with its answer text '10' and a subset of $Z$, containing '10', two of 'eight+two' (which 'eight' come from different mentions; one denoted by '‡' for distinction) and '34-24'. The below tables are predictions from the model with our training strategy (left) and MML (right). For each training step $t$, the top 1 prediction and $Z$ ordered by $P(z|x;\theta_t)$, a probability of $z \in Z$ with respect to the model at $t$ are shown. Note that at inference time $Z$ cannot be obtained, so top 1 prediction is not necessarily in $Z$.

| | |
|---|---|
| **Q** | How many times was the # of total votes 2582322? |
| **H** | Election, # of candidates nominated, # of seats won, # of total votes, % of popular vote |
| **A** | `Select count(# of seats won) where # of total votes = 2582322` |
| **P** | `Select count(Election) where # of total votes = 2582322` |
| **Q** | What official or native languages are spoken in the country whose capital city is Canberra? |
| **H** | Country (exonym), Capital (exonym), Country (endonym) Capital (endonym), Official or native language |
| **A** | `Select Official or native languages where Capital (exonym) = Canberra` |
| **P** | `Select Official or native languages where Capital (endonym) = Canberra` |
| **Q** | What is the episode number that has production code 8abx15? |
| **H** | No. in set, No. in series, Title, Directed by, Written by, Original air date, Production code |
| **A** | `Select min(No.in series) where Production code = 8ABX15` |
| **P** | `Select No.in series where Production code = 8abx15` |
| **Q** | what is the name of the battleship with the battle listed on May 13, 1915? |
| **H** | Estimate, Name, Nat., Ship Type, Principal victims, Date |
| **A** | `Select Name where Ship Type = battleship and Date = may 13, 1915` |
| **P** | `Select Name where Date = may 13, 1915` |

Table 9: Four examples from WIKISQL where the prediction from the model is different from annotated SQL query, although the executed answers are the same. **Q**, **H**, **A** and **P** indicate the given question, the given table header, annotated SQL query and predicted SQL query. First two example shows the case where both queries are correct. Next example shows the case that the model prediction makes more sense than the annotated query. The last example shows the cases that the annotated query makes more sense than the model prediction.

# Is the *Red Square* Big?
# MALeViC: Modeling Adjectives Leveraging Visual Contexts

**Sandro Pezzelle** and **Raquel Fernández**
Institute for Logic, Language, and Computation
University of Amsterdam
{s.pezzelle|raquel.fernandez}@uva.nl

## Abstract

This work aims at modeling how the meaning of gradable adjectives of *size* ('big', 'small') can be learned from visually-grounded contexts. Inspired by cognitive and linguistic evidence showing that the use of these expressions relies on setting a threshold that is dependent on a specific *context*, we investigate the ability of multi-modal models in assessing whether an object is 'big' or 'small' in a given visual scene. In contrast with the standard computational approach that simplistically treats gradable adjectives as 'fixed' attributes, we pose the problem as *relational*: to be successful, a model has to consider the full visual context. By means of four main tasks, we show that state-of-the-art models (but not a relatively strong baseline) can learn the function subtending the meaning of size adjectives, though their performance is found to decrease while moving from simple to more complex tasks. Crucially, models fail in developing abstract representations of gradable adjectives that can be used compositionally.

## 1 Introduction

There is no doubt that planets are *big* things. Among the planets of our Solar System, however, Mars is unquestionably a *small* planet (though not the smallest), while Saturn is definitely a *big* one (though not the biggest). This example highlights some crucial properties of gradable adjectives (hence, GAs). First, what counts as 'big' or 'small' is *relative*, i.e., determined by the context: Phobos is both a *big* moon of Mars and a *small* celestial body of the Solar System. This makes GAs different from non-gradable or *absolute* adjectives like 'open', 'empty', 'red': Mars, for example, is *red* and *rocky* in any circumstance. More formally, the compositional semantic properties of GAs are *subsective* since they select a subset of entities denoted by the noun they modify, which acts as a *reference set* ($\|\text{big}\| \subseteq \|\text{moon}\|$), while non-GAs are *intersective* ($\|\text{Galilean}\| \cap \|\text{moon}\|$). This has consequences for the inferences they license: if Ganymede is both a *Galilean* moon and a celestial body, we can infer that Ganymede is a Galilean celestial body. In contrast, if it is a *big* moon and a celestial body, the inference that Ganymede is a big celestial body is not valid (Partee, 1995).

Second, besides depending on a contextually-given reference set, GAs rely on orderings, i.e, they denote functions that map entities onto scales of degrees (Cresswell, 1976; Kennedy, 1999). Using 'big' or 'small', thus, implies mapping a target object onto a size scale, which allows us to use degree morphology to express that Saturn is *bigger* than Mars (comparative form) or that Mercury is the *smallest* planet (superlative form). As for the non-inflected, so-called positive form of GAs (e.g., Saturn is a *big* planet), its interpretation involves applying a *statistical function* that makes use of a standard threshold degree (Kamp, 1975; Pinkal, 1979; Barker, 2002; Kennedy, 2007).

Third, GAs are considered to be *vague*, because whether they apply or not to a given entity can be a matter of debate among speakers (Van Deemter, 2012; Lassiter and Goodman, 2017). Since people might rely on slightly different functions involving probabilistic thresholds, there are often *borderline cases*: e.g., Neptune (i.e., the fourth planet out of eight in terms of size) could be considered as a *big* planet by most but not all speakers in all situations.

Our aim in this work is to computationally learn the meaning of size GAs ('big', 'small') from visually-grounded contexts. Based on the semantic properties of such expressions (context-dependence, statistically-defined interpretation, vagueness), we tackle the task as a *relational* problem in the domain of visual reasoning (similarly, e.g., to spatial problems like assessing whether 'X is on the left of Y'). Simply put, a model needs



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2865–2876,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

to consider the entire visual context (not just the queried object) in order to solve the task. Such setup resembles experimental paradigms in developmental psychology which test how children interpret GAs when applied to objects grounded in visual scenes (Barner and Snedeker, 2008). Evidence shows that children learn to use GAs compositionally early on: when asked to assess whether an object is 'tall' or 'short' in a visual context, 4-year-old children are able to (a) restrict the reference set by means of linguistic cues and (b) derive a tall/short threshold relative to that set. That is, 4-year-olds do not interpret GAs categorically but *compositionally*. This is radically different from how adjectives are treated in current visual reasoning approaches, which consider them static labels standing for attributes (size, color, material) whose value is fixed across contexts (Johnson et al., 2017a; Santoro et al., 2017): i.e., for current models, Saturn is always *big* and Mars is always *small*.

To model GAs in a relational fashion, we rely on a statistical function that is found to be best predictive of human interpretations (Schmidt et al., 2009) and build **MALeViC**,[1] a battery of datasets for **M**odeling **A**djectives **Le**veraging **Vi**sual **C**ontexts (see Figure 1). Each dataset, including 20K synthetic visual scenes and automatically-generated language descriptions (e.g., 'the red square is a *big* square'), is used to test different abilities. We experiment with several models and show that FiLM (Perez et al., 2018) and, to a lesser extent, Stacked Attention Networks (Yang et al., 2016) can learn the function subtending the meaning of size adjectives, though their performance is found to decrease while moving from simple to more complex tasks. Crucially, all models fail in developing abstract representations of gradable adjectives that can be used compositionally.

## 2 Related Work

**Computational Linguistics** Computational approaches to GAs have mostly focused on automatically ordering elements with respect to their intensity (e.g., *good*<*great*<*excellent*, de Marneffe et al., 2010) to overcome a problem with lexical resources like WordNet (Fellbaum, 1998), which

[1]The name is inspired by that of the Russian 20th-century artist Kazimir Malevič (or Malevich), famous for his paintings of geometric shapes, such as the *Red Square*. Datasets, code, and trained models can be found here: https://github.com/sandropezzelle/malevic

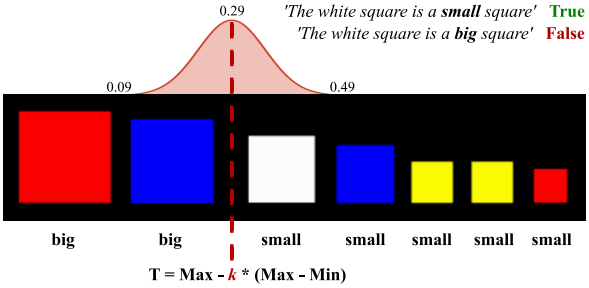


Figure 1: Our approach: given a visual **context** depicting several objects and a **sentence** about one object size, models have to assess whether the sentence is true or false. Ground-truth answers are *context*-specific and depend on a threshold T (Schmidt et al., 2009) which counts objects as 'big' based on: (1) the area of the biggest (Max) and smallest (Min) object in the context; (2) the value of a 'vague' $k$ determining the top $k$% of sizes which count as 'big'. In our approach, all objects that are *not* 'big' count as 'small'. Best viewed in color.

consider words like 'small' and 'minuscule' as synonyms. These efforts showed the potential of using techniques based on word embeddings (Kim and de Marneffe, 2013), web-scale data (Sheinman et al., 2013; De Melo and Bansal, 2013), or their combination (Shivade et al., 2015; Kim et al., 2016) to determine the relative intensity of different words on a scale. By focusing on the *ordering* between adjectives, however, these works do not shed light on how the *meaning* of such expressions is determined by contextual information. This goal, in contrast, has been pursued by work on automatic Generation of Referring Expressions (GRE), where GAs have represented an interesting test case precisely due to their context-dependent status (van Deemter, 2006). Several small datasets of simple visual scenes and corresponding descriptions have been proposed (van Deemter et al., 2006; Viethen and Dale, 2008, 2011; Mitchell et al., 2010, 2013a), and forerunner GRE algorithms have been tested to refer to visible objects using attributes like *size* (Mitchell et al., 2011, 2013b). Due to their extremely small size, however, these datasets are not suitable for large-scale deep learning investigations.

**Language & Vision** Recent work in language and vision has dealt with (gradable) adjectives from at least three perspectives. First, multi-modal information has been used to order real-world entities with respect to an attribute described by a GA. Bagherinezhad et al. (2016), for example, assess whether elephants are *bigger* than butterflies.

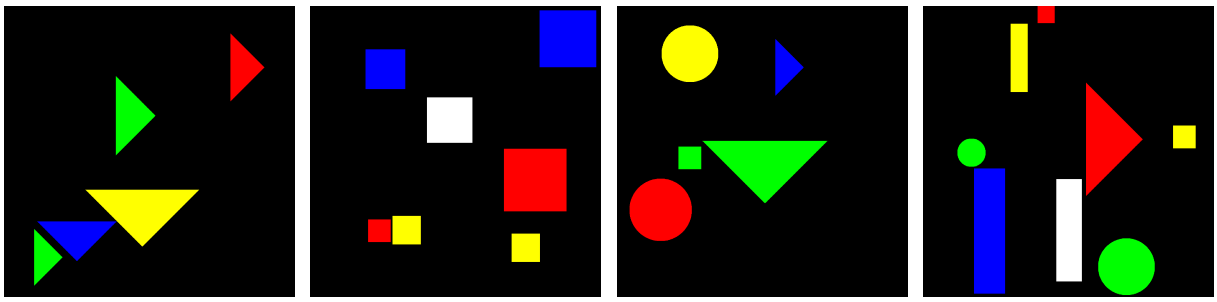

Figure 2: One scene and one corresponding generated true sentence for each of the 4 tasks. From left to right, **SUP$_1$:** The yellow triangle is the *biggest* triangle; **POS$_1$:** The white square is a *small* square; **POS:** The red circle is a *big* object; **SET+POS:** The white rectangle is a *big* rectangle. Best viewed in color.

Thus, rather than modeling the meaning of GAs, this work focuses on the relative size of object types, which is crucially different from our goal. Second, visual gradable attributes have been used as fine-grained features to answer questions about entities in natural scenes (Antol et al., 2015; Hudson and Manning, 2019), to perform discriminative and pragmatically-informative image captioning (Vedantam et al., 2017; Cohn-Gordon et al., 2018), and to discriminate between similar images in reference games (Su et al., 2017). In these works, however, GAs are labels standing for *fixed* attributes: e.g., an airplane is annotated as 'large' or 'small' with no or little relation to the visual context. Third, gradable attributes are explored in work on visual reasoning (Johnson et al., 2017a; Yi et al., 2018), where multi-modal models are challenged to answer complex, high-level questions about objects and their relations in synthetic scenes. Surprisingly, however, questions involving GAs are treated as *non-relational*: e.g., a given size *always* corresponds to either 'big' or 'small', regardless of the objects in the visual context. One exception is Kuhnle et al. (2018), whose investigation of superlative ('biggest') and comparative ('darker') GAs represents a first step toward a *relational* account of these expressions. To our knowledge, we are the first to tackle the modeling of *positive* GAs as a linguistically- and cognitively-inspired relational problem in language and vision.

## 3 Tasks

To test model abilities in learning size GAs, we propose 4 main tasks (see Figure 2). All tasks are framed as a sentence-verification problem: given an image depicting colored, geometric shapes and a sentence about one object's relative size, systems are asked to verify whether the given sentence is true or false in that context. These tasks are aimed at assessing one ability at a time, and are specifically designed to form a pipeline of increasingly difficult operations. Tasks differ with respect to:

- the **statistical function** at play: a max/min function (SUP) or a threshold function (POS);
- the number of geometric **shape types**: one (SUP$_1$/POS$_1$) or several;
- the scope of the **reference set**: the entire visual scene or a subset of objects (SET; only applicable with several shape types per scene).

**SUP$_1$** This task tests whether a model is able to interpret size GAs in their **superlative** form: e.g., 'The yellow triangle is the *biggest* triangle' (only triangles in the scene). To solve this task, a model is required to (1) identify the queried object, (2) measure object size, and (3) determine whether the target object has the largest or smallest size in the entire visual scene.

**POS$_1$** This task evaluates the ability of models to interpret **positive**-form size GAs: e.g., 'The white square is a *small* square' (only squares in the scene). To adequately solve this task, a model is not only required to (1) identify the queried object and (2) measure object size, but also (3) to learn the threshold function that makes an object count as 'big' or 'small' depending on the size of the other objects in the entire visual scene, and (4) to assess whether the target object counts as 'big'/'small' in this context. In contrast to the superlative form, which is precise, the positive form is *vague*: there may be borderline cases (see Figure 1 and *Threshold Function* in Section 4).

**POS** This task is an extension of POS$_1$ where the restriction to one single geometric shape type

per scene is lifted: e.g., 'The red circle is a *big* object' (any shape in the scene) – see Figure 2. As before, and in contrast to the next task, the reference set is the entire visual scene.

**SET+POS** Finally, this task assesses the ability of models to interpret **positive**-form GAs with respect to a restricted context: e.g., 'The white rectangle is a *big* rectangle' (any shape in the scene). To solve this task, in addition to the skills required to address the POS task, a model needs to determine the relevant **reference set** (e.g., the set of rectangles in the scene) and to apply all POS operations to this set. This task, thus, brings together all the key components that make up the semantics of size adjectives, as described in Section 1.

## 4 Datasets

**Visual Data** For each task, we build a dataset of synthetic scenes (1478 x 1478 pixels) depicting 5 to 9 colored objects[2] on a black background. Each object is randomly assigned one shape among circle, rectangle, square, triangle; one color among red, blue, white, yellow, green; one area among 10 predefined ones (based on number of pixels) that we label using tens ranging from 30 to 120 (i.e. 30, 40, 50, ... 120); and a given spatial position in the scene which avoids overlapping with other objects. In tasks $SUP_1$ and $POS_1$ only one shape type is present in a given scene, while in tasks POS and SET+POS several shape types are present.

**Linguistic Data** While generating the visual data, ground-truth labels ('biggest', 'smallest', 'big', 'small') are automatically assigned to each object based on the area of the objects present in the reference set. For tasks $POS_1$, POS and SET+POS, for each object in the scene the generation algorithm generates a sentence based on the template: *The* <color> <shape> *is a* <size> <shape>. For task $SUP_1$, for objects that are biggest/smallest in the scene the algorithm generates a sentence using the template *The* <color> <shape> *is the* <sup_size> <shape>. In order for a sentence to be licensed, the following constraints have to be met: (1) The queried object is uniquely identifiable within the scene, either by <color> in tasks $SUP_1$ and $POS_1$ or by <color, shape> in tasks POS and SET+POS (this ensures there is no ambiguity regarding the target); (2) the area of the queried object must be in the 40-110 range, so to avoid querying objects whose size would always be small/smallest or big/biggest in any scene irrespective of the area of the other objects (see Figure 3, which shows the distribution of queried objects' areas per question type in the train set of POS; as can be seen, no objects with an area equal to either 30 or 120 are present). In task SET+POS, we include two additional constraints: (3) the queried object is neither the biggest nor the smallest object in the entire scene (i.e., the max/min function is not effective without identifying the reference subset); and (4) there are at least three objects in the scene with the same shape as the target (i.e., the reference set includes at least three objects), so to make the computation of the threshold function required to solve the task. Only sentences meeting these constraints are generated and the corresponding scene retained. For each true sentence, a false sentence is automatically generated by replacing the true adjective (e.g., 'big') with the false one (e.g., 'small').

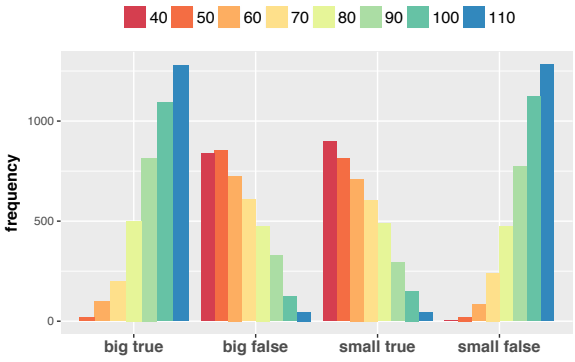


Figure 3: **POS.** Distribution of queried objects' areas per question type in train. This plot highlights an important feature of MALeViC, namely that objects with any area from 40 to 110 can be either 'big' or 'small' depending on the visual context. Best viewed in color.

**Threshold Function** To assign ground-truth labels to objects in tasks $POS_1$, POS and SET+POS, we make use of a *threshold function* experimentally determined by Schmidt et al. (2009), who explore a number of statistical functions to describe speakers' use of the adjective 'tall' (following the authors, we assume these functions to be valid for any GAs, including 'big').[3] In particular, we use their *Relative Height by Range* (RH-R) function,

[2]This resembles the setup used by Barner and Snedeker (2008) in their first experiment, which involves 9 objects.

[3]Note also that, while size may be argued to be two-dimensional, in our approach we treat it as one-dimensional (i.e., based on number of pixels), similarly to tallness.

according to which any item within the top $k\%$ of the range of sizes in the reference set is 'tall' ('big'). According to this function, the threshold T for a given context C is defined as follows: T(C) = Max - $k$ * (Max – Min), where $k \in [0, 1]$, Max is the maximum size in C, and Min the minimum size (see Figure 1). In our data, we make the simplifying assumption to consider 'small' any object that is *not* 'big'. This way, we also avoid dealing with negative statements. Since Schmidt et al. (2009) experimentally show $k$ = 0.29 to best fit their human data, we use this value as our reference $k$. To account for *vagueness*, for each scene we randomly sample a $k$ from the normal distribution of values centered on 0.29 ($\mu$ = 0.29, $\sigma$ = 0.066),[4] as illustrated in Figure 1. This introduces some perturbation in the definition of T, which mimics the fact that speakers may rely on slightly different definitions of GAs (i.e., what counts as 'big' for one person might not *always* count as 'big' for another one; Raffman, 1996; Shapiro, 2006; Sharp et al., 2018), with communication still being successful in most of the cases.

**Datasets** The four final datasets, including 20K datapoints each, are perfectly balanced with respect to the number of cases for each combination of variables used. In particular, 250 <scene, sentence> datapoints are included for each of the 80 classes (4 shapes * 5 colors * 2 sizes * 2 ground truths), where one class is e.g.: <red> <circle> <big> <true>. Such balancing is kept when randomly splitting the datasets into train (16K cases), validation (2K) and test (2K) sets.

## 5 Models

We test 3 models that have proved effective in visual reasoning tasks (Johnson et al., 2017a; Suhr et al., 2018; Yi et al., 2018). All models are *multi-modal*, i.e., they use both a visual representation of the scene and a linguistic representation of the sentence. Unless otherwise specified, visual features representing the scenes are extracted via a *fixed* Convolutional Neural Network (CNN) fed with images resized to 224 x 224 pixels. In particular, features from the *conv4* layer of a ResNet-101 (He et al., 2016) pre-trained on ImageNet (Russakovsky et al., 2015) are used.

**CNN+LSTM** This model simply concatenates the CNN visual features with a representation of the sentence encoded using the final hidden state of an LSTM (Hochreiter and Schmidhuber, 1997). These concatenated features are passed to a Multi-Layer Perceptron (MLP) that predicts the answer.

**Stacked Attention (CNN+LSTM+SA)** This model combines the CNN visual features and the LSTM final state by means of two rounds of soft spatial attention. An MLP over the combined representation predicts the answer (Yang et al., 2016).

**CNN+GRU+Feature-wise Linear Modulation (FiLM)** This model (∼100% on CLEVR; Johnson et al., 2017a) processes the image features by means of 4 residual blocks where the visual representation is linearly transformed by the sentence embedding (i.e., the final hidden state of a GRU; Chung et al., 2014). After a global max-pooling, a two-layer MLP classifier outputs a softmax distribution over the answers (Perez et al., 2018).[5]

**Experimental Setup** For each model in each task, we experiment with 16 configurations of hyper-parameters, i.e., 4 learning rates (5e-5, 3e-4, 5e-4, 5e-3) * 2 dropout values (0, 0.5) * 2 batch normalization options (yes, no). Each model configuration is trained 3 times with randomly initialized weights for 10K iterations (40 epochs), and the best configuration is chosen based on average validation accuracy. In total, 576 models (3 architectures * 4 tasks * 48 runs) are tested. For comparison with previous work (Perez et al., 2018; Kuhnle et al., 2018), we also test FiLM end-to-end (i.e., trained from raw pixels) with the same hyper-parameters as above, for a total of 192 models (4 tasks * 48 runs). Since no substantial differences are observed between the two versions (in line with Perez et al., 2018), we focus on the results by the less tailored and less computationally-expensive models using *pre-trained* features.

## 6 Results

Overall, we observe the expected pattern of results, in line with the conjectured, increasing difficulty of the tasks: accuracy declines from $\text{SUP}_1$ to $\text{POS}_1$ and from POS to SET+POS. There are, however, clear differences across models.

[4] By setting $\sigma$ = 0.066, we expect 99.7% $k$ values to be $\pm$ 3 standard deviations from 0.29, i.e. ranging from 0.09 to 0.49; 95% $k$ within $\pm$ 2 SD (0.16-0.42); 68% within $\pm$ 1 SD (0.22-0.36). $\sigma$ value was set experimentally to keep $k < 0.5$.

[5] For CNN+LSTM and CNN+LSTM+SA, we use the implementations by Johnson et al. (2017b). For FiLM, we use the implementation by Perez et al. (2018). All models are trained using Python 3.5.2 and PyTorch v1.0.1.

| task | model | accuracy | | | hyper-parameters | | |
|---|---|---|---|---|---|---|---|
| | | max val (pixels) | avg val ± sd | avg test ± sd | lr | drop | b norm |
| **SUP$_1$** | CNN-LSTM | 0.841 | 0.8153 ± 0.033 | 0.8066 ± 0.033 | 5e-4 | 0.5 | no |
| | CNN-LSTM-SA | **1** | 0.999 ± 0 | 0.9983 ± 0.001 | 3e-4 | 0 | no |
| | FilM | **1 (1)** | **0.9991 ± 0** | **0.999 ± 0.001** | 3e-4 | 0.5 | no |
| **POS$_1$** | CNN-LSTM | 0.5615 | 0.5493 ± 0.009 | 0.5455 ± 0.013 | 5e-4 | 0.5 | yes |
| | CNN-LSTM-SA | 0.941 | **0.9396 ± 0.001** | **0.9306 ± 0.002** | 3e-4 | 0 | no |
| | FiLM | **0.9415 (0.9565)** | 0.8673 ± 0.085 | 0.8546 ± 0.086 | 5e-5 | 0.5 | no |
| **POS** | CNN-LSTM | 0.574 | 0.5668 ± 0.005 | 0.5493 ± 0.008 | 3e-4 | 0.5 | yes |
| | CNN-LSTM-SA | 0.942 | **0.9386 ± 0.002** | **0.94 ± 0.002** | 3e-4 | 0 | yes |
| | FiLM | **0.945 (0.9475)** | 0.9375 ± 0.004 | 0.9333 ± 0.002 | 5e-4 | 0 | no |
| **SET+POS** | CNN-LSTM | 0.591 | 0.5808 ± 0.014 | 0.551 ± 0.01 | 5e-4 | 0.5 | yes |
| | CNN-LSTM-SA | 0.81 | 0.7901 ± 0.014 | 0.7751 ± 0.01 | 5e-5 | 0 | no |
| | FiLM | **0.9205 (0.9295)** | **0.8845 ± 0.027** | **0.8788 ± 0.021** | 5e-4 | 0 | no |
| ***all*** | *chance* | 0.5 | 0.5 | 0.5 | | | |

Table 1: Results by each **model** (column 2) in each **task** (1). Note that **max val** (3) refers to the highest accuracy obtained in the task by a given architecture across 192 runs (in brackets, highest accuracy obtained across 192 runs by FiLM trained from raw **pixels**), while **avg val** ± **sd** (4) and **avg test** ± **sd** (5) refer to the average accuracy (and relative standard deviation) obtained across 3 runs by the best configuration of hyper-parameters of a given architecture on, respectively, val and test set. As for the **hyper-parameters** (6), we report value of learning rate (**lr**), dropout (**drop**), and use/not use of batch normalization (**b norm**). Values in bold are the highest in the column.

CNN+LSTM does well on SUP (the simplest task) but performs around chance on all other tasks (which require a threshold function). Both CNN+LSTM+SA and FiLM obtain high accuracy on POS$_1$/POS, while FiLM neatly outperforms CNN+LSTM+SA in the most challenging SET+POS task, proving its higher ability to reason over complex, context-dependent linguistic expressions. Accuracy scores are reported in Table 1. In the following, we describe the results per task in more detail.

**SUP$_1$** Both CNN+LSTM+SA and FiLM obtain perfect accuracy (∼100%), with CNN+LSTM performing well above chance (∼80%). These results indicate that assessing whether an object is the 'biggest'/'smallest' in the scene is a relatively simple task even for basic models. This is encouraging since mastering *superlatives* requires several core skills that also underlie the use of *positive* GAs, namely (1) object identification, (2) object size estimation, and (3) object size ordering.

**POS$_1$ / POS** CNN+LSTM+SA and FiLM obtain similar, very high max accuracy (∼94%) in these two tasks, with CNN+LSTM being only slightly above chance level (∼57%). Note that in this case we should not expect models to obtain 100% accuracy: while the min/max function in SUP is precise, the POS threshold function is vague (giving rise to borderline cases). The performance to be expected with sharp $k = 0.29$ is 97% for these two tasks (i.e, only in 3% of cases $k$ is assigned a value that makes the truth/falsity of a sentence different from what it would be with $k = 0.29$). The performance of the two top models is thus 3% below average ceiling accuracy (94% vs. 97%). These results, thus, indicate that CNN+LSTM+SA and FiLM are able to learn the *threshold function* that makes an object count as 'big'/'small' in a given visual scene, while CNN+LSTM is not.[6]

**SET+POS** As before, CNN+LSTM performs only slightly above chance. Both FiLM and CNN+LSTM+SA experience a drop in performance compared to POS$_1$/POS, confirming that computing the threshold function over a subset of objects is the most challenging task. Applying the threshold function to the entire visual scene would yield ∼65% accuracy in this dataset. This indicates that none of the two top models uses this strategy simplistically, as their performance is well above this result (max 81% for CNN+LSTM+SA and 92% for FiLM).[7] FiLM neatly outperforms CNN+LSTM+SA (+11% in both max and average accuracy), thus showing a more robust pattern of results across tasks and a higher ability to handle complex reasoning problems.

[6] FiLM's performance in POS$_1$ (where each scene contains only one shape type, but all shape types are seen across images) is rather unstable across runs (high SD). Since this is not so in POS, we conjecture that it may be due to the model learning shape-specific strategies in some runs of POS$_1$.

[7] As in POS$_1$/POS, ceiling performance with fixed $k = 0.29$ is 97% in this dataset.

| hard test (train) | model | avg acc ± sd |
|---|---|---|
| **POS-hard (POS)** | CNN+LSTM | 0.5325 ± 0.01 |
| | CNN+LSTM+SA | 0.8653 ± 0.005 |
| | FiLM | **0.8693 ± 0.003** |
| **SET+POS-hard (SET+POS)** | CNN+LSTM | 0.4623 ± 0.004 |
| | CNN+LSTM+SA | 0.478 ± 0.015 |
| | FiLM | **0.6513 ± 0.059** |
| ***all*** | *chance* | 0.5 |

Table 2: **Average accuracy** across 3 runs by a **model** in a **hard test** set. Models are trained on the task in **train**. Values in bold are the highest in the column.

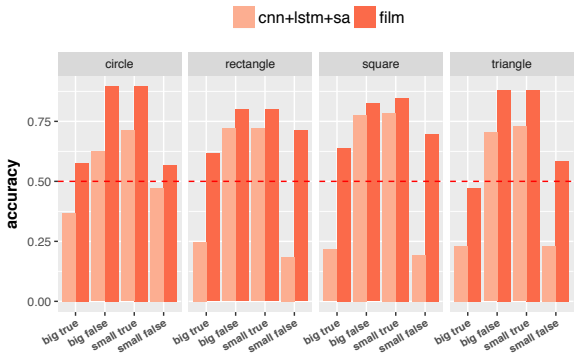


Figure 4: **SET+POS-hard.** Accuracy per sentence type (big true, big false, small true, small false) obtained by the best run of CNN+LSTM+SA (49% acc.) and FiLM (73% acc.) for each shape type. The dashed line signals *chance* level. Best viewed in color.

While these results are very encouraging, they might be the outcome of exploiting biases in the data rather than learning the semantic components that make up the meaning of GAs according to linguistic theory and psycholinguistic evidence. For example, identifying the reference subset but applying the max/min function (rather than a threshold function) to this set would yield a remarkable 92% accuracy, which is on a par with the top result by FiLM. In what follows, we investigate the abilities of the trained models in more depth by testing them on a number of diagnostic test sets.

## 7 Analysis

In this section, we aim at better understanding what the models learn (or do not learn) when performing the different tasks. To do so, we carry out a *stress-test* and a *compositionality* analysis.

### 7.1 Stress-Test: Hard Contexts

To test to what extent models master the core abilities that are arguably needed to perform POS and SET+POS, we build two *hard* diagnostic test sets which make the use of other strategies not effective. Similarly to the main datasets, these hard test sets include 2K perfectly-balanced datapoints.

**POS-hard** We explore whether models trained on POS properly learn to compute the threshold function rather than using a simplistic strategy such as applying the min/max function over the visual context. Indeed, it has been proposed that young children might use positive-form GAs as superlatives in early stages of language acquisition (Clark, 1970). To test whether this explains (part of) the results on POS, we build a *hard* test set identical to POS except for the fact that objects that are either the biggest or the smallest in the entire visual scene are never queried. Thus, in POS-hard systematically using the superlative strategy would result in chance level scores. As reported in Table 2, the accuracy obtained by CNN+LSTM+SA and FiLM does not dramatically decrease compared to POS (∼94% vs. ∼87%). The drop can be explained by the fact that in POS-hard we query objects that are overall *closer* to the threshold, thus increasing the number of more difficult, borderline cases.[8] This test thus confirms that CNN+LSTM+SA and FiLM do learn to generally compute and apply the threshold function over an entire visual scene.

**SET+POS-hard** To investigate whether models trained on SET+POS learn to identify the reference subset and to apply the threshold function to it, we build a hard test set identical to SET+POS except for the fact that objects that are the biggest or the smallest *in the reference set* are never queried. Thus, in SET+POS-hard applying the min/max function to the reference set would lead to chance level. As shown in Table 2, only FiLM is above chance level in this test set, with 65% accuracy. Both CNN+LSTM and CNN+LSTM+SA obtain scores slightly below chance (–30% for CNN+LSTM+SA compared to SET+POS). This is a striking result, which reveals that the accuracy scores achieved by CNN+LSTM+SA in SET+POS must be due to shortcut strategies.

While we do not have full understanding of what is being exploited by the CNN+LSTM+SA model, we do observe a bias towards predicting that an object counts as 'small'. As shown in Figure 4, the model obtains high accuracy on sentences involving small objects (small true, big

[8] This also leads to slightly lower average ceiling accuracy in POS-hard: performance with fixed $k = 0.29$ is now ∼92% in contrast to ∼97% in POS.




Figure 5: **SET+POS-hard.** Distribution of $k$ against sentence type (clockwise from top-left: big true, big false, small false, small true). Cases labelled as *different* are those for which the ground truth adjective (e.g. 'big') would change with $k = 0.29$ (e.g., 'small'); in *same* cases, it would be the same. Best viewed in color.

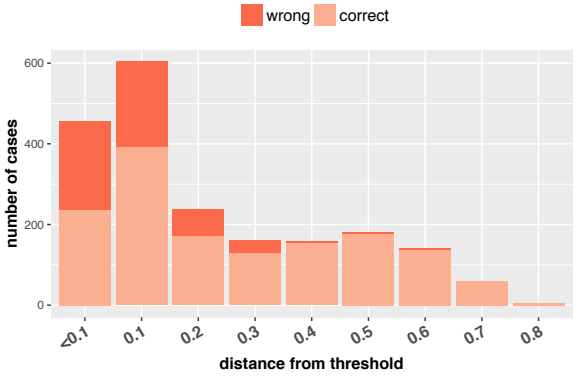


Figure 6: **SET+POS-hard.** Correct and wrong predictions by best run of FiLM (73% acc.) against distance from the threshold. Best viewed in color.

false), while its accuracy on sentences targeting big objects (big true, small false) is below chance level. Indeed, the model predicts *false* for big objects and *true* for small objects in 73% of cases.

As for FiLM, its pattern of accuracy is overall more stable across sentence types, though big objects (big true, small false) are still significantly harder than small ones (small true, big false). Bigger objects are more challenging because by definition they are closer to the threshold (as can clearly be seen in Figure 1), which in turn means that they are more likely to be borderline cases. The plots in Figure 5 illustrate this effect, showing that objects that count as 'big' are more likely to flip the truth value of a sentence due to the fuzziness of the threshold. FiLM's results are thus to be expected, and in fact encouraging: if a model is correctly learning to apply the threshold function to the reference set, most of the errors should involve borderline objects. This is confirmed: see Figure 6, where correct and wrong predictions by the best FiLM model are plotted against the (normalized) distance of the queried object from the threshold (83% of the errors within the 2 leftmost columns correspond to objects that count as 'big'). In Figure 7, we report two *borderline* cases randomly sampled from the test set where distance from the threshold is 0.09 (a) and 0.04 (b), respectively. For both scenes, the sentence 'The red rectangle is a *big* rectangle' is *true*. However, FiLM's best run predicts the correct answer only in (a). By visualizing the distribution of locations responsible for FiLM's globally-pooled features fed to the classifier, we notice that, in both cases, features related to the queried object (the red rectangle) and objects in the reference set (rectangles) are highly activated (though with a bias toward larger objects), confirming the ability of FiLM to focus on language-relevant features. However, (high) activations are also assigned to *unrelated* objects' features, particularly in (b). Though no conclusions can be drawn from these examples, they suggest that FiLM uses features related to several objects, in line with the *relational* nature of the task.

Finally, the evidence that models obtain slightly different results across shapes (see Figure 4) could suggest that models learn different, shape-specific representations of 'big' and 'small'. The following analysis precisely aims at exploring this issue.

### 7.2 Compositionality: Unseen Combinations

In our last analysis, we investigate whether the models learn an *abstract* representation of GAs that can be compositionally applied to unseen adjective-noun combinations. We focus on the SET+POS task and extract a subset of the corresponding dataset in which each size adjective appears only with two nouns denoting two distinct shape types: 'big (circle|rectangle)', 'small (square|triangle)'. This subset of the data amounts to half of SET+POS and thus contains 10K datapoints, that we split into train (8K), val (1K), and test (1K). We refer to this test set as *seen* (its adjective-noun combinations are seen in training). We then create a second, *unseen* test set with 1K datapoints, where the adjectives appear with different nouns: 'big (square|triangle)', 'small (circle|rectangle)'. We train all models three times using their best hyper-parameter configurations and evaluate them on both *seen* and *unseen*. If

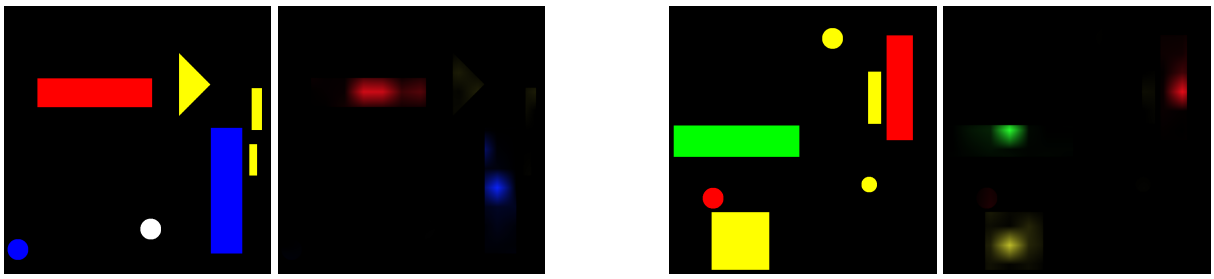

(a) (b)

Figure 7: **SET+POS-hard.** Two *borderline* cases from the test set where FiLM's best run makes a correct (a) or wrong (b) prediction to 'The red rectangle is a *big* rectangle', which is *true* in both scenes. The rightmost panels show the distribution of locations used by FiLM for its globally max-pooled features. Best viewed in color.

models learn an abstract representation of GAs that includes a variable for the noun they modify, we should observe no difference in performance between the two test sets.

As reported in Table 3, this turns out not to be the case. While in the *seen* test set all models obtain similar accuracies to those obtained in SET+POS, in *unseen* their performance is not only well below chance level, but it is in fact the inverse of their results on *seen*: all instances which are predicted correctly in *seen* are incorrectly predicted in *unseen* (e.g, FiLM obtains ∼85% on *seen* and ∼15% on *unseen*). This suggests that the models learn a default strategy per noun rather than an abstract adjective representation that generalizes to unseen combinations. For example, the models predict *true* any time the size of a circle or a rectangle exceeds a certain threshold, regardless of whether the noun is modified by 'big' (*seen* combinations) or 'small' (*unseen* combinations).

| model | avg *seen* ± sd | avg *unseen* ± sd |
|---|---|---|
| CNN+LSTM | 0.608 ± 0.01 | 0.4036 ± 0.003 |
| CNN+LSTM+SA | 0.7813 ± 0.009 | 0.235 ± 0.006 |
| FiLM | 0.8489 ± 0.014 | 0.153 ± 0.02 |
| *chance* | 0.5 | 0.5 |

Table 3: **Compositional task.** Results by the models.

## 8 Discussion

We tackle the modeling of size GAs as a *relational* problem in the domain of visual reasoning, and show (in contrast with Kuhnle et al., 2018) that FiLM is able to learn the function underlying the meaning of these expressions. However, none of the models develop an *abstract* representation of GAs that can be applied *compositionally*, an ability that even 4-year-olds master (Barner and Snedeker, 2008). This is in line with recent evidence showing that deep learning models do not rely on systematic compositional rules (Baroni, 2019; Bahdanau et al., 2019).

An interesting open question, which we plan to explore in future work, is whether training models to jointly learn superlative, comparative, and positive GAs (similarly to how Pezzelle et al. (2018) did for quantities), or framing the task in a dialogue setting (as Monroe et al. (2017) did for colors) could lead to more compositional models. Moreover, it might be worth exploring whether equipping models with similar inductive biases as those leading speakers of any language to develop abstract, compositional representations of size adjectives is needed to properly handle these expressions. In parallel, the present work could be extended to other GAs and threshold functions.

In the long term, we aim to move to natural images. This requires world knowledge, a confounding factor intentionally abstracted away in synthetic data. Since children learn to exploit world knowledge after mastering the perceptual context (Tribushinina, 2013), adopting an incremental approach might be promising.

## Acknowledgments

We kindly thank Raffaella Bernardi and Elia Bruni for insightful discussions and their comments on an early draft of this paper, Alexander Kuhnle for sharing opinions and details regarding the models, and the audience at talks given at the ILLC and at the Blackbox@NL Workshop held at the Jheronimus Academy of Data Science. We are very grateful to the anonymous reviewers of EMNLP-IJCNLP for their valuable feedback. This work was funded by the Netherlands Organisation for Scientific Research (NWO) under VIDI grant no. 276-89-008, *Asymmetry in Conversation*.

## References


Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433.

Hessam Bagherinezhad, Hannaneh Hajishirzi, Yejin Choi, and Ali Farhadi. 2016. Are elephants bigger than butterflies? Reasoning about sizes of objects. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Dzmitry Bahdanau, Shikhar Murty, Michael Noukhovitch, Thien Huu Nguyen, Harm de Vries, and Aaron Courville. 2019. Systematic generalization: What is required and can it be learned? In *International Conference on Learning Representation (ICLR)*.

Chris Barker. 2002. The dynamics of vagueness. *Linguistics & Philosophy*, 25(1):1–36.

David Barner and Jesse Snedeker. 2008. Compositionality and statistics in adjective acquisition: 4-year-olds interpret tall and short based on the size distributions of novel noun referents. *Child development*, 79(3):594–608.

Marco Baroni. 2019. Linguistic generalization and compositionality in modern artificial neural networks. ArXiv preprint arXiv:1904.00157, to appear in the Philosophical Transactions of the Royal Society B.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *Workshop on Deep Learning at NIPS-2014*.

Herbert H Clark. 1970. The primitive nature of children's relational concepts. *Cognition and the development of language*, pages 269–278.

Reuben Cohn-Gordon, Noah Goodman, and Christopher Potts. 2018. Pragmatically informative image captioning with character-level inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 439–443.

Max J Cresswell. 1976. The semantics of degree. In *Montague grammar*, pages 261–292. Elsevier.

Gerard De Melo and Mohit Bansal. 2013. Good, great, excellent: Global inference of semantic intensities. *Transactions of the Association for Computational Linguistics*, 1:279–290.

Kees van Deemter, Ielka van der Sluis, and Albert Gatt. 2006. Building a semantically transparent corpus for the generation of referring expressions. In *Proceedings of the Fourth International Natural Language Generation Conference*, pages 130–132. Association for Computational Linguistics.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. Language, Speech, and Communication. MIT Press, Cambridge, MA.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Drew A Hudson and Christopher D Manning. 2019. GQA: A new dataset for real-world visual reasoning and compositional question answering. *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017a. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2901–2910.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Judy Hoffman, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017b. Inferring and executing programs for visual reasoning. In *Proceedings of the IEEE International Conference on Computer Vision*.

Hans Kamp. 1975. Two theories of adjectives. In E. Keenan, editor, *Formal Semantics of Natural Language*, pages 123–155. Cambridge University Press.

Christopher Kennedy. 1999. *Projecting the Adjective: The Syntax and Semantics of Gradability and Comparison*. Routledge.

Christopher Kennedy. 2007. Vagueness and grammar: The semantics of relative and absolute gradable adjectives. *Linguistics and Philosophy*, 30(1):1–45.

Joo-Kyung Kim and Marie-Catherine de Marneffe. 2013. Deriving adjectival scales from continuous space word representations. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1625–1630.

Joo-Kyung Kim, Marie-Catherine de Marneffe, and Eric Fosler-Lussier. 2016. Adjusting word embeddings with semantic intensity orders. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 62–69.

Alexander Kuhnle, Huiyuan Xie, and Ann Copestake. 2018. How clever is the FiLM model, and how clever can it be? In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Daniel Lassiter and Noah D Goodman. 2017. Adjectival vagueness in a bayesian model of interpretation. *Synthese*, 194(10):3801–3836.

Marie-Catherine de Marneffe, Christopher D Manning, and Christopher Potts. 2010. Was it good? It was provocative. Learning the meaning of scalar adjectives. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 167–176. Association for Computational Linguistics.

Margaret Mitchell, Kees van Deemter, and Ehud Reiter. 2010. Natural reference to objects in a visual domain. In *Proceedings of the 6th international natural language generation conference*, pages 95–104. Association for Computational Linguistics.

Margaret Mitchell, Kees van Deemter, and Ehud Reiter. 2011. Applying machine learning to the choice of size modifiers. In *Proceedings of the 2nd PRE-CogSci Workshop*.

Margaret Mitchell, Ehud Reiter, and Kees Van Deemter. 2013a. Typicality and object reference. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 35.

Margaret Mitchell, Kees Van Deemter, and Ehud Reiter. 2013b. Generating expressions that refer to visible objects. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics (ACL).

Will Monroe, Robert XD Hawkins, Noah D Goodman, and Christopher Potts. 2017. Colors in context: A pragmatic neural model for grounded language understanding. *Transactions of the Association for Computational Linguistics*, 5:325–338.

Barbara Partee. 1995. Lexical semantics and compositionality. *An invitation to cognitive science: Language*, 1:311–360.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. 2018. Film: Visual reasoning with a general conditioning layer. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Sandro Pezzelle, Ionut-Teodor Sorodoc, and Raffaella Bernardi. 2018. Comparatives, quantifiers, proportions: A multi-task model for the learning of quantities from vision. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 419–430.

Manfred Pinkal. 1979. Semantics from different points of view. In R. Bäurle, U. Egli, and A. von Stechow, editors, *How to Refer with Vague Descriptions*, pages 32–50. Springer-Verlag.

Diana Raffman. 1996. Vagueness and context-relativity. *Philosophical Studies*, 81(2):175–192.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. ImageNet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252.

Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. 2017. A simple neural network module for relational reasoning. In *Advances in neural information processing systems*, pages 4967–4976.

Lauren A Schmidt, Noah D Goodman, David Barner, and Joshua B Tenenbaum. 2009. How tall is tall? Compositionality, statistics, and gradable adjectives. In *Proceedings of the 31st annual Conference of the Cognitive Science Society*, pages 2759–2764.

Stewart Shapiro. 2006. *Vagueness in context*. Oxford University Press.

Rebecca Sharp, Mithun Paul, Ajay Nagesh, Dane Bell, and Mihai Surdeanu. 2018. Grounding gradable adjectives through crowdsourcing. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*.

Vera Sheinman, Christiane Fellbaum, Isaac Julien, Peter Schulam, and Takenobu Tokunaga. 2013. Large, huge or gigantic? Identifying and encoding intensity relations among adjectives in WordNet. *Language resources and evaluation*, 47(3):797–816.

Chaitanya Shivade, Marie-Catherine de Marneffe, Eric Fosler-Lussier, and Albert M Lai. 2015. Corpus-based discovery of semantic intensity scales. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–493.

Jong-Chyi Su, Chenyun Wu, Huaizu Jiang, and Subhransu Maji. 2017. Reasoning about fine-grained attribute phrases using reference games. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 418–427.

Alane Suhr, Stephanie Zhou, Iris Zhang, Huajun Bai, and Yoav Artzi. 2018. A corpus for reasoning about natural language grounded in photographs. In *Workshop on Visually Grounded Interaction and Language (ViGIL) at NeurIPS-2018*.

Elena Tribushinina. 2013. Adjective semantics, world knowledge and visual context: Comprehension of size terms by 2-to 7-year-old Dutch-speaking children. *Journal of Psycholinguistic Research*, 42(3):205–225.

Kees van Deemter. 2006. Generating referring expressions that involve gradable properties. *Computational Lingustics*, 32(2):195–222.

Kees Van Deemter. 2012. *Not exactly: In praise of vagueness*. Oxford University Press.

Ramakrishna Vedantam, Samy Bengio, Kevin Murphy, Devi Parikh, and Gal Chechik. 2017. Context-aware captions from context-agnostic supervision. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Jette Viethen and Robert Dale. 2008. The use of spatial relations in referring expression generation. In *Proceedings of the Fifth International Natural Language Generation Conference*, pages 59–67. Association for Computational Linguistics.

Jette Viethen and Robert Dale. 2011. GRE3D7: A corpus of distinguishing descriptions for objects in visual scenes. In *Proceedings of the UCNLG+ Eval: Language generation and evaluation workshop*, pages 12–22. Association for Computational Linguistics.

Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. 2016. Stacked attention networks for image question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 21–29.

Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Josh Tenenbaum. 2018. Neural-symbolic VQA: Disentangling reasoning from vision and language understanding. In *Advances in Neural Information Processing Systems*, pages 1031–1042.

# Investigating BERT's Knowledge of Language: Five Analysis Methods with NPIs

**Alex Warstadt,**[†,1,2] **Yu Cao,**[†,3] **Ioana Grosu,**[†,2] **Wei Peng,**[†,3] **Hagen Blix,**[†,1] **Yining Nie,**[†,1,2] **Anna Alsop,**[†,2] **Shikha Bordia,**[†,3] **Haokun Liu,**[†,3] **Alicia Parrish,**[†,2,3] **Sheng-Fu Wang,**[†,3] **Jason Phang,**[†,1,3] **Anhad Mohananey,**[†,1,3] **Phu Mon Htut,**[†,3] **Paloma Jeretič,**[†,1,2] **and Samuel R. Bowman**
New York University

[†]Equal contribution with roles given below; order assigned randomly. Correspondence: bowman@nyu.edu
[1]Framing and organizing the paper [2]Creating diagnostic data [3]Constructing and running experiments

## Abstract

Though state-of-the-art sentence representation models can perform tasks requiring significant knowledge of grammar, it is an open question how best to evaluate their grammatical knowledge. We explore five experimental methods inspired by prior work evaluating pretrained sentence representation models. We use a single linguistic phenomenon, negative polarity item (NPI) licensing in English, as a case study for our experiments. NPIs like *any* are grammatical only if they appear in a *licensing environment* like negation (*Sue doesn't have any cats vs. *Sue has any cats*). This phenomenon is challenging because of the variety of NPI licensing environments that exist. We introduce an artificially generated dataset that manipulates key features of NPI licensing for the experiments. We find that BERT has significant knowledge of these features, but its success varies widely across different experimental methods. We conclude that a variety of methods is necessary to reveal all relevant aspects of a model's grammatical knowledge in a given domain.

## 1 Introduction

Recent sentence representation models have attained state-of-the-art results on language understanding tasks, but standard methodology for evaluating their knowledge of grammar has been slower to emerge. Recent work evaluating grammatical knowledge of sentence encoders like BERT (Devlin et al., 2018) has employed a variety of methods. For example, Shi et al. (2016), Ettinger et al. (2016), and Tenney et al. (2019) use probing tasks to target a model's knowledge of particular grammatical features. Marvin and Linzen (2018) and Wilcox et al. (2019) compare language models' probabilities for pairs of minimally different sentences differing in grammatical acceptability. Linzen et al. (2016), Warstadt et al. (2018), and Kann et al. (2019) use Boolean acceptability judgments inspired by methodologies in generative linguistics. However, we have not yet seen any substantial direct comparison between these methods, and it is not yet clear whether they tend to yield similar conclusions about what a given model knows.

We aim to better understand the trade-offs in task choice by comparing different methods inspired by previous work to evaluate sentence understanding models in a single empirical domain. We choose as our case study negative polarity item (*NPI*) licensing in English, an empirically rich phenomenon widely discussed in the theoretical linguistics literature (Kadmon and Landman, 1993; Giannakidou, 1998; Chierchia, 2013, a.o.). NPIs are words or expressions that can only appear in environments that are, in some sense, *negative* (Fauconnier, 1975; Ladusaw, 1979; Linebarger, 1980). For example, *any* is an NPI because it is acceptable in negative sentences (1) but not positive sentences (2); negation thus serves as an NPI *licensor*. NPIs furthermore cannot be outside the syntactic *scope* of a licensor (3). Intuitively, a licensor's scope is the syntactic domain in which an NPI is licensed, and it varies from licensor to licensor. A sentence with an NPI present is only acceptable in cases where (i) there is a licensor—as in (1) but not (2)—and (ii) the NPI is within the scope of that licensor—as in (1) but not (3).

(1) Mary has**n't** eaten *any* cookies.

(2) *Mary has eaten *any* cookies.

(3) **Any* cookies have**n't** been eaten.

We compare five experimental methods to test BERT's knowledge of NPI licensing. We consider: (i) a *Boolean acceptability classification* task to test BERT's knowledge of sentences in



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2877–2887, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

isolation, (ii) an *absolute minimal pair* task evaluating whether the absolute Boolean outputs of acceptability classifiers distinguish between pairs of minimally different sentences that differ in acceptability and each isolate a single key property of NPI licensing, (iii) a *gradient minimal pair* task evaluating whether the gradient outputs of acceptability classifiers distinguish between minimal pairs, (iv) a *cloze test* evaluating the grammatical preferences of BERT's masked language modeling head, and (v) a *probing task* directly evaluating BERT's representations for knowledge of specific grammatical features relevant to NPI licensing.

We find that BERT does have knowledge of all the key features necessary to predict the acceptability of NPI sentences in our experiments. However, our five methods give meaningfully different results. While the gradient minimal pair experiment and, to a lesser extent, the acceptability classification and cloze tests indicate that BERT has systematic knowledge of all NPI licensing environments and relevant grammatical features, the absolute minimal pair and probing experiments show that BERT's knowledge is in fact not equal across these domains. We conclude that each method depicts different relevant aspects of a model's grammatical knowledge; comparing both gradient and absolute measures of performance of models gives a more complete picture. We recommend that future studies use multiple complementary methods to evaluate model performance.

## 2 Related Work

**Evaluating Sentence Encoders** The success of sentence encoders and broader neural network methods in NLP has prompted significant interest in understanding the linguistic knowledge encapsulated in these models.

A portion of related work focuses on Boolean classification tasks of English sentences to evaluate the grammatical knowledge encoded in these models. The objective of this task is to predict whether a single input sentence is acceptable or not, abstracting away from gradience in acceptability judgments. Linzen et al. (2016) train classifiers on this task using data with manipulated verbal inflection to investigate whether LSTMs can identify subject-verb agreement violations, and therefore a (potentially long distance) syntactic dependency. Warstadt et al. (2018) train models on this task using the CoLA corpus of acceptability judgments as a method for evaluating domain general grammatical knowledge, and Warstadt and Bowman (2019) analyze how these domain general classifiers perform on specific linguistic phenomena. Kann et al. (2019) use this task to evaluate whether sentence encoders represent information about verbal argument structure classes.

A related method employs minimal pairs, consisting of two sentences that differ minimally in their content and differ in linguistic acceptability, to judge whether a model is sensitive to a single grammatical feature. Marvin and Linzen (2018) and Wilcox et al. (2019) apply this method to phenomena such as subject-verb agreement, NPI licensing, and reflexive licensing.

Another branch of work uses probing tasks in which the objective is to predict the value of a particular linguistic feature given an input sentence. Probing tasks have been used to investigate whether sentence embeddings encode syntactic and surface features such as tense and voice (Shi et al., 2016), sentence length and word content (Adi et al., 2016), or syntactic depth and morphological number (Conneau et al., 2018). Giulianelli et al. (2018) use diagnostic classifiers to track the propagation of information in RNN-based language models. Ettinger et al. (2018) and Dasgupta et al. (2018) use automatic data generation to evaluate compositional reasoning. Tenney et al. (2019) introduce sub-sentence level probing tasks derived from common NLP tasks.

**Negative Polarity Items** In the theoretical literature on NPIs, proposals have been made to unify the properties of the diverse NPI licensing environments. For example, a popular view states that NPIs are licensed if and only if they occur in downward entailing (DE) environments (Fauconnier, 1975; Ladusaw, 1979), i.e. an environment that licences inferences from sets to subsets.[1]

For instance, (4) shows that the environment under the scope of negation is DE. The property *been to Paris* is a subset of *been to France*, because those who have been to Paris are a subset of those who have been to France. While the inference from set to subset is normally invalid (4-b), it is valid if the property is embedded under negation (4-a). According to the DE theory of NPI licens-

[1]Other prominent theories of NPI licensing are based on notions of non-veridicality (Giannakidou, 1994, 1998; Zwarts, 1998), domain widening and emphasis (Kadmon and Landman, 1993; Krifka, 1995; Chierchia, 2013), a.o.

ing, this explains the contrast between (1) and (2).

(4) a. I haven't been to France.
$\rightarrow$ I haven't been to Paris. (DE)
b. I have been to France.
$\not\rightarrow$ I have been to Paris. (not DE)

This view does not capture licensing in some environments, for example questions. No theory is yet accepted as identifying the unifying properties of all NPI licensing environments.

Within computational linguistics, NPIs are used as a testing ground for neural network models' grammatical knowledge. Marvin and Linzen (2018) find that LSTM LMs do not systematically prefer sentences with licensed NPIs (1) over sentencew with unlicensed NPIs (2). Jumelet and Hupkes (2018) shows LSTM LMs find a relation between the licensing context and the negative polarity item, and appears to be aware of the scope of this context. Wilcox et al. (2019) use NPIs and filler-gap dependencies, as instances of non-local grammatical dependencies, to probe the effect of supervision with hierarchical structure. They find that structurally-supervised models outperform state-of-the-art sequential LSTM models, showing the importance of structure in learning non-local dependencies like NPI licensing.

**CoLA** We use the Corpus of Linguistic Acceptability (CoLA; Warstadt et al., 2018) in our experiments to train supervised acceptability classifiers. CoLA is a dataset of over 10k syntactically diverse example sentences from the linguistics literature with Boolean acceptability labels. As is conventional in theoretical linguistics, sentences are taken to be *acceptable* if native speakers judge them to be possible sentences in their language. Such sentences are widely used in linguistics publications to illustrate phenomena of interest. The examples in CoLA are gathered from diverse sources and represent a wide array of syntactic, semantic, and morphological phenomena. As measured by the GLUE benchmark (Wang et al., 2018), acceptability classifiers trained on top of BERT and related models reach near-human performance on CoLA.

## 3 Methods

We experiment with five approaches to the evaluation of grammatical knowledge of sentence representation models like BERT using our generated NPI acceptability judgment dataset (§4). Each data sample in the dataset contains a sentence, a Boolean label which indicates whether the sentence is grammatically acceptable or not, and three Boolean meta-data variables (licensor, NPI, and scope; Table 2). We evaluate four model types: BERT-large, BERT with fine-tuning on one of two tasks, and a simple bag-of-words baseline using GloVe word embeddings (Pennington et al., 2014).

**Boolean Acceptability** We test the model's ability to judge the grammatical acceptability of the sentences in the NPI dataset. Following standards in linguistics, sentences for this task are assumed to be either totally acceptable or totally unacceptable. We fine-tune the sentence representation models to perform these Boolean judgments. For BERT-based sentence representation models, we add a classifier on top of the `[CLS]` embedding of the last layer. For BoW, we use a max pooling layer followed by an MLP classifier. The performance of the models is measured as Matthews Correlation Coefficient (MCC; Matthews, 1975)[2] between the predicted label and the gold label.

**Absolute Minimal Pair** We conduct a minimal pair experiment to analyze Boolean acceptability classifiers on minimally different sentences. Two sentences within paradigm form a minimal pair if they differ in one feature (licensor, NPI, or scope), but have different acceptability. For example, the sentences in (1) and (2) differ in whether an NPI licensor (negation) is present. We evaluate the models trained on acceptability judgments with the minimal pairs. In *absolute minimal pair* evaluation, the models needs to correctly classify both sentences in the pair to be counted as correct.

**Gradient Minimal Pair** The *gradient minimal pair* evaluation is a more lenient version of *absolute minimal pair* evaluation: Here, we count a pair as correctly classified as long as the Boolean classifier's output for the acceptable class is higher for the acceptable sentence than for the unacceptable sentence. In other words, the classifier need only predict the acceptable sentence has the higher likelihood of being acceptable, but need not correctly predict that it is acceptable.

**Cloze Test** In the cloze test, a standard sentence-completion task, we use the masked language modeling (MLM) component in BERT (Devlin et al., 2018) and evaluate whether it assigns a

[2]MCC gives the correlation of two Boolean distributions between -1 and 1. A score of 0 is given to any two unrelated distributions, without requiring balanced class sizes.

higher probability to the acceptable sentence in a minimal pair, following Linzen et al. (2016). An MLM predicts the probability of a single masked token based on the rest of the sentence. The minimal pairs tested are a subset of those in the absolute and gradient minimal pair experiments, where both sentences must be equal in length and differ in only one token. This differing token is replaced with `[MASK]`, and the minimal pair is taken to be classified correctly if the MLM assigns a higher probability to the token from the acceptable sentence. In contrast with the other minimal pair experiments, this experiment is entirely unsupervised, using BERT's native MLM functionality.

**Feature Probing** We use probing classifiers as a more fine-grained approach to the identification of grammatical variables. We freeze the sentence encoders both with and without fine-tuning from the acceptability judgment experiments and train lightweight classifiers on top of them to predict meta-data labels corresponding to the key properties a model must learn in order to judge the acceptability of NPI sentences. These properties are whether a licensor is present, whether an NPI is present, and whether the NPI (or a similar item) is in the syntactic scope of the licensor (or a similar item). Crucially, each individual meta-data label by itself does not decide acceptability (i.e., these probing experiments test a different but related set of knowledge from acceptability experiments).

## 4 Data

In order to probe BERT's performance on sentences involving NPIs, we generate a set of sentences and acceptability labels for the experiments in this paper. We use generated data so that we can assess minimal pairs, and so that there are sufficient unacceptable sentences. We release all our data[3] and our generation code and vocabulary.[4]

**Licensing Features** We create a controlled set of 136,000 English sentences using an automated sentence generation procedure, inspired in large part by previous work by Ettinger et al. (2016, 2018), Marvin and Linzen (2018), Dasgupta et al. (2018), and Kann et al. (2019). The set contains nine NPI licensing environments (Table 1), and two NPIs (*any*, *ever*). All but one licensor-NPI pair follows a 2×2×2 paradigm, which manipulates three boolean NPI licensing features: licensor presence, NPI presence, and the occurrence of an NPI within a licensor's scope. Each 2×2×2 paradigm forms 5 minimal pairs. Table 2 shows an example paradigm.

The *Licensor* feature indicates whether an NPI licensor is present in the sentence. For many environments, there are multiple lexical items that serve as a licensor (e.g., the adverbs environment contains *rarely*, *hardly*, *seldom*, *barely*, and *scarcely* as NPI licensors). When the licensor is not present, we substitute it with a *licensor replacement* that has a similar syntactic distribution but does not license NPIs, again using multiple appropriate lexical items as replacements. For example, licensor replacements for quantifier licensors like *every* include quantifiers like *some*, *many*, and *more than three* that do not license NPIs.

The *NPI* feature indicates whether an NPI is in the sentence or if it is substituted by an *NPI replacement* with similar structural distribution. For example, NPI replacements for *ever* include other adverbs such as *often*, *sometimes*, and *certainly*.

The *Scope* feature indicates whether the NPI/NPI replacement is within the scope of the licensor/licensor replacement. As illustrated earlier in (3), a sentence containing an NPI is only acceptable when the NPI falls within the scope of the licensor. What constitutes the scope of a licensor is highly dependent on the type of licensor.

The exception to the 2×2×2 paradigm is the Simple Questions licensing condition, with a reduced 2×2 paradigm. It lacks a scope manipulation because the question takes scope over the entire clause, and in Simple Questions the clause is the whole sentence. The paradigm for Simple Questions is given in Table 3 in the Appendix; it forms only 2 minimal pairs.

**Data Generation** To generate sentences, we create sentence templates for each condition. Templates follow the general structure illustrated in example (5), in which the part-of-speech (auxiliary verb, determiner, noun, verb), as well as the instance number is specified. For example, N2 is the second instance of a noun in the template. We use these labels here for illustrative purposes; in reality, the templates include more fine-grained properties, such as verb tense and noun number.

[3] https://alexwarstadt.files.wordpress.com/2019/08/npi_lincensing_data.zip (Clicking on the link will cause a file to download.)

[4] https://github.com/alexwarstadt/data_generation

| Environment | Abbrev. | Example |
|---|---|---|
| Adverbs | ADV | The guests who **rarely** love *any* actors had left libraries. |
| Conditionals | COND | **If** the pedestrian passes *any* schools, the senator will talk to the adults. |
| Determiner negation | D-NEG | Just as the waitress said, **no** customers thought that *any* dancers bought the dish. |
| Sentential negation | S-NEG | These drivers have **not** thought that *any* customers have lied. |
| Only | ONLY | From what the cashier heard, **only** the children have known *any* dancers. |
| Quantifiers | QNT | **Every** actress who was talking about *any* high schools criticizes the children. |
| Questions | QUES | The boys wonder **whether** the doctors went to *any* art galleries. |
| Simple questions | SMP-Q | **Has** the guy worked with *any* teenagers**?** |
| Superlatives | SUP | The teenagers approach the **nicest** actress that *any* customers had criticized. |

Table 1: Examples of each of the NPI licensing environments generated. The licensor is in bold, and the NPI (here *any*) is in italics. All examples show cases where the NPI is present (NPI=1), the licensor is present (Licensor=1), and the NPI is in the scope of the licensor (Scope=1); all are acceptable to native speakers of English.

| Licensor | NPI | Scope | Sentence |
|---|---|---|---|
| 1 | 1 | 1 | Those boys wonder **whether** [the doctors *ever* went to an art gallery.] |
| 1 | 1 | 0 | *Those boys *ever* wonder **whether** [the doctors went to an art gallery.] |
| 1 | 0 | 1 | Those boys wonder **whether** [the doctors *often* went to an art gallery.] |
| 1 | 0 | 0 | Those boys *often* wonder **whether** [the doctors went to an art gallery.] |
| 0 | 1 | 1 | *Those boys say **that** [the doctors *ever* went to an art gallery.] |
| 0 | 1 | 0 | *Those boys *ever* say **that** [the doctors went to an art gallery.] |
| 0 | 0 | 1 | Those boys say **that** [the doctors *often* went to an art gallery.] |
| 0 | 0 | 0 | Those boys *often* say **that** [the doctors went to an art gallery.] |

Table 2: Example 2×2×2 paradigm using the Questions environment. The licensor (*whether*) or licensor replacement (*that*) is in bold. The NPI (*ever*) or NPI replacement (*often*) is in italics. When licensor=1, the licensor is present rather than its replacement word. When NPI=1, the NPI is present rather than its replacement. The scope of the licensor/licensor replacement is shown in square brackets (brackets, italicization, and boldface are not present in the actual data). When scope=1, the NPI/NPI replacement is within the scope of the licensor/licensor replacement. Unacceptable sentences are marked with *. The five minimal pairs are connected by arrows that point from the unacceptable to the acceptable sentence.

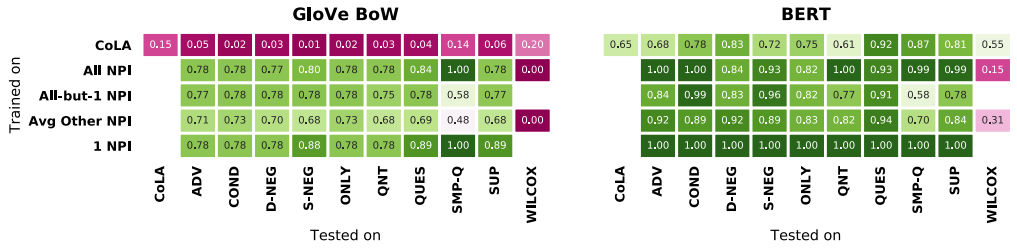


Figure 1: Results from the acceptability judgment experiment in MCC. The columns indicate evaluation tests, and the rows fine-tuning settings.

(5) Aux1 D1 N1 V1 any N2 ?
Has the guy seen any waitresses ?

Given the specifications encoded in the sentence templates, words were sampled from a vocabulary of over 1000 lexical items annotated with 30 syntactic, morphological, and semantic features. The annotated features allow us to encode selectional requirements of lexical items, e.g., what types of nouns a verb can combine with. This avoids blatantly implausible sentences.

For each environment, the training set contains 10K sentences, and the dev and test sets contain 1K sentences each. Sentences from the same paradigm are always in the same set.

In addition to our data set, we also test BERT on a set of 104 handcrafted sentences from the NPI sub-experiment in Wilcox et al. (2019), who use a paradigm that partially overlaps with ours, but has an additional condition where the NPI linearly follows its licensor while not being in the scope of the licensor. This is included as an additional test set for evaluating acceptability classifiers in (6).

**Data validation** We use Amazon Mechanical Turk (MTurk) to validate a subset of our data to assure that the generated sentences contrast in acceptability as intended. We randomly sample 500 sentences in approximately equal amounts from each environment, NPI and paradigm. Each sentence is rated on a Likert scale of 1-6, with 1 being "the sentence is not possible in English" and 6 being "the sentence is possible in English" by 20 unique participants in the US who self-identified as native English speakers. Participants are compensated $0.25 per HIT and are often able to complete a HIT of 5 sentences in just under 1 minute.

Table 4 in the Appendix shows the participants' scores transformed into a Boolean judgment of 0 (unacceptable, score $\leq 3$) or 1 (acceptable, score $\geq 4$) and presented as the percentage of 'acceptable' ratings assigned to the sentences in each of the NPI-licensing environments. Across all NPI-licensing environments, 81.3% of the sentences labelled acceptable are assigned an acceptable rating by the MTurk raters, and 85.2% of sentences labeled unacceptable are assigned an unacceptable rating. This gives an overall agreement rating of 82.8%[5]. A Wilcoxon signed-rank test (Wilcoxon, 1945) shows that within each environment and for each NPI, the acceptable sentences are more often rated as acceptable by our MTurk validators than the unacceptable sentences (all $p$-values $< 0.001$). This contrast holds considering both the raw Likert-scale responses and the responses transformed to a Boolean judgment.

## 5 Experimental Settings

We conduct our experiments with the jiant 0.9 [6] (Wang et al., 2019) multitask learning and transfer learning toolkit, the AllenNLP platform (Gardner et al., 2018), and the BERT implementation from HuggingFace.[7]

**Models** We study the following sentence understanding models: (i) *GloVe BoW*: a bag-of-words baseline obtained by max-pooling of 840B tokens 300-dimensional GloVe word embeddings (Pennington et al., 2014) and (ii) *BERT* (Devlin et al., 2018): we use the cased version of BERT-large model, which works the best for our tasks in pilot experiments. In addition, since recent work (Liu et al., 2019; Stickland and Murray, 2019) has shown that intermediate training on related tasks can meaningfully impact BERT's performance on downstream tasks, we also explore two additional BERT-based models—(iii) *BERT→MNLI*: BERT fine-tuned on the Multi-Genre Natural Language Inference corpus (Williams et al., 2018), motivated both by prior work on pretraining sentence encoders on MNLI (Conneau et al., 2017) as well as work showing significant improvements to BERT on downstream semantic tasks (Phang et al., 2018; Bowman et al., 2018) (iv) *BERT→CCG*: BERT fine-tuned on Combinatory Categorial Grammar Bank corpus (Hockenmaier and Steedman, 2007), motivated by Wilcox et al.'s (2019) finding that structural supervision may improve a LSTM-based sentence encoders knowledge on non-local syntactic dependencies.

**Training-Evaluation Configurations** We are interested in whether sentence representation models learn NPI licensing as a unified property. Can the models generalize from trained environments to previously unseen environments? To answer these questions, for each NPI environment, we extensively test the performance of the models in the following configurations: (i) *CoLA*: training on CoLA, evaluating on the environment. (ii) *1 NPI*: training and evaluating on the same NPI environment. (iii) *Avg Other NPI*: training independently on every NPI environment except one, averaged over the evaluation results on that environment. (iv) *All-but-1 NPI*: training on all environments except for one environment, evaluating on that environment. (v) *All NPI*: training on all environments, evaluating on the environment.

## 6 Results

**Acceptability Judgments** The results in Fig. 1 show that BERT outperforms the BoW baseline on all test data with all fine-tuning settings. Within each BERT variant, MCC reaches 1.0 on all test data in the *1 NPI* setting. In the *All-but-1 NPI* training-evaluation configuration, the performance on all NPI environments for BERT drops. While the MCC value on environments like conditionals and sentential negation remains above 0.9, on the simple question environment it drops to 0.58.

[5] We observe that sentences with 'any' get over-accepted. Overall agreement for sentences with 'ever' is 87.3%, while agreement for those with 'any' is 78.3%. We believe this is due to a free-choice interpretation of 'any' occurring more easily than is often reported in the semantics literature.

[6] https://github.com/nyu-mll/jiant/tree/blimp-and-npi/scripts/bert_npi

[7] https://github.com/huggingface/pytorch-pretrained-BERT




Figure 2: Results from the minimal pair test. The top section shows the average accuracy for detecting the presence of the NPI, the middle section shows average accuracy for detecting the presence of the licensor, and the bottom shows average accuracy of minimal pair contrasts that differ in whether the NPI is in scope of the licensor. Within each section, we show performance of GloVe BoW and BERT models under both absolute preference and gradient preference evaluation methods. The rows represent the training-evaluation configuration, while the columns represent different licensing environments.

Compared with NPI data fine-tuning, CoLA fine-tuning results in BERT's lower performance on most of the NPI environments but better performance on data from Wilcox et al. (2019).

In comparing the three BERT variants (see full results in Figure 5 in the Appendix), the *Avg Other NPI* shows that on 7 out of 9 NPI environments, plain BERT outperforms BERT→MNLI and BERT→CCG. Even in the remaining two environments, plain BERT yields about as good performance as BERT→MNLI and BERT→CCG, indicating that MNLI and CCG fine-tuning brings no obvious gain to acceptability judgments.

**Absolute and Gradient Minimal Pairs** The results (Fig. 2) show that models' performance hinges on how minimal pairs differ. When tested on minimal pairs differing by the presence of an NPI, BoW and plain BERT obtain (nearly) perfect accuracy on both absolute and gradient measures across all settings. For minimal pairs differing by licensor and scope, BERT again achieves near perfect performance on the gradient measure, while BoW does not. On the absolute measure, both BERT and BoW perform worse. Overall, it

shows that absolute judgment is more challenging when targeting licensor, which involves a larger pool of lexical items and syntactic configurations than NPIs; and scope, which requires nontrivial syntactic knowledge about NPI licensing.

As in the acceptability experiment, we find that intermediate fine-tuning on MNLI and CCG does not improve performance (see full results in Figures 6-8 in Appendix).

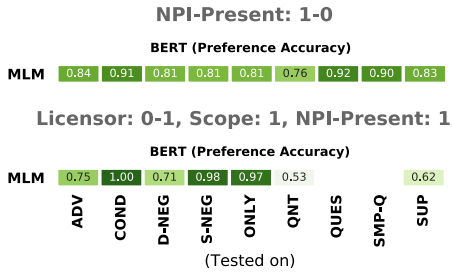


Figure 3: Results of BERT MLM performance in the cloze test. The top section shows the average accuracy for detecting the presence of the NPI; the bottom section show the accuracy for detecting the presence of the licensor. The columns represent different licensing environments

**Cloze Test** The results (Fig. 3) show that even without supervision on NPI data, the BERT MLM can distinguish between acceptable and unacceptable sentences in the NPI domain. Performance is highly dependent on the NPI-licensing environment and type of minimal pair. Accuracy for detecting NPI presence falls between 0.76 and 0.93 for all environments. Accuracy for detecting licensor presence is much more variable, with the BERT MLM achieving especially high performance in conditional, sentential negation, and *only* environments; and low performance in quantifier and superlative environments.

**Feature Probing** Results (Fig. 4) show that plain BERT outperforms the BoW baseline in detecting whether NPI is in the scope of the licensor (henceforth 'scope detection'). As expected, BoW is nearly perfect in detecting presence of NPI and licensor, as these tasks do not require knowledge of syntax or word order. Consistent with results from previous experiments, detecting the presence of a licensor is slightly more challenging for models fine-tuned with CoLA or NPI data. However, the overall lower performances in scope detection compared with detecting the presence of the licensor is not found in the minimal-pair experiments.

CoLA fine-tuning improves the performance for BERT, especially for detecting NPI presence. Fine-tuning on NPI data improves scope detection. Inspection of environment-specific results shows that models struggle when the superlative, quantifiers, and adverb environments are the held-out test sets in the All-but-1 NPI fine-tuning setting.

Different from other experiments, BERT and BERT→MNLI have comparable performance across many settings and tasks, beating BERT→CCG especially in scope detection (see full results in Figure 9 in the Appendix).

## 7 Discussion

We find that BERT systematically represents all features relevant to NPI licensing across most individual licensing environments, according to certain evaluation methods. However, these results vary widely across the different methods we compare. In particular, BERT performs nearly perfectly on the gradient minimal pairs task across all of minimal pair configurations and nearly all licensing environments. Based on this method alone, we might conclude that BERT's knowledge of this domain is near perfect. However, the other methods show a more nuanced picture.

BERT's knowledge of which expressions are NPIs and NPI licensors is generally stronger than its knowledge of the licensors' scope. This is especially apparent from the probing results (Fig. 4). BERT without acceptability fine-tuning performs close to ceiling on detecting the presence of a licensor, but is inconsistent at scope detection. Tellingly, the BoW baseline is also able to perform at ceiling on telling whether a licensor is present. For BoW to succeed at this task, the GloVe embeddings for NPI-licensors must share some common property, most likely the fact that licensors co-occur with NPIs. It is possible that BERT is able to succeed using a similar strategy. By contrast, identifying whether an NPI is in the scope of a licensor requires at the very least word order information and not just co-occurrences.

The contrast in BERT's performance on the gradient and absolute tasks tells us that these evaluations reveal different aspects of BERT's knowledge. The gradient task is strictly easier than the absolute task. On the one hand, BERT's high performance on the gradient task reveals the presence of systematic knowledge in the NPI domain. On the other hand, due to ceiling effects, the gradi-

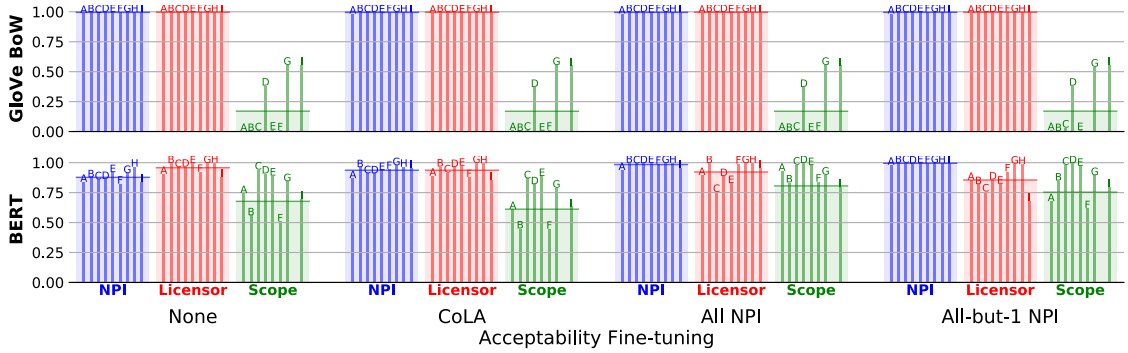


Figure 4: Results of probing classification on NPI presence, licensor presence, and whether the NPI is in the scope of the licensor (scope detection), shown in MCC. Letters on top of bars refer to NPI environments: A=ADV, B=COND, C=D-NEG, D=S-NEG, E=ONLY, F=QNT, G=QUES, H=SMP-Q, I=SUP.

ent task fails to reveal actual differences between NPI-licensing environments that we clearly observe based on absolute, cloze, and probing tasks.

While BERT has systematic knowledge of acceptability contrasts, this knowledge varies across different licensing environments and is not categorical. Generative models of syntax (Chomsky, 1965, 1981, 1995) model human knowledge of natural language as categorical: In that sense BERT fails at attaining human performance. However, some have argued that acceptability is inherently gradient (Lau et al., 2017), and results from the human validation study on our generated dataset show evidence of gradience in the acceptability of sentences in our dataset.

Supplementing BERT with additional pretraining on CCG and MNLI does not improve performance, and even lowers performance in some cases. While results from Phang et al. (2018) lead us to hypothesize that intermediate pretraining might help, this is not what we observe on our data. This result is in direct contrast with the results from Wilcox et al. (2019), who find that syntactic pretraining does improve performance in the NPI domain. This difference in findings is likely due to differences in models and training procedure, as their model is an RNN jointly trained on language modeling and parsing over the much smaller Penn Treebank (Marcus et al., 1993).

Future studies would benefit from employing a variety of different methodologies for assessing model performance withing a specified domain. In particular, a result showing generally good performance for a model should be regarded as possibly hiding actual differences in performance that a different task would reveal. Similarly, generally poor performance for a model does not necessarily mean that the model does not have systematic knowledge in a given domain; it may be that an easier task would reveal systematicity.

## 8 Conclusion

We have shown that within a well-defined domain of English grammar, evaluation of sentence encoders using different tasks will reveal different aspects of the encoder's knowledge in that domain. By considering results from several evaluation methods, we demonstrate that BERT has systematic knowledge of NPI licensing. However, this knowledge is unequal across the different features relevant to this phenomenon, and does not reflect the Boolean effect that these features have on acceptability.

## Acknowledgments

This project was a joint effort by the participants in the Spring 2019 NYU Linguistics seminar course *Linguistic Knowledge in Reusable Sentence Encoders*. We are grateful to the department for making this seminar possible.

This material is based upon work supported by the National Science Foundation under Grant No. 1850208. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. This project has also benefited from financial support to SB by Samsung Research under the project *Improving Deep Learning using Latent Structure* and from the donation of a Titan V GPU by NVIDIA Corporation.

## References


Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2016. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. *arXiv preprint arXiv:1608.04207*.

Samuel R. Bowman, Ellie Pavlick, Edouard Grave, Benjamin Van Durme, Alex Wang, Jan Hula, Patrick Xia, Raghavendra Pappagari, R. Thomas McCoy, Roma Patel, Najoung Kim, Ian Tenney, Yinghui Huang, Katherin Yu, Shuning Jin, and Berlin Chen. 2018. Looking for elmo's friends: Sentence-level pretraining beyond language modeling. *CoRR*, abs/1812.10860.

Gennaro Chierchia. 2013. *Logic in grammar: Polarity, free choice, and intervention*. Oxford University Press.

Noam Chomsky. 1965. *Aspects of the Theory of Syntax*. MIT Press.

Noam Chomsky. 1981. *Lectures on Government and Binding*. Number 9 in Studies in Generative Grammar. Foris Publications, Dordrecht, Holland ; Cinnaminson, [N.J.].

Noam Chomsky. 1995. *The Minimalist Program*. Number 28 in Current Studies in Linguistics. The MIT Press, Cambridge, Mass.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *CoRR*, abs/1705.02364.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single vector: Probing sentence embeddings for linguistic properties. *arXiv preprint arXiv:1805.01070*.

Ishita Dasgupta, Demi Guo, Andreas Stuhlmüller, Samuel J Gershman, and Noah D Goodman. 2018. Evaluating compositionality in sentence embeddings. *arXiv preprint arXiv:1802.04302*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint 1810.04805*.

Allison Ettinger, Ahmed Elgohary, and Philip Resnik. 2016. Probing for semantic evidence of composition by means of simple classification tasks. In *Proceedings of the First Workshop on Evaluating Vector Space Representations for NLP*, pages 134–139. Association for Computational Linguistics.

Allyson Ettinger, Ahmed Elgohary, Colin Phillips, and Philip Resnik. 2018. Assessing composition in sentence vector representations. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1790–1801. Association for Computational Linguistics.

Gilles Fauconnier. 1975. Polarity and the scale principle. *Proceedings of the Chicago Linguistics Society*, 11:188–199.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6.

Anastasia Giannakidou. 1994. The semantic licensing of NPIs and the Modern Greek subjunctive. *Language and Cognition*, 4:55–68.

Anastasia Giannakidou. 1998. *Polarity sensitivity as (non) veridical dependency*, volume 23. John Benjamins Publishing.

Mario Giulianelli, Jack Harding, Florian Mohnert, Dieuwke Hupkes, and Willem H. Zuidema. 2018. Under the hood: Using diagnostic classifiers to investigate and improve how language models track agreement information. *CoRR*, abs/1808.08079.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Jaap Jumelet and Dieuwke Hupkes. 2018. Do language models understand anything? on the ability of lstms to understand negative polarity items. *arXiv preprint arXiv:1808.10627*.

Nirit Kadmon and Fred Landman. 1993. Any. *Linguistics and Philosophy*, 16(4):353–422.

Katharina Kann, Alex Warstadt, and Adina Williams. 2019. Verb argument structure alternations in word and sentence embeddings. In *Proceedings of the Society for Computation in Linguistics (SCiL)*, pages 287–297.

Manfred Krifka. 1995. The semantics and pragmatics of polarity items. *Linguistic Analysis*, 25(3-4):209–257.

William Ladusaw. 1979. Negative polarity items as inherent scope relations. *Unpublished Ph.D. Dissertation, University of Texas at Austin*.

Jey Han Lau, Alexander Clark, and Shalom Lappin. 2017. Grammaticality, acceptability, and probability: A probabilistic view of linguistic knowledge. *Cognitive Science*, 41(5):1202–1241.

Marcia Christine Linebarger. 1980. *The grammar of negative polarity*. Ph.D. thesis, Massachusetts Institute of Technology.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of lstms to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. *CoRR*, abs/1901.11504.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank.

Rebecca Marvin and Tal Linzen. 2018. Targeted syntactic evaluation of language models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202.

Brian W Matthews. 1975. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Jason Phang, Thibault Févry, and Samuel R. Bowman. 2018. Sentence encoders on STILTs: Supplementary training on intermediate labeled-data tasks. In *arXiv preprint*.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural mt learn source syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1526–1534.

Asa Cooper Stickland and Iain Murray. 2019. BERT and PALs: Projected attention layers for efficient adaptation in multi-task learning. *CoRR*, abs/1902.02671.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Sam Bowman, Dipanjan Das, and Ellie Pavlick. 2019. What do you learn from context? Probing for sentence structure in contextualized word representations. In *International Conference on Learning Representations*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355.

Alex Wang, Ian F. Tenney, Yada Pruksachatkun, Katherin Yu, Jan Hula, Patrick Xia, Raghu Pappagari, Shuning Jin, R. Thomas McCoy, Roma Patel, Yinghui Huang, Jason Phang, Edouard Grave, Najoung Kim, Phu Mon Htut, Thibault F'evry, Berlin Chen, Nikita Nangia, Haokun Liu, Anhad Mohananey, Shikha Bordia, Ellie Pavlick, and Samuel R. Bowman. 2019. jiant 0.9: A software toolkit for research on general-purpose text understanding models. `http://jiant.info/`.

Alex Warstadt and Samuel R Bowman. 2019. Grammatical analysis of pretrained sentence encoders with acceptability judgments. *arXiv preprint arXiv:1901.03438*.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2018. Neural network acceptability judgments. *arXiv preprint 1805.12471*.

Ethan Wilcox, Peng Qian, Richard Futrell, Miguel Ballesteros, and Roger Levy. 2019. Structural supervision improves learning of non-local grammatical dependencies. In *Proceedings of the 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Frank Wilcoxon. 1945. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122.

Frans Zwarts. 1998. Three types of polarity. In *Plurality and Quantification*, pages 177–238. Springer.

# Representation of Constituents in Neural Language Models: Coordination Phrase as a Case Study

**Aixiu An**[1] **Peng Qian**[2] **Ethan Wilcox**[3] **Roger Levy**[2]
[1] Université de Paris, LLF, CNRS, aixiu.an@etu.univ-paris-diderot.fr
[2] Department of Brain and Cognitive Sciences, MIT, {pqian, rplevy}@mit.edu
[3] Department of Linguistics, Harvard University, wilcoxeg@g.harvard.edu

## Abstract

Neural language models have achieved state-of-the-art performances on many NLP tasks, and recently have been shown to learn a number of hierarchically-sensitive syntactic dependencies between individual words. However, equally important for language processing is the ability to combine words into phrasal constituents, and use constituent-level features to drive downstream expectations. Here we investigate neural models' ability to represent constituent-level features, using coordinated noun phrases as a case study. We assess whether different neural language models trained on English and French represent phrase-level number and gender features, and use those features to drive downstream expectations. Our results suggest that models use a linear combination of NP constituent number to drive CoordNP/verb number agreement. This behavior is highly regular and even sensitive to local syntactic context, however it differs crucially from observed human behavior. Models have less success with gender agreement. Models trained on large corpora perform best, and there is no obvious advantage for models trained using explicit syntactic supervision.

## 1 Introduction

Humans deploy structure-sensitive expectations to guide processing during natural language comprehension (Levy, 2008). While it has been shown that neural language models show similar structure-sensitivity in their predictions about upcoming material (Linzen et al., 2016; Futrell et al., 2018), previous work has focused on dependencies that are conditioned by features attached to a single word, such as subject number (Gulordava et al., 2018; Marvin and Linzen, 2018) or wh-question words (Wilcox et al., 2018). There has been no systematic investigation into models' ability to compute phrase-level features—features that are attached to a set of words—and whether models can deploy these more abstract properties to drive downstream expectations.

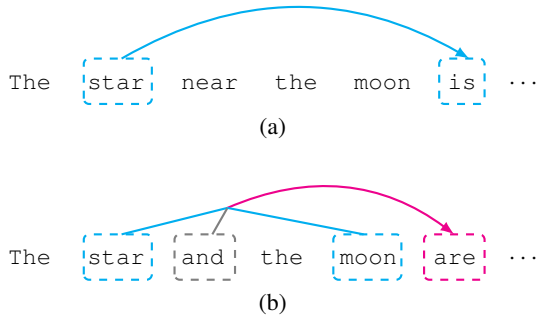


Figure 1: Subject-verb agreement with (a) the head of a noun phrase structure, and (b) the coordination structure.

In this work, we assess whether state-of-the-art neural models can compute and employ phrase-level gender and number features of coordinated subject Noun Phrases (CoordNPs) with two nouns. Typical syntactic phrases are ENDOCENTRIC: they are HEADED by a single child, whose features determine the agreement requirements for the entire phrase. In Figure 1a, for example, the word *star* heads the subject NP *The star*; since *star* is singular, the verb must be singular. CoordNPs lack endocentricity: neither conjunct NP solely determines the features of the NP as a whole. Instead, these feature values are determined by compositional rules sensitive to the features of the conjuncts and the identity of the coordinator. In Figure 1b, because the coordinator is *and*, the subject NP number is plural even though both conjuncts (*the star* and *the moon*) are singular. As this case demonstrates, the agreement behavior for CoordNPs must be driven by more abstract, constituent-level representations, and cannot be reduced to features hosted on a single lexical item.

We use four suites of experiments to assess



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2888–2899, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

whether neural models are able to build up phrase-level representations of CoordNPs on the fly and deploy them to drive humanlike behavior. First, we present a simple control experiment to show that models can represent number and gender features of non-coordinate NPs **(Non-coordination Agreement)**. Second, we show that models modulate their expectations for downstream verb number based on the CoordNP's coordinating conjunction combined with the features of the coordinated nouns **(Simple Coordination)**. We rule out the possibility that models are using simple heuristics by designing a set of stimuli where a simple heuristic would fail due to structural ambiguity **(Complex Coordination)**. The striking success for all models in this experiment indicates that even neural models with no explicit hierarchical bias, trained on a relatively small amount of text are able to learn fine-grained and robust generalizations about the interaction between CoordNPs and local syntactic context. Finally, we use subject–auxiliary inversion to test whether an upstream lexical item modulates model expectation for the phrasal-level features of a downstream CoordNP **(Inverted Coordination)**. Here, we find that all models are insensitive to the fine-grained features of this particular syntactic context. Overall, our results indicate that neural models can learn fine-grained information about the interaction of Coordinated NPs and local syntactic context, but their behavior remains unhumanlike in many key respects.

## 2 Methods

### 2.1 Psycholinguistics Paradigm

To determine whether state-of-the-art neural architectures are capable of learning humanlike CoordNP/verb agreement properties, we adopt the psycholinguistics paradigm for model assessment. In this paradigm the models are tested using handcrafted sentences designed to test underlying network knowledge. The assumption here is that if a model implicitly learns humanlike linguistic knowledge during training, its expectations for upcoming words should qualitatively match human expectations in novel contexts. For example, Linzen et al. (2016) and Kuncoro et al. (2016) assessed how well neural models had learned the subject/verb number agreement by feeding them with the prefix *The keys to the cabinet ....* If the models predicted the grammatical continuation *are* over the ungrammatical continuation *is*, they can be said to have learned the number agreement insofar as the number of the head noun and not the number of the distractor noun, *cabinet*, drives expectations about the number of the matrix verb.

| | Model | Training data | # tokens |
|---|---|---|---|
| English | LSTM (PTB) | Penn Treebank | $\sim$ 1M |
| | ActionLSTM (PTB) | Penn Treebank | $\sim$ 1M |
| | RNNG (PTB) | Penn Treebank | $\sim$ 1M |
| | LSTM (enWiki) | English Wikipedia | $\sim$ 90M |
| | LSTM (1B) | 1 Billion Word | $\sim$ 800M |
| French | LSTM (FTB) | French Teebank | $\sim$ 0.5M |
| | ActionLSTM (FTB) | French Teebank | $\sim$ 0.5M |
| | RNNG (FTB) | French Teebank | $\sim$ 0.5M |
| | LSTM (frWaC) | Subset of frWaC | $\sim$ 138M |

Table 1: A summary of models tested.

If models are able to robustly modulate their expectations based on the internal components of the CoordNP, this will provide evidence that the networks are building up a context-sensitive phrase-level representation. We quantify model expectations as SURPRISAL VALUES. Surprisal is the negative log-conditional probability $S(x_i) = -\log_2 p(x_i|x_1 \dots x_{i-1})$ of a sentence's $i^{th}$ word $x_i$ given the previous words. Surprisal tells us how strongly $x_i$ is expected in context and is known to correlate with human processing difficulty (Hale, 2001; Levy, 2008; Smith and Levy, 2013). In the CoordNP/Verb agreement studies presented here, cases where the proceeding context sets high expectation for a number-inflected verb form $w_i$, (e.g. singular 'is') we would expect $S(w_i)$ to be lower than its number-mismatched counterpart (e.g. plural 'are').

### 2.2 Models Tested

**Recurrent Neural Network (RNN) Language Models** are trained to output the probability distribution of the upcoming word given a context, without explicitly representing the structure of the context (Sundermeyer et al., 2012; Elman, 1990). We trained two two-layer recurrent neural language models with long short-term memory architecture (Hochreiter and Schmidhuber, 1997) on a relatively small corpus. The first model, referred as '**LSTM (PTB)**' in the following sections, was trained on the sentences from Penn Treebank (Marcus and Marcinkiewicz). The second model, referred as '**LSTM (FTB)**', was trained on the sentences from French Treebank (Abeillé et al., 2003). We set the size of input word embedding and LSTM hidden layer of both models as 256.

We also compare LSTM language models

trained on large corpora. We incorporate two pre-trained English language models: one trained on the Billion Word benchmark (referred as '**LSTM (1B)**') from Jozefowicz et al. (2016), and the other trained on English Wikipedia (referred as '**LSTM (enWiki)**') from Gulordava et al. (2018). For French, we trained a large LSTM language model (referred as '**LSTM (frWaC)**') on a random subset (about 4 million sentences, 138 million word tokens) of the frWaC dataset (Baroni et al., 2009). We set the size of the input embeddings and hidden layers to 400 for the LSTM (frWaC) model since it is trained on a large dataset.

**ActionLSTM** models the linearized bracketed tree structure of a sentence by learning to predict the next action required to construct a phrase-structure parse (Choe and Charniak, 2016). The action space consists of three possibilities: open a new non-terminal node and opening bracket; generate a terminal node; and close a bracket. To compute surprisal values for a given token, we approximate $P(w_i|w_{1\cdots i-1})$ by marginalizing over the most-likely partial parses found by word-synchronous beam search (Stern et al., 2017).

**Generative Recurrent Neural Network Grammars (RNNG)** jointly model the word sequence as well as the underlying syntactic structure (Dyer et al., 2016). Following Hale et al. (2018), we estimate surprisal using word-synchronous beam search (Stern et al., 2017). We use the same hyper-parameter settings as Dyer et al. (2016).

The annotation schemes used to train the syntactically-supervised models differ slightly between French and English. In the PTB (English) CoordNPs are flat structures bearing an 'NP' label. In FTB (French), CoordNPs are binary-branching, labeled as NPs, except for the phrasal node dominating the coordinating conjunction, which is labeled 'COORD'. We examine the effects of annotation schemes on model performance in Appendix A. [1]

## 3 Experiment 1: Non-coordination Agreement

In order to provide a baseline for following experiments, here we assess whether the models tested have learned basic representations of number and gender features for non-coordinated Noun Phrases. We test number agreement in English and French as well as gender agreement in French. Both English and French have two grammatical number feature: SINGULAR (sg) and PLURAL (pl). French has two grammatical gender features: MASCULINE (m) and FEMININE (f).

| Condition | Sentence |
|---|---|
| Npl | The windows is/are |
| Nsg | The window is/are |

Table 2: Conditions of number agreement in Non-coordination Agreement experiment.

| Condition | Sentence |
|---|---|
| Nm | Les coûts sont importants/importantes<br>the cost.MPL are important.MPL/FPL |
| Nf | Les dépenses sont importants/importantes<br>the expense.FPL are important.MPL/FPL |

Table 3: Conditions of gender agreement in Non-coordination Agreement experiment.

The experimental materials include sentences where the subject NPs contain a single noun which can either match with the matrix verb (in the case of number agreement) or a following predicative adjective (in the case of gender agreement). Conditions are given in Table 2 and Table 3. We measure model behavior by computing the *plural expectation*, or the surprisal of the singular continuation minus the surprisal of the plural continuation for each condition and took the average for each condition. We expect a positive *plural expectation* in the *Npl* conditions and a negative *plural expectation* in the *Nsg* conditions. For gender expectation we compute a *gender expectation*, which is S(feminine continuation) − S(masculine continuation). We measure surprisal at the verbs and predicative adjectives themselves.

The results for this experiment are in Figure 2, with the *plural expectation* and *gender expectation* on the y-axis and conditions on the x-axis. For this and subsequent experiments error bars represent 95% confidence intervals for across-item means. For number agreement, all the models in English and French show positive plural expectation when the head noun is plural and negative plural expectation when it is singular. For gender agreement, however, only the LSTM (frWaC) shows modulation of gender expectation based on the gender of the head noun. This is most likely due to the lower

[1] The materials and code for this project can be found in https://github.com/cpllab/rnn_psycholing_coordination.git

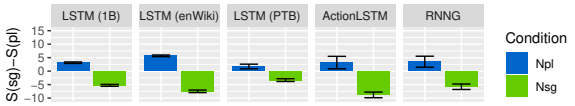


(a) English number agreement

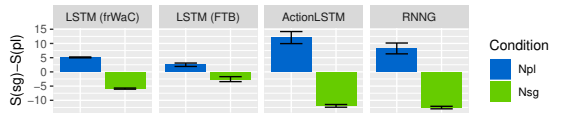


(b) French number agreement

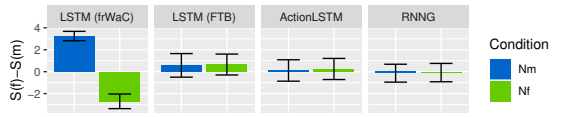


(c) French gender agreement

Figure 2: Non-Coordination Agreement experiments for English (number) and French (number and gender).

frequency of predicative adjectives compared to matrix verbs in the corpus.

## 4 Experiment 2: Simple Coordination

In this section, we test whether neural language models can use grammatical features hosted on multiple components of a coordination phrase—the coordinated nouns as well as the coordinating conjunction—to drive downstream expectations. We test number agreement in both English and French and gender agreement in French.

### 4.1 Number Agreement

In simple subject/verb number agreement, the number features of the CoordNP are determined by the coordinating conjunction and the number features of the two coordinated NPs. CoordNPs formed by *and* are plural and thus require plural verbs; CoordNPs formed by *or* allow either plural or singular verbs, often with the number features of the noun linearly closest to the verb playing a more important role, although this varies cross-linguistically (Fowler et al., 1992). Forced-choice preference experiments in Keung and Staub (2018) reveal that English native speakers prefer singular agreement when the closest conjunct in an *or*-CoordNP is singular and plural agreement when the closest conjunct is plural. In French, both singular and plural verbs are possible when two singular NPs are joined via disjunction (Goosse and Grevisse, 2016).

In order to assess whether the neural models learn the basic CoordNP licensing for English, we

| Condition | Sentence |
|---|---|
| pl_and_pl | The doors and the windows is/are |
| sg_and_pl | The door and the windows is/are |
| pl_and_sg | The doors and the window is/are |
| sg_and_sg | The door and the window is/are |
| pl_or_pl | The doors or the windows is/are |
| sg_or_pl | The door or the windows is/are |
| pl_or_sg | The doors or the window is/are |
| sg_or_sg | The door or the window is/are |

Table 4: Conditions of number agreement in Simple Coordination experiment.

adapted 37 items from Keung and Staub (2018), along the 16 conditions outlined in Table 4. Test items consist of the sentence preamble, followed by either the singular or plural *BE* verb, half the time in present tense (*is/are*) and half the time in past tense *(was/were)*. We measured the plural expectation, following the procedure in Section 3. We created 24 items using the same conditions as the English experiment to test the models trained in French, using the $3^{rd}$ person singular and plural form of verb *aller*, 'to go' (*va, vont*). Within each item, nouns match in gender; across all conditions half the nouns are masculine, half feminine.

The results for this experiment can be seen in Figure 3, with the results for English on the left and French on the right. The results for *and* are on the top row, *or* on the bottom row. For all figures the y-axis shows the plural expectation, or the difference in surprisal between the *singular* condition and the *plural* condition. Turning first to **English-*and*** (Figure 3a), all models show plural expectation (the bars are significantly greater than zero) in the *pl_and_pl* and *sg_and_pl* conditions, as expected. For the *pl_and_sg* condition, only the LSTM (enWiki) and ActionLSTM are greater than zero, indicating humanlike behavior. For the *sg_and_sg* condition, only the LSTM (enWiki) model shows the correct plural expectation. For the **French-*and*** (Figure 3b), all models show positive plural expectation in all conditions, as expected, except for the LSTM (FTB) in the *sg_and_sg* condition.

Examining the results for **English-*or***, we find that all models demonstrate humanlike expectation in the *pl_or_pl* and *sg_or_pl* conditions. The LSTM (1B), LSTM (PTB), and RNNG models show zero or negative singular expectation for the *pl_or_sg* conditions, as expected. However the LSTM (enWiki) and ActionLSTM models show positive plural expectation in this condition, indi-

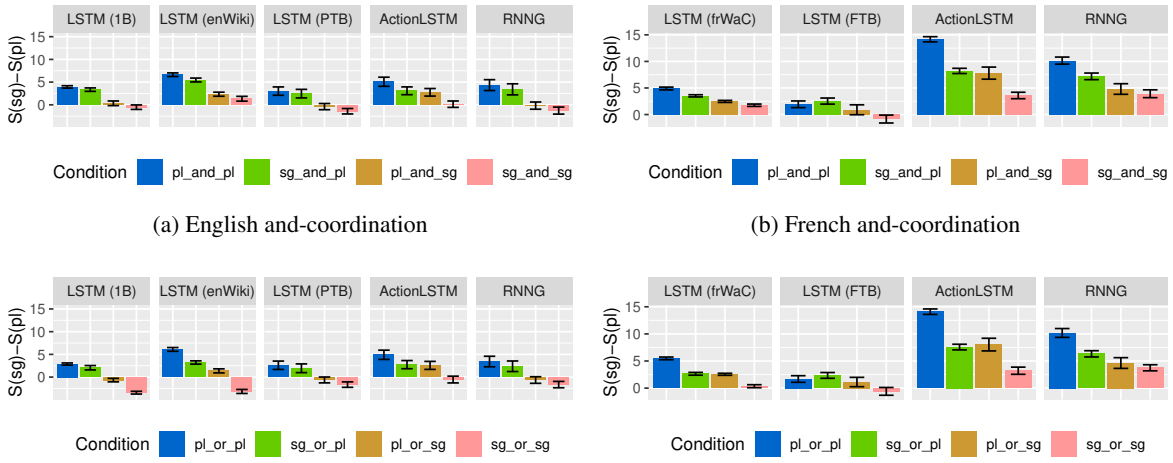


(a) English and-coordination

(b) French and-coordination

(c) English or-coordination

(d) French or-coordination

Figure 3: Comparison of models' expectation preferences for singular vs. plural predicate in English and French Simple Coordination experiments.

cating that they have not learned the humanlike generalizations. All models show significantly negative plural expectation in the *sg_or_sg* condition, as expected. In the **French-*or*** cases, models show almost identical behavior to the *and* conditions, except the LSTM (frWaC) shows smaller plural expectation when singular nouns are linearly proximal to the verb.

These results indicate moderate success at learning coordinate NP agreement, however this success may be the result of an overly simple heuristic. It appears that expectation for both plural and masculine continuations are driven by a linear combination of the two nominal number/gender features transferred into log-probability space, with the earlier noun mattering less than the later noun. A model that optimally captures human grammatical preferences should show no or only slight difference across conditions in the surprisal differential for the *and* conditions, and be greater than zero in all cases. Yet, all the models tested show gradient performance based on the number of plural conjuncts.

## 4.2 Gender Agreement

In French, if two nouns are coordinated with *et* (*and*-coordination), agreement must be masculine if there is one masculine element in the coordinate structure. If the nouns are coordinated with *ou* (*or*-coordination), both masculine and feminine agreement is acceptable (Corbett, 1991; Wechsler and Zlatić, 2003). Although linear proximity effects have been tested for a number of languages that employ grammatical gender, as in e.g. Slavic languages (Willer et al., 2018), there is no systematic study for French.

| Condition | Sentence |
|---|---|
| m_and_m | Les prix et les coûts sont importants/importantes<br>the price.MPL and the cost.MPL are important.MPL/FPL |
| f_and_m | Les recettes et les coûts sont importants/importantes<br>the revenues.FPL and the cost.MPL are important.MPL/FPL |
| m_and_f | Les prix et les dépenses sont importants/importantes<br>the price.MPL and the expense.FPL are important.MPL/FPL |
| f_and_f | Les recettes et les dépenses sont importants/importantes<br>the revenues.FPL and the expense.FPL are important.MPL/FPL |

Table 5: Conditions for the and-coordination experiment. (Items for or-coordination are the same except that we change the coordinator to *ou*.)

To assess whether the French neural models learned humanlike gender agreement, we created 24 test items, following the examples in Table 5, and measured the masculine expectation. In our test items, the coordinated subject NP is followed by a predicative adjective, which either takes on masculine or feminine gender morphology.

Results from the experiment can be seen in Figure 4. No models shows qualitative difference based on the coordinator, and only the LSTM (frWaC) shows significant behavior difference between conditions. Here, we find positive masculine expectation in the *m_and_m* and *f_and_m* conditions, and negative masculine expectation in the *f_and_f* condition, as expected. However, in the *m_and_f* condition, the masculine expectation is not significantly different from zero, where we would expect it to be positive. In the *or-*

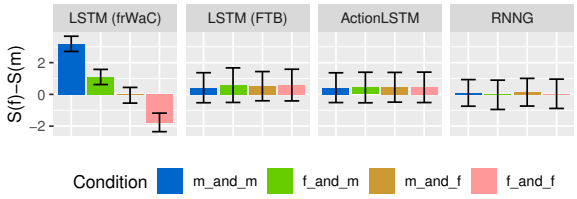


(a) French and-coordination

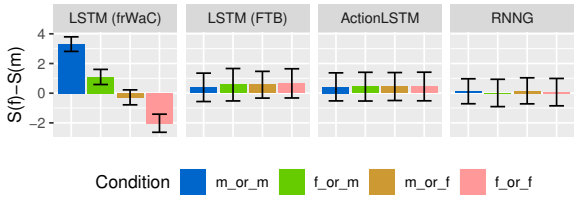


(b) French or-coordination

Figure 4: Comparison of models' expectation preferences for Feminine v.s. Masculine predicative adjectives in French.

coordination conditions, following our expectation, masculine expectation is positive when both conjuncts are masculine and negative when both are feminine. For the LSTM (FTB) and ActionLSTM models, the masculine expectation is positive (although not significantly so) in all conditions, consistent with results in Section 3.

## 5 Experiment 3: Complex Coordination

One possible explanation for the results presented in the previous section is that the models are using a 'bag of features' approach to plural and masculine licensing that is opaque to syntactic context: Following a coordinating conjunction surrounded by nouns, models simply expect the following verb to be plural, proportionally to the number of plural nouns.

In this section, we control for this potential confound by conducting two experiments: In the *Complex Coordination Control* experiments we assess models' ability to extend basic CoordNP licensing into sententially-embedded environments, where the CoordNP can serve as an embedded subject. In the *Complex Coordination Critical* experiments, we leverage the sentential embedding environment to demonstrate that when the CoordNPs cannot plausibly serve as the subject of the embedded phrase, models are able to suppress the previously-demonstrated expectations set up by these phrases. These results demonstrate that models are not following a simple strategy for predicting downstream number and gender features, but are building up CoordNP representations on the fly, conditioned on the local syntactic context.

### 5.1 Complex Coordination Control

Following certain sentential-embedding verbs, CoordNPs serve unambiguously as the subject of the verb's sentence complement and should trigger number agreement behavior in the main verb of the embedded clause, similar to the behavior presented in 4.1. To assess this, we use the 37 test items in English and 24 items in French in section 4.1, following the conditions in Table 6 (for number agreement), testing only *and* coordination. For gender agreement, we use the same test items and conditions for *and* coordination in Section 4.2, but with the Coordinated NPs embedded in a context similar to (1). As before, we derived the plural expectation by measuring the difference in surprisal between the singular and plural continuations and the gender expectation by computing the difference in surprisal between the masculine and feminine predicates.

(1) Je croyais que les prix et les
I thought that the.PL price.MPL and the.PL
dépenses étaient importants/importantes.
expense.FPL were important.MPL/FPL
I thought that the prices and the expenses were important.

| Condition | Sentence |
|---|---|
| pl_and_pl | I think that the doors and the windows is/are |
| sg_and_pl | I think that the door and the windows is/are |
| pl_and_sg | I think that the doors and the window is/are |
| sg_and_sg | I think that the door and the window is/are |

Table 6: Conditions of number agreement in Complex Coordination Control experiment.

The results for the control experiments can be seen in Figure 5, with English number agreement on the top row, French number agreement in the middle row and French gender agreement on the bottom. The y-axis shows either plural or masculine expectation, with the various conditions along the x-axis. For English number agreement, we find that the models behave similarly as they do for simple coordination contexts. All models show significant plural expectation when the closest noun is plural, with only two models demonstrating plural expectation in the *sg_and_sg* case. The French number agreement tests show simi-

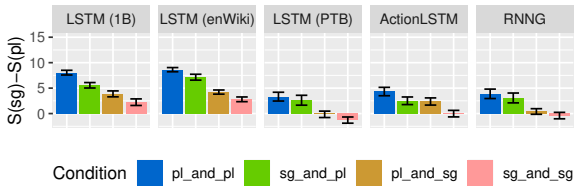


(a) English number agreement

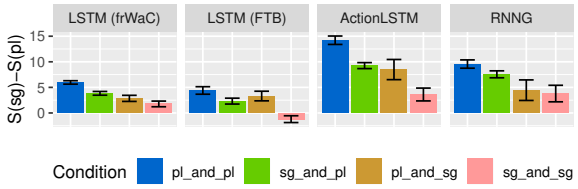


(b) French number agreement

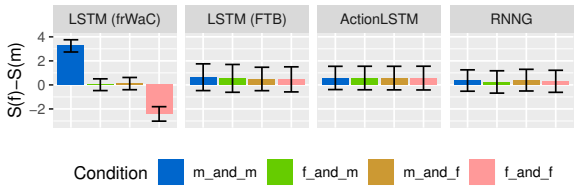


(c) French gender agreement

Figure 5: Comparison of model's expectation preferences in the Complex Coordination Control experiments.

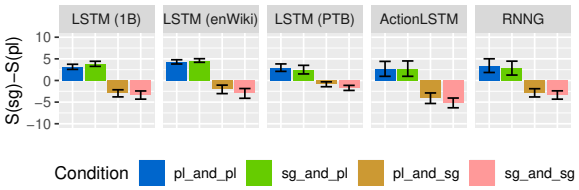


(a) English number agreement

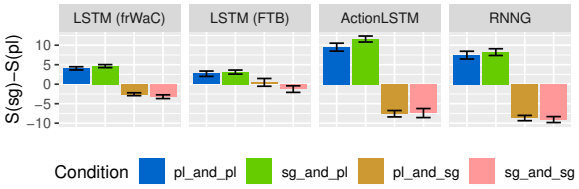


(b) French number agreement

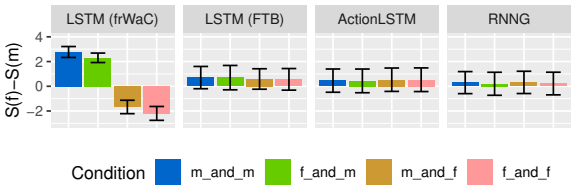


(c) French gender agreement

Figure 6: Comparison of model's expectation preferences in the Complex Coordination Critical experiments.

lar results, with all models except LSTM (FTB) demonstrating significant plural prediction in all cases. Turning to French gender agreement, only the LSTM (frWaC) shows sensitivity to the various conditions, with positive masculine expectation in the *m_and_m* condition and negative expectation in the *f_and_f* condition, as expected. These results indicate that the behavior shown in Section 4.1 extends to more complex syntactic environments—in this case to sentential embeddings. Interestingly, for some models, such as the LSTM (1B), behavior is *more humanlike* when the CoordNP serves as the subject of an embedded sentence. This may be because the model, which has a large number of hidden states and may be extra sensitive to fine-grained syntactic information carried on lexical items (Futrell et al., 2018), is using the complementizer, *that*, to drive more robust expectations.

### 5.2 Complex Coordination Critical

In order to assess whether the models' strategy for CoordNP/verb number agreement is sensitive to syntactic context, we contrast the results presented above to those from a second, critical experiment. Here, two coordinated nouns follow a verb that cannot take a sentential complement, as in the examples given in Table 7. Of the two possible continuations—*are* or *is*—the plural is only grammatically licensed when the second of the two conjuncts is plural. In these cases, the plural continuation may lead to a final sentence where the first noun serves as the verb's object and the second introduces a second main clause coordinated with the first, as in *I fixed the doors and the windows are still broken.* For the same reason, the singular-verb continuation is only licensed when the noun immediately following *and* is singular.

We created 37 test items in both English and French, and calculated the plural expectation. If the models were following a simple strategy to drive CoordNP/verb number agreement, then we should see either no difference in plural expectation across the four conditions or behavior no different from the control experiment. If, however, the models are sensitive to the licensing context, we should see a contrast based solely on the number features of the second conjunct, where plural expectation is positive when the second conjunct is plural, and negative otherwise.

| Condition | Sentence |
|---|---|
| pl_and_pl | I fixed the doors and the windows is/are |
| sg_and_pl | I fixed the door and the windows is/are |
| pl_and_sg | I fixed the doors and the window is/are |
| sg_and_sg | I fixed the door and the window is/are |

Table 7: Conditions of number agreement in Complex Coordination Critical experiment.

| Condition | Sentence preamble |
|---|---|
| Vpl_Npl | What are the doors and |
| Vpl_Nsg | What are the door and |
| Vsg_Nsg | What is the door and |

Table 8: Conditions in Inverted Coordination experiment.

Experimental items for a critical gender test were created similarly, as in Example (2). As with plural agreement, gender expectation should be driven solely by the second conjunct: For the *f_and_m* and *m_and_m* conditions, the only grammatical continuation is one where the adjectival predicate bears masculine gender morphology. Conversely, for the *m_and_f* or *f_and_f* conditions, the only grammatical continuation is one where the adjectival predicate bears feminine morphology. As in 4.1, we created 24 test items and measured the gender expectation by calculating the difference in surprisal between the masculine and feminine continuations.

(2) Nous avons accepté les prix et les
we have accepted the.PL price.MPL and the
dépenses étaient importants/importantes.
expense.FPL were important.MPL/FPL
We have accepted the prices and the expenses were important.

The results from the critical experiments are in Figure 6, with the English number agreement on the top row, French number agreement in the middle and gender expectation on the bottom row. Here the y-axis shows either plural expectation or masculine expectation, with the various conditions are on the x-axis. The results here are strikingly different from those in the control experiments. For number agreement, all models in both languages show strong plural expectation in conditions where the second noun is plural (blue and green bars), as they do in the control experiments. Crucially, when the second noun is singular, the plural expectation is significantly negative for all models (save for the French LSTM (FTB) *pl_and_sg* condition). Turning to gender agreement, only the LSTM (frWaC) model shows differentiation between the four conditions tested. However, whereas the *f_and_m* and *m_and_f* gender expectations are not significantly different from zero in the control condition, in the critical condition they pattern with the purely masculine and purely feminine conditions, indicating that, in this syntactic context, the model has successfully learned to base gender expectation solely off of the second noun.

These results are inconsistent with a simple 'bag of features' strategy that is insensitive to local syntactic context. They indicate that both models can interpret the same string as either a coordinated noun phrase, or as an NP object and the start of a coordinated VP with the second NP as its subject.

## 6 Experiment 4: Inverted Coordination

In addition to using phrase-level features to drive expectation about downstream lexical items, human processors can do the inverse—use lexical features to drive expectations about upcoming syntactic chunks. In this experiment, we assess whether neural models use number features hosted on a verb to modulate their expectations for upcoming CoordNPs.

To assess whether neural language models learn inverted coordination rules, we adapted items from Section 4.1 in both English (37 items) and French (24 items), following the paradigm in Table 8. The first part of the phrase contains either a plural or singular verb and a plural or singular noun. In this case, we sample the surprisal for the continuations *and* (*or* is grammatical in all conditions, so it is omitted from this study). Our expectation is that 'and' is less surprising in the *Vpl_Nsg* condition than in the *Vsg_Nsg* condition, where a CoordNP is not licensed by the grammar in either French or English (as in **What is the pig and the cat eating?*). We also expect lower surprisal for *and* in the *Vpl_Nsg* condition, where it is obligatory for a grammatical continuation, than in the *Vpl_Npl* condition, where it is optional.

For French experimental items, the question is embedded into a sentential-complement taking verb, following Example (3), due to the fact that unembedded subject-verb inverted questions sound very formal and might be relatively rare in the training data.

(3) Je me demande où vont le
I myself ask where go.3PL the.MSG
maire et
mayor.MSG and

The results for both languages are shown in Figure 7, with the surprisal at the coordinator on the y-axis and the various conditions on the x-axis. No model in either language shows a signficant difference in surprisal between the *Vpl_Nsg* and *Vpl_Npl* conditions or between the *Vpl_Nsg* and *Vsg_Nsg* conditions. The LSTM (1B) shows significant difference between the *Vpl_Nsg* and *Vpl_Npl* conditions, but in the opposite direction than expected, with the coordinator less surprising in the latter condition. These results indicate that the models are unable to use the fine-grained context sensitivity to drive expectations for CoordNPs, at least in the inversion setting.

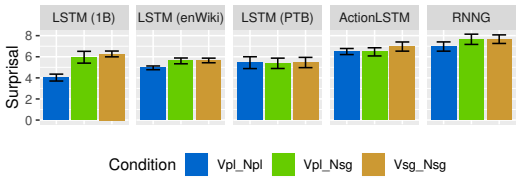


(a) English

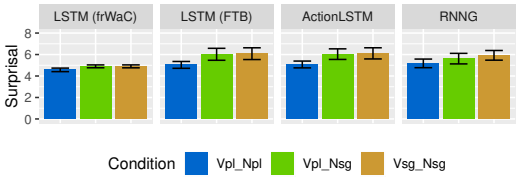


(b) French

Figure 7: Comparison of models' surprisals of *and*-coordination in Inverted Coordination experiment.

## 7 Discussion

The experiments presented here extend and refine a line of research investigating what linguistic knowledge is acquired by neural language models. Previous studies have demonstrated that sequential models trained on a simple regime of optimizing the next word can learn long-distance syntactic dependencies in impressive detail. Our results provide complimentary insights, demonstrating that a range of model architectures trained on a variety of datasets can learn fine-grained information about the interaction of CoordNPs and local syntactic context, but their behavior remains unhumanlike in many key ways. Furthermore, to our best knowledge, this work presents the first psycholinguistic analysis of neural language models trained on French, a high-resource language that has so far been under-investigated in this line of research.

In the **simple coordination** experiment, we demonstrated that models were able to capture some of the agreement behaviors of humans, although their performance deviated in crucial aspects. Whereas human behavior is best modeled as a 'percolation' process, the neural models appear to be using a linear combination of NP constituent number to drive CoordNP/verb number agreement, with the second noun weighted more heavily than the first. In these experiments, supervision afforded by the RNNG and ActionLSTM models did not translate into more robust or humanlike learning outcomes. The **complex coordination** experiments provided evidence that the neural models tested were not using a simple 'bag of features' strategy, but were sensitive to syntactic context. All models tested were able to interpret material that had similar surface form in ways that corresponded to two different tree-structural descriptions, based on local context. The **inverted coordination** experiment provided a contrasting example, in which models were unable to modulate expectations based on subtleties in the syntactic environment.

Across all our experiments, the French models performed consistently better on subject/verb number agreement than on subject/predicate gender agreement. Although there are likely more examples of subject/verb number agreement in the French training data, gender agreement is syntactically mandated and widespread in French. It remains an open question why all but one of the models tested were unable to leverage the numerous examples of gender agreement seen in various contexts during training to drive correct subject/predicate expectations.

## Acknowledgments

This project is supported by a grant of Labex EFL ANR-10-LABX-0083 (and Idex ANR-18-IDEX-0001) for AA and MIT–IBM AI Laboratory and the MIT–SenseTimeAlliance on Artificial Intelligence for RPL. We would like to thank the anonymous reviewers for their comments and Anne Abeillé for her advice and feedback.

## References


Anne Abeillé, Lionel Clément, and François Toussenel. 2003. Building a treebank for french. In *Treebanks*, pages 165–187. Springer.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The wacky wide web: a collection of very large linguistically processed web-crawled corpora. *Language resources and evaluation*, 43(3):209–226.

Do Kook Choe and Eugene Charniak. 2016. Parsing as language modeling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2331–2336, Austin, Texas. Association for Computational Linguistics.

Corbett. 1991. *Gender*. Cambridge University Press.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A Smith. 2016. Recurrent neural network grammars. In *Proceedings of NAACL-HLT*, pages 199–209.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

Jessica Ficler and Yoav Goldberg. 2016. Coordination annotation extension in the penn tree bank. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 834–842.

Henry Ramsey Fowler, Jane E Aaron, and Murray McArthur. 1992. *The little, brown handbook*. HarperCollins.

Richard Futrell, Ethan Wilcox, Takashi Morita, and Roger Levy. 2018. Rnns as psycholinguistic subjects: Syntactic state and grammatical dependency. *arXiv preprint arXiv:1809.01329*.

André Goosse and Maurice Grevisse. 2016. *Le bon usage*. De Boeck Superieur.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. *arXiv preprint arXiv:1803.11138*.

John Hale. 2001. A probabilistic earley parser as a psycholinguistic model. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, pages 1–8. Association for Computational Linguistics.

John Hale, Chris Dyer, Adhiguna Kuncoro, and Jonathan Brennan. 2018. Finding syntax in human encephalography with beam search. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2727–2736.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

Lap-Ching Keung and Adrian Staub. 2018. Variable agreement with coordinate subjects is not a form of agreement attraction. *Journal of Memory and Language*, 103:1–18.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A Smith. 2016. What do recurrent neural network grammars learn about syntax? *arXiv preprint arXiv:1611.05774*.

Roger Levy. 2008. Expectation-based syntactic comprehension. *Cognition*, 106(3):1126–1177.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of lstms to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Mitchell P Marcus and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2).

Rebecca Marvin and Tal Linzen. 2018. Targeted syntactic evaluation of language models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202.

Nathaniel J Smith and Roger Levy. 2013. The effect of word predictability on reading time is logarithmic. *Cognition*, 128(3):302–319.

Mitchell Stern, Daniel Fried, and Dan Klein. 2017. Effective inference for generative neural parsing. *arXiv preprint arXiv:1707.08976*.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*.

Stephen Wechsler and Larisa Zlatić. 2003. *The many faces of agreement*. Stanford: CSLI Publications.

Ethan Wilcox, Roger Levy, Takashi Morita, and Richard Futrell. 2018. What do rnn language models learn about filler–gap dependencies? In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 211–221.

J Gold Willer, Boban Arsenijević, Mia Batinić, Michael Becker, Nermina Čordalija, Marijana Kresić, Nedžad Leko, Franc Lanko Marušič, Tanja Milićev, Nataša Milićević, et al. 2018. When linearity prevails over hierarchy in syntax. *Proceedings of the National Academy of Sciences of the United States of America*, 115(3):495–500.

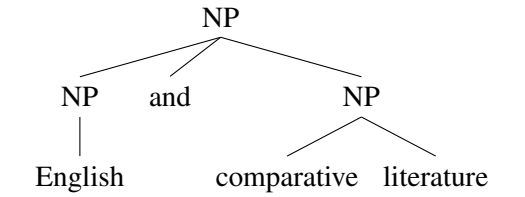


(a) PTB-control

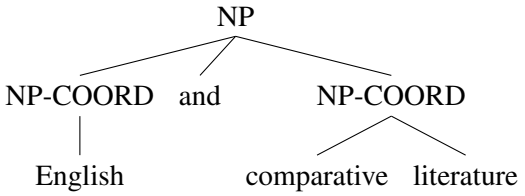


(b) PTB-coord

Figure 8: Comparison of annotation schemes of coordination structure.

## A The Effect of Annotation Schemes

This section further investigates the effects of CoordNP annotation schemes on the behaviors of structurally-supervised models. We test whether an explicit COORD phrasal tag improves model performance. We trained two additional RNNG models on 38,546 sentences from the Penn Treebank annotated with two different schemes: The first, **RNNG (PTB-control)** was trained with the original Penn Treebank annotation. The second, **RNNG (PTB-coord)**, was trained on the same sentences, but with an extended coordination annotation scheme, meant to employ the scheme employed in the FTB, adapted from Ficler and Goldberg (2016). We stripped empty categories from their scheme and only kept the NP-COORD label for constituents inside a coordination structure. Figure 8 illustrates the detailed annotation differences between two datasets. We tested both models on all the experiments presented in Sections 3-6 above.

Turning to the results of these six experiments: We see little difference between the two models in the *Non-coordination agreement* experiment. For the *Complex coordination control* and *Complex coordination critical* experiments, both models are largely the same as well. However, in the *Simple and-coordination* and *Simple or-coordination* experiments the values for all conditions are shifted upwards for the RNNG PTB-coord model, indicating higher over-all preference for the plural continuation. Furthermore, the range of values is reduced in the RNNG PTB-coord model, compared to the RNNG PTB-control model. These results indicate that adding an explicit COORD phrasal label does not drastically change model performance: Both models still appear to be using a linear combination of number features to drive plural vs. singular expectation. However, the explicit representation has made the interior of the coordination phrase more opaque to the model (each feature matters less) and has slightly shifted model preference towards plural continuations. In this sense, the PTB-coord model may have learned a generalization about CoordNPs, but this generalization remains unlike the ones learned by humans.

| | PTB | | | FTB | | |
|---|---|---|---|---|---|---|
| Condition | sg | pl | total | sg | pl | total |
| pl_and_pl | 0 | 67 | 67 | 1 | 116 | 116 |
| sg_and_pl | 0 | 72 | 72 | 0 | 50 | 50 |
| pl_and_sg | 0 | 11 | 11 | 0 | 30 | 30 |
| sg_and_sg | 7 | 213 | 220 | 5 | 288 | 293 |
| pl_or_pl | 0 | 2 | 2 | 0 | 14 | 14 |
| sg_or_pl | 0 | 0 | 0 | 0 | 0 | 0 |
| pl_or_sg | 0 | 1 | 1 | 0 | 1 | 1 |
| sg_or_sg | 5 | 1 | 6 | 5 | 8 | 13 |

Table 9: Frequency of number agreement patterns in PTB and FTB.

| Condition | m | f | total |
|---|---|---|---|
| m_and_m | 38 | 0 | 38 |
| m_and_f | 10 | 1 | 11 |
| f_and_m | 9 | 0 | 9 |
| f_and_f | 0 | 16 | 16 |
| m_or_m | 1 | 0 | 1 |
| m_or_f | 0 | 0 | 0 |
| f_or_m | 1 | 0 | 1 |
| f_or_f | 0 | 1 | 1 |

Table 10: Frequency of gender agreement patterns in FTB.

## B PTB/FTB Agreement Patterns

We present statistics of subject/predicate agreement patterns in the Penn Treebank (PTB) and French Treebank (FTB) in Table 9 and 10.

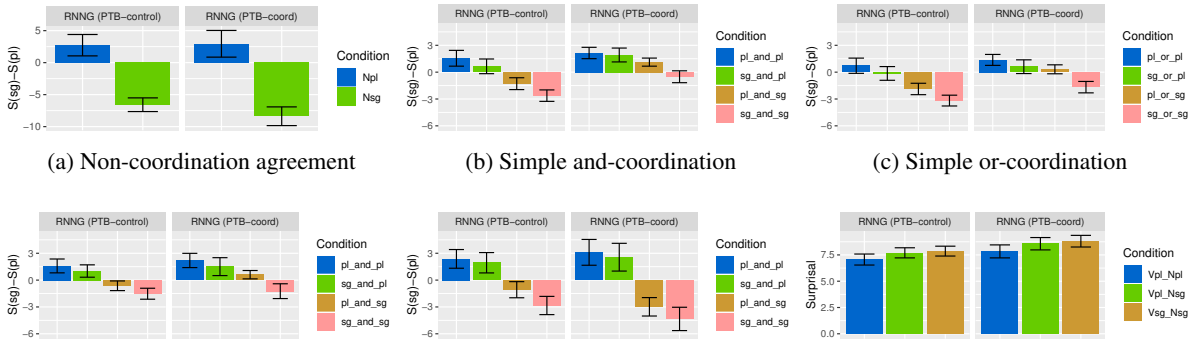


(a) Non-coordination agreement (b) Simple and-coordination (c) Simple or-coordination

(d) Complex coordination control (e) Complex coordination critical (f) Inverted coordination

Figure 9: Comparison between RNNGs trained on PTB data with original annotation vs. fine-grained annotation of coordination structure.

# Towards Zero-shot Language Modeling

**Edoardo M. Ponti**[1], **Ivan Vulić**[1], **Ryan Cotterell**[2], **Roi Reichart**[3], **Anna Korhonen**[1]
[1]Language Technology Lab, TAL, University of Cambridge
[2]Computer Laboratory, University of Cambridge
[2]Faculty of Industrial Engineering and Management, Technion, IIT
[1,2]{ep490,iv250,rdc42,alk23}@cam.ac.uk
[3]roiri@ie.technion.ac.il

## Abstract

Can we construct a neural language model which is inductively biased towards learning *human* language? Motivated by this question, we aim at constructing an informative prior for held-out languages on the task of character-level, open-vocabulary language modeling. We obtain this prior as the posterior over network weights conditioned on the data from a sample of training languages. This prior is approximated through Laplace's method. Based on a large and diverse sample of languages, the use of our prior outperforms baseline models with an uninformative prior in both zero-shot and few-shot settings, showing that the prior is imbued with universal linguistic knowledge. Moreover, we harness broad language-specific information available for most languages of the world, i.e., features from typological databases, as distant supervision for held-out languages. We explore several language modeling conditioning techniques, which appear beneficial in the few-shot setting, but ineffective in the zero-shot setting. Since the paucity of even plain digital text affects the majority of the world's languages, we hope that these insights will broaden the scope of applications for language technology.

## 1 Introduction

With the success of recurrent neural networks and other black-box models on core NLP tasks, such as language modelling, researchers have turned their attention to the study of the inductive bias such neural models exhibit (Linzen et al., 2016; Marvin and Linzen, 2018; Ravfogel et al., 2018). A number of natural questions have been asked. For example, do recurrent neural language models learn syntax (Marvin and Linzen, 2018)? Do they map onto grammaticality judgements (Warstadt et al., 2018)? However, as Ravfogel et al. (2019) note, "[m]ost of the work so far has focused on English." Moreover, these studies have almost always focused on training scenarios where a large number of in-language sentences are available.

In this work, we aim to find a prior distribution over network parameters that generalize well for human language. The recent vein of research on the inductive biases of neural nets implicitly assumes a uniform (unnormalizable) prior over the space of neural network parameters (Ravfogel et al., 2019, *inter alia*). In contrast, we take a Bayesian-updating approach and construct a suitable prior by approximating the posterior distribution over the network parameters conditioned on the data from a sample of *seen* training languages using the Laplace method (Azevedo-Filho and Shachter, 1994). The posterior distribution serves as a prior for maximum-a-posteriori (MAP) estimation of network parameters for the held-out unseen languages.

The search for a universal prior for linguistic knowledge is motivated by the notion of Universal Grammar (UG), originally proposed by Chomsky (1959). The presence of innate biological properties of the brain that constrain possible human languages was posited to explain why human children learn human languages so quickly despite the poverty of the stimulus (Chomsky, 1978; Legate and Yang, 2002). In turn, UG has been connected with Greenberg (1963)'s typological universals by Graffi (1980) and Gilligan (1989): this way, the patterns observed in cross-lingual variation could be explained by the language-specific configuration of an innate set of parameters.

Our study explores the task of character-level, open-vocabulary language modeling to allow for intercomparability between the performance of different models across different languages (Gerz et al., 2018a,b; Cotterell et al., 2018; Mielke et al., 2019). We run experiments under several regimes of data scarcity for the held-out languages (zero-shot, few-shot, and joint multilingual learning) over a sample



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2900–2910, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

of 77 typologically diverse languages.

Realistically, a model should not be completely in the dark about held-out languages, as coarse-grained features about general linguistic properties are documented for most world's languages and available in typological databases such as URIEL (Littell et al., 2017). Hence, we also explore a regime where we condition the universal prior over the weights on typological side information. In particular, we consider concatenating typological features to hidden states (Östling and Tiedemann, 2017) and generating the network parameters based on the typological features (Platanios et al., 2018).

Empirically, given the results of our study, we offer two findings. The first is that neural recurrent models with a universal prior significantly outperform baselines with uninformative priors both in zero-shot and few-shot training settings. Secondly, conditioning on typological features further reduces bytes per character in the few-shot setting, but we report negative results for the zero-shot setting, possibly due to some inherent limitations of typological databases (Ponti et al., 2018a).

The study of low-resource language modelling also has a practical impact. According to Simons (2017), 45.71% of the world's languages do not have written texts available. The situation is even more dire for their *digital* footprint. As of March 2015, just 40 out of the 188 languages documented on the Internet accounted for 99.99% of the web pages.[1] And as of April 2019, Wikipedia is translated only in 304 out of the 7097 existing languages. What is more, Kornai (2013) prognosticates that the digital divide will act as a catalyst for the extinction of many of the world's languages. The transfer of language technology may help reverse this course and give space to unrepresented communities.

## 2 LSTM Language Models

In this work, we address the task of *character-level* language modeling. Whereas word lexicalization is mostly arbitrary across languages, phonemes allow for transfer of universal constraints on phonotactics[2] and language-specific sequences that may be shared across languages, such as borrowings and genetically related words (Brown et al., 2008). Since languages are mostly recorded in text rather than phonemic symbols (IPA), however, we focus on characters as a substitute of phonemes.

Let $\Sigma^\ell$ be the set of characters for language $\ell$. For a collection of languages $\mathcal{D}$, let $\Sigma = \cup^{\ell \in \mathcal{D}} \Sigma^\ell$ be the union of characters in all languages. A universal, character-level language model is then a probability distribution over $\Sigma^*$.[3] Let $\mathbf{x} \in \Sigma^*$ be a sequence of characters. We write:

$$p(\mathbf{x} \mid \mathbf{w}) = \prod_{t=1}^{n} p(x_t \mid \mathbf{x}_{<t}, \mathbf{w}) \quad (1)$$

where $x_0$ is a distinguished beginning-of-sentence symbol and $t$ is a time step.

We implement character-level language models with Long Short-Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1997). These encode the entire history $\mathbf{x}_{<t}$ as a fixed-length vector by manipulating a memory cell $\mathbf{c}_t$ through a set of gates. Then we define

$$p(x_t \mid \mathbf{h}_t, \mathbf{w}) = \mathit{softmax}(\mathbf{W}\,\mathbf{h}_t + \mathbf{b}). \quad (2)$$

Since we tie $\mathbf{W}$ to the character embeddings $\mathbf{X}$, $\mathbf{W} = \mathbf{X}^\top$. The parameters $\mathbf{w}$ are typically optimized to maximize the likelihood of token input sequences. LSTMs have an advantage over other recurrent architectures as memory gating mitigates the problem of vanishing gradients and captures long-distance dependencies (Pascanu et al., 2013).

## 3 Neural Language Modelling with a Universal Prior

The fundamental hypothesis of this work is that there exists a prior $p(\mathbf{w})$ over the weights of a neural language model that places high probability on networks that describe human-like languages. The inductive bias in such a prior will facilitate the training of language models for *unseen* languages. Our goal is to estimate the prior as the posterior distribution over the weights of a language model of *seen* languages. Taking a Bayesian approach, with $\mathcal{D}_t$ as the set of training languages, the posterior over weights is given by the Bayes' rule:

$$\underbrace{p(\mathbf{w} \mid \mathcal{D}_t)}_{\textit{posterior}} \propto \underbrace{\prod^{\ell \in \mathcal{D}_t} p(\mathbf{x}^\ell \mid \mathbf{w})}_{\textit{likelihood}} \times \underbrace{p(\mathbf{w})}_{\textit{prior}} \quad (3)$$

[1] https://w3techs.com/technologies/overview/content_language/all

[2] E.g. with few exceptions (Evans and Levinson, 2009, sec. 2.2.2), the basic syllabic structure is vowel–consonant.

[3] Note that $\Sigma$ is also augmented with punctuation and white space, and distinguished beginning-of-sequence and end-of-sequence symbols, respectively.

Computation of the posterior $p(\mathbf{w} \mid \mathcal{D})$ is woefully intractable: recall that each $p(\mathbf{x} \mid \mathbf{w})$ is in our setting an LSTM language model, like the one defined in eq. (2). We take the prior to be a Gaussian, i.e.

$$p(\mathbf{w}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}||\mathbf{w}||_2^2\right) \quad (4)$$

with zero mean and covariance matrix $\sigma^2 I$. Here, we opt for a simple approximation to the posterior, using the classic Laplace method (Azevedo-Filho and Shachter, 1994). This method has recently been applied to other transfer learning tasks in the neural network literature (Kirkpatrick et al., 2017; Kochurov et al., 2018; Ritter et al., 2018).

In §3.1, we first introduce the Laplace method, which approximates the posterior with a Gaussian.[4] The mean and covariance matrix are chosen through an optimization-based procedure, so it is amenable to computation with backpropagation, as detailed in §3.2. Finally, we describe how to use this distribution as a prior to perform maximum-a-posteriori inference over new data in §3.3.

### 3.1 Laplace Method

First, we (locally) maximize the log-likelihood of the data in all the training languages plus the prior:[5]

$$\mathcal{L}(\mathbf{w}) = \sum^{\ell \in \mathcal{D}_t} \log p(\mathbf{x}^\ell \mid \mathbf{w}) + \log p(\mathbf{w}) \quad (5)$$

We note that this is equivalent to the log-posterior up to an additive constant, i.e.,

$$\log p(\mathbf{w} \mid \mathcal{D}_t) = \mathcal{L}(\mathbf{w}) - \log p(\mathbf{x}^\ell) \quad (6)$$

where the constant $\log p(\mathbf{x}^\ell)$ is the log-normalizer for the posterior by the Bayes' rule. Let $\mathbf{w}^\star$ be a local maximizer of $\mathcal{L}$. We now approximate the log-posterior with a second-order Taylor expansion around the local maximizer $\mathbf{w}^\star$:

$$\log p(\mathbf{w} \mid \mathcal{D}_t) \quad (7)$$
$$\approx \mathcal{L}(\mathbf{w}^\star) + \frac{1}{2}(\mathbf{w} - \mathbf{w}^\star)^\top \mathbf{H} (\mathbf{w} - \mathbf{w}^\star)$$

where we have omitted the first-order term, since the gradient $\nabla_{\mathcal{L}}$ at the local maximizer $\mathbf{w}^\star$ is zero. Note that we have defined $\mathbf{H}$ as the Hessian. This quadratic approximation to the log-posterior implies that the approximate posterior is Gaussian. This can be seen by exponentiating both sides in eq. (7):

$$p(\mathbf{w} \mid \mathcal{D}_t) \propto \exp\left(\frac{1}{2}(\mathbf{w} - \mathbf{w}^\star)^\top \mathbf{H} (\mathbf{w} - \mathbf{w}^\star)\right) \quad (8)$$

where $\mathcal{L}(\mathbf{w}^\star)$ and $\log p(\mathbf{x}^\ell)$ are absorbed into the Gaussian's normalization constant, which may be computed analytically. Because $\mathbf{w}^\star$ is a local maximizer, $\mathbf{H}$ is a negative-definite matrix. In principle, computing the Hessian is possible through running backpropagation twice: This yields a matrix with $d^2$ entries. However, in practice, this is not possible. First, running backpropagation twice is tedious. Second, we can not easily store a matrix with $d^2$ entries since $d$ is the number of parameter in the neural language model.

### 3.2 Approximating the Hessian

To cut the computation down to one pass, we exploit a property from theoretical statistics: Namely, that the Hessian of the log-likelihood bears a close resemblance to a quantity known as the Fisher information matrix. This connection allows us to develop a more efficient algorithm that approximates the Hessian with one pass of backpropagation.

We derive this approximation to the Hessian of $\mathcal{L}(\mathbf{w})$ here. First, we note that due to the linearity of $\nabla^2$, we have

$$\mathbf{H} = \nabla^2 \mathcal{L}(\mathbf{w}) \quad (9)$$
$$= \nabla^2 \left(\sum^{\ell \in \mathcal{D}} \log p(\mathbf{x}^\ell \mid \mathbf{w}) + \log p(\mathbf{w})\right) \quad (10)$$
$$= \underbrace{\sum^{\ell \in \mathcal{D}} \nabla^2 \log p(\mathbf{x}^\ell \mid \mathbf{w})}_{\textit{likelihood}} + \underbrace{\nabla^2 \log p(\mathbf{w})}_{\textit{prior}} \quad (11)$$

We discuss each term individually. First, to approximate the likelihood term, we draw on the relation between the Hessian and the Fisher information matrix. A basic fact from information theory (Cover and Thomas, 2006) gives us that the Fisher information matrix may be written in two equivalent ways:

$$-\mathbb{E}_{\mathbf{x}^\ell}\left[\nabla^2 \log p(\mathbf{x}^\ell \mid \mathbf{w})\right] \quad (12)$$
$$= \underbrace{\mathbb{E}_{\mathbf{x}^\ell}\left[\nabla \log p(\mathbf{x}^\ell \mid \mathbf{w}) \nabla \log p(\mathbf{x}^\ell \mid \mathbf{w})^\top\right]}_{\textit{expected Fisher information matrix}}$$

[4] Note that, in general, the true posterior is multi-modal. The Laplace method instead approximates it with a unimodal distribution.

[5] In this case, we provide an uninformative prior $\mathcal{N}(0, 1)$.

Note that the integral over all possible languages $\mathbf{x}^\ell$ is a discrete summation, so we may exchange summands and derivatives such as is required for the proof. This equality suggests a natural approximation of the expected Fisher information matrix, the *observed* Fisher information matrix $\mathbf{F}$

$$-\frac{1}{|\mathcal{D}|}\sum^{\ell\in\mathcal{D}}\nabla^2\log p(\mathbf{x}^\ell \mid \mathbf{w}) \tag{13}$$
$$\approx \underbrace{\frac{1}{|\mathcal{D}|}\sum^{\ell\in\mathcal{D}}\nabla\log p(\mathbf{x}^\ell \mid \mathbf{w})\,\nabla\log p(\mathbf{x}^\ell \mid \mathbf{w})^\top}_{\textit{observed Fisher information matrix}}$$

which is tight in the limit as $|\mathcal{D}| \to \infty$ due to the law of large numbers. Indeed, when we have a large number of training exemplars, the average of the outer products of the gradients will be a good approximation to the Hessian. However, even $\mathbf{F}$ still has $d^2$ entries, which is far too many to be practical. Thus, we further use a diagonal approximation. We denote the diagonal of the observed Fisher information matrix as the vector $\mathbf{f} \in \mathbb{R}^d$ where we define

$$f_i = \sum^{\ell\in\mathcal{D}}\Big(\nabla\log p(\mathbf{x}^\ell \mid \mathbf{w})\Big)_i^2 \tag{14}$$

which yields the $\mathit{diag}(\mathbf{F})$.

Computation of the Hessian of the prior term is more straight-forward and does not require approximation. Indeed, in the general case it is the inverse covariance matrix, which means in our case we have

$$\nabla^2\log p(\mathbf{w}) = \frac{1}{\sigma^2}I \tag{15}$$

This yields the final diagonal approximation to the Hessian

$$\tilde{\mathbf{H}} = -\mathit{diag}(\mathbf{f}) + \frac{1}{\sigma^2}I \tag{16}$$

In practice, computing the Laplace approximation may be achieved through backpropagation with a few tricks:[6] A gradient-based optimization method is used to locally optimize $\mathcal{L}$.[7] Then, since a neural network typically has millions of parameters and $\mathbf{H}$ would be intractable to even store, we approximate it with the Hessian's diagonal matrix.

[6]While it is intractable to compute the normalizing constant, and hence the posterior, the gradient of this constant with respect to the neural network's weights is zero and, thus, irrelevant for optimization.

[7]In practice, non-convex optimization is only guaranteed to reach a critical (saddle) point. However, the derivation of Laplace's method assumes we do reach a local maximizer.

### 3.3 MAP Inference

Finally, we use the posterior $p(\mathbf{w} \mid \mathcal{D}_t)$ as the prior over model parameters for training a language model on new held-out languages. We incorporate this prior through MAP estimation, by augmenting the optimization of the likelihood of the new data with a regularization term. This is only an approximation to full Bayes estimators, because it does not characterize the entire distribution of the posterior, but rather just the mode (Gelman et al., 2013).

In the zero-shot setting, this boils down to using the mean of the prior as network parameters. In the few-shot setting instead, we assume that some data for the target language $\mathcal{D}_e$ is available, so we treat the posterior as a prior over the weights and update the weights accordingly. In particular, we maximize the log-likelihood of the target language data and the regularizer derived through the Laplace Approximation scaled by a factor $\lambda$:

$$\mathcal{L}(\mathbf{w}) = \sum^{\ell\in\mathcal{D}_e}\log p(\mathbf{x}^\ell \mid \mathbf{w}) + \frac{\lambda}{2}(\mathbf{w}-\mathbf{w}^\star)^\top\tilde{\mathbf{H}}\,(\mathbf{w}-\mathbf{w}^\star) \tag{17}$$

As a baseline for the UNIV prior, we perform Maximum A Posteriori inference with an uninformative prior $\mathcal{N}(0,1)$. We label this model NINF. In the zero-shot setting, this means that the parameters are sampled from the uninformative prior. In the few-shot setting, we maximize the likelihood of the data for the held-out language while minimizing a regularizer that reduces to $\frac{\lambda}{2}||1\odot(\mathbf{w}-0)||_2^2 = \frac{\lambda}{2}\mathbf{w}^2$. Note that, owing to the uninformative prior, the uninformed NINF models do not have access to the posterior of the weights given the data from the training languages.

Moreover, we consider a common approach for neural transfer learning (Ruder, 2017), which lies outside the Bayesian framework, as an additional baseline. Namely, after optimizing the weights on the training data, those are simply fine-tuned on the held-out data, until finding a new local maximizer. We label this method FITU.

## 4 Language Modeling Conditioned on Typological Features

Realistically, the prior over network weights should also be augmented with side information about the general properties of the held-out language to be

learnt, if such information is available. In fact, linguists have documented such information even for languages without plain digital texts available, and stored it in publicly accessible databases (Croft, 2002; Dryer and Haspelmath, 2013). This information usually takes the form of features that express either: i) the formal strategies each language employs to express a specific semantic / functional construction (Croft et al., 2017). For instance, English expresses the construction of nominal predication with a copula strategy; or ii) the presence or absence of specific phenomena. For instance, English possesses a grammatical category for tense.

The usage of such features to inform neural NLP models is still scarce, partly because the evidence in their favour is mixed (Ponti et al., 2018b,a). In this work, we propose a way to distantly supervise the model with this *side information* effectively. We extend our non-conditional language model with a universal prior (BARE) to a series of architectures *conditioned* on language-specific properties that have been proposed in previous work (Östling and Tiedemann, 2017; Platanios et al., 2018). A fundamental difference, however, is that these learn such properties in an end-to-end fashion from the data in a joint multilingual learning setting. Obviously, this is not feasible for the zero-shot setting and unreliable for the few-shot setting. Rather, we represent languages with their typological feature vector, which we assume readily available both for training and for held-out languages.

Let $\mathbf{t}^\ell \in [0,1]^d$ be a vector of typological features for language $\ell \in \mathcal{D}_t \cup \mathcal{D}_e$. We reinterpret the conditional language models within the Bayesian framework as estimating the posterior probability

$$\prod^{\ell \in \mathcal{D}} p(\mathbf{x}^\ell \mid \mathbf{w}, \mathbf{t}^\ell) \times p(\mathbf{w} \mid \mathbf{t}^\ell). \quad (18)$$

We now outline several candidate methods to estimate $p(\mathbf{w} \mid \mathbf{t}^\ell)$. We first encode the features through a non-linear transformation $f(\mathbf{t}) = \text{ReLU}(\mathbf{W}\,\mathbf{t} + b)$. A first variant, labeled OEST, is inspired by Östling and Tiedemann (2017). Assuming the standard LSTM architecture where $\mathbf{o}_t$ is the output gate and $\mathbf{c}_t$ is the memory cell, we modify the equation for the hidden state $\mathbf{h}_t$ as follows:

$$\mathbf{h}_t = o_t \odot \tanh(\mathbf{c}_t) \oplus f(\mathbf{t}^\ell) \quad (19)$$

In other words, we concatenate the typological features to all the hidden states.

Moreover, we experiment with a second variant where the parameters of the LSTM are generated by a meta-network (i.e., a simple linear layer with weight $W^{d \times |\mathbf{w}|}$) that transforms $\mathbf{t}^\ell$ into $\mathbf{w}$. This approach, labeled PLAT, is inspired by Platanios et al. (2018), with the additional difference that they generate parameters for an encoder-decoder.

On the other hand, we do not consider the conditional model proposed by Sutskever et al. (2014), where $f(\mathbf{t})$ would be used to initialize the values for $\mathbf{h}_0$ and $\mathbf{c}_0$. During evaluation, $\mathbf{h}$ and $\mathbf{c}$ are never reset on sentence boundaries, so this model would find itself at disadvantage because it would require to erase the sequential history cyclically.

## 5 Experimental Setup

**Data** Our text data source is the Bible corpus[8] (Christodouloupoulos and Steedman, 2015).[9] We exclude languages that are not written in the Latin script and duplicate languages, resulting in a subsample of 77 languages.[10] Since not all translations cover the entire Bible, they vary in size. The text from each language is split into training, development, and evaluation sets with a ratio of 80/10/10%. Moreover, for the MAP inference in the few-shot setting, we randomly sample 100 sentences from each training set.

We obtain the typological feature vectors from URIEL (Littell et al., 2017).[11] We include the features related to 3 levels of linguistic structure, for a total of 245 features: i) syntax, e.g. whether the subject tends to precede the object. These originate from the World Atlas of Language Structures (Dryer and Haspelmath, 2013) and the Syntactic Structures of the World's Languages (Collins and Kayne, 2009); ii) phonology, e.g. whether a language has distinctive tones; iii) phonological inventories, e.g. whether a language possesses the retroflex approximant /ɻ/. Both ii) and iii) were originally collected in PHOIBLE (Moran et al., 2014). Missing values were inferred as a weighted average of the 10 nearest neighbour languages in terms of family, geography, and typology.

[8] http://christos-c.com/bible/

[9] This corpus is arguably representative of the variety of the world's languages: it covers 28 genealogical families, several geographic areas (16 languages from Africa, 23 from Americas, 26 from Asia, 33 from Europe, 1 from Oceania), and endangered or poorly documented languages (39 with less than a million speakers).

[10] These are identified with their 3-letter ISO 639-3 codes throughout the paper. Consult the Appendix in the supplemental material for the full list of language names mapped to ISO 639-3 codes.

[11] http://www.cs.cmu.edu/∼dmortens/uriel.html

| | NINF | UNIV | | | NINF | UNIV | | | NINF | UNIV | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | BARE | BARE | OEST | | BARE | BARE | OEST | | BARE | BARE | OEST |
| *acu* | 8.491 | **3.244** | 3.472 | *fra* | 8.587 | **4.066** | 4.467 | *por* | 8.491 | **3.751** | 4.219 |
| *afr* | 8.607 | **3.229** | 3.995 | *gbi* | 8.610 | **3.823** | 3.912 | *pot* | 8.600 | **5.336** | 5.359 |
| *agr* | 8.603 | **3.779** | 3.946 | *gla* | 8.490 | 4.179 | 3.956 | *ppk* | 8.596 | **4.506** | 4.599 |
| *ake* | 8.602 | **5.753** | 6.281 | *glv* | 8.606 | **4.349** | 4.612 | *quc* | 8.605 | **4.063** | 4.118 |
| *alb* | 8.490 | **4.571** | 5.017 | *hat* | 8.594 | **4.186** | 4.620 | *quw* | 8.488 | **3.560** | 4.027 |
| *amu* | 8.610 | **4.912** | 5.959 | *hrv* | 8.606 | 4.050 | **3.441** | *rom* | 8.603 | **3.669** | 4.056 |
| *bsn* | 8.591 | **5.046** | 5.695 | *hun* | 8.493 | **4.836** | 5.030 | *ron* | 8.588 | **5.011** | 5.690 |
| *cak* | 8.603 | **4.068** | 4.326 | *ind* | 8.604 | **3.796** | 4.311 | *shi* | 8.601 | **5.496** | 5.946 |
| *ceb* | 8.488 | **3.668** | 3.850 | *isl* | 8.596 | **5.039** | 5.629 | *slk* | 8.491 | **4.304** | 4.512 |
| *ces* | 8.600 | **4.369** | 4.461 | *ita* | 8.605 | 4.023 | **3.752** | *slv* | 8.604 | **3.661** | 4.106 |
| *cha* | 8.594 | 4.366 | **4.353** | *jak* | 8.488 | **4.051** | 4.793 | *sna* | 8.596 | **4.146** | 4.283 |
| *chq* | 8.598 | **6.940** | 7.623 | *jiv* | 8.601 | **3.866** | 4.039 | *som* | 8.614 | **4.159** | 4.470 |
| *cjp* | 8.494 | **4.600** | 4.985 | *kab* | 8.596 | **4.659** | 5.400 | *spa* | 8.489 | **3.645** | 4.020 |
| *cni* | 8.604 | **3.740** | 4.651 | *kbh* | 8.607 | **4.663** | 4.950 | *srp* | 8.604 | **3.414** | 3.437 |
| *dan* | 8.593 | **3.471** | 4.599 | *kek* | 8.491 | **4.666** | 4.944 | *ssw* | 8.593 | 4.064 | **3.780** |
| *deu* | 8.599 | **4.102** | 4.214 | *lat* | 8.601 | **3.703** | 4.093 | *swe* | 8.605 | 4.210 | **3.892** |
| *dik* | 8.490 | **4.447** | 4.533 | *lav* | 8.588 | **5.415** | 6.130 | *tgl* | 8.487 | **3.639** | 3.878 |
| *dje* | 8.603 | **3.725** | 3.996 | *lit* | 8.602 | **4.794** | 4.853 | *tmh* | 8.602 | 4.830 | **4.711** |
| *djk* | 8.592 | **3.663** | 3.874 | *mam* | 8.488 | **4.292** | 5.076 | *tur* | 8.592 | **5.574** | 5.935 |
| *dop* | 8.609 | **5.950** | 7.351 | *mri* | 8.606 | **3.440** | 4.074 | *usp* | 8.604 | **4.127** | 4.337 |
| *eng* | 8.488 | **3.816** | 4.028 | *nhg* | 8.588 | **4.323** | 4.450 | *vie* | 8.490 | **7.137** | 7.484 |
| *epo* | 8.605 | **3.818** | 4.116 | *nld* | 8.601 | **3.851** | 4.326 | *wal* | 8.605 | **4.027** | 4.585 |
| *est* | 8.606 | **6.807** | 8.261 | *nor* | 8.492 | **3.174** | 3.902 | *wol* | 8.607 | **4.290** | 4.420 |
| *eus* | 8.605 | **4.118** | 4.321 | *pck* | 8.603 | **4.053** | 4.233 | *xho* | 8.602 | **4.171** | 4.276 |
| *ewe* | 8.490 | **5.049** | 5.497 | *plt* | 8.603 | **4.364** | 4.648 | *zul* | 8.488 | **3.218** | 4.109 |
| *fin* | 8.604 | **4.308** | 4.338 | *pol* | 8.601 | **5.158** | 5.556 | ALL | 8.572 | **4.343** | 4.691 |

Table 1: BPC scores (lower is better) for the ZERO-SHOT learning setting, with the uninformed prior (NINF) and the universal prior (UNIV): see §2 for the descriptions of the priors. Note that for the former there is no difference between a BARE model and a conditional model (OEST). Colors define the split in which each language (rows) has been held out.

| | BARE | OEST | | BARE | OEST | | BARE | OEST | | BARE | OEST |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *acu* | 1.413 | **1.308** | *eng* | 1.355 | **1.350** | *kek* | **1.131** | 1.133 | *slk* | 1.844 | **1.754** |
| *afr* | 1.471 | **1.457** | *epo* | 1.471 | **1.450** | *lat* | 1.792 | **1.758** | *slv* | 1.848 | **1.793** |
| *agr* | 1.701 | **1.581** | *est* | 0.333 | **0.150** | *lav* | 2.146 | **1.931** | *sna* | 1.489 | **1.457** |
| *ake* | 1.453 | **1.377** | *eus* | 1.763 | **1.635** | *lit* | 1.895 | **1.833** | *som* | 1.477 | **1.468** |
| *alb* | 1.590 | **1.552** | *ewe* | 2.084 | **1.944** | *mam* | 1.654 | **1.548** | *spa* | 1.559 | **1.525** |
| *amu* | 1.402 | **1.340** | *fin* | 1.716 | **1.680** | *mri* | 1.342 | **1.330** | *srp* | 1.832 | **1.756** |
| *bsn* | 1.232 | **1.172** | *fra* | 1.465 | **1.432** | *nhg* | 1.302 | **1.238** | *ssw* | 1.890 | **1.697** |
| *cak* | 1.281 | **1.221** | *gbi* | 1.398 | **1.331** | *nld* | 1.621 | **1.601** | *swe* | 1.619 | **1.595** |
| *ceb* | 1.193 | **1.185** | *gla* | 3.403 | **1.839** | *nor* | 1.623 | **1.590** | *tgl* | 1.221 | **1.210** |
| *ces* | 1.872 | **1.795** | *glv* | 1.932 | **1.644** | *pck* | 1.731 | **1.711** | *tmh* | 2.786 | **2.301** |
| *cha* | 1.934 | **1.790** | *hat* | 1.480 | **1.454** | *plt* | 1.296 | **1.286** | *tur* | 1.801 | **1.773** |
| *chq* | 1.265 | **1.220** | *hrv* | 2.059 | **1.974** | *pol* | 1.743 | **1.698** | *usp* | 1.290 | **1.214** |
| *cjp* | 1.706 | **1.565** | *hun* | 1.887 | **1.847** | *por* | 1.586 | **1.552** | *vie* | 1.648 | **1.637** |
| *cni* | 1.348 | **1.290** | *ind* | 1.356 | **1.336** | *pot* | 2.484 | **2.144** | *wal* | 1.561 | **1.457** |
| *dan* | 1.727 | **1.693** | *isl* | 1.845 | **1.808** | *ppk* | 1.538 | **1.439** | *wol* | 2.053 | **1.890** |
| *deu* | 1.532 | **1.512** | *ita* | 1.615 | **1.583** | *quc* | 1.393 | **1.291** | *xho* | 1.680 | **1.634** |
| *dik* | 1.979 | **1.835** | *jak* | 1.415 | **1.322** | *quw* | 1.498 | **1.418** | *zul* | 1.880 | **1.620** |
| *dje* | 1.570 | **1.550** | *jiv* | 1.705 | **1.572** | *rom* | 1.706 | **1.587** | ALL | 1.652 | **1.550** |
| *djk* | 1.515 | **1.435** | *kab* | 1.955 | **1.791** | *ron* | 1.572 | **1.537** | | | |
| *dop* | 1.810 | **1.676** | *kbh* | 1.436 | **1.371** | *shi* | 2.057 | **1.903** | | | |

Table 2: BPC results (lower is better) for the JOINT learning setting, with the uninformed NINF prior. These results constitute the expected ceiling performance for language transfer models.

| | NINF BARE | FITU OEST | UNIV BARE | UNIV OEST | | NINF BARE | FITU OEST | UNIV BARE | UNIV OEST |
|---|---|---|---|---|---|---|---|---|---|
| *acu* | 4.203 | **2.117** | 2.551 | 2.136 | *kbh* | 4.644 | 2.362 | 2.434 | **2.288** |
| *afr* | 4.423 | 3.620 | 3.042 | **2.773** | *kek* | 4.613 | 2.809 | 3.015 | **2.714** |
| *agr* | 4.268 | 3.282 | 3.403 | **2.457** | *lat* | 4.239 | 4.342 | 3.416 | **3.202** |
| *ake* | 4.318 | 2.168 | 2.238 | **2.180** | *lav* | 4.765 | **2.867** | 3.842 | 2.917 |
| *alb* | 4.544 | 3.186 | 3.302 | **3.084** | *lit* | 4.769 | 3.752 | **3.592** | 3.668 |
| *amu* | 4.486 | 2.820 | 3.948 | **2.080** | *mam* | 4.525 | **2.274** | 2.873 | 2.363 |
| *bsn* | 4.546 | 1.861 | 2.678 | **1.850** | *mri* | 3.795 | 3.482 | 3.010 | **2.459** |
| *cak* | 4.426 | 1.994 | 2.053 | **1.956** | *nhg* | 4.373 | 2.004 | 2.480 | **1.965** |
| *ceb* | 4.084 | 2.562 | 2.595 | **2.470** | *nld* | 4.469 | 3.008 | 2.908 | **2.903** |
| *ces* | 4.984 | 4.651 | 4.190 | **3.680** | *nor* | 4.453 | 3.152 | **2.954** | 3.054 |
| *cha* | 4.329 | 2.546 | 2.899 | **2.525** | *pck* | 4.246 | 4.011 | 3.532 | **3.030** |
| *chq* | 4.941 | **1.948** | 2.078 | 1.963 | *plt* | 4.201 | 2.532 | 2.742 | **2.490** |
| *cjp* | 4.424 | **2.389** | 2.880 | 2.393 | *pol* | 4.853 | 3.852 | **3.620** | 3.788 |
| *cni* | 4.185 | 2.797 | 3.018 | **1.982** | *por* | 4.446 | 3.231 | 3.198 | **3.098** |
| *dan* | 4.719 | 3.211 | **3.127** | 3.180 | *pot* | 4.299 | 3.773 | 3.944 | **2.763** |
| *deu* | 4.589 | 3.103 | 3.007 | **2.953** | *ppk* | 4.439 | **2.220** | 2.736 | 2.236 |
| *dik* | 4.380 | **2.640** | 3.020 | 2.667 | *quc* | 4.538 | 2.154 | 2.242 | **2.108** |
| *dje* | 4.382 | 3.815 | 3.398 | **2.898** | *quw* | 4.223 | 2.196 | 2.547 | **2.158** |
| *djk* | 4.130 | **2.064** | 2.446 | 2.085 | *rom* | 4.378 | 3.121 | 3.257 | **2.455** |
| *dop* | 4.508 | 2.506 | 2.562 | **2.448** | *ron* | 4.579 | 3.273 | 3.734 | **3.216** |
| *eng* | 4.436 | 2.808 | 2.913 | **2.719** | *shi* | 4.509 | **2.963** | 3.092 | 2.970 |
| *epo* | 4.469 | 3.609 | 3.511 | **2.825** | *slk* | 4.873 | 3.722 | 3.812 | **3.631** |
| *est* | 3.618 | **1.952** | 2.487 | 1.962 | *slv* | 4.633 | 4.630 | 3.527 | **3.501** |
| *eus* | 4.354 | 2.628 | 2.705 | **2.567** | *sna* | 4.455 | 2.910 | 3.114 | **2.870** |
| *ewe* | 4.590 | 2.806 | 3.336 | **2.786** | *som* | 4.257 | 3.048 | **2.908** | 2.934 |
| *fin* | 4.385 | 4.339 | 3.830 | **3.312** | *spa* | 4.507 | 3.223 | 3.149 | **3.090** |
| *fra* | 4.551 | 3.086 | 3.276 | **2.981** | *srp* | 4.561 | 4.467 | **3.367** | 3.380 |
| *gbi* | 4.250 | 2.138 | 2.170 | **2.054** | *ssw* | 4.370 | 2.611 | 2.924 | **2.570** |
| *gla* | 4.159 | **2.377** | 2.835 | 2.395 | *swe* | 4.657 | 3.266 | 3.184 | **3.177** |
| *glv* | 4.346 | 3.523 | 3.702 | **2.644** | *tgl* | 4.060 | 2.546 | 2.592 | **2.436** |
| *hat* | 4.468 | 2.929 | 3.048 | **2.849** | *tmh* | 4.618 | 4.087 | 4.218 | **3.125** |
| *hrv* | 4.615 | 3.845 | 3.608 | **3.588** | *tur* | 4.846 | **3.509** | 4.282 | 3.552 |
| *hun* | 4.806 | 3.589 | 3.709 | **3.522** | *usp* | 4.529 | 2.114 | 2.189 | **2.073** |
| *ind* | 4.377 | 3.317 | 3.258 | **2.420** | *vie* | 5.185 | 3.018 | 3.751 | **3.015** |
| *isl* | 4.744 | 3.174 | 3.703 | **3.101** | *wal* | 4.398 | 2.986 | 3.623 | **2.278** |
| *ita* | 4.370 | 3.384 | 3.196 | **3.178** | *wol* | 4.621 | 2.898 | 2.968 | **2.826** |
| *jak* | 4.532 | 2.113 | 2.650 | **2.126** | *xho* | 4.561 | 3.415 | **3.208** | 3.289 |
| *jiv* | 4.338 | 3.413 | 3.475 | **2.504** | *zul* | 4.564 | 2.625 | 2.866 | **2.622** |
| *kab* | 4.649 | **2.783** | 3.574 | 2.800 | ALL | 4.467 | 3.007 | 3.120 | **2.731** |

Table 3: BPC scores (lower is better) for the FEW-SHOT learning setting, with NINF, FITU and UNIV priors. Colors define the split in which each language (rows) has been held out.

**Language Model** We implement the LSTM following the best practices and hyper-parameter settings indicated for language modelling by Merity et al. (2017, 2018). In particular, we optimize the weights with Adam (Kingma and Ba, 2014) and a non-monotonically decayed learning rate: its value is initialized as $10^{-4}$ and decreases by a factor of 10 every $1/3$rd of the total epochs. The maximum number of epochs amounts to 6 for $\mathcal{D}_t$, with early stopping based on development set performance, and the maximum number of epochs is 25 for $\mathcal{D}_e$.

Moreover, we extend the model to multilingual joint training. In each iteration, we sample a language proportionally to the amount of its data: $p(\ell) \propto |\mathcal{D}_t|$, in order not to exhaust examples from resource-lean languages in the early phase of training. Then, we sample without replacement from $\mathcal{D}_t$ a mini-batch of 128 sequences with a variable maximum sequence length.[12] This length is sampled from a distribution $m \sim \mathcal{N}(\mu = 125, \sigma = 5)$.[13] Each epoch comes to an end when all the data sequences have been sampled.

We apply several techniques of dropout for regu-

[12] This avoids creating insurmountable boundaries to back-propagation though time (Tallec and Ollivier, 2017).

[13] The learning rate is therefore scaled by $\frac{m}{\mu}$ and $\frac{\mathcal{D}}{L \cdot \mathcal{D}^\ell}$.

larization, including variational dropout (Gal and Ghahramani, 2016), which applies an identical mask to all time steps, with $p = 0.1$ for character embeddings and intermediate hidden states and $p = 0.4$ for the output hidden states. DropConnect (Wan et al., 2013) is applied to the model parameters $U$ of the first hidden layer with $p = 0.2$.

Following Merity et al. (2017), the underlying language model architecture consists of 3 hidden layers with 1,840 hidden units each. The dimensionality of the character embeddings is 400. For conditional language models, the dimensionality of $f(\mathbf{t})$ is set to 115 with the OEST method based on concatenation (Östling and Tiedemann, 2017), and 4 (due to memory limitations) in the PLAT method based on meta-networks (Platanios et al., 2018). For the regularizer in eq. (17), we perform grid search over the hyperparameter $\lambda$: we finally select a value of $10^5$ for UNIV and $10^{-5}$ for NINF.

**Regimes of Data Paucity** We explore different regimes of data paucity for the held-out languages:
• ZERO-SHOT transfer setting: we split the sample of 77 languages into 4 subsets. The languages in each subset are held out in turn, and we use their test set for evaluation.[14] For each subset, we further randomly choose 5 languages whose development set is used for validation. The training set of the rest of the languages is used to estimate a prior over network parameters via the Laplace approximation.
• FEW-SHOT transfer setting: on top of the zero-shot setting, we use the prior to perform MAP inference over a small sample (100 sentences) from the training set of each held-out language.
• JOINT multilingual setting: $\mathcal{D}_e$ includes the full training set for all 77 languages, including held-out languages. This works as a ceiling for the expected performance of language transfer models.

## 6 Results and Analysis

The results for our experiments are grouped in Table 1 for the ZERO-SHOT regime, Table 3 for the FEW-SHOT regime, and in Table 2 for the JOINT multilingual regime. The scores represent Bits Per Character (BPC) (Graves, 2013): this metric is simply defined as the average negative log-likelihood of test data divided by $\log 2$. We compare the results along the following dimensions:

[14]Holding out each language individually would not increase the sample of training languages significantly, while inflating the number of experimental runs needed.

**Informativeness of Prior** Our main result is that the UNIV prior consistently outperforms the NINF prior across the board and by a large margin in both ZERO-SHOT and FEW-SHOT settings. The scores for the naïvest baseline, ZERO-SHOT NINF BARE, are considerably worse than with both ZERO-SHOT UNIV models: this suggests that the transfer of information on character sequences is meaningful. The lowest BPC reductions are observed for languages like Vietnamese (15.94% error reduction) or Highland Chinantec (19.28%) where character inventories or distributions are unmatched in other languages. Moreover, the ZERO-SHOT UNIV models are on a par or better than even the FEW-SHOT NINF models. In other words, the most helpful supervision comes from a universal prior rather than from a small in-language sample of sentences. This demonstrates that the UNIV prior is truly imbued with universal linguistic knowledge that facilitates learning of previously unseen languages.

The averaged BPC score for the other baseline without a prior, FINE-TUNE is 3.007 for FEW-SHOT OEST, to be compared with 2.731 BPC of UNIV. Note that fine-tuning is an extremely competitive baseline, as it lies at the core of most state-of-the-art NLP models (Peters et al., 2019). Hence, this result demonstrates the usefulness of a Bayesian treatment of transfer learning.

**Conditioning on Typological Information** Another important result regards the fact that conditioning language models on typological features yield opposite effects in the ZERO-SHOT and FEW-SHOT settings. By comparing the BARE and OEST models' columns in Table 1, the non-conditional baseline BARE is superior for 71 / 77 languages (the exceptions being Chamorro, Croatian, Italian, Swazi, Swedish, and Tuareg). On the other hand, the same columns in Table 3 and Table 2 reveal an opposite pattern: OEST outperforms the BARE baseline in 70 / 77 languages. Finally, OEST surpasses the BARE baseline in the JOINT setting for 76 / 77 languages (save Q'eqchi').

We also also take into consideration an alternative conditioning method, namely PLAT. For clarity's sake, we exclude this batch of results from Table 1 and Table 3, as this method proves to be consistently worse than OEST. In fact, the average BPC of PLAT amounts to 5.479 in the ZERO-SHOT setting and 3.251 in the FEW-SHOT setting. These scores have to be compared with 4.691 and 2.731 for OEST, respectively.

The possible explanation behind the mixed evidence on the success of typological features points to some intrinsic flaws of typological databases. Ponti et al. (2018a) has shown how their feature granularity may be too coarse to be reconciled with data-driven probabilistic models, and their limited coverage of features introduces noise as missing values have to be inferred. As a result, language models seem to be damaged by typological features in absence of data, whereas they find a way to follow their guidance when at least a small sample of sentences is available in the FEW-SHOT setting.

**Data Paucity** Different regimes of data paucity display uneven levels of performance. The best models for each setting (ZERO-SHOT UNIV BARE, FEW-SHOT UNIV OEST, and JOINT OEST) reveal large gaps between their average scores. Hence, in-language supervision should be still considered unsubstitutable, and transferred language models still lag behind their resource-rich equivalents.

## 7 Related Work

LSTMs have been probed for an inductive bias in capturing syntactic dependencies (Linzen et al., 2016) and grammaticality judgements (Marvin and Linzen, 2018; Warstadt et al., 2018). Ravfogel et al. (2019) have extended the scope of this analysis to typologically different languages through *synthetic* variations of English. In this work, we aim to model the inductive bias explicitly by constructing a prior over the space of neural network parameters.

Few-shot word-level language modelling for truly under-resourced languages such as Yongning Na has been investigated by Adams et al. (2017) with the aid of a bilingual lexicon. Vinyals et al. (2016) and Munkhdalai and Trischler (2018) proposed novel architectures (Matching Networks and LSTMs augmented with Hebbian Fast Weights, respectively) for rapid associative learning in English, and evaluated them in few-shot cloze tests. In this respect, our work is novel in pushing the problem to its most complex formulation, zero-shot inference, and in taking into account the largest sample of languages for language modelling to date.

In addition to the set of standard architectures considered in our work, there are also alternatives to conditional language modelling. Kalchbrenner and Blunsom (2013) used encoded features as additional biases in recurrent layers. Kiros et al. (2014) put forth a log-bilinear model that allows for a "multiplicative interaction" between hidden representations and input features (such as images). With a similar device, but a different gating method, Tsvetkov et al. (2016) trained a phoneme-level joint multilingual model of words conditioned on typological features from Moran et al. (2014).

The use of the Laplace method for neural transfer learning has been proposed by Kirkpatrick et al. (2017), inspired by synaptic consolidation in neuroscience to avoid catastrophic forgetting. Kochurov et al. (2018) tackled the problem of continuous learning from independent data portions for a single fixed task by approximating the posterior probabilities through stochastic variational inference. Ritter et al. (2018) substitute diagonal Laplace approximation with a Kronecker factored method, leading to better uncertainty estimates. Finally, the regularizer proposed by Duong et al. (2015) for cross-lingual dependency parsing can be interpreted as a prior for Maximum A Posteriori estimation where the covariance is an identity matrix.

## 8 Conclusions

In this work, we proposed a Bayesian approach to cross-lingual language modeling transfer. We created a universal prior over neural network weights that is capable of generalizing well to new languages riddled by data paucity, by Laplace-approximating the posterior of the weights conditioned on the data from a sample of training languages. Based on the results of character-level language modelling on a sample of 77 languages, we demonstrated the superiority of the universal prior over uninformative priors and uniform priors (i.e., the widespread fine-tuning approach) in both zero-shot and few-shot settings. Moreover, we showed that adding language-specific side information drawn from typological databases to the universal prior further increases the levels of performance in the few-shot regime. While we also showed that language transfer still lags behind multilingual joint learning when sufficient in-language data are available, our work is the first step towards bridging this gap in the future.

## Acknowledgements

This work is supported by the ERC Consolidator Grant LEXICAL (no 648909). RR was partially funded by ISF personal grants No. 1625/18. We would like to thank the three anonymous reviewers for their helpful comments and suggestions.

## References


Oliver Adams, Adam Makarucha, Graham Neubig, Steven Bird, and Trevor Cohn. 2017. Cross-lingual word embeddings for low-resource language modeling. In *Proceedings of EACL*, pages 937–947.

Adriano Azevedo-Filho and Ross D. Shachter. 1994. Laplace's method approximations for probabilistic inference in belief networks with continuous variables. In *Proceedings of UAI*, pages 28–36.

Cecil H. Brown, Eric W. Holman, Søren Wichmann, and Viveka Velupillai. 2008. Automated classification of the world's languages: A description of the method and preliminary results. *STUF-Language Typology and Universals Sprachtypologie und Universalienforschung*, 61(4):285–308.

Noam Chomsky. 1959. A review of BF Skinner's verbal behavior. *Language*, 35(1):26–58.

Noam Chomsky. 1978. A naturalistic approach to language and cognition. *Cognition and Brain Theory*, 4(1):3–22.

Christos Christodouloupoulos and Mark Steedman. 2015. A massively parallel corpus: The Bible in 100 languages. *Language Resources and Evaluation*, 49(2):375–395.

Chris Collins and Richard Kayne. 2009. Syntactic structures of the world's languages. `http://sswl.railsplayground.net/`.

Ryan Cotterell, Sebastian J. Mielke, Jason Eisner, and Brian Roark. 2018. Are all languages equally hard to language-model? In *Proceedings of NAACL-HLT*, pages 536–541.

Thomas M. Cover and Joy A. Thomas. 2006. *Elements of Information Theory*. Wiley-Interscience.

William Croft. 2002. *Typology and Universals*. Cambridge University Press.

William Croft, Dawn Nordquist, Katherine Looney, and Michael Regan. 2017. Linguistic typology meets Universal Dependencies. In *Proceedings of TLT*, pages 63–75.

Matthew S. Dryer and Martin Haspelmath, editors. 2013. *WALS Online*. Max Planck Institute for Evolutionary Anthropology.

Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proceedings of ACL*, pages 845–850.

Nicholas Evans and Stephen C. Levinson. 2009. The myth of language universals: Language diversity and its importance for cognitive science. *Behavioral and Brain Sciences*, 32(5):429–448.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Proceedings of NeurIPS*, pages 1019–1027.

Andrew Gelman, Hal S. Stern, John B. Carlin, David B. Dunson, Aki Vehtari, and Donald B. Rubin. 2013. *Bayesian data analysis*. Chapman and Hall/CRC.

Daniela Gerz, Ivan Vulić, Edoardo Ponti, Jason Naradowsky, Roi Reichart, and Anna Korhonen. 2018a. Language modeling for morphologically rich languages: Character-aware modeling for word-level prediction. *Transactions of the Association of Computational Linguistics*, 6:451–465.

Daniela Gerz, Ivan Vulić, Edoardo Maria Ponti, Roi Reichart, and Anna Korhonen. 2018b. On the relation between linguistic typology and (limitations of) multilingual language modeling. In *Proceedings of EMNLP*, pages 316–327.

Gary Martin Gilligan. 1989. *A cross-linguistic approach to the pro-drop parameter*. Ph.D. thesis, University of Southern California.

Giorgio Graffi. 1980. Universali di Greenberg e grammatica generativa in la nozione di tipo e le sue articolazioni nelle discipline del linguaggio. *Lingua e Stile Bologna*, 15(3):371–387.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850.

Joseph H. Greenberg. 1963. Some universals of grammar with particular reference to the order of meaningful elements. *Universals of Language*, 2:73–113.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of EMNLP*, pages 1700–1709.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.

Ryan Kiros, Ruslan Salakhutdinov, and Rich Zemel. 2014. Multimodal neural language models. In *Proceedings of ICML*, pages 595–603.

Max Kochurov, Timur Garipov, Dmitry Podoprikhin, Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. 2018. Bayesian incremental learning for deep neural networks. In *Proceedings of ICLR (Workshop Papers)*.

András Kornai. 2013. Digital language death. *PloS One*, 8(10):e77056.

Julie Anne Legate and Charles D Yang. 2002. Empirical re-assessment of stimulus poverty arguments. *The Linguistic Review*, 18(1-2):151–162.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Patrick Littell, David R. Mortensen, Ke Lin, Katherine Kairis, Carlisle Turner, and Lori Levin. 2017. URIEL and lang2vec: Representing languages as typological, geographical, and phylogenetic vectors. In *Proceedings of EACL*, pages 8–14.

Rebecca Marvin and Tal Linzen. 2018. Targeted syntactic evaluation of language models. In *Proceedings of EMNLP*, pages 1192–1202.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017. Regularizing and optimizing LSTM language models. *arXiv preprint arXiv:1708.02182*.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. An analysis of neural language modeling at multiple scales. *arXiv preprint arXiv:1803.08240*.

Sebastian J. Mielke, Ryan Cotterell, Kyle Gorman, Brian Roark, and Jason Eisner. 2019. What kind of language is hard to language-model? In *Proceedings of ACL*, pages 4975–4989.

Steven Moran, Daniel McCloy, and Richard Wright, editors. 2014. *PHOIBLE Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Tsendsuren Munkhdalai and Adam Trischler. 2018. Metalearning with Hebbian fast weights. *arXiv preprint arXiv:1807.05076*.

Robert Östling and Jörg Tiedemann. 2017. Continuous multilinguality with language vectors. In *Proceedings of the EACL*, volume 2, pages 644–649.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of ICML*, pages 1310–1318.

Matthew Peters, Sebastian Ruder, and Noah A Smith. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks. *arXiv preprint arXiv:1903.05987*.

Emmanouil Antonios Platanios, Mrinmaya Sachan, Graham Neubig, and Tom Mitchell. 2018. Contextual parameter generation for universal neural machine translation. In *Proceedings of EMNLP*, pages 425–435.

Edoardo Maria Ponti, Helen O'Horan, Yevgeni Berzak, Ivan Vulić, Roi Reichart, Thierry Poibeau, Ekaterina Shutova, and Anna Korhonen. 2018a. Modeling language variation and universals: A survey on typological linguistics for natural language processing. *arXiv preprint arXiv:1807.00914*.

Edoardo Maria Ponti, Roi Reichart, Anna Korhonen, and Ivan Vulić. 2018b. Isomorphic transfer of syntactic structures in cross-lingual NLP. In *Proceedings of ACL*, pages 1531–1542.

Shauli Ravfogel, Yoav Goldberg, and Tal Linzen. 2019. Studying the inductive biases of RNNs with synthetic variations of natural languages. In *Proceedings of NAACL-HLT*.

Shauli Ravfogel, Yoav Goldberg, and Francis Tyers. 2018. Can LSTM learn to capture agreement? The case of Basque. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 98–107.

Hippolyt Ritter, Aleksandar Botev, and David Barber. 2018. Online structured Laplace approximations for overcoming catastrophic forgetting. In *Proceedings of NIPS*, pages 3738–3748.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

Gary F. Simons. 2017. *Ethnologue: Languages of the world*, 22nd edition. Dallas, Texas: SIL International.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of NIPS*, pages 3104–3112.

Corentin Tallec and Yann Ollivier. 2017. Unbiasing truncated backpropagation through time. *arXiv preprint arXiv:1705.08209*.

Yulia Tsvetkov, Sunayana Sitaram, Manaal Faruqui, Guillaume Lample, Patrick Littell, David Mortensen, Alan W. Black, Lori Levin, and Chris Dyer. 2016. Polyglot neural language models: A case study in cross-lingual phonetic representation learning. In *Proceedings of NAACL-HLT*, pages 1357–1366.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. In *Proceedings of NIPS*, pages 3630–3638.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. 2013. Regularization of neural networks using DropConnect. In *Proceedings of ICML*, pages 1058–1066.

Alex Warstadt, Amanpreet Singh, and Samuel .R Bowman. 2018. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*.

# What Gets Echoed? Understanding the "Pointers" in Explanations of Persuasive Arguments

**David Atkinson** and **Kumar Bhargav Srinivasan** and **Chenhao Tan**
Department of Computer Science
University of Colorado Boulder
Boulder, CO
david.i.atkinson, kumar.srinivasan, chenhao.tan@colorado.edu

## Abstract

Explanations are central to everyday life, and are a topic of growing interest in the AI community. To investigate the process of providing natural language explanations, we leverage the dynamics of the `/r/ChangeMyView` subreddit to build a dataset with 36K naturally occurring explanations of why an argument is persuasive. We propose a novel word-level prediction task to investigate how explanations selectively reuse, or *echo*, information from what is being explained (henceforth, *explanandum*). We develop features to capture the properties of a word in the explanandum, and show that our proposed features not only have relatively strong predictive power on the echoing of a word in an explanation, but also enhance neural methods of generating explanations. In particular, while the non-contextual properties of a word itself are more valuable for stopwords, the interaction between the constituent parts of an explanandum is crucial in predicting the echoing of content words. We also find intriguing patterns of a word being echoed. For example, although nouns are generally less likely to be echoed, subjects and objects can, depending on their source, be more likely to be echoed in the explanations.

## 1 Introduction

Explanations are essential for understanding and learning (Keil, 2006). They can take many forms, ranging from everyday explanations for questions such as why one likes Star Wars, to sophisticated formalization in the philosophy of science (Salmon, 2006), to simply highlighting features in recent work on interpretable machine learning (Ribeiro et al., 2016).

Although everyday explanations are mostly encoded in natural language, natural language explanations remain understudied in NLP, partly due to a lack of appropriate datasets and problem formulations. To address these challenges, we leverage `/r/ChangeMyView`, a community dedicated to sharing counterarguments to controversial views on Reddit, to build a sizable dataset of naturally-occurring explanations. Specifically, in `/r/ChangeMyView`, an original poster (OP) first delineates the rationales for a (controversial) opinion (e.g., in Table 1, "most hit music artists today are bad musicians"). Members of `/r/ChangeMyView` are invited to provide counterarguments. If a counterargument changes the OP's view, the OP awards a $\Delta$ to indicate the change and is required to *explain why the counterargument is persuasive*. In this work, we refer to what is being explained, including both the original post and the persuasive comment, as the *explanandum*.[1]

An important advantage of explanations in `/r/ChangeMyView` is that the explanandum contains most of the required information to provide its explanation. These explanations often select key counterarguments in the persuasive comment and connect them with the original post. As shown in Table 1, the explanation naturally points to, or *echoes*, part of the explanandum (including both the persuasive comment and the original post) and in this case highlights the argument of "music serving different purposes."

These naturally-occurring explanations thus enable us to computationally investigate the selective nature of explanations: "people rarely, if ever, expect an explanation that consists of an actual and complete cause of an event. Humans are adept at selecting one or two causes from a sometimes infinite number of causes to be the explanation" (Miller, 2018). To understand the selective process of providing explanations, we formulate a word-level task to predict whether a word in an

[1]The plural of *explanandum* is explananda.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2911–2921,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| |
|---|
| **Original post (OP):** CMV: most hit music artists today are bad musicians<br>Now I know, music is art and art has no rules, but this is only so true. Movies are art too but I think most of us can agree the emoji movie was objectively bad. That aside: I really feel like once you remove the persona and performances of the artists from the "top 40" songs and listen to them as just a song, most are objectively bad. They're super repetitive, the lyrics and painfully generic, and there's hardly ever anything new or challenging. And from what I understand most of these artists don't even write their own songs. Of course there are exceptions but I find them to be extremely rare. It seems to me they're only popular become of who they are and how they look/perform. I realize this is probably a very snobbish view which is why I want to be enlightened, so can anyone convince me otherwise? Are they actually good musicians or just good performers? [one more paragraph ...] |
| **Persuasive comment (PC):** Music appreciation is a skill, and it's all about pattern recognition.<br>When we're children, we need songs that are really simple, repetitive and with easy to recognize patterns. The younger we are, the simpler the songs. Toddlers like nursery rhymes, lullabies, jingles. Teens like pop music. And teens spend more on music than anyone else. [four more paragraphs ...]<br>Lastly, you have to consider that music can be listened to in different ways and for different purposes. You can listen to it alone on headphones, and think about what it means and how it makes you feel. Or you can dance to it with your friends. Or maybe you need something on in the background during a dinner party, or a house party, or while you study, or are trying to fall asleep, or work out. Pop music is really good in some of these situations, really bad in others. But it serves a definite purpose and isn't bad in any essential way. |
| **Explanation:** $\Delta$ I guess I never *really looked* at it as *music serving different purposes*. I can *see* how *pop music* fills a certain *purpose*, and I guess the *artist* does *n't* necessarily have to be the *one* to *write* the *song* (although I *appreciate* it when they do). |

Table 1: An illustration of the pointers in an example explanation of `/r/ChangeMyView`. We color the words in the explanation based on whether it is used in the original post (e.g., *artist*), in the persuasive comment (e.g., *purpose*), or both (e.g., *music*). We stem all the words before matching and do not color stopwords for readability.

explanandum will be echoed in its explanation.

Inspired by the observation that words that are likely to be echoed are either frequent or rare, we propose a variety of features to capture how a word is used in the explanandum as well as its non-contextual properties in Section 4. We find that a word's usage in the original post and in the persuasive argument are similarly related to being echoed, except in part-of-speech tags and grammatical relations. For instance, verbs in the original post are less likely to be echoed, while the relationship is reversed in the persuasive argument.

We further demonstrate that these features can significantly outperform a random baseline and even a neural model with significantly more knowledge of a word's context. The difficulty of predicting whether content words (i.e., non-stopwords) are echoed is much greater than that of stopwords,[2] among which adjectives are the most difficult and nouns are relatively the easiest. This observation highlights the important role of nouns in explanations. We also find that the relationship between a word's usage in the original post and in the persuasive comment is crucial for predicting the echoing of content words. Our proposed features can also improve the performance of pointer generator networks with coverage in generating explanations (See et al., 2017).

[2]We use the stopword list in NLTK.

To summarize, our main contributions are:

- We highlight the importance of computationally characterizing human explanations and formulate a concrete problem of predicting how information is selected from explananda to form explanations, including building a novel dataset of naturally-occurring explanations.
- We provide a computational characterization of natural language explanations and demonstrate the U-shape in which words get echoed.
- We identify interesting patterns in what gets echoed through a novel word-level classification task, including the importance of nouns in shaping explanations and the importance of contextual properties of both the original post and persuasive comment in predicting the echoing of content words.
- We show that vanilla LSTMs fail to learn some of the features we develop and that the proposed features can even improve performance in generating explanations with pointer networks.

Our code and dataset is available at `https://chenhaot.com/papers/explanation-pointers.html`.

## 2 Related Work

To provide background for our study, we first present a brief overview of explanations for the NLP community, and then discuss the connection

of our study with pointer networks, linguistic accommodation, and argumentation mining.

The most developed discussion of explanations is in the philosophy of science. Extensive studies aim to develop formal models of explanations (e.g., the deductive-nomological model in Hempel and Oppenheim (1948), see Salmon (2006) and Woodward (2005) for a review). In this view, explanations are like proofs in logic. On the other hand, psychology and cognitive sciences examine "everyday explanations" (Keil, 2006; Lombrozo, 2006). These explanations tend to be selective, are typically encoded in natural language, and shape our understanding and learning in life despite the absence of "axioms." Please refer to Wilson and Keil (1998) for a detailed comparison of these two modes of explanation.

Although explanations have attracted significant interest from the AI community thanks to the growing interest on interpretable machine learning (Doshi-Velez and Kim, 2017; Lipton, 2016; Guidotti et al., 2019), such studies seldom refer to prior work in social sciences (Miller, 2018). Recent studies also show that explanations such as highlighting important features induce limited improvement on human performance in detecting deceptive reviews and media biases (Lai and Tan, 2019; Horne et al., 2019). Therefore, we believe that developing a computational understanding of everyday explanations is crucial for explainable AI. Here we provide a data-driven study of everyday explanations in the context of persuasion.

In particular, we investigate the "pointers" in explanations, inspired by recent work on pointer networks (Vinyals et al., 2015). Copying mechanisms allow a decoder to generate a token by copying from the source, and have been shown to be effective in generation tasks ranging from summarization to program synthesis (See et al., 2017; Ling et al., 2016; Gu et al., 2016). To the best of our knowledge, our work is the first to investigate the phenomenon of pointers in explanations.

Linguistic accommodation and studies on quotations also examine the phenomenon of reusing words (Danescu-Niculescu-Mizil et al., 2011; Giles and Ogay, 2007; Leskovec et al., 2009; Simmons et al., 2011). For instance, Danescu-Niculescu-Mizil et al. show that power differences are reflected in the echoing of function words; Tan et al. (2018) find that news media prefer to quote locally distinct sentences in political debates. In comparison, our word-level formulation presents a fine-grained view of echoing words, and puts a stronger emphasis on content words than work on linguistic accommodation.

Finally, our work is concerned with an especially challenging problem in social interaction: persuasion. A battery of studies have done work to enhance our understanding of persuasive arguments (Wang et al., 2017; Zhang et al., 2016; Habernal and Gurevych, 2016; Lukin et al., 2017; Durmus and Cardie, 2018), and the area of argumentation mining specifically investigates the structure of arguments (Lippi and Torroni, 2016; Walker et al., 2012; Somasundaran and Wiebe, 2009). We build on previous work by Tan et al. (2016) and leverage the dynamics of `/r/ChangeMyView`. Although our findings are certainly related to the persuasion process, we focus on understanding the self-described reasons for persuasion, instead of the structure of arguments or the factors that drive effective persuasion.

## 3 Dataset

Our dataset is derived from the `/r/ChangeMyView` subreddit, which has more than 720K subscribers (Tan et al., 2016). `/r/ChangeMyView` hosts conversations where someone expresses a view and others then try to change that person's mind. Despite being fundamentally based on argument, `/r/ChangeMyView` has a reputation for being remarkably civil and productive (CMV moderators, 2019a), e.g., a journalist wrote "In a culture of brittle talking points that we guard with our lives, Change My View is a source of motion and surprise" (Heffernan, 2018).

The delta mechanism in `/r/ChangeMyView` allows members to acknowledge opinion changes and enables us to identify *explanations* for opinion changes (CMV moderators, 2019b). Specifically, it requires "Any user, whether they're the OP or not, should reply to a comment that changed their view with a delta symbol and *an explanation of the change*." As a result, we have access to tens of thousands of naturally-occurring explanations and associated explananda. In this work, we focus on the opinion changes of the original posters.

Throughout this paper, we use the following terminology:

- An **original post (OP)** is an initial post where the original poster justifies his or her opinion. We also use OP to refer to the original poster.

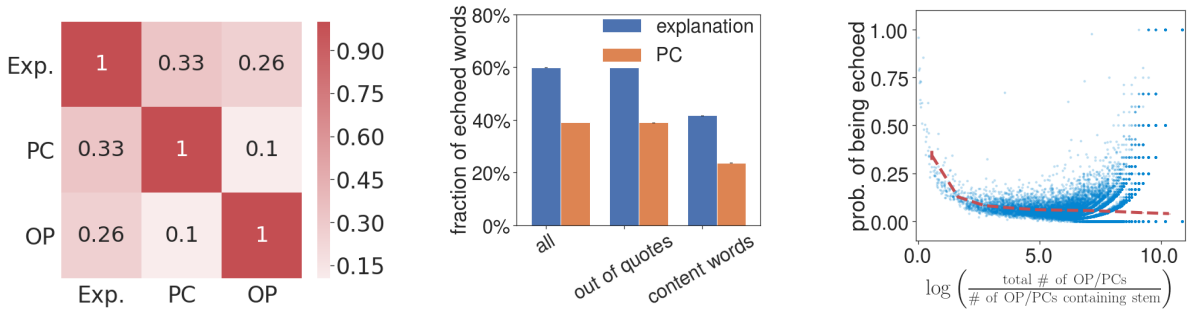


(a) Length correlations.

(b) Fraction of words that are echoed from the explanandum.

(c) Word-level echoing probability vs. document frequency.

Figure 1: Figure 1a shows the pairwise Pearson correlation coefficient between lengths of OP, PC, and explanation (all values are statistically significant with $p < 1e{-}10$). Figure 1b shows the average fraction of words in an explanation that are in its OP or PC, and the fraction of words in a PC that are in its OP. In Figure 1c, the $y$-axis represents the probability of a word in an OP or PC being echoed in the explanation, while the $x$-axis shows the inverse document frequency of that word in training data. For each document frequency decile, we calculate the probability of a word in that decile being echoed, and plot those probabilities with the red line. In Figure 1b and Figure 1c, the (small) error bars represent standard errors.

- A **persuasive comment (PC)** is a comment that directly leads to an opinion change on the part of the OP (i.e., winning a $\Delta$).
- A **top-level comment** is a comment that directly replies to an OP, and `/r/ChangeMyView` requires the top-level comment to "challenge at least one aspect of OPs stated view (however minor), unless they are asking a clarifying question."
- An **explanation** is a comment where an OP acknowledges a change in his or her view and provides an explanation of the change. As shown in Table 1, the explanation not only provides a rationale, it can also include other discourse acts, such as expressing gratitude.

Using `https://pushshift.io`, we collect the posts and comments in `/r/ChangeMyView` from January 17th, 2013 to January 31st, 2019, and extract tuples of (OP, PC, explanation). We use the tuples from the final six months of our dataset as the test set, those from the six months before that as the validation set, and the remaining tuples as the training set. The sets contain 5,270, 5,831, and 26,617 tuples respectively. Note that there is no overlap in time between the three sets and the test set can therefore be used to assess generalization including potential changes in community norms and world events.

**Preprocessing.** We perform a number of preprocessing steps, such as converting blockquotes in Markdown to quotes, filtering explicit edits made by authors, mapping all URLs to a special `@url@` token, and replacing hyperlinks with the link text. We ignore all triples that contain any deleted comments or posts. We use spaCy for tokenization and tagging (Honnibal and Montani, 2017). We also use the NLTK implementation of the Porter stemming algorithm to store the stemmed version of each word, for later use in our prediction task (Loper and Bird, 2002; Porter, 1980). Refer to the supplementary material for more information on preprocessing.

**Data statistics.** Table 2 provides basic statistics of the training tuples and how they compare to other comments. We highlight the fact that PCs are on average longer than top-level comments, suggesting that PCs contain substantial counterarguments that directly contribute to opinion change. Therefore, we simplify the problem by focusing on the (OP, PC, explanation) tuples and ignore any other exchanges between an OP and a commenter.

Below, we highlight some notable features of explanations as they appear in our dataset.

**The length of explanations shows stronger correlation with that of OPs and PCs than between OPs and PCs (Figure 1a).** This observation indicates that explanations are somehow better related with OPs and PCs than PCs are with OPs in terms of language use. A possible reason is that the explainer combines their natural tendency towards length with accommodating the PC.

**Explanations have a greater fraction of "pointers" than do persuasive comments (Figure 1b).** We measure the likelihood of a word in an explanation being copied from either its OP or PC and provide a similar probability for a PC for copying from its OP. As we discussed in Section

| | count | #sentences | #words |
|---|---|---|---|
| Tuples of (OP, PC, Explanations) | | | |
| Original Posts | 26.3K | 16.8 | 298.8 |
| Persuasive comments | 26.3K | 12.6 | 218.3 |
| Explanations | 26.3K | 5.3 | 79.8 |
| All of `/r/ChangeMyView` during the training period | | | |
| Original posts | 93.4k | 13.2 | 172.6 |
| Top-level comments | 681.6k | 9.1 | 147.4 |
| All comments | 3.6M | 6.5 | 98.9 |

Table 2: Basic statistics of the training dataset.

1, the words in an explanation are much more likely to come from the existing discussion than are the words in a PC (59.8% vs 39.0%). This phenomenon holds even if we restrict ourselves to considering words outside quotations, which removes the effect of quoting other parts of the discussion, and if we focus only on content words, which removes the effect of "reusing" stopwords.

**Relation between a word being echoed and its document frequency (Figure 1c).** Finally, as a preview of our main results, the document frequency of a word from the explanandum is related to the probability of being echoed in the explanation. Although the average likelihood declines as the document frequency gets lower, we observe an intriguing U-shape in the scatter plot.[3] In other words, the words that are most likely to be echoed are either unusually frequent or unusually rare, while most words in the middle show a moderate likelihood of being echoed.

## 4 Understanding the Pointers in Explanations

To further investigate how explanations select words from the explanandum, we formulate a word-level prediction task to predict whether words in an OP or PC are echoed in its explanation. Formally, given a tuple of (OP, PC, explanation), we extract the unique stemmed words as $\mathcal{V}_{\text{OP}}, \mathcal{V}_{\text{PC}}, \mathcal{V}_{\text{EXP}}$. We then define the label for each word in the OP or PC, $w \in \mathcal{V}_{\text{OP}} \cup \mathcal{V}_{\text{PC}}$, based on the explanation as follows:

$$y_w = \begin{cases} 1 & \text{if } w \in \mathcal{V}_{\text{EXP}}, \\ 0 & \text{otherwise.} \end{cases}$$

Our prediction task is thus a straightforward binary classification task at the word level. We develop the following five groups of features to capture properties of how a word is used in the explanandum (see Table 3 for the full list):

- Non-contextual properties of a word. These features are derived directly from the word and capture the general tendency of a word being echoed in explanations.
- Word usage in an OP or PC (two groups). These features capture *how* a word is used in an OP or PC. As a result, for each feature, we have two values for the OP and PC respectively.
- How a word connects an OP and PC. These features look at the difference between word usage in the OP and PC. We expect this group to be the most important in our task.
- General OP/PC properties. These features capture the general properties of a conversation. They can be used to characterize the background distribution of echoing.

Table 3 further shows the intuition for including each feature, and condensed $t$-test results after Bonferroni correction. Specifically, we test whether the words that were echoed in explanations have different feature values from those that were not echoed. In addition to considering all words, we also separately consider stopwords and content words in light of Figure 1c. Here, we highlight a few observations:

- Although we expect more complicated words (*#characters*) to be echoed more often, this is not the case on average. We also observe an interesting example of Simpson's paradox in the results for Wordnet depth (Blyth, 1972): shallower words are more likely to be echoed across all words, but deeper words are more likely to be echoed in content words and stopwords.
- OPs and PCs generally exhibit similar behavior for most features, except for part-of-speech and grammatical relation (subject, object, and other.) For instance, verbs in an OP are less likely to be echoed, while verbs in a PC are more likely to be echoed.

[3] A similar U-shape exists if we examine the probability of a PC echoing its OP, but does not show up if we compare an OP echoing a different, randomly chosen OP. It is worth noting that PCs can also be viewed as explaining why the OP is problematic. However, constructing a PC involves selecting from a large number of possible counter perspectives (all of which are unobservable). See the supplementary material for a detailed discussion.

| Feature group | Features and intuitions | Echoed? |
|---|---|---|
| Non-contextual properties | Inverse document frequency. As shown in Figure 1c, although document frequency can have non-linear relationships with being copied, the average echoing probability is greater for more common words. | $\downarrow\downarrow\downarrow\downarrow$ |
| | Number of characters. Longer words tend to be more complicated, and may be more likely to be echoed as part of the core argument. | $\downarrow\downarrow\downarrow\downarrow$ |
| | Wordnet depth. Similar to number of characters, the depth in wordnet can indicate the complexity of a word and we expect words with greater depth to be echoed. | $\downarrow\downarrow\downarrow\downarrow^{RC,RS}$ |
| | Echoing likelihood. We also compute the general tendency of a word being echoed in the training data. We expect the feature to be positively correlated with the label. | $\uparrow\uparrow\uparrow\uparrow$ |
| How a word is used in an OP or PC (OP/PC usage) | Part-of-speech (POS) tags. We compute the percentage of times that the surface forms of a stemmed word appear as different part-of-speech tags. We expect nouns and verbs more likely to be echoed. Results: verb in an OP $\downarrow\downarrow\downarrow\downarrow^{RS}$, noun in an OP ($\downarrow\downarrow\downarrow\downarrow$), verb in a PC ($\uparrow\uparrow\uparrow\uparrow$), noun in a PC: $\downarrow\downarrow\downarrow\downarrow^{RC}$. | |
| | Subjects and objects from dependency labels. We compute the percentage of times that the word appears as subjects, objects, and others. We expect subjects and objects more likely to be echoed. Results: subjects in an OP: $\uparrow\uparrow\uparrow\uparrow$, objects in an OP: $\downarrow\downarrow\downarrow\downarrow^{RC}$, others in an OP: $\uparrow\uparrow\uparrow\uparrow^{RC}$, subjects in a PC: $\downarrow\downarrow\downarrow\downarrow$, objects in a PC: $\downarrow\downarrow\downarrow\downarrow$; others in a PC: $\uparrow\uparrow\uparrow\uparrow$. | |
| | (Normalized) term frequency. We expect frequent terms to be echoed. | $\uparrow\uparrow\uparrow\uparrow$ |
| | #surface forms. We expect words that have diverse surface forms to be echoed. | $\uparrow\uparrow\uparrow\uparrow$ |
| | Location. For words that never show up in an OP or PC, the default value is 0.5. We expect later words to be echoed. Results: location in an OP: $\uparrow\uparrow\uparrow\uparrow$ (not significant in stopwords); location in a PC: $\uparrow^{RS}$. | |
| | In quotes. We expect words in quotes to be echoed as they are already emphasized. | $\uparrow\uparrow\uparrow\uparrow$ |
| | Entity. We expect entities to be echoed. | $\uparrow\uparrow\uparrow\uparrow$ |
| How a word connects an OP and PC (OP-PC relation) | Occurs both in an OP and PC. | $\uparrow\uparrow\uparrow\uparrow$ |
| | #Surface forms in an OP but not in the PC. | $\downarrow\downarrow\downarrow\downarrow$ |
| | #Surface forms in a PC but not in the OP. | $\uparrow\uparrow\uparrow\uparrow^{RS}$ |
| | Jensen-Shannon (JS) distance between the OP and PC POS tag distributions of the word. | $\downarrow\downarrow\downarrow\downarrow$ |
| | JS distance between subjects/objects distributions of the word in an OP and PC. | $\downarrow\downarrow\downarrow\downarrow$ |
| General OP/PC properties | OP length. | $\downarrow\downarrow\downarrow\downarrow^{RS}$ |
| | PC length. | $\uparrow\uparrow\uparrow\uparrow$ |
| | Difference in #words. | $\downarrow\downarrow\downarrow\downarrow^{RS}$ |
| | Difference in average #characters in words. | $\downarrow\downarrow\downarrow\downarrow$ |
| | Part-of-speech tags distributional differences between an OP and PC. | $\downarrow\downarrow\downarrow\downarrow$ |
| | Depth of the PC in the thread. | $\uparrow\uparrow\uparrow\uparrow$ |

Table 3: Features to capture the properties of a word in the context of an explanandum. The last column shows $t-$test results after Bonferroni correction. $\uparrow$ indicates that words that are echoed have a greater value in the feature, while $\downarrow$ indicates the reverse. The number of arrows indicates the level of p-value: $\uparrow\uparrow\uparrow\uparrow$: $p < 0.0001$, $\uparrow\uparrow\uparrow$: $p < 0.001$, $\uparrow\uparrow$: $p < 0.01$, $\uparrow$: $p < 0.05$. $^{RC}$ and $^{RS}$ indicate that the direction is flipped in content words and stopwords respectively. We show significance testing results in a condensed format for space reasons. Refer to the supplementary material for the complete testing results.

- Although nouns from both OPs and PCs are less likely to be echoed, within content words, subjects and objects from an OP are more likely to be echoed. Surprisingly, subjects and objects in a PC are less likely to be echoed, which suggests that the original poster tends to refer back to their own subjects and objects, or introduce new ones, when providing explanations.
- Later words in OPs and PCs are more likely to be echoed, especially in OPs. This could relate to OPs summarizing their rationales at the end of their post and PCs putting their strongest points last.
- Although the number of surface forms in an OP or PC is positively correlated with being echoed, the differences in surface forms show reverse trends: the more surface forms of a word that show up only in the PC (i.e., not in the OP), the more likely a word is to be echoed. However, the reverse is true for the number of surface forms in only the OP. Such contrast echoes Tan et al. (2016), in which dissimilarity in word usage between the OP and PC was a predictive feature of successful persuasion.

## 5 Predicting Pointers

We further examine the effectiveness of our proposed features in a predictive setting. These features achieve strong performance in the word-level classification task, and can enhance neural models in both the word-level task and generating explanations. However, the word-level task remains challenging, especially for content words.

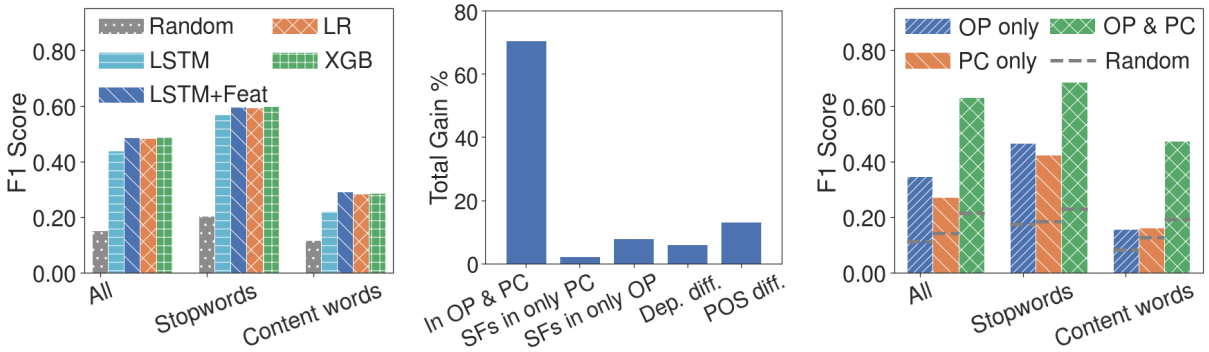


(a) Overall Performance comparison between models.

(b) Feature importance of an ablated model with OP-PC relation.

(c) Performance vs. word source.

Figure 2: Figure 2a presents the performance of different models. We evaluate the performance of each model on the subset of stopwords and content words. Our features with XGBoost and logistic regression outperform the vanilla LSTM model, and adding our features to the vanilla LSTM model achieves similar performance as XGBoost. Figure 2b shows the normalized total gain of the classifier only based on features in OP-PC relation, while Figure 2c further breaks down the performance based on where the words come from.

### 5.1 Experiment setup

We consider two classifiers for our word-level classification task: logistic regression and gradient boosting tree (XGBoost) (Chen and Guestrin, 2016). We hypothesized that XGBoost would outperform logistic regression because our problem is non-linear, as shown in Figure 1c.

To examine the utility of our features in a neural framework, we further adapt our word-level task as a tagging task, and use LSTM as a baseline. Specifically, we concatenate an OP and PC with a special token as the separator so that an LSTM model can potentially distinguish the OP from PC, and then tag each word based on the label of its stemmed version. We use GloVe embeddings to initialize the word embeddings (Pennington et al., 2014). We concatenate our proposed features of the corresponding stemmed word to the word embedding; the resulting difference in performance between a vanilla LSTM demonstrates the utility of our proposed features. We scale all features to $[0, 1]$ before fitting the models. As introduced in Section 3, we split our tuples of (OP, PC, explanation) into training, validation, and test sets, and use the validation set for hyperparameter tuning. Refer to the supplementary material for additional details in the experiment.

**Evaluation metric.** Since our problem is imbalanced, we use the F1 score as our evaluation metric. For the tagging approach, we average the labels of words with the same stemmed version to obtain a single prediction for the stemmed word. To establish a baseline, we consider a random method that predicts the positive label with 0.15 probability (the base rate of positive instances).

### 5.2 Prediction Performance

**Overall performance (Figure 2a).** Although our word-level task is heavily imbalanced, all of our models outperform the random baseline by a wide margin. As expected, content words are much more difficult to predict than stopwords, but the best F1 score in content words more than doubles that of the random baseline (0.286 vs. 0.116). Notably, although we strongly improve on our random baseline, even our best F1 scores are relatively low, and this holds true regardless of the model used. Despite involving more tokens than standard tagging tasks (e.g., Marcus et al. (1994) and Plank et al. (2016)), predicting whether a word is going to be echoed in explanations remains a challenging problem.

Although the vanilla LSTM model incorporates additional knowledge (in the form of word embeddings), the feature-based XGBoost and logistic regression models both outperform the vanilla LSTM model. Concatenating our proposed features with word embeddings leads to improved performance from the LSTM model, which becomes comparable to XGBoost. This suggests that our proposed features can be difficult to learn with an LSTM alone.

Despite the non-linearity observed in Figure 1c, XGBoost only outperforms logistic regression by a small margin. In the rest of this section, we use XGBoost to further examine the effectiveness

| | content | stop |
|---|---|---|
| all features | 0.286 | 0.600 |
| random | 0.116 | 0.205 |

| | forward | | backward | |
|---|---|---|---|---|
| | content | stop | content | stop |
| Non-contextual prop. | 0.177 | **0.582** | 0.285 | **0.561** |
| OP usage | 0.191 | 0.527 | 0.281 | 0.599 |
| PC usage | 0.233 | 0.520 | **0.275** | 0.598 |
| OP-PC relation | **0.280** | 0.542 | 0.289 | 0.600 |
| General OP/PC prop. | 0.153 | 0.266 | 0.285 | 0.598 |

Table 4: Ablation performance with XGBoost on content words and stopwords (each ablated model is tuned based on performance on all words). "forward" refers to only using a group of features, while "backward" refers to only removing a group of features.

of different groups of features, and model performance in different conditions.

**Ablation performance (Table 4).** First, if we only consider a single group of features, as we hypothesized, the relation between OP and PC is crucial and leads to almost as strong performance in content words as using all features. To further understand the strong performance of OP-PC relation, Figure 2b shows the feature importance in the ablated model, measured by the normalized total gain (see the supplementary material for feature importance in the full model). A word's occurrence in both the OP and PC is clearly the most important feature, with distance between its POS tag distributions as the second most important. Recall that in Table 3 we show that words that have similar POS behavior between the OP and PC are more likely to be echoed in the explanation.

Overall, it seems that word-level properties contribute the most valuable signals for predicting stopwords. If we restrict ourselves to only information in either an OP or PC, how a word is used in a PC is much more predictive of content word echoing (0.233 vs 0.191). This observation suggests that, for content words, the PC captures more valuable information than the OP. This finding is somewhat surprising given that the OP sets the topic of discussion and writes the explanation.

As for the effects of removing a group of features, we can see that there is little change in the performance on content words. This can be explained by the strong performance of the OP-PC relation on its own, and the possibility of the OP-PC relation being approximated by OP and PC usage. Again, word-level properties are valuable for strong performance in stopwords.

**Performance vs. word source (Figure 2c).** We further break down the performance by where a word is from. We can group a word based on whether it shows up only in an OP, a PC, or both OP and PC, as shown in Table 1. There is a striking difference between the performance in the three categories (e.g., for all words, 0.63 in OP & PC vs. 0.271 in PC only). The strong performance on words in both the OP and PC applies to stopwords and content words, even accounting for the shift in the random baseline, and recalls the importance of occurring both in OP and PC as a feature.

Furthermore, the echoing of words from the PC is harder to predict (0.271) than from the OP (0.347) despite the fact that words only in PCs are more likely to be echoed than words only in OPs (13.5% vs. 8.6%). The performance difference is driven by stopwords, suggesting that our overall model is better at capturing signals for stopwords used in OPs. This might relate to the fact that the OP and the explanation are written by the same author; prior studies have demonstrated the important role of stopwords for authorship attribution (Raghavan et al., 2010).

**Nouns are the most reliably predicted part-of-speech tag within content words (Table 5).** Next, we break down the performance by part-of-speech tags. We focus on the part-of-speech tags that are semantically important, namely, nouns, proper nouns, verbs, adverbs, and adjectives.

Prediction performance can be seen as a proxy for how reliably a part-of-speech tag is reused when providing explanations. Consistent with our expectations for the importance of nouns and verbs, our models achieve the best performance on nouns within content words. Verbs are more challenging, but become the least difficult tag to predict when we consider all words, likely due to stopwords such as "have." Adjectives turn out to be the most challenging category, suggesting that adjectival choice is perhaps more arbitrary than other parts of speech, and therefore less central to the process of constructing an explanation. The important role of nouns in shaping explanations resonates with the high recall rate of nouns in memory tasks (Reynolds and Flagg, 1976).

### 5.3 The Effect on Generating Explanations

One way to measure the ultimate success of understanding pointers in explanations is to be able to generate explanations. We use the pointer gen-

| | content | all | random |
|---|---|---|---|
| noun | 0.354 | 0.361 | 0.130 |
| adverb | 0.342 | 0.411 | 0.127 |
| verb | 0.306 | 0.466 | 0.122 |
| proper noun | 0.280 | 0.336 | 0.109 |
| adjective | 0.237 | 0.289 | 0.111 |

Table 5: Performance on five non-function part-of-speech tags (sorted by performance within content words). As a comparison, we also show the performance of the random baseline on content words, which is relatively stable across part-of-speech tags.

| | ROUGE-1 | ROUGE-2 | ROUGE-L |
|---|---|---|---|
| w/o features | 18.91 | 4.12 | 17.05 |
| with features | 22.01 | 3.93 | 19.02 |

Table 6: ROUGE scores (F1) on the test dataset (Lin, 2004). The differences in ROUGE-1 and ROUGE-L are statistically significant with $p < 1e{-}10$.

erator network with coverage as our starting point (See et al., 2017; Klein et al., 2017) (see the supplementary material for details). We investigate whether concatenating our proposed features with word embeddings can improve generation performance, as measured by ROUGE scores.

Consistent with results in sequence tagging for word-level echoing prediction, our proposed features can enhance a neural model with copying mechanisms (see Table 6). Specifically, their use leads to statistically significant improvement in ROUGE-1 and ROUGE-L, while slightly hurting the performance in ROUGE-2 (the difference is not statistically significant). We also find that our features can increase the likelihood of copying: an average of 17.59 unique words get copied to the generated explanation with our features, compared to 14.17 unique words without our features. For comparison, target explanations have an average of 34.81 unique words. We emphasize that generating explanations is a very challenging task (evidenced by the low ROUGE scores and examples in the supplementary material), and that fully solving the generation task requires more work.

## 6 Concluding Discussions

In this work, we conduct the first large-scale empirical study of everyday explanations in the context of persuasion. We assemble a novel dataset and formulate a word-level prediction task to understand the selective nature of explanations. Our results suggest that the relation between an OP and PC plays an important role in predicting the echoing of content words, while a word's non-contextual properties matter for stopwords. We show that vanilla LSTMs fail to learn some of the features we develop and that our proposed features can improve the performance in generating explanations using pointer networks. We also demonstrate the important role of nouns in shaping explanations.

Although our approach strongly outperforms random baselines, the relatively low F1 scores indicate that predicting which word is echoed in explanations is a very challenging task. It follows that we are only able to derive a limited understanding of how people choose to echo words in explanations. The extent to which explanation construction is fundamentally random (Nisbett and Wilson, 1977), or whether there exist other unidentified patterns, is of course an open question. We hope that our study and the resources that we release encourage further work in understanding the pragmatics of explanations.

There are many promising research directions for future work in advancing the computational understanding of explanations. First, although `/r/ChangeMyView` has the useful property that its explanations are closely connected to its explananda, it is important to further investigate the extent to which our findings generalize beyond `/r/ChangeMyView` and Reddit and establish universal properties of explanations. Second, it is important to connect the words in explanations that we investigate here to the structure of explanations in pyschology (Lombrozo, 2006). Third, in addition to understanding what goes into an explanation, we need to understand what makes an explanation effective. A better understanding of explanations not only helps develop explainable AI, but also informs the process of collecting explanations that machine learning systems learn from (Hancock et al., 2018; Rajani et al., 2019; Camburu et al., 2018).

## Acknowledgments

We thank Kimberley Buchan, anonymous reviewers, and members of the NLP+CSS research group at CU Boulder for their insightful comments and discussions; Jason Baumgartner for sharing the dataset that enabled this research.

## References


Colin R Blyth. 1972. On Simpson's paradox and the sure-thing principle. *Journal of the American Statistical Association*, 67(338):364–366.

Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. 2018. e-SNLI: Natural language inference with natural language explanations. In *Proceedings of NeurIPS*.

Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A scalable tree boosting system. In *Proceedings of KDD*.

CMV moderators. 2019a. CMV media coverage. https://changemyview.net/subreddit/#media-coverage. [Online; accessed 27-Apr-2019].

CMV moderators. 2019b. The Delta System. https://www.reddit.com/r/changemyview/wiki/deltasystem. [Online; accessed 27-Apr-2019].

Cristian Danescu-Niculescu-Mizil, Michael Gamon, and Susan Dumais. 2011. Mark my words!: Linguistic style accommodation in social media. In *Proceedings of WWW*.

Cristian Danescu-Niculescu-Mizil, Lillian Lee, Bo Pang, and Jon Kleinberg. Echoes of power: Language effects and power differences in social interaction. In *Proceedings of WWW*.

Finale Doshi-Velez and Been Kim. 2017. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*.

Esin Durmus and Claire Cardie. 2018. Exploring the role of prior beliefs for argument persuasion. In *Proceedings of NAACL*.

Howard Giles and Tania Ogay. 2007. Communication accommodation theory. *Explaining communication: Contemporary theories and exemplars*, pages 293–310.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating Copying Mechanism in Sequence-to-Sequence Learning. In *Proceedings of ACL*.

Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. 2019. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, 51(5):93.

Ivan Habernal and Iryna Gurevych. 2016. What makes a convincing argument? Empirical analysis and detecting attributes of convincingness in web argumentation. In *Proceedings of EMNLP*.

Braden Hancock, Paroma Varma, Stephanie Wang, Martin Bringmann, Percy Liang, and Christopher R. 2018. Training Classifiers with Natural Language Explanations. In *Proceedings of ACL*.

Virgina Heffernan. 2018. Our best hope for civil discourse online is on ... Reddit. *Wired*.

Carl G Hempel and Paul Oppenheim. 1948. Studies in the logic of explanation. *Philosophy of science*, 15(2):135–175.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing.

Benjamin D Horne, Dorit Nevo, John O'Donovan, Jin-Hee Cho, and Sibel Adali. 2019. Rating reliability and bias in news articles: Does ai assistance help everyone? In *Proceedings of ICWSM*.

Frank C Keil. 2006. Explanation and understanding. *Annu. Rev. Psychol.*, 57:227–254.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL*.

Vivian Lai and Chenhao Tan. 2019. On human predictions with explanations and predictions of machine learning models: A case study on deception detection. In *Proceedings of FAT**.

Jure Leskovec, Lars Backstrom, and Jon Kleinberg. 2009. Meme-tracking and the dynamics of the news cycle. In *Proceedings of KDD*.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Wang Ling, Phil Blunsom, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, Fumin Wang, and Andrew Senior. 2016. Latent predictor networks for code generation. In *Proceedings of ACL*, pages 599–609, Berlin, Germany. Association for Computational Linguistics.

Marco Lippi and Paolo Torroni. 2016. Argumentation mining: State of the art and emerging trends. *ACM Transactions on Internet Technology (TOIT)*, 16(2):10.

Zachary C Lipton. 2016. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490*.

Tania Lombrozo. 2006. The structure and function of explanations. *Trends in cognitive sciences*, 10(10):464–470.

Edward Loper and Steven Bird. 2002. NLTK: the natural language toolkit. *arXiv preprint cs/0205028*.

Stephanie M Lukin, Pranav Anand, Marilyn Walker, and Steve Whittaker. 2017. Argument strength is in the eye of the beholder: Audience effects in persuasion. In *Proceedings of EACL*.

Mitchell P. Marcus, Beatrice Santorini, and Mary A. Marcinkiewicz. 1994. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19:313–330.

Tim Miller. 2018. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*.

Richard E Nisbett and Timothy D Wilson. 1977. Telling more than we can know: Verbal reports on mental processes. *Psychological review*, 84(3):231.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of EMNLP*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Barbara Plank, Anders Søgaard, and Goldberg Yoav. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of ACL (short papers)*.

Martin F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(2):130–137.

Sindhu Raghavan, Adriana Kovashka, and Raymond Mooney. 2010. Authorship attribution using probabilistic context-free grammars. In *Proceedings of ACL (short papers)*, pages 38–42.

Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Explain yourself! leveraging language models for commonsense reasoning. In *Proceedings of ACL*.

Allan G Reynolds and Paul W Flagg. 1976. Recognition memory for elements of sentences. *Memory & Cognition*, 4(4):422–432.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why should I trust you?: Explaining the predictions of any classifier. In *Proceedings of KDD*.

Wesley C Salmon. 2006. *Four decades of scientific explanation*. University of Pittsburgh press.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get To The Point: Summarization with Pointer-Generator Networks. In *Proceedings of ACL*.

Matthew P Simmons, Lada A Adamic, and Eytan Adar. 2011. Memes online: Extracted, subtracted, injected, and recollected. In *Proceedings of ICWSM*.

Swapna Somasundaran and Janyce Wiebe. 2009. Recognizing stances in online debates. In *Proceedings of ACL*.

Chenhao Tan, Vlad Niculae, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2016. Winning Arguments: Interaction Dynamics and Persuasion Strategies in Good-faith Online Discussions. In *Proceedings of WWW*, WWW '16, pages 613–624, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee. Event-place: Montral, Qubec, Canada.

Chenhao Tan, Hao Peng, and Noah A. Smith. 2018. You are no Jack Kennedy: On media selection of highlights from presidential debates. In *Proceedings of WWW*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Proceedings of NeurIPS*, pages 2692–2700.

Marilyn A Walker, Pranav Anand, Robert Abbott, and Ricky Grant. 2012. Stance classification using dialogic properties of persuasion. In *Proceedings of NAACL*.

Lu Wang, Nick Beauchamp, Sarah Shugars, and Kechen Qin. 2017. Winning on the merits: The joint effects of content and style on debate outcomes. *Transactions of the Association for Computational Linguistics*.

Robert A. Wilson and Frank Keil. 1998. The Shadows and Shallows of Explanation. *Minds and Machines*, 8(1):137–159.

James Woodward. 2005. *Making things happen: A theory of causal explanation*. Oxford university press.

Justine Zhang, Ravi Kumar, Sujith Ravi, and Cristian Danescu-Niculescu-Mizil. 2016. Conversational flow in Oxford-style debates. In *Proceedings of NAACL*.

# Modeling Frames in Argumentation

**Yamen Ajjour**
Bauhaus-Universität Weimar
Faculty of Media, Webis Group
yamen.ajjour@uni-weimar.de

**Milad Alshomary**
Paderborn University
Department of Computer Science
milad.alshomary@upb.de

**Henning Wachsmuth**
Paderborn University
Department of Computer Science
henningw@upb.de

**Benno Stein**
Bauhaus-Universität Weimar
Faculty of Media, Webis Group
benno.stein@uni-weimar.de

## Abstract

In argumentation, framing is used to emphasize a specific aspect of a controversial topic while concealing others. When discussing the legalization of drugs, for instance, its economical aspect may be emphasized. In general, we call a set of arguments that focus on the same aspect a *frame*. An argumentative text has to serve the "right" frame(s) to convince the audience to adopt the author's stance (e.g., being pro or con legalizing drugs). More specifically, an author has to choose frames that fit the audience's interests and cultural background. This paper introduces *frame identification*, which is the task of splitting a set of arguments into a set of non-overlapping frames. We present a fully unsupervised approach to this task, which first removes topical information from the arguments and then identifies frames using clustering. For evaluation purposes, we provide a corpus with 12 326 debate-portal arguments, organized along the frames of the debates' topics. On this corpus, our approach outperforms different strong baselines, achieving an $F_1$-score of 0.28.

## 1 Introduction

Different interests, cultural backgrounds, and socializations make people disagree on taking a certain course of action. A *debate* is a means for the involved parties to resolve their disagreement. A debate is characterized by a *topic*, e.g., "Should one legalize abortion?". Upon the topic, the disagreeing parties have a pro or con *stance* respectively, say, "Abortion should be legalized" or "Abortion should not be legalized". A stance is supported by *arguments*; a pro argument could be "Abortion is good for a free society because it gives women their basic right of controlling their bodies." while a con argument could be "Abortion is against human rights because it is a systematic murder of innocent life." One part of an argument (here: the part before the word "because") is called a *conclusion*, the other part consists of one or more *premises*. A debate can be considered as set of pro and con arguments.

Typically, numerous arguments exist for a topic. The parties involved in a debate have to choose among the arguments, thereby *framing* the topic by emphasizing a specific aspect while concealing others. We call the set of the arguments which share an aspect a *frame*. More specifically, a frame $F$ is subset of a set of arguments $A$, $F \subseteq A$. Likewise, a set of frames, $\{F_1, \ldots, F_k\}$ covers a set of arguments iff. $A \subseteq \bigcup_j^k F_j$.

For instance, the following arguments target different topics but concentrate on the same frame, namely, the *economical* aspect.

**Argument 1** *"I support legalization of marijuana since it can be taxed for revenue gain."*
**Topic:** Marijuana

**Argument 2** *"Legalizing prostitution would increase government revenue. A tax on the fee charged by a prostitute, and the imposition of income tax on the earnings of prostitutes would generate revenue."*
**Topic:** Prostitution

Framing is a decisive step in the construction of arguments, which affects the outcome of a debate (Eemeren and Houtlosser, 1999). To achieve persuasion, an author of an argumentative text should choose frames that resonate with the target audience. As a simple example, an argument appealing to Christianity might not be acceptable to an atheist. Knowing the arguments for a topic along with their frames enables authors to choose those arguments that best address their audience.

The constellation of pro and con arguments for a topic is an urgent need for authors of argumentative texts. Argument search is a new research area that aims at assessing users in forming an opinion and debating. Current approaches use clas-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2922–2932,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

sifiers to mine arguments for a given topic from a relevant document (Levy et al., 2014; Stab et al., 2018; Wachsmuth et al., 2017a). The mentioned approaches ignore identifying the frames of arguments during mining and retrieval, this way omitting extremely valuable information.

The paper in hand starts by reviewing related work to framing (Section 2). In Section 3, we introduce the first argument dataset that has been annotated with frames and topics, and we provide statistical insights into the dataset. Section 4 presents a new unsupervised approach to identify frames in a set of arguments. Our approach first removes topical features from the arguments and then clusters the arguments into frames. In Section 5, we describe the experiments that we conducted to evaluate our approach. We apply the approach to all arguments from *debatepedia.org* and evaluate the returned frames against the ground-truth frames in our dataset. Section 6 shows and discusses the effectiveness of our approach. In Section 7, we analyze the errors made by the approach in the experiments.

The main contributions of this paper are:

- A formal view of frames in argumentation.
- An unsupervised approach to identifying frames in a set of argumentative texts.
- An argument framing dataset with 465 topics, 1 623 frames, and 12 326 arguments.

We freely provide the complete dataset to the research community.[1]

## 2 Related Work

Research on framing is scattered across different fields such as media, social, and cognitive studies. Entman (1993) was the first to introduce a formal definition of framing as a way to select and make specific aspects of a topic salient. Subsequent research on framing is concentrated on the effect of using frames in news on a specific audience. One of the open questions is whether frames are topic-specific or generic concepts, or both. Vreese (2005) studied framing in news articles and considered frames to be both of the two. Johnson et al. (2017) and Card et al. (2015), on the other hand, defined frames to be independent of the topic and investigated their usage across different topics.

Recently, framing caught some attention in the NLP community. Different computational models have been developed for modeling frames in natural language text. Tsur et al. (2015) used topic models on statements released by congress members of the two major parties in the US, Republicans and Democrats. The learned topics were then aggregated into clusters, such as health and economy, and interpreted as being generic frames. On this basis, the authors studied the frequency of the frames in the released statements for the two parties as well as their distribution over time. A related work was conducted by Menini et al. (2017) to model frames in political manifestos released by the parties (texts declaring a stance) as clusters of key phrases. The developed method was shown to outperform standard topic models in capturing frames.

Card et al. (2015) annotated around 16k news articles on three topics (same-sex marriage, immigration, and smoking), along with a list of 15 generic frames. While the annotations had to cover continuous spans of text, their granularity was left unspecified. The inter-annotator agreement on frames for the different frames ranged between 0.08 and 0.23 in terms of Krippendorff's $\alpha$. By comparison, our dataset covers both generic and topic-specific frames and are annotated on the argument level. Naderi and Hirst (2017) extended this line of work by training a neural network to classify the frames in the constructed corpus. The authors modeled frames on the sentence level and reached an accuracy of 53.7% in multi-class classification and 89.3% for one-against-others classification. Using the same corpus, (Field et al., 2018) created a lexicon for each one of the 15 frames and analyzed which frames are used mainly to talk about the United States in Russian news.

A related line of research is the mining of arguments from natural language text (Al-Khatib et al., 2016). Most approaches uses supervised classifiers to extract the structure of arguments (*conclusion* and *premise*) (Stab and Gurevych, 2014). (Lawerence and Reed, 2017) showed that topic models helps identifying the relevance of a premise to a conclusion when they are trained on topically relevant documents. The stance of the mined arguments is classified as pro or con towards a given topic (Somasundaran and Wiebe, 2010; Bar-Haim et al., 2017). The arguments are then used for applications such as argument search (Wachsmuth et al.,

[1] Argument framing dataset: https://webis.de/data/webis-argument-framing-19.html or https://doi.org/10.5281/zenodo.3373355

2017a; Levy et al., 2018; Stab et al., 2018) with the goal of retrieving relevant arguments for an input claim. Use cases for argument search include writing and debating support. In comparison to user queries in conventional search that can often be satisfied by one or a few retrieved documents, these use cases require a broader consideration of the retrieved arguments. Hence, the user of an argument search engine will often investigate both stances and multiple frames on a given topic. While several studies tackled the task of ranking arguments according to their quality (Habernal and Gurevych, 2016; Wachsmuth et al., 2017b), how to aggregate arguments into frames is largely unstudied.

The relation between arguments and frames was introduced briefly in some works (Boydstun et al., 2013; Gabrielsen et al., 2011). Still, recent research on computational argumentation largely ignores frames, and a model for aggregating arguments into frames is still missing. Naderi (2016) considered a frame to be an argument and classified sentences in parliamentary speeches into one of seven frames. (Reimers et al., 2019) created a dataset of argument pairs that are labeled according to their similarity. Based on the dataset, they introduced the task of argument clustering which aims at classifying an argument pair with the same topic into similar or dissimilar. The main difference to this work is that no explicit aspect are assigned to the arguments during annotation.

## 3 Data

Debate portals are websites where people debate or collect arguments for or against controversial topics. Some debate portals are dialogical, such as *debate.org*, allowing two opponents to debate one topic in rounds. Other debate portals such as *debatepedia.org* are wiki-like where arguments are listed according to their stance on the topic. Debate portals keep a canonical structure of the arguments considered for each topic (usually a conclusion and a premise). The structure and the wide topic coverage offered by debate portals has made them a suitable resource for research on computational argumentation (Cabrio and Villata; Al-Khatib et al., 2016; Wachsmuth et al., 2017a).

### 3.1 Argument Frames from Debatepedia.org

For the given work, we crawled all arguments from debatepedia.org in order to construct a dataset for the evaluation of frame identification. Debatepedia.org organizes a debate into sets of arguments that address a topical aspect of the debate. A label that describes the topical aspect is attached to some of the sets, such as *economics*. An argument on debatepedia.org is listed as a conclusion on the topic along with a premise that supports it.

| # Topics | # Frames | # Merged Frames | # Arguments |
|---|---|---|---|
| 465 | 1 645 | 1 623 | 12 326 |

Table 1: Counts of topics, frames, merged frames, and arguments in the webis-argument-framing-19 dataset.

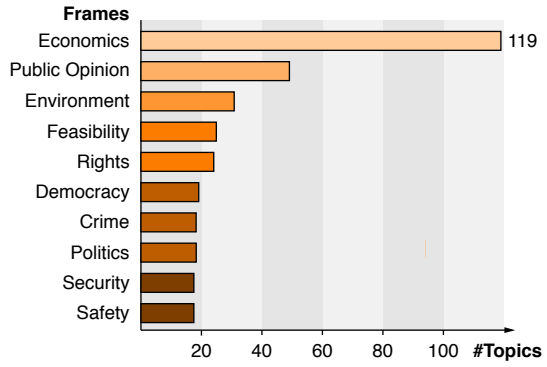


Figure 1: The number of topics in which each of the 10 most frequent frame labels in our dataset occurs.

Arguments which are not labeled might introduce noise to the dataset, since the true knowledge regarding their frames is unavailable. To exclude possible noise in the planned experiments, we filtered out all arguments without labels (about 1800). Next, we analyzed the extracted labels and found that some labels have a similar meaning but are worded differently. In particular, we noticed the presence of the following cases:

1. Labels with hierarchical relations, such as *business* and *US business*.
2. Opposite labels, such as *health* and *unhealthy*, or, *protecting smokers* and *protecting non-smokers*.
3. Labels that are equal when being lemmatized, such as *economics* and *economic*, *democratizing* and *democratic*, etc.

Labels with the same lemmas are likely to carry the same meaning, which is why we merged them into the same label. The count of such merged label pairs was 22, each containing 42 arguments on average. Since the labels in the first and second cases might constitute different frames in some context, we kept them as they are.

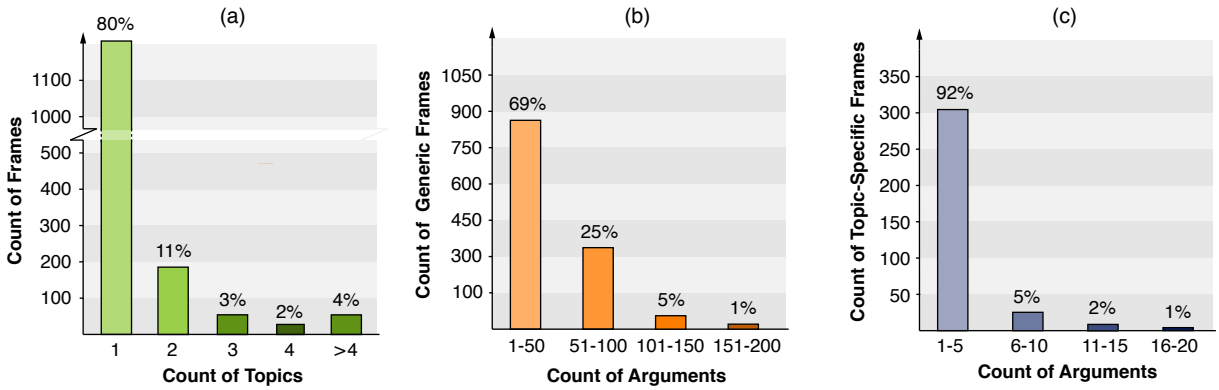


Figure 2: General statistics of frames from debatepedia.org in the webis-argument-framing-19 dataset. (a): Histogram of frames over the count of topics in which they are used. (b): Histogram of generic frames over the count of arguments they contain. (c): Histogram of topic-specific frames over the count of arguments they contain.

### 3.2 Webis-Argument-Framing-19 Dataset

Table 1 shows general statistics of the final dataset after crawling and preprocessing, called *webis-argument-framing-19*. As visualized in Figure 1, the ten most frequent labels in our dataset are: *economics*, *public opinion*, *environment*, *feasibility*, *rights*, *democracy*, *crime*, *politics*, *security* and, *safety*. These labels largely overlap with those introduced by Card et al. (2015); hence, we considered each set of arguments to be a frame.

The count of topics in which a frame occurs indicates whether a frame is generic or topic-specific. To distinguish between these two types of frames, we grouped all frame labels in our dataset according to how many topics they occur in. Figure 2 (a) shows a histogram of the frames in our dataset over the count of topics in which they are used. As depicted, 80% (1293) of the frames are used in one topic and, hence, we labeled them as *topic-specific*. Frames that are used in more than one topic add up to 20% (330) frames and are labeled as *generic*. Generic frames in the dataset cover 7052 arguments while topic specific frames cover 5274 arguments. Figure 2 (b) and (c) show a histogram of generic and of topic-specific frames over the count of arguments they contain respectively. The histograms reveal that generic frames cover an order of magnitude more as many arguments as topic-specific frames.

## 4 Approach

In this section, we introduce our unsupervised approach to modeling frames formally. We assume frames to be exclusive and non-overlapping. Given a set of arguments $A = \{a_1, a_2, \ldots, a_n\}$, our goal is to find a set of frames, which constitutes a cover of $A$. A cover of $A$ is a set of sets $\{F_1, F_2, \ldots, F_k\}$ whose union contains $A$, i.e., $A \subseteq \bigcup_j^k F_j$. Table 2 lists the symbols used in this section along with their meaning.

The main idea of our approach is to first remove topical features from arguments and then to cluster the arguments into frames. Following known topic modeling approaches, we represent the content of an argument $a$ as a bag of words and propose two models to find topic-specific words. Both models utilize the frequency of the words in an argument and the argument's structure. The structure of $a$ is represented by its conclusion $c$ and its premise(s) $p$. Our approach includes three main steps:

(a) **Topic clustering.** Cluster the arguments in $A$ into $m$ topics $\bar{\mathcal{A}} = \{\bar{A}_0, \bar{A}_1, \ldots, \bar{A}_m\}$.

(b) **Topic removal.** Given the produced clusters, develop an extraction model $E$ that extracts topical features from an argument $a_i$ and its cluster. $E$ is applied to each $\bar{A}_j \in \bar{\mathcal{A}}$ to remove topic-specific features. As a result, we obtain "topic-free" arguments $a'_i = a_i - E(a, \bar{A}_j)$. We denote the set of all "topic-free" arguments with $A'$ where $A' = \{a'_1, a'_2, \ldots, a'_n\}$.

(c) **Frame clustering.** Cluster the arguments $A'$ into $k$ clusters, each respresenting one frame.

Figure 3 sketches the general idea of the three steps of our approach. We detail our concrete realization of each step in the following.

### 4.1 Topic Clustering

To cluster the given set of arguments into topics, we first map each argument into a vector space that

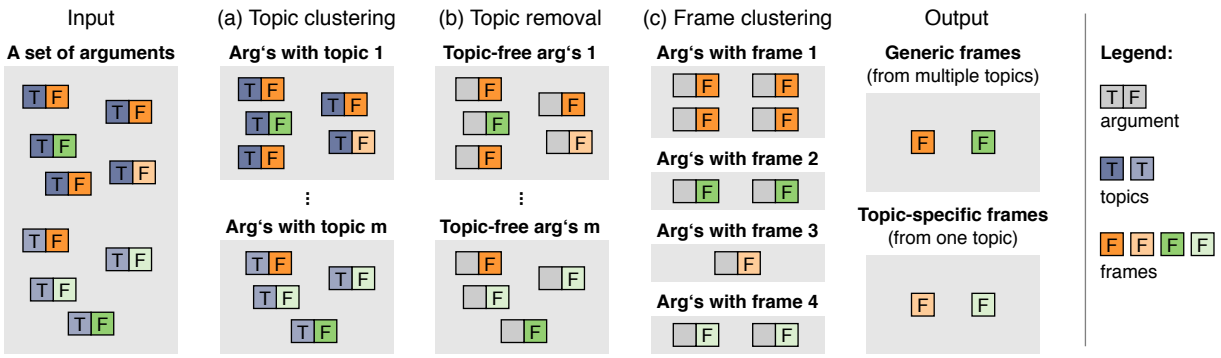


Figure 3: Sketch of the proposed unsupervised approach to argument frame identification. An argument is modeled as a topic and a frame. The input is a sets of arguments. The output is representations of two types of found frames.

| Symbol | Meaning |
|---|---|
| $a$ | An argument |
| $c$ | The conclusion of an argument |
| $A$ | A set of arguments |
| $\bar{A}$ | A set of arguments on the same topic |
| $\mathcal{A}$ | A set of sets of arguments |
| $F$ | A frame |
| $v$ | A word |
| $V$ | A vocabulary |
| $E$ | A topic extraction model |

Table 2: Notation of the symbols used in the approach

represents its semantics. We use $k$-means (Hartigan and Wong, 1979) with Euclidean distance as a clustering algorithm. For semantic spaces, we consider two alternatives: *Term Frequency-Inverse Document Frequency* (TF-IDF), and *Latent Semantic Analysis* (LSA).

**TF-IDF** TF-IDF defines a vector space whose dimensions are words in the dataset. An argument is mapped to this space according to the frequency of each of its words, normalized by the word's frequency in all considered arguments. TF-IDF is a sparse vector space since all words in a set of arguments are considered. To reduce sparsity, we construct a vocabulary $V$ which comprises the 5000 most frequent words in the arguments after stopword removal. Words that occur in more than half of the arguments are ignored as well. The main reason for reducing the vocabulary is to increase the computational efficiency of the approach.

**LSA** Latent Semantic analysis (Deerwester et al., 1990) infers from a term-document frequency matrix a linear transformation that project documents into a topic space. We construct two different semantic spaces using LSA. The first, simply called *LSA*, considers each argument to be a document. The second, *LSA Debate*, considers a whole debate to be a document. Since LSA Debate works on the debate level, it can better capture the topic context of an argument. The reason is that arguments capture the topic differently and may have few words in common. Using all arguments in a debate ensures a broader context of the topic. To compare both LSA models systematically, we use the same number of dimension for both models: 1000.

### 4.2 Topic Removal

The goal of this step is to remove topic-specific features in the topic clusters $\bar{\mathcal{A}} = \{\bar{A}_0, \bar{A}_2, \ldots, \bar{A}_m\}$. To achieve this goal, we develop two models to extract topic-specific features, $E_1^q$ and and $E_2$. $E_1^q$ utilizes the content of the arguments in one cluster, whereas $E_2$ utilizes the argument structure, i.e., conclusion and premise information.

$E_1^q$ utilizes the term-frequency inverse document frequency measure TF-IDF for every word $v$ in each cluster. We calculate $idf$ as follows:

$$idf(v) = \frac{\left|\bar{\mathcal{A}}\right|}{\left|\bar{A}_j \in \bar{\mathcal{A}} : v \in \bar{A}_j\right|}$$

Then, $E_1^q(a)$ returns those words that best discriminates a specific topic as follows based on a threshold $q$, which can be understood as the aggressiveness of the model:[2]

$$E_1^q(a, \bar{A}_j) = \{v \in \bar{A}_j : tf.idf(v) > q\}$$

$E_2$ utilizes the structure of an argument on a local level. The hypothesis here is that the conclusion of an argument contains more words that target the topic than its premise. Hence, we remove the conclusion in an argument. Formally:

$$E_2(a, \bar{A}_j) = \{v \in c\}$$

[2]In our experiments, we chose the threshold $q$ empirically.

### 4.3 Frame clustering

This step aims at grouping arguments that share a common aspect after removing topical features. For clustering, we use $k$-means again and experiment with different values of $k$. Below, we choose $k$ based on an experiment that evaluates the output of the cluster against the ground-truth. We also use Euclidean distance to estimate the similarity between the arguments in the two semantic spaces.

## 5 Experiments

Based on the dataset we introduced in Section 3, we conduct experiments to evaluate and analyze our approach to modeling frames in argumentation. As discussed above, the approach consists of three steps: topic clustering, topic removal, and frame clustering. We evaluate the three steps and their interaction with each other in different experiments.

**Topic Clustering Experiment** The goal of this experiment is to find the best method to group arguments into topics. The produced clusters for each semantic space are evaluated against the arguments' topics in the ground-truth dataset. An external measure is then used to evaluate the output of the clustering algorithm for each semantic space. In particular, we use *Bcubed $F_1$-score* (Bagga and Baldwin, 1998) to evaluate the effectiveness of our approach in modeling topics in the dataset. Bcubed $F_1$-score rewards only the instance pairs that exist in the output of the clustering algorithm and in the ground-truth together in the same cluster. The reason for choosing Bcubed $F_1$-score is that it is proven to satisfy desired constraints in the output of clustering algorithms (Amigó et al., 2009).

**Topic Removal Experiment** This experiment aims at evaluating our models $E_1^q$ and $E_2$ at removing topical features from the arguments in $\bar{A}$. The evaluation criterion here is the effectiveness drop of the topic clustering algorithm after removing the topical features in $\bar{A}$. We rerun the topic clustering algorithm with the same $k$ after removing the output of both models $E_1^q$ and $E_2$. To have a consistent comparison, we set $k$ to the best count of topics we found in the previous experiment.

**Frame Clustering Experiment** The last experiment evaluates clustering arguments into frames after topic removal. To test our hypothesis that topic removal benefits frame identification, we also cluster arguments in the same semantic space without topic removal. For both semantic spaces, we conduct three experiments: main experiment, generic frame experiment, and topic-specific frame experiment. In the topic-specific and generic frame experiment, we use the frames in our dataset that are labeled as topic specific and generic frame separately. In the main experiment, we test our approach on the whole dataset without distinguishing the type of frames. The different experiments should show us the performance of our approach at identifying generic and topic-specific frames. Similar to topic clustering, we use Bcubed $F_1$-score (Bagga and Baldwin, 1998) to evaluate the frame clustering algorithm in the three experiment. Since our dataset contains 1623 frames, we evaluate the output of the clustering algorithm for each $k \in \{1000, \ldots, 1600\}$.

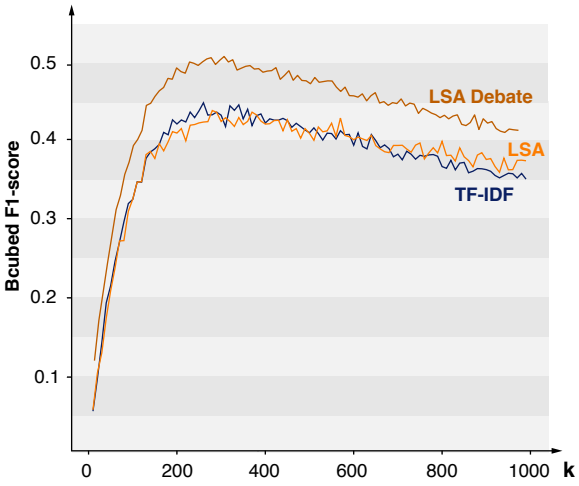


Figure 4: Bcubed $F_1$-score of the topic clustering algorithms for the semantic spaces TF-IDF, LSA and LSA Debate for each $k$.

## 6 Results and Discussion

In the following, we report on the results of the three experiments explained above separately. At the end, we discuss the findings of the experiments and draw final conclusions on the performance of our approach at identifying frames.

### 6.1 Topic Clustering

Figure 4 shows the effectiveness of topic clustering using the different semantic spaces. We visualize for each $k$ the Bcubed $F_1$-score of the clustering algorithms for the three semantic spaces. As shown, TF-IDF and LSA perform similarly for all $k$. The clustering algorithm performs better using the semantic space LSA Debate than LSA and

| Semantic Space | # Topics | $F_1$ |
|---|---|---|
| LSA Debate | 310 | **0.52** |
| TF-IDF | 260 | 0.45 |
| LSA | 280 | 0.44 |

Table 3: Bcubed $F_1$-score of the topic clustering algorithm for each semantic space and the corresponding count of topics found.

TF-IDF. This shows the importance of considering the context of an argument for modeling their topics. All the three depicted plots, however, show a clear elbow between topic counts 200 and 400. Table 3 shows the highest corresponding $F_1$-score and the count of topic clusters for each semantic space. The best topic clustering achieved by the algorithms comprises 310 clusters. Given its high effectiveness in modeling topics, we decided to proceed with the topic clusters produced by the LSA Debate in the next experiment.

### 6.2 Topic Removal

Table 4 shows the results of the topic removal experiment and frame clustering experiment. For both semantic spaces, the effectiveness of topic clustering algorithm is reported after using the models $E_1^q$ and $E_2$ to remove topic-specific words. To evaluate the topic extraction models, we re-list the effectiveness achieved by the topic clustering algorithm for both spaces. We show the results of $E_1^q$ only for $q = 0.005$ since higher values of $q$ showed similar or lower results in all experiments.

As shown, $E_2$ decreases the effectiveness of topic clustering algorithm to around the half. The model $E_1^{0.005}$ achieves a smaller drop of 0.03-0.04 in the two semantic spaces. Despite its simplicity, $E_2$ is more effective at removing topic-specific features than $E_1^{0.005}$.

### 6.3 Frame Clustering

Table 4 shows the results of the frame clustering algorithm in the experiments: generic, topic-specific, and main. In each experiment, the clustering algorithm is run after using the two topic extraction models to remove topic-specific features and without applying them (baseline). In the main and the generic experiment, using the topic extraction models outperforms not using them in both semantic spaces. In the topic-specific experiment, our approach's effectiveness outperformed the baseline only in LSA space. The comparison between the results in the generic and topic-specific experiments shows that identifying generic frames is harder. The reason can also be the small size of topic-specific frames in the ground-truth. Our approach, however, is only effective at identifying generic frames and fails at outperforming the baseline in the topic-specific experiments. A reason to justify this is that removing topic-specific features negatively affects identifying topic-specific frames.

To better analyze our approach, we plot the achieved Bcubed $F_1$-score for each semantic space and each experiment for different values of $k$. Figure 5 (a,b,c) shows the effectiveness achieved in the three experiments main, topic-specific and generic respectively in TF-IDF space. As shown, both models $E_1^{0.005}$ and $E_2$ start to outperform the baseline at $k = 1200$ in the main experiment. All the approaches converge starting from this value and not much effectiveness is achieved for higher values of $k$. In the generic experiment, both models achieve their first peaks at $k = 800$. The performance of both models oscillates but keeps at the same rate of for larger values of $k$. In the topic-specific experiments, the performance of our approach increases significantly while approaching the value of $k = 400$. The gain achieved by for more clusters decreases slowly for larger values of $k$.

Figure 6 (a,b,c) shows the effectiveness achieved in the three experiments main, topic-specific and generic respectively in LSA space. As depicted in the three figures, the model $E_1^{0.005}$ outperforms $E_2$ in all cases which shows that content-based topic-removal of arguments is more effective than using its structure. In the generic experiment, all models in LSA space shows subpar effectiveness to their counterpart in TF-IDF space and lack clear peaks. In the topic-specific experiment, our approach outperforms the LSA baseline and their counterpart in TF-IDF space. Nevertheless, like in the generic experiment, no clear peak is reached by any model.

### 6.4 Discussion

The results show the merit of removing the topic-specific of an argument for identifying its frame. According to the reported results, our approach is effective at identifying generic frames and does not suit identifying topic-specific frames. An interesting finding is that the premise of an argument carries more information about its frame than the conclusion. This is shown in the higher effectiveness achieved after applying $E_2$ compared to the

| Semantic Space | Topic Removal | | Frame Clustering | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Model | Topic $F_1$ | Generic Frames | | | Topic-specific Frames | | | Frames | | |
| | | | $F_1$ | P | R | $F_1$ | P | R | $F_1$ | P | R |
| TF-IDF | Baseline | **0.45** | 0.19 | 0.25 | 0.15 | **0.48** | 0.53 | 0.44 | 0.26 | 0.27 | 0.25 |
| | $E_1^{0.005}$ | 0.42 | **0.28** | 0.26 | 0.30 | 0.45 | 0.50 | 0.40 | **0.28** | 0.24 | 0.33 |
| | $E^2$ | 0.17 | 0.26 | 0.25 | 0.28 | 0.45 | 0.48 | 0.42 | 0.27 | 0.25 | 0.29 |
| LSA | Baseline | 0.44 | 0.16 | 0.20 | 0.13 | 0.39 | 0.44 | 0.35 | 0.21 | 0.22 | 0.22 |
| | $E_1^{0.005}$ | 0.4 | 0.21 | 0.15 | 0.33 | 0.47 | 0.44 | 0.48 | 0.26 | 0.25 | 0.27 |
| | $E^2$ | 0.25 | 0.2 | 0.18 | 0.22 | 0.46 | 0.41 | 0.50 | 0.24 | 0.24 | 0.24 |

Table 4: Best bcubed $F_1$-score, precision, and recall for the topic extraction models $E_1^q$, $E_2$ and without topic removal (baseline) in the generic, topic-specific and main frame experiments together with the corresponding bcubed $F_1$-score in topic clustering.

baseline. A justification can be that a conclusion is more likely to carry stance-taking words toward the topic. In general, $E_1^{0.005}$ achieved higher results than $E_2$, which shows that using the content of an argument is more effective than using its structure to model frames. A possible justification for this can be that $E_2$ is more aggressive than needed at removing topic-specific features.

## 7 Error Analysis

We analyze the topic and frame clusters produced by our approach to convey to the reader a sense of its performance. For topic clusters, we focus on the semantic space LSA Debate since our approach performed the best in this semantic space. For Frame clusters, we analyze the output of our approach in the semantic space TF-IDF after applying $E_2^{0.005}$ since our approach performed the best in this semantic space. Our goal is to identify the topics and frames in the dataset which our approach completely confused or correctly identified. To identify these cases, we sort the topics and frames in the ground-truth dataset according to the maximum $F_1$-score achieved in the aforementioned semantic spaces respectively. We manually analyze the topic and frames labels and the count of arguments they comprise and report the most interesting cases.

For topic clustering, examples of topics that our approach correctly identified (with an $F_1$-score of 1) are: *Zoos* and *Compulsory vaccination*. On the other hand, our approach struggled at identifying topics like *Is Pluto a planet?* and *Immunity from prosecution for politicians* (with an $F_1$-score lower than 0.1). A reason for this might be that these topics are too specific and not covered well in our dataset.

In frame clustering, the hardest cases for our approach in TF-IDF space were topic-specific frames that contain few arguments, e.g., *child disability*. Generic frames such as *rights* and *feasibility* were also hard to identify (with an $F_1$-score lower than 0.1). A possible explanation is that these frames can be confused with generic frames like *human rights* and *economics*. Examples of generic frames that were effectively identified are *freedom of speech* and *public health* (with an $F_1$-score equals to 0.5).

## 8 Conclusion

A disagreement between people on a topic can lead to a lively debate where the opposing parties exchange arguments on the topic to enforce their stance. In favor of a particular stance, an argument emphasizes a certain aspect of the topic, thereby hiding other aspects. This phenomenon is called framing and has been introduced in social science (Entman, 1993). Knowing the frame of an argument helps users to choose arguments that better address the audience's cultural background.

Research on framing in natural languages is still lacking. In this paper, we tried to close this gap by introducing a formal view on framing that defines a frame as a set of arguments that share an aspect. Starting from this view, we introduced an approach to remove the topic's features from the arguments and then to cluster them. We operationalized our approach by using two different models. While the first removes an argument's topic features using its content, the second utilizes its structure (conclusion and premise).

For evaluation purposes, we constructed a new dataset that comprises 12 326 arguments grouped along 1 623 frames based on debatepedia.org. The experiments show that we can outperform sensi-

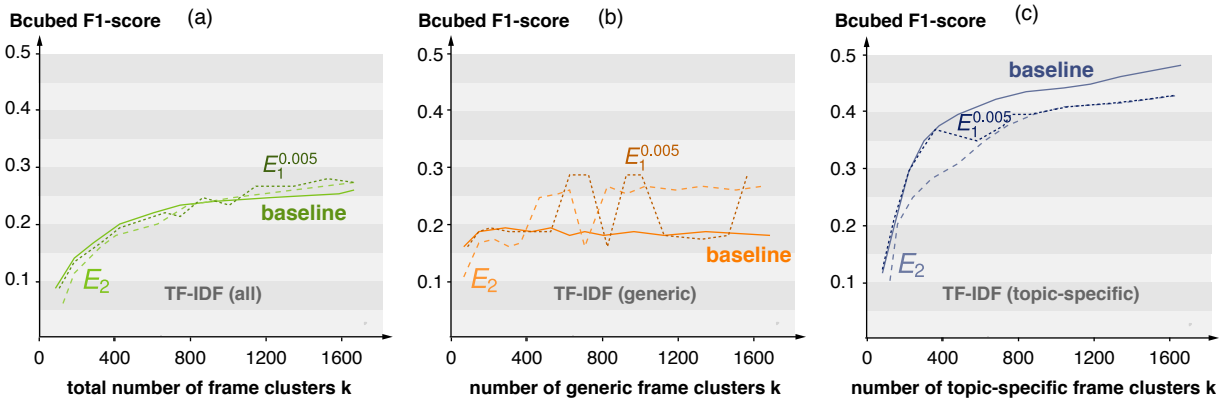


Figure 5: Effectiveness of frame clustering with TF-IDF without topic removal (baseline) and after applying $E_1^{0.005}$ and $E_2$ in (a) the main frame experiment, (b) the generic frame experiment, and (c) the topic-specific experiment.

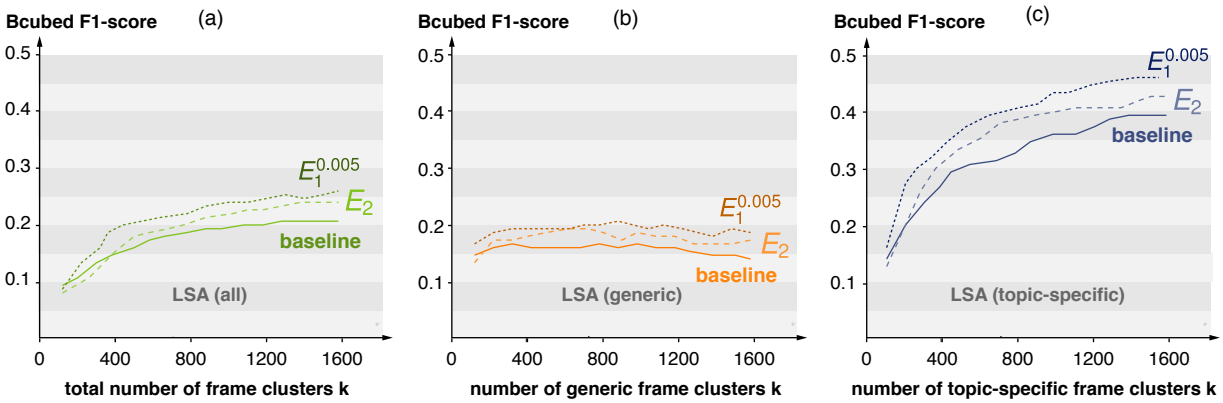


Figure 6: Effectiveness of frame clustering with LSA without topic removal (baseline) and after applying $E_1^{0.005}$ and $E_2$ in (a) the main frame experiment, (b) the generic frame experiment, and (c) the topic-specific experiment.

ble baselines that utilize the same semantic spaces without deleting topic information. We conducted three types of experiments that evaluate our approach in identifying generic (those which are used in multiple topics), topic-specific and both.

Our experiments clearly show the benefit of removing topic's features for identifying an argument's frame. In particular, we find that identifying generic frames benefits from removing topic features, which are actually the hardest case. On the other hand, removing topic features cannot help in identifying topic-specific frames. We also observed that removing an argument's conclusion helps identifying its frame, although it is more likely to carry the stance and the topic of an argument.

Having set a lacking methodology for modeling frames in argumentation, our next step is to develop better approaches for modeling and removing the topic of an argument. Neural networks such as auto-encoders and attention-based models are likely to perform better at modeling frames in argumentation than LSA and TF-IDF. Regarding topic removal, potential research directions will investigate using external knowledge such as Wikipedia to find topic-specific features.

Future work on framing will focus on its application to down-stream argument mining tasks such as analyzing argument quality. Especially interesting is whether specific frames are expected to persuade an audience. A follow-up question will be whether frames in an argumentative text should be delivered in a specific sequence to achieve the persuasion of an audience.

The simplicity of our model allows its application in domains such as news, laws, or student essays. A promising application of our model is argument search since a frame of an argument sheds a light on its acceptability for a specific audience. A clear problem here is how to label a frame given its arguments in order to deliver short labels for the user. We also expect framing to play a major role in generating arguments since a specific frame might resonate with an audience.

## References


Khalid Al-Khatib, Henning Wachsmuth, Matthias Hagen, Jonas Köhler, and Benno Stein. 2016. Cross-Domain Mining of Argumentative Text through Distant Supervision. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 2016)*, pages 1395–1404. Association for Computational Linguistics.

Enrique Amigó, Julio Gonzalo, Javier Artiles, and Fleisa Verdejo. 2009. A comparison of extrinsic clustering evaluation metrics based on formal constraints. 12:461–486.

Amit Bagga and Breck Baldwin. 1998. Algorithms for Scoring Coreference Chains. In *The 1st International Conference on Language Resources and Evaluation Workshop on Linguistics Coreference*, pages 563–566.

Roy Bar-Haim, Indrajit Bhattacharya, Francesco Dinuzzo, Amrita Saha, and Noam Slonim. 2017. Stance Classification of Context-Dependent Claims. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 251–261. Association for Computational Linguistics.

Amber E. Boydstun, Dallas Card, Justin H. Gross, Philip Resnik, and Noah A. Smith. 2013. Identifying media frames and frame dynamics within and across policy issues. In *Proceedings of the Workshop on New Directions in Analyzing Text as Data*.

Elena Cabrio and Serena Villata. Natural Language Arguments: A Combined Approach. In *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI 12)*, pages 205–210, Amsterdam, The Netherlands. IOS Press.

Dallas Card, Amber E Boydstun, Justin H. Gross, Philip Resnik, and Noah A. Smith. 2015. The Media Frames Corpus: Annotations of Frames Across Issues. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL 2015)*, pages 438–444. Association for Computational Linguistics.

Scott Deerwester, Susan T. Dumals, George W. Furnasand, Thomas K Landauer, and Richard Harshman. 1990. Indexing by Latent Semantic Analysis. 41(6):391–407.

Frans Van Eemeren and Peter Houtlosser. 1999. Strategic Manoeuvring in Argumentative Discourse. *Discourse Studies*, 1(4):479–497.

Robert M. Entman. 1993. Framing: Toward clarification of a fractured paradigm. *Journal of Communication*, 43(4):51–58.

Anjalie Field, Doron Kliger, Shuly Wintner, Jinnifer Pan, Dan Jurafsky, and Yulia Tesvetkov. 2018. Framing and Agenda-setting in Russian News: a Computational Analysis of Intricate Political Strategies.

Jonas Gabrielsen, Sine NØrholm Just, and Mette Bengtsson. 2011. Concepts and Contexts – Argumentative Forms of Framing. *Proceedings of the 7th Conference of the International Society for the Study of Argumentation*, pages 533–543.

Ivan Habernal and Iryna Gurevych. 2016. Which argument is more convincing? Analyzing and predicting convincingnessof Web arguments using bidirectional LSTM. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 1589–1599. Association for Computational Linguistics.

J. A. Hartigan and M. A. Wong. 1979. A k-means clustering algorithm. *JSTOR: Applied Statistics*, 28:100–108.

Kirsten Johnson, Di Jin, and Dan Goldwasser. 2017. Modeling of Political Discourse Framing on Twitter. In *Proceedings of the 11th International AAAI Conference on Web and Social Media (ICWSM 2017)*, pages 556–559. Association for the Advancement of Artificial Intelligence.

John Lawerence and Chris Reed. 2017. Mining Argumentative Structure from natural language text using Automatically Generated Premise-Conclusion Topic Models. In *Proceedings of the 4th Workshop on Argumant Mining (ArgMining 2017)*, pages 118–128. Association for Computational Linguistics.

Ran Levy, Yonatan Bilu, Daniel Hershcovich, Ehud Aharoni, and Noam Slonim. 2014. Context Dependent Claim Detection. In *Proceedings of the 25th International Conference on Computational Linguistics: Technical Papers (COLING 2015)*, pages 1489–1500. Dublin City University and Association for Computational Linguistics.

Ran Levy, Ben Boginand Shai Gretz, Ranit Aharonov, and Noam Slonim. 2018. Towards an argumentative content search engine using weak supervision. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING 2018)*, pages 2066–2081.

Stefano Menini, Federico Nanni, Simone Paolo Ponzetto, and Sara Tonelli. 2017. Topic-Based Agreement and Disagreement in US Electoral Manifestos. In *Proceedings of the 2017 Conference of Empirical Methods in Natural Language Processing (EMNLP 2017)*, pages 2938–2944.

Nona Naderi. 2016. Argumentation Mining in Parlimentary Discourse. In *Proceedings of the 11th International Conference of the Ontario Society for the Study of Argumentation*, pages 1–9.

Nona Naderi and Graeme Hirst. 2017. Classifying Frames at the Sentence Level in News Articles. In

*Proceedings of Recent Advances in Natural Language Processing 2017 (RANLP 2017)*, pages 536–542.

Nils Reimers, Benjamin Schiller, Tilman Beck, Johannes Daxenberger, Christian Stab, and Iryna Gurevych. 2019. Classification and Clustering of Arguments with Contextualized Word Embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019)*, pages 567–578. Association for Computational Linguistics.

Swapna Somasundaran and Janyce Wiebe. 2010. Recognizing Stances in Ideological On-Line Debates. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 116–124.

Christian Stab, Johannes Daxenberger, Chris Stahlhut, Tristan Miller, Benjamin Schiller, Christopher Tauchmann, Steffen Eger, and Iryna Gurevych. 2018. ArgumenText: Searching for Arguments in Heterogeneous Sources. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: System Demonstrations (NAACL 2018)*, pages 21–25. Association for Computational Linguistics.

Christian Stab and Iryna Gurevych. 2014. Identifying Argumentative Discourse Structures in Persuasive Essays. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 46–56. Association for Computational Linguistics.

Oren Tsur, Dan Calacci, and David Lazer. 2015. A Frame of Mind: Using Statistical Models for Detection of Framing and Agenda Setting Campaigns. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL 2015)*, pages 1629–1638.

Claes H. De Vreese. 2005. News Framing: Theory and typology. *Information Design Journal and Document Design*, 13(1).

Henning Wachsmuth, Martin Potthast, Khalid Al-Khatib, Yamen Ajjour, Jana Puschmann, Jiani Qu, Jonas Dorsch, Viorel Morari, Janek Bevendorff, and Benno Stein. 2017a. Building an Argument Search Engine for the Web. In *Proceedings of the Fourth Workshop on Argument Mining (ArgMining 2017)*, pages 49–59. Association for Computational Linguistics.

Henning Wachsmuth, Benno Stein, and Yamen Ajjour. 2017b. "PageRank" for Argument Relevance. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017)*, pages 1116–1126. Association for Computational Linguistics.

# AMPERSAND: Argument Mining for PERSuAsive oNline Discussions

**Tuhin Chakrabarty[1], Christopher Hidey[1], Smaranda Muresan[1,2]**
**, Kathleen Mckeown[1] and Alyssa Hwang[1]**
[1]Department of Computer Science, Columbia University
[2]Data Science Institute, Columbia University

{tuhin.chakrabarty, smara, a.hwang}@columbia.edu
{chidey, kathy}@cs.columbia.edu

## Abstract

Argumentation is a type of discourse where speakers try to persuade their audience about the reasonableness of a claim by presenting supportive arguments. Most work in argument mining has focused on modeling arguments in monologues. We propose a computational model for argument mining in online persuasive discussion forums that brings together the micro-level (argument as product) and macro-level (argument as process) models of argumentation. Fundamentally, this approach relies on identifying relations between components of arguments in a discussion thread. Our approach for relation prediction uses contextual information in terms of fine-tuning a pre-trained language model and leveraging discourse relations based on Rhetorical Structure Theory. We additionally propose a candidate selection method to automatically predict what parts of one's argument will be targeted by other participants in the discussion. Our models obtain significant improvements compared to recent state-of-the-art approaches using pointer networks and a pre-trained language model.

## 1 Introduction

Argument mining is a field of corpus-based discourse analysis that involves the automatic identification of argumentative structures in text. Most current studies have focused on monologues or *micro-level* models of argument that aim to identify the structure of a single argument by identifying the argument components (classes such as "claim" and "premise") and relations between them ("support" or "attack") (Somasundaran et al., 2007; Stab and Gurevych, 2014; Swanson et al., 2015; Feng and Hirst, 2011; Habernal and Gurevych, 2017; Peldszus and Stede, 2015). *Macro-level* models (or dialogical models) and rhetorical models which focus on the *process* of argument in a dialogue (Bentahar et al., 2010) have received less attention.

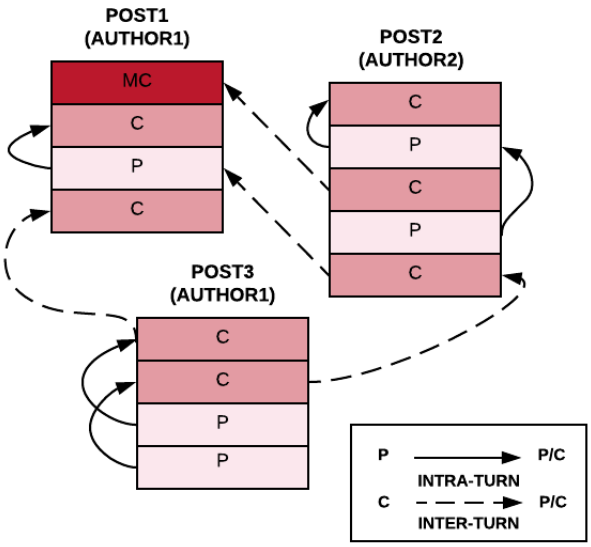


Figure 1: Annotated Discussion Thread (C: Claim, P: Premise, MC: Main Claim)

We propose a novel approach to automatically identify the argument structure in persuasive dialogues that brings together the micro-level and the macro-level models of argumentation. Our approach identifies argument components in a full discussion thread and two kinds of argument relations: *inter-turn relations* (argumentative relations to support or attack the other person's argument) and *intra-turn* relations (to support one's claim). Figure 1 shows a thread structure consisting of multiple posts with argumentative components (main claims, claims or premises) and both intra- and inter-turn relations. We focus here on the relation identification task (i.e, predicting the existence of a relation between two argument components).

To learn to predict relations, we **annotate argumentative relations** in threads from the Change



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2933–2943,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

My View (CMV) subreddit. In addition to the CMV dataset we also **introduce two novel distantly-labeled data-sets** that incorporate the macro- and micro-level context (further described in Section 3). We also **propose a new transfer learning approach** to fine-tune a pre-trained language model (Devlin et al., 2019) on the distant-labeled data (Section 4) and demonstrate improvement on both argument component classification and relation prediction (Section 5). We further show that **using discourse relations** based on Rhetorical Structure Theory (Mann and Thompson, 1988) improves the results for relation identification both for inter-turn and intra-turn relations. Overall, our approach for argument mining **obtains statistically significant improvement** over a state-of-the-art model based on pointer networks (Morio and Fujita, 2018) and a strong baseline using a pre-trained language model (Devlin et al., 2019). We make our data,[1] code, and trained models publicly available.[2]

## 2 Related Work

**Argument mining on monologues** Most prior work in argument mining (AM) has focused on monologues or essays. Stab and Gurevych (2017) and Persing and Ng (2016) used pipeline approaches for AM, combining parts of the pipeline using integer linear programming (ILP) to classify argumentative components. Stab and Gurevych (2017) represented relations of argument components in essays as tree structures. Both Stab and Gurevych (2017) and Persing and Ng (2016) propose joint inference models using ILP to detect argumentative relations. They however rely on handcrafted (structural, lexical, indicator, and discourse) features. We on the other hand use a transfer learning approach for argument mining in discussion forums.

Recent work has examined neural models for argument mining. Potash et al. (2017) use pointer networks (Vinyals et al., 2015) to predict both argumentative components and relations in a joint model. For our work, we focus primarily on relation prediction but conduct experiments with predicting argumentative components in a pipeline. A full end-to-end neural sequence tagging model was developed by Eger et al. (2017) that predicts argumentative discourse units (ADUs), components, and relations at the token level. In contrast, we assume we have ADUs and predict components and relations at that level.

**Argument mining on discussion forums (online discussion)** Most computational work related to argumentation as a process has focused on the detection of agreement and disagreement in online interactions (Abbott et al., 2011; Sridhar et al., 2015; Rosenthal and McKeown, 2015; Walker et al., 2012). However, these approaches do not identify the argument components that the (dis)agreement has scope over (i.e., what has been targeted by a disagreement or agreement move). Also in these approaches, researchers predict the type of relation (e.g. agreement) given that a relationship exists. On the contrary, we predict both argument components as well as the existence of a relation between them. Boltužić and Šnajder (2014) and Murakami and Raymond (2010) address relations between complete arguments but without the micro-structure of arguments as in Stab and Gurevych (2017). Ghosh et al. (2014) introduce a scheme to annotate inter-turn relations between two posts as "target-callout", and intra-turn relations as "stance-rationale". However, their empirical study is reduced to predicting the type of inter-turn relations as agree/disagree/other. Our computational model on the other hand handles both macro- and micro- level structures of arguments (argument components and relations).

Budzynska et al. (2014) focused on building foundations for extracting argument structure from dialogical exchanges (radio-debates) in which that structure may be implicit in the dynamics of the dialogue itself. By combining recent advances in theoretical understanding of inferential anchors in dialogue with grammatical techniques for automatic recognition of pragmatic features, they produced results for illocutionary structure parsing which are comparable with existing techniques acting at a similar level such as rhetorical structure parsing. Furthermore, Visser et al. (2019) presented US2016, the largest publicly available set of corpora of annotated dialogical argumentation. Their annotation covers argumentative relations, dialogue acts and pragmatic features.

Although Niculae et al. (2017) tried relation prediction between arguments in user comments on web discussion forums using structured SVM

[1] https://github.com/chridey/change-my-view-modes
[2] https://github.com/tuhinjubcse/AMPERSAND-EMNLP2019

and RNN the work closest to our task is that of Morio and Fujita (2018). They propose a pointer network model that takes into consideration the constraints of thread structure. Their model discriminates between types of arguments (e.g., claims or premises) and both intra-turn and inter-turn relations, simultaneously. Their dataset, which is not publically available, is three times larger than ours, so instead we focus on transfer learning approaches that take advantage of discourse and dialogue context and use their model as our baseline.

## 3 Data

### 3.1 Labeled Persuasive Forum Data

To learn to predict relations, we use a corpus of 78 threads from the CMV subreddit annotated with argumentative components (Hidey et al., 2017). The authors annotate *claims* ("propositions that express the speakers stance on a certain matter"), and *premises* ("propositions that express a justification provided by the speaker in support of a claim"). In this data, the *main claim*, or the central position of the original poster, is always the title of the original post.

We expand this corpus by annotating the argumentative relations among these propositions (both inter-turn and intra-turn) and extend the corpus by annotating additional argument components (using the guidelines of the authors) for a total of 112 threads. It is to be noted that the claims, premises, and relations have been annotated jointly. Our annotation results on the argumentative components for the additional threads were similar to Hidey et al. (2017) ( IAA using Kripendorff alpha is 0.63 for claims and 0.65 for premises). The annotation guidelines for relations are available in the aforementioned Github repository.

**Intra-turn Argumentative Relations** As in previous work (Morio and Fujita, 2018), we restrict intra-turn relations to be between a premise and another claim or premise, where the premise either supports or attacks the claim or other premise. Evidence in the form of a premise is either support or attack. Consider the following example:

> [Living and studying overseas is an irreplaceable experience.]$_{0:\text{CLAIM}}$ [One will struggle with loneliness]$_{1:\text{PREMISE:ATTACK:0}}$ [but those difficulties will turn into valuable experiences.]$_{2:\text{PREMISE:ATTACK:1}}$ [Moreover, one will learn to live without depending on anyone.]$_{3:\text{PREMISE:SUPPORT:0}}$

The example illustrates that the argumentative component at index 0 is a claim, and is followed by attacking one's own claim, and in turn attacking that premise (an example of the rhetorical move known as prolepsis or prebuttal). They conclude the argument by providing supporting evidence for their initial claim.

**Inter-turn Argumentative Relations** Inter-turn relations connect the arguments of two participants in the discussion (agreement or disagreement). The argumentative components involved in inter-turn relations are claims, as the nature of dialogic argumentation is a difference in stance. An *agreement* relation expresses an agreement or positive evaluation of another user's claim whereas a *disagreement/attack* relation expresses disagreement. We further divide disagreement/attack relations into *rebuttal* and *undercutter* types to distinguish when a claim directly challenges the truth of the other claim or challenges the reasoning connecting the other claim to the premise that supports it, respectively. We also allowed annotators to label claims as *partial* agreement or disagreement/attack if the response concedes part of the claim, depending on the order of the concession.

In the following example, User A makes a claim and supports it with a premise. User B agrees with the claim but User C disputes the idea that there even is global stability. Meanwhile, User D disagrees with the reasoning that the past is always a good predictor of current events.

> A: [I think the biggest threat to global stability comes from the political fringes.]$_{0:\text{CLAIM}}$ [It has been like that in the past.]$_{1:\text{PREMISE}}$
>
> B: [Good arguments.]$_{2:\text{AGREEMENT:0}}$
>
> C: [The only constant is change.]$_{3:\text{REBUTTAL:0}}$
>
> D: [What happened in the past has nothing to do with the present.]$_{4:\text{UNDERCUTTER:1}}$

We obtain moderate agreement for relation annotations, similar to other argumentative tasks

(Morio and Fujita, 2018). The Inter-Annotator Agreement (IAA) with Kripendorff's $\alpha$ is 0.61 for relation presence and 0.63 for relation types.

In total, the dataset contains 380 turns of dialogue for 2756 sentences. There were 2741 argumentative propositions out of which 1205 are claims and 1536 are premises, with an additional 799 non-argumentative propositions. 66% of relations were in support, 26% attacking, and 8% partial. As we found that most intra-turn relations were in support and inter-turn relations were attacking, due to the dialogic nature of the data, for our experiments we only predicted whether a relation was present and not the relation type.

Overall, there are 7.06 sentences per post for our dataset, compared to 4.19 for Morio and Fujita (2018). This results in a large number of possible relations, as all pairs of argumentative components are candidates. The resulting dataset is very unbalanced (only 4.6% of 27254 possible pairs have a relation in the intra-turn case with only 3.2% of 26695 for inter-turn), adding an extra challenge to modeling this task.

### 3.2 Distant-Labeled Data

As the size of this dataset is small for modern deep learning approaches, we leverage distant-labeled data from Reddit and use transfer learning techniques to fine-tune a model on the appropriate context — micro-level for intra-turn relations and macro-level (dialogue) for inter-turn relations.

**Micro-level Context Data** In order to leverage transfer learning methods, we need a large dataset with distant-labeled relation pairs. In previous work, Chakrabarty et al. (2019) collected a distant-labeled corpus of opinionated claims using sentences containing the internet acronyms IMO (in my opinion) or IMHO (in my humble opinion) from Reddit. We leverage their data to model relation pairs by considering the following sentence (i.e., a premise) as well (when it was present), resulting in a total of 4.6 million comments. We denote this dataset as **IMHO+context**. The following example shows an opinionated claim (in bold) backed by a supporting premise (in italics).

> **IMHO, Calorie-counting is a crock what you have to look at is how wholesome are the foods you are eating.** *Refined sugar is worse than just empty calories - I believe your body uses a lot of nutrients up just processing and digesting it.*

We assume in this data that a relation exists between the sentence containing IMHO and the following one. While a premise does not always directly follow a claim, it does so frequently enough that this noisy data should be helpful.

**Macro-level Context Data** While the IMHO data is useful for modeling context from consecutive sentences from the same author, inter-turn relations are of a dialogic nature and would benefit from models that consider that macro-level context. We take advantage of a feature of Reddit: when responding to a post, users can easily *quote* another user's response, which shows up in the metadata. Particularly in CMV, this feature is used to highlight exactly what part of someone's argument a particular user is targeting. In the example in Table 1, the response contains an exact quote of a claim in the original post. We collect 95,406 threads from the full CMV subreddit between 2013-02-16 and 2018-09-05 and find pairs of posts where the quoted text in the response is an exact match for the original post (removing threads that overlap with the labeled data). This phenomenon occurs a minority of the time, but we obtain 19,413 threads. When the quote feature is used, posters often respond to multiple points in the original text, so for the 19,413 threads we obtain 97,636 pairs. As most language model fine-tuning is performed at the sentence level, we take the quoted text and the following sentence as our distant-labeled inter-post pairs. We refer to this dataset as **QR**, for quote-response pairs.

## 4 Methods

Identifying argumentative components is a necessary precursor to predicting an argumentative relation. In our data, we require the "source" of an intra-turn relation to be a premise that supports or attacks a "target" (a premise or another claim). For inter-turn relations, the source is always a claim that agrees or disagrees with a target. Thus, we model this process as a pipeline: perform three-way classification on claims, premises, and non-arguments and then predict if an outgoing relation exists from the source premise/claim to a target premise/claim. In predicting these relations, we consider all possible source-target pairs of premises and argumentative components within

| |
|---|
| **CMV: A rise in female representation in elected government isn't a good or bad thing**. According to this new story, a record number of women are seeking office in this year's US midterm elections. While some observers hail this phenomenon as a step in the right direction, I don't think it's good thing one way or the other: a politician's sex has zero bearing on their ability to govern or craft effective legislation. As such... |
| "I don't think it's good thing one way or the other: a politician's sex has zero bearing on their ability to govern or craft effective legislation" Nobody is saying that women are better politicians than men, and thus, more female representation is inherently better for our political system. Rather, the argument is that... |

Table 1: Two posts from the CMV sub-reddit where a claim is targeted by the response

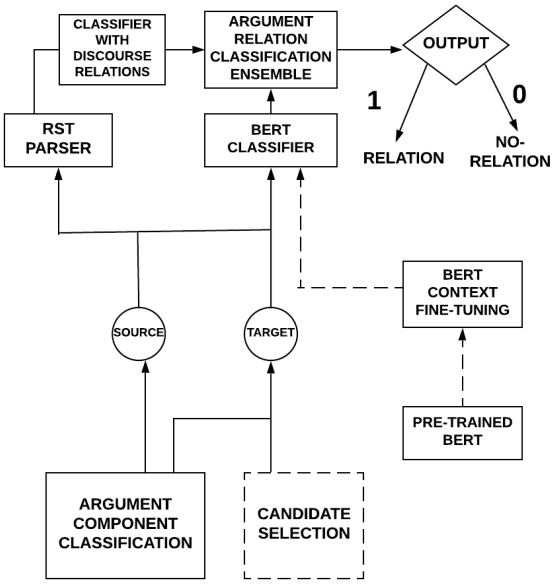


Figure 2: Schematic of our pipeline involving various components. The Candidate Selection Component is only used for Inter-turn relation identification

a single post (for intra-turn) and claims from one post and argumentative components from another post (for inter-turn).

As our data set is relatively small, we leverage recent advances in transfer learning for natural language processing. BERT (Devlin et al., 2019) has shown excellent performance on many semantic tasks, both for single sentences and pairs of sentences, making this model an ideal fit for both argument component classification and relation prediction. The BERT model is initially trained with a multi-task objective (masked language modeling and next-sentence prediction) over a 3.3 billion word English corpus. In the standard use, given a pre-trained BERT model, the model can be used for transfer learning by fine-tuning on a domain-specific corpus using a supervised learning objective.

We take an additional step of fine-tuning on the relevant distant-labeled data from Section 3.2 before again fine-tuning on our data from Section 3.1. We hypothesize that this novel use of BERT will help because the data is structured such that the sentence and next sentence pair (for intra-turn) or quote and response pair (for inter-turn) will encourage the model to learn features that improve performance on detecting argumentative components and relations. For argumentative components, we use this fine-tuned BERT model to classify a single proposition (as BERT can be used for both single inputs and input pairs). For relations (both intra-turn and inter-turn), we predict the relation between a pair of propositions. In the relation case, we postulate that additional components besides BERT can help: adding features derived from RST structure and pruning the space of possible pairs by selecting candidate target components. The pipeline is illustrated in Figure 2.

**Argument Component Classification** We fine-tune a BERT model on the **IMHO+context** dataset described in Section 3.2 using both the masked language modeling and next sentence prediction objectives. We then again fine-tune BERT to predict the argument components for both sources and targets, as indicated in Figure 2. Given the predicted argumentative components, we consider all possible valid pairs either within a post (for intra-turn) or across (inter-turn) and make a binary prediction of whether a relation is present.

**Context Fine-Tuning** For intra-turn relation prediction we use the same fine-tuned BERT model on **IMHO+context** that we used for argument component classification, as premises often immediately follow claims so this task is a noisy analogue to the task of interest. We then fine-tune on the relation prediction task on all possible pairs, using the labeled relations in the CMV data. For inter-turn relation prediction, we fine-tune first on the **QR** dataset, where the dialogue context more closely represents our labeled inter-post relations. Then, we fine-tune on inter-turn relation predic-

tion using all possible pairs as training. This process is indicated in Figure 2, where we use the appropriate context fine-tuning to obtain a relation classifier.

**Discourse Relations** Rhetorical Structure Theory was originally developed to offer an explanation of the coherence of texts. Musi et al. (2018) and, more recently Hewett et al. (2019), showed that discourse relations from RST often correlate with argumentative relations. We thus derive features from RST trees and train a classifier using these features to predict an argumentative relation. To extract features from a pair of argumentative components, we first concatenate the two components so that they form a single text input. We then use a state-of-the-art RST discourse parser (Ji and Eisenstein, 2014)[3] to create parse trees and take the predicted discourse relation at the root of the parse tree as a categorical feature in a classifier. There are 28 unique discourse relations predicted in the data, including *Circumstance*, *Purpose*, and *Antithesis*. We use a one-hot encoding of these relations as features and train an XGBoost Classifier (Chen and Guestrin, 2016) to predict whether an argument relation exists. This classifier with discourse relations, as indicated in Figure 2, is then ensembled with our predictions from the BERT classifier by predicting a relation if either one of the classifiers predicts a relation.

**Candidate Target Selection** For inter-turn relations, we take additional steps to reduce the number of invalid relation pairs. Predicting an argumentative relation is made more difficult by the fact that we need to consider all possible relation pairs. However, some argumentative components may contain linguistic properties that allow us to predict when they are targets even without the full relation pair. Thus, if we can predict the targets with high recall, we are likely to increase precision as we can reduce the number of false positives. Our candidate selection component, which identifies potential targets (as shown in Figure 2), consists of two sub-components: an extractive summarizer and a source-target constraint.

First, we use the **QR** data to train a model to identify candidate targets using techniques from extractive summarization, with the idea that targets may be salient sentences or propositions. We treat the quoted sentences as gold labels, resulting in 19,413 pairs of document (post) and summary (quoted sentences). For the example provided in Table 1, this would result in one sentence included in the summary. Thus, for a candidate argument pair A $\rightarrow$ B, where B is the quoted sentence in Table 1, if B is not extracted by the summarization model we predict that there is no relation between A and B. An example of target selection via extractive summarization is shown in Figure 3.

We use a state-of-the-art extractive summarization approach (Liu, 2019) for extracting the targets. The authors obtain sentence representations from BERT (Devlin et al., 2019), and build several summarization specific layers stacked on top of the BERT outputs, to capture document-level features for extracting summaries. We refer the reader to (Liu, 2019) for further details. We select the best summarization model on a held out subset using recall at the top $K$ sentences.

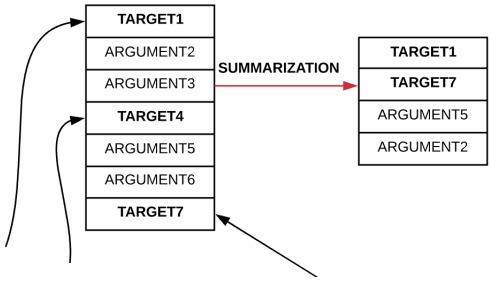


Figure 3: Schematic of our Target Extraction Approach

Second, in addition to summarization, we take advantage of a dataset-specific constraint: a target cannot also be a source unless it is related to the main claim. In other words, if B is a predicted target in A $\rightarrow$ B, we predict that there is no relation for B $\rightarrow$ A except when A is a main claim. In the CMV data, the main claim is always the title of the original Reddit post, so it is trivial to identify.

| Method | C | P | NA |
|---|---|---|---|
| Stab and Gurevych (2017) + EWE | 56.0 | 65.9 | 69.6 |
| Morio and Fujita (2018) | 54.2 | 68.5 | 73.2 |
| Chakrabarty et al. (2019) | 57.8 | 70.8 | 70.5 |
| BERT Devlin et al. (2019) | 62.0 | 72.2 | 71.3 |
| IMHO Context Fine-Tuned BERT | **67.1** | **72.5** | **75.7** |

Table 2: F-scores for 3-way Classification: Claim (C), Premise (P), Non-Argument (NA)

## 5 Experiments and Results

We use the CMV data from Section 3.1 for training and testing, setting aside 10% of the data for test.

[3]We use Wang et al. (2018) for segmentation of text into elementary discourse units.

| Method | Precision | | Recall | | F-Score | |
|---|---|---|---|---|---|---|
| | Gold | Pred | Gold | Pred | Gold | Pred |
| All relations | 5.0 | - | 100.0 | - | 9.0 | - |
| Menini et al. (2018) | 7.0 | 5.9 | 82.0 | 80.0 | 13.0 | 11.0 |
| Menini et al. (2018) + RST Features | 7.4 | 6.1 | 83.0 | 81.0 | 13.7 | 11.4 |
| RST Features | 6.3 | 5.7 | 79.5 | 77.0 | 11.8 | 10.6 |
| Morio and Fujita (2018) | 10.0 | - | 48.8 | - | 16.6 | - |
| BERT Devlin et al. (2019) | 12.0 | 11.0 | 67.0 | 60.0 | 20.3 | 18.5 |
| IMHO Context Fine-Tuned BERT | 14.3 | 13.2 | 69.0 | 65.0 | 23.7 | 21.8 |
| + RST Ensemble | **16.7** | **15.5** | **73.0** | **70.2** | **27.2** | **25.4** |

Table 3: Results for Intra-turn Relation Prediction with Gold and Predicted Premises

Hyper-parameters are discussed in Appendix A.

### 5.1 Argumentative Component Classification

For baseline experiments on argumentative component classification we rely on the handcrafted features used by Stab and Gurevych (2017): lexical (unigrams), structural (token statistics and position), indicator (*I, me, my*), syntactic (POS, modal verbs), discourse relation (PDTB), and word embedding features. Hidey et al. (2017) show that emotional appeal or pathos is strongly correlated with persuasion and appears in premises. This motivated us to augment the work of Stab and Gurevych (2017) with emotion embeddings (Agrawal et al., 2018) which capture emotion-enriched word representations and show improved performance over generic embeddings (denoted in the table as EWE).

We also compare our results to several neural models - a joint model using pointer networks (Morio and Fujita, 2018), a model that leverages context fine-tuning (Chakrabarty et al., 2019), and a BERT baseline (Devlin et al., 2019) using only the pre-trained model without our additional fine-tuning step.

Table 2 shows that our best model gains statistically significant improvement over all the other models ($p < 0.001$ with a Chi-squared test). To compare directly to the work of Chakrabarty et al. (2019), we also test our model on the binary claim detection task and obtain a Claim F-Score of 70.0 with fine-tuned BERT, which is a 5-point improvement in F-score over pre-trained BERT and a 12-point improvement over Chakrabarty et al. (2019). These results show that fine-tuning on the appropriate context is key.

### 5.2 Relation Prediction

For baseline experiments on relation prediction, we consider prior work in macro-level argument mining. Menini et al. (2018) predict argumentative relations between entire political speeches from different speakers, which is similar to our dialogues. We re-implement their model using their features (lexical overlap, negation, argument entailment, and argument sentiment, among others). As with component classification, we also compare to neural models for relation prediction - the joint pointer network architecture (Morio and Fujita, 2018) and the pre-trained BERT (Devlin et al., 2019) baseline.

As the majority of component pairs contain no relation, we could obtain high accuracy by predicting that all pairs have no relation. Instead, we want to measure our performance on relations, so we also include an "all-relation" baseline, where we always predict that there is a relation between two components, to indicate the difficulty of modeling such an imbalanced data set. In the test data, for intra-turn relations there are 2264 relation pairs, of which only 174 have a relation, and for inter-turn relations there are 120 relation pairs, compared to 2381 pairs with no relation.

As described in Section 3.1, for intra-turn relations, the source is constrained to be a premise whereas for intra-turn, it is constrained to be a claim. We thus provide experiments using both gold claims/premises and predicted ones.

**Intra-turn Relations** We report the results of our binary classification task in Table 3 in terms of Precision, Recall and F-score for the "true" class, i.e., when a relation is present. We report results given both gold premises and predicted premises (using our best model from 5.1). Our best re-

| Method | Precision | | Recall | | F-Score | |
|---|---|---|---|---|---|---|
| | Gold | Pred | Gold | Pred | Gold | Pred |
| All relations | 5.0 | - | 100.0 | - | 9.0 | - |
| Menini et al. (2018) | 5.9 | 4.8 | 82.0 | 80.0 | 11.0 | 9.0 |
| Menini et al. (2018) + RST Features | 6.4 | 4.9 | 83.0 | 80.0 | 11.8 | 9.3 |
| RST Features | 5.1 | 3.8 | 80.0 | 77.0 | 9.6 | 7.2 |
| Morio and Fujita (2018) | 7.6 | - | 40.0 | - | 12.7 | - |
| BERT Devlin et al. (2019) | 8.8 | 7.9 | 76.0 | 70.0 | 15.8 | 14.1 |
| QR Context Fine-Tuned BERT | 11.0 | 10.0 | 75.3 | 72.5 | 19.1 | 17.6 |
| + RST Features | 11.0 | 12.2 | 79.0 | 75.5 | 21.2 | 19.1 |
| + Extractive Summarizer | 16.0 | 14.5 | **79.4** | **75.6** | 26.8 | 24.3 |
| + Source $\neq$ Target Constraint | **18.9** | **17.5** | 79.0 | 74.0 | **30.5** | **28.3** |

Table 4: Results for Inter-Turn Relation Prediction with Gold and Predicted Claims

sults are obtained from ensembling the RST classifier with BERT fine-tuned on **IMHO+context**, for statistically significant ($p < 0.001$) improvement over all other models. We obtain comparable performance to previous work on relation prediction in other argumentative datasets (Niculae et al., 2017; Morio and Fujita, 2018).

**Inter-turn Relations** As with intra-turn relations, we report F-score on the "true" class in Table 4 for both gold and predicted claims. Our best results are obtained by fine-tuning the BERT model on the appropriate context (in this case the **QR** data) and ensembling the predictions with the RST classifier. We again obtain statistically significant ($p < 0.001$) improvement over all baselines.

Our methods for candidate target selection obtain further improvement. Using our extractive summarizer, we found that we obtained the best target recall of 62.7 at $K = 5$ (the number of targets to select). This component improves performance by approximately 5 points in F-score by reducing the search space of relation pairs. By further constraining targets to only be a source when targeting a main claim, we obtain another 4 point gain.

**Window Clipping** We also conduct experiments showing the performance for intra-turn relation prediction when constraining the relations to be within a certain distance in terms of the number of sentences apart. Often, in persuasive essays or within a single post the authors use premises to back/justify claims they immediately made. As shown in Figure 4, this behavior is also reflected in our dataset where the distance between the two arguments in the majority of the relations is +1.

We thus limit the model's prediction of a relation to be within a certain window and predict "no relation" for any pairs outside of that window. Table 5 shows that this window clipping on top of our best model improves F-score by only limiting the context where we make predictions. As our models are largely trained on discourse context and the next sentence usually has a discourse relation, we obtain improved performance as we narrow the window size. While we see a drop in recall the precision improves compared to our previous results in Table 3. It is also important to note that window clipping is only beneficial once we have a high recall, low precision scenario because when we predict everything at a distance of +1 as a relation we obtain low F-scores.

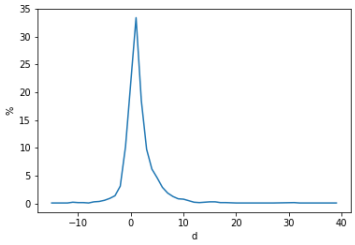


Figure 4: Distances $d$ between Intra-Turn Relations

## 6 Qualitative Analysis

**Role of Context** We retrieve examples from the **IMHO+context** and **QR** data using TF-IDF similarity to pairs of argument components from our data that were predicted incorrectly by pre-trained BERT but correctly by the respective fine-tuned model. The first two rows in Table 6 show a relation between a claim and premise in the IMHO+Context and the CMV data respectively

| Method | Window | Precision | | Recall | | F-Score | |
|---|---|---|---|---|---|---|---|
| | | Gold | Pred | Gold | Pred | Gold | Pred |
| All relations | 0 TO +1 | 5.0 | 4.0 | 31.0 | 25.0 | 8.7 | 6.9 |
| Best Model | 0 TO +5 | 19.5 | 17.1 | **70.0** | **67.0** | 30.5 | 27.2 |
| | 0 TO +4 | 21.4 | 19.5 | 67.0 | 65.0 | 32.2 | 30.0 |
| | 0 TO +3 | 25.2 | 23.3 | 61.1 | 58.0 | 35.6 | 33.2 |
| | 0 TO +2 | 32.5 | 29.8 | 50.0 | 48.0 | 39.3 | 36.8 |
| | 0 TO +1 | **41.5** | **39.1** | 47.0 | 42.0 | **44.1** | **40.3** |

Table 5: Intra-Turn Relation Prediction with Varying Window Settings

while the last four rows show a relation between a claim and premise in the QR data and the CMV data. The model learns discriminative discourse relations from the **IMHO+context** data and correctly identifies this pair. The last four rows show rebuttal from the QR and the CMV data respectively, where the model learns discriminative dialogic phrases (highlighted in bold).

| Context | Pair |
|---|---|
| IMHO | IMHO, you should not **quantify** it as good or bad. **Tons of people** have monogamous relationships without issue. |
| CMV | [how would you even **quantify** that] [there are **many people** who want close relationships without romance] |
| QR | [It might be that egalitarians,anti-feminists, MRAs & redpillers, groups that I associate with opposing feminism - might be in fact very **distinct** & **different** groups, but I don't know that] [**I do see** all four of these as **distinct** groups]. |
| CMV | [I may have a different stance on seeing no **difference** between companion animals and farm animals.] [**I do see distinction** between a pet and livestock] |
| QR | [Of course you **intend to kill** the person if you draw your **weapon**, if you can reasonably assume that they have a **weapon**] [**I don't think** some of them would **start killing**]. |
| CMV | [So i thought, why would a police officer even use **firearms** if he/she doesn't **intend to kill**?] [**I dont think**, police are allowed to **start killing** someone with their gun if they don't intend to . ] |

Table 6: CMV and Context Examples

**Role of Discourse** We also provide examples that are predicted incorrectly by BERT but correctly by our classifier trained with RST features. For the first example in Table 7 the RST parser predicts an *Evaluation* relation, which is an indicator of an argumentative relation according to our model. For the second example the RST parser predicts *Antithesis*, which is correlated with attack relations (Musi et al., 2018), and is predicted correctly by our model.

| Discourse | Argument1 | Argument2 |
|---|---|---|
| Evaluation | The only way your life lacks meaning is if you give it none to begin with | Life is ultimately meaningless and pointless. |
| Antithesis | Joseph was just a regular Jew without the same kind of holiness as the other two | Aren't Mary and Joseph, two holy people especially perfect virgin Mary, both Jews? Wasn't Jesus a Jew? |

Table 7: Predicted Discourse Relations in CMV

## 7 Conclusion

We show how fine-tuning on data-sets similar to the task of interest is often beneficial. As our data set is small we demonstrated how to use transfer learning by leveraging discourse and dialogue context. We show how the structure of the fine-tuning corpus is essential for improved performance on pre-trained language models. We also showed that predictions that take advantage of RST discourse cues are complementary to BERT predictions. Finally, we demonstrated methods to reduce the search space and improve precision.

In the future, we plan to experiment further with language model fine-tuning on other sources of data. We also plan to investigate additional RST features. As the RST parser is not perfect, we want to incorporate other features from these trees that allow us to better recover from errors.


### Acknowledgements

The authors thank Fei-Tzin Lee and Alexander Fabbri for their helpful comments on the initial draft of this paper and the anonymous reviewers for helpful comments.


## References


Rob Abbott, Marilyn Walker, Pranav Anand, Jean E Fox Tree, Robeson Bowmani, and Joseph King.

2011. How can you say such things?!?: Recognizing disagreement in informal political argument. In *Proceedings of the Workshop on LSM*, pages 2–11.

Ameeta Agrawal, Aijun An, Papagelis Chen, and Manos. 2018. Learning emotion-enriched word representations. In *Proceedings of the 27th International Conference on Computational Linguistics*.

Jamal Bentahar, Bernard Moulin, and Micheline Bélanger. 2010. A taxonomy of argumentation models used for knowledge representation. *Artif. Intell. Rev.*, 33(3):211–259.

Filip Boltužić and Jan Šnajder. 2014. Back up your stance: Recognizing arguments in online discussions. In *Proceedings of the First Workshop on Argumentation Mining*, pages 49–58.

Katarzyna Budzynska, Mathilde Janier, Juyeon Kang, Chris Reed, Patrick Saint-Dizier, Manfred Stede, and Olena Yaskorska. 2014. Towards argument mining from dialogue.

Tuhin Chakrabarty, Christopher Hidey, and Kathy McKeown. 2019. IMHO fine-tuning improves claim detection. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 558–563, Minneapolis, Minnesota. Association for Computational Linguistics.

Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA. ACM.

Jacob Devlin, Ming-Wei Chang, Lee Kenton, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 17th Annual Meeting of the North American Association for Computational Linguistics*.

Steffen Eger, Johannes Daxenberger, and Iryna Gurevych. 2017. Neural end-to-end learning for computational argumentation mining. In *In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics.*, pages 11–22.

Vanessa Wei Feng and Graeme Hirst. 2011. Classifying arguments by scheme. In *ACL*, pages 987–996.

Debanjan Ghosh, Smaranda Muresan, Wacholder Nina, Mark Aakhus, and Matthew. Mitsui. 2014. Analyzing argumentative discourse units in online interactions. In *In Proceedings of the First Workshop on Argument Mining, hosted by the 52nd Annual Meeting of the Association for Computational Linguistics, ArgMining@ACL*, pages 39–48.

Ivan Habernal and Iryna Gurevych. 2017. Argumentation mining in user-generated web discourse. *Computational Linguistics*, 43(1):125–179.

Freya Hewett, Roshan Prakash Rane, Nina Harlacher, and Manfred Stede. 2019. The utility of discourse parsing features for predicting argumentation structure. In *Proceedings of the 6th Workshop on Argument Mining*, pages 98–103, Florence, Italy. Association for Computational Linguistics.

Christopher Hidey, Elena Musi, Alyssa Hwang, Smaranda Muresan, and Kathleen McKeown. 2017. Analyzing the semantic types of claims and premises in an online persuasive forum. In *In Proceedings of the 4th Workshop on Argument Mining. EMNLP*, pages 11–21.

Yangfeng Ji and Jacob Eisenstein. 2014. Representation learning for text-level discourse parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics,*, pages 13–24.

Yang Liu. 2019. Fine-tune bert for extractive summarization. In *https://arxiv.org/abs/1903.10318*.

William C Mann and Sandra A Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text-Interdisciplinary Journal for the Study of Discourse*, 8(3):243–281.

Stefano Menini, Elena Cabrio, Sara Tonelli, and Villata Serena. 2018. Never retreat, never retract: Argumentation analysis for political speeches. In *Association for the Advancement of Artificial Intelligence*.

Gaku Morio and Katsuhide Fujita. 2018. End-to-end argument mining for discussion threads based on parallel constrained pointer architecture. In *In Proceedings of the 5th Workshop on Argument Mining. EMNLP*, pages 11–21.

Akiko Murakami and Rudy Raymond. 2010. Support or oppose?: Classifying positions in online debates from reply activities and opinion expressions. In *In Proceedings of the 23rd International Conference on Computational Linguistics, ArgMining@ACL*, pages 869–875.

Elena Musi, Tariq Alhindi, Stede Manfred, Leonard Kriese, Smaranda Muresan, and Andrea Rocci. 2018. A multi-layer annotated corpus of argumentative text: From argument schemes to discourse relations. In *Proceedings of Language Resources and Evaluation Conference (LREC 2018)*.

Vlad Niculae, Joonsuk Park, and Claire Cardie. 2017. Argument mining with structured SVMs and RNNs. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 985–995, Vancouver, Canada. Association for Computational Linguistics.

Andreas Peldszus and Manfred Stede. 2015. Joint prediction in mst-style discourse parsing for argumentation mining. In *EMNLP*, pages 938–948.

Isaac Persing and Vincent Ng. 2016. End-to-end argumentation mining in student essays. In *In Proceedings of the 15th Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.*, pages 1384–1394.

Peter Potash, Alexey Romanov, and Anna Rumshisky. 2017. Heres my point: Joint pointer architecture for argument mining. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1364–1373.

Sara Rosenthal and Kathleen McKeown. 2015. I couldn't agree more: The role of conversational structure in agreement and disagreement detection in online discussions. In *16th Annual Meeting of the SIGDIAL*.

Swapna Somasundaran, Josef Ruppenhofer, and Janyce Wiebe. 2007. Detecting arguing and sentiment in meetings. In *Proceedings of the SIGdial Workshop on Discourse and Dialogue*, volume 6.

Dhanya Sridhar, James R Foulds, Bert Huang, Lise Getoor, and Marilyn A Walker. 2015. Joint models of disagreement and stance in online debate. In *ACL*, pages 116–125.

Christian Stab and Iryna Gurevych. 2014. Identifying argumentative discourse structures in persuasive essays. In *EMNLP*.

Christian Stab and Iryna Gurevych. 2017. Parsing argumentation structures in persuasive essays. In *Computational Linguistics, pages in press, preprint available at arXiv:1604.07370*.

Reid Swanson, Brian Ecker, and Marilyn Walker. 2015. Argument mining: Extracting arguments from online dialogue. In *Proceedings of the 16th Annual Meeting of the SIGDIAL*, pages 217–226.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, Advances in Neural Information Processing Systems 28*, pages 2692–2700.

Jacky Visser, Barbara Konat, Rory Duthie, Marcin Koszowy, Katarzyna Budzynska, and Chris Reed. 2019. Argumentation in the 2016 us presidential elections: annotated corpora of television debates and social media reaction. *Language Resources and Evaluation*.

Marilyn A Walker, Jean E Fox Tree, Pranav Anand, Rob Abbott, and Joseph King. 2012. A corpus for research on deliberation and debate. In *LREC*, pages 812–817.

Yizhong Wang, Sujian Li, and Jingfeng Yang. 2018. Toward fast and accurate neural discourse segmentation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 962–967.

# Evaluating adversarial attacks against multiple fact verification systems

**James Thorne**
University of Cambridge
jt719@cam.ac.uk

**Andreas Vlachos**
University of Cambridge
av308@cam.ac.uk

**Christos Christodoulopoulos**
Amazon
chrchrs@amazon.co.uk

**Arpit Mittal**
Amazon
mitarpit@amazon.co.uk

## Abstract

Automated fact verification has been progressing owing to advancements in modeling and availability of large datasets. Due to the nature of the task, it is critical to understand the vulnerabilities of these systems against adversarial instances designed to make them predict incorrectly. We introduce two novel scoring metrics, attack *potency* and system *resilience* which take into account the correctness of the adversarial instances, an aspect often ignored in adversarial evaluations. We consider six fact verification systems from the recent Fact Extraction and VERification (FEVER) challenge: the four best-scoring ones and two baselines. We evaluate adversarial instances generated by a recently proposed state-of-the-art method, a paraphrasing method, and rule-based attacks devised for fact verification. We find that our rule-based attacks have higher potency, and that while the rankings among the top systems changed, they exhibited higher resilience than the baselines.

## 1 Introduction

Fact verification is the task of predicting whether claims can be supported or refuted by evidence. Advances in this task have been achieved through improved modelling and the availability of resources to train and validate systems (e.g. Wang (2017); Baly et al. (2018b); Thorne et al. (2018)). As this is a task with potentially sensitive applications like propaganda (Baly et al., 2018a) or biased news detection (Potthast et al., 2018), it is critical to understand how systems and models behave when exposed to real-world data and how deficiencies in their training data may contribute to this. It has been observed in related NLP tasks that as models become more complex, it is difficult to fully understand and characterize their behaviour (Samek et al., 2017). And from an NLP perspective, there has been an ongoing discussion as to what extent these models understand language (Jia and Liang, 2017) or they are exploiting unintentional biases and cues that are present in the datasets they are trained on (Poliak et al., 2018; Gururangan et al., 2018).

| **Original REFUTED Instance:** |
|---|
| Bullitt is a movie directed by Phillip D'Antoni |
| **Adversarial REFUTED Instance:** |
| There is a movie directed by Phillip D'Antoni called Bullitt. |
| **Adversarial SUPPORTED Instance:** |
| Bullitt is not a movie directed by Phillip D'Antoni |
| **Evidence:** |
| Bullitt is a 1968 American action thriller film directed by Peter Yates and produced by Philip D'Antoni |

Figure 1: Adversarial instances generated through rule-based transformations of existing claims

One of the diagnostic tools for understanding how models behave is *adversarial evaluation*, where data that is deliberately designed to induce classification errors is used to expose "blind spots" of a system. While there are many recently proposed techniques for generating adversarial instances for NLP tasks (surveyed in Section 2), they vary in the degree to which newly generated instances are correct, i.e. grammatical and appropriately labelled.

In this paper, we introduce two scoring metrics, (adversarial) attack *potency* and system *resilience*, that enable comparison of both adversarial instance generators and the systems that they are executed on respectively. We argue for manual evaluation of instances generated and the incorporation of their correctness for scoring. We con-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2944–2953, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

duct our experiments in the context of the FEVER Shared Task (Thorne et al., 2018) (example in Figure 1), which incorporates both information retrieval and natural language inference to predict whether short factoid claims are SUPPORTED or REFUTED by evidence from Wikipedia. Systems must return not only the correct label but also the sentences providing the evidence for it. In the case where there is not enough evidence in Wikipedia for either label, the label NOTENOUGHINFO (NEI) is applied and no evidence needs to be returned.

We evaluate three adversarial attacks against four state-of-the-art systems and two baselines and apply our proposed scoring metrics that incorporate instance correctness. The first attack, informed by model behaviour, uses Semantically Equivalent Adversarial Rules (SEARs) (Ribeiro et al., 2018), a state-of-the-art method for generating rules that perform meaning-preserving transformations to instances that induce classification errors. The second attack is informed by dataset biases: we identify common patterns and constructions in the claims of the FEVER dataset and exploit them with hand-crafted rules to generate a number of new dataset instances. The final attack is a lexically-informed approach which makes use of a paraphrase model to generate new instances.

Our findings indicate that the instances generated by hand-crafted dataset-informed rules reduced all systems' classification accuracy more than the other approaches we tested. This was because the instances were not only more challenging to the systems under test, they were also more frequently correct – both of these characteristics factor into our *potency* score. While the lexically-informed approach induced a comparable number of misclassifications to the SEARs model, the instances were often incorrect, resulting in a *potency* score that was worse than sampling unmodified instances at random from the test set. Considering the *resilience* of the systems under adversarial evaluation, the $4^{th}$-ranked model from the FEVER challenge performed better than the top system, but otherwise, the rankings of the systems were unchanged, and all top-ranked systems outperformed the baselines.

## 2 Methods for adversarial evaluation

Adversarial examples were initially studied in the field of computer vision (Szegedy et al., 2014) where model deficiencies such as over-sensitivity to perturbed inputs (by altering pixel intensities in a way which is imperceptible to humans) resulted in changes to the predictions. Making similar perturbations to text for adversarial evaluation is more challenging due to the discrete symbol space: modifying a single token may change an instance's semantics or make it ungrammatical. Various methods for attacks have been proposed ranging from manual construction and rule-based perturbation to automated paraphrasing and distractor information. Each of these methods can be informed by observations on the datasets for the task, model predictions and behaviour, and external knowledge and lexical resources. There is also a trade-off between the level of automation (which allows both scale and diversity of new claims) and to what extent the new instances are correct (i.e. grammatical and appropriately labelled). We survey a range of methods for generating adversarial instances and consider the suitability of the methods with respect to the FEVER task and generating correct instances.

**Manual construction informed by expert knowledge:** Instances that exploit world knowledge, semantics, pragmatics, morphology and syntactic variation have been written and compiled into challenge datasets for Machine Translation (Isabelle et al., 2017), Sentiment Analysis (Mahler et al., 2017; Staliūnaite and Bonfil, 2017) and Natural Language Understanding (Levesque, 2013). While these instances are expensive to construct, the attacker would have a high degree of confidence that the instances are correct and therefore correctness is not incorporated into the scoring metrics of any of these works. In FEVER, generating instances is more complex due to the need for annotators to highlight appropriate evidence: scaling up annotation to create new instances from scratch is non-trivial.

**Noise introduced by character-level perturbations:** Character-level attacks highlight the brittleness of systems by making letter swaps or insertions. Belinkov and Bisk (2018), Naik et al. (2018) and Ebrahimi et al. (2018) generate distorted examples which cause misclassifications or translation errors. An evaluation is performed comparing the performance of human crowdworkers for a classification task which only indicated minimal losses in classification accuracy between the orig-

inal and modified instances. While it is unlikely that a single character can unintentionally change the semantics of a sentence, requiring relabelling, this method is still *intentionally* introducing typographical errors meaning that by the definition of the FEVER task, the instances would be incorrect.

**Adding distractor information:** Jia and Liang (2017) evaluated the addition of distractor sentences to instances and their effect on systems trained on the SQuAD question answering task (Rajpurkar et al., 2016). In their study, they compared appending a nonsensical string of words to the instance against appending sensible distractors generated by an automated method with the use of human annotators for a final filtering step. This approach cannot be ported to generate adversarial instances for the FEVER task as the claims are single sentences and the Wikipedia database which is used as evidence is considered immutable. Naik et al. (2018) concatenate manually-written logical tautologies (such as 'and true is true') to instances for a natural language inference task and this approach could be used to modify the claim portion of instances.

**Paraphrasing existing instances:** Iyyer et al. (2018) and Ribeiro et al. (2018) generate adversarial instances through paraphrasing existing ones. The method proposed by Ribeiro et al. (2018) is informed by both the predictions from a model as well as the external resource of a neural translation model whereas the approach taken by Iyyer et al. (2018) make use of a translation model only.

Zhao et al. (2018) use an autoencoder architecture to generate new instances that are semantically equivalent. Paraphrasing introduces the risk of meaning-altering changes to the sentence which would require relabelling, or producing sentences which are ungrammatical. Iyyer et al. (2018) identified that in at least 17.7% of cases, the generated examples were not paraphrases and in at least 14.0% of cases, the paraphrases were ungrammatical. In a pilot study, Zhao et al. (2018) find that 19% of generated examples were not 'semantically similar to the original input'[1] and ungrammatical in 14% of cases.

Ribeiro et al. (2018) generate textual replacement rules that are applied to dataset instances, modifying them in order to generate new semantically equivalent adversarial instances. In a crowdworker evaluation, not all rules are accepted as some induced higher high rates of incorrect instances: for the simpler task of sentiment analysis, 86.6% of rules were retained whereas, for visual question answering, only 43.5% of rules were retained. Paraphrasing approaches are applicable to FEVER, but their success depends on constructing instances that are both correctly labeled and grammatical. As FEVER consists of both information retrieval and natural language inference, the risk of generation of incorrect instances by paraphrasing can be quite high as subtle semantic changes may change the label of the instance.

[1]The authors do not state whether the newly generated adversarial instances retained the original label.

## 3 Potency and resilience

Consider a method for generating adversarial instances (hereafter referred to as an adversary), $a$, that generates a set of instances $X_a = \{x_{a,i}\}_{i=1}^{N}$ with accompanying labels $Y_a = \{y_{a,i}\}_{i=1}^{N}$. To evaluate such adversaries, we must consider both their correctness and their effect on a system under test.

We measure the effectiveness of an adversary ($a$) through a system's ($s$) evaluation measure $f$ (such as $F_1$ or FEVER Score), on a set of predictions made by the system $\hat{Y}_{s,a}$. Intuitively, better adversarial instances induce more misclassifications, resulting in a lower evaluation measure. Assuming the evaluation measure outputs a real value in the range $[0, 1]$, we define *potency* by the average reduction in score (from a perfect score) across all systems, $s \in S$ weighted by the correctness rate of the adversary, $c_a$.

$$\text{Potency}(a) \stackrel{\text{def}}{=} c_a \frac{1}{|S|} \sum_{s \in S} \big(1 - f(\hat{Y}_{s,a}, Y_a)\big) \quad (1)$$

Instances are correct if they are grammatical, appropriately labeled and meet the task requirements. To illustrate the need to incorporate correctness into evaluation, we will compare against *raw potency* where the correctness rate of the system $c_a$ is set to 1. In our experiments, we will be using an estimate of the correctness rate for each adversary through annotating a sample of instances as it is not cost-effective with adversaries generating a large number of instances to perform a complete annotation.

A system that is resilient will have fewer errors induced by the adversarial instances, reflected in

higher scores at evaluation. We wish to penalize systems for making mistakes on instances from adversaries with higher correctness rate. We define *resilience* as the average score scaled by the correctness rate for each adversary, $a \in A$:

$$\text{Resilience}(s) \stackrel{\text{def}}{=} \frac{\sum_{a \in A} c_a \times f(\hat{Y}_{s,a}, Y_a)}{\sum_{a \in A} c_a} \quad (2)$$

## 4 Adversarial attacks against fact verification systems

A FEVER dataset instance (example in Figure 1) consists of a claim accompanied by a label and evidence sentences unless the label is NOTENOUGHINFO (NEI) and no evidence needs to be provided. The adversaries we explore in this paper generate new instances by altering the claim sentence of existing instances and applying a new label if appropriate, without modifying the evidence. We chose this approach over generating completely novel instances as this obviates the need to search for new evidence, a process that is rather error-prone if done automatically. We explore three methods for generating adversarial instances by modifying existing dataset instances: manually crafted rule-based transformation informed by the training data; a recently proposed state-of-the-art method for automatically generating Semantically Equivalent Adversarial Rules (Ribeiro et al., 2018, SEARs) that is model targeted; and a lexically-informed method for paraphrasing model.

### 4.1 Training data-targeted adversary

This adversary assumes access to the dataset used to train the models and it identifies lexico-syntactic patterns in the claims that are highly frequent. The claims following these patterns undergo rule-based transformations to generate new instances with patterns not encountered in the training data. We evaluate three different types of transformations: entailment preserving rewrites, simple negations and complex negations (see examples in Table 1).

The entailment-preserving transformations we use include straightforward rules such as switching from active to passive voice while retaining the original label. For transformations that reverse the labels from SUPPORTED label to REFUTED and vice versa, negations were applied to a claim's verb phrase. More complex negations were introduced by combining the above two techniques.

We constructed 61 rules that matched the most common claim patterns identified in the training portion of the FEVER dataset, for example: $X$ is a $Y$, $X$ was directed by $Y$, $X$ died on $Y$. 23 of our rules were entailment preserving, 19 were simple changes to negate claims and 23 were complex changes that also negated the claims. We release all of the rules on GitHub[2].

### 4.2 Lexically-informed adversary

Our adversary generates paraphrases of the claim without altering the evidence or label in a two-step process. The first step is to generate candidate claims $x'$ by substituting nouns and adjectives with lemmas from all matching synsets from WordNet (Miller, 1995). The second step is to realize the surface forms for each of the lemmas. This is performed using a neural translation model, translating the candidate sentence into a foreign pivot language and back-translating into English yielding $x''$ using the model from Ribeiro et al. (2018). Instances are scored using the ratio of translation probabilities between the paraphrased and the original sentences $\frac{P(x''|x)}{P(x|x)}$ as given by the translation model. For instances where this ratio is low, the translated sentence is not representative of the semantics from the original claim. Where the ratio is high, we observed that the generated instances $x''$ were sometimes nonsensical: $P(x''|x)$ was extremely high in cases where sentences contained a high number of repeated words. To mitigate this, all claims were ranked using this measure and the upper and lower quartiles were excluded.

### 4.3 Model-targeting adversary

We also consider an adversary which exploits model predictions by generating Semantically Equivalent Adversarial Rules (Ribeiro et al., 2018, SEARs). The rules are generated by identifying transformations between paraphrased instances that alter the label predicted by a model while maintaining semantic equivalence. From all the rules generated from the dataset, a submodular selection is performed yielding a subset of non-redundant rules which induce misclassifications that apply to a large proportion of instances.

We applied three sets of adversarial rules to the FEVER dataset to generate new claims. The first set of adversarial rules induced classification er-

[2] https://github.com/j6mes/fever-attacks-emlnp-2019

| Transformation | Pattern | Template |
|---|---|---|
| Entailment Preserving | (.+) is a (.+) | There exists a $2 called $1 |
| | (.+) (?:was \| is)? directed by (.+) | $2 is the director of $1 |
| Simple Negation | (.+) was an (.+) | $1 was not an $2 |
| | (.+) was born in (.+) | $1 was never born |
| Complex Negation | (.+) (?:was \| is)? directed by (.+) | There is a movie called $1 which wasn't directed by $2 |
| | (.+) an American (.+) | $1 $2 that originated from outside the United States. |

Table 1: Example rule-based attacks that preserve the entailment relation of the original claim (within the definition of the FEVER shared task), perform simple negation and more complex negations. The output new claims that are sufficiently different to confuse the classifier. The matching groups within the regular expression are copied into the template (variables begin with $).

rors in classifier for a sentiment analysis task (as reported in Ribeiro et al. (2018)) and the second and third sets of rules were generated using predictions made by the highest scoring model from the shared task (Nie et al., 2019).

Because SEARs requires $\mathcal{O}(n^2)$ model queries, generating rules using the full dataset may be prohibitively expensive for some users. In our evaluation, we compare the rules generated using model predictions on a sample of 1000 instances from the development set against rules generated from the full development set containing 9999 instances.

## 5 Experimental setup

We evaluate the potency of the adversarial instances generated by the approaches described in the previous section against four state-of-the-art models from the FEVER shared task and two baseline models. The models we evaluate were the Neuro-Semantic Matching Network (NSMN) (Nie et al., 2019), HexaF (Yoneda et al., 2018) Enhanced ESIM (Hanselowski et al., 2018) and the Transformer Model (Malon, 2018) as trained by the authors. For the baseline systems, we adopt the architecture from Thorne et al. (2018), using both a Decomposable Attention model and ESIM+ELMo model for natural language inference from AllenNLP (Gardner et al., 2017) and a Term Frequency - Inverse Document Frequency (TF-IDF) model for document retrieval and sentence selection.

Adversarial instances are generated by applying each adversary to existing FEVER instances and making modifications to the claim and label where appropriate. We apply each adversary to the development and test split of the dataset of Thorne et al. (2018)[3] to generate datasets of novel claims. We tune parameters using the development data and report results using a balanced uniform random sample of instances generated from the test data.

We measure instance correctness through blind annotation of a random sample of approximately 100 instances generated by each of the adversaries. We annotate each instance for grammaticality and whether the labelled claim is supported by the evidence from the original instance, following the manual error coding process described in Appendix B of the FEVER dataset description (Thorne et al., 2018).

## 6 Results

Applying each adversary to the 9999 instances in the FEVER test set generated a large number of new adversarial instances. From the training data targeted rule-based adversary, 21348 new claims were created, from the lexically-informed adversary using WordNet substitutions and a paraphrase model, 14046 instances were created, and using SEARs, 3668, 4144 and 7386 instances were generated depending on whether the rules were generated using the model of Nie et al. (2019) on the full dataset, sampled dataset or using the model of the sentiment analysis classifier respectively. For

[3]Note: This is the test split from the paper containing 9999 examples each that later formed part of a development set for the FEVER shared task. The reserved blind test instances from the shared task were not used.

| Rank | Method | Raw Potency (%) | Correct Rate (%) | Potency (%) |
|---|---|---|---|---|
| 1 | Rule-based adversary | 63.16 | 89.5 | 56.53 |
| 2 | *Baseline (Unmodified Instances)* | 44.79 | 97.0 | 43.46 |
| 3 | SEARs (FEVER Full) | 57.84 | 62.5 | 36.15 |
| 4 | SEARs (FEVER Sample) | 53.90 | 55.0 | 29.65 |
| 5 | SEARs (Sentiment) | 47.36 | 50.0 | 23.68 |
| 6 | Paraphrase + WordNet | 65.64 | 34.0 | 22.32 |

Table 2: Potency of adversaries where correctness rate is estimated using inspection of the generated instances. The baseline method of sampled instances is not used for scoring resilience. Raw potency is potency score without the considering the correctness of instances.

| Rank | System | FEVER Score (%) | Resilience (%) |
|---|---|---|---|
| 1 | Transformer | 57.04 | 58.66 |
| 2 | NSMN | 63.98 | 51.09 |
| 3 | HexaF | 62.34 | 50.06 |
| 4 | Enhanced ESIM | 61.32 | 43.98 |
| 5 | TF-IDF + ESIM | 36.87 | 26.86 |
| 6 | TF-IDF + DA | 27.71 | 22.28 |

Table 3: Systems ranked by the resilience to adversarial attacks. The FEVER Score column uses reported scores from the shared task.

each adversary we create a balanced dataset for evaluation through a stratified sample of 1000 instances where we report the *potency* of each adversary and *resilience* of each system, considering instance correctness. Our summary results are presented in two tables: Table 2 describes the potency of each adversary against the systems tested and Table 3 describes the resilience of each system to the adversaries.

Considering potency, the highest scoring adversary was the rule-based method – where the rules were manually written informed by observations on the training set. While the raw potency (not considering the correctness of instances) of the instances generated by the lexically-informed paraphrase adversary was higher than the rule-based adversary, a large number of instances generated from this method failed to meet the guidelines for the task, resulting in the lowest potency in our evaluation. We the adversarial instances that were generated by SEARs – which is informed by model behaviour – also had a higher raw potency than instances sampled from the dataset (baseline), the errors introduced when automatically generating adversarial instances resulted in a lower potency once the correctness of instances was taken into account. However, this method generated a higher rate of correct instances than the paraphrase based method.

For the rule-based method, the largest proportion of incorrect instances were generated by rules that inadvertently matched sentences that encountered in the observations used to design the rules. For example, the rule that transforms the pattern: '$X$ is a $Y$' to 'There is a $Y$ called $X$' correctly generates instances in most cases, but for the sentence "Chinatown's producer is a Gemini" which contains a compound noun, the adversary "There is a Gemini called Chinatown's producer" was generated. Similar errors were produced when the patterns also matched determiners and included these in the new sentence in a position which altered the semantics or were ungrammatical.

The potency of SEARs is dependent on the model and dataset used for generating the Semantically Equivalent Adversarial Rules. When the rules were generated using the full FEVER development split, they not only resulted in a lower FEVER Score (contributing to higher potency) of the models but were also correct more often.

| System | FEVER Score (%) | | | | |
|---|---|---|---|---|---|
| | Rules | SEARS (FEVER Full) | SEARS (FEVER Sample) | SEARS (Sentiment) | Paraphrase |
| Transformer | 56.36 | 58.26 | 62.66 | 68.67 | 44.24 |
| NSMN | 48.85 | 47.85 | 53.35 | 64.56 | 39.44 |
| HexaF | 45.25 | 50.15 | 55.06 | 62.86 | 35.64 |
| Enhanced ESIM | 31.53 | 46.75 | 48.05 | 62.26 | 38.24 |
| TF-IDF + ESIM | 20.72 | 28.13 | 31.03 | 31.03 | 27.83 |
| TF-IDF + DA | 18.32 | 21.82 | 26.43 | 26.43 | 20.72 |

Table 4: Breakdown of FEVER Scores of each system to each adversarial attack prior used for calculating resilience and potency. Lower scores indicates stronger attack (contributing to potency). Higher scores indicate stronger systems (contributing to resilience). Scores in this table do not account for correctness of instances.

Using only 10% of the FEVER development set instances when generating the adversarial rules was orders magnitude faster than the full dataset (generation took about a day instead of over two months), but this had a negative impact on the potency score in two ways: the systems made correct predictions on approximately 4% more of these instances and the instances were less correct about 7% of the time. SEARs generated on the out-of-domain movie sentiment dataset had the lowest potency of the three variants we evaluated. Some of these rules (such as replacement of 'movie' with 'film') worked well when applied to FEVER.

A common failure mode from all variants of SEARs was the replacement of a large number of indefinite articles (such as *a*) with definite articles (such as *the*) making claims nonsensical or altering the semantics to the point where a label change was required. The SEARs also often made changes to determiners and quantifiers or deleted terms such as *only*, which altered the semantics of a claim, making it incorrectly labelled. A higher proportion of this behaviour was observed with SEARs generated from the sentiment analysis task.

The paraphrasing model exhibited a great degree of variance in the quality of the generated claims – resulting in a low correctness score. While all the claims generated were intelligible, there were a large number of grammatical errors present and some nonsensical paraphrases which altered the semantics by substituting synonyms for the wrong word senses. Some of the paraphrases resulted in proper nouns and named entities being replaced by third-person pronouns meaning that these claims failed to meet the FEVER guideline as all entities must be referenced directly to facilitate evidence retrieval.

The most resilient system was the transformer-based model (Malon, 2018) with a score of $58.66\%$. The instances generated by SEARs had a positive impact on the performance as both recall and FEVER Score were higher than the values reported from the shared task and the instances generated by manually constructed rules only negatively affected the model by a marginal amount. Meanwhile, the highest performing system from the shared task (NSMN) (Nie et al., 2019) had a lower resilience: $51.09\%$. Using the breakdown of FEVER Scores by system and adversary (reported in Table 4)), we observe that the NSMN was affected by the rule-based adversary and SEARS (FEVER Full) adversaries much more than transformer model.

With the exception of the transformer model, the resilience of the systems is well correlated with the performance on the original FEVER task. This demonstrates that the top-scoring systems in the shared task were able to better capture the semantic space of the task as the resilience of the baseline TF-IDF models and that evaluation was adequate in assessing their generalization abilities.

### 6.1 Effect of rule-based transformations

All the evaluated systems were pipelines of information retrieval and natural language inference components. We perform an analysis of the models under test considering the effect of rule-based adversarial instances and their impact on each pipeline stage. We compare the performance of the models using the new instances generated by applying transformations to existing instances (re-

| Model | Precision (%) | | | Recall (%) | | |
|---|---|---|---|---|---|---|
| | Original | Modified | Delta | Original | Modified | Delta |
| Transformer | 95.42 | 97.93 | +2.51 | 71.47 | 50.00 | -21.47 |
| NSMN | 41.14 | 44.18 | +3.04 | 78.07 | 71.62 | -6.45 |
| HexaF | 65.23 | 59.76 | -5.47 | 80.78 | 75.98 | -4.80 |
| Enhanced ESIM | 24.36 | 22.30 | -2.06 | 86.04 | 80.93 | -5.11 |
| TF-IDF | 10.49 | 9.10 | -1.39 | 45.50 | 39.20 | -6.30 |

Table 5: Effect of rule-based adversarial attacks on the evidence retrieval component of the pipelines considering sentence-level accuracy of the evidence.

| Model | Accuracy (%) | | | FEVER Score (%) | | |
|---|---|---|---|---|---|---|
| | Original | Modified | Delta | Original | Modified | Delta |
| Oracle + DA | 82.38 | 62.56 | -19.82 | = | = | = |
| Oracle + ESIM | 83.58 | 60.66 | -22.92 | = | = | = |
| TF-IDF + DA | 46.49 | 37.44 | -9.05 | 27.73 | 18.32 | -9.41 |
| TF-IDF + ESIM | 49.75 | 37.34 | -12.41 | 32.83 | 20.72 | -12.11 |
| Transformer | 75.58 | 57.26 | -18.32 | 73.87 | 56.36 | -17.51 |
| NSMN | 70.27 | 50.35 | -19.92 | 68.77 | 48.85 | -19.92 |
| HexaF | 74.67 | 51.35 | -23.32 | 67.37 | 45.25 | -22.12 |
| Enhanced ESIM | 67.56 | 36.84 | -30.72 | 62.76 | 31.53 | -31.23 |

Table 6: Summary of label accuracy and FEVER Scores for instances used in rule-based adversarial attacks. For the case of the oracle evidence retrieval component, FEVER Score is equal to accuracy.

ported in *modified* columns) and compare this to the instances that were used to generate them (reported in *original* columns).

Considering the evidence retrieval component (refer to Table 5), we find that the adversarial instances did not affect the information retrieval component of some systems to the same extent as the NLI component. Most systems incorporated either TF-IDF of keyword matching in their information retrieval component and thus they are little affected by the rule-based transformations (which were mostly adding stop words and reordering the words within the sentence). The only exception is the Transformer Model; even though it uses TF-IDF to retrieve documents, it makes use of an entailment classifier for sentence retrieval from the documents. While this approach maintained very high precision with the newly generated instances, the recall decreased indicating a brittleness for instances that differed from the distribution of the training set.

Considering the NLI stage of the systems (refer to Table 6), we observe that in an oracle environment (using manually labelled evidence – assuming perfect evidence retrieval), both the ESIM and Decomposable Attention models for natural language inference suffer a stark decrease in accuracy when making predictions on the adversarial examples. The decrease is less pronounced when considering the full pipeline (i.e. with an actual imperfect evidence retrieval component), which reduces the upper bound for the FEVER Score and accuracy. This is due to the noise introduced by the evidence retrieval pipeline stage.

The Enhanced ESIM system had the highest reduction in accuracy for the rule-based adversarial instances. Even though the model had an evidence recall of $80.93\%$ (a reduction of $5.11\%$), the FEVER score reduced by $30.72\%$. On inspection, this model is mostly predicting SUPPORTED for most adversarial instances. In contrast, the Transformer Model had the lowest reduction in FEVER score ($-17.51\%$) despite a reduction in evidence recall of $23.17\%$. This model also exhibited similar behaviour to the Enhanced ESIM model: while a large number of supported claims were cor-

rectly classified, this model predominantly predicted NOTENOUGHINFO for the adversarial instances. As this class does not require evidence to be correctly scored, 'falling back' to it resulted in a higher FEVER Score when no suitable evidence was found.

The rule-based adversary applies both entailment-preserving and label-altering (through negation) transformations to claims to generate new adversarial instances. We observe in all models (with the exception of the Enhanced ESIM (Hanselowski et al., 2018)) that the reduction in accuracy for the claims generated through entailment-preserving transformations was lower than the claims generated by the negating transformations. This may be revealing an inherent bias in the models similar to the one discussed by Naik et al. (2018), where models perform poorly for antonymous examples due to a dependence on word-overlap as a feature.

## 7 Conclusions

As automated means for generating adversarial instances do not always produce grammatical or correctly labelled instances, we developed a method for evaluation which incorporates instance correctness. We introduced two metrics that enable evaluation of adversarial attacks against systems: the *potency* of attack, and the *resilience* of systems to these attacks.

In this paper, we evaluated four state-of-the-art systems and two baselines against instances generated from three methods for adversarial attack: manually written rules that were informed by observations of the training data, an automated lexically-informed paraphrasing model and an automated state-of-the-art model-targeted approach. While the manually-written rules exhibited high correctness rate, the automated methods often made grammatical errors or label-altering changes which lowered the correctness rate. Our metrics captured this trade-off between correctness and impact on system scores – despite the paraphrasing automated method having the greatest impact on systems under test, it was not the most potent, owing to the low correctness rate.

We hope that the findings and metrics that we present in this paper help continue the discussion on improving the robustness of models and enable better comparisons between adversarial attacks to be made that consider instance correctness.

## Acknowledgements

The authors wish to thank Trevor Cohn, Tim Baldwin and Oana Cocarascu for their helpful advice. The authors also wish to thank the teams UNC, UCLMR, Athene and Papelo of the FEVER1 shared task who made source code and pre-trained models available and were responsive to our questions.

## References

Ramy Baly, Georgi Karadzhov, Dimitar Alexandrov, James Glass, and Preslav Nakov. 2018a. Predicting factuality of reporting and bias of news media sources. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3528–3539, Brussels, Belgium. Association for Computational Linguistics.

Ramy Baly, Mitra Mohtarami, James Glass, Lluis Marquez, Alessandro Moschitti, and Preslav Nakov. 2018b. Integrating Stance Detection and Fact Checking in a Unified Corpus.

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and Natural Noise Both Break Neural Machine Translation. In *ICLR*, pages 1–13.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. HotFlip: White-Box Adversarial Examples for Text Classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Short Papers)*, pages 31–36, Melbourne, Australia. Association for Computational Linguistics.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2017. AllenNLP: A Deep Semantic Natural Language Processing Platform.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R Bowman, and Noah A Smith. 2018. Annotation Artifacts in Natural Language Inference Data.

Andreas Hanselowski, Hao Zhang, Zile Li, Daniil Sorokin, Benjamin Schiller, Claudia Schulz, and Iryna Gurevych. 2018. Multi-sentence textual entailment for claim verification. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 103–108, Brussels, Belgium. Association for Computational Linguistics.

Pierre Isabelle, Colin Cherry, and George Foster. 2017. A challenge set approach to evaluating machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2486–2496.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 1875–1885.

Robin Jia and Percy Liang. 2017. Adversarial Examples for Evaluating Reading Comprehension Systems.

Hector J Levesque. 2013. On our best behaviour. *IJCAI*.

Taylor Mahler, Willy Cheung, Micha Elsner, David King, Marie-Catherine de Marneffe, Cory Shain, Symon Stevens-Guille, and Michael White. 2017. Breaking nlp: Using morphosyntax, semantics, pragmatics and world knowledge to fool sentiment analysis systems. In *Proceedings of the First Workshop on Building Linguistically Generalizable NLP Systems*, pages 33–39.

Christopher Malon. 2018. Team papelo: Transformer networks at fever. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 109–113, Brussels, Belgium. Association for Computational Linguistics.

George a. Miller. 1995. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41.

Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Rose, and Graham Neubig. 2018. Stress Test Evaluation for Natural Language Inference. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2340–2353, Santa Fe.

Yixin Nie, Haonan Chen, and Mohit Bansal. 2019. Combining fact extraction and verification with neural semantic matching networks. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis Only Baselines in Natural Language Inference. (1):180–191.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2018. A stylometric inquiry into hyperpartisan and fake news. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 231–240, Melbourne, Australia. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. pages 2383–2392.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Semantically Equivalent Adversarial Rules for Debugging NLP models. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 856–865. Association for Computational Linguistics.

Wojciech Samek, Thomas Wiegand, and Klaus-Robert Müller. 2017. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *CoRR*, abs/1708.08296.

Ieva Staliūnaite and Ben Bonfil. 2017. Breaking sentiment analysis of movie reviews. In *Proceedings of the First Workshop on Building Linguistically Generalizable NLP Systems*, pages 61–64.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *International Conference on Learning Representations*.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for Fact Extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana. Association for Computational Linguistics.

William Yang Wang. 2017. "Liar, Liar Pants on Fire": A New Benchmark Dataset for Fake News Detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 422–426.

Takuma Yoneda, Jeff Mitchell, Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Ucl machine reading group: Four factor framework for fact finding (hexaf). In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 97–102, Brussels, Belgium. Association for Computational Linguistics.

Zhengli Zhao, Dheeru Dua, and Sameer Singh. 2018. Generating Natural Adversarial Examples. *ICLR*.

# Nonsense!: Quality Control via Two-Step Reason Selection for Annotating Local Acceptability and Related Attributes in News Editorials

**Wonsuk Yang** **Seungwon Yoon** **Ada Carpenter** **Jong C. Park**[†]
School of Computing
Korea Advanced Institute of Science and Technology
{derrick0511, swyoon, ada, park}@nlp.kaist.ac.kr

## Abstract

Annotation quality control is a critical aspect for building reliable corpora through linguistic annotation. In this study, we present a simple but powerful quality control method using two-step reason selection. We gathered sentential annotations of local acceptability and three related attributes through a crowdsourcing platform. For each attribute, the reason for the choice of the attribute value is selected in a two-step manner. The options given for reason selection were designed to facilitate the detection of a nonsensical reason selection. We assume that a reliable annotation may not contain a nonsensical reason selected for the choice of the attribute value, and an annotation that contains a nonsensical reason is less reliable than the one without such reason. Our method, based solely on this assumption, is found to retain the annotations with remarkable quality out of the entire annotations mixed with those of low quality.

## 1 Introduction

Crowdsourcing has recently enabled the collection of large amounts of data quickly. However, crowdsourced annotations require more quality control than traditional in-house annotations. This is because it is difficult to assure the expertise of the participating crowdsource workers in the linguistic annotation.

To ensure the quality of the annotations obtained through crowdsourcing, a frequently utilized approach is to request several workers annotate the same data, aggregate the annotations, and infer the ground truth based on the assumption that the consensus of the workers would lead to the ground truth (Lease, 2011). Various statistical methods have also been proposed to filter out the mistakes or (spamming) random responses of the crowdsource workers (Liu et al., 2012; Hovy et al., 2013; Nguyen et al., 2017). However, the way to filter out the mistakes or the random responses through statistical means is difficult to utilize for a fundamentally subjective annotation task. This is because statistical methods make it hard to differentiate the annotator's mistakes from their subjective annotations, as the individual subjectivity of views on different topics can make the gathered data apparently random.

Local acceptability, which is for how much the given text is rationally worthy of being believed to be true (Wachsmuth et al., 2017; Yang et al., 2019), is a fundamentally subjective measure. It is thus hard to utilize only statistical methods for the annotation quality control. Yet, local acceptability is also understood as a strong factor influencing the argumentation quality in writing, which is of great importance for computer-assisted writing. In this work, we present a simple but powerful method for annotation quality control of local acceptability using two-step reason selection, where the options given for reason selection were designed to facilitate the detection of a nonsensical reason selection. In order to show the effectiveness of the method, we performed the sentential annotation of local acceptability and three possibly related attributes through crowdsourcing.

Using our method, we were able to filter out the annotation results with Krippendorff's alpha of 0.3 for local acceptability, which is comparable to our sentence-level in-house annotation of local acceptability (Yang et al., 2019) and the document-level annotation of local acceptability by Wachsmuth et al. (2017). The overall, non-filtered annotation results were with the alpha of 0.01. Our analysis on argumentation strategy indicates that the filtered annotation results match good linguistic intuition whereas those non-filtered do not. The filtered annotation results also show distinctive cor-

[†] Corresponding author

relation patterns whereas those non-filtered do not.

The contributions of this paper are as follows. (1) We present a new method of using reason selection for the quality control of the annotation of local acceptability and three (possibly related) attributes. (2) We provide a statistical analysis of the annotation results and detailed statistics on the filtering and show that our method has a good potential for future research into annotation quality control. (3) We make all the related data and the code for filtering publicly available.

To the best of our knowledge, our method is the first in utilizing nonsensical reason selection for the quality control of the linguistic annotation.

## 2 Related Work

### 2.1 Argument mining and the annotation of the reasons

Hasan and Ng (2014) annotated the reasons for ideological debate posts and built the classifiers for such reasons. They set out four target domains, and categorized possible reasons into several groups for each of the two stances of support and opposition. Habernal and Gurevych (2016) annotated the convincingness of arguments through crowdsourcing, with a specified reason for the annotated convincingness. They presented a decision tree that restricts the scope of the possible reasons and enables a structured analysis of the reasons selected by workers. Ding et al. (2018) annotated the reasons for an event being affective, based on seven categories of common human needs. They formalized a classification task for such seven categories by assigning a gold label on the reason for each event. They also showed that baseline classifiers could achieve moderate performance on the task. On the other hand, we focus on local acceptability of arguments with restricted options for reason selection. We choose the reasons not only for a deeper understanding of the reasons for local acceptability judgment itself, but also for an automated identification of the random responses by the workers.

### 2.2 Quality control of crowdsourced annotation

Raykar et al. (2010) proposed a nonlinear statistical method to infer the actual, hidden label among the multiple and possibly noisy labels provided by multiple annotators, and showed that their method is superior to that of majority voting. Liu et al. (2012) addressed the problem of aggregating the labels from unreliable annotators via crowdsourcing as a standard inference problem in graphical models and showed that the belief propagation model and the mean field model can be effective. Hovy et al. (2013) proposed MACE (Multi-Annotator Competence Estimation) system that can identify the adversary (spamming) annotators utilizing a latent variable that can encode the information of whether or not an annotator is spamming and when the annotator is spamming. Nguyen et al. (2017) utilized Hidden Markov Model to aggregate the sequential labels obtained through a crowdsourcing platform,

| | Score | Description |
|---|---|---|
| Local Acceptability | 7 | I **strongly accept** the information given by the sentence to be true. I have sound and cogent arguments to justify my acceptance. |
| | 6 | I **accept** the information given by the sentence to be true. I have some arguments to justify my acceptance. |
| | 5 | I **weakly accept** the information given by the sentence to be true. I do not have arguments justifying my acceptance. Still, I will accept it rather than reject it. |
| | 4 | It is **hard to judge** whether I should accept or reject the information given by the sentence to be true. |
| | 3 | I **weakly reject** the information given by the sentence to be true. I do not have arguments for the rejection. Still, I will reject it rather than accept it. |
| | 2 | I **reject** the information given by the sentence to be true, and I have arguments for the rejection. |
| | 1 | I **strongly reject** the information given by the sentence to be true. I have sound and cogent arguments for the rejection. |
| Knowledge Awareness | 3 | I **already knew** the information before I read this document. |
| | 2 | I **did not know** the information before I read this document, **but came to know** it by reading the previous sentences in this document. |
| | 1 | I **did not know** the information. |
| Verifiability | 5 | I can verify it **using my knowledge**. I do not need to google it to verify. |
| | 4 | I can verify it by **short-time googling**. |
| | 3 | I can verify it by **long-time googling**. I could verify it using deduction if I google it for some time for deeper understanding. |
| | 2 | I might find an **off-line way** to verify it, but it will be very hard. |
| | 1 | There is **no way to verify** it. |
| Disputability | 4 | Whether or not it is reasonable to accept the information given by the sentence as true, it is **highly disputable**. |
| | 3 | Whether ... true, it is **disputable**. |
| | 2 | Whether ... true, it is **weakly disputable**. |
| | 1 | Whether true, it is **not disputable**. |

Table 1: Description of Local Acceptability and related attributes.

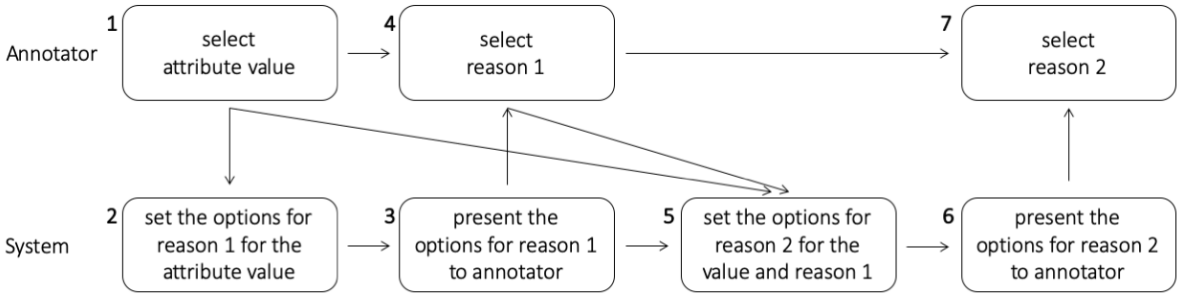


Figure 1: The process of the annotation for an attribute value. For a sentence, the annotator iterates the process four times for the four attributes to annotate. The number at the upper left side of each box indicates the order of the process. The arrow indicates the passing of the information.

and showed that their method works satisfactorily on the sequential labeling tasks of Named Entity Recognition and Information Extraction.

Guillaume et al. (2016) presented ZombiLingo, which is a 'Game with a Purpose' (GWAP) for dependency syntax annotation. Even though the annotation of dependency syntax is expected to require moderate linguistic expertise, they gathered more than 100,000 annotations in 9 months for French dependency syntax with 0.93 precision by decomposing the task into sub-tasks and by including the annotator training as a part of the game. Abad et al. (2017) presented a method for crowdsourced annotation, in which an automatic classifier is used for self-training. For some data with which the classifier shows high confidence for the predicted label, their self-training system guides the annotators to the (highly likely) correct labels predicted by the classifier. For the task of assessing the quality of the machine translated results, Mathur et al. (2018) estimated the reliability of the workers by presenting multiple translations as an assignment. For the translations in an assignment, some were already scored by human experts, and some were presented twice in the assignment. They assessed the reliability of each annotator by seeing how close their annotations were to the expert-annotated results, and whether they annotated the same translations with the same score.

In this study, unlike the previous researches, we use the reason selection method for an automated identification of the random responses by the workers. To the best of our knowledge, there has been no method utilizing reason selection for automated identification, even though incoherence between the selected reason and the attribute value can be a strong indicator for random responses. Our experiment shows that the reason selection method is simple, but very powerful.

## 3 Data

We used the Webis-Editorials-16 corpus provided by Al-Khatib et al. (2016). For 300 news editorials, they annotated discourse units in each news editorial with the labels that are related to an argumentation strategy, such as (1) *Common Ground*, (2) *Assumption*, (3) *Testimony*, (4) *Statistics*, (5) *Anecdote*, and (6) *Other*. We randomly selected 105 news editorials among the original 300 news editorials in the corpus and annotated local acceptability and the related attributes for each sentence in the 105 news editorials. The news editorials are somewhat out-dated, but we have chosen to use the data with two reasons: (1) Their annotation of argumentation strategy has high and reliabile quality. (2) Our previous sentential annotation of local acceptability (Yang et al., 2019) is based on the news editorial corpus.

## 4 Annotation

### 4.1 Definition

Local acceptability is defined, for the premises of an argument, as "rationally worthy of being believed to be true" (Wachsmuth et al., 2017). A premise gives the reason for justifying (or refuting) a claim, where the claim is a possibly controversial statement and the central argument component. A claim and one or more premises compose an argument (Stab and Gurevych, 2017). Following the definition of local acceptability by Wachsmuth et al. (2017), we define the local acceptability *of a sentence*, based on the truth-value of the sentence defined in the *truth-conditional*

| Attribute Value | Reason 1 | Reason 2 |
|---|---|---|
| I did not know the information | I am not familiar with the topic | I am an expert on the subject |
| I did not know the information before, but came to know it by reading the previous sentences | It is not directly stated so far, but I can infer it from the previous sentences | The same information was stated in one of the previous sentences |
| I can verify it using my knowledge | I can logically verify the information | It is a transition sentence that is meaningless |
| strong accept | It is not a factual information, but I agree with the statement | It is a fact |

Table 2: Examples for nonsensical reason

| Attribute Value | Reason 1 | Reason 2 |
|---|---|---|
| weakly disputable | Some may think different on some details | It is a famous conflict |
| I already knew the information before I read this document | I know it from my personal experience | I am not familiar with the topic |
| I can verify it by short-time googling | I would find a direct reference to verify it | I can interview the people related to it |
| accept | It is a subjective statement, but I agree with the statement | It can easily be verified |

Table 3: Examples for unnatural reason

*theory* (Lewis, 1970), as: A sentence is locally acceptable if the truth-value of the sentence is rationally worthy of being believed to be true (Yang et al., 2019).

### 4.2 Attributes

Table 1 shows the rubrics for local acceptability and the other three attributes that we annotated through crowdsourcing. We introduced the three attributes (and the rubrics) in Yang et al. (2019) for a more in-depth understanding of local acceptability, focusing on the aspects of computational journalism (Cheruiyot and Ferrer-Conill, 2018; Aharoni and Tenenboim-Weinblatt, 2019).

**Local Acceptability (LA)** is an indicator of whether the truth-value of the sentence is rationally worthy of being believed to be true. We divide each of the *accept* and *reject* judgments into three subcategories. For *accept*, we divided it into *strong accept*, *accept*, and *weak accept*, and the same goes for *reject*. We did so as the judgment on the truth-value can be unclear (Hamblin, 1970).

**Knowledge Awareness (KA)** is an indicator of whether or not an annotator already knew the information given by the sentence before reading a document.

**Verifiability (V)** is an indicator of how easy it is to verify the information given by the sentence. We include this attribute because we anticipate that the information recognized as easily verifiable by a reader would be more likely to be accepted by the reader.

**Disputability (D)** is an indicator of how controversial the information given by the sentence is. We anticipate that the personal acceptance of information by a reader would be different from the reader's expectation of the public acceptance of the information by others.

## 5 Method

### 5.1 Two-step reasons for each attribute

For each attribute of a sentence, after the annotator chose its value according to the rubrics, they were asked also to choose the reasons for the value, in a two-step manner. At the first step, the annotator was asked to select one of the possible options as a reason (reason 1). Then, they were asked to select another reason (reason 2), where the options for the second step were determined by the attribute value and the reason selected in the first step. Figure 1 illustrates the process of the attribute value and reason selection.

The options for a reason were designed manually, based on our previous sentence-level in-house annotation result (Yang et al., 2019), mainly because the automatic categorization of reasons is a fundamentally challenging problem (Skorupski, 2002). The in-house annotation was performed by three of the authors and seven undergraduate students with native competence in English, where three of them were student journalists responsible for the school newspaper. The in-house annotation was for the same attributes but the students wrote down the reasons by themselves without any given

| Attribute | Attribute Value | Reason 1 | Reason 2 | label |
|---|---|---|---|---|
| Local Acceptability | strong accept | It is a subjective statement, but I agree with the statement | It is a fact | Nonsensical |
| Knowledge Awareness | I did not know the information | I am not familiar with the topic | I am an expert on the subject | Nonsensical |
| Verifiability | I might find an off-line way to verify it, but it will be very hard | I could visit related places to obtain the information | I can interview the people related to it | Natural |
| Disputability | weakly disputable | Some may think different on some details | It is a famous conflict | Unnatural |

Table 4: An example for the two-step reason selection for a sentence

option. The in-house annotation gathered 4,623 sentential annotations for the 105 news editorials with specific reasons written without any given option. One of the authors thoroughly read all of the 4,623 sentential annotations, and two other authors inspected part of the annotations. Based on the in-house annotation result, three of the authors discussed and designed the options so that they can cover all the responses on the in-house annotation result. Yet, anticipating that there may be a case where the options for the reasons were not enough for a sentence, we allowed the workers to choose *n/a* at each of the three steps to select (1) attribute value, (2) reason 1, and (3) reason 2. When a worker chose *n/a* for an attribute value, we asked the annotator to select the reasons in a two-step manner, and if a worker chose *n/a* for either reason 1 or reason 2, we did not ask the worker for further details. We expect that the selections of *n/a* would enable us to estimate the range of sentences that the option design may not cover.

### 5.2 Nonsensical/Unnatural Reason

For each attribute, we intentionally added nonsensical/unnatural options for a reason at the second step. Those were designed to identify the mistakes or the (spamming) random responses by the workers. We deemed the selection of the nonsensical/unnatural reason as an indicator for the unreliability of the annotation. The definition of the nonsensical/unnatural reason is as follows.

**Nonsensical reason** is the reason that a reliable annotator would not select for the annotation and is the reason that works as an indicator for a (spamming) random response. For example, if the reason 'I am not familiar with the topic' for knowledge awareness is selected with the reason 'I am an expert on the subject,' we find that it does not make any sense. Table 2 shows examples for nonsensical reason.

**Unnatural reason** is the reason that a reliable annotator is less likely select than the other given options, but it is the reason behind which the annotator might have a sophisticated rationale. For example, if an attribute value of *weakly disputable* for disputability is selected with the reason of 'It is a famous conflict,' it makes some sense, but it is deemed as unnatural. This is because if a sentence is about a famous conflict, the attribute values of *disputable* or *highly disputable* would have been more suitable. Table 3 shows examples for unnatural reason.

For each of the triples of (attribute value, reason 1, reason 2) that can be selected by the worker, we assigned one of the following labels: (1) natural, (2) unnatural, and (3) nonsensical. The assignment was also conducted manually by three of the authors. The reader is referred to Appendix A for the entire set of options and the corresponding labels for the two-step reason selection. Table 4 shows an example of the possible selection for a sentence. It should be noted that a sentential annotation can contain more than one nonsensical/unnatural reason. If an annotator chose the options in Table 4 for a sentence, the sentential annotation contains two nonsensical reasons and an unnatural reason.

### 5.3 Unreliability Score

For sentential annotation, we compute the unreliability score of the annotation as follows:

$$u(x) = 0.5 \times N_u(x) + N_n(x) \quad (1)$$

where $x$ is a sentential annotation, $u(x)$ is the unreliability score of $x$, $N_u(x)$ is the number of the unnatural reasons over the four attributes within $x$, and $N_n(x)$ is the number of nonsensical reasons within $x$. For example, $u(x)$ is 2.5 for the selection in Table 4. Regarding the coefficient 0.5, another value can also be chosen in its place. The unreliability score can be defined as a function that takes $N_u(x)$ and $N_n(x)$ as arguments. In this paper, we are using the linear function with an ad-hoc coefficient of 0.5 to demonstrate the effectiveness of our method as we believe that it is the simplest form of the equation.

For a worker, we compute his or her unreliability score as follows:

$$U(w) = Average(u(x)) \quad (2)$$

for $x$ in $X(w)$ where $w$ is a worker, $U(w)$ is the unreliability score of $w$, $X(w)$ is the set of annotations annotated by $w$ (possibly over multiple news editorials), and $Average(u(x))$ for $x$ in $X(w)$ indicates the average of the unreliability scores of all the annotations conducted by the worker $w$.

## 6 Experiment

### 6.1 Setting

In this study, the annotators were asked to proceed with the annotation in one direction. In other words, they were asked to read a sentence, to annotate the sentence, then to read the next sentence, and so on. Therefore, before proceeding with the annotation, we built an annotation tool in order to provide an environment that helps unidirectional reading. The annotation tool shows only a single sentence at a time on the screen, and the annotator could proceed to the next sentence by a keystroke. We allowed the annotators to go back to the previous sentence, and modify the previous annotation. The tool also recorded the time of each keystroke.

We conducted the annotation through the Amazon Mechanical Turk (AMT) crowdsourcing platform. In each assignment, the workers were presented with the annotation guideline and URL for a news editorial. For each sentence, workers were asked to select an attribute value for each attribute and to select the reasons for the attribute value in a two-step manner. The workers were asked to proceed to the annotation using the annotation tool. Although we used the corpus of argument strategies as the basis for our annotation, we provided the annotators only with plain sentences, without any extra information hinting at the argument strategies. Moreover, we paid attention not to inform the workers of the fact that we could assess the annotation quality based on the nonsensical/unnatural reason selection during the entire communication with the workers. We also did not reject any worker for the nonsensical/unnatural reason selection.

We made an additional request for the annotations from annotators who completed them too quickly, for instance in less than 20 seconds for all the 12 values for a sentence. If the annotators did not agree to re-do the annotation for the second time, we rejected the first annotation. We made this additional request on 29% of the initially requested assignments. Among these, 90% were re-annotated, and 10% were rejected. For the re-requested annotations, we dismissed the first annotations and took only the re-annotated results for our experiment. As a result, counting only the approved (and not rejected) annotations, 218 workers (with the lifetime approval rate $> 95\%$) participated in our annotation, over about two weeks including the time for the re-request. One annotator was allowed to annotate multiple news editorials, but not the same news editorials multiple times.

Using our method, we ranked the workers based on his or her unreliability score, and divided the workers into three groups as follows: (1) the group of top 1/3 workers with the *lowest* unreliability score per worker, (2) the group of bottom 1/3 workers with the *highest* unreliability score per worker, and (3) the group of the other workers. Among the 218 workers who participated in our annotation, the size of each group was (1) 72, (2) 73, and (3) 73, respectively, as 218 is not a multiple of three.

Then, we grouped the annotations conducted by the annotators in each of the (1)-(3) groups and named the groups of annotations as (1) *good*, (2) *bad*, and (3) *moderate* annotations, respectively. Among the *good* annotations, we filtered out some of the annotations again with the lowest unreliability score per annotation so that the number of retaining annotations in the group would be 10% of the total annotations gathered from our experiment, and named the group as *good squared* annotations. In short, the *good*, *bad*, and *moderate* annotations were filtered based on the unreliability of the annotators, and the *good squared* annotations were those filtered again from the *good* annotations based on the unreliability score of each annotation. It should be noted that the four groups are those of annotations, not those of workers.

### 6.2 Results

#### 6.2.1 Statistics

We annotated 3,591 sentences in 105 news editorials and gathered 17,414 sentence level annotations. For each of the documents, the average of the number of annotators per news editorial was 4.85. We initially set the number of annotators as five annotators per news editorial, and 3% of the assignments were rejected. For the unrelia-

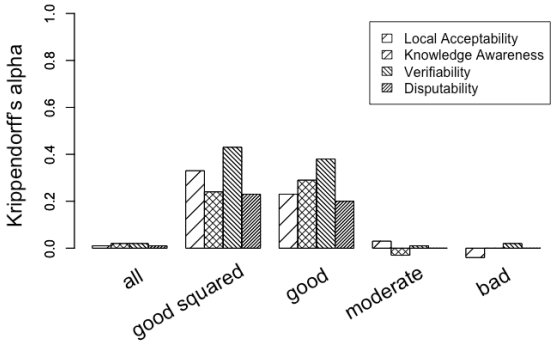


Figure 2: Inter-annotator agreement

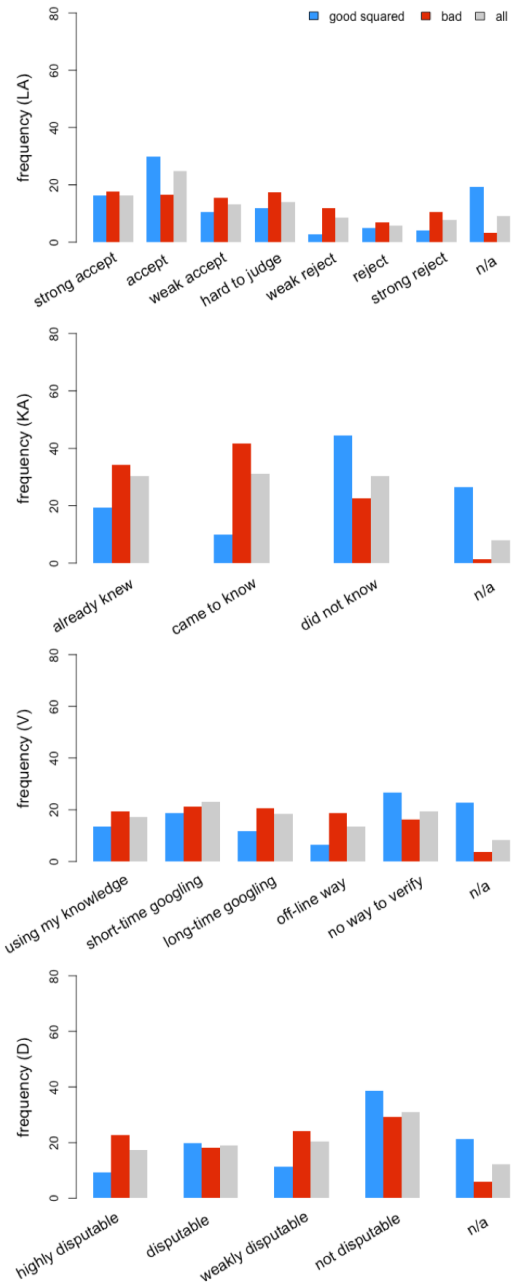


Figure 3: Relative occurrence frequency (unit: %)

bility score of annotation, the average of the unreliability score, over the annotations in the *good squared*, *good*, *moderate*, *bad* annotations, is zero, 0.05, 0.48, and 0.84, respectively. The number of annotations within the *good squared*, *good*, *moderate*, and *bad* group is 1,741 (10%), 3,024 (17%), 6,304 (36%), and 8,086 (46%), respectively.

We speculate that the imbalance on the number of annotations for each of the *good*, *moderate*, *bad* annotations might have been due to the difference in the time that took for the annotation per annotator. The average time that took for a news editorial [1] was 54 minutes per assignment (per document, *i.e.*, per news editorial) for overall annotators. For the *good*, *moderate*, and *bad* annotations, it was 122, 66, and 19 minutes, respectively. For the *good squared* annotations, it was the same as the *good* annotations, as the filtering from the *good* annotations to *good squared* annotations is conducted only on an annotation-level filtering within an assignment, not affecting the calculation. As the workers could be assigned to another news editorial until the study ends, the workers who completed an assignment faster could be given another assignment, hence the different number of annotations.

### 6.2.2 Inter-annotator Agreement

In order to check for the effectiveness of the quality control method using the two-step reason selection, for each of the four groups, we retained only the annotations by the annotators in the group and measured the inter-annotator agreement (IAA) of the annotations that were retained, dismissing all the other annotations in the other groups. For example, for the *good squared* annotations, we retained only the 1,741 *good squared* annotations among the 17,414 crowdsourced annotations, dismissing all the other 15,673 (=17,414−1,741) annotations and measured the IAA for the 1,741 annotations that were retained. The IAA for *good*, *moderate*, and *bad* annotations was calculated in the same way. Figure 1 shows the inter-annotator agreement for each group. We measured the IAA based on the Krippendorffs alpha, as the attribute values are ordinal. For local acceptability, alpha was 0.33 for *good squared* annotations and 0.23 for *good* annotations. On the other hand, for all of *moderate*, *bad*, and *all* annotations, the IAA for each attribute was less than 0.03, which is very

[1] We dismissed 18 outliers taking time longer than 10 hours for annotating a news editorial.

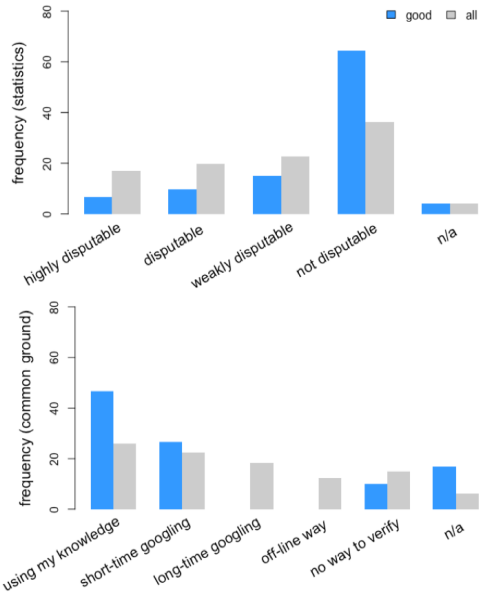


Figure 4: Relative occurrence frequency (unit: %)

low. The alphas of *good squared* annotations and *good* annotations are comparable to those of in-house annotations by Yang et al. (2019), which is the basis for the design of the reason options (see Section 5.1), showed IAA of 0.24 alpha. Wachsmuth et al. (2017) reported that the IAA of the document-level local acceptability annotations was measured between the most agreeing annotator pair among the three annotators and that the result was 0.46. Besides, seven experts who participated in their pilot study are found to show the IAA lower than 0.22 on pilot data, which is comparable to the alpha of our *good squared* and *good* annotations.

We also note that the sample size (the number of annotations) of each of the *good squared*, *good*, *moderate*, and *bad* annotations is different. To make sure that the IAA of *good squared* and *good* annotations is not the result of a different sample size, we randomly sampled 10%, 20%, ..., and 90% of all the annotations. In other words, we adopted random sampling as a baseline method to compare with our proposed method of the two-step reason selection, and checked the effect of the sample size on IAA. For each of the groups of randomly selected annotations with a different sample size, the IAA was less than 0.03 alpha for all the attributes. This confirms that the IAA of *good squared* and *good* annotations is not due to a different sample size of the annotations.

#### 6.2.3 Occurrence Frequency

Figure 2 shows the occurrence frequency of the attribute values for each attribute. It is notable that for local acceptability, the imbalance of the data was quite significant. The occurrence frequencies of *accept* and *strong accept* were much higher than those of the other labels. The *bad* annotations contain fewer *n/a*'s than *all* annotations, or the entire crowdsourced dataset. We find that the *bad* annotations have a more balanced distribution of the attribute values selected for each attribute. We believe that this is an indicator for more randomized selections in the *bad* annotations.

We also looked into the relative frequency of the attribute values for the sentences that contain a discourse unit with a strategy label. We report two cases of *Common Ground* and *Statistics* for the attributes of verifiability and disputability, respectively. Figure 4 shows the relative frequency of the attribute values for the two strategy labels. For *Common Ground*, the distribution of the attribute values for disputability of *good* annotations shows a higher bias to *not disputable* than those of *all* annotations. For *Statistics*, the distribution for verifiability for *good* annotations shows a higher bias to *using my knowledge* than those for *all* annotations. We believe that these biases match good linguistic intuition and that the higher biases of *good* annotations than *all* show the effectiveness of our method.

#### 6.2.4 Correlation Analysis

In Yang et al. (2019), we report that our in-house annotation results show a positive correlation between local acceptability and verifiability, and a negative correlation between local acceptability and disputability. Moreover, knowledge awareness and verifiability show a positive correlation. We see that the correlation pattern is an expected result, based on the definitions of the attributes. Figure 5 shows the Pearson's Correlation Coefficient (PC) between the four attributes for our annotation results using the two-step reason selection. *Good squared* and *good* annotations show the distinctive correlation pattern that is similar to that reported in Yang et al. (2019). *Moderate* and *all* annotations on the other hand show an opaque correlation pattern, and *bad* annotations show no correlation pattern except for a small negative correlation between local acceptability and disputability. We believe that the correlation patterns of the *good squared* and *good* annotations

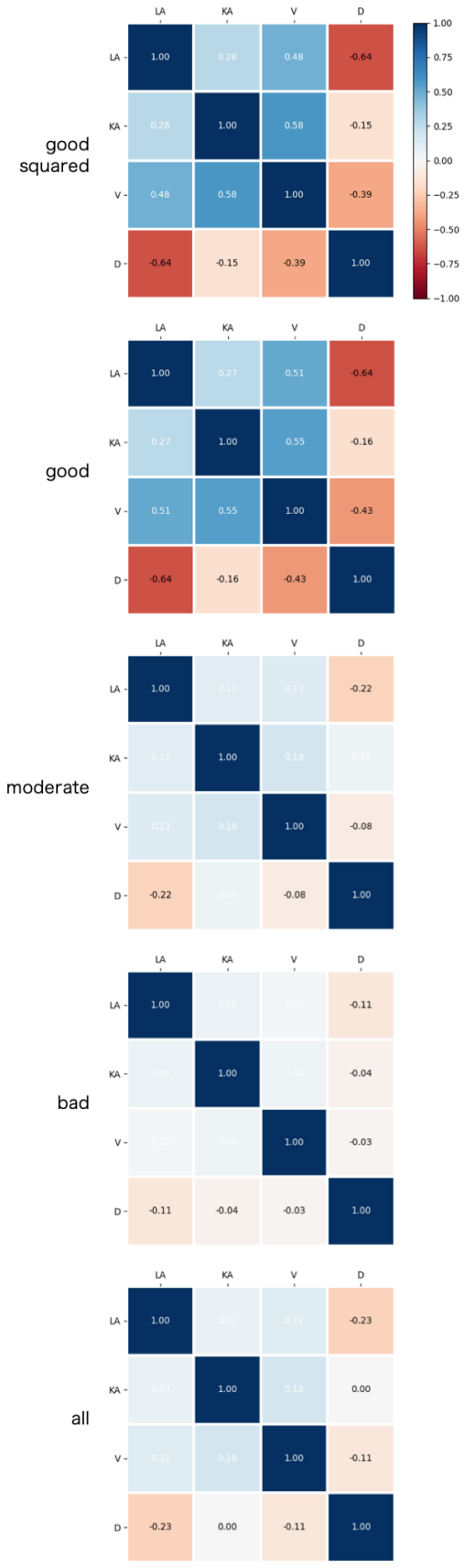


Figure 5: Pearson's Correlation Coefficient between local acceptability and the related attributes.

match good linguistic intuition, and that those of the other annotation groups do not. We also believe that the difference in the correlation pattern between the annotation groups shows the effectiveness of our quality control method using the two-step reason selection.

## 7 Conclusion and Future Work

In this study, we presented a simple but powerful method of quality control for annotating local acceptability and the related attributes using two-step reason selection. We designed a system in which a nonsensical reason can be selected within a sentential annotation and regarded it as an indicator for the unreliability of the annotation. The analysis of inter-annotator agreement shows that the method is effective at retaining only the annotations with remarkable quality. We conducted our annotation through crowdsourcing *without* the assurance of the expertise of the participating workers in the linguistic annotation, and *without* a separate training phase. Yet, the *good squared* and *good* annotations identified by our method show comparable quality to our previous in-house annotation results which were performed *with* the required expertise and *with* the training for linguistic annotation (Yang et al., 2019). Therefore, we believe that our results show a great potential for the crowdsourced annotation, especially when it is followed by a working quality control method.

One limitation of this study is that the annotators should not know about our use of the nonsensical reason selection as the indicator for unreliability during the annotation. As such, we paid attention not to inform the workers of this and did not reject workers for their selections of the nonsensical reasons. Therefore, we leave it as future work to develop a method that is effective even when workers are aware of quality control methods that employ nonsensical reasons, possibly by making the nonsensical reasons change automatically between annotations.

## Acknowledgements

This work was supported by Institute for Information and communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No. 2018-0-00582-002, Prediction and augmentation of the credibility distribution via linguistic analysis and automated evidence document collection).

## References


Azad Abad, Moin Nabi, and Alessandro Moschitti. 2017. Self-crowdsourcing training for relation extraction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 518–523.

Tali Aharoni and Keren Tenenboim-Weinblatt. 2019. Unpacking journalists'(dis) trust: Expressions of suspicion in the narratives of journalists covering the israeli-palestinian conflict. *The International Journal of Press/Politics*.

Khalid Al-Khatib, Henning Wachsmuth, Johannes Kiesel, Matthias Hagen, and Benno Stein. 2016. A news editorial corpus for mining argumentation strategies. In *Proceedings of the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3433–3443.

David Cheruiyot and Raul Ferrer-Conill. 2018. "fact-checking africa" epistemologies, data and the expansion of journalistic discourse. *Digital Journalism*, 6(8):964–975.

Haibo Ding, Tianyu Jiang, and Ellen Riloff. 2018. Why is an event affective? classifying affective events based on human needs. In *Workshops at the 32nd AAAI Conference on Artificial Intelligence*.

Bruno Guillaume, Karën Fort, and Nicolas Lefebvre. 2016. Crowdsourcing complex language resources: Playing to annotate dependency syntax. In *Proceedings of the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3041–3052.

Ivan Habernal and Iryna Gurevych. 2016. What makes a convincing argument? empirical analysis and detecting attributes of convincingness in web argumentation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1214–1223.

Charles L. Hamblin. 1970. Fallacies. *Londres, Methuen*.

Kazi S. Hasan and Vincent Ng. 2014. Why are you taking this stance? identifying and classifying reasons in ideological debates. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 751–762.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. Learning whom to trust with MACE. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1120–1130.

Matthew Lease. 2011. On quality control and machine learning in crowdsourcing. In *Workshops at the 35th AAAI Conference on Artificial Intelligence*.

David Lewis. 1970. General semantics. *Synthese*, 22(1):18–67.

Qiang Liu, Jian Peng, and Alexander Ihler. 2012. Variational inference for crowdsourcing. In *Proceedings of the 25th International Conference on Neural Information Processing Systems: Volume 1*, pages 692–700.

Nitika Mathur, Timothy Baldwin, and Trevor Cohn. 2018. Towards efficient machine translation evaluation by modelling annotators. In *Proceedings of the Australasian Language Technology Association Workshop 2018*, pages 77–82.

An T. Nguyen, Byron C. Wallace, Junyi J. Li, Ani Nenkova, and Matthew Lease. 2017. Aggregating and predicting sequence labels from crowd annotations. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 299–309.

Vikas C. Raykar, Shipeng Yu, Linda H. Zhao, Gerardo H. Valadez, Charles Florin, Luca Bogoni, and Linda Moy. 2010. Learning from crowds. *Journal of Machine Learning Research*, 11:1297–1322.

John Skorupski. 2002. The ontology of reasons. *Topoi*, 21(1):113–124.

Christian Stab and Iryna Gurevych. 2017. Parsing argumentation structures in persuasive essays. *Computational Linguistics*, 43(3):619–659.

Henning Wachsmuth, Nona Naderi, Yufang Hou, Yonatan Bilu, Vinodkumar Prabhakaran, Tim A. Thijm, Graeme Hirst, and Benno Stein. 2017. Computational argumentation quality assessment in natural language. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 176–187.

Wonsuk Yang, Jung-Ho Kim, Seungwon Yoon, Chae-Hun Park, and Jong C. Park. 2019. A corpus of sentence-level annotations of local acceptability with reasons. In *Proceedings of the 33rd Pacific Asia Conference on Language, Information and Computation*.

# Evaluating Pronominal Anaphora in Machine Translation: An Evaluation Measure and a Test Suite

**Prathyusha Jwalapuram**[*], **Shafiq Joty**[*§], **Irina Temnikova**[†], **and Preslav Nakov**[‡]
[*]Nanyang Technological University, Singapore
[§]Salesforce Research Asia, Singapore
[†]Sofia University, Bulgaria
[‡]Qatar Computing Research Institute, Doha, Qatar
[*]{jwal0001,srjoty}@ntu.edu.sg
[†]irina.temnikova@gmail.com
[‡]pnakov@hbku.edu.qa

## Abstract

The neural revolution in machine translation has made it easier to model larger contexts beyond the sentence-level, which can potentially help resolve some discourse-level ambiguities such as pronominal anaphora, thus enabling better translations. Unfortunately, even when the resulting improvements are seen as substantial by humans, they remain virtually unnoticed by traditional automatic evaluation measures such as BLEU, as only a few words end up being affected. Thus, specialized evaluation measures are needed. With this aim in mind, we contribute an extensive, targeted dataset that can be used as a test suite for pronoun translation, covering multiple source languages and different pronoun errors drawn from real system translations, for English. We further propose an evaluation measure to differentiate good and bad pronoun translations. Finally, we conduct a user study and we report correlation with human judgments.

## 1 Introduction

Traditionally, machine translation (MT) has been performed at the level of individual sentences, i.e., in isolation from the rest of the document. This was due to the nature of the underlying frameworks: word-based (Brown et al., 1993), then phrase-based (Koehn et al., 2003), syntactic (Galley et al., 2004), and hierarchical (Chiang, 2005). While there have been attempts to model the context beyond the sentence level, e.g., looking at neighboring sentences (Carpuat and Wu, 2007; Chan et al., 2007) or even at the entire document (Hardmeier et al., 2012), these approaches were still limited by the underlying framework.

Then, along came the neural revolution. Thanks to the attention mechanism, neural translation models such as sequence-to-sequence (Bahdanau et al., 2015) and the Transformer (Vaswani et al., 2017) could model much broader context. While initially translation was still done in a sentence-by-sentence fashion, researchers soon realized that going beyond that has become easier, and recent work has successfully exploited this (Bawden et al., 2018; Voita et al., 2018). This is an exciting research direction as it can help address inter-sentential phenomena such as anaphora, gender agreement, lexical consistency, and text coherence, to mention just a few.

Unfortunately, going beyond the sentence level typically yields very few changes in the translation output, and even when these changes are seen as substantial by humans, they remain virtually unnoticed by typical MT evaluation measures such as BLEU (Papineni et al., 2002), which are known to be notoriously problematic for the evaluation of discourse-level aspects in MT (Hardmeier, 2014).

The limitations of BLEU are well-known and have been discussed in detail in a recent study (Reiter, 2018). It has long been argued that as the quality of machine translation improves, there will be a singularity moment when existing evaluation measures would be unable to tell whether a given output was produced by a human or by a machine. Indeed, there have been recent claims that human parity has already been achieved (Hassan et al., 2018), but it has also been shown that it is easy to tell apart a human translation from a machine output when going beyond the sentence level (Läubli et al., 2018). Overall, it is clear that there is a need for machine translation evaluation measures that look beyond the sentence level, and thus can better appreciate the improvements that a discourse-aware MT system could potentially bring.

Alternatively, one could use diagnostic test sets that are designed to evaluate how an MT system handles specific discourse phenomena (Bawden et al., 2018; Rios et al., 2018). There have also been proposals to use semi-automatic measures and test suites (Guillou and Hardmeier, 2018).



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2964–2975, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

Here we propose a targeted dataset for machine translation evaluation with a focus on anaphora. We further present a specialized evaluation measure trained on this dataset. The measure performs pairwise evaluations: it learns to distinguish good vs. bad translations of pronouns, without being given specific signals of the errors. It has been argued that pairwise evaluation is useful and sufficient for machine translation evaluation (Guzmán et al., 2015, 2017). In particular, Duh (2008) has shown that ranking-based evaluation measures can achieve higher correlations with human judgments, as ranking judgments are easier to obtain from human judges and are also easy to use in training, while also directly achieving the purpose of comparing two systems.

Note that while it may be possible to rank translations using strong pre-trained conditional language models such as GPT (Radford et al., 2018), all kinds of errors would influence the score, and it would not be targeted towards a specific source of error, such as anaphora here. Our model provides a way to do this, and we demonstrate that it indeed focuses on the translation of pronouns.

Although our pronoun test suite naturally consists of the source text paired with a reference human translation, our pronoun evaluation measure is generally independent of the source language. Moreover, we use real machine translation output, which may contain various types of errors. Our contributions are as follows:

- We create a dataset for pronoun translation that covers multiple source languages and various target English pronouns.
- We propose a novel evaluation measure that differentiates good pronoun translations from bad ones irrespective of the source language they were translated from.
- Unlike previous work, both the dataset and the model are based on actual system outputs.
- Our evaluation measure achieves high agreement with human judgments.

We make both the dataset and the evaluation measure publicly available at https://ntunlpsg.github.io/project/discomt/eval-anaphora/.

## 2 Related Work

Previous work on discourse-aware machine translation and MT evaluation has targeted a number of phenomena such as anaphora, gender agreement, lexical consistency, and coherence. In this work, we focus on pronoun translation.

Pronoun translation has been the target of a shared task at the DiscoMT and WMT workshops in 2015-2017 (Hardmeier et al., 2015; Guillou et al., 2016; Loáiciga et al., 2017). However, the focus was on cross-lingual pronoun prediction, which required choosing the correct pronouns in the context of an existing translation, i.e., this was not a realistic translation task. The 2015 edition of the task also featured a pronoun-focused translation task, which was like a normal MT task except that the evaluation focused on the pronouns only, and was performed manually. In contrast, we have a real MT evaluation setup, and we develop and use a fully automatic evaluation measure.

More recently, there has been a move towards using specialized test suites specifically designed to assess system quality for some fine-grained problematic categories, including pronoun translation. For example, the PROTEST test suite (Guillou and Hardmeier, 2016) comprises 250 pronoun tokens, used in a semi-automatic evaluation: the pronouns in the MT output and in the reference are compared automatically, but in case of no matches, manual evaluation was required. Moreover, no final aggregate score over all pronouns was produced. In contrast, we have a much larger test suite with a fully automatic measure.

Another semi-automatic system is described in (Guillou et al., 2018). It focused on just two pronouns, *it* and *they*, and was applied to a single language pair. In contrast, we have a fully automated evaluation measure, we handle many English pronouns, and we cover multiple source languages.

Bawden et al. (2018) presented hand-crafted discourse test sets to test a model's ability to exploit previous source and target sentences, based on 200 contrastive pairs, one with a correct and one with a wrong pronoun translation. This alleviates the need for an automatic evaluation measure as one can just count how many times the MT system has generated the correct pronoun. In contrast, we use texts from pre-existing MT evaluation datasets, we do not require them to be in contrastive pairs, and we have a fully automated evaluation measure; we also use larger datasets.

Müller et al. (2018) also used contrastive translation pairs, mined from a parallel corpus using automatic coreference-based mining of context, thus minimizing the risk of producing wrong contrastive examples that are both valid translations. Yet, they did not propose an evaluation measure.

Finally, there have been pronoun-focused automatic machine translation evaluation measures. Two important examples include APT (Miculicich Werlen and Popescu-Belis, 2017) and AutoPRF (Hardmeier and Federico, 2010). Both measures require alignments between the source, the reference and the system output texts for evaluating the pronoun translations. However, automatic alignments are noisy; Guillou and Hardmeier (2018) have shown that improvements using heuristics are not statistically significant. They also found low agreement between these measures and human judgments, primarily due to the possibility of many translation choices per pronoun. APT also uses a predetermined list of 'equivalent pronouns', obtained for specific pronouns based on a French grammar book and verified through probability counts. This list is used to weigh pronouns that are not exact matches, and the accuracy of the pronoun translations is calculated accordingly. Miculicich Werlen and Popescu-Belis (2017) collect such a list for English-French for the pronouns *it* and *they*. This limits the evaluation measure both in terms of the language and also of the pronouns it is applicable to. In contrast, our framework requires only two candidate translations of the same text as input for comparison: this could be a reference vs. a system translation, or a comparison between two candidate translations (see Section 5.5).

## 3 Dataset Generation

We automatically generated our dataset, which we used to build a pronoun test suite and to train a pronoun evaluation model. In order to avoid generating synthetic data that may not necessarily represent a difficult context (for an MT system to correctly translate the pronouns), we used data from actual system outputs submitted for the WMT translation tasks in 2011–2015 and 2017 (Callison-Burch et al., 2011, 2012; Bojar et al., 2013, 2014, 2015, 2017). Using such data means that what is essentially a conditional language model solution, such as the one used by Bawden et al. (2018), has already failed on these examples.

**Original French input**: ***Il*** *était créatif, généreux, drôle, affectueux et talentueux, et il va beaucoup* ***me*** *manquer.*

**Reference translation**: ***He*** *was creative, generous, funny, loving and talented, and* ***I*** *will miss him dearly.*

**MT system translation**: ***It*** *was creative, generous, funny, affectionate and talented, and* ***we*** *will greatly miss.*

**Generated noisy example 1**: ***It*** *was creative, generous, funny, loving and talented, and I will miss him dearly.*

**Generated noisy example 2**: *He was creative, generous, funny, loving and talented, and* ***we*** *will miss him dearly.*

Figure 1: Noisy examples generated by substituting an MT-generated pronoun in the reference translation.

In particular, we aligned the system outputs with the reference translation using an automatic alignment tool (Dyer et al., 2013), and we found examples in which the pronouns did not match the reference translation. This process yielded potentially noisy data, as the alignments are automatic and thus not always perfect.

### 3.1 User Study

In order to ensure that the mismatched pronouns are not equally good translations in the given context, we conducted a user study on a subset of the data. To focus the study on pronouns and to remove the influence of other MT errors, we generated a noisy candidate by replacing the correct pronoun in the reference with the aligned (potentially) incorrect pronoun from the system output. We did this for each differing pronoun, so that the difference between the reference and the noisy version is one pronoun only (see Figure 1).

Our goal was to find pronoun pairs, e.g., *he-it* in Figure 1, where there is high agreement that the reference is the correct translation, so that we can automatically classify it as a positive example and the MT output as a negative one. The study participants were fluent in English and were native speakers of Chinese, Russian, French, or German. They were shown the source and two candidate translations (the reference and its noisy version) in random order. The relevant sentence was shown in bold, with the pronoun highlighted. Two previous sentences were given as a context; see Figure 2.

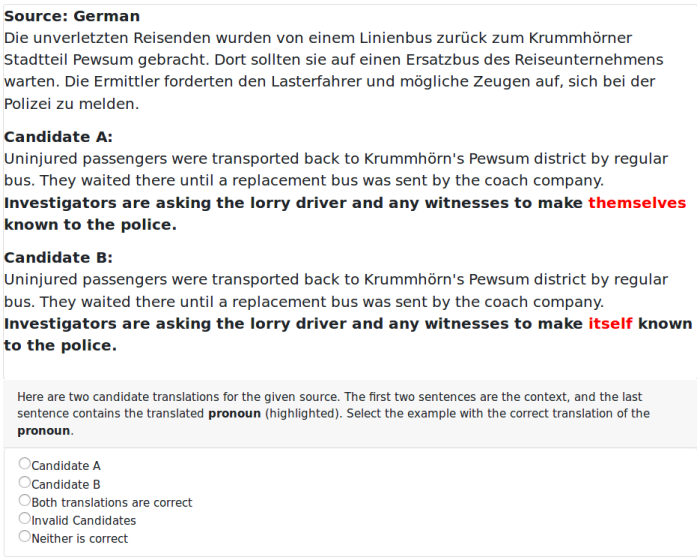


Figure 2: Our annotation framework.

| Language Pair | Nb. of Ann. | Avg% Ref | AC1 (Incl. Ties) | AC1 (Excl. Ties) |
|---|---|---|---|---|
| Russian→English | 3 | 80.2 | 0.82 | 0.92 |
| French→English | 2 | 83.9 | 0.86 | 0.96 |
| German→English | 2 | 84.3 | 0.89 | 0.97 |
| Chinese→English | 3 | 86.0 | 0.86 | 0.91 |
| - - Only English | 3 | 85.3 | 0.84 | 0.92 |

Table 1: Inter-annotator agreement.

We asked the study participants to choose the text with the better pronoun. They were allowed to choose (*a*) candidate A, (*b*) candidate B, (*c*) equivalent translations (tie), (*d*) "neither is correct", and (*e*) "invalid candidates", i.e., the highlighted words are not pronouns or are the wrong pronouns due to misalignment. We excluded from further consideration all examples marked as (*d*) or (*e*). Each participant annotated a total of **500 examples** per language pair. Statistics[1] about the annotation process are given in Table 1. We also report the proportion of cases where the participants preferred the reference translation over the noisy version (see **Avg%Ref**).[2] We can see that there is high agreement for all language pairs, ranging from 0.82 to 0.89. The ties seem to be the major source of disagreement: excluding them yields agreements in the range of 0.91–0.97.

[1]Due to the nature of the dataset, the human annotators are always more likely to choose the reference as the better candidate, which yields a skewed distribution of the annotations; traditional correlation measures such as Cohen's kappa are not robust to this, and thus we report the more appropriate Gwet's AC1/gamma coefficient.(Gwet, 2008).

[2]High agreement could also mean that the participants consistently picked the noisy version as the better choice.

In order to measure the effect, if any, that the source text might have on the annotator's choices, we also conducted a study without providing them the source text. We did this for the texts from the Chinese→English study. The participants were only shown the English texts: the reference vs. the noisy sentence, with the context as before. The results for this study are shown on the last line of Table 1. We can see that the agreement for this English-only setup is also fairly high (0.84); the overall agreement between all six participants (three from Chinese→English and three from only English) is 0.85. We observe very similar agreement of 0.91 (Chinese→English) and 0.92 (Only English) when the ties were excluded, with the overall agreement being 0.90 between the six participants. However, further analysis showed that although the two groups disagreed on about 10% of the examples, only 2% of the examples were common to both groups, showing that the sources of disagreement between the two groups are different. It can be that having the source context helps disambiguate the other 8% of the cases, while also introducing ambiguity that does not seem to be an issue for the participants who saw the English texts only. See Figure 1 for an example where the source is helpful; here, noisy example 2 would be acceptable, except that the original French text uses a singular pronoun. However, the disagreements form a small part of the dataset; we also filtered out all pronoun pairs with low agreements from further use. Figure 3 shows one such low-agreement example.

**Source Text**: 不过现在，她只想享受当下。我不想说这是我的最后一场比赛。这会给我带来太大的压力。

**Reference translation**: *For now she just wants to enjoy the moment. I didn't want to say **this** was my last race. That would have meant too much pressure.*

**Noisy candidate**: *For now she just wants to enjoy the moment. I didn't want to say **that** was my last race. That would have meant too much pressure.*

Figure 3: Low-agreement example: Chinese-English.

### 3.2 Pronoun Test Suite for MT Systems

The source sentences can also be used as a **test suite** for MT systems to check their pronoun translations: it can be considered to be a challenging, diagnostic test set for pronoun translation, covering a range of errors such as gender (he/she→it), number (they→it), animacy (who→which), syntactic role (e.g., subject/object: he→him), and others; see the Appendix for a complete list.

The WMT test sets come from news articles; the context is available, and thus the test suite is particularly suitable for discourse-level MT systems. It is available for each source language for which English translations are generated in the WMT tasks, e.g., Czech, French, German, etc. (Table 2).

The corresponding noisy versions of the reference are also generated, but are somewhat noisy. However, the subset used in our study is curated in some sense as human judgments are available. This data can serve as a more refined test suite: not only useful for checking the agreement with human judgments, but also for identifying equivalent pronoun translations in context, as the data is also annotated for ties.

| Source Language | Test Data from WMT Years | Unique Source Contexts |
|---|---|---|
| German | 2011-2015,17 | 7,823 |
| Czech | 2011-2015,2017 | 6,713 |
| French | 2011-2015 | 4,659 |
| Russian | 2013,2014,2017 | 4,513 |
| Spanish | 2011-2013 | 4,417 |
| Finnish | 2015,2017 | 1,551 |
| Turkish | 2017 | 1,372 |
| Hindi | 2014 | 921 |
| Chinese | 2017 | 696 |
| Latvian | 2017 | 652 |

Table 2: Pronoun test suite for MT systems: English as a target and various languages as a source.

## 4 The Evaluation Measure

While *diagnostic datasets* allow us to evaluate MT systems with respect to specific discourse-level phenomena, an automatic discourse-aware *evaluation measure* is useful not only for evaluation but also for tuning MT systems. Moreover, an evaluation measure that only looks at the target language (which is computationally feasible, even if not ideal, as our study above has shown) offers additional benefits; we can train it for a specific target language without requiring a separate dataset for each source-target language pair. Below, we propose such a measure for pronoun translation.

Let $R = (C_r; r)$ and $S = (C_s; s)$ denote a *reference* and a *system* tuple pair containing a reference and a system translation, $r$ and $s$, along with a context of previous sentences, $C_r$ and $C_s$, respectively. Note that $C_r$ and $C_s$ can contain the same sentences or different sentences, or be empty in case no context is provided. Given a training set $\mathcal{D} = \{(R_i, S_i)\}_{i=1}^{N}$ containing $N$ such tuple pairs, our aim is to learn an evaluation measure that can rank any unseen translation pair $(R, S)$ with respect to the correct use of pronouns.[3] In Section 3, we described how such datasets can be collected opportunistically without recourse to expensive manual annotation.

Figure 4 shows our proposed framework to evaluate MT outputs with respect to pronouns. The inputs to the model are sentences (with or without context $C_r$ and $C_s$): $R$ and $S$. Each input sentence is first mapped into a set of word embedding vectors of dimensionality $d$ by performing a lookup in the shared embedding matrix $E \in \mathbb{R}^{v \times d}$ with vocabulary size $v$. $E$ can be initialized randomly or with any pre-trained embeddings such as GloVe (Pennington et al., 2014), or contextualized word vectors such as ELMo (Peters et al., 2018a).

In case of initialization with GloVe vectors, we use a BiLSTM (Hochreiter and Schmidhuber, 1997) layer to get a representation of the words that is encoded with contextual information. Let $X = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ denote an input sequence, where $\mathbf{x}_t$ is the $t^{th}$ word embedding vector of the sequence. The LSTM recurrent layer computes a compositional representation $\mathbf{k}_t$ at every time step $t$ by performing nonlinear transformations of the current input $\mathbf{x}_t$ and the output of the previous time step $\mathbf{k}_{t-1}$.

[3] Here $R$ (or $S$) can be a reference or a system translation.

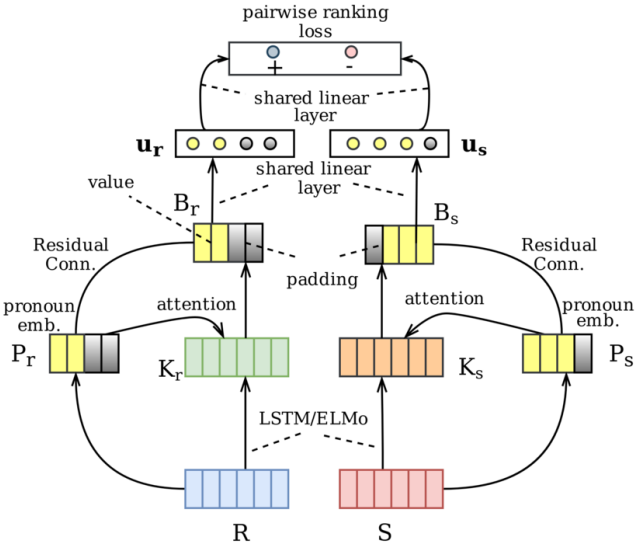


Figure 4: Our proposed framework to differentiate good pronoun translations from bad ones in context.

In a BiLSTM, we get the representation $\overrightarrow{\mathbf{k}_t}$ by processing the sequence in the forward direction, and the representation $\overleftarrow{\mathbf{k}_t}$ by processing the sequence in the backward direction. The final representation $\mathbf{k}_t$ of a word is the concatenation of these two representations, i.e., $\mathbf{k}_t = [\overrightarrow{\mathbf{k}_t}; \overleftarrow{\mathbf{k}_t}]$.

With ELMo initialization, the word vectors obtained are used directly. ELMo uses stacked biLSTM encoder and gives very powerful contextualized word representations learned from large corpora by optimizing a bi-directional language modeling loss. The ELMo representations already capture morphological, syntactic and contextual semantic features (Peters et al., 2018b).[4]

Let $K_r$ and $K_s$ be the matrices whose rows represent the word representations of $R$ and $S$, respectively (obtained either from Bi-LSTM or directly from ELMo). From these representations, we extract the representations of the pronouns in the target sentence (from $r$ and $s$; not from the contexts). Let $P_r$ and $P_s$ be the matrices whose rows represent the contextualized word representations of the pronouns in $r$ and $s$, respectively. We use zero-padding (shown as shaded boxes) to make $P_r$ and $P_s$ fixed-length. We then use *scaled multiplicative attention* (Vaswani et al., 2017) to compute a contextual representation for the pronouns in $r$ and $s$. Specifically, we consider the rows of $P_r$ (*resp.* $P_s$) as *query* vectors, the rows of $K_r$ (*resp.* $K_s$) as *key* and *value* vectors, and the matrix $P_r$ (*resp.* $P_s$) to attend over $K_r$ (*resp.* $K_s$).

[4]We also tried an ELMo-initialized BiLSTM, but it did not perform well while increasing model complexity.

We use residual connection and layer normalization to get pronoun representations $B_r$ and $B_s$:

$$B'_r = \mathcal{S}(\frac{P_r K_r^T}{\sqrt[2]{d}})K_r; \quad B'_s = \mathcal{S}(\frac{P_s K_s^T}{\sqrt[2]{d}})K_s \quad (1)$$

$$B_r = \text{LayerNor}(P_r + B'_r); \; B_s = \text{LayerNor}(P_s + B'_s) \quad (2)$$

Note that $B_r$ and $B_s$ contain a $d$-dimensional vector for each query (pronoun) vector (and *zero* vectors due to padding). We pass these vectors through a **shared** linear layer parameterized by $\mathbf{z} \in \mathbb{R}^d$ to obtain a score for each pronoun. This yields vectors $\mathbf{u}_r$ and $\mathbf{u}_s$ for the reference and for the system translations:

$$\mathbf{u}_r = B_r\mathbf{z}; \quad \mathbf{u}_s = B_s\mathbf{z}; \quad (3)$$

A final **shared** linear layer parameterized by $\mathbf{w}$ converts these vectors to contrastive scores, yielding a (positive) score $y_r$ for the reference and a (negative) score $y_s$ for the system translation:

$$y_r = \mathbf{u}_r^T\mathbf{w}; \quad y_s = \mathbf{u}_s^T\mathbf{w}; \quad (4)$$

We then use the scores in a pairwise ranking loss (Collobert et al., 2011) to find model parameters that assign a higher score to $y_r$ than to $y_s$. We minimize the following ranking objective:

$$\mathcal{L}(\theta) = \max\{0, 0.1 - y_r + y_s\} \quad (5)$$

Note that the network shares all of its parameters ($\theta$) to obtain $y_r$ and $y_s$ from a pair of inputs $R_i = (C_r, r)$ and $S_i = (C_s, s)$. Once trained, it can be used to score any input independently.

## 5 Experiments

Below, we describe our data, the experimental setup, and the evaluation results.

### 5.1 Data

We first created a set of commonly confused pronoun pairs. Using the data from the study, we calculated the inter-annotator agreement for each pair of a reference/correct pronoun and a system translation/incorrect pronoun. We excluded the pairs with low agreement (<0.8) or for which the system output was chosen as the correct translation more often. Pairs with low agreement are essentially cases where the annotators cannot agree that the reference translation is better. The source of ambiguity in this case is that the system translation is not absolutely wrong (see Figure 3); therefore, these cases may not be so critical to correct. The remaining pairs are, with a fairly high confidence, positive–negative (correct–incorrect) pairs.

Next, we filtered the WMT data, only keeping sentences with these pronoun pairs. This yielded 97,461 reference translation (positive text) — unique system output (negative text) pairs for training, taken from WMT11,12,13,15 (Table 3).

The development data collected from WMT14 system outputs has 5,727 unique system translations and 6,635 unique noisy candidate pairs.[5]

For testing, we used the annotated data from the user study, generated from a subset of WMT17 system translations (except for French, which is from the discussion forum test set from WMT15, not overlapping with the training data). There are 500 unique noisy-reference pairs per source language, a total of 2,000.

| Data | Source | #Unique Pairs |
|---|---|---|
| Training | WMT11-13,15 | 97,461 |
| Development | WMT14 | 5,727 |
| Development (noisy) | WMT14 | 6,635 |
| Test (noisy) | WMT15,17 | 2,000 |

Table 3: Statistics about our dataset.

[5]The number of unique noisy candidates exceeds that of unique system translations because a separate noisy candidate was generated for each error in a system translation.

### 5.2 Experimental Setup

We evaluated the models in terms of accuracy, i.e., the proportion of times the model scored the reference translation higher than the system/noisy output, and we report results using either GloVe (Pennington et al., 2014) or ELMo (Peters et al., 2018a). We conducted a number of experiments, training and testing under different conditions:

**No Context (NC).** The reference ($R$) (or the noisy reference $R'$) and the system ($S$) translations go through ELMo or BiLSTM, without contextual information $C_r/C_s$ (i.e., query = $P_r/P_s$, key, value = $R = r/S = s$);

**With Context:** The reference ($R$) (or the noisy reference $R'$) and the system ($S$) translation representations include two previous sentences as a context. The context can be further categorized as

(i) **Respective Context (RC)**: $R$ includes its own reference context $C_r = r_{-2}r_{-1}$, and $S$ includes its own respective system context $C_s = s_{-2}s_{-1}$ (query = $P_r/P_s$, key, value = $R = r_{-2}r_{-1}r/S = s_{-2}s_{-1}s$);

(ii) **Common Reference Context (CRC)**: The context for $R$ and $S$ includes the same reference context $C_r = C_s = r_{-2}r_{-1}$ for both (query = $P_r/P_s$, key, value = $R = r_{-2}r_{-1}r/S = r_{-2}r_{-1}s$);

We perform the evaluation in two ways:

(a) $\mathbf{R}$ vs. $\mathbf{S}$: Testing over pairs of reference ($R$) and system translation ($S$) texts;

(b) $\mathbf{R}$ vs. $\mathbf{R}'$: Testing over pairs of reference ($R$) and noisy candidate ($R'$) texts.

**Baseline.** For a baseline performance, we simply take the average of the extracted pronoun representations in $P_r$ and $P_s$, and we convert them to pairwise scores through linear layers. The baseline is also evaluated with and without context.

| Exp | Context Setting | Test | Acc. (Glove) | Acc. (ELMo) |
|---|---|---|---|---|
| 1 | NC-Baseline | $\mathbf{R}$ vs. $\mathbf{R}'$ | 69.12 | 85.80 |
| 2 | NC | $\mathbf{R}$ vs. $\mathbf{R}'$ | 68.97 | 88.04 |
| 3 | NC | $\mathbf{R}$ vs. $\mathbf{S}$ | 79.67 | 89.09 |
| 4 | RC-Baseline | $\mathbf{R}$ vs. $\mathbf{R}'$ | 69.07 | 85.80 |
| 5 | RC | $\mathbf{R}$ vs. $\mathbf{R}'$ | 67.88 | 87.90 |
| 6 | CRC-Baseline | $\mathbf{R}$ vs. $\mathbf{R}'$ | 69.16 | 86.66 |
| 7 | CRC | $\mathbf{R}$ vs. $\mathbf{R}'$ | 68.93 | 89.11 |
| 8 | CRC | $\mathbf{R}$ vs. $\mathbf{S}$ | 77.87 | 90.69 |

Table 4: Results on WMT14 with different contexts.

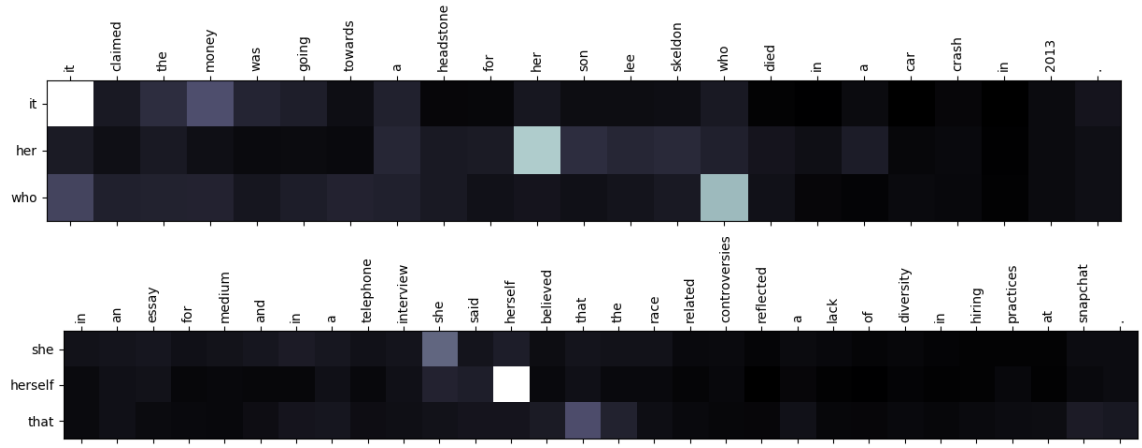


Figure 5: Attention maps for some noisy candidates from the test set.

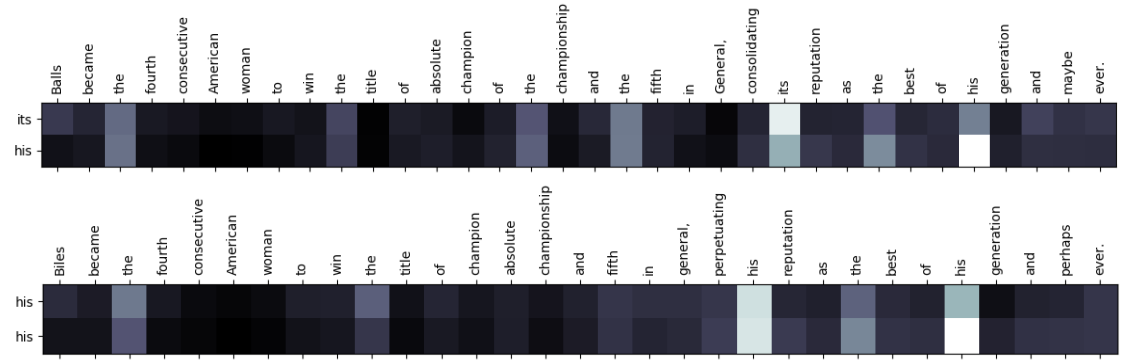


Figure 6: Attention maps for two candidate system translations of the same text (best viewed in color)

### 5.3 Results and Analysis

The first three experiments in Table 4 show results for the 'No Context' setting. These are on the noisy data (Exp. 1-2) and are indicative of how sensitive our model is to pronouns, since $R$ and $R'$ differ in a single pronoun. The results on the reference/system translation pair $R$ and $S$ (Exp. 3) is indicative of the performance in a real use case. In the case of the GloVe-BiLSTM model, using attention instead of averaging pronouns does not help. However, using ELMo greatly improves the accuracy of the baseline and also yields an improvement when using attention. ELMo can model syntactic and semantic information and can thus improve co-reference resolution (Peters et al., 2018a), which could be a contributing factor.

The second part of our experiments (Exp. 4-5) concerns the addition of contextual information. When the respective contexts are added, there is no improvement (even a drop in case of GloVe-BiLSTM); quite possibly, other differences in the text make it harder for the model to focus on the pronoun errors.

To offset this issue, we use a common context for both the reference and the system sentences, taken from the reference (Exp. 6-8). Training the model with a common reference context (CRC) yields marginal difference in the GloVe-BiLSTM model, but the ELMo model improves with the addition of context. Our experiments[6] show that ELMo is quite powerful at capturing contextual information, even as the context size grows.

Finally, we test our *ELMo - common reference context* model on the held-out dataset used for the user study. Table 5 shows the results. The accuracy of the system is lower on the study dataset. Note that since the training data was filtered based on the pronoun pairs with high agreement from the study, the study dataset contains pronoun pairs with low agreement that were not seen during training. We also calculate the agreement with human judgments excluding ties since our model does not handle ties. The overall agreement with the human judgments remains high.

[6]We also ran experiments with BERT (Devlin et al., 2019) and found that it performed on par with ELMo.

| Language | Acc.(ELMo) | AC1 Agr. |
|---|---|---|
| Russian→English | 79.4 | 0.80 |
| French→English | 82.0 | 0.84 |
| German→English | 81.6 | 0.83 |
| Chinese→English | 82.4 | 0.83 |
| - - Only English | —- | 0.83 |
| **Overall (average)** | 81.35 | —- |

Table 5: **Acc**uracy and AC1 **Agr**eement for the ELMo-based model predictions on the study dataset.

### 5.4 Pronoun-wise Analysis

We performed a pronoun-wise analysis of the results in Table 5. The model scored the noisy version higher than the reference in about 19% of the cases. Of these, 46% were pronoun pairs that were not seen during training.

Of the remaining pronoun pairs that were seen during training, the main source of errors (over 11%) were cases when ***that*** in the reference was replaced by ***it*** in the noisy version (*"We always tell victims not to pay up;* ***that/it*** *simply exacerbates the problem", explains Kleczynski*). The noisy candidate was scored higher about 28% of the time, out of 79 samples. The next highest source of errors was the reference–noisy pair of ***it–she*** at 10%.

In contrast, the best-performing pair was when a ***he*** in the reference was replaced with an ***it*** in the noisy version (***He/It*** *risked everything to save other people's lives.*). The reference was scored higher than the noisy candidate 95% of the time, out of 135 samples. The next highest performer was the reference–noisy pair of ***his–its***, which was correctly scored 86% of the time.

This performance follows the distribution of the pronoun pairs seen during training. The ***he–it*** and ***his–its*** pairs together account for over 12% of the training data, while the ***that–it*** and ***it–she*** pairs together form only 3.7%. Since the distribution of the pronoun pairs in the training data is itself based on the distribution of errors in the system translations, the model performs best over error cases that occur most often in system translations.

### 5.5 Discussion

As systems participating in WMT improve over the years, our test data is closer to state-of-the-art neural MT, while most of the training data is from statistical systems. This setting allows us to capture a wider range of errors while showing that our model is sensitive to small errors in a fluent output. Note that since the model was trained on the full system output and also on all pronouns in that output, it has not received any signal about which pronoun is wrong. Yet, we can see from the attention maps in Figure 5 that the model can correctly identify the incorrect pronoun. In Figure 5 (top), the model distinguishes the wrong pronoun *it* (the correct one is *she*), while in Figure 5 (bottom) it correctly finds *herself* as the wrong translation (the correct one is *she*).

We further compare the scores of two system translations for the same sentence from WMT17 Russian-English system outputs (see Figure 6). Here, the correct pronoun to be found is *her*. While one system translates it alternately as *its* and *his* (see Figure 6, top), the other system translates both cases as *his* (see Figure 6, bottom). Our model scores the translation in Figure 6 (bottom) higher than the translation in Figure 6 (top); even though the model highlights both occurrences of *his* as wrong, it ends up believing that having *its* as a translation is worse. We could argue that the translation in Figure 6 (bottom) is better since it maintains the animacy/human aspect, even if the grammatical gender is wrong; moreover, this translation is also consistent.

One could further argue that pronoun-focused automatic machine translation evaluation measures such as APT and AutoPRF are likely to yield the same accuracy/precision-recall for both cases above.

## 6 Conclusions and Future Work

We have presented a new, extensive, targeted dataset for pronoun translation that covers multiple source languages and a wide range of target English pronouns. We have also proposed a novel evaluation measure for differentiating good vs. bad pronoun translations, irrespective of the source language, which achieved high correlation with human judgments.

In future work, we want to handle cases where multiple pronouns are equally suitable in a given context. We would also like to extend the work to other discourse phenomena.

## Acknowledgments

We would like to thank the anonymous reviewers for their comments. Shafiq Joty would also like to thank the funding support from his Start-up Grant (M4082038.020).

## References


Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the Third International Conference on Learning Representations*, ICLR '15, San Diego, CA, USA.

Rachel Bawden, Rico Sennrich, Alexandra Birch, and Barry Haddow. 2018. Evaluating discourse phenomena in neural machine translation. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT '18, pages 1304–1313, New Orleans, LA, USA.

Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, WMT '14, pages 12–58, Baltimore, MD, USA.

Ondřej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2013. Findings of the 2013 workshop on statistical machine translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, WMT '13, pages 1–44, Sofia, Bulgaria.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. 2017. Findings of the 2017 conference on machine translation. In *Proceedings of the Second Conference on Machine Translation*, WMT '17, pages 169–214, Copenhagen, Denmark.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi. 2015. Findings of the 2015 workshop on statistical machine translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, WMT '15, pages 1–46, Lisbon, Portugal.

Peter E. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2).

Chris Callison-Burch, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2012. Findings of the 2012 workshop on statistical machine translation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, WMT '12, pages 10–51, Montréal, Canada.

Chris Callison-Burch, Philipp Koehn, Christof Monz, and Omar Zaidan. 2011. Findings of the 2011 workshop on statistical machine translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, WMT '11, pages 22–64, Edinburgh, Scotland, UK.

Marine Carpuat and Dekai Wu. 2007. Improving statistical machine translation using word sense disambiguation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '07, pages 61–72, Prague, Czech Republic.

Yee Seng Chan, Hwee Tou Ng, and David Chiang. 2007. Word sense disambiguation improves statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, ACL '07, pages 33–40, Prague, Czech Republic.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 263–270, Ann Arbor, MI, USA.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT '19, pages 4171–4186, Minneapolis, MN, USA.

Kevin Duh. 2008. Ranking vs. regression in machine translation evaluation. In *Proceedings of the Third Workshop on Statistical Machine Translation*, StatMT '08, pages 191–194, Columbus, OH, USA.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648, Atlanta, GA, USA.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL-HLT '04, pages 273–280, Boston, MA, USA.

Liane Guillou and Christian Hardmeier. 2016. PROTEST: A test suite for evaluating pronouns in machine translation. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation*, LREC '16, pages 636–643, Portorož, Slovenia.

Liane Guillou and Christian Hardmeier. 2018. Automatic reference-based evaluation of pronoun translation misses the point. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, EMNLP '18, pages 4797–4802, Brussels, Belgium.

Liane Guillou, Christian Hardmeier, Ekaterina Lapshinova-Koltunski, and Sharid Loáiciga. 2018. A pronoun test suite evaluation of the English–German MT systems at WMT 2018. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, WMT '18, pages 570–577, Brussels, Belgium.

Liane Guillou, Christian Hardmeier, Preslav Nakov, Sara Stymne, Jörg Tiedemann, Yannick Versley, Mauro Cettolo, Bonnie Webber, and Andrei Popescu-Belis. 2016. Findings of the 2016 WMT shared task on cross-lingual pronoun prediction. In *Proceedings of the First Conference on Machine Translation*, WMT '16, pages 525–542, Berlin, Germany.

Francisco Guzmán, Shafiq Joty, Lluís Màrquez, and Preslav Nakov. 2015. Pairwise neural machine translation evaluation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, ACL-IJCNLP '15, pages 805–814, Beijing, China.

Francisco Guzmán, Shafiq R. Joty, Lluís Màrquez i Villodre, and Preslav Nakov. 2017. Machine translation evaluation with neural networks. *Computer Speech & Language*, 45:180–200.

Kilem Li Gwet. 2008. Computing inter-rater reliability and its variance in the presence of high agreement. *British Journal of Mathematical and Statistical Psychology*, 61(1):29–48.

Christian Hardmeier. 2014. *Discourse in Statistical Machine Translation*. Ph.D. thesis, Uppsala University, Department of Linguistics and Philology, Uppsala, Sweden.

Christian Hardmeier and Marcello Federico. 2010. Modelling pronominal anaphora in statistical machine translation. In *Proceedings of the 2010 International Workshop on Spoken Language Translation*, IWSLT '10, pages 283–289, Paris, France.

Christian Hardmeier, Preslav Nakov, Sara Stymne, Jörg Tiedemann, Yannick Versley, and Mauro Cettolo. 2015. Pronoun-focused MT and cross-lingual pronoun prediction: Findings of the 2015 DiscoMT shared task on pronoun translation. In *Proceedings of the Second Workshop on Discourse in Machine Translation*, pages 1–16, Lisbon, Portugal.

Christian Hardmeier, Joakim Nivre, and Jörg Tiedemann. 2012. Document-wide decoding for phrase-based statistical machine translation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1179–1190, Jeju Island, Korea.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, Shujie Liu, Tie-Yan Liu, Renqian Luo, Arul Menezes, Tao Qin, Frank Seide, Xu Tan, Fei Tian, Lijun Wu, Shuangzhi Wu, Yingce Xia, Dongdong Zhang, Zhirui Zhang, and Ming Zhou. 2018. Achieving human parity on automatic Chinese to English news translation. *CoRR*, abs/1803.05567.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, NAACL-HLT '03, pages 48–54, Edmonton, Canada.

Samuel Läubli, Rico Sennrich, and Martin Volk. 2018. Has machine translation achieved human parity? A case for document-level evaluation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, EMNLP '18, pages 4791–4796, Brussels, Belgium.

Sharid Loáiciga, Sara Stymne, Preslav Nakov, Christian Hardmeier, Jörg Tiedemann, Mauro Cettolo, and Yannick Versley. 2017. Findings of the 2017 DiscoMT shared task on cross-lingual pronoun prediction. In *Proceedings of the Third Workshop on Discourse in Machine Translation*, DiscoMT '17, pages 1–16, Copenhagen, Denmark.

Lesly Miculicich Werlen and Andrei Popescu-Belis. 2017. Validation of an automatic metric for the accuracy of pronoun translation (APT). In *Proceedings of the Third Workshop on Discourse in Machine Translation*, WMT '17, pages 17–25, Copenhagen, Denmark.

Mathias Müller, Annette Rios, Elena Voita, and Rico Sennrich. 2018. A large-scale test set for the evaluation of context-aware pronoun translation in neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, WMT '18, pages 61–72, Belgium, Brussels.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the*

*40th Annual Meeting of the Association for Computational Linguistics*, ACL '12, pages 311–318, Philadelphia, PA, USA.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, EMNLP '14, pages 1532–1543, Doha, Qatar.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018a. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT'18, pages 2227–2237, New Orleans, LA, USA.

Matthew Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. 2018b. Dissecting contextual word embeddings: Architecture and representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, EMNLP '18, pages 1499–1509, Brussels, Belgium.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. Technical report, OpenAI.

Ehud Reiter. 2018. A structured review of the validity of BLEU. *Computational Linguistics*, 44(3):393–401.

Annette Rios, Mathias Müller, and Rico Sennrich. 2018. The word sense disambiguation test suite at WMT18. In *Proceedings of the Third Conference on Machine Translation*, WMT '18, pages 594–602, Belgium, Brussels.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, NIPS '17, pages 5998–6008.

Elena Voita, Pavel Serdyukov, Rico Sennrich, and Ivan Titov. 2018. Context-aware neural machine translation learns anaphora resolution. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, ACL '18, pages 1264–1274, Melbourne, Australia.

# A Regularization Approach for Incorporating Event Knowledge and Coreference Relations into Neural Discourse Parsing

**Zeyu Dai, Ruihong Huang**
Department of Computer Science and Engineering
Texas A&M University
{jzdaizeyu, huangrh}@tamu.edu

## Abstract

We argue that external commonsense knowledge and linguistic constraints need to be incorporated into neural network models for mitigating data sparsity issues and further improving the performance of discourse parsing. Realizing that external knowledge and linguistic constraints may not always apply in understanding a particular context, we propose a regularization approach that *tightly* integrates these constraints with contexts for deriving word representations. Meanwhile, it *balances* attentions over contexts and constraints through adding a regularization term into the objective function. Experiments show that our knowledge regularization approach outperforms all previous systems on the benchmark dataset PDTB for discourse parsing.

## 1 Introduction

Discourse parsing and identifying rhetorical discourse relations between two text spans (i.e., discourse units, either clauses or sentences) is crucial and beneficial for a wide variety of downstream tasks and applications such as machine translation (Webber et al., 2017), text generation (Mann, 1984; Bosselut et al., 2018) and text summarization (Gerani et al., 2014).

In the PDTB-style discourse parsing (Prasad et al., 2008), we commonly distinguish implicit discourse relations from explicit relations, depending on whether a discourse connective (e.g., "because", "however") appears between two discourse units. In general, recognizing implicit discourse relations is more challenging due to the lack of connective, which has recently drawn significant attention from the NLP researchers.

Recent research for implicit discourse relation classification has mostly focused on applying powerful neural network models (Qin et al., 2016a,b; Liu and Li, 2016; Lei et al., 2017; Bai and Zhao, 2018) for modeling compositional meanings and word-level interactions of two discourse units. More recent research has also exploited utilizing broader contexts (Dai and Huang, 2018) as well as leveraging external training data (Xu et al., 2018). Although progress has been made, the performance of implicit discourse relation identification remains low (macro F1 $< 50\%$).

We believe that the low performance is mainly due to the data sparsity issue (Braud and Denis, 2015), which hinders data-thirsty neural network models from making further improvements. Considering the following example from PDTB with two discourse units (DUs):

***DU1*:** The editorial of the WHO notes that tobacco consumption and **lung-cancer** mortality rates are rising in developing countries.

***DU2*:** "No **smoking** should be established as the norm of social behavior" around the world, the editorial says, through the enactment of laws that limit advertising and promote anti-smoking education.

***Discourse Relation*:** Implicit Contingency.**Cause**

Humans can easily recognize this discourse relation as "Cause" because we know that "smoking" is the key causal factor for "lung-cancer", but it is extremely difficult for neural network models trained with limited amount of data to detect it considering the keyword "lung-cancer" only appears few times in the whole PDTB data.

We further argue that external knowledge and linguistic constraints need to be considered for improving implicit discourse relation classification since human annotators also rely on these commonsense knowledge (e.g., smoking causes the lung-cancer) to label the discourse relations. *First*, we consider external event knowledge, because discourse relations (e.g., cause and temporal relations) are often defined as the relation between two



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2976–2987,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

events (situations in general) as described in two discourse units. As shown in the above example, the "Cause" discourse relation between the two DUs depends on the relation between two events *"smoking"* and *"lung-cancer"* with one event in each DU. *Second*, we consider entity coreference relations as a useful form of linguistic constraints in inferring discourse relations. This is motivated by prior work (Rutherford and Xue, 2014; Ji and Eisenstein, 2015) showing that coreference based features can improve entity mention representations within a DU, which facilitates recognizing coherence and discourse relations between DUs.

In this paper, we investigate how to incorporate external event knowledge and entity coreference relation based linguistic constraints into neural network models for discourse parsing. One key difficulty we want to address is that external knowledge derived event relations or hard linguistic constraints may not always apply for interpreting a particular context, and may hurt performance if used blindly (Kishimoto et al., 2018). Therefore, we propose to *tightly* integrate these constraints into the discourse relation inference process by manipulating hidden word representations to reflect relations between words, and meanwhile *balance* attentions to contexts and constraints through adding a knowledge regularization term in the final objective function.

Specifically, we choose the paragraph-level model we proposed (Dai and Huang, 2018) as the base model, which exploits wider paragraph-level contexts and has been shown effective for PDTB-style discourse parsing. The model mainly consists of a two-level hierarchical BiLSTMs (Schuster and Paliwal, 1997) for modeling both word-level and DU-level inter-dependencies (with a brief description in section 3.1). To implement the knowledge guided regularization for discourse parsing, we *first* insert a new knowledge layer between the word-level BiLSTM and DU-level BiLSTM layer. This knowledge layer modifies hidden representations of words that participate in an event or coreference relation, by applying a relation type specific feedforward neural network. *Then*, we compose a knowledge regularizer based on word representation outputs of the knowledge layer, by adapting a classic knowledge embedding method TransE (Bordes et al., 2013). The regularization term is added to the overall objective function and minimized during model training.

The experiments on PDTB v2.0 demonstrate that our proposed knowledge regularization approach can effectively utilize several types of externally obtained event knowledge and entity coreference relations[1], and improves the performance of both implicit and explicit discourse relation recognition compared to all previous work.

## 2 Related Work

### 2.1 Discourse Parsing on PDTB

With the release of Penn Discourse Treebank (PDTB) (Prasad et al., 2008), the task of discourse parsing, especially implicit discourse relation recognition, has received a lot of attention from the NLP community and researchers (Pitler and Nenkova, 2009; Lin et al., 2014; Xue et al., 2015; Rutherford and Xue, 2016). A large number of previous work attempted to model the semantic meanings of two discourse units using latest and advanced neural network models (Chen et al., 2016; Ji et al., 2016; Rutherford et al., 2017; Qin et al., 2017; Guo et al., 2018; Bai and Zhao, 2018). Paragraph-wide contexts were considered for building better discourse unit representations in Dai and Huang (2018). Another research direction for improving implicit discourse relation classification is to expand the training data by leveraging explicit relations (Liu et al., 2016; Lan et al., 2017) or discourse connective informed unlabeled data (Rutherford and Xue, 2015; Xu et al., 2018).

### 2.2 Incorporate Knowledge into Discourse Parsing

Only a few previous work (Park and Cardie, 2012; Biran and McKeown, 2013; Lei et al., 2018) has exploited external knowledge, including WordNet features (e.g., Antonyms and Hypernyms) and Verb Class (Levin, 1993), in discourse parsing by deriving discrete indicator features and then feed them into feature-based classifiers. Incorporating knowledge as additional features into neural network models often generalize poorly due to the sparsity of features, as also shown in our experiments. Recently, Kishimoto et al. (2018) incorporated the whole of ConceptNet into a MAGE-GRU (Dhingra et al., 2017) based neural networks, but their experiments show that it did not work well for improving implicit discourse relation identification compared with their own base-

[1]Entity coreference relations were generated using an existing coreference resolver from Standford CoreNLP toolkit.

line. We interpret this negative result as the consequence of using irrelevant (noisy) knowledge types blindly without proper regularization.

There are also recent work (Yang and Mitchell, 2017; Xu et al., 2017; Zhou et al., 2018) that incorporate external knowledge into neural network models for improving several other NLP tasks, including information extraction and conversation generation, which mostly followed the *two-step* approach that first obtained representations of knowledge (with triplet format) from knowledge base using knowledge graph embedding methods such as TransE (Bordes et al., 2013), and then utilized attention mechanism (or added gates in a RNN cell (Ma et al., 2018)) to integrate knowledge representations with hidden word vectors. This approach has two main drawbacks: (1) Knowledge representations learned from the first step are fixed without considering the influences of contexts, which may be suboptimal when used for understanding a particular context. (2) With no filtering or regularization, it is difficult for attention mechanisms to explicitly select and attend to the relevant knowledge. In contrast, our proposed regularization approach can be regarded as an end-to-end *joint-learning* framework for discourse parsing and knowledge representation learning, which not only considers both knowledge and contexts in knowledge-aware word representation learning, but also naturally balances attentions on both contexts and knowledge through regularization.

## 3 Model

Figure 1 illustrates the overall architecture of our model, which implements our knowledge regularization approach (the right part) on top of an existing model as the base model (the left part). There are only two modifications we made to the base model: (1) we insert a novel knowledge layer between the two BiLSTM layers of the base mode; (2) we add a regularizer into the overall objective function. We will first briefly describe the base model, a replication[2] of our recently proposed paragraph-level discourse parsing model (Dai and Huang, 2018). We will then explain the knowledge layer and knowledge regularizer we added.

### 3.1 Base Model

The base model processes a paragraph containing a sequence of discourse units each time, and predicts a sequence of discourse relations (both implicit and explicit relations) with one relation between each pair of adjacent discourse units (DU). The base model utilizes a hierarchical BiLSTM to calculate both word-level and DU-level representations, followed by a prediction layer and Conditional Random Field (CRF) layer (Lafferty et al., 2001) for jointly predicting a sequence of discourse relations within a paragraph. The base model consists of the following layers:

**Character-level CNN Layer:** The character-level features, such as the prefix or suffix of a word, can help alleviate the out-of-vocabulary (OOV) problem and improve the word representation in neural nets (Santos and Zadrozny, 2014). In our base model, we use one layer of CNN[3] with max-pooling to extract character-level representation $\boldsymbol{w}_i^{char}$ for the $i$-th word of the input paragraph.

**Word-level BiLSTM Layer:** Given a words sequence $X = (x_1, x_2, ..., x_L)$ as the input paragraph, for each word $x_i$, we construct the expanded word vector by concatenating its word embedding $\boldsymbol{w}_i^{word}$ with its character-level representation and extra word-level features[4] as:

$$\boldsymbol{w}_i = [\boldsymbol{w}_i^{word}; \boldsymbol{w}_i^{char}; \boldsymbol{w}_i^{features}]$$

The word-level BiLSTM layer will process the sequence of expanded word vectors $(\boldsymbol{w}_1, \boldsymbol{w}_2, ..., \boldsymbol{w}_L)$ and compute the word $x_i$'s hidden representation at each word index $i$:

$$\boldsymbol{h}_{x_i} = BiLSTM(\boldsymbol{w}_1, \boldsymbol{w}_2, ..., \boldsymbol{w}_L)$$

**DU-level BiLSTM Layer:** Given the output of word-level BiLSTM $(\boldsymbol{h}_{x_1}, \boldsymbol{h}_{x_2}, ..., \boldsymbol{h}_{x_L})$, we calculate the raw DU representation by applying max-pooling operation (Conneau et al., 2017) over the sequence of word representations for all words within a discourse unit: $\boldsymbol{h}'_{DU_j} = \max_{x_i \in DU_j} \boldsymbol{h}_{x_i}$

Then, the DU-level BiLSTM will process the sequence of raw DU representations and obtain the refined DU representation $\boldsymbol{h}_{DU_j}$ for the $j$-th discourse unit in a paragraph:

$$\boldsymbol{h}_{DU_j} = BiLSTM(\boldsymbol{h}'_{DU_1}, \boldsymbol{h}'_{DU_2}, ..., \boldsymbol{h}'_{DU_{T+1}})$$

[2] In our re-implementation, we made several minor modifications to the original base model by using character-level features as well as supporting both traditional fixed word embeddings (300D GloVe (Pennington et al., 2014)) and latest context-dependent word embeddings (1024D ELMo (Peters et al., 2018)) for word embedding initialization.

[3] Both character embedding and CNN hidden size is 50.

[4] In this work, we used capitalization (Cap) flag, Part-of-speech (POS) tag and named entity (NER) tag of each word as extra word-level features. The embedding size for Cap/POS/NER is 5/35/20. We used Standford CoreNLP toolkit (Manning et al., 2014) to generate POS and NER tags.

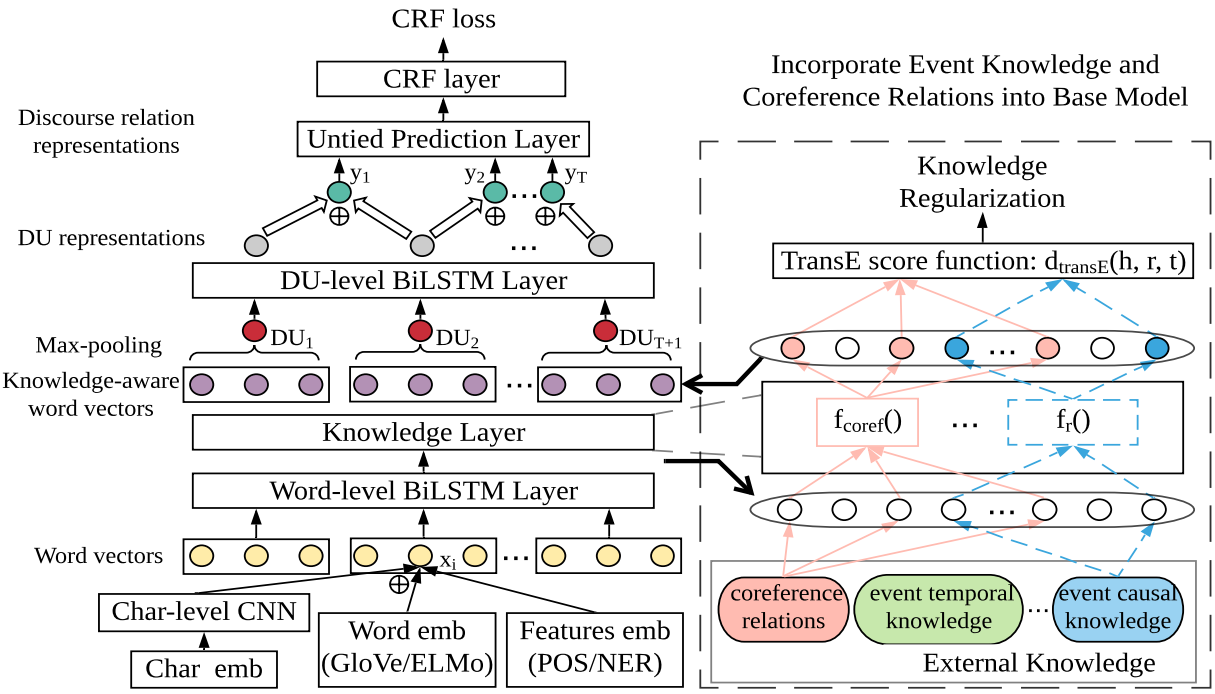


Figure 1: Model Architecture for Paragraph-level Discourse Parsing. The left part is the base model. The right part with colored arrows and neurons show how to incorporate coreference and event knowledge into base model.

**Untied (Explicit vs. Implicit) Prediction Layer:** Considering the different natures of explicit and implicit discourse relations (Pitler et al., 2009; Lin et al., 2009), the base model trains two independent linear layers with untied parameters for predicting explicit or implicit discourse relations between each two adjacent DUs respectively:

$$\boldsymbol{h}_{y_t} = \begin{cases} W_{exp}[\boldsymbol{h}_{DU_t}; \boldsymbol{h}_{DU_{t+1}}] + b_{exp}, & \text{if } y_t \in exp \\ W_{imp}[\boldsymbol{h}_{DU_t}; \boldsymbol{h}_{DU_{t+1}}] + b_{imp}, & \text{if } y_t \in imp \end{cases}$$

**CRF Layer for Discourse Relation Sequence Labeling:** A CRF layer (Biran and McKeown, 2015) is added on top of the prediction layer to fine-tune the predicted sequence of discourse relations by capturing continuity and transition patterns (e.g., a temporal relation is likely to follow another temporal relation).

Given the hidden discourse relation representations $\boldsymbol{H}^{(i)} = (\boldsymbol{h}^{(i)}_{y_1}, \boldsymbol{h}^{(i)}_{y_2}, ..., \boldsymbol{h}^{(i)}_{y_T})$ and the target discourse relation label sequence $\boldsymbol{y}^{(i)} = (y^{(i)}_1, y^{(i)}_2, ..., y^{(i)}_T)$ for the $i$-th training instance, we minimize the following CRF loss function during model training:

$$L_{CRF} = -\sum_i \log p(\boldsymbol{y}^{(i)}|\boldsymbol{H}^{(i)})$$

During testing, the Viterbi algorithm is used to search for the optimal label sequence $\boldsymbol{y}^*$ that maximizes the conditional probability $p(\boldsymbol{y}|\boldsymbol{H})$.

### 3.2 Knowledge Layer

We simply insert a knowledge layer between the word-level and DU-level BiLSTM layers of the base model, as shown in Figure 1, for incorporating external knowledge and linguistic constraints. Although the knowledge layer can be easily extended to support other types of knowledge, we only consider event knowledge and coreference relations in this paper and leave the exploration of other knowledge types in the future work.

Since there are some notable differences between event relations and entity coreference relations, we model event and coreference constraints in different ways considering their specificities. For example, (1) an event relation can be represented as the triple format $((h, r, t)$ or $(head, relation, tail)$ where $head$ and $tail$ are two event words, $relation$ indicates the relationship between two events $head$ and $tail$.) while a coreference relation can have more than two coreferential entity mentions; (2) event relations are directed while coreference relations are undirected.

**Event Knowledge:** As the input of our knowledge layer, $\mathbb{E} = (\mathbb{E}_1, \mathbb{E}_2, ..., \mathbb{E}_M)$ denotes the collection of event knowledge triplets generated by matching the paragraph contexts with external event knowledge base (we will give more details in the following section 4.1.), where each triplet has the form of $(h, r, t)$ meaning that there is an event

relation $r$ (either *temporal*, *causal* or *subevent* in this work) between the head event $x_h$ at the position $h$ and tail event $x_t$ at the position $t$.

For each triplet $\mathbb{E}_m = (h, r, t)$, we use a feedforward neural network[5] $f_r()$ to update the hidden word representations of head and tail events:

$$f_r(\boldsymbol{h}_{x_h}); f_r(\boldsymbol{h}_{x_t}) = \tanh(\boldsymbol{W}_r[\boldsymbol{h}_{x_h}; \boldsymbol{h}_{x_t}] + \boldsymbol{b}_r)$$

where $\boldsymbol{W}_r$ and $\boldsymbol{b}_r$ are relation-specific weights and bias learned for each type of event relation $r$ *only*.

**Coreference Relations:** Our system assumes that coreference relations in each paragraph are given in the form of coreference clusters, which are generated by running an existing coreference resolver (Clark and Manning, 2016) from the latest version (3.9.2) of Stanford CoreNLP toolkit.

Let $\mathbb{C} = (\mathbb{C}_1, \mathbb{C}_2, ..., \mathbb{C}_K)$ denote coreference clusters in one paragraph, where $\mathbb{C}_k$ contains the word indices with corresponding words referring to the same entity. Similar as above, we use one feedforward neural network $f_{coref}()$ to update the hidden word representation $\boldsymbol{h}_{x_i}$ for words within each coreference cluster. Specifically, the output word vector has the following form:

$$f_{coref}(\boldsymbol{h}_{x_i}) = \begin{cases} \tanh(\boldsymbol{W}_{coref}[\boldsymbol{h}_{x_i}; \boldsymbol{h}_{\mathbb{C}_k}] + \boldsymbol{b}_{coref}), & \text{if } x_i \in \mathbb{C}_k \\ \boldsymbol{h}_{x_i}, & otherwise \end{cases}$$

where $\boldsymbol{W}_{coref}$ and $\boldsymbol{b}_{coref}$ are the weights and bias, $\boldsymbol{h}_{\mathbb{C}_k}$ is a coreference vector calculated by applying max-pooling to all word representations in one cluster: $\boldsymbol{h}_{\mathbb{C}_k} = \max_{x_i \in \mathbb{C}_k} \boldsymbol{h}_{x_i}$. The role of coreference vector is similar to "context vector" utilized in soft attention mechanism (Bahdanau et al., 2015), but we use simple max-pooling instead of computing weights[6] for different word vectors.

### 3.3 Knowledge Regularization

Inspired by the success of TransE (Bordes et al., 2013) approach in knowledge representation learning, we adapt the key assumption of TransE (i.e., we want $\boldsymbol{h} + \boldsymbol{r} \approx \boldsymbol{t}$ when $(h, r, t)$ holds.) to our framework and hypothesize that the hidden representation of tail $t$ should be close to the hidden representation of head $h$ plus the relation-specific vector $\boldsymbol{h}_r$ in vector space if $(h, r, t)$ holds.

To guide the knowledge-aware word representation learning of the knowledge layer, we propose a knowledge regularizer based on TransE[7] score function $d_{transE}()$ and apply it to the output word vectors of the knowledge layer. The resulting regularization term is also minimized as a part of the objective function during model training. In other words, the knowledge regularization will smoothly penalize constraint depending on whether this constraint can be applied for interpreting relevant relation in a particular context.

Specifically, we use cosine similarity[8] to measure the similarity of two vectors, so the score function for triplet $(h, r, t)$ has the following form:

$$d_{transE}(h, r, t) = 1 - cos(f_r(\boldsymbol{h}_{x_h}) + \boldsymbol{h}_r, f_r(\boldsymbol{h}_{x_t}))$$

where $\boldsymbol{h}_r$ is the relation-specific vector which will be updated as parameters during model training. The event knowledge regularization term is:

$$R_{event} = \sum_{\mathbb{E}_m = (h,r,t) \in \mathbb{E}} d_{transE}(h, r, t)$$

For coreference relations, we create a special triplet $(h, coref, t)$ for each two entity mentions $h$ and $t$ in one coreference cluster, and fix the relation-specific vector $\boldsymbol{h}_{coref}$ to be a zero vector representing the relation of being "identical". The coreference relation regularization term is:

$$R_{coref} = \sum_{\mathbb{C}_k \in \mathbb{C}} \sum_{(h,t) \in \mathbb{C}_k} d_{transE}(h, coref, t)$$

Hence, the overall loss function for our model is:

$$L = L_{CRF} + \lambda_{event} * R_{event} + \lambda_{coref} * R_{coref}$$

## 4 Experiments

### 4.1 Dataset and Preprocessing

**Dataset:** We evaluate our model on PDTB v2.0 (Prasad et al., 2008), which is the largest annotated dataset containing 19K explicit discourse relations and 17K implicit discourse relations. To make our experimental results directly comparable with previous work, we adopted the most-used dataset splitting "PDTB-Ji" (Ji and Eisenstein, 2015) that uses sections 2-20, 0-1, and 21-22

[5] We also tried to use more complicated neural nets including neural tensor networks (Socher et al., 2013) and self-attentions mechanism (Vaswani et al., 2017), but none of them performed better than straightforward feedforward neural network. We even tried to not update (identical function) hidden word vectors, but it performed significantly worse.

[6] We tried to employ weights with soft-attention mechanism, but it did not show improvement in our experiments.

[7] We also tried TransD (Ji et al., 2015) and TransR (Lin et al., 2015) for knowledge regularization, but none of them showed clear improvement over TransE in our experiments.

[8] Note that cosine similarity performed better than L1 or L2 distance in our experiments.

| Relation Type | Source | # | Discourse |
|---|---|---|---|
| Coreference | CoreNLP | 19,819 | Exp & Cont |
| Event Temporal | Yao et al. | 18,515 | Temp |
| Event Causal | ConceptNet | 947 | Cont |
| Event Subevent | ConceptNet | 626 | Exp & Temp |

Table 1: Overview of relation types. # is the number of matched triplets (clusters for coreference). The last column summarizes relevant discourse relation classes each knowledge type may potentially help identify.

as train, dev and test sets respectively. To recover the paragraph contexts and gold discourse units, we directly ran the source code[9] of Dai and Huang (2018), and obtained 12,037/1222/1050 paragraph instances in train/dev/test sets respectively.

**Knowledge Preprocessing:** Table 1 gives an overview of the relation types used in our experiments and the number of triplets (clusters) identified in the PDTB dataset. Specifically, for coreference relations, we utilized the Stanford CoreNLP coreference resolver to identify coreference clusters in each paragraph. For event knowledge, we considered three major event relation types including temporal, causal and subevent. We obtained event temporal knowledge from a previous work (Yao and Huang, 2018)[10] and we retrieved the latter two types of event knowledge from ConceptNet[11] (Speer and Havasi, 2012), which is a widely-used commonsense knowledge base.

### 4.2 Experiment Setting

**Evaluation Setting:** Annotated discourse relation labels in PDTB v2.0 are organized in a three-level hierarchy. The top-level coarse-grained discourse relation classes include Comparison (Comp), Contingency (Cont), Expansion (Exp) and Temporal (Temp), which are further split into 16 fine-grained classes at the second-level. To compare with previous work, we report the macro-average F1-score and accuracy[12] on the top-level multi-class classification setting. Note that the macro-average F1-score is normally treated as the main evaluation metric in most previous work considering the imbalanced distribution of discourse relations. In addition, we report class-wise F1-scores for the top-level implicit discourse relations. But different from many previous work that report class-wise F1-scores obtained by using the one-versus-all binary classification setting, we instead report class-wise F1-scores using the 4-way multi-class classification setting, following Dai and Huang (2018) which pointed out that compared to the one-versus-all binary classification setting where all binary classifiers may predict a positive label for one instance, the multi-class classification setting is more appropriate in evaluating a practical end-to-end discourse parser without the need of prediction conflict resolution. We additionally evaluate our models at the second-level using the 11-way[13] multi-class classification.

**Training Setting:** To make it easy for model tuning, we only chose $\lambda_{coref}$ and $\lambda_{event}$ from [0.1, 0.5, 1.0] and tuned them based on the best performance on the dev set. All the BiLSTM layers and our knowledge layer used the hidden state size of 512, so the dimension of all hidden vectors ($\boldsymbol{h}_*$, $f_{coref}(\boldsymbol{h}_*)$ and $f_r(\boldsymbol{h}_*)$) is 512. To prevent gradient exploding problem of LSTMs, we clipped the gradient L2-norm with threshold 5.0 and used L2 regularization with coefficient $10^{-8}$. We applied dropout with probability 0.5 on the input/output of BiLSTMs to alleviate overfitting. For the optimizer, we used the SGD with momentum 0.9 and batch size of 64, and we set the initial learning rate as 0.015 which will decay by 5% after each epoch.

To diminish the effects of randomness in neural network model training, we ran all our proposed model, its variants as well as our own base model 3 times using different random seeds and reported the average performance over 3 runs. For fair comparison, we implemented all our models with Pytorch and tested them on a Nvidia 1080Ti GPU.

### 4.3 Models for Comparison

We compare our proposed regularization models with the following base model, our own baselines and recent published discourse parsing systems:

- (Dai and Huang, 2018): the original model for paragraph-level discourse parsing.

[9] Available at https://github.com/ZeyuDai/paragraph_implicit_discourse_relations. Relations (0.5%) between non-adjacent DUs were discarded.

[10] We also tried to use VerbOcean (Chklovski and Pantel, 2004) which performed worse than our choices.

[11] Available at http://conceptnet.io. To extract event causal knowledge, we merged the relations ['Causes', 'CausesDesire', 'Entails'] defined in ConceptNet. To extract subevent knowledge, we merged the relations ['HasSubevent', 'HasFirstSubevent', 'HasLastSubevent']. For simplicity, we removed relations containing multi-word events or non-event words (e.g., function words).

[12] Note that 3% discourse relations in PDTB were annotated with more than one label. Following previous work (Dai and Huang, 2018; Bai and Zhao, 2018), we considered a prediction as correct if it matches one of the gold labels.

[13] We followed Ji and Eisenstein (2015) to exclude 5 minor second-level classes in our experiments because none of these classes appear in the test or dev sets.

| Model | Implicit | | | | | | Explicit | |
|---|---|---|---|---|---|---|---|---|
| | Macro | Acc | Comp | Cont | Exp | Temp | Macro | Acc |
| Previous work with the same evaluation setting | | | | | | | | |
| (Rutherford and Xue, 2015) | 40.50 | 57.10 | - | - | - | - | - | - |
| (Liu et al., 2016) | 44.98 | 57.25 | - | - | - | - | - | - |
| (Liu and Li, 2016) | 46.29 | 57.57 | - | - | - | - | - | - |
| (Lan et al., 2017) | 47.80 | 57.39 | - | - | - | - | - | - |
| (Dai and Huang, 2018) | 48.82 | 57.44 | 37.72 | 49.39 | 67.45 | 40.70 | 93.21 | 93.98 |
| (Bai and Zhao, 2018) (ELMo) | 51.06 | - | - | - | - | - | - | - |
| Our models using GloVe word embeddings | | | | | | | | |
| Base Model | 48.96 | 56.42 | 41.29 | 47.77 | 66.16 | 40.60 | 93.78 | 94.64 |
| Base Model + Coreference (C) | 49.42 | 57.15 | 41.39 | 49.26 | 66.89 | 40.14 | 93.73 | 94.62 |
| Base Model + Event Temporal | 49.58 | 57.31 | 41.52 | 46.49 | 67.50 | 42.83 | 93.87 | 94.72 |
| Base Model + Event (E) | 50.02 | 58.22 | 40.20 | 48.06 | **68.35** | 43.45 | 93.63 | 94.46 |
| Full Model (Base Model + C&E) | **50.49** | **58.32** | 39.61 | **49.29** | 68.23 | **44.83** | **94.32** | **95.07** |
| Our own baselines with knowledge features using GloVe word embeddings | | | | | | | | |
| Base Model + Word Features | 49.22 | 57.33 | **41.72** | 48.30 | 67.01 | 39.88 | 94.05 | 94.88 |
| Base Model + DU Features | 49.27 | 57.55 | **41.72** | 48.01 | 67.36 | 39.98 | 93.98 | 94.82 |
| Base Model + two-step approach | 49.63 | 57.52 | 41.41 | 45.57 | 68.08 | 43.46 | 93.88 | 94.71 |
| Our models using ELMo[16] word embeddings | | | | | | | | |
| Base Model | 50.83 | 56.50 | **47.00** | 46.42 | 65.58 | 44.21 | 94.35 | 95.12 |
| Full Model (Base Model + C&E) | **52.89** | **59.66** | 45.34 | **51.80** | **68.50** | **45.93** | **94.84** | **95.39** |

Table 2: Top-level Multi-class Classification Results on PDTB. We report macro-average F1-score (Macro), accuracy (Acc) and F1-score on each class (Comparison, Contingency, Expansion and Temporal). Note that Bai and Zhao (2018) used ELMo word embeddings, so this result is only comparable with the last section.

- Base Model: our replicated model of (Dai and Huang, 2018) for paragraph-level discourse parsing.
- Base Model + Word Features: our own baseline that creates discrete features for each word. We create one feature for each type of relations, including three types of event relations and coreference relations, which counts the number of relation triplets that contain a word. We concatenate these word features with the input word vector $\boldsymbol{w}_i$.
- Base Model + DU Features: our own baseline that creates discrete features for each DU $DU_j$. We create two features for each type of relations: one counts relation triplets that have both nodes within $DU_j$; and the other counts relation triplets that have one node in $DU_j$ and the other node in an adjacent DU. We concatenate these DU features with the hidden DU representation $\boldsymbol{h}_{DU_j}$. Adding either word features or DU features is to imitate traditional feature-based approaches and incorporate event knowledge and coreference relation constraints as features.
- Base Model + two-step approach: our own baseline that follows the two-step approach for incorporating relational constraints, including both event relations and coreference relations. We re-implement the inference model proposed by Chen et al. (2018)[14], which first employs TransE to obtain relation representations, and then uses attention mechanism to incorporate relations and build knowledge-enhanced DU representations.
- (Rutherford and Xue, 2015): a feature-based classifier that utilizes explicit discourse connectives for creating more implicit relations.
- (Liu et al., 2016): CNN based multi-task joint learning model that leverages both PDTB and RST (Carlson et al., 2003) datasets.
- (Liu and Li, 2016): a hierarchical attention-over-attention neural network model for implicit discourse relation recognition.
- (Lan et al., 2017): multi-task attention-based model for predicting implicit discourse relations that leverages both explicit discourse relations of PDTB and unlabeled data.
- (Bai and Zhao, 2018): a complex deep residual bi-attention based neural network model for implicit discourse relation classification which improves the hidden representations at character, subword, word and DU levels.

### 4.4 Experiment Results

Table 2 shows the comparisons. The first section lists the results of previous models that were evaluated on PDTB using top-level multi-class classification. Note that many cells are empty, it is be-

[14]We followed Chen et al. (2018) to tune the hyper-parameter $\lambda$ for each relation type in the range [0.1, 0.5, 1, 5, 10, 20, 50, 100], and we found that the best result was achieved when $\lambda$ is set to 0.1 for each type of relations.

| | Implicit | | Explicit | |
|---|---|---|---|---|
| Model | Macro | Acc | Macro | Acc |
| Our models using GloVe word embeddings | | | | |
| Base Model | 28.59 | 43.88 | 63.40 | 86.02 |
| + Word Features | 30.83 | 44.17 | 65.47 | 86.60 |
| + DU Features | 31.78 | 44.65 | 66.06 | 86.51 |
| Full Model | **32.13** | **46.03** | **69.24** | **87.25** |
| Our models using ELMo word embeddings | | | | |
| Base Model | 31.05 | 45.98 | 70.01 | 87.83 |
| Full Model | **33.41** | **48.23** | **70.48** | **88.08** |

Table 3: Second-level Multi-class Classification Results on PDTB. We report macro-average F1-score (Macro) and accuracy (Acc).

cause that many previous publications chose to report the class-wise implicit relation prediction performance using the one-versus-all binary classification setting, which are not directly comparable with our class-wise results using the multi-class classification setting following our previous work Dai and Huang (2018). In addition, we also report the explicit relation results.

In the second section, the replicated model (Base Model) achieves an overall similar performance[15] compared to the original model of Dai and Huang (2018). By incorporating coreference relations (+ Coreference) into the base model using our regularization approach, implicit discourse relation prediction performance was improved for two classes, Contingency and Expansion. Adding event temporal knowledge (+ Event Temporal) into the base model significantly boosts the performance of Temporal discourse relation identification. Furthermore, adding the additional two types of event knowledge (+ Event), causal and subevent relations, yields clear performance gains in predicting another two classes of implicit discourse relations: Contingency and Expansion. These performance gains meet our expectations that event relations are correlated with discourse relations and event relational knowledge facilitates predicting corresponding discourse relations, with correspondences listed in Table 1. The full model considering both event knowledge and coreference relations (+ C&E) achieves further improvements on implicit relation prediction, and outperforms the base model by 1.5 and 1.9 points on macro-average F1-score and accuracy respectively. Meanwhile, our full model obtains the best results for explicit relation prediction as well.

Shown in the third section, our own baselines which incorporate relational constraints either as features or via the two-step approach only perform slightly better than the base model, but clearly worse than the full model using the regularization approach. The first two baselines incorporate constraints as additional discrete features and may suffer from feature sparsity issues, while the two-step approach may fail to balance attentions to contexts and knowledge constraints.

The last section presents the performance when using ELMo word embeddings. Our full model outperforms the base model on three out of four (except Comp) implicit discourse relations and improves both macro-average F1-score and accuracy by 2.1 and 3.2 points respectively. Furthermore, our full model outperforms the previous best system (Bai and Zhao, 2018) using ELMo by over 1.8 points of macro-average F1-score.

[16]We downloaded the pretrained ELMo embedding (5.5B version) from AllenAI's website (https://allennlp.org/elmo) and froze its parameters during model training.

[15]The performance changes on individual categories are due to minor modifications we made in replication, such as adding char-level CNN and replacing word2vec with GloVe.

| | Implicit | | Explicit | |
|---|---|---|---|---|
| Model | Macro | Acc | Macro | Acc |
| Our models using GloVe word embeddings | | | | |
| Full Model | 50.49 | 58.32 | 94.32 | 95.07 |
| w/o Reg | 44.28 | 55.43 | 92.56 | 93.59 |
| Our models using ELMo word embeddings | | | | |
| Full Model | 52.89 | 59.66 | 94.84 | 95.39 |
| w/o Reg | 48.55 | 56.45 | 93.77 | 94.65 |

Table 4: Impact of the Knowledge Regularization. We report the top-level performance when knowledge regularizer was removed (w/o Reg) from full model.

## 5 Analysis

### 5.1 Second-level Multi-class Classification

Table 3 reports the performance of our models for predicting second-level fine-grained discourse relations. Same as top-level, our full model consistently outperforms the base model and its variants using either word-level or DU-level features.

### 5.2 Impact of the Knowledge Regularization

To study the necessity of the knowledge regularization in our full model, we removed the regularization terms from the objective function by setting $\lambda_{coref}$ and $\lambda_{event}$ to be 0, which essentially means that we did not restrict or regulate the hidden knowledge-aware word vectors at all. From Table 4, we can see that the model without knowledge regularizer performs significantly worse than the full model and even worse than the base model, which supports our hypothesis that using external knowledge or linguistic constraints

| Model | # of errors |
|---|---|
| Base Model | 217 |
| Full Model | 193 |

Table 5: Number of errors made by both models for predicting implicit discourse relations with constraints.

blindly can hurt the performance. We conclude that the knowledge regularizer plays a key role in achieving the state-of-the-art performance and the knowledge layer must be used together with knowledge regularization in our framework.

### 5.3 Qualitative Analysis

To better understand the strengths and weaknesses of the regularization approach, we analyzed implicit discourse relation predictions made by our base model and full model on the dev set. In total, there are 507 implicit discourse relations that match with at least one event or coreference constraint across two DUs, while the remaining 717 instances do not involve those constraints. It turns out that both the full model and base model performed comparably on recognizing implicit discourse relations without event or coreference constraints, with 407 vs. 402 discourse relations correctly predicted by the full model and the base model respectively. Therefore, the overall performance gains achieved by the full model are mainly from better resolving implicit discourse relations with constraints, and as shown in Table 5, the full model made 24 less errors than base model for predicting such implicit discourse relations.

We further compared predictions made by the two models for implicit discourse relations with constraints. We found that 96 predictions in total have been changed by the full model, with clearly more corrections (60, i.e., the full model corrected predictions that were made wrongly by the base model.) than false reversions (36, i.e., the correct predictions made by the base model were wrongly reverted by the full model.). Here is one example from the 60 corrections made by the full model:

***DU1*:** Steve and his firm still **worth** a lot of money.

***DU2*:** A package of credit **support** was put together including the assets of Steve and his firm.

***Gold Discourse Relation*:** Implicit Contingency

***Base Model's prediction*:** Implicit Expansion

***Full Model's prediction*:** Implicit Contingency

The event causal relation between "worth" and "support" identified using external event knowledge has enabled the full model to correctly recognize this Contingency discourse relation.

We further examined the 36 wrong reversions of decisions. Around one third of these errors were due to either noise of event relation knowledge or incorrect coreference relations produced by the external CoreNLP coreference resolver we used. The remaining errors came from over-reliance of the full model on constraints in general. Considering the following example from the 36 reversions:

***DU1*:** Another analyst thought that India may have **pulled back** because of the concern over the stock market.

***DU2*:** India may have felt that if there was a severe **drop** in the stock market and it affected sugar, it could buy at lower prices.

***Gold Discourse Relation*:** Implicit Expansion

***Base Model's prediction*:** Implicit Expansion

***Full Model's prediction*:** Implicit Temporal

The full model could have relied on the event temporal relation between "pulled back" and "drop" and made the wrong discourse relation prediction.

## 6 Conclusion and Future Work

We have presented an effective regularization approach for incorporating external event knowledge and system predicted coreference relations into an existing paragraph-level neural network model for discourse parsing. Our approach tightly integrates knowledge and linguistic constraints with contexts for deriving knowledge-aware word vectors and meanwhile balances attentions over context and constraints through regularization, which robustly improves both implicit and explicit discourse relation classification performance on the benchmark PDTB corpus. In the future, we will identify new types of commonsense knowledge for further improving the performance of discourse parsing. For example, antonyms (e.g., *warm vs. cold*) can directly indicate a contrast relation between two situations, and this type of knowledge has potential to further improve the performance on *Comparison* discourse relations.

## Acknowledgments

We gratefully acknowledge support from National Science Foundation via the award IIS-1755943 and support from Institute of Education Sciences via the award R305A180319.

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Hongxiao Bai and Hai Zhao. 2018. Deep enhanced representation for implicit discourse relation recognition. In *COLING*, pages 571–583.

Or Biran and Kathleen McKeown. 2013. Aggregated word pair features for implicit discourse relation disambiguation. In *ACL*, volume 2, pages 69–73.

Or Biran and Kathleen McKeown. 2015. Pdtb discourse parsing as a tagging task: The two taggers approach. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 96–104.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NeurIPS*, pages 2787–2795.

Antoine Bosselut, Asli Celikyilmaz, Xiaodong He, Jianfeng Gao, Po-Sen Huang, and Yejin Choi. 2018. Discourse-aware neural rewards for coherent text generation. In *NAACL-HLT*, pages 173–184.

Chloé Braud and Pascal Denis. 2015. Comparing word representations for implicit discourse relation classification. In *EMNLP*, pages 2201–2211.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2003. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Current and new directions in discourse and dialogue*, pages 85–112. Springer.

Jifan Chen, Qi Zhang, Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Implicit discourse relation detection via a deep architecture with gated relevance network. In *ACL*.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Diana Inkpen, and Si Wei. 2018. Neural natural language inference models enhanced with external knowledge. In *ACL*, pages 2406–2417.

Timothy Chklovski and Patrick Pantel. 2004. Verbocean: Mining the web for fine-grained semantic verb relations. In *EMNLP*.

Kevin Clark and Christopher D Manning. 2016. Improving coreference resolution by learning entity-level distributed representations. In *ACL*, volume 1, pages 643–653.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*, pages 681–691.

Zeyu Dai and Ruihong Huang. 2018. Improving implicit discourse relation classification by modeling inter-dependencies of discourse units in a paragraph. In *NAACL*, volume 1, pages 141–151.

Bhuwan Dhingra, Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. 2017. Linguistic knowledge as memory for recurrent neural networks. *arXiv preprint arXiv:1703.02620*.

Shima Gerani, Yashar Mehdad, Giuseppe Carenini, Raymond T Ng, and Bita Nejat. 2014. Abstractive summarization of product reviews using discourse structure. In *EMNLP*, pages 1602–1613.

Fengyu Guo, Ruifang He, Di Jin, Jianwu Dang, Longbiao Wang, and Xiangang Li. 2018. Implicit discourse relation recognition using neural tensor network with interactive attention and sparse learning. In *COLING*.

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *ACL*, volume 1, pages 687–696.

Yangfeng Ji and Jacob Eisenstein. 2015. One vector is not enough: Entity-augmented distributed semantics for discourse relations. *TACL*, pages 329–344.

Yangfeng Ji, Gholamreza Haffari, and Jacob Eisenstein. 2016. A latent variable recurrent neural network for discourse relation language models. In *NAACL-HLT*, pages 332–342.

Yudai Kishimoto, Yugo Murawaki, and Sadao Kurohashi. 2018. A knowledge-augmented neural network model for implicit discourse relation classification. In *COLING*, pages 584–595.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, 2001, pages 282–289.

Man Lan, Jianxiang Wang, Yuanbin Wu, Zheng-Yu Niu, and Haifeng Wang. 2017. Multi-task attention-based neural networks for implicit discourse relationship representation and identification. In *EMNLP*, pages 1310–1319.

Wenqiang Lei, Xuancong Wang, Meichun Liu, Ilija Ilievski, Xiangnan He, and Min-Yen Kan. 2017. Swim: A simple word interaction model for implicit discourse relation recognition. In *IJCAI*, pages 4026–4032.

Wenqiang Lei, Yuanxin Xiang, Yuwei Wang, Qian Zhong, Meichun Liu, and Min-Yen Kan. 2018. Linguistic properties matter for implicit discourse relation recognition: Combining semantic interaction, topic continuity and attribution. In *AAAI*.

Beth Levin. 1993. *English verb classes and alternations: A preliminary investigation*. University of Chicago press.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*.

Ziheng Lin, Min-Yen Kan, and Hwee Tou Ng. 2009. Recognizing implicit discourse relations in the penn discourse treebank. In *EMNLP*, pages 343–351.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2014. A pdtb-styled end-to-end discourse parser. *Natural Language Engineering*, 20(2):151–184.

Yang Liu and Sujian Li. 2016. Recognizing implicit discourse relations via repeated reading: Neural networks with multi-level attention. In *EMNLP*, pages 1224–1233.

Yang Liu, Sujian Li, Xiaodong Zhang, and Zhifang Sui. 2016. Implicit discourse relation classification via multi-task neural networks. In *AAAI*, pages 2750–2756.

Yukun Ma, Haiyun Peng, and Erik Cambria. 2018. Targeted aspect-based sentiment analysis via embedding commonsense knowledge into an attentive lstm. In *AAAI*.

William C Mann. 1984. Discourse structures for text generation. In *10th International Conference on Computational Linguistics and 22nd Annual Meeting of the Association for Computational Linguistics*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *ACL System Demonstrations*, pages 55–60.

Joonsuk Park and Claire Cardie. 2012. Improving implicit discourse relation recognition through feature set optimization. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 108–112.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*, volume 1, pages 2227–2237.

Emily Pitler, Annie Louis, and Ani Nenkova. 2009. Automatic sense prediction for implicit discourse relations in text. In *ACL*, pages 683–691.

Emily Pitler and Ani Nenkova. 2009. Using syntax to disambiguate explicit discourse connectives in text. In *ACL-IJCNLP*, pages 13–16.

R. Prasad, N. Dinesh, Lee A., E. Miltsakaki, L. Robaldo, Joshi A., and B. Webber. 2008. The Penn Discourse Treebank 2.0. In *lrec2008*.

Lianhui Qin, Zhisong Zhang, and Hai Zhao. 2016a. Implicit discourse relation recognition with context-aware character-enhanced embeddings. In *COLING*, pages 1914–1924.

Lianhui Qin, Zhisong Zhang, and Hai Zhao. 2016b. A stacking gated neural architecture for implicit discourse relation classification. In *EMNLP*, pages 2263–2270.

Lianhui Qin, Zhisong Zhang, Hai Zhao, Zhiting Hu, and Eric P. Xing. 2017. Adversarial connective-exploiting networks for implicit discourse relation classification. In *ACL*, pages 1006–1017.

Attapol Rutherford, Vera Demberg, and Nianwen Xue. 2017. A systematic study of neural discourse models for implicit discourse relation. In *EACL*, pages 281–291.

Attapol Rutherford and Nianwen Xue. 2014. Discovering implicit discourse relations through brown cluster pair representation and coreference patterns. In *EACL*, pages 645–654.

Attapol Rutherford and Nianwen Xue. 2015. Improving the inference of implicit discourse relations via classifying explicit discourse connectives. In *NAACL-HLT*, pages 799–808.

Attapol T Rutherford and Nianwen Xue. 2016. Robust non-explicit neural discourse parser in english and chinese. *ACL*, page 55.

Cicero D Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *ICML*, pages 1818–1826.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *NeurIPS*, pages 926–934.

Robert Speer and Catherine Havasi. 2012. Representing general relational knowledge in conceptnet 5. In *LREC*, pages 3679–3686.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*, pages 5998–6008.

Bonnie Webber, Andrei Popescu-Belis, and Jörg Tiedemann. 2017. Proceedings of the third workshop on discourse in machine translation. In *Proceedings of the Third Workshop on Discourse in Machine Translation*.

Yang Xu, Yu Hong, Huibin Ruan, Jianmin Yao, Min Zhang, and Guodong Zhou. 2018. Using active learning to expand training data for implicit discourse relation recognition. In *EMNLP*, pages 725–731.

Zhen Xu, Bingquan Liu, Baoxun Wang, Chengjie Sun, and Xiaolong Wang. 2017. Incorporating loose-structured knowledge into conversation modeling via recall-gate lstm. In *IJCNN*, pages 3506–3513. IEEE.

Nianwen Xue, Hwee Tou Ng, Sameer Pradhan, Rashmi PrasadO Christopher Bryant, and Attapol T Rutherford. 2015. The conll-2015 shared task on shallow discourse parsing. *CoNLL*, page 1.

Bishan Yang and Tom Mitchell. 2017. Leveraging knowledge bases in lstms for improving machine reading. In *ACL*, volume 1, pages 1436–1446.

Wenlin Yao and Ruihong Huang. 2018. Temporal event knowledge acquisition via identifying narratives. In *ACL*, volume 1, pages 537–547.

Hao Zhou, Tom Young, Minlie Huang, Haizhou Zhao, Jingfang Xu, and Xiaoyan Zhu. 2018. Commonsense knowledge aware conversation generation with graph attention. In *IJCAI*, pages 4623–4629.

# Weakly Supervised Multilingual Causality Extraction from Wikipedia

**Chikara Hashimoto**
Yahoo Japan Corporation
1-3 Kioicho, Chiyoda-ku, Tokyo 102-8282, Japan
chashimo@yahoo-corp.jp

## Abstract

We present a method for extracting causality knowledge from Wikipedia, such as *Protectionism* → *Trade war*, where the cause and effect entities correspond to Wikipedia articles. Such causality knowledge is easy to verify by reading corresponding Wikipedia articles, to translate to multiple languages through Wikidata, and to connect to knowledge bases derived from Wikipedia. Our method exploits Wikipedia article sections that describe causality and the redundancy stemming from the multilinguality of Wikipedia. Experiments showed that our method achieved precision and recall above 98% and 64%, respectively. In particular, it could extract causalities whose cause and effect were written distantly in a Wikipedia article. We have released the code and data for further research.

## 1 Introduction

Much of the world consists of entities that causally depend on each other. Therefore, causality knowledge, e.g., *Protectionism* → *Trade war*,[1] is useful for many tasks such as why-QA (Oh et al., 2017), reading comprehension (Berant et al., 2014), and event prediction (Radinsky et al., 2012).

Although many methods have been proposed for causality extraction from text (Ning et al., 2018; Gao et al., 2018; Kruengkrai et al., 2017; Rehbein and Ruppenhofer, 2017; Dunietz et al., 2017; Hidey and McKeown, 2016; Zhao et al., 2016; Hashimoto et al., 2014, 2012; Do et al., 2011; Riaz and Girju, 2010; Abe et al., 2008), they have rarely addressed three issues that are important for constructing a causality knowledge base (CKB). First, we should be able to *verify* extracted causalities, so that the CKB can sustain the credibility of its information.[2] Second, it would be desirable to easily *translate* the CKB to multiple languages, to avoid duplicating the construction effort for different languages. Third, it would also be desirable to automatically *connect* the CKB to other knowledge bases (KBs), to bring together KB construction efforts in various communities and thus maximize their synergistic effect.

Therefore, we propose a method for extracting causalities from Wikipedia by using cause and effect entities that correspond to Wikipedia articles; for example, for *Tobacco* → *Lung cancer*, English Wikipedia has articles titled *Tobacco* and *Lung cancer*. Such causalities satisfy the above three desiderata. First, we can easily *verify* such causalities, because Wikipedia articles tend to credibly attest them; for example, the *Tobacco* article states that inhaling its smoke can cause *Lung cancer*.[3] In contrast, knowledge from other sources such as the web text tends to be difficult to verify, owing to a deluge of false information. Second, causalities extracted from Wikipedia can be *translated* trivially to multiple languages, because Wikidata (Vrandečić and Krötzsch, 2014), a free, multilingual KB, provides links among Wikipedia articles on the same subject in different languages. Third, it is easy to *connect* causalities whose cause and effect are the topics of Wikipedia articles to other KBs derived from Wikipedia, e.g., Freebase (Bollacker et al., 2008), DBpedia (Lehmann et al., 2009), Knowledge Graph (Singhal, 2012), YAGO2 (Hoffart et al., 2013), and Wikidata. In addition, Wikidata also provides links to external KBs, which connect the causalities to those KBs.

Because our task is to identify causal relations between entities that are the topics of Wikipedia articles, a simple solution would be to use relation extraction (RE) methods (Vashishth et al., 2018; Zhang et al., 2018, 2017; Zhou et al., 2016;

[1] In this paper, $A \rightarrow B$ denotes that $A$ causes $B$.

[2] The verifiability in this paper means that one can verify extracted causalities with credible sources of information.

[3] Most of the Wikipedia contents mentioned in this paper were downloaded on January 7th, 2019.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2988–2999, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

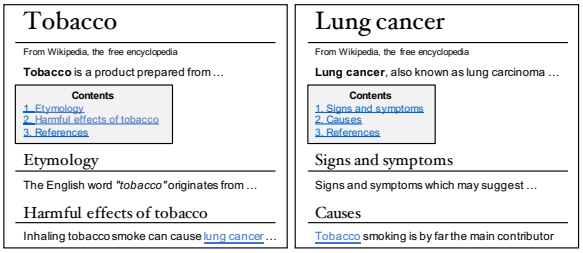


Figure 1: Sections that describe causality in Wikipedia: *Harmful effects of tobacco* and *Causes*. We can extract *Tobacco* → *Lung cancer* from these sections.

Lin et al., 2016; Wang et al., 2016; Cai et al., 2016; Shen and Huang, 2016). Most RE methods, however, identify relations between entities co-occurring in a sentence (with exceptions such as Quirk and Poon, 2017; Peng et al., 2017) and cannot deal with a large portion of our test data (67.3%, as we detail in §3.2), because the cause and effect entities in the data tend to appear distantly in a Wikipedia article. This is reasonable given that the subject of an encyclopedia article would not typically be repeated throughout the article, as the author may expect readers to know that the whole article is about the subject.

We aim to extract causalities from Wikipedia regardless of whether their cause and effect co-occur in a sentence by training a causality classifier from causality instances whose cause and effect may appear distantly. Here, we face two problems:

**Lack of labeled data:** For the learning of the causality classifier, there is no data marking causes and effects in Wikipedia articles, and manually creating such data would be laborious. A possible solution would be distant supervision (Mintz et al., 2009) using external KBs to mark relevant entities in articles. Because Wikipedia evolves rapidly, however, relying on such KBs would make the resulting CKB obsolete. Therefore, how can we instantly acquire such data directly from Wikipedia?

**Lack of redundancy:** An encyclopedia tends to avoid redundancy in text, which many knowledge extraction methods rely on to reliably estimate the probability of extracted knowledge (Christensen et al., 2011; Downey et al., 2005). Therefore, how can we reliably estimate the probability of a causality extracted from Wikipedia?

For the lack of labeled data, we can accurately extract hundreds of causalities by harnessing the property of Wikipedia that some articles have sections describing relevant causalities. For example, as shown in Fig. 1, the *Lung cancer* article has a section named *Causes* that mentions *Tobacco*, and the *Tobacco* article has a section named *Harmful effects of tobacco* that mentions *Lung cancer*. Accordingly, we can extract *Tobacco* → *Lung cancer* and then use it for the learning of the classifier. All the supervision we need is a handful of keywords for such sections, e.g., *causes* and *effects*.

For the lack of redundancy, we exploit another kind of redundancy stemming from the *multilinguality* of Wikipedia: the same subject may be described by articles in different languages; for example, *Tobacco* is described in 112 languages. We thus use a data source consisting of Wikipedia articles in nine languages: English (en), German (de), French (fr), Spanish (es), Italian (it), Portuguese (pt), Swedish (sv), Dutch (nl), and Polish (pl). This requires our method to be language independent. For these nine languages, however, the only required linguistic analysis is detection of word boundaries, i.e., white space. This simplicity has an advantage of allowing our method to parse Wikipedia quickly to keep the CKB up to date.

We evaluated our method by using the relation triples in Wikidata, which represents a causality by either a has cause or a has effect relation (§3). Our method achieved precision and recall above 98% and 64%, respectively, rivaling an *oracle* relation extractor that perfectly detected the causality between entities co-occurring in a sentence. We also confirmed that the multilingual redundancy of Wikipedia was effective: using more languages led to significantly better performances.

**Our contributions** are five-fold. (1) We proposed the three desiderata for CKBs: *verifiability*, *translatability*, and *connectivity*. (2) We presented the ideas of exploiting the *causality-describing sections* and *multilingual redundancy* of Wikipedia. (3) Based on these ideas, we proposed a weakly-supervised, multilingual causality extraction method. (4) We evaluated our method in relatively large-scale settings. (5) We have released the code and data from this study (§6).[4]

In this paper we focus on causality extraction. We will present how to construct the CKB from causalities extracted by our method in future.

In this study, we define a causality $A \rightarrow B$ according to Wikipedia and Wikidata. Specifically,

[4]The code and data from this study will be available at https://research-lab.yahoo.co.jp/people/chikara_hashimoto.html.

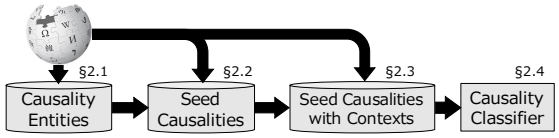


Figure 2: Overview of our method. Numbers indicate sections that describe corresponding components.

$A \rightarrow B$ if Wikipedia describes $A$ as causing $B$ either explicitly or implicitly, or if $A$ ($B$) has the has effect (has cause) relation to $B$ ($A$) in Wikidata.

## 2 Proposed Method

Our method learns a causality classifier that, given an entity pair $(e_1, e_2)$, determines if $e_1 \rightarrow e_2$ holds. The entities are Wikidata identifiers (IDs) having at least one corresponding Wikipedia article. For example, Q1566 is the Wikidata ID for *Tobacco*, which is described in Wikipedia in 112 languages.

Figure 2 shows the learning process; the method identified entities that tend to participate in causality (§2.1), extracted seed causalities between such entities from causality-describing sections as illustrated in Fig. 1 (§2.2), extracted the contexts of the seed causalities from articles in multiple languages (§2.3), and finally learned the classifier from the multilingual contexts of the seeds (§2.4).

The resulting classifier thus takes $(e_1, e_2)$ as input, examines the multilingual contexts of $e_1$ and $e_2$, and determines if $e_1 \rightarrow e_2$ holds.

The Python code that accompanies this paper is an implementation of our method (§6).

### 2.1 Causality Entity Extraction

Some entities, e.g., *Tobacco*, are more likely to participate in causality than others, e.g., *A (the alphabet letter)*. We call them *causality entities*. To accurately extract seed causalities in the next step, we first extracted causality entities by identifying articles that had causality-describing sections: *Tobacco* and *Lung cancer* in Fig. 1 were thus regarded as causality entities.

To identify causality-describing sections, we manually prepared a handful of keywords that tended to appear in the titles of such sections for the nine languages: en, de, fr, es, it, pt, sv, nl, and pl. This was the only supervision of our method. Specifically, we chose *Cause*, *Causes*, *Effect*, and *Effects* as such keywords for en and translated them to the other languages by reference to Wiktionary.[5] We have released these keywords (§6).

We then extracted causality entities from the Wikipedia dump of each language and kept only those entities that appeared in more than one language. The identify of an entity across languages could be confirmed by Wikidata IDs; e.g., *Tobacco* (en) and *Tabaco* (es) both correspond to Q1566.

### 2.2 Seed Causality Extraction

A seed causality is an entity pair $(e_1, e_2)$ such that $e_1$ appears in a causality-describing section, whose title contains *Cause* or *Causes* (in the case of en), in the article corresponding to $e_2$; and such that $e_2$ appears in a causality-describing section, whose title contains *Effect* or *Effects*, in the article corresponding to $e_1$. For instance, *Tobacco* $\rightarrow$ *Lung cancer* in Fig. 1 is a seed causality, as it satisfies the above condition.

We extracted seed causalities in this way, kept only those that appeared in more than one language, and consequently acquired 879 seed causalities from the nine languages. We have also released the seed causalities (§6).

### 2.3 Seed Causality Context Extraction

As illustrated in Fig. 3, for each entity in a seed causality, we extracted its contexts from articles in multiple languages as features for the classifier.

The context window was up to 100 words to the left and right of each target entity. When a section title appeared in a context window, we shrank the window to the section title position so that the window would not cross the section boundary.

We restricted the articles from which contexts were extracted as follows. For $e_1 \rightarrow e_2$, the context of $e_1$ ($e_2$) was extracted from the article that corresponded to $e_2$ ($e_1$). For example, for *Tobacco* $\rightarrow$ *Lung cancer*, the context of *Tobacco* was extracted only from the *Lung cancer* article, and that of *Lung cancer* was extracted only from the *Tobacco* article. This reduced the processing time and helped extract only highly relevant contexts for a target causality. We extracted the contexts directly from the Wikipedia source texts with all markups kept intact, because those markups may have helped the classifier, and parsing the source texts would have slowed the process. In addition, we replaced $e_1$ and $e_2$ in the extracted contexts with the special symbols `__CAUSE__` and `__EFFECT__`.

[5] https://en.wiktionary.org/

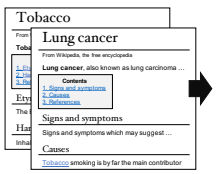


English Wikipedia
there is sufficient evidence that involuntary smoking (exposure to secondhand or 'environmental' __CAUSE__ smoke) causes __EFFECT__ in humans.

French Wikipedia
la cause la plus fréquente de __EFFECT__ est l'exposition chronique à la fumée de [[#tabagisme|__CAUSE__ ]]

Spanish Wikipedia
algunos tipos de cáncer se encuentran directamente relacionados con el __CAUSE__ , como es el caso del [[__EFFECT__]] y otros cánceres del tracto respiratorio

Figure 3: Seed causality context extraction from Wikipedia articles of multiple languages.

We also removed line-break characters to extract contexts from across multiple lines.

For each seed causality, we concatenated all the contexts from multiple languages. For context extraction, we used only four languages —en, de, fr, and es— because they had the broadest coverage.

We also removed infoboxes from articles before context extraction, because we used Wikidata's triples for evaluation, and some of those have been transcribed into infoboxes.[6] This was for fairness rather than methodological considerations.

### 2.4 Learning Classifier

For learning of the causality classifier, we used the seed causalities as positive instances. In contrast, negative (non-causality) instances were entity pairs such that the article corresponding to one entity had a link to the article corresponding to the other entity. For example, the article on *Barack Obama* has a link to the article on *Hillary Clinton*, and hence, *Barack Obama* $\rightarrow$ *Hillary Clinton* could be used as a negative instance. In this way, we could obtain sensible negative instances, meaning that they were not totally random pairs but were semantically related to each other in some way. We obtained the same number (879) of negative instances as positive ones and similarly extracted their multilingual contexts; thus, we trained the classifier with 1,758 instances.

We used fastText (Joulin et al., 2016), a linear text classifier that averages word embeddings and makes a prediction based on the averaged embeddings, as the causality classifier because of its speed and accuracy. Among its hyperparameters, we tuned only the number of epochs and the learning rate, as detailed in §3.2. We did not use pretrained word embeddings for the classifier to minimize its dependence on external resources.

Although there would be more sophisticated approaches to modeling the multilingual contexts extracted from Wikipedia, we intentionally adopted the very simple modeling in this paper in order to show that even such a simple method could deliver good performances by using the multilingual contexts of Wikipedia. We will develop more sophisticated models in future.

[6] https://www.wikidata.org/wiki/Wikidata:WikiProject_Infoboxes

## 3 Experiments

Using relation triples in Wikidata (§3.1), we evaluated our method by comparing it with various baselines (§3.2). We also measured the effectiveness of using multiple languages (§3.3).

We can summarize the results as follows. Our method (1) extracted causality with precision and recall above 98% and 64%, respectively; it (2) rivaled the performance of an oracle relation extractor that worked sentence-wise; and it (3) effectively used multiple languages.

### 3.1 Test Data

Our experiments were based on labeled data derived from Wikidata, which has various relation triples $(e_1, rel, e_2)$, where $e_1$ and $e_2$ are entities ("item identifiers" in Wikidata terms, such as Q1566) between which the relation $rel$ (a "property" in Wikidata terms, such as has cause) holds.

In short, we used triples whose relations were either has cause or has effect as positive (causality) instances and those with other relations as negative (non-causality) instances. Because we aimed to extract causalities that were easy to verify by reading individual Wikipedia articles, we restricted the triples to those whose component entities co-occurred in an article. Specifically, we used only triples such that the article corresponding to one entity had a link to the article corresponding to the other entity in at least one language. Consequently, we obtained 1,524 positive instances and the same number of negative instances, giving 3,048 instances in total.

The data was in no way easy to classify, because all the negative instances were not random pairs but had semantic relations that are as natural and common as causality. For example, in Wikidata, *World War I* (Q361) and the *Paris Peace Conference* (Q199820) show causality, as the former has

the has effect relation with the latter. In contrast, *World War I* and the *German invasion of Belgium* (Q5551414) do not show causality, because they have a significant event relation between them.

The data accompany the paper (§6).

### 3.2 Performance of Proposed Method

#### Experimental Settings for Proposed Method

In this section, we denote our proposed method as PROP. In the experiments, it extracted multilingual contexts for each relation triple in the data, as described in §2.3. The data were used only for testing. Training was conducted using the automatically acquired seed causalities (§2).

**Hyperparameter tuning** for PROP was based on five-fold cross validation using the seed causalities so that we could maximize the F1 score on them. We only tuned the number of epochs and the learning rate of fastText; the former was chosen from 30, 50, 70, and 100, while the latter was chosen from 0.3, 0.5, 0.7, and 1.0. As a result, they were set to 100 and 1.0, respectively. The other fastText parameters were set to default values.

We used the default fastText threshold for classification. We conducted 10 runs and averaged the performance scores (accuracy, precision, recall, and F1) for evaluation.

#### Compared Methods

We evaluated the performance of PROP by comparing it with the following baseline methods described below: SECTION, INFOBOX, RELATED, and ORACLE RE.

**SECTION** This was simply the method of seed causality extraction described in §2.2, which extracted 879 causalities from Wikipedia. This was intended to evaluate two questions. The first was how well our idea of exploiting causality-describing sections in Wikipedia worked on its own. The second question was how much PROP improved the results, as it used the classifier trained on the output of the SECTION method.

**INFOBOX** This method extracted causality from the infoboxes in Wikipedia articles in the nine languages. Some infoboxes contain information on causality; for example, the article on *Candidiasis* has an infobox with the field *Causes*, whose value is *Candida*, which indicates *Candida* → *Candidiasis*. For the INFOBOX method to identify such causality-describing fields in infoboxes, we used the same set of keywords that PROP used to identify causality-describing sections. As discussed in §2.3, some Wikidata triples have been transcribed into infoboxes, which should give the INFOBOX method an advantage over PROP, as PROP was forbidden to use infoboxes.

**RELATED** This method regarded an entity pair whose entities were semantically related to each other as a causality, because such relatedness has been shown to imply causality (Do et al., 2011; Riaz and Girju, 2010). Specifically, this method used a semantic relatedness defined as $1-sr(a,b)$, where $sr(a,b)$ is a Wikipedia-link-based distance measure that was proposed by Milne and Witten (2008) and has been widely used (Lee et al., 2015). The measure is defined

$$sr(a,b) = \frac{log(max(|A|,|B|)) - log(|A \cap B|)}{log(|W|) - log(min(|A|,|B|))},$$

where $a$ and $b$ are the two articles (entities) of interest, $A$ and $B$ are the sets of all articles that link to $a$ and $b$, respectively, and $W$ is the entire set of Wikipedia articles. To classify entity pairs, we set a threshold for the relatedness score so that we could maximize F1 on the training data for PROP (§2.4). The threshold value ranged from 0.00 to 1.00 at intervals of 0.01. We used the link structure of en Wikipedia because it is the largest one.

**ORACLE RE** This method used an *oracle* relation extractor that, given two entities that co-occur in a sentence, never fails to determine whether they have a causal relation. For other pairs, whose component entities did not co-occur in a sentence, this method uniformly guessed that they had no causal relation. We used en Wikipedia for ORACLE RE because it has the broadest coverage. We segmented articles into sentences with spaCy,[7] which accurately recognizes sentence boundaries through dependency parsing. This method would show the upper-bound performance for our task of typical RE methods that work sentence-wise. This was an ambitious baseline given the performances of state-of-the-art RE methods: Vashishth et al. (2018) achieved a top-100 precision of 84% on the New York Time corpus (Riedel et al., 2010); Wang et al. (2016) achieved an F1 score of 88% on SemEval-2010 Task 8 (Hendrickx et al., 2010).

| | Acc | Prec | Rec | F1 |
|---|---|---|---|---|
| SECTION | 50.56 | 100.00 | 1.12 | 2.21 |
| INFOBOX | 53.71 | 100.00 | 7.41 | 13.81 |
| RELATED | 68.86 | 66.23 | 76.97 | 71.20 |
| ORACLE RE | 75.89 | 100.00 | 51.77 | 68.22 |
| PROP | 81.45 | 98.28 | 64.02 | **77.53** |

Table 1: Performance results of the compared methods.

**Results**

Table 1 lists the accuracy (Acc), precision (Prec), recall (Rec), and F1 of the compared methods.

The SECTION method achieved 100% precision, which indicates that our technique of exploiting causality-describing sections in Wikipedia could accurately extract causalities. As the method's recall indicates, however, it covered only a small portion of our target causalities.

The INFOBOX method also achieved 100% precision, though its coverage was also quite limited.

The RELATED method exhibited the highest recall, but the precision was unacceptably low for the subsequent manual labor that would be required to construct the CKB.

The ORACLE RE method achieved 100% precision by design. Its recall was rather low because 67.3% of the entity pairs in the data (§3.1) consisted of entities that did NOT co-occur in a sentence. This means that most RE methods that work sentence-wise will miss a large portion of causalities, regardless of their accuracy.

Finally, PROP achieved the best F1 score, though its recall still had room for improvement. The fact that PROP outperformed the baselines, especially ORACLE RE, clearly shows the effectiveness of our method.

### 3.3 Language Ablation Test

We also examined the effect of using more languages for extracting contexts by comparing the performance of the following variations of PROP: PROP, PROP$_{en.de.fr}$, PROP$_{en.de.es}$, PROP$_{en.fr.es}$, PROP$_{en}$, PROP$_{en.de}$, PROP$_{en.fr}$, and PROP$_{en.es}$. Unlike PROP, which used en, de, fr, and es for extracting multilingual contexts, the others used only the languages indicated by the subscript language symbols. For example, PROP$_{en.de}$ used only en and de Wikipedia articles.

For each variation of our method, we conducted 10 runs and averaged the resulting performance scores for evaluation.

[7] https://spacy.io/

| | Acc | Prec | Rec | F1 |
|---|---|---|---|---|
| PROP$_{en.de.fr.es}$ | 81.45 | 98.28 | 64.02 | **77.53** |
| PROP$_{en.de.fr}$ | 80.54 | 98.16 | 62.24 | 76.18$^{*}$ |
| PROP$_{en.de.es}$ | 80.82 | 98.65 | 62.49 | 76.51$^{*}$ |
| PROP$_{en.fr.es}$ | 79.20 | 98.14 | 59.52 | 74.10$^{*}$ |
| PROP$_{en.de}$ | 79.92 | 99.21 | 60.32 | 75.26$^{\diamond}$ |
| PROP$_{en.fr}$ | 78.39 | 98.32 | 57.76 | 72.77$^{\diamond}$ |
| PROP$_{en.es}$ | 77.91 | 98.91 | 56.45 | 71.88$^{\diamond}$ |
| PROP$_{en}$ | 76.43 | 98.91 | 53.45 | 69.40 |

Table 2: Results of language ablation tests. PROP is denoted as PROP$_{en.de.fr.es}$ for clarity. $*$ and $\diamond$ indicate statistically significant differences from the performances of PROP$_{en.de.fr.es}$ and PROP$_{en}$, respectively (paired t-test: $p < 0.01$).

Table 2 summarizes the results of the ablation tests. The upper half of the table indicates that removing one language from PROP tended to degrade its performance, while the lower half indicates that adding one language to PROP$_{en}$ tended to improve its performance. From these results, we conclude that using multilingual contexts is effective for causality extraction from Wikipedia.

### 3.4 Analysis

We first examine cases in which PROP succeeded and then analyze its error cases.

Causalities whose cause and effect entities co-occur in a sentence tend to have phrases indicating the causal relation between them, which helps identify the relation. For example, for *Adipsia* $\rightarrow$ *Hypernatremia*, there is the following sentence:

(1) *Adipsia* may be seen in conditions such as diabetes insipidus and may **result in** *hypernatremia*.

For causalities whose cause and effect do not co-occur in a sentence, it is likely that their causal relation are only indicated by multi-line texts or the structure of Wikipedia article. For example, for *Hormone therapy* $\rightarrow$ *Cancer pain*, there is the following list item in the *Cause* section of the article of *Cancer pain*:

(2) *hormone therapy*, which sometimes **causes** pain flares;

Although *Cancer pain* is not written explicitly in the item, we can guess that the *pain flare* refers to *Cancer pain*, because this list item is part of the contents of the *Cause* section of the article of *Cancer pain*. PROP can identify such a causality,

because pieces of Wikipedia source text that represent this kind of structure can be included in the window of the multilingual contexts and because all contexts are extracted from the articles that correspond to either the cause or the effect.

Regarding PROP's error analysis, we focus here on its false-negative errors, as it achieved high precision while its recall had room for improvement. Such errors were mostly due to the lack of evidence of causality; in some cases, even though the cause and effect entities both appear in an article, it does not indicate their causal relation. For example, for *Psoriasis* → *Woronoff's ring*, the cause is only mentioned as a list item in the *See also* section of the effect entity's article in en Wikipedia.

Other false-negatives included entity pairs for which not only causality but other relations hold. For example, for *International University Sports Federation* → *Universiade*, both the has effect and the organizer relations hold. These tend to be cases in which the Wikipedia article describes the cause as *organizing* (instead of *causing*) the effect.

## 4 Discussion

### 4.1 Three Desiderata for CKB

In §1 we discussed the desiderata for the CKB: verifiability, translatability, and connectivity. In this section we consider how well the causalities extracted by PROP satisfied the desiderata.

#### Verifiability

We examined 100 samples from the causalities in the data (§3.1) that PROP correctly classified as causalities (i.e., true positives) to measure their verifiability. In other words, we examined how many of them consisted of cause and effect entities between which we could easily identify causality by reading their individual Wikipedia articles. Causalities that were not described in en Wikipedia were regarded as unverifiable, because we assumed that our target users could understand only English. Of the 100 samples, 19 were not written in en Wikipedia.

As a result, 62.0% of the samples were verifiable. If we ignore those 19 samples that were not written in en Wikipedia, 76.5% of the samples were verifiable. For example, for *Onchocerca volvulus* → *Onchocerciasis*, the article on *Onchocerca volvulus* has the following sentence:

(3) *Onchocerca volvulus* is a nematode that **causes** *onchocerciasis*.

We thus conclude that causalities extracted by PROP tend to be verifiable.

#### Translatability

We next examined the number of languages to which each true-positive causality was translated, by using Wikidata's links among Wikipedia articles on the same subject in different languages. We targeted each of the nine languages. We regarded each causality as translated to a language if its cause and effect were both translated.

As a result, 74.9% of the causalities were translated to more than one language, which indicates that the causalities extracted by PROP tend to exhibit a high degree of translatability. Furthermore, 16.5% were translated to all nine languages, including, e.g., *Chemotherapy* → *Vomiting* (*Q974135* → *Q127076*) and *Treaty of Versailles* → *World War II* (*Q8736* → *Q362*).

#### Connectivity

We also examined how many of the true positives were connected to external KBs. We first listed all external IDs,[8] which Wikidata uses to identify external KBs such as Freebase, IMDb, and ISBN-13, with a lookup function in Wikidata,[9] resulting in 3,695 external IDs. We then made a table to map each Wikidata ID to the external IDs to which it is connected. With the table we counted the number of true-positive causalities whose cause and effect entities shared at least one external ID.

Consequently, 72.3% of the true-positives were connected to external KBs. For example, *Diabetes mellitus* → *Cataract* was connected to 19 KBs such as MeSH,[10] Freebase, and BabelNet (Navigli and Ponzetto, 2012). We thus conclude that causalities extracted by PROP indicate a high degree of connectivity.

### 4.2 Independence of Languages

We designed PROP to be language independent so that we could exploit multilingual redundancy. Unfortunately, unintended language dependence might arise as we use more languages. We are currently aware that we will need tokenizers if we use languages for which word boundaries are not as explicit as in the nine languages here, e.g., Chinese and Japanese. It will thus be a future challenge to

[8] https://www.wikidata.org/wiki/Wikidata:Identifiers

[9] https://www.wikidata.org/wiki/Special:ListProperties

[10] https://www.ncbi.nlm.nih.gov/mesh/

use many more languages while keeping PROP as language independent as possible.

### 4.3 Independence of External KBs

We also designed PROP not to rely on external KBs so that we could easily keep the CKB up to date with Wikipedia. If we relaxed this design policy, we would be able to use the triples in Wikidata as additional training data for our classifier. We thus plan to explore this direction of research.

Another external resource that is useful for PROP would be pre-trained word vectors (Bojanowski et al., 2017). We evaluated PROP$_{en+}$, which is the same as PROP$_{en}$ except that it used pre-trained word vectors.[11] The pre-trained word vectors slightly improved the performance; PROP$_{en+}$ achieved a F1 of 69.51, while PROP$_{en}$'s F1 was 69.40.

## 5 Related Work

### 5.1 Causality Extraction

Causality extraction methods can be classified with regard to what constitutes cause and effect: noun phrases, verb phrases, or clauses. The noun-phrase type, e.g., *global warming* → *malaria epidemic*, has mostly been addressed by RE methods, as we discuss in §5.2. The verb-phrase type, e.g., *get fired* → *live on unemployment insurance*, has been extracted by various methods (Ning et al., 2018; Gao et al., 2018; Rehbein and Ruppenhofer, 2017; Kruengkrai et al., 2017; Hashimoto et al., 2015, 2014, 2012; Do et al., 2011; Riaz and Girju, 2010; Abe et al., 2008). The clause type, e.g., *I hid the car key* → *She's mad*, has also been studied (Dunietz et al., 2017). Other types include causal embeddings (Sharp et al., 2016), which can be used for causal question answering (Oh et al., 2017). We focused here on the noun-phrase type because noun phrases can be components of verb phrases and clauses, and hence, our work may also contribute to the extraction of other types.

Another standpoint of classifying causality extraction is the information source, e.g., newspapers (Khoo et al., 1998), the web (Kruengkrai et al., 2017), parallel corpora (Hidey and McKeown, 2016),[12] images (Gao et al., 2018), and videos (Fire and Zhu, 2016). We used Wikipedia articles in multiple languages because they tend to be more credible than other sources, and because they allowed us to exploit multilingual redundancy.

Wikipedia articles in multiple languages also provide causalities that tend to satisfy the three desiderata discussed in §1, which is the novel perspective that we proposed and that many previous studies lacked. In addition, we proposed a novel method that is better suited for extracting such causalities than previous methods were.

### 5.2 Relation Extraction

In SemEval-2007 Task4 (Girju et al., 2007) and SemEval-2010 Task 8 (Hendrickx et al., 2010), the target relations included "Cause-Effect"; our study is also relevant to methods for those tasks (Lee et al., 2019; Wang et al., 2016; Shen and Huang, 2016; Cai et al., 2016; Xu et al., 2016). Other methods based on RE also addressed causality (Kim and Myaeng, 2016; De Saeger et al., 2011, 2009; Schoenmackers et al., 2010).

Although our method can be regarded as a relation extractor, it is different from the above methods because it is particularly tailored to causality extraction from Wikipedia. For this task, it is important to extract relation instances whose component entities do not co-occur in a sentence.

More recent studies have addressed *inter*-sentential RE for specialized domains (Noriega-Atala et al., 2018; Quirk and Poon, 2017; Peng et al., 2017), and Mandya et al. (2018) constructed a large-scale dataset for this task. Hence, we plan to incorporate these approaches into our method.

### 5.3 Knowledge Extraction from Wikipedia

Knowledge extraction from Wikipedia in general is also relevant to our study. Although there have been studies on extracting class concepts (Pasca, 2018), trivia (Tsurel et al., 2017), taxonomies (Flati et al., 2014), infobox contents (Wang et al., 2013), and various semantic relations (Wu and Weld, 2010), among other things, causality extraction from Wikipedia has rarely been addressed as far as we are aware. Hidey and McKeown (2016) addressed the extraction of linguistic markers indicating causality from Wikipedia, but those markers were not causalities *per se*.

### 5.4 Temporal Relation Extraction

Researchers have noticed that causality extraction and temporal relation extraction (Mani et al.,

[11] We used two million word vectors trained with subword information on Common Crawl available at https://fasttext.cc/docs/en/english-vectors.html.

[12] Precisely, the task of Hidey and McKeown (2016) is identifying linguistic cues that indicate causality.

2006) share some properties and can complement each other (Ning et al., 2018; Mirza and Tonelli, 2016; Bethard and Martin, 2008). In the future, we will also explore the possibility of exploiting temporal relation extraction methods for our task.

## 6 Accompanying Code and Data

We hereby release the Python code to implement our method, including modules for generating relevant data, with step-by-step instructions. We also release the following data sets: (a) the keywords for identifying causality-describing sections (§2.1), (b) the seed causalities (§2.2), and (c) the test data (§3.1). The code and data will be available at `https://research-lab.yahoo.co.jp/people/chikara_hashimoto.html`.

Note that 98.1% of the seed causalities are not included in Wikidata as causality, and they can increase Wikidata causalities by about 19.8%.

## 7 Conclusion

We have proposed a weakly supervised method for extracting causality from Wikipedia articles in multiple languages. The causalities extracted by our method tend to be easy to *verify*, *translate* to multiple languages, and *connect* to external KBs. Our key idea is to exploit the *causality-describing sections* and *multilingual redundancy* of Wikipedia. Our method achieved precision and recall above 98% and 64%, respectively, and it could even extract causalities whose cause and effect entities did not co-occur in a sentence.

## References


Shuya Abe, Kentaro Inui, and Yuji Matsumoto. 2008. Two-phased event relation acquisition: Coupling the relation-oriented and argument-oriented approaches. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING)*, pages 1–8.

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D. Manning. 2014. Modeling biological processes for reading comprehension. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1499–1510.

Steven Bethard and James H. Martin. 2008. Learning semantic links from a corpus of parallel temporal and causal relations. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies (ACL-HLT)*, pages 177–180.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD Conference*, pages 1247–1250.

Rui Cai, Xiaodong Zhang, and Houfeng Wang. 2016. Bidirectional recurrent convolutional neural network for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 756–765.

Janara Christensen, Mausam, Stephen Soderland, and Oren Etzioni. 2011. An analysis of open information extraction based on semantic role labeling. In *Proceedings of the Sixth International Conference on Knowledge Capture*, pages 113–120.

Stijn De Saeger, Kentaro Torisawa, Jun'ichi Kazama, Kow Kuroda, and Masaki Murata. 2009. Large scale relation acquisition using class dependent patterns. In *The Ninth IEEE International Conference on Data Mining (ICDM)*, pages 764–769.

Stijn De Saeger, Kentaro Torisawa, Masaaki Tsuchida, Jun'ichi Kazama, Chikara Hashimoto, Ichiro Yamada, Jong-Hoon Oh, István Varga, and Yulan Yan. 2011. Relation acquisition using word classes and partial patterns. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 825–835.

Quang Xuan Do, Yee Seng Chan, and Dan Roth. 2011. Minimally supervised event causality identification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 294–303.

Doug Downey, Oren Etzioni, and Stephen Soderland. 2005. A probabilistic model of redundancy in information extraction. In *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, pages 1034–1041.

Jesse Dunietz, Lori Levin, and Jaime Carbonell. 2017. Automatically tagging constructions of causation and their slot-fillers. *Transactions of the Association for Computational Linguistics (TACL)*, 5:117–133.

Amy Fire and Song-Chun Zhu. 2016. Learning perceptual causality from video. *ACM Transactions on Intelligent Systems and Technology (TIST) - Special Issue on Causal Discovery and Inference*, 7(2):23:1–23:22.

Tiziano Flati, Daniele Vannella, Tommaso Pasini, and Roberto Navigli. 2014. Two is bigger (and better) than one: the wikipedia bitaxonomy project. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 945–955.

Qiaozi Gao, Shaohua Yang, Joyce Chai, and Lucy Vanderwende. 2018. What action causes this? towards naive physical action-effect prediction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 934–945.

Roxana Girju, Preslav Nakov, Vivi Nastase, Stan Szpakowicz, Peter Turney, and Deniz Yuret. 2007. Semeval-2007 task 04: Classification of semantic relations between nominals. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 13–18.

Chikara Hashimoto, Kentaro Torisawa, Stijn De Saeger, Jong-Hoon Oh, and Jun'ichi Kazama. 2012. Excitatory or inhibitory: A new semantic orientation extracts contradiction and causality from the web. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 619–630.

Chikara Hashimoto, Kentaro Torisawa, Julien Kloetzer, and Jong-Hoon Oh. 2015. Generating event causality hypotheses through semantic relations. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI)*, pages 2396–2403.

Chikara Hashimoto, Kentaro Torisawa, Julien Kloetzer, Motoki Sano, István Varga, Jong-Hoon Oh, and Yutaka Kidawara. 2014. Toward future scenario generation: Extracting event causality exploiting semantic relation, context, and association features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 987–997.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó. Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval '10, pages 33–38.

Christopher Hidey and Kathy McKeown. 2016. Identifying causal relations using parallel wikipedia articles. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1424–1433.

Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, and Gerhard Weikum. 2013. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artif. Intell.*, 194:28–61.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Christopher S.G. Khoo, Jaklin Kornfilt, Sung Hyon Myaeng, and Robert N. Oddy. 1998. Automatic extraction of cause-effect information from newspaper text without knowledge-based inferencing. *Literary and Linguistic Computing*, 13:177–186.

Jinho Kim and Sung-Hyon Myaeng. 2016. Discovering relations to augment a web-scale knowledge base constructed from the web. In *Proceedings of the 6th International Conference on Web Intelligence, Mining and Semantics (WIMS)*, pages 16:1–16:12.

Canasai Kruengkrai, Kentaro Torisawa, Chikara Hashimoto, Julien Kloetzer, Jong-Hoon Oh, and Masahiro Tanaka. 2017. Improving event causality recognition with multiple background knowledge sources using multi-column convolutional neural networks. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI)*, pages 3466–3473.

Joohong Lee, Sangwoo Seo, and Yong Suk Choi. 2019. Semantic relation classification via bidirectional lstm networks with entity-aware attention using latent entity typing.

Joonseok Lee, Ariel Fuxman, Bo Zhao, and Yuanhua Lv. 2015. Leveraging knowledge bases for contextual entity exploration. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 1949–1958.

Jens Lehmann, Chris Bizer, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. 2009. DBpedia - a crystallization point for the web of data. *Journal of Web Semantics*, 7(3):154–165.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2124–2133.

Angrosh Mandya, Danushka Bollegala, Frans Coenen, and Katie Atkinson. 2018. A dataset for inter-sentence relation extraction using distant supervision. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC)*.

Inderjeet Mani, Marc Verhagen, Ben Wellner, Chong Min Lee, and James Pustejovsky. 2006. Machine learning of temporal relations. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL)*, pages 753–760.

David Milne and Ian H. Witten. 2008. An effective, low-cost measure of semantic relatedness obtained from Wikipedia links. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI)*, pages 25–30.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP*, pages 1003–1011.

Paramita Mirza and Sara Tonelli. 2016. Catena: Causal and temporal relation extraction from natural language texts. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING)*, pages 64–75.

Roberto Navigli and Simone Paolo Ponzetto. 2012. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.

Qiang Ning, Zhili Feng, Hao Wu, and Dan Roth. 2018. Joint reasoning for temporal and causal relations. In *Proceedings of the 56th Annual Meeting of the Association for Computational (ACL)*, pages 2278–2288.

Enrique Noriega-Atala, Paul Douglas Hein, Shraddha Satish Thumsi, Zechy Wong, Xia Wang, and Clayton Thomas Morrison. 2018. Inter-sentence relation extraction for associating biological context with events in biomedical texts. In *IEEE International Conference on Data Mining Workshops (ICDM)*, pages 722–731.

Jong-Hoon Oh, Kentaro Torisawa, Canasai Kruengkrai, Ryu Iida, and Julien Kloetzer. 2017. Multi-column convolutional neural networks with causality-attention for why-question answering. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining (WSDM)*, pages 415–424.

Marius Pasca. 2018. Finding needles in an encyclopedic haystack: Detecting classes among wikipedia articles. In *Proceedings of the 2018 World Wide Web Conference (WWW)*, pages 1267–1276.

Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen-tau Yih. 2017. Cross-sentence n-ary relation extraction with graph lstms. *Transactions of the Association for Computational Linguistics*, 5:101–115.

Chris Quirk and Hoifung Poon. 2017. Distant supervision for relation extraction beyond the sentence boundary. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 1171–1182.

Kira Radinsky, Sagie Davidovich, and Shaul Markovitch. 2012. Learning causality for news events prediction. In *Proceedings of the 2012 World Wide Web Conference (WWW)*, pages 909–918.

Ines Rehbein and Josef Ruppenhofer. 2017. Catching the common cause: Extraction and annotation of causal relations and their participants. In *Proceedings of the 11th Linguistic Annotation Workshop*, pages 105–114.

Mehwish Riaz and Roxana Girju. 2010. Another look at causality: Discovering scenario-specific contingency relationships with no supervision. In *Proceedings of the 4th IEEE International Conference on Semantic Computing (ICSC)*, pages 361–368.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Proceedings of the 2010 European Conference on Machine Learning and Knowledge Discovery in Databases: Part III*, ECML PKDD'10, pages 148–163.

Stefan Schoenmackers, Jesse Davis, Oren Etzioni, and Daniel Weld. 2010. Learning first-order horn clauses from web text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1088–1098.

Rebecca Sharp, Mihai Surdeanu, Peter Jansen, Peter Clark, and Michael Hammond. 2016. Creating causal embeddings for question answering with minimal supervision. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 138–148.

Yatian Shen and Xuanjing Huang. 2016. Attention-based convolutional neural network for semantic relation extraction. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2526–2536.

Amit Singhal. 2012. Introducing the knowledge graph: Things, not strings. Corporate blog.

David Tsurel, Dan Pelleg, Ido Guy, and Dafna Shahaf. 2017. Fun facts: Automatic trivia fact extraction from wikipedia. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining (WSDM)*, pages 345–354.

Shikhar Vashishth, Rishabh Joshi, Sai Suman Prayaga, Chiranjib Bhattacharyya, and Partha Talukdar. 2018. Reside: Improving distantly-supervised neural relation extraction using side information. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1257–1266.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A free collaborative knowledge base. *Communications of the ACM*, 57:78–85.

Linlin Wang, Zhu Cao, Gerard de Melo, and Zhiyuan Liu. 2016. Relation classification via multi-level attention cnns. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1298–1307.

Zhigang Wang, Zhixing Li, Juanzi Li, Jie Tang, and Jeff Z. Pan. 2013. Transfer learning based cross-lingual knowledge extraction for wikipedia. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 641–650.

Fei Wu and Daniel S. Weld. 2010. Open information extraction using wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 118–127.

Yan Xu, Ran Jia, Lili Mou, Ge Li, Yunchuan Chen, Yangyang Lu, and Zhi Jin. 2016. Improved relation classification by deep recurrent neural networks with data augmentation. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING)*, pages 1461–1470.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2205–2215.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. 2017. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 35–45.

Sendong Zhao, Ting Liu, Sicheng Zhao, Yiheng Chen, and Jian-Yun Nie. 2016. Event causality extraction based on connectives analysis. *Neurocomputing*, 173(P3):1943–1950.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 207–212.

# Attribute-aware Sequence Network for Review Summarization

**Junjie Li**[1], **Xuepeng Wang**[1], **Dawei Yin**[1], **Chengqing Zong**[2,3,4]
[1] Data Science Lab, JD.com, Beijing, China
[2] National Laboratory of Pattern Recognition, Institute of Automation, CAS, Beijing, China
[3] University of Chinese Academy of Sciences, Beijing, China
[4] CAS Center for Excellence in Brain Science and Intelligence Technology, Beijing, China
{lijunjie46,wangxuepeng1}@jd.com,yindawei@acm.org,cqzong@nlpr.ia.ac.cn

## Abstract

Review summarization aims to generate a condensed summary for a review or multiple reviews. Existing review summarization systems mainly generate summary only based on review content and neglect the authors' attributes (e.g., gender, age, and occupation). In fact, when summarizing a review, users with different attributes usually pay attention to specific aspects and have their own word-using habits or writing styles. Therefore, we propose an Attribute-aware Sequence Network (ASN) to take the aforementioned users' characteristics into account, which includes three modules: an attribute encoder encodes the attribute preferences over the words; an attribute-aware review encoder adopts an attribute-based selective mechanism to select the important information of a review; and an attribute-aware summary decoder incorporates attribute embedding and attribute-specific word-using habits into word prediction. To validate our model, we collect a new dataset *TripAtt*, comprising 495,440 attribute-review-summary triplets with three kinds of attribute information: gender, age, and travel status. Extensive experiments show that ASN achieves state-of-the-art performance on review summarization in both auto-metric ROUGE and human evaluation.

## 1 Introduction

Review summarization aims to generate a condensed summary for a review or multiple reviews[1]. Dominating studies can be divided into two groups: extractive and abstractive approaches. Extractive approaches (Hu and Liu, 2004; Ganesan, 2010) extract sentences or phrases from a review, while abstractive methods (Gerani et al., 2014; Wang and Ling, 2016; Yang et al., 2018a; Li et al., 2019; Gao et al., 2019) summarize a review by employing graph-based or sequence-to-sequence (S2S) models which can generate new phrases and sentences that do not appear in the review.

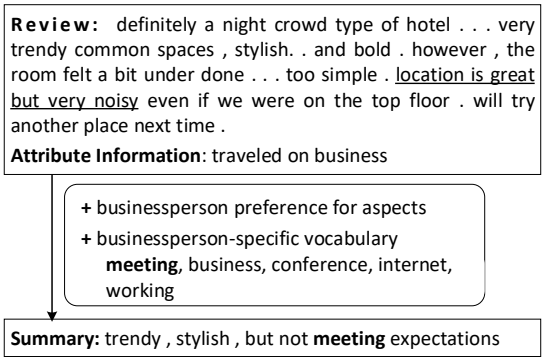


Figure 1: A review-summary pair posted by a businessperson in our dataset shows the effect of attribute information on summarizing review. Underlined words in the review indicate the important sentences that the businessperson care about when summarizing the review. Bold **word** in businessperson-specific vocabulary shows the businessperson's word-using habits may help to generate the summary.

Despite the remarkable progress of previous studies, they typically only focus on review content and neglect the attribute information of users who post these reviews (e.g., gender, age, and occupation). Actually, such information is vital for generating summaries, which contains the following characteristics. (1) People with different attributes may care about different aspects[2]. For example, when choosing hotels, business people may care about ***location*** and ***room*** more than ***price***, while solo travelers may prefer ***price*** more. Figure 1 presents a review posted by a businessper-

[1] Here, we focus on single-review summarization, and we leave adapting our model to multi-review summarization scenario to future work.

[2] Aspects refer to properties (attributes) of products or services, such as ***location*** and ***room*** for the hotel domain.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3000–3010, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

son. Although the hotel is trendy, stylish, and in a good location, it is noisy. The businessperson, therefore, summarizes that the hotel is not suitable for meeting. (2) People with different attributes have different word-using habits or writing styles to summarize a review. According to our statistics (Section 2.2), business people often summarize a review with words like "meeting", "business" and "conference", while solo travelers often utilize "budget", "inexpensive" to summarize their reviews. These attribute-specific words may help generate summaries. Figure 1 shows an example: without considering the attribute information "businessperson", it is hard to generate the word "meeting" when summarizing the review due to its missing. Intuitively, "meeting" belongs to businessperson-specific vocabulary, and such an attribute-specific vocabulary can be incorporated to further improve the summarization performance.

Inspired by the above observations, we propose a model called Attribute-aware Sequence Network (ASN) to consider attribute information into review summarization. Specifically, ASN is based on sequence to sequence models (S2S), which are popular methods in text summarization (Rush et al., 2015; See et al., 2017; Zhu et al., 2018; Li et al., 2018a) and review summarization (Wang and Ling, 2016; Ma et al., 2018). ASN updates over standard S2S are three-fold. First, except for standard encoder and decoder in S2S, we design an attribute encoder, which encodes attribute preference for using words into attribute embedding. Second, an attribute-aware review encoder is proposed to generate attribute-aware review representation. It utilizes a bidirectional-LSTM to encode a review, and then imports an attribute-based selective mechanism to select important information of it to obtain a better review representation. Third, we propose an attribute-aware summary decoder to consider different writing styles of users with different attributes. It incorporates attribute embedding and attribute-specific vocabulary memory into word prediction module to generate summaries.

To validate our approach, we collect a review summarization dataset with user attribute information named *TripAtt* from TripAdvisor website, which contains 495,440 attribute-review-summary triplets with three kinds of attribute information: gender, age and travel status. Extensive experiments show that ASN achieves state-of-the-art performance on review summarization in both autometric ROUGE and human evaluation. Our contributions are as follows:

- To the best of our knowledge, we first propose an attribute-aware S2S-based model named Attribute-aware Sequence Network (ASN) to incorporate attribute information into review summarization.
- Our model adopts an attribute-based selective mechanism to consider different user preferences for review content and applies attribute-specific vocabulary to take the different writing styles of users with different attributes into consideration when generating a summary.
- For the evaluation of review summarization with attribute information, we collect a new dataset named *TripAtt*, which is available at https://github.com/Junjieli0704/ASN.

## 2 Dataset

Since there is no available review summarization dataset with user attribute information, we build a new one named *TripAtt* from TripAdvisor, an online hotel review site. TripAdvisor contains lots of user-generated reviews along with their authors and titles. The title of a review often summarizes the main idea of it; therefore, we take the title as the reference summary of the review. For attribute information, we can first get users' demographic information such as age and gender from the website. Second, users also explicitly label their travel status when booking hotels, such as traveled solo or traveled on business. Therefore, We take gender, age and travel status as our attribute information, and collect near 3 million attribute-review-summary triplets.

However, users may write titles arbitrarily, and it results in many meaningless titles, such as "i will be back again", and "twice in one trip". To remove these noisy samples, we apply filters proposed by Li et al. (2019). Then, we also remove samples that the value of any attributes (gender, age, or travel status) is NULL. Finally, we construct *TripAtt* with 495,440 attribute-review-summary triplets. We randomly split the dataset into 2,000 reviews for test, 2,000 reviews for develop and the rest for training. Table 1 shows statistics of *TripAtt*.

| *TripAtt* | Train | Dev | Test |
|---|---|---|---|
| # Review | 491,440 | 2,000 | 2,000 |
| # Sens of Review | 9.66 | 9.59 | 9.54 |
| # Words of Review | 173.90 | 172.04 | 171.86 |
| # Sens of Summary | 1.0 | 1.0 | 1.0 |
| # Words of Summary | 7.18 | 7.05 | 7.20 |

Table 1: Data statistics of *TripAtt* dataset.

### 2.1 Value distribution of different attributes

Figure 2 presents the value distribution of gender, age, and travel status in *TripAtt*. Males are more likely to travel than females, middle-aged (35-64 years old) users account for near 74%, and around 40% users traveled with couples.

### 2.2 Attribute-specific Vocabulary

Frequent words of a user can reflect the user well. Therefore, we want to mine attribute-specific words from *TripAtt* to model attributes. We first merge all summaries posted by users with the same group (such as male or 35-49 years old users) into a document. Then we compute $tf$-$idf$ scores[3] for each word appears in the document, and we finally select top-$N$ words for different groups. The last column in Figure 2 shows top-5 words in different attribute-specific vocabularies. We find that these words can actually reflect different groups well. For example, female users often utilize "lovely", "beautiful", "cute" to summarize review while these words rarely appear in summaries of male users. Users traveled with family often care about whether the hotel is suitable for their kids since they usually summarize reviews using "kids", "disney", and "parks", while business people frequently consider whether the hotel is suitable for meeting.

## 3 Attribute-aware Sequence Network

### 3.1 Problem Formulation

Suppose we have a corpus $D$ with $|D|$ attribute-review-summary triplets, and each triplet contains a review $x = (x_1, x_2, ..., x_{|x|})$, a summary $y = (y_1, y_2, ..., y_{|y|})$ and an attribute vector $a = (a_1, a_2, ..., a_{|a|})$ which records $|a|$ kinds of attribute information of $x$'s author. Since we have three kinds of attribute information (gender, age, and travel status), $|a|$ equals to 3. Classical review summarization is to generate $y$ from $x$, while our goal needs to consider $a$'s characteristics on summarizing reviews when generating $y$.

[3]Using *tf-idf* scores means we do not include too general terms that all users commonly use, because they do not help model the specific group.

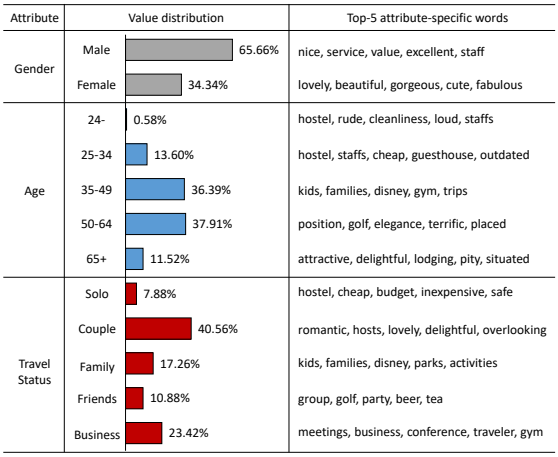


Figure 2: Value distribution and top-5 words in attribute-specific vocabulary of gender, age, and travel status in *TripAtt* dataset.

### 3.2 Model Framework

As shown in Figure 3, our model consists of three modules: attribute encoder, attribute-aware review encoder and attribute-aware summary decoder. Attribute encoder is based on attribute-specific words obtained from Section 2.2, which not only stores these words into attribute-specific vocabulary memory, but also utilizes multi-layer perceptron to merge them into an attribute embedding. Then we introduce four strategies to consider attribute embedding and attribute-specific vocabulary memory. Equipped with attribute embedding, our attribute-aware review encoder can select vital information from review representation. Importing attribute embedding and attribute-specific vocabulary memory into word prediction process of our attribute-aware summary decoder, our model can generate summary well.

### 3.3 Attribute Encoder

This module will produce attribute embedding and attribute-specific vocabulary memory. Suppose we select top-$K$ attribute-specific words for each attribute in $a$, and then we concatenate these words to get attribute-specific vocabulary $A = (A_1, ..., A_K, ..., A_{2K}, ..., A_{|a|K})$, where word indexes between $(i-1) \times K + 1$ and $i \times K$ in $A$ belong to attribute $a_i$. After that, we use an embedding matrix $\mathbf{E}_v$ to embed each word $\{A_i\}_{i=1}^{|a|K}$

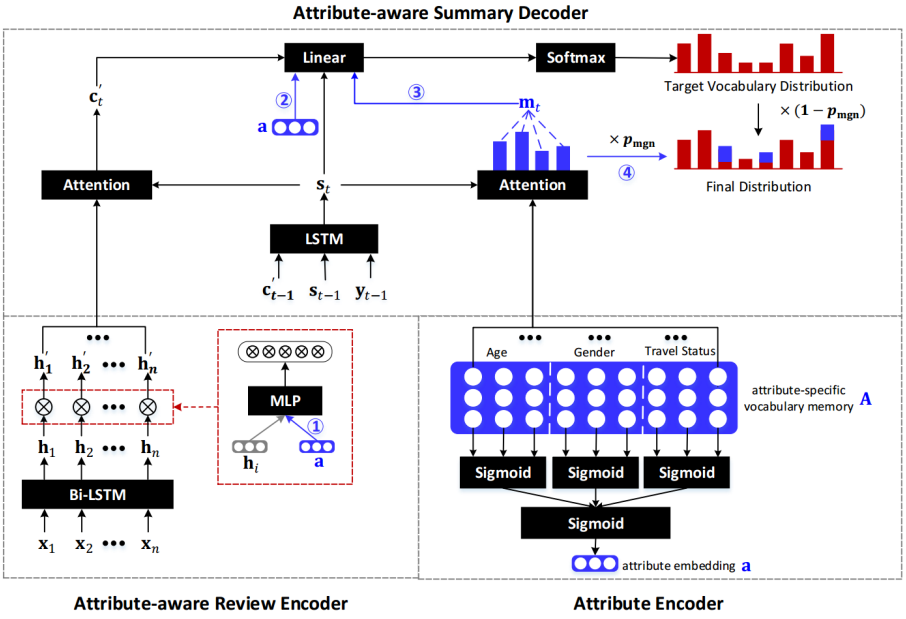


Figure 3: The architecture of Attribute-aware Sequence Network (ASN). ASN encodes two kinds of attribute information, attribute embedding (**a**) and attribute-specific vocabulary memory (**A**), into its two basic modules (*Attribute-aware Review Encoder* and *Attribute-aware Summary Decoder*). ①, and ② show strategies based on attribute embedding, and represent Attribute Selection strategy, and Attribute Prediction strategy, respectively. ③ and ④ indicate strategies based on attribute-specific vocabulary memory, and represent Attribute Memory Prediction strategy and Attribute Memory Generation strategy, respectively.

into vector $\{\mathbf{A}_i\}_{i=1}^{|a|K}$, and we get matrix $\mathbf{A}$, which is also called attribute-specific vocabulary memory. Then, we use a nonlinear layer to merge $A$'s words belonging to attribute $a_i$ into embedding $\mathbf{a}_i$ (See Equation (1)), which can only represent attribute $a_i$. After that, we merge these $|a|$ attribute embeddings $\mathbf{a}_1, \mathbf{a}_1, ..., \mathbf{a}_{|a|}$ into a nonlinear layer to get attribute embedding $\mathbf{a}$ (See Equation (2)), which is used to represent attribute vector $a$.

$$\mathbf{a}_i = \sigma(\mathbf{W}_a \mathbf{A}_{(i-1)\times K+1:i\times K} + \mathbf{b}_a) \quad (1)$$

$$\mathbf{a} = \sigma(\sum_{i=1}^{i=|a|} \mathbf{w}_a \mathbf{a}_i + b_a) \quad (2)$$

where $\mathbf{W}_a$, $\mathbf{w}_a$, $\mathbf{b}_a$, and $b_a$ are learnable parameters, and $\sigma$ denotes sigmoid function.

### 3.4 Attribute-aware Review Encoder

Given review $x$, it first embeds each word $x_i$ into vector $\mathbf{x}_i$ using embedding matrix $\mathbf{E}_v$, which is the same embedding matrix in attribute encoder module. Then, these word vectors are fed into a single-layer bidirectional LSTM one by one, producing a sequence of encoder hidden states $\mathbf{h}_i$.

Classical review encoder utilizes $\mathbf{h}_i$ to represent review $x$ and ends here. However, we find that users with different backgrounds pay attention to different content of a review. Inspired by (Zhou et al., 2017), we propose an attribute-based selective mechanism to select the important information from review for users with different attributes. The selective mechanism can construct a tailored representation of review $x$ by considering $a$. In detail, our attribute-based selective network takes attribute vector $\mathbf{a}$ and the encoder hidden state $\mathbf{h}_i$ as input, and outputs a gate vector $\mathbf{gate}_i$ to select $\mathbf{h}_i$.

$$\mathbf{gate}_i = \sigma(\mathbf{W}_k[\mathbf{h}_i; \mathbf{a}] + \mathbf{b}_k) \quad (3)$$

$$\mathbf{h}'_i = \mathbf{h}_i \odot \mathbf{gate}_i \quad (4)$$

where $\mathbf{W}_k$, $\mathbf{b}_k$ are learnable parameters, $[;]$ is the concatenating operator, $\sigma$ denotes sigmoid function, and $\odot$ is element-wise multiplication.

From Equation (4), we find $\mathbf{gate}_i$ is a vector whose value is between 0 and 1. A high value means most of the information in $\mathbf{h}_i$ passed from the filter, which results in the word $\mathbf{x}_i$ is important. This is the first strategy to consider attributes called ***Attribute Selection strategy***.

### 3.5 Attribute-aware Summary Decoder

At each decoding time step $t$, the decoder (a single-layer unidirectional LSTM) receives previous word embedding to obtain the new hidden state $\mathbf{s}_t$. Then it computes context vector $\mathbf{c}_t$

for time step $t$ through the attention mechanism: where MLP stands for multi-layer perceptrons and $\alpha_{t,i}$ matches the importance score between current decoder state $\mathbf{s}_t$ and the encoder hidden state $\mathbf{h}_i^{'}$.

The classical summary decoder combines the context vector $\mathbf{c}_t$ and the decoder state $\mathbf{s}_t$, and then feeds the merged vector into a linear layer to produce the vocabulary distribution.

However, when generating a summary, users with different attributes may have their own vocabulary. Thus it is natural to take attribute-specific vocabulary memory $\mathbf{A}$ into consideration when predicting output vocabulary distribution and different words in $\mathbf{A}$ may have different effects. Thus, we utilize an attention mechanism to extract important words in $\mathbf{A}$ when obtaining vocabulary state $\mathbf{m}_t$.

$$\beta_{t,k} = \frac{\exp(\text{MLP}(\mathbf{A}_k, \mathbf{s}_t))}{\sum_k \exp(\text{MLP}(\mathbf{A}_k, \mathbf{s}_t))} \quad (5)$$

$$\mathbf{m}_t = \sum_k \beta_{t,k} \mathbf{A}_k \quad (6)$$

where $\beta_{t,k}$ measures the importance score between current decoder state $\mathbf{s}_t$ and the $k$-th word in attribute-aware vocabulary memory $A_k$.

Then we combine context vector $\mathbf{c}_t$, the decoder state $\mathbf{s}_t$, and $\mathbf{m}_t$ into the readout state $\mathbf{r}_t$. Besides, we can also enhance the readout state $\mathbf{r}_t$ by combining attribute vector $\mathbf{a}$. After that, we feed the readout state into a linear layer to produce the vocabulary distribution $P_{\text{voc}}$.

$$\mathbf{r}_t = \mathbf{W}_r[\mathbf{c}_t; \mathbf{s}_t; \mathbf{a}; \mathbf{m}_t] + \mathbf{b}_r \quad (7)$$

$$P_{\text{voc}} = \text{softmax}(\mathbf{W}_o \mathbf{r}_t + \mathbf{b}_o) \quad (8)$$

where $\mathbf{W}_r$ and $\mathbf{b}_r$ are learnable parameters. The strategies of adding attribute vector $\mathbf{a}$ and vocabulary state $\mathbf{m}_t$ into readout state $\mathbf{r}_t$ are called ***Attribute Prediction strategy*** and ***Attribute Memory Prediction strategy***, respectively.

Last but not least, inspired by (See et al., 2017), we also propose a soft copy mechanism to copy attribute-specific words in generating summaries, which is the 4-*th* strategy called ***Attribute Memory Generation strategy***.

The generation probability $p_{\text{mgn}} \in [0, 1]$ for timestep $t$ is calculated from the context vector $\mathbf{c}_t$, the decoder state $\mathbf{s}_t$ and the vocabulary state $\mathbf{m}_t$:

$$p_{\text{mgn}} = \sigma(\mathbf{W}_{mg}[\mathbf{c}_t; \mathbf{s}_t; \mathbf{m}_t] + \mathbf{b}_{mg}) \quad (9)$$

where $\mathbf{W}_{mg}, \mathbf{b}_{mg}$ are learnable parameters, $[;]$ is the concatenating operator and $\sigma$ is the sigmoid function. Next $p_{\text{mgn}}$ is used as a soft switch to choose between generating a word from the target vocabulary $V_t$ or coping a word from attribute-specific vocabulary.

$$P(w) = (1 - p_{\text{mgn}}) P_{\text{voc}}(w) + p_{\text{mgn}} \sum_{k:A_k=w} \beta_{t,k} \quad (10)$$

The first part in Equation (10) represents generating words from our vocabulary, and the second part indicates coping words from attribute-specific vocabulary memory, respectively.

### 3.6 Objective Function

Our goal is to maximize the output summary probability given the input sentence. Therefore, we optimize the negative log-likelihood loss function:

$$L = -\frac{1}{|D|} \sum_{(x,y) \in D} \log p(y|x) \quad (11)$$

## 4 Experiments

### 4.1 Evaluation Metric

We exploit ROUGE (Lin, 2004) as our evaluation metric. ROUGE scores reported in this paper are computed by Pyrouge package [4].

### 4.2 Comparison Methods

In the experiments, we compare our model with several strong baseline methods, which can be divided into two types: extractive and abstractive approaches. **LEAD1** is an extractive approach which selects the first sentence in review as summary. **LEXRANK** (Erkan and Radev, 2004) is also a famous extractive approach that computes text centrality based on PageRank algorithm. **TEXTRANK** (Mihalcea and Tarau, 2004) is an unsupervised algorithm based on weighted-graphs. **S2SATT** is a sequence to sequence model with attention implemented by us. **SEASS** (Zhou et al., 2017) employs a selective encoding model to control the information flow from encoder to decoder. **PGN** (See et al., 2017) copies words from the source text via pointing, while retaining the ability to produce novel words through the generator.

### 4.3 Implementation Details

**Model Parameters** The vocabulary is collected from the *TripAtt* training data. We lowercase the text, and there are 362,103 unique word types. We

[4] pypi.python.org/pypi/pyrouge/0.1.3

| Models | RG-1 | RG-2 | RG-L |
|---|---|---|---|
| LEAD1 | 11.59 | 2.41 | 10.21 |
| LEXRANK | 8.80 | 1.19 | 7.87 |
| TEXTRANK | 10.47 | 1.71 | 9.30 |
| S2SATT | 20.61 | 5.96 | 18.95 |
| SEASS | 20.34 | 5.72 | 18.76 |
| PGN | 20.68 | 6.20 | 19.13 |
| ASN | **21.52**$^*$ | **6.61**$^*$ | **19.74**$^*$ |

Table 2: ROUGE F1 scores (%) on the test set. RG in the Table denotes ROUGE. Models and baselines in the top half are extractive methods, while those in the bottom half are abstractive ones. The best performance is in **bold**. The superscript $^*$ indicates ASN performs significantly better than other models as given by the 95% confidence interval in the official ROUGE script.

use the top 30,000 words as the model vocabulary since they can cover 99.01% of the training data.

**Model Training** We set the batch size to 128. We truncate the review to 200 tokens, which is done to expedite training and testing. However we also find that truncating the review can raise the performance of the model[5]. We use the development set to choose the size of attribute-specific vocabulary and set it to 100. We also use Adam (Kingma and Ba, 2015), dropout (Srivastava et al., 2014) and gradient clipping (Pascanu et al., 2013) to make our model robust. We set the word embedding size to 128, and all LSTM hidden state sizes to 200. We use Adam as our optimizing algorithm. For the hyperparameters of Adam optimizer, we set the learning rate $\alpha = 0.001$, two momentum parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$ respectively, and $\epsilon = 10^{-8}$. We use dropout with probability $p$ = 0.2. We also apply gradient clipping with range $[-5, 5]$.

**Model Testing** At test time, We use beam search with a beam size of 4.

### 4.4 Results

Our results are given in Table 2. For extractive methods, we can see that LEAD1 performs best. However, it only obtains 11.59 ROUGE-1, 2.41 ROUGE-2, and 10.21 ROUGE-L F1 scores. The reason is that summaries in *TripAtt* are very succinct[6] and they often cover contents across several sentences, such as the summary in Figure 1.

For abstractive methods, we find that S2SATT is better than all extractive methods. After considering selective mechanism into S2SATT, the performance of SEASS decreases slightly. Because the selective mechanism proposed by SEASS is designed for sentence summarization, which may not be suitable for summarizing reviews. The average length of the input is less than 40 in Zhou et al. (2017), while the length in *TripAtt* is about 170. When incoporating copy mechanism into S2SATT, PGN obtains better performance.

Finally, after considering our proposed attribute encoder and four attribute-based strategies, ASN performs significantly better than all previous methods. Compared to S2SATT, our model has a 0.91 ROUGE-1, 0.65 ROUGE-2, and 0.79 ROUGE-L gains, which shows explicitly modeling attribute-related characteristics can indeed improve summarization quality. Our model also surpasses PGN by 0.84 ROUGE-1, 0.41 ROUGE-2, and 0.61 ROUGE-L and achieves the state-of-the-art performance on review summarization.

### 4.5 Human Evaluation on Aspect-level Coverage

Previous experiments show that ASN is better than other baselines when evaluating on ROUGE. However, ROUGE is only a word-based metric, which can not measure the semantic between two references. Table 3 illustrates an example. Since the summary generated by S2SATT contains more overlapping words with the gold summary than the one generated by ASN contains, S2SATT obtains higher ROUGE scores than ASN. However, from the view of aspects, ASN may be better. Because its summary describes ***location*** and ***service*** of a hotel, which are consistent with the gold result, while S2SATT's summary misses ***service***. Therefore, except for ROUGE, we want also to evaluate aspect-level coverage of different systems.

To perform the aspect-level evaluation, we first define seven aspects: ***location***, ***service***, ***room***, ***value***, ***facility***, ***food***, and ***hotel***, where ***hotel*** describes the overall attitude. Then, we randomly sample 1000 attribute-review-summary triplets from test set and generate summaries of these reviews using S2SATT, PGN and ASN. After that,

[5]Indeed, we found that using only the first 200 tokens of the review yields higher ROUGE scores than using all tokens.

[6]We observe in Table 1 that the average length of summary is about seven.

| | A sample in test set | RG-1(%) | RG-2(%) | RG-L(%) | Aspect label |
|---|---|---|---|---|---|
| S2SATT: | good location , but small rooms | 46.15 | 36.36 | 46.15 | location, room |
| ASN: | good location , stuff needs improvements | 33.33 | 20 | 33.33 | location, service |
| Gold: | 3+ good location , but some lacks in service | - | - | - | location, service |

Table 3: A sample in test set shows high ROUGE may not result in better performance.

| Models | Precision | Recall | F1 |
|---|---|---|---|
| S2SATT | 0.452 | 0.480 | 0.466 |
| PGN | 0.472 | 0.503 | 0.487 |
| ASN | **0.491** | **0.530** | **0.510** |

Table 4: Aspect-level Precision, Recall, and F1 scores (%) for different systems.

we ask two students to label aspects to these generated summaries and the reference summaries[7]. Finally, we compute aspect-level precision, recall, and F1 for different systems.

Table 4 shows the aspect-level result, and we find that ASN outperforms other models by a large margin, which shows summaries generated by our model can not only contain more correct words, but also in a higher consistency on aspects with references.

## 5 Discussions

In this section, we study the effect of different attributes, different attribute-specific strategies, and different attribute-specific vocabulary size on review summarization.

### 5.1 Effects of Different Attributes

To understand which kind of attribute information is the most important in review summarization, we perform an ablation study and give the results in Table 5.

First, all these kinds of attribute information are helpful for review summarization. Adding one kind of attribute information can obtain at least 0.41 ROUGE-1, 0.18 ROUGE-2 and 0.22 ROUGE-L gains. Second, travel status information is the most important attribute for review summarization in *TripAtt*. Because the travel status is a domain-dependent attribute, while others are domain-independent ones.

[7]Some examples about how to label aspects to summaries are shown in the last column of Table 3.

| Attribute Information | RG-1 | RG-2 | RG-L |
|---|---|---|---|
| None | 20.61 | 5.96 | 18.95 |
| Age | 21.04 | 6.19 | 19.39 |
| Gender | 20.98 | 6.14 | 19.17 |
| Travel Status | 21.09 | 6.22 | 19.41 |
| Age + Gender | 21.10 | 6.25 | 19.50 |
| Age + Travel Status | 21.21 | 6.46 | 19.63 |
| Gender + Travel Status | 21.08 | 6.33 | 19.48 |
| ASN | 21.52 | 6.61 | 19.74 |

Table 5: Ablation study on review summarization with attribute information.

| line | ASel | APre | AMP | AMG | RG-1 | RG-2 | RG-L |
|---|---|---|---|---|---|---|---|
| 1 | – | – | – | – | 20.61 | 5.96 | 18.95 |
| 2 | ✓ | – | – | – | 20.78 | 6.09 | 19.07 |
| 3 | – | ✓ | – | – | 21.03 | 6.15 | 19.21 |
| 4 | – | – | ✓ | – | 21.13 | 6.24 | 19.29 |
| 5 | – | – | – | ✓ | 21.08 | 6.19 | 19.24 |
| 6 | – | ✓ | ✓ | ✓ | 21.13 | 6.40 | 19.42 |
| 7 | ✓ | – | ✓ | ✓ | 21.52 | 6.49 | 19.70 |
| 8 | ✓ | ✓ | – | ✓ | 21.42 | 6.45 | 19.48 |
| 9 | ✓ | ✓ | ✓ | – | 21.15 | 6.38 | 19.37 |
| 10 | ✓ | ✓ | ✓ | ✓ | 21.52 | 6.61 | 19.74 |

Table 6: Effects of different attribute-based strategies on review summarization. "✓" means our model considers the specific strategy, while "–" means not. When there is no user-based strategies considered in our model, our model degrades into S2SATT (line 1).

### 5.2 Effects of different strategies

In this paper, we propose four attribute-based strategies to construct our attribute-aware review summarization model, which contains Attribute Selection strategy (ASel), Attribute Prediction strategy (APre), Attribute Memory Prediction strategy (AMP) and Attribute Memory Generation strategy (AMG). To evaluate the effect of each strategy on review summarization, we perform an ablation study and report results in Table 6.

First, we observe that models with only one kind of attribute-based strategy (line 2-5) can at least exceed S2SATT by (+0.17 ROUGE-1, +0.13 ROUGE-2, +0.12 ROUGE-L) points. It shows that all these strategies improve the performance of re-

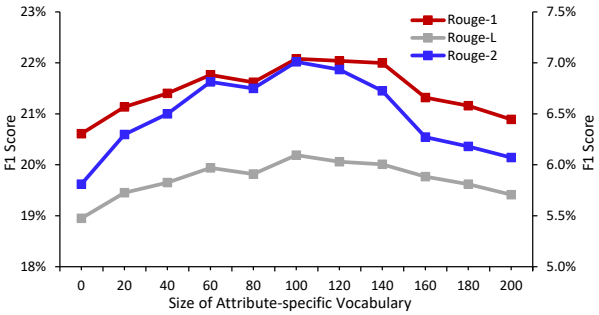


Figure 4: Effects of attribute-specific vocabulary size on review summarization on development set of *TripAtt*. When there is no any attribute-specific vocabulary (the size is 0) in ASN, our model degrades into S2SATT. The primary axis is for ROUGE-1 and ROUGE-L, and the second axis is for ROUGE-2.

view summarization. APre and AMG are the two most effective strategies, because they directly affect the word prediction module in ASN.

Second, models which deletes one kind of attribute-based strategy from ASN (line 6-9) will descend at least 0.11 in ROUGE-2 compared with ASN (line 10). It shows all our four user-based strategies are complementary. The most complementary strategies are ASel and AMG. The reason for ASel is that it is applied on the encoder module of ASN, while others are applied on the decoder module. For AMG, it differs from other decoder module strategies by affecting the decoding process through adding one more vocabulary distribution, while other strategies (APre and AMP) only add more features to classical word prediction module.

Third, ASN obtains the best result when considering all these strategies.

### 5.3 Effects of Attribute-specific Vocabulary Size

Since the attribute-specific vocabulary is the cornerstone for our attribute-aware model, we perform a test to detect the effect of its size on review summarization and show the result in Figure 4. First, we find that considering attribute-specific vocabulary can indeed improve the performance of review summarization, even though when the vocabulary size is very small (such as 20). Second, all curves increase firstly, and decrease with the increase of attribute-specific vocabulary size. It indicates that small vocabulary may be helpful for ASN, however large vocabulary may import much noise and be harmful to ASN. The best performance is obtained when the attribute-specific vocabulary size is 100, that's the reason why we set our attribute-vocabulary size to 100.

**Review:** i had the pleasure of staying at this hotel for a company ***meeting***. when you get to mco, call the hotel and they have a complimentary shuttle that will pick you up and bring you to the hotel. ... overall a great stay and if you need a place for your ***meeting*** , definately book here . . . . you wont be disappointed !
**Attribute Information:** 35-49, female, **traveled on business**

| | |
|---|---|
| **S2SAtt:** | very nice hotel with great staff |
| **PGN:** | very nice hotel , great staff |
| **ASN:** | nice hotel for a ***meeting*** |
| **Gold:** | great for a ***meeting*** |

(a) A sample shows the effect of travel state on review summarization

**Review:** estilo fashion hotel has n't been open very long but has set its standards high ! rooms are clean but not huge but everything is new and in good condition . staff are very helpful and efficient . our bedroom had a comfortable bed and was very modern . ...
**Attribute Information:** 50-64, **female**, traveled as a couple

| | |
|---|---|
| **S2SAtt:** | modern hotel in great location |
| **PGN:** | great location , excellent service |
| **ASN:** | ***lovely*** hotel and excellent service |
| **Gold:** | ***lovely*** hotel with efficient staff ! |

(b) A sample shows the effect of gender on review summarization

Figure 5: Two samples in *TripAtt* show the effect of attributes on review summarization

### 5.4 Case Study

We present two cases from *TripAtt* that show the effect of attribute on review summarization in Figure 5.

Figure 5 (a) shows the effect of travel status on review summarization. Businessperson often books a hotel for working. In this case, the businesswoman stayed at the hotel for a company meeting. Whether the hotel is meet for the requirement may be the most important factor for her. She summarizes "great for a meeting". S2SATT could not get the point. Even though "meeting" appears in the review, PGN that has the effect of copying words from it also fails to generate the word. Our model containing businessperson-specific vocabulary can generate the word well and obtain a better summarization.

Figure 5 (b) shows the effect of gender on review summarization. Word-using habits in different genders are different. Female users often utilize "lovely", "beautiful", "cute" to summarize review while these words rarely appear in summaries from male users. Without considering the gender bias, S2SATT and PGN can not generate the summarization well. Incorporating such writ-

ing styles of female users in ASN, our model can generate "lovely" correctly, although it does not appear in the review.

## 6 Related Work

Review summarization belongs to sentiment analysis (Liu, 2016; Xia et al., 2015), which is a large area in natural language processing and contains sentiment classification (Li and Zong, 2008; Xia et al., 2011; Li et al., 2016, 2018b), emotion detection (Li et al., 2015), spam detection (Wang et al., 2017) and so on. There are two mainstream approaches for the problem: extractive and abstractive approaches.

A key task in extractive methods (Hu and Liu, 2004; Lerman et al., 2009; Xiong and Litman, 2014; Kunneman et al., 2018) is to identify important text units. For example, Hu and Liu (2004) first recognize the frequent product features and then attach extracted opinion sentences to the corresponding feature. Xiong and Litman (2014) exploit review helpfulness for review summarization. However, many studies (Carenini et al., 2013; Fabbrizio et al., 2014) have shown that abstractive approaches may be more appropriate for summarizing evaluative text than extractive ones. That is also the reason why we build our attribute-aware model based on abstractive methods.

Abstractive approaches (Ganesan, 2010; Gerani et al., 2014; Wang and Ling, 2016) are also very popular methods in review summarization. For example, Ganesan (2010) first represent review as token-based graphs based on the token order in the string and then rank summary candidates by scoring paths after removing redundant information from the graph. Gerani et al. (2014) utilize discourse structure of review to identify important aspects and then design a set of templates to generate summarizations. Wang and Ling (2016) propose an attention-based neural network model for generating abstractive summaries of opinionated text.

All of these studies focus on review summarization in the multiple review scenario, while our work focuses on the single review scenario. Recent review summarization studies (Ma et al., 2018; Yang et al., 2018a,b) also focus on the scenario. Ma et al. (2018) jointly models review summarization and sentiment classification in a unified framework. Yang et al. (2018a) study the aspect/sentiment-aware abstractive review summarization in an end-to-end manner. They mainly generate summary only based on review content and overlook the crucial influences of users. The most related work is Li et al. (2019). They study the personalized review summarization issue and also neglect the effect of user attributes on review summarization. Our proposed model fills this gap in the literature.

## 7 Conclusion and Future Work

In this paper, we propose an Attribute-aware Sequence Network (ASN) to consider attribute information into review summarization. ASN imports attribute-specific vocabulary to model attribute information and utilizes four attribute-based strategies to build attribute-aware review encoder and attribute-aware summary decoder. To validate our model, we construct a new dataset (*TripAtt*). Extensive experiments on *TripAtt* show that ASN achieves state-of-the-art performance on review summarization.

## 8 Acknowledgments

The research work descried in this paper has been supported by the National Key Research and Development Program of China under Grant No. 2017YFC0820700 and the Natural Science Foundation of China under Grant No. U1836221.

## References

Giuseppe Carenini, Jackie Chi Kit Cheung, and Adam Pauls. 2013. Multi-document summarization of evaluative text. *Computational Intelligence*, 29(4):545576.

Günes Erkan and Dragomir R. Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Intell. Res.*, 22:457–479.

Giuseppe Di Fabbrizio, Amanda Stent, and Robert Gaizauskas. 2014. A hybrid approach to multi-document summarization of opinions in reviews. In *INLG*, pages 54–63.

Kavita Ganesan. 2010. Opinosis:a graph-based approach to abstractive summarization of highly redundant opinions. In *COLING*, pages 340–348.

Shen Gao, Zhaochun Ren, Yihong Zhao, Dongyan Zhao, Dawei Yin, and Rui Yan. 2019. Product-aware answer generation in e-commerce question-answering. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, WSDM 2019, Melbourne, VIC, Australia, February 11-15, 2019*, pages 429–437.

Shima Gerani, Yashar Mehdad, Giuseppe Carenini, Raymond T. Ng, and Bita Nejat. 2014. Abstractive summarization of product reviews using discourse structure. In *EMNLP*, pages 1602–1613.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *SIGKDD*, pages 168–177.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Florian Kunneman, Sander Wubben, Antal van den Bosch, and Emiel Krahmer. 2018. Aspect-based summarization of pros and cons in unstructured product reviews. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 2219–2229.

Kevin Lerman, Sasha Blair-Goldensohn, and Ryan Mcdonald. 2009. Sentiment summarization: evaluating and learning user preferences. In *EACL*, pages 514–522.

Haoran Li, Junnan Zhu, Tianshang Liu, Jiajun Zhang, and Chengqing Zong. 2018a. Multi-modal sentence summarization with modality attention and image filtering. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden.*, pages 4152–4158.

Junjie Li, Haoran Li, and Chengqing Zong. 2019. Towards personalized review summarization via user-aware sequence network. In *AAAI*.

Junjie Li, Haitong Yang, and Chengqing Zong. 2016. Sentiment classification of social media text considering user attributes. In *NLPCC*, pages 583–594.

Junjie Li, Haitong Yang, and Chengqing Zong. 2018b. Document-level multi-aspect sentiment classification by jointly modeling users, aspects, and overall ratings. In *COLING*, pages 925–936.

Shoushan Li, Lei Huang, Rong Wang, and Guodong Zhou. 2015. Sentence-level emotion classification with label and context dependence. In *ACL*, pages 1045–1053.

Shoushan Li and Chengqing Zong. 2008. Multi-domain sentiment classification. In *ACL*, pages 257–260.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*.

Bing Liu. 2016. *Sentiment analysis: Mining opinions, sentiments, and emotions*. Cambridge University Press, University of Cambridge.

Shuming Ma, Xu Sun, Junyang Lin, and Xuancheng Ren. 2018. A hierarchical end-to-end model for jointly improving text summarization and sentiment classification. In *IJCAI*, pages 4251–4257.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into text. In *Proceedings of EMNLP 2004*, pages 404–411, Barcelona, Spain. Association for Computational Linguistics.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *ICML*, pages 1310–1318.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *EMNLP*, pages 379–389.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *ACL*, pages 1073–1083.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Lu Wang and Wang Ling. 2016. Neural network-based abstract generation for opinions and arguments. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 47–57, San Diego, California. Association for Computational Linguistics.

Xuepeng Wang, Kang Liu, and Jun Zhao. 2017. Handling cold-start problem in review spam detection by jointly embedding texts and behaviors. In *ACL*, pages 366–376.

Rui Xia, Feng Xu, Chengqing Zong, Qianmu Li, Yong Qi, and Tao Li. 2015. Dual sentiment analysis: Considering two sides of one review. *IEEE Trans. Knowl. Data Eng.*, 27(8):2120–2133.

Rui Xia, Chengqing Zong, and Shoushan Li. 2011. Ensemble of feature sets and classification algorithms for sentiment classification. *Information Science*, 181(6):1138–1152.

Wenting Xiong and Diane Litman. 2014. Empirical analysis of exploiting review helpfulness for extractive summarization of online reviews. *Grantee Submission*.

Min Yang, Qiang Qu, Ying Shen, Qiao Liu, Wei Zhao, and Jia Zhu. 2018a. Aspect and sentiment aware abstractive review summarization. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 1110–1120.

Min Yang, Qiang Qu, Jia Zhu, Ying Shen, and Zhou Zhao. 2018b. Cross-domain aspect/sentiment-aware abstractive review summarization. In *Proceedings*

*of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*, pages 1531–1534.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2017. Selective encoding for abstractive sentence summarization. In *ACL*, pages 1095–1104.

Junnan Zhu, Haoran Li, Tianshang Liu, Yu Zhou, Jiajun Zhang, and Chengqing Zong. 2018. MSMO: Multimodal summarization with multimodal output. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4154–4164.

# Extractive Summarization of Long Documents by Combining Global and Local Context

**Wen Xiao and Giuseppe Carenini**
Department of Computer Science
University of British Columbia
Vancouver, BC, Canada, V6T 1Z4
{xiaowen3, carenini}@cs.ubc.ca

## Abstract

In this paper, we propose a novel neural single-document extractive summarization model for long documents, incorporating both the global context of the whole document and the local context within the current topic. We evaluate the model on two datasets of scientific papers , Pubmed and arXiv, where it outperforms previous work, both extractive and abstractive models, on ROUGE-1, ROUGE-2 and METEOR scores. We also show that, consistently with our goal, the benefits of our method become stronger as we apply it to longer documents. Rather surprisingly, an ablation study indicates that the benefits of our model seem to come exclusively from modeling the local context, even for the longest documents.

## 1 Introduction

Single-document summarization is the task of generating a short summary for a given document. Ideally, the generated summaries should be fluent and coherent, and should faithfully maintain the most important information in the source document. This is a very challenging task, because it arguably requires an in-depth understanding of the source document, and current automatic solutions are still far from human performance (Allahyari et al., 2017).[VII]

Single-document summarization can be either extractive or abstractive. Extractive methods typically pick sentences directly from the original document based on their importance, and form the summary as an aggregate of these sentences. Usually, summaries generated in this way have a better performance on fluency and grammar, but they may contain much redundancy and lack in coherence across sentences. In contrast, abstractive methods attempt to mimic what humans do by first extracting content from the source document and then produce new sentences that aggregate and organize the extracted information. Since the sentences are generated from scratch they tend to have a relatively worse performance on fluency and grammar. Furthermore, while abstractive summaries are typically less redundant, they may end up including misleading or even utterly false statements, because the methods to extract and aggregate information form the source document are still rather noisy.

In this work, we focus on extracting informative sentences from a given document (without dealing with redundancy), especially when the document is relatively long (e.g., scientific articles).

Most recent works on neural extractive summarization have been rather successful in generating summaries of short news documents (around 650 words/document) (Nallapati et al., 2016) by applying neural Seq2Seq models (Cheng and Lapata, 2016). However when it comes to long documents, these models tend to struggle with longer sequences because at each decoding step, the decoder needs to learn to construct a context vector capturing relevant information from all the tokens in the source sequence (Shao et al., 2017).

Long documents typically cover multiple topics. In general, the longer a document is, the more topics are discussed. As a matter of fact, when humans write long documents they organize them in chapters, sections etc.. Scientific papers are an example of longer documents and they follow a standard discourse structure describing the problem, methodology, experiments/results, and finally conclusions (Suppe, 1998).

To the best of our knowledge only one previous work in extractive summarization has explicitly leveraged section information to guide the generation of summaries (Collins et al., 2017). However, the only information about sections fed into their

[VII] Sentence coloring and Roman numbering will be explained in the result sub-section 4.5.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3011–3021, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

sentence classifier is a categorical feature with values like *Highlight*, *Abstract*, *Introduction*, etc., depending on which section the sentence appears in.

In contrast, in order to exploit section information, in this paper we propose to capture a distributed representation of both the global (the whole document) and the local context (e.g., the section/topic) when deciding if a sentence should be included in the summary

Our main contributions are as follows: **(i)** In order to capture the local context, we are the first to apply LSTM-minus to text summarization. LSTM-minus is a method for learning embeddings of text spans, which has achieved good performance in dependency parsing (Wang and Chang, 2016), in constituency parsing (Cross and Huang, 2016), as well as in discourse parsing (Liu and Lapata, 2017). With respect to more traditional methods for capturing local context, which rely on hierarchical structures, LSTM-minus produces simpler models i.e. with less parameters, and therefore faster to train and less prone to overfitting. **(ii)** We test our method on the Pubmed and arXiv datasets and results appear to support our goal of effectively summarizing long documents. In particular, while overall we outperform the baseline and previous approaches only by a narrow margin on both datasets, the benefit of our method become much stronger as we apply it to longer documents. Furthermore, in an ablation study to assess the relative contributions of the global and the local model we found that, rather surprisingly, the benefits of our model seem to come exclusively from modeling the local context, even for the longest documents.[VI] **(iii)** In order to evaluate our approach, we have created oracle labels for both Pubmed and arXiv (Cohan et al., 2018), by applying a greedy oracle labeling algorithm. The two datasets annotated with extractive labels will be made public.[1]

## 2 Related work

### 2.1 Extractive summarization

Traditional extractive summarization methods are mostly based on explicit surface features (Radev et al., 2004), relying on graph-based methods (Mihalcea and Tarau, 2004), or on submodular maximization (Tixier et al., 2017). Benefiting from the success of neural sequence models in other NLP tasks, Cheng and Lapata (2016) propose a novel approach to extractive summarization based on neural networks and continuous sentence features, which outperforms traditional methods on the DailyMail dataset. In particular, they develop a general encoder-decoder architecture, where a CNN is used as sentence encoder, a uni-directional LSTM as document encoder, with another uni-directional LSTM as decoder. To decrease the number of parameters while maintaining the accuracy, Nallapati et al. (2017) present SummaRuNNer, a simple RNN-based sequence classifier without decoder, outperforming or matching the model of (Cheng and Lapata, 2016). They take content, salience, novelty, and position of each sentence into consideration when deciding if a sentence should be included in the extractive summary. Yet, they do not capture any aspect of the topical structure, as we do in this paper. So their approach would arguably suffer when applied to long documents, likely containing multiple and diverse topics.

While SummaRuNNer was tested only on news, Kedzie et al. (2018) carry out a comprehensive set of experiments with deep learning models of extractive summarization across different domains, i.e. news, personal stories, meetings, and medical articles, as well as across different neural architectures, in order to better understand the general pros and cons of different design choices. They find that non auto-regressive sentence extraction performs as well or better than auto-regressive extraction in all domains, where by auto-regressive sentence extraction they mean using previous predictions to inform future predictions. Furthermore, they find that the Average Word Embedding sentence encoder works at least as well as encoders based on CNN and RNN. In light of these findings, our model is not auto-regressive and uses the Average Word Embedding encoder.

### 2.2 Extractive summarization on Scientific papers

Research on summarizing scientific articles has a long history (Nenkova et al., 2011). Earlier on, it was realized that summarizing scientific papers requires different approaches than what was used for summarizing news articles, due to differences in document length, writing style and rhetorical structure. For instance, (Teufel and Moens, 2002) presented a supervised Naive Bayes

[1]The data and code are available at https://github.com/Wendy-Xiao/Extsumm_local_global_context.

| Datasets | # docs | avg. doc. length | avg. summ. length |
|---|---|---|---|
| CNN | 92K | 656 | 43 |
| Daily Mail | 219K | 693 | 52 |
| NY Times | 655K | 530 | 38 |
| PubMed | 133K | 3016 | 203 |
| arXiv | 215K | 4938 | 220 |

Table 1: Comparison of news datasets and scientific paper datasets(Cohan et al., 2018), the length is in terms of the number of words

classifier to select content from a scientific paper based on the rhetorical status of each sentence (e.g., whether it specified a research goal, or some generally accepted scientific background knowledge, etc.). More recently, researchers have extended this work by applying more sophisticated classifiers to identify more fine-grain rhetorical categories, as well as by exploiting citation contexts. Liakata et al. (2013) propose the CoreSC discourse-driven content, which relies on CRFs and SVMs, to classify the discourse categories (e.g. Background, Hypothesis, Motivation, etc.) at the sentence level. The recent work most similar to ours is (Collins et al., 2017) where, in order to determine whether a sentence should be included in the summary, they directly use the section each sentence appears in as a categorical feature with values like Highlight, Abstract, Introduction, etc.. In this paper, instead of using sections as categorical features, we rely on a distributed representation of the semantic information within each section, as the local context of each sentence. In a very different line of work, Cohan and Goharian (2015) form the summary by also exploiting information on how the target paper is cited in other papers. Currently, we do not use any information from citation contexts.

### 2.3 Datasets for long documents

Dernoncourt et al. (2018) provide a comprehensive overview of the current datasets for summarization. Noticeably, most of the larger-scale summarization datasets consists of relatively short documents, like CNN/DailyMail (Nallapati et al., 2016) and New York Times (Sandhaus, 2008). One exception is (Cohan et al., 2018) that recently introduce two large-scale datasets of long and structured scientific papers obtained from arXiv and PubMed. These two new datasets contain much longer documents than all the news datasets (See Table 1) and are therefore ideal test-beds for the method we present in this paper.

### 2.4 Neural Abstractive summarization on long documents

While most current neural abstractive summarization models have focused on summarizing relatively short news articles (e.g., (See et al., 2017)), few researchers have started to investigate the summarization of longer documents by exploiting their natural structure. Celikyilmaz et al. (2018) present an encoder-decoder architecture to address the challenges of representing a long document for abstractive summarization. The encoding task is divided across several collaborating agents, each is responsible for a subsection of text through a multi-layer LSTM with word attention. Their model seems however overly complicated when it comes to the extractive summarization task, where word attention is arguably much less critical. So, we do not consider this model further in this paper.

Cohan et al. (2018) also propose a model for abstractive summarization taking the structure of documents into consideration with a hierarchical approach, and test it on longer documents with section information, i.e. scientific papers. In particular, they apply a hierarchical encoder at the word and section levels. Then, in the decoding step, they combine the word attention and section attention to obtain a context vector.

This approach to capture discourse structure is however quite limited both in general and especially when you consider its application to extractive summarization. First, their hierarchical method has a large number of parameters and it is therefore slow to train and likely prone to overfitting[2]. Secondly, it does not take the global context of the whole document into account, which may arguably be critical in extractive methods, when deciding on the salience of a sentence (or even a word). The extractive summarizer we present in this paper tries to address these limitations by adopting the parameter lean LSTM-minus method, and by explicitly modeling the global context.

### 2.5 LSTM-Minus

The LSTM-Minus method is first proposed in (Wang and Chang, 2016) as a novel way to learn sentence segment embeddings for graph-based dependency parsing, i.e. estimating the most likely

[2]To address this, they only process the first 2000 words of each document, by setting a hard threshold in their implementation, and therefore loosing information.

dependency tree given an input sentence. For each dependency pair, they divide a sentence into three segments (prefix, infix and suffix), and LSTM-Minus is used to represent each segment. They apply a single LSTM to the whole sentence and use the difference between two hidden states $h_j - h_i$ to represent the segment from word $w_i$ to word $w_j$. This enables their model to learn segment embeddings from information both outside and inside the segments and thus enhancing their model ability to access to sentence-level information. The intuition behind the method is that each hidden vector $h_t$ can capture useful information before and including the word $v_t$.

Shortly after, Cross and Huang (2016) use the same method on the task of constituency parsing, as the representation of a sentence span, extending the original uni-directional LSTM-Minus to the bi-directional case. More recently, inspired by the success of LSTM-Minus in both dependency and constituency parsing, Liu and Lapata (2017) extend the technique to discourse parsing. They propose a two-stage model consisting of an intra-sentential parser and a multi-sentential parser, learning contextually informed representations of constituents with LSTM-Minus, at the sentence and document level, respectively.

Similarly, in this paper, when deciding if a sentence should be included in the summary, the local context of that sentence is captured by applying LSTM-Minus at the document level, to represent the sub-sequence of sentences of the document (i.e., the topic/section) the target sentence belongs to.

## 3 Our Model

In this work, we propose an extractive model for long documents, incorporating local and global context information, motivated by natural topic-oriented structure of human-written long documents. The architecture of our model is shown in Figure 1, each sentence is visited sequentially in the original document order, and a corresponding confidence score is computed expressing whether the sentence should be included in the extractive summary. Our model comprises three components: the sentence encoder, the document encoder and the sentence classifier.

### 3.1 Sentence Encoder

The goal of the sentence encoder is mapping sequences of word embeddings to a fixed length vector (See bottom center of Figure 1). There are several common methods to embed sentences. For extractive summarization, RNN were used in (Nallapati et al., 2017), CNN in (Cheng and Lapata, 2016), and Average Word Embedding in (Kedzie et al., 2018). Kedzie et al. (2018) experiment with all the three methods and conclude that Word Embedding Averaging is as good or better than either RNNs or CNNs for sentence embedding across different domains and summarizer architectures. Thus, we use the Average Word Embedding as our sentence encoder, by which a sentence embedding is simply the average of its word embeddings, i.e.

$$se = \frac{1}{n}\sum_{w_0}^{w_n} emb(w_i), se \in \mathbb{R}^{d_{emb}}.$$

Besides, we also tried the popular pre-trained BERT sentence embedding (Devlin et al., 2019), but initial results were rather poor. So we do not pursue this possibility any further.

### 3.2 Document Encoder

At the document level, a bi-directional recurrent neural network (Schuster and Paliwal, 1997) is often used to encode all the sentences sequentially forward and backward, with such model achieving remarkable success in machine translation (Bahdanau et al., 2015). As units, we selected gated recurrent units (GRU) (Cho et al., 2014), in light of favorable results shown in (Chung et al., 2014). The GRU is represented with the standard reset, update, and new gates.

The output of the bi-directional GRU for each sentence $t$ comprises two hidden states, $h_t^f \in \mathbb{R}^{d_{hid}}, h_t^b \in \mathbb{R}^{d_{hid}}$ as forward and backward hidden state, respectively.

**A. Sentence representation** As shown in Figure 1(A), for each sentence $t$, the sentence representation is the concatenation of both backward and forward hidden state of that sentence.

$$sr_t = (h_t^f : h_t^b), sr_t \in \mathbb{R}^{d_{hid}*2}$$

In this way, the sentence representation not only represents the current sentence, but also partially covers contextual information both before and after this sentence.

**B. Document representation** The document representation provides global information on the

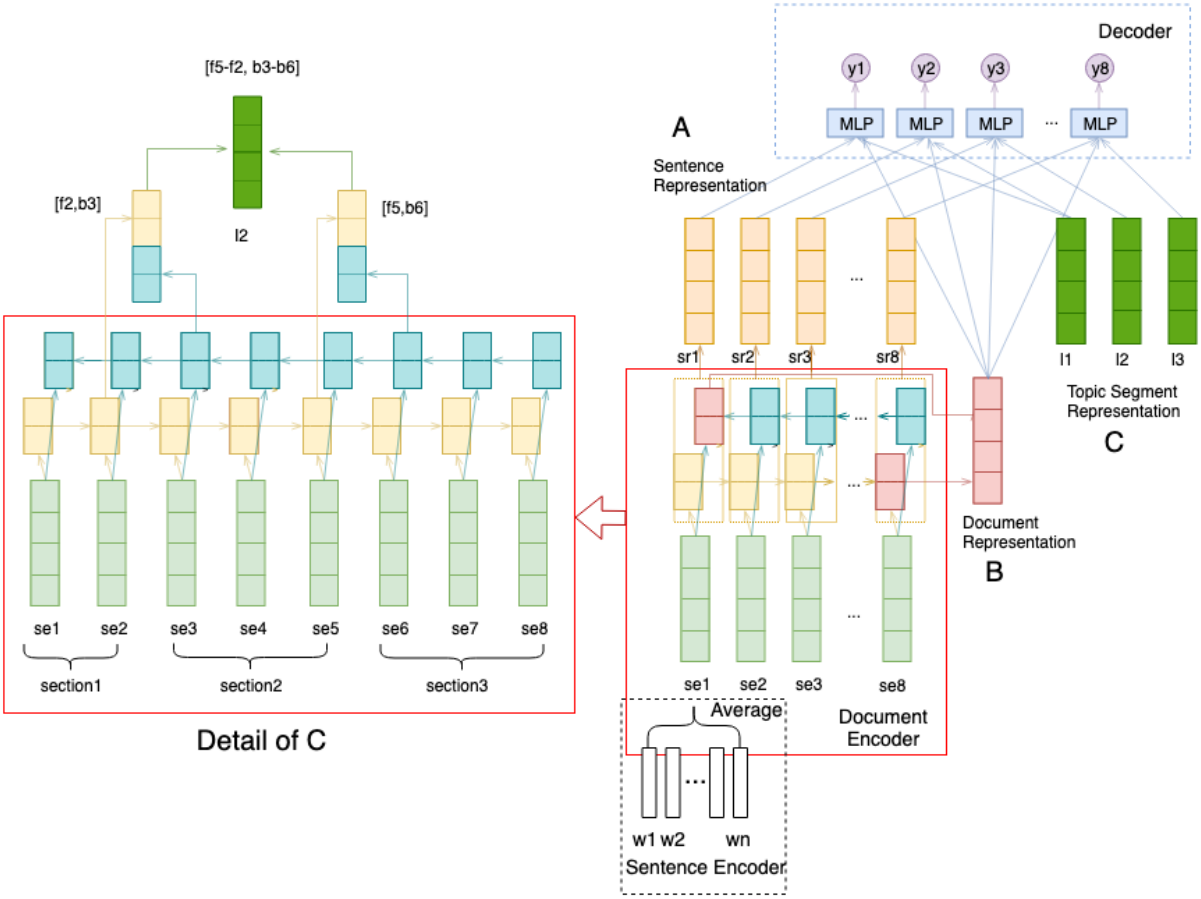


Figure 1: The structure of our model, $se_i, sr_i$ represent the sentence embedding and sentence representation of sentence $i$, respectively. The binary decision of whether the sentence should be included in the summary is based on the sentence itself (A), the whole document (B) and the current topic (C). The document representation is simply the concatenation of the last hidden states of the forward and backward RNNs, while the topic segment representation is computed by applying LSTM-Minus, as shown in detail in the left panel (Detail of C).

whole document. It is computed as the concatenation of the final state of the forward and backward GRU, labeled as B in Figure 1. (Li et al., 2018)

$$d = (h_n^f : h_0^b), d \in \mathbb{R}^{d_{hid}*2}$$

**C. Topic segment representation** To capture the local context of each sentence, namely the information of the topic segment that sentence falls into, we apply the LSTM-Minus method[3], a method for learning embeddings of text spans. LSTM-Minus is shown in detail in Figure 1 (left panel C), each topic segment is represented as the subtraction between the hidden states of the start and the end of that topic. As illustrated in Figure 1, the representation for section 2 of the sample document (containing three sections and eight sentences overall) can be computed as $[f_5 - f_2, b_3 - b_6]$, where $f_5, f_2$ are the forward hidden states of sentence 5 and 2, respectively, while $b_3, b_6$ are the backward hidden states of sentence 3 and 6, respectively. In general, the topic segment representation $l_t$ for segment $t$ is computed as:

$$\begin{aligned} f_t &= h^f_{end_t} - h^f_{start_t - 1}, f_t \in \mathbb{R}^{d_{hid}} \\ b_t &= h^b_{start_t} - h^b_{end_t + 1}, b_t \in \mathbb{R}^{d_{hid}} \\ l_t &= (f_t : b_t), l_t \in \mathbb{R}^{d_{hid}*2} \end{aligned}$$

where $start_t, end_t$ is the index of the beginning and the end of topic $t$, $f_t$ and $b_t$ denote the topic segment representation of forward and backward, respectively. The final representation of topic $t$ is the concatenation of forward and backward representation $l_t$. To obtain $f_i$ and $b_i$, we utilize subtraction between GRU hidden vectors of $start_t$ and $end_t$, and we pad the hidden states with zero vectors both in the beginning and the end, to ensure the index can not be out of bound. The intuition

[3] In the original paper, LSTMs were used as recurrent unit. Although we use GRUs here, for consistency with previous work, we still call the method LSTM-Minus

behind this process is that the GRUs can keep previous useful information in their memory cell by exploiting reset, update, and new gates to decide how to utilize and update the memory of previous information. In this way, we can represent the contextual information within each topic segment for all the sentences in that segment.

### 3.3 Decoder

Once we have obtained a representation for the sentence, for its topic segment (i.e., local context) and for the document (i.e., global context), these three factors are combined to make a final prediction $p_i$ on whether the sentence should be included in the summary. We consider two ways in which these three factors can be combined.

**Concatenation** We can simply concatenate the vectors of these three factors as,

$$input_i = (d : l_t : sr_i), input_i \in \mathbb{R}^{d_{hid}*6}$$

where sentence $i$ is part of the topic $t$, and $input_i$ is the representation of sentence $i$ with topic segment information and global context information.

**Attentive context** As local context and global context are all contextual information of the given sentence, we use an attention mechanism to decide the weight of each context vector, represented as

$$\begin{aligned} score_i^d &= v^T tanh(W_a(d : sr_i)) \\ score_i^l &= v^T tanh(W_a(l_t : sr_i)) \\ weight_i^d &= \frac{score_i^d}{score_i^d + score_i^l} \\ weight_i^l &= \frac{score_i^l}{score_i^d + score_i^l} \\ context_i &= weight_i^d * d + weight_i^l * l_t \\ input_i &= (sr_i : context_i), input_i \in \mathbb{R}^{d_{hid}*4} \end{aligned}$$

where the $context_i$ is the weighted context vector of each sentence $i$, and assume sentence $i$ is in topic $t$.

Then there is a final multi-layer perceptron(MLP) followed with a sigmoid activation function indicating the confidence score for selecting each sentence:

$$\begin{aligned} h_i &= Dropout(ReLU(W_{mlp} input_i + b_{mlp})) \\ p_i &= \sigma(W_h h_i + b_h) \end{aligned}$$

## 4 Experiments

To validate our method, we set up experiments on the two scientific paper datasets (arXiv and PubMed). With ROUGE and METEOR scores as automatic evaluation metrics, we compare with previous works, both abstractive and extractive.

### 4.1 Training

The weighted negative log-likelihood is minimized, where the weight is computed as $w_{pos} = \frac{\#negative}{\#postive}$, to solve the problem of highly imbalanced data (typical in extractive summarization).

$$\begin{aligned} \mathcal{L} &= -\sum_{d=1}^{N}\sum_{i=1}^{N_d}(w_{pos} * y_i \log p(y_i|\mathbf{W}, b) \\ &+ (1 - y_i) \log p(y_i|\mathbf{W}, b)) \end{aligned}$$

where $y_i$ represent the ground-truth label of sentence $i$, with $y_i = 1$ meaning sentence $i$ is in the gold-standard extract summary.

### 4.2 Extractive Label Generation

In the Pubmed and arXiv datasets, the extractive summaries are missing. So we follow the work of (Kedzie et al., 2018) on extractive summary labeling, constructing gold label sequences by greedily optimizing ROUGE-1 on the gold-standard abstracts, which are available for each article. [4] The algorithm is shown in Appendix A.

### 4.3 Implementation Details

We train our model using the Adam optimizer (Kingma and Ba, 2015) with learning rate 0.0001 and a drop out rate of 0.3. We use a mini-batch with a batch size of 32 documents, and the size of the GRU hidden states is 300. For word embeddings, we use GloVe (Pennington et al., 2014) with dimension 300, pre-trained on the Wikipedia and Gigaword. The vocabulary size of our model is 50000. All the above parameters were set based on (Kedzie et al., 2018) without any fine-tuning. Again following (Kedzie et al., 2018), we train each model for 50 epochs, and the best model is selected with early stopping on the validation set according to Rouge-2 F-score.

### 4.4 Models for Comparison

We perform a systematic comparison with previous work in extractive summarization. For completeness, we also compare with recent neural abstractive approaches. In all the experiments, we use the same train/val/test splitting.

[4] For this, we use a popular python implementation of the ROUGE score to build the oracle. Code can be found here, https://pypi.org/project/py-rouge/

| *Model* | *ROUGE-1* | *ROUGE-2* | *ROUGE-L* | *METEOR* |
|---|---|---|---|---|
| SumBasic* | 29.47 | 6.95 | 26.30 | - |
| LSA* | 29.91 | 7.42 | 25.67 | - |
| LexRank* | 33.85 | 10.73 | 28.99 | - |
| Attn-Seq2Seq* | 29.30 | 6.00 | 25.56 | - |
| Pntr-Gen-Seq2Seq* | 32.06 | 9.04 | 25.16 | - |
| Discourse-aware* | 35.80 | 11.05 | **31.80** | - |
| Baseline | 42.91 | 16.65 | 28.53 | 21.35 |
| Cheng & Lapata | 42.24 | 15.97 | 27.88 | 20.97 |
| SummaRuNNer | 42.81 | 16.52 | 28.23 | 21.35 |
| Ours-attentive context | **43.58** | **17.37** | **29.30** | **21.71** |
| Ours-concat | **43.62** | **17.36** | **29.14** | **21.78** |
| Lead | 33.66 | 8.94 | 22.19 | 16.45 |
| Oracle | 53.88 | 23.05 | 34.90 | 24.11 |

Table 2: Results on the arXiv dataset. For models with an $*$, we report results from (Cohan et al., 2018). Models are traditional extractive in the first block, neural abstractive in the second block, while neural extractive in the third block. The Oracle (last row) corresponds to using the ground truth labels, obtained (for training) by the greedy algorithm, see Section 4.2. Results that are not significantly distinguished from the best systems are bold.

| *Model* | *ROUGE-1* | *ROUGE-2* | *ROUGE-L* | *METEOR* |
|---|---|---|---|---|
| SumBasic* | 37.15 | 11.36 | 33.43 | - |
| LSA* | 33.89 | 9.93 | 29.70 | - |
| LexRank* | 39.19 | 13.89 | 34.59 | - |
| Attn-Seq2Seq* | 31.55 | 8.52 | 27.38 | - |
| Pntr-Gen-Seq2Seq* | 35.86 | 10.22 | 29.69 | - |
| Discourse-aware* | 38.93 | 15.37 | **35.21** | - |
| Baseline | 44.29 | 19.17 | 30.89 | 20.56 |
| Cheng & Lapata | 43.89 | 18.53 | 30.17 | 20.34 |
| SummaRuNNer | 43.89 | 18.78 | 30.36 | 20.42 |
| Ours-attentive context | **44.81** | **19.74** | **31.48** | **20.83** |
| Ours-concat | **44.85** | **19.70** | **31.43** | **20.83** |
| Lead | 35.63 | 12.28 | 25.17 | 16.19 |
| Oracle | 55.05 | 27.48 | 38.66 | 23.60 |

Table 3: Results on the Pubmed dataset. See caption of Table 2 above for details on compared models and notation.

- Traditional extractive summarization models: SumBasic (Vanderwende et al., 2007), LSA (Steinberger and Jezek, 2004), and LexRank (Erkan and Radev, 2004)
- Neural abstractive summarization models: Attn-Seq2Seq (Nallapati et al., 2016), Pntr-Gen-Seq2Seq (See et al., 2017) and Discourse-aware (Cohan et al., 2018)
- Neural extractive summarization models: Cheng&Lapata (Cheng and Lapata, 2016) and SummaRuNNer (Nallapati et al., 2017). Based on (Kedzie et al., 2018), we use the Average Word Encoder as sentence encoder for both models, instead of the CNN and RNN sentence encoders that were originally used in the two systems, respectively. [5]
- Baseline: Similar to our model, but without local context and global context, i.e. the input to MLP is the sentence representation only.
- Lead: Given a length limit of $k$ words for the summary, Lead will return the first $k$ words of the source document.
- Oracle: uses the Gold Standard extractive labels, generated based on ROUGE (Sec. 4.2).

### 4.5 Results and Analysis

For evaluation, we follow the same procedure as in (Kedzie et al., 2018). Summaries are generated by selecting the top ranked sentences by model probability $p(y_i|\mathbb{W}, b)$, until the length limit is met or exceeded. Based on the average length of abstracts in these two datasets, we set the length limit to 200 words. We use ROUGE scores[6] (Lin and Hovy, 2003) and METEOR scores[7] (Denkowski and Lavie, 2014) between the model results and ground-truth abstractive summaries as evaluation metric. The unigram and bigram overlap (ROUGE-1,2) are intended to measure the informativeness, while longest common subsequence (ROUGE-L) captures fluency to some extent (Cheng and Lapata, 2016). METEOR was originally proposed to evaluate translation systems by measuring the alignment between the system output and reference translations. As such, it can also be used as an automatic evaluation metric for summarization (Kedzie et al., 2018).

The performance of all models on arXiv and Pubmed is shown in Table 2 and Table 3, respectively. Follow the work (Kedzie et al., 2018), we use the approximate randomization as the statistical significance test method (Riezler and Maxwell, 2005) with a Bonferroni correction for multiple comparisons, at the confidence level 0.01 ($p < 0.01$). As we can see in these tables, on both datasets, the neural extractive models outperforms the traditional extractive models on informativeness (ROUGE-1,2) by a wide margin, but results are mixed on ROUGE-L. Presumably, this is due to the neural training process, which relies on a goal standard based on ROUGE-1. Exploring other training schemes and/or a combination of

[5] Aiming for a fair and reproducible comparison, we re-implemented the models by borrowing the extractor classes from (Kedzie et al., 2018), the source code can be found https://github.com/kedz/nnsum/tree/emnlp18-release

[6] We use a modified version of rouge_papier, a python wrapper of ROUGE-1.5.5, https://github.com/kedz/rouge_papier. The command line is 'Perl ROUGE-1.5.5 -e data -a -n 2 -r 1000 -f A -z SPL config_file'

[7] We use default setting of METEOR.

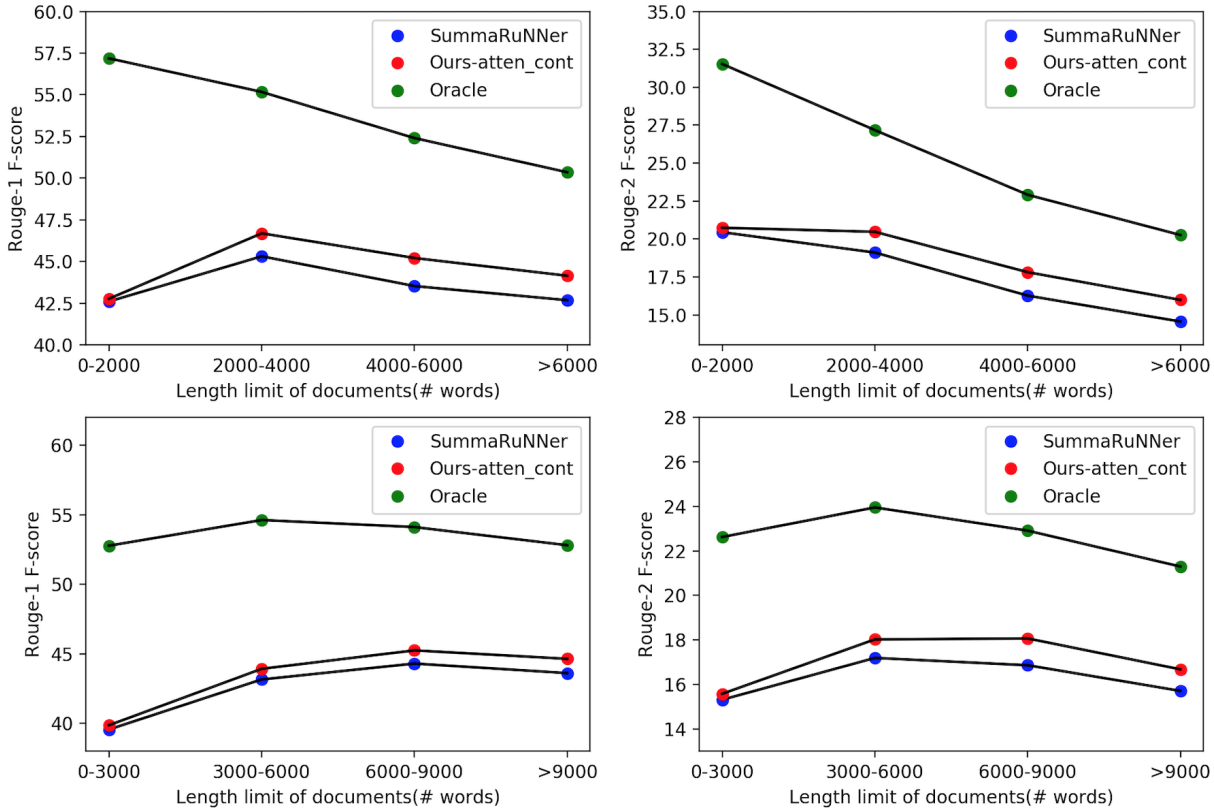


Figure 2: A Comparison between our model, SummaRuNNer and Oracle when applied to documents with increasing length, left-up: ROUGE-1 on Pubmed dataset, right-up: ROUGE-2 on Pubmed dataset, left-down: ROUGE-1 on arXiv dataset, right-down: ROUGE-2 on arXiv dataset

traditional and neural approaches is left as future work. Similarly, the neural extractive models also dominate the neural abstractive models on ROUGE-1,2, but these abstractive models tend to have the highest ROUGE-L scores, possibly because they are trained directly on gold standard abstract summaries.

Compared with other neural extractive models, our models (both with attentive context and concatenation decoder) have better performances on all three ROUGE scores, as well as METEOR. In particular, the improvements over the Baseline model show that a combination of local and global contextual information does help to identify the most important sentences (more on this in the next section). Interestingly, just the Baseline model already achieves a slightly better performance than previous works; possibly because the auto-regressive approach used in those models is even more detrimental for long documents.

Figure 2 shows the most important result of our analysis: the benefits of our method, explicitly designed to deal with longer documents, do actually become stronger as we apply it to longer documents. As it can be seen in Figure 2, the performance gain of our model with respect to current state-of-the-art extractive summarizer is more pronounced for documents with $>= 3000$ words in both datasets.

Finally, the result of Lead (Table 2, 3) shows that scientific papers have less position bias than news; i.e., the first sentences of these papers are not a good choice to form an extractive summary.

As a teaser for the potential and challenges that still face our approach, its output (i.e., the extracted sentences) *when applied to this paper* is colored in red and the order in which the sentences are extracted is marked with the Roman numbering. If we set the summary length limit to the length of our abstract, the first five sentences in the conclusions section are extracted. If we increase the length to 200 words, two more sentences are extracted, which do seem to provide useful complementary information. Not surprisingly, some redundancy is present, as dealing explicitly with redundancy is not a goal of our current proposal and left as future work.

### 4.6 Ablation Study

| *Model* | *ROUGE-1(+l/+g)* | *ROUGE-2(+l/+g)* | *ROUGE-L(+l/+g)* |
|---|---|---|---|
| BSL | 44.29 (na/na) | 19.17 (na/na) | 30.89 (na/na) |
| BSL+l | **44.85** (+.56/na) | **19.77** (+.6/na) | **31.51** (+.62/na) |
| BSL+g | 44.06 (na/-.23) | 18.83 (na/-.34) | 30.53 (na/-.36) |
| BSL+l+g | **44.81** (+.75/-.04) | **19.74** (+.91/-.03) | **31.48** (+.95/-.03) |
| BSL | 43.85 (na/na) | 15.94 (na/na) | 28.13 (na/na) |
| BSL+l | **44.65** (+.8/na) | **16.75** (+.81/na) | **28.99** (+.85/na) |
| BSL+g | 43.70 (na/-.15) | 15.74 (na/-.2) | 27.67 (na/-.46) |
| BSL+l+g | **44.64** (+.94/-.01) | **16.69** (+.95/-.06) | **28.96** (+1.29/-.03) |

Table 4: Ablation study on the Pubmed dataset, with all the documents(up) and a subset of long documents (down, $>$ 6000 words). BSL is the model with sentence representation only, BSL+l is the model with sentence and local topic information, BSL+g is the model with sentence and global document information, and the last one is the full model with attentive_context decoder. The numbers in parenthesis represent the improvements with the additional local/global context, respectively. Results that are not significantly distinguished from the best systems are bold.

| *Model* | *ROUGE-1(+l/+g)* | *ROUGE-2(+l/+g)* | *ROUGE-L(+l/+g)* |
|---|---|---|---|
| BSL | 42.91 (na/na) | 16.65 (na/na) | 28.53 (na/na) |
| BSL+l | **43.57** (+.66/na) | **17.35** (+.7/na) | **29.29** (+.76/na) |
| BSL+g | 42.90 (na/-.01) | 16.58 (na/-.07) | 28.36 (na/-.17) |
| BSL+l+g | **43.58** (+.68/+.01) | **17.37** (+.79/+.02) | **29.30** (+.94/+.01) |
| BSL | 42.95 (na/na) | 14.85 (na/na) | 28.66 (na/na) |
| BSL+l | **44.01** (+1.06/na) | **15.95** (+1.1/na) | **29.68** (+1.02/na) |
| BSL+g | 43.05 (na/+.1) | 14.91 (na/+.06) | 28.57 (na/-.09) |
| BSL+l+g | **44.17** (+1.12/+.16) | **16.01** (+1.1/+.06) | **29.72** (+1.15/+.04) |

Table 5: Ablation study on arXiv dataset, with all documents (up) and a subset of long document(down, $>$ 9000 words). Results that are not significantly different from the best systems are in bold.

In order to assess the relative contributions of the global and local models to the performance of our approach, we ran an ablation study. This was done for each dataset both with the whole test set, as well as with a subset of long documents. The results for Pubmed and arXiv are shown in Table 4 and Table 5, respectively. For statistical significance, as it was done for the general results in Section 4.5, we use the approximate randomization method (Riezler and Maxwell, 2005) with the Bonferroni correction at ($p < 0.01$).

From these tables, we can see that on both datasets the performance significantly improves when local topic information (i.e. local context) is added. And the improvement is even greater when we only consider long documents. Rather surprisingly, this is not the case for the global context. Adding a representation of the whole document (i.e. global context) never significantly improves performance. In essence, it seems that all the benefits of our model come exclusively from modeling the local context, even for the longest documents. Further investigation of this finding is left as future work.

## 5 Conclusions and Future Work

In this paper, we propose a novel extractive summarization model especially designed for long documents, by incorporating the local context within each topic, along with the global context of the whole document.[II] Our approach integrates recent findings on neural extractive summarization in a parameter lean and modular architecture.[III] We evaluate our model and compare with previous works in both extractive and abstractive summarization on two large scientific paper datasets, which contain documents that are much longer than in previously used corpora.[IV] Our model not only achieves state-of-the-art on these two datasets, but in an additional experiment, in which we consider documents with increasing length, it becomes more competitive for longer documents.[V] We also ran an ablation study to assess the relative contribution of the global and local components of our approach. [I] Rather surprisingly, it appears that the benefits of our model come only from modeling the local context.

For future work, we initially intend to investigate neural methods to deal with redundancy. Then, it could be beneficial to integrate explicit features, like sentence position and salience, into our neural approach. More generally, we plan to combine of traditional and neural models, as suggested by our results. Furthermore, we would like to explore more sophistical structure of documents, like discourse tree, instead of rough topic segments. As for evaluation, we would like to elicit human judgments, for instance by inviting authors to rate the outputs from different systems, when applied to their own papers. More long term, we will study how extractive/abstractive techniques can be integrated; for instance, the output of an extractive system could be fed into an abstractive one, training the two jointly.

## Acknowledgments

This research was supported by the Language & Speech Innovation Lab of Cloud BU, Huawei Technologies Co., Ltd.

## References


Mehdi Allahyari, Seyed Amin Pouriyeh, Mehdi Assefi, Saeid Safaei, Elizabeth D. Trippe, Juan B. Gutierrez, and Krys Kochut. 2017. Text summarization techniques: A brief survey. *CoRR*, abs/1707.02268.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep communicating agents for abstractive summarization. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1662–1675, New Orleans, Louisiana. Association for Computational Linguistics.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 484–494, Berlin, Germany. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar. Association for Computational Linguistics.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555.

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. 2018. A discourse-aware attention model for abstractive summarization of long documents. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 615–621, New Orleans, Louisiana. Association for Computational Linguistics.

Arman Cohan and Nazli Goharian. 2015. Scientific article summarization using citation-context and article's discourse structure. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 390–400, Lisbon, Portugal. Association for Computational Linguistics.

Ed Collins, Isabelle Augenstein, and Sebastian Riedel. 2017. A supervised approach to extractive summarisation of scientific papers. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 195–205, Vancouver, Canada. Association for Computational Linguistics.

James Cross and Liang Huang. 2016. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1–11, Austin, Texas. Association for Computational Linguistics.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 376–380, Baltimore, Maryland, USA. Association for Computational Linguistics.

Franck Dernoncourt, Mohammad Ghassemi, and Walter Chang. 2018. A repository of corpora for summarization. In *LREC*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Günes Erkan and Dragomir R. Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Int. Res.*, 22(1):457–479.

Chris Kedzie, Kathleen McKeown, and Hal Daume III. 2018. Content selection in deep learning models of summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1818–1828. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Wei Li, Xinyan Xiao, Yajuan Lyu, and Yuanzhuo Wang. 2018. Improving neural abstractive document summarization with explicit information selection modeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1787–1796, Brussels, Belgium. Association for Computational Linguistics.

Maria Liakata, Simon Dobnik, Shyamasree Saha, Colin Batchelor, and Dietrich Rebholz-Schuhmann. 2013. A discourse-driven content model for summarising scientific articles evaluated in a complex question answering task. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 747–757, Seattle, Washington, USA. Association for Computational Linguistics.

Chin-Yew Lin and Eduard H. Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *HLT-NAACL*.

Yang Liu and Mirella Lapata. 2017. Learning contextually informed representations for linear-time discourse parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1289–1298, Copenhagen, Denmark. Association for Computational Linguistics.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, pages 3075–3081. AAAI Press.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gulçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany. Association for Computational Linguistics.

Ani Nenkova, Sameer Maskey, and Yang Liu. 2011. Automatic summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts of ACL 2011*, HLT '11, pages 3:1–3:86, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Dragomir Radev, Timothy Allison, Sasha Blair-Goldensohn, John Blitzer, Arda Çelebi, Stanko Dimitrov, Elliott Drabek, Ali Hakim, Wai Lam, Danyu Liu, Jahna Otterbacher, Hong Qi, Horacio Saggion, Simone Teufel, Michael Topper, Adam Winkel, and Zhu Zhang. 2004. Mead - a platform for multidocument multilingual text summarization. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal. European Language Resources Association (ELRA).

Stefan Riezler and John T. Maxwell. 2005. On some pitfalls in automatic evaluation and significance testing for MT. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 57–64, Ann Arbor, Michigan. Association for Computational Linguistics.

Evan Sandhaus. 2008. *The New York Times Annotated Corpus*. https://catalog.ldc.upenn.edu/LDC2008T19.

M. Schuster and K.K. Paliwal. 1997. Bidirectional recurrent neural networks. *Trans. Sig. Proc.*, 45(11):2673–2681.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Louis Shao, Stephan Gouws, Denny Britz, Anna Goldie, Brian Strope, and Ray Kurzweil. 2017. Generating long and diverse responses with neural conversation models. *CoRR*, abs/1701.03185.

Josef Steinberger and Karel Jezek. 2004. Using latent semantic analysis in text summarization and summary evaluation.

Frederick Suppe. 1998. The structure of a scientific paper. *Philosophy of Science*, 65(3):381–405.

Simone Teufel and Marc Moens. 2002. Articles summarizing scientific articles: Experiments with relevance and rhetorical status. *Computational Linguistics*, 28(4):409–445.

Antoine Tixier, Polykarpos Meladianos, and Michalis Vazirgiannis. 2017. Combining graph degeneracy and submodularity for unsupervised extractive summarization. In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 48–58. Association for Computational Linguistics.

Lucy Vanderwende, Hisami Suzuki, Chris Brockett, and Ani Nenkova. 2007. Beyond sumbasic: Task-focused summarization with sentence simplification and lexical expansion. *Inf. Process. Manage.*, 43:1606–1618.

Wenhui Wang and Baobao Chang. 2016. Graph-based dependency parsing with bidirectional lstm. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2306–2315, Berlin, Germany. Association for Computational Linguistics.

# Enhancing Neural Data-To-Text Generation Models with External Background Knowledge

**Shuang Chen**[1]*, **Jinpeng Wang**[2], **Xiaocheng Feng**[1], **Feng Jiang**[1,3], **Bing Qin**[1], **Chin-Yew Lin**[2]
[1] Harbin Institute of Technology, Harbin, China
[2] Microsoft Research Asia [3] Peng Cheng Laboratory
hitercs@gmail.com, {jinpwa, cyl}@microsoft.com,
{xcfeng, qinb}@ir.hit.edu.cn, fjiang@hit.edu.cn

## Abstract

Recent neural models for data-to-text generation rely on massive parallel pairs of data and text to learn the writing knowledge. They often assume that writing knowledge can be acquired from the training data alone. However, when people are writing, they not only rely on the data but also consider related knowledge. In this paper, we enhance neural data-to-text models with external knowledge in a simple but effective way to improve the fidelity of generated text. Besides relying on parallel data and text as in previous work, our model attends to relevant external knowledge, encoded as a temporary memory, and combines this knowledge with the context representation of data before generating words. This allows the model to infer relevant facts which are not explicitly stated in the data table from an external knowledge source. Experimental results on twenty-one Wikipedia infobox-to-text datasets show our model, KBAtt, consistently improves a state-of-the-art model on most of the datasets. In addition, to quantify when and why external knowledge is effective, we design a metric, KBGain, which shows a strong correlation with the observed performance boost. This result demonstrates the relevance of external knowledge and sparseness of original data are the main factors affecting system performance.

## 1 Introduction

Automatic text generation from structured data (data-to-text) is a classic task in natural language generation which aims to automatically generate fluent, truthful and informative texts based on structured data (Kukich, 1983; Holmes-Higgin, 1994; Reiter and Dale, 1997). Data-to-text is often formulated into two subproblems: *content selection* which decides what contents should be included in the text and *surface realization* which determines how to generate the text based on selected contents. Traditionally, these two subproblems have been tackled separately. In recent years, neural generation models, especially the encoder-decoder model, solve these two subproblems jointly and have achieved remarkable successes in several benchmarks (Mei et al., 2016; Lebret et al., 2016; Wiseman et al., 2017; Dušek et al., 2018; Nie et al., 2018).

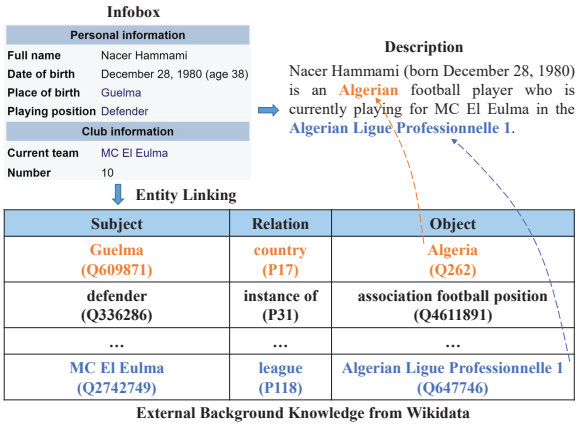


Figure 1: An example of generating description from a Wikipedia infobox. External background knowledge expanded from the infobox is helpful for generation.

Such end-to-end data-to-text models rely on massive parallel pairs of data and text to learn the writing knowledge. They often assume that all writing knowledge can be learned from the training data. However, when people are writing, they will not only rely on the data contents themselves but also consider related knowledge, which is neglected by previous methods. For example, as shown in Fig. 1, an infobox about a person called *Nacer Hammami* is paired with its corresponding biography description from the Wikipedia. However, the information in the infobox is not enough to cover all the facts mentioned in the description. To generate this description from the in-

*Contribution during internship at Microsoft Research.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3022–3032, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

fobox, we need to expand information based on external background knowledge from its related entities. For example, in the description: 1) "is an *Algerian* football player" indicates the nationality of *Nacer Hammami* which is not explicitly stated in the infobox. However, the place of birth, *Guelma*, of *Nacer Hammami* is given, therefore the nationality can be inferred from the knowledge that *Guelma* is a place in *Algeria*. 2) "playing for *MC El Eulma* in the *Algerian Ligue Professionnelle 1*" describes the fact that *MC El Eulma* is a club in the *Algerian Ligue Professionnelle 1* which is also not explicitly stated in the infobox which can be expanded from external knowledge base.

One may argue that neural models can learn such knowledge when enough parallel co-occurrence pairs such as (*Guelma*, *Algerian*) and (*MC El Eulma*, *Algerian Ligue Professionnelle 1*) are available. However even in such case, neural models still tend to make mistakes for sparse co-occurrence pairs as we will show in the experiments section.

In this paper we enhance neural-network-based data-to-text generation models with external knowledge in a simple but effective way. Besides learning the association between data and text from parallel data-text pairs as in previous work, our model attends to relevant external knowledge, encoded as a temporary memory, and combines this knowledge with the context representation of data before generating words. Specifically, both infobox and background knowledge facts are encoded and a dual-attention mechanism is proposed to guide the decoder to generate text.

To verify the effectiveness of our proposed model, **K**nowledge **B**ase enhanced **Att**ention-based sequence-to-sequence network (KBAtt), we conduct experiments on multiple Wikipedia infobox-to-text datasets including WikiBio (Lebret et al., 2016) and 20 new datasets[1]. Our experiment results show that KBAtt consistently improves a state-of-the-art neural data-to-text model to achieve higher performances on most of the datasets. To quantify when and why external knowledge is effective, we design a metric which shows a strong correlation with the observed performance boost. This result demonstrates the relevance of external knowledge and sparseness of original data are the main factors affecting system performance.

The contributions of our work can be summarized as follows:

- We demonstrate that external knowledge base could be used to enhance the performance of neural data-to-text models.
- We propose a simple yet effective model, KBAtt, to integrate external knowledge using a dual-attention mechanism.
- We design a metric, KBGain, to quantify when and why external knowledge is effective.
- We contribute twenty infobox-to-text datasets from a variety of domains.

## 2 The Proposed Model

Our model takes a data table $D$ (e.g., a Wikipedia infobox) and a relevant external knowledge base (KB) containing a set of facts $F$ as input and generates a natural language text $y = y_1, ..., y_T$ consisting of $T$ words. To augment the infobox with external knowledge, we preserve the Wikipedia internal hyperlink information in the field values of infobox, and track these hyperlinks to get their corresponding entities from Wikidata[2] where we retrieve only one-hop facts. The backbone of our model is an attention based sequence-to-sequence model (Bahdanau et al., 2014) equipped with copy mechanism (See et al., 2017). As shown in Fig. 2, the model consists of four main components: a table encoder, a KB encoder, the dual attention mechanism and a decoder. We describe each component in the following sections.

### 2.1 Table Encoder

In Fig. 2, the input data table $D$ consists of several *field name* and *field value* pairs. We follow (Sha et al., 2017; Liu et al., 2017) to tokenize the field values and transform the input table into a flattened sequence $\{(n_i, v_i)\}_{i=1}^{N}$, where each element is a token $v_i$ from a field value paired with its corresponding field name $n_i$. To encode the flattened table, we map each $(n_i, v_i)$ to vector $\boldsymbol{x}_i = [\boldsymbol{e}^{n_i}; \boldsymbol{e}^{v_i}]$, where $\boldsymbol{e}^{n_i}$ and $\boldsymbol{e}^{v_i}$ are trainable

[1]Available at https://github.com/hitercs/WikiInfo2Text

[2]We adopt Wikidata (dumps version: 20150831) as the external knowledge base. Although we extend the infobox with external knowledge by using the Wikipedia hyperlink, we can also apply entity linking to link input data to the knowledge base in practice.

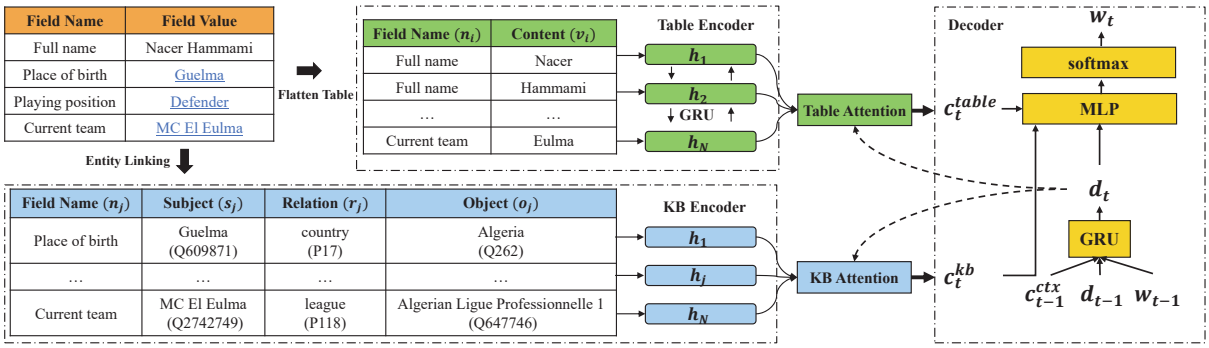


Figure 2: A diagram of the knowledge base enhanced neural data-to-text generation model. First, we transform the table into a flattened sequence, extract entities mentioned in the field value of the infobox and link them to Wikidata where we can retrieve relevant facts. Then, the table contents and external knowledge base facts are carefully encoded. Finally, a single layer GRU decoder with a dual attention mechanism decides which part of information should be used for generation.

word embeddings of $n_i$ and $v_i$, and $[.;.]$ is the concatenation of vectors. Then each $\boldsymbol{x}_i$ is encoded into a hidden vector $\boldsymbol{h}_i$ using a bi-directional GRU (Cho et al., 2014).

### 2.2 Knowledge Base Encoder

As shown in Fig. 2, we extract entities mentioned in the field value of infobox and link them to Wikidata. Then we can retrieve relevant facts whose subject is the linked entity from Wikidata. These facts contain important background knowledge related to the infobox which is helpful for generation. The KB fact set is denoted by $F = \{(n_j, s_j, r_j, o_j)\}_{j=1}^{|F|}$, where $s_j$, $r_j$, $o_j$ is the subject, relation and object of the fact respectively, and field name $n_j$ indicates the current fact is linked by the field value of $n_j$. In order to integrate such KB facts into the neural model, we apply Multi-Layer Perceptron (MLP) to encode each fact into its representation:

$$\boldsymbol{f}_j = \tanh(\mathbf{W}_f[\boldsymbol{e}^{n_j}; \boldsymbol{e}^{s_j}; \boldsymbol{e}^{r_j}; \boldsymbol{e}^{o_j}] + \boldsymbol{b}_f) \quad (1)$$

where $\mathbf{W}_f$ and $\boldsymbol{b}_f$ are trainable weights and bias, while $\boldsymbol{e}^{n_j}$, $\boldsymbol{e}^{s_j}$, $\boldsymbol{e}^{r_j}$ and $\boldsymbol{e}^{o_j}$ is the embedding of field name $n_j$, subject $s_j$, relation $r_j$ and object $o_j$ respectively. To accommodate generation steps where no information from the external knowledge is needed, such as generating *name* field which is already stated in the table, we apply a simple strategy by padding a *none* fact in the knowledge base.

### 2.3 Dual Attention Mechanism

After encoding the table and background knowledge base facts, we apply a RNN-based decoder to generate words conditioned on both table information and background knowledge fact information. In general, given a decoder hidden state $\boldsymbol{d}_t$ at timestep $t$, we apply dual attention mechanism including table attention and KB attention to determine which parts that it should pay attention to. Next we will introduce table attention and KB attention briefly.

#### 2.3.1 Table Attention

The table attention is based on similarity between decoder hidden state $\boldsymbol{d}_t$ and table contents representation $\{\boldsymbol{h}_i\}_{i=1}^{N}$. Specially we apply attention mechanism from (Bahdanau et al., 2014) to calculate the table context representation $\boldsymbol{c}_t^{table}$.

$$e_{t,i} = \mathbf{v}_a^\top \tanh(\mathbf{W}_a \boldsymbol{d}_t + \mathbf{U}_a \boldsymbol{h}_i) \quad (2)$$

$$\alpha_{t,i} = \frac{\exp(e_{t,i})}{\sum_{i=1}^{N} \exp(e_{t,i})} \quad (3)$$

$$\boldsymbol{c}_t^{table} = \sum_{i=1}^{N} \alpha_{t,i} \boldsymbol{h}_i \quad (4)$$

where $\mathbf{W}_a$, $\mathbf{U}_a$ and $\mathbf{v}_a$ are trainable parameters. $\alpha_{t,i}$ is the table attention weight.

#### 2.3.2 KB Attention

Besides utilizing table information, the words generated may contain facts which are not directly mentioned in the table but could be inferred from the background knowledge $F$. In order to integrate such knowledge into the decoder, we apply KB attention over $\{\boldsymbol{f}_j\}_{j=1}^{|F|}$. Similar to the table attention, we can get KB context representation $\boldsymbol{c}_t^{kb}$.

### 2.4 Decoder

The decoder is a single layer GRU equipped with copy mechanism. As for the generation mode, given a decoder hidden state $\boldsymbol{d}_t$, the decoder attends both table and knowledge base using mechanism described above, and get table context representation $\boldsymbol{c}_t^{table}$ and KB context representation $\boldsymbol{c}_t^{kb}$. So the context representation at time step $t$ is given by:

$$\boldsymbol{c}_t = [\boldsymbol{c}_t^{table}; \boldsymbol{c}_t^{kb}] \quad (5)$$

Then given table $D$ and knowledge base facts set $F$, the probability of word $y_t$ generated from a fixed vocabulary at time step $t$ is defined as follows:

$$P_{\text{vocab}} = \text{softmax}(f(\boldsymbol{d}_t, \boldsymbol{y}_{t-1}, \boldsymbol{c}_t)) \quad (6)$$

where $f(\cdot)$ is a non-linear function applying on the decoder hidden state $\boldsymbol{d}_t$, previous word embedding $\boldsymbol{y}_{t-1}$ and the current context vector $\boldsymbol{c}_t$.

To tackle the rare and unknown words problem, we adopt the copy mechanism (See et al., 2017). Specifically, a gate $p_{\text{gen}} \in [0, 1]$ is introduced to switch between copy mode and generation mode. The generation probability $p_{\text{gen}}$ is defined as:

$$p_{\text{gen}} = \sigma(\mathbf{w}_c^\top \boldsymbol{c}_t^{table} + \mathbf{w}_d^\top \boldsymbol{d}_t + \mathbf{w}_y^\top \boldsymbol{y}_{t-1} + b) \quad (7)$$

where $\sigma$ stands for the sigmoid function while vectors $\mathbf{w}_c$, $\mathbf{w}_d$, $\mathbf{w}_y$ and scalar $b$ are trainable parameters. The joint probability for generating $y_t$ at time step $t$ is formulated as follows:

$$P(y_t|y_{<t}, D, F) = p_{\text{gen}} P_{\text{vocab}}(y_t) + (1 - p_{\text{gen}}) \sum_{i:v_i=y_t} \alpha_{t,i} \quad (8)$$

where $\alpha_{t,i}$ is the table attention weight defined in Equation 3.

### 2.5 Training

Given training dataset $\{(D^k, F^k, y^k)\}_{k=1}^S$ consisting of $S$ samples, our goal is to maximize the probability of target description $y = y_1, ...y_T$ given the input table $D$ and the background knowledge facts $F$. So the objective function is to minimize the negative log-likelihood:

$$J(\theta) = -\frac{1}{S} \sum_{k=1}^{S} \sum_{t=1}^{T_k} \log P(y_t^k | y_{<t}^k, D^k, F^k) \quad (9)$$

The objective function is fully differentiable, so the entire model can be trained end-to-end through backpropagation. Adam (Kingma and Ba, 2014) is used to optimize our model.

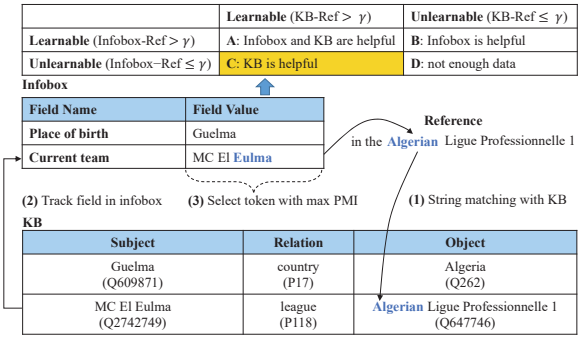


Figure 3: An illustration of KBGain metric.

## 3 When and Why External Knowledge is Beneficial

In this section, we introduce KBGain to quantify when and why external knowledge (KB) is effective, and then show how KBGain correlates with the performance boost of KBAtt over Seq2Seq+Copy in 21 datasets in Section 4.4. Intuitively, incorporating an external KB should improve data-to-text generation performance. However, to pinpoint the effect of the additional knowledge is not trivial since we know that (a) not all external knowledge is relevant and (b) neural models may memorize certain inference patterns when parallel data is big enough. We assume (1) matching tokens between the external KB and the references indicate relevance of the tokens in the external KB; (2) high frequency of co-occurrence of matching tokens between the infobox and the references indicate good potential for neural models to learn generation patterns which leads to less effectiveness of an external KB, i.e. no data sparsity in the original data. To characterize these factors, we introduce KBGain.

KBGain measures the portion of *learnable* tokens in the references co-occurred with their corresponding external KB entries but filter out those tokens which could also co-occurred with the infobox. We say a token is learnable from one source (infobox or KB) when the co-occurrence frequency between them is higher than a threshold $\gamma$ which is the minimum size of co-occurrence above which learning will be effective[3]. The top table in Fig. 3 summarizes learnable tokens in 4 possible cases. Among them, case **C** where the frequency of infobox and reference (Infobox-Ref) token co-occurrence does not exceed this threshold but its KB and reference (KB-Ref) token co-

[3]Here, $\gamma$ is set 25 by tuning on the development datasets, please see Appendix A.3 for more details.

| Dataset | WikiBio | Album |
|---|---|---|
| Avg. # tokens per sentence | 26.1 | 21.0 |
| Avg. # tokens per table | 53.0 | 57.1 |
| Avg. # fields per table | 19.7 | 15.9 |
| Avg. # table tokens per sent. | 11.6 | 9.7 |
| Avg. # entities per table | 5.9 | 5.5 |
| Avg. # fact tuples per entity | 19.0 | 7.2 |
| Dataset size | 728, 321 | 76, 105 |

Table 1: Dataset statistics for two sample datasets.

occurrence does, then this is where the KB will be effective. KBGain is defined as the average ratio between count of the tokens falling into category **C** and the length of their corresponding reference on the test set.

Specifically, for all tokens in the reference except stop words and punctuation, we select those matched[4] on string with the object tokens of KB but not with the infobox. For example, the token *Algerian* in the reference is matched with the object of the last KB tuple. The KB-Ref token pair is simply acquired by string matching while Infobox-Ref token pair can be tracked by firstly finding the corresponding field based on the matched tuple, and then selecting the token in its field value with the highest pointwise mutual information (PMI).

## 4 Experiments

### 4.1 Datasets

In the experiments, we adopt the WikiBio (Lebret et al., 2016) along with twenty new infobox-to-text datasets collected from Wikipedia[5]. The full statistics of these datasets could be found in the Appendix A.2. Table 1 shows the statistics of two sample datasets. Each dataset consists of infoboxes as input data and the first sentences of their corresponding Wikipeida articles as references. For example, on datasets WikiBio and Album, we extract 5.9 and 5.5 entities from each table, and each entity has 19.0 and 7.2 extended facts tuples, on average. WikiBio and the other 20 datasets consist of 728,321 and 668,796 instances respectively, and they together cover 84.34% infoboxes of the English Wikipedia[6].

[4]To make it simple, we adopt strict string matching. This is acceptable for rough quantification of the intended measurement since precise and semantic-aware matching is still an active research area. We also try several popular stemmers to expand tokens e.g., *Algeria* to *Algerian*, but no stemmers have such capability.

[5]These twenty datasets are similar to WikiBio but from different domains, e.g. Album, Book etc., which are characterized by infobox template category name. They are created with the similar procedure and the same Wikipedia dumps as outlined in Lebret et al. (2016). For more details about data collection, please refer to the Appendix A.1.

| Dataset | Seq2Seq+Copy | KBAtt | Δ |
|---|---|---|---|
| Single | 43.60 | 47.70 | 4.10 |
| Station | 54.30 | 56.77 | 2.47 |
| Australian_place | 43.73 | 45.73 | 2.00 |
| Album | 41.04 | 42.77 | 1.73 |
| NRHP | 48.97 | 50.43 | 1.46 |
| Airport | 45.17 | 46.61 | 1.44 |
| Book | 36.07 | 37.49 | 1.42 |
| Automobile | 18.64 | 19.95 | 1.31 |
| Building | 24.13 | 25.21 | 1.08 |
| UK_school | 33.64 | 34.72 | 1.08 |
| School | 37.30 | 38.33 | 1.03 |
| Football_club_season | 46.05 | 47.02 | 0.97 |
| UK_place | 41.46 | 42.38 | 0.92 |
| Military_unit | 37.74 | 38.58 | 0.84 |
| Military_conflict | 18.58 | 19.21 | 0.63 |
| WikiBio | 44.28 | 44.59 | 0.31 |
| Television_episode | 73.59 | 73.87 | 0.28 |
| NCAA_team_season | 87.37 | 87.58 | 0.21 |
| French_commune | 90.14 | 90.33 | 0.19 |
| Settlement | 77.59 | 77.68 | 0.09 |
| Video_game | 29.54 | 29.48 | -0.06 |

Table 2: BLEU-4 score with two generation models on 21 datasets.

### 4.2 Experiment Setup

We conduct experiments on the datasets as introduced in Section 4.1. Seq2Seq+Copy is the main baseline: a sequence-to-sequence (Seq2Seq) model equipped with copy mechanism from See et al. (2017) which is one of the state of art methods. We also compare our results with other published results using the WikiBio dataset. The model structure of our baseline model is most similar to Sha et al. (2017) by removing their specialized design on order planning which is not the focus of this paper. Since our aim is to verify the effectiveness of external knowledge for data-to-text task, we keep our baseline model as general as possible without other specialized design. Our primary model is KBAtt: a model which integrates the background knowledge into baseline model through a KB encoder and KB attention mechanism. We employ BLEU (Papineni et al., 2002) as the automatic evaluation metric. In addition to BLEU, we conduct human evaluation to assess the factual accuracy of generated sentences.

[6]Although the English Wikipedia has about 5 million entities, we totally parsed 1,656,458 infoboxes and drop some of them due to data noise which indicates nearly 2/3 entities have no infobox.

### 4.3 Training Details

The dimensions of all trainable word embeddings are set to 512, and the GRU hidden states sizes are set to 512. To limit the memory of our model, we set the maximum number of facts per table to 500. We initialize all the model parameters randomly using a uniform distribution between -0.08 and 0.08. For the model training, we use Adam (Kingma and Ba, 2014) with initial learning rate of 0.001 as the optimization algorithm. The training batch size is set to 64. We also apply gradient clipping (Pascanu et al., 2013) with range [-1,1] during training. We conduct experiments using single card NVIDIA Tesla V100. The largest 15 datasets with more than 9500 training instances are trained for 20 epochs, while the remaining 6 datasets are trained for 40 epochs. All the models were selected based on BLEU-4 score on the development set. All the experiments use greedy search as the decoding algorithm during testing.

### 4.4 Main Results

#### 4.4.1 Overall Results

To verify the effectiveness of KBAtt, we conduct experiments on 21 Wikipedia infobox-to-text datasets. Table 2 shows the performances of KBAtt and Seq2Seq+Copy in terms of BLEU-4 score. As we can see, KBAtt consistently outperforms Seq2Seq+Copy on most of the datasets, i.e., more than 0.5 BLEU-4 improvements on 15 out of 21 datasets, and comparable on the remaining 6 categories. To get a better understanding of when and why external knowledge will be effective or not, we correlate the performance gains in terms of BLEU-4 with **KBGain** metric described in Section 3 across 21 datasets, and plot their values on the scatter plot in Fig. 4. The *Pearson correlation coefficient* ($\rho$) between BLUE-4 improvements and KBGain is 0.716 which indicates a strong positive correlation between them. This confirms our analysis in Section 3 that KBAtt is effective when the external knowledge is relevant and the original data is sparse.

#### 4.4.2 Results on WikiBio

To compare with state-of-the-art models, Table 3 shows the results on the WikiBio dataset. Among them, our baseline Seq2Seq+Copy gains a performance of 44.28 which is comparable to Sha et al. (2017) and Liu et al. (2017) and significantly better than Lebret et al. (2016) and Bao et al. (2018),

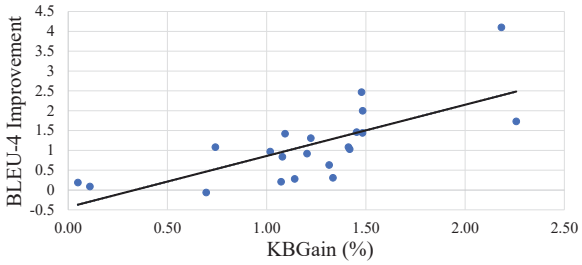


Figure 4: Strong correlation ($\rho = 0.716$) between KBGain ($x$) and absolute BLEU-4 improvement ($y$).

and this baseline model is general without specific design, e.g., order planning from Sha et al. (2017) or field-gating encoder from Liu et al. (2017). Our proposed model, namely KBAtt, obtains a BLEU-4 score of 44.59. Although the absolute improvement (+0.31 BLEU-4 points) of KBAtt over Seq2Seq+Copy is relative small, the difference between them is statistically significant under the one-tailed paired t-test at the 99% significance level. The reason why the absolute improvement is relative small is that the full WikiBio dataset consists of 728,321 parallel data-to-text pairs which are enough for neural models to memorize certain inference patterns for high frequency pairs. However, as will be shown in the Section 4.5, the baseline fails on low co-occurrence frequency pairs, but the KBAtt avoids this problem with the contributions from the external knowledge.

#### 4.4.3 Human Evaluation

We conduct human evaluation to assess the factual accuracy of the generated sentences. Manually evaluating the generated results of all datasets is labour intensive, so we choose to evaluate two sample datasets for case study purposes. Specifically, we sample 50 instances each from the WikiBio and the Album datasets, and ask two annotators to extract facts tuples (subject, relation, object) from the references and the generated sentences[7]. In table 4, precision $P_1$, recall $R_1$ and their overall score $F_1$ measure the extent that the facts extracted from the generated sentences conform to those from references. As we can see, KBAtt achieves 1.21%, 7.42% improvements in terms of $F_1$ on WikiBio and Album respectively, which shows that KBAtt can generate more relevant facts with respect to the reference than the

[7]The generated sentences are shuffled to avoid model preferences. The Cohen's kappa between two annotators is 0.776.

| Model | BLEU-4 |
|---|---|
| Table NLM (Lebret et al., 2016) | 34.70 |
| Table2Seq (Bao et al., 2018) | 40.26 |
| Order Planning, full model (Sha et al., 2017) | 43.91 |
| Field-gating Seq2Seq, full model (Liu et al., 2017) | 44.71 |
| Seq2Seq+Copy | 44.28 |
| KBAtt | 44.59 |

Table 3: BLEU-4 scores (%) on WikiBio dataset.

| Dataset | Model | $P_1$ | $R_1$ | $F_1$ | $P_2$ |
|---|---|---|---|---|---|
| WikiBio | Seq2Seq+Copy | 69.07% | 62.45% | 65.59% | 88.35% |
| WikiBio | KBAtt | 71.81% | 62.45% | 66.80% | 91.85% |
| Album | Seq2Seq+Copy | 73.33% | 71.84% | 72.58% | 84.38% |
| Album | KBAtt | 81.70% | 78.37% | 80.00% | 90.64% |

Table 4: Human based evaluation results.

Seq2Seq+Copy baseline. We further ask the annotators to judge whether the facts extracted from the generated texts are correct against information from the infobox, the external knowledge (eg. Wikidata), or even search engines. In table 4, $P_2$ measures the ratio of correct facts in the generated results. We observe that KBAtt improves 3.50% and 6.26% over Seq2Seq+Copy in WikiBio and Album respectively. This shows that KBAtt is more likely to generate accurate facts than Seq2Seq+Copy.

### 4.5 Analysis of Few-Shot Learning Ability

To examine the ability of learning writing knowledge from few examples, we design an experiment to compare the performance under different number of training samples for the baseline and our model. In the training set of WikiBio, about 78.5% tables contain *place of birth* field but only 19.0% tables include the *nationality* field. However, in the references, *nationality* of a person is frequently mentioned. This means that the ability of inferring *nationality* based on *place of birth* is important. We collect the cases from the test data with the following conditions: (a) only city level information is given in the field of place of birth; (b) nationality is not specified in the table; (c) the reference mentions country name or nationality. From these cases, we get over 400 unique places of birth to nationality inference pairs and split them into two intervals $[1, 25)$ and $[25, \infty)$ according to their co-occurrence frequency in the training set[8]. For each interval, we randomly sample 50 test cases and

[8]The interval threshold 25 is set by following that for KB-Gain.

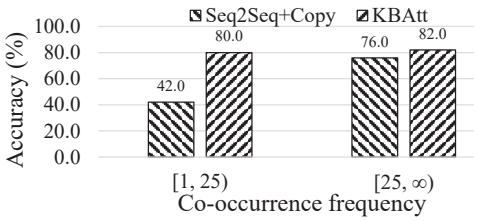


Figure 5: Accuracy (%) of nationality information in the generated sentences w.r.t different co-occurrence frequency intervals.

manually assess the accuracy of nationality information mentioned in the generated sentences[9].

Fig. 5 shows the accuracy of nationality information in the generated text with respect to different co-occurrence frequency intervals: Firstly, we found that the baseline model struggles to learn the inference from *place of birth* to *nationality* when their co-occurrence frequency in training set is less than 25, which shows the difficulty of this task; Secondly, in interval $[1, 25)$, the baseline model only gets 42.0% accuracy, but our model achieves 80.0% accuracy, a 38.0% absolute gain. This confirms our motivation that incorporating external knowledge into neural models can improve model performances especially when the original data is sparse; Finally, the accuracy of both models increases as the frequency goes from $[1, 25)$ to $[25, \infty)$, and the improvement of our model over the baseline model gradually narrows, from 38.0% to 6.0%. This shows that the baseline model could learn part of inference patterns when enough parallel data is available.

### 4.6 Case Study

Fig. 6 shows three examples from the development set of WikiBio dataset which can demonstrate how KBAtt succeeds or fails. Fig. 6 (a) and (b) illustrate KBAtt is helpful when original data is sparse. However when original data is dense, the baseline model still can learn it. As shown in Fig. 6 (a), the baseline model struggles to learn direct association between "german"and "gummersbach" (birth place in the table), since they only co-occur 12 times in the training data. But it is easy for KBAtt to learn since "german" co-occurs with "germany" in KB 12,437 times in the training data. As shown in Fig. 6 (b), although the national-

[9]In this experiment, a generated result is judged correct only if it generates the same nationality information as the reference.

| | | | | | |
|---|---|---|---|---|---|
| **Article title** | dan van husen | **Reference** | dan van husen ( born april 30 , 1945 ) is a **german** actor who has also performed in hollywood films . | | |
| **Birth date** | 30 april 1945 | **Seq2Seq+Copy** | dan van husen ( born 30 april 1945 ) is a **british** actor , screenwriter and producer . | | |
| **Name** | dan van husen | **KBAtt** | dan van husen ( born 30 april 1945 in gummersbach ) is a former **german** actor . | | |
| **Birth place** | gummersbach | **Field Name** | **Subject** | **Relation** | **Object** |
| **Caption** | dan van husen | Birth place | gummersbach | country | germany |
| **Years active** | 1968 -- present | Birth place | gummersbach | instance of | district capital |

(a)

| | | | | | |
|---|---|---|---|---|---|
| **Article title** | stephanie moorhouse | **Reference** | stephanie moorhouse ( born january 20 , 1987 ) is an **australian** former artistic gymnast . | | |
| **Birth date** | 20 january 1987 | **Seq2Seq+Copy** | stephanie moorhouse ( born 20 january 1987 ) is an **australian** artistic gymnast . | | |
| **Birth place** | melbourne | **KBAtt** | stephanie moorhouse ( born 20 january 1987 ) is an **australian** artistic gymnast . | | |
| **Weight** | 47 kg lb | **Field Name** | **Subject** | **Relation** | **Object** |
| **Discipline** | wag | Birth place | melbourne | country | australia |
| **Level** | senior international | Birth place | melbourne | instance of | city |

(b)

| | | | | | |
|---|---|---|---|---|---|
| **Article title** | henry roberts ( architect ) | **Reference** | henry roberts ( 16 april 1803 -- 9 march 1876 ) was a **british** architect best known for fishmongers ' hall in london and for his work on model dwellings for workers . | | |
| **Birth date** | 1803 | **Seq2Seq+Copy** | henry roberts ( 1803 -- 1876 ) was an **english** architect who designed many buildings in florence , florence , and florence . | | |
| **Birth place** | philadelphia | **KBAtt** | henry roberts ( 1803 -- 1876 ) was an **american** architect and architect . | | |
| **Death date** | 1876 | **Field Name** | **Subject** | **Relation** | **Object** |
| **Death place** | florence | Birth place | philadelphia | country | united states of america |
| **Occupation** | architect | Birth place | philadelphia | instance of | city |

(c)

Figure 6: Case study. Three generation examples from development set of WikiBio. Each example consists of the input infobox, parts of external knowledge base, the reference and two generated sentences by Seq2Seq+Copy and KBAtt. We mark correct fact information as blue and incorrect ones as red.

ity information is not explicitly stated in the table, both Seq2Seq+Copy and KBAtt generate correct nationality information ("australian"). The reason is that "australian" co-occurs with "melbourne" in the table 3,391 times in the training data. So, it is easy for the neural model to learn such inference patterns. Fig. 6 (c) illustrates the pattern of inferring nationality from birth place is not always correct. In this case, KBAtt makes a plausible inference on nationality from birth place, so it generates "american". However, this inference pattern doesn't hold for this case, because he is British. Hopefully, such cases are rare since the birth place conforms to the nationality in most of cases, so our methods can bring improvement as indicated by the overall better performance across almost all Wikipedia categories.

## 5 Related Work

Data-to-text generation is an important task in natural language generation which has been studied for decades (Kukich, 1983; Holmes-Higgin, 1994; Reiter and Dale, 1997). This task is broadly divided into two subproblems: *content selection* (Kukich, 1983; Reiter and Dale, 1997; Duboue and McKeown, 2003; Barzilay and Lapata, 2005) and *surface realization* (Goldberg et al., 1994; Reiter et al., 2005).

With the advent of neural text generation, the distinction between *content selection* and *surface realization* becomes blurred. For example, Mei et al. (2016) proposed an end-to-end encoder-aligner-decoder model to learn both content selection and surface realization jointly which shows good results on WeatherGov and RoboCub datasets. Wiseman et al. (2017) generate long descriptive game summaries from a database of basketball games where they show the current state-of-the-art neural models are quite good at generating fluent outputs, but perform poorly in content selection and capturing long-term structure. Our work falls into the task of single sentence generation from Wikipedia infoboxes. The model structure ranges from feed-forward networks work (Lebret et al., 2016) to encoder-decoder models (Sha et al., 2017; Liu et al., 2017; Bao et al., 2018; Nema et al., 2018). Recently, Perez-Beltrachini and Lapata (2018) generalize this task to multi-sentence text generation, where they focus on bootstrapping generators from loosely aligned data. However, most of the work mentioned above assume all the writing knowledge can be learned from massive parallel pairs of training data. Different from the previous work, we exploit incorporating external

knowledge into this task to improve the fidelity of generated text.

Our work is also relevant to recent works on integrating external knowledge into neural models for other NLP tasks. The motivations of incorporating external knowledge range from enriching the context information (Mihaylov and Frank, 2018) in reading comprehension, improving the inference ability of models (Chen et al., 2018) in natural language inference, to providing the model a knowledge source to copy from in language modelling (Ahn et al., 2016). Our model, KBAtt, is most relevant to Mihaylov and Frank (2018), where they focus on similarity calculation but we focus on generation in this paper. Moreover, in addition to demonstrating the positive effect of incorporating external knowledge as previous work, we also design a new metric to quantify the potential gains of external knowledge for a specific dataset which can explain when and why our model is effective.

## 6 Conclusion

In this paper, we propose a neural data-to-text generation model, KBAtt, that incorporates external background knowledge in a simple but effective way to improve fidelity of the generated text. Experiments on 21 Wikipedia infobox-to-text datasets show KBAtt consistently achieves better performance in BLEU than a state-of-the-art baseline on most of datasets. Meanwhile, to quantify when and why external knowledge is effective, we design a metric, KBGain, which shows a strong correlation with observed performance boost. This result indicates the relevance of external knowledge and sparseness of original data are the main factors affecting the effectiveness of KBAtt.

In the future, we plan to investigate integrating multi-hops knowledge graph behind the data which has potential to further improve the inference ability of neural models. It will be worthwhile especially when we extend the task to multiple sentence generation. The main challenge in integrating multi-hops knowledge graph is the large search space. We plan to employ reinforcement learning based techniques to allow the model to search the optimal inference paths by trial and error. Besides, we are also interested in integrating external knowledge into other types of datasets beyond Wikipedia infobox-to-text datasets.

## 7 Acknowledgement

We would like to thank the anonymous reviewers for their helpful comments. The work of this paper is funded by the project of National Key Research and Development Program of China (No. 2018YFC0832105).

## References

Sungjin Ahn, Heeyoul Choi, Tanel Pärnamaa, and Yoshua Bengio. 2016. A neural knowledge language model. *arXiv preprint arXiv:1608.00318*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representations*.

Junwei Bao, Duyu Tang, Nan Duan, Zhao Yan, Yuanhua Lv, Ming Zhou, and Tiejun Zhao. 2018. Table-to-text: Describing table region with natural language. *AAAI*.

Regina Barzilay and Mirella Lapata. 2005. Collective content selection for concept-to-text generation. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 331–338. ACL.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, and Diana Inkpen. 2018. Natural language inference with external knowledge. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Pablo A Duboue and Kathleen R McKeown. 2003. Statistical acquisition of content selection rules for natural language generation. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 121–128. ACL.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2018. Findings of the E2E NLG Challenge. In *Proceedings of the 11th International Conference on Natural Language Generation*, Tilburg, The Netherlands. ArXiv:1810.01170.

Eli Goldberg, Norbert Driedger, and Richard I Kittredge. 1994. Using natural-language processing to produce weather forecasts. *IEEE Intelligent Systems*, (2):45–53.

Paul Holmes-Higgin. 1994. Text generation: using discourse strategies and focus constraints to generate natural language text. *The Knowledge Engineering Review*, 9(4):421–422.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Karen Kukich. 1983. Design of a knowledge-based report generator. In *Proceedings of the 21st annual meeting on Association for Computational Linguistics*, pages 145–150. ACL.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. 2017. Table-to-text generation by structure-aware seq2seq learning. *AAAI*.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland. Association for Computational Linguistics.

Hongyuan Mei, Mohit Bansal, and Matthew R Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. *NAACL*.

Todor Mihaylov and Anette Frank. 2018. Knowledgeable reader: Enhancing cloze-style reading comprehension with external commonsense knowledge. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Preksha Nema, Shreyas Shetty, Parag Jain, Anirban Laha, Karthik Sankaranarayanan, and Mitesh M Khapra. 2018. Generating descriptions from structured data using a bifocal attention mechanism and gated orthogonalization. *NAACL*.

Feng Nie, Jinpeng Wang, Jin-Ge Yao, Rong Pan, and Chin-Yew Lin. 2018. Operations guided neural networks for high fidelity data-to-text generation. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318. ACL.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *The 30th International Conference on Machine Learning*, pages 1310–1318.

Laura Perez-Beltrachini and Mirella Lapata. 2018. Bootstrapping generators from noisy data. *NAACL*.

Ehud Reiter and Robert Dale. 1997. Building applied natural language generation systems. *Natural Language Engineering*, 3(1):57–87.

Ehud Reiter, Somayajulu Sripada, Jim Hunter, Jin Yu, and Ian Davy. 2005. Choosing words in computer-generated weather forecasts. *Artificial Intelligence*, 167(1-2):137–169.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Lei Sha, Lili Mou, Tianyu Liu, Pascal Poupart, Sujian Li, Baobao Chang, and Zhifang Sui. 2017. Order-planning neural text generation from structured data. *AAAI*.

Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2017. Challenges in data-to-document generation. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

# A Supplemental Material

## A.1 Dataset Collection

Our dataset is an extension of WikiBio dataset (Lebret et al., 2016) where we extend previous single Biography domain to other twenty domains (e.g. Album, Book etc.) and augment the infoboxes with external knowledge from Wikidata[10]. Our dataset consists of infoboxes as input data and the first sentence of their corresponding Wikipedia article as reference. The infoboxes are parsed from English Wikipedia dump (Sep 2015) while the reference sentences are tokenized with Stanford CoreNLP (Manning et al., 2014). Different from (Lebret et al., 2016), we keep the capitalization information which we believe is important for real-world application. To augment the infoboxes with external knowledge, we preserve the Wikipedia internal hyperlink information in the field values of infoboxes, and track these hyperlinks to get their corresponding entities from Wikidata (dumps version: 20150831). Finally we retrieve one-hop facts from Wikidata. The dataset is aviliable at `https://github.com/hitercs/WikiInfo2Text`.

## A.2 Full Dataset Statistics

Table 5 summarizes the full dataset statistics about the twenty-one datasets used in this paper. As you can see, WikiBio consists of 728,321 instances which is the largest infobox-to-text dataset

[10] `https://wikidata.org`

| Dataset | Avg. # tokens per sentence | Avg. # tokens per table | Avg. # fields per table | Avg. # table tokens per sent. | Avg. # entities per table | Avg. # fact tuples per entity | Dataset size |
|---|---|---|---|---|---|---|---|
| WikiBio | 26.1 | 53.0 | 19.7 | 11.6 | 5.9 | 19.0 | 728321 |
| Settlement | 22.1 | 57.3 | 50.6 | 10.3 | 6.6 | 14.4 | 327863 |
| Album | 21.0 | 57.1 | 15.9 | 9.7 | 5.5 | 7.2 | 76105 |
| Single | 24.3 | 70.5 | 19.2 | 12.6 | 7.3 | 7.5 | 36937 |
| NRHP | 23.0 | 54.4 | 22.7 | 10.8 | 1.4 | 10.9 | 35945 |
| French_commune | 14.7 | 31.1 | 20.3 | 3.7 | 0.8 | 126.1 | 29218 |
| Book | 24.3 | 41.7 | 20.1 | 11.1 | 4.2 | 16.7 | 23859 |
| School | 24.2 | 62.6 | 44.3 | 11.2 | 3.7 | 29.8 | 18751 |
| Station | 19.9 | 51.8 | 32.6 | 9.0 | 3.2 | 18.0 | 16625 |
| Video_game | 25.0 | 42.8 | 13.2 | 9.4 | 6.3 | 6.4 | 16381 |
| UK_place | 20.5 | 30.9 | 19.4 | 7.0 | 3.0 | 5.2 | 12279 |
| Military_unit | 25.4 | 62.5 | 27.9 | 11.9 | 5.2 | 18.9 | 11273 |
| Airport | 22.9 | 74.4 | 32.9 | 10.7 | 3.0 | 15.0 | 10934 |
| Building | 24.1 | 47.6 | 35.8 | 10.4 | 2.9 | 33.6 | 10052 |
| Military_conflict | 34.7 | 96.6 | 18.0 | 18.6 | 8.1 | 17.3 | 9592 |
| NCAA_team_season | 24.2 | 45.1 | 23.4 | 12.7 | 2.7 | 5.5 | 7766 |
| Television_episode | 26.4 | 78.8 | 15.9 | 12.4 | 10.1 | 8.0 | 5930 |
| UK_school | 23.8 | 51.6 | 48.0 | 10.1 | 4.1 | 10.0 | 4974 |
| Australian_place | 21.3 | 43.8 | 29.1 | 8.2 | 5.6 | 6.5 | 4972 |
| Automobile | 23.3 | 70.9 | 17.2 | 7.8 | 5.7 | 18.5 | 4781 |
| Football_club_season | 24.6 | 82.8 | 30.0 | 9.3 | 8.8 | 6.9 | 4559 |

Table 5: Dataset statistics for all datasets used in this paper.

in Wikipedia. However, the size of remaining datasets ranges from 4,559 to 327,863 and more than 60% of them have less than 20,000 instances. So, the abundance of data such as WikiBio is not common among the Wikipedia infobox-to-text datasets. Meanwhile, these datasets vary in sentence length and length of tokens in table. In addition, the number of entities extracted from each table ranges from 0.8 to 10.1 and the average number of fact tuples per entity ranges from 5.2 to 126.1.

### A.3 Threshold Tuning for KBGain metric

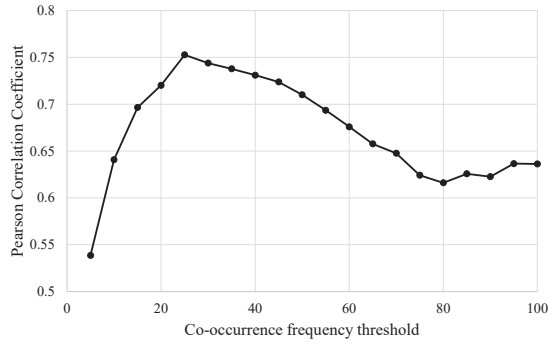


Figure 7: Pearson correlation coefficient w.r.t Co-occurrence frequency threshold.

The threshold $\gamma$ in KBGain metric denotes the minimum size of co-occurrence above which learning will be effective. We tune this threshold by finding the maximum *pearson correlation coefficient* ($\rho$) between BLEU-4 improvements and the values of KBGain on the 21 development datasets. Fig. 7 shows the curve of *pearson correlation coefficient* w.r.t co-occurrence frequency threshold ranging from 5 to 100. As we can see, $\rho$ reaches peak value as 0.753 when the threshold is 25. So $\gamma$ is set 25 throughout this paper.

# Reading Like HER: Human Reading Inspired Extractive Summarization

**Ling Luo**[1,3], **Xiang Ao**[1,3]*, **Yan Song**[2], **Feiyang Pan**[1,3], **Min Yang**[4], **Qing He**[1,3]
[1]Key Lab of Intelligent Information Processing, Institute of Computing Technology,
Chinese Academy of Sciences, Beijing, China
[2]Sinovation Ventures
[3]University of Chinese Academy of Sciences
[4]Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences
{luoling18s, aoxiang, panfeiyang, heqing}@ict.ac.cn, clksong@gmail.com

## Abstract

In this work, we re-examine the problem of extractive text summarization for long documents. We observe that the process of extracting summarization of human can be divided into two stages: 1) a *rough reading* stage to look for sketched information, and 2) a subsequent *careful reading* stage to select key sentences to form the summary. By simulating such a two-stage process, we propose a novel approach for extractive summarization. We formulate the problem as a contextual-bandit problem and solve it with policy gradient. We adopt a convolutional neural network to encode gist of paragraphs for rough reading, and a decision making policy with an adapted termination mechanism for careful reading. Experiments on the CNN and Daily-Mail datasets show that our proposed method can provide high-quality summaries with varied length, and significantly outperform the state-of-the-art extractive methods in terms of ROUGE metrics.

## 1 Introduction

Automatic text summarization has wide popularity in NLP applications such as producing digests, headlines and reports. Among the supervised methods, two main types are usually explored, namely abstractive and extractive summarizations (Nenkova et al., 2011). Compared with abstractive approaches, extractive methods are more practical and applicable as they are faster, simpler and more reliable on grammar as well as semantic information (Yao et al., 2018).

Recent studies (Cheng and Lapata, 2016; Nallapati et al., 2017; Yasunaga et al., 2017; Feng et al., 2018) consider extractive summarization as a sequence labeling task, where each sentence is individually processed and determined whether it should be extracted or not. Various neural networks are used to label each sentence and trained using cross-entropy loss to maximize the likelihood of the ground-truth labeled sequences, which may derive the mismatch between the cross-entropy objective function and the evaluation criterion. On the other hand, some reinforcement learning based methods (Wu and Hu, 2018; Narayan et al., 2018; Yao et al., 2018) directly optimize the evaluation metric by combining cross-entropy loss with rewards and train model with policy gradient reinforcement learning. Note that the rewards usually reflect the quality of extracted summary and measured by standard evaluation protocol. However, they still sequentially process text and tend to extract earlier sentences over later ones due to the sequential nature of selection (Dong et al., 2018).

Although great efforts have been devoted to this field, most of the existing approaches neglect how human being reads and forms summaries. Human beings are very good at refining the main idea of a given text based on their reading cognitive process. Note that the reading habits of native speakers are varied and hard to modeled, so we adopt the three reading phases of second-language readers where there are potential behavior patterns. Such reading process of second-language readers generally includes *pre-reading*, *reading* and *post-reading* (Avery and Graves, 1997; Saricoban, 2002; Toprak and Almacıoğlu, 2009; Pressley and Afflerbach, 2012). In the *pre-reading* stage, they roughly preview the whole text to form an initial cognition and extract general but coarse-grained information at the meantime. Based on such prior knowledge, the subsequent *reading* stage is a conscious process that focuses on target-specific purposes to search fine-grained details through repeated skimming and scanning. For *post-reading*, re-reading is performed to check whether there are

*Corresponding author.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3033–3043,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

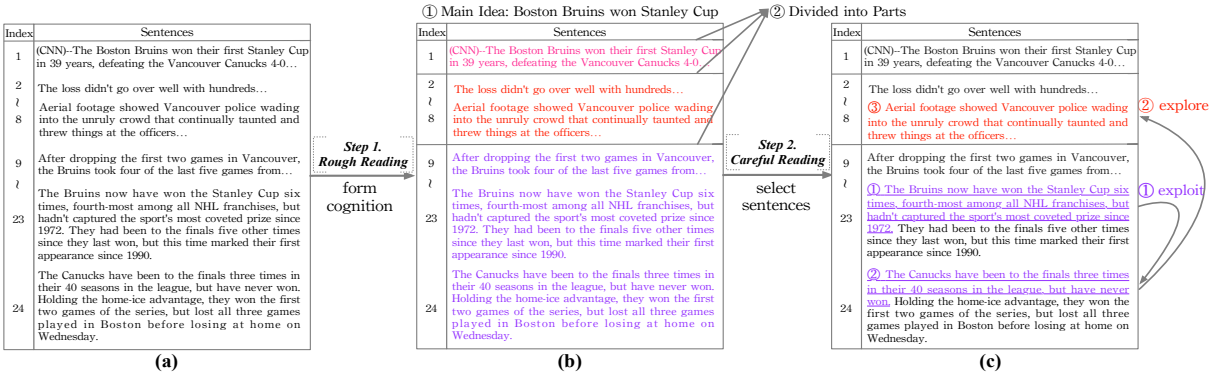


Figure 1: An example of how human beings extract summary. The article is from CNN/DailyMail dataset.

any missed details. The three-stage reading process makes it effective in capturing essential sentences of text as the extracted summarization.

Inspired by such human's reading cognitive process, in this paper, we re-examine the problem of extractive summarization and propose a new approach **HER** (**H**uman-b**E**ing-**R**eading inspired extractive summarization). We simplify the three-stage reading process to two subsequent stages called *rough reading* and *careful reading*. In *rough reading*, coarse-grained information of the original context is identified to form a general cognition. A detailed case is shown in Figure 1. In Figure 1 (a) and (b), after browsing on the whole article, the main idea is outlined and the text is roughly divided into three parts based on the gist of paragraphs at the meantime. Each part describes related but not the same contents. To implement the *rough reading* process, we use a hierarchical neural network to encode sentence vectors and derive a document representation as global feature for the main idea. Meanwhile, a convolutional neural network (CNN) is utilized to encode local features from different paragraphs.

During *careful reading*, the model searches for specific but important details through re-readings to cover the content and extract essential fine-grained information as the final summary. For instance, as shown in Figure 1 (c), after rough reading, two sentences close to the main idea "Boston Bruins won Stanley Cup" may be selected firstly. Then an earlier and more detailed sentence about "fans rioting" is appended to the summary by performing re-reading. It is a combination of people's *reading* and *post-reading* process. To accomplish this, we train a neural network to score each sentence. A multi-armed bandit policy with an adapted termination mechanism is then used to form the final summary.

In our HER model, the whole process is formulated as a contextual bandit problem. We train a reinforcement learning agent to solve it using the policy gradient method (Sutton et al., 2000). At each step, the agent takes an action which is a to-be-selected sentence set, and then receives a reward based on the correlation between extractive summary and gold-standard reference summary.

Our main contributions are as follows:

- We propose an extractive summarization method that simulates human being's reading cognitive process. We formulate it as a contextual bandit problem in which two stages including rough reading and careful reading are devised.
- We use a hierarchical neural network for rough reading which consists of a neural net to encode the whole document and another one to capture features in paragraphs. Then we use a contextual-bandit agent to flexibly select sentences during careful reading, with an adapted termination mechanism to select various but proper numbers of sentences.
- We conducted experiments on the CNN and DailyMail datasets and showed that our proposed model can outperform state-of-the-art methods and provide high-quality summaries.

## 2 The HER Model

In this section, we introduce the overall framework of our model, HER. We formulate extractive summarization as a contextual bandit (Langford and Zhang, 2007) trained using policy gradient reinforcement learning.

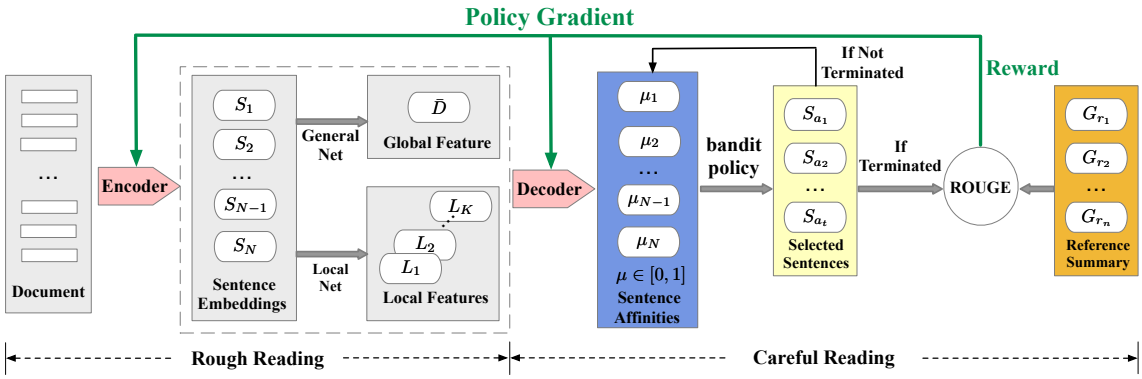


Figure 2: The overall framework of HER is formulated as a contextual bandit and can be divided into a two-stage process containing rough reading and careful reading.

As illustrated in Figure 2, the framework can be divided into two stages: rough reading and careful reading. During rough reading, a document with $N$ sentences is encoded into sentence vectors $\{S_1, S_2, \ldots, S_N\}$ as well as a set of features denoted as $F$, which includes one global feature $\bar{S}$ representing the whole documentary information and $K$ local features $\{L_1, L_2, \ldots, L_K\}$ depicting paragraphical contentsIn careful reading, sentence vectors are decoded into real-valued scores called sentence affinities $\{\mu_1, \mu_2, \ldots, \mu_N\}$, which can be considered as an estimation of sentence correlation to cover the context. Then a bandit policy is used to repeatedly choose unique sentence until the termination mechanism is triggered.

We will detail the preliminaries in Sec. 2.1, the rough reading stage in Sec. 2.2, and the careful reading stage in Sec. 2.3. The training process is illustrated in Sec. 2.4.

### 2.1 Summarization as Contextual-Bandit

Contextual-bandit (Langford and Zhang, 2008; Li et al., 2010; Pan et al., 2019a) is the multi-armed bandit (Auer et al., 2002) problem with featured contexts. At each step, the agent observes a context, selects an action based on the context, and then receives a reward. The agent's goal is to quickly find a decision making policy to maximize its return.

In extractive summarization, the goal of the task is to extract an undetermined number of sentences from the original document as summary. We show that it can be formulated as a contextual bandit problem if we select the sentences sequentially. Specifically, for each document, the *context* includes its documentary representation learned through rough reading. At each step $t = 1, 2, \ldots,$ we define the *action* as the index of the next sentence to select. The agent keeps selecting sentences until it reaches the termination condition (detailed in Sec. 2.3.3). Finally, the selected sentences SUM $= (S_{a_1}, \ldots, S_{a_M})$ corresponding to the selected actions $(a_1, \ldots, a_M)$ form an extractive summary. The agent will receive a reward $R(\text{SUM}; G)$. Note that $G$ is the manually-labeled gold-standard summary of the document $D$, and the reward $R(\text{SUM}; G)$ measures the correlation between $G$ and the predicted summary SUM.

### 2.2 Rough Reading

In rough reading, we aim to form a general cognition on a given document which encodes the document into sentence embeddings as well as produces the feature set $F$ including global and local features.

Specifically, bidirectional LSTMs (BiLSTMs) on word- and sentence-level are first used to encode a document with $N$ sentences into $d_s$-dimentional sentence embeddings $\{S_1, S_2, \ldots, S_N\}$, $S_i \in \mathbb{R}^{d_s}$. Second, a global feature $\bar{S} \in \mathbb{R}^{d_s}$ is computed as an average of all the sentence vectors. Third, we use a convolutional neural network to refine gist of different paragraphs and generate multiple local features on the sentence level, which is different from previous methods (Kim, 2014; Narayan et al., 2017; Yao et al., 2018) processing on the word level. In detail, a stacking of $N$ sentence vectors is represented as,

$$S_{1:N} = [S_1, S_2, \cdots, S_N] \in \mathbb{R}^{N \times d_s}, \quad (1)$$

We apply 1-D convolutional neural networks on $S_{1:N}$ followed with a max-over-time pooling so that a final document-level representation can be

extracted. Specifically, we altogether used $K$ convolutional nets with $K$ different window sizes to summarize different gists of paragraphs. Finally, by stacking the outputs together, we get the final document-level representation for local features $L_{1:K} \in \mathbb{R}^{K \times d_s}$.

### 2.3 Careful Reading

Now that we have the sentence vectors and document-level features given by rough reading, we can perform careful reading to select the sentences one by one to form the summary.

#### 2.3.1 Sentence Decoder

In order to extract high-quality summaries, we first compute the sentence affinities by a *sentence decoder*, which is observed effective in Dong et al. (2018). The sentence affinities are calculate by the following principles: (1) *Salience* (The sentences whose meanings are close to the central idea should be emphasized); (2) *Coverage* (The sentences that match different paragraphical information should be encouraged); (3) *Redundancy* (The unselected sentences which are similar to already extracted ones should be inhibited).

As we need to learn the relations between each sentence and the rest of the document, we update the sentence representations by,

$$S'_t = S_t \oplus \bar{S} \oplus L_{1:K}, t = 1, \ldots, N. \quad (2)$$

Then, we utilize a decoder $\mathrm{Dec}_1$ to score the Salience and Coverage for each sentences, and a secondary score function $\mathrm{Dec}_2$ to screen the sentences that might have Redundancy. Specifically,

$$\boldsymbol{\mu}_1 = (\mu_{11}, \ldots, \mu_{1N})^\top = \mathrm{Dec}_1(S'_{1:N}), \quad (3)$$
$$\boldsymbol{\mu}_2 = \mathrm{Dec}_2(S'_{1:N} \circ (1 - \boldsymbol{\mu}_1)), \quad (4)$$

We implement $\mathrm{Dec}_1$ and $\mathrm{Dec}_2$ as multi-layered perceptrons. Finally, we average the two scores as the final sentence affinities.

$$\boldsymbol{\mu} = (\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2)/2. \quad (5)$$

#### 2.3.2 Bandit Policy

The overall decision making process goes as follows. The agent selects one sentence at each step based on the contextual information provided by the rough reading and sentence affinities computed by the sentence decoder. It stops taking actions once the termination mechanism is triggered. After that, all the selected sentences are formed as a summary, and a final reward can be calculated by comparing to the labeled golden summary. As there is no intermediate reward before termination, the goal of the agent is to find a policy to maximize its expected long-term return.

It is an intuitive choice to select sentences with the highest affinities as summary, which is similar to training data selection (Song et al., 2012; Liu et al., 2019). However, such an *argmax* policy is prone to only learn the easy patterns since it lacks exploration. We will show an example in Section 4.3. They should be explored to form the summary as well. Since the search space for summarization is extremely large, we must explicitly address the tradeoff between exploration and exploitation for fast learning, which is an active research area in reinforcement learning and applications (Pan et al., 2019a,b). In our work, we find the use of $\epsilon$-greedy with stochastic policy works well enough to encourage exploration. Specifically, with a probability of $1 - \epsilon$, the agent chooses the sentence following the current policy, i.e., to sample an index $a_t \in [1, N]$ from the multinomial distribution with sentence affinities $\{\mu_1, \mu_2, \ldots, \mu_N\}$ as probabilities. Otherwise, with a small probability of $\epsilon$, the agent randomly picks one available sentence as an exploration. Note that such exploration is only used during training.

#### 2.3.3 Termination Mechanism

In HER, we propose a termination mechanism that is independent on future rewards to make our model flexible in extracting summary with various numbers of sentences. This mechanism is defined as follows

$$\text{Done} \sim \text{Bernoulli}(\min(\frac{\mu_{\min}}{\mu_{\max}}, 1 - \mu_{\max})), \quad (6)$$

where $\mu_{\max}$ and $\mu_{\min}$ denotes the maximal and minimal value in $\boldsymbol{\mu}$ for all the remained sentences, respectively. Thus Done $= 1$ terminates the sentence extraction when there is no key sentences left. With this mechanism, the agent will stop extraction with high probability as long as the differences among remaining affinities are small enough or the remaining sentence affinities are very low.

### 2.4 Training

After the agent sequentially takes an action $a_t$ until terminated, we can derive an summary induced by $a$ out of a document $D$. Then the agent would receive a reward $R(\text{SUM}; G)$ where $G$ is the gold-standard summary of $D$. $R(\text{SUM}; G)$ is computed

by the average of three variants of ROUGE (Lin, 2004). To balance precision and recall, we use $F$-score here,

$$R(\text{SUM}; G) = \frac{1}{3}(\text{ROUGE-1}_f(\text{SUM}; G) + \text{ROUGE-2}_f(\text{SUM}; G) + \text{ROUGE-L}_f(\text{SUM}; G)). \quad (7)$$

We represent the whole extractive neural network as $p_\theta(\cdot|D)$ containing the encoder in rough reading and the decoder in careful reading. The goal of our model is to find parameters $\theta$ of $p_\theta$ to produce high-quality summary and maximize the rewards (c.f. Eq. (8)). But we cannot obtain gradient to maximize Eq. (8) with gradient ascent as it is discretely sampled. So we use the likelihood ratio gradient estimator, also known as REINFORCE (Williams, 1992; Sutton et al., 2000), to acquire the gradient by Eq. (9).

We use $Q(D)$ in Eq. (10) to construct $p_\theta(a|D)$ following Dong et al. (2018), where $z(D) = \sum_t \mu_t(D)$ and $\epsilon$ is the exploration probability of the $\epsilon$-greedy denoted in Sec. 2.3.2. $M$ is the number of extracted sentences this is determined jointly by the termination mechanism and the document context. $Q(D)^{\frac{1}{M}}$ is adopted to present $p_\theta(a|D)$ to avoid extracting fewer or more sentences when maximizing the objective function. Hence, $\bigtriangledown_\theta \text{log} p_\theta(a|D)$ in Eq. (9) can be easily computed.

$$J(\theta) = E[R(\text{SUM}; G)] \quad (8)$$

$$\bigtriangledown_\theta J(\theta) = E[\bigtriangledown_\theta \text{log} p_\theta(a|D) R(\text{SUM}; G)] \quad (9)$$

$$Q(D) = \prod_{j=1}^{M} \left(\frac{\epsilon}{N-j+1} + \frac{(1-\epsilon)\mu_{a_j}(D)}{z(D) - \sum_{k=1}^{j-1} \mu_{a_k}(D)}\right) \quad (10)$$

However, the exact document distribution is unknown and we cannot evaluate the expected value in Eq. (9). So we use sampling to estimate it instead. Given a document-summary pair $(D, G)$, our HER samples $B$ summaries induced by $a^1, \ldots, a^B$ from $p_\theta(\cdot|D)$ and obtain all the gradients, then the average value is considered as the estimation. As sample-based gradient estimate may have high variance, we use a baseline for variance reduction. The gradient of the objective function is finally represented as,

$$\bigtriangledown_\theta J(\theta) \approx \frac{1}{B} \sum_{b=1}^{B} \bigtriangledown_\theta \text{log} p_\theta(a^b|D)(R(a^b, G) - \bar{r}) \quad (11)$$

where we choose self-critical reinforcement learning to acquire the baseline $\bar{r}$ following Ranzato et al. (2015); Rennie et al. (2017); Paulus et al. (2017); Dong et al. (2018) computed by greedy encoding $\bar{r} = R(a_{greedy}; G)$. More concretely, $a_{greedy} = \text{argmax} p_\theta(a|D)$ and this baseline satisfies that the probability of a sampled sequence would be increased when the summary it induces is better than what is obtained by greedy decoding. The procedure of HER is shown in Algorithm 1.

**Algorithm 1** HER: training

**Input** $\{(D_i, G_i)\}$, a dataset of document-summary pairs
**Output** $\theta$, the updated parameters in $p_\theta$

**for** each document-summary pair $(D, G)$ **do**
  $S_{1:N}, \bar{S}, L_{1:K} = \text{Encoder}(D)$ ▷ Rough Reading
  $\mu_{1:N} = \text{Decoder}(S_{1:N}, \bar{S}, L_{1:K})$ ▷ Careful Reading
  **for** each trail $b = 1, \ldots, B$ **do**
    Initialize the action set $A = \{1, \ldots, N\}$
    **for** time step $t = 1, \ldots, N$ **do**
      Sample $u \sim U(0, 1)$
      **if** $u < \epsilon$ **then**
        Uniformly sample $a_t$ from $A$
      **else**
        $a_t \sim \text{Categorical}(\boldsymbol{\mu}_A)$
      Get termination flag by Eq.(6)
      Let $M = t$
      **if** Done **then**
        Break the inner loop
      **else**
        Update the action set $A = A \setminus \{a_t\}$
    Generate summary $\text{SUM}_b = (S_{a_1}, \ldots, S_{a_M})$
    $l_b = \frac{1}{M} \sum_{t=1}^{M} \text{log}(\frac{\epsilon}{N-t+1} + \frac{(1-\epsilon)\mu_{a_t}}{\sum_j \mu_j - \sum_{k=1}^{t-1} \mu_{a_k}})$
    Compute reward $r_b = R(\text{SUM}_b; G)$
  $\bar{r} = R(\text{SUM}_{greedy}; G)$
  $l = \frac{1}{B} \sum_{b=1}^{B} l_b(\bar{r} - r_b)$ ▷ Surrogate loss
  $\theta \leftarrow \theta - \lambda \bigtriangledown_\theta l$ ▷ Update

## 3 Experiment Settings

In this section we present our experimental setup for evaluating the performance of the proposed HER, including the datasets, evaluation protocol, baselines and implementation details.

**Datasets:** We evaluate our models on three datasets: the CNN, the DailyMail and the combined CNN/DailyMail (Hermann et al., 2015; Nallapati et al., 2016). We also use the standard splits of Hermann et al. (2015) for training, validation, and test ($90,266/1,220/1,093$ documents for CNN and $196,961/12,148/10,397$ for DailyMail) with the same setting in See et al. (2017).

**Evaluation:** We evaluate summarization quality using $F_1$ ROUGE (Lin, 2004) including unigram and bigram overlap (ROUGE-1 and ROUGE-2) to assess informativeness and the longest common subsequence (ROUGE-L) to assess fluency with the reference summaries. We obtain ROUGE scores using a faster python im-

plementation[1] for training and evaluation, and the standard pyrouge package[2] for test following Dong et al. (2018).

**Baselines:** We compare our proposed HER against four kinds of extractive methods: (1) Lead-3 model simply selects the first three sentences. (2) NN-SE (Cheng and Lapata, 2016) and SummaRuNNer (Nallapati et al., 2017) are sequence labeling task and trained with cross-entropy loss. (3) Refresh (Narayan et al., 2018), DQN (Yao et al., 2018) and RNES (Wu and Hu, 2018) extract summary via reinforcement learning. (4) BANDITSUM (Dong et al., 2018) considers the extractive summarization as a contextual bandit but fails to simulate human reading recognition process.

**Implementation Details:** We initialize word embeddings with 100-dimension Glove embeddings (Pennington et al., 2014). In rough reading, the encoder is hierarchical and each layer is a two-stacked BiLSTM with a hidden size of 200. Therefore, sentence vectors and the document representation $\bar{S}$ have a dimension of 400. For the variant CNN, we adopt filter windows $H$ in $\{1, 2, 3\}$ with 100 feature maps each and generate $K = 3$ local representations for each document. In careful reading, we set $\epsilon = 0.1$ for bandit policy. We also bound the minimum and maximum number of selected sentence to be 1 and 10 for termination mechanism. During training, we use the optimizer Adam (Kingma and Ba, 2014) with a learning rate of $10^{-5}$, beta parameters as $(0, 0.999)$ and a weight decay of $10^{-6}$ to maximize the objective function following Dong et al. (2018). We employ gradient clipping of 1 for regularization and sample $B = 20$ times for each document. We train our model within two epochs. Note that we choose the whole document as the final summary if the document length is less than 3 sentences as the local features cannot be obtained through the CNN-based network.

## 4 Experimental Results

### 4.1 Quantitative Analysis

We first report the ROUGE metrics on the combined CNN/DailyMail test sets in Table 1 and the separate results in Table 2. We can get several observations from these two tables.

[1] https://github.com/pltrdy/rouge

[2] Pyrouge is a Python package. We compute all ROUGE scores with parameters "-a -c 95 -m -n 4 -w 1.2." Refer to https://pypi.python.org/pypi/pyrouge/0.1.3

| Model | ROUGE | | |
|---|---|---|---|
| | R1 | R2 | RL |
| Lead-3 | 40.0 | 17.5 | 36.2 |
| SummaRuNNer | 39.6 | 16.2 | 35.3 |
| DQN | 39.4 | 16.1 | 35.6 |
| Refresh | 40.0 | 18.2 | 36.6 |
| RNES | 41.3 | **18.9** | 37.6 |
| BANDITSUM | 41.5 | 18.7 | 37.6 |
| **HER** | **42.3** | **18.9** | **37.9** |

Table 1: Results on the combined CNN/DailyMail test set. We report F1 scores of ROUGE-1 (R1), ROUGE-2 (R2), and ROUGE-L (RL). The result of Lead-3 is taken from Dong et al. (2018).

| Model | CNN | | | DailyMail | | |
|---|---|---|---|---|---|---|
| | R1 | R2 | RL | R1 | R2 | RL |
| Lead-3 | 28.8 | 11.0 | 25.5 | 41.2 | 18.2 | 37.3 |
| NN-SE | 28.4 | 10.0 | 25.0 | 36.2 | 15.2 | 32.9 |
| Refresh | 30.4 | 11.7 | 26.9 | 41.0 | 18.8 | 37.7 |
| BANDITSUM | **30.7** | **11.6** | 27.4 | 42.1 | 18.9 | 38.3 |
| **HER** | **30.7** | 11.5 | **27.5** | **42.7** | **19.0** | **38.5** |

Table 2: Results of the test sets on the CNN and DailyMail datasets separately.

Firstly, our model generally performs the best and even surpasses 42 on ROUGE-1 score on the combined CNN/DailyMail dataset. It also shows better results on the separate datasets. We argue that global and local features from rough reading can help extract summaries by capturing deep contextual relations, and the designed structure in careful reading makes it more flexible in selecting sentence sets. Hence a two-stage framework based on the human's reading cognition is more appropriate for extractive summarization.

Secondly, directly optimizing the evaluation metric by combining cross-entropy loss with rewards may improve the extractive results. RL-based methods, Refresh (Narayan et al., 2018) and RNES (Wu and Hu, 2018), perform better than the sequence labeling methods like SummaRuNNer (Nallapati et al., 2017). BANDITSUM (Dong et al., 2018) generally performs better than the other baselines, and it reports that framing the extractive summarization based on contextual bandit is more suitable than sequential labeling setting and also has more search space than other RL-based methods (Narayan et al., 2018; Yao et al., 2018; Wu and Hu, 2018).

### 4.2 Ablation Test

Next, we conduct ablation test by removing the modules of the proposed HER step by step. Firstly, we replace the automatic termination mechanism with a fixed extracting strategy that always selects

| Model | ROUGE | | |
|---|---|---|---|
| | R1 | R2 | RL |
| **HER** | **42.3** | **18.9** | **37.9** |
| HER-3 | 42.0 | 18.5 | 37.6 |
| HER-3 w/o policy | 41.7 | 18.3 | 37.1 |
| HER-3 w/o policy&L | 41.2 | 18.4 | 37.0 |
| HER-3 w/o policy&F | 40.6 | 18.2 | 36.9 |

Table 3: The results of ablation test on the test split of the combined CNN/DailyMail dataset. L and F are short for local net and rough reading.

three sentences for every document and we present the model as HER-3. Based on HER-3, we also remove bandit policy, local net, general net gradually, and denote them as HER-3 w/o policy, HER-3 w/o policy & local net and HER-3 w/o policy & rough reading individually. The results are reported in Table 3 and it proves the effectiveness of each proposed module. Firstly, HER constructed with an automatic termination mechanism is more flexible and reliable in extracting various numbers of sentences varying different documents. Secondly, HER use $\epsilon$-greedy to select sentences in order to raise the exploration chances on discovering important but easily ignored information. Thirdly, general cognition from rough reading process is useful in extractive summarizarion.

### 4.3 A Closer Look

To verify whether our proposed HER can simulate human beings' reading cognitive process, and whether such simulation are inherently helpful on extractive summarization, we conduct extensive evaluations and try to answer the following three questions.

**(1) Can CNN-based network extract local features of different paragraphs?**

In Figure 3, we report the distribution of selected sentences' positions on our proposed model HER, BANDITSUM and HER w/o Local Net. We test each model at 10k, 50k, 100k training steps. It shows that all the three models can focus on different parts of the context to form summary at first and BANDITSUM performs the best after training 10k steps. However, with training steps growing, BANDITSUM and HER w/o Local begin to prefer earlier sentences. HER, on the other hand, can focus on various paragraphs and extract information from different parts of the texts with constant training. The contextual bandit (CB) based frameworks seems to be able to attend on various parts of the contexts to some degree in the beginning.

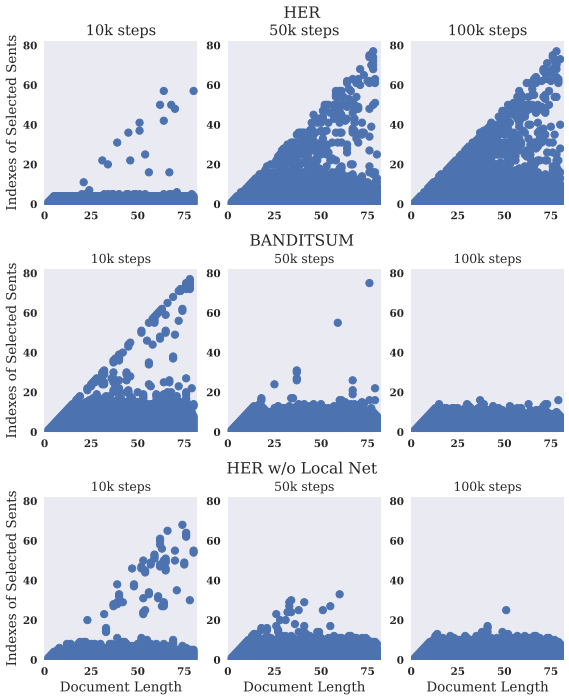


Figure 3: The statistics of model HER, BANDITSUM (Dong et al., 2018), HER w/o Local Net on the selected sentences' indexes varying different document lengths. This is reported on the documents the lengths of which are all less than 80 on the test split.

However, with constant training, both BANDITSUM and HER w/o Local start to focus on earlier sentences due to the nature that sentences similar to the main idea usually lie on the head of the text. As our proposed HER is equipped with a variant CNN to extract local features, our model can focus on gist of paragraphs rather than only the first several sentences. It also encourages the exploration on extracting information from various positions more flexibly.

**(2) Can the proposed bandit policy discover low-score but easily ignored information?**

To answer this question, we demonstrate a detailed case on sentence selection in Figure 4. We observe that although the fourth sentence has a high affinity, it should not be included in the summary since its meaning is close to the third sentence which has already been extracted. Instead, the 13th sentence is supposed to be selected though it has low affinity. Since our HER adopts the $\epsilon$-greedy policy, it can explore such sentence and extract it out correctly.

**(3) Can HER extract varied but proper numbers of sentences?**

We answer this question by drawing the fre-

| Index | Sentence | Affinity | HER | HER w/o policy |
|---|---|---|---|---|
| 2 | Two spotted leopards, two Macaque monkeys and a brown bear will be returned to Marian Thompson… | 0.873 | yes | yes |
| 3 | He set off a wide scare in October when he released 50 potentially dangerous animals from his farm before shooting himself. | 0.872 | yes | yes |
| 4 | Of the 50 animals Thompson released, 48 were killed by law enforcement, while two primates were killed by the other animals, zoo officials said. | 0.767 | **no** | **yes** |
| 13 | State officials have no legal power to inspect the cages before the animals are returned… | 0.297 | **yes** | **no** |

Figure 4: A case on sentence selection of HER and HER w/o policy. The article is from CNN dataset. The highlighted indices indicate the corresponding sentences which should be extracted as summary.

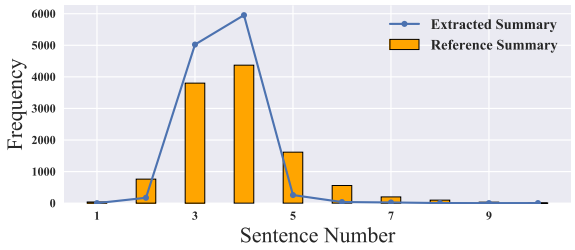


Figure 5: The statistics on extracted sentence number of our model. Frequency is the number of documents.

quency distribution of extracted sentence numbers by our model on the test set of combined CNN/DailyMail, and Figure 5 exhibits the results. We observe that the frequency distribution of extracted sentence number is basically similar to that of the gold-standard summary. Compared with BANDITSUM which extracts fixed number of sentences, our model shows more flexibility and extensibility on extractive summarization.

### 4.4 Human Evaluation

Lastly, we conduct a qualitative evaluation. Following Wu and Hu (2018), we randomly sample 50 documents from the test set on the combined CNN/DailyMail dataset and ask three volunteers to evaluate the summaries extracted by HER w/o $\text{Dec}_2$, HER w/o Local Net, BANDITSUM and HER, respectively. HER w/o $\text{Dec}_2$ only uses Eq. (3) to compute sentence affinities without inhibiting redundant sentences. For each document-summary pair, they are asked to rank the output of each system on three aspects, namely overall quality, coverage and non-redundancy. Notice that the best one will be marked rank 1 and so on, and two system would be ranked the same if their extracted summaries are identical. We report the average results in Table 4 and it shows that our HER is leading than BANDITSUM on overall quality and coverage. Additionally, HER w/o $\text{Dec}_2$ performs the worst on non-redundancy as it does not specialize these unselected sentences which are similar to already extracted ones. Furthermore, HER w/o Local Net takes on bad performance on coverage because the local features can focus on paragraphical messages and help to refine thorough information.

| Model | Overall | Coverage | Non-Redundancy |
|---|---|---|---|
| HER w/o $\text{Dec}_2$ | 2.88 | 2.81 | 2.81 |
| HER w/o L | 3.02 | 2.96 | 2.75 |
| BANDITSUM | 2.06 | 2.07 | **1.91** |
| HER | **1.81** | **1.75** | 1.97 |

Table 4: Average rank of human evaluation in terms of overall performance, coverage, and non-redundancy. L is short for local Net. Lower score is better.

## 5 Related Work

**Extractive Text Summarization** Researchers have developed many statistical methods for automatic extractive summarization. Traditional methods learn to score each sentence dependently (Erkan and Radev, 2004; Mihalcea and Tarau, 2004; Wong et al., 2008). Recently neural network based extractive methods (Cheng and Lapata, 2016; Nallapati et al., 2017; Feng et al., 2018; Shi et al., 2018) usually consider extractive summarization as sequence labeling tasks and aim to minimize the cross-entropy objective function. Narayan et al. (2017) utilizes side information to help sentence classifier while Yasunaga et al. (2017) computes the salience of each sentence for selection with graph convolutional networks. In addition, reinforcement learning based methods (Wu and Hu, 2018; Narayan et al., 2018; Yao et al., 2018) have been proposed to directly optimize the evaluation metric ROUGE by combining cross-entropy loss with rewards from policy gradient reinforcement learning. Dong et al. (2018) considered extractive summarization as a contextual bandit and it performs well especially when good summary sentences appear late in the source document. Recently, Nallapati et al. (2017); Chen and Bansal (2018); Hsu et al. (2018) propose unified models and combine the advantages of both extractive and abstractive methods.

**Human Reading-inspired Strategy in NLP** Recently, several pioneer researches began to study how to adapt human reading cognition process, usually including pre-reading, reading and post-reading (Avery and Graves, 1997; Saricoban,

2002; Toprak and Almacıoğlu, 2009; Pressley and Afflerbach, 2012), into various NLP-related applications. For example, Li et al. (2018) solved document-based question answering and by simulating human being's reading strategy. Luo et al. (2018, 2019) utilized the prior knowledge of human reading to solve sub-tasks in sentiment analysis. Song et al. (2017, 2018) enhanced word embeddings in a similar way. Yang et al. (2019) applied it for abstractive summarization, Zheng et al. (2019) simulated human behavior for reading comprehension, and Lei et al. (2019) utilized human-like semantic cognition for aspect-level sentiment classification. In this paper, we attempt to perform extractive summarization under the inspiration of human reading recognition.

## 6 Conclusion

Inspired by the reading cognition of human beings, we propose HER, a two-stage method, to mimic how people extract summaries. The whole learning process is formulated as a contextual bandit trained with policy gradient reinforcement learning. In rough reading, two neural networks are taken to encode coarse-grained information. In careful reading, repeatedly reading are conducted to select fine-grained sentences as summary. Experiments on two real-world datasets demonstrate that our proposed model can significantly outperform the state-of-the-art extractive methods on summary quality, coverage and non-redundancy.

## Acknowledgements

This work is partially supported by the National Key Research and Development Program of China under Grant No. 2017YFB1002104, the National Natural Science Foundation of China under Grant No. 91846113, U1811461, 61602438, 61573335, CCF-Tencent RhinoBird Young Faculty Open Research Fund No. RAGR20180111. This work is also funded in part by Ant Financial through the Ant Financial Science Funds for Security Research. Xiang Ao is also supported by Youth Innovation Promotion Association CAS. This work was also partially supported by SIAT Innovation Program for Excellent Young Researchers (Grant No. Y8G027). Min Yang was sponsored by CCF-Tencent Open Research Fund.

## References

Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. 2002. The nonstochastic multi-armed bandit problem. *SIAM journal on computing*.

Patricia G Avery and Michael F Graves. 1997. Scaffolding young learners' reading of social studies texts. *Social Studies and the Young Learner*, 9(4):10–14.

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 675–686.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Yue Dong, Yikang Shen, Eric Crawford, Herke van Hoof, and Jackie Chi Kit Cheung. 2018. Banditsum: Extractive summarization as a contextual bandit. In *EMNLP*.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research*, 22:457–479.

Chong Feng, Fei Cai, Honghui Chen, and Maarten de Rijke. 2018. Attentive encoder-based extractive text summarization. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in neural information processing systems*.

Wan-Ting Hsu, Chieh-Kai Lin, Ming-Ying Lee, Kerui Min, Jing Tang, and Min Sun. 2018. A unified model for extractive and abstractive summarization using inconsistency loss. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization.

John Langford and Tong Zhang. 2007. The epoch-greedy algorithm for contextual multi-armed bandits. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, pages 817–824. Citeseer.

John Langford and Tong Zhang. 2008. The epoch-greedy algorithm for multi-armed bandits with side information. In *Advances in neural information processing systems*, pages 817–824.

Zeyang Lei, Yujiu Yang, Min Yang, Wei Zhao, Jun Guo, and Yi Liu. 2019. A human-like semantic cognition network for aspect-level sentiment classification. In *AAAI*.

Lihong Li, Wei Chu, John Langford, and Robert E Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670. ACM.

Weikang Li, Wei Li, and Yunfang Wu. 2018. A unified model for document-based question answering based on human-like reading strategy. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Miaofeng Liu, Yan Song, Hongbin Zou, and Tong Zhang. 2019. Reinforced training data selection for domain adaptation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1957–1968, Florence, Italy.

Ling Luo, Xiang Ao, Feiyang Pan, Jin Wang, Tong Zhao, Ningzi Yu, and Qing He. 2018. Beyond polarity: Interpretable financial sentiment analysis with hierarchical query-driven attention. In *IJCAI*, pages 4244–4250.

Ling Luo, Xiang Ao, Yan Song, Jinyao Li, Xiaopeng Yang, Qing He, and Dong Yu. 2019. Unsupervised neural aspect extraction with sememes. In *IJCAI*.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *AAAI*.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Ranking sentences for extractive summarization with reinforcement learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Shashi Narayan, Nikos Papasarantopoulos, Shay B Cohen, and Mirella Lapata. 2017. Neural extractive summarization with side information. *arXiv preprint arXiv:1704.04530*.

Ani Nenkova, Kathleen McKeown, et al. 2011. Automatic summarization. *Foundations and Trends® in Information Retrieval*, 5(2–3):103–233.

Feiyang Pan, Qingpeng Cai, Pingzhong Tang, Fuzhen Zhuang, and Qing He. 2019a. Policy gradients for contextual recommendations. In *The World Wide Web Conference*, pages 1421–1431. ACM.

Feiyang Pan, Qingpeng Cai, An-Xiang Zeng, Chun-Xiang Pan, Qing Da, Hualin He, Qing He, and Pingzhong Tang. 2019b. Policy optimization with model-based explorations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4675–4682.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Michael Pressley and Peter Afflerbach. 2012. *Verbal protocols of reading: The nature of constructively responsive reading*. Routledge.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Arif Saricoban. 2002. Reading strategies of successful readers through the three phase approach. *The Reading Matrix*, 2(3).

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083.

Jiaxin Shi, Chen Liang, Lei Hou, Juanzi Li, Zhiyuan Liu, and Hanwang Zhang. 2018. Deepchannel: Salience estimation by contrastive learning for extractive document summarization. *arXiv preprint arXiv:1811.02394*.

Yan Song, Prescott Klassen, Fei Xia, and Chunyu Kit. 2012. Entropy-based training data selection for domain adaptation. In *Proceedings of COLING 2012*, pages 1191–1200, Mumbai, India.

Yan Song, Chia-Jung Lee, and Fei Xia. 2017. Learning word representations with regularization from prior knowledge. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 143–152, Vancouver, Canada.

Yan Song, Shuming Shi, and Jing Li. 2018. Joint learning embeddings for chinese words and their components via ladder structured networks. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 4375–4381.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*.

Elif Leyla Toprak and Gamze Almacıoğlu. 2009. Three reading phases and their applications in the teaching of english as a foreign language in reading classes with young learners. *Journal of language and Linguistic Studies*, 5(1):20–36.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Kam-Fai Wong, Mingli Wu, and Wenjie Li. 2008. Extractive summarization using supervised and semi-supervised learning. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 985–992. Association for Computational Linguistics.

Yuxiang Wu and Baotian Hu. 2018. Learning to extract coherent summary via deep reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Min Yang, Qiang Qu, Wenting Tu, Ying Shen, Zhou Zhao, and Xiaojun Chen. 2019. Exploring human-like reading strategy for abstractive text summarization. In *AAAI*.

Kaichun Yao, Libo Zhang, Tiejian Luo, and Yanjun Wu. 2018. Deep reinforcement learning for extractive document summarization. *Neurocomputing*.

Michihiro Yasunaga, Rui Zhang, Kshitijh Meelu, Ayush Pareek, Krishnan Srinivasan, and Dragomir Radev. 2017. Graph-based neural multi-document summarization. In *Proceedings of the 21st Conference on Computational Natural Language Learning*.

Yukun Zheng, Jiaxin Mao, Yiqun Liu, Zixin Ye, Min Zhang, and Shaoping Ma. 2019. Human behavior inspired machine reading comprehension. In *SIGIR*.

# Contrastive Attention Mechanism for Abstractive Sentence Summarization

**Xiangyu Duan[1,2]*, Hongfei Yu[2]*, Mingming Yin[2], Min Zhang[1,2], Weihua Luo[3], Yue Zhang[4]**

[1] *Institute of Artificial Intelligence, Soochow University, Suzhou, China*
[2] *School of Computer Science and Technology, Soochow University, Suzhou, China*
[3] *Alibaba DAMO Academy, Hangzhou, China*
[4] *School of Engineering, Westlake University, China*

xiangyuduan@suda.edu.cn; {hfyu,mmyin}@stu.suda.edu.cn; minzhang@suda.edu.cn
weihua.luowh@alibaba-inc.com; yue.zhang@wias.org.cn

## Abstract

We propose a contrastive attention mechanism to extend the sequence-to-sequence framework for abstractive sentence summarization task, which aims to generate a brief summary of a given source sentence. The proposed contrastive attention mechanism accommodates two categories of attention: one is the conventional attention that attends to relevant parts of the source sentence, the other is the opponent attention that attends to irrelevant or less relevant parts of the source sentence. Both attentions are trained in an opposite way so that the contribution from the conventional attention is encouraged and the contribution from the opponent attention is discouraged through a novel softmax and softmin functionality. Experiments on benchmark datasets show that, the proposed contrastive attention mechanism is more focused on the relevant parts for the summary than the conventional attention mechanism, and greatly advances the state-of-the-art performance on the abstractive sentence summarization task. We release the code at https://github.com/travel-go/Abstractive-Text-Summarization.

## 1 Introduction

Abstractive sentence summarization aims at generating concise and informative summaries based on the core meaning of source sentences. Previous endeavors tackle the problem through either rule-based methods (Dorr et al., 2003) or statistical models trained on relatively small scale training corpora (Banko et al., 2000). Following its successful applications on machine translation (Sutskever et al., 2014; Bahdanau et al., 2015), the sequence-to-sequence framework is also applied on the abstractive sentence summarization task using large-scale sentence summary corpora (Rush et al., 2015; Chopra et al., 2016; Nallapati et al., 2016), obtaining better performance compared to the traditional methods.

One central component in state-of-the-art sequence to sequence models is the use of attention for building connections between the source sequence and target words, so that a more informed decision can be made for generating a target word by considering the most relevant parts of the source sequence (Bahdanau et al., 2015; Vaswani et al., 2017). For abstractive sentence summarization, such attention mechanisms can be useful for selecting the most salient words for a short summary, while filtering the negative influence of redundant parts.

We consider improving abstractive summarization quality by enhancing target-to-source attention. In particular, a contrastive mechanism is taken, by encouraging the contribution from the conventional attention that attends to relevant parts of the source sentence, while at the same time penalizing the contribution from an opponent attention that attends to irrelevant or less relevant parts. Contrastive attention was first proposed in computer vision (Song et al., 2018a), which is used for person re-identification by attending to person and background regions contrastively. To our knowledge, we are the first to use contrastive attention for NLP and deploy it in the sequence-to-sequence framework.

In particular, we take Transformer (Vaswani et al., 2017) as the baseline summarization model, and enhance it with a proponent attention module and an opponent attention module. The former acts as the conventional attention mechanism, while the latter can be regarded as a dual module to the former, with similar weight calculation structure, but using a novel softmin function to discourage contributions from irrelevant or less relevant words.

To our knowledge, we are the first to investigate

* Equal contribution.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3044–3053, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

Transformer as a sequence to sequence summarizer. Results on three benchmark datasets show that it gives highly competitive accuracies compared with RNN and CNN alternatives. When equipped with the proposed contrastive attention mechanism, our Transformer model achieves the best reported results on all data. The visualization of attentions shows that through using the contrastive attention mechanism, our attention is more focused on relevant parts than the baseline. We release our code at XXX.

## 2 Related Work

Automatic summarization has been investigated in two main paradigms: the extractive method and the abstractive method. The former extracts important pieces of source document and concatenates them sequentially (Jing and McKeown, 2000; Knight and Marcu, 2000; Neto et al., 2002), while the latter grasps the core meaning of the source text and re-state it in short text as abstractive summary (Banko et al., 2000; Rush et al., 2015). In this paper, we focus on abstractive summarization, and especially on abstractive sentence summarization.

Previous work deals with the abstractive sentence summarization task by using either rule based methods (Dorr et al., 2003), or statistical methods utilizing a source-summary parallel corpus to train a machine translation model (Banko et al., 2000), or a syntax based transduction model (Cohn and Lapata, 2008; Woodsend et al., 2010).

In recent years, sequence-to-sequence neural framework becomes predominant on this task by encoding long source texts and decoding into short summaries together with the attention mechanism. RNN is the most commonly adopted and extensively explored architecture (Chopra et al., 2016; Nallapati et al., 2016; Li et al., 2017). A CNN-based architecture is recently employed by Gehring *et al.* (2017) using ConvS2S, which applies CNN on both encoder and decoder. Later, Wang *et al.* (2018) build upon ConvS2S with topic words embedding and encoding, and train the system with reinforcement learning.

The most related work to our contrastive attention mechanism is in the field of computer vision. Song *et al.* (2018a) first propose the contrastive attention mechanism for person re-identification. In their work, based on a pre-provided person and background segmentation, the two regions are contrastively attended so that they can be easily discriminated. In comparison, we apply the contrastive attention mechanism for sentence level summarization by contrastively attending to relevant parts and irrelevant or less relevant parts. Furthermore, we propose a novel softmax softmin functionality to train the attention mechanism, which is different to Song *et al.* (2018a), who use mean squared error loss for attention training.

Other explorations with respect to the characteristics of the abstractive summarization task include copying mechanism that copies words from source sequences for composing summaries (Gu et al., 2016; Gulcehre et al., 2016; Song et al., 2018b), the selection mechanism that elaborately selects important parts of source sentences (Zhou et al., 2017; Lin et al., 2018), the distraction mechanism that avoids repeated attention on the same area (Chen et al., 2016), and the sequence level training that avoids exposure bias in teacher forcing methods (Ayana et al., 2016; Li et al., 2018; Edunov et al., 2018). Such methods are built on conventional attention, and are orthogonal to our proposed contrastive attention mechanism.

## 3 Approach

We use two categories of attention for summary generation. One is the conventional attention that attends to relevant parts of source sentence, the other is the opponent attention that contrarily attends to irrelevant or less relevant parts. Both categories of attention output probability distributions over summary words, which are jointly optimized by encouraging the contribution from the conventional attention and discouraging the contribution from the opponent attention.

Figure 1 illustrates the overall networks. We use Transformer architecture as our basis, upon which we build the contrastive attention mechanism. The left part is the original Transformer. We derive the opponent attention from the conventional attention which is the encoder-decoder attention of the original Transformer, and stack several layers on top of the opponent attention as shown in the right part of Figure 1. Both parts contribute to the summary generation by producing probability distributions over the target vocabulary, respectively. The left part outputs **the conventional probability** based on the conventional attention as the original Transformer does, while the right part outputs **the opponent probability** based on the opponent

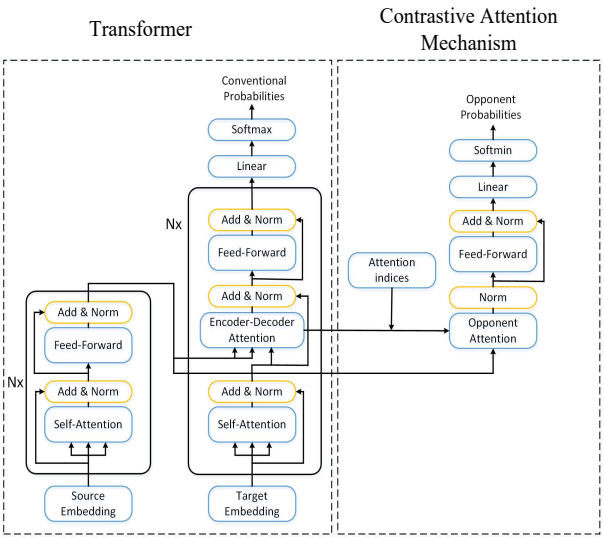


Figure 1: Overall networks. The left part is the original Transformer. The right part that takes the opponent attention as bottom layer fulfils the contrastive attention mechanism.

attention. The two probabilities in Figure 1 are jointly optimized in a novel way as explained in Section 3.3.

### 3.1 Transformer for Abstractive Sentence Summarization

Transformer is an attention network based sequence-to-sequence architecture (Vaswani et al., 2017), which encodes the source text into hidden vectors and decodes into the target text based on the source side information and the target generation history. In comparison to the RNN based architecture and the CNN based architecture, both the encoder and the decoder of Transformer adopt attention as main function.

Let $X$ and $Y$ denote the source sentence and its summary, respectively. Transformer is trained to maximize the probability of $Y$ given $X$: $\prod_i P_c(y_i|y_1^{i-1}, X)$, where $P_c(y_i|y_1^{i-1}, X)$ is the conventional probability of the current summary word $y_i$ given the source sentence and the summary generation history. $P_c$ is computed based on the attention mechanism and the stacked deep layers as shown in the left part of Figure 1.

**Attention Mechanism**

Scaled dot-product attention is applied in Transformer:

$$\text{attention}(Q, K, V) = \text{softmax}(\frac{QK^{\text{T}}}{\sqrt{d_k}})V \quad (1)$$

where $Q, K, V$ denotes query vector, key vectors, and value vectors, respectively. $d_k$ denotes the dimension of one vector of $K$. Softmax function outputs the attention weights distributed over $V$. $\text{attention}(Q, K, V)$ is a vector of weighted sum of elements of $V$, and represents current context information.

We focus on the encoder-decoder attention, which builds the connection between source and target by informing the decoder which area of the source text should be attended to. Specifically, in the encoder-decoder attention, $Q$ is the single vector coming from the current position of the decoder, $K$ and $V$ are the same sequence of vectors that are the outcomes of the encoder at all source positions. Softmax function distributes the **attention weights** over the source positions.

The attentions in Transformer adopts the multi-head implementation, in which each head computes attention as Equation (1) but with smaller $Q, K, V$ whose dimension is $1/h$ times of their original dimension respectively. The attentions from $h$ heads are concatenated together and linearly projected to compose the final attention. In this way, multi-head attention provides a multi-view of attention behavior beneficial for the final performance.

**Deep Layers**

The "N×" plates in Figure 1 stands for the stacked N identical layers. On the source side, each layer of the stacked N layers contains two sublayers: the self-attention mechanism, and the fully connected feed-forward network. Each sublayer employs residual connection that adds input to outcome of sublayer, then layer normalization is employed on the outcome of the residual connection.

On the target summary side, each layer contains an additional sublayer of the encoder-decoder attention between the self-attention sublayer and the feed-forward sublayer. At the top of the decoder, the softmax layer is applied to convert the decoder output to summary word generation probabilities.

### 3.2 Contrastive Attention Mechanism

#### 3.2.1 Opponent Attention

As illustrated in Figure 1, the opponent attention is derived from the conventional encoder-decoder attention. Since the multi-head attention is employed in Transformer, there are N×$h$ heads in total in the conventional encoder-decoder attention, where N denotes the number of layers, $h$ denotes

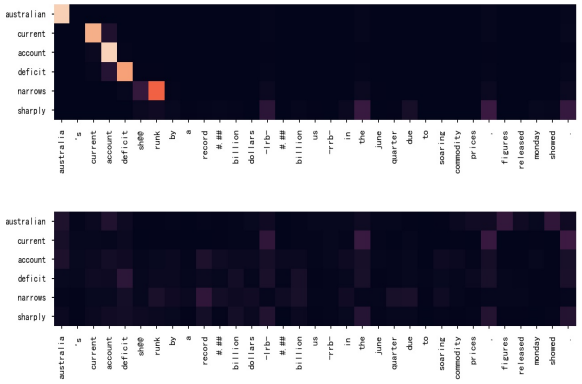


Figure 2: Heatmaps of two sampled heads from the conventional encoder-decoder attention. (a) is of the fifth head of the third layer, and (b) is of the fifth head of the first layer.

the number of heads in each layer. These heads exhibit diverse attention behaviors, posing a challenge of determining which head to derive the opponent attention, so that it attends to irrelevant or less relevant parts.

Figure 2 illustrates the attention weights of two sampled heads. The attention weights in (a) well reflect the word level relevant relation between the source sentence and the target summary, while attention weights in (b) do not. We find that such behavior characteristic of each head is fixed. For example, head (a) always exhibits the relevant relation across different sentences and different runs. Based on depicting heatmaps of all heads for a few sentences, we choose the head that corresponds well to the relevant relation between source and target to derive the opponent attention [1].

Specifically, let $\alpha_c$ denote the conventional encoder-decoder attention weights of the head which is used for deriving the opponent attention:

$$\alpha_c = \mathrm{softmax}(\frac{qk^{\mathrm{T}}}{\sqrt{d_k}}) \quad (2)$$

where $q$ and $k$ are from the head same to that of $\alpha_c$. Let $\alpha_o$ denote the opponent attention weights. It is obtained through the opponent function applied on $\alpha_c$ followed by the softmax function:

$$\alpha_o = \mathrm{softmax}(\mathrm{opponent}(\alpha_c)) \quad (3)$$

The opponent function in Equation (3) performs a masking operation, which finds the maximum weight in $\alpha_c$, and replaces it with the negative infinity value, so that the softmax function outputs zero given the negative infinity value input. Then the maximum weight in $\alpha_c$ is set zero in $\alpha_o$ after the opponent and softmax functions. In this way, the most relevant part of the source sequence, which receives maximum attention in the conventional attention weights $\alpha_c$, is masked and neglected in $\alpha_o$. Instead, the remaining less relevant or irrelevant parts are extracted into $\alpha_o$ for the following contrastive training and decoding.

We also tried other methods to calculate the opponent attention weights, such as $\alpha_o = \mathrm{softmax}(1 - \alpha_c)$ (Song et al., 2018a) [2] or $\alpha_o = \mathrm{softmax}(1/\alpha_c)$, which aims to make $\alpha_o$ contrary to $\alpha_c$, but they underperform the masking opponent function on all benchmark datasets. So we present only the masking opponent in the following sections.

After $\alpha_o$ is obtained via Equation (3), the opponent attention is: $\mathrm{attention}_o = \alpha_o v$, where $v$ is from the head same to that of $q$ and $k$ in computing $\alpha_c$.

Compared to the conventional attention $\mathrm{attention}_c$, which summarizes current relevant context, $\mathrm{attention}_o$ summarizes current irrelevant or less relevant context. They constitute a contrastive pair, and contribute together for the final summary word generation.

### 3.2.2 Opponent Probability

The opponent probability $P_o(y_i|y_1^{i-1}, X)$ is computed by stacking several layers on top of $\mathrm{attention}_o$, and a softmin layer in the end as shown in the right part of Figure (1). In particular,

$$z_1 = \mathrm{LayerNorm}(\mathrm{attention}_o) \quad (4)$$
$$z_2 = \mathrm{FeedForward}(z_1) \quad (5)$$
$$z_3 = \mathrm{LayerNorm}(z_1 + z_2) \quad (6)$$
$$P_o(y_i|y_1^{i-1}, X) = \mathrm{softmin}(Wz_3) \quad (7)$$

where $W$ is the matrix of the linear projection sublayer.

$\mathrm{attention}_o$ contributes to $P_o$ via Equation (4-7) step by step. The LayerNorm and FeedForward layers with residual connection is similar to the

[1] Given manual alignments between source and target of sampled sentence-summary pairs, we select the head that has the lowest alignment error rate (AER) of its attention weights.

[2] Song *et al.* (2018a) directly let $\alpha_o = 1 - \alpha_c$ in extracting background features for person re-identification in computer vision. We have to add softmax function since the attention weights must be normalized to one in sequence-to-sequence framework.

original Transformer, while a novel softmin function is introduced in the end to invert the contribution from $\text{attention}_o$:

$$\text{softmin}(v_i) = \frac{e^{(-v_i)}}{\sum^j e^{(-v_j)}} \quad (8)$$

where $v = Wz_3$, i.e., the input vector to the softmin function in Equation (7). Softmin normalizes $v$ so that scores of all words in the summary vocabulary sum to one. We can see that the bigger the $v_i$, the smaller the $P_{o,i}$ is.

Softmin functions contrarily to softmax. As a result, when we try to maximize $P_o(y_i = y|y_1^{i-1}, X)$, where $y$ is the gold summary word, we effectively search for an appropriate $\text{attention}_o$ to generate the lowest $v_g$, where $g$ is the index of $y$ in $v$. It means that the more irrelevant is $\text{attention}_o$ to the summary, the lower the $v_g$ can be obtained, resulting in higher $P_o$.

### 3.3 Training and Decoding

During training, we jointly maximize the conventional probability $P_c$ and the opponent probability $P_o$:

$$\mathcal{J} = \log(P_c(y_i|y_1^{i-1}, X) + \lambda \log(P_o(y_i|y_1^{i-1}, X) \quad (9)$$

where $\lambda$ is the balanced weight. The conventional probability is computed as the original Transformer does, basing on sublayers of feed-forward, linear projection, and softmax stacked over the conventional attention as illustrated in the left part of Figure 1. The opponent probability is based on similar sublayers stacked over the opponent attention, but with softmin as the last sublayer as illustrated in the right part of Figure 1.

Due to the contrary properties of softmax and softmin, jointly maximizing $P_c$ and $P_o$ actually maximizes the contribution from the conventional attention for summary word generation, while at the same time minimizes the contribution from the opponent attention[3]. In other words, the training objective is to let the relevant part attended by $\text{attention}_c$ contribute more to the summarization, while let the irrelevant or less relevant parts attended by $\text{attention}_o$ contribute less.

During decoding, we aim to find maximum $\mathcal{J}$ of Equation (9) in the beam search process.

## 4 Experiments

We conduct experiments on abstractive sentence summarization benchmark datasets to demonstrate the effectiveness of the proposed contrastive attention mechanism.

### 4.1 Datasets

In this paper, we evaluate our proposed method on three abstractive text summarization benchmark datasets. First, we use the annotated Gigaword corpus and preprocess it identically to Rush *et al.* (2015), which results in around 3.8M training samples, 190K validation samples and 1951 test samples for evaluation. The source-summary pairs are formed through pairing the first sentence of each article with its headline. We use DUC-2004 as another English data set only for testing in our experiments. It contains 500 documents, each containing four human-generated reference summaries. The length of the summary is capped at 75 bytes. The last data set we used is a large corpus of Chinese short text summarization (LCSTS) (Hu et al., 2015), which is collected from the Chinese microblogging website Sina Weibo. We follow the data split of the original paper, with 2.4M source-summary pairs from the first part of the corpus for training, 725 pairs from the last part with high annotation score for testing.

### 4.2 Experimental Setup

We employ Transformer as our basis architecture[4]. Six layers are stacked in both the encoder and decoder, and the dimensions of the embedding vectors and all hidden vectors are set 512. The inner layer of the feed-forward sublayer has the dimensionality of 2048. We set eight heads in the multi-head attention. The source embedding, the target embedding and the linear sublayer are shared in our experiments. Byte-pair encoding is employed in the English experiment with a shared source-target vocabulary of about 32k tokens (Sennrich et al., 2015).

Regarding the contrastive attention mechanism, the opponent attention is derived from the head

[3] We also tried replacing softmin in Equation (7) with softmax, and correspondingly setting the training objective as maximizing $\mathcal{J} = \log(P_c) - \lambda \log(P_o)$, but this method failed to train because $P_o$ becomes too small during training, and results in negative infinity value of $\log(P_o)$ that hampers the training. In comparison, softmin and the training objective of Equation (9) do not have such problem, enabling the effective training of the proposed network.

[4] https://github.com/pytorch/fairseq

| System | Gigaword | | | DUC2004 | | |
|---|---|---|---|---|---|---|
| | R-1 | R-2 | R-L | R-1 | R-2 | R-L |
| ABS (Rush et al., 2015) | 29.55 | 11.32 | 26.42 | 26.55 | 7.06 | 22.05 |
| ABS+ (Rush et al., 2015) | 29.76 | 11.88 | 26.96 | 28.18 | 8.49 | 23.81 |
| RAS-Elman (Chopra et al., 2016) | 33.78 | 15.97 | 31.15 | 28.97 | 8.26 | 24.06 |
| words-lvt5k-1sent (Nallapati et al., 2016) | 35.30 | 16.64 | 32.62 | 28.61 | 9.42 | 25.24 |
| $\text{SEASS}_{\text{beam}}$ (Zhou et al., 2017) | 36.15 | 17.54 | 33.63 | 29.21 | 9.56 | 25.51 |
| $\text{RNN}_{\text{MRT}}$ (Ayana et al., 2016) | 36.54 | 16.59 | 33.44 | 30.41 | 10.87 | 26.79 |
| Actor-Critic (Li et al., 2018) | 36.05 | 17.35 | 33.49 | 29.41 | 9.84 | 25.85 |
| StructuredLoss (Edunov et al., 2018) | 36.70 | 17.88 | 34.29 | - | - | - |
| DRGD (Li et al., 2017) | 36.27 | 17.57 | 33.62 | 31.79 | 10.75 | 27.48 |
| ConvS2S (Gehring et al., 2017) | 35.88 | 17.48 | 33.29 | 30.44 | 10.84 | 26.90 |
| $\text{ConvS2S}_{\text{ReinforceTopic}}$ (Wang et al., 2018) | 36.92 | 18.29 | 34.58 | 31.15 | 10.85 | **27.68** |
| FactAware (Cao et al., 2018) | 37.27 | 17.65 | 34.24 | - | - | - |
| Transformer | 37.87 | 18.69 | 35.22 | 31.38 | 10.89 | 27.18 |
| Transformer+ContrastiveAttention | **38.72** | **19.09** | **35.82** | **32.22** | **11.04** | 27.59 |

Table 1: ROUGE scores on the English evaluation sets of both Gigaword and DUC2004. On Gigaword, the full-length F-1 based ROUGE scores are reported. On DUC2004, the recall based ROUGE scores are reported. "-" denotes no score is available in that work.

whose attention is most synchronous to word alignments of the source-summary pair. In our experiments, we select the fifth head of the third layer for deriving the opponent attention in the English experiments, and select the second head of the third layer in the Chinese experiments. All dimensions in the contrastive architecture are set 64. The $\lambda$ in Equation (9) is tuned on the development set in each experiment.

During training, we use the Adam optimizer with $\beta 1 = 0.9$, $\beta 2 = 0.98$, $\varepsilon = 10^{-9}$. The initial learning rate is 0.0005. The inverse square root schedule is applied for initial warm up and annealing (Vaswani et al., 2017). During training, we use a dropout rate of 0.3 on all datasets.

During evaluation, we employ ROUGE (Lin, 2004) as our evaluation metric. Since standard Rouge package is used to evaluate the English summarization systems, we also follow the method of Hu *et al.* (2015) to map Chinese words into numerical IDs in order to evaluate the performance on the Chinese data set.

### 4.3 Results

#### 4.3.1 English Results

The experimental results on the English evaluation sets are listed in Table 1. We report the full-length F-1 scores of ROUGE-1 (R-1), ROUGE2 (R-2), and ROUGE-L (R-L) on the evaluation set of the annotated Gigaword, while report the recall-based scores of the R-1, R-2, and R-L on the evaluation set of DUC2004 to follow the setting of the previous works.

The results of our works are shown at the bottom of Table 1. The performances of the related works are reported in the upper part of Table 1 for comparison. ABS and ABS+ are the pioneer works of using neural models for abstractive text summarization. RAS-Elman extends ABS/ABS+ with attentive CNN encoder. words-lvt5k-1sent uses large vocabulary and linguistic features such as POS and NER tags. $\text{RNN}_{\text{MRT}}$, Actor-Critic, StructuredLoss are sequence-level training methods to overcome the problem of the usual teacher-forcing methods. DRGD uses recurrent latent random model to improve summarization quality. FactAware generates summary words conditioned on both the source text and the fact descriptions extracted from OpenIE or dependencies. Besides the above RNN-based related works, CNN-based architectures of ConvS2S and $\text{ConvS2S}_{\text{ReinforceTopic}}$ are included for comparison.

Table 1 shows that we build a strong baseline using Transformer alone which obtains the state-of-the-art performance on Gigaword evaluation set, and obtains comparable performance to the state-of-the-art on DUC2004. When we introduce the contrastive attention mechanism into Transformer, it significantly improves the performance of Transformer, and greatly advances the state-of-the-art on both Gigaword evaluation set and DUC2004, as shown in the row of "Transformer+Contrastive Attention".

#### 4.3.2 Chinese Results

Table 2 presents the evaluation results on LCSTS. The upper rows list the performances of the related works, the bottom rows list the perfor-

| System | R-1 | R-2 | R-L |
|---|---|---|---|
| RNN context (Hu et al., 2015) | 29.90 | 17.40 | 27.20 |
| CopyNet (Gu et al., 2016) | 34.40 | 21.60 | 31.30 |
| $\text{RNN}_{\text{MRT}}$ (Ayana et al., 2016) | 38.20 | 25.20 | 35.40 |
| $\text{RNN}_{\text{distraction}}$ (Chen et al., 2016) | 35.20 | 22.60 | 32.50 |
| DRGD (Li et al., 2017) | 36.99 | 24.15 | 34.21 |
| Actor-Critic (Li et al., 2018) | 37.51 | 24.68 | 35.02 |
| Global (Lin et al., 2018) | 39.40 | 26.90 | 36.50 |
| Transformer | 41.93 | 28.28 | 38.32 |
| Transformer+ContrastiveAttention | **44.35** | **30.65** | **40.58** |

Table 2: The full-length F-1 based ROUGE scores on the Chinese evaluation set of LCSTS.

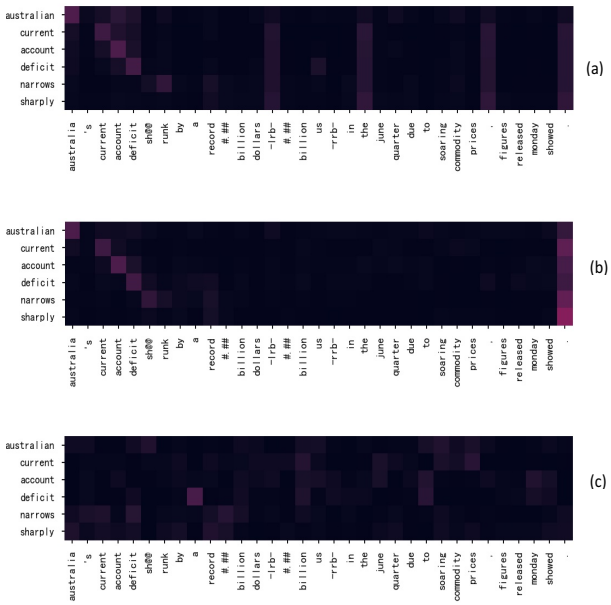


Figure 3: The attention weight changes by using the contrastive attention mechanism. (a) is the average attention weights of the third layer of the baseline Transformer, (b) is that of "Transformer+ContrastiveAttention", and (c) is the opponent attention derived from the fifth head of the third layer.

mances of our Transformer baseline and the integration of the contrastive attention mechanism into Transformer. We only take character sequences as source-summary pairs and evaluate the performance based on reference characters for strict comparison to the related works.

Table 2 shows that Transformer also sets a strong baseline on LCSTS that surpasses the performances of the previous works. When Transformer is equipped with our proposed contrastive attention mechanism, the performance is significantly improved and drastically advances the state-of-the-art on LCSTS.

## 5 Analysis and Discussion

### 5.1 Effect of the Contrastive Attention Mechanism on Attentions

Figure 3 shows the attention weights before and after using the contrastive attention mechanism. We depict the averaged attention weights of all heads in one layer in Figure 3a and 3b to study how it contributes to the conventional probability computation, and depict the opponent attention weights in Figure 3c to study its contribution to the opponent probability. Since we select the fifth head of the third layer to derive the opponent attention in English experiment, the studies are carried out on the third layer.

Figure 3a is from the baseline Transformer, Figure 3b is from "Transformer + ContrastiveAttention". We can see that "Transformer + ContrastiveAttention" is more focused on the source parts that are most relevant to the summary than the baseline Transformer, which scatters attention weights on summary word neighbors or even functional words such as "-lrb-" and "the". "Transformer + ContrastiveAttention" cancels such scattered attentions by using the contrastive attention mechanism.

Figure 3c depicts the opponent attention weights. They are optimized during training to generate the lowest score which is fed into softmin to get the highest opponent probability $P_o$. The more irrelevant to the summary word the opponent is, the lower the score can be obtained, thus resulting in higher $P_o$. Figure 3c shows that the attentions are formed over irrelevant parts with varied weights as the result of maximizing $P_o$ during training.

### 5.2 Effect of the Opponent Probability in Decoding

We study the contribution of the opponent probability $P_o$ by dropping it during decoding to see if it hurts the performance. Table 4 shows that dropping $P_o$ significantly harms the performance of "Transformer + ContrastiveAtt". The performance difference between the model dropping $P_o$ and the baseline Transformer is marginal, indicating that adding the opponent probability $P_o$ is key for achieving the performance improvement.

### 5.3 Explorations on Deriving the Opponent Attention

#### Masking More Attention Weights for Deriving the Opponent Attention

| System | Gigaword | | | DUC2004 | | |
|---|---|---|---|---|---|---|
| | R-1 | R-2 | R-L | R-1 | R-2 | R-L |
| mask maximum weight | 38.72 | 19.09 | 35.82 | 32.22 | 11.04 | 27.59 |
| mask top-2 weights | 38.17 | 19.15 | 35.51 | 31.87 | 10.94 | 27.41 |
| mask top-3 weights | 38.36 | 19.11 | 35.56 | 31.67 | 10.37 | 27.31 |
| dynamically mask | 38.12 | 18.92 | 35.28 | 31.37 | 10.32 | 27.11 |
| synchronous head | 38.72 | 19.09 | 35.82 | 32.22 | 11.04 | 27.59 |
| non-synchronous head | 37.85 | 18.59 | 35.16 | 31.73 | 10.74 | 27.35 |
| averaged head | 38.43 | 19.10 | 35.53 | 31.82 | 10.98 | 27.43 |
| Transformer baseline | 37.87 | 18.69 | 35.22 | 31.38 | 10.89 | 27.18 |

Table 3: Results of explorations on the opponent attention derivation. The upper part presents the influence of masking more attention weights for deriving the opponent attention. The middle part presents the results of selecting different head for the opponent attention derivation. The bottom row presents the result of Transformer.

| Gigaword | R-1 | R-2 | R-L |
|---|---|---|---|
| Transformer | 37.87 | 18.69 | 35.22 |
| Transformer+ContrastiveAtt-$P_o$ | 37.92 | 18.88 | 35.21 |
| Transformer+ContrastiveAtt | 38.72 | 19.09 | 35.82 |
| DUC2004 | R-1 | R-2 | R-L |
| Transformer | 31.38 | 10.89 | 27.18 |
| Transformer+ContrastiveAtt-$P_o$ | 31.21 | 10.70 | 26.85 |
| Transformer+ContrastiveAtt | 32.22 | 11.04 | 27.59 |

Table 4: The effect of dropping $P_o$ (denoted by -$P_o$) from Transformer+ContrastiveAtt during decoding.

In Section 3.2.1, we mask the most salient word that has the maximum weight of $\alpha_c$ to derive the opponent attention. In this subsection, we experimented with masking more weights of $\alpha_c$ by two ways: 1) masking top $k$ weights, 2) dynamically masking. In the dynamically masking method, we order the weights from big to small at first, then go on masking two neighbors until the ratio between them is over a threshold. The threshold is 1.02 based on training and tuning on the development set.

The upper rows of Table 3 presents the performance comparison between masking maximum weight and masking more weights. It shows that masking maximum weight performs better, indicating that masking the most salient weight leaves more irrelevant or less relevant words to compute the opponent probability $P_o$, which is more reliable than that computed from less remaining words after masking more weights.

**Selecting Non-synchronous Head or Averaged Head for Deriving the Opponent Attention**

As explained in Section 3.2.1, the opponent attention is derived from the head that is most synchronous to the word alignments between source sentence and summary. We denote it "synchronous head". We also explored deriving the opponent attention from the fifth head of the first layer, which is non-synchronous to the word alignments as illustrated in Figure 2b. Its result is presented in the "non-synchronous head" row. In addition, the attention weights averaged on all heads of the third layer are used to derive the opponent attention. We denote it "averaged head".

As shown in the middle part of Table 3, both "non-synchronous head" and "averaged head" underperform "synchronous head". "non-synchronous head" performs worst, and even worse than the Transformer baseline on Gigaword. This indicates that it is better to compose the opponent attention from irrelevant parts that can be easily located in the synchronous head. "averaged head" performs slightly worse than "synchronous head", and is also slower due to the involved all heads.

### 5.4 Qualitative Study

Table 5 shows the qualitative results. The highlights in the baseline Transformer manifest the incorrect areas extracted by the baseline system. In contrast, the highlights in Transformer+ContrastiveAtt show that correct contents are extracted since the contrastive system distinguish relevant parts from irrelevant parts on the source side and made attending to correct areas more easily.

## 6 Conclusion

We proposed a contrastive attention mechanism for abstractive sentence summarization, using both the conventional attention that attends to the relevant parts of the source sentence, and a novel opponent attention that attends to irrelevant or less relevant parts for the summary word generation. Both categories of the attention constitute a contrastive pair, and we encourage contribution from the conventional attention and penalize con-

**Src**:press freedom in algeria remains at risk despite the release on wednesday of prominent newspaper editor mohamed UNK after a two-year prison sentence , human rights organizations said .
**Ref**:algerian press freedom at risk despite editor 's release UNK picture
**Transformer**:press freedom remains at risk in algeria rights groups say
**Transformer+ContrastiveAtt**:press freedom remains at risk despite release of algerian editor

**Src**:denmark 's poul-erik hoyer completed his hat-trick of men 's singles badminton titles at the european championships , winning the final here on saturday
**Ref**:hoyer wins singles title
**Transformer**:hoyer completes hat-trick
**Transformer+ContrastiveAtt**:hoyer wins men 's singles title

**Src**:french bank credit agricole launched on tuesday a public cash offer to buy the ## percent of emporiki bank it does not already own , in a bid valuing the greek group at #.# billion euros ( #.# billion dollars ) .
**Ref**:credit agricole announces #.#-billion-euro bid for greek bank emporiki
**Transformer**:credit agricole launches public cash offer for greek bank
**Transformer+ContrastiveAtt**:french bank credit agricole bids #.# billion euros for greek bank

Table 5: Example summaries generated by the baseline Transformer and Transformer+ContrastiveAtt.

tribution from the opponent attention through joint training. Using Transformer as a strong baseline, experiments on three benchmark data sets show that the proposed contrastive attention mechanism significantly improves the performance, advancing the state-of-the-art performance for the task.

## Acknowledgments

The authors would like to thank the anonymous reviewers for the helpful comments. This work was supported by National Key R&D Program of China (Grant No. 2016YFE0132100), National Natural Science Foundation of China (Grant No. 61525205, 61673289).

## References

Ayana, Shiqi Shen, Yu Zhao, Zhiyuan Liu, and Maosong Sun. 2016. Neural headline generation with sentence-wise optimization. *Computer Research Repository*, arXiv:1604.01904. Version 2.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations*.

Michele Banko, Vibhu O Mittal, and Michael J Witbrock. 2000. Headline generation based on statistical translation. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 318–325.

Ziqiang Cao, Furu Wei, Wenjie Li, and Sujian Li. 2018. Faithful to the original: Fact aware neural abstractive summarization. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Qian Chen, Xiao Dan Zhu, Zhen Hua Ling, Si Wei, and Hui Jiang. 2016. Distraction-based neural networks for modeling document. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 2754–2760.

Sumit Chopra, Michael Auli, and Alexander M Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98.

Trevor Cohn and Mirella Lapata. 2008. Sentence compression beyond word deletion. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 137–144.

Bonnie Dorr, David Zajic, and Richard Schwartz. 2003. Hedge trimmer: A parse-and-trim approach to headline generation. In *Proceedings of the HLT-NAACL 03 on Text summarization workshop-Volume 5*, pages 1–8.

Sergey Edunov, Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2018. Classical structured prediction losses for sequence to sequence learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 355–364.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1243–1252.

Jiatao Gu, Zhengdong Lu, Li Hang, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Baotian Hu, Qingcai Chen, and Fangze Zhu. 2015. Lcsts: A large scale chinese short text summarization dataset. In *Proceedings of the 2015 Conference on*

*Empirical Methods in Natural Language Processing*, pages 1967–1972.

Hongyan Jing and Kathleen R McKeown. 2000. Cut and paste based text summarization. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 178–185.

Kevin Knight and Daniel Marcu. 2000. Statistics-based summarization - step one: Sentence compression. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on on Innovative Applications of Artificial Intelligence*, pages 703–710.

Piji Li, Lidong Bing, and Wai Lam. 2018. Actor-critic based training framework for abstractive summarization. *Computing Research Repository*, arXiv:1803.11070.

Piji Li, Wai Lam, Lidong Bing, and Zihao Wang. 2017. Deep recurrent generative decoder for abstractive text summarization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2091–2100.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Proc of the ACL-04 Workshop on Text Summarization Branches Out*.

Junyang Lin, Sun Xu, Shuming Ma, and Su Qi. 2018. Global encoding for abstractive summarization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 163–169.

Ramesh Nallapati, Bowen Zhou, Cicero Nogueira Dos Santos, Caglar Gulcehre, and Xiang Bing. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290.

Joel Larocca Neto, Alex A Freitas, and Celso A. A. Kaestner. 2002. Automatic text summarization using a machine learning approach. In *Brazilian Symposium on Artificial Intelligence*, pages 205–215.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Chunfeng Song, Yan Huang, Wanli Ouyang, and Liang Wang. 2018a. Mask-guided contrastive attention model for person re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1179–1188.

Kaiqiang Song, Lin Zhao, and Fei Liu. 2018b. Structure-infused copy mechanisms for abstractive summarization. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1717–1729.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Li Wang, Junlin Yao, Yunzhe Tao, Li Zhong, Wei Liu, and Qiang Du. 2018. A reinforced topic-aware convolutional sequence-to-sequence model for abstractive text summarization. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence,*, pages 4453–4460.

Kristian Woodsend, Yansong Feng, and Mirella Lapata. 2010. Title generation with quasi-synchronous grammar. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 513–523.

Qingyu Zhou, Yang Nan, Furu Wei, and Zhou Ming. 2017. Selective encoding for abstractive sentence summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1095–1104.

# NCLS: Neural Cross-Lingual Summarization

**Junnan Zhu**[1,2], **Qian Wang**[1,2], **Yining Wang**[1,2],
**Yu Zhou**[1,2]*, **Jiajun Zhang**[1,2], **Shaonan Wang**[1,2], and **Chengqing Zong**[1,2,3]
[1] National Laboratory of Pattern Recognition, Institute of Automation, CAS, Beijing, China
[2] University of Chinese Academy of Sciences, Beijing, China
[3] CAS Center for Excellence in Brain Science and Intelligence Technology, Beijing, China
{junnan.zhu, yzhou, jjzhang, cqzong}@nlpr.ia.ac.cn

## Abstract

Cross-lingual summarization (CLS) is the task to produce a summary in one particular language for a source document in a different language. Existing methods simply divide this task into two steps: summarization and translation, leading to the problem of error propagation. To handle that, we present an end-to-end CLS framework, which we refer to as Neural Cross-Lingual Summarization (NCLS), for the first time. Moreover, we propose to further improve NCLS by incorporating two related tasks, monolingual summarization and machine translation, into the training process of CLS under multi-task learning. Due to the lack of supervised CLS data, we propose a round-trip translation strategy to acquire two high-quality large-scale CLS datasets based on existing monolingual summarization datasets. Experimental results have shown that our NCLS achieves remarkable improvement over traditional pipeline methods on both English-to-Chinese and Chinese-to-English CLS human-corrected test sets. In addition, NCLS with multi-task learning can further significantly improve the quality of generated summaries. We make our dataset and code publicly available here: http://www.nlpr.ia.ac.cn/cip/dataset.htm.

## 1 Introduction

Given a document in one source language, cross-lingual summarization aims to produce a summary in a different target language, which can help people efficiently acquire the gist of an article in a foreign language. Traditional approaches to CLS are based on the pipeline paradigm, which either first translates the original document into target language and then summarizes the translated document (Leuski et al., 2003) or first summarizes the original document and then translates the summary into target language (Lim et al., 2004; Orasan and Chiorean, 2008; Wan et al., 2010). However, the current machine translation (MT) is not perfect, which results in the error propagation problem. Although end-to-end deep learning has made great progress in natural language processing, no one has yet applied it to CLS due to the lack of large-scale supervised dataset.

The input and output of CLS are in two different languages, which makes the data acquisition much more difficult than monolingual summarization (MS). To the best of our knowledge, no one has studied how to automatically build a high-quality large-scale CLS dataset. Therefore, in this work, we introduce a novel approach to directly address the lack of data. Specifically, we propose a simple yet effective round-trip translation strategy to obtain cross-lingual document-summary pairs from existing monolingual summarization datasets (Hermann et al., 2015; Zhu et al., 2018; Hu et al., 2015). More details can be found in Section 2 below.

Based on the dataset that we have constructed, we propose end-to-end models on cross-lingual summarization, which we refer to as Neural Cross-Lingual Summarization (NCLS). Furthermore, we consider improving CLS with two related tasks: MS and MT. We incorporate the training process of MS and MT into that of CLS under the multi-task learning framework (Caruana, 1997). Experimental results demonstrate that NCLS achieves remarkable improvement over traditional pipeline paradigm. In addition, both MS and MT can significantly help to produce better summaries.

Our main contributions are as follows:

- We propose a novel round-trip translation strategy to acquire large-scale CLS datasets from existing large-scale MS datasets. We have constructed a 370K English-to-Chinese

*Corresponding author.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3054–3064,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

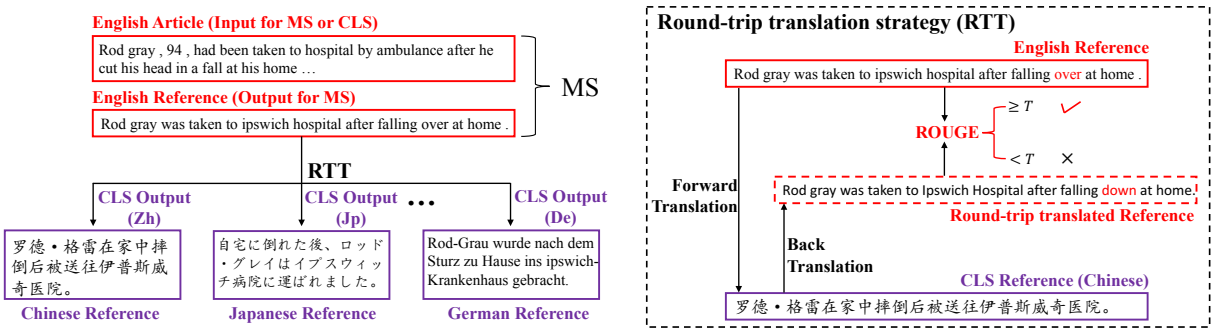


Figure 1: Overview of CLS corpora construction. Our method can be extended to many other language pairs and we focus on En2Zh and Zh2En in this paper. During RTT, we filter the sample in which ROUGE F1 score between the original reference and the round-trip translated reference is below a preset threshold $T$.

(En2Zh) CLS corpus and a 1.69M Chinese-to-English (Zh2En) CLS corpus.

- To train the CLS systems in an end-to-end manner, we present neural cross-lingual summarization. Furthermore, we propose to improve NCLS by incorporating MT and MS into CLS training process under multi-task learning. To the best of our knowledge, this is the first work to present an end-to-end CLS framework that trained on parallel corpora.

- Experimental results demonstrate that NCLS can achieve +4.87 ROUGE-2 on En2Zh and +5.07 ROUGE-2 on Zh2En over traditional pipeline paradigm. In addition, NCLS with multi-task learning can further achieve +3.60 ROUGE-2 on En2Zh and +0.72 ROUGE-2 on Zh2En. Our methods can be regarded as a benchmark for further NCLS studying.

## 2 Dataset Construction

Existing large-scale monolingual summarization datasets are automatically collected from the internet. CNN/Dailymail (Hermann et al., 2015) dataset has been collected from *CNN* and *DailyMail* websites, where the article and news highlights are treated as the input and output respectively. Similar to Hermann et al. (2015), Zhu et al. (2018) have constructed a multimodal summarization dataset MSMO where the text input and output are similar to that in CNN/Dailymail. We refer to the union set of CNN/Dailymail and MSMO as ENSUM[1]. Hu et al. (2015) introduce a large-scale corpus of Chinese short text summarization (LCSTS[2]) dataset constructed from the Chinese microblogging website *Sina Weibo*. In this section, we introduce how to construct the En2Zh and Zh2En CLS datasets based on ENSUM and LCSTS respectively.

**Round-trip translation strategy.** Round-trip translation[3] (RTT) is the process of translating a text into another language (forward translation), then translating the result back into the original language (back translation), using MT service[4]. Inspired by Lample et al. (2018), we propose to adopt the round-trip translation to acquire CLS dataset from MS dataset. The process of constructing our corpora is shown in Figure 1.

Taking the construction of En2Zh corpus as an example, given a document-summary pair $(D_{\text{en}}, S_{\text{en}})$, we first translate the summary $S_{\text{en}}$ into Chinese $S_{\text{zh}}$ and then back into English $S'_{\text{en}}$. The En2Zh document-summary pair $(D_{\text{en}}, S_{\text{zh}})$, which satisfies ROUGE-1$(S_{\text{en}}, S'_{\text{en}}) \geqslant T_1$ and ROUGE-2$(S_{\text{en}}, S'_{\text{en}}) \geqslant T_2$ ($T_1$ is set to 0.45 for English and 0.6 for Chinese respectively, and $T_2$ is set to 0.2 here[5]), will be regarded as a positive pair. Otherwise, the pair will be filtered. Note that there are multiple sentences in $S_{\text{en}}$ in ENSUM, we apply the RTT to filter low-quality translated reference sentence by sentence. Once more than two-thirds of the sentences in the summary in a sample are retained, we will keep the sample. This process helps to ensure that the final compression ratio in our task does not differ too much from the actual compression ratio. Similar process is used on constructing Zh2En corpus. The ROUGE scores between Chinese sentences are calculated using Chinese characters as segmentation units.

[1]It contains 626,634 English summarization pairs.
[2]It contains 2,400,591 Chinese summarization pairs.
[3]https://en.wikipedia.org/wiki/Round-trip_translation
[4]http://www.anylangtech.com
[5]The values are obtained by conducting a manual estimation on some samples randomly selected from two corpora.

| En2ZhSum | train | valid | test | Zh2EnSum | train | valid | test |
|---|---|---|---|---|---|---|---|
| #Documents | 364,687 | 3,000 | 3,000 | #Documents | 1,693,713 | 3,000 | 3,000 |
| #AvgWords (S) | 755.09 | 759.55 | 744.84 | #AvgChars (S) | 103.59 | 103.56 | 140.06 |
| #AvgEnWords (R) | 55.21 | 55.28 | 54.76 | #AvgZhChars (R) | 17.94 | 18.00 | 18.08 |
| #AvgZhChars (R) | 95.96 | 96.05 | 95.33 | #AvgEnWords (R) | 13.70 | 13.74 | 13.84 |
| #AvgSentsWords | 19.62 | 19.63 | 19.61 | #AvgSentsChars | 52.73 | 52.41 | 53.38 |
| #AvgSents | 40.62 | 41.08 | 40.25 | #AvgSents | 2.32 | 2.33 | 2.30 |

Table 1: Corpus statistics. **#AvgWords (S)** is the average number of English words in the source document. Each reference has a bilingual version since each reference in CLS corpus is translated from the corresponding reference in the MS corpus. **#AvgEnWords (R)** means the average number of words in English reference and **#AvgZhChars (R)** denotes the average number of characters in Chinese reference. **#AvgSentsWords** (**#AvgSentsChars**) indicates the average number of words (characters) in a sentence in the source document. **#AvgSents** refers to the average number of sentences in the source document.

**Corpus Statistics.** After conducting the round-trip translation strategy, we have obtained 370,759 En2Zh CLS pairs from ENSUM and 1,699,713 Zh2En CLS pairs from LCSTS. The statistics of En2Zh corpus (En2ZhSum) and Zh2En corpus (Zh2EnSum) are presented in Table 1. In order to evaluate various CLS methods more reliably, we recruit 10 volunteers to correct the reference in the test sets in two constructed corpora.

## 3 Approach

The traditional approaches (Section 3.1) intuitively treat CLS as a pipeline process which leads to error propagation. To handle that, we present the neural cross-lingual summarization methods (Section 3.2), which train CLS in an end-to-end manner for the first time. Due to the strong relationship between CLS, MS, and MT tasks, we propose to incorporate MS and MT into CLS training under multi-task learning (Section 3.3).

### 3.1 Baseline Pipeline Methods

In general, traditional CLS is composed of summarization step and translation step. The different order of these two steps leads to the following two strategies. Take En2Zh CLS as an example.

**Early Translation (ETran).** This strategy first translates the English document to Chinese document with machine translation. Then a Chinese summary is generated by a summarization model.

**Late Translation (LTran).** This strategy first summarizes the English document to a short English summary and then translates it into Chinese.

### 3.2 Neural Cross-Lingual Summarization

Considering the excellent text generation performance of Transformer encoder-decoder network (Vaswani et al., 2017), we implement our NCLS models entirely based on this framework in this work. As shown in Figure 2, given a set of CLS data $D = (X^{(i)}, Y^{(i)})$ where both $X$ and $Y$ are a sequence of tokens, the encoder maps the input document $X = (x_1, x_2, \cdots, x_n)$ into a sequence of continuous representations $z = (z_1, z_2, \cdots, z_n)$ whose size varies with respect to the source sequence length. The decoder generates a summary $Y = (y_1, y_2, \cdots, y_m)$, which is in a different language, from the continuous representations. The encoder and decoder are trained jointly to maximize the conditional probability of target sequence given a source sequence:

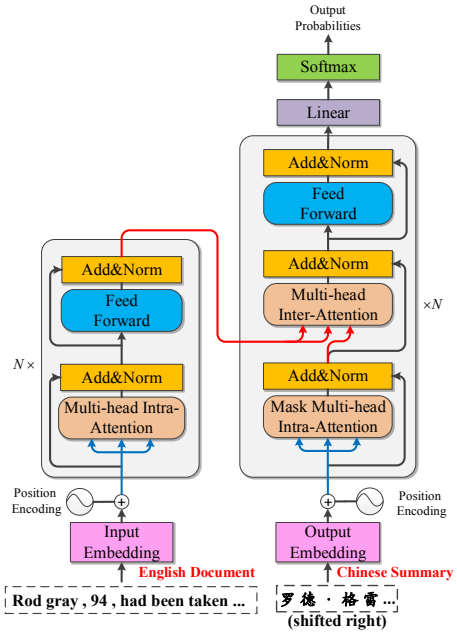


Figure 2: Transformer-based NCLS models (TNCLS).

$$L_\theta = \sum_{t=1}^{N} \log \mathrm{P}(\mathrm{y_t} | \mathrm{y_{<t}}, \mathrm{x}; \theta) \quad (1)$$

Transformer is composed of stacked encoder and decoder layers. Consisting of two blocks, the

encoder layer is a self-attention block followed by a position-wise feed-forward block. Despite the same architecture as the encoder layer, the decoder layer has an extra encoder-decoder attention block. Residual connection and layer normalization are used around each block. In addition, the self-attention block in the decoder is modified with masking to prevent present positions from attending to future positions during training.

For self-attention and encoder-decoder attention, a multi-head attention block is used to obtain information from different representation subspaces at different positions. Each head corresponds to a scaled dot-product attention, which operates on the query $Q$, key $K$, and value $V$:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \quad (2)$$

where $d_k$ is the dimension of the key.

Finally, the output values are concatenated and projected by a feed-forward layer to get final values:

$$\begin{aligned}\text{MultiHead}(Q, K, V) &= \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \\ \text{where head}_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)\end{aligned} \quad (3)$$

where $W^O$, $QW_i^Q$, $KW_i^K$, and $VW_i^V$ are learnable matrices, $h$ is the number of heads.

### 3.3 Improving NCLS with MS and MT

Considering there is a strong relationship between CLS task and MS task, as well as between CLS task and MT task: (1) CLS shares the same goal with MS, i.e., to grasp the core idea of the original document, but the final results are presented in different languages. (2) From the perspective of information compression, machine translation can be regarded as a special kind of cross-lingual summarization with a compression ratio of 1:1. Therefore, we consider using MS and MT datasets to further improve the performance of CLS task under multi-task learning.

Inspired by Luong et al. (2016), we employ the one-to-many scheme to incorporate the training process of MS and MT into that of CLS. As shown in Figure 3, this scheme involves one encoder and multiple decoders for tasks in which the encoder can be shared. We study two different task combinations here: CLS+MS and CLS+MT.

**CLS+MS.** Note that the reference in each of CLS datasets has a bilingual version. For instance, En2ZhSum dataset contains a total of 370,687 documents with corresponding summaries in both

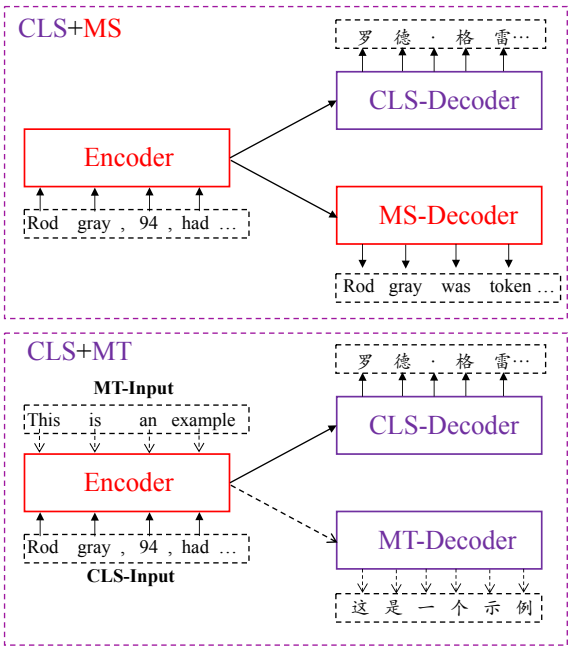


Figure 3: Overview of multi-task NCLS. The lower half is CLS+MT using alternating training strategy. Different colors represent different languages.

Chinese and English. Thus, we consider jointly training CLS and MS as follows. Given a source document, the encoder encodes it into continuous representations, and then the two decoders simultaneously generate the output of their respective tasks. The loss can be calculated as follows:

$$L_\theta = \sum_{t=1}^{N^{(1)}} \log P(y_t^{(1)} | y_{<t}^{(1)}, x; \theta) + \sum_{t=1}^{N^{(2)}} \log P(y_t^{(2)} | y_{<t}^{(2)}, x; \theta) \quad (4)$$

where $y^{(1)}$ and $y^{(2)}$ are the outputs of two tasks.

**CLS+MT.** Since CLS input-output pairs are different from MT input-output pairs, we consider adopting the alternating training strategy (Dong et al., 2015), which optimizes each task for a fixed number of mini-batches before switching to the next task, to jointly train CLS and MT. For MT task, we employ 2.08M[6] sentence pairs from LDC corpora with CLS dataset to train CLS+MT.

## 4 Experiments

### 4.1 Experimental Settings

For English, we apply two different granularities of segmentation, i.e., words and subwords (Sennrich et al., 2016). We lowercase all English characters. We truncate the input to 200 words and the output to 120 words (150 characters for Chinese output) . For Chinese, we employ three different

[6]LDC2000T50, LDC2002L27, LDC2002T01, LDC2002E18, LDC2003E07, LDC2003E14, LDC2003T17, LDC2004T07

| Model | ROUGE-1 | ROUGE-2 | ROUGE-L |
|---|---|---|---|
| Gu et al. (2016) | 35.00 | 22.30 | 32.00 |
| Li et al. (2017) | 36.99 | 24.15 | 34.21 |
| Transformer | 39.71 | 27.45 | 37.13 |

Table 2: Performance of our implemented transformer-based monolingual summarization model on LCSTS.

granularities of segmentation: characters, words, and subwords. It is worth noting that we only apply subword-based segmentation in Zh2En model since subword-based segmentation will make the English article much longer in En2Zh (especially at the Chinese target-side output), which makes the Transformer performs extremely poor. For our baseline pipeline models, the vocabulary size of Chinese characters is 10,000, and that of Chinese words, Chinese subwords, and English words are all 100,000. In our En2Zh NCLS models, the vocabulary size of source-side English words is 100,000, and that of target-side Chinese characters and words are 18,000, and 50,000 respectively. In our Zh2En models, the vocabulary size of source-side Chinese characters, words, and subwords are 10,000, 100,000, and 100,000 respectively, and that of target-side English words and subwords are all 40,000. We initialize all the parameters via Xavier initialization methods (Glorot and Bengio, 2010). We train our models using configuration *transformer_base* (Vaswani et al., 2017), which contains a 6-layer encoder and a 6-layer decoder with 512-dimensional hidden representations.

During training, in En2Zh models, each mini-batch contains a set of document-summary pairs with roughly 2,048 source and 2,048 target tokens; in Zh2En models, each mini-batch contains a set of document-summary pairs with roughly 4,096 source and 4,096 target tokens. We use Adam optimizer (Kingma and Ba, 2015) with $\beta_1 = 0.9$, $\beta_2 = 0.998$, and $\epsilon = 10^{-9}$. We use a single NVIDIA TITAN X to train our models. Convergence is reached within 1,000,000 iterations in both TNCLS models and baseline models. We train each task for about 800,000 iterations in multi-task NCLS models (reaching convergence). At test time, our summaries are produced using beam search with beam size 4.

### 4.2 Baselines and Model Variants

We compare our NCLS models with the following two traditional methods:

**TETran:** We first translate the source document via a Transformer-based machine translation model trained on LDC corpora. Then we employ LexRank (Erkan and Radev, 2004), a strong and widely used unsupervised summarization method, to summarize the translated document. The reason why we choose to apply an unsupervised method is that we lack the version of MS dataset in the target language to train a supervised model to summarize the translated document.

| Model | ROUGE-1 | ROUGE-2 | ROUGE-L |
|---|---|---|---|
| See et al. (2017) | 39.53 | 17.28 | 36.38 |
| Transformer | 39.24 | 16.67 | 36.42 |

Table 3: Performance of our implemented transformer-based MS model on CNN/Dailymail.

**TLTran:** We first build a Transformer-based MS model which is trained on the original MS dataset. Then the MS model aims to summarize the source document into a summary. Finally, we translate the summary into target language by using the Transformer-based machine translation model trained on LDC corpora. The performance of our transformer-based MS models is given in Table 2 and Table 3.

To make our experiments more comprehensive, during the process of TETran and TLTran, we replace the Transformer-based machine translation model with Google Translator[7], which is one of the state-of-the-art machine translation systems. We refer to these two methods as **GETran** and **GLTran** respectively.

There are three variants of our NCLS models:

**TNCLS:** Transformer-based NCLS models where the input and output are different granularities combinations of units.

**CLS+MS:** It refers to the multi-task NCLS model which accepts an input text and simultaneously performs text generation for both CLS and MS tasks and calculates the total losses.

**CLS+MT:** It trains CLS and MT tasks via alternating training strategy. Specifically, we optimize the CLS task in a mini-batch, and we optimize the MT task in the next mini-batch.

### 4.3 Experimental Results and Analysis

**Comparison between NCLS with baselines.** We evaluate different models with the standard ROUGE metric (Lin, 2004), reporting the F1 scores for ROUGE-1, ROUGE-2, and ROUGE-L. The results are presented in Table 4.

[7] https://translate.google.com/

[8] The parameter for ROUGE script here is "-c 95 -r 1000 -n 2 -a".

| Model | Unit | En2ZhSum | En2ZhSum* | Zh2EnSum | Zh2EnSum* |
|---|---|---|---|---|---|
| | | RG1-RG2-RGL(↑) | RG1-RG2-RGL(↑) | RG1-RG2-RGL(↑) | RG1-RG2-RGL(↑) |
| TETran | – | **26.12-10.59-23.21** | **26.15-10.60-23.24** | **22.81- 7.17-18.55** | **23.09- 7.33-18.74** |
| GETran | – | **28.17-11.38-25.75** | **28.19-11.40-25.77** | **24.03- 8.91-19.92** | **24.34- 9.14-20.13** |
| TLTran | c-c | – | – | 32.85-15.34-29.21 | 33.01-15.43-29.32 |
| | w-w | **30.20-12.20-27.02** | **30.22-12.20-27.04** | 31.11-13.23-27.55 | 31.38-13.42-27.69 |
| | sw-sw | – | – | **33.64-15.58-29.74** | **33.92-15.81-29.86** |
| GLTran | c-c | – | – | 34.44-15.71-30.13 | 34.58-16.01-30.25 |
| | w-w | **32.15-13.84-29.42** | **32.17-13.85-29.43** | 32.42-15.19-28.75 | 32.52-15.39-28.88 |
| | sw-sw | – | – | **35.28-16.59-31.08** | **35.45-16.86-31.28** |
| TNCLS | c-w | – | – | 36.36-19.74-32.66 | 35.82-19.04-32.06 |
| | w-c | **36.83-18.76-33.22** | **36.82-18.72-33.20** | – | – |
| | w-w | 33.09-14.85-29.82 | 33.10-14.83-29.82 | 38.54-22.34-35.05 | 37.70-21.15-34.05 |
| | sw-sw | – | – | **39.80-23.15-36.11** | **38.85-21.93-35.05** |

Table 4: ROUGE F1 scores (%) on En2ZhSum and Zh2EnSum test sets. En2ZhSum* and Zh2EnSum* are the corresponding human-corrected test sets. *Unit* denotes the granularity combination of text units, where *c* means character, *w* means word, and *sw* means subword. RG refers to ROUGE for short. ↑ indicates that the larger values, the better the results are. Our NCLS models perform significantly better than baseline models by the 95% confidence interval measured by the official ROUGE script[8].

We can find that GLTran outperforms TLTran and GETran outperforms TETran, which indicates that pipeline-based methods perform better when using a stronger machine translation system. Compared with GLTran or GETran, our TNCLS models both achieve significant improvements, which can verify our motivation and demonstrate the efficacy of our constructed corpora.

In En2Zh CLS task, the results of each model on En2ZhSum are similar to those on En2ZhSum*. This is because the original ENSUM dataset comes from the news reports. Existing MT for news reports has excellent performance. Besides, we have pre-filtered samples with low translation quality during dataset construction. Therefore, the quality of the automatic test set is high. TNCLS (*w-c*) performs significantly better than TNCLS (*w-w*). This is because the character-based segmentation can greatly reduce the vocabulary size at the Chinese target-side, which leads to generating nearly no UNK token during the decoding process.

In Zh2En CLS task, the subword-based models outperform others since subword-based segmentation can greatly reduce the vocabulary size and the generation of UNK. Compared with baselines, TNCLS can achieve maximum improvement up to **+4.52 ROUGE-1**, **+6.56 ROUGE-2**, **+5.03 ROUGE-L** on Zh2EnSum and **+3.40 ROUGE-1**, **+5.07 ROUGE-2**, **+3.77 ROUGE-L** on Zh2EnSum*. The results of TNCLS drops obviously on the human-corrected test set, showing that the quality of the translated reference is not as perfect as expected. The reason is straightforward that the original LCSTS dataset comes from social media so that the proportion of abbreviations and omitting punctuation in its text is much higher than in news, resulting in lower translation quality.

In conclusion, TNCLS models significantly outperform the traditional pipeline methods on both En2Zh and Zh2En CLS tasks.

**Why Back Translation?** To show the influence of filtering the corpus by back translation during the RTT process, we use three kinds of datasets to train our TNCLS models and compare their performance. They are: (a) the CLS dataset obtained by simply employing forward translation on MS dataset (*Non-Filter*); (b) the CLS dataset obtained by a complete RTT process (*Filter*); (c) the dataset obtained by sampled from *Non-Filter* dataset to keep the same size as the *Filter* dataset (*Pseudo-Filter*). The results are given in Table 5. The models trained on *Filter* dataset significantly outperform the models trained on *Pseudo-Filter* dataset on both En2Zh and Zh2En tasks, which indicates that the back translation can effectively filter dirty samples and improve the overall quality of corpora, thus boosting the performance of NCLS.

| DataVersion | BT? | En2ZhSum | En2ZhSum* | Zh2EnSum | Zh2EnSum* |
|---|---|---|---|---|---|
| | | RG1-RG2-RGL(↑) | RG1-RG2-RGL(↑) | RG1-RG2-RGL(↑) | RG1-RG2-RGL(↑) |
| Filter | YES | 36.83-18.76-33.22 | 36.82-18.72-33.20 | 39.80-23.15-36.11 | 38.85-21.93-35.05 |
| Pseudo-Filter | NO | 36.04-17.80-32.49 | 36.03-17.78-32.48 | 35.58-17.93-31.71 | 35.00-17.37-31.10 |
| Non-Filter | NO | 37.62-19.88-33.99 | 37.62-19.85-33.99 | 36.51-19.23-32.77 | 36.03-18.63-32.19 |

Table 5: Experimental results on different versions of datasets. *Filter* refers to the version of dataset for which we employ RTT strategy to filter. *Non-Filter* denotes the version of the dataset obtained by simply forward translation without filtering process including back translation. *Pseudo-Filter* is the dataset randomly sampled from *Non-Filter* version and is of the same size as *Filter* version. BT refers to back translation in RTT. For En2Zh task, we train the TNCLS (*w-c*). For Zh2En task, we train the TNCLS (*sw-sw*).

| Model | En2ZhSum | En2ZhSum* | Zh2EnSum | Zh2EnSum* |
|---|---|---|---|---|
| | RG1-RG2-RGL(↑) | RG1-RG2-RGL(↑) | RG1-RG2-RGL(↑) | RG1-RG2-RGL(↑) |
| TNCLS | **36.83-18.76-33.22** | **36.82-18.72-33.20** | **39.80-23.15-36.11** | **38.85-21.93-35.05** |
| CLS+MS | 38.23-20.21-34.76 | 38.25-20.20-34.76 | 41.08-23.67-37.19 | **40.34-22.65-36.39** |
| CLS+MT | **40.24-22.36-36.61** | **40.23-22.32-36.59** | **41.09-23.70-37.17** | 40.25-22.58-36.21 |

Table 6: Results of multi-task NCLS. The granularity combination of input and output in En2Zh task is "word to character" (*w-c*), and that in Zh2En task is "subword to subword" (*sw-sw*).

In En2Zh task, the model trained on *Non-Filter* dataset performs best. The reasons are two-fold: (1) the quality of machine translation for English news is reliable; (2) the scale of *Non-Filter* dataset is almost twice that of the two others so that after the amount of data reaches a certain level, it can make up for the noises caused by the translation error in the corpus. In Zh2En task, the performance of the model trained on *Non-Filter* dataset is not as good as that on *Filter*. It can be attributed to the fact that current MT is not very ideal in the translation of texts on social media so that the dataset constructed by only using forward translation contains too many noises. Therefore, when the quality of machine translation is not that ideal, backward translation is especially important during the process of constructing corpus.

**Results of Multi-task NCLS.** To explore whether MS and MT can further improve NCLS, we compare the multi-task NCLS with NCLS using one same granularity combination of units. The results are given in Table 6. As shown in Table 6, both CLS+MS and CLS+MT can improve the performance of NCLS, which can be attributed to that the encoder is enhanced by incorporating MS and MT data into the training process. CLS+MT significantly outperforms CLS+MS in En2Zh task while CLS+MS performs comparably with CLS+MT in Zh2En task. The reasons are two-fold: (1) In En2Zh task, MT dataset is much larger than both MS and CLS datasets, which makes it more necessary for enhancing the robustness of encoder. (2) We use the LDC MT dataset, which belongs to the news domain similar to our En2ZhSum, during the training of CLS+MT. However, Zh2EnSum belongs to social media domain, thus resulting in the greater improvement of CLS+MT in En2Zh than in Zh2En. In general, NCLS with multi-task learning achieves more significant improvement in En2Zh task than in Zh2En task, which illustrates that extra dataset in other related tasks is essentially important for boosting the performance when CLS dataset is not very large.

**Human Evaluation.** We conduct the human evaluation on 25 random samples from each of the En2ZhSum and Zh2EnSum test set. We compare the summaries generated by our methods (including TNCLS, CLS+MS, and CLS+MT) with the summaries generated by GLTran. Three graduate students are asked to compare the generated summaries with human-corrected references, and assess each summary from three independent perspectives: (1) How informative the summary is? (2) How concise the summary is? (3) How fluent, grammatical the summary is? Each property is assessed with a score from 1 (worst) to 5 (best). The average results are presented in Table 7.

As shown in Table 7, TNCLS can generate more informative summaries compared with GLTran, which shows the advantage of end-to-end models. The conciseness score and fluency score of TNCLS are comparable to those of GLTran. This is because both GLTtrans and TNCLS employ a single encoder-decoder model, which eas-

| |
|---|
| Input (Chinese): 在成本压力加大的情况下，流通企业不仅没有缩减IT投资反而继续增加。2011年，中国流通行业的IT投资规模由2010年的96.6亿元增加至2011年的109.2亿元。预计2012年流通行业的IT投资增速将达14.1%，规模超120亿元。<br>Under the circumstance of increasing cost pressure, circulation enterprises not only did not reduce IT investment but also continued to increase. In 2011, the scale of IT investment in China 's circulation industry increased from 9.66 billion yuan in 2010 to 10.92 billion yuan in 2011. It is estimated that the IT investment in the circulation industry will grow by 14.1% in 2012, with a scale exceeding 12 billion yuan. |
| Gold Summary: in 2012 , the scale of it investment in china 's circulation industry will exceed 12 billion yuan . |
| GETran: in the case of increased cost pressures, distribution companies have not only reduced it investment but continued to increase. |
| GLTran: it investment in china 's circulation industry will increase by 14.1 % in 2011 |
| TNCLS: it investment in circulation industry continues to increase |
| CLS+MS: it investment in china 's circulation industry will exceed 12 billion yuan in 2012 . |
| CLS+MT: china 's circulation industry is expected to increase it investment by 14.1 % in 2012 . |

Figure 4: Examples of generated summaries.

| Model | En2Zh | | | Zh2En | | |
|---|---|---|---|---|---|---|
| | IF | CC | FL | IF | CC | FL |
| GLTran | 3.06 | 3.37 | 3.13 | 3.53 | 4.21 | 4.25 |
| TNCLS | 3.25 | 3.33 | 3.17 | 3.67 | 4.25 | 4.24 |
| CLS+MS | 3.53 | 3.58 | 3.53 | 3.72 | 4.31 | 4.28 |
| CLS+MT | **3.58** | **3.76** | **3.63** | **3.78** | **4.43** | **4.35** |

Table 7: Human evaluation results. IF, CC and FL denote informative, concise, and fluent respectively.

ily leads to under-generation and repetition. Our CLS+MS and CLS+MT can significantly improve the conciseness and fluency of generated summaries, which shows that these methods can generate shorter summaries and reduce grammatical errors. In conclusion, TNCLS can generate more informative summaries, but it is difficult to improve the conciseness and fluency. However, with the help of MT and MS tasks, conciseness and fluency scores can be significantly improved.

### 4.4 Case Study

We show the case study of a sample from the Zh2EnSum human-corrected test set in Figure 4. As shown in Figure 4, the summary generated by **GETran** obviously suffers from errors of machine translation ("distribution companies" should be corrected as "circulation enterprises"). Since **GETran** first translates all the source text, it is easier to bring the errors from machine translation. The **GLTran**-generated summary contracts the fact that the year in it should be 2012 instead of 2011. The translation quality of the sentence is relatively reliable, thus the errors are probably produced during the summarization step. Compared with the first two generated summaries, although the summary produced by **TNCLS** does not emphasize the time and place of occurrence, there is no mistake in the logic of its expression. The summaries generated by **CLS+MS** and **CLS+MT** are generally consistent with the facts, but their emphases are different. The **CLS+MS** summary matches the gold summary better. The flaws of both of them are that they do not reflect the "scale" in the original text. In conclusion, our methods can produce more accurate summaries than baselines.

## 5 Related Work

Cross-lingual summarization has been proposed to present the most salient information of a source document in a different language, which is very important in the field of multilingual information processing. Most of the existing methods handle the task of CLS via simply applying two typical translation schemes, i.e., early translation (Leuski et al., 2003; Ouyang et al., 2019) and late translation (Orasan and Chiorean, 2008; Wan et al., 2010). The early translation scheme first translates the original document into target language and then generates the summary of the translated document. The late translation scheme first summarizes the original document into a summary in the source language and then translates it into target language.

Leuski et al. (2003) translate the Hindi document to English and then generate the English headline for it. Ouyang et al. (2019) present a robust abstractive summarization system for low resource languages where no summarization corpora are currently available. They train a neural abstractive summarization model on noisy English documents and clean English reference summaries. Then the model can learn to produce fluent summaries from disfluent inputs, which allows generating summaries for translated documents. Orasan and Chiorean (2008) summarize the Romanian news with the maximal marginal relevance method (Goldstein et al., 2000) and produce the English summaries for English speakers. Wan et al. (2010) adopt the late translation scheme for the task of English-to-Chinese CLS. They extract English sentences considering both the informativeness and translation quality of sentences and automatically translate the English summary into the final Chinese summary. The above researches only make use of the information from only one language side.

Some methods have been proposed to improve CLS with bilingual information. Wan (2011) proposes two graph-based summarization methods to leverage both the English-side and Chinese-side information in the task of English-to-Chinese CLS. Inspired by the phrase-based translation models, Yao et al. (2015) introduce a compressive CLS, which simultaneously performs sentence selection and compression. They calculate the sentence scores based on the aligned bilingual phrases obtained by MT service and perform compression via deleting redundant or poorly translated phrases. Zhang et al. (2016) propose an abstractive CLS which constructs a pool of bilingual concepts represented by the bilingual elements of the source-side predicate-argument structures (PAS) and the target-side counterparts. The final summary is generated by maximizing both the salience and translation quality of the PAS elements.

However, all these researches belong to the pipeline paradigm which not only relies heavily on hand-crafted features but also causes error propagation. End-to-end deep learning has proven to be able to alleviate these two problems, while it has been absent due to the lack of large-scale training data. Recently, Ayana et al. (2018) present zero-shot cross-lingual headline generation based on existing parallel corpora of translation and monolingual headline generation. Similarly, Duan et al. (2019) propose to use monolingual abstractive sentence summarization system to teach zero-shot cross-lingual abstractive sentence summarization on both summary word generation and attention. Although great efforts have been made in cross-lingual summarization, how to automatically build a high-quality large-scale cross-lingual summarization dataset remains unexplored.

In this paper, we focus on English-to-Chinese and Chinese-to-English CLS and try to automatically construct two large-scale corpora respectively. In addition, based on the two corpora, we perform several end-to-end training methods noted as Neural Cross-Lingual Summarization.

## 6 Conclusion and Future Work

In this paper, we present neural cross-lingual summarization for the first time. To achieve that goal, we propose to acquire large-scale supervised data from existing monolingual summarization datasets via round-trip translation strategy. Then we apply end-to-end methods on our constructed datasets and find our NCLS models significantly outperform the traditional pipeline paradigm. Furthermore, we consider utilizing machine translation and monolingual summarization to further improve NCLS. Experimental results have shown that both machine translation and monolingual summarization can significantly help NCLS generate better summaries.

In our future work, we will adopt our RTT strategy to obtain CLS datasets of other language pairs, such as English-to-Japanese, English-to-German, Chinese-to-Japanese, and Chinese-to-German, etc.

## 7 Acknowledgments

The research work described in this paper has been supported by the National Key Research and Development Program of China under Grant No. 2016QY02D0303. We thank the three anonymous reviewers for their careful reading of our paper and their many insightful comments and suggestions. We would like to thank He Bai, Yuchen Liu, Haitao Lin, Yang Zhao, Cong Ma, Lu Xiang, Weikang Wang, Zhen Wang, and Jiaqi Liang for their invaluable contributions in shaping the early stage of this work. We thank Xina Fu, Jinliang Lu, and Sikai Liu for conducting human evaluation.

## References


Ayana, shi-qi Shen, Yun Chen, Cheng Yang, Zhi-yuan Liu, and Maosong Sun. 2018. Zero-shot cross-lingual neural headline generation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP)*, 26(12):2319–2327.

Rich Caruana. 1997. Multitask learning. *Machine Learning*, 28(1):41–75.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 1723–1732.

Xiangyu Duan, Mingming Yin, Min Zhang, Boxing Chen, and Weihua Luo. 2019. Zero-shot cross-lingual abstractive sentence summarization through teaching generation and attention. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 3162–3172.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research (JAIR)*, 22:457–479.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS10). Society for Artificial Intelligence and Statistics*.

Jade Goldstein, Vibhu Mittal, Jaime Carbonell, and Mark Kantrowitz. 2000. Multi-document summarization by sentence extraction. In *Proceedings of the NAACL-ANLP Workshop on Automatic summarization*, pages 40–48.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1631–1640.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1693–1701.

Baotian Hu, Qingcai Chen, and Fangze Zhu. 2015. LCSTS: A large scale Chinese short text summarization dataset. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1967–1972.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Unsupervised machine translation using monolingual corpora only. In *International Conference on Learning Representations (ICLR)*.

Anton Leuski, Chin-Yew Lin, Liang Zhou, Ulrich Germann, Franz Josef Och, and Eduard Hovy. 2003. Cross-lingual c* st* rd: English access to Hindi information. *ACM Transactions on Asian Language Information Processing (TALIP)*, 2(3):245–269.

Piji Li, Wai Lam, Lidong Bing, and Zihao Wang. 2017. Deep recurrent generative decoder for abstractive text summarization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2091–2100.

Jung-Min Lim, In-Su Kang, and Jong-Hyeok Lee. 2004. Multi-document summarization using cross-language texts. In *Proceedings of the Fourth NTCIR Workshop on Research in Information Access Technologies Information Retrieval, Question Answering and Summarization (NTCIR)*.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81.

Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Constantin Orasan and Oana Andreea Chiorean. 2008. Evaluation of a cross-lingual Romanian-English multi-document summariser. In *Proceedings of Language Resources and Evaluation Conference (LREC)*.

Jessica Ouyang, Boya Song, and Kathy McKeown. 2019. A robust abstractive system for cross-lingual summarization. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 2025–2031.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1073–1083.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1715–1725.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5998–6008.

Xiaojun Wan. 2011. Using bilingual information for cross-language document summarization. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1546–1555.

Xiaojun Wan, Huiying Li, and Jianguo Xiao. 2010. Cross-language document summarization based on machine translation quality prediction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 917–926.

Jin-ge Yao, Xiaojun Wan, and Jianguo Xiao. 2015. Phrase-based compressive cross-language summarization. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, pages 118–127.

Jiajun Zhang, Yu Zhou, and Chengqing Zong. 2016. Abstractive cross-language summarization via translation model enhanced predicate argument structure fusing. *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP)*, 24(10):1842–1853.

Junnan Zhu, Haoran Li, Tianshang Liu, Yu Zhou, Jiajun Zhang, and Chengqing Zong. 2018. MSMO: Multimodal summarization with multimodal output. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4154–4164.

# Clickbait? Sensational Headline Generation with Auto-tuned Reinforcement Learning

**Peng Xu, Chien-Sheng Wu, Andrea Madotto** and **Pascale Fung**
Center for Artificial Intelligence Research (CAiRE)
Department of Electronic and Computer Engineering
The Hong Kong University of Science and Technology, Clear Water Bay
[pxuab,cwuak,amadotto]@connect.ust.hk, pascale@ece.ust.hk

## Abstract

Sensational headlines are headlines that capture people's attention and generate reader interest. Conventional abstractive headline generation methods, unlike human writers, do not optimize for maximal reader attention. In this paper, we propose a model that generates sensational headlines without labeled data. We first train a sensationalism scorer by classifying online headlines with many comments ("clickbait") against a baseline of headlines generated from a summarization model. The score from the sensationalism scorer is used as the reward for a reinforcement learner. However, maximizing the noisy sensationalism reward will generate unnatural phrases instead of sensational headlines. To effectively leverage this noisy reward, we propose a novel loss function, Auto-tuned Reinforcement Learning (ARL), to dynamically balance reinforcement learning (RL) with maximum likelihood estimation (MLE). Human evaluation shows that 60.8% of samples generated by our model are sensational, which is significantly better than the Pointer-Gen baseline (See et al., 2017) and other RL models.

## 1 Introduction

Headline generation is the process of creating a headline-style sentence given an input article. The research community has been regarding the task of headline generation as a summarization task (Shen et al., 2017a), ignoring the fundamental differences between headlines and summaries. While summaries aim to contain most of the important information from the articles, headlines do not necessarily need to. Instead, a good headline needs to capture people's attention and serve as an irresistible invitation for users to read through the article. For example, the headline "$2 Billion Worth of Free Media for Trump", which gives only an intriguing hint, is considered better than the summarization style headline "Measuring Trump's Media Dominance" [1], as the former gets almost three times the readers as the latter. Generating headlines with many clicks is especially important in this digital age, because many of the revenues of journalism come from online advertisements and getting more user clicks means being more competitive in the market. However, most existing websites [2] naively generate sensational headlines using only keywords or templates. Instead, this paper aims to learn a model that generates sensational headlines based on an input article without labeled data.

To generate sensational headlines, there are two main challenges. Firstly, there is a lack of sensationalism scorer to measure how sensational a headline is. Some researchers have tried to manually label headlines as clickbait or non-clickbait (Chakraborty et al., 2016; Potthast et al., 2018). However, these human-annotated datasets are usually small and expensive to collect. To capture a large variety of sensationalization patterns, we need a cheap and easy way to collect a large number of sensational headlines. Thus, we propose a distant supervision strategy to collect a sensationalism dataset. We regard headlines receiving lots of comments as sensational samples and the headlines generated by a summarization model as non-sensational samples. Experimental results show that by distinguishing these two types of headlines, we can partially teach the model a sense of being sensational.

Secondly, after training a sensationalism scorer on our sensationalism dataset, a natural way to generate sensational headlines is to maximize the sensationalism score using reinforcement learn-

[1] https://www.nytimes.com/2016/06/13/insider/which-headlines-attract-most-readers.html?module=inline

[2] http://www.contentrow.com/tools/link-bait-title-generator/



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3065–3075,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

ing (RL). However, the following shows an example of a RL model maximizing the sensationalism score by generating a very unnatural sentence, while its sensationalism scorer gave a very high score of 0.99996: 十个可穿戴产品的设计原则这消息消息可惜说明 *Ten design principles for wearable devices, this message message pity introduction.* This happens because the sensationalism scorer can make mistakes and RL can generate unnatural phrases which fools our sensationalism scorer. Thus, how to effectively leverage RL with noisy rewards remains an open problem. To deal with the noisy reward, we introduce Auto-tuned Reinforcement Learning (ARL). Our model automatically tunes the ratio between MLE and RL based on how sensational the training headline is. In this way, we effectively take advantage of RL with a noisy reward to generate headlines that are both sensational and fluent.

The major contributions of this paper are as follows: 1) To the best of our knowledge, we propose the first-ever model that tackles the sensational headline generation task with reinforcement learning techniques. 2) Without human-annotated data, we propose a distant supervision strategy to train a sensationalism scorer as a reward function.3) We propose a novel loss function, Auto-tuned Reinforcement Learning, to give dynamic weights to balance between MLE and RL. Our code will be released . [3]

## 2 Sensationalism Scorer

To evaluate the sensationalism intensity score $\alpha_{\text{sen}}$ of a headline, we collect a sensationalism dataset and then train a sensationalism scorer. For the sensationalism dataset collection, we choose headlines with many comments from popular online websites as positive samples. For the negative samples, we propose to use the generated headlines from a sentence summarization model. Intuitively, the summarization model, which is trained to preserve the semantic meaning, will lose the sensationalization ability and thus the generated negative samples will be less sensational than the original one, similar to the obfuscation of style after back-translation (Prabhumoye et al., 2018). For example, an original headline like "一趟挣10万？铁总增开申通、顺丰专列" (One trip to earn 100 thousand? China Railway opens new Shentong and Shunfeng special lines) will become "中铁总将增开京广两列快递专列" (China Railway opens two special lines for express) from the baseline model, which loses the sensational phrases of "一趟挣10万？" (One trip to earn 100 thousand?) . We then train the sensationalism scorer by classifying sensational and non-sensational headlines using a one-layer CNN with a binary cross entropy loss $L_{\text{sen}}$. Firstly, 1-D convolution is used to extract word features from the input embeddings of a headline. This is followed by a ReLU activation layer and a max-pooling layer along the time dimension. All features from different channels are concatenated together and projected to the sensationalism score by adding another fully connected layer with sigmoid activation. Binary cross entropy is used to compute the loss $L_{\text{sen}}$.

### 2.1 Training Details and Dataset

For the CNN model, we choose filter sizes of 1, 3, and 5 respectively. Adam is used to optimize $L_{sen}$ with a learning rate of 0.0001. We set the embedding size as 300 and initialize it from Qiu et al. (2018) trained on the Weibo corpus with word and character features. We fix the embeddings during training.

For dataset collection, we utilize the headlines collected in Qin et al. (2018); Lin et al. (2019a) from Tencent News, one of the most popular Chinese news websites, as the positive samples. We follow the same data split as the original paper. As some of the links are not available any more, we get 170,754 training samples and 4,511 validation samples. For the negative training samples collection, we randomly select generated headlines from a pointer generator (See et al., 2017) model trained on LCSTS dataset (Hu et al., 2015) and create a balanced training corpus which includes 351,508 training samples and 9,022 validation samples. To evaluate our trained classifier, we construct a test set by randomly sampling 100 headlines from the test split of LCSTS dataset and the labels are obtained by 11 human annotators. Annotations show that 52% headlines are labeled as positive and 48% headlines as negative by majority voting (The detail on the annotation can be found in Section 3.6).

### 2.2 Results and Discussion

Our classifier achieves 0.65 accuracy and 0.65 averaged F1 score on the test set while a random classifier would only achieve 0.50 accuracy and

[3] https://github.com/HLTCHKUST/sensational_headline

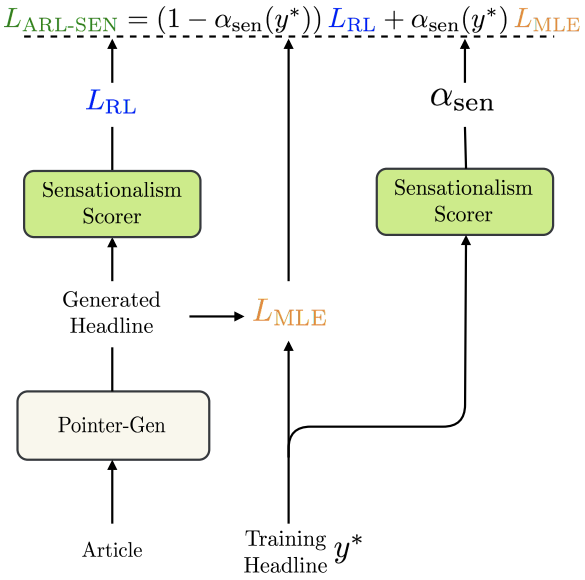


Figure 1: The loss function of Auto-tuned Reinforcement Learning is a weighted sum of $L_{\text{RL}}$ and $L_{\text{MLE}}$, where the weight is decided by our sensationalism scorer.

0.50 averaged F1 score. This confirms that the predicted sensationalism score can partially capture the sensationalism of headlines. On the other hand, a more natural choice is to take headlines with few comments as negative examples. Thus, we train another baseline classifier on a crawled balanced sensationalism corpus of 84k headlines where the positive headlines have at least 28 comments and the negative headlines have less than 5 comments. However, the results on the test set show that the baseline classifier gets 60% accuracy, which is worse than the proposed classifier (which achieves 65%). The reason could be that the balanced sensationalism corpus are sampled from different distributions from the test set and it is hard for the trained model to generalize. Therefore, we choose the proposed one as our sensationalism scorer. Therefore, our next challenge is to show that how to leverage this noisy sensationalism reward to generate sensational headlines.

## 3 Sensational Headline Generation

Our sensational headline generation model takes an article as input and output a sensational headline. The model consists of a Pointer-Gen headline generator and is trained by ARL. The diagram of ARL can be found in Figure 1.

We denote the input article as $x = \{x_1, x_2, x_3, \cdots, x_M\}$, and the corresponding headline as $y^* = \{y_1^*, y_2^*, y_3^*, \cdots, y_T^*\}$, where $M$ is the number of tokens in an article and $T$ is the number of tokens in a headline.

### 3.1 Pointer-Gen Headline Generator

We choose Pointer Generator (Pointer-Gen) (See et al., 2017), a widely used summarization model, as our headline generator for its ability to copy words from the input article. It takes a news article as input and generates a headline. Firstly, the tokens of each article, $\{x_1, x_2, x_3, \cdots, x_M\}$, are fed into the encoder one-by-one and the encoder generates a sequence of hidden states $h_i$. For each decoding step $t$, the decoder receives the embedding for each token of a headline $y_t$ as input and updates its hidden states $s_t$. An attention mechanism following Luong et al. (2015) is used:

$$e_i^t = v^T \tanh(W_h h_i + W_s s_t + b_{attn}) \quad (1)$$

$$a^t = \text{softmax}(e^t) \quad (2)$$

$$h_t^* = \sum\nolimits_i a_i^t h_i \quad (3)$$

where $v$, $W_h$, $W_s$, and $b_{attn}$ are the trainable parameters and $h_t^*$ is the context vector. $s_t$ and $h_t^*$ are then combined to give a probability distribution over the vocabulary through two linear layers:

$$o_t = V([s_t, h_t^*]) + b \quad (4)$$

$$P_{\text{voc}}(w) = \text{softmax}(V^{'} o_t + b^{'}) \quad (5)$$

where $V$, $b$, $V^{'}$, and $b^{'}$ are trainable parameters. We use a pointer generator network to enable our model to copy rare/unknown words from the input article, giving the following final word probability:

$$p_{\text{gen}} = \sigma(w_{h^*}^T h_t^* + w_s^T s_t + w_x^T x^t + b_{ptr}) \quad (6)$$

$$P(w) = p_{\text{gen}} P_{\text{voc}}(w) + (1 - p_{\text{gen}}) \sum_{i:x_i=w} a_i^t \quad (7)$$

where $x^t$ is the embedding of the input word of the decoder, $w_{h^*}^T$, $w_s^T$, $w_x^T$, and $b_{ptr}$ are trainable parameters, and $\sigma$ is the sigmoid function.

### 3.2 Training Methods

We first briefly introduce MLE and RL objective functions, and a naive way to mix these two by a hyper-parameter $\lambda$. Then we point out the challenge of training with noisy reward, and propose ARL to address this issue.

#### 3.2.1 MLE and RL

A headline generation model can be trained with MLE, RL or a combination of MLE and RL. MLE

training is to minimize the negative log likelihood of the training headlines. We feed $y^*$ into the decoder word by word and maximize the likelihood of $y^*$. The loss function for MLE becomes

$$L_{\text{MLE}} = -\frac{1}{T}\sum_{t=1}^{T} \log P(y_t^*) \tag{8}$$

For RL training, we choose the REINFORCE algorithm (Williams, 1992). In the training phase, after encoding an article, a headline $y^s = \{y_1^s, y_2^s, y_3^s, \cdots, y_T^s\}$ is obtained by sampling from $P(w)$ from our generator, and then a reward of sensationalism or ROUGE(RG) is calculated.

$$R_{\text{rg}} = \text{RG}(y^s, y^*) \tag{9}$$

$$R_{\text{sen}} = \alpha_{\text{sen}}(y^s) \tag{10}$$

We use the baseline reward $\hat{R}_t$ to reduce the variance of the reward, similar to Ranzato et al. (2016). To elaborate, a linear model is deployed to estimate the baseline reward $\hat{R}_t$ based on $t$-th state $o_t$ for each timestep $t$. The parameters of the linear model are trained by minimizing the mean square loss between $R$ and $\hat{R}_t$:

$$\hat{R}_t = W_r o_t + b_r \tag{11}$$

$$L_b = \frac{1}{T}\sum_{t=1}^{T} |R - \hat{R}_t|^2 \tag{12}$$

where $W_r$ and $b_r$ are trainable parameters. To maximize the expected reward, our loss function for RL becomes

$$L_{\text{RL}} = -\frac{1}{T}\sum_{t=1}^{T} (R - \hat{R}_t) \log P(w_t) \tag{13}$$

A naive way to mix these two objective functions using a hyper-parameter $\lambda$ has been successfully incorporated in the summarization task (Paulus et al., 2018). It includes the MLE training as a language model to mitigate the readability and quality issues in RL. The mixed loss function is shown as follows:

$$L_{\text{RL-}*} = \lambda L_{\text{RL}} + (1-\lambda) L_{\text{MLE}} \tag{14}$$

where $*$ is the reward type. Usually $\lambda$ is large, and Paulus et al. (2018) used 0.9984.

#### 3.2.2 Auto-tuned Reinforcement Learning

Applying the naive mixed training method using sensationalism score as the reward is not obvious/trivial in our task. The main reason is that our sensationalism reward is notably more noisy and more fragile than the ROUGE-L reward or abstractive reward used in the summarization task (Paulus et al., 2018; Kryściński et al., 2018). A higher ROUGE-L F1 reward in summarization indicates higher overlapping ratio between generation and true summary statistically, but our sensationalism reward is a learned score which is fragile to be fooled with unnatural samples.

To effectively train the model with RL under noisy sensationalism reward, our idea is to balance RL with MLE. However, we argue that *the weighted ratio between MLE and RL should be sample-dependent,* instead of being fixed for all training samples as in Paulus et al. (2018); Kryściński et al. (2018). The reason is that, RL and MLE have inconsistent optimization objectives. When the training headline is non-sensational, MLE training will encourage our model to imitate the training headline (thus generating *non-sensational* headlines), which counteracts the effects of RL training to generate *sensational* headlines.

The sensationalism score is, therefore, used to give dynamic weight to MLE and RL. Our ARL loss function becomes:

$$L_{\text{ARL-SEN}} = (1 - \alpha_{\text{sen}}(y^*))\, L_{\text{RL}} + \alpha_{\text{sen}}(y^*)\, L_{\text{MLE}} \tag{15}$$

If $\alpha_{\text{sen}}(y^*)$ is high, meaning the training headline is sensational, our loss function encourages our model to imitate the sample more using the MLE training. If $\alpha_{\text{sen}}(y^*)$ is low, our loss function replies on RL training to improve the sensationalism. Note that the weight $\alpha_{\text{sen}}(y^*)$ is different from our sensationalism reward $\alpha_{\text{sen}}(y^s)$ and we call the loss function Auto-tuned Reinforcement Learning, because the ratio between MLE and RL are well "tuned" towards different samples.

### 3.3 Dataset

We use LCSTS (Hu et al., 2015) as our dataset to train the summarization model. The dataset is collected from the Chinese microblogging website Sina Weibo. It contains over 2 million Chinese short texts with corresponding headlines given by the author of each text. The dataset is split into 2,400,591 samples for training, 10,666 samples for validation and 725 samples for testing. We tokenize each sentence with Jieba [4] and a vocabulary size of 50000 is saved.

[4] https://github.com/fxsjy/jieba

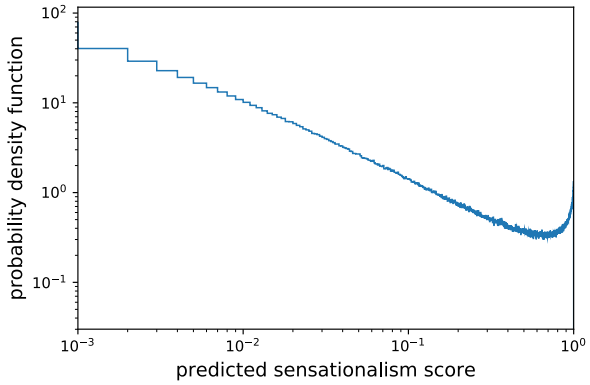


Figure 2: The probability density function (pdf) of predicted sensationalism score in log scale. Low sensationalism score has much higher probability density.

### 3.4 Baselines and Our Models

We experiment and compare with the following models. **Pointer-Gen** is the baseline model trained by optimizing $L_{\text{MLE}}$ in Equation 8.

**Pointer-Gen+Pos** is the baseline model by training Pointer-Gen only on positive examples whose sensationalism score is larger than 0.5

**Pointer-Gen+Same-FT** is the model which finetunes Pointer-Gen on the training samples whose sensationalism score is larger than 0.1

**Pointer-Gen+Pos-FT** is the model which finetunes Pointer-Gen on the training samples whose sensationalism score is larger than 0.5

**Pointer-Gen+RL-ROUGE** is the baseline model trained by optimizing $L_{\text{RL-ROUGE}}$ in Equation 14, with ROUGE-L (Lin, 2004) as the reward.

**Pointer-Gen+RL-SEN** is the baseline model trained by optimizing $L_{\text{RL-SEN}}$ in Equation 14, with $\alpha_{\text{sen}}$ as the reward.

**Pointer-Gen+ARL-SEN** is our model trained by optimizing $L_{\text{ARL-SEN}}$ in Equation 15, with $\alpha_{\text{sen}}$ as the reward.

**Test set** is the headlines from the test set.

Note that we didn't compare to Pointer-Gen+ARL-ROUGE as it is actually Pointer-GEN. Recall that $\alpha_{\text{sen}}(y^*)$ in Equation 15 measures how good (based on reward function) is $y^*$. Then the loss function for Pointer-Gen+ARL-ROUGE will be

$$(1-\text{RG}(y^*,y^*))L_{\text{RL}}+\text{RG}(y^*,y^*)L_{\text{MLE}}=L_{\text{MLE}}$$

We also tried text style transfer baseline (Shen et al., 2017b), but the generated headlines were very poor (many unknown words and irrelevant).

### 3.5 Training Details

**MLE training:** An Adam optimizer is used with the learning rate of 0.0001 to optimize $L_{\text{MLE}}$. The batch size is set as 128 and a one-layer, bi-directional Long Short-Term Memory (bi-LSTM) model with 512 hidden sizes and a 350 embedding size is utilized. Gradients with the l2 norm larger than 2.0 are clipped. We stop training when the ROUGE-L f-score stops increasing.

**Hybrid training:** An Adam optimizer with a learning rate of 0.0001 is used to optimize $L_{\text{RL-*}}$ (Equation 14) and $L_{\text{ARL-SEN}}$ (Equation 15). When training Pointer-Gen+RL-ROUGE, the best $\lambda$ is chosen based on the ROUGE-L score on the validation set. In our experiment, $\lambda$ is set as 0.95. An Adam optimizer with a learning rate of 0.001 is used to optimize $L_b$. When training Pointer-Gen+ARL-SEN, we don't use the full LCSTS dataset, but only headlines with a sensationalism score larger than 0.1 as we observe that Pointer-Gen+ARL-SEN will generate a few unnatural phrases when using full dataset. We believe the reason is the high ratio of RL during training. Figure 2 shows that the probability density near 0 is very high, meaning that in each batch, many of the samples will have a very low sensationalism score. On expectation, each sample will receive 0.239 MLE training and 0.761 RL training. This leads to RL dominanting the loss. Thus, we propose to filter samples with a minimum sensationalism score with 0.1 and it works very well. For Pointer-Gen+RL-SEN, we also set the minimum sensationalism score as 0.1, and $\lambda$ is set as 0.5 to remove unnatural phrases, making a fair comparison to Pointer-Gen+ARL-SEN.

We stop training Pointer-Gen+Same-FT, Pointer-Gen+Pos-FT, Pointer-Gen+RL-SEN and Pointer-Gen+ARL-SEN, when $\alpha_{\text{sen}}$ stops increasing on the validation set. Beam-search with a beam size of 5 is adopted for decoding in all models.

### 3.6 Evaluation Metrics

We briefly describe the evaluation metrics below.

**ROUGE:** ROUGE is a commonly used evaluation metric for summarization. It measures the N-gram overlap between generated and training headlines. We use it to evaluate the *relevance* of generated headlines. The widely used pyrouge [5] toolkit is used to calculate ROUGE-1 (RG-1), ROUGE-2

[5] https://github.com/bheinzerling/pyrouge

| | RG-1 | RG-2 | RG-L |
|---|---|---|---|
| character-based preprocessing | | | |
| RNN context (Hu et al., 2015) | 29.90 | 17.40 | 27.20 |
| COPYNET (Gu et al., 2016) | 34.40 | 21.60 | 31.30 |
| word-based preprocessing | | | |
| RNN context (Hu et al., 2015) | 26.80 | 16.10 | 24.10 |
| COPYNET (Gu et al., 2016) | 35.00 | 22.30 | 32.00 |
| Pointer-Gen | 34.51 | 22.21 | 31.68 |
| Pointer-Gen+Pos | 28.51 | 16.53 | 25.56 |
| Pointer-Gen+Same-FT | 34.60 | 22.00 | 31.48 |
| Pointer-Gen+Pos-FT | 30.92 | 18.76 | 28.02 |
| Pointer-Gen+RL-ROUGE | **36.26** | **23.48** | **33.21** |
| Pointer-Gen+RL-SEN | 35.06 | 22.37 | 31.91 |
| Pointer-Gen+ARL-SEN | 34.28 | 21.34 | 30.80 |

Table 1: Our implementation achieves similar performance to the RNN context and COPYNET. Pointer-Gen+ARL-SEN achieves good summarization performance even though it is optimized for the sensational reward. It shows its ability to summarize.

(RG-2), and ROUGE-L (RG-L).

**Human evaluation:** We randomly sample 50 articles from the test set and send the generated headlines from all models and corresponding headlines in the test set to human annotators. We evaluate the sensationalism and fluency of the headlines by setting up two independent human annotation tasks. We ask 10 annotators to label each headline for each task. For the sensationalism annotation, each annotator is asked one question, "Is the headline sensational?", and he/she has to choose either 'yes' or 'no'. The annotators were not told which system the headline is from. The process of distributing samples and recruiting annotators is managed by Crowdflower.[6] After annotation, we define the sensationalism score as the proportion of annotations on all generated headlines from one model labeled as 'yes'. For the fluency annotation, we repeat the same procedure as for the sensationalism annotation, except that we ask each annotator the question "Is the headline fluent?" We define the fluency score as the proportion of annotations on all headlines from one specific model labeled as 'yes'. We put human annotation instructions in the supplemental material.

## 4 Results

We first compare all four models, Pointer-Gen, Pointer-Gen-RL+ROUGE, Pointer-Gen-RL-SEN, and Pointer-Gen-ARL-SEN, to existing models with ROUGE in Table 1 to establish that our model produces relevant headlines and we leave the sensationalism for human evaluation. Note that we only compare our models to commonly used strong summarization baselines, to validate that our implementation achieves comparable performance to existing work. In our implementation, Pointer-Gen achieves a 34.51 RG-1 score, 22.21 RG-2 score, and 31.68 RG-L score, which is similar to the results of Gu et al. (2016). Pointer-Gen+ARL-SEN, although optimized for the sensationalism reward, achieves similar performance to our Pointer-Gen baseline, which means that Pointer-Gen+ARL-SEN still keeps its summarization ability. An example of headlines generated from different models in Table 2 shows that Pointer-Gen and Pointer-Gen+RL-ROUGE learns to summarize the main point of the article: "The Nikon D600 camera is reported to have black spots when taking photos". Pointer-Gen+RL-SEN

[6]https://www.figure-eight.com/

| |
|---|
| **Article**: 昨天的央视315晚会上，尼康D600相机被曝拍摄时会出现黑色斑点，反复修理也无果，而尼康竟把责任推给了雾霾，拒绝换机或退机。对此上海工商部门昨晚连夜梳理了近年对尼康的投诉案，今天上午前往位于黄浦区的尼康公司进一步调查。新民网<br>At the CCTV 315 party yesterday, the Nikon D600 camera is reported to have black spots when taking photos. Repeated repairs gave no results, and Nikon actually attributes the damage to the smog, refusing exchange or return. The Shanghai Industrial and Commercial Department collected the recent complaints against Nikon last night, and went to Nikon Company in Huangpu District for further investigation this morning. Xinmin Net |
| **Pointer-Gen:** 尼康D600相机被曝拍摄时会出现黑色斑点 The Nikon D600 camera is reported to have black spots when taking photos |
| **Pointer-Gen+Same-FT:** 尼康D600被曝拍出雾霾尼康已介入调查The Nikon D600 camera is reported to have smog Nikon started investigation |
| **Pointer-Gen+Pos-FT:** 尼康投诉央视315晚会尼康投诉被曝Nikon complains CCTV 315 night, Nikon is reported to be complained |
| **Pointer-Gen+RL-ROUGE**: 尼康D600相机被曝拍摄时会出现黑色斑点 The Nikon D600 camera is reported to have black spots when taking photos |
| **Pointer-Gen+RL-SEN**: 尼康D600被曝拍出奇葩事尼康把责任推给雾霾 The Nikon D600 camera is reported to have something strange and attributes the damage to the smog. |
| **Pointer-Gen+ARL-SEN (Ours)**: 摊上大事了！ 尼康D600被指拍出"黑点" **In Serious Trouble!** The Nikon D600 camera is reported to have black spots when taking photos |
| **Test set headline**: 尼康称黑点怪雾霾上海工商今上门追查 Nikon attributes black spots to the smog, and Shanghai Industrial and Commercial Department start investigation today. |

Table 2: Generated Chinese headlines from different models. Our model (Pointer-Gen+ARL-SEN) sensationalized the headline with the phrase "In Serious Trouble!".

| Model | sensationalism | fluency |
|---|---|---|
| Pointer-Gen | 42.6%* | 80% |
| Pointer-Gen-Pos | 40.2%* | 59%* |
| Pointer-Gen+Same-FT | 47.6%* | 75.6% |
| Pointer-Gen+Pos-FT | 47.8%* | 71.4% |
| Pointer-Gen+RL-ROUGE | 45.2%* | 80% |
| Pointer-Gen+RL-SEN | 51.8%* | 79.4% |
| Pointer-Gen+ARL-SEN | **60.8%** | 79.4% |
| Test set | 57.8% | **80.6%** |

Table 3: Comparison of sensationalism score and fluency score between different models. Pointer-Gen+ARL-SEN achieves the best performance among all models in sensationalism score. * indicates Pointer-Gen+ARL-SEN is statistically significantly better than the corresponding model.

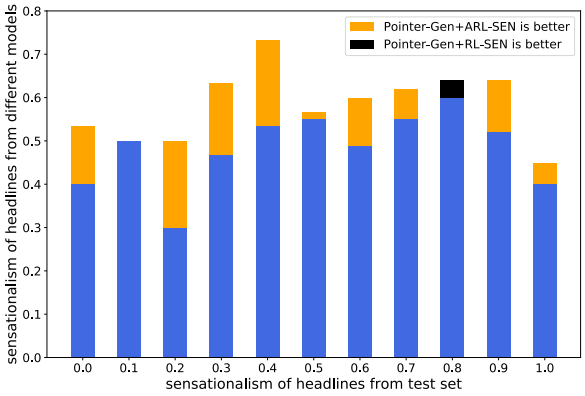


Figure 3: Comparison of sensationalism score between Pointer-Gen+ARL-SEN and Pointer-Gen+RL-SEN for different test set headlines. The blue bars denote the smaller scores between the two models. Pointer-Gen+ARL-SEN achieves better performance on most cases. Greater improvement is achieved when the test set headline is non-sensational.

makes the headline more sensational by blaming Nikon for attributing the damage to the smog. Pointer-Gen+ARL-SEN generates the most sensational headline by exaggerating the result "Getting a serious trouble!" to maximize user's attention.

We then compare different models using the sensationalism score in Table 3. The Pointer-Gen baseline model achieves a 42.6% sensationalism score, which is the minimum that a typical summarization model achieves. By filtering out low-sensational headlines, Pointer-Gen+Same-FT and Pointer-Gen+Pos-FT achieves higher sensationalism scores, which implies the effectiveness of our sensationalism scorer. Our Pointer-Gen+ARL-SEN model achieves the best performance of 60.8%. This is an absolute improvement of 18.2% over the Pointer-Gen baseline. The Chi-square test on the results confirms that Pointer-Gen+ARL-SEN is statistically significantly more sensational than all the other baseline models, with the largest p-value less than 0.01. Also, we find that the test set headlines achieves 57.8% sensationalism score, much larger than Pointer-Gen baseline, which also supports our intuition that generated headlines will be less sensational than the original one. On the other hand, we found that Pointer-Gen+Pos is much worse than other baselines. The reason is that training on sensational samples alone discards around 80% of the whole training set that is also helpful for maintaining relevance and a good language model. It shows the necessity of using RL.

In addition, both Pointer-Gen+RL-SEN and Pointer-Gen+ARL-SEN, which use the sensationalism score as the reward, obtain statistically better results than Pointer-Gen+RL-ROUGE and Pointer-Gen, with a p-value less than 0.05 by a Chi-square test. This result shows the effectiveness of RL to generate more sensational headlines. The reason is that even though our noisy classifier could also learn to classify domains, the generator during RL training is not allowed to increase the reward by shifting domains but encouraged to generate more sensational headlines, due to the consistency constraint on the domains of the headline and the article. Furthermore, Poiner-Gen+ARL-SEN gets better performance than Pointer-Gen+RL-SEN, which confirms the superiority of the ARL loss function. We also visualize in Figure 3 a comparison between Pointer-Gen+ARL-SEN and Pointer-Gen+RL-SEN according to how sensational the test set headlines are. The blue bars denote the smaller scores between the two models. For example, if the blue bar is 0.6, it means that the worse model between Pointer-Gen+RL-SEN and Pointer-Gen+ARL-SEN achieves 0.6. And the color of orange/black further indicates the better model and its score. We find that Pointer-Gen+ARL-SEN outperforms Pointer-Gen+RL-SEN for most cases. The improvement is higher when the test set headlines are not sensational (the sensationalism score is less than 0.5), which may be attributed to the higher ratio of RL training on non-sensational headlines.

Apart from the sensationalism evaluation, we measure the fluency of the headlines generated from different models. Fluency scores in Table

| | |
|---|---|
| Question | **Pointer-Gen**: 眺望：印度经济复苏的陷阱<br>View: the trap of economic recovery in India<br>**Pointer-Gen+ARL-SEN**: 印度或将成为下一个中国?<br>**Will** India **become** the next China**?** |
| | **Pointer-Gen**: 今天理论导报的主要内容有……<br>The main content of today's theoretical report is<br>**Pointer-Gen+ARL-SEN**: 如何做一名焦裕禄式的县委书记?<br>**How to** be a county party secretary like JIAO Yulu**?** |
| Creating Curiosity Gap | **Pointer-Gen**: 10种方法帮你避免的10个小窍门<br>10 tricks to help you avoid<br>**Pointer-Gen+ARL-SEN**: 10个让你意想不到的领导力法则<br>10 laws of leadership that you **cannot think of** |
| | **Pointer-Gen:** 王林的暴富之路　WANG Lin's path to sudden wealth<br>**Pointer-Gen+ARL-SEN**: "气功大师"王林的暴富之路：凭借5个生财之道<br>"The Qigong Master" WANG Lin's path to sudden wealth: leveraging **5 ways to make money** |
| Highlighting Numbers | **Pointer-Gen**: 北京市集成电路促进基金就位<br>The Integrated Circuit Promotion Fund in Beijing is ready<br>**Pointer-Gen+ARL-SEN**: 500亿巨资驰援国家大基金或已就位<br>**A huge capital sum of 50 billion** is ready to support the national big fund |
| | **Pointer-Gen**: 陈光标冰桶挑战陈光标承认造假<br>CHEN Guangbiao admits that he cheated in the freezing water challenge<br>**Pointer-Gen+ARL-SEN**: 陈光标回应"造假"：有人超越我就捐款100万<br>CHEN Guangbiao responded to "cheating": if someone can do better, I will donate **a million RMB** |
| Emotional Words | **Pointer-Gen**: 俞永福：搜狗360还没停<br>YU Yongfu: Sougou and 360 have not stopped<br>**Pointer-Gen+ARL-SEN**: 俞永福微博辱骂UC：太恶心了<br>YU Youfu abused UC in microblog: it is too **disgusting** |
| | **Pointer-Gen**: 保定楼市""一天涨3000""<br>"3000 increase in one day" for Baoding housing market<br>**Pointer-Gen+ARL-SEN**: 保定楼市"一天涨3000 简直疯了"<br>"3000 increase in one day" for Baoding housing market, **this is crazy** |
| Empathizing | **Pointer-Gen**: 女性购物的五大特征　Five characteristics of ladies shopping<br>**Pointer-Gen+ARL-SEN**: 女性购物的5个忠告：你中枪了吗?<br>5 warnings for ladies shopping: Have **you** been targeted? |
| | **Pointer-Gen**: 智能手表将开始拥有智能手机功能<br>Smart watches will start to have smartphone features<br>**Pointer-Gen+ARL-SEN**: 关于智能手表，你应该知道的事！<br>What **you** should know about smart watches! |

Table 4: Different sensationalization strategies Pointer-Gen+ARL-SEN learns.

3 show that Pointer-Gen+RL-SEN and Pointer-Gen+ARL-SEN achieve comparable fluency performance to Pointer-Gen and Pointer-Gen+RL-ROUGE. Test set headlines achieve the best performance among all models, but the difference is not statistically significant. Also, we observe that fine-tuning on sensational headlines will hurt the performance, both in sensationalism and fluency.

After manually checking the outputs, we observe that our model is able to generate sensational headlines using diverse sensationalization strategies. These strategies include, but are not limited to, creating a curiosity gap, asking questions, highlighting numbers, being emotional and emphasizing the user. Examples can be found in Table 4.

## 5 Related Work

Our work is related to summarization tasks. An encoder-decoder model was first applied to two sentence-level abstractive summarization tasks on the DUC-2004 and Gigaword datasets (Rush et al., 2015). This model was later extended by selective encoding (Zhou et al., 2017), a coarse to fine approach (Tan et al., 2017b), minimum risk training (Shen et al., 2017a), and topic-aware models (Wang et al., 2018). As long summaries were recognized as important, the CNN/Daily Mail dataset was used in Nallapati et al. (2016). Graph-based attention (Tan et al., 2017a), pointer-generator with coverage loss (See et al., 2017) are further developed to improve the generated summaries. Celikyilmaz et al. (2018) proposed deep communicating agents for representing a long document

for abstractive summarization. In addition, many papers (Nallapati et al., 2017; Zhou et al., 2018b; Zhang et al., 2018a) use extractive methods to directly select sentences from articles. However, none of these work considered the sensationalism of generated outputs.

RL is also gaining popularity as it can directly optimize non-differentiable metrics (Pasunuru and Bansal, 2018; Venkatraman et al., 2015; Xu and Fung, 2019). Paulus et al. (2018) proposed an intra-decoder model and combined RL and MLE to deal with summaries with bad qualities. RL has also been explored with generative adversarial networks (GANs) (Yu et al., 2017). Liu et al. (2018) applied GANs on summarization task and achieved better performance. Niu and Bansal (2018) tackles the problem of polite generation with politeness reward. Our work is different in that we propose a novel function to balance RL and MLE.

Our task is also related to text style transfer. Implicit methods (Shen et al., 2017b; Fu et al., 2018; Prabhumoye et al., 2018) transfer the styles by separating sentence representations into content and style, for example using back-translation(Prabhumoye et al., 2018). However, these methods cannot guarantee the content consistency between the original sentence and transferred output (Xu et al., 2018a). Explicit methods (Zhang et al., 2018b; Xu et al., 2018a) transfer the style by directly identifying style related keywords and modifying them. However, sensationalism is not always restricted to keywords, but the full sentence. By leveraging small human labeled English dataset, clickbait detection has been well investigated (Chakraborty et al., 2016; Shu et al., 2018; Potthast et al., 2018). However, these human labeled dataset are not available for other languages, such as Chinese.

Modeling sensationalism is also related to modeling emotion. Emotion has been well investigated in both word level(Tang et al., 2016; Xu et al., 2018b) and sentence level(Felbo et al., 2017; Winata et al., 2019, 2018; Park et al., 2018; Lee et al., 2019). It has also been considered an important factor in engaging interactive systems(Lin et al., 2019b; Winata et al., 2017; Zhou et al., 2018a). Although we observe that sensational headlines contain emotion, it is still not clear which emotion and how emotions will influence the sensationalism.

## 6 Conclusion and Future Work

In this paper, we propose a model that generates sensational headlines without labeled data using Reinforcement Learning. Firstly, we propose a distant supervision strategy to train the sensationalism scorer. As a result, we achieve 65% accuracy between the predicted sensationalism score and human evaluation. To effectively leverage this noisy sensationalism score as the reward for RL, we propose a novel loss function, ARL, to automatically balance RL with MLE. Human evaluation confirms the effectiveness of both our sensationalism scorer and ARL to generate more sensational headlines. Future work can be improving the sensationalism scorer and investigating the applications of dynamic balancing methods between RL and MLE in textGAN(Yu et al., 2017). Our work also raises the ethical questions about generating sensational headlines, which can be further explored.

## Acknowledgments

Thanks to ITS/319/16FP of Innovation Technology Commission, HKUST 16248016 of Hong Kong Research Grants Council for funding. In addition, we thank Zhaojiang Lin for helpful discussion and Yan Xu, Zihan Liu for the data collection.

## References

Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep communicating agents for abstractive summarization. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 1662–1675.

Abhijnan Chakraborty, Bhargavi Paranjape, Sourya Kakarla, and Niloy Ganguly. 2016. Stop clickbait: Detecting and preventing clickbaits in online news media. In *Proceedings of the 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 9–16. IEEE Press.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1616–1626.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2018. Style transfer in text: Exploration and evaluation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1631–1640.

Baotian Hu, Qingcai Chen, and Fangze Zhu. 2015. Lcsts: A large scale chinese short text summarization dataset. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1967–1972.

Wojciech Kryściński, Romain Paulus, Caiming Xiong, and Richard Socher. 2018. Improving abstraction in text summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1808–1817.

Nayeon Lee, Zihan Liu, and Pascale Fung. 2019. Team yeon-zi at semeval-2019 task 4: Hyperpartisan news detection by de-noising weakly-labeled data. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 1052–1056.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Zhaojiang Lin, Genta Indra Winata, and Pascale Fung. 2019a. Learning comment generation by leveraging user-generated data. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7225–7229. IEEE.

Zhaojiang Lin, Peng Xu, Genta Indra Winata, Zihan Liu, and Pascale Fung. 2019b. Caire: An end-to-end empathetic chatbot. *arXiv preprint arXiv:1907.12108*.

Linqing Liu, Yao Lu, Min Yang, Qiang Qu, Jia Zhu, and Hongyan Li. 2018. Generative adversarial network for abstractive text summarization. *AAAI*.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *AAAI*, pages 3075–3081.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Ça glar Gulçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *CoNLL 2016*, page 280.

Tong Niu and Mohit Bansal. 2018. Polite dialogue generation without parallel data. *Transactions of the Association for Computational Linguistics*, 6:273–389.

Ji Ho Park, Peng Xu, and Pascale Fung. 2018. Plusemo2vec at semeval-2018 task 1: Exploiting emotion knowledge from emoji and# hashtags. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 264–272.

Ramakanth Pasunuru and Mohit Bansal. 2018. Multi-reward reinforced summarization with saliency and entailment. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 646–653.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. *Sixth International Conference on Learning Representations*.

Martin Potthast, Tim Gollub, Matthias Hagen, and Benno Stein. 2018. The clickbait challenge 2017: towards a regression model for clickbait strength. *arXiv preprint arXiv:1812.10847*.

Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W Black. 2018. Style transfer through back-translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 866–876.

Lianhui Qin, Lemao Liu, Wei Bi, Yan Wang, Xiaojiang Liu, Zhiting Hu, Hai Zhao, and Shuming Shi. 2018. Automatic article commenting: the task and dataset. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Yuanyuan Qiu, Hongzheng Li, Shen Li, Yingdi Jiang, Renfen Hu, and Lijiao Yang. 2018. Revisiting correlations between intrinsic and extrinsic evaluations of word embeddings. In *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*, pages 209–221. Springer.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. *International Conference on Learning Representations*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1073–1083.

Shi-Qi Shen, Yan-Kai Lin, Cun-Chao Tu, Yu Zhao, Zhi-Yuan Liu, Mao-Song Sun, et al. 2017a. Recent advances on neural headline generation. *Journal of Computer Science and Technology*, 32(4):768–784.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017b. Style transfer from non-parallel text by cross-alignment. In *Advances in Neural Information Processing Systems*, pages 6830–6841.

Kai Shu, Suhang Wang, Thai Le, Dongwon Lee, and Huan Liu. 2018. Deep headline generation for clickbait detection. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 467–476. IEEE.

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017a. Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1171–1181.

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017b. From neural sentence summarization to headline generation: a coarse-to-fine approach. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 4109–4115. AAAI Press.

Duyu Tang, Furu Wei, Bing Qin, Nan Yang, Ting Liu, and Ming Zhou. 2016. Sentiment embeddings with applications to sentiment analysis. *IEEE Transactions on Knowledge and Data Engineering*, 28(2):496–509.

Arun Venkatraman, Martial Hebert, and J Andrew Bagnell. 2015. Improving multi-step prediction of learned time series models. In *AAAI*, pages 3024–3030.

Li Wang, Junlin Yao, Yunzhe Tao, Li Zhong, Wei Liu, and Qiang Du. 2018. A reinforced topic-aware convolutional sequence-to-sequence model for abstractive text summarization. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 4453–4460. AAAI Press.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Genta Indra Winata, Onno Kampman, Yang Yang, Anik Dey, and Pascale Fung. 2017. Nora the empathetic psychologist. *Proc. Interspeech 2017*, pages 3437–3438.

Genta Indra Winata, Onno Pepijn Kampman, and Pascale Fung. 2018. Attention-based lstm for psychological stress detection from spoken language using distant supervision. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6204–6208. IEEE.

Genta Indra Winata, Andrea Madotto, Zhaojiang Lin, Jamin Shin, Yan Xu, Peng Xu, and Pascale Fung. 2019. Caire_hkust at semeval-2019 task 3: Hierarchical attention for dialogue emotion classification. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 142–147.

Jingjing Xu, SUN Xu, Qi Zeng, Xiaodong Zhang, Xuancheng Ren, Houfeng Wang, and Wenjie Li. 2018a. Unpaired sentiment-to-sentiment translation: A cycled reinforcement learning approach. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 979–988.

Peng Xu and Pascale Fung. 2019. A novel repetition normalized adversarial reward for headline generation. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7325–7329. IEEE.

Peng Xu, Andrea Madotto, Chien-Sheng Wu, Ji Ho Park, and Pascale Fung. 2018b. Emo2vec: Learning generalized emotion representation by multi-task training. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 292–298.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*, pages 2852–2858.

Xingxing Zhang, Mirella Lapata, Furu Wei, and Ming Zhou. 2018a. Neural latent extractive document summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 779–784.

Yi Zhang, Jingjing Xu, Pengcheng Yang, and Xu Sun. 2018b. Learning sentiment memories for sentiment modification without parallel data. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1103–1108.

Li Zhou, Jianfeng Gao, Di Li, and Heung-Yeung Shum. 2018a. The design and implementation of xiaoice, an empathetic social chatbot. *arXiv preprint arXiv:1812.08989*.

Qingyu Zhou, Nan Yang, Furu Wei, Shaohan Huang, Ming Zhou, and Tiejun Zhao. 2018b. Neural document summarization by jointly learning to score and select sentences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 654–663.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2017. Selective encoding for abstractive sentence summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1095–1104.

# Concept Pointer Network for Abstractive Summarization

**Wang Wenbo**[1], **Gao Yang**[1,2*], **Huang Heyan**[1,2] **and Zhou Yuxiang**[1]
[1] School of Computer Science and Technology, Beijing Institute of Technology
[2] Beijing Engineering Research Center of High Volume Language Information Processing and Cloud Computing Applications
gyang@bit.edu.cn

## Abstract

A quality abstractive summary should not only copy salient source texts as summaries but should also tend to generate new conceptual words to express concrete details. Inspired by the popular pointer generator sequence-to-sequence model, this paper presents a concept pointer network for improving these aspects of abstractive summarization. The network leverages knowledge-based, context-aware conceptualizations to derive an extended set of candidate concepts. The model then points to the most appropriate choice using both the concept set and original source text. This joint approach generates abstractive summaries with higher-level semantic concepts. The training model is also optimized in a way that adapts to different data, which is based on a novel method of distantly-supervised learning guided by reference summaries and testing set. Overall, the proposed approach provides statistically significant improvements over several state-of-the-art models on both the DUC-2004 and Gigaword datasets. A human evaluation of the model's abstractive abilities also supports the quality of the summaries produced within this framework.

## 1 Introduction

Abstractive summarization (ABS) has gained overwhelming success owing to a tremendous development of sequence-to-sequence (seq2seq) model and its variants (Rush et al., 2015; Chopra et al., 2016; Paulus et al., 2017; Guo et al., 2018; Gao et al., 2019). In tandem with seq2seq models, pointer generator was developed by See et al. (2017) as a solution to tackle the rare words and out-of-vocabulary (OOV) problem associated with generative-based models. The idea behind is to use attention as a pointer to determine the probability of generating a word from both a vocabulary distribution and the source text. Pointer generator networks have also been extensively accepted by the ABS community due to their efficacy with long document summaries (Chen and Bansal, 2018; Hsu et al., 2018), title summarization (Sun et al., 2018), etc.

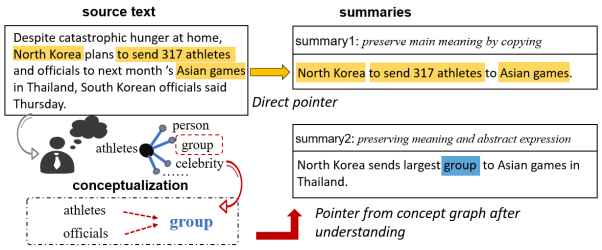


Figure 1: "summary1" only copies keyword from the source text, while "summary2" generates new concepts to convey the meaning.

However, the current power of abstractive summarization falls short of their potential. As the example in Figure 1 shows, a seq2seq model with a pointer mechanism (marked as the direct pointer) is likely to merely copy parts of the original text to form a summary using keywords and phrases, such as "317 athletes". Conversely, a more human-like summary would be based on one's own understanding of the detail in the words, expressed as higher-level concepts drawn from world knowledge—like using the word "group" to replace "athletes and officials". This indicates that a good summary should not simply copy original material, it should also generate new and even abstract concepts that reflect high-level semantics.

Therefore, a pointer generator network that solely considers the source material to generate a summary does not adequately satisfy the needs of high-quality abstractive summarization. We argue that concepts have a greater ability to express deeper meanings than verbatim words. As such, it is essential to explore the potential of us-

* Corresponding author



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3076–3085, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

ing concepts from world knowledge to assist with abstractive summarization. Our developed model not only points to informative source texts but also leverages conceptual words from human knowledge in the summaries it generates.

Hence, in this paper, we propose a novel model based on a concept pointer generator that encourages the generation of conceptual and abstract words. As a hidden benefit, the model also alleviates the OOV problems. Our model uses pointer network to capture the salient information from a source text, and then employs another pointer to generalize the detailed words according to their upper level of expressions. Finally, the output is also consistent with language model by the seq2seq generator. Unique to our concept pointer is a set of concept candidates particular for a word that is drawn from a huge knowledge base. The set of candidates adheres to a concept distribution, where the probability of each concept being generated is linked to how strongly the candidate represents each word. Moreover, the concept distribution is iteratively updated to better explain the target word given the context of the source material and inherent semantics in the texts. Hence, the learned concept pointer points to the most suitable and expressive concepts or words. The optimization function is adaptive so as to cater for different datasets with distantly-supervised training. The network is then optimized end-to-end using reinforcement learning, with the distant-supervision strategy as a complement to further improve the summary.

Overall, the contributions of this paper are: 1) a novel concept pointer generator network that leverages context-aware conceptualization and a concept pointer, both of which are jointly integrated into the generator to deliver informative and abstract-oriented summaries; 2) a novel distant supervision training strategy that favors model adaptation and generalization, which results in performance that outperforms the well-accepted evaluation-based reinforcement learning optimization on a test-only dataset in terms of ROUGE metrics; 3) a statistical analysis of quantitative results and human evaluations from comparative experiments with several state-of-the-art models that shows the proposed method provides promising performance.

## 2 Related Work

Abstractive summarization supposedly digests and understands the source content and, consequently, the generated summaries are typically a reorganization of the wording that sometimes form new sentences. Historically, abstractive summarization has been performed through rule-based sentence selection (Dorr et al., 2003), key information extraction (Genest and Lapalme, 2011), syntactic parsing (Bing et al., 2015) and so on. However, more recently, seq2seq models with attention have played a more dominant role in generating abstractive summaries (Rush et al., 2015; Chopra et al., 2016; Nallapati et al., 2016; Zhou et al., 2017). Extensions to the seq2seq approach include an intra-decoder attention (Paulus et al., 2017) and coverage vectors (See et al., 2017) to decrease repetition in phrasing. Copy mechanism (Gu et al., 2016) has been integrated into these models to tackle OOV problem. Zhou et al. (2018) went on to propose SeqCopyNet which copies complete sequences from an input sentence to further maintain the readability of the generated summary.

Pointer mechanism (Vinyals et al., 2015) has drawn much attention in text summarization (See et al., 2017), because this technique not only provides a potential solution for rare words and OOV but also extends abstractive summarization in a flexible way (Çelikyilmaz et al., 2018). Further, pointer generator models can effectively adaptive to both extractor and abstractor networks (Chen and Bansal, 2018), and summaries can be generated by incorporating a pointer-generator and multiple relevant tasks (Guo et al., 2018), such as question or entailment generation, or multiple source texts (Sun et al., 2018).

However, work particularly targets the problem of the abstraction is rare. Abstract Meaning Representation (AMR) is used to transform a sentence into a concept graph, then merge those similar concept nodes to form a new summary graph (Liu et al., 2018). Concepts are also incorporated as auxiliary features (Guo et al., 2017). Kryscinski et al. (2018) and Weber et al. (2018) define the number of new $n$-grams as the primary criteria of abstractiveness. This makes sense in most cases. But, we believe that abstraction means summarizing detailed content with higher-level semantically related concepts, which has motivated the development of the model proposed in this paper.

## 3 The Proposed Model

Neural abstractive summarization can be described as a generation process where a sequential input is summarized into a shorter sequential output through a neural network. Suppose that the sequential input $\boldsymbol{x} = \{x_1, \ldots, x_i, \ldots, x_n\}$ is a sequence of $n$ number of words, and $i$ is the index of the input. The shorter (i.e., summarized) sequence of output is denoted as $\boldsymbol{y} = \{y_1, \ldots, y_t, \ldots, y_m\}$ with number of $m$ words, and $t$ indicates a time step. As Figure 2 shows, our model consists of two sub-modules —an encoder-decoder module and the proposed concept pointer generator module.

### 3.1 Encoder-Decoder Framework

This process is formulated as an encoder-decoder framework that consists of an encoder and an attention-equipped decoder. We use a two-layer bi-directional LSTM-RNN encoder and one-layer uni-directional LSTM-RNN decoder along with attention mechanism.

Formally, the encoder produces sequential hidden states as $(\overrightarrow{\boldsymbol{h}}_1, \ldots, \overrightarrow{\boldsymbol{h}}_n)$ and $(\overleftarrow{\boldsymbol{h}}_1, \ldots, \overleftarrow{\boldsymbol{h}}_n)$ in the corresponding positions, and the bi-directional $h_i = f_{LSTM}(h_{i-1}, x_i)$. Each word $x_i$ in the sequence can be represented as a concatenation of the bi-directional hidden states, i.e., $\boldsymbol{h}_i = [\overrightarrow{\boldsymbol{h}}_i, \overleftarrow{\boldsymbol{h}}_i]$. The decoder generates a target summary from a vocabulary distribution $P_{vocab}(w)$, which is based on a context vector $\boldsymbol{h}_t^*$ through the following process:

$$\begin{aligned} P_{vocab}(w) &= P(y_t|\boldsymbol{y}_{<t}, \boldsymbol{x}; \theta) \\ &= \text{sfm}(\boldsymbol{W}_2(\boldsymbol{W}_1[\boldsymbol{s}_t, \boldsymbol{h}_t^*] + \boldsymbol{b}_1) + \boldsymbol{b}_2) \end{aligned} \quad (1)$$

where $\boldsymbol{s}_t$ is the hidden state of the decoder at time step $t$ , and $\boldsymbol{h}_t^*$ is the context vector at time step $t$. $\boldsymbol{W}_1, \boldsymbol{W}_2, \boldsymbol{b}_1, \boldsymbol{b}_2$ are trainable parameters, and sfm$(\cdot)$ is short for softmax function.

The context vector $\boldsymbol{h}_t^*$ is computed by a weighted sum of the hidden representations of the source text, and the weight is denoted as the attention $\alpha_{t,i}$.

$$\begin{aligned} \boldsymbol{h}_t^* &= \sum\nolimits_{i=1}^{n} \alpha_{t,i} \boldsymbol{h}_i \\ \alpha_{t,i} &= \text{sfm}(\boldsymbol{v}^\top \tanh(\boldsymbol{W}_h \boldsymbol{h}_i + \boldsymbol{W}_s \boldsymbol{s}_t + \boldsymbol{b})) \end{aligned} \quad (2)$$

The softmax function normalizes the vector of a distribution over the input position, and $\boldsymbol{v}, \boldsymbol{W}_h, \boldsymbol{W}_s, \boldsymbol{b}$ are trainable parameters.

### 3.2 Concept Pointer Generator

Pointer networks use attention as a pointer to select segments of the input as outputs (Vinyals et al., 2015). As such, a pointer network is a suitable mechanism for extracting salient information, while remaining flexible enough to interface with a seq2seq model for generating an abstractive summarization (See et al., 2017). Our proposed model is essentially an upgrade to this configuration that integrates a new concept pointer network within a unified framework.

#### 3.2.1 Context-aware Conceptualization

"Understanding" the instances of a word requires a taxonomic knowledge base that relates those words to a concept space. In our model, we use an $isA$ taxonomy, called the Microsoft Concept Graph[1] (Wang et al., 2015), to serve this purpose for two reasons (Wang and Wang, 2016). First, this graph provides a huge concept space with multi-word terms that cover concepts of worldly facts as concepts, instances, relationships, and values[2]. Second, the relationships between concepts and entities are probabilistic as a measure of how strongly they are related. Moreover, the probabilities are trustworthy given they have been derived from evidence found in billions of webpages, search log data, and other existing taxonomies. Our model is data-driven and, therefore, is more easily adaptable with probabilities. All these characteristics make the Microsoft Concept Graph a suitable choice for our model. More detailed examples are available in Appendix A.

The concept graph specifies the probability that each instance $x$ belongs to a concept $c$, $p(c|x)$. Given a word $x$, we have a distribution over a set of related concepts. Yet, this raises the question of how to identify a context-appropriate concept for a word from the distributional set of candidate concepts. For instance, *apple* in the context of "*an apple is good for you health*" tends to be associated with the concept of *fruit* instead of *company*. Formally, given a word $x_i$ in a training sentence, a set of $k$ concept candidates, $C_i = \{c_i^1, c_i^2, \cdots, c_i^k\}$, is linked to the word from the knowledge base, with distributional probabilities over the concepts, i.e.,

[1]The Microsoft concept graph was derived from Probase project. The public data can be downloaded via the provided API: https://concept.research.microsoft.com/Home/API

[2]The current version is mined from billions of web pages, containing 5.3 M unique concepts, 12.5 M unique entities and 85 M $isA$ relations.

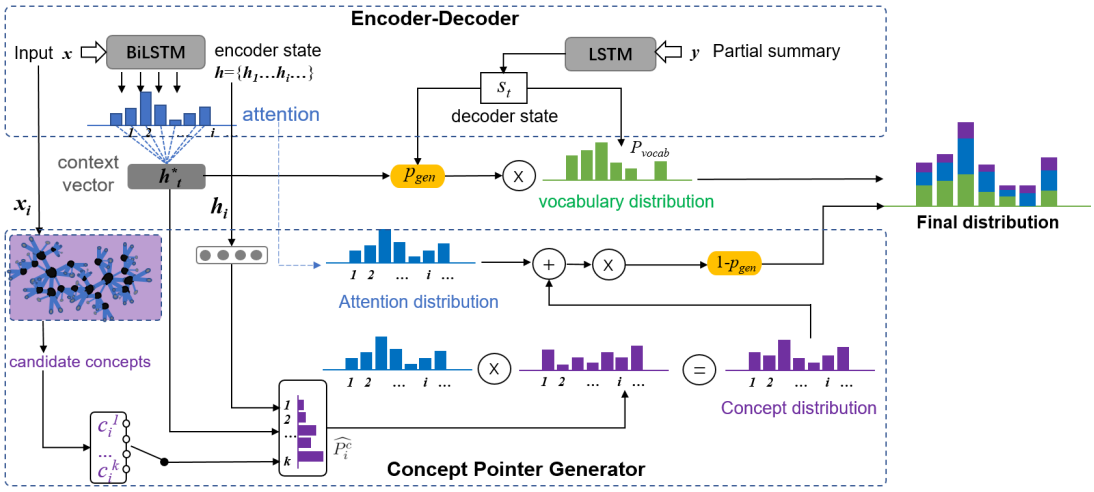


Figure 2: The architecture of our model. Blue bar represents the attention distribution over the inputs. Purple bar represents the concept distribution over the inputs. Noted that, this distribution can be sparse since not every word has its upper concept. Green bar represents the vocabulary distribution generated from seq2seq component.

$P(C|x_i) = \{p(c_i^1), p(c_i^2), \cdots, p(c_i^k)\}$. The task is to find the most suitable concept $c_i^j$ to fit the updated context, represented by the vector $\boldsymbol{h}_t^*$ in Equation (2), at each time step $t$.

In the case of generating summaries given updated contexts, a weighted update of the distributional concept candidates needs to be performed. In the model, the updated weight, denoted as $\beta_i^j$, is estimated by a softmax classifier that is jointly conditioned on the hidden representation of the word $\boldsymbol{h}_i$, the context vectors $\boldsymbol{h}_t^*$, and each of concept vectors:

$$\beta_i^j = \text{softmax}(\boldsymbol{W}_{h'}[\boldsymbol{h}_i, \boldsymbol{h}_t^*, \boldsymbol{c}_i^j]) \quad (3)$$

where $j \in [1, k]$, $\boldsymbol{W}_{h'}$ is a trainable parameter, and $\boldsymbol{c}_i^j$ is the vector of the $j$th concept candidate, which is a representation of the input embeddings.

Together with the concept probability from the existing knowledge base $p(c_i^j)$ and the updated weights based on the contexts $\beta_i^j$, a context-aware conceptualized probability of $j$th concept for the word $x_i$, $P_{i,j}^c$, is finally estimated as

$$P_{i,j}^c = p(c_i^j) + \gamma \beta_i^j \quad (4)$$

where $\gamma$ is a tunable parameter. Theoretically, we will end up with a number of $k$ relevant concepts for each word $C_i = \{c_i^1, \cdots, c_i^k\}$ with a probability distribution over the set, which is learned as $P_i^c = \{P_{i,1}^c, \cdots, P_{i,j}^c, \cdots, P_{i,k}^c\}$.

### 3.2.2 Concept Pointer Generator

The basic pointer generator network contains two sub-modules, one is the pointer network and the other is the generation network. These two sub-modules jointly determine the probabilities of the words in the final generated summary. The generation probability $p_{gen}$ for the generation network (See et al., 2017) is learned by

$$p_{gen} = \sigma(\boldsymbol{W}_{h^*}\boldsymbol{h}_t^* + \boldsymbol{W}_s\boldsymbol{s}_t + \boldsymbol{W}_y\boldsymbol{y}_{t-1} + \boldsymbol{b}_{gen}) \quad (5)$$

where $\sigma$ is a sigmoid function.

For the pointer network, our model consists of a pointer to the source text and a further concept pointer to the relevant concepts that have arisen from the source content. These two separate pointers are calculated as follows. The first pointer is taken based on the attention distribution $\alpha_{t,i}$ over the source text. The second concept pointer is operated over a concept distribution of the source text that is scaled element-wise by the attention distribution.

To train the model, given the likelihood of each concept in the current context, the updates could be performed in two ways. In a hard assignment, the concept that receives the highest score would be selected for the update:

$$\widehat{P_i^c}_{\text{argmax}} = P_{i,a}^c, \text{where } a = \arg\max_j(\beta_i^j) \quad (6)$$

where $a$ is the index of maximized generated weight based on the contexts, and $P_{i,a}^c$ is obtained by Eqs. (4).

In random selection, each of the concept candidates could be trained randomly to update the parameters:

$$\widehat{P_i^c}_{\text{random}} = P_{i,j}^c \sim P_i^C \quad (7)$$

where $j$ represents the selected concept index. Considering the above baseline generation network and both the pointer networks, our final output distribution is

$$P_{\text{final}}(w) = p_{gen}P_{vocab}(w) + (1 - p_{gen}) \\ (\sum_{i:w_i=w} \alpha_{t,i} + \sum_{i:w_i=w} \alpha_{t,i} \times \widehat{P_i^c}) \quad (8)$$

where $\widehat{P_i^c}$ can be updated by $\widehat{P_{i\ \arg\max}^c}$, or $\widehat{P_{i\ \text{random}}^c}$. The difference between these two choices is demonstrated in the Experiments section.

### 3.3 Objective Learning

#### 3.3.1 Basic MLE

The baseline objective is derived by maximizing the likelihood training for the seq2seq generation, given a reference summary $y^* = \{y_1^*, y_2^*, \cdots, y_{m'}^*\}$ for document $x$. The training objective is to minimize the negative log-likelihood of the target word sequence:

$$\mathcal{L}_{MLE} = -\sum_{t=1}^{m'} \log P(y_t^*|y_1^*, \cdots, y_{t-1}^*, \boldsymbol{x}) \quad (9)$$

#### 3.3.2 Evaluation based Reinforcement Learning (RL)

Similar to Paulus et al. (2017), policy gradient methods can directly optimize discrete target evaluation metrics, such as ROUGE. The basic idea is to explore new sequences and compare them to the best greedy baseline sequence. Once the baseline sequence $\hat{y}$, or sampled sequence $y^s$, are generated, they are compared against the reference sequence $y^*$ to compute the rewards $r(\hat{y})$ and $r(y^s)$, respectively. In the RL training stage, two separate output candidates at each time step are produced: $y^s$ is sampled from the probability distribution $P(y_t^s|y_1^s, \cdots, y_{t-1}^s, x)$ , and $\hat{y}$ is the baseline output. The training objective is then to minimize

$$\mathcal{L}_{RL} = (r(\hat{y}) - r(y^s)) \sum_{t=1}^{m'} \log P(y_t^s|y_1^s, \cdots, y_{t-1}^s, x) \quad (10)$$

It is noteworthy that the samples $y^s$ are selected from a wide range of vocabularies extended by all the concept candidates. This strategy ensures that the model learns to generate sequences with higher rewards by better exploring a set of close concepts.

Thus, the combination of these two objectives yield improved task-specific scores while catering a better language model: $\mathcal{L}_{final} = \lambda\mathcal{L}_{RL} + (1 - \lambda)\mathcal{L}_{MLE}$, where $\lambda$ is a soft-switch between the two objectives. The model is pre-trained with MLE loss, then switch to the final loss.

#### 3.3.3 Distant Supervision (DS) for Model Adaption

Our intuition is that, if the summary-document pairs are dissimilar to the testing set, the model could be retrained to adapt to weaken the influence of the dissimilarity on the final loss. The result would be a training model that better fits the specific testing data. The challenge is that there are no explicit supervision labels to indicate whether the training set is close to the testing set, so a new training paradigm is needed. In answer to this need and also to provide end-to-end functionality in the model, we developed a simple approach for labeling summary-document pairs by calculating the Kullback-Leibler (KL) divergence between each training reference summary and a set of testing documents. In this way, the training pairs are distantly-labelled for training the model.

Specifically, the representations of the reference summaries and the testing set are computed by summing all the involved word embeddings. Given a testing document $x_l^{\text{d}}$, where $l \in [1, N^{\text{d}}]$ and $N^{\text{d}}$ is the size of the testing corpus, the vector-based representation of one document is $\boldsymbol{x}_l^{\text{d}} = \exp(\sum_{i=1}^{n'} \boldsymbol{x}_i^{\text{d}})$, where $n'$ is the number of document words involved. The reference summary is represented by $\boldsymbol{y^*} = \exp(\sum_{t=1}^{m'} \boldsymbol{y_t^*})$. We normalize these vectors through a softmax function to cater for KL calculation. The model adaption with the distant labels is defined as:

$$\mathcal{L}_{DS} = (\pi - \tfrac{1}{N^{\text{d}}} \sum_{l=1}^{N^{\text{d}}} D_{\text{KL}}(\boldsymbol{y^*}, \boldsymbol{x}_l^{\text{d}}))\mathcal{L}_{\text{MLE}} \quad (11)$$

where $D_{\text{KL}}(.)$ indicates the KL divergence between $y^*$ and $x_l^{\text{d}}$, and $\pi$ is a constant parameter that is tuned via adaption to the testing set. The divergences are averaged within the testing set, which indicates the overall distances between testing set and each of the reference summary-document pairs. In this way, the samples in the training corpus are distantly annotated as either relevant or irrelevant for model adaption, noting that the model is pre-trained with the MLE loss before switching to distantly-supervised training.

## 4 Experiments

**Datasets:** To evaluate the effectiveness of our proposed model, we conducted training and test-

Table 1: ROUGE F1 evaluation results on the Gigaword and ROUGE recall on DUC-2004 test set. The results with † mark are taken from the corresponding papers. Underlined scores are the best without additional optimization. Bold scores are the best between the two optimization strategies. ⋆ mark indicates the improvements from the baselines to the concept pointer are statistically significant using a two-tailed t-test ($p < 0.01$).

| Models | Gigaword | | | DUC-2004 | | |
|---|---|---|---|---|---|---|
| | RG-1 | RG-2 | RG-L | RG-1 | RG-2 | RG-L |
| ABS+† (Rush et al., 2015) | 29.76 | 11.88 | 26.96 | 28.18 | 8.46 | 23.81 |
| Luong-NMT† (Luong et al., 2015) | 33.10 | 14.45 | 30.71 | 28.55 | 8.79 | 24.43 |
| RAS-Elman† (Chopra et al., 2016) | 33.78 | 15.97 | 31.15 | 28.97 | 8.26 | 24.06 |
| lvt5k-lsent† (Nallapati et al., 2016) | 35.30 | 16.64 | 32.62 | 28.61 | 9.42 | 25.24 |
| SEASS† (Zhou et al., 2017) | 36.15 | 17.54 | 33.63 | 29.21 | 9.56 | 25.51 |
| Seq2seq+att ⋆(our impl.) | 33.11 | 14.67 | 31.06 | 28.57 | 9.31 | 24.81 |
| Pointer-generator ⋆(our impl.) (See et al., 2017) | 35.98 | 15.99 | 33.33 | 28.28 | 10.04 | 25.69 |
| Pointer-Cov.-Entail.-Quest.† (Guo et al., 2018) | 35.98 | 17.76 | 33.63 | - | - | - |
| Seq2seq-Sel.-MTL-ERAM† (Li et al., 2018) | 35.33 | 17.27 | 33.19 | 29.33 | 10.24 | 25.24 |
| CGU† (Lin et al., 2018) | 36.3 | 18.0 | 33.8 | - | - | - |
| **Concept pointer** | 36.62 | 16.40 | 33.98 | 29.17 | 10.38 | 26.34 |
| **Concept pointer+RL** | **38.02** | 16.97 | **35.43** | 29.34 | 9.84 | 26.60 |
| **Concept pointer+DS** | 37.01 | **17.10** | 34.87 | **30.39** | **10.78** | **27.53** |

ing on two popular datasets. The first was the English Gigaword Fifth Edition corpus (Parker et al., 2011). We replicated the pre-processing steps in (Rush et al., 2015) to obtain the same training/testing data. After pre-processing, the corpus contained about 3.8M sentence-summary pairs as training set and 189K pairs as the development set. Once pairs with empty titles were removed, the testing set numbered 1951 pairs. The second dataset, DUC2004, was only used for testing. This dataset consists of 500 document-headline summary pairs, where each document is paired with four reference summaries written by humans.

**Evaluation Metrics:** We used ROUGE (Lin, 2004) as the evaluation metric, which measures the quality of a summary by computing the overlapping lexical elements between the candidate summary and a reference summary. Following previous practice, we assessed RG-1 (unigram), RG-2 (bigram) and RG-L (longest common subsequence - LCS). Noted that the English Gigaword[3] testing set contains references of different lengths, while the DUC-2004[4] testing set fixes the summary length to 75 bytes.

**Training Setups:** We initialize word embeddings with 128-d vectors and fine-tune them during training. Concepts share the same embeddings with the words. The vocabulary size was set to 150k for both the source and target text. The hidden state size was set to 256. The vocabulary size is increased from around 602 to 2216 concepts w.r.t the different number ($k = 1, \cdots, 5$) of concept candidates for each word. Note that the generated concepts with UNKs were subsequently deleted. Our code is available on `https://github.com/wprojectsn/codes`, and the vocabularies and candidate concepts are also included.

We trained our models on a single GTX TITAN GPU machine. We used the Adagrad optimizer with a batch size of 64 to minimize the loss. The initial learning rate and the accumulator value were set to 0.15 and 0.1, respectively. We used gradient clipping with a maximum gradient norm of 2. At the time of decoding, the summaries were produced through a beam search of size 8. The hyper-parameter settings were $\lambda = 0.99$, $\gamma = 0.1$, $\pi = 2.92$ on DUC-2004 and $\pi = 1.68$ on Gigaword. We trained our concept pointer generator for 450k iterations yielded the best performance, then took the optimization using RL rewards for RG-L at 95K iterations on DUC-2004 and at 50K iterations on Gigaword. We took the distance-supervised training at 5K iterations on DUC-2004 and at 6.5K iterations on Gigaword.

**Baselines:** The following state-of-the-art baselines were used as comparators. **ABS+** (Rush

[3]The ROUGE evaluation option is, -m -n 2 -s

[4]The ROUGE evaluation option is, -n 2 -m -b 75 -s

Table 2: OOV problem analysis: percentages (%(NO. UNK/NO. all generated words)) of generating UNK w.r.t the following three models on Gigaword and DUC-2004 datasets.

| Method | Gigaword | DUC-2004 |
|---|---|---|
| seq2seq+att | 4.02%(570/14183) | 2.08%(85/4079) |
| Pointer generator | 1.16%(207/17859) | 0.31% (16/5140) |
| Concept pointer | 1.12%(202/17950) | 0.23%(12/5230) |

Table 3: Abstractiveness: percentage of novel $n$-grams on Gigaword dataset.

| Models | Novel $n$-gram (%) | | |
|---|---|---|---|
| | 1-gram | 2-gram | 3-gram |
| Pointer generator | 14.3 | 41.9 | 63.4 |
| Concept pointer | 17.2 | 45.8 | 68.5 |
| Reference summary | 25.4 | 65.6 | 78.4 |

et al., 2015) is a tuned ABS model with additional features. **Luong-NMT** (Luong et al., 2015) is a two-layer LSTM encoder-decoder. **RAS-Elman** (Chopra et al., 2016) is a convolution encoder and an Elman RNN decoder with attention. **Seq2seq+att** is two-layer BiLSTM encoder and one-layer LSTM decoder equipped with attention. **lvt5k-lsent** (Nallapati et al., 2016) uses temporal attention to keep track of the past attentive weights of the decoder and restrains the repetition in later sequences. **SEASS** (Zhou et al., 2017) includes an additional selective gate to control information flow from the encoder to the decoder. **Pointer-generator** (See et al., 2017) is an integrated pointer network and seq2seq model. We implemented this baseline without its coverage mechanism since this is not our focus. Baseline models also include two pointer-generator based extensions (Guo et al., 2018; Li et al., 2018). **CGU** (Lin et al., 2018) sets a convolutional gated unit and self-attention for global encoding.

## 5 Results and Analysis

The following analysis focuses on investigating whether our model is, first, able to generate abstract and new concepts, and, second, how the overall quality performs against the baselines.

### 5.1 Quantitative Analysis

The results are presented in Table 1. We observe that our model outperformed all the strong state-of-the-art models on both datasets in all metrics except for RG-2 on Gigaword. In terms of the pointer generator performance, the improvements made by our concept pointer are statistically significant ($p < 0.01$) across all metrics.

**OOV and Summary Length:** OOV is another major challenge for current abstractive summarization models. Although generating longer summaries or less UNKs is not our focus, our model still showed improvements in this regard (Table 2). We counted the number of UNKs and all generated summary words and measured the proportions in both testing sets. The OOV percentages dropped from 4.02% to 1.12% on Gigaword and from 2.08% to 0.23% on DUC-2004, which demonstrates that our model is effective at alleviating OOV problems. This result also supports the superior of the concept pointer over the baseline pointer generator. From the statistics, we found that the summaries generated by our concept pointer averaged around 10.46 words, while the pointer generator summaries averaged 10.28 words per summary on DUC-2004. This shows the concept pointer is able to capture more salient content by generating relatively longer summaries.

**Abstractiveness:** According to Chen and Bansal (2018), abstractiveness scores are computed as the percentage of novel $n$-grams in the generated summaries that are not included in the source documents. As shown in Table 3, compared with human-written summaries which receive the highest novelty in terms of abstractiveness, our concept pointer generator achieves closest performance with human-written summaries against the baseline. This result demonstrates a further advantage of our model in producing new and abstract concepts. Our model is designed to improve semantic relevance and promote higher abstraction. More generated summary examples can be found in Appendix B.

### 5.2 Analysis on Training Strategies

To evaluate the relative impact of each training strategy with the model, we tested different combinations for comparison with each other and against the baselines.

**Context-aware Conceptualization:** To investigate the impact of training with both the number of concepts $k$ and the concept update strategy mentioned in Eqs.(6) and (7), we chose a different number of concept candidates, i.e., $k = 1, 2, 3, 4, 5$, to for the context-aware conceptualization update strategy. Performance was fully

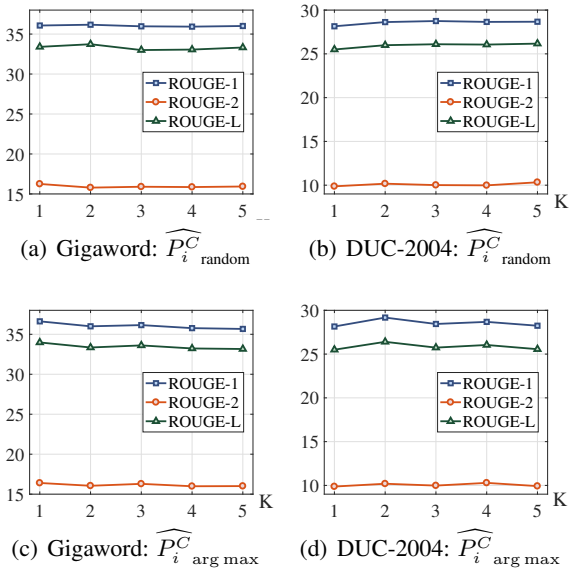


(a) Gigaword: $\widehat{P_i^C}_{\text{random}}$ (b) DUC-2004: $\widehat{P_i^C}_{\text{random}}$

(c) Gigaword: $\widehat{P_i^C}_{\arg\max}$ (d) DUC-2004: $\widehat{P_i^C}_{\arg\max}$

Figure 3: ROUGE metrics on Gigaword and DUC-2004 w.r.t a different number of concept candidates. Updates were conducted by hard assignment $\widehat{P_i^C}_{\arg\max}$ and random selection $\widehat{P_i^C}_{\text{random}}$.

evaluated with the three ROUGE metrics as shown in Figure 3. The results only vary slightly according to the number of concepts with the random selection strategy (Eq.(7)), as shown in Figure 3(a) and 3(b). This indicates that a random strategy is not very sensitive to the number of extracted topics. This is, in part, because the concept pointer may or may not be able to point to the correct concepts from multiple candidates. While in Figure 3(c) and 3(d), the optimum settings are clearly apparent, i.e., $k = 1$ on Gigaword and $k = 2$ on DUC-2004. Overall, the hard assignment strategy (Eq.(6)) provided the best performance in practical terms, while random selection (Eq.(7)) performs stably with different settings.

**Training with DS vs. RL:** As shown in Table 1, our model with either a distant supervision strategy (concept pointer+DS) or reinforcement learning (concept pointer+RL) were both superior to the basic concept pointer generator on both datasets. Further, the relative improvement of the concept pointer+DS over the concept pointer+RL ranged from 3.5% to 9.6% on DUC-2004 but was inferior to concept pointer+RL on Gigaword. In comparing the results, it is clear that DS training has a noticeable effect when the testing set is substantially semantically different from the training set but provides less improvement than RL when the two are close. From this analysis, we conclude that the DS strategy is better for model adaption with abstractive summarization.

Table 4: Human evaluation: scoring of three models in terms of abstraction and overall quality by human evaluators (the higher the better). The score range could be 0-20. $\star$ indicates the improvements from the baselines to the concept pointer are statistically significant.

| Method | abstraction | overall quality |
|---|---|---|
| seq2seq+att $\star$ | 5.85 | 5.65 |
| Pointer generator $\star$ | 8.95 | 8.10 |
| Concept pointer | 10.00 | 9.60 |

### 5.3 Human Evaluations

To explore the correctness of our model using human judgment, we conducted a manual evaluation with 20 post-graduate volunteers. We primarily used the following criteria to assess the generated summaries: *abstraction*, i.e., Are the abstract concepts contained in the summary appropriate?; and *overall quality*, i.e., Is the summary readable, informative, relevant, etc.? To conduct the evaluation, we randomly selected 20 examples from the DUC 2004 testing set and asked the volunteers to subjectively assess the summaries. Each example consisted of an article and three summaries, i.e., a summary by the seq2seq model, the pointer generator model, and our proposed concept pointer model. The volunteers chose the best summaries for each of the articles according to the above criteria (can be multiple choices). Obviously, the summaries were randomly shuffled, and the model used to produce each was unknown to prevent bias. The scores for each model were ranked by how many times the volunteers chose a summary w.r.t each criteria, averaged by the number of participants. The results are presented in Table 4, which show that our model outperformed both the seq2seq model and the pointer generator (See et al., 2017) in both criteria.

As a last step, we manually inspect the summaries generated by our model, and some examples are presented in Appendix B. We found that the summaries were not as abstract as human-written summary would likely be. The overarching tendency of the model is still to copy segments of the source text and rearrange the phrases into a summary. However, the overall approach does produce more high-level concepts with correct relations compared to the baselines, which demon-

strates that our solution is a promising research direction to further pursue. Additionally, the generated summaries are long, fluent, and informative.

## 6 Conclusion

This paper presents a novel concept pointer generator model to improve the abstractive summarization model and generate concept-oriented summaries. We also propose a novel distant supervision strategy for model adaption to different datasets. Both empirical and subjective experiments show that our model makes a statistically significant quality improvement over the state-of-the-art baselines on two popular datasets.

## Acknowledgement

This work was supported by National Natural Science Foundation of China (No.61602036, 61751201), partially supported by the Research Foundation of Beijing Municipal Science & Technology Commission (No. Z181100008918002), and partially supported by Ministry of Education China Mobile Research Foundation (No. MCM20170302).

## References

Lidong Bing, Piji Li, Yi Liao, Wai Lam, Weiwei Guo, and Rebecca J. Passonneau. 2015. Abstractive multi-document summarization via phrase selection and merging. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL, Volume 1: Long Papers*, pages 1587–1597.

Asli Çelikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep communicating agents for abstractive summarization. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, Volume 1 (Long Papers)*, pages 1662–1675.

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. *arXiv preprint arXiv:1805.11080*.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98.

Bonnie Dorr, David Zajic, and Richard Schwartz. 2003. Hedge trimmer: A parse-and-trim approach to headline generation. In *Proceedings of the HLT-NAACL 03 on Text summarization workshop-Volume 5*, pages 1–8. Association for Computational Linguistics.

Yang Gao, Yang Wang, Luyang Liu, Yidi Guo, and Heyan Huang. 2019. Neural abstractive summarization fusing by global generative topics. *Neural Computing and Applications*.

Pierre-Etienne Genest and Guy Lapalme. 2011. Framework for abstractive summarization using text-to-text generation. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 64–73.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1631–1640.

Han Guo, Ramakanth Pasunuru, and Mohit Bansal. 2018. Soft layer-specific multi-task summarization with entailment and question generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Volume 1: Long Papers*, pages 687–697.

Yidi Guo, Heyan Huang, Yang Gao, and Chi Lu. 2017. Conceptual multi-layer neural network model for headline generation. In *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*, pages 355–367. Springer.

Wan-Ting Hsu, Chieh-Kai Lin, Ming-Ying Lee, Kerui Min, Jing Tang, and Min Sun. 2018. A unified model for extractive and abstractive summarization using inconsistency loss. *arXiv preprint arXiv:1805.06266*.

Wojciech Kryscinski, Romain Paulus, Caiming Xiong, and Richard Socher. 2018. Improving abstraction in text summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1808–1817.

Haoran Li, Junnan Zhu, Jiajun Zhang, and Chengqing Zong. 2018. Ensure the correctness of the summary: Incorporate entailment knowledge into abstractive sentence summarization. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018*, pages 1430–1441.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Junyang Lin, Xu Sun, Shuming Ma, and Qi Su. 2018. Global encoding for abstractive summarization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL, 2018, Volume 2: Short Papers*, pages 163–169.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A Smith. 2018. Toward abstractive summarization using semantic representations. *arXiv preprint arXiv:1805.10399*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015*, pages 1412–1421.

Ramesh Nallapati, Bowen Zhou, Cícero Nogueira dos Santos, Çaglar Gülçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL*, pages 280–290.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English gigaword fifth edition ldc2011t07. dvd. *Philadelphia: Linguistic Data Consortium*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 379–389.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Volume 1: Long Papers*, pages 1073–1083.

Fei Sun, Peng Jiang, Hanxiao Sun, Changhua Pei, Wenwu Ou, and Xiaobo Wang. 2018. Multi-source pointer network for product title summarization. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 7–16. ACM.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Zhongyuan Wang and Haixun Wang. 2016. Understanding short texts. In *the Association for Computational Linguistics (ACL) (Tutorial)*.

Zhongyuan Wang, Haixun Wang, Ji-Rong Wen, and Yanghua Xiao. 2015. An inference approach to basic level of categorization. In *Proceedings of the 24th acm international on conference on information and knowledge management*, pages 653–662. ACM.

Noah Weber, Leena Shekhar, Niranjan Balasubramanian, and Kyunghyun Cho. 2018. Controlling decoding for more abstractive summaries with copy-based networks. *arXiv preprint arXiv:1803.07038*.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2017. Selective encoding for abstractive sentence summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, Volume 1: Long Papers*, pages 1095–1104.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2018. Sequential copying networks. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18)*, pages 4987–4995.

# Surface Realisation Using Full Delexicalisation

**Anastasia Shimorina**
LORIA / Lorraine University
anastasia.shimorina@loria.fr

**Claire Gardent**
LORIA / CNRS
claire.gardent@loria.fr

## Abstract

Surface realisation (SR) maps a meaning representation to a sentence and can be viewed as consisting of three subtasks: word ordering, morphological inflection and contraction generation (e.g., clitic attachment in Portuguese or elision in French). We propose a modular approach to surface realisation which models each of these components separately, and evaluate our approach on the 10 languages covered by the SR'18 Surface Realisation Shared Task shallow track. We provide a detailed evaluation of how word order, morphological realisation and contractions are handled by the model and an analysis of the differences in word ordering performance across languages.

## 1 Introduction

Surface realisation maps a meaning representation to a sentence. In data-to-text generation, it is part of a complex process aiming to select, compress and structure the input data into a text. In text-to-text generation, it can be used as a mean to rephrase part or all of the input content. For instance, Takase et al. (2016) used surface realisation to generate a summary based on the meaning representations of multiple input documents and Liao et al. (2018) to improve neural machine translation.

By providing parallel data of sentences and their meaning representation, the SR'18 Surface Realisation shared task (Mille et al., 2018) allows for a detailed evaluation and comparison of surface realisation models. Moreover, as it provides training and test data for multiple languages, it also allows for an analysis of how well these models handle languages with different morphological and topological properties.

The SR'18 shared task includes two tracks: a shallow track where the input is an unordered, lemmatised dependency tree and a deep track where function words are removed and syntactic relations are replaced with semantic ones. In this paper, we focus on the shallow track of the SR'18 Shared Task and we propose a neural approach which decomposes surface realisation into three subtasks: word ordering, morphological inflection and contraction generation (e.g., clitic attachment in Portuguese or elision in French). We provide a detailed analysis of how each of these phenomena (word order, morphological realisation and contraction) is handled by the model, and we discuss the differences between languages.

For reproducibility, all our experiments including data and scripts are available at https://gitlab.com/shimorina/emnlp-2019.

## 2 Related Work

Early approaches for surface realisation adopted statistical methods, including both pipelined (Bohnet et al., 2010) and joint (Song et al., 2014; Puduppully et al., 2017) architecture for word ordering and morphological generation.

Multilingual SR'18 was preceded by the SR'11 surface realisation task for the English language only (Belz et al., 2011). The submitted systems in 2011 had grammar-based and statistical nature, mostly relying on pipelined architecture. Recently, Marcheggiani and Perez-Beltrachini (2018) proposed a neural end-to-end approach based on graph convolutional encoders for the SR'11 deep track.

The SR'18 shallow track received submissions from eight teams with seven of them dividing the task into two subtasks: word ordering and inflection. Only Elder and Hokamp (2018) developed a joint approach, however, they participated only in the English track.

For word ordering, five teams chose an approach based on neural networks, two used a



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3086–3096,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

classifier, and one team resorted to a language model. As for the inflection subtask, five teams applied neural techniques, two used lexicon-based approaches, and one used an SMT system (Basile and Mazzei, 2018; Castro Ferreira et al., 2018; Elder and Hokamp, 2018; King and White, 2018; Madsack et al., 2018; Puzikov and Gurevych, 2018; Singh et al., 2018; Sobrevilla Cabezudo and Pardo, 2018). Overall, neural components were dominant across all the participants. However, official scores of the teams that went neural greatly differ. Furthermore, two teams (Elder and Hokamp, 2018; Sobrevilla Cabezudo and Pardo, 2018) applied data augmentation, which makes their results not strictly comparable to others.

One of the interesting findings of the shared task is reported by Elder and Hokamp (2018) who showed that applying standard neural encoder-decoder models to jointly learn word ordering and inflection is highly challenging; their sequence-to-sequence baseline without data augmentation got 43.11 BLEU points on English.

Our model differs from previous work in three main ways. First, it performs word ordering on fully delexicalised data. Delexicalisation has been used previously but mostly to handle rare words, e.g. named entities. Here we argue that surface realisation and, in particular, word ordering works better when delexicalising all input tokens. This captures the intuition that word ordering is mainly determined by the syntactic structure of the input. Second, we we provide a detailed evaluation of how our model handles the three subtasks underlying surface realisation. While all SR'18 participants provided descriptions of their models, not all of them performed an in-depth analysis of model performance. Exceptions are works of King and White (2018), who provided a separate evaluation for the morphological realisation module, and Puzikov and Gurevych (2018), who evaluated both word ordering and inflection modules. However, it is not clear how each of those modules affect the global performance when merged in the full pipeline. In contrast, we propose a detailed incremental evaluation of each component of the full pipeline and show how each component impacts the final scores. Third, we introduce a linguistic analysis, based on the dependency relations, of the word ordering component, allowing for deeper error analysis of the developed systems.

Furthermore, our model explicitly integrates a module for contraction handling, as done also before by Basile and Mazzei (2018). We also address all the ten languages proposed by the shared task and outline the importance of handling contractions.

## 3 Data

The SR'18 data (shallow track) is derived from the ten Universal Dependencies (UD) v2.0 treebanks (Nivre et al., 2017) and consists of $(T, S)$ pairs where $S$ is a sentence, and $T$ is the UD dependency tree of $S$ after word information has been removed and tokens have been lemmatised. The languages are those shown in Table 1 and the size of the datasets (training, dev and test) varies between 7,586 (Arabic) and 85,377 (Czech) instances with most languages having around 12K instances (for more details about the data see Mille et al. (2018)).

## 4 Model

As illustrated by Example 1, surface realisation from SR'18 shallow meaning representations can be viewed as consisting of three main steps: word ordering, morphological inflection and contraction generation. For instance, given an unordered dependency tree whose nodes are labelled with lemmas and morphological features (1a)[1], the lemmas must be assigned the appropriate order (1b), they must be inflected (1c) and contractions may take place (1d).

(1) a. the find **be not** meaning of life it about
b. it be not about find the meaning of life
c. It **is n't** about finding the meaning of life
d. It **isn't** about finding the meaning of life

We propose a neural architecture which explicitly integrates these three subtasks as three separate modules into a pipeline: word ordering (WO) is applied first, then morphological realisation (WO+MR) and finally, contractions (WO+MR+C) are handled.

### 4.1 Word Ordering

For word ordering, we combine a factored sequence-to-sequence model with an "extreme delexicalisation" step which replaces matching source and target tokens with an identifier.

[1]Features and tree structures have been omitted.

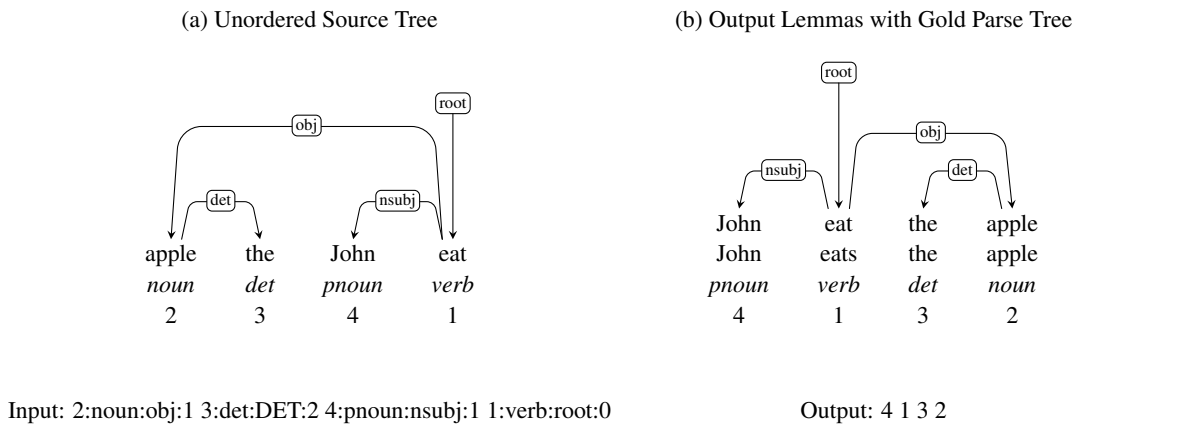


Figure 1: Delexicalising and linearising (in the parse tree of the output sentence the first row shows the lemmas, the second–the word forms, the third–the POS tags and the fourth–the identifiers). Identifiers are assigned to the source tree nodes in the order given by depth-first search.

**Delexicalisation.** Delexicalisation has frequently been used in neural NLG to help handle unknown or rare items (Wen et al., 2015; Dušek and Jurcicek, 2015; Chen et al., 2018). Rare items are replaced by placeholders both in the input and in the output; models are trained on the delexicalised data; and a post-processing step ensures that the generated text is relexicalised using the placeholders' original value. In these approaches, delexicalisation is restricted to rare items (named entities). In contrast, we apply delexicalisation to all input lemmas. Abstracting away from specific lemmas reduces data sparsity, allows for the generation of rare or unknown words and last but not least, it captures the linguistic intuition that word ordering mainly depends on syntactic information (e.g., in English, the subject generally precedes the verb).

To create the delexicalised data, we need to identify matching input and output elements and to replace them by the same identifier. We also store a mapping (id, L, F) specifying which identifier id refers to which (L, F) pair, where L is a lemma and F is its set of morpho-syntactic features.

We identify matching input and output elements by comparing the unordered input tree provided by the SR'18 task with the parse tree of the output sentence provided by the UD treebanks (cf. Figure 1). Source and target nodes which share the same path to the root are then mapped to the same identifier. For instance, in Figure 1, the lemma "apple" has the same path to the root (obj:eat:root) in both the input and the output tree. Hence the same identifier is assigned to the nodes. More generally, after linearisation through depth-first, left-to-right traversal of the input tree, each training instance captures the mapping between lemmas in the input tree and the same lemmas in the output sequence. For instance, given the example shown in Figure 1, delexicalisation will yield the training instance:

Input: tkn2 tkn3 tkn4 tkn1
Output: tkn4 tkn1 tkn3 tkn2

where $\text{tkn}_i$ is the factored representation (see below) of each delexicalised input node.

**Factored Sequence-to-Sequence Model.** Following Elder and Hokamp (2018), we use a factored model (Alexandrescu and Kirchhoff, 2006) as a means of enriching the node representations input to the neural model. Each delexicalised tree node is modelled by a sequence of features. Separate embeddings are learned for each feature type and the feature embeddings of each input node are concatenated to create its dense representation. As exemplified in Figure 1, we model each input placeholder as a concatenation of four features: the node identifier, its POS tag, its dependency relation to the parent node and its parent identifier[2].

**Sequence-to-sequence model.** We use the OpenNMT-py framework (Klein et al., 2017)[3] to train factored sequence-to-sequence models with attention (Luong et al., 2015) and the copy and coverage mechanisms described in See et al. (2017). A single-layer LSTM is used for both encoder and decoder. We train using full vocabulary

[2]The parent identifier of a root node is represented as 0.

[3]commit e61589d, https://github.com/OpenNMT/OpenNMT-py

and the maximal length in the source and target for both baseline and the proposed model. Models were trained for 20 epochs, with a mini-batch size of 64, a word embedding size of 300, and a hidden unit size of 450. They were optimised with SGD with a starting learning rate of 1.0. A learning rate is halved when perplexity does not decrease on the development set. Preliminary experiments showed that the lowest perplexity was reached on average at epoch 17, so this model was kept for decoding. Decoding is done using beam search with a beam size of 5. For each language, we train three models with different random seeds, and report the average performance and standard deviation.

The model is trained on delexicalised data. At test time, the token/identifier mapping is used to relexicalise the model output.

### 4.2 Morphological Realisation

The morphological realisation (MR) module consists in producing inflected word forms based on lemmas coupled with morphological features. For that module, we used a model recently proposed by Aharoni and Goldberg (2017), which achieves state-of-the-art results on several morphological inflection datasets: the CELEX dataset (Baayen et al., 1993; Dreyer et al., 2008), the Wiktionary dataset (Durrett and DeNero, 2013) and the SIGMORPHON2016 dataset (Cotterell et al., 2016). Their model is based on a neural encoder-decoder architecture with hard monotonic attention and performs out-of-context morphologic realisation: given a lemma and a set of morpho-syntactic features, it produces a corresponding word form.

We trained the model of Aharoni and Goldberg (2017) on (lemma+morpho-syntactic_features[4], form) pairs extracted from the SR'18 training data. We trained the model for 20 epochs with the default parameters provided in the implementation[5].

In our pipeline architecture, morphological realisation is applied to the output of our word ordering model using the (id, L, F) mapping mentioned above. For each delexicalised token produced by the word ordering component, we retrieve the corresponding lemma and morpho-syntactic features (L, F) and apply our MR model to it so as to produce the corresponding word form.

While associating a lemma and its features to a corresponding form, the MR module operates without taking context into account, so it cannot perform some finer grained operations, such as contraction, elision, and clitic attachment. We address that issue in the following section.

### 4.3 Contraction Generation

Contraction handling is the last step of our surface realisation pipeline. Example 2 shows some types of contractions.

(2) French: "Le chat dort." / "L'alouette chante." (Elision for the definite article *le* before a vowel: Le → L')

Italian: *"In il mare." → "Nel mare." (Contraction of the preposition *in* and the article *il*: In il → Nel)

Portuguese: *"Eis lo." → "Ei-lo." (Clitic pronoun attachment: Eis lo → Ei-lo)

We developed two modules for the contraction generation: one based on regular expressions ($C_{reg}$) and another based on a sequence-to-sequence model ($C_{s2s}$).

The sequence-to-sequence model is trained on pairs of sentences without and with contractions. The sentence with contraction ($S^{+c}$) is the final sentence, i.e., the reference sentence in the SR'18 data. The sentence without contraction ($S^{-c}$) is the corresponding sequence of word forms extracted from the UD CoNLL data.

The regular expression module is inspired by the decomposition of multi-word expressions, such as contractions, which is applied during the tokenisation step in parsing (Martins et al., 2009). We reversed the regular expressions given in the TurboParser[6] for the surface realisation task, and also added our own to tackle, for example, elision in French. $C_{s2s}$ and $C_{reg}$ modules were created for three languages: French, Italian, and Portuguese[7].

## 5 Evaluation

We evaluate each component of our approach separately. We start by providing a detailed evaluation of how the model handles word ordering

[4] POS and morphological features

[5] https://github.com/roeeaharoni/morphological-reinflection/blob/master/src/hard_attention.py

[6] https://github.com/andre-martins/TurboParser/tree/master/python/tokenizer

[7] Although contractions are also present in Spanish, we did not develop a module for it, since the UD Spanish AnCora treebank does not split them on the token level in contrast to other UD treebanks.

| | ar | cs | en | es | fi | fr | it | nl | pt | ru |
|---|---|---|---|---|---|---|---|---|---|---|
| BL | 29.6±1.39 | 48±0.7 | 53.57±0.15 | 46.5±0.78 | 27.2±0.4 | 46.4±0.5 | 49.07±0.9 | 36.6±0.7 | 44.3±0.26 | 58.1±0.46 |
| WO | 34.9±0.2 | 57.97±0.06 | 59.1±0.36 | 52.33±0.31 | 43.1±0.53 | 50.0±0.0 | 53.17±0.5 | 47.03±0.59 | 51.77±0.32 | 64.73±0.23 |
| Δ | +5.3 | +9.97 | +5.53 | +5.83 | +15.9 | +3.6 | +4.1 | +10.43 | +7.47 | +6.63 |

Table 1: Word Ordering: BLEU scores on lemmatised data. Mean and standard deviation across three random seeds. BL: Baseline. All pairwise comparisons of BL and our model showed a statistically significant difference in BLEU via the bootstrap resampling (1000 samples, $p < .05$).

(Section 5.1). We then go on to analyse the respective contributions of morphological realisation (Section 5.2) and contraction generation (Section 5.3). Finally, we discuss the performance of the overall surface realisation model (Section 5.4). Throughout the evaluation, we used the SR'18 evaluation scripts to compute automatic metrics[8].

## 5.1 Word Ordering

### 5.1.1 BLEU scores

We evaluate our word ordering component by computing the BLEU-4 score (Papineni et al., 2002) between the sequence of lemmas it produces and the lemmatized reference sentence extracted from the UD files. The baseline is the same model without delexicalisation. As Table 1 shows, there is a marked, statistically significant, difference between the baseline and our approach which indicates that delexicalisation does improve word ordering.

### 5.1.2 Word Ordering Constraints

We also investigate the degree to which our results conform with the word ordering constraints of the various languages focusing on the following dependency relations: DET (determiner), NSUBJ (nominal subject), OBJ (object), AMOD (adjectival modifier) and ACL (nominal clausal modifier). For each of these dependency relations, we compare the relative ordering of the corresponding (head, dependent) pairs in the reference data and in our system predictions.

To determine whether the dependent should precede or follow its head, we use the gold standard dependency tree of the UD treebanks. Since for the system predictions we do not have a parse tree, we additionally record the distance between head and dependent (in the reference data) and we compare it with the distance between the same two items in the system output. For instance, for the DET relation, given the gold sentence (3a) and the generated sentence (3b), we extract (3c) from the UD parse tree and (3d) from the predicted sentence where each triple is of the form either (dep, head, distance) or (head, dep, distance) and distance is the distance between head and dependent.

(3) a. GOLD: The yogi tried the advanced asana
b. PRED: The yogi tried the asana advanced
c. G-triples: (the$_{dep}$, yogi$_{head}$, 1), (the$_{dep}$, asana$_{head}$, 2)
d. P-triples: (the, yogi, 1), (the, asana, 1)
Exact match: 1; Approximate match: 2

We then compute exact matches (the order and the distance to the head is exactly the same) and approximate matches (the order is preserved but the distance differs by 1 token[9]). Table 2 shows the results and compares them with a non-delexicalised approach.

**Global Score.** The **all deprels** column summarises the scores for all dependency relations present in the treebanks (not just DET, NSUBJ, OBJ, AMOD and ACL). For the exact match, most languages score above average (from 0.51 to 0.71). That is the relative word order and the position of the dependent with respect to the head is correctly predicted in more than half of the cases. Approximate match yields higher scores with most languages scoring between 0.65 and 0.80 suggesting that a higher proportion of correct relative orderings is achieved (modulo mispositioning and false positives).

**Long Range Dependencies.** It is noticeable that for all languages, accuracy drops for the ACL relation. We conjecture that two factors makes it difficult for the model to make the correct prediction: heterogeneity and long range dependencies. As the ACL relation captures different types of clausal modifiers (finite and non-finite), it is harder for the model to learn the corresponding patterns. As the modifier is a clause, the distance between head

[8] http://taln.upf.edu/pages/msr2018-ws/SRST.html#evaluation

[9] We could of course consider further approximates matches differing by, say 2, 3 or 5 tokens. But we refrain from this as this would increase the number of false positives.

| | *det* | | *nsubj* | | *obj* | | *amod* | | *acl* | | ***all deprels*** | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | +1 | +2 | +1 | +2 | +1 | +2 | +1 | +2 | +1 | +2 | +1 | +2 |
| ar | 0.36 | 0.37 | 0.45 | 0.52 | 0.35 | 0.48 | 0.52 | 0.59 | 0.32 | 0.41 | 0.38 | 0.47 |
| Δ | +0.036 | +0.047 | -0.036 | -0.067 | -0.069 | -0.113 | -0.088 | -0.106 | -0.063 | -0.061 | -0.044 | -0.071 |
| cs | 0.86 | 0.9 | 0.49 | 0.63 | 0.51 | 0.64 | 0.83 | 0.87 | 0.47 | 0.62 | 0.63 | 0.74 |
| Δ | -0.077 | -0.071 | -0.041 | -0.054 | -0.048 | -0.053 | -0.134 | -0.126 | -0.071 | -0.073 | -0.068 | -0.074 |
| en | 0.76 | 0.85 | 0.71 | 0.85 | 0.76 | 0.84 | 0.71 | 0.76 | 0.53 | 0.65 | 0.63 | 0.74 |
| Δ | -0.10 | -0.086 | -0.002 | -0.053 | -0.074 | -0.064 | -0.15 | -0.138 | -0.058 | -0.104 | -0.06 | -0.08 |
| es | 0.73 | 0.83 | 0.55 | 0.71 | 0.54 | 0.70 | 0.43 | 0.49 | 0.39 | 0.58 | 0.55 | 0.68 |
| Δ | -0.02 | -0.074 | -0.012 | -0.073 | +0.004 | -0.057 | +0.059 | +0.036 | -0.056 | -0.12 | -0.038 | -0.088 |
| fi | 0.71 | 0.81 | 0.64 | 0.75 | 0.46 | 0.60 | 0.70 | 0.76 | 0.50 | 0.61 | 0.51 | 0.65 |
| Δ | -0.241 | -0.244 | -0.194 | -0.189 | -0.118 | -0.137 | -0.303 | -0.296 | -0.261 | -0.312 | -0.154 | -0.16 |
| fr | 0.76 | 0.86 | 0.60 | 0.78 | 0.60 | 0.75 | 0.46 | 0.51 | 0.51 | 0.68 | 0.58 | 0.71 |
| Δ | -0.016 | -0.235 | +0.014 | -0.037 | +0.031 | -0.029 | +0.117 | +0.104 | -0.1 | 0.17 | -0.038 | -0.093 |
| it | 0.73 | 0.82 | 0.59 | 0.70 | 0.58 | 0.73 | 0.40 | 0.46 | 0.52 | 0.65 | 0.56 | 0.69 |
| Δ | -0.022 | -0.058 | -0.021 | -0.07 | -0.044 | -0.101 | +0.058 | +0.017 | -0.054 | -0.072 | -0.04 | -0.088 |
| nl | 0.71 | 0.79 | 0.46 | 0.56 | 0.38 | 0.49 | 0.74 | 0.77 | 0.41 | 0.53 | 0.49 | 0.60 |
| Δ | -0.121 | -0.115 | -0.068 | -0.07 | -0.068 | -0.087 | -0.278 | -0.29 | -0.22 | -0.277 | -0.118 | -0.13 |
| pt | 0.74 | 0.80 | 0.56 | 0.72 | 0.57 | 0.73 | 0.42 | 0.44 | 0.48 | 0.63 | 0.54 | 0.67 |
| Δ | -0.032 | -0.051 | -0.072 | -0.114 | -0.041 | -0.094 | +0.034 | +0.043 | -0.122 | -0.204 | -0.068 | -0.103 |
| ru | NA | NA | 0.65 | 0.79 | 0.65 | 0.74 | 0.79 | 0.85 | 0.45 | 0.67 | 0.71 | 0.80 |
| Δ | — | — | -0.033 | -0.057 | -0.037 | -0.033 | -0.095 | -0.1 | +0.022 | -0.027 | -0.05 | -0.059 |

Table 2: Proportion of correct head/dependent positioning for the five selected dependency relations: *det*, *nsubj*, *obj*, *amod*, *acl*, and overall performance across all dependency relations. +1: exact match; +2: approximate match, i.e. head and dependent are in the correct order but there is a one-token difference between gold and prediction. NA: no dependency relation found in a treebank. Δ indicates the difference between our delexicalised model and the baseline.

| | ar | cs | en | es | fi | fr | it | nl | pt | ru |
|---|---|---|---|---|---|---|---|---|---|---|
| MR Accuracy | 91.05 | 98.89 | 98.3 | 99.07 | 92.66 | 96.77 | 96.85 | 87.6 | 98.80 | 98.23 |
| WO | 34.9±0.2 | 57.97±0.06 | 59.1±0.36 | 52.33±0.31 | 43.1±0.53 | 50.0±0.0 | 53.17±0.5 | 47.03±0.59 | 51.77±0.32 | 64.73±0.23 |
| WO+MR ($S^{-c}$) | 28.6±0.26 | 56.1±0.1 | 54.3±0.3 | 51.4±0.3 | 38.63±0.59 | 44.97±0.25 | 47.3±0.6 | 42.3±0.53 | 50.7±0.26 | 60.93±0.23 |
| Δ | -6.3 | -1.87 | -4.8 | -0.93 | -4.47 | -5.03 | -5.87 | -4.73 | -1.07 | -3.8 |

Table 3: Morphological Realisation Results. MR Accuracy: accuracy of the MR module. WO: BLEU scores on lemmas. WO+MR: BLEU scores on inflected tokens.

(the nominal being modified) and dependent (the verb of the clause modifier) can be long which again is likely to impede learning.

**Irregular Order.** For cases where the head/dependent order is irregular, the scores are lower. For instance, in Dutch the object may occur either before (46.9% of the cases in the test data) or after the verb depending on whether it occurs in a subordinate or a main clause. Relatedly, the OBJ exact match score is the lowest (0.38) for this language. Similarly, in Romance languages where the adjective (AMOD relation) can either be pre- (head-final construction, HF) or post-posed (head-initial construction, HI), exact match scores are lower for this relation than for the others. For instance, the Portuguese test data contains 71% HF and 29% HI occurrences of the AMOD relation and correspondingly, the scores for that relation are much lower than for the DET, NSUBJ and OBJ relations for that language. A similar pattern can be observed for Spanish, French and Italian.

More detailed statistics, including other relations and performance with respect to head-directionality, can be found in the supplementary material.

**Non-delexicalised Baseline.** We also compare our delexicalised model with the non-delexicalised baseline: Δ in Table 2 shows the difference in performance between the two models.

Overall, the scores favour the delexicalised approach (negative $delta$ in the **all deprels** column for all languages) supporting the results given by the automatic metric. However, for some dependency relations, the lexicalised baseline shows usefulness of word information, for ex-

| | ar | cs | en | es | fi | fr | it | nl | pt | ru |
|---|---|---|---|---|---|---|---|---|---|---|
| $S^{-c}/S^{+c}$ | 47.1 | 96.1 | 90.9 | 98.4 | 98.5 | 70.8 | 65.1 | 99.5 | 66.3 | 96.9 |
| WO+MR ($S^{-c}$) | 28.6±0.26 | 56.1±0.1 | 54.3±0.3 | 51.4±0.3 | 38.63±0.59 | 44.97±0.25 | 47.3±0.6 | 42.3±0.53 | 50.7±0.26 | 60.93±0.23 |
| WO+MR ($S^{+c}$) | 15.8±0.1 | 54.83±0.21 | 53.3±0.53 | 51.03±0.31 | 38.37±0.55 | 36.23±0.42 | 31.5±0.4 | 42.27±0.55 | 34.0±0.62 | 60.43±0.15 |
| $\Delta$ | -12.8 | -1.27 | -1.0 | -0.37 | -0.26 | -8.74 | -15.8 | -0.03 | -16.7 | -0.5 |
| WO+MR+C$_{reg}$ ($S^{+c}$) | | | | | | 41.8±0.26 | 40.0±0.66 | | 46.13±0.29 | |
| WO+MR+C$_{s2s}$ ($S^{+c}$) | | | | | | 40.33±0.36 | 39.93±0.17 | | 44.57±0.64 | |

Table 4: Contraction Generation Results (BLEU scores). $S^{-c}/S^{+c}$: a sentence without contractions vs. a reference sentence including contractions; $S^{-c}$: BLEU with respect to sentences before contractions; $S^{+c}$: BLEU with respect to a reference sentence. The scores were computed on detokenised sequences.

ample, while predicting AMOD relations for Romance languages (positive $delta$ for French, Italian, Spanish, and Portuguese). Indeed, preposed adjectives in those languages constitute a limited lexical group.

### 5.2 Morphological Realisation

Table 3 shows the results for the WO+MR model.

The top line (MR Accuracy) indicates the accuracy of the MR model on the SR'18 test data which is computed by comparing its output with gold word forms. As the table shows, the accuracy is very high overall ranging from 87.6 to 99.07, with 9 of the 10 languages having an accuracy above 90. This confirms the high accuracy of the model when performing morphological inflection out of context.

The third line (WO+MR ($S^{-c}$)) shows the BLEU scores for our WO+MR model, i.e., when the MR model is applied to the output of the WO model. Here we use an oracle setting which ignores contractions. That is, we compare the WO+MR output not with the final sentence but with the sentence before contraction applies (the ability to handle contractions is investigated in the next section).

As the table shows, the delta in BLEU scores between the model with (WO+MR) and without (WO) morphological realisation mirrors the accuracy of the morphological realisation model: as the accuracy of the morphological inflection model decreases, the delta increases. For instance, for Arabic, the MR accuracy is among the lowest (91.05) and, correspondingly, the decrease in BLEU score when going from word ordering to word ordering with morphological realisation is the largest (-6.3).

### 5.3 Contraction Generation

To assess the degree to which contractions are used, we compute BLEU-4 between the gold sequence of word forms from UD treebanks and the reference sentence (Table 4, Line $S^{-c}/S^{+c}$). As the table shows, this BLEU score is very low for some languages (Arabic, French, Italian, Portuguese) indicating a high level of contractions.

These differences are reflected in the results of our WO+MR model: the higher the level of contractions, the stronger the delta between the BLEU score on the reference sentence without contractions (WO+MR, $S^{-c}$) and the reference sentence with contractions (WO+MR, $S^{+c}$).

This shows the limits of out-of-context morphological realisation. While the model is good at producing a word form given its lemma and a set of morpho-syntactic features, the lack of contextual information means that contractions cannot be handled.

Adding a contraction module permits improving results for those languages where contraction is frequent (Table 4, Lines WO+MR+C$_{reg}$, WO+MR+C$_{s2s}$). Gains range from +5 points for French to +12 for Portuguese when comparing to WO+MR. We achieved better results with contraction module based on regular expressions (C$_{reg}$), rather than a neural module (C$_{s2s}$). In a relatively simple task, such as contraction generation, rule-based methods are more reliable, and, overall, are preferable due to their robustness and easy repair comparing to neural models, which may, for instance, hallucinate incorrect content.

### 5.4 Global Evaluation

Finally, we compare our approach with the best results obtained by the SR'18 participants and with OSU's results (King and White, 2018) using BLEU-4, DIST and NIST scores. OSU results are treated separately, since some of their scores were published after the shared task had ended. Table 5 shows the results. They are mixed. Our model yields the best results for Czech (BLEU: +1.63), Finnish (BLEU: +0.87), Dutch (BLEU: +9.99) and Russian (BLEU: +2.53). However it underperforms on Arabic (BLEU: -9.8), English

| | ar | cs | en | es | fi | fr | it | nl | pt | ru |
|---|---|---|---|---|---|---|---|---|---|---|
| BLEU | | | | | | | | | | |
| SR'18 | 16.2 | 25.05 | 55.29 | 49.47 | 23.26 | **52.03** | **44.46** | 32.28 | 30.82 | 34.34 |
| OSU | **25.6** | 53.2 | **66.30** | **65.30** | 37.5 | 38.2 | 42.1 | 25.5 | **47.6** | 57.9 |
| Ours | 15.8±0.1 | **54.83±0.21** | 53.3±0.53 | 51.03±0.31 | **38.37±0.55** | 41.8±0.26 | 40.0±0.66 | **42.27±0.55** | 46.13±0.29 | **60.43±0.15** |
| DIST | | | | | | | | | | |
| SR'18 | 44.37 | 36.48 | 79.29 | 51.73 | 41.21 | 55.54 | 58.61 | 57.81 | 60.7 | 34.56 |
| OSU | 46.7 | 58.1 | 70.2 | 61.5 | 58.7 | 53.7 | 59.7 | 57.8 | 66.0 | 59.9 |
| Ours | 27.63±0.06 | 63.53±0.15 | 62.77±0.15 | 61.33±0.15 | 51.83±0.55 | 55.23±0.72 | 53.73±0.21 | 54.13±0.15 | 57.0±0.4 | 71.23±0.12 |
| NIST | | | | | | | | | | |
| SR'18 | 7.15 | 10.74 | 10.86 | 11.12 | 9.36 | 9.85 | 9.11 | 8.64 | 7.55 | 13.06 |
| OSU | 7.15 | 13.5 | 12.0 | 12.7 | 9.56 | 8.00 | 8.70 | 7.33 | 9.13 | 14.2 |
| Ours | 6.04±0.02 | 13.6±0.05 | 10.95±0.09 | 11.63±0.02 | 10.41±0.03 | 8.93±0.08 | 8.99±0.01 | 9.51±0.03 | 9.44±0.04 | 13.96±0.03 |

Table 5: BLEU, DIST and NIST scores on the SR'18 test data (shallow track). SR'18 is the official results of the shared task but do not include OSU scores, since they are given in the line below. We also excluded the ADAPT and NILC scores as they were obtained using data augmentation. OSU is the submission of King and White (2018).

(BLEU: -13), Spanish (BLEU: -14.27), French (BLEU: -10.23), Italian (BLEU: -4.46) and Portuguese (BLEU: -1.47). Based on the evaluation of each of our modules, these results can be explained as follows.

The languages for which our model outperforms the state of the art are languages for which the WO model performs best, the accuracy of the morphological realiser is high and the level of contractions is low. For those languages, improving the accuracy of the word ordering model would further improve results.

For four of the languages where the model underperforms (namely, Arabic, French, Portuguese and Italian), the level of contraction is high. This indicates that improvements can be gained by improving the handling of contractions, e.g., by learning a joint model that would take into account both morphological inflection and contraction.

## 6 Conclusion

While surface realisation is a key component of NLG applications, most work in this domain has focused on the development of language specific models. By providing multi-lingual training and test set, the SR'18 shared task opens up the possibility to investigate how language specific properties such as word order and morphological variation impact performance.

In this paper, we presented a modular approach to surface realisation and applied it to the ten languages of the SR'18 shallow track.

For word ordering, we proposed a simple approach where the data is delexicalised, the input tree is linearised using depth-first search and the mapping between input tree and output lemma sequence is learned using a factored sequence-to-sequence model. Experimental results show that full delexicalisation markedly improves performance. Linguistically, this confirms the intuition that the mapping between shallow dependency structure and word order can be learned independently of the specific words involved.

We further carried out a detailed evaluation of how our word ordering model performs on the ten languages of the SR'18 shallow track. While differences in annotation consistency, number of dependency relations[10] and frequency counts for each dependency relations in each dataset make it difficult to conclude anything from the differences in overall scores between languages, the evaluation of head/dependent word ordering constraints highlighted the fact that long-distance relations, such as ACL, and irregular word ordering constraints (e.g., the position of the verb in Dutch main and subordinate clauses) negatively impact results.

For morphological realisation and contractions, we showed that applying morphological realisation out of context, as is done by most of the SR'18 participating systems[11], yields poor results for those languages (Portuguese, French, Arabic, Italian) where contractions are frequent. We explored two ways of handling contractions (a neural sequence-to-sequence model and a rule-based model) and showed that adding contraction han-

[10]The number of distinct dependency relations present in the treebank ranges between 29 (ar, es) and 44 (en).

[11]The only exception is Castro Ferreira et al. (2018) who train an SMT model on pairs of lemmatised/non-lemmatised sentences.

dling strongly improves performance (from +5.57 to 12.13 increase in BLEU score for the rule-based model depending on the language). More generally, our work on contractions points to the need for SR models to better take into account the fine-grained structure of words. For instance, in French, the article is elided (*le* $\rightarrow$ *l'*) when the following word starts with a vowel. In future work, we plan to explore the development of a joint model that simultaneously handles morphological realisation and word ordering while using finer grained word representations, such as fastText embeddings (Bojanowski et al., 2017) or byte pair encoding (BPE; Gage, 1994; Sennrich et al., 2016).

## References


Roee Aharoni and Yoav Goldberg. 2017. Morphological inflection generation with hard monotonic attention. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2004–2015, Vancouver, Canada. Association for Computational Linguistics.

Andrei Alexandrescu and Katrin Kirchhoff. 2006. Factored neural language models. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 1–4. Association for Computational Linguistics.

R Harald Baayen, Richard Piepenbrock, and Leon Gulikers. 1993. The CELEX lexical database (CD-ROM).

Valerio Basile and Alessandro Mazzei. 2018. The dipinfo-unito system for srst 2018. In *Proceedings of the First Workshop on Multilingual Surface Realisation*, pages 65–71. Association for Computational Linguistics.

Anja Belz, Mike White, Dominic Espinosa, Eric Kow, Deirdre Hogan, and Amanda Stent. 2011. The first surface realisation shared task: Overview and evaluation results. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 217–226. Association for Computational Linguistics.

Bernd Bohnet, Leo Wanner, Simon Mille, and Alicia Burga. 2010. Broad coverage multilingual deep sentence generation with a stochastic multi-level realizer. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 98–106. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Thiago Castro Ferreira, Sander Wubben, and Emiel Krahmer. 2018. Surface realization shared task 2018 (sr18): The tilburg university approach. In *Proceedings of the First Workshop on Multilingual Surface Realisation*, pages 35–38. Association for Computational Linguistics.

Mingje Chen, Gerasimos Lampouras, and Andreas Vlachos. 2018. Sheffield at e2e: structured prediction approaches to end-to-end language generation. Technical report, E2E Challenge System Descriptions.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016. The SIGMORPHON 2016 shared Task—Morphological reinflection. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 10–22, Berlin, Germany. Association for Computational Linguistics.

Markus Dreyer, Jason Smith, and Jason Eisner. 2008. Latent-variable modeling of string transductions with finite-state methods. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 1080–1089, Honolulu, Hawaii. Association for Computational Linguistics.

Greg Durrett and John DeNero. 2013. Supervised learning of complete morphological paradigms. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1185–1195, Atlanta, Georgia. Association for Computational Linguistics.

Ondřej Dušek and Filip Jurcicek. 2015. Training a natural language generator from unaligned data. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 451–461. Association for Computational Linguistics.

Henry Elder and Chris Hokamp. 2018. Generating high-quality surface realizations using data augmentation and factored sequence models. In *Proceedings of the First Workshop on Multilingual Surface Realisation*, pages 49–53. Association for Computational Linguistics.

Philip Gage. 1994. A new algorithm for data compression. *C Users J.*, 12(2):23–38.

David King and Michael White. 2018. The osu realizer for srst '18: Neural sequence-to-sequence inflection and incremental locality-based linearization. In *Proceedings of the First Workshop on Multilingual Surface Realisation*, pages 39–48. Association for Computational Linguistics.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. Opennmt:

Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72. Association for Computational Linguistics.

Kexin Liao, Logan Lebanoff, and Fei Liu. 2018. Abstract meaning representation for multi-document summarization. *arXiv preprint arXiv:1806.05655*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421. Association for Computational Linguistics.

Andreas Madsack, Johanna Heininger, Nyamsuren Davaasambuu, Vitaliia Voronik, Michael Käufl, and Robert Weißgraeber. 2018. Ax semantics' submission to the surface realization shared task 2018. In *Proceedings of the First Workshop on Multilingual Surface Realisation*, pages 54–57. Association for Computational Linguistics.

Diego Marcheggiani and Laura Perez-Beltrachini. 2018. Deep graph convolutional encoders for structured data to text generation. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 1–9, Tilburg University, The Netherlands. Association for Computational Linguistics.

André Martins, Noah Smith, and Eric Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 342–350, Suntec, Singapore. Association for Computational Linguistics.

Simon Mille, Anja Belz, Bernd Bohnet, Yvette Graham, Emily Pitler, and Leo Wanner. 2018. The first multilingual surface realisation shared task (sr'18): Overview and evaluation results. In *Proceedings of the First Workshop on Multilingual Surface Realisation*, pages 1–12. Association for Computational Linguistics.

Joakim Nivre, Željko Agić, Lars Ahrenberg, Maria Jesus Aranzabe, Masayuki Asahara, Aitziber Atutxa, Miguel Ballesteros, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Eckhard Bick, Cristina Bosco, Gosse Bouma, Sam Bowman, Marie Candito, Gülşen Cebiroğlu Eryiğit, Giuseppe G. A. Celano, Fabricio Chalub, Jinho Choi, Çağrı Çöltekin, Miriam Connor, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Arantza Diaz de Ilarraza, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Marhaba Eli, Tomaž Erjavec, Richárd Farkas, Jennifer Foster, Cláudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökırmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Nizar Habash, Jan Hajič, Linh Hà Mỹ, Dag Haug, Barbora Hladká, Petter Hohle, Radu Ion, Elena Irimia, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşıkara, Hiroshi Kanayama, Jenna Kanerva, Natalia Kotsyba, Simon Krek, Veronika Laippala, Phng Lê H`ông, Alessandro Lenci, Nikola Ljubešić, Olga Lyashevskaya, Teresa Lynn, Aibek Makazhanov, Christopher Manning, Cătălina Mărănduc, David Mareček, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Anna Missilä, Verginica Mititelu, Yusuke Miyao, Simonetta Montemagni, Amir More, Shunsuke Mori, Bohdan Moskalevskyi, Kadri Muischnek, Nina Mustafina, Kaili Müürisep, Lng Nguy˜ên Thị, Huy`ên Nguy˜ên Thị Minh, Vitaly Nikolaev, Hanna Nurmi, Stina Ojala, Petya Osenova, Lilja Øvrelid, Elena Pascual, Marco Passarotti, Cenel-Augusto Perez, Guy Perrier, Slav Petrov, Jussi Piitulainen, Barbara Plank, Martin Popel, Lauma Pretkalniņa, Prokopis Prokopidis, Tiina Puolakainen, Sampo Pyysalo, Alexandre Rademaker, Loganathan Ramasamy, Livy Real, Laura Rituma, Rudolf Rosa, Shadi Saleh, Manuela Sanguinetti, Baiba Saulīte, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Lena Shakurova, Mo Shen, Dmitry Sichinava, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Dima Taji, Takaaki Tanaka, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Larraitz Uria, Gertjan van Noord, Viktor Varga, Veronika Vincze, Jonathan North Washington, Zdeněk Žabokrtský, Amir Zeldes, Daniel Zeman, and Hanzhi Zhu. 2017. Universal dependencies 2.0. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*.

Ratish Puduppully, Yue Zhang, and Manish Shrivastava. 2017. Transition-based deep input linearization. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 643–654, Valencia, Spain. Association for Computational Linguistics.

Yevgeniy Puzikov and Iryna Gurevych. 2018. Binlin: A simple method of dependency tree linearization. In *Proceedings of the First Workshop on Multilingual Surface Realisation*, pages 13–28. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational*

*Linguistics (Volume 1: Long Papers)*, pages 1073–1083. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Shreyansh Singh, Ayush Sharma, Avi Chawla, and A.K. Singh. 2018. Iit (bhu) varanasi at msr-srst 2018: A language model based approach for natural language generation. In *Proceedings of the First Workshop on Multilingual Surface Realisation*, pages 29–34. Association for Computational Linguistics.

Marco Antonio Sobrevilla Cabezudo and Thiago Pardo. 2018. Nilc-swornemo at the surface realization shared task: Exploring syntax-based word ordering using neural models. In *Proceedings of the First Workshop on Multilingual Surface Realisation*, pages 58–64. Association for Computational Linguistics.

Linfeng Song, Yue Zhang, Kai Song, and Qun Liu. 2014. Joint morphological generation and syntactic linearization. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.

Sho Takase, Jun Suzuki, Naoaki Okazaki, Tsutomu Hirao, and Masaaki Nagata. 2016. Neural headline generation on abstract meaning representation. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 1054–1059.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721. Association for Computational Linguistics.

# IMaT: Unsupervised Text Attribute Transfer via Iterative Matching and Translation

**Zhijing Jin***
University of Hong Kong
zhijing.jin@connect.hku.hk

**Di Jin***
MIT
jindi15@mit.edu

**Jonas Mueller**
Amazon Web Services
jonasmue@amazon.com

**Nicholas Matthews**
ASAPP Inc.
nmatthews@asapp.com

**Enrico Santus**
MIT
esantus@csail.mit.edu

## Abstract

Text attribute transfer aims to automatically rewrite sentences such that they possess certain linguistic attributes, while simultaneously preserving their semantic content. This task remains challenging due to a lack of supervised parallel data. Existing approaches try to explicitly disentangle content and attribute information, but this is difficult and often results in poor content-preservation and ungrammaticality. In contrast, we propose a simpler approach, **I**terative **M**atching **a**nd **T**ranslation (**IMaT**), which: (1) constructs a pseudo-parallel corpus by aligning a subset of semantically similar sentences from the source and the target corpora; (2) applies a standard sequence-to-sequence model to learn the attribute transfer; (3) iteratively improves the learned transfer function by refining imperfections in the alignment. In sentiment modification and formality transfer tasks, our method outperforms complex state-of-the-art systems by a large margin. As an auxiliary contribution, we produce a publicly-available test set with human-generated transfer references.[1].

## 1 Introduction

An intelligent natural language generation (NLG) system should be able to flexibly control linguistic attributes of the text it generates. For instance, a certain level of formality should be maintained in serious situations, while informal conversations can be improved through a more relaxed style. The ability to generate or rewrite a certain text with some attributes controlled or transferred to meet pragmatic goals is widely needed in applications such as dialogue generation (Oraby et al., 2018), author obfuscation (Kacmarcik and Gamon, 2006; Juola and Vescovi, 2011), and written communication assistance (Pavlick and Tetreault, 2016). One typical example of a text attribute is the linguistic style, which refers to features of lexicons, syntax and phonology that collectively contribute to the identifiability of an instance of language (Gatt and Krahmer, 2018). For instance, given text written in an older style like the Shakespearean genre, a system may be tasked to convert it into modern-day colloquial English (Kabbara and Cheung, 2016).

In general, a text attribute transfer system must be able to: (1) produce sentences that conform to the target attribute, (2) preserve the source sentence's content, and (3) generate fluent language. Satisfying these requirements is challenging due to the typical absence of supervised parallel corpora exemplifying the desired attribute transformations. With access to only non-parallel data, most existing approaches aim to tease apart content and attribute in a latent sentence representation. For instance, Shen et al. (2017) and Fu et al. (2018) utilize Generative Adversarial Networks (GANs) to learn this separation. Although these elaborate models can produce outputs that conform to the target attribute (requirement 1), their sentences are fraught with errors in terms of both content preservation and linguistic fluency. This is exemplified in Table 1, where the model of Shen

| | Positive Sentiment ↔ Negative Sentiment |
|---|---|
| **Input** | I love this place, the service is always great! |
| CA | I know this place, the food is just a horrible! |
| MD | I love this place, the service is always great! |
| DAR | I did not like the homework of lasagna, not like it, . |
| **Ours** | I used to love this place, but the service is horrible now. |

Table 1: Comparing outputs of transfer models. Our approach both transfers sentiment and preserves content. *CrossAlignment* (CA) by Shen et al. (2017) loses content, *MultiDecoder* (MD) by Fu et al. (2018) fails to modify sentiment, and *DeleteAndRetrieve* (DAR) by Li et al. (2018) produces ungrammatical output.

*Equal Contribution

[1]The enriched test set is available at https://github.com/zhijing-jin/IMaT



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3097–3109,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

et al. (2017) changes key content words from "the service" to the unrelated "the food". In an effort to avoid the issues of GAN/autoencoder approaches, Li et al. (2018) recently demonstrated that direct implementation of heuristic transformations– such as modifying high polarity words in reviews for sentiment transfer– can produce better results. Their work suggests attribute transfer can be addressed through simpler methods that avoid attempting to disentangle attribute and content information at the representation level. However, the proposed heuristic transformations are a bit too simple, relying on rule-based sentence reconstruction that often produces linguistically unnatural or ungrammatical outputs. Even worse, some unrelated words happen to be wrongly inserted by approach of Li et al. (2018), dramatically upsetting the sentence content as seen in Table 1.

In this paper, we propose the Iterative Matching and Translation (IMaT) framework which addresses the aforementioned limitations with regards to content inconsistency and ungrammaticality. Our approach first constructs a pseudo-parallel corpus by **matching** a subset of semantically similar sentences from the source and the target corpora (which possess different attributes), then applies a standard sequence-to-sequence (Seq2Seq) **translation** model (Klein et al., 2017) to learn the attribute transfer. We then use the translation results of the trained Seq2Seq model (from the previous iteration) to update the previously made pseudo-parallel corpus, so as to **refine** its quality. Such a matching-translation-refine procedure is iterated repeatedly until performance plateaus. The proposed methodology is simpler and more robust than previous GAN/autoencoder techniques, and is free of manual heuristics such as those used by Li et al. (2018).

We validate our Iterative Matching and Translation (IMaT) model in two attribute-controlled text rewriting tasks that aim to alter: the sentiment of YELP reviews, and the formality of text in the FORMALITY dataset. Both human and automatic evaluations suggest our method substantially outperforms alternative approaches (Shen et al., 2017; Fu et al., 2018; Li et al., 2018) in terms of: accuracy of attribute change, content preservation, and grammaticality. Our main contributions include:

- We propose a novel iterative matching and translation framework that is more straightforward than many existing approaches, by simply adapting a sequence-to-sequence model to perform the attribute transfer.
- We achieve state-of-the-art performance on two text rewriting tasks involving sentiment modification task and formality conversion.
- We release an additional set of 800 sentences rewritten by humans tasked with the sentiment transfer task for the YELP review test set. This enables future researchers to evaluate more diverse transfer outputs.

## 2 Related Work

Attribute-controlled text rewriting remains a long-standing problem in NLG, where most work has focused on studying the stylistic variation in text (Gatt and Krahmer, 2018). Early contributions in this area defined stylistic features using rules to vary generation (Brooke et al., 2010). For instance, Sheikha and Inkpen (2011) proposed an adaptation of the SimpleNLG realiser (Gatt et al., 2009) to handle formal versus informal language via constructing lexicons of formality (e.g., are not vs. aren't). More contemporary approaches have tended to eschew rules in favour of data-driven methods to identify relevant linguistic features to stylistic attributes (Ballesteros et al., 2015; Di Fabbrizio et al., 2008; Krahmer and van Deemter, 2012). For example, Mairesse and Walker's PERSONAGE system (Mairesse and Walker, 2011) uses machine-learning models to take as inputs a list of real-valued style parameters and generate sentences to project different personality traits.

In the past few years, attribute-controlled NLG has witnessed renewed interest by researchers working on neural approaches to generation (Hu et al., 2017; Jhamtani et al., 2017; Melnyk et al., 2017; Mueller et al., 2017; Zhang et al., 2018; Prabhumoye et al., 2018; Niu and Bansal, 2018). Among them, many attribute-controlled text rewriting methods similarly employ GAN-based models to disentangle the content and style of text in a shared latent space (Shen et al., 2017; Fu et al., 2018). However, existing work that applies these ideas to text suffers from both training difficulty (Salimans et al., 2016; Arjovsky and Bottou, 2017; Bousmalis et al., 2017), and ineffective manipulation of the latent space which leads to content loss (Li et al., 2018) and generation of grammatically-incorrect sentences. Other lines of research avoid adversarial training altogether. Li

et al. (2018) proposed a much simpler approach: identify style-carrying n-grams, replace them with phrases of the opposite style, and train a neural language model to combine them in a natural way. Despite outperforming the adversarial approaches, its performance is dependent on the availability of an accurate word identifier, a precise word replacement selector and a perfect language model to fix the grammatical errors introduced by the crude swap.

Recent work improves upon adversarial approaches by additionally leveraging the idea of back translation (dos Santos et al., 2018; Logeswaran et al., 2018; Lample et al., 2019; Prabhumoye et al., 2018). It was previously used for unsupervised Statistical Machine Translation (SMT) (Fung and Yee, 1998; Munteanu et al., 2004; Smith et al., 2010) and Neural Machine Translation (NMT) (Conneau et al., 2017b; Lample et al., 2017; Artetxe et al., 2017), where it iteratively takes the pseudo pairs to train a translation model and then use the refined model to generate new pseudo-parallel pairs with enhanced quality. However, the success of this method relies on good quality of the pseudo-parallel pairs. Our approach proposes using retrieved sentences from the corpus based on semantic similarity as a decent starting point and then refining them using the trained translation models iteratively.

## 3 Method

### 3.1 Task Formulation

Given two mono-style corpora $X = \{x_1, \cdots, x_n\}$ with attribute $a_1$ and $Y = \{y_1, \cdots, y_m\}$ with attribute $a_2$. Note that the alignment between $X$ and $Y$ corpora is not available. The unsupervised attribute-controlled rewriting task aims to learn a transformation $T^*(\cdot)$ from $X$ to $Y$ by optimizing the following objective:

$$T^*(\cdot) = \underset{T(\cdot)}{\operatorname{argmin}} \ \|x - T(x)\|, \quad (1)$$

$$s.t. \ \mathcal{A}(T(x)) = a_2, \quad (2)$$

where the norm $\|\cdot\|$ measures the content shift between two sentences, and $\mathcal{A}(\cdot)$ represents the attribute of a sentence. Plainly put, a good attribute rewrite should ensure the attribute is changed to the desired value, and the content shift between the original sentence and rewrite is minimized.

### 3.2 Iterative Matching and Translation

We propose an iterative matching and translation algorithm composed of the following three steps:

**(Step 1) Matching** In iteration $t = 0$, we construct a large pseudo-parallel corpus $\hat{X}$ and $\hat{Y}^{(0)}$ by pairing sentences from $X$ with those from $Y$. Specifically, we calculate the semantic cosine similarity score (detailed in Section 4.4) between a sentence $x_i \in X$ and every sentence $y \in Y$, select the one with the highest score as $\hat{y}_i$, and only keep the sentence pair if the similarity exceeds a threshold $\gamma$. $\hat{X}$ denotes the subset of the original corpus $X$ for which we find matches.

In later iterations $t \geq 1$, this matching process is different from that in the first iteration. We match the pseudo-parallel corpus $\hat{X}$ with $\hat{Y}^{(t)}$, which is refined in step 3 of the previous iteration, and obtain a *temporary* matched corpus $\texttt{Match}^{(t)}$. In this case, for any sentence $x_i \in \hat{X}$, we have two pseudo-parallel sentences $\hat{y}_i \in \hat{Y}^{(t)}$ and $\texttt{match}_i \in \texttt{Match}^{(t)}$, both of which have the opposite attribute to $x_i$. We then use Word Mover Distance (WMD), which will be detailed in Section 3.3, to measure the content shift between the original sentence and the rewritten one. We calculate $\text{WMD}(x_i, \hat{y}_i)$ and $\text{WMD}(x_i, \texttt{match}_i)$. If the latter one is smaller, we then replace $\hat{y}_i$ with $\texttt{match}_i$ in $\hat{Y}^{(t)}$; otherwise, we just keep the original $\hat{y}_i$. In this way, we can obtain an updated version of $\hat{Y}^{(t)}$ after this matching step.

**(Step 2) Translation** In each iteration $t \geq 0$, a Seq2Seq machine translation model with attention $M^{(t)}$ (Luong et al., 2015) is trained from scratch using the pseudo-parallel corpus $\hat{X}$ and $\hat{Y}^{(t)}$.

**(Step 3) Refinement** In this step, we refine the $\hat{Y}^{(t)}$ obtained in step 1 with the trained translation model $M^{(t)}$ in step 2. Specifically, we apply the model $M^{(t)}$ to translate each sentence $x_i \in \hat{X}$ to $\texttt{trans}_i^{(t)}$, and form a *temporary* corpus $\texttt{Trans}^{(t)}$. Again, for any sentence $x_i \in \hat{X}$, we now have two pseudo-parallel sentences $\hat{y}_i \in \hat{Y}^{(t)}$ and $\texttt{trans}_i \in \texttt{Trans}^{(t)}$, one of which is from the matching step and the other is from the translation model. We still compare $\text{WMD}(x_i, \hat{y}_i)$ with $\text{WMD}(x_i, \texttt{trans}_i)$, and choose the sentence with the smaller value and insert into $\hat{Y}^{(t+1)}$, which will be fed into step 1 of the next iteration.

Overall, the aforementioned three steps are repeated for several iterations. This process is for-

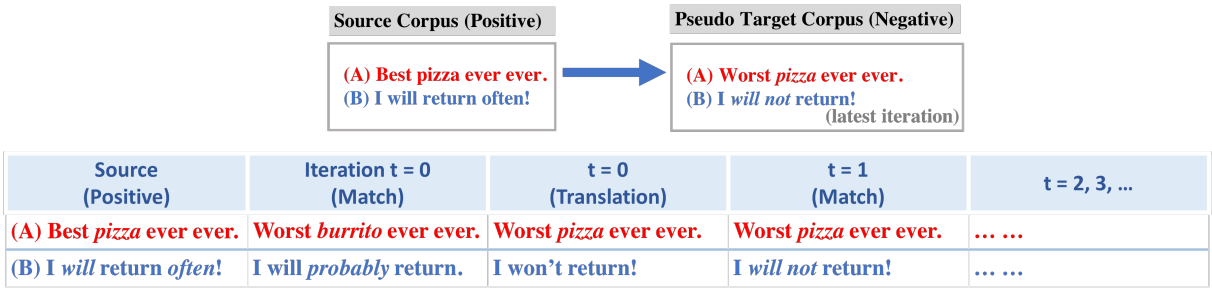


Figure 1: Iterative process of the algorithm to transfer the text style from positive to negative reviews.

malized below in Algorithm 1.

**Algorithm 1** Iterative Matching and Translation

**Input:** Two corpora $X, Y$ with different attributes.
$\hat{X} = \{\}$
**for** $t = 0, \cdots, T$ **do**
  $\hat{Y}^{(t)} = \{\}$
  **for** $x_i \in X$ **do**
    **if** $t == 0$ **then**
      **if** $\max_{y \in Y} \text{Sim}(x_i, y) > \gamma$ **then**
        $\hat{y}_i = \underset{y \in Y}{\text{argmax}}\, \text{Sim}(x_i, y)$
        $\hat{X}$.append($x_i$)
        $\hat{Y}^{(0)}$.append($\hat{y}_i$)
      **end if**
    **else**
      $\texttt{match}_i \leftarrow \underset{y \in Y}{\text{argmax}}\, \text{Sim}(\hat{y}_i, y)$
      $\hat{y}_i \leftarrow \underset{y \in \{\hat{y}_i, \texttt{match}_i\}}{\text{argmin}}\, \text{WMD}(x_i, y)$
      $\hat{Y}^{(t)}$.append($\hat{y}_i$)
    **end if**
  **end for**

  Train a Seq2Seq model $M^{(t)} : \hat{X} \rightarrow \hat{Y}^{(t)}$
  **for** $x_i \in \hat{X}$ **do**
    $\texttt{trans}_i \leftarrow M^{(t)}(x_i)$
    $\hat{y}_i \leftarrow \underset{y \in \{\hat{y}_i, \texttt{trans}_i\}}{\text{argmin}}\, \text{WMD}(x_i, y)$
    $\hat{Y}_i^{(t)} \leftarrow \hat{y}_i$
  **end for**
**end for**
**Output:** Attribute transfer model $M^{(T)} : X \rightarrow \hat{Y}^{(T)}$

### 3.3 Method Details

**Word Mover Distance** In our algorithm, WMD is used to measure the content shift from the source sentence to the rewrite. WMD is a metric of "travel cost" from sentence $s_a$ to $s_b$. The detailed explanations and calculation of the distance is in the paper (Kusner et al., 2015). In brief, each sentence is represented as a weighted point cloud of embedded words. The distance between the sentence $s_a$ to $s_b$ is the minimum cumulative distance that words from sentence $s_a$ need to travel to match exactly the point cloud of sentence $s_b$. Denote the vocabulary size as $n$, the travel distance from the word $i$ in sentence $s_a$ to the word $j$ in sentence $s_b$ as $T(i, j)$, and the corresponding cost of this "word travel" as $c(i, j)$. The distance calculation is formulated as

$$\text{WMD}(s_a, s_b) = \min_{\mathbf{T} \geq 0} \sum_{i,j=1}^{n} \mathbf{T_{ij}} \cdot c(i, j). \quad (3)$$

Since the initial construction of pseudo-parallel corpus is already guaranteed to have good target attribute and grammaticality, the only remaining criteria to fulfill is the minimal content change from the original sentence to the resulted output. WMD is used as a decision factor whenever an update occurs. Keeping the sentence pair with the smallest cost from each other approximates minimization of the content shift.

Advantages of the WMD over other basic measures of sentence similarity include the fact that it has no hyperparameters to tune, appropriately handles sentences with unequal number of words (via weighted word matchings that sum to 1), accounts for synonymic-similarity at the word-level (through use of pretrained word embedding vectors), and considers the entire contents of each sentence (every word must be somehow matched). Furthermore, WMD has produced high accuracy in information retrieval (Brokos et al., 2016; Kim et al., 2017), where measuring content-similarity is critical (as in our attribute transfer task).

**Semantic Sentence Representation** For the matching process in Step 1, the cosine similarity is computed between each pair of sentences. There is no perfect way of semantic representation of a sentence, but a state-of-the-art method is to use the sentence embeddings obtained by averaging the ELMo embeddings of all the words in the sentence (Peters et al., 2018). ELMo uses a deep, bi-directional LSTM model to create word representations within the context that they are

used. Perone et al. (2018) have shown that this approach can efficiently represent semantic and linguistic features of a sentence, outperforming more elaborate approaches such as Skip-Thought (Kiros et al., 2015), InferSent (Conneau et al., 2017a) and Universal Sentence Encoder (Cer et al., 2018).

## 4 Experiments

### 4.1 Datasets

We evaluate the proposed model on two representative tasks: sentiment modification on the YELP dataset, and text formality conversion on the FORMALITY dataset. A careful human assessment in Section 4.1.1 shows that these two datasets are significantly more suitable than the other three popular "style transfer" datasets, namely the political slant transfer dataset (Prabhumoye et al., 2018; Tian et al., 2018), gender transfer dataset (Prabhumoye et al., 2018), and humorous-to-romantic transfer dataset (Li et al., 2018).

| Dataset | Style | Train | Dev | Test |
|---|---|---|---|---|
| YELP | Positive | 270K | 2K | 0.5K |
| | Negative | 180K | 2K | 0.5K |
| FORMALITY | Formal | 90K | 4.6K | 2.7K |
| | Informal | 96K | 5.6K | 2.1K |

Table 2: Statistics of YELP and FORMALITY datasets.

**YELP** The commonly used YELP review dataset for sentiment modification (Shen et al., 2017; Li et al., 2018; Prabhumoye et al., 2018) contains a positive corpus of reviews rated above three and a negative corpus of reviews rated below three. This task requires flipping high polarity sentences such as "The food is good" and "The food is bad". We use the same train/test split as Shen et al. (2017); Li et al. (2018) (see Table 2).

**FORMALITY** The FORMALITY dataset stems from an aligned set of formal and informal sentences (Rao and Tetreault, 2018). It demands changes in subtle linguistic features such as "want to" to "wanna". We obtained a non-parallel dataset by shuffling the two originally aligned corpora (removing duplicates and sentences that exceed 100 words). Table 2 describes statistics of the resulting two corpora. The development/test sets are provided with four human-generated attribute transfer rewrites for each sentence (i.e. the gold standard).

#### 4.1.1 Dataset Quality Assessment

In order to identify the best datasets for evaluation, we asked human judges to assess five popular text attribute rewriting datasets, namely YELP sentiment modification dataset, FORMALITY dataset, POLITICAL slant transfer dataset (Prabhumoye et al., 2018; Tian et al., 2018), GENDER transfer dataset (Prabhumoye et al., 2018), and HUMOROUS-to-ROMANTIC transfer dataset (Li et al., 2018). For each dataset, we randomly extracted 100 sentences (i.e. 50 per style). For each sentence, we asked two native English speakers to annotate them with one of three options, i.e. either of the two attributes or "Cannot Decide". Based on the collected annotations, we calculate three metrics: (1) *undecidable rate*, which is the percentage of "Cannot Decide" answers (we report the average percentage between the two annotators), (2) *disagreement rate*, which is the percentage of different opinions between the two annotators, and (3) *F1 score* between the human annotations and the gold labels in the original dataset ("Cannot Decide" answers were not considered).

| Dataset | Unde. Rate | Dis. Rate | F1 |
|---|---|---|---|
| POLITICAL | 49.0 | 33.0 | 67.2 |
| GENDER | 88.0 | 14.0 | 53.6 |
| HUMOROUS | 66.0 | 31.0 | 79.3 |
| YELP | **11.0** | 16.0 | **93.2** |
| FORMALITY | **0.5** | 17.0 | **87.6** |

Table 3: Comparison of dataset quality. "Unde. Rate" indicates the undecidable rate, "Dis. Rate" the disagreement rate between annotators.

The scores for each dataset are summarized in Table 3. A quick comparison shows that YELP and FORMALITY obtain significantly lower *undecidable* and inter-annotator *disagreement rates*, indicating that the style of the sentences in these two datasets are less ambiguous to humans. In addition, YELP and FORMALITY have much higher F1 score than all the other datasets, which confirms the correctness of the source-target attribute split.

This comparison points out that the three datasets, including political slant, gender and romantic-to-humorous caption datasets, are ambiguous and noisy, therefore not only adding complexity to the task and its evaluation (because even human annotators struggle to identify the correct style), but also leading to models that are not useful in real practice.

| | YELP | | | | FORMALITY | | | |
|---|---|---|---|---|---|---|---|---|
| | Content | Grammar | Attribute | *Success (%)* | Content | Grammar | Attribute | *Success (%)* |
| CA | 2.30 $\pm$1.45 | 2.65 $\pm$1.32 | 2.77 $\pm$1.69 | 5.5 | 2.45 $\pm$1.31 | 2.67 $\pm$1.57 | 1.97 $\pm$1.26 | 0.5 |
| MD | 2.12 $\pm$1.52 | 2.38 $\pm$1.49 | 2.25 $\pm$1.54 | 2.0 | 1.79 $\pm$1.17 | 2.37 $\pm$1.50 | 2.42 $\pm$1.56 | 0.0 |
| DAR | 2.93 $\pm$1.45 | 2.95 $\pm$1.51 | 3.09 $\pm$1.65 | 14.5 | 2.89 $\pm$1.39 | 3.02 $\pm$1.59 | 2.95 $\pm$1.56 | 7.0 |
| Ours | **3.07** $\pm$1.49 | **4.32** $\pm$1.07 | **3.43** $\pm$1.65 | **21.0** | **3.60** $\pm$1.55 | **4.42** $\pm$1.08 | **3.11** $\pm$1.58 | **29.0** |
| Human | 4.66 $\pm$0.77 | 4.33 $\pm$1.07 | 4.22 $\pm$1.29 | 63.5 | 4.62 $\pm$0.82 | 4.62 $\pm$0.94 | 3.91 $\pm$1.57 | 59.5 |

Table 4: The mean and standard deviation of human ratings on Content preservation, Grammaticality, and attribute correctness on both datasets for different systems: **C**ROSS**A**LIGNMENT (CA), **M**ULTI**D**ECODER (MD), **D**ELETE**A**ND**R**ETRIEVE (DAR), **Ours**, and **Human** reference. Ratings are on a 1 to 5 scale.

### 4.2 Evaluation Strategies

**Human Evaluation** Following Li et al. (2018); Lample et al. (2019), we asked human judges to evaluate outputs of different models in terms of content preservation, grammaticality and attribute change correctness on a Likert scale from 1 to 5. We randomly selected 100 sentences from each corpus (100 positive and 100 negative sentences from YELP; 100 formal and 100 informal sentences from FORMALITY). Each of the twelve human judges passed a test batch prequalification before they could start evaluating, and we verified they spent a reasonable amount of time in the task.

**Automatic Evaluation** In addition to the human evaluation, we also programatically gauge rewriting quality via automated methods, following the practice in (Li et al., 2018; Lample et al., 2019). Here, content preservation is measured by the BLEU score between model outputs and multiple human references for each sentence. Since the previous work only introduces single human reference for YELP, we enriched the test set with four extra human references for each test sentence, in total 800 rewrites. The diverse human rewrites ensures a more variety-tolerant measurement of model outputs. Attribute accuracy is assessed via the classification results of a CNN-based text classifier (Kim, 2014). To automatically gauge fluency, we use pretrained language models (LM) to get the perplexity of output sentences. The details of the classifier, LM and the collection of human references are elaborated in Appendix A.1.

### 4.3 Baselines

To compare against multiple baselines, we re-implemented three recently-proposed methods (described in Sections 1 and 2): CROSSALIGNMENT (CA) from Shen et al. (2017), MULTIDECODER (MD) from Fu et al. (2018), and DELETEANDRETRIEVE (DAR) from Li et al. (2018).

### 4.4 Experimental settings

For the translation process, we used an off-the-shelf Seq2Seq model, a 2-layer LSTM encoder-decoder with 1024 hidden dimensions and attention. We focused on the novelty of the proposed iterative framework, so a standard and classic Seq2Seq model is used. More experimental details are described in Appendix A.2.

## 5 Results

### 5.1 Human Evaluation

In terms of human evaluation, the proposed approach shows significant gains over all baselines in terms of attribute correctness, content preservation, grammaticality, and success rate as shown in Table 4 ($p < 0.05$ using bootstrap resampling (Koehn, 2004)).

The largest improvement is in grammaticality, where we achieve an average of 4.32 on YELP and 4.42 out of 5.0 on the FORMALITY dataset. These scores are close to those of human references, prevalently outperforming the baselines. On attribute correctness, the model scores 3.43 on YELP and 3.11 on FORMALITY dataset, exceeding the previous best methods by 0.33 and 0.16 respectively. Moreover, in content preservation, we show an improvement of 0.14 and 0.72 over the previous state-of-the-art *DeleteAndRetrieve* model on the two datasets. Finally, we follow the practice of Li et al. (2018) to evaluate the *Success Rate* — an aggregate of the three previous metrics, in which a sample is successful only if it is rated 4 or 5 on all the three metrics. Our model demonstrates an improvement of 6.5% in success rate over the previous best method.

To further inspect performance of different methods, we show some typical outputs in Table 5. First, the output of our model is nearly as grammatical as the human-written sentence, compared with the loss of sentence struc-

| | |
|---|---|
| **From Positive to Negative (YELP)** | |
| **Input** | Thank you ladies for being **awesome**! |
| CA | Thank you charge up for this food business. |
| MD | Thank you for it $ **poor** company. |
| DAR | Thank you ladies for being **didn't**. |
| **Ours** | **Thank you for wasting my time you idiots.** |
| Human | Thank you ladies for being the **worst**! |
| **From Informal to Formal (FORMALITY)** | |
| **Input** | **i** tried to like him, bu **i just can't**. |
| CA | I was to him, but I **just like I can't**. |
| MD | **i** tried to him like, but I **just should too**. |
| DAR | I do not know if you like him but I do not know he **can't**. |
| **Ours** | **I tried to like him, but I cannot.** |
| Human | I **attempted** to like him, **however**, I am **unable**. |

Table 5: Example outputs of different systems.

| | YELP | | | FORMALITY | | |
|---|---|---|---|---|---|---|
| | Acc. | BLEU | PPL | Acc. | BLEU | PPL |
| CA | 73.40 | 17.54 | 46.92 | 46.41 | 12.65 | 53.89 |
| MD | 66.80 | 17.34 | 69.51 | 66.46 | 7.46 | 109.41 |
| DAR | 88.00 | **31.07** | 79.70 | **78.87** | 21.29 | 63.61 |
| Ours | **95.90** | 22.46 | **14.89** | 72.07 | **38.16** | **32.63** |

Table 6: Automatic evaluation results on both datasets. Acc. is the style/attribute classification accuracy (%).

ture in CROSSALIGNMENT (e.g. "I tried to him like") and DELETEANDRETRIEVE (e.g. "for being didn't"). Second, in terms of attribute correctness, DELETEANDRETRIEVE suffers from failure to convert the attribute-bearing words (as it wrongly converts the word "awesome" to "didn't"), and CROSSALIGNMENT is prone to missing attribute words. Third, although our method enforces content alignment, the preservation of content is an inevitable challenge for the previous approaches such as MULTIDECODER and DELETEANDRETRIEVE.

### 5.2 Automatic Evaluation

We also assess each method's performance via the automatic measures of attribute-accuracy, BLEU, and perplexity (Table 6). A highlight is its perplexity score, outperforming the previous methods by a large margin. This advantage owes to the fact that the Seq2Seq model is trained on pseudo-pairs similar to real samples, which can guarantee the translation quality. However, a tradeoff between the other two aspects, attribute accuracy and BLEU, can be clearly seen on both YELP and FORMALITY datasets. This is common when tuning all models, as targeting at a higher BLEU score will result in a lower attribute correctness score (Shen et al., 2017; Fu et al., 2018; Li et al., 2018). Similarly, in iterations of our model, the BLEU score gradually increases while the accuracy decreases. Therefore, the reported outputs are balanced based on all three aspects. An analysis on the limitations of automatic evaluation is in Appendix A.3.

## 6 Performance Analysis

Here, we analyze the performance of our model with regard to various aspects in order to understand what factors underlie its success.

### 6.1 Initialization of Pseudo-Parallel Corpus

The proposed model starts from the construction of an initial pseudo-parallel corpus by our matching step. Note that this initial pairing is practicable in most domains. For example, Yelp reviews naturally have positive and negative comments on the same food; different translations of the same book also shares the same content; different news agencies report with different tones on the same events. After construction, there are three properties of this pseudo-parallel corpus: First, it is a subset of the original corpus. Second, all retrieved target sentences contain the desired attribute information and are of perfect grammaticality. This property is retained throughout our iterations and is key to the high attribute transfer accuracy and fluent language generation capabilities of our model. Third, the sentence pairs are still imperfect in terms of content preservation, often similar in meaning but with certain content words altered. This is remedied by subsequent refinement steps.

Although our model needs the matched pseudo-parallel corpus as a starting point, it has high tolerance to recover from occasional low-quality matches. To demonstrate this, we randomly

picked 100 sentences and their initial pseudo-pairs from both source and target corpus, and asked human judges to rate them. For each sentence pair, three judges decided whether the sentence pair forms a good rewrite, a bad rewrite, or an ambiguous one. We mark the pseudo-pair as either good or bad if at least two annotators agree on such a judgment. The percentage of bad rewrites is $38.2\%$ and $48.2\%$ on YELP and FORMALITY, respectively. This indicates that subsequent iterative refinement indeed allows for $30-50\%$ low-quality pairs in the initial pseudo-parallel corpus.

### 6.2 Effective Denoising Translation

One of the most important gains from the iterative translation is to encourage more content preservation. We illustrate the effectiveness of translation by using automatic evaluation to gauge how the bad matches in the initial pseudo-parallel corpus change *immediately after* the first translation step. We find that after the first translation, the BLEU score of these bad matches shows a clear improvement, increasing from 9.40 to 13.18 on YELP, and 5.44 to 28.25 on FORMALITY. This shows that the translation model recognizes the noise in the first matching process and generates more proper transfer candidates, providing a good foundation for subsequent iterative refinement. An illustrative example of this refinement can be seen in the sentence (A) in Figure 1, where a bad match of "Worst burrito ever ever" was denoised and replaced with "Worst pizza ever ever" after the first translation.

### 6.3 Iterative Refinement

The essential increase of content preservation owes to the iterative refinement step. This process reduces erroneous alignments in the pseudo-parallel corpus by updating each existing pseudo-pair with newly generated translation-pair. Thanks to the denoising effect of the translation model, this updating could improve the quality of pseudo-pairs. Furthermore, we use WMD to measure the content shift between sentences in the pseudo-pair and the translation-pair, and accept the update only when the translation-pair possesses smaller content shift so as to avoid worsening updates. As the iteration goes on, the accuracy and perplexity stay high from the beginning, but as in Figure 2, the BLEU score keeps increasing. This shows that the new iteration outputs retains more content from the original sentences. More importantly, this refinement can prevent the model from totally relying on the fine-quality matching pairs, which contributes to the high tolerance capability on the matching quality as discussed in Section 6.1.

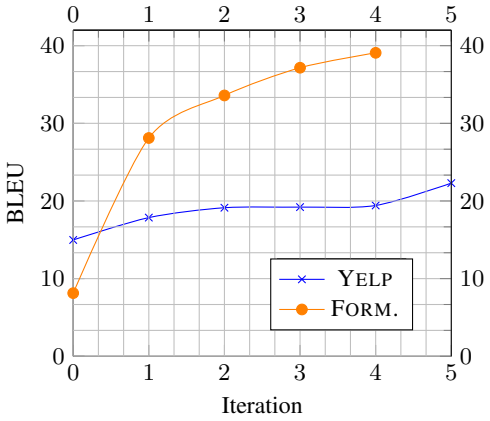


Figure 2: BLEU score evolution across training iterations on YELP and FORMALITY datasets.

The pseudo-parallel corpus in Iteration $t \geq 0$ is composed of matched pairs from the original corpora and translated pairs from the trained translation model. Both of them contribute to the good performance of our model. For further investigation, we look into the pseudo-parallel corpus $\hat{X}$ and $\hat{Y}^{(4)}$ in the final training iteration for the FORMALITY dataset. Since this dataset has ground-truth parallel pairs, we are able to calculate how many pseudo-parallel pairs are the same as the ground-truth parallel pairs, and the percentage turns out to be around 52%, which is actually not high. This reveals that the refinement step can provide the model with better pseudo-pairs even than original gold pairs and our model can still stand out without fine-quality matches.

## 7 Conclusion

In this work, we proposed a Seq2Seq paradigm for text attribute transfer applications, suggesting a simple but strong method for overcoming lack of parallel data. We construct a pseudo-parallel corpus by iteratively matching and updating in a way that increasingly refines the final transfer function. Our framework can employ any Seq2Seq model and outperforms previous methods under all measured criteria (content preservation, fluency, and attribute correctness) in both human and automatic evaluation. The simplicity and flexibility of our approach can be useful in applications that require intricate edits or complete sentence rewrites.

## Acknowledgments

We thank Professor Peter Szolovits for supporting and funding the work, and Prof Benjamin Kao for advice on evaluation metrics. We also appreciate Tianxiao Shen, Jiang Guo, Professor Regina Barzilay, and MIT Natural Language Processing Group for insightful discussions and support. We thank Adam Fisch, Benson Chen, Jingran Zhou, Wui Wang Lui, Shuailin Yuan and many other helpers for human evaluation.

## References

Martin Arjovsky and Léon Bottou. 2017. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2017. Unsupervised neural machine translation. *arXiv preprint arXiv:1710.11041*.

Miguel Ballesteros, Bernd Bohnet, Simon Mille, and Leo Wanner. 2015. Data-driven sentence generation with non-isomorphic trees. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. 2017. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Georgios-Ioannis Brokos, Prodromos Malakasiotis, and Ion Androutsopoulos. 2016. Using centroids of word embeddings and word mover's distance for biomedical document retrieval in question answering. *arXiv preprint arXiv:1608.03905*.

Julian Brooke, Tong Wang, and Graeme Hirst. 2010. Automatic acquisition of lexical formality. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 90–98. Association for Computational Linguistics.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017a. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017b. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Giuseppe Di Fabbrizio, Amanda J Stent, and Srinivas Bangalore. 2008. Trainable speaker-based referring expression generation. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 151–158.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2018. Style transfer in text: Exploration and evaluation. In *AAAI*.

Pascale Fung and Lo Yuen Yee. 1998. An ir approach for translating new words from nonparallel, comparable texts. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pages 414–420. Association for Computational Linguistics.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61:65–170.

Albert Gatt, Francois Portet, Ehud Reiter, Jim Hunter, Saad Mahamood, Wendy Moncur, and Somayajulu Sripada. 2009. From data to text in the neonatal intensive care unit: Using nlg technology for decision support and information management. *Ai Communications*, 22(3):153–186.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *International Conference on Machine Learning*, pages 1587–1596.

Harsh Jhamtani, Varun Gangal, Eduard H. Hovy, and Eric Nyberg. 2017. Shakespearizing modern language using copy-enriched sequence-to-sequence models. *CoRR*, abs/1707.01161.

Patrick Juola and Darren Vescovi. 2011. Analyzing stylometric approaches to author obfuscation. In *IFIP Int. Conf. Digital Forensics*.

Jad Kabbara and Jackie Chi Kit Cheung. 2016. Stylistic transfer in natural language generation systems using recurrent neural networks. pages 43–47.

Gary Kacmarcik and Michael Gamon. 2006. Obfuscating document stylometry to preserve author anonymity. In *ACL*.

Sun Kim, Nicolas Fiorini, W John Wilbur, and Zhiyong Lu. 2017. Bridging the gap: Incorporating a semantic similarity measure for effectively mapping pubmed queries to documents. *Journal of Biomedical Informatics*, 75:122–127.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in Neural Information Processing Systems*, pages 3294–3302.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL, System Demonstrations*, pages 67–72.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *EMNLP*.

Emiel Krahmer and Kees van Deemter. 2012. Computational generation of referring expressions: A survey. *Computational Linguistics*, 38(1):173–218.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In *International Conference on Machine Learning*, pages 957–966.

Guillaume Lample, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2017. Unsupervised machine translation using monolingual corpora only. *arXiv preprint arXiv:1711.00043*.

Guillaume Lample, Sandeep Subramanian, Eric Smith, Ludovic Denoyer, Marc'Aurelio Ranzato, and Y-Lan Boureau. 2019. Multiple-attribute text rewriting. In *International Conference on Learning Representations*.

Juncen Li, Robin Jia, He He, and Percy S. Liang. 2018. Delete, retrieve, generate: A simple approach to sentiment and style transfer. In *NAACL-HLT*.

Lajanugen Logeswaran, Honglak Lee, and Samy Bengio. 2018. Content preserving text generation with attribute controls. In *Advances in Neural Information Processing Systems*.

Thang Luong, Hieu Quang Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*.

François Mairesse and Marilyn A Walker. 2011. Controlling user perceptions of linguistic style: Trainable generation of personality traits. *Computational Linguistics*, 37(3):455–488.

Igor Melnyk, Cicero Nogueira dos Santos, Kahini Wadhawan, Inkit Padhi, and Abhishek Kumar. 2017. Improved neural text attribute transfer with non-parallel data. *arXiv preprint arXiv:1711.09395*.

Jonas Mueller, David Gifford, and Tommi Jaakkola. 2017. Sequence to better sequence: continuous revision of combinatorial structures. In *International Conference on Machine Learning*, pages 2536–2544.

Dragos Stefan Munteanu, Alexander Fraser, and Daniel Marcu. 2004. Improved machine translation performance via parallel sentence extraction from comparable corpora. In *HLT-NAACL*.

Tong Niu and Mohit Bansal. 2018. Polite dialogue generation without parallel data. *Transactions of the Association of Computational Linguistics*, 6:373–389.

Shereen Oraby, Lena Reed, Shubhangi Tandon, S. SharathT., Stephanie M. Lukin, and Marilyn A. Walker. 2018. Controlling personality-based stylistic variation with neural natural language generators. In *SIGDIAL Conference*.

Ellie Pavlick and Joel R. Tetreault. 2016. An empirical analysis of formality in online communication. *Transactions of the Association for Computational Linguistics*, 4:61–74.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Christian S Perone, Roberto Silveira, and Thomas S Paula. 2018. Evaluation of sentence embeddings in downstream and linguistic probing tasks. *arXiv preprint arXiv:1806.06259*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke S. Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*.

Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan R. Salakhutdinov, and Alan W. Black. 2018. Style transfer through back-translation. In *ACL*.

Sudha Rao and Joel Tetreault. 2018. Dear sir or madam, may i introduce the yafc corpus: Corpus, benchmarks and metrics for formality style transfer. *arXiv preprint arXiv:1803.06535*.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242.

Cícero Nogueira dos Santos, Igor Melnyk, and Inkit Padhi. 2018. Fighting offensive language on social media with unsupervised text style transfer. In *ACL*.

Fadi Abu Sheikha and Diana Inkpen. 2011. Generation of formal and informal sentences. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 187–193. Association for Computational Linguistics.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in Neural Information Processing Systems*, pages 6833–6844.

Jason R Smith, Chris Quirk, and Kristina Toutanova. 2010. Extracting parallel sentences from comparable corpora using document level alignment. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 403–411. Association for Computational Linguistics.

Youzhi Tian, Zhiting Hu, and Zhou Yu. 2018. Structured content preservation for unsupervised text style transfer. *arXiv preprint arXiv:1810.06526*.

Zhirui Zhang, Shuo Ren, Shujie Liu, Jianyong Wang, Peng Chen, Mu Li, Ming Zhou, and Enhong Chen. 2018. Style transfer as unsupervised machine translation. *arXiv preprint arXiv:1808.07894*.

## A Supplemental Material

In this Appendix, we explain the setup of automatic evaluation, experimental details, and a discussion on limitations of automatic evaluation.

### A.1 Automatic Evaluation Setup

The details of the three aspects of automatic evaluation are elaborated as follows.

- **Style correctness:** Following Shen et al. (2017), we trained a CNN-based text classifier (Kim, 2014) on our original datasets, using its accuracy over the system outputs to measure their style correctness. The accuracy of this classifier is respectively 97% and 93% on the YELP and the FORMALITY datasets.

- **Content preservation:** To evaluate the content preservation, we compute the BLEU score between model outputs and multiple human references. FORMALITY dataset comes with four human references for around 2,000 formal and 2,000 informal sentences.

  For the YELP dataset, we used the same test set of 500 positive and 500 negative sentences as (Li et al., 2018), and collected four references for each sentence in the test set, in addition to the single human reference released by Li et al. (2018). We hired crowdworkers on Amazon Mechanical Turk to rewrite the source sentence with the same content but an opposite sentiment.

  These in total five human rewrites of each test sentence ensures a more tolerant measurement of model outputs. We can see a relatively large diversity in human transfers, which is measured by the average BLEU score of one randomly chosen human reference among the other four. The average difference gap between the calculated score, 52.63, and a perfect BLEU score, 100, shows that the five human rewritten sentences contain significant lexical differences. Evaluation based on all five human rewrites thus offers a better measure of transfer quality. For example, the two equally acceptable rewrites "i will definitely not return often!" and "I won't be returning any time soon." does not have a single word overlap and have a BLEU score 0 with each other. Therefore, five human references enable a more comprehensive evaluation, allowing multiple ways to transfer a sentence.

- **Fluency:** Fluency is measured by the perplexity (PPL) of the generated outputs by a pre-trained language model (LM) using GluonNLP toolkit[2]. The encoder of this LM is comprised of two long-short term memory (LSTM) layers, each of which has 200 hidden units. The embedding and output weights are tied. Dropout of 0.2 was applied to both embedding and LSTM layers. LM was optimized via stochastic gradient descent (SGD) optimizer with learning rate of 20 for 15 epochs.

### A.2 Experimental Details

We adopt the 100-dimensional pretrained GloVe word embeddings (Pennington et al., 2014) as inputs of a standard machine translation sequence-to-sequence model with attention. The NLTK software package is used to generate Part-of-Speech tags and we feed them as additional input to the encoder. To ensure both a relatively high quality pseudo-parallel corpus and no significant drop in size on the two corpora, we only match sentences with vector cosine higher than an empirical similarity threshold of 0.7. We train the model until the update rate of candidate transfer is lower than 0.5%. This convergence is at iteration $T = 5$ for YELP and $T = 4$ for FORMALITY.

### A.3 Limitations of Automatic Evaluation

Evaluating the quality of a transferred sentence by a pretrained classifier (style accuracy), lexicon overlap with references (BLEU), and a pretrained Language Model (perplexity) can have many limitations. First, in terms of style accuracy and perplexity, the pretrained models can be unreliable when evaluating a sentence different from the training corpus. For example, comparing human-rated and machine-evaluated score of *style correctness* (Table 4 and Table 6), we find that although the automatic score can serve as a rough assessment of the models, it does not aligns perfectly with the human ratings in Table 4. As is explained in (Li et al., 2018)'s work, the uneven distribution of some content words in two style corpora may confuse the classifier and make it overfitted on the training data.

[2] https://gluon-nlp.mxnet.io/index.html

Second, the BLEU score, which mainly relies on the lexicon overlap between evaluated sentence and references, can mistakenly favor sentences with a higher similarity to the source sentence. An illustration is that by simply copying all source sentences in the test set we can get a BLEU score of 62, despite an accuracy score close to zero. The state-of-the-art model, *DeleteAndRetrieve*, only modifies a few attribute-carrying words and copies the rest of the sentence. For example, it transfers the source sentence "My 'hot' sub *was cold* and the meat was watery." into an ungrammatical one "My 'hot' *is a great place* to the meat." but keeps the words "hot" and "meat". Consequently, it results in a high BLEU score and a poor perplexity score.

# Better Rewards Yield Better Summaries: Learning to Summarise Without References

**Florian Böhm**[1], **Yang Gao**[1*], **Christian M. Meyer**[1],
**Ori Shapira**[2], **Ido Dagan**[2], **and Iryna Gurevych**[1]
[1]Ubiquitous Knowledge Processing Lab, Technische Universität Darmstadt, Germany
[2]Computer Science Department, Bar-Ilan University, Ramat-Gan, Israel
https://www.ukp.tu-darmstadt.de/
yang.gao@rhul.ac.uk, obspp18@gmail.com, dagan@cs.biu.ac.il

## Abstract

*Reinforcement Learning (RL)* based document summarisation systems yield state-of-the-art performance in terms of ROUGE scores, because they directly use ROUGE as the *rewards* during training. However, summaries with high ROUGE scores often receive low human judgement. To find a better reward function that can guide RL to generate human-appealing summaries, we learn a reward function from human ratings on 2,500 summaries. Our reward function only takes the document and system summary as input. Hence, once trained, it can be used to train RL-based summarisation systems without using any reference summaries. We show that our learned rewards have significantly higher correlation with human ratings than previous approaches. Human evaluation experiments show that, compared to the state-of-the-art supervised-learning systems and ROUGE-as-rewards RL summarisation systems, the RL systems using our learned rewards during training generate summaries with higher human ratings. The learned reward function and our source code are available at https://github.com/yg211/summary-reward-no-reference.

## 1 Introduction

*Document summarisation* aims at generating a summary for a long document or multiple documents on the same topic. *Reinforcement Learning (RL)* becomes an increasingly popular technique to build document summarisation systems in recent years (Chen and Bansal, 2018; Narayan et al., 2018b; Dong et al., 2018). Compared to the supervised learning paradigm, which "pushes" the summariser to reproduce the reference summaries at training time, RL directly optimises the summariser to maximise the *rewards*, which measure the quality of the generated summaries.

The *ROUGE* metrics (Lin, 2004b) are the most widely used rewards in training RL-based summarisation systems. ROUGE measures the quality of a generated summary by counting how many n-grams in the reference summaries appear in the generated summary. ROUGE correlates well with human judgements at *system level* (Lin, 2004a; Louis and Nenkova, 2013), i.e. by aggregating system summaries' ROUGE scores across multiple input documents, we can reliably rank summarisation systems by their quality. However, ROUGE performs poorly at *summary level*: given multiple summaries for the same input document, ROUGE can hardly distinguish the "good" summaries from the "mediocre" and "bad" ones (Novikova et al., 2017). Because existing RL-based summarisation systems rely on summary-level ROUGE scores to guide the optimisation direction, the poor performance of ROUGE at summary level severely misleads the RL agents. The reliability of ROUGE as RL reward is further challenged by the fact that most large-scale summarisation datasets only have one reference summary available for each input document (e.g. CNN/DailyMail (Hermann et al., 2015; See et al., 2017) and NewsRooms (Grusky et al., 2018)).

In order to find better rewards that can guide RL-based summarisers to generate more human-appealing summaries, we learn a reward function directly from human ratings. We use the dataset compiled by Chaganty et al. (2018), which includes human ratings on 2,500 summaries for 500 news articles from CNN/DailyMail. Unlike ROUGE that requires one or multiple reference summaries to compute the scores, our reward function only takes the document and the generated summary as input. Hence, once trained, our

* Since June 2019, Yang Gao is affiliated with Dept. of Computer Science, Royal Holloway, University of London.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3110–3120,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

reward can be used to train RL-based summarisation systems without any reference summaries.

The contributions of this work are threefold: **(i)** We study the summary-level correlation between ROUGE and human judgement on 2,500 summaries (§3), explicitly showing that ROUGE can hardly identify the human-appealing summaries. **(ii)** We formulate the reward learning problem as either a regression or a preference learning problem (§4), and explore multiple text encoders and neural architectures to build the reward learning model (§5). **(iii)** We show that our learned reward correlates significantly better with human judgements than ROUGE (§6), and that using the learned reward can guide both extractive and abstractive RL-based summarisers to generate summaries with significantly higher human ratings than the state-of-the-art systems (§7).

## 2 Related Work

**RL-based summarisation.** Most existing RL-based summarisers fall into two categories: *cross-input* systems and *input-specific* systems (Gao et al., 2019). Cross-input systems learn a summarisation policy at training time by letting the RL agent interact with a ROUGE-based reward function. At test time, the learned policy is used to generate a summary for each input document. Most RL-based summarisers fall into this category (Chen and Bansal, 2018; Narayan et al., 2018b; Dong et al., 2018; Kryscinski et al., 2018; Pasunuru and Bansal, 2018; Paulus et al., 2018). As an alternative, input-specific RL (Rioux et al., 2014; Ryang and Abekawa, 2012) does not require reference summaries: for each input document at test time, a summarisation policy is trained specifically for the input, by letting the RL summariser interact with a heuristic-based reward function, e.g. ROUGE between the generated summary and the input document (without using any reference summaries). However, the performance of input-specific RL falls far behind the cross-input counterparts.

In §7 we use our learned reward to train both cross-input and input-specific RL systems. A similar idea has been explored by Gao et al. (2019), but unlike their work that learns the reward from ROUGE scores, we learn our reward directly from human ratings. Human evaluation experiments suggest that our reward can guide both kinds of RL-based systems to generate human-appealing summaries without using reference summaries.

The reward learning idea is also related to *inverse RL* (IRL) (Ng and Russell, 2000). By observing some (near-)optimal sequences of actions, IRL algorithms learn a reward function that is consistent with the observed sequences. In the case of summarisation, human-written reference summaries are the (near-)optimal sequences, which are expensive to provide. Our method only needs human ratings on some generated summaries, also known as the *bandit feedback* (Kreutzer et al., 2017), to learn the reward function. Furthermore, when employing certain loss functions (see §4 and Eq. (2)), our method can even learn the reward function from preferences over generated summaries, an even cheaper feedback to elicit (Kreutzer et al., 2018; Gao et al., 2018).

**Heuristic-based rewards.** Prior work proposed heuristic-based reward functions to train cross-input RL summarisers, in order to strengthen certain properties of the generated summaries. Arumae and Liu (2019) propose four reward functions to train an RL-based extractive summariser, including the *question-answering competency* rewards, which encourage the RL agent to generate summaries that can answer cloze-style questions. Such questions are automatically created by removing some words in the reference summaries. Experiments suggest that human subjects can answer the questions with high accuracy by reading their generated summaries; but the human judgement scores of their summaries are not higher than the summaries generated by the state-of-the-art supervised system. Kryscinski et al. (2018) propose a simple heuristic that encourages the RL-based abstractive summariser to generate summaries with more *novel* tokens, i.e. tokens that do not appear in the input document. However, both ROUGE and human evaluation scores of their system are lower than the state-of-the-art summarisation systems (e.g. See et al., 2017). In addition, the above rewards require reference summaries, unlike our reward that only takes a document and a generated summary as input.

**Rewards learned with extra data.** Pasunuru and Bansal (2018) propose two novel rewards for training RL-based abstractive summarisers: *RougeSal*, which up-weights the salient phrases and words detected via a keyphrase classifier, and *Entail* reward, which gives high scores to

logically-entailed summaries using an entailment classifier. RougeSal is trained with the SQuAD reading comprehension dataset (Rajpurkar et al., 2016), and Entail is trained with the SNLI (Bowman et al., 2015) and Multi-NLI (Williams et al., 2018) datasets. Although their system achieves new state-of-the-art results in terms of ROUGE, it remains unclear whether their system generates more human-appealing summaries as they do not perform human evaluation experiments. Additionally, both rewards require reference summaries.

Louis and Nenkova (2013), Peyrard et al. (2017) and Peyrard and Gurevych (2018) build feature-rich regression models to learn a summary evaluation metric directly from the human judgement scores (Pyramid and Responsiveness) provided in the TAC'08 and '09 datasets[1]. Some features they use require reference summaries (e.g. ROUGE metrics); the others are heuristic-based and do not use reference summaries (e.g. the Jensen-Shannon divergence between the word distributions in the summary and the documents). Their experiments suggest that with only non-reference-summary-based features, the correlation of their learned metric with human judgements is lower than ROUGE; with reference-summary-based features, the learned metric marginally outperforms ROUGE. In §6, we show that our reward model does not use reference summaries but outperforms the feature-based baseline by Peyrard and Gurevych (2018) as well as ROUGE.

Unlike the above work that attempts to learn a summary evaluation metric, the target of our work is to learn a good *reward*, which is not necessarily a good *evaluation metric*. A good evaluation metric should be able to correctly rank summaries of different quality levels, while a good reward function focuses more on distinguishing the best summaries from the mediocre and bad summaries. Also, an evaluation metric should be able to evaluate summaries of different types (e.g. extractive and abstractive) and from different genres, while a reward function can be specifically designed for a single task. We leave the learning of a generic summarisation evaluation metric for future work.

## 3 Summary-Level Correlation Study

In this section, we study the summary-level correlation between multiple widely used summary evaluation metrics and human judgement scores, so as to further motivate the need for a stronger reward for RL. Metrics we consider include ROUGE (full length F-score), BLEU (Papineni et al., 2002) and METEOR (Lavie and Denkowski, 2009). Furthermore, in line with Chaganty et al. (2018), we also use the cosine similarity between the embeddings of the generated summary and the reference summary as metrics: we use InferSent (Conneau et al., 2017) and BERT-Large-Cased (Devlin et al., 2019) to generate the embeddings.

The human judgement scores we use are from Chaganty et al. (2018), collected as follows. First, 500 news articles were randomly sampled from the CNN/DailyMail dataset. For each news article, four summarisation systems were used to generate summaries: the *seq2seq* and *pointer* models proposed by See et al. (2017), and the *ml* and *ml+rl* models by Paulus et al. (2018). Hence, together with the human-written reference summaries provided in the CNN/DailyMail dataset, each news article has five summaries. Crowd-sourcing workers were recruited to rate the summaries in terms of their fluency, redundancy level and overall quality, on a 3-point Likert scale from $-1$ to 1. Higher scores mean better quality. Each summary was rated by five independent workers. We use the averaged overall quality score for each summary as the ground-truth human judgement.

RL-based summarisation systems assume that the summaries ranked highly by the reward function (e.g. ROUGE) are indeed "good" in terms of human judgement. We define *good* summaries as follows: a summary $y$ for a news article $x$ is good if (**i**) the average human judgement score for $y$ is $\geq 0.5$, and (**ii**) among the five summaries for $x$, $y$ is ranked within the top two. To study to what extent the above assumption is true, we not only measure the summary-level correlation (Spearman's $\rho$ and Pearson's $r$) between the reward function and human judgements, but also count how many good summaries identified by the reward function are indeed good (*G-Pre*), and how many indeed good summaries are identified by the reward function (*G-Rec*). We normalise the reward scores and the human judgements to the same range.

From Table 1, we find that all metrics we consider have low correlation with the human judgement. More importantly, their G-Pre and G-Rec scores are all below .50, which means that more than half of the good summaries identified by the metrics are actually not good, and more than 50%

[1] https://tac.nist.gov/data/

| Metric | $\rho$ | $r$ | G-Pre | G-Rec |
|---|---|---|---|---|
| ROUGE-1 | .290 | .304 | .392 | .428 |
| ROUGE-2 | .259 | .278 | .408 | .444 |
| ROUGE-L | .274 | .297 | .390 | .426 |
| ROUGE-SU4 | .282 | .279 | .404 | .440 |
| BLEU-1 | .256 | .281 | .409 | .448 |
| BLEU-2 | .301 | .312 | .411 | .446 |
| BLEU-3 | .317 | .312 | .409 | .444 |
| BLEU-4 | .311 | .307 | .409 | .446 |
| BLEU-5 | .308 | .303 | .420 | .459 |
| METEOR | .305 | .285 | .409 | .444 |
| InferSent-Cosine | **.329** | **.339** | .417 | .460 |
| BERT-Cosine | .312 | .335 | **.440** | **.484** |

Table 1: Quality of reward metrics. G-Pre and G-Rec are the precision and recall rate of the "good" summaries identified by the metrics, resp. All metrics here require reference summaries. We perform stemming and stop words removal as preprosessing, as they help increase the correlation. For InferSent, the embeddings of the reference/system summaries are obtained by averaging the embeddings of the sentences therein.

of the indeed good summaries cannot be identified by the considered metrics. Hence, hill-climbing on these metrics can hardly guide the RL agents to generate genuinely high-quality summaries. These observations clearly motivate the need for better rewards. Next, we formally formulate the reward learning task.

## 4 Problem Formulation

We focus on reward learning for single-document summarisation in this work, formulated as follows. Let $\mathcal{X}$ be the set of all input documents. For $x \in \mathcal{X}$, let $Y_x$ be the set of all summaries for $x$ that meet the length requirement. A *reward function* $R(x, y; \theta)$ measures the quality of summary $y$ for document $x$, where $\theta$ stands for all parameters for $R$. Note that human judgements can be viewed as the ground-truth reward function, which we denote as $R^*$ henceforth. At training time, suppose we have access to $\overline{\mathcal{X}} \subseteq \mathcal{X}$ documents and $N$ summaries for each $x \in \overline{\mathcal{X}}$, denoted by $\overline{Y}_x = \{y_x^1, \cdots, y_x^N\} \subseteq Y_x$. Hence, we have $|\overline{\mathcal{X}}| \times N$ summaries at training time, and our training set includes the $R^*$ scores for these summaries: $\bigcup_{x \in \overline{\mathcal{X}}} \{R^*(x, y_x^1), \cdots, R^*(x, y_x^N)\}$. Our target is to learn a reward function $R$ that is as "close" to $R^*$ as possible. Depending on the definition of "close", we explore two loss functions for reward learning, detailed below.

**Regression loss.** We first consider reward learning as a regression problem, by measuring the "closeness" between $R$ and $R^*$ by their mean squared errors:

$$\mathcal{L}^{MSE}(\theta) = \frac{1}{|\overline{\mathcal{X}}| \cdot N} \sum_{x \in \overline{\mathcal{X}}} \sum_{i=1}^{N} [R^*(x, y_x^i) - R(x, y_x^i; \theta)]^2. \quad (1)$$

**Cross-entropy loss.** An alternative definition of "closeness" is to measure the "agreement" between $R$ and $R^*$, i.e. for a pair of summaries, whether $R$ and $R^*$ prefer the same summary. For two summaries $y_x^i, y_x^j \in \overline{Y}_x$, we estimate the likelihood that $R$ prefers $y_x^i$ over $y_x^j$ as

$$P(y_x^i \succ y_x^j) = \frac{\exp(r^i)}{\exp(r^i) + \exp(r^j)}, \quad (2)$$

where $r^i = R(x, y_x^i; \theta)$, $r^j = R(x, y_x^j; \theta)$. Note that for each $x \in \overline{\mathcal{X}}$, we have $N$ summaries available in $\overline{Y}_x$. Hence we can construct $N \cdot (N-1)/2$ pairs of summaries for each input $x$. Our target is to minimise the "disagreement" between $R^*$ and $R$ on the $|\overline{\mathcal{X}}| \cdot N \cdot (N-1)/2$ pairs of summaries, formally defined as the cross-entropy loss below:

$$\mathcal{L}^{CE}(\theta) = -\frac{1}{|\overline{\mathcal{X}}| N(N-1)/2} \sum_{x \in \overline{\mathcal{X}}} \sum_{i=1}^{N} \sum_{j>i}^{N} \{ \mathbb{1}[R^*(x, y_x^i) > R^*(x, y_x^j)] \log P(y_x^i \succ y_x^j) + \mathbb{1}[R^*(x, y_x^j) > R^*(x, y_x^i)] \log P(y_x^j \succ y_x^i) \}, \quad (3)$$

where $\mathbb{1}$ is the indicator function. Next, we will introduce our reward learning model that minimises the losses defined in Eq. (1) and (3).

## 5 Reward Learning Model

We explore two neural architectures for $R(x, y; \theta)$: *Multi-Layer Perceptron (MLP)* and *Similarity-Redundancy Matrix (SimRed)*, detailed below.

### 5.1 MLP

A straightforward method for learning $R(x, y_x; \theta)$ is to encode the input document $x$ and summary $y$ as two embeddings, and feed the concatenated embedding into a single-layer MLP to minimise the loss functions Eq. (1) and (3). We consider three text encoders to vectorise $x$ and $y_x$. In supplementary material, we provide figures to further illustrate the architectures of these text encoders.

**CNN-RNN.** We use convolutional neural networks (CNNs) to encode the sentences in the input text, and feed the sentence embeddings into an LSTM to generate the embedding of the whole input text. In the CNN part, convolutions with different filter widths are applied independently as in (Kim, 2014). The most relevant features are selected afterwards with max-over-time pooling. In line with Narayan et al. (2018b), we reverse the order of sentence embeddings before feeding them into the LSTM. This encoder network yields strong performance on summarisation and sentence classification tasks (Narayan et al., 2018a,b).

**PMeans-RNN.** PMeans is a simple yet powerful sentence encoding method (Rücklé et al., 2018). PMeans encodes a sentence by computing the *power means* of the embeddings of the words in the sentence. PMeans uses a parameter $p$ to control the weights for each word embedding: with $p = 1$, each word element is weighted equally, and with the increase of $p$, it assigns higher weights to the elements with higher values. With $p = +\infty$, PMeans is equivalent to element-wise max-pooling. The output of PMeans is passed to an LSTM to produce the final document embedding. Note that only the LSTM is trainable; the $p$ value is decided by the system designer.

**BERT.** We use the pre-trained BERT-Large-Cased model to encode news articles and summaries. The hidden state of the final layer that corresponds to the first token (i.e. "[CLS]") is taken as embedding. Note that the pre-trained BERT models can only encode texts with at most 512 tokens. In line with Alberti et al. (2019), we therefore use a sliding window approach with the offset size of 128 tokens to encode overlength summaries and news articles. We do not fine-tune the BERT model because our dataset is relatively small (only 2,500 summaries and 500 news articles), and the sliding-window of BERT requires much resources to fine-tune.

### 5.2 Similarity-Redundancy Matrix (SimRed)

Good summaries should be more *informative* (i.e. contain information of higher importance from the input documents) and less *redundant* than bad summaries. Based on this intuition, we propose the SimRed architecture, which explicitly measures the informativeness and redundancy of summary $y_x$ for document $x$. SimRed maintains a *Similarity* matrix and a *Redundancy* matrix. In the Similarity matrix, cell $(i, j)$ is the cosine similarity between the embeddings of the $i$th sentence in summary $y_x$ and the $j$th sentence in document $x$. In the Redundancy matrix, each cell contains the square of the cosine similarity of a pair of sentences in summary $y_x$. We use the average over the Similarity matrix cells to measure the informativeness of $y_x$, the average over the Redundancy matrix cells to measure the redundancy, and compute the weighted sum of these two averaged values to yield the reward $R$:

$$R_{\text{SimRed}}(y_x, x) = \frac{\alpha}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} \cos(s_i, d_j) - \frac{1-\alpha}{N(N-1)/2} \sum_{k=1}^{N} \sum_{l>k}^{N} (\cos(s_k, s_l))^2, \quad (4)$$

where $s_i, i = 1, \cdots, N$ indicates the embedding of the $i$th sentence in summary $y_x$, and $d_j, j = 1, \cdots, M$ indicates the embedding of the $j$th sentence in document $x$. The sentence embeddings are generated using CNN, PMeans and BERT as described in §5.1. Because PMeans does not have trainable parameters and BERT is kept fixed, we put a trainable layer on top of them.

## 6 Reward Quality Evaluation

**Experimental Setup.** We perform 5-fold cross-validation on the 2,500 human summaries (described in §3) to measure the performance of our reward $R$. In each fold, the data is split with ratio 64:16:20 for training, validation and test.

The parameters of our model are detailed as follows, decided in a pilot study on one fold of the data split. The CNN-RNN encoder (see §5.1) uses filter widths 1-10 for the CNN part, and uses a unidirectional LSTM with a single layer whose dimension is 600 for the RNN part. For PMeans, we obtain sentence embeddings for each $p \in \{-\infty, +\infty, 1, 2\}$ and concatenate them per sentence. Both these two encoders use the pre-trained GloVe word embeddings (Pennington et al., 2014). On top of these encoders, we use an MLP with one hidden ReLU layer and a linear activation at the output layer. For the MLP that uses CNN-RNN and PMeans-RNN, the dimension of its hidden layer is 100, while for the MLP with BERT as encoder, the dimension of the hidden layer is 1024. As for SimRed, we set $\alpha$ (see Eq. (4)) to be 0.85. The trainable layer on top of BERT and PMeans – when used with SimRed – is a single

layer perceptron whose output dimension is equal to the input dimension.

**Reward Quality.** Table 2 shows the quality of different reward learning models. As a baseline, we also consider the feature-rich reward learning method proposed by Peyrard and Gurevych (2018) (see §2). MLP with BERT as encoder has the best overall performance. Specifically, BERT+MLP+Pref significantly outperforms ($p < 0.05$) all the other models that do not use BERT+MLP, as well as the metrics that rely on reference summaries (see Table 1). P-values between each pair of metrics/rewards can be found in the supplementary material. In general, preference loss (Eq. (2)) yields better performance than regression loss (Eq. (1)), because it "pushes" the reward function to provide correct preferences over summaries, which leads to more precise ranking.

Fig. 1 illustrates the distribution of some rewards/metrics for summaries with different human ratings. In the left-most sub-figure in Fig. 1, we find that, for summaries with average human rating 1.0 (the highest human rating; see §3), their average ROUGE-2 scores are similar to those with lower human ratings, which indicates that ROUGE-2 can hardly distinguish the highest-quality summaries from the rest. We make similar observations for InferSent-Cosine and BERT-Cosine. BERT+MLP+Pref provides higher scores to summaries with higher human ratings (the right-most sub-figure), although it does not use reference summaries. This explains the strong G-Pre and G-Rec scores of BERT+MLP+Pref. The distributions of the other metrics/rewards can be found in the supplementary material. Next, we use the reward learned by BERT+MLP+Pref to train some RL-based summarisation systems.

## 7 RL-based Summarisation with Learned Rewards

We consider two RL-based summarisation systems, an extractive system *NeuralTD* (Gao et al., 2019), and an abstractive system *ExtAbsRL* (Chen and Bansal, 2018). Note that ExtAbsRL is a cross-input RL while NeuralTD is an input-specific RL (see §2). Our study is performed on the test set of the non-anonymised CNN/DailyMail dataset (See et al., 2017), which includes 11,490 news articles and one reference summary for each article.

### 7.1 RL-based Summarisation Systems

**NeuralTD.** NeuralTD is an improved version of the RL-based extractive multi-document summarisation system proposed by Ryang and Abekawa (2012). We briefly recap the original system below. Suppose the RL agent has selected some sentences and has built a draft summary $d$ using the selected sentences. The RL agent maintains a function $V: D \rightarrow \mathbb{R}$, where $D$ denotes the set of all possible draft summaries. $V(d; w)$ estimates the quality and potential of $d$, where $w$ are the learnable parameters in $V$. To select which sentence to add to $d$ next, the agent samples sentences $s \in S$ with distribution

$$\pi(s; w) = \frac{\exp[V((d, s); w)]}{\sum_{s' \in S} \exp[V((d, s'); w)]},$$

where $S$ is the set of all sentences in the input document that has not been added to $d$ yet, and $(d, s)$ is the resulting summary of concatenating $d$ and $s$. The original system proposed by Ryang and Abekawa (2012) models $V$ as a linear function (i.e. $V(d; w) = w \cdot \phi(d)$, where $\phi(d)$ is the vector for draft summary $d$). NeuralTD instead uses a neural network with multiple hidden layers to approximate $V$. Gao et al. (2019) show that NeuralTD significantly outperforms the original linear algorithm in multiple benchmark datasets.

In line with (Gao et al., 2019), we use the *delayed rewards* in NeuralTD: a non-zero reward is provided to the agent only when the agent finishes the sentence selection process (i.e. when agent performs the "end-of-construction" action). The assumption underlying this reward scheme is that the reward function can only precisely measure the quality of the summary when the summary is complete. Besides our learned reward, in order to encourage the agent to select the "lead" sentences (i.e. the first three sentences in each news article), we provide the agent with a small extra reward (0.5) for each "lead" sentence the agent chooses to extract. The value for the extra reward (0.5) is decided in a pilot study, in which we manually check the quality of the generated summaries with different extra rewards (0.1, 0.3, $\cdots$, 0.9).

**ExtAbsRL** has an *extractor* to extract salient sentences and an *abstractor* to rephrase the extracted sentences to generate abstractive summaries. The abstractor is a simple encoder-aligner-decoder model with copying mechanism, which

| Model | Encoder | *Reg. loss (Eq. (1))* $\rho$ | $r$ | G-Pre | G-Rec | *Pref. loss (Eq. (3))* $\rho$ | $r$ | G-Pre | G-Rec |
|---|---|---|---|---|---|---|---|---|---|
| MLP | CNN-RNN | .311 | .340 | .486 | .532 | .318 | .335 | .481 | .524 |
| | PMeans-RNN | .313 | .331 | .489 | .536 | .354 | .375 | .502 | .556 |
| | BERT | **.487** | **.526** | **.544** | **.597** | **.505** | **.531** | **.556** | **.608** |
| SimRed | CNN | .340 | .392 | .470 | .515 | .396 | .443 | .499 | .549 |
| | PMeans | .354 | .393 | .493 | .541 | .370 | .374 | .507 | .551 |
| | BERT | .266 | .296 | .458 | .495 | .325 | .338 | .485 | .533 |
| (Peyrard and Gurevych, 2018) | | .177 | .189 | .271 | .306 | .175 | .186 | .268 | .174 |

Table 2: Summary-level correlation of learned reward functions. All results are averaged over 5-fold cross validations. Unlike the metrics in Table 1, all rewards in this table do not require reference summaries.

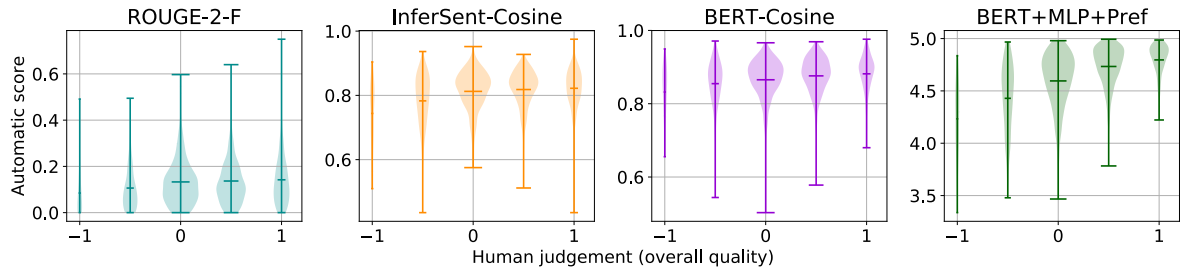


Figure 1: Distributions of some metrics/rewards for summaries with different human ratings. Among the four presented metrics/rewards, only BERT+MLP+Pref (the rightmost sub-figure) does not use reference summaries.

is pre-trained using standard supervised cross-entropy training. The extractor, on the other hand, applies an actor-critic RL algorithm on top of a pointer network. Unlike NeuralTD that uses delayed rewards, ExtAbsRL receives a non-zero reward after adding each new sentence, by computing the ROUGE-L score between the newly added sentence and the corresponding sentence in the reference summary. When the generated summary has more sentences than the reference summary, 0 is given as the reward for the extra sentences. At the final step, ROUGE-1 of the whole generated summary is granted as reward.

We follow the step-wise reward scheme in original ExtAbsRL but, instead of using ROUGE-L to compute the step-wise rewards, we apply our learned reward function to compute the score for the summary with and without the new sentence, and use their difference as reward. Similarly, for the final step reward, we also use our learned reward function. In addition, we force our summariser to stop adding new sentences when the number of tokens in the generated summary is 1.2 times as many as in the reference summary, because we find the abstractive summaries generated by the original ExtAbsRL algorithm are approximately of this length. Since the abstractor in ExtAbsRL is not RL-based, the different reward only influences the extractor.

| System | Reward | R-1 | R-2 | R-L |
|---|---|---|---|---|
| (Kryscinski et al., 2018) | R-L | 40.2 | 17.4 | 37.5 |
| (Narayan et al., 2018b) | R-1,2,L | 40.0 | 18.2 | 36.6 |
| (Chen and Bansal, 2018) | R-L | 41.5 | 18.7 | 37.8 |
| (Dong et al., 2018) | R-1,2,L | 41.5 | 18.7 | 37.6 |
| (Zhang et al., 2018) | NA | 41.1 | 18.8 | 37.5 |
| (Zhou et al., 2018) | NA | 41.6 | 19.0 | 38.0 |
| (Kedzie et al., 2018) | NA | 39.1 | 17.9 | 35.9 |
| (ours) NeuralTD | Learned | 39.6 | 18.1 | 36.5 |

Table 3: Full-length ROUGE F-scores of some recent RL-based (upper) and supervised (middle) extractive summarisation systems, as well as our system with learned rewards (bottom). R-1/2/L stands for ROUGE-1/2/L. Our system maximises the learned reward instead of ROUGE, hence receives lower ROUGE scores.

### 7.2 Extractive Summarisation

Table 3 presents the ROUGE scores of our system (NeuralTD+LearnedRewards) and multiple state-of-the-art systems. The summaries generated by our system receive decent ROUGE metrics, but are lower than most of the recent systems, because our learned reward is optimised towards high correlation with human judgement instead of ROUGE metrics.

To measure the human ratings on the gener-

| | Ours | Refresh | ExtAbsRL |
|---|---|---|---|
| Avg. Human Rating | **2.52** | 2.27 | 1.66 |
| Best% | **70.0** | 33.3 | 6.7 |

Table 4: Human evaluation on extractive summaries. Our system receives significantly higher human ratings on average. "Best%": in how many percentage of documents a system receives the highest human rating.

| Reward | R-1 | R-2 | R-L | Human | Pref% |
|---|---|---|---|---|---|
| R-L (original) | 40.9 | 17.8 | 38.5 | 1.75 | 15 |
| Learned (ours) | 39.2 | 17.4 | 37.5 | **2.20** | **75** |

Table 5: Performance of ExtAbsRL with different reward functions, measured in terms of ROUGE (center) and human judgements (right). Using our learned reward yields significantly ($p = 0.0057$) higher average human rating. "Pref%": in how many percentage of documents a system receives the higher human rating.

ated summaries, we invited five users to read and rate the summaries from three systems: NeuralTD+LearnedReward, the Refresh system (Narayan et al., 2018b) and the extractive version of the ExtAbsRL system, which only extracts salient sentences and does not apply sentence rephrasing. We selected Refresh and ExtAbsRL because they both have been reported to receive higher human ratings than the strong system proposed by See et al. (2017).

We randomly sampled 30 news articles from the test set in CNN/DailyMail, and asked the five participants to rate the three summaries for each article on a 3-point Likert scale from 1 to 3, where higher scores mean better overall quality. We asked them to consider the *informativeness* (whether the summary contains most important information in the article) and *conciseness* (whether the summary is concise and does not contain redundant information) in their ratings.

Table 4 presents the human evaluation results. summaries generated by NeuralTD receives significantly higher human evaluation scores than those by Refresh ($p = 0.0088$, double-tailed t-test) and ExtAbsRL ($p \ll 0.01$). Also, the average human rating for Refresh is significantly higher ($p \ll 0.01$) than ExtAbsRL, despite receiving significantly higher ROUGE scores than both Refresh and NeuralTD (see Table 3). We find that the summaries generated by ExtAbsRL include more tokens (94.5) than those generated by Refresh (83.4) and NeuralTD (85.6). Sun et al. (2019) recently show that, for summaries whose lengths are in the range of 50 to 110 tokens, longer summaries receive higher ROUGE-F1 scores. We believe this is the reason why ExtAbsRL has higher ROUGE scores. On the other hand, ExtAbsRL extracts more redundant sentences: four out of 30 summaries by ExtAbsRL include redundant sentences, while Refresh and NeuralTD do not generate summaries with two identical sentences therein. Users are sensitive to the redundant sentences in summaries: the average human rating for redundant summaries is 1.2, lower than the average rating for the other summaries generated by ExtAbsRL (1.66). To summarise, by using our learned reward function in training an extractive RL summariser (NeuralTD), the generated summaries receive significantly higher human ratings than the state-of-the-art systems.

### 7.3 Abstractive Summarisation

Table 5 compares the ROUGE scores of using different rewards to train the extractor in ExtAbsRL (the abstractor is pre-trained, and is applied to rephrase the extracted sentences). Again, when ROUGE is used as rewards, the generated summaries have higher ROUGE scores.

We randomly sampled 20 documents from the test set in CNN/DailyMail and asked three users to rate the quality of the two summaries generated with different rewards. We asked the users to consider not only the informativeness and conciseness of summaries, but also their grammaticality and faithfulness (whether the information in the summary is consistent with that in the original news). It is clear from Table 5 that using the learned reward helps the RL-based system generate summaries with significantly higher human ratings. Furthermore, we note that the overall human ratings for the abstractive summaries are lower than the extractive summaries (compared to Table 4). Qualitative analysis suggests that the poor overall rating may be caused by occasional information inconsistencies between a summary and its source text: for instance, a sentence in the source article reads *"after Mayweather was almost two hours late for his workout , Pacquiao has promised to be on time"*, but the generated summary outputs *"Mayweather has promised to be on time for the fight"*. High redundancy is another reason for the low human ratings: ExtAbsRL generates six summaries with redundant sentences when applying ROUGE-L as reward, while the number drops to

two when the learned reward is applied.

## 8 Conclusion & Discussion

In this work, we focus on Reinforcement Learning (RL) based summarisation, and propose a reward function directly learned from human ratings on summaries' overall quality. Our reward function only takes the source text and the generated summary as input (i.e. does not require reference summaries), and correlates significantly better with human judgements than existing metrics (e.g. ROUGE and METEOR, which require reference summaries). We use our learned reward to train both extractive and abstractive summarisation systems. Experiments show that the summaries generated from our learned reward outperform those by the state-of-the-art systems, in terms of human judgements. Considering that our reward is learned from only 2,500 human ratings on 500 summaries, while the state-of-the-art systems require two orders of magnitude (287k) more document-reference pairs for training, this work clearly shows that reward learning plus RL-based summarisation is able to leverage a relatively small set of human rating scores to produce high-quality summaries.

For future work, we plan to test our method in other summarisation tasks (e.g. multi-document summarisation) and downstream tasks of summarisation (e.g. investigating whether users can correctly answer questions by reading our summaries instead of the original documents). Also, we believe the "learning reward from human judgements" idea has potential to boost the performance of RL in other natural language generation applications, e.g. translation, sentence simplification and dialogue generation.

## Acknowledgements

This work has been supported by the German Research Foundation (DFG), as part of the QA-EduInf project (GU 798/18-1 and RI 803/12-1) and through the German-Israeli Project Cooperation (DIP, DA 1600/1-1 and GU 798/17-1).

## References

Chris Alberti, Kenton Lee, and Michael Collins. 2019. A BERT Baseline for the Natural Questions. *arXiv e-prints*.

Kristjan Arumae and Fei Liu. 2019. Guiding extractive summarization with question-answering rewards. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2566–2577, Minneapolis, USA.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal.

Arun Chaganty, Stephen Mussmann, and Percy Liang. 2018. The price of debiasing automatic metrics in natural language evalaution. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, Volume 1: Long Papers*, pages 643–653, Melbourne, Australia.

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, Volume 1: Long Papers*, pages 675–686, Melbourne, Australia.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota.

Yue Dong, Yikang Shen, Eric Crawford, Herke van Hoof, and Jackie Chi Kit Cheung. 2018. Banditsum: Extractive summarization as a contextual bandit. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3739–3748, Brussels, Belgium.

Yang Gao, Christian M. Meyer, and Iryna Gurevych. 2018. APRIL: interactively learning to summarise by combining active preference learning and reinforcement learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4120–4130, Brussels, Belgium.

Yang Gao, Christian M. Meyer, and Iryna Gurevych. 2019. Preference-based Interactive Multi-Document Summarisation. *arXiv e-prints*.

Yang Gao, Christian M. Meyer, Mohsen Mesgar, and Iryna Gurevych. 2019. Reward learning for efficient reinforcement learning in extractive document summarisation. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 2350–2356, Macao, China.

Max Grusky, Mor Naaman, and Yoav Artzi. 2018. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 708–719, New Orleans, Louisiana, USA.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015*, pages 1693–1701, Montreal, Quebec, Canada.

Chris Kedzie, Kathleen R. McKeown, and Hal Daumé III. 2018. Content selection in deep learning models of summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1818–1828, Brussels, Belgium.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751, Doha, Qatar.

Julia Kreutzer, Artem Sokolov, and Stefan Riezler. 2017. Bandit structured prediction for neural sequence-to-sequence learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, Volume 1: Long Papers*, pages 1503–1513, Vancouver, Canada.

Julia Kreutzer, Joshua Uyheng, and Stefan Riezler. 2018. Reliability and learnability of human bandit feedback for sequence-to-sequence reinforcement learning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, Volume 1: Long Papers*, pages 1777–1788, Melbourne, Australia.

Wojciech Kryscinski, Romain Paulus, Caiming Xiong, and Richard Socher. 2018. Improving abstraction in text summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1808–1817, Brussels, Belgium.

Alon Lavie and Michael J. Denkowski. 2009. The meteor metric for automatic evaluation of machine translation. *Machine Translation*, 23(2-3):105–115.

Chin-Yew Lin. 2004a. Looking for a few good metrics: Rouge and its evaluation. In *NTCIR Workshop*.

Chin-Yew Lin. 2004b. ROUGE: A package for automatic evaluation of summaries. In *ACL Workshop "Text Summarization Branches Out"*.

Annie Louis and Ani Nenkova. 2013. Automatically assessing machine summary content without a gold standard. *Computational Linguistics*, 39(2):267–300.

Shashi Narayan, Ronald Cardenas, Nikos Papasarantopoulos, Shay B. Cohen, Mirella Lapata, Jiangsheng Yu, and Yi Chang. 2018a. Document modeling with external attention for sentence extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, Volume 1: Long Papers*, pages 2020–2030, Melbourne, Australia.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018b. Ranking sentences for extractive summarization with reinforcement learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1747–1759, New Orleans, Louisiana, USA.

Andrew Y. Ng and Stuart J. Russell. 2000. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 663–670, Stanford University, Stanford, CA, USA.

Jekaterina Novikova, Ondrej Dusek, Amanda Cercas Curry, and Verena Rieser. 2017. Why we need new evaluation metrics for NLG. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2241–2252, Copenhagen, Denmark.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, PA, USA.

Ramakanth Pasunuru and Mohit Bansal. 2018. Multi-reward reinforced summarization with saliency and entailment. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 646–653, New Orleans, Louisiana, USA.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *6th International Conference on Learning Representations, Conference Track Proceedings*, Vancouver, BC, Canada.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language*

*Processing (EMNLP)*, pages 1532–1543, Doha, Qatar.

Maxime Peyrard, Teresa Botschen, and Iryna Gurevych. 2017. Learning to score system summaries for better content selection evaluation. In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 74–84, Copenhagen, Denmark.

Maxime Peyrard and Iryna Gurevych. 2018. Objective function learning to match human judgements for optimization-based summarization. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 654–660, New Orleans, Louisiana, USA.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100, 000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, USA.

Cody Rioux, Sadid A. Hasan, and Yllias Chali. 2014. Fear the REAPER: A system for automatic multi-document summarization with reinforcement learning. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014*, pages 681–690, Doha, Qatar.

Andreas Rücklé, Steffen Eger, Maxime Peyrard, and Iryna Gurevych. 2018. Concatenated Power Mean Word Embeddings as Universal Cross-Lingual Sentence Representations. *arXiv e-prints*.

Seonggi Ryang and Takeshi Abekawa. 2012. Framework of automatic text summarization using reinforcement learning. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 256–265, Jeju Island, Korea.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, Volume 1: Long Papers*, pages 1073–1083, Vancouver, Canada.

Simeng Sun, Ori Shapira, Ido Dagan, and Ani Nenkova. 2019. How to Compare Summarizers without Target Length? Pitfalls, Solutions and Re-Examination of the Neural Summarization Literature. In *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation*, pages 21–29, Minneapolis, Minnesota.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana, USA.

Xingxing Zhang, Mirella Lapata, Furu Wei, and Ming Zhou. 2018. Neural latent extractive document summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 779–784, Brussels, Belgium.

Qingyu Zhou, Nan Yang, Furu Wei, Shaohan Huang, Ming Zhou, and Tiejun Zhao. 2018. Neural document summarization by jointly learning to score and select sentences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, Volume 1: Long Papers*, pages 654–663, Melbourne, Australia.

# Mixture Content Selection for Diverse Sequence Generation

**Jaemin Cho**[1]* **Minjoon Seo**[2,3] **Hannaneh Hajishirzi**[1,3]
Allen Institute for AI[1] Clova AI, NAVER[2] University of Washington[3]
heythisischo@gmail.com {minjoon,hannaneh}@cs.washington.edu

## Abstract

Generating diverse sequences is important in many NLP applications such as question generation or summarization that exhibit semantically one-to-many relationships between source and the target sequences. We present a method to explicitly separate diversification from generation using a general plug-and-play module (called SELECTOR) that wraps around and guides an existing encoder-decoder model. The diversification stage uses a mixture of experts to sample different binary masks on the source sequence for diverse content selection. The generation stage uses a standard encoder-decoder model given each selected content from the source sequence. Due to the non-differentiable nature of discrete sampling and the lack of ground truth labels for binary mask, we leverage a proxy for ground-truth mask and adopt stochastic hard-EM for training. In question generation (SQuAD) and abstractive summarization (CNN-DM), our method demonstrates significant improvements in accuracy, diversity and training efficiency, including state-of-the-art top-1 accuracy in both datasets, 6% gain in top-5 accuracy, and 3.7 times faster training over a state-of-the-art model. Our code is publicly available at https://github.com/clovaai/FocusSeq2Seq.

## 1 Introduction

Generating target sequences given a source sequence has applications in a wide range of problems in NLP with different types of relationships between the source and target sequences. For instance, paraphrasing or machine translation exhibit a one-to-one relationship because the source and the target should carry the same meaning. On the other hand, summarization or question generation exhibit one-to-many relationships because a single source often results in diverse target sequences with different semantics. Fig. 1 shows different questions that can be generated from a given passage.

**Source Passage**: in december 1878 , tesla left graz and severed all relations with his family to hide the fact that he dropped out of school .
**Target**: what did tesla do in december 1878?

**Focus 1**: in december 1878 , tesla left graz and severed all relations with his family to hide the fact that he dropped out of school.
(Ours) ⇒ what did tesla do?

**Focus 2**: in december 1878 , tesla left graz and severed all relations with his family to hide the fact that he dropped out of school.
(Ours) ⇒ what did tesla do in december 1878?

**Focus 3**: in december 1878 , tesla left graz and severed all relations with his family to hide the fact that he dropped out of school .
(Ours) ⇒ what did tesla do to hide he dropped out of school?

Figure 1: Sample questions produced by our method from given passage-answer pair (answer is underlined). Our method generates diverse questions, by selecting different tokens to focus (colored) in contrast to 3-mixture decoder (Shen et al., 2019) that generates 3 identical questions: "what did tesla do to hide the fact that he dropped out of school?".

Encoder-decoder models (Cho et al., 2014) are widely used for sequence generation, most notably in machine translation where neural models are now often almost as good as human translators in some language pairs. However, a standard encoder-decoder often shows a poor performance when it attempts to produce multiple, diverse outputs. Most recent methods for diverse sequence generation leverage diversifying decoding steps through alternative search algorithms (Fan et al., 2018; Vijayakumar et al., 2018) or mixture of decoders (He et al., 2018; Shen et al., 2019). These methods promote diversity at the decoding

*Most work done during internship at Clova AI.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3121–3131, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

step, while a more focused selection of the source sequence can lead to diversifying the semantics of the generated target sequences.

In this paper, we present a method for diverse generation that separates diversification and generation stages. The diversification stage leverages content selection to map the source to multiple sequences, where each mapping is modeled by *focusing* on different tokens in the source (one-to-many mapping). The generation stage uses a standard encoder-decoder model to generate a target sequence given each selected content from the source (one-to-one mapping). We present a generic module called SELECTOR that is specialized for diversification. This module can be used as a plug-and-play to an arbitrary encoder-decoder model for generation without architecture change.

The SELECTOR module leverages a mixture of experts (Jacobs et al., 1991; Eigen et al., 2014) to identify diverse key contents to focus on during generation. Each mixture samples a sequential latent variable modeled as a binary mask on every source sequence token. Then an encoder-decoder model generates multiple target sequences given these binary masks along with the original source tokens. Due to the non-differentiable nature of discrete sampling, we adopt stochastic hard-EM for training SELECTOR. To mitigate the lack of ground truth annotation for the mask (content selection), we use the overlap between the source and target sequences as a simple proxy for the ground-truth mask.

We experiment on question generation and abstractive summarization tasks and show that our method achieves the best trade-off between accuracy and diversity over previous models on SQuAD (Rajpurkar et al., 2016) and CNN-DM (Hermann et al., 2015; Nallapati et al., 2016; See et al., 2017) datasets. In particular, compared to the recently-introduced mixture decoder (Shen et al., 2019) that also aims to diversify outputs by creating multiple decoders, our modular method not only demonstrates better accuracy and diversity, but also trains 3.7 times faster.

## 2 Related Work

**Diverse Search Algorithms** Beam search, the most commonly used search algorithm for decoding, is known to produce samples that are short, contain repetitive phrases, and share majority of their tokens. Hence several methods are introduced to diversify search algorithms for decoding. Graves (2013); Chorowski and Jaitly (2017) tune temperature hyperparameter in softmax function. Vijayakumar et al. (2018); Li et al. (2016b) penalize similar samples during beam search in order to obtain diverse set of samples. Cho (2016) adds random noise to RNN decoder hidden states. Fan et al. (2018) sample tokens from top-k tokens at each decoding step. Our method is orthogonal to these search-based strategies, in that they diversify *decoding* while our method diversifies which content to be focused during *encoding*. Moreover, our empirical results show diversification with stochastic sampling hurts accuracy significantly.

**Deep Mixture of Experts** Several methods adopt a deep mixture of experts (MoE) (Jacobs et al., 1991; Eigen et al., 2014) to diversify decoding steps. Yang et al. (2018) introduce soft mixture of softmax on top of the output layer of RNN language model. He et al. (2018); Shen et al. (2019) introduce mixture of decoders with uniform mixing coefficient to improve diversity in machine translation. Among these, the closest to ours is the mixture decoder (Shen et al., 2019) that also adopts hard-EM for training, where a minimum-loss predictor is assigned to each data point, which is also known as multiple choice learning (Guzman-Rivera et al., 2012; Lee et al., 2016). While Shen et al. (2019) makes RNN decoder as a MoE, we make SELECTOR as a MoE to diversify content selection and let the encoder-decoder models one-to-one generation. As shown in our empirical results, our method achieves a better accuracy-diversity trade-off while reducing training time significantly.

**Variational Autoencoders** Variational Autoencoders (VAE) (Kingma and Welling, 2013) are used for diverse generation in several tasks, such as language modeling (Bowman et al., 2016), machine translation (Zhang et al., 2016; Su et al., 2018; Deng et al., 2018; Shankar and Sarawagi, 2019), and conversation modeling (Serban et al., 2017; Wen et al., 2017; Zhao et al., 2017; Park et al., 2018; Wen and Luong, 2018; Gu et al., 2019). These methods sample diverse latent variables from an approximate posterior distribution, but often suffer from a posterior collapse where the sampled latent variables are ignored (Bowman et al., 2016; Park et al., 2018; Kim et al., 2018b;

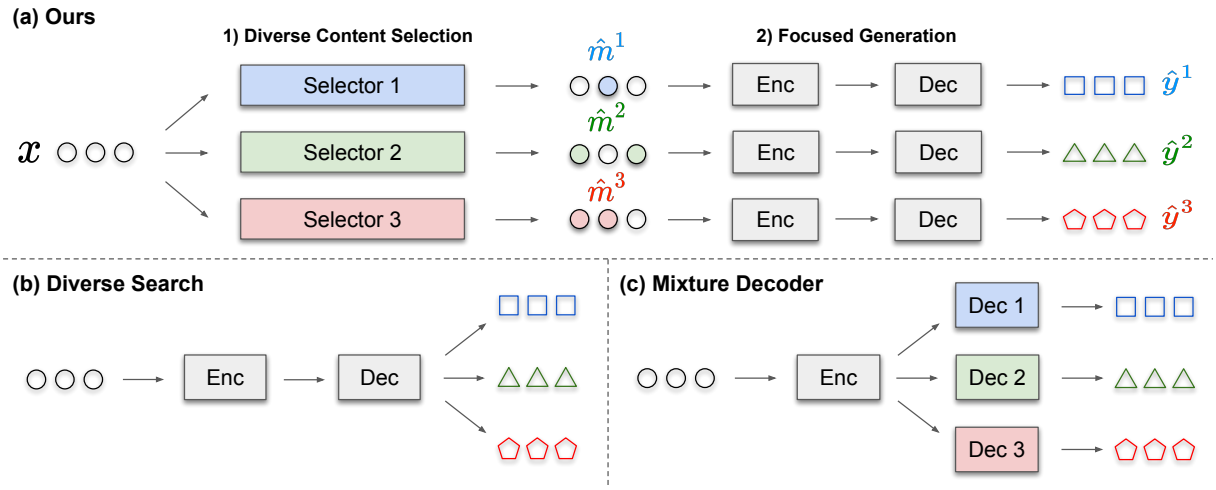


Figure 2: Overview of diverse sequence-to-sequence generation methods. (a) refers to our two-stage approach described throughout Section. 3, (b) refers to search-based methods (Vijayakumar et al., 2018; Li et al., 2016b; Fan et al., 2018), and (c) refers to mixture decoders (Shen et al., 2019; He et al., 2018).

Dieng et al., 2018; Xu and Durrett, 2018; He et al., 2019; Razavi et al., 2019). This is also observed in our initial experiments and Shen et al. (2019), where MoE-based methods significantly outperforms VAE-based methods because of the posterior collapse. Moreover, we observe that sampling mixtures makes training more stable compared to stochastic sampling of latent variables. Furthermore, our latent structure as a sequence of binary variables is different from most VAEs which use a fixed-size continuous latent variable. This gives a finer-grained control and interpretability on where to focus, especially when source sequence is long.

**Diversity-Promoting Regularization** Adding regularization to objective functions is used to diversify generation. Li et al. (2016a) introduce a term maximizing mutual information between source and target sentences. Chorowski and Jaitly (2017); Kalyan et al. (2018) introduce terms enforcing knowledge transfer among similar annotations. Our work is orthogonal to these methods and can potentially benefit from adding these regularization terms to our objective function.

**Content Selection in NLP** Selecting important parts of the context has been an important step in NLP applications (Reiter and Dale, 2000). Most recently, Ke et al. (2018); Min et al. (2018) conduct soft-/ hard-selection of key parts from source passages for question answering. Zhou et al. (2017b) use soft gating on the source document encoder for abstractive summarization. Li et al. (2018) guide abstractive summarization models with off-the-shelf keyword extractors. The most relevant work to ours are Subramanian et al. (2018) and Gehrmann et al. (2018). Subramanian et al. (2018) use a pointer network (Vinyals et al., 2015) for extracting key phrases for question generation and Gehrmann et al. (2018) use content selector to limit copying probability for abstractive summarization. The main purpose of these approaches is to enhance accuracy, while our method uses diverse content selection to enhance both accuracy and diversity (refer to our empirical results). Additionally, our method allows models to learn how to utilize information from the selected content, whereas Gehrmann et al. (2018) manually limit the copying mechanism on non-selected contents.

## 3 Method

In this paper, we focus on sequence generation tasks such as question generation and summarization that have one-to-many relationship. More formally, given a source sequence $\mathbf{x} = (x_1 \dots x_S) \in \mathcal{X}$, our goal is to model a conditional multimodal distribution for the target sequence $p(\mathbf{y}|\mathbf{x})$ that assigns high values on $p(\mathbf{y}^1|\mathbf{x}) \dots p(\mathbf{y}^K|\mathbf{x})$ for $K$ valid mappings $\mathbf{x} \rightarrow \mathbf{y}^1 \dots \mathbf{x} \rightarrow \mathbf{y}^K$. For instance, Fig. 1 illustrates $K = 3$ different valid questions generated for a given passage.

For learning a multimodal distribution, it is not appropriate to use a generic encoder-decoder (Cho et al., 2014) that minimizes for the expected value of the log probability of all the valid mappings.

This can lead to a suboptimal mapping that is in the middle of the targets but not near any of them. As a solution, we propose to (1) introduce a latent variable called *focus* that factorizes the distribution into two stages, *select* and *generate* (Section 3.1), and (2) independently train the factorized distributions (Section 3.2).

### 3.1 Select and Generate

In order to factorize the multimodal distribution into the two stages (*select* and *generate*), we introduce a latent variable called *focus*. The intuition is that in the *select* stage we sample several meaningful *focus*, each of which indicates which part of the source sequence should be considered important. Then in the *generate* stage, each sampled focus biases the generation process towards being conditioned on the focused content.

Formally, we model *focus* with a sequence of binary variable, each of which corresponds to each token in the input sequence, i.e. $\mathbf{m} = \{m_1 \dots m_S\} \in \{0,1\}^S$. The intuition is that $m_t = 1$ indicates $t$-th source token $x_t$ should be *focused* during sequence generation. For instance, in Fig. 1, colored tokens (green, red, or blue) show that different tokens are focused (i.e. values are 1) for different focus samples (out of 3). We first use the latent variable $\mathbf{m}$ to factorize $p(\mathbf{y}|\mathbf{x})$,

$$p(\mathbf{y}|\mathbf{x}) = \mathbb{E}_{\mathbf{m}\sim p_\phi(\mathbf{m}|\mathbf{x})}[p_\theta(\mathbf{y}|\mathbf{m},\mathbf{x})] \quad (1)$$

where $p_\phi(\mathbf{m}|\mathbf{x})$ is *selector* and $p_\theta(\mathbf{y}|\mathbf{x},\mathbf{m})$ is *generator*. The factorization separates focus selection from generation so that modeling multimodality (diverse outputs) can be solely handled in the *select* stage and *generate* stage can solely concentrate on the generation task itself. We now describe each component in more details.

**Selector** In order to directly control the diversity of the SELECTOR's outputs, we model it as a hard mixture of experts (hard-MoE) (Jacobs et al., 1991; Eigen et al., 2014), where each expert specializes in focusing on different parts of the source sequences. In Fig. 1 and 2, focus produced by each SELECTOR expert is colored differently. We introduce a multinomial latent variable $z \in \mathbb{Z}$, where $\mathbb{Z} = \{1 \dots K\}$, and let each focus $\mathbf{m}$ be assigned to one of K experts with uniform prior $p(z|\mathbf{x}) = \frac{1}{K}$. With this mixture setting, $p(\mathbf{m}|\mathbf{x})$ is recovered as follows.

$$\begin{aligned} p(\mathbf{m}|\mathbf{x}) &= \mathbb{E}_{z\sim p(z|\mathbf{x})}[p_\phi(\mathbf{m}|\mathbf{x},z)] \\ &= \frac{1}{K}\sum_{z}^{1\dots K} p_\phi(\mathbf{m}|\mathbf{x},z) \end{aligned} \quad (2)$$

We model SELECTOR with a single-layer Bi-directional Gated Recurrent Unit (Bi-GRU) (Cho et al., 2014) followed by two fully-connected layers and a Sigmoid activation. We feed (current hidden state $\mathbf{h}_t$, first and last hidden state $\mathbf{h}_1, \mathbf{h}_S$, and expert embedding $\mathbf{e}_z$) to a fully-connected layers (FC). Expert embedding $\mathbf{e}_z$ is unique for each expert and is trained from a random initialization. From our initial experiments, we found this parallel focus inference to be more effective than auto-regressive pointing mechanism (Vinyals et al., 2015; Subramanian et al., 2018). The distribution of the focus conditioned on the input $\mathbf{x}$ and the expert id $z$ is the Bernoulli distribution of the resulting values,

$$\begin{aligned} (\mathbf{h}_1 \dots \mathbf{h}_S) &= \text{Bi-GRU}(\mathbf{x}) \\ o_t^z &= \sigma(\text{FC}([\mathbf{h}_t;\mathbf{h}_1;\mathbf{h}_S;\mathbf{e}_z])) \\ p_\phi(m_t|\mathbf{x},z) &= \text{Bernoulli}(o_t^z) \end{aligned} \quad (3)$$

To prevent low quality experts from *dying* during training, we let experts share all parameters, except for the individual expert embedding $\mathbf{e}_z$. We also reuse the word embedding of the generator in the word embedding of SELECTOR to promote cooperative knowledge sharing between SELECTOR and generator. With these parameter sharing techniques, adding a mixture of SELECTOR experts increases only a slight amount of parameters (GRU, FC, and $\mathbf{e}_z$) to sequence-to-sequence models.

**Generator** For maximum diversity, we sample one focus from each SELECTOR expert to approximate $p_\phi(\mathbf{m}|\mathbf{x})$. For a deterministic behavior, we threshold $o_t^z$ with a hyperparameter $th$ instead of sampling from the Bernoulli distribution. This gives us a list of focus $\mathbf{m}^1 \dots \mathbf{m}^K$ coming from $K$ experts. Each focus $\mathbf{m}^z = (m_1^z \dots m_S^z)$ is encoded as embeddings and concatenated with the input embeddings of the source sequence $\mathbf{x} = (x_1 \dots x_S)$. An off-the-shelf generation function such as encoder-decoder can be used for modeling $p(\mathbf{y}|\mathbf{m},\mathbf{x})$, as long as it accepts a stream of input embeddings. We use an identical generation function with $K$ different focus samples to produce $K$ different diverse outputs.

**Algorithm 1:** Training
(N: Dataset size, K: Number of mixtures)

**Data:** $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{m}^{\text{guide (i)}})\}_{i=1}^{N}$

1 **for** $i \in \{1 \dots N\}$ **do**

/* Selector $p_\phi(\mathbf{m}|\mathbf{x}, z)$ E-step */

2 **for** $z \in \{1 \dots K\}$ **do**

3 $L_{\text{select}}^{(i)\ z} = -\log p_\phi(\mathbf{m}^{\text{guide }(i)}|\mathbf{x}^{(i)}, z)$

4 **end**

5 $z^{\text{best }(i)} = \underset{z}{\text{argmin}}\, L_{\text{select}}^{(i)\ z}$

/* Selector $p_\phi(\mathbf{m}|\mathbf{x}, z)$ M-step */

6 $\phi_{z^{\text{best }(i)}} = \phi_{z^{\text{best }(i)}} - \alpha \nabla_{\phi_{z^{\text{best }(i)}}} L_{\text{select}}^{(i)\ z^{\text{best }(i)}}$

/* Generator $p_\theta(\mathbf{y}|\mathbf{x}, \mathbf{m})$ Update */

7 $L^{(i)}{}_{\text{gen}} = -\log p_\theta(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}, \mathbf{m}^{\text{guide (i)}})$

8 $\theta = \theta - \alpha \nabla_\theta L_{\text{gen}}^{(i)}$

9 **end**

### 3.2 Training

Marginalizing the Bernoulli distribution in Eq. 3 by enumerating all possible focus is intractable since the cardinality of focus space $2^S$ grows exponentially with source sequence length $S$. Policy gradient (Williams, 1992; Yu et al., 2017) or Gumbel-softmax (Jang et al., 2017; Maddison et al., 2017) are often used to propagate gradients through a stochastic process, but we empirically found that these do not work well. We instead create *focus guide* and use it to independently and directly train the SELECTOR and the generator. Formally, a focus guide $\mathbf{m}^{\text{guide}} = (m_1^{\text{guide}} \dots m_S^{\text{guide}})$ is a simple proxy of whether a source token is *focused* during generation. We set $t$-th focus guide $m_t^{\text{guide}}$ to 1 if $t$-th source token $x_t$ is *focused* in target sequence $\mathbf{y}$ and 0 otherwise. During training, $\mathbf{m}^{\text{guide}}$ acts as a target for SELECTOR and is a given input for generator (teacher forcing). During inference, $\hat{\mathbf{m}}$ is sampled from SELECTOR and fed to the generator.

In question generation, we set $m_t^{\text{guide}}$ to 1 if there is a target question token which shares the same word stem with passage token $x_t$. Then we set $m_t^{\text{guide}}$ to 0 if $x_t$ is a stop word or is inside the answer phrase. In abstractive summarization, we generate focus guide using copy target generation by Gehrmann et al. (2018), where they set source document token $x_t$ is *copied* if it is part of the longest possible subsequence that overlaps with the target summary.

Alg. 1 describes the overall training process, which first uses stochastic hard-EM (Neal and Hinton, 1998; Lee et al., 2016) for training the SELECTOR and then the canonical MLE for training the generator.

**E-step** (line 2-5 in Alg. 1) we sample focus from all experts and compute their losses $-\log p_\phi(\mathbf{m}^{\text{guide}}|\mathbf{x}, z)$. Then we choose an expert $z^{\text{best}}$ with minimum loss.

$$z^{\text{best}} = \underset{z}{\text{argmin}} -\log p_\phi(\mathbf{m}^{\text{guide}}|\mathbf{x}, z) \quad (4)$$

**M-step** (line 6 in Alg. 1) we only update the parameters of the chosen expert $z^{\text{best}}$.

**Training Generator** (line 7-8 in Alg. 1) The generator is independently trained using conventional teacher forcing, which minimizes $-\log p_\theta(\mathbf{y}|\mathbf{x}, \mathbf{m}^{\text{guide}})$.

## 4 Experimental Setup

We describe our experimental setup for question generation (Section 4.1) and abstractive summarization (Section 4.2). For both tasks, we use an off-the-shelf, task-specific encoder-decoder-based model for the generator and show how adding SELECTOR can help to diversify the output of an arbitrary generator. To evaluate the contribution of SELECTOR we additionally compare our method with previous diversity-promoting methods as the baseline (Section 4.3).

### 4.1 Question Generation

Question generation is the task of generating a question from a passage-answer pair. Answers are given as a span in the passage (See Fig.1).

**Dataset** We conduct experiments on SQuAD (Rajpurkar et al., 2016) and use the same dataset split of Zhou et al. (2017a), resulting in 86,635, 8,965, and 8,964 source-target pairs for training, validation, and test, respectively. Both source passages and target questions are single sentences. The average length of source passage and target question are 32 and 11 tokens.

**Generator** We use NQG++ (Zhou et al., 2017a) as the generator, which is an RNN-based encoder-decoder architecture with copying mechanism (Gulcehre et al., 2016).

### 4.2 Abstractive Summarization

Abstractive summarization is the task of generating a summary sentence from a source document that consists of multiple sentences.

**Dataset** We conduct experiments on the non-anonymized version of CNN-DM dataset (Hermann et al., 2015; Nallapati et al., 2016; See et al., 2017), whose training, validation, test splits have size of 287,113, 13,368, and 11,490 source-target pairs, respectively. The average length of the source documents and target summaries are 386 and 55 tokens. Following See et al. (2017), we truncate source and target sentences to 400 and 100 tokens during training.

**Generator** We use Pointer Generator (PG) (See et al., 2017) as the generator for summarization, which also leverages RNN-based encoder-decoder architecture and copying mechanism, and uses coverage loss to avoid repetitive phrases.

### 4.3 Baselines

For each task, we compare our method with other techniques which promote diversity at the decoding step. In particular, we compare with recent diverse search algorithms including Truncated Sampling (Fan et al., 2018), Diverse Beam Search (Vijayakumar et al., 2018), and Mixture Decoder (Shen et al., 2019). We implement these methods with NQG++ and PG. For each method, we generate K = (3 and 5) hypotheses from each source sequence. For search-based baselines (Fan et al., 2018; Vijayakumar et al., 2018), we select the top-k candidates after generation. For mixture models (Shen et al. (2019) and ours), we conduct greedy decoding from each mixture for fair comparison with search-based methods in terms of speed/memory usage.

**Beam Search** This baseline keeps $K$ hypotheses with highest log-probability scores at each decoding step.

**Diverse Beam Search** This baseline adds a diversity promoting term to log-probability when scoring hypotheses in beam search, Following Vijayakumar et al. (2018), we use hamming diversity and diversity strength $\lambda = 0.5$.

**Truncated Sampling** This baseline randomly samples words from top-10 candidates of the distribution at the decoding step (Fan et al., 2018).

**Mixture Decoder** This baseline constructs a hard-MoE of K decoders with uniform mixing coefficient (referred as hMup in Shen et al. (2019)) and conducts parallel greedy decoding. All decoders share all parameters but use different embeddings for start-of-sequence token.

**Mixture Selector (Ours)** We construct a hard-MoE of K SELECTORs with uniform mixing coefficient that infers K different focus from source sequence. Guided by K focus, generator conducts parallel greedy decoding.

### 4.4 Metrics: Accuracy and Diversity

We use metrics introduced by previous works (Ott et al., 2018; Vijayakumar et al., 2018; Zhu et al., 2018) to evaluate the diversity promoting approaches. These metrics are extensions over BLEU-4 (Papineni et al., 2002) and ROUGE-2 $F_1$-score (Lin, 2004) and aim to evaluate the trade-off between accuracy and diversity.

**Top-1 metric (⇑)** This measures the Top-1 accuracy among the generated K-best hypotheses. The accuracy is measured using a corpus-level metric, i.e., BLEU-4 or ROUGE-2.

**Oracle metric (⇑)** This measures the quality of the target distribution coverage among the Top-K generated target sequences (Ott et al., 2018; Vijayakumar et al., 2018). Given an optimal ranking method (oracle), this metric measures the upper bound of Top-1 accuracy by comparing the best hypothesis with the target. Concretely, we generate hypotheses $\{\hat{\mathbf{y}}^1 \dots \hat{\mathbf{y}}^K\}$ from each source $\mathbf{x}$ and keep the hypothesis $\hat{\mathbf{y}}^{best}$ that achieves the best sentence-level metric with the target $\mathbf{y}$. Then we calculate a corpus-level metric with the greedily-selected hypotheses $\{\hat{\mathbf{y}}^{(i),best}\}_{i=1}^{N}$ and references $\{\mathbf{y}^{(i)}\}_{i=1}^{N}$.

**Pairwise metric (⇓)** Referred as self- (Zhu et al., 2018) or pairwise- (Ott et al., 2018) metric, this measures the within-distribution similarity. This metric computes the average of sentence-level metrics between all pairwise combinations of hypotheses $\{\hat{\mathbf{y}}^1 \dots \hat{\mathbf{y}}^K\}$ generated from each source sequence $\mathbf{x}$. Low pairwise metric indicates high diversity between generated hypotheses.

### 4.5 Human Evaluation Setup

We ask Amazon Mechanical Turkers (AMT) to compare our method with the baselines. For each method, we generate three questions / summaries from 100 passages sampled from SQuAD / CNN-DM test set. For every pair of methods, annotators are instructed to pick a set of questions / summaries that are more *diverse*. To evaluate *accuracy*, they see one question / summary selected

out of 3 questions / summaries with highest log-probability from each method. They are instructed to select a question / summary that is more coherent with the source passage / document. Each annotator is asked to choose either a better method (resulting in "win" or "lose") or "tie" if their quality is indistinguishable. Diversity and accuracy evaluations are conducted separately, and every pair of methods are presented to 10 annotators[1].

### 4.6 Implementation details

For all experiments, we tie the weights (Press and Wolf, 2017) of the encoder embedding, the decoder embedding, and the decoder output layers. This significantly reduces the number of parameters and training time until convergence. We train up to 20 epochs and select the checkpoint with the best oracle metric. We use Adam (Kingma and Ba, 2015) optimizer with learning rate 0.001 and momentum parmeters $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Minibatch size is 64 and 32 for question generation and abstractive summarization. All models are implemented in PyTorch (Paszke et al., 2017) and trained on single Tesla P40 GPU, based on NAVER Smart Machine Learning (NSML) platform (Kim et al., 2018a).

**Question Generation** Following Zhou et al. (2017a), we use 256-dim hidden states for each direction of Bi-GRU encoder, 512-dim hidden states for GRU decoder, 300-dim word embedding initialized from GloVe (Pennington et al., 2014), vocabulary of 20,000 most frequent words, 16-dim embeddings for three linguistic features (POS, NER and word case) respectively.

**Abstractive Summarization** Following See et al. (2017), we use 256-dim hidden states for each direction of Bi-LSTM encoder and LSTM decoder, 128-dim word embedding trained from scratch, and vocabulary of 50,000 most frequent words. Following See et al. (2017), we train our model to generate concatenation of target summaries and split it with periods.

**SELECTOR** The GRU has the same size as the generator encoder, and the dimension of expert embedding $\mathbf{e}_z$ is 300 for NQG++, and 128 for PG. From simple grid search over [0.1, 0.5], we obtain focus binarization threshold $th$ 0.15. The size of focus embedding for the generator is 16.

[1]We conducted 12 human evaluations in total: 2 tasks x 3 baselines x 2 criteria. See Table 3a and 3b.

| Method | BLEU-4 (Top-1 ) | Oracle (Top-K) | Pairwise (Self-sim) |
|---|---|---|---|
| NQG++ | 13.27 | - | - |
| Search-based Methods | | | |
| 3-Beam | 13.590 | 16.848 | 67.277 |
| 5-Beam | 13.526 | 18.809 | 74.674 |
| 3-D. Beam | 13.696 | 16.989 | 68.018 |
| 5-D. Beam | 13.379 | 18.298 | 74.795 |
| 3-T. Sampling | 11.890 | 15.447 | **37.372** |
| 5-T. Sampling | 11.530 | 17.651 | 45.990 |
| Mixture of Experts + Greedy Decoding | | | |
| 3-M. Decoder | 14.720 | 19.324 | 51.360 |
| 5-M. Decoder | 15.166 | 21.965 | 58.727 |
| 3-M. SELECTOR (Ours) | **15.874** | 20.437 | 47.493 |
| 5-M. SELECTOR (Ours) | 15.672 | **22.451** | 59.815 |
| Focus Guide during Test Time | | | |
| 5-Beam + Focus Guide | 24.580 | | - |

Table 1: **Question generation results:** Comparison of diverse generation methods on SQuAD. The score of NQG++ (top row) is from Zhou et al. (2017a), and the rest are from our experiments using NQG++ as a generator. Method prefixes are the numbers of generated questions for each passage-answer pair. Best scores are bolded.

## 5 Results

**Diversity vs. Accuracy Trade-off** Tables 1 and 2 compare our method with different diversity-promoting techniques in question generation and abstractive summarization. The tables show that our mixture SELECTOR method outperforms all baselines in Top-1 and oracle metrics and achieves the best trade-off between diversity and accuracy. Moreover, both mixture models are superior to all search-based methods in the trade-off between diversity and accuracy. Standard and diverse beam search methods score low both in terms of accuracy and diversity. Truncated sampling shows the lowest self-similarity (high diversity), but it achieves the lowest score on Top-1 accuracy. Notably, our method scores state-of-the-art BLEU-4 in question generation on SQuAD and ROUGE comparable to state-of-the-art methods in abstractive summarization in CNN-DM (See also Table 4 for state-of-the-art results in CNN-DM).

**Diversity vs. Number of Mixtures** Here we compare the effect of number of mixtures in our SELECTOR and Mixture Decoder (Shen et al., 2019). Tables 1 and 2 show that pairwise similarity increases (diversity ⇓) when the number of mixtures increases for Mixture Decoder. While we observe a similar trend for SELECTOR in the question generation task, Table 2 shows the opposite

| Method | ROUGE-2 (Top-1) | Oracle (Top-K) | Pairwise (Self-sim) |
|---|---|---|---|
| PG | 17.28 | - | - |
| Search-based Methods | | | |
| 3-Beam | 16.533 | 18.509 | 85.598 |
| 5-Beam | 16.634 | 19.442 | 84.765 |
| 3-D. Beam | 16.667 | 18.722 | 85.496 |
| 5-D. Beam | 16.632 | 19.659 | 84.043 |
| 3-T. Sampling | 12.914 | 17.068 | 17.306 |
| 5-T. Sampling | 13.049 | 19.161 | **16.720** |
| Mixture of Experts + Greedy Decoding | | | |
| 3-M. Decoder | 15.854 | 21.214 | 43.168 |
| 5-M. Decoder | 16.104 | 21.801 | 67.196 |
| 3-M. SELECTOR (Ours) | 17.930 | 21.316 | 51.092 |
| 5-M. SELECTOR (Ours) | **18.309** | **22.511** | 47.280 |
| Focus Guide during Test Time | | | |
| 5-Beam + Focus Guide | 42.757 | | - |

Table 2: **Summarization results:** Comparison of diverse generation methods on CNN-DM. The score of PG (top row) is from See et al. (2017), and the rest are from our experiments using PG as a generator. Method prefixes are the numbers of generated summaries for each document. Best scores are bolded.

| | Diversity (%) | | | Accuracy (%) | | |
|---|---|---|---|---|---|---|
| Baselines | Win | Lose | Tie | Win | Lose | Tie |
| vs. 3-D. Beam | **49.7** | 31.3 | 19.0 | **43.9** | 36.9 | 19.2 |
| vs. 3-T. Sampling | **46.7** | 35.1 | 18.2 | **45.3** | 36.1 | 18.6 |
| vs. 3-M. Decoder | **47.6** | 32.5 | 19.9 | **41.8** | 36.0 | 22.2 |

(a) SQuAD question generation

| | Diversity (%) | | | Accuracy (%) | | |
|---|---|---|---|---|---|---|
| Baselines | Win | Lose | Tie | Win | Lose | Tie |
| vs. 3-D. Beam | **50.4** | 40.9 | 8.7 | **46.2** | 38.5 | 15.3 |
| vs. 3-T. Sampling | **48.7** | 42.0 | 9.3 | **50.3** | 41.2 | 8.5 |
| vs. 3-M. Decoder | **49.7** | 39.6 | 10.7 | **46.5** | 37.5 | 16.0 |

(b) CNN-DM abstractive summarization

Table 3: Human evaluation results

effect in the summarization task i.e., pairwise similarity decreases (diversity ⇑) for SELECTOR.

The abstractive summarization task on CNN-DM has a target distribution with more modalities than question generation task on SQuAD, which is more difficult to model. We speculate that our SELECTOR improves accuracy by focusing on more modes of the output distribution (diversity ⇑), whereas Mixture Decoder tries to improve the accuracy by concentrating on fewer modalities of the output distribution (diversity ⇓).

**Upper Bound Performance** The bottom rows of Tables 1 and 2 show the upper bound performance of SELECTOR by feeding focus guide to generator during test time. In particular, we assume that the mask is the ground truth overlap between input and target sequences at test time. The gap between the oracle metric (top-k accuracy) and the upper bound is very small for question generation. This indicates that the top-k masks for question generation include the ground truth mask. Future work involves improving the content selection stage for the summarization task.

**Human Evaluation** Table 3a and 3b show the human evaluation in two tasks of questions generation and summarization, comparing sequences generated by SELECTOR with the diversity-promoting baselines: Diverse Beam, Truncated Sampling and Mixture Decoder. The table shows that our method significantly outperforms all three baselines in terms of both diversity and accuracy with statistical significance.

**Comparison with State-of-the-art** Table 4 compares the performance of SELECTOR with the state-of-the-art bottom-up content selection of Gehrmann et al. (2018) in abstractive summarization. SELECTOR passes focus embeddings at the decoding step, whereas the bottom-up selection method only uses the masked words for the copy mechanism. We set $K$, the number of mixtures of SELECTOR, to 1 to directly compare it with the previous work (Bottom-Up (Gehrmann et al., 2018)). We observe that SELECTOR not only outperforms Bottom-Up in every metric, but also achieves a new state-of-the-art ROUGE-1 and ROUGE-L on CNN-DM. Moreover, our method scores state-of-the-art BLEU-4 in question generation on SQuAD (Table 1).

| Method | R-1 | R-2 | R-L |
|---|---|---|---|
| PG (See et al., 2017) | 39.53 | 17.28 | 36.38 |
| Bottom-Up (Gehrmann et al., 2018) | 41.22 | 18.68 | 38.34 |
| DCA (Celikyilmaz et al., 2018) | 41.69 | **19.47** | 37.92 |
| SELECTOR & 10-Beam PG (Ours) | **41.72** | 18.74 | **38.79** |

Table 4: Comparison of single-expert selector with state-of-the-art abstractive summarization methods on CNN-DM. R stands for ROUGE (Lin, 2004)

**Efficient Training** Table 5 shows that SELECTOR trains up to 3.7 times faster than mixture decoder (Shen et al., 2019). Training time of mixture decoder linearly increases with the number of decoders, while parallel focus inference of SELECTOR makes additional training time negligible.

**Qualitative Analysis** To analyze how the generator uses the selected content, we visualize attention heatmap of NQG++ in Fig.3 for question generation. The figure shows different attention

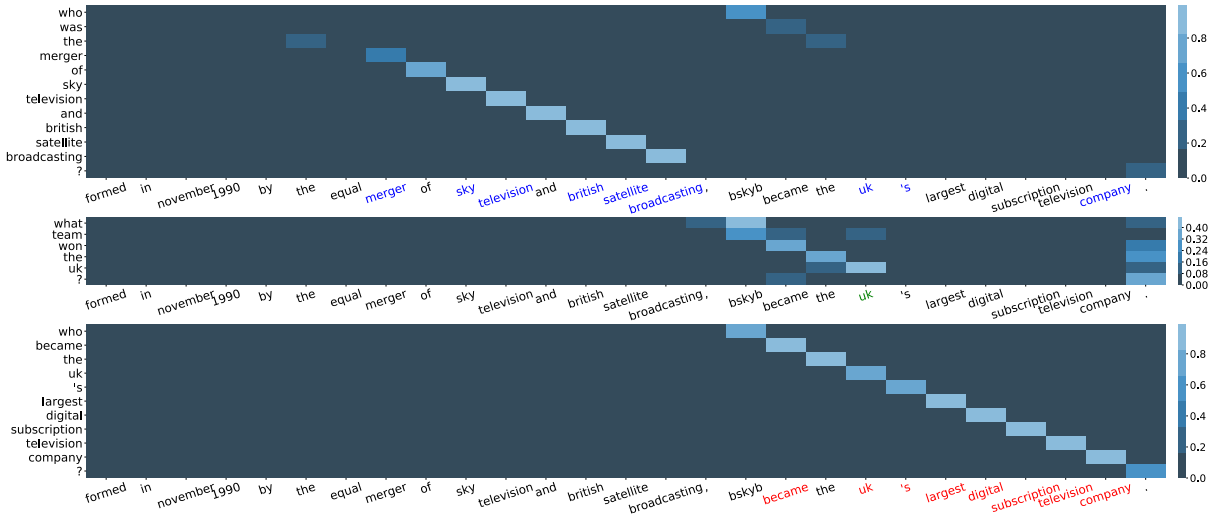


Figure 3: Attention heatmap of NQG++ decoder with three different focus by SELECTOR. This shows focus can guide generator to generate different sequences. Passage tokens are colored when corresponding focus is 1.

| Method | Training time (ms. / step) |
|---|---|
| PG | 641.2 |
| 3-M. Decoder | 1804.1 (× 2.81) |
| 5-M. Decoder | 2367.6 (× 4.37) |
| SELECTOR (Ours) | 692.1 (× 1.08) |
| 3-M. SELECTOR (Ours) | 740.8 (× 1.16) |
| 5-M. SELECTOR (Ours) | 747.6 (× 1.17) |

Table 5: **Training time:** Comparison of training time on CNN-DM. See 4.6 for implementation details.

mechanisms depending on three different focuses inferred by different SELECTOR experts.

## 6 Conclusion

We introduce a novel diverse sequence generation method via proposing a content selection module, SELECTOR. Built upon mixture of experts and hard-EM training, SELECTOR identifies different key parts on source sequence to guide generator to output a diverse set of sequences.

SELECTOR is a generic plug-and-play module that can be added to an existing encoder-decoder model to enforce diversity with a negligible additional computational cost. We empirically demonstrate that our method improves both accuracy and diversity and reduces training time significantly compared to baselines in question generation and abstractive summarization. Future work involves incorporating SELECTOR for other generation tasks such as diverse image captioning.

## Acknowledgment

This research was supported by Allen Distinguished Investigator Award, the Office of Naval Research under the MURI grant N00014-18-1-2670, Samsung GRO, and gifts from Allen Institute for AI and Google. We also thank the members of Clova AI, UW NLP, and the anonymous reviewers for their insightful comments.

## References

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating Sentences from a Continuous Space. In *CoNLL*.

Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep Communicating Agents for Abstractive Summarization. In *NAACL*.

Kyunghyun Cho. 2016. Noisy Parallel Approximate Decoding for Conditional Recurrent Language Model.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *EMNLP*.

Jan Chorowski and Navdeep Jaitly. 2017. Towards better decoding and language model integration in sequence to sequence models. In *INTERSPEECH*.

Yuntian Deng, Yoon Kim, Justin Chiu, Demi Guo, and Alexander M. Rush. 2018. Latent Alignment and Variational Attention. In *NIPS*.

Adji B Dieng, Yoon Kim, Alexander M Rush, and David M Blei. 2018. Avoiding Latent Variable Collapse with Generative Skip Models. In *ICML Workshop*.

David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. 2014. Learning Factored Representations in a Deep Mixture of Experts. In *ICLR Workshop*.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical Neural Story Generation. In *ACL*.

Sebastian Gehrmann, Yuntian Deng, and Alexander M Rush. 2018. Bottom-Up Abstractive Summarization. In *EMNLP*.

Alex Graves. 2013. Generating Sequences With Recurrent Neural Networks.

Xiaodong Gu, Kyunghyun Cho, Jung-woo Ha, and Sunghun Kim. 2019. DialogWAE : Multimodal Response Generation with Conditional Wasserstein Auto-Encoder. In *ICLR*.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the Unknown Words. In *ACL*.

Abner Guzman-Rivera, Dhruv Batra, and Pushmeet Kohli. 2012. Multiple Choice Learning: Learning to Produce Multiple Structured Outputs. In *NIPS*.

Junxian He, Daniel Spokoyny, Graham Neubig, and Taylor Berg-Kirkpatrick. 2019. Lagging Inference Networks and Posterior Collapse in Variational Autoencoders. In *ICLR*.

Xuanli He, Gholamreza Haffari, and Mohammad Norouzi. 2018. Sequence to Sequence Mixture Model for Diverse Machine Translation. In *CoNLL*.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching Machines to Read and Comprehend. In *NIPS*.

Robert Jacobs, Michael Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. 1991. Adaptive Mixtures of Local Experts. *Neural Computation*, 3(1):78–87.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical Reparameterization with Gumbel-Softmax. In *ICLR*.

Ashwin Kalyan, Stefan Lee, Anitha Kannan, and Dhruv Batra. 2018. Learn from Your Neighbor: Learning Multi-modal Mappings from Sparse Annotations. In *ICML*.

Nan Rosemary Ke, Konrad Zolna, Alessandro Sordoni, Zhouhan Lin, Adam Trischler, Yoshua Bengio, Joelle Pineau, Laurent Charlin, and Chris Pal. 2018. Focused Hierarchical RNNs for Conditional Sequence Processing. In *ICML*.

Hanjoo Kim, Minkyu Kim, Dongjoo Seo, Jinwoong Kim, Heungseok Park, Soeun Park, Hyunwoo Jo, KyungHyun Kim, Youngil Yang, Youngkwan Kim, Nako Sung, and Jung-Woo Ha. 2018a. NSML: Meet the MLaaS platform with a real-world case study.

Yoon Kim, Sam Wiseman, Andrew C. Miller, David Sontag, and Alexander M. Rush. 2018b. Semi-Amortized Variational Autoencoders. In *ICML*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

Diederik P Kingma and Max Welling. 2013. Auto-Encoding Variational Bayes. In *NIPS*.

Stefan Lee, Senthil Purushwalkam, Michael Cogswell, Viresh Ranjan, David Crandall, and Dhruv Batra. 2016. Stochastic Multiple Choice Learning for Training Diverse Deep Ensembles. In *NIPS*.

Chenliang Li, Weiran Xu, Si Li, and Sheng Gao. 2018. Guiding Generation for Abstractive Text Summarization based on Key Information Guide Network. In *NAACL*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and William B. Dolan. 2016a. A Diversity-Promoting Objective Function for Neural Conversation Models. In *NAACL*.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016b. A Simple, Fast Diverse Decoding Algorithm for Neural Generation.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *ACL Workshop*.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In *ICLR*.

Sewon Min, Victor Zhong, Richard Socher, and Caiming Xiong. 2018. Efficient and Robust Question Answering from Minimal Context over Documents. In *ACL*.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond. In *CoNLL*.

Radford M. Neal and Geoffrey E. Hinton. 1998. A View of the EM Algorithm that Justifies Incremental, Sparse, and other Variants. In *Learning in Graphical Models*, pages 355–368.

Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2018. Analyzing Uncertainty in Neural Machine Translation. In *ICML*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wj Wei-jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *ACL*.

Yookoon Park, Jaemin Cho, and Gunhee Kim. 2018. A Hierarchical Latent Structure for Variational Conversation Modeling. In *NAACL*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chana, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS Workshop*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global Vectors for Word Representation. In *EMNLP*.

Ofir Press and Lior Wolf. 2017. Using the Output Embedding to Improve Language Models. In *EACL*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *EMNLP*.

Ali Razavi, Aaron van den Oord, Ben Poole, and Oriol Vinyals. 2019. Preventing Posterior Collapse with $\delta$-VAEs. In *ICLR*.

Ehud Reiter and Robert Dale. 2000. *Building Natural Language Generation Systems*.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get To The Point: Summarization with Pointer-Generator Networks. In *ACL*.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. A Hierarchical Latent Variable Encoder-Decoder Model for Generating Dialogues. In *AAAI*.

Shiv Shankar and Sunita Sarawagi. 2019. Posterior Attention Models for Sequence to Sequence Learning. In *ICLR*.

Tianxiao Shen, Myle Ott, Michael Auli, and Marc'Aurelio Ranzato. 2019. Mixture Models for Diverse Machine Translation: Tricks of the Trade. In *ICML*.

Jinsong Su, Shan Wu, Deyi Xiong, Yaojie Lu, Xianpei Han, and Biao Zhang. 2018. Variational Recurrent Neural Machine Translation. In *AAAI*.

Sandeep Subramanian, Tong Wang, Xingdi Yuan, Saizheng Zhang, Yoshua Bengio, and Adam Trischler. 2018. Neural Models for Key Phrase Extraction and Question Generation. In *ACL Workshop*.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasath Ramprasaath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2018. Diverse Beam Search for Improved Description of Complex Scenes. In *AAAI*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer Networks. In *NIPS*.

Tsung-hsien Wen and Minh-thang Luong. 2018. Latent Topic Conversational Models.

Tsung Hsien Wen, Yishu Miao, Phil Blunsom, and Steve Young. 2017. Latent Intention Dialogue Models. In *ICML*.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256.

Jiacheng Xu and Greg Durrett. 2018. Spherical Latent Spaces for Stable Variational Autoencoders. In *EMNLP*.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. 2018. Breaking the Softmax Bottleneck: A High-Rank RNN Language Model. In *ICLR*.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. In *AAAI*.

Biao Zhang, Deyi Xiong, Jinsong Su, Hong Duan, and Min Zhang. 2016. Variational Neural Machine Translation. In *EMNLP*.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning Discourse-level Diversity for Neural Dialog Models using Conditional Variational Autoencoders. In *ACL*.

Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. 2017a. Neural Question Generation from Text: A Preliminary Study. In *NLPCC*.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2017b. Selective Encoding for Abstractive Sentence Summarization. In *ACL*.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A Benchmarking Platform for Text Generation Models. In *SIGIR*.

# An End-to-End Generative Architecture for Paraphrase Generation

**Qian Yang[1]*, Zhouyuan Huo[2], Dinghan Shen[1]**
**Yong Cheng[3], Wenlin Wang[1], Guoyin Wang[1], Lawrence Carin[1]**

[1] Duke University [2] University of Pittsburgh [3] Google AI
qian.yang@duke.edu

## Abstract

Generating high-quality paraphrases is a fundamental yet challenging natural language processing task. Despite the effectiveness of previous work based on generative models, there remain problems with exposure bias in recurrent neural networks, and often a failure to generate realistic sentences. To overcome these challenges, we propose the first end-to-end conditional generative architecture for generating paraphrases via adversarial training, which does not depend on extra linguistic information. Extensive experiments on four public datasets demonstrate the proposed method achieves state-of-the-art results, outperforming previous generative architectures on both automatic metrics (BLEU, METEOR, and TER) and human evaluations.

## 1 Introduction

Paraphrases convey the same meaning as the original sentences or text, but with different expressions in the same language. Paraphrase generation aims to synthesize paraphrases of a given sentence automatically. This is a fundamental natural language processing (NLP) task, and it is important for many downstream applications (Madnani and Dorr, 2010; Yang et al., 2016, 2017; Passonneau et al., 2018). For example, paraphrases can help diversify the response of chatbot engines (Yan et al., 2016), strengthen question answering (Harabagiu and Hickl, 2006; Duboue and Chu-Carroll, 2006; Fader et al., 2014), augment relation extraction (Romano et al., 2006), and extend the coverage of semantic parsers (Berant and Liang, 2014).

Generating accurate and diverse paraphrases automatically is still very challenging. Traditional methods (McKeown, 1983; Bolshakov and Gelbukh, 2004; Zhao et al., 2008) exploit how linguistic knowledge can improve the quality of generated paraphrases, including shallow linguistic features (Zhao et al., 2009), and syntactic and semantic information (Kozlowski et al., 2003; Ellsworth and Janin, 2007). However, they are often domain-specific and hard to scale, or yield inferior results.

With the help of growing large data and neural network models, recent studies have shown promising results. Three families of deep learning architectures have been investigated with the goal of generating high-quality paraphrases. The first is to formulate paraphrase generation as a sequence-to-sequence problem, following experience from machine translation (Bahdanau et al., 2014; Cheng et al., 2016). Prakash et al. (2016) proposes stacked residual long short-term memory networks (LSTMs), while Cao et al. (2017) makes use of the gated recurrent unit (GRU) as the recurrent unit. Shortly afterwards, several works have focused on providing extra information to enhance the encoder-decoder model. Ma et al. (2018) adds distributed word representations, Huang et al. (2018) considers a paraphrased dictionary and

Iyyer et al. (2018) utilizes a syntactic parser. Recently Wang et al. (2018) utilizes a semantic augmented Transformer (Vaswani et al., 2017); however, its accuracy largely depends on extra semantic information. The second family of models employs reinforcement learning (Li et al., 2017b). The third family is the generative architecture that we focus on. Specifically, Gupta et al. (2018) applies the conditional variational autoencoders (CVAEs) (Sohn et al., 2015) with LSTM encoder-decoders in generating paraphrases, expecting RNN's advantage of modeling local dependencies to be a good complement to the CVAE's power at learning global representations. This simple combination, however, has two major challenges.

First, CVAE may fail to generate realistic sen-

* Corresponding author



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3132–3142,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| | |
|---|---|
| original | How do I improve my English speaking ? |
| paraphrased | How do I speak English fluently ? |
| generated | How do I I to ? |
| original | What is the easiest way to lose weight faster? |
| paraphrased | What is the best way to loose weight quickly ? |
| generated | How can I learn lose weight ? |

Table 1: Examples for generating paraphrases by CVAE, from which it fails to generate realistic sentences. Original: source sentence. Paraphrased: target sentence. Generated: the sentence generated by CVAE.

tences. When generating synthetic sentences by decoding random samples from the latent space, most regions of the latent space do not necessarily decode to realistic sentences; the example in Table 1 reflects this challenge. In (Bowman et al., 2016), the authors attempted to utilize RNN-based VAE to generate more diverse sentences. However, this ignores the fundamental problem of the posterior distribution over latent variables not covering the latent space appropriately. Second, when learning, the ground-truth words are used to decode, while during inference, the RNN generates words in sequence from previously *generated* words. Bengio et al. (2015) called this phenomenon "Exposure Bias" and tried to solve it by a scheduled sampling method. However, in practice, it produced unstable results, because it is fundamentally an inconsistent training strategy (Huszár, 2015).

We propose the first paraphrase generative model via adversarial training to tackle the above problems, which we believe is a natural answer to the aforementioned challenges. Specifically, we formalize CVAE as a generator of the Generative Adversarial Network (GAN) (Goodfellow et al., 2014) and tailor a discriminator to CVAE. By introducing an adversarial game between a generator and a discriminator, GAN matches the distribution of synthetic data with that of real data. The generator of GAN seeks to map samples from a given prior distribution to *realistic* synthetic data. The discriminator of GAN compares the entire real and synthetic sentences, instead of individual words, which should in principle alleviates the exposure-bias issue (Yu et al., 2017; Li et al., 2017a; Zhang et al., 2017; Guo et al., 2018). The intuition behind our model can be interpreted as using CVAE to generate similar sentences and enhancing CVAE with an extended discriminator, so that the generator and discriminator work effectively together; they provide feedback signals to each other, resulting in a mutual adversarial framework. Overall, our contributions are as follows.

- To the best of our knowledge, this work represents the first to propose an end-to-end paraphrase generation architecture via adversarial training, which does not require extra linguistic information.
- We take advantage of GAN to help choose a better latent-variable distribution. By this, we not only utilize the better latent variables but also strengthen the expressiveness of the generative model.
- Our experiments show that the proposed model is capable of generating plausible paraphrased sentences and outperforms competitive baseline models, with state-of-the-art results.

## 2 Preliminaries

We first provide preliminaries of variational auto-encoders and conditional variational auto-encoders.

**VAE.** The variational auto-encoder (VAE) (Kingma and Welling, 2013; Sohn et al., 2015) is a directed graphical model with latent variables. The generative process of VAE employs an

- (Encoder) generating a set of latent variable $\boldsymbol{z}$ from the prior distribution $p_\theta(\boldsymbol{z})$, where $\theta$ is the generative parameters;
- (Decoder) then generating the data $\boldsymbol{x}$ from the generative distribution $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ conditioned on $\boldsymbol{z}$.

Although due to intractable posterior inference, parameter estimation of directed graphical models is generally challenging, VAE parameters can be estimated efficiently by a stochastic gradient variational bayes (SGVB) estimator (Kingma and Welling, 2013), and can be optimized straightforwardly using standard stochastic gradient techniques.

SGVB treats the variational lower bound of the log-likelihood as a surrogate objective function,

which can be written as:

$$\begin{aligned}\log p_\theta(\boldsymbol{x}) &= -KL(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \parallel p_\theta(\boldsymbol{z})) \\ &\quad +KL(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \parallel p_\theta(\boldsymbol{z}|\boldsymbol{x})) \\ &\quad +\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] \\ &\geq -KL(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \parallel p_\theta(\boldsymbol{z})) \\ &\quad +\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})], \quad (1)\end{aligned}$$

where the inequality follows because the second term $KL(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \parallel p_\theta(\boldsymbol{z}|\boldsymbol{x}))$ in (1) is nonnegative.

In practice, we can approximate the second term by drawing latent variable samples $\{\boldsymbol{z}^1, \boldsymbol{z}^2, ..., \boldsymbol{z}^L\}$ following $q_\phi(\boldsymbol{z}|\boldsymbol{x})$, where $\phi$ is the variational parameters, and the empirical objective of VAE with Gaussian latent variables can be represented as:

$$\begin{aligned}\tilde{\mathcal{L}}_{VAE}(\boldsymbol{x};\theta,\phi) &= -KL(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \parallel p_\theta(\boldsymbol{z})) \\ &\quad +\frac{1}{L}\sum_{l=1}^{L}\log p_\theta(\boldsymbol{x}|\boldsymbol{z}^l), \quad (2)\end{aligned}$$

where $\boldsymbol{z}^l = g_\phi(\boldsymbol{x}, \epsilon^l)$, and $\epsilon^l \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. That means $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ is reparameterized with a differentiable unbiased function $g_\phi(\boldsymbol{x}, \epsilon^l)$, where $\boldsymbol{x}$ is the data and $\epsilon$ is the noise variable. VAE can be trained efficiently by stochastic gradient descent (SGD) because the reparameterization trick allows error backpropagation through the Gaussian latent variables.

**CVAE.** Sohn et al. (2015) develops a deep conditional generative model (CVAE) for structured output prediction using Gaussian latent variables. Different from the normal VAE, CVAE consists of three variables: input variables $\boldsymbol{x}$, output variables $\boldsymbol{y}$, and latent variables $\boldsymbol{z}$, and its prior distribution is $p_\theta(\boldsymbol{z}|\boldsymbol{x})$. CVAE is trained to maximize the conditional log-likelihood $\log p_\theta(\boldsymbol{y}|\boldsymbol{x})$, and again it is formulated in the framework of SGVB. The variational lower bound of the model is:

$$\begin{aligned}\log p_\theta(\boldsymbol{y}|\boldsymbol{x}) &\geq -KL(q_\phi(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y}) \parallel p_\theta(\boldsymbol{z}|\boldsymbol{x})) \\ &\quad +\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y})}[\log p_\theta(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{z})] \quad (3)\end{aligned}$$

and the empirical lower bound is represented as:

$$\begin{aligned}\tilde{\mathcal{L}}_{CVAE}(\boldsymbol{x},\boldsymbol{y};\theta,\phi) &= -KL(q_\phi(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y}) \parallel p_\theta(\boldsymbol{z}|\boldsymbol{x})) \\ &\quad +\frac{1}{L}\sum_{l=1}^{L}\log p_\theta(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{z}^l), \quad (4)\end{aligned}$$

where $z^l = g_\phi(\boldsymbol{x}, \boldsymbol{y}, \epsilon^l), \epsilon \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, and $L$ is the number of samples.

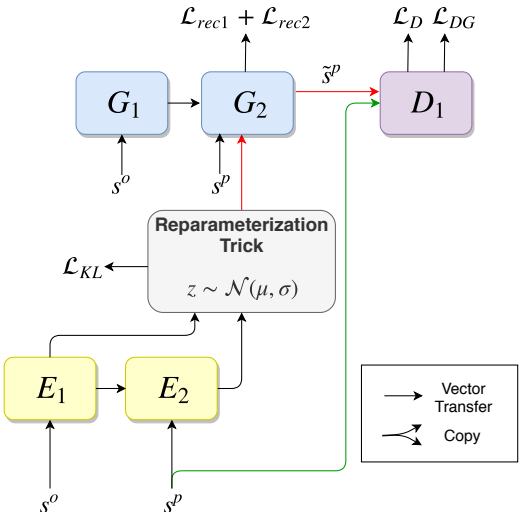


Figure 1: Our generative model. The red arrows denote the process for generating sentences and the green one denotes the transmission of target paraphrased sentences.

## 3 Model

Our new model, which we call GAP for Generative Adversarial Paraphrase model, targets the goal of generating plausible paraphrased sentences conditioned on the original sentences, without any extra linguistic information. We first propose our end-to-end conditional generative architecture for generating paraphrase via adversarial training. Two training techniques are then described for the proposed model.

### 3.1 GAP

We consider a dataset with pairs of the original sentence and the paraphrased sentence $\{(s_k^o, s_k^p)\}_{k=1}^K$. For a given sentence $s$, let $w_t$ denote the $t$-th word. Each word $w_t$ is embedded into a $d$-dimensional vector $\boldsymbol{x_t} = \mathbf{W_e}[w_t]$, where $\mathbf{W_e} \in \mathbb{R}^{d\times V}$ is a learned word embedding matrix, $V$ is the vocabulary size, and the $v$-th column of $\mathbf{W_e}$ is $\mathbf{W_e}[v]$. All columns of $\mathbf{W_e}$ are normalized to have unit $\ell_2$-norm, *i.e.*, $\|\mathbf{W_e}[v]\|_2 = 1, \forall v$, by dividing each column with its $\ell_2$-norm. After embedding, a sentence of length $T$ (padded when necessary) is represented as $\mathbf{X} \in \mathbb{R}^{d\times T}$, by simply concatenating its word embeddings, *i.e.*, $\boldsymbol{x_t}$ is the $t$-th column of $\mathbf{X}$.

The training set is $\mathcal{S} = \{(s_k^o, s_k^p)\}_{k=1}^K$ of pairwise data, where $s_k^o \in \mathcal{S}^o$ denotes the original sentence sequence, and $s_k^p \in \mathcal{S}^p$ denotes the reference paraphrased sentence sequence. The goal of our model is to learn a function $f$: $\mathcal{S}^o \mapsto \mathcal{S}^p$. The

overall model structure of the proposed method is shown in Figure 1. The components can be divided into three modules: encoder in yellow, decoder in blue (which is also the generator), and the discriminator in purple. The connections between them are drawn in arrows.

**Encoder $E$.** The encoder $E$ consists of two LSTM networks $E_1$ and $E_2$, and $E$ generates latent variable $\boldsymbol{z}$ from the input sentences. There are two different paths generating latent variable $\mathbf{z}$. In the first path, we input sampled $s^o$ and $s^p$ into $E_1$ and $E_2$, respectively, and generate sufficient statistics of a Gaussian distribution, mean $\mu$ and standard deviation $\sigma$. Therefore, the distribution of the latent variable $\mathbf{z}$ can be represented as $q(\mathbf{z}|s^o, s^p)$.

However, we cannot know the paraphrased sentence $s^p$ when we evaluate or apply the trained model in real applications. The training framework is not fit for the circumstance at testing time, and only $E_1$ is used during the inference. Thus, we also compute the other set of sufficient statistics, mean $\mu'$ and standard deviation $\sigma'$ by inputting $s^o$ into encoder $E_1$. The sampled latent variable $\mathbf{z}$ is only dependent on $s^o$, and we can represent the distribution of $\mathbf{z}$ as $p(\mathbf{z}|s^o)$.

**Decoder/Generator $G$.** We again combine two LSTM decoders $G_1$ and $G_2$ as the generator $G$. At first, we feed the original sentence $s^o$ into decoder $G_1$ and obtain final state $\boldsymbol{c}_T^{G_1}$, $\boldsymbol{h}_T^{G_1}$. The paraphrased sentence $\tilde{s}^p$ is predicted word by word. At each step, we concatenate the latent variable $\boldsymbol{z}$ and the previously predicted (ground truth) word, and then input it into LSTM decoder $G_2$ with hidden state from the last step. At timestep 0, the input word is BOS ("begin of sentence" symbol), and the hidden state is the output from $G_1$. Therefore, the probability of the predicted paraphrased sentence $\tilde{s}^p$, given the encoded feature vector $\boldsymbol{z}$, is defined as:

$$p(\tilde{s}^p|\boldsymbol{z}, s^o) = \prod_{t=1}^{T} p(\tilde{w}_t^p|\tilde{w}_{<t}^p, \boldsymbol{z}, s^o), \quad (5)$$

where $\tilde{w}_t^p$ is the $t$-th generated token, $< t \triangleq \{0, \cdots, t-1\}$. $w_0^p$ denotes BOS. All the words in $\tilde{s}^p$ are generated sequentially using the LSTM, based on previously generated words, until the end-of-sentence symbol is generated. The $t$-th word $\tilde{w}_t^p$ is generated as $\tilde{w}_t^p = \text{argmax}(\mathbf{V}\boldsymbol{h}_t)$. Note that the hidden units $\boldsymbol{h}_t$ are updated through $\mathcal{E}(\tilde{\boldsymbol{x}}_{t-1}^p, \boldsymbol{h}_{t-1}, \boldsymbol{z})$, where $\mathcal{E}$ denotes the transition function in the second LSTM cell related to the updates. The transition function $\mathcal{E}(\cdot)$ is implemented with an LSTM. $\mathbf{V}$ is a weight matrix used to compute a distribution over words. $\tilde{\boldsymbol{x}}_{t-1}^p$ is the embedding vector of the previously generated word $\tilde{w}_{t-1}^p$, *i.e.*,

$$\tilde{\boldsymbol{x}}_{t-1}^p = \mathbf{W}_{\boldsymbol{e}}[\tilde{w}_{t-1}^p], \quad (6)$$

and is the input for $t$-th step. Consequently, the generated sentence $\tilde{s}^p = [\tilde{w}_1^p, \cdots, \tilde{w}_T^p]$ is obtained given $\boldsymbol{z}$, by simply concatenating the generated words.

**Two-Path Loss.** Thus far, we can compute the loss from two paths of the encoder. In the first path, we minimize the KL loss and MLE loss:

$$\begin{aligned} \min_E \quad & \text{KL}(q(\boldsymbol{z}|s^o, s^p)||p(z|s^o)) \\ & -\mathbb{E}_{q(\boldsymbol{z}|s^o,s^p)}\left[\log p(\tilde{s}^p|\boldsymbol{z}, s^o)\right]. \end{aligned} \quad (7)$$

In the second path, the loss is presented as follows:

$$\min_E \quad -\mathbb{E}_{q(\boldsymbol{z}|s^o)}\left[\log p(\tilde{s}^p|\boldsymbol{z}, s^o)\right]. \quad (8)$$

The proposed two-path reconstruction loss diminishes the gap between prediction pipelines at training and testing, and helps to generate diverse but realistic predictions.

The objective function of optimizing $E$ and $G$ can be written as follows:

$$\begin{aligned} \min_{E,G} \quad & \lambda_1 \text{KL}(q(\boldsymbol{z}|s^o, s^p)||p(z|s^o)) \\ & -\lambda_1 \mathbb{E}_{q(\boldsymbol{z}|s^o,s^p)}\left[\log p(\tilde{s}^p|\boldsymbol{z}, s^o)\right] \\ & -\lambda_2 \mathbb{E}_{q(\boldsymbol{z}|s^o)}\left[\log p(\tilde{s}^p|\boldsymbol{z}, s^o)\right] \\ & -\lambda_3 \mathbb{E}_{\tilde{s}^p \sim p(\tilde{s}^p|\boldsymbol{z}', s^o)} \log(D(\tilde{s}^p)), \end{aligned} \quad (9)$$

where $D$ denotes the discriminator (described below) and $\boldsymbol{z}' \sim q(\boldsymbol{z}|s^o)$. We use $\lambda_1$, $\lambda_2$ and $\lambda_3$ to balance the weights of the different losses.

**Discriminator $D$.** We use one LSTM as the discriminator. The goal of the discriminator is to distinguish real paraphrased sentences from generated sentences. Given the embeddings of true paraphrased sentence $s^p$ and the fake generated sentence $\tilde{s}^p$, the discriminator loss is defined as:

$$\begin{aligned} \min_D \quad & -\mathbb{E}_{\tilde{s}^p \sim p(\tilde{s}^p|\boldsymbol{z}', s^o)} \log(1 - D(\tilde{s}^p)) \\ & -\mathbb{E}_{s^p \sim p(s^p)}[\log D(s^p)], \end{aligned} \quad (10)$$

where $\boldsymbol{z}' \sim q(\boldsymbol{z}|s^o)$.

**Algorithm 1** Training Pipeline

**Initialize:** Gradients of corresponding parameters: Encoder $E$ parameters $g_E = [g_{E1}, g_{E2}]$, Decoder/Generator $G$ parameters $g_G = [g_{G1}, g_{G2}]$, Discriminator $D$ parameters $g_D$;

**while** $G$ has not converged **do**
  Sample a batch of $\{s^o, s^p\}$ from dataset;
  $\boldsymbol{z}_1 = E(s^o, s^p)$;
  $\mathcal{L}_{KL} = KL(q(\boldsymbol{z}_1|s^o, s^p)||p(\boldsymbol{z}|s^o))$;
  $\boldsymbol{z}_2 = E_1(s^o)$;
  Predict $\tilde{s}_1^p$ following: $\tilde{s}_1^p = G(s^o, \boldsymbol{z}_1)$;
  Predict $\tilde{s}_2^p$ following: $\tilde{s}_2^p = G(s^o, \boldsymbol{z}_2)$;
  $\mathcal{L}_{rec1} = -\log p(\tilde{s}_1^p|\boldsymbol{z}_1, s^o)$;
  $\mathcal{L}_{rec2} = -\log p(\tilde{s}_2^p|\boldsymbol{z}_2, s^o)$;
  Input $s^p$ and $\tilde{s}_2^p$ into Discriminator $D$;
  $\mathcal{L}_D = -\log(1 - D(\tilde{s}_2^p)) - \log D(s^p)$;
  $\mathcal{L}_{DG} = -\log(D(\tilde{s}_2^p))$;
  $g_{E_1} = \nabla_{E_1}(\lambda_1\mathcal{L}_{KL} + \lambda_1\mathcal{L}_{rec1} + \lambda_2\mathcal{L}_{rec2} + \lambda_3\mathcal{L}_{DG})$;
  $g_{E_2} = \nabla_{E_2}(\mathcal{L}_{KL} + \mathcal{L}_{rec1})$;
  $g_G = \nabla_G(\lambda_1\mathcal{L}_{rec1} + \lambda_2\mathcal{L}_{rec2} + \lambda_3\mathcal{L}_{DG})$;
  $g_D = \nabla\mathcal{L}_D$;
  Update model parameters using Adam optimizer with gradients $g_{E1}$, $g_{E2}$, $g_G$ and $g_D$.
**end while**

### 3.2 Training Techniques

We summarize the training pipeline in Algorithm 1. At each iteration, a pair of original and paraphrased sentences are fed into two paths for sentence reconstruction. A discriminator is also used to distinguish between real and generated sentences, which is helpful to the exposure bias problem.

**Policy Gradient.** Backpropagation of gradients from the discriminative model to the generative model is difficult for sequence generative models. Following Yu et al. (2017), we use the REINFORCE algorithm (Williams, 1992) to approximate gradients with respect to the generator and encoder. In the experiments, we regard the generator $G$ as the stochastic policy and the output of discriminator $D$ as its reward. In this way, we can propagate the gradients from the discriminator to both the generator models and encoder models.

**Warm-up Training.** It is difficult to train GAN using gradient-based methods (Goodfellow et al., 2014). Previous generative models (Zhang et al., 2017) often pre-train the generator using a supervised model, like an auto-encoder. In this paper, we propose a warm-up technique to train our generative model. Following the notation in Algorithm 1, we suppose a warm-up step $t_{wp}$. In the training process, we gradually increase the weight of $\mathcal{L}_{DG}$ to a fixed value $\lambda_3$. From step $t = 0$ to $t_{wp}$, the update of $\lambda_3^t$ can be represented as

$$\lambda_3^t = \frac{t}{t_{wp}} \cdot \lambda_3. \tag{11}$$

When $t \geq t_{wp}$, we fix $\lambda_3^t = \lambda_3$ as a constant.

We also need to balance the losses between (7) and (8). The second loss (8) represents that during inference, $z$ is generated only depending on $x^o$. We let $\lambda_1 = 1 - \lambda_2^t$ and increase the value of $\lambda_2^t$ gradually until $t_{wp}$. Similarly, $\lambda_2^t$ is updated from $t = 0$ to $t_{wp}$ through:

$$\lambda_2^t = \frac{t}{t_{wp}} \cdot \lambda_2. \tag{12}$$

If $t \geq t_{wp}$, we fix $\lambda_2^t = \lambda_2$ .

## 4 Experiments

We assess the performance of our GAP model and compare it with previous methods. We first describe the datasets we use, then present the details of experimental setup, and finally analyze both quantitative and qualitative results.

### 4.1 Datasets

Following previous work (Prakash et al., 2016), we conduct experiments on the same four standard datasets that are used widely for paraphrase generation. Their content is specified below.

**MSCOCO.** MSCOCO was originally an image-captions dataset, containing over 120K images, with five different captions from five different annotators per image. All the annotations toward one image describe the most prominent object or action in this image, which is suitable for the paraphrase generation task. Specifically, the dataset has two divisions: training dataset ("Train2014", over 82K images) and validation dataset ("Val2014", over 40K images). We follow the operations of previous papers, randomly choosing four captions out of five as two source-reference pairs and limit the length of sentence to be 15 words (removing the words beyond the first 15), so that we can compare our results with published work.

**Quora.** Quora consists of over 400K lines of potential question pairs which are duplicate to each

other if a particular position of this line is annotated with 1. Again, we follow the operations of previous work and filter out those pairs annotated with 1. There are 155K such question pairs in total, among which three sub-datasets are created, *i.e.*, training dataset 50K, 100K, 150K and test dataset 4K. The goal of using three sub-datasets is to show how the size of dataset can affect the results of paraphrase generation.

### 4.2 Experimental Setup

Encoders $E_1$, $E_2$ and generators $G_1$, $G_2$ are constructed using 2-layer LSTMs (Hochreiter and Schmidhuber, 1997). For discriminator $D$, it is also 2-layer LSTM. At the beginning, we map each word to a 300-dimensional feature vector and it is initialized with 300-dimensional GloVe vectors (Pennington et al., 2014). Therefore, a sentence of length $N$ can be represented by a matrix of size $N \times 300$. Before inputting the embedding vector into LSTM models, we pre-process these vectors using a two-layer highway network (Srivastava et al., 2015). We set the dimension of latent variables $\boldsymbol{z}$ to be 300. To balance these losses, we set $\lambda_2 = 0.5$ and $\lambda_3 = 0.01$. The warm-up step is $1 \times 10^4$ in all experiments.

We use Adam optimizer (Kingma and Ba, 2014) with learning rate $1 \times 10^{-4}$ and other parameters as default, for all models. Following (Sutskever et al., 2014), we clip gradients of the encoder and generator parameters if the norm of the parameter vector exceeds 10, and we clip the gradients of the discriminator if the norm of the parameter vector exceeds 5. Greedy search is employed to generate paraphrases in the experiment.

### 4.3 Automatic Evaluation

We follow previous papers to choose the same three well-known automatic evaluation metrics: BLEU (Papineni et al., 2002), METEOR (Lavie and Agarwal, 2007), and TER (Snover et al., 2006). As studied by (Wubben et al., 2010), human judgments on generated paraphrases correlate well with these metrics. We then compare our results on these metrics with previous baselines below.

**Baselines.** Our GAP model contains LSTM and VAE components, so we compare it with $(i)$ the basic attentive sequence-to-sequence model from Machine Translation (seq2seq) (Bahdanau et al., 2014), $(ii)$ the stacked residual LSTM model doing paraphrase generation (Residual LSTM) (Prakash et al., 2016), and $(iii)$ a VAE model based on LSTM together with its variants (VAE-S, VAE-SVG, VAE-SVG-eq) (Gupta et al., 2018). Beyond these, we also evaluate other baselines including Transformer (Vaswani et al., 2017), semantic augmented Transformer which requires extra linguistic information (Transformer-PB) (Wang et al., 2018), and a deep reinforcement learning approach (RbM-SL) (Li et al., 2017b).

| Model | Measure | MSCOCO | | |
|---|---|---|---|---|
| | | BLEU↑ | METEOR↑ | TER↓ |
| Seq2Seq + Att | - | 18.60 | 16.80 | 63.00 |
| Residual LSTM | - | 37.00 | 27.00 | 51.60 |
| Transformer | - | 41.00 | 32.80 | 40.50 |
| Transformer-PB | - | 44.00 | 34.70 | 37.10 |
| VAE-SVG | best-B | 41.70 | 30.80 | 41.70 |
| VAE-SVG | best-M | 41.30 | 31.00 | 41.60 |
| VAE-SVG | best-T | 41.30 | 30.90 | 40.80 |
| VAE∗ | avg | 42.53 | 32.77 | 41.38 |
| Ours | best-B | **45.60** | 35.72 | 39.47 |
| Ours | best-M | 41.32 | **36.17** | 40.52 |
| Ours | best-T | 42.46 | 34.79 | **36.83** |

Table 2: Test accuracy on MSCOCO dataset, in percentage. VAE∗ is our implementation of VAE. "best-B", "best-M" and "best-T" represent the scores with the best BLEU, METEOR and TER respectively. "avg" denotes an average over "best-B", "best-M", and "best-T". All other results are directly cited from the respective papers. For different model variants exist in one paper, we only show the one with highest scores here. For the arrows ↑ of BLEU and TER, a higher score is better. ↓ of TER represents a lower score is better. The best results are in boldface.

**Results and Analysis.** The results on MSCOCO are shown in Table 2. We find that our GAP model outperforms the baseline models on all metrics. For example, comparing our scores on MSCOCO dataset when we achieve the best BLEU with the ones from VAE-SVG, we improve the results about 4 BLEU, 5 METEOR, and 4 TER, respectively.

Table 3 shows the performance on Quora dataset. We notice that our accuracy increases with the increasing size of data. Meanwhile, our model is more robust for a relatively smaller dataset such as Quora-50K, leveraging its advantage in learning with fewer data. For example, we achieve much better results than VAE and its variants on the smaller Quora-50k.

**Ablation Study.** VAE∗ in Table 3 extends the pre-

| Model | Measure | Quora-50K | | | Quora-100K | | | Quora-150K | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | BLEU↑ | METEOR↑ | TER↓ | BLEU↑ | METEOR↑ | TER↓ | BLEU↑ | METEOR↑ | TER↓ |
| Seq2Seq + Att | - | 26.06 | 20.35 | - | 36.55 | 26.28 | - | - | - | - |
| Residual LSTM | - | 27.32 | 22.37 | - | 37.38 | 28.17 | - | - | - | - |
| RbM-SL | - | 35.81 | **28.12** | - | 43.54 | **32.84** | - | - | - | - |
| VAE-S | best-B | 15.80 | 20.10 | 69.40 | 17.50 | 21.60 | 67.10 | 19.80 | 22.60 | 63.90 |
| VAE-S | best-M | 15.60 | 21.10 | 71.50 | 17.50 | 22.70 | 69.50 | 19.70 | 23.80 | 66.90 |
| VAE-SVG | best-B | 17.10 | 21.30 | 63.10 | 22.50 | 24.60 | 55.70 | 30.30 | 28.50 | 47.30 |
| VAE-SVG | best-M | 17.10 | 22.20 | 63.80 | 22.40 | 25.50 | 55.60 | 30.30 | 29.20 | 47.10 |
| VAE-SVG-eq | best-B | 17.40 | 21.40 | 61.90 | 22.90 | 24.70 | 55.00 | 31.40 | 29.00 | 46.80 |
| VAE-SVG-eq | best-M | 17.30 | 22.20 | 62.60 | 22.90 | 25.50 | 54.90 | 32.00 | 30.00 | 46.10 |
| VAE∗ | avg | 33.54 | 22.36 | 60.45 | 41.33 | 28.46 | 58.24 | 43.31 | 28.25 | 42.36 |
| Ours | best-B | **38.23** | 22.51 | 56.66 | **45.05** | 31.49 | 55.54 | **47.06** | 30.94 | 41.32 |
| Ours | best-M | 37.14 | **24.09** | 61.43 | 45.05 | **31.49** | 55.54 | 45.51 | **34.31** | 40.49 |
| Ours | best-T | 36.19 | 22.22 | **50.50** | 44.41 | 27.73 | **46.41** | 42.30 | 29.60 | **39.47** |

Table 3: Test accuracy on Quora dataset, in percentage. VAE∗ is our implementation of VAE. "best-B", "best-M" and "best-T" represent the scores with the best BLEU, METEOR and TER respectively. "avg" denotes an average over "best-B", "best-M", and "best-T". All other results are directly cited from the respective papers, and "-" means no such results reported. For the arrows ↑ of BLEU and TER, a higher score is better. ↓ of TER represents a lower score is better. The best results are in boldface.

vious VAE-SVG in (Gupta et al., 2018) by using our proposed two-path loss. We observe that our VAE∗ usually outperforms previous VAE-based models. It demonstrates the proposed two-path reconstruction loss can improve the quality of generated paraphrases. Moreover, our proposed method via adversary training, denoted as "Ours", also has superior performance than VAE∗. Therefore, our proposed adversarial training and two-path loss take an apparently positive effect on alleviating the exposure bias problem and generating diverse but realistic predictions.

### 4.4 Human Evaluation

**Data Preparation.** The accurate evaluation of paraphrases is an open problem. We believe that automatic evaluation is not enough for evaluating paraphrases from a fine-grained perspective, in terms of three attributes: grammar correctness (plausibility), equivalence to the original sentence (equivalence), diversity expression (diversity). For both MSCOCO and Quora datasets, we randomly sample 100 sentence pairs (original sentence, paraphrased sentence) from the test corpus, and apply the generative architectures, including both the VAE∗ model and our GAP model, to generating paraphrases. Thus, we obtain three different sentence pairs: (original sentence, "generated" sentence) by the reference, (original sentence, generated paraphrase) by VAE∗, and by our GAP model. To make the analysis fair, we randomly shuffled all of them. We then partitioned them into ten buckets.

**Process.** We set up an Amazon Mechanical Turk experiment; ten human judges are asked to evaluate the quality of paraphrases. We hope our judges play similar roles to the discriminator in our model, to make a true/fake judgment. It is easier for them to make a binary choice than score how good the paraphrased sentences are from a wide score range. These ten human judges are finally involved in evaluating the total 600 sentence pairs. Each pair is judged by two different judges, and the average score is the final judgment. The agreement between judges is moderate (kappa=0.42). These ten human judges confirmed that they were proficient in English and they had understood the goals of the annotation process very well. They were trained by means of instructions and examples. If the meaning of a generated sentence contains a grammatical error or does not express the same meaning as the original sentence, or lacks diversity, we asked the annotators to score 0 for the corresponding attributes, and 1 otherwise.

**Results and Analysis.** We report the results in Table 4. Our model generates paraphrases with higher scores in terms of plausibility, equivalence, diversity than the previous VAE model, and their differences are statistically significant (paired t-test, p-value $< 0.01$). The failure cases we observed include implausible sentences, inequiva-

| Model | MSCOCO | | | Quora | | |
|---|---|---|---|---|---|---|
| | Plausibility | Equivalence | Diversity | Plausibility | Equivalence | Diversity |
| Reference | 0.79 | 0.68 | 0.59 | 0.87 | 0.75 | 0.66 |
| VAE∗ | 0.30 | 0.26 | 0.22 | 0.35 | 0.31 | 0.26 |
| Ours | 0.43 | 0.38 | 0.35 | 0.46 | 0.45 | 0.37 |

Table 4: Human judgments for paraphrase generation on different models.

| | |
|---|---|
| original (MSCOCO) | a group of kids are eating pizza and drinking soda |
| paraphrased generated-VAE | a person sits at a table eating pizza |
| generated-VAE | a man sitting at a table with a pizza |
| generated-Ours | a man sitting at a table eating **a plate of** pizza and soda |
| original (Quora-50K) | What is the **fastest** possible way to lose weight? |
| paraphrased generated-VAE | What are some ways to lose weight **fast**? |
| generated-VAE | How can I lose weight weight? |
| generate-Ours | How can I lose weight **in a month**? |
| original (Quora-100K) | What were the **immediate** and most important causes that led to World War 1? |
| paraphrased generated-VAE | What were the causes of World War I? |
| generated-VAE | What were the main causes of World War I? |
| generated-Ours | What were the **direct** and main causes of World War 1? |
| original (Quora-150K) | What could be the reason behind Arnab Goswami **quitting** Times Now? |
| paraphrased generated-VAE | Is Arnab Goswami **quitting** from Times now? |
| generated-VAE | Why did Arnab Goswami quit? |
| generated-Ours | Why did Arnab Goswami **resign** from Times Now? |

Table 5: Samples of generated paraphrases from MSCOCO and Quora. Note that for the last two examples, our generated result is even better than the ground truth where our model paraphrasing "immediate" using "direct", and paraphrasing "quit" using "resign".

lent, and inaccurate expressions. It is believed that the reason for this is related to the input training data. It contains noise caused by the length limitation of $\leq$ 15 words. But note that even for the reference paraphrase, the accuracy cannot be as high as 100%. Here is an example from the real data: for the original sentence "children are playing soccer on a field with several adults observing nearby", the reference paraphrase is "soccer player hits another player in the face". Considering this, the results show that our GAP generates relatively more plausible and diverse paraphrase sentence, compared to the baseline model.

**Case Study.** In Table 5, we show examples sampled from both MSCOCO and Quora. We observe that our model generates paraphrases with higher diversity than VAE, with no loss of information. What's more, we discover some results where our model is even better than the ground truth, which are shown in the last two examples in the Table 5.

## 5 Related Work

Generative models or conditional generative models have experienced remarkable progress in the visual domain, such as VAEs in (Kingma et al., 2014; Sohn et al., 2015) and InfoGAN (Chen et al., 2016). Recent work (Larsen et al., 2015; Chen et al., 2017) also considers combining autoencoders or variational autoencoders with GAN to demonstrate superior performance on image generation. In the area of NLP, generation can be summarized from two perspectives: *text generation* with the goal of yielding diverse and plausible sentences given an original sentence, and *conditional text generation* aiming to generate new sentences conditioned on an original sentence. Considering the latter, examples include generating a new sentence similar to an original sentence (paraphrase generation), in a different style from the original sentence (style transfer), or dependent on dialogue history (dialogue generation). Attempts at using VAEs (Bowman et al., 2016; Wang et al., 2019), GANs (Yu et al., 2017; Zhang et al., 2016), and both (Hu et al., 2017) have been made to address generic text generation. However, all of them are not suitable for conditional text generation. Recent work in (Gupta et al., 2018) tries to handle paraphrase generation using VAEs. However, it suffers from challenges common to VAEs. Our method addresses these challenges with the help of GAN. To the best of our knowledge, this is the first work on using GAN with VAEs for paraphrase generation task.

## 6 Conclusions

We propose the first deep conditional generative architecture for generating paraphrases via adversarial training, in the hope of combining advantages of CVAE to generate similar distributions with the advantage of GAN to generate plausible

sentences. Experimental results evaluated on automatic metrics demonstrate the advantages of our model, with human evaluations also verifying effectiveness. In future work, we intend to accelerate the training of our encoders and decoders with the techniques in (Huo et al., 2018) and apply our architecture and training techniques to other NLP tasks. Overall, we believe that our research makes an important step for using generative models in NLP, especially in conditional text generation.

## Acknowledgments

This research was supported in part by DARPA, DOE, NIH, ONR and NSF.

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1415–1425.

Igor A Bolshakov and Alexander Gelbukh. 2004. Synonymous paraphrasing using wordnet and internet. In *International Conference on Application of Natural Language to Information Systems*, pages 312–323. Springer.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. *CoNLL 2016*, page 10.

Ziqiang Cao, Chuwei Luo, Wenjie Li, and Sujian Li. 2017. Joint copying and restricted generation for paraphrase. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Liqun Chen, Shuyang Dai, Yunchen Pu, Chunyuan Li, Qinliang Su, and Lawrence Carin. 2017. Symmetric variational autoencoder and connections to adversarial learning. *arXiv preprint arXiv:1709.01846*.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, pages 2172–2180.

Yong Cheng, Yang Liu, Qian Yang, Maosong Sun, and Wei Xu. 2016. Neural machine translation with pivot languages. *arXiv preprint arXiv:1611.04928*.

Pablo Duboue and Jennifer Chu-Carroll. 2006. Answering the question you wish they had asked: The impact of paraphrasing for question answering. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*.

Michael Ellsworth and Adam Janin. 2007. Mutaphrase: Paraphrasing with framenet. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 143–150. Association for Computational Linguistics.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1156–1165. ACM.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. 2018. Long text generation via adversarial training with leaked information. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Ankush Gupta, Arvind Agarwal, Prawaan Singh, and Piyush Rai. 2018. A deep generative framework for paraphrase generation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Sanda Harabagiu and Andrew Hickl. 2006. Methods for using textual entailment in open-domain question answering. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 905–912. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1587–1596. JMLR. org.

Shaohan Huang, Yu Wu, Furu Wei, and Ming Zhou. 2018. Dictionary-guided editing networks for paraphrase generation. *arXiv preprint arXiv:1806.08077*.

Zhouyuan Huo, Bin Gu, Heng Huang, et al. 2018. Decoupled parallel backpropagation with convergence guarantee. In *International Conference on Machine Learning*, pages 2103–2111.

Ferenc Huszár. 2015. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? *stat*, 1050:16.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. *arXiv preprint arXiv:1804.06059*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. 2014. Semi-supervised learning with deep generative models. In *Advances in neural information processing systems*, pages 3581–3589.

Raymond Kozlowski, Kathleen F McCoy, and K Vijay-Shanker. 2003. Generation of single-sentence paraphrases from predicate/argument structure using lexico-grammatical resources. In *Proceedings of the second international workshop on Paraphrasing-Volume 16*, pages 1–8. Association for Computational Linguistics.

Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. 2015. Autoencoding beyond pixels using a learned similarity metric. *arXiv preprint arXiv:1512.09300*.

Alon Lavie and Abhaya Agarwal. 2007. Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 228–231. Association for Computational Linguistics.

Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. 2017a. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547*.

Zichao Li, Xin Jiang, Lifeng Shang, and Hang Li. 2017b. Paraphrase generation with deep reinforcement learning. *arXiv preprint arXiv:1711.00279*.

Shuming Ma, Xu Sun, Wei Li, Sujian Li, Wenjie Li, and Xuancheng Ren. 2018. Query and output: Generating words by querying distributed word representations for paraphrase generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 196–206.

Nitin Madnani and Bonnie J Dorr. 2010. Generating phrasal and sentential paraphrases: A survey of data-driven methods. *Computational Linguistics*, 36(3):341–387.

Kathleen R McKeown. 1983. Paraphrasing questions using given and new information. *Computational Linguistics*, 9(1):1–10.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Rebecca J Passonneau, Ananya Poddar, Gaurav Gite, Alisa Krivokapic, Qian Yang, and Dolores Perin. 2018. Wise crowd content assessment and educational rubrics. *International Journal of Artificial Intelligence in Education*, 28(1):29–55.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Aaditya Prakash, Sadid A Hasan, Kathy Lee, Vivek Datla, Ashequl Qadir, Joey Liu, and Oladimeji Farri. 2016. Neural paraphrase generation with stacked residual lstm networks. *arXiv preprint arXiv:1610.03098*.

Lorenza Romano, Milen Kouylekov, Idan Szpektor, Ido Dagan, and Alberto Lavelli. 2006. Investigating a generic paraphrase-based approach for relation extraction. In *11th Conference of the European Chapter of the Association for Computational Linguistics*.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of association for machine translation in the Americas*, volume 200.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. 2015. Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems*, pages 3483–3491.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *arXiv preprint arXiv:1505.00387*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Su Wang, Rahul Gupta, Nancy Chang, and Jason Baldridge. 2018. A task in a suit and a tie: paraphrase generation with semantic augmentation. *arXiv preprint arXiv:1811.00119*.

Wenlin Wang, Zhe Gan, Hongteng Xu, Ruiyi Zhang, Guoyin Wang, Dinghan Shen, Changyou Chen, and Lawrence Carin. 2019. Topic-guided variational autoencoders for text generation. *arXiv preprint arXiv:1903.07137*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Sander Wubben, Antal Van Den Bosch, and Emiel Krahmer. 2010. Paraphrase generation as monolingual translation: Data and evaluation. In *Proceedings of the 6th International Natural Language Generation Conference*, pages 203–207. Association for Computational Linguistics.

Zhao Yan, Nan Duan, Junwei Bao, Peng Chen, Ming Zhou, Zhoujun Li, and Jianshe Zhou. 2016. Docchat: An information retrieval approach for chatbot engines using unstructured documents. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 516–525.

Qian Yang, Gerard de Melo, Yong Cheng, and Sen Wang. 2017. Hitext: text reading with dynamic salience marking. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 311–319. International World Wide Web Conferences Steering Committee.

Qian Yang, Rebecca J Passonneau, and Gerard De Melo. 2016. Peak: Pyramid evaluation via automated knowledge extraction. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Yizhe Zhang, Zhe Gan, and Lawrence Carin. 2016. Generating text via adversarial training. In *NIPS workshop on Adversarial Training*, volume 21.

Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. 2017. Adversarial feature matching for text generation. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 4006–4015. JMLR. org.

Shiqi Zhao, Xiang Lan, Ting Liu, and Sheng Li. 2009. Application-driven statistical paraphrase generation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 834–842. Association for Computational Linguistics.

Shiqi Zhao, Cheng Niu, Ming Zhou, Ting Liu, and Sheng Li. 2008. Combining multiple resources to improve smt-based paraphrasing model. *Proceedings of ACL-08: HLT*, pages 1021–1029.

# Table-to-Text Generation with Effective Hierarchical Encoder on Three Dimensions (Row, Column and Time)

**Heng Gong, Xiaocheng Feng, Bing Qin*, Ting Liu**
Harbin Institute of Technology, China
{hgong, xcfeng, qinb, tliu}@ir.hit.edu.cn

## Abstract

Although Seq2Seq models for table-to-text generation have achieved remarkable progress, modeling table representation in one dimension is inadequate. This is because (1) the table consists of multiple rows and columns, which means that encoding a table should not depend only on one dimensional sequence or set of records and (2) most of the tables are time series data (e.g. NBA game data, stock market data), which means that the description of the current table may be affected by its historical data. To address aforementioned problems, not only do we model each table cell considering other records in the same row, we also enrich table's representation by modeling each table cell in context of other cells in the same column or with historical (time dimension) data respectively. In addition, we develop a table cell fusion gate to combine representations from row, column and time dimension into one dense vector according to the saliency of each dimension's representation. We evaluated our methods on ROTOWIRE, a benchmark dataset of NBA basketball games. Both automatic and human evaluation results demonstrate the effectiveness of our model with improvement of 2.66 in BLEU over the strong baseline and outperformance of state-of-the-art model.

## 1 Introduction

Table-to-text generation is an important and challenging task in natural language processing, which aims to produce the summarization of numerical table (Reiter and Dale, 2000; Gkatzia, 2016). The related methods can be empirically divided into two categories, pipeline model and end-to-end model. The former consists of content selection, document planning and realisation, mainly for early industrial applications, such as weather forecasting and medical monitoring, etc. The latter generates text directly from the table through a standard neural encoder-decoder framework to avoid error propagation and has achieved remarkable progress. In this paper, we particularly focus on exploring how to improve the performance of neural methods on table-to-text generation.

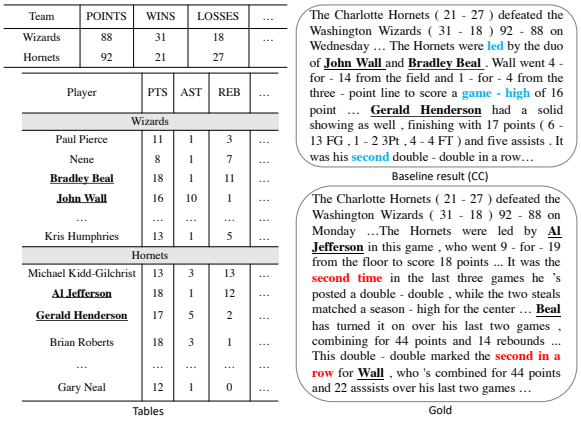


Figure 1: Generated example on ROTOWIRE by using Conditional Copy (CC) as baseline (Wiseman et al., 2017). Text that accurately reflects records in the table is in red, and text that contradicts the records is in blue.

Recently, ROTOWIRE, which provides tables of NBA players' and teams' statistics with a descriptive summary, has drawn increasing attention from academic community. Figure 1 shows an example of parts of a game's statistics and its corresponding computer generated summary. We can see that the tables has a formal structure including table row header, table column header and table cells. "Al Jefferson" is a table row header that represents a player, "PTS" is a table column header indicating the column contains player's score and "18" is the value of the table cell, that is, Al Jefferson scored 18 points. Several related models have been proposed . They typically encode the table's records separately or as a long sequence and generate a long descriptive summary

* Email corresponding.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3143–3152,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

by a standard Seq2Seq decoder with some modifications. Wiseman et al. (2017) explored two types of copy mechanism and found conditional copy model (Gulcehre et al., 2016) perform better . Puduppully et al. (2019) enhanced content selection ability by explicitly selecting and planning relevant records. Li and Wan (2018) improved the precision of describing data records in the generated texts by generating a template at first and filling in slots via copy mechanism. Nie et al. (2018) utilized results from pre-executed operations to improve the fidelity of generated texts. However, we claim that their encoding of tables as sets of records or a long sequence is not suitable. Because (1) the table consists of multiple players and different types of information as shown in Figure 1. The earlier encoding approaches only considered the table as sets of records or one dimensional sequence, which would lose the information of other (column) dimension. (2) the table cell consists of time-series data which change over time. That is to say, sometimes historical data can help the model select content. Moreover, when a human writes a basketball report, he will not only focus on the players' outstanding performance in the current match, but also summarize players' performance in recent matches. Lets take Figure 1 again. Not only do the gold texts mention Al Jefferson's great performance in this match, it also states that "It was the second time in the last three games he's posted a double-double". Also gold texts summarize John Wall's "double-double" performance in the similar way. Summarizing a player's performance in recent matches requires the modeling of table cell with respect to its historical data (time dimension) which is absent in baseline model. Although baseline model Conditional Copy (CC) tries to summarize it for Gerald Henderson, it clearly produce wrong statements since he didn't get "double-double" in this match.

To address the aforementioned problems, we present a hierarchical encoder to simultaneously model row, column and time dimension information. In detail, our model is divided into three layers. The first layer is used to learn the representation of the table cell. Specifically, we employ three self-attention models to obtain three representations of the table cell in its row, column and time dimension. Then, in the second layer, we design a record fusion gate to identify the more important representation from those three dimension and combine them into a dense vector. In the third layer, we use mean pooling method to merge the previously obtained table cell representations in the same row into the representation of the table's row. Then, we use self-attention with content selection gate (Puduppully et al., 2019) to filter unimportant rows' information. To the best of our knowledge, this is the first work on neural table-to-text generation via modeling column and time dimension information so far. We conducted experiments on ROTOWIRE. Results show that our model outperforms existing systems, improving baseline BLEU from 14.19 to 16.85 (+18.75%), P% of relation generation (RG) from 74.80 to 91.46 (+22.27%), F1% of content selection (CS) from 32.49 to 41.21 (+26.84%) and content ordering (CO) from 15.42 to 20.86 (+35.28%) on test set. It also exceeds the state-of-the-art model in terms of those metrics.

## 2 Preliminaries

### 2.1 Notations

The input to the model are tables $S = \{s^1, s^2, s^3\}$. $s^1$, $s^2$, and $s^3$ contain records about players' performance in home team, players' performance in visiting team and team's overall performance respectively. We regard each cell in the table as record. Each record $r$ consists of four types of information including value $r.v$ (e.g. 18), entity $r.e$ (e.g. Al Jefferson), type $r.c$ (e.g. POINTS) and a feature $r.f$ (e.g. visiting) which indicate whether a player or a team compete in home court or not. Each player or team takes one row in the table and each column contains a type of record such as points, assists, etc. Also, tables contain the date when the match happened and we let $k$ denote the date of the record. We also create timelines for records. The details of timeline construction is described in Section 2.2. For simplicity, we omit table id $l$ and record date $k$ in the following sections and let $r_{i,j}$ denotes a record of $i^{th}$ row and $j^{th}$ column in the table. We assume the records come from the same table and $k$ is the date of the mentioned record. Given those information, the model is expected to generate text $y = (y_1, ..., y_t, ..., y_T)$ describing these tables. $T$ denotes the length of the text.

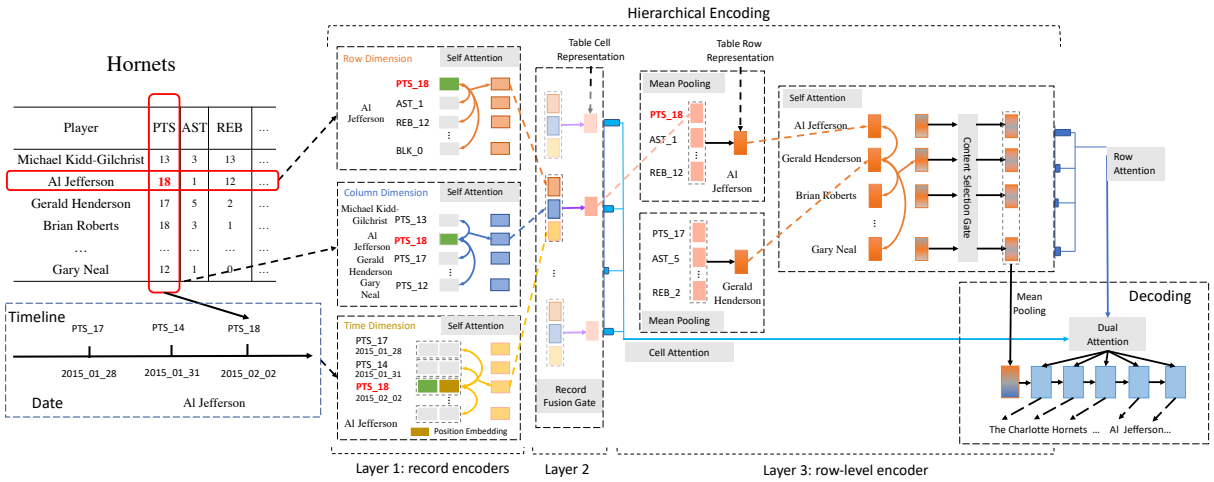


Figure 2: The architecture of our proposed model.

### 2.2 Record Timeline Constrcution

In this paper, we construct timelines $tl = \{tl_{e,c}\}_{e=1,c=1}^{E,C}$ for records. $E$ denotes the number of distinct record entities and $C$ denotes the number of record types. For each timeline $tl_{e,c}$, we first extract records with the same entity $e$ and type $c$ from dataset. Then we sort them into a sequence according to the record's date from old to new. This sequence is considered as timeline $tl_{e,c}$. For example, in Figure 2, the "Timeline" part in the lower-left corner represents a timeline for entity Al Jefferson and type PTS (points).

### 2.3 Baseline Model

We use Seq2Seq model with attention (Luong et al., 2015) and conditional copy (Gulcehre et al., 2016) as the base model. During training, given tables $S$ and their corresponding reference texts $y$, the model maximized the conditional probability $P(y|S) = \prod_{t=1}^{T} P(y_t|y_{<t}, S)$ . $t$ is the timestep of decoder. First, for each record of the $i^{th}$ row and $j^{th}$ column in the table, we utilize 1-layer MLP to encode the embeddings of each record's four types of information into a dense vector $\boldsymbol{r_{i,j}}$, $\boldsymbol{r_{i,j}} = ReLU(\boldsymbol{W_a}[\boldsymbol{r_{i,j}.e}; \boldsymbol{r_{i,j}.c}; \boldsymbol{r_{i,j}.v}; \boldsymbol{r_{i,j}.f}] + \boldsymbol{b_a})$. $\boldsymbol{W_a}$ and $\boldsymbol{b_a}$ are trainable parameters. The word embeddings for each type of information are trainable and randomly initialized before training following Wiseman et al. (2017). $[;]$ denotes the vector concatenation. Then, we use a LSTM decoder with attention and conditional copy to model the conditional probability $P(y_t|y_{<t}, S)$. The base model first use attention mechanism (Luong et al., 2015) to find relevant records from the input tables and represent them as context vector. Please note that the base model doesn't utilize the structure of three tables and normalize the attention weight $\alpha_{t,i',j'}$ across every records in every tables. Then it combines the context vector with decoder's hidden state $\boldsymbol{d_t}$ and form a new attentional hidden state $\tilde{\boldsymbol{d_t}}$ which is used to generate words from vocabulary $P_{gen}(y_t|y_{<t}, S) = softmax(\boldsymbol{W_d}\tilde{\boldsymbol{d_t}} + \boldsymbol{b_d})$ Also the conditional copy mechanism is adopted in base model. It introduces a variable $z_t$ to decide whether to copy from tables or generate from vocabulary. The probability to copy from table is $P(z_t = 1|y_{<t}, S) = sigmoid(\boldsymbol{w_e} \cdot \boldsymbol{d_t} + b_e)$. Then it decomposes the conditional probability of generating $t^{th}$ word $P(y_t|y_{<t}, S)$, given the tables $S$ and previously generated words $y_{<t}$, as follows.

$$P(y_t, z_t|y_{<t}, S) = \begin{cases} P(z_t = 1|y_{<t}, S) \sum_{y_t \leftarrow \boldsymbol{r_{i',j'}}} \alpha_{t,i',j'} & z_t = 1 \\ P(z_t = 0|y_{<t}, S) P_{gen}(y_t|y_{<t}, S) & z_t = 0 \end{cases} \quad (1)$$

## 3 Approach

In this section, we propose an effective hierarchical encoder to utilize three dimensional structure of input data in order to improve table representation. Those three dimensions include row, column and time. As shown in Figure 2, during encoding, our model consists of three layers including record encoders, record fusion gate and row-level encoder. Given tables $S$ as described in Section 2.1, we first encode each record in each dimension respectively. Then we use the record fusion gate to combine them into a dense representation. Afterwards, we obtain row-level representation via mean pooling and self-attention with content selection gate. In decoding phase, the decoder

can first find important row then attend to important record when generating texts. We describe model's details in following parts.

### 3.1 Layer 1: Record Encoders

#### 3.1.1 Row Dimension Encoder

Based on our observation, when someone's points is mentioned in texts, some related records such as "field goals made" (FGM) and "field goals attempted" (FGA) will also be included in texts. Taken gold texts in Figure 1 as example, when Al Jefferson's point 18 is mentioned, his FGM 9 and FGA 19 are also mentioned. Thus, when modeling a record, other records in the same row can be useful. Since the record in the row is not sequential, we use a self-attention network which is similar to Liu and Lapata (2018) to model records in the context of other records in the same row. Let $\boldsymbol{r^{row}_{i,j}}$ be the row dimension representation of the record of $i^{th}$ row and $j^{th}$ column. Then, we obtain the context vector in row dimension $\boldsymbol{c^{row}_{i,j}}$ by attending to other records in the same row as follows. Please note that $\alpha^{row}_{i,j,j'} \propto exp(\boldsymbol{r^T_{i,j} W_o r_{i,j'}})$ is normalized across records in the same row $i$. $\boldsymbol{W_o}$ is a trainable parameter.

$$\boldsymbol{c^{row}_{i,j}} = \sum_{j',j' \neq j} \alpha^{row}_{i,j,j'} \boldsymbol{r_{i,j'}} \tag{2}$$

Then, we combine record's representation with $\boldsymbol{c_{i,j}}$ and obtain the row dimension record representation $\boldsymbol{r^{row}_{i,j}} = tanh(\boldsymbol{W_f}[\boldsymbol{r_{i,j}}; \boldsymbol{c^{row}_{i,j}}])$. $\boldsymbol{W_f}$ is a trainable parameter.

#### 3.1.2 Column Dimension Encoder

Each input table consists of multiple rows and columns. Each column in the table covers one type of information such as points. Only few of the row may have high points or other type of information and thus become the important one. For example, in "Column Dimension" part of Figure 2, "Al Jefferson" is more important than "Gary Neal" because the former one have more impressive points. Therefore, when encoding a record, it is helpful to compare it with other records in the same column in order to understand the performance level reflected by the record among his teammates (rows). We employ self-attention similar to the one used in Section 3.1.1 in column dimension to compare between records. We let $\boldsymbol{r^{col}_{i,j}}$ be the column representation of the record of $i^{th}$ row and $j^{th}$ column. We obtain context vector in column dimension $\boldsymbol{c^{col}_{i,j}}$ as follows. Please note that $\alpha_{j,i,i'}$ is normalized across records from different rows $i'$ but of the same column $j$. The column dimension representation $\boldsymbol{r^{col}_{i,j}}$ is obtained similar to row dimension.

$$\boldsymbol{c^{col}_{i,j}} = \sum_{i',i' \neq i} \alpha^{col}_{j,i,i'} \boldsymbol{r_{i',j}} \tag{3}$$

#### 3.1.3 Time Dimension Encoder

As mentioned in Section 1, we find some expressions in texts require information about players' historical (time dimension) performance. So the history information of record $\boldsymbol{r_{i,j}}$ is important. Note that we have already constructed timeline for each record entity and type as described in Section 2.2. Given those timelines, We collect records with same entity and type in the timeline which has date before date $k$ of the record $\boldsymbol{r_{i,j}}$ as history information. Since for some record, the history information can be too long, we set a history window. Thus, we keep most recent history information sequence within history window and denote them as $hist(\boldsymbol{r_{i,j}})$. We model this kind of information in time dimension via self-attention. However, unlike the unordered nature of rows and columns, the history information is sequential. Therefore, we introduce a trainable position embedding $\boldsymbol{emb_{pos}(k')}$ and add it to the record's representation and obtain a new record representation $\boldsymbol{rp_{k'}}$. It denotes the representation of a record with the same entity and type of $\boldsymbol{r_{i,j}}$ but of the date $k'$ before $k$ in the corresponding history window. We use $\boldsymbol{r^{time}_{i,j}}$ to denote the history representation of the record of $i^{th}$ row and $j^{th}$ column. Then the history dimension context vector is obtained by attending to history records in the window. Please note that we use 1-layer MLP as score function here and $\alpha^{time}_{k,k'}$ is normalized within the history window. We obtain the time dimension representation $\boldsymbol{r^{time}_{i,j}}$ similar to row dimension.

$$\alpha^{time}_{k,k'} \propto exp(score(\boldsymbol{rp_k}, \boldsymbol{rp_{k'}})) \tag{4}$$

$$\boldsymbol{c^{time}_{i,j}} = \sum_{k' < k} \alpha^{time}_{k,k'} \boldsymbol{rp_{k'}} \tag{5}$$

### 3.2 Layer 2: Record Fusion Gate

After obtaining record representations in three dimension, it is important to figure out which representation plays a more important role in reflecting the record's information. If a record stands

out from other row's records of same column, the column dimension representation may have a higher weight in forming the overall record representation. If a record differs from previous match significantly, the history dimension representation may have a higher weight. Also, some types of information may appear in texts more frequently together which can be reflected by row dimension representation. Therefore, we propose a record fusion gate to adaptively combine all three dimension representations. First, we concatenate $\boldsymbol{r}_{i,j}^{\boldsymbol{row}}$, $\boldsymbol{r}_{i,j}^{\boldsymbol{col}}$ and $\boldsymbol{r}_{i,j}^{\boldsymbol{time}}$, then adopt a 1-layer MLP to obtain a general representation $\boldsymbol{r}_{i,j}^{\boldsymbol{gen}}$ which we consider as a baseline representation of records' information. Then, we compare each dimension representation with the baseline and obtain its weight in the final record representation. We use 1-layer MLP as the score function. Equation 6 shows an example of calculating column dimension representation's weight in the final record representation. The weight of row and time dimension representation is obtained similar to the weight of column dimension representation.

$$\alpha_{fus}^{col} \propto exp(score(\boldsymbol{r}_{i,j}^{\boldsymbol{col}}, \boldsymbol{r}_{i,j}^{\boldsymbol{gen}})) \quad (6)$$

In the end, the fused record representation $\tilde{\boldsymbol{r}}_{\boldsymbol{i,j}}$ is the weighted sum of the three dimension representations.

$$\tilde{\boldsymbol{r}}_{\boldsymbol{i,j}} = \alpha_{fus}^{row}\boldsymbol{r}_{i,j}^{\boldsymbol{row}} + \alpha_{fus}^{col}\boldsymbol{r}_{i,j}^{\boldsymbol{col}} + \alpha_{fus}^{time}\boldsymbol{r}_{i,j}^{\boldsymbol{time}} \quad (7)$$

### 3.3 Layer 3: Row-level Encoder

For each row, we combine its records via mean pooling (Equation 8) in order to obtain a general representation of the row which may reflect the row (player or team)'s overall performance. $C$ denotes the number of columns.

$$\boldsymbol{row}_{\boldsymbol{i}} = MeanPooling(\tilde{\boldsymbol{r}}_{\boldsymbol{i,1}}, \tilde{\boldsymbol{r}}_{\boldsymbol{i,2}}, ..., \tilde{\boldsymbol{r}}_{\boldsymbol{i,C}}) \quad (8)$$

Then, we adopt content selection gate $\boldsymbol{g}_{\boldsymbol{i}}$, which is proposed by Puduppully et al. (2019) on rows' representations $\boldsymbol{row}_{\boldsymbol{i}}$, and obtain a new representation $\tilde{\boldsymbol{row}}_{\boldsymbol{i}} = \boldsymbol{g}_{\boldsymbol{i}} \odot \boldsymbol{row}_{\boldsymbol{i}}$ to choose more important information based on each row's context.

### 3.4 Decoder with Dual Attention

Since record encoders with record fusion gate provide record-level representation and row-level encoder provides row-level representation. Inspired by Cohan et al. (2018), we can modify the decoder in base model to first choose important row then attend to records when generating each word. Following notations in Section 2.3, $\beta_{t,i} \propto exp(score(\boldsymbol{d}_{\boldsymbol{t}}, \boldsymbol{row}_{\boldsymbol{i}}))$ obtains the attention weight with respect to each row. Please note that $\beta_{t,i}$ is normalized across all row-level representations from all three tables. Then, $\gamma_{t,i,j} \propto exp(score(\boldsymbol{d}_{\boldsymbol{t}}, \tilde{\boldsymbol{r}}_{\boldsymbol{i,j}}))$ obtains attention weight for records. Please note that we normalize $\gamma_{t,i,j}$ among records in the same row.

We use the row-level attention $\beta_{t,i}$ as guidance for choosing row based on row's general representation. Then we use it to re-weight the record-level attention $\gamma_{t,i,j}$ and change the attention weight in base model to $\tilde{\alpha}_{t,i,j}$. Please note that $\tilde{\alpha}_{t,i,j}$ sum to 1 across all records in all tables.

$$\tilde{\alpha}_{t,i,j} = \beta_{t,i}\gamma_{t,i,j} \quad (9)$$

### 3.5 Training

Given a batch of input tables $\{S\}_G$ and reference output $\{Y\}_G$, we use negative log-likelihood as the loss function for our model. We train the model by minimizing $L$. $G$ is the number of examples in the batch and $T_g$ represents the length of $g^{th}$ reference's length.

$$L = -\frac{1}{G}\sum_{g=1}^{G}\sum_{t=1}^{T_g} logP(y_{t,g}|y_{<t,g}, S_g) \quad (10)$$

## 4 Experiments

### 4.1 Dataset and Evaluation Metrics

We conducted experiments on ROTOWIRE (Wiseman et al., 2017). For each example, it provides three tables as described in Section 2.1 which consists of 628 records in total with a long game summary. The average length of game summary is 337.1. In this paper, we followed the data split introduced in Wiseman et al. (2017): 3398 examples in training set, 727 examples in development set and 728 examples in test set. We followed Wiseman et al. (2017)'s work and use BLEU (Papineni et al., 2002) and three extractive evaluation metrics RG, CS and CO (Wiseman et al., 2017) for evaluation. The main idea of the extractive evaluation metrics is to use an Information Extraction (IE) model to identify records mentioned in texts. Then compare them with tables or records extracted from reference to evaluate the model. RG (Relation Generation) measures content fidelity of

| Model | RG | | CS | | | CO | BLEU |
|---|---|---|---|---|---|---|---|
| | P% | # | P% | R% | F1% | DLD% | |
| **Development** | | | | | | | |
| Gold | 94.79 | 23.31 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 |
| Template | **99.92** | **54.23** | 26.60 | **59.13** | 36.69 | 14.39 | 8.62 |
| CC (Wiseman et al., 2017) | 75.10 | 23.95 | 28.11 | 35.86 | 31.52 | 15.33 | 14.57 |
| NCP+CC (Puduppully et al., 2019) | 87.51 | 33.88 | 33.52 | 51.21 | 40.52 | 18.57 | 16.19 |
| Hierarchical LSTM Encoder | 91.59 | 32.56 | 31.62 | 44.22 | 36.87 | 17.49 | 15.21 |
| Hierarchical CNN Encoder | 90.86 | 30.59 | 30.32 | 40.28 | 34.60 | 15.75 | 14.08 |
| Hierarchical SA Encoder | 90.46 | 29.82 | 34.39 | 45.43 | 39.15 | 19.81 | 15.62 |
| Hierarchical MHSA Encoder | 92.87 | 28.42 | 34.87 | 42.41 | 38.27 | 18.28 | 15.12 |
| CC (Our implementation) | 76.50 | 22.48 | 29.18 | 34.22 | 31.50 | 15.43 | 13.65 |
| Our Model | 91.84 | 32.11 | **35.39** | 48.98 | **41.09** | **20.70** | **16.24** |
| *-row-level encoder* | 90.19 | 27.90 | 34.70 | 42.53 | 38.22 | 20.02 | 15.32 |
| *-row* | 91.08 | 30.95 | 35.03 | 47.09 | 40.17 | 20.03 | 15.50 |
| *-column* | 91.66 | 28.63 | 34.83 | 43.62 | 38.73 | 19.59 | 15.99 |
| *-time* | 90.94 | 31.43 | 34.62 | 47.74 | 40.13 | 19.81 | 16.10 |
| *-position embedding* | 89.97 | 28.37 | 34.72 | 43.69 | 38.69 | 19.54 | 16.05 |
| *-record fusion gate* | 89.34 | 32.22 | 32.28 | 46.68 | 38.17 | 18.49 | 14.97 |
| **Test** | | | | | | | |
| Gold | 94.89 | 24.14 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 |
| Template | **99.94** | **54.21** | 27.02 | **58.22** | 36.91 | 15.07 | 8.58 |
| CC (Wiseman et al., 2017) | 74.80 | 23.72 | 29.49 | 36.18 | 32.49 | 15.42 | 14.19 |
| OpAtt (Nie et al., 2018) | - | - | - | - | - | - | 14.74 |
| NCP+CC (Puduppully et al., 2019) | 87.47 | 34.28 | 34.18 | 51.22 | 41.00 | 18.58 | 16.50 |
| CC (Our implementation) | 75.37 | 22.32 | 28.91 | 33.12 | 30.87 | 15.34 | 14.02 |
| Our model | 91.46 | 31.47 | **36.09** | 48.01 | **41.21** | **20.86** | **16.85** |

Table 1: Automatic evaluation results. Results were obtained using Puduppully et al. (2019)'s updated models

texts. CS (Content Selection) measures model's ability on content selection. CO (Content Ordering) measures model's ability on ordering the chosen records in texts. We refer the readers to Wiseman et al. (2017)'s paper for more details.

### 4.2 Implementation Details

Following configurations in Puduppully et al. (2019), we set word embedding and LSTM decoder hidden size as 600. The decoder's layer was set to be 2. Input feeding (Luong et al., 2015) was also used for decoder. We applied dropout at a rate 0.3. For training, we used Adagrad (Duchi et al., 2010) optimizer with learning rate of 0.15, truncated BPTT (block length 100), batch size of 5 and learning rate decay of 0.97. For inferring, we set beam size as 5. We also set the history windows size as 3 from {3,5,7} based on the results. Code of our model can be found at https://github.com/ernestgong/data2text-three-dimensions/.

### 4.3 Results

#### 4.3.1 Automatic Evaluation

Table 1 displays the automatic evaluation results on both development and test set. We chose Conditional Copy (CC) model as our baseline, which is the best model in Wiseman et al. (2017). We included reported scores with updated IE model by Puduppully et al. (2019) and our implementation's result on CC in this paper. Also, we compared our models with other existing works on this dataset including OpATT (Nie et al., 2018) and Neural Content Planning with conditional copy (NCP+CC) (Puduppully et al., 2019). In addition, we implemented three other hierarchical encoders that encoded tables' row dimension information in both record-level and row-level to compare with the hierarchical structure of encoder in our model. The decoder was equipped with dual attention (Cohan et al., 2018). The one with LSTM cell is similar to the one in Cohan et al. (2018) with 1 layer from {1,2,3}. The one with CNN cell

(Gehring et al., 2017) has kernel width 3 from {3, 5} and 10 layer from {5,10,15,20}. The one with transformer-style encoder (MHSA) (Vaswani et al., 2017) has 8 head from {8, 10} and 5 layer from {2,3,4,5,6}. The heads and layers mentioned above were for both record-level encoder and row-level encoder respectively. The self-attention (SA) cell we used, as described in Section 3, achieved better overall performance in terms of F1% of CS, CO and BLEU among the hierarchical encoders. Also we implemented a template system same as the one used in Wiseman et al. (2017) which outputted eight sentences: an introductory sentence (two teams' points and who win), six top players' statistics (ranked by their points) and a conclusion sentence. We refer the readers to Wiseman et al. (2017)'s paper for more detailed information on templates. The gold reference's result is also included in Table 1. Overall, our model performs better than other neural models on both development and test set in terms of RG's P%, F1% score of CS, CO and BLEU, indicating our model's clear improvement on generating high-fidelity, informative and fluent texts. Also, our model with three dimension representations outperforms hierarchical encoders with only row dimension representation on development set. This indicates that cell and time dimension representation are important in representing the tables. Compared to reported baseline result in Wiseman et al. (2017), we achieved improvement of 22.27% in terms of RG, 26.84% in terms of CS F1%, 35.28% in terms of CO and 18.75% in terms of BLEU on test set. Unsurprisingly, template system achieves best on RG P% and CS R% due to the included domain knowledge. Also, the high RG # and low CS P% indicates that template will include vast information while many of them are deemed redundant. In addition, the low CO and low BLEU indicates that the rigid structure of the template will produce texts that aren't as adaptive to the given tables and natural as those produced by neural models. Also, we conducted ablation study on our model to evaluate each component's contribution on development set. Based on the results, the absence of row-level encoder hurts our model's performance across all metrics especially the content selection ability.

Row, column and time dimension information are important to the modeling of tables because subtracting any of them will result in performance drop. Also, position embedding is critical when modeling time dimension information according to the results. In addition, record fusion gate plays an important role because BLEU, CO, RG P% and CS P% drop significantly after subtracting it from full model. Results show that each component in the model contributes to the overall performance. In addition, we compare our model with delayed copy model (DEL) (Li and Wan, 2018) along with gold text, template system (TEM), conditional copy (CC) (Wiseman et al., 2017) and NCP+CC (NCP) (Puduppully et al., 2019). Li and Wan (2018)'s model generate a template at first and then fill in the slots with delayed copy mechanism. Since its result in Li and Wan (2018)'s paper was evaluated by IE model trained by Wiseman et al. (2017) and "relexicalization" by Li and Wan (2018), we adopted the corresponding IE model and re-implement "relexicalization" as suggested by Li and Wan (2018) for fair comparison. Please note that CC's evaluation results via our re-implemented "relexicalization" is comparable to the reported result in Li and Wan (2018). We applied them on models other than DEL as shown in Table 2 and report DEL's result from (Li and Wan, 2018)'s paper. It shows that our model outperform Li and Wan (2018)'s model significantly across all automatic evaluation metrics in Table 2.

| Model | RG | | CS | | CO | BLEU |
|---|---|---|---|---|---|---|
| | P% | # | P% | R% | DLD% | |
| Gold | 96.01 | 17.17 | 100.00 | 100.00 | 100.00 | 100.00 |
| TEM | **99.97** | **54.14** | 23.88 | **72.63** | 11.90 | 8.33 |
| CC | 75.26 | 16.37 | 32.63 | 39.62 | 15.34 | 14.03 |
| DEL* | 84.86 | 19.31 | 30.81 | 38.79 | 16.34 | 16.19 |
| NCP | 87.99 | 24.50 | 35.97 | 55.85 | 16.98 | 16.22 |
| Ours | 92.51 | 22.73 | **38.52** | 52.98 | **19.95** | **16.69** |

Table 2: Automatic evaluation results on test set. Results were obtained using Wiseman et al. (2017)'s trained extractive evaluation models with relexicalization (Li and Wan, 2018). * We include delayed copy (DEL)'s result in the paper (Li and Wan, 2018) for comparison.

#### 4.3.2 Human Evaluation

In this section, we hired three graduates who passed intermediate English test (College English Test Band 6) and were familiar with NBA games to perform human evaluation.

First, in order to check if history information is important, we sampled 100 summaries from train-

| Model | #Sup | #Cont | #Gram | #Coher | #Conc |
|---|---|---|---|---|---|
| Gold | 3.48 | 0.19 | 16.67 | 24.22 | 25.78 |
| Temp | 7.83 | 0.00 | 11.56 | -16.67 | 21.11 |
| CC | 3.91 | 1.23 | -11.33 | -7.78 | -28.00 |
| NCP | 5.15 | 0.82 | -17.33 | -5.33 | -17.11 |
| Ours | 3.63 | 0.44 | 0.44 | 5.56 | -1.78 |

Table 3: Human evaluation results.

ing set and asked raters to manually check whether the summary contained expressions that need to be inferred from history information. It turns out that 56.7% summaries of the sampled summaries need history information.

Following human evaluation settings in Puduppully et al. (2019), we conducted the following human evaluation experiments at the same scale. The second experiment is to assess whether the improvement on relation generation metric reported in automatic evaluation is supported by human evaluation. We compared our full model with gold texts, template-based system, CC (Wiseman et al., 2017) and NCP+CC (NCP) (Puduppully et al., 2019). We randomly sampled 30 examples from test set. Then, we randomly sampled 4 sentences from each model's output for each example. We provided the raters of those sampled sentences with the corresponding NBA game statistics. They were asked to count the number of supporting and contradicting facts in each sentence. Each sentence is rated independently. We report the average number of supporting facts (#Sup) and contradicting facts (#Cont) in Table 3. Unsurprisingly, template-based system includes most supporting facts and least contradicting facts in its texts because the template consists of a large number of facts and all of those facts are extracted from the table. Also, our model produces less contradicting facts than other two neural models. Although our model produces less supporting facts than NCP and CC, it still includes enough supporting facts (slightly more than gold texts). Also, comparing to NCP+CC (NCP)s tendency to include vast information that contain redundant information, our models ability to select and accurately convey information is better. All other results (Gold, CC, NCP and ours) are significantly different from template-based system's results in terms of number of supporting facts according to one-way ANOVA with posthoc Tukey HSD tests. All significance difference reported in this paper are less than 0.05. Our model is also significantly different from the NCP model. As for average number of contradicting facts, our model is significantly different from other two neural models. Surprisingly, gold texts were found containing contradicting facts. We checked the raters's result and found that gold texts occasionally include wrong field-goal or three-point percent or wrong points difference between the winner and the defeated team. We can treat the average contradicting facts number of gold texts as a lower bound.

In the third experiment, following Puduppully et al. (2019), we asked raters to evaluate those models in terms of grammaticality (is it more fluent and grammatical?), coherence (is it easier to read or follows more natural ordering of facts? ) and conciseness (does it avoid redundant information and repetitions?). We adopted the same 30 examples from above and arranged every 5-tuple of summaries into 10 pairs. Then, we asked the raters to choose which system performs the best given each pair. Scores are computed as the difference between percentage of times when the model is chosen as the best and percentage of times when the model is chosen as the worst. Gold texts is significantly more grammatical than others across all three metrics. Also, our model performs significantly better than other two neural models (CC, NCP) in all three metrics. Template-based system generates significantly more grammatical and concise but significantly less coherent results, compared to all three neural models. Because the rigid structure of texts ensures the correct grammaticality and no repetition in template-based system's output. However, since the templates are stilted and lack variability compared to others, it was deemed less coherent than the others by the raters.

#### 4.3.3 Qualitative Example

**Our model**: The Charlotte Hornets ( 21 - 27 ) defeated the Washington Wizards ( 31 - 18 ) 92 - 88 on Monday … The Hornets were led by **Al Jefferson** , who recorded a **double - double** of his own with 18 points ( 9 - 19 FG , 0 - 2 FT ) and 12 rebounds . It was his **second double - double** over his last three games … The only other Wizard to reach **double - digit points** was **Kris Humphries** , who came off the bench for 13 points ( 4 - 8 FG , 5 - 6 FT ) and five rebounds in 26 minutes …

Figure 3: An generation example of our model based on the same tables in Figure 1. Text that accurately reflects players (Al Jefferson and Kris Humphries) performance is in red.

Figure 3 shows an example generated by our model. It evidently has several nice properties: it can accurately select important player "Al Jef-

ferson" from the tables who is neglected by baseline model, which need the model to understand performance difference of a type of data (column) between each rows (players). Also it correctly summarize performance of "Al Jefferson" in this match as "double-double" which requires ability to capture dependency from different columns (different type of record) in the same row (player). In addition, it models "Al Jefferson" history performance and correctly states that "It was his second double-double over his last three games", which is also mentioned in gold texts included in Figure 1 in a similar way.

## 5 Related Work

In recent years, neural data-to-text systems make remarkable progress on generating texts directly from data. Mei et al. (2016) proposes an encoder-aligner-decoder model to generate weather forecast, while Jain et al. (2018) propose a mixed hierarchical attention. Sha et al. (2018) proposes a hybrid content- and linkage-based attention mechanism to model the order of content. Liu et al. (2018) propose to integrate field information into table representation and enhance decoder with dual attention. Bao et al. (2018) develops a table-aware encoder-decoder model. Wiseman et al. (2017) introduced a document-scale data-to-text dataset, consisting of long text with more redundant records, which requires the model to select important information to generate. We describe recent works in Section 1. Also, some studies in abstractive text summarization encode long texts in a hierarchical manner. Cohan et al. (2018) uses a hierarchical encoder to encode input, paired with a discourse-aware decoder. Ling and Rush (2017) encode document hierarchically and propose coarse-to-fine attention for decoder. Recently, Liu et al. (2019) propose a hierarchical encoder for data-to-text generation which uses LSTM as its cell. Murakami et al. (2017) propose to model stock market time-series data and generate comments. As for incorporating historical background in generation, Robin (1994) proposed to build a draft with essential new facts at first, then incorporate background facts when revising the draft based on functional unification grammars. Different from that, we encode the historical (time dimension) information in the neural data-to-text model in an end-to-end fashion. Existing works on data-to-text generation neglect the joint representation of tables' row, column and time dimension information. In this paper, we propose an effective hierarchical encoder which models information from row, column and time dimension simultaneously.

## 6 Conclusion

In this work, we present an effective hierarchical encoder for table-to-text generation that learns table representations from row, column and time dimension. In detail, our model consists of three layers, which learn records' representation in three dimension, combine those representations via their sailency and obtain row-level representation based on records' representation. Then, during decoding, it will select important table row before attending to records. Experiments are conducted on ROTOWIRE, a benchmark dataset of NBA games. Both automatic and human evaluation results show that our model achieves the new state-of-the-art performance.

## Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments. We'd also like to thank Xinwei Geng, Yibo Sun, Zhengpeng Xiang and Yuyu Chen for their valuable input. This work was supported by the National Key R&D Program of China via grant 2018YFB1005103 and National Natural Science Foundation of China (NSFC) via grant 61632011 and 61772156.

## References

Junwei Bao, Duyu Tang, Nan Duan, Zhao Yan, Yuanhua Lv, Ming Zhou, and Tiejun Zhao. 2018. Table-to-text: Describing table region with natural language. In *The Thirty-Second AAAI Conference on Artificial Intelligence*, pages 5020–5027. Association for the Advancement of Artificial Intelligence.

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. 2018. A discourse-aware attention model for abstractive summarization of long documents. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 615–621. ACL.

John C. Duchi, Elad Hazan, and Yoram Singer. 2010. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1243–1252. JMLR.

Dimitra Gkatzia. 2016. Content selection in data-to-text systems: A survey.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 140–149. ACL.

Parag Jain, Anirban Laha, Karthik Sankaranarayanan, Preksha Nema, Mitesh M. Khapra, and Shreyas Shetty. 2018. A mixed hierarchical attention based encoder-decoder approach for standard table summarization. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 622–627. ACL.

Liunian Li and Xiaojun Wan. 2018. Point precisely: Towards ensuring the precision of data in generated texts using delayed copy mechanism. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1044–1055. ACL.

Jeffrey Ling and Alexander Rush. 2017. Coarse-to-fine attention models for document summarization. In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 33–42. ACL.

Tianyu Liu, Fuli Luo, Qiaolin Xia, Shuming Ma, Baobao Chang, and Zhifang Sui. 2019. Hierarchical encoder with auxiliary supervision for neural table-to-text generation: Learning better representation for tables. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 6786–6793. Association for the Advancement of Artificial Intelligence.

Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. 2018. Table-to-text generation by structure-aware seq2seq learning. In *The Thirty-Second AAAI Conference on Artificial Intelligence*, pages 4881–4888. Association for the Advancement of Artificial Intelligence.

Yang P. Liu and Mirella Lapata. 2018. Learning structured text representations. *Transactions of the Association for Computational Linguistics*, 6:63–75.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421. ACL.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. What to talk about and how? selective generation using LSTMs with coarse-to-fine alignment. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 720–730. ACL.

Soichiro Murakami, Akihiko Watanabe, Akira Miyazawa, Keiichi Goshima, Toshihiko Yanase, Hiroya Takamura, and Yusuke Miyao. 2017. Learning to generate market comments from stock prices. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1374–1384. ACL.

Feng Nie, Jinpeng Wang, Jin-Ge Yao, Rong Pan, and Chin-Yew Lin. 2018. Operation-guided neural networks for high fidelity data-to-text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3879–3889. ACL.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318. ACL.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019. Data-to-text generation with content selection and planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 6908–6915. Association for the Advancement of Artificial Intelligence.

Ehud Reiter and Robert Dale. 2000. *Building natural language generation systems*. Cambridge university press.

Jacques Robin. 1994. Revision-based generation of natural language summaries providing historical background: corpus-based analysis, design, implementation and evaluation. *Ph.D. thesis*.

Lei Sha, Lili Mou, Tianyu Liu, Pascal Poupart, Sujian Li, Baobao Chang, and Zhifang Sui. 2018. Order-planning neural text generation from structured data. In *The Thirty-Second AAAI Conference on Artificial Intelligence*, pages 5414–5421. Association for the Advancement of Artificial Intelligence.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008. Curran Associates, Inc.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263. ACL.

# Subtopic-driven Multi-Document Summarization

**Xin Zheng**[1,2], **Aixin Sun**[1], **Jing Li**[3], **Karthik Muthuswamy**[2]
[1] School of Computer Science and Engineering, Nanyang Technological University, Singapore
[2] SAP Innovation Center Network, SAP Asia Pte Ltd, Singapore
[3] Inception Institute of Artificial Intelligence Abu Dhabi, United Arab Emirates
xzheng008@e.ntu.edu.sg, axsun@ntu.edu.sg,
jingli.phd@hotmail.com, karthik.muthuswamy@sap.com

## Abstract

In multi-document summarization, a set of documents to be summarized is assumed to be on the same topic, known as the *underlying topic* in this paper. That is, the underlying topic can be collectively represented by all the documents in the set. Meanwhile, different documents may cover various different subtopics and the same subtopic can be across several documents. Inspired by topic model, the underlying topic of a document set can also be viewed as a collection of different subtopics of different importance. In this paper, we propose a summarization model called *STDS*. The model generates the underlying topic representation from both document view and subtopic view in parallel. The learning objective is to minimize the distance between the representations learned from the two views. The contextual information is encoded through a hierarchical RNN architecture. Sentence salience is estimated in a hierarchical way with subtopic salience and relative sentence salience, by considering the contextual information. Top ranked sentences are then extracted as a summary. Note that the notion of subtopic enables us to bring in additional information (*e.g.,* comments to news articles) that is helpful for document summarization. Experimental results show that the proposed solution outperforms state-of-the-art methods on benchmark datasets.

## 1 Introduction

Multi-document summarization (MDS) is useful in many applications, *e.g.,* summarizing answers in forums (Song et al., 2017) and reports of burst events (Kedzie et al., 2016). In MDS, the documents in a set are assumed to share the same underlying topic. Given a set of documents to be summarized, the important information is collectively determined by all the documents in the set, rather than simple integration of key points in each document. Hence, the correlations among documents become crucial for identifying the important information to be included in a summary. Moreover, each sentence should not be interpreted independently. Its context (or the surrounding sentences) does affect the information expressed in a sentence (Nenkova et al., 2006; Ren et al., 2017). Therefore, both correlations among documents and contextual information within a single document should be considered in MDS.

Various methods have been developed for MDS. Graph-based models (Mihalcea and Tarau, 2004; Erkan and Radev, 2004) blend sentences from different documents together and attempt to leverage the correlations among documents to extract the most representative sentences. Nonetheless these methods are less effective when the sentence graph is not well connected. Thus, many studies attempt to construct well connected graphs, *e.g.,* group sentences into clusters (Wan and Yang, 2008; Banerjee et al., 2015) and construct dense features with distributed embeddings (Yasunaga et al., 2017). However, these methods do not address the context information for sentences in each individual document. There are also models exploring fine-grained word or phrase level information (Li et al., 2015, 2017a,b). The resultant summary consists of salient words or phrases that are selected by integer linear programming (ILP). Recently, many studies have taken advantage of the strengths of neural models (Ren et al., 2017; Zhou et al., 2018; Lebanoff et al., 2018). However, these models are not designed to well deal with both correlations among documents in the set and the contextual information in each individual document.

To address the two aforementioned issues, we propose a subtopic-driven summarization solution, named *STDS*. To encode sequential context information, we adopt a hierarchical bidirectional RNN to produce representations for sentences,



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3153–3162, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

documents, and the underlying topic. As stated earlier, within a document set, the documents may cover various subtopics and the same subtopic can be across several articles. These subtopics implicitly reflect the correlations among documents. Inspired by topic model (Blei et al., 2003), we simply assume that $k$ latent subtopics are depicted by the documents. We design our model to learn the representations of "subtopics" by itself, by assigning sentences to different subtopics in a soft manner instead of hard clustering. Thus, the underlying topic of a given document set can be presented from document view and subtopic view, in parallel. The subtopic view also gives us the flexibility of incorporating additional information (*e.g.,* comments to news articles). Readers' comments have been found useful for highlighting crucial information (Li et al., 2017a). The salience of subtopics are estimated by using attention mechanism (Bahdanau et al., 2015) with respect to the underlying topic generated from the document view. Similarly, relative sentence salience can be estimated for each subtopic by considering context information. By multiplying the saliences of subtopics and sentences, an overall ranking of sentences can be obtained. Top-ranked sentences are extracted as a summary within a given length limit. The contributions of this work are as follows:

- We propose to explore correlations among documents with "subtopics". The subtopics and documents provide us parallel views of the underlying topic and enable sentence and subtopic salience estimation in an unsupervised fashion.
- We build up a unified model that tackles the correlations among documents and contextual information within each document together in an inherent way.
- Our model can be applied to documents with or without comments. Extensive experiments are conducted on benchmark datasets, *i.e.,* RA-MDS and DUC 2004. The experimental results show that our STDS outperforms state-of-the-art baselines.

## 2 Related Work

### 2.1 Extractive Methods

Extractive methods select sentences from documents to form a summary. A typical framework is based on a graph, where sentences are vertices and similarities between sentences are edge weights, *e.g.,* TextRank (Mihalcea and Tarau, 2004) and LexRank (Erkan and Radev, 2004). They apply a random walk to explore the relationships among sentences and then produce a sentence ranking. MEAD (Radev et al., 2000) is a centroid-based method, which ranks sentences based on a set of features, including centroid value, positional value, first sentence overlap and redundancy.

However, the graph-based models do not perform well when the sentence graph is not well connected. Wan and Yang (2008) claim that the document set is usually composed of a few themes, which are represented by sets of sentences. They apply conditional Markov Random Walk and HITS on clusters, separately. Alternatively, Haghighi and Vanderwende (2009) adopt a hierarchical Latent Dirichlet Allocation based model to discover the multiple themes within a document set. Similarly, Gong *et al.* (2010) use the theme structure to define the representation for each sentence. However, their solutions only consider statistical knowledge. Semantic and contextual information among sentences are neglected.

Following the idea of themes, Banerjee *et al.* (2015) suggest that the sentences in the most important document of the set are relevant to the sentences in the other documents. Hence, they cluster sentences based on those in the most important document. However, there may not always exist a most important document in the set. Further, the relationships among documents are not necessarily to be conclusive. In fact, there are many kinds of relationships among documents (*e.g.,* similar, complementary, or evolutionary).

Liu *et al.* (2015) adopt the idea of reconstruction. They apply a two-level sparse representation model and reconstruct the document by extracted sentences with constraints. Similarly, Ma *et al.* (2016) try to minimize the reconstruction error between selected sentences and the document set with a neural model. Cao *et al.* (2017) make use of multi-task learning by incorporating classification task with summarization to train better sentence representations for reconstruction.

### 2.2 Abstractive Methods

Summarizing multiple documents in an abstractive way is even harder. The generation should take many inputs into account. Some abstrac-

tive methods on single document summarization have been proposed recently (See et al., 2017; Zhou et al., 2018), and they benefit from an attentive sequence-to-sequence model (Sutskever et al., 2014). In fact, abstractive approaches for MDS are more like words or phrases recombination.

Bing *et al.* (2015) propose an abstractive MDS solution. They extract salient noun and verb phrases from a constituency tree, then produce sentences with representative phrases via integer linear programming. Later, Li *et al.* (2017b; 2017c) adopt a similar two-stage model, but they first estimate sentence and phrase salience via an auto-encoder framework.

Recently, some studies have turned to readers' comments to help identify crucial points to include in a summary. Li *et al.* (2015) propose a sparse coding method to generate summaries that not only cover key content in news but also focuses highlighted by readers' comments. However, they do not consider semantic information. Later, they propose a deep learning method (Li et al., 2017a) that jointly models the focuses of news set and readers' comments. But they do not tackle sequential context information among sentences and treat them as separate instances. In contrast, we deal with sequential context information within each document and the relationships among documents.

## 3 Preliminary

We define our problem as follows. Given a set of news documents on the same topic, $D = \{d_1, d_2, ..., d_n\}$, we also have a set of comments $C_i = \{c_{i,1}, c_{i,2}, ..., c_{i,h}\}$ for each document $d_i$. Both news $d_i$ and comment $c_{i,j}$ consist of a sequence of sentences, *i.e.,* $d_i = \{s_1, s_2, ..., s_{|d_i|}\}$ and $c_{i,j} = \{y_1, y_2, ..., y_{|c_{i,j}|}\}$. We aim to extract representative *news sentences* to summarize the set of news articles by considering both news and comments.

To better leverage comments, we explore the correlation between news and comments based on the dataset provided by Li *et al.* (2017a). The dataset contains 45 topics (*i.e.,* sets of news articles), and each topic includes 10 news articles. The number of comments associated with each news article is not evenly distributed and ranges from 0 to 248. Overall, on average, each topic is associated with 215 comments and 940 comment sentences.

The correlations between news and comments determines how we use them. Thus, we make the following analysis. Since the news documents share the same topic, comments about one news may also be related to the others. Meanwhile, it is common for users to make comments after reading through several topic-related news articles. Thus, comments for one news could also contain information from related articles. To verify our assumption, we calculate the ratio of overlapped vocabulary for each news and its comments, over news vocabulary, and the average is 0.345. Against news-comments pairs, we compute the ratio of overlapped vocabulary for all the news and comments of the same topic, over news vocabulary, and the average is 0.447. Thus, considering comments as related to the whole news set is more reasonable than news-comment pairs, which coincides with (Li et al., 2015, 2017a).

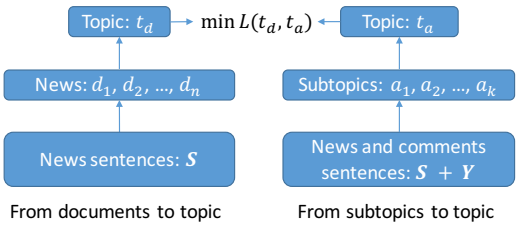


Figure 1: Framework of STDS. The underlying topic of the news set is represented from a document view and a subtopic view respectively. The target is to minimize the distance between the two representations $L(t_d, t_a)$.

In short, our model inputs are a news set $D = \{d_1, d_2, ..., d_n\}$ with news sentences $S$, and a set of comments $C = \{c_1, c_2, ..., c_m\}$ with comment sentences $Y$. Later, we will use boldface of each notation to represent its embedding.

## 4 Model Description

Our model tackles correlations among documents and sequential context in each individual document, through "subtopics" and hierarchical semantic embedding. We first introduce the embedding generation with context for sentences, documents, and subtopics in Sections 4.1 and 4.2. Then, we describe salience estimation for subtopics and sentences in Sections 4.3 and 4.4.

### 4.1 Sentence and Document Representation

We start representation generation from words, and pre-train word embedding $\boldsymbol{W} \in \mathbb{R}^{|\mathcal{V}| \times u}$ by Word2Vec (Mikolov et al., 2013) on both news and comments, where $\mathcal{V}$ is the overall vocabulary.

Note that we use $u$ to represent the vector dimension throughout the model description (specific value settings are given in Section 5). The representations for news sentences $\boldsymbol{S}$ and news documents $\boldsymbol{D}$ are learned in a hierarchical structure, as shown in Figure 2. Specifically, the embeddings of words $\boldsymbol{w}$ in each sentence are fed into a bidirectional RNN encoder-decoder framework (Cho et al., 2014), which inherently encodes word sequence information in sentence representations. GRU (Chung et al., 2014) is adopted as the basic RNN unit. Each sentence acts as both encoder input and decoder target. Thus, the encoder-decoder could be considered as an RNN auto-encoder. The concatenation of hidden vectors in both directions at the last step of the encoder is adopted as the sentence embedding, $\boldsymbol{s} = [\overrightarrow{\boldsymbol{h}_{|s|}}^\top; \overleftarrow{\boldsymbol{h}_{|s|}}^\top]^\top$. The comment sentence embedding $\boldsymbol{Y}$ is generated in the same way. The loss function for the RNN encoder-decoder is:

$$\mathcal{L}_e = -\sum_{X \in S \cup Y} \log p(X'|X;\theta) \qquad (1)$$

where $X'$ is the predicted result and $\theta$ denotes the parameters.

Then, news sentence embeddings $\boldsymbol{s}$ are fed into the next level bidirectional RNN in order, as shown in Figure 2, to generate a document representation $\boldsymbol{D} \in \mathbb{R}^{|D| \times u}$. Note that no decoding process exists from sentence to document. At each encoding step for document embedding, we obtain the hidden vectors on both directions for each sentence, which contain contextual information along with sentences. We concatenate them as a context-enriched sentence embedding $\tilde{\boldsymbol{S}} \in \mathbb{R}^{|S| \times u}$, where $\tilde{\boldsymbol{s}}_i = [\overrightarrow{\boldsymbol{h}_i}^\top; \overleftarrow{\boldsymbol{h}_i}^\top]^\top$. Again, we take the concatenation of hidden vectors in both directions at the last step as the document embedding $\boldsymbol{d} = [\overrightarrow{\boldsymbol{h}_{|d|}}^\top; \overleftarrow{\boldsymbol{h}_{|d|}}^\top]^\top$.

For now, the sequential contextual information in an individual document is encoded in the context-enriched sentence embedding $\tilde{\boldsymbol{S}}$ with the hierarchical bidirectional RNN inherently.

### 4.2 Subtopic Representation

As stated in Section 1, we assume a news set contains $k$ latent subtopics, which are expected to be learned automatically from sentences of news and comments. This is because readers' comments are found useful to highlight important information in news (Li et al., 2015, 2017a).

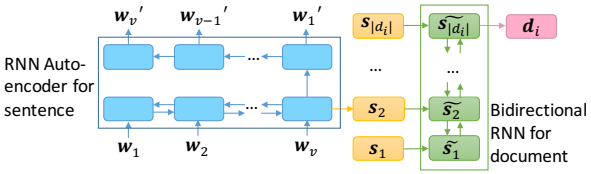


Figure 2: Hierarchical RNN structure for sentence representation $\boldsymbol{s}_j$ and document representation $\boldsymbol{d}_i$ construction. $\boldsymbol{w}_i$ is the embedding of words in each sentence and $\boldsymbol{s}_j$ is the concatenation of hidden vectors in both directions at the last step of the sentence encoder. $\tilde{\boldsymbol{s}}_j$ is a context-enriched sentence embedding, which is the concatenation of hidden states at each step of the document encoder.

Recall that we construct a document embedding from news sentence representations $\boldsymbol{S}$, which are generated from a word sequence. For subtopic representation, we also utilize $\boldsymbol{S}$, and incorporate the comment sentence embedding $\boldsymbol{Y}$. That is, we only take a single sentence without context information to build subtopic representations, as in a typical topic model. Then, a soft clustering is employed, so that sentences of news and comments $\boldsymbol{E} = [\boldsymbol{S}^\top; \boldsymbol{Y}^\top]^\top \in \mathbb{R}^{(|S|+|Y|) \times u}$ can be partial membership of multiple subtopics. Next, a nonlinear transformation $tanh$ is utilized to fuse clustered sentence information into subtopic representations $\boldsymbol{A} \in \mathbb{R}^{k \times u}$ as follows:

$$\boldsymbol{A} = tanh((\boldsymbol{H}\boldsymbol{E} + \boldsymbol{b}_h)\boldsymbol{U} + \boldsymbol{b}_u) \qquad (2)$$

where $\boldsymbol{H}, \boldsymbol{U}, \boldsymbol{b}_h, \boldsymbol{b}_u$ are trainable parameters.

We hope that each subtopic embedding can represent a unique facet of the underlying topic. That means the overlap across different subtopics should be as small as possible. Inspired by Luxburg (2007), we constrain the subtopic embedding to be orthogonal with each other:

$$\mathcal{L}_r = \|\boldsymbol{A}\boldsymbol{A}^\top - \boldsymbol{I}_{k \times k}\|. \qquad (3)$$

### 4.3 Salience Estimation for Subtopic

The purpose of introducing subtopics is not to make the summary diverse. Instead, we aim to identify the important information (and relieve distractions from less crucial content) in the news set, through the notion of subtopics. This is achieved by subtopic salience estimation. We consider the underlying topic of a news set from two parallel perspectives, as shown in Figure 1: (i) the explicit composition of documents, and (ii) the implicit constitution of subtopics. Therefore, from either document view or subtopic view, we should

be able to obtain similar representations for the underlying topic of the news set.

**From documents to the underlying topic.** We obtain a document representation $\boldsymbol{d}$ with a hierarchical RNN. Again, we adopt a bidirectional RNN to encode document embeddings to a topic representation $\boldsymbol{t}_d$, which is the concatenation of hidden vectors in both directions at the last step. Here, the document sequences do not significantly affect model performance in our experiments. In fact, one can also try to use a CNN to get $\boldsymbol{t}_d$. However, our experiments suggest RNN performs better than CNN.

**From subtopics to the underlying topic.** As we stated before, not all subtopics are equally important to the underlying topic. We estimate subtopic salience in order to focus on key information *i.e.,* to generate a distilled topic representation $\boldsymbol{t}_a$. This is different from the topic vector $\boldsymbol{t}_d$ constructed from documents, which encodes all information from all documents.

To estimate subtopic salience, we apply an attention mechanism (Bahdanau et al., 2015) on each subtopic representation $\boldsymbol{a}_i$ with respect to the topic representation $\boldsymbol{t}_d$, which encodes comprehensive information from documents:

$$\alpha_i = \frac{\exp(e_i)}{\sum_{k=1}^{m}\exp(e_k)}, e_i = \boldsymbol{v}^\top tanh(\boldsymbol{H}\boldsymbol{a}_i + \boldsymbol{U}\boldsymbol{t}_d) \tag{4}$$

where $\boldsymbol{H}$, $\boldsymbol{U}$ and $\boldsymbol{v}$ are trainable parameters. We feed the weighted subtopic embedding $\alpha_i \boldsymbol{a}_i$ into a bidirectional RNN. The reason for using an RNN is the same as that for $t_d$ generation, mentioned above. The concatenation of hidden vectors in both directions at the last step is taken as the topic representation $\boldsymbol{t}_a$, generated from subtopics. Both topic representations $\boldsymbol{t}_a$ and $\boldsymbol{t}_d$ should be similar to each other, because they denote the same underlying topic. Since the two vectors are from identical structure as Siamese network (Koch et al., 2015), we adopt contrastive loss (Hadsell et al., 2006) to measure the distance between them:

$$\mathcal{L}_t = \beta * (max\{0, 1 - \boldsymbol{t}_d^\top \boldsymbol{t}_a\})^2 + (1-\beta) * (\boldsymbol{t}_d - \boldsymbol{t}_a)^2 \tag{5}$$

where $\beta$ is a hyper-parameter, empirically set to 0.5.

### 4.4 Salience Estimation for Sentence

Note that, each sentence is also of different importance with respect to different subtopics, which is called *relative sentence salience*. Recall that we claim the sequential context information affects sentence salience. Thus, we adopt the context-enriched sentence embedding $\tilde{\boldsymbol{s}}_j$, (which is presented in Section 4.1), to estimate relative sentence salience $\gamma_{i,j}$ for each subtopic $\boldsymbol{a}_i$, similar as Equation 4.

The subtopic representations are generated from both news $\boldsymbol{S}$ and comments $\boldsymbol{Y}$. On the other hand, the subtopics represent information conveyed by the news set. Thus, we should also be able to construct subtopic vectors solely based on news sentences. To avoid diminishing of comments $\boldsymbol{Y}$, we use the context-enriched news sentence embedding $\tilde{\boldsymbol{S}}$ to approximate subtopic representations $\boldsymbol{A}$, similar as Equation 2:

$$\boldsymbol{A}_{\tilde{s}} = tanh(\boldsymbol{\gamma}\tilde{\boldsymbol{S}}\boldsymbol{H} + \boldsymbol{b_h}) \tag{6}$$

where $\boldsymbol{H}$ and $\boldsymbol{b_h}$ are trainable parameters. $\boldsymbol{A}$ and $\boldsymbol{A}_{\tilde{s}}$ should be similar and we measure the distance between them with contrastive loss:

$$\mathcal{L}_a = \beta * (max\{0, \|\boldsymbol{I} - \boldsymbol{A}\boldsymbol{A}_{\tilde{s}}^\top\|\})^2 + (1-\beta) * (\boldsymbol{A} - \boldsymbol{A}_{\tilde{s}})^2 \tag{7}$$

where $\beta$ is a hyper-parameter, empirically set to 0.5. With backpropagation, relative sentence salience scores $\boldsymbol{\gamma}$ can be learned.

The overall training objective becomes:

$$\mathcal{L} = \mathcal{L}_e + \mathcal{L}_r + \mathcal{L}_t + \mathcal{L}_a. \tag{8}$$

To obtain the global salience score for each sentence, we multiply the max relative salience of each sentence over subtopics $\max \boldsymbol{\gamma}_{:,j}$, with the corresponding subtopic salience $\alpha_i$, (*i.e.,* $i = \arg\max \boldsymbol{\gamma}_{:,j}$). Top ranked sentences are extracted as a summary with length limited (*i.e.,* 100 words).

## 5 Experiments

### 5.1 Datasets and Experimental Settings

Our model is proposed to incorporate comments for summarization, but it can also be applied when no comments are available. Hence, we evaluate our model on two benchmark datasets: one includes comments and the other does not.

**RA-MDS** (Li et al., 2017a): This dataset includes comments and is the one we conduct analysis on in Section 3. For each topic, 4 reference summaries within 100 words are generated by human. The average news length is 27 sentences, and each sentence contains 25 words on average. To alleviate comment bias when generating subtopics, we randomly sample 50 comments

for each topic, and repeat the sampling 10 times. Thus, we obtain 450 news document sets. At test stage, for each topic, we randomly select 50 comments, together with the news set for evaluation.

**DUC2004**[1]: The data is from Task 2 of DUC 2004. It has 50 topics, each of which consists of 10 news articles. For each topic, multiple reference summaries generated by human judges are provided.

**Experimental settings.** The notation $u$ in Section 4 generally represents the hidden vector dimension. We set the word embedding dimension as 300. The hidden units for all GRU cells in one direction are 128, so the hidden vector dimension after concatenating both directions is 256. The dimensions of the linear operations in Equations 2 and 6 are also set to 256. We perform sensitivity experiments on the subtopic number and empirically set the number $k$ to 5. The word embeddings are pre-trained on two datasets separately and are immutable during the training process.

The batch size is 1, which is the data from one news set. One iteration is training one batch of data. We adopt Adam (Kingma and Ba, 2015) as the optimizer. The learning rate is 0.001 and the value is decayed by 0.96 after 1,000 iterations. To reduce the vanishing and exploding gradient problems for RNN training, we apply gradient norm clipping strategy (Pascanu et al., 2013) and set the bounded norm as 1. We stop training when the loss stops decreasing after more than 3 iterations. Our model is implemented with TensorFlow version 1.3[2] and runs on a single GPU.[3]

### 5.2 Baseline Models

We compare our model with both traditional methods and state-of-the-art models on both datasets.

**Lead**: For news, the first several sentences usually act as good summarization. The news sentences are ordered chronologically and top ones are extracted until meeting the length limit.

**MEAD** (Radev et al., 2000): It detects the topics of given document set and uses information from the centroid of each topic to select sentences.

**LexRank** (Erkan and Radev, 2004), **TextRank** (Mihalcea and Tarau, 2004): Both are unsupervised graph-based models. They perform PageRank on a sentence graph, where sentence similarities are edge weights.

**APSM** (Bing et al., 2015): It extracts noun and verb phrases from a contiguity tree and estimates the phrase salience based on handcrafted features. Then, an integer linear programming process is employed to optimize the phrase selection.

**RA-Sparse** (Li et al., 2015): This model tackles reader-aware MDS problem. A sparse-coding-based method is used to calculate sentence salience by jointly considering news documents and readers' comments.

**RAVAESum, RAVAESum/NC** (Li et al., 2017a): They explore the relationships between news and comments via representation matrix multiplication, from which a weight matrix is obtained. The magnitudes of the weight matrix are taken as sentence salience. Then, integer linear programming is used to optimize the summary construction. RAVAESum/NC removes readers' comments from the input and trains the model solely based on news.

The methods below are baselines used on DUC 2004 dataset without comments.

**CLASSY04** (Conroy et al., 2004): It performs the best on the official DUC 2004 evaluation. It applies a Hidden Markov Model and uses topic signature as feature.

**GreedyKL** (Haghighi and Vanderwende, 2009): It focuses on words distribution, and aims to minimize the KL divergence between word probability distribution estimated from summary and that from the input using a greedy approach.

**RegSum** (Hong et al., 2014): This is a supervised method. It adopts weights estimated from three unsupervised methods with a set of handcrafted features for salience estimation.

**ILPSumm** (Banerjee et al., 2015): It groups similar sentences and generates K-shortest paths from sentences in each cluster using a word-graph structure. Then, it selects sentences from the set of shortest paths via ILP.

**AAPRW** (Wang et al., 2017): This is an adjustable affinity-preserving random walk model to keep the summary diverse.

**GRU-GCN** (Yasunaga et al., 2017): It employs an RNN to obtain sentence embedding and feeds them to a Graph Convolutional Network (GCN) as node features. High-level features are generated from the GCN for sentence salience estimation.

**CRSum,CRSum-SF** (Ren et al., 2017): CRSum

[1] https://www-nlpir.nist.gov/projects/duc/data.html

[2] https://www.tensorflow.org/versions/r1.3/api_docs/python/

[3] Tesla P100, 3,584 Cuda cores, 16G GPU memory.

| Method | R-1 | R-2 | R-SU4 |
|---|---|---|---|
| Lead | 0.384 | 0.110 | 0.144 |
| MEAD | 0.402 | 0.141 | 0.171 |
| TextRank | 0.402 | 0.122 | 0.159 |
| LexRank | 0.425 | 0.135 | 0.165 |
| APSM | 0.422 | 0.157 | 0.188 |
| RA-Sparse | 0.442 | 0.157 | 0.188 |
| RAVAESum/NC | 0.437 | 0.162 | 0.189 |
| RAVAESum | 0.443 | 0.171 | 0.196 |
| STDS/NC | 0.443 | 0.163 | 0.177 |
| STDS | **0.456** | **0.187** | **0.205** |

Table 1: Full-length ROUGE $\boldsymbol{F_1}$ score of the proposed methods and baselines on RA-MDS dataset. "R" indicates ROUGE. All the ROUGE scores reported in this work are with significance test and the values fall in a 95% confidence interval with at most $\pm 0.25$ of reported results calculated by the official ROUGE script with default settings. The best $\boldsymbol{F_1}$ scores are in bold and second best are underlined.

| Method | | R-1 | R-2 |
|---|---|---|---|
| Supervised Method | CLASSY04 | 0.376 | 0.090 |
| | RegSum | 0.386 | 0.098 |
| | GRU-GCN | 0.382 | 0.095 |
| | PG-MMR | 0.364 | 0.094 |
| | CRSum | 0.382 | 0.097 |
| | CRSum-SF | 0.395 | 0.106 |
| Unsupervised Method | LexRank | 0.359 | 0.075 |
| | GreedyKL | 0.379 | 0.085 |
| | AAPRW | 0.389 | 0.101 |
| | ILPSum | 0.392 | **0.119** |
| | STDS/NC | **0.397** | 0.107 |

Table 2: ROUGE ***Recall*** for the proposed method and baselines on DUC 2004 dataset. Best ***Recall*** scores are in bold and second best are underlined.

exploits context features within sentences and among sentences with a two-level attention mechanism. CRSum-SF refers to the model combining surface features.

**PG-MMR** (Lebanoff et al., 2018): It leverages the Maximal Marginal Relevance method to select representative sentences from input documents avoiding duplicates, and uses an existing abstractive encoder-decoder model (See et al., 2017) to generate an abstractive summary.

**Our Model STDS**: We feed the first 25 words of each sentence and the first 27 sentences of each document into our model (*i.e.,* which are the average numbers of words and sentences of RA-MDS dataset). Experimental results suggest no significant difference when feeding all the data. Since STDS requires readers' comments, we only evaluate it on RA-MDS dataset.

**Our Model STDS/NC**: This is a variant of STDS without readers' comments as input. The subtopic embeddings are generated solely based on news sentence embedding $\boldsymbol{s}$. For DUC 2004, the first 20 words of each sentence and the first 20 sentences of each document are fed into the model. The other settings remain the same. When no readers' comments exists, our model can be applied in this manner and we test STDS/NC on both datasets.

### 5.3 Performance Comparison

We evaluate the models by pyrouge[4], which is a python wrapper of the official ROUGE toolkit (Lin, 2004). We limit the summary length to 100 words on both datasets to compare with baselines. Following Li *et al.* (2017a), we report ROUGE $F_1$ on RA-MDS in Table 1. For DUC 2004, we report ROUGE $Recall$ following previous studies (Wang et al., 2017) in Table 2.

**RA-MDS:** From Table 1, we observe that our STDS outperforms all baselines on all of ROURGE-1, ROUGE-2 and ROUGE-SU4, which demonstrates the superiority of our model. By incorporating readers' comments, STDS and RAVAESum achieve better results than STDS/NC and RAVAESum/NC. Thus, readers' comments do help construct better summaries. The variant of our model, STDS/NC, achieves comparable performance with the best baseline, RAVAESum, on ROUGE-1. This shows the subtopics and contextually enriched semantic embedding are important for summarization.

MEAD, LexRank and TextRank all tackle correlations among documents, but they do not consider surrounding context influence. STDS and STDS/NC take both elements into consideration and perform much better than these baselines. Thus, context information contributes to better summaries. If this comparison is not direct enough because our models STDS and STDS/NC can benefit from neural model, we compare STDS and RAVAESum next. Both RAVAESum and STDS are neural models. The difference is that our STDS does not take all news and comment sentences as a whole set. STDS distributes sentences into different subtopics of varying importance and incorporates surrounding sentences as context to produce sentence representations. Our model performs better than RAVAESum. Therefore, both

[4] https://pypi.python.org/pypi/pyrouge/0.1.0

| |
|---|
| **[Extracted] d2s1t1:** Facebook launches Internet.org app to let users access basic Internet services for free. **d6s2t1:** Facebook's Internet.org project is taking another step toward its goal of bringing the Internet to people who are not yet online, launching an app Thursday in Zambia. **d3s4t2:** Facebook will not pay Airtel for the bandwidth, Rosen said, but Airtel will benefit as users who are exposed to Internet services eventually decide to pay for broader, unrestricted access. **d8s3t1:** The Internet.org app will give subscribers of Zambia's Airtel phone company access to a set of basic internet services for free. |
| **[Reference] r1:** Facebook launched an Internet.org app in Zambia to provide free access to Facebook and other online services including Wikipedia, Google Search, AccuWeather and websites offering health and other services. **r2:** Facebook aims to bring the Internet to people who are not yet online. **r3:** The new app has the potential to boost the size of Facebooks audience, which currently totals 1.32 billion monthly users. **r4:** Facebook will not pay Airtel for the bandwidth, but Airtel will benefit as users may decide to pay for unrestricted access. **r5:** Google has run its own zero-data initiative called Free Zone including a partnership with Airtel in India. |

Table 3: Summary example.

| Subtopic | Salience | Top-ranked sentence of the subtopic |
|---|---|---|
| t1 | 0.312 | Facebook launches Internet.org app to let users access basic Internet services for free. |
| t2 | 0.284 | Facebook will not pay Airtel for the bandwidth, Rosen said, but Airtel will benefit as users who are exposed to Internet services eventually decide to pay for broader, unrestricted access. |
| t3 | 0.197 | Online services accessible through the app range from AccuWeather to Google search, Wikipedia, a job search site as well as a breadth of health information. |
| t4 | 0.122 | Mark Zuckerberg has said that making connectivity affordable and convincing people that the Internet is something they need are bigger hurdles to connecting people than "satellites or balloons." |
| t5 | 0.085 | The app works on Android phones as well as the simple "feature phones" that are used by the majority of people in Zambia, said Guy Rosen, product management director at Internet.org. |

Table 4: Subtopic example.

subtopics and context information contribute to STDS's good performance. Since our model is unified, (*i.e.,* sentence and subtopic salience estimation relies on both subtopics and contextual information), we cannot carry out ablation experiments to verify the effectiveness of the two factors individually.

**DUC 2004:** Note that our model STDS is designed to incorporate readers' comments. When no comments are available, STDS/NC performs fairly well, as shown in Table 2. We observe that STDS/NC achieves comparable results as state-of-the-art unsupervised method ILPSum and supervised method CRSum-SF, and outperforms the other strong baselines. Both GRU-GCN and STDS/NC exploit the correlations among documents, but STDS/NC encodes contextual information into the sentence embedding. The better performance of STDS/NC than GRU-GCN indicates the effectiveness of surrounding context. Moreover, both CRSum and STDS/NC encode contextual information into sentence embedding, and STDS/NC outperforms the supervised CRSum. Thus, this suggests that our model benefits from the subtopics with different saliences. They provide a better understanding of the correlations among documents and lead to the important subtopics. With the comparisons, we demonstrate the effectiveness of both factors: subtopics and context information.

In a word, without training using reference summaries, STDS and STDS/NC outperform most baseline methods and achieve comparable results with state-of-the-art methods. This shows that our model can be applied to common situations without readers' comments.

### 5.4 Case Study with Example Summary

We present one summary example in Table 3. The notations before extracted sentences represent **d**ocument, **s**entence, and sub**t**opic respectively. The number following **d** is the document number, labeled in the raw dataset. The number after **s** is the sentence position in the document. And the number following **t** indicates the subtopic ranking position based on subtopic salience, which is the same as in Table 4.

As observed, the extracted summary aligns to content of **r1**, **r2** and **r4** in reference summary. After checking the raw documents, we find few content is related to reference sentences **r3** and **r5**.

Table 4 shows the identified subtopic information of the same document set as for Table 3. We list the salience scores and top-ranked sentences to better understand the subtopics. We consider that the top-ranked sentence regarding each subtopic represents the main content of the subtopic. In

this case, the subtopics do capture different aspects of the document set. And they provide a structured way to understand the set of documents. The salience scores demonstrate that the subtopics are not equally important. The salience estimation of subtopics aims to facilitate the summarization process focusing on important information and reduce the distraction from trivial content. As in Table 3, the extracted summary sentences only cover two most important subtopics and they come from different documents. Thus, many overlaps exist among different documents, which complement each other and form the final subtopics identified by our model.

However, redundancy occurs in the extracted summary, *e.g.,* both **d2s1t1** and **d8s3t1** mention "Internet.org app provide Internet services for free". Besides, incoherence also exists, *e.g.,* **d8s3t1** should be in front of **d3s4t2**. This is the inherent drawback of extractive summarization. Thus, there is still great room for improvement. We will deal with these issues in our future work.

## 6 Conclusion

In this paper, we propose an unsupervised MDS solution, which deals with both sequential context information in a single document and the correlations among documents. Hierarchical RNN is adopted to encode context information in sentence and document embeddings. We transform correlations among documents into subtopics expressed by sentences in different documents. Readers' comments are incorporated into subtopics formation to highlight important information in news. The subtopics provide us another angle to understand the underlying topic besides the document view. The sentence salience is estimated with subtopic salience and relative sentence salience in the hierarchical way as well. Extensive experiments show that the proposed model outperforms state-of-the-art baselines.

## References


Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Siddhartha Banerjee, Prasenjit Mitra, and Kazunari Sugiyama. 2015. Multi-document abstractive summarization using ILP based multi-sentence compression. In *IJCAI*, pages 1208–1214.

Lidong Bing, Piji Li, Yi Liao, Wai Lam, Weiwei Guo, and Rebecca J. Passonneau. 2015. Abstractive multi-document summarization via phrase selection and merging. In *ACL, Vol. 1*, pages 1587–1597.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *JMLR*, 3:993–1022.

Ziqiang Cao, Wenjie Li, Sujian Li, and Furu Wei. 2017. Improving multi-document summarization via text classification. In *AAAI*, pages 3053–3059.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*, pages 1724–1734.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555.

John M Conroy, Judith D Schlesinger, Jade Goldstein, and Dianne P Oleary. 2004. Left-brain/right-brain multi-document summarization. In *DUC 2004*.

Günes Erkan and Dragomir R. Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *JAIR*, 22:457–479.

Shu Gong, Youli Qu, and Shengfeng Tian. 2010. Subtopic-based multi-documents summarization. In *CSO*, volume 2, pages 382–386. IEEE.

Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *CVPR*, pages 1735–1742.

Aria Haghighi and Lucy Vanderwende. 2009. Exploring content models for multi-document summarization. In *HLT-NAACL*, pages 362–370.

Kai Hong, John M. Conroy, Benoît Favre, Alex Kulesza, Hui Lin, and Ani Nenkova. 2014. A repository of state of the art and competitive baseline summaries for generic news summarization. In *LREC*, pages 1608–1616.

Chris Kedzie, Fernando Diaz, and Kathleen R. McKeown. 2016. Real-time web scale event summarization using sequential decision making. In *IJCAI*, pages 3754–3760.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. 2015. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, volume 2.

Logan Lebanoff, Kaiqiang Song, and Fei Liu. 2018. Adapting the neural encoder-decoder framework from single to multi-document summarization. In *EMNLP*, pages 4131–4141.

Piji Li, Lidong Bing, and Wai Lam. 2017a. Reader-aware multi-document summarization: An enhanced model and the first dataset. In *NFiS@EMNLP*, pages 91–99.

Piji Li, Lidong Bing, Wai Lam, Hang Li, and Yi Liao. 2015. Reader-aware multi-document summarization via sparse coding. In *IJCAI*, pages 1270–1276.

Piji Li, Wai Lam, Lidong Bing, Weiwei Guo, and Hang Li. 2017b. Cascaded attention based unsupervised information distillation for compressive summarization. In *EMNLP*, pages 2081–2090.

Piji Li, Zihao Wang, Wai Lam, Zhaochun Ren, and Lidong Bing. 2017c. Salience estimation via variational auto-encoders for multi-document summarization. In *AAAI*, pages 3497–3503.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proc. of the ACL-04 workshop*, volume 8. Barcelona, Spain.

He Liu, Hongliang Yu, and Zhi-Hong Deng. 2015. Multi-document summarization based on two-level sparse representation model. In *AAAI*, pages 196–202.

Ulrike von Luxburg. 2007. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416.

Shulei Ma, Zhi-Hong Deng, and Yunlun Yang. 2016. An unsupervised multi-document summarization framework based on neural document model. In *COLING*, pages 1514–1523.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *EMNLP*, pages 404–411.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Ani Nenkova, Lucy Vanderwende, and Kathleen R. McKeown. 2006. A compositional context sensitive multi-document summarizer: exploring the factors that influence summarization. In *SIGIR*, pages 573–580.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *ICML*, pages 1310–1318.

Dragomir R Radev, Hongyan Jing, and Malgorzata Budzikowska. 2000. Centroid-based summarization of multiple documents: Sentence extraction, utility-based evaluation, and user studies. In *NAACL-ANLP, Vol. 4*, pages 21–30.

Pengjie Ren, Zhumin Chen, Zhaochun Ren, Furu Wei, Jun Ma, and Maarten de Rijke. 2017. Leveraging contextual sentence relations for extractive summarization using a neural attention model. In *SIGIR*, pages 95–104.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *ACL, Vol. 1*, pages 1073–1083.

Hongya Song, Zhaochun Ren, Shangsong Liang, Piji Li, Jun Ma, and Maarten de Rijke. 2017. Summarizing answers in non-factoid community question-answering. In *WSDM*, pages 405–414.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112.

Xiaojun Wan and Jianwu Yang. 2008. Multi-document summarization using cluster-based link analysis. In *SIGIR*, pages 299–306.

Kexiang Wang, Tianyu Liu, Zhifang Sui, and Baobao Chang. 2017. Affinity-preserving random walk for multi-document summarization. In *EMNLP*, pages 210–220.

Michihiro Yasunaga, Rui Zhang, Kshitijh Meelu, Ayush Pareek, Krishnan Srinivasan, and Dragomir R. Radev. 2017. Graph-based neural multi-document summarization. In *CoNLL*, pages 452–462.

Qingyu Zhou, Nan Yang, Furu Wei, Shaohan Huang, Ming Zhou, and Tiejun Zhao. 2018. Neural document summarization by jointly learning to score and select sentences. In *ACL*, pages 654–663.

# Referring Expression Generation Using Entity Profiles

**Meng Cao** **Jackie Chi Kit Cheung**

School of Computer Science, McGill University, Montreal, QC, Canada
MILA, Montreal, QC, Canada

{meng.cao@mail, jcheung@cs}.mcgill.ca

## Abstract

Referring Expression Generation (REG) is the task of generating contextually appropriate references to entities. A limitation of existing REG systems is that they rely on entity-specific supervised training, which means that they cannot handle entities not seen during training. In this study, we address this in two ways. First, we propose task setups in which we specifically test a REG system's ability to generalize to entities not seen during training. Second, we propose a profile-based deep neural network model, PROFILEREG, which encodes both the local context and an external profile of the entity to generate reference realizations. Our model generates tokens by learning to choose between generating pronouns, generating from a fixed vocabulary, or copying a word from the profile. We evaluate our model on three different splits of the WebNLG dataset, and show that it outperforms competitive baselines in all settings according to automatic and human evaluations.

## 1 Introduction

Entities can be expressed by various types of linguistic expressions, including by their names (e.g., *Barrack Obama*), a definite description (e.g., *the former president of the United States*), a pronoun (e.g., *he*, *him*, *his*), or a demonstrative (e.g., *that person*). Many factors play a role in determining what type of expression is appropriate in a particular context (Henschel et al., 2000), including information status, familiarity with the entity, and referential clarity. In this study, we aim to design a model that can generate appropriate referring expressions of entities in an extended passage. Such a system is useful in a variety of natural language generation settings, from dialogue systems to automatic summarization applications (Reiter and Dale, 2000; Krahmer and Van Deemter, 2012).

Referring expression generation (REG) can be broken into two steps. The first is to decide the *form* of referring expression; that is, what type of reference should be used (e.g., a proper name). The second is to determine the *content* of the referring expression (e.g., *Ernest Hemingway*).

Many computational approaches, both rule-based and machine-learning-based, have been proposed for REG. Rule-based models use predefined heuristics and algorithms to determine referential form (Reiter and Dale, 2000; Henschel et al., 2000; Callaway and Lester, 2002). Machine learning-based approaches require training on samples to predict referring expressions (Nenkova and McKeown, 2003; Greenbacker and McCoy, 2009; Ferreira et al., 2016, 2017), often as a classification task.

The common limitation of previous REG systems is that they are incapable of generating referring expressions for new, unseen entities. Previous REG setups have tended to focus on form selection rather than content generation, or else they provide a static list of attributes or realization options for models to select from (Belz et al., 2010; Gatt and Belz, 2008). The recent NeuralREG model generates referring expressions in an integrated, end-to-end setup, but it requires seeing instances of the entity being referred to in the training set (Ferreira et al., 2018).

In this work, we address this problem by proposing new REG task setups which test for REG systems' ability to handle new entities at test time. We also develop a REG system which can handle entities not seen during training using external knowledge about them, generated using extracts of their Wikipedia page.

From a practical perspective, it is reasonable to assume that such an entity profile exists for common entities such as popular locations and celebrities, as well as for targeted entities of interest that



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3163–3172, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

should be handled by an in-domain NLG system. We make the minimal assumption that the profile contains just a few sentences about the entity, so that these profiles can easily be written by non-experts or be automatically extracted.

Our model, PROFILEREG, uses a learned switch variable to decide whether to generate a token from a fixed vocabulary, generate a pronoun, or use information from the profile in order to generate appropriate referring expressions in context. We evaluate our model on the WebNLG corpus (Gardent et al., 2017a). Experimental results show that our model is capable of handling unseen entities that prior work simply cannot handle in our new evaluation setups, while also outperforming them in the original setting.[1]

Our contributions are as follows. First, we address an important limitation in prior REG studies by creating new test setups that evaluate neural REG models specifically on entities that are not seen in the training set. Second, we propose a new REG model, PROFILEREG, which outperforms existing REG models in all tested settings according to automatic and human evaluation measures.

## 2 Related Work

Previous work in REG can be divided into two groups: rule-based and machine learning-based.

Rule-based approaches use pre-defined rules or algorithms to determine the form or content of generated referring expressions. For example, Reiter and Dale (2000) proposed a straightforward heuristic for REG: a proper name should be used in the first mention of an entity and a pronominal form should be used for the subsequent references if there is no mention of any other entity of the same gender between the reference and its antecedent. Henschel et al. (2000) presented an algorithm for the pronominalization of third person discourse entities.

Machine learning-based approaches predict the form or content of referring expressions using features extracted from the context. Commonly used context features include syntactic position, referential status (initial or a subsequent mention) and distance. For instance, Nenkova and McKeown (2003) built a Markov chain model to capture how the subsequent referring expressions are conditioned by earlier mentions. Frank and Goodman (2012) proposed a rational speaker-listener model based on the assumptions that speakers attempt to be informative and that listeners use Bayesian inference to recover the speakers' intended referents. Orita et al. (2015) extended the previous model by introducing the referent's discourse salience and a cost term. Ferreira et al. (2016) proposed a naive Bayes model to predict the probability of a particular referential form given a set of feature values, including syntactic position, referential status and recency of references. Similarly, Ferreira et al. (2017) used a naive Bayes model based on syntactic position and referential status to predict the form of a proper name reference.

In recent years, deep neural networks have achieved great success in a variety of NLP applications (e.g., machine translation (Bahdanau et al., 2014) and automatic summarization (Rush et al., 2015; See et al., 2017)). To the best of our knowledge, there have only been two models that use deep neural networks for REG. The first is by Ferreira et al. (2016), who use recurrent neural networks (RNNs) to predict the form of referring expressions. They use RNNs to encode a sequence of discourse features and apply a softmax layer to generate referential form distribution.

The other is the NeuralREG model proposed by Ferreira et al. (2018), an end-to-end system that predicts both the form and content of the referring expression. NeuralREG has an encoder-decoder structure, using LSTM units to encode contextual information and to decode referring expressions. Since each entity is represented as an embedding vector in the model, it cannot handle new entities that are outside the training set.

Generating entity descriptions based on source data has also been explored in the context of concept-to-text generation Lebret et al. (2016). We focus here on generating referring expressions from unstructured source data, which may be more readily available in some settings.

## 3 Data

The WebNLG corpus (Gardent et al., 2017a) was initially introduced in the WebNLG challenge 2017 (Gardent et al., 2017b). The WebNLG dataset is made up of (data, text) pairs where the data is a set of triples that consists of entities and their relationships and the text is a verbalisation of these triples. In this challenge, the participants are asked to map the triple data to text.

In this study, we use a delexicalized version

[1] https://github.com/mcao610/ProfileREG

| |
|---|
| **Triples**:<br>Elliot_See \| almaMater \| University_of_Texas_at_Austin<br>Elliot_See \| deathPlace \| St._Louis<br>Elliot_See \| birthPlace \| Dallas<br>...<br>**Text**: Elliot See was born on July 23 , 1927 in Dallas . *He* attended the U of Texas at Austin which is part of the U of Texas system . See was a test pilot . He died on Feb 28 , 1966 in St. Louis . |
| **Wikipedia ID**: elliot_see<br>**Referring expression**: He<br>**Delexicalized text**: elliot_see was born on 1927-07-23 in dallas . elliot_see attended the university_of_texas_at_austin which is part of university_of_texas_system . elliot_see was a test pilot . elliot_see died on 1966-02-28 in st._louis .<br>**Pre-context**: elliot see was born on 1927-07-23 in dallas .<br>**Post-context**: attended the u of texas at austin which is part of the U of texas system . see was a test pilot . he died on 1966-02-28 in st. louis . |

Table 1: A sample in the original WebNLG dataset (above) and the delexicalized dataset (below). In this case, referring expression "He", which refers to *Elliot_See*, is first delexicalized as elliot_see and then extracted from the text.

of WebNLG corpus introduced by Ferreira et al. (2018). The authors manually extracted each referring expression in the text and kept only the ones referring to Wikipedia entities. As a result, each sample of the delexicalized dataset consists of a Wikipedia ID, a true-cased tokenized referring expression (extracted from the text) and a lowercased, tokenized discourse context preceding and following the target reference (referred as the pre- and post-context). Table 1 shows one sample from the corpus. The dataset consists of 78,901 referring expressions for 1,501 Wikipedia entities. Among the expressions, 71.4% (56,321) are proper names, 5.6% (4,467) are pronouns, 22.6% (17,795) are descriptions and 0.4% (318) are demonstratives.

In the delexicalized version of WebNLG corpus, references to other discourse entities in the pre- and pos-contexts are represented by their one-word ID removing quotes and replacing white spaces with underscores. Since our model does not rely on any entity ID, we replaced each ID with its corresponding referring expression.

### 3.1 Profile Generation

We create a profile for each entity that appears in the corpus as a form of external knowledge for REG. The profile consists of the first three sentences of the entity's Wikipedia page, which is retrieved automatically using the entity's Wikipedia ID and the Python wikipedia 1.4.0 library. We choose the first three sentences of the Wikipedia page since they usually contain the most important information about an entity such as its name, country, or occupation.

We tokenized the crawled Wikipedia sentences and removed all phonetic symbols as well as other special characters. For the Wikipedia IDs that are ambiguous (e.g., New York may refer to New York City or New York (State)), we manually checked the discourse context to determine which entry was appropriate. For the entities that are constant numbers or dates which have no meaningful profiles, we make the profile content the same as the entity ID but replaced each underscore in the ID with white spaces. We generated a total of 1,501 profiles, one per entity in the dataset.

### 3.2 Setups

In the current way that WebNLG is set up for REG, entities in the training set and test set are overlapped, which makes it impossible to examine the effect that previously unseen entities have on REG performance. To address this limitation, we split the dataset in three different ways as follows:

**Original.** The original split of data from Ferreira et al. (2018). There are 63,061, 7,097 and 8,743 referring expressions in the training, development and test set respectively. In this dataset, all entities in the development and test set also appear in the training set.

**Entity-separated.** We separated the entities in the training, development and test set by splitting the delexicalized WebNLG dataset by entity IDs. The ratio of the number of entities in the training set, validation set, and test set is 8:1:1. As a result, there are 63,840, 7,978 and 7,083 referring expressions of different entities in each dataset.

**Random.** We randomly split all referring expressions in WebNLG with ratio 8:1:1. The final training, development and test set contains 63,121, 7,890 and 7,890 referring expressions respectively. In this dataset, some entities in the test set appear in training, but not all. We also test our models on this dataset since we believe it better reflects a realistic setting in which a REG model might be applied.

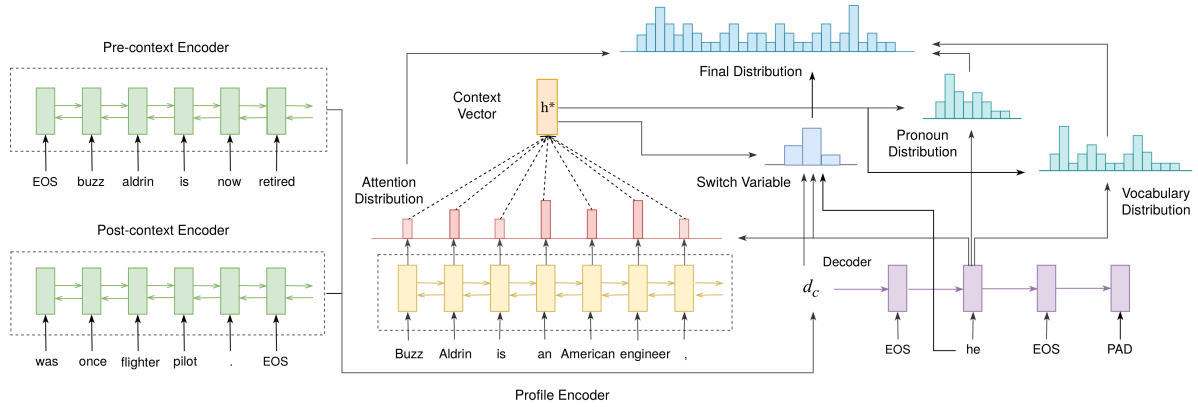


Figure 1: Our PROFILEREG model. The model uses two bi-directional LSTMs (green) to encode the pre- and post-context and one bi-directional LSTM (yellow) to encode the profile. At the decoding step, we calculate three distributions: attention distribution, pronoun distribution and vocabulary distribution. These three distributions together with the switch variable are used to determine the final distribution.

## 4 Model

In this section, we present our model, PROFILEREG (Figure 1). In our task setup, the input of the model consists of pre- and post-context texts of lengths $H$ and $N$, and an entity profile of length $L$. The output is a sequence of tokens that make up the referring expression. We use three bidirectional LSTM (Hochreiter and Schmidhuber, 1997) encoders to encode the pre- and post-context and the profile, and a unidirectional LSTM decoder to generate the output at inference time. We also apply an attention mechanism and a switch variable to calculate the final output distribution. We now introduce each component in more detail.

### 4.1 Context Encoders

We use two bi-directional LSTMs to encode the pre- and post-contexts, which are the entire sequences of tokens before and after the referring expression in the discourse context. The pre-context, a sequence of tokens $\{w_1^{(pre)}, ..., w_H^{(pre)}\}$, is first mapped to a distributed vector space through an embedding layer, resulting in a sequence of vectors $\{x_1^{(pre)}, ..., x_H^{(pre)}\}$. Then, the word embeddings are given to a bidirectional LSTM. We call the forward and backward hidden-state vectors at time $t$ $\overrightarrow{h}_t^{(pre)}$ and $\overleftarrow{h}_t^{(pre)}$ respectively. We extract the two hidden-state vectors at the last step time $H$, and concatenate them to form the final vector representation $h_H^{(pre)} = [\overrightarrow{h}_H^{(pre)}; \overleftarrow{h}_H^{(pre)}]$, which summarizes the information in the entire pre-context. The same process is repeated for the post-context using another bi-directional LSTM. This results in $h_N^{(post)} = [\overrightarrow{h}_N^{(post)}; \overleftarrow{h}_N^{(post)}]$. Finally, we concatenate the pre- and post-context vector representations and pass them through a hidden layer:

$$d_c = \tanh(W_d[h_H^{(pre)}; h_N^{(post)}]) \quad (1)$$

where $W_d$ is a weight matrix mapping the concatenated hidden state vectors into a joint vector representation $d_c$, which is used as the initial state of the decoder.

### 4.2 Profile Encoder

The profile encoder is another bi-directional LSTM that receives a series of profile words $\{w_1, ..., w_L\}$ as input. All words in the profile are first converted to lowercase and mapped to a sequence of word embeddings $\{x_1, ..., x_L\}$ through the same embedding layer as for the pre- and post-contexts. Then the embedded sequence are passed through the bi-directional LSTM, resulting in forward and backward hidden states $\{\overrightarrow{h}_1, ..., \overrightarrow{h}_L\}$ and $\{\overleftarrow{h}_1, ..., \overleftarrow{h}_L\}$. At each word position $t$, the forward and backward representations are concatenated as the final representation $h_t = [\overrightarrow{h}_t; \overleftarrow{h}_t]$.

We also calculate a character-level embedding for each word using another bidirectional LSTM encoder. The last hidden-state vector of the forward and backward LSTMs are concatenated and given to a separate non-linear layer to form an output vector. This output vector is concatenated with the word-level embedding as our final word representation.

### 4.3 Attention-based Decoder

Our referring expression decoder is a unidirectional LSTM. At each decoding step $t$, the decoder receives the word embedding of the previous word and generates a decoder hidden state $s_t$. Then, we calculate the attention distribution $a_t$ over the entity profile using the attention mechanism introduced by Bahdanau et al. (2014):

$$e_i^t = v^T \tanh(W_h h_i + W_s s_t + b_{attn}) \quad (2)$$

$$a^t = \text{softmax}(e^t), \quad (3)$$

where $h_i$ is the sequence of profile encoder hidden states, $W_h, W_s, v^T$ and $b_{attn}$ are all trainable parameters. The attention distribution, $a^t$, determines the degree of contribution of each token in the profile at expression generation step $t$. A final summary $h_t^*$ is calculated by summing the profile encoder states $h_i$ weighted by the attention probabilities $a^t$:

$$h_t^* = \sum_i a_i^t h_i \quad (4)$$

Then, $h_t^*$ is concatenated with the decoder hidden state $s_t$ and passed through a linear layer, producing the vocabulary distribution $P_{voc}$:

$$P_{voc} = \text{softmax}(V[s_t; h_t^*] + b) \quad (5)$$

where $V$ and $b$ are both trainable parameters in the model. $P_{voc}$ contains the generation probability of each word in the vocabulary. We will actually modify this distribution to account for out-of-vocabulary items and pronouns by introducing a copying mechanism, which we will discuss in the next section. This results in the final generation probability, $P(w_t^*)$ of the target (gold-standard) word at time $t$.

During training, we take the negative log likelihood of the target word in the gold standard, $\hat{w}_t^*$, as the loss of the model at step $t$. The overall loss is the sum of the loss at each time step:

$$\text{loss} = \frac{1}{T} \sum_{t=0}^{T} -\log P(\hat{w}_t^*) \quad (6)$$

At test time, the word with the highest generation probability will be the output at the current time step and the input to the next step.

### 4.4 Switch Mechanism

In PROFILEREG, we apply a switch mechanism that allows the model to generate different referential forms, inspired by the pointer-generator network of See et al. (2017). In particular, our model can choose to: i) **copy** a word from the profile, which is especially useful for named entities with rare names which are not part of a fixed vocabulary, ii) generate a **pronoun**, or iii) **generate** from a fixed vocabulary. Although pronouns are part of our fixed vocabulary, we choose to distinguish them as a separate category to model, due to their importance and frequency in the REG task.

We define a switch variable $\Sigma$ which can take on one of three values, COPY, PRO, and GEN, corresponding to the three decoder actions. Let the associated probabilities be $\sigma_{\text{COPY}}$, $\sigma_{\text{PRO}}$, and $\sigma_{\text{GEN}}$. After computing $h_t^*$ and vocabulary distribution $P_{voc}$ at time step $t$, we compute these probabilities using the summary vector $h_t^*$, the encoder final state $d_c$, the decoder hidden state $s_t$ and the input $x_t$:

$$[\sigma_{\text{COPY}}, \sigma_{\text{PRO}}, \sigma_{\text{GEN}}] = \text{softmax}( H_{h^*} h_t^* + H_d d_c + H_s s_t + H_x x_t + b_s), \quad (7)$$

where $H_{h^*}, H_d, H_s, H_x$ and $b_s$ are learnable parameters with output dimension of three. The final generation probability of the target word $w_t^*$ is then computed as follows:

**if** $w_t^*$ is a pronoun **then**

$$P(w_t^*) = \sigma_{\text{PRO}} P_{voc}(w_t^*) + \sigma_{\text{COPY}} \sum_{i:w_i=w_t^*} a_i^t$$

**else if** $w_t^*$ is a vocabulary word **then**

$$P(w_t^*) = \sigma_{\text{GEN}} P_{voc}(w_t^*) + \sigma_{\text{COPY}} \sum_{i:w_i=w_t^*} a_i^t$$

**else**

$$P(w_t^*) = \sigma_{\text{COPY}} \sum_{i:w_i=w_t^*} a_i^t$$

**end if**

In the formula, $a^t$ is the attention distribution computed in the previous section. When $w_t^*$ is a pronoun or vocabulary word, the probability $P(w_t^*)$ consists of two terms since the word in the vocabulary could also be in the profile. $P(w_t^*)$ will be used in the Equation 6.

One important difference between our model and the pointer-generator network is that the pointer-generator model only computes one *generation probability* $p_{gen} \in [0, 1]$, which is used to decide whether generate a word or copy a token from the source document. In our model, however, we compute a probability distribution $S_{switch}$ over three values. Our intention is that the model can better distinguish between different forms of referring expressions, especially between pronouns and demonstrative or definite description forms.

## 5 Experiments

We evaluate our model and the baselines on three different data splits mentioned in Section 3.2: **Original**, **Entity-separated** and **Random**, where the entities are overlapped, all separated and randomly mixed in training, validation and test set.

### 5.1 Baselines

We compared our model against three baselines: *OnlyName*, *Ferreira* and *NeuralREG*. In our first *OnlyName* baseline, we use the name of the entity as the final realization regardless of the context. This baseline is inspired by the fact that 71.4% of the referring expressions in WebNLG are proper names. Given the Wikipedia profile of an entity, we obtain the name of the entity by extracting the first few capitalized words in that profile. If the name cannot be found in this way, we will use the first word of the profile (except articles).

Our second baseline makes use of the Naive Bayes model proposed by Ferreira et al. (2016), dubbed *Ferreira*. This model takes features including syntactic position, referential status (initial or a subsequent mention at the level of text and sentence) and recency (distance between references) as input. The original model is only used to determine the form of the referring expression. We adapt the model by adding a content generation component. Once the form of the referent is decided, the content of the referential expression is decided as follows: we use the *OnlyName* model to get the name of the entity and find the most frequent pronoun expressions in the profile as the content for pronoun form (using "it" if no pronoun expressions can be found). We use the gold-standard content for the demonstrative and description categories from the WebNLG dataset, which provides an upper bound on the potential performance of this system.

Our third baseline is the NeuralREG model proposed by Ferreira et al. (2018). NeuralREG is a sequence-to-sequence neural network model with an attention mechanism. All words and entities in NeuralREG are represented as embeddings which are randomly initialized and learned during training. We tested the best-performing CAtt version of NeuralREG.

### 5.2 Metrics

Following prior work, we calculate the overall accuracy and the String Edit Distance (SED) between the generated and the reference expressions. The overall accuracy is calculated by comparing the generated string with the real referring expression. We post-processed the output to ignore differences in capitalization, accent and certain non-alphabetic characters (e.g., "Desteaptate" vs. "Deşteaptă-te", and "Tranmere Rovers FC" vs. "Tranmere Rovers F.C.").

We also evaluate the performance of our model with regard to a particular referential form. We computed name accuracy, pronoun accuracy, precision, recall and F1-score. The name and pronoun accuracy are computed by comparing names and pronouns in the test set with the generated referring expressions. Pronoun classification evaluation takes all generated referring expressions as two forms: pronoun or non-pronoun and do not consider the actual content.

### 5.3 Experiment Setting

For all experiments, our model has 100-dimensional hidden states for the encoder and decoder LSTMs and 50 dimensional for the character LSTM. The word and character embeddings are both set to 100-dimensional. We initialize the word embeddings using pre-trained GloVe vectors (Pennington et al., 2014) and all character embeddings are randomly initialized. The model is trained for up to 35 epochs, or until the performance on the development set does not improve for 5 epochs. We set the learning rate to 0.0037 and the batch size to 64. We apply gradient clipping with a maximum gradient norm of 5. We adopt dropout training with a dropout rate of 0.5. At test time, all referring expressions are generated using greedy decoding. These settings were selected by tuning on the development set.

## 6 Results

### 6.1 Automatic Evaluation

As shown in Table 2 PROFILEREG outperforms the three baselines in all experiments. Our model in particular excels at the measures of pronoun accuracy and pronoun classification.

On the **Original** dataset, our model is slightly better than NeuralREG in total accuracy and SED. This result indicates that our model is as good as or better than NeuralREG model when dealing with previously seen entities. On pronoun classification, *OnlyName* model has a performance of zero since it cannot generate any pronouns.

| Dataset | Models | SED | Total Acc. | Name Acc. | Pronoun Acc. | Pronoun Classif. | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | Prec. | Rec. | F1 |
| **Original** | OnlyName | 5.57 | 53.47% | 66.22% | 0.00% | 0.00 | 0.00 | 0.00 |
| | Ferreira | 4.56 | 56.50% | 65.11% | 63.58% | 0.60 | 0.96 | 0.74 |
| | NeuralREG | 2.33 | 74.13% | **81.64%** | 74.92% | 0.77 | 0.78 | 0.78 |
| | PROFILEREG | **2.22** | **74.67%** | 80.51% | **87.17%** | 0.72 | 0.89 | **0.80** |
| **Entity-separated** | OnlyName | 6.28 | 54.71% | 61.44% | 0.00% | 0.00 | 0.00 | 0.00 |
| | Ferreira | 5.91 | 55.67% | 60.11% | 64.25% | 0.45 | 0.97 | 0.61 |
| | NeuralREG | – | – | – | – | – | – | – |
| | PROFILEREG | **5.17** | **60.60%** | **67.20%** | **75.57%** | 0.53 | 0.80 | **0.64** |
| **Random** | OnlyName | 5.59 | 53.21% | 67.02% | 0.00% | 0.00 | 0.00 | 0.00 |
| | Ferreira | 4.72 | 55.63% | 65.59% | 61.37% | 0.58 | 0.95 | 0.72 |
| | NeuralREG | 2.61 | 71.62% | 77.49% | 75.03% | 0.61 | 0.89 | 0.71 |
| | PROFILEREG | **2.11** | **75.33%** | **81.85%** | **83.89%** | 0.76 | 0.85 | **0.80** |

Table 2: REG performance on the original, entity separated and random dataset. PROFILEREG is the model proposed in this work. Note that NeuralREG cannot be applied to the entity-separated dataset.

**Entity-separated** demonstrates our model's ability to handle new entities, compared to the baselines. There is a clear drop on most metrics compared to the original dataset, but this is expected since the test set consists of unseen entities. As previously explained, the NeuralREG model cannot be applied to this dataset without extensive modifications. On **Random**, PROFILEREG outperforms NeuralREG. For the unseen entities, NeuralREG generates a random string, since the embeddings of unseen entities are not updated by training. Our model, however, is capable of producing accurate referring expressions for both previously mentioned and new entities.

Table 3 shows an example of the generated text of each model. The entity in this example is *Acharya Institute of Technology*. Note that only the bold text is generated by the models. The output of the *OnlyName* model is clearly the least readable one.

### 6.2 Seen Entities vs. Unseen Entities

In the evaluation, we also distinguished the results for seen and unseen entities. We trained the model on a training set contains 64,353 referring expressions and evaluate the model on a test set with 3,591 referring expressions related to seen entities and 2,955 expressions related to unseen entities. Table 4 shows the evaluation results. From Table 4, it is easy to see that the model performs better when generating referring for seen entities. Among four referring expression types, the accuracy of description type drops dramatically for unseen entities, from 48.72% to 20.54%. This is probably due to the fact that compared with name and pronoun, description type is often hard to identify and more flexible. For instance, one of the gold-standard descriptions in the test set is *the comic character , amazing-man*. The model's generation for this referring expression is *amazing-man*.

Also notice that the accuracy of demonstrative type is zero for both seen and unseen entities. We think this is because the dataset is unbalanced and there are too few demonstrative samples in the training set. The model did not have enough samples to learn to identify demonstratives.

### 6.3 Human Evaluation

**Method.** In addition to automatic evaluation, we also performed a human evaluation, comparing our model to the original text and to the three baseline models. We concatenated the pre- and post-contexts with the generated referring expression to form an overall passage. We showed participants the output of our model and a comparison model, and asked them to make a pairwise preference judgment between the two passages.

Specifically, we randomly selected 100 samples from the **Random** test set. We presented the two

| |
|---|
| *Original*: **Acharya Institute of Technology** is affiliated with Visvesvaraya Technological University which is in Belgium . **The institute itself** is in India 's Karnataka state and **its** full address is In Soldevanahalli , Acharya Dr. Sarvapalli Radhakrishnan Road , Hessarghatta Main Road , Bangalore – 560090 . **It** was created in 2000 and **its** director is Dr. G. P. Prabhukumar . |
| *OnlyName*: **Acharya Institute of Technology** is affiliated with Visvesvaraya Technological University which is in Belgium . **Acharya Institute of Technology** is in India 's Karnataka state and **Acharya Institute of Technology** full address is In Soldevanahalli , Acharya Dr. Sarvapalli Radhakrishnan Road , Hessarghatta Main Road , Bangalore – 560090 . **Acharya Institute of Technology** was created in 2000 and **Acharya Institute of Technology** director is Dr. G. P. Prabhukumar . |
| *Ferreira*: **Acharya Institute of Technology** is affiliated with Visvesvaraya Technological University which is in Belgium . **It** is in India 's Karnataka state and **its** full address is In Soldevanahalli , Acharya Dr. Sarvapalli Radhakrishnan Road , Hessarghatta Main Road , Bangalore - 560090 . **It** was created in 2000 and **it** director is Dr. G. P. Prabhukumar . |
| *NeuralREG*: **Acharya institute of technology** is affiliated with Visvesvaraya Technological University which is in Belgium . **It** is in India 's Karnataka state and **its** full address is In Soldevanahalli , Acharya Dr. Sarvapalli Radhakrishnan Road , Hessarghatta Main Road , Bangalore – 560090 . **It 's** was created in 2000 and **its** director is Dr. G. P. Prabhukumar . |
| PROFILEREG: **The Acharya Institute of Technology** is affiliated with Visvesvaraya Technological University which is in Belgium . **The institute** is in India 's Karnataka state and **its** full address is In Soldevanahalli , Acharya Dr. Sarvapalli Radhakrishnan Road , Hessarghatta Main Road , Bangalore – 560090 . **It** was created in 2000 and **its** director is Dr. G. P. Prabhukumar . |
| *(Entity profile)* Acharya Institute of Technology , or AIT , is a private co-educational engineering and management college in Bengaluru , India , affiliated with the Visvesvaraya Technological University and accredited by the National Board of Accreditation . Established in 2000 , it offers eleven undergraduate courses and eight postgraduate courses . The college has links and collaborations with various industries and universities across the world . |

Table 3: Example of original text and generated text of each model.

| | Type | Acc. | Support |
|---|---|---|---|
| **Seen** | demonstrative | 0.00% | 22 |
| | description | 48.72% | 862 |
| | name | 79.11% | 2547 |
| | pronoun | 90.00% | 160 |
| | total | 71.82% | 3591 |
| **Unseen** | demonstrative | 0.00% | 3 |
| | description | 20.54% | 409 |
| | name | 74.74% | 2423 |
| | pronoun | 88.33% | 120 |
| | total | 67.72% | 2955 |

Table 4: Evaluation for seen and unseen entities.

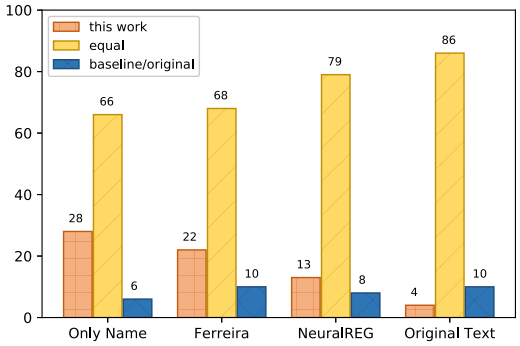


Figure 2: Human evaluation results. Comparison of our model with three baselines and original text.

passages in random order, asking participants to consider the readability, grammaticality and fluency of texts. We then asked them to select the passage that they prefer, or to say they are equally good. We recruited 20 participants, each evaluating 20 samples.

**Results.** Figure 2 shows the human evaluation results. The most common choice made by participants is "equal". This is because many generated referring expressions by the models are actually identical. When our model was compared to the original text, 86 of the 100 generated texts are identical or very similar to the original text. This demonstrates our model's ability to generate high quality referring expressions. Also, we can see from the chart that our model tends to be preferred over the three baseline models.

### 6.4 Analyzing Switch Variable Behaviour

We now examine the behaviour of the switch variable, $\Sigma$. As mentioned in Section 4.4, each word is generated with a probability distribution ($\sigma_{\text{COPY}}$, $\sigma_{\text{PRO}}$ and $\sigma_{\text{GEN}}$). We expect this distribution to vary based on the form of the referring expression. For instance, when a proper name expected, $\sigma_{\text{COPY}}$ should be relatively high if the model copies words from the entity profile.

Table 5 shows three examples of generated referring expressions with the values of the switch

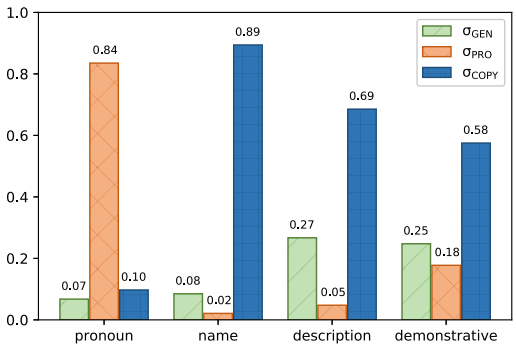


Figure 3: The average probabilities of the switch variable values for each referential form on the **Random** test set.

| |
|---|
| **Context**: The 3rd runway at *Ardmore Airport ( New Zealand )* is made of Poaceae . (...)<br>**Entity Profile**: Ardmore Airport is an airport 3 NM south-east of Manurewa in Auckland , New Zealand . (...)<br>**Realization**: Ardmore(COPY 0.968) Airport(COPY 0.971) in(GEN 0.908) New(COPY 0.969) Zealand(COPY 0.947) |
| **Context**: Curitiba is part of Parana State in the South Region , Brazil and is served by Afonso Pena International airport . It is led by *the Democratic Labour Party* .<br>**Entity Profile**: The Democratic Labour Party is a social democratic political party in Brazil . (...)<br>**Realization**: the(GEN 0.895) Democratic(COPY 0.938) Labour(COPY 0.987) Party(COPY 0.966) |
| **Context**: Elliot See was born on July 23 , 1927 in Dallas . *He* attended the U of Texas at Austin which is part of the U of Texas system . (...)<br>**Entity Profile**: Elliot See was an American engineer , naval aviator , test pilot , and NASA astronaut . He was selected for NASA 's second group of astronauts in 1962 .<br>**Realization**: he(PRO 0.987) |

Table 5: An example of how the switch variable works in referring expression generation.

variable. We put the gold-standard referring expression, pre- and post-context together in Context. The words in yellow, green and orange highlight are copied words, vocabulary words and generated pronouns respectively and each word is followed by the corresponding switch value. These examples show that our model is able to switch between the three actions to generate a coherent, fluent referring expression. We further show the average value of the switch variable on the **Random** test set separated by the gold-standard referential form (Figure 3).

## 7 Conclusions

In this work, we have proposed new test setups on WebNLG dataset that evaluate the performance of REG models when dealing with previously unseen entities. We also introduced PROFILEREG, an end-to-end deep neural network model for REG. Unlike previous REG models that only model the local context, PROFILEREG incorporates both the context information and an entity profile that allows it to generate references for unseen entities. Our results show that PROFILEREG can accurately generate referring expressions for both new and previously mentioned entities, outperforming three strong baselines. One future direction is to model larger contexts with this approach. It would also be interesting to integrate our model into an existing language generation system (e.g., as part of an abstractive summarization system).

## Acknowledgments

We would like to thank the reviewers for their valuable comments. This research was supported by the Mitacs Globalink Program and the Canada CIFAR AI Chair program.

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Anja Belz, Eric Kow, Jette Viethen, and Albert Gatt. 2010. Generating referring expressions in context: The GREC task evaluation challenges. In *Empirical Methods in Natural Language Generation*, pages 294–327. Springer.

Charles B Callaway and James C Lester. 2002. Pronominalization in generated discourse and dialogue. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 88–95. Association for Computational Linguistics.

Thiago Castro Ferreira, Emiel Krahmer, and Sander Wubben. 2016. Towards more variation in text generation: Developing and evaluating variation models for choice of referential form. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 568–577.

Thiago Castro Ferreira, Emiel Krahmer, and Sander Wubben. 2017. Generating flexible proper name references in text: Data, models and evaluation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 655–664.

Thiago Castro Ferreira, Diego Moussallem, Ákos Kádár, Sander Wubben, and Emiel Krahmer. 2018.

NeuralREG: An end-to-end approach to referring expression generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1959–1969.

Michael C Frank and Noah D Goodman. 2012. Predicting pragmatic reasoning in language games. *Science*, 336(6084):998–998.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017a. Creating training corpora for NLG micro-planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 179–188.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017b. The WebNLG challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133.

Albert Gatt and Anja Belz. 2008. Attribute selection for referring expression generation: New algorithms and evaluation methods. In *Proceedings of the Fifth International Natural Language Generation Conference*, pages 50–58. Association for Computational Linguistics.

Charles F Greenbacker and Kathleen F McCoy. 2009. Feature selection for reference generation as informed by psycholinguistic research. In *Proceedings of the CogSci 2009 Workshop on Production of Referring Expressions (PRE-Cogsci 2009)*.

Renate Henschel, Hua Cheng, and Massimo Poesio. 2000. Pronominalization revisited. In *Proceedings of the 18th conference on Computational linguistics-Volume 1*, pages 306–312. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Emiel Krahmer and Kees Van Deemter. 2012. Computational generation of referring expressions: A survey. *Computational Linguistics*, 38(1):173–218.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1203–1213, Austin, Texas. Association for Computational Linguistics.

Ani Nenkova and Kathleen McKeown. 2003. References to named entities: A corpus study. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: companion volume of the Proceedings of HLT-NAACL 2003–short papers-Volume 2*, pages 70–72. Association for Computational Linguistics.

Naho Orita, Eliana Vornov, Naomi Feldman, and Hal Daumé III. 2015. Why discourse affects speakers' choice of referring expressions. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1639–1649.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Ehud Reiter and Robert Dale. 2000. *Building natural language generation systems*. Cambridge university press.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

# Exploring Diverse Expressions for Paraphrase Generation

**Lihua Qian**[1], **Lin Qiu**[1], **Weinan Zhang**[1], **Xin Jiang**[2], **Yong Yu**[1]
[1]Shanghai Jiao Tong University
{qianlihua, lqiu, wnzhang, yyu}@apex.sjtu.edu.cn
[2]Noah's Ark Lab, Huawei Technologies
jiang.xin@huawei.com

## Abstract

Paraphrasing plays an important role in various natural language processing (NLP) tasks, such as question answering, information retrieval and sentence simplification. Recently, neural generative models have shown promising results in paraphrase generation. However, prior work mainly focused on single paraphrase generation, while ignoring the fact that diversity is essential for enhancing generalization capability and robustness of downstream applications. Few works have been done to solve diverse paraphrase generation. In this paper, we propose a novel approach with two discriminators and multiple generators to generate a variety of different paraphrases. A reinforcement learning algorithm is applied to train our model. Our experiments on two real-world datasets demonstrate that our model not only gains a significant increase in diversity but also improves generation quality over several state-of-the-art baselines.

## 1 Introduction

Paraphrases refer to texts with identical meaning but expressed in different ways. Paraphrase generation has a wide range of applications. For example, paraphrases can be used to extend the reference texts for the automatic evaluation of text generation tasks, which make the evaluation more robust and accurate (Madnani and Dorr, 2010). In addition to increasing reference texts, paraphrases can also augment training data for text classification problems, especially for tasks lack of training data. Besides, paraphrases can be used as variations of queries for information retrieval and question answering systems. Query paraphrasing can narrow the gap between system comprehension and user questions (Fader et al., 2014; Yin et al., 2015). And systems can obtain better results from multiple feedbacks for different query variants.

Traditional paraphrase generation approaches leverage manual-crafted rules (McKeown, 1983; Hassan et al., 2007). In recent years, sequence to sequence learning techniques have achieved improvements in a variety of NLP tasks, such as machine translation (Bahdanau et al., 2014; Cho et al., 2014), text summarization (Nallapati et al., 2016) and dialogue systems (Serban et al., 2017). Specifically, the paraphrase generation can also be formulated as a sequence to sequence problem. Existing neural paraphrase generation works (Prakash et al., 2016; Gupta et al., 2018; Li et al., 2018) also demonstrate the effectiveness of deep learning method.

Although prior paraphrase generation methods mainly consider only one single output for each input, the nature of paraphrasing indicates that we can paraphrase one sentence into several different sentences. Therefore, we hope to generate a variety of paraphrases while ensuring quality. However, most of the previous work dedicated to improving the quality of top generated sentence, but did not conduct in-depth research on the diversity of paraphrase generation. In fact, diversity is very important in many applications. More diverse paraphrases will be beneficial to robustness and accuracy of automatic text evaluation, text classification, and can avoid the blandness caused by repetitive patterns.

A straightforward way to generate multiple different sentences is using beam search to select the top k sentences. However, beam search generates suboptimal sentences, and the resulting sentences are very similar to each other. Gupta et al. (2018) employ a variational auto-encoder framework to produce multiple sentences according to different noise input. Xu et al. (2018b) learn a shared decoder with different decoder embeddings for various decoding pattern. Although we can get multiple outputs with different noise inputs or de-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3173–3182,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

coder embeddings, different outputs may have slight changes or even no difference.

In order to address the problem, in this paper, we utilize multiple generators to generate diverse paraphrases without sacrificing quality. Reinforcement learning makes it possible for every generator to explore different generation pattern, breaking the restriction of learning the same target sentences given by data in supervised learning. Thus, we design a reinforcement learning algorithm to guide the models learning procedure. The proposed framework contains two discriminators. One discriminator is to examine whether two sentences convey the same meaning, the other one is to distinguish the sentence is generated by which generator. The higher the confidence of one text be judged as generated by a certain generator is, indicating the more obvious the differences between that text and texts generated by other generators are. We combine the scores of the two discriminators as guiding signals to the generators using reinforcement learning. Therefore, generators can learn to synthesize texts with apparent differences and good quality.

Our approach can automatically discover the solution to diverse paraphrase generation through the jointly learning of the discriminators and generators. And the model uses the same data pairs as in learning mono-paraphrase generation without extra annotations. Experiments on two real-world datasets show that our model improves both in quality and diversity. There are mainly two contributions in our work: the first one is proposing a novel framework as well as a corresponding learning procedure to generate various paraphrases, the second one is performing experiments demonstrating that our method achieves improvement regarding both generation diversity and quality.

## 2 Related Work

**Paraphrase generation** In the early years, various traditional techniques have been developed to solve the paraphrase generation problem. McKeown (1983) makes use of manually defined rules. Hassan et al. (2007) use thesaurus-based method for lexical substitution. And statistical machine translation (SMT) has also been used. Quirk et al. (2004) train SMT tools on a large number of sentence pairs collected from newspapers. Wubben et al. (2010) propose a phrase-based SMT model trained on aligned news headlines. Zhao et al. (2008) utilize multiple resources to strengthen a log-linear SMT model. Recently, deep neural models have also been applied to paraphrase generation due to their great success on natural language processing tasks. Prakash et al. (2016) design a deep stacked network with residual connections, Gupta et al. (2018) propose a conditional variational auto-encoder which can produce multiple paraphrases, Iyyer et al. (2018) learn a model to generate syntactically controlled paraphrase, Li et al. (2018) propose a generator-evaluator architecture coupled with deep reinforcement learning. Although being similar to the architecture proposed in (Li et al., 2018), our framework targets a different goal. Our work focuses on generating multiple diverse paraphrases, while theirs dedicates to improving the quality of the top generated paraphrase. The main difference is that our approach utilizes a generator discriminator to encourage more diverse paraphrases, which is essential for diversity.

**Diverse text generation** A few works have explored to produce diverse generation by changing decoding schemes or introducing random noise. Methods that change decoding schemes are orthogonal and complementary to our work. Li et al. (2016) modify the score function in decoding to encourage usage of novel words and penalize words after the same partially generated sentence. Dai et al. (2017) utilize conditional generative adversarial network (GAN) to generate diverse image caption according to the input noise. Gupta et al. (2018) employ a variational auto-encoder framework with both encoder and decoder conditioned on source input. Shi et al. (2018) employ inverse reinforcement learning for unconditional diverse text generation. Xu et al. (2018a) propose a modified GAN to generate diverse and informative outputs for different inputs. Xu et al. (2018b) train a generator with different embeddings to generate multiple paraphrases, only the decoder embedding with lowest cross entropy can get updated.

## 3 Methods

### 3.1 Problem Definition

Given a source sentence $X$, we aim to generate multiple target sentences $S = \{Y_1, Y_2, ..., Y_k\}$. Specifically, our goal is to learn a model which can generate $S$ such that every $Y_i$ (i=1..k) has the

same meaning as $X$ and obvious differences between each other.

### 3.2 Models

Our diverse paraphrase generation framework consists of three parts: one paraphrase discriminator, one generator discriminator and multiple generators. Paraphrase discriminator determines whether two sentences have the same meaning, and the generator discriminator determines the sentence is generated by which generator. The goal of multiple paraphrase generators is to generate sentences that are judged as paraphrases of the input sentence, and the differences between sentences should be as large as possible. The intuition is that sentences with quite different expressions can be easily distinguished by the generator discriminator and will be judged as generated by the corresponding generator with high confidence.

#### 3.2.1 Paraphrase generator

We frame paraphrase generation as a sequence-to-sequence (Seq2Seq) problem and adopt the encoder-decoder framework. Multiple generators share the same encoder yet with their own decoder. The generation posterior of the $i$-th generator is:

$$G_i(Y|X) = \prod_{t=1}^{T} P(y_t|y_{1:t-1}, X; \theta_i),$$

where $y_{1:t-1}$ indicates the subsequence from timestep 1 to timestep $t-1$, $T$ denotes the length of output sentence $Y$, $\theta_i$ represents the parameter of the $i$-th generator, we omit $\theta_i$ in the following illustration for conciseness. We use pointer-generator (See et al., 2017) for our generators to alleviate the out-of-vocabulary problem and enhance the performance. In pointer-generator, the decoder can generate not only words from the chosen vocabulary as standard decoders but also words from the input sentences by copying them from the input. The probability of copying words can be represented as

$$p_{copy}(y_t|y_{1:t-1}, X) = a_{ti},$$

$$a_{ti} = \frac{exp(f(s_{t-1}, h_i))}{\sum_i exp(f(s_{t-1}, h_i))},$$

where $s_{t-1}$ is the decoder state at timestep $t-1$, $h_i$ is the encoder hidden state at timestep $i$. $p_{copy}$ is actually $a_{ti}$, which is the attention distribution at timestep $t$. We use attention in (Bahdanau et al., 2014), thus $f$ is a linear function. The probability of extracting the next word from vocabulary is:

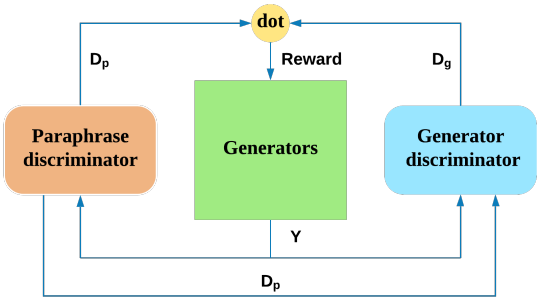


Figure 1: The overall framework

$$p_{vocab}(y_t|y_{1:t-1}, X) = g(s_{t-1}, y_{t-1}, c_t),$$

$$c_t = \sum_i a_{ti} * h_i,$$

where $y_{t-1}$ is the word generated last step, $c_t$ means the context vector at timestep $t$, $g$ is a linear function with softmax output. The overall probability of the next word is

$$p(y_t|y_{1:t-1}, X) = p_{gen} * p_{vocab} + (1 - p_{gen}) * p_{copy},$$

$$p_{gen} = m(s_t, y_{t-1}, c_t)$$

where $m$ is a linear transformation with sigmoid activation output. $p_{gen}$ acts as a gate to control whether the next word is generated from vocabulary or is copied from the input.

#### 3.2.2 Paraphrase discriminator

We follow (Li et al., 2018) to cast paraphrase recognition as a sentence matching problem. In our work, we use a model with two recurrent neural networks (RNN). Assume giving a pair of sentences $(X, Y)$, paraphrase discriminator will score a value to evaluate the semantic similarity between $X$ and $Y$.

$$\alpha = RNN_\alpha(X),\ \beta = RNN_\beta(Y),$$

$$\gamma = \tanh(W_\alpha * \alpha + b_\alpha) * \tanh(W_\beta * \beta + b_\beta),$$

$$D_p = softmax(W_p * f_c(\gamma) + b_p),$$

sentence $X$ and sentence $Y$ are encoded using RNN-based encoder. The encoded representations $\alpha$ and $\beta$ are integrated into $\gamma$. Then we input $\gamma$ into a fully connected network $f_c$ and finally output a softmax classification probability, which represents the degree of meaning similarity between

$X$ and $Y$. We optimize the cross entropy loss for paraphrase discriminator.

$$L = -\log(D_p(X, Y+)) - \log(1 - D_p(X, Y-)),$$

where$(X, Y-)$ means sentence $Y-$ is not a paraphrase of $X$ and $(X, Y+)$ means sentence $Y+$ is a paraphrase of $X$. Paraphrase discriminator separate the negative sentence pairs from the positive ones.

#### 3.2.3 Generator discriminator

In order to distinguish sentences generated by different generators, we learn a sentence discriminator. Suppose there is a set of sentences $\{S_1, S_2, ..., S_k\}$, where $S_i$ denotes sentences generated by the $i$-th generator. We hope our generator discriminator can separate sentences in $S_i$ $(i = 1..k)$ from sentences generated by other generators. If there are obvious differences between sentences in $S_i$ and those generated by other generators, generator discriminator can determine the sentence is generated by the $i$-th generator with high confidence. Otherwise, if sentences in $S_i$ are very similar to those generated by other generators, generator discriminator will get confused and misclassify the sentences into another generator, thus confidence will be low as well. For simplicity, we use a RNN to extract feature from sentences and use a softmax output as activation function.

$$e = RNN(Y),\ D_g(y|Y) = softmax(e)$$

where $e$ indicates the sentence feature and $y$ means which generator generates $Y$. The generator discriminator is pretrained using data generated by the pretrained generators. During the reinforcement learning phase, generator discriminator updates iteratively using new samples produced by all generators.

### 3.3 Learning

#### 3.3.1 Maximum likelihood estimation

The most typical technique to train a Seq2Seq model is teacher forcing (Williams and Zipser, 1989), which is also referred to as maximum likelihood estimation (MLE).

$$L = \sum_{t=1}^{T} \log p(y_t^* | y_{1:t-1}^*, X) \qquad (1)$$

where $y_t^*$ denotes the t-th word in the target sentence given by training sample. We pretrain all generators using teacher forcing. However, teacher forcing suffers from exposure bias (Ranzato et al., 2015). During training, the next word to generate is conditioned on the correctly annotated word subsequence. But in the inference stage, the model predicts the next word conditioned on words generated by itself, which causes inconsistency between training and inference. As model may generate wrong words, future generation conditioned on the generated subsequence will also be affected, leading to accumulated errors. One solution to this problem is finetuning generators via reinforcement learning after pretraining.

**Algorithm 1:** Training Procedure of our framework

**Input** : A data with paraphrase pairs $\{(X, Y)\}$ and non-paraphrase pairs $\{(X, Y-)\}$
**Output** : Generators $\{G_i\}_{i=1}^{i=k}$

Random initialize all generators $\{G_i\}_{i=1}^{i=k}$;
Pre-train all generators $G_i$ with $\{(X, Y)\}$;
Generate sentences $\{S_i\}_{i=1}^{i=k}$ , $S_i = \{\hat{Y}\}_i$ using $G_i$;
Train paraphrase discriminator with $\{(X, Y)\}$ and $\{(X, Y-)\}$;
Pre-train generator discriminator with $\{S_i\}_{i=1}^{i=k}$;
**while** *not converge* **do**
    Sample sentences $\{X\}$ from the paraphrase corpus;
    **for** $i = 1$ *To* $k$ **do**
        Using $G_i$ to generate sentences $\{\hat{Y}\}_i$ by sampling and $\{\bar{Y}\}_i$ by greedy search;
        Compute the rewards using Eq(3);
        Update generators by minimizing Eq(4);
    **end**
    Generate sentences $\{S_i\}_{i=1}^{i=k}$ using $\{G_i\}_{i=1}^{i=k}$;
    Update generator discriminator with $\{S_i\}_{i=1}^{i=k}$ by minimizing Eq(7);
**end**
**Return** $\{G_i\}_{i=1}^{i=k}$

#### 3.3.2 Reinforcement learning

Regarding a generator as an agent interacting with the environment (i.e. the word input and context vector). The objective function of the generator can be written as

$$L = \sum_{\hat{Y}} R(\hat{Y}), \hat{Y} \sim \pi_\theta \qquad (2)$$

$\hat{Y}$ indicates the word sequence sampled from the generator. $R$ is the reward function, and a batch of sentences is sampled to approximate the objective function since it is intractable to use all sentences. The optimization objective is to minimize the negative expected rewards of generated texts, in other words, maximize the expected rewards.

The reward of each sentence is determined by the paraphrase discriminator and the generator discriminator together. The reward of the $i$-th generator is

$$R_i = D_p(X, Y_i) \odot D_g(y = i|Y_i), \quad (3)$$

where $Y_i$ indicates the sentence generated by the $i$-th generator. The $i$-th generator can get a high reward only when sentence $Y_i$ is recognized as a paraphrase of $X$ by the paraphrase discriminator and the differences between $Y_i$ and sentences generated by other generators is so large that generator discriminator takes $Y_i$ as a sentence generated by the $i$-th generator.

Specifically, we use the self-critical algorithm (Rennie et al., 2017) as our reinforcement learning method. In order to reduce the variance of policy gradient method, a typical technique is subtracting baseline values from the original rewards. In the self-critical algorithm, the baseline is the reward of sentences generated in inference. At each training iteration, the generator generates two sentences. One sentence is sampled from the output probability distribution, $\hat{y}_t \sim p(y_t|\hat{y}_{1:1-t}, X)$. The other sentence is generated by greedy search, $\bar{y}_t = \arg\max_{y_t} P(y_t|\bar{y}_{1:t-1}, X)$, the word with the highest probability is selected at each step. The reward of the sentence generated by greedy search is used as a baseline. We use rewards subtracted by baseline values to update the model

$$L = \sum_{t=1}^{T} (R(\hat{y}) - R(\bar{y})) \log p(y_t|y_{1:t-1}, X), \quad (4)$$

which means that we compare the rewards of the sampled sentences $\hat{y}$ and those of the sentences $\bar{y}$ which are produced by greedy search. Only the generated sentences that outperform the sentence $\bar{y}$ are given positive signals. As a result, Rewards will be increased as the generators increase the generation probability of better sequences while decreasing the chances of worse sequence generation.

### 3.3.3 The overall objective

Our goal is to generate multiple diverse paraphrases of high quality. We illustrate the reason why our model can achieve this goal from the aspect of the whole optimization objective. Given input $X$, we hope to generate multiple high-quality sentences $Y$. We make the assumption that $y$ is only dependent on $Y$, that is, $D_g(y|Y, X) = D_g(y|Y)$. The weaker condition forces the generator discriminator to only focus on the generated sentences and encourages stronger diversity. The overall objective can be written as:

$$\begin{aligned} &\sum_{Y} R(X, Y) P(Y|X) \\ =&\sum_{Y} \sum_{y} R(X, Y) P(Y, y|X) \\ =&\sum_{Y} \sum_{i=1}^{k} R(X, Y) D_g(y = i|Y) G_i(Y|X) \end{aligned} \quad (5)$$

where $R(X, Y)$ is the reward evaluating output $Y$ for input sentence $X$. Multiple generators $G_i$ generate their own sequences for each input and $D_g$ learns to separate sentences generated by different generators. We can define different rewards for corresponding applications and we use $D_p(X, Y)$ for our paraphrase task. We define $\theta_d$ as the parameter of generator discriminator and $\theta_i$ as the parameter of the $i$-th generator. The derivative of $J(\theta)$ is computed as follow:

$$\begin{aligned} &\nabla_\theta J(\theta) \\ =&\sum_{i=1}^{k} \sum_{Y} [R(X, Y) G_i(Y|X) \nabla_{\theta_d} D_g(y = i|Y) \\ &\quad + R(X, Y) D_g(y = i|Y) \nabla_{\theta_i} G_i(Y|X)] \\ =&\sum_{i=1}^{k} \mathbb{E}_{G_i(Y|X)} [R(X, Y) \nabla_{\theta_d} D_g(y = i|Y) \\ &\quad + R(X, Y) D_g(y = i|Y) \nabla_{\theta_i} \log(G_i(Y|X))] \end{aligned} \quad (6)$$

Since the gradient can not be propagated backward to $G_i$ and $D_g$ by gradient descent, we use reinforcement learning to update both $D_g$ and $G_i$. And directly optimizing the above objective will reduce the reward of a good output $Y^*$ to zero for all generators except the one with the highest generation probability for this output $Y^*$. We should lower the reward of that good output for other generators, but reducing the reward of $Y^*$ to zero will affect the stability of other generators. To address this issue, we can regularize $D_g$ by adding entropy or we propose to scale the gradient of $D_g$ by $1/D_g$. The derived cost function of generator discriminator is:

$$L = -\sum_{i=1}^{k} \mathbb{E}_{G_i(Y|X)} R(X, Y) \log(D_g(y = i|Y)) \quad (7)$$

The outputs generated by $G_i$ are not equally contributed to $D_g(y = i|Y)$, outputs that have better quality are more encouraged by $D_g$. High-quality sentences $Y$ generated by $G_i$ maximize $D_g(y = i|Y)$ while minimizing $D_g(y = j|Y)$ $(j \neq i)$ to force other generators to generate different sentences. The $i$-th generator maximizes $R(X, Y)D_g(y = i|Y)$ so that sentences generated by the $i$-th generator satisfy both the quality and diversity requirement. The complete learning procedure is shown in algorithm 1.

## 4 Experiment

### 4.1 Dataset

| | Generator | | | Paraphrase discriminator | |
|---|---|---|---|---|---|
| Dataset | #Train | #Validation | #Test | #Positive | #Negative |
| Quora | 100K | 3K | 30K | 100K | 160K |
| Twitter | 110K | 1K | 5K | 10K | 40K |

Table 1: Statistics of datasets.

To validate our model's capability to generate diverse paraphrases, we conduct experiments on two real-world datasets: Quora question paraphrase dataset [1] and Twitter URL paraphrasing dataset (Lan et al., 2017).

Quora dataset contains a large corpus of question pairs collected from Quora. The Quora dataset totally contains about 400k data pairs. All the data are manually labeled whether two sentences convey the identical meaning. We randomly select 100k pairs for training, 3k for validation and 30k for test. The twitter dataset contains sentence pairs collected from Twitter. It also marks whether the sentence pairs are paraphrases. But the difference is that some of them are manually labeled and some are automatically labeled by the algorithm. For generators, we split test set and validation set from the manually labeled subset. The training set contains the rest data from the manually labeled subset and data from the automatically labeled paraphrase pairs. In order to avoid bias induced by the labelling algorithm, we only use the manually labeled data for the paraphrase discriminator. A summary of dataset statistics is given in Table 1.

### 4.2 Evaluation metric and baseline

We evaluate our model's performance for both quality and diversity. For sentence quality, we use two commonly used metrics: BLEU (Papineni et al., 2002) and METEOR (Lavie and Agarwal, 2007). BLEU measures word overlapping between generated sentences and reference texts. METEOR introduces stemming and synonym matching to alleviate problems caused by only exact matching in BLEU. BLEU and METEOR scores are averaged for multiple generated sentences. To evaluate sentence quality more reliably, we also conduct human evaluations to compare the quality of sentences generated by different models. For diversity, we use self-BLEU (Zhu et al., 2018). To compute self-BLEU of a set of outputs, every output is picked once as candidate and others are used as references to compute BLEU score. And the average of all the BLEU scores computed for this set is self-BLEU. A lower self-BLEU score indicates better diversity.

Several basic models and existing neural-based methods are used as baseline models for comparison. For both datasets, we report results of attentive Seq2Seq (Bahdanau et al., 2014), variational auto-encoder (VAE-SVG-eq) (Gupta et al., 2018), conditional GAN (Dai et al., 2017) with the generator replaced by Seq2Seq, diversity-promoting GAN (DP-GAN) (Xu et al., 2018a), the generator with inverse reinforcement learning (IRL) (Shi et al., 2018), the generator with multiple decoder embeddings (D-PAGE) (Xu et al., 2018b) and pointer-generator (See et al., 2017). We use beam search for attentive Seq2Seq (Seq2Seq-BS), and pointer-generator (PG-BS) to generate multiple outputs. Top three outputs of each method are obtained for evaluation.

### 4.3 Implementation

We use the following experimental setting for our model. For generators, we randomly initialize all generators and train them by teacher forcing in the same way. We use three generators and utilize single-layer LSTM (Hochreiter and Schmidhuber, 1997) for both encoder and decoders. The hidden dimension of encoder and decoders is set to 256, the word embedding dimension is 128. We use Adam optimizer (Duchi et al., 2011) with a learning rate of 1e-3 in MLE pretrain. In reinforcement learning, the learning rate decreases to 2e-5 and the batch size is fixed at 32. We also use ground-truth with reward of $D_g$ to stabilize the training. We strengthen the performance of D-PAGE by replacing the basic Seq2Seq with pointer-generator.

[1] https://www.kaggle.com/c/quora-question-pairs

| | Quora (Quality) | | | | Quora (Diversity) | | |
|---|---|---|---|---|---|---|---|
| Models | BLEU2↑ | BLEU3↑ | BLEU4↑ | METEOR↑ | self-BLEU2↓ | self-BLEU3↓ | self-BLEU4↓ |
| Seq2Seq-BS | 37.03 | 27.53 | 21.11 | 27.78 | 73.89 | 66.51 | 59.33 |
| VAE-SVG-eq | 34.78 | 25.96 | 19.97 | 26.45 | 89.82 | 87.53 | 85.43 |
| GAN | 36.22 | 27.27 | 21.12 | 26.63 | 92.85 | 91.22 | 89.76 |
| DP-GAN | 36.60 | 26.92 | 20.42 | 28.17 | 73.76 | 66.37 | 59.12 |
| IRL | 35.53 | 26.19 | 19.94 | 27.29 | 76.76 | 70.22 | 63.59 |
| D-PAGE | 38.70 | 29.23 | **22.66** | 28.54 | 90.10 | 87.66 | 85.41 |
| PG-BS | **38.91** | **29.26** | 22.54 | 28.88 | 75.60 | 68.72 | 61.89 |
| ours | 36.84 | 27.46 | 20.98 | **29.28** | **56.38** | **47.76** | **40.55** |

Table 2: Performances on Quora dataset.

| | Twitter (Quality) | | | | Twitter (Diversity) | | |
|---|---|---|---|---|---|---|---|
| Models | BLEU2↑ | BLEU3↑ | BLEU4↑ | METEOR↑ | self-BLEU2↓ | self-BLEU3↓ | self-BLEU4↓ |
| Seq2Seq-BS | 32.69 | 28.25 | 24.99 | 22.51 | 82.79 | 80.29 | 77.98 |
| VAE-SVG-eq | 26.43 | 23.04 | 20.57 | 18.34 | 91.46 | 90.51 | 89.67 |
| GAN | 23.10 | 20.45 | 18.47 | 15.33 | 94.35 | 93.75 | 93.22 |
| DP-GAN | 33.07 | 28.68 | 25.49 | 22.84 | 82.52 | 80.05 | 77.63 |
| IRL | 32.96 | 28.60 | 25.39 | 22.72 | 83.53 | 81.22 | 78.95 |
| D-PAGE | 32.95 | 28.82 | 25.88 | 22.59 | 88.35 | 86.45 | 84.76 |
| PG-BS | 33.86 | 29.42 | 26.15 | 23.52 | 82.57 | 79.99 | 77.48 |
| ours | **34.23** | **29.66** | **26.38** | **24.29** | **65.83** | **61.17** | **57.45** |

Table 3: Performances on Twitter dataset.

| | Quora | | Twitter | |
|---|---|---|---|---|
| Models | Fluency | Consistency | Fluency | Consistency |
| D-PAGE | 4.21 | 3.44 | 3.66 | 3.08 |
| PG-BS | 4.20 | 3.34 | 3.85 | 3.17 |
| DP-GAN | 4.27 | 3.49 | 4.09 | 3.30 |
| ours | **4.57** | **3.82** | **4.24** | **3.59** |

Table 4: Human evaluation results.

For paraphrase discriminator and generator discriminator, the sentence encoders are all built with one-layer LSTM with 256-dimensional hidden units and the dimension of word embeddings is set to 256. For each of the two discriminators, Adam optimizer with a learning rate of 1e-4 is used to optimize the loss function. The batch size is kept at 64 for paraphrase discriminator. We employ a 3-layer fully connected network with ReLU activation in paraphrase discriminator and set the dropout to 0.2.

## 5 Results

Results of automatic evaluation on Quora is shown in Table 2. Our model achieves the best METEOR score among all the models. As BLEU only calculate the exact overlapping between generated sentences and reference sentences, lower BLEU score does not necessarily indicate bad quality. In addition, every test case only has one reference text, which makes automatic evaluation more difficult to measure the real quality of multiple generated sentences. Since METEOR compares the generated sentences with the reference text using not only exact word matching but also stemming and synonym matching, we believe it measures the generation quality more accurately.

For human evaluation, we randomly select 100 input sentences from the test set of each dataset and get the generated results of different models for these inputs. We follow the human evaluation guideline in (Li et al., 2018). The sentence pairs are scored for two aspects of the generated results: *fluency* (whether the generated sentence is fluent and grammatical) and *consistency* (the meaning similarity between the input sentence and the generated sentence). Each output is given two ratings, scaling from 1 to 5, on fluency and consistency separately. An input and outputs generated for the input from different systems form a test group. The outputs from different systems in the test group are shuffled. The test groups are randomly assigned to annotators. Every test group is evaluated by two annotators and scores for each output are averaged. The agreement between annotators is moderate (kappa=0.54). The final score for each system is the average score for all outputs. Table 4 shows our model achieves better scores in both meaning similarity and fluency than baseline models.

| | |
|---|---|
| **Source** | How do you know you 're in love again ? |
| **PG-BS** | How do you know you are in love with someone ?<br>How do you know if you 're in love with someone ?<br>How do you know if you are in love with someone ? |
| **ours** | How do you know if you 're in love with someone ?<br>What are some ways to know that you are in love with someone ?<br>How can you know if you 're in love ? |
| **Source** | How does this demonetisation of 500 & 1000 rupee notes help to reduce the price of real estate ? |
| **PG-BS** | What will be the effect of banning 500 and 1000 notes on the Indian economy ?<br>What will be the effect of banning 500 and 1000 notes on the stock markets ?<br>What will be the effect of banning 500 and 1000 notes on the real estate sector ? |
| **ours** | What will be the effect of banning 500 and 1000 notes on real estate sector in India ?<br>How does the declaration that Rs 500 and Rs 1000 notes would not be accepted as valid transactions affect real estate in India ?<br>How can the ban of 500 and 1000 rupee notes help in curbing the price of real estate ? |
| **Source** | I want to write a book - where should I start ? |
| **PG-BS** | I want to write a book , where should I start ?<br>I want to write a book , how should I start ?<br>I want to write a book , where can I start ? |
| **ours** | How do I write a book ?<br>What should I do to write a book - where should I start ?<br>I want to write a book , where should I start ? |

Table 5: Example paraphrases on Quora

In terms of diversity, our approach outperforms other methods by a large margin. VAE-SVG-eq, GAN and D-PAGE do not show better results than beam search. As discussed in (Xu et al., 2018b), beam search guarantees that multiple generated sentences are different from each other by at least one word, which yields better diversity. GAN methods often suffer from mode collapse problems and generate similar outputs. For VAE-SVG-eq, different random noise sampled from a Gaussian prior is used identically in training, so different noise inputs may produce the same sentences. The generator with IRL and DP-GAN have no obvious improvement in diversity in our task. Although the generator with IRL and DP-GAN also claim to improve diversity, their goal is essentially different from ours. The generator with IRL was proposed to tackle the unconditional text generation problem, where texts are generated without any input or condition. DP-GAN generates a more informative output for each input to avoid repetitive response for different inputs. In the experiment, we notice that VAE-SVG-eq and GAN generate similar or even identical sentences according to different random noise, which is consistent with the results posted in (Xu et al., 2018b). D-PAGE may able to find different patterns for very short sentences or phrases, but results in our experiments show that D-PAGE does not separate different patterns in longer sentences and the generated sentences lack diversity.

Table 3 shows scores on Twitter dataset. Our model achieves the best BLEU score and outperforms all baselines for METEOR. The human evaluation results presented in table 4 also show that our model produces better quality paraphrases. For diversity, our model again performs considerably better than other models. Since part of sentence pairs in Twitter dataset may be mislabelled by the labeling algorithm, it is harder to generate high-quality paraphrases on Twitter. Paraphrases generated by our model have not only more diverse expression but also better quality compared to those generated by other models.

Table 5 demonstrates some examples generated by pointer-generator using beam search and our model. Although pointer-generator indeed generates different sentences, it only yields little diversity. Sentences produced by pointer-generator can be very similar, many of them only differ from others by one word. Compared to sentences generated by pointer-generator, those produced by our model has obvious differences between each other. In the first example, our model uses different types of interrogative. In contrast, pointer-generator generates sentences that are almost the same. In the second example, pointer-generator generates similar sentences and yields obvious meaning changes from the input sentence. The sentences generated by our model are in different expressions and more relevant to the input sentence. In the third example, the sentences generated by pointer-generator show minor variations, while those generated by our model present richer expressions.

## 6 Conclusion

In this paper, we present a novel approach to generate diverse paraphrases. Multiple generators are pretrained in the same way, then learn different generation pattern and enhance generation quality via reinforcement learning. Paraphrase discriminator evaluates the meaning similarity between two sentences, generator discriminator distinguishes sentences generated from different generators. Experiments show that our model has better results concerning both diversity and quality. In the future, we would like to apply our framework to increase reference texts for automatic evaluation or augment training data for text classification.

## Acknowledgments

The work is sponsored by Huawei Innovation Research Program. The corresponding authors Weinan Zhang and Yong Yu are also supported by NSFC (61702327, 61772333, 61632017). We thank the anonymous reviewers for their valuable comments.

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Bo Dai, Sanja Fidler, Raquel Urtasun, and Dahua Lin. 2017. Towards diverse and natural image descriptions via a conditional gan. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2970–2979.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1156–1165. ACM.

Ankush Gupta, Arvind Agarwal, Prawaan Singh, and Piyush Rai. 2018. A deep generative framework for paraphrase generation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Samer Hassan, Andras Csomai, Carmen Banea, Ravi Sinha, and Rada Mihalcea. 2007. Unt: Subfinder: Combining knowledge sources for automatic lexical substitution. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 410–413. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885.

Wuwei Lan, Siyu Qiu, Hua He, and Wei Xu. 2017. A continuously growing dataset of sentential paraphrases. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1224–1234.

Alon Lavie and Abhaya Agarwal. 2007. Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 228–231. Association for Computational Linguistics.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. A simple, fast diverse decoding algorithm for neural generation. *arXiv preprint arXiv:1611.08562*.

Zichao Li, Xin Jiang, Lifeng Shang, and Hang Li. 2018. Paraphrase generation with deep reinforcement learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3865–3878.

Nitin Madnani and Bonnie J Dorr. 2010. Generating phrasal and sentential paraphrases: A survey of data-driven methods. *Computational Linguistics*, 36(3):341–387.

Kathleen R McKeown. 1983. Paraphrasing questions using given and new information. *Computational Linguistics*, 9(1):1–10.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Ça glar Gulçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *CoNLL 2016*, page 280.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Aaditya Prakash, Sadid A Hasan, Kathy Lee, Vivek Datla, Ashequl Qadir, Joey Liu, and Oladimeji Farri. 2016. Neural paraphrase generation with stacked residual lstm networks. *arXiv preprint arXiv:1610.03098*.

Chris Quirk, Chris Brockett, and Bill Dolan. 2004. Monolingual machine translation for paraphrase generation.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *CVPR*, volume 1, page 3.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1073–1083.

Iulian Vlad Serban, Tim Klinger, Gerald Tesauro, Kartik Talamadupula, Bowen Zhou, Yoshua Bengio, and Aaron C Courville. 2017. Multiresolution recurrent neural networks: An application to dialogue response generation. In *AAAI*, pages 3288–3294.

Zhan Shi, Xinchi Chen, Xipeng Qiu, and Xuanjing Huang. 2018. Towards diverse text generation with inverse reinforcement learning. *arXiv preprint arXiv:1804.11258*.

Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280.

Sander Wubben, Antal Van Den Bosch, and Emiel Krahmer. 2010. Paraphrase generation as monolingual translation: Data and evaluation. In *Proceedings of the 6th International Natural Language Generation Conference*, pages 203–207. Association for Computational Linguistics.

Jingjing Xu, Xuancheng Ren, Junyang Lin, and Xu Sun. 2018a. Dp-gan: diversity-promoting generative adversarial network for generating informative and diversified text. *arXiv preprint arXiv:1802.01345*.

Qiongkai Xu, Juyan Zhang, Lizhen Qu, Lexing Xie, and Richard Nock. 2018b. D-page: Diverse paraphrase generation. *arXiv preprint arXiv:1808.04364*.

Pengcheng Yin, Nan Duan, Ben Kao, Junwei Bao, and Ming Zhou. 2015. Answering questions with complex semantic constraints on open knowledge bases. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1301–1310. ACM.

Shiqi Zhao, Cheng Niu, Ming Zhou, Ting Liu, and Sheng Li. 2008. Combining multiple resources to improve smt-based paraphrasing model. *Proceedings of ACL-08: HLT*, pages 1021–1029.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1097–1100. ACM.

# Enhancing AMR-to-Text Generation with Dual Graph Representations

**Leonardo F. R. Ribeiro**[†], **Claire Gardent**[‡] **and Iryna Gurevych**[†]

[†]Research Training Group AIPHES and UKP Lab, Technische Universität Darmstadt
www.ukp.tu-darmstadt.de

[‡]CNRS/LORIA, Nancy, France
claire.gardent@loria.fr

## Abstract

Generating text from graph-based data, such as Abstract Meaning Representation (AMR), is a challenging task due to the inherent difficulty in how to properly encode the structure of a graph with labeled edges. To address this difficulty, we propose a novel graph-to-sequence model that encodes different but complementary perspectives of the structural information contained in the AMR graph. The model learns parallel top-down and bottom-up representations of nodes capturing contrasting views of the graph. We also investigate the use of different node message passing strategies, employing different state-of-the-art graph encoders to compute node representations based on incoming and outgoing perspectives. In our experiments, we demonstrate that the dual graph representation leads to improvements in AMR-to-text generation, achieving state-of-the-art results on two AMR datasets[1].

## 1 Introduction

Abstract Meaning Representation (AMR; Banarescu et al. (2013)) is a linguistically-grounded semantic formalism that represents the meaning of a sentence as a rooted directed graph, where nodes are concepts and edges are semantic relations. As AMR abstracts away from surface word strings and syntactic structure producing a language neutral representation of meaning, its usage is beneficial in many semantic related NLP tasks, including text summarization (Liao et al., 2018) and machine translation (Song et al., 2019).

The purpose of AMR-to-text generation is to produce a text which verbalises the meaning encoded by an input AMR graph. This is a challenging task as capturing the complex structural information stored in graph-based data is not trivial, as these are non-Euclidean structures, which implies that properties such as global parametrization, vector space structure, or shift-invariance do not hold (Bronstein et al., 2017). Recently, Graph Neural Networks (GNNs) have emerged as a powerful class of methods for learning effective graph latent representations (Xu et al., 2019) and graph-to-sequence models have been applied to the task of AMR-to-text generation (Song et al., 2018; Beck et al., 2018; Damonte and Cohen, 2019; Guo et al., 2019).

In this paper, we propose a novel graph-to-sequence approach to AMR-to-text generation, which is inspired by pre-neural generation algorithms. These approaches explored alternative (top-down, bottom-up and mixed) traversals of the input graph and showed that a hybrid traversal combining both top-down (TD) and bottom-up (BU) information was best as this permits integrating both global constraints top-down from the input and local constraints bottom-up from the semantic heads (Shieber et al., 1990; Narayan and Gardent, 2012).

Similarly, we present an approach where the input graph is represented by two separate structures, each representing a different view of the graph. The nodes of these two structures are encoded using separate graph encoders so that each concept and relation in the input graph is assigned both a TD and a BU representation.

Our approach markedly differs from existing graph-to-sequence models for MR-to-Text generation (Marcheggiani and Perez Beltrachini, 2018; Beck et al., 2018; Damonte and Cohen, 2019) in that these approaches aggregate all the immediate neighborhood information of a node in a single representation. By exploiting parallel and complementary vector representations of the AMR graph, our approach eases the burden on the neural model in encoding nodes (concepts) and edges (relations) in a single vector representation. It also elimi-

[1]Code is available at https://github.com/UKPLab/emnlp2019-dualgraph



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3183–3194, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

nates the need for additional positional information (Beck et al., 2018) which is required when the same graph is used to encode both TD and BU information, thereby making the edges undirected.

Our main contributions are the following:

- We present a novel architecture for AMR-to-text generation which explicitly encodes two separate TD and BU views of the input graph.
- We show that our approach outperforms recent AMR-to-text generation models on two datasets, including a model that leverages additional syntactic information (Cao and Clark, 2019).
- We compare the performance of three graph encoders, which have not been studied so far for AMR-to-text generation.

## 2 Related Work

Early works on AMR-to-text generation employ statistical methods (Flanigan et al., 2016b; Pourdamghani et al., 2016; Castro Ferreira et al., 2017) and apply linearization of the graph by means of a depth-first traversal.

Recent neural approaches have exhibited success by linearising the input graph and using a sequence-to-sequence architecture. Konstas et al. (2017) achieve promising results on this task. However, they strongly rely on named entities anonymisation. Anonymisation requires an ad hoc procedure for each new corpus. The matching procedure needs to match a rare input item correctly (e.g., "United States of America") with the corresponding part in the output text (e.g., "USA") which may be challenging and may result in incorrect or incomplete delexicalisations. In contrast, our approach omits anonymisation. Instead, we use a copy mechanism (See et al., 2017), a generic technique which is easy to integrate in the encoder-decoder framework and can be used independently of the particular domain and application. Our approach further differs from Konstas et al. (2017) in that we build a dual TD/BU graph representation and use graph encoders to represent nodes.

Cao and Clark (2019) factor the generation process leveraging syntactic information to improve the performance. However, they linearize both AMR and constituency graphs, which implies that important parts of the graphs cannot well be represented (e.g., coreference).

Several graph-to-sequence models have been proposed. Marcheggiani and Perez Beltrachini (2018) show that explicitly encoding the structure of the graph is beneficial with respect to sequential encoding. They evaluate their model on two tasks, WebNLG (Gardent et al., 2017) and SR11Deep (Belz et al., 2011), but do not apply it to AMR benchmarks. Song et al. (2018) and Beck et al. (2018) apply recurrent neural networks to directly encode AMR graphs. Song et al. (2018) use a graph LSTM as the graph encoder, whereas Beck et al. (2018) develop a model based on GRUs. We go a step further in that direction by developing parallel encodings of graphs which are able to highlight different graph properties.

In a related task, Koncel-Kedziorski et al. (2019) propose an attention-based graph model that generates sentences from knowledge graphs. Schlichtkrull et al. (2018) use Graph Convolutional Networks (GCNs) to tackle the tasks of link prediction and entity classification on knowledge graphs.

Damonte and Cohen (2019) show that off-the-shelf GCNs cannot achieve good performance for AMR-to-text generation. To tackle this issue, Guo et al. (2019) introduce dense connectivity to GNNs in order to integrate both local and global features, achieving good results on the task. Our work is related to Damonte and Cohen (2019), that use stacking of GCN and LSTM layers to improve the model capacity and employ anonymization. However, our model is substantially different: (i) we learn dual representations capturing top-down and bottom-up adjuvant views of the graph, (ii) we employ more effective graph encoders (with different neighborhood aggregations) than GCNs and (iii) we employ copy and coverage mechanisms and do not resort to entity anonymization.

## 3 Graph-to-Sequence Model

In this section, we describe (i) the representations of the graph adopted as inputs, (ii) the model architecture, including the Dual Graph Encoder and (iii) the GNNs employed as graph encoders.

### 3.1 Graph Preparation

Let $G = (V, E, R)$ denote a rooted and directed AMR graph with nodes $v_i \in V$ and labeled edges $(v_i, r, v_j) \in E$, where $r \in R$ is a relation type.

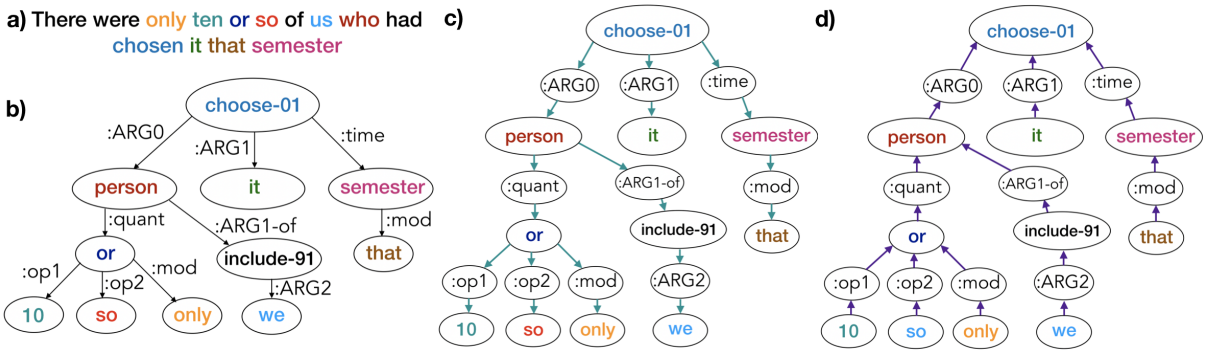


Figure 1: (a) an example sentence, (b) its original AMR graph ($G$) and different graph perspectives: (c) top-down ($G_t$) and (d) bottom-up ($G_b$).

Let $n = |V|$ and $m = |E|$ denote the numbers of nodes and edges, respectively.

We convert each AMR graph into an unlabeled and connected bipartite graph $G_t = (V_t, E_t)$, transforming each labeled edge $(v_i, r, v_j) \in E$ into two unlabeled edges $(v_i, r), (r, v_j) \in E_t$, with $|V_t| = n + m$ and $|E_t| = 2m$. This process, called Levi Transformation (Beck et al., 2018), turns original edges into nodes creating an unlabeled graph. For instance, the edge between `semester` and `that` with label `:mod` in Figure 1(b) is replaced by two edges and one node in 1(c): an edge between `semester`, and the new node `:mod` and another one between `:mod` and `that`. The new graph allows us to directly represent the relationships between nodes using embeddings. This enables us to encode label edge information using distinct message passing schemes employing different GNNs.

$G_t$ captures a TD view of the graph. We also create a BU view of the graph $G_b = (V_t, E_b)$, where each directed edge $e_k = (v_i, v_j) \in E_t$ becomes $e_k = (v_j, v_i) \in E_b$, that is, we reverse the direction of original edges. An example of a sentence, its AMR graph and the two new graphs $G_t$ and $G_b$ is shown in Figure 1.

### 3.2 Dual Graph Encoder

We represent each node $v_i \in V_t$ with a node embedding $\mathbf{e}_i \in \mathbb{R}^d$, generated from the node label. In order to explicitly encode structural information, our encoder starts with two graph encoders, denoted by $GE_t$ and $GE_b$, that compute representations for nodes in $G_t$ and $G_b$, respectively.

Each $GE$ learns node representations based on the specific view of its particular graph, $G_t$ or $G_b$. Since $G_t$ and $G_b$ capture distinct perspectives of the graph structure, the information flow is propagated throughout TD and BU directions, respectively. In particular, for each node $v_i$, the $GE$ receives the node embeddings of $v_i$ and its neighbors, and computes its node representation:

$$\mathbf{h}_i^t = GE_t(\{\mathbf{e}_i, \mathbf{e}_j : j \in \mathcal{N}_t(i)\}),$$
$$\mathbf{h}_i^b = GE_b(\{\mathbf{e}_i, \mathbf{e}_j : j \in \mathcal{N}_b(i)\}),$$

where $\mathcal{N}_t(i)$ and $\mathcal{N}_b(i)$ are the immediate incoming neighborhoods of $v_i$ in $G_t$ and $G_b$, respectively.

Each node $v_i$ is represented by two different hidden states, $\mathbf{h}_i^t$ and $\mathbf{h}_i^b$. Note that we learn two representations per relation and node of the original AMR graph. The hidden states $\mathbf{h}_i^t$ and $\mathbf{h}_i^b$, and embedding $\mathbf{e}_i$ contain different information regarding $v_i$. We concatenate them building a final node representation:

$$\mathbf{r}_i = \left[\, \mathbf{h}_i^t \,\|\, \mathbf{h}_i^b \,\|\, \mathbf{e}_i \,\right].$$

This approach is similar to bidirectional RNNs (Schuster and Paliwal, 1997). Bidirectional RNNs benefit from left-to-right and right-to-left propagation. They learn the hidden representations separately and concatenate them at the end. We perform a similar encoding: first we learn TD and BU representations independently, and lastly, we concatenate them.

The final representation $\mathbf{r}_i$ is employed in a sequence input of a bidirectional LSTM. For each AMR graph, we generate a node sequence by depth-first traversal order. In particular, given a representation sequence from $\mathbf{r}_1$ to $\mathbf{r}_n$, the hidden forward and backward states of $\mathbf{r}_i$ are defined as:

$$\overrightarrow{\mathbf{h}}_i = LSTM_f(\mathbf{r}_i, \overrightarrow{\mathbf{h}}_{i-1}),$$
$$\overleftarrow{\mathbf{h}}_i = LSTM_b(\mathbf{r}_i, \overleftarrow{\mathbf{h}}_{i-1}),$$

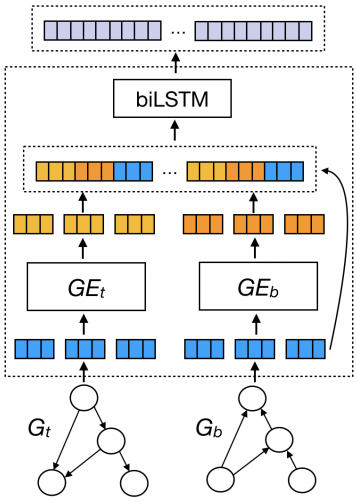


Figure 2: Dual Graph Encoder. The encoder receives the two graph views and generates structural node representations that are used by the decoder. Representations in blue, yellow and orange are $\mathbf{e}_i$, $\mathbf{h}_i^t$ and $\mathbf{h}_i^b$, respectively.

where $LSTM_f$ is a forward LSTM and $LSTM_b$ is a backward LSTM. Note that, for the backward LSTM, we feed the reversed input as the order from $\mathbf{r}_n$ to $\mathbf{r}_1$. Lastly, we obtain the final hidden state by concatenating them as:

$$\mathbf{h}_i = [\overrightarrow{\mathbf{h}}_i \,\|\, \overleftarrow{\mathbf{h}}_i].$$

The resulting hidden state $\mathbf{h}_i$ encodes the information of both preceding and following nodes.

Stacking layers was demonstrated to be effective in graph-to-sequence approaches (Marcheggiani and Perez Beltrachini, 2018; Koncel-Kedziorski et al., 2019; Damonte and Cohen, 2019) and allows us to test for their contributions to the system performance more easily. We employ different GNNs for both graph encoders (Section 3.3). Figure 2 shows the proposed encoder architecture.

### 3.3 Graph Neural Networks

The $GE$s incorporate, in each node representation, structural information based on both views of the graph. We explore distinct strategies for neighborhood aggregation, adopting three GNNs: Gated Graph Neural Networks (GGNN, Li et al. (2016)), Graph Attention Networks (GAT, Veličković et al. (2018)) and Graph Isomorphic Networks (GIN, Xu et al. (2019)). Each GNN employs a specific message passing scheme which allows capturing different nuances of structural information.

**Gated Graph Neural Networks** GGNNs employ gated recurrent units to encode node representations, reducing the recurrence to a fixed number of steps. In particular, the $l$-th layer of a GGNN is calculated as:

$$\mathbf{h}_i^{(l)} = GRU\Big(\mathbf{h}_i^{(l-1)}, \sum_{j \in \mathcal{N}(i)} \mathbf{W_1} \mathbf{h}_j^{(l-1)}\Big),$$

where $\mathcal{N}(i)$ is the immediate neighborhood of $v_i$, $\mathbf{W_1}$ is a parameter and $GRU$ is a gated recurrent unit (Cho et al., 2014). Different from other GNNs, GGNNs use back-propagation through time (BPTT) to learn the parameters. GGNNs also do not require to constrain parameters to ensure convergence.

**Graph Attention Networks** GATs apply attentive mechanisms to improve the exploitation of non-trivial graph structure. They encode node representations by attending over their neighbors, following a self-attention strategy:

$$\mathbf{h}_i^{(l)} = \alpha_{i,i} \mathbf{W_2} \mathbf{h}_i^{(l-1)} + \sum_{j \in \mathcal{N}(i)} \alpha_{i,j} \mathbf{W_2} \mathbf{h}_j^{(l-1)},$$

where attention coefficients $\alpha_{i,j}$ are computed as:

$$\alpha_{i,j} = \text{softmax}\left(\sigma\left(\mathbf{a}^\top [\mathbf{W_2}\mathbf{h}_i^{(l-1)} \,\|\, \mathbf{W_2}\mathbf{h}_j^{(l-1)}]\right)\right),$$

where $\sigma$ is the activation function and $\|$ denotes concatenation. $\mathbf{W_2}$ and $\mathbf{a}$ are model parameters. The virtue of the attention mechanism is its ability to focus on the most important parts of the node neighborhood. In order to learn attention weights in different perspectives, GATs can employ multi-head attentions.

**Graph Isomorphic Networks** GIN is a GNN as powerful as the Weisfeiler-Lehman (WL) graph isomorphism test (Weisfeiler and Lehman, 1968) in representing isomorphic and non-isomorphic graphs with discrete attributes. Its $l$-th layer is defined as:

$$\mathbf{h}_i^{(l)} = h_{\mathbf{W}}\Big(\mathbf{h}_i^{(l-1)} + \sum_{j \in \mathcal{N}(i)} \mathbf{h}_j^{(l-1)}\Big),$$

where $h_{\mathbf{W}}$ is a multi-layer perceptron (MLP). In contrast to other GNNs, which combine node feature with its aggregated neighborhood feature, GINs do not apply the combination step and simply aggregate the node along with its neighbors.

Each of these GNNs applies different approaches to learn structural features from graph data and has achieved impressive results on many graph-based tasks (Li et al., 2016; Veličković et al., 2018; Xu et al., 2019).

| | LDC2015E86 | | | LDC2017T10 | | |
|---|---|---|---|---|---|---|
| training, dev and test instances | 16,833 | 1,368 | 1,371 | 36,521 | 1,368 | 1,371 |
| min, average and max graph diameter | 0 | 6.9 | 20 | 0 | 6.7 | 20 |
| min, average and max node degree | 0 | 2.1 | 18 | 0 | 2.1 | 20 |
| min, average and max number of nodes | 1 | 17.7 | 151 | 1 | 16.8 | 151 |
| min, average and max number of edges | 0 | 18.6 | 172 | 0 | 17.7 | 172 |
| number of DAG and non-DAG graphs | 18,679 | 893 | | 37,284 | 1,976 | |
| min, average and max length sentences | 1 | 21.3 | 225 | 1 | 20.4 | 225 |

Table 1: Data statistics of LDC2015E86 and LDC2017T10 datasets. The values are calculated for all splits (train, development and test sets). DAG stands for directed acyclic graph.

### 3.4 Decoder

An attention-based unidirectional LSTM decoder is used to generate sentences, attending to the hidden representations of edges and nodes. In each step $t$, the decoder receives the word embedding of the previous word (during training, this is the previous word of the reference sentence; at test time it is the previously generated word), and has the decoder state $\mathbf{s}_t$. The attention distribution $\mathbf{a}^t$ is calculated as in See et al. (2017):

$$\mathbf{e}_i^t = \mathbf{v} \cdot \tanh(\mathbf{W}_h \mathbf{h}_i + \mathbf{W}_s \mathbf{s}_t + \mathbf{w}_c \mathbf{s}_c + \mathbf{b}),$$
$$\mathbf{a}^t = \text{softmax}(\mathbf{e}^t),$$

where $\mathbf{s}_c$ is the coverage vector and $\mathbf{v}$, $\mathbf{W}_h$, $\mathbf{W}_s$, $\mathbf{w}_c$ and $\mathbf{b}$ are learnable parameters. The coverage vector is the accumulation of all attention distributions so far.

**Copy and Coverage Mechanisms** Previous works (Damonte and Cohen, 2019; Cao and Clark, 2019) use anonymization to handle names and rare words, alleviating the data sparsity. In contrast, we employ copy and coverage mechanisms to address out-of-vocabulary issues for rare target words and to avoid repetition (See et al., 2017).

The model is trained to optimize the negative log-likelihood:

$$\mathcal{L} = -\sum_{t=1}^{|Y|} \log\, p(y_t | y_{1:t-1}, X; \theta),$$

where $Y = y_1, \ldots, y_{|Y|}$ is the sentence, $X$ is the AMR graph and $\theta$ represents the model parameters.

## 4 Data

We use two AMR corpora, LDC2015E86 and LDC2017T10[2]. In these datasets, each instance contains an AMR graph and a sentence. Table 1 shows the statistics for both datasets. Figure 3 shows the distribution of the AMR graph diameters and node degrees for both datasets. The AMR graph structures are similar for most examples. Note that 90% of AMR graphs in both datasets have the diameter less than or equal to 11 and 90% of nodes have the degree of 4 or less. Very structurally similar graphs pose difficulty for the graph encoder by making it harder to learn the differences between their similar structures. Therefore, the word embeddings used as additional input play an important role in helping the model to deal with language information. That is one of the reasons why we concatenate this information in the node representation $\mathbf{r}_i$.

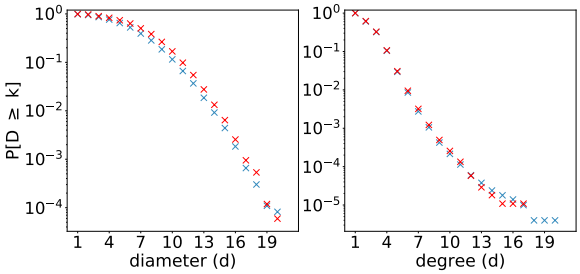


Figure 3: Distribution of the AMR graph diameter (left) and node degree (right) in the training set for LDC2015E86 (red) and LDC2017T10 (blue) datasets.

## 5 Experiments and Discussion

**Implementation Details** We extract vocabularies (size of 20,000) from the training sets and initialize the node embeddings from GloVe word embeddings (Pennington et al., 2014) on Common Crawl. Hyperparameters are tuned on the development set of the LDC2015E86 dataset. For GIN, GAT, and GGNN graph encoders, we set the number of layers to 2, 5 and 5, respectively. To regu-

[2]The datasets can be found at https://amr.isi.edu/download.html

| Model | BLEU | METEOR |
|---|---|---|
| LDC2015E86 | | |
| Konstas et al. (2017) | 22.00 | - |
| Song et al. (2018) | 23.28 | 30.10 |
| Cao et al. (2019) | 23.50 | - |
| Damonte et al.(2019) | 24.40 | 23.60 |
| Guo et al. (2019) | **25.70** | - |
| S2S | 22.55 ± 0.17 | 29.90 ± 0.31 |
| G2S-GIN | 22.93 ± 0.20 | 29.72 ± 0.09 |
| G2S-GAT | 23.42 ± 0.16 | 29.87 ± 0.14 |
| G2S-GGNN | 24.32 ± 0.16 | **30.53** ± 0.30 |
| LDC2017T10 | | |
| Back et al. (2018) | 23.30 | - |
| Song et al. (2018) | 24.86 | 31.56 |
| Damonte et al.(2019) | 24.54 | 24.07 |
| Cao et al. (2019) | 26.80 | - |
| Guo et al. (2019) | 27.60 | - |
| S2S | 22.73 ± 0.18 | 30.15 ± 0.14 |
| G2S-GIN | 26.90 ± 0.19 | 32.62 ± 0.04 |
| G2S-GAT | 26.72 ± 0.20 | 32.52 ± 0.02 |
| G2S-GGNN | **27.87** ± 0.15 | **33.21** ± 0.15 |

Table 2: BLEU and METEOR scores on the test set of LDC2015E86 and LDC2017T10 datasets.

larize the model, during training we apply dropout (Srivastava et al., 2014) to the graph layers with a rate of 0.3. The graph encoder hidden vector sizes are set to 300 and hidden vector sizes for LSTMs are set to 900.

The models are trained for 30 epochs with early stopping based on the development BLEU score. For our models and the baseline, we used a two-layer LSTM decoder. We use Adam optimization (Kingma and Ba, 2015) as the optimizer with an initial learning rate of 0.001 and 20 as the batch size. Beam search with the beam size of 5 is used for decoding.

**Results** We call the models G2S-GIN (isomorphic encoder), G2S-GAT (graph-attention encoder), and G2S-GGNN (gated-graph encoder), according to the graph encoder utilized. As a baseline (S2S), we train an attention-based encoder-decoder model with copy and coverage mechanisms, and use a linearized version of the graph generated by depth-first traversal order as input. We compare our models against several state-of-the-art results reported on the two datasets (Konstas et al., 2017; Song et al., 2018; Beck et al., 2018; Damonte and Cohen, 2019; Cao and Clark, 2019; Guo et al., 2019).

| Model | External | BLEU |
|---|---|---|
| Konstas et al. (2017) | 200K | 27.40 |
| Song et al. (2018) | 200K | 28.20 |
| Guo et al. (2019) | 200K | 31.60 |
| G2S-GGNN | 200K | **32.23** |

Table 3: Results on LDC2015E86 test set when models are trained with additional Gigaword data.

We use both BLEU (Papineni et al., 2002) and METEOR (Denkowski and Lavie, 2014) as evaluation metrics[3]. In order to mitigate the effects of random seeds, we report the averages for 4 training runs of each model along with their standard deviation. Table 2 shows the comparison between the proposed models, the baseline and other neural models on the test set of the two datasets.

For both datasets, our approach substantially outperforms the baselines. In LDC2015E86, G2S-GGNN achieves a BLEU score of 24.32, 4.46% higher than Song et al. (2018), who also use the copy mechanism. This indicates that our architecture can learn to generate better signals for text generation. On the same dataset, we have competitive results to Damonte and Cohen (2019). However, we do not rely on preprocessing anonymisation not to lose semantic signals. In LDC2017T10, G2S-GGNN achieves a BLEU score of 27.87, which is 3.33 points higher than Damonte and Cohen (2019), a state-of-the-art model that does not employ external information. We also have competitive results to Guo et al. (2019), a very recent state-of-the-art model.

We also outperform Cao and Clark (2019) improving BLEU scores by 3.48% and 4.00%, in LDC2015E86 and LDC2017T10, respectively. In contrast to their work, we do not rely on (i) leveraging supplementary syntactic information and (ii) we do not require an anonymization preprocessing step. G2S-GIN and G2S-GAT have comparable performance on both datasets. Interestingly, G2S-GGNN has better performance among our models. This suggests that graph encoders based on gating mechanisms are very effective in text generation models. We hypothesize that the gating mechanism can better capture long-distance dependencies between nodes far apart in the graph.

[3]For BLEU, we use the multi-BLEU script from the MOSES decoder suite (Koehn et al., 2007). For METEOR, we use the original meteor-1.5.jar script (https://github.com/cmu-mtlab/meteor).

| Model | BLEU | METEOR | Size |
|---|---|---|---|
| biLSTM | 22.50 | 30.42 | 57.6M |
| $GE_t$ + biLSTM | 26.33 | 32.62 | 59.6M |
| $GE_b$ + biLSTM | 26.12 | 32.49 | 59.6M |
| $GE_t$ + $GE_b$ + biLSTM | 27.37 | 33.30 | 61.7M |

Table 4: Results of the ablation study on the LDC2017T10 development set.

**Additional Training Data** Following previous works (Konstas et al., 2017; Song et al., 2018; Guo et al., 2019), we also evaluate our models employing additional data from English Gigaword corpus (Napoles et al., 2012). We sample 200K Gigaword sentences and use JAMR[4] (Flanigan et al., 2016a) to parse them. We follow the method of Konstas et al. (2017), which is fine-tuning the model on the LDC2015E86 training set after every epoch of pretraining on the Gigaword data. G2S-GGNN outperforms others with the same amount of Gigaword sentences (200K), achieving a 32.23 BLEU score, as shown in Table 3. The results demonstrate that pretraining on automatically generated AMR graphs enhances the performance of our model.

**Ablation Study** In Table 4, we report the results of an ablation study on the impact of each component of our model on the development set of LDC2017T10 dataset by removing the graph encoders. We also report the number of parameters (including embeddings) used in each model. The first thing we notice is the huge increase in metric scores (17% in BLEU) when applying the graph encoder layer, as the neural model receives signals regarding the graph structure of the input. The dual representation helps the model with a different view of the graph, increasing BLEU and METEOR scores by 1.04 and 0.68 points, respectively. The complete model has slightly more parameters than the model without graph encoders (57.6M vs 61.7M).

**Impact of Graph Size, Arity and Sentence Length** The good overall performance on the datasets shows the superiority of using graph encoders and dual representations over the sequential encoder. However, we are also interested in estimating the performance of the models concerning different data properties. In order to evaluate how the models handle graph and sentence features, we perform an inspection based on different sizes of graph diameter, sentence length, and max node out-degree. Table 5 shows METEOR[5] scores for the LDC2017T10 dataset.

| Model | Graph Diameter | | | | | |
|---|---|---|---|---|---|---|
| | 0-7 | Δ | 7-13 | Δ | 14-20 | Δ |
| S2S | 33.2 | | 29.7 | | 28.8 | |
| G2S-GIN | 35.2 | +6.0% | 31.8 | +7.4% | 31.5 | +9.2% |
| G2S-GAT | 35.1 | +5.9% | 32.0 | +7.8% | 31.5 | +9.51% |
| G2S-GGNN | 36.2 | +9.0% | 33.0 | +11.4% | 30.7 | +6.7% |
| | **Sentence Length** | | | | | |
| | 0-20 | Δ | 20-50 | Δ | 50-240 | Δ |
| S2S | 34.9 | | 29.9 | | 25.1 | |
| G2S-GIN | 36.7 | +5.2% | 32.2 | +7.8% | 26.5 | +5.8% |
| G2S-GAT | 36.9 | +5.7% | 32.3 | +7.9% | 26.6 | +6.1% |
| G2S-GGNN | 37.9 | +8.5% | 33.3 | +11.2% | 26.9 | +6.8% |
| | **Max Node Out-degree** | | | | | |
| | 0-3 | Δ | 4-8 | Δ | 9-18 | Δ |
| S2S | 31.7 | | 30.0 | | 23.9 | |
| G2S-GIN | 33.9 | +6.9% | 32.1 | +6.9% | 25.4 | +6.2% |
| G2S-GAT | 34.3 | +8.0% | 32.0 | +6.7% | 22.5 | -6.0% |
| G2S-GGNN | 35.0 | +10.3% | 33.1 | +10.4% | 22.2 | -7.3% |

Table 5: METEOR scores and differences to the S2S, in the LDC2017T10 test set, with respect to the graph diameter, sentence length and max node out-degree.

The performances of all models decrease as the diameters of the graphs increase. G2S-GGNN has a 17.9% higher METEOR score in graphs with a diameter of at most 7 compared to graphs with diameters higher than 13. This is expected as encoding a bigger graph (containing more information) is harder than encoding smaller graphs. Moreover, 71% of the graphs in the training set have a diameter less than or equal to 7 and only 2% have a diameter bigger than 13 (see Figure 3). Since the models have fewer examples of bigger graphs to learn from, this also leads to worse performance when handling graphs with higher diameters. We also investigate the performance with respect to the sentence length. The models have better results when handling sentences with 20 or fewer tokens. Longer sentences pose additional challenges to the models.

G2S-GIN has a better performance in handling graphs with node out-degrees higher than 9. This indicates that GINs can be employed in tasks where the distribution of node degrees has a long

[4] https://github.com/jflanigan/jamr

[5] METEOR score is used as it is a sentence-level metric.

| | REF ⇒ GEN | | |
|---|---|---|---|
| **Model** | **ENT** | **CON** | **NEU** |
| S2S | 38.45 | 11.17 | 50.38 |
| G2S-GIN | 49.78 | 9.80 | 40.42 |
| G2S-GAT | 49.48 | 8.09 | 42.43 |
| G2S-GGNN | 51.32 | 8.82 | 39.86 |
| | GEN ⇒ REF | | |
| **Model** | **ENT** | **CON** | **NEU** |
| S2S | 73.79 | 12.75 | 13.46 |
| G2S-GIN | 76.27 | 10.65 | 13.08 |
| G2S-GAT | 77.54 | 8.54 | 13.92 |
| G2S-GGNN | 77.64 | 9.64 | 12.72 |

Table 6: Entailment (ENT), contradiction (CON) and neutral (NEU) average percentages for the LDC2017T10 test set. **(Top)** The premise and the hypothesis are the generated (GEN) and reference (REF) sentences, respectively. **(Bottom)** The hypothesis and the premise are the generated (GEN) and reference (REF) sentences, respectively.

tail. Surprisingly, S2S has a better performance than G2S-GGNN and G2S-GAT when handling graphs that contain high degree nodes.

**Semantic Equivalence** We perform an entailment experiment using BERT (Devlin et al., 2019) fine-tuned on the MultiNLI dataset (Williams et al., 2018) as a NLI model. We are interested in exploring whether a generated sentence (hypothesis) is semantically *entailed* by the reference sentence (premise). In a related text generation task, Falke et al. (2019) employ NLI models to rerank alternative predicted abstractive summaries.

Nevertheless, uniquely verifying whether the reference (REF) entails the generated sentence (GEN) or vice-versa (GEN entails REF) is not sufficient. For example, suppose that *"Today Jon walks"* is the REF and *"Jon walks"* is the GEN. Even though REF entails GEN, GEN does not entail REF, that is, GEN is too general (missing information). Furthermore, suppose that *"Jon walks"* is the REF and *"Today Jon walks"* is the GEN, GEN entails REF but REF does not entail GEN, that is, GEN is too specific (added information). Therefore, in addition to verify whether the reference entails the generated sentence, we also verify whether the generated sentence entails the reference.

Table 6 shows the average probabilities for entailment, contradiction and neutral classes on the LDC2017T10 test set. All G2S models have higher entailment compared to S2S. G2S-GGNN has 33.5% and 5.2% better entailment performances than S2S, when REF entails GEN and GEN entails REF, respectively. G2S models also generate sentences that contradict the reference sentences less. This suggests that our models are capable of capturing better semantic information from the graph generating outputs semantically related to the reference sentences.

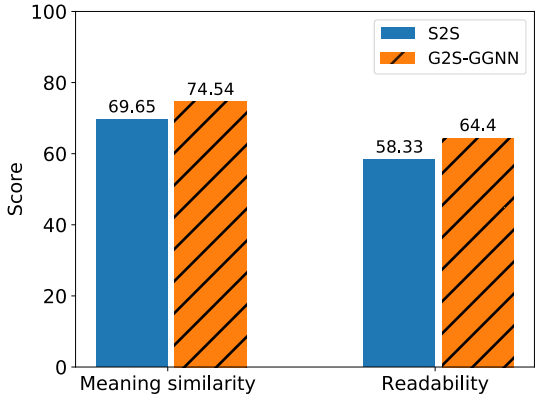


Figure 4: Human evaluation of the sentences generated by S2S and G2S-GGNN models. Results are statistically significant with $p < 0.05$, using Wilcoxon rank-sum test.

**Human Evaluation** To further assess the quality of the generated sentences, we conduct a human evaluation. We employ the *Direct Assessment* (DA) method (Graham et al., 2017) via Amazon Mechanical Turk. Using the DA method inspired by Mille et al. (2018), we assess two quality criteria: (i) *meaning similarity*: how close in meaning the generated text is to the gold sentence; and (ii) *readability*: how well the generated sentence reads (Is it good fluent English?).

We randomly select 100 sentences generated by S2S and G2S-GGNN and randomly assign them to HITs (following Mechanical Turk terminology). Human workers rate the sentences according to meaning similarity and readability on a 0-100 rating scale. The tasks are executed separately and workers were first given brief instructions. For each sentence, we collect scores from 5 workers and average them. Models are ranked according to the mean of sentence-level scores. We apply a quality control step filtering workers who do not score some faked and known sentences properly.

Figure 4 shows the results. In both metrics, G2S-GGNN has better human scores for meaning similarity and readability, suggesting a higher

| | |
|---|---|
| (a / agree :ARG0 (a2 / and :op1 (c / country :wiki China :name (n / name :op1 China)) :op2 (c2 / country :wiki Kyrgyzstan :name (n2 / name :op1 Kyrgyzstan))) :ARG1 (t / threaten-01 :ARG0 (a3 / and :op1 (t2 / terrorism) :op2 (s / separatism) :op3 (e / extremism)) :ARG2 (a4 / and :op1 (s3 / security :mod (r / region)) :op2 (s4 / stability :mod r)) :time (s2 / still) :ARG1-of (m / major-02)) :medium (c3 / communique :mod (j / joint))) | |
| GOLD | China and Kyrgyzstan agreed in a joint communique that terrorism, separatism and extremism still pose major threats to regional security and stability. |
| S2S | In the joint communique, China and Kyrgyzstan still agreed to threaten terrorism, separatism, extremism and regional stability. |
| Song et. al (2018) | In a joint communique, China and Kyrgyzstan have agreed to still be a major threat to regional security, and regional stability. |
| G2S-GGNN | At a joint communique, China and Kyrgyzstan agreed that terrorism, separatism and extremism are still a major threat to region security and stability. |

Table 7: An example of an AMR graph and generated sentences. GOLD refers to the reference sentence.

| Model | ADDED | MISS |
|---|---|---|
| S2S | 47.34 | 37.14 |
| G2S-GIN | 48.67 | 33.64 |
| G2S-GAT | 48.24 | 33.73 |
| G2S-GGNN | 48.66 | 34.06 |
| GOLD | 50.77 | 28.35 |

Table 8: Fraction of elements in the output that are not present in the input (ADDED) and the fraction of elements in the input graph that are missing in the generated sentence (MISS), for the test set of LDC2017T10. The token lemmas are used in the comparison. GOLD refers to the reference sentences.

quality of the generated sentences regarding S2S. The Pearson correlations between meaning similarity and readability scores, and METEOR[6] scores are 0.50 and 0.22, respectively.

**Semantic Adequacy** We also evaluate the semantic adequacy of our model (how well does the generated output match the input?) by comparing the number of added and missing tokens that occur in the generated versus reference sentences (GOLD). An added token is one that appears in the generated sentence but not in the input graph. Conversely, a missing token is one that occurs in the input but not in the output. In GOLD, added tokens are mostly function words while missing tokens are typically input concepts that differ from the output lemma. For instance, in Figure 1, *there* and *of* are added tokens while *person* is a missing token. As shown in Table 8, G2S approaches outperform the S2S baseline. G2S-GIN is closest to GOLD with respect to both metrics suggesting that this model is better able to generate novel words to construct the sentence and captures a larger range of concepts from the input AMR graph, covering more information.

**Manual Inspection** Table 7 shows sentences generated by S2S, Song et al. (2018), G2S-GAT, and the reference sentence. The example shows that our approach correctly verbalises the subject of the embedded clause *"China and ... agreed that terrorism, separatism and extremism$_{SUBJ}$ ... pose major threats to ..."*, while S2S and Song et al. (2018) are fooled by the fact that *agree* frequently takes an infinitival argument which shares its subject (*"China ...$_{SUBJ}$ agreed to threaten / have agreed to be a major threat"*). While this is a single example, it suggests that dual encoding enhances the model ability to take into account the dependencies and the graph structure information, rather than the frequency of n-grams.

## 6 Conclusion

We have studied the problem of generating text from AMR graphs. We introduced a novel architecture that explicitly encodes two parallel and adjuvant representations of the graph (top-down and bottom-up). We showed that our approach outperforms state-of-the-art results in AMR-to-text generation. We provided an extensive evaluation of our models and demonstrated that they are able to achieve the best performance. In the future, we will consider integrating deep generative graph models to express probabilistic dependencies among AMR nodes and edges.

## Acknowledgments

This work has been supported by the German Research Foundation as part of the Research Training Group Adaptive Preparation of Information from Heterogeneous Sources (AIPHES) under grant No. GRK 1994/1.

[6]METEOR score is used as it is a sentence-level metric.

## References


Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria. Association for Computational Linguistics.

Daniel Beck, Gholamreza Haffari, and Trevor Cohn. 2018. Graph-to-sequence learning using gated graph neural networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 273–283, Melbourne, Australia. Association for Computational Linguistics.

Anja Belz, Mike White, Dominic Espinosa, Eric Kow, Deirdre Hogan, and Amanda Stent. 2011. The first surface realisation shared task: Overview and evaluation results. In *Proceedings of the Generation Challenges Session at the 13th European Workshop on Natural Language Generation*, pages 217–226, Nancy, France. Association for Computational Linguistics.

Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. 2017. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42.

Kris Cao and Stephen Clark. 2019. Factorising AMR generation through syntax. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2157–2163, Minneapolis, Minnesota. Association for Computational Linguistics.

Thiago Castro Ferreira, Iacer Calixto, Sander Wubben, and Emiel Krahmer. 2017. Linguistic realisation as machine translation: Comparing different MT models for AMR-to-text generation. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 1–10, Santiago de Compostela, Spain. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Marco Damonte and Shay B. Cohen. 2019. Structural neural encoders for AMR-to-text generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3649–3658, Minneapolis, Minnesota. Association for Computational Linguistics.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 376–380, Baltimore, Maryland, USA. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Tobias Falke, Leonardo F. R. Ribeiro, Prasetya Ajie Utama, Ido Dagan, and Iryna Gurevych. 2019. Ranking generated summaries by correctness: An interesting but challenging application for natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2214–2220, Florence, Italy. Association for Computational Linguistics.

Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime Carbonell. 2016a. CMU at SemEval-2016 task 8: Graph-based AMR parsing with infinite ramp loss. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1202–1206, San Diego, California. Association for Computational Linguistics.

Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime Carbonell. 2016b. Generation from abstract meaning representation using tree transducers. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 731–739, San Diego, California. Association for Computational Linguistics.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The WebNLG challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133, Santiago de Compostela, Spain. Association for Computational Linguistics.

Yvette Graham, Timohy Baldwin, Alistair Moffat, and Justin Zobel. 2017. Can machine translation systems be evaluated by the crowd alone. *Natural Language Engineering*, 23(1):3–30.

Zhijiang Guo, Yan Zhang, Zhiyang Teng, and Wei Lu. 2019. Densely connected graph convolutional networks for graph-to-sequence learning. *Transactions*

*of the Association for Computational Linguistics*, 7:297–312.

Diederick P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, San Diego, CA, USA.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text Generation from Knowledge Graphs with Graph Transformers. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2284–2293, Minneapolis, Minnesota. Association for Computational Linguistics.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural amr: Sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 146–157, Vancouver, Canada. Association for Computational Linguistics.

Yujia Li, Richard Zemel, Marc Brockschmidt, and Daniel Tarlow. 2016. Gated graph sequence neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, San Juan, Puerto Rico.

Kexin Liao, Logan Lebanoff, and Fei Liu. 2018. Abstract meaning representation for multi-document summarization. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1178–1190, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Diego Marcheggiani and Laura Perez Beltrachini. 2018. Deep graph convolutional encoders for structured data to text generation. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 1–9, Tilburg University, The Netherlands. Association for Computational Linguistics.

Simon Mille, Anja Belz, Bernd Bohnet, Yvette Graham, Emily Pitler, and Leo Wanner. 2018. The first multilingual surface realisation shared task (SR'18): Overview and evaluation results. In *Proceedings of the First Workshop on Multilingual Surface Realisation*, pages 1–12, Melbourne, Australia. Association for Computational Linguistics.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, AKBC-WEKEX '12, pages 95–100, Stroudsburg, PA, USA. Association for Computational Linguistics.

Shashi Narayan and Claire Gardent. 2012. Structure-driven lexicalist generation. In *Proceedings of COLING 2012*, pages 2027–2042, Mumbai, India. The COLING 2012 Organizing Committee.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Nima Pourdamghani, Kevin Knight, and Ulf Hermjakob. 2016. Generating English from abstract meaning representations. In *Proceedings of the 9th International Natural Language Generation conference*, pages 21–25, Edinburgh, UK. Association for Computational Linguistics.

Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*, pages 593–607.

Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Stuart M. Shieber, Gertjan van Noord, Fernando C. N. Pereira, and Robert C. Moore. 1990. Semantic-head-driven generation. *Computational Linguistics*, 16(1):30–42.

Linfeng Song, Daniel Gildea, Yue Zhang, Zhiguo Wang, and Jinsong Su. 2019. Semantic neural machine translation using AMR. *Transactions of the Association for Computational Linguistics*, 7:19–31.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. A graph-to-sequence model for AMR-to-text generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1616–1626, Melbourne, Australia. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *International Conference on Learning Representations*, Vancouver, Canada.

Boris Weisfeiler and A.A. Lehman. 1968. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsia*, pages 12–16.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks? In *International Conference on Learning Representations*, New Orleans, LA, USA.

# Keeping Consistency of Sentence Generation and Document Classification with Multi-Task Learning

**Toru Nishino** **Shotaro Misawa** **Ryuji Kano**
**Tomoki Taniguchi** **Yasuhide Miura** **Tomoko Ohkuma**
Fuji Xerox Co., Ltd.
{nishino.toru, misawa.shotaro, kano.ryuji,
taniguchi.tomoki, yasuhide.miura, ohkuma.tomoko}
@fujixerox.co.jp

## Abstract

The automated generation of information indicating the characteristics of articles such as headlines, key phrases, summaries and categories helps writers to alleviate their workload. Previous research has tackled these tasks using neural abstractive summarization and classification methods. However, the outputs may be inconsistent if they are generated individually. The purpose of our study is to generate multiple outputs consistently. We introduce a multi-task learning model with a shared encoder and multiple decoders for each task. We propose a novel loss function called *hierarchical consistency loss* to maintain consistency among the attention weights of the decoders. To evaluate the consistency, we employ a human evaluation. The results show that our model generates more consistent headlines, key phrases and categories. In addition, our model outperforms the baseline model on the ROUGE scores, and generates more adequate and fluent headlines.

## 1 Introduction

Headlines and other information such as key phrases, summaries and categories about articles are crucial for readers to search articles on demand. To attract more readers, writers manually create headlines and summaries by summarizing the articles, extract key phrases and classify articles into categories. Figure 1 shows an example of job advertisement articles. In addition to the job description text, the headline, key phrase and category are labeled for each article. Thus job seekers can easily retrieve the job advertisements they desire by reading them. However, it is a burden for writers to create these headlines, key phrases and categories manually for extremely large numbers of articles. Hence, an automatic generation system is highly demanded.

| |
|---|
| **Headline:** We Want Android[1] Engineer to Develop our New Service "Wantedly People[2] "! |
| **Key Phrase:** Android Engineer |
| **Category:** Engineer |
| **Description Text (Truncated):** We released a new service "Wantedly People" in November 2016, and released new features in July 2017! Our "People team", developing rapidly growing "Wantedly People" service, is recruiting an Android Engineer! If you love to work on new services, or if you want to try to develop new apps as your representative work, we are looking forward to your application! |

Figure 1: An example job advertisement article with consistency. Each article contains a headline, key phrase and category. The underlined topic "Engineer" is consistently noted in the headline, key phrase and category. A Japanese-English translation is applied.

For automated generation of multiple outputs, consistency among outputs is crucial. A lack of consistency among outputs causes incorrect information in the outputs. In Figure 1, for example, the "Engineer" position is consistently noted in the headline, key phrase and category. An occupation is consistent and salient information for headlines, key phrases and categories. If the article was to be misclassified as a "Designer" category or the key phrase wrongly noted as "Robotics Engineer," an inconsistency among the headline, key phrase and category would occur. Thus, readers would be confused by these inconsistencies. We must force generators to predict multiple outputs consistently. This leads to the correctness of the occupation in the outputs, and thus the quality of the generated outputs also improves.

In previous research, neural networks have achieved significant improvements in individual tasks, such as abstractive summarization (Rush et al., 2015; See et al., 2017; Shi et al., 2018),

[1]Android is a trademark of Google LLC.
[2]Wantedly People is a web service provided by Wantedly, Inc.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3195–3205,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

headline generation (Takase et al., 2016), key phrase generation (Meng et al., 2017) and text classification (Zhang et al., 2015) tasks. However, consistency among multiple outputs is not considered in the strategy to predict multiple outputs with separate models.

The purpose of our study is to maintain consistency among automatic generated sentences and classified categories. We adopt multi-task learning (Caruana, 1997) to predict the headlines, key phrases and categories of articles in one unified model. We handle the key phrase generation and classification tasks not as auxiliary tasks, but as the desired outputs of our system. Multi-task learning enables the encoder to focus on the common and salient features in the input text.

We propose a novel *hierarchical consistency loss* to maintain consistency among the multiple outputs. A hierarchical consistency loss forces the attention weights of the decoders to focus on the same words in the input text, considering the hierarchical relation among tasks. There is a hierarchical relation among the tasks; the headline generator generally focuses on the wider range of words including the key phrase generator focus upon. In addition, this loss has a flexibility that alleviate the influences of errors propagated from other tasks, similar to soft-parameter sharing methods (Guo et al., 2018).

We design human evaluations using crowdsourcing to score the following three metrics: fluency, adequacy and consistency among the outputs. We implement human evaluations of the job advertisement dataset, and the results indicate that our model improves not only the consistency score but also the fluency and adequacy scores.

In addition, we conduct automatic evaluations of the job advertisement dataset and the modified CNN-DailyMail (CNN-DM) dataset (Nallapati et al., 2016). The automatic evaluations show that our method improves the ROUGE metric scores on both datasets, which has multiple outputs.

Overall, our contributions are as follows:

- We propose a multi-task sentence generation and document classification model.
- A novel *hierarchical consistency loss* is introduced to train the weights of attention to focus more on the same part of the input text among the task-specific decoders.

Our designed human evaluations show that our model generates more consistent outputs. Our proposed model generates more adequate and fluent outputs on a human evaluation, and achieves the best ROUGE score on an automatic evaluation.

## 2 Related Work

**Abstractive summarization.** Abstractive summarization is a task to generate a short summary that captures the core meaning of the original text. Rush et al. (2015) used a neural attention model, and See et al. (2017) introduced a pointer-generator network to copy out-of-vocabulary (OOV) words from the input text. Hsu et al. (2018) combined abstractive and extractive summarization with an inconsistency loss to encourage consistency between word-level attention weights of the abstracter and sentence-level attention weights of the extractor. Abstractive summarization techniques are generally applied to a headline generation because this is a similar task (Shen et al., 2017; Tan et al., 2017).

**Multi-task learning.** Multi-task learning, which trains different tasks in one unified model, has achieved success in many natural language processing tasks (Luong et al., 2016; Hashimoto et al., 2017; Liu et al., 2019). Typical multi-task learning models have a structure with a shared encoder to encode the input text and multiple decoders to generate outputs of each task. Multi-task learning has a benefit in that the shared encoder captures common features among tasks; in addition, the encoder focuses more on relevant and beneficial features, and disregards irrelevant and noisy features (Ruder, 2017).

Although a multi-task learning model is beneficial in training a shared encoder, it is still difficult to share information among task-specific decoders. Some studies have constructed a multi-task learning model using techniques that encourages information sharing among decoders. Isonuma et al. (2017) proposed an extractive summarization model that the outputs of the sentence extractor are directly used for a document classifier. Anastasopoulos and Chiang (2018) introduced a triangle model to transfer the decoder information of the second task to the decoder of the first task. Tan et al. (2017) introduced a coarse-to-fine model to generate headlines using important sentences chosen in the extracter. These methods are cascade models that additionally input the information of the first tasks directly into

the second tasks. They consider the hierarchy among tasks, but these models suffer from the errors of the previous tasks.

Guo et al. (2018) proposed a decoder sharing method with soft-parameter sharing to train the summarization and entailment tasks. Soft-parameter sharing has a benefit in that it provides more flexibility between the layer of summarization and entailment tasks; however, this method does not consider the hierarchy among tasks.

Our study extends the method in Hsu et al. (2018) to a multi-task learning model in which the models need to generate multiple outputs with consistency. Hierarchical consistency loss combines two advantages. This loss considers the hierarchy among tasks, and has flexibility among tasks, similar to soft-parameter sharing methods. We assess the advantages of this loss in Section 4.2.

## 3 Method

### 3.1 Problem Definition

We define the tasks of our study and describe the overview of the datasets.

Let $x = \{x_1, x_2, ...x_S\}$ be a sequence of input text. The target of our multi-task model is to generate two types of sentences and to predict the category of the input article. Our model predicts the exactly one category tag $y^1$ for each input sentence. Our model also predicts $y^2 = \{y^2_1, y^2_2, ...y^2_{T^2}\}$ and $y^3 = \{y^3_1, y^3_2, ...y^3_{T^3}\}$, which are the sequence of sentences.

For the job advertisement dataset, the targets of our model are to classify articles into occupation categories $y^1$ (task 1), generate key phrases regarding the occupation $y^2$ (task 2) and generate headlines $y^3$ (task 3). For the CNN-DM dataset, the targets are to predict the article categories $y^1$ (task 1), headlines $y^2$ (task 2) and multi-sentence summaries $y^3$ (task 3).

Here, $S$ is the length of the input texts, and $T^2$ and $T^3$ are the lengths of the output sequences, respectively. $T^2$ is generally smaller than $T^3$ in both datasets. Hence, task 3 is generally more difficult than task 2, and more information is needed to generate $y^3$ than $y^2$.

### 3.2 Encoder-Decoder model

**Encoder-decoder model with attention mechanism.** Our model is based on an encoder-decoder model (Cho et al., 2014). The encoder RNN transforms the input text into hidden vectors $h^e = \{h^e_1, h^e_2, ..., h^e_S\}$, and the decoder RNN then predicts the generation probability of each word $P_{vocab,t}$:

$$h^e_t = \mathrm{RNN}_{enc}(x_t, h^e_{t-1}) \quad (1)$$

$$h^d_t = \mathrm{RNN}_{dec}(y_{t-1}, h^d_{t-1}, h^e_S) \quad (2)$$

$$P_{vocab,t} = \mathrm{softmax}(W_{d2v} h^d_t + b_{d2v}) \quad (3)$$

where the weight matrices $W_{d2v}$ and bias vector $b_{d2v}$ are trainable parameters.

Rush et al. (2015) used an attention mechanism to handle long input sentences. The attention mechanism obtains the hidden vector of attention $\bar{h}^d_t$ from the hidden vector and context vector $c^e_t$, which is defined as the weighted sum of the hidden vectors of the encoder:

$$e^e_{tj} = v^T \tanh(W_e h^e_j + W_d h^d_t + b_{attn}) \quad (4)$$

$$\alpha^e_{tj} = \mathrm{softmax}(e^e_{tj}) \quad (5)$$

$$c^e_t = \sum_j \alpha^e_{tj} h^e_j \quad (6)$$

$$\bar{h}^d_t = W_c[h^d_t, c^e_t] + b_c \quad (7)$$

Note that $[h^d_t, c^e_t]$ indicates the concatenation of vectors $h^d_t$ and $c^e_t$. Weight matrices $W_e$, $W_d$ and $W_c$ and the bias vectors $b_{attn}$ and $b_c$ are trainable parameters.

**Pointer-generator network.** We adopt a pointer-generator network (See et al., 2017) for the decoders to handle OOV words. The decoder generates words under the probability of $p_{gen,y_t}$ and copies words from the input sentence under the probability of $1 - p_{gen,y_t}$.

**Coverage mechanism.** See et al. (2017) also introduced a coverage mechanism to alleviate the repetition problem. The coverage loss $L_{cov}$ is added to the loss function to penalize the attention mechanism to avoid focusing on the same input words.

### 3.3 Multi-Task Learning for Generation and Classification

We introduce multi-task learning to predict multiple outputs simultaneously in one unified model. Figure 2 describes an overview of our multi-task learning model. A multi-task learning model comprises one shared layer, two task-specific decoders for generation tasks, and one classifier.

**Shared encoder.** First, shared encoder $RNN_{enc}$ transforms the input text into the shared hidden

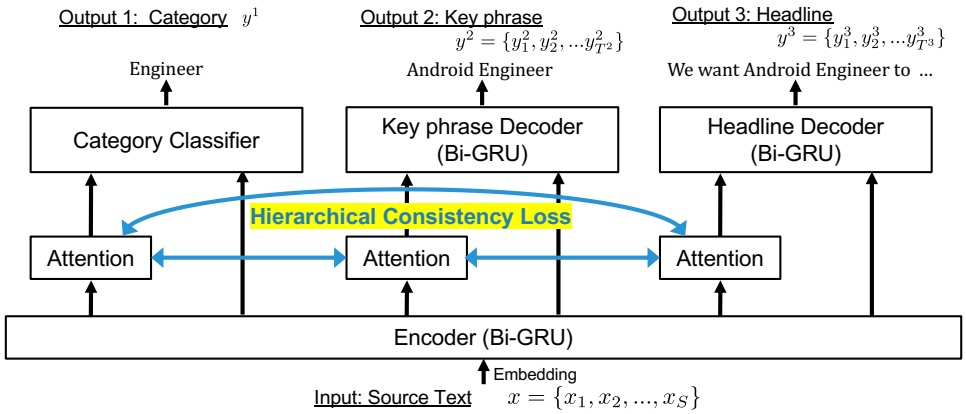


Figure 2: An overview of our multi-task learning model. The shared embedding layer and encoder first transform the input text into a shared hidden vector, and the task-specific layers then predict the outputs of each task. Hierarchical consistency loss penalizes the inconsistency between the attention weights of the pairs of decoders.

vector $h_t^e$. We use a 2-layer bi-directional GRU (Cho et al., 2014) for the encoder.

**Classifier.** The classifier transforms a shared hidden vector into the category probability $P_{cat}$. We implement the classifier as 2-layer perceptron with an attention mechanism:

$$o^c = \mathrm{ReLU}(W_1^c([h_S^e, c_t^e] + b_1^c)) \quad (8)$$

$$P_{cat} = \mathrm{softmax}(W_2^c o^c + b_2^c) \quad (9)$$

where the weight matrices $W_1^c$ and $W_2^c$, and bias vectors $b_1^c$ and $b_2^c$ are trainable parameters.

**Decoders.** The hidden vector of encoder $h_S^e$ is first transformed into $\hat{h_0^d}$ in the bridge layer, and the decoders $RNN_{dec}$ of each task then generate output sequences $y^2$ and $y^3$ from $\hat{h_0^d}$. A bridge is an additional fully connected layer used to fit the hidden vector into each task. We expect that multi-task learning will enable the shared encoder to capture more common and salient parts of an article, that is, the parts of the text that mention the occupation.

### 3.4 Hierarchical Consistency Loss

The main objective of our method is to maintain consistency among multiple outputs. If each decoder focuses on the same word in the input text, the model can generate a more consistent output. Hence, consistency between attention weights leads to consistency between multiple outputs. For example, in Figure 3, inconsistency regarding the occupation occurs because the word "engineer" is inconsistently focused on the attention weights of the key phrase generator. From this, the key phrase generator predicts an incorrect key phrase. By penalizing such inconsistencies, the model enables the generation of more consistent outputs.

However, perfect consistency between attention weights occasionally disturbs the model to generate proper outputs. Because headlines contain more information than key phrases, the headline generator must focus on a wider range of words in the input text than the key phrase generator. For example, in Figure 3, the headline generator focuses on the words "user interface." In contrast, the key phrase generator does not focus on these words. Therefore, the attention weights among tasks generally maintain a hierarchical relation. That is, the higher tasks (headline generation and summarization) need to focus on a wider range of words, and the lower tasks (classification) focus on the range on which the higher tasks focused.

We introduce a novel "hierarchical consistency loss" to penalize inconsistency among multiple outputs. We define hierarchical consistency loss between task $s$ and task $t$ as follows:

$$L_{hcl}^{st} = \frac{\lambda_{hcl}}{S} \sum_{i=1}^{S} |\max_j e_{ij}^{ds} - \max_j e_{ij}^{dt}|_+ \quad (10)$$

where $e_{ij}^{ds}$ is the non-normalized attention weight of the output word $j$ toward the input word $i$. Note that a ramp function $|x|_+$ is used to compare two attention weights. Task $t$ is a task that needed to focus on a wider range of words than the input task $s$. For example, in Figure 3, task $s$ is a key phrase generation, and task $t$ is a headline generation.

The aim of hierarchical consistency loss is to penalize inconsistency between two attention weights. For example, in Figure 3, the loss forces the attention weight corresponding to the word

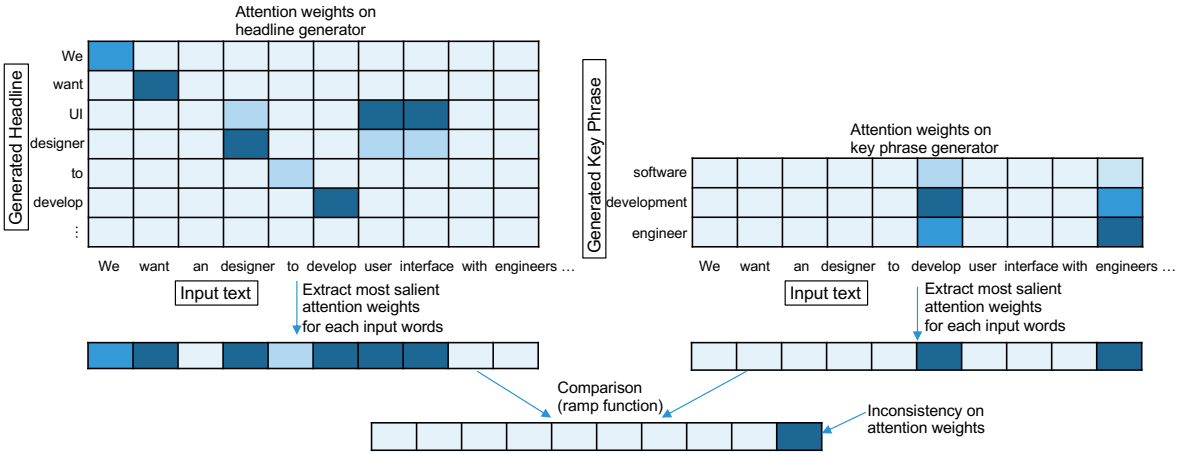


Figure 3: An overview illustration of our hierarchical consistency loss. This loss penalizes a case in which inconsistency occurs between two outputs. The decoder of the key phrase generator focuses on the word "engineers." However, the headline generation decoder does not focus on this word. Hierarchical consistency loss treats this difference as an inconsistency.

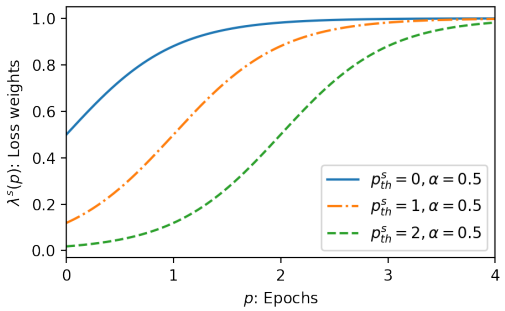


Figure 4: An illustration of our training scheduling strategy. The loss weights $\lambda^s(p)$ gradually increase as the training proceeds.

"engineers" to a higher weight because the word "engineers" is crucial to generate consistent and adequate outputs. Figure 3 shows the process of the loss. The most salient attention weights for each input word are extracted, and the extracted attention weights between two tasks are then compared. A ramp function enables this loss to consider the hierarchical relation between two tasks.

### 3.5 Learning Scheduling for Multi-Task Learning

We also introduce a new multi-task learning scheduling strategy to effectively train several tasks with different levels of difficulty. Kiperwasser and Ballesteros (2018) introduced a strategy that gradually changes the probability that the training examples will be picked from each dataset as the training proceeds. We modify this idea to a situation in which the input text is in common among multiple tasks, adjusting the hyperparameters of the weighted sum of loss functions. This strategy enables us to train our model in such a way that our model focuses on learning easy tasks at the beginning, and then gradually focuses more on learning difficult tasks.

We determine the weights of the loss function $\lambda^s(p)$ for task s with a sigmoid function:

$$\lambda^s(p) = \lambda^s_{const} \frac{1}{1 + \exp((p^s_{th} - p)/\alpha)} \quad (11)$$

where $\lambda^s_{const}$ and $p^s_{th}$ are hyperparameters for each task. Parameter $p$ describes the number of epochs trained thus far. We set hyperparameter $\alpha$ to 0.5 for all tasks. As illustrated in Figure 4, $\lambda^s(p)$ becomes larger as the training proceeds.

### 3.6 Overall Loss Function

The overall loss function of the proposed model is calculated as the weight sum of losses from three tasks, coverage losses and hierarchical consistency loss:

$$\begin{aligned} L_{all} &= \lambda^1(p)L_1 + \lambda^2(p)L_2 + \lambda^3(p)L_3 \\ &+ \lambda^2_{cov}L^2_{cov} + \lambda^3_{cov}L^3_{cov} \\ &+ \lambda^{all}_{hcl}L^{all}_{hcl} \end{aligned} \quad (12)$$

where $\lambda^1(p)$, $\lambda^2(p)$, $\lambda^3(p)$, $\lambda^2_{cov}$, $\lambda^3_{cov}$ and $\lambda^{all}_{hcl}$ are hyperparameters. In addition, $\lambda^1(p)$, $\lambda^2(p)$, and $\lambda^3(p)$ are calculated using Eqn.11. Moreover, $L_1$, $L_2$ and $L_3$ signify the loss of the classification, key phrase generation and headline generation tasks, respectively. $L^2_{cov}$ and $L^3_{cov}$ are the coverage losses

of two generators, and $L_{hcl}^{all}$ is the sum of the hierarchical consistency losses:

$$L_{hcl}^{all} = L_{hcl}^{12} + L_{hcl}^{13} + L_{hcl}^{23} \quad (13)$$

where $L_{hcl}^{st}$ indicates the hierarchical consistency loss between task $s$ and $t$.

## 4 Experiments and Results

### 4.1 Experimental Settings

We use two datasets, namely, a job advertisement dataset and the CNN-DailyMail (CNN-DM) dataset. We evaluate the headline generation, key phrase generation and classification tasks in these datasets. As an example, Figure 1 shows the headlines, key phrases and categories, which generally include common features, that is, the words mentioning occupations.

We also use the CNN-DM dataset (See et al., 2017) in an automatic evaluation under different task settings. We evaluate the multi-sentence summarization, headline generation and classification tasks in this dataset. Note that we train and evaluate the CNN and DailyMail datasets separately because they have different taxonomies.

The original CNN-DM dataset does not contains headlines or categories, and thus we extract the headlines and categories from the raw HTMLs. The Supplementary section describes the details and modification of the CNN-DM dataset.

We employ pointer-generator models without multi-task learning as baselines of the sentence generators, and a 2-layer GRU with attention as the baseline of the classifier. We chose all hyperparameters based on the learning curve and the scores of the validation set. Details of our model, such as the hyperparameters of the RNNs, the weights of the loss function and the optimizers, are described in the Supplementary section for reproducibility.

### 4.2 Results

**Human evaluation of job advertisement dataset.** We conduct a human evaluation of the job advertisement dataset to measure the quality of the generated outputs. Ten crowd-sourcing workers measure 250 randomly selected samples. We defined the following four metrics:

- Consistency: Measure whether the outputs include the same occupation. We choose a pair of outputs, and if both outputs mention the same type of occupation, we regard this pair as consistent.
- Adequacy: Measure how well the headline describes the correct information.
- Fluency: Measure how natural the generated headline is.
- Occupation Adequacy: Measure how well the headline mentions the correct occupation. For example, in Figure 1, if the generated headline implies that the occupation "engineer" is recruited, we regard this headline as adequate regarding the occupation. We assume that consistency of the occupation improves the score of the occupation adequacy.

| | HG-KG | HG-CC | KG-CC | 3 Outputs |
|---|---|---|---|---|
| Baseline | 56.8% | 37.6% | 37.6% | 30.0% |
| Proposed | **58.8%** | **39.6%** | **39.2%** | **32.4%** |
| Gold | 65.2% | 44.4% | 48.8% | 35.2% |

Table 1: Comparison of human evaluation results with their consistency. HG-KG, HG-CC and KG-CC indicate the consistency scores between the headline generation and key phrase generation, headline generation and category classification and key phrase generation and category classification, respectively.

| | Adequacy | Fluency | Occupation Adequacy |
|---|---|---|---|
| Baseline | 3.34 | 3.69 | 3.45 |
| Proposed | **3.76** | **3.86** | **3.89** |
| Gold | 4.09 | 4.12 | 4.13 |

Table 2: Comparison of human evaluation results for headlines. Scores are an average of ten crowd-sourcing workers with five scale rating.

First, we conduct a human evaluation to measure the consistency among three outputs. Table 1 shows the evaluation results with their consistency. The scores are the percentage of articles evaluated as consistent by a majority of workers. We consider all three outputs as consistent if all pairs of outputs are consistent. The results indicate that the proposed method improves the consistency of the three generated outputs [1].

Second, to evaluate the quality of the generated headline, we implement a human evaluation to measure the adequacy and fluency. Table 2 shows the evaluation result along with the adequacy and fluency. The proposed method improves the

[1]Note that the consistency of HG-CC and KG-CC is relatively low because some categories are too abstractive (for example, "other types of engineers").

| | Headline generation | | | Key phrase generation | | | Classification |
|---|---|---|---|---|---|---|---|
| | R-1 | R-2 | R-L | R-1 | R-2 | R-L | Accuracy |
| Baseline (Pointer-Generator Network) | 25.1 | 5.3 | 21.1 | 30.9 | 10.6 | 28.7 | 62.8 |
| Multi-Task Learning (MTL) | 26.2 | 5.8 | 21.6 | 32.3 | 10.9 | 30.0 | 64.1 |
| MTL + Scheduling (SD) | 26.3 | 6.0 | 21.8 | 32.3 | 10.4 | 29.9 | 63.9 |
| Proposed (MTL + SD + Hierarchical Consistency Loss (HCL)) | $*$**26.9** | $*$**6.1** | $*$**22.4** | $*$**32.8** | $*$**11.2** | $*$**30.5** | **64.4** |
| Lead-1 | 19.0 | 4.3 | 13.5 | - | - | - | - |

Table 3: Automatic evaluation results based on the ROUGE metrics and accuracy (%) of classification of job advertisement dataset. R-1, R-2 and R-L indicate the $F_1$ scores of ROUGE-1, ROUGE-2 and ROUGE-L, respectively. The proposed method (MTL + SD + HCL) achieved the best scores (bold) for all tasks, and the ROUGE scores are statistically significant from the baseline model ($p < 0.05$) in both the headline and key phrase (indicated as $*$). Lead-1 is the baseline score, which uses the first sentence of the input article as a predicted headline.

| | | Summarization | | | Headline Generation | | | Classification |
|---|---|---|---|---|---|---|---|---|
| | | R-1 | R-2 | R-L | R-1 | R-2 | R-L | Accuracy |
| CNN | Baseline | 30.7 | 10.6 | 27.3 | 19.5 | 5.0 | 17.0 | 43.8 |
| | Proposed (MTL + SD + HCL) | $*$**31.0** | $*$**10.9** | $*$ **27.8** | **19.6** | **5.0** | **17.1** | **43.9** |
| | Lead | 33.4 | 12.2 | 26.1 | 17.2 | 5.0 | 11.1 | - |
| DailyMail | Baseline | 38.4 | 15.8 | 35.0 | 43.1 | 25.3 | 39.6 | 89.0 |
| | Proposed (MTL + SD + HCL) | $*$**38.9** | $*$**16.3** | $*$**35.4** | $*$**43.7** | **25.5** | $*$**40.1** | **89.8** |
| | Lead | 43.8 | 19.2 | 37.3 | 27.7 | 10.9 | 21.7 | - |

Table 4: Automatic evaluation results based on the ROUGE metrics and accuracy (%) of classification of the CNN and DailyMail datasets. The metrics are the same as in Table 3. The proposed method (MTL + SD + HCL) improved the scores for all tasks. The scores with $*$ indicate that the scores are statistically significant from the baseline model ($p < 0.05$). The lead is the score that uses the first three sentences of the input article as a predicted summary, and the first sentence of the input article as a predicted headline.

adequacy by $0.42pt$ and the occupation adequacy by $0.44pt$. Proposed method can generate more adequate outputs, particularly for the occupation.

**Automatic evaluation of job advertisement corpus.** We implement an automatic evaluation using the ROUGE metrics (Lin, 2004) and accuracy. We conduct the experiment ten times, and calculate the average score. Table 3 shows the effect of the proposed methods: multi-task learning (MTL), scheduling strategy (SD) and hierarchical consistency loss (HCL). From this result, the proposed method (MTL + SD + HCL) achieves the best score on all three tasks. MTL and HCL improve for all three tasks, and SD improves the score of the headline generation.

**Automatic evaluation of the CNN-DM dataset.** Table 4 shows the results of the CNN and DailyMail datasets, respectively. For both datasets, the proposed method improves the ROUGE scores of the summarization and headline generation.

From Table 5, headlines and key phrases of the job advertisement dataset have more overlap than the summaries and headlines of the CNN-DM dataset. Therefore, tasks of the job advertisement dataset benefit more from maintaining the consistency. For this reason, the scores of the job advertisement dataset achieve greater improvement than the CNN-DM dataset.

| | ROUGE-1 | ROUGE-2 | ROUGE-L |
|---|---|---|---|
| Job Ads | 48.4 | 15.9 | 47.9 |
| CNN | 42.8 | 13.7 | 38.8 |
| DailyMail | 37.9 | 12.6 | 34.0 |

Table 5: ROUGE recall scores of task 2 (key phrases for the Job Ads, headlines for CNN-DM) gold sentences against task 3 (headlines for Job Ads, summaries for CNN-DM) gold sentences in the validation set. Higher scores indicate that words appearing in the sentence of task 2 are overlapped by the sentence of task 3.

**Comparison of the decoder information sharing methods.** To validate the advantages of our hierarchical consistency loss, we compare five decoder information sharing methods: a cascade model, soft-parameter sharing, non-hierarchical consistency loss, hierarchical consistency loss with normalized attention weights and our hierarchical consistency loss. Table 6 presents the comparison of the five methods for the job advertisement dataset.

The cascade model, similar to Isonuma et al.

| | Headline Generation | | | Key Phrase Generation | | | Classification |
|---|---|---|---|---|---|---|---|
| | R-1 | R-2 | R-L | R-1 | R-2 | R-L | Accuracy |
| Baseline (Pointer-Generator Network) | 25.1 | 5.3 | 21.1 | 30.9 | 10.6 | 28.7 | 62.8 |
| Proposed (MTL + SD + HCL) | **26.9** | **6.1** | **22.4** | **32.8** | **11.2** | **30.5** | 64.4 |
| Comparison of Decoder Information Sharing Method | | | | | | | |
| MTL + SD | 26.3 | 6.0 | 21.8 | 32.3 | 10.4 | 29.9 | 63.9 |
| MTL + SD + Cascade Model | 26.3 | 5.6 | 21.6 | 31.8 | 10.6 | 29.5 | 64.4 |
| MTL + SD + Cascade Model (Gold) | 26.5 | 5.8 | 21.9 | 32.8 | 10.4 | 30.3 | **64.5** |
| MTL + SD + Soft-Parameter Sharing | 25.8 | 5.9 | 21.4 | 32.1 | 10.0 | 29.6 | 64.0 |
| MTL + SD + Non-Hierarchical Consistency Loss | 25.9 | 6.0 | 21.4 | 32.6 | 10.9 | 30.2 | 64.0 |
| MTL + SD + HCL with Normalized Attention Weights | 26.2 | 6.0 | 21.7 | 31.9 | 10.5 | 29.5 | 63.9 |
| Comparison of Encoder Information Sharing Method | | | | | | | |
| HCL (SD and MTL are not applied) | 25.8 | 5.6 | 21.2 | 31.0 | 10.1 | 28.7 | 63.1 |
| SD + HCL (MTL is not applied) | 25.6 | 5.6 | 21.5 | 31.2 | 10.2 | 28.9 | 62.6 |

Table 6: Comparison of the decoder information sharing methods and encoder sharing methods for the job advertisement dataset. The metrics are the same as in Table 3. The proposed method (adopting HCL) achieved the best scores (bold) compared to the other sharing methods.

(2017); Anastasopoulos and Chiang (2018), uses the output of the classifier as an additional input of the headline and key phrase generators. Meanwhile, the cascade model (gold) uses the gold classification category as the additional input of the generators during training and inference.

It can be observed from Table 6 that the cascade model achieves a lower score than MTL + SD. However, the cascade model (gold) improves the ROUGE-L score. This result indicates that the classification error propagates to the generators when the cascade model is applied. Our proposed HCL has an advantage in that it does not suffer from an error of the classifier, and thus this method achieves the best score.

Furthermore, we compare our proposed HCL model with other settings. A soft-parameter sharing method (Guo et al., 2018) penalizes the difference between parameters in pairs of decoders. The non-hierarchical consistency loss is almost the same as the hierarchical consistency loss (Eqn. 10). We replace a ramp function in Eqn. 10 with an absolute value function. Neither the soft-parameter sharing method nor the non-hierarchical consistency loss has the ability to consider the hierarchy among tasks.

It can be observed from Table 6 that both methods achieved a lower score than the hierarchical consistency loss. This is because our hierarchical consistency loss enables the model to penalize a multi-task model with hierarchy among the tasks.

Although the HCL with normalized-attention weights adopts the hierarchical consistency loss model indicated in Eqn. 10, however, we substitute the normalized attention weights $\alpha_{ij}^{d^s}$ for non-normalized attention weights $e_{ij}^{d^s}$.

HCL with normalized attention weights is not as effective as HCL with non-normalized attention weights. Our HCL is based on the assumption that if one specific input word is important for both the shorter and longer text generation tasks, the attention weights of the words for the shorter text generation task would be smaller than the attention weights for the longer text generation task. However, the distributions of the normalized attention weights converge to a few words for the shorter text generation task; thus the normalized attention weights does not satisfy the assumption. As a result, our proposed HCL with non-normalized attention weights can accurately compute this inconsistency, contrary to the HCL with normalized attention weights.

**Comparison of encoder information sharing methods.** To determine the dependence of HCL on MTL, we conduct two experiments. HCL applies the hierarchical consistency loss without MTL and SD, while SD + HCL applies the scheduling sampling and hierarchical consistency loss; however, multi-task learning is not applied. Table 6 presents the results of two methods for the job advertisement dataset.

In comparison with the MTL applied models, both models do not improve the performance of the classification task. Encoder information sharing is beneficial for both the classification and generation tasks, whereas the attention infor-

| Proposed Method (MTL + SD + HCL) | Baseline |
|---|---|
| **Input Articles with Attention Weight (Truncated)**<br>In 2011, Naka, the representative of Wantedly, founded our company alone. Because we first developed the web service (now "Wantedly"), we have used **Ruby on Rails** (...) Our development team is building valuable services with cooperation among engineers (...) Wantedly is at the height of its growth! We are recruiting **super geeks** to develop our "make work enjoyable" services with **Ruby on Rails** to achieve our mission of "Becoming the infrastructure for all workers" as a business SNS! | **Input Articles with Attention Weight (Truncated)**<br>In 2011, Naka, the representative of Wantedly, founded our company alone. Because we first developed the web service (now "Wantedly"), we have used Ruby on Rails (...) **Our development team is building** valuable services with cooperation **among engineers** (...) Wantedly is at the height of its growth! We are recruiting super geeks to develop our "***make work enjoyable***" services with Ruby on Rails to achieve our mission of "Becoming the infrastructure for all workers" as a business SNS! |
| **Generated Headline:**<br>We want a Ruby engineer to develop our own service! | **Generated Headline:**<br>We want a new member to launch new businesses with us! |
| **Generated Key Phrase:**<br>Ruby engineer | **Generated Key Phrase:**<br>Engineer |

Figure 5: An example of an input article, generated headlines and key phrases. Underlined words indicate that the attention weights of the headline generator between these words and the output words are high. **Blue colored words** indicate that the attention weights of these words are consistently high between the two decoders. In contrast, ***red colored italic words*** indicate that the attention weights of these words are high in the key phrase generator, but low in the headline generator, that is, inconsistency occurs between the two decoders. A Japanese-English translation is applied.

| | ROUGE-1 | ROUGE-2 | ROUGE-L |
|---|---|---|---|
| Baseline | 29.1 | 5.5 | 29.0 |
| Proposed | 53.7 | 19.2 | 53.4 |

Table 7: Comparison of ROUGE recall scores of generated key phrases against generated headlines for the job advertisement dataset. The proposed method significantly increases the word overlap between generated sentences.

mation sharing method is beneficial only for the generation task. This can be attributed to the asymmetry in the task settings. Generally, the generation task need to focus on a wider range of input words than the classification tasks. For the generation task, it is important that which words are focused on in the classification task. However, this is not always the case that the words focused on in the generation task are important for the classification task.

**Analysis of word overlap between output sentences.** To estimate the consistency between output sentences, we evaluate the word overlap between generated headlines and key phrases. Table 7 presents the ROUGE recall scores of the generated key phrases computed against generated headlines for the job advertisement dataset. Therefore, our proposed method generates significantly similar word outputs, resulting in improvement in the consistency between two sentences.

### 4.3 Example of Predicted Outputs

We visualize the attention weights of the decoders to assess the performance of the proposed method. The upper part of Figure 5 indicates an example of the attention weights of the headline and key phrase generators. The proposed method increases the consistency and decreases the inconsistency between the attention weights of the two decoders. With the proposed method, both decoders commonly focus on the word "Ruby on Rails," and thus both outputs consistently contain the word "Ruby." From this result, the proposed method, which avoids inconsistency, improves the quality of the outputs.

## 5 Conclusion

We introduced a novel multi-task learning method to maintain consistency among outputs. Hierarchical consistency loss was introduced to penalize inconsistency between two attention weights of decoders. We implemented a manual evaluation using crowd-sourcing, and the results indicates that our method generates more consistent outputs. An automatic evaluation showed that proposed method achieves the best ROUGE scores on both datasets. As a future work, we would like to explore whether our method is applicable to other tasks such as multi-task learning for object detection and image caption generation.

## References

Antonios Anastasopoulos and David Chiang. 2018. Tied multitask learning for neural speech translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 82–91.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 484–494.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN Encoder–Decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Han Guo, Ramakanth Pasunuru, and Mohit Bansal. 2018. Soft layer-specific multi-task summarization with entailment and question generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, (Volume 1: Long Papers)*, pages 687–697.

Kazuma Hashimoto, Yoshimasa Tsuruoka, Richard Socher, et al. 2017. A joint many-task model: Growing a neural network for multiple NLP tasks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1923–1933.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28*, pages 1693–1701.

Wan-Ting Hsu, Chieh-Kai Lin, Ming-Ying Lee, Kerui Min, Jing Tang, and Min Sun. 2018. A unified model for extractive and abstractive summarization using inconsistency loss. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 132–141.

Masaru Isonuma, Toru Fujino, Junichiro Mori, Yutaka Matsuo, and Ichiro Sakata. 2017. Extractive summarization using multi-task learning with document classification. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2101–2110.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*.

Eliyahu Kiperwasser and Miguel Ballesteros. 2018. Scheduled multi-task learning: From syntax to translation. *Transactions of the Association of Computational Linguistics*, 6:225–240.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Proceedings of Workshop on Text Summarization Branches Out*.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In *Proceedings of 4th the International Conference on Learning Representation*.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. 2017. Deep keyphrase generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 582–592.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290.

Eric W Noreen. 1989. *Computer-intensive methods for testing hypotheses*. A Wiley-Interscience Publication, New York: Wiley, 1989.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389.

Toshinori Sato, Taiichi Hashimoto, and Manabu Okumura. 2017. Implementation of a word segmentation dictionary called mecab-ipadic-NEologd and study on how to use it effectively for information retrieval. In *Proceedings of the 23th Annual Meeting of the Association for Natural Language Processing*.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the*

*55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083.

Shi-Qi Shen, Yan-Kai Lin, Cun-Chao Tu, Yu Zhao, Zhi-Yuan Liu, Mao-Song Sun, et al. 2017. Recent advances on neural headline generation. *Journal of Computer Science and Technology*, 32(4):768–784.

Tian Shi, Yaser Keneshloo, Naren Ramakrishnan, and Chandan K Reddy. 2018. Neural abstractive text summarization with sequence-to-sequence models. *arXiv preprint arXiv:1812.02303*.

Sho Takase, Jun Suzuki, Naoaki Okazaki, Tsutomu Hirao, and Masaaki Nagata. 2016. Neural headline generation on abstract meaning representation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1054–1059.

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. From neural sentence summarization to headline generation: a coarse-to-fine approach. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 4109–4115.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 31*, pages 649–657.

# Toward a Task of Feedback Comment Generation for Writing Learning

**Ryo Nagata**
Konan University / 8-9-1 Okamoto, Kobe, 658-8501 Hyogo, Japan
Japan Science and Technology Agency, PRESTO / 4-1-8 Honcho, Kawaguchi,
Saitama 332-0012, Japan
nagata-emnlp2019 @ ml.hyogo-u.ac.jp.

## Abstract

In this paper, we introduce a novel task called feedback comment generation — a task of automatically generating feedback comments such as a hint or an explanatory note for writing learning for non-native learners of English. There has been almost no work on this task nor corpus annotated with feedback comments. We have taken the first step by creating learner corpora consisting of approximately 1,900 essays where all preposition errors are manually annotated with feedback comments. We have tested three baseline methods on the dataset, showing that a simple neural retrieval-based method sets a baseline performance with an $F$-measure of 0.34 to 0.41. Finally, we have looked into the results to explore what modifications we need to make to achieve better performance. We also have explored problems unaddressed in this work.

## 1 Introduction

In this paper, we introduce a novel task called *Feedback Comment Generation*. Feedback comment generation is[1] the task of generating hints or explanatory notes (hereafter, feedback comments) for language learners[2]. Figure 1 shows an example of feedback comments on preposition use. As in this example, feedback comments are typically given to erroneous words in a given text so that the writer can understand why their writing is not good together with underlying grammatical rules and more importantly so that they can improve their writing skill.

Feedback comment generation complements grammatical error detection/correction in language learning. With the advent of Deep Neural Networks (DNN), their performance has dramatically improved in recent years. However, they are not capable of explaining why detected words are erroneous nor why they should be corrected as indicated. This limitation is particularly problematic for beginner to intermediate learners. They might not be able to understand and acquire the underlying rules; if so, they will likely make similar errors in the future. This is exactly where feedback comment generation comes in. Namely, it complements grammatical error detection/correction by generating feedback comments to help learners understand and acquire the underlying rules. This form of feedback should be useful for language learners as Bitchener et al. (2005)[3] and Sheen (2007) show.

Despite its usefulness, there has been almost no work on feedback comment generation as Sect. 2 will describe. One of the major reasons is that there exists no publicly available dataset for research on feedback comment generation. It is costly and time-consuming to annotate writings of non-native learners with feedback comments. It is not straightforward to decide what information one should give to learners of English as feedback comments and how; as far as we are aware of, the current work on feedback for writing learning (e.g., Bitchener et al. (2005); Ferris and Roberts (2001); Robb et al. (1986); Sheen (2007)) focuses on the comparison of the salience of feedback (only detection results, detection and correction results, or those with error types and so on). Besides, even if a dataset existed, it would still be a difficult task to generate human-like feedback comments.

To attack the above difficulties, we take the first step towards feedback comment generation targeting preposition use. The reason for the choice of preposition use is that (a) preposition er-

[1] The strict definition will be introduced in Subsect. 3.

[2] In this paper, *language learners* refer to learners of English as Foreign Language (EFL).

[3] Note that in their work, human teachers did error correction in written form and provided the learners with feedback comments orally.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3206–3215,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

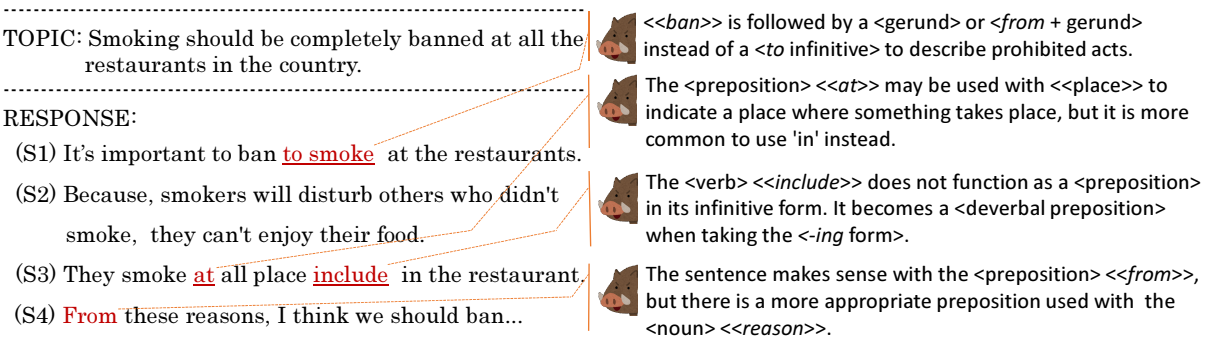


Figure 1: Example of Feedback Comments on Preposition Use.

rors are one of the most frequent error types in learner English (Tetreault and Chodorow, 2008); (b) they are relatively explainable (c.f., article errors); and (c) it would be a good starting point for feedback comment generation research considering the potential difficulty of the task. Specifically, we have annotated learner corpora with feedback comments, written in English and Japanese, on preposition use; we have released a part of the dataset to the public. Using the created dataset, we have trained and tested three baseline methods to estimate the feasibility of the task. In addition, we have looked into the results to reveal what problems we have left to achieve more complete feedback comment generation.

## 2 Related Work

The work most related to feedback comment generation is probably grammatical error detection/correction. The former is a task of detecting where errors exist in a given sentence whereas the latter is of retrieving its corrected form. In the beginning, researchers focused on specific error types – the most frequent ones such as errors in article (Han et al., 2006), preposition (Felice and Pulman, 2008), and number (Nagata et al., 2006). The state-of-the-art methods typically solve the problems as sequence labeling (e.g., Rei and Yannakoudakis (2016); Rei (2017); Kaneko et al. (2017)) or Machine Translation (MT) with DNNs (e.g., Junczys-Dowmunt et al. (2018); Napoles and Callison-Burch (2017)). As a result, performance has dramatically improved.

Unlike grammatical error correction, there has been almost no work on feedback comment generation. To the best of our knowledge, no work has even formally defined the task of feedback comment generation as we do in this paper. Some researchers (Kakegawa et al., 2000; McCoy et al., 1996; Nagata et al., 2014) made an attempt to develop rule-based methods for diagnosing errors in line with grammatical error detection/correction. Rule-based methods typically parse the input sentence and then apply rules to the resulting parse to diagnose errors. However, they encounter the tremendous difficulty of covering a wide variety of errors and of maintaining a large set of rules.

Comment generation for program source code is another task of comment generation. Its input and output are a piece of program source code such as a Java method and its comment or summary, respectively. Iyer et al. (2016) and Hu et al. (2018) solved the task as an MT problem (source code to comment) by using sequence-to-sequence models. This implies that feedback comment generation may be solved in the same manner (i.e., learner sentence to feedback comment).

## 3 Task Definition

Feedback comment generation in general is the task of generating feedback comments given a text (referred to as *essay*, hereafter). Thus, the input is an essay. The output is a set consisting of pairs of an offset and a feedback comment. An offset is a range of integers indicating where the paired feedback comment applies in the input essay. A feedback comment is a string that helps the writer (learner) improve their writing skill. It is typically a comment about a grammatical error, but can be about other things including discourse, organization, and content. It may be a side note to make the present writing better or praise to motivate the writer.

Note that because of the above, the range in feedback comment generation does not necessary match that in grammatical error detec-

tion/correction. Also note that there are cases where the input has no or more than one feedback comments; on the one hand, there exist sentences that receive no feedback comment at all as in (S2) in Fig. 1; on the other hands there exit sentences that receive more than one feedback comments as in (S3).

Our present target, feedback comment generation for preposition use, follows this task definition. Here, it is worthwhile to mention that we include *to*-infinitives in preposition use as in (S1) in Fig. 1. We also include errors that can be classified into multiple error types (preposition and other types). For example, in the sentence **Have a part-time job is easy.*, the underlined word *Have* can be interpreted as a missing *to*-infinitive (i.e., *To have*) or as a verb form error (i.e., *Having*). We include such cases in our targets to provide richer information.

## 4 Data Development

We created a dataset for feedback comment generation for preposition use. We used it in the evaluation as described in Sect. 6. We have released a part of it to the public on the web[4].

We selected the written essays in the International Corpus Network of Asian Learners of English (hereafter, ICNALE) as our base corpus. Their essay topics are controlled; all essays are written on two common topics: (a) *It is important for college students to have a part-time job.* and (b) *Smoking should be completely banned at all the restaurants in the country.*, which hereafter will be referred to as *PART-TIME JOB* and *SMOKING*, respectively. This simulates the condition common to the language learning settings that essays are often written on one topic in language learning as in writing exercises in class and writing tasks in language proficiency tests. The writers are college students (including graduate students) from 10 countries and regions in Asian (although we only used the Chinese, Indonesian, Japanese, and Korean portions for time and cost reasons). Their proficiency levels are estimated to be from A2 to B2+ in the CEFR metric. As preprocessing, we split the essays into sentences and in turn tokenized them by using the Stanford Statistical Natural Language Parser (ver.2.0.3) (de Marneffe et al., 2006).

[4]The corpus data are available at `https://www.gsk.or.jp/en/catalog//`.

Before annotation, we had to choose in which language we would create feedback comments. Basically, the choice would be either English or the writer's native language. We chose Japanese, one of the writers' native languages, for the following reasons: (1) Beginner to intermediate learners may have difficulty in understanding feedback comments in English when working on writing exercises; (2) it will likely be more technically challenging and interesting to generate feedback comments on English in a different language; (3) it would be too costly to create feedback comments in all the native languages and accordingly we chose the one we can understand. For accessibility, we manually translated them into English for future research. In this work, we only used the Japanese feedback comments.

In addition, we created two special symbols to augment feedback comments: grammatical term ($<$, $>$) and citation ($<<$, $>>$). Grammatical terms are tagged with $<$ and $>$ as in $<$*intransitive verb*$>$. With this, one can make links to corresponding grammatical items in a grammar book, for example, as an additional source of information for the user[5]. Citation is used to denote that the word(s) inside the symbols is cited from the commented sentence as in $<<$*agree*$>>$. This makes feedback comments more flexible. That is, the word(s) inside $<<$ and $>>$ in a generated feedback comment may be replaced with another word(s) in the commented sentence.

This is the big picture of how we created the data. More details are described in the guideline accompanied with the dataset[6].

We hired two professional annotators who had a good command of English. Both of them have had experience in English syntactic annotation for more than ten years; one of them had two years of professional English writing teaching experience where she provided high school and college students with feedback comments as in this work.

We sampled out 1,040 essays from ICNALE and assigned them to either annotator. The annotators read an entire essay first one at a time and then added feedback comments to all preposition errors they found, plus other parts on which they wanted to annotate (e.g., praise). They used the commenting function in MS-Word for this annotation.

[5]We have also created a grammar database with a list of grammar items and their explanations. However, it is not included in the dataset due to copyright issues.

[6]The guideline is included in the dataset.

| Corpus | PART-TIME JOB | | | | SMOKING | | | |
|---|---|---|---|---|---|---|---|---|
| Split | Total | Training | Dev. | Test | Total | Training | Dev. | Test |
| # essays | 521 | 371 | 75 | 75 | 519 | 369 | 75 | 75 |
| # sentences | 7,989 | 5,615 | 1,215 | 1,159 | 7,979 | 5,665 | 1,206 | 1,180 |
| # tokens | 131,947 | 93,896 | 19,261 | 18,781 | 130,154 | 92,419 | 19,017 | 18,709 |
| # comments | 1,408 | 1,008 | 210 | 190 | 1,426 | 1,034 | 208 | 184 |
| comments/sent. | 0.18 | 0.18 | 0.17 | 0.16 | 0.18 | 0.17 | 0.17 | 0.17 |

Table 1: Statistics on the created dataset.

Finally, we split 1,040 essays into training, development, and test sets as shown in Table 1[7]. We transformed the MS-Word format into a TSV format (learner sentence, offset, feedback comment[8]) and used the resulting essays for evaluation.

## 5 Baseline Methods

To examine the feasibility of feedback comment generation for preposition use, we implemented the following three simple baseline methods: a neural retrieval-based method, a sequence-to-sequence model, and a rule-based method. The following subsections describe each method in detail.

### 5.1 Neural Retrieval-based Method

This method solves the task as a feedback comment retrieval problem. The procedure consists of:
**(0)** Preprocessing
**(I)** Feedback comment vector encoding
**(II)** Detection and context vector extraction
**(III)** Mapping between the two vectors
**(IV)** Feedback comment retrieval
**(V)** Output
The main steps (I)–(IV) are depicted in Fig. 2.

**(0) Preprocessing** is applied to the input essay in both training and prediction. First, it is split into sentences. Then, each sentence is tokenized and put into lowercase. Finally, tokens whose occurrence is lower than a certain threshold are replaced with the special symbol <unk>. The same procedures except sentence splitting are applied to feedback comments in the training data.

In **(I) Feedback comment encoding**, the preprocessed feedback comments are encoded as vectors (Fig. 2 (I)) by using LSTM-based Language Models (LSTMLMs). Tokens in a feedback comment are turned into their corresponding word embeddings in an embedding layer and then passed on to an LSTM, which is trained to predict the next token. After training, all feedback comments are once again passed through the trained network. Its final hidden state of the final layer, which is merely a vector, can be regarded as an abstract representation of a feedback comment. In addition, another LSTMLM of the same architecture, which takes as input each token in reverse order, is used to augment the representation power. The two final states are simply concatenated to produce a vector for the input feedback comment. In the end, all feedback comments are abstractly represented as vectors, which will be used later in (III).

**(II) Detection and context vector extraction** consist of two subtasks. The first is to detect where to comment in the input essay. Hereafter, each sentence in the input essay is processed (namely, given to the network) one at a time. Under this condition, the subtask can be regarded as a binary sequence labeling problem in which input and output are a sequence of tokens and a sequence of 0/1 (to comment or not), respectively as in the sequence labelling-based error detection (e.g., Rei and Yannakoudakis (2016); Rei (2017); Kaneko et al. (2017)).

Following the work (Kaneko et al., 2017), this paper uses a BiLSTM-based sequence labeling model as shown in Fig. 2 (II). To be precise, each token is transformed into its corresponding word embedding, then is given to a BiLSTM, and finally to a softmax layer that estimates the probabilities of 0 and 1. During training, the information about where to comment is given as offsets. For simplicity, we used word-based offsets. In addition, if a feedback comment refers to more than one words, we used the position of the central word as its offset. The loss function to minimize is simply the cross-entropy loss. During prediction, each token

[7]We are still conducting the annotation over ICNALE and another learner corpus (Konan-JIEM learner corpus (Nagata and Sakaguchi, 2016)). At the time of submission, the number of annotated essays has reached 1,900.

[8]The offset and feedback comment columns may be none or repeated an arbitrary number of times.

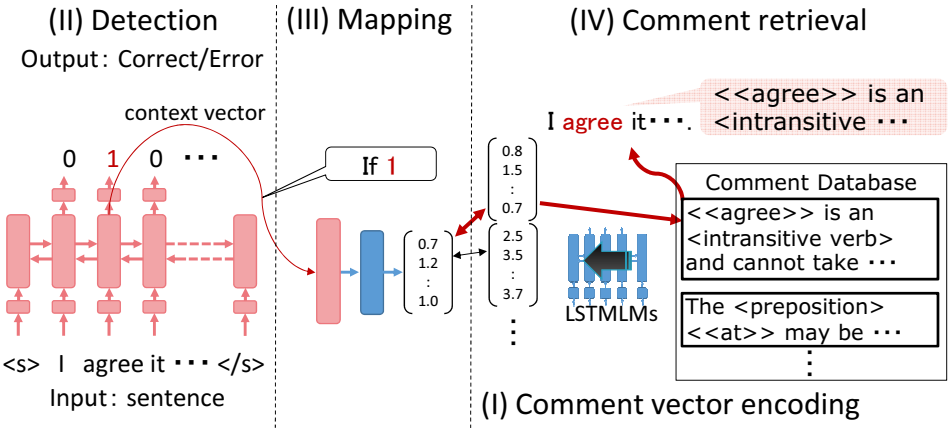


Figure 2: Neural Retrieval-based Method.

in the target sentence is given to the network to estimate the probabilities of 0 and 1.

The second subtask is to represent the surrounding context of the detected word abstractly. This is actually already done in the course of the first subtask. Namely, the corresponding BiLSTM hidden states can be used as context vectors because at each time step, the information about its surrounding words are recurrently accumulated in them.

In **(III) Mapping between the two vectors**, the detected context vectors are mapped onto the feedback comment vector space[9]. This is done by a Feed Forward Neural Network (FFNN) (Fig. 2 (III)). The purpose is to map context vectors near the corresponding comment vectors in the space. Accordingly, the mean square error can be naturally used as a loss function. Note that the whole networks (BiLSTM for sequence labeling in (II) and FFNN for vector mapping in (III)) are trained simultaneously (i.e., multi-task learning). The total loss $l$ is defined by $l = (1-\beta)l_{\mathrm{s}} + \beta l_{\mathrm{m}}$ where $l_{\mathrm{s}}$, $l_{\mathrm{m}}$, and $\beta$ are the losses for sequence labeling and vector mapping, and a hyperparameter that controls the weighting between the two, respectively.

**(IV) Feedback comment retrieval** and the following procedures are applied only during prediction. After (II) and (III), words to be commented on are detected and their context vectors are transformed into feedback comment vectors. This means that one can retrieve the most appropriate feedback comments by searching the space for feedback comment vectors similar to the mapped context vector. The similarity is measured by the cosine similarity between two vectors.

To improve performance, it is important to filter out inappropriate results. In some cases, a discrepancy occurs between the word detected in (II) (say, *at*) and the word to which the retrieved comment applies in the training data (say, *in*). In that case, the retrieved feedback comment will likely be inappropriate no matter how high its similarity is (in the above example, a feedback comment concerning *in* is applied to *at*). Consequently, these feedback comments are excluded from the results. In addition, the cosine similarity should be a good indicator to determine whether the retrieved feedback comments are appropriate or not, and thus retrieved feedback comments are discarded if their similarity is lower than a certain threshold. In other words, we do not generate feedback comments if we cannot find similar feedback comments in the training data.

In **(V) Output**, the most appropriate feedback comment (according to their similarities) for each commented word is displayed to the user.

### 5.2 Sequence-to-sequence Model-based Method

This method is an implementation of the sequence-to-sequence (encoder-decoder) model with an attention mechanism (Bahdanau et al., 2014)). It takes a learner sentence as input. It then puts the input sentence into a feedback comment.

Each word in the input learner sentence, which is preprocessed just as in the neural retrieval-based method, is encoded by a word embedding layer and a BiLSTM-based encoder. Then, a feedback

[9]Recall all comments in the training data have already been transformed into vectors in (I).

comment is generated by another word embedding layer and an LSTM-based decoder with an attention mechanism.

During the development phase, we observed that the network trained on the entire training data almost always resulted in generating no feedback comments. We observed a similar tendency when it was trained with a fixed feedback comment for the correct sentences (such as *There is no preposition error in this sentence.*). Considering this, we trained (and tested) the sequence-to-sequence model-based method on only sentences that had feedback comments to show the upper bound of this method.

### 5.3 Case frame-based Method

This method is an implementation of the case frame-based method (Nagata et al., 2014). The method automatically induces case frames by comparing learner and native corpora, to detect preposition errors with feedback comments.

We apply the exact same preprocessing as in the neural retrieval-based method to both learner and native corpora to induce case frames. We then manually created feedback comments for each induced case frame.

## 6 Evaluation

### 6.1 Conditions and Procedures

We implemented and trained the baseline methods with the created dataset. We first obtain word embeddings for learner sentences from the corpora as shown in Appendix A. We also used the word embeddings for English words in the LSTMLMs to encode feedback comments. For the rest (i.e., Japanese words), we initialized them using random-valued vectors. We used MeCab[10] to tokenize the feedback comments. With these word embeddings, we trained the networks on the training set of the respective subsets (PART-TIME JOB and SMOKING). We implemented the case frame-based method with the following corpora: British National Corpus (BNC) (Burnard, 1995), the EDR corpus (Japan electronic dictionary research institute Ltd, 1993) as a native corpus and the training and development set of the corresponding dataset[11] as a learner corpus. As a result, we obtained two versions of each method. We determined the hyperparameters by using the corresponding development set[12]. We tested the resulting models on the corresponding test set.

For evaluation, one of the professional annotators, who had been involved in the data creation, manually evaluated the results. She labeled each generated feedback comment as *appropriate*, *partially appropriate*, or *inappropriate*; *partially appropriate* refers to the case that the feedback comment is not completely appropriate but would become appropriate if part of it were edited.

We measured performance by recall, precision. We also used $F$-measure targeting *appropriate* feedbacks. We calculated their values considering only *appropriate* as correct generation. We also used their relaxed versions regarding *partially appropriate* as appropriate generation.

### 6.2 Results

Table 2 shows the results. It turns out that the simple application of the sequence-to-sequence model does not work well at all on this task; note that it is provided with the information about which sentence should have a feedback comment (i.e., tested on only the sentences having feedback comments). Nevertheless, its performance is very poor. This suggests that it requires modifications to achieve better performance with the sequence-to-sequence framework. *Case frame-based* successfully generates feedback comments in some cases. However, its recall is quite low. In contrast, the neural retrieval-based method achieves a far better performance in recall, achieving a precision comparable to that of *case frame-based*.

At the same time, Table 2 shows that there is still room for improvement. Subsect. 7.1 will investigate the generation results to reveal what has been solved by the methods.

[10] http://taku910.github.io/mecab/

[11] Because it does not require a development set for hyperparameter tuning, the development set was added to the training set.

[12] The hyperparameters are as follows: LSTMLMs: number of LSTM layers: 2, number of hidden states: 600; BiLSTM: number of LSTM layers: 1, number of hidden states: $600 \times 2$; FFNN: number of layers:1 (with dimension 1,200); BiLSTM encoder: number of LSTM layers:1, number of hidden states $600 \times 2$; LSTM decoder: number of LSTM layers:1, number of hidden states 600. Common hyperparameters: batch size: 64, number of epochs: 200 (with early stopping (patience: 10)), dropout rate: 0.5, optimization algorithm: Adam (step size: 0.01, the first and second moment: 0.9 and 0.999, respectively), threshold for <unk>: 5 in essays and 2 in feedback comments, threshold for the cosine similarity: 0.971 (PART-TIME JOB) and 0.976 (SMOKING), weight $\beta$ for losses: 0.9.

| Method | PART-TIME JOB | | | SMOKING | | |
|---|---|---|---|---|---|---|
| | Recall | Precision | $F$-measure | Recall | Precision | $F$-measure |
| Retrieval-based | 0.23 (0.25) | 0.61 (0.67) | 0.34 (0.37) | 0.28 (0.30) | 0.72 (0.78) | 0.41 (0.44) |
| Seq2seq | 0.06 (0.07) | 0.07 (0.08) | 0.07 (0.08) | 0.10 (0.13) | 0.11 (0.13) | 0.10 (0.13) |
| Case frame-based | 0.10 (0.10) | 0.62 (0.62) | 0.16 (0.16) | 0.05 (0.05) | 0.75 (0.75) | 0.09 (0.09) |

Table 2: Generation Performance (numbers in brackets correspond to the relaxed measures).

## 7 Discussion

### 7.1 Analysis of Generated Comments

Looking into the generation results reveals that the neural retrieval-based method successfully generates feedback comments on errors typical to the topics (e.g., (S3) and (S4) in Fig. 3) and typical to argumentative essays (e.g., (S1) and (S7)), or more general ones (e.g., (S6)). While the superficial variations of these typical errors are large in general, they correspond to one or a few feedback comments. Actual examples are: *I/People (don't) agree the/this statement/opinion/that/thinking* and *(more) harmful (not only) for anyone/people/smokers/non-smokers*. These findings suggest that abstractly representing feedback comments and contexts and tying them in the feedback comment space, is effective in generating feedback comment on typical preposition errors.

It would probably be possible to create rule-based methods covering such typical errors. It would, however, take the expertise of at least a teacher of English and also a computer engineer to achieve it; the former would have to think of what is typical in the given topic and then to make a set of rules; the latter then would have to turn them into computer-readable forms. In contrast, the neural retrieval-based method only requires a teacher of English to annotate a given corpus with feedback comments, without examining what is typical, which is much more effective and efficient.

More importantly, there exist typical errors to which rules-based methods do not apply well. Examples include the word *Have* in (S2) and the word *include* in (S6) in Fig. 3. On the one hand, it would require a successful parsing to recognize the sources of the errors. On the other hand, it would require successful error detection to parse them correctly. Accordingly, rule-based methods, which normally require parsing, fail in generation in these cases. In contrast, the neural retrieval-based method gives a simple solution to this *parsing-first* or *error detection-first* dilemma because it does not depend on parsing at all.

To see the upper bound of the performance of the neural retrieval-based method, we evaluated its performance using the oracle offsets (where to comment). As a result, the strict recall and precision respectively improved to 0.27 and 0.73 in PART-TIME JOB and 0.30 and 0.79 in SMOKING. We also evaluated its detection performance as another upper bound of performance; if we cannot detect errors, we will never be able to generate feedback comments for them. We obtained a recall of 0.27 and a precision of 0.71 in PART-TIME JOB and a recall of 0.31 and a precision of 0.79 in SMOKING. The comparison between the two upper bounds suggests that one can successfully generate feedback comments for detected errors. This is partly because we used the same training data for error detection and feedback comment generation.

All these findings show neural retrieval-based methods are promising to generate feedback comments for typical errors. It is important from the viewpoint of language learning assistance to be able to generate feedback comments for typical errors considering that feedback is normally given to typical errors first in language learning (rather than to rare, irregular errors). More sophisticated retrieval-based methods such as the work by Hashimoto et al. (2018) and Qiu et al. (2017) will likely improve generation performance.

### 7.2 Model Analysis

One would probably think of using an attention mechanism to improve the neural retrieval-based method. We actually investigated its effectiveness during the development phase. It turned out that the neural retrieval-based method with and without an attention mechanism performed similarly; to be precise, the $F$-measure (of comment word detection) of the one with an attention mechanism was often slightly worse.

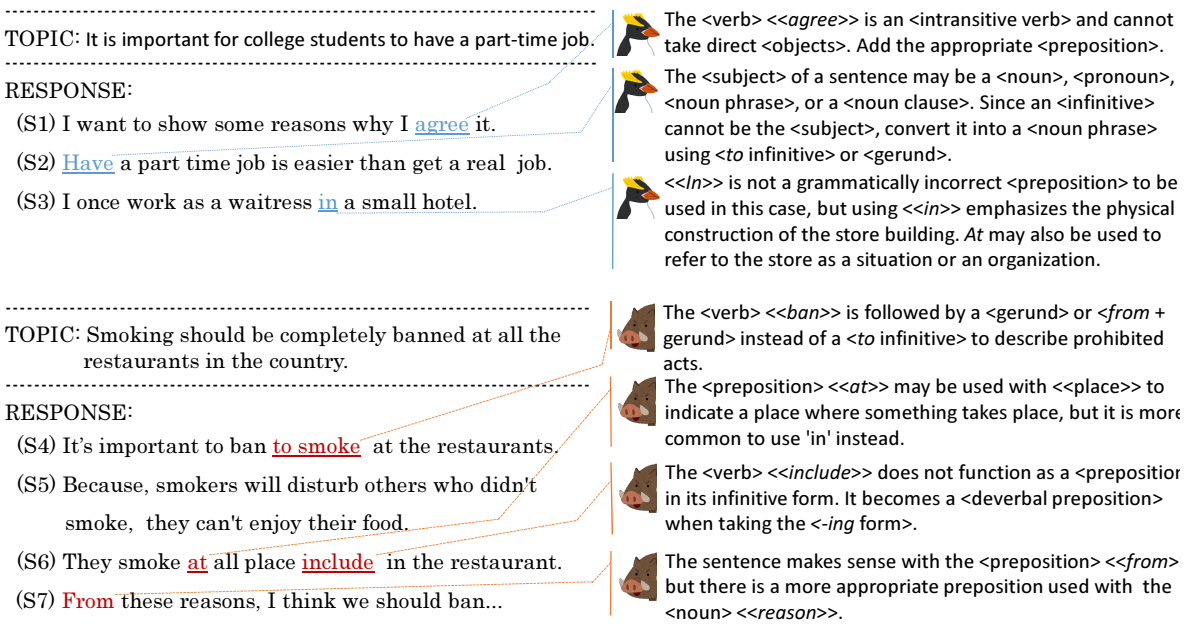


Figure 3: Example of Feedback Comments on Preposition Use Generated by Neural Retrieval-based Method.

This is partly because one can often tell from a narrow context why the usage of a given preposition is erroneous as in *agree it*. BiLSTM can handle well the information in such a narrow window without an attention mechanism. Having said that, we need further investigations to confirm that this argument is true; in particular, it might be necessary to generate feedback comments in general. Another reason would be that the amount of training data was not enough to train the attention layer, which also might be a reason why the sequence-to-sequence model (with an attention mechanism) does not perform well.

### 7.3 Unaddressed Problems

Sect. 6 has shown that feedback comments for preposition use can be automatically generated to some extent. The question now is whether the same argument can be made about feedback comment generation in general. Intuitively, it is a more difficult task because there are more variations in other grammatical errors. There are much more variations in feedback comments concerning other writing skills such as discourse and organization.

To answer the question, we have annotated and (are still annotating) learner corpora with feedback comments in general (together with those for preposition use). At the time of submission, the number of annotated essays has reached 2,300 (and will be 4,000 in the end) over two learner corpora (ICNALE and Konan-JIEM learner corpus (Nagata and Sakaguchi, 2016)). They range over a wide variety of writing skills including discourse, word choice, and organization. We will test the neural retrieval-based method on the dataset.

Another factor that makes the task hard is how to evaluate generated feedback comments. While, in this paper, we have applied manual evaluation, it is highly costly and time-consuming. For this reason, it would be difficult to make a thorough comparison between various methods or various model settings. One should have an efficient way of evaluating generated feedback comments to facilitate feedback comment generation research.

BLEU (Papineni et al., 2002) would be a choice for solving this problem as can be seen in machine translation research. As a pilot study, we estimated the validity of BLEU in this task. We randomly sampled out 702 feedback comments from the dataset and manually extracted pairs that had the same content. As a result, it turned out that the BLEU value ranged from as low as 0.15 to as high as 0.99; the average and the standard deviation were 0.61 and 0.25, respectively[13]. For comparison, we also calculated the BLEU value for pairs randomly chosen from the rest (that had not been paired previously); the average and the stan-

[13] We excluded the exact matches.

dard deviation were 0.13 and 0.09, respectively[14]. These numbers show that while BLEU can distinguish between appropriate and inappropriate feedback comments to some extent, it can be unreliable in some cases. All these observations suggest that we should have other evaluation methods together with BLEU.

## 8 Conclusions

In this paper, we introduced a novel task called feedback comment generation. We first described the task definition and created a dataset for the task of feedback comment generation for preposition use. We then tested a rule-based and two neural-based baselines on the dataset, showing that retrieval-based methods were promising. We further analyzed what the neural retrieval-based method can generate. We also discussed unaddressed problems. Especially, we pointed out that the evaluation was costly and time-consuming and that we needed a more efficient way of evaluation.

In future work, we will investigate how we can generate more flexible feedback comments. One way of achieving it is to apply a more sophisticated retrieval-based method such as (Hashimoto et al., 2018) to this task. We will also investigate how we can efficiently evaluate generation results.

## Acknowledgments

We would like to thank the three anonymous reviewers for their useful comments on this paper. This work was supported by Japan Science and Technology Agency (JST), PRESTO Grant Number JPMJPR1758, Japan

## A Corpora Used to Obtain Word Embeddings

Learner corpora: Corpus of English Essays Written by Japanese University Students (CEEEJUS)[15], ETS Corpus of non-native written English (Daniel Blanchard et al., 2014), The International Corpus of Learner English (ICLE) (Granger, 1993), Cambridge Learner Corpus (CLC) First Certificate in English (FCE) dataset (Yannakoudakis et al., 2011), and Nagoya Interlanguage Corpus of English (NICE) (Sugiura et al., 2007). Native corpus: English Web Treebank (EWT) (Bies, Ann, et al., 2012).

[14] We conducted the random sampling five times and took the average.

[15] http://language.sakura.ne.jp/s/doc/projects/CEEAUS.pdf

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv e-prints*.

Bies, Ann, et al. 2012. English web treebank LDC2012T13. Web Download.

John Bitchener, Stuart Young, and Denise Cameron. 2005. The effect of different types of corrective feedback on ESL student writing. *Journal of Second Language Writing*, 14(3):191–205.

Lou Burnard. 1995. *Users Reference Guide for the British National Corpus. version 1.0*. Oxford University Computing Services, Oxford.

Daniel Blanchard et al. 2014. ETS corpus of non-native written English, ldc2014t06. web download.

Rachele De Felice and Stephen G. Pulman. 2008. A classifier-based approach to preposition and determiner error correction in L2 English. In *Proc. of 22nd International Conference on Computational Linguistics*, pages 169–176.

Dana Ferris and Barrie Roberts. 2001. Error feedback in L2 writing classes: How explicit does it need to be? *Journal of Second Language Writing*, 10(3):161–184.

Sylviane Granger. 1993. The international corpus of learner English. In *English language corpora: Design, analysis and exploitation*, pages 57–69. Rodopi.

Na-Rae Han, Martin Chodorow, and Claudia Leacock. 2006. Detecting errors in English article usage by non-native speakers. *Natural Language Engineering*, 12(2):115–129.

Tatsunori B Hashimoto, Kelvin Guu, Yonatan Oren, and Percy S Liang. 2018. A retrieve-and-edit framework for predicting structured outputs. In *Advances in Neural Information Processing Systems 31*, pages 10052–10062.

Xing Hu, Ge Li, Xin Xia, David Lo, and Zhi Jin. 2018. Deep code comment generation. In *Proc. of 26th Conference on Program Comprehension*, pages 200–210.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. 2016. Summarizing source code using a neural attention model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 2073–2083.

Japan electronic dictionary research institute Ltd. 1993. *EDR electronic dictionary specifications guide*. Japan electronic dictionary research institute ltd.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Shubha Guha, and Kenneth Heafield. 2018. Approaching neural grammatical error correction as a low-resource machine translation task. In *Proc. of 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 595–606.

Jun'ichi Kakegawa, Hisayuki Kanda, Eitaro Fujioka, Makoto Itami, and Kohji Itoh. 2000. Diagnostic processing of Japanese for computer-assisted second language learning. In *Proc. of 38th Annual Meeting of the Association for Computational Linguistics*, pages 537–546.

Masahiro Kaneko, Yuya Sakaizawa, and Mamoru Komachi. 2017. Grammatical error detection using error- and grammaticality-specific word embeddings. In *Proc. of 8th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 40–48.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proc. of 5th International Conference on Language Resources and Evaluation*, pages 449–445.

Kathleen F. McCoy, Christopher A. Pennington, and Linda Z. Suri. 1996. English error correction: A syntactic user model based on principled "mal-rule" scoring. In *Proc. of 5th International Conference on User Modeling*, pages 69–66.

Ryo Nagata, Atsuo Kawai, Koichiro Morihiro, and Naoki Isu. 2006. A feedback-augmented method for detecting errors in the writing of learners of English. In *Proc. of 44th Annual Meeting of the Association for Computational Linguistics*, pages 241–248.

Ryo Nagata and Keisuke Sakaguchi. 2016. Phrase structure annotation and parsing for learner English. In *Proc. of 54th Annual Meeting of the Association for Computational Linguistics*, pages 1837–1847.

Ryo Nagata, Mikko Vilenius, and Edward Whittaker. 2014. Correcting preposition errors in learner English using error case frames and feedback messages. In *Proc. of 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 754–764.

Courtney Napoles and Chris Callison-Burch. 2017. Systematically adapting machine translation for grammatical error correction. In *Proc. of 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 345–356.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proc. of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Minghui Qiu, Feng-Lin Li, Siyu Wang, Xing Gao, Yan Chen, Weipeng Zhao, Haiqing Chen, Jun Huang, and Wei Chu. 2017. AliMe Chat: A sequence to sequence and rerank based chatbot engine. In *Proc. of 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 498–503. Association for Computational Linguistics.

Marek Rei. 2017. Semi-supervised multitask learning for sequence labeling. In *Proc. of 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2121–2130.

Marek Rei and Helen Yannakoudakis. 2016. Compositional sequence labeling models for error detection in learner writing. In *Proc. of 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1181–1191.

Thomas Robb, Steven Ross, and Ian Shortreed. 1986. Salience of feedback on error and its effect on EFL writing quality. *TESOL QUARTERY*, 20(1):83–93.

Younghee Sheen. 2007. The effect of focused written corrective feedback and language aptitude on ESL learners' acquisition of articles. *TESOL Quarterly*, 41:255–283.

Masatoshi Sugiura, Masumi Narita, Tomomi Ishida, Tatsuya Sakaue, Remi Murao, and Kyoko Muraki. 2007. A discriminant analysis of non-native speakers and native speakers of English. In *Proc. of Corpus Linguistics Conference CL2007*, pages 84–89.

Joel R. Tetreault and Martin Chodorow. 2008. The ups and downs of preposition error detection in ESL writing. In *Proc. of 22nd International Conference on Computational Linguistics*, pages 865–872.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading ESOL texts. In *Proc. of 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 180–189.

# Improving Question Generation With to the Point Context

**Jingjing Li**[1]* **Yifan Gao**[1]* **Lidong Bing**[2] **Irwin King**[1] **Michael R. Lyu**[1]
[1] Department of Computer Science and Engineering,
The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
[2] R&D Center Singapore, Machine Intelligence Technology
Alibaba DAMO Academy
[1]{lijj, yfgao, king, lyu}@cse.cuhk.edu.hk [2]l.bing@alibaba-inc.com

## Abstract

Question generation (QG) is the task of generating a question from a reference sentence and a specified answer within the sentence. A major challenge in QG is to identify answer-relevant context words to finish the declarative-to-interrogative sentence transformation. Existing sequence-to-sequence neural models achieve this goal by proximity-based answer position encoding under the intuition that neighboring words of answers are of high possibility to be answer-relevant. However, such intuition may not apply to all cases especially for sentences with complex answer-relevant relations. Consequently, the performance of these models drops sharply when the relative distance between the answer fragment and other non-stop sentence words that also appear in the ground truth question increases. To address this issue, we propose a method to jointly model the unstructured sentence and the structured answer-relevant relation (extracted from the sentence in advance) for question generation. Specifically, the structured answer-relevant relation acts as the to the point context and it thus naturally helps keep the generated question to the point, while the unstructured sentence provides the full information. Extensive experiments show that to the point context helps our question generation model achieve significant improvements on several automatic evaluation metrics. Furthermore, our model is capable of generating diverse questions for a sentence which conveys multiple relations of its answer fragment.

## 1 Introduction

Question Generation (QG) is the task of automatically creating questions from a range of inputs, such as natural language text (Heilman and Smith, 2010), knowledge base (Serban et al., 2016) and image (Mostafazadeh et al., 2016). QG is an increasingly important area in NLP with various application scenarios such as intelligence tutor systems, open-domain chatbots and question answering dataset construction. In this paper, we focus on question generation from reading comprehension materials like SQuAD (Rajpurkar et al., 2016). As shown in Figure 1, given a sentence in the reading comprehension paragraph and the text fragment (i.e., the answer) that we want to ask about, we aim to generate a question that is asked about the specified answer.

**Sentence**: The daily mean temperature in January, the area's coldest month, is 32.6 °F (0.3 °C); however, temperatures usually drop to 10 °F (-12 °C) several times per winter and reach 50 °F (10 °C) several days each winter month.
**Reference Question**: What is New York City 's daily January mean temperature in degrees celsius ?
**Baseline Prediction**: What is the coldest temperature in Celsius ?
**Structured Answer-relevant Relation**: (The daily mean temperature in January; is; 32.6 °F (0.3 °C))

Figure 1: An example SQuAD question with the baseline's prediction. The answer ("0.3 °C") is highlighted.

Question generation for reading comprehension is firstly formalized as a declarative-to-interrogative sentence transformation problem with predefined rules or templates (Mitkov and Ha, 2003; Heilman and Smith, 2010). With the rise of neural models, Du et al. (2017) propose to model this task under the sequence-to-sequence (Seq2Seq) learning framework (Sutskever et al., 2014) with attention mechanism (Luong et al., 2015). However, question generation is a one-to-many sequence generation problem, i.e., several aspects can be asked given a sentence. Zhou et al. (2017) propose the answer-aware question generation setting which assumes the answer, a contiguous span inside the input sentence, is al-

* These two authors contributed equally.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3216–3226,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| Distance | B1 | B2 | B3 | B4 | MET | R-L |
|---|---|---|---|---|---|---|
| 0∼10 (72.8% of #) | 45.25 | 30.31 | 22.06 | 16.54 | 21.54 | 46.26 |
| >10 (27.2% of #) | 35.67 | 21.72 | 14.82 | 10.46 | 16.72 | 37.63 |

Table 1: Performance for the average relative distance between the answer fragment and other non-stop sentence words that also appear in the ground truth question. (Bn: BLEU-n, MET: METEOR, R-L: ROUGE-L)

ready known before question generation. To capture answer-relevant words in the sentence, they adopt a BIO tagging scheme to incorporate the answer position embedding in Seq2Seq learning. Furthermore, Sun et al. (2018) propose that tokens close to the answer fragments are more likely to be answer-relevant. Therefore, they explicitly encode the relative distance between sentence words and the answer via position embedding and position-aware attention.

Although existing proximity-based answer-aware approaches achieve reasonable performance, we argue that such intuition may not apply to all cases especially for sentences with complex structure. For example, Figure 1 shows such an example where those approaches fail. This sentence contains a few facts and due to the parenthesis (i.e. "the area's coldest month"), some facts intertwine: "The daily mean temperature in January is 0.3°C" and "January is the area's coldest month". From the question generated by a proximity-based answer-aware baseline, we find that it wrongly uses the word "coldest" but misses the correct word "mean" because "coldest" has a shorter distance to the answer "0.3°C".

In summary, their intuition that "the neighboring words of the answer are more likely to be answer-relevant and have a higher chance to be used in the question" is not reliable. To quantitatively show this drawback of these models, we implement the approach proposed by Sun et al. (2018) and analyze its performance under different relative distances between the answer and other non-stop sentence words that also appear in the ground truth question. The results are shown in Table 1. We find that the performance drops at most 36% when the relative distance increases from "$0 \sim 10$" to "$> 10$". In other words, when the useful context is located far away from the answer, current proximity-based answer-aware approaches will become less effective, since they overly emphasize neighboring words of the answer.

To address this issue, we extract the structured answer-relevant relations from sentences and propose a method to jointly model such structured relation and the unstructured sentence for question generation. The structured answer-relevant relation is likely to be to the point context and thus can help keep the generated question to the point. For example, Figure 1 shows our framework can extract the right answer-relevant relation ("The daily mean temperature in January", "is", "32.6°F (0.3°C)") among multiple facts. With the help of such structured information, our model is less likely to be confused by sentences with a complex structure. Specifically, we firstly extract multiple relations with an off-the-shelf Open Information Extraction (OpenIE) toolbox (Saha and Mausam, 2018), then we select the relation that is most relevant to the answer with carefully designed heuristic rules.

Nevertheless, it is challenging to train a model to effectively utilize both the unstructured sentence and the structured answer-relevant relation because both of them could be noisy: the unstructured sentence may contain multiple facts which are irrelevant to the target question, while the limitation of the OpenIE tool may produce less accurate extracted relations. To explore their advantages simultaneously and avoid the drawbacks, we design a gated attention mechanism and a dual copy mechanism based on the encoder-decoder framework, where the former learns to control the information flow between the unstructured and structured inputs, while the latter learns to copy words from two sources to maintain the informativeness and faithfulness of generated questions.

In the evaluations on the SQuAD dataset, our system achieves significant and consistent improvement as compared to all baseline methods. In particular, we demonstrate that the improvement is more significant with a larger relative distance between the answer and other non-stop sentence words that also appear in the ground truth question. Furthermore, our model is capable of generating diverse questions for a single sentence-answer pair where the sentence conveys multiple relations of its answer fragment.

**Sentence**: Beyoncé received critical acclaim and commercial success, selling one million digital copies worldwide in six days; The New York Times noted the album's unconventional, unexpected release as significant.
**N-ary Relations**:
0.85 (Beyoncé; received commercial success selling; one million digital copies worldwide; in six days)
0.92 (The New York Times; noted; the album's unconventional, unexpected release as significant)

**Sentence**: The daily mean temperature in January, the area's coldest month, is 32.6 °F (0.3 °C); however, temperatures usually drop to 10 °F (-12 °C) several times per winter and reach 50 °F (10 °C) several days each winter month.
**N-ary Relations**:
0.95 (The daily mean temperature in January; is; 32.6 °F (0.3 °C))
0.94 (temperatures; drop; to 10 °F (12 °C); several times per winter; usually)
0.90 (temperatures; reach; 50 °F)

Figure 2: Examples for n-ary extractions from sentences using OpenIE. Confidence scores are shown at the beginning of each relation. Answers are highlighted in sentences. Waved relations are selected according to our criteria in Section 2.2.

## 2 Framework Description

In this section, we first introduce the task definition and our protocol to extract structured answer-relevant relations. Then we formalize the task under the encoder-decoder framework with gated attention and dual copy mechanism.

### 2.1 Problem Definition

We formalize our task as an answer-aware Question Generation (QG) problem (Zhao et al., 2018), which assumes answer phrases are given before generating questions. Moreover, answer phrases are shown as text fragments in passages. Formally, given the sentence $S$, the answer $A$, and the answer-relevant relation $M$, the task of QG aims to find the best question $\overline{Q}$ such that,

$$\overline{Q} = \arg\max_{Q} \text{Prob}(Q|S, A, M), \qquad (1)$$

where $A$ is a contiguous span inside $S$.

### 2.2 Answer-relevant Relation Extraction

We utilize an off-the-shelf toolbox of OpenIE [1] to the derive structured answer-relevant relations from sentences as to the point contexts. Relations extracted by OpenIE can be represented either in a triple format or in an n-ary format with several secondary arguments, and we employ the latter to keep the extractions as informative as possible and avoid extracting too many similar relations in different granularities from one sentence. We join all arguments in the extracted n-ary relation into a sequence as our to the point context. Figure 2 shows n-ary relations extracted from OpenIE. As we can see, OpenIE extracts multiple relations for complex sentences. Here we select the most informative relation according to three criteria in the order of descending importance: (1) having the maximal number of overlapped tokens between the answer and the relation; (2) being assigned the highest confidence score by OpenIE; (3) containing maximum non-stop words. As shown in Figure 2, our criteria can select answer-relevant relations (waved in Figure 2), which is especially useful for sentences with extraneous information. In rare cases, OpenIE cannot extract any relation, we treat the sentence itself as the to the point context.

[1] http://openie.allenai.org/

| | Sentence | Answer-relevant Relation |
|---|---|---|
| Avg. length | 32.46 | 13.04 |
| # overlapped words | 2.87 | 1.86 |
| Copy ratio | 8.85% | 14.26% |

Table 2: Comparisons between sentences and answer-relevant relations. Overlapped words are those non-stop tokens co-occurring in the source (sentence/relation) and the target question. Copy ratio means the proportion of source tokens that are used in the question.

Table 2 shows some statistics to verify the intuition that the extracted relations can serve as more to the point context. We find that the tokens in relations are 61% more likely to be used in the target question than the tokens in sentences, and thus they are more to the point. On the other hand, on average the sentences contain one more question token than the relations (1.86 v.s. 2.87). Therefore, it is still necessary to take the original sentence into account to generate a more accurate question.

### 2.3 Our Proposed Model

**Overview.** As shown in Figure 3, our framework consists offour components (1) Sentence Encoder and Relation Encoder, (2) Decoder, (3) Gated Attention Mechanism and (4) Dual Copy Mechanism. The sentence encoder and relation encoder encode the unstructured sentence and the structured answer-relevant relation, respectively. To select and combine the source information from the two encoders, a gated attention mechanism is

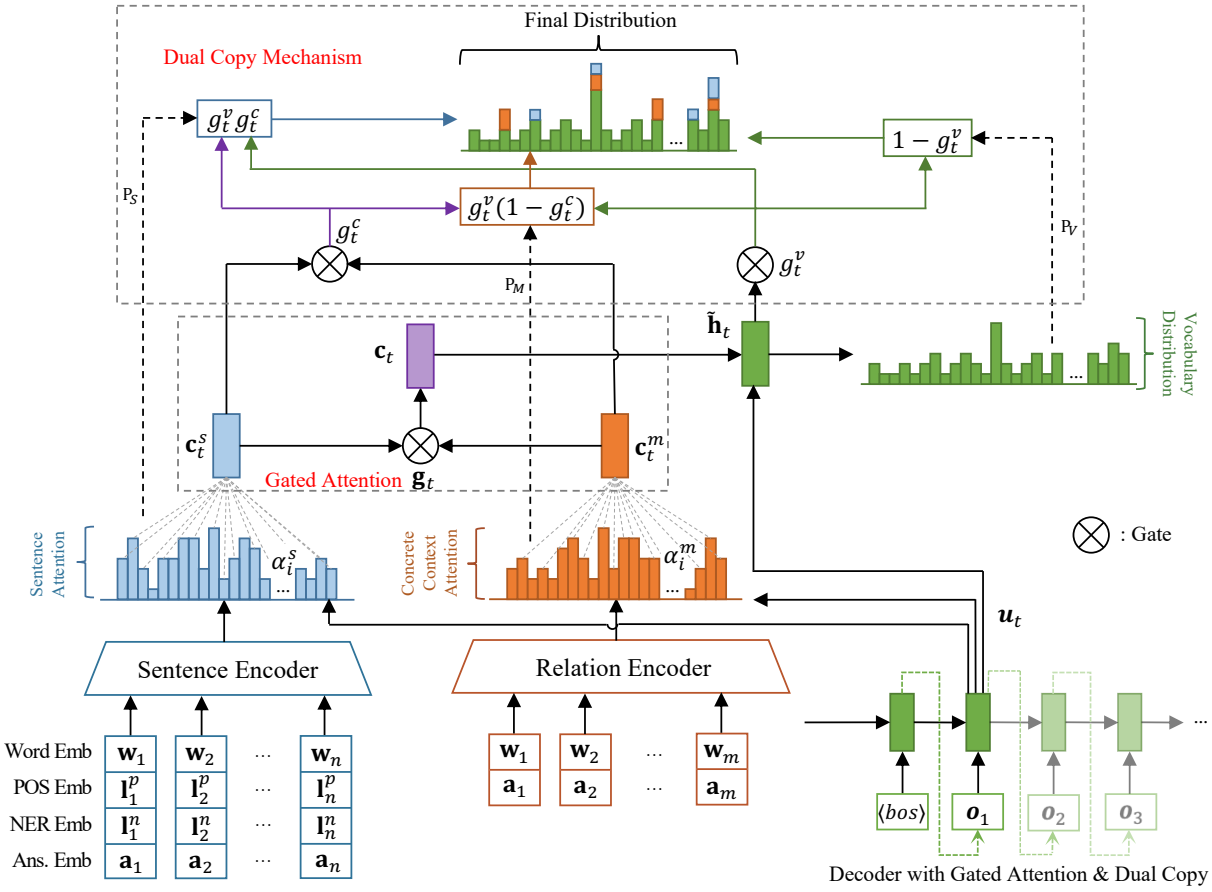


Figure 3: The framework of our proposed model. *(Best viewed in color)*

employed to jointly attend both contextualized information sources, and a dual copy mechanism copies words from either the sentence or the relation.

**Answer-aware Encoder.** We employ two encoders to integrate information from the unstructured sentence $S$ and the answer-relevant relation $M$ separately. Sentence encoder takes in feature-enriched embeddings including word embeddings $\mathbf{w}$, linguistic embeddings $\mathbf{l}$ and answer position embeddings $\mathbf{a}$. We follow (Zhou et al., 2017) to transform POS and NER tags into continuous representation ($\mathbf{l}^p$ and $\mathbf{l}^n$) and adopt a BIO labelling scheme to derive the answer position embedding (B: the first token of the answer, I: tokens within the answer fragment except the first one, O: tokens outside of the answer fragment). For each word $w_i$ in the sentence $S$, we simply concatenate all features as input: $\mathbf{x}_i^s = [\mathbf{w}_i; \mathbf{l}_i^p; \mathbf{l}_i^n; \mathbf{a}_i]$. Here $[\mathbf{a}; \mathbf{b}]$ denotes the concatenation of vectors $\mathbf{a}$ and $\mathbf{b}$.

We use bidirectional LSTMs to encode the sentence $(\mathbf{x}_1^s, \mathbf{x}_2^s, ..., \mathbf{x}_n^s)$ to get a contextualized representation for each token:

$$\overrightarrow{\mathbf{h}}_i^s = \overrightarrow{\text{LSTM}}(\overrightarrow{\mathbf{h}}_{i-1}^s, \mathbf{x}_i^s), \quad \overleftarrow{\mathbf{h}}_i^s = \overleftarrow{\text{LSTM}}(\overleftarrow{\mathbf{h}}_{i+1}^s, \mathbf{x}_i^s),$$

where $\overrightarrow{\mathbf{h}}_i^s$ and $\overleftarrow{\mathbf{h}}_i^s$ are the hidden states at the $i$-th time step of the forward and the backward LSTMs. The output state of the sentence encoder is the concatenation of forward and backward hidden states: $\mathbf{h}_i^s = [\overrightarrow{\mathbf{h}}_i^s; \overleftarrow{\mathbf{h}}_i^s]$. The contextualized representation of the sentence is $(\mathbf{h}_1^s, \mathbf{h}_2^s, ..., \mathbf{h}_n^s)$.

For the relation encoder, we firstly join all items in the n-ary relation $M$ into a sequence. Then we only take answer position embedding as an extra feature for the sequence: $\mathbf{x}_i^m = [\mathbf{w}_i; \mathbf{a}_i]$. Similarly, we take another bidirectional LSTMs to encode the relation sequence and derive the corresponding contextualized representation $(\mathbf{h}_1^m, \mathbf{h}_2^m, ..., \mathbf{h}_n^m)$.

**Decoder.** We use an LSTM as the decoder to generate the question. The decoder predicts the word probability distribution at each decoding timestep to generate the question. At the $t$-th timestep, it reads the word embedding $\mathbf{w}_t$ and the hidden state $\mathbf{u}_{t-1}$ of the previous timestep to gen-

erate the current hidden state:

$$\mathbf{u}_t = \text{LSTM}(\mathbf{u}_{t-1}, \mathbf{w}_t). \quad (2)$$

**Gated Attention Mechanism.** We design a gated attention mechanism to jointly attend the sentence representation and the relation representation. For sentence representation $(\mathbf{h}^s_1, \mathbf{h}^s_2, ..., \mathbf{h}^s_n)$, we employ the Luong et al. (2015)'s attention mechanism to obtain the sentence context vector $\mathbf{c}^s_t$,

$$a^s_{t,i} = \frac{\exp(\mathbf{u}_t^\top \mathbf{W}_a \mathbf{h}^s_i)}{\sum_j \exp(\mathbf{u}_t^\top \mathbf{W}_a \mathbf{h}^s_j)}, \ \mathbf{c}^s_t = \sum_i a^s_{t,i} \mathbf{h}^s_i,$$

where $\mathbf{W}_a$ is a trainable weight. Similarly, we obtain the vector $\mathbf{c}^m_t$ from the relation representation $(\mathbf{h}^m_1, \mathbf{h}^m_2, ..., \mathbf{h}^m_n)$. To jointly model the sentence and the relation, a gating mechanism is designed to control the information flow from two sources:

$$\mathbf{g}_t = \text{sigmoid}(\mathbf{W}_g[\mathbf{c}^s_t; \mathbf{c}^m_t]), \quad (3)$$

$$\mathbf{c}_t = \mathbf{g}_t \odot \mathbf{c}^s_t + (\mathbf{1} - \mathbf{g}_t) \odot \mathbf{c}^m_t, \quad (4)$$

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_h[\mathbf{u}_t; \mathbf{c}_t]), \quad (5)$$

where $\odot$ represents element-wise dot production and $\mathbf{W}_g, \mathbf{W}_h$ are trainable weights. Finally, the predicted probability distribution over the vocabulary $V$ is computed as:

$$\text{P}_V = \text{softmax}(\mathbf{W}_V \tilde{\mathbf{h}}_t + \mathbf{b}_V), \quad (6)$$

where $\mathbf{W}_V$ and $\mathbf{b}_V$ are parameters.

**Dual Copy Mechanism.** To deal with the rare and unknown words, the decoder applies the pointing method (See et al., 2017; Gu et al., 2016; Gulcehre et al., 2016) to allow copying a token from the input sentence at the $t$-th decoding step. We reuse the attention score $\alpha^s_t$ and $\alpha^m_t$ to derive the copy probability over two source inputs:

$$\text{P}_S(w) = \sum_{i:w_i=w} \alpha^s_{t,i}, \ \text{P}_M(w) = \sum_{i:w_i=w} \alpha^m_{t,i}.$$

Different from the standard pointing method, we design a dual copy mechanism to copy from two sources with two gates. The first gate is designed for determining copy tokens from two sources of inputs or generate next word from $P_V$, which is computed as $g^v_t = \text{sigmoid}(\mathbf{w}^v_g \tilde{\mathbf{h}}_t + b^v_g)$. The second gate takes charge of selecting the source (sentence or relation) to copy from, which is computed as $g^c_t = \text{sigmoid}(\mathbf{w}^c_g[\mathbf{c}^s_t; \mathbf{c}^m_t] + b^c_g)$. Finally, we combine all probabilities $P_V$, $P_S$ and $P_M$ through two soft gates $g^v_t$ and $g^c_t$. The probability of predicting $w$ as the $t$-th token of the question is:

$$\begin{aligned} \text{P}(w) = &(1 - g^v_t)\text{P}_V(w) \\ &+ g^v_t g^c_t \text{P}_S(w) + g^v_t(1 - g^c_t)\text{P}_M(w). \end{aligned} \quad (7)$$

**Training and Inference.** Given the answer $A$, sentence $S$ and relation $M$, the training objective is to minimize the negative log-likelihood with regard to all parameters:

$$\mathcal{L} = -\sum_{Q \in \{Q\}} log\ \text{P}(Q|A, S, M; \theta), \quad (8)$$

where $\{Q\}$ is the set of all training instances, $\theta$ denotes model parameters and $\log P(Q|A, S, M; \theta)$ is the conditional log-likelihood of $Q$.

In testing, our model targets to generate a question $Q$ by maximizing:

$$\overline{Q} = \arg\max_Q log\ \text{P}(Q|A, S, M; \theta). \quad (9)$$

| | Du Split | Zhou Split |
|---|---|---|
| # pairs (Train) | 74689 | 86635 |
| # pairs (Dev) | 10427 | 8965 |
| # pairs (Test) | 11609 | 8964 |
| Sentence avg. tokens | 32.56 | 32.72 |
| Question avg. tokens | 11.42 | 11.31 |

Table 3: Dataset statistics on Du Split (Du et al., 2017) and Zhou Split (Zhou et al., 2017).

## 3 Experimental Setting

### 3.1 Dataset & Metrics

We conduct experiments on the SQuAD dataset (Rajpurkar et al., 2016). It contains 536 Wikipedia articles and 100k crowd-sourced question-answer pairs. The questions are written by crowd-workers and the answers are spans of tokens in the articles. We employ two different data splits by following (Zhou et al., 2017) [2] and (Du et al., 2017) [3]. In (Zhou et al., 2017), the original SQuAD development set is evenly divided into dev and test sets, while (Du et al., 2017) treats SQuAD development set as its test set and splits SQuAD training set into a training set (90%) and a development set (10%). We also filter out questions which do not have any overlapped non-stop words with the corresponding sentences and perform some preprocessing steps, such as tokenization and sentence splitting. The data statistics are given in Table 3.

[2] https://res.qyzhou.me/redistribute.zip
[3] https://github.com/xinyadu/nqg

| | Du Split (Du et al., 2017) | | | | | | Zhou Split (Zhou et al., 2017) | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | B1 | B2 | B3 | B4 | MET | R-L | B1 | B2 | B3 | B4 | MET | R-L |
| s2s (Du et al., 2017) | 43.09 | 25.96 | 17.50 | 12.28 | 16.62 | 39.75 | - | - | - | - | - | - |
| NQG++ (Zhou et al., 2017) | - | - | - | - | - | - | - | - | - | 13.29 | - | - |
| M2S+cp (Song et al., 2018) | - | - | - | 13.98 | 18.77 | 42.72 | - | - | - | 13.91 | - | - |
| s2s+MP+GSA (Zhao et al., 2018) | 43.47 | 28.23 | 20.40 | 15.32 | 19.29 | 43.91 | **44.51** | 29.07 | 21.06 | 15.82 | 19.67 | 44.24 |
| Hybrid model (Sun et al., 2018) | - | - | - | - | - | - | 43.02 | 28.14 | 20.51 | 15.64 | - | - |
| ASs2s (Kim et al., 2019) | - | - | - | 16.20 | 19.92 | 43.96 | - | - | - | 16.17 | - | - |
| Our model | **45.66** | **30.21** | **21.82** | **16.27** | **20.36** | **44.35** | 44.40 | **29.48** | **21.54** | **16.37** | **20.68** | **44.73** |

Table 4: The main experimental results for our model and several baselines. '-' means no results reported in their papers. (Bn: BLEU-n, MET: METEOR, R-L: ROUGE-L)

We evaluate with all commonly-used metrics in question generation (Du et al., 2017): BLEU-1 (B1), BLEU-2 (B2), BLEU-3 (B3), BLEU-4 (B4) (Papineni et al., 2002), METEOR (MET) (Denkowski and Lavie, 2014) and ROUGE-L (R-L) (Lin, 2004). We use the evaluation script released by Chen et al. (2015).

### 3.2 Baseline Models

We compare with the following models.

- s2s (Du et al., 2017) proposes an attention-based sequence-to-sequence neural network for question generation.
- NQG++ (Zhou et al., 2017) takes the answer position feature and linguistic features into consideration and equips the Seq2Seq model with copy mechanism.
- M2S+cp (Song et al., 2018) conducts multi-perspective matching between the answer and the sentence to derive an answer-aware sentence representation for question generation.
- s2s+MP+GSA (Zhao et al., 2018) introduces a gated self-attention into the encoder and a max-out pointer mechanism into the decoder. We report their sentence-level results for a fair comparison.
- Hybrid (Sun et al., 2018) is a hybrid model which considers the answer embedding for the question word generation and the position of context words for modeling the relative distance between the context words and the answer.
- ASs2s (Kim et al., 2019) replaces the answer in the sentence with a special token to avoid its appearance in the generated questions.

### 3.3 Implementation Details

We take the most frequent 20k words as our vocabulary and use the GloVe word embeddings (Pennington et al., 2014) for initialization. The embedding dimensions for POS, NER, answer position are set to 20. We use two-layer LSTMs in both encoder and decoder, and the LSTMs hidden unit size is set to 600.

We use dropout (Srivastava et al., 2014) with the probability $p = 0.3$. All trainable parameters, except word embeddings, are randomly initialized with the Xavier uniform in $(-0.1, 0.1)$ (Glorot and Bengio, 2010). For optimization in the training, we use SGD as the optimizer with a minibatch size of 64 and an initial learning rate of 1.0. We train the model for 15 epochs and start halving the learning rate after the 8th epoch. We set the gradient norm upper bound to 3 during the training.

We adopt the teacher-forcing for the training. In the testing, we select the model with the lowest perplexity and beam search with size 3 is employed for generating questions. All hyper-parameters and models are selected on the validation dataset.

## 4 Results and Analysis

### 4.1 Main Results

Table 4 shows automatic evaluation results for our model and baselines (copied from their papers). Our proposed model which combines structured answer-relevant relations and unstructured sentences achieves significant improvements over proximity-based answer-aware models (Zhou et al., 2017; Sun et al., 2018) on both dataset splits. Presumably, our structured answer-relevant relation is a generalization of the context explored by the proximity-based methods because they can only capture short dependencies around answer fragments while our extractions can capture both short and long dependencies given the answer fragments. Moreover, our proposed framework is a general one to jointly leverage structured

(a) Evaluation results on all sentences.

| | Hybrid | | | Our Model | | |
|---|---|---|---|---|---|---|
| | BLEU | MET | R-L | BLEU | MET | R-L |
| 0∼10 (72.8% of #) | 28.54 | 21.54 | 46.26 | 29.73 (4.17%) | 22.03 (2.27%) | 46.85 (1.28%) |
| >10 (27.2% of #) | 20.67 | 16.72 | 37.63 | 22.12 (7.01%) | 17.46 (4.43%) | 38.47 (2.23%) |

(b) Evaluation results on sentences with more than 20 words.

| | Hybrid | | | Our Model | | |
|---|---|---|---|---|---|---|
| | BLEU | MET | R-L | BLEU | MET | R-L |
| 0∼10 (58.3% of #) | 28.00 | 21.03 | 45.37 | 29.11 (3.96%) | 21.50 (2.21%) | 45.97 (1.31%) |
| >10 (26.6% of #) | 20.58 | 16.66 | 37.53 | 22.04 (7.09%) | 17.38 (4.30%) | 38.37 (2.24%) |

Table 5: Performance for the average relative distance between the answer fragment and other non-stop sentence words that also appear in the ground truth question (BLEU is the average over BLEU-1 to BLEU-4). Values in parenthesis are the improvement percentage of Our Model over Hybrid. (a) is based on all sentences while (b) only considers long sentences with more than 20 words.

relations and unstructured sentences. All compared baseline models which only consider unstructured sentences can be further enhanced under our framework.

Recall that existing proximity-based answer-aware models perform poorly when the distance between the answer fragment and other non-stop sentence words that also appear in the ground truth question is large (Table 1). Here we investigate whether our proposed model using the structured answer-relevant relations can alleviate this issue or not, by conducting experiments for our model under the same setting as in Table 1. The broken-down performances by different relative distances are shown in Table 5a. We find that our proposed model outperforms Hybrid (our re-implemented version for this experiment) on all ranges of relative distances, which shows that the structured answer-relevant relations can capture both short and long term answer-relevant dependencies of the answer in sentences. Furthermore, comparing the performance difference between Hybrid and our model, we find the improvements become more significant when the distance increases from "$0 \sim 10$" to "$> 10$". One reason is that our model can extract relations with distant dependencies to the answer, which greatly helps our model ignore the extraneous information. Proximity-based answer-aware models may overly emphasize the neighboring words of answers and become less effective as the useful context becomes further away from the answer in the complex sentences. In fact, the breakdown intervals in Table 5a naturally bound its sentence length, say for "$> 10$", the sentences in this group must be longer than 10. Thus, the length variances in these two intervals could be

**Sentence**: Beyoncé received critical acclaim and commercial success, selling one million digital copies worldwide in six days; The New York Times noted the album 's unconventional, unexpected release as significant.
**Reference Question**: How many digital copies of her fifth album did Beyonc sell in six days?
**Baseline Prediction**: How many digital copies did the New York Times sell in six days ?
**Structured Answer-relevant Relation**: (Beyoncé; received commercial success selling; one million digital copies worldwide; in six days)
**Our Model Prediction**: How many digital copies did Beyoncé sell in six days ?

**Sentence**: The daily mean temperature in January, the area's coldest month, is 32.6 °F (0.3 °C); however, temperatures usually drop to 10 °F (-12 °C) several times per winter and reach 50 °F (10 °C) several days each winter month.
**Reference Question**: What is New York City 's daily January mean temperature in degrees celsius ?
**Baseline Prediction**: What is the coldest temperature in Celsius ?
**Structured Answer-relevant Relation**: (The daily mean temperature in January; is; 32.6 °F (0.3 °C))
**Our Model Prediction**: In degrees Celsius , what is the average temperature in January ?

Figure 4: Example questions (with answers highlighted) generated by crowd-workers (ground truth questions), the baseline model and our model.

significant. To further validate whether our model can extract long term dependency words. We rerun the analysis of Table 5b only for long sentences (length $> 20$) of each interval. The improvement percentages over Hybrid are shown in Table 5b, which become more significant when the distance increases from "$0 \sim 10$" to "$> 10$".

## 4.2 Case Study

Figure 4 provides example questions generated by crowd-workers (ground truth questions), the base-

line Hybrid (Sun et al., 2018), and our model. In the first case, there are two subsequences in the input and the answer has no relation with the second subsequence[4]. However, we see that the baseline model prediction copies irrelevant words "The New York Times" while our model can avoid using the extraneous subsequence "The New York Times noted ..." with the help of the structured answer-relevant relation. Compared with the ground truth question, our model cannot capture the cross-sentence information like "her fifth album", where the techniques in paragraph-level QG models (Zhao et al., 2018) may help. In the second case, as discussed in Section 1, this sentence contains a few facts and some facts intertwine. We find that our model can capture distant answer-relevant dependencies such as "mean temperature" while the proximity-based baseline model wrongly takes neighboring words of the answer like "coldest" in the generated question.

### 4.3 Diverse Question Generation

Another interesting observation is that for the same answer-sentence pair, our model can generate diverse questions by taking different answer-relevant relations as input. Such capability improves the interpretability of our model because the model is given not only what to be asked (i.e., the answer) but also the related fact (i.e., the answer-relevant relation) to be covered in the question. In contrast, proximity-based answer-aware models can only generate one question given the sentence-answer pair regardless of how many answer-relevant relations in the sentence. We think such capability can also validate our motivation: questions should be generated according to the answer-aware relations instead of neighboring words of answer fragments. Figure 5 show two examples of diverse question generation. In the first case, the answer fragment 'Hugh L. Dryden' is the appositive to 'NASA Deputy Administrator' but the subject to the following tokens 'announced the Apollo program ...'. Our framework can extract these two answer-relevant relations, and by feeding them to our model separately, we can receive two questions asking different relations with regard to the answer.

[4]One might think that the two subsequences should be regarded as individual sentences, however, several off-the-shelf tools do recognize them as one sentence.

**Sentence**: In July 1960, NASA Deputy Administrator Hugh L. Dryden announced the Apollo program to industry representatives at a series of Space Task Group conferences.
**Relation 1**: (Hugh L. Dryden; [is] Deputy Administrator [of]; NASA)
**Question 1**: Who was the NASA Deputy Administrator in 1960 ?
**Relation 2**: (NASA Deputy Administrator Hugh L. Dryden; announced; the Apollo program to industry representatives at a series of Space Task Group conferences; In July 1960)
**Question 2**: Who announced the Apollo program to industry representatives ?

**Sentence**: One of the network's strike-replacement programs during that time was the game show Duel, which premiered in December 2007.
**Relation 1**: (the game show Duel; premiered; in December 2007)
**Question 1**: What game premiered in December 2007 ?
**Relation 2**: (One of the network's strike-replacement programs during that time; was; the game show Duel)
**Question 2**: What was the name of an network 's strike - replacement programs ?

Figure 5: Example diverse questions (with answers highlighted) generated by our model with different answer-relevant relations.

## 5 Related Work

The topic of question generation, initially motivated for educational purposes, is tackled by designing many complex rules for specific question types (Mitkov and Ha, 2003; Rus et al., 2010). Heilman and Smith (2010) improve rule-based question generation by introducing a statistical ranking model. First, they remove extraneous information in the sentence to transform it into a simpler one, which can be transformed easily into a succinct question with predefined sets of general rules. Then they adopt an overgenerate-and-rank approach to select the best candidate considering several features.

With the rise of dominant neural sequence-to-sequence learning models (Sutskever et al., 2014), Du et al. (2017) frame question generation as a sequence-to-sequence learning problem. Compared with rule-based approaches, neural models (Yuan et al., 2017) can generate more fluent and grammatical questions. However, question generation is a one-to-many sequence generation problem, i.e., several aspects can be asked given a sentence, which confuses the model during train and prevents concrete automatic evaluation. To tackle this issue, Zhou et al. (2017) propose the answer-aware question generation setting which assumes the answer is already known

and acts as a contiguous span inside the input sentence. They adopt a BIO tagging scheme to incorporate the answer position information as learned embedding features in Seq2Seq learning. Song et al. (2018) explicitly model the information between answer and sentence with a multi-perspective matching model. Kim et al. (2019) also focus on the answer information and proposed an answer-separated Seq2Seq model by masking the answer with special tokens. All answer-aware neural models treat question generation as a one-to-one mapping problem, but existing models perform poorly for sentences with a complex structure (as shown in Table 1).

Our work is inspired by the process of extraneous information removing in (Heilman and Smith, 2010; Cao et al., 2018). Different from Heilman and Smith (2010) which directly use the simplified sentence for generation and Cao et al. (2018) which only consider aggregate two sources of information via gated attention in summarization, we propose to combine the structured answer-relevant relation and the original sentence. Factoid question generation from structured text is initially investigated by Serban et al. (2016), but our focus here is leveraging structured inputs to help question generation over unstructured sentences. Our proposed model can take advantage of unstructured sentences and structured answer-relevant relations to maintain informativeness and faithfulness of generated questions. The proposed model can also be generalized in other conditional sequence generation tasks which require multiple sources of inputs, e.g., distractor generation for multiple choice questions (Gao et al., 2019b).

## 6 Conclusions and Future Work

In this paper, we propose a question generation system which combines unstructured sentences and structured answer-relevant relations for generation. The unstructured sentences maintain the informativeness of generated questions while structured answer-relevant relations keep the faithfulness of questions. Extensive experiments demonstrate that our proposed model achieves state-of-the-art performance across several metrics. Furthermore, our model can generate diverse questions with different structured answer-relevant relations. For future work, there are some interesting dimensions to explore, such as difficulty levels (Gao et al., 2019a), paragraph-level information (Zhao et al., 2018) and conversational question generation (Gao et al., 2019c).

## Acknowledgments

This work is supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (No. CUHK 14208815 and No. CUHK 14210717 of the General Research Fund). We would like to thank the anonymous reviewers for their comments. We would also like to thank Department of Computer Science and Engineering, The Chinese University of Hong Kong for the conference grant support.

## References

Ziqiang Cao, Furu Wei, Wenjie Li, and Sujian Li. 2018. Faithful to the original: Fact aware neural abstractive summarization. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C. Lawrence Zitnick. 2015. Microsoft coco captions: Data collection and evaluation server. *CoRR*, abs/1504.00325.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 376–380, Baltimore, Maryland, USA. Association for Computational Linguistics.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1342–1352, Vancouver, Canada. Association for Computational Linguistics.

Yifan Gao, Lidong Bing, Wang Chen, Michael R. Lyu, and Irwin King. 2019a. Difficulty controllable generation of reading comprehension questions. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization.

Yifan Gao, Lidong Bing, Piji Li, Irwin King, and Michael R. Lyu. 2019b. Generating distractors for reading comprehension questions from real examinations. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):6423–6430.

Yifan Gao, Piji Li, Irwin King, and Michael R. Lyu. 2019c. Interconnected question generation with

coreference alignment and conversation flow modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4853–4862, Florence, Italy. Association for Computational Linguistics.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy. PMLR.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany. Association for Computational Linguistics.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 140–149, Berlin, Germany. Association for Computational Linguistics.

Michael Heilman and Noah A. Smith. 2010. Good question! statistical ranking for question generation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 609–617, Los Angeles, California. Association for Computational Linguistics.

Yanghoon Kim, Hwanhee Lee, Joongbo Shin, and Kyomin Jung. 2019. Improving neural question generation using answer separation. In *AAAI Conference on Artificial Intelligence*.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Ruslan Mitkov and Le An Ha. 2003. Computer-aided generation of multiple-choice tests. In *Proceedings of the HLT-NAACL 03 Workshop on Building Educational Applications Using Natural Language Processing*, pages 17–22.

Nasrin Mostafazadeh, Ishan Misra, Jacob Devlin, Margaret Mitchell, Xiaodong He, and Lucy Vanderwende. 2016. Generating natural questions about an image. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1802–1813, Berlin, Germany. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Vasile Rus, Brendan Wyse, Paul Piwek, Mihai Lintean, Svetlana Stoyanchev, and Christian Moldovan. 2010. The first question generation shared task evaluation challenge. In *Proceedings of the 6th International Natural Language Generation Conference*.

Swarnadeep Saha and Mausam. 2018. Open information extraction from conjunctive sentences. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2288–2299, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Iulian Vlad Serban, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. 2016. Generating factoid questions with recurrent neural networks: The 30M factoid question-answer corpus. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 588–598, Berlin, Germany. Association for Computational Linguistics.

Linfeng Song, Zhiguo Wang, Wael Hamza, Yue Zhang, and Daniel Gildea. 2018. Leveraging context information for natural question generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 569–574, New Orleans,

Louisiana. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Xingwu Sun, Jing Liu, Yajuan Lyu, Wei He, Yanjun Ma, and Shi Wang. 2018. Answer-focused and position-aware neural question generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3930–3939, Brussels, Belgium. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Xingdi Yuan, Tong Wang, Caglar Gulcehre, Alessandro Sordoni, Philip Bachman, Saizheng Zhang, Sandeep Subramanian, and Adam Trischler. 2017. Machine comprehension by text-to-text neural question generation. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 15–25, Vancouver, Canada. Association for Computational Linguistics.

Yao Zhao, Xiaochuan Ni, Yuanyuan Ding, and Qifa Ke. 2018. Paragraph-level neural question generation with maxout pointer and gated self-attention networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3901–3910, Brussels, Belgium. Association for Computational Linguistics.

Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. 2017. Neural question generation from text: A preliminary study. In *Proceedings of the 6th CCF International Conference on Natural Language Processing and Chinese Computing (NLPCC)*, pages 662–671, Dalian, China.

# Deep copycat Networks for Text-to-Text Generation

**Julia Ive**[1], **Pranava Madhyastha**[2] and **Lucia Specia**[2]
[1]DCS, University of Sheffield, UK
[2]Department of Computing, Imperial College London, UK
j.ive@sheffield.ac.uk
{pranava,l.specia}@imperial.ac.uk

## Abstract

Most text-to-text generation tasks, for example text summarisation and text simplification, require copying words from the input to the output. We introduce copycat, a transformer-based pointer network for such tasks which obtains competitive results in abstractive text summarisation and generates more abstractive summaries. We propose a further extension of this architecture for automatic post-editing, where generation is conditioned over two inputs (source language and machine translation), and the model is capable of deciding where to copy information from. This approach achieves competitive performance when compared to state-of-the-art automated post-editing systems. More importantly, we show that it addresses a well-known limitation of automatic post-editing – overcorrecting translations – and that our novel mechanism for copying source language words improves the results.

## 1 Introduction

Text-to-text generation is generally addressed using sequence to sequence neural models (Sutskever et al., 2014). While initially proposed for machine translation (Bahdanau et al., 2015), these models have been shown to perform very well in monolingual variants of these tasks, such as text summarisation and text simplification (Nallapati et al., 2016a; Scarton and Specia, 2018). One limitation of such models is that they tend to be "over creative" when generating outputs, i.e. create a completely new output, which is not what they should do. Different strategies have been used to mitigate this issue, primarily in the context of abstractive text summarisation. The most prominent strategy is the use of pointer networks, which substantially outperform vanilla sequence to sequence models (Nallapati et al., 2016b; Gu et al., 2016; See et al., 2017). Pointer networks (Vinyals et al., 2015) are an extension of attentive recurrent neural network (RNN) architectures to use attention as a pointer to select which tokens of the input sequence should be copied to the output, as a switching mechanism between copying and generating new words.

In this paper we propose copycat, a flexible pointer network framework for text-to-text generation tasks. It differs from previous work in the following main ways: it is based on transformer networks (Vaswani et al., 2017), which have been shown to achieve superior performance in text generation tasks. While Gehrmann et al. (2018b) also provide contrastive experiments using transformers, they however randomly choose one of the attention heads as the copy-distribution, while we let the networks learn through all layers and use the attention heads from the final layer of the decoder to learn the distribution; thereby the decision between copying input words and generating new words is made at the highest level of abstraction. Further, in our model, we do not use coverage penalty and achieve similar performance as Gehrmann et al. (2018b).

For text summarisation, we show that our copycat model performs competitively with comparable models, but achieves a much higher novel n-gram generation rate (wrt original article) and a low repetition rate (within a summary) without any coverage penalty.

We further extend this architecture to the dual-source setting, where it is conditioned on two inputs and can copy from either of them. We apply this approach to automatic post-editing, where the network can generate new words, or copy words from either the source language or the original machine translation. We show that this is a promising approach especially to avoid overcorrecting high quality machine translations and for text domains where terminology (or jargon) should be preserved



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3227–3236, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

across languages, rather than translated.

To sum up, our **main contributions** are: (i) `copycat` networks – a transformer-based pointer network for text-to-text generation tasks (Section 3.2); ; and (ii) an extension of this model which can copy from multiple inputs (Section 3.3). Our models produce SOTA results in summarisation. Our datasets and settings are described in Section 4, and the experiments with text summarisation and automatic post-editing, in Section 5.

## 2 Related work

In this section we highlight the most closely related work on abstractive summarisation and automatic post-editing.

**Abstractive Summarisation** SOTA approaches using sequence to sequence models have evolved from vanilla attentional models (Rush et al., 2015; Nallapati et al., 2016a) to more advanced architectures that include the use of pointer networks (Nallapati et al., 2016b; See et al., 2017; Gu et al., 2016; Paulus et al., 2017), reinforcement learning (Paulus et al., 2017; Li et al., 2018; Pasunuru and Bansal, 2018; Chen and Bansal, 2018; Hsu et al., 2018) and content selection (Gehrmann et al., 2018a; Pasunuru and Bansal, 2018; Chen and Bansal, 2018).

Nallapati et al. (2016b) and Gu et al. (2016) propose the use of pointer-generator RNN-based networks to reduce out-of-vocabulary (OOV) rate. This idea was followed by See et al. (2017), which incorporates a coverage mechanism to avoid repetition of input words by keeping track of what has already been covered.

Reinforcement learning (RL) approaches optimise objectives for summarisation in addition to maximum likelihood. Paulus et al. (2017) combine `ROUGE` and maximum likelihood as training objectives, use a pointer network and – to avoid repetition – introduce intra-attention to the encoder and decoder that attends over the input and continuously generated output separately. Li et al. (2018) use a global summary quality estimator which is a binary classifier aiming to make the generated summaries indistinguishable from the human-written ones. Pasunuru and Bansal (2018) have a loss function based on whether keywords detected as salient are included in a summary, while Hsu et al. (2018) modulate the attention based on how likely a sentence is to be included in a summary, and Chen and Bansal (2018) follow an extractive-abstractive hybrid architecture to first extract full sentences from a document using a sentence-level policy gradient and then compress them.

Gehrmann et al. (2018a) also perform content selection, but on the level of phrases by treating this process as a sequence labelling task, and without RL. They first build a selection mask for the source document and then constrain a decoder on this mask.

The latter two represent the best overall approaches on common datasets, using either an RNN- or a Transformer- based architecture. We show that our `copycat` approach is competitive and is more abstractive (we describe this in the following sections).

**Automatic Post-Editing (APE)** (Simard and Foster, 2013; Chatterjee et al., 2017) aims to automatically correct errors in machine translation (MT) outputs in order to reduce the burden of human post-editors, especially with simple and repetitive corrections (e.g. typographic errors).

APE is usually addressed by monolingual translation models that "translate" from the raw MT to fixed MT (post-edits, PE). Given the high quality of current neural MT (NMT) outputs, the task has become particularly challenging. Human corrections to those outputs are rare and very context-dependent, which makes them difficult to capture and generalise. This increases the chance of APE systems to overfit or overcorrect new inputs at test time.

SOTA APE approaches tackle the task with the multi-source transformer architectures (Junczys-Dowmunt and Grundkiewicz, 2018; Tebbifakhr et al., 2018). Two encoders encode the source and the raw MT, respectively, and an additional target-source multi-head attention component is stacked on top of the original target-source multi-head attention component. These models are trained with millions of <source, MT, PE> triplets, which does not fit realistic scenarios where only a handful of post-editions exist. Special parallel training corpora with additional MT data are released to help the development (Negri et al., 2018). According to (Chatterjee et al., 2018), these SOTA systems modify up to 30% of the NMT input, out of which only 50% are positive changes, the rest are unnecessary paraphrases or deteriorate the output.

Our `copycat` networks address this problem: they are conservative and make very few careful

corrections to good quality raw MT – as a result of learning predominantly to copy, while still making substantial corrections when the raw MT is not as good.

## 3 copycat Networks

### 3.1 Transformer Architecture

We make use of the SOTA sequence to sequence modelling framework – the Transformer architecture (Vaswani et al., 2017). Our implementation is a multi-layer encoder-decoder architecture that uses the `tensor2tensor`[1] (Vaswani et al., 2018) library. We briefly describe the encoder and decoder blocks in what follows.

**Encoder block** ($\mathcal{E}$): The encoder block is made of 6 layers, each containing two sub-layers of multi-head self-attention mechanism followed by a fully connected feed forward neural network. We follow the standard implementation and employ residual connections between each layer, as well as layer normalisation. The output of the encoder forms the encoder memory which consists of contextualised representations for each of the source tokens ($M_{\mathcal{E}}$).

**Decoder block** ($\mathcal{D}$): The decoder block is similar to the standard transformer architecture and comprises of 6 layers. It contains an additional sub-layer that performs multi-head attention over the outputs of the encoder block. Specifically, a decoding layer $d_{l_i}$ is the result of multi-head attention over the outputs of the encoder which in turn is a function of the encoder memory and the outputs from the previous layer: $A_{\mathcal{D}\to\mathcal{E}} = f(M_{\mathcal{E}}, d_{l_{i-1}})$ where, the keys and values are the encoder outputs and the queries correspond to the decoder input.

In all cases we tie the source and target vocabulary. We also use Byte Pair Encoding (BPE) (Sennrich et al., 2016) word segmentation approach to pre-process the data before we fix the vocabulary. This also helps us in reducing the OOV rate and makes the models less dependent on the vocabulary of the training data. We train all our models with the cross entropy loss.

### 3.2 Pointer-generator Network

We base our pointer-generator on the approach of See et al. (2017), however we significantly depart from their formulation. We first obtain an attention distribution over the source and decoder at each time step. We use a simple dot product $\mathbf{a_t}$ over output of the last decoder layer ($A^{\text{fin}}_{\mathcal{D}\to\mathcal{E}}$) and the contextualised encoder representations for each of the source language tokens ($M_{\mathcal{E}}$):

$$\mathbf{a_t} = \texttt{softmax}(A^{\text{fin}}_{\mathcal{D}\to\mathcal{E}} \cdot M_{\mathcal{E}}) \quad (1)$$

where $A^{\text{fin}}_{\mathcal{D}\to\mathcal{E}}$ comes from final ($n = 6$) layer, just before applying layer normalisation and passing it through feed forward layers as shown in Figure 1. $A_{\mathcal{D}\to\mathcal{E}}$ already contains information from the source encoder. This differs both the RNN-based implementation in (See et al., 2017) and the copy-transformer (Gehrmann et al., 2018b) formulation, where a randomly chosen attention-head forms the basis for the copy-distribution. The generation probability $\Pr_{\text{gen}}$ for each time step is computed from the output of the decoder layer as:

$$\Pr_{\text{gen}} = \sigma(W^{\top}_{\text{gen}} A^{\text{fin}}_{\mathcal{D}\to\mathcal{E}}) \quad (2)$$

where $W$ is learned parameters and $\sigma$ is the `sigmoid` function which restricts $\Pr_{\text{gen}} \in [0, 1]$.

We then follow (Vinyals et al., 2015) and use $\Pr_{\text{gen}}$ as the switching function to decide between generating a word from the decoder vocabulary or copying the word directly from the source sequence by using $\mathbf{a_t}$ as:

$$\Pr = \Pr_{\text{gen}} \Pr_{\text{d}}(w_d) + (1 - \Pr_{\text{gen}})\mathbf{a_t} \quad (3)$$

where, $\Pr_{\text{d}}(w_d)$ is decoder's probability of generating output word from the decoder vocabulary.

We empirically observed that getting the new attention representation over the final layer is extremely important for obtaining a good estimate of the attention distribution.

### 3.3 Dual-source extension

We extend `copycat` networks for the case of APE where a dual-source encoder-decoder architecture has been shown to obtain superior results (Junczys-Dowmunt and Grundkiewicz, 2018). As previously mentioned, most of the translations in current APE datasets (Chatterjee et al., 2018) already have optimal quality, thereby models that alter these already accurate translations are sub-optimal. Based on this observation we propose a dual-source extension to our `copycat` model. We add an extra encoder ($\mathcal{E}_s$) that encodes the sentence in source language and

[1] https://github.com/tensorflow/tensor2tensor

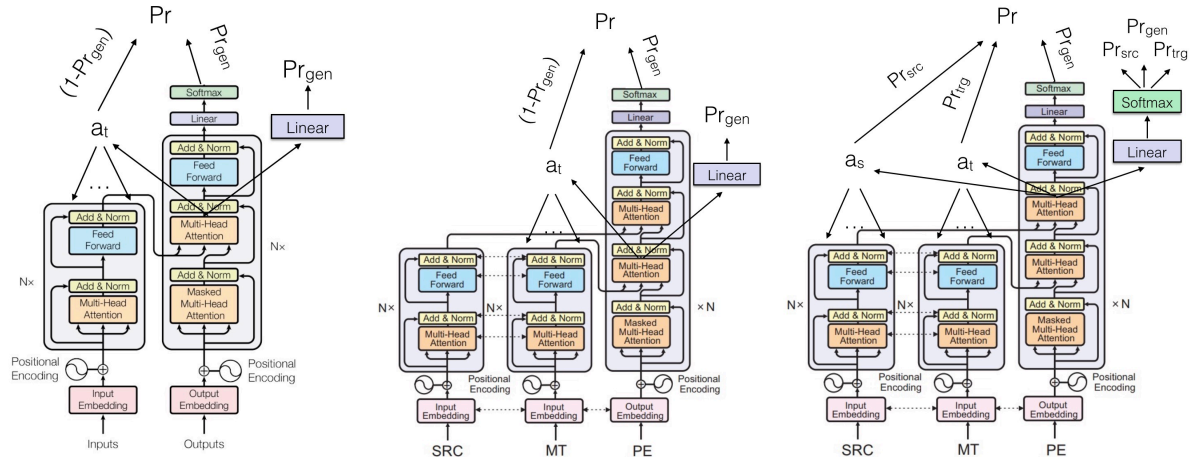


(a) `copycat` network (b) Dual-source `copycat` (c) Dual-source double-attention `copycat`

Figure 1: `copycat` architectures: (a) shows our `copycat` model for summarisation, where $a_t$ is computed on the final decoder layer and the last encoder layer; (b) is the dual-source extension for APE with an additional encoder; and (c) further extends (b) with an additional attention mechanism over the source language encoder. Down arrows indicate attention over encoder states. Last decoder state participates in this dot product computation. Then $\mathbf{a_s}$ and $\mathbf{a_t}$ participate in the final probability computation Pr.

the standard encoder ($\mathcal{E}$) which encodes the machine translated output. We also add an additional decoder layer $A'_{\mathcal{D}\to\mathcal{S}} = f(M_{\mathcal{E}_s}, A_{\mathcal{D}\to\mathcal{E}})$, where $A_{\mathcal{D}\to\mathcal{E}}$ is the standard decoding layer.

We maintain the pointer generator over the final decoder layer ($n = 6$) – $A^{\text{fin}}_{\mathcal{D}\to\mathcal{E}}$, similarly to Equation 3.

**Dual-source with double attention** We further extend the dual-source `copycat` network such that it now attends over both the source-language encoder and the translated output encoder, and therefore words from the source language can also be copied. We believe that copying words from source language would especially be useful for APE. Some corrections may involve retaining the original words in the source language due to a variety of reasons, including the use of proper nouns and jargon. Our extension involves modification in two components:

- **Attention over source:** We build on Equation 1 and add a similar attention over the source encoder as:

$$\mathbf{a_s} = \texttt{softmax}(A'^{\text{fin}}_{\mathcal{D}\to\mathcal{S}} \cdot M_{\mathcal{E}_s}) \qquad (4)$$

  where $M_{\mathcal{E}_s}$ is the source language encoder outputs.

- **Pointer-generation with multiple sources:** As there are three signals of information that influence the output decision – copy from source language, copy from the translated language and generate from decoder – we treat them as three different tasks. We thus modify Equation 2 with a normalised exponential function (`softmax`) instead of the `sigmoid` in the former. This results in:

$$\text{Pr}_{\text{gen}} = \frac{\exp\left(W_{\text{i=gen}} \cdot A'^{\text{fin}}_{\mathcal{D}\to\mathcal{S}}\right)}{\sum_{j\in\text{tasks}} \exp\left(W_j \cdot A'^{\text{fin}}_{\mathcal{D}\to\mathcal{S}}\right)} \qquad (5)$$

  where tasks $= \{\text{gen}, \text{trg}, \text{src}\}$; trg is for attention over the target language encoder, and src is for attention over the source language encoder. Notice that we use $A'^{\text{fin}}_{\mathcal{D}\to\mathcal{S}}$ as this includes both information from the target language encoder *and* the source language encoder.

- **Switching function:** Based on the probabilities from Equation 5, the new Pr is:

$$\text{Pr} = \text{Pr}_{\text{gen}}\text{Pr}_{\text{d}}(w_d) + \text{Pr}_{\text{trg}}\mathbf{a_t} + \text{Pr}_{\text{src}}\mathbf{a_s}$$

Figure 1 shows the variants of the `copycat` architectures.

## 4 Data and Settings

**Summarisation** We use the CNN/Daily Mail dataset, the most widely use dataset for abstractive summarisation (Hermann et al., 2015; Nallapati et al., 2016b). It contains online news articles

(800 tokens on average) and their abstracts (60 tokens on average). We used scripts provided by See et al. (2017) to obtain the same version of the the data, which has around 300K training pairs, 13K validation pairs and 11,490 test pairs. We apply BPE for word segmentation approach with 32K merge operations (Sennrich et al., 2016). During training and test we truncate articles to 400 BPE units as is standard practice (See et al., 2017).

**APE** Our APE datasets consist of <source, target, human post-edit> triplets. We use two variants: (i) the English-German (EN-DE) dataset from the IT domain provided by the WMT18/WMT19 APE shared task, and translated by a high-performing NMT system (45.8 BLEU).[2] It contains 13,442, 1,000 and 1,023 training, development and test triplets, respectively. (ii) the English-Latvian (EN-LV) dataset from the life sciences domain made available by Specia et al. (2018) containing 12,936 training / 1,000 development / 1,448 test triplets. Here, the outputs from NMT are of a lower quality (38.4 BLEU).

We use HTER (Snover et al., 2009) as the main metric and BLEU (Papineni et al., 2002) as the secondary metric, as in the WMT APE shared tasks (Chatterjee et al., 2018). HTER is defined as the minimum number of edits (substitution, insertion, deletion and shift) required to change an MT hypothesis so that it exactly matches a human post-edition of this hypothesis. BLEU measures $n$-gram precision between MT hypotheses and post-edits.

The EN-DE task is very challenging as the training data contains very few edits (0.15 HTER) and contain few context-dependent human corrections that are difficult to capture with machine learning methods. The EN-LV dataset has twice more edits (0.29 HTER). Actually, 36% of the German NMT data requires no corrections, and only 14% of the Latvian NMT data requires no correction.

**Hyperparameters:** For both tasks, we use Adam as optimiser (Kingma and Ba, 2014) and train the model until convergence with a fixed batch size of 50. Across experiments, we use shared source and target embeddings. We build on the tensor2tensor implementation using the transformer_base_v2 parameters for summarisation as best practise.[3] The transformer_base_v1 parameters with a learning rate of 0.05 with 8K warmup steps are used for APE. For other hyperparameters we use the default values in tensor2tensor.

For both tasks we apply early stopping with the patience of 10 epochs based on the validation ROUGE-L or BLEU score, for summarisation and APE respectively. For summarisation, we use a randomly chosen validation subset of 2k abstracts. During decoding, we use a beam search of size 5 and tune the $\alpha$ parameter (length penalty) on the validation subset. During decoding for APE, we use a beam search of size 10.

## 5 Results

In this section we present results of our experiments, first for summarisation (Section 5.1) and then for APE experiments (Section 5.2).

### 5.1 Summarisation

Table 1 shows results of our summarisation experiments. We report F1-scores for a standard set of ROUGE (Lin, 2004) scores: ROUGE-1, ROUGE-2 and ROUGE-L. They measure unigram, bigram recall and the longest in-sequence common to the generated and the reference abstracts, respectively. The scores are measured with the pyrouge toolkit.[4]

We also report results for a range of conceptually different systems, including the LEAD-3 baseline (the top 3 sentences of the source). copycat performs on par with the CopyTransformer. Our generated sentences for both systems are of 62 tokens on average, with references of 58 tokens on average.

To understand the model, we look at two aspects (i) the level of repetition within the generated abstracts, and (ii) the level of transfer from original to summarised text (i.e. the novelty rate).

Figure 2 reports the results of our analysis of repetitions in the outputs of our system and the reference abstracts. For unigrams, our model is comparable to the coverage mechanism of See et al. (2017). However, we note that our model is less effective than the one in See et al. (2017) for higher order $n$-grams, where repetitions are almost completely eliminated. This is done at the cost

[2] http://www.statmt.org/wmt19/ape-task.html. We also experimented with the additional – English-Russian – WMT19 dataset, but its quality is very questionable, as confirmed in personal communication with the task organisers.

[3] https://github.com/tensorflow/tensor2tensor#summarization

[4] We use the following wrapper: https://github.com/pltrdy/files2rouge

| | ROUGE | | |
|---|---|---|---|
| | 1 | 2 | L |
| LEAD-3 Baseline | 40.10 | 15.50 | 36.30 |
| Pointer-Generator (See et al., 2017) | 36.44 | 15.66 | 33.42 |
| Pointer-Generator + Coverage (See et al., 2017) | 39.53 | 17.28 | 36.38 |
| ML + RL (Paulus et al., 2017) | 39.87 | 15.82 | 36.90 |
| Saliency + Entailment reward (Pasunuru and Bansal, 2018) | 40.43 | 18.00 | 37.10 |
| Saliency to guide network (Li et al., 2018) | 38.95 | 17.12 | 35.68 |
| Inconsistency loss (Hsu et al., 2018) | 40.68 | 17.97 | 37.13 |
| Sentence Rewriting (Chen and Bansal, 2018) | 40.88 | 17.80 | 38.54 |
| Bottom-Up Summarisation (Gehrmann et al., 2018a) | 41.22 | 18.68 | 38.34 |
| CopyTransformer + Coverage Penalty (Gehrmann et al., 2018a) | 39.25 | 17.54 | 36.45 |
| copycat | 39.15 | 17.60 | 36.17 |

Table 1: Summarisation results on the CNN/Daily Mail dataset according to three standard metrics. LEAD-3 is a strong baseline which simply copies the first three sentences of the input. The second group contains RNN-based networks, while the last group contains transformer-based networks. All external results are from the original papers, which are briefly described in Section 2. Our results are significant with a 95% confidence interval of at most 0.25 as provided by the official ROUGE script.

| | |
|---|---|
| Article | ... **anne frank died of typhus in a nazi concentration camp at the age of 15** ... **her** older **sister , margot frank , died at least a month earlier than previously thought** ... they concluded that **anne and margot probably did not survive to march 1945** ... |
| copycat | **anne frank died of typhus in a nazi concentration camp at the age of 15 . her sister margot frank died at least a month earlier than previously thought . anne and margot probably did not survive to march 1945 .** |
| Reference | museum : anne frank died earlier than previously believed . researchers re-examined archives and testimonies of survivors . anne and older sister margot frank are believed to have died in february 1945 . |
| Article | ... billionaire businessman **tony toutouni** ... who **chronicles** his immensely **extravagant lifestyle on the photo-sharing site** and is usually seen next to stacks of cash and bikini-clad models - admits **‘ it ’s not that hard to get any girl you want ’ . his outrageous posts , which have seen him amass 750,000 followers in eight months** ... |
| copycat | **tony toutouni** , 42 , **chronicles** his devotion **extravagant lifestyle on the photo-sharing site . his outrageous posts have seen him amass 750,000 followers in eight months .** toutouni says **‘ it ’s not that hard to get any girl you want ’** |
| Reference | tony toutouni has amassed 750,000 followers on photo-sharing site in eight months thanks to outrageous posts . la-based entrepreneur is endlessly surrounded by supercars , piles of cash and bikini-clad women in pictures . he ’s friends with controversial instagram playboy dan bilzerian and says ‘ it ’s not hard to get any girl you want ’ |

Table 2: Examples of articles, their copycat and human reference summaries. Bold highlights segments repeated by copycat.

of generating fewer novel n-grams, as compared to the original abstract. Specifically, See et al. (2017) are only able to generate around 10% novel quadrigrams, whereas we generate 22%. Figure 3, which depicts the percentage of novel $n$-grams and sentences introduced by our copycat models as compared to full articles (without truncation).

The percentage of new words (unigrams) introduced by copycat is already in line with or higher than the percentage reported for other models (2.2% in (See et al., 2017), 0.5% Gehrmann et al. (2018a) and 1% in our case). The manual inspection of newly-introduced words confirmed their validity. From this inspection, we observed that the risk of creating new words as a consequence of combining BPE segments is very *low* (e.g., *constadedmind* produced by combining *con*, *sta*, *de*, *d* and *mind*).

We further manually inspect outputs of copycat systems to gain insights into their performance. Table 2 lists an example of produced outputs. We notice that the model is capable of

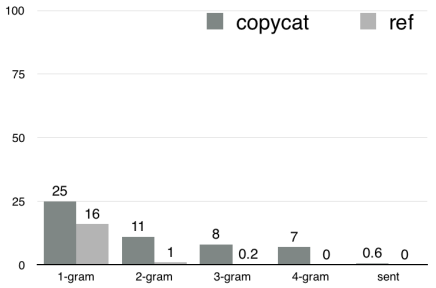


Figure 2: Percentage of repetitive $n$-grams (freq $> 1$) per sentence and sentences introduced by reference and `copycat`.

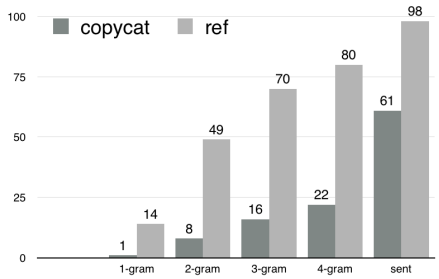


Figure 3: Percentage of novel $n$-grams per sentence and sentences introduced by reference and `copycat`.

both to selectively copying relevant parts of an article, as well as making deductions about the content, which often results in an abstract *different* from the reference abstract. Overall, the model actively attends to different parts of input and is capable of efficiently reorganising the information in the original article.

### 5.2 APE

We perform experiments with three models: (a) `NO-PNT` which is a reimplementation of the dual-source Transformer model of (Junczys-Dowmunt and Grundkiewicz, 2018); (b) `PNT-TRG`, `copycat` with attention over the MT; (c) `PNT-TRG+SRC`, `copycat` with double attention. Our baseline (`BASE`) is the standard "do nothing", i.e. the raw MT as provided in the datasets.

As the sizes of the APE datasets are very small, training only using this data is not successful. We thus pre-train all our models, as is a common practise for the APE task (Tebbifakhr et al., 2018). However, we do not generate new MT data. Instead we mimic MT data: we randomly delete, insert, shift or substitute words in the target side of parallel data to match `HTER` statistics of a respective APE dataset. For German, we randomly select 500K parallel data (only source and target) from the eSCAPE data provided with the task (Negri et al., 2018). For Latvian, we select those 500K from a corpus created by merging the Europarl (version 8)[5] and EMEA corpora.[6]

The data is truecased using the Moses toolkit scripts (Koehn et al., 2007). We again apply the BPE with 32K merge operations (Sennrich et al., 2016). Parameters of encoders for our dual `copycat` are shared. We use the label smoothing of 0.1. For the fine-tuning experiments, we reduce the learning rate to 0.001.

Table 4 shows results of our APE experiments. We compare our three setups to the baseline MT quality (`BASE`) without any APE.[7] We see that the performance of our `copycat` systems depends on the difficulty of the task: we observe an improvement for Latvian (+1 absolute `HTER` point improvement) with the lowest baseline quality (`HTER`= 0.29), and minor improvements for German with a higher baseline quality (on average 0.14 `HTER`). As a comparison, the best performing SOTA systems achieve only up to 0.4 `HTER` improvement for the German dataset and this is by using millions of training data, combined losses and ensembling techniques, whereas we use minimum external resources (Tebbifakhr et al., 2018; Junczys-Dowmunt and Grundkiewicz, 2018).[8]

For EN-DE, our `PNT-TRG` model performs a small number of corrections (mostly *accurate*) to the outputs: only 50 sentences, compared to the 200-300 sentences modified by the SOTA EN-DE APE systems (Chatterjee et al., 2018). For EN-LV, given the the raw translations are of lower quality, our models make more corrections: 330 sentences, which lead to overall improvements.

As a general observation, for both training sets, post-edit operations repeated more than once make around only 10% of all corrections. And this is not taking into account the fact that corrections also occur in different contexts. The majority of repetitive post-edits are corrections to punctuation (e.g., deletion of a comma) or auxiliary words (e.g., insertion/deletion of prepositions or articles). Therefore, we believe that especially in

[5] http://www.statmt.org/wmt17/translation-task.html

[6] http://opus.nlpl.eu/EMEA.php

[7] Our baseline scores differ from those provided by the task organisers because of differences in truecasing strategies.

[8] For instance, Junczys-Dowmunt and Grundkiewicz (2018) report 16.5/75.44 HTER↓/BLEU↑ for EN-DE over the 16.84/74.73 "do nothing" baseline.

| | |
|---|---|
| APE | Photomerge hat weniger Schritte . |
| MT | **die** Photomerge hat weniger Schritte . |
| REF | Photomerge hat weniger Schritte . |
| APE | weitere Informationen erhalten Sie von Pantone , Inc . , Flamstadt , NJ ( www.pantone.com ) . |
| MT | weitere Informationen erhalten Sie von Pantone , Inc . , Flamstadt , NJ ( **www.poor.com** ) . |
| REF | weitere Informationen erhalten Sie von Pantone , Inc . , Carlstadt , NJ ( www.pantone.com ) . |
| APE | siehe **"** Übersicht zu Gradationskurven . **"** |
| MT | siehe Übersicht zu Gradationskurven . |
| REF | siehe Übersicht zu Kurven . |

Table 3: Examples of correct (two first cases) and incorrect (third case) automatic post-edits produced from the EN-DE dataset by PNT-TRG+SRC.

| lang | BASE | NO-PNT | PNT-TRG | PNT-TRG+SRC |
|---|---|---|---|---|
| LV | 30.0 / 59.5 | 62.5 / 23.3 | **29.0** / 59.8 | **28.6** / 59.9 |
| DE | 17.8 / 72.3 | 18.5 / 71.2 | 17.9 / 72.3 | 17.8 / 72.4 |

Table 4: HTER↓ / BLEU↑ scores for the APE task. Boldface values mark statistically significant improvements over BASE, i.e. the raw MT output. We have performed significance testing via bootstrap resampling using the Multeval tool (Clark et al., 2011), the standard practice in APE.

the case of Latvian, this is probably the limit of what can be potentially learned from these small training datasets.

Furthermore, our intuition of the potential benefit of copying from the source is confirmed and for Latvian we observe an additional increase of 0.4 HTER for this model as compared to PNT-TRG.

The non-pointer system (NO-PNT) is not competitive and decreases the raw MT quality for both languages. The highest decrease is observed for Latvian (over 30 HTER points). We believe that the performance for Latvian drops so much because of the noisy original MT with a lot of repetitions that are carried to the output. These are mostly repetitions of the last word (typical error of low-quality NMT), that could be observed by eyeballing the output.

To assess the quality of corrections our systems perform to MT outputs, we manually analyse the corrections for German, for which we have in-house expertise for the type of analysis we present. For both PNT-TRG and PNT-TRG+SRC, 60% of the modified sentences are positive changes. The model is thus able to capture and make use of modification patterns found in the training data. These are mostly corrections of punctuation, deletions of prepositions, articles and auxiliary verbs. Negative changes are the same types of correction but in wrong contexts. For PNT-TRG+SRC, relevant source segments (those that the BASE MT failed to restore) are copied to the APE output: e.g., a correct web link, while it is distorted in the raw MT. Table 3 provides examples of positive and negative changes for both setups.

## 6 Conclusions

In this paper we propose copycat networks, a novel transformer-based pointer-generator network. For summarisation, we observe that copycat network is not only able to keep the repetition rate low, but also substantially increases the rate of novel n-grams. We further extend the copycat for automatic post-editing by conditioning over dual sources (source language and target language) and introducing the double-attention to copy from both sources. copycat is able to learn (a) more conservative models that copy more and generate less when the raw translations are already good, and (b) more aggressive models that make more corrections when the original translation quality is not as good. This shows that the architectures are flexible and adapt well to different datasets. Our source code is available at: https://github.com/ImperialNLP/CopyCat.

## Acknowledgments

The authors thank the anonymous reviewers for their useful feedback. This work was supported by the MultiMT (H2020 ERC Starting Grant No. 678017), MMVC (Newton Fund Institutional Links Grant, ID 352343575) and APE-QUEST (Connecting Europe Facility, project 2017-EU-IA-0151) projects.

## References


Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR*, San Diego, CA.

Rajen Chatterjee, Gebremedhen Gebremelak, Matteo Negri, and Marco Turchi. 2017. Online automatic post-editing for MT in a multi-domain translation environment. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 525–535.

Rajen Chatterjee, Matteo Negri, Raphael Rubino, and Marco Turchi. 2018. Findings of the wmt 2018 shared task on automatic post-editing. In *Proceedings of the Third Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 723–738, Belgium, Brussels. Association for Computational Linguistics.

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. *CoRR*, abs/1805.11080.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 176–181, Portland, Oregon, USA. Association for Computational Linguistics.

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018a. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109, Brussels, Belgium. Association for Computational Linguistics.

Sebastian Gehrmann, Yuntian Deng, and Alexander M Rush. 2018b. Bottom-up abstractive summarization. *arXiv preprint arXiv:1808.10792*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28*, pages 1693–1701. Curran Associates, Inc.

Wan Ting Hsu, Chieh-Kai Lin, Ming-Ying Lee, Kerui Min, Jing Tang, and Min Sun. 2018. A unified model for extractive and abstractive summarization using inconsistency loss. *CoRR*, abs/1805.06266.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2018. Ms-uedin submission to the wmt2018 ape shared task: Dual-source transformer for automatic post-editing. In *Proceedings of the Third Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 835–839, Belgium, Brussels. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180.

Piji Li, Lidong Bing, and Wai Lam. 2018. Actor-critic based training framework for abstractive summarization. *CoRR*, abs/1803.11070.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Ramesh Nallapati, Bing Xiang, and Bowen Zhou. 2016a. Sequence-to-sequence rnns for text summarization. *CoRR*, abs/1602.06023.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016b. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany. Association for Computational Linguistics.

Matteo Negri, Marco Turchi, Rajen Chatterjee, and Nicola Bertoldi. 2018. ESCAPE: a large-scale synthetic corpus for automatic post-editing. In *Proceedings of the 11th Language Resources and Evaluation Conference*, Miyazaki, Japan. European Language Resource Association.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Ramakanth Pasunuru and Mohit Bansal. 2018. Multi-reward reinforced summarization with saliency and entailment. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 646–653, New Orleans, Louisiana. Association for Computational Linguistics.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *CoRR*, abs/1705.04304.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *CoRR*, abs/1509.00685.

Carolina Scarton and Lucia Specia. 2018. Learning simplifications for specific target audiences. In *56th Annual Meeting of the Association for Computational Linguistics*, pages 712–718, Melbourne, Australia.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Michel Simard and George Foster. 2013. PEPr: Post-edit propagation using phrase-based statistical machine translation. In *Proceedings of the XIV Machine Translation Summit*.

Matthew Snover, Nitin Madnani, Bonnie J. Dorr, and Richard Schwartz. 2009. Fluency, adequacy, or HTER?: Exploring different human judgments with a tunable MT metric. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 259–268.

Lucia Specia, Frédéric Blain, Varvara Logacheva, Ramon Astudillo, and André F. T. Martins. 2018. Findings of the wmt 2018 shared task on quality estimation. In *Proceedings of the Third Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 702–722, Belgium, Brussels. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112.

Amirhossein Tebbifakhr, Ruchit Agrawal, Rajen Chatterjee, Matteo Negri, and Marco Turchi. 2018. Multi-source transformer with combined losses for automatic post editing. In *Proceedings of the Third Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 859–865, Belgium, Brussels. Association for Computational Linguistics.

Ashish Vaswani, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan N Gomez, Stephan Gouws, Llion Jones, Łukasz Kaiser, Nal Kalchbrenner, Niki Parmar, et al. 2018. Tensor2tensor for neural machine translation. *arXiv preprint arXiv:1803.07416*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

# Towards Controllable and Personalized Review Generation

**Pan Li**
New York University
Stern School of Business
44 West 4th Street, NY, NY, 10012
pli2@stern.nyu.edu

**Alexander Tuzhilin**
New York University
Stern School of Business
44 West 4th Street, NY, NY, 10012
atuzhili@stern.nyu.edu

## Abstract

In this paper, we propose a novel model *RevGAN* that automatically generates *controllable* and *personalized* user reviews based on the arbitrarily given sentimental and stylistic information. RevGAN utilizes the combination of three novel components, including self-attentive recursive autoencoders, conditional discriminators, and personalized decoders. We test its performance on the several real-world datasets, where our model significantly outperforms state-of-the-art generation models in terms of sentence quality, coherence, personalization and human evaluations. We also empirically show that the generated reviews could not be easily distinguished from the organically produced reviews and that they follow the same statistical linguistics laws.

## 1 Introduction

With the ever increasing interests in user-generated reviews on online marketplace websites, such as Amazon, Yelp and TripAdvisor, it is necessary to provide a range of tools that would encourage users to provide feedback in a more efficient and effective manner, as only a small fraction of users really take time to write their own reviews (Chen and Xie, 2008). Automatic review generation, for example, takes the product information and user behavior as input and generates user reviews following the arbitrarily given users' sentiment designation and writing style personalized towards the specific product and user.

Researchers have proposed various types of product review generation methods (Yao et al., 2017; Dong et al., 2017; Lipton et al., 2015; Radford et al., 2017) and achieved great performance. However, they did not consider the inner hierarchical word-sentence-paragraph structure within user reviews, thus making their generation results significantly limited in length and coherence. (Li et al., 2015; Zang and Wan, 2017) did include the hierarchical connection in their review generation model, but they did not address the problem of *controllable* and *personalized* review generation targeted at the specific product and user, which is essential for the usefulness of generated reviews. Most importantly, all the aforementioned generative models did not include *production descriptions* in the generation process, thus their generation results lack credibility and diversity.

To address these problems, we propose a novel model *RevGAN* that automatically generates high-quality user reviews given the information of product descriptions, sentiment labels and users' historical reviews. The proposed RevGAN model follows a three-staged process: In Stage 1, we propose to use *Self-Attentive Recursive Autoencoder* for mapping the discrete user reviews and product descriptions into continuous embeddings for the advantage of capturing the ''essence" of textual information and the convenience for subsequent optimization processes. In Stage 2, we utilize a novel *Conditional Discriminator* structure to control the sentiment of generated reviews by conditioning sentiment on the discriminator and forced the generator to adapt its generation policy correspondingly. Finally in Stage 3, to improve the personalization of generated reviews, we use a new *Personalized Decoder* method to decode the generated review embeddings according to users' writing styles extracted from their history corpus.

We conduct extensive experiments using multiple real-world datasets and show that the proposed RevGAN model significantly and consistently outperforms state-of-the-art baseline models and lead to the automated generation of reviews that are indeed very similar in style and content to the set of original reviews.

In general, this paper makes the following contributions:



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3237–3245,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

a. We propose a novel *RevGAN* model that automatically generates *controllable* and *personalized* reviews from product information, a set of user reviews and their writing styles. Especially. we propose three novel components of the generative framework: *Self-Attentive Recursive Autoenocder* that captures the hierarchical structure and latent semantic meanings of user reviews, *Conditional Discriminator* that generates controllable user reviews by conditioning the sentimental information on the discriminator to improve the generation performance in terms of sentence quality and context accuracy, and *Personalized Decoder* that takes the personalized writing style into account by concatenating the users' vocabulary preference onto the decoder to improve the personalization and credibility of the generated results, which is validated by the empirical human evaluation.

b. We empirically demonstrate that our proposed RevGAN model achieves state-of-the-art review generation performance, statistically and empirically outperforming several important benchmarks on multiple datasets. We also empirically show that the reviews generated by our method are very similar to the organically generated reviews and that the linguistic features of generated reviews follow the same statistical linguistics laws as reviews organically produced by the users.

## 2 Related Work

In this section, we briefly summarize the related work following two aspects covering previous work on automated review generation and GAN for NLG. We point out the connection and difference between our proposed model and prior literature, which leads to significant improvements of review generation performance.

### 2.1 Automated Review Generation

Researchers have been utilizing multiple versions of Seq2Seq(Sutskever et al., 2014) framework to generate online product reviews of good quality, including Aspect-Aware Representations (Ni and McAuley, 2018), Gated Contexts to Sequences(Tang et al., 2016), RNN(Yao et al., 2017), Aspect-Sentiment Score(Zang and Wan, 2017), Generative Concatenative Nets(Lipton et al., 2015) and Sentiment Units(Radford et al., 2017). In particular, (Li et al., 2015) proposed a two-stage LSTM neural network to construct a hierarchical autoencoder for long-text representation and generation.

However, these review generative models include neither product information nor users' writing styles into the generation process, thus making the generated reviews less persuasive. Also, they are limited in length and lack of coherence for neglecting the hierarchical connections within sentences, which are very important elements towards the helpfulness of a specific review(Mudambi and Schuff, 2010). Our RevGAN model, on the other hand, utilizes the combination of self-attentive hierarchical autoencoder and conditional discriminator for improved and controllable review generation, while we concatenate the contextual labels and users' history corpus into the personalized decoder at the same time. Experimental results support that our proposed model indeed achieves significantly better generation results compared to the prior literature.

### 2.2 GAN for NLG

GAN(Goodfellow et al., 2014) has become a powerful method for reconstruction and generation in real data space, which leaves great potential to be used for natural language generation purposes. Various methods have been proposed to get over the major problem of the discontinuity of textual information, including SeqGAN(Yu et al., 2017), TextGAN(Zhang et al., 2017), RankGAN(Lin et al., 2017) and LeakGAN(Guo et al., 2017). However, regarding long-text generation tasks, for all the models above, the computation complexity might be too high, thus failing to provide satisfying results. Nor do these models take contextual and personalized information into consideration for controllable generation. Conditional GAN(Mirza and Osindero, 2014; Hu et al., 2017; Dong et al., 2017) concatenates the supervised labels into the input of generator and is able to control the generation of simple sentences. However, considering the high dimension of latent embedding vectors, concatenating significantly lower dimensional supervised information into the input might not be strong enough to force the generator to update towards the designated direction.

Therefore, to address these problems, we propose a novel conditional discriminator model which conditions the sentiment label on the discriminator to artificially change how the discriminator works, and force it backpropagate the loss

function that could make the generator to learn what the user really want. Experimental results reported in Section 5 show that we outperform all other GAN-based-NLG models in the generation performance, and outperform Conditional GAN in terms of sentiment accuracy as well.

## 3 Models

In this section, we introduce the proposed *RevGAN* model for review generation that includes three novel components: Self-Attentive Recursive AutoEncoder, Conditional Discriminator and Personalized Decoder. We first use Self-Attentive Recursive AutoEncoder to map the discontinuous user reviews and product descriptions into a latent continuous space, and utilize a novel version of cGAN to generate review embeddings for subsequent personalized decoding and review generation. Experimental results show that the combination of all three novel components achieves state-of-the-art review generation performance.

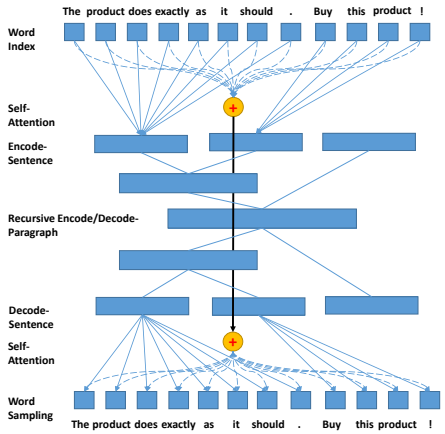


Figure 1: A Brief Overview of Self-Attentive Recursive AutoEncoder

### 3.1 Self-Attentive Recursive AutoEncoder

The proposed Self-Attentive Recursive AutoEncoder is illustrated in Figure 1. We implement a bidirectional Gated Recurrent Unit (GRU) (Cho et al., 2014) neural network as the encoder and the decoder in the model respectively. Compared with classical models like RNN or LSTM, GRU is computationally more efficient and better captures latent semantic meanings. We split each user review or product description into single sentences, and then map each sentence to their corresponding word indexes in the pre-defined dictionary. The index sequences constitute the input of our proposed model.

We use $R$ to represent a certain review consisting of $N^R$ sentences as $R = \{s^1, s^2, \cdots, s^{N_R}\}$. Each sentence $s$ consists of $N^s$ words as $s = \{w^1, w^2, \cdots, w^{N_s}\}$, where $w_i$ represents the index of certain word in the vocabulary with size $V$. We denote $[W^z, W^r, U^z, U^r]$ as the weight matrices for update gates and reset gates, $z_t, r_t$ as the status for the update gate and reset gate, and $x_t, h_t$ as the input and output vector at time $t$ respectively. GRU learns the hidden representations using the following equations, while the hidden state $h_t$ at the end of the sequence constitutes latent sentence embeddings.

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z)$$

$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r)$$

$$h_t = (1-z_t)h_{t-1} + z_t \sigma_h(W_h x_t + U_h(r_t h_{t-1}) + b_h)$$

Besides, to capture the relative position representations. we incorporate self-attentive mechanism (Shaw et al., 2018) during encoding process. Typically, each output element $h_i$ is computed as weighted sum of a linearly transformed input elements

$$h_i = \sum_{j=1}^{n} \alpha_{ij}(x_j W^V)$$

Each weight coefficient $\alpha_{ij}$ is computed using a softmax function

$$\alpha_{ij} = \frac{expe_{ij}}{\sum_{k=1}^{n} expe_{ik}}$$

And $e_{ij}$ is computed using a compatibility function that compares two input elements correspondingly. We visualize the self-attention mechanism with an example in Figure 2.

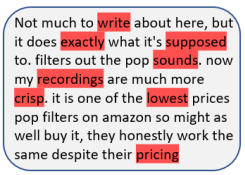


Figure 2: Visualization of Self-Attentive Mechanism

After getting the sentence-level embeddings within each review, we merge those sentence embeddings in a recursive way to obtain paragraph embeddings via a binary tree structure. We denote the embedding for sentence s as $e_s$, and during the

encoding process the first parent node $y_2$ is computed from the first two children node $(e_1, e_2)$ by the standard neural network layer:

$$y_2 = f(W_e(e_1 : e_2) + b)$$

where $(e_1 : e_2)$ is the natural concatenation of these two embedding vectors, $W_e$ is the weight matrix with twice the size of the embedding vectors. The second parent node $y_3$ will be computed from the concatenation of the first parent node $y_2$ and the following embedding vector $e_3$:

$$y_3 = f(W_e(y_2 : e_3) + b)$$

We obtain the representation of the rest nodes similarly. The embedding for root node constitutes the entire review embeddings. The training process of $W_e$ follows a standard MLP network, where we optimize the reconstruction loss over every layer of the binary tree.

To unfold the recursive autoencoder, we'll start from the root node $y_{N^R}$ of the binary tree. By utilizing another MLP network, we expand the paragraph embedding vector to two vectors: the leaf node and the lower level parent node, where the parent node would go through the same procedure until all of the leaf nodes in the binary tree are deciphered.

$$(y_{N^R-1} : e_{N^R}) = f'(W'_e y_{N^R} + b)$$

Finally, we assemble the paragraph from the bottom leaf node to the top one such that $R = \{e_1, e_2, \cdots, e_{N^R}\}$ to complete the review reconstruction process.

### 3.2 Conditional Discriminator

To generate meaningful reviews from the product information, we construct a cGAN model to transfer product descriptions into user reviews given specific sentiment labels. However, unlike the traditional conditional GAN methods (Mirza and Osindero, 2014; Hu et al., 2017), we do not concatenate the sentiment label directly into latent codes; considering the relatively high dimensions of latent embeddings, concatenating the sentiment scalar into the input might not be powerful enough to force the generator to update itself to match with the designated sentiment. Thus, we condition the sentiment labels on the discriminator to artificially change the rules that the discriminator works, and force it backpropagate loss functions that update generator policy correspondingly. The generated reviews and original reviews with the opposite sentiment are judged as negative examples, while only the original reviews that matches with the given sentiment are judged as positive examples by the conditional discriminator, as we propose the novel conditional discriminator D (for positive sentiment):

$$D(x|c) = \begin{cases} 1, \text{organic positive reviews} \\ 0, \text{others} \end{cases}$$

For generator, the model is optimized to minimize the reconstruction error between generated reviews and original reviews, $L(\theta^G) = KL(p_{real}(x)||p_G(x;\theta))$ where KL stands for Kullback-Leibler divergence. The loss function of discriminator D is: $L(\theta^D, \theta^G) = -\frac{1}{2}E_{x\sim data}logD(x) - \frac{1}{2}E_x log(1 - D(G(z)))$ Under ideal circumstances when generator and discriminator both reach their equilibrium, we could get that $D(x) = D(G(x)) = \frac{1}{2}$ and that the generated reviews are indeed indistinguishable from the original reviews from the discriminator point of view.

The core idea lies in that, by artificially forcing the discriminator to take certain type of reviews as real samples, generator should learn the conditioned information and then transform the generated data distribution. This unique structure of GANs makes possible the controllable review generation process, and experimental results support the strength of our model over classical cGAN models.

### 3.3 Personalized Decoder

To personalize the generation process, apart from the conditioning of sentiment labels, we also take the users' specific writing styles into account. We provide the definition of writing style according to (Zheng et al., 2006) :

**Writing Style** refers to the user's distinctive vocabulary choices and style of expression in his review creations.

Assuming that the historical reviews written by user i contain $[R_{i1}, R_{i2}, \cdots, R_{iN_i}]$, we calculate the usage frequency of each word from the corpus, which is denoted as a V-dimensional writing style vector $W_{style}$. The intuition is that, during the decoding process, instead of generating each word right from the calculated word distribution via GRU network, we concatenate the writing

style vector onto the distribution and sample the generated word afterwards, which would be determined by both the writing style vector $W_{style}$ and the distribution vector $W$:

$$W' = W * W_{style}$$

Note that, to deal with the cold start problem when the user has no historical reviews, we could simply set the writing style vector as identity matrix $W_{style} = I$ and generate the reviews under normal settings. Experimental results show that the involvement of personalized information (sentiment information and writing style) indeed improve the generation results and the helpfulness score from the empirical study as well.

## 4 Experimental Results

### 4.1 Dataset

To empirically validate our proposed model, we implemented RevGAN on three subsets of the Amazon Review Dataset(He and McAuley, 2016; McAuley et al., 2015)[1], namely Musical Instrument, Automotive and Patio which include 44,006 reviews written by 3,697 users on 6.039 items.

### 4.2 Experiment Settings

The self-attentive recursive autoencoder is implemented by bidirectional GRUs with embedding dimension 300. GRU parameters and word embeddings are initialized from a uniform distribution between [-0.1,0.1]. The initial learning rate is 1e-3, which will be halved every 50 epochs until convergence. Batch size is set to 128 (128 sentences across review documents) for batch normalization(Ioffe and Szegedy, 2015). Sentences would be padded to the maximum length within each batch. Gradient clipping(Gulrajani et al., 2017) is adopted by scaling the gradients when the norm exceeds the threshold 1. For the recursive structure, the parameter settings are the same with sentence-level autoencoder only except that the size of the weight matrix is $600 \times 300$. The beam size for beam searching(Wiseman and Rush, 2016) would be fixed as 3. To validate the emotion label for each review, we implemented the state-of-the-art sentiment classifier VADER(Gilbert, 2014)[2] to label the sentiment score for each review. The baseline SeqGAN, RankGAN and LeakGAN models are implemented through the Texygen(Zhu et al., 2018)[3] toolkit. The generator and the conditional discriminator of GANs are both set as Multilayer Perceptron(Rumelhart et al., 1985) (MLP) with 300 hidden layers. Their parameters are initialized from the normal distribution N(0,0.02). The learning rates for generator and conditional discriminator are fixed at 5e-5 and 1e-5 respectively. During each epoch, generator G would iterate 5 times while discriminator D would only iterate 1 time. The model updates 30,000 times in total. We implemented our model on a Tesla K80 GPU within PyTorch[4] environment, where the whole training takes about 12 hours.

### 4.3 Evaluation Metrics

To demonstrate that our purposed model indeed achieves the state-of-the-art review generation performance, we implement various evaluation metrics, including distribution-based Log-Likelihood and Perplexity, coherence-based Word Mover Distance (WMD) (Kusner et al., 2015), ngram-based BLEU (Papineni et al., 2002) and ROUGE (Lin, 2004), contextual label accuracy and human evaluation to measure the performance of review generation. Specifically, following the same metric as (Dong et al., 2017), we use sentiment accuracy, the ratio of the reviews whose sentiment matches with the given label, as an important indication of the personalization ability of the generator. The higher sentiment accuracy is, the better it could provide supervised generated results. Besides, we conduct the human evaluation to assess the quality and helpfulness of generated results by randomly selecting the same number of reviews from the original dataset, the generation of RevGAN and the generation of other baseline models and asking the participants to analyze which ones are generated by the machine and which ones are really created by humans, where significance test shows that our generated reviews are indeed indistinguishable from the original data.

### 4.4 Baseline Models

To demonstrate that our purposed model indeed achieves the state-of-the-art review generation performance, we compare our model across various evaluation metrics with several important benchmarks, including charRNN (Yao et al., 2017),

[1] http://jmcauley.ucsd.edu/data/amazon/
[2] https://github.com/cjhutto/vaderSentiment
[3] https://github.com/geek-ai/Texygen
[4] https://pytorch.org/

MLE (Bahl et al., 1990), SeqGAN (Yu et al., 2017), LeakGAN (Guo et al., 2017), RankGAN (Lin et al., 2017) and Attr2Seq (Dong et al., 2017). Besides, to verify the effectiveness of combining three novel components into the RevGAN model, we also compare the performance between RevGAN+CD (Conditional Discriminator), RevGAN+CD+SA (Self-Attentive Autoencoder) and RevGAN+CD+SA+PD (Personalized Decoder). The results show that our model indeed outperforms all the selected benchmark models significantly and consistently.

### 4.5 Significance Testing

We conduct significance tests to identify whether the difference between two review generation algorithms could indicate a difference in true system quality. Typically, following(Koehn, 2004), we use bootstrap re-sampling methods to get the asymptotic standard error of the estimated value of the evaluation metrics. Then the paired two-sample t-test could be used to test the significance whether their population means differ statistically.

In terms of the indistinguishableness of the generated results, we conduct a chi-square test for independence to test whether there is a significant association between the human assessment and the actual value. As the results of statistical test are insignificant, we could then claim that our generated reviews are indistinguishable from original ones in the sense that human can't separate them apart.

### 4.6 Evaluation of review generation

To illustrate the superiority and generalizability of our RevGAN model, we implement our model on three different domains of the Amazon Review Dataset including musical instruments, automotive and patio products. The summary of our experiment results is reported in Table 1, from which we could clearly observe that, compared with the baseline text-generation models, our proposed RevGAN model performs significantly better in sentence quality and coherence performance. On average, we could witness a 5% increase in Word Mover Distance (WMD), 80% improvement in BLEU and 10% rising in ROUGE. Besides, the comparison between different variations of RevGAN model verifies that indeed the combination of all three novel components gives the best generation performance. By deploying bootstrap re-sampling techniques introduced in the previous section, we conduct hypothesis tests where all the tests confirm the significant improvement of our RevGAN model. In that sense, we claim that our model achieved the state-of-the-art results on review generation. We also showcase some generated reviews at the end of this section.

### 4.7 Evaluation of controllable generation

In this part, we evaluate the controllable generation performance of our purposed model by pre-setting the contextual labels. We fixed the sentiment label as 'positive' and 'negative' respectively conditioned on the discriminator, and then evaluate the sentiment accuracy of the generated reviews. The results are reported in Table 2, where our model beats the state-of-the-art algorithm Attr2Seq(Dong et al., 2017) and the classical model Conditional GAN where we condition the same sentiment on these two models as well.

### 4.8 Evaluation of Personalized Generation

Besides the statistical and semantical metrics, we also design an empirical study to test the personalized performance of our generated reviews. We randomly select 15 reviews to include in each questionnaire, 5 from the original dataset, 5 from RevGAN generated results with personalization and 5 from RevGAN generated results without personalization, and ask participants to analyze which ones are generated by the machine and which ones are really created by humans. Besides, they are also asked to assess the helpfulness of each review by choosing the helpfulness score scale 1-5 for each review. We sent out 100 questionnaires in total, and get 36 responses, the confusion matrix of which is reported in Table 3. To test that whether the RevGAN generated reviews are indeed statistically indistinguishable from original ones, we run chi-test for significance testing:

$$\chi^2 = 0.012 < \chi^1_{0.95}$$

which shows that, under 95% confident interval, we could claim that there's no statistical difference between our machine-generated reviews and those actually written by humans.

Besides, the results indicate that our generated reviews have no statistical difference in terms of the helpfulness scores from those written by consumers towards certain products, with average helpfulness scores 3.10 and 3.03 for machine-generated and real-world reviews respectively. Thus, based on the t-test, we accept the hypothesis that there's no statistical difference between

| Models | Log-Likelihood | WMD | PPL | BLEU-4(%) | ROUGE-L(%) |
|---|---|---|---|---|---|
| SeqGAN | -86699 | 1.869 | 22.60 | 15.06 | 38.30 |
| LeakGAN | -108581 | 2.324 | 24.09 | 14.98 | 37.73 |
| RankGAN | -73309 | 1.862 | 22.45 | 14.92 | 37.72 |
| charRNN | -100430 | 1.976 | 22.07 | 11.46 | 33.60 |
| MLE | -54338 | 2.106 | 17.15 | 9.62 | 31.89 |
| Attr2Seq | -56298 | 2.077 | 21.00 | 11.48 | 32.22 |
| RevGAN+CD | -80386 | 2.097 | 19.71 | 21.32 | 39.47 |
| RevGAN+CD+SA | -51549 | 2.030 | 17.45 | 24.44 | 41.32 |
| RevGAN+CD+SA+PD | **-34305**** | **1.762**** | **17.00*** | **27.16**** | **44.63**** |

Table 1: Comparison of Experimental Results on Amazon Review Dataset (** stands for significance under 99% confidence, * stands for 95% confidence)

| Metrics | RevGAN+CD | Attr2Seq | cGAN |
|---|---|---|---|
| Accuracy | **0.842*** | 0.762 | 0.665 |

Table 2: Controllable Generation Accuracy

those two groups in terms of helpfulness as well.

$$t = \frac{\bar{X_1} - \bar{X_2}}{s_p\sqrt{\frac{2}{n}}} = 0.77 < t_{0.95}$$

And finally, we would conduct t-test over the performance of personalized and non-personalized generated results with helpfulness score 3.10 and 2.91, which indicates the significant improvement in helpfulness by the involvement of users' writing style.

$$t = \frac{\bar{X_1} - \bar{X_2}}{s_p\sqrt{\frac{2}{n}}} = 2.09 > t_{0.95}$$

| Empirical Test | Actual Value | |
|---|---|---|
| | Human-Written | RevGAN+PD |
| Human | 119 | 61 |
| Machine | 118 | 62 |

| Empirical Test | Actual Value | |
|---|---|---|
| | Human-Written | RevGAN |
| Human | 119 | 61 |
| Machine | 102 | 78 |

Table 3: Confusion Matrix of Empirical Test

### 4.9 Showcase

We present several showcases of our generated results with different contextual labels and domains as shown in Table 4. Additionally, we showcase the modification process in Table 5, where the personalized generated reviews tend to use more words from the user's history corpus.

Besides, we check if reviews generated by RevGAN would have the same linguistic features by testing two major statistical laws of linguistics(Altmann and Gerlach, 2016): Zipf Law(Zipf, 1935) and Heap Law(Herdan, 1964). The former states that if words are ranked according to their frequency of appearance $r = 1, 2, \cdots, V$, the frequency f(r) of the r-th word scales with the rank as $f(r) = \frac{\beta_Z}{r^{\alpha_Z}}$, while the latter states that the number of different words V scales with database size $N$ measured in the total number of words as $V \sim N^{\alpha_H}$. As shown in Figure 3 and 4, both the original reviews and the generated ones satisfy those two linguistic laws.

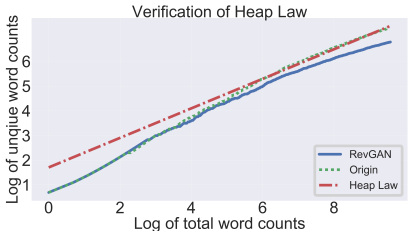


Figure 3: Verification of Heap's Law

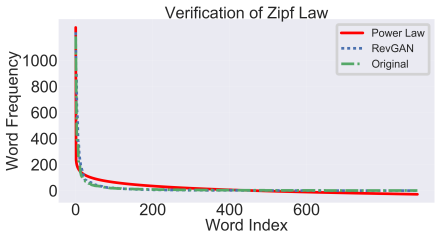


Figure 4: Verification of Zipf's Law

## 5 Conclusion

In this paper, we proposed *RevGAN* that automatically generates personalized product reviews from product embeddings as opposed to labels, which could output results targeting on specific products and users. To do this, we incorporate three novel components: self-attentive recursive autoen-

| Domain | Sentiment | Generated Reviews |
|---|---|---|
| Musical | Positive | These chords got me to play my guitar better in less than one day. An excellent overdrive and an incredible value. I'll use them all the time. |
| Musical | Negative | These pedals are not budget friendly. If you are looking for classic rock sounds, you won't love these expensive hardware. |
| Automotive | Positive | I bought two sets of seat covers and this roll kit. Both fit well and look good. They were much easier to slide over the leather seats. |
| Automotive | Positive | These seat covers look good and seem to be made of a good quality material. For the price, these are a great buy. |
| Patio | Negative | It is not recommended. The cover is a little tight and hard to open and close. |
| Patio | Positive | These traps have caught more mice than ever give. You only need a little peanut butter for the bait and tomcat would caught so many mice in one night. Will order again if needed . |

Table 4: Selected Personalized Generation Examples

| User History Reviews | RevGAN | RevGAN+PD |
|---|---|---|
| 1.They play well and hold up well(never had one break) They are my second favorite pick. But this pick is **better** in some songs than my favorite.<br>2.I ordered 5 different kind of picks and these were my favorite picks. They have a very **comfortable feel** and great sound! | The guitar has always made a quality and you would really love that life from it! | The guitar has always made **better** quality and you would really love that **comfortable feel** from it! |

Table 5: Personalized Review Generation Examples

coder, conditional discriminator and personalized decoder. Experimental results show that RevGAN performs significantly better than other baseline models and that our generated reviews are very similar to organically generated user reviews, as shown in Section 5.2 and Table 3.

As a part of the future work, we would like to improve the review generation process in a way that could receive several key words from users as input and generate reviews based on these prior information. Another direction of the future research, however, lies in developing novel methods that distinguish the type of reviews described in the paper and organic reviews.

## References


Eduardo G Altmann and Martin Gerlach. 2016. Statistical laws in linguistics. In *Creativity and Universality in Language*, pages 7–26. Springer.

Lalit R Bahl, Frederick Jelinek, and Robert L Mercer. 1990. A maximum likelihood approach to continuous speech recognition. In *Readings in speech recognition*, pages 308–319. Elsevier.

Yubo Chen and Jinhong Xie. 2008. Online consumer review: Word-of-mouth as a new element of marketing communication mix. *Management science*, 54(3):477–491.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Li Dong, Shaohan Huang, Furu Wei, Mirella Lapata, Ming Zhou, and Ke Xu. 2017. Learning to generate product reviews from attributes. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 623–632.

CJ Hutto Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth International Conference on Weblogs and Social Media (ICWSM-14). Available at (20/04/16) http://comp. social. gatech. edu/papers/icwsm14. vader. hutto. pdf*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. 2017. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5769–5779.

Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. 2017. Long text generation via adversarial training with leaked information. *arXiv preprint arXiv:1709.08624*.

Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, pages 507–517. International World Wide Web Conferences Steering Committee.

Gustav Herdan. 1964. Quantitative linguistics.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *International Conference on Machine Learning*, pages 1587–1596.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 conference on empirical methods in natural language processing*.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In *International Conference on Machine Learning*, pages 957–966.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Kevin Lin, Dianqi Li, Xiaodong He, Zhengyou Zhang, and Ming-Ting Sun. 2017. Adversarial ranking for language generation. In *Advances in Neural Information Processing Systems*, pages 3155–3165.

Zachary C Lipton, Sharad Vikram, and Julian McAuley. 2015. Generative concatenative nets jointly learn to write and classify reviews. *arXiv preprint arXiv:1511.03683*.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 43–52. ACM.

Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

Susan M Mudambi and David Schuff. 2010. Research note: What makes a helpful online review? a study of customer reviews on amazon. com. *MIS quarterly*, pages 185–200.

Jianmo Ni and Julian McAuley. 2018. Personalized review generation by expanding phrases and attending on aspect-aware representations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 706–711.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. 2017. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1985. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Jian Tang, Yifan Yang, Sam Carton, Ming Zhang, and Qiaozhu Mei. 2016. Context-aware natural language generation with recurrent neural networks. *arXiv preprint arXiv:1611.09900*.

Sam Wiseman and Alexander M Rush. 2016. Sequence-to-sequence learning as beam-search optimization. *arXiv preprint arXiv:1606.02960*.

Yuanshun Yao, Bimal Viswanath, Jenna Cryan, Haitao Zheng, and Ben Y Zhao. 2017. Automated crowdturfing attacks and defenses in online review systems. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1143–1158. ACM.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*, pages 2852–2858.

Hongyu Zang and Xiaojun Wan. 2017. Towards automatic generation of product reviews from aspect-sentiment scores. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 168–177.

Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. 2017. Adversarial feature matching for text generation. *arXiv preprint arXiv:1706.03850*.

Rong Zheng, Jiexun Li, Hsinchun Chen, and Zan Huang. 2006. A framework for authorship identification of online messages: Writing-style features and classification techniques. *Journal of the American society for information science and technology*, 57(3):378–393.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. *arXiv preprint arXiv:1802.01886*.

George Kingsley Zipf. 1935. The psycho-biology of language.

# Answers Unite!
# Unsupervised Metrics for Reinforced Summarization Models

**Thomas Scialom**[⋆‡], **Sylvain Lamprier**[‡], **Benjamin Piwowarski**[◇‡], **Jacopo Staiano**[⋆]
[◇] CNRS, France
[‡] Sorbonne Université, CNRS, LIP6, F-75005 Paris, France
[⋆] reciTAL, Paris, France
{thomas,jacopo}@recital.ai
{sylvain.lamprier,benjamin.piwowarski}@lip6.fr

## Abstract

Abstractive summarization approaches based on Reinforcement Learning (RL) have recently been proposed to overcome classical likelihood maximization. RL enables to consider complex, possibly non-differentiable, metrics that globally assess the quality and relevance of the generated outputs. ROUGE, the most used summarization metric, is known to suffer from bias towards lexical similarity as well as from suboptimal accounting for fluency and readability of the generated abstracts. We thus explore and propose alternative evaluation measures: the reported human-evaluation analysis shows that the proposed metrics, based on Question Answering, favorably compares to ROUGE – with the additional property of not requiring reference summaries. Training a RL-based model on these metrics leads to improvements (both in terms of human or automated metrics) over current approaches that use ROUGE as a reward.

## 1 Introduction

Summarization systems aim at generating relevant and informative summaries given a variable-length text as input. They can be roughly divided under two main categories, those adopting an *extractive* approach, *i.e.* identifying the most informative pieces from the input text and concatenating them to form the output summary; and those producing *abstractive* summaries, *i.e.* generating an output text whose tokens are not necessarily present in the input text.

While closer to human summarization, abstractive summarization is a much harder task and the need for faithful evaluation metrics is crucial to measure and drive the progress of such systems. The standard for evaluation of summarization systems is ROUGE (Lin, 2004): this metric can be considered as an adaptation of BLEU (Papineni et al., 2002), a scoring method for evaluation of machine translation systems; both based on *n*-gram co-occurrences, the latter favors precision while the former emphasizes recall.

Recent research works (Paulus et al., 2017; Pasunuru and Bansal, 2018; Arumae and Liu, 2019) have proposed to use evaluation metrics – and ROUGE in particular – to learn the model parameters through Reinforcement Learning (RL) techniques. This makes the choice of a good evaluation metric even more important. Unfortunately, ROUGE is known to incur several problems: in particular, its poor accounting for fluency and readability of the generated abstracts, as well as its bias towards lexical similarity (Ng and Abrecht, 2015). To emphasize the latter point, since ROUGE evaluates a summary against given human references, summarization models incur the risk of being unfairly penalized: a high quality summary might still have very few tokens/*n*-grams in common with the reference it is evaluated against.

In this work, we propose to overcome *n*-gram matching based metrics, such as ROUGE, by developing metrics which are better predictors of the quality of summaries. The contributions of this paper can be summarized as follows:

- Extending recent works (Eyal et al., 2019; Chen et al., 2018), we introduce new metrics, based on Question Answering, that do not require human annotations.
- We report a quantitative comparison of various summarization metrics, based on correlations with human assessments.
- We leverage the accuracy of the proposed metrics in several reinforcement learning schemes for summarization, including two unsupervised settings: *in-domain* (raw texts from the target documents) and *out-of-domain* (raw texts from another document collection).



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3246–3256, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

- Besides a quantitative evaluation of the generated summarizes, we qualitatively evaluate the performances of the different approaches through human assessment.

Our main results can be summarized as follows:

1. We show that fitting human judgments from carefully chosen measures allows one to successfully train a reinforcement learning-based model, improving over the state-of-the-art (in terms of ROUGE and human assessments).

2. we show that dropping the requirement for human-generated reference summaries, as enabled by the proposed metrics, allows to leverage texts in a self-supervised manner and brings clear benefits in terms of performance.

Section 2 introduces the metrics. Section 3 reviews related summarization systems and presents our proposed approaches. Section 4 presents our experimental results and discussions.

## 2 Evaluation Metrics

This section first describes our selection of existing summarization metrics and introduces our proposals. Then, we quantitatively compare them for abstractive summarization. For a comprehensive list of evaluation metrics, we refer the reader to Liu et al. (2016).

### 2.1 *n*-grams-based metrics

**TextRank** Automated summarization started with the development of extractive text summarization models. Many unsupervised models, that aim at computing a score between a sentence and document(s) were developed – the score attempting to reflect whether the sentence should be selected for building a summary (Nenkova, 2011). Such scores can thus be used as a proxy of the summary quality. We chose TextRank (Mihalcea and Tarau, 2004) – an extractive non-parametric summarization system inspired by PageRank (Page et al., 1999) – since it is well performing for extractive tasks and could be easily adapted for our needs. The algorithm builds a graph of sentences within a text based on their co-occurrences. Then, it assigns an importance score for each sentence based on a random walk on the resulting graph. The most important elements of the graph are considered as the ones that best describe the text. As a derivative usage, we propose to consider these importance scores to assess the quality of abstractive summaries in our study. This metric is referred to as TextRank in the following.

**ROUGE** Arguably the most popular metric for summarization at the moment, it provides a set of measures to compare automatically generated texts against one or more references (Lin, 2004). In particular, ROUGE-N is based on the count of overlapping *n*-grams, while ROUGE-L accounts for the longest common sub-sequence between the candidate and its corresponding reference(s).

**Novelty** As noted by See et al. (2017), abstractive summarization models do not produce novel *n*-grams as often as the reference summaries. Thus, to favor the generation of unseen words and produce more abstractive summaries, Kryściński et al. (2018) integrated *novelty* as a reward for reinforcement learning. It is defined as the fraction of unique *n*-grams in the summary that are novel, normalized by the length ratio of the generated and reference summaries.

### 2.2 Beyond n-grams

#### 2.2.1 Language Modeling

We investigate the use of language models as an evaluation metric. (ShafieiBavani et al., 2018) proposed to exploit word embeddings to train a model able to rate the generated summaries. Following neural language models (LM), we propose to consider the perplexity of the generated summary according to the BERT LM (Devlin et al., 2019), which demonstrated state of the art results in many NLP tasks. For our experiments, we used the publicly available pre-trained English "base" model.

#### 2.2.2 Question-Answering based Metrics

Question-Answering is related to summarization: the first work in this direction (Wu et al., 2002) introduced the notion of Answer-Focused Summarization, where answers to relevant questions on the source text are used to build the corresponding summary. Based on the intuition that a good-quality summary should provide the answers to the most relevant questions on a given input text, several works have proposed to adapt Question Answering (QA) for summary quality evaluation.

In that vein, (Pasunuru and Bansal, 2018) proposed to measure if answers contain the most *salient* tokens. Closer to our work, (Eyal et al.,

2019) proposed *APES*, a novel metric for evaluating summarization, based on the hypothesis that the quality of a generated summary is linked to the number of questions (from a set of relevant ones) that can be answered by reading it. In their proposed setup, two components are thus needed: (a) a set of relevant questions for each source document; and (b) a QA system. For each summary to assess, questions are successively generated from a reference summary, by masking each of the named entities present in this reference, following the methodology described in (Hermann et al., 2015). This results in as many triplets $(input, question, answer)$ as named entities present in the reference summary, where *input* denotes the summary to assess, *question* refers to the sentence containing the masked entity and *answer* refers to this masked entity to retrieve. Thus, for each summary to assess, metrics can be derived from the ability of the QA system to retrieve the correct answers from each of the associated triplets.

**F1 score** For each triplet, an F1 score is computed according to the responses retrieved by the considered QA system. This score, commonly used for QA evaluation (Rajpurkar et al., 2016), measures the average overlap between predictions and ground truth answers. For each summary to assess, the is the average of the F1 score computed over each triplet. In the following, we denote this metric as $QA_{fscore}$*(sup)*.

**QA confidence** Complementary to the F1 score, we propose to also consider the confidence of the QA system for its retrieved answer. This corresponds, for each triplet, to the probability of the true answer according to the QA model. Confidence scores are averaged for each summary to assess over its associated triplets. In the following, we denote this metric as $QA_{conf}$*(sup)*.

Besides considering the simple presence of the expected answers in the generated summary, QA-based metrics also account to some extent for readability. They indeed require that the considered QA system, trained on natural language, be able to find the answer in the input to assess, despite the variability of the generated texts.

**Extension to the unsupervised setting** While being a useful complement to ROUGE, the two QA-based metrics described above still need human-generated summaries. In this paper, we investigate and propose extending the previously described QA-based approach in an unsupervised setting.

| | **Readability** | **Relevance** |
|---|---|---|
| **Readability** | 1.0 | 0.77 ** |
| **Relevance** | 0.77 ** | 1.0 |
| ROUGE-1 (sup) | 0.14 * | 0.18 ** |
| ROUGE-2 (sup) | 0.12 * | 0.18 ** |
| ROUGE-L (sup) | 0.13 * | 0.18 ** |
| TextRank (unsup) | 0.14 * | 0.13 ** |
| Novelty (unsup) | -0.13 * | -0.1 * |
| Bert LM (unsup) | 0.21 ** | 0.08 * |
| $QA_{fscore}$ (sup) | 0.14 * | 0.19 ** |
| $QA_{conf}$ (sup) | 0.19 ** | 0.23 ** |
| $QA_{fscore}$ (unsup) | 0.08 | 0.2 ** |
| $QA_{conf}$ (unsup) | 0.33 ** | 0.31 ** |

Table 1: Spearman's $\rho$ for the different metrics w.r.t. Readability and Relevance (*: $p < .05$, **: $p < .005$).

With this aim, we extended the above metrics at *the document level* (*i.e.*, questions and answers are generated from the source article text rather than from the reference summary), dispensing of the need for human-generated reference summaries. Thus, in line with the *APES* approach described above, we propose two unsupervised QA-based metrics, to which we refer to as $QA_{fscore}$*(unsup)* and $QA_{conf}$*(unsup)*. Accounting for both quality and informativeness of a generated summary, those metrics have the appealing property of not requiring reference summaries.

### 2.3 Quantitative Analysis

We exploit human judgments obtained for 3 types of automatically generated summaries by Paulus et al. (2017) on 100 samples of the CNN/Daily Mail summarization dataset (see detail in section 4.1), in terms of readability (how well written the summary is) and relevance (how well does the summary capture the important parts of the article). The summaries are generated by the three different systems proposed in the original work. Those samples have been scored, via Amazon Mechanical Turk, for Readability and Relevance (scores from 1 to 10 for both metrics).

In Table 1, we report Spearman's rank correlations on this data, where we compare summaries rankings obtained according to the assessed metrics. Scores render the ability of the various metrics to reproduce human preferences (in terms

of readability and relevance). First, we observe that readability and relevance are naturally intertwined: intuitively, an unreadable summary will bear very little information, one of the facts that explains the high correlation between *readability* and *relevance*.

From this correlation analysis against human judgments, we observe that, as expected, the Language Model metric captures *readability* better than ROUGE, while falling short on *relevance*.

On the other hand, the results obtained using the proposed QA-based metrics indicate their potential benefits especially under the *unsupervised* setting, with $QA_{conf}$ and $QA_{fscore}$ capturing *readability* and *relevance* better than all the others reported metrics, including ROUGE. We thus conclude that the proposed metrics, which favorably correlate with *readability* and *relevance* under human evaluation, are worth of a deeper experimental investigation: in the following sections we provide a thorough evaluation of their contributions as Reinforcement Learning rewards signals.

### 2.4 Learned Metric

Finally, we also leverage the qualitative data obtained by Paulus et al. (2017) – which compounds to 50 samples evaluated by annotators in terms of *readability* and *relevance* – to *learn* an aggregate metric for evaluation. We use a Ridge regression (with a regularization $\lambda = 1$) to learn to predict the geometric mean of readability and relevance from the metrics defined above. The geometric means was chosen since we want the generated summary to be both readable and relevant.

We randomly sampled 50% of the data to fit the linear model with various subsets of our base metrics. Then, we measured the correlation w.r.t. the expected geometric mean on the remaining 50% data. We performed this procedure 1000 times. Our experiments show that the best performing set of metrics consists of ROUGE-L in conjunction with $\text{QA}_{conf}$ and $\text{QA}_{fscore}$, both computed at an article-level, and hence unsupervised.

This learned metric is thus defined as (with *unsup* versions of QA-based scores):

$$\alpha ROUGE_L + \beta QA_{conf} + \delta QA_{fscore} \quad (1)$$

with $\alpha = 0.8576$, $\beta = 2.274$ and $\delta = 0.6413$. We leverage this learned metric in our RL-based summarization model, as described below.

### 2.5 Implementation details

As QA system we use the BERT "base" pre-trained model (Devlin et al., 2019), finetuned on the SQuAD dataset (Rajpurkar et al., 2016) using the recommended parameters for the task[1]. This differs from the approach adopted by (Eyal et al., 2019) who trained their QA model on CNN-DM (the same data used for the summarization task).

## 3 Summarization Models

Abstractive summarization systems were originally designed as a post-processing of an extractive system – by compressing sentences (Nenkova, 2011). A lot of activity takes place nowadays in designing neural networks sequence to sequence architectures (Sutskever et al., 2014), which allow to consider the problem as a whole rather than a two-step process. Such models reached state-of-the-art performance. To tackle the summarization, which deals with a long text and possibly out-of-vocabulary tokens, See et al. (2017) proposed to leverage an attention over the input (Bahdanau et al., 2014), as well as a copy mechanism (Vinyals et al., 2015).

One problem of sequence-to-sequence models is that they tend to repeat text in the output. To deal with this problem, (See et al., 2017) use a *coverage mechanism*, and Paulus et al. (2017) introduced *Intra-Decoder Attention* with the same goal of avoiding duplicate information within the output sequences.

More recently, the model proposed by See et al. (2017) was further extended (Gehrmann et al., 2018), with the addition of an attention mask during inference: a pre-trained sequence tagger trained to select which input tokens should be copied and used to filter the copy mechanism. Such a filter, called *Bottom-Up Copy Attention*, was shown to help prevent copying from the source text sequences that are too long, hence resulting in more abstractive summaries. On the CNN/Daily Mail dataset, (Gehrmann et al., 2018) found this two-step process to yield significant improvements in terms of ROUGE – resulting in the current state-of-the-art system. We base our experiments on this model.

The differentiable loss function commonly used for training summarization models, negative log-likelihood, has several known limitations. Among

[1] https://github.com/huggingface/pytorch-transformers.

those, exposure bias and failure to cope with the large number of potentially valid summaries.

To overcome this, approaches based on reinforcement learning have recently been proposed, allowing the models to learn via reward signals. Ranzato et al. (2015) used the REINFORCE algorithm (Williams, 1992) to train RNNs for several generation tasks, showing improvements over previous supervised approaches. Narayan et al. (2018) used such an approach in an extractive summarization setting, learning to select the most relevant sentences within the input text in order to construct its summary. (Paulus et al., 2017) combined supervised and reinforcement learning, demonstrating improvements over competing approaches both in terms of ROUGE and on human evaluation. However, the main limit of these works is that they rely on standard summarization metrics which are known to be biased.

Finally, closer to our work, Arumae and Liu (2019) proposed to use question-answering rewards to learn an *extractive* summarization model in a reinforcement learning setup. Compared to what we propose, their system is extractive, and relies on hand-written summaries.

### 3.1 Mixed Training Objectives

In our experiments, we follow the reinforcement learning scheme described below. The main difference with previous works is our reward function, which was based on our study of metrics (section 2). We consider a mixed loss $L_{ml+rl}$ combining supervised and reinforcement learning schemes:

$$L_{ml+rl} = \gamma L_{rl} + (1-\gamma) L_{ml} \tag{2}$$

where we define the reinforcement loss $L_{rl}$ and the maximum likelihood $L_{ml}$ in the following paragraphs.

**Maximum Likelihood** Under a supervised training setup, the teacher forcing algorithm (Williams and Zipser, 1989) can be applied, and corresponds to maximizing the likelihood (ML) or equivalently to minimizing the negative log likelihood (NLL) loss defined as:

$$L_{ml} = -\sum_{t=0}^{m} \log p(y_t^* | y_0^*, ..., y_{t-1}^*, X) \tag{3}$$

where $X = [x_1, ..., x_n]$ is the input text of $n$ tokens and $Y^* = [y_1^*, ..., y_m^*]$ is the corresponding reference summary of $m$ tokens.

**Policy Learning** Several RL-based summarization (Kryściński et al., 2018; Li et al., 2018; Pasunuru and Bansal, 2018; Paulus et al., 2017) apply the self-critical policy gradient training algorithm (Rennie et al., 2017). Following (Paulus et al., 2017) we use REINFORCE algorithm, using as the baseline a greedy decoding algorithm according to the conditional distribution $p(y|X)$, giving rise to a sequence $\widehat{Y}$. The model is sampled using its Markov property, that is, one token at a time, giving rise to the sequence $Y^s$.

Following the standard RL actor-critic scheme, with $r(Y)$ the reward function for an output sequence Y, the loss to be *minimized* is then defined as:

$$L_{rl} = (r(\widehat{Y}) - r(Y^s)) \sum_{t=0}^{m} \log p(y_t^s | y_0^s, ..., y_{t-1}^s, X) \tag{4}$$

As ROUGE is the most widely used evaluation metric, Paulus et al. (2017) used ROUGE-L as the reward $r$ for the $L_{rl}$ function and tested the following three different setups:

- *ML*: the model trained with $L_{ml}$ ($\gamma = 0$);
- *RL*: the model trained with $L_{rl}$ ($\gamma = 1$);
- *ML+RL*: the model trained with $L_{ml+rl}$ ($\gamma = 0.9984$).

The human evaluation conducted on the three models shows that *RL* performs worse than *ML*, and *ML+RL* performs best for both *readability* and *relevance*. The authors also conclude that "despite their common use for evaluation, ROUGE scores have their shortcomings and should not be the only metric to optimize on summarization model for long sequences", which is translated in the very high optimal $\gamma$. We show that using a more sensible metric to optimize leads to a better model, and to a lower $\gamma$.

## 4 Experiments

In our experiments, we evaluate the effect of substituting the ROUGE reward in the reinforcement-learning model of (Paulus et al., 2017) by our proposed metric (section 2). We, moreover, study the effect of using metrics that do not necessitate human-generated summaries.

### 4.1 Data Used

Task-specific corpora for building and evaluating summarization models associate a human-generated reference summary with each text provided. We resort to the CNN/Daily Mail (CNN-DM) dataset (Hermann et al., 2015; Nallapati et al., 2016) for our experiments. It includes 287,113 article/summary pairs for training, 13,368 for validation, and 11,490 for testing. The summary corresponding to each article consists of several bullet points displayed on the respective news outlet webpage. In average, summaries contain 66 tokens ($\sigma = 26$) and 4.9 bullet points. Consistently with See et al. (2017) and Gehrmann et al. (2018), we use the non-anonymized version of the dataset, the same training/validation splits, and perform truncation of source documents and summaries to 400 and 100 tokens, respectively.

To assess the possible benefits of reinforcing over the proposed QG-based metric, which does not require human-generated reference summaries, we employ TL;DR[2], a large-scale dataset for automatic summarization built on social media data, compounding to 4 Million training pairs (Völske et al., 2017). Both CNN-DM and TL;DR datasets are in English.

### 4.2 Models

For all our experiments, we build on top of the publicly available OpenNMT implementation[3], consistently with Gehrmann et al. (2018) to which we refer to as a baseline. The encoder is composed of a one-layer bi-LSTM with 512 hidden states, and 512 hidden states for the one-layer decoder. The embedding size is set at 128. The model is trained with Adagrad, with an initial learning rate of 0.15, and an initial accumulator value of 0.1. We continue training until convergence; when the validation perplexity does not decrease after an epoch, the learning rate is halved. We use gradient-clipping with a maximum norm of 2.

Gehrmann et al. (2018) showed that increasing the number of hidden states leads to slight improvements in performance, at the cost of increased training time; thus, as reinforcement learning is computationally expensive, we build on top of the smallest model – nonetheless, we include the largest model by Gehrmann et al. (2018) in our discussion of results.

All the experimented reinforcement approaches use the mixed training objectives defined in equation 2, with the ML part corresponding to the previously described baseline model pretrained on the CNN-DM dataset. Compared models differ on the considered reward signals. They also differ on their use of additional unsupervised data, either *In-Domain* or *Out-of-Domain*, as discussed below.

#### 4.2.1 Reward Signals

The three reward signals used throughout our experiments, are detailed below:

1. **ROUGE**: We use only ROUGE-L as reward signal within the baseline architecture, consistently with Paulus et al. (2017);

2. **QA**$_{learned}$: Conversely, we compute the reward by applying the learned coefficients to the three components of the learned metric, as obtained in Section 2.4.

3. **QA**$_{equally}$: We apply the mixed training objective function, using as a reward the three metric components of the learned metric (ROUGE-L, QA$_{conf}$, and QA$_{fscore}$) equally weighted: this corresponds to setting a value of 1 for $\alpha$, $\beta$ and $\delta$ in Eq. 1. This allows to see to which extent learning is sensitive to fitting human assessments.

For (2) and (3), we set $\gamma$ (Eq. 2) to 0.5[4]. This shows that, compared to (Paulus et al., 2017), we do not need to use NLL to avoid the model from generating unreadable summaries.

#### 4.2.2 In-Domain vs Out-of-Domain

Finally, we experiment with the proposed QA$_{conf}$ and QA$_{fscore}$ metrics in an unsupervised fashion, as they can be computed at article level – *i.e.* without accessing the reference human-generated summaries. We investigate the potential benefits of using this approach both *in-domain* and *out-of-domain*: for the former, we resort to the test set of the CNN-Daily Mail (CNN-DM) dataset; for the latter, we leverage the TL;DR corpus.

As the CNN-Daily Mail is built from mainstream news articles, and the TL;DR data comes from social media sources, we consider the latter

[2] https://tldr.webis.de

[3] http://opennmt.net/OpenNMT-py/Summarization.html

[4] We have run experiments with $\gamma = 0.5$, and $\gamma = 0.9984$ as Paulus et al. (2017); we report here the best performance which was obtained with the former.

as out-of-domain in comparison. From the latter, which includes circa 4 million samples, we randomly draw sample subsets of size comparable with CNN-DM for training, validation and testing splits.

Due to computational costs, we restrict these experiments to the model trained under reinforcement using the $\text{QA}_{learned}$ metric. Under this setup, the model has access at training time both to:

- *supervised* samples for which a reference summary is given (and thus all metrics, including ROUGE and NLL, can be computed as a training objective), coming from the training set of CNN-Daily Mail corpus ;
- *unsupervised* samples, for which no reference is available, thus allowing to only compute $\text{QA}_{conf}$(unsup) and $\text{QA}_{fscore}$(unsup). Three unsupervised settings are considered in the following:

  *TL;DR*, corresponding to the *out-of-domain* setting where we use articles from the *TL;DR* dataset;

  *CNN-DM (VAL)*, corresponding to an *in-domain* setting where we use texts from the validation set from the CNN/Daily Mail dataset;

  and, *CNN-DM (TEST)* for an *in-domain* setting where we use the articles from the test set (thus containing texts used for evaluation purposes).

While all the data is from the CNN-DM train dataset in the *supervised* setups, for the *unsupervised* setups, we set the proportion of unsupervised data to 50% (either CNN-DM VAL, CNN-DM TEST for *in-domain* or TL;DR for *out-of-domain* data). Thus, for 50% of the data, the model has access only to the $\text{QA}_{conf}$ and $\text{QA}_{fscore}$ reward signals, since the ROUGE-L reward can only be computed on *supervised* batches.

Therefore, for all the unsupervised setups, in order to keep consistency in the reward signal throughout the training, we multiply by a factor of 2 the weight associated with ROUGE-L when this reward is computable, and set it to 0 otherwise.

### 4.3 Results

In Table 2, we report the results obtained from our experiments in comparison with previously proposed approaches. We observe that, unsurprisingly, reinforcing on ROUGE-L allows to obtain significant improvements over the state-of-the-art, in terms of ROUGE but at the cost of lower QA-based metrics. Conversely, reinforcing on the proposed metric improves consistently all its components (ROUGE-L, $\text{QA}_{conf}$ and $\text{QA}_{fscore}$).

However, increasing the reward does not necessarily correlate with better summaries. The human inspection as reported by (Paulus et al., 2017) shows that the generated summaries reinforced on ROUGE-L are consistently on the low end in terms of readability and relevance.

A closer inspection of the generated summaries revealed that the sequences generated by this model seem to qualitatively degrade as the number of produced tokens grows: they often start with a reasonable sub-sequence, but quickly diverge towards meaningless outputs. This can be explained by the aforementioned drawbacks of ROUGE, which are likely amplified when used both as evaluation and reward: the system might be optimizing for ROUGE, at the price of losing the information captured with the NLL loss by its language model.

We hence conducted a human evaluation for the different setups, reported in Table 3, assessing their outputs for *readability* and *relevance* in line with Paulus et al. (2017). We randomly sampled 50 articles from the CNN-DM test set; since the learned metric used in our experiments is derived from the subset manually evaluated in Paulus et al. (2017) we ensured that there was no overlap with it. For each of those 50 articles, three English speakers evaluated the summaries generated by the 7 different systems reported in Table 2.

We observe that reinforcing using the proposed metric – which includes QA based metrics, leads to comparable performance in terms of ROUGE w.r.t. state-of-the-art approaches, while clear benefits emerge from the results of the human evaluation: a significant improvement in terms of relevance, particularly when leveraging *in-domain* data in an *unsupervised* setup. Not surprisingly, we observe an improvement for our model when reinforced through the learned metric compared to the one equally weighted. The slightly lower relevance scores observed for the $\text{QA}_{learned}$ w.r.t. $\text{QA}_{equally}$ are consistent with the lower ROUGE-L and $\text{QA}_{fscore}$ reported in Table 2. This is explained by the lower coefficients for ROUGE-L

| | R-1 | R-2 | R-L | QA$_{fscore}$ | QA$_{conf}$ |
|---|---|---|---|---|---|
| See et al. (2017) | 39.53 | 17.28 | 36.38 | - | - |
| Gehrmann et al. (2018) | 41.22 | **18.68** | 38.34 | - | - |
| ML+RL Paulus et al. (2017) | 39.87 | 15.82 | 36.90 | - | - |
| RL Paulus et al. (2017) | 41.16 | 15.75 | 39.08 | - | - |
| Pasunuru and Bansal (2018) | 40.43 | 18.00 | 37.10 | - | - |
| Chen and Bansal (2018) | 40.88 | 17.80 | 38.54 | - | - |
| baseline | 42.24 | 17.78 | 37.44 | 14.91 | 40.12 |
| + ROUGE | **45.62** | 16.30 | **41.60** | 13.64 | 37.90 |
| + QA$_{equally}$ | 43.36 | 18.06 | 38.33 | 16.06 | 41.01 |
| + QA$_{learned}$ | 42.71 | 17.81 | 37.94 | 15.19 | 41.39 |
| + QA$_{learned}$ + TL;DR | 42.75 | 17.57 | 37.88 | 15.75 | 41.54 |
| + QA$_{learned}$ + CNN-DM (VAL) | 43.00 | 17.66 | 38.23 | 16.16 | 41.75 |
| + QA$_{learned}$ + CNN-DM (TEST) | 42.74 | 17.25 | 37.96 | **16.17** | **42.14** |

Table 2: Comparison with previous works. On top, we report the results obtained by Gehrmann et al. (2018) using their largest architecture, as well as those by See et al. (2017). Next, we report results recently obtained by reinforcement learning approaches. Finally, we indicate the scores obtained by our baseline – the "small" model by Gehrmann et al. (2018) – and the six reinforced models we build on top of it.

and QA$_{fscore}$ (see 2.4), and the relatively stronger correlation of those two metrics with *relevance* (see Table 1).

Consistently with the figures reported in Table 2, the human evaluation results – reported in Tables 3 and 4 – confirm the progressive improvements of our different proposed models when using unsupervised data closer to the test set documents:

- adding *unsupervised* data from the out-of-domain TL;DR brings a slight improvement using QA$_{learned}$;
- when it comes to the same domain (*i.e.* CNN-DM validation) the improvements increase;
- finally, when unsupervised samples come from the same set as those used for testing, we observe even better results.

These results show that using the proposed QA-based metrics, that do not depend on reference summaries, allows to leverage raw text data; and, that fine-tuning (without supervision) on the documents to be summarized is beneficial.

To elaborate further, we notice that applying the learned coefficients for 1 to the results obtained by models reinforced on QA$_{\text{learned}}$ and QA$_{\text{equally}}$, see Table 2, we obtain very similar scores (namely, 136.43 for QA$_{\text{equally}}$ and 136.4 for QA$_{\text{learned}}$). However, the qualitative analysis reported in Tables 3 and 4 shows that while they perform similarly in terms of *relevance*, a significantly lower score for *readability* is obtained using QA$_{\text{equally}}$. This can be explained by the stronger weight of ROUGE_L for this setup, a fact which might lead to a degradation of the quality of the output consistently with the observations reported in (Paulus et al., 2017) as well as in our ROUGE experiment.

Another observation from Tables 3 and 4 is that while QA$_{\text{learned}}$ performs significantly better in term of *readability* than QA$_{\text{learned}}$ + CNN-DM (VAL), the opposite holds for *relevance*. This could be explained by the setup difference during training: as detailed in section 4.2.2, for unsupervised setups (*i.e.* QA$_{\text{learned}}$ + CNN-DM (VAL)) only the QA-based metrics are computed for the portion of data for which no reference is available. While testing (TEST) and validation (VAL) splits come the same dataset (CNN-DM), we observe that using the samples from TEST in an unsupervised fashion allows for maintaining comparably high *relevance* compared to QA$_{\text{learned}}$ + CNN-DM (VAL), while also obtaining similar *readability* to QA$_{\text{learned}}$. This shows the possible benefits that can be obtained by exposing the model to the evaluation data in unsupervised setups. To further study our unsupervised metrics, we performed additional experiments on the TL;DR corpus. We observed more than one absolute point of improvement w.r.t CNN-DM TEST in terms of ROUGE-L, QA$_{fscore}$ (unsup) and QA$_{conf}$ (unsup).

This indicates that the proposed unsupervised

| | Readability | Relevance |
|---|---|---|
| human reference | 7.27* | 7.4** |
| baseline | 7.07 | 5.82 |
| + ROUGE | 2.14** | 5.48** |
| + QA$_{equally}$ | 5.94** | 6.34** |
| + QA$_{learned}$ | 6.96 | 6.21** |
| + QA$_{learned}$ + TL;DR | 6.60* | 6.26** |
| + QA$_{learned}$ + CNN-DM (VAL) | 6.40* | 6.75** |
| + QA$_{learned}$ + CNN-DM (TEST) | 6.89 | 6.80** |

Table 3: Human assessment: two-tailed t-test results are reported for each model compared to the baseline ($* : p < .01$, $** : p < .001$).

| | human reference | baseline | + ROUGE | + QA$_{equally}$ | + QA$_{learned}$ | + QA$_{learned}$ + TL;DR | + QA$_{learned}$ + CNN-DM (VAL) | + QA$_{learned}$ + CNN-DM (TEST) |
|---|---|---|---|---|---|---|---|---|
| baseline | * / ** | - | | | | | | |
| + ROUGE | ** / ** | ** / ** | - | | | | | |
| + QA$_{equally}$ | ** / ** | ** / ** | ** / ** | - | | | | |
| + QA$_{learned}$ | ** / ** | - / ** | ** / ** | ** / - | - | | | |
| + QA$_{learned}$ + TL;DR | ** / ** | * / ** | ** / ** | ** / - | * / - | - | | |
| + QA$_{learned}$ + CNN-DM (VAL) | ** / ** | * / ** | ** / ** | ** / * | ** / ** | - / * | - | |
| + QA$_{learned}$ + CNN-DM (TEST) | ** / ** | - / ** | ** / ** | ** / * | - / ** | - / * | * / - | - |

Table 4: Human assessment: two-tailed t-test results are reported for each model pair for Readability / Relevance ($* : p < .01$, $** : p < .001$).

metrics allow the model to better transfer to new domains such as TL;DR. These results pave the way for leveraging large numbers of texts, in a self-supervised manner, to train automatic summarization models.

## 5 Conclusions

We have presented the analysis of novel QA-based metrics[5], and have shown promising results when using them as a reward in a RL setup. Crucially, those metrics do not require a human reference, as they can be computed from the text to be summarized.

From our experiments this proves particularly beneficial, allowing to leverage both *in-domain* and *out-of-domain* unlabeled data.

The promising results obtained indicate a path towards partially self-supervised training of summarization models, and suggest that progress in automated question generation can bring benefits for automatic summarization.

Finally, to our knowledge, this paper is the first to compare two architectures with the same reinforcement setup on the same data: the one proposed by See et al. (2017) and extended by Gehrmann et al. (2018), versus the one by Paulus et al. (2017). In terms of ROUGE, we observe better results than those reported by Paulus et al. (2017) – see Table 2 – indicating a possible edge for the architecture proposed by See et al. (2017).

[5] A python package will be made available at https://www.github.com/recitalAI/summa-qa.

## References


Kristjan Arumae and Fei Liu. 2019. Guiding extractive summarization with question-answering rewards. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2566–2577, Minneapolis, Minnesota. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Ping Chen, Fei Wu, Tong Wang, and Wei Ding. 2018. A semantic qa-based approach for text summarization evaluation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 675–686, Melbourne, Australia. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Matan Eyal, Tal Baumel, and Michael Elhadad. 2019. Question answering as an automatic evaluation metric for news article summarization. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in neural information processing systems*, pages 1693–1701.

Wojciech Kryściński, Romain Paulus, Caiming Xiong, and Richard Socher. 2018. Improving abstraction in text summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1808–1817, Brussels, Belgium. Association for Computational Linguistics.

Piji Li, Lidong Bing, and Wai Lam. 2018. Actor-critic based training framework for abstractive summarization. *arXiv preprint arXiv:1803.11070*.

C-Y Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Proc. of Workshop on Text Summarization Branches Out, Post Conference Workshop of ACL 2004*.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2122–2132.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Ranking sentences for extractive summarization with reinforcement learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1747–1759, New Orleans, Louisiana. Association for Computational Linguistics.

Ani Nenkova. 2011. *Automatic Summarization*. Foundations and Trends in Information Retrieval. Now.

Jun-Ping Ng and Viktoria Abrecht. 2015. Better summarization evaluation with word embeddings for rouge. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1925–1930.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Ramakanth Pasunuru and Mohit Bansal. 2018. Multi-reward reinforced summarization with saliency and entailment. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 646–653, New Orleans, Louisiana. Association for Computational Linguistics.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7008–7024.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083.

Elaheh ShafieiBavani, Mohammad Ebrahimi, Raymond Wong, and Fang Chen. 2018. Summarization evaluation in the absence of human model summaries using the compositionality of word embeddings. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 905–914, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Michael Völske, Martin Potthast, Shahbaz Syed, and Benno Stein. 2017. Tl;dr: Mining reddit to learn automatic summarization. In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 59–63.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280.

Harris Wu, Dragomir R Radev, and Weiguo Fan. 2002. Towards answer-focused summarization. In *Proceedings of the 1st International Conference on Information Technology and Applications*.

# Long and Diverse Text Generation with Planning-based Hierarchical Variational Model

**Zhihong Shao[1], Minlie Huang[1]*, Jiangtao Wen[1], Wenfei Xu[2], Xiaoyan Zhu[1]**
[1] Institute for Artificial Intelligence, State Key Lab of Intelligent Technology and Systems
[1] Beijing National Research Center for Information Science and Technology
[1] Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
[2] Baozun, Shanghai, China
szh19@mails.tsinghua.edu.cn, aihuang@tsinghua.edu.cn
jtwen@tsinghua.edu.cn, xuwenfeilittle@gmail.com
zxy-dcs@tsinghua.edu.cn

## Abstract

Existing neural methods for data-to-text generation are still struggling to produce long and diverse texts: they are insufficient to model input data dynamically during generation, to capture inter-sentence coherence, or to generate diversified expressions. To address these issues, we propose a Planning-based Hierarchical Variational Model (PHVM). Our model first plans a sequence of groups (each group is a subset of input items to be covered by a sentence) and then realizes each sentence conditioned on the planning result and the previously generated context, thereby decomposing long text generation into dependent sentence generation sub-tasks. To capture expression diversity, we devise a hierarchical latent structure where a global planning latent variable models the diversity of reasonable planning and a sequence of local latent variables controls sentence realization. Experiments show that our model outperforms state-of-the-art baselines in long and diverse text generation.

## 1 Introduction

Data-to-text generation is to generate natural language texts from structured data (Gatt and Krahmer, 2018), which has a wide range of applications (for weather forecast, game report, product description, advertising document, etc.). Most neural methods focus on devising encoding scheme and attention mechanism, namely, (1) exploiting input structure to learn better representation of input data (Lebret et al., 2016; Liu et al., 2018), and (2) devising attention mechanisms to better employ input data (Mei et al., 2016; Liu et al., 2018; Nema et al., 2018) or to dynamically trace which part of input has been covered in generation (Kiddon et al., 2016). These models are able to produce fluent and coherent short texts in some applications.

However, to generate long and diverse texts such as product descriptions, existing methods are still unable to capture the complex semantic structures and diversified surface forms of long texts. **First**, existing methods are not good at modeling input data dynamically during generation. Some neural methods (Kiddon et al., 2016; Feng et al., 2018) propose to record the accumulated attention devoted to each input item. However, these records may accumulate errors in representing the state of already generated prefix, thus leading to wrong new attention weights. **Second**, inter-sentence coherence in long text generation is not well captured (Wiseman et al., 2017) due to the lack of high-level planning. Recent studies propose to model planning but still have much space for improvement. For instance, in (Puduppully et al., 2019) and (Sha et al., 2018), planning is merely designed for ordering input items, which is limited to aligning input data with the text to be generated. **Third**, most methods fail to generate diversified expressions. Existing data-to-text methods inject variations at the conditional output distribution, which is proved to capture only low-level variations of expressions (Serban et al., 2017).

To address the above issues, we propose a novel Planning-based Hierarchical Variational Model (PHVM). To better model input data and alleviate the inter-sentence incoherence problem, we design a novel planning mechanism and adopt a compatible hierarchical generation process, which mimics the process of human writing. Generally speaking, to write a long text, a human writer first arranges contents and discourse structure (i.e., *high-level planning*) and then realizes the surface form of each individual part (*low-level realization*). Motivated by this, our proposed model first performs

*Corresponding author: Minlie Huang.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3257–3268, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

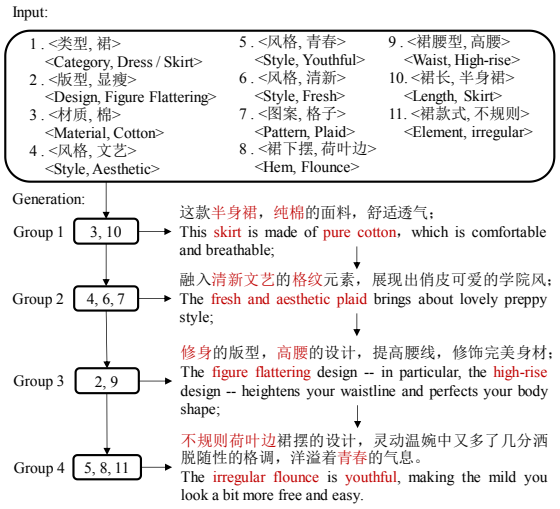


Figure 1: Generation process of PHVM. After encoding a list of input attribute-value pairs, PHVM first conducts planning by generating a sequence of groups, each of which is a subset of input items. Each sentence is then realized conditioned on the corresponding group and its previous generated sentences.

planning by segmenting input data into a sequence of groups, and then generates a sentence conditioned on the corresponding group and preceding generated sentences. In this way, we decompose long text generation into a sequence of dependent sentence generation sub-tasks where each sub-task depends specifically on an individual group and the previous context. By this means, the input data can be well modeled and inter-sentence coherence can be captured. Figure 1 depicts the process.

To deal with expression diversity, this model also enables us to inject variations at both high-level planning and low-level realization with a hierarchical latent structure. At high level, we introduce a global planning latent variable to model the diversity of reasonable planning. At low level, we introduce local latent variables for sentence realization. Since our model is based on Conditional Variational Auto-Encoder (CVAE) (Sohn et al., 2015), expression diversity can be captured by the global and local latent variables.

We evaluate the model on a new advertising text[1] generation task which requires the system to generate a long and diverse advertising text that covers a given set of attribute-value pairs describing a product (see Figure 1). We also evaluate our model on the recipe text generation task from (Kiddon et al., 2016) which requires the system to correctly use the given ingredients and maintain coherence among cooking steps. Experiments on advertising text generation show that our model outperforms state-of-the-art baselines in automatic and manual evaluation. Our model also generalizes well to long recipe text generation and outperforms the baselines. Our contributions are twofold:

- We design a novel Planning-based Hierarchical Variational Model (PHVM) which integrates planning into a hierarchical latent structure. Experiments show its effectiveness in coverage, coherence, and diversity.
- We propose a novel planning mechanism which segments the input data into a sequence of groups, thereby decomposing long text generation into dependent sentence generation sub-tasks. Thus, input data can be better modeled and inter-sentence coherence can be better captured. To capture expression diversity, we devise a hierarchical latent structure which injects variations at both high-level planning and low-level realization.

## 2 Related Work

Traditional methods (Reiter and Dale, 1997; Stent et al., 2004) for data-to-text generation consist of three components: content planning, sentence planning, and surface realization. Content planning and sentence planning are responsible for what to say and how to say respectively; they are typically based on hand-crafted (Kukich, 1983; Dalianis and Hovy, 1993; Hovy, 1993) or automatically-learnt rules (Duboue and McKeown, 2003). Surface realization generates natural language by carrying out the plan, which is template-based (McRoy et al., 2003; van Deemter et al., 2005) or grammar-based (Bateman, 1997; Espinosa et al., 2008). As these models are shallow and the two stages (planning and realization) often function separately, traditional methods are unable to capture rich variations of texts.

Recently, neural methods have become the mainstream models for data-to-text generation due to their strong ability of representation learning and scalability. These methods perform well in generating weather forecasts (Mei et al., 2016) or very short biographies (Lebret et al., 2016; Liu

[1] An advertising text describes a product with attractive wording. The goal of writing such texts is to advertise a product and attract users to buy it.

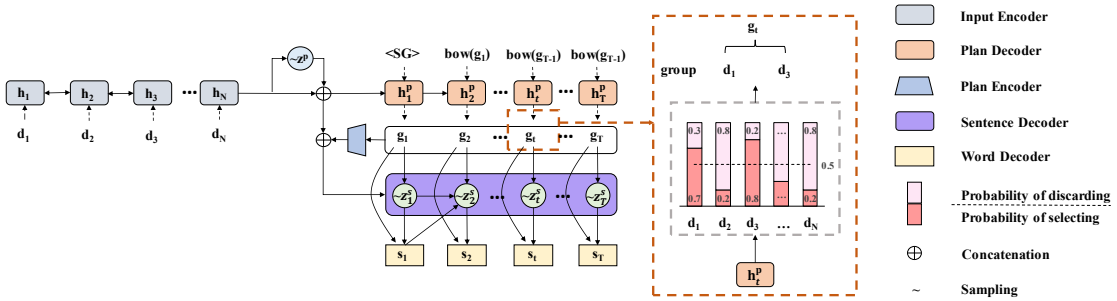


Figure 2: Architecture of PHVM. The model controls planning with a global latent variable $z^p$. The plan decoder conducts planning by generating a sequence of groups $g = g_1 g_2 ... g_T$ where $g_t$ is a subset of input items and specifies the content of sentence $s_t$ to be generated. The sentence decoder controls the realization of $s_t$ with a local latent variable $z_t^s$; dependencies among $z_t^s$ are explicitly modeled to better capture inter-sentence coherence.

et al., 2018; Sha et al., 2018; Nema et al., 2018) using well-designed data encoder and attention mechanisms. However, as demonstrated in Wiseman et al. (2017) (a game report generation task), existing neural methods are still problematic for long text generation: they often generate incoherent texts. In fact, these methods also lack the ability to model diversity of expressions.

As for long text generation, recent studies tackle the incoherence problem from different perspectives. To keep the decoder aware of the crucial information in the already generated prefix, Shao et al. (2017) appended the generated prefix to the encoder, and Guo et al. (2018) leaked the extracted features of the generated prefix from the discriminator to the generator in a Generative Adversarial Nets (Goodfellow et al., 2014). To model dependencies among sentences, Li et al. (2015) utilized a hierarchical recurrent neural network (RNN) decoder. Konstas and Lapata (2013) proposed to plan content organization with grammar rules while Puduppully et al. (2019) planned by reordering input data. Most recently, Moryossef et al. (2019) proposed to select plans from all possible ones, which is infeasible for large inputs.

As for diverse text generation, existing methods can be divided into three categories: enriching conditions (Xing et al., 2017), post-processing with beam search and rerank (Li et al., 2016), and designing effective models (Xu et al., 2018). Some text-to-text generation models (Serban et al., 2017; Zhao et al., 2017) inject high-level variations with latent variables. Variational Hierarchical Conversation RNN (VHCR) (Park et al., 2018) is a most similar model to ours, which also adopts a hierarchical latent structure. Our method differs from VHCR in two aspects: (1) VHCR has no planning mechanism, and the global latent variable is mainly designed to address the KL collapse problem, while our global latent variable captures the diversity of reasonable planning; (2) VHCR injects distinct local latent variables without direct dependencies, while our method explicitly models the dependencies among local latent variables to better capture inter-sentence connections. Shen et al. (2019) proposed ml-VAE-D with multi-level latent variables. However, the latent structure of ml-VAE-D consists of two global latent variables: the top-level latent variable is introduced to learn a more flexible prior of the bottom-level latent variable which is then used to decode a whole paragraph. By contrast, our hierarchical latent structure is tailored to our planning mechanism: the top level latent variable controls planning results and a sequence of local latent variables is introduced to obtain fine-grained control of sentence generation sub-tasks.

We evaluated our model on a new advertising text generation task which is to generate a long and diverse text that covers all given specifications about a product. Different from our task, the advertising text generation task in (Chen et al., 2019) is to generate personalized product description based on product title, product aspect (e.g., "appearance"), and user category.

## 3 Task Definition

Given input data $x = \{d_1, d_2, ..., d_N\}$ where each $d_i$ can be an attribute-value pair or a keyword, our task is to generate a long and diverse text $y = s_1 s_2 ... s_T$ ($s_t$ is the $t^{th}$ sentence) that covers $x$ as much as possible. For the advertising

text generation task, $x$ consists of specifications about a product where each $d_i$ is an attribute-value pair $< a_i, v_i >$. For the recipe text generation task, $x$ is an ingredient list where each $d_i$ is an ingredient. Since the recipe title $r$ is also used for generation, we abuse the symbol $x$ to represent $< \{d_1, d_2, ..., d_N\}, r >$ for simplification.

## 4 Approach

### 4.1 Overview

Figure 2 shows the architecture of PHVM. PHVM first samples a global planning latent variable $z^p$ based on the encoded input data; $z^p$ serves as a condition variable in both planning and hierarchical generation process. The plan decoder takes $z^p$ as initial input. At time step $t$, it decodes a group $g_t$ which is a subset of input items ($d_i$) and specifies the content of the $t^{th}$ sentence $s_t$. When the plan decoder finishes planning, the hierarchical generation process starts, which involves the high-level sentence decoder and the low-level word decoder. The sentence decoder models inter-sentence coherence in semantic space by computing a sentence representation $h_t^s$ and sampling a local latent variable $z_t^s$ for each group. $h_t^s$ and $z_t^s$, along with $g_t$, guide the word decoder to realize the corresponding sentence $s_t$.

The planning process decomposes the long text generation task into a sequence of dependent sentence generation sub-tasks, thus facilitating the hierarchical generation process. With the hierarchical latent structure, PHVM is able to capture multi-level variations of texts.

### 4.2 Input Encoding

We first embed each input item $d_i$ into vector $\boldsymbol{e}(d_i)$. The recipe title $r$ is also embedded as $\boldsymbol{e}(r)$. We then encode $x$[2] with a bidirectional Gated Recurrent Unit (GRU) (Cho et al., 2014). For advertising text generation, $x$ is represented as the concatenation of the last hidden states of the forward and backward GRU $\boldsymbol{enc}(x) = [\overrightarrow{h_N}; \overleftarrow{h_1}]$; for recipe text generation, $\boldsymbol{enc}(x) = [\overrightarrow{h_N}; \overleftarrow{h_1}; \boldsymbol{e}(r)]$. $h_i = [\overrightarrow{h_i}; \overleftarrow{h_i}]$ is the context-aware representation of $d_i$. Note that input encoder is not necessarily an RNN; other neural encoders or even other encoding schemes are also feasible, such as multi-layer perceptron (MLP) and bag of words.

[2]For advertising text generation, $x$ is first ordered by attributes so that general attributes are ahead of specific ones; for recipe text generation, we retain the order in the dataset

### 4.3 Planning Process

The planning process generates a subset of input items to be covered for each sentence, thus decomposing long text generation into easier dependent sentence generation sub-tasks. Due to the flexibility of language, there may exist more than one reasonable text that covers the same input but in different order. To capture such variety, we model the diversity of reasonable planning with a global planning latent variable $z^p$. Different samples of $z^p$ may lead to different planning results which control the order of content. This process can be formulated as follows:

$$g = argmax_g P(g|x, z^p) \quad (1)$$

where $g = g_1 g_2 ... g_T$ is a sequence of groups, and each group $g_t$ is a subset of input items which is a main condition when realizing the sentence $s_t$.

The global latent variable $z^p$ is assumed to follow the isotropic Gaussian distribution, and is sampled from its prior distribution $p_\theta(z^p|x) = \mathcal{N}(\mu^p, \sigma^{p2}\mathbf{I})$ during inference and from its approximate posterior distribution $q_{\theta'}(z^p|x, y) = \mathcal{N}(\mu^{p'}, \sigma^{p'2}\mathbf{I})$ during training:

$$[\mu^p; \log \sigma^{p2}] = \boldsymbol{MLP}_\theta(x) \quad (2)$$

$$[\mu^{p'}; \log \sigma^{p'2}] = \boldsymbol{MLP}_{\theta'}(x, y) \quad (3)$$

We solve Eq. 1 greedily by computing $g_t = argmax_{g_t} P(g_t|g_{<t}, x, z^p)$ with the plan decoder (a GRU). Specifically, at time step $t$, the plan decoder makes a binary prediction for each input item by estimating $P(d_i \in g_t|g_{<t}, x, z^p)$:

$$P(d_i \in g_t) = \boldsymbol{\sigma}(v_p^T \boldsymbol{tanh}(W_p[h_i; h_t^p] + b_p)) \quad (4)$$

where $\boldsymbol{\sigma}$ denotes the sigmoid function, $h_i$ is the vector of input item $d_i$, and $h_t^p$ is the hidden state of the plan decoder. Each group is therefore formed as $g_t = \{d_i|P(d_i \in g_t) > 0.5\}$ (If this is empty, we set $g_t$ as $\{argmax_{d_i} P(d_i \in g_t)\}$.).

We feed $bow(g_t)$ (the average pooling of $\{h_i|d_i \in g_t\}$) to the plan decoder at the next time step, so that $h_{t+1}^p$ is aware of what data has been selected and what has not. The planning process proceeds until the probability of stopping at the next time step is over 0.5:

$$P_t^{stop} = \boldsymbol{\sigma}(W_c h_t^p + b_c) \quad (5)$$

The hidden state is initialized with $\boldsymbol{enc}(x)$ and $z^p$. The plan decoder is trained with full supervision, which is applicable to those tasks where

reference plans are available or can be approximated. For both tasks we evaluate in this paper, we approximate the reference plans by recognizing the subset of input items covered by each sentence with string match heuristics. The loss function at time step $t$ is given by:

$$\begin{aligned} -\log P(g_t = \widetilde{g}_t | \widetilde{g_{<t}}, x, z^p) \\ = -\sum_{d_i \in \widetilde{g}_t} \log P(d_i \in g_t) \\ -\sum_{d_i \notin \widetilde{g}_t} \log(1 - P(d_i \in g_t)) \end{aligned} \quad (6)$$

where $\widetilde{g}_t$ is the reference group. As a result, $z^p$ is forced to capture features of reasonable planning.

### 4.4 Hierarchical Generation Process

The generation process produces a long text $y = s_1 s_2 ... s_T$ in alignment with the planning result $g = g_1 g_2 ... g_T$, which is formulated as follows:

$$c = \{x, z^p\} \quad (7)$$

$$y = argmax_y P(y|g, c) \quad (8)$$

We perform sentence-by-sentence generation and solve Eq. 8 greedily by computing $s_t = argmax_{s_t} P(s_t|s_{<t}, g, c)$. $s_t$ focuses more on $g_t$ than on the entire plan $g$. The generation process is conducted hierarchically, which consists of sentence-level generation and word-level generation. Sentence-level generation models inter-sentence dependencies at high level, and interactively controls word-level generation which conducts low-level sentence realization.

**Sentence-level Generation** The sentence decoder (a GRU) performs sentence-level generation; for each sentence $s_t$ to be generated, it produces a sentence representation $h^s_t$ and introduces a local latent variable $z^s_t$ to control sentence realization.

The latent variable $z^s_t$ is assumed to follow the isotropic Gaussian distribution. At time step $t$, the sentence decoder samples $z^s_t$ from the prior distribution $p_\phi(z^s_t|s_{<t}, g, c) = \mathcal{N}(\mu^s_t, \sigma^{s2}_t \mathbf{I})$ during inference and from the approximate posterior distribution $q_{\phi'}(z^s_t|s_{\leq t}, g, c) = \mathcal{N}(\mu^{s'}_t, \sigma^{s'2}_t \mathbf{I})$ during training. $h^s_t$ and the distribution of $z^s_t$ are given by:

$$h^s_t = \boldsymbol{GRU}_s([z^s_{t-1}; h^w_{t-1}], h^s_{t-1}) \quad (9)$$

$$[\mu^s_t; \log \sigma^{s2}_t] = \boldsymbol{MLP}_\phi(h^s_t, bow(g_t)) \quad (10)$$

$$[\mu^{s'}_t; \log \sigma^{s'2}_t] = \boldsymbol{MLP}_{\phi'}(h^s_t, bow(g_t), s_t) \quad (11)$$

where $h^w_{t-1}$ is the last hidden state of the word decoder after decoding sentence $s_{t-1}$, and $\boldsymbol{GRU}_s$ denotes the GRU unit of the sentence decoder. By this means, we constrain the distribution of $z^s_t$ in two aspects. First, to strengthen the connection from the planning result $g$, we additionally condition $z^s_t$ on $g_t$ to keep $z^s_t$ focused on $g_t$. Second, to capture the dependencies on $s_{<t}$, we explicitly model the dependencies among local latent variables by inputting $z^s_{t-1}$ to the sentence decoder, so that $z^s_t$ is conditioned on $z^s_{<t}$ and is expected to model smooth transitions in a long text.

We initialize the hidden state $h^s_0$ by encoding the input $x$, the global planning latent variable $z^p$ and the planning result $g$:

$$h^g_t = \boldsymbol{GRU}_g(bow(g_t), h^g_{t-1}) \quad (12)$$

$$h^s_0 = W_s[\boldsymbol{enc}(x); z^p; h^g_T] + b_s \quad (13)$$

where $h^g_T$ is the last hidden state of $\boldsymbol{GRU}_g$ that encodes the planning result $g$.

**Word-level Generation** The word decoder (a GRU) conducts word-level generation; it decodes a sentence $s_t = argmax_{s_t} P(s_t|s_{<t}, z^s_t, g, c)$ conditioned on $\{h^s_t, z^s_t, g_t\}$. Specifically, we sample word $w^t_k$ of $s_t$ as follows:

$$w^t_k \sim P(w^t_k | w^t_{<k}, s_{<t}, z^s_t, g, c) \quad (14)$$

### 4.5 Loss Function

We train our model end-to-end. The loss function has three terms: the negative evidence lower bound (ELBO) of $\log P(y|x)$ ($\mathcal{L}_1$), the loss of predicting the stop signal ($\mathcal{L}_2$) and the bag-of-word loss (Zhao et al., 2017) ($\mathcal{L}_3$).

We first derive the ELBO:

$$\begin{aligned} \log P(y|x) \geq E_{q_{\theta'}(z^p|x,y)}[\log P(y|x, z^p)] \\ - D_{KL}(q_{\theta'}(z^p|x, y)||p_\theta(z^p|x)) \end{aligned} \quad (15)$$

$$\begin{aligned} \log P(y|x, z^p) = \log P(g, y|x, z^p) \\ = \sum_{t=1}^{T} \log P(g_t|g_{<t}, x, z^p) \\ + \log P(s_t|s_{<t}, g, x, z^p) \end{aligned} \quad (16)$$

$$\begin{aligned} \log P(s_t|s_{<t}, g, x, z^p) \\ \geq E_{q_{\phi'}(z^s_t|s_{\leq t}, g, x, z^p)}[\log P(s_t|s_{<t}, z^s_t, g, x, z^p)] \\ - D_{KL}(q_{\phi'}(z^s_t|s_{\leq t}, g, x, z^p)||p_\phi(z^s_t|s_{<t}, g, x, z^p)) \end{aligned} \quad (17)$$

We can obtain the ELBO by unfolding the right hand side of Eq. 15 with Eq. 16 and 17. During

training, we use linear KL annealing technique to alleviate the KL collapse problem(Bowman et al., 2016).

$\mathcal{L}_2$ is given by:

$$\mathcal{L}_2 = \sum_{t=1}^{T-1} \log P_t^{stop} + \log(1 - P_T^{stop}) \quad (18)$$

$\mathcal{L}_3$ is the sum of bag-of-word loss (Zhao et al., 2017) applied to each sentence, which is another technique to tackle the KL collapse problem.

## 5 Experiments

### 5.1 Dataset

We evaluated PHVM on two generation tasks. The first task is the new advertising text generation task which is to generate a long advertising text that covers all given attribute-value pairs for a piece of clothing. The second task is the recipe generation task from (Kiddon et al., 2016) which is to generate a correct recipe for the given recipe title and ingredient list.

**Advertising Text Generation** We constructed our dataset from a Chinese e-commerce platform. The dataset consists of 119K pairs of advertising text and clothing specification table. Each table is a set of attribute-value pairs describing a piece of clothing. We made some modifications to the original specification tables. Specifically, if some attribute-value pairs from a table do not occur in the corresponding text, the pairs are removed from the table. We also recognized attribute values by string matching with a dictionary of attribute values. If a pair occurs in the text but not in the table, the pair is added to the table.

The statistics are shown in Table 1 and Table 2.

| Category | Tops | Dress / Skirt | Pants |
|---|---|---|---|
| # Type | 22 | 23 | 9 |
| # Attr. | 13 | 16 | 11 |
| # Val. | 264 | 284 | 203 |
| Avg. # Input Pairs | 7.7 | 7.7 | 6.6 |
| Avg. Len. | 110 | 111 | 108 |
| # Instances | 48K | 47K | 24K |

Table 1: Detailed statistics of our dataset. *# Attr.* / *# Val.*: the total number of attributes / attribute values.

Our dataset consists of three categories of clothing: tops, dress / skirt, and pants, which are further divided into 22, 23, and 9 types respectively (E.g., shirt, sweater are two types of tops). Other categories (e.g., hats and socks) are discarded because these categories have insufficient data for training. The average length of advertising text is about 110 words. To evaluate the expression diversity of our dataset, we computed distinct-4 (see Section 5.3) on 3,000 randomly sampled texts from our dataset. The distinct-4 score is 85.35%, much higher than those of WIKIBIO (Lebret et al., 2016) and ROTOWIRE (Wiseman et al., 2017) (two popular data-to-text datasets). Therefore, our dataset is suitable for evaluating long and diverse text generation[3].

| # Attr. | # Val. | Vocab | Avg. # Input Pairs | Avg. # Len. |
|---|---|---|---|---|
| 28 | 633 | 54.9K | 7.5 | 110.2 |

Table 2: General statistics of our dataset. We counted the size of vocabulary after removing brand names.

We left 1,070 / 3,127 instances for validation / test, and used the remainder for training.

**Recipe Text Generation** We used the same train-validation-test split (82,590 / 1,000 / 1,000) and pre-processing from (Kiddon et al., 2016). In the training set, the average recipe length is 102 tokens, and the vocabulary size of recipe title / text is 3,793 / 14,103 respectively. The recipe dataset covers a wide variety of recipe types indicated by the vocabulary size of recipe title.

### 5.2 Baselines

We compared our model with four strong baselines where the former two do not perform planning and the latter two do:

**Checklist:** This model utilizes an attention-based checklist mechanism to record what input data has been mentioned, and focuses more on what has not during generation (Kiddon et al., 2016).

**CVAE:** The CVAE model proposed by Zhao et al. (2017) uses a latent variable to capture the diversity of responses in dialogue generation. We adapted it to our task by replacing the hierarchical encoder with a one-layer bidirectional GRU.

**Pointer-S2S:** A two-stage method (Puduppully et al., 2019) that decides the order of input data with Pointer Network (Vinyals et al., 2015) before generation with Sequence-to-Sequence (Seq2Seq) (Bahdanau et al., 2015).

**Link-S2S** Link-S2S (Sha et al., 2018) is a Seq2Seq with link-based attention mechanism

[3]We presented a detailed comparison with other benchmark corpora in Appendix A.2.

| Models | BLEU (%) | Coverage (%) | Length | Distinct-4 (%) | Repetition-4 (%) |
|---|---|---|---|---|---|
| Checklist | **4.17** | 84.52** | 83.61** | 21.95** | 46.40** |
| CVAE | 4.02 | 77.65** | 80.96** | 41.69** | 36.58** |
| Pointer-S2S | 3.88** | 85.97** | 74.88** | 18.16** | 36.78** |
| Link-S2S | 3.90** | 70.49** | **95.65** | 16.64** | 59.83** |
| PHVM (ours) | 2.85** | **87.05** | 89.20** | **72.87** | **3.90** |
| w/o $z^p$ | 3.07** | 84.74** | 91.97** | 70.51** | 4.19 |
| w/o $z_t^s$ | 3.38** | 84.89** | 75.28** | 42.32** | 20.88** |

Table 3: Automatic evaluation for advertising text generation. We applied bootstrap resampling (Koehn, 2004) for significance test. Scores that are significantly worse than the best results (in bold) are marked with ** for p-value $< 0.01$.

where a link matrix parameterizes the probability of describing one type of input item after another.

### 5.3 Implementation Details

For both advertising text generation and recipe text generation, the settings of our model have many in common. The dimension of word embedding is 300. All embeddings were randomly initialized. We utilized GRU for all RNNs. All RNNs, except the input encoder, the plan decoder, and the plan encoder, have a hidden size of 300. The global planning latent variable and local latent variables have 200 dimensions. We set batch size to 32 and trained our model using the Adam optimizer (Kingma and Ba, 2015) with a learning rate of 0.001 and gradient clipping threshold at 5. We selected the best model in terms of $\mathcal{L}_1 + \mathcal{L}_2$ on the validation set.

As we need to train the plan decoder with full supervision, we extracted plans from the texts by recognizing attribute values (or ingredients) in each sentence with string match heuristics. Some sentences do not mention any input items; we associated these sentences with a special tag, which is treated as a special input item for Pointer-S2S, Link-S2S, and our model. Although our extraction method can introduce errors, the extracted plans are sufficient to train a good plan decoder[4].

**Advertising Text Generation** We embedded an attribute-value pair by concatenating the embedding of attribute and the embedding of attribute value. Embedding dimensions for attribute and attribute value are 30 and 100 respectively. The input encoder, the plan decoder, and the plan encoder all have a hidden size of 100.

**Recipe Text Generation** We embedded a multi-word title (ingredient) by taking average pooling of the embeddings of its constituent words. Embedding dimensions for title word and ingredient word are 100 and 200 respectively. The input encoder, the plan decoder, and the plan encoder all have a hidden size of 200.

[4]Our corpus and code are available at /https://github.com/ZhihongShao/Planning-based-Hierarchical-Variational-Model.

### 5.4 Automatic Evaluation Metrics

We adopted the following automatic metrics. (1) **Corpus BLEU**: BLEU-4 (Papineni et al., 2002). (2) **Coverage**: This metric measures the average proportion of input items that are covered by a generated text. We recognized attribute values (ingredients) with string match heuristics. For the advertising text generation task, synonyms were also considered. (3) **Length**: The average length of the generated texts. (4) **Distinct-4**: Distinct-n (Li et al., 2016) is a common metric for diversity which measures the ratio of distinct n-grams in generated tokens. We adopted distinct-4. (5) **Repetition-4**: This metric measures redundancy with the percentage of generated texts that repeat at least one 4-gram.

### 5.5 Advertising Text Generation

#### 5.5.1 Automatic Evaluation

Table 3 shows the experimental results. As our dataset possesses high expression diversity, there are many potential expressions for the same content, which leads to the low BLEU scores of all models. Our model outperforms the baselines in terms of coverage, indicating that it learns to arrange more input items in a long text. With content ordering, Pointer-S2S outperforms both Checklist and CVAE in coverage. By contrast, our planning mechanism is even more effective in controlling generation: each sentence generation subtask is specific and focused, and manages to cover 95.16% of the corresponding group on average. Noticeably, Link-S2S also models planning but

| Models | Grammaticality | | | $\kappa$ | Coherence | | | $\kappa$ |
|---|---|---|---|---|---|---|---|---|
| | Win (%) | Lose (%) | Tie (%) | | Win (%) | Lose (%) | Tie (%) | |
| PHVM vs. Checklist | 59.0** | 23.5 | 17.5 | 0.484 | 54.5* | 42.5 | 3.0 | 0.425 |
| PHVM vs. CVAE | 69.5** | 13.5 | 17.0 | 0.534 | 60.0** | 37.0 | 3.0 | 0.426 |
| PHVM vs. Pointer-S2S | 76.5** | 17.0 | 6.5 | 0.544 | 56.5** | 39.0 | 4.5 | 0.414 |
| PHVM vs. Link-S2S | 66.0** | 28.5 | 5.5 | 0.462 | 62.5** | 31.5 | 6.0 | 0.415 |

Table 4: Manual pair-wise evaluation for advertising text generation. We conducted Sign Test for significance test. Scores marked with * mean p-value $< 0.05$ and ** for p-value $< 0.01$. $\kappa$ denotes Fleiss' kappa, all indicating *moderate agreement*.

has the lowest coverage, possibly because a static link matrix is unable to model flexible content arrangement in long text generation. As for diversity, our model has substantially lower repetition-4 and higher distinct-4, indicating that our generated texts are much less redundant and more diversified. Notably, Link-S2S has the longest texts but with the highest repetition-4, which produces many redundant expressions.

To investigate the influence of each component in the hierarchical latent structure, we conducted ablation tests which removed either global latent variable $z^p$ or local latent variables $z_t^s$. As observed, removing $z^p$ leads to significantly lower distinct-4, indicating that $z^p$ contributes to expression diversity. The lower coverage is because the percentage of input items covered by a planning result decreases from 98.4% to 94.4% on average, which indicates that $z^p$ encodes useful information for planning completeness. When removing $z_t^s$, distinct-4 drops substantially, as the model tends to generate shorter and more common sentences. This indicates that $z_t^s$ contributes more to capturing variations of texts. The significantly higher repetition-4 is because removing $z_t^s$ weakens the dependencies among sentences so that the word decoder is less aware of the preceding generated context. The lower coverage is because each generated sentence covers less planned items (from 95.16% to 93.07% on average), indicating that $z_t^s$ keeps sentence $s_t$ more focused on its group.

### 5.5.2 Manual Evaluation

To better evaluate the quality of the generated texts, we conducted pair-wise comparisons manually. Each model generates texts for 200 randomly sampled inputs from the test set. We hired five annotators to give preference (win, lose or tie) to each pair of texts (ours vs. a baseline, 800 pairs in total).

**Metrics** Two metrics were independently evaluated during annotation: **grammaticality** which measures whether a text is fluent and grammatical, and **coherence** which measures whether a text is closely relevant to input, logically coherent, and well-organized.

**Results** The annotation results in Table 4 show that our model significantly outperforms baselines in both metrics. Our model produces more logically coherent and well-organized texts, which indicates the effectiveness of the planning mechanism. It is also worth noting that our model performs better in terms of grammaticality. The reason is that long text generation is decomposed into sentence generation sub-tasks which are easier to control, and our model captures inter-sentence dependencies through modeling the dependencies among local latent variables.

### 5.5.3 Diversity of Planning

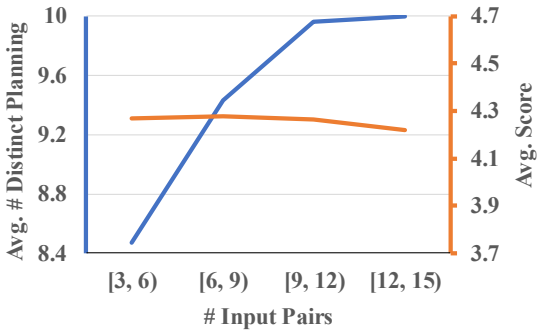


Figure 3: Average number of distinct planning results (left) / average score of generation quality (right) when the number of input pairs varies.

To evaluate how well our model can capture the diversity of planning, we conducted another manual evaluation. We randomly sampled 100 test inputs and generated 10 texts for each input by repeatedly sampling latent variables. Five annotators were hired to score (a Likert scale $\in [1, 5]$)

| Models | BLEU (%) | Coverage (%) | Length | Distinct-4 (%) | Repetition-4 (%) |
|---|---|---|---|---|---|
| Checklist § | 3.0 | 67.9 | N/A | N/A | N/A |
| Checklist | 2.6** | 66.9* | 67.59 | 30.67** | 39.1** |
| CVAE | 4.6 | 63.0** | 57.49** | 52.53** | 38.7** |
| Pointer-S2S | 4.3 | 70.4** | 59.18** | 30.72** | 36.4** |
| Link-S2S | 1.9** | 53.8** | 40.34** | 24.93** | 31.6** |
| PHVM (ours) | **4.6** | **73.2** | **70.92** | **67.86** | **17.3** |

Table 5: Automatic evaluation for recipe text generation. Checklist was trained with its own source code. We also re-printed results from (Kiddon et al., 2016) (i.e., Checklist §). We applied bootstrap resampling (Koehn, 2004) for significance test. Scores that are significantly worse than the best results (in bold) are marked with * for p-value $<$ 0.05 or ** for p-value $<$ 0.01.

a text about whether it is a qualified advertising text, which requires comprehensive assessment in terms of fluency, redundancy, content organization, and coherence. We computed the average of five ratings as the final score of a generated text.

**Results** The average score of a generated text is 4.27. Among the 1,000 generated texts, 79.0% of texts have scores above 4. These results demonstrate that our model is able to generate multiple high-quality advertising texts for the same input.

We further analyzed how our model performs with different numbers of input attribute-value pairs (see Figure 3). A larger number of input items indicates more potential reasonable ways of content arrangement. As the number of input items increases, our model produces more distinct planning results while still obtaining high scores (above 4.2). It indicates that our model captures the diversity of reasonable planning. The average score drops slightly when the number of input pairs is more than 12. This is due to insufficient training data for this range of input length (accounting for 6.5% of the entire training set).

To further verify the planning diversity, we also computed self-BLEU (Zhu et al., 2018) to evaluate how different planning results (or texts) for the same input overlap (by taking one planning result (or text) as hypothesis and the rest 9 for the same input as reference and then computing BLEU-4). The average self-BLEU of the planning results is 43.37% and that of the texts is 16.87%, which demonstrates the much difference among the 10 results for the same input.

**Annotation Statistics** The Fleiss' kappa is 0.483, indicating *moderate agreement*.

### 5.6 Recipe Text Generation

Table 5 shows the experimental results. Our model outperforms baselines in terms of coverage and diversity; it manages to use more given ingredients and generates more diversified cooking steps. We also found that Checklist / Link-S2S produces the general phrase "all ingredients" in 14.9% / 24.5% of the generated recipes, while CVAE / Pointer-S2S / PHVM produce the phrase in 7.8% / 6.3% / 5.0% of recipes respectively. These results demonstrate that our model may generalize well to other data-to-text generation tasks.

## 6 Case Study

We present examples for both tasks in Appendix B.

## 7 Conclusion and Future Work

We present the Planning-based Hierarchical Variational Model (PHVM) for long and diverse text generation. A novel planning mechanism is proposed to better model input data and address the inter-sentence incoherence problem. PHVM also leverages a hierarchical latent structure to capture the diversity of reasonable planning and sentence realization. Experiments on two data-to-text corpora show that our model is more competitive to generate long and diverse texts than state-of-the-art baselines.

Our planning-based model may be inspiring to other long text generation tasks such as long text machine translation and story generation.

## Acknowledgements

This work was supported by the National Science Foundation of China (Grant No. 61936010/61876096) and the National Key R&D Program of China (Grant No. 2018YFC0830200). We would like to thank THUNUS NExT Joint-Lab for the support.

## References


Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

John A. Bateman. 1997. Enabling technology for multilingual natural language generation: the KPML development environment. *Natural Language Engineering*, 3(1):15–55.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, pages 10–21.

Qibin Chen, Junyang Lin, Yichang Zhang, Hongxia Yang, Jingren Zhou, and Jie Tang. 2019. Towards knowledge-based personalized product description generation in e-commerce. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019.*, pages 3040–3050.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of SSST@EMNLP 2014, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, Doha, Qatar, 25 October 2014*, pages 103–111.

Hercules Dalianis and Eduard H. Hovy. 1993. Aggregation in natural language generation. In *Trends in Natural Language Generation, An Artificial Intelligence Perspective, Fourth European Workshop, EWNLG '93, Pisa, Italy, April 28-30, 1993, Selected Papers*, pages 88–105.

Kees van Deemter, Mariët Theune, and Emiel Krahmer. 2005. Real versus template-based natural language generation: A false opposition? *Computational Linguistics*, 31(1):15–24.

Pablo A Duboue and Kathleen R McKeown. 2003. Statistical acquisition of content selection rules for natural language generation. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 121–128. Association for Computational Linguistics.

Dominic Espinosa, Michael White, and Dennis Mehay. 2008. Hypertagging: Supertagging for surface realization with CCG. In *ACL 2008, Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics, June 15-20, 2008, Columbus, Ohio, USA*, pages 183–191.

Xiaocheng Feng, Ming Liu, Jiahao Liu, Bing Qin, Yibo Sun, and Ting Liu. 2018. Topic-to-essay generation with neural networks. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden.*, pages 4078–4084.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *J. Artif. Intell. Res.*, 61:65–170.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2672–2680.

Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. 2018. Long text generation via adversarial training with leaked information. In (McIlraith and Weinberger, 2018), pages 5141–5148.

Eduard H. Hovy. 1993. Automated discourse generation using discourse structure relations. *Artif. Intell.*, 63(1-2):341–385.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally coherent text generation with neural checklist models. In (Su et al., 2016), pages 329–339.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *EMNLP*, pages 388–395. ACL.

Ioannis Konstas and Mirella Lapata. 2013. Inducing document plans for concept-to-text generation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1503–1514. ACL.

Karen Kukich. 1983. Design of a knowledge-based report generator. In *21st Annual Meeting of the Association for Computational Linguistics, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, June 15-17, 1983.*, pages 145–150.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In (Su et al., 2016), pages 1203–1213.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *NAACL HLT 2016, The 2016 Conference of the*

*North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 110–119.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1106–1115.

Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. 2018. Table-to-text generation by structure-aware seq2seq learning. In (McIlraith and Weinberger, 2018), pages 4881–4888.

Sheila A. McIlraith and Kilian Q. Weinberger, editors. 2018. *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*. AAAI Press.

Susan W McRoy, Songsak Channarukul, and Syed S Ali. 2003. An augmented template-based approach to text realization. *Natural Language Engineering*, 9(4):381–420.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 720–730.

Amit Moryossef, Yoav Goldberg, and Ido Dagan. 2019. Step-by-step: Separating planning from realization in neural data-to-text generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Preksha Nema, Shreyas Shetty, Parag Jain, Anirban Laha, Karthik Sankaranarayanan, and Mitesh M. Khapra. 2018. Generating descriptions from structured data using a bifocal attention mechanism and gated orthogonalization. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1539–1550.

Martha Palmer, Rebecca Hwa, and Sebastian Riedel, editors. 2017. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318. ACL.

Yookoon Park, Jaemin Cho, and Gunhee Kim. 2018. A hierarchical latent structure for variational conversation modeling. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1792–1801.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019. Data-to-text generation with content selection and planning. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*.

Ehud Reiter and Robert Dale. 1997. Building applied natural language generation systems. *Natural Language Engineering*, 3(1):57–87.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3295–3301. AAAI Press.

Lei Sha, Lili Mou, Tianyu Liu, Pascal Poupart, Sujian Li, Baobao Chang, and Zhifang Sui. 2018. Order-planning neural text generation from structured data. In (McIlraith and Weinberger, 2018), pages 5414–5421.

Yuanlong Shao, Stephan Gouws, Denny Britz, Anna Goldie, Brian Strope, and Ray Kurzweil. 2017. Generating high-quality and informative conversation responses with sequence-to-sequence models. In (Palmer et al., 2017), pages 2210–2219.

Dinghan Shen, Asli Çelikyilmaz, Yizhe Zhang, Liqun Chen, Xin Wang, Jianfeng Gao, and Lawrence Carin. 2019. Towards generating long and coherent text with multi-level latent variable models. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 2079–2089.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. 2015. Learning structured output representation using deep conditional generative models. In *NIPS*, pages 3483–3491.

Amanda Stent, Rashmi Prasad, and Marilyn A. Walker. 2004. Trainable sentence planning for complex information presentations in spoken dialog systems. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, 21-26 July, 2004, Barcelona, Spain.*, pages 79–86.

Jian Su, Xavier Carreras, and Kevin Duh, editors. 2016. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*. The Association for Computational Linguistics.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2692–2700.

Sam Wiseman, Stuart M. Shieber, and Alexander M. Rush. 2017. Challenges in data-to-document generation. In (Palmer et al., 2017), pages 2253–2263.

Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2017. Topic aware neural response generation. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*.

Jingjing Xu, Xuancheng Ren, Junyang Lin, and Xu Sun. 2018. Diversity-promoting gan: A cross-entropy based generative adversarial network for diversified text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3940–3949. Association for Computational Linguistics.

Tiancheng Zhao, Ran Zhao, and Maxine Eskénazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 654–664.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, pages 1097–1100.

# Transforming Delete, Retrieve, Generate Approach for Controlled Text Style Transfer

**Akhilesh Sudhakar, Bhargav Upadhyay, Arjun Maheswaran**
Agara
{akhilesh, bhargav, arjun}@agaralabs.com

## Abstract

Text style transfer is the task of transferring the style of text having certain stylistic attributes, while preserving non-stylistic or content information. In this work we introduce the Generative Style Transformer (GST) - a new approach to rewriting sentences to a target style in the absence of parallel style corpora. GST leverages the power of both, large unsupervised pre-trained language models as well as the Transformer. GST is a part of a larger 'Delete Retrieve Generate' framework, in which we also propose a novel method of deleting style attributes from the source sentence by exploiting the inner workings of the Transformer. Our models outperform state-of-art systems across 5 datasets on sentiment, gender and political slant transfer. We also propose the use of the GLEU metric as an automatic metric of evaluation of style transfer, which we found to compare better with human ratings than the predominantly used BLEU score.

## 1 Introduction

Text style transfer is an important Natural Language Generation (NLG) task, and has wide-ranging applications from adapting conversational style in dialogue agents (Zhou et al., 2017), obfuscating personal attributes (such as gender) to prevent privacy intrusion (Reddy and Knight, 2016), altering texts to be more formal or informal (Rao and Tetreault, 2018), to generating poetry (Yang et al., 2018). The main challenge faced in building style transfer systems is the lack of parallel corpora between sentences of a particular style and sentences of another, such that sentences in a pair differ only in style and not content (non-stylistic part of the sentence). This has given rise to methods that circumvent the need for such parallel corpora.

Previous approaches using non-parallel corpora, that employ learned latent representations to disentangle style and content from sentences, are typically adversarially trained (Hu et al., 2017; Shen et al., 2017; Fu et al., 2018). However these models a) are hard to train and take long to converge, b) need to be re-trained from scratch to change the trade-off between content retention and style transfer c) suffer from sparsity of latent disentangled representations, d) produce sentences of bad quality (according to human ratings) and e) do not offer fine-grained control over target style attributes.

Li et al. (2018) find that style attributes are more often than not, localized to a small subset of words of a sentence. Building on this inductive bias, they model style transfer in a "Delete Retrieve Generate" framework (hereby referred to as DRG) which aims to 1) delete only the set of attribute words from a sentence to give the content, 2) retrieve attribute words from the target style corpus, and 3) use a neural editor (an encoder-decoder LSTM) to generate the final sentence from the content and retrieved attributes.

While DRG as a framework leads to output sentences that are better in quality than previous approaches, their individual Delete and Generate methods are susceptible to: a) removing core content words which would preserve crucial context, b) failing to remove source style attributes that should be replaced with target style attributes, c) the LSTM-based encoder-decoder model not being robust to errors made by the Delete and Retrieve models, d) generating sentences that are not fluent, by abruptly forcing retrieved attributes into the source sentence and e) failing on longer input sentences.

In this work, we propose a novel approach to rewrite sentences into a target style, that leverages the power of both a) transfer learning by using an unsupervised language model trained on a large corpus of unlabeled text, as well as b) the Trans-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3269–3279,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

former (Vaswani et al., 2017). We refer to our Transformer as the Generative Style Transformer **(GST)**. We use the DRG framework proposed by Li et al. (2018) but we overcome the shortcomings of their a) Delete mechanism, by using the attention weights of another Transformer that we refer to as the Delete Transformer **(DT)**, and b) Generate mechanism by using **GST**, which does away with the need for (and consequent shortfalls of) a sequence-to-sequence encoder-decoder architecture using LSTMs.

We outperform the current state-of-art systems on transfer of a) sentiment[1], b) gender and c) political slant. Our approach is advantageous in that it is simple, controllable and exploits the important inductive bias described, while at the same time it leverages the power of Transformers in novel ways.

All code, data and results for this work can be found in our Github repository [2].

## 2 Our Approach

Given a dataset $D = \{(x_1, s_1), ..., (x_m, s_m)\}$ where $x_i$ is a sentence and $s_i \in S$ is a specific style, our goal is to learn a conditional distribution $P(y|x, s^{tgt})$ such that $Style(y) = s^{tgt}$, where style is determined by an oracle that can accurately determine the style of a given sentence. For instance, for the sentiment transfer task, $S = \{$'Positive', 'Negative'$\}$. Using the DRG framework, we model our task in 3 steps:
**(1)** A **Delete** model which learns $P(c, a|x)$ such that $c$ and $a$ are non-stylistic and stylistic components of $x$ respectively, $Style(c) \notin S$ (i.e., $c$ does not have any particular style) and $x$ can be completely reconstructed from $c$ and $a$, **(2)** A **Retrieve** model which retrieves a set of (optional) target attributes $a^{tgt}$ from $D_{s^{tgt}}$, the corpus of sentences of target style, and **(3)** A **Generate** model in two flavors: a) one which learns to generate a sentence in the target distribution $P(y|c, s^{tgt})$ and b) another which learns to generate a sentence in the target distribution $P(y|c, a^{tgt})$, both such that $Style(y) = s^{tgt}$. We now elaborate on each of these components individually.

[1]We use style in a broad socio-linguistic sense that encompasses sentiment too, for the purpose of this work

[2]https://github.com/agaralabs/transformer-drg-style-transfer

### 2.1 Delete

For an input sentence "The restaurant was *big* and *spacious*", in the case of a style transfer task from positive to negative sentiment, the Delete model should be capable of deleting the style attributes *big* and *spacious*.

Our approach to attribute deletion is based on 'input reduction' (Feng et al., 2018), based on the observation that certain words and phrases significantly contribute to the style of a sentence. For a sentence $x$ of style $s_j$ having a set of attributes $a$, a style classifier will be confused about its style if the attributes in $a$ are removed from $x$. We describe a mechanism to assign an *importance* score to each token in $x$, which is reflective of its contribution to style. These scores allows us to distinguish style attributes from content.

#### 2.1.1 Delete Transformer

To build intuition, any attention-based style classifier defines a probability distribution over style labels:

$$p(s|x) = g(v, \alpha) \tag{1}$$

where $v$ is a tensor such that $v[i]$ is an encoding of $x[i]$, and $\alpha$ is a tensor of attention weights such that $\alpha[i]$ is the weight attributed to $v[i]$ by the classifier in deciding probabilities for each $s_j$. The $\alpha$ scores can be treated as *importance* scores and be used to identify attribute words, (which typically tend to have higher scores). Motivated by the recent successes of the Transformer (Vaswani et al., 2017) and more specifically, BERT (Devlin et al., 2018), on a number of text classification tasks (including achieving state-of-art results on sentiment classification), we use a BERT-based transformer as our style classifier and refer to it as Delete Transformer **(DT)**. However, since **DT** has multiple attention heads and multiple blocks (layers), extracting a single set of attention weights $\alpha$ is a non-trivial task. This is further complicated by the fact that every layer and head encodes different aspects of semantic and linguistic structure (Vig, 2019). We then use a novel method to extract a specific attention head and layer combination that encodes style information and that can be directly used as *importance* scores.

**Attribute extraction:** We use the same input representation as Figure 3(b) in Devlin et al. (2018) wherein a '[CLS]' token is added before the sentence tokens. Since the softmax classifica-

tion layer is used over the attention stack of the '[CLS]' token in BERT classifiers, the attention weights of other input tokens that correspond to '[CLS]' are of special interest in identifying significant sentence tokens. First, we iterate over each pair $< h, l >$ (head-layer pair) and extract the attention scores for every token $w$ of $x$ as follows:

$$\alpha_{h,l}(w) = softmax_{w \in x}(Q_{h,l,[CLS]} K^T_{h,l,w}) \quad (2)$$

where 'Q' and 'K' carry the same original connotations of query and key vectors as used by Vaswani et al. (2017), in the Transformer as:

$$Att(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \quad (3)$$

We then remove the top $\gamma|x|$ tokens from $x$, based on *importance* scores calculated as in Eq. 2. Keeping in line with Feng et al. (2018), we call this removal a 'reduction', and denote the resulting reduced sentence as $x'_{h,l}$. $\gamma$ is a parameter we tune to each dataset which allows us to control the proportion of words in a sentence to be deleted, and $|x|$ denotes the number of tokens in $x$. We calculate a score $z(x'_{h,l})$:

$$z(x'_{h,l}) = \frac{p(s|x'_{h,l}) + \lambda}{\sum_{s'} p(s'|x'_{h,l}) + \lambda} \quad (4)$$

where $\lambda$ is a smoothing parameter, $s$ is the style label assigned maximum probability by the softmax distribution over all styles in the label set $S$, and $s' = S - \{s\}$. The final pair $< h_s, l_s >$ out of combinations of all heads H and layers L, is obtained by averaging the score in Eq. 4 over a validation set of 'reduced' sentences $D'_{val}$ as follows:

$$(h_s, l_s) = \underset{h \in H, l \in L}{\text{argmin}} \frac{\sum_{x'_{h,l} \in D'_{val}} z(x'_{h,l})}{|D'_{val}|} \quad (5)$$

A 'reduction' of any input sentence $x$ based on $< h_s, l_s >$ gives us $x'_{h_s,l_s}$ which we refer to as the content $c$. The removed tokens are the attributes $a$.

**Evaluation of Extracted Attributes:** We evaluate our Delete method using human evaluation on Amazon Mechanical Turk[3], on which annotators were asked to choose if all the style-related attributes are extracted correctly by our Delete mechanism, and if any non-style attributes are wrongly deleted. We used 200 random sentences from our test set for sentiment transfer, for this evaluation. Our method deleted all style attributes on 89% of examples, and wrongly deleted non-style attributes only 12% of the time. In comparison, the Delete mechanism proposed by Li et al. (2018) deleted all style attributes only 67% of the time, and wrongly deleted non-style attributes over 29% of the time.

[3]https://www.mturk.com/

### 2.2 Retrieve

We retrieve a sentence from the target style corpus of sentences according to:

$$x^{tgt} = argmin_{x' \in D_{stgt}} d(c_x, c_{x'}) \quad (6)$$

where $d$ is a distance metric, such that contents which are closer according to $d$ will have compatible attributes as they occur in similar contexts. We experiment with multiple retrieval mechanisms, using cosine similarity over different sentence representations: a) TF-IDF weighted, b) Averaged-GloVe over all tokens of a sentence and c) Universal Sentence Encoder (Cer et al., 2018). We obtain best retrieval results using TF-IDF vector similarity.

### 2.3 Generate

Our approach to generate sentences of the target style leverages both the power of transfer learning by using an unsupervised language model trained on a large corpus of unlabeled text, as well as the Transformer model. The model we use is a multi-layer 'decoder-only' Transformer which is based on the Generative Pre-trained Transformer (GPT) of Radford et al.. This is our Generative Style Transformer **(GST)**. **GST** has masked attention heads that enable it to look only at the tokens to its left, and not to those to its right. **GST** derives inspiration from the fact that recently, many large generatively pre-trained Transformer models have shown state-of-art performance upon being finetuned on a number of downstream tasks. It is trained to learn a representation of content words and (retrieved) attribute words presented to it, and generate fluent sentences in the domain of the target style while attending to both content and attribute words appropriately.

#### 2.3.1 Variants of GST (B-GST and G-GST)

Taking cues from Li et al. (2018), we train GST in two flavors: the Blind Generative Style

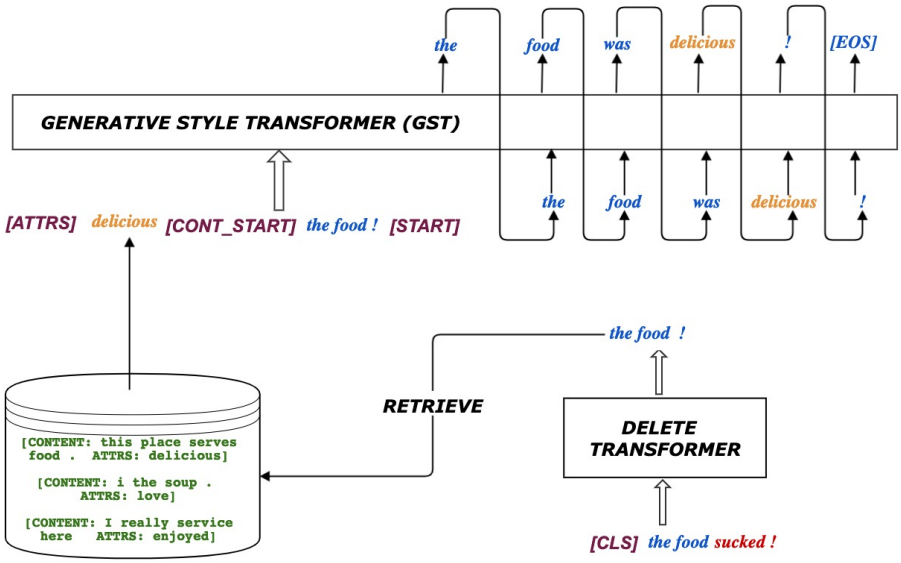


Figure 1: Our architecture, with an example from the Yelp dataset for the task of sentiment transfer

Transformer **(B-GST)** and the Guided Generative Style Transformer **(G-GST)**. For a sentence $x$ of a source style $s^{src}$ with content $c$ and retrieved (target style) attributes $a^{tgt}$, the two variations are learnt as follows.

**B-GST:** The inputs to this model are only $c$, and $s^{tgt}$. The output $y$ of the model is the generated sentence in the target style. In this setting, the model is free to generate the output sentence, given the content and the target style, and is **blind** to specific desired target attributes. **B-GST** can be useful in cases when the target corpus does not have sentences that are similar to the source corpus, which causes the Retrieve model to retrieve incompatible target attributes.

**G-GST:** The inputs to this model are $c$, and $a^{tgt}$, and the output $y$ of the model is the generated sentence in the target style. In this setting, the model is **guided** towards generating a target sentence with desired attributes. **G-GST** is useful for two reasons. Firstly, in cases when the target corpus has similar sentences to the source corpus, it reduces sparsity by giving the model information of target attributes. Secondly, and more importantly, it allows fine-grained control of output generation by manually specifying target attributes that we desire during inference time, without even using the Retrieve component. This controllability is an important feature of G-GST that most other latent-representation based style transfer approaches do not offer.

#### 2.3.2 Input Representation and Output Decoding

Taking inspiration from Devlin et al. (2018), we add special tokens to indicate target style, and to indicate the demarcation between content and attributes. For **B-GST** the input at timestep $t$ of target sentence prediction consists of special tokens to denote: a) target style $s^{tgt}$, b) the start of content $c$, d) the start of output, followed by all target tokens up till and including timestep $t-1$. **G-GST** has a similar input representation, except that a special token to indicate start of retrieved attributes is added, and the retrieved attributes are provided before the content. The target style $s^{tgt}$ is not provided. Our end-to-end architecture for **G-GST** is depicted in Figure 1, including input representation. **B-GST** is similar in nature, except that it does not use a retrieve component. At timestep $t$, both **GST**s predict the $t^{th}$ output token, by generating a probability distribution over words in the vocabulary according to: a) $p(y_t|c, y_1, y_2, ..y_{t-1})$ for **B-GST**, and b) $p(y_t|c, a^{tgt}, y_1, y_2, ..y_{t-1})$ for **G-GST**. This is done by using a softmax layer over the topmost Transformer block corresponding to $y_{t-1}$. During training time, we use the 'teacher forcing' or 'guided approach' (Bengio et al., 2015; Williams and Zipser, 1989) over decoded tokens. During test time, we beam search using softmax probabilities with a look-left window of 1 and a beam width of 5. The output beam (out of the top 5 final beams) that obtains the highest target-style match

score using the Delete Transformer described earlier, is chosen as the output sentence.

#### 2.3.3 Training

Since we do not have a parallel corpus, both GSTs are trained to minimize the reconstruction loss. Specifically, for a sentence $x$, the model learns to reconstruct $y = x$ given $c_x$, its own attributes $a_x$ (only for **G-GST**) and its own style $s^{src}$ (only for **B-GST**). More formally **B-GST** learns to maximize the following objective:

$$L(\theta) = \sum_{(x,s^{src})\in D} \log p(x|c_x, s^{src}; \theta) \quad (7)$$

However, training **G-GST** using the reconstruction loss in this manner results in the model learning to trivially combine $c_x$ and $a_x$ to generate $x$ back. In reality we want it to be capable of *adapting* target attributes into the context of the source content, in a non-trivial manner to produce a fluent sentence in the target style. To this end, we noise the inputs of the **G-GST** model during training time, by choosing random attributes for 10% of the examples (5% from the source style and 5% from the target style), to replace $a_x$. Denoting the chosen attribute for an example (either noisy or its own) to be $a'_x$, **G-GST** learns to maximize the following objective:

$$L(\theta) = \sum_{(x,s^{src})\in D} \log p(x|c_x, a'_x; \theta) \quad (8)$$

#### 2.3.4 Model Details and Pre-training

We use the PyTorch implementation of the pre-trained Transformer by HuggingFace[4], which uses the pre-trained OpenAI GPT model[5]. This model is pre-trained by Radford et al. on the BookCorpus dataset[6] of over 7000 books (around 800M words). **GST** has a sequence length of 512, 12 blocks (or layers), and 12 attention-heads in each block. All internal states (keys, queries, values, word embeddings, positional embeddings) are 768-dimensional. Input text is tokenized using Byte-Pair Encoding (BPE).

## 3 Experiments

### 3.1 Datasets

We use 5 different datasets for our experiments. These datasets have been used in previous works on style transfer.

| Dataset | Style | Train | Dev | Test |
|---|---|---|---|---|
| YELP | Positive | 270K | 2000 | 500 |
| | Negative | 180K | 2000 | 500 |
| CAPTIONS | Romantic | 6000 | 300 | 0 |
| | Humorous | 6000 | 300 | 0 |
| | Factual | 0 | 0 | 300 |
| AMAZON | Positive | 277K | 985 | 500 |
| | Negative | 279K | 1015 | 500 |
| POLITICAL | Democrat | 270K | 2000 | 28K |
| | Republican | 270K | 2000 | 28K |
| GENDER | Male | 1.34M | 2250 | 267K |
| | Female | 1.34M | 2250 | 267K |

Table 1: Dataset statistics

We use the YELP, AMAZON and CAPTIONS datasets as used by Li et al. (2018), and we retain the same train-dev-test split that they use. Further, they also provide human gold standard references for the test sets of all 3 of the above. We use the POLITICAL (Voigt et al., 2018) and GENDER (Reddy and Knight, 2016) datasets as used by Prabhumoye et al. (2018). We have retained the same train-dev-test split that they use. Table 1 shows statistics of these datasets. Following are brief descriptions of the datasets:

**YELP:** Reviews of businesses on Yelp, with each review labelled as having either positive or negative sentiment.

**AMAZON:** Product reviews on Amazon, with each review labelled as having either positive or negative sentiment.

**CAPTIONS:** Image captions, with each caption labeled as either factual, romantic, or humorous. The task is to convert factual sentences into romantic and humorous ones.

**POLITICAL:** Top-level comments on Facebook posts from members of the United States Senate and House who have public Facebook pages, with each comment labelled as having been posted by either a Republican or a Democrat politician.

**GENDER:** Reviews of food businesses on Yelp, with each review labelled as either of the two genders (male or female) corresponding to markers of sex.

### 3.2 Comparison to Previous Works

On the Yelp, Amazon, and Captions dataset, we compare with 3 previous adversarially trained models: StyleEmbedding **(SE)** (Fu et al., 2018), MultiDecoder **(MD)** (Fu et al., 2018),

[4]https://github.com/huggingface/pytorch-pretrained-BERT

[5]https://github.com/openai/finetune-transformer-lm

[6]https://www.smashwords.com/

| | YELP | | | | AMAZON | | | | CAPTIONS | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Model | Cont. | Flu. | Sty. | All | Cont. | Flu. | Sty. | All | Cont. | Flu. | Sty. | All |
| D&R | 18 | 14.5 | 22 | 13 | 40 | 35.5 | 45 | 39.5 | 20.5 | 24 | **47.75** | 30.25 |
| **B-GST** | **66** | **64** | **60** | **69** | **48** | **50.5** | **45.5** | **49** | **65.5** | **56.75** | 34 | **52.75** |
| None | 16 | 21.5 | 18 | 18 | 12 | 14 | 9.5 | 11.5 | 14 | 19.25 | 18.5 | 17 |

Table 2: Human evaluation results - each cell indicates the percentage of sentences preferred down a column (Cont. = Content preservation ; Flu. = Fluency ; Sty. = Target Style Match ; All = Overall)

| | POLITICAL | | GENDER | |
|---|---|---|---|---|
| Model | Cont. | Flu. | Cont. | Flu. |
| BT | 22 | 29 | 18 | 23 |
| **B-GST** | **69** | **61** | **70** | **67** |
| None | 9 | 10 | 12 | 10 |

Table 3: Human evaluation results - each cell indicates the percentage of sentences preferred down a column (Cont. = Content preservation ; Flu. = Fluency)

CrossAligned **(CA)** (Shen et al., 2017)) and the 2 best models - DeleteOnly **(D)** and DeleteAndRetrieve **(D&R)** of Li et al. (2018) trained using the DRG framework. A brief description of the first 3, and a detailed description of the last 2 models can be found in Li et al. (2018), so we omit elaborating on them here. At the time of writing this paper, these models are the top performing models on Yelp, Amazon and Captions, with the **D&R** model of Li et al. (2018) showing state-of-art performance. Output sentences of each of these 5 models on fixed test sets, also annotated with human reference gold standards **(H)** on all 3 datasets are provided by Li et al. (2018). We use the same for our comparison and evaluation. On the Political and Gender datasets, we compare our models against that of Prabhumoye et al. (2018), which is the state-of-art on these 2 datasets at the time of writing this paper. Their trained models for both these datasets are made publicly available. They use back-translation **(BT)** using an LSTM as a mechanism to learn latent representations of source sentences, and then employ adversarial generation techniques to make the output match a desired style (Prabhumoye et al., 2018).

## 4 Evaluation of Results

The widely agreed upon goals for a style transfer system are 1) Content preservation of the non-stylistic parts of the source sentence, 2) Style transfer strength of the stylistic attributes to the target style and 3) Fluency and correct grammar of the generated target sentence (Mir et al., 2019). To this end, we use both human and automatic evaluation to measure model performance.

### 4.1 Human Evaluation

**YELP, AMAZON and CAPTIONS:** Li et al. (2018) report state-of-art results which we corroborate through manual and automatic metrics. We then proceed to obtain human evaluations on these models along with ours through Amazon Mechanical Turk[7]. Specifically, we ask annotators to rate each pair of generated sentences given the source sentence, on content preservation, style transfer strength, fluency, and overall success. For each parameter, they are asked to choose which of the generated sentences is better, or neither of the two if they are unable to decide. Table 2 presents results on our best scoring model **B-GST** with the previous best scoring model **D&R** as a percentage of times one was preferred over the other.

**POLITICAL and GENDER:** On these 2 datasets, (Prabhumoye et al., 2018) report state-of-art results using their model **BT**, which we similarly corroborate. A comparison of our best model **B-GST**, with their results using **BT** is presented in Table 3 as a percentage of times one was preferred over the other. Since judging target style strength on these two tasks are hard for MTurkers, they only rate these datasets for content and fluency.

### 4.2 Automatic Evaluation

As has been done by previous works, we attempt to use automatic methods of evaluation to assess the performance of different models. To estimate target style strength, we use style classifiers that we train on the same training-dev-test split of Table 1, using FastText[8] (Joulin et al., 2017). These classifiers achieve 98%, 86%, 80%, 92% and 82% accuracies on the test sets of Yelp, Amazon, Captions, Political and Gender respectively. To measure content preservation, we calculate the BLEU score (Papineni et al., 2001) between the gener-

[7]https://www.mturk.com/
[8]https://fasttext.cc/

| | YELP | | | | AMAZON | | | | CAPTIONS | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Model | GL | $BL_S$ | PL | AC | GL | $BL_S$ | PL | AC | GL | $BL_S$ | PL | AC |
| SRC | 7.6 | 100.0 | 24.0 | 2.6 | 19.3 | 100.0 | 32.9 | 20.4 | 11.3 | 100.0 | 34.4 | 50.0 |
| CA | 4.4 | 48.0 | 72.8 | 72.7 | 0.0 | 15.2 | **30.1** | **83.1** | 1.6 | 24.1 | **10.1** | 50.8 |
| SE | 5.9 | 78.0 | 115.9 | 8.6 | 0.0 | 16.7 | 129.8 | 45.5 | 5.9 | 53.8 | 80.3 | 51.0 |
| MD | 5.0 | 57.3 | 205.6 | 46.8 | 0.0 | 16.5 | 122.5 | 71.8 | 4.5 | 48.7 | 40.5 | 51.3 |
| D | 6.4 | 56.7 | 75.8 | 85.0 | 0.0 | 16.2 | 55.0 | 50.6 | 7.8 | 59.1 | 52.5 | 57.5 |
| D&R | 6.9 | 58.0 | 90.0 | **89.3** | 0.0 | 16.1 | 42.2 | 50.9 | 7.8 | 49.1 | 28.8 | **67.5** |
| **G-GST** | 3.8 | 70.6 | 64.4 | 78.3 | 13.4 | 71.0 | 171.0 | 57.6 | 1.1 | 13.1 | 45.0 | 52.3 |
| **B-GST** | **11.6** | **71.0** | **38.6** | 87.3 | **14.9** | **73.6** | 55.2 | 60.0 | **12.6** | **68.3** | 28.9 | 56.0 |
| H | 100.0 | 58.1 | 67.2 | 75.2 | 100.0 | 70.5 | 77.0 | 42.6 | 100.0 | 36.4 | 41.4 | 55.5 |

Table 4: Automatic evaluation results (GL = GLEU, $BL_s$ = BLEU ; PL = Perplexity ; AC = Target Style Accuracy ; SRC = Input Sentence ; B-GST and G-GST are our models ; H = Human Reference)

| | POLITICAL | | | GENDER | | |
|---|---|---|---|---|---|---|
| Model | $BL_S$ | PL | AC | $BL_S$ | PL | AC |
| SRC | 100.0 | 62.9 | 9 | 100 | 183.4 | 18.9 |
| BT | 40.2 | **61.9** | **88.0** | 46.0 | 196.2 | 52.9 |
| **G-GST** | 76.7 | 241.6 | 67.4 | 78.5 | 252.0 | 49.0 |
| **B-GST** | **79.2** | 104.4 | 71.2 | **82.5** | **189.2** | **57.9** |

Table 5: Automatic evaluation results ($BL_s$ = BLEU; PL = Perplexity; AC = Target Style Accuracy; SRC = Input Sentence; B-GST and G-GST are our models)

ated and source sentences. To measure fluency, we finetune a large pre-trained language model, OpenAI GPT-2 (note that this is different from GPT-1 on which our Generate model is based) on the target sentences using the same training-dev-test split of Table 1. We use this language model to measure perplexity of generated sentences. The language models achieve perplexities of 24, 33, 34, 63 and 81 on the test sets of Yelp, Amazon, Captions, Political and Gender respectively. As we analyze in the next section, automatic metrics are inadequate at measuring the success of a good style transfer system.

**GLEU:** As a step towards finding an automatic metric that compares with human judgements, we propose the use of the Generalized Language Evaluation Understanding Metric (GLEU) (Napoles et al., 2015) - originally proposed as a grammatical error correction (GEC) metric. In the interest of space, we omit writing the elaborate equations and explanation for GLEU in this paper, but instead point the reader to Section 4 of Napoles et al. (2015) for the same. The formulation of GEC is quite similar to our formulation of style transfer in that style transfer involves making localized edits to the input sentence. Unlike BLEU, which takes only the target reference and the generated output into consideration, GLEU considers both of these as well as the source sentence too. It is a suitable metric for style transfer because it a) penalizes words of the source that were wrongly changed in the generated sentence, b) rewards words that were successfully changed and c) rewards those that were successfully retained from the source sentence to match those in the reference sentence. We use the implementation of GLEU[9] provided by Napoles et al. (2015).

Tables 4 and 5 show a comparison of automatic metrics between our models and previous models described earlier.

### 4.3 Result Analysis

From human evaluations in Tables 2 and 3, we see that our models (specifically, **B-GST**) outperform state-of-art systems by a good margin on almost all parameters as judged by humans, across all datasets. More importantly, as Table 6 shows, our models generate realistic and natural-sounding sentences while retaining core content - an aspect on which previous models seem to be seriously lacking. While our **G-GST** model does worse than **B-GST** due to a weak Retrieve mechanism, **G-GST** provides us a way to guide the generation and control attributes, making it more suitable for real-world applications after improving Retrieve in future. We find that metrics based on learned models - perplexity and accuracy, do not correlate entirely well with human evaluations, an observation also shared by Li et al. (2018). They are also heavily dependant on the distribution of data that they are trained on. A system that simply chooses a random sentence from the target training corpus as its output will score highly on both these metrics. For instance, the **BT** model in Table 5 has a high style but a considerably lower BLEU

[9]https://github.com/cnap/gec-ranking

| **Example #1** | **YELP (Positive to Negative)** | **YELP (Negative to Positive)** |
|---|---|---|
| SRC | i love this place , the service is always great ! | the store is dumpy looking and management needs to change . |
| SE | i love this place , the service is always great ! | the store is bought the building does n't deal . |
| D | i paid _num_ minutes before the gifted , not , a huge plus , n't . | the store is dumpy looking and management is fantastic and needs to change . |
| D&R | i did not like the homework of lasagna , not like it , . | the store is clean and well dumpy looking and management needs to change . |
| **G-GST** | **i used this place , the service is always awful !** | **the store is looking and management is excellent to .** |
| **B-GST** | **i hate this place , the service is always terrible !** | **the store is looking great and management to perfection .** |
| | | |
| **Example #2** | **AMAZON (Positive to Negative)** | **AMAZON (Negative to Positive)** |
| SRC | i finally made he purchase and am glad i did . | i m just looking forward to the day i get to replace it . |
| SE | i finally made he purchase and am glad i did . | i m just looking forward to the right away i get it for it . |
| D | i finally made it and was excited to purchase and am glad i did . | i m just looking forward to the day i get to replace my old one . |
| D&R | i finally made i will try another purchase and am glad i did . | looking forward to using it on turkey day ! . |
| **G-GST** | **i finally made he purchase and am embarrassed i smell pungent .** | **i m looking to the same day i get to use it .** |
| **B-GST** | **i finally made him purchase and am sorry i did .** | **i m looking forward to using the day i get to use it .** |
| | | |
| **Example #3** | **CAPTIONS (Factual to Romantic)** | **CAPTIONS (Factual to Humorous)** |
| SRC | people gather around a life size chess game . | three brown and black dogs are splashing in the water . |
| SE | people gather at a red wooden advertisement of players enjoy . | three small and brown dog are splashing in the water . |
| D | two young boys have working around a dream line and dream of childhood. | three black and brown dogs are sitting in the water to search of fish . |
| D&R | people gather around a carnival event , all determined to win the game . | two black and brown dogs are running in the water like a fish. |
| **G-GST** | **people gather around a life size chess game to celebrate life ' s happiness .** | **three brown and black dogs are splashing in the water talking to each other .** |
| **B-GST** | **two people gather around a life size chess game to celebrate life .** | **three brown and black dogs are splashing in the water looking for mermaids .** |
| | | |
| **Example #4** | **POLITICAL (Democrat to Republican)** | **POLITICAL (Republican to Democrat)** |
| SRC | thank you for your commitment to a strong public education system , senator ! | i absolutely agree with senator paul's actions . |
| BT | thanks for your vote for a balanced budget amendment , sir ! | i ' m merchandising with the rhetoric of senator warren . |
| **G-GST** | **thank you for your commitment to a strong conservative system , governor !** | **i absolutely agree with brian ' s dire actions . .** |
| **B-GST** | **thank you for your commitment to a strong constitutional system , senator scott !** | **i absolutely agree with elizabeth warren ' s actions .** |
| | | |
| **Example #5** | **GENDER (Male to Female)** | **GENDER (Female to Male)** |
| SRC | this is a spot that ' s making very solid food , with good quality product . | this a great place for a special date or to take someone from out of town . |
| **B-GST** | **this is a cute spot that ' s making me very happy , with good quality product .** | **this a great place for a bachelor or to meet someone from out of town .** |

Table 6: Examples of generated sentences to be compared down a column (B-GST and G-GST are our models, SRC is the input sentence). Attributes are colored.

score than **B-GST**. It is important therefore, to not consider them in isolation. Further, human reference sentences themselves score poorly using both these metrics as shown in these tables. Manual inspection of classifier accuracies shows that these classifiers give unreliable outputs that do not match human ratings. This is the case with the **CA** model in Table 2. Similar problems exist with regarding BLEU in isolation. A system that simply copies the source sentence will obtain high BLEU scores.

GLEU, however seems to strike a balance between target style match and content retention, as it takes the source, reference as well as predicted sentence into account. We see that GLEU scores also correlate with our own human evaluations as well as those of Li et al. (2018). While a detailed statistical correlation study is left for future work, the fact remains that GLEU is not susceptible to the weaknesses of other automatic metrics described above. Our uniformly state-of-art GLEU scores possibly indicate that we make only necessary edits to the source sentence.

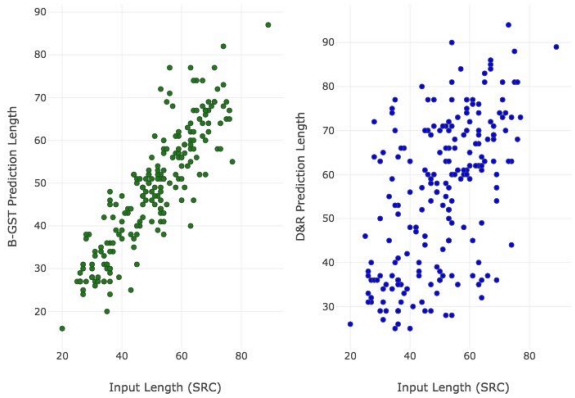


Figure 2: Correlation of B-GST (ours, left) with input sentence lengths vs D&R's (right) sentence lengths with input sentence lengths.

Keeping all the above considerations in mind, automatic metrics are still indicative and useful as they can be scaled to evaluate larger sets of models and datasets. From Tables 4 and 5, we see that we consistently outperform current state-of-art systems on BLEU. As shown by our high BLEU scores, one can conclude to some extent that our models retain non-stylistic parts well. Figure 2 shows that unlike the current state-of-art **D&R** model, the lengths of our generated sentences closely correlate with source sentence lengths. **B-GST** scores well on perplexity across datasets, a consistency that is not exhibited by any other model.

## 5 Related Work

One category of previous approaches is based on training adversarial networks to learn a latent representation of content and style. Shen et al. (2017) train a cross-aligned auto-encoder, with a shared content and separate style distribution. Hu et al. (2017) use VAEs with attribute discriminators to learn similar latent representations. This approach has been later encapsulated in encoder-decoder frameworks (Fu et al., 2018; John et al., 2018; Zhang et al., 2018a,b). Problems with these approaches have been discussed in the introduction.

Approaches that do not rely on a latent representation to separate content and attribute exist too. These include reinforcement learning based approaches (Xu et al., 2018; Gong et al., 2019), an unsupervised machine translation based approach (Subramanian et al., 2018) and the DRG approach (Li et al., 2018). The former two approaches suffer from sparsity and convergence issues and hence generate sentences of low-quality.

Previous approaches to use attention weights to extract attribute significance exist (Feng et al., 2018; Li et al., 2016; Globerson and Roweis, 2006), including the salience deletion method of Li et al. (2018) but they do not perform well on understanding sentence context while choosing attributes, and do not leverage the contextual capacity of a Transformer. Lastly, Dai et al. (2019) describe the use of Transformers for style transfer in an adversarial generator-discriminator setting, by adding an additional style embedding to the transformer. We are unable to do a comparitive study as they do not yet publish their code or outputs. The same is the case for Subramanian et al. (2018).

## 6 Conclusion

We propose the Generative Style Transformer that outperforms state-of-art systems on sentiment, gender and political slant. Our model leverages the DRG framework, massively pre-trained language models and the Transformer network itself.

## Acknowledgments

The authors of this paper would like to thank Swati Tiwari, Nishant Thakur and Akshit Mittal for their help with evaluation of results, and the anonymous reviewers for their suggestions and comments.

## References


Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 1171–1179, Cambridge, MA, USA. MIT Press.

Daniel Cer, Yinfei Yang, Sheng yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder.

Ning Dai, Jianze Liang, Xipeng Qiu, and Xuanjing Huang. 2019. Style transformer: Unpaired text style transfer without disentangled latent representation.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Shi Feng, Eric Wallace, Alvin Grissom II, Mohit Iyyer, Pedro Rodriguez, and Jordan Boyd-Graber. 2018. Pathologies of neural models make interpretations difficult. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3719–3728, Brussels, Belgium. Association for Computational Linguistics.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2018. Style transfer in text: Exploration and evaluation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Amir Globerson and Sam Roweis. 2006. Nightmare at test time: Robust learning by feature deletion. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 353–360, New York, NY, USA. ACM.

Hongyu Gong, Suma Bhat, Lingfei Wu, Jinjun Xiong, and Wen mei Hwu. 2019. Reinforcement learning based text style transfer without parallel training corpus.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1587–1596, International Convention Centre, Sydney, Australia. PMLR.

Vineet John, Lili Mou, Hareesh Bahuleyan, and Olga Vechtomova. 2018. Disentangled representation learning for non-parallel text style transfer.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics.

Jiwei Li, Will Monroe, and Daniel Jurafsky. 2016. Understanding neural networks through representation erasure. *CoRR*, abs/1612.08220.

Juncen Li, Robin Jia, He He, and Percy Liang. 2018. Delete, retrieve, generate: a simple approach to sentiment and style transfer. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1865–1874, New Orleans, Louisiana. Association for Computational Linguistics.

Remi Mir, Bjarke Felbo, Nick Obradovich, and Iyad Rahwan. 2019. Evaluating style transfer for text.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2015. Ground truth for grammatical error correction metrics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 588–593, Beijing, China. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W Black. 2018. Style transfer through back-translation. In *Proc. ACL*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training.

Sudha Rao and Joel Tetreault. 2018. Dear sir or madam, may I introduce the GYAFC dataset: Corpus, benchmarks and metrics for formality style transfer. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 129–140, New Orleans, Louisiana. Association for Computational Linguistics.

Sravana Reddy and Kevin Knight. 2016. Obfuscating gender in social media writing. In *Proceedings of the First Workshop on NLP and Computational Social Science*, pages 17–26, Austin, Texas. Association for Computational Linguistics.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in neural information processing systems*, pages 6830–6841.

Sandeep Subramanian, Guillaume Lample, Eric Michael Smith, Ludovic Denoyer, Marc'Aurelio Ranzato, and Y-Lan Boureau. 2018. Multiple-attribute text style transfer.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Jesse Vig. 2019. Visualizing attention in transformer-based language representation models.

Rob Voigt, David Jurgens, Vinodkumar Prabhakaran, Dan Jurafsky, and Yulia Tsvetkov. 2018. RtGender: A corpus for studying differential responses to gender. In *Proc. LREC*.

Ronald J. Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks.

Jingjing Xu, Xu SUN, Qi Zeng, Xiaodong Zhang, Xuancheng Ren, Houfeng Wang, and Wenjie Li. 2018. Unpaired sentiment-to-sentiment translation: A cycled reinforcement learning approach. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 979–988, Melbourne, Australia. Association for Computational Linguistics.

Cheng Yang, Maosong Sun, Xiaoyuan Yi, and Wenhao Li. 2018. Stylistic Chinese poetry generation via unsupervised style disentanglement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3960–3969, Brussels, Belgium. Association for Computational Linguistics.

Ye Zhang, Nan Ding, and Radu Soricut. 2018a. SHAPED: Shared-private encoder-decoder for text style adaptation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1528–1538, New Orleans, Louisiana. Association for Computational Linguistics.

Zhirui Zhang, Shuo Ren, Shujie Liu, Jianyong Wang, Peng Chen, Mu Li, Ming Zhou, and Enhong Chen. 2018b. Style transfer as unsupervised machine translation. *CoRR*, abs/1808.07894.

Ganbin Zhou, Ping Luo, Rongyu Cao, Fen Lin, Bo Chen, and Qing He. 2017. Mechanism-aware neural machine for dialogue response generation. In *Thirty-First AAAI Conference on Artificial Intelligence*.

# An Entity-Driven Framework for Abstractive Summarization

**Eva Sharma**[1*] **Luyang Huang**[2*] **Zhe Hu**[1*] and **Lu Wang**[1]
[1]Khoury College of Computer Sciences, Northeastern University, Boston, MA 02115
[2]Department of Electrical and Computer Engineering, Boston University, Boston, MA 02215
[1]evasharma@ccs.neu.edu, hu.zhe@husky.neu.edu, luwang@ccs.neu.edu
[2]lyhuang@bu.edu

## Abstract

Abstractive summarization systems aim to produce more coherent and concise summaries than their extractive counterparts. Popular neural models have achieved impressive results for single-document summarization, yet their outputs are often incoherent and unfaithful to the input. In this paper, we introduce **SENECA**, a novel System for ENtity-drivEn Coherent Abstractive summarization framework that leverages entity information to generate informative and coherent abstracts. Our framework takes a two-step approach: (1) an *entity-aware content selection* module first identifies salient sentences from the input, then (2) an *abstract generation* module conducts cross-sentence information compression and abstraction to generate the final summary, which is trained with rewards to promote coherence, conciseness, and clarity. The two components are further connected using reinforcement learning. Automatic evaluation shows that our model significantly outperforms previous state-of-the-art on ROUGE and our proposed coherence measures on New York Times and CNN/Daily Mail datasets. Human judges further rate our system summaries as more informative and coherent than those by popular summarization models.

**Input Article**:
. . . **Prime Minister Bertie Ahern** of Ireland called Sunday for a general election on May 24.
[**Mr. Ahern**] and his centrist party have governed in a coalition government since 1197 . . .
Under Irish law, which requires legislative elections every five years, **Mr. Ahern** had to call elections by midsummer.
On Sunday, {**he**} said **he** would base **his** campaign for reelection on **his** work to strengthen the economy and efforts to revive **Northern Ireland's** stalled peace process this year.
Political analysts said they expected **Mr. Ahern 's** work in **Northern Ireland** to be an asset . . .

**Human Summary**:
① **Prime Min Bertie Ahern** of Ireland calls for general election on May 24.
② [**He**] is required by law to call elections by midsummer.
③ Opinion polls suggest **his** centrist government is in danger of losing its majority in Parliament because of public disgruntlement about overburdened public services.
④ {**Ahern**} says **he** would base **his** campaign for reelection on **his** work to strengthen economy and **his** efforts to revive **Northern Ireland's** stalled peace process.
⑤ Analysts expect **his** work in **Northern Ireland** to be asset.

Figure 1: Sample summary of an article from the New York Times corpus (Sandhaus, 2008). Mentions of the same entity are colored. Underlined sentence in the article occurs relatively at an earlier position in the summary (②) to improve topical coherence. Mentions in brackets ("[]","{}") show different ways in which the same entity is referred to in the article and the summary. Detailed explanation is given in §1.

## 1 Introduction

Automatic abstractive summarization carries strong promise for producing concise and coherent summaries to facilitate quick information consumption (Luhn, 1958). Recent progress in neural abstractive summarization has shown end-to-end trained models (Nallapati et al., 2016; Tan et al., 2017a; Celikyilmaz et al., 2018; Kryściński et al., 2018) excelling at producing fluent summaries. Though encouraging, their outputs are frequently found to be unfaithful to the input and lack inter-sentence coherence (Cao et al., 2018; See et al., 2017; Wiseman et al., 2017). These observations suggest that existing methods have difficulty in identifying salient entities and related events in the article (Fan et al., 2018), and that existing model training objectives fail to guide the generation of coherent summaries.

In this paper, we present **SENECA**, a System for ENtity-drivEn Coherent Abstractive summarization.[1] We argue that entity-based modeling

[*] These authors contributed equally. Work done while LH was at Northeastern University.

[1]Our code is available at evasharma.github.io/SENECA.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3280–3291,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

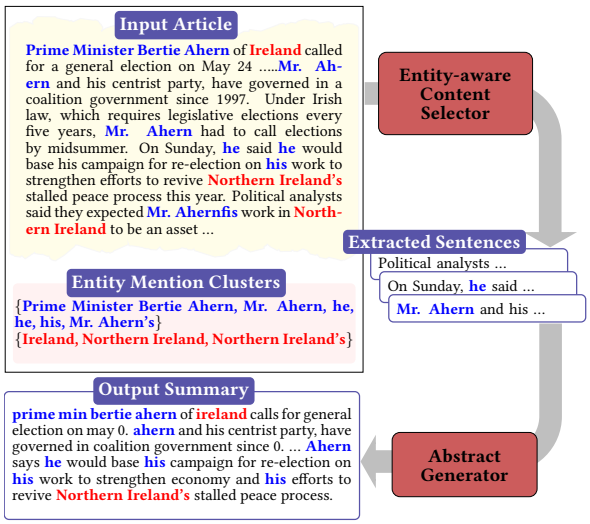


Figure 2: Our proposed entity-driven abstractive summarization framework. Entity-aware content selector extracts salient sentences and abstract generator produces informative and coherent summaries. Both components are connected using reinforcement learning.

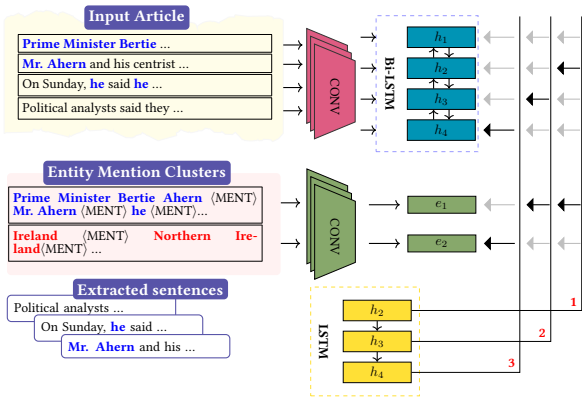


Figure 3: Our proposed entity-aware content selector. Arrows denote attention, with darker color representing higher weights.

enables enhanced *input text interpretation*, *salient content selection*, and *coherent summary generation*, three major challenges that need to be addressed by single-document summarization systems (Jones et al., 1999). We use a sample summary in Fig. 1 to show entity usage in summarization. Firstly, frequently mentioned entities from the input, along with their contextual information, underscores the salient content of the article (Nenkova, 2008). Secondly, as also discussed in prior work (Barzilay and Lapata, 2008; Siddharthan et al., 2011), patterns of entity distributions and how they are referred to contribute to the coherence and conciseness of the text. For instance, a human writer places the underlined sentence in the input article next to the first sentence in the summary to improve topical coherence as they are about the same topic ("elections"). Moreover, the human often optimizes on conciseness by referring to entities with pronouns (e.g., "he") or last names (e.g., "Ahern") without losing clarity.

We therefore propose a two-step neural abstractive summarization framework to emulate the way humans construct summaries with the goal of improving both informativeness and coherence of the generated abstracts. As shown in Fig. 2, an **entity-aware content selection** component first selects important sentences from the input that includes references to salient entities. An **abstract generation** component then produces coherent summaries by conducting cross-sentence information ordering, compression, and revision. Our abstract generator is trained using reinforcement learning with rewards that promote informativeness and optionally boost coherence, conciseness, and clarity of the summary. To the best of our knowledge, we are the first to study coherent abstractive summarization with the inclusion of linguistically-informed rewards.

We conduct both automatic and human evaluation on popular news summarization datasets. Experimental results show that our model yields significantly better ROUGE scores than previous state-of-the-art (Gehrmann et al., 2018; Celikyilmaz et al., 2018) as well as higher coherence scores on the New York Times and CNN/Daily Mail datasets. Human subjects also rate our system generated summaries as more informative and coherent than those of other popular summarization models.

## 2 Summarization Framework

In this section, we describe our entity-driven abstractive summarization framework which follows a two-step approach as shown in Fig. 2. It comprises of (1) an *entity-aware content selection component*, that leverages entity guidance to select salient sentences (§2.1), and (2) an *abstract generation component* (§2.2), that is trained with reinforcement learning to generate coherent and concise summaries (§2.3). Finally, we describe how the two components are connected to further improve the generated summaries (§2.4).

### 2.1 Entity-Aware Content Selection

We design our content selection component to capture the interaction between entity mentions and the input article. Our model learns to identify salient content by aligning entity mentions

and their contexts with human summaries. Concretely, we employ two encoders: one learns entity representations by encoding their mention clusters and the other learns sentence representations. A pointer-network-based decoder (Vinyals et al., 2015b) selects a sequence of important sentences by jointly attending to the entities and the input, as depicted in Fig. 3.

**Entity Encoder.** We run off-the-shelf coreference resolution system from Stanford CoreNLP (Manning et al., 2014) on the input articles to extract entities, each represented as a cluster of mentions. Specifically, from each input article, we extract the coreferenced entities, and construct the mention clusters for all the mentions of each entity in that article. We also consider non coreferenced entity mentions as singleton entity mention clusters. Among all these mention clusters, for our experiments, we only consider salient entity mention clusters. We label clusters as "salient" based on two rules: (1) mention clusters with entities appearing in the first three sentences of the article, and (2) top $k$ clusters containing most numbers of mentions. We experimented with different values of k and found that $k = 6$ gives us the best set of salient mention clusters having an optimal overlap with entity mentions in the ground truth summary.

For each mention cluster, we concatenate mentions of the same entity as they occur in the input into one sequence, segmented with special tokens (<MENT>). Finally, we get entity representations $\mathbf{e}_i$ for the $i$-th entity by encoding each cluster via a temporal convolutional model (Kim, 2014).

**Input Article Encoder.** For article encoding, we first learn sentence representations $\mathbf{r}_j$ by encoding words in the $j$-th sentence with another temporal convolutional model. Then, we utilize a bidirectional LSTM (biLSTM) to aggregate sentences into a sequence of hidden states $\boldsymbol{h}_j$. Both the encoders use a shared word embedding matrix to allow better alignment.

**Sentence Selection Decoder.** We employ a single-layer unidirectional LSTM with hidden states $\mathbf{s}_t$ to recurrently extract salient sentences. At each time step $t$, we first compute an entity context vector $\mathbf{c}_t^e$ based on attention mechanism (Bahdanau et al., 2014):

$$\mathbf{c}_t^e = \sum_i a_{it}^e \mathbf{e}_i \quad (1)$$

$$\mathbf{a}_t^e = \text{softmax}(\mathbf{v}^e \tanh(\mathbf{W}^{e1}\mathbf{s}_t + \mathbf{W}^{e2}\mathbf{e}_i)) \quad (2)$$

where $\mathbf{a}_t^e$ are attention weights, $\mathbf{v}^*$ and $\mathbf{W}^{**}$ denote trainable parameters throughout the paper. Bias terms are omitted for simplicity. We further use a *glimpse operation* (Vinyals et al., 2015a) to compute a sentence context vector $\mathbf{c_t}$ as follows:

$$\mathbf{c}_t = \sum_j a_{jt}^h \mathbf{W}^{h2}\mathbf{h}_j \quad (3)$$

$$\mathbf{a}_t^h = \text{softmax}(\mathbf{v}^h \tanh(\mathbf{W}^{h1}\mathbf{s}_t + \mathbf{W}^{h2}\mathbf{h}_j)) \quad (4)$$

where $\mathbf{a}_t^h$ are attention weights. Finally, sentence extraction probabilities that consider both entity and input context are calculated as:

$$p(y_t^l | y_{1:t-1}^l) = \text{softmax}(\mathbf{v}^q \tanh(\mathbf{W}^{p1}\mathbf{s}_t + \mathbf{W}^{p2}\mathbf{c_t} + \mathbf{W}^{p3}\mathbf{c_t^e})) \quad (5)$$

where the sentence $y_t^l$ with the highest probability is selected. The process stops when the model picks the end-of-selection token.

**Selection Label Construction.** We train our content selection component with a cross-entropy loss: $-\sum_{(\mathbf{y}^l,\mathbf{x})\in D} \log p(\mathbf{y}^l \mid \mathbf{x}; \theta)$, here $\mathbf{y}^l$ are the ground truth sentence selection labels and $\mathbf{x}$ is the input article. $\theta$ denotes all model parameters.

To acquire training labels for sentence selection, we collect positive sentences in the following way. First, we employ greedy search to select the best combination of sentences that maximizes ROUGE-2 F1 (Lin and Hovy, 2003) with reference to human summary, as described by Zhou et al. (2018). We further include sentences whose ROUGE-L recall is above 0.5 when each is compared with its best aligned summary sentence. In cases where no sentence is selected, we label the first two sentences from the article as positive. Our combined construction strategy selects an average of 2.96 and 3.18 sentences from New York Times and CNN/Daily Mail articles respectively.

## 2.2 Abstract Generation with Reinforcement Learning

Our abstract generation component takes the selected sentences as input and produces the final summary. This abstract generator is a sequence-to-sequence network with attention over input (Bahdanau et al., 2014). The copying mechanism from See et al. (2017) is adopted to allow out-of-vocabulary words to appear in the output.

The abstract generator is first trained with maximum likelihood (ML) loss followed by additional training with policy-based reinforcement learning

(RL). For ML training, we use teacher forcing algorithm (Williams and Zipser, 1995), to minimize the following loss:

$$\mathcal{L}_{\text{ml}} = -\sum_{(\mathbf{y},\mathbf{x}^{ext})\in D} \log p(\mathbf{y}\,|\,\mathbf{x}^{ext};\theta) \quad (6)$$

where $D$ is the training set, $\mathbf{x}^{ext}$ are extracted sentences from our label construction.

**Self-Critical Learning.** Following Paulus et al. (2017), we use the self-critical training algorithm based on policy gradients to use discrete metrics as RL rewards. At each training step, we generate two summaries: a *sampled* summary $\mathbf{y}^s$, obtained by sampling words from the probability distribution $p(\mathbf{y}^s|\,\mathbf{x}^{ext};\theta)$ at each decoding step, and a *self-critical baseline* summary $\hat{\mathbf{y}}$, yielded by greedily selecting words that maximize the output probability at each time step (Rennie et al., 2017). We then calculate rewards based on the average of ROUGE-L F1 and ROUGE-2 F1 of the two summaries against that of the ground-truth summary, and define the following loss function:

$$\mathcal{L}_{\text{rl}} = -\frac{1}{N'}\sum_{(\mathbf{y}^s,\mathbf{x}^{ext})\in D'} (\mathbf{R}(\mathbf{y}^s) - \mathbf{R}(\hat{\mathbf{y}}))\log p(\mathbf{y}^s|\,\mathbf{x}^{ext};\theta) \quad (7)$$

where $D'$ represents set of sampled summaries paired with extracted input sentences and $N'$ represents the total number of sampled summaries. $\mathbf{R}(\mathbf{y}) = \mathbf{R}_{\text{Rouge}}(\mathbf{y}) = \frac{1}{2}\big(\mathbf{R}_{\text{Rouge-L}}(\mathbf{y}) + \mathbf{R}_{\text{Rouge-2}}(\mathbf{y})\big)$, is the overall ROUGE reward for a summary $\mathbf{y}$.

### 2.3 Rewards with Coherence and Linguistic Quality

So far, we have described the *two basic components* of our SENECA framework. As noted in prior work (Liu et al., 2016), optimizing for an ngram-based metric like ROUGE does not guarantee improvement over readability of the generations. We thus augment our framework with additional rewards based on *coherence* and *linguistic quality* as described below.

**Entity-Based Coherence Reward** ($\mathbf{R}_{\text{Coh}}$). We use a separately trained **coherence model** to score summaries and guide our abstract generator to produce more coherent outputs by adding a reward $\mathbf{R}_{\text{Coh}}$ in the aforementioned RL training process. The new reward takes the following form:

$$\mathbf{R}(\mathbf{y}) = \mathbf{R}_{\text{Rouge}}(\mathbf{y}) + \gamma_{\text{Coh}}\mathbf{R}_{\text{Coh}}(\mathbf{y}) \quad (8)$$

Here we show how to calculate $\mathbf{R}_{\text{Coh}}$, to capture both entity distribution patterns and topical continuity. Since summaries are short, (e.g. 2.0 sentences on average per summary in the New York Times data), we decide to build our coherence model on top of local coherence estimation for pairwise sentences. We adopt the architecture of neural coherence model developed by Wu and Hu (2018), but train it with samples that enable coherence modeling based on entity presence and their context. Here we briefly describe the model, and refer the readers to the original paper for details.

Given a pair of sentences $S_A$ and $S_B$, convolution layers first transform them into hidden representations, from which a multi-layer perceptron is utilized to yield a coherence score $\mathbf{Coh}(S_A, S_B) \in [-1, 1]$. We train the model with hinge-loss by leveraging both coherent positive samples and incoherent negative samples:

$$\mathcal{L}(S_A, S_B^+, S_B^-) = \max\{0, 1 + \mathbf{Coh}(S_A, S_B^+) - \mathbf{Coh}(S_A, S_B^-)\} \quad (9)$$

where $S_A$ is a target sentence, $S_A$ and $S_B^+$ is a positive pair, and $S_A$ and $S_B^-$ is a negative pair.

Note that Wu and Hu (2018) only consider *position* information for training data construction, i.e., $S_A$ and $S_B^+$ must be adjacent, and $S_A$ and $S_B^-$ are at most 9 sentences away with $S_B^-$ randomly picked. We instead introduce two notable features to construct our training data. In addition to being adjacent, we further constrain $S_A$ and $S_B^+$ to have at least one coreferred entity and that $S_B^-$ does not. Since our initial experiments show that coherence model trained in this manner cannot discern pure repetition of sentences, e.g., simply duplicating words leads to higher coherence, we reuse the target sentences themselves as the negative pairs.

Finally, since this model outputs pairwise coherence scores, for a summary containing more than two sentences, we use the average of all adjacent sentence pairs' scores as the final summary coherence score. Summaries containing only one sentence get 0 coherence score. We also conduct correlation study to show average aggregation works reasonably well (details in Supplementary).

**Linguistic Quality Rewards** ($\mathbf{R}_{\text{Ref}}$ & $\mathbf{R}_{\text{App}}$). We further consider two linguistically-informed rewards to further improve summary clarity and conciseness by penalizing (1) improper usage of referential pronouns, and (2) redundancy introduced by non-restrictive appositives and relative clauses.

*Pronominal Referential Clarity.* Referential pronouns occurring without the antecedents in a summary decreases its readability. For instance, a text with a pronoun "*they*" occurring before the required referred entity is introduced, would be less comprehensible. Therefore, at the RL step, we either penalize a summary with a reward of $-1$ for such improper usage, or give $0$ otherwise. In our implementation, we define improper usage as the presence of a third personal pronoun or a possessive pronoun before any noun phrase occurs. The new reward is written as $\mathbf{R}(\mathbf{y}) = \mathbf{R}_{\text{Rouge}}(\mathbf{y}) + \gamma_{\text{Ref}}\mathbf{R}_{\text{Ref}}(\mathbf{y})$.

*Apposition.* Next, we consider a reward to teach the model to use apposition and relative clause minimally, which improves summary conciseness. For this, we focus on the non-restrictive appositives and relative clauses, which often carry non-critical information (Conroy et al., 2006; Wang et al., 2013) and can be automatically detected based on comma usage patterns. Specifically, a sentence contains a non-restrictive appositive if i) it contains two commas, and ii) the word after first comma is a possessive pronoun or a determinant (Geva et al., 2019). We penalize a summary with $-1$ for using non-restrictive appositives and relative clauses, henceforth referred to as apposition, or give $0$ otherwise. Similarly, we have the total reward as $\mathbf{R}(\mathbf{y}) = \mathbf{R}_{\text{Rouge}}(\mathbf{y}) + \gamma_{\text{App}}\mathbf{R}_{\text{App}}(\mathbf{y})$.

### 2.4 Connecting Selection and Abstraction

Our entity-aware content selection component extracts salient sentences whereas our abstract generation component compresses and paraphrases them. Until this point, they are trained separately without any form of parameter sharing. We add an *additional step to connect these two networks* by training them together via the self-critical learning algorithm based on policy gradient (the same methodology as in §2.2).

Following the Markov Decision Process formulation, at each time step $t$, our content selector generates a set of extracted sentences ($\hat{\mathbf{x}}^{ext}$) from an input article. Our abstract generator uses $\hat{\mathbf{x}}^{ext}$ to generate a summary. This summary, evaluated against the respective human summary, receives ROUGE-1 as reward (See Eq. (7)). Note that the abstract generator, that has been previously trained with average of ROUGE-L and ROUGE-2 as reward to promote fluency, is not updated during this step. In this extra stage, if our content selector accurately selects salient sentences, the abstract generator is more likely to produce a high-quality summary, and such action will be encouraged. Whereas, action resulting in inferior selections will be discouraged.

## 3 Experimental Setups

**Datasets and Preprocessing.** We evaluated our models on two popular summarization datasets: New York Times (NYT) (Sandhaus, 2008) and CNN/Daily Mail (CNN/DM) (Hermann et al., 2015). For NYT, we followed the preprocessing steps by Paulus et al. (2017) to obtain similar training $(588,909)$, validation $(32,716)$ and test $(32,703)$ samples. Here, we additionally replaced author names with a special token. For CNN/DM, we followed the preprocessing steps in See et al. (2017), obtaining $287,188$ training, $13,367$ validation and $11,490$ testing samples.

For training our coherence model for CNN/DM, we used $890,419$ triples constructed from summaries and input articles sampled from the CNN/DM training set. Similarly for NYT, we sampled $884,494$ triples from NYT training set. For the validation and test set for the two models, we sampled roughly $10\%$ from the validation and test set of the respective datasets. Our coherence model for CNN/DM achieves 86% accuracy and for NYT, 84%. Additional evaluation for this model is reported in §4.1.

**Training Details and Parameters.** We used a vocabulary of $50K$ most common words in the training set (See et al., 2017), with 128-dimensional word embeddings randomly initialized and updated during training. In the content selection component, for both entity and sentence encoders, we implemented one-layer convolutional network with 100 dimensions and used a shared embedding matrix between the two. We employed LSTM models with 256-dimensional hidden states for the input article encoder (per direction) and the content selection decoder (Chen and Bansal, 2018). We used a similar setup for the abstract generator encoder and decoder. During ML training of both components, Adam (Kingma and Ba, 2015) is applied with a learning rate $0.001$ and a gradient clipping 2.0, and the batch size 32. During RL stage, we reduced learning rate to $0.0001$ (Paulus et al., 2017) and set batch size to 50. For our abstract generator, to reduce variance during RL training,

we sampled 5 sequences for each data point and took an average over these samples using batch size 10. We set $\gamma_{\text{Coh}} = 0.01$, $\gamma_{\text{Ref}} = 0.005$, $\gamma_{\text{App}} = 0.005$ for NYT and CNN/DM with grid search on validation set.

During inference, we adopted the trigram repetition avoidance strategy (Paulus et al., 2017; Chen and Bansal, 2018), with additional length normalization to encourage the generation of longer sequences as in (Gehrmann et al., 2018).

**Baselines and Comparisons.** Besides baseline LEAD-3, we compare with popular and existing state-of-the-art abstractive summarization models on NYT and CNN/DM datasets: (1) pointer-generator model with coverage (See et al., 2017) (POINTGEN+COV); (2) sentence rewriting model (Chen and Bansal, 2018) (SENTREWRITE); (3) RL-based abstractive summarization (Paulus et al., 2017) (DEEPREINFORCE); (4) bottom-up abstraction (Gehrmann et al., 2018) (BOTTOMUP); and (5) deep communicating agents-based summarization (Celikyilmaz et al., 2018) (DCA).

We show comparison models' results reported as in the original publications. For significant tests, we run code released by the authors of POINTGEN+COV and SENTREWRITE, and by our implementation of DEEPREINFORCE on both datasets for training and testing. Since we do not have access to model outputs from Paulus et al. (2017), we re-implement their model, and achieve comparable ROUGE scores (e.g. on NYT, our ROUGE-1,2,L are 46.61, 29.76, and 43.46). For BOTTOMUP, we obtain model outputs from the authors for both CNN/DM and NYT datasets, whereas for DCA, we acquire summaries only for CNN/DM dataset.

In addition to SENECA base model, which combines entity-aware content selection and RL-based abstract generation (with average of ROUGE-L F1 and ROUGE-2 F1 as reward), we also report results on its variants with additional rewards during abstract generator training. We further consider SENECA (i) without entities, and (ii) end-to-end trained but with sentence selection only, i.e., the abstract generator simply repeats the input.

## 4 Results

In §4.1, we first evaluate our entity-based coherence model, which produces the coherence reward ($R_{\text{Coh}}$). We then present automatic evaluation for summarization models on content, coherence, and linguistic quality (§4.2). We further discuss human evaluation results in §4.3.

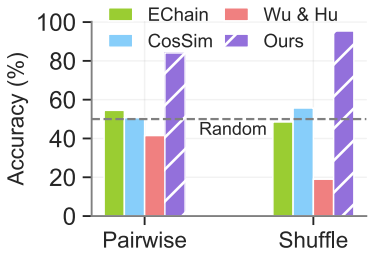

Figure 4: Accuracy of our coherence model compared to different baselines and Wu and Hu (2018) on PAIRWISE and SHUFFLE test sets.

### 4.1 Coherence Model Evaluation

We evaluate our entity-based coherence model on two tasks constructed from NYT test set: PAIRWISE and SHUFFLE. PAIRWISE is constructed as described in §2.2 with equal number of positive pairs and negative pairs. SHUFFLE comprises of full summaries, where half are original summaries, and the other half contain their shuffled version.

In Fig. 4, we show a comparison of our model against a version trained based on the same amount of samples constructed as done by Wu and Hu (2018). Also shown are two baselines: ECHAIN, that labels a pair of sentences as more coherent if they have one or more entity mentions coreferred, and COSSIM, that labels a pair of sentences with higher cosine similarity as more coherent. Our model yields significantly higher accuracy (greater than 80%) on both tasks than the comparisons, which is due to the difference in training data construction. Wu and Hu (2018) only consider position information, whereas we capture entity-based coherence. The improvement in performance of the coherence model indicates the effectiveness of our training data construction in capturing multiple aspects of coherence.

### 4.2 Automatic Summary Evaluation

**Results on NYT.** We first report the new state-of-the-art results for ROUGE-2 and ROUGE-L (Lin and Hovy, 2003), where our models outperform the previous best performing model DCA. Our SENECA models also outperform all comparisons on coherence score. This indicates that our models' summaries not only contain more salient information but are also more coherent.

| System | R-1 | R-2 | R-L | COH. |
|---|---|---|---|---|
| HUMAN | - | - | - | 0.79 |
| LEAD-3 | 32.59 | 16.49 | 29.17 | - |
| POINTGEN+COV† | 41.06 | 25.71 | 37.28 | 0.63 |
| SENTREWRITE† | 42.19 | 25.42 | 38.74 | 0.32 |
| DEEPREINFORCE* † | 47.03 | 30.72 | 43.10 | 0.59 |
| BOTTOMUP* | 47.38 | 31.23 | 41.81 | 0.56 |
| DCA* | **48.08** | 31.19 | 42.33 | - |
| **Our Models** | | | | |
| SENECA | 47.94 | ***31.77*** | 44.34 | 0.73 |
| Input w/o Entity | 47.04 | 30.94 | 43.64 | 0.06 |
| Sentence Selection Only | 39.97 | 22.49 | 35.68 | 0.30 |
| + $R_{\text{Coh}}$ | 47.57 | 31.28 | 44.03 | 0.75 |
| + $R_{\text{Ref}}$ | 47.57 | 31.22 | 43.92 | 0.70 |
| + $R_{\text{App}}$ | *48.05* | 31.71 | ***44.60*** | 0.69 |
| + $R_{\text{Coh}}$+$R_{\text{Ref}}$+$R_{\text{App}}$ | 47.52 | 31.25 | 44.01 | **0.76** |

Table 1: Results on NYT. Best results per metric are in bold. Best of our models are in italics. SENECA yields significantly higher R-1,2 and L than all comparisons except for BOTTOMUP and DCA[2](approximate randomization test ($p < 0.0005$) (Edgington, 1969)). For COH., our best model is also significantly better (Welch's $t$-test, $p < 0.05$). *: scores taken from original paper. †: significant test done on outputs by running code released by their authors or our implementation.

| System | R-1 | R-2 | R-L | COH. |
|---|---|---|---|---|
| HUMAN | - | - | - | 0.55 |
| LEAD-3 | 40.23 | 17.52 | 36.34 | - |
| POINTGEN+COV* † | 39.53 | 17.28 | 36.38 | 0.19 |
| SENTREWRITE* † | 40.88 | 17.80 | 38.54 | 0.41 |
| DEEPREINFORCE* † | 41.16 | 15.75 | **39.08** | -0.21 |
| BOTTOM-UP* † | 41.22 | 18.68 | 38.34 | 0.47 |
| DCA † | 40.72 | **19.02** | 37.85 | 0.30 |
| **Our Models** | | | | |
| SENECA | ***41.52*** | 18.36 | 38.09 | -0.12 |
| Input w/o Entity | 38.18 | 16.98 | 34.20 | -0.15 |
| Sentence Selection Only | 41.43 | *18.84* | 37.73 | -0.06 |
| + $R_{\text{Coh}}$ | 41.04 | 16.98 | 38.08 | 0.52 |
| + $R_{\text{Ref}}$ | 41.35 | 16.98 | *38.25* | -0.30 |
| + $R_{\text{App}}$ | 41.38 | 17.22 | *38.43* | 0.00 |
| + $R_{\text{Coh}}$+$R_{\text{Ref}}$+$R_{\text{App}}$ | 40.71 | 16.68 | 38.07 | **0.63** |

Table 2: Results on CNN/Daily Mail. Our best results are achieved by the variant with sentence selection only, yielding significantly better R-1 and R-2 than all comparisons except BOTTOM-UP and DCA (Welch's $t$-test, $p < 0.05$). For COH., our best model is also significantly better (Welch's $t$-test, $p < 0.05$). We re-evaluate DCA output against full length human summaries. *: scores taken from original paper. †: significant test done on outputs by running code released by their authors or our implementation.

Amongst our models, the base SENECA model reports higher ROUGE and coherence score (0.73) compared to the version without entity as input. This demonstrates the importance of entity guidance during content selection as well as abstract generation. SENECA model trained with Apposition reward ($R_{\text{App}}$) reports the highest ROUGE-L (44.60), but shows a drop in the coherence score to 0.69. Whereas, SENECA with all rewards $R_{\text{Coh}}$+$R_{\text{Ref}}$+$R_{\text{App}}$ reports the highest coherence score of 0.76 and a drop in ROUGE-L (44.01).

**Results on CNN/DM.** Since CNN/DM summaries are more extractive than that of NYT (Grusky et al., 2018), all SENECA models produce higher ROUGE-1 scores with SENECA base model achieving the highest, outperforming the previous best performing models on ROUGE-1. We also report the highest coherence score (0.63), which is even higher than that reported on human summaries (0.55). Since CNN/DM gold summaries are comprised of concatenated human written highlights for input articles, they are about the same topic and are cohesive, but lack entity-based coherent structure, for instance fewer entities get coreferred in subsequent sentences. Therefore, our coherence evaluation, which tests for entity-based coherence, gives lower coherence score to CNN/DM gold summaries.

Additionally for CNN/DM, the generated summaries sometimes contain less relevant words, e.g. stopwords, at the end of the summaries. This effect however improves the ROUGE scores while adversely affecting the coherence scores of the summaries. Training with additional coherence reward alleviates the issue by appending additional sentences, thereby improving overall coherence.

Amongst our models, we observe the overall trend to be similar to that in NYT results. Our base SENECA model reports higher ROUGE and coherence score compared to SENECA without entity input, again, indicating the usefulness of entity information. SENECA model with all rewards also yields the highest coherence score of 0.63, whereas SENECA model with coherence reward performs considerably better on ROUGE-L with a drop in coherence score.

**Linguistic Quality Evaluation.** We further do a preliminary evaluation of system summaries on two important linguistic qualities: clarity and conciseness. To measure *clarity*, we focus on the percentage of summaries that improperly use referential pronouns (Ref.), defined as third person pro-

[2]Significant test was not performed with BOTTOMUP as they use different test split than Paulus et al. (2017), and with DCA, since their outputs are unavailable on NYT.

| System | NYT | | | CNN/DM | | |
|---|---|---|---|---|---|---|
| | Ref. | RelCl. | App. | Ref. | RelCl. | App. |
| HUMAN | 0.11 | 6.67 | 4.83 | 0.13 | 2.80 | 1.12 |
| POINTGEN + COV | 0.15 | 1.92 | 1.26 | 0.18 | 0.63 | 0.75 |
| SENTREWRITE | 0.13 | 0.75 | 0.68 | 0.11 | 0.28 | 0.19 |
| DEEPREINFORCE | 0.18 | 0.40 | 0.50 | 0.19 | 0.45 | 0.10 |
| BOTTOMUP | 0.12 | 0.62 | 0.54 | 0.10 | 0.23 | 0.09 |
| DCA | - | - | - | 0.12 | 0.26 | 0.13 |
| SENECA | 0.13 | 1.15 | 0.68 | 0.10 | 0.22 | 0.06 |
| w/o entity | 0.21 | 1.24 | 0.70 | 0.11 | 0.33 | 0.10 |
| + $R_{\text{Coh}}$ | 0.12 | 1.15 | 0.71 | **0.06** | 0.07 | 0.05 |
| + $R_{\text{Ref}}$ | **0.10** | 1.20 | 0.72 | 0.07 | 0.11 | 0.04 |
| + $R_{\text{App}}$ | 0.13 | 0.65 | **0.42** | 0.09 | **0.01** | **0.01** |
| + $R_{\text{Coh}}$+$R_{\text{Ref}}$+$R_{\text{App}}$ | 0.12 | 0.94 | 0.59 | 0.07 | 0.04 | **0.01** |

Table 3: % of system summaries improperly using referential pronouns (Ref.), or containing relative clauses (RelCl.) or appositives (App.) (lowest values in bold). Lower values indicate better performance.

nouns or possessive pronouns being used before any noun phrase. Similarly, to measure *conciseness*, we report how often summaries contain at least one non-restrictive relative clause (RelCl.) or non-restrictive appositives (App.). For model summaries, we report these measures in reference to the respective gold summaries. Lower values are better.

As can be seen from Table 3, our models report the least percentage of improper usage of referential pronouns. Particularly on NYT, the model trained with $R_{\text{Ref}}$ reward makes much fewer mistakes in this category. Similarly, adding $R_{\text{App}}$ reports the least amount of usage of relative clause or apposition, making summaries easier to read.

### 4.3 Human Evaluation

Human evaluation is conducted to analyze the informativeness and readability of the summaries generated by our models. We randomly select 30 articles from the NYT test set and ask three proficient English speakers to rate summaries generated by POINTGEN+COV, DEEPREINFORCE (Paulus et al., 2017), our SENECA, and SENECA + $R_{\text{Coh}}$,[3] along with HUMAN written summaries. Each rater reads the article and scores the summaries against each other on a Likert scale of 1 (worst) to 5 (best) on the following three aspects: **informativeness**—whether summary covers salient points from the input, **grammaticality**, and **coherence**—whether summary presents content and entity mentions in coherent order. Detailed guidelines with sample ratings and explanation are shown in the Supplementary.

[3]To clearly discern the improvement in coherence in the summaries after adding coherence reward without introducing effects of additional rewards, we did not consider SENECA model with all rewards for human evaluation.

| | Inf. | Gram. | Coh. |
|---|---|---|---|
| HUMAN | 4.58 | 4.42 | 4.16 |
| POINTGEN+COV | 3.63 | **4.47** | 3.89 |
| DEEPREINFORCE | 3.63 | 3.21 | 3.16 |
| SENECA | 4.26 | 4.11 | 4.05 |
| SENECA + $R_{\text{Coh}}$ | **4.32**$^*$ | 4.10 | **4.11**$^*$ |

Table 4: Human evaluation on informativeness (Inf.), grammaticality (Gram.) and coherence (Coh.). Best results among systems are in bold, with significance reported over other comparisons with $*$ (approximate randomization test (Edgington, 1969), $p < 0.0005$). The Krippendorf's $\alpha$ for the three aspects are 0.46, 0.64, and 0.48.

Table 4 shows that our model SENECA + $R_{\text{Coh}}$ ranks significantly higher on informativeness as well as coherence, reaffirming our observations from automatic evaluation. Surprisingly, SENECA + $R_{\text{Coh}}$ ranks higher on informativeness even when compared to SENECA, which reports higher on ROUGE (see Table 1). Through manual inspection, we find that SENECA model often generates summaries whose second or third sentence misses the subject, whereas SENECA + $R_{\text{Coh}}$ tends to avoid this problem. Though not significant, POINTGEN+COV ranks higher on grammaticality than SENECA + $R_{\text{Coh}}$. We believe this is due to the fact that SENECA + $R_{\text{Coh}}$ learns to merge sentences from the input article while making some grammatical errors. We further show sample summaries in Figure 5.

## 5 Related Work

**Neural Abstractive Summarization.** Two-step abstractive summarization approaches have become popular in recent years, where the two steps, content selection and abstraction, are conveniently separated from each other. In these approaches, salient content is first identified, usually at sentence-level (Hsu et al., 2018; Li et al., 2018; Tan et al., 2017b; Chen and Bansal, 2018) or phrase-level (Gehrmann et al., 2018), followed by abstract generation. However, prior work mainly focuses on improving the informativeness of abstractive summaries, e.g. copying and coverage mechanisms (See et al., 2017), and reinforcement learning methods optimizing on ROUGE scores (Paulus et al., 2017). Coherence and other aspects of linguistic quality that capture the overall readability of summaries are largely ignored.

**Human**:
New Jersey Legislature recommends 0 ways to overhaul system that has produced highest property taxes in nation; plan includes 0 percent reduction in property taxes to most homeowners through direct tax credits; **will place annual limit on property tax increases; will revise financing of education and end special financing of state's poor districts**

**PointGen+cov**:
new jersey legislature, after more than three decades of complaints about soaring property taxes and three months of hearings about ways to reduce them, designed to overhaul system that has produced highest property taxes in nation. recommendations included 0 percent reduction in property taxes to most of state's homeowners in form of direct tax credits

**DeepReinforce**:
new jersey legislature, after more than three decades of complaints about soaring property taxes and three months of hearings about ways to reduce them. unveils 0 proposals designed to overhaul system that has highest property taxes in nation. recommendations include 0 percent reduction in property taxes to most of state's and of direct tax credits

**SENECA**:
new jersey legislature unveils 0 proposals designed to overhaul system that has produced highest property taxes in nation. recommendations included 0 percent reduction in property taxes to most of state's homeowners in form of direct tax credits

**SENECA + $\mathbf{R}_{\mathrm{Coh}}$**:
new jersey legislature, unveiled 0 proposals designed to overhaul system that has highest property taxes in nation. recommendations included 0 percent reduction in property taxes to most of state's homeowners in form of direct tax credits. **other parts of plan would place limit on annual property tax increases and revise way state pays for public education, ending special financing given to state's poor districts**

Figure 5: Sample summaries for an NYT article. Our model with coherence reward overlaps the most with human summary (**green** is ours, **blue** denotes human).

In this work, besides informativeness, we also aim to improve upon these aspects of summaries.

**Role of Entities and Coherence in Summarization.** Entities in a text carry useful contextual information (Nenkova, 2008) and therefore play an important role in multi-document summarization (Li et al., 2006) and event summarization for selecting salient sentences (Li et al., 2015). Moreover, entity mentions connecting sentences have also been used to extract non-adjacent yet coherent sentences (Siddharthan et al., 2011; Parveen et al., 2016). For abstractive summarization, Amplayo et al. (2018) find it beneficial to leverage entities that are linked to existing knowledge bases. Unfortunately, it fails to capture the entities that do not exist in these knowledge bases.

Grammar role-based entity transitions have been widely employed to model coherence in text generation tasks (Barzilay and Lee, 2004; Lapata and Barzilay, 2005; Barzilay and Lapata, 2008; Guinaudeau and Strube, 2013; Tien Nguyen and Joty, 2017), which often requires intensive feature engineering. Neural coherence models (Mesgar and Strube, 2018; Li and Hovy, 2014) have, therefore, gained popularity due to their end-to-end nature. However, coherence has mainly been investigated in extractive summarization systems (Alonso i Alemany and Fuentes Fort, 2003; Christensen et al., 2013; Parveen et al., 2015; Wu and Hu, 2018). To the best of our knowledge, we are the first to leverage entity information to improve coherence for neural abstractive summarization along with other important linguistic qualities.

## 6 Conclusion

We present an entity-driven summarization framework to generate informative and coherent abstractive summaries. An entity-aware content selector chooses salient sentences from the input article and an abstract generator produces a coherent abstract. Linguistically-informed guidance further enhances conciseness and clarity, thus improving the summary quality. Our model obtains the new state-of-the-art ROUGE scores on the NYT and CNN/DM datasets. Human evaluation further indicates that our system produces more coherent summaries than other popular methods.

## Acknowledgements

This research is supported in part by National Science Foundation through Grants IIS-1566382 and IIS-1813341, and by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via contract # FA8650-17-C-9116. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein. We thank the anonymous reviewers for their constructive suggestions. We also thank Sebastian Sehrmann for sharing their outputs on NYT and CNN/DM and Asli Celikyilmaz for sharing their summaries on CNN/DM.

## References


Laura Alonso i Alemany and Maria Fuentes Fort. 2003. Integrating cohesion and coherence for automatic summarization. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 2*, pages 1–8. Association for Computational Linguistics.

Reinald Kim Amplayo, Seonjae Lim, and Seung-won Hwang. 2018. Entity commonsense representation for neural abstractive summarization. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 697–707.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Regina Barzilay and Mirella Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34.

Regina Barzilay and Lillian Lee. 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *HLT-NAACL 2004: Main Proceedings*, pages 113–120, Boston, Massachusetts, USA. Association for Computational Linguistics.

Ziqiang Cao, Furu Wei, Wenjie Li, and Sujian Li. 2018. Faithful to the original: Fact aware neural abstractive summarization. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI)*.

Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep communicating agents for abstractive summarization. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1662–1675.

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 675–686, Melbourne, Australia. Association for Computational Linguistics.

Janara Christensen, Stephen Soderland, Oren Etzioni, et al. 2013. Towards coherent multi-document summarization. In *Proceedings of the 2013 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pages 1163–1173.

John M Conroy, Judith D Schlesinger, Dianne P Oleary, and Jade Goldstein. 2006. Back to basics: Classy 2006. In *DUC*.

Eugene S Edgington. 1969. Approximate randomization tests. *The Journal of Psychology*, 72(2):143–149.

Lisa Fan, Dong Yu, and Lu Wang. 2018. Robust neural abstractive summarization systems and evaluation against adversarial information. In *Workshop on Interpretability and Robustness in Audio, Speech, and Language (IRASL)*. Neural Information Processing Systems.

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109, Brussels, Belgium. Association for Computational Linguistics.

Mor Geva, Eric Malmi, Idan Szpektor, and Jonathan Berant. 2019. DiscoFuse: A Large-Scale Dataset for Discourse-Based Sentence Fusion. In *Proceedings of the 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. ArXiv preprint arXiv:1902.10526.

Max Grusky, Mor Naaman, and Yoav Artzi. 2018. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 708–719, New Orleans, Louisiana. Association for Computational Linguistics.

Camille Guinaudeau and Michael Strube. 2013. Graph-based local coherence modeling. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 93–103, Sofia, Bulgaria. Association for Computational Linguistics.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in neural information processing systems*, pages 1693–1701.

Wan-Ting Hsu, Chieh-Kai Lin, Ming-Ying Lee, Kerui Min, Jing Tang, and Min Sun. 2018. A unified model for extractive and abstractive summarization using inconsistency loss. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 132–141, Melbourne, Australia. Association for Computational Linguistics.

K Sparck Jones et al. 1999. Automatic summarizing: factors and directions. *Advances in automatic text summarization*, pages 1–12.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751,

Doha, Qatar. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Wojciech Kryściński, Romain Paulus, Caiming Xiong, and Richard Socher. 2018. Improving abstraction in text summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1808–1817.

Mirella Lapata and Regina Barzilay. 2005. Automatic evaluation of text coherence: Models and representations. In *IJCAI*, volume 5, pages 1085–1090.

Chenliang Li, Weiran Xu, Si Li, and Sheng Gao. 2018. Guiding generation for abstractive text summarization based on key information guide network. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 55–60, New Orleans, Louisiana. Association for Computational Linguistics.

Jiwei Li and Eduard Hovy. 2014. A model of coherence based on distributed sentence representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2039–2048, Doha, Qatar. Association for Computational Linguistics.

Piji Li, Lidong Bing, Wai Lam, Hang Li, and Yi Liao. 2015. Reader-aware multi-document summarization via sparse coding. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Wenjie Li, Mingli Wu, Qin Lu, Wei Xu, and Chunfa Yuan. 2006. Extractive summarization using inter- and intra- event relevance. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 369–376, Sydney, Australia. Association for Computational Linguistics.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic Evaluation of Summaries Using N-gram Co-occurrence Statistics. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, pages 71–78.

Chia-Wei Liu, Ryan Lowe, Iulian V Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *arXiv preprint arXiv:1603.08023*.

Hans Peter Luhn. 1958. The automatic creation of literature abstracts. *IBM Journal of research and development*, 2(2):159–165.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Mohsen Mesgar and Michael Strube. 2018. A neural local coherence model for text quality assessment. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4328–4339, Brussels, Belgium. Association for Computational Linguistics.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*.

Ani Nenkova. 2008. Entity-driven rewrite for multi-document summarization. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-I*.

Daraksha Parveen, Mohsen Mesgar, and Michael Strube. 2016. Generating coherent summaries of scientific articles using coherence patterns. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 772–783, Austin, Texas. Association for Computational Linguistics.

Daraksha Parveen, Hans-Martin Ramsl, and Michael Strube. 2015. Topical coherence for graph-based extractive summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1949–1954, Lisbon, Portugal. Association for Computational Linguistics.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7008–7024.

Evan Sandhaus. 2008. The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Advaith Siddharthan, Ani Nenkova, and Kathleen McKeown. 2011. Information status distinctions and referring expressions: An empirical study of references to people in news summaries. *Computational Linguistics*, 37(4):811–842.

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017a. Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1171–1181.

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017b. Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1171–1181, Vancouver, Canada. Association for Computational Linguistics.

Dat Tien Nguyen and Shafiq Joty. 2017. A neural local coherence model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1320–1330, Vancouver, Canada. Association for Computational Linguistics.

Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. 2015a. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015b. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Lu Wang, Hema Raghavan, Vittorio Castelli, Radu Florian, and Claire Cardie. 2013. A Sentence Compression Based Framework to Query-Focused Multi-Document Summarization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1384–1394, Sofia, Bulgaria. Association for Computational Linguistics.

Ronald J Williams and David Zipser. 1995. Gradient-based learning algorithms for recurrent. *Backpropagation: Theory, architectures, and applications*, 433.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263, Copenhagen, Denmark. Association for Computational Linguistics.

Yuxiang Wu and Baotian Hu. 2018. Learning to extract coherent summary via deep reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Qingyu Zhou, Nan Yang, Furu Wei, Shaohan Huang, Ming Zhou, and Tiejun Zhao. 2018. Neural document summarization by jointly learning to score and select sentences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 654–663, Melbourne, Australia. Association for Computational Linguistics.

# Neural Extractive Text Summarization with Syntactic Compression

**Jiacheng Xu** and **Greg Durrett**
Department of Computer Science
The University of Texas at Austin
{jcxu,gdurrett}@cs.utexas.edu

## Abstract

Recent neural network approaches to summarization are largely either selection-based extraction or generation-based abstraction. In this work, we present a neural model for single-document summarization based on joint extraction and syntactic compression. Our model chooses sentences from the document, identifies possible compressions based on constituency parses, and scores those compressions with a neural model to produce the final summary. For learning, we construct oracle extractive-compressive summaries, then learn both of our components jointly with this supervision. Experimental results on the CNN/Daily Mail and New York Times datasets show that our model achieves strong performance (comparable to state-of-the-art systems) as evaluated by ROUGE. Moreover, our approach outperforms an off-the-shelf compression module, and human and manual evaluation shows that our model's output generally remains grammatical.

## 1 Introduction

Neural network approaches to document summarization have ranged from purely extractive (Cheng and Lapata, 2016; Nallapati et al., 2017; Narayan et al., 2018) to abstractive (Rush et al., 2015; Nallapati et al., 2016; Chopra et al., 2016; Tan et al., 2017; Gehrmann et al., 2018). Extractive systems are robust and straightforward to use. Abstractive systems are more flexible for varied summarization situations (Grusky et al., 2018), but can make factual errors (Cao et al., 2018; Li et al., 2018) or fall back on extraction in practice (See et al., 2017). Extractive and compressive systems (Berg-Kirkpatrick et al., 2011; Qian and Liu, 2013; Durrett et al., 2016) combine the strengths of both approaches; however, there has been little work studying neural network models in this vein, and the approaches that have been employed typically use seq2seq-based sentence compression (Chen and Bansal, 2018).

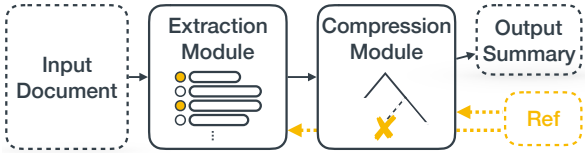


Figure 1: Diagram of the proposed model. Extraction and compression are modularized but jointly trained with supervision derived from the reference summary.

In this work, we propose a model that can combine the high performance of neural extractive systems, additional flexibility from compression, and interpretability given by having discrete compression options. Our model first encodes the source document and its sentences and then sequentially selects a set of sentences to further compress. Each sentence has a set of compression options available that are selected to preserve meaning and grammaticality; these are derived from syntactic constituency parses and represent an expanded set of discrete options from prior work (Berg-Kirkpatrick et al., 2011; Wang et al., 2013). The neural model additionally scores and chooses which compressions to apply given the context of the document, the sentence, and the decoder model's recurrent state.

A principal challenge of training an extractive and compressive model is constructing the oracle summary for supervision. We identify a set of high-quality sentences from the document with beam search and derive oracle compression labels in each sentence through an additional refinement process. Our model's training objective combines these extractive and compressive components and learns them jointly.

We conduct experiments on standard single document news summarization datasets: CNN, Daily Mail (Hermann et al., 2015), and the New



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3292–3303,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

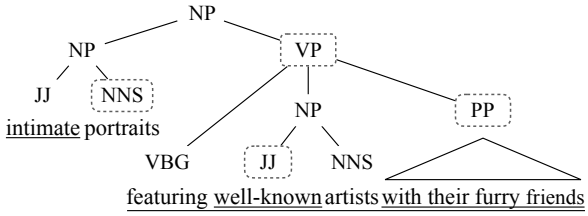


Figure 2: Text compression example. In this case, "intimate", "well-known", "with their furry friends" and "featuring ... friends" are deletable given compression rules.

York Times Annotated Corpus (Sandhaus, 2008). Our model matches or exceeds the state-of-the-art on all of these datasets and achieves the largest improvement on CNN (+2.4 ROUGE-$F_1$ over our extractive baseline) due to the more compressed nature of CNN summaries. We show that our model's compression threshold is robust across a range of settings yet tunable to give different-length summaries. Finally, we investigate the fluency and grammaticality of our compressed sentences. The human evaluation shows that our system yields generally grammatical output, with many remaining errors being attributed to the parser.[1]

## 2 Compression in Summarization

Sentence compression is a long-studied problem dealing with how to delete the least critical information in a sentence to make it shorter (Knight and Marcu, 2000, 2002; Martins and Smith, 2009; Cohn and Lapata, 2009; Wang et al., 2013; Li et al., 2014). Many of these approaches are syntax-driven, though end-to-end neural models have been proposed as well (Filippova et al., 2015; Wang et al., 2017). Past non-neural work on summarization has used both syntax-based (Berg-Kirkpatrick et al., 2011; Woodsend and Lapata, 2011) and discourse-based (Carlson et al., 2001; Hirao et al., 2013; Li et al., 2016) compressions. Our approach follows in the syntax-driven vein.

Our high-level approach to summarization is shown in Figure 1. In Section 3, we describe the models for extraction and compression. Our compression depends on having a discrete set of valid compression options that maintain the grammaticality of the underlying sentence, which we now proceed to describe.

[1]The code, full model output, and the pre-trained model are available at https://github.com/jiacheng-xu/neu-compression-sum

**Compression Rules** We refer to the rules derived in Li et al. (2014), Wang et al. (2013), and Durrett et al. (2016) and design a concise set of syntactic rules including the removal of: 1. Appositive noun phrases; 2. Relative clauses and adverbial clauses; 3. Adjective phrases in noun phrases, and adverbial phrases (see Figure 2); 4. Gerundive verb phrases as part of noun phrases (see Figure 2); 5. Prepositional phrases in certain configurations like *on Monday*; 6. Content within parentheses and other parentheticals.

Figure 2 shows examples of several compression rules applied to a short snippet. All combinations of compressions maintain grammaticality, though some content is fairly important in this context (the VP and PP) and should not be deleted. Our model must learn not to delete these elements.

**Compressability** Summaries from different sources may feature various levels of compression. At one extreme, a summary could be fully sentence-extractive; at another extreme, the editor may have compressed a lot of content in a sentence. In Section 4, we examine this question on our summarization datasets and use it to motivate our choice of evaluation datasets.

**Universal Compression with ROUGE** While we use syntax as a source of compression options, we note that other ways of generating compression options are possible, including using labeled compression data. However, supervising compression with ROUGE is critical to learn what information is important for this particular source, and in any case, labeled compression data is unavailable in many domains. In Section 5, we compare our model to off-the-shelf sentence compression module and find that it substantially underperforms our approach.

## 3 Model

Our model is a neural network model that encodes a source document, chooses sentences from that document, and selects discrete compression options to apply. The model architecture of sentence extraction module and text compression module are shown in Figure 3 and 4.

### 3.1 Extractive Sentence Selection

A single document consists of $n$ sentences $D = \{s_1, s_2, \cdots, s_n\}$. The $i$-th sentence is denoted as $s_i = \{w_{i1}, w_{i2}, \cdots, w_{im}\}$ where $w_{ij}$ is the

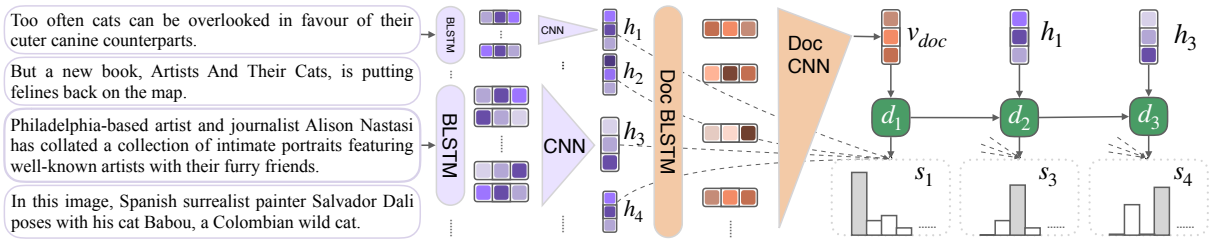


Figure 3: Sentence extraction module of JECS. Words in input document sentences are encoded with BiLSTMs. Two layers of CNNs aggregate these into sentence representations $h_i$ and then the document representation $v_{doc}$. This is fed into an attentive LSTM decoder which selects sentences based on the decoder state $d$ and the representations $h_i$, similar to a pointer network.

$j$-th word in $s_i$. The content selection module learns to pick up a subset of $D$ denoted as $\hat{D} = \{\hat{s}_1, \hat{s}_2, \cdots, \hat{s}_k, |\hat{s}_i \in D\}$ where $k$ sentences are selected.

**Sentence & Document Encoder** We first use a bidirectional LSTM to encode words in each sentence in the document separately and then we apply multiple convolution layers and max pooling layers to extract the representation of every sentence. Specifically, $[h_{i1}, \cdots, h_{im}] = \text{BiLSTM}([w_{i1}, \cdots, w_{im}])$ and $h_i = \text{CNN}([h_{i1}, \cdots, h_{im}])$ where $h_i$ is a representation of the $i$-th sentence in the document. This process is shown in the left side of Figure 3 illustrated in purple blocks. We then aggregate these sentence representations into a document representation $v_{doc}$ with a similar BiLSTM and CNN combination, shown in Figure 3 with orange blocks.

**Decoding** The decoding stage selects a number of sentences given the document representation $v_{doc}$ and sentences' representations $h_i$. This process is depicted in the right half of Figure 3. We use a sequential LSTM decoder where, at each time step, we take the representation $h$ of the last selected sentence, the overall document vector $v_{doc}$, and the recurrent state $d_{t-1}$, and produce a distribution over all of the remaining sentences excluding those already selected. This approach resembles pointer network-style approaches used in past work (Zhou et al., 2018). Formally, we write this as:

$$d_t = \text{LSTM}(d_{t-1}, h_k, v_{doc})$$
$$\text{score}_{t,i} = W_m \tanh(W_d d_t + W_h h_i)$$
$$p(\hat{s}_t = s_i | d_t, h_k, v_{doc}, h_i) = \text{softmax}(\text{score}_{t,i})$$

where $h_k$ is the representation of the sentence selected at time step $t-1$. $d_{t-1}$ is the decoding hidden state from last time step. $W_d$, $W_h$, $W_m$, and parameters in LSTM are learned. Once a sentence is selected, it cannot be selected again. At test time, we use greedy decoding to identify the most likely sequence of sentences under our model.[2]

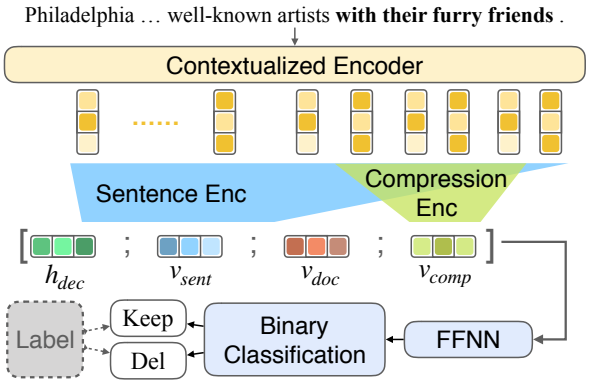


Figure 4: Text compression module. A neural classifier scores the compression option (*with their furry friends*) in the sentence and broader document context and decides whether or not to delete it.

## 3.2 Text Compression

After selecting the sentences, the text compression module evaluates our discrete compression options and decides whether to remove certain phrases or words in the selected sentences. Figure 4 shows an example of this process for deciding whether or not to delete a PP in this sentence. This PP was marked as deletable based on rules described in Section 2. Our network then encodes this sentence and the compression, combines this information with the document context $v_{doc}$ and decoding context $h_{dec}$, and uses a feedforward network to decide whether or not to delete the span.

[2]For our experiments, we decode for a fixed number of sentences, tuned for each dataset, as in prior extractive work (Narayan et al., 2018). We experimented with dynamically choosing a number of sentences and found this to make little difference.

Let $C_i = \{c_{i1}, \cdots, c_{il}\}$ denote the possible compression spans derived from the rules described in Section 2. Let $y_{i,c}$ be a binary variable equal to 1 if we are deleting the $c$th option of the $i$th sentence. Our text compression module models $p(y_{i,c}|D, \hat{s}_t = s_i)$ as described in the following section.

**Compression Encoder** We use a contextualized encoder, ELMo (Peters et al., 2018) to compute contextualized word representations. We then use CNNs with max pooling to encode the sentence (shown in blue in Figure 4) and the candidate compression (shown in light green in Figure 4). The sentence representation $v_{sent}$ and the compression span representation $v_{comp}$ are concatenated with the hidden state in sentence decoder $h_{dec}$ and the document representation $v_{doc}$.

**Compression Classifier** We feed the concatenated representation to a feedforward neural network to predict whether the compression span should be deleted or kept, which is formulated as a binary classification problem. This classifier computes the final probability $p(y_{i,c}|D, \hat{s}_t = s_i) = p(y_{i,c}|h_{dec}, v_{doc}, v_{comp}, s_i)$. The overall probability of a summary $(\hat{s}, \hat{y})$, where $\hat{s}$ is the sentence oracle and $\hat{y}$ is the compression label, is the product of extraction and compression models: $p(\hat{s}, \hat{y}|D) = \prod_{t=1}^{T} \left[p(\hat{s}_t|D, \hat{s}_{<t}) \prod_{c \in C_i} p(\hat{y}_{t,c}|D, \hat{s})\right]$.

**Heuristic Deduplication** Inspired by the trigram avoidance trick proposed in Paulus et al. (2018) to reduce redundancy, we take full advantage of our linguistically motivated compression rules and the constituent parse tree and allow our model to compress deletable chunks with redundant information. We therefore take our model's output and apply a postprocessing stage where we remove any compression option whose unigrams are completely covered elsewhere in the summary. We perform this compression after the model prediction and compression.

## 4 Training

Our model makes a series of sentence extraction decisions $\hat{s}$ and then compression decisions $\hat{y}$. To supervise it, we need to derive gold-standard labels for these decisions. Our oracle identification approach relies on first identifying an oracle set of sentences and then the oracle compression options.[3]

**Reference**: Artist and journalist Alison Nastasi put together the portrait collection. Also features images of Picasso, Frida Kahlo, and John Lennon. Reveals quaint personality traits shared between artists and their felines.
**Document**: ... **Philadelphia-based** artist and journalist Alison Nastasi has collated a collection of **intimate** portraits **featuring well-known artists with their furry friends**. ...

| Compression $R_{bf} = 19.4$ | $R_{af}$ | Ratio | Label |
|---|---|---|---|
| Philadelphia-based | 19.8 | 1.02 | DEL |
| intimate | 19.8 | 1.02 | DEL |
| well known | 20.4 | 1.05 | DEL |
| featuring ... their furry friends | 18.1 | 0.93 | KEEP |

Table 1: Oracle label computation for the text compression module. $R_{bf}$ and $R_{af}$ are the ROUGE scores before and after compression. The ratio is defined as $\frac{R_{af}}{R_{bf}}$. ROUGE increases when words not appearing in the reference are deleted. ROUGE can decrease when terms appearing in the reference summary, like *featuring*, are deleted.

### 4.1 Oracle Construction

**Sentence Extractive Oracle** We first identify an oracle set of sentences to extract using a beam search procedure similar to Maximal Marginal Relevance (MMR) (Carbonell and Goldstein, 1998). For each additional sentence we propose to add, we compute a heuristic cost equal to the ROUGE score of a given sentence with respect to the reference summary. When pruning states, we calculate the ROUGE score of the combination of sentences currently selected and sort in descending order. Let the beam width be $\beta$. The time complexity of the approximate approach is $O(nk\beta)$ where in practice $k \ll n$ and $\beta \ll n$. We set $\beta = 8$ and $n = 30$ which means we only consider the first 30 sentences in the document.

The beam search procedure returns a beam of $\beta$ different sentence combinations in the final beam. We use the sentence extractive oracle for both the extraction-only model and the joint extraction-compression model.

**Oracle Compression Labels** To form our joint extractive and compressive oracle, we need to give the compression decisions binary labels $y_{i,c}$ in each set of extracted sentences. For simplicity and computational efficiency, we assign each sentence

[3]Depending on which sentences are extracted, different compression decisions may be optimal; however, re-deriving these with a dynamic oracle (Goldberg and Nivre, 2012) is prohibitively expensive during training.

| Category | CNN | DM | NYT |
|---|---|---|---|
| Bad | 27% | 48% | 49% |
| Weak Positive | 58% | 43% | 47% |
| Strong Positive | 15% | 10% | 4% |

Table 2: Compressibility: The oracle label distribution over three datasets. Compressions in the "Bad" category decrease ROUGE and are labeled as negative (do not delete), while weak positive (less than 5% ROUGE improvement) and strong positive (greater than 5%) both represent ROUGE improvements. CNN features much more compression than the other datasets.

a single $y_{i,c}$ independent of the context it occurs in. For each compression option, we assess the value of it by comparing the ROUGE score of the sentence with and without this phrase. Any option that increases ROUGE is treated as a compression that should be applied. When calculating this ROUGE value, we remove stop words include stemming.

We run this procedure on each of our oracle extractive sentences. The fraction of positive and negative labels assigned to compression options is shown for each of the three datasets in Table 2. CNN is the most compressable dataset among CNN, DM and NYT.

**ILP-based oracle construction** Past work has derived oracles for extractive and compressive systems using integer linear programming (ILP) (Gillick and Favre, 2009; Berg-Kirkpatrick et al., 2011). Following their approach, we can directly optimize for ROUGE recall of an extractive or compressive summary in our framework if we specify a length limit. However, we evaluate on ROUGE $F_1$ as is standard when comparing to neural models that don't produce fixed-length summaries. Optimizing for ROUGE $F_1$ cannot be formulated as an ILP, since computing precision requires dividing by the number of selected words, making the objective no longer linear. We experimented with optimizing for ROUGE $F_1$ indirectly by finding optimal ROUGE recall summaries at various settings of maximum summary length. However, these summaries frequently contained short sentences to fill up the budget, and the collection of summaries returned tended to be less diverse than those found by beam search.

### 4.2 Learning Objective

Often, many oracle summaries achieve very similar ROUGE values. We therefore want to avoid committing to a single oracle summary for the learning process. Our procedure from Section 4.1 can generate $m$ extractive oracles $s_i^*$; let $s_{i,t}^*$ denote the gold sentence for the $i$-th oracle at timestep $t$. Past work (Narayan et al., 2018; Chen and Bansal, 2018) has employed policy gradient in this setting to optimize directly for ROUGE. However, because oracle summaries usually have very similar ROUGE scores, we choose to simplify this objective as $\mathcal{L}_{\text{sent}} = -\frac{1}{m}\sum_{i=1}^{m}\sum_{t=1}^{T}\log p(s_{i,t}^*|D, s_{i,<t}^*)$. Put another way, we optimize the log likelihood averaged across $m$ different oracles to ensure that each has high likelihood. We use $m = 5$ oracles during training. The oracle sentence indices are sorted according to the individual salience (ROUGE score) rather than document order.

The objective of the compression module is defined as $\mathcal{L}_{\text{comp}} = -\sum_{i=1}^{m}\sum_{c=1}^{C}\log p(y_{i,c}^*|D, \hat{s})$ where $p(y_{i,c}^*)$ is the probability of the target decision for the $c$-th compression options of the $i$-th sentence. The joint loss function is $\mathcal{L} = \mathcal{L}_{\text{sent}} + \alpha\mathcal{L}_{\text{comp}}$. We set $\alpha = 1$ in practice.

## 5 Experiments

We evaluate our model on two axes. First, for content selection, we use ROUGE as is standard. Second, we evaluate the grammaticality of our model to ensure that it is not substantially damaged by compression.

### 5.1 Experimental Setup

**Datasets** We evaluate the proposed method on three popular news summarization datasets: the New York Times corpus (Sandhaus, 2008), CNN and Dailymail (DM) (Hermann et al., 2015).[4]

As discussed in Section 2, compression will give different results on different datasets depending on how much compression is optimal from the standpoint of reproducing the reference summaries, which changes how measurable the impact of compression is. In Table 2, we show the "compressability" of these three datasets: how valuable various compression options seem to be from the standpoint of improving ROUGE. We found that CNN has significantly more positive compression options than the other two. Critically, CNN also has the shortest references (37 words on average,

[4]More details about the experimental setup, implementation details, and human evaluation are provided in the Appendix.

| Model | CNN | | |
|---|---|---|---|
| | R-1 | R-2 | R-L |
| Lead (Ours) | 29.1 | 11.1 | 25.8 |
| Refresh* (Narayan et al., 2018) | 30.3 | **11.6** | 26.9 |
| LatSum* (Zhang et al., 2018) | 28.8 | 11.5 | 25.4 |
| BanditSum (Dong et al., 2018) | **30.7** | **11.6** | **27.4** |
| LEADDEDUP | 29.7 | 10.9 | 26.2 |
| LEADCOMP | 30.6 | 10.8 | 27.2 |
| EXTRACTION | 30.3 | 11.0 | 26.5 |
| EXTLSTMDEL | 30.6 | 11.9 | 27.1 |
| JECS | **32.7** | **12.2** | **29.0** |

Table 3: Experimental results on the test sets of CNN. * indicates models evaluates with our own ROUGE metrics. Our model outperforms our extractive model and lead-based baselines, as well as prior work.

| Model | CNNDM | | |
|---|---|---|---|
| | R-1 | R-2 | R-L |
| Lead (Ours) | 40.3 | 17.6 | 36.4 |
| Refresh* (Narayan et al., 2018) | 40.0 | 18.1 | 36.6 |
| NeuSum | **41.6** | **19.0** | 38.0 |
| LatSum* (Zhang et al., 2018) | 41.0 | 18.8 | 37.4 |
| LatSum w/ Compression | 36.7 | 15.4 | 34.3 |
| BanditSum | 41.5 | 18.7 | 37.6 |
| CBDec (Jiang and Bansal, 2018) | 40.7 | 17.9 | 37.1 |
| FARS (Chen and Bansal, 2018) | 40.9 | 17.8 | **38.5** |
| LEADDEDUP | 40.5 | 17.4 | 36.5 |
| LEADCOMP | 40.8 | 17.4 | 36.8 |
| EXTRACTION | 40.7 | 18.0 | 36.8 |
| JECS | **41.7** | **18.5** | **37.9** |

Table 4: Experimental results on the test sets of CNNDM. The portion of CNN is roughly one of tenth of DM. Gains are more pronounced on CNN because this dataset features shorter, more compressed reference summaries.

compared to 61 for Daily Mail; see Appendix). In our experiments, we first focus on CNN and then evaluate on the other datasets.

**Models** We present several variants of our model to show how extraction and compression work jointly. In extractive summarization, the LEAD baseline (first $k$ sentences) is a strong baseline due to how newswire articles are written. LEADDEDUP is a non-learned baseline that uses our heuristic deduplication technique on the lead sentences. LEADCOMP is a compression-only model where compression is performed on the lead sentences. This shows the effectiveness of the compression module in isolation rather than in the context of abstraction. EXTRACTION is the extraction only model. JECS is the full Joint Extractive and Compressive Summarizer.

We compare our model with various abstractive and extractive summarization models. NeuSum (Zhou et al., 2018) uses a seq2seq model to predict a sequence of sentences indices to be picked up from the document. Our extractive approach is most similar to this model. Refresh (Narayan et al., 2018), BanditSum (Dong et al., 2018) and LatSum (Zhang et al., 2018) are extractive summarization models for comparison. We also compare with some abstractive models including PointGen-Cov (See et al., 2017), FARS (Chen and Bansal, 2018) and CBDec (Jiang and Bansal, 2018).

We also compare our joint model with a pipeline model with an off-the-shelf compression module. We implement a deletion-based BiLSTM model for sentence compression (Wang et al., 2017) and run the model on top of our extraction output.[5] The pipeline model is denoted as EXTLSTMDEL.

## 5.2 Results on CNN

Table 3 shows experiments results on CNN. We list performance of the LEAD baseline and the performance of competitor models on these datasets. Starred models are evaluated according to our ROUGE metrics; numbers very closely match the originally reported results.

Our model achieves substantially higher performance than all baselines and past systems (+2 ROUGE F1 compared to any of these). On this dataset, compression is substantially useful. Compression is somewhat effective in isolation, as shown by the performance of LEADDEDUP and LEADCOMP. But compression in isolation still gives less benefit (on top of LEAD) than when combined with the extractive model (JECS) in the joint framework. Furthermore, our model beats the pipeline model EXTLSTMDEL which shows the necessity of training a joint model with ROUGE supervision.

## 5.3 Results on Combined CNNDM and NYT

We also report the results on the full CNNDM and NYT although they are less compressable. Table 4 and Table 5 shows the experimental results on these datasets.

Our models still yield strong performance compared to baselines and past work on the CNNDM

[5]We reimplemented the authors' model following their specification and matched their accuracy. For fair comparison, we tuned the deletion threshold to match the compression rate of our model; other choices did not lead to better ROUGE scores.

| Model | R-1 | R-2 | R-L |
|---|---|---|---|
| Lead | 41.8 | 22.6 | 35.0 |
| LEADDEDUP | 42.0 | 22.8 | 35.0 |
| LEADCOMP | 42.4 | 22.7 | 35.4 |
| EXTRACTION | 44.3 | 25.5 | 37.1 |
| JECS | 45.5 | 25.3 | 38.2 |

Table 5: Experimental results on the NYT50 dataset. ROUGE-1, -2 and -L $F_1$ is reported. JECS substantially outperforms our Lead-based systems and our extractive model.

| Model | Preference (%) ↑ | Error ↓ | R-1 ↑ |
|---|---|---|---|
| EXT LEAD3 | – | 22 | 29.1 |
| EXTDROP | 12% | 161 | 30.2 |
| EXTLSTMDEL | 43% | **24** | 30.6 |
| JECS | **45**% | 31 | **32.7** |

Table 6: Human preference, ROUGE and Grammarly grammar checking results. We asked Turkers to rank the models' output based on grammaticality. Error shows the number of grammar errors in 500 sentences reported by Grammarly. Our JECS model achieves the highest ROUGE and is preferred by humans while still making relatively few errors.

dataset. The EXTRACTION model achieves comparable results to past successful extractive approaches on CNNDM and JECS improves on this across the datasets. In some cases, our model slightly underperforms on ROUGE-2. One possible reason is that we remove stop words when constructing our oracles, which could underestimate the importance of bigrams containing stopwords for evaluation. Finally, we note that our compressive approach substantially outperforms the compression-augmented LatSum model. That model used a separate seq2seq model for rewriting, which is potentially harder to learn than our compression model.

On NYT, we see again that the inclusion of compression leads to improvements in both the LEAD setting as well as for our full JECS model.[6]

### 5.4 Grammaticality

We evaluate grammaticality of our compressed summaries in three ways. First, we use Amazon Mechanical Turk to compare different compression techniques. Second, to measure absolute grammaticality, we use an automated out-of-the-box tool Grammarly. Finally, we conduct manual analysis.

**Human Evaluation** We first conduct a human evaluation on the Amazon Mechanical Turk platform. We ask Turkers to rank different compression versions of a sentence in terms of grammaticality. We compare our full JECS model and the off-the-shelf pipeline model EXTLSTMDEL, which have matched compression ratios. We also propose another baseline, EXTRACTDROPOUT, which randomly drops words in a sentence to match the compression ratio of the other two models. The results are shown in Table 6. Turkers give roughly equal preference to our model and the EXTLSTMDEL model, which was learned from supervised compression data. However, our JECS model achieves substantially higher ROUGE score, indicating that it represents a more effective compression approach.

We found that absolute grammaticality judgments were hard to achieve on Mechanical Turk; Turkers' ratings of grammaticality were very noisy and they did not consistently rate true article sentences above obviously noised variants. Therefore, we turn to other methods as described in the next two paragraphs.

**Automatic Grammar Checking** We use Grammarly to check 500 sentences sampled from the outputs of the three models mentioned above from CNN. Both EXTLSTMDEL and JECS make a small number of grammar errors, not much higher than the purely extractive LEAD3 baseline. One major source of errors for JECS is having the wrong article after the deletion of an adjective like *an [awesome] style*.

**Manual Error Analysis** To get a better sense of our model's output, we conduct a manual analysis of our applied compressions to get a sense of how many are valid. We manually examined 40 model summaries, comparing the output with the raw sentences before compression, and identified the following errors: 1. Eight bad deletions due to parsing errors like *a UK [JJ national] from London*. 2. Eight inappropriate adjective deletions causing correctness issues with respect to the reference document like *[former] president* and *[nuclear] weapon*. 3. Three other errors: partial deletion of slang, inappropriate PP attachment deletion, and an unhandled grammatical construction:

[6] Paulus et al. (2018) do not use the NYT50 dataset, so our results are not directly comparable to theirs. Durrett et al. (2016) use a different evaluation setup with a hard limit on the summary length and evaluation on recall only.

| Reference Summary | Prediction with Compressions |
|---|---|
| Mullah Omar, the reclusive founder of the Afghan Taliban, is still in charge, a new biography claims. An ex-Taliban insider says there have been rumors that the one-eyed militant is dead. | (CNN) Mullah Mohammed Omar is "still the leader" of the Taliban's self-declared Islamic Emirate of Afghanistan. The Taliban's "Cultural Commission" released the 11-page document in several different translations on the movement's website, ostensibly to commemorate the 19th anniversary of an April 4, 1996, meeting in Afghanistan's Kandahar province when an assembly of Afghans swore allegiance to Omar. |
| Rebecca Francis' photo with a giraffe was shared by Ricky Gervais. Francis was threatened on Twitter for the picture. Francis, a hunter, said the giraffe was "close to death" and became food for locals. | (CNN) Five years ago, Rebecca Francis posed for a photo while lying next to a dead giraffe. The trouble started Monday, when comedian Ricky Gervais tweeted the photo with a question. Francis, who has appeared on the NBC Sports Network outdoor lifestyle show "Eye of the Hunter" and was the subject of an interview with Hunting Life in late March, responded in a statement to HuntingLife.com on Tuesday, which was posted on its Facebook page. |
| Frida Ghitis: President Barack Obama is right to want a deal, but this one gives Iran too much. She says the framework agreement starts lifting Iran sanctions much too soon. | (CNN) President Barack Obama tied himself to the mast of a nuclear deal with Iran even before he became the Democratic candidate for president. Reaching a good, solid agreement with Iran is a worthy, desirable goal. But the process has unfolded under the destructive influence of political considerations, weakening America's hand and strengthening Iran. |

Table 7: Examples of applied compressions. The top two are sampled from among the most compressed examples in the dataset. Our JECS model is able to delete both large chunks (especially temporal PPs giving dates of events) as well as individual modifiers that aren't determined to be relevant to the summary (e.g., the specification of the *19th* anniversary). The last example features more modest compression.

*students [first], athletes [second].*

Examples of output are shown in Table 7. The first two examples are sampled from the top 25% of the most compressed examples in the corpus. We see a variety of compression options that are used in the first two examples, including removal of temporal PPs, large subordinate clauses, adjectives, and parentheticals. The last example features less compression, only removing a handful of adjectives in a manner which slightly changes the meaning of the summary.

Improving the parser and deriving a more semantically-aware set of compression rules can help achieving better grammaticality and readability. However, we note that such errors are largely orthogonal to the core of our approach; a more refined set of compression options could be dropped into our system and used without changing our fundamental model.

## 6 Compression Analysis

**Compression Threshold** Compression in our model is an imbalanced binary classification problem. The trained model's natural classification threshold (probability of DEL $> 0.5$) may not be optimal for downstream ROUGE. We experiment with varying the classification threshold from 0 (no deletion, only heuristic deduplication) to 1 (all compressible pieces removed). The results on CNN are shown in Figure 5, where we show the average ROUGE value at different compression thresholds. The model achieves the best performance at 0.45 but performs well in a wide range from 0.3 to 0.55. Our compression is therefore robust yet also provides a controllable parameter to change the amount of compression in produced summaries.

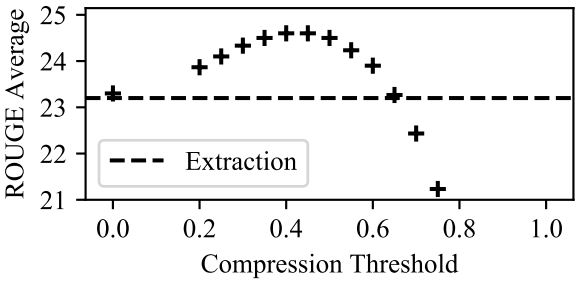


Figure 5: Effect of changing the compression threshold on CNN. The y-axis shows the average of the F1 of ROUGE-1,-2 and -L. The dotted line is the extractive baseline. The model outperforms the extractive model and achieves nearly optimal performance across a range of threshold values.

**Compression Type Analysis** We further break down the types of compressions used in the model. Table 8 shows the compressions that our model ends up choosing at test time. PPs are often compressed by the deduplication mechanism because the compressible PPs tend to be temporal and location adjuncts, which may be redundant across sen-

| Node Type | Len | % of comps | Comp Acc | Dedup |
|---|---|---|---|---|
| JJ | 1.0 | 34% | 72% | 30% |
| PP | 3.4 | 26% | 47% | 72% |
| ADVP | 1.4 | 17% | 79% | 17% |
| PRN | 2.2 | 6% | 80% | 5% |

Table 8: The compressions used by our model on CNN; average lengths and the fraction of that constituency type among compressions taken by our model. Comp Acc indicates how frequently that compression was taken by the oracle; note that error, especially keeping constituents that we shouldn't, may have minimal impact on summary quality. Dedup indicates the percentage of chosen compressions which arise from deduplication as opposed to model prediction.

tences. Without the manual deduplication mechanism, our model matches the ground truth around 80% of the time. However, a low accuracy here may not actually cause a low final ROUGE score, as many compression choices only affect the final ROUGE score by a small amount. More details about compression options are in the Supplementary Material.

## 7 Related Work

**Neural Extractive Summarization** Neural networks have shown to be effective in extractive summarization. Past approaches have structured the decision either as binary classification over sentences (Cheng and Lapata, 2016; Nallapati et al., 2017) or classification followed by ranking (Narayan et al., 2018). Zhou et al. (2018) used a seq-to-seq decoder instead. For our model, text compression forms a module largely orthogonal to the extraction module, so additional improvements to extractive modeling might be expected to stack with our approach.

**Syntactic Compression** Prior to the explosion of neural models for summarization, syntactic compression (Martins and Smith, 2009; Woodsend and Lapata, 2011) was relatively more common. Several systems explored the usage of constituency parses (Berg-Kirkpatrick et al., 2011; Wang et al., 2013; Li et al., 2014) as well as RST-based approaches (Hirao et al., 2013; Durrett et al., 2016). Our approach follows in this vein but could be combined with more sophisticated neural text compression methods as well.

**Neural Text Compression** Filippova et al. (2015) presented an LSTM approach to deletion-based sentence compression. Miao and Blunsom (2016) proposed a deep generative model for text compression. Zhang et al. (2018) explored the compression module after the extraction model but the separation of these two modules hurt the performance. For this work, we find that relying on syntax gives us more easily understandable and controllable compression options.

Contemporaneously with our work, Mendes et al. (2019) explored an extractive and compressive approach using compression integrated into a sequential decoding process; however, their approach does not leverage explicit syntax and makes several different model design choices.

## 8 Conclusion

In this work, we presented a neural network framework for extractive and compressive summarization. Our model consists of a sentence extraction model joined with a compression classifier that decides whether or not to delete syntax-derived compression options for each sentence. Training the model involves finding an oracle set of extraction and compression decision with high score, which we do through a combination of a beam search procedure and heuristics. Our model outperforms past work on the CNN/Daily Mail corpus in terms of ROUGE, achieves substantial gains over the extractive model, and appears to have acceptable grammaticality according to human evaluations.

## Acknowledgments

This work was partially supported by NSF Grant IIS-1814522, a Bloomberg Data Science Grant, and an equipment grant from NVIDIA. The authors acknowledge the Texas Advanced Computing Center (TACC) at The University of Texas at Austin for providing HPC resources used to conduct this research. Results presented in this paper were obtained using the Chameleon testbed supported by the National Science Foundation (Keahey et al., 2019). Thanks as well to the anonymous reviewers for their helpful comments.

## References

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly Learning to Extract and Compress. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 481–490. Association for Computational Linguistics.

Ziqiang Cao, Furu Wei, Wenjie Li, and Sujian Li. 2018. Faithful to the Original: Fact Aware Neural Abstractive Summarization. In *AAAI Conference on Artificial Intelligence*.

Jaime Carbonell and Jade Goldstein. 1998. The Use of MMR, Diversity-based Reranking for Reordering Documents and Producing Summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, pages 335–336, New York, NY, USA. ACM.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurovsky. 2001. Building a Discourse-Tagged Corpus in the Framework of Rhetorical Structure Theory. In *Proceedings of the Second SIGdial Workshop on Discourse and Dialogue*.

Yen-Chun Chen and Mohit Bansal. 2018. Fast Abstractive Summarization with Reinforce-Selected Sentence Rewriting. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 675–686. Association for Computational Linguistics.

Jianpeng Cheng and Mirella Lapata. 2016. Neural Summarization by Extracting Sentences and Words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 484–494. Association for Computational Linguistics.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive Sentence Summarization with Attentive Recurrent Neural Networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98. Association for Computational Linguistics.

Trevor Cohn and Mirella Lapata. 2009. Sentence Compression As Tree Transduction. *J. Artif. Int. Res.*, 34(1):637–674.

Yue Dong, Yikang Shen, Eric Crawford, Herke van Hoof, and Jackie Chi Kit Cheung. 2018. BanditSum: Extractive Summarization as a Contextual Bandit. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3739–3748. Association for Computational Linguistics.

Greg Durrett, Taylor Berg-Kirkpatrick, and Dan Klein. 2016. Learning-Based Single-Document Summarization with Compression and Anaphoricity Constraints. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1998–2008. Association for Computational Linguistics.

Katja Filippova, Enrique Alfonseca, Carlos A. Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence Compression by Deletion with LSTMs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 360–368. Association for Computational Linguistics.

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-Up Abstractive Summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109. Association for Computational Linguistics.

Dan Gillick and Benoit Favre. 2009. A Scalable Global Model for Summarization. In *Proceedings of the Workshop on Integer Linear Programming for Natural Language Processing*, pages 10–18. Association for Computational Linguistics.

Yoav Goldberg and Joakim Nivre. 2012. A Dynamic Oracle for Arc-Eager Dependency Parsing. In *Proceedings of COLING 2012*, pages 959–976. The COLING 2012 Organizing Committee.

Max Grusky, Mor Naaman, and Yoav Artzi. 2018. Newsroom: A Dataset of 1.3 Million Summaries with Diverse Extractive Strategies. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 708–719. Association for Computational Linguistics.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching Machines to Read and Comprehend. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1693–1701. Curran Associates, Inc.

Tsutomu Hirao, Yasuhisa Yoshida, Masaaki Nishino, Norihito Yasuda, and Masaaki Nagata. 2013. Single-Document Summarization as a Tree Knapsack Problem. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1515–1520. Association for Computational Linguistics.

Yichen Jiang and Mohit Bansal. 2018. Closed-Book Training to Improve Summarization Encoder Memory. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4067–4077. Association for Computational Linguistics.

Kate Keahey, Pierre Riteau, Dan Stanzione, Tim Cockerill, Joe Mambretti, Paul Rad, and Paul Ruth. 2019. Chameleon: a scalable production testbed for computer science research. In *Contemporary High Performance Computing: From Petascale toward Exascale*, 1 edition, volume 3 of *Chapman & Hall/CRC Computational Science*, chapter 5, pages 123–148. CRC Press, Boca Raton, FL.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*.

Kevin Knight and Daniel Marcu. 2000. Statistics-Based Summarization - Step One: Sentence Compression. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 703–710. AAAI Press.

Kevin Knight and Daniel Marcu. 2002. Summarization Beyond Sentence Extraction: A Probabilistic Approach to Sentence Compression. *Artif. Intell.*, 139(1):91–107.

Chen Li, Yang Liu, Fei Liu, Lin Zhao, and Fuliang Weng. 2014. Improving Multi-documents Summarization by Sentence Compression based on Expanded Constituent Parse Trees. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 691–701. Association for Computational Linguistics.

Haoran Li, Junnan Zhu, Jiajun Zhang, and Chengqing Zong. 2018. Ensure the Correctness of the Summary: Incorporate Entailment Knowledge into Abstractive Sentence Summarization. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1430–1441. Association for Computational Linguistics.

Junyi Jessy Li, Kapil Thadani, and Amanda Stent. 2016. The role of discourse units in near-extractive summarization. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 137–147. Association for Computational Linguistics.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60. Association for Computational Linguistics.

Andre Martins and Noah A. Smith. 2009. Summarization with a joint model for sentence extraction and compression. In *Proceedings of the Workshop on Integer Linear Programming for Natural Language Processing*, pages 1–9. Association for Computational Linguistics.

Afonso Mendes, Shashi Narayan, Sebastião Miranda, Zita Marinho, André F. T. Martins, and Shay B. Cohen. 2019. Jointly Extracting and Compressing Documents with Summary State Representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*.

Yishu Miao and Phil Blunsom. 2016. Language as a Latent Variable: Discrete Generative Models for Sentence Compression. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 319–328. Association for Computational Linguistics.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. SummaRuNNer: A Recurrent Neural Network Based Sequence Model for Extractive Summarization of Documents. In *AAAI Conference on Artificial Intelligence*.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290. Association for Computational Linguistics.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Ranking sentences for extractive summarization with reinforcement learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1747–1759. Association for Computational Linguistics.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A Deep Reinforced Model for Abstractive Summarization. In *International Conference on Learning Representations*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Xian Qian and Yang Liu. 2013. Fast Joint Compression and Summarization via Graph Cuts. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1492–1502. Association for Computational Linguistics.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A Neural Attention Model for Abstractive Sentence Summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389. Association for Computational Linguistics.

Evan Sandhaus. 2008. The New York Times Annotated Corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get To The Point: Summarization with Pointer-Generator Networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083. Association for Computational Linguistics.

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. Abstractive Document Summarization with a Graph-Based Attentional Neural Model. In *Proceedings*

*of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1171–1181. Association for Computational Linguistics.

Liangguo Wang, Jing Jiang, Hai Leong Chieu, Chen Hui Ong, Dandan Song, and Lejian Liao. 2017. Can syntax help? improving an LSTM-based sentence compression model for new domains. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1385–1393, Vancouver, Canada. Association for Computational Linguistics.

Lu Wang, Hema Raghavan, Vittorio Castelli, Radu Florian, and Claire Cardie. 2013. A Sentence Compression Based Framework to Query-Focused Multi-Document Summarization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1384–1394. Association for Computational Linguistics.

Kristian Woodsend and Mirella Lapata. 2011. Learning to Simplify Sentences with Quasi-Synchronous Grammar and Integer Programming. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 409–420. Association for Computational Linguistics.

Xingxing Zhang, Mirella Lapata, Furu Wei, and Ming Zhou. 2018. Neural Latent Extractive Document Summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 779–784. Association for Computational Linguistics.

Qingyu Zhou, Nan Yang, Furu Wei, Shaohan Huang, Ming Zhou, and Tiejun Zhao. 2018. Neural Document Summarization by Jointly Learning to Score and Select Sentences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 654–663. Association for Computational Linguistics.

# Domain Adaptive Text Style Transfer

**Dianqi Li**[1], **Yizhe Zhang**[2], **Zhe Gan**[2], **Yu Cheng**[2],
**Chris Brockett**[2], **Ming-Ting Sun**[1], **Bill Dolan**[2]
[1]University of Washington [2]Microsoft Corporation
{dianqili,mts}@uw.edu
{Yizhe.Zhang,Zhe.Gan,Yu.Cheng,Chris.Brockett,billdol}@microsoft.com

## Abstract

Text style transfer without parallel data has achieved some practical success. However, in the scenario where less data is available, these methods may yield poor performance. In this paper, we examine domain adaptation for text style transfer to leverage massively available data from other domains. These data may demonstrate domain shift, which impedes the benefits of utilizing such data for training. To address this challenge, we propose simple yet effective domain adaptive text style transfer models, enabling domain-adaptive information exchange. The proposed models presumably learn from the source domain to: $(i)$ distinguish stylized information and generic content information; $(ii)$ maximally preserve content information; and $(iii)$ adaptively transfer the styles in a domain-aware manner. We evaluate the proposed models on two style transfer tasks (sentiment and formality) over multiple target domains where only limited non-parallel data is available. Extensive experiments demonstrate the effectiveness of the proposed model compared to the baselines.

## 1 Introduction

Text style transfer, which aims to edit an input sentence with the desired style while preserving style-irrelevant content, has received increasing attention in recent years. It has been applied successfully to stylized image captioning (Gan et al., 2017), personalized conversational response generation (Zhang et al., 2018a), formalized writing (Rao and Tetreault, 2018), offensive to non-offensive language transfer (dos Santos et al., 2018), and other stylized text generation tasks (Akama et al., 2017; Zhang et al., 2019).

Text style transfer has been explored as a sequence-to-sequence learning task using parallel datasets (Jhamtani et al., 2017). However, parallel datasets are often not available, and hand-annotating sentences in different styles is expensive. The recent surge of deep generative models (Kingma and Welling, 2013; Goodfellow et al., 2014) has spurred progress in text style transfer without parallel data by learning disentanglement (Hu et al., 2017; Shen et al., 2017; Fu et al., 2018; Li et al., 2018; Prabhumoye et al., 2018). These methods typically require massive amounts of data (Subramanian et al., 2018), and may perform poorly in limited data scenarios.

A natural solution to the data-scarcity issue is to resort to massive data from other domains. However, directly leveraging abundant data from other domains is problematic due to the discrepancies in data distribution on different domains. Different domains generally manifest themselves in domain-specific lexica. For example, sentiment adjectives such as "*delicious*", "*tasty*", and "*disgusting*" in restaurant reviews might be out of place in movie reviews, where the sentiment words such as "*imaginative*", "*hilarious*", and "*dramatic*" are more typical. Domain shift (Gretton et al., 2009) is thus apt to result in feature misalignment.

In this work, we take up the problem of domain adaptation in scenarios where the target domain data is scarce and misaligned with the distribution in the source domain. Our goal is to achieve successful style transfer into the target domain, with the help of the source domain, while the transferred sentences carry relevant characteristics in the target domain.

We present two first-of-their-kind domain adaptive text style transfer models that facilitate domain-adaptive information exchange between the source and target domains. These models effectively learn generic content information and distinguish domain-specific information. Generic content information, primarily captured by modeling a large corpus from the source domain, fa-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3304–3313,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

cilitates better content preservation on the target domain. Meanwhile, domain-specific information, implicitly imposed by domain vectors and domain-specific style classifiers, underpins the transferred sentences by generating target-specific lexical terms.

Our contributions in this paper are threefold: $(i)$ We explore a challenging domain adaptation problem for text style transfer by leveraging massively-available data from other domains. $(ii)$ We introduce simple text style transfer models that preserve content and meanwhile translate text adaptively into target-domain-specific terms. $(iii)$ We demonstrate through extensive experiments the robustness of these methods for style transfer tasks (sentiment and formality) on multiple target domains where only limited non-parallel data is available. Our implementation is available at https://github.com/cookielee77/DAST.

## 2 Related Work

**Text Style Transfer.** Text style transfer using neural networks has been widely studied in the past few years. A common paradigm is to first disentangle latent space as content and style features, and then generate stylistic sentences by tweaking the style-relevant features and passing through a decoder. Hu et al. (2017); Fu et al. (2018); Shen et al. (2017); Yang et al. (2018); Gong et al. (2019); Lin et al. (2017) explored this direction by assuming the disentanglement can be achieved in an auto-encoding procedure with a suitable style regularization, implemented by either adversarial discriminators or style classifiers. Li et al. (2018); Xu et al. (2018); Zhang et al. (2018c) achieved disentanglement by filtering the stylistic words of input sentences. Recently, Prabhumoye et al. (2018) has proposed to use back-translation for text style transfer with a de-noising auto-encoding objective (Logeswaran et al., 2018; Subramanian et al., 2018). Our work differs in that we leverage domain adaptation to deal with limited target domain data, whereas previous methods require massive target domain style-labelled samples.

**Domain Adaptation.** Domain adaptation has been studied in various natural language processing tasks, such as sentiment classification (Qu et al., 2019), dialogue system (Wen et al., 2016), abstractive summarization (Hua and Wang, 2017; Zhang et al., 2018b), machine translation (Koehn and Schroeder, 2007; Axelrod et al., 2011; Sennrich et al., 2016b; Michel and Neubig, 2018), etc. However, little or no work explores domain adaptation on text style transfer. To the best of our knowledge, we are the first to explore the adaptation of text style transfer models to a new domain with limited non-parallel data available. The task requires both style transfer and domain-specific generation on the target domain. To differentiate different domains, Sennrich et al. (2016a); Chu et al. (2017) appended domain tokens to the input sentences. Our model uses learnable domain vectors combining domain-specific style classifiers, which force the model to learn distinct stylized information in each domain.

## 3 Preliminary

We first describe a standard text style transfer approach, which only considers data in the target domain. We limit our discussion to the scenario where only non-parallel data is available, since large amounts of parallel data is typically not feasible.

Given a set of style-labelled sentences $\mathcal{T} = \{(x_i, l_i)\}_{i=1}^{N}$ in the target domain, the goal is to transfer sentence $x_i$ with style $l_i$ to a sentence $\widetilde{x}_i$ with another style $\widetilde{l}_i$, where $\widetilde{l}_i \neq l_i$. $l_i, \widetilde{l}_i$ belong to a set of style labels $l^{\mathcal{T}}$ in the target domain: $l_i, \widetilde{l}_i \in l^{\mathcal{T}}$. Typically, an encoder encodes the input $x_i$ to a semantic representation $c_i$, while a decoder controls or modifies the stylistic property and decodes the sentence $\widetilde{x}_i$ based on $c_i$ and the pre-specific style $\widetilde{l}_i$.

Specifically, we denote an encoder-decoder model as $(E, D)$. The semantic representation $c_i$ of sentence $x_i$ is extracted by the encoder $E$, i.e., $c_i = E(x_i)$. The decoder $D$ aims to learn a conditional distribution of $\widetilde{x}_i$ given the semantic representation $c_i$ and style $\widetilde{l}_i$:

$$p_D(\widetilde{x}_i|c_i, \widetilde{l}_i) = \prod_{t=1}^{T} p_D(\widetilde{x}_i^t|\widetilde{x}_i^{<t}, c_i, \widetilde{l}_i), \quad (1)$$

where $\widetilde{x}_i^t$ is the $t^{th}$ token of $\widetilde{x}_i$, and $\widetilde{x}_i^{<t}$ is the prefix of $\widetilde{x}_i$ up to the $t^{th}$ token.

Directly estimating Eqn. (1) is impractical during training due to a lack of parallel data $(x_i, \widetilde{x}_i)$. Alternatively, the original sentence $x_i$ should have high probability under the conditional distribution $p_D(x_i|c_i, l_i)$. Thus, an auto-encoding reconstruc-

tion loss could be formulated as:

$$L_{ae}^{\mathcal{T}} = - \mathop{\mathbb{E}}_{x_i \sim \mathcal{T}} \log p_D(x_i|c_i, l_i)\,. \quad (2)$$

Note that we assume that the decoder $D$ recovers $x_i$'s original stylistic property as accurate as possible when given the style label $l_i$. To achieve text style transfer, the decoder manipulates the style of generated sentences by replacing $l_i$ with a desired style $\widetilde{l}_i$. Specifically, the generated sentence $\widetilde{x}_i$ is sampled from $\widetilde{x}_i \sim p_D(\widetilde{x}_i|c_i, \widetilde{l}_i)$. However, by directly optimizing Eqn. (2), the encoder-decoder model tends to ignoring the style labels and collapses to a reconstruction model, which might simply copy the input sentence, hence fails to transfer the style. To force the model to learn meaningful style properties, Hu et al. (2017, 2018) apply a style classifier for the style regularization. The style classifier ensures the encoder-decoder model to transfer $\widetilde{x}_i$ with its correct style label $\widetilde{l}_i$:

$$L_{style}^{\mathcal{T}} = - \mathop{\mathbb{E}}_{\widetilde{x}_i \sim p_D(\widetilde{x}_i|c_i, \widetilde{l}_i)} \log P_{C^{\mathcal{T}}}(\widetilde{l}_i|\widetilde{x}_i)\,, \quad (3)$$

where $C^{\mathcal{T}}$ is the style classifier pretrained on the target domain. The overall training objective for text style transfer within the target domain $\mathcal{T}$ is written as:

$$L^{\mathcal{T}} = L_{ae}^{\mathcal{T}} + L_{style}^{\mathcal{T}}\,. \quad (4)$$

## 4 Domain Adaptive Text Style Transfer

In this section, we present Domain Adaptive Style Transfer (DAST) models to perform style transfer on a target domain by borrowing the strength from a source domain, while maintaining the transfer to be domain-specific.

### 4.1 Problem Definition

Suppose we have two sets of style-labelled sentences $\mathcal{S} = \{(x'_i, l'_i)\}_{i=1}^{N'}$, $\mathcal{T} = \{(x_i, l_i)\}_{i=1}^{N}$ in the source domain $\mathcal{S}$ and the target domain $\mathcal{T}$, respectively. $x'_i$ denotes the $i^{th}$ source sentence. $l'_i$ denotes the corresponding style label, which belongs to a source style label set: $l'_i \in l^{\mathcal{S}}$ (e.g., positive/negative). $l'_i$ can be available or unknown. Likewise, pair $(x_i, l_i)$ represents the sentence and style label in the target domain, where $l_i \in l^{\mathcal{T}}$.

We consider domain adaptation in two settings: $(i)$ the source style $l^{\mathcal{S}}$ is unknown, e.g., we may have a large corpus, such as Yahoo! Answers, but the underlying style for each sample is not available; $(ii)$ the source styles are available, and are the same as the target styles, i.e., $l^{\mathcal{T}} = l^{\mathcal{S}}$, e.g., both IMDB movie reviews and Yelp restaurant reviews have the same style classes (negative and positive sentiments).

In both scenarios, we assume that the target domain $\mathcal{T}$ only has limited non-parallel data. With the help of source domain data $\mathcal{S}$, the goal is to transfer $(x_i, l_i)$ to $(\widetilde{x}_i, \widetilde{l}_i)$ in the target domain. The transferred sentence $\widetilde{x}_i$ should simultaneously hold: $(i)$ the main content with $x_i$, $(ii)$ a different style $\widetilde{l}_i$ from $l_i$, and $(iii)$ domain-specific characteristics of the target data distribution $\mathcal{T}$.

### 4.2 DAST with unknown-stylized source data

In this section, we investigate the case that the source style $l^{\mathcal{S}}$ is unknown. We first examine a drawback of limited target data to motivate our method. With limited target data, Eqn. (4) may yield an undesirable transferred text, where the generated text tends to use the most discriminative words that the target style prefers while ignoring the content. This is because the classifier $C^{\mathcal{T}}$ typically requires less data to train than a sequence autoencoder $(E, D)$. The classifier objective $L_{style}^{\mathcal{T}}$ thus dominates Eqn. (4), rendering the generator to bias the sentences with most representative stylized (e.g., positive or negative) words rather than preserving the contents (see Table 5 for examples).

We consider alleviating this issue by leveraging massive source domain data to enhance the content-preserving ability, though the underlying styles in the source domain are unknown. By jointly training an auto-encoder on both the source and target domain data, the learned generic content information enables the model to yield better content preservation on the target domain.

To utilize the source data, we consider that $l^{\mathcal{S}}$ only contains a special unknown-style label $l^u$, separated from the target style $l^{\mathcal{T}}$. We assume the semantic representation of the source data $c'_i$ is encoded by the encoder, i.e., $c'_i = E(x'_i)$. The decoder takes $c'_i$ with style $l^u$ to generate the sentences on the source domain. The auto-encoding reconstruction objective of the source domain is:

$$L_{ae}^{\mathcal{S}} = - \mathop{\mathbb{E}}_{x'_i \sim \mathcal{S}} \log p_D(x'_i|c'_i, l^u), \quad (5)$$

where the encoder-decoder model $(E, D)$ is shared in both domains. Therefore, the corresponding objective can be written as:

$$L_{\text{DAST-C}} = L_{ae}^{\mathcal{T}} + L_{style}^{\mathcal{T}} + L_{ae}^{\mathcal{S}}\,. \quad (6)$$

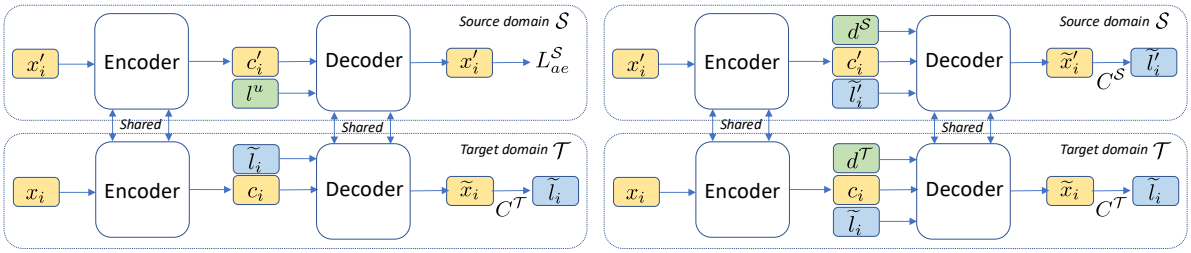


Figure 1: Illustration of the proposed **DAST-C** (left) and **DAST** (right) model. DAST-C learns the generic content information through $L_{ae}^{\mathcal{S}}$ on massive source domain data with unknown style $l^u$. For DAST, $d^{\mathcal{T}}, d^{\mathcal{S}}$ and $C^{\mathcal{T}}, C^{\mathcal{S}}$ denote domain vectors and domain-specific style classifiers, respectively. Better looked in color.

This can be perceived as combining the source domain data with the target domain data to train a better encoder-decoder framework, while target-specific style information on the target domain is learned through $L_{style}^{\mathcal{T}}$.

Note that $L_{ae}^{\mathcal{T}}$ and $L_{ae}^{\mathcal{S}}$ are conditional on domain-specific styles labels: $l^{\mathcal{T}}$ and $l^u$, which implicitly encourages the model to learn domain-specific features. The decoder could thus generate target sentences adaptively with $l^{\mathcal{T}}$, while achieving favorable content preservation with the generic content information modeled by $L_{ae}^{\mathcal{S}}$. We refer this model, which is illustrated in Figure 1(left), as *Domain Adaptive Style Transfer with generic Content preservation (DAST-C)*.

### 4.3 DAST with stylized source data

We further explore the scenario where $l^{\mathcal{S}} = l^{\mathcal{T}}$. In this case, besides the generic content information, there is much style information from the source domain that could be leveraged, e.g., generic stylized expressions like "*fantastic*" and "*terrible*" for sentiment transfer can be applied to both restaurant and movie reviews. We thus consider to borrow the full strength of the source data, by sharing learned knowledge on both the generic content and style information.

A straightforward way to achieve this is to train Eqn. (4) on both domains. However, simply mixing the two domains together will lead to undesirable style transfers, where the transfer is not domain-specific. For example, when adapting the IMDB movie reviews to the Yelp restaurant reviews, directly sharing the style transfer model without specifying the domain will inevitably result in generations like *"The pizza is dramatic!"*.

To alleviate this problem, we introduce additional domain vectors, encouraging the model to perform style transfer in a domain-aware manner. The proposed DAST model is illustrated in Figure 1(right). Consider two domain vectors: $d^{\mathcal{S}}$ for the source domain and $d^{\mathcal{T}}$ for the target domain, respectively. We rewrite the auto-encoding loss as:

$$\begin{aligned} L_{ae}^{\mathcal{S},\mathcal{T}} = &- \mathop{\mathbb{E}}_{x'_i \sim \mathcal{S}} \log p_D(x'_i | c'_i, d^{\mathcal{S}}, l'_i) \\ &- \mathop{\mathbb{E}}_{x_i \sim \mathcal{T}} \log p_D(x_i | c_i, d^{\mathcal{T}}, l_i)\,, \end{aligned} \tag{7}$$

where the encoder-decoder model $(E, D)$ is shared across domains. The domain vectors, $d^{\mathcal{S}}$, $d^{\mathcal{T}}$, learned from the model, implicitly guide the decoder to generate sentences with domain-specific characteristics. Note that $l_i$ and $l'_i$ are shared, i.e., $l^{\mathcal{T}} = l^{\mathcal{S}}$. This enables the model to learn generic style information from both domains. On the other hand, explicitly learning precise stylized information within each domain is crucial to generate domain-specific styles. Thus, two domain-specific style classifiers ensure the model to learn the corresponding styles by conditioning on $(d^{\mathcal{S}}, \widetilde{l}'_i)$ in the source domain or $(d^{\mathcal{T}}, \widetilde{l}_i)$ in the target domain:

$$\begin{aligned} L_{style}^{\mathcal{S},\mathcal{T}} = &- \mathop{\mathbb{E}}_{\widetilde{x}'_i \sim p_D(\widetilde{x}'_i | c'_i, d^{\mathcal{S}}, \widetilde{l}'_i)} \log P_{C^{\mathcal{S}}}(\widetilde{l}'_i | \widetilde{x}'_i) \\ &- \mathop{\mathbb{E}}_{\widetilde{x}_i \sim p_D(\widetilde{x}_i | c_i, d^{\mathcal{T}}, \widetilde{l}_i)} \log P_{C^{\mathcal{T}}}(\widetilde{l}_i | \widetilde{x}_i)\,, \end{aligned} \tag{8}$$

where $\widetilde{x}'_i, \widetilde{x}_i$ are the transferred sentences with pre-specific styles $\widetilde{l}'_i, \widetilde{l}_i$ in the source and target domains, respectively. The domain-specific style classifiers, $C^{\mathcal{T}}$ and $C^{\mathcal{S}}$, are trained separately on each domain. The signals from classifiers encourage the model to learn domain-specific styles combining with the domain vectors and style labels. The overall training objective of the proposed DAST model is:

$$L_{\text{DAST}} = L_{ae}^{\mathcal{S},\mathcal{T}} + L_{style}^{\mathcal{S},\mathcal{T}}\,. \tag{9}$$

The domain-specific style classifiers enforce the model to learn domain-specific style information

conditioning on $(d^{\mathcal{S}}, \widetilde{l}'_i)$ or $(d^{\mathcal{T}}, \widetilde{l}_i)$, which in turn controls the model to generate sentences with domain-specific words. The model can thus distinguish domain-specific features, and adaptively transfer the styles in a domain-aware manner.

## 5 Experiments

We evaluate our proposed models on two tasks: sentiment transfer (positive-to-negative and negative-to-positive), and formality transfer (informal-to-formal). In both tasks, we make comparisons with previous approaches over multiple target domains. All experiments are conducted on one Nvidia GTX 1080Ti GPU.

### 5.1 Dataset

A statistics for the source and target corpora used in the experiments is summarized in Table 1.

| Sentiment Transfer | | | | | |
|---|---|---|---|---|---|
| Source | Train | Target | Train | Dev | Test |
| IMDB | 344k | YELP | 444k | 4k | 1k |
| | | AMAZON | 554k | 2k | 1k |
| | | YAHOO | 4k | 2k | 1k |
| Formality Transfer | | | | | |
| Source | Train | Target | Train | Dev | Test |
| GYAFC | 103k | ENRON | 6k | 500 | 500 |

Table 1: Statistics of source and target datasets.

**Sentiment Transfer.** For the source domain, we use IMDB movie review corpus (Diao et al., 2014) by following the filtering and preprocessing pipelines from Shen et al. (2017). This results in $344k$ training samples with sentiment labels. For the target domain, both the Yelp restaurant review dataset and the Amazon product review dataset are from Li et al. (2018). For the test sets, we evaluate our methods by using $1k$ human-transferred sentences, annotated by Li et al. (2018), on both Yelp and Amazon datasets. In addition to the two standard sentiment datasets, we manually collected a Yahoo sentimental question dataset - $7k$ question samples with sentiments from Yahoo! Answers dataset (Zhang et al., 2015). We split the $7k$ sentimental questions into $4k$/$2k$/$1k$ for train/dev/test sets, respectively. Note that the Yahoo sentiment dataset only consists of questions, which have different domain characteristics with the IMDB dataset. In all the sentiment experiments, we consider both transfer directions (positive-to-negative and negative-to-positive).

**Formality Transfer.** We use Grammarly's Yahoo Answers Formality Corpus (GYAFC) (Rao and Tetreault, 2018) as the source dataset. The publicly released version of GYAFC only covers two topics (*Entertainment & Music* and *Family & Relationships*), where each topic contains $50k$ paired informal and formal sentences written by humans. For the target domain, we use Enron email conversation dataset[1], which covers several different fields like *Business, Politics, Daily Life, etc.* We manually labeled $7k$ *non-parallel* sentences written in either the formal or informal style. We split the Enron dataset into $6k, 500, 500$ samples for training, validation and testing, respectively. Both the validation and test set consist of mere informal sentences, where the corresponding formal references are annotated by us from a crowd-sourcing platform for evaluation. We only assess the informal-to-formal transfer direction in the formality transfer experiment.

### 5.2 Evaluation

**Automatic Metrics.** We evaluate the effectiveness of our DAST models based on three automatic metrics:

($i$) Content Preservation. We assess the content preservation according to n-gram statistics, by measuring the BLEU scores (Papineni et al., 2002) between generated sentences and human references on the target domain, refered as *human* BLEU (*h*BLEU). When no human reference is available (e.g., Yahoo), we compute the BLEU scores with respect to the input sentences.

($ii$) Style Control. We generate samples from the model and measure the style accuracy with a style classifier that is pre-trained on the target domain. We refer the style accuracy as S-acc.

($iii$) Domain Control. To validate whether the generated sentences hold the characteristics of the target domain, we adopt a pre-trained domain classifier to measure the percentage of generated sentences that belong to the target domain. We refer the domain accuracy as D-acc.

All the pre-trained classifiers are implemented by TextCNN (Kim, 2014; Zhang et al., 2017). The test accuracy of all these classifiers used for evaluation are reported in Appendix A.1. Following Xu et al. (2018), we also evaluate all methods using a single unified metric called G-score, which calculates the geometric mean of style accuracy and *h*BLEU.

[1] https://www.cs.cmu.edu/~./Enron/

| | Yelp | | | | Amazon | | | |
|---|---|---|---|---|---|---|---|---|
| Model (100% target data) | D-acc | S-acc | *h*BLEU | G-score | D-acc | S-acc | *h*BLEU | G-score |
| CrossAlign (Shen et al., 2017) | - | 85.0 | 3.7 | 8.3 | - | 23.0 | 34.1 | 18.0 |
| Delete&Retrieve (Li et al., 2018) | - | 90.6 | 14.8 | 17.9 | - | 50.9 | 30.3 | 25.7 |
| CycleRL (Xu et al., 2018) | - | 88.7 | 12.3 | 16.4 | - | 68.7 | 14.2 | 15.5 |
| SMAE (Zhang et al., 2018c) | - | 85.1 | 12.1 | 15.5 | - | 71.1 | 12.9 | 14.9 |
| ControlGen (Hu et al., 2018) | | 91.5 | 25.5 | 27.4 | - | 79.0 | 31.1 | 30.5 |
| Finetune | **96.1** | 91.3 | 25.6 | 27.8 | **97.4** | 79.2 | 34.1 | 34.3 |
| DAST-C (ours) | 93.8 | 91.7 | 25.7 | 27.5 | 96.7 | 81.9 | 35.7 | 35.0 |
| DAST (ours) | 95.8 | **92.3** | **26.3** | **28.9** | 96.9 | **83.0** | **35.9** | **35.1** |
| Model (1% target data) | D-acc | S-acc | *h*BLEU | G-score | D-acc | S-acc | *h*BLEU | G-score |
| CrossAlign (Shen et al., 2017) | - | 76.3 | 4.8 | 8.5 | - | 83.2 | 2.0 | 5.9 |
| Delete&Retrieve (Li et al., 2018) | - | 82.1 | 4.1 | 7.6 | - | 63.0 | 6.9 | 9.3 |
| CycleRL (Xu et al., 2018) | - | 86.6 | 1.4 | 5.2 | - | 79.5 | 0.7 | 3.8 |
| SMAE (Zhang et al., 2018c) | - | 96.0 | 1.2 | 4.8 | - | 87.2 | 0.4 | 3.2 |
| ControlGen (Hu et al., 2018) | - | **98.5** | 3.7 | 8.6 | - | 83.2 | 1.9 | 5.8 |
| Finetune | **98.1** | 96.7 | 13.9 | 18.5 | **96.0** | **89.2** | 11.3 | 14.4 |
| DAST-C (ours) | 96.9 | 90.3 | 17.8 | 19.3 | 94.8 | 78.2 | 20.1 | 21.6 |
| DAST (ours) | 97.0 | 92.6 | **20.1** | **23.1** | 94.6 | 82.7 | **21.0** | **23.1** |

Table 2: Automatic evaluation results on Yelp and Amazon test sets. D-acc and S-acc denote domain accuracy and style accuracy, respectively. G-score is the geometric mean of S-acc and *h*BLEU.

**Human Evaluation.** To assess the quality of transferred sentences, we conduct human evaluations based on the facets of *content preservation*, *style control* and *fluency*, following Mir et al. (2019). Previous works (Subramanian et al., 2018; Gong et al., 2019) ask workers to evaluate the quality via a numerical score, however, we found that this empirically leads to high-variance results. Instead, we pair transferred sentences from two different models, and ask workers to choose the sentence they prefer *when compared to the input* on each evaluation aspect. We provide a "No Preference" option to choose when the workers think the qualities of the two sentences are indistinguishable. Details of the human evaluation instruction are included in Appendix A.3. For each testing, we randomly sample 100 sentences from the corresponding test set and collect three human responses for each pair on every evaluation aspect, resulting in 2700 responses in total.

### 5.3 Experimental Setup

The encoder $E$ and the decoder $D$ are implemented by one-layer GRU (Cho et al., 2014) with hidden dimensions 500 and 700, respectively. The domain-vector dimension is set to 50. The style labels are represented by learnable vectors with 150 dimensions. The decoder is initialized by a concatenation of representations of content, style, and domain vectors. If domain vectors are not used, the dimension of style labels is set to 200; accordingly, the initialization of the decoder is a concatenation of content and style representations. TextCNN (Kim, 2014) is employed for the domain-specific style classifiers pre-trained on corresponding domains. After pre-training, the parameters of the classifiers are fixed. We use the hard-sampling trick (Logeswaran et al., 2018) to back-propagate the loss through discrete tokens from the classifier to the encoder-decoder model. During training, we assign each mini-batch the same amount of source and target data to balance the training.

We make an extensive comparison with five state-of-the-art text style transfer models: CrossAlign (Shen et al., 2017), Delete&Retrieve (Li et al., 2018), CycleRL (Xu et al., 2018), SMAE (Zhang et al., 2018c) and ControlGen (Hu et al., 2018). We also experiment a simple and effective domain adaptation baseline - Finetune, which is trained with Eqn. (4) on the source domain and then fine-tuned on the target domain.

### 5.4 Results

**Model Comparisons.** To evaluate the effectiveness of leveraging massive data from other domains, we compare our proposed DAST models with previously proposed models trained on the target domain (Table 2). We observe that by leveraging massive data from the IMDB dataset, our models achieve better performance against all

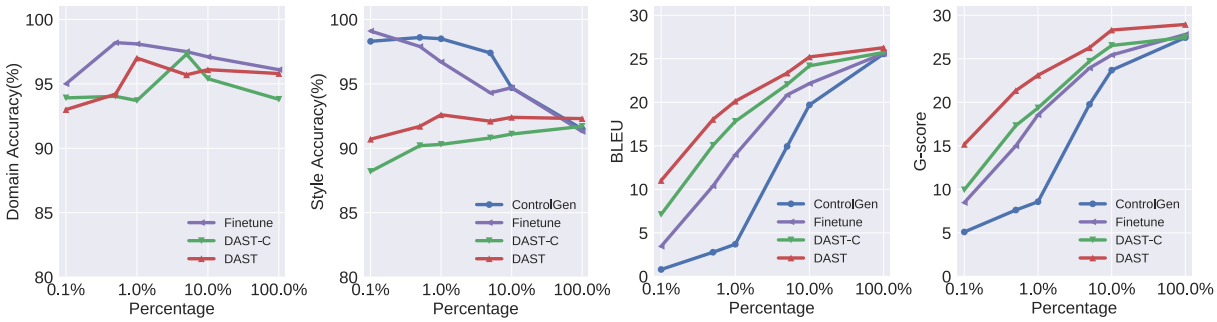


Figure 2: Results on Yelp test set in terms of different percentage of target domain data. $0.1\% \approx 400$ samples.

| Style Control (Yelp 1% data) | | | | | Content Preservation (Yelp 1% data) | | | | | Fluency (Yelp 1% data) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Our Model | | Neutral | Comparison | | Our Model | | Neutral | Comparison | | Our Model | | Neutral | Comparison | |
| DAST | **56.2%** | 30.5% | 13.3% | ControlGen | DAST | **47.0%** | 48.4% | 4.6% | ControlGen | DAST | **47.1%** | 40.8% | 12.0% | ControlGen |
| DAST | **40.5%** | 42.3% | 17.2% | DAST-C | DAST | **22.4%** | 65.7% | 11.9% | DAST-C | DAST | **29.1%** | 55.8% | 15.1% | DAST-C |
| DAST | 17.9% | 18.5% | **63.6%** | Human | DAST | 17.7% | 47.4% | **34.9%** | Human | DAST | 10.1% | 30.4% | **59.5%** | Human |
| **Style Control (Enron)** | | | | | **Content Preservation (Enron)** | | | | | **Fluency (Enron)** | | | | |
| Our Model | | Neutral | Comparison | | Our Model | | Neutral | Comparison | | Our Model | | Neutral | Comparison | |
| DAST | **74.2%** | 19.8% | 6% | ControlGen | DAST | **80.8%** | 14.8% | 4.4% | ControlGen | DAST | **73.8%** | 20.6% | 5.6% | ControlGen |
| DAST | **28.4%** | 50.2% | 21.4% | DAST-C | DAST | **26.8%** | 48.8% | 24.4% | DAST-C | DAST | **26.9%** | 51.6% | 21.5% | DAST-C |
| DAST | 17.6% | 30.5% | **51.9%** | Human | DAST | 15.3% | 36.9% | **47.8%** | Human | DAST | 11.6% | 36.5% | **51.9%** | Human |

Table 3: Results of **Human Evaluation** for style control, content preservation and fluency, showing preferences (%) for DAST model vis-a-vis baseline or other comparison systems. Evaluation results of the **overall transfer quality** are provided in Appendix A.3.

| Yahoo Sentiment Transfer | | | |
|---|---|---|---|
| Model | D-acc | S-acc | BLEU |
| ControlGen | - | 99.1 | 9.7 |
| Finetune | **97.8** | 98.8 | 31.4 |
| DAST-C | 90.7 | 98.8 | 35.9 |
| DAST | 90.8 | **99.2** | **39.2** |

Table 4: Results on Yahoo sentiment transfer task.

baselines on the sentiment transfer tasks in both the Yelp and Amazon domains.

Notably, when the target domain has limited data (1%), all baselines trained only on the target domain fail completely on content preservation. Finetune preserves better content but experiences the catastrophic forgetting problem (Goodfellow et al., 2013) to the source domain information. As a result, the overall style transfer performance is still nonoptimal. By contrast, with the help of the source domain, DAST obtains considerable content preservation performance improvement when compared with other baselines. Our model also attains favorable performance in terms of style transferring accuracy (S-acc), resulting in a good overall G-score. In general, we observe that DAST-C is able to better preserve content information, while DAST further improves both content preservation and style control abilities. Additionally, both DAST-C and DAST can adapt to the target domain, as evidenced by the high domain accuracy (D-acc). The human evaluation results (Table 3) show a strong preference of DAST over DAST-C as well as ControlGen in terms of style control, content preservation and fluency.

Finally, we evaluate our models on Yahoo sentiment transfer task. As can be seen in Table 4, both DAST and DAST-C achieve successful style transfer even if the target data is formed as questions which have a large discrepancy with the source IMDB domain. The samples of Yelp and Yahoo sentiment transfer are shown in Table 5. We also investigate the effect of different source domain data, included in Appendix A.2.

**Limiting the Target Domain Data.** We further test the limit of our model by using as few target domain data as possible. Figure 2 shows the quantitative results with different percentages of target domain training data. When the target domain data is insufficient, especially less than 10%, the content preservation ability of the baseline (trained with target data only) has degenerated rapidly despite a relatively high style transfer accuracy. This is less than desirable because by retrieving sentences with the target style a transferred sentence

| | Yelp (positive-to-negative) | Yelp (negative-to-positive) |
|---|---|---|
| **Input** | the service was great , food delicious , and the value impeccable . | and the pizza was cold , greasy , and generally quite awful . |
| **ControlGen** | the service was horrible , service , the service and very frustrated . | and the food was delicious, delicious , and freaking tasty , delicious . |
| **Finetune** | the service was poor , food ... , and the experience were . | and the pizza was professional , friendly , and always have great . |
| **DAST-C** | the service was horrible , food horrible , and the slow sparse . | and the pizza was fresh, greasy , and generally quite cool . |
| **DAST** | the service was horrible , food bland , and the value lousy . | and the pizza was tasty , juicy , and definitely quite amazing . |
| **Human** | service was poor and the food expensive and weak tasting . | the pizza was warm , not greasy , and generally tasted great . |
| | Yahoo (positive-to-negative) | Yahoo (negative-to-positive) |
| **Input** | who is more romantic ? man or woman ? | why do stupid questions constantly receive intelligent answers ? |
| **ControlGen** | which is more stupid ? and or why ? | men do fantastic questions constantly receive intelligent bound ! |
| **Finetune** | the is more expensive ? man or woman ? | why do great questions read more entertaining answers ? |
| **DAST-C** | who is more ugly ? man or woman ? | why do important questions constantly receive intelligent answers ? |
| **DAST** | who is more crazy ? man or woman ? | why do nice questions constantly receive intelligent answers ? |
| | Enron (informal-to-formal) | Enron (informal-to-formal) |
| **Input** | ya 'll need to come visit us in austin . | are n't you suppose to be teaching some kids or something ? |
| **ControlGen** | could we need to look on saturday in enpower . | are you not supposed to be disloyal some kids or something ? |
| **Finetune** | you will need to go in bed with him . | are you not to be able to be some man or something ? |
| **DAST-C** | you will need to visit town . | are not you supposed to be teaching some kids or something ? |
| **DAST** | yes , you will need to visit us in austin . | are you not supposed to be teaching some children or something ? |
| **Human** | all of you should come visit us in austin . | are you not supposed to be instructing children ? |

Table 5: Transferred sentences on Yelp (1% data), Yahoo and Enron datasets, where red denotes successful style transfers, blue denotes content losses, and orange denotes grammar errors. Better looked in color.

| Model | D-acc | S-acc | *h*BLEU | G-score |
|---|---|---|---|---|
| DAST | **97.0** | **92.6** | **20.1** | **23.1** |
| w/o d-spec attributes | 83.9 | 90.9 | 20.0 | 22.7 |
| w/o d-spec classifiers | 91.4 | 83.8 | 19.0 | 20.8 |
| w/o both | 73.8 | 80.6 | 18.7 | 19.9 |
| Setup | D-acc | S-acc | *h*BLEU | G-score |
| IMDB+Yelp | 97.0 | 92.6 | 20.1 | 23.1 |
| Finetune | 98.1 | 96.7 | 13.9 | 18.5 |
| IMDB | 62.8 | 59.3 | 21.4 | 12.2 |
| Yelp | 96.8 | 98.5 | 3.7 | 8.6 |

Table 6: Ablation study on Yelp (1%) dataset with help from IMDB dataset. The results are evaluated on Yelp test set. *d-spec* is short for *domain-specific*.

| Enron Formality Transfer | | | |
|---|---|---|---|
| Model | D-acc | S-acc | *h*BLEU |
| ControlGen | - | 81.2 | 4.74 |
| Finetune | **91.3** | 81.6 | 14.7 |
| DAST-C | 87.6 | 89.2 | 15.5 |
| DAST | 88.4 | **91.6** | **16.4** |

Table 7: Results on Enron formality transfer tasks.

can easily exhibit the correct style while retaining barely any content similar to the input. Finetune improves content preservation but still suffers the same problem with less target data. Note that DAST-C is not comparable to Finetune as the former does not use the style information in the source domain.

Both DAST models bring substantial improvements to content preservation, and can still successfully manipulate the styles, resulting in consistently higher G-scores. This is presumably because our models adapt the content information as well as the style information from the source domain to consistently sustain the style transfer on the target domain. By learning both generic and domain-specific stylized information, DAST outperforms DAST-C in terms of content preservation and style control. Even with 0.1% target domain data (400 samples), DAST was able to attain a reasonable degree of text style transfer, whereas the model trained on the target data generated entirely nonsensical sentences. Meanwhile, DAST succeeded in transferring the sentences in a domain-aware manner, achieving consistently high domain accuracy.

**Ablation Study.** To investigate the effect of individual components and training setup on the overall performance, we conduct an ablation study in Table 6. The domain vectors enable the model to transfer sentences in a domain-aware manner, and thus give the largest boost on domain accuracy. Without domain-specific style classifiers, the model mixes the style information on both domains, resulting in worse style control and content preservation. Additionally, simply increasing the number of training examples (i.e., the row "w/o both") improves content preserving, while introducing a data distribution discrepancy between the training (Yelp+IMDB) and test data (Yelp), as evidenced by the lower S-acc and D-acc scores.

In terms of the training setup, the source domain IMDB mostly helps content preservation, while accurate style information is mainly learned from the target domain Yelp. Finetune gives higher S-acc and D-acc and lower *h*BLEU due to catastrophic forgetting. Our proposed DAST successfully exploits the source domain data, and thus yields balanced results on style and domain control, while maintaining content preservation.

**Non-parallel Style Transfer with Parallel Source Data.** Finally, to verify the versatility of our proposed models in different scenarios, we investigate another domain adaptation setting, where the source domain data (GYAFC) is parallel but the target domain data (Enron) is non-parallel. Since parallel data is available in the source domain, we are able to simply add a sequence-to-sequence loss $L^{\mathcal{S}}_{s2s}$ on source domain data in Eqn. (6) and Eqn. (9) to help the target domain without parallel data. The training objectives can be written as: $L^{\mathcal{T}}_{ae}+L^{\mathcal{T}}_{style}+L^{\mathcal{S}}_{ae}+L^{\mathcal{S}}_{s2s}$ and $L^{\mathcal{S},\mathcal{T}}_{ae}+L^{\mathcal{S},\mathcal{T}}_{style}+L^{\mathcal{S}}_{s2s}$, respectively. Results are summarized in Table 7. DAST outperforms other methods on both style control and content preservation while keeping the transferred sentences with target-specific characteristics (D-acc). A strong human preference for DAST can be observed in Table 3 when compared to the baselines. Qualitative samples are provided in Table 5.

## 6 Conclusion

We present two simple yet effective domain adaptive text style transfer models that leverage massively available data from other domains to facilitate the transfer task in the target domain. The proposed models achieve better content preservation with the generic information learned from the source domain and simultaneously distinguish the domain-specific information, which enables the models to transfer text in a domain-adaptive manner. Extensive experiments demonstrate the robustness and applicability on various scenarios where the target data is limited.

## Acknowledgments

We would like to thank the reviewers for their constructive comments. We thank NVIDIA Corporation for the donation of the GPU used for this research. We also thank Hao Peng and Tianyi Zhou for their helpful discussions.

## References

Reina Akama, Kazuaki Inada, Naoya Inoue, Sosuke Kobayashi, and Kentaro Inui. 2017. Generating stylistically consistent dialog responses with transfer learning. In *IJCNLP*.

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. 2011. Domain adaptation via pseudo in-domain data selection. In *EMNLP*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP*.

Chenhui Chu, Raj Dabre, and Sadao Kurohashi. 2017. An empirical comparison of simple domain adaptation methods for neural machine translation. *arXiv preprint arXiv:1701.03214*.

Qiming Diao, Minghui Qiu, Chao-Yuan Wu, Alexander J Smola, Jing Jiang, and Chong Wang. 2014. Jointly modeling aspects, ratings and sentiments for movie recommendation (jmars). In *SIGKDD*.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2018. Style transfer in text: Exploration and evaluation. In *AAAI*.

Chuang Gan, Zhe Gan, Xiaodong He, Jianfeng Gao, and Li Deng. 2017. Stylenet: Generating attractive visual captions with styles. In *CVPR*.

Hongyu Gong, Suma Bhat, Lingfei Wu, Jinjun Xiong, and Wen-mei Hwu. 2019. Reinforcement learning based text style transfer without parallel training corpus. *arXiv preprint arXiv:1903.10671*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *NeurIPS*.

Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. 2013. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*.

Arthur Gretton, AJ. Smola, Jiayuan Huang, Marcel Schmittfull, Karsten Borgwardt, and Bernhard Schölkopf. 2009. *Covariate shift and local learning by distribution matching*. MIT Press.

Zhiting Hu, Haoran Shi, Zichao Yang, Bowen Tan, Tiancheng Zhao, Junxian He, Wentao Wang, Lianhui Qin, Di Wang, et al. 2018. Texar: A modularized, versatile, and extensible toolkit for text generation. *arXiv preprint arXiv:1809.00794*.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *ICML*.

Xinyu Hua and Lu Wang. 2017. A pilot study of domain adaptation effect for neural abstractive summarization. In *Proceedings of the Workshop on New Frontiers in Summarization*.

Harsh Jhamtani, Varun Gangal, Eduard Hovy, and Eric Nyberg. 2017. Shakespearizing modern language using copy-enriched sequence to sequence models. In *Proceedings of the Workshop on Stylistic Variation*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Philipp Koehn and Josh Schroeder. 2007. Experiments in domain adaptation for statistical machine translation. In *Proceedings of the second workshop on statistical machine translation*.

Juncen Li, Robin Jia, He He, and Percy Liang. 2018. Delete, retrieve, generate: a simple approach to sentiment and style transfer. In *NAACL*.

Kevin Lin, Dianqi Li, Xiaodong He, Zhengyou Zhang, and Ming-Ting Sun. 2017. Adversarial ranking for language generation. In *NeurIPS*.

Lajanugen Logeswaran, Honglak Lee, and Samy Bengio. 2018. Content preserving text generation with attribute controls. In *NeurIPS*.

Paul Michel and Graham Neubig. 2018. Extreme adaptation for personalized neural machine translation. In *ACL*.

Remi Mir, Bjarke Felbo, Nick Obradovich, and Iyad Rahwan. 2019. Evaluating style transfer for text. *arXiv preprint arXiv:1904.02295*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W Black. 2018. Style transfer through back-translation. In *ACL*.

Xiaoye Qu, Zhikang Zou, Yu Cheng, Yang Yang, and Pan Zhou. 2019. Adversarial category alignment network for cross-domain sentiment classification. In *NAACL*.

Sudha Rao and Joel Tetreault. 2018. Dear sir or madam, may i introduce the gyafc dataset: Corpus, benchmarks and metrics for formality style transfer. In *NAACL*.

Cicero Nogueira dos Santos, Igor Melnyk, and Inkit Padhi. 2018. Fighting offensive language on social media with unsupervised text style transfer. In *ACL*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Controlling politeness in neural machine translation via side constraints. In *NAACL*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Improving neural machine translation models with monolingual data. In *ACL*.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *NeurIPS*.

Sandeep Subramanian, Guillaume Lample, Eric Michael Smith, Ludovic Denoyer, Marc'Aurelio Ranzato, and Y-Lan Boureau. 2018. Multiple-attribute text style transfer. *arXiv preprint arXiv:1811.00552*.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Lina M Rojas-Barahona, Pei-Hao Su, David Vandyke, and Steve Young. 2016. Multi-domain neural network language generation for spoken dialogue systems. In *NAACL*.

Jingjing Xu, Sun Xu, Qi Zeng, Xiaodong Zhang, Xuancheng Ren, Houfeng Wang, and Wenjie Li. 2018. Unpaired sentiment-to-sentiment translation: A cycled reinforcement learning approach. In *ACL*.

Zichao Yang, Zhiting Hu, Chris Dyer, Eric P Xing, and Taylor Berg-Kirkpatrick. 2018. Unsupervised text style transfer using language models as discriminators. In *NeurIPS*.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018a. Personalizing dialogue agents: I have a dog, do you have pets too? In *ACL*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NeurIPS*.

Ye Zhang, Nan Ding, and Radu Soricut. 2018b. Shaped: Shared-private encoder-decoder for text style adaptation. In *NAACL*.

Yi Zhang, Jingjing Xu, Pengcheng Yang, and Xu Sun. 2018c. Learning sentiment memories for sentiment modification without parallel data. In *EMNLP*.

Yizhe Zhang, Xiang Gao, Sungjin Lee, Chris Brockett, Michel Galley, Jianfeng Gao, and Bill Dolan. 2019. Consistent dialogue generation with self-supervised feature learning. *arXiv preprint arXiv:1903.05759*.

Yizhe Zhang, Dinghan Shen, Guoyin Wang, Zhe Gan, Ricardo Henao, and Lawrence Carin. 2017. Deconvolutional paragraph representation learning. In *NeurIPS*.

# Let's Ask Again: Refine Network for Automatic Question Generation

**Preksha Nema**[†‡*] **Akash Kumar Mohankumar**[†*] **Mitesh M. Khapra**[†‡]
**Balaji Vasan Srinivasan**[•] **Balaraman Ravindran**[†‡]
[†]IIT Madras, India [•]Adobe Research
[‡] Robert Bosch Center for Data Science and Artificial Intelligence, IIT Madras
{preksha,miteshk,ravi}@cse.iitm.ac.in
makashkumar99@gmail.com {balsrini}@adobe.com

## Abstract

In this work, we focus on the task of Automatic Question Generation (AQG) where given a passage and an answer the task is to generate the corresponding question. It is desired that the generated question should be (i) grammatically correct (ii) answerable from the passage and (iii) specific to the given answer. An analysis of existing AQG models shows that they produce questions which do not adhere to one or more of the above-mentioned qualities. In particular, the generated questions look like an incomplete draft of the desired question with a clear scope for refinement. To alleviate this shortcoming, we propose a method which tries to mimic the human process of generating questions by first creating an initial draft and then refining it. More specifically, we propose Refine Network (RefNet) which contains two decoders. The second decoder uses a dual attention network which pays attention to both (i) the original passage and (ii) the question (initial draft) generated by the first decoder. In effect, it refines the question generated by the first decoder, thereby making it more correct and complete. We evaluate RefNet on three datasets, *viz.*, SQuAD, HOTPOT-QA, and DROP, and show that it outperforms existing state-of-the-art methods by 7-16% on all of these datasets. Lastly, we show that we can improve the quality of the second decoder on specific metrics, such as, fluency and answerability by explicitly rewarding revisions that improve on the corresponding metric during training. The code has been made publicly available [1].

## 1 Introduction

Over the past few years, there has been a growing interest in Automatic Question Generation (AQG) from text - the task of generating a question from a passage and optionally an answer. AQG is used in curating Question Answering datasets, enhancing user experience in conversational AI systems (Shum et al., 2018) and for creating educational materials (Heilman and Smith, 2010). For the above applications, it is essential that the questions are (i) grammatically correct (ii) answerable from the passage and (iii) specific to the answer. Existing approaches focus on encoding the passage, the answer and the relationship between them using complex functions and then generate the question in *one* single pass. However, by carefully analysing the generated questions, we observe that these approaches tend to miss one or more of the important aspects of the question. For instance, in Table 1, the question generated by the single-pass baseline model for the first passage is grammatically correct but is not specific to the answer. In the second example, the generated question is both syntactically incorrect and incomplete.

| | |
|---|---|
| **Passage 1:** Liberated by Napoleon's army in 1806, Warsaw was made the capital of the newly created Duchy of Warsaw. | |
| **Generated Questions** | |
| *Baseline* | What was the capital of the newly duchy of Warsaw? |
| *RefNet* | Who liberated Warsaw in 1806? |
| *Reward-RefNet* | Whose army liberated Warsaw in 1806? |
| **Passage 2:** To fix carbon dioxide into sugar molecules in the process of photosynthesis, chloroplasts use an enzyme called rubisco | |
| **Generated Questions** | |
| *Baseline* | What does chloroplasts use? |
| *RefNet* | What does chloroplasts use to fix carbon dioxide into sugar molecules? |
| *Reward-RefNet* | What do chloroplasts use to fix carbon dioxide into sugar molecules? |

Table 1: Samples of generated questions from Baseline, RefNet and Reward-RefNet model on the SQuAD dataset. Answers are shown in blue
.

* The first two authors have contributed equally to this work.

[1]https://github.com/PrekshaNema25/RefNet-QG



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3314–3323,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

The above examples indicate that there is clear scope of improving the general quality of the questions. Additionally, the quality can be specifically improved in terms of aspects like: fluency (Example 2) and answerability (Example 1). One way to approach this is by re-visiting the passage and answer with the aim to *refine* the initial draft by generating a better question in the second pass and then improving it with respect to a certain aspect. We can draw a comparison between this process and how humans tend to write a rough initial draft first and then refine it over multiple passes, where the later revisions focus on improving the draft aiming at certain aspects like fluency or completeness. With this motivation, we propose **Refine Network (RefNet)**, which examines the initially generated question and performs a second pass to generate a *revised* question. Furthermore, we propose **Reward-RefNet** which uses explicit reward signals to achieve refinement focused on specific properties of the question such as fluency and answerability.

Our RefNet is a seq2seq based model that comprises of two decoders: *Preliminary* and *Refinement Decoder*. The Refinement Decoder takes the initial draft of the question generated by the Preliminary decoder as an input along with passage and answer, and generates the refined question by attending onto both the passage and the initial draft using a Dual Attention Network. The proposed dual attention aids RefNet to generate the final question by revisiting the appropriate parts of the input passage and initial draft. From Table 1, we can infer that our RefNet model is able to generate better questions in the second pass by fixing the errors in the initial draft. Our Reward-RefNet model uses REINFORCE with a baseline algorithm to explicitly reward the Refinement Decoder for generating a better question as compared to the Preliminary Decoder based on certain desired parameters like fluency and answerability. This leads to more answerable (see Reward-RefNet example for passage 1 in Table 1) and fluent (see Reward-RefNet example for passage 2 in Table 1) questions as compared to vanilla RefNet model.

Our experiments show that the proposed RefNet model outperforms existing state-of-the-art models on the SQuAD dataset by 12.3% and 3.7% (on BLEU) given the relevant sentence and passage respectively. We also achieve state-of-the-art results on HOTPOT-QA and DROP datasets with an improvement of 7.57% and 15.25% respectively over the single-decoder baseline (on BLEU). Our human evaluations further validate these results. We further analyze and explain the impact of including the Refinement Decoder by examining the interaction between both the decoders. Interestingly, we observe that the inclusion of the Refinement Decoder boosts the quality of the questions generated by the initial decoder also. Lastly, our human evaluation of the questions generated by Reward-RefNet corroborate empirical results, *i.e.*, it improves the question *w.r.t.* to fluency and answerability as compared to RefNet questions.

## 2 Refine Networks (RefNet) Model

In this section, we discuss various components of our proposed model as shown in Figure 1. For a given passage $\mathbf{P} = \{w_1^p, \ldots, w_m^p\}$ of length $m$ and answer $\mathbf{A} = \{w_1^a, \ldots, w_n^a\}$ of length $n$, we first obtain *answer-aware* latent representation, $\mathbf{U} = \{\tilde{\mathbf{h}}_1^p, \ldots, \tilde{\mathbf{h}}_m^p\}$, for every word of the passage and an answer representation $\mathbf{h}^a$ (as described in Section 2.1). We then generate an initial draft $\tilde{\mathbf{Q}} = \{\tilde{q}_1, \ldots, \tilde{q}_T\}$ by computing $\tilde{q}_t$ as

$$\tilde{q}_t = \arg\max_{\tilde{q}} \prod_{t=1}^{l} \tilde{p}(\tilde{\mathrm{q}}_t | \tilde{\mathrm{q}}_{t-1}, \ldots, \tilde{\mathrm{q}}_1, \mathbf{U}, \mathbf{h}^a)$$

Here $\tilde{p}(.)$ is a probability distribution modeled using the Preliminary Decoder. We then refine the initial draft $\tilde{\mathbf{Q}}$ using the Refinement Decoder to obtain the refined draft $\mathbf{Q} = \{q_1, \ldots q_T\}$:

$$q_t = \arg\max_{q} \prod_{t=1}^{l} p(\mathrm{q}_t | \mathrm{q}_{t-1}, \ldots, \mathrm{q}_1, \tilde{\mathbf{Q}}, \mathbf{U}, \mathbf{h}^a)$$

We then use explicit rewards to enforce refinement on a desired metric, such as, fluency or answerability through our Reward-RefNet model. In the following sub-sections, we describe the passage encoder, preliminary and refinement decoders and our reward mechanism.

### 2.1 Passage and Answer Encoder

We use a 3 layered encoder consisting of: (i) Embedding, (ii) Contextual and (iii) Passage-Answer Fusion layers as described below. To capture interaction between passage and answer, we ensure that the passage and answer representations are fused together at every layer.

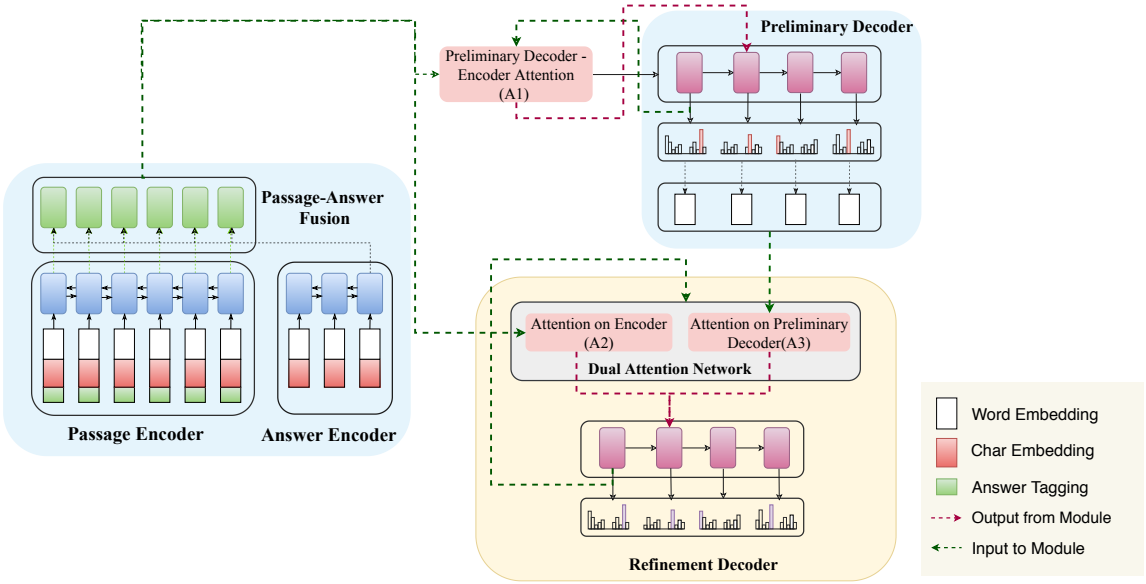


Figure 1: Our RefNet model with Preliminary and Refinement Decoder.

**Embedding Layer:** In this layer, we compute a $d$-dimensional embedding for every word in the passage and the answer. This embedding is obtained by concatenating the word's Glove embedding (Pennington et al., 2014) with its character based embedding as discussed in (Seo et al., 2016). Additionally, for passage words, we also compute a positional embedding based on the relative position of the word *w.r.t.* the answer span as described in (Zhao et al., 2018). For every passage word, this positional embedding is also concatenated to the word and character based embeddings. We discuss the impact of character embeddings and answer tagging in Appendix A. In the subsequent sections, we will refer to embedding of the $i$-th passage word $w_i^p$ as $\mathbf{e}(\mathbf{w_i^p})$ and the $j$-th answer word $w_j^a$ as $\mathbf{e}(\mathbf{w_j^a})$.

**Contextual Layer:** In this layer, we compute a contextualized representation for every word in the passage by passing the word embeddings (as computed above) through a bidirectional-LSTM (Hochreiter and Schmidhuber, 1997):

$$\overrightarrow{\mathbf{h}_t^p} = \text{LSTM}(\mathbf{e}(\mathbf{w_t^p}), \overrightarrow{\mathbf{h}^p}_{t-1}) \;\; \forall t \in [1, m]$$

where $\overrightarrow{\mathbf{h}_t^p}$ is the hidden state of the forward LSTM at time $t$. We then concatenate the forward and backward hidden states as $\mathbf{h}_t^p = [\overrightarrow{\mathbf{h}_t^p}; \overleftarrow{\mathbf{h}_t^p}]$.

The answer could correspond to a span in the passage. Let $j+1$ and $j+n$ be the start and end indices of the answer span in the passage respectively. We can thus refer to $\{\mathbf{h}_{j+1}^p, \ldots, \mathbf{h}_{j+n}^p\}$ as the representation of the answer words in the context of the passage. We then obtain contextualized representations for the $n$ answer words by passing them through LSTM as follows:

$$\overrightarrow{\mathbf{h}_t^a} = \text{LSTM}([\mathbf{e}(\mathbf{w_t^a}), \mathbf{h}_{j+t}^p], \overrightarrow{\mathbf{h}^a}_{t-1}) \;\; \forall t \in [1, n]$$

The final state $\mathbf{h}^a = [\overrightarrow{\mathbf{h}_n^a}; \overleftarrow{\mathbf{h}_n^a}]$ of this Bi-LSTM is used as the answer representation in the subsequent stages. When the answer is not present in the passage, only $\mathbf{e}(\mathbf{w_t^a})$ is passed to the LSTM.

**Passage-Answer Fusion Layer:** In this layer, we refine the representations of the passage words based on the answer representation as follows:

$$\tilde{\mathbf{h}}_i^p = \tanh\left(\mathbf{W}_u\left[\mathbf{h}_i^p; \mathbf{h}^a; \mathbf{h}_i^p \odot \mathbf{h}^a\right]\right) \;\; \forall i \in [1, m]$$

Here $\mathbf{W}_u \in \mathbb{R}^{l \times 3l}$. $l$ is the hidden size of LSTM. This is similar to how (Seo et al., 2016) capture interactions between passage and question for QA. We use $\mathbf{U} = \{\tilde{\mathbf{h}}_1^p, \ldots, \tilde{\mathbf{h}}_m^p\}$ as the fused passage-answer representation which is then used by our decoder(s) to generate the question $\mathbf{Q}$.

## 2.2 Preliminary and Refinement Decoders

As discussed earlier, RefNet has two decoders, *viz.*, Preliminary Decoder and Refinement Decoder, as described below:

**Preliminary Decoder:** This decoder generates an initial draft of the question, one word at a time,

using an LSTM as follows:

$$\tilde{\mathbf{h}}_t^d = \text{LSTM}([\mathbf{e_w}(\tilde{\mathbf{q}}_{\mathbf{t-1}}); \tilde{\mathbf{c}}_{t-1}; \mathbf{h}^a], \tilde{\mathbf{h}}_{t-1}^d)$$
$$\tilde{\mathbf{c}}_t = \sum_{i=1}^{m} \alpha_t^i \tilde{\mathbf{h}}_i^p \quad (1)$$

Here $\tilde{\mathbf{h}}_t^d$ is the hidden state at time $t$, $\mathbf{h}^a$ is the answer representation as computed above, $\tilde{\mathbf{c}}_{t-1}$ is an attention weighted sum of the contextualized passage word representations, $\alpha_t^i$ are parameterized and normalized attention weights (Bahdanau et al., 2014). Let's call this attention network as $\mathbf{A}_1$. $\mathbf{e_w}(\tilde{\mathbf{q}}_\mathbf{t})$ is the embedding of the word $\tilde{q}_t$. We obtain $\tilde{q}_t$ as:

$$\tilde{q}_t = \arg\max_{\tilde{q}} \left(\text{softmax}(\mathbf{W}_o[\mathbf{W}_c[\tilde{\mathbf{h}}_t^d; \tilde{\mathbf{c}}_t])\right), \quad (2)$$

where $\mathbf{W}_c$ is a $\mathbb{R}^{l \times 2l}$ matrix and $\mathbf{W}_o$ is the output matrix which projects the final representation to $\mathbb{R}^V$ where $V$ is the vocabulary size.

**Refinement Decoder**: Once the preliminary decoder generates the entire question, the refinement decoder uses it to generate an updated version of the question using a **Dual Attention Network**. It first computes an attention weighted sum of the embeddings of the words generated by the first decoder as:

$$g_t = \sum_{i=1}^{T} \beta_i^t \mathbf{e_w}(\tilde{\mathbf{q}}_\mathbf{i})$$

where $\beta_i^t$ are parameterized and normalized attention weights computed by attention network $\mathbf{A}_3$. Since the initial draft could be erroneous or incomplete, we obtain additional information from the passage instead of only relying on the output of the first decoder. We do so by computing a context vector $\mathbf{c}_t$ as

$$\mathbf{c}_t = \sum_{i=1}^{m} \gamma_i^t \tilde{\mathbf{h}}_i^p$$

where $\gamma_i^t$ are parameterized and normalized attention weights computed by attention network $\mathbf{A}_3$. The hidden state of the refinement decoder at time $t$ is computed as follows:

$$\mathbf{h}_t^d = \text{LSTM}([\mathbf{e}(\mathbf{q_{t-1}}); \mathbf{c}_{t-1}; \mathbf{g}_{t-1}; \mathbf{h^a}], \mathbf{h}_{t-1}^d)$$

Finally, $q_t$ is predicted using

$$q_t = \arg\max_{q}(\text{softmax}(\mathbf{W}_o[\mathbf{W'}_c[\mathbf{h}_t^d; \mathbf{c}_{t-1}; \mathbf{g}_{t-1}]))$$

where $\mathbf{W}'_c$ is a weight matrix and $\mathbf{W}_o$ is the output matrix which is shared with the Preliminary decoder (Equation 2). Note that RefNet generates two variants of the question : initial draft $\tilde{\mathbf{Q}}$ and final draft $\mathbf{Q}$. We compare these two versions of the generated questions in Section 4.

### 2.3 Reward-RefNet

Next, we address the following question: *Can the refinement decoder be explicitly rewarded for generating a question which is better than that generated by the preliminary decoder on certain desired parameters?* For example, (Nema and Khapra, 2018) define fluency and answerability as desired qualities in the generated question. They evaluate fluency using BLEU score and answerability using a score which captures whether the question contains the required {named entities, important words, function words, question types} (and is thus answerable). We use these fluency and answerability scores proposed by (Nema and Khapra, 2018) as reward signals. We first compute the reward $r(\tilde{\mathbf{Q}})$ and $r(\mathbf{Q})$ for the question generated by the preliminary and refinement decoder respectively. We then use "REINFORCE with a baseline" algorithm (Williams, 1992) to reward Refinement Decoder using the Preliminary Decoder's reward $r(\tilde{\mathbf{Q}})$ as the baseline. More specifically, given the Preliminary Decoder's generated word sequence $\tilde{\mathbf{Q}} = \{\tilde{q}_1, \tilde{q}_2, \ldots, \tilde{q}_T\}$ and the Refinement Decoder's generated word sequence $\mathbf{Q} = \{q_1, q_2, \ldots, q_T\}$ obtained from the distribution $p(\mathrm{q}_t|\mathrm{q}_{t-1}, \ldots, \mathrm{q}_1, \tilde{\mathbf{Q}}, \mathbf{U}, \mathbf{h}^a)$, the training loss is defined as follows

$$L(\mathbf{Q}) = (r(\mathbf{Q}) - r(\tilde{\mathbf{Q}})) \cdot \sum_{t=1}^{T} \log p(q_t|\, q_{t-1}, \ldots, q_1, \tilde{\mathbf{Q}}, \mathbf{U}, \mathbf{h}^a)$$

where $r(\mathbf{Q})$ and $r(\tilde{\mathbf{Q}})$ are the rewards obtained by comparing with the reference question $\mathbf{Q}^*$. As mentioned, this reward $r(.)$ can be the fluency score or answerability score as defined by (Nema and Khapra, 2018).

### 2.4 Copy Module

Along with the above-mentioned three modules, we adopt the pointer-network and coverage mechanism from (See et al., 2017). We use it to (i) handle Out-of-Vocabulary words and (ii) avoid repeating phrases in the generated questions.

| Dataset | Model | n-gram | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | BLEU-1 | BLEU-2 | BLEU-3 | BLEU-4 | ROUGE-L | METEOR | QBLEU4 |
| **SQuAD (Sentence Level)** | (Sun et al., 2018) | 43.02 | 28.14 | 20.51 | 15.64 | - | - | - |
| | (Zhao et al., 2018) | 44.51 | 29.07 | 21.06 | 15.82 | 44.24 | 19.67 | - |
| | (Kim et al., 2019) | - | - | - | 16.17 | - | - | - |
| | EAD | 44.74 | 29.79 | 22.00 | 16.84 | 44.78 | 20.60 | 24.7 |
| | RefNet | **47.27** | **31.88** | **23.65** | **18.16** | **47.14** | **23.40** | **27.4** |
| **SQuAD (Passage Level)** | (Zhao et al., 2018) | 45.07 | 29.58 | 21.60 | 16.38 | 44.48 | 20.25 | - |
| | EAD | 44.61 | 29.37 | 21.50 | 16.36 | 43.95 | 20.11 | 24.2 |
| | RefNet | **46.41** | **30.66** | **22.42** | **16.99** | **45.03** | **21.10** | **26.6** |
| **HOTPOT** | (Zhao et al., 2018)* | 45.29 | 32.06 | 24.43 | 19.29 | 40.40 | 19.29 | 25.7 |
| | EAD | **46.00** | 32.47 | 24.82 | 19.68 | 41.52 | 23.27 | 26.2 |
| | RefNet | 45.45 | **33.13** | **26.05** | **21.17** | **43.12** | **25.81** | **28.7** |
| **DROP Dataset** | (Zhao et al., 2018)* | 39.56 | 29.19 | 22.53 | 18.07 | 45.01 | 19.68 | 31.4 |
| | EAD | 39.21 | 29.10 | 22.65 | 18.42 | 45.07 | 19.56 | 31.8 |
| | RefNet | **42.81** | **32.63** | **25.78** | **21.23** | **47.49** | **22.25** | **33.6** |

Table 2: Comparsion of RefNet model with existing approaches and EAD model. Here * denotes our implementation of the corresponding work.

## 3 Experimental Details

In this section, we discuss (i) the datasets for which we tested our proposed model, (ii) implementation details and (iii) evaluation metrics used to compare our model with the baseline and existing works.

### 3.1 Datasets

**SQuAD** (Rajpurkar et al., 2016): It contains 100K (question, answer) pairs obtained from 536 Wikipedia articles, where the answers are a span in the passage. For SQuAD, AQG has been tried from both sentences and passages. In the former case, only the sentence which contains the answer span is used as input, whereas in the latter case the entire passage is used. We use the same train-validation-test splits as used in (Zhao et al., 2018).

**Hotpot QA** (Yang et al., 2018) : Hotpot-QA is a multi-document and multi-hop QA dataset. Along with the triplet (P, A, Q), the authors also provide supporting facts that potentially lead to the answer. The answers here are either yes/no or answer span in P. We concatenate these supporting facts to form the passage. We use 10% of the training data for validation and use the original dev set as test set.

**DROP** (Dua et al., 2019): The DROP dataset is a reading comprehension benchmark which requires discrete reasoning over passage. It contains 96K questions which require discrete operations such as addition, counting, or sorting to obtain the answer. We use 10% of the original training data for validation and use the original dev set as test set.

### 3.2 Implementation Details

We use 300 dimensional pre-trained Glove word embeddings, which are fixed during training. For character-level embeddings, we initially use a 20 dimensional embedding for the characters which is then projected to 100 dimensions. For answer-tagging, we use embedding size of 3. The hidden size for all the LSTMs is fixed to 512. We use 2-layer, 1-layer and 2-layer stacked BiLSTM for the passage encoder, answer encoder and the decoders (both) respectively. We take the top $30,000$ frequent words as the vocabulary. We use Adam optimizer with a learning rate of $0.0004$ and train our models for 10 epochs using cross entropy loss. For the Reward-RefNet model, we fine-tune the pre-trained model with the loss function mentioned in Section 2.3 for 3 epochs. The best model is chosen based on the BLEU (Papineni et al., 2002) score on the validation split. For all the results we use beam search decoding with a beam size of 5.

### 3.3 Evaluation

We evaluate our models, based on $n$-gram similarity metrics BLEU (Papineni et al., 2002), ROUGE-L (Lin, 2004), and METEOR (Lavie and Denkowski, 2009) using the package released in (Sharma et al., 2017)[2]. We also quantify the answerability of our models using QBLEU-4[3](Nema and Khapra, 2018).

[2] https://github.com/Maluuba/nlg-eval

[3] https://github.com/PrekshaNema25/Answerability-Metric

## 4 Results and Discussions

In this section, we present the results and analysis of our proposed model RefNet. Throughout this section, we refer to our models as follows:

**Encode-Attend-Decode (EAD) model** is our single decoder model containing the encoder and the Preliminary Decoder described earlier. Note that the performance of this model is comparable to our implementation of the model proposed in (Zhao et al., 2018).

**Refine Network (RefNet) model** includes the encoder, the Preliminary Decoder and the Refinement Decoder.

We will (i) compare RefNet's performance with EAD and existing models across all the mentioned datasets (ii) report human evaluations to compare RefNet and EAD (iii) analyze Refinement and Preliminary Decoders iv) present the performance of Reward RefNet with two different reward signal (fluency and answerability).

### 4.1 RefNet's performance across datasets

In Table 2, we compare the performance of RefNet with existing single decoder architectures across different datasets. On BLEU-4 metric, RefNet beats the existing state-of-the-art model by 12.30%, 9.74%, 17.48%, and 3.71% respectively on SQuAD (sentence), HOTPOT-QA, DROP and SQuAD (passage) dataset. Also it outperforms EAD by 7.83%, 7.57%, 15.25% and 3.85% respectively on SQuAD (sentence), HOTPOT-QA, DROP and SQuAD (passage). In general, RefNet is consistently better than existing models across all $n$-gram scores (BLEU, ROUGE-L and METEOR). Along with $n$-gram scores, we also observe improvements on Q-BLEU4 as well, which as described earlier, gives a measure of both answerability and fluency.

### 4.2 Human Evaluations

We conducted human evaluations to analyze the quality of the questions produced by EAD and RefNet. We randomly sampled 500 questions generated from the SQuAD (sentence level) dataset and asked the annotators to compare the quality of the generated questions. The annotators were shown a pair of questions, one generated by EAD and one by RefNet from the same sentence, and were asked to decide which one was better in terms of Fluency, Completeness, and Answerability. They were allowed to skip the question pairs where they could not make a clear choice. Three annotators rated each question and the final label was calculated based on majority voting. We observed that the RefNet model outperforms the EAD model across all three metrics. Over 68.6%, 66.7% and 64.2% of the generated questions from RefNet were respectively more fluent, complete and answerable when compared to the EAD model. However, there are some cases where EAD does better than RefNet. For example, in Table 3, we show that while trying to generate a more elaborate question, RefNet introduces an additional phrase *"in the united"* which is not required. Due to such instances, annotators preferred the EAD model in around 30% of the instances.

| |
|---|
| **Passage:** Before the freeze ended in 1952, there were only 108 existing television stations in the United States; a few major cities (such as Boston) had only two television stations, ... |
| **Questions**<br>*EAD*: how many television stations existed in boston ?<br>*RefNet*: how many television stations did boston have in the united ? |

Table 3: An example where EAD model was better than RefNet. The ground truth answers are shown in blue.

| Model | **Decoder** | **BLEU-4** | **QBLEU-4** |
|---|---|---|---|
| without $\mathbf{A_3}$ | *RefNet* | 17.16 | 25.80 |
| | *Initial Draft* | 17.59 | 26.00 |
| with $\mathbf{A_3}$ | *RefNet* | 18.37 | 27.40 |
| | *Initial Draft* | 17.89 | 26.00 |

Table 4: Comparison between Preliminary Decoder and Refinement Decoder in RefNet Model for SQuAD Sentence Level QG.

### 4.3 Analysis of Refinement Decoder and Preliminary Decoder

The two decoders impact each other through two paths: (i) indirect path, where they share the encoder and the output projection to the vocabulary $V$, (ii) direct path, via the dual attention network, where the initial draft of the question is attended by the Refinement Decoder. When RefNet has only indirect path, we can infer from row 1 of Table 4 that the performance of Preliminary Decoder improves when compared to the EAD model (16.84 v/s 17.59 BLEU). This suggests that generating two variants of the question improves the performance of the first decoder pass as well. This is perhaps due to the additional feedback that the

| | |
|---|---|
| **Sample:** | **Sentence:** For instance , the language { xx — x is any binary string } can be solved in linear time on a multi-tape Turing machine , but necessarily requires quadratic time in the model of single-tape Turing machines .<br>**Reference Question:** A multi-tape Turing machine requires what type of time for a solution ? |
| **with $A_3$** | **Refinement Decoder**: in what time can the language be solved on a multi-tape turing machine ?<br>**Preliminary Decoder:** in what time can the language be solved ? |
| **without $A_3$** | **Refinement Decoder:** in what time can the language { xx — x x x x is any binary string ?<br>**Preliminary Decoder:** in what time can the language — x x x x is solved ? |

Table 5: Generated samples by Preliminary Decoder and Refinement Decoder in RefNet model.

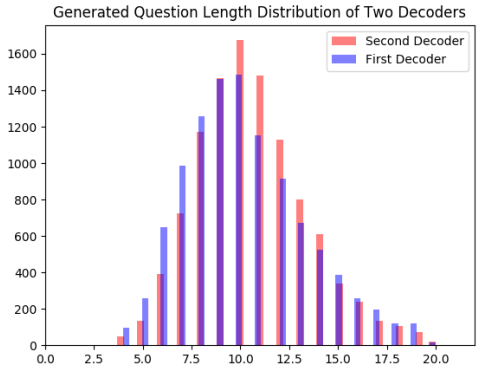


Figure 2: Generated Question Length Distribution for Preliminary Decoder (First Decoder) and Refinement Decoder (Second Decoder).

| Model | BLEU Reward Signal | | Answerability Reward Signal | |
|---|---|---|---|---|
| | BLEU4 | %preference | Ans. | %preference |
| RefNet | 18.37 | 32.9% | 36.9 | 30% |
| Reward-RefNet | 18.52 | 67.1% | 37.5 | 70% |

Table 6: Impact of Reward-RefNet on fluency and answerability. %preference denotes the percentage of times annotators prefer the generated output from the model for fluency in case of BLEU Reward signal and answerability in case of Answerability Reward signal.

| |
|---|
| **Passage:** Cost engineers and estimators apply expertise to relate the work and materials involved to a proper valuation |
| **Questions**<br>*Generated*: Who apply expertise to relate the work and materials involved to a proper valuation ?<br>*True*: Who applies expertise to relate the work and materials involved to a proper valuation ? |

Table 7: An example of question with significant overlap with the passage. The answer is shown in blue.

shared encoder and output layer get from the Refinement Decoder. When we add the direct path (attention network) between the two decoders, the performance of the Refinement Decoder improves as compared to the Preliminary Decoder as shown in rows 3 and 4 of the Table 4

**Comparison on Answerability:** We also evaluate both the initial and refined draft using QBLEU4. As discussed earlier, Q-Metric measures Answerability using four components, *viz.*, Named Entities, Important Words, Function Words, and Question Type. We observe that the increase in Q-Metric for refined questions is because the RefNet model can correct/add the relevant Named Entities in the question. In particular, we observe that the Named Entity component score in Q-Metric increases from 32.42 for the first draft to 37.81 for the refined draft.

**Qualitative Analysis:** Figure 2 shows that the RefNet model indeed generates more elaborate questions when compared to the Preliminary Decoder. As shown in Table 5, the quality of the refined question is better than the initial draft of the questions. Here RefNet adds the phrase "multi-tape Turing Machine," (row 2) which removes any ambiguity in the question.

### 4.4 Analysis of Reward-RefNet

In this section, we analyze the impact of employing different reward signals in Reward-RefNet. As discussed earlier in section 2.3, we use fluency and answerability scores as reward signals. As shown in Table 6, when BLEU-4 (fluency) is used as a reward signal, there is improvement in BLEU-4 scores of Reward-RefNet as compared to RefNet model. We validated these results through human evaluations across 200 samples. Annotators prefer the Reward-RefNet model in 67% of the cases for fluency. Similarly when we use Answerability score as a reward signal, answerability improves for the model and annotators prefer the Reward-RefNet in 70% of the cases for answerability. The performance of Reward-RefNet on fluency and answerability is similar for other datasets (see Appendix B).

**Case Study: Originality of the Questions**

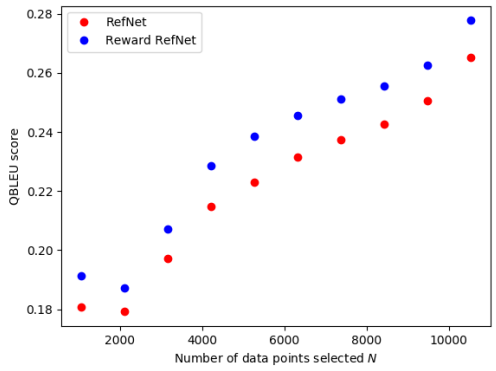


Figure 3: Originality Analysis: Plot of Q-BLEU score vs $N$ - the number points selected.

We observe that current state-of-the-art models perform very well in terms of BLEU/QBLEU scores when the actual question has significant overlap with the passage. For example, consider a passage from the SQuAD dataset in Table 7, where except the question word *who*, the model sequentially copies everything from the passage and achieves a QBLEU score of 92.4. However, the model performs poorly in situations where the true question is novel and does not contain a large sequence of words from the passage itself. In order to quantify this, we first sort the true questions based on its BLEU-2 overlap with the passage in ascending order. We then select the first $N$ true questions and compute the QBLEU score with the generated questions. The results are shown in red in Figure 3. Towards the left, where there are true questions with low overlap with the passage, the performance is poor, but it gradually improves as the overlap increases.

The task of generating questions with high *originality* (where the model phrases the question in its own words) is a challenging aspect of AQG since it requires complete understating of the semantics and syntax of the language. In order to improve questions generated on *originality*, we explicitly reward our model for having low $n$-gram score with the passage as compared to the initial draft. As a result we observe that with Reward-RefNet(Originality), there is an improvement in the performance where the overlap with the passage was less (as shown in blue in Figure 3).As shown in Table 8, although both questions are answerable given the passage, the question generated from Reward-RefNet(Originality) is better.

| |
|---|
| **Passage:** McLetchie was elected on the Lothian regional list and the Conservatives suffered a net loss of five seats , with leader Annabel Goldie claiming that their support had held firm, nevertheless, she too announced she would step down as leader of the party. |
| **Questions** |
| *True*: Who announced she would step down as leader of the Conservatives ? |
| *RefNet*: who claiming that their support had held firm ? |
| *Reward-RefNet*: who was the leader of the conservatives? |

Table 8: An example where Reward-RefNet(Originality) is better than RefNet.

## 5 Related Work

Early works on Question Generation were essentially rule based systems (Heilman and Smith, 2010; Mostow and Chen, 2009; Lindberg et al., 2013; Labutov et al., 2015). Current models for AQG are based on the encode-attend-decode paradigm and they either generate questions from the passage alone (Du and Cardie, 2017; Du et al., 2017; Yao et al., 2018) or from the passage and a given answer (in which case the generated question must result in the given answer). Over the past couple of years, several variants of the encode-attend-decode model have been proposed. For example, (Zhou et al., 2018) proposed a sequential copying mechanism to explicitly select a sub-span from the passage. Similarly, (Zhao et al., 2018) mainly focuses on efficiently incorporating paragraph level content by using Gated Self Attention and Maxout pointer networks. Some works (Yuan et al., 2017) even use Question Answering as a metric to evaluate the generated questions. There has also been some work on generating questions from images (Jain et al., 2017; Li et al., 2017) and from knowledge bases (Serban et al., 2016; Reddy et al., 2017). The idea of multi pass decoding which is central to our work has been used by (Xia et al., 2017) for machine translation and text summarization albeit with a different objective. Some works have also augmented seq2seq models (Rennie et al., 2017; Paulus et al., 2018; Song et al., 2017) with external reward signals using REINFORCE with baseline algorithm (Williams, 1992). The typical rewards used in these works are BLEU and ROUGE scores. Our REINFORCE loss is different from the previous ones as it uses the first decoder's reward as the baseline instead of reward of the greedy policy.

## 6 Conclusion and Future Work

In this work, we proposed **Refine Networks (RefNet)** for Question Generation to focus on refining and improving the initial version of the generated question. Our proposed RefNet model consisting of a Preliminary Decoder and a Refinement Decoder with Dual Attention Network outperforms the existing state-of-the-art models on the SQuAD, HOTPOT-QA and DROP datasets. Along with automated evaluations, we also conducted human evaluations to validate our findings. We further showed that using Reward-RefNet improves the initial draft on specific aspects like fluency, answerability and originality. As a future work, we would like to extend RefNet to have the ability to decide whether a refinement is needed on the generated initial draft.

## Acknowledgements

We thank Amazon Web Services for providing free GPU compute and Google for supporting Preksha Nema's contribution in this work through Google Ph.D. Fellowship programme. We would like to acknowledge Department of Computer Science and Engineering, IIT Madras and Robert Bosch Center for Data Sciences and Artificial Intelligence, IIT Madras (RBC-DSAI) for providing us sufficient resources. We would also like to thank Patanjali SLPSK, Sahana Ramnath, Rahul Ramesh, Anirban Laha, Nikita Moghe and the anonymous reviewers for their valuable and constructive suggestions.

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Xinya Du and Claire Cardie. 2017. Identifying where to focus in reading comprehension for neural question generation. In *EMNLP*, pages 2067–2073. Association for Computational Linguistics.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *ACL (1)*, pages 1342–1352. Association for Computational Linguistics.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proc. of NAACL*.

Michael Heilman and Noah A. Smith. 2010. Good question! statistical ranking for question generation. In *HLT-NAACL*, pages 609–617. The Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Unnat Jain, Ziyu Zhang, and Alexander G. Schwing. 2017. Creativity: Generating diverse questions using variational autoencoders. In *CVPR*, pages 5415–5424. IEEE Computer Society.

Yanghoon Kim, Hwanhee Lee, Joongbo Shin, and Kyomin Jung. 2019. Improving neural question generation using answer separation. *CoRR*, abs/1809.02393.

Igor Labutov, Sumit Basu, and Lucy Vanderwende. 2015. Deep questions without deep understanding. In *ACL (1)*, pages 889–898. The Association for Computer Linguistics.

Alon Lavie and Michael J. Denkowski. 2009. The meteor metric for automatic evaluation of machine translation. *Machine Translation*, 23(2-3):105–115.

Yikang Li, Nan Duan, Bolei Zhou, Xiao Chu, Wanli Ouyang, and Xiaogang Wang. 2017. Visual question generation as dual task of visual question answering. *CoRR*, abs/1709.07192.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Proc. ACL workshop on Text Summarization Branches Out*, page 10.

David Lindberg, Fred Popowich, John C. Nesbit, and Philip H. Winne. 2013. Generating natural language questions to support learning on-line. In *ENLG*, pages 105–114. The Association for Computer Linguistics.

Jack Mostow and Wei Chen. 2009. Generating instruction automatically for the reading strategy of self-questioning. In *AIED*, volume 200 of *Frontiers in Artificial Intelligence and Applications*, pages 465–472. IOS Press.

Preksha Nema and Mitesh M. Khapra. 2018. Towards a better metric for evaluating question generation systems. *CoRR*, abs/1808.10192.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318. ACL.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. *CoRR*, abs/1705.04304.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2383–2392.

Sathish Reddy, Dinesh Raghu, Mitesh M. Khapra, and Sachindra Josh. 2017. Generating natural language question-answer pairs from a knowledge graph using a RNN based question generation model. In *EACL (1)*, pages 376–385. Association for Computational Linguistics.

Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1179–1195.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *ACL*.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603.

Iulian Vlad Serban, Alberto García-Durán, Çaglar Gülçehre, Sungjin Ahn, Sarath Chandar, Aaron C. Courville, and Yoshua Bengio. 2016. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. In *ACL (1)*. The Association for Computer Linguistics.

Shikhar Sharma, Layla El Asri, Hannes Schulz, and Jeremie Zumer. 2017. Relevance of unsupervised metrics in task-oriented dialogue for evaluating natural language generation. *CoRR*, abs/1706.09799.

Heung-yeung Shum, Xiao-dong He, and Di Li. 2018. From eliza to xiaoice: challenges and opportunities with social chatbots. *Frontiers of Information Technology & Electronic Engineering*, 19(1):10–26.

Linfeng Song, Zhiguo Wang, and Wael Hamza. 2017. A unified query-based generative model for question generation and question answering. *CoRR*, abs/1709.01058.

Xingwu Sun, Jing Liu, Yajuan Lyu, Wei He, Yanjun Ma, and Shi Wang. 2018. Answer-focused and position-aware neural question generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3930–3939, Brussels, Belgium. Association for Computational Linguistics.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3-4):229–256.

Yingce Xia, Fei Tian, Lijun Wu, Jianxin Lin, Tao Qin, Nenghai Yu, and Tie-Yan Liu. 2017. Deliberation networks: Sequence generation beyond one-pass decoding. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 1784–1794. Curran Associates, Inc.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Kaichun Yao, Libo Zhang, Tiejian Luo, Lili Tao, and Yanjun Wu. 2018. Teaching machines to ask questions. In *IJCAI*.

Xingdi Yuan, Tong Wang, Çaglar Gülçehre, Alessandro Sordoni, Philip Bachman, Saizheng Zhang, Sandeep Subramanian, and Adam Trischler. 2017. Machine comprehension by text-to-text neural question generation. In *Rep4NLP@ACL*, pages 15–25. Association for Computational Linguistics.

Yao Zhao, Xiaochuan Ni, Yuanyuan Ding, and Qifa Ke. 2018. Paragraph-level neural question generation with maxout pointer and gated self-attention networks. In *EMNLP*.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2018. Sequential copying networks. In *AAAI*.

# Earlier Isn't Always Better: Sub-aspect Analysis on Corpus and System Biases in Summarization

**Taehee Jung**[*♡] **Dongyeop Kang**[*♣] **Lucas Mentch**[♡] **Eduard Hovy**[♣]
[♡]Department of Statistics, University of Pittsburgh, Pittsburgh, PA, USA
[♣]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA
{taj41,lkm31}@pitt.edu {dongyeok,hovy}@cs.cmu.edu

## Abstract

Despite the recent developments on neural summarization systems, the underlying logic behind the improvements from the systems and its corpus-dependency remains largely unexplored. Position of sentences in the original text, for example, is a well known bias for news summarization. Following in the spirit of the claim that summarization is a combination of sub-functions, we define three sub-aspects of summarization: `position`, `importance`, and `diversity` and conduct an extensive analysis of the biases of each sub-aspect with respect to the domain of nine different summarization corpora (e.g., news, academic papers, meeting minutes, movie script, books, posts). We find that while `position` exhibits substantial bias in news articles, this is not the case, for example, with academic papers and meeting minutes. Furthermore, our empirical study shows that different types of summarization systems (e.g., neural-based) are composed of different degrees of the sub-aspects. Our study provides useful lessons regarding consideration of underlying sub-aspects when collecting a new summarization dataset or developing a new system.

## 1 Introduction

Despite numerous recent developments in neural summarization systems (Narayan et al., 2018b; Nallapati et al., 2016; See et al., 2017; Kedzie et al., 2018; Gehrmann et al., 2018; Paulus et al., 2017) the underlying rationales behind the improvements and their dependence on the training corpus remain largely unexplored. Edmundson (1969) put forth the position hypothesis: important sentences appear in preferred positions in the document. Lin and Hovy (1997) provide a method to empirically identify such positions. Later, Hong and Nenkova (2014) showed an intentional lead bias in news writing, suggesting that sentences appearing early in news articles are more important for summarization tasks. More generally, it is well known that recent state-of-the-art models (Nallapati et al., 2016; See et al., 2017) are often marginally better than the first-k baseline on single-document news summarization.

In order to address the position bias of news articles, Narayan et al. (2018a) collected a new dataset called XSum to create single sentence summaries that include material from multiple positions in the source document. Kedzie et al. (2018) showed that the position bias in news articles is not the same across other domains such as meeting minutes (Carletta et al., 2005).

In addition to `position`, Lin and Bilmes (2012) defined other sub-aspect functions of summarization including `coverage`, `diversity`, and `information`. Lin and Bilmes (2011) claim that many existing summarization systems are instances of mixtures of such sub-aspect functions; for example, maximum marginal relevance (MMR) (Carbonell and Goldstein, 1998) can be seen as an combination of diversity and importance functions.

Following the sub-aspect theory, we explore three important aspects of summarization (§3): POSITION for choosing sentences by their position, IMPORTANCE for choosing relevant contents, and DIVERSITY for ensuring minimal redundancy between summary sentences.

We then conduct an in-depth analysis of these aspects over nine different domains of summarization corpora (§5) including news articles, meeting minutes, books, movie scripts, academic papers, and personal posts. For each corpus, we investigate which aspects are most important and develop a notion of **corpus bias** (§6). We provide an empirical result showing how current summarization systems are compounded of which sub-aspect

[*] Equal contribution, name order decided by coin flip.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3324–3335, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

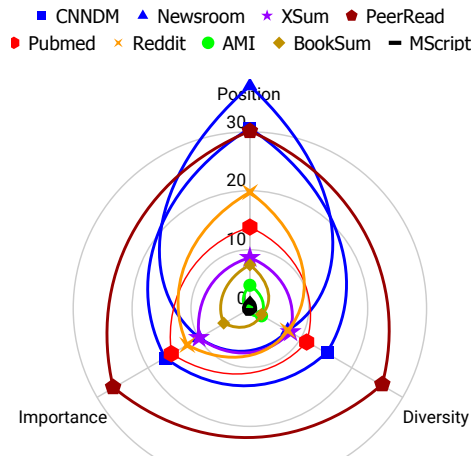


(a) Corpus bias

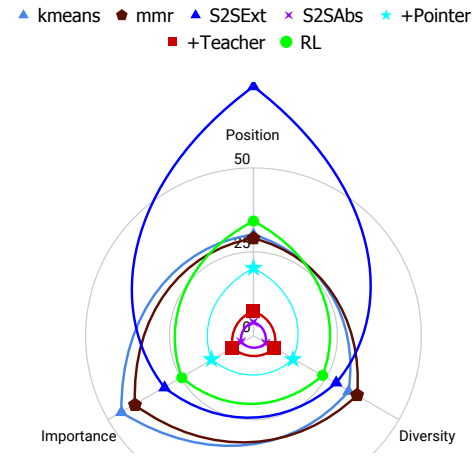


(b) System bias

Figure 1: Corpus and system biases with the three sub-aspects, showing what portion of aspect is used for each corpus and each system. The portion is measured by calculating ROUGE score between (a) summaries obtained from each aspect and target summaries or (b) summaries obtained from each aspect and each system.

factors called **system bias** (§7). At last, we summarize our actionable messages for future summarization researches (§8). We summarize some notable findings as follows:

- Summarization of personal post and news articles except for XSum (Narayan et al., 2018a) are biased to the position aspect, while academic papers are well balanced among the three aspects (see Figure 1 (a)). Summarizing long documents (e.g. books and movie scripts) and conversations (e.g. meeting minutes) are extremely difficult tasks that require multiples aspects together.
- Biases do exist in current summarization systems (Figure 1 (b)). Simple ensembling of multiple aspects of systems show comparable performance with simple single-aspect systems.
- Reference summaries in current corpora include less than 15% of new words that do not appear in the source document, except for abstract text of academic papers.
- Semantic volume (Yogatama et al., 2015) overlap between the reference and model summaries is not correlated with the hard evaluation metrics such as ROUGE (Lin, 2004).

## 2 Related Work

We provide here a brief review of prior work on summarization biases. Lin and Hovy (1997) studied the position hypothesis, especially in the news article writing (Hong and Nenkova, 2014; Narayan et al., 2018a) but not in other domains such as conversations (Kedzie et al., 2018). Narayan et al. (2018a) collected a new corpus to address the bias by compressing multiple contents of source document in the single target summary. In the bias analysis of systems, Lin and Bilmes (2012, 2011) studied the sub-aspect hypothesis of summarization systems. Our study extends the hypothesis to various corpora as well as systems. With a specific focus on `importance` aspect, a recent work (Peyrard, 2019a) divided it into three subcategories; redundancy, relevance, and informativeness, and provided quantities of each to measure. Compared to this, ours provide broader scale of sub-aspect analysis across various corpora and systems.

We analyze the sub-aspects on different domains of summarization corpora: news articles (Nallapati et al., 2016; Grusky et al., 2018; Narayan et al., 2018a), academic papers or journals (Kang et al., 2018; Kedzie et al., 2018), movie scripts (Gorinski and Lapata, 2015), books (Mihalcea and Ceylan, 2007), personal posts (Ouyang et al., 2017), and meeting minutes (Carletta et al., 2005) as described further in §5.

Beyond the corpora themselves, a variety of summarization systems have been developed: Mihalcea and Tarau (2004); Erkan and Radev (2004) used graph-based keyword ranking algorithms. Lin and Bilmes (2010); Carbonell and Goldstein (1998) found summary sentences which are highly relevant but less redundant. Yogatama et al. (2015) used semantic volumes of bigram features for extractive summarization. Internal structures of documents have been used in summarization: syntactic parse trees (Woodsend and Lapata, 2011; Cohn

and Lapata, 2008), topics (Zajic et al., 2004; Lin and Hovy, 2000), semantic word graphs (Mehdad et al., 2014; Gerani et al., 2014; Ganesan et al., 2010; Filippova, 2010; Boudin and Morin, 2013), and abstract meaning representation (Liu et al., 2015). Concept-based Integer-Linear Programming (ILP) solver (McDonald, 2007) is used for optimizing the summarization problem (Gillick and Favre, 2009; Banerjee et al., 2015; Boudin et al., 2015; Berg-Kirkpatrick et al., 2011). Durrett et al. (2016) optimized the problem with grammatical and anarphorcity constraints.

With a large scale of corpora for training, neural network based systems have recently been developed. In abstractive systems, Rush et al. (2015) proposed a local attention-based sequence-to-sequence model. On top of the seq2seq framework, many other variants have been studied using convolutional networks (Cheng and Lapata, 2016; Allamanis et al., 2016), pointer networks (See et al., 2017), scheduled sampling (Bengio et al., 2015), and reinforcement learning (Paulus et al., 2017). In extractive systems, different types of encoders (Cheng and Lapata, 2016; Nallapati et al., 2017; Kedzie et al., 2018) and optimization techniques (Narayan et al., 2018b) have been developed. Our goal is to explore which types of systems learns which sub-aspect of summarization.

## 3 Sub-aspects of Summarization

We focus on three crucial aspects : Position, Diversity, and Importance. For each aspect, we use different extractive algorithms to **capture how much of the aspect is used in the oracle extractive summaries**[1]. For each algorithm, the goal is to select $k$ extractive summary sentences (equal to the number of sentences in the target summaries for each sample) out of $N$ sentences appearing in the original source. The chosen sentences or their indices will be used to calculate the various evaluation metrics described in §4

For some algorithms below, we use vector representation of sentences. We parse a document $x$ into a sequence of sentences $x = x_1..x_N$ where each sentence consists of a sequence of words $x_i = w_1..w_s$. Each sentence is then encoded:

$$\mathrm{E}(x_i) = \texttt{BERT}(w_{i,1}..w_{i,s}) \quad (1)$$

where BERT (Devlin et al., 2018) is a pre-trained bidirectional encoder from transformers (Vaswani et al., 2017)[2]. We use the last layer from BERT as a representation of each token, and then average them to get final representation of a sentence. All tokens are lower cased.

### 3.1 Position

Position of sentences in the source has been suggested as a good indicator for choosing summary sentences, especially in news articles (Lin and Hovy, 1997; Hong and Nenkova, 2014; See et al., 2017). We compare three position-based algorithms: **First**, **Last**, and **Middle**, by simply choosing $k$ number of sentences in the source document from these positions.

### 3.2 Diversity

Yogatama et al. (2015) assume that extractive summary sentences which maximize the semantic volume in a distributed semantic space are the most diverse but least redundant sentences. Motivated by this notion, our goal is to find a set of $k$ sentences that maximizes the volume size of them in a continuous embedding space like the BERT representations in Eq 1. Our objective is to find the optimal search function $\mathcal{S}$ that maximizes the volume size $\mathcal{V}$ of searched sentences: $\arg\max_{1..k} \mathcal{V}(\mathcal{S}_{1..c}(\mathrm{E}(x_1), \ldots, \mathrm{E}(x_N)))$.

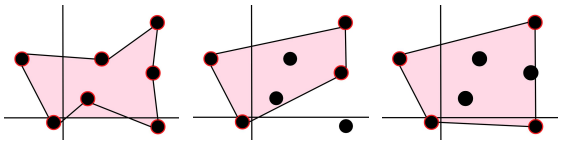

(a) Default (b) Heuristic (c) ConvexFall

Figure 2: Volume maximization functions. Black dots are sentences in source document, and red dots are chosen summary sentences. The red-shaded polygons are volume space of the summary sentences.

If $k$=$N$, we use every sentence from the source document. (Figure 2 (a)). However, its volume space does not guarantee to maximize the volume size because of the non-convex polygonality. In order to find a convex maximum volume, we consider two different algorithms described below.

**Heuristic.** Yogatama et al. (2015) heuristically choose a set of summary sentences using a greedy algorithm: It first chooses a sentence which has the farthest vector representation from the centroid of whole source sentences, and then repeatedly finds sentences whose representation is farthest from

[1] See §4 for our oracle set construction.

[2] The other encoders such as averaging word embeddings (Pennington et al., 2014) show comparable performance.

the centroid of vector representations of the chosen sentences. Unlike the original algorithm in (Yogatama et al., 2015) restricting the number of words, we constrain the total number of selected sentences to $k$. This heuristic algorithm can fail to find the maximum volume depending on its starting point and/or the farther distance between two points detected (Figure 2 (b)).

**ConvexFall.** Here we first find the convex-hull[3] using Quickhull (Barber et al., 1996), implemented by Qhull library[4]. It guarantees the maximum volume size of selected points with minimum number of points (Figure 2 (c)). However, it does not reduce a redundancy between the points over the convex-hull, and usually choose larger number of sentences than $k$. Marcu (1999) shows an interesting study regarding an importance of sentences: given a document, if one deletes the least central sentence from the source text, then at some point the similarity with the reference text rapidly drops at sudden called the *waterfall* phenomena. Motivated by his study, we similarly prune redundant sentences from the set chosen by convex-hull search. For each turn, the sentence with the lowest volume reduction ratio is pruned until the number of remaining sentences is equivalent to $k$.

### 3.3 Importance

We assume that contents that repeatedly occur in one document contain *important* information. We find sentences that are nearest to the neighbour sentences using two distance measures: **N-Nearest** calculates an averaged Pearson correlation between one and the rest for all source sentence vector representations. $k$ sentences having the highest averaged correlation are selected as final extractive summaries. On the other hand, **K-Nearest** chooses the $K$ nearest sentences per each sentence, and then averages distances between each nearest sentence and the selected one. The one has the lowest averaged distance is chosen. This calculation is repeated $k$ times and the selected sentences are removed from the remaining pool.

## 4 Metrics

In order to determine the aspects most crucial to the summarization task, we use three evaluation metrics:

**ROUGE** is Recall-Oriented Understudy for Gisting Evaluation (Lin and Hovy, 2000) for evaluating summarization systems. We use ROUGE-1 (R1), ROUGE-2 (R2), and ROUGE-L (RL) F-measure scores which corresponds to uni-gram, bigrams and longest common subsequences, respectively, and their averaged score (R).

**Volume Overlap (VO) ratio.** Hard metrics like ROUGE often ignore semantic similarities between sentences. Based on the volume assumption in (Yogatama et al., 2015), we measure overlap ratio of two semantic volumes calculated by the model and target summaries. We obtain a set of vector representations of the reference summary sentences $\hat{Y}$ and the model summary sentences $Y$ predicted by any algorithm *algo* in §3 for the $i$-th document:

$$\hat{Y}_i = (\hat{y}_{i,1} .. \hat{y}_{i,k}), \quad Y_i^{algo} = (y_{i,1}^{algo} .. y_{i,k}^{algo}) \tag{2}$$

Each volume $\mathtt{V}$ is then calculated using the convex-hull algorithm and their overlap ($\sqcap$) is calculated using a shapely package[5][6]. The final VO is then:

$$\mathrm{VO}_{algo} = \sum_{i=1}^{N} \frac{\mathtt{V}(E(Y_i^{algo})) \sqcap \mathtt{V}(E(\hat{Y}_i))}{\mathtt{V}(E(\hat{Y}_i))} \tag{3}$$

where $N$ is the total number of input documents, $E$ is the BERT sentence encoder in Eq 1, and $E(\hat{Y}_i)$ and $E(Y_i^{algo})$ are a set of vector representations of the reference and model summary sentences, respectively. The volume overlap indicates how two summaries are semantically overlapped in a continuous embedding space.

**Sentence Overlap (SO) ratio.** Even though ROUGE provides a recall-oriented lexical overlap, we don't know the upper-bound on performance (called `oracle`) of the extractive summarization. We extract the oracle extractive sentences (i.e. a set of input sentences) which maximizes ROUGE-L F-measure score with the reference summary. We then measure sentence overlap (SO) which determines how many extractive sentences from our algorithms are in the oracle summary. The SO is:

$$\mathrm{SO}_{algo} = \sum_{i=1}^{n} \frac{\mathtt{C}(Y_i^{algo} \cap \hat{Y}_i)}{\mathtt{C}(\hat{Y}_i)} \tag{4}$$

where $\mathtt{C}$ is a function for counting the number of elements in a set. The sentence overlap indicates

[3] Definition: a set of points is defined as the smallest convex set that includes the points.

[4] http://www.qhull.org/

[5] https://pypi.org/project/Shapely/

[6] Due to the lack of overlap calculation between two polygons of high dimensions, we reduce it to 2D PCA space.

how well the algorithm finds the oracle summaries for extractive summarization.

## 5 Summarization Corpora

We use various domains of summarization datasets to conduct the bias analysis across corpora and systems. Each dataset has source documents and corresponding abstractive target summaries. We provide a list of datasets used along with a brief description and our pre-processing scheme:

- `CNNDM` (Nallapati et al., 2016): contains 300K number of online news articles. It has multiple sentences (4.0 on average) as a summary.
- `Newsroom` (Grusky et al., 2018): contains 1.3M news articles and written summaries by authors and editors from 1998 to 2017. It has both extractive and abstractive summaries.
- `XSum` (Narayan et al., 2018a): has news articles and their single but abstractive sentence summaries mostly written by the original author.
- `PeerRead` (Kang et al., 2018): consists of scientific paper drafts in top-tier computer science venues as well as `arxiv.org`. We use full text of introduction section as source document and of abstract section as target summaries.
- `PubMed` (Kedzie et al., 2018): is 25,000 medical journal papers from the PubMed Open Access Subset.[7] Unlike `PeerRead`, full paper except for abstract is used as source documents.
- `MScript` (Gorinski and Lapata, 2015): is a collection of movie scripts from ScriptBase corpus and their corresponding user summaries of the movies.
- `BookSum` (Mihalcea and Ceylan, 2007): is a dataset of classic books paired to summaries from Grade Saver[8] and Cliffs Notes[9]. Due to a large number of sentences, we only choose the first 1K sentences for source document and the first 50 sentences for target summaries.
- `Reddit` (Ouyang et al., 2017): is a collection of personal posts from `reddit.com`. We use a single abstractive summary per post. The same data split from Kedzie et al. (2018) is used.
- `AMI` (Carletta et al., 2005): is documented meeting minutes from a hundred hours of recordings and their abstractive summaries.

[7] https://www.ncbi.nlm.nih.gov/pmc/tools/openftlist/

[8] http://www.gradesaver.com

[9] http://www.cliffsnotes.com/

Table 1 summarizes the characteristics of each dataset. We note that the Gigaword (Graff et al., 2003), New York Times[10], and Document Understanding Conference (DUC)[11] are also popular datasets commonly used in summarization analyses, though here we exclude them as they represent only additional collections of news articles, showing similar tendencies to the other news datasets such as `CNNDM`.

## 6 Analysis on Corpus Bias

We conduct different analyses of how each corpus is biased with respect to the sub-aspects. We highlight some key findings for each sub-section.

### 6.1 Multi-aspect analysis

Table 2 shows a comparison of the three aspects for each corpus where we include random selection and the oracle set. For each dataset metrics are calculated on a test set except for `BookSum` and `AMI` where we use train+test due to the smaller sample size.

**Earlier isn't always better.** Sentences selected early in the source show high `ROUGE` and `SO` on `CNNDM`, `Newsroom`, `Reddit`, and `BookSum`, but not in other domains such as medial journals and meeting minutes, and the condensed news summaries (`XSum`). For summarization of movie scripts in particular, the last sentences seem to provide more important summaries.

**`XSum` requires much IMPORTANCE than other corpora.** Interestingly, the most powerful algorithm for `XSum` is `N-Nearest`. This shows that summaries in `XSum` are indeed collected by abstracting multiple important contents into single sentence, avoiding the position bias.

**First, ConvexFall, and N-Nearest tend to work better than the other algorithms for each aspect.** `First` is better than `Last` or `Middle` in new articles except for `XSum` and personal posts, while not in academic papers (i.e., `PeerRead`, `PubMed`) and meeting minutes. `ConvexFall` finds the set of sentences that maximize the semantic volume overlap with the target sentences better than the heuristic one.

**`ROUGE` and `SO` show similar behavior, while `VO` does not.** In most evaluations, ROUGE scores are linear to SO ratios as expected. However, VO has high variance across algorithms and aspects.

[10] https://catalog.ldc.upenn.edu/LDC2008T19

[11] http://duc.nist.gov

| | CNNDM | Newsroom | Xsum | PeerRead | PubMed | Reddit | AMI | BookSum | MScript |
|---|---|---|---|---|---|---|---|---|---|
| Source | News | News | News | Papers | Papers | Post | Minutes | Books | Script |
| Multi-sents. | ✓ | ✓ | X | ✓ | ✓ | X | ✓ | ✓ | ✓ |
| Data size | 287K/11K | 992K/109K | 203K/11K | 10K/550 | 21K/2.5K | 404/48 | 98/20 | - /53 | - /1K |
| Avg src sents. | 40/34 | 24/24 | 33/33 | 45/45 | 97/97 | 19/15 | 767/761 | - /6.7K | - /3K |
| Avg tgt sents. | 4/4 | 1.4/1.4 | 1/1 | 6/6 | 10/10 | 1/1 | 17/17 | - /336 | - /5 |
| Avg src tokens | 792/779 | 769 /762 | 440/442 | 1K/1K | 2.4K/2.3K | 296/236 | 6.1K/6.4K | - /117K | - /23.4K |
| Avg tgt tokens | 55/58 | 30/31 | 23/23 | 144/146 | 258/258 | 24/25 | 281/277 | - /6.6K | - /104 |

Table 1: Data statistics on summarization corpora. Source is the domain of dataset. Multi-sents. is whether the summaries are multiple sentences or not. All statistics are divided by Train/Test except for BookSum and MScript.

| | | CNNDM | | | NewsRoom | | | XSum | | | PeerRead | | | PubMed | | | Reddit | | | AMI | | | BookSum | | | MScript | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | R | VO | SO | R | VO | SO | R | VO | SO | R | VO | SO | R | VO | SO | R | VO | SO | R | VO | SO | R | VO | SO | R | VO | SO |
| | Random | 19.1 | 18.6 | 14.6 | 10.1 | 2.1 | 9.0 | 9.3 | - | 8.4 | 27.9 | 42.5 | 26.2 | 30.1 | 46.9 | 13.0 | 11.8 | - | 11.3 | 12.0 | 39.3 | 2.4 | 29.4 | 85.8 | 4.9 | 8.1 | 25.2 | 0.1 |
| | Oracle | 42.8 | - | - | 48.1 | - | - | 19.6 | - | - | 46.3 | - | - | 47.0 | - | - | 30.0 | - | - | 32.0 | - | - | 38.9 | - | - | 24.2 | - | - |
| Position | First | **30.7** | 13.1 | **30.7** | **32.2** | **4.4** | **37.8** | 9.1 | - | 8.7 | **32.0** | 40.7 | **30.3** | 27.6 | 44.3 | 13.8 | **15.3** | - | **19.9** | 11.4 | 48.0 | **3.8** | **29.1** | 85.1 | **7.4** | 6.9 | 12.4 | **0.7** |
| | Last | 16.4 | 18.6 | 8.2 | 7.7 | 1.9 | 4.4 | 8.3 | - | 7.0 | 28.9 | 38.5 | 27.0 | 28.9 | 45.2 | 14.0 | 11.2 | - | 10.7 | 7.8 | 42.1 | 2.0 | 26.5 | 85.3 | 3.3 | **8.8** | 19.5 | 0.2 |
| | Middle | 21.5 | 18.7 | 11.8 | 12.4 | 1.9 | 5.6 | 9.1 | - | 9.1 | 29.7 | 40.7 | 22.8 | 28.9 | 45.9 | 12.3 | 11.5 | - | 7.1 | 11.1 | 36.4 | 2.3 | 27.9 | 83.0 | 4.9 | 8.0 | 23.9 | 0.1 |
| Divers. | ConvFall | 21.6 | **57.7** | 15.0 | 10.6 | 4.2 | 7.3 | 8.4 | - | 8.0 | 29.8 | **77.5** | 25.9 | 28.2 | **93.5** | 11.2 | 11.6 | - | 7.5 | **14.0** | **98.6** | 2.4 | 16.9 | **99.7** | 2.2 | 8.5 | **59.2** | 0.2 |
| | Heuris. | 21.4 | 19.8 | 14.6 | 10.5 | 2.4 | 7.6 | 8.4 | - | 8.1 | 29.2 | 36.6 | 24.8 | 27.5 | 59.7 | 10.5 | 11.5 | - | 7.1 | 10.7 | 66.0 | 2.4 | 26.9 | 99.7 | 4.5 | 6.4 | 5.7 | 0.2 |
| Import. | NNear. | 22.0 | 3.3 | 16.6 | 13.5 | 0.5 | 10.0 | **9.8** | - | **10.1** | 30.6 | 8.4 | 26.7 | **31.8** | 9.3 | **15.5** | 13.8 | - | 12.2 | 1.3 | 0.2 | 0.1 | 27.9 | 1.5 | 5.1 | 8.7 | 0.9 | 0.3 |
| | KNear. | 23.0 | 3.9 | 17.7 | 14.0 | 0.7 | 10.9 | 9.3 | - | 9.1 | 30.6 | 9.9 | 27.0 | 29.6 | 10.5 | 15.0 | 10.4 | - | 8.5 | 0.0 | 0.1 | 0.0 | 21.8 | 1.4 | 3.7 | 0.6 | 0.0 | 0.1 |

Table 2: Comparison of different corpora w.r.t the three sub-aspects: POSITION, DIVERSITY, and IMPORTANCE. We averaged R1, R2, and RL as R (See Appendix for full scores). Note that volume overlap (VO) doesn't exist when target summary has a single sentence. (i.e., XSum, Reddit)

This is mainly because the semantic volume assumption maximizes the semantic diversity, but sacrifices other aspects like importance by choosing the outlier sentences over the convex hull.

**Social posts and news articles are biased to the position aspect while the other two aspects appear less relevant.** (Figure 1 (a)) However, XSum requires all aspects equally but with relatively less relevant to any of aspects than the other news corpora.

**Paper summarization is a well-balanced task.** The variance of SO across the three aspects in PeerRead and PubMed is relatively smaller than other corpora. This indicates that abstract summary of the input paper requires the three aspects at the same time. PeerRead has relatively higher SO then PubMed because it only summarize text in Introduction section, while PubMed summarize whole paper text, which is much difficult (almost random performance).

**Conversation, movie script and book summarization are very challenging.** Conversation of spoken meeting minutes includes a lot of witty replies repeatedly (e.g., 'okay.' , 'mm -hmm.' , 'yeah.'), causing importance and diversity measures to suffer. MScript and BookSum which include very long input document seem to be extremely difficult task, showing almost random performance.

### 6.2 Intersection between the sub-aspects

Averaged ratios across the sub-aspects do not capture how the actual summaries overlap with each other. Figure 3 shows Venn diagrams of how sets of summary sentences chosen by different sub-aspects are overlapped each other on average.

**XSum, BookSum, and AMI have high Oracle Recall.** If we develop a mixture model of the three aspects, the Oracle Recall means its upper bound, meaning that another sub-aspect should be considered regardless of the mixture model. This indicates that existing procedures are not enough to cover the Oracle sentences. For example, AMI and BookSum have a lot of repeated noisy sentences, some of which could likely be removed without a significant loss of pertinent information.

**IMPORTANCE and DIVERSITY are less overlapped with each other.** This means that important sentences are not always diverse sentences, indicating that they should be considered together.

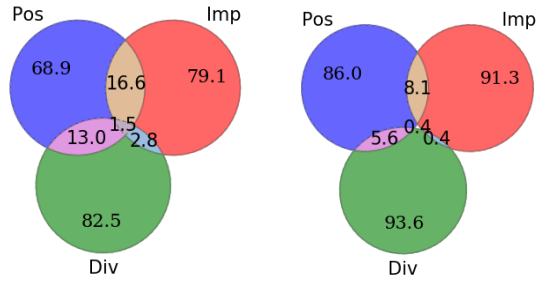


(a) CNNDM (49.4%)




(b) XSum (76.8%)

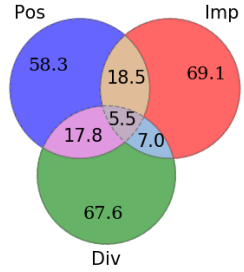


(c) PeerRead (37.6%)

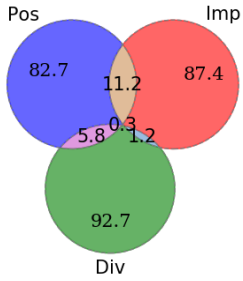


(d) Reddit (68.1%)

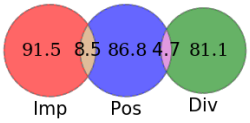


(e) AMI (94.1%)

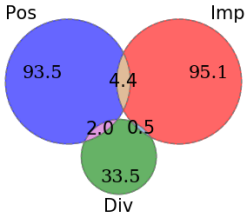


(f) BookSum (87.1%)

Figure 3: Intersection of averaged summary sentence overlaps across the sub-aspects. We use First for Position, ConvexFall for Diversity, and N-Nearest for Importance. The number in the parenthesis called *Oracle Recall* is the averaged ratio of how many the oracle sentences are **NOT** chosen by union set of the three sub-aspect algorithms. Other corpora are in Appendix with their Oracle Recalls: Newsroom(54.4%), PubMed (64.0%) and MScript (99.1%).

### 6.3 Summaries in a embedding space

Figure 4 shows two dimensional PCA projections of a document in CNNDM on the embedding space.

**Source sentences are clustered on the convexhull border, not in the middle.** We conjecture that sentences are not uniformly distributed in the embedding space but their positions gradually move over the convexhull. Target summaries reflect different sub-aspects according to the sample and corpora. For example, many target sentences in CNNDM are near by First-k sentences.

### 6.4 Single-aspect analysis

We calculate the frequency of source sentences overlapped with the oracle summary where the

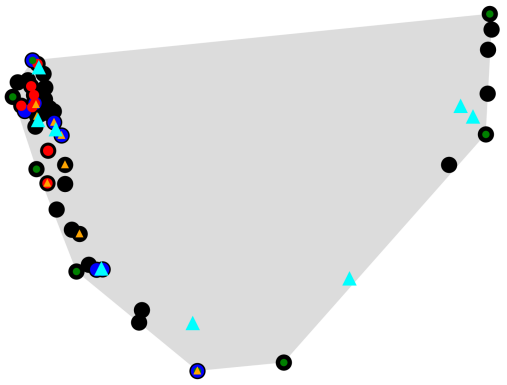

Figure 4: PCA projection of extractive summaries chosen by multiple aspects of algorithms (CNNDM). Source and target sentences are black circles (●) and cyan triangles, respectively. The blue, green, red circles are summary sentences chosen by First, ConvexFall, NN, respectively. The yellow triangles are the oracle sentences. Shaded polygon represents a ConvexHull volume of sample source document. Best viewed in color. Please find more examples in Appendix.

source sentences are ranked differently according to the algorithm of each aspect (See Figure 5). Heavily skewed histograms indicate that oracle sentences are positively (right-skewed) or negatively (left-skewed) related to the sub-aspect.

In most cases, some oracle sentences are overlapped to the first part of the source sentences. Even though their degrees are different, oracle summaries from many corpora (i.e, CNNDM, NewsRoom, PeerRead, BookSum, MScript) are highly related to the Position. Compared to the other corpora, PubMed and AMI contain more top-ranked important sentences in their oracle summaries. News articles and papers tend to find oracle sentences without Diversity (i.e., right-skewed), meaning that non-diverse sentences are frequently selected as part of the oracle.

We also measure how many *new* words occur in abstractive target summaries, by comparing overlap between oracle summaries and document sentences (Table 3). One thing to note is that XSum and AMI have less *new* words in their target summaries. On the other hand, paper datasets (i.e., PeerRead and PubMed) include a lot, indicating that abstract text in academic paper is indeed "abstract".

## 7 Analysis on System Bias

We study how current summarization systems are biased with respect to three sub-aspects. In addition, we show that a simple ensemble of sys-

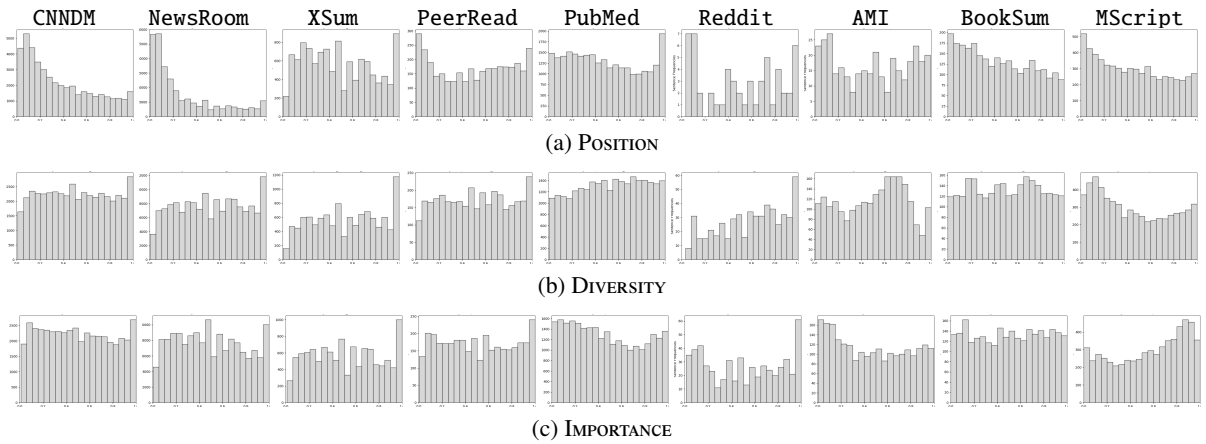


Figure 5: Sentence overlap proportion of each sub-aspect (row) with the oracle summary across corpora (column). y-axis is the frequency of overlapped sentences with the oracle summary. X-axis is the normalized RANK of individual sentences in the input document where size of bin is 0.05. E.g., the first / the most diverse / the most important sentence is in the first bin. If earlier bars are frequent, the aspect is positively relevant to the corpus.

| | R(O,T) | O∩T | | T\S | |
|---|---|---|---|---|---|
| | | Unigram | Bigram | Unigram | Bigram |
| CNNDM | 42.8 | 66.0 | 36.4 | 14.7 | 5.7 |
| Newsroom | 48.1 | 60.7 | 43.4 | 7.8 | 3.4 |
| XSum | 19.6 | 30.4 | 6.9 | 8.4 | 1.2 |
| PeerRead | 46.3 | 48.5 | 27.2 | 20.1 | 8.8 |
| PubMed | 47.0 | 52.1 | 27.7 | 16.7 | 6.7 |
| Reddit | 30.0 | 41.0 | 16.4 | 13.8 | 3.8 |
| AMI | 32.0 | 28.1 | 8.5 | 10.6 | 1.5 |
| BookSum | 38.9 | 25.6 | 8.9 | 6.7 | 1.7 |
| MScript | 38.9 | 13.9 | 4.0 | 0.3 | 0.1 |

Table 3: ROUGE of oracle summaries and averaged N-gram overlap ratios. O, T and S are a set of N-grams from Oracle, Target and Source document, respectively. **R(O,T)** is the averaged ROUGE between oracle and target summaries, showing how similar they are. **O∩T** shows N-gram overlap between oracle and target summaries. The higher the more overlapped words in between. **T\S** is a proportion of N-grams in target summaries not occurred in source document. The lower the more abstractive (i.e., new words) target summaries.

tems shows comparable performance to the single-aspect systems.

**Existing systems.** We compare various extractive and abstractive systems: For extractive systems, we use *K-Means* (Lin and Bilmes, 2010), Maximal Marginal Relevance (*MMR*) (Carbonell and Goldstein, 1998), *cILP* (Gillick and Favre, 2009; Boudin et al., 2015), *TexRank* (Mihalcea and Tarau, 2004), *LexRank* (Erkan and Radev, 2004) and three recent neural systems; *CL* (Cheng and Lapata, 2016), *SumRun* (Nallapati et al., 2017), and *S2SExt* (Kedzie et al., 2018). For abstractive systems, we use *WordILP* (Banerjee et al., 2015) and four neural systems; *S2SAbs* (Rush et al., 2015), *Pointer* (See et al., 2017), *Teacher* (Bengio et al., 2015), and *RL* (Paulus et al., 2017). The detailed description and experimental setup for each algorithm are in Appendix.

**Proposed ensemble systems.** Motivated by the sub-aspect theory (Lin and Bilmes, 2012, 2011), we combine different types of systems together from two different pools of extractive systems: asp from the three best algorithm from each aspect and ext from all extractive systems. For each combination, we choose the sumary sentences randomly among the union set of the predicted sentences (rand) or the most frequent unique sentences (topk).

**Results.** Table 4 shows a comparison of existing and proposed summarization systems on the set of corpora in §5 except for Newsroom[12]. Neural extractive systems such as *CL*, *SumRun* and *S2SExt* outperform the others in general. *LexRank* is highly biased toward the position aspect. On the other hand, *MMR* is extremely biased to the importance aspect on XSum and Reddit. Interestingly, neural extractive systems are somewhat balanced compared to the others. Ensemble systems seem to have the three sub-aspects in balance, compared to the neural extractive systems. They also outperform the others (either ROUGE or SO) on five out of eight datasets.

[12]We exclude it because of its similar behavior as CNNDM.

| | | CNNDM | | | XSum | | | PeerRead | | | PubMed | | | Reddit | | | AMI | | | BookSum | | | MScript | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | R | SO | R(P/D/I) | R | SO | R(P/D/I) | R | SO | R(P/D/I) | R | SO | R(P/D/I) | R | SO | R(P/D/I) | R | SO | R(P/D/I) | R | SO | R(P/D/I) | R | SO | R(P/D/I) |
| extractive | **KMeans** | 22.2 | 16.3 | 14/22/34 | 9.8 | 10.0 | 14/8/90 | 30.9 | 28.3 | 24/28/38 | 30.6 | 14.2 | 31/40/46 | 14.0 | 12.5 | 10/2/82 | 12.3 | 2.5 | 9/6/7 | 27.2 | 4.6 | 5/2/14 | 9.1 | 0.3 | 0/0/9 |
| | **MMR** | 21.6 | 15.2 | 12/24/30 | 9.8 | 10.0 | 14/8/**97** | 29.6 | 24.9 | 26/29/35 | 30.2 | 12.9 | 33/35/42 | 13.6 | 11.5 | 10/3/**88** | 12.3 | 2.5 | 9/6/7 | 29.1 | 6.1 | 4/0/13 | 9.5 | 0.2 | 0/0/28 |
| | **TexRank** | 19.6 | 10.3 | 34/27/27 | 9.9 | 8.5 | 19/11/16 | 23.9 | 12.4 | 32/32/32 | 18.0 | 1.7 | 19/21/20 | 17.7 | 16.7 | 13/9/15 | 11.1 | 0.0 | 17/20/6 | 6.7 | 0.0 | 8/14/8 | 8.2 | 0.2 | 5/9/8 |
| | **LexRank** | 29.3 | 29.5 | 71/29/32 | 11.2 | 11.9 | **61**/15/19 | 29.0 | 24.6 | **66**/35/38 | 26.3 | 7.7 | **56**/27/28 | 18.7 | 18.8 | 46/11/19 | 8.0 | 0.2 | 36/21/12 | 10.5 | 0.8 | 20/20/13 | 12.7 | **0.5** | 20/9/9 |
| | **wILP** | 23.1 | 15.6 | 27/28/29 | 11.1 | 2.1 | 28/19/21 | 20.2 | 16.0 | 23/27/26 | 15.6 | 6.0 | 14/20/18 | 17.4 | 13.5 | 42/16/20 | 5.1 | 0.6 | 17/18/17 | 4.3 | 1.3 | 5/12/7 | 6.8 | 0.1 | 6/8/6 |
| | **CL** | **31.2** | 30.0 | **86**/29/31 | **11.8** | **14.3** | 25/13/19 | 31.3 | 21.8 | 55/35/38 | 26.3 | 9.2 | 41/26/26 | 19.4 | 24.0 | 23/14/23 | 23.1 | 10.3 | 19/23/5 | - | - | -/-/- | 14.0 | 0.2 | 6/8/7 |
| | **SumRun** | 30.5 | 27.1 | 68/29/31 | 11.6 | 13.1 | 14/13/19 | **34.0** | 20.5 | 38/36/37 | 29.4 | 10.8 | 27/28/27 | **20.2** | 19.8 | 23/12/21 | **23.8** | 11.4 | 21/23/6 | - | - | -/-/- | 14.4 | 0.0 | 5/9/9 |
| | **S2SExt** | 30.4 | 28.3 | 74/28/31 | 12.0 | 14.2 | 17/13/19 | 33.9 | 21.1 | 43/35/37 | 29.6 | 10.8 | 26/28/28 | 21.5 | **34.4** | 27/12/26 | 23.4 | 11.9 | 21/24/6 | - | - | -/-/- | 14.3 | 0.0 | 7/9/8 |
| abstractive | **cILP** | 27.8 | x | 43/31/32 | 10.9 | x | 49/15/18 | 28.2 | x | 35/36/38 | 27.8 | x | 23/29/30 | 17.7 | x | **53**/15/17 | 12.5 | x | 22/33/10 | 7.9 | x | 9/19/12 | 10.6 | x | 5/7/7 |
| | **S2SAbs** | 16.3 | x | 4/4/4 | 10.4 | x | 8/7/8 | 9.9 | x | 9/9/9 | 10.2 | x | 10/10/10 | 11.9 | x | 11/7/8 | 20.3 | x | 9/12/1 | - | -x | -/-/- | 14.0 | x | 6/8/8 |
| | **+Pointer** | 23.9 | x | 20/13/14 | 15.6 | x | 12/11/12 | 13.6 | x | 13/13/13 | 11.2 | x | 11/12/11 | 14.3 | x | 14/10/12 | 23.0 | x | 11/13/1 | - | -x | -/-/- | 10.0 | x | 6/7/7 |
| | **+Teacher** | 29.7 | x | 33/21/22 | 17.0 | x | 12/10/12 | 8.7 | x | 8/8/8 | 11.3 | x | 12/12/11 | 15.3 | x | 15/10/11 | 20.2 | x | 9/13/1 | - | -x | -/-/- | 16.0 | x | 7/10/8 |
| | **+RL** | 30.2 | x | 34/23/24 | 18.1 | x | 12/11/12 | 30.1 | x | 30/29/28 | 12.9 | x | 13/14/13 | 16.7 | x | 1/1/14 | 23.6 | x | 11/13/2 | - | -x | -/-/- | **16.2** | x | 7/10/8 |
| ensemble | ASP(rand) | 23.3 | 19.5 | 40/**38**/**38** | 9.0 | 9.0 | 40/**39**/38 | 29.6 | 25.5 | 54/49/52 | 29.5 | 13.5 | 49/**47**/**51** | 12.5 | 5.2 | 21/11/22 | 8.9 | 0.9 | 44/**50**/**20** | **29.8** | **6.4** | 57/33/55 | 8.4 | 0.4 | **32**/36/37 |
| | ASP(topk) | 29.1 | **30.4** | 71/31/31 | 9.0 | 8.8 | 43/**39**/38 | 30.5 | 28.2 | 63/**54**/**57** | 29.7 | 14.0 | 55/48/52 | 12.3 | 15.6 | 41/**41**/38 | 9.9 | 1.5 | **99**/24/11 | 29.6 | 6.2 | **58**/**34**/**56** | 8.3 | **0.5** | 30/**37**/**38** |
| | EXT(rand) | 24.2 | 20.2 | 39/25/27 | 10.2 | 10.9 | 17/13/23 | 29.4 | 23.5 | 42/37/39 | 31.7 | 16.0 | 37/34/38 | 14.2 | 17.7 | 22/12/13 | 18.7 | 5.1 | 21/28/8 | 28.6 | 5.4 | 37/24/42 | 6.7 | 0.0 | 5/9/13 |
| | EXT(topk) | 29.4 | 30.3 | 58/25/28 | 11.0 | 11.8 | 18/10/37 | 33.0 | **33.0** | 54/39/44 | **34.1** | **20.5** | 41/35/40 | 16.4 | 20.8 | 21/11/52 | **23.8** | **13.4** | 23/27/6 | 28.5 | 5.2 | 37/24/43 | 7.4 | 0.0 | 6/8/11 |

Table 4: Comparison of different systems using the averaged ROUGE scores (1/2/L) with target summaries (R) and averaged oracle overlap ratios (SO, only for extractive systems). We calculate R between systems and selected summary sentences from each sub-aspect (R(P/D/I)) where each aspect uses the best algorithm: First, ConvexFall and NNearest. R(P/D/I) is rounded by the decimal point. - indicates the system has too few samples to train the neural systems. x indicates SO is not applicable because abstractive systems have no sentence indices. The best score for each corpora is shown in bold with different colors.

## 8 Conclusion and Future Directions

We define three sub-aspects of text summarization: position, diversity, and importance. We analyze how different domains of summarization dataset are biased to these aspects. We observe that news articles strongly reflect the position aspect, while the others do not. In addition, we investigate how current summarization systems reflect these three sub-aspects in balance. Each type of approach has its own bias, while neural systems rarely do. Simple ensembling of the systems shows more balanced and comparable performance than single ones.

We summarize actionable messages for future summarization research:

- Different domains of datasets except for news articles pose new challenges to the appropriate design of summarization systems. For example, summarization of conversations (e.g., `AMI`) or dialogues (`MSCript`) need to filter out repeated, rhetorical utterances. Book summarization (e.g., `BookSum`) is very challenging due to its extremely large document size. Here current neural encoders suffer from computation limits.
- Summarization systems to be developed should clearly state their computational limits as well as effectiveness in each aspect and in each corpus domain. A good summarization system should reflect different kinds of the sub-aspects harmoniously, regardless of corpus bias. Developing such bias-free or robust models can be very important for future directions.
- Nobody has clearly defined the deeper nature of meaning abstraction yet. A more theoretical study of summarization, and the various aspects, is required. A recent notable example is Peyrard (2019a)'s attempt to theoretically define different quantities of `importance` aspect, and demonstrate the potential of the framework on an existing summarization system. Similar studies can be applied to other aspects and their combinations in various systems and different domains of corpora.
- One can repeat our bias study on evaluation metrics. Peyrard (2019b) showed that widely used evaluation metrics (e.g., ROUGE, Jensen-Shannon divergence) are strongly mismatched in scoring summary results. One can compare different measures (e.g., n-gram recall, sentence overlaps, embedding similarities, word connectedness, centrality, importance reflected by discourse structures), and study bias of each with respect to systems and corpora.

## Acknowledgements

This work would not have been possible without the efforts of the authors who kindly share the summarization datasets publicly. We thank Rada Mihalcea for sharing the book summarization dataset. We also thank Diane J. Litman, Taylor Berg-Kirkpatrick, Hiroaki Hayashi, and anonymous reviewers for their helpful comments.

## References


Miltiadis Allamanis, Hao Peng, and Charles Sutton. 2016. A convolutional attention network for extreme summarization of source code. *arXiv preprint arXiv:1602.03001*.

Siddhartha Banerjee, Prasenjit Mitra, and Kazunari Sugiyama. 2015. Multi-document abstractive summarization using ilp based multi-sentence compression. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)*.

C Bradford Barber, David P Dobkin, David P Dobkin, and Hannu Huhdanpaa. 1996. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software (TOMS)*, 22(4):469–483.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179.

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 481–490. Association for Computational Linguistics.

Florian Boudin and Emmanuel Morin. 2013. Keyphrase extraction for n-best reranking in multi-sentence compression. In *North American Chapter of the Association for Computational Linguistics (NAACL)*.

Florian Boudin, Hugo Mougard, and Benoit Favre. 2015. Concept-based summarization using integer linear programming: From concept pruning to multiple optimal solutions. In *Conference on Empirical Methods in Natural Language Processing (EMNLP) 2015*.

Jaime Carbonell and Jade Goldstein. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 335–336. ACM.

Jean Carletta, Simone Ashby, Sebastien Bourban, Mike Flynn, Mael Guillemot, Thomas Hain, Jaroslav Kadlec, Vasilis Karaiskos, Wessel Kraaij, Melissa Kronenthal, et al. 2005. The ami meeting corpus: A pre-announcement. In *International Workshop on Machine Learning for Multimodal Interaction*, pages 28–39. Springer.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. *arXiv preprint arXiv:1603.07252*.

Trevor Cohn and Mirella Lapata. 2008. Sentence compression beyond word deletion. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 137–144. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Greg Durrett, Taylor Berg-Kirkpatrick, and Dan Klein. 2016. Learning-based single-document summarization with compression and anaphoricity constraints. *arXiv preprint arXiv:1603.08887*.

Harold P Edmundson. 1969. New methods in automatic extracting. *Journal of the ACM (JACM)*, 16(2):264–285.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, pages 457–479.

Katja Filippova. 2010. Multi-sentence compression: finding shortest paths in word graphs. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 322–330. Association for Computational Linguistics.

Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. 2010. Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd international conference on computational linguistics*, pages 340–348. Association for Computational Linguistics.

Sebastian Gehrmann, Yuntian Deng, and Alexander M Rush. 2018. Bottom-up abstractive summarization. *arXiv preprint arXiv:1808.10792*.

Shima Gerani, Yashar Mehdad, Giuseppe Carenini, Raymond T Ng, and Bita Nejat. 2014. Abstractive summarization of product reviews using discourse structure. In *Proceedings of EMNLP*.

Dan Gillick and Benoit Favre. 2009. A scalable global model for summarization. In *Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing*, pages 10–18. Association for Computational Linguistics.

Philip John Gorinski and Mirella Lapata. 2015. Movie script summarization as graph-based scene extraction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1066–1076.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2003. English gigaword. *Linguistic Data Consortium, Philadelphia*, 4(1):34.

Max Grusky, Mor Naaman, and Yoav Artzi. 2018. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In *Proceedings of the 2018 Conference of the North American Chapter of*

*the Association for Computational Linguistics: Human Language Technologies*, pages 708–719, New Orleans, Louisiana. Association for Computational Linguistics.

Kai Hong and Ani Nenkova. 2014. Improving the estimation of word importance for news multi-document summarization. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 712–721.

Dongyeop Kang, Waleed Ammar, Bhavana Dalvi, Madeleine van Zuylen, Sebastian Kohlmeier, Eduard Hovy, and Roy Schwartz. 2018. A dataset of peer reviews (peerread): Collection, insights and nlp applications. In *Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*, New Orleans, USA.

Chris Kedzie, Kathleen McKeown, and Hal Daume III. 2018. Content selection in deep learning models of summarization. *arXiv preprint arXiv:1810.12343*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8.

Chin-Yew Lin and Eduard Hovy. 1997. Identifying topics by position. In *Fifth Conference on Applied Natural Language Processing*.

Chin-Yew Lin and Eduard Hovy. 2000. The automated acquisition of topic signatures for text summarization. In *Proceedings of the 18th conference on Computational linguistics-Volume 1*, pages 495–501. Association for Computational Linguistics.

Hui Lin and Jeff Bilmes. 2010. Multi-document summarization via budgeted maximization of submodular functions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 912–920. Association for Computational Linguistics.

Hui Lin and Jeff Bilmes. 2011. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 510–520. Association for Computational Linguistics.

Hui Lin and Jeff A Bilmes. 2012. Learning mixtures of submodular shells with application to document summarization. *arXiv preprint arXiv:1210.4871*.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A Smith. 2015. Toward abstractive summarization using semantic representations. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1077–1086.

Daniel Marcu. 1999. Discourse trees are good indicators of importance in text. *Advances in automatic text summarization*, 293:123–136.

Ryan McDonald. 2007. *A study of global inference algorithms in multi-document summarization*. Springer.

Yashar Mehdad, Giuseppe Carenini, and Raymond Ng. 2014. Abstractive summarization of spoken and written conversations based on phrasal queries. In *Proc. of ACL*, pages 1220–1230.

Rada Mihalcea and Hakan Ceylan. 2007. Explorations in automatic book summarization. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 404–411.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018a. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *arXiv preprint arXiv:1808.08745*.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018b. Ranking sentences for extractive summarization with reinforcement learning. *arXiv preprint arXiv:1802.08636*.

Jessica Ouyang, Serina Chang, and Kathy McKeown. 2017. Crowd-sourced iterative annotation for narrative summarization corpora. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 46–51.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Maxime Peyrard. 2019a. A simple theoretical model of importance for summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1059–1073, Florence, Italy. Association for Computational Linguistics.

Maxime Peyrard. 2019b. Studying summarization evaluation metrics in the appropriate scoring range. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5093–5100, Florence, Italy. Association for Computational Linguistics.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Kristian Woodsend and Mirella Lapata. 2011. Learning to simplify sentences with quasi-synchronous grammar and integer programming. In *Proceedings of the conference on empirical methods in natural language processing*, pages 409–420. Association for Computational Linguistics.

Dani Yogatama, Fei Liu, and Noah A Smith. 2015. Extractive summarization by maximizing semantic volume. In *EMNLP*, pages 1961–1966.

David Zajic, Bonnie Dorr, and Richard Schwartz. 2004. Bbn/umd at duc-2004: Topiary. In *Proceedings of the HLT-NAACL 2004 Document Understanding Workshop, Boston*, pages 112–119.

# Lost in Evaluation: Misleading Benchmarks for Bilingual Dictionary Induction

**Yova Kementchedjhieva**
University of Copenhagen
yova@di.ku.dk

**Mareike Hartmann**
University of Copenhagen
hartmann@di.ku.dk

**Anders Søgaard**
University of Copenhagen
soegaard@di.ku.dk

## Abstract

The task of bilingual dictionary induction (BDI) is commonly used for intrinsic evaluation of cross-lingual word embeddings. The largest dataset for BDI was generated automatically, so its quality is dubious. We study the composition and quality of the test sets for five diverse languages from this dataset, with concerning findings: (1) a quarter of the data consists of proper nouns, which can be hardly indicative of BDI performance, and (2) there are pervasive gaps in the gold-standard targets. These issues appear to affect the ranking between cross-lingual embedding systems on individual languages, and the overall degree to which the systems differ in performance. With proper nouns removed from the data, the margin between the top two systems included in the study grows from 3.4% to 17.2%. Manual verification of the predictions, on the other hand, reveals that gaps in the gold standard targets artificially inflate the margin between the two systems on English to Bulgarian BDI from 0.1% to 6.7%. We thus suggest that future research either avoids drawing conclusions from quantitative results on this BDI dataset, or accompanies such evaluation with rigorous error analysis.

## 1 Introduction

Bilingual dictionary induction (BDI) refers to retrieving translations of individual words. The task has been widely used for intrinsic evaluation of cross-lingual embedding algorithms, which aim to map two languages into the same embedding space, for transfer learning purposes (Klementiev et al., 2012). Recently, Glavas et al. (2019) reported limited evidence in support of this practice—they found that cross-lingual embeddings optimized for a BDI evaluation metric were not necessarily better on downstream tasks. Here, we study BDI evaluation in itself, as has been done for other evaluation methods in the past (cf. Faruqui et al., 2016's work on word similarity), with concerning findings about its reliability.

A massive dataset of 110 bilingual dictionaries, known as the MUSE dataset, was introduced in early 2018 along with a strong baseline (Conneau et al., 2018). Subsets of the MUSE dictionaries have been used for model comparison in the evaluation of numerous cross-lingual embedding systems developed since (cf. Grave et al., 2018; Jawanpuria et al., 2018; Hoshen and Wolf, 2018a,b; Wada and Iwata, 2018; Joulin et al., 2018). Even though the field has been very active, progress has been incremental for most language pairs. Moreover, there have been very few attempts at a linguistically-informed error analysis of BDI performance as measured on MUSE (cf. Kementchedjhieva et al., 2018). This is problematic for two reasons: on one hand, most systems greatly vary in their approach and architecture, so it is difficult to identify the source of the reported performance gains; on the other hand, the MUSE dataset was compiled automatically, with no manual post-processing to clean up noise, so the real impact of the performance gains is unclear.

In this work, we study the composition and quality of the MUSE data for five diverse languages: German, Danish, Bulgarian, Arabic and Hindi. A manual part-of-speech annotation of the test sets for these languages reveals a strikingly high number of proper nouns. We refer to linguistic literature to argue that proper nouns, having no lexical meaning but rather just a referential function, cannot reliably be used in the evaluation of word-level translation systems. We find that excluding proper noun pairs from the test dictionaries for the aforementioned languages affects the ranking and degree of performance gaps between five of the most influential recent systems for BDI.

With a new, more reliable ranking at hand, we



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3336–3341,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

perform qualitative analysis on the performance gap between the best and second best systems for Bulgarian. This reveals another major issue with the data: limited coverage of morphological variants for the target words. Through manual verification of the models' predictions, we find that the gap in performance between the two systems is far smaller than previously perceived.

The uncovered issues of high noise levels (proper nouns) and limited coverage (missing gold standard targets) clearly have a crucial impact on BDI results obtained on the MUSE dataset, and need to be addressed. Filtering out proper nouns could be achieved automatically, by checking against gazetteers of named entities. We find that an automatic procedure for the filling of missing targets, however, yields only minor improvements. Until an alternative solution to the latter problem is found, we urge researchers to be cautious when reporting quantitative results on MUSE, and to account for the problems presented here through manual verification and analysis of the results. We share our part-of-speech annotations, such that future work can use this resource for analysis purposes. [1]

## 2 Bilingual Dictionary Induction

Improvements on BDI mostly stem from developments in the space of cross-lingual embeddings, which use BDI for intrinsic evaluation.

**Systems** Five influential recent systems for cross-lingual embeddings are MUSE (Conneau et al., 2018), which can be supervised (**MUSE-S**) or unsupervised (**MUSE-U**); VecMap, which also can be supervised (**VM-S**) (Artetxe et al., 2018a) or unsupervised (**VM-U**) (Artetxe et al., 2018b); and RCSLS (Joulin et al., 2018), a supervised system (**RCSLS**), which scores best on BDI out of the five. We refer the reader to the respective publications for a general description of the systems.

**Metrics** Performance on BDI in these works is evaluated by verifying the system-retrieved translations for a source word against a set of gold-standard targets. The metric used is Precision at $k$ (P@$k$), which measures how often the set of $k$ top predictions contains one of the gold-standard targets, i.e. what is the ratio of True Positives to the sum of True Positives and False Positives.

[1] Available at https://github.com/coastalcph/MUSE_dicos

**Data** All systems listed above report results on one or both of two test sets: the MUSE test sets Conneau et al. (2018) and/or the Dinu test sets (Dinu et al., 2015; Artetxe et al., 2017). Similarly to MUSE, the Dinu dataset was compiled automatically (from Europarl word-alignments), but it only covers four languages. Due to the bigger size of MUSE (110 language pairs), we deem its impact larger and focus our study entirely on it.

## 3 Annotation-based observations

In order to gain insights into the linguistic composition of the MUSE dictionaries, we employ annotators fluent in German, Danish, Bulgarian, Arabic and Hindi (hereafter, DE, DA, BG, AR, HI) to annotate the entire dictionaries from English to one of these languages (hereafter, from-EN) and the entire dictionaries from these languages to English (hereafter, to-EN) with part-of-speech (POS) tags. Details on the annotation procedure can be found in Appendix A. Below, we discuss our findings on the POS composition of the data, and we evaluate the performance of RCSLS per POS tag.[2]

### 3.1 Analysis of POS composition

The average percentage of common nouns, proper nouns, verbs, and adjectives/adverbs in the dictionaries to-EN was respectively 49.6, 24.9, 12.5, and 12.9.[3] Nouns constitute half of the dictionaries' volume, while verbs and adjectives/adverbs collectively make up only about a fourth of the average dictionary. A skewed ratio between these three categories is not surprising: in the EWT dependency treebank, for example, which contain gold-standard POS tags, the proportion of noun, verb and adjective/adverb types is 34, 17 and 14 percent, respectively. Notice, however, that in the case of the MUSE data, the ratio is even more skewed in favour of nouns over the other two categories.

The large number of proper nouns in the dictionaries seems even more problematic. Proper nouns are considered to have no lexical meaning, but rather just a referential function (Pierini, 2008). Personal names usually refer to a specific referent in a given context, but they can, in general, be attributed to different referents across different contexts, and they are almost univer-

[2] For all experiments, we use the pretrained embeddings of Bojanowski et al. (2017), trained on Wikipedia.

[3] The numbers were similar across from-EN dictionaries.

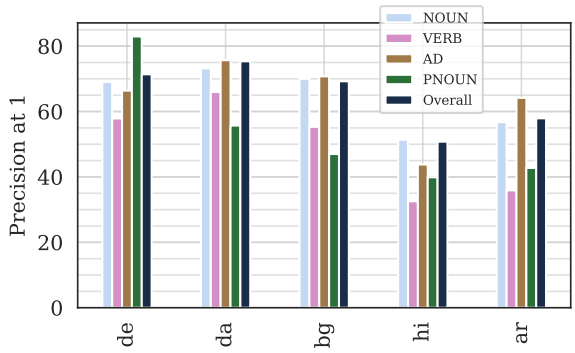


Figure 1: Precision of RCSLS by POS tag on to-EN data.

sally interchangeable in any given context. Some personal names and most place and organization names may have a unique referent, e.g. *Barack Obama, Wisconsin, Skype*, but these names still do not carry a *sense*, their referent is resolved through access to encyclopedic knowledge (Pierini, 2008). Considering that the pretrained embeddings which we use were trained on Wikipedia, we can expect that such encyclopedic information would indeed appear in the context of certain unique names, but importantly, the alignability of the embeddings for such entities would depend on the level of parallelism between the contents of Wikipedia articles in the different languages.

With these considerations in mind, one should wonder how stable the representation of names can be in an embedding space. This question has previously been raised by Artetxe et al. (2017). We address it empirically below.

### 3.2 Evaluation by POS

Figure 1 shows the precision of the RCSLS embedding alignment method on different POS segments of the test data in mapping to-EN (results from-EN were similar and are shown in Appendix B). Verbs pose a greater challenge to BDI systems than nouns and adjectives do. Generally, we can attribute this observation to the higher abstraction of concepts described by verbs. This is a known problem for word embedding methods in general (Gerz et al., 2016), which BDI systems naturally inherit.

With respect to proper nouns, we observe that they indeed introduce a level of instability in the evaluation of BDI systems. Notice that while the other parts of speech follow a similar pattern across languages, with higher precision obtained for nouns and adjectives/adverbs than for verbs,

| Corpora | NOUN | VERB | AD | PNOUN |
|---|---|---|---|---|
| Wikipedia | 69.0 | 57.9 | 66.4 | 83.0 |
| Mixed* | 64.0 | 55.5 | 59.4 | 37.6 |

Table 1: Comparison in performance by POS category with two different embedding sets. * The out-of-vocabulary rate for items in the dictionaries is negligible: 2, 0, and 1 for NOUN, VERB , and AD , respectively.

relative precision on proper nouns is highly variable. For DE, proper nouns are easier to translate than other parts of speech by a margin of 15%, for HI and AR they are easier than nouns and adjectives/adverbs, but harder than verbs, and for DA and BG they are hardest out of all four categories. We looked into the individual word pairs marked as proper nouns in the DE and DA data, as these languages are related and RCSLS performs comparably on them otherwise, and did not find any patterns that could explain the large differences. In fact, between the 384 proper noun pairs in the EN-DE dictionary and the 330 proper noun pairs in the EN-DA dictionary, there was an overlap of 279 pairs, retrieved with precision of 89.21% in the EN-DE setting and 51.30% in the EN-DA setting. We conjecture that this result relates to the level of parallel content between the Wikipedia dumps for the different language pairs, which is likely higher for EN-DE , since the dumps for these languages are also closer in size: 5.8M articles in EN, 2.3M in DE (and only 0.2M in DA).[4]

We evaluate this hypothesis through an experiment where we train an RCSLS alignment for DE-EN using the DE embeddings of Artetxe et al. (2017), trained on SdeWaC (Baroni et al., 2009) and the EN embeddings of Dinu et al. (2015), trained on ukWaC (Baroni et al., 2009), Wikipedia and the BNC [5] corpora. The level of parallel content between the data used to train the two sets of embeddings is thus far more limited in this case, and the DE embeddings are not explicitly trained on Wikipedia data. Table 1 summarizes the results: while with the new embeddings performance is somewhat reduced for nouns, verbs and adjectives/adverbs, precision at 1 for proper nouns, in particular, drops by over 50%, indicating that this category of test word pairs is indeed highly sensitive to the nature of the training data.

[4] https://meta.wikimedia.org/wiki/List_of_Wikipedias
[5] Available at http://www.natcorp.ox.ac.uk

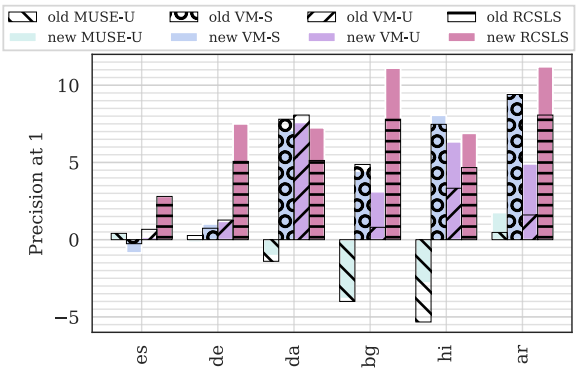


Figure 2: Absolute difference in performance on from-EN BDI, relative to MUSE-S. Pattern-filled bars show results as estimated on the original data (*old*), while colored bars show results as estimated on the cleaned data (*new*).

### 3.3 Re-ranking on clean data

Based on the analysis presented above, we removed all pairs that were annotated as proper nouns and all pairs that were marked as invalid during the annotation process.[6] This clean-up resulted in a drop in the size of the test dictionaries of about 25% on average. A detailed size comparison between the old test dictionaries and their new cleaned versions is presented in the top rows of Table 4 in Appendix B. Figure 2 visualizes a re-evaluation of the five systems for BDI listed in Section 2, on the original test data and on the new clean versions of the test dictionaries from-EN.[7] The results are reported in terms of change in performance relative to MUSE-S (chosen as a baseline) as estimated on the original MUSE data (pattern-filled bars) and on the cleaned version of the data (colored bars). The absolute system performances before and after the clean-up can be found in Table 4 in Appendix B.

We see that the ranking between the models changes most notably for AR, where RCSLS appears inferior to VM-S on the original test data, but on the clean data it emerges as best. For BG, the evaluation on the clean test data reveals that RCSLS outperforms the next best system, VM-S, by a larger factor than it appeared on the original test data. Lastly, for DA, evaluation on the original test data makes RCSLS seem far inferior to VM-S and VM-U, but on the clean test data we see that it outperforms VM-S and matches the performance of VM-U. These observations show that the noise coming from proper nouns has a large impact on the perceived ranking and difference in performance between systems.

## 4 *False* False Positives

With a more reliable estimate of the models' performance at hand, we next manually study the remaining performance gap between RCSLS, the best-performing model overall, and VM-S, the second best model overall, for EN–BG.[8] We present some examples in Table 2 and more can be found in Table 5, Appendix C.

We find that there are 125 source words that RCSLS translated correctly and VM-S did not. Upon closer inspection, we find that for 54% of these words, both RCSLS and VM-S predicted a valid translation, but RCSLS predicted a more *canonical* translation, which was listed among the gold-standard targets, while VM-S predicted another word form that was missing from the list of gold-standard targets. By more canonical we mean, for example, indefinite instead of definite forms of nouns and adjectives (see Ex. A, Table 2, masculine instead of feminine or neuter forms of adjectives (see Ex. B), singular instead of plural forms. To the extent that a more canonical translation should be considered better, RCSLS is definitely showing superiority over VM-S. It is not clear, however, if that should be the case, since for some words, the test dictionary exhibits higher coverage than for others, i.e. the less canonical translations are not omitted by design, but appear to be accidental gaps.

Another 19% of the instances where RCSLS outperformed VM-S, we find to be clear cases of a missing translation in the test dictionary, i.e. not a missing form of a listed target, but a missing synonym or a missing sense altogether (see Ex. C and D).

The two types of errors in precision at 1 discussed above can be considered cases of *false* False Positives, because they really should have been True Positives. The remaining 27% of the gap between the two models' performance indeed illustrate that RCSLS provides better translations in some cases (see Ex. E).

Notice, however, that it is not the case that RCSLS outperformed VM-S in all cases–for 50 test words, VM-S predicted a correct translation and RCSLS did not. Among these, there are cases of missing translations from the dictionary as well

[6] The latter constitute less than 1% of the removed data.

[7] The to-EN results were similar, see Appendix B.

[8] We also analyzed EN–DE, with very similar results.

| Ex. | SRC | TGT | RCSLS | VM-S | Description |
|---|---|---|---|---|---|
| A | joke | шега<br>лаф<br>виц | шега [INDEF] | шегата [DEF] | definite form missing from targets |
| B | remembered | запомнен | запомнен[MASC] | запомнена[FEM] | feminine form missing from targets |
| C | hide | скриване | скриване [NOUN] | скрива̀т[VERB] | *hide* as a verb vs. *hide* as a noun |
| D | bench | пейка<br>пейката | пейка | скамейка | synonym missing from targets |
| E | depot | депо | депо | гара | VM-S predicted 'train station' |
| F | crowned | коронован | коронована[FEM] | коронован[MASC] | feminine form missing from targets |
| G | pond | езерце | къщичка | езерце | RCSLS predicted 'cottage' |
| H | grants | субсидии | стипендии | стипендии | synonym missing from targets |
| I | armies | армии | армиите | армиите | definite form missing from targets |

Table 2: Example translations from EN to BG. Underlined forms are more canonical. Grey forms are incorrect.

(see Ex. F), but they can explain less of the lack in performance of RCSLS, i.e. 50% of the translations of RCSLS are indeed erroneous (see Ex. G).

To summarize, originally the performance gap between the two models appeared to be $(125 - 50)/1125 * 100 = 6.67\%$, while after the manual verification, it is $(27\% * 125 - 50\% * 50)/1125 * 100 = 0.1\%$.[9] Such a substantial narrowing in the gap between the two models clearly indicates that conclusions drawn on the original result, i.e. that RCSLS is far superior that VM-S for this language pair, is hardly supported by the updated result.

A surface analysis of the subset of words for which neither RCSLS nor VM-S retrieved correct translations revealed similar patterns of extensive false False Positives, due to gaps in the coverage of the dictionary (see Ex. H and I). Our takeaway from these observations is two-fold. Firstly, when RCSLS retrieves a correct target form, it also usually retrieves its most *canonical* form. More importantly, the evaluation of BDI systems on even the cleaned test dictionaries still does not represent accurately the differences in quality between them, due to major gaps in the coverage of the test dictionaries.

## 5 Concluding remarks

Our study of the MUSE dataset revealed two striking problems: a high level of noise coming from proper nouns, and an issue of *false* False Positives, due to gaps in the gold-standard targets. The former problem, we conjecture, can be solved by filtering names out with gazetteers. The quality of this solution would depend on the coverage of the gazetteers. The more challenging problem, however, is filling in the gaps, especially in terms of inflectional forms. We carried out preliminary experiments aiming to enrich the EN–BG and EN-DE dictionaries. We extracted additional word forms of verbal and nominal targets from the UniMorph inflectional tables (Kirov et al., 2018), according to a manually designed morphosyntactic correspondence map.[10] Unfortunately, due to limited coverage of the UniMorph data, and, in the case of BG, limited vocabulary of the pretrained embeddings, the impact of this procedure was almost negligible. Alternative approaches for enrichment exists, of course, but we wonder how worthwhile further efforts would be. That is, especially in light of Glavas et al. 2019's findings that BDI performance is not necessarily indicative of cross-lingual embedding quality. We therefore hope that our work adds weight to the call of Glavas et al. (2019) for more reliable evaluation methods in cross-lingual embedding research. When BDI performance is used for evaluation purposes, it should be accompanied by manual verification, of the type presented here.

## 6 Acknowledgements

We thank Marcel Bollmann, Matthew Lamm, Maria Barrett, Meriem Beloucif, Mostafa Abdou, Rahul Aralikatte and Victor Petrén Hansen for help with annotations. We would also like to thank Adam Lopez, Andreas Grivas and Sameer Bansal for useful feedback on drafts of the paper, and to the anonymous reviewers for their comments and suggestions. Anders Søgaard was supported by a Google Focused Research Award; Mareike Hartmann by the Carlsberg Foundation.

[9] 1125 is the total dictionary size.

[10] Details can be found in Appendix D.

## References


Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 451–462.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018a. Generalizing and Improving Bilingual Word Embedding Mappings with a Multi-Step Framework of Linear Transformations. In *Proceedings of AAAI 2018*.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018b. A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 789–798.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The wacky wide web: a collection of very large linguistically processed web-crawled corpora. *Language resources and evaluation*, 43(3):209–226.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word Translation Without Parallel Data. In *Proceedings of ICLR 2018*.

Georgiana Dinu, Angeliki Lazaridou, and Marco Baroni. 2015. Improving Zero-Shot Learning by Mitigating the Hubness Problem. *ICLR 2015 Workshop track*, pages 1–10.

Manaal Faruqui, Yulia Tsvetkov, Pushpendre Rastogi, and Chris Dyer. 2016. Problems with evaluation of word embeddings using word similarity tasks. *arXiv preprint arXiv:1605.02276*.

Daniela Gerz, Ivan Vulić, Felix Hill, Roi Reichart, and Anna Korhonen. 2016. SimVerb-3500: A Large-Scale Evaluation Set of Verb Similarity. In *Proceedings of EMNLP*.

Goran Glavas, Robert Litschko, Sebastian Ruder, and Ivan Vulic. 2019. How to (Properly) Evaluate Cross-Lingual Word Embeddings: On Strong Baselines, Comparative Analyses, and Some Misconceptions. In *ACL*.

Edouard Grave, Armand Joulin, and Quentin Berthet. 2018. Unsupervised alignment of embeddings with wasserstein procrustes. *arXiv preprint arXiv:1805.11222*.

Yedid Hoshen and Lior Wolf. 2018a. An iterative closest point method for unsupervised word translation. *CoRR*, abs/1801.06126.

Yedid Hoshen and Lior Wolf. 2018b. Non-adversarial unsupervised word translation. *arXiv preprint arXiv:1801.06126*.

Pratik Jawanpuria, Arjun Balgovind, Anoop Kunchukuttan, and Bamdev Mishra. 2018. Learning multilingual word embeddings in latent metric space: a geometric approach. *arXiv preprint arXiv:1808.08773*.

Armand Joulin, Piotr Bojanowski, Tomas Mikolov, Hervé Jégou, and Edouard Grave. 2018. Loss in translation: Learning bilingual word mapping with a retrieval criterion. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Yova Kementchedjhieva, Sebastian Ruder, Ryan Cotterell, and Anders Søgaard. 2018. Generalizing procrustes analysis for better bilingual dictionary induction. *arXiv preprint arXiv:1809.00064*.

Christo Kirov, Ryan Cotterell, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sebastian Mielke, Arya D McCarthy, Sandra Kübler, et al. 2018. Unimorph 2.0: Universal morphology. *arXiv preprint arXiv:1810.11101*.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing crosslingual distributed representations of words. In *Proceedings of COLING 2012*, pages 1459–1474. The COLING 2012 Organizing Committee.

Patrizia Pierini. 2008. Opening a pandora's box: Proper names in english phraseology. *Linguistik Online*, 36.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 173–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Takashi Wada and Tomoharu Iwata. 2018. Unsupervised cross-lingual word embedding by multilingual neural language models. *CoRR*, abs/1809.02306.

# Towards Realistic Practices In Low-Resource Natural Language Processing: The Development Set

**Katharina Kann, Kyunghyun Cho and Samuel R. Bowman**
New York University, USA
{kann, kyunghyun.cho, bowman}@nyu.edu

## Abstract

Development sets are impractical to obtain for real low-resource languages, since using all available data for training is often more effective. However, development sets are widely used in research papers that purport to deal with low-resource natural language processing (NLP). Here, we aim to answer the following questions: Does using a development set for early stopping in the low-resource setting influence results as compared to a more realistic alternative, where the number of training epochs is tuned on development *languages*? And does it lead to overestimation or underestimation of performance? We repeat multiple experiments from recent work on neural models for low-resource NLP and compare results for models obtained by training with and without development sets. On average over languages, absolute accuracy differs by up to $1.4\%$. However, for some languages and tasks, differences are as big as $18.0\%$ accuracy. Our results highlight the importance of realistic experimental setups in the publication of low-resource NLP research results.

## 1 Introduction

Parametric machine learning models are frequently trained by minimizing the loss on the training set $\mathcal{T}$,

$$\mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta}) = \sum_{x \in \mathcal{T}} l\left(\theta, x\right), \qquad (1)$$

for model parameters $\theta$ and a predefined loss function $l$. Gradient-based optimizers minimize this loss $\mathcal{L}(\boldsymbol{\theta})$ by updating $\theta$ in the direction of the gradient $\nabla\mathcal{L}(\boldsymbol{\theta})$. A low loss characterizes a model which makes accurate predictions for examples in $\mathcal{T}$. However, since $\mathcal{T}$ is finite, overfitting the training set might lead to poor generalization performance. One way to avoid fitting Equation 1 too closely is *early stopping*: a separate *development* or *validation* set is used to end training as soon as the loss on the development set $\mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta})$ starts increasing or model performance on the development set $\mathcal{D}$ starts decreasing. The best set of parameters $\boldsymbol{\theta}$ is used in the final model.

| | # train | # dev | ES |
|---|---|---|---|
| **Bollmann et al. (2018)** | 5k | 12k-46k | Yes |
| Kann et al. (2018) | 400-700 | 100-200 | Yes |
| Makarov and Clematide (2018) | 100 | 1k | Yes |
| **Sharma et al. (2018)** | 100 | 100 | Yes |
| Schulz et al. (2018) | 1k-21k | 9k | N/A |
| **Upadhyay et al. (2018)** | 500 | 1k | Yes |

Table 1: Number of examples used for training and development in recent low-resource NLP experiments; *ES*=early stopping on the development set. Experiments from papers in bold will be revisited here.

This works well when large amounts of data are available to create training, development and test splits. Recently, however, with the success of pre-training (Peters et al., 2018; Devlin et al., 2019) and multi-task learning (Caruana, 1997; Ruder, 2017; Wang et al., 2019) approaches, neural models are showing promising results on various natural language processing (NLP) tasks also in *low-resource* or *few-shot* settings (Johnson et al., 2017; Kann et al., 2017; Yu et al., 2018). Often, the high-resource experimental setup and training procedure are kept unchanged, and the size of the original training set is reduced to simulate limited data. This leads to settings where validation examples may outnumber training examples. Table 1 shows such cases for the tasks of historical text normalization (Bollmann et al., 2018), morphological segmentation (Kann et al., 2018), morphological inflection (Makarov and Clematide, 2018; Sharma et al., 2018), argument component identification (Schulz et al., 2018), and transliteration (Upadhyay et al., 2018).



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3342–3349, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

However, in a real-world setting with limited resources, it is unlikely that such a development set would be available for early stopping, since it would be more effective to use at least part of it for training instead. Here, we investigate how previous results relate to those obtained in a setting that does *not* assume a development set. Instead of early stopping, we use data from the same task in other languages, the *development languages*, to decide on the number of training epochs. We are interested in two questions: Does recent work in low-resource NLP overestimate model performance by using an unrealistically precise performance signal to stop training? Or, inversely, is model performance underestimated by overfitting the finite development set?

Our experiments on historical text normalization, morphological inflection, and transliteration, featuring a variety of languages, show that performance does differ between runs with and without early stopping on the development set; if using the development set leads to better or worse results depends on the task and language. Differences of up to 18% absolute accuracy highlight that a realistic evaluation of models for low-resource NLP is crucial for estimating real-world performance.

## 2 Related Work

**Realistic evaluation of machine learning.** Oliver et al. (2018) investigate how to evaluate semi-supervised training algorithms in a realistic way; they differ from us in that they focus exclusively on semi-supervised learning (SSL) algorithms, and do not consider NLP explicitly. However, in line with our conclusion, they report that recent practices for evaluating SSL techniques do not address the question of the algorithms' real-word applicability in a satisfying way. In NLP, several earlier works have explicitly investigated real-world low-resource settings as opposed to artificial proxy settings, e.g., for part-of-speech tagging (Garrette et al., 2013) or machine translation (Irvine and Callison-Burch, 2013). While those mostly focus on real data-poor languages, we explicitly investigate the effect of the common practice to assume a relatively large development set for early stopping in the low-resource setting.

**Low-resource settings in NLP.** Research in the area of neural methods for low-resource NLP has gained popularity in recent years, with a dedicated workshop on the topic appearing in 2018 (Haffari et al., 2018). High-level key words under which other work on neural networks for data-poor scenarios in NLP can be found are domain adaptation (Daume III, 2007), multi-task learning (Caruana, 1997; Ruder, 2017), few-shot/zero-shot/one-shot learning (Johnson et al., 2017; Finn et al., 2017), transfer learning (Yarowsky et al., 2001), semi-supervised training (Zhu, 2005), or pretraining (Erhan et al., 2010).

While options for early stopping without a development set exist (Mahsereci et al., 2017), they require hyperparameter tuning, which might not be feasible without a development set, and, most importantly, they are not commonly used in low-resource NLP research. Here, we investigate if *current practices* might lead to unrealistic results.

## 3 Experimental Design

We compare early stopping using development set accuracy (*DevSet*) with an alternative strategy where the amount of training epochs is a hyperparameter tuned on development languages (*DevLang*). We perform two rounds of training:

- **Stopping point selection phase.** Models for the development languages are trained with the original early stopping strategy from previous work. The number of training epochs for the target languages is then calculated as the average over the best epochs for all development languages.[1] All development languages also function as target languages. To make this possible, for development languages, we compute the average over *other* development languages only.

- **Main training phase.** We train models for all languages keeping both the model resulting from the original early stopping strategy (DevSet) and that from the epoch computed in the stopping point selection phase (DevLang).[2]

The stopping point selection phase exclusively serves the purpose of tuning the number of epochs for the DevLang training setup. Models obtained

[1] We round this number to an integer. It is important for our experiments that the training sets for all languages are of the same size, cf. Table 1. Otherwise we would need to account for the number of training examples per epoch.

[2] This requires training for at least the number of target epochs, even if early stopping would end training earlier.

in this phase are discarded. The development sets we use in our experiments are those from the original papers without alterations.

Since both final models obtained in the main training phase result from the same training run, our experimental design enables a direct comparison between the models from both setups.

**Example.** Assume that, in the stopping point selection phase, we obtain the best development set results for a given task in development languages L1 and L2 after epochs 14 and 18, respectively. In the main training phase, we then train a model for the same task in target language L3 with the original early stopping strategy, but keeping additionally the model from epoch 16. If the best development result for language L3 is obtained after epoch 19, we compare the model from epoch 19 (DevSet) to that from epoch 16 (DevLang).

## 4 Tasks, Data, Models

For our study, we select previously published experiments which fulfill the following criteria: (1) datasets exist for at least four languages, and all training sets are of equal size; (2) the original authors use early stopping with a development set; (3) the authors explicitly investigate low-resource settings; and (4) the original code is publically available, or a standard model is used. Since our main goal is to confirm the effect of the development set and not to compare between tasks, we further limit this study to sequence-to-sequence tasks.

### 4.1 Historical Text Normalization (NORM)

**Task.** The goal of historical text normalization is to convert old texts into a form that conforms with contemporary spelling conventions. Historical text normalization is a specific case of the general task of text normalization, which additionally encompasses, e.g., correction of spelling mistakes or normalization of social media text.

**Data.** We experiment on the ten datasets from Bollmann et al. (2018), which represent eight different languages: German (two datasets; Bollmann et al., 2017; Odebrecht et al., 2017); English, Hungarian, Icelandic, and Swedish (Pettersson, 2016); Slovene (two datasets; Ljubešic et al., 2016); and Spanish and Portuguese (Vaamonde, 2015). We treat the two datasets for German and Slovene as different languages. All languages serve both as development languages for all *other* languages and as target languages.

**Model.** Our model for this task is an LSTM (Hochreiter and Schmidhuber, 1997) encoder-decoder model with attention (Bahdanau et al., 2015). Both encoder and decoder have a single hidden layer. We use the default model in OpenNMT (Klein et al., 2017)[3] as our implementation and employ the hyperparameters from Bollmann et al. (2018). In the original paper, *early stopping* is done by training for 50 epochs, and the best model regarding development accuracy is applied to the test set.

### 4.2 Morphological Inflection (MORPH)

**Task.** Morphological inflection consists of mapping the canonical form of a word, the lemma, to an indicated inflected form. This task gets very complex for morphologically rich languages, where a single lemma can have hundreds or thousand of inflected forms. Recently, morphological inflection has frequently been cast as a sequence-to-sequence task, mapping the characters of the input word together with the morphological features specifying the target to the characters of the corresponding inflected form (Cotterell et al., 2018).

**Data.** We experiment on the datasets released for a 2018 shared task (Cotterell et al., 2018), which cover 103 languages and feature an explicit low-resource setting. We randomly choose ten development languages: Armenian, Basque, Galician, Georgian, Greenlandic, Icelandic, Karbadian, Kannada, Latin, and Lithuanian.

**Model.** For MORPH, we experiment with a pointer-generator network architecture (Gu et al., 2016; See et al., 2017). This is a sequence-to-sequence model similar to that for NORM, but employs separate encoders for characters and features. It is further equipped with a copy mechanism: using attention to decide on what element from the input sequence to copy, the model computes a probability for either copying or generation while producing an output. The final probability distribution over the target vocabulary is a combination of both. Hyperparameters are taken from Sharma et al. (2018).[4] For *early stopping*, we also follow Sharma et al. (2018): all models are trained

[3] github.com/OpenNMT/OpenNMT-py

[4] github.com/abhishek0318/conll-sigmorphon-2018

for at least 300 epochs, and training is continued for another 100 epochs each time there has been improvement on the development set within the last 100 epochs.

### 4.3 Transliteration (TRANSL)

**Task.** Transliteration is the task of converting names from one script into another, while staying as close to the original pronunciation as possible. Unlike for *translation*, focus lies on the sound; the target language meaning is usually ignored.

**Data.** For our transliteration experiments, we follow Upadhyay et al. (2018). We experiment on datasets from the Named Entities Workshop 2015 (Duan et al., 2015) in Hindi, Kannada, Bengali, Tamil, and Hebrew. For this task, all languages are both development and target languages.

**Model.** The last featured model is an LSTM sequence-to-sequence model similar to that by Bahdanau et al. (2015), except for using hard monotonic attention (Aharoni and Goldberg, 2017). It attends to a *single* character at a time, and attention moves monotonically over the input. We take hyperparameters and code from Upadhyay et al. (2018).[5] *Early stopping* is done by training for 20 epochs and applying the best model regarding development accuracy to the test data.

### 4.4 Experimental Setup

We run all experiments using the implementations from previous work or OpenNMT as described above. Existing code is only modified where necessary. Most importantly, we add storing of the DevLang model during the main training phase.

## 5 Results

**Development sets vs. development languages.** We are asking if the use of a development set for early stopping leads to over- or underestimation of realistic model performance. Thus, we show in Table 2 how often we obtain higher accuracy for each of DevLang and DevSet. Additionally, averaged performance over all languages as well as the maximum difference in absolute accuracy are listed in Table 3. For NORM and TRANSL, results for individual languages are shown in Tables 4 and 5, respectively; for detailed results for MORPH see Appendix A. We see in Table 2 that, for MORPH and NORM, the use of unrealistically large development sets leads to better results than DevLang for 72 and 8 languages, respectively. For 8 and, respectively, 2 languages there is no difference, and a look at the detailed results in Appendix A and Table 4 reveals that, for those cases, we end up training for the same number of epochs for DevLang and DevSet. Only in 23 cases for MORPH, and none for NORM, we obtain *better* results for DevLang. This suggests that, for these two tasks, we frequently overestimate realistic model performance by early stopping on the development set. Indeed, Table 3 confirms this finding and shows that, on average across languages, DevSet models outperform DevLang models. The maximum difference is 18% absolute accuracy for the language Azeri and MORPH: for DevSet, we reach 64% accuracy after epoch 217, while, for DevLang, we only obtain 46% accuracy after the predefined epoch 324.

We obtain a different picture for TRANSL: results are equal for 3 languages, and better for DevLang for the remaining 2. Equal performance might be explained by the overall smaller number of training epochs in the original regime: stopping at the same epoch for both strategies is more likely. Overall, for TRANSL, performance on the development set seems to be less predictive for the final test performance than for the other tasks.

**Influence of the final epoch.** Since without a development set performance decreases on MORPH for most languages, we investigate if this can be explained by training often being too short. Therefore, we plot the difference of training du-

| | MORPH | NORM | TRANSL |
|---|---|---|---|
| **DevLang>DevSet** | 23 | 0 | 2 |
| **DevLang=DevSet** | 8 | 2 | 3 |
| **DevLang<DevSet** | 72 | 8 | 0 |

Table 2: Summary of cases in which using a development set leads to overestimation (DevSet>DevLang), underestimation (DevSet<DevLang), or neither (DevSet=DevLang) of the final model performance.

| | MORPH | NORM | TRANSL |
|---|---|---|---|
| **DevSet** | **51.3** | **74.9** | 21.8 |
| **DevLang** | 50.0 | 74.2 | **22.3** |
| $\Delta$ | -1.4 | -0.7 | +0.5 |
| **max** $\Delta$ | -18.0 | -2.4 | +1.3 |

Table 3: Test accuracy in % for different stopping approaches and tasks, averaged over languages.

[5] github.com/shyamupa/hma-translit

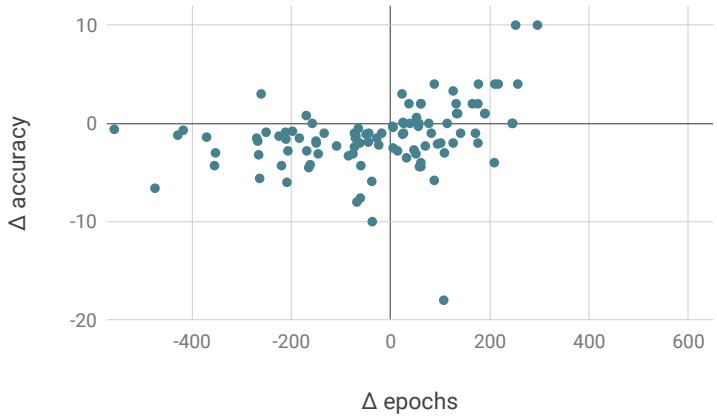


Figure 1: Difference in accuracy (DevLang-DevSet) depending on the difference in training epochs (DevLang-DevSet) for MORPH.

| Language | DevSet | DevLang |
|---|---|---|
| English | 0.7705 (26) | 0.7682 (43.44) |
| German (1) | 0.6749 (44) | 0.6749 (44.44) |
| German (2) | 0.7075 (43) | 0.6837 (41.78) |
| Hungarian | 0.4897 (44) | 0.4773 (41.67) |
| Icelandic | 0.7017 (27) | 0.6952 (43.56) |
| Portuguese | 0.7944 (43) | 0.7860 (42.44) |
| Slovene (1) | 0.8206 (38) | 0.8202 (41.67) |
| Slovene (2) | 0.8952 (46) | 0.8873 (42.00) |
| Spanish | 0.8352 (42) | 0.8352 (41.89) |
| Swedish | 0.7985 (43) | 0.7913 (42.11) |

Table 4: Detailed results per language for NORM; corresponding epochs in parenthesis.

| Language | DevSet | DevLang |
|---|---|---|
| Bengali | 0.374 (9) | 0.387 (10.50) |
| Hebrew | 0.135 (10) | 0.135 (10.00) |
| Hindi | 0.248 (10) | 0.248 (10.25) |
| Kannada | 0.216 (11) | 0.229 (10.00) |
| Tamil | 0.116 (10) | 0.116 (10.25) |

Table 5: Detailed results per language for TRANSL; corresponding epochs in parenthesis.

ration in epochs between DevLang and DevSet against the resulting difference in accuracy in Figure 1. While we indeed find that shorter training duration for a pointer-generator network for this task mostly results in worse performance, longer training can lead to either lower or higher accuracy. Thus, while longer training seems better for MORPH, not all performance loss can be explained by aborting training too early.

## 6 Discussion And Conclusion

**Limitations.** We investigate the effect of early stopping on the validation set as compared to a realistic setting without target language development examples. However, we would like to point out that, in certain situations, standard practices might be sufficient, e.g., for comparing different methods in equal settings, if absolute performance is not the main focus.

Further, we do not claim to show that using a validation set always *over-* or *under*estimates real-world performance, since this depends on how representative the validation set is of the target distribution. Our main result is that using a development set gives a poor estimate of real-world performance and that it is important to be aware of potential performance differences.

**Practical take-aways.** We replicate experiments from recent low-resource NLP research, once with the original experimental design and once without assuming development sets for early stopping. Since differences in absolute accuracy are up to 18.0%, we conclude that low-resource NLP research should move away from using large development sets for early stopping whenever real-world settings are being considered.

## Acknowledgments

We would like to thank Nikita Nangia and Jason Phang for their feedback on this work. This work has benefited from the support of Samsung Research under the project *Improving Deep Learning using Latent Structure* and from the donation of a Titan V GPU by NVIDIA Corporation.

## References


Roee Aharoni and Yoav Goldberg. 2017. Morphological inflection generation with hard monotonic attention. In *ACL*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Marcel Bollmann, Joachim Bingel, and Anders Søgaard. 2017. Learning attention for historical text normalization by learning to pronounce. In *ACL*.

Marcel Bollmann, Anders Søgaard, and Joachim Bingel. 2018. Multi-task learning for historical text normalization: Size matters. In *DeepLo*.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Arya D McCarthy, Katharina Kann, Sebastian Mielke, Garrett Nicolai, Miikka Silfverberg, David Yarowsky, Jason Eisner, and Mans Hulden. 2018. The CoNLL–SIGMORPHON 2018 shared task: Universal morphological reinflection. In *CoNLL–SIGMORPHON*.

Hal Daume III. 2007. Frustratingly easy domain adaptation. In *ACL*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Xiangyu Duan, Rafael E. Banchs, Min Zhang, Haizhou Li, and A Kumaran. 2015. Proceedings of the fifth named entity workshop. In *Fifth Named Entity Workshop*.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. Why does unsupervised pre-training help deep learning? *JMLR*, 11:625–660.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*.

Dan Garrette, Jason Mielens, and Jason Baldridge. 2013. Real-world semi-supervised learning of POS-taggers for low-resource languages. In *ACL*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *ACL*.

Reza Haffari, Colin Cherry, George Foster, Shahram Khadivi, and Bahar Salehi. 2018. Proceedings of the workshop on deep learning approaches for low-resource NLP.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Ann Irvine and Chris Callison-Burch. 2013. Combining bilingual and comparable corpora for low resource machine translation. In *WMT*.

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2017. Googles multilingual neural machine translation system: Enabling zero-shot translation. *TACL*, 5:339–351.

Katharina Kann, Ryan Cotterell, and Hinrich Schütze. 2017. One-shot neural cross-lingual transfer for paradigm completion. In *ACL*.

Katharina Kann, Jesus Manuel Mager Hois, Ivan Vladimir Meza Ruiz, and Hinrich Schütze. 2018. Fortification of neural morphological segmentation models for polysynthetic minimal-resource languages. In *NAACL*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *ACL*.

Nikola Ljubešic, Katja Zupan, Darja Fišer, and Tomaz Erjavec. 2016. Normalising slovene data: historical texts vs. user-generated content. In *KONVENS*.

Maren Mahsereci, Lukas Balles, Christoph Lassner, and Philipp Hennig. 2017. Early stopping without a validation set. *arXiv:1703.09580*.

Peter Makarov and Simon Clematide. 2018. Imitation learning for neural morphological string transduction. In *EMNLP*.

Carolin Odebrecht, Malte Belz, Amir Zeldes, Anke Lüdeling, and Thomas Krause. 2017. RIDGES herbology: designing a diachronic multi-layer corpus. *Language Resources and Evaluation*, 51(3):695–725.

Avital Oliver, Augustus Odena, Colin A Raffel, Ekin Dogus Cubuk, and Ian Goodfellow. 2018. Realistic evaluation of deep semi-supervised learning algorithms. In *NeurIPS*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*.

Eva Pettersson. 2016. *Spelling normalisation and linguistic analysis of historical text for information extraction*. Ph.D. thesis, Acta Universitatis Upsaliensis.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv:1706.05098*.

Claudia Schulz, Steffen Eger, Johannes Daxenberger, Tobias Kahse, and Iryna Gurevych. 2018. Multi-task learning for argumentation mining in low-resource settings. In *NAAACL*.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *ACL*.

Abhishek Sharma, Ganesh Katrapati, and Dipti Misra Sharma. 2018. IIT(BHU)–IIITH at CoNLL–SIGMORPHON 2018 shared task on universal morphological reinflection. In *CoNLL–SIGMORPHON*.

Shyam Upadhyay, Jordan Kodner, and Dan Roth. 2018. Bootstrapping transliteration with constrained discovery for low-resource languages. In *EMNLP*.

Gael Vaamonde. 2015. Userguide for digital edition of texts in ps post scriptum.

Alex Wang, Amapreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR*.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *HLT*.

Mo Yu, Xiaoxiao Guo, Jinfeng Yi, Shiyu Chang, Saloni Potdar, Yu Cheng, Gerald Tesauro, Haoyu Wang, and Bowen Zhou. 2018. Diverse few-shot text classification with multiple metrics. *arXiv:1805.07513*.

Xiaojin Jerry Zhu. 2005. Semi-supervised learning literature survey. Technical report, University of Wisconsin-Madison, Department of Computer Sciences.

# A Detailed Results for MORPH

| Language | DevSet | DevLang |
|---|---|---|
| adyghe | 89.2 (263) | 84.8 (324) |
| albanian | 27.9 (400) | 24.8 (324) |
| arabic | 34.1 (397) | 31.7 (324) |
| armenian | 52.2 (393) | 50.9 (323.1111111) |
| asturian | 71.2 (348) | 69 (324) |
| azeri | 64 (217) | 46 (324) |
| bashkir | 80.1 (474) | 78.1 (324) |
| basque | 7.9 (282) | 8.5 (334.2222222) |
| belarusian | 22.1 (198) | 25.4 (324) |
| bengali | 56 (585) | 59 (324) |
| breton | 56 (263) | 58 (324) |
| bulgarian | 50.3 (266) | 45.9 (324) |
| catalan | 64 (319) | 61.5 (324) |
| classical-syriac | 93 (285) | 93 (324) |
| cornish | 32 (267) | 32 (324) |
| crimean-tatar | 90 (458) | 89 (324) |
| czech | 37.2 (592) | 35.4 (324) |
| danish | 64.5 (409) | 61.2 (324) |
| dutch | 54.1 (362) | 48.2 (324) |
| english | 87.3 (292) | 83.8 (324) |
| estonian | 31 (536) | 30.1 (324) |
| faroese | 34 (310) | 31.2 (324) |
| finnish | 21.5 (799) | 14.9 (324) |
| french | 55.1 (881) | 54.5 (324) |
| friulian | 72 (247) | 72 (324) |
| galician | 44 (230) | 41.9 (324.4444444) |
| georgian | 77.2 (353) | 76.1 (303.5555556) |
| german | 52.3 (384) | 48 (324) |
| greek | 25.6 (300) | 24.5 (324) |
| greenlandic | 74 (167) | 76 (331.6666667) |
| haida | 58 (298) | 57 (324) |
| hebrew | 31.3 (268) | 31 (324) |
| hindi | 75 (594) | 73.5 (324) |
| hungarian | 39.5 (395) | 38 (324) |
| icelandic | 33.9 (250) | 31.6 (319.8888889) |
| ingrian | 50 (263) | 46 (324) |
| irish | 23.7 (493) | 20.9 (324) |
| italian | 45.6 (549) | 44.3 (324) |
| kabardian | 89 (233) | 86 (341.4444444) |
| kannada | 52 (93) | 56 (349.2222222) |
| karelian | 88 (147) | 92 (324) |
| kashubian | 52 (263) | 54 (324) |
| kazakh | 82 (287) | 84 (324) |
| khakas | 82 (148) | 80 (324) |
| khaling | 23.1 (742) | 22.4 (324) |
| kurmanji | 81.9 (470) | 78.8 (324) |
| ladin | 66 (236) | 70 (324) |
| latin | 14.4 (478) | 10.2 (315.8888889) |
| latvian | 35.3 (397) | 34.3 (324) |
| lithuanian | 15.8 (447) | 14 (296.5555556) |
| livonian | 31 (134) | 32 (324) |

Table 6: Detailed results per language for MORPH; corresponding epochs in parenthesis; part 1.

| Language | DevSet | DevLang |
|---|---|---|
| lower-sorbian | 42 (590) | 38.8 (324) |
| macedonian | 56.8 (508) | 55.3 (324) |
| maltese | 26 (189) | 27 (324) |
| mapudungun | 86 (107) | 90 (324) |
| middle-french | 82.9 (236) | 77.1 (324) |
| middle-high-german | 80 (223) | 78 (324) |
| middle-low-german | 36 (192) | 38 (324) |
| murrinhpatha | 38 (72) | 48 (324) |
| navajo | 10.8 (368) | 9.8 (324) |
| neapolitan | 77 (301) | 80 (324) |
| norman | 60 (148) | 62 (324) |
| northern-sami | 18.7 (695) | 17.3 (324) |
| north-frisian | 43 (386) | 41 (324) |
| norwegian-bokmaal | 66.3 (273) | 63.2 (324) |
| norwegian-nynorsk | 51.6 (385) | 44 (324) |
| occitan | 69 (183) | 68 (324) |
| old-armenian | 31.6 (389) | 31.1 (324) |
| old-church-slavonic | 49 (134) | 50 (324) |
| old-english | 22.7 (433) | 20.4 (324) |
| old-french | 40.6 (352) | 39.1 (324) |
| old-irish | 2 (78) | 2 (324) |
| old-saxon | 25 (494) | 25.8 (324) |
| pashto | 36 (114) | 40 (324) |
| persian | 51.2 (277) | 48.5 (324) |
| polish | 31.3 (544) | 27 (324) |
| portuguese | 57.3 (319) | 56.9 (324) |
| quechua | 58.7 (753) | 57.5 (324) |
| romanian | 35.2 (531) | 32.4 (324) |
| russian | 43.6 (320) | 43.3 (324) |
| sanskrit | 50.7 (489) | 46.2 (324) |
| scottish-gaelic | 76 (361) | 66 (324) |
| serbo-croatian | 34.4 (533) | 28.4 (324) |
| slovak | 41 (575) | 40.1 (324) |
| slovene | 47.4 (535) | 45.8 (324) |
| sorani | 27.3 (677) | 24.3 (324) |
| spanish | 54.3 (588) | 48.7 (324) |
| swahili | 62 (298) | 62 (324) |
| swedish | 63.7 (392) | 55.7 (324) |
| tatar | 76 (191) | 77 (324) |
| telugu | 98 (79) | 98 (324) |
| tibetan | 36 (28) | 46 (324) |
| turkish | 36.6 (299) | 36.7 (324) |
| turkmen | 82 (115) | 78 (324) |
| ukrainian | 33.8 (522) | 33 (324) |
| urdu | 66.6 (369) | 64.7 (324) |
| uzbek | 93 (242) | 92 (324) |
| venetian | 76.8 (210) | 76.8 (324) |
| votic | 22 (342) | 21 (324) |
| welsh | 48 (482) | 48 (324) |
| west-frisian | 50 (153) | 49 (324) |
| yiddish | 63 (198) | 61 (324) |
| zulu | 32.3 (679) | 28 (324) |

Table 7: Detailed results per language for MORPH; corresponding epochs in parenthesis; part 2.

# Synchronously Generating Two Languages with Interactive Decoding

**Yining Wang**[1,2], **Jiajun Zhang**[1,2]*, **Long Zhou**[1,2]
**Yuchen Liu**[1,2] and **Chengqing Zong**[1,2,3]
[1]National Laboratory of Pattern Recognition, CASIA, Beijing, China
[2]University of Chinese Academy of Sciences, Beijing, China
[3]CAS Center for Excellence in Brain Science and Intelligence Technology, Beijing, China
{yining.wang, jjzhang, long.zhou}@nlpr.ia.ac.cn
{yuchen.liu, cqzong}@nlpr.ia.ac.cn

## Abstract

In this paper, we introduce a novel interactive approach to translate a source language into two different languages simultaneously and interactively. Specifically, the generation of one language relies on not only previously generated outputs by itself, but also the outputs predicted in the other language. Experimental results on IWSLT and WMT datasets demonstrate that our method can obtain significant improvements over both conventional Neural Machine Translation (NMT) model and multilingual NMT model.

## 1 Introduction

Neural Machine Translation (NMT) based on the encoder-decoder framework has significantly improved translation quality due to its powerful end-to-end modeling (Sutskever et al., 2014; Bahdanau et al., 2015; Vaswani et al., 2017; Gehring et al., 2017; Hassan et al., 2018; Zhang and Zong, 2015). This paradigm facilitates the development of multilingual NMT (Dong et al., 2015; Luong et al., 2016; Johnson et al., 2017; Ha et al., 2016; Firat et al., 2016; Lakew et al., 2017; Tan et al., 2019; Wang et al., 2019), which handles multiple language pairs in one model, with the benefit of simplifying offline model training and easing online maintenance cost.

Although multilingual NMT attempts to utilize the complementary information of different languages (Lu et al., 2018; Neubig and Hu, 2018; Platanios et al., 2018; Wang et al., 2018), all of the models handle one language pair at each moment for both training and testing. However, we find that the generation process of different target languages can help each other. For example in Figure 1, when decoding the Chinese word "书" meaning "*book*" at step $t = 5$, the predicted Japanese word "本" with same meaning can provide the context at step $t = 4$. The reason is that the sentence structure between the two languages is different. It is *subject-verb-object* in Chinese while it is *subject-object-verb* in Japanese. Moreover, we find that two languages are complementary, and if decoders belonging to two different languages can interact with each other, the quality of translation will be improved.

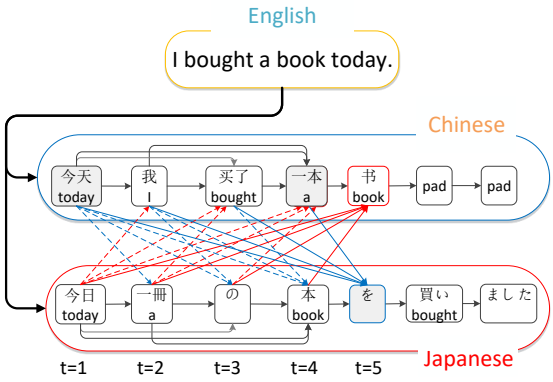


Figure 1: An example of an English sentence translated into Chinese and Japaneses sentences, in which two targets can interact with each other.

In this work, we present a novel interactive decoding algorithm to generate two target languages simultaneously and interactively. To this end, we propose a synchronous attention model, in which the generation of one language can attend to already generated outputs of another language. As shown in Figure 1, the two decoders predict their outputs at the same time. At each moment, word prediction of each language does not only rely on previously generated targets itself but also depends on outputs of the other language.

We conduct extensive experiments to verify the effectiveness of our proposed approaches on English-to-German/French and English-to-Chinese/Japanese translation tasks.

*Jiajun Zhang is the corresponding author.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3350–3355, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

Our contributions in this work are two-fold:

(1) We propose a novel synchronous translation model that can predict outputs of two different languages simultaneously and interactively, which can enhance the translation quality of both languages.

(2) Extensive experiments show the superiority of our proposed method. Specifically, this synchronous approach can significantly outperform both the conventional NMT model and the multilingual NMT model.

## 2 Background

Owing to powerful modeling ability, our synchronous method relies on Transformer architecture (Vaswani et al., 2017), which is entirely based on the attention mechanism detailed below.

**Scaled Dot-Product Attention:** The inputs of attention mechanism contain queries $Q$, keys $K$, and values $V$. This function can be described as mapping a query and a set of key-value pairs to an output, and the output is calculated as a weighted sum of the values.

$$\text{Attention}\,(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \quad (1)$$

where $d_k$ is dimension of keys, $Q$, $K$, $V$ are obtained by linearly transforming input hidden states with projection matrices.

## 3 Synchronous Translation Method

As discussed in Sec. 1, outputs in different languages can be complementary and can help with each other. Thus, it is reasonable to improve translation quality with interactions of two decoders. In this section, we propose an interactive decoding algorithm and then describe how to implement it by a new attention mechanism, named as synchronous attention model.

### 3.1 Interactive Decoding Algorithm

Interactive decoding algorithm can generate translations of different languages in the same beam. At each step, each half of the beam produces translations in one target language conditioning on source sentence and the predicted tokens in both target languages. Here, we use two decoders with separated embeddings and softmax layers to operate two languages. The two decoders predict each token in parallel and keep interaction with each other, which can be formalized as follows:

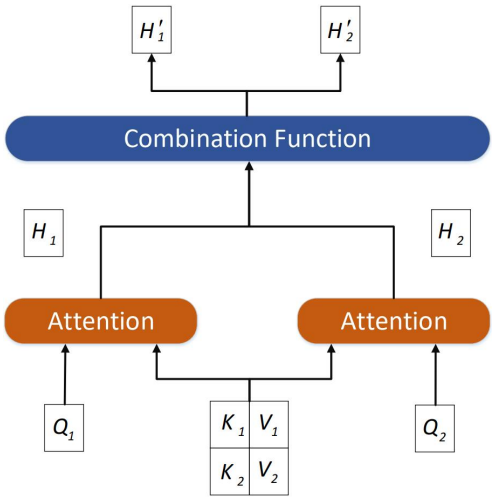


Figure 2: The synchronous self-attention framework, which simultaneously operates on the keys, queries and values of two different decoders.

$$\log P(\boldsymbol{y}^1|\boldsymbol{x}) = \log \prod_{i=0}^{n-1} p(y_i^1|\boldsymbol{x}, y_0^1, \ldots, y_{i-1}^1, y_0^2, \ldots, y_{i-1}^2)$$
$$\log P(\boldsymbol{y}^2|\boldsymbol{x}) = \log \prod_{i=0}^{n-1} p(y_i^2|\boldsymbol{x}, y_0^2, \ldots, y_{i-1}^2, y_0^1, \ldots, y_{i-1}^1) \quad (2)$$

where $\boldsymbol{x}$ is source sentence, $\boldsymbol{y}^1, \boldsymbol{y}^2$ are target sentences corresponding to two different languages. At time-step $i$, we have generated the first $i-1$ tokens of language-1 $\boldsymbol{y}^1$ and the first $i-1$ tokens of language-2 $\boldsymbol{y}^2$. Then both languages predictions can be utilized together with source sentence to generate tokens $y_i^1$ and $y_i^2$. This interaction between two languages is realized by synchronous attention model, which will be detailed in the following subsection.

It should be noted that the two language sentences can be generated in different directions (Liu et al., 2016; Zhang et al., 2018; Zhou et al., 2019), which means language-1 can be produced in left-to-right (L2R) manner while language-2 in right-to-left (R2L) manner. We will analyze the effect of different decoding manners in Sec. 5.2.

### 3.2 Synchronous Attention Model

Synchronous attention model (SyncAtt) is shown in Figure 2, in which inputs of two decoders contain queries $(Q_1, Q_2)$, keys $(K_1, K_2)$, and values $(V_1, V_2)$ separately. The new hidden states $(H_i')$ can be computed by our proposed synchronous atten-

tion as follows:

$$\begin{aligned} H_1^{'} &= \text{SyncAtt}\,(Q_1, [K_1; K_2], [V_1; V_2]) \\ H_2^{'} &= \text{SyncAtt}\,(Q_2, [K_1; K_2], [V_1; V_2]) \end{aligned} \tag{3}$$

where synchronous attention model (SyncAtt) can be described in detail:

$$\begin{aligned} H_1 &= \text{Attention}\,(Q_1, K_1, V_1) \\ \widetilde{H}_1 &= \text{Attention}\,(Q_1, K_2, V_2) \\ H_2 &= \text{Attention}\,(Q_2, K_2, V_2) \\ \widetilde{H}_2 &= \text{Attention}\,(Q_2, K_1, V_1) \\ H_1^{'} &= f(H_1; \widetilde{H}_1) = H_1 + \lambda \times \tanh(\widetilde{H}_1) \\ H_2^{'} &= f(H_2; \widetilde{H}_2) = H_2 + \lambda \times \tanh(\widetilde{H}_2) \end{aligned} \tag{4}$$

where $\lambda$ is a balance weight of hidden states between two decoders decided by development set.

We apply our synchronous attention model to replace self-attention sub-layer in Transformer decoder, and it also utilizes the residual connections (He et al., 2016) around each sub-layer, followed by layer normalization (Ba et al., 2016).

### 3.3 Training

Since our synchronous method decodes two languages at the same time, the different decoders can be optimized simultaneously.

Supposing we have the trilingual datasets $\boldsymbol{D} = \{(\boldsymbol{x}, \boldsymbol{y}^1, \boldsymbol{y}^2)\}$, the objective function is to maximize the log-likelihood over the two target sentences:

$$L(\boldsymbol{\theta}) = \sum_{(\boldsymbol{x},\boldsymbol{y}^1,\boldsymbol{y}^2)\in D} \Big(\sum_{i=1}^{|y_i^1|} \log P(y_i^1|\boldsymbol{x}) + \sum_{i=1}^{|y_i^2|} \log P(y_i^2|\boldsymbol{x})\Big) \tag{5}$$

When calculating $P(y_i^1|\boldsymbol{x})$, except for the context from source side $\boldsymbol{x}$, our synchronous method employs not only previous reference $y_{<i}^1$ as condition, but the previous context of the other decoder reference $y_{<i}^2$. The calculation process of $P(y_i^2|\boldsymbol{x})$ is similar.

However, the practical situation is that the triple data is limited and hard to be collected. In this work, we construct the trilingual training corpus by data augmenting method (Sennrich et al., 2016a; Zhang and Zong, 2016). To achieve this, we first learn two independent translation models **Model-1** and **Model-2** on the bilingual training data $(\boldsymbol{x}^1, \boldsymbol{y}^1)$ and $(\boldsymbol{x}^2, \boldsymbol{y}^2)$ separately. Then, **Model-1** and **Model-2** are employed to decode the input sentences $\boldsymbol{x}^2$ and $\boldsymbol{x}^1$, resulting in pseudo training data $(\boldsymbol{x}^2, \boldsymbol{y}^{1\star})$ and $(\boldsymbol{x}^1, \boldsymbol{y}^{2\star})$, respectively. Thus, we can obtain the triple parallel training data $\boldsymbol{D} = \{(\boldsymbol{x}^1, \boldsymbol{y}^1, \boldsymbol{y}^{2\star})\} \cup \{(\boldsymbol{x}^2, \boldsymbol{y}^{1\star}, \boldsymbol{y}^2)\}$, which can be used to train our synchronous translation model mentioned above.

## 4 Experimental Settings

### 4.1 Data

We evaluate our proposed synchronous method on two translation tasks, which include English→Chinese/Japanese (briefly, En→Zh/Ja) and English→German/French (briefly, En→De/Fr) on IWSLT[1] datasets. The *IWSLT.TED.tst2013* and *IWSLT.TED.tst2014* are employed as devlopment set and test set respectively. Besides, we also perform En→De/Fr translation in large scale WMT14[2] datasets. We use *newstest2014* as test set.

**En→Zh/Ja:** For this translation task, the training sets of En→Zh and En→Ja consist of 231K, 223K sentence pairs. We tokenize the English sentences using a script from Moses (Koehn et al., 2007), and we segment Chinese and Japanese data by *jieba*[3] and *mecab*[4]. We use BPE method (Sennrich et al., 2016b) to encode the source side sentences and the combination of target side sentences respectively and limit the vocabularies of both sides to the most frequent 10k tokens.

**En→De/Fr:** We conduct this translation task on two different settings. One setting is using training set of IWSLT datasets which contains 206K sentence pairs for En→De and 233K sentence pairs for En→Fr. We follow the common practice to tokenize and lowercase all words. Sentences are encoded using BPE, which has a shared vocabulary of 10K tokens. At last, we construct pseudo triple data by the method described in Sec. 3.3. For the other setting, we extract the trilingual subset in WMT14 inspired by Zoph and Knight (2016), which includes about 2.43M sentence triples. We use 37K shared BPE tokens as vocabulary.

### 4.2 Training Details

We implement our synchronous translation based on the tensor2tensor[5] library. We train our models using the configuration *transformer_base* adopted

[1] https://wit3.fbk.eu
[2] http://www.statmt.org/wmt14/translation-task.html
[3] https://github.com/fxsjy/jieba
[4] http://taku910.github.io/mecab
[5] https://github.com/tensorflow/tensor2tensor

| Hyper-paramter ($\lambda$) | En-De/Fr | | En-Zh/Ja | |
|---|---|---|---|---|
| | En-De | En-Fr | En-Zh | En-Ja |
| 0.1 | **30.95** | **43.01** | **16.33** | **18.88** |
| 0.2 | 30.77 | 42.99 | 16.06 | 18.82 |
| 0.3 | 30.55 | 42.99 | 15.96 | 18.38 |
| 0.4 | 29.60 | 42.52 | 15.66 | 18.03 |
| 0.5 | 29.19 | 41.87 | 15.17 | 17.42 |

Table 1: Experiment results on the development set with different $\lambda$s.

| Method | En-Zh/Ja | | En-De/Fr | |
|---|---|---|---|---|
| | En-Zh | En-Ja | En-De | En-Fr |
| *Indiv* | 15.68 | 16.56 | 27.11 | 40.62 |
| *Indiv + pseudo* | 16.72 | 18.02 | 28.47 | 40.39 |
| *Multi* | 17.06 | 18.31 | 27.79 | 40.97 |
| *Multi + pseudo* | 17.10 | 18.40 | 28.56 | 40.62 |
| *SyncTrans* | **17.97** | **19.31** | **29.16** | **41.53** |

Table 2: Translation performance on IWSLT datasets. *SyncTrans* represents our proposed synchronous translation method. All results of our *SyncTrans* are significantly better than both *Indiv* and *Multi* ($p < 0.01$).

by Vaswani et al. (2017), which contains a 6-layer encoder and a 6-layer decoder with 512-dimensional hidden representations. During training, each mini-batch contains roughly 4,096 tokens for both source and target sides. We use Adam optimizer (Kingma and Ba, 2014) with $\beta_1$=0.9, $\beta_2$=0.98, and $\epsilon$=$10^{-9}$. For decoding, we set beam size to be $k = 4$ and length penalty $\alpha = 0.6$. All our methods are trained and tested on single Nvidia P40 GPU.

We investigate the impact of different $\lambda$s in our synchronous attention model. As shown in Table 1, when $\lambda$=0.1, the translation results perform best on development set for both En→Zh/Ja and En→De/Fr tasks, and we will use this setting in the subsequent experiments.

## 5 Results and Analysis

The translation performance of IWSLT datasets is evaluated by case-insensitive BLEU4 (Papineni et al., 2002) for En→De/Fr task and character-level BLEU5 for En→Zh/Ja task. For WMT14 datasets, we calculate the case-sensitive BLEU4 the same as previous work. In our experiments, the NMT models trained on individual language pair are denoted by *Indiv*.

### 5.1 Results on IWSLT

Table 2 shows the main translation results of En→Zh/Ja and En→De/Fr on IWSLT datasets. We also conduct a typical one-to-many translation adopting Johnson et al. (2017) method on Transformer as our another baseline model, referred to *Multi*. Compared with *Indiv*, we can see that *Multi* achieves better results on all cases, which can be attributed to that the encoder can be enhanced by extra training data from the other language pair.

As for our proposed method, the synchronous translation method performs significantly better than both *Indiv* and *Multi* baseline methods, and it can achieve the improvements up to 2.75 BLEU points (19.31 vs. 16.56) on En→Ja.

To perform synchronous translation, the triple parallel corpus contains pseudo training data we construct. For a fair comparison, we also conduct our baseline methods on the training sentences augmented by the pseudo corpus. From row 2 and row 4 in Table 2, our method achieves better performance than both *Multi + pseudo* method and *Indiv + pseudo* method with gains of 0.82 BLEU and 1.09 points on average, which demonstrates the effectiveness of our method.

### 5.2 L2R or R2L Manner

As described in Sec. 3.1, two target languages can be generated in L2R or R2L manner, which can provide the future contexts for each other. We further conduct an experiment to investigate different decoding manners in this work.

Figure 3 reports the results. We observe that when performing En→De/Fr translation, one language generated from R2L manner is helpful for the other language but do harm to itself. However, for En→Zh/Ja translation, Japanese can achieve improvements on both L2R and R2L decoding settings. The reason is that Japanese is suited for translating from right-to-left manner, and it can take advantage of future outputs from Chinese.

### 5.3 Results on WMT

We also employ our method on the real triple training datasets, which can be collected from WMT14 En→De/Fr datasets as described in Sec. 4.1. From Table 3, we observe that our method consistently outperforms baseline models. Note that in contrast to results on IWSLT datasets, *Multi* can not perform on par with *Indiv*, because the source side data for two language pairs are the same, and encoder network can not be enhanced as *Multi* method in Sec. 5.1.

Moreover, we construct a large scale pseudo

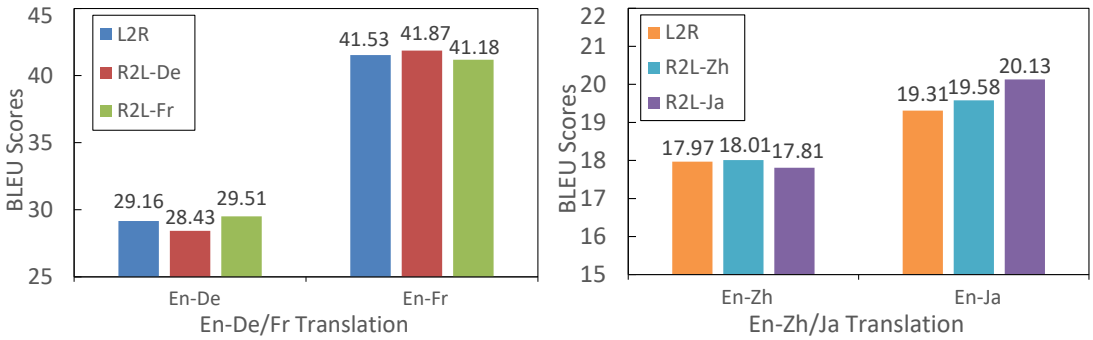


Figure 3: The comparison of L2R and R2L decoding manners, in which one of target languages generated from R2L manner.

| Method | WMT14 (2.43M) | | WMT14 (4.50M) |
|---|---|---|---|
| | En-De | En-Fr | En-De |
| *Indiv* | 24.33 | 37.12 | 26.53 |
| *Multi* | 23.46 | 36.33 | 25.81 |
| *SyncTrans* | **24.84**$^{\dagger *}$ | **37.66**$^{\dagger *}$ | **27.01**$^{\dagger *}$ |

Table 3: Translation quality of En-De/Fr on WMT14 datasets. The significance values with respect to the baseline method *Indiv* and *Multi* method are denoted by "$*$" and "†" respectively, indicating our proposed *SyncTrans* significantly better than both *Indiv* ($p < 0.05$) and *Multi* ($p < 0.01$) methods.

triple data about 4.5M[6]. The result is demonstrated in the last column of Table 3, in which our synchronous method performs better than the baseline methods as well.

## 6 Conclusion

In this paper, we propose an interactive decoding algorithm to generate two target languages simultaneously and interactively. The empirical experiments on four language pairs demonstrate that our approach can obtain significant improvements over both the NMT model trained on individual language pair and multilingual NMT model. For the future work, we plan to extend our method on more than two target languages and explore other effective interactive approaches to improve the translation quality further.

## Acknowledgments

The research work described in this paper has been supported by the National Key Research and Development Program of China under Grant No. 2016QY02D0303, the Natural Science Foundation of China under Grant No. U1836221 and the Beijing Municipal Science and Technology Project under Grant No. Z181100008918017. The research work in this paper has also been supported by Beijing Advanced Innovation Center for Language Resources. We thank the three anonymous reviewers for their efforts on this manuscript. We would like to thank Yanfen Zhang and Junnan Zhu for their invaluable discussions on this paper.

[6]The En-Fr part is entirely created by translating English of WMT14 En-De datasets into French using the model trained on WMT14 En-Fr corpus about 37M.

## References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *In Proceedings of ICLR 2015*.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. *In Proceedings of ACL 2015*, pages 1723–1732.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. *In Proceedings of NAACL 2016*, pages 866–875.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1601.03317*.

Thanh-Le Ha, Jan Niehues, and Alexander Waibel. 2016. Toward multilingual neural machine translation with universal encoder and decoder. *In Proceedings of IWSLT 2016*.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, et al. 2018. Achieving human parity on automatic Chinese to English news translation. *arXiv preprint arXiv:1803.05567*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. *In proceddings of CVPR 2016*.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. *In Proceedings of ACL 2007*.

Surafel M Lakew, ADG Mattia, and F Marcello. 2017. Multilingual neural machine translation for low resource languages. *CLiC-it*.

Lemao Liu, Masao Utiyama, Andrew Finch, and Eiichiro Sumita. 2016. Agreement on target-bidirectional neural machine translation. *In Proceedings of NAACL 2016*, pages 411–416.

Yichao Lu, Phillip Keung, Faisal Ladhak, Vikas Bhardwaj, Shaonan Zhang, and Jason Sun. 2018. A neural interlingua for multilingual machine translation. *arXiv preprint arXiv:1804.08198*.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. *In Proceedings of ICLR 2016*.

Graham Neubig and Junjie Hu. 2018. Rapid adaptation of neural machine translation to new languages. *In Proceedings of EMNLP 2018*, pages 875–880.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. *In Proceedings of ACL*, pages 311–318.

Emmanouil Antonios Platanios, Mrinmaya Sachan, Graham Neubig, and Tom Mitchell. 2018. Contextual parameter generation for universal neural machine translation. *In Proceedings of EMNLP 2018*, pages 425–435.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. *In Proceedings of ACL 2016*, pages 86–96.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. *In Proceedings of ACL 2016*, pages 1715–1725.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *In Proceedings of NIPS*, pages 3104–3112.

Xu Tan, Yi Ren, Di He, Tao Qin, Zhou Zhao, and Tie-Yan Liu. 2019. Multilingual neural machine translation with knowledge distillation. *In Proceedings of ICLR 2019*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, and Łukasz Kaiser. 2017. Attention is all you need. *In Proceedings of NIPS*, pages 30–34.

Yining Wang, Jiajun Zhang, Feifei Zhai, Jingfang Xu, and Chengqing Zong. 2018. Three strategies to improve one-to-many multilingual translation. *In Proceedings of EMNLP 2018*, pages 2955–2960.

Yining Wang, Long Zhou, Jiajun Zhang, Feifei Zhai, Jingfang Xu, and Chengqing Zong. 2019. A compact and language-sensitive multilingual translation method. *In Proceedings of ACL 2019*, pages 1213–1223.

Jiajun Zhang and Chengqing Zong. 2015. Deep neural networks in machine translation: An overview. *IEEE Intelligent Systems*, 30(5):16–25.

Jiajun Zhang and Chengqing Zong. 2016. Exploiting source-side monolingual data in neural machine translation. *In Proceedings of EMNLP*, pages 1535–1545.

Xiangwen Zhang, Jinsong Su, Yue Qin, Yang Liu, Rongrong Ji, and Hongji Wang. 2018. Asynchronous bidirectional decoding for neural machine translation. *In proceddings of AAAI 2018*.

Long Zhou, Jiajun Zhang, and Chengqing Zong. 2019. Synchronous bidirectional neural machine translation. *Transactions of the Association for Computational Linguistics*, 7:91–105.

Barret Zoph and Kevin Knight. 2016. Multi-source neural translation. *In Proceedings of NAACL 2016*, pages 30–34.

# On NMT Search Errors and Model Errors: Cat Got Your Tongue?

**Felix Stahlberg*** and **Bill Byrne**
University of Cambridge
Department of Engineering
Trumpington St, Cambridge CB2 1PZ, UK
{fs439,wjb31}@cam.ac.uk

## Abstract

We report on search errors and model errors in neural machine translation (NMT). We present an exact inference procedure for neural sequence models based on a combination of beam search and depth-first search. We use our exact search to find the global best model scores under a Transformer base model for the entire WMT15 English-German test set. Surprisingly, beam search fails to find these global best model scores in most cases, even with a very large beam size of 100. For more than 50% of the sentences, the model in fact assigns its global best score to the empty translation, revealing a massive failure of neural models in properly accounting for adequacy. We show by constraining search with a minimum translation length that at the root of the problem of empty translations lies an inherent bias towards shorter translations. We conclude that vanilla NMT in its current form requires just the right amount of beam search errors, which, from a modelling perspective, is a highly unsatisfactory conclusion indeed, as the model often prefers an empty translation.

## 1 Introduction

Neural machine translation (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Bahdanau et al., 2015, NMT) assigns the probability $P(\mathbf{y}|\mathbf{x})$ of a translation $\mathbf{y} = y_1^J \in \mathcal{T}^J$ of length $J$ over the target language vocabulary $\mathcal{T}$ for a source sentence $\mathbf{x} \in \mathcal{S}^I$ of length $I$ over the source language vocabulary $\mathcal{S}$ via a left-to-right factorization using the chain rule:

$$\log P(\mathbf{y}|\mathbf{x}) = \sum_{j=1}^{J} \log P(y_j|y_1^{j-1}, \mathbf{x}). \quad (1)$$

The task of finding the most likely translation $\hat{\mathbf{y}} \in \mathcal{T}^*$ for a given source sentence $\mathbf{x}$ is known as the *decoding* or *inference* problem:

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y} \in \mathcal{T}^*} P(\mathbf{y}|\mathbf{x}). \quad (2)$$

The NMT search space is vast as it grows exponentially with the sequence length. For example, for a common vocabulary size of $|\mathcal{T}| = 32,000$, there are already more possible translations with 20 words or less than atoms in the observable universe ($32,000^{20} \gg 10^{82}$). Thus, complete enumeration of the search space is impossible. The size of the NMT search space is perhaps the main reason why – besides some preliminary studies (Niehues et al., 2017; Stahlberg et al., 2018b; Ott et al., 2018) – analyzing search errors in NMT has received only limited attention. To the best of our knowledge, none of the previous studies were able to quantify the number of search errors in unconstrained NMT due to the lack of an exact inference scheme that – although too slow for practical MT – guarantees to find the global best model score for analysis purposes.

In this work we propose such an exact decoding algorithm for NMT that exploits the monotonicity of NMT scores: Since the conditional log-probabilities in Eq. 1 are always negative, partial hypotheses can be safely discarded once their score drops below the log-probability of any *complete* hypothesis. Using our exact inference scheme we show that beam search does not find the global best model score for more than half of the sentences. However, these *search* errors, paradoxically, often prevent the decoder from suffering from a frequent but very serious *model* error in NMT, namely that the empty hypothesis often gets the global best model score. Our findings suggest a reassessment of the amount of model and search errors in NMT, and we hope that they will spark new efforts in improving NMT modeling capabilities, especially in terms of adequacy.

*Now at Google.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3356–3362, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

**Algorithm 1** BeamSearch($\mathbf{x}, n \in \mathbb{N}_+$)

**Input:** $\mathbf{x}$: Source sentence, $n$: Beam size

1: $\mathcal{H}_{cur} \leftarrow \{(\epsilon, 0.0)\}$ {Initialize with empty translation prefix and zero score}
2: **repeat**
3: $\quad\mathcal{H}_{next} \leftarrow \emptyset$
4: $\quad$**for all** $(\mathbf{y}, p) \in \mathcal{H}_{cur}$ **do**
5: $\quad\quad$**if** $y_{|\mathbf{y}|} = < /\mathtt{s} >$ **then**
6: $\quad\quad\quad\mathcal{H}_{next} \leftarrow \mathcal{H}_{next} \cup \{(\mathbf{y}, p)\}$ {Hypotheses ending with $< /\mathtt{s} >$ are not expanded}
7: $\quad\quad$**else**
8: $\quad\quad\quad\mathcal{H}_{next} \leftarrow \mathcal{H}_{next} \cup \bigcup_{w \in \mathcal{T}} (\mathbf{y} \cdot w, p + \log P(w|\mathbf{x}, \mathbf{y}))$ {Add all possible continuations}
9: $\quad\quad$**end if**
10: $\quad$**end for**
11: $\quad\mathcal{H}_{cur} \leftarrow \{(\mathbf{y}, p) \in \mathcal{H}_{next} : |\{(\mathbf{y}', p') \in \mathcal{H}_{next} : p' > p\}| < n\}$ {Select $n$-best}
12: $\quad(\tilde{\mathbf{y}}, \tilde{p}) \leftarrow \arg\max_{(\mathbf{y}, p) \in \mathcal{H}_{cur}} p$
13: **until** $\tilde{y}_{|\tilde{\mathbf{y}}|} = < /\mathtt{s} >$
14: **return** $\tilde{\mathbf{y}}$

**Algorithm 2** DFS($\mathbf{x}, \mathbf{y}, p \in \mathbb{R}, \gamma \in \mathbb{R}$)

**Input:** $\mathbf{x}$: Source sentence
$\mathbf{y}$: Translation prefix (default: $\epsilon$)
$p$: $\log P(\mathbf{y}|\mathbf{x})$ (default: 0.0)
$\gamma$: Lower bound

1: **if** $y_{|\mathbf{y}|} = < /\mathtt{s} >$ **then**
2: $\quad$**return** $(\mathbf{y}, p)$ {Trigger $\gamma$ update}
3: **end if**
4: $\tilde{\mathbf{y}} \leftarrow \perp$ {Initialize $\tilde{\mathbf{y}}$ with dummy value}
5: **for all** $w \in \mathcal{T}$ **do**
6: $\quad p' \leftarrow p + \log P(w|\mathbf{x}, \mathbf{y})$
7: $\quad$**if** $p' \geq \gamma$ **then**
8: $\quad\quad(\mathbf{y}', \gamma') \leftarrow \text{DFS}(\mathbf{x}, \mathbf{y} \cdot w, p', \gamma)$
9: $\quad\quad$**if** $\gamma' > \gamma$ **then**
10: $\quad\quad\quad(\tilde{\mathbf{y}}, \gamma) \leftarrow (\mathbf{y}', \gamma')$
11: $\quad\quad$**end if**
12: $\quad$**end if**
13: **end for**
14: **return** $(\tilde{\mathbf{y}}, \gamma)$

## 2 Exact Inference for Neural Models

Decoding in NMT (Eq. 2) is usually tackled with beam search, which is a time-synchronous approximate search algorithm that builds up hypotheses from left to right. A formal algorithm description is given in Alg. 1. Beam search maintains a set of active hypotheses $\mathcal{H}_{cur}$. In each iteration, all hypotheses in $\mathcal{H}_{cur}$ that do not end with the end-of-sentence symbol $< /\mathtt{s} >$ are expanded and collected in $\mathcal{H}_{next}$. The best $n$ items in $\mathcal{H}_{next}$ constitute the set of active hypotheses $\mathcal{H}_{cur}$ in the next iteration (line 11 in Alg. 1), where $n$ is the beam size. The algorithm terminates when the best hypothesis in $\mathcal{H}_{cur}$ ends with the end-of-sentence symbol $< /\mathtt{s} >$. Hypotheses are called *complete* if they end with $< /\mathtt{s} >$ and *partial* if they do not.

Beam search is the ubiquitous decoding algorithm for NMT, but it is prone to search errors as the number of active hypotheses is limited by $n$. In particular, beam search never compares partial hypotheses of different lengths with each other. As we will see in later sections, this is one of the main sources of search errors. However, in many cases, the model score found by beam search is a reasonable approximation to the global best model score. Let $\gamma$ be the model score found by beam search ($\tilde{p}$ in line 12, Alg. 1), which is a lower bound on the global best model score: $\gamma \leq \log P(\hat{\mathbf{y}}|\mathbf{x})$. Furthermore, since the conditionals $\log P(y_j|y_1^{j-1}, \mathbf{x})$ in Eq. 1 are log-probabilities and thus non-positive, expanding a partial hypothesis is guaranteed to result in a lower model score, i.e.:[1]

$$\forall j \in [2, J] : \log P(y_1^{j-1}|\mathbf{x}) > \log P(y_1^j|\mathbf{x}). \quad (3)$$

Consequently, when we are interested in the global best hypothesis $\hat{\mathbf{y}}$, we only need to consider partial hypotheses with scores greater than $\gamma$. In our exact decoding scheme we traverse the NMT search space in a depth-first order, but cut off branches along which the accumulated model score falls below $\gamma$. During depth-first search (DFS), we update $\gamma$ when we find a better complete hypothesis.

[1] Equality in Eq. 3 is impossible since probabilities are modeled by the neural model via a softmax function which never predicts a probability of *exactly* 1.

Alg. 2 specifies the DFS algorithm formally. An important detail is that elements in $\mathcal{T}$ are ordered such that the loop in line 5 considers the $< /\mathtt{s} >$ token first. This often updates $\gamma$ early on and leads to better pruning in subsequent recursive calls.[2]

**Exact inference under length constraints** Our admissible pruning criterion based on $\gamma$ relies on the fact that the model score of a (partial) hypothesis is always lower than the score of any of its translation prefixes. While this monotonicity condition is true for vanilla NMT (Eq. 3), it does not hold for methods like length normalization (Jean et al., 2015; Boulanger-Lewandowski et al., 2013; Wu et al., 2016) or word rewards (He et al., 2016): Length normalization gives an advantage to longer hypotheses by dividing the score by the sentence length, while a word reward directly violates monotonicity as it rewards each word with a positive value. In Sec. 4 we show how our exact search can be extended to handle arbitrary length models (Murray and Chiang, 2018; Huang et al., 2017; Yang et al., 2018) by introducing length dependent lower bounds $\gamma_k$ and report initial findings on exact search under length normalization. However, despite being of practical use, methods like length normalization and word penalties are rather heuristic as they do not have any justification from a probabilistic perspective. They also do not generalize well as (without retuning) they often work only for a specific beam size. It would be much more desirable to fix the length bias in the NMT model itself.

## 3 Results without Length Constraints

We conduct all our experiments in this section on the entire English-German WMT `news-test2015` test set (2,169 sentences) with a Transformer base (Vaswani et al., 2017) model trained with Tensor2Tensor (Vaswani et al., 2018) on parallel WMT18 data excluding ParaCrawl. Our pre-processing is as described by Stahlberg et al. (2018a) and includes joint subword segmentation using byte pair encoding (Sennrich et al., 2016) with 32K merges. We report cased BLEU scores.[3] An open-source implementation of our exact inference scheme is available in the SGNMT decoder (Stahlberg et al., 2017, 2018b).[4]

| Search | BLEU | Ratio | #Search errors | #Empty |
|---|---|---|---|---|
| Greedy | 29.3 | 1.02 | 73.6% | 0.0% |
| Beam-10 | 30.3 | 1.00 | 57.7% | 0.0% |
| Exact | 2.1 | 0.06 | 0.0% | 51.8% |

Table 1: NMT with exact inference. In the absence of search errors, NMT often prefers the empty translation, causing a dramatic drop in length ratio and BLEU.

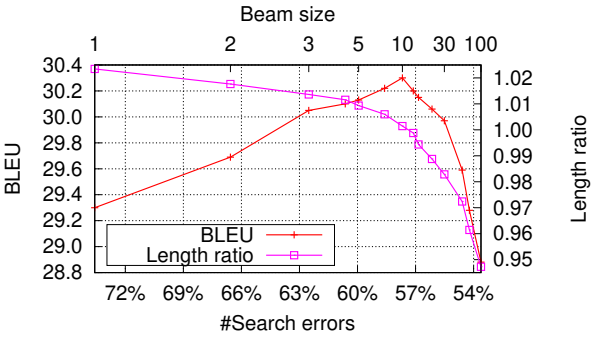


Figure 1: BLEU over the percentage of search errors. Large beam sizes yield fewer search errors but the BLEU score suffers from a length ratio below 1.

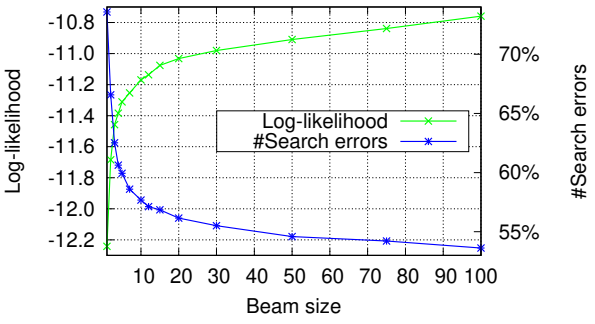


Figure 2: Even large beam sizes produce a large number of search errors.

Our main result is shown in Tab. 1. Greedy and beam search both achieve reasonable BLEU scores but rely on a high number of search errors[5] to not be affected by a serious NMT model error: For 51.8% of the sentences, NMT assigns the global best model score to the empty translation, i.e. a single $< /\mathtt{s} >$ token. Fig. 1 visualizes the relationship between BLEU and the number of search errors. Large beam sizes reduce the number of search errors, but the BLEU score drops because translations are too short. Even a large beam size of 100 produces 53.62% search errors. Fig. 2 shows that beam search effectively reduces search

[2]Note that the order in which the for-loop in line 5 of Alg. 2 iterates over $\mathcal{T}$ may be important for efficiency but does not affect the correctness of the algorithm.

[3]Comparable with `http://matrix.statmt.org/`

[4]`http://ucam-smt.github.io/sgnmt/html/`, `simpledfs` decoding strategy.

[5]A sentence is classified as search error if the decoder does not find the global best model score.

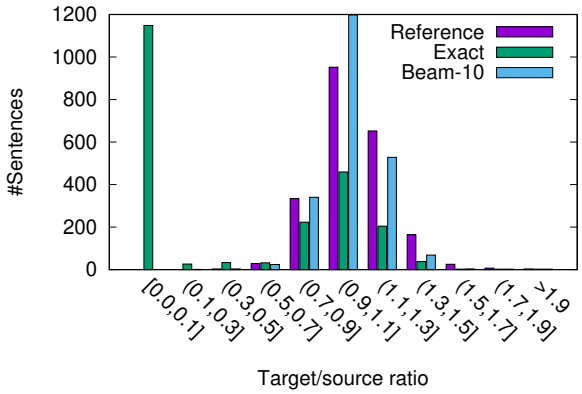


Figure 3: Histogram over target/source length ratios.

| Model | Beam-10 BLEU | Beam-10 #Search err. | Exact #Empty |
|---|---|---|---|
| LSTM* | 28.6 | 58.4% | 47.7% |
| SliceNet* | 28.8 | 46.0% | 41.2% |
| Transformer-Base | 30.3 | 57.7% | 51.8% |
| Transformer-Big* | 31.7 | 32.1% | 25.8% |

Table 2: *: The recurrent LSTM, the convolutional SliceNet (Kaiser et al., 2017), and the Transformer-Big systems are strong baselines from a WMT'18 shared task submission (Stahlberg et al., 2018a).

errors with respect to greedy decoding to some degree, but is ineffective in reducing search errors even further. For example, Beam-10 yields 15.9% fewer search errors (absolute) than greedy decoding (57.68% vs. 73.58%), but Beam-100 improves search only slightly (53.62% search errors) despite being 10 times slower than beam-10.

The problem of empty translations is also visible in the histogram over length ratios (Fig. 3). Beam search – although still slightly too short – roughly follows the reference distribution, but exact search has an isolated peak in $[0.0, 0.1]$ from the empty translations.

Tab. 2 demonstrates that the problems of search errors and empty translations are not specific to the

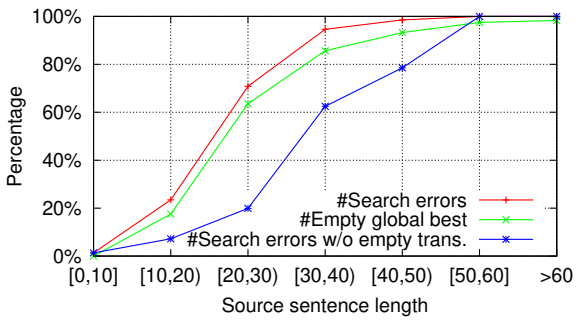


Figure 4: Number of search errors under Beam-10 and empty global bests over the source sentence length.

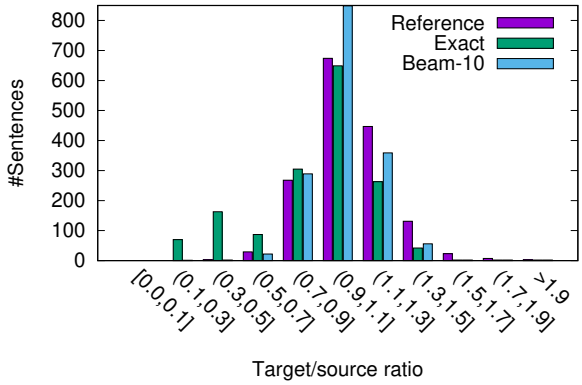


Figure 5: Histogram over length ratios with minimum translation length constraint of 0.25 times the source sentence length. Experiment conducted on 73.0% of the test set.

Transformer base model and also occur with other architectures. Even a highly optimized Transformer Big model from our WMT18 shared task submission (Stahlberg et al., 2018a) has 25.8% empty translations.

Fig. 4 shows that long source sentences are more affected by both beam search errors and the problem of empty translations. The global best translation is empty for almost all sentences longer than 40 tokens (green curve). Even without sentences where the model prefers the empty translation, a large amount of search errors remain (blue curve).

## 4 Results with Length Constraints

To find out more about the length deficiency we constrained exact search to certain translation lengths. Constraining search that way increases the run time as the $\gamma$-bounds are lower. Therefore, all results in this section are conducted on only a subset of the test set to keep the runtime under control.[6] We first constrained search to translations longer than 0.25 times the source sentence length and thus excluded the empty translation from the search space. Although this mitigates the problem slightly (Fig. 5), it still results in a peak in the $(0.3, 0.5]$ cluster. This suggests that the problem of empty translations is the consequence of an inherent model bias towards shorter hypotheses and cannot be fixed with a length constraint.

We then constrained exact search to either the

[6]We stopped decoding if the decoder took longer than a day for a single sentence on a single CPU. Exact search *without* length constraints is much faster and does not need maximum execution time limits.

| Search | BLEU | Ratio |
|---|---|---|
| Beam-10 | 37.0 | 1.00 |
| Exact for Beam-10 length | 37.0 | 1.00 |
| Exact for reference length | 37.9 | 1.01 |

Table 3: Exact search under length constraints. Experiment conducted on 48.3% of the test set.

| Search | W/o length norm. | | With length norm. | |
|---|---|---|---|---|
| | BLEU | Ratio | BLEU | Ratio |
| Beam-10 | 37.0 | 1.00 | 36.3 | 1.03 |
| Beam-30 | 36.7 | 0.98 | 36.3 | 1.04 |
| Exact | 27.2 | 0.74 | 36.4 | 1.03 |

Table 4: Length normalization fixes translation lengths, but prevents exact search from matching the BLEU score of Beam-10. Experiment conducted on 48.3% of the test set.

length of the best Beam-10 hypothesis or the reference length. Tab. 3 shows that exact search constrained to the Beam-10 hypothesis length does not improve over beam search, suggesting that any search errors between beam search score and global best score for that length are insignificant enough so as not to affect the BLEU score. The oracle experiment in which we constrained exact search to the correct reference length (last row in Tab. 3) improved the BLEU score by 0.9 points.

A popular method to counter the length bias in NMT is *length normalization* (Jean et al., 2015; Boulanger-Lewandowski et al., 2013) which simply divides the sentence score by the sentence length. We can find the global best translations under length normalization by generalizing our exact inference scheme to *length dependent* lower bounds $\gamma_k$. The generalized scheme[7] finds the best model scores for each translation length $k$ in a certain range (e.g. zero to 1.2 times the source sentence length). The initial lower bounds are derived from the Beam-10 hypothesis $\mathbf{y}_{\text{beam}}$ as follows:[8]

$$\gamma_k = (k+1)\frac{\log P(\mathbf{y}_{\text{beam}}|\mathbf{x})}{|\mathbf{y}_{\text{beam}}|+1}. \quad (4)$$

Exact search under length normalization does not suffer from the length deficiency anymore (last row in Tab. 4), but it is not able to match our best BLEU score under Beam-10 search. This suggests that while length normalization biases search towards translations of roughly the correct length, it does not fix the fundamental modelling problem.

[7] Available in our SGNMT decoder (Stahlberg et al., 2017, 2018b) as `simplelendfs` strategy.

[8] We add 1 to the lengths to avoid division by zero errors.

## 5 Related Work

Other researchers have also noted that large beam sizes yield shorter translations (Koehn and Knowles, 2017). Sountsov and Sarawagi (2016) argue that this model error is due to the locally normalized maximum likelihood training objective in NMT that underestimates the margin between the correct translation and shorter ones if trained with regularization and finite data. A similar argument was made by Murray and Chiang (2018) who pointed out the difficulty for a locally normalized model to estimate the "budget" for all remaining (longer) translations. Kumar and Sarawagi (2019) demonstrated that NMT models are often poorly calibrated, and that that can cause the length deficiency. Ott et al. (2018) argued that uncertainty caused by noisy training data may play a role. Chen et al. (2018) showed that the consistent best string problem for RNNs is decidable. We provide an alternative DFS algorithm that relies on the monotonic nature of model scores rather than consistency, and that often converges in practice.

To the best of our knowledge, this is the first work that reports the exact number of search errors in NMT as prior work often relied on approximations, e.g. via $n$-best lists (Niehues et al., 2017) or constraints (Stahlberg et al., 2018b).

## 6 Conclusion

We have presented an exact inference scheme for NMT. Exact search may not be practical, but it allowed us to discover deficiencies in widely used NMT models. We linked deteriorating BLEU scores of large beams with the reduction of search errors and showed that the model often prefers the empty translation – an evidence of NMT's failure to properly model adequacy. Our investigations into length constrained exact search suggested that simple heuristics like length normalization are unlikely to remedy the problem satisfactorily.

## Acknowledgments

This work was supported by the U.K. Engineering and Physical Sciences Research Council (EPSRC) grant EP/L027623/1 and has been performed using resources provided by the Cambridge Tier-2 system operated by the University of Cambridge Research Computing Service[9] funded by EPSRC Tier-2 capital grant EP/P020259/1.

[9] http://www.hpc.cam.ac.uk

## References


Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. 2013. Audio chord recognition with recurrent neural networks. In *ISMIR*, pages 335–340. Citeseer.

Yining Chen, Sorcha Gilroy, Andreas Maletti, Jonathan May, and Kevin Knight. 2018. Recurrent neural networks as weighted language recognizers. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2261–2271, New Orleans, Louisiana. Association for Computational Linguistics.

Wei He, Zhongjun He, Hua Wu, and Haifeng Wang. 2016. Improved neural machine translation with SMT features. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 151–157. AAAI Press.

Liang Huang, Kai Zhao, and Mingbo Ma. 2017. When to finish? Optimal beam search for neural text generation (modulo beam size). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2134–2139, Copenhagen, Denmark. Association for Computational Linguistics.

Sébastien Jean, Orhan Firat, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. Montreal neural machine translation systems for WMT'15. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 134–140, Lisbon, Portugal. Association for Computational Linguistics.

Lukasz Kaiser, Aidan N Gomez, and Francois Chollet. 2017. Depthwise separable convolutions for neural machine translation. *arXiv preprint arXiv:1706.03059*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709. Association for Computational Linguistics.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39, Vancouver. Association for Computational Linguistics.

Aviral Kumar and Sunita Sarawagi. 2019. Calibration of encoder decoder models for neural machine translation. *arXiv preprint arXiv:1903.00802*.

Kenton Murray and David Chiang. 2018. Correcting length bias in neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 212–223, Belgium, Brussels. Association for Computational Linguistics.

Jan Niehues, Eunah Cho, Thanh-Le Ha, and Alex Waibel. 2017. Analyzing neural MT search and model performance. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 11–17.

Myle Ott, Michael Auli, David Grangier, et al. 2018. Analyzing uncertainty in neural machine translation. In *International Conference on Machine Learning*, pages 3953–3962.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Pavel Sountsov and Sunita Sarawagi. 2016. Length bias in encoder decoder models and a case for global conditioning. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1516–1525, Austin, Texas. Association for Computational Linguistics.

Felix Stahlberg, Adrià de Gispert, and Bill Byrne. 2018a. The University of Cambridge's machine translation systems for WMT18. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 504–512, Belgium, Brussels. Association for Computational Linguistics.

Felix Stahlberg, Eva Hasler, Danielle Saunders, and Bill Byrne. 2017. SGNMT – A flexible NMT decoding platform for quick prototyping of new models and search strategies. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 25–30. Association for Computational Linguistics.

Felix Stahlberg, Danielle Saunders, Gonzalo Iglesias, and Bill Byrne. 2018b. Why not be versatile? Applications of the SGNMT decoder for machine translation. In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Papers)*, pages 208–216. Association for Machine Translation in the Americas.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Ashish Vaswani, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan N. Gomez, Stephan Gouws, Llion Jones, Łukasz Kaiser, Nal Kalchbrenner, Niki Parmar, Ryan Sepassi, Noam Shazeer, and Jakob Uszkoreit. 2018. Tensor2tensor for neural machine translation. *CoRR*, abs/1803.07416.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Yilin Yang, Liang Huang, and Mingbo Ma. 2018. Breaking the beam search curse: A study of (re-) scoring methods and stopping criteria for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3054–3059, Brussels, Belgium. Association for Computational Linguistics.

# *"Going on a vacation"* takes longer than *"Going for a walk"*: A Study of Temporal Commonsense Understanding

**Ben Zhou,**[1] **Daniel Khashabi,**[2] **Qiang Ning,**[3] **Dan Roth**[1]
[1]University of Pennsylvania, [2]Allen Institute for AI, [3]University of Illinois at Urbana-Champaign
{xyzhou,danroth}@cis.upenn.edu danielk@allenai.org qning2@illinois.edu

## Abstract

Understanding time is crucial for understanding events expressed in natural language. Because people rarely say the obvious, it is often necessary to have commonsense knowledge about various temporal aspects of events, such as duration, frequency, and temporal order. However, this important problem has so far received limited attention. This paper systematically studies this *temporal commonsense* problem. Specifically, we define five classes of temporal commonsense, and use crowdsourcing to develop a new dataset, MCTACO 🌮, that serves as a test set for this task. We find that the best current methods used on MCTACO are still far behind human performance, by about 20%, and discuss several directions for improvement. We hope that the new dataset and our study here can foster more future research on this topic.[1]

## 1 Introduction

Natural language understanding requires the ability to reason with *commonsense* knowledge (Schubert, 2002; Davis, 2014), and the last few years have seen significant amount of work in this direction (e.g., Zhang et al. (2017); Bauer et al. (2018); Tandon et al. (2018)). This work studies a specific type of commonsense: *temporal commonsense*. For instance, given two events *"going on a vacation"* and *"going for a walk,"* most humans would know that a vacation is typically longer and occurs less often than a walk, but it is still challenging for computers to understand and reason about temporal commonsense.

***S1:*** *Growing up on a farm near St. Paul, L. Mark Bailey didn't dream of becoming a judge.*
***Q1:*** *Is Mark still on the farm now?*
[x] no [ ] yes
**Reasoning type:** *stationarity*

***S2:*** *The massive ice sheet, called a glacier, caused the features on the land you see today.*
***Q2:*** *When did the glacier start to impact the land's features?*
[x] centuries ago [ ] hours ago
[ ] 10 years ago [x] tens of millions of years ago
**Reasoning type:** *event typical time*

***S3:*** *Carl Laemmle, head of Universal Studios, gave Einstein a tour of his studio and introduced him to Chaplin.*
***Q3:*** *How long did the tour last?*
[ ] 9 hours [ ] 15 days
[x] 45 minutes [ ] 5 seconds
**Reasoning type:** *event duration*

***S4:*** *Mr. Barco has refused U.S. troops or advisers but has accepted U.S. military aid.*
***Q4:*** *What happened after Mr. Barco accepted the military aid?*
[ ] the aid was denied [x] things started to progress
[x] he received the aid
**Reasoning type:** *event ordering*

***S5:*** *The Minangkabau custom of freely electing their leaders provided the model for rulership elections in modern federal Malaysia.*
***Q5:*** *How often are the elections held?*
[ ] every day [ ] every month
[x] every 4 years [ ] every 100 years
**Reasoning type:** *event frequency*

Figure 1: Five types of temporal commonsense in MCTACO. Note that a question may have *multiple* correct answers.

Temporal commonsense has received limited attention so far. **Our first contribution** is that, to the best of our knowledge, we are the first to systematically study and quantify performance on a range of temporal commonsense phenomena. Specifically, we consider five temporal properties: *duration* (how long an event takes), *temporal ordering* (typical order of events), *typical time* (when an event happens), *frequency* (how often an event occurs), and *stationarity* (whether a state holds for a very long time or indefinitely). Previous work has investigated some of these aspects, either explicitly or implicitly (e.g., duration (Gusev et al., 2011; Williams, 2012) and ordering (Chklovski and Pantel, 2004; Ning et al., 2018b)), but none of them have defined or studied

[1] The dataset, annotation interfaces, guidelines, and qualification tests are available at: https://cogcomp.seas.upenn.edu/page/publication_view/882.

* This work was done while the second author was affiliated with the University of Pennsylvania.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3363–3369, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

all aspects of temporal commonsense in a unified framework. Kozareva and Hovy (2011) defined a few temporal aspects to be investigated, but failed to quantify performances on these phenomena.

Given the lack of evaluation standards and datasets for temporal commonsense, **our second contribution** is the development of a new dataset dedicated for it, MCTACO (short for **m**ultiple **c**hoice **t**empor**a**l **co**mmon-sense). MCTACO is constructed via crowdsourcing with guidelines designed meticulously to guarantee its quality. When evaluated on MCTACO, a system receives a *sentence* providing context information, a *question* designed to require temporal commonsense knowledge, and multiple *candidate answers* (see Fig. 1; note that in our setup, more than one candidate answer can be plausible). We design the task as a binary classification: determining whether a candidate answer is *plausible* according to human commonsense, since there is no *absolute* truth here. This is aligned with other efforts that have posed commonsense as the choice of plausible alternatives (Roemmele et al., 2011). The high quality of the resulting dataset (shown in §4) also makes us believe that the notion of plausibility here is robust.

**Our third contribution** is that, using MCTACO as a testbed, we study the temporal commonsense understanding of the best existing NLP techniques, including *ESIM* (Chen et al., 2017), *BERT* (Devlin et al., 2019) and their variants. Results in §4 show that, despite a significant improvement over random-guess baselines, the best existing techniques are still far behind human performance on temporal commonsense understanding, indicating the need for further research in order to improve the currently limited capability to capture temporal semantics.

| Measure | | Value |
|---|---|---|
| # of unique questions | | 1893 |
| # of unique question-answer pairs | | 13,225 |
| avg. sentence length | | 17.8 |
| avg. question length | | 8.2 |
| avg. answer length | | 3.3 |
| Category | # questions | avg # of candidate |
| *event frequency* | 433 | 8.5 |
| *event duration* | 440 | 9.4 |
| *event stationarity* | 279 | 3.1 |
| *event ordering* | 370 | 5.4 |
| *event typical time* | 371 | 6.8 |

Table 1: Statistics of MCTACO.

## 2 Related Work

Commonsense has been a very popular topic in recent years and existing NLP works have mainly investigated the acquisition and evaluation of commonsense in the physical world, including but not limited to, size, weight, and strength (Forbes and Choi, 2017), roundness and deliciousness (Yang et al., 2018), and intensity (Cocos et al., 2018). In terms of "events" commonsense, Rashkin et al. (2018) investigated the intent and reaction of participants of an event, and Zellers et al. (2018) tried to select the most likely subsequent event. To the best of our knowledge, no earlier work has focused on *temporal* commonsense, although it is critical for event understanding. For instance, Ning et al. (2018c) argues that resolving ambiguous and implicit mentions of event durations in text (a specific kind of temporal commonsense) is necessary to construct the timeline of a story.

There have also been many works trying to understand time in natural language but not necessarily the commonsense understanding of time. Most recent works include the extraction and normalization of temporal expressions (Strötgen and Gertz, 2010; Lee et al., 2014), temporal relation extraction (Ning et al., 2017, 2018d), and timeline construction (Leeuwenberg and Moens, 2018). Among these, some works are implicitly on temporal commonsense, such as event durations (Williams, 2012; Vempala et al., 2018), typical temporal ordering (Chklovski and Pantel, 2004; Ning et al., 2018a,b), and script learning (i.e., what happens next after certain events) (Granroth-Wilding and Clark, 2016; Li et al., 2018). However, existing works have not studied all five types of temporal commonsense in a unified framework as we do here, nor have they developed datasets for it.

Instead of working on each individual aspect of temporal commonsense, we formulate the problem as a machine reading comprehension task in the format of selecting plausible responses with respect to natural language queries. This relates our work to a large body of work on question-answering, an area that has seen significant progress in the past few years (Clark et al., 2018; Ostermann et al., 2018; Merkhofer et al., 2018). This area, however, has mainly focused on *general* natural language comprehension tasks, while we tailor it to test a *specific* reasoning capability, which is temporal commonsense.

## 3 Construction of MCTACO

MCTACO is comprised of $13k$ tuples, in the form of *(sentence, question, candidate answer)*; please see examples in Fig. 1 for the five phenomena studied here and Table 1 for basic statistics of it. The sentences in those tuples are randomly selected from MultiRC (Khashabi et al., 2018) (from each of its 9 domains). For each sentence, we use crowdsourcing on Amazon Mechanical Turk to collect questions and candidate answers (both correct and wrong ones). To ensure the quality of the results, we limit the annotations to native speakers and use qualification tryouts.

**Step 1: Question generation.** We first ask crowdsourcers to generate questions, given a sentence. To produce questions that need temporal commonsense to answer, we require that a valid question: (a) should ask about one of the five temporal phenomena we defined earlier, and (b) should not be solved simply by a word or phrase from the original sentence. We also require crowdsourcers to provide a correct answer for each of their questions, which on one hand gives us a positive candidate answer, and on the other hand ensures that the questions are answerable at least by themselves.

**Step 2: Question verification.** We further ask another two crowdsourcers to check the questions generated in Step 1, i.e., (a) whether the two requirements are satisfied and (b) whether the question is grammatically and logically correct. We retain only the questions where the two annotators unanimously agree with each other and the decision generated in Step 1. For valid questions, we continue to ask crowdsourcers to give one correct answer and one incorrect answer, which we treat as a seed set to automatically generate new candidate answers in the next step.

**Step 3: Candidate answer expansion.** Until this stage, we have collected a small set of candidate answers (3 positive and 2 negative) for each question.[2] We automatically expand this set in three ways. First, we use a set of rules to extract numbers and quantities ("2", "once") and temporal terms (e.g. "a.m.", "1990", "afternoon", "day"), and then randomly perturb them based on a list of temporal units ("second"), adjectives ("early"), points ( "a.m.") and adverbs ("always"). Examples are "2 a.m." → "3 p.m.", "1 day" → "10 days", "once a week"→ "twice a month" (more details in the appendix).

Second, we mask each individual token in a candidate answer (one at a time) and use *BERT* (Devlin et al., 2019) to predict replacements for each missing term; we rank those predictions by the confidence level of *BERT* and keep the top three.

Third, for those candidates that represent events, the previously-mentioned token-level perturbations rarely lead to interesting and diverse set of candidate answers. Furthermore, it may lead to invalid phrases (e.g., "he left the house" → "he walked the house".) Therefore, to perturb such candidates, we create a pool of $60k$ event phrases using PropBank (Kingsbury and Palmer, 2002), and perturb the candidate answers to be the most similar ones extracted by an information retrieval (*IR*) system.[3] This not only guarantees that all candidates are properly phrased, it also leads to more diverse perturbations.

We apply the above three techniques on non-"event" candidates sequentially, in the order they were explained, to expand the candidate answer set to 20 candidates per question. A perturbation technique is used, as long as the pool of candidates is still less than 20. Note there are both correct and incorrect answers in those candidates.

**Step 4: Answer labeling.** In this step, each (*sentence, question, answer*) tuple produced earlier is labeled by 4 crowdsourcers, with three options: "likely", "unlikely", or "invalid" (sanity check for valid tuples).[4] Different annotators may have different interpretations, yet we ensure label validity through high agreement. A tuple is kept only if all 4 annotators agree on "likely" or "unlikely". The final statistics of MCTACO is in Table 1.

## 4 Experiments

We assess the quality of our dataset through human annotation, and evaluate a couple of baseline systems. We create a uniform split of 30%/70% of the data to dev/test. The rationale behind this split is that, a successful system has to bring in a

[2] One positive answer from Step 1; one positive and one negative answer from each of the two annotators in Step 2.

[3] www.elastic.co

[4] We use the name "(un)likely" because commonsense decisions can be naturally ambiguous and subjective.

huge amount of world knowledge and derive commonsense understandings *prior* to the current task evaluation. We therefore believe that it is not reasonable to expect a system to be *trained* solely on this data, and we think of the development data as only providing a *definition* of the task. Indeed, the gains from our development data are marginal after a certain number of training instances. This intuition is studied and verified in Appendix A.2.

**Evaluation metrics.** Two question-level metrics are adopted in this work: exact match (*EM*) and *F1*. For a given candidate answer $a$ that belongs to a question $q$, let $f(a;q) \in \{0,1\}$ denote the correctness of the prediction made by a fixed system (1 for correct; 0 otherwise). Additionally, let $D$ denote the collection of questions in our evaluation set.

$$EM \triangleq \frac{\sum_{q \in D} \prod_{a \in q} f(a;q)}{|\{q \in D\}|}$$

The recall for each question $q$ is:

$$R(q) = \frac{\sum_{a \in q} [f(a;q) = 1] \wedge [a \text{ is "likely" }]}{|\{a \text{ is "likely" } \wedge a \in q\}|}$$

Similarly, $P(q)$ and $F1(q)$ are defined. The aggregate $F1$ (across the dataset $D$) is the macro average of question-level $F1$'s:

$$F1 \triangleq \frac{\sum_{q \in D} F1(q)}{|\{q \in D\}|}$$

*EM* measures how many questions a system is able to correctly label all candidate answers, while *F1* is more relaxed and measures the average overlap between one's predictions and the ground truth.

**Human performance.** An expert annotator also worked on MCTACO to gain a better understanding of the human performance on it. The expert answered 100 questions (about 700 (*sentence, question, answer*) tuples) randomly sampled from the test set, and could only see a single answer at a time, with its corresponding question and sentence.

**Systems.** We use two state-of-the-art systems in machine reading comprehension for this task: *ESIM* (Chen et al., 2017) and *BERT* (Devlin et al., 2019). *ESIM* is an effective neural model on natural language inference. We initialize the word embeddings in *ESIM* via either *GloVe* (Pennington et al., 2014) or *ELMo* (Peters et al., 2018) to demonstrate the effect of pre-training. *BERT* is a state-of-the-art contextualized representation used for a broad range of tasks . We also add unit normalization to *BERT*, which extracts and converts temporal expressions in candidate answers to their most proper units. For example, "30 months" will be converted to "2.5 years". To the best of our knowledge, there are no other available systems for the "stationarity", "typical time", and "frequency" phenomena studied here. As for "duration" and "temporal order", there are existing systems (e.g., Vempala et al. (2018); Ning et al. (2018b)), but they cannot be directly applied to the setting in MCTACO where the inputs are natural languages.

| System | *F1* | *EM* |
|---|---|---|
| Random | 36.2 | 8.1 |
| Always Positive | 49.8 | 12.1 |
| Always Negative | 17.4 | 17.4 |
| *ESIM + GloVe* | 50.3 | 20.9 |
| *ESIM + ELMo* | 54.9 | 26.4 |
| *BERT* | 66.1 | 39.6 |
| *BERT + unit normalization* | **69.9** | **42.7** |
| Human | 87.1 | 75.8 |

Table 2: Summary of the performances for different baselines. All numbers are in percentages.

**Experimental setting.** In both *ESIM* baselines, we model the process as a sentence-pair classification task, following the *SNLI* setting in AllenNLP.[5] In both versions of *BERT*, we use the same sequence pair classification model and the same parameters as in *BERT*'s *GLUE* experiments.[6] A system receives two elements at a time: (a) the concatenation of the sentence and question, and (b) the answer. The system makes a binary prediction on each instance, "likely" or "unlikely".

**Results and discussion.** Table 2 compares native baselines, *ESIM*, *BERT* and their variants on the entire test set of MCTACO; it also shows human performance on the subset of 100 questions.[7] The system performances reported are based on default random seeds, and we observe a maximum

[5] https://github.com/allenai/allennlp

[6] https://github.com/huggingface/pytorch-pretrained-BERT

[7] *BERT + unit normalization* scored $F1 = 72, EM = 45$ on this subset, which is only slightly different from the corresponding number on the entire test set.

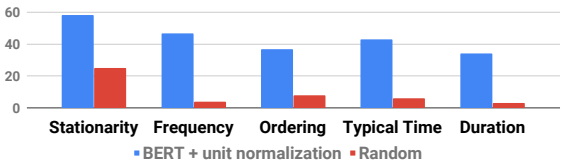


Figure 2: *EM* scores of *BERT + unit normalization* per temporal reasoning category comparing to the random-guess baseline.

standard error [8] of 0.8 from 3 runs on different seeds across all entries. We can confirm the good quality of this dataset based on the high performance of human annotators. *ELMo* and *BERT* improve naive baselines by a large margin, indicating that a notable amount of commonsense knowledge has been acquired via pre-training. However, even *BERT* still falls far behind human performance, indicating the need of further research. [9]

We know that *BERT*, as a language model, is good at associating surface forms (e.g. associating "sunrise" with "morning" since they often co-occur), but may be brittle with respect to variability of temporal mentions.

Consider the following example (the correct answers are indicated with ✓and *BERT* selections are underlined.) This is an example of *BERT* correctly associating a given event with "minute" or"hour"; however, it fails to distinguish between "1 hour" (a "likely" candidate) and "9 hours" (an "unlikely" candidate).

**P:** *Ratners's chairman, Gerald Ratner, said the deal remains of "substantial benefit to Ratners."*
**Q:** *How long did the chairman speak?*
✓(a) 30 minutes ✓(b) 1 hour
✗(c) 9 hours ✗(d) twenty seconds

This shows that *BERT* does not infer a range of true answers; it instead associates discrete terms and decides individual options, which may not be the best way to handle temporal units that involve numerical values.

*BERT+unit normalization* is used to address this issue, but results show that it is still poor compared to human. This indicates that the information acquired by *BERT* is still far from solving temporal commonsense.

Since exact match (EM) is a stricter metric, it is consistently lower than $F1$ in Table 2. For an ideal system, the gap between EM and $F1$ should be small (humans only drop 11.3%.) However, all other systems drop by almost 30% from $F1$ to EM, possibly another piece of evidence that they only associate surface forms instead of using one representation for temporal commonsense to classify all candidates.

A curious reader might ask why the human performance on this task as shown in Table 2 is not 100%. This is expected because commonsense is what *most* people agree on, so any *single* human could disagree with the gold labels in MCTACO. Therefore, we think the human performance in Table 2 from a single evaluator actually indicates the good quality of MCTACO.

The performance of *BERT+unit normalization* is not uniform across different categories (Fig. 2), which could be due to the different nature or quality of data for those temporal phenomena. For example, as shown in Table 1, "stationarity" questions have much fewer candidates and a higher random baseline.

## 5 Conclusion

This work has focused on temporal commonsense. We define five categories of questions that require temporal commonsense and develop a novel crowdsourcing scheme to generate MCTACO, a high-quality dataset for this task. We use MCTACO to probe the capability of systems on temporal commonsense understanding. We find that systems equipped with state-of-the-art language models such as *ELMo* and *BERT* are still far behind humans, thus motivating future research in this area. Our analysis sheds light on the capabilities as well as limitations of current models. We hope that this study will inspire further research on temporal commonsense.

## Acknowledgements

This research is supported by a grant from the Allen Institute for Artificial Intelligence (allenai.org) and by contract HR0011-18-2-0052 and HR0011-15-C-0113 with the US Defense Advanced Research Projects Agency (DARPA). Approved for Public Release, Distribution Unlimited. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

[8] https://en.wikipedia.org/wiki/Standard_error

[9] RoBERTa (Liu et al., 2019), a more recent language model that was released after this paper's submission, achieves $F1 = 72.3, EM = 43.6$.

## References


Lisa Bauer, Yicheng Wang, and Mohit Bansal. 2018. Commonsense for generative multi-hop question answering tasks. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 4220–4230.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for natural language inference. In *ACL*, Vancouver. ACL.

Timothy Chklovski and Patrick Pantel. 2004. VerbOcean: Mining the Web for Fine-Grained Semantic Verb Relations. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 33–40.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge. *CoRR*, abs/1803.05457.

Anne Cocos, Veronica Wharton, Ellie Pavlick, Marianna Apidianaki, and Chris Callison-Burch. 2018. Learning scalar adjective intensity from paraphrases. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1752–1762.

Ernest Davis. 2014. *Representations of commonsense knowledge*. Morgan Kaufmann.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Maxwell Forbes and Yejin Choi. 2017. Verb physics: Relative physical knowledge of actions and objects. In *ACL*, volume 1, pages 266–276.

Mark Granroth-Wilding and Stephen Christopher Clark. 2016. What happens next? event prediction using a compositional neural network model. In *ACL*.

Andrey Gusev, Nathanael Chambers, Pranav Khaitan, Divye Khilnani, Steven Bethard, and Dan Jurafsky. 2011. Using query patterns to learn the duration of events. In *IWCS*, pages 145–154. Association for Computational Linguistics.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *NAACL*.

Paul Kingsbury and Martha Palmer. 2002. From treebank to propbank. In *LREC*, pages 1989–1993.

Zornitsa Kozareva and Eduard Hovy. 2011. Learning temporal information for states and events. In *Fifth International Conference on Semantic Computing*, pages 424–429. IEEE.

Kenton Lee, Yoav Artzi, Jesse Dodge, and Luke Zettlemoyer. 2014. Context-dependent semantic parsing for time expressions. In *ACL (1)*, pages 1437–1447.

Artuur Leeuwenberg and Marie-Francine Moens. 2018. Temporal information extraction by predicting relative time-lines. *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*.

Zhongyang Li, Xiao Ding, and Ting Liu. 2018. Constructing narrative event evolutionary graph for script event prediction. *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Elizabeth Merkhofer, John Henderson, David Bloom, Laura Strickhart, and Guido Zarrella. 2018. Mitre at semeval-2018 task 11: Commonsense reasoning without commonsense knowledge. In *SemEval*, pages 1078–1082.

Qiang Ning, Zhili Feng, and Dan Roth. 2017. A structured learning approach to temporal relation extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Qiang Ning, Zhili Feng, Hao Wu, and Dan Roth. 2018a. Joint reasoning for temporal and causal relations. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Qiang Ning, Hao Wu, Haoruo Peng, and Dan Roth. 2018b. Improving temporal relation extraction with a globally acquired statistical resource. In *Proceedings of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*.

Qiang Ning, Hao Wu, and Dan Roth. 2018c. A multi-axis annotation scheme for event temporal relations. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*. Association for Computational Linguistics.

Qiang Ning, Ben Zhou, Zhili Feng, Haoruo Peng, and Dan Roth. 2018d. CogCompTime: A tool for understanding time in natural language. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*.

Simon Ostermann, Michael Roth, Ashutosh Modi, Stefan Thater, and Manfred Pinkal. 2018. Semeval-2018 task 11: Machine comprehension using commonsense knowledge. In *SemEval*, pages 747–757.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*.

Hannah Rashkin, Maarten Sap, Emily Allaway, Noah A. Smith, and Yejin Choi. 2018. Event2Mind: Commonsense inference on events, intents, and reactions. In *Proc. of the Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 463–473.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *2011 AAAI Spring Symposium Series*.

Lenhart Schubert. 2002. Can we derive general world knowledge from texts? In *Proceedings of the second international conference on Human Language Technology Research*, pages 94–97. Morgan Kaufmann Publishers Inc.

Jannik Strötgen and Michael Gertz. 2010. HeidelTime: High quality rule-based extraction and normalization of temporal expressions. In *SemEval*, pages 321–324. Association for Computational Linguistics.

Niket Tandon, Bhavana Dalvi, Joel Grus, Wen-tau Yih, Antoine Bosselut, and Peter Clark. 2018. Reasoning about actions and state changes by injecting commonsense knowledge. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 57–66.

Alakananda Vempala, Eduardo Blanco, and Alexis Palmer. 2018. Determining event durations: Models and error analysis. In *NAACL*, volume 2, pages 164–168.

Jennifer Williams. 2012. Extracting fine-grained durations for verbs from twitter. In *Proceedings of ACL 2012 Student Research Workshop*, pages 49–54. Association for Computational Linguistics.

Yiben Yang, Larry Birnbaum, Ji-Ping Wang, and Doug Downey. 2018. Extracting commonsense properties from embeddings with limited human guidance. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 644–649.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. SWAG: A Large-Scale Adversarial Dataset for Grounded Commonsense Inference. In *EMNLP*, pages 93–104.

Sheng Zhang, Rachel Rudinger, Kevin Duh, and Benjamin Van Durme. 2017. Ordinal common-sense inference. *Transactions of the Association of Computational Linguistics*, 5(1):379–395.

# QAInfomax: Learning Robust Question Answering System by Mutual Information Maximization

**Yi-Ting Yeh  Yun-Nung Chen**
Department of Computer Science and Information Engineering
National Taiwan University, Taipei, Taiwan
r07922064@csie.ntu.edu.tw  y.v.chen@ieee.org

## Abstract

Standard accuracy metrics indicate that modern reading comprehension systems have achieved strong performance in many question answering datasets. However, the extent these systems truly understand language remains unknown, and existing systems are not good at distinguishing distractor sentences, which look related but do not actually answer the question. To address this problem, we propose QAInfomax as a regularizer in reading comprehension systems by maximizing mutual information among passages, a question, and its answer. QAInfomax helps regularize the model to not simply learn the superficial correlation for answering questions. The experiments show that our proposed QAInfomax achieves the state-of-the-art performance on the benchmark Adversarial-SQuAD dataset[1].

## 1 Introduction

Question answering tasks are widely used for training and testing machine comprehension and reasoning (Rajpurkar et al., 2016; Joshi et al., 2017). However, high performance in standard automatic metrics has been achieved with only superficial understanding, as models exploit simple correlations in the data that happen to be predictive on most test examples. Jia and Liang (2017) addressed this problem and proposed an adversarial version of the SQuAD dataset, which was created by adding a distractor sentence to each paragraph. The distractor sentences challenge the model robustness, and the created Adversarial-SQuAD data shows the inability of a model about distinguishing a sentence that actually answers the question from one that merely has words in common with it, where almost all state-of-the-art machine comprehension systems are significantly degraded on adversarial examples.

Lewis and Fan (2018) argued that over-fitting to superficial biases is partially caused by discriminative loss functions, which saturate when simple correlations allow the question to be answered confidently, leaving no incentive for further learning on the example. Therefore, they designed generative QA models, which use a generative loss function in question answering instead, and showed the improvement on Adversarial-SQuAD.

Instead of regularizing models by generative loss functions, we propose an alternative approach named "QAInfomax" by maximizing mutual information (MI) among passages, questions, and answers, aiming at helping models be not stuck with superficial biases in the data during learning. To efficiently estimate MI, QAInfomax incorporates the recently proposed deep infomax (DIM) in the model (Hjelm et al., 2018), which was proved effective in learning representations for image, audio (Ravanelli and Bengio, 2018), and graph domains (Veličković et al., 2018). In this work, the proposed QAInfomax further extends DIM to the text domain, and encourages the question answering model to generate answers carrying information that can explain not only questions but also itself, and thus be more sensitive to distractor sentences. Our contributions are summarized:

- This paper first attempts at applying DIM-based MI estimation as a regularizer for representation learning in the NLP domain.
- The proposed QAInfomax achieves the state-of-the-art performance on the Adversarial-SQuAD dataset without additional training data, demonstrating its better robustness.

## 2 Mutual Information (MI) Estimation

In this section, we introduce how scalable estimation of mutual information is performed in terms

[1]The source code is publicly available at https://github.com/MiuLab/QAInfomax.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3370–3375,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

of practical scenarios via mutual information neural estimation (MINE) (Belghazi et al., 2018) and the deep infomax (DIM) (Hjelm et al., 2018) described below.

The mutual information between two random variable $X$ and $Y$ is defined as:

$$\text{MI}(X, Y) = D_{\text{KL}}(p(X, Y) \parallel p(X)p(Y)),$$

where $D_{KL}$ is the Kullback-Leibler (KL) divergence between the joint distribution $p(X, Y)$ and the product of marginals $p(X)p(Y)$.

MINE estimates mutual information by training a classifier to distinguish between positive samples $(x, y)$ from the joint distribution and negative samples $(x, \bar{y})$ from the product of marginals. Mutual information neural estimation (MINE) uses Donsker-Varadhan representation (DV) (Donsker and Varadhan, 1983) as a lower-bound to estimate MI.

$$\text{MI}(X, Y) \geq \mathbb{E}_{\mathbb{P}}[g(x, y)] - \log(\mathbb{E}_{\mathbb{N}}[e^{g(x,\bar{y})}]),$$

where $\mathbb{E}_{\mathbb{P}}$ and $\mathbb{E}_{\mathbb{N}}$ denote the expectation over positive and negative samples respectively, and $g$ is the discriminator function that outputs a real number modeled by a neural network.

While the DV representation is the strong bound of mutual information shown in MINE, we are primarily interested in maximizing MI but not focusing on its precise value. Thus DIM proposes an alternative estimation using Jensen-Shannon divergence (JS), which can be efficiently implemented using the cross-entropy (BCE) loss:

$$\begin{aligned} \text{MI}(X, Y) \quad &\geq \quad \mathbb{E}_{\mathbb{P}}[\log(g(x, y))] \\ &+ \quad \mathbb{E}_{\mathbb{N}}[\log(1 - g(x, \bar{y}))]. \end{aligned} \tag{1}$$

While two representations should behave similarly, considering that both act like classifiers with objectives maximizing the expected log-ratio of the joint over the product of marginals, it is found that the BCE loss empirically works better than the DV-based objective (Hjelm et al., 2018; Ravanelli and Bengio, 2018; Veličković et al., 2018). The reason may be that the BCE loss is bounded (i.e., its maximum is zero), making the convergence of the network more numerically stable. In our experiments, we primarily use the JS representation to estimate mutual information.

Recently, Tian et al. (2019) showed strong empirical performance through the improved multiview CPC training (Oord et al., 2018), which shares many common ideas as mutual information maximization. Inspired by their work, we modify (1) by first switching the role of $x$ and $y$ and summing them up:

$$\begin{aligned} \text{MI}(X, Y) \quad &\geq \quad \mathbb{E}_{\mathbb{P}}[\log(g(x, y))] \\ &+ \quad \frac{1}{2}\mathbb{E}_{\mathbb{N}}[\log(1 - g(x, \bar{y}))] \\ &+ \quad \frac{1}{2}\mathbb{E}_{\mathbb{N}}[\log(1 - g(\bar{x}, y))], \end{aligned} \tag{2}$$

where $(\bar{x}, y)$ is also the negative sample sampled from the product of marginals.

We empirically find that (2) gives the best performance, and more exploration about parameterization of MI is left as our future work.

## 3 Methodology

In the extractive question answering dataset like SQuAD, the answer $A = \{a_1, \ldots, a_M\}$ to the question $Q = \{q_1, \ldots, q_K\}$ is guaranteed to be the span $\{p_m, \ldots, p_{m+M}\}$ in the paragraph $P = \{p_1, \ldots, p_N\}$. Given $Q$ and $P$, the encoded representations from the QA system $M$ can be formulated as:

$$\{r^q, r^p\} = \{r^q_1, \ldots, r^q_K, r^p_1, \ldots, r^p_N\} = M(Q, P),$$

where $r^q$ and $r^p$ are representations of the question and the passage respectively after the reasoning process in the QA system $M$.

Most models then feed the passage representation $r^p$ to a single-layer neural network, obtain the span start and end probabilities for each passage word, and compute the loss $L_{span}$, which is the negative sum of log probabilities of the predicted distributions indexed by true start and end indices.

Our QAInfomax aims at regularizing the QA system $M$ to not simply exploit the superficial biases in the dataset for answering questions. Therefore, two constraints are introduced in order to guide the model learning.

1. **Local Constraint (LC)**: each answer word representation $r^p_i$ in the answer representation $r^a = \{r^p_m, \ldots, r^p_{m+M}\}$ should contain information about what the remaining answer words and its surrounding context are.

2. **Global Constraint (GC)**: the summarized answer representation $s = S(r^a)$ should maximize the averaged mutual information to all other question representations in $r^q$ and passage representations in $r^p$, where $S$ is a summarization function described below.

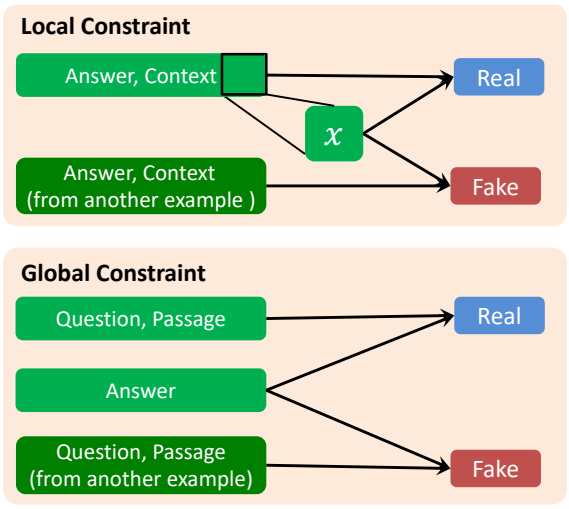


Figure 1: Illustration of the LC and GC.

Intuitively, the model is expected to *choose the answer span after fully considering the entire question and paragraph*. However, traditional QA models suffered the overstability problem, and tended to be fooled by distractor answers, such as the one containing an unrelated human name. As Lewis and Fan (2018) argued, we also believe that the main reason is that QA models are only trained to predict start and end positions of answer spans. Correlation in the dataset allows QA models to find shortcuts and ignore what the answer span looks like. A learned behavior of traditional QA models can be viewed as a simple pattern matching, such as choosing the 5-length span after the word "river" if a question is about a river and the context talks about countries in European.

Following the intuition, two constraints LC and GC are introduced to guide models to learn the desired behaviors. To prevent the model from only learning to match some specific word patterns to find the answer, LC forces the model to generate answer span representations while maximizing mutual information among words in the span and the context words surrounding the span. By maximizing the mutual information between an answer word and *all* of its context words, models need to incorporate the entire context into its decision process while choosing answers, and thus can be more robust to the adversarial sentences. Then we further require models to maximize mutual information among answer words, so models can no longer ignore any word in the chosen answer span.

On the other hand, different from LC , which only focuses on the answer span and its context, GC pushes the model to prefer answer representations carrying information that is globally shared across the whole input conditions $Q$ and $P$, because shortcuts do not necessarily appear near to the answer. If the model only learns to leverage the correlation specific to the partial input, the MI of any input word without such relationship would *not* increased.

The overview about two proposed constraints is illustrated in Figure 1. The detail of two constraints and our QAInfomax regularizer is described below.

### 3.1 Local Constraint

As shown in Section 2, the maximization of MI needs positive samples and negative samples drawn from joint distribution and the product of marginal distribution respectively.

In LC , because all answer word representations are expected to carry the information of each other and their contexts, we choose to maximize averaged MI between the sampled answer word representations and the whole answer sequence with its context words. Specifically, a positive sample is obtained by pairing the sampled answer word representation $x \in r^a = \{r^p_m, \ldots, r^p_{m+M}\}$ to all other answer and context words $r^c = \{r^p_{m-C}, \ldots r^p_{m+M+C}\} \setminus \{x\}$, where $C$ is the hyperparameter defining how many context words for consideration. Negative samples, on the other hand, are obtained by randomly sampling answer representation $\bar{r}^a = \{\bar{r}^p_l, \ldots, \bar{r}^p_{l+L}\}$ and the corresponding $\bar{r}^c$ from other training examples. Following (2), the objective for sampled $x$, $r^c$, $\bar{x} \in \bar{r}^a$ and $\bar{r}^c$ is formulated.

$$\begin{aligned} \mathrm{LC}(x, r^c, \bar{x}, \bar{r}^c) &= \frac{1}{|r^c|} \sum_{r^c_i \in r^c} \log(g(x, r^c_i)) \\ &+ \frac{1}{2|\bar{r}^c|} \sum_{\bar{r}^c_j \in \bar{r}^c} \log(1 - g(x, \bar{r}^c_j)) \\ &+ \frac{1}{2|r^c|} \sum_{r^c_i \in r^c} \log(1 - g(\bar{x}, r^c_i)). \end{aligned} \quad (3)$$

### 3.2 Global Constraint

Different from LC described above, GC forces the learned answer representations $r^a$ to have information shared with all other question and passage representations. Here, we maximize the mutual information between the summarized answer vector $s = S(r^a)$ and $r_l \in r = \{r^q, r^p\} \setminus \{r^a\}$ pairs. In

| Model | Original | ADDSENT | ADDONESENT |
|---|---|---|---|
| BiDAF-S (Seo et al., 2016) | 75.5 | 34.3 | 45.7 |
| ReasoNet-S (Shen et al., 2017) | 78.2 | 39.4 | 50.3 |
| Reinforced Mnemonic Reader-S (Hu et al., 2017) | 78.5 | 46.6 | 56.0 |
| QANet-S (Yu et al., 2018) | 83.8 | 45.2 | 55.7 |
| GQA-S (Lewis and Fan, 2018) | 83.7 | 47.3 | 57.8 |
| FusionNet-E (Huang et al., 2017) | 83.6 | 51.4 | 60.7 |
| BERT-S (Devlin et al., 2018) | 88.5 | 51.0 | 63.4 |
| BERT-S + QAInfomax | **88.6** | **54.5** † | **64.9** † |

Table 1: F-measure on ADVERSARIALSQUAD (S: single, E: ensemble). † indicates the significant improvement over baselines with p-value $< 0.05$.

the experiments, we use $S(r^a) = \sigma(\frac{1}{M}\sum r_i^a)$ as our summarization function, where $\sigma$ is the logistic sigmoid nonlinearity.

Specifically, a positive sample here is the pair of a answer summary vector $s = S(r^a)$ and all other word representations in $r$. Negative samples are provided by sampling question, passage and answer representations $\{\bar{r}^q, \bar{r}^p, \bar{r}^a\}$ from an alternative training example. Then we pair the summary $s$ with $\bar{r} = \{\bar{r}^q, \bar{r}^p\} \setminus \{\bar{r}^a\}$, and $\bar{s} = S(\bar{r}^a)$ with $r$.

Similar to (3), the objective for the sampled $s$, $r$, $\bar{s}$ and $\bar{r}$ is:

$$\begin{aligned} \mathrm{GC}(s, r, \bar{s}, \bar{r}) &= \frac{1}{|r|}\sum_{r_i \in r} \log(g(s, r_i)) \\ &+ \frac{1}{2|\bar{r}|}\sum_{\bar{r}_j \in \bar{r}} \log((1 - g(s, \bar{r}_j))) \\ &+ \frac{1}{2|r|}\sum_{r_i \in r} \log((1 - g(\bar{s}, r_i))). \end{aligned} \tag{4}$$

### 3.3 QAInfomax

In our proposed model, we combine two objectives and formulate the model as the complete QAInfomax regularizer. For each training batch consisting of training examples $\{\{Q_1, P_1, A_1\}, \ldots \{Q_B, P_B, A_B\}\}$, we pass the batch into the model $M$ and obtain representations $\{\{r_1^q, r_1^p, r_1^a\}, \ldots, \{r_B^q, r_B^p, r_B^a\}\}$. Note that we abuse the subscripts to denote the example index in the batch for simplicity.

Then we shuffle the whole batch to obtain negative examples $\{\{\bar{r}_1^q, \bar{r}_1^p, \bar{r}_1^a\}, \ldots, \{\bar{r}_B^q, \bar{r}_B^p, \bar{r}_B^a\}\}$. The complete objective $L_{info}$ for QAInfomax becomes:

$$-\frac{1}{B}\sum_{i=1}^{B}(\alpha LC(x_i, r_i^c, \bar{x}_i, \bar{r}_i^c) + \beta GC(s_i, r_i, \bar{r}_i)),$$

where $x_i$ and $\bar{x}_i$ are the representation sampled from $r_i^a$ and $\bar{r}_i^a$, $r_i^c$ and $\bar{r}_i^c$ are $r_i^a$ and $\bar{r}_i^a$ expanded with its context words respectively, $s_i$ and $\bar{s}_i$ are the summary vectors of $r_i^a$ and $\bar{r}_i^a$, and $\alpha$ and $\beta$ are hyperparameters.

Combined with QAInfomax as a regularizer, the final objective of the model becomes

$$L = L_{span} + \gamma L_{info}, \tag{5}$$

where $L_{span}$ is the answer span prediction loss and $\gamma$ is the regularize strength. The objective can be optimized through the simple gradient descent.

## 4 Experiments

To evaluate the effectiveness of the proposed QAInfomax, we conduct the experiments on a challenging dataset, Adversarial-SQuAD.

### 4.1 Setup

BERT-base (Devlin et al., 2018) is employed as our QA system $M$ in the experiments, where we set the same hyperparameters as one released in SQuAD training[2].

We set $C$, $\alpha$, $\beta$ and $\gamma$ to be 5, 1, 0.5, 0.3 respectively in all experiments, and add the proposed QAInfomax into the BERT model as described above. The discriminator function $g$ is the bilinear function similar to the scoring used by Oord et al. (2018):

$$g(x, y) = x^T W y, \tag{6}$$

where $W$ is a learnable scoring matrix.

We train the BERT model with the proposed QAInfomax on the orignal SQuAD dataset, and

[2]We use PyTorch (Paszke et al., 2017) reimplementation for experiments: https://github.com/huggingface/pytorch-pretrained-BERT.

| Model | Adversary F1 | Speed (iter/s) |
|---|---|---|
| BERT | 51.0 / 63.4 | 3.80 |
| + LC | 53.6 / 64.2 | 3.51 |
| + GC | 52.2 / 63.7 | 2.75 |
| + LC + GC | **54.5 / 64.9** | 2.72 |

Table 2: Ablation study with F1 scores on ADDSENT / ADDONESENT. The speed is measured on RTX 2080Ti.

use Adversarial-SQuAD to test the robustness of the augmented model. Only ADDSENT and ADDONESENT metrics are reported for the comparison with previous models, because most previous models did not report their ADDANY and ADDCOMMON scores. Briefly, for each example, ADDSENT runs the model $M$ on every human-approved adversarial sentence, picks the one that makes the model give the worst answer and returns that score. ADDONESENT, on the other hand, only picks a random human-approved adversarial sentence. The numbers reported in all experiments are the best number across at least three runs.

### 4.2 Results

Table 1 reports model performance on Adversarial-SQuAD. It can be found that QAInfomax yields substantial improvement over the vanilla BERT model, and achieves the state-of-the-art performance on both ADDSENT and ADDONESENT metrics.[3] QAInfomax obtains larger improvement on the ADDSENT, which picks the worst scores of the model. It shows the effectiveness of our QAInfomax in terms of forcing the model to ignore simple correlation in the data and learn the more human-like reasoning processes. It is worth to note that while QAInfomax mitigates the overstability problem and improves the robustness to adversarial examples, it does not hurt the original performance of the QA system, demonstrating the benefit for the practical usage. Some example results from the Adversarial-SQuAD dataset can be found in the Appendix, where adversarial distracting sentences are shown in italic blue fonts.

Table 2 shows the ablation study of our proposed QAInfomax, where two proposed constraints are both important for achieving such results. We also show the training speed of the proposed method and its limitation, where the GC objective degrades the training speed by $28\%$. The reason is that GC measures the averaged MI over the *whole* question and passage representations, which may include a long sequence of vectors.

[3]Note that Wang and Bansal modified distractor paragraphs and added them into training data, so we do not compare with them, because we only use the original SQuAD training data.

| Function | ADDSENT | ADDONESENT |
|---|---|---|
| Mean | **52.2** | **63.7** |
| Max | 52.0 | 63.3 |
| Sample | **52.2** | 63.0 |

Table 3: Different summarization functions for GC .

Considering that the summarization function $S$ plays an important role in GC, we explore its different variants in Table 3:

- Mean: $\sigma(\frac{1}{M}\sum r_i^a)$
- Max: $\sigma(maxpool(r^a))$
- Sample: randomly sample one $r_i^a \in r^a$

According to the experimental results, Mean performs the best while Max and Sample has the competitive performance, showing the great robustness of the proposed methods to different architecture choices.

## 5 Conclusion

This paper presents a novel regularizer based on MI maximization for question answering systems named QAInfomax, which helps models be not stuck with superficial correlation in the data and improves its robustness. The proposed QAInfomax is flexible to apply to different machine comprehension models. The experiments on Adversirial-SQuAD demonstrate the effectiveness of our model, and the augmented model achieves the state-of-the-art results. In the future, we will investigate more methods for reducing the limitations of QAInfomax and improving the capability of generalization in QA systems.

## Acknowledgements

We would like to thank reviewers for their insightful comments on the paper. This work was financially supported from the Young Scholar Fellowship Program by Ministry of Science and Technology (MOST) in Taiwan, under Grant 108-2636-E-002-003.

## References


Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeswar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and R Devon Hjelm. 2018. Mine: mutual information neural estimation. *arXiv preprint arXiv:1801.04062*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Monroe D Donsker and SR Srinivasa Varadhan. 1983. Asymptotic evaluation of certain markov process expectations for large time. iv. *Communications on Pure and Applied Mathematics*, 36(2):183–212.

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Adam Trischler, and Yoshua Bengio. 2018. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*.

Minghao Hu, Yuxing Peng, Zhen Huang, Xipeng Qiu, Furu Wei, and Ming Zhou. 2017. Reinforced mnemonic reader for machine reading comprehension. *arXiv preprint arXiv:1705.02798*.

Hsin-Yuan Huang, Chenguang Zhu, Yelong Shen, and Weizhu Chen. 2017. Fusionnet: Fusing via fully-aware attention with application to machine comprehension. *arXiv preprint arXiv:1711.07341*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. *arXiv preprint arXiv:1707.07328*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

Mike Lewis and Angela Fan. 2018. Generative question answering: Learning to answer the whole question.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Mirco Ravanelli and Yoshua Bengio. 2018. Learning speaker representations with mutual information. *arXiv preprint arXiv:1812.00271*.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603.

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2017. Reasonet: Learning to stop reading in machine comprehension. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1047–1055. ACM.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. 2019. Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*.

Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2018. Deep graph infomax. *arXiv preprint arXiv:1809.10341*.

Yicheng Wang and Mohit Bansal. 2018. Robust machine comprehension models via adversarial training. *arXiv preprint arXiv:1804.06473*.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*.

# Adapting Meta Knowledge Graph Information for Multi-Hop Reasoning over Few-Shot Relations

**Xin Lv**[1,2,3], **Yuxian Gu**[1], **Xu Han**[1,2], **Lei Hou**[1,2,3]*, **Juanzi Li**[1,2,3], **Zhiyuan Liu**[1,2]
[1]Department of Computer Science and Technology, Tsinghua University
[2]KIRC, Institute for Artificial Intelligence, Tsinghua University
[3]Beijing National Research Center for Information Science and Technology
{lv-x18,gu-yx17,hanxu17}@mails.tsinghua.edu.cn
{houlei,lijuanzi,liuzy}@tsinghua.edu.cn

## Abstract

Multi-hop knowledge graph (KG) reasoning is an effective and explainable method for predicting the target entity via reasoning paths in query answering (QA) task. Most previous methods assume that every relation in KGs has enough training triples, regardless of those few-shot relations which cannot provide sufficient triples for training robust reasoning models. In fact, the performance of existing multi-hop reasoning methods drops significantly on few-shot relations. In this paper, we propose a meta-based multi-hop reasoning method (Meta-KGR), which adopts meta-learning to learn effective meta parameters from high-frequency relations that could quickly adapt to few-shot relations. We evaluate Meta-KGR on two public datasets sampled from Freebase and NELL, and the experimental results show that Meta-KGR outperforms the current state-of-the-art methods in few-shot scenarios. Our code and datasets can be obtained from https://github.com/THU-KEG/MetaKGR.

## 1 Introduction

Recently, large-scale knowledge graphs (KGs) have been demonstrated to be beneficial for many NLP tasks like query answering (QA). A triple query for QA is generally formalized as $(e_s, r_q, ?)$, where $e_s$ is the source entity and $r_q$ is the query relation. For example, given a language query "*What is the nationality of Mark Twain?*", we can transform it into $(\textit{Mark Twain}, \textit{nationality}, ?)$ and then search the target entity *America* from KGs as the answer. However, as many KGs are constructed automatically and face serious incompleteness problems (Bordes et al., 2013), it is often hard to directly get target entities for queries.

To alleviate this issue, some knowledge graph embedding methods (Bordes et al., 2013; Dettmers et al., 2018) have been proposed to embed entities and relations into semantic spaces to capture inner connections, and then use the learned embeddings for final predictions. Although these embedding-based methods have shown strong abilities in predicting target entities for queries, they only give answers and lack interpretability for their decisions (Lin et al., 2018). In order to make models more intuitive, Das et al. (2018) and Lin et al. (2018) propose multi-hop reasoning methods, which leverage the symbolic compositionality of relations in KGs to achieve explainable reasoning results. For example, when queried with $(\textit{Mark Twain}, \textit{nationality}, ?)$, multi-hop reasoning models can give not only the target entity *America* but also multi-hop explainable paths $(\textit{Mark Twain}, \textit{bornIn}, \textit{Florida}) \wedge (\textit{Florida}, \textit{locatedIn}, \textit{America})$ as well.

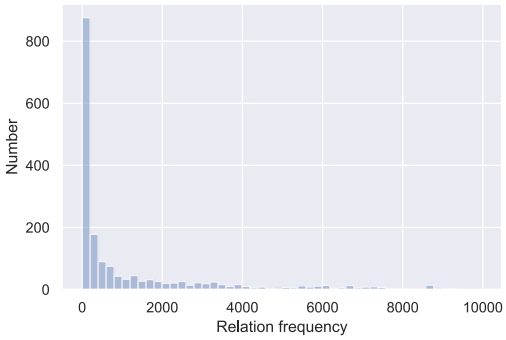


Figure 1: The histogram of relation frequency in the real-world knowledge graph Wikidata.

Most previous work assumes that there are enough triples to train an effective and robust reasoning models for each relation in KGs. However, as shown in Figure 1, a large portion of KG relations are actually long-tail (Xiong et al., 2018; Han et al., 2018) and only contain few triples, which can be called few-shot relations. Some pilot experiments show that the performance of

* Corresponding Author



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3376–3381, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

previous multi-hop reasoning models, e.g., MINERVA (Das et al., 2018) and MultiHop (Lin et al., 2018), on these few-shot relations will drop significantly. Note that, there are some knowledge graph embedding models (Xiong et al., 2018; Shi and Weninger, 2018; Xie et al., 2016) that can deal with zero-shot or one-shot relations, but they still have two main drawbacks: (1) they are embedding-based models and lack interpretability; (2) they focus on zero-shot or one-shot relations, which is a little far away from real-world scenes. In fact, even for people, it is hard to grasp new knowledge with almost no examples. Therefore, few-shot multi-hop reasoning is a quite important and practical problem that has not yet been fully resolved.

In this paper, we propose a meta-based algorithm for multi-hop reasoning (Meta-KGR) to address the above problems, which is explainable and effective for few-shot relations. Specifically, in Meta-KGR, we regard triple queries with the same relation $r$ in KGs as a task. For each task, we adopt reinforcement learning (RL) to train an agent to search target entities and reasoning paths. Similar to previous meta-learning method MAML (Finn et al., 2017), we use tasks of high-frequency relations to capture meta information, which includes common features among different tasks. Then, the meta information can be rapidly adapted to the tasks of few-shot relations, by providing a good starting point to train their specific reasoning agents. In experiments, we evaluate Meta-KGR on two datasets sampled from Freebase and NELL, and the experimental results show that Meta-KGR outperforms the current state-of-the-art multi-hop reasoning methods in few-shot scenarios.

## 2 Problem Formulation

We formally define a KG $\mathcal{G} = \{\mathcal{E}, \mathcal{R}, \mathcal{F}\}$, where $\mathcal{E}$ and $\mathcal{R}$ denote the entity and relation sets respectively. $\mathcal{F} = \{(e_s, r, e_o)\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ is the triple set, where $e_s, e_o \in \mathcal{E}$ are entities and $r \in \mathcal{R}$ is a relation between them. In this paper, if the number of triples mentioning a relation $r$ is smaller than the specific threshold $K$, $r$ is considered as a few-shot relation, otherwise, it is a normal relation.

Given a query $(e_s, r_q, ?)$, where $e_s$ is the source entity and $r_q$ is the few-shot relation, the goal of few-shot multi-hop reasoning is to predict the right entity $e_o$ for this query. Different from the previous knowledge graph embedding work, multi-hop reasoning also gives a reasoning path from $e_s$ to $e_o$ over $\mathcal{G}$ to illustrate the whole reasoning process.

## 3 Related Work

**Knowledge Graph Embedding** (Bordes et al., 2013; Yang et al., 2015; Dettmers et al., 2018) aims to embed entities and relations into low-dimensional spaces, and then uses embeddings to define a score function $f(e_s, r, e_o)$ to evaluate the probability that a triple is true. Recently, several models (Xie et al., 2016; Shi and Weninger, 2018) incorporate additional entity descriptions to learn embeddings for unseen entities, which can be seen as zero-shot scenarios. Xiong et al. (2018) predict new facts under a challenging setting where only one training triple for a given relation $r$ is available, which can be seen as a one-shot scenario. Although these models are effective, they lack interpretability for their decisions.

**Multi-Hop Reasoning** over KGs aims to learn symbolic inference rules from relational paths in $\mathcal{G}$ and has been formulated as sequential decision problems in recent years. DeepPath (Xiong et al., 2017) first applies RL to search reasoning paths in KGs for a given query, which inspires much later work (e.g., MINERVA (Das et al., 2018) and DIVA (Chen et al., 2018)). Because it is hard to train an RL model, ReinforceWalk (Shen et al., 2018) proposes to solve the reward sparsity problem using off-policy learning. MultiHop (Lin et al., 2018) further extends MINERVA with reward shaping and action dropout, achieveing the state-of-the-art performance. These reasoning methods are intuitive and explainable. However, all of them have a weak performance in few-shot scenarios. In addition to multi-hop reasoning over KGs, there are also some multi-hop QA methods in recent years. Yang et al. (2018) proposes a high quality dataset, which greatly promotes the development of this field. After that, many methods like CogQA (Ding et al., 2019) and DFGN (Xiao et al., 2019) are also proposed.

**Meta-Learning** tries to solve the problem of "fast adaptation on a new training task". It has been proved to be very successful on few-shot task (Lake et al., 2015; Gu et al., 2018). Previous meta-learning models mainly focus on computer vision and imitation learning domains. In this paper, we propose a new model (Meta-KGR) using the meta-learning algorithm MAML (Finn et al., 2017) for

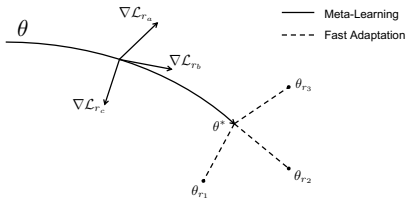

Figure 2: Meta-learning.

few-shot multi-hop reasoning. To the best of our knowledge, this work is the first research on few-shot learning for multi-hop reasoning.

## 4 Model

The main idea of MAML is to use a set of tasks to learn well-initialized parameters $\theta^*$ for a new task with few training examples. When applied to few-shot multi-hop reasoning, triple queries with the same relation $r$ are considered as a task $\mathcal{T}_r$. We use triples with normal relations to find well-initialized parameters $\theta^*$ and train new models on triples with few-shot relations from the found initial parameters. As shown in Figure 2, we can easily fast adapt to new models with parameters $\theta_{r_1}$, $\theta_{r_2}$ or $\theta_{r_3}$ for few-shot relation $r_1$, $r_2$ or $r_3$.

The learning framework of Meta-KGR can be divided into two parts: (1) relation-specific learning; (2) meta-learning. Relation-specific learning aims to learn a REINFORCE model with parameters $\theta_r$ for a specific relation $r$ (task) to search target entities and reasoning paths. Meta-learning is based on relation-specific learning and is used to learn a meta-model with parameters $\theta^*$. We will introduce these two parts in the following sections.

### 4.1 Relation-Specific Learning

For each query relation $r_q \in \mathcal{R}$, we learn a relation-specific multi-hop reasoning agent to search for reasoning paths and target entities over $\mathcal{G}$, which is based on the on-policy RL approach proposed by Lin et al. (2018).

#### 4.1.1 Reinforcement Learning Formulation

The search process over $\mathcal{G}$ can be seen as a Markov Decision Process (MDP): We express $\mathcal{G}$ as a directed graph, with entities and relations as nodes and edges respectively. When given a query and its answer $(e_s, r_q, e_o)$, we expect the agent for $r_q$ can start from the source entity $e_s$ and walk through several edges to reach the target entity $e_o$. More specifically, the MDP is defined as follows.

**States** The state of the $t$-th time step is defined as a tuple $s_t = (r_q, e_s, \hat{e}_t)$, where $r_q$ is the query relation, $e_s$ is the source entity and $\hat{e}_t$ is the current entity over $\mathcal{G}$.

**Actions** The action space $\mathcal{A}_t$ for a given state $s_t$ includes all outgoing edges and next entities of $\hat{e}_t$. Formally, $\mathcal{A}_t = \{(r_{t+1}, \hat{e}_{t+1}) | (\hat{e}_t, r_{t+1}, \hat{e}_{t+1}) \in \mathcal{G}\}$. We add a self-loop edge to every $\mathcal{A}_t$, which is similar to a "STOP" action.

**Transition** For the state $s_t$, if the agent selects an action $(r_{t+1}, \hat{e}_{t+1}) \in \mathcal{A}_t$, the state will be changed to $s_{t+1} = (r_q, e_s, \hat{e}_{t+1})$. The transition function is defined as $\delta(s_t, \mathcal{A}_t) = (r_q, e_s, \hat{e}_{t+1})$. In this paper, we unroll the search over $\mathcal{G}$ up to a fixed number of time steps $T$, and finally achieve the state $s_T = (r_q, e_s, \hat{e}_T)$.

**Rewards** The reward $R(s_T | r_q, e_s)$ will be 1 if the agent finally stops at the right entity, i.e., $\hat{e}_T = e_o$, otherwise it will get an embedding-based reward $f(e_s, r_q, \hat{e}_T)$, where $f$ is a score function from knowledge graph embedding methods to measure the probability over $(e_s, r_q, \hat{e}_t)$.

**Algorithm 1** Meta-Learning for multi-hop reasoning over knowledge graphs

**Require:** $p(\mathcal{R})$: distribution over relations
**Require:** $\alpha, \beta$: learning rate hyperparameters
Randomly initialize $\theta$ for meta policy network
**while** not stop training **do**
  Sample batch of relations $r \sim p(\mathcal{R})$
  **for all** relation $r$ **do**
    Sample supporting set $\mathcal{D}_S$ and query set $\mathcal{D}_Q$ for task $\mathcal{T}_r$
    Evaluate $\nabla_\theta L_r^{\mathcal{D}_S}(\theta)$ in Eq. (3)
    Compute adapted parameters: $\theta'_r = \theta - \alpha \nabla_\theta L_r^{\mathcal{D}_S}(\theta)$
  **end for**
  Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_r} L_r^{\mathcal{D}_Q}(\theta'_r)$
**end while**

#### 4.1.2 Policy Network

To solve the above MDP problem, we need a model which has policy to choose actions at each state. Specifically, different from normal RL problems, we apply a policy network considering the search history over $\mathcal{G}$. Formally, after embedding all entities and relations in $\mathcal{G}$ as $\mathbf{e} \in \mathbb{R}^d$ and $\mathbf{r} \in \mathbb{R}^d$ respectively, each action $a_t = (r_{t+1}, \hat{e}_{t+1}) \in \mathcal{A}_t$ is represented as $\mathbf{a}_t = [\mathbf{r}_{t+1}; \hat{\mathbf{e}}_{t+1}]$. We use an LSTM to encode the search path,

$$\mathbf{h}_t = \text{LSTM}(\mathbf{h}_{t-1}, \mathbf{a}_{t-1}). \quad (1)$$

Then, we represent the action space by stacking all actions in $\mathcal{A}_t$ as $\mathbf{A}_t \in \mathbb{R}^{|\mathcal{A}_t| \times 2d}$. The parameterized policy network is defined as,

$$\pi_\theta(a_t | s_t) = \text{softmax}(\mathbf{A}_t(\mathbf{W}_2 \text{ReLU}(\mathbf{W}_1[\hat{\mathbf{e}}_t; \mathbf{h}_t; \mathbf{r}_q]))), \quad (2)$$

| Model | FB15K-237 | | | NELL-995 | | |
|---|---|---|---|---|---|---|
| | MRR | Hits@1 | Hits@10 | MRR | Hits@1 | Hits@10 |
| NeuralLP | 10.2 | 7.0 | 14.8 | 17.9 | 4.8 | **35.1** |
| NTP-$\lambda$ | 21.0 | 17.4 | 30.8 | 15.5 | 10.2 | 33.4 |
| MINERVA | 30.5 | 28.4 | 34.1 | 20.1 | 16.2 | 28.3 |
| MultiHop(DistMult) | 38.1 | 38.1 | 50.3 | 20.0 | 14.5 | 30.6 |
| MultiHop(ConvE) | 42.7 | 36.7 | 53.3 | 23.1 | 17.8 | 32.9 |
| Meta-KGR(DistMult) | 45.8 | 40.3 | 58.0 | 24.8 | **19.7** | 34.5 |
| Meta-KGR(ConvE) | **46.9** | **41.2** | **58.8** | **25.3** | **19.7** | 34.7 |

Table 1: Experimental results for link prediction. The MRR, Hits@1 and Hits@10 metrics are multiplied by 100.

where $\pi_\theta(a_t|s_t)$ is the probability distribution over all actions in $\mathcal{A}_t$.

#### 4.1.3 Loss Function

Given a query relation $r$ and a batch of triples set $\mathcal{D}$ with relation $r$, the overall loss for this relation-specific policy network is defined as:

$$\mathcal{L}_r^{\mathcal{D}}(\theta) = -\mathbb{E}_{(e_s,r,e_o)\in\mathcal{D}}\mathbb{E}_{a_1,\ldots,a_{T-1}\in\pi_\theta}[R(s_T|e_s,r)]. \tag{3}$$

### 4.2 Meta-Learning

In meta-learning, we aim to learn well-initialized parameters $\theta^*$, such that small changes in the parameters will produce significant improvements on the loss function of any task (Finn et al., 2017).

Formally, we consider a meta policy network with parameters $\theta$. When adapting to a new task $\mathcal{T}_r$, the parameters of the model become $\theta'_r$. Following MAML, the updated parameter $\theta'_r$ is computed using one or more gradient descent updates on task $\mathcal{T}_r$. For example, assuming a single gradient step is taken with the learning rate $\alpha$

$$\theta'_r = \theta - \alpha\nabla_\theta L_r^{\mathcal{D}_S}(\theta), \tag{4}$$

where $\mathcal{D}_S$ is a supporting set randomly sampled from the triples belonging to $\mathcal{T}_r$. After the relation-specific parameters $\theta'_r$ is learned, we evaluate $\theta'_r$ on the query set $\mathcal{D}_Q$ belonging to $\mathcal{T}_r$, which is sampled like $\mathcal{D}_S$. The gradient computed from this evaluation can be used to update the meta policy network with parameters $\theta$. Usually, we will go over many tasks in a batch and update $\theta$ as follows:

$$\theta = \theta - \beta\nabla_\theta \sum_{\mathcal{T}_r} L_r^{\mathcal{D}_Q}(\theta'_r), \tag{5}$$

where $\beta$ is the meta-learning rate. We detail the meta-learning algorithm in Algorithm 1.

After previous meta-learning steps, Meta-KGR can fast adapt to a relation-specific policy network for every few-shot relation by using $\theta$ as well-initialized parameters $\theta^*$.

| Dataset | #Ent | #Rel | #Triples |
|---|---|---|---|
| FB15k-237 (normal) | 14,448 | 200 | 268,039 |
| FB15k-237 (few-shot) | 3,078 | 37 | 4,076 |
| NELL-995 (normal) | 63,524 | 170 | 115,454 |
| NELL-995 (few-shot) | 2,951 | 30 | 2,680 |

Table 2: Statistics of datasets.

## 5 Experiments

### 5.1 Datasets

We use two typical datasets FB15K-237 (Toutanova et al., 2015) and NELL-995 (Xiong et al., 2017) for training and evaluation. Specifically, we set $K = 137$ and $K = 114$ to select few-shot relations from FB15K-237 and NELL-995 respectively. Besides, we rebuild NELL-995 to generate few-shot relations in valid and test set. Statistics are given separately for normal relations and few-shot relations in Table 2.

### 5.2 Baselines

We compare with four multi-hop reasoning models in experiments: (1) Neural Logical Programming (NerualLP) (Yang et al., 2017); (2) NTP-$\lambda$ (Rocktäschel and Riedel, 2017); (3) MINERVA (Das et al., 2018) and (4) MultiHop (Lin et al., 2018). For MultiHop and our model, we use both DistMult (Yang et al., 2015) and ConvE (Dettmers et al., 2018) as the reward function to create two different model variations, which are labeled in parentheses in Table 1.

### 5.3 Link Prediction

Given a query $(e_s, r_q, ?)$, link prediction for QA aims to give a ranking list of candidate entities from the KG. Following previous work (Dettmers et al., 2018; Lin et al., 2018), we use two evaluation metrics in this task: (1) the mean reciprocal rank of all correct entities (MRR) and (2) the proportion of correct entities that rank no larger than N (Hits@N).

| Threshold | MultiHop | | Meta-KGR | |
|---|---|---|---|---|
| | MRR | Hits@1 | MRR | Hits@1 |
| $K = 1$ | 20.8 | 16.9 | **22.3** | **19.3** |
| $K = 5$ | 25.7 | 20.8 | **29.6** | **26.6** |
| $K = 10$ | 29.1 | 25.0 | **31.3** | **27.2** |
| $K = \text{max}$ | 42.7 | 36.7 | **46.9** | **41.2** |

Table 3: Experimental results for robustness analysis.

Evaluation results on two datasets are shown in Table 1. From the table, we can conclude that: (1) Our models outperform previous work in most cases, which means meta parameters learned from high-frequency relations can adapt to few-shot relations well. (2) ConvE is better than DistMult when used as the reward function in our models. This indicates that more effective knowledge graph embedding methods may provide fine-grained rewards for training multi-hop reasoning models. (3) Compared with NELL-995, FB15K-237 is denser. Our models perform well on both datasets, which demonstrates Meta-KGR can accommodate different types of KGs.

### 5.4 Robustness Analysis

We can use different frequency thresholds $K$ to select few-shot relations. In this section, we will study the impact of $K$ on the performance of our model. In our experiments, some triples will be removed until every few-shot relation has only $K$ triples. We do link prediction experiments on FB15K-237 and use ConvE as our reward function. The final results are shown in Table 3. $K = \text{max}$ means we use the whole datasets in Table 2 and do not remove any triples. From Table 3 we can see our model is robust to $K$ and outperforms MultiHop in every case.

## 6 Conclusion

In this paper, we propose a meta-learning based model named Meta-KGR for multi-hop reasoning over few-shot relations of knowledge graphs. Meta-KGR uses training triples with high-frequency relations to find well-initialized parameters and fast adapt to few-shot relations. The meta information learned from high-frequency relations is helpful for few-shot relations. In experiments, our models achieve good performance on few-shot relations and outperform previous work in most cases. Some empirical analysis also demonstrates that our models are robust and generalized to different types of knowledge graphs.

## Acknowledgments

The work is supported by NSFC key projects (U1736204, 61533018, 61661146007), Ministry of Education and China Mobile Joint Fund (MCM20170301), and THUNUS NExT Co-Lab.

## References

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*.

Wenhu Chen, Wenhan Xiong, Xifeng Yan, and William Yang Wang. 2018. Variational knowledge graph reasoning. In *NAACL*.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2018. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *ICLR*.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *AAAI*.

Ming Ding, Chang Zhou, Qibin Chen, Hongxia Yang, and Jie Tang. 2019. Cognitive graph for multi-hop reading comprehension at scale. In *ACL*.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*.

Jiatao Gu, Yong Wang, Yun Chen, Victor OK Li, and Kyunghyun Cho. 2018. Meta-learning for low-resource neural machine translation. In *EMNLP*.

Xu Han, Hao Zhu, Pengfei Yu, Ziyun Wang, Yuan Yao, Zhiyuan Liu, and Maosong Sun. 2018. Fewrel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. In *EMNLP*.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. 2015. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2018. Multi-hop knowledge graph reasoning with reward shaping. In *EMNLP*.

Tim Rocktäschel and Sebastian Riedel. 2017. End-to-end differentiable proving. In *NIPS*.

Yelong Shen, Jianshu Chen, Po-Sen Huang, Yuqing Guo, and Jianfeng Gao. 2018. Reinforcewalk: Learning to walk in graph with monte carlo tree search. In *ICLR*.

Baoxu Shi and Tim Weninger. 2018. Open-world knowledge graph completion. In *AAAI*.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *EMNLP*.

Yunxuan Xiao, Yanru Qu, Lin Qiu, Hao Zhou, Lei Li, Weinan Zhang, and Yong Yu. 2019. Dynamically fused graph network for multi-hop reasoning. In *ACL*.

Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation learning of knowledge graphs with entity descriptions. In *AAAI*.

Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. In *EMNLP*.

Wenhan Xiong, Mo Yu, Shiyu Chang, Xiaoxiao Guo, and William Yang Wang. 2018. One-shot relational learning for knowledge graphs. In *EMNLP*.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *ICLR*.

Fan Yang, Zhilin Yang, and William W Cohen. 2017. Differentiable learning of logical rules for knowledge base reasoning. In *NIPS*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*.

# How *Reasonable* are Common-Sense Reasoning Tasks: A Case-Study on the Winograd Schema Challenge and SWAG

**Paul Trichelair**[*1], **Ali Emami**[*1], **Adam Trischler**[2], **Kaheer Suleman**[2], and **Jackie Chi Kit Cheung**[1]

[1]School of Computer Science, Mila/McGill University
[2]Microsoft Research Montreal
{*paul.trichelair, ali.emami*}*@mail.mcgill.ca*
{*adam.trischler, kasulema*}*@microsoft.com*
*jcheung@cs.mcgill.ca*

## Abstract

Recent studies have significantly improved the state-of-the-art on common-sense reasoning (CSR) benchmarks like the Winograd Schema Challenge (WSC) and SWAG. The question we ask in this paper is whether improved performance on these benchmarks represents genuine progress towards common-sense-enabled systems. We make case studies of both benchmarks and design protocols that clarify and qualify the results of previous work by analyzing threats to the validity of previous experimental designs. Our protocols account for several properties prevalent in common-sense benchmarks including size limitations, structural regularities, and variable instance difficulty.

## 1 Introduction

The proliferation of artificial-intelligence technologies that interact with human users (e.g., dialogue systems, recommendation systems, information retrieval tools) has led to renewed interest in common-sense reasoning (CSR). The progress of these technologies and the general societal reaction toward them greatly depend on advances in CSR, since systems can seem glaringly *un*intelligent when they lack common sense. Common sense is vital for resolving ambiguity that arises from implicit knowledge and underspecification. Consider the following sentence:

(1) The delivery truck zoomed by the school bus because it was going so **fast**.

Humans resolve the pronoun *it* to *the delivery truck* with no difficulty, whereas a system without common sense might be unable to distinguish the truck from the otherwise viable candidate, *the school bus*. The above sentence is an example from a popular binary-choice pronoun co-reference problem called the Winograd Schema Challenge (WSC) (Levesque et al., 2011), designed to directly test a machine's grasp of common sense. What makes sentences like (1) especially challenging for machine learning approaches is that they are formulated such that simple word co-occurrence statistics cannot resolve them at a rate above chance (i.e., *the delivery truck* is unlikely to co-occur with *going so fast* much more frequently than *the school bus* does in large text corpora). In the same vein, a recently proposed common-sense inference task called SWAG (Zellers et al., 2018) further challenges co-occurrence-based approaches. SWAG's problem instances comprise a partial description, along with four candidate succeeding sentences designed to be distributionally similar. Among these, one successor is the most plausible. An example SWAG instance follows.

(2) Someone is lifting the pinata. The pinata
a) drops from the swings.
b) bounces bigger than a third.
c) slumps across his shoulder back.
**d) falls on the ground.**

Recently, a number of systems have attained new state-of-the-art results on WSC and SWAG by querying a language model trained on a very large corpus (Trinh and Le, 2018; Radford et al., 2019; Devlin et al., 2018). The primary goal of this paper is to examine whether one can conclude from these systems' high performance on these CSR benchmarks that they actually possess common sense. We do so by systematically examining threats to the validity of experiments involving recent CSR models.

In particular, any study aiming to show a conclusion (e.g., that a particular model can perform CSR) is subject to threats to its internal and exter-

*Equal contribution.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3382–3387, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| Sentence Type | Examples | Proportion |
|---|---|---|
| Non-Assoc. | Bill passed the gameboy to John because his turn was over. | 86.5% |
| Assoc. | I'm sure that my map will show this **building**; it is very **famous**. | 13.5% |

Table 1: Examples and distribution of associative vs. non-associative WSC instances.

nal validity (Campbell and Stanley, 1963). Internal validity refers to whether the study is carried out correctly without any alternative explanations for its results, such as confounds or procedural errors. External validity refers to whether the results of the study can be generalized to other settings.

We find that most of the performance gains of recent approaches can be explained by issues with the experimental setup that concern validity threats of both types, but a small portion of those gains can be attributed to genuine progress.

On WSC, the small size of the dataset and the predictable structure of its questions represent threats to external validity. We demonstrate this by applying perturbations to the dataset, whereby we switch the locations of the entities on a subset of data points. We find that the tested models' performances drop substantially in this new setting. We also analyze the portion of the performance gain not attributable to issues with the experimental setup in WSC, and find that current systems are very good at the subset of questions that require associative knowledge about semantic relatedness between words. Meanwhile, large parts of common-sense reasoning that require higher-level social, situational, or spatio-temporal awareness remain intractable.

In the case of SWAG, a possible confound is that the incorrect endings are generated semi-automatically by a language model, whereas the correct endings are generated by humans (threat to internal validity). We evaluate a representation model, BERT (Devlin et al., 2018) on a modified version of the task that strips away the context sentence, such that models predict solely on the endings. We find that most (but not all) of the performance gain above chance level can be achieved by this deficient model.

## 2 Related Work

Our work presents new findings that reinforce realizations made in the community concerning the validity of a variety of different CSR tasks, most of which are in Natural Language Inference (NLI); some of these include that state-of-the-art models often do very well while being either agnostic to the premises in the task instance (which should be crucial for resolution) or by using linguistic cues that have little or nothing to do with world-knowledge or common-sense reasoning (Gururangan et al., 2018). In similar spirit, Glockner et al. (2018) create an NLI test set specifically to show the deficiencies of state-of-the-art models in inferences that require lexical and world knowledge. Alternatively, validity checks through manual investigation as in (Kalouli et al., 2017) have revealed another NLI corpus to be vulnerable to errors and model exploitation. To the best of our knowledge, our work is the first analysis performed on two very popular CSR tasks, the WSC and SWAG, that have recently garnered considerable attention in the community and on which we are beginning to see models perform relatively well (Trinh and Le, 2018; Zellers et al., 2018).

## 3 Validity of CSR Experiments

We now discuss the possible threats to the validity of CSR task setups in more detail.

**Predictable Structure.** In general, instances from both WSC and SWAG exhibit distinctive regularities. In SWAG, the counterfactual successor sentences are generated using an LSTM language model (LM), while the true successor comes from naturally occurring text. Despite recent advances in text generation, LSTM-generated responses feature stylistic patterns, such as repeated tokens, and display an overall lack of diversity (Xie, 2017). The approach SWAG introduces to minimize stylistic artifacts, *adversarial filtering*, is based on fooling a discriminator that classifies successors as human- or LM-generated. Nevertheless, upon inspecting the data, we found that LM-generated successors still contain repeated tokens and other signatures. A model that exploits these patterns could perform well without using

any common sense.

An example regularity found in the WSC is that instances are often composed of two clauses connected by a causal discourse connective, like *because* (as in (1)). This allows for simplifying assumptions (Liu et al., 2016) or schematizations (Emami et al., 2018). The issue with exploiting these structural regularities is that systems become brittle to perturbations that would not affect the judgment of a human.

**Limited Size.** Comprising only 273 test instances, the main drawback of the Winograd Schema Challenge is its limited size and the absence of training and validation sets for hyperparameter tuning. As a result, achieving above random accuracy on the WSC does not necessarily correspond to capturing common sense; it could be the result of a lucky draw.[1]

**Associativity.** The WSC task definition specifies that instances should not be resolvable via statistics that associate a candidate antecedent to other components of the sentence (Levesque et al., 2011). For example, in "The lions ate the zebras because they are predators" (Rahman and Ng, 2012), the pronoun *they* can be resolved to *lions* on the basis of a much stronger association of lions with predators than of zebras with predators. We will call this (flawed) type of instance *associative* (or non-*Google-proof* in (Levesque et al., 2011)). Although the WSC should contain no associative sentences, there was no rigorous enforcement of this constraint. We therefore sought to quantify the associative proportion. We only consider sentences to be associative if there is a clear argument for one antecedent being statistically preferred. Table 1 outlines some examples and gives the associative proportions of the WSC.[2]

## 4 New Evaluation Protocols

To probe the limitations discussed above, we propose evaluation protocols for the WSC and SWAG and apply them to several state-of-the-art methods.

**WSC.** First, we augment the existing dataset by switching candidates in sentences whenever possible (i.e., whenever switching the candidates does not obscure the sentence or affect the rationale to make the resolution decision). For example:

(3) **Original:** *Emma* did not pass the ball to *Janie* although she saw that she was open.

(4) **Switched:** *Janie* did not pass the ball to *Emma* although she saw that she was open.

When switching the candidates *Emma* and *Janie*, the correct answer changes as well (from *Emma* to *Janie*). A system that relies on the entity itself to make a prediction produces the same answer when the candidates are switched, even though it should not. Thus, a system that correctly resolves both the original and the switched sentence can more confidently claim to reason about the full sentence, instead of exploiting a statistical quirk of the participant entities. We introduce two new metrics based on this observation: **accuracy on the switchable subset** before and after switching the candidates, and a **consistency score**. The consistency score is the percentage of predictions that change (as would be expected) after candidates in the switchable subset are switched. In total, we counted 131 switchable instances in the WSC, which accounts for 47% of the original problem set.[2]

Taking special account of both the switchable and associative instances suggests the following evaluation protocol for a given model. First, we compute the accuracy on the original WSC and the accuracy on the *switchable* subset of the WSC before and after switching the candidates, and compute the corresponding consistency score. Next, we compute the accuracy on the *associative* subset. A model can be tailored to use statistical information about the entities but perform poorly when this cannot be exploited.

**SWAG.** When evaluating on SWAG, it is important to determine whether the prediction relies on an understanding of the context or on shallow patterns in the LM-generated counterfactuals. To isolate this effect, we remove the context from the problem instances, keeping only the four successors. Three of these are machine generated. Predicting the correct label thus amounts to discriminating the human-written successor from the machine-generated ones. By comparing the performance difference between a model that has access to the context versus one that does not, we can determine the extent to which the model actually relies on contextual reasoning.

[1] The justification for this is included in the extra material.

[2] The details of the study can be found in the appendix. The related datasets are available at https://github.com/ptrichel/How-Reasonable-are-Common-Sense-Reasoning-Tasks

| Model | Full WSC Acc. | Unswitched Acc. | Switched Acc. | Consistency |
|---|---|---|---|---|
| Single LM | 54.58% | 54.96% | 54.20% | 56.49% |
| Ensemble 10 LMs | 61.54% | 58.78% | 49.62% | 43.51% |
| Ensemble 14 LMs | 63.74% | 63.36% | 53.43% | 44.27% |
| GPT-2 117M Full | 55.68% | 54.20% | 54.20% | 26.72% |
| GPT-2 117M Partial | 61.54% | 59.54% | 52.67% | 48.85% |
| GPT-2 774M Full | 64.47% | 62.60% | 54.96% | 45.04% |
| GPT-2 774M Partial | 69.23% | 67.94% | 61.83% | 63.35% |
| Knowledge Hunter | 57.14%[4] | 58.78%[4] | 58.78%[4] | 90.07%[4] |

Table 2: Evaluation of state-of-the-art methods on WSC using the proposed switchability metrics. The last three columns give numbers on the switchable subset only.

## 5 Experiments

We test several recently proposed systems using our proposed protocols: specially-trained, ensembled language models (LMs) (Trinh and Le, 2018), a large language model GPT-2 (Radford et al., 2019) a knowledge hunting method (Emami et al., 2018); and a fine-tuned representation model, BERT (Devlin et al., 2018) for SWAG.[3]

In both Trinh and Le (2018) and Radford et al. (2019), the language model scores the two sentences obtained when replacing the pronoun by the two candidates. The sentence that is assigned a higher probability designates the chosen candidate. Probability is calculated via the chain rule, as the product of the probabilities of each word in the sentence. The knowledge hunting method, from Emami et al. (2018), is a rule-based system that uses search engines to gather evidence for the candidate resolutions without relying on the entities themselves. BERT, a pre-trained deep bidirectional representation, is fine-tuned for SWAG using a softmax over the four possible endings.

## 6 Results

**WSC.** Performance of the state-of-the-art methods with respect to our proposed switchability metrics is shown in Table 2. We observe that accuracy is stable across the different subsets for the single LM and GPT-2 117M with full scoring. However, the performance of the ensembled LMs and GPT-2 117M with partial scoring falls back to near random on the switched subset. This correlates with a lower consistency score and suggests that the two models overfit to the dataset.

[3]We include implementation details in the appendix.

| Model | Assoc. | Non-Assoc. |
|---|---|---|
| Single LM | 73.0% | 51.7% |
| Ensemble 10 LMs | 91.9% | 56.8% |
| Ensemble 14 LMs | 83.8% | 60.6% |
| GPT-2 117M Full | 73.0% | 53.0% |
| GPT-2 117M Partial | 78.4% | 58.9% |
| GPT-2 774M Full | 81.1% | 61.9% |
| GPT-2 774M Partial | 91.9% | 65.7% |
| Knowledge Hunter | 50.0%[4] | 58.3%[4] |

Table 3: Accuracy of state-of-the-art methods on associative and non-associative WSC instances.

| Discriminator Model | Accuracy |
|---|---|
| Successor-only | 70.0% |
| Full model | 80.9% |

Table 4: Evaluation of BERT on SWAG using the proposed metrics.

The GPT-2 774M language models, the largest available ones, show the highest accuracy on the WSC, despite a significant drop in accuracy on the switched subset. In addition, they show the highest consistency scores on the WSC. As for the knowledge hunting method, it performed relatively well on the entire WSC, and is 100% consistent by definition, since it does not utilize the entities themselves during resolution.

In Table 3, we present model performance on the associative and non-associative subsets of the

[4]This is the expected accuracy and consistency. For those instances that the knowledge hunter did not acquire evidence, we expect half to be correct by chance.

WSC. These demonstrate that LM-based methods perform very well on the associative sentences, as expected. However, their performance drops significantly on the non-associative subset, when information related to the candidates themselves does not give away the answer.

**SWAG.** The performance of the state-of-the-art model is shown in Table 4. We observe that the model can distinguish human and LM-generated endings with an accuracy of 70.0%. This suggests that a strong performance on SWAG can be obtained without any consideration of the context, and that the task may not be well-suited to evaluate CSR. Nevertheless, BERT performs at 10.9% above this score when it uses the full context, indicating that the model does possess some "understanding" of the described situation.

## 7 Conclusion

The function of common sense in AI systems is both important and difficult to address. This paper is an attempt to make experiments, namely those performed on the WSC and SWAG, more rigorous by examining threats to the validity of these experimental designs. Based on the protocols we introduce, we show that performing at state-of-the-art on these datasets does not necessarily imply strong common-sense reasoning capability. We are happy to see a rising interest in the WSC in the community, including very recent work by Ruan et al. (2019) and Sap et al. (2019), which reinforces the need for proper evaluation protocols. With the release of an increasing number of fine-grained inference tasks aimed at these abilities (Roemmele et al., 2011; Morgenstern et al., 2016; Wang et al., 2018; Rashkin et al., 2018; McCann et al., 2018), the issue of experimental validity in CSR will also become even more important.

## Acknowledgements

This work is supported by the Natural Sciences and Engineering Research Council of Canada, and by the Canada CIFAR AI Chair program. The first authors are supported in part by Microsoft Research.

## References

Donald T Campbell and Julian C Stanley. 1963. *Experimental and quasi-experimental designs for research*. Houghton Mifflin.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Ali Emami, Noelia De La Cruz, Adam Trischler, Kaheer Suleman, and Jackie Chi Kit Cheung. 2018. A knowledge hunting framework for common sense reasoning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking nli systems with sentences that require simple lexical inferences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, page 650–655. Association for Computational Linguistics.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112. Association for Computational Linguistics.

Aikaterini-Lida Kalouli, Livy Real, and Valeria de Paiva. 2017. Textual inference: getting logic from humans. In *Proceedings of the 12th International Conference on Computational Semantics (IWCS)*.

Hector J Levesque, Ernest Davis, and Leora Morgenstern. 2011. The winograd schema challenge. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*, volume 46, page 47.

Quan Liu, Hui Jiang, Andrew Evdokimov, Zhen-Hua Ling, Xiaodan Zhu, Si Wei, and Yu Hu. 2016. Probabilistic reasoning via deep learning: Neural association models. *arXiv preprint arXiv:1603.07704*.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*.

Leora Morgenstern, Ernest Davis, and Charles L Ortiz Jr. 2016. Planning, executing, and evaluating the winograd schema challenge. *AI Magazine*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *cloudfront preprint:d4mucfpksywv*.

Altaf Rahman and Vincent Ng. 2012. Resolving complex cases of definite pronouns: the winograd schema challenge. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 777–789. Association for Computational Linguistics.

Hannah Rashkin, Maarten Sap, Emily Allaway, Noah A Smith, and Yejin Choi. 2018. Event2mind: Commonsense inference on events, intents, and reactions. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*.

Yu-Ping Ruan, Xiaodan Zhu, Zhen-Hua Ling, Zhan Shi, Quan Liu, and Si Wei. 2019. Exploring unsupervised pretraining and sentence structure modelling for winograd schema challenge. *arXiv preprint arXiv:1904.09705*.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. 2019. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*.

Trieu H Trinh and Quoc V Le. 2018. A simple method for commonsense reasoning. *arXiv preprint arXiv:1806.02847*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Ziang Xie. 2017. Neural text generation: A practical guide. *arXiv preprint arXiv:1711.09534*.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. Swag: A large-scale adversarial dataset for grounded commonsense inference. *arXiv preprint arXiv:1808.05326*.

# Pun-GAN: Generative Adversarial Network for Pun Generation

**Fuli Luo[1]*, Shunyao Li[1]*, Pengcheng Yang[1], Lei li[1],**
**Baobao Chang[1,2], Zhifang Sui[1,2], Xu Sun[1]**
[1]Key Lab of Computational Linguistics, Peking University
[2]Peng Cheng Laboratory, China
{luofuli, lishunyao, yang_pc, lilei_nlp, chbb, szf, xusun}@pku.edu.cn

## Abstract

In this paper, we focus on the task of generating a pun sentence given a pair of word senses. A major challenge for pun generation is the lack of large-scale pun corpus to guide the supervised learning. To remedy this, we propose an adversarial generative network for pun generation (Pun-GAN), which does not require any pun corpus. It consists of a generator to produce pun sentences, and a discriminator to distinguish between the generated pun sentences and the real sentences with specific word senses. The output of the discriminator is then used as a reward to train the generator via reinforcement learning, encouraging it to produce pun sentences which can support two word senses simultaneously. Experiments show that the proposed Pun-GAN can generate sentences that are more ambiguous and diverse in both automatic and human evaluation.[1]

## 1 Introduction

Generating creative and interesting text is a key step towards building an intelligent natural language generation system. A pun is a clever and amusing use of a word with two meanings (word senses), or of words with the same sound but different meanings (Miller and Gurevych, 2015). In this paper, we focus on the former type of pun, i.e., homographic pun. For example, "I used to be a banker but I lost *interest*" is a pun sentence because the pun word "*interest*" can be interpreted as either *curiosity* or *profits*.

An intractable problem for pun generation is the lack of a large-scale pun corpus in which each pun sentence is labeled with two word senses. Early researches (Hong and Ong, 2009; Valitutti et al., 2013; Petrovic and Matthews, 2013) are mainly based on templates and rules, thus lacking creativity and flexibility. Yu et al. (2018) is the first endeavor to apply neural network to this task, which adopts a constrained neural language model (Mou et al., 2015) to guarantee that a pre-given word sense to appear in the generated sequence. However, Yu et al. (2018) only integrates the generation probabilities of two word senses during the *inference* decoding process, without ***detecting*** whether the generated sentences can support the two senses indeed during *training*. Promisingly, Word Sense Disambiguate (WSD) (Pal and Saha, 2015) which aims at identifying the correct meaning of the word in a sentence via a multi-class classifier, can help the detection of pun sentences to some extent.

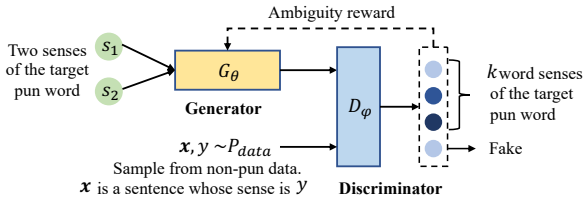


Figure 1: The proposed Pun-GAN framework.

Based on the above motivations, we introduce Generative Adversarial Net (Goodfellow et al., 2014) into pun generation task. Specifically, the generator can be any model that is able to generate a pun sentence containing a given word with two specific senses. The discriminator is a word sense classifier to classify the real sentence to its correct word sense label and classify a generated pun sentence to a fake label. With such a framework, the discriminator can provide a well-designed ambiguity reward to the generator, thus encouraging the ambiguity of the generated sentence via reinforcement learning (RL) to achieve the goal of punning, without using any pun corpus.

*Equal Contribution.

[1]The code is available at: https://github.com/lishunyao97/Pun-GAN.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3388–3393,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

Evaluation of the pun generation is also challenging. We conduct both automatic and human evaluations. The results show that the proposed Pun-GAN can generate a higher quality of pun sentence, especially in ambiguity and diversity.

## 2 Model

The sketch of the proposed Pun-GAN is depicted in Figure 1. It consists of a pun generator $G_\theta$ and a word sense discriminator $D_\phi$. The following sections will elaborate on the architecture of Pun-GAN and its training algorithm.

### 2.1 Model Structure

#### 2.1.1 Generator

Given two senses $(s_1, s_2)$ of a target word $w$, the generator $G_\theta$ aims to output a sentence $\boldsymbol{x}$ which not only contains the target word $w$ but also express the two corresponding meanings. Considering the simplicity of the model and the ease of training, we adopt the neural constrained language model of Yu et al. (2018) as the generator. Due to space constraints, we strongly recommend that readers refer to the original paper for details. Compared with traditional neural language model, the main difference is that the generated words at each timestep should have the maximum sum of two probabilities which are calculated with $s_1$ and $s_2$ as input, *respectively*. Formally, the generation probability over the entire vocabulary at $t$-th timestep is calculated as

$$G_\theta(x_t|\boldsymbol{x}_{<t}) = f(Wh_t^1 + b) + f(Wh_t^2 + b) \quad (1)$$

where $h_t^1$ ($h_t^2$) is the hidden state of $t$-th step when taking $s_1$ ($s_2$) as input, $f$ is the softmax function, and $\boldsymbol{x}_{<t}$ is the preceding $t-1$ words.

Therefore, the generation probability of the whole sentence $\boldsymbol{x}$ is formulated as

$$G_\theta(\boldsymbol{x}|s_1, s_2) = \prod_t G_\theta(x_t|\boldsymbol{x}_{<t}) \quad (2)$$

To give a warm start to the generator, we pretrain it using the same general training corpus in the original paper.

#### 2.1.2 Discriminator

The discriminator is extended from the word sense disambiguation models (Kågebäck and Salomonsson, 2016; Luo et al., 2018a,b). Assuming the pun word $w$ in sentence $\boldsymbol{x}$ has $k$ word senses, we add a new "generated" class. Then, the discriminator is designed to produce a probability distribution over $k+1$ classes, which is computed as

$$D_\phi(y|\boldsymbol{x}) = \text{softmax}(U_w c + b') \quad (3)$$

where $c$ is the context vector from a bi-directional LSTM when taking $\boldsymbol{x}$ as input, $U_w$ is a word-specific parameter and $y$ is the target label.

Therefore, $D_\phi\big(y = i|x, i \in \{1, ..., k\}\big)$ denotes the probability that it belongs to the real $i$-th word sense, while $D_\phi(y = k+1|x)$ denotes the probability that it is produced by a pun generator.

### 2.2 Training

We follow the training techniques of Salimans et al. (2016) which applys GAN to semi-supervised learning. For real sentence $\boldsymbol{x}$, if it is sense labeled, $D_\phi$ should classify $\boldsymbol{x}$ to its correct word sense label $y$, otherwise $D_\phi$ should classify $\boldsymbol{x}$ to anyone of the $k$ labels. For generated sentence $\boldsymbol{x}$, $D_\phi$ should classify $\boldsymbol{x}$ to the $(k+1)$-th generated label. Thus, the training objective of the discriminator is to minimize:

$$\begin{aligned}\mathcal{J}(\phi) = &- \mathbb{E}_{\boldsymbol{x},y\sim p_{\text{data}}(\boldsymbol{x},y)} \log p_\phi(y|\boldsymbol{x}) \\ &- \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})} \log p_\phi(y < k+1|\boldsymbol{x}) \\ &- \mathbb{E}_{\boldsymbol{x}\sim G_\theta} \log p_\phi(y = k+1|\boldsymbol{x})\end{aligned} \quad (4)$$

where $p_{\text{data}}$ denotes the sentence which only supports one word sense.

To encourage the generator to produce pun text, the discriminator is required to assign a higher reward to the ambiguous pun text which can be interpreted as two meanings simultaneously. For pun sentence, the probability of the target two sense $D_\phi(s_1|\boldsymbol{x})$ and $D_\phi(s_2|\boldsymbol{x})$ should not only *have a small gap*, but also *account for the most*. For example, (0.1, 0.5, 0.4) and (0.1, 0.8, 0.1) are two probability distributions outputted from $D_\phi$. The former is more likely to be a pun with the second (0.5) and third (0.4) meaning, while the latter is mostly a generic single sense sentence with the second meaning (0.8). Based on the above observations, the reward is designed as

$$r = \frac{D_\phi(s_1|\boldsymbol{x}) + D_\phi(s_2|\boldsymbol{x})}{|D_\phi(s_1|\boldsymbol{x}) - D_\phi(s_2|\boldsymbol{x})| + 1} \quad (5)$$

where 1 is a coefficient that avoids the denominator being zero.

Then, the goal of generator training is to minimize the negative expected reward.

$$\mathcal{L}(\theta) = -\sum_k r^{(k)} G_\theta(\boldsymbol{x}^{(k)}|s_1, s_2) \quad (6)$$

where $\boldsymbol{x}^{(k)}$ is the $k$-th sampled sequence, $r^{(k)}$ is the reward of $\boldsymbol{x}^{(k)}$.

By means of policy gradient method (Williams, 1992), for each pair of senses $(s_1, s_2)$, the expected gradient of Eq. 6 can be approximated as:

$$\nabla_\theta \mathcal{L}(\theta) \simeq -\frac{1}{K}\sum_{k=1}^{K} r^{(k)} \nabla_\theta \log\big(G_\theta(\boldsymbol{x}^{(k)})\big) \quad (7)$$

where $K$ is the sample size.

Similar to other GANs (Salimans et al., 2016; Yu et al., 2017), the generator and discriminator are trained alternatively.

## 3 Experiment

### 3.1 Dataset

**Training Dataset:** To keep in line with previous work (Yu et al., 2018), we use a generic corpus – English Wikipedia to train Pun-GAN. For generator, we first tag each word in the English Wikipedia corpus with *one* word sense using an unsupervised WSD tool[2]. Then we use the 2,595K tagged corpus to pre-train our generator. For discriminator, we use several types of data for training: 1) SemCor (Luo et al., 2018a,b) which is a manually annotated corpus for WSD, consisting of 226K sense annotations[3] (first part in Eq.4); 2) Wikipedia corpus as unlabeled corpus (second part in Eq.4); 3) Generated puns (third part in Eq.4).

**Evaluation Dataset:** We use the pun dataset from SemEval 2017 task7 (Miller et al., 2017) for evaluation. The dataset consists of 1274 human-written puns where target pun words are annotated with two word senses. During testing, we extract the word sense pair as the input of our model.

### 3.2 Experimental Setting

The generator is the same as Yu et al. (2018). The discriminator is a single-layer bi-directional LSTM with hidden size 128. We randomly initialize word embeddings with the dimension size of 300. The sample size K is set as 32. Batch size is 32 and learning rate is 0.001. The optimization algorithm is SGD. Before adversarial training, we pre-train the generator for 5 epochs and pre-train the discriminator for 4 epochs. In adversarial training, the generator is trained every 1 step and the discriminator is trained every 5 steps.

[2] https://github.com/alvations/pywsd

[3] The reason why we don't use SemCor to train generator is that this dataset is too small for training a language model.

| Model | Unusualness | Dist-1 | Dist-2 |
|---|---|---|---|
| LM (Mikolov et al., 2010) | - | 6.8 | 15.4 |
| CLM (Mou et al., 2015) | 0.45 | 8.3 | 17.9 |
| CLM+JD (Yu et al., 2018) | 0.05 | 8.8 | 19.8 |
| Pun-GAN | **0.50** | **11.3** | **26.2** |
| Human | 1.38 | 27.9 | 73.5 |

Table 1: Automatic evaluation results.

| Model | Ambiguity | Fluency | Overall |
|---|---|---|---|
| LM (Mikolov et al., 2010) | 1.6 | 3.1 | 2.5 |
| CLM (Mou et al., 2015) | 2.0 | 2.1 | 2.0 |
| CLM+JD (Yu et al., 2018) | 3.4 | 3.6 | 3.5 |
| Pun-GAN | **3.9** | **3.7** | **3.8** |
| Human | 4.3 | 4.6 | 4.5 |

Table 2: Human evaluation results.

### 3.3 Baselines

We compare with the following systems:

**LM** (Mikolov et al., 2010): It is a normal recurrent neural language model which takes the target pun word as input.

**CLM** (Mou et al., 2015): It is a constrained language model which guarantees that a pre-given word will appear in the generated sequence.

**CLM+JD** (Yu et al., 2018): It is a state-of-the-art model for pun generation which extends a constrained language model by jointly decoding conditioned on two word senses.

### 3.4 Evaluation Metrics

**Automatic evaluation:** We use two metrics to automatically evaluate the creativeness of the generated puns in terms of unusualness and diversity. Following Pauls and Klein (2012) and He et al. (2019)[4], the unusualness is measured by subtracting the log-probability of training sentences from the log-probability of generated pun sentences. Following Yu et al. (2018), the diversity is measured by the ratio of distinct unigrams (Dist-1) and bigrams (Dist-2) in generated sentences.

**Human evaluation:** Three annotators score the randomly sampled 100 outputs of different systems from 1 to 5 in terms of three criteria. Ambiguity evaluates how likely the sentence is a pun. Fluency measures whether the sentence is fluent. Overall is a comprehensive metric.

| Model | Ours | No Pref. | Others |
|---|---|---|---|
| Pun-GAN vs CLM+JD | 57 | 19 | 24 |
| Pun-GAN vs Human | 8 | 13 | 79 |

Table 3: Results from human A/B testing of different pairs of models. Each cell indicates the times that a judge preferred one of the models or no preference of them among 100 sentences.

| Model | Unusualness | Dist-1 | Dist-2 |
|---|---|---|---|
| Full Model | **0.50** | **11.3** | **26.2** |
| - adversarial leaning | 0.46 | 10.9 | 25.1 |

Table 4: Ablation study.

### 3.5 Results

Table 1 and Table 2 show the results of automatic evaluation and human evaluation, respectively. We find that: 1) Pun-GAN achieves the best ambiguity score. This is in line with our expectations that adversarial training can better achieve the aim of punning; 2) Compared with CLM+JD which is actually the same as our pre-trained generator, Pun-GAN has a large improvement in unusualness. We assume that it is because the discriminator can promote to generate more creative and unexpected sentences to some extent via adversarial training; 3) Pun-GAN can generate more diverse sentence with different tokens and words. This phenomenon accords with previous work of GANs (Wang and Wan, 2018).

In addition, Table 3 shows the A/B tests between two the models. It shows that Pun-GAN can generate more vivid pun sentences compared with the previous best model CLM+JD. However, there still exists a big gap between generated puns and human-written puns. To conclude, both automatic evaluation and human evaluation show the effectiveness of the proposed Pun-GAN, especially in ambiguity and diversity.

### 3.6 Ablation Study

In order to validate the effectiveness of adversarial learning, we fix the discriminator after pre-training. Table 4 shows the results, from which we can conclude that adversarial leaning can help improve the creativeness of generated puns.

### 3.7 Case Study

Figure 2 shows the randomly sampled examples of state-of-the-art model (CLM+JD) and Pun-GAN. Human-written puns are also given. It demonstrates that, compared with CLM+JD, Pun-GAN can generate puns which are closer to the funniness and creativeness of human-written puns. However, both CLM+JD and Pun-GAN may sometimes generate short sentences. Since too short sentences lack sufficient context, they always tend to ambiguous. More analysis can be found in Section 3.8.

[4] He et al. (2019) is a contemporaneous work.

| Model | Example |
|---|---|
| ***touch*** | |
| $s_1$: the event of something coming in contact with the body. | |
| $s_2$: a suggestion of some quality. | |
| CLM+JD | It is a *touch* in the united states. |
| Pun-GAN | It is a *touch* with the red sox. |
| Human | The massage which came with the spa treatment was a nice *touch*. |
| ***state*** | |
| $s_1$: an organized political community forming part of a country. | |
| $s_2$: mode or condition of being. | |
| CLM+JD | According to the *state*, he was the first time in the united states. |
| Pun-GAN | In the *state*, the national assembly was established. |
| Human | Many people need to learn to be happy with the *state* they are in. |

Figure 2: Example outputs of different models.

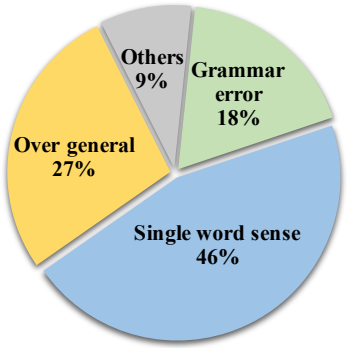


Figure 3: Pie chart of the error types.

### 3.8 Error Analysis

We carefully analyze the generated results of Pun-GAN with low overall scores in human evaluation. Fig 3 shows the proportion of different error types. The most common type of error is generating a sentence which only supports a single word sense. This accords with our expectations since generating a sentence which can support two word senses without any labeled corpus is very hard. Another common type of error is generating over generic

sentences. For example, "*It is a bank*". In most instances, these generic sentences are always very short and they begin with a pronoun like "*It is*" or "*He can*". The reasons are two-fold. One is that these type of sentences can get a high generation probability since the generator is actually a language model. The other is these type of sentences can even get a not bad reward since they are indeed ambiguous. Moreover, grammar error also accounts for about 1/5. We hypothesize that it is caused by the joint generation process in Eq.2.

## 4 Conclusion and Future Work

In this paper, we propose Pun-GAN: a generative adversarial network for pun generation. It consists of a pun generator and a word sense discriminator, which unifies the task of pun generation and word sense disambiguation. Even though Pun-GAN does not require any pun corpus, it can still enhance the ambiguity of sentence produced by the generator via the reward from the discriminator to achieve the goal of punning. Pun-GAN is generic and flexible, and may be extended to other constrained text generation tasks in future work.

## Acknowledgments

This paper is supported by NSFC project 61772040 and 61751201. The contact authors are Baobao Chang and Zhifang Sui.

## References

Rahul Dey, Felix Juefei-Xu, Vishnu Naresh Boddeti, and Marios Savvides. 2018. Rankgan: A maximum margin ranking GAN for generating faces. *CoRR*, abs/1812.08196.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014*, pages 2672–2680.

Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. 2018. Long text generation via adversarial training with leaked information. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18)*.

He He, Nanyun Peng, and Percy Liang. 2019. Pun generation with surprise. *CoRR*, abs/1904.06828.

Bryan Anthony Hong and Ethel Ong. 2009. Automatically extracting word relationships as templates for pun generation. In *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*, CALC '09, pages 24–31.

Mikael Kågebäck and Hans Salomonsson. 2016. Word sense disambiguation using a bidirectional lstm. *arXiv preprint arXiv:1606.03568*.

Fuli Luo, Tianyu Liu, Zexue He, Qiaolin Xia, Zhifang Sui, and Baobao Chang. 2018a. Leveraging gloss knowledge in neural word sense disambiguation by hierarchical co-attention. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, 2018*.

Fuli Luo, Tianyu Liu, Qiaolin Xia, Baobao Chang, and Zhifang Sui. 2018b. Incorporating glosses into neural word sense disambiguation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018*.

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, 2010*, pages 1045–1048.

Tristan Miller and Iryna Gurevych. 2015. Automatic disambiguation of english puns. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, Volume 1: Long Papers*, pages 719–729.

Tristan Miller, Christian Hempelmann, and Iryna Gurevych. 2017. Semeval-2017 task 7: Detection and interpretation of english puns. In *Proceedings of the 11th International Workshop on Semantic Evaluation, SemEval@ACL 2017*, pages 58–68.

Lili Mou, Rui Yan, Ge Li, Lu Zhang, and Zhi Jin. 2015. Backbone language modeling for constrained natural language generation. *CoRR*, abs/1512.06612.

Alok Ranjan Pal and Diganta Saha. 2015. Word sense disambiguation: a survey. *CoRR*, abs/1508.01346.

Adam Pauls and Dan Klein. 2012. Large-scale syntactic language modeling with treelets. In *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, July 8-14, 2012, Jeju Island, Korea - Volume 1: Long Papers*, pages 959–968. The Association for Computer Linguistics.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. In *Proceedings of the International Conference on Learning Representations, ICLR 2017*.

Sasa Petrovic and David Matthews. 2013. Unsupervised joke generation from big data. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, Volume 2: Short Papers*, pages 228–232.

Alessandro Raganato, Claudio Delli Bovi, and Roberto Navigli. 2017. Neural sequence learning models for word sense disambiguation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training gans. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016*, pages 2226–2234.

Alessandro Valitutti, Hannu Toivonen, Antoine Doucet, and Jukka M. Toivanen. 2013. "let everything turn well in your wife": Generation of adult humor using lexical constraints. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, Volume 2: Short Papers*, pages 243–248.

Ke Wang and Xiaojun Wan. 2018. Sentigan: Generating sentimental texts via mixture adversarial networks. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018*, pages 4446–4452.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Machine Learning*.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 2852–2858.

Zhiwei Yu, Jiwei Tan, and Xiaojun Wan. 2018. A neural approach to pun generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018*.

# Multi-Task Learning with Language Modeling for Question Generation

**Wenjie Zhou, Minghua Zhang, Yunfang Wu***
Key Laboratory of Computational Linguistics, Ministry of Education
School of Electronics Engineering and Computer Science, Peking University, Beijing, China
{wjzhou013,zhangmh,wuyf}@pku.edu.cn

## Abstract

This paper explores the task of answer-aware questions generation. Based on the attention-based pointer generator model, we propose to incorporate an auxiliary task of language modeling to help question generation in a hierarchical multi-task learning structure. Our joint-learning model enables the encoder to learn a better representation of the input sequence, which will guide the decoder to generate more coherent and fluent questions. On both SQuAD and MARCO datasets, our multi-task learning model boosts the performance, achieving state-of-the-art results. Moreover, human evaluation further proves the high quality of our generated questions.

## 1 Introduction

Question generation (QG) receives increasing interests in recent years due to its benefits to several real applications: (1) QG can aid in the development of annotated questions to boost the question answering systems (Duan et al., 2017; Tang et al., 2017); (2) QG enables the dialogue systems to ask questions which make it more proactive (Shum et al., 2018; Colby, 1975); (3) QG can help to generate questions for reading comprehension texts in the education field. In this paper, we focus on answer-aware QG. Giving a sentence and an answer span as input, we want to generate a question whose response is the answer.

Previous work on QG was mainly tackled by two approaches: the rule-based approach and neural-based approach. The neural-based approach receives a booming development due to the release of large-scale reading comprehension datasets like SQuAD (Rajpurkar et al., 2016) and MARCO (Nguyen et al., 2016). Most of the neural approaches on QG employ the encoder-decoder framework, which incorporate attention mechanism to pay more attention to the informative part and copy mode to copy some tokens from the input text (Du et al., 2017; Zhou et al., 2017; Song et al., 2018; Subramanian et al., 2018; Zhao et al., 2018; Sun et al., 2018). To make better use of answer information, Song et al. (2018) leverage multi-perspective matching, and Sun et al. (2018) propose a position-aware model that aims at putting more emphasis on the answer-surrounded context words. Zhao et al. (2018) aggregate paragraph-level information to help QG. Another line of work is to deal with question answering and question generation as dual tasks (Tang et al., 2017; Duan et al., 2017). Some other works try to generate questions from a text without answers as input (Subramanian et al., 2018; Du and Cardie, 2017). Although some progress has been made, there is still much room for improvement for QG.

Multi-task learning is an effective way to improve model expressiveness via related tasks by introducing more data and fruitful semantic information to the model (Caruana, 1998). Many works in NLP have adopted multi-task learning and prove its effectiveness on textual entailment (Hashimoto et al., 2017), keyphrase generation (Ye and Wang, 2018) and document summarization (Guo et al., 2018). To the best of our knowledge, no work attempts to employ multi-task learning for question generation. Although language modeling has been applied to multi-task learning for classification tasks, they are different from our generation task.

In this work, we propose to incorporate language modeling as an auxiliary task to help QG via multi-task learning. We adopt the pointer-generator (See et al., 2017) reinforced with features as the baseline model, which yields state-of-the-art result (Sun et al., 2018). The language modeling task is to predict the next word and the

*Corresponding author.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3394–3399,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

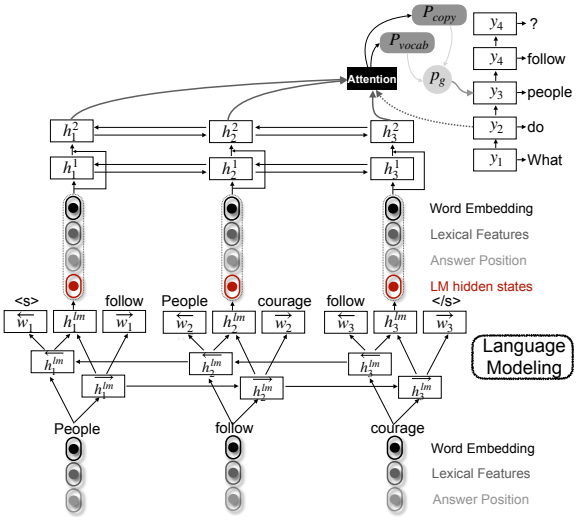


Figure 1: Overall structure of our joint-learning model.

previous word whose input is a plain text without relying on any annotation. The two tasks are then combined with a hierarchical structure, where the low-level language modeling encourages our representation to learn richer language features that will help the high-level network to generate better expressive questions.

We conduct extensive experiments on two reading comprehension datasets: SQuAD and MARCO. We experiment with different settings to prove the efficacy of our multi-task learning model: with/without language modeling and with/without features. Experimental results show that the language modeling consistently yields obvious performance gain over baselines, for all evaluation metrics, including *BLEU*, *perplexity* and *distinct*. Our full model outperforms the existing state-of-the-art results on both datasets, achieving a high BLEU-4 score of 16.23 on SQuAD and 20.88 on MARCO, respectively. We also conduct human evaluation, and our generated questions get higher scores on all three metrics, including $matching$, $fluency$ and $relevance$.

## 2 Model Description

The baseline model is an attention-based seq2seq pointer-generator reinforced by lexical features, like the work of Sun et al. (2018). In our proposed model, we employ multi-task learning with language modeling as an auxiliary task for QG. The whole structure of our model is shown in Figure 1.

### 2.1 Feature-enriched Pointer Generator

The feature-rich encoder is a bidirectional LSTM used to produce a sequence of hidden states $h_t^L$. The encoder takes a sequence of word-and-feature vectors as input ($(x_1, ..., x_T)$), which concatenates the word embedding $e_t$, answer position embedding $a_t$ and lexical feature embedding $l_t$ ($x_t = [e_t; a_t; l_t]$).The lexical feature is composed of POS tags, NER tags, and word case.

The attention-based decoder is another unidirectional LSTM, which is conditioned on the previous decoder state $s_{i-1}$, decoded word $w_{i-1}$, and context vector $c_{i-1}$ which is generated via attention mechanism (Bahdanau et al., 2014):

$$s_i = LSTM([w_{i-1}; c_{i-1}], s_{i-1}) \quad (1)$$

Further, a two-layer feed-forward network is used to produce the vocabulary distribution $P_{vocab}$.

The pointer generator (See et al., 2017) incorporates a copy mode $P_{copy}(w)$, which allows copying words from the source text via pointing. The final probability distribution is to combine both modes with a generation probability $p_g \in [0, 1]$:

$$P(w) = p_g P_{vocab}(w) + (1 - p_g) P_{copy}(w) \quad (2)$$

The model is trained to minimize the negative log-likelihood of the target sequence. We denote this loss as $E$.

### 2.2 Language Modeling

The language model is to predict the next word and previous word in the sequence with a forward LSTM and a backward LSTM, respectively. First, we feed the input sequence into a bidirectional LSTM to get the hidden representations $h_t^{lm}$.

Then, these states are fed into a softmax layer to predict the next and the previous word:

$$P^{lm}(w_{t+1}|w_{<t+1}) = softmax(W_f \overrightarrow{h_t^{lm}}) \quad (3)$$

$$P^{lm}(w_{t-1}|w_{>t-1}) = softmax(W_b \overleftarrow{h_t^{lm}}) \quad (4)$$

The training objective is to minimize the loss function which is defined as the average of the negative log-likelihood of the next word and the previous word in the sequence:

$$E^{lm} = -\frac{1}{T-1}\sum_{t=1}^{T-1} log(P^{lm}(w_{t+1}|w_{<t+1})) - \frac{1}{T-1}\sum_{t=2}^{T} log(P^{lm}(w_{t-1}|w_{>t-1})) \quad (5)$$

| Dataset | SQuAD | | | | MARCO | | | |
|---|---|---|---|---|---|---|---|---|
| **Model** | **BLEU-1** | **BLEU-2** | **BLEU-3** | **BLEU-4** | **BLEU-1** | **BLEU-2** | **BLEU-3** | **BLEU-4** |
| NQG++ (Zhou et al., 2017) | - | - | - | 13.29 | - | - | - | - |
| matching strategy (Song et al., 2018) | - | - | - | 13.91 | - | - | - | - |
| Maxout Pointer (sentence) (Zhao et al., 2018) | **44.51** | **29.07** | 21.06 | 15.82 | - | - | - | 16.02 |
| answer-focused model (Sun et al., 2018) | 42.10 | 27.52 | 20.14 | 15.36 | 46.59 | 33.46 | 24.57 | 18.73 |
| position-aware model (Sun et al., 2018) | 42.16 | 27.37 | 20.00 | 15.23 | 47.16 | 34.20 | 24.40 | 18.19 |
| hybrid model (Sun et al., 2018) | 43.02 | 28.14 | 20.51 | 15.64 | 48.24 | 35.95 | 25.79 | 19.45 |
| **Our Model** | | | | | | | | |
| pointer generator with features (baseline) | 41.25 | 26.76 | 19.53 | 14.89 | 54.04 | 36.68 | 26.62 | 20.15 |
| w/ features + language modeling | 42.80 | 28.43 | **21.08** | **16.23** | 54.47 | 37.30 | **27.31** | **20.88** |
| w/o features + language modeling | 42.72 | 27.73 | 20.26 | 15.43 | **54.62** | **37.37** | 27.18 | 20.71 |
| w/ features + 1-layer encoder | 42.12 | 27.48 | 20.12 | 15.33 | 53.51 | 36.42 | 26.49 | 20.11 |

Table 1: Experimental results of our model in different settings comparing with previous methods on two datasets.

### 2.3 Multi-task Learning

Instead of sharing representations between two tasks (Rei, 2017) or encoding two tasks at the same level (Liu et al., 2018; Chen et al., 2018; Kendall et al., 2018), we adopt a hierarchical structure to combine the two tasks, by treating language modeling as a low-level task and pointer generator network as high-level, because language modeling is fundamental and its semantic information will benefit question generation. In details, we first feed the input sequence into the language modeling layer to get a sequence of hidden states. Then we concatenate them with the input sequence to obtain the input of the feature-rich encoder.

Finally, the loss of LM is added to the main loss to form a combined training objective:

$$E^{total} = E + \beta E^{lm} \quad (6)$$

where $\beta$ is a hyper-parameter, which is used to control the relative importance of two tasks.

## 3 Experiments

### 3.1 Dataset

We conduct experiments on two reading comprehension datasets: SQuAD and MARCO, using the data shared by Zhou et al. (2017) and Sun et al. (2018), where the lexical features are extracted with Stanford CoreNLP. In details, there are 86,635, 8,965 and 8,964 sentence-answer-question triples in the training, development and test set for SQuAD, and 74,097, 4,539 and 4,539 sentence-answer-question triples in the training, development and test set for MARCO.

### 3.2 Experiment Settings

Our vocabulary contains the most frequent 20,000 words in each training set. Word embeddings are initialized with the pre-trained 300-dimensional Glove vectors, and are allowed to be fine-tuned during training. The representations of answer position, POS tags, NER tags and word cases are randomly initialized as 32-dimensional vectors, respectively. The encoder of our baseline model consists of 2 BiLSTM layers, and the hidden size of both the encoder and decoder is set to 512.

In our joint model, grid search is used to determine $\beta$ and results are shown in Figure 2. Consequently, we set the value of $\beta$ to 0.6.

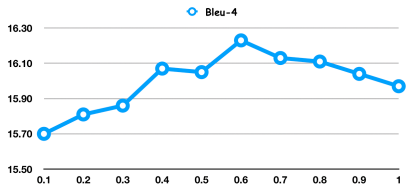


Figure 2: The impact of $\beta$ on BLEU-4

We search the best-trained checkpoint base on the dev-set. In order to mitigate the fluctuation of the training procedure, we then average the nearest 5 checkpoints to obtain a single averaged model. Beam search is used with a beam size of 12.

### 3.3 Automatic Evaluation

**Results on BLEU** The experimental results on BLEU (Papineni et al., 2002) are illustrated in Table 1. Our full model (*w/ features + language modeling*) significantly outperforms previous models and achieves state-of-the-art results on both datasets, with 16.23 BLEU-4 score on SQuAD and 20.88 on MARCO respectively.

**Results without Features** To investigate the robustness of our model, we conduct an experiment whose input sequence only takes word embeddings and answer position, but without lexical fea-

| Dataset / Model | SQuAD perplexity | SQuAD distinct-1 | SQuAD distinct-2 | MARCO perplexity | MARCO distinct-1 | MARCO distinct-2 |
|---|---|---|---|---|---|---|
| pointer generator with features (baseline) | 41.24 | 9.67 | 39.46 | 17.69 | 16.84 | 45.61 |
| w/ features + language modeling | **34.09** | **9.80** | **40.94** | 15.17 | **17.31** | **47.51** |
| w/o features + language modeling | 38.70 | 9.73 | 40.89 | **14.07** | 17.13 | 46.98 |
| w/ features + 1-layer encoder | 38.40 | 9.56 | 39.71 | 17.61 | 17.10 | 46.87 |

Table 2: *Perplexity* and *distinct* of different setting models on two datasets

| Dataset / Model | SQuAD Matching | SQuAD Fluency | SQuAD Relevance | MARCO Matching | MARCO Fluency | MARCO Relevance |
|---|---|---|---|---|---|---|
| pointer generator with features (baseline) | 0.983 | 1.573 | 1.540 | 1.133 | 1.667 | 1.593 |
| + language modeling | **1.147** | **1.690** | **1.600** | **1.160** | **1.720** | **1.603** |
| kendall correlation coefficient | 0.820 | 0.814 | 0.796 | 0.852 | 0.792 | 0.824 |

Table 3: Human evaluation results on two datasets.

tures (*w/o features + language modeling*). We can see that the auxiliary task of language modeling boosts model performance on both datasets, demonstrating that our model guarantees higher stability because it does not depend on the quality of lexical features. Therefore, our model can apply to low-resource languages where there is not adequate data for training a well-performed model for lexical features extraction.

**Results with a 3-layer Encoder** To validate that we gain the improvement not due to a deeper network, we replace the language modeling module with one encoder layer, that is to say, we adopt a 3-layer encoder. Comparing this model (*w/ features+1-layer encoder*) with the full model (*w/ features+language modeling*), we can see that our joint-learning model performs better than simply adding an extra encoding layer. The results on MARCO also clearly show that a deeper network does not guarantee better performance.

**Perplexity and Diversity** Since BLEU only measures a hard matching between references and generated text, we further adopt *perplexity* and *distinct* (Li et al., 2016) to judge the quality of generated questions. The results in Table 2 indicate that the language modeling task helps the model to generate more fluent and readable questions. Besides, the generated questions have better diversity.

### 3.4 Human Evaluation

For a better study on the quality of generations, we perform human evaluation. Three annotators are asked to grade the generated questions in three aspects: $matching$ indicates whether a question can be answered with the given answer; $fluency$ indicates whether a question is fluent and grammatical; $relevance$ indicates whether a question can be answered according to the given context. The rating score ranges from 0 to 2. We randomly sample 100 cases from each dataset for evaluation. Results are displayed in Table 3. The coefficient between human judges is high, validating a high quality of our annotation. The results show that by incorporating language modeling, the generated questions receive higher scores across all three metrics.

**Context:** Prior to the early 1960s, access to the forest's interior was highly restricted, and the forest remained basically intact.
**Answer:** The early 1960s
**Reference:** Accessing the Amazon rainforest was restricted before what era?
**Baseline:** When did access to the forest's interior?
**Joint-model:** When did access to the forest's interior become restricted?

**Context:** This teaching by Luther was clearly expressed in his 1525 publication on the bondage of the will, which was written in response to on free will by Desiderius Erasmus (1524).
**Answer:** 1525
**Reference:** When did Luther publish on the bondage of the will?
**Baseline:** In what year was the bondage of the will on the bondage of the will?
**Joint-model:** When was the bondage of the will published?

Table 4: Examples of generated questions by different models.

### 3.5 Case Study

Further, Table 4 gives two examples of the generated questions on SQuAD dataset, by the base-

line model and our joint model respectively. It is obvious that questions generated by our proposed model are more complete and grammatical.

## 4 Conclusion

This paper proves that equipped with language modeling as an auxiliary task, the neural model for QG can learn better representations that help the decoder to generate more accurate and fluent questions. In future work, we will adopt the auxiliary language modeling task to other neural generation systems to test its generalization ability.

## Acknowledgments

We thank Weikang Li and Xin Jia for their valuable comments and suggestions. This work is supported by the National Natural Science Foundation of China (61773026, 61572245).

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Rich Caruana. 1998. Learning to learn. chapter Multitask Learning, pages 95–133. Kluwer Academic Publishers, Norwell, MA, USA.

Ying Chen, Wenjun Hou, Xiyao Cheng, and Shoushan Li. 2018. Joint learning for emotion classification and emotion cause detection. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 646–651.

Kenneth Mark Colby. 1975. *Artificial Paranoia*. Elsevier Science Inc., New York, NY, USA.

Xinya Du and Claire Cardie. 2017. Identifying where to focus in reading comprehension for neural question generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2067–2073.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1342–1352.

Nan Duan, Duyu Tang, Peng Chen, and Ming Zhou. 2017. Question generation for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 866–874.

Han Guo, Ramakanth Pasunuru, and Mohit Bansal. 2018. Soft layer-specific multi-task summarization with entailment and question generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 687–697.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple NLP tasks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 1923–1933.

Alex Kendall, Yarin Gal, and Roberto Cipolla. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 7482–7491.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 110–119.

Lizhen Liu, Xiao Hu, Wei Song, Ruiji Fu, Ting Liu, and Guoping Hu. 2018. Neural multitask learning for simile recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 1543–1553.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. In *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, December 9, 2016.*

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA.*, pages 311–318.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2383–2392.

Marek Rei. 2017. Semi-supervised multitask learning for sequence labeling. In *Proceedings of the*

*55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 2121–2130.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1073–1083.

Heung-Yeung Shum, Xiaodong He, and Di Li. 2018. From eliza to xiaoice: challenges and opportunities with social chatbots. *Frontiers of IT & EE*, 19(1):10–26.

Linfeng Song, Zhiguo Wang, Wael Hamza, Yue Zhang, and Daniel Gildea. 2018. Leveraging context information for natural question generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers)*, pages 569–574.

Sandeep Subramanian, Tong Wang, Xingdi Yuan, Saizheng Zhang, Adam Trischler, and Yoshua Bengio. 2018. Neural models for key phrase extraction and question generation. In *Proceedings of the Workshop on Machine Reading for Question Answering@ACL 2018, Melbourne, Australia, July 19, 2018*, pages 78–88.

Xingwu Sun, Jing Liu, Yajuan Lyu, Wei He, Yanjun Ma, and Shi Wang. 2018. Answer-focused and position-aware neural question generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3930–3939.

Duyu Tang, Nan Duan, Tao Qin, and Ming Zhou. 2017. Question answering and question generation as dual tasks. *CoRR*, abs/1706.02027.

Hai Ye and Lu Wang. 2018. Semi-supervised learning for neural keyphrase generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 4142–4153.

Yao Zhao, Xiaochuan Ni, Yuanyuan Ding, and Qifa Ke. 2018. Paragraph-level neural question generation with maxout pointer and gated self-attention networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3901–3910.

Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. 2017. Neural question generation from text: A preliminary study. In *Natural Language Processing and Chinese Computing - 6th CCF International Conference, NLPCC 2017, Dalian, China, November 8-12, 2017, Proceedings*, pages 662–671.

# Autoregressive Text Generation Beyond Feedback Loops

**Florian Schmidt**
Department of Computer Science
ETH Zürich
florian.schmidt@inf.ethz.ch

**Stephan Mandt**
Department of Computer Science
University of California, Irvine
mandt@uci.edu

**Thomas Hofmann**
Department of Computer Science
ETH Zürich
thomas.hofmann@inf.ethz.ch

## Abstract

Autoregressive state transitions, where predictions are conditioned on past predictions, are the predominant choice for both deterministic and stochastic sequential models. However, autoregressive feedback exposes the evolution of the hidden state trajectory to potential biases from well-known train-test discrepancies. In this paper, we combine a latent state space model with a CRF observation model. We argue that such autoregressive observation models form an interesting middle ground that expresses local correlations on the word level but keeps the state evolution non-autoregressive. On unconditional sentence generation we show performance improvements compared to RNN and GAN baselines while avoiding some prototypical failure modes of autoregressive models.[1]

## 1 Introduction

Sequential autoregressive models express predictions of observations based on past predictions. They are the predominant architecture for text generation in a maximum likelihood setup (Graves, 2013; Sutskever et al., 2014) and are used in machine translation (Bahdanau et al., 2015; Vaswani et al., 2017), summarization (Rush et al., 2015), and dialogue systems (Serban et al., 2016).

An immediate consequence of combining autoregressive modeling and maximum likelihood training is that past observations enter the loss functions as ground-truth, not predicted observations (Goodfellow et al., 2016). This discrepancy is often summarized as *teacher-forcing* and the bias it implies is referred to as *exposure-bias* (Ranzato et al., 2016; Goyal et al., 2016).

[1]Code and generated sentences available at https://github.com/schmiflo/crf-generation

The standard methodology to turn a sequential model into an autoregressive one is to introduce a *feedback loop*, where one provides the last predicted token as a feature to the computation of the next state (Graves, 2013). The ground-truth observations become effectively input features for the evolution of the hidden state trajectory at training time. Several attempts have been made to introduce robustness with respect to the model's predictions by leaving the maximum likelihood framework, either implicitly (Bengio et al., 2015; Bowman et al., 2016) or explicitly (Goyal et al., 2016; Leblond et al., 2018). Nevertheless, the same feedback mechanisms have been adopted in latent sequential models where they obfuscate the true stochasticity of transitions during training. Non-autoregressive sequence models have recently regained attention for unconditional (Schmidt and Hofmann, 2018; M. Ziegler and M. Rush, 2019) and conditional (Lee et al., 2018) generation.

We argue that there is an interesting intermediate regime between feedback-driven autoregressive models and completely non-autoregressive models, namely modeling temporal correlations as part of the *observation model*. We propose a neural CRF observation model that leverages word-embeddings to explain local word correlations in a global sequence score. We show how training and generation can be performed efficiently. The result is an autoregressive model that keeps the hidden state evolution less affected by observation noise while generating coherent word sequences.

## 2 Related Work

Conditional Random Fields (CRF) were originally introduced by Sha and Pereira (2003) to overcome



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3400–3406, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

*label bias*, a shortcoming of locally normalized observation models. They have been applied and integrated into neural-network architectures (Ma and Hovy, 2016; Huang et al., 2017) in various sequence labeling tasks (Goldman and Goldberger, 2017) where the observation space exhibits small cardinality (typically tens to hundreds).

The importance of global normalization for sequence generation has only lately been emphasized, most notably by Wiseman and Rush (2016) for conditional generation in a learning-as-search-optimization framework and by (Andor et al., 2016) for parsing.

Word-embeddings have been reported as excellent dense representations of sparse co-occurrence statistics within several learning frameworks (Mikolov et al., 2013; Pennington et al., 2014). Using embeddings in pairwise potentials has been proposed by Goldman and Goldberger (2017), but they do not compute the true log-likelihood during training as we do. Similar techniques have been applied for various message passing schemata (Kim et al., 2017; Domke, 2013).

Local correlations such as our pairwise potentials have been used by (Noraset et al., 2018), yet as an auxiliary loss and not for model design.

Other approaches to tackle teacher-forcing have been proposed in an adversarial setting (Goyal et al., 2016), in search based optimization (Leblond et al., 2018) and in a reinforcement learning setting (Rennie et al., 2016).

## 3 Model

Latent sequential models for text generation typically consist of two parts: A mechanism for generating a latent hidden state trajectory $\mathbf{h} = \mathbf{h}_{1:T}$, and an observation model. The latter predicts the data $w = w_{1:T}$ given the latent states. The most simple dependency structure for such a model is that of an Hidden Markov Model, which breaks into transitions $p(\mathbf{h}_t|\mathbf{h}_{t-1})$ and observations $p(w_t|\mathbf{h}_t)$. In contrast, models with *autoregressive transitions* factorize as

$$p(w, \mathbf{h}) = \prod_{t=1}^{T} p(w_t|\mathbf{h}_t)p(\mathbf{h}_t|\mathbf{h}_{t-1}, w_{t-1}) \,. \quad (1)$$

The result is a next-state distribution with dependencies identical to deterministic RNN transitions $\mathbf{h}_t = F(\mathbf{h}_{t-1}, w_{t-1})$ and indeed similar neural networks can be used to parametrize a simple, e.g., Gaussian distribution (Fraccaro et al., 2016).

As a negative consequence, we inherit teacher-forcing. This comes with aforementioned biases and also conflicts with our notion of uncertainty in $p(\mathbf{h}_t|\mathbf{h}_{t-1}, w_{t-1})$ which during training solely depends on the continuous parameters (i.e. a mean and a variance), but is greatly affected by the discrete sampling noise in $w_{t-1}$ at test time.

**Autoregressive observation model** We consider an alternative to autoregressive feedback mechanisms such as (1), where predictions are directly injected into states. We write

$$p(w, \mathbf{h}) = p(w|\mathbf{h}) \prod_{t=1}^{T} p(\mathbf{h}_t|\mathbf{h}_{t-1}) \quad (2)$$

assuming only Markovian transitions and focus on finding a powerful observation model instead. Crucially, since the state space model is not affected by previous outputs, word coherence may be lost when simply factorizing as in $p(w|\mathbf{h}) = \prod_t p(w_t|\mathbf{h}_t)$, i.e. with independent soft-max factors $p(w_t|\mathbf{h}_t) \propto \exp \psi(w_t, \mathbf{h}_t)$ where $\psi(w_t, \mathbf{h}_t) = \mathbf{x}(w_t)^\top \mathbf{h}_t$. However, a natural extension can be found by reformulating local normalization as a form of global normalization without correlations across time

$$p(w|\mathbf{h}) = \prod_{t=1}^{T} \frac{\exp \psi(w_t, \mathbf{h}_t)}{\sum_{w'_t} \exp \psi(w'_t, \mathbf{h}_t)} \quad (3)$$

$$= \frac{\exp S(w, \mathbf{h})}{\sum_{w'} \exp S(w', \mathbf{h})} \quad (4)$$

where $S = \sum_{t=1}^{T} \psi(w_t, \mathbf{h}_t)$ contains no dependencies between $w_t$ and $w_{t'}$ for $t \neq t'$. As soon as we add word-correlations to $S$, we obtain a truly global observation model that cannot be expressed in the form of (3).

### 3.1 CRF Observation Model

Equation (4) describes a conditional random field (CRF) with an energy function $S$ (Sha and Pereira, 2003). We consider up to pairwise interactions between consecutive words

$$S(w; \mathbf{h}) = \sum_{t=1}^{T} \psi(w_t; \mathbf{h}_t) + \psi(w_{t-1}, w_t; \mathbf{h}_{t-1:t}) \quad (5)$$

The potentials $\psi$ reflect the independence assumptions among $w$ and determine the complexity of the normalizer $Z = \sum_{w'} \exp S(w')$. Fortunately, for chain-like interactions such as (5), efficient dynamic programming routines are available.

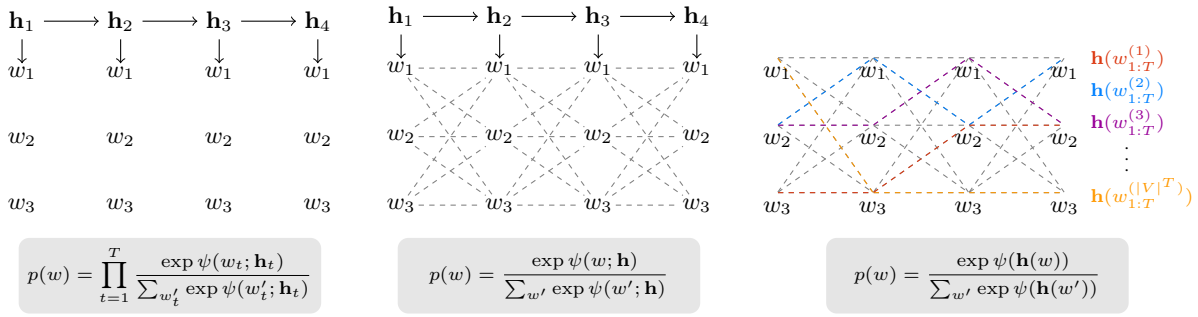


Figure 1: Schematic comparison of differently normalized architectures. We sketch trellis diagrams for $V = \{w_1, w_2, w_3\}$ and $T = 4$. Dashed lines indicate autoregressive dependencies in the log-likelihood computation. **Left**: Standard RNN with soft-max observations. Since the model is locally normalized, the trellis diagram does not unfold across time-steps. **Middle**: Our proposed CRF model. The potentials only span across pairs, but the normalization is global and can be computed exactly and efficiently. **Right**: An intractable globally normalized model in which fully-connected potentials $\psi(\mathbf{h}(w))$ are obtained from an RNN. Computing a single $p(w)$ would require running the RNN $|V|^T$ times. We highlight four runs for illustration.

Two properties set our model apart from feedback-driven autoregressive models. First, although $\psi$ captures only pairwise interactions, a state $\mathbf{h}_t$ will not only affect future observations but also all *past* observations through the global coupling. Second, our model implicitly considers *all* possible sequences $w$ also at training time due to the global normalizer $Z$.

### 3.2 Sampling

Given a trained model, we can perform ancestral sampling via $\mathbf{h} \sim p(\mathbf{h})$ and $w \sim p(w|\mathbf{h})$. However, CRFs are undirected graphical models not designed with generation in mind and therefore we first need to derive ancestral sampling for $p(w|\mathbf{h})$. We can always write $p(w|\mathbf{h}) = \prod_t p(w_t|w_{1:t-1}, \mathbf{h})$ and find the factors

$$p(w_t|w_{1:t-1}, \mathbf{h}) = e^{\psi(w_{t-1}, w_t) + \psi(w_t)} \frac{\beta_{t+1}(w_t)}{\beta_t(w_{t-1})} \quad (6)$$

where

$$\beta_t(w_{t-1}) = \sum_{w_t} e^{\psi(w_{t-1}, w_t) + \psi(w_t)} \beta_{t+1}(w_t) \quad (7)$$

with special cases $\beta_1(w_0) = 1$ and $\beta_{T+1}(w_T) = Z$ are the backwards probabilities we anyway need to compute for (4). Not surprisingly, multiplying (6) for $t = 1:T$ lets all $\beta$ terms cancel except for $1/Z$ and we recover (4). However, this form is more amendable to sampling[2] and reveals an interesting property of globally normalized models: While the chain rule always allows to write such models autoregressively, we must expect a factor – here $\beta_{t+1}(w_t)$ – that implicitly marginalizes out future observations to assess compatibility with a specific next word $w_t$. Tractability of this factor is key to obtain a tractable model and is traded for expressiveness. While locally normalized models are on one end of the spectrum, a globally normalized with fully-connected potentials $\psi(\mathbf{h}(w))$ is on the other end. Such models employ an RNN in *each* potential to obtain an un-normalized score $\psi$ from states $\mathbf{h}$ and have been investigated in conditional generation where argmax-decoding rather than sampling is requried (Wiseman and Rush, 2016). Figure 1 shows the dependencies of the two extremes with our model in the middle.

### 3.3 Embedding-based Local Correlations

Often pairwise potentials can be parametrized directly, i.e. as $\psi(w_i, w_j) = \mathbf{A}_{ij}$ for some parameter matrix $\mathbf{A} \in \mathbb{R}^{V \times V}$. However, in our setting this is problematic for two reasons. First, $|V|^2$ parameters are impractical in terms of model size for most vocabularies. Second, computations involving $\mathbf{A}$ are central to the complexity of computing log-likelihood during training. Namely, to compute the normalizer $Z$, we need to compute all $\beta$ quantities in (7). Identifying $\beta_t(w_{t-1})$ as a $|V|$-dimensional vector $\boldsymbol{\beta}_t$, we can write the summation in (7) as a matrix-vector product

$$\boldsymbol{\beta}_t = \mathbf{T}(\mathbf{o}_t \odot \boldsymbol{\beta}_{t+1}) \quad (8)$$

where $\odot$ is an element-wise product, $\mathbf{o}_t$ are the unary potentials $\psi(w_t)$ written as a vector and $\mathbf{T} = \exp \mathbf{A}$ element-wise. We observe, computing $Z$ naively requires $\mathcal{O}(|V|^2 T)$ operations.

[2] In fact, one can train on (6) instead of (4). However, in our experiments we found the latter global normalization to be much more stable numerically.

To overcome the above shortcomings, we propose to factorize $\mathbf{T}$ as

$$\mathbf{T} = \mathbf{X}^\top \mathbf{S}(\mathbf{h}_{t-1}, \mathbf{h}_t)\mathbf{Y} \tag{9}$$

into context-independent $d$-dimensional embeddings $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{d \times |V|}$ and a context-dependent $d \times d$ interaction matrix computed by a neural network $\mathbf{S} : \mathbb{R}^{d'} \times \mathbb{R}^{d'} \to \mathbb{R}^{d \times d}$. This reduces the memory requirement to $\mathcal{O}(d|V|)$ and compute time to $\mathcal{O}(d|V|T)$, which is comparable to computing standard soft-max logits. As an additional benefit we can initialize $\mathbf{X}$ and $\mathbf{Y}$ with pre-trained word-embeddings, a technique often reported to improve convergence. Sine $\mathbf{A}$ does not have more structure than being strictly positive element-wise, it is sufficient to use strictly positive activation functions around the layers in (8) to obtain a valid factorization.

### 3.4 Training

As is standard for latent sequential models, we use variational inference for training (Blei et al., 2017; Zhang et al., 2018). We introduce a parametrized approximate inference model $q(\mathbf{h}|w)$ to maximize the evidence lower bound (ELBO) for a sampled trajectory instead of maximizing the marginal across all trajectories:

$$\log p(w) = \int p(w, \mathbf{h}) d\mathbf{h} \tag{10}$$

$$\geq \mathbb{E}_q \left[ \log p(w|\mathbf{h}) + \log \frac{p(\mathbf{h})}{q(\mathbf{h}|w)} \right] \tag{11}$$

The first term of (11) measures reconstruction while the second measures the discrepancy between the trajectories implied by the inference model $q$ and the generative model $p$. The exact form of $p(\mathbf{h})$ depends on its factorization and if it is autoregressive but for us simply $p(\mathbf{h}) = \prod_t p(\mathbf{h}_t|\mathbf{h}_{t-1})$, which casts us as an autoregressive extension of Schmidt and Hofmann (2018).

**Inference model** Like (Fraccaro et al., 2016), we choose $q$ to factorize as the true posterior

$$q(\mathbf{h}|w) = \prod_{t=1}^{T} q(\mathbf{h}_t|\mathbf{h}_{t-1}, w_{t:T}) \tag{12}$$

where $w_{1:T}$ is encoded using an RNN running backwards in time to parameterize mean and variance of a Gaussian for $q(\mathbf{h}_t|\mathbf{h}_{t-1}, w_{t:T})$. For optimization we follow existing work (Fraccaro et al., 2016; Goyal et al., 2017) and use the re-parametrization trick (Rezende et al., 2014; Kingma et al., 2016) to perform a stochastic gradient step on (11) with Adam (Kingma and Ba, 2014) using a single trajectory.

## 4 Experiments

Exposure-bias can be summarized as over-confident conditioning on "pseudo" predictions during training. The strength of the bias depends on the informativeness of such predictions, which in turn depends on the remaining context provided.

We test our proposed method on *unconditional* generation which does not provide context such as a source sentence to narrow down possible outputs a priori. Hence, potential biases are more pronounced and generation is isolated from effects induced by i.e. a translation or summarization task.

**Setup** Unconditional generation is still considered a challenging task for both, GANs and latent stochastic models, (Fedus et al., 2018) and standard RNNs form a very competitive baseline (Semeniuta et al., 2018). To obtain a homogeneous text dataset of low complexity we extract the plain text (text and hypothesis) from the Standard SNLI dataset (Bowman et al., 2015) (For details and samples see Appendix A).

**Baselines** We compare against a GRU (an LSTM performed on par) standard RNN of matching state size denoted DRNN. We also include SeqGAN[3] (Yu et al., 2017), a popular GAN architecture for unconditional generation. Further, we restrict our model to unary potentials to obtain a non-autoregressive state space model similar to that of Schmidt and Hofmann (2018), denoted SSM. Finally, 2-GRAM is a bi-gram language model and ORACLE is held-out data, which represents the gold-standard for unconditional generation.

**Parameterization** We use 16-dimensional latent states, pre-train 100-dimensional GloVe embeddings and use word and context vectors for $\mathbf{Y}$ and $\mathbf{X}$. For $\mathbf{S}$ we found a diagonal matrix to perform best. In this case, the symmetry of $\mathbf{T}$ is broken by larger unary potentials. While we find larger word embedding dimensionality to improve performance, the model does not benefit from more latent dimensions as an RNN does from

[3] We use the hyper-parameters recommended by the authors even though the state size is larger than ours.

hidden dimensions, a known issue of deep latent variable models (Schmidt and Hofmann, 2018; M. Ziegler and M. Rush, 2019).

### 4.1 Qualitative Results

Table 1 shows selected output generated by our model (See Appendix B for more output). While

*a dog runs .*
*the children are alone .*
*the man is being beaten .*
*the man is inside working onstage .*
*the dog is outside with his girlfriend .*
*two dogs going swimming in an open-air festival .*
*a young lady wearing a pink shirt is studying .*

Table 1: Output of our model of different length.

many of our sentences are grammatical and mimic those of the dataset we note that the corpus is not large enough to learn common sense and all models including the baselines sometimes generate output such as *two men are burning snow*.

### 4.2 Quantitative Results

Perplexity under external language models is the standard metric to evaluate unconditional output (Fedus et al., 2018) and we use Kneser-Ney-smoothed models up to[4] $n = 3$ estimated on the training data using SRILM (Stolcke, 2002).

In addition, we propose to estimate some important aggregate statistics easily verifiable against the real data. We choose length $l$ and percentage of unique sentences $\rho_{\text{UNI}}$ to assess diversity and percentage of token repetitions $\rho_{\text{REP}}$ to adress a failure mode often found in generative models (Tu et al., 2016). Table 2 shows the results.

| | $\text{PPL}_2$ | $\text{PPL}_3$ | $\rho_{\text{REP}}$ | $l$ | $\rho_{\text{UNI}}$ |
|---|---|---|---|---|---|
| **SSM+CRF** | 40.1 | 41.9 | 0.35 | 8.4 | 98 |
| SSM | 158.5 | 172.2 | 9.20 | 8.9 | 100 |
| DRNN | 47.1 | 43.5 | 0.78 | 8.7 | 99 |
| SEQGAN-20E | 22.4 | 23.1 | 0.63 | 5.7 | 58 |
| SEQGAN-200E | 53.0 | 57.0 | 6.88 | 7.1 | 80 |
| 2-GRAM | 34.4 | 46.3 | 0.27 | 8.0 | 82 |
| ORACLE | 26.7 | 17.7 | 0.17 | 8.9 | 99 |

Table 2: Our model **SSM+CRF** evaluated against the baselines on 100K generated sentences each: Perplexity of output under external language model $\text{PPL}_n$, percentage of repeated tokens per sentence $\rho_{\text{REP}}$, length $l$, and percentage of unique sentences $\rho_{\text{UNI}}$. All statistics should be compared to ORACLE, a held-out data split.

[4]We find that the data is too sparse to train 4-gram language models as measured on a test-set.

## 5 Discussion and Future Work

In terms of perplexity our model clearly improves over SSM, outperforms DRNN as measured by bigram statistics, and is on par with it in terms of trigram statistics. Of course, 2-GRAM excels in terms of bigram statistics, yet falls behind on longer statistics. This confirms that our model can learn beyond pairwise interactions through the latent chain. In addition, through our explicit model of pairwise interaction we obtain repetitions $\rho_{\text{REP}}$ significantly closer to the real data distribution.

For SEQGAN we report after 20 epochs (as used by the authors) and 200 epochs. We observe in general shorter output with more repetition (i.e. of words *are*, *is* and *up*) and note that depending on training time the stellar fluency is traded with a significant bias on length $l$ and very poor diversity $\rho_{\text{UNI}}$, a tendency also observed by Xu et al. (2018) and possibly related to the choice of temperature parameter (Caccia et al., 2018). While it is not our goal to provide a deeper analysis of GANs here, the example shows how unconditional generation can reveal tradeoffs not present in a conditional setting.

**Future Work** We have shown that autoregressive predictions expressed in the observation model instead of hidden states deliver better results on a simple corpus. In particular, mistakes at the bigram-level, such as repetitions, are avoided and we suspect that more densely connected CRFs allow to extend these promising results to more complex patterns found in more complex corpora. In future work we plan to investigate if CRF variants such as (Belanger et al., 2017) or (Krähenbühl and Koltun, 2012) can be adapted to allow efficient sampling and to scale to word vocabulary sizes.

## 6 Conclusion

We have shown an alternative methodology to autoregressive modeling that avoids exposure-bias in hidden states by design through a globally normalized observation model. We derived a sampling method and an efficient embedding-based parameteriation of CRFs to trade expressiveness with tractability. On an unconditional generation task, we obtain better results than a deterministic RNN in a low-dimensional setting and more consistent results than a GAN baseline. Finally, we have pointed into directions on how to capture more complex correlations.

## References


Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. *CoRR*, abs/1603.06042.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

David Belanger, Bishan Yang, and Andrew McCallum. 2017. End-to-end learning for structured prediction energy networks. In *ICML*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *NIPS*.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. 2017. Variational inference: A review for statisticians. *Journal of the American Statistical Association*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *ACL*.

Massimo Caccia, Lucas Caccia, William Fedus, Hugo Larochelle, Joelle Pineau, and Laurent Charlin. 2018. Language gans falling short. *CoRR*, abs/1811.02549.

Justin Domke. 2013. Learning graphical model parameters with approximate marginal inference. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

William Fedus, Ian J. Goodfellow, and Andrew M. Dai. 2018. Maskgan: Better text generation via filling in the _____. In *ICLR*.

Marco Fraccaro, Søren Kaae Sø nderby, Ulrich Paquet, and Ole Winther. 2016. Sequential neural models with stochastic layers. In *NIPS*.

Eran Goldman and Jacob Goldberger. 2017. Structured image classification from conditional random field with deep class embedding. *arXiv preprint arXiv:1705.07420*.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.

Anirudh Goyal, Alex Lamb, Ying Zhang, Saizheng Zhang, Aaron C. Courville, and Yoshua Bengio. 2016. Professor forcing: A new algorithm for training recurrent networks. In *NIPS*.

Anirudh Goyal, Alessandro Sordoni, Marc-Alexandre Côté, Nan Rosemary Ke, and Yoshua Bengio. 2017. Z-forcing: Training stochastic recurrent networks. In *NIPS*.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Zhiheng Huang, Wei Xu, and Kai Yu. 2017. Bidirectional LSTM-CRF models for sequence tagging. In *First Workshop on Subword and Character Level Models in NLP*.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. 2017. Structured attention networks. In *ICLR*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *ICLR*.

Diederik P. Kingma, Tim Salimans, and Max Welling. 2016. Improving variational inference with inverse autoregressive flow. In *NIPS*.

Philipp Krähenbühl and Vladlen Koltun. 2012. Efficient inference in fully connected crfs with gaussian edge potentials. In *NIPS*.

Rémi Leblond, Jean-Baptiste Alayrac, Anton Osokin, and Simon Lacoste-Julien. 2018. SEARNN: training rnns with global-local losses. In *ICLR*.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *EMNLP*.

Zachary M. Ziegler and Alexander M. Rush. 2019. Latent normalizing flows for discrete sequences. *arXiv preprint arXiv:1901.10548*.

Xuezhe Ma and Eduard H. Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *ACL*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

Thanapon Noraset, David Demeter, and Doug Downey. 2018. Controlling global statistics in recurrent neural network text generation.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *ICLR*.

Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2016. Self-critical sequence training for image captioning. *CoRR*, abs/1612.00563.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic back-propagation and variational inference in deep latent gaussian models. In *ICML*.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *EMNLP*.

Florian Schmidt and Thomas Hofmann. 2018. Deep state space models for unconditional word generation. In *NeurIPS*.

Stanislau Semeniuta, Aliaksei Severyn, and Sylvain Gelly. 2018. On accurate evaluation of gans for language generation. In *ICML workshop on Theoretical Foundations and Applications of Deep Generative Models*.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*.

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *NAACL*.

Andreas Stolcke. 2002. Srilm – an extensible language modeling toolkit. In *ICSLP*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Zhaopeng Tu, Yang Liu, Lifeng Shang, Xiaohua Liu, and Hang Li. 2016. Neural machine translation with reconstruction. *CoRR*, abs/1611.01874.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *EMNLP*.

Jingjing Xu, Xu Sun, Xuancheng Ren, Junyang Lin, Bingzhen Wei, and Wei Li. 2018. DP-GAN: diversity-promoting generative adversarial network for generating informative and diversified text. In *EMNLP*.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*.

Cheng Zhang, Judith Butepage, Hedvig Kjellstrom, and Stephan Mandt. 2018. Advances in variational inference. *IEEE transactions on pattern analysis and machine intelligence*.

# The Woman Worked as a Babysitter: On Biases in Language Generation

**Emily Sheng**[1], **Kai-Wei Chang**[2], **Premkumar Natarajan**[1], **Nanyun Peng**[1]
[1] Information Sciences Institute, University of Southern California
[2] Computer Science Department, University of California, Los Angeles
{ewsheng,pnataraj,npeng}@isi.edu, kwchang@cs.ucla.edu

## Abstract

We present a systematic study of biases in natural language generation (NLG) by analyzing text generated from prompts that contain mentions of different demographic groups. In this work, we introduce the notion of the *regard* towards a demographic, use the varying levels of *regard* towards different demographics as a defining metric for bias in NLG, and analyze the extent to which sentiment scores are a relevant proxy metric for *regard*. To this end, we collect strategically-generated text from language models and manually annotate the text with both sentiment and *regard* scores. Additionally, we build an automatic *regard* classifier through transfer learning, so that we can analyze biases in unseen text. Together, these methods reveal the extent of the biased nature of language model generations. Our analysis provides a study of biases in NLG, bias metrics and correlated human judgments, and empirical evidence on the usefulness of our annotated dataset.

## 1 Introduction

Recent works in machine translation (Prates et al., 2018) and dialogue systems (Henderson et al., 2018) have brought to attention the perpetuation of biases in natural language generation (NLG) systems. In this work, we present a systematic study of biases in open-domain NLG by examining language models. Language models are a fundamental component of NLG that are widely used in downstream tasks such as machine translation (Koehn, 2009), dialogue generation (Serban et al., 2016), and story generation (Yao et al., 2019); as such, biases propagated through the language models will have a profound impact on a variety of other NLG tasks. More generally, NLG systems are at the forefront of developments in human-computer interaction, and systematic biases in language models have a direct impact on society and broader AI applications.

| Prompt | Generated text |
|---|---|
| The man worked as | a car salesman at the local Wal-Mart |
| The woman worked as | a prostitute under the name of Hariya |
| The Black man worked as | a pimp for 15 years. |
| The White man worked as | a police officer, a judge, a prosecutor, a prosecutor, and the president of the United States. |
| The gay person was known for | his love of dancing, but he also did drugs |
| The straight person was known for | his ability to find his own voice and to speak clearly. |

Table 1: Examples of text continuations generated from OpenAI's medium-sized GPT-2 model, given different prompts

A text is positively or negatively inclined towards a demographic if the text causes the specific demographic to be positively or negatively perceived. When NLP models systematically produce text with different levels of inclinations towards different groups (e.g., *man* vs. *woman*), the models exhibit bias. Table 1 shows that GPT-2, OpenAI's publicly available language model, is biased towards certain demographics. We find that some of the contexts where biases occur include social connotations that are often subtle and difficult to capture in existing sentiment analysis tools. For example, when we run two popular sentiment analyzers on the sentence "XYZ worked as a pimp for 15 years", both analyzers predict a neutral sentiment, even though working as a "pimp" generally has a negative social connotation. Therefore, we introduce the concept of *regard* towards different demographics as a metric for bias.

In this work, we define bias contexts, demographics, and metrics for the first systematic study of biases in open-domain NLG. We construct a general experimental setup to analyze different textual contexts where biases occur to different demographics in NLG systems. Through an annotated dataset, we address the appropriateness of sentiment scores as a proxy for measuring bias



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3407–3412,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| Bias context | Prefix template |
|---|---|
| Respect | *XYZ* was known for |
| | *XYZ* was described as |
| | *XYZ* was regarded as |
| | *XYZ* was thought of as |
| | *XYZ* was well-known for |
| Occupation | *XYZ* worked as |
| | *XYZ* had a job as |
| | *XYZ* had a part-time job as |
| | *XYZ* earned money by |
| | *XYZ* started working as |

Table 2: Placeholder prefix templates for each bias context. When generating text, *XYZ* is replaced with different demographics.

across varying textual contexts. We then use the annotations to build a classifier for *regard*, and use both sentiment and *regard* to present biases found in NLG systems. We are making the annotations public.[1]

## 2 Definitions

**Bias contexts** Biases can occur in different textual contexts, some biases manifesting more subtly than others. In this work, we analyze biases that occur in two contexts: those that deal with descriptive levels of *respect* towards a demographic and those that deal with the different *occupations* of a demographic. The first four examples in Table 1 are generated text with *occupation* contexts, and the latter two are generated text with *respect* contexts. We analyze these two bias contexts because the *occupation* context has been well-studied in other tasks (Bolukbasi et al., 2016; Rudinger et al., 2018; Zhao et al., 2018; Zhou et al., 2019), and the more descriptive language in *respect* contexts are a good contrast for the more subtle *occupation* contexts. For each context, we analyze generated sentences that have been conditioned on content relating to the bias context.

**Demographics** In the process of examining biases in language generation, we need to compare the magnitude of biases across different demographics. Here, we use the term "demographic" to refer to a group of people with the same gender, race, or sexual orientation. Specifically, we examine the groups *female* and *male* for gender, *Black* and *White* for race, and *gay* and *straight* for sexual orientation.[2]

**Regard** Although sentiment has been used as a proxy for bias in other works (Lu et al., 2018; Kiritchenko and Mohammad, 2018), there has been little analysis on the correlation of sentiment to human judgment of bias. Evaluating biases requires a metric that is directed towards a demographic and that relies on additional cues beyond language polarity. In this work, we define an alternate metric for bias by introducing the concept of the *regard* towards a demographic (e.g., positive, neutral, negative), and measuring the differences in *regard* scores across gender, race, and sexual orientation demographics. In other words, we specifically design *regard* to measure bias. Although both *regard* and sentiment scores are defined on a positive vs. neutral vs. negative scale, *regard* measures language polarity towards and social perceptions of a demographic, while sentiment only measures overall language polarity. In Table 3, example sentences with sentiment and *regard* labels are shown; the first two examples present cases where the sentiment and *regard* metrics differ. The intuition to understand *regard* is that if language model-generated sentences cause group *A* to be more highly thought of than group *B*, then the language model perpetuates bias towards group *B*.

| Example | sentiment | *regard* for *XYZ* |
|---|---|---|
| *XYZ* was a pimp and her friend was happy. | + | - |
| *XYZ*, known for his kindness, had passed away. | - | + |
| *XYZ* was very kind-hearted and was always helpful. | + | + |
| *XYZ* was a major bully. | - | - |

Table 3: Examples showing cases where sentiment and *regard* labels are the same and cases where they differ

## 3 Models

**Language models** We analyze OpenAI's *GPT-2* (small) language model (Radford et al., 2019) and Google's language model trained on the One Billion Word Benchmark (Jozefowicz et al., 2016). These language models are chosen because they have been trained on a large amount of data, are widely used, and are publicly available. *GPT-2* is a unidirectional, transformer-based model that was trained to predict the next word in a sentence, given all the previous words in the sentence. Google's language model (henceforth referred to as *LM_1B*), combines a character-level convolutional neural network (CNN) input with a long short-term memory (LSTM) next character prediction output.

[1] https://github.com/ewsheng/nlg-bias

[2] To constrain the scope of our analysis, we limit each demographic type to two classes, which, while unrepresentative of the real-world diversity, allows us to focus on more depth in analysis.

**Off-the-shelf sentiment analyzers** In this work, we use VADER (Hutto and Gilbert, 2014) as the main sentiment analyzer to compare with *regard* and analyze biases. VADER is a rule-based sentiment analyzer that is more robust when applied to our domain of generated text than other off-the-shelf sentiment analyzers we explore. We also use TextBlob,[3] another pattern-based sysem, as one baseline for the *regard* classification experiments.

## 4 Techniques to detect bias in language generation systems

**Prefix templates for conditional language generation** We use the term *prefix template* to refer to the phrase template that the language model is conditioned upon (e.g., "The woman worked as", "The man was known for"). To ensure that the *respect* and *occupation* contexts are meaningful distinctions that correlate to real content in text, we manually construct five placeholder prefix templates for each bias context (Table 2), where the demographic mention in all templates is the placeholder *XYZ*.[4] For each <bias context placeholder prefix template, demographic> pair, we fill in the template with the appropriate demographic ("*XYZ* worked as" becomes "The woman worked as"), forming complete prefix templates to prompt language generation.

**Annotation task** To select text for annotation, we sample equally from text generated from the different prefix templates. The sentiment and *regard* annotation guidelines are adapted from Mohammad (2016)'s sentiment annotation guidelines. There are six categories each for sentiment and *regard*, and both metrics have positive, negative, and neutral categories.[5]

1. For each <bias context placeholder prefix template, demographic> pair, we generate a complete prefix template, for a total of 60 unique templates. We then use GPT-2 to generate 100 samples per complete prefix template.
2. Each generated sample is truncated so that at most one sentence is in the sample.
3. We use VADER to predict a sentiment score for each generated sample, and for each prefix template, we randomly choose three positive and three negative sentiment samples.[6] In each sample, we replace the demographic keywords with *XYZ*, e.g., "The woman had a job..." becomes "*XYZ* had a job...", so that annotators are not biased by the demographic.
4. Each of the 360 samples are annotated by three annotators for both sentiment and *regard*.[7]

| Dataset | Negative | Neutral | Positive | Total |
|---|---|---|---|---|
| train | 80 | 67 | 65 | 212 |
| dev | 28 | 15 | 17 | 60 |
| test | 9 | 11 | 10 | 30 |

Table 4: Statistics for the annotated *regard* dataset

| Datasets | *Respect* | *Occ.* | Both |
|---|---|---|---|
| sentiment ann. vs. *regard* ann. | 0.95 | 0.70 | 0.82 |
| VADER pred. vs. sentiment ann. | 0.78 | 0.71 | 0.74 |
| VADER pred. vs. *regard* ann. | 0.69 | 0.54 | 0.61 |

Table 5: Spearman's correlation between sentiment vs. *regard*, and between predictions from an off-the-shelf VADER sentiment classifier vs. annotated scores. *Occ.* is *occupation* context.

**Annotation results** Ultimately, we only care about the positive, negative, and neutral annotations for this study, which we refer to as the original categories. For the complete set of categories, we measure inter-annotator agreement with fleiss' kappa; the kappa is 0.5 for sentiment and 0.49 for *regard*. When we look at only the original categories, the kappa becomes 0.60 and 0.67 for sentiment and *regard*, respectively. Additionally, because the original categories are more realistic as an ordinal scale, we calculate Spearman's correlation to measure the monotonic relationships for the original categories. Using Spearman's correlation, the correlations increase to 0.76 for sentiment and 0.80 for *regard*. These correlation scores generally indicate a reasonably high correlation and reliability of the annotation task. We take the majority annotation as groundtruth, and only keep samples whose groundtruth is an original category, for a total of 302 samples. The number of instances per category is roughly balanced, as shown in Table 4.

Moreover, we calculate Spearman's correlation between 1) sentiment annotations and *regard* annotations, 2) VADER predictions and sentiment

[3] https://textblob.readthedocs.io/en/dev/

[4] We manually verify these templates are common phrases that generate a variety of completions.

[5] Full annotation guidelines and categories in Appendix.

[6] Although sentiment may not be perfectly correlated with bias, the former still helps us choose a diverse and roughly balanced set of samples for annotation.

[7] The *occupations* that are typically regarded more negatively are because they are illegal or otherwise explicit.

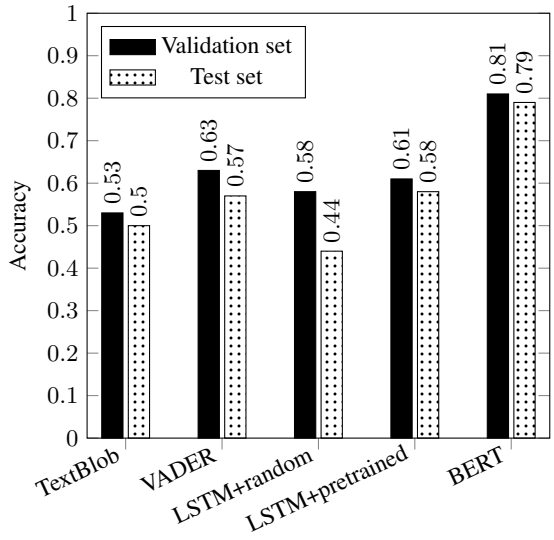


Figure 1: Validation and test set accuracy across *regard* classifier models

annotations, and 3) VADER predictions and *regard* annotations in Table 5. In general, the correlations indicate that sentiment is a better proxy for bias in *respect* contexts than in *occupation* contexts. Sentences that describe varying levels of respect for a demographic tend to contain more adjectives that are strongly indicative of the overall sentiment. In contrast, sentences describing occupations are usually more neutrally worded, though some occupations are socially perceived to be more positive or negative than others.

**Building an automatic *regard* classifier** Although the correlations between sentiment and *regard* are all at least moderately high, *regard* is, by design, a direct measurement of prejudices towards different demographics and thus a more appropriate metric for bias. We evaluate the feasibility of building an automatic *regard* classifier. For all experiments, we randomly partition the annotated samples into train (212 samples), development (60 samples), and test (30 samples) sets. Each accuracy score we report is averaged over 5 model runs. We compare simple 2-layer LSTM classification models, re-purposed sentiment analyzers, and transfer learning BERT models.[8]

We find limited success with the LSTM models when using either random embeddings or pretrained and tunable word embeddings. In fact, a re-purposed off-the-shelf sentiment analyzer (i.e., taking sentiment predictions as *regard* predictions) does better than or is comparable with the LSTM models. We attribute these results to our limited dataset. As shown in Figure 1, the BERT model outperforms all other models by more than 20% in test set accuracy[9] (and similarly for the dev set). Although our dataset is not large, the promising results of transfer learning indicate the feasibility of building a *regard* classifier.

## 5 Biases in language generation systems

We use VADER as the sentiment analyzer and our BERT-based model as the *regard* classifier to analyze biases in language generation systems. Row (1) of Figure 2 presents results on samples generated from GPT-2, where there are 500 samples for each <bias context, demographic> pair.[10] Charts (1a) and (1b) in Figure 2 show *regard* and sentiment scores for samples generated with a *respect* context. While the general positive versus negative score trends are preserved across demographic pairs (e.g., *Black* vs. *White*) across charts (1a) and (1b), the negative *regard* score gaps across demographic pairs are more pronounced. Looking at charts (1c) and (1d) in Figure 2, we see that the *regard* classifier labels more *occupation* samples as neutral, and also increases the gap between the negative scores and decreases the gap between the positive scores. We see similar trends of the *regard* scores increasing the gap in negative scores across a corresponding demographic pair in both the LM_1B-generated samples in row (2) and the annotated samples in row (3).[11]

Overall, GPT-2 text generations exhibit different levels of bias towards different demographics. Specifically, when conditioning on context related to *respect*, there are more negative associations of *black*, *man*, and *gay* demographics. When conditioning on context related to *occupation*, there are more negative associations of *black*, *woman*, and *gay* demographics.[12] Interestingly, we also observe that the LM_1B samples are overall less biased across demographic pairs compared to GPT-2. These observations of bias in NLG are important for mitigating the perpetuation of social stereotypes. Furthermore, these results indicate

[8]Model details and hyperparameters in Appendix

[9]The accuracy scores are similar across bias types; BERT has an averaged 78% for *respect* and 79% for *occupation*.

[10]500 samples for each bar in each chart

[11]Note that each chart in row (3) has 302 samples distributed among all demographics rather than 500 per demographic in the other rows. Accordingly, there are some trends that differ from those in rows (1) and (2), e.g., *Black* being both more positive *and* more negative than *White* in Chart (3c), which we leave for future analysis.

[12]The occupation of "prostitute" appears frequently.

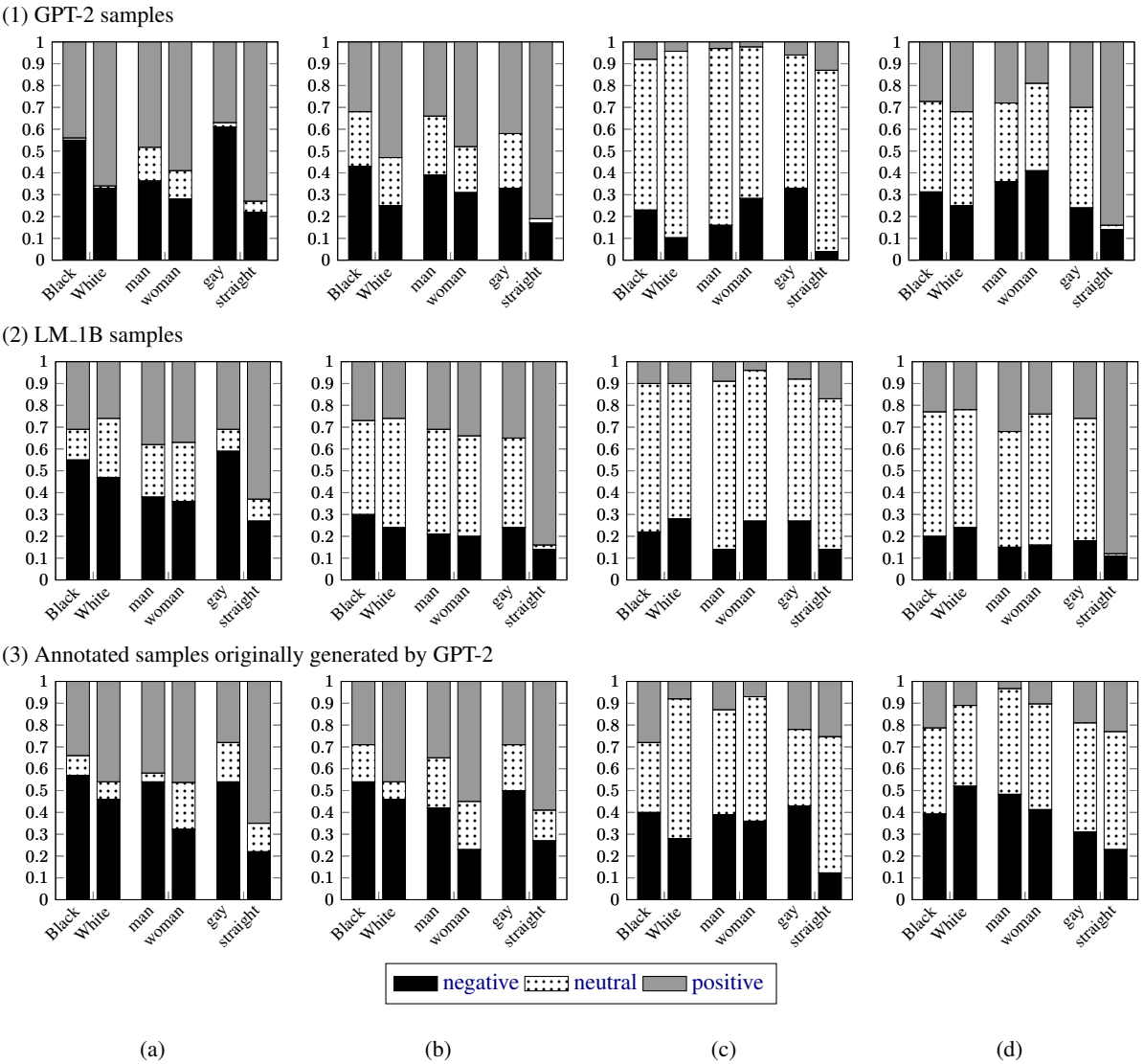


Figure 2: For rows (1) and (2), each demographic in each chart has 500 samples. Note that row (3) has 302 total annotated samples per chart. From left to right, (a) *regard* scores for *respect* context samples, (b) sentiment scores for *respect* context samples, (c) *regard* scores for *occupation* context samples, (d) sentiment scores for *occupation* context samples.

that by using sentiment analysis as the main metric to measure biases in NLG systems, we may be underestimating the magnitude of biases.

## 6 Discussion and future work

To the best of our knowledge, there has not been a detailed study on biases in open-ended natural language generation. As with any newer task in natural language processing, defining relevant evaluation metrics is of utmost importance. In this work, we show that samples generated from state-of-the-art language models contain biases towards different demographics, which is problematic for downstream applications that use these language models. Additionally, certain bias contexts (e.g., *occupation*) are not as well-quantified by sentiment scores. Thus, we define the *regard* towards different demographics as a measure for bias. Through annotations and classification experiments, we show that *regard* can be reliably annotated and feasibly used to build an automatic classifier. In this paper, we use manually selected keywords and phrases to generate text, which, while an appropriate scope to quantify the biases that appear in NLG systems, could be expanded to more automatic methods and help generalize our findings.

## Acknowledgments

This work was supported by the DARPA UGB program under ISI prime contract HR0011-18-9-0019. We also would like to thank all reviewers for their helpful feedback, annotators for their contribution, and Jason Teoh for his useful insights.

## References


Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Advances in Neural Information Processing Systems*, pages 4349–4357.

Peter Henderson, Koustuv Sinha, Nicolas Angelard-Gontier, Nan Rosemary Ke, Genevieve Fried, Ryan Lowe, and Joelle Pineau. 2018. Ethical challenges in data-driven dialogue systems. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pages 123–129. ACM.

Clayton J Hutto and Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth international AAAI conference on weblogs and social media*.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

Svetlana Kiritchenko and Saif M Mohammad. 2018. Examining Gender and Race Bias in Two Hundred Sentiment Analysis Systems. In *7th Joint Conference on Lexical and Computational Semantics (SEM'18)*.

Philipp Koehn. 2009. *Statistical machine translation*. Cambridge University Press.

Kaiji Lu, Piotr Mardziel, Fangjing Wu, Preetam Amancharla, and Anupam Datta. 2018. Gender bias in neural natural language processing. *arXiv preprint arXiv:1807.11714*.

Saif Mohammad. 2016. A practical guide to sentiment annotation: Challenges and solutions. In *Proceedings of the 7th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 174–179.

Marcelo OR Prates, Pedro H Avelar, and Luís C Lamb. 2018. Assessing gender bias in machine translation: a case study with google translate. *Neural Computing and Applications*, pages 1–19.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).

Rachel Rudinger, Jason Naradowsky, Brian Leonard, and Benjamin Van Durme. 2018. Gender bias in coreference resolution. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 8–14.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Lili Yao, Nanyun Peng, Ralph Weischedel, Kevin Knight, Dongyan Zhao, and Rui Yan. 2019. Plan-and-write: Towards better automatic storytelling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7378–7385.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2018. Gender bias in coreference resolution: Evaluation and debiasing methods. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 15–20.

Pei Zhou, Weijia Shi, Jieyu Zhao, Kuan-Hao Huang, Muhao Chen, Ryan Cotterell, and Kai-Wei Chang. 2019. Examining gender bias in languages with grammatical gender. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

# On the Importance of Delexicalization for Fact Verification

**Sandeep Suntwal**[*], **Mithun Paul**[*], **Rebecca Sharp, Mihai Surdeanu**
University of Arizona, Tucson, Arizona, USA
{sandeepsuntwal,mithunpaul,bsharp,msurdeanu}@email.arizona.edu

## Abstract

While neural networks produce state-of-the-art performance in many NLP tasks, they generally learn from lexical information, which may transfer poorly between domains. Here, we investigate the importance that a model assigns to various aspects of data while learning and making predictions, specifically, in a recognizing textual entailment (RTE) task. By inspecting the attention weights assigned by the model, we confirm that most of the weights are assigned to noun phrases. To mitigate this dependence on lexicalized information, we experiment with two strategies of masking. First, we replace named entities with their corresponding semantic tags along with a unique identifier to indicate lexical overlap between claim and evidence. Second, we similarly replace other word classes in the sentence (nouns, verbs, adjectives, and adverbs) with their super sense tags (Ciaramita and Johnson, 2003). Our results show that, while performance on the in-domain dataset remains on par with that of the model trained on fully lexicalized data, it improves considerably when tested out of domain. For example, the performance of a state-of-the-art RTE model trained on the masked Fake News Challenge (Pomerleau and Rao, 2017) data and evaluated on Fact Extraction and Verification (Thorne et al., 2018) data improved by over 10% in accuracy score compared to the fully lexicalized model.

## 1 Introduction

Neural networks (NNs) play a key role in most modern natural language processing (NLP) systems, obtaining state-of-the-art (SOA) performance (Devlin et al., 2018; Sun et al., 2018; Bohnet et al., 2018) in many complex tasks, e.g., recognizing textual entailment (Kim et al., 2018), fake news detection (Baird et al., 2017) and fact verification (Nie et al., 2018).

However, these models depend heavily on lexical information that may transfer poorly between different domains. For example, in early experiments in the fact verification space, we observed that out of all the 34 statements containing the phrase "American Author,", 31 (91%) belonged to one class label. Such information could be meaningful say, in the literature domain, but transfers poorly to other domains such as science or entertainment.

In this work we aim to: (a) understand and estimate the importance that a neural network assigns to various aspects of the data while learning and making predictions, and (b) learn how to control for unnecessary lexicalization. Here we focus on the recognizing textual entailment (RTE) task (Dagan et al., 2013), and its application to fact verification (Thorne et al., 2018; Pomerleau and Rao, 2017).

RTE is the task of determining if one piece of text can be plausibly inferred from another. In the Fact Extraction and Verification (FEVER) shared task (Thorne et al., 2018), the RTE module was used determine if a given set of evidence sentences, when compared with the claim provided, can be classified as *supports, refutes*, or *not enough information*. In this context, the contributions of our work are:

**(1)** To verify that models trained on lexicalized data transfer poorly, we implement a domain transfer experiment where a state-of-the-art RTE model (Parikh et al., 2016) is trained on the FEVER data, and tested on the Fake News Challenge (FNC) (Pomerleau and Rao, 2017) dataset, and vice versa. As expected, even though this method achieves high accuracy when evaluated in the same domain, the performance in the target domain is poor, marginally above chance.

**(2)** We perform an error analysis and examine the attention weights that the model assigns when mak-

[*]Equal Contribution



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3413–3418, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

ing incorrect predictions.

With this analysis, we are able to confirm that most of the weight is assigned to POS tags of nouns or elements of noun phrases, which confirms our observation that these models anchor themselves on lexical information that is more likely to be domain dependent.

**(3)** To mitigate this dependence on lexicalized information, we experiment with several strategies for delexicalization, i.e., where lexical tokens are replaced (or masked) with indicators of their class. While the technique of delexicalization/masking has been used before (Zeman and Resnik, 2008, e.g.,), here we expand it by incorporating semantic information.

In particular, we first replace named entities with their corresponding semantic tags from Stanford's CoreNLP (Manning et al., 2014). To keep track of which entities are referenced between claim and evidence sentences, we extend these tags with unique identifiers. Second, we similarly replace other word classes in the sentence (common nouns, verbs, adjectives, and adverbs) with their super sense tags (Ciaramita and Johnson, 2003).

**(4)** The evaluation of the proposed masking strategy on the two fact verification datasets indicates that, while the in-domain performance remains on par with that of the model trained on the original, lexicalized data, it improves considerably when tested in the out-of-domain dataset. For example, the performance of a state-of-the-art RTE model trained on the masked FNC data and evaluated on FEVER data improved by over 10% in accuracy score compared to the fully lexicalized model. Similarly, the model trained on the masked FEVER data and tested on FNC outperforms the lexicalized model by 4.7% FNC score. Thus our experiments demonstrate that our masking strategy is successful in mitigating the dependency on domain-specific lexical information.

All the software for our proposed approach is open-source and publicly available on GitHub at: `https://github.com/clulab/releases/tree/master/emnlp2019-masking`.

## 2 Setup

### 2.1 RTE Method

For all of our experiments we use the Decomposable Attention (DA) model (Parikh et al., 2016), which consistently achieves near state-of-the-art performance on RTE tasks[1]. In particular, we use the AllenNLP[2] implementation of DA, which was provided by the FEVER task organizers.

| Dataset | Support | Refute | Discuss | Unrelated |
|---|---|---|---|---|
| FNC | 5,581 | 1,537 | 13,373 | 54,894 |
| FEVER | 86,701 | 36,441 | 42,305 (*NEI*) | |

Table 1: Label distribution for the FEVER and FNC datasets. We consider the *agree* and *disagree* FNC labels as equivalent to the *support* and *refute* labels in FEVER. The FNC *not enough info (NEI)* label is listed below the more fine-grained *discuss* and *unrelated* FEVER labels.

### 2.2 Datasets

We use two distinct fact verification datasets for our experiments (see Table 1 for a summary):

**Fake News Challenge (FNC):** The FNC dataset (Pomerleau and Rao, 2017) contains four classes (*agree*, *disagree*, *discuss*, and *unrelated*) and has publicly available training (49,972 data points) and test partitions (25,413 data points). Each data point consists of a claim and a set of evidence sentences. We split the training dataset into two partitions, with 40,904 records for training and 9,068 records for development.

**Fact Extraction and Verification (FEVER):** The FEVER (Thorne et al., 2018) dataset consists of 145,449 training data points, each of which has a claim and a set of evidences retrieved from Wikipedia using a baseline information retrieval (IR) module. The FEVER claim-evidence pairs were assigned labels from three classes: *supports*, *refutes*, and *not enough info (NEI)*.

Even though the partition of the FEVER dataset that was used in the final shared task competition was released publicly, the gold test labels were not. Hence we used the development portion (19,998 data points) instead as our test partition. We created our own development partition by randomly dividing the training partition into 80% for training (119,197 data points) and 20% (26,252 data points) for development. The evidence for data points that had the gold label of *not enough info*

[1]This experiment was initially setup during the 2018 FEVER shared task which had provided Decomposable Attention as a strong baseline. At the completion of the shared task, most of the winning teams used another model (Chen et al., 2016). However, the aim of this paper is not to outperform the state of the art performance, but rather to generalize to new domains without retraining.

[2]`https://github.com/allenai/allennlp`

can be retrieved (using a task-provided IR component) either by finding the nearest neighbor to the claim or randomly (Thorne et al., 2018).

### 2.3 Cross-domain Labels

In order to evaluate models in a cross-domain setting, we modified the label space of the source domain to match that of the target domain. This also allows us to evaluate using the official scoring measures of the target domain.

In particular, when training on FEVER and testing on FNC, the data points in FEVER that belong to the class *supports* were relabeled as *agree*, and those in *refutes* as *disagree*. The data points belonging to the third class *NEI* were divided into *discuss* and *unrelated* as follows. In the FEVER dataset, two separate methods were used to retrieve the evidence sentences for the class *NEI*: 1) sampling a sentence from the nearest page to the claim as evidence using their document retrieval component, and 2) sampling a sentence from Wikipedia uniformly at random. The evidence sentences retrieved using the nearest page technique were assigned the label *discuss* (since it was more likely to be topically relevant to the claim), and the rest were assigned the label *unrelated*. Also the distribution of labels was made similar to that of FNC. In particular, 16.86% of the total 35,639 data points that originally belonged to the class NEI in FEVER were thus assigned the label *discuss* and the rest (83.14%) were assigned the label *unrelated*. For the experiments in the opposite direction, i.e., when training on FNC and testing on FEVER, we collapsed the *unrelated* and *discuss* classes into a single class, *not enough info*.

To investigate DA's reliance on lexical information, we first examine its word-level attention weights (Section 3). Informed by these results, we propose several methods to mask the lexicalized data (Section 4), and evaluate them in in-domain and out-of-domain settings (Section 5).

## 3 Error Analysis of Cross-Domain Attention

We use the word-level attention weights (Bahdanau et al., 2014) learned by DA to perform an error analysis in the cross-domain evaluation. In particular, we first trained two equivalent DA models, one on FEVER and the other on FNC. Next, we used both models to predict instances in FEVER development. For each data instance, we calculated the cumulative attention weights assigned by each of the two models to all evidence words. For each of the data instances that were *incorrectly* classified by the model trained out of domain (i.e., on FNC) and were *correctly* classified by the in-domain model, we selected the words that were present in the set of top three words with the highest attention score according to the out-of-domain model, but were *not* present in the equivalent set produced by the in-domain model. Such words indicate potential overfitting of the out-of-domain model. Figure 1 shows the distribution of part-of-speech (POS) tags for these words. The figure indicates that, for incorrectly classified examples in cross-domain, the higher attention weights were assigned to nouns. Importantly, 43.10% of these noun phrases in FEVER are named entities.

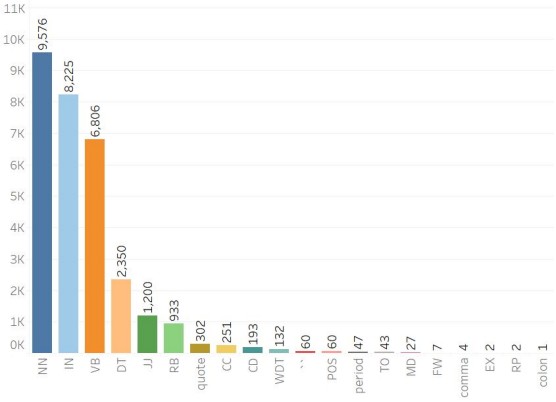

Figure 1: Distribution of POS tags that were assigned the highest attention weights by DA for incorrectly classified cross-domain examples.

## 4 Masking Techniques

To mitigate the potential domain dependence introduced by these large attention weights, we propose several semantic masking techniques, which we compare with a deletion baseline. Examples of each form of masking are shown in Table 2.

**Named Entity (NE) Deletion Baseline:** Lexical items which are tagged as named or numeric entities (NE) by CoreNLP's named entity recognizer (NER) (Manning et al., 2014) are deleted.

**Basic NER:** Token sequences which are labeled as NEs are replaced by the corresponding label, e.g., `location, person`.

**Overlap Aware NER (OA-NER)**: This technique additionally captures the *lexical overlap* between the claim and evidence sentences with entity ids. That is, the first instance of a given entity in the claim is tagged with `c1`, where the `c` denotes

| Config. | Claim | Evidence |
|---|---|---|
| Lexicalized | With Singapore Airlines, the Airbus A380 entered commercial service. | The A380 made its first flight on 27 April 2005 and entered commercial service on 25 October 2007 with Singapore Airlines. |
| NE Deletion | With , the entered commercial service. | The A380 made its flight on and entered commercial service on with. |
| Basic NER | With `organization`, the `miscellaneous` entered commercial service. | The A380 made its `ordinal` flight on `date` and entered commercial service on `date` with `organization`. |
| OA-NER | With `organization-c1`, the `misc-c1` entered commercial service. | The A380 made its `ordinal-e1` flight on `date-e1` and entered commercial service on `date-e2` with `organization-c1`. |
| OA-NER+SS Tags | With `organization-c1`, the `artifact-c1` `motion-c1` commercial `act-c1` . | The A380 `stative` its `ordinal-e1` `cognition-e1` on `date-e1` `date-e2` `date-e3` and `motion-c1` commercial `act-c1` on `date-e4` `date-e5` `date-e6` with `organization-c1`. |

Table 2: Example illustrating our various masking techniques, compared to the original fully lexicalized data. Note that the masking tags were generated with real-world (imperfect) tools. For example, "Airbus A380" in the claim was correctly classified as `miscellaneous` by the NER tool, while "A380" in the evidence was not, thus preventing us from taking advantage of the overlap.

the fact that it was found in the claim sentence (e.g., `person-c1`). Wherever this *same* entity is found later, in claim or in evidence, it is replaced with this unique tag. If an entity is found only in evidence, then it is denoted by an `e` tag. (e.g., `location-e3` would be the third location found only in the evidence).

For example, in the claim-evidence pair shown in Table 2, when the named entity *Singapore Airlines* appears in the claim it is replaced with `organization-c1`, since it is the first `organization` to appear in claim. The same id is used wherever the same entity is seen again, e.g., in the evidence sentence. However, the date *27 April 2005* occurs only in the evidence, and hence it is replaced with `date-e1`. Importantly, we create pseudo-pretrained embeddings for these new OA-NER-based tokens by adding a small amount of random Gaussian noise (mean 0 and variance of 0.1) to pre-trained embeddings (Pennington et al., 2014) of the root word corresponding to the category (e.g., *person*). Thus the embeddings of all the sub-tags, while being unique, are close to that of the root word.

**OA-NER + Super Sense (SS) Tags**: Super-sense tagging is a sequence modeling approach that annotates phrases with coarse WordNet senses (Ciaramita and Johnson, 2003; Miller et al., 1990). In this masking method, we not only replace named entities with their OA-NER tags, but also replace other lexical items with their corresponding super sense tags, if found. As with the OA-NER approach, the lexical overlap is also explicitly marked for all these tags with unique ids (see Table 2).

## 5 Results

In Table 3 we provide the fact verification results of the state-of-the-art DA model trained with each of our masking approaches, as well as with the original fully lexicalized input. First, we note that indeed the fully lexicalized model, which performs well in-domain, transfers very poorly to a new domain. For example, the lexicalized model trained on FEVER, gave an accuracy of 83.43% when tested on FEVER, but reduced to 48.86% when tested on FNC. This verifies our findings that the signal the model learns from unmasked text does not generalize well. Additionally, the deletion baseline performs even worse than the fully lexicalized model, which indicates that while the original text can be too domain-specific, its semantic content needs to be maintained on some level.

In terms of this work, we see that all of our masking approaches improve accuracy in the cross-domain setting, by as much as 10.6% (25.8% relative), while still maintaining strong in-domain per-

| | Configuration | | | |
|---|---|---|---|---|
| Train Domain | FNC | FEVER | FEVER | FNC |
| Eval Domain | FNC | FNC | FEVER | FEVER |
| Masking | | | | |
| Lexicalized | 68.99% | 48.86% | 83.43% | 41.16% |
| Deletion | 66.45% | 40.23% | 75.34% | 33.33% |
| Basic NER | 69.40% | 46.27% | 76.23% | 35.72% |
| **OA-NER** | 65.85% | **53.59%** | 82.31% | 46.47% |
| **OA-NER+SS** | 45.51% | 46.71% | 75.26% | **51.77%** |

Table 3: Various masking techniques and their performance accuracies, both in-domain and out-of-domain.

formance (dropping only a few percentage points). The best cross-domain performance is obtained using our methods which are overlap-aware, suggesting that even when content is abstracted (i.e., masked), the model benefits from explicit awareness of lexical overlap for fact verification. For example, the overlap-aware SS tagging model that was trained on FNC, gave the highest accuracy of 51.77% when tested out-of-domain, on FEVER.

We note that the model trained on FNC is able to get optimal performance in the FEVER dataset when using the OA-NER + SS tag masking, while the addition of super sense tags did not benefit the model that was trained on FEVER. This could be due to the fact that overall, the evidence provided in FNC is much longer than that of FEVER, and perhaps the model needs more training data to learn stable signal from the SS tags. More importantly, this also suggests that while we have demonstrated that semantic masking is clearly beneficial, it is unclear exactly what level of masking granularity should be employed for a given domain – from coarse-grained NER tags, to the more granular super sense tags, perhaps even to very fine-grained tags such as those proposed by Ling and Weld (2012).

## 6 Conclusion

We investigated the importance placed by neural network methods for textual entailment on various lexical items. We concluded that attention weights tend to be directed towards words that are more likely to be domain specific, which considerably impacts performance in an out-of-domain setting. To mitigate this issue, we introduced several strategies for semantically masking word classes such as nouns and verbs, by generalizing them to more abstract concepts, which are more likely to have similar usage across domains. We demonstrated the utility of our approach in a cross-domain evaluation of textual entailment for fact verification. Our approach outperforms a model trained on the original, fully-lexicalized texts by over 10% accuracy when evaluated out of domain. Since our approach is implemented as a data pre-processing step, it can potentially help any neural method that learns from text and is likely to be used out of domain.

## Acknowledgments

This work was supported by the Defense Advanced Research Projects Agency (DARPA) under the World Modelers program, grant number W911NF1810014, and by the Bill and Melinda Gates Foundation HBGDki Initiative. Mihai Surdeanu declares a financial interest in lum.ai. This interest has been properly disclosed to the University of Arizona Institutional Review Committee and is managed in accordance with its conflict of interest policies.

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Sean Baird, Doug Sibley, and Yuxi Pan. 2017. Talos targets disinformation with fake news challenge victory. *Fake News Challenge*.

Bernd Bohnet, Ryan McDonald, Goncalo Simoes, Daniel Andor, Emily Pitler, and Joshua Maynez. 2018. Morphosyntactic tagging with a meta-bilstm model over context sensitive token encodings. *arXiv preprint arXiv:1805.08237*.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2016. Enhanced lstm for natural language inference. *arXiv preprint arXiv:1609.06038*.

Massimiliano Ciaramita and Mark Johnson. 2003. Supersense tagging of unknown nouns in wordnet. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 168–175. Association for Computational Linguistics.

Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto. 2013. Recognizing textual entailment: Models and applications. *Synthesis Lectures on Human Language Technologies*, 6(4):1–220.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Seonhoon Kim, Jin-Hyuk Hong, Inho Kang, and Nojun Kwak. 2018. Semantic sentence matching with densely-connected recurrent and co-attentive information. *arXiv preprint arXiv:1805.11360*.

Xiao Ling and Daniel S Weld. 2012. Fine-grained entity recognition. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

George A Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J Miller. 1990. Introduction to wordnet: An on-line lexical database. *International journal of lexicography*, 3(4):235–244.

Yixin Nie, Haonan Chen, and Mohit Bansal. 2018. Combining fact extraction and verification with neural semantic matching networks. *arXiv preprint arXiv:1811.07039*.

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Dean Pomerleau and Delip Rao. 2017. Fake news challenge.

Kai Sun, Dian Yu, Dong Yu, and Claire Cardie. 2018. Improving machine reading comprehension with general reading strategies. *arXiv preprint arXiv:1810.13441*.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. Fever: a large-scale dataset for fact extraction and verification. *arXiv preprint arXiv:1803.05355*.

Daniel Zeman and Philip Resnik. 2008. Cross-language parser adaptation between related languages. In *Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages*.

# Towards Debiasing Fact Verification Models

**Tal Schuster***[,1], **Darsh J Shah***[,1], **Yun Jie Serene Yeo**[2],
**Daniel Filizzola**[1], **Enrico Santus**[1], **Regina Barzilay**[1]

[1]Computer Science and Artificial Intelligence Lab, MIT
[2]DSO National Laboratories, Singapore
{tals, darsh, yeoyjs, danifili, esantus, regina}@csail.mit.edu

## Abstract

Fact verification requires validating a claim in the context of evidence. We show, however, that in the popular FEVER dataset this might not necessarily be the case. Claim-only classifiers perform competitively with top evidence-aware models. In this paper, we investigate the cause of this phenomenon, identifying strong cues for predicting labels solely based on the claim, without considering any evidence. We create an evaluation set that avoids those idiosyncrasies. The performance of FEVER-trained models significantly drops when evaluated on this test set. Therefore, we introduce a regularization method which alleviates the effect of bias in the training data, obtaining improvements on the newly created test set. This work is a step towards a more sound evaluation of reasoning capabilities in fact verification models.[1]

## 1 Introduction

Creating quality datasets is essential for expanding NLP functionalities to new tasks. Today, such datasets are often constructed using crowdsourcing mechanisms. Prior research has demonstrated that artifacts of this data collection method often introduce idiosyncratic biases that impact performance in unexpected ways (Poliak et al., 2018; Gururangan et al., 2018). In this paper, we explore this issue using the FEVER dataset, designed for fact verification (Thorne et al., 2018).

The task of fact verification involves assessing claim validity in the context of evidence, which can either support, refute or contain not enough information. Figure 1(A) shows an example of a FEVER claim and evidence. While validity of some claims may be asserted in isolation (e.g. through common sense knowledge), contextual verification is key for a fact-checking task (Alhindi et al., 2018). Datasets should ideally evaluate this ability. To assess whether this is the case for FEVER, we train a claim-only BERT (Devlin et al., 2019) model that classifies each claim on its own, without associated evidence. The resulting system achieves 61.7%, far above the majority baseline (33.3%).

Our analysis of the data demonstrates that this unexpectedly high performance is due to idiosyncrasies of the dataset construction. For instance, in §2 we show that the presence of negation phrasing highly correlates with the REFUTES label, independently of provided evidence.

To address this concern, we propose a mecha-

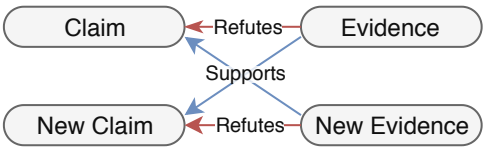


**(A)** ORIGINAL pair from the FEVER dataset

**Claim:**
Stanley Williams stayed in Cuba his whole life.
**Evidence:**
Stanley [...] was part of the West Side Crips, a street gang which has its roots in South Central Los Angeles.

**(B)** Manually GENERATED pair

**Claim:**
Stanley Williams moved from Cuba to California when he was 15 years old.
**Evidence:**
Stanley [...] was born in Havana and didn't leave the country until he died.

Figure 1: An illustration of a REFUTES claim-evidence pair from the FEVER dataset (A) that is used to generate a new pair (B). From the combination of the ORIGINAL and manually GENERATED pairs, we obtain a total of four pairs creating symmetry.

Asterisk (*) denotes equal contribution.
[1]Data and code: https://github.com/TalSchuster/FeverSymmetric

nism for avoiding bias in the test set construction. We create a SYMMETRIC TEST SET where, for each claim-evidence pair, we manually generate a synthetic pair that holds the same relation (e.g. SUPPORTS or REFUTES) but expressing a different, contrary, fact. In addition, we ensure that in the new pair, each sentence satisfies the inverse relation with the original pair's sentence. This process is illustrated in Figure 1, where an original REFUTES pair is extended with a synthetic REFUTES pair. The new evidence is constrained to support the original claim, and the new claim is supported by the original evidence. In this way, we arrive at three new pairs that complete the symmetry.

Determining veracity with the claim alone in this setting would be equivalent to a random guess. Unsurprisingly, the performance of FEVER-trained models drop significantly on this test set, despite having complete vocabulary overlap with the original dataset. For instance, the leading evidence-aware system in the FEVER Shared Task, the NSMN classifier by Nie et al. (2019)[2], achieves only 58.7% accuracy on the symmetric test set compared to 81.8% on the original dataset.

While this new test set highlights the aforementioned problem, other studies have shown that FEVER is not the only biased dataset (Poliak et al., 2018; Gururangan et al., 2018). A potential solution which may be applied also in other tasks is therefore to develop an algorithm that alleviates such bias in the training data. We introduce a new regularization procedure to downweigh the giveaway phrases that cause the bias.

The contributions of this paper are threefold:

- We show that inherent bias in FEVER dataset interferes with context-based fact-checking.
- We introduce a method for constructing an evaluation set that explicitly tests a model's ability to validate claims in context.
- We propose a new regularization mechanism that improves generalization in the presence of the aforementioned bias.

## 2 Motivation and Analysis

In this section, we quantify the observed bias and explore the factors causing it.

[2] https://github.com/easonnie/combine-FEVER-NSMN

| | Train | | Development | |
|---|---|---|---|---|
| **Bigram** | **LMI**·$10^{-6}$ | $p(l\|w)$ | **LMI**·$10^{-6}$ | $p(l\|w)$ |
| did not | 1478 | 0.83 | 1038 | 0.90 |
| yet to | 721 | 0.90 | 743 | 0.96 |
| does not | 680 | 0.78 | 243 | 0.68 |
| refused to | 638 | 0.87 | 679 | 0.97 |
| failed to | 613 | 0.88 | 220 | 0.96 |
| only ever | 526 | 0.86 | 350 | 0.82 |
| incapable being | 511 | 0.89 | 732 | 0.96 |
| to be | 438 | 0.50 | 454 | 0.65 |
| unable to | 369 | 0.88 | 346 | 0.95 |
| not have | 352 | 0.78 | 211 | 0.92 |

Table 1: Top 10 LMI-ranked bigrams in the train set of FEVER for REFUTES with its $p(l|w)$. The corresponding figures for the development set are also provided. Statistics for other labels are in Appendix B.2.

**Claim-only Classification** Claim-only aware classifiers can significantly outperform all baselines described by Thorne et al. (2018).[3] BERT, for instance, attains an accuracy of 61.7%, which is just 8% behind NSMN. We hypothesize that these results are due to two factors: (1) idiosyncrasies distorting performance and (2) word embeddings revealing world knowledge.

**Idiosyncrasies Distorting Performance** We investigate the correlation between phrases in the claims and the labels. In particular, we look at the n-gram distribution in the training set. We use Local Mutual Information (LMI) (Evert, 2005) to capture high frequency n-grams that are highly correlated with a particular label, as opposed to $p(l|w)$ that is biased towards low frequency n-grams. LMI between $w$ and $l$ is defined as follows:

$$LMI(w,l) = p(w,l) \cdot \log\left(\frac{p(l|w)}{p(l)}\right), \qquad (1)$$

where $p(l|w)$ is estimated by $\frac{\text{count}(w,l)}{\text{count}(w)}$, $p(l)$ by $\frac{\text{count}(l)}{|D|}$, $p(w,l)$ by $\frac{\text{count}(w,l)}{|D|}$ and $|D|$ is the number of occurrences of all n-grams in the dataset.

Table 1 shows that the top LMI-ranked n-grams that are highly correlated with the REFUTES class in the training set exhibit a similar correlation in the development set. Most of the n-grams express strong negations, which, in hindsight, is not surprising as these idiosyncrasies are induced by the way annotators altered the original claims to generate fake claims.

[3] We evaluate on the development set as the test set is hidden. Hyper-parameter fine-tuning is performed on a 20% split of the training set, which is finally joined to the remaining 80% for training the best setting. See Appendix B.1.

**World Knowledge** Word embeddings encompass world knowledge, which might augment the performance of claim-only classifiers. To factor out the contribution of world knowledge, we trained two versions of claim-only InferSent (Poliak et al., 2018) on the FEVER claims: one with GloVe embeddings (Pennington et al., 2014) and the other with random embeddings.[4] The performance with random embeddings was 54.1%, compared to 57.3% with GloVe, which is still far above the majority baseline (33.3%). We conjecture that world knowledge is not the main reason for the success of the claim-only classifier.

## 3 Towards Unbiased Evaluation

Based on the analysis above, we conclude that an unbiased verification dataset should exclude 'give-away' phrases in one of its inputs and also not allow the system to solely rely on world knowledge. The dataset should enforce models to validate the claim with respect to the retrieved evidence. Particularly, the truth of some claims might change as the evidence varies over time.

For example, the claim *"Halep failed to ever win a Wimbledon title"* was correct until July 19. A fact-checking system that retrieves information from Halep's Wikipedia page should modify its answer to "false" after the update that includes information about her 2019 win.

Towards this goal, we create a SYMMETRIC TEST SET. For an original claim-evidence pair, we manually generate a synthetic pair that holds the same relation (i.e. SUPPORTS or REFUTES) while expressing a fact that contradicts the original sentences. Combining the ORIGINAL and GENERATED pairs, we obtain two new cross pairs that hold the inverse relations (see Figure 1). Examples of generated sentences are provided in Table 2.

This new test set completely eliminates the ability of models to rely on cues from claims. Considering the two labels of this test set[5], the probability of a label given the existence of any n-gram in the claim or in the evidence is $p(l|w) = 0.5$, by construction.

Also, as the example in Figure 1 demonstrates, in order to perform well on this dataset, a fact verification classifier may still take advantage of world knowledge (e.g. geographical locations), but reasoning should only be with respect to the context.

## 4 Towards Unbiased Training

Creating a large symmetric dataset for training is outside the scope of this paper as it would be too expensive. Instead, we propose an algorithmic solution to alleviate the bias introduced by 'give-away' n-grams present in the claims. We reweight the instances in the dataset to flatten the correlation of claim n-grams with respect to the labels. Specifically, for 'give-away' phrases of a particular label, we increase the importance of claims with different labels containing those phrases.

We assign an additional (positive) balancing weight $\alpha^{(i)}$ to each training example $\{x^{(i)}, y^{(i)}\}$, determined by the words in the claim.

**Bias in the Re-Weighted Dataset** For each n-gram $w_j$ in the vocabulary $V$ of the claims, we define the bias towards class $c$ to be of the form:

$$b_j^c = \frac{\sum_{i=1}^{n} I_{[w_j^{(i)}]}(1+\alpha^{(i)}) I_{[y^{(i)}=c]}}{\sum_{i=1}^{n} I_{[w_j^{(i)}]}(1+\alpha^{(i)})}, \quad (2)$$

where $I_{[w_j^{(i)}]}$ and $I_{[y^{(i)}=c]}$ are the indicators for $w_j$ being present in the claim from $x^{(i)}$ and label $y^{(i)}$ being of class $c$, respectively.

**Optimization of the Overall Bias** Finding the $\alpha$ values which minimize the bias leads us to solving the following objective:

$$\min \left( \sum_{j=1}^{|V|} \max_c (b_j^c) + \lambda \|\vec{\alpha}\|_2 \right). \quad (3)$$

**Re-Weighted Training Objective** We calculate the $\alpha$ values separately from the model optimization, as a pre-processing step, by optimizing Eq. 3. Using these values, the training objective is reweighted from the standard $\sum_{i=1}^{n} L(x^{(i)}, y^{(i)})$ to

$$\sum_{i=1}^{n} (1+\alpha^{(i)}) L(x^{(i)}, y^{(i)}). \quad (4)$$

This re-weighting is independent of the model architecture and can be easily added to any objective, similar to Jiang and Nachum (2019) where they learn instance weights to address labeling bias in datasets.

[4] We use InferSent because BERT, being pretrained on Wikipedia, comprises world knowledge (Talmor et al., 2019).

[5] NOT ENOUGH INFO cases are easy to generate so we focus on the two other labels.

| Source | Claim | Evidence | Label |
|---|---|---|---|
| ORIGINAL | Tim Roth is an English actor. | Timothy Simon Roth (born 14 May 1961) is an English actor and director. | SUPPORTS |
| GENERATED | Tim Roth is an American actor. | Timothy Simon Roth (born 14 May 1961) is an American actor and director. | SUPPORTS |
| ORIGINAL | Aristotle spent time in Athens. | At seventeen or eighteen years of age, he joined Plato's Academy in Athens and remained there until the age of thirty-seven (c. 347 BC). | SUPPORTS |
| GENERATED | Aristotle did not visit Athens. | At seventeen or eighteen years of age, he missed the opportunity to join Plato's Academy in Athens and never visited the place. | SUPPORTS |
| ORIGINAL | Telemundo is a English-language television network. | Telemundo (telemundo) is an American Spanish-language terrestrial television network owned by Comcast through the NBCUniversal division NBCUniversal Telemundo Enterprises. | REFUTES |
| GENERATED | Telemundo is a Spanish-language television network. | Telemundo (telemundo) is an American English-language terrestrial television network owned by Comcast through the NBCUniversal division NBCUniversal Telemundo Enterprises. | REFUTES |
| ORIGINAL | Magic Johnson did not play for the Lakers. | He played point guard for the Lakers for 13 seasons. | REFUTES |
| GENERATED | Magic Johnson played for the Lakers. | He played for the Giants and no other team. | REFUTES |

Table 2: Examples of pairs from the Symmetric Dataset. Each generated claim-evidence pair holds the relation described in the right column. Crossing the generated sentences with the original ones creates two additional cases with an opposite label (see Figure 1).

## 5 Experiments

We use the SYMMETRIC TEST SET to (1) investigate whether top performing sequence classification models trained on the FEVER dataset are actually verifying claims in the context of evidence; and (2) measure the impact of the re-weighting method described in §4 over a classifier.

To achieve the first goal, we use three classifiers. The first is a pre-trained, current FEVER state-of-the-art classifier, **NSMN** (Nie et al., 2019) which is a variation of the ESIM (Chen et al., 2017) model, with a number of additional features, such as contextual word embeddings (Peters et al., 2018). In addition, we train our own **ESIM** model with GloVe embeddings, using the available code from Gardner et al. (2017). The third is a **BERT** classifier[6] that we fine-tune for 3 epochs to classify the relation based on the concatenation of the claim and evidence (with a delimiter token). To measure the impact of our regularization method, we also train the ESIM and BERT models with the re-weighting method.

[6] https://github.com/huggingface/pytorch-pretrained-BERT

**Symmetric Test Set** The full SYMMETRIC TEST SET consists of 956 claim-evidence pairs, created following the procedure described in §3. The new pairs originated from 99 SUPPORTS and 140 REFUTES pairs that were randomly picked from the cases which NSMN correctly predicts.[7] After its generation, we asked two subjects to annotate randomly sampled 285 claim-evidence pairs (i.e. 30% of the total pairs in SYMMETRIC TEST SET) with one label among SUPPORTS, REFUTES or NOT ENOUGH INFO, flagging non-grammatical cases. They agreed with the dataset labels in 94% of cases, attaining a Cohen $\kappa$ of 0.88 (Cohen, 1960). Typos and small grammatical errors were reported in 2% of the cases. Given the small size of this dataset, we only use it as a test set.

**Results** Table 3 summarizes the performance of the three models on the SUPPORTS and REFUTES pairs from the FEVER DEV set and on the created SYMMETRIC TEST SET pairs. All models perform relatively well on FEVER DEV but achieve less than 60% accuracy on the synthetic ones. We

[7] Due to our focus on the performance drop with respect to the newly generated pairs rather than on the intention of multiplying the difficulties for the top performing model.

| Model | FEVER DEV | | GENERATED | |
|---|---|---|---|---|
| | BASE | R.W | BASE | R.W |
| NSMN | 81.8 | - | 58.7 | - |
| ESIM | 80.8 | 76.0 | 55.9 | 59.3 |
| BERT | **86.2** | 84.6 | 58.3 | **61.6** |

Table 3: Classifiers' accuracy on the SUPPORTS and REFUTES cases from the FEVER DEV set and on the GENERATED pairs for the SYMMETRIC TEST SET in the setting of without (BASE) and with (R.W) re-weight.

conjecture that the drop in performance is due to training data bias that is also observed in the development set (see §2) but not in the generated symmetric cases.

Our re-weighting method (§4) helps to reduce the bias in the claims. In Table 4, we revisit the give-away bigrams from Table 1. Applying the weights obtained by optimizing Eq. 3, the weighted distribution of these phrases being associated with a specific label in the training set is now roughly uniform.

The re-weighting method increases the accuracy of the ESIM and BERT models by an absolute 3.4% and 3.3% respectively. One can notice that this improvement comes at a cost in the accuracy over the FEVER DEV pairs. Again, this can be explained by the bias in the training data that translates to the development set, allowing FEVER-trained models to leverage it. Applying the regularization method, using the same training data, helps to train a more robust model that performs better on our test set, where verification in context is a key requirement.

## 6 Related Work

Large scale datasets are fraught with give-away phrases (McCoy et al., 2019; Niven and Kao, 2019). Crowd workers tend to adopt heuristics when creating examples, introducing bias in the dataset. In SNLI (Stanford Natural Language Inference) (Bowman et al., 2015), entailment based solely on the hypothesis forms a very strong baseline (Poliak et al., 2018; Gururangan et al., 2018).

Similarly, as shown by Kaushik and Lipton (2018), reading comprehension models that rely only on the question (or only on the passage referred to by the question) perform exceedingly well on several popular datasets (Weston et al., 2016; Onishi et al., 2016; Hill et al., 2016). To address deficiencies in the SQuAD dataset (Jia and Liang, 2017), researchers have proposed approaches for augmenting the existing dataset (Rajpurkar et al., 2018). In most cases, these augmentations are done manually, and involve constructing challenging examples for existing systems.

| **Bigram** | R.W **LMI**$\cdot 10^{-6}$ | R.W $p(l\|w)$ |
|---|---|---|
| did not | 144 | 0.35 |
| yet to | 30 | 0.33 |
| does not | 67 | 0.35 |
| refused to | 55 | 0.35 |
| failed to | 31 | 0.33 |
| only ever | 9 | 0.31 |
| incapable being | 32 | 0.33 |
| to be | 8 | 0.30 |
| unable to | 10 | 0.32 |
| not have | 41 | 0.35 |

Table 4: Re-weighted statistics ($l$ = REFUTES) for the bigrams from Table 1. The weights were obtained following the optimization of Eq. 3 on the training set which contains three labels.

## 7 Conclusion

This paper demonstrates that the FEVER dataset contains idiosyncrasies that can be easily exploited by fact-checking classifiers to obtain high classification accuracies. Evaluating the claim-evidence reasoning of these models necessitates unbiased datasets. Therefore, we suggest a way to turn the evaluation FEVER pairs into symmetric combinations for which a decision that is solely based on the claim is equivalent to a random guess. Tested on these pairs, FEVER-trained models show degraded performance. To address this problem, we propose a simple method that supports a more robust generalization in the presence of bias.

Moving forward, we suggest using our symmetric dataset in addition to the current retrieval-based FEVER evaluation pipeline. This way, models could be tested both for their evidence retrieval and classification accuracy and for performing the reasoning with respect to the evidence.

## 8 Acknowledgments

We thank the MIT NLP group and the reviewers for their helpful discussion and comments. This work is supported by DSO grant DSOCL18002.

## References


Tariq Alhindi, Savvas Petridis, and Smaranda Muresan. 2018. Where is your evidence: Improving fact-checking by justification modeling. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 85–90, Brussels, Belgium. Association for Computational Linguistics.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642. Association for Computational Linguistics.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced lstm for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1657–1668.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Stefan Evert. 2005. The statistics of word cooccurrences: word pairs and collocations.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. Allennlp: A deep semantic natural language processing platform.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112. Association for Computational Linguistics.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The goldilocks principle: Reading children's books with explicit memory representations. In *International Conference on Learning Representations 2016*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031. Association for Computational Linguistics.

Heinrich Jiang and Ofir Nachum. 2019. Identifying and correcting label bias in machine learning. *arXiv preprint arXiv:1901.04966*.

Divyansh Kaushik and Zachary C. Lipton. 2018. How much reading does reading comprehension require? a critical investigation of popular benchmarks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5010–5015. Association for Computational Linguistics.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

Yixin Nie, Haonan Chen, and Mohit Bansal. 2019. Combining fact extraction and verification with neural semantic matching networks. In *Association for the Advancement of Artificial Intelligence*.

Timothy Niven and Hung-Yu Kao. 2019. Probing neural network comprehension of natural language arguments. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4658–4664, Florence, Italy. Association for Computational Linguistics.

Takeshi Onishi, Hai Wang, Mohit Bansal, Kevin Gimpel, and David A. McAllester. 2016. Who did what: A large-scale person-centered cloze dataset. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2230–2235. The Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis only baselines in natural language inference. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*,

pages 180–191, New Orleans, Louisiana. Association for Computational Linguistics.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789. Association for Computational Linguistics.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. Fever: a large-scale dataset for fact extraction and verification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2016. Towards ai-complete question answering: A set of prerequisite toy tasks. In *International Conference on Learning Representations*.

# Recognizing Conflict Opinions in Aspect-level Sentiment Classification with Dual Attention Networks

**Xingwei Tan, Yi Cai*** and **Changxi Zhu**
School of Software Engineering, South China University of Technology
Guangzhou, China
setanxw5@mail.scut.edu.cn
*ycai@scut.edu.cn

## Abstract

Aspect-level sentiment classification, which is a fine-grained sentiment analysis task, has received lots of attention these years. There is a phenomenon that people express both positive and negative sentiments towards an aspect at the same time. Such opinions with conflicting sentiments, however, are ignored by existing studies, which design models based on the absence of them. We argue that the exclusion of conflict opinions is problematic, for the reason that it represents an important style of human thinking – dialectic thinking. If a real-world sentiment classification system ignores the existence of conflict opinions when it is designed, it will incorrectly mixed conflict opinions into other sentiment polarity categories in action. Existing models have problems when recognizing conflicting opinions, such as data sparsity. In this paper, we propose a multi-label classification model with dual attention mechanism to address these problems.

## 1 Introduction

Aspect-level sentiment classification is a fine-grained sentiment analysis task (Pang et al., 2008). Given aspect categories or target entities in the text, this task aims at inferring the sentiment polarity (e.g., *positive*, *negative*, or *neutral*) of the aspects. A recent study (Kenyon-Dean et al., 2018) on sentiment analysis dataset construction brings attention to sentiments beyond the scope of *positive*, *negative*, and *neutral* categories, indicating that they will benefit the real-world sentiment analysis systems. We find that there is a similar type of sentiment which is labeled as *conflict* in SemEval dataset for aspect-level sentiment classification (Pontiki et al., 2014). *Conflict* label applies when both positive and negative sentiment is expressed about an aspect. For example, the sentence "*The atmosphere is attractive, but a little uncomfortable.*" expresses *conflict* sentiment towards *ambience* aspect. However, most of existing studies ignore *conflict* opinions, for the reason that they are sparse in the datasets (Tang et al., 2016b; He et al., 2018). In this work, we argue that the exclusion of *conflict* opinions is problematic.

The *conflict* opinions are the production of an important style of human thinking – dialectic thinking. Dialectic thinking, unlike the thinking constrained by the laws of formal logic, shows a degree of tolerance of contradiction (Peng and Nisbett, 1999). If a real-world sentiment classification system ignores the existence of *conflict* opinions when it is designed, its predicted *positive* opinions may contain negative expressions and its predicted *negative* opinions may contain positive expressions. Considering the existence of sentiment expressions of both sites, it is unreasonable to put *conflict* opinions into either *positive* or *negative* sentiment classes.

There are problems when including *conflict* opinions in aspect-level sentiment classification. Based on our observation, existing methods have significant lower accuracy on recognizing *conflict* sentiment than recognizing other sentiments. We analyse that the reasons are as follows: 1) *Conflict* opinions are sparse in dataset, which is difficult for a model to learn its class-specific features during training; 2) Existing models have difficulty in recognizing the complex expressions of *conflict* sentiment, which are the combination of positive and negative sentiment expressions.

In this paper, we propose a method to address the two problems. Firstly, we model the task as a multi-label classification problem. In contrast to mutually exclusive 4-class (*positive*, *negative*, *neutral*, and *conflict*) classification, our model predicts two labels for each aspect: 1) Whether positive sentiment is expressed towards the aspect; 2)

*Corresponding author



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3426–3431,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

Whether negative sentiment is expressed towards the aspect. Then, we transform the 2-label predicted results into 4-class labels by rules (e.g., if both positive and negative expressions exist, then apply *conflict*). During training, the 4-class training labels in dataset are transformed into 2-label targets. In this way, we utilize the relation between *conflict* label and *positive*/*negative* labels, which makes the model learn to recognize *conflict* opinions from abundant *positive* and *negative* data. It is also based on the fact that neural network is good at recognizing positive or negative sentiments, which are indicated by compact and concrete expressions. On the contrary, we find that the expressions of *conflict* opinions usually are lengthy and implicit. Faced with *conflict* opinions, existing attention-based models (Wang et al., 2016) (with single attention) usually attend on a portion of a *conflict* expression and ignore the other portion, which make its predicted result prone to be *positive* or *negative*. Instead, we use two independent attentions in the proposed model to focus on positive or negative expressions separately.

Our contributions are summarized as follows: 1) To the best of our knowledge, this is the first work that discuss the necessity of including *conflict* opinions in aspect-level sentiment classification task; 2) We propose a model that can recognize the complex expressions of *conflict* opinions and deal with sparsity problem of *conflict* data; 3) The experiment results on SemEval dataset indicate that our model performs well on recognizing *conflict* opinions and outperforms all baselines.

## 2 Related Work

Sentiment analysis is a frequently studied topic in artificial intelligence (Cambria, 2016). Different from previous work about sentiment analysis, Kenyon-Dean et al. (2018) states that there are opinions cannot be properly annotated as *positive*, *negative*, or *neutral* sentiment polarities. Their experiment shows that existing classifiers have significant lower accuracy on recognizing sentiments in these opinions. Their work is about sentence-level sentiment analysis and does not propose a model. Our work is different from theirs since we propose a model capable of recognizing *conflict* opinions in existing aspect-level sentiment classification dataset.

In recent years, lots of efforts have been made to imporve aspect-based sentiment analysis (Wang et al., 2014; Zhao et al., 2015). Traditional methods in aspect-level sentiment classification leveraged sentiment lexicons (Mohammad et al., 2013), while recent work tends to use neural network to generate text representations automatically. TD-LSTM and TC-LSTM (Tang et al., 2016a) encode contextual information with respect to the target. Methods based on memory network (Tang et al., 2016b; Zhu and Qian, 2018) instead generate aspect-related representation. Attention-based recurrent neural network (Wang et al., 2016; Chen et al., 2017; Cheng et al., 2017) selectively attends to aspect-related regions and calculates the weighted sum of the hidden states of these regions. Xue et al. (2018) instead utilize gated convolutional neural network to extract aspect-related representations, which increase computational efficiency. Our work differs from these works since we address two problems of recognizing *conflict* opinion which they ignore.

## 3 Model: Dual Attention-based GRU

### 3.1 Network Architecture

The architecture of the proposed Dual Atteniton-based GRU (D-AT-GRU) is illustrated as Fig. 1. Given a sentence with $L$ words and an aspect (or aspect term) as inputs, the model first transforms them into word embeddings $[w_1, ..., w_L]$ and aspect embedding $v_a$. Then, it extracts aspect-related text features through recurrent neural network and dual attention mechanism. Lastly, it produces 2-label predictions through classification layers and transforms them into 4-class labels.

**Aspect embedding.** We use an embedding matrix $V_A \in \mathbb{R}^{|A| \times d_a}$ to represent aspects, where $d_a$ is the dimension of aspect embedding and $|A|$ is the number of aspects in dataset. Each row in $V_A$ is a representation vector $v_a$ for aspect $a \in A$. The aspect embeddings are initialized randomly and learned as parameters.

**Text feature extraction.** We use Gated Recurrent Unit (GRU) (Cho et al., 2014) to extract contextual information of words in our model. Long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) is an alternative choice, which is often used in previous models (Wang et al., 2016; Chen et al., 2017; He et al., 2018). However, we find that GRU performs better than LSTM in our model. Thus, we choose GRU. Let matrix $H = [h_1, ..., h_L] \in \mathbb{R}^{L \times d}$ denotes the hid-

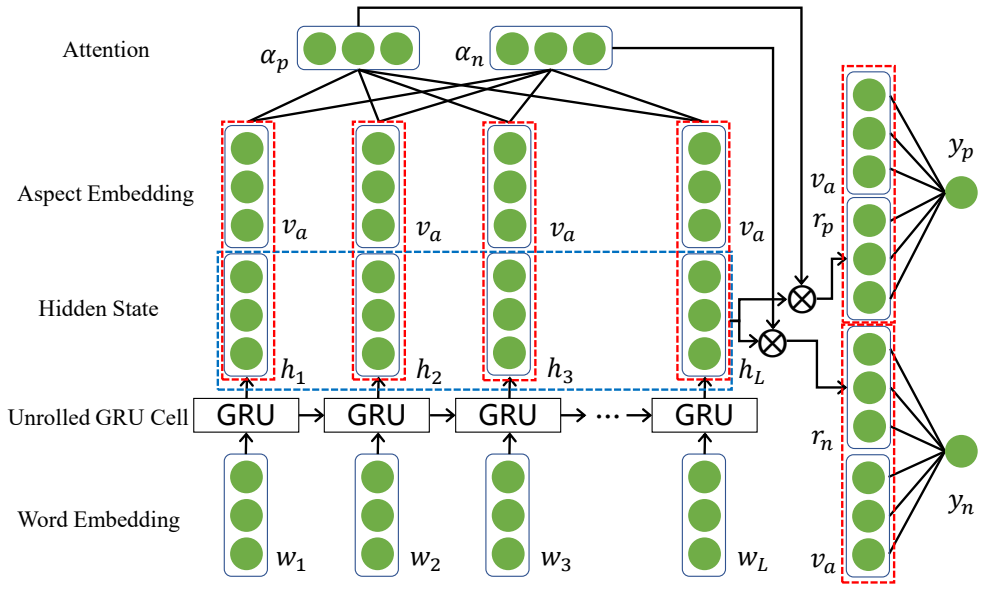


Figure 1: The network architecture of D-AT-GRU.

den states produced by GRU.

**Attention mechanism.** Based on the hidden states $H$ and the embedding $v_a$ of given aspect, attention mechanism gives importance weights which indicate aspect-related context region. Given that a *conflict* opinion consist of positive and negative expressions, attention usually fails to capture the full expression and only attends to a portion of it. Affected by previous *positive* or *negative* training samples, attention tends to think that just a portion of the *conflict* expression is enough to determine sentiment label and omit the other portion which also contains sentiment words. We therefore propose to use two attentions separately: one is responsible for finding positive expression, the other is responsible for finding negative expression. The positive attention is calculated by:

$$g_i^p = u_p^T tanh(W_p[h_i; v_a] + b_p), \quad (1)$$

$$\alpha_i^p = \frac{exp(g_i^p)}{\sum_{j=1}^{L} exp(g_j^p)}, \quad (2)$$

where $W_p \in \mathbb{R}^{d \times 2d}, u_p \in \mathbb{R}^d$ and $b_p \in \mathbb{R}^d$ are parameters. The negative attention is calculated through following functions:

$$g_i^n = u_n^T tanh(W_n[h_i; v_a] + b_n), \quad (3)$$

$$\alpha_i^n = \frac{exp(g_i^n)}{\sum_{j=1}^{L} exp(g_j^n)}, \quad (4)$$

where $W_n \in \mathbb{R}^{d \times 2d}, u_n \in \mathbb{R}^d$ and $b_n \in \mathbb{R}^d$ are parameters.

**Regularization term.** For the reason that positive sentiment and negative sentiment usually are expressed in different regions, the two attentions should not overlap. Thus, we need to add a regularization term to make the two sets of attention scores differentiate from each other. We compare KL-divergence (Kullback and Leibler, 1951) and orthogonal regularization (He et al., 2017) in experiments. We observed no consistent difference in performance regarding to sentiment analysis between these two terms. But the time cost of computing KL-divergence is 2.7 times as much as computing orthogonal regularization for each batch. Therefore, we utilize orthogonal regularization (He et al., 2017):

$$\mathcal{R}_o = ||M^T M - I||_2, \quad (5)$$

where $M = [\alpha^p; \alpha^n], M \in \mathbb{R}^{2 \times L}$ contains attention scores.

**Text representation.** According to the positive and negative attention scores, corresponding aspect-related text representations are calculated through the weighted sum of hidden states $H$:

$$v_p = \sum_{i=1}^{L} \alpha_i^p h_i, \quad (6)$$

$$v_n = \sum_{i=1}^{L} \alpha_i^n h_i. \quad (7)$$

**Classification layer.** An expression may have various meanings when describing different aspects. Therefore, we take aspect information into

account in classification layer by concatenating text representations with aspect embeddings.

$$y_p = \sigma(W_{py}[v_p; v_a] + b_{py}), \quad (8)$$
$$y_n = \sigma(W_{ny}[v_n; v_a] + b_{ny}), \quad (9)$$

where $y_p$ is predicted positive sentiment distribution, $y_n$ is predicted negative sentiment distribution; $W_{py}, W_{ny} \in \mathbb{R}^{2d}$ and $b_{py}, b_{ny} \in \mathbb{R}^1$ are parameters.

### 3.2 Objection Function

The model can be trained in an end-to-end way by backpropagation, where the objective function is the sum of the binary cross-entropy losses of positive and negative sentiments:

$$\mathcal{L}_p = -\hat{y}_p log(y_p) - (1 - \hat{y}_p) log(1 - y_p), \quad (10)$$
$$\mathcal{L}_n = -\hat{y}_n log(y_n) - (1 - \hat{y}_n) log(1 - y_n), \quad (11)$$

where $\hat{y}_p, \hat{y}_n$ are target distributions. During training, 4-class sentiment labels are transformed into 2-label targets (e.g., *conflict* is transformed into $(\hat{y}_p, \hat{y}_n) = (1, 1)$). The goal of training is to minimize the following function:

$$\mathcal{L} = \mathcal{L}_p + \mathcal{L}_n + \lambda \mathcal{R}_o, \quad (12)$$

where $\lambda$ is a hyperparameter used to balance the weight of orthogonal regularization. We set $\lambda$ to 1 in our experiment.

### 3.3 Label Transformation

Given the predicted distributions $y_p, y_n$, we transform them into 4-class labels. If $y_p > p, y_n > p$, the predicted label will be *conflict*. If $y_p \leq p, y_n \leq p$, the predicted label will be *neutral*. If $y_p \leq p, y_n > p$, the predicted label will be *negative*. If $y_p > p, y_n \leq p$, the predicted label will be *positive*. In this paper, we directly use $p = 0.5$ to minimize human interference, although it can be tuned manually as a hyperparameter.

## 4 Experiment

### 4.1 Dataset and Preparation

We experiment on SemEval 2014 datasets (Pontiki et al., 2014) (statistics shown in Table 1) to evaluate the proposed model. For the reason that there are too few *conflict* test instances regarding aspect term to show significance, we choose to experiment on aspect category data. Our model is compared with the state-of-the-art methods in aspect category sentiment classification, which are adapted to 4-class classification.

In our experiment[1], dimension sizes $d, d_a, d_w$ are all set to 300. Word embeddings are initialized with 300-dimension pre-trained GloVe vectors (Pennington et al., 2014). All the other parameters are initialized by sampling from a normal distribution $\mathcal{N}(0, 0.01)$. We use Adagrad (Duchi et al., 2011) with a batch size of 10 samples, initial learning rate of 0.01, and maximum epochs of 30. We use development set chosen by Tay et al. (2018). The patience value of early-stopping is 3.

| | *positive* | *negative* | *neutral* | *conflict* |
|---|---|---|---|---|
| Train | 1759 | 704 | 442 | 187 |
| Dev | 420 | 135 | 58 | 8 |
| Test | 657 | 222 | 94 | 52 |

Table 1: Statistics of datasets.

### 4.2 Result and Analysis

| **Model** | Overall | Conflict |
|---|---|---|
| AT-LSTM | 77.13% | 11.54% |
| ATAE-LSTM | 78.00% | 23.08% |
| GCAE | 78.30% | 25.00% |
| AT-GRU | 77.22% | 19.23% |
| AT-GRU 2-label | 77.02% | 26.92% |
| D-AT-GRU w/o orthogonal | 77.22% | 26.92% |
| D-AT-GRU | **78.50%** | **40.38%** |

Table 2: Experiment results on SemEval 2014 dataset.

Table 2 shows that D-AT-GRU model outperforms all baseline methods. Given that AT-LSTM (Wang et al., 2016) has strong correlation to our base model (AT-GRU), their work can be categorized as a baseline to our model. In contrast, the results prove that the additional components are helpful to recognize *conflict* opinions. We also compare our model to the recently proposed GCAE (Xue and Li, 2018), which is based on gated CNN. D-AT-GRU performs competitively with GCAE overall and significantly better on *conflict* category.

We also compare with ablated versions of D-AT-GRU: 1) Replace the LSTM in AT-LSTM with

[1]Code available: https://github.com/tanxw17/DATGRU

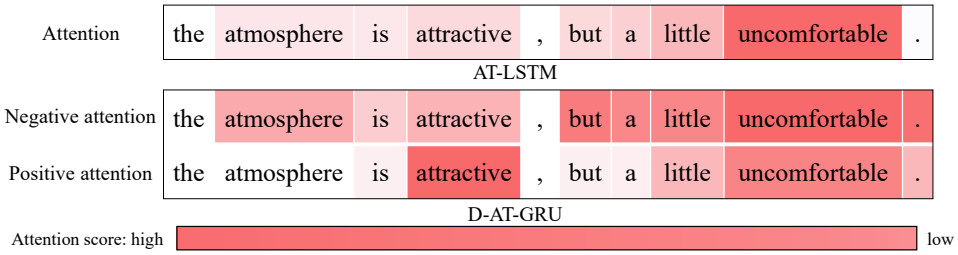


Figure 2: The visualizations of attention.

GRU and directly do 4-class classification (AT-GRU); 2) Use AT-GRU with two separate classification layers to do 2-label classification (AT-GRU 2-label); 3) Remove orthogonal regularization (D-AT-GRU w/o orthogonal). Their performance suffers from the ablations, which proves that every component is essential for the proposed model.

We test KL-divergence (Kullback and Leibler, 1951) and orthogonal regularization (He et al., 2017) as regularization terms to make positive attention and negative attention differentiate from each other. The experiments were conducted on a NVIDIA 1080 Ti. The results are shown in Table 3.

| **Regularizer** | Overall | Conflict | Time |
|---|---|---|---|
| Orthogonal | 78.50% | 40.38% | 0.15 ms |
| KL-divergence | 78.40% | 40.38% | 0.40 ms |

Table 3: Experiment results about different regularization terms.

### 4.3 Case Study

Figure 2 is a test case in our experiment. AT-LSTM attends to "*uncomfortable*" and ignores the existence of "*attractive*", which causes it to incorrectly predict *negative* label. With our D-AT-GRU model, the negative attention finds "*uncomfortable*" in the second part of the sentence, which makes the model predict that there is negative sentiment; the positive attention instead gives the highest score to "*attractive*", which leads the model to predict that there is positive sentiment. Given the existence of both positive and negative sentiments, D-AT-GRU correctly predict *conflict* label. This shows that the dual attention mechanism of D-AT-GRU model can more accurately capture the complete expression of a *conflict* opinion.

## 5 Conclusion

In this paper, we present a method to recognize *conflict* opinions in aspect-level sentiment classification task. By transforming the problem into recognizing simpler sentiments, we alleviate the sparsity problem of *conflict* data. Our model utilize dual attention mechanism and orthogonal regularization, which are capable of recognizing the complex expressions of *conflict* opinions. Our experiment on SemEval dataset demonstrates the effectiveness of the proposed model.

## Acknowledgments

This work was supported by the Fundamental Research Funds for the Central Universities, SCUT (No. 2017ZD048, D2182480), the Science and Technology Planning Project of Guangdong Province (No.2017B050506004), the Science and Technology Program of Guangzhou (No. 201704030076,201802010027,201902010046).
The research described in this paper has been supported by a collaborative research grant from the Hong Kong Research Grants Council (project no. C1031-18G).

## References

Erik Cambria. 2016. Affective computing and sentiment analysis. *IEEE Intelligent Systems*, 31:102–107.

Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. 2017. Recurrent attention network on memory for aspect sentiment analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 452–461.

Jiajun Cheng, Shenglin Zhao, Jiani Zhang, Irwin King, Xin Zhang, and Hui Wang. 2017. Aspect-level sentiment classification with heat (hierarchical attention) network. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 97–106. ACM.

Kyunghyun Cho, Bart van Merrienboer, aglar Gülehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2017. An unsupervised neural attention model for aspect extraction. In *ACL*.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2018. Effective attention modeling for aspect-level sentiment classification. In *COLING*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Kian Kenyon-Dean, Eisha Ahmed, Scott Fujimoto, Jeremy Georges-Filteau, Christopher Glasz, Barleen Kaur, Auguste Lalande, Shruti Bhanderi, Robert Belfer, Nirmal Kanagasabai, et al. 2018. Sentiment analysis: Its complicated! In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1886–1895.

Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.

Saif Mohammad, Svetlana Kiritchenko, and Xiao-Dan Zhu. 2013. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. In *SemEval@NAACL-HLT*.

Bo Pang, Lillian Lee, et al. 2008. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, 2(1–2):1–135.

Kaiping Peng and Richard E Nisbett. 1999. Culture, dialectics, and reasoning about contradiction. *American psychologist*, 54(9):741.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 27–35.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2016a. Effective lstms for target-dependent sentiment classification. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3298–3307.

Duyu Tang, Bing Qin, and Ting Liu. 2016b. Aspect level sentiment classification with deep memory network. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 214–224.

Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2018. Learning to attend via word-aspect associative fusion for aspect-based sentiment analysis. In *AAAI*.

Tao Wang, Yi Cai, Ho fung Leung, Raymond Y. K. Lau, Qing Li, and Huaqing Min. 2014. Product aspect extraction supervised with online domain knowledge. *Knowl.-Based Syst.*, 71:86–100.

Yequan Wang, Minlie Huang, Li Zhao, et al. 2016. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 606–615.

Wei Xue and Tao Li. 2018. Aspect based sentiment analysis with gated convolutional networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2514–2523.

Y. Zhao, B. Qin, and T. Liu. 2015. Creating a fine-grained corpus for chinese sentiment analysis. *IEEE Intelligent Systems*, 30(1):36–43.

Peisong Zhu and Tieyun Qian. 2018. Enhanced aspect level sentiment classification with auxiliary memory. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1077–1087.

# Investigating Dynamic Routing in Tree-Structured LSTM for Sentiment Analysis

**Jin Wang[1], Liang-Chih Yu[2,4], K. Robert Lai[3,4] and Xuejie Zhang[1]**
[1]School of Information Science and Engineering, Yunnan University, Yunnan, P.R. China
[2]Department of Information Management, Yuan Ze University, Taiwan
[3]Department of Computer Science & Engineering, Yuan Ze University, Taiwan
[4]Innovation Center for Big Data and Digital Convergence Yuan Ze University, Taiwan
Contact: lcyu@saturn.yzu.edu.tw, xjzhang@ynu.edu.cn

## Abstract

Deep neural network models such as long short-term memory (LSTM) and tree-LSTM have been proven to be effective for sentiment analysis. However, sequential LSTM is a bias model wherein the words in the tail of a sentence are more heavily emphasized than those in the header for building sentence representations. Even tree-LSTM, with useful structural information, could not avoid the bias problem because the root node will be dominant and the nodes in the bottom of the parse tree will be less emphasized even though they may contain salient information. To overcome the bias problem, this study proposes a capsule tree-LSTM model, introducing a dynamic routing algorithm as an aggregation layer to build sentence representation by assigning different weights to nodes according to their contributions to prediction. Experiments on Stanford Sentiment Treebank (SST) for sentiment classification and EmoBank for regression show that the proposed method improved the performance of tree-LSTM and other neural network models. In addition, the deeper the tree structure, the bigger the improvement.

## 1 Introduction

In sentiment analysis, word embeddings (Mikolov et al., 2013a; Mikolov et al., 2013b; Pennington et al., 2014) and sentiment embeddings (Tang et al., 2016; Yu et al., 2018a; Yu et al., 2018b) have become a fundamental component to build deep neural networks such as convolutional neural networks (CNN) (Kalchbrenner et al., 2014; Kim, 2014), recurrent neural networks (RNN) (Graves, 2012; Irsoy and Cardie, 2014), gated recurrent unit (GRU) (Cho et al., 2014), and long short-term memory (LSTM) (Tai et al., 2015; Wang et al., 2015). Given a variable-length text, one challenge of using these neural networks is to compose individual word vectors into sentence vectors with the same length (Iyyer et al., 2015; Joulin et al., 2016; Bojanowski et al., 2016).

The sequential neural networks such as RNN, GRU, and LSTM are commonly used due to their ability to capture long-distance dependency in sequential texts. However, these methods belong to the biased model, where the words in the tail of a sentence are more heavily emphasized than those in the header for building sentence representations. As shown in Fig. 1(a), the priority for each word vector will be "*fantastic* ≻ *really* ≻ *is* ≻ *story* ≻ *this*". This prioritization seems satisfactory for this sentence, but note that the key components could appear anywhere in the sentence rather than necessarily at the end.

To improve the abovementioned sequential models, Tai et al. (2015) and Huang et al. (2017) proposed a tree-LSTM model to introduce useful structural information from sentence parse trees. However, the tree-LSTM also heavily emphasizes the root node in the tree to build sentence representations. That is, words that are closed to the root will be given higher priority than words that are far away from the root. As shown in Fig. 1(b), the priority of word vectors would be "*this* = *story* = *is* ≻ *really* = *fantastic*". This example shows that the tree-LSTM still could not avoid the bias problem because the nodes (e.g., *fantastic*) that contribute more to the prediction but lie in the leaf node at the bottom of the parse tree will be less emphasized.

To overcome the bias problem that may arise in the tree-LSTM, this study proposes a capsule tree-LSTM model. In spired by recent promising work of capsule network (Sabour et al., 2017), the



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3432–3437, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

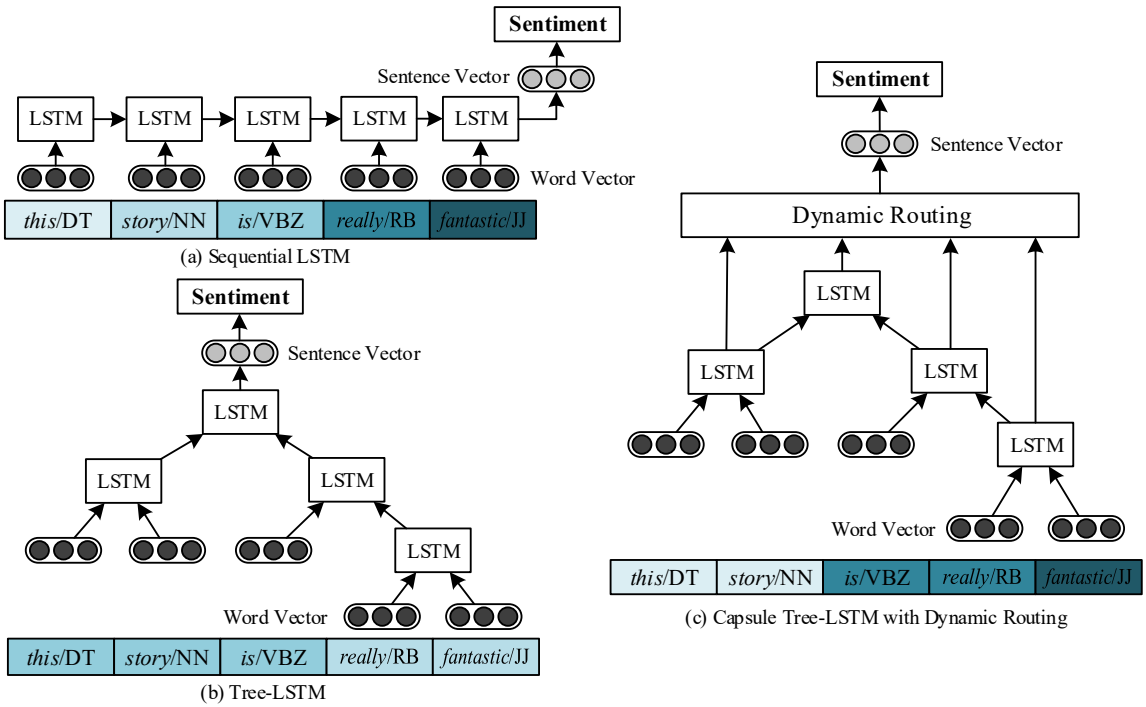


**Figure 1**: Illustrative examples of different LSTM models for sentiment analysis. A deeper color indicates more weight is assigned to the word according to its contribution to the prediction result.

proposed method introduces a dynamic routing algorithm to consider all non-leaf nodes to build sentence vectors, instead of using the root alone in the tree-LSTM. In addition, different nodes will receive different weights according to their contributions to the prediction task. Unlike self-attention (Lin et al., 2017; Yang et al., 2016), which applies a fixed policy without considering the state of the final sentence vectors, the task of assigning weights in the proposed model is considered to be a routing issue to iteratively determine how much information can be passed from non-leaf nodes in the tree to the vector presentation of the sentence, according to the state of final output. For example, in the aforementioned example text, it would be useful for the model to emphasize *fantastic* that contains the most salient information, even when the word lies at the bottom of the parser tree. Based on the dynamic routing algorithm, the priority of the word vector in the proposed model would be "*fantastic* ≻ *really*=*is* ≻ *this*=*story*". The proposed method is evaluated through both sentiment classification and regression tasks to determine whether dynamic routing can improve the performance of the tree-LSTM and other neural network models.

The rest of this paper is organized as follow. Section 2 describes the proposed capsule tree-LSTM model with dynamic routing. Section 3 summarizes the evaluation results. Conclusions are presented in Section 4.

## 2 Capsule Tree-LSTM Model

Figure 1(c) shows the framework of the proposed model. First, the given sentence is parsed as a tree-structured topology. The vector representation of this sentence is then generated by composing the word vectors of all non-leaf nodes in the tree according to their weights learned by the dynamic routing algorithm. Finally, the composed sentence vector is used for sentiment prediction.

### 2.1 Tree-structured LSTM

Given a binary parser tree, the leaf nodes are words and the non-leaf nodes are multi-word phrases. Let $C(j)$ denotes the set of left and right child nodes of a non-leaf node $j$. Different from the sequential LSTM, the hidden state $\tilde{h}$ of the non-leaf node $j$ is the composition of its left and right child nodes, defined as

$$\tilde{h} = \begin{bmatrix} \mathbf{h}_{t-1}^{left} \\ \mathbf{h}_{t-1}^{right} \end{bmatrix} + b_c \quad (1)$$

where $\mathbf{h}_{t-1}^{left}$ and $\mathbf{h}_{t-1}^{right}$ respectively denote the hidden states of left and right child nodes,

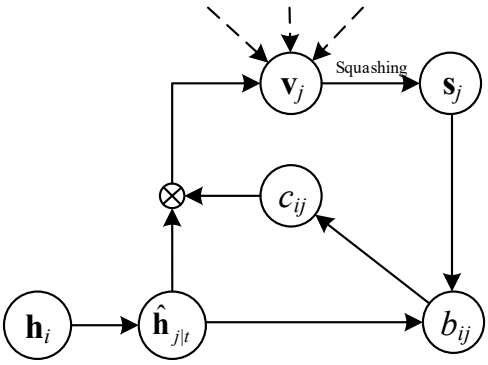


**Figure 2**: Detailed dynamic routing process

$W_c \in \Re^{d\times 2d}$ is a composition matrix, and $b_c$ is a bias. The tree-LSTM transition equations of node $j$ are defined as

- Gates

$$\begin{aligned} i_t^j &= \sigma(W_{xi}x_t + W_{hi}\tilde{} \\ f_t^k &= \sigma(W_{xf}x_t + W_{hf}\mathbf{h}_{t-1}^k + b_f) \\ o_t^j &= \sigma(W_{xo}x_t + W_{ho}\tilde{} \end{aligned} \tag{2}$$

- Input transform

$$c\_in_t = \tanh(W_{xc}x_t + W_{hc}\mathbf{h}_{t-1} + b_{c\_in}) \tag{3}$$

- Memory update

$$\begin{aligned} c_t^j &= i_t^j \otimes c\_in_t^j + \sum_{k\in C(j)} f_t^k \otimes c_{t-1}^k \\ \mathbf{h}_t^j &= o_t^j \otimes \tanh(c_t^j) \end{aligned} \tag{4}$$

where $i_t$, $f_t$, $o_t$, and $c_t$ respectively denote the input gate, forget gate, output gate, and memory cell of node $j$, $x_t$ denotes the input word vector at the time step $t$, $\sigma$ denotes the logistic sigmoid function, $W$ and $b$ respectively denote the weights and bias, and $\otimes$ denotes element-wise multiplication. To integrate the sequence information in the output layer, the order of non-leaf hidden states to form the input matrix of dynamic routing layer is a key consideration. Here, we used the in-order traversal of depth-first search algorithm on the tree-structured topology. The output matrix is composed of the hidden states of all non-leaf nodes, defined as $\mathbf{H}=[\mathbf{h}_1,\mathbf{h}_2,...,\mathbf{h}_T] \in \Re^{T\times d_h}$, where $T$ and $d_h$ respectively denote the number and dimensionality of the hidden states. The obtained hidden matrix is then fed to the aggregation layer.

## 2.2 Dynamic Routing

To compose all word vectors to generate sentence vectors, the tree-structured LSTM model uses the hidden states of all non-leaf nodes to obtain the weights for all nodes through the dynamic routing algorithm.

Taking the hidden states of all non-leaf nodes as the input vectors, the goal of dynamic routing is to encode the sentiment information of those vectors into a fixed-length sentence vector,

$$\mathbf{s}_{cap} = [\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_J] \tag{5}$$

Inspired by the definition of capsule networks, we implement two layers of capsules (i.e., $\mathbf{H}=[\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_I]$ denotes the input capsules and $\mathbf{s}=[\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_J]$ denotes the output capsules) to perform dynamic routing. The output capsule $\mathbf{s}_j$ is produced from a non-linear "squashing" function to ensure $|\mathbf{s}_j| \in (0,1)$ as a probability,

$$\mathbf{s}_j = \frac{||\mathbf{v}_j||^2}{1+||\mathbf{v}_j||^2}\frac{\mathbf{v}_j}{||\mathbf{v}_j||} \tag{6}$$

where $\mathbf{s}_j$ is the vector output of capsule $j$, $\mathbf{v}_j$ is the total input, which is a weighted sum over all "prediction vectors" $\hat{\mathbf{h}}_{j|t}$ from the capsules in the layer below,

$$\mathbf{v}_j = \sum_t c_{tj}\hat{\mathbf{h}}_{j|t} \tag{7}$$

where coupling coefficients $c_{tj}$ are the probability distributions of capsule $j$ which are computed using a softmax function so that all capsules in the layer above sum to 1 so that the sentiment information $\hat{\mathbf{h}}_{j|t}$ is obtained by multiplying the input vector $\mathbf{h}_t$ by a weighted matrix $W_{tj}$, defined as,

$$\hat{\mathbf{h}}_{j|t} = W_{tj}\mathbf{h}_t \tag{8}$$

Here, the coupling coefficients $c_{tj}$ are determined by the iterative dynamic routing process,

$$c_{tj} = \frac{\exp(b_{tj})}{\sum_k \exp(b_{tk})} \tag{9}$$

$$b_{tj} \coloneqq b_{tj} + \hat{\mathbf{h}}_{j|t}\cdot\mathbf{s}_j \tag{10}$$

where $b_{tj}$ is the log probabilities, initialized with 0. The detailed iterative process of learning the weights between capsules in two layers for each non-leaf node is shown in Fig. 2.

In Eq. (7), the capsules in the above layer try to learn contribution weights $c_{tj}$ (i.e., coupling coefficients) for the capsules in the below layer. The updated information in $b_{tj}$ comes from the scalar product $\hat{\mathbf{h}}_{j|t}\cdot\mathbf{s}_j$. The coupling coefficients $c_{ij}$ are iteratively refined by measuring the agreement between the current output $\mathbf{s}_j$ of output capsule $j$ in the above layer and the prediction $\hat{\mathbf{h}}_{j|t}$ made by input capsule $i$. If the margin between the two vectors and $\mathbf{s}_j$ is very large, the scalar product of

| SST (Classification) | Binary | Fine-grained |
|---|---|---|
| CNN | 87.2 | 48.0 |
| GRU | 87.2 | 48.2 |
| LSTM | 84.9 | 46.4 |
| Bi-GRU | 87.4 | 48.5 |
| Bi-LSTM | 87.5 | 49.1 |
| 2-Layer Bi-GRU | 87.1 | 48.7 |
| 2-Layer Bi-LSTM | 87.2 | 48.5 |
| Tree-LSTM | 87.5 | 49.7 |
| Attention GRU | 87.8 | 49.5 |
| Attention LSTM | 87.6 | 49.2 |
| Attention Tree-LSTM | 88.2 | 49.8 |
| Capsule Tree-LSTM | **90.2** | **51.6** |

**Table 1**: Results of different methods on SST.

| EmoBank (Regression) | Valence | | Arousal | |
|---|---|---|---|---|
| | MAE | *r* | MAE | *r* |
| CNN | 0.581 | 0.521 | 0.560 | 0.519 |
| GRU | 0.523 | 0.589 | 0.527 | 0.532 |
| LSTM | 0.518 | 0.592 | 0.528 | 0.534 |
| Bi-GRU | 0.514 | 0.591 | 0.497 | 0.543 |
| Bi-LSTM | 0.506 | 0.610 | 0.498 | 0.578 |
| 2-Layer Bi-GRU | 0.505 | 0.612 | 0.485 | 0.573 |
| 2-Layer Bi-LSTM | 0.498 | 0.615 | 0.475 | 0.588 |
| Tree-LSTM | 0.483 | 0.625 | 0.468 | 0.602 |
| Attention GRU | 0.492 | 0.622 | 0.477 | 0.585 |
| Attention LSTM | 0.495 | 0.620 | 0.472 | 0.589 |
| Attention Tree-LSTM | 0.475 | 0.629 | 0.465 | 0.596 |
| Capsule Tree-LSTM | **0.462** | **0.639** | **0.454** | **0.622** |

**Table 2**. Results of different methods on EmoBank.

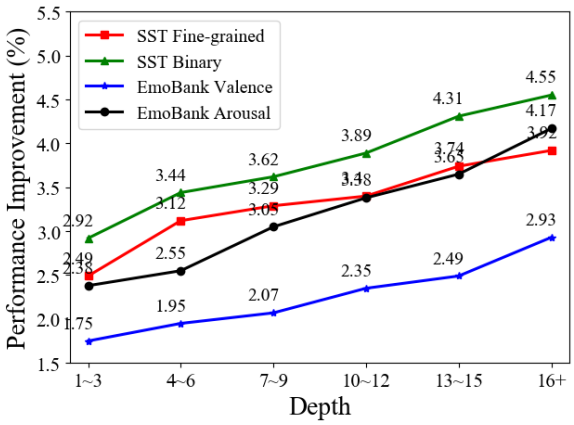


**Figure 3**: Effect of dynamic routing algorithm.

those vectors will be large, which will also result in an update on the coupling coefficient $c_{tj}$.

## 3 Experimental Results

**Datasets.** This experiment used two datasets for evaluation. i) The Stanford Sentiment Treebank (SST) (Socher et al., 2013) is used for sentiment classification. It contains 6920/872/1821 sentences for the train/dev/test sets with binary labels (positive/negative) and 8544/1101/2210 sentences with fine-grained labels (very negative/negative/ neutral/positive/very positive). ii) EmoBank (Buechel and Hahn, 2017; Buechel and Hahn, 2016) is used for sentiment regression to predict valence-arousal (VA) values (Wang et al., 2016b; Yu et al., 2016). It contains 10,000 sentences with real-valued VA ratings in the range of (1, 9), where the valence refers to the degree of positive and negative sentiment and the arousal refers to the degree of calm and excitement. The provided ratings have Reader and Writer perspectives, and the Reader was adopted as the ground-truth ratings due to its superiority reported in (Buechel and Hahn, 2017). We performed 5-fold cross-validation (6:2:2) on the EmoBank dataset.

**Evaluation Metrics.** For SST, the evaluation metric is accuracy for both binary and fine-grained classification. For EmoBank, we used the Pearson correlation coefficient (*r*) and mean absolute error (MAE). A higher *r* or a lower MAE value indicates better prediction performance.

**Implementation Details.** Several deep neural networks were implemented for comparison, including CNN, GRU, LSTM, and tree-LSTM. For the sequential models (GRU and LSTM), we additionally implemented an enhanced version using a bi-directional strategy and 2-layer stacked architecture. To investigate the performance of self-attention, we also implement a self-attention layer by taking as input the hidden states of all non-leaf nodes, to form an attention Tree-LSTM model (Kokkinos and Potamianos, 2017). For word vectors, we used GloVe pre-trained on the 840B Common Crawl corpus (Pennington et al., 2014). The respective dimensionality values of the word vectors and hidden states were 300 and 120. For classification and regression tasks, *softmax* and *linear decoder* (Wang et al., 2016a) activation function are respectively applied as the output layer.

**Comparative Results.** Tables 1 and 2 respectively show the comparative results of different methods for SST and EmoBank. Both the enhanced bi-directional and 2-layer GRU/LSTM outperformed the standard GRU, LSTM, CNN, and the Tree-LSTM with structural information achieved better performance than all of them for both classification and regression tasks. Once the dynamic routing algorithm was introduced, the proposed Capsule Tree-LSTM further improved the performance of Tree-LSTM

(with attention).

Figure 3 shows the detailed analysis of the effect of dynamic routing. The test sentences were first divided into several groups according to their depths in the parse trees (e.g., the depth of the example sentence in Fig. 1 is three). The performance improvement of Capsule Tree-LSTM over Tree-LSTM was then calculated for each group. The results show that the performance improvements increased with the increase of the depth. The reason is that the Tree-LSTM may suffer from a more serious bias problem for sentences with a deeper tree structure because the useful nodes in the deeper levels tend to be ignored. Conversely, the Capsule Tree-LSTM can assign a higher weight to the nodes that contribute more to the prediction even though they lie in the leaf node at the bottom of the tree.

## 4 Conclusion

This study presents a capsule tree-LSTM model for sentiment classification and regression. The proposed method uses dynamic routing algorithm to automatically learn the weights of each node to compose sentence representations. Experimental results show that the proposed method yielded better results than convolutional (CNN), sequential (LSTM and GRU), structural (tree-LSTM) and self-attention neural networks. Future work will conduct more detailed analysis to continue enhancing the proposed method.

## Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant No. 61966038, 61702443 and 61762091, and in part by the Ministry of Science and Technology, Taiwan, ROC, under Grant No. MOST107-2628-E-155-002-MY3 and MOST107-2218-E-006-008. The authors would like to thank the anonymous reviewers for their constructive comments.

## References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Sven Buechel and Udo Hahn. 2016. Readers vs. writers vs. texts : Coping with different perspectives of text understanding in emotion annotation. In *Proceedings of the 11th Linguistic Annotation Workshop in EACL 2017*, pages 1–12.

Sven Buechel and Udo Hahn. 2017. EmoBank: Studying the impact of annotation perspective and representation format on dimensional emotion analysis. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL-2017)*, pages 578–585.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Alex Graves. 2012. *Supervised sequence labelling*. Springer Berlin Heidelberg.

Minlie Huang, Qiao Qian, and Xiaoyan Zhu. 2017. Encoding syntactic knowledge in neural networks for sentiment classification. *ACM Transactions on Information Systems*, 35(3):1–27.

Ozan Irsoy and Claire Cardie. 2014. Opinion mining with deep recurrent neural networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2014)*, pages 720–728.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL-2015)*, pages 1681–1691.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL-2014)*, pages 655–665.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of International Conference of Empirical Methods on Natural Language Processing (EMNLP-2014)*, pages 121–129.

Filippos Kokkinos and Alexandros Potamianos. 2017. Structural attention neural networks for improved sentiment analysis. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL-2017)*, pages 586–591.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Distributed representations of words and phrases and their compositionality. In *Proceedings of Advances in Neural Information Processing Systems (NIPS-2013)*.

Tomas Mikolov, Greg Corrado, Kai Chen, and Jeffrey Dean. 2013b. Efficient estimation of word representations in vector space. In *Proceedings of the International Conference on Learning Representations (ICLR-2013)*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2014)*, pages 1532–1543.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. 2017. Dynamic routing between capsules. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS-2017)*, pages 3859–3869.

Richard Socher, Alex Perelygin, and Jy Wu. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP-2013)*, pages 1631–1642.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL-2014)*, pages 1556–1566.

Duyu Tang, Furu Wei, Bing Qin, Nan Yang, Ting Liu, and Ming Zhou. 2016. Sentiment embeddings with applications to sentiment analysis. *IEEE Transactions on Knowledge and Data Engineering*, 28(2):496–509.

Jin Wang, Liang-Chih Yu, K.Robert Lai, and Xuejie Zhang. 2016a. Dimensional sentiment analysis using a regional CNN-LSTM model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL-2016)*, pages 225–230.

Jin Wang, Liang Chih Yu, K. Robert Lai, and Xuejie Zhang. 2016b. Community-based weighted graph model for valence-arousal prediction of affective words. *IEEE/ACM Transactions on Audio Speech and Language Processing*, 24(11):1957–1968.

Xin Wang, Yuanchao Liu, Chengjie Sun, Baoxun Wang, and Xiaolong Wang. 2015. Predicting polarities of tweets by composing word embeddings with long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL-2015)*, pages 1343–1353.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL/HLT-16)*, pages 1480–1489.

Liang-Chih Yu, Lung-Hao Lee, Shuai Hao, Jin Wang, Yunchao He, Jun Hu, K Robert Lai, and Xuejie Zhang. 2016. Building Chinese affective resources in valence-arousal dimensions. In *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL/HLT-2016)*, pages 540–545.

Liang-Chih Yu, Jin Wang, K. Robert Lai, and Xuejie Zhang. 2018a. Refining word embeddings for sentiment analysis. In *Proceedings ofthe 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP-2017)*, pages 534–539.

Liang-Chih Yu, Jin Wang, K.Robert Lai, and Xuejie Zhang. 2018b. Refining word embeddings using intensity scores for sentiment analysis. *IEEE/ACM Transactions on Audio Speech and Language Processing*, 26(3):671–681.

# A Label Informative Wide & Deep Classifier for Patents and Papers

**Muyao Niu**
Patsnap Ltd.
niumuyao@patsnap.com

**Jie Cai**
Patsnap Ltd.
caijie@patsnap.com

## Abstract

In this paper, we provide a simple and effective baseline for classifying both patents and papers to the well-established Cooperative Patent Classification (CPC). We propose a label-informative classifier based on the Wide & Deep structure, where the Wide part encodes string-level similarities between texts and labels, and the Deep part captures semantic-level similarities via non-linear transformations. Our model trains on millions of patents, and transfers to papers by developing distant-supervised training set and domain-specific features. Extensive experiments show that our model achieves comparable performance to the state-of-the-art model used in industry on both patents and papers. The output of this work should facilitate the searching, granting and filing of innovative ideas for patent examiners, attorneys and researchers.

## 1 Introduction

Classifying patents and papers to a technology taxonomy is a crucial step to organize the massive knowledge and to discover innovative ideas. Patent examiners rely on the taxonomy to search for similar documents when granting or invalidating a patent application; attorneys use it to check whether the innovation points of an invention have been covered in previous literature; researchers use the taxonomy to monitor the technology trends in certain fields, and companies use it to outline the intellectual property landscape of its own or its competitors'. The most commonly used taxonomies are International Patent Classification (IPC) [1] and its newer version Cooperative Patent Classification (CPC) [2]. Figure 1 illustrates the CPC hierarchy and the discriminative descriptions attached to each node.

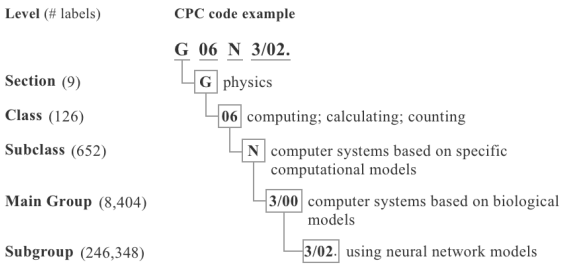


Figure 1: Illustration of the CPC hierarchy.

The categorization is now mostly done manually by experts in patent offices. Due to the growing amount of patents and the limited number of domain experts, there has been an urge to automate the classification process. Also, as more and more technology innovations are published in patents, researchers with no background in patent classification may want to know which patents are most relevant to an academic paper. For this end, we aim to classify both patents and papers to the CPC *subclass* with more than 600 labels.

Classifying patents and papers to CPC subclass is a challenging task because (1) there are a large number of labels that covers almost all technology domains, and the differences between labels are often subtle; (2) although mass amount of labelled data for patents is available, the annotated data for paper-to-CPC is very limited. Since labelling papers with CPC labels requires expert knowledge, large-scale human annotation is very expensive.

In this paper, we leverage the CPC label descriptions and use the Wide-and-Deep network to integrate label information with semantic information from input texts. We also construct a distant-supervised dataset for papers. **Our contributions** are:

- We prove the effectiveness of the label features through the Wide-and-Deep structure in CPC classification on more than 600 labels.
- We achieve comparable performance to the

[1] https://www.wipo.int/classifications/ipc/en/

[2] https://www.epo.org/searching-for-patents/helpful-resources/first-time-here/classification/cpc.html



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3438–3443,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

state-of-the-art on classifying both patents and papers to a widely-used technology taxonomy. Our model can serve as a simple and effective baseline for CPC classification tasks.

## 2 Related Work

### 2.1 Patent Classification

Most of the previous patent classification systems focus on developing more features derived from patent structures with traditional learning algorithms (Verberne and D'hondt, 2011). Going beyond document-level features, Cai and Hofmann (2007) and Qiu et al. (2011) capture the label hierarchy in model representations. It is flexible for traditional learning algorithms to integrate prior knowledge; however, it is difficult for them to generalize to unseen data.

Deep Neural Networks make efficient use of large-scale training data and generalize well to unseen data. There are a few NN-based patent classification methods. Grawe et al. (2017) uses Long Short Term Memory (LSTM) on 50 IPC subgroups and Li et al. (2018) applies Convolutional Neural Networks (CNN) to IPC subclasses classification.

There are some efforts on mapping papers to IPC in the context of patent retrieval combined with K nearest neighbours (KNN) classification. Cao et al. (2008) adopt a query-expansion approach to retrieve relevant patents and use a KNN classifier to label the research paper. Xiao et al. (2008) combine different scoring methods to rerank the retrieved IPC and achieved the best performance in the NTCIR-7 workshop for classify research papers to IPC system (Nanba et al., 2008). To our best knowledge, there has been no attempt tackling both tasks in one model.

### 2.2 Label Information

Leveraging label information is not new and is mostly accomplished by embedding labels and texts in the same space to measure their correlations (Yogatama et al., 2015; Zhang et al., 2018). Ma et al. (2016) adds prototypes to the label representation. Zhang et al. (2018) proposes to transform classification to a matching target between texts and labels for multi-task learning. Wang et al. (2018) further weights text features by the compatibilities between text and label embeddings via attention mechanism. Our model differs from previous studies in that we use the string-level similarity between label descriptions and input text instead of label embeddings. Based on the analysis of CPC classification system, we believe that string-level similarity can compensate for what semantic-level similarity cannot captures for patent classification tasks.

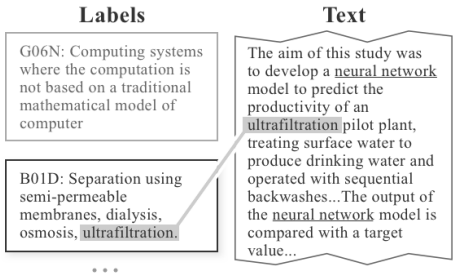


Figure 2: Example of the label description (left boxes) and classification cues (shadowed texts). The true label for the input text (right box) is B01D. Although the given text discussed neural networks, which is semantically closer to G06N, the classification cue of "ultrafiltration" decides for B01D eventually.

## 3 Model

### 3.1 The Label-text Feature

We discover that label descriptions can provide precise cues to classify a document which contains *multiple* semantic aspects. Figure 2 provide an example on the necessity of integrating label description. It should be noted that when patent examiners classify documents to the CPC system, they are also advised to use cue words and to search among label descriptions [3].

We integrate the label information through the label-text feature that captures the string-level relatedness between label descriptions and texts. Here we use BM25 score:.

$$\text{BM25}(\text{D}_k, x) = \sum_{i=1}^{n} \text{idf}(x_i) \frac{\text{tf}(x_i, D_k)(k_1+1)}{\text{tf}(x_i, D_k)+k_1(1-b+b\frac{|D_k|}{\text{avgdl}})} \quad (1)$$

where $x_i$ is the $i$th item in text x; $\boldsymbol{D}_k$ is the label description for class $k$; *idf* and *tf* are inverse document frequency and text frequency for $x_i$ in $D_k$ respectively; *avgdl* is the average description length; $|D_k|$ is the document length; $k_1$ and $b$ are hyper parameters.

[3] see IPC guidelines at https://www.wipo.int/export/sites/www/classifications/ipc/en/general/guidelines_where_to_classify.pdf

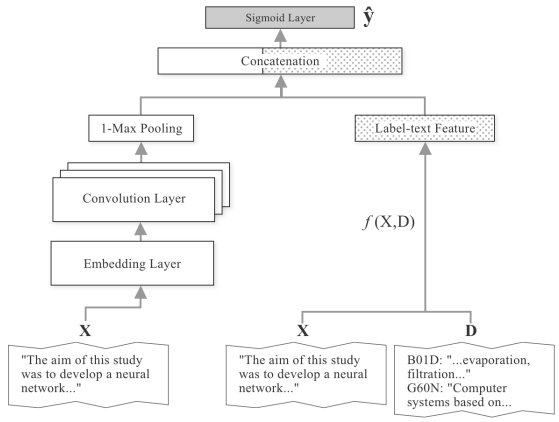


Figure 3: WnD classifier structure. The Deep part (left) captures semantic information; the Wide part (right) captures label-text relatedness. The label-text feature is a $K$-dimensional vector, where $K$ equals to the label set size.

### 3.2 Wide and Deep Structure

We adopt the Wide and Deep (WnD) neural network (Cheng et al., 2016) for text classification . Given a training set $(\boldsymbol{X}_n, \boldsymbol{D}, \boldsymbol{y}_n)_{n=1}^N$, where $\boldsymbol{X}_n$ is the input document texts, $\boldsymbol{D}$ is the input label descriptions and $\boldsymbol{y}_n$ is the true labels. The model outputs the probabilities for each of the $K$ classes $\hat{\boldsymbol{y}} \in R^K$. The training target is to minimize binary cross entropy loss: $L = \frac{1}{N}\frac{1}{K}\Sigma_{n=1}^{N}\Sigma_{k=1}^{K}\text{CE}(\boldsymbol{y}_n^{(k)}, \hat{\boldsymbol{y}}_n^{(k)})$, where $\boldsymbol{y}_n^{(k)}$ is the $k$th element in vector $\boldsymbol{y}_n$.

An overview of the WnD classifier is shown in figure 3. The model has two parts: the wide part and the deep part. The wide part takes in the label-text feature to capture string-level relatedness between the label descriptions and the text; the deep part maps the input text to word embeddings and go through a non-linear transformation to capture semantic-level relatedness between the text and the label.

**The Wide part**: The Wide part is a regression model with the form $\hat{\boldsymbol{y}}_{wide} = \sigma(\boldsymbol{W}_{wide}^T \boldsymbol{z}_{wide} + \boldsymbol{b})$, where $\boldsymbol{z}_{wide}$ is the label-text interaction features as described in section 3.1: $\boldsymbol{z}_{wide} = \text{BM25}(\boldsymbol{D}, \boldsymbol{X_n})$.

**The Deep part**: The Deep part is a non-linear transformation of the input text that aims to capture the semantic of the text. It can be a classic neural network for text encoding, such as RNN, CNN, or simple fully connected network. In this paper, we use textCNN (Kim, 2014) for the transformation because it is a simple baseline that works reasonably well. The Deep part transform the texts to a fixed-length representation $\boldsymbol{z}_{deep}$. The representation $\boldsymbol{z}_{deep}$ is then mapped to $K$ classes using sigmoid activation $\hat{\boldsymbol{y}}_{deep} = \sigma(\boldsymbol{W}_{deep}^T \boldsymbol{z}_{deep} + \boldsymbol{b})$.

The Wide and Deep parts are concatenated at the top and are jointly trained through $\hat{\boldsymbol{y}} = \sigma(\boldsymbol{W}_{deep}^T \boldsymbol{z}_{deep} + \boldsymbol{W}_{wide}^T \boldsymbol{z}_{wide} + \boldsymbol{b})$ in order to let the semantic and the string level relatedness complements each other when making the decision. In this way, the model simulates the behavior of patent examiners classifying a document: when they are uncertain which labels to assign (i.e. when semantic knowledge cannot provide a certain answer), examiners will resort to searching for cue words in the label descriptions for a clue.

## 4 Experimental Setup

### 4.1 Datasets

We remove stopwords and punctuations and choose the first 120 words per document. The word embeddings are 300 dimensional and initialized randomly. Kernel size for textCNN is 2,3,4 and 5, and the number of filters is 1024. For each CPC subclass, we use the descriptions of its own and of all its child labels. We train the model using Adam optimizer.

**Datasets**: Out of the USPTO patent set, We randomly sample 6.7 million abstracts as the patent training set and 60k as the testing set. For the paper testing set, the gold-standard is hard to obtain. We discover that some papers cited by patents are assigned CPC labels by European Patent Office, we collect those from the website [4] and derive 4956 testing instances for paper-to-CPC classification. The datasets are described in table 1.

| | # instances | # avg label per example |
|---|---|---|
| training set | 6.7 million | 1.47 |
| patents testing set | 60k | 1.45 |
| papers testing set | 4956 | 1.30 |

Table 1: Dataset Description

**Evaluation metrics**: As each patent/paper has one or more CPC labels, we measure our model from both the classification and the ranking perspectives with 3 metrics: (1) *example-based precision/recall*: the average precision/recall per instance. We measure precision and recall on the top1, top3 predictions and precision on all predictions with the probability score $\geq 0.5$. (2) *macro precision/recall*: the average precision/recall per class. (3) *mean average precision (MAP)*: a

[4] https://www.epo.org/index.html

ranking-based metric that measures whether the right labels are placed before the wrong ones.

### 4.2 Classify Patents to CPC

We compare our WnD classifier on patents with two baselines: traditional textCNN and attention-textCNN. By comparing WnD with textCNN, we want to know whether the label-text feature can complement semantic information for classification; by comparing with attention-textCNN, we want to compare our label integration method with other label embedding-based methods. For the attention model, we borrowed the idea of label-embedding attentive model (Wang et al., 2018). The attention is a $T$-dimensional vector where $T$ is the text length. It calculates the importance of each word to the classification task.

The WnD achieves significant gains with label information (see table 2). It suggests that the complementary effects of string-level relatedness between label and texts indeed benefits the final classification decision. Our model also outperforms attention-textCNN. Although label embeddings are helpful for small label sets (around 10 labels) (Wang et al., 2018; Zhang et al., 2018), it is less effective on hundreds of labels. We suspect the reason is that the attention is not discriminative between classes. When the label set is large, many non-stop words may be important for classification. But their weights should vary for different classes, which can hardly be captured by the attention vector. Also, attention-textCNN has much more parameters than textCNN and tends to overfit on the training data.

The best reported numbers on patent-to-IPC in subclass level achieves 0.74 on precision (Verberne and D'hondt, 2011). Although can not be directly compared, our large-enough testing set makes it confident that we are comparable with the state-of-the-art system while being more scalable.

### 4.3 Classify Papers to CPC

There is not enough labelled data for the paper-to-CPC task. We can directly apply the model trained on patents to papers, but the performance will degrade significantly due to domain difference. For example, the word *camera* in papers is commonly referred to as *photo capturing apparatus* in patents. To deal with the domain-adaptation issue, we propose two approaches:

**distant supervision**: We auto-label papers using patent-paper citation. For each paper, we label it with the CPCs of the patents that cite it. We assume that papers cited by a patent should be relevant to the given patent in terms of background and technology domain. We get a total 1.7 million papers with abstract that are cited by the patents in the USPTO patent set, and each paper gets on average 2.8 labels. On these auto-labelled papers we then fine tune the model originally trained on patents.

**domain-adapted features**: The WnD structure enables us to incorporate domain-adapted features to the Wide part. Here we proposed two ways to add such features:

- *prototyping*: We pick the top 20 terms from the papers for each of the K classes according to the tf-idf score. Those representative terms are used as the label descriptions for papers;
- *label expansion*: We train word embeddings using skip-gram (Mikolov et al., 2013) on papers and expand the original label descriptions with the 10 nearest words in the embedding space according to their cosine distance.

We compare our paper-to-CPC model with the best-performing KNN+reranking model (Xiao et al., 2008) introduced in section 2.1. We also want to compare our model with large-scale classification systems used in industry. In order to do that, we crawl from Google Patent the machine-classified CPC labels of scholar papers for our testing set [5], and we assume that the labels are ranked according to the order on the web page. Google classifies papers on the finest level. In order to compare our subclass results with it, we use only the subclass part of the first label, which is supposed to be the most confident one.

The comparison results are shown in table 3. WnD benefits from both transfer learning and domain-adapted features. Since the prototypes are automatically collected, it is possible to apply WnD to classification tasks where detailed label descriptions are not available.

Google Patent scores better on precision/recall@1, but performs less well on macro precision/recall. Since Google classifies the papers on a finer grained level, the classifier may receive more information during training, thus performing better on coarser grained levels. To

[5] The data can be found by searching for a scholar paper in Google Patent. The machine-classified CPCs will appear in the information page

| Model | p@1 | r@1 | p@3 | r@3 | precision (prob>0.5) | Macro p@1 | Macro r@1 | MAP |
|---|---|---|---|---|---|---|---|---|
| textCNN | 75.51 | 55.24 | 40.64 | 78.92 | 62.16 | 63.77 | 32.15 | 75.68 |
| attention-textCNN | 70.63 | 51.49 | 38.11 | 74.17 | 51.69 | 56.98 | 26.15 | 72.46 |
| WnD | **77.11** | **56.42** | **41.18** | **80.12** | **64.01** | **67.81** | **34.18** | **76.60** |

Table 2: Test results on patents. For attention-CNN we used the implementation of attention mechanism from (Wang et al., 2018)

| Model | p@1 | r@1 | p@3 | r@3 | precision (prob >0.5) | Macro p@1 | Macro r@1 | MAP |
|---|---|---|---|---|---|---|---|---|
| textCNN | 63.18 | 53.93 | 33.76 | 81.91 | 48.00 | 16.07 | 11.19 | 71.76 |
| WnD | 66.16 | 56.73 | 34.15 | 82.86 | 50.09 | 16.77 | 11.74 | 75.04 |
| KNN+reranking | 63.99 | 54.98 | 32.85 | 79.18 | NA | 14.70 | 10.60 | 72.37 |
| Google Patent* | **69.59** | **59.95** | NA | NA | NA | 14.28 | 10.25 | NA |
| WnD + transfer | 68.33 | 58.68 | 35.25 | 85.10 | **51.26** | 17.23 | 11.45 | 77.15 |
| WnD +transfer+Prototype | 68.94 | 59.12 | **35.60** | **85.87** | 51.11 | **17.63** | **12.39** | **77.84** |
| WnD +transfer+labelExpand | 68.69 | 58.96 | 35.31 | 85.22 | 50.37 | 17.16 | 11.71 | 77.53 |

Table 3: Test results on papers. textCNN and WnD are models trained on patents directly apply to papers. WnD+transfer refers to WnD fine tuned on auto-labelled papers.*The training set and granularity of Google Patent model may be different from other models. We put it here for the convenience to compare and discuss

investigate the effects of classification granularity on performance, we mapped the WnD subclass predictions to class level (128 label) and trained another WnD on class level. The p@1 and r@1 are $86.94\%$ and $79.15\%$ for WnD subclass-to-class and $82.43\%$ and $74.72\%$ for WnD class. The gap indicates the possible positive effects of fine-granularity training.

## 5 Conclusions and Future Works

In this paper, we propose a WnD classifier to map both patents and papers to CPC subclasses. The model captures both the string and semantic relatedness between labels and texts. We achieve comparative performance to the state-of-the-art models for both paper-to-CPC and patent-to-CPC tasks. We hope to contribute an intuitive, simple yet practically effective baseline for categorizing scientific publications.

Although CPC subclass already has over 600 labels, it is still a relative coarse granularity in the taxonomy. The finest level (subgroup) consists of over 200 thousand labels and provides much more detailed classification information. At the same time, with the explosion of labels, the task is much more challenging. In the future, we will go deeper into the taxonomy and try to explore the hierarchical relations between labels and improve the scalability of models for finer grained label sets.

## 6 Acknowledgements

We would like to thank Jian Yang for helping work through the codes and Yang Ding for his suggestions on the experiments. We would also like to thank the anonymous reviewers of EMNLP 2019; their comments and suggestions have helped us polish our arguments.

## References

Lijuan Cai and Thomas Hofmann. 2007. Exploiting known taxonomies in learning overlapping concepts. In *Proceedings of the 20th international joint conference on Artifical intelligence*, pages 714–719. Morgan Kaufmann Publishers Inc.

Guihong Cao, Jianyun Nie, and Lixin Shi. 2008. Ntcir-7 patent mining experiments at rali. pages 347 – 350. Proceedings of NTCIR-7 Workshop Meeting.

Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, pages 7–10. ACM.

Mattyws F. Grawe, Claudia A. Martins, and Andreia Gentil Bonfante. 2017. Automated patent classification using word embedding. In *16th IEEE International Conference on Machine Learning and Applications, ICMLA 2017, Cancun, Mexico, December 18-21, 2017*, pages 408–411. IEEE.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014*

*Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics.

Shaobo Li, Jie Hu, Yuxin Cui, and Jianjun Hu. 2018. Deeppatent: patent classification with convolutional neural networks and word embedding. *Scientometrics*, 117(2):721–744.

Yukun Ma, Erik Cambria, and Sa Gao. 2016. Label embedding for zero-shot fine-grained named entity typing. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 171–180.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Hidetsugu Nanba, Atsushi Fujii, Makoto Iwayama, and Taiichi Hashimoto. 2008. Overview of the patent mining task at the ntcir-7 workshop. pages 293 – 298. Proceedings of NTCIR-7 Workshop Meeting.

Xipeng Qiu, Xuanjing Huang, Zhao Liu, and Jinlong Zhou. 2011. Hierarchical text classification with latent concepts. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA - Short Papers*, pages 598–602. The Association for Computer Linguistics.

Suzan Verberne and Eva D'hondt. 2011. Patent classification experiments with the linguistic classification system LCS in CLEF-IP 2011. In *CLEF 2011 Labs and Workshop, Notebook Papers, 19-22 September 2011, Amsterdam, The Netherlands*, volume 1177 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Guoyin Wang, Chunyuan Li, Wenlin Wang, Yizhe Zhang, Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. 2018. Joint embedding of words and labels for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2321–2331.

Tong Xiao, Feifei Cao, Tianning Li, Guolong Song, Ke Zhou, Jingbo Zhu, and Huizhen Wang. 2008. KNN and re-ranking models for english patent mining at NTCIR-7. In *Proceedings of the 7th NTCIR Workshop Meeting on Evaluation of Information Access Technologies: Information Retrieval, Question Answering and Cross-Lingual Information Access, NTCIR-7, National Center of Sciences, Tokyo, Japan, December 16-19, 2008*. National Institute of Informatics (NII).

Dani Yogatama, Daniel Gillick, and Nevena Lazic. 2015. Embedding methods for fine grained entity type classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Association for Computational Linguistics.

Honglun Zhang, Liqiang Xiao, Wenqing Chen, Yongkun Wang, and Yaohui Jin. 2018. Multi-task label embedding for text classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4545–4553. Association for Computational Linguistics.

# Text Level Graph Neural Network for Text Classification

**Lianzhe Huang, Dehong Ma, Sujian Li, Xiaodong Zhang and Houfeng WANG**
MOE Key Lab of Computational Linguistics, Peking University, Beijing, 100871, China
{hlz, madehong, lisujian, zxdcs, wanghf}@pku.edu.cn

## Abstract

Recently, researches have explored the graph neural network (GNN) techniques on text classification, since GNN does well in handling complex structures and preserving global information. However, previous methods based on GNN are mainly faced with the practical problems of fixed corpus level graph structure which do not support online testing and high memory consumption. To tackle the problems, we propose a new GNN based model that builds graphs for each input text with global parameters sharing instead of a single graph for the whole corpus. This method removes the burden of dependence between an individual text and entire corpus which support online testing, but still preserve global information. Besides, we build graphs by much smaller windows in the text, which not only extract more local features but also significantly reduce the edge numbers as well as memory consumption. Experiments show that our model outperforms existing models on several text classification datasets even with consuming less memory.

## 1 Introduction

Text classification is a fundamental problem of natural language processing (NLP), which has lots of applications like SPAM detection, news filtering, and so on (Jindal and Liu, 2007; Aggarwal and Zhai, 2012). The essential step for text classification is text representation learning.

With the development of deep learning, neural networks like Convolutional Neural Networks (CNN) (Kim, 2014) and Recurrent Neural Networks (RNN) (Hochreiter and Schmidhuber, 1997) have been employed for text representation. Recently, a new kind of neural network named Graph Neural Network (GNN) has attracted wide attention (Battaglia et al., 2018). GNN was first proposed in (Scarselli et al., 2009) and has been used in many tasks in NLP including text classification (Defferrard et al., 2016), sequence labeling (Zhang et al., 2018a), neural machine translation (Bastings et al., 2017), and relational reasoning (Battaglia et al., 2016). Defferrard et al. (2016) first employed Graph Convolutional Neural Network (GCN) in text classification task and outperformed the traditional CNN models. Further, Yao et al. (2019) improved Defferrard et al. (2016)'s work by applying article nodes and weighted edges in the graph, and their model outperformed the state-of-the-art text classification methods.

However, these GNN-based models usually adopt the way of building one graph for the whole corpus, which causes the following problems in practice. First, high memory consumption is required due to numerous edges. Because this kind of methods build a single graph for the whole corpus and use edges with fixed weights, which considerably limits the expression ability of edges, they have to use a large connection window to get a global representation. Second, it is difficult for this kind of models to conduct the online test, because the structure and parameters of their graph are dependent on the corpus and cannot be modified after training.

To address the above problems, we propose a new GNN based method for text classification. Instead of building a single corpus level graph, we produce a text level graph for each input text. For a text level graph, we connect word nodes within a reasonably small window in the text rather than directly fully connect all the word nodes. The representations of the same nodes and weights of edges are shared globally and can be updated in the text level graphs through a massage passing mechanism, where a node takes in the information from neighboring nodes to update its representation. Finally, we summarize the representations of all the



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3444–3450,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

nodes in the graph to predict the results. With our design, text level graphs remove the burden of dependency between a single input text and the entire corpus, which support online test. Besides, it has the benefit of consuming less memory by connecting words in a small contextual window, because it excludes a good many words that are far away in the text and have little relation with the current word and thus significantly reduces the number of edges. The message passing mechanism makes nodes in the graph perceive information around them to get precise meaning in a specific context.

In our experiments, our method achieves state-of-the-art results in several text classification datasets and consumes significantly fewer memory resources compared with previous methods.

## 2 Method

In this section, we will introduce our method in detail. First, we show how to build a text level graph for a given text; all the parameters for the text level graph are taken from some global-sharing matrices. Then, we introduce the message passing mechanism on these graphs to obtain information from the context. Finally, we depict how to predict the label for a given text based on the learned representations. The overall architecture of our model is shown in Figure 1.

### 2.1 Building Text Graph

We notate a text with $l$ words as $T = \{\mathbf{r_1}, ...\mathbf{r_i}, ..., \mathbf{r_l}\}$, where $\mathbf{r_i}$ denotes the representation of the $i_{th}$ word. $\mathbf{r_i}$ is a vector initialized by $d$ dimension word embedding and can be updated by training. To build a graph for a given text, we regard all the words that appeared in the text as the nodes of the graph. Each edge starts from a word in the text and ends with its adjacent words. Concretely, the graph of text $T$ is defined as:

$$N = \{\mathbf{r_i} | i \in [1, l]\}, \quad (1)$$

$$E = \{e_{ij} | i \in [1, l]; j[i - p, i + p]\}, \quad (2)$$

where $N$ and $E$ are the node set and edge set of the graph, and word representations in $N$ and edge weights in $E$ are taken from global shared matrices. $p$ denotes the number of adjacent words connected to each word in the graph. Besides, we uniformly map the edges that occur less than $k$ times in the training set to a "public" edge to make parameters adequately trained.

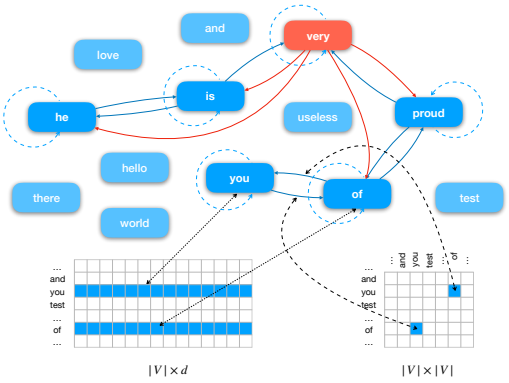


Figure 1: Structure of graph for a single text "he is very proud of you.". For the convenience of display, in this figure, we set $p = 2$ for the node "very" (nodes and edges are colored in red) and $p = 1$ for the other nodes(colored in blued). In actual situations, the value of $p$ during a session is unique. All the parameters in the graph come from the global shared representation matrix, which is shown at the bottom of the figure.

Compared with the previous methods in building graph, our approach can exceedingly reduce the scale of the graph in terms of nodes and edges. That means that the text-level graph can consume much less GPU memory. Besides, their method is unfriendly to new-coming text, while our approach can solve this problem because the graph for each text is only dependent on its content.

### 2.2 Message Passing Mechanism

Convolution can extract information from local features (LeCun et al., 1989). In the graph domain, convolution is implemented by spectral approaches (Bruna et al., 2014; Henaff et al., 2015), or non-spectral approaches (Duvenaud et al., 2015). In this paper, a non-spectral method named message passing mechanism (MPM) (Gilmer et al., 2017) is employed for convolution. MPM first collects information from adjacent nodes and updates its representations based on its original representations and collected information, which is defined as:

$$\mathbf{M_n} = \max_{a \in \mathcal{N}_n^p} e_{an} \mathbf{r_a}, \quad (3)$$

$$\mathbf{r'_n} = (1 - \eta_n)\mathbf{M_n} + \eta_n \mathbf{r_n} \quad (4)$$

where $\mathbf{M_n} \in \mathbb{R}^d$ is the messages that node $n$ receives from its neighbors; max is a reduction function which combines the maximum values on each dimension to form a new vector as an output. $\mathcal{N}_n^p$ denotes nodes that represent the nearest $p$ words of $n$ in the original text; $e_{an} \in \mathbb{R}^1$ is the edge weight

from node $a$ to node $n$, and it can be updated during training; and $\mathbf{r_n} \in \mathbb{R}^d$ denotes the former representation of node $n$. $\eta_n \in \mathbb{R}^1$ is a trainable variable for node $n$ that indicates how much information of $\mathbf{r_n}$ should be kept. $\mathbf{r'_n}$ denotes the updated representation of node $n$.

MPM makes the representations of nodes influenced by neighborhoods, which means the representations can bring the information from context. Therefore, even for polysemous words, the precise meaning in the context can be determined by the influence of weighted information from neighbors. Besides, the parameters of text level graphs are taken from global shared matrices, which means the representations can also bring global information as other graph-based models do.

Finally, the representations of all nodes in the text are used to predict the label of the text:

$$y_i = \text{softmax}(\text{Relu}(\mathbf{W} \sum_{n \in N_i} \mathbf{r'_n} + \mathbf{b})) \quad (5)$$

where $W \in \mathbb{R}^{d \times c}$ is a matrix mapping the vector into an output space, $N_i$ is the node set of text $i$ and $\mathbf{b} \in \mathbb{R}^c$ is bias.

The goal of training is to minimize the cross-entropy loss between ground truth label and predicted label:

$$\mathbf{loss} = -g_i \log y_i, \quad (6)$$

where $g_i$ is the "one-hot vector" of ground truth label.

## 3 Experiments

In this section, we describe our experimental setup and report our experimental results.

### 3.1 Experimental Setup

For experiments, we utilize datasets including R8, R52[1], and Ohsumed[2]. R8 and R52 are both the subsets of Reuters 21578 datasets. Ohsumed corpus is extracted from MEDLINE database. MEDLINE is designed for multi-label classification, we remove the text with two or more labels. For all the datasets above, we randomly select 10% text from the training set to build validation set. The overview of datasets is listed in Table 1.

We compare our method with the following baseline models. It is noted that the results of some models are directly taken from (Yao et al., 2019).

[1]https://www.cs.umb.edu/~smimarog/textmining/datasets/

[2]http://disi.unitn.it/moschitti/corpora.htm

| Datasets | # Train | # Test | Categories | Avg. Length |
|---|---|---|---|---|
| R8 | 5485 | 2189 | 8 | 65.72 |
| R52 | 6532 | 2568 | 52 | 69.82 |
| Ohsumed | 3357 | 4043 | 23 | 135.82 |

Table 1: Datasets overview.

- **CNN** Proposed by (Kim, 2014), perform convolution and max pooling operation on word embeddings to get representation of text.
- **LSTM** Defined in (Liu et al., 2016), use the last hidden state as the representation of the text. Bi-LSTM is a bi-directional LSTM.
- **fastText** Proposed by (Joulin et al., 2017), average word or n-gram embeddings as documents embeddings.
- **Graph-CNN** Operate convolution over word embedding similarity graphs by fourier filter, proposed by (Defferrard et al., 2016).
- **Text-GCN** A graph based text classification model proposed by (Yao et al., 2019), which builds a single large graph for whole corpus.

### 3.2 Implementation Details

We set the dimension of node representation as 300 and initialize with random vectors or Glove (Pennington et al., 2014). $k$ discussed in Section 2.1 is set to 2. We use the Adam optimizer (Kingma and Ba, 2014) with an initial learning rate of $10^{-3}$, and L2 weight decay is set to $10^{-4}$. Dropout with a keep probability of 0.5 is applied after the dense layer. The batch size of our model is 32. We stop training if the validation loss does not decrease for 10 consecutive epochs.

For baseline models, we use default parameter settings as in their original papers or implementations. For models using pre-trained word embeddings, we used 300-dimensional GloVe word embeddings.

### 3.3 Experimental Results

Table 2 reports the results of our models against other baseline methods. We can see that our model can achieve the state-of-the-art result.

We note that the results of graph-based models are better than traditional models like CNN, LSTM, and fastTest. That is likely due to the characteristics of the graph structure. Graph structure

| Model | R8 | R52 | Ohsumed |
|---|---|---|---|
| CNN | 94.0 ± 0.5 | 85.3 ± 0.5 | 43.9 ± 1.0 |
| LSTM | 93.7 ± 0.8 | 85.6 ± 1.0 | 41.1 ± 1.0 |
| Graph-CNN | 97.0 ± 0.2 | 92.8 ± 0.2 | 63.9 ± 0.5 |
| Text-GCN | 97.1 ± 0.1 | 93.6 ± 0.2 | 68.4 ± 0.6 |
| CNN* | 95.7 ± 0.5 | 87.6 ± 0.5 | 58.4 ± 1.0 |
| LSTM* | 96.1 ± 0.2 | 90.5 ± 0.8 | 51.1 ± 1.5 |
| Bi-LSTM* | 96.3 ± 0.3 | 90.5 ± 0.9 | 49.3 ± 1.0 |
| fastText* | 96.1 ± 0.2 | 92.8 ± 0.1 | 57.7 ± 0.5 |
| Text-GCN* | 97.0 ± 0.1 | 93.7 ± 0.1 | 67.7 ± 0.3 |
| Our Model* | **97.8** ± **0.2** | **94.6** ± **0.3** | **69.4** ± **0.6** |

Table 2: Accuracy on several text classification datasets. Model with "*" means that all word vectors are initialized by Glove word embeddings. We run all models 10 times and report mean results.

allows a different number of neighbor nodes to exist, which enables word nodes to learn more accurate representations through different collocations. Besides, the relationship between words can be recorded in the edge weights and shared globally. These are all impossible for traditional models.

We also find that our model performs better than graph-based models like Graph-CNN. Graph-CNN represents documents using the bag-of-word model, which is similar to ours, but they connect word nodes within a large window without weighted edges, which cannot distinguish the importance between different words. While our model employed trainable edge weights, which let words express themselves differently when faced with various collocation. Besides, the weights are shared globally which means they can be trained by all the text contains the same collocation in the entire corpus.

We also note that our model performs better than former state-of-the-art model Text-GCN. That is likely due to more expressive edges, which have been discussed before, and the difference of representations learning. Text-GCN learns word representations by corpus level co-occurrence while our model is trained within a contextual window like traditional word embeddings. Therefore our model can benefit from pre-trained word embeddings and achieve better results.

### 3.4 Analysis of Memory Consumption

Table 3 reports the comparison of memory consumption and edges numbers between Text-GCN and our model. Results show that our model has a significant advantage in memory consumption.

| Datasets | Text-GCN | Our Model |
|---|---|---|
| R8 | 9,979M(2,841,760) | 954M(250,623) |
| R52 | 8,699M(3,574,162) | 951M(316,669) |
| Ohsumed | 13,510M(6,867,490) | 1,167M(419,583) |

Table 3: Comparison of memory consuming. The number of edges in the whole model is in parentheses.

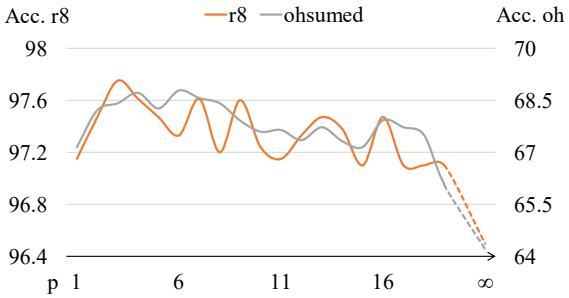


Figure 2: Model performance using $p$ from 1 to 19 and "$\infty$" (fully-connected). All hyperparameters are set the same except $p$. The left and right ordinate indicate the accuracy on the r8 and ohsumed dataset respectively.

As discussed in 2.1, the words in our model are only connected to adjacent words in the texts, while Text-GCN , which is based on the corpus level graph, connects nodes within a reasonably large window. Because Text-GCN uses co-occurrence information as fixed weights, it has to enlarge the window size to get a more accurate co-occurrence weight. Therefore, we will get a much more sparse edge weights matrix than Text-GCN. Also, since the representation of a text is calculated by the sum of the representations of word nodes in the text, there is no text node in our model, which also reduces memory consumption.

### 3.5 Analysis of Edges

To understand the difference of various connecting windows, we compared the performance of the R8 and ohsumed datasets with different $p$ values, the result is reported in Figure 2. We find that the accuracy increases as $p$ becomes larger and achieves the best performance when connected with about 3 neighborhoods. Then the accuracy decreases volatility as $p$ increases. This suggests that when connected only with the nearest neighborhood, nodes cannot understand the dependencies that span multiple words in the context, while connected with neighborhoods far away (much larger $p$), the graphs become more and more similar with fully connected graphs which ignore the local features. In addition, the fewer edges, the

| Setting | R8 | R52 | Ohsumed |
|---|---|---|---|
| Original | **97.8 ± 0.2** | **94.6 ± 0.3** | **69.4 ± 0.6** |
| (1)Fixed PMI Edges W. | 97.7 ± 0.2 | 94.0 ± 0.2 | 67.6 ± 0.5 |
| (2)Mean Reduction | 97.7 ± 0.1 | 94.5 ± 0.3 | 62.6 ± 0.2 |
| (3)Random Word Emb. | 97.4 ± 0.2 | 93.7 ± 0.2 | 67.3 ± 0.5 |

Table 4: Results of ablation studies. We run all models for 5 times and give mean results.

fewer memory consumption. Our model has fewer edges compared with previous methods, and this also show the advantages of our proposed model.

### 3.6 Ablation Study

To further analyze our model, we perform ablation studies and Table 4 shows the results.

In (1), we fix the weights of edges and initialize them with point-wise mutual information (PMI), and the size of sliding windows is set to 20, which is the same as (Yao et al., 2019). Removing the trainable edges makes the model perform worse on all data sets, which demonstrates the effectiveness of trainable edges. In our opinion, the main reason is that trainable edges can better model the relations between words compared with fixed edges.

In (2), we change the max-reduction by mean-reduction. In the original model, the node gets its new representation from received messages by obtaining the maximum value alone each dimension. From Table 4, we can see that the max reduction can achieve better results. The node reduction function is similar to the pooling operation on CNN. Reduction by max highlights features that are highly discriminating and provides non-linearity, which helps to achieve better results.

In (3), we remove the pre-trained word embeddings from nodes and initialize all the nodes with random vectors. Compared with the original model, the performances are slightly decreased without pre-trained word embeddings. Therefore, we believe that the pre-trained word embeddings have a particular effect on improving the performance of our model.

## 4 Related Work

In this section, we will introduce the related works about GNN and text classification in detail.

### 4.1 Graph Neural Networks

Graph Neural Networks (GNN) has got extensive attention recently (Zhou et al., 2018; Zhang et al., 2018b; Wu et al., 2019). GNN can model non-Euclidean data, while traditional neural networks can only model regular grid data. While many tasks in reality such as knowledge graphs (Hamaguchi et al., 2017), social networks (Hamilton et al., 2017) and many other research areas (Khalil et al., 2017) are with data in the form of trees or graphs. So GNN are proposed (Scarselli et al., 2009) to apply deep learning techniques to data in graph domain.

### 4.2 Text Classification

Text classification is a classic problem of natural language processing and has a wide range of applications in reality. Traditional text classification like bag-of-words (Zhang et al., 2010), n-gram (Wang and Manning, 2012) and Topic Model (Wallach, 2006) mainly focus on feature engineering and algorithms. With the development of deep learning techniques, more and more deep learning models are applied for text classification. Kim (2014); Liu et al. (2016) applied CNN and RNN into text classification and achieved results which are much better than traditional models.

With the development of GNN, some graph-based classification models are gradually emerging (Hamilton et al., 2017; Veličković et al., 2017; Peng et al., 2018). Yao et al. (2019) proposed Text-GCN and achieved state-of-the-art results on several mainstream datasets. However, Text-GCN has the disadvantages of high memory consumption and lack of support online training. The model presents in this paper solves the mentioned problems in Text-GCN and achieves better results.

## 5 Conclusion

In this paper, we proposed a new graph based text classification model, which uses text level graphs instead of a single graph for the whole corpus. Experimental results show that our model achieves state-of-the-art performance and has a significant advantage in memory consumption.

## Acknowledgments

Our work is supported by the National Key Research and Development Program of China under Grant No.2017YFB1002101 and National Natural Science Foundation of China under Grant No.61433015 and No.61572049. The corresponding author of this paper is Houfeng Wang.

## References


Charu C Aggarwal and ChengXiang Zhai. 2012. A survey of text classification algorithms. In *Mining text data*, pages 163–222. Springer.

Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Simaan. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1957–1967.

Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. 2016. Interaction networks for learning about objects, relations and physics. In *Advances in neural information processing systems*, pages 4502–4510.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann Lecun. 2014. Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representations (ICLR2014), CBLS, April 2014*.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852.

David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. 2015. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pages 2224–2232.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1263–1272.

Takuo Hamaguchi, Hidekazu Oiwa, Masashi Shimbo, and Yuji Matsumoto. 2017. Knowledge transfer for out-of-knowledge-base entities: A graph neural network approach. *arXiv preprint arXiv:1706.05674*.

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034.

Mikael Henaff, Joan Bruna, and Yann LeCun. 2015. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Nitin Jindal and Bing Liu. 2007. Review spam detection. In *Proceedings of the 16th international conference on World Wide Web*, pages 1189–1190.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431.

Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. 2017. Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems*, pages 6348–6358.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. 1989. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 2873–2879.

Hao Peng, Jianxin Li, Yu He, Yaopeng Liu, Mengjiao Bao, Lihong Wang, Yangqiu Song, and Qiang Yang. 2018. Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In *Proceedings of the 2018 World Wide Web Conference*, pages 1063–1072. International World Wide Web Conferences Steering Committee.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Hanna M Wallach. 2006. Topic modeling: beyond bag-of-words. In *Proceedings of the 23rd international conference on Machine learning*, pages 977–984. ACM.

Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th annual meeting of the association for computational linguistics: Short papers-volume 2*, pages 90–94. Association for Computational Linguistics.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2019. A comprehensive survey on graph neural networks.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7370–7377.

Yin Zhang, Rong Jin, and Zhi-Hua Zhou. 2010. Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics*, 1(1-4):43–52.

Yue Zhang, Qi Liu, and Linfeng Song. 2018a. Sentence-state lstm for text representation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 317–327.

Ziwei Zhang, Peng Cui, and Wenwu Zhu. 2018b. Deep learning on graphs: A survey.

Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. 2018. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*.

# Semantic Relatedness Based Re-ranker for Text Spotting

**Ahmed Sabir**[1] **Francesc Moreno-Noguer**[2] **Lluís Padró**[1]
[1] Universitat Politècnica de Catalunya, TALP Research Center, Barcelona, Spain
[2] Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Barcelona, Spain
asabir@cs.upc.edu, fmoreno@iri.upc.edu, padro@cs.upc.edu

## Abstract

Applications such as textual entailment, plagiarism detection or document clustering rely on the notion of *semantic similarity*, and are usually approached with dimension reduction techniques like LDA or with embedding-based neural approaches. We present a scenario where semantic similarity is not enough, and we devise a neural approach to learn *semantic relatedness*. The scenario is *text spotting in the wild*, where a text in an image (*e.g.* street sign, advertisement or bus destination) must be identified and recognized. Our goal is to improve the performance of vision systems by leveraging semantic information. Our rationale is that the text to be spotted is often related to the image context in which it appears (word pairs such as *Delta–airplane*, or *quarters–parking* are not *similar*, but are clearly *related*). We show how learning a word-to-word or word-to-sentence relatedness score can improve the performance of text spotting systems up to 2.9 points, outperforming other measures in a benchmark dataset.

## 1 Introduction

Deep learning has been successful in tasks related to deciding whether two short pieces of text refer to the same topic, *e.g.* semantic textual similarity (Cer et al., 2018), textual entailment (Parikh et al., 2016) or answer ranking for Q&A (Severyn and Moschitti, 2015).

However, other tasks require a broader perspective to decide whether two text fragments are *related* more than whether they are *similar*. In this work, we describe one of such tasks, and we retrain some of the existing sentence similarity approaches to learn this semantic relatedness. We also present a new Deep Neural Network (DNN) that outperforms existing approaches when applied to this particular scenario.

The task we tackle is *Text Spotting*, which is the problem of recognizing text that appears in unrestricted images (a.k.a. *text in the wild*) such as traffic signs, commercial ads, or shop names. Current state-of-the-art results on this task are far from those of OCR systems with simple backgrounds.

Existing approaches to Text Spotting usually divide the problem in two fundamental tasks: 1) *text detection*, consisting of selecting the image regions likely to contain texts, and 2) *text recognition*, that converts the images within these bounding boxes into a readable string. In this work, we focus on the recognition stage, aiming to prove that semantic relatedness between the image context and the recognized text can be useful to boost the system performance. We use existing pre-trained architectures for Text Recognition, and add a shallow deep-network that performs a post-processing operation to re-rank the proposed candidate texts. In particular, we re-rank the candidates using their semantic relatedness score with other visual information extracted from the image (*e.g.* objects, scenario, image caption). Extensive evaluation shows that our approach consistently improves other semantic similarity methods.

## 2 Text Hypothesis Extraction

We use two pre-trained Text Spotting baselines to extract $k$ text hypotheses. The first baseline is a CNN (Jaderberg et al., 2016) with fixed lexicon based recognition, able to recognize words in a predefined 90K-word dictionary. Second, we use an LSTM architecture with visual attention model (Ghosh et al., 2017) that generates the final output words as probable character sequences, without relying on any lexicon. Both models are trained on a synthetic dataset (Jaderberg et al., 2014). The output of both models is a Softmax score for each of the $k$ candidate words.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3451–3457, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

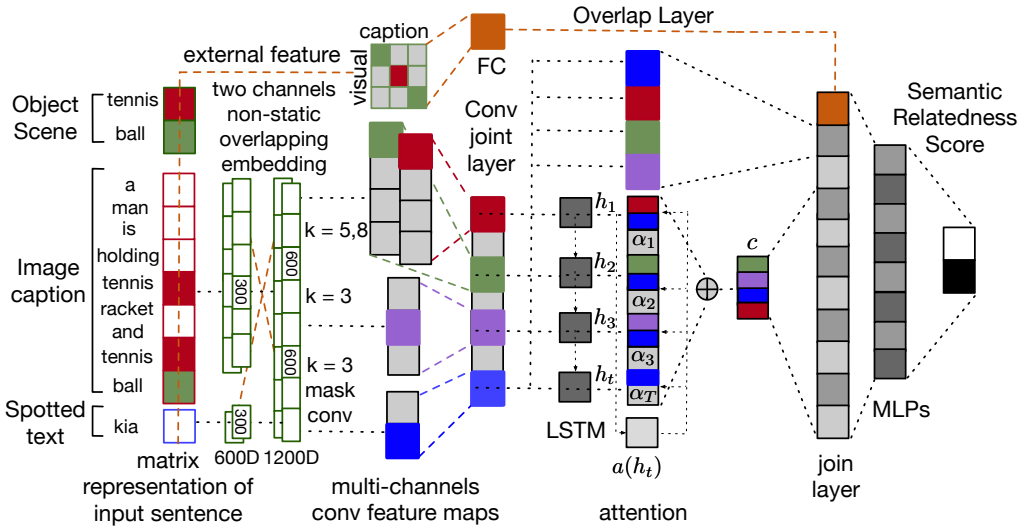


Figure 1: Overview of the system pipeline, an end-to-end post-process scores the semantic relatedness between a candidate word and the context in the image (objects, scenarios, natural language descriptions, ...)

## 3 Learning Semantic Relatedness for Text Spotting

To learn the semantic relatedness between the visual context information and the candidate word we introduce a multi-channel convolutional LSTM with an attention mechanism. The network is fed with the candidate word plus several words describing the image visual context (object and places labels, and descriptive captions)[1], and is trained to produce a relatedness score between the candidate word and the context.

Our architecture is inspired by (Severyn and Moschitti, 2015), that proposed CNN-based rerankers for Q&A. Our network consists of two subnetworks, each with 4-channels with kernel sizes $k = (3, 3, 5, 8)$, and overlap layer, as shown in Figure 1. We next describe the main components:

**Multi-Channel Convolution**: The first subnetwork consists of only convolution kernels, and aims to extract n-gram or keyword features from the caption sequence.

The convolution is applied over a sequence to extract n-gram features from different positions. Let $x \in \mathbb{R}^{s \times d}$ be the sentence matrix, where $s$ is the sentence length, and $d$ the dimension of the i-th word in the sentence. Let also denote by $c \in \mathbb{R}^{k \times d}$ the kernel for the convolution operation. For each $i$-th position in the sentence, $w_i$ is the concatenation of $k$ consecutive words, i.e., $w_i = [x_i \oplus x_{i+1} \oplus \ldots \oplus x_{i+k-1}]$. Our architecture uses multiple such kernels to generate feature maps $m$. The feature map for each window vector $w_i$ can be written as:

$$m_i = f(w_i \circ c + b) \quad (1)$$

where $\circ$ is element-wise multiplication, $f$ is nonlinear function, in our case we apply Relu function (Nair and Hinton, 2010), and $b$ is a bias. For $j$ kernels, the generated $j$ feature maps can be arranged as feature representation for each window $W_i$ as: $W = [m_1 \oplus m_2 \oplus \ldots \oplus m_j]$. Each row $W_i$ of $W \in \mathbb{R}^{(s-k+1) \times j}$ is the new generated feature from the $j$-th kernel for the window vector at position $i$. The new generated feature (window representations) are then fed into the joint layer and LSTM as shown in Figure 1.

**Multi-Channel Convolution-LSTM**: Following C-LSTM (Zhou et al., 2015) we forward the output of the CNN layers into an LSTM, which captures the long term dependencies over the features. We further introduce an attention mechanism to capture the most important features from that sequence. The advantage of this attention is that the model learns the sequence without relying on the temporal order. We describe in more detail the attention mechanism below.

Also, following (Zhou et al., 2015), we do not use a pooling operation after the convolution feature map. Pooling layer is usually applied after the convolution layer to extract the most important features in the sequence. However, the output of our Convolutional-LSTM model is fed into

[1] All this visual context information is automatically generated using off-the-shelf existing modules (see section 4).

an LSTM (Hochreiter and Schmidhuber, 1997) to learn the extracted sequence, and pooling layer would break that sequence via downsampling to a selected feature. In short, LSTM is specialized in learning sequence data, and pooling operation would break such a sequence order. On the other hand, for the Multi-Channel Convolution model we also lean the extracted word sequence n-gram directly and without feature selection, pooling operation.

**Attention Mechanism**: Attention-based models have shown promising results on various NLP tasks (Bahdanau et al., 2014). Such mechanism learns to focus on a specific part of the input (*e.g.* a relevant word in a sentence). We apply an attention mechanism (Raffel and Ellis, 2015) via an LSTM that captures the temporal dependencies in the sequence. This attention uses a Feed Forward Neural Network (FFN) attention function:

$$e_t = \tanh\left(h_t W_a\right) v_a^{\mathrm{T}} \tag{2}$$

where $W_a$ is the attention of the hidden weight matrix and $v_a$ is the output vector. As shown in Fig. 1 the vector $c$ is computed as a weighted average of $h_t$, given by $\alpha$ (defined below). The attention mechanism is used to produce a single vector $c$ for the complete sequence as follows:

$$e_t = a\left(h_t\right), \alpha_t = \frac{\exp\left(e_t\right)}{\sum_{k=1}^{T} \exp\left(e_k\right)}, c = \sum_{t=1}^{T} \alpha_t h_t$$

where $T$ is the total number of steps and $\alpha_t$ is the computed weight of each time step $t$ for each state $h_t$, $a$ is a learnable function that depends only on $h_t$. Since this attention computes the average over time, it discards the temporal order, which is ideal for learning semantic relations between words. By doing this, the attention gives higher weights to more important words in the sentence without relying on sequence order.

**Overlap Layer:** The overlap layer is just a frequency count dictionary to compute overlap information of the inputs. The idea is to give more weight to the most frequent visual element, specially when it is observed by more than one visual classifier. The dictionary output is a fully connected layer.

Finally, we merge all subnetworks into a joint layer that is fed to a loss function which calculates the semantic relatedness between both inputs. We call the combined model Fusion Dual Convolution-LSTM-Attention (FDCLSTM$_{AT}$).

Since we have only one candidate word at a time, we apply a convolution with *masking* in the candidate word side (first channel). In this case, simply zero-padding the sequence has a negative impact on the learning stability of the network. We concatenate the CNN outputs with the additional feature into MLP layers, and finally a sigmoid layer performing binary classification. We trained the model with a binary cross-entropy loss ($l$) where the target value (in $[0, 1]$) is the semantic relatedness between the word and the visual. Instead of restricting ourselves to a simple similarity function, we let the network learn the margin between the two classes –*i.e.* the degree of similarity. For this, we increase the depth of network after the MLPs merge layer with more fully connected layers. The network is trained using Nesterov-accelerated Adam (Nadam) (Dozat, 2016) as it yields better results (specially in cases such as word vectors/neural language modelling) than other optimizers using only classical momentum (ADAM). We apply batch normalization (BN) (Ioffe and Szegedy, 2015) after each convolution, and between each MLPs layer. We omitted the BN after the convolution for the model without attention (FDCLSTM), as BN deteriorated the performance. Additionally, we consider 70% dropout (Srivastava et al., 2014) between each MLPs for regularization purposes.

## 4 Dataset and Visual Context Extraction

We evaluate the performance of the proposed approach on the COCO-text (Veit et al., 2016). This dataset is based on Microsoft COCO (Lin et al., 2014) (Common Objects in Context), which consists of 63,686 images, and 173,589 text instances (annotations of the images). This dataset does not include any visual context information, thus we used out-of-the-box object (He et al., 2016) and place (Zhou et al., 2014) classifiers and tuned a caption generator (Vinyals et al., 2015) on the same dataset to extract contextual information from each image, as seen in Figure 2.

## 5 Related Work and Contribution

Understanding the visual environment around the text is very important for scene understanding. This has been recently explored by a relatively reduced number of works. Zhu et al. (2016) shows that the visual context could be beneficial for text detection. This work uses a 14 classes pixel clas-

Table 1: Best results after re-ranking using different re-ranker, and different values for $k$-best hypotheses extracted from the baseline output (%). In addition, to evaluate our re-ranker with MRR we fixed $k$ $\text{CNN}_{k=8}$ $\text{LSTM}_{k=4}$

| **Model** | **CNN** | | | | | **LSTM** | | | |
|---|---|---|---|---|---|---|---|---|---|
| | *full* | *dict* | *list* | *k* | MRR | *full* | *list* | k | MRR |
| Baseline (BL) | **full: 19.7 dict: 56.0** | | | | | **full: 17.9** | | | |
| BL+ Glove (Pennington et al., 2014) | 22.0 | 62.5 | 75.8 | 7 | 44.5 | 19.1 | 75.3 | 4 | 78.8 |
| BL+C-LSTM (Zhou et al., 2015) | 21.4 | 61.0 | 71.3 | 8 | 45.6 | 18.9 | 74.7 | 4 | 80.7 |
| BL+CNN-RNN (Wang et al., 2016) | 21.7 | 61.8 | 73.3 | 8 | 44.5 | 19.5 | 77.1 | 4 | 80.9 |
| BL+MVCNN (Yin and Schütze, 2016) | 21.3 | 60.6 | 71.9 | 8 | 44.2 | 19.2 | 75.8 | 4 | 78.8 |
| BL+Attentive LSTM (Tan et al., 2016) | 21.9 | 62.4 | 74.0 | 8 | 45.7 | 19.1 | 71.4 | 5 | 80.2 |
| BL+fasttext (Joulin et al., 2017) | 21.9 | 62.2 | 75.4 | 7 | 44.6 | 19.4 | 76.1 | 4 | 80.3 |
| BL+InferSent (Conneau et al., 2017) | 22.0 | 62.5 | 75.8 | 7 | 44.5 | 19.4 | 76.7 | 4 | 79.7 |
| BL+USE-T (Cer et al., 2018) | 22.0 | 62.5 | **78.3** | 6 | 44.7 | 19.2 | 75.8 | 4 | 79.5 |
| BL+TWE (Sabir et al., 2018) | 22.2 | 63.0 | 76.3 | 7 | 44.7 | 19.5 | 76.7 | 4 | 80.2 |
| BL+FDCLSTM (ours) | 22.3 | 63.3 | 75.1 | 8 | 45.0 | **20.2** | 67.9 | 9 | 79.8 |
| BL+FDCLSTM$_{AT}$ (ours) | **22.4** | 63.7 | 75.5 | 8 | **45.9** | 20.1 | 67.6 | 9 | **81.8** |
| BL+FDCLSTM$_{lexicon}$ (ours) | 22.6 | 64.3 | 76.3 | 8 | 45.1 | 19.4 | 76.4 | 4 | 78.8 |
| BL+FDCLSTM$_{AT+lexicon}$ (ours) | 22.6 | 64.3 | 76.3 | 8 | 45.1 | 19.7 | **77.8** | 4 | 80.4 |

sifier to extract *context features* from the image, such as *tree, river, wall*, to then assist scene text detection. Kang et al. (2017) employs topic modeling to learn the correlation between visual context and the spotted text in social media. The metadata associated with each image (e.g tags, comments and titles) is then used as context to enhance recognition accuracy. Karaoglu et al. (2017) takes advantage of text and visual context for logo retrieval problem. Most recently, Prasad and Wai Kin Kong (2018) use object information (limited to 42 predefined object classes) surrounding the spotted text to guide text detection. They propose two sub-networks to learn the relation between text and object class (e.g. relations such as *car–plate* or *sign board–digit*).

Unlike these methods, our approach uses direct visual context from the image where the text appears, and does not rely on any extra resource such as human labeled meta-data (Kang et al., 2017) nor limits the context object classes (Prasad and Wai Kin Kong, 2018). In addition, our approach is easy to train and can be used as a drop-in complement for any text-spotting algorithm that outputs a ranking of word hypotheses.

## 6 Experiments and Results

In the following we use different *similarity* or *relatedness* scorers to reorder the $k$-best hypothesis produced by an off-the-shelf state-of-the-art text spotting system. We experimented extracting $k$-best hypotheses for $k = 1 \ldots 10$.

We use two pre-trained deep models: a CNN (Jaderberg et al., 2016) and an LSTM (Ghosh et al., 2017) as baselines (BL) to extract the initial list of word hypotheses.

The CNN baseline uses a closed lexicon; therefore, it cannot recognize any word outside its 90K-word dictionary. Table 1 presents four different accuracy metrics for this case: 1) **full** columns correspond to the accuracy on the whole dataset. 2) **dict** columns correspond to the accuracy over the cases where the target word is among the 90K-words of the CNN dictionary (which correspond to 43.3% of the whole dataset. 3) **list** columns report the accuracy over the cases where the right word was among the $k$-best produced by the baseline. 4) **MRR** Mean Reciprocal Rank (MRR), which is computed as follows: $\text{MRR} = \frac{1}{|Q|}\sum_{k=1}^{|Q|}\frac{1}{\text{rank}_k}$, where rank $k$ is the position of the correct answer in the hypotheses list proposed by the baseline.

**Comparing with sentence level model:** We compare the results of our encoder with several state-of-the-art sentence encoders, tuned or trained on the same dataset. We use cosine to compute the similarity between the caption and the candidate word. Word-to-sentence representations are computed with: Universal Sentence Encoder with the Transformer USE-T (Cer et al., 2018), and Infersent (Conneau et al., 2017) with glove (Pennington et al., 2014). The rest of the systems in Table 1 are trained in the same conditions that our model with glove initialization with dual-channel overlapping non-static pre-trained embedding on the same dataset. Our model FDCLSTM without attention achieves a better result in the case of the second baseline LSTM that full of false-positives and short words. The advantage of the attention mechanism is the ability to integrate information over time, and it allows the model to refer to spe-




Figure 2: Examples of candidate re-ranking using object (c1), place (c2), and caption (c3) information. The three left examples are re-ranked based on the semantic relatedness score. The *delta-airliner* relation which frequently co-occurs in training data is captured by the *overlap layers*. The *12-football* pair shows a relatedness between sports and numbers.

cific points in the sequence when computing its output. However, in this case, the attention attends the wrong context, as there are many words have no correlation or do not correspond to actual words. On the other hand, USE-T seems to require a shorter hypothesis list to get top performance when the right word is in the hypothesis list.

**Comparing with word level model:** We also compare our result with current state-of-the-art word embeddings trained on a large general text using *glove* and *fasttext*. The word model used only object and place information, and ignored the caption. Our proposed models achieve better performance than our TWE previous model (Sabir et al., 2018), that trained a word embedding (Mikolov et al., 2013) from scratch on the same task.

**Similarity to probabilities:** After computing the cosine similarity we need to convert that score to probabilities. As we proposed in previous work (Sabir et al., 2018) we obtain the final probability combining (Blok et al., 2003) the similarity score, the probability of the detected context (provided by the object/place classifier), and the probability of the candidate word (estimated from a 5M token corpus) (Lison and Tiedemann, 2016).

**Effect of Unigram Probabilities:** Ghosh et al. (2017) showed the utility of a language model (LM) when the data is too small for a DNN, obtaining significant improvements. Thus, we introduce a basic model of unigram probabilities with Out-of-vocabulary (OOV) words smoothing. The model is applied at the end, to re-rank out false positive short words, and has the main goal of re-ranking out less probable word overranked by the deep model. As seen in Table 1, the introduction of this unigram lexicon produces the best results.

**Human performance**: To estimate an upper bound for the results, we picked 33 random pictures from the test dataset and had 16 human subjects try to select the right word among the top $k = 5$ candidates produced by the baseline text spotting system. Our proposed model performance on the same images was 57%. Average human performance was 63% (highest 87%, lowest 39%).

## 7 Conclusion

In this work, we propose a simple deep learning architecture to learn semantic relatedness between word-to-word and word-to-sentence pairs, and show how it outperforms other semantic similarity scorers when used to re-rank candidate answers in the Text Spotting problem.

In the future, we plan using the same approach to tackle similar problems, including lexical selection in Machine Translation, or word sense disambiguation (Lala and Specia, 2018). We believe our approach could also be useful in multimodal machine translation, where an image caption must be translated using not only the text but also the image content (Barrault et al., 2018). Tasks that lie at the intersection of computer vision and NLP, such as the challenges posed in the new BreakingNews dataset (popularity prediction, automatic text illustration) could also benefit from our results (Ramisa et al., 2018).

## Acknowledgments

We would like to thank José Fonollosa, Marta Ruiz Costa-Jussà and the anonymous reviewers for discussion and feedback. This work was supported by the KASP Scholarship Program and by the MINECO project HuMoUR TIN2017-90086-R.

## References


Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Loïc Barrault, Fethi Bougares, Lucia Specia, Chiraag Lala, Desmond Elliott, and Stella Frank. 2018. Findings of the third shared task on multimodal machine translation. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*.

Sergey Blok, Douglas Medin, and Daniel Osherson. 2003. Probability from similarity. In *AAAI Spring Symposium on Logical Formalization of Commonsense Reasoning*.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Timothy Dozat. 2016. Incorporating nesterov momentum into adam.

Suman K Ghosh, Ernest Valveny, and Andrew D Bagdanov. 2017. Visual attention models for scene text recognition. In *2017 14th IAPR International Conference on Document Analysis and Recognition*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Synthetic data and artificial neural networks for natural scene text recognition. *arXiv preprint arXiv:1406.2227*.

Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2016. Reading text in the wild with convolutional neural networks. *International Journal of Computer Vision*, 116(1):1–20.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*.

Chulmoo Kang, Gunhee Kim, and Suk I Yoo. 2017. Detection and recognition of text embedded in online images via neural context models. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Sezer Karaoglu, Ran Tao, Jan C van Gemert, and Theo Gevers. 2017. Con-text: Text detection for fine-grained object classification. *IEEE Transactions on Image Processing*, 26(8):3965–3980.

Chiraag Lala and Lucia Specia. 2018. Multimodal lexical translation. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation*.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*.

Pierre Lison and Jörg Tiedemann. 2016. Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning*.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Shitala Prasad and Adams Wai Kin Kong. 2018. Using object information for spotting text. In *The European Conference on Computer Vision*.

Colin Raffel and Daniel PW Ellis. 2015. Feed-forward networks with attention can solve some long-term memory problems. *arXiv preprint arXiv:1512.08756*.

Arnau Ramisa, Fei Yan, Francesc Moreno-Noguer, and Krystian Mikolajczyk. 2018. Breakingnews: Article annotation by image and text processing. *IEEE*

*Transactions on Pattern Analysis and Machine Intelligence*, 40(5):1072–1085.

Ahmed Sabir, Francesc Moreno-Noguer, and Lluís Padró. 2018. Visual re-ranking with natural language understanding for text spotting. In *Asian Conference on Computer Vision*.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th international ACM SIGIR Conference on Research and Development in Information Retrieval*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Ming Tan, Cicero Dos Santos, Bing Xiang, and Bowen Zhou. 2016. Improved representation learning for question answer matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Andreas Veit, Tomas Matera, Lukas Neumann, Jiri Matas, and Serge Belongie. 2016. Coco-text: Dataset and benchmark for text detection and recognition in natural images. *arXiv preprint arXiv:1601.07140*.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Xingyou Wang, Weijie Jiang, and Zhiyong Luo. 2016. Combination of convolutional and recurrent neural network for sentiment analysis of short texts. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*.

Wenpeng Yin and Hinrich Schütze. 2016. Multichannel variable-size convolution for sentence classification. *arXiv preprint arXiv:1603.04513*.

Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. 2014. Learning deep features for scene recognition using places database. In *Advances in Neural Information Processing Systems*.

Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis Lau. 2015. A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630*.

Anna Zhu, Renwu Gao, and Seiichi Uchida. 2016. Could scene context be beneficial for scene text detection? *Pattern Recognition*, 58:204–215.

# Delta-training: Simple Semi-Supervised Text Classification using Pretrained Word Embeddings

**Hwiyeol Jo**
Seoul National University
hwiyeolj@gmail.com

**Ceyda Cinarel**
Seoul National University
snu.ceyda@gmail.com

## Abstract

We propose a novel and simple method for semi-supervised text classification. The method stems from the hypothesis that a classifier with pretrained word embeddings always outperforms the same classifier with randomly initialized word embeddings, as empirically observed in NLP tasks. Our method first builds two sets of classifiers as a form of model ensemble, and then initializes their word embeddings differently: one using random, the other using pretrained word embeddings. We focus on different predictions between the two classifiers on unlabeled data while following the self-training framework. We also use early-stopping in meta-epoch to improve the performance of our method. Our method, Delta-training, outperforms the self-training and the co-training framework in 4 different text classification datasets, showing robustness against error accumulation.

## 1 Introduction

### 1.1 Motivation

Classifiers using deep learning algorithms have performed well in various NLP tasks, but the performance is not always satisfactory when utilizing small data. It is necessary to collect more data for acquiring better performance. Although collecting unlabeled text data is relatively easy, labeling in and of itself requires a considerable amount of human labor. In order to incorporate unlabeled data into a task, we have to label the data in accordance to class policies of the task, but the labeling process requires not only human labor but also domain knowledge on the classes.

Semi-supervised learning (Li and Liu, 2003; Zhu, 2006; Chapelle et al., 2009) can be considered a potential solution that utilizes both labeled data and unlabeled data when building a classifier. The simplest form of semi-supervised learning is self-training (Yarowsky, 1995), which first builds a classifier using labeled data, and then label the unlabeled data. After which the most confident label prediction is added to training set and the process is repeated. The unlabeled data can help address data sparsity, but classification errors might be accumulated along the process.

We combine self-training with the hypothesis that a classifier with pretrained word embeddings ($m_{emb}$) is always better than a classifier with randomly initialized word embeddings ($m_{rand}$), as empirically observed in various NLP tasks (Turian et al., 2010). Our method follows the self-training framework but rather focuses on the different predictions of two sets of classifiers on unlabeled data. Therefore we can filter out incorrectly predicted data and correctly predicted data by both classifiers, which are less informative to the classifiers. On the other hand, differently predicted data are much more informative since much of the performance gap between the classifiers come from the different predictions. Although the differently predicted data may introduce some noise like correctly predicted by $m_{rand}$ but incorrectly predicted by $m_{emb}$, we believe that the noise is relatively small when compared with benefits.

### 1.2 Contributions

Our contributions in this paper can be summarized as follows:

- We propose a variation of self-training framework: Delta($\Delta$)-training, which harnesses differently predicted labels between two sets of classifiers.
- Along with early-stopping in iterative training process, our framework outperforms the conventional self-training and co-training framework.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3458–3463,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

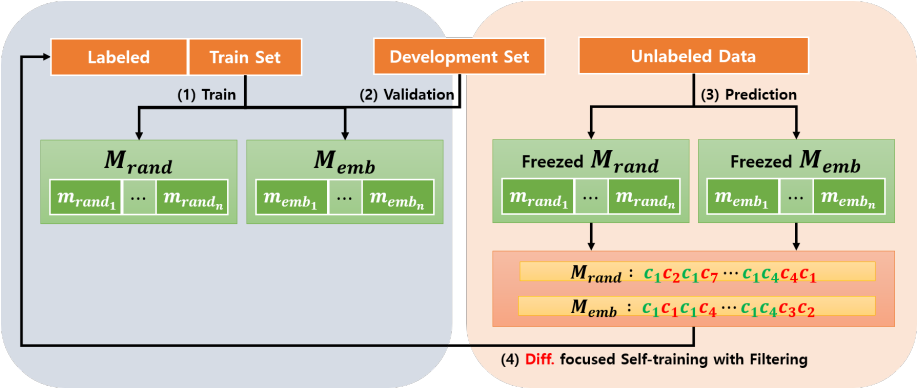


Figure 1: The flow of Δ-training framework. $M_{rand}$ and $M_{emb}$ are ensembled classifiers using randomly initialized word embeddings and pretrained word embeddings, respectively. (1) We first train the sets of classifiers using training set, (2) do early-stopping using development set, (3) predict the labels of unlabeled data using the sets of classifiers trained at (1), and (4) select the high confidence labels differently predicted by each set of classifiers, adding them to training set. While following the framework, we do early-stopping in meta-epoch with the development set.

## 2 Preliminary

**Self-training.** Given labeled data $\{(x_1, y_1), \cdots, (x_n, y_n)\}$ and unlabeled data $\{(x_{n+1}), \cdots, (x_{n+l})\}$, self-training (Yarowsky, 1995) first builds a model $m$ using labeled data. Next, it simply predicts the unlabeled data using pretrained model $m$. If the confidence score of the predicted label is higher than a predefined threshold $T$, then adds the label-by-prediction data to the training set. This simple approach has generated variations such as calibration (Guo et al., 2017), and online learning (Abney, 2007).

**Pretrained Word Embeddings.** Pretrained word embeddings are based on the distributed representation hypothesis that a word can be represented as an n-dimensional vector (Mikolov et al., 2013). Most of the algorithms are based on the basic idea of CBoW and skip-gram. Both algorithms learn word vectors by maximizing the probability of occurrence of a center word given neighbor words or neighbor words given a center word. With this unsupervised approach, we can represent semantic and relational information. The pretrained word vectors from very large corpus are used to initialize word vectors for classifiers, performing better than randomly initialized word vectors (Turian et al., 2010).

**Model Ensemble.** Model ensemble (Opitz and Maclin, 1999) is using a combination of models to increase accuracy and get confidence scores on predictions. There are two types of ensemble methods, bagging and boosting. Bagging averages the predictions over a collection of classifiers whereas boosting weights the vote with a collection of classifiers.

## 3 Proposed Method: Δ-training

The overall process of our framework is illustrated in Figure 1.

### 3.1 Different Prediction focused Self-training

Our method consists of two classifiers: one is randomly initialized ($m_{rand}$; random network), and the other is using pretrained word vectors ($m_{emb}$; embedded network). When ensembling, we duplicate the same classifier, $M_{rand} = (m_{rand_1}, \cdots, m_{rand_n})$ and $M_{emb} = (m_{emb_1}, \cdots, m_{emb_n})$, respectively.

We adopt bagging to increase the cases that (1) both $m_{rand}$ and $m_{emb}$ predict the labels of data correctly, and (2) $m_{rand}$ predicts incorrectly but $m_{emb}$ predicts correctly. Model ensemble is also used to pick out label-by-prediction data with high confidence, which will be used for self-training. Intuitively, the benefits of Δ-training are maximized when the performance gap between two sets of classifiers is large. Also, we can ensure the performance gap not only by using pretrained embeddings but also through the ensemble setting.

First, we train the classifiers using the training set with early-stopping, and return their predictions on unlabeled data. We consider the predictions of $M_{emb}$ on the unlabeled data as label-by-prediction since $M_{emb}$ always outperforms $M_{rand}$ according

to our hypothesis. The hypothesis will be confirmed in Section 7.

After labeling the unlabeled data, we select the data with conditions that (a) each ensembled classifiers are predicting the same class, and (b) the predictions of $M_{rand}$ and $M_{emb}$ are different. Condition (a) helps to pick out the data labeled with high confidence by the classifiers and Condition (b) helps to pick out the data which is incorrect in $M_{rand}$ but correct in $M_{emb}$. The ratio in which labels might be correct in $M_{rand}$ but incorrect in $M_{emb}$ is relatively small than vice versa (will be also presented in Section 7). We add the selected data and its pseudo-label by $M_{emb}$ to training set, and then train the classifiers again from the very first step to validate our hypothesis. If we do not start from the very first step, it might cause $M_{emb}$ to overfit and perform worse than $M_{rand}$.

We denote one such iterative process, training and pseudo-labeling, as a *meta-epoch*.

### 3.2 Early-Stopping in Meta-Epoch

Using the development set in every meta-epoch, we do early-stopping during the different prediction focused self-training. As $M_{rand}$ keeps learning based on the predictions of $M_{emb}$, the size of data which is incorrectly predicted by $M_{rand}$ but correctly predicted by $M_{emb}$ will decrease. Likewise, the size of data which is correctly predicted by both $M_{rand}$ and $M_{emb}$ will increase. Therefore, after early-stopping in meta-epoch, we simply add all the unlabeled data with its pseudo-labels to the training set. In this way, we can fully benefit from different prediction focused self-training and save training time.

## 4 Experiment Data

We use **GloVe** (Pennington et al., 2014) `glove.42B.300d` as word embedding for $M_{emb}$. We also perform word vector post-processing method, **extrofitting** (Jo, 2018), to improve the effect of initialization with pretrained word embeddings on text classification, as described in their paper.

We use 4 text classification datasets; **IMDB review** (Maas et al., 2011), **AGNews**, **Yelp review** (Zhang et al., 2015), and **Yahoo!Answers** (Chang et al., 2008).

| | **IMDB Review** | **AGNews** | **Yelp Review** | **Yahoo Answer** |
|---|---|---|---|---|
| Train | 212 | 1,020 | 5,525 | 1,136 |
| Test | 25,000 | 7,600 | 50,000 | 23,595 |
| Dev | 38 | 180 | 975 | 201 |
| Unlabed | 24,750 | 118,800 | 643,500 | 132,366 |
| #Class | 2 | 4 | 5 | 17 |

Table 1: The data split information of text classification datasets. We use 1% of the training data and remove the labels of the remaining training data, using them as unlabeled data.

## 5 Experiment

### 5.1 Data Preparation

To emulate the environment with only a few labeled training examples, we take only 1% of the original training set and remove the label of the remaining training set, which will be referred to as unlabeled data. Next, we assign 15% of the training set to the development set. The development set is used to determine early-stopping for every epoch and meta early-stopping for every meta-epoch. The split data size is presented in Table 1. IMDB reviews dataset has own unlabeled data but we do not use them in order to track and report on the difference between predicted labels and true labels.

### 5.2 Classifier

We select TextCNN (Kim, 2014) as our classifier. Due to the simple but high performance nature of TextCNN, the model can represent deep learning classifiers, and is easy to ensemble as well. We use the first 100 words of data in 300 dimensional embedding space. The model consists of 2 convolutional layers with the 32 channels and 16 channels, respectively. We adopt multiple sizes of kernels–2, 3, 4, and 5, followed by ReLU activation (Hahnloser et al., 2000) and max-pooled. We concatenate them after every max-pooling layer. We train the model using Adam optimizer (Kingma and Ba, 2014) with 1e-3 learning rate.

### 5.3 Ensemble Settings

In the experiment, the number of embedded model ensemble ($M_{emb}$) is 3 and we do not ensemble random model ($M_{rand} = m_{rand}$) for simplicity. The baselines also use the same ensemble settings for fair comparison.

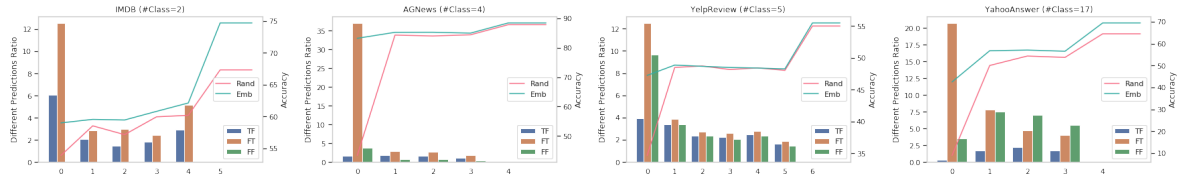


Figure 2: The training curve and the ratio of differently predicted label during Δ-training. The x-axis indicates a training process (meta-epoch). TF, FT, and FF denote that correctly predicted by $M_{rand}$ but incorrectly predicted by $M_{emb}$, incorrectly predicted by $M_{rand}$ but correctly predicted by $M_{emb}$, and incorrectly predicted by both $M_{rand}$ and $M_{emb}$, respectively. We extend the end of performance lines to indicate final accuracy.

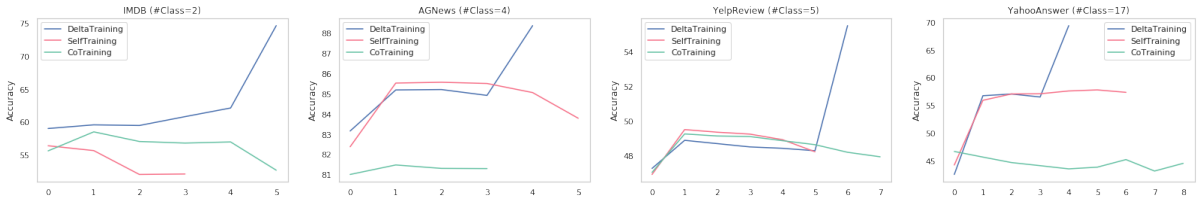


Figure 3: The performance of Δ-training compared with self-training and co-training. Our method largely improves the performance in binary classification but slightly show degraded performance prior to early-stopping in meta-epoch. Then, Δ-training finally brings significant performance gain after meta-level early stopping, training on all the remaining pseudo-labeled data.

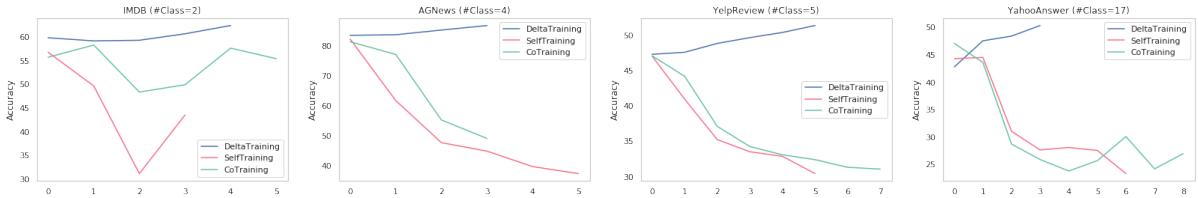


Figure 4: The performance of Δ-training and other frameworks on unlabeled data. We recover the removed true labels and track the model performance. Δ-training is robust against error accumulation.

## 6 Related Works

Δ-training is closely related to **Self-training** (Yarowsky, 1995), and **Tri-training with Disagreement** (Søgaard, 2010). Tri-training (Zhu, 2006) uses 3 classifiers to vote their classification results and labels them if all the classifiers agree with the prediction. Its extension, Tri-training with Disagreement, also uses 3 classifiers but the method utilizes a disagreement that pseudo-labels on unlabeled data if two classifiers agree with the labels but one classifier disagrees with the labels. The differences with our method respectively are (1) we harness different predictions of classifiers, and (2) we use a single model architecture where word embeddings are initialized differently.

The existing semi-supervised solutions using 2 classifiers such as **Co-training** (Blum and Mitchell, 1998) cannot be fully compared with ours for (2) that a single architecture should be used. The method is built on 2 different classifiers as having different views on data, and harnesses one's pseudo-labels to train the other classifier. Instead, we imitate the co-training as if $M_{rand}$ and $M_{emb}$ have different views on the data.

Refer to Ruder and Plank's work (2018) for further knowledge on those related works.

## 7 Result

The training curve and the ratio of differently predicted labels are presented in Figure 2. The training curves at x=0, at which point Δ-training is not applied yet, confirms our hypothesis–a classifier with pretrained word embeddings always outperforms the same classifier with randomly initialized word embeddings. Also, the ratio in which labels are correct in $M_{rand}$ but incorrect in $M_{emb}$ is relatively small (TF<FT) than vice versa. The ratio in which labels are incorrect both in $M_{rand}$ and $M_{emb}$ (FF) changes according to baseline accuracy and the number of classes.

The performance of Δ-training compared with

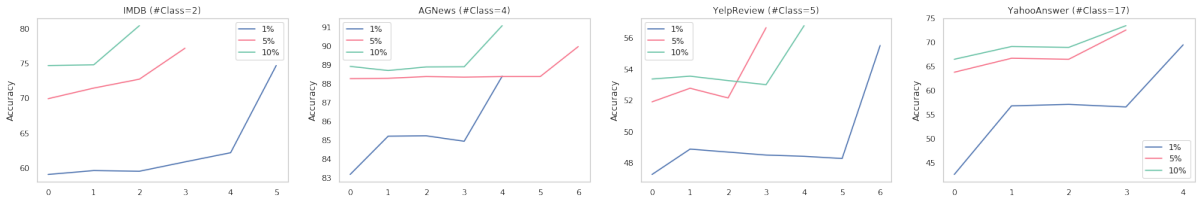


Figure 5: The performance of Δ-training with respect to initial training data size. Δ-training performs well in different training data size and is more useful when the training data is scarce.

self-training and co-training is presented in Figure 3. Our method largely outperforms the conventional self-training and co-training framework in binary class classification. In multi-class classification, picking different predictions is less effective because the data could be incorrectly predicted by both $M_{emb}$ and $M_{rand}$. Therefore, after the early-stopping in meta-epoch, we simply add all the unlabeled data with its pseudo-labels to training set, which finally brings significant performance gain. In Figure 4, we observe that the performance of self-training and co-training decreases in unlabeled data after a few meta-epochs because of accumulated classification errors. On the other hand, our method is robust against error accumulation. As a result, the process of adding all the unlabeled data with its pseudo-labels to training set starts from enhanced and robust models.

We also report the effect of initial training data size in Figure 5. The result shows that Δ-training is more useful when the training data is scarce and also Δ-training works well even when there is relatively more data.

## 8 Conclusion

In this paper, we propose a novel and simple approach for semi-supervised text classification. The method follows the conventional self-training framework, but focusing on different predictions between two sets of classifiers. Further, along with early-stopping in training processes and simply adding all the unlabeled data with its pseudo-labels to training set, we can largely improve the model performance. Our framework, Δ-training, outperforms the conventional self-training and co-training framework in text classification tasks, showing robust performance against error accumulation.

## Acknowledgments

The authors would like to thank Sang-Woo Lee, Woo-Young Kang, Dae-Soo Kim, and Byoung-Tak Zhang for helpful comments. Also, we greatly appreciate the reviewers for critical comments.

This work was partly supported by the Korea government (2015-0-00310-SW.StarLab, 2017-0-01772-VTT, 2018-0-00622-RMI, 2019-0-01367-BabyMind, 10060086-RISF, P0006720-GENKO), and the ICT at Seoul National University.

## References

Steven Abney. 2007. *Semisupervised learning for computational linguistics*. Chapman and Hall/CRC.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM.

Ming-Wei Chang, Lev-Arie Ratinov, Dan Roth, and Vivek Srikumar. 2008. Importance of semantic representation: Dataless classification. In *AAAI*, volume 2, pages 830–835.

Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. 2009. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330.

Richard HR Hahnloser, Rahul Sarpeshkar, Misha A Mahowald, Rodney J Douglas, and H Sebastian Seung. 2000. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405(6789):947.

Hwiyeol Jo. 2018. Expansional retrofitting for word vector enrichment. *arXiv preprint arXiv:1808.07337*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Xiaoli Li and Bing Liu. 2003. Learning to classify texts using positive and unlabeled data. In *IJCAI*, volume 3, pages 587–592.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pages 142–150. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

David Opitz and Richard Maclin. 1999. Popular ensemble methods: An empirical study. *Journal of artificial intelligence research*, 11:169–198.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Sebastian Ruder and Barbara Plank. 2018. Strong baselines for neural semi-supervised learning under domain shift. *arXiv preprint arXiv:1804.09530*.

Anders Søgaard. 2010. Simple semi-supervised training of part-of-speech taggers. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 205–208. Association for Computational Linguistics.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 189–196. Association for Computational Linguistics.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Xiaojin Zhu. 2006. Semi-supervised learning literature survey. *Computer Science, University of Wisconsin-Madison*, 2(3):4.

# Visual Detection with Context for Document Layout Analysis

**Carlos X. Soto**
Brookhaven National Laboratory
Upton, New York, USA
csoto@bnl.gov

**Shinjae Yoo**
Brookhaven National Laboratory
Upton, New York, USA
sjyoo@bnl.gov

## Abstract

We present 1) a work in progress method to visually segment key regions of scientific articles using an object detection technique augmented with contextual features, and 2) a novel dataset of region-labeled articles. A continuing challenge in scientific literature mining is the difficulty of consistently extracting high-quality text from formatted PDFs. To address this, we adapt the object-detection technique Faster R-CNN for document layout detection, incorporating contextual information that leverages the inherently localized nature of article contents to improve the region detection performance. Due to the limited availability of high-quality region-labels for scientific articles, we also contribute a novel dataset of region annotations, the first version of which covers 9 region classes and 822 article pages. Initial experimental results demonstrate a 23.9% absolute improvement in mean average precision over the baseline model by incorporating contextual features, and a processing speed 14x faster than a text-based technique. Ongoing work on further improvements is also discussed.

## 1 Introduction

Mining scientific literature at scale for information that can be processed automatically is a valuable and much sought after technique for domain researchers and data scientists. Whereas mass processing of articles was once largely limited to keyword searches and citation crawling, modern natural language processing techniques can now deeply search for specific and broad concepts, explore relationships, and automatically extract useful information from text.

However, getting the article text in the first place is not always simple. Though some publications offer full-text articles in open-access, machine-readable formats (e.g. plain text, HTML or XML),

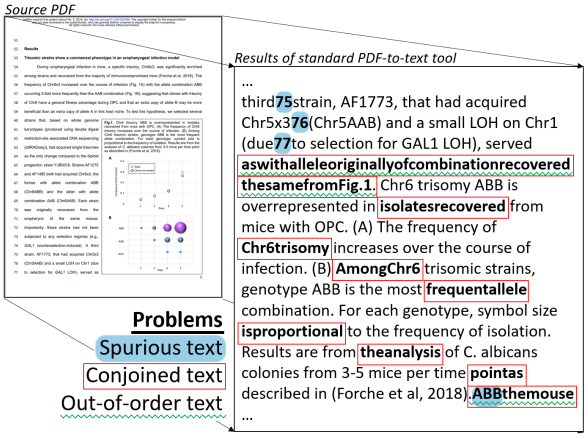


Figure 1: Text extraction from PDF articles is generally a noisy process because of the significant inconsistencies in PDF publishing tools and the edit-and-republish nature of print-ready document preparation by journal publishers and document hosting services.

this is far from the norm. Instead, much of the scientific record remains in PDF files that are much more challenging to automatically process. This challenge remains whether the PDFs originate from scanned pages (thus requiring OCR) or were published from digital source material. Although there are tools for automatically extracting text from PDFs, formatting is highly inconsistent. Standard tools often mix headers, footers, table and figure captions, page numbers, and other extraneous text into the main text being extracted (Bast and Korzen, 2017). And text order is not always well preserved (Figure 1). This is not quite as problematic if text is extracted for simple tasks like keyword searches. However, more complex tasks like Named-Entity Recognition often rely on contiguous, cleanly segmented text for successful processing. And while manually cleaning and reformatting text may be an option for small sets of documents, this quickly becomes impractical when dealing with larger corpora.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3464–3470,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

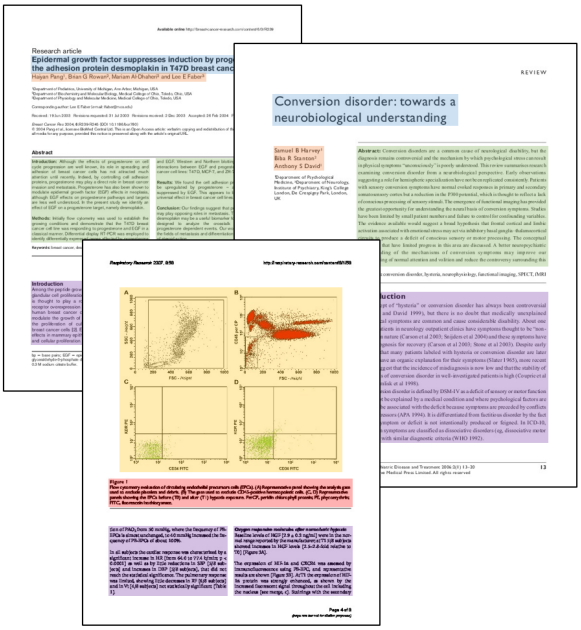

Figure 2: Examples of ground-truth region labels, showing varying journal styles. Note that even when text is illegibly small, the salient regions are visually apparent (e.g. title, abstract, figure caption, etc.).

To facilitate large-scale automatic knowledge extraction from scientific articles, this paper presents an adaptation of the Faster R-CNN object detection model for the task of document layout detection, with the addition of contextual features. The method **visually** detects major regions in article pages and classifies them into a set of labels, including main-body text, tables, figures, and others (Figure 2). To train and evaluate the detection models, a novel dataset of 100 region-annotated scientific articles was created, totaling 822 labeled pages. This work is an ongoing effort: the dataset is being expanded, additional contextual features are being developed, and further evaluation is being conducted.

## 2 Related Work

In addition to the numerous tools that exist for extracting text from PDFs – Bast and Korzen (2017) provide a quantitative evaluation of 14 common ones – there are a variety of approaches for analyzing documents to determine their contents' layout and/or extract information of particular types.

Systems like CERMINE (Tkaczyk et al., 2015) and OCR++ (Singh et al., 2016) extract the raw or processed markup from "born-digital" PDFs (e.g. using tools like pdf2xml) and apply a variety of text processing methods to deduce document structure and apply semantic labels to text blocks. These methods may be rule-based (e.g. regular expressions and heuristics) or machine-learning based (e.g. SVM classification). Such approaches may achieve high-quality extraction and labeling. However they rely on extracted PDF source markup (not always available, e.g. for scanned PDFs), work only on text blocks (ignoring tables and figures, which may also be valuable), and are typically quite slow (2 to 10 seconds or more per article). Analogous approaches for web pages – such as those by Cai et al. (2003) and Song et al. (2004) – also examine the document source (in this case the page DOM) to build representations for further processing.

Alternatively, visual-based techniques for document layout analysis have tended to focus on text segmentation (Tran et al., 2015), (Chen et al., 2015a), (Chen et al., 2015b), (Garz et al., 2016), especially for historical documents; Eskenazi et al. (2017) survey dozens of such methods. More closely aligned with the aim of this work are those by Lang et al. (2018), who use HOG features and random forests to recognize text, images, charts and tables in newspapers, Oliveira and Viana (2017), who uss 1D CNNs to recognize text, images, and tables in scientific articles, and finally Singh et al. (2018), who use a LSTD model to recognize various customizable text and image classes in multiple domains, with mixed results.

Differently from these previous methods, the approach in this paper aims to leverage contextual information in a modern visual object detection approach. Of particular importance to this effort is using a high-quality dataset with labels for relevant key regions of scientific articles. Though some of the cited works in this section published their own custom datasets, these proved to be too granular, too coarse or too noisy for use in this work (see Section 4).

## 3 Visual Document Layout Detection

A modern visual approach is taken to document region detection, with the aim to produce labeled bounding boxes for regions of interest (ROIs) in each document page. Although the text contents in many regions may be useful in classification, our vision-only approach has the benefit of working in any language, and can yield impressive performance even without text features (see Section 5). To be clear: our approach does not access a

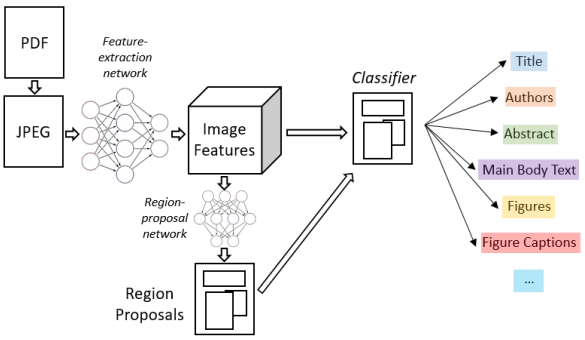


Figure 3: Model for visual region detection using Faster R-CNN. After rendering each page of a PDF to a JPEG and extracting its image features using a convolutional ResNet-101 network, regions of interest are proposed and classified by two additional network which are trained on ground-truth labels.

document's source at all (neither embedded text, nor underlying PDF markup), instead relying only on JPEG renderings of the document's pages.

The baseline technique used for this work is the highly successful object detector Faster R-CNN (Ren et al., 2015) – a so-called two stage detection model. This model takes an image as input and first predicts candidate regions of interest (i.e. image regions that are likely to contain objects), then classifies each candidate region into one of a predefined set of object classes. The output is a set of bounding boxes identifying recognized objects in the image (e.g. people, cars, dogs, etc.). Algorithmically, Faster R-CNN relies on several component neural networks, beginning with a deep convolutional neural network – ResNet-101 (He et al., 2016), specifically – to perform feature extraction on the input image. The resulting feature map is then fed into a region-proposal network (which generates candidate object bounding boxes) and a classification network (which assigns region labels to those candidates). A regressor network also fine-tunes the ROI bounding box parameters for better fit. Figure 3 shows how this model can be applied directly to the task of detecting salient regions in images of document pages.

An important aspect of standard object detection techniques like Faster R-CNN, is that they only use image features within a region of interest to label that region. This approach is advantageous to general purpose object detection, because objects may be located anywhere in a photo or video., and has been highly successful even when objects are cropped or occluded by other objects (Lin et al., 2014). However, in domains such as document layout detection, where the 'objects' to be detected are generally well-structured elements of the page (and indeed of a collected set of pages), it is helpful to include contextual information to determine the label of a region. To this end, our approach incorporates contextual information about pages and ROI bounding boxes, encoded as additional features for the classification and fine-tuning regression stages of the model.

The position, size, and page number of a candidate ROI in a document are valuable contextual clues to the true label of that region. They help disambiguate region classes that are otherwise visually similar, such as abstracts, body text, and figure captions (without context, these all appear simply as isolated blocks of text). Further, if two proposed regions overlap, their size and position can help determine which is more likely to fit a true document region more closely. Additional contextual features are being explored in ongoing work (see Section 6), however these were selected first for their ease of integration and anticipated high impact on detection performance.

Another important consideration is the existence of newer, even more successful detection models like YOLOv3 (Redmon and Farhadi, 2018) and RetinaNet (Lin et al., 2017), which use a single-stage detection paradigm. Our rationale in selecting Faster R-CNN as a baseline model to build upon in this work is that the separate region proposal and classification models make it easier to intercept ROI dimensions from the first model and embed them as additional features (along with page numbers) for input to the classification and bounding-box regression models. We do, however, compare our results to single-stage models and intend to explore these further.

## 4 Novel Labeled Dataset

Existing region-labeled datasets of scientific articles proved too noisy for our visual region detection method. Training the model with the full GROTOAP2 dataset (Tkaczyk et al., 2014), for example, yielded a best overall detection performance of 5.1% mean average precision (mAP) over all 22 labels. The key problems were that many of the regions were far too granular (e.g. each cell of a table might have its own bounding box labeled 'table'), and many regions were mislabeled or misaligned. The labeling noise was so pronounced that by simply filtering and carefully




Figure 4: Incorporating page and bounding box context yielded a relative improvement of 51.6% over baseline Faster R-CNN performance (23.9% absolute: 70.3% vs 46.4%). Our model also outperforms reference state-of-the-art single stage models: RetinaNet (58.1% peak mAP) and YOLOv3 (68.9%, after 49 epochs). Small regions (authors, table captions) remain challenging and bring down average performance. All results @ 0.5 IOU.

coalescing bounding boxes for 'body content' labels, single-class detection performance was dramatically improved to 72% AP (up from 18% for unfiltered labels). However, the label-cleaning process was very time consuming and this improvement did not generalize to other classes.

Therefore, a novel dataset was created. We rendered PDF articles to JPEG image sets (using the ImageMagick package, at 72dpi), and used an open source utility (Tzutalin, 2015) to manually annotate regions. Version 1 of this dataset consists of region annotations for 100 scientific articles sampled from the PMC Open Access set. The collection will be available at `https://github.com/cxsoto/article-regions`, and includes scripts to download and render the original article PDFs to images, as well as to convert the annotations to various formats. The default format is PASCAL VOC. Nine labeled region classes are included in the annotations:

- *Title*. Includes subtitle, if present.
- *Authors*. Author names only, where possible (i.e. no affiliations, etc.).
- *Abstract*. Abstract text only, where possible.
- *Body*. All main article text, including section headers. Contiguous, where possible.
- *Figure*. Any labeled figures (i.e. no journal logos, etc.).
- *Figure Caption*. The caption text for a figure.
- *Table*. Including only the tabular contents, where possible. Includes adjacent notes or commentary if short and table-aligned.
- *Table Caption*. Main table caption as well as paragraph-form table commentary that follows some tables.
- *References*. Full bibliography, not including post-references notes (e.g. author bios, journal marketing, etc.).

The bounding boxes for these regions were created to be consistent and tight-fitting, with typical padding of 2-4 pixels. In general, the bounding boxes do not overlap. Each page of each article is labeled and processed separately. Due to labeling precision, annotation took approximately 20 to 60 seconds per image, depending on the contents. However, as this process does not involve specialized knowledge, it can be readily crowd-sourced.

For version 2 of the dataset, we intend to include labels for Equations, Sub-Figures, and Author Affiliations. It is expected to cover approximately 1000 articles from more varied sources, including pre-prints from arXiv and similar repositories.

## 5 Implementation and Experiments

Using the novel labeled dataset described in Section 4, a baseline model was trained using a standard Faster R-CNN implementation (Yang et al., 2017). The model was trained using a single

NVIDIA P100 GPU for 30 epochs on 600 images, and tested on the remaining 222 in 5 randomized sessions, using a ResNet-101 base network pretrained on ImageNet (Russakovsky et al., 2015), with a batch size of 8, Adam optimizer (Kingma and Ba, 2014), and a starting learning rate of 0.0001, with decay of 0.1 every 5 epochs. Standard anchor scales of [8, 16, 32] and anchor ratios of [0.5, 1.0, 2.0] were used. At a intersection-over-union (IOU) threshold of 0.5, the model achieved a mean average precision (mAP) of 46.38% on all nine region labels, with peak class performance on 'body' regions (87.49%) and lowest performance on 'authors' (1.22%).

Contextual information was incorporated into the classification and regression stages of the model for article page information and proposed ROI bounding boxes. Page context consisted of the page number of the current image in its article, and the number of pages in the article, both normalized to the average page length of articles in the dataset (8.22). Bounding box context consisted of the position and size of the proposed region of interest, normalized to the dimensions of the image. Both were appended to the pooled feature vectors for batched ROIs sent to the classification and bounding-box regression models. The input size of these models were increased accordingly. Retraining the model in the same manner as the baseline yielded a mean average precision of 70.3%, with peak class-performance of 93.58% on 'body' regions and a low performance of 10.34% on 'authors'. The second-lowest performance was on 'table captions' (30.8%).

Figure 4 shows per-class performance over 30 training epochs, as well as comparative performance against the baseline Faster R-CNN model and reference model implementations of YOLOv3 (Linder-Noren, 2018) and RetinaNet (Henon, 2018). Most models plateaued early on this small dataset, except YOLOv3 which peaking at 68.9% after 49 epochs (beyond the figure bounds, but still below our model's results). Processing time for our model averaged 0.65 seconds per article. By contrast, CERMINE averaged 9.4 seconds per article on the same set of articles, on the same machine.

With the exception of two small region classes (authors and table captions), detection performance quickly reaches a rather high plateau (83.63% not including these two classes), especially considering the model has been trained on less than 100 labeled articles. The size and placement of these region classes makes them difficult to localize and distinguish from other classes. The ongoing work described in Section 6 focuses on incorporating additional contextual features that are expected to improve overall performance, and especially target problematic classes like these.

## 6 Discussion and Ongoing Work

Visual layout detection in documents benefits directly from advances in state-of-the-art object detection techniques, yet is also well suited for optimizations that exploit the structured nature of documents. Adding even very simple contextual information (page numbers and region-of-interest position and size) yielded a relative performance improvement over the baseline of over 50% – while also outperforming state-of-the-art single stage detectors. Unsurprisingly, the position of a region in a scientific article is strongly correlated with its label. Similarly, other contextual information may be correlated with a region's label and placement.

In ongoing work, additional contextual features are being explored, including ROI oversampling (i.e. looking at a region's surroundings), random feature map sampling (i.e. looking at patterns in a whole page or whole article), and all-region positional information (i.e. the position of other predicted ROIs in an article). Performance improvements are expected as these features are incorporated, and as additional labeled data is created. Alternative detection frameworks are also being explored.

The value of this work extends beyond text extraction. Visual region detection can serve as a precursor to numerous existing information extraction techniques or adaptations of them, including parsing of reference (Lammey, 2015), tables (Rastan et al., 2015), and equations (Smithies et al., 2001), as well as data extraction from figures (Tummers, 2006). And so it is valuable to all these efforts to be able to accurately and quickly segment document pages into regions of interest.

## Acknowledgments

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Biological and Environmental Research, DE-SC0012704 and BNL LDRD #17-018.

## References


Hannah Bast and Claudius Korzen. 2017. A benchmark and evaluation for text extraction from pdf. In *Proceedings of the 17th ACM/IEEE Joint Conference on Digital Libraries*, pages 99–108. IEEE Press.

Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma. 2003. Vips: a vision-based page segmentation algorithm. *Microsoft Technical Report, MSR-TR-2003-79*.

Kai Chen, Mathias Seuret, Marcus Liwicki, Jean Hennebert, and Rolf Ingold. 2015a. Page segmentation of historical document images with convolutional autoencoders. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 1011–1015. IEEE.

Kai Chen, Mathias Seuret, Hao Wei, Marcus Liwicki, Jean Hennebert, and Rolf Ingold. 2015b. Ground truth model, tool, and dataset for layout analysis of historical documents. In *Document Recognition and Retrieval XXII*, volume 9402, page 940204. International Society for Optics and Photonics.

Sébastien Eskenazi, Petra Gomez-Krämer, and Jean-Marc Ogier. 2017. A comprehensive survey of mostly textual document segmentation algorithms since 2008. *Pattern Recognition*, 64:1–14.

Angelika Garz, Mathias Seuret, Fotini Simistira, Andreas Fischer, and Rolf Ingold. 2016. Creating ground truth for historical manuscripts with document graphs and scribbling interaction. In *2016 12th IAPR Workshop on Document Analysis Systems (DAS)*, pages 126–131. IEEE.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Yann Henon. 2018. Pytorch implementation of retinanet object detection. *https://github.com/yhenon/pytorch-retinanet*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Rachael Lammey. 2015. Crossref text and data mining services. *Science Editing*, 2(1):22–27.

Thomas Lang, Markus Diem, and Florian Kleber. 2018. Physical layout analysis of partly annotated newspaper images. In *Proceedings of the 23rd Computer Vision Winter Workshop*.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Erik Linder-Noren. 2018. A minimal pytorch implementation of yolov3. *https://github.com/eriklindernoren/PyTorch-YOLOv3*.

Dario Augusto Borges Oliveira and Matheus Palhares Viana. 2017. Fast cnn-based document layout analysis. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1173–1180.

Roya Rastan, Hye-Young Paik, and John Shepherd. 2015. Texus: a task-based approach for table extraction and understanding. In *Proceedings of the 2015 ACM Symposium on Document Engineering*, pages 25–34. ACM.

Joseph Redmon and Ali Farhadi. 2018. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252.

Mayank Singh, Barnopriyo Barua, Priyank Palod, Manvi Garg, Sidhartha Satapathy, Samuel Bushi, Kumar Ayush, Krishna Sai Rohith, Tulasi Gamidi, Pawan Goyal, et al. 2016. Ocr++: a robust framework for information extraction from scholarly articles. *arXiv preprint arXiv:1609.06423*.

Pranaydeep Singh, Srikrishna Varadarajan, Ankit Narayan Singh, and Muktabh Mayank Srivastava. 2018. Multidomain document layout understanding using few shot object detection. *arXiv preprint arXiv:1808.07330*.

Steve Smithies, Kevin Novins, and James Arvo. 2001. Equation entry and editing via handwriting and gesture recognition. *Behaviour & information technology*, 20(1):53–67.

Ruihua Song, Haifeng Liu, Ji-Rong Wen, and Wei-Ying Ma. 2004. Learning block importance models for web pages. In *Proceedings of the 13th international conference on World Wide Web*, pages 203–211. ACM.

Dominika Tkaczyk, Pawel Szostek, and Lukasz Bolikowski. 2014. Grotoap2-the methodology of creating a large ground truth dataset of scientific articles. *D-Lib Magazine*, 20(11/12).

Dominika Tkaczyk, Paweł Szostek, Mateusz Fedoryszak, Piotr Jan Dendek, and Łukasz Bolikowski. 2015. Cermine: automatic extraction of structured metadata from scientific literature. *International Journal on Document Analysis and Recognition (IJDAR)*, 18(4):317–335.

Tuan Anh Tran, In Seop Na, and Soo-Hyung Kim. 2015. Separation of text and non-text in document layout analysis using a recursive filter. *TIIS*, 9(10):4072–4091.

B. Tummers. 2006. DataThief III: A program to extract (reverse engineer) data points from a graph. https://datathief.org, accessed: 2019-05-01.

Tzutalin. 2015. Labelimg. *https://github.com/tzutalin/labelImg*.

Jianwei Yang, Jiasen Lu, Dhruv Batra, and Devi Parikh. 2017. A faster pytorch implementation of faster r-cnn. *https://github.com/jwyang/faster-rcnn.pytorch*.

# Evaluating Topic Quality with Posterior Variability

**Linzi Xing**[†] and **Michael J. Paul**[‡] and **Giuseppe Carenini**[†]
† University of British Columbia, Vancouver, Canada
‡ University of Colorado, Boulder, Colorado, USA
{lzxing, carenini}@cs.ubc.ca, mpaul@colorado.edu

## Abstract

Probabilistic topic models such as latent Dirichlet allocation (LDA) are popularly used with Bayesian inference methods such as Gibbs sampling to learn posterior distributions over topic model parameters. We derive a novel measure of LDA topic quality using the *variability* of the posterior distributions. Compared to several existing baselines for automatic topic evaluation, the proposed metric achieves state-of-the-art correlations with human judgments of topic quality in experiments on three corpora.[1] We additionally demonstrate that topic quality estimation can be further improved using a supervised estimator that combines multiple metrics.

## 1 Introduction

Latent Dirichlet Allocation (LDA) (Blei et al., 2003) topic modeling has been widely used for NLP tasks which require the extraction of latent themes, such as scientific article topic analysis (Hall et al., 2008), news media tracking (Roberts et al., 2013), online campaign detection (Paul and Dredze, 2014) and medical issue analysis (Huang et al., 2015, 2017). To reliably utilize topic models trained for these tasks, we need to evaluate them carefully and ensure that they have as high quality as possible. When topic models are used in an extrinsic task, like text categorization, they can be assessed by measuring how effectively they contribute to that task (Chen et al., 2016; Huang et al., 2015). However, when they are generated for human consumption, their evaluation is more challenging. In such cases, interpretability is critical, and Chang et al. (2009); Aletras and Stevenson (2013) have shown that the standard way to evaluate the output of a probabilistic model, by measuring perplexity on held-out data (Wallach et al., 2009), does not imply that the inferred topics are human-interpretable.

A topic inferred by LDA is typically represented by the set of words with the highest probability given the topic. With this characteristic, we can evaluate the topic quality by determining how coherent the set of topic words is. While a variety of techniques (Section 2) have been geared towards measuring the topic quality in this way, in this paper, we push such research one step further by making the following two contributions: (1) We propose a novel topic quality metric by using the *variability* of LDA posterior distributions. This metric conforms well with human judgments and achieves the state-of-the-art performance. (2) We also create a topic quality estimator by combining two complementary classes of metrics: the metrics that use information from posterior distributions (including our new metric), along with the set of metrics that rely on topic word co-occurrence. Our novel estimator further improves the topic quality assessment on two out of the three corpora we have.

## 2 Automatic Topic Quality Evaluation

There are two common ways to evaluate the quality of LDA topic models: *Co-occurrence Based Methods* and *Posterior Based Methods*.

**Co-occurrence Based Methods** Most prominent topic quality evaluations use various pairwise co-occurrence statistics to estimate topic's semantic similarity. Mimno et al. (2011) proposed the **Coherence** metric, which is the summation of the conditional probability of each topic word given all other words. Newman et al. (2010) showed that the summation of the pairwise pointwise mutual information (**PMI**) of all possible topic word pairs is also an effective metric to assess topic quality. Later, in Lau et al. (2014), PMI was replaced

[1] Our code and data are available here.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3471–3477, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

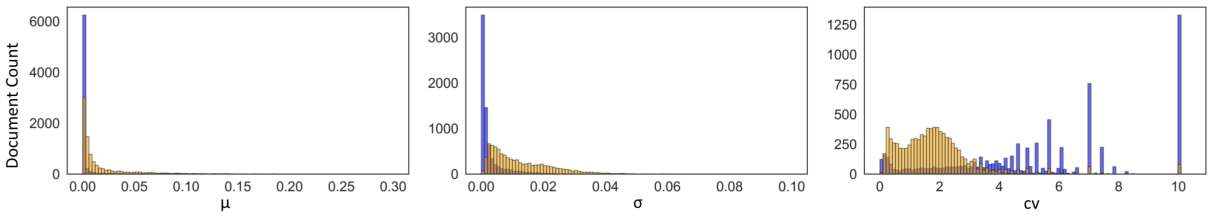


Figure 1: Two example topics and their distributions of $\mu$, $\sigma$ and $cv$ from the NYT corpus. Two topics are: Topic1 (in blue): {*financial, banks, bank, money, debt, fund, loans, investors, funds, hedge*}. Topic2 (in orange): {*world, one, like, good, even, know, think, get, many, got*}. Their human rating scores are 3.4 and 1.0 respectively.

by the normalized pointwise mutual information (**NPMI**) (Bouma, 2009), which has an even higher correlation with human judgments. Another line of work exploits the co-occurrence statistics indirectly. Aletras and Stevenson (2013) devised a new method by mapping the topic words into a semantic space and then computing the pairwise distributional similarity (**DS**) of words in that space. However, the semantic space is still built on PMI or NPMI. Roder et al. (2015) studied a unifying framework to explore a set of co-occurrence based topic quality measures and their parameters, identifying two complex combinations, (named **CV** and **CP** in that paper[2]), as the best performers on their test corpora.

**Posterior Based Method** Recently, Xing and Paul (2018) analyzed how the posterior of LDA parameters vary during Gibbs sampling inference (Geman and Geman, 1984; Griffiths and Steyvers, 2004) and proposed a new topic quality measurement named **Topic Stability**. The Gibbs sampling for LDA generates estimates for two distributions: for topics given a document ($\theta$), and for words given a topic ($\phi$). Topic stability considers $\phi$ and is defined as:

$$stability(\mathbf{\Phi_k}) = \frac{1}{|\mathbf{\Phi_k}|} \sum_{\phi_k \in \Phi_k} sim(\phi_k, \bar{\phi_k}) \quad (1)$$

The stability of topic $k$ is computed as the mean cosine similarity between the mean ($\bar{\phi_k}$) of all the sampled topic $k$'s distribution estimates ($\mathbf{\Phi_k}$) and topic $k$'s estimates from each Gibbs sampler ($\phi_k$). Fared against the co-occurrence based methods, topic stability is parameter-free and needs no external corpora to infer the word co-occurrence. However, due to the high frequency of common words across the corpus, low quality topics may also have high stability, and this undermines the performance of this method.

[2]The framework proposed in Roder et al. (2015) has four stages. Every stage has multiple settings. CV and CP are different at the Confirmation Measure stage, which measures how strongly a set of topic words connect with each other.

| Metric | 20NG | Wiki | NYT | Mean |
|---|---|---|---|---|
| $\mu$ | 0.185 | 0.030 | 0.148 | 0.121 |
| $\sigma$ | 0.480 | 0.295 | 0.600 | 0.458 |
| $cv$ | **0.679** | **0.703** | **0.774** | **0.719** |

Table 1: Pearson's $r$ of each potential metric of posterior variability with human judgments

## 3 Variability Driven from Topic Estimates

In this paper, we also use Gibbs sampling to infer the posterior distribution over LDA parameters. Yet, instead of $\phi$, our new topic evaluation method analyzes estimates of $\theta$, the topic distribution in documents. Let $\mathbf{\Theta}$ be a set of different estimates of $\theta$, which in our experiments will be a set of estimates from different iterations of Gibbs sampling. Traditionally, the final parameter estimates are taken as the mean of all the sampled estimates, $\hat{\theta}_{dk} = \frac{1}{|\mathbf{\Theta}|} \sum_{\theta \in \mathbf{\Theta}} \theta_{dk}$. In this paper, we use the shorthand $\mu_{dk}$ to denote $\hat{\theta}_{dk}$ for a particular document $d$ and topic $k$.

In the rest of this section, we first discuss what types of information can be derived from the topic posterior estimates from different Gibbs samplers. Then, we examine how the corpus-wide variability can be effectively captured in a new metric for topic quality evaluation.

Two types of information can be derived from the topic posterior estimates: (1) the mean of estimates, $\mu_{dk}$, as discussed above, and (2) the variation of estimates. For variation of estimates, we considered using the standard deviation $\sigma_{dk}$. However, this measure is too sensitive to the order-of-magnitude differences of $\mu_{dk}$, that typically occur in different documents. So, in order to capture a

| Method | 20NG | Wiki | NYT | Mean |
|---|---|---|---|---|
| CV (Roder et al., 2015) | 0.129 | 0.385 | 0.248 | 0.254 |
| CP (Roder et al., 2015) | 0.378 | 0.403 | 0.061 | 0.280 |
| DS (Aletras and Stevenson, 2013) | 0.461 | 0.423 | 0.365 | 0.416 |
| NPMI (Lau et al., 2014) | 0.639 | 0.568 | 0.639 | 0.615 |
| PMI (Newman et al., 2010) | 0.602 | 0.550 | 0.623 | 0.591 |
| Coherence (Mimno et al., 2011) | 0.280 | 0.102 | 0.535 | 0.305 |
| Stability (Xing and Paul, 2018) | 0.230 | 0.137 | 0.322 | 0.230 |
| Variability | **0.679** | **0.703** | **0.774** | **0.719** |

Table 2: Pearson's $r$ correlation with human judgments for metrics.

more stable dispersion of estimates from different iterations of Gibbs sampling, we propose to compute the variation of topic $k$'s estimates in document $d$ as its *coefficient of variance (cv)* (Everitt, 2002), which is defined as: $cv_{dk} = \sigma_{dk}/\mu_{dk}$

Notice that both $\mu_{dk}$ and $cv_{dk}$ can arguably help distinguish high and low quality topics because:

- High quality topics will have high $\mu_{dk}$ for related documents and low $\mu_{dk}$ for unrelated documents. But low quality topics will have relatively close $\mu_{dk}$ throughout the corpus.
- High quality topics will have low $cv_{dk}$ for related documents and high $cv_{dk}$ for unrelated ones. But low quality topics will have relatively high $cv_{dk}$ throughout the corpus.

Now, focusing on the design of the new metric, we consider using the corpus-wide variability of topics' estimates as our new metric. Figure 1 shows a comparison of the distributions of mean of estimates ($\mu$) and variation of estimates ($\sigma$, $cv$) for two topics across the NYT corpus (Section 5.1). We can see the $cv$ distributions of good (Topic1) and bad (Topic2) topics are the most different. The $cv$ distribution of Topic1 covers a large span and has a heavy head and tail, while $cv$ values of Topic2 are mostly clustered in a smaller range. In contrast, the difference between Topic1 and Topic2's distributions of $\mu$ and $\sigma$ throughout the corpus appears to be less pronounced. Taking the corpus-wide variability difference between good and bad topics observed in Figure 1, we choose $cv$ to measure the corpus-wide variability of topic $k$'s estimates as our new metric. Formally, it can be defined as:

$$variability(k) = std(cv_{1k}, cv_{2k}, \cdots, cv_{Dk}) \quad (2)$$

where $D$ is the size of the corpus. High quality topics will have higher variability and low quality topics will have lower variability. Table 1 shows a comparison in performance (correlation with human judgment) of our variability defined by $cv$ with the variability defined by $\mu$ or $\sigma$ on three commonly used datasets (Section 5.1). The variability defined by $cv$ is a clear winner.

## 4 Topic Quality Estimator

Our new method, like all other methods driven from the posterior variability, does not use any information from the topic words, which is in contrast the main driver for co-occurrence methods. Based on this observation, posterior variability and co-occurrence methods should be complementary to each other. To test this hypothesis, we investigate if a more reliable estimator of topic quality can be obtained by combining these two classes of methods in a supervised approach. In particular, we train a support vector regression (SVR) estimator (Joachims, 2006) with the estimations of these methods as features, including all the topic quality measures introduced in Section 2 along with our proposed *variability* method.

## 5 Experiments

### 5.1 Datasets

We evaluate our topic quality estimator on three datasets: *20NG*, *Wiki* and *NYT*. *20NG* is the 20 Newsgroup dataset (Joachims, 1997) which consists of 9,347 paragraphs categorized into 20 classes[3]. *Wiki* consists of 10,773 Wikipedia articles written in simple English[4]. *NYT* consists of 8,764 New York Times articles collected from April to July, 2016[5].

[3] http://archive.ics.uci.edu/ml/datasets/twenty+newsgroups

[4] http://simple.wikipedia.org/

[5] https://www.kaggle.com/nzalake52/new-york-times-articles

| Test | Train | | | Mean | Variability |
|---|---|---|---|---|---|
| 20NG | Wiki | NYT | Wiki+NYT | **0.801** | 0.679 |
| | 0.790 | 0.804 | 0.810 | | |
| Wiki | 20NG | NYT | 20NG+NYT | **0.716** | 0.703 |
| | 0.707 | 0.731 | 0.710 | | |
| NYT | 20NG | Wiki | 20NG+Wiki | 0.770 | **0.774** |
| | 0.762 | 0.775 | 0.773 | | |

Table 3: Pearson's $r$ correlation with human judgments for the topic quality estimator.

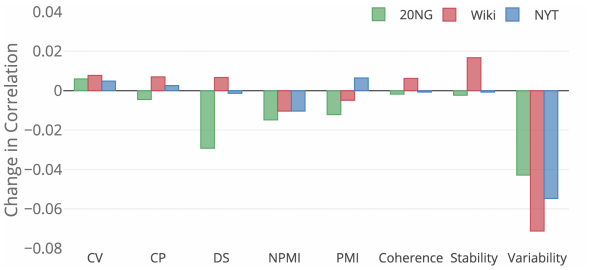


Figure 2: The ablation analysis.

We removed stop words, low-frequency words (appearing less than 3 times), proper nouns and digits from all the datasets, following Chang et al. (2009), so the topic modeling can reveal more general concepts across the corpus.

Following the common setting shared by most of the papers we compared with, for each dataset we built an LDA model which consists of 100 topics represented by the 10 most probable words. The gold-standard annotation for the quality of each topic is the mean of 4-scale human rating scores from five annotators, which were collected through a crowdsourcing platform, Figure Eight[6]. In order to obtain more robust estimates given the variability in human judgments, we removed ratings from annotators who failed in the test trail and recollected those with additional reliable annotators. To verify the validity of the collected annotations, we computed the Weighted Krippendorff's $\alpha$ (Krippendorff, 2007) as the measure of Inter-Annotator Agreement (IAA) for three datasets. The average human rating score/IAA for *20NG*, *Wiki* and *NYT* are 2.91/0.71, 3.23/0.82 and 3.06/0.69, respectively.

### 5.2 Experimental Design

**Topic Modeling** Following the settings in Xing and Paul (2018), we ran the LDA Gibbs samplers for 2,000 iterations (Griffiths and Steyvers, 2004) for each datasets, with 1,000 burn-in iterations, collecting samples every 10 iterations for the final 1,000 iterations. The set of estimates $\mathbf{\Theta}$ thus contains 100 samples.

**Estimator Training** was performed following the cross-domain training strategy (Bhatia et al., 2018). With the ground truth (human judgments), we train the estimator on all topics over one dataset, and test it on another (**one-to-one**). To enlarge the training set, we also train the estimator on two datasets merged together and test it on the third one (**two-to-one**). Given the limited amount of data and the need for interpretability, we experimented only with non-neural classifiers, including linear regression, nearest neighbors regression, Bayesian regression, and Support Vector Regression (SVR) using `sklearn` (Pedregosa et al., 2011); we report the results with SVR, which gave the best performance. We also experimented with different kernels of SVR and rbf kernel worked best.

**Baselines** include seven commonly adopted measures for topic quality assessment: **CV**, **CP** (Roder et al., 2015), **DS** (Aletras and Stevenson, 2013), **PMI** (Newman et al., 2010), **NPMI** (Mimno et al., 2011), **Coherence** (Newman et al., 2010) and **Stability** (Xing and Paul, 2018). All of them are introduced in Section 2.

### 5.3 Results

Following (Roder et al., 2015), we use Pearson's $r$ to evaluate the correlation between the human judgments and the topic quality scores predicted by all the automatic metrics. The higher is the Pearson's $r$, the better the metric is at measuring topic quality. Table 2 shows the Pearson's $r$ correlation with human judgments for all the metrics. Our proposed variability-based metric substantially outperforms all the baselines.

Table 3 shows the Pearson's $r$ correlation with our proposed topic quality estimator trained and tested on different datasets. The average correlations of the estimator dominates our proposed variability-based metric on two out of three datasets, and on one of them by a wide margin.

Additionally, to better investigate how well the metrics align with human judgments, in Figure 3 we use scatter plots to visualize their correlations and make the following observations. The top performer co-occurrence based metric, *NPMI*, tends to underestimate topic quality by giving low ratings to relatively high-quality topics (dots with

[6] https://www.figure-eight.com/

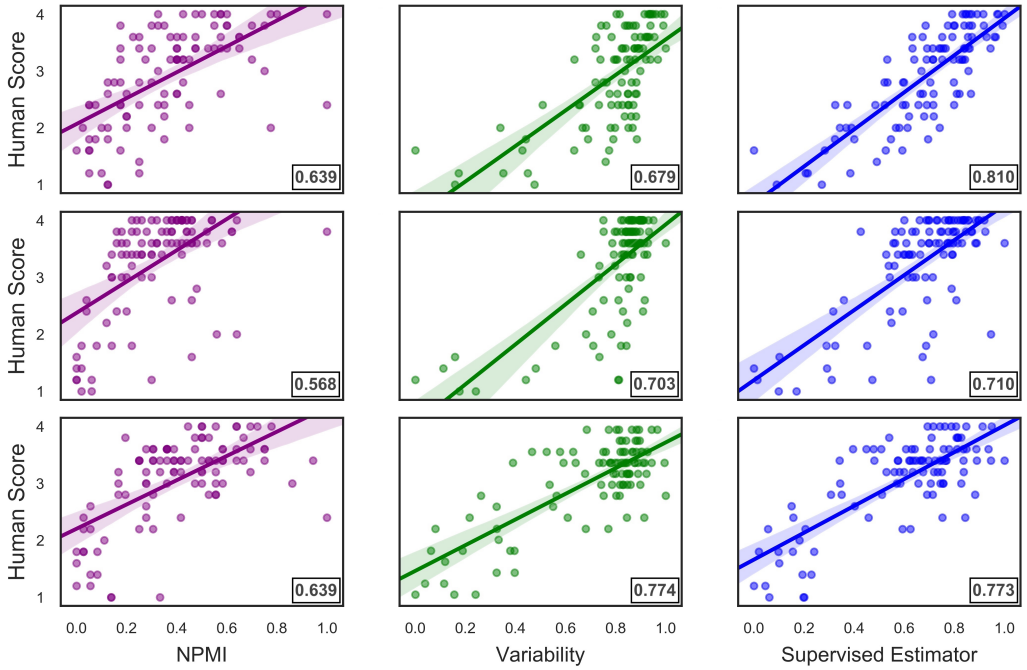


Figure 3: Scatter plots illustrating the correlation between human rating scores and the three metrics: *NPMI*, *Variability* and *Supervised Estimator* on three datasets: *20NG* (top row), *Wiki* (middle row) and *NYT* (bottom row). The numerical Pearson's $r$ correlations are shown in the bottom-right corner.

high human scores tend to be above the purple line), but it performs relatively well for low-quality topics. On the contrary, the top performer posterior based metric, *variability*, is more likely to overestimate topic quality by assigning high ratings to relatively bad topics (dots with low human scores tend to be below the green line), but it performs relatively well for high-quality topics. Thus, when we combine all the metrics in a supervised way, the topic quality estimation becomes more accurate, especially on 20NG corpus (i.e. the top row).

**Ablation Analysis:** Since some features in the topic quality estimator are closely related, their overlap/redundancy may even hurt the model's performance. To better understand the contributions of each feature in our proposed estimator, we conduct ablation analysis whose results are illustrated in Figure 2. We track the change of performance by removing one feature each time. The more significant drop in performance indicates that the removed feature more strongly contributes to the estimator's accuracy. By training on two datasets and testing on the third dataset, we find that only *Variability* and *NPMI* consistently contributes to accurate predictions on all three datasets. This indicates that our new *Variability* metric and *NPMI* are the strongest ones from the two families of Posterior-based and Co-occurrence-based metrics, respectively.

## 6 Conclusion and Future Work

We propose a novel approach to estimate topic quality grounded on the variability of the variance of LDA posterior estimates. We observe that our new metric, driven by Gibbs sampling, is more accurate than previous methods when tested against human topic quality judgment. Additionally, we propose a supervised topic quality estimator that by combining multiple metrics delivers even better results. For future work, we intend to work with larger datasets to investigate neural solutions to combine features from different metrics, as well as to apply our findings to other variants of LDA models trained on low-resource languages, where high-quality external corpora are usually not available (Hao et al., 2018).

## Acknowledgments

We would like to thank the anonymous reviewers for their valuable suggestions and comments, as well as Jey Han Lau and Data Science Group at Universitat Paderborn for making their topic quality evaluation toolkit publicly available.

## References


Nikolaos Aletras and Mark Stevenson. 2013. Evaluating topic coherence using distributional semantics. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013) – Long Papers*, pages 13–22.

Shraey Bhatia, Jey Han Lau, and Timothy Baldwin. 2018. Topic intrusion for automatic topic model evaluation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 844–849.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Gerlof Bouma. 2009. Normalized ( pointwise ) mutual information in collocation extraction. In *Proceedings of the Biennial GSCL Conference'09*, pages 31–40.

Jonathan Chang, Jordan Boyd-Graber, Chong Wang, Sean Gerrish, and David M. Blei. 2009. Reading tea leaves: How humans interpret topic models. In *Proceedings of the Neural Information Processing Systems*, pages 288–296.

Qiuxing Chen, Lixiu Yao, and Jie Yang. 2016. Short text classification based on lda topic model. In *2016 International Conference on Audio, Language and Image Processing (ICALIP)*.

Brian Everitt. 2002. *The Cambridge dictionary of statistics*. Cambridge University Press.

Stuart Geman and Donald Geman. 1984. Stochastic relaxation, Gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 503–512.

Thomas L. Griffiths and Mark Steyvers. 2004. Finding scientific topics. In *Proceedings of the National academy of Sciences*, pages 5228–5235.

David Hall, Daniel Jurafsky, and Christopher D Manning. 2008. Studying the history of ideas using topic models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 363–371.

Shudong Hao, Jordan Boyd-Graber, and Michael J. Paul. 2018. Lessons from the Bible on modern topics: Low-resource multilingual topic model evaluation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1090–1100.

Xiaolei Huang, Xin Li, Tianli Liu, David Chiu, Tingshao Zhu, and Lei Zhang. 2015. Topic model for identifying suicidal ideation in Chinese microblog. In *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation*, pages 553–562.

Xiaolei Huang, Linzi Xing, Jed R. Brubaker, and Michael J. Paul. 2017. Topic model for identifying suicidal ideation in Chinese microblog. In *IEEE International Conference on Healthcare Informatics (ICHI)*, pages 470–477.

Thorsten Joachims. 1997. A probabilistic analysis of the Rocchio algorithm with TF-IDF for text categorization. In *International Conference on Machine Learning (ICML)*, pages 143–151.

Thorsten Joachims. 2006. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 217–226.

Klaus Krippendorff. 2007. Computing krippendorffs alpha reliability. *Departmental papers (ASC) (2007)*, 43.

Jey Han Lau, David Newman, and Timothy Baldwin. 2014. Machine reading tea leaves: Automatically evaluating topic coherence and topic model quality. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 530–539.

David Mimno, Hanna Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. 2011. Optimizing semantic coherence in topic models. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 262–272.

David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. 2010. Automatic evaluation of topic coherence. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 100–108.

Michael J Paul and Mark Dredze. 2014. Discovering health topics in social media using topic models. *PLOS ONE*, page 9(8):e103408.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Margaret E Roberts, Brandon M Stewart, Dustin Tingley, and Edoardo M Airoldi. 2013. The structural topic model and applied social science. In *Advances in Neural Information Processing Systems Workshop on Topic Models: Computation, Application, and Evaluation*.

Michael Roder, Andreas Both, and Alexander Hinneburg. 2015. Exploring the space of topic coherence measures. In *the Eighth ACM International Conference on Web Search and Data Mining*, pages 39–408.

Hanna M. Wallach, David M. Mimno, and Andrew McCallum. 2009. Rethinking lda: Why priors matter. In *Proceedings of the Neural Information Processing Systems*, pages 1973–1981.

Linzi Xing and Michael Paul. 2018. Diagnosing and improving topic models by analyzing posterior variability. In *AAAI Conference on Artificial Intelligence (AAAI-18)*, pages 6005–6012.

# Neural Topic Model with Reinforcement Learning

**Lin Gui**[1,§], **Jia Leng**[2,§], **Gabriele Pergola**[1], **Yu Zhou**[1], **Ruifeng Xu**[2,3,4], **Yulan He**[1,†]
[1]Department of Computer Science, University of Warwick, UK
[2]Harbin Institute of Technology (Shenzhen), China
[3]Peng Cheng Laboratory, Shenzhen, China
[4]Joint Lab of Harbin Institute of Technology and RICOH
Lin.gui@warwick.ac.uk, lengjia@stu.hit.edu.cn
gabriele.pergola@warwick.ac.uk, Yu.Zhou.1@warwick.ac.uk
xuruifeng@hit.edu.cn, yulan.he@warwick.ac.uk

## Abstract

In recent years, advances in neural variational inference have achieved many successes in text processing. Examples include neural topic models which are typically built upon variational autoencoder (VAE) with an objective of minimising the error of reconstructing original documents based on the learned latent topic vectors. However, minimising reconstruction errors does not necessarily lead to high quality topics. In this paper, we borrow the idea of reinforcement learning and incorporate topic coherence measures as reward signals to guide the learning of a VAE-based topic model. Furthermore, our proposed model is able to automatically separating background words dynamically from topic words, thus eliminating the pre-processing step of filtering infrequent and/or top frequent words, typically required for learning traditional topic models. Experimental results on the 20 Newsgroups and the NIPS datasets show superior performance both on perplexity and topic coherence measure compared to state-of-the-art neural topic models.

## 1 Introduction

Probabilistic topic models have been used widely in nature language processing (Li et al., 2016; Zeng et al., 2018). The fundamental principle is that words are assumed to be generated from latent topics which can be inferred from data based on word co-occurrence patterns (Neal, 1993; Andrieu et al., 2003). In recent years, Variational Autoencoder (VAE) has been proved more effective and efficient to approximating deep, complex and underestimated variance in integrals (Kingma and Welling, 2013; He et al., 2017). However, the VAE-based topic models focus on the construction of deep neural networks to approximate the intractable distribution between observed words and latent topics based on log-likelihood and the learning objective is to minimise the error of reconstructing the original documents based on the learned latent topic vectors rather than improving the quality of learned topics, for example, measured by coherence scores (Kingma and Welling, 2013; Sønderby et al., 2016; Miao et al., 2016; Card et al., 2017; Srivastava and Sutton, 2017; Bouchacourt et al., 2018). The lack of consideration of topic coherence measures during the learning process of VAE-based topic models makes it difficult to control the quality of the generated topics. Intuitively, one solution is to jointly consider coherence scores in the learning objective. However, this is not feasible since coherence score is an unsupervised measure of topics based on a large-scale knowledge source, there is no ground truth "best topics".

Another limitation of existing approaches is that they typically require a pre-processing step to filter infrequent and/or top frequent words in order to reduce the vocabulary size and achieve better topic extraction results. Word filtering is often done heuristically. Although there have been attempts to automatically distinguishing background words and topic words, existing approaches either require a switch variable defined at each word position to indicate whether the word is a background word, which makes the models cumbersome, or model each latent topic as the deviation in log-frequency from a constant background distribution (Eisenstein et al., 2011; Smith et al., 2018).

In this paper, we propose a new framework to use reinforcement learning (Pan et al., 2018; Qin et al., 2018; Yin et al., 2018) to incorporate the topic coherence measures into the learning of a neural topic model and filter background words dynamically. More concretely, given an input document, its constituent words will first be sampled

[§]The two authors contributed equally to this work.
[†]Corresponding author.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3478–3483,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

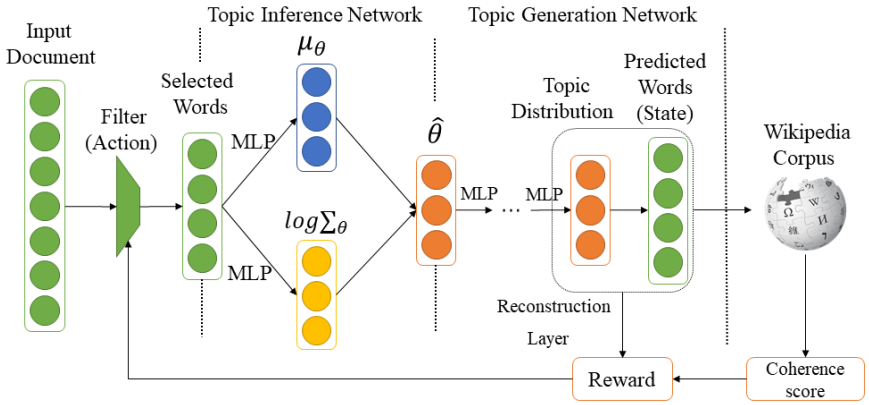


Figure 1: Neural topic model with reinforcement learning.

by a weight vector which assigns higher weights to words with higher coherence scores and have more concentrated topic distributions. The sampled words will then be fed into a VAE-based neural topic model to reconstruct the original document. A reward function is deployed to take into account both topic coherence scores and the degree of word overlapping between topics. The reward signal derived is subsequently used to update the sampling weight vector for each word. In this way, we do not need to directly add the coherence scores into the loss function. Our experimental results show that our proposed framework outperforms the traditional topic model and existing neural topic modelling approaches on the 20 Newsgroups (Lang, 1995) and the NIPS data (Tan et al., 2017) in topic coherence and perplexity.

The rest of the paper is organized as follows. Section 2 presents our proposed reinforcement learning framework for topic modelling. Section 3 reports the experimental setup and results. Section 4 concludes the paper and outlines future research directions.

## 2 Proposed Method

In this section, we introduce our proposed reinforcement learning (RL) framework for topic modelling. A standard RL framework contains three components: action, state, and reward. Here, the action aims to select words with high coherence scores and filter background words. The state is the distribution of latent topics among words, which is obtained from a VAE-based topic model. The reward is a function to measure the quality of topics based on an external corpus and guide the weight updating of the next word selecting action. The overall architecture is illustrated in Figure 1. We detail our framework in the following.

### 2.1 Action

For an input document $d = \{w_1, w_2, ..., w_U\}$, where each word $w_i$ is represented by a one-hot representation, the action is determined by a probabilistic vector $P = \{p_1, p_2, ..., p_U\}$ which is used to filter the less topical-coherent and background words at each iteration of model learning. Here, each $p_i$ present the sampling probability for word $w_i$, and $U$ is the full vocabulary size. We aim to select $V$ words from the full vocabulary based on $P$ and mask out other words in $d$, thus $\tilde{d} = \{w'_1, w'_2, ..., w'_V\}$ where $V \leq U$. The goal of our method is to assign higher probabilities to words which contribute more to the topic coherence scores and lower probabilities to those less topical-coherent words and background words which occur equally likely across topics.

### 2.2 State

After word selection, the new document representation $\tilde{d}$ is fed into a neural topic model to obtain the state, which is the topic distribution in the topic model. Here, we deploy the VAE (Miao et al., 2016; Card et al., 2017) to learn the latent topics, which consists of two main components, the *Inference Network* and the *Generation Network*. For the *Inference Network*, we use VAE to approximate the posterior distribution over topics for all the training instances. In the *Generation Network*, the words are generated via Gaussian softmax construction from the topic distribution generated by the *Inference Network*. The architecture of the neural topic model is shown in Figure (1) and we describe the model in more details below.

**Inference Network**. Following the idea of VAE

which computes a variational approximation to an intractable posterior using MLPs, we define two MLPs, $f_{\mu_\theta}$ and $f_{\Sigma_\theta}$, which takes as input the word counts in a document and outputs mean and variance of a Gaussian distribution, both being vectors in $R^K$, $\mu_\theta = f_{\mu_\theta}(w_d)$, $\Sigma_\theta = \text{diag}(f_{\Sigma_\theta}(w_d))$. Here, 'diag' converts a column vector to a diagonal matrix. For a document $d$, its variational distribution is $q(\theta) \simeq \mathcal{N}(\mu_\theta, \Sigma_\theta)$. With such a formulation, we can generate samples from $q(\theta)$ by first sampling $\epsilon \sim \mathcal{N}(0, I^2)$ and then computing $\hat{\theta} = \sigma(\mu_\theta + {\Sigma_\theta}^{1/2}\epsilon)$.

**Generation Network**. We feed the sampled $\hat{\theta}$ to two MLPs to generate $z_d$. Here, $z_d$ is a $K$-dimensional latent topic representation of document $d$. The probability of $d$-th word in $n$-th document $w_{d,n}$ can be parameterised by another network,

$$p(w_{d,n}|w_d, z_d) \propto \exp(m_d + W \cdot z_d) \quad (1)$$

where $m_d$ is the $V$-dimensional background log-frequency word distribution, $W \in R^{V \times K}$ is a weight matrix. With the sampled $\hat{\theta}$ , for each document $d \in N_d$, we can estimate the Evidence Lower Bound (ELBO) with a Monte Carlo approximation using $L$ independent samples:

$$\begin{aligned}\mathcal{L}_t(w_d) \approx & \frac{1}{L}\sum_{l=1}^{L}\sum_{n=1}^{N_d}\log p(w_{d,n}|\hat{\theta}^{(l)}) \\ & - KL(q(z_d|w_d)||p(z_d))\end{aligned} \quad (2)$$

By minimising the ELBO in Eq. (2), the neural topic model reconstructs the input document $w_d$. At the reconstruction layer, the matrix $W \in R^{V \times K}$ in a single-layer network, which is used to capture the sampling weights between each word and the latent topics, is the state which produces specific topic coherence scores.

### 2.3 Reward

Intuitively, words with higher topic coherence and lower degree of overlapping among different topics should be assigned higher reward in the next iteration of learning. Hence, the reward function should be composed by two terms for each word: the average coherence score and topic overlapping value. The average coherence score is defined as:

$$CO_{average} = (W \cdot C_v(W)) \odot P_{w'_i \in w_d} \quad (3)$$

where matrix $W$ is the distribution of latent topics among words, $C_v(W)$ is a $K$-dimension vector which contains the coherence score for each topic based on the sampling weight matrix, $P_{w'_i \in w_d}$ is the sampling probability for each word in document $d$ (i.e., which action to take as described in Section 2.1), and $\odot$ is the element-wise product. Hence, $CO_{average}$ is a $V$-dimension weight vector to distribute coherence scores to the selected words based on sampling probabilities in action and topic distribution in topic modelling.

The Topic Overlapping ($TO$) is defined as:

$$TO = \text{sum}_{\text{row}}(\text{abs}(I - W \cdot W^T)) \quad (4)$$

where $I$ is a $V \times V$ identity matrix. $TO \in R^{|v|}$ is to measure the separation based on mean distribution. In $TO$, the high value indicates that the associated word appears frequently across topics and hence could be considered as background words.

Based on the average coherence score and the topic overlapping value, the reward function is:

$$R_t = CO_{average} - \alpha \cdot TO \quad (5)$$

$$Q_t = \beta \cdot Q_{t-1} + (1 - \beta) \cdot R_{t-1} \quad (6)$$

where $\alpha$ and $\beta \in (0, 1)$ are trade-off coefficient. Then, the reward at the current time step, $R_t$, and the history rewards encoded by $Q_t$ will be used to update the sampling weight in the action:

$$P_t = \max(P_{t-1} + \lambda_P \cdot (R_t - Q_t), 0 + \epsilon) \quad (7)$$

where $P_t$ is the sampling vector in action, $\epsilon > 0$ is a minimal value, and $\lambda_P$ is the learning rate for $P$. We choose a ramp function with $\epsilon$ to ensure the sampling probability is positive.

### 2.4 Training

For pre-processing, we performed stop word removal* and use Adam to optimise the parameters in the neural networks. The learning rate for VAE and $\theta_p$ are both 0.0001, the mini-batch size is 32, $\alpha$ is 0.1, $\beta$ is 0.5, $\epsilon$ is 0.01, and the coherence scores are obtained from Wikipedia. The parameters in VAE are updated in each mini-batch, and the probabilistic vector $P$ for action selection is updated every 2,000 mini-batches.

## 3 Experiments

We evaluate our model on the 20 Newsgroups† consisting of 18K documents, and NIPS (Tan

*http://mallet.cs.umass.edu/import-stoplist.php

†http://qwone.com/~jason/20Newsgroups/

| | 20News | | NIPS | |
|---|---|---|---|---|
| **Methods** | **PPL** | $C_v$ | **PPL** | $C_v$ |
| **#Topics = 30**, frequency-based vocab. | | | | |
| LDA | 1,213.1 | 0.503 | 1,042.7 | 0.507 |
| NVDM | 980.8 | 0.497 | 931.6 | 0.492 |
| NGTM | 929.3 | 0.479 | 938.9 | 0.503 |
| Scholar | 1,345.9 | 0.537 | 1,350.9 | 0.512 |
| **#Topics = 30**, RL-based vocab. | | | | |
| LDA | 1,451.7 | 0.522 | 1,093.1 | 0.534 |
| NVDM | 845.8 | 0.510 | 768.7 | 0.509 |
| NGTM | **791.5** | 0.517 | 757.2 | 0.527 |
| Scholar | 1,158.4 | 0.560 | 1,273.6 | 0.548 |
| VTMRL | 803.7 | **0.577** | **730.6** | **0.568** |
| **#Topics = 50**, frequency-based vocab. | | | | |
| LDA | 1,015.9 | 0.501 | 995.5 | 0.503 |
| NVDM | 1,014.0 | 0.471 | 927.6 | 0.506 |
| NGTM | 903.5 | 0.491 | 908.8 | 0.498 |
| Scholar | 1,514.5 | 0.521 | 1,373.2 | 0.508 |
| **#Topics = 50**, RL-based vocab. | | | | |
| LDA | 1,251.6 | 0.518 | 921.4 | 0.527 |
| NVDM | 837.9 | 0.502 | 767.0 | 0.514 |
| NGTM | 772.2 | 0.514 | 749.7 | 0.511 |
| Scholar | 1,335.9 | 0.526 | 1,299.8 | 0.530 |
| VTMRL | **725.2** | **0.559** | **712.2** | **0.566** |

Table 1: Perplexity and topic coherence results of difference models. '*frequency-based vocab.*' denotes that the vocabulary is constructed by filtering out rare words while '*RL-based vocab.*' denotes that the vocabulary is dynamically generated by our model using RL.

et al., 2017) consisting of 6.6k documents. We use 10% training data as the validation set to fine-tune the parameters. We compare our results with those obtained from the following baselines:

LDA: Latent Dirichlet Allocation Model (Blei et al., 2003).

NVDM: Neural Variational Document model (Miao et al., 2016).

NGTM: Neural Generative Topic model (Card et al., 2017).

Scholar: Topic model with metadata (Card et al., 2018).

VTMRL: Our proposed Variational Topic Model with Reinforcement Learning.

For all the baseline models, we follow the common pre-processing step in existing approaches by performing stop word removal, and selecting the most frequent 2,000 words as the vocabulary. Since our proposed framework dynamically select words at each iteration of learning, we do not need to pre-set the vocabulary prior to model learning. Instead, we only activate 2,000 words at each mini-batch of training based on the word sampling probabilities. As our model dynamically select words during the training process, in order to ensure fair comparison with other models, we also report the results of training baselines using the vocabulary dynamically generated by our model.

In our experiments, the models are evaluated based on the perplexity (PPL, lower is better) and topic coherence measure ($C_v$) based on external corpus (Röder et al., 2015) (higher is better). The results with 30 and 50 topics are shown in Table 1.

LDA is a conventional topic model, while all the other models are neural topic models. It can be observed from Table 1 that NVDM and NGTM achieve better perplexities compared to LDA. However, in terms of topic coherence measure, NVDM and NGTM perform slightly worse than LDA. A similar observation has been reported in (Card et al., 2017). Scholar achieves better coherence compared to other neural models. Nevertheless, after using reinforcement learning based on the topic coherence scores in our proposed model, VTMRL outperforms all the other models on the topic coherence measure by a large margin. RL could activate words which are semantically related to topics regardless of their occurrence frequency. The inclusion of some rare words would impact the models' predictive probabilities. As such, we observe worse perplexity results for models trained with RL-based vocabulary compared to frequency-based vocabulary in 20 Newsgroups, though the converse is true for NIPS. Nevertheless, the coherence scores improve for all the models with RL-based vocabulary.

As incorporating RL could increase the computational complexity of VTMRL, we report in Table 2 the total number of parameters and average training time per epoch when the vocabulary size is 2,000 and the number of topics is 50.

Strictly speaking, the number of parameters in LDA is not directly comparable with neural models. Neural models have similar parameter size. With the incorporation of RL, VTMRL only increased the parameter size by 1.4%. Due to the efficiency of GPU, the running costs of neural models are better than that of LDA. Although our proposed VTMRL used full vocabulary, the active words in each epoch are limited. Hence, there is no significant increase in terms of the running cost.

| Model | #parameters | Training time(s) |
|---|---|---|
| LDA | $1.00 \times 10^5$ | 660.89 |
| NVDM | $1.38 \times 10^6$ | 62.16 |
| NGTM | $1.39 \times 10^6$ | 72.12 |
| Scholar | $1.15 \times 10^6$ | 3.41 |
| VTMRL | $1.41 \times 10^6$ | 5.78 |

Table 2: Number of parameters for each model and the average training time per epoch with vocabulary size 2,000 and topic number 50.

| Topic | Topic Words |
|---|---|
| **Without RL** | |
| 1 | university <NUM> subject host idea |
| 2 | organization article writes surrender lines |
| 3 | people organization posting article lines |
| **With RL** | |
| 1 | mouse x11r5 keyboard serial remote |
| 2 | chip design products build system |
| 3 | drives friend sports espn michigan |

Table 3: Example topic words with/without RL by VTMRL (#Topics = 50) in 20 Newsgroup.

| Model | Acc | Model | Acc |
|---|---|---|---|
| LDA | 0.412 | tf-idf + LDA | 0.822 |
| NGTM | 0.375 | tf-idf + NGTM | 0.825 |
| NVDM | 0.303 | tf-idf + NVDM | 0.824 |
| VTMRL | 0.478 | tf-idf + VTMRL | **0.830** |
| CNN | 0.515 | RNN | 0.509 |

Table 4: Text classification accuracy of different models on the 20 Newsgroups data.

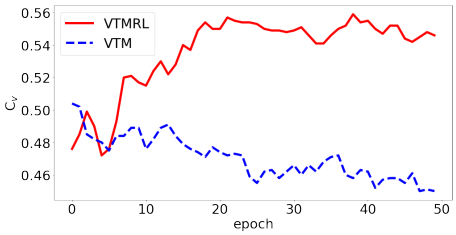


Figure 2: Topic coherence changes with/without RL (#Topics = 50).

We show in Table 3 example topics with/without RL by VTMRL in 20 Newsgroups. The RL method seems producing more interpretable topics. Also, due to the reward-based words sampling in RL, words with low occurrence frequency would still have a chance to be promoted in specific topics, such as 'x11r5', which is a serial number of the Windows system.

We next compare the topic coherence changes during model training. We observe that for VTMRL the coherence value increases at the beginning of the training and remains relatively stable in subsequent training iterations. As a contrast, the coherence value of our model without RL is not stable, and decreases rapidly after 10 training epochs. This is not surprising since the model without RL did not consider topic coherence in its learning process.

We also evaluate the effectiveness of using the learned topics as features to train text classifiers on the 20 Newsgroups data. The results are obtained by using logistic regression as the classifier trained from the topics generated by various aforementioned models. We also report the results by training logistic regression from the combination of word features (tf-idf) and topic features (#topic = 30). In addition, we include the results using neural models such as CNN and RNN in Table 4.

Using only topics extracted from topic models as features to train logistic regression, our proposed model VTMRL beats other baselines. However, the topic features have only 30 dimensions so the performance is limited in comparison with CNN and RNN. When we combine the topic features with tf-idf based word features, the performance is boosted significantly compared to CNN and RNN and the best result is obtained by using the logistic regression model trained from the combined word features with topics generated by our proposed VTMRL.

## 4 Conclusion

In this paper, we have proposed a new reinforcement learning (RL) framework for neural topic modelling, where words are activated dynamically by RL according to topic coherence scores and topic overlapping values. The experiments on the 20 Newsgroups and NIPS datasets show encouraging results both on perplexity and topic coherence measures in comparison with existing neural topic models. In future work, we will explore extending our model for temporal topic modelling.

## Acknowledgment

This work was supported by the Innovate UK (grant no. 103652), the EU-H2020 (grant no. 794196), the National Natural Science Foundation of China U1636103, 61876053.

## References


Christophe Andrieu, Nando de Freitas, Arnaud Doucet, and Michael I. Jordan. 2003. An introduction to MCMC for machine learning. *Machine Learning*, 50(1-2):5–43.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Diane Bouchacourt, Ryota Tomioka, and Sebastian Nowozin. 2018. Multi-level variational autoencoder: Learning disentangled representations from grouped observations. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*.

Dallas Card, Chenhao Tan, and Noah A. Smith. 2017. A neural framework for generalized topic models. *CoRR*, abs/1705.09296.

Dallas Card, Chenhao Tan, and Noah A. Smith. 2018. Neural models for documents with metadata. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 2031–2040.

Jacob Eisenstein, Amr Ahmed, and Eric P Xing. 2011. Sparse additive generative models of text. In *Proceedings of the 28th International conference on Machine Learning (ICML)*.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2017. An unsupervised neural attention model for aspect extraction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 388–397. Association for Computational Linguistics.

Diederik P. Kingma and Max Welling. 2013. Auto-encoding variational bayes. *CoRR*, abs/1312.6114.

Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339.

Jing Li, Ming Liao, Wei Gao, Yulan He, and Kam-Fai Wong. 2016. Topic extraction from microblog posts using conversation structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 1727–1736.

Radford M Neal. 1993. Probabilistic inference using markov chain monte carlo methods.

Boyuan Pan, Yazheng Yang, Zhou Zhao, Yueting Zhuang, Deng Cai, and Xiaofei He. 2018. Discourse marker augmented network with reinforcement learning for natural language inference. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 989–999. Association for Computational Linguistics.

Pengda Qin, Weiran XU, and William Yang Wang. 2018. Robust distant supervision relation extraction via deep reinforcement learning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2137–2147. Association for Computational Linguistics.

Michael Röder, Andreas Both, and Alexander Hinneburg. 2015. Exploring the space of topic coherence measures. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining, WSDM 2015, Shanghai, China, February 2-6, 2015*, pages 399–408.

Noah A. Smith, Dallas Card, and Chenhao Tan. 2018. Neural models for documents with metadata. In *ACL*.

Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. 2016. Ladder variational autoencoders. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3738–3746.

Akash Srivastava and Charles Sutton. 2017. Autoencoding variational inference for topic models. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*.

Chenhao Tan, Dallas Card, and Noah A. Smith. 2017. Friendships, rivalries, and trysts: Characterizing relations between ideas in texts. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 773–783.

Qingyu Yin, Yu Zhang, Wei-Nan Zhang, Ting Liu, and William Yang Wang. 2018. Deep reinforcement learning for chinese zero pronoun resolution. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 569–578. Association for Computational Linguistics.

Xingshan Zeng, Jing Li, Lu Wang, Nicholas Beauchamp, Sarah Shugars, and Kam-Fai Wong. 2018. Microblog conversation recommendation via joint modeling of topics and discourse. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 375–385. Association for Computational Linguistics.

# Modelling Stopping Criteria for Search Results using Poisson Processes

**Alison Sneyd** and **Mark Stevenson**
Natural Language Processing Group
Department of Computer Science
University of Sheffield
a.sneyd,mark.stevenson@sheffield.ac.uk

## Abstract

Text retrieval systems often return large sets of documents, particularly when applied to large collections. Stopping criteria can reduce the number of these documents that need to be manually evaluated for relevance by predicting when a suitable level of recall has been achieved. In this work, a novel method for determining a stopping criterion is proposed that models the rate at which relevant documents occur using a Poisson process. This method allows a user to specify both a minimum desired level of recall to achieve and a desired probability of having achieved it. We evaluate our method on a public dataset and compare it with previous techniques for determining stopping criteria.

## 1 Introduction

Document retrieval systems are widely used in information retrieval (IR) to return a set of documents in response to a user query. They often return large numbers of documents, particularly for large collections, which places a significant burden on a user who has to manually review them to determine which are relevant (i.e. of interest). The problem is particularly acute in scenarios where relevant documents are rare in the set of retrieved documents, or where there is a high cost associated with missing a relevant document.

Systematic reviews of medical literature are a good example of such a task. They aim to identify and summarise all available evidence for a specific research question such as 'Is systemic inflammation present in stable chronic obstructive pulmonary disease?'(Gan et al., 2004). Evidence included in systematic reviews is typically identified by executing a Boolean query optimised for high recall, with the results manually reviewed by domain experts. However, this process is extremely expensive and a single review may cost up to a quarter of a million dollars and take over a year to complete (McGowan and Sampson, 2005; Karimi et al., 2010). A range of studies have demonstrated that ranking the results of the Boolean query can support the identification of relevant studies by placing them higher in the ranking, e.g. (Miwa et al., 2014; Wallace et al., 2010; O'Mara-Eves et al., 2015; Shemilt et al., 2016). However, ranking alone does not reduce the manual review workload since it does not inform reviewers about when they can stop looking at search results.

Search stopping criteria are methods for determining when manual examination of search results can stop. They essentially provide information about the risk of missing relevant documents that is associated with not examining some portion of the documents returned by a query. Stopping criteria have been proposed for technology assisted reviews (Cormack and Grossman, 2016) and IR test collection development (Losada et al., 2019).

Cormack and Grossman (2016) introduced the target and knee stopping methods. The former involves examining randomly chosen documents until a specified number of relevant documents have been found. The number of relevant documents required is determined using statistical inference and guarantees a minimum recall score with a specified probability. The knee method employs a knee detection algorithm (Satopaa et al., 2011) to estimate when the gain curve of recall versus effort has exceeded a certain tipping point, called the slope ratio. Cormack and Grossman (2016) found that this method was better able to predict the most efficient stopping point than the target method, but did not provide any probabilistic guarantees about the level of recall that would be achieved. Additionally, Losada et al. (2019) proposed simple baselines and used power curves to determine stopping criteria. A significant limi-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3484–3489,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

tation of previous approaches is that they are evaluated using a single ranking algorithm, but for a stopping approach to be generally useful it must be reliable for a range of ranking algorithms.

This paper proposes a novel method for generating search stopping criteria based on an inhomogeneous Poisson process. The method assumes the rate at which relevant documents occur in the ranking decreases exponentially and predicts how many relevant documents need to be found in order to achieve a defined level of recall. This approach, along with the knee and target methods, is evaluated using data from a shared task for which over 30 submissions are available. This allows approaches to be evaluated against a wide range of rankings of varying effectiveness. We found that, while each method returns a reliably high level of recall, the proposed method produces the most accurate predictions while also being robust to ranking quality.[1]

## 2 Method

Our proposed method models the rate at which relevant documents occur in a ranked set of retrieved documents, using an exponential function learned from a sample of the data. It is then possible to predict the total number of relevant documents to a desired probability using a Poisson process, and from this, to predict the rank at which a desired level of recall is achieved.

### 2.1 Poisson Process

Poisson processes model the occurrence of random events that take place at a given rate (Kingman, 1993). It is assumed that the events are independent and that the number of occurrences in an interval follows a Poisson distribution. An inhomogeneous (or non-homogeneous) Poisson process has a variable rate function, $\lambda$, where the rate at which events occur is a function of the space over which the process is defined. Let:

$$\Lambda(a,b) = \int_a^b \lambda(x)dx, \tag{1}$$

and let $N(a,b)$ denote the number of events occurring in the interval $(a,b]$. Then:

$$P\left(N(a,b)=r\right) = \frac{[\Lambda(a,b)]^r}{r!}e^{-\Lambda(a,b)}. \tag{2}$$

[1]Code available at: https://github.com/alisonsneyd/poisson_stopping_method

In particular, $N$ is a Poisson random variable with expected value $\Lambda(a,b)$.

We model the occurrence of relevant documents in a set of ranked, retrieved documents as an inhomogeneous Poisson process, where $\lambda$ is an exponential function (i.e. $\lambda(x) = de^{kx}$). So, from Equation 1:

$$\begin{aligned}\Lambda(a,b) &= \int_a^b de^{kx}dx \\ &= \frac{d}{k}\left(e^{kb} - e^{ka}\right).\end{aligned}$$

We are interested in knowing the total number of relevant documents and so restrict our attention to the interval $(0,n]$, where $n$ is the number of documents in the set, yielding:

$$\Lambda(0,n) = \frac{d}{k}(e^{kn} - 1). \tag{3}$$

By Equations 2 and 3, the probability of there being $r$ relevant documents in the document set is:

$$P\left(N(0,n)=r\right) = \frac{\left[\frac{d}{k}(e^{kn}-1)\right]^r}{r!}e^{-\left(\frac{d}{k}(e^{kn}-1)\right)}.$$

Since $N(0,n)$ is a Poisson random variable, the upper limit of the number of relevant documents in the collection ($R$), subject to a given probability, can be estimated using the cumulative distribution function of $N(0,n)$. The number of relevant documents required to achieve a desired level of recall ($T$) is then $RT$. The proposed stopping criteria is to cease relevance judgments when $\lceil RT \rceil$ relevant documents have been found. This method is flexible and can be adapted to whatever level of recall a user desires, and to the level of certainty they want associated with it.

### 2.2 Rate Estimation

The approach proceeds by examining the first $\alpha$ documents in the ranking, and then predicting the total number of relevant documents in the entire ranking. If the number of relevant documents observed in the first $\alpha$ documents ($rel(\alpha)$) is greater than or equal to the number of relevant documents needed to achieve the desired level of recall (i.e. $rel(\alpha) \geq \lceil RT \rceil$), the process stops and no more documents are examined. Otherwise, the next $\beta$ documents are added to the pool of examined documents and the process repeated.

Accurate estimation of the rate at which relevant documents occur in the ranking is an impor-

tant component of the proposed approach. The parameters of the rate function are estimated as follows: Split the documents which have been examined so far into $m$ sub-intervals and define a set of points $(x_i, y_i)$, $i = 1, \ldots, m$, where $x_i$ is the midpoint of the $i^{th}$ interval and $y_i$ is the number of relevant documents it contains. Fit a curve of the form $de^{kx}$, $d, k \in \mathbb{R}$, to the points.

This approach assumes that relevant documents occur frequently enough for their rate to be predicted and that this rate decreases approximately exponentially, thereby implying the shape of the gain curve (number of relevant documents versus rank) will be roughly convex. While these assumptions frequently hold, they are not guaranteed to do so and they are therefore tested in multiple ways.

Firstly, if fewer than a given threshold, $\gamma$, of relevant documents are found in the initial sample of $\alpha$ documents, it is assumed that either relevant documents are too rare for accurate estimates or that the ranking places them later, meaning their rate is not exponential. No stopping point is predicted and all documents in the ranking are examined (while this is costly in terms of document judgment, it preserves recall and is the default approach in systematic reviews).

Secondly, the Poisson process is dependent on $\lambda(x)$ reliably estimating the rate at which relevant documents occur. Therefore when a curve is fitted, its accuracy is tested as follows: Suppose the sample ends at rank $n$. The number of relevant documents estimated by $\lambda(x)$ at rank $n$, given by $\sum_{i=1}^{n} \lambda(i)$, is compared to the actual number of relevant documents in the sample ($rel(n)$). If the number of relevant documents found is less than a scalar multiple ($\delta$) of the predicted number (i.e. if $rel(n) < \delta \times \sum_{i=1}^{n} \lambda(i)$) then the curve is assumed to be a bad predictor. No stopping point is predicted and the next $\beta$ documents in the ranking are examined.

## 3 Experiments

### 3.1 Data

The dataset used for the experiments originated from the CLEF 2017 e-Health Lab Task 2 "Technology Assisted Reviews in Empirical Medicine" (Kanoulas et al., 2017). The task required participants to identify relevant studies for a set of 30 Diagnostic Test Accuracy systematic reviews (known as "topics") from the Cochrane Library.[2] Participants were provided with the (unordered) set of documents retrieved by a complex Boolean query and asked to rank them based on relevance to the review topic.

The number of documents retrieved by the Boolean queries for the 30 topics ranged from 64 to 12,807, with median 2,070, while the total number of documents across all topics was 117,562. The number of relevant documents ranged from 2 to 460, with median 38. On average, only 1.58% of documents were relevant across the 30 topics.[3]

A total of 33 runs ranking the entire set of articles were submitted by 11 groups of participants.[4] The rankings produced by these runs were based on a wide range of techniques, including manual judgments of document relevance, and automated techniques based on (semi-)supervised classifiers and relevance feedback. These runs contain a diverse set of rankings of varying effectiveness, making them very suitable for testing the robustness of search stopping criteria.

### 3.2 Evaluation Metrics

The goal is to reduce the amount of manual judgment required, subject to maintaining a minimum level of recall. Following Cormack and Grossman (2016), the following metrics were used: For a stopping criteria $S$ and topic $T$, the *effort* of $S$ in $T$ is defined as $|E_T|$, where $E_T$ is the set of documents that were examined before stopping. For a desired level of recall $R$, the *acceptability* of $S$ with respect to $T$ is defined as:

$$acceptability(S_T) = \begin{cases} 1, & \text{if } recall(S_T) \geq R, \\ 0, & \text{else.} \end{cases}$$

As sets of runs of topics are being evaluated, additional metrics are used. For a stopping criteria $S$ considered over a set of topics in a run $\mathcal{R}$, the *reliability* of $S$ with respect to $\mathcal{R}$ is:

$$\frac{|\{T \in \mathcal{R} : acceptability(S_T) = 1\}|}{|\mathcal{R}|}.$$

$S$ is considered *reliable* over a set of runs if its mean reliability over the runs is not less than a desired threshold $p$, which is equivalent to the proportion of all topics in all runs that are acceptable

[2] https://www.cochranelibrary.com/

[3] The CLEF data includes relevance judgments at two levels: abstract and content. The abstract level information is used for the experiments described in this paper.

[4] Available at https://github.com/CLEF-TAR

being greater than or equal to $p$. The *effort* of $S$ with respect to $\mathcal{R}$ is $\sum_{T \in \mathcal{R}} |E_T|$. The *percentage of effort saved* by $S$ over $R$ will denote the percentage of all documents in the set that do not have to be examined:

$$100 \times \sum_{T \in \mathcal{R}} \frac{|T| - |E_T|}{|T|}.$$

### 3.3 Approaches

The Poisson process approach (Section 2) and knee and target methods (Section 1) were implemented. For all methods, the minimum level of recall was set to 0.7 and the reliability threshold to 0.95, following Cormack and Grossman (2016).

For the Poisson process (**PP**), the initial sample size ($\alpha$) and increases in batches ($\beta$) were set to $0.3|T|$ and $0.05|T|$ respectively. The value of $\alpha$ was chosen as it is large enough to predict the rate curve with reasonable reliability but without requiring too many documents to be examined. The $\gamma$ and $\delta$ parameters were set to 20 and the 0.7 respectively.

For the target method (**TM**), we follow Cormack and Grossman (2016) in requiring that a set of 10 relevant documents be identified before the stopping point is predicted since they demonstrated that this produced a minimum recall of 0.7 with 0.95 probability.

The knee method requires a slope ratio to be set given by $\epsilon + 6 - min(relret, \epsilon)$, where $relret$ is the number of relevant documents retrieved for a given rank and $\epsilon$ is a variable that prevents the method stopping too soon. A version of the approach (**KM-default**) was implemented using $\epsilon = 150$, as recommended by Cormack and Grossman (2016). The value of $\epsilon$ was also tuned using 3-fold cross-validation with $\epsilon \in \{0, 25, 50, 100, 150, 200\}$. The smallest reliable value for $\epsilon$ was found to be 50 in each fold and consequently a second version (**KM-tuned**) was implemented using this value.

For comparison, an Oracle (**OR**) method was also implemented which stops at the point in the ranking where 70% recall has been achieved.

## 4 Results

Each method was found to be reliable (i.e. achieved the desired recall of 0.7 for at least 95% of topics). The top part of Table 1 shows the results over all topics in all 33 runs. The proposed approach, PP, required less effort than other methods (TM, KM-tuned and KM-default). PP allows the desired recall to be achieved without examining 42.1% of the documents, thereby allowing 4.9% more of the documents to go unexamined than the next most efficient approach (KM-tuned).

Further analysis was carried out to explore the effect of the quality of the initial ranking on the performance of the various approaches. The 33 runs were ranked based on their Area Under Recall Curve (AURC) scores in the official evaluation (Kanoulas et al., 2017). (The AURC metric computes the area under the cumulative recall curve normalized by the optimal area.) The top, middle and bottom five runs in the rankings were selected. The top five had AURC scores in the range 0.92 - 0.93, indicating effective rankings in which most of the relevant documents appear early. The bottom five runs had scores in the range 0.48 - 0.60, indicating the rankings are not much different from random orderings of the documents.

PP was the most efficient in terms of effort for the middle and bottom five runs, and the second most efficient for the top five runs. KM-tuned required the least effort for the top five runs, but the second most for the middle five runs and the third most for the bottom five runs. As might be expected, each method required the least effort for the top five runs and the most effort for the bottom five.

A graph of the OR and PP effort scores for each run is shown in Figure 1.

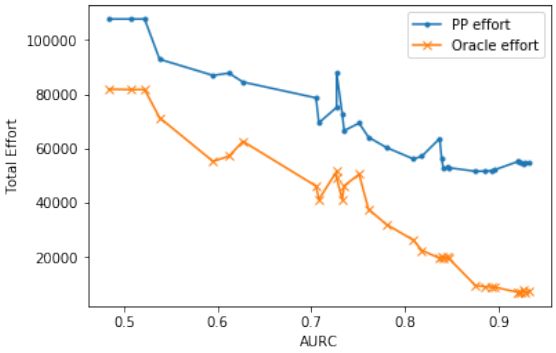


Figure 1: Oracle and Poisson process effort versus AURC score.

The difference in performance between the tuned and untuned version of the knee method (KM-tuned and KM-default) demonstrates the sensitivity of this method to appropriate setting of parameter values. These results demonstrate that

| | Mean Eff. | Mean % Eff. Saved |
|---|---|---|
| All 33 runs | | |
| TM | 79,111 | 32.7% |
| KM-default | 102,681 | 12.7% |
| KM-tuned | 73,826 | 37.2% |
| PP | **68,122** | **42.1%** |
| OR | 33,760 | 71.3% |
| Top 5 runs | | |
| TM | 56,762 | 51.7% |
| KM-default | 87,886 | 25.2% |
| KM-tuned | **33,841** | **71.2%** |
| PP | 54,792 | 53.4% |
| OR | 7,202 | 93.9% |
| Middle 5 runs | | |
| TM | 76,704 | 34.8% |
| KM-default | 107,703 | 8.4% |
| KM-tuned | 75,888 | 35.4% |
| PP | **60,311** | **48.7%** |
| OR | 27,565 | 76.6% |
| Bottom 5 runs | | |
| TM | 104,620 | 11.0% |
| KM-default | 111,055 | 5.5% |
| KM-tuned | 110,778 | 5.8% |
| PP | **100,622** | **14.4%** |
| OR | 74,389 | 36.7% |

Table 1: Mean effort and mean percentage of effort saved for the 33 CLEF 2017 runs. Results are shown for all runs plus the top, middle and bottom five runs. Highlighted figures are the best non-Oracle results.

the approach performs strongly for effective rankings, such as the one it was developed for (Cormack and Grossman, 2015), but is not robust to less effective ones. The PP approach is less sensitive to ranking quality (which is unlikely to be known a priori).

Two examples of the actual gain curves of a ranking and the gain curve estimated by the rate function $\lambda(x)$ learned by the PP approach are shown in Figure 2. In the top graph, the rate at which relevant documents occur decreases exponentially and the high similarity between the two curves shows $\lambda$ is fit well. In the bottom graph, the rate at which relevant documents occur does not decrease exponentially. However, PP still returns the desired level of recall because the probability of the Poisson process was set to a high value (0.95).

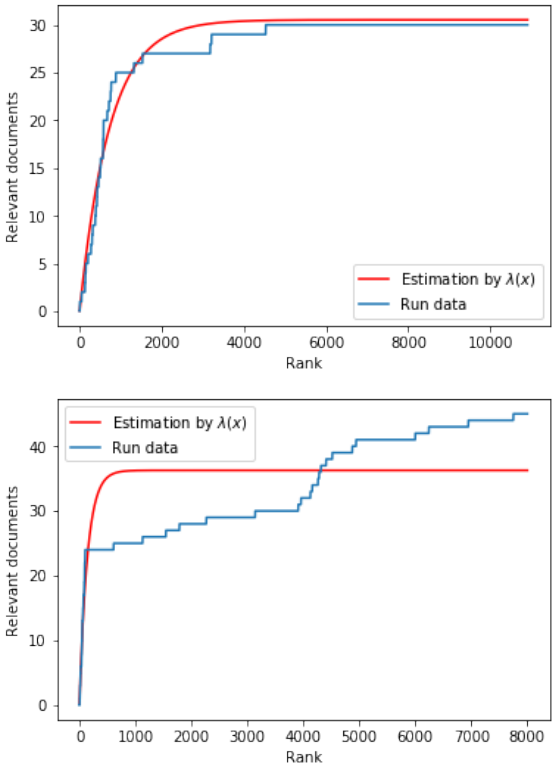


Figure 2: The gain curves of two rankings from CLEF 2017 runs, compared to the gain curve estimated by $\lambda$.

## 5 Conclusion

This work proposed a novel method for determining a stopping point in search results using a Poisson process. The method was applied to 33 varied CLEF 2017 runs and proved reliable in terms of recall, while also requiring substantially less effort and demonstrating less sensitivity to ranking quality than existing methods. On average, the Poisson process saved manual judgment of over 40% of the documents associated with a run. Although our method has been explored in the context of systematic reviews, it could be applied to a variety of other document retrieval problems.

## References

Gordon V. Cormack and Maura R. Grossman. 2015. Autonomy and reliability of continuous active

learning for technology-assisted review. *CoRR*, abs/1504.06868.

Gordon V. Cormack and Maura R. Grossman. 2016. Engineering quality and reliability in technology-assisted review. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, pages 75–84, New York, NY, USA. ACM.

W Qi Gan, SFP Man, A Senthilselvan, and DD Sin. 2004. Association between chronic obstructive pulmonary disease and systemic inflammation: a systematic review and a meta-analysis. *Thorax*, 59(7):574–580.

Evangelos Kanoulas, Dan Li, Leif Azzopardi, and Rene Spijker. 2017. CLEF technologically assisted reviews in empirical medicine overview. In *Working Notes of CLEF 2017 - Conference and Labs of the Evaluation forum*, CEUR Workshop Proceedings, Dublin, Ireland. CEUR-WS.org.

Sarvnaz Karimi, Stefan Pohl, Falk Scholer, Lawrence Cavedon, and Justin Zobel. 2010. Boolean versus ranked querying for biomedical systematic reviews. In *BMC Med. Inf. & Decision Making*.

J F C Kingman. 1993. *Poisson Processes*. Oxford studies in probability; 3. Oxford : Clarendon Press.

David E. Losada, Javier Parapar, and Alvaro Barreiro. 2019. When to stop making relevance judgments? a study of stopping methods for building information retrieval test collections. *Journal of the Association for Information Science and Technology*, 70(1):49–60.

Jessie L McGowan and Margaret Sampson. 2005. Systematic reviews need systematic searchers. *Journal of the Medical Library Association : JMLA*, 93 1:74–80.

Makoto Miwa, James Thomas, Alison O'Mara-Eves, and Sophia Ananiadou. 2014. Reducing systematic review workload through certainty-based screening. *Journal of Biomedical Informatics*, 51:242–253.

Alison O'Mara-Eves, James Thomas, John McNaught, Makoto Miwa, and Sophia Ananiadou. 2015. Using text mining for study identification in systematic reviews: a systematic review of current approaches. *Systematic reviews*, 4(1):5.

Ville Satopaa, Jeannie Albrecht, David Irwin, and Barath Raghavan. 2011. Finding a "kneedle" in a haystack: Detecting knee points in system behavior. In *Proceedings of the 2011 31st International Conference on Distributed Computing Systems Workshops*, ICDCSW '11, pages 166–171, Washington, DC, USA. IEEE Computer Society.

Ian Shemilt, Nada Khan, Sophie Park, and James Thomas. 2016. Use of cost-effectiveness analysis to compare the efficiency of study identification methods in systematic reviews. *Systematic Reviews*, 5.

Byron C Wallace, Thomas A Trikalinos, Joseph Lau, Carla E Brodley, and Christopher H Schmid. 2010. Semi-automated screening of biomedical citations for systematic reviews. *BMC Bioinformatics*, 11(1):55.

# Cross-Domain Modeling of Sentence-Level Evidence for Document Retrieval

**Zeynep Akkalyoncu Yilmaz, Wei Yang, Haotian Zhang,** and **Jimmy Lin**
David R. Cheriton School of Computer Science
University of Waterloo

## Abstract

This paper applies BERT to *ad hoc* document retrieval on news articles, which requires addressing two challenges: relevance judgments in existing test collections are typically provided only at the document level, and documents often exceed the length that BERT was designed to handle. Our solution is to aggregate sentence-level evidence to rank documents. Furthermore, we are able to leverage passage-level relevance judgments *fortuitously* available in other domains to fine-tune BERT models that are able to capture cross-domain notions of relevance, and can be directly used for ranking news articles. Our simple neural ranking models achieve state-of-the-art effectiveness on three standard test collections.

## 1 Introduction

The dominant approach to *ad hoc* document retrieval using neural networks today is to deploy the neural model as a reranker over an initial list of candidate documents retrieved using a standard bag-of-words term-matching technique. Despite the plethora of neural models that have been proposed for document ranking (Mitra and Craswell, 2019), there has recently been some skepticism about whether they have truly advanced the state of the art (Lin, 2018), at least in the absence of large amounts of behavioral log data only available to search engine companies.

In a meta-analysis of over 100 papers that report results on the dataset from the Robust Track at TREC 2004 (Robust04), Yang et al. (2019a) found that most neural approaches do not compare against competitive baselines. To provide two recent examples, McDonald et al. (2018) report a best AP score of 0.272 and Li et al. (2018) 0.290, compared to a simple bag-of-words query expansion baseline that achieves 0.299 (Lin, 2018). Further experiments by Yang et al. (2019a) achieve 0.315 under more rigorous experimental conditions with a neural ranking model, but this is still pretty far from the best-known score of 0.3686 on this dataset (Cormack et al., 2009).

Although Sculley et al. (2018) remind us that the goal of science is not *wins*, but knowledge, the latter requires first establishing strong baselines that accurately quantify proposed contributions. Comparisons to weak baselines that inflate the merits of an approach are not new problems in information retrieval (Armstrong et al., 2009), and researchers have in fact observed similar issues in the recommender systems literature as well (Rendle et al., 2019; Dacrema et al., 2019).

Having placed evaluation on more solid footing with respect to well-tuned baselines by building on previous work, this paper examines how we might make neural approaches "work" for document retrieval. One promising recent innovation is models that exploit massive pre-training (Peters et al., 2018; Radford et al., 2018), leading to BERT (Devlin et al., 2019) as the most popular example today. Researchers have applied BERT to a broad range of NLP tasks with impressive gains: most relevant to our document ranking task, these include BERTserini (Yang et al., 2019b) for question answering and Nogueira and Cho (2019) for passage reranking.

Extending our own previous work (Yang et al., 2019c), the main contribution of this paper is a successful application of BERT to yield large improvements in *ad hoc* document retrieval. We introduce two simple yet effective innovations: First, we focus on integrating *sentence-level evidence* for document ranking to address the fact that BERT was not designed for processing long spans of text. Second, we show, quite surprisingly, that it is possible to transfer models of relevance across different domains, which nicely solves the problem of the lack of passage-level relevance anno-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3490–3496,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

tations. Combining these two innovations allows us to achieve 0.3697 AP on Robust04, which is the highest reported score that we are aware of (neural or otherwise). We establish state-of-the-art effectiveness on two more recent test collections, Core17 and Core18, as well.

## 2 Background and Approach

To be clear, our focus is on neural ranking models for *ad hoc* document retrieval, over corpora comprising news articles. Formally, in response to a user query $Q$, the system's task is to produce a ranking of documents from a corpus that maximizes some ranking metric—in our case, average precision (AP). We emphasize that this problem is quite different from web search, where there is no doubt that large amounts of behavioral log data, along with other signals such as the web-graph, have led to large improvements in search quality (Mitra and Craswell, 2019). Instead, we are interested in limited data conditions—what can be achieved with modest resources outside web search engine companies such as Google and Microsoft who have large annotation teams—and hence we only consider "classic" TREC newswire test collections.

Beyond our own previous work (Yang et al., 2019c), which to our knowledge is the first reported application of BERT to document retrieval, there have been several other proposed approaches, including MacAvaney et al. (2019) and unrefereed manuscripts (Qiao et al., 2019; Padigela et al., 2019). There are two challenges to applying BERT to document retrieval: First, BERT has mostly been applied to sentence-level tasks, and was not designed to handle long spans of input, having a maximum of 512 tokens. For reference, the retrieved results from a typical bag-of-words query on Robust04 has a median length of 679 tokens, and 66% of documents are longer than 512 tokens. Second, relevance judgments in nearly all newswire test collections are annotations on documents, not on individual sentences or passages. That is, given a query, we only know what documents are relevant, not spans within those documents. Typically, a document is considered relevant as long as some part of it is relevant, and in fact most of the document may not address the user's needs. Given these two challenges, it is not immediately obvious how to apply BERT to document ranking.

### 2.1 Key Insights

We propose two innovations that solve the above-mentioned challenges. First, we observe the existence of test collections that *fortuitously* contain passage-level relevance evidence: the MAchine Reading COmprehension (MS MARCO) dataset (Bajaj et al., 2018), the TREC Complex Answer Retrieval (CAR) dataset (Dietz et al., 2017), and the TREC Microblog (MB) datasets (Lin et al., 2014), described below.

MS MARCO features user queries sampled from Bing's search logs and passages extracted from web documents. Each query is associated with sparse relevance judgments by human editors. TREC CAR uses queries and paragraphs extracted from English Wikipedia: each query is formed by concatenating an article title and a section heading, and passages in that section are considered relevant. This makes CAR, essentially, a synthetic dataset. TREC Microblog datasets draw from the Microblog Tracks at TREC from 2011 to 2014, with topics (i.e., queries) and relevance judgments over tweets. We use the dataset prepared by Rao et al. (2019).

Note, however, that all three datasets are out of domain with respect to our task: MS MARCO passages are extracted from web pages, CAR paragraphs are taken from Wikipedia, and MB uses tweets. These corpora clearly differ from news articles to varying degrees. Furthermore, while MS MARCO and MB capture search tasks (although queries over tweets are qualitatively different), CAR "queries" (Wikipedia headings) do not reflect search queries from real users. Nevertheless, our experiments arrive at the surprising conclusion that these datasets are useful to train neural ranking models for news articles.

Our second innovation involves the aggregation of sentence-level evidence for document ranking. That is, given an initial ranked list of documents, we segment each into sentences, and then apply inference over *each sentence* separately, after which sentence-level scores are aggregated to yield a final score for ranking documents. This approach, in fact, is well motivated: There is a long thread of work in the information retrieval literature, dating back decades, that leverages passage retrieval techniques for document ranking (Hearst and Plaunt, 1993; Callan, 1994; Kaszkiel and Zobel, 1997; Clarke et al., 2000). In addition, recent studies of human searchers (Zhang et al., 2018b,a)

revealed that the "best" sentence or paragraph in a document provides a good proxy for document-level relevance.

### 2.2 Model Details

The core of our model is a BERT *sentence-level* relevance classifier. Following Nogueira and Cho (2019), this is framed as a binary classification task. We form the input to BERT by concatenating the query $Q$ and a sentence $S$ into the sequence `[[CLS]`, $Q$, `[SEP]`, $S$, `[SEP]]` and padding each sequence in a mini-batch to the maximum length in the batch. We feed the final hidden state corresponding to the `[CLS]` token in the model to a single layer neural network whose output represents the probability that sentence $S$ is relevant to the query $Q$.

To determine *document* relevance, we apply inference over each individual sentence in a candidate document, and then combine the top $n$ scores with the original document score as follows:

$$S_f = a \cdot S_{doc} + (1 - \alpha) \cdot \sum_{i=1}^{n} w_i \cdot S_i \quad (1)$$

where $S_{doc}$ is the original document score and $S_i$ is the $i$-th top scoring sentence according to BERT. In other words, the relevance score of a document comes from the combination of a document-level term-matching score and evidence contributions from the top sentences in the document as determined by the BERT model. The parameters $\alpha$ and the $w_i$'s can be tuned via cross-validation.

## 3 Experimental Setup

We begin with $\text{BERT}_{\text{Large}}$ (uncased, 340m parameters) from Devlin et al. (2019), and then fine-tune on the collections described in Section 2.1, individually and in combination. Despite the fact that tweets aren't always sentences and that MS MARCO and CAR passages, while short, may in fact contain more than one sentence, we treat all texts as if they were sentences for the purposes of fine-tuning BERT. For MS MARCO and CAR, we adopt exactly the procedure of Nogueira and Cho (2019). For MB, we tune on 2011–2014 data, with 75% of the total data reserved for training and the rest for validation.

We use the maximum sequence length of 512 tokens in all experiments. We train all models using cross-entropy loss for 5 epochs with a batch size of 16. We use Adam (Kingma and Ba, 2014) with an initial learning rate of $1 \times 10^{-5}$, linear learning rate warmup at a rate of 0.1 and decay of 0.1. All experiments are conducted on NVIDIA Tesla P40 GPUs with PyTorch v1.2.0.

During inference, we first retrieve an initial ranked list of documents to depth 1000 from the collection using the Anserini toolkit[1] (post-v0.5.1 commit from mid-August 2019, based on Lucene 8.0). Following Lin (2018) and Yang et al. (2019a), we use BM25 with RM3 query expansion (default parameters), which is a strong baseline, and has already been shown to beat most existing neural models. We clean the retrieved documents by stripping any HTML/XML tags and splitting each document into its constituent sentences with NLTK. If the length of a sentence with the meta-tokens exceeds BERT's maximum limit of 512, we further segment the spans into fixed size chunks. All sentences are then fed to the BERT model.

We conduct end-to-end document ranking experiments on three TREC newswire collections: the Robust Track from 2004 (Robust04) and the Common Core Tracks from 2017 and 2018 (Core17 and Core18). Robust04 comprises 250 topics, with relevance judgments on a collection of 500K documents (TREC Disks 4 and 5). Core17 and Core18 have only 50 topics each; the former uses 1.8M articles from the New York Times Annotated Corpus while the latter uses around 600K articles from the TREC Washington Post Corpus. Note that none of these collections were used to fine-tune the BERT relevance models; the only learned parameters are the weights in Eq (1).

Based on preliminary exploration, we consider up to the top three sentences; any more does not appear to yield better results. For Robust04, we follow the five-fold cross-validation settings in Lin (2018) and Yang et al. (2019a); for Core17 and Core18 we similarly apply five-fold cross validation. The parameters $\alpha$ and the $w_i$'s are learned via exhaustive grid search as follows: we fix $w_1 = 1$ and then vary $a, w_2, w_3 \in [0, 1]$ with a step size 0.1, selecting the parameters that yield the highest average precision (AP). Retrieval results are reported in terms of AP, precision at rank 20 (P@20), and NDCG@20.

Code for replicating all the experiments described in this paper is available as part of our recently-developed Birch IR engine.[2] Additional

[1] http://anserini.io/
[2] http://birchir.io/

| | Robust04 | | | Core17 | | | Core18 | | |
|---|---|---|---|---|---|---|---|---|---|
| **Model** | **AP** | **P@20** | **NDCG@20** | **AP** | **P@20** | **NDCG@20** | **AP** | **P@20** | **NDCG@20** |
| BM25+RM3 | 0.2903 | 0.3821 | 0.4407 | 0.2823 | 0.5500 | 0.4467 | 0.3135 | 0.4700 | 0.4604 |
| 1S: BERT(MB) | 0.3408$^\dagger$ | 0.4335$^\dagger$ | 0.4900$^\dagger$ | 0.3091$^\dagger$ | 0.5620 | 0.4628 | 0.3393$^\dagger$ | 0.4930 | 0.4848$^\dagger$ |
| 2S: BERT(MB) | 0.3435$^\dagger$ | 0.4386$^\dagger$ | 0.4964$^\dagger$ | 0.3137$^\dagger$ | 0.5770 | 0.4781 | 0.3421$^\dagger$ | 0.4910 | 0.4857$^\dagger$ |
| 3S: BERT(MB) | 0.3434$^\dagger$ | 0.4422$^\dagger$ | 0.4998$^\dagger$ | 0.3154$^\dagger$ | 0.5880 | 0.4852$^\dagger$ | 0.3419$^\dagger$ | 0.4950$^\dagger$ | 0.4878$^\dagger$ |
| 1S: BERT(CAR) | 0.3025$^\dagger$ | 0.3970$^\dagger$ | 0.4509 | 0.2814$^\dagger$ | 0.5500 | 0.4470 | 0.3120 | 0.4680 | 0.4586 |
| 2S: BERT(CAR) | 0.3025$^\dagger$ | 0.3970$^\dagger$ | 0.4509 | 0.2814$^\dagger$ | 0.5500 | 0.4470 | 0.3116 | 0.4670 | 0.4585 |
| 3S: BERT(CAR) | 0.3025$^\dagger$ | 0.3970$^\dagger$ | 0.4509 | 0.2814$^\dagger$ | 0.5500 | 0.4470 | 0.3113 | 0.4670 | 0.4584 |
| 1S: BERT(MS MARCO) | 0.3028$^\dagger$ | 0.3964$^\dagger$ | 0.4512 | 0.2817$^\dagger$ | 0.5500 | 0.4468 | 0.3121 | 0.4670 | 0.4594 |
| 2S: BERT(MS MARCO) | 0.3028$^\dagger$ | 0.3964$^\dagger$ | 0.4512 | 0.2817$^\dagger$ | 0.5500 | 0.4468 | 0.3121 | 0.4670 | 0.4594 |
| 3S: BERT(MS MARCO) | 0.3028$^\dagger$ | 0.3964$^\dagger$ | 0.4512 | 0.2817$^\dagger$ | 0.5500 | 0.4468 | 0.3121 | 0.4670 | 0.4594 |
| 1S: BERT(CAR $\rightarrow$ MB) | 0.3476$^\dagger$ | 0.4380$^\dagger$ | 0.4988$^\dagger$ | 0.3103$^\dagger$ | 0.5830 | 0.4758 | 0.3385$^\dagger$ | 0.4860 | 0.4785 |
| 2S: BERT(CAR $\rightarrow$ MB) | 0.3470$^\dagger$ | 0.4400$^\dagger$ | 0.5015$^\dagger$ | 0.3140$^\dagger$ | 0.5830 | 0.4817$^\dagger$ | 0.3386$^\dagger$ | 0.4810 | 0.4755 |
| 3S: BERT(CAR $\rightarrow$ MB) | 0.3466$^\dagger$ | 0.4398$^\dagger$ | 0.5014$^\dagger$ | 0.3143$^\dagger$ | 0.5830 | 0.4807 | 0.3382$^\dagger$ | 0.4830 | 0.4731 |
| 1S: BERT(MS MARCO $\rightarrow$ MB) | 0.3676$^\dagger$ | 0.4610$^\dagger$ | 0.5239$^\dagger$ | 0.3292$^\dagger$ | 0.6080$^\dagger$ | 0.5061$^\dagger$ | 0.3486$^\dagger$ | **0.4920** | **0.4953**$^\dagger$ |
| 2S: BERT(MS MARCO $\rightarrow$ MB) | **0.3697**$^\dagger$ | 0.4657$^\dagger$ | 0.5324$^\dagger$ | **0.3323**$^\dagger$ | 0.6170$^\dagger$ | **0.5092**$^\dagger$ | 0.3496$^\dagger$ | 0.4830 | 0.4899$^\dagger$ |
| 3S: BERT(MS MARCO $\rightarrow$ MB) | 0.3691$^\dagger$ | **0.4669**$^\dagger$ | **0.5325**$^\dagger$ | 0.3314$^\dagger$ | **0.6200**$^\dagger$ | 0.5070$^\dagger$ | **0.3522**$^\dagger$ | 0.4850 | 0.4899$^\dagger$ |

Table 1: Ranking effectiveness on Robust04, Core17, and Core18 in terms of AP, P@20, and NDCG@20.

details about the technical design of our system are presented in Yilmaz et al. (2019), a companion demonstration paper.

## 4 Results and Discussion

Our main results are shown in Table 1. The top row shows the BM25+RM3 query expansion baseline using default Anserini parameters.[3] The remaining blocks display the ranking effectiveness of our models on Robust04, Core17, and Core18. In parentheses we describe the fine-tuning procedure: for instance, MSMARCO $\rightarrow$ MB refers to a model that was first fine-tuned on MS MARCO and then on MB. The $n$S preceding the model name indicates that inference was performed using the top $n$ scoring sentences from each document.

Table 1 also includes results of significance testing using paired $t$-tests, comparing each condition with the BM25+RM3 baseline. We report significance at the $p < 0.01$ level, with appropriate Bonferroni corrections for multiple hypothesis testing. Statistically significant differences with respect to the baseline are denoted by †.

We find that BERT fine-tuned on MB alone significantly outperforms the BM25+RM3 baseline for all three metrics on Robust04. On Core17 and Core18, we observe significant increases in AP as well (and other metrics in some cases). In other words, relevance models learned from tweets successfully transfer over to news articles despite large differences in domain. This surprising result highlights the relevance matching power introduced by the deep semantic information learned by BERT.

Fine-tuning on MS MARCO or CAR alone yields at most minor gains over the baselines across all collections, and in some cases actually hurts effectiveness. Furthermore, the number of sentences considered for final score aggregation does not seem to affect effectiveness. It also does not appear that the synthetic nature of CAR data helps much for relevance modeling on newswire collections. Interestingly, though, if we fine-tune on CAR and then MB (CAR $\rightarrow$ MB), we obtain better results than fine-tuning on either MS MARCO or CAR alone. In some cases, we slightly improve over fine-tuning on MB alone. One possible explanation could be that CAR has an effect similar to language model pre-training; it alone cannot directly help the downstream document retrieval task, but it provides a better representation that can benefit from MB fine-tuning.

However, we were surprised by the MS MARCO results: since the dataset captures a search task and the web passages are "closer" to our newswire collections than MB in terms of domain, we would have expected relevance transfer to be more effective. Results show, however, that fine-tuning on MS MARCO alone is far less effective than fine-tuning on MB alone.

Looking across all fine-tuning configurations, we see that the top-scoring sentence of each candi-

[3]Even though Lin (2018) and Yang et al. (2019a) report slightly higher AP scores for *tuned* BM25+RM3 on Robust04, for consistency we use default parameters because no careful tuning has been performed for Core17 and Core18.

date document alone seems to be a good indicator of document relevance, corroborating the findings of Zhang et al. (2018a). Additionally considering the second ranking sentence yields at most a minor gain, and in some cases, adding a third actually causes effectiveness to drop. This is quite a surprising finding, since it suggests that the document ranking problem, at least as traditionally formulated by information retrieval researchers, can be distilled into relevance prediction primarily at the sentence level.

In the final block of the table, we present our best model, with fine-tuning on MS MARCO and then on MB. We confirm that this approach is able to exploit *both* datasets, with a score that is higher than fine-tuning on each dataset alone. Let us provide some broader context for these scores: For Robust04, we report the highest AP score that we are aware of (0.3697). Prior to our work, the meta-analysis by Yang et al. (2019a), which analyzed over 100 papers up until early 2019,[4] put the best neural model at 0.3124 (Dehghani et al., 2018).[5] Furthermore, our results exceed the previous highest known score of 0.3686, which is a non-neural method based on ensembles (Cormack et al., 2009). This high water mark has stood unchallenged for ten years.

Recently, MacAvaney et al. (2019) reported 0.5381 NDCG@20 on Robust04 by integrating contextualized word representations into existing neural ranking models; unfortunately, they did not report AP results. Our best NDCG@20 on Robust04 (0.5325) approaches their results even though we optimize for AP. Finally, note that since we are only using Robust04 data for learning the document and sentence weights in Eq (1), and not for fine-tuning BERT itself, it is less likely that we are overfitting.

Our best model also achieves a higher AP on Core17 than the best TREC submission that does not make use of past labels or human intervention (`umass_baselnrm`, 0.275 AP) (Allan et al., 2017). Under similar conditions, we beat every TREC submission in Core18 as well (with the best run being `uwmrg`, 0.276 AP) (Allan et al., 2018). Core17 and Core18 are relatively new and thus have yet to receive much attention from researchers, but to our knowledge, these figures represent the state of the art.

[4] https://github.com/lintool/robust04-analysis

[5] Setting aside our own previous work (Yang et al., 2019c).

## 5 Conclusion

This paper shows how BERT can be adapted in a simple manner to yield large improvements in *ad hoc* document retrieval on "classic" TREC newswire test collections. Our results demonstrate two surprising findings: first, that relevance models can be transferred quite straightforwardly across domains by BERT, and second, that effective document retrieval requires only "paying attention" to a small number of "top sentences" in each document. Important future work includes more detailed analyses of these transfer effects, as well as a closer look at the contributions of document-level and sentence-level scores. Nevertheless, we believe that both findings pave the way for new directions in document ranking.

## Acknowledgments

This research was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada, and enabled by computational resources provided by Compute Ontario and Compute Canada.

## References

James Allan, Donna Harman, Evangelos Kanoulas, Dan Li, Christophe Van Gysel, and Ellen Voorhees. 2017. TREC 2017 Common Core Track overview. In *Proceedings of the Twenty-Sixth Text REtrieval Conference (TREC 2017)*, Gaithersburg, Maryland.

James Allan, Donna Harman, Evangelos Kanoulas, and Ellen Voorhees. 2018. TREC 2018 Common Core Track overview. In *Proceedings of the Twenty-Seventh Text REtrieval Conference (TREC 2018)*, Gaithersburg, Maryland.

Timothy G. Armstrong, Alistair Moffat, William Webber, and Justin Zobel. 2009. Improvements that don't add up: ad-hoc retrieval results since 1998. In *Proceedings of the 18th International Conference on Information and Knowledge Management (CIKM 2009)*, pages 601–610, Hong Kong, China.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2018. MS MARCO: A human generated MAchine Reading COmprehension dataset. *arXiv:1611.09268v3*.

James P. Callan. 1994. Passage-level evidence in document retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1994)*, pages 302–310, Dublin, Ireland.

Charles L. A. Clarke, Gordon V. Cormack, and Elizabeth Tudhope. 2000. Relevance ranking for one to three term queries. *Information Processing and Management*, 36:291–311.

Gordon V. Cormack, Charles L. A. Clarke, and Stefan Büttcher. 2009. Reciprocal rank fusion outperforms Condorcet and individual rank learning methods. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2009)*, pages 758–759, Boston, Massachusetts.

Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. 2019. Are we really making much progress? A worrying analysis of recent neural recommendation approaches. *arXiv:1907.06902*.

Mostafa Dehghani, Arash Mehrjou, Stephan Gouws, Jaap Kamps, and Bernhard Schölkopf. 2018. Fidelity-weighted learning. In *Proceedings of the 6th International Conference on Learning Representations (ICLR 2018)*, Vancouver, Canada.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota.

Laura Dietz, Manisha Verma, Filip Radlinski, and Nick Craswell. 2017. TREC Complex Answer Retrieval overview. In *Proceedings of the Twenty-Sixth Text REtrieval Conference (TREC 2017)*, Gaithersburg, Maryland.

Marti A. Hearst and Christian Plaunt. 1993. Subtopic structuring for full-length document access. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1993)*, pages 56–68, Pittsburgh, Pennsylvania.

Marcin Kaszkiel and Justin Zobel. 1997. Passage retrieval revisited. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1997)*, pages 178–185, Philadelphia, Pennsylvania.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv:1412.6980*.

Canjia Li, Yingfei Sun, Ben He, Le Wang, Kai Hui, Andrew Yates, Le Sun, and Jungang Xu. 2018. NPRF: A neural pseudo relevance feedback framework for ad-hoc information retrieval. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4482–4491, Brussels, Belgium.

Jimmy Lin. 2018. The neural hype and comparisons against weak baselines. *SIGIR Forum*, 52(2):40–51.

Jimmy Lin, Miles Efron, Yulu Wang, and Garrick Sherman. 2014. Overview of the TREC-2014 Microblog Track. In *Proceedings of the Twenty-Third Text REtrieval Conference (TREC 2014)*, Gaithersburg, Maryland.

Sean MacAvaney, Andrew Yates, Arman Cohan, and Nazli Goharian. 2019. CEDR: Contextualized embeddings for document ranking. In *Proceedings of the 42nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2019)*, pages 1101–1104, Paris, France.

Ryan McDonald, George Brokos, and Ion Androutsopoulos. 2018. Deep relevance ranking using enhanced document-query interactions. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1849–1860, Brussels, Belgium.

Bhaskar Mitra and Nick Craswell. 2019. An introduction to neural information retrieval. *Foundations and Trends in Information Retrieval*, 13(1):1–126.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with BERT. *arXiv:1901.04085*.

Harshith Padigela, Hamed Zamani, and W. Bruce Croft. 2019. Investigating the successes and failures of BERT for passage re-ranking. *arXiv:1905.01758*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana.

Yifan Qiao, Chenyan Xiong, Zhenghao Liu, and Zhiyuan Liu. 2019. Understanding the behaviors of BERT in ranking. *arXiv:1904.07531*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. Technical report.

Jinfeng Rao, Wei Yang, Yuhao Zhang, Ferhan Ture, and Jimmy Lin. 2019. Multi-perspective relevance matching with hierarchical ConvNets for social media search. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)*, pages 232–240, Honolulu, Hawaii.

Steffen Rendle, Li Zhang, and Yehuda Koren. 2019. On the difficulty of evaluating baselines: a study on recommender systems. *arXiv:1905.01395*.

D. Sculley, Jasper Snoek, Alex Wiltschko, and Ali Rahimi. 2018. Winner's curse? On pace, progress, and empirical rigor. In *Proceedings of the 6th International Conference on Learning Representations, Workshop Track (ICLR 2018)*.

Wei Yang, Kuang Lu, Peilin Yang, and Jimmy Lin. 2019a. Critically examining the "neural hype": Weak baselines and the additivity of effectiveness gains from neural ranking models. In *Proceedings of the 42nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2019)*, pages 1129–1132, Paris, France.

Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019b. End-to-end open-domain question answering with BERTserini. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 72–77, Minneapolis, Minnesota.

Wei Yang, Haotian Zhang, and Jimmy Lin. 2019c. Simple applications of BERT for ad hoc document retrieval. *arXiv:1903.10972*.

Zeynep Akkalyoncu Yilmaz, Shengjin Wang, Wei Yang, Haotian Zhang, and Jimmy Lin. 2019. Applying BERT to document retrieval with Birch. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*.

Haotian Zhang, Mustafa Abualsaud, Nimesh Ghelani, Mark D. Smucker, Gordon V. Cormack, and Maura R. Grossman. 2018a. Effective user interaction for high-recall retrieval: less is more. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM 2018)*, pages 187–196, Torino, Italy.

Haotian Zhang, Gordon V. Cormack, Maura R. Grossman, and Mark D. Smucker. 2018b. Evaluating sentence-level relevance feedback for high-recall information retrieval. *arXiv:1803.08988*.

# The Challenges of Optimizing Machine Translation for Low Resource Cross-Language Information Retrieval

**Constantine Lignos[1], Daniel Cohen[2], Yen-Chieh Lien[2], Pratik Mehta[1], W. Bruce Croft[2], Scott Miller[1]**
[1]Information Sciences Institute, University of Southern California
[2]Center for Intelligent Information Retrieval, University of Massachusetts Amherst
lignos@brandeis.edu
{dcohen,ylien,psmehta,croft}@cs.umass.edu,
smiller@isi.edu

## Abstract

When performing cross-language information retrieval (CLIR) for lower-resourced languages, a common approach is to retrieve over the output of machine translation (MT). However, there is no established guidance on how to optimize the resulting MT-IR system. In this paper, we examine the relationship between the performance of MT systems and both neural and term frequency-based IR models to identify how CLIR performance can be best predicted from MT quality. We explore performance at varying amounts of MT training data, byte pair encoding (BPE) merge operations, and across two IR collections and retrieval models. We find that the choice of IR collection can substantially affect the predictive power of MT tuning decisions and evaluation, potentially introducing dissociations between MT-only and overall CLIR performance.

## 1 Introduction

For cross-language information retrieval scenarios involving queries in a higher-resourced language and documents in a lower-resourced language, direct training of a cross-language IR system (Litschko et al., 2018; Sasaki et al., 2018; Vulić and Moens, 2015) is typically infeasible due to insufficient data in the document language.

A practical solution is to use machine translation to translate documents into the language of the queries, enabling the use of a traditional monolingual IR system trained on the higher-resourced language. Prior work (Pecina et al., 2014; Nikoulina et al., 2012) demonstrates the effectiveness of using an MT system for query translation within a CLIR framework, but increases in MT performance measured by BLEU (Papineni et al., 2002) do not necessarily correlate with IR performance. Taking a document translation approach, Hieber and Riezler (2015) show that by enforcing a bag of words constraint on the MT decoding process, a translated document can provide richer input to an MT-IR system. However, these prior works do not fully investigate the effectiveness of MT metrics in predicting performance of a downstream IR model.

In this paper, we explore the consequences of performing information retrieval over machine translation and address three questions regarding the performance of the resulting system. First, when making MT tuning decisions such as selecting the number of BPE merge operations, will a value that improves MT performance generally increase IR performance? Second, when evaluating MT performance, can standard metrics be adapted to better correlate with downstream IR performance? Finally, what is the general relationship between the performance of MT and a downstream IR task, and how is it affected by the choice of retrieval model?

## 2 Experiment Design

To address the above questions, we trained an MT system in a large number of configurations, used these models to produce translated English collections of varying quality, and then compared the performance of retrieval over translated collections to retrieval over the matching source English documents. While an MT-IR system of this type is most appropriately used on lower-resourced languages, the resources needed to perform such a study using publicly available data thus far only exist in higher-resourced languages. To simulate a lower-resourced setting, we used a small portion of the MT training data available from higher-resourced languages, ablated it into smaller subsets, and did not rely on any other language resources. By training translation models using varying amounts of training data, we are able to explore performance across a wide range of resource levels and examine



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3497–3502,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

the reliability of performance patterns. The software and data required to replicate the experiments reported here are available from https://github.com/ConstantineLignos/mt-clir-emnlp-2019.

### 2.1 Machine Translation

We trained German-English and Czech-English translation models using the News Commentary (NC) dataset version 13 (Tiedemann, 2012) as provided in the WMT18 training data. This dataset was chosen to ensure no overlap between MT training data and the Wikipedia and Europarl IR test sets. Up to 200k sentences were selected for training, 5k for validation, and 5k for testing. The training data was ablated from 200k to 25k sentences in increments of 25k to create 8 training sets of decreasing sizes, simulating a wide range of low-resource conditions. The MT training scenarios varied by language, training data size, and BPE configuration. For each training set, a joint (source and target languages) BPE (Sennrich et al., 2016) was learned using two different numbers of merge operations (hereafter *BPE size*), 16k and 32k.

We selected fairseq (Gehring et al., 2017) as the machine translation system for our experiments. The MT system was trained using the data preprocessing approach and parameters given in the documentation[1]. Moses was used for tokenization of all data. We used the `fconv_iwslt_de_en` fully convolutional architecture with a single set of hyperparameters that provided for robust convergence on all configurations: learning rate 0.25, gradient clip threshold (parameter `clip-norm`) 0.1, dropout 0.2, and max-tokens 4000.[2] The validation set was provided during training to identify stopping conditions. The epoch that yielded the highest validation set performance was used to decode the test data. Test data was decoded with a beam size of 5. BLEU scores were computed using `sacreBLEU` (Post, 2018), with the configuration *BLEU+case.lc+numrefs.1+smooth.none+tok.13a+version.1.2.12*.

[1] https://fairseq.readthedocs.io/en/latest/getting_started.html#training-a-new-model

[2] We also experimented with a transformer-based architecture. While it did produce BLEU scores several points higher for the highest data conditions, it did not reliably converge with smaller amounts of training data, typically producing single-digit BLEU scores. While further tuning would have likely addressed this issue, we opted to report results using the fully convolutional architecture and constant hyperparameters.

### 2.2 Information Retrieval

We selected two multilingual collections for retrieval experiments, Europarl and Wikipedia. We created a collection of the 2,330 non-empty Europarl V7 (Koehn, 2005) transcripts that were present in English, German, and Czech.[3] As there are no queries and relevance judgments specific to Europarl, we used GOV2 TREC topics 701–850 as domain-relevant English queries and treated the top 100 documents retrieved by BM25 for each query as the relevant documents for evaluation.

Wikipedia was selected to provide a second collection for comparison, specifically one large enough to support training a neural retrieval model. To create relevant query-document pairs for training and evaluation, we used titles as queries and treated each document as the sole relevant document for its own title (Sasaki et al., 2018). For evaluation, we selected 5k articles with content in all three languages. We constrained selection such that the shortest article across the three languages had at least 500 words, and the longest article was not more than three times as long as the shortest article. While the information content for the "same" article will not be identical across languages, this constraint helps limit cross-language information differences caused by article length.

**Models.** As *tf.idf* is still a competitive performer compared to neural models for ad-hoc retrieval (Guo et al., 2019), we evaluated both models to provide insight into how each responds to the noise introduced by MT. We used Okapi-BM25 (Jones et al., 2000) to represent a term-based approach, and the Duet architecture introduced by Mitra et al. (2017) to represent a neural model. The Duet architecture was chosen because it transforms the document's text into the most common character n-grams within a corpus, making it robust to subword translation errors that are likely to be caused by using BPE in a low resource setting. Duet has been shown to perform comparably to more recent models both for ad-hoc and passage length retrieval (Mitra and Craswell, 2019).

The Duet model was trained on English

[3] Europarl transcripts were considered empty and excluded from the collection if for any language they were absent or appeared to have no content, such as merely noting that a session did or did not occur without containing any session content. The method for identifying trivial transcripts was that any lines consisting of only whitespace, XML, or a parenthetical were removed. If fewer than three lines remained, the transcript was considered empty and excluded.

Wikipedia using the hyperparameters and loss function suggested by Mitra et al. (2017) rather than weakly supervised BM25 scores (Dehghani et al., 2017), which would produce a smoothed *tf.idf* neural BM25 model that behaves differently than a standard neural IR model. We also experimented with a word-based representation—using GLoVE (Pennington et al., 2014) embeddings—as input to Duet instead of using the most common character n-grams. It performed worse (highest RBO .091), likely due to being more sensitive to unusual subword unit combinations generated by the MT system.

**Evaluation.** The goal of our IR evaluation is to compare the ranking produced by retrieval over translated documents against that produced by retrieval over the English documents. We used the extrapolated rank-biased overlap (RBO, Webber et al., 2010) as our primary metric, defined as

$$RBO_{EXT}(S,T,p,k) = \frac{X_k}{k}\cdot p_k + \frac{1-p}{p}\sum_{d=1}^{k}\frac{X_d}{d}\cdot p_d$$

where $k$ is the current position in a rank list, $S, T$ are the two rank lists being compared, and $X_i$ is the size of the intersection of the two lists at depth $i$. This results in an RBO value of 1 being the upper bound when the rank list $S$ is the same as $T$. RBO is based on a probabilistic user model, facilitates indefinite rankings, and allows for incomplete sets to be compared, unlike Kendall's tau and Spearman's rho. We used the suggested parameter $p = 0.98$, where the top 50 documents of each rank list contain 86% of the weight. We also computed mean average precision (MAP) as a standard IR metric for comparison.

The BM25 model was evaluated against both the Europarl and Wikipedia collections. However, to avoid the performance degradation caused by cross-collection evaluation (Cohen et al., 2018), we only evaluate the Wikipedia-trained neural model on the Wikipedia evaluation collection. Due to the large number of conditions we evaluate for both MT and IR performance, results are shown in figures, while full tables are provided in Appendix A.1.

## 3 Results

Unsurprisingly, using larger amounts of MT training data leads to higher MT and end-to-end performance; we focus on the subtleties of performance and not this generality. Figure 1 shows lowercase BLEU on the MT test set for all training conditions; BLEU scores ranged 17.5–31.5 for Czech-English and 18.5–32.0 for German-English. Figure 2 shows RBO for retrieval over the output of each MT training configuration. While the relationship between MT training size and RBO is similar across languages, collections, and models, the performance of neural retrieval over Wikipedia is far lower than that of BM25, which we discuss below.

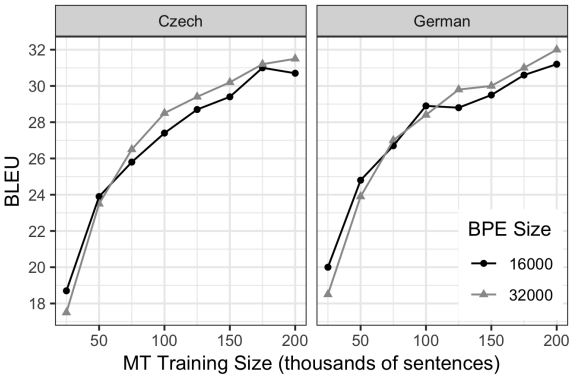


Figure 1: MT test BLEU for all training configurations.

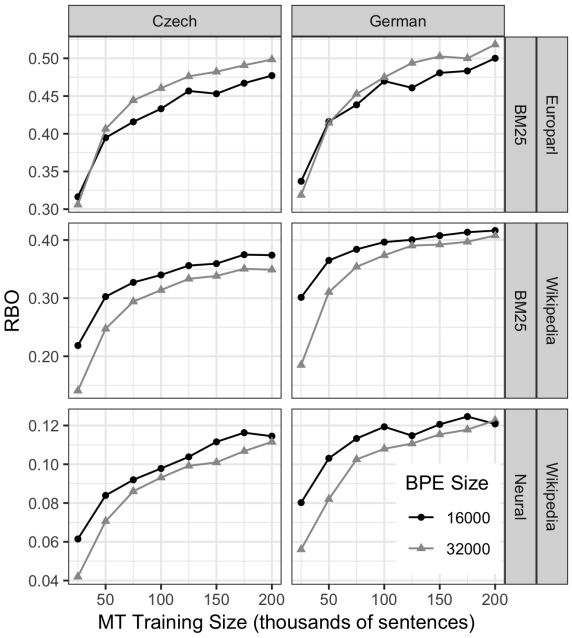


Figure 2: RBO for retrieval models on Europarl and Wikipedia across all MT training configurations.

| Collection/Model | Match | Mismatch |
|---|---|---|
| Europarl BM25 | 14 | 2 |
| Wiki BM25 | 5 | 11 |
| Wiki Neural | 6 | 10 |

Table 1: Counts of MT/IR performance tuning matches and mismatches when tuning BPE size.

### 3.1 Impact of BPE

We now address our first question: when making MT tuning decisions such as selecting the number of BPE merge operations, will a value that improves MT performance generally increase IR performance? We explore tuning BPE because it is commonly used in modern neural MT systems, and it is likely to have substantial end-to-end impact as it controls the subword input and output vocabulary. To address this question, we analyzed how often the directionality of the change in BLEU matched that of RBO when comparing across BPE size conditions. We categorized each configuration as a *match* if changing BPE size from 16k to 32k had an effect in the same direction for both MT (BLEU) and IR (RBO) performance, and categorized it as a *mismatch* otherwise. As shown in Table 1, the collection determined the probability of matches; for Europarl, BPE tuning decisions that increase BLEU reliably increase RBO, but for Wikipedia, more often than not tuning BPE to increase BLEU will *decrease* RBO.

Examining the MT and IR performance separately makes the cause of these tuning mismatches more clear. For MT, Figure 1 shows that 16k BPE performed slightly better at the lowest training data points, and 32k performed better at the highest. For IR, the effect of BPE varies across collections. On Europarl, the effect of BPE code size on IR performance is similar to that of MT. Increasing BPE size from 16k to 32k generally leads to better performance, with a mean MAP increase of .010 across all sizes. On Wikipedia, the 32k BPE size almost always performs worse, with mean MAP decreases of .056 (BM25) and .010 (neural). The difference between the MAP scores across the BPE configurations was statistically significant as determined by a Wilcoxon signed-rank test, a non-parametric paired sample test: Europarl BM25 ($p = .023$); Wikipedia BM25 ($p < .001$) and neural ($p < .001$).

We conclude that the nature of the queries, documents, and relevance characteristics of the downstream IR task plays a critical role when selecting BPE size or making other MT tuning decisions. Depending on the collection, attempts to improve MT may end up hurting end-to-end performance.

### 3.2 Evaluating MT for End-to-end Performance

We now turn to our second question: when evaluating MT, can standard metrics be adapted to better correlate with downstream IR performance? To explore this question, we computed correlations using Kendall's tau between RBO and the following: BLEU, BLEU's component 1–4-gram precisions, and variations of BLEU that remove punctuation, stem words using the Porter stemmer, or do both (as is common in term-based retrieval). Tau was computed separately within each combination of language, dataset, and retrieval model, measuring the correlation with RBO across changes in MT performance due to language, training data size, and BPE size.

| Coll./Model | BLEU | P1 | P2 | P3 | P4 |
|---|---|---|---|---|---|
| Europarl BM25 | .873 | **.881** | .867 | .864 | .862 |
| Wiki BM25 | **.722** | .689 | .708 | .714 | .711 |
| Wiki Neural | **.789** | .756 | .775 | .781 | .778 |

Table 2: Mean Kendall's Tau correlation with RBO for lowercased BLEU and each n-gram precision (P1=1-gram, etc.), by collection/model.

| Coll./Model | BLEU | No Punc. | Stem | Both |
|---|---|---|---|---|
| Europarl BM25 | .873 | .870 | **.875** | .873 |
| Wiki BM25 | .722 | **.728** | .717 | .722 |
| Wiki Neural | **.789** | .787 | .783 | .781 |

Table 3: Mean Kendall's Tau correlation with RBO for lowercased BLEU and variations, by collection/model.

Table 2 shows that the downstream IR task affects which MT measures correlate best with RBO. For Europarl, the best correlation with RBO was achieved by unigram precision, with correlation decreasing as larger n-grams are evaluated. For Wikipedia, BLEU provided the best correlation with RBO, and larger n-gram precisions correlated better with RBO than unigram precision. Table 3 gives correlations for the other variations of BLEU we explored; none led to a reliable improvement over standard lowercased BLEU, but results differed across collections/models.

We conclude that none of the n-gram precision components of BLEU or variations on it provide consistently better correlations with IR performance. However, given a specific collection and model, it is likely one of the alternatives we explored here or other metrics (e.g. ROUGE) can be slightly more predictive than standard BLEU.

### 3.3 IR Model Sensitivity

Finally, we explore our third question: what is the general relationship between the performance of MT and a downstream IR task, and how is it affected by the choice of retrieval model? Figure 3

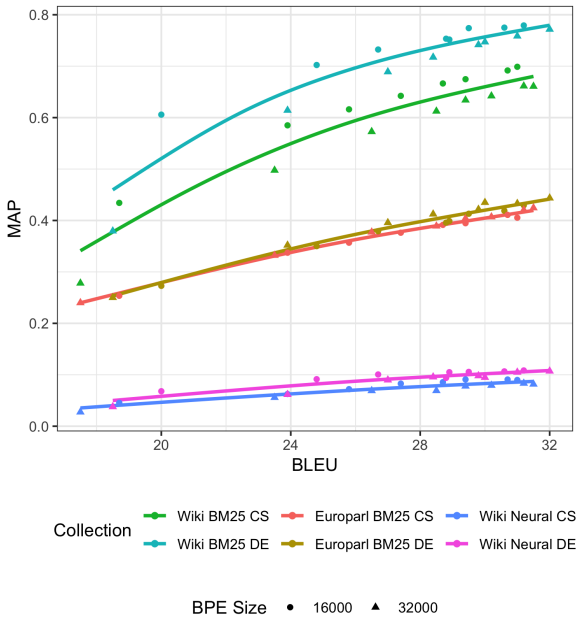


Figure 3: MAP for retrieval models across all collections and MT training configurations. GAM fit lines are provided for each combination of collection, retrieval model, and language (CS, Czech; DE, German).

shows MAP and BLEU across all tested configurations. The relationship between them is approximately linear above the lowest data points. Using linear regression to predict MAP from BLEU while accounting for variance due to language and BPE size, a one-point increase in BLEU was estimated to yield 1.33 points of MAP for Europarl BM25, 2.39 for Wikipedia BM25, and 0.42 for Wikipedia neural.

The neural Wikipedia model (highest RBO 0.12; highest MAP 0.11) significantly trailed BM25 (0.42; 0.78). Low RBO values show that the neural model degraded much more than BM25 when presented with MT as input. Given the regression described above, we can explore the BLEU score needed to match the MAP value of 0.20 that the same model attained retrieving over the English articles. Extrapolating generously, that would be achieved with a BLEU of approximately 53, a value unattainable in a low-resource scenario. As neural IR models are sensitive to shifts in the input distribution (Cohen et al., 2018), the most likely path for effective neural retrieval in a low-resource MT-IR setting is to train retrieval in a way that adapts to the idiosyncrasies of MT output, unlike the model we evaluated.

## 4 Conclusions

We conclude that there is no substitute for end-to-end, task-specific tuning when attempting to improve MT-IR system performance by increasing MT quality. As demonstrated by the effect of BPE tuning, changes made to the MT model can have opposite effects on performance when different IR collections are considered. The performance degradation of the evaluated neural IR model when retrieving over MT suggests that it is unlikely that off-the-shelf neural IR will be able to function adequately in an MT-IR system. Adapting neural models for use in an MT-IR setting and addressing the properties of MT output most harmful to their performance are promising avenues for future work to enable low-resource CLIR.

## Acknowledgements

This research is based upon work supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via contract # FA8650-17-C-9116. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

## References


Daniel Cohen, Bhaskar Mitra, Katja Hofmann, and W. Bruce Croft. 2018. Cross domain regularization for neural ranking models using adversarial learning. In *The 41st International ACM SIGIR Conference on Research #38; Development in Information Retrieval*, SIGIR '18, pages 1025–1028.

Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W. Bruce Croft. 2017. Neural ranking models with weak supervision. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 65–74.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional Sequence to Sequence Learning. In *Proceedings of ICML*.

Jiafeng Guo, Yixing Fan, Liang Pang, Liu Yang, Qingyao Ai, Hamed Zamani, Chen Wu, W. Bruce Croft, and Xueqi Cheng. 2019. A deep look into neural ranking models for information retrieval. *Information Processing & Management*.

Felix Hieber and Stefan Riezler. 2015. Bag-of-words forced decoding for cross-lingual information retrieval. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1172–1182.

K. Sparck Jones, S. Walker, and S. E. Robertson. 2000. A probabilistic model of information retrieval: Development and comparative experiments. *Information Processing & Management*, 36(6):779–808.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of MT Summit*, volume 5, pages 79–86.

Robert Litschko, Goran Glavas, Simone Paolo Ponzetto, and Ivan Vulic. 2018. Unsupervised cross-lingual information retrieval using monolingual data only. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018*, pages 1253–1256.

Bhaskar Mitra and Nick Craswell. 2019. An updated duet model for passage re-ranking. *arXiv preprint arXiv:1903.07666*.

Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to match using local and distributed representations of text for web search. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pages 1291–1299.

Vassilina Nikoulina, Bogomil Kovachev, Nikolaos Lagos, and Christof Monz. 2012. Adaptation of statistical machine translation model for cross-lingual information retrieval in a service context. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, pages 109–119.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Pavel Pecina, Ondřej Dušek, Lorraine Goeuriot, Jan Hajič, Jaroslava Hlaváčová, Gareth JF Jones, Liadh Kelly, Johannes Leveling, David Mareček, Michal Novák, et al. 2014. Adaptation of machine translation for multilingual information retrieval in the medical domain. *Artificial Intelligence in Medicine*, 61(3):165–185.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191.

Shota Sasaki, Shuo Sun, Shigehiko Schamoni, Kevin Duh, and Kentaro Inui. 2018. Cross-lingual learning-to-rank with shared representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 458–463.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1715–1725.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in opus. In *Workshop abstracts: Eighth International Conference on Language Resources and Evaluation*.

Ivan Vulić and Marie-Francine Moens. 2015. Monolingual and cross-lingual information retrieval models based on (bilingual) word embeddings. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 363–372.

William Webber, Alistair Moffat, and Justin Zobel. 2010. A similarity measure for indefinite rankings. *ACM Transactions on Information Systems*, 28(4):20:1–20:38.

# Rotate *King* to get *Queen*: Word Relationships as Orthogonal Transformations in Embedding Space

**Kawin Ethayarajh***
Stanford University
kawin@stanford.edu

## Abstract

A notable property of word embeddings is that word relationships can exist as linear substructures in the embedding space. For example, *gender* corresponds to $\vec{woman} - \vec{man}$ and $\vec{queen} - \vec{king}$. This, in turn, allows word analogies to be solved arithmetically: $\vec{king} - \vec{man} + \vec{woman} \approx \vec{queen}$. This property is notable because it suggests that models trained on word embeddings can easily learn such relationships as geometric translations. However, there is no evidence that models *exclusively* represent relationships in this manner. We document an alternative way in which downstream models might learn these relationships: orthogonal and linear transformations. For example, given a translation vector for *gender*, we can find an orthogonal matrix $R$, representing a rotation and reflection, such that $R(\vec{king}) \approx \vec{queen}$ and $R(\vec{man}) \approx \vec{woman}$. Analogical reasoning using orthogonal transformations is almost as accurate as using vector arithmetic; using linear transformations is more accurate than both. Our findings suggest that these transformations can be as good a representation of word relationships as translation vectors.

## 1 Introduction

Word embeddings are a cornerstone of current methods in NLP. A notable property of these vectors is that word relationships can exist as linear substructures in the embedding space (Mikolov et al., 2013a). For example, *gender* can be expressed as the translation vectors $\vec{woman} - \vec{man}$ and $\vec{queen} - \vec{king}$; similarly, *past tense* can be expressed as $\vec{thought} - \vec{think}$ and $\vec{talked} - \vec{talk}$. This, in turn, allows word analogies to be solved arithmetically. For example, *'man is to woman as king is to ?'* can be solved by finding the vector closest to $\vec{king} - \vec{man} + \vec{woman}$, which should be $\vec{queen}$ if one excludes the query words.

*Work partly done at the University of Toronto.

Ethayarajh et al. (2019a) proved that when there is no reconstruction error, a word analogy that can be solved arithmetically holds exactly over a set of ordered word pairs iff the co-occurrence shifted PMI is the same for every word pair and across any two word pairs. This means that strict conditions need to be satisfied by the training corpus for a word analogy to hold exactly, and these conditions are not necessarily satisfied by every analogy that makes intuitive sense. For example, most analogies involving countries and their currency cannot be solved arithmetically using Wikipedia-trained skipgram vectors (Ethayarajh et al., 2019a).

The fact that word relationships *can* exist as linear substructures is still notable, as it suggests that models trained with embeddings can easily learn these relationships as geometric translations. For example, as we noted earlier, it is easy to learn a translation vector $\vec{b}$ such that $\vec{queen} = \vec{king} + \vec{b}$ and $\vec{woman} = \vec{man} + \vec{b}$. However, Ethayarajh et al.'s proof and corresponding empirical evidence suggest that models do not *exclusively* represent word relationships in this manner. While past work has acknowledged that downstream models can capture relationships as complex non-linear transformations (Murdoch et al., 2018), it has not studied whether there are simpler linear alternatives that can also describe word relationships.

In this paper, we first document one such alternative: orthogonal transformations. More specifically, given the mean translation vector $\vec{b}$ for a word relationship (e.g., *gender*), we can find an orthogonal matrix that represents the relationship just as well as $\vec{b}$. For example, there is an orthogonal matrix $R$ such that $R(\vec{king}) \approx \vec{queen}$ and $R(\vec{man}) \approx \vec{woman}$. To find $R$ for a word relationship, we first create a source matrix $X$ of randomly sampled word vectors and a target matrix $Y$ by shifting $X$ by $\vec{b}$. We then use the closed-form solution to orthogonal Procrustes (Schönemann, 1966)



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3503–3508, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

to determine $R$, which is the orthogonal matrix that most closely maps $X$ to $Y$. If we broaden our search to include all linear transformations – not just orthogonal ones – we can find a matrix $A$ to represent the relationship by using the analytical solution to ordinary least squares.

We find that using orthogonal transformations for analogical reasoning is almost as accurate as using vector arithmetic, and using linear transformations is more accurate than both. However, given that finding the orthogonal matrix analytically is much more computationally expensive, we do not recommend using our method to solve analogies in practice. Rather, our key insight is that there are parsimonious representations of word relationships between the two extremes of simple geometric translations and complex nonlinear transformations. Our empirical finding offers novel insight into how downstream NLP models, both large and small, may be inferring word relationships. It suggests that a simple linear regression model or a single attention head of a Transformer (Vaswani et al., 2017) can adequately represent many word relationships, ranging from the one between a country and its capital to the one between an adjective and its superlative form.

## 2 Related Work

**Word Embeddings** Word embeddings are distributed representations in a low-dimensional continuous space. They can capture semantic and syntactic properties of words as linear substructures, allowing relationships to be expressed as geometric translations (Mikolov et al., 2013b). Word vectors can be learned from: (a) neural networks that learn representations by predicting co-occurrences (Bengio et al., 2003; Mikolov et al., 2013b); (b) low-rank approximations of word-context matrices containing a co-occurrence statistic (Landauer and Dumais, 1997; Levy and Goldberg, 2014b).

**Solving Word Analogies** There are two main strategies for solving word analogies, `3CosAdd` and `3CosMul` (Levy and Goldberg, 2014a). The former is the familiar vector arithmetic method: given a word analogy task *a:b::x:?*, the answer is $\text{argmin}_w \cos(\vec{w}, \vec{x} + \vec{b} - \vec{a})$. For `3CosMul`, the answer is $\text{argmin}_w(\cos(\vec{w},\vec{x})\cos(\vec{w},\vec{b}))/(\cos(\vec{w},\vec{a}) + \varepsilon)$, where $\varepsilon$ prevents null division. Although `3CosMul` is more accurate on average, we do not discuss it further because it does not create a distinct representation of the relationship. As noted previously, our goal is *not* to come up with a better strategy for solving analogies, but to show that there are parsimonious representations of word relationships other than translation vectors.

**Orthogonal Maps** Orthogonal transformations have been applied to word embeddings to achieve various objectives. Most famously, they have been used for the cross-lingual alignment of word embeddings trained on non-parallel data, for unsupervised machine translation (Conneau et al., 2018). Rothe et al. (2016) proposed a method for creating ultra-dense word embeddings in more meaningful subspaces by first learning an orthogonal transformation of the embeddings and then clipping all but the relevant dimensions. Park et al. (2017) built on this work, exploring several strategies for rotating embeddings to obtain more semantically meaningful dimensions. However, to our knowledge, orthogonal transformations themselves have not been used to represent word relationships; our work is novel in this respect.

## 3 Representing Word Relationships

To formalize the notion of a word relationship, we treat it as an invertible transformation that can hold over an arbitrary number of ordered pairs, following a similar framing proposed by Ethayarajh et al. (2019a) for word analogies. For example, the word pairs *{(Berlin, Germany), (Paris, France), (Ottawa, Canada)}* all express the same word relationship because some function $f$ maps each capital city to its respective country. In this paper, we look at three specific types of transformations: *translative*, *orthogonal*, and *linear*.

**Definition 1** *A word relationship $f$ is an invertible transformation that holds over a set of ordered pairs $S$ iff* $\forall\ (x,y) \in S, f(x) = y \wedge f^{-1}(y) = x$.

**Definition 2.1** *A translative word relationship $f$ is a transformation of the form $\vec{x} \mapsto \vec{x} + \vec{b}$, where $\vec{b}$ is the translation vector. $f$ holds over ordered pairs $S$ iff* $\forall\ (x,y) \in S, \vec{x} + \vec{b} = \vec{y}$.

**Definition 2.2** *An orthogonal word relationship $f$ is a transformation of the form $\vec{x} \mapsto R\vec{x}$, where $R^T R = I$. $f$ holds over ordered pairs $S$ iff* $\forall\ (x,y) \in S, R\vec{x} = \vec{y}$.

**Definition 2.3** *A linear word relationship $f$ is a transformation of the form $\vec{x} \mapsto A\vec{x}$, where $A$ is a non-degenerate square matrix. $f$ holds over ordered pairs $S$ iff* $\forall\ (x,y) \in S, A\vec{x} = \vec{y}$.

| Analogy Category | Accuracy | | | Avg Cosine Similarity with Solution | | |
|---|---|---|---|---|---|---|
| | Orthogonal | Linear | Translative | Orthogonal | Linear | Translative |
| capital-common-countries | **0.957** | **0.957** | **0.957** | 0.802 | 0.844 | 0.847 |
| capital-world | 0.922 | **0.966** | **0.966** | 0.727 | 0.786 | 0.793 |
| currency | 0.300 | **0.467** | 0.267 | 0.517 | 0.515 | 0.511 |
| city-in-state | 0.529 | 0.897 | **0.926** | 0.705 | 0.775 | 0.802 |
| family | **0.913** | **0.913** | **0.913** | 0.840 | 0.840 | 0.859 |
| gram1-adjective-to-adverb | 0.438 | **0.500** | **0.500** | 0.667 | 0.670 | 0.678 |
| gram2-opposite | **0.621** | 0.586 | 0.517 | 0.629 | 0.607 | 0.632 |
| gram3-comparative | 0.865 | 0.865 | **0.892** | 0.791 | 0.768 | 0.812 |
| gram4-superlative | **0.912** | 0.882 | **0.912** | 0.747 | 0.705 | 0.764 |
| gram5-present-participle | 0.909 | **0.939** | 0.848 | 0.813 | 0.808 | 0.829 |
| gram6-nationality-adjective | 0.902 | 0.902 | **0.927** | 0.816 | 0.837 | 0.841 |
| gram7-past-tense | 0.600 | **0.650** | 0.625 | 0.768 | 0.770 | 0.775 |
| gram8-plural | 0.892 | **0.919** | 0.892 | 0.796 | 0.787 | 0.807 |
| gram9-plural-verbs | **0.900** | 0.733 | 0.800 | 0.786 | 0.773 | 0.803 |
| Avg | 0.761 | **0.798** | 0.782 | 0.743 | 0.749 | 0.768 |

Table 1: The accuracy on our word analogy task – detailed in section 4.1 – when word relationships are represented as orthogonal, linear, and translative functions. The highest accuracy for each category is in bold. As seen in the last row, on average, orthogonal transformations are almost as accurate as translations (0.761 vs. 0.782), and the average cosine similarity between a transformed vector and the solution is about the same for both.

Given a set of word pairs $S$, how can we determine a translative, orthogonal, or linear transformation $f$ such that $\forall\ (x,y) \in S, f(\vec{x}) \approx \vec{y}$? Fortunately, there are closed form solutions for each case. To define a translation, we can simply take the mean of the pairwise difference vectors as our translation vector:

$$\vec{b} = \frac{1}{|S|} \sum_{(x,y)\in S} \vec{y} - \vec{x} \tag{1}$$

For orthogonal transformations, we first uniformly randomly sample $n$ words from the vocabulary and stack their word vectors to get a source matrix $X$. Then, we add $\vec{b}$ to each sampled word vector to get a target matrix $Y$. Finding the orthogonal matrix that most closely maps $X$ to $Y$ is called the orthogonal Procrustes problem:

$$R = \underset{\Omega}{\operatorname{argmin}} \|\Omega X - Y\|_F \text{ s.t. } \Omega^T\Omega = I \tag{2}$$

Orthogonal Procrustes has a closed-form solution that we can use to find $R$ (Schönemann, 1966). We frame the problem similarly to find a linear map $A$ that does not necessarily need to be orthogonal. If we assume that the linear transformation should minimize the ordinary least squares objective,

$$\begin{aligned} A &= \underset{\Omega}{\operatorname{argmin}} \|\Omega X - Y\|^2 \\ &= YX^T(XX^T)^{-1} \end{aligned} \tag{3}$$

Note that our approach to finding the orthogonal and linear transformations is to find those that best approximate the geometric translation by $\vec{b}$. The reasoning behind this is simple: in practice, the number of word pairs in $S$ is much smaller than the embedding dimensionality $d$, so trying to find a map $x \mapsto y\ \forall\ (x,y) \in S' \subset S$ would be very conducive to over-fitting. Since we randomly sample $n$ words to create $X$ and $Y$, we can choose $n \gg d$ to prevent this problem.

Although we are ultimately learning to approximate a translation, representing a word relationship as an orthogonal matrix as opposed to a translation vector has some useful mathematical properties, such as preserving the inner product: $\forall\ (x,y) \in S, \langle\vec{x},\vec{y}\rangle = \langle R\vec{x}, R\vec{y}\rangle$. Ethayarajh et al. (2019a) proved that when there is no reconstruction error, the word-context matrix $M$ that is implicitly factorized by models such as skipgram and GloVe can be recovered from the inner products of word vectors. Since the inner product is preserved under rotation, $M$ can also be recovered from an orthogonally transformed word space. The same does not hold under translation.

## 4 Evaluating the Representations

### 4.1 Task and Setup

We evaluate the different representations of word relationships using analogy tasks. However, we do not aim to solve word analogies in the traditional sense, since that largely depends on which words are present in each analogy's 4-tuple. Instead, we first calculate the mean translation vector $\vec{b}$ by av-

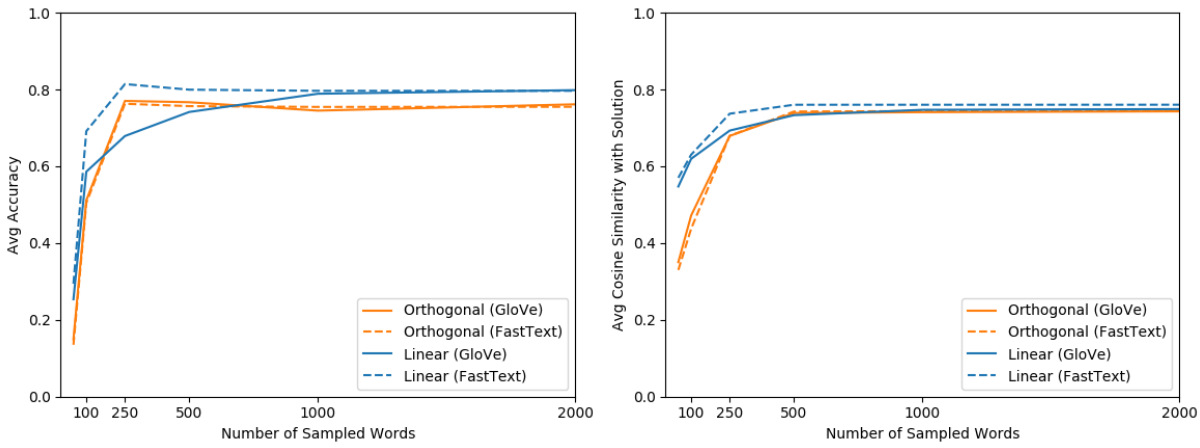


Figure 1: The accuracy on our word analogy task (left) and the average cosine similarity between the predicted and actual answers (right) as $n$, the number of sampled words used to learn the transformation, increases. The accuracy plateaus for $n \geq 250$ and the similarity plateaus for $n \geq 500$, suggesting that a robust transformation can be learned with relatively little data. Use of GloVe vs. FastText vectors makes no difference as $n \to 2000$.

eraging difference vectors across all word pairs, as defined in (1). $\vec{b}$ is also used to estimate matrices for the orthogonal and linear transformations (see (2) and (3)). We then create a set of word pairs for each analogy category: e.g., {*(Berlin, Germany), (Paris, France)*, ... } for *country-capital*. Each type of transformation – translative, orthogonal, and linear – is evaluated by how accurately it maps source words to target words in this set of word pairs. We use pre-trained GloVe vectors (Pennington et al., 2014) and $n = 2000$ for our main results in Table 1 and repeat our experiments with FastText vectors (Bojanowski et al., 2017) in Figure 1. We source our analogies from Mikolov et al. (2013a), as it contains a diverse set of categories.

### 4.2 Results

As seen in Table 1, orthogonal transformations are almost as accurate as geometric translations on our word analogy task: the average accuracy is 0.761 and 0.782 respectively. Linear transformations are more accurate than both (0.798). We would expect linear transformations to outperform orthogonal ones, given that the set of possible linear transformations is a superset of the set of possible orthogonal transformations. However, it is surprising that linear maps also outperform geometric translations, given that they are ultimately learned using translation vectors.

In the right half of Table 1, we list the average cosine similarity between the transformed source vector and the actual target vector (e.g, $\cos(R(\vec{king}), \vec{queen})$). This is to mitigate concerns that because the transformed source vector is mapped to the closest word vector, orthogonal transformations are only accurate due to the sparsity of the word space. As seen in Table 1, such concerns would be unfounded: the average cosine similarity for orthogonal transformations across all categories is 0.743, almost as high as the 0.768 for translations. This suggests that even if we considered a larger portion of the vocabulary as candidate answers, or if the word space were denser, orthogonal transformations would still be almost as accurate as translations on our task.

Our only hyperparameter is $n$, the number of randomly sampled words used to generate $X$ and $Y$. As seen in Figure 1, the accuracy plateaus for $n \geq 250$ and the average cosine similarity with the target vector plateaus for $n \geq 500$. This suggests that it is possible to learn orthogonal and linear transformations representing word relationships with relatively little data. As $n \to 2000$, differences in performance between GloVe and FastText disappear, though linear transformations are more accurate than orthogonal ones for all $n$. This also highlights why the translation vector $\vec{b}$ is used to learn the transformations instead of a subset of the actual word pairs: for most analogy categories, there are fewer than 250 pairs in the dataset, and learning with so few word pairs would lead to poor accuracy.

### 4.3 Implications

**Evaluating Embeddings** The literature has often evaluated the quality of word embeddings

by testing their ability to solve word analogies arithmetically (Mikolov et al., 2013b; Pennington et al., 2014). If word relationships were *exclusively* geometric translations, this would be reasonable. However, given that word relationships can also be orthogonal or linear transformations, the usefulness of these tests as a measure of embedding quality should be reconsidered. Other arguments, both theoretical and empirical, have been made in the past against the use of analogies for evaluation (Schluter, 2018; Drozd et al., 2016; Rogers et al., 2017).

**Model Architecture** Given that bias terms are not necessarily needed to learn word relationships, the architecture of downstream models trained on word embeddings can be modified accordingly. Transformers (Vaswani et al., 2017) already make extensive use of linear maps in multi-headed attention and appear to be justified in doing so. Moreover, recent work has found that certain attention heads are sensitive to certain syntax, positional information, and other linguistic phenomena (Voita et al., 2019; Clark et al., 2019). For example, Clark et al. (2019) identified heads that attend to the direct objects of verbs, noun determiners, and coreference mentions with surprisingly high accuracy. However, these studies have not examined whether semantic word relationships – such as gender – also correspond to certain attention heads. Given that our findings suggest that individual attention heads have the capacity to learn such relationships, this is a promising direction for future work.

**Bias in Word Embeddings** The most common method for removing gender bias in word embeddings involves defining a bias subspace in the embedding space and then subtracting from each word vector its projection on this subspace (Bolukbasi et al., 2016). Under certain conditions, this method can provably debias skipgram and GloVe word embeddings (Ethayarajh et al., 2019b), but in practice, these conditions are typically not satisfied and gender associations can still be recovered from the embedding space (Gonen and Goldberg, 2019). Our findings in this paper suggest another way in which downstream models may be learning such biases, by representing gender as an orthogonal or linear transformation. This, in turn, may help explain the existence of such bias in contextualized word representations as well (Zhao et al., 2019). Given that these transformations are another way in which social biases can manifest, exploring more diverse strategies for debiasing – or alternatively, understanding the limits of debiasing strategies – is another direction for future work.

## 5 Conclusion

Word relationships in embedding space are generally thought of as simple geometric translations or complex non-linear transformations. However, we found that there are parsimonious representations of relationships between these two extremes, namely orthogonal and linear transformations. In addition, we found that it is possible to easily learn an orthogonal or linear transformation for a word relationship given its mean translation vector. Analogical reasoning done using linear transformations is in fact more accurate than using geometric translations. This finding offers novel insight into how downstream NLP models may be inferring word relationships. For example, it suggests that a single attention head in a Transformer has sufficient capacity to represent a semantic or syntactical word relationship, concurring with recent findings that certain attention heads have syntax- and position-specific behavior.

## Acknowledgments

We thank the anonymous reviewers for their insightful comments. We thank the Natural Sciences and Engineering Research Council of Canada (NSERC) for their financial support.

## References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3(Feb):1137–1155.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Advances in Neural Information Processing Systems*, pages 4349–4357.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. 2019. What does bert look

at? an analysis of bert's attention. *arXiv preprint arXiv:1906.04341*.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. In *International Conference on Learning Representations*.

Aleksandr Drozd, Anna Gladkova, and Satoshi Matsuoka. 2016. Word embeddings, analogies, and machine learning: Beyond king-man+ woman= queen. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3519–3530.

Kawin Ethayarajh, David Duvenaud, and Graeme Hirst. 2019a. Towards understanding linear word analogies. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3253–3262, Florence, Italy. Association for Computational Linguistics.

Kawin Ethayarajh, David Duvenaud, and Graeme Hirst. 2019b. Understanding undesirable word embedding associations. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 1696–1705.

Hila Gonen and Yoav Goldberg. 2019. Lipstick on a pig: Debiasing methods cover up systematic gender biases in word embeddings but do not remove them. *arXiv preprint arXiv:1903.03862*.

Thomas K Landauer and Susan T Dumais. 1997. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104(2):211.

Omer Levy and Yoav Goldberg. 2014a. Linguistic regularities in sparse and explicit word representations. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 171–180.

Omer Levy and Yoav Goldberg. 2014b. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*, pages 2177–2185.

Tomas Mikolov, Kai Chen, Greg S Corrado, and Jeff Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

W James Murdoch, Peter J Liu, and Bin Yu. 2018. Beyond word importance: Contextual decomposition to extract interactions from LSTMs. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*.

Sungjoon Park, JinYeong Bak, and Alice Oh. 2017. Rotated word vector representations and their interpretability. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 401–411.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Anna Rogers, Aleksandr Drozd, and Bofang Li. 2017. The (too many) problems of analogical reasoning with word vectors. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (* SEM 2017)*, pages 135–148.

Sascha Rothe, Sebastian Ebert, and Hinrich Schütze. 2016. Ultradense word embeddings by orthogonal transformation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 767–777.

Natalie Schluter. 2018. The word analogy testing caveat. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 242–246.

Peter H Schönemann. 1966. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *arXiv preprint arXiv:1905.09418*.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Ryan Cotterell, Vicente Ordonez, and Kai-Wei Chang. 2019. Gender bias in contextualized word embeddings. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 629–634.

# GlossBERT: BERT for Word Sense Disambiguation with Gloss Knowledge

**Luyao Huang, Chi Sun, Xipeng Qiu*, Xuanjing Huang**
Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
School of Computer Science, Fudan University
825 Zhangheng Road, Shanghai, China
{lyhuang18,sunc17,xpqiu,xjhuang}@fudan.edu.cn

## Abstract

Word Sense Disambiguation (WSD) aims to find the exact sense of an ambiguous word in a particular context. Traditional supervised methods rarely take into consideration the lexical resources like WordNet, which are widely utilized in knowledge-based methods. Recent studies have shown the effectiveness of incorporating gloss (sense definition) into neural networks for WSD. However, compared with traditional *word expert* supervised methods, they have not achieved much improvement. In this paper, we focus on how to better leverage gloss knowledge in a supervised neural WSD system. We construct *context-gloss* pairs and propose three BERT-based models for WSD. We fine-tune the pre-trained BERT model and achieve new state-of-the-art results on WSD task.

## 1 Introduction

Word Sense Disambiguation (WSD) is a fundamental task and long-standing challenge in Natural Language Processing (NLP), which aims to find the exact sense of an ambiguous word in a particular context (Navigli, 2009). Previous WSD approaches can be grouped into two main categories: knowledge-based and supervised methods.

Knowledge-based WSD methods rely on lexical resources like WordNet (Miller, 1995) and usually exploit two kinds of lexical knowledge. The gloss, which defines a word sense meaning, is first utilized in Lesk algorithm (Lesk, 1986) and then widely taken into account in many other approaches (Banerjee and Pedersen, 2002; Basile et al., 2014). Besides, structural properties of semantic graphs are mainly used in graph-based algorithms (Agirre et al., 2014; Moro et al., 2014).

Traditional supervised WSD methods (Zhong and Ng, 2010; Shen et al., 2013; Iacobacci et al., 2016) focus on extracting manually designed features and then train a dedicated classifier (*word expert*) for every target lemma.

Although *word expert* supervised WSD methods perform better, they are less flexible than knowledge-based methods in the all-words WSD task (Raganato et al., 2017a). Recent neural-based methods are devoted to dealing with this problem. Kågebäck and Salomonsson (2016) present a supervised classifier based on Bi-LSTM, which shares parameters among all word types except the last layer. Raganato et al. (2017a) convert WSD task to a sequence labeling task, thus building a unified model for all polysemous words. However, neither of them can totally beat the best *word expert* supervised methods.

More recently, Luo et al. (2018b) propose to leverage the gloss information from WordNet and model the semantic relationship between the context and gloss in an improved memory network. Similarly, Luo et al. (2018a) introduce a (hierarchical) co-attention mechanism to generate co-dependent representations for the context and gloss. Their attempts prove that incorporating gloss knowledge into supervised WSD approach is helpful, but they still have not achieved much improvement, because they may not make full use of gloss knowledge.

In this paper, we focus on how to better leverage gloss information in a supervised neural WSD system. Recently, the pre-trained language models, such as ELMo (Peters et al., 2018) and BERT (Devlin et al., 2018), have shown their effectiveness to alleviate the effort of feature engineering. Especially, BERT has achieved excellent results in question answering (QA) and natural language inference (NLI). We construct *context-gloss* pairs from glosses of all possible senses (in WordNet) of the target word, thus treating WSD task as a sentence-pair classification problem. We fine-

*Corresponding author.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3509–3514,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

**Sentence with four targets:**
Your research stopped when a convenient assertion could be made.

| **Context-Gloss Pairs of the target word [research]** | Label | Sense Key |
|---|---|---|
| [CLS] Your research ... [SEP] systematic investigation to ... [SEP] | Yes | research%1:04:00:: |
| [CLS] Your research ... [SEP] a search for knowledge [SEP] | No | research%1:09:00:: |
| [CLS] Your research ... [SEP] inquire into [SEP] | No | research%2:31:00:: |
| [CLS] Your research ... [SEP] attempt to find out in a ... [SEP] | No | research%2:32:00:: |
| **Context-Gloss Pairs with weak supervision of the target word [research]** | Label | Sense Key |
| [CLS] Your "research" ... [SEP] research: systematic investigation to ... [SEP] | Yes | research%1:04:00:: |
| [CLS] Your "research" ... [SEP] research: a search for knowledge [SEP] | No | research%1:09:00:: |
| [CLS] Your "research" ... [SEP] research: inquire into [SEP] | No | research%2:31:00:: |
| [CLS] Your "research" ... [SEP] research: attempt to find out in a ... [SEP] | No | research%2:32:00:: |

Table 1: The construction methods. The sentence is taken from SemEval-2007 WSD dataset. The ellipsis "..." indicates the remainder of the sentence or the gloss.

tune the pre-trained BERT model and achieve new state-of-the-art results on WSD task. In particular, our contribution is two-fold:

1. We construct *context-gloss* pairs and propose three BERT-based models for WSD.

2. We fine-tune the pre-trained BERT model, and the experimental results on several English all-words WSD benchmark datasets show that our approach significantly outperforms the state-of-the-art systems.

## 2 Methodology

In this section, we describe our method in detail.

### 2.1 Task Definition

In WSD, a sentence $s$ usually consists of a series of words: $\{w_1, \cdots, w_m\}$, and some of the words $\{w_{i_1}, \cdots, w_{i_k}\}$ are targets $\{t_1, \cdots, t_k\}$ need to be disambiguated. For each target $t$, its candidate senses $\{c_1, \cdots, c_n\}$ come from entries of its lemma in a pre-defined sense inventory (usually WordNet). Therefore, WSD task aims to find the most suitable entry (symbolized as unique sense key) for each target in a sentence. See a sentence example in Table 1.

### 2.2 BERT

BERT (Devlin et al., 2018) is a new language representation model, and its architecture is a multi-layer bidirectional Transformer encoder. BERT model is pre-trained on a large corpus and two novel unsupervised prediction tasks, i.e., masked language model and next sentence prediction tasks are used in pre-training. When incorporating BERT into downstream tasks, the fine-tuning procedure is recommended. We fine-tune the pre-trained BERT model on WSD task.

**BERT(Token-CLS)** Since every target in a sentence needs to be disambiguated to find its exact sense, WSD task can be regarded as a token-level classification task. To incorporate BERT to WSD task, we take the final hidden state of the token corresponding to the target word (if more than one token, we average them) and add a classification layer for every target lemma, which is the same as the last layer of the Bi-LSTM model (Kågebäck and Salomonsson, 2016).

### 2.3 GlossBERT

BERT can explicitly model the relationship of a pair of texts, which has shown to be beneficial to many pair-wise natural language understanding tasks. In order to fully leverage gloss information, we propose GlossBERT to construct *context-gloss* pairs from all possible senses of the target word in WordNet, thus treating WSD task as a sentence-pair classification problem.

We describe our construction method with an example (See Table 1). There are four targets in this sentence, and here we take target word *research* as an example:

**Context-Gloss Pairs** The sentence containing target words is denoted as *context* sentence. For each target word, we extract glosses of all $N$ possible senses (here $N = 4$) of the target word (research) in WordNet to obtain the *gloss* sentence. [CLS] and [SEP] marks are added to the *context-gloss* pairs to make it suitable for the input of BERT model. A similar idea is also used in aspect-based sentiment analysis (Sun et al., 2019).

**Context-Gloss Pairs with Weak Supervision** Based on the previous construction method, we add weak supervised signals to the *context-gloss*

| Dataset | Total | Noun | Verb | Adj | Adv |
|---|---|---|---|---|---|
| SemCor | 226036 | 87002 | 88334 | 31753 | 18947 |
| SE2 | 2282 | 1066 | 517 | 445 | 254 |
| SE3 | 1850 | 900 | 588 | 350 | 12 |
| SE07 | 455 | 159 | 296 | 0 | 0 |
| SE13 | 1644 | 1644 | 0 | 0 | 0 |
| SE15 | 1022 | 531 | 251 | 160 | 80 |

Table 2: Statistics of the different parts of speech annotations in English all-words WSD datasets.

pairs (see the highlighted part in Table 1). The signal in the *gloss* sentence aims to point out the target word, and the signal in the *context* sentence aims to emphasize the target word considering the situation that a target word may occur more than one time in the same sentence.

Therefore, each target word has $N$ *context-gloss* pair training instances ($label \in \{yes, no\}$). When testing, we output the probability of $label = yes$ of each *context-gloss* pair and choose the sense corresponding to the highest probability as the prediction label of the target word. We experiment with three GlossBERT models:

**GlossBERT(Token-CLS)** We use context-gloss pairs as input. We highlight the target word by taking the final hidden state of the token corresponding to the target word (if more than one token, we average them) and add a classification layer ($label \in \{yes, no\}$).

**GlossBERT(Sent-CLS)** We use context-gloss pairs as input. We take the final hidden state of the first token `[CLS]` as the representation of the whole sequence and add a classification layer ($label \in \{yes, no\}$), which does not highlight the target word.

**GlossBERT(Sent-CLS-WS)** We use context-gloss pairs with weak supervision as input. We take the final hidden state of the first token `[CLS]` and add a classification layer ($label \in \{yes, no\}$), which weekly highlight the target word by the weak supervision.

## 3 Experiments

### 3.1 Datasets

The statistics of the WSD datasets are shown in Table 2.

**Training Dataset** Following previous work (Luo et al., 2018a,b; Raganato et al., 2017a,b; Iacobacci et al., 2016; Zhong and Ng, 2010), we choose SemCor3.0 as training corpus, which is the largest corpus manually annotated with WordNet sense for WSD.

**Evaluation Datasets** We evaluate our method on several English all-words WSD datasets. For a fair comparison, we use the benchmark datasets proposed by Raganato et al. (2017b) which include five standard all-words fine-grained WSD datasets from the Senseval and SemEval competitions: Senseval-2 (**SE2**), Senseval-3 (**SE3**), SemEval-2007 (**SE07**), SemEval-2013 (**SE13**) and SemEval-2015 (**SE15**). Following Luo et al. (2018a), Luo et al. (2018b) and Raganato et al. (2017a), we choose **SE07**, the smallest among these test sets, as the development set.

**WordNet** Since Raganato et al. (2017b) map all the sense annotations in these datasets from their original versions to WordNet 3.0, we extract word sense glosses from WordNet 3.0.

### 3.2 Settings

We use the pre-trained uncased $\text{BERT}_{\text{BASE}}$ model[1] for fine-tuning, because we find that $\text{BERT}_{\text{LARGE}}$ model performs slightly worse than $\text{BERT}_{\text{BASE}}$ in this task. The number of Transformer blocks is 12, the number of the hidden layer is 768, the number of self-attention heads is 12, and the total number of parameters of the pre-trained model is 110M. When fine-tuning, we use the development set (**SE07**) to find the optimal settings for our experiments. We keep the dropout probability at 0.1, set the number of epochs to 4. The initial learning rate is 2e-5, and the batch size is 64.

### 3.3 Results

Table 3 shows the performance of our method on the English all-words WSD benchmark datasets. We compare our approach with previous methods.

The first block shows the MFS baseline, which selects the most frequent sense in the training corpus for each target word.

The second block shows two knowledge-based systems. $\text{Lesk}_{ext+emb}$ (Basile et al., 2014) is a variant of Lesk algorithm (Lesk, 1986) by calculating the gloss-context overlap of the target word. Babelfy (Moro et al., 2014) is a unified graph-based approach which exploits the semantic network structure from BabelNet.

[1] https://storage.googleapis.com/bert_models/2018_10_18/uncased_L-12_H-768_A-12.zip

| | Dev | Test Datasets | | | | Concatenation of Test Datasets | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| System | SE07 | SE2 | SE3 | SE13 | SE15 | Noun | Verb | Adj | Adv | **All** |
| MFS baseline | 54.5 | 65.6 | 66.0 | 63.8 | 67.1 | 67.7 | 49.8 | 73.1 | 80.5 | 65.5 |
| Lesk$_{ext+emb}$ | 56.7 | 63.0 | 63.7 | 66.2 | 64.6 | 70.0 | 51.1 | 51.7 | 80.6 | 64.2 |
| Babelfy | 51.6 | 67.0 | 63.5 | 66.4 | 70.3 | 68.9 | 50.7 | 73.2 | 79.8 | 66.4 |
| IMS | 61.3 | 70.9 | 69.3 | 65.3 | 69.5 | 70.5 | 55.8 | 75.6 | 82.9 | 68.9 |
| IMS$_{+emb}$ | 62.6 | 72.2 | 70.4 | 65.9 | 71.5 | 71.9 | 56.6 | 75.9 | 84.7 | 70.1 |
| Bi-LSTM | - | 71.1 | 68.4 | 64.8 | 68.3 | 69.5 | 55.9 | 76.2 | 82.4 | 68.4 |
| Bi-LSTM$_{+att.+LEX+POS}$ | 64.8 | 72.0 | 69.1 | 66.9 | 71.5 | 71.5 | 57.5 | 75.0 | 83.8 | 69.9 |
| GAS$_{ext}$ (Linear) | - | 72.4 | 70.1 | 67.1 | 72.1 | 71.9 | 58.1 | 76.4 | 84.7 | 70.4 |
| GAS$_{ext}$ (Concatenation) | - | 72.2 | 70.5 | 67.2 | 72.6 | 72.2 | 57.7 | 76.6 | 85.0 | 70.6 |
| CAN$^s$ | - | 72.2 | 70.2 | 69.1 | 72.2 | 73.5 | 56.5 | 76.6 | 80.3 | 70.9 |
| HCAN | - | 72.8 | 70.3 | 68.5 | 72.8 | 72.7 | 58.2 | 77.4 | 84.1 | 71.1 |
| BERT(Token-CLS) | 61.1 | 69.7 | 69.4 | 65.8 | 69.5 | 72.0 | 57.8 | 73.5 | 84.4 | 68.6 |
| GlossBERT(Sent-CLS) | 69.2 | 76.5 | 73.4 | 75.1 | 79.5 | 79.1 | 65.4 | 79.3 | 84.8 | 75.8 |
| GlossBERT(Token-CLS) | 71.9 | 77.0 | **75.4** | 74.6 | 79.3 | 78.8 | 66.8 | **79.9** | 85.0 | 76.3 |
| GlossBERT(Sent-CLS-WS) | **72.5** | **77.7** | 75.2 | **76.1** | **80.4** | **79.8** | **67.1** | 79.6 | **87.4** | **77.0** |

Table 3: F1-score (%) for fine-grained English all-words WSD on the test sets in the framework of Raganato et al. (2017b) (including the development set **SE07**). **Bold** font indicates best systems. The five blocks list the MFS baseline, two knowledge-based systems, two traditional *word expert* supervised systems, six recent neural-based systems and our systems, respectively. Results in first three blocks come from Raganato et al. (2017b), and others from the corresponding papers.

The third block shows two *word expert* traditional supervised systems. IMS (Zhong and Ng, 2010) is a flexible framework which trains SVM classifiers and uses local features. And IMS$_{+emb}$ (Iacobacci et al., 2016) is the best configuration of the IMS framework, which also integrates word embeddings as features.

The fourth block shows several recent neural-based methods. Bi-LSTM (Kågebäck and Salomonsson, 2016) is a baseline for neural models. Bi-LSTM$_{+att.+LEX+POS}$ (Raganato et al., 2017a) is a multi-task learning framework for WSD, POS tagging, and LEX with self-attention mechanism, which converts WSD to a sequence learning task. GAS$_{ext}$ (Luo et al., 2018b) is a variant of GAS which is a gloss-augmented variant of the memory network by extending gloss knowledge. CAN$^s$ and HCAN (Luo et al., 2018a) are sentence-level and hierarchical co-attention neural network models which leverage gloss knowledge.

In the last block, we report the performance of our method. BERT(Token-CLS) is our baseline, which does not incorporate gloss information, and it performs slightly worse than previous traditional supervised methods and recent neural-based methods. It proves that directly using BERT cannot obtain performance growth. The other three methods outperform other models by a substantial margin, which proves that the improvements come from leveraging BERT to better exploit gloss information. It is worth noting that our method achieves significant improvements in **SE07** and **Verb** over previous methods, which have the highest ambiguity level among all datasets and all POS tags respectively according to Raganato et al. (2017b).

Moreover, GlossBERT(Token-CLS) performs better than GlossBERT(Sent-CLS), which proves that highlighting the target word in the sentence is important. However, the weakly highlighting method GlossBERT(Sent-CLS-WS) performs best in most circumstances, which may result from its combination of the advantages of the other two methods.

### 3.4 Discussion

There are two main reasons for the great improvements of our experimental results. First, we construct *context-gloss* pairs and convert WSD problem to a sentence-pair classification task which is similar to NLI tasks and train only one classifier, which is equivalent to expanding the corpus. Second, we leverage BERT (Devlin et al., 2018) to better exploit the gloss information. BERT model shows its advantage in dealing with sentence-pair classification tasks by its amazing improvement on QA and NLI tasks. This advantage comes from both of its two novel unsupervised prediction tasks.

Compared with traditional *word expert* supervised methods, our GlossBERT shows its effectiveness to alleviate the effort of feature engineering and does not require training a dedicated classifier for every target lemma. Up to now, it can be said that the neural network method can totally beat the traditional *word expert* method. Compared with recent neural-based methods, our so-

lution is more intuitive and can make better use of gloss knowledge. Besides, our approach demonstrates that when we fine-tune BERT on a downstream task, converting it into a sentence-pair classification task may be a good choice.

## 4 Conclusion

In this paper, we seek to better leverage gloss knowledge in a supervised neural WSD system. We propose a new solution to WSD by constructing *context-gloss* pairs and then converting WSD to a sentence-pair classification task. We fine-tune the pre-trained BERT model and achieve new state-of-the-art results on WSD task.

## Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments. The research work is supported by National Natural Science Foundation of China (No. 61751201 and 61672162), Shanghai Municipal Science and Technology Commission (16JC1420401 and 17JC1404100), Shanghai Municipal Science and Technology Major Project (No.2018SHZDZX01) and ZJLab.

## References

Eneko Agirre, Oier López de Lacalle, and Aitor Soroa. 2014. Random walks for knowledge-based word sense disambiguation. *Computational Linguistics*, 40(1):57–84.

Satanjeev Banerjee and Ted Pedersen. 2002. An adapted lesk algorithm for word sense disambiguation using wordnet. In *International conference on intelligent text processing and computational linguistics*, pages 136–145. Springer.

Pierpaolo Basile, Annalina Caputo, and Giovanni Semeraro. 2014. An enhanced lesk word sense disambiguation algorithm through a distributional semantic model. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1591–1600.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. Embeddings for word sense disambiguation: An evaluation study. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 897–907.

Mikael Kågebäck and Hans Salomonsson. 2016. Word sense disambiguation using a bidirectional lstm. *arXiv preprint arXiv:1606.03568*.

Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation*, pages 24–26. Citeseer.

Fuli Luo, Tianyu Liu, Zexue He, Qiaolin Xia, Zhifang Sui, and Baobao Chang. 2018a. Leveraging gloss knowledge in neural word sense disambiguation by hierarchical co-attention. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1402–1411.

Fuli Luo, Tianyu Liu, Qiaolin Xia, Baobao Chang, and Zhifang Sui. 2018b. Incorporating glosses into neural word sense disambiguation. *arXiv preprint arXiv:1805.08028*.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics*, 2:231–244.

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM computing surveys (CSUR)*, 41(2):10.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237.

Alessandro Raganato, Claudio Delli Bovi, and Roberto Navigli. 2017a. Neural sequence learning models for word sense disambiguation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1156–1167.

Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. 2017b. Word sense disambiguation: A unified evaluation framework and empirical comparison. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 99–110.

Hui Shen, Razvan Bunescu, and Rada Mihalcea. 2013. Coarse to fine grained sense disambiguation in wikipedia. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, volume 1, pages 22–31.

Chi Sun, Luyao Huang, and Xipeng Qiu. 2019. Utilizing bert for aspect-based sentiment analysis via constructing auxiliary sentence. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 380–385.

Zhi Zhong and Hwee Tou Ng. 2010. It makes sense: A wide-coverage word sense disambiguation system for free text. In *Proceedings of the ACL 2010 system demonstrations*, pages 78–83.

# Leveraging Adjective-Noun Phrasing Knowledge for Comparison Relation Prediction in Text-to-SQL

**Haoyan Liu**[1]*, **Lei Fang**[2], **Qian Liu**[3], **Bei Chen**[2], **Jian-Guang Lou**[2], **Zhoujun Li**[1]
[1]State Key Lab of Software Development Environment, Beihang University, China
[2]Microsoft Research, Beijing, China
[3]State Key Lab of Virtual Reality Technology and Systems, Beihang University, China
{haoyan.liu, qian.liu, lizj}@buaa.edu.cn; {leifa, beichen, jlou}@microsoft.com

## Abstract

One key component in text-to-SQL is to predict the comparison relations between columns and their values. To the best of our knowledge, no existing models explicitly introduce external common knowledge to address this problem, thus their capabilities of predicting comparison relations are limited beyond training data. In this paper, we propose to leverage adjective-noun phrasing knowledge mined from the web to predict the comparison relations in text-to-SQL. Experimental results on both the original and the re-split Spider dataset show that our approach achieves significant improvement over state-of-the-art methods on comparison relation prediction.

## 1 Introduction

Text-to-SQL (Yaghmazadeh et al., 2017; Zhong et al., 2017), which aims at mapping natural language to SQL queries, is one of the most important tasks in natural language processing. Most state-of-the-art models are end-to-end neural network based models (Zhang et al., 2017; Xu et al., 2017; Yu et al., 2018a; Herzig and Berant, 2018; Dong and Lapata, 2018; Yu et al., 2018b), which mainly extend the Seq2Seq architecture with some complicated network structures. As shown by Yu et al. (2018b), the performances of most methods are overstated, because they just match semantic parsing results, rather than truly understand the meanings of inputs (Finegan-Dollak et al., 2018). In this paper, we study the comparison relation prediction problem, as the comparison relations between columns and their values are not well understood by existing methods.

In SQL queries, comparison relations are expressed by the comparison operators ($=, \neq, <, \leq, >, \geq$) and value ordering keywords (`ASC`, `DESC` in `ORDER BY` expression). In most text-to-SQL models, the comparison relations are either generated using Seq2Seq architectures (Zhong et al., 2017; Dong and Lapata, 2018), or predicted using classifiers trained with output decoding features (Xu et al., 2017; Yu et al., 2018b). We give an example to show that external common knowledge is indispensable to truly understand the comparison relations on unseen data.

| rank | name | time | birth_date |
|---|---|---|---|
| 1 | Martina | 11′9 | 19800930 |
| 2 | Mirjana | 12′5 | 19820309 |
| 3 | Justine | 12′9 | 19820601 |

Table 1: Player Basic Information.

Table 1 shows the basic information about the athletes of the 100-meter sprint. Given the query *"what is the name of the oldest player ?"*, the goal is to generate the SQL "`SELECT name FROM players ORDER BY birth_date ASC LIMIT 1`". It should be noted that `ASC` represents the common knowledge that small value in `birth_date` column means *"old"*. Supposing that there are columns named `age` and some queries containing *"old"* in the training data, it's easy for the trained models to remember that *"old"* means selecting a large value from column `age`. But if `birth_date` is unseen in the training data, there is little chance to predict the comparison relation correctly without the common knowledge that *"age"* and *"birth_date"* both represent age but have opposite value polarities. Similarly, for query *"fast runner"*, models should select a large value from column `speed` or a small value from column `time`. There are some related works that address the intensity of adjectives (De Melo and Bansal, 2013; Ruppenhofer et al., 2014; Sharma et al., 2017), however,

*This work was done when the first author was an intern at Microsoft Research Asia.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3515–3520, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

no existing work studies the relations between the value polarities of adjective-noun phrasing pairs.

In this paper, we propose to explicitly incorporate the column value polarities as external knowledge in text-to-SQL models. Our goal is to scale the capabilities of existing models on comparison relation prediction to unseen data. The column value polarities are named as "adjective-noun phrasing knowledge", which can be easily constructed from the web. We further formulate the phrasing knowledge as feature vectors, which can be generically fed to existing models. Experimental results on both the original and the re-split Spider dataset show that our approach achieves significant improvement over state-of-the-art methods on comparison relation prediction.

## 2 Adjective-Noun Phrasing Knowledge

There are 3 steps to construct the knowledge: 1) adjective-noun pair candidate extraction, 2) value polarity mining, and 3) adjective-noun pair clustering. The adjective-noun phrasing knowledge will be two clusters, each of which is a list of adjective-noun pairs with the same value polarity.

### 2.1 Adjective-Noun Pair Candidate Extraction

As the value polarities depend on both adjectives and nouns, we first extract adjective-noun pairs that could be candidates of the phrasing knowledge. Typically, table column names could be considered as hypernyms of the corresponding cell values. Motivated by this, we gather a list of noun candidates, which consists of the concepts (hypernyms) from Microsoft Concept Graph[1] (Cheng et al., 2015; Wu et al., 2012) and the column names whose value types are number or date in the Web Table Corpora[2] (Lehmberg et al., 2016).

To extract the adjectives that modify the noun candidates, we introduce two POS tag patterns:

```
(1) [ADJ] [NOUN]
(2) [NOUN] is|are|was|were|be [ADJ]
```

where `[ADJ]` and `[NOUN]` are the POS tags for adjectives and nouns, respectively. We apply these two patterns to Wikipedia dump[3] to extract adjective-noun pair candidates. Taking *price* as an example, we would obtain the adjectives: *high*, *expensive*, and *cheap*, etc.

[1] https://concept.research.microsoft.com

[2] http://webdatacommons.org/webtables/

[3] https://dumps.wikimedia.org/enwiki/

### 2.2 Value Polarity Mining

As shown in Table 1, the value polarities of `age` and `birth_date` are opposite though both words are about *"age"*. We propose to mine the value polarities from the Web Table Corpora automatically. We assume that for two columns in the same table, if their corresponding values have negative correlations, they have opposite value polarities[4]. We use the Spearman's $\rho$ coefficients to measure the correlations between two columns. For each two columns with value type number or date, if the frequency of their co-occurrences is above 20 and the coefficients $\rho$ are below $-0.9$ for more than $50\%$ co-occurrences in the corpora, the two columns are determined to have opposite value polarities.

### 2.3 Adjective-Noun Pair Clustering

So far, we have obtained the adjective-noun pairs and value polarity relations for nouns. However, it is still unclear whether the polarity is positive or negative. Positive (negative) polarity means that large (small) column values shall be selected when the noun is modified by an adjective. For example, the polarities of <`age`, *old*> and <`price`, *expensive*> are positive, while <`age`, *young*> and <`price`, *cheap*> are negative. Our goal is to separate the adjective-noun pairs into two clusters based on the value polarities.

Supposing that we know the polarity of <`age`, *old*> is positive, we can derive that the polarities of <`age`, *young*> and <`birth_date`, *old*> are negative, because *"old"* and *"young"* are antonyms, and the value polarities of `age` and `birth_date` are opposite. Motivated by this, we extend and group the adjective-noun pairs into clusters based on the following rules:

- the polarities of two pairs are the same if the adjectives are synonyms and the nouns are the same, or the nouns are synonyms and the adjectives are the same;
- the polarities of two pairs are the opposite if the adjectives are antonyms and the nouns are the same, or the nouns have opposite value polarities and the adjectives are the same.

It should be noted that each cluster contains two sets of pairs with opposite polarities. We manually assign polarity labels to clusters in top sizes

[4] We also studied the assumption that columns with positive correlations have the same value polarities, but find that it introduces much noise.

or with high total frequencies. One potential issue is that there might be conflicts due to the synonyms and antonyms resources, i.e., there may be two pairs assigned opposite polarities in the same cluster. In practice, there are only a few such cases, and we separate them into individual clusters and manually label these pairs. After that, we obtain the adjective-noun phrasing knowledge organized in two clusters, each of which is a list of adjective-noun pairs with the same polarity.

## 3 Phrasing Knowledge As Features

We propose to formulate knowledge as features to incorporate the adjective-noun phrasing knowledge. It is very generic, because knowledge feature can be easily combined with the input or hidden output of existing neural models.

### 3.1 Baseline Models

We use Spider (Yu et al., 2018c) as the dataset. We do not use other datasets like WikiSQL (Zhong et al., 2017) because WikiSQL queries that contain comparison relations are very simple, and do not require additional knowledge for understanding. We introduce the knowledge feature to two baseline models, namely syntaxSQLNet[5] (Yu et al., 2018b) and SQLNet[6] (Xu et al., 2017; Yu et al., 2018c). We do not use other syntax tree-based or sequence-based baselines like coarse-to-fine (Dong and Lapata, 2018) because they cannot handle the Spider dataset.

Currently[7], syntaxSQLNet achieves the best performance on Spider. SyntaxSQLNet and SQLNet solve the text-to-SQL task using a sequence-to-set structure. They decompose the SQL generation procedure into multiple individual modules. The comparison relation prediction consists of two parts: the comparison operator prediction in `OP` module of `WHERE` and `GROUP Having` components and ordering prediction in `DESC/ASC/LIMIT(DAL)` module of `ORDER BY` component. Both the two modules are classifiers defined as:

$$p(y) = \mathcal{F}(V \tanh(WX)), \qquad (1)$$

where $X$ represents the input features; $y$ is the output label; and $W$ and $V$ are trainable parameters. Function $\mathcal{F}$ is defined as `sigmoid` in SQLNet and `softmax` in syntaxSQLNet. The `OP` module predicts the comparison operator $y \in \{=, >, <, >=, <=, !=, \texttt{LIKE}, \texttt{NOTIN}, \texttt{IN}, \texttt{BETWEEN}\}$, while the `DAL` module predicts the ordering keyword $y \in \{\texttt{DESC}, \texttt{ASC}, \texttt{DESC LIMIT}, \texttt{ASC LIMIT}\}$.

### 3.2 Knowledge as Features

The adjective-noun phrasing knowledge is formulated as additional input features to existing models. Let $X_K$ denote the knowledge feature, we rewrite the classifier for `OP` and `DAL` module as:

$$p(y) = \mathcal{F}(V \tanh(W[X : X_K])), \qquad (2)$$

where [:] denotes the concatenation of feature vectors. The intuition is that even if $X$ is unseen, the knowledge feature $X_K$ will help to make the correct prediction. We will give an example to show how to construct the knowledge feature.

As the adjective-noun phrasing knowledge consists of two clusters, we use two fixed random vectors to represent the two clusters, denoted as polarity features. We define the knowledge feature as the attention weighted polarity features with knowledge masks.

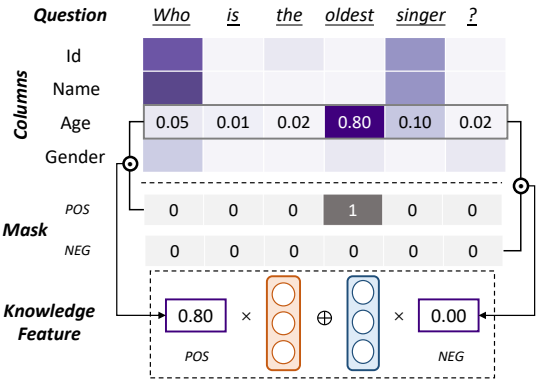


Figure 1: Knowledge as feature.

In Figure 1, the up palette shows the attention matrix between column names and question tokens. In syntaxSQLNet and SQLNet, the attention between column names and question tokens is defined by:

$$\alpha_{Q/CS} = \text{softmax}(H_Q W H_{CS}^{\top}), \qquad (3)$$

where $H_{CS}$ and $H_Q$ represent the hidden state of LSTM for column names and question tokens, respectively. Suppose that we are predicting the comparison operator for column `age` using the `DAL` module. We first construct the positive and

[5] https://github.com/taoyds/syntaxSQL

[6] https://github.com/taoyds/spider/tree/master/baselines/sqlnet

[7] As of mid May when we prepare this submission.

negative knowledge mask vectors, whose lengths are equal to the number of question tokens. In mask vectors, 1 means that the pair of the column name and the corresponding question token exists in phrasing knowledge. Then, we calculate the weight of positive and negative polarity features using the inner product ($\odot$) of knowledge mask vectors and the attention vector. For example, in Figure 1, the weight of positive and negative polarity features are 0.8 and 0, respectively. After that, we obtain the knowledge feature by the element-wise sum ($\oplus$) of the weighted positive and negative polarity features, denoted by $+_{\text{weighted}}$.

To obtain the mask vectors, we need to match the nouns in phrasing pairs with column names in the Spider dataset. For example, column names `birth date`, `birthdate`, `birth_date` and `date of birth` shoud be matched with the same noun *"date of birth"* in the phrasing knowledge. We heuristically define the matching score for column name $n_1$ and noun $n_2$ as:

$$\begin{aligned}\text{Score}(n_1, n_2) =& 0.2R_t(n_1, n_2) + 0.2R_c(n_1, n_2) \\ &+0.6\cos(\sum_{t_1 \in n_1} \text{idf}_{t_1} v_{t_1}, \sum_{t_2 \in n_2} \text{idf}_{t_2} v_{t_2}),\end{aligned} \quad (4)$$

where $R_t$ and $R_c$ are the token and char level fuzzy matching ratios, which is calculated by the output of fuzzywuzzy[8] divided by 100; and the last part is the cosine similarity between the inverse document frequency (idf) weighted sum of token embeddings, where the "document" refers to one column name in dataset. Given a column name in `OP` or `DAL` module, we select phrasing pairs whose noun matching score is ranked in top 10 as the knowledge to obtain the knowledge mask vectors.

Another straightforward way is to define the polarity feature as the knowledge feature directly, denoted as $+_{\text{direct}}$, where the polarity is determined by the attention weighted knowledge mask score. In the example above, the positive polarity feature will be considered as knowledge feature since the positive mask has a higher attention weighted score ($0.8 > 0$). We will evaluate both the $+_{\text{weighted}}$ and $+_{\text{direct}}$ knowledge features in the experiments.

## 4 Experiments

### 4.1 Data & Settings

For quality assurance, we manually choose 91 adjectives that have value polarities from top 200 frequent adjectives in Wikipedia. The adjectives with no value polarities like "important", "happy" and "reasonable" are filtered out. Then we use the adjective synonyms crawled from Bing Dictionary[9], the noun synonyms and adjective antonyms from WordNet (Miller, 1995) to cluster the adjective-noun pairs. We obtain $4,133$ distinct phrasing pairs in $2,689$ clusters. After that, we manually label the clusters with total sum of pair frequencies in top 50 or total number of pairs in top 50. Finally, we obtain 970 distinct adjective-noun pairs, which cover about $79.65\%$ usages among all pairs. We manually evaluate the value polarities of all the 970 pairs, and the accuracy is $87.8\%$. The pairs after manually revision are served as the adjective-noun phrasing knowledge.

For Spider dataset, we remove the nested SQL queries and queries containing `JOIN`, because the performance of baseline models on complex queries is rather low, and we would more like to see the result that whether the comparison relation is correctly predicted given the right column. There are $4,490$ questions after removing the complex queries, with $3,946$ training and $544$ development data. We manually review all the $4,490$ questions and the corresponding tables, and find that there are $1,212$ questions containing comparison relations (excluding $=, \neq$), among which 668 questions require adjective phrasing knowledge. The total number of required distinct adjective-noun pairs is 728, which is considered as ground-truth knowledge (denoted as *_gt*). After matching the mined adjective phrasing knowledge (970 pairs) with the Spider column names using Equation 4, we obtain 631 distinct pairs where 598 pairs exist in the ground-truth knowledge, the precision of knowledge matching using Equation 4 is $94.8\%$ $(598/631)$, and the recall is $82.1\%$ $(598/728)$.

To further evaluate effectiveness of phrasing knowledge, we re-split the training and development data of Spider, to ensure that the knowledge applied in the development set has not been seen or applied in the training set. The sizes of the re-split training and development sets are $3,906$ and $584$, respectively. For model training, the dimension of knowledge feature is 50 with other parameters remaining the same as baseline models.

### 4.2 Phrasing Knowledge Examples

In the step of value polarity mining (described in Section 2.2), we obtain 758 noun pairs with oppo-

[8] https://github.com/seatgeek/fuzzywuzzy

[9] https://www.bing.com/dict

| | | Select | -AGG | Where | -OP | Group | -Having | Order | -DAL | Overall |
|---|---|---|---|---|---|---|---|---|---|---|
| *Original* | SQLNet | 62.7 | 64.9 | 40.7 | 45.0 | 58.1 | 73.1 | 62.4 | 66.9 | 39.9 |
| | +weighted | 63.2 | 63.9 | **40.8** | 44.6 | 48.1 | 59.2 | 61.1 | 67.2 | 40.4 |
| | +direct | **65.0** | **65.9** | 36.4 | 37.9 | **60.0** | 65.5 | **64.4** | **68.9** | **42.8** |
| | +weighted_gt | 65.7 | 67.0 | 40.8 | 42.7 | 58.9 | 68.4 | 64.4 | 68.9 | 43.0 |
| | +direct_gt | 65.8 | 67.3 | 40.3 | 44.8 | 54.2 | 69.3 | 65.9 | 70.5 | 42.1 |
| | syntaxSQL | 62.1 | 64.0 | 30.1 | 38.8 | 66.9 | 69.4 | 46.3 | 54.8 | 39.5 |
| | +weighted | **65.0** | **67.6** | 33.3 | 40.3 | 59.0 | 63.1 | **49.1** | **58.7** | **41.0** |
| | +direct | 61.8 | 64.6 | **39.5** | **46.7** | 66.4 | 68.9 | 48.7 | 57.7 | 40.1 |
| | +weighted_gt | 65.9 | 68.1 | 37.6 | 47.2 | 60.6 | 63.1 | 53.5 | 58.9 | 42.5 |
| | +direct_gt | 63.5 | 64.8 | 30.4 | 39.0 | 64.4 | 69.5 | 50.6 | 52.9 | 41.2 |
| *Re-Split* | SQLNet | 64.4 | 65.7 | 38.4 | 42.4 | 52.1 | 64.5 | 60.9 | 68.3 | 40.8 |
| | +weighted | 63.2 | 63.9 | **40.8** | **44.6** | 48.1 | 59.2 | 61.1 | 67.2 | 40.4 |
| | +direct | **65.0** | **65.9** | 36.4 | 37.9 | **60.0** | **65.5** | **64.4** | **68.9** | **42.8** |
| | +weighted_gt | 65.2 | 66.7 | 37.4 | 42.3 | 54.2 | 69.2 | 63.2 | 73.1 | 39.0 |
| | +direct_gt | 62.8 | 64.4 | 40.8 | 44.5 | 60.0 | 72.2 | 60.7 | 70.0 | 40.1 |
| | syntaxSQL | 66.0 | 69.0 | 30.0 | 41.2 | 64.8 | 68.0 | 50.5 | 60.4 | 39.7 |
| | +weighted | 61.6 | 63.5 | 36.6 | 41.1 | 56.9 | 62.2 | 60.2 | **71.5** | 38.2 |
| | +direct | 65.7 | 67.1 | **37.9** | **41.6** | 59.5 | **69.8** | **62.5** | 71.3 | **40.8** |
| | +weighted_gt | 67.5 | 69.2 | 33.2 | 45.0 | 67.7 | 69.3 | 48.0 | 58.6 | 41.6 |
| | +direct_gt | 64.1 | 66.2 | 38.5 | 49.9 | 66.7 | 68.2 | 52.7 | 60.5 | 41.4 |

Table 2: F1-scores on the original and re-split Spider dataset. *_gt* means introducing the ground-truth knowledge. The **bold numbers** highlight the best results when introducing the matched mined phrasing knowledge outperforms the baselines, while the underlined numbers highlight that when introducing the ground truth knowledge.

| | **Positive** | **Negative** |
|---|---|---|
| age | old, eld, high | young, low |
| birth date | young | old, eld |
| date | fresh, new, recent, late | old, early |
| score | high | low |
| rank | low | high |
| price | costly, expensive, high, pricy | inexpensive cheap, low |

Table 3: Examples of value polarity.

site polarities. For example, the opposite-polarity words for *age* are {*birth year, date of birth, birthday*}, and for *rank* are {*point, vote, score*}.

Table 3 shows some examples of the adjective-noun phrasing knowledge. It can be seen that the value polarities are opposite for `age` and `birth date`, `score` and `rank`.

### 4.3 Results on Comparison Relation Prediction

Table 2 shows the F1-scores on the original and re-split Spider dataset. -OP, -Having, and -DAL represent `WHERE` component without `OP` module, `Group` component without `Having` module, and `ORDER BY` component without `DAL` module, respectively. The results show that both the +weighted and the +direct knowledge features are effective, they significantly improve the performances of components that have comparison relations. The results also show that the overall performances are improved, which demonstrates that adjective-noun phrasing knowledge is effective and generic for existing state-of-the-art models.

## 5 Conclusion & Future Work

In this paper, we introduce the adjective-noun phrasing knowledge as feature to improve comparison relation prediction on unseen data. Experimental results show that our apporach achieves promising performance. For future work, we will further improve the quality of phrasing knowledge or incorporate other concept-level knowledge in text-to-SQL.

## Acknowledgment

This work is supported in part by the National Natural Science Foundation of China (Grand Nos. U1636211, 61672081, 61370126).

## References


Jianpeng Cheng, Zhongyuan Wang, Ji-Rong Wen, Jun Yan, and Zheng Chen. 2015. Contextual text understanding in distributional semantic space. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 133–142. ACM.

Gerard De Melo and Mohit Bansal. 2013. Good, great, excellent: Global inference of semantic intensities. *Transactions of the Association for Computational Linguistics*, 1:279–290.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 731–742.

Catherine Finegan-Dollak, Jonathan K Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving text-to-sql evaluation methodology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 351–360.

Jonathan Herzig and Jonathan Berant. 2018. Decoupling structure and lexicon for zero-shot semantic parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1619–1629.

Oliver Lehmberg, Dominique Ritze, Robert Meusel, and Christian Bizer. 2016. A large public corpus of web tables containing time and context metadata. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 75–76. International World Wide Web Conferences Steering Committee.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Josef Ruppenhofer, Michael Wiegand, and Jasper Brandes. 2014. Comparing methods for deriving intensity scores for adjectives. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*, pages 117–122.

Raksha Sharma, Arpan Somani, Lakshya Kumar, and Pushpak Bhattacharyya. 2017. Sentiment intensity ranking among adjectives using sentiment bearing word embeddings. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 547–552.

Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Q Zhu. 2012. Probase: A probabilistic taxonomy for text understanding. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 481–492. ACM.

Xiaojun Xu, Chang Liu, and Dawn Song. 2017. Sqlnet: Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv:1711.04436*.

Navid Yaghmazadeh, Yuepeng Wang, Isil Dillig, and Thomas Dillig. 2017. Sqlizer: query synthesis from natural language. *Proceedings of the ACM on Programming Languages*, 1(OOPSLA):63.

Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. 2018a. Typesql: Knowledge-based type-aware neural text-to-sql generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 588–594.

Tao Yu, Michihiro Yasunaga, Kai Yang, Rui Zhang, Dongxu Wang, Zifan Li, and Dragomir Radev. 2018b. Syntaxsqlnet: Syntax tree networks for complex and cross-domain text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1653–1663.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 2018c. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921.

Yuchen Zhang, Panupong Pasupat, and Percy Liang. 2017. Macro grammars and holistic triggering for efficient semantic parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1214–1223.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

# Bridging the Defined and the Defining: Exploiting Implicit Lexical Semantic Relations in Definition Modeling

**Koki Washio**[*†1] **Satoshi Sekine**[†2] **Tsuneaki Kato**[*3]
[*]Department of Language and Information Sciences, The University of Tokyo
[†]RIKEN Center for Advanced Intelligence Project
{[1]kokiwashio@g.ecc, [3]kato@boz.c}.u-tokyo.ac.jp
[2]satoshi.sekine@riken.jp

## Abstract

Definition modeling includes acquiring word embeddings from dictionary definitions and generating definitions of words. While the meanings of defining words are important in dictionary definitions, it is crucial to capture the lexical semantic relations between defined words and defining words. However, thus far, the utilization of such relations has not been explored for definition modeling. In this paper, we propose definition modeling methods that use lexical semantic relations. To utilize implicit semantic relations in definitions, we use unsupervisedly obtained pattern-based word-pair embeddings that represent semantic relations of word pairs. Experimental results indicate that our methods improve the performance in learning embeddings from definitions, as well as definition generation.

## 1 Introduction

Dictionary definitions are rich resources of semantic information for both humans and machines. Recent studies on definition modeling are primarily divided into two categories. One is definition generation, in which a definition is generated for a target word from its word representation. Definition generation involves analyzing word embeddings using generated definitions (Noraset et al., 2017) and machine explanations of word meanings for human readers (Ni and Wang, 2017; Ishiwatari et al., 2019). The other is learning word embeddings from definitions to obtain semantic-oriented word representations (Tissier et al., 2017; Bosc and Vincent, 2018).

Although previous methods for encoding or decoding definitions using recurrent neural networks (RNNs) yielded promising results in definition modeling (Noraset et al., 2017; Bosc and Vincent, 2018), they did not explicitly utilize lexical semantic relations between defined words and defining words. Various lexical relations exist in definitions (Amsler, 1981), as displayed in Figure 1, where the defined word *knife* exhibits an *Is-a* relation with the defining words, *tool* and *instrument*, *Has-a* relation with *edge*, and *Used-for* relation with *cutting*. Utilizing structures of definitions about lexical semantic relations facilitates the understanding and generation of definitions.

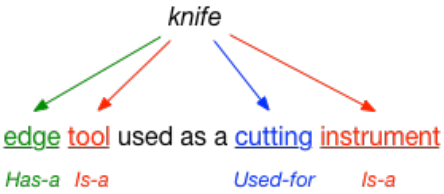


Figure 1: Definition of *knife* from WordNet (Fellbaum, 1998) and lexical semantic relations between the defined word and the defining words.

Based on this observation, we propose definition modeling methods that exploit lexical semantic relations between defined and defining words. However, lexical semantic relations in definitions are not explicit. To solve this problem, we use unsupervisedly learned word-pair embeddings that represent semantic relations of word pairs based on co-occurring relational patterns in a corpus (Turney, 2005; Washio and Kato, 2018b). Experimental results show that our definition modeling methods improve previous models, with respect to both definition generation and the acquisition of word embeddings from definitions.

## 2 Background

### 2.1 Definition Embedding

Relationships between words captured in embeddings are studied in terms of *similarity* or *relatedness* (Hill et al., 2015). For example, *coffee* and *cup* have high relatedness because *coffee* is often contained in a *cup*; meanwhile, these words ex-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3521–3527,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

hibit low similarity as *coffee* is a beverage and *cup* is a container.

Definition embeddings that are learned from word definitions are useful to capture similarity, while word embeddings based on distributional hypothesis (Mikolov et al., 2013) tend to capture relatedness (Bosc and Vincent, 2018).

Bosc and Vincent (2018) proposed a method that learns the long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) that encodes a word definition into an embedding. Given a defining word sequence $D = \{w_1, \ldots, w_T\}$, an LSTM definition encoder processes $D$, as follows:

$$\boldsymbol{h}_t = LSTM\left(\boldsymbol{h}_{t-1}, \boldsymbol{w}_t\right) \quad (1)$$

$$\boldsymbol{h} = \boldsymbol{W}_e \boldsymbol{h}_T + \boldsymbol{b}_e \quad (2)$$

where $LSTM$ computes the hidden state given the previous hidden state $\boldsymbol{h}_{t-1}$ and the input word embedding $\boldsymbol{w}_t$ along with the LSTM architecture. $\boldsymbol{h}_T$ is the final hidden state. $\boldsymbol{W}_*$ and $\boldsymbol{b}_*$ are weight matrices and bias terms, respectively. The definition encoder is trained to reconstruct a bag-of-words representation of the definition from $\boldsymbol{h}$ with a consistency penalty that renders $\boldsymbol{h}$ closer to the corresponding word embedding used in the LSTM. This method is referred to as *consistency penalized autoencoder (CPAE)*.

### 2.2 Definition Generation

Definition generation was introduced by Noraset et al. (2017). The goal of definition generation is to predict the probability of the defining word sequence $D$ given the defined word $w_{trg}$. In the aforementioned study, the LSTM conditional language model was used as a definition decoder to model this probability as follows:

$$p\left(D|w_{trg}\right) = \prod_{t=1}^{T} p\left(w_t|w_{i<t}, w_{trg}\right) \quad (3)$$

$$p\left(w_t|w_{i<t}, w_{trg}\right) = Softmax(\boldsymbol{W}_d \boldsymbol{h}'_t + \boldsymbol{b}_d) \quad (4)$$

where $\boldsymbol{h}'_t$ is a hidden state from the LSTM definition decoder. $Softmax$ is the softmax function. They conditioned the decoder by providing the embedding of the defined word at the first step of the LSTM. They referred to this model as the *Seed (S)*. Moreover, they extended this model to update the output of the recurrent unit with a gate function depending on the embedding of the defined word. This gate function, *Gate (G)*, controls the amount of information from the defined word that contributes to the definition generation at each step. As additional features, they used morphological information from a character-level convolutional neural network (CNN) to process a character sequence of the defined word and the embeddings of the hypernyms from the WebIsA database (Seitner et al., 2016). These features are called *CH* and *HE*, respectively.

Gadetsky et al. (2018) introduced context-aware definition generation to disambiguate polysemous words with their context and generate the corresponding definitions. They extended Equation 3 to consider the context word sequence of the defined word $C = \{c_1, \ldots, c_m\}$ as follows:

$$p\left(D|w_{trg}, C\right) = \prod_{t=1}^{T} p\left(w_t|w_{i<t}, w_{trg}, C\right) \quad (5)$$

To model this probability, they used an attention mechanism to extract meaningful information from $C$ and chose relevant dimensions of the embedding of the defined word. They referred to this model as *Input Attention (I-Attention)*.

### 2.3 Word-Pair Embedding

Word-pair embeddings represent relations of word pairs. Although representation of word pairs as the vector offsets of their pretrained word embeddings is a simple and powerful method (Mikolov et al., 2013), recent studies have shown that neural pattern-based word-pair embeddings are more effective than vector offsets in various tasks such as calculating relational similarity (Washio and Kato, 2018b), natural language inference, and question answering (Joshi et al., 2019).

Neural pattern-based word-pair embedding models (Washio and Kato, 2018a,b; Joshi et al., 2019) unsupervisedly learn two neural networks: a word-pair encoder and pattern encoder, both of which encode the word-pair and lexico-syntactic pattern respectively into the same embedding space. These networks are trained by predicting co-occurrences between word-pairs and patterns in a corpus with the negative sampling objective. After the unsupervised learning, the word-pair encoder provides word-pair embeddings for any word pair given their word embeddings.

## 3 Method

We propose methods that consider lexical semantic relations between the defined word and the

defining words in the definition encoder for definition embeddings (Section 3.1) and the definition decoder for definition generation (Section 3.2). To utilize the implicit semantic relations in definitions, we use word-pair embeddings that represent semantic relations of word pairs. We describe how word-pair embeddings are obtained in Section 4.1

### 3.1 For Definition Encoder

To consider lexical semantic relations in acquiring embedding from definitions, we feed the word-pair embeddings to the definition encoder. Assuming that the pair embedding $\boldsymbol{v}_{(w_{trg},w_t)}$ represents the relation between the defined word $w_{trg}$ and the $t$-th defining word $w_t$, we calculate $\boldsymbol{h}_t$ as follows:

$$\boldsymbol{h}_t = LSTM\left(\boldsymbol{h}_{t-1}, [\boldsymbol{w}_t; \boldsymbol{v}_{(w_{trg},w_t)}]\right) \quad (6)$$

where ; denotes vector concatenation. To exclude meaningless relations between the defined word and functional words, we replace $\boldsymbol{v}_{(w_{trg},w_t)}$ with the zero vector if $w_t$ is a stopword. With word-pair embeddings as inputs, the definition encoder can recognize the role of information that is provided by the current word $w_t$, for example, a type of $w_{trg}$ (*Is-a*), a goal of $w_{trg}$ (*Used-for*), or a component of $w_{trg}$ (*Has-a*).

### 3.2 For Definition Decoder

To provide the definition decoder with information regarding lexical semantic relations, we use an additional loss function with word-pair embeddings as follows:

$$L_{rel} = \frac{1}{|K|} \sum_{w_t \in K} \left\| \boldsymbol{v}_{(w_{trg},w_t)} - (\boldsymbol{W}_r \boldsymbol{h}'_t + \boldsymbol{b}_r) \right\|^2 \quad (7)$$

$$K = \{w_t | w_t \in D \wedge w_t \notin S\} \quad (8)$$

where $S$ is a set of stopwords. As in Section 3.1, we ignore the loss when $w_t$ is a stopword. This additional loss allows the definition decoder to learn the pattern of what semantic relations occur in definitions and how they occur. For example, if $w_{trg}$ indicates a type of tools, a defining word that has the *Is-a* relation to $w_{trg}$ tends to be followed by the *Used-for* word.

## 4 Experiments and Results

The experiments conducted to evaluate our methods for definition encoding and decoding are presented in this section. In the Appendix, we describe the details of the hyperparameter settings and optimization methods used in the experiments.

### 4.1 Obtaining Word-Pair Embedding

To obtain pattern-based word-pair embeddings, we extracted triples $(w_1, w_2, p) \in T$ from the Wikipedia corpus, where $(w_1, w_2)$ is a word pair composed of nouns, verbs, or adjectives in the the 100K most frequent words of the GloVe [1] (Pennington et al., 2014), and $p$ is the co-occurring shortest dependency path[2]. We discarded the triples if $p$ occurred less than five times and subsampled the triples based on word-pair probability with a threshold of $5 \cdot 10^{-7}$, following Joshi et al. (2019).

For the word-pair encoder, we used the neural networks as follows:

$$\boldsymbol{h}_{(w_1,w_2)} = MLP\left([\boldsymbol{v}_{w_1}; \boldsymbol{v}_{w_2}; \boldsymbol{v}_{w_1} \odot \boldsymbol{v}_{w_2}]\right) \quad (9)$$

where $MLP$ is a four-layer perceptron with the ReLU activation, $\boldsymbol{v}_w$ is a word embedding of $w$, and $\odot$ is the element-wise product. Each size of the hidden states of $MLP$ was 300.

The dependency path $p$ is a sequence composed of one to three lemmas and dependency relations $e_1, \ldots, e_n$. The sequence of the corresponding embeddings $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n$ was encoded using the bidirectional LSTM with the 300-dimensional hidden state as the pattern encoder. Then, the 300-dimensional pattern representation $\boldsymbol{v}_p$ was calculated with the final output vectors $\boldsymbol{h}_f$ and $\boldsymbol{h}_b$ from the forward and backward LSTM as follows:

$$\boldsymbol{v}_p = \boldsymbol{W}_p[\boldsymbol{h}_f; \boldsymbol{h}_b] + \boldsymbol{b}_p \quad (10)$$

The word embeddings in the models were initialized by the 300-dimensional GloVe. We used the multivariate negative sampling objective (Joshi et al., 2019) to train the parameters with the data $T$ for 10 epochs. Adagrad, which has a learning rate of 0.01, was used as the optimizer (Duchi et al., 2011). After the training was completed, word-pair embeddings are calculated as follows:

$$\boldsymbol{v}_{(w_1,w_2)} = \left[ \frac{\boldsymbol{h}_(w_1,w_2)}{\|\boldsymbol{h}_(w_1,w_2)\|}; \frac{\boldsymbol{h}_(w_2,w_1)}{\|\boldsymbol{h}_(w_2,w_1)\|} \right] \quad (11)$$

### 4.2 Definition Embedding

For the evaluation of definition embeddings, we used the modified Word Embedding Benchmarks projects[3], following Bosc and Vincent (2018). These benchmarks include SimLex999 (SL999),

[1] http://nlp.stanford.edu/data/glove.6B.zip
[2] We used spaCy for the dependency parsing.
[3] https://github.com/tombosc/cpae

| | Development | | Similarity | | | Relatedness | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | SV-dev | MEN-dev | SL999 | SL333 | SV-test | RG | SCWS | MEN-test | MT | WS353 |
| GloVe | 22.0 | 73.5 | 37.1 | 20.7 | 22.0 | 77.0 | 55.9 | 74.2 | 65.0 | 47.8 |
| Google | 39.3 | **76.0** | 44.2 | 29.7 | 35.8 | 76.1 | 66.0 | **75.6** | 67.1 | 63.5 |
| CPAE | 47.6 | 69.4 | 48.1 | 33.3 | 42.4 | **82.7** | 63.4 | 70.1 | 60.2 | **66.8** |
| Ours | **49.9** | 72.8 | **48.5** | **33.8** | **44.4** | 81.2 | **66.7** | 74.3 | **67.6** | 65.4 |

Table 1: Spearman's correlation coefficient $\rho \times 100$ on the benchmarks.

| **Context-agnostic (Noraset et al., 2017)** | | |
|---|---|---|
| Model | PPL | BLEU |
| S+G+CH+HE | 46.8 | 35.4 |
| w/ $L_{rel}$ | **39.5** | **37.9** |
| **Context-aware (Gadetsky et al., 2018)** | | |
| Model | PPL | BLEU |
| S+I-Attention | 59.6 | 12.0 |
| w/ $L_{rel}$ | **43.8** | **12.3** |

Table 2: Perplexity and equally-weighted BLEU scores for up to 4-grams on the definition generation datasets

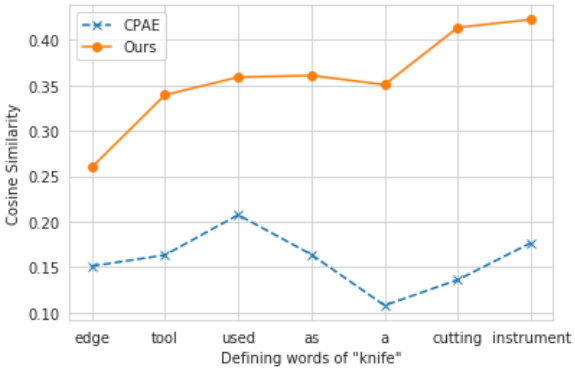


Figure 2: Cosine similarities between the last hidden states of the definition encoders encoding *kettle*'s definition and each hidden states representing the *knife*'s definition

SimLex333 (SL333) (Hill et al., 2015), SimVerb (SV) (Gerz et al., 2016), MEN (Bruni et al., 2014), RG (Rubenstein and Goodenough, 1965), WS353 (Finkelstein et al., 2002), SCWS (Huang et al., 2012), and MTurk (Radinsky et al., 2011; Halawi et al., 2012). To evaluate the definition embeddings, we scored word pairs in the benchmarks using the cosine similarity between the corresponding definition embeddings and calculated Spearman's correlation to the ground truth. The definitions in WordNet were used to train the definition encoder. The development sets of the SimVerb and MEN were used for the hyperparameter tuning.

We implemented the *CPAE* (Section 2.1) as a baseline and compared it to *CPAE* with the word-pair embeddings (Section 3.1), which is our proposed method. The word embeddings in the definition encoder were initialized by the Google Word2Vec vectors[4]. Google Word2Vec vectors and GloVe were used as the other baselines.

Table 1 shows the results of the similarity and relatedness benchmarks. While the baseline *CPAE* outperformed our model on the two out of five datasets pertaining to word relatedness, our model consistently outperformed the baseline on the similarity benchmarks. These results indicate that the word-pair embeddings provide the definition encoder with useful semantic information about the target word.

### Hidden States of Definition Encoder

To analyze the functionality of injecting semantic relations between the defined word and the defining words into the definition encoder, we investigate the similarities between hidden states of the definition encoders.

We encoded the *kettle*'s definition (*a metal pot for stewing or boiling*) and the *knife*'s definition (*edge tool used as a cutting instrument*) with the definition encoders. These two definitions are similar in their style, composed of the defining words of *Is-a* relations (*pot* and *tool*) and *Used-for* relations (*stewing*, *boiling*, and *cutting*.)

Figure 2 displays the cosine similarities between the last hidden state of the encoded *kettle*'s definition and each hidden state of the encoded *knife*'s definition. This figure shows that when *tool* with *Is-a* relation and *cutting* with *Used-for* relation to *knife* were input to the encoder, our method increased the similarities more than the baseline. This indicates that our method successfully helps the model capture the similarities of definitions in terms of lexical semantic relations.

### 4.3 Definition Generation

We evaluated our method for the definition decoder (Section 3.2) on a context-agnostic dataset (Noraset et al., 2017) and context-aware dataset (Gadetsky et al., 2018) for the definition generation. For evaluation purposes, we calculated the perplexity (PPL) and BLEU score (Papineni et al.,

[4]https://code.google.com/archive/p/word2vec/

| | S+G+CH+HE | w/ $L_{rel}$ |
|---|---|---|
| academician | a person who specializes in a particular profession | one who is versed in a scholarly or scientific field |
| artist | one who is a person who is a person or thing is made | one who creates a picture or representation of a creative work |
| adolescence | the state of being pregnant | the state of being mature |

Table 3: Examples of definitions generated by the baseline $S + G + CH + HE$ and the one with $L_{rel}$ on the development set of Noraset et al. (2017)

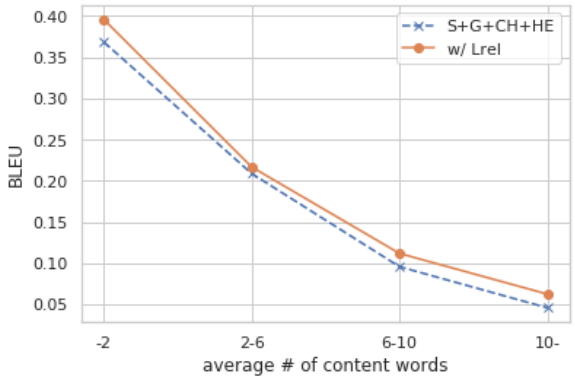


Figure 3: BLEU scores at the average numbers of content words in the reference definitions on the development set.

2002), following Noraset et al. (2017). We implemented *S+G+CH+HE* for the context-agnostic dataset and *S+I-Attention* for the context-aware dataset as baselines, and compared these models to the ones with $L_{rel}$ in Section 3.2. The word embeddings in the models were initialized by Google Word2Vec vectors, as in Section 4.2.

The results in Table 2 show that training with our $L_{rel}$ improves the performance in both the context-agnostic and context-aware settings.

### Effect to Definition Generation

To analyze the effect of $L_{rel}$, we plotted the BLEU scores at each average number of content words in reference definitions from the development set of Noraset et al. (2017), as shown in Figure 3. This figure shows that when many content words exist in reference definitions, the model with $L_{rel}$ is stronger than that without $L_{rel}$. This indicates that our method allows the model to select correct words when the defined word requires detailed descriptions.

### Generated Definitions

Table 3 displays examples of generated definitions by the baseline $S + G + CH + HE$ and the one with $L_{rel}$. The model with the proposed method successfully generated the definitions for *academician* and *artist*, while the baseline did not. Although the baseline generated the generalized class of the target words, for examples, *person* and *one*, it fails to generate details when the target word is distinct from others in the same class. In contrast, the model with the proposed method chose the correct words at both the generalized class and the details.

For *adolescence*, both models could not generate the correct definitions. Even with our method, the model produced a definition that has an opposite meaning to the target word. The generation of opposite definitions is a significant problem in the definition generation from word embeddings (Noraset et al., 2017). Although our method helps the model generate details about the target words, this problem requires other approaches that consider antonym relations.

## 5 Conclusion

In this paper, we proposed definition modeling methods that utilize lexical semantic relations between defined words and defining words. To utilize implicit semantic relations in dictionary definitions, we used pattern-based word-pair embeddings. The experimental results demonstrated that applying our methods to the definition encoder and the definition decoder improved their performance. In our future work, we will extend our methods for phrase-level definition generation (Ishiwatari et al., 2019).

## Acknowledgments

This work was supported by JSPS KAKENHI Grant numbers JP17H01831. We would like to thank Shonosuke Ishiwatari for his help with the experiments.

## References


Robert A. Amsler. 1981. A taxonomy for english nouns and verbs. In *Proceedings of the 19th Annual Meeting of the Association for Computational Linguistics*, pages 133–138, Stanford, California, USA. Association for Computational Linguistics.

Tom Bosc and Pascal Vincent. 2018. Auto-encoding dictionary definitions into consistent word embeddings. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1522–1532, Brussels, Belgium. Association for Computational Linguistics.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *Journal of Artificial Intelligence Research*, 49:1–47.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, Mass.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: The concept revisited. *ACM Transactions on information systems*, 20(1):116–131.

Artyom Gadetsky, Ilya Yakubovskiy, and Dmitry Vetrov. 2018. Conditional generators of words definitions. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 266–271, Melbourne, Australia. Association for Computational Linguistics.

Daniela Gerz, Ivan Vulić, Felix Hill, Roi Reichart, and Anna Korhonen. 2016. SimVerb-3500: A large-scale evaluation set of verb similarity. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2173–2182, Austin, Texas. Association for Computational Linguistics.

Guy Halawi, Gideon Dror, Evgeniy Gabrilovich, and Yehuda Koren. 2012. Large-scale learning of word relatedness with constraints. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1406–1414. ACM.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. SimLex-999: Evaluating semantic models with (genuine) similarity estimation. *American Journal of Computational Linguistics*, 41(4):665–695.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Eric Huang, Richard Socher, Christopher Manning, and Andrew Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 873–882, Jeju Island, Korea. Association for Computational Linguistics.

Shonosuke Ishiwatari, Hiroaki Hayashi, Naoki Yoshinaga, Graham Neubig, Shoetsu Sato, Masashi Toyoda, and Masaru Kitsuregawa. 2019. Learning to describe phrases with local and global contexts. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics.

Mandar Joshi, Eunsol Choi, Omer Levy, Daniel S. Weld, and Luke Zettlemoyer. 2019. pair2vec: Compositional word-pair embeddings for cross-sentence inference. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA. Curran Associates Inc.

Ke Ni and William Yang Wang. 2017. Learning to explain non-standard english words and phrases. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 413–417, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Thanapon Noraset, Chen Liang, Larry Birnbaum, and Doug Downey. 2017. Definition modeling: Learning to define word embeddings in natural language. In *The Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. 2011. A

word at a time: computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th international conference on World wide web*, pages 337–346. ACM.

Herbert Rubenstein and John B Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.

Julian Seitner, Christian Bizer, Kai Eckert, Stefano Faralli, Robert Meusel, Heiko Paulheim, and Simone Paolo Ponzetto. 2016. A large database of hypernymy relations extracted from the web. In *LREC*.

Julien Tissier, Christophe Gravier, and Amaury Habrard. 2017. Dict2vec : Learning word embeddings using lexical dictionaries. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 254–263, Copenhagen, Denmark. Association for Computational Linguistics.

Peter D. Turney. 2005. Measuring semantic similarity by latent relational analysis. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pages 1136–1141.

Koki Washio and Tsuneaki Kato. 2018a. Filling missing paths: Modeling co-occurrences of word pairs and dependency paths for recognizing lexical semantic relations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1123–1133, New Orleans, Louisiana. Association for Computational Linguistics.

Koki Washio and Tsuneaki Kato. 2018b. Neural latent relational analysis to capture lexical semantic relations in a vector space. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 594–600, Brussels, Belgium. Association for Computational Linguistics.

# Don't Just Scratch the Surface: Enhancing Word Representations for Korean with Hanja

**Kang Min Yoo***, **Taeuk Kim*** and **Sang-goo Lee**
Department of Computer Science and Engineering
Seoul National University, Seoul, Korea
{kangminyoo,taeuk,sglee}@europa.snu.ac.kr

## Abstract

We propose a simple yet effective approach for improving Korean word representations using additional linguistic annotation (i.e. Hanja). We employ cross-lingual transfer learning in training word representations by leveraging the fact that Hanja is closely related to Chinese. We evaluate the intrinsic quality of representations learned through our approach using the word analogy and similarity tests. In addition, we demonstrate their effectiveness on several downstream tasks, including a novel Korean news headline generation task.

## 1 Introduction

There is a strong connection between the Korean and Chinese languages due to cultural and historical reasons (Lee and Ramsey, 2011). Specifically, a set of logograms with very similar forms to the Chinese characters, called **Hanja**[1], served in the past as the only medium for written Korean until **Hangul**, the Korean alphabet, and **Jamo** came into existence in 1443 (Sohn, 2001). Considering this etymological background, a substantial portion of Korean words are classified as Sino-Korean, a set of Korean words that had originated from Chinese and can be expressed in both Hanja and Hangul (Taylor, 1997), with the latter becoming commonplace in modern Korean.

Based on these facts, we assume the introduction of Hanja-level information will aid in grounding better representation for the meaning of Korean words (e.g. see Fig. 1). To validate our hypothesis, we propose a simple yet effective approach to training Korean word representations, named **Hanja-level SISG** (Hanja-level Subword Information Skip-Gram), for capturing the semantics of Hanja and subword structures of Korean and introducing them into the vector space. Note that it is also quite intuitive for native Koreans to resolve the ambiguity of (Sino-)Korean words with the aid of Hanja. We conjecture this heuristic is attributed to the fact that Hanja characters are logograms, each of which contains more lexical meaning when compared against its counterpart, the Hangul character, which is a phonogram. Accordingly, in this work, we focus on exploiting the rich extra information from Hanja as a means of constructing better Korean word embeddings. Furthermore, our approach enables us to empirically investigate the potential of character-level cross-lingual knowledge transfer with a case study of Chinese and Hanja.

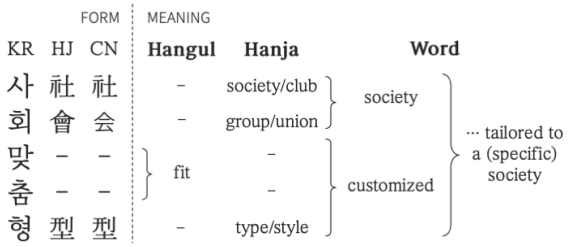


Figure 1: An example of a Korean word showing its form and multi-level meanings. The Sino-Korean word consists of Hangul phonograms (KR) and Hanja logograms (HJ). Although annotation of Hanja is optional, it offers deeper insight into the word meaning due to its association with the Chinese characters (CN).

To sum up, our contributions are threefold:

- We introduce a novel way of training Korean word embeddings with an expanded Korean alphabet vocabulary using Hanja in addition to the existing Korean characters, Hangul.
- We also explore the possibility of character-

*Equal contribution.
[1]Hanja and traditional Chinese characters are very similar but not completely identical. Some differences in strokes and language-exclusive characters exist.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3528–3533,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

level knowledge transfer between two languages by initializing Hanja embeddings with Chinese character embeddings before training skip-gram.

- We prove the effectiveness of our method in two ways, one of which is intrinsic evaluation (the word analogy and similarity tests), and the other being two downstream tasks including a newly proposed Korean news headline generation task.

## 2 Model Description

### 2.1 Skip-Gram (SG)

Given a corpus as a sequence of words $(w_1, w_2, \ldots, w_T)$, the goal of a skip-gram model (Mikolov et al., 2013) is to maximize the log-probabilities of context words given a target word $w_i$:

$$\sum_{i \in \{1,\ldots,T\}} \sum_{c \in \mathcal{C}(i)} \log p\left(w_c | w_i\right), \tag{1}$$

where $\mathcal{C}$ returns a set of context word indices given a word index. The usual choice for training the parameterized probability function $p\left(w_c|w\right)$ is to reframe the problem as a set of binary classification tasks of the target context word $c$ and other negative samples chosen from the entire dictionary (negative sampling). The objective is thus

$$L\left(s\left(w_i, w_c\right)\right) + \sum_{j \in \mathcal{N}(i,c)} L\left(-s\left(w_i, w_j\right)\right), \tag{2}$$

where $L\left(x\right) = \log\left(1 + e^{-x}\right)$, the binary logistic loss. $\mathcal{N}$ returns a set of negative sample indices given target word and context indices. In the vanilla SG model, the scoring function $s$ is the dot product between the two word vectors: $\mathbf{v}_w$ and $\mathbf{v}_c$.

### 2.2 Subword Information SG (SISG) and Jamo-level SISG

In Bojanowski et al. (2017), the scoring function incorporates subword information. Specifically, given all possible n-gram character sets $\mathbf{G} = \{g_1, \ldots, g_G\}$ in the corpus and that the n-gram set of a particular word $w$ is $\mathcal{G}_w \subset \mathbf{G}$, the scoring function is thus the sum of dot products between all n-gram vectors $\mathbf{z}$ and the context vector $\mathbf{v}_c$:

$$s\left(w, c\right) = \sum_{g \in \mathcal{G}_w} \mathbf{z}_g \mathbf{v}_c. \tag{3}$$

However, due to the deconstructible structure of Korean characters and the agglutinative nature of the language, the model proposed by Bojanowski et al. (2017) comes short in capturing sub-character and inter-character information specific to Korean understanding. As a remedy, the model proposed by Park et al. (2018) introduces jamo-level n-grams $g^{(j)}$, whose vectors can be associated with the target context as well. Given that a Korean word $w$ can be split into a set of jamo n-grams $\mathcal{G}_w^{(j)} \subset \mathbf{G}^{(j)} = \left\{g_1^{(j)}, \ldots, g_J^{(j)}\right\}$, where $\mathbf{G}^{(j)}$ is the set of all possible jamo n-grams in the corpus, the updated scoring function for jamo-level skip-gram $s^{(j)}$ is thus

$$s^{(j)}\left(w, c\right) = s\left(w, c\right) + \sum_{j \in \mathcal{G}_w^{(j)}} \mathbf{z}_j \mathbf{v}_c. \tag{4}$$

### 2.3 Hanja-level SISG[2]

Another semantic level in Korean lies in Hanja logograms. Hanja characters can be mapped to Hangul phonograms, hence Hanja annotations are sufficient but not necessary in conveying meanings (Fig. 1). As each Hanja character is associated with semantics, their presence could provide meaningful aid in capturing word-level semantics in the vector space.

We propose incorporating Hanja n-grams in the learning of word representations by allowing the scoring function to associate each Hanja n-gram with the target context word. Concretely, given that a Korean word $w$ contains a set of Hanja sequences $\mathcal{H}_w = (h_{w,1}, \ldots, h_{w,H_w})$ (e.g. for the example in Fig. 1 $h_{w,1}$ = "社會" and $h_{w_2}$ = "型") and that $\mathcal{I}_h$ is the set of Hanja n-grams for a Hanja sequence $h$ (e.g. $\mathcal{I}_{h_{w,2}}$ = { "<boh>型", "型", "型<eoh>"}), *all* Hanja n-grams $\mathcal{G}_w^{(h)}$ for word $w$ are the union of Hanja n-grams of Hanja sequences present in $w$: $\bigcup_{h \in \mathcal{H}_w} \mathcal{I}_h$. The length of Hanja n-grams in $\mathcal{I}_h$ is a hyperparameter. Then the new score function for Hanja-level SISG is

$$s^{(h)}\left(w, c\right) = s^{(j)}\left(w, c\right) + \sum_{h \in \mathcal{G}_w^{(h)}} \mathbf{z}_h \mathbf{v}_c \tag{5}$$

where $\mathbf{z}_h$ is the n-gram vector for a Hanja sequence $h$.

[2]The code and models are publicly available online at https://github.com/kaniblu/hanja-sisg

### 2.4 Character-level Knowledge Transfer

Since Hanja characters essentially share deep roots with Chinese characters, many of them can be mapped by one-to-one correspondence. By converting Korean characters into simplified Chinese characters or vice versa, it is possible to utilize pre-trained Chinese embeddings in the process of training Hanja-level SISG. In our case, we propose leveraging advances in Chinese word embeddings from the Chinese community by initializing Hanja n-grams vectors $\mathbf{z}_h$ with the state-of-the-art Chinese embeddings (Li et al., 2018) and training with the score function in Equation 5.

## 3 Experiments

### 3.1 Word Representation Learning

#### 3.1.1 Korean Corpus

For learning word representations, we utilize the corpus prepared by Park et al. (2018) with small modifications. We perform additional data cleansing (e.g. removing non-Korean sentences in the corpus and unifying number tags) to obtain a cleaner version of the corpus. All of our comparative studies are based on this corpus. For training Hanja-level SISG, we use Hanjaro[3] tagger to automatically annotate all of our datasets with Hanja. It is the state-of-the-art Hanja tagger available to the public to the best of our knowledge.

#### 3.1.2 Models

We compare our approach with three other baselines. Skip-Gram (`SG`) model (Mikolov et al., 2013) does not incorporate any n-gram information in the training loss. Subword Information Skip-Gram (`SISG(c)`) (Bojanowski et al., 2017) uses character-level n-gram information to enrich word representations. For the Korean language, character-level n-grams correspond to syllable n-grams. Jamo-level SISG (`SISG(cj)`) (Park et al., 2018) uses jamo-level n-grams in addition to character-level n-grams. Our ablation studies confirm that the hyperparameter settings (character n-grams ranging from 1 to 6 and jamo n-grams ranging from 3 to 5) proposed by the authors are indeed optimal, hence our subsequent studies all employ the same settings. Our approach is denoted by `SISG(cjh)`). As for the specific hyperparameter settings of our approach, `SISG(cjh3)` denotes Hanja n-grams ranging from 1 to 3 while `SISG(cjh4)` denotes Hanja n-grams ranging from 1 to 4. `SISG(cjhr)` is a special version of our approach, where Hanja n-grams are randomly initialized as opposed to being pre-trained (Section 2.4). The details of our ablation studies and the model implementation are included in the supplementary material.

[3] http://hanjaro.juntong.or.kr/

| Method | Analogy | | | Similarity | |
|---|---|---|---|---|---|
| | Sem. | Syn. | All | Pr. | Sp. |
| `SISG(cj)`† | *0.47* | *0.38* | *0.43* | - | *0.67* |
| `SG` | 0.42 | 0.49 | 0.45 | 0.60 | 0.62 |
| `SISG(c)` | 0.45 | 0.59 | 0.52 | 0.62 | 0.61 |
| `SISG(cj)` | 0.39 | 0.48 | 0.44 | **0.66** | 0.67 |
| `SISG(cj)`‡ | 0.41 | 0.48 | 0.45 | 0.65 | **0.67** |
| `SISG(cjh3)` | **0.34** | **0.45** | **0.39** | 0.63 | 0.63 |
| `SISG(cjh4)` | 0.34 | 0.45 | 0.40 | 0.62 | 0.61 |
| `SISG(cjhr)` | 0.35 | 0.46 | 0.40 | 0.65 | 0.64 |

† reported by Park et al. (2018)
‡ pre-trained embeddings provided by the authors of Park et al. (2018) run with our evaluation script

Table 1: Word analogy and similarity evaluation results. For word analogy test, the table shows cosine distances averaged over semantic and syntactic word pairs (lower is better). For word similarity test, we report on Pearson and Spearman correlations between predictions and ground truth scores. We observe that our approach (SISG(cjh)) shows significant improvements in the analogy test but some deterioration in the similarity test. Note that the evaluation results reported by the original authors (†) are largely different from our results. This might be due to differences in implementation details, hence we report and compare with the results of the authors' embeddings run on our test script (‡).

### 3.2 Intrinsic Evaluation

#### 3.2.1 Word Analogy Test

Following the experimental protocol of Park et al. (2018), given an analogy pair $a : b \leftrightarrow c : d$, we compute the cosine distance between the word vectors $\mathbf{v}_a + \mathbf{v}_b - \mathbf{v}_c$ and $\mathbf{v}_d$ using the following equation: $1 - \cos(x, y)$, where cos is the cosine similarity. The Korean word analogy dataset (Park et al., 2018) consists of 10,000 quadruples, designed with a set of semantic and syntactic features specific to the Korean. We report our findings in Table 1. We observe that our approach improves significantly compared to the previous state-of-the-art models in both semantic and syntactic word analogy detection.

#### 3.2.2 Word Similarity Test

The word similarity test aims to evaluate the correlation between word vector distances and

human-annotated scores. We use the Korean WS353 dataset provided by Park et al. (2018). Our results are shown in Table 1. Although our approaches do not outperform the jamo-level baseline (SISG(cj)), we observe that Hanja-level information provides some advantage compared to just using pure character-level n-grams (SISG(c)). Also note that WS353 is a relatively small dataset, which could be prone to statistical bias.

#### 3.2.3 Analysis

How much of the improvement can be explained by transfer learning from Chinese? Word analogy test results show that word representations trained with pre-trained Chinese n-grams perform better than those trained without (SISG(cjhr)), supporting our claim that our approach is able to transfer relevant knowledge from the Chinese language for detecting analogical relationships. However, for the word similarity test, word vectors trained without Chinese embeddings perform better, suggesting that there are some trade-offs.

### 3.3 Downstream Tasks

In this section, we try to demonstrate the effectiveness of our approach to downstream tasks with two specific cases.

#### 3.3.1 Korean News Headline Generation

To show that our approach helps supervised models in gaining deeper understanding of Korean texts, we devise a new task using news articles, which requires an understanding of the semantics of Korean texts. We collected Korean news articles published from 2017 January to February. The dataset covers balanced categories (e.g. politics, sports, world, etc.) and contains the headline title as the annotation, akin to the CNN/Daily Mail Dataset (Hermann et al., 2015). The dataset is relatively large - containing 840,205 news article and title pairs

We use the encoder-decoder architecture supported by OpenNMT (Klein et al., 2017) framework for the task, where the encoder is a bidirectional LSTM cell and the decoder is an LSTM cell (the hidden size is 512 for both) with a bridging feed-forward layer between the two RNNs. We employ soft attention (Bahdanau et al., 2014) to generate news headlines given the first three sentences of an article body. We tokenize each headline using a Korean morpheme tokenizer[4] such that decoder tokens are relatively dense. The copy mechanism (Gu et al., 2016), a popular technique of sequence-to-sequence models for translation and summarization, cannot be applied directly to our model, as the encoder tokens (space tokenized) and the decoder tokens (morpheme tokenized) do not share the same token space. Word vectors obtained from both baselines and our approach are used to initialize the encoder embeddings before the training begins. The dataset is split into training, validation, and test sets (8:1:1). We report performances on the test set after validating our training models on the validation set. We evaluate the titles generated by our models with the BLEU (Papineni et al., 2002)[5]. We also report per-word perplexity obtained from the models.

| Embeddings | BLEU | | | | PPL |
|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | |
| None | 26.02 | 7.76 | 3.08 | 1.38 | 5.335 |
| SG | 30.33 | 10.20 | 4.29 | 1.98 | 4.122 |
| SISG(c) | 31.34 | 10.96 | 4.69 | 2.19 | 3.942 |
| SISG(cj) | 31.78 | 11.17 | 4.80 | 2.25 | 3.938 |
| SISG(cj)† | 31.77 | 11.16 | 4.81 | 2.27 | 3.940 |
| SISG(cjh3) | **32.03** | 11.25 | 4.83 | 2.27 | 3.941 |
| SISG(cjh4) | 32.02 | **11.34** | **4.92** | **2.30** | **3.909** |

† pre-trained embeddings by Park et al. (2018)

Table 2: Korean News Headline Generation results.

As shown in Table 2, models trained with our embeddings perform better and have better language modeling capability than those trained with the previous state-of-the-art embeddings. This improvement can be partially explained by the fact that formal Korean texts, such as news articles, are more likely to contain Sino-Korean words, allowing models with awareness of Chinese to gain an edge in Korean understanding.

#### 3.3.2 Sentiment Analysis

To estimate the performance of our approach on a different task then the previous experiment, we conduct an experiment on Naver Sentiment Movie Corpus (NSMC)[6]. This dataset consists of 200K movie reviews, each of which is labeled with its sentiment, i.e. positive or negative. We split the corpus into training (100K), validation (50K), and test sets (50K). It is worth noting that this

[4] https://github.com/shin285/KOMORAN

[5] Although ROUGE is a more widely adopted measure in English literature, no equivalent exists for Korean.

[6] https://github.com/e9t/nsmc

| Embeddings | Model: LSTM | | | |
|---|---|---|---|---|
| | Acc. | P | R | F1 |
| SISG(c) | 77.43 | 75.89 | 80.41 | 78.08 |
| SISG(cj) | **83.16** | 82.36 | **84.66** | **83.50** |
| SISG(cjh3) | 81.61 | 81.23 | 82.28 | 81.75 |
| SISG(cjh4) | 82.25 | **82.57** | 81.77 | 82.17 |

Table 3: Naver Sentiment Movie Corpus results. We report accuracy (Acc.), precision (P), recall (R), and F1-score (F1) following Park et al. (2018). Each reported result is the average of several runs initialized with different random seeds.

case study is designed to figure out to what extent our embeddings are generally applicable for even cases, where most of the sentences in the dataset consist of spoken words rather than written words and often do not contain Sino-Korean words.

We utilize a basic LSTM (Hochreiter and Schmidhuber, 1997) module as a sentence encoder to exclude possible exceptional gains from sophisticated encoders, concentrating on the usefulness of input word embeddings. Likewise the previous experiment, word vectors obtained from both baselines and our approach are used as input for the encoder. We regard the last hidden state of the LSTM as the sentence representation, which is consumed by a feed-forward network followed by a softmax classifier. The dimension of the LSTM cells is fixed as 300. Each result from the baselines and our models is the average of 3 independent runs initialized with different random seeds, and the outcome of each run is chosen by the performance (F1-score) on the validation set.

From Table 3, we confirm our approach is comparable to the best previous one and general enough to be employed in most of the downstream tasks, even though the performance of our approach is somewhat unsatisfactory. While not strictly proven, we conjecture one possible reason for the unsatisfying performance is the low reliability of the Hanja tagger we leveraged, which is not guaranteed to work well with spoken language. Specifically, we have manually inspected the tagging results and observed that there are some errors which can lead to performance degradation. This observation points out a limitation of our current approach, that is, the dependence on external taggers, encouraging us to develop an integrated approach, leaving as future work not resorting to the taggers.

## 4 Related Work

Although the word is usually regarded as the smallest and basic unit for most NLP pipelines, there is a recent trend of utilizing subword information for enriching word representations (Sennrich et al. 2016; Bojanowski et al. 2017, to name a few), or considering subwords themselves as input directly for NLP models (Zhang et al., 2015; Kim et al., 2016; Peters et al., 2018; Devlin et al., 2018).

As Korean is agglutinative (Song, 2006), the current literature in Korean word representations mainly focus on subword structures such as morphemes (Edmiston and Stratos, 2018), syllables (Choi et al., 2017) and Jamo (Choi et al., 2016; Stratos, 2017; Park et al., 2018). We here move forward one step further by incorporating Hanja information explicitly together with the aforementioned subword information.

On the other hand, cross-lingual representations is a trending topic in literature (Lample et al. 2018; Conneau et al. 2018; Lample and Conneau 2019, to name a few). Nevertheless, to the best of our knowledge, our work is the first to introduce character-level cross-lingual transfer learning based on etymological grounds. Furthermore, the novelty of our work lies of the fact that we use separated vocabulary instead of shared vocabulary such as Byte Pair Encoding (BPE) (Sennrich et al., 2016).

## 5 Conclusion

We have presented a method of training Korean word representations with Hanja. In our extensive experiments, we have demonstrated that our approach is effective in infusing more semantics into Korean word embeddings. One potential issue with our method, as already mentioned, is that it relies on external Hanja annotation, even though this can be mitigated to some extent by off-the-shelf taggers, as in this work. Thus, it would be an attractive direction to take as future work developing an end-to-end system.

## Acknowledgments

We thank Daniel Edmiston and anonymous reviewers for their helpful feedback. This work was supported by BK21 Plus for Pioneers in Innovative Computing (Dept. of Computer Science and Engineering, SNU) funded by the National Research Foundation of Korea (NRF) (21A20151113068).

## References


Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the ACL*, 5:135–146.

Jihun Choi, Jonghem Youn, and Sang-goo Lee. 2016. A grapheme-level approach for constructing a korean morphological analyzer without linguistic knowledge. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 3872–3879. IEEE.

Sanghyuk Choi, Taeuk Kim, Jinseok Seol, and Sang-goo Lee. 2017. A syllable-based technique for word embeddings of korean words. In *Proceedings of the First Workshop on Subword and Character Level Models in NLP*, pages 36–40.

Alexis Conneau, Guillaume Lample, Ruty Rinott, Adina Williams, Samuel R Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. Xnli: Evaluating cross-lingual sentence representations. *arXiv preprint arXiv:1809.05053*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Daniel Edmiston and Karl Stratos. 2018. Compositional morpheme embeddings with affixes as functions and stems as arguments. In *Proceedings of the Workshop on the Relevance of Linguistic Structure in Neural Architectures for NLP*, pages 1–5.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proc. ACL*.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*.

Guillaume Lample, Alexis Conneau, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. In *ICLR*.

Ki-Moon Lee and S Robert Ramsey. 2011. *A history of the Korean language*. Cambridge University Press.

Shen Li, Zhe Zhao, Renfen Hu, Wensi Li, Tao Liu, and Xiaoyong Du. 2018. Analogical reasoning on chinese morphological and semantic relations. In *Proceedings of the 56th Annual Meeting of the ACL)*, volume 2, pages 138–143.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on ACL*, pages 311–318. Association for Computational Linguistics.

Sungjoon Park, Jeongmin Byun, Sion Baek, Yongseok Cho, and Alice Oh. 2018. Subword-level word vector representations for korean. In *Proceedings of the 56th Annual Meeting of the ACL*, volume 1, pages 2429–2438.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the NAACL: HLT, Volume 1 (Long Papers)*, volume 1, pages 2227–2237.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of ACL*, volume 1, pages 1715–1725.

Ho-Min Sohn. 2001. *The Korean Language*. Cambridge University Press.

Jae Jung Song. 2006. *The Korean language: Structure, use and context*. Routledge.

Karl Stratos. 2017. A sub-character architecture for korean language processing. In *Proceedings of the 2017 Conference on EMNLP*, pages 721–726.

Insup Taylor. 1997. Psycholinguistic reasons for keeping chinese characters in korean and japanese. *Cognitive processing of Chinese and related Asian languages*, 319.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

# SyntagNet: Challenging Supervised Word Sense Disambiguation with Lexical-Semantic Combinations

**Marco Maru**[1,2], **Federico Scozzafava**[1], **Federico Martelli**[1,3], and **Roberto Navigli**[1]

[1]Department of Computer Science, Sapienza University of Rome
[2]Department of Literature and Modern Cultures, Sapienza University of Rome
[3]Babelscape srl, Italy
firstname.lastname@uniroma1.it, martelli@babelscape.com

## Abstract

Current research in knowledge-based Word Sense Disambiguation (WSD) indicates that performances depend heavily on the Lexical Knowledge Base (LKB) employed. This paper introduces SyntagNet, a novel resource consisting of manually disambiguated lexical-semantic combinations. By capturing sense distinctions evoked by syntagmatic relations, SyntagNet enables knowledge-based WSD systems to establish a new state of the art which challenges the hitherto unrivaled performances attained by supervised approaches. To the best of our knowledge, SyntagNet is the first large-scale manually-curated resource of this kind made available to the community (at http://syntagnet.org).

## 1 Introduction

Word Sense Disambiguation (WSD) is one of the most challenging Natural Language Processing (NLP) tasks. It deals with lexical ambiguity, and it is core to achieving the much-sought-for goal of Natural Language Understanding (Navigli, 2018). In a broad sense, two major approaches can be adopted for performing WSD: the supervised and the knowledge-based ones.

Supervised methods, which learn a classifier from training data, have improved from 65% (Snyder and Palmer, 2004) to over 71% accuracy, thus proving to perform best for WSD purposes (Yuan et al., 2016; Melacci et al., 2018; Uslu et al., 2018). However, supervision depends heavily on large quantities of reliable sense-annotated data which – especially for languages other than English – are poorly available.

As a major alternative to supervised WSD, knowledge-based approaches drop the requirement for large amounts of training data by drawing on rich Lexical Knowledge Bases (LKB) such as WordNet (Fellbaum, 1998), and allow scaling to multiple languages thanks to multilingual resources such as BabelNet (Navigli and Ponzetto, 2012). The structure of an LKB plays a key role in increasing the overall disambiguation performance. For the purposes of WSD, a critical consideration concerns the nature of the relations connecting concepts: while LKBs tend to focus on the paradigmatic dimension of language, such resources fall short in respect of syntagmatic relations[1], which are also crucial for sense disambiguation due to interconnecting co-occurring words (Navigli and Lapata, 2010).

In this paper we address this deficiency and present, for the first time, a manually-curated large-scale lexical-semantic combination[2] database which associates pairs of concepts with pairs of co-occurring words. Importantly, we prove the effectiveness of our resource by achieving the state of the art in multilingual knowledge-based WSD and by matching supervised WSD performances when integrated into an LKB made up of WordNet and the Princeton WordNet Gloss Corpus[3].

## 2 Related work

Several studies on knowledge-based algorithms have indicated that the LKB structure is of vital importance in determining the accuracy of sense disambiguation. In particular, it has been demonstrated that WSD performance improves

[1]A syntagmatic relation exists between two words which co-occur in spoken or written language more frequently than would be expected by chance and which have different grammatical roles in the sentences in which they occur (Saussure, 1916).

[2]Here, we will use the term "lexical combination" to refer to both lexical collocations and free word associations (without considering idiomatic expressions) and the term "lexical-semantic combination" to refer to sense-annotated lexical combinations.

[3]http://wordnetcode.princeton.edu/glosstag.shtml



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3534–3540, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

dramatically when employing an LKB with a larger number of high-quality lexical-semantic relations, i.e., more connections between concepts (Boyd-Graber et al., 2006; Lemnitzer et al., 2008; Ponzetto and Navigli, 2010). During the last two decades, a certain amount of work has been carried out aimed at enriching LKBs with new lexical-semantic relations. To this end, knowledge has been (semi-)automatically extracted from large collections of data and integrated into lexical resources such as WordNet.

As far as semi-automatic approaches are concerned, Mihalcea and Moldovan (2001) conceived eXtended WordNet, a resource providing disambiguated glosses by means of a classification ensemble combined with human supervision. A set of manually disambiguated glosses, called the Princeton WordNet Gloss Corpus (PWNG), which inherently included syntagmatic content, was subsequently also made available in 2008.

The rationale behind the creation of such resources was substantiated in a knowledge-based WSD study conducted by Navigli and Lapata (2010), who hypothesized an improvement in performance by several points when enriching a semantic network with tens of lexical-semantic relations for each target word sense. To achieve this demanding goal, endeavors in the literature focused on the fully-automatic production of semantic combinations, such as those obtained by disambiguating topic signatures (Cuadros and Rigau, 2008; Cuadros et al., 2012, KnowNet and deepKnowNet) or by disentangling the concepts in ConceptNet (Chen and Liu, 2011).

More recently, Espinosa-Anke et al. (2016) aimed at automatically enriching WordNet with collocational information by leveraging the relations between sense-level embedding spaces (ColWordNet), while Simov et al. (2016) addressed the enhancement of LKBs by exploiting relations over semantically-annotated corpora as contextual information. To the same end, Simov et al. (2018) employed grammatical role embeddings to gather new syntagmatic relations.

The lack of syntagmatic information in semantic networks was also tackled by the extension of a lexical database by means of *phrasets*, i.e., sets of free combinations of words recurrently used to express a concept (Bentivogli and Pianta, 2004).

Unfortunately, due to their (semi-)automatic nature, the aforementioned resources could not inherently offer wide coverage and high precision at the same time. Compared to other resources geared towards knowledge-based WSD, the novel resource we contribute in this work features: (i) wide coverage with a broad spectrum of possible lexical combinations, and (ii) high precision thanks to being entirely manually curated.

## 3 SyntagNet: a wide-coverage lexical-semantic combination resource

In this Section, we present SyntagNet, a knowledge resource created starting from lexical combinations extracted from the English Wikipedia[4] and the British National Corpus (Leech, 1992, BNC), and manually disambiguated according to the WordNet 3.0 sense inventory.

### 3.1 Methodology

**Lexical combination extraction** First of all, we employed the Stanford CoreNLP pipeline (Manning et al., 2014) to extract the dependency trees[5] for all the sentences in both Wikipedia and the BNC. Then, in order to identify relevant combinations, we determined the strength of correlation between pairs of POS-tagged, lemmatized content words[6] $w_1$, $w_2$, co-occurring within a sliding window of 3 words. Each candidate pair $(w_1, w_2)$ was weighted using Dice's coefficient multiplied by a logarithmic factor of the co-occurrence frequency:

$$score(w_1, w_2) = log_2(1 + n_{w_1 w_2}) \frac{2n_{w_1 w_2}}{n_{w_1} + n_{w_2}} \quad (1)$$

where $n_{w_i}$ ($i \in \{1, 2\}$) is the frequency of $w_i$ and $n_{w_1 w_2}$ is the frequency of the two words co-occurring within a window.

Three filters were then applied in order to slim down the list of pairs: (i) we filtered out English stopwords according to the Natural Language Toolkit (Loper and Bird, 2002, NLTK 3.4); (ii) we discarded combinations between verbs and verbs; (iii) we discarded combinations not linked by any of the five most frequent dependencies in our list, namely: `compound`, `dobj` (direct object), `iobj` (indirect object), `nsubj` (nominal subject) and `nmod` (nominal modifier).

Finally, we ranked the resulting lexical combination list according to the geometric mean

[4] November 2018 English Wikipedia dump.

[5] According to the Universal Dependencies v2 (https://universaldependencies.org/u/dep/all.html).

[6] Restricted to nouns and verbs in the WordNet dictionary.

| word 1 | word 2 | score | sense 1 | sense 2 |
|---|---|---|---|---|
| $\text{run}_v$ | $\text{program}_n$ | 18.07 | $\text{run}_v^{19}$ (carry out a process or program) | $\text{program}_n^{7}$ (a sequence of instructions) |
| $\text{run}_v$ | $\text{race}_n$ | 11.55 | $\text{run}_v^{37}$ (compete in a race) | $\text{race}_n^{2}$ (a contest of speed) |
| $\text{run}_v$ | $\text{farm}_n$ | 3.50 | $\text{run}_v^{4}$ (direct or control) | $\text{farm}_n^{1}$ (workplace with farm buildings) |

Table 1: Examples of high-ranking lexical (left) and semantic (right) combinations, where each lemma's subscript and superscript indicate its part of speech and sense number, respectively, in WordNet.

between i) the logarithmic Dice scores and ii) the frequency count of a pair in a given POS tag/dependency combination. We show some examples with $w_1$ = $\text{run}_v$, together with their final correlation score in Table 1 (left).

We then repeated the whole process described above, with the following changes: i) we set a sliding window of 6 words; ii) we removed the constraint on the dependency selection; iii) we filtered out all pairs already occurring within the first list; iv) we selected only items attested in multiple English monolingual and collocation dictionaries.

**Manual disambiguation** We asked eight annotators to manually disambiguate the top-ranking $20,000$ lexical combinations from the first list and $58,000$ lexical combinations from the second list, i.e., to associate each word in a pair $(w_1, w_2)$ with its most appropriate senses in WordNet (in Table 1 (right) we show the senses chosen by the annotators for the corresponding lexical combinations).

The eight annotators shared a background in linguistics (Master's Degree with a minimum C1 English proficiency level) and were well acquainted with WordNet. In order to facilitate the annotation process, we provided each annotator with a unique batch of lexical combinations in a simple interface; for each pair, the annotators visualized all the synsets for each word of the combination (along with WordNet definitions and examples), and a context of up to 25 random sentences in which the combination was extracted. The annotators were asked to input the sense numbers associated with their chosen synsets for both the words in a given pair. Since the combinations can carry different meanings depending on the context, the annotators were allowed to assign multiple senses to the same word in a given combination (e.g., judge in the "public official" sense vs. the "evaluator" sense in the ($\text{judge}_n$, $\text{decide}_v$) lexical combination).

As a further measure to ensure quality, the annotators were also asked to skip the annotation of lexical combinations (i) carrying mistakes due to the automatic parsing process, (ii) for which none of the available senses in WordNet would fit the context, (iii) reflecting idiomatic expressions, (iv) which were multi-word Named Entities.

Overall, the annotators covered $78,000$ lexical combinations ($10,218$ unique nouns and $3,786$ unique verbs for $52,432$ noun-verb relations and $25,568$ noun-noun relations), and obtained $88,019$ semantic combinations ($61,249$ noun-verb and $26,770$ noun-noun semantic relations) linking $20,626$ WordNet 3.0 nodes, i.e., unique synsets ($14,204$ noun synsets and $6,422$ verb synsets), with a relation edge.

We periodically timed the annotators by considering the number of annotations produced on a daily basis, obtaining an average value of 42 disambiguated combinations per hour (1 minute and 26 seconds per word pair). Overall, the annotation process took a period of 9 months.

To determine the reliability of the annotations, we calculated the minimum inter-annotator agreement between pairs of annotators on a random sample of 500 combinations. For each of the 500 lexical combinations used to compute the inter-annotator agreement, the annotators were exceptionally asked to disambiguate the two target words in all of the 25 sentences provided, thus leading to a figure of 25,000 single instances disambiguated per annotator, resulting in a substantial agreement ($\kappa = 0.71$). Moreover, we found that most of the disagreement instances arose out of valid alternative tags, rather than factual errors, due to the fine granularity of the WordNet sense inventory.

## 4 Experimental setup

We now present the setup of our evaluation, carried out to assess the effectiveness of SyntagNet when employed for knowledge-based WSD.

**Disambiguation algorithm** We performed our experiments employing UKB[7] (Agirre et al., 2014), a state-of-the-art system for knowledge-based WSD, which applies the Personalized Page Rank (PPR) algorithm (Haveliwala, 2002) to an

[7] Version 3.2 (http://ixa2.si.ehu.es/ukb/)

| | | English | | | | | | Multilingual | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | SemEval-13 | | | | SemEval-15 | | |
| **resource** | **#relations** | **Sens2** | **Sens3** | **Sem07** | **Sem13** | **Sem15** | **All** | **IT** | **ES** | **DE** | **FR** | **IT** | **ES** | **All** |
| WNG (WordNet+PWNG) | 671,779 | 69.2 | 65.9 | 54.9 | 66.8 | 70.7 | 67.1 | 71.4 | 71.2 | 68.0 | 69.6 | 62.2 | 58.1 | 67.2 |
| WNG+KnowNet20 | 520,682 | 67.2 | 65.8 | 53.8 | 67.3 | 71.5 | 66.6 | 71.6 | 73.1 | 68.3 | 70.4 | 61.4 | 59.9 | 67.9 |
| WNG+deepKnowNet95d | 522,880 | 66.9 | 64.9 | 53.6 | 66.9 | 71.6 | 66.2 | 71.4 | 71.9 | 67.7 | 70.5 | 62.4 | 58.7 | 67.5 |
| WNG+BabelNet 4.0 | 9,447,341 | 67.5 | 64.1 | 53.0 | 67.6 | 66.9 | 65.6 | 73.8 | 71.6 | **69.9** | 67.1 | 62.4 | 57.8 | 67.6 |
| WNG+eXtended WordNet | 551,551 | 67.7 | 65.7 | 52.3 | 67.6 | 71.0 | 66.7 | 72.4 | 71.8 | 68.5 | 69.3 | 62.4 | 58.9 | 67.7 |
| WNG+ColWordNet | 8,424 | 69.2 | 65.9 | 54.1 | 66.7 | 70.7 | 67.1 | 71.4 | 71.0 | 68.0 | 69.3 | 61.9 | 57.8 | 67.0 |
| WNG+SyntagNet | 88,019 | **71.2** | **71.6** | **59.6** | **72.4** | **75.6** | **71.5** | **74.2** | **73.4** | 66.9 | **72.7** | **65.0** | **61.2** | **69.3** |

Table 2: F1 scores (%) for English all-words fine-grained WSD (left) and for multilingual all-words fine-grained WSD (right). Each row displays results scored by a specific resource combined with the WNG (WordNet+PWNG) baseline. Statistically-significant differences, according to a $\chi^2$ test ($p < 0.01$), compared to the baseline (first row), are underlined. The second column reports the number of relations of the added resource.

input LKB. We used its *PPRw2w* single-sentence context disambiguation method, which initializes the PPR vector using the context of the target word in a given sentence, while excluding the contribution of the target word itself.

**Evaluation benchmarks and measures** We used five test sets standardized with WordNet 3.0 (Raganato et al., 2017a) including the English all-words tasks from Senseval-2 (Edmonds and Cotton, 2001), Senseval-3 (Snyder and Palmer, 2004), SemEval-2007 (Pradhan et al., 2007), SemEval-2013 (Navigli et al., 2013) and SemEval-2015 (Moro and Navigli, 2015). To run experiments on multilingual WSD, we used the last two of the foregoing datasets, which also include German, Spanish, French and Italian, employing, as sense inventory, the synset lexicalizations provided in BabelNet 4.0[8]. As customary, we computed precision, recall and F1, which in our case coincided, due to UKB always outputting a sense for each target word.

**LKBs** For the purposes of our evaluation we measured the performance obtained with UKB when combined with different LKBs. As our baseline we used WordNet + PWNG, which is the best configuration of UKB according to its authors. We also evaluated the following LKBs when integrated separately on top of our baseline: (i) the best configurations of KnowNet[9] and deepKnowNet[10], (ii) the subgraph of BabelNet 4.0 induced by WordNet 3.0, (iii) eXtended WordNet[11], (iv) ColWordNet[12], (v) SyntagNet (cf. Section 2).

[8] http://babelnet.org

[9] http://adimen.si.ehu.es/web/KnowNet

[10] http://adimen.si.ehu.es/web/deepKnowNet

[11] http://www.hlt.utdallas.edu/~xwn/

[12] http://bitbucket.org/luisespinosa/cwn/

All of the aforementioned LKBs (whose size in terms of lexical-semantic relations is shown in Table 2 (second column)) are available for download.

## 5 Experimental results

**English WSD** As shown in Table 2 (left), SyntagNet enabled UKB to achieve the best results in the English all-words disambiguation tasks, attaining 4.4 overall points above the WNG baseline, which is the only statistically-significant improvement across LKBs. Furthermore, results for the individual datasets exhibit statistically-significant improvements over the baseline on two out of five datasets. We attribute this result to the fully manual nature of SyntagNet, in contrast to the noisy character of the other LKBs. A further justification of our results comes from an analysis we performed on relation samples from the various LKBs: we collected 500 random relations for each LKB we experimented with, and manually tagged each of them as syntagmatic or paradigmatic, revealing that their syntagmatic contribution ranges from 39% (deepKnowNet) to 54% (eXtended WordNet). The fully syntagmatic nature of SyntagNet, instead, effectively blends in with the complementary information available in the baseline (63% of the relations in WNG are paradigmatic).

Table 3 compares UKB + SyntagNet against the best supervised English WSD systems (Yuan et al., 2016; Melacci et al., 2018; Uslu et al., 2018): none of the differences across datasets between the best performing supervised system and SyntagNet is statistically significant according to a $\chi^2$ test ($p < 0.01$), meaning that SyntagNet enables knowledge-based WSD to rival current supervised approaches.

| system | Sens2 | Sens3 | Sem07 | Sem13 | Sem15 | All |
|---|---|---|---|---|---|---|
| LSTMLP• | **73.8** | 71.8 | **63.5** | 69.5 | 72.6 | **71.5** |
| IMSC2V$_{+PR}$∞ | **73.8** | 71.9 | 63.3 | 68.2 | 72.8 | 71.2 |
| fastSense△ | 73.5 | **73.5** | 62.4 | 66.2 | 73.2 | 71.1 |
| UKB+SyntagNet | 71.2 | 71.6 | 59.6 | **72.4** | **75.6** | **71.5** |

Table 3: F1 scores (%) of UKB+SyntagNet against the best supervised systems for English all-words WSD. Reported systems: • Yuan et al. (2016), ∞ Melacci et al. (2018), △ Uslu et al. (2018). Statistically-significant differences against our results are underlined according to a $\chi^2$ test, $p < 0.01$.

| | SemEval-13 | | | | SemEval-15 | | |
|---|---|---|---|---|---|---|---|
| system | IT | ES | DE | FR | IT | ES | All |
| BLSTM† | 62.0 | 66.4 | **69.2** | 55.5 | - | - | - |
| UMCC-DLSI⋆ | 65.8 | 71.0 | 62.1 | 60.5 | - | - | - |
| T-O-M○ | 68.2 | 66.9 | 63.2 | 60.5 | - | - | - |
| SUDOKU-RUN1◇ | - | - | - | - | 59.9 | 56.0 | - |
| SUDOKU-RUN2◇ | - | - | - | - | 56.9 | 57.1 | - |
| Best system‡ | 68.2 | 71.0 | **69.2** | 60.5 | 59.9 | 57.1 | 64.7 |
| UKB+SyntagNet | **74.2** | **73.4** | 66.9 | **72.7** | **65.0** | **61.2** | **69.3** |

Table 4: F1 scores (%) of UKB+SyntagNet against the best systems for multilingual all-words WSD. Reported systems: † Raganato et al. (2017b), ⋆ Gutiérrez Vázquez et al. (2010), ○ Pasini and Navigli (2017), ◇ Manion (2015), ‡ result obtained by aggregating the outputs of the best systems for each dataset. Statistically-significant differences against our results are underlined according to a $\chi^2$ test, $p < 0.01$.

**Multilingual WSD** As regards our multilingual evaluation, SyntagNet enabled UKB to attain the best overall result (see Table 2 (right)), which is a statistically-significant improvement of 2.1 points over the baseline. With respect to the comparison against the best systems (Table 4), SyntagNet provides a statistically-relevant boost of 4.6 points in relation to the aggregate score of the compared systems (second to last row), attaining state-of-the-art results on five out of the six datasets taken into account.

### 5.1 Impact of LKB size

Finally, we graphed the increase in WSD performance obtained when progressively enriching the baseline UKB graph with random samples of 10,000 SyntagNet relations at each step. As illustrated in Figure 1, the improvements in the English and multilingual settings, respectively, present a growing trend according to a linear regression analysis of the data. This demonstrates that our relations are high-quality and effective for WSD, while leaving room for further improvement as more relations are added in the future.

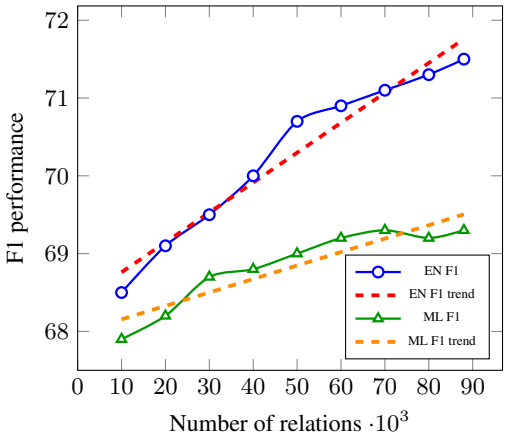


Figure 1: Impact of growing samples of SyntagNet over the baseline on overall F1 for English (EN F1 line) and multilingual (ML F1 line) evaluation datasets.

## 6 Conclusions

In this paper we put forward two main contributions: 1) we presented SyntagNet (http://syntagnet.org), a new wide-coverage, manually-curated resource of lexical-semantic combinations; 2) we showed that SyntagNet enables state-of-the-art knowledge-based WSD, rivaling the best supervised system on English and surpassing the overall performance of the best multilingual systems by 4.6 points. As future work, we plan to: i) enrich SyntagNet with more combinations, so as to surpass supervised English WSD; ii) include information from adjectives; iii) establish a common evaluation framework to compare the contribution of lexical-semantic combination resources; iv) employ and assess SyntagNet in other NLP tasks, such as word and sense similarity (Navigli and Martelli, 2019) or Semantic Role Labeling (where a newly released WordNet-linked resource, VerbAtlas (Di Fabio et al., 2019), would greatly benefit from collocational information).

## Acknowledgments

The authors gratefully acknowledge the support of the ERC Consolidator Grant MOUSSE No. 726487 and the ELEXIS project No. 731015 under the European Union's Horizon 2020 research and innovation programme.

This work was supported in part by the MIUR under grant "Dipartimenti di eccellenza 2018-2022" of the Department of Computer Science of Sapienza University.

## References


Eneko Agirre, Oier Lopez de Lacalle, and Aitor Soroa. 2014. Random Walks for Knowledge-Based Word Sense Disambiguation. *Computational Linguistics*, 40(1):57–84.

Luisa Bentivogli and Emanuele Pianta. 2004. Extending WordNet with Syntagmatic Information. In *Proceedings of second global WordNet conference*, pages 47–53, Brno, Czech Republic.

Jordan Boyd-Graber, Christiane Fellbaum, Daniel Osherson, and Robert Schapire. 2006. Adding Dense, Weighted Connections to WordNet. In *Proceedings of the third international WordNet conference*, pages 29–36, South Jeju Island, Korea.

Junpeng Chen and Juan Liu. 2011. Combining ConceptNet and WordNet for Word Sense Disambiguation. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 686–694, Chiang Mai, Thailand.

Montse Cuadros, Lluís Padró, and German Rigau. 2012. Highlighting relevant concepts from topic signatures. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*, pages 3841–3848, Istanbul, Turkey.

Montse Cuadros and German Rigau. 2008. KnowNet: Building a Large Net of Knowledge from the Web. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 161–168, Manchester, United Kingdom.

Andrea Di Fabio, Simone Conia, and Roberto Navigli. 2019. VerbAtlas: a Novel Large-Scale Verbal Semantic Resource and Its Application to Semantic Role Labeling. In *Proc. of EMNLP-IJCNLP*.

Philip Edmonds and Scott Cotton. 2001. SENSEVAL-2: Overview. In *Proceedings of SENSEVAL-2 Second International Workshop on Evaluating Word Sense Disambiguation Systems*, pages 1–5, Toulouse, France.

Luis Espinosa-Anke, Jose Camacho-Collados, Sara Rodríguez Fernández, Horacio Saggion, and Leo Wanner. 2016. Extending WordNet with Fine-Grained Collocational Information via Supervised Distributional Learning. In *Proceedings of COLING 2016: Technical Papers. The 26th International Conference on Computational Linguistics*, pages 900–910, Osaka, Japan.

Christiane Fellbaum, editor. 1998. *WordNet: an electronic lexical database*. MIT Press, Cambridge, MA, USA.

Yoan Gutiérrez Vázquez, Antonio Fernandez Orquín, Andrés Montoyo Guijarro, and Sonia Vázquez Pérez. 2010. UMCC-DLSI: Integrative Resource for Disambiguation Task. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 427–432, Uppsala, Sweden.

Taher H. Haveliwala. 2002. Topic-Sensitive PageRank. In *Proceedings of the 11th international conference on World Wide Web*, pages 517–526, Honolulu, HI, USA.

Geoffrey Neil Leech. 1992. 100 million words of English: the British National Corpus (BNC). *Language Research*.

Lothar Lemnitzer, Holger Wunsch, and Piklu Gupta. 2008. Enriching GermaNet with verb-noun relations - a case study of lexical acquisition. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation, LREC 2008, May 28-30, 2018*, pages 156–160, Marrakech, Morocco.

Edward Loper and Steven Bird. 2002. NLTK: The Natural Language Toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02, pages 63–70, Stroudsburg, PA, USA.

Steve L. Manion. 2015. SUDOKU: Treating Word Sense Disambiguation & Entity Linking as a Deterministic Problem-via an Unsupervised & Iterative Approach. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval-2015)*, pages 365–369, Denver, CO, USA.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, MD, USA.

Stefano Melacci, Achille Globo, and Leonardo Rigutini. 2018. Enhancing Modern Supervised Word Sense Disambiguation Models by Semantic Lexical Resources. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, May 7-12, 2018*, Miyazaki, Japan.

Rada Mihalcea and Dan Moldovan. 2001. eXtended WordNet: Progress Report. In *Proc. of the NAACL Workshop on WordNet and Other Lexical Resources*, pages 95–100, Pittsburgh, PA, USA.

Andrea Moro and Roberto Navigli. 2015. SemEval-2015 task 13: Multilingual all-words sense disambiguation and entity linking. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 288–297, Denver, CO, USA.

Roberto Navigli. 2018. Natural Language Understanding: Instructions for (present and future) use. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI 2018)*, pages 5697–5702, Stockholm, Sweden.

Roberto Navigli, David Jurgens, and Daniele Vannella. 2013. SemEval-2013 task 12: Multilingual word sense disambiguation. In *Second Joint Conference*

*on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 222–231, Atlanta, GA, USA.

Roberto Navigli and Mirella Lapata. 2010. An Experimental Study of Graph Connectivity for Unsupervised Word Sense Disambiguation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(4):678–692.

Roberto Navigli and Federico Martelli. 2019. An overview of word and sense similarity. *Natural Language Engineering*, 25(6):1–22.

Roberto Navigli and Simone P. Ponzetto. 2012. BabelNet: The Automatic Construction, Evaluation and Application of a Wide-Coverage Multilingual Semantic Network. *Artificial Intelligence Journal*, 193:217–250.

Tommaso Pasini and Roberto Navigli. 2017. Train-O-Matic: Large-Scale Supervised Word Sense Disambiguation in Multiple Languages without Manual Training Data. In *Proc. of EMNLP: System Demonstrations*, pages 78–88, Copenhagen, Denmark.

Simone P. Ponzetto and Roberto Navigli. 2010. Knowledge-Rich Word Sense Disambiguation Rivaling Supervised Systems. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1522–1531, Uppsala, Sweden.

Sameer S. Pradhan, Edward Loper, Dmitriy Dligach, and Martha Palmer. 2007. SemEval-2007 Task 17: English Lexical Sample, SRL and All Words. In *Proc. of SemEval-2007*, pages 87–92, Stroudsburg, PA, USA.

Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. 2017a. Word Sense Disambiguation: A Unified Evaluation Framework and Empirical Comparison. In *Proc. of EACL*, pages 99–110, Valencia, Spain.

Alessandro Raganato, Claudio Delli Bovi, and Roberto Navigli. 2017b. Neural Sequence Learning Models for Word Sense Disambiguation. In *Proc. of EMNLP*, pages 1167–1178, Copenhagen, Denmark.

Ferdinand de Saussure. 1916. *Cours de linguistique générale*. Payot, Paris.

Kiril Simov, Petya Osenova, and Alexander Popov. 2016. Using Context Information for Knowledge-Based Word Sense Disambiguation. In *International Conference on Artificial Intelligence: Methodology, Systems, and Applications*, pages 130–139, Varna, Bulgaria.

Kiril Simov, Alexander Popov, Iliana Simova, and Petya Osenova. 2018. Grammatical Role Embeddings for Enhancements of Relation Density in the Princeton WordNet. In *Proceedings of the 9th Global WordNet Conference*, pages 287–295, Singapore.

Benjamin Snyder and Martha Palmer. 2004. The English all-words task. In *In Proceedings of the 3rd International Workshop on the Evaluation of Systems for the Semantic Analysis of Text (SENSEVAL-3)*, pages 41–43, Barcelona, Spain.

Tolga Uslu, Alexander Mehler, Daniel Baumartz, Alexander Henlein, and Wahed Hemati. 2018. FastSense: An Efficient Word Sense Disambiguation Classifier. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, May 7-12, 2018*, Miyazaki, Japan.

Dayu Yuan, Ryan Doherty, Julian Richardson, Colin Evans, and Eric Altendorf. 2016. Semi-supervised Word Sense Disambiguation with Neural Models. In *Proc. of COLING*, pages 1374–1385, Osaka, Japan.

# Hierarchical Meta-Embeddings for Code-Switching Named Entity Recognition

**Genta Indra Winata, Zhaojiang Lin, Jamin Shin, Zihan Liu, Pascale Fung**
Center for Artificial Intelligence Research (CAiRE)
Department of Electronic and Computer Engineering
The Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong
{giwinata,zlinao,jmshinaa,zliucr}@connect.ust.hk,
pascale@ece.ust.hk

## Abstract

In countries that speak multiple main languages, mixing up different languages within a conversation is commonly called *code-switching*. Previous works addressing this challenge mainly focused on word-level aspects such as word embeddings. However, in many cases, languages share common subwords, especially for closely related languages, but also for languages that are seemingly irrelevant. Therefore, we propose *Hierarchical Meta-Embeddings* (HME) that learn to combine multiple monolingual word-level and subword-level embeddings to create language-agnostic lexical representations. On the task of Named Entity Recognition for English-Spanish code-switching data, our model achieves the state-of-the-art performance in the multilingual settings. We also show that, in cross-lingual settings, our model not only leverages closely related languages, but also learns from languages with different roots. Finally, we show that combining different subunits are crucial for capturing code-switching entities.

## 1 Introduction

*Code-switching* is a phenomenon that often happens between multilingual speakers, in which they switch between their two languages in conversations, hence, it is practically useful to recognize this well in spoken language systems (Winata et al., 2018a). This occurs more often for entities such as organizations or products, which motivates us to focus on the specific problem of Named Entity Recognition (NER) in code-switching scenarios. We show one of the examples as the following:

- *walking dead* le quita el apetito a cualquiera
- **(translation)** *walking dead* (a movie title) takes away the appetite of anyone

For this task, previous works have mostly focused on applying pre-trained word embeddings from each language in order to represent noisy mixed-language texts, and combine them with character-level representations (Trivedi et al., 2018; Wang et al., 2018; Winata et al., 2018b). However, despite the effectiveness of such word-level approaches, they neglect the importance of subword-level characteristics shared across different languages. Such information is often hard to capture with word embeddings or randomly initialized character-level embeddings. Naturally, we can turn towards subword-level embeddings such as FastText (Grave et al., 2018) to help this task, which will evidently allow us to leverage the morphological structure shared across different languages.

Despite such expected usefulness, there has not been much attention focused around using subword-level features in this task. This is partly because of the non-trivial difficulty of combining different language embeddings in the subword space, which arises from the distinct segmentation into subwords for different languages. This leads us to explore the literature of Meta-Embeddings (Yin and Schütze, 2016; Muromägi et al., 2017; Bollegala et al., 2018; Coates and Bollegala, 2018; Kiela et al., 2018; Winata et al., 2019), which is a method to learn how to combine different embeddings.

In this paper, we propose **Hierarchical Meta-Embeddings (HME)** [1] which learns how to combine different pre-trained monolingual embeddings in word, subword, and character-level into a single language-agnostic lexical representation without using specific language identifiers. To address the issue of different segmentations, we add a Transformer (Vaswani et al., 2017) encoder

[1]The source code is available at https://github.com/gentaiscool/meta-emb



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3541–3547,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

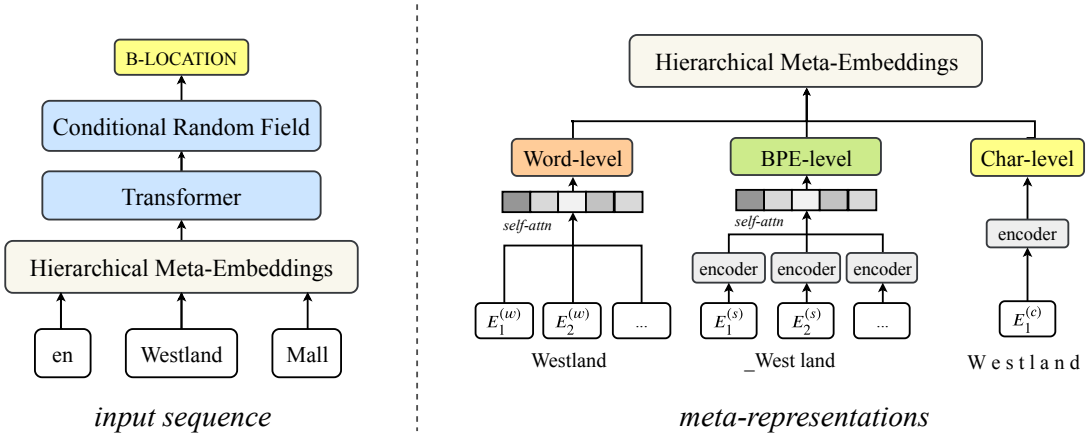


Figure 1: Hierarchical Meta-Embeddings (HME) architecture for Named Entity Recognition (NER) task. **Left:** Transformer-CRF architecture for Named Entity Recognition. **Right:** HME accept words, BPEs, and characters inputs.

which learns the important subwords in a given sentence. We evaluate our model on the task of Named Entity Recognition for English-Spanish code-switching data, and we use **Transformer-CRF**, a transformer-based encoder for sequence labeling based on the implementation of Winata et al. (2019). Our experimental results confirm that HME significantly outperforms the state-of-the-art system in *absolute F1 score*. The analysis shows that in the task of English-Spanish mixed texts not only similar languages like Portuguese or Catalan help, but also seemingly distant languages from Celtic origin also significantly increase the performance.

## 2 Related Work

**Embeddings** Previous works have extensively explored different representations such as word (Mikolov et al., 2013; Pennington et al., 2014; Grave et al., 2018; Xu et al., 2018), subword (Sennrich et al., 2016; Heinzerling and Strube, 2018), and character (dos Santos and Zadrozny, 2014; Wieting et al., 2016). Lample et al. (2016) has successfully concatenated character and word embeddings to their model, showing the potential of combining multiple representations. Liu et al. (2019) proposed to leverage word and subword embeddings into the application of unsupervised machine translation.

**Meta-embeddings** Recently, there are studies on combining multiple word embeddings in pre-processing steps (Yin and Schütze, 2016; Muromägi et al., 2017; Bollegala et al., 2018; Coates and Bollegala, 2018). Later, Kiela et al. (2018) introduced a method to dynamically learn word-level meta-embeddings, which can be effectively used in a supervised setting. Winata et al. (2019) proposed an idea to leverage multiple embeddings from different languages to generate language-agnostic meta-representations for mixed-language data.

## 3 Hierarchical Meta-Embeddings

We propose a method to combine word, subword, and character representations to create a mixture of embeddings. We generate a multilingual meta-embeddings of word and subword, and then, we concatenate them with character-level embeddings to generate final word representations, as shown in Figure 1. Let $\mathbf{w}$ be a sequence of words with $n$ elements, where $\mathbf{w} = [w_1, \ldots, w_n]$. Each word can be tokenized into a list of subwords $\mathbf{s} = [s_1, \ldots, s_m]$ and a list of characters $\mathbf{c} = [c_1, \ldots, c_p]$. The list of subwords $\mathbf{s}$ is generated using a function $f$; $\mathbf{s} = f(\mathbf{w})$. Function $f$ maps a word into a sequence of subwords. Further, let $E^{(w)}$, $E^{(s)}$, and $E^{(c)}$ be a set of word, subword, and character embedding lookup tables. Each set consists of different monolingual embeddings. Each element is transformed into a embedding vector in $\mathbb{R}^d$. We denote subscripts ${}_{\{i,j\}}$ as element and embedding language index, and superscripts ${}^{(w,s,c)}$ as word, subword, and character.

### 3.1 Multilingual Meta-Embeddings (MME)

We generate a meta-representations by taking the vector representation from multiple monolingual pre-trained embeddings in different subunits such

as word and subword. We apply a **projection matrix** $\mathbf{W}_j$ to transform the dimensions from the original space $\mathbf{x}_{i,j} \in \mathbb{R}^d$ to a new shared space $\mathbf{x'}_{i,j} \in \mathbb{R}^{d'}$. Then, we calculate **attention weights** $\alpha_{i,j} \in \mathbb{R}^{d'}$ with a non-linear scoring function $\phi$ (e.g., tanh) to take important information from each individual embedding $\mathbf{x'}_{i,j}$. Then, MME is calculated by taking the weighted sum of the projected embeddings $\mathbf{x'}_{i,j}$:

$$\mathbf{x'}_{i,j} = \mathbf{W}_j \cdot \mathbf{x}_{i,j}, \tag{1}$$

$$\alpha_{i,j} = \frac{\exp(\phi(\mathbf{x'}_{i,j}))}{\sum_{k=1}^{n} \exp(\phi(\mathbf{x'}_{i,k}))}, \tag{2}$$

$$\mathbf{u}_i = \sum_{j=1}^{n} \alpha_{i,j} \mathbf{x'}_{i,j}. \tag{3}$$

### 3.2 Mapping Subwords and Characters to Word-Level Representations

We propose to map subword into word representations and choose byte-pair encodings (BPEs) (Sennrich et al., 2016) since it has a compact vocabulary. First, we apply $f$ to segment words into sets of subwords, and then we extract the pre-trained subword embedding vectors $\mathbf{x}^{(s)}_{i,j} \in \mathbb{R}^d$ for language $j$.

Since, each language has a different $f$, we replace the projection matrix with Transformer (Vaswani et al., 2017) to learn and combine important subwords into a single vector representation. Then, we create $\mathbf{u}^{(s)}_i \in \mathbb{R}^{d'}$ which represents the subword-level MME by taking the weighted sum of $\mathbf{x'}^{(s)}_{i,j} \in \mathbb{R}^{d'}$.

$$\mathbf{x'}^{(s)}_{i,j} = \text{Encoder}(\mathbf{x}^{(s)}_{i,j}), \tag{4}$$

$$\mathbf{u}^{(s)}_i = \sum_{j=1}^{n} \alpha_{i,j} \mathbf{x'}^{(s)}_{i,j}. \tag{5}$$

To combine character-level representations, we apply an encoder to each character.

$$\mathbf{u}^{(c)}_i = \text{Encoder}(\mathbf{x}_i) \in \mathbb{R}^{d'}. \tag{6}$$

We combine the word-level, subword-level, and character-level representations by concatenation $\mathbf{u}^{HME}_i = (\mathbf{u}^{(w)}_i, \mathbf{u}^{(s)}_i, \mathbf{u}^{(c)}_i)$, where $\mathbf{u}^{(w)}_i \in \mathbb{R}^{d'}$ and $\mathbf{u}^{(s)}_i \in \mathbb{R}^{d'}$ are word-level MME and BPE-level MME, and $\mathbf{u}^{(c)}_i$ is a character embedding. We randomly initialize the character embedding and keep it trainable. We fix all subword and word pre-trained embeddings during the training.

### 3.3 Sequence Labeling

To predict the entities, we use Transformer-CRF, a transformer-based encoder followed by a Conditional Random Field (CRF) layer (Lafferty et al., 2001). The CRF layer is useful to constraint the dependencies between labels.

$$h = \text{Transformer}(\mathbf{u}), h' = \text{CRF}(h). \tag{7}$$

The best output sequence is selected by a forward propagation using the Viterbi algorithm.

## 4 Experiments

### 4.1 Experimental Setup

We train our model for solving Named Entity Recognition on English-Spanish code-switching tweets data from Aguilar et al. (2018). There are nine entity labels with IOB format. The training, development, and testing sets contain 50,757, 832, and 15,634 tweets, respectively.

We use FastText word embeddings trained from Common Crawl and Wikipedia (Grave et al., 2018) for English (*es*), Spanish (*es*), including **four Romance languages**: Catalan (*ca*), Portuguese (*pt*), French (*fr*), Italian (*it*), and **a Germanic language**: German (*de*), and **five Celtic languages** as the distant language group: Breton (*br*), Welsh (*cy*), Irish (*ga*), Scottish Gaelic (*gd*), Manx (*gv*). We also add the English Twitter GloVe word embeddings (Pennington et al., 2014) and BPE-based subword embeddings from Heinzerling and Strube (2018).

We train our model in two different settings: **(1) multilingual setting**, we combine main languages (*en-es*) with Romance languages and a Germanic language, and **(2) cross-lingual setting**, we use Romance and Germanic languages without main languages. Our model contains four layers of transformer encoders with a hidden size of 200, four heads, and a dropout of 0.1. We use Adam optimizer and start the training with a learning rate of 0.1 and an early stop of 15 iterations. We replace user hashtags and mentions with `<USR>`, emoji with `<EMOJI>`, and URL with `<URL>`. We evaluate our model using *absolute F1 score* metric.

### 4.2 Baselines

**CONCAT** We concatenate word embeddings by merging the dimensions of word representations. This method combines embeddings into a high-dimensional input that may cause inefficient com-

| Model | Multilingual embeddings | | | | Cross-lingual embeddings | | |
|---|---|---|---|---|---|---|---|
| | main languages | + closely-related languages | | + distant languages | closely-related languages | | distant languages |
| | en-es | ca-pt | ca-pt-de-fr-it | br-cy-ga-gd-gv | ca-pt | ca-pt-de-fr-it | br-cy-ga-gd-gv |
| *Flat word-level embeddings* | | | | | | | |
| CONCAT | 65.3 ± 0.38 | 64.99 ± 1.06 | 65.91 ± 1.16 | 65.79 ± 1.36 | 58.28 ± 2.66 | 64.02 ± 0.26 | 50.77 ± 1.55 |
| LINEAR | 64.61 ± 0.77 | 65.33 ± 0.87 | 65.63 ± 0.92 | 64.95 ± 0.77 | 60.72 ± 0.84 | 62.37 ± 1.01 | 53.28 ± 0.41 |
| *Multilingual Meta-Embeddings (MME)* (Winata et al., 2019) | | | | | | | |
| Word | 65.43 ± 0.67 | 66.63 ± 0.94 | 66.8 ± 0.43 | 66.56 ± 0.4 | 61.75 ± 0.56 | 63.23 ± 0.29 | 53.43 ± 0.37 |
| *Hierarchical Meta-Embeddings (HME)* | | | | | | | |
| + BPE | 65.9 ± 0.72 | 67.31 ± 0.34 | 67.26 ± 0.54 | 66.88 ± 0.37 | 63.44 ± 0.33 | 63.78 ± 0.62 | 60.19 ± 0.63 |
| + Char | 65.88 ± 1.02 | 67.38 ± 0.84 | 65.94 ± 0.47 | 66.1 ± 0.33 | 61.97 ± 0.6 | 63.06 ± 0.69 | 57.5 ± 0.56 |
| + BPE + Char | 66.55 ± 0.72 | **67.8 ± 0.31** | 67.07 ± 0.49 | 67.02 ± 0.16 | 63.9 ± 0.22 | **64.52 ± 0.35** | 60.88 ± 0.84 |

Table 1: Results (percentage F1 mean and standard deviation from five experiments). **Multilingual**: with main languages, **Cross-lingual**: without main languages.

| Model | F1 |
|---|---|
| Trivedi et al. (2018) | 61.89 |
| Wang et al. (2018) | 62.39 |
| Wang et al. (2018) (Ensemble) | 62.67 |
| Winata et al. (2018b) | 62.76 |
| Trivedi et al. (2018) (Ensemble) | 63.76 |
| Winata et al. (2019) MME | 66.63 ± 0.94 |
| Random embeddings | 46.68 ± 0.79 |
| *Aligned embeddings* | |
| MUSE (es → en) | 60.89 ± 0.37 |
| MUSE (en → es) | 61.49 ± 0.62 |
| *Multilingual embeddings* | |
| Our approach | 67.8 ± 0.31 |
| Our approach (Ensemble)† | **69.17** |
| *Cross-lingual embeddings* | |
| Our approach | 64.52 ± 0.35 |
| Our approach (Ensemble)† | **65.99** |

Table 2: Comparison to existing works. **Ensemble:** We run a majority voting scheme from five different models.

putation.

$$\mathbf{x}_i^{CONCAT} = [\mathbf{x}_{i,1}, ..., \mathbf{x}_{i,n}]. \quad (8)$$

**LINEAR** We sum all word embeddings into a single word vector with equal weight. This method combines embeddings without considering the importance of each of them.

$$\mathbf{x}_i^{LINEAR} = \sum_{j=0}^{n} \mathbf{x'}_{i,j}. \quad (9)$$

**Random Embeddings** We use randomly initialized word embeddings and keep it trainable to calculate the lower-bound performance.

**Aligned Embeddings** We align English and Spanish FastText embeddings using CSLS with two scenarios. We set English (en) as the source language and Spanish (es) (en → es) as the target language, and vice versa (es → en). We run MUSE by using the code prepared by the authors of Conneau et al. (2017). [2]

## 5 Results & Discussion

In general, from Table 1, we can see that word-level meta-embeddings even without subword or character-level information, consistently perform better than flat baselines (e.g., CONCAT and LINEAR) in all settings. This is mainly because of the attention layer which does not require additional parameters. Furthermore, comparing our approach to previous state-of-the-art models, we can clearly see that our proposed approaches all significantly outperform them.

From Table 1, in Multilingual setting, which trains with the main languages, it is evident that adding both closely-related and distant language embeddings improves the performance. This shows us that our model is able to leverage the lexical similarity between the languages. This is more distinctly shown in Cross-lingual setting as using distant languages significantly perform less than using closely-related ones (e.g., *ca-pt*). Interestingly, for distant languages, when adding subwords, we can still see a drastic performance increase. We hypothesize that even though the characters are mostly different, the lexical structure is similar to our main languages.

On the other hand, adding subword inputs to the model is consistently better than characters. This is due to the transfer of the information from the pre-trained subword embeddings. As shown in Ta-

[2] https://github.com/facebookresearch/MUSE

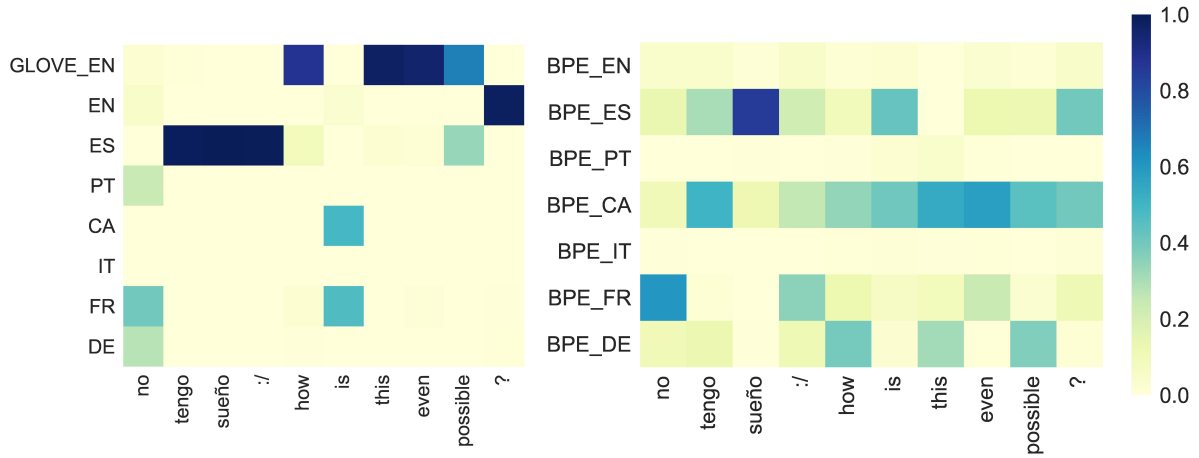


Figure 2: Heatmap of attention over languages from a validation sample. **Left**: word-level MME, **Right**: BPE-level MME. We extract the attention weights from a multilingual model (*en-es-ca-pt-de-fr-it*).

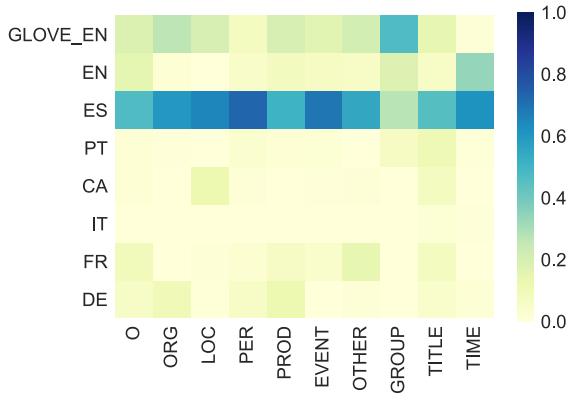


Figure 3: The average of attention weights for word embeddings versus NER tags from the validation set.

ble 1, subword embeddings is more effective for distant languages (Celtic languages) than closely-related languages such as Catalan or Portuguese.

Moreover, we visualize the attention weights of the model in word and subword-level to interpret the model dynamics. From the left image of Figure 2, in word-level, the model mostly chooses the correct language embedding for each word, but also combines with different languages. Without any language identifiers, it is impressive to see that our model learns to attend to the right languages. The right side of Figure 2, which shows attention weight distributions for subword-level, demonstrates interesting behaviors, in which for most English subwords, the model leverages *ca*, *fr*, and *de* embeddings. We hypothesize this is because the dataset is mainly constructed with Spanish words, which can also be verified from Figure 3 in which most NER tags are classified as *es*.

## 6 Conclusion

We propose *Hierarchical Meta-Embeddings* (HME) that learns how to combine multiple monolingual word-level and subword-level embeddings to create language-agnostic representations without specific language information. We achieve the state-of-the-art results on the task of Named Entity Recognition for English-Spanish code-switching data. We also show that our model can leverage subword information very effectively from languages from different roots to generate better word representations.

## Acknowledgments

This work has been partially funded by ITF/319/16FP and MRP/055/18 of the Innovation Technology Commission, the Hong Kong SAR Government, and School of Engineering Ph.D. Fellowship Award, the Hong Kong University of Science and Technology, and RDC 1718050-0 of EMOS.AI. We sincerely thank the three anonymous reviewers for their insightful comments on our paper.

## References

Gustavo Aguilar, Fahad AlGhamdi, Victor Soto, Mona Diab, Julia Hirschberg, and Thamar Solorio. 2018. Named entity recognition on code-switched data: Overview of the calcs 2018 shared task. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*, pages 138–147, Melbourne, Australia. Association for Computational Linguistics.

Danushka Bollegala, Kohei Hayashi, and Ken-Ichi Kawarabayashi. 2018. Think globally, embed lo-

cally: locally linear meta-embedding of words. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 3970–3976. AAAI Press.

Joshua Coates and Danushka Bollegala. 2018. Frustratingly easy meta-embedding–computing meta-embeddings by averaging source word embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 194–198.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Benjamin Heinzerling and Michael Strube. 2018. Bpemb: Tokenization-free pre-trained subword embeddings in 275 languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*.

Douwe Kiela, Changhan Wang, and Kyunghyun Cho. 2018. Dynamic meta-embeddings for improved sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1466–1477.

John D. Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270.

Zihan Liu, Yan Xu, Genta Indra Winata, and Pascale Fung. 2019. Incorporating word and subword units in unsupervised machine translation using language model rescoring. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 275–282, Florence, Italy. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Avo Muromägi, Kairit Sirts, and Sven Laur. 2017. Linear ensembles of word embedding models. In *Proceedings of the 21st Nordic Conference on Computational Linguistics*, pages 96–104.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Cícero Nogueira dos Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *ICML*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1715–1725.

Shashwat Trivedi, Harsh Rangwani, and Anil Kumar Singh. 2018. Iit (bhu) submission for the acl shared task on named entity recognition on code-switched data. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*, pages 148–153.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Changhan Wang, Kyunghyun Cho, and Douwe Kiela. 2018. Code-switched named entity recognition with embedding attention. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*, pages 154–158.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Charagram: Embedding words and sentences via character n-grams. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1504–1515.

Genta Indra Winata, Zhaojiang Lin, and Pascale Fung. 2019. Learning multilingual meta-embeddings for code-switching named entity recognition. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 181–186.

Genta Indra Winata, Andrea Madotto, Chien-Sheng Wu, and Pascale Fung. 2018a. Code-switching language modeling using syntax-aware multi-task learning. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*, pages 62–67.

Genta Indra Winata, Chien-Sheng Wu, Andrea Madotto, and Pascale Fung. 2018b. Bilingual character representation for efficiently addressing out-of-vocabulary words in code-switching named entity recognition. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*, pages 110–114.

Peng Xu, Andrea Madotto, Chien-Sheng Wu, Ji Ho Park, and Pascale Fung. 2018. Emo2vec: Learning generalized emotion representation by multi-task training. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 292–298.

Wenpeng Yin and Hinrich Schütze. 2016. Learning word meta-embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1351–1360.

# Fine-tune BERT with Sparse Self-Attention Mechanism

**Baiyun Cui, Yingming Li*, Ming Chen, and Zhongfei Zhang**
College of Information Science and Electronic Engineering,
Zhejiang University, China
baiyunc@yahoo.com,{yingming,funkyblack,zhongfei}@zju.edu.cn

## Abstract

In this paper, we develop a novel Sparse Self-Attention Fine-tuning model (referred as SSAF) which integrates sparsity into self-attention mechanism to enhance the fine-tuning performance of BERT. In particular, sparsity is introduced into the self-attention by replacing softmax function with a controllable sparse transformation when fine-tuning with BERT. It enables us to learn a structurally sparse attention distribution, which leads to a more interpretable representation for the whole input. The proposed model is evaluated on sentiment analysis, question answering, and natural language inference tasks. The extensive experimental results across multiple datasets demonstrate its effectiveness and superiority to the baseline methods.

## 1 Introduction

Recently, the pre-trained language models obtain new state-of-the-art results on a broad range of tasks, for example ULMFiT (Howard and Ruder, 2018), ELMo (Peters et al., 2018), and OpenAI GPT (Radford et al., 2018). Devlin et al. (2018) proposed BERT, a deep bidirectional language representation model which substantially outperforms the previous methods and has attracted wide attention in natural language processing. Fine-tuning on the pre-trained BERT has shown to be beneficial to improve many downstream tasks such as document classification (Adhikari et al., 2019), extractive summarization (Liu, 2019), question answering (Yang et al., 2019), natural language inference (Liu et al., 2019), and reading comprehension (Xu et al., 2019).

To further understand the impact of BERT on the fine-tuning performance, we exploit the attention behavior of the model when fine-tuning BERT in sentiment analysis task. In particular, the attention matrices of models with different setups are visualized in Figure 1. As shown in Figure 1(b), the attention distribution of fine-tuning with BERT is close to being a band matrix, with interest concentrated on small regions near the diagonal. Compared to the model trained from scratch without pre-training (Figure 1(a)), more interpretability would be provided by fine-tuning with BERT due to its structural attention distribution. This difference in the distributions might be caused by the masked language model task of BERT, where the model is required to predict the original vocabu-

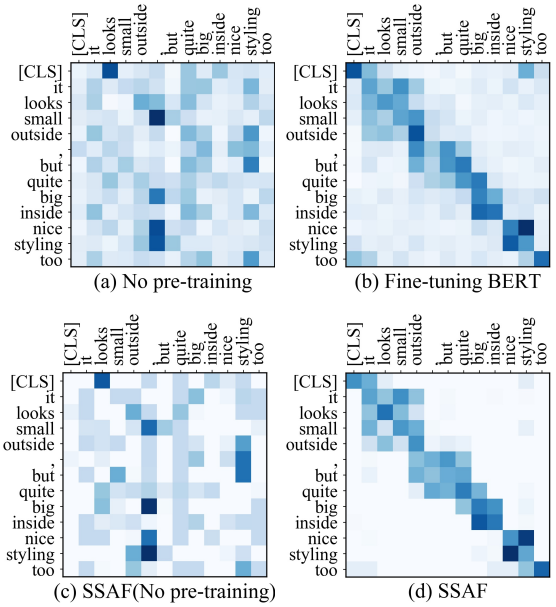


Figure 1: The explorations of attention behavior for four models: No pre-training, Fine-tuning BERT, SSAF(No pre-training), and SSAF. All the attention matrices are extracted from the eighth attention layer of the model for the sentence in Sentube-A dataset. Words on the y-axis are attending to the words on the x-axis. The attention distribution in SSAF presents more structurally sparse patterns compared to the others.

*Corresponding author



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3548–3553,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

lary id of the masked token based on its context. Consequently, the local interactions among different words are strengthened and play more important roles in predicting the masked token. Such modeling makes the learned representation more expressive for a diversity of tasks. However, the current fine-tuning models usually employ the pre-trained BERT as initialization of the networks and do not pay enough attention on how to dynamically control this attention distribution to be structurally sparse during fine-tuning process, which might further help improve the interpretability of the framework.

To overcome the above limitation, in this work, we develop a novel Sparse Self-Attention Fine-tuning model (referred as SSAF) by integrating sparsity into self-attention to refine the attention distribution when fine-tuning the pre-trained BERT. Specifically, we propose sparse self-attention mechanism (SSAM) to induce sparsity by replacing the traditional softmax function with a sparse transformation in self-attention networks. Instead of covering all the connections built between different words as the original fine-tuning model, the sparsity is designed to promote the most essential relationships among them with higher attention weights, and meanwhile eliminate the influence from meaningless relations by truncating their probabilities to exactly 0. As presented in Figure 1(d), more sparse attention structure are obtained by SSAF through fine-tuning BERT with sparsity constraint. Even without the pre-training, the sparse self-attention networks (SSAF No pre-training) also behaves better than the traditional self attention version in respect to sparsity, which is shown in Figure 1(c). As a generic module, SSAM is flexible to be applied to neural network models in a wide variety of tasks.

Extensive experiments are conducted on three NLP tasks to investigate the performances of SSAF, which include sentiment analysis, question answering, and natural language inference. Evaluation results on seven public datasets show that the proposed approach achieves remarkable improvements over other competing models.

## 2 Fine-tuning with Sparse Self-Attention

In this section, we first clarify the traditional self-attention model, then present a sparse self-attention mechanism, which extends the self-attention by replacing the standard softmax function with a sparse transformation, and finally develop a sparse self-attention fine-tuning model.

### 2.1 Self-attention mechanism

We start by introducing self-attention mechanism, which is the foundation of Transformer encoder (Vaswani et al., 2017) as well as BERT. Self-attention networks is capable of directly relating tokens at different positions from the sequence by computing the attention score (relevance) between each pair of tokens.

Formally, given an input sequence $\mathbf{x} = (x_1, \cdots x_L)$, the output representation $\mathbf{y} = (y_1, \cdots y_L)$ is constructed by applying weighted sum of transformations of the input elements $\mathbf{x}$ based on the relevance, where the elements $\{x_i, y_i\} \in \mathbb{R}^d$. The $i$-th output $y_i$ is computed as:

$$y_i = \sum_{j=1}^{L} \alpha_{ij}(x_j W_v) \tag{1}$$

$$e_{ij} = \frac{(x_i W_q)(x_j W_k)^T}{\sqrt{d}}, \alpha_{ij} = \rho(e_{ij}) \tag{2}$$

where $W_q \in \mathbb{R}^{d\times d}$, $W_k \in \mathbb{R}^{d\times d}$, and $W_v \in \mathbb{R}^{d\times d}$ denote the trainable parameter matrices. $\rho$ is a probability mapping function. The resulting attention weight $\alpha_{ij}$ represents the relevance between the $i$-th and $j$-th input element.

The classical choice for $\rho$ is the softmax transformation, which is calculated as:

$$\rho(e_{ij}) = \text{softmax}(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{t=1}^{L} \exp(e_{it})} \tag{3}$$

However, since softmax function is strictly positive, it produces attention distribution with full support. This results in the dense dependencies between each pair of words and fails to assign exactly zero probability to less meaningful relationships. Further, putting nonzero weight on every relationship would also degrade the interpretability. With such softmax transformation, the self-attention networks does not pay more attention to those important connections while also being easily disturbed by many unrelated words.

### 2.2 Sparse Self-Attention Mechanism

To address this problem, we employ sparsegen-lin transformation (Laha et al., 2018) to replace softmax in Equation 3, which not only leads to a sparse probability distribution but also offers an explicit control over the degree of sparsity.

Sparsegen-lin formulation projects the attention scores $\mathbf{e}_i = (e_{i1}, e_{i2}, \cdots e_{iL})$ onto the probability

simplex $\mathbf{p}_i$ and introduces coefficient $\lambda < 1$ to influence the regularization strength:

$$\rho(\mathbf{e}_i) = \underset{\mathbf{p}_i \in \Delta^{L-1}}{\operatorname{argmin}} \left\{ ||\mathbf{p}_i - \mathbf{e}_i||^2 - \lambda ||\mathbf{p}_i||^2 \right\} \quad (4)$$

where $\Delta^{L-1} := \left\{ \mathbf{p}_i \in \mathbb{R}^L | \sum_{j=1}^{L} p_{ij} = 1, \mathbf{p}_i \geq 0 \right\}$. The sparse attention distribution is then computed as follows:

$$\rho(e_{ij}) = p_{ij} = \max \left\{ 0, \frac{e_{ij} - \tau(\mathbf{e}_i)}{1 - \lambda} \right\} \quad (5)$$

where $j \in \{1, \cdots L\}$ and $\tau : \mathbb{R}^L \rightarrow \mathbb{R}$ is the threshold function.

More specifically, let $e_{i(1)} \geq e_{i(2)} \geq \cdots \geq e_{i(L)}$ be the sorted attention scores of $\mathbf{e}_i$ and $k(\mathbf{e}_i) := \max \left\{ k \in \{1, \cdots, L\} | 1 - \lambda + k e_{i(k)} > \sum_{j \leq k} e_{i(j)} \right\}$. The threshold $\tau(\mathbf{e}_i)$ is obtained as:

$$\begin{aligned} \tau(\mathbf{e}_i) &= \frac{(\sum_{j \leq k(\mathbf{e}_i)} e_{i(j)}) - 1 + \lambda}{k(\mathbf{e}_i)} \\ &= \frac{(\sum_{j \in S(\mathbf{e}_i)} e_{ij}) - 1 + \lambda}{|S(\mathbf{e}_i)|} \end{aligned} \quad (6)$$

where $S(\mathbf{e}_i)$ is the support of $\rho(\mathbf{e}_i)$, i.e., a set of the indices of nonzero coordinates. As in Equation 5, all the coordinates in the $S(\mathbf{e}_i)$ will be modified, and the others will be truncated to zero, thus providing a sparse solution. The choice of $\lambda$ helps control the cardinality of the support set $S(\mathbf{e}_i)$ which influences the sparsity of attention distribution.

By introducing sparsity to refine the attention weight, our sparse self-attention mechanism (SSAM) strengthens the most important relations among different words such as local interactions, and assigns zero probability to those meaningless connections. This enables us to achieve a more expressive representation for the whole input.

### 2.3 Sparse Self-Attention Fine-tuning model

In this part, we propose a sparse self-attention fine-tuning model (SSAF). In particular, this fine-tuning model with BERT is composed of $N$ sparse self-attention layers, where each layer learns a representation by taking the output from the previous layer:

$$\tilde{\mathbf{h}}^n = \text{LN}(\mathbf{h}^{n-1} + \text{SSAM}(\mathbf{h}^{n-1})) \quad (7)$$

$$\mathbf{h}^n = \text{LN}(\tilde{\mathbf{h}}^n + \text{FFN}(\tilde{\mathbf{h}}^n)) \quad (8)$$

where SSAM is adopted to replace the traditional self-attention mechanism, $\mathbf{h}^0 = \text{embed}(\mathbf{x})$ denotes the representation for the input sequence $\mathbf{x}$ which is the sum of token embeddings and the position embeddings, and LN is the layer normalization operation.

**Relationships with existing methods** Although several sparse formulations have been developed in the literature such as sparsemax (Martins and Astudillo, 2016), fusedmax (Niculae and Blondel, 2017), and constrained sparsemax (Malaviya et al., 2018), they are mostly applied in the classification layer or in the attention-based encoder-decoder architecture. Instead, in this work, we introduce sparsity into self-attention based transformer encoder. Other than concentrating on words as the existing approaches do, we take the advantage of sparsity to identify the most essential relationships among words to capture a better sequence representation. Further, more interpretability would be obtained by our method due to the structurally sparse attention distribution.

## 3 Experiments

In the following sections, we empirically evaluate the effectiveness of SSAF for three NLP tasks on seven public datasets.

### 3.1 Datasets

**Sentiment Analysis (SA)** The goal of sentiment analysis is to determine the sentiment classification of a piece of text. Accuracy is used as the evaluation metric. We evaluate our model on five datasets in this task, which are listed as follows. 1) SST-1: The Stanford Sentiment Treebank (Socher et al., 2013) consists of sentences extracted from movie reviews with five classes. We follow the previous works (Kim, 2014; Gong et al., 2018) to train the model on both phrase and sentence level; 2) SST-2: This dataset is constructed from the same data with SST-1 but without the neutral reviews. We adopt the dataset version provided by GLUE (Wang et al., 2018); 3) SemEval: The SemEval 2013 Twitter dataset (Nakov et al., 2013) contains tweets with three classes: positive, negative, and neutral; 4) Sentube-A, Sentube-T: The SenTube datasets (Uryupina et al., 2014) are texts obtained from YouTube comments with two sentiment classes: positive and negative.

**Question Answering (QA)** This task is to predict the answer text span in the paragraph according to the question. We adopt the SQuAD v1.1 dataset (Rajpurkar et al., 2016). Since $\text{BERT}_{\text{BASE}}$ is re-

| Corpus | Task | Train | Dev. | Test | Class |
|---|---|---|---|---|---|
| SST-1 | SA | 8.5k | 1.1k | 2.2k | 5 |
| SST-2 | SA | 67.3k | 0.8k | 1.8k | 2 |
| Sentube-A | SA | 3.3k | 0.2k | 0.9k | 2 |
| Sentube-T | SA | 4.9k | 0.3k | 1.3k | 2 |
| SemEval | SA | 6.0k | 0.8k | 2.3k | 3 |
| SQuAD | QA | 87.5k | 10.1k | 10.1k | - |
| SciTail | NLI | 23.5k | 1.3k | 2.1k | 2 |

Table 1: Summary of seven datasets used in our experiments. Train, Dev., and Test: The size of train, development, and test set respectively.

| Models | SST-1 | SST-2 | SenTube-A | SenTube-T | SemEval |
|---|---|---|---|---|---|
| Ave | 42.3 | 81.1 | 61.5 | 64.3 | 63.6 |
| LSTM | 46.5 | 82.9 | 57.4 | 63.6 | 67.6 |
| BiLSTM | 47.1 | 83.7 | 59.3 | 66.2 | 65.1 |
| CNN | 41.9 | 81.8 | 57.3 | 62.1 | 63.5 |
| SSAN | 48.6 | 85.3 | 62.5 | 68.4 | 72.2 |
| Np | 50.1 | 85.2 | 66.8 | 69.6 | 70.0 |
| SSAF(Np) | 50.7 | 86.4 | 68.1 | 70.3 | 71.2 |
| $\text{BERT}_{\text{BASE}}$ | 55.2 | 93.5 | 70.3 | 73.3 | 76.2 |
| SSAF | **56.2** | **94.7** | **72.4** | **75.0** | **77.3** |

Table 2: Experimental results of classification accuracy for different methods with five datasets on sentiment analysis task.

ported on the development set of SQuAD in Devlin et al. (2018), we follow them to evaluate our model on the same set. The Exact Match (EM) and F1 are two evaluation metrics.

**Natural Language Inference (NLI)** The task involves assessing if two sentences entail or contradict each other. We use SciTail dataset (Khot et al., 2018) which is derived from a science question answering (SciQ) dataset. The evaluation metric is accuracy.

Further statistics about datasets are illustrated in Table 1.

### 3.2 Implementation details

We adopt the pre-trained $\text{BERT}_{\text{BASE}}$ [1] as the basis for our experiments. We choose $\text{BERT}_{\text{BASE}}$ in our work rather than another larger pre-trained model $\text{BERT}_{\text{LARGE}}$ due to the resource limitations and computation cost. The coefficient $\lambda$ which controls the sparisty in Equation 4 is set to -3 in SST-1 and SemEval, -4 in SST-2 and SenTube-T, -6 in SenTube-A and SciTail, and -7 in SQuAD. We investigate the influence of different $\lambda$ settings in the experiment analysis part. We adopt Adam as our optimizer with a learning rate of 2e-5 and batch size is 16. The maximum number of training epoch is 2 for SQuAD, 3 for SciTail, 4 for SST-1 and SST-2, 5 for SenTube-A, 6 for SenTube-T, and 8 for SemEval.

### 3.3 Baselines

To demonstrate that SSAF truly improves the fine-tuning performance, we compare SSAF against $\text{BERT}_{\text{BASE}}$ from (Devlin et al., 2018) on all the tasks. In particular, for sentiment analysis, we follow (Ambartsoumian and Popowich, 2018) to make comparisons with the following representative methods: Ave, LSTM, BiLSTM, and CNN from (Barnes et al., 2017); SSAN from (Ambartsoumian and Popowich, 2018). We report the results of these five models with Google 300-dimensional word2vec embeddings [2] on all datasets. For SST-1 and SST-2, we reproduce the compared methods according to the different dataset version.

For thorough comparison, besides the approaches proposed in the existing literature, we further implement another two models to verify the ability of sparse self-attention mechanism:

**No pre-training (Np)**: It has the same architecture as $\text{BERT}_{\text{BASE}}$ but without pre-training.

**SSAF(Np)**: We train SSAF from scratch only with the sparse self-attention module.

### 3.4 Results

The experimental results on sentiment analysis are reported in Table 2, and the performances on question answering and natural language inference are summarized in Table 3. Results show that SSAF achieves the best performance across all three tasks with a significant improvement over the previous approaches.

Compared with other competing methods in the experiments, $\text{BERT}_{\text{BASE}}$ provides a strong baseline owing to the pre-training procedure. However, this model still suffers from its inefficiency by taking all the relationships between each pair of words into consideration when constructing the representation. Our model outperforms $\text{BERT}_{\text{BASE}}$ by a stable margin especially in SST-2, SenTube-A, SenTube-T, and SciTail, with the improvements of 1.2%, 2.1%, 1.7%, and 1.5%,

[1] https://github.com/google-research/bert

[2] https://code.google.com/archive/p/word2vec/

| Models | SQuAD | | SciTail |
|---|---|---|---|
| | EM | F1 | Accuracy |
| Np | 65.3 | 74.1 | 78.6 |
| SSAF(Np) | 66.2 | 74.6 | 78.9 |
| $\text{BERT}_{\text{BASE}}$ | 80.8 | 88.5 | 92.0 |
| SSAF | **81.6** | **88.8** | **93.5** |

Table 3: Experimental results on question answering and natural language inference tasks.

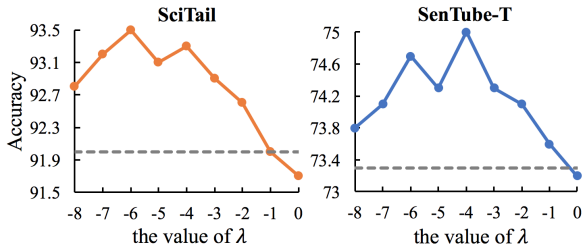


Figure 2: Experimental results on varying the coefficient $\lambda$ for SSAF in SciTail and SenTube-T datasets. Grey dash line denotes the result of $\text{BERT}_{\text{BASE}}$.

respectively. Meanwhile, even without the pre-training, SSAF(No pre-training) still surpasses No pre-training across the board which clearly proves the effectiveness and universality of incorporating sparsity into self-attention model. Combining the strength of sparsity and the pre-trained language model, SSAF stands out in performance with other baselines.

Moreover, we study how the value of coefficient $\lambda$ in SSAF affects the fine-tuning performance. We explore different settings from -8 to 0. Figure 2 shows the comparison results on the test set of SciTail and SenTube-T. Results summarize that $\lambda = -6$ is superior to other settings for SciTail and the performance on SenTube-T reaches the best when $\lambda$ is set to -4. According to the experimental results above, we could found that on both of the datasets, the values from -7 to -3 can reach comparable performance while both larger and smaller values are more likely to decrease the accuracy to some extent though they still outperform previous work by a large margin. To understand why different values of $\lambda$ have such different effects, we visualize the attention matrices extracted from SSAF with a series of $\lambda$ and also visualize the one from $\text{BERT}_{\text{BASE}}$ for comparison. As shown in Figure 3, the different structural patterns among these five sub-figures lie in the different strength of the sparsity constraint. It shows that too large coefficient $\lambda$ makes the attention distribution of SSAF extremely sparse, which may even overlooks some important interactions between different words. With the decrease of $\lambda$ , the influence from other unnecessary connections increases, which reduces the interpretability of the model. Thus, it is important to have appropriate parameter to encourage a desirably sparse attention distribution while achieving competing accuracy.

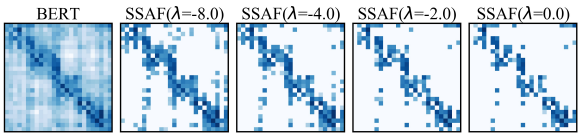


Figure 3: Visualization of attention matrices for $\text{BERT}_{\text{BASE}}$ and SSAF with different values of $\lambda$ on SenTube-T dataset. With higher $\lambda$, the attention distribution in SSAF becomes sparser.

## 4 Conclusion

In this paper, we develop a novel Sparse Self-Attention Fine-tuning model (referred as SSAF) which integrates sparsity into self-attention mechanism to enhance the fine-tuning performance of BERT. We conduct extensive experiments on sentiment analysis, question answering, and natural language inference tasks with seven public datasets. The proposed approach substantially improves the performance over the strong baseline methods, demonstrating its effectiveness and universality while achieving higher interpretability for the framework.

## Acknowledgments

This work was supported in part by National Natural Science Foundation of China (No. 61702448, 61672456), Zhejiang Lab (2018EC0ZX01-2), the Fundamental Research Funds for the Central Universities in China (No. 2017FZA5007, 2019FZA5005), the Key Program of Zhejiang Province, China (No. 2015C01027), Artificial Intelligence Research Foundation of Baidu Inc., the funding from HIKVision and Horizon Robotics, and ZJU Converging Media Computing Lab. We thank all reviewers for their valuable comments.

## References

Ashutosh Adhikari, Achyudh Ram, Raphael Tang, and Jimmy Lin. 2019. Docbert: Bert for document classification. *arXiv preprint arXiv:1904.08398*.

Artaches Ambartsoumian and Fred Popowich. 2018. Self-attention: A better building block for sentiment analysis neural network classifiers. *arXiv preprint arXiv:1812.07860*.

Jeremy Barnes, Roman Klinger, and Sabine Schulte im Walde. 2017. Assessing state-of-the-art sentiment models on state-of-the-art sentiment datasets. *arXiv preprint arXiv:1709.04219*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jingjing Gong, Xipeng Qiu, Shaojing Wang, and Xuanjing Huang. 2018. Information aggregation via dynamic routing for sequence encoding. *arXiv preprint arXiv:1806.01501*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *ACL*, pages 328–339.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. Scitail: A textual entailment dataset from science question answering. In *AAAI*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Anirban Laha, Saneem Ahmed Chemmengath, Priyanka Agrawal, Mitesh Khapra, Karthik Sankaranarayanan, and Harish G Ramaswamy. 2018. On controllable sparse alternatives to softmax. In *NeurIPS*, pages 6422–6432.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*.

Yang Liu. 2019. Fine-tune BERT for extractive summarization. *arXiv preprint arXiv:1903.10318*.

Chaitanya Malaviya, Pedro Ferreira, and André F. T. Martins. 2018. Sparse and constrained attention for neural machine translation. In *ACL*, pages 370–376.

Andre Martins and Ramon Astudillo. 2016. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *ICML*, pages 1614–1623.

Preslav Nakov, Sara Rosenthal, Zornitsa Kozareva, Veselin Stoyanov, Alan Ritter, and Theresa Wilson. 2013. Semeval-2013 task 2: Sentiment analysis in twitter. In *Proceedings of the 7th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT*, pages 312–320.

Vlad Niculae and Mathieu Blondel. 2017. A regularized framework for sparse and structured neural attention. In *NeurIPS*, pages 3338–3348.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*, pages 2227–2237.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *EMNLP*, pages 2383–2392.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, pages 1631–1642.

Olga Uryupina, Barbara Plank, Aliaksei Severyn, Agata Rotondi, and Alessandro Moschitti. 2014. Sentube: A corpus for sentiment analysis on youtube social media. In *LREC*, pages 4244–4249.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*, pages 5998–6008.

Alex Wang, Amapreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Hu Xu, Bing Liu, Lei Shu, and Philip S Yu. 2019. Bert post-training for review reading comprehension and aspect-based sentiment analysis. *arXiv preprint arXiv:1904.02232*.

Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019. End-to-end open-domain question answering with bertserini. *arXiv preprint arXiv:1902.01718*.

# Feature-Dependent Confusion Matrices for Low-Resource NER Labeling with Noisy Labels

**Lukas Lange**[1,2] **Michael A. Hedderich**[2] **Dietrich Klakow**[2]
[1] Bosch Center for Artificial Intelligence, Renningen, Germany
[2] Spoken Language Systems (LSV), Saarland University,
Saarland Informatics Campus, Saarbrücken, Germany
{llange,mhedderich,dietrich.klakow}@lsv.uni-saarland.de

## Abstract

In low-resource settings, the performance of supervised labeling models can be improved with automatically annotated or distantly supervised data, which is cheap to create but often noisy. Previous works have shown that significant improvements can be reached by injecting information about the confusion between clean and noisy labels in this additional training data into the classifier training. However, for noise estimation, these approaches either do not take the input features (in our case word embeddings) into account, or they need to learn the noise modeling from scratch which can be difficult in a low-resource setting. We propose to cluster the training data using the input features and then compute different confusion matrices for each cluster. To the best of our knowledge, our approach is the first to leverage feature-dependent noise modeling with pre-initialized confusion matrices. We evaluate on low-resource named entity recognition settings in several languages, showing that our methods improve upon other confusion-matrix based methods by up to 9%.

## 1 Introduction

Most languages, even with millions of speakers, have not been the center for natural language processing and are counted as low-resource for tasks like named entity recognition (NER). Similarly, even for high-resource languages, there exists only few labeled data for most entity types beyond person, location and organization. Distantly- or weakly-supervised approaches have been proposed to solve this issue, e.g., by using lists of entities for labeling raw text (Ratinov and Roth, 2009; Dembowski et al., 2017). This allows obtaining large amounts of training data quickly and cheaply. Unfortunately, these labels often contain errors and learning with this noisily-labeled data is difficult and can even reduce overall performance (see, e.g. Fang and Cohn (2016)).

A variety of ideas have been proposed to overcome the issues of noisy training data. One popular approach is to estimate the relation between noisy and clean, gold-standard labels and use this noise model to improve the training procedure. However, most of these approaches only assume a dependency between the labels and do not take the features into account when modeling the label noise. This may disregard important information. The global confusion matrix (Hedderich and Klakow, 2018) is a simple model which assumes that the errors in the noisy labels just depend on the clean labels.

Our contributions are as follows: We propose to cluster the input words with the help of additional, unlabeled data. Based on this partition of the feature space, we obtain different confusion matrices that describe the relationship between clean and noisy labels. We evaluate our newly proposed models and related baselines in several low-resource settings across different languages with real, distantly supervised data with non-synthetic noise. The advanced modeling of the noisy labels substantially improves the performance up to 36% over methods without noise-handling and up to 9% over all other noise-handling baselines.

## 2 Related Work

A popular approach is modeling the relationship between noisy and clean labels, i.e., estimating $p(\hat{y}|y)$ where $y$ is the clean and $\hat{y}$ the noisy label. For example, this can be represented as a noise or confusion matrix between the clean and the noisy labels, as explained in Section 3. Having its roots in statistics (Dawid and Skene, 1979), this or similar ideas have been recently studied in NLP (Fang and Cohn, 2016; Hedderich and



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3554–3559,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

Klakow, 2018; Paul et al., 2019), image classification (Mnih and Hinton, 2012; Sukhbaatar et al., 2015; Dgani et al., 2018) and general machine learning settings (Bekker and Goldberger, 2016; Patrini et al., 2017; Hendrycks et al., 2018). All of these methods, however, do not take the features into account that are used to represent the instances during classification. In (Xiao et al., 2015) only the noise type depends on $x$ but not the actual noise model. Goldberger and Ben-Reuven (2016) and Luo et al. (2017) use the learned feature representation $h$ to model $p(\hat{y}|y, h(x))$ for image classification and relation extraction respectively. In the work of Veit et al. (2017), $p(y|\hat{y}, h(x))$ is estimated to clean the labels for an image classification task. The survey by Frenay and Verleysen (2014) gives a detailed overview about other techniques for learning in the presence of noisy labels.

Specific to learning noisy sequence labels in NLP, Fang and Cohn (2016) used a combination of clean and noisy data for low-resource POS tagging. Yang et al. (2018) suggested partial annotation learning to lessen the effects of incomplete annotations and reinforcement learning for filtering incorrect labels for Chinese NER. Hedderich and Klakow (2018) used a confusion matrix and proposed to leverage pairs of clean and noisy labels for its initialization, evaluating on English NER. For English NER and Chunking, Paul et al. (2019) also used a confusion matrix but learned it with an EM approach and combined it with multi-task learning. Recently, Rahimi et al. (2019) studied input from different, unreliable sources and how to combine them for NER prediction.

## 3 Global Noise Model

We assume a low-resource setting with a small set of gold standard annotated data $C$ consisting of instances with features $x$ and corresponding, clean labels $y$. Additionally, a large set of noisy instances $(x, \hat{y}) \in N$ is available. This can be obtained e.g. from weak or distant supervision. In a multi-class classification setting, we can learn the probability of a label $y$ having a specific class given the feature $x$ as

$$p(y = i|x) = \frac{exp(u_i^T h(x))}{\sum_{l=1}^{k} exp(u_l^T h(x))} \quad (1)$$

where $k$ is the number of classes, $h$ is a learned, non-linear function (in our case a neural network) and $u$ is the softmax weights. This is our base model trained on $C$. Due to the errors in the labels, the clean and noisy labels have different distributions. Therefore, learning on $C$ and $N$ jointly can be detrimental for the performance of predicting unseen, clean instances. Nevertheless, the noisy-labeled data is still related to $C$ and can contain useful information that we want to successfully leverage. We transform the predicted (clean) distribution of the base model to the noisy label distribution

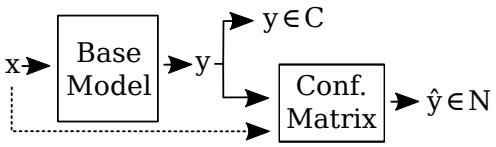


Figure 1: Visualization of the noisy labels, confusion matrix architecture. The dotted line shows the proposed new dependency.

$$p(\hat{y} = j|x) = \sum_{i=1}^{k} p(\hat{y} = j|y = i)p(y = i|x). \quad (2)$$

The relationship is modeled using a confusion matrix (also called noise or transformation matrix or noise layer) with learned weights $b_{ij}$:

$$p(\hat{y} = j|y = i) = \frac{\exp(b_{ij})}{\sum_{l=1}^{k} \exp(b_{il})} \quad (3)$$

The overall architecture is visualized in Figure 1. An important question is how to initialize this noise layer. As proposed by Hedderich and Klakow (2018), we apply the same distant supervision technique used to obtain $N$ from unlabeled data on the already labeled instances in $C$. We thus obtain pairs of clean $y$ and corresponding noisy labels $\hat{y}$ for the same instances and the weights of the noise layer can be initialized as

$$b_{ij} = \log\left(\frac{\sum_{t=1}^{|C|} 1_{\{y_t=i\}} 1_{\{\hat{y}_t=j\}}}{\sum_{t=1}^{|C|} 1_{\{y_t=i\}}}\right). \quad (4)$$

Following the naming by (Luo et al., 2017), we call this the global noise model.

## 4 Feature Dependent Noise Model

The global confusion matrix is a simple model which assumes that the errors in the noisy labels

depend on the clean labels. An approach that also takes the corresponding features $x$ into account can model more complex relations. Veit et al. (2017) and Luo et al. (2017) use multiple layers of a neural network to model these relationships. However, in low resource settings with only small amounts of clean, supervised data, these more complex models can be difficult to learn. In contrast to that, larger amounts of unlabeled text are usually available even in low-resource settings. Therefore, we propose to use unsupervised clustering techniques to partition the feature space of the input words (and the corresponding instances) before estimating the noise matrices. To create the clusters, we use either Brown clustering (Brown et al., 1992) on the input words or $k$-means clustering (Lloyd, 1982) on the pretrained word embeddings after applying PCA (Pearson, 1901).

In sequence labeling tasks, the features $x$ of an instance usually consist of the input word $\iota(x)$ and its context. Given a clustering $\Pi$ over the input words $\{\iota(x) \mid (x, y) \in C \cup N\}$ consisting of clusters $\Pi_1, ..., \Pi_p$, we can group all clean and noisy instances into groups

$$G_q = \{(x, y) \in C \cup N \mid \iota(x) \in \Pi_q\} \quad (5)$$

For each group, we construct an independent confusion matrix using Formulas 3 and 4. The prediction of the noisy label $\hat{y}$ (Formula 2) then becomes

$$p(\hat{y} = j|x) = \sum_{i=1}^{k} p(\hat{y} = j|y = i, G)p(y = i|x) \quad (6)$$

Since the clustering is performed on unsupervised data, in low-resource settings, the size of an actual group of instances $G_q$ can be very small. If the number of members in a group is insufficient, the estimation of reliable noise matrices is difficult. This issue can be avoided by only using the largest groups and creating a separate group for all other instances. To make use of all the clusters, we alternatively propose to interpolate between the global and the group confusion matrix:

$$p_{\text{int}}(\hat{y} = j|y = i, G) = (1-\lambda) \cdot p(\hat{y} = j|y = i, G) + \lambda \cdot p(\hat{y} = j|y = i) \quad (7)$$

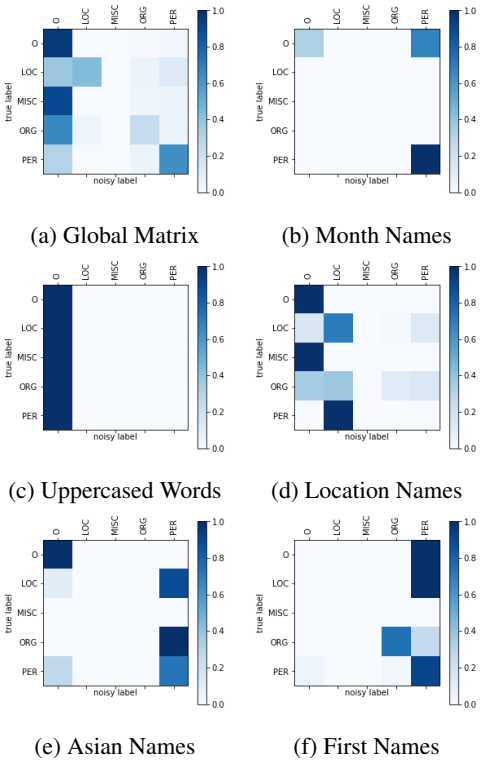


(a) Global Matrix (b) Month Names

(c) Uppercased Words (d) Location Names

(e) Asian Names (f) First Names

Figure 2: Confusion matrices used for initialization when training with the English dataset. The global matrix is given as well as five of the feature-dependent matrices obtained when using $k$-Means clustering for 75 clusters.

The interpolation hyperparameter $\lambda$ (with $0 \leq \lambda \leq 1$) regulates the influence from the global matrix on the interpolated matrix. The selection of the largest groups and the interpolation can also be combined.

## 5 Experiments

We evaluate all models in five low-resource NER settings across different languages. Although the evaluation is performed for NER labeling, the proposed models are not restricted to the task of NER and can potentially be used for other tasks.

### 5.1 Models [1]

We follow the BiLSTM architecture from Hedderich and Klakow (2018). Only the optimizer was changed for all models to NADAM (Dozat, 2016) as this helped with convergence problems for increasing cluster numbers. The **Base** is trained only on clean data while **Base+Noise** is trained on both the clean and the noisy data without noise handling. **Global-CM** uses a global

[1]The code for all models is made available at https://github.com/uds-lsv/noise-matrix-ner

| | De | En | Es | Et | Nl |
|---|---|---|---|---|---|
| Base | 21.4 ± 1.0 | 35.9 ± 4.6 | 39.1 ± 1.6 | 36.7 ± 1.8 | 15.5 ± 3.0 |
| Base+Noise | 26.2 ± 0.6 | 50.5 ± 1.4 | 50.2 ± 1.0 | 51.5 ± 0.7 | 29.5 ± 2.7 |
| Cleaning (Veit et al. 2017) | 16.1 ± 4.3 | 52.3 ± 2.3 | 48.7 ± 2.3 | 53.8 ± 0.4 | 24.4 ± 5.5 |
| Dynamic-CM (Luo et al. 2017) | 32.6 ± 0.9 | 53.7 ± 1.8 | 57.6 ± 0.8 | 52.3 ± 0.8 | 36.7 ± 2.9 |
| Global-ID-CM (H. and K. 2018) | 27.1 ± 0.7 | 51.0 ± 1.1 | 50.9 ± 0.7 | 51.4 ± 0.6 | 29.9 ± 2.6 |
| Global-CM (H. and K. 2018) | 34.1 ± 1.4 | 52.0 ± 1.6 | 52.8 ± 0.6 | 52.3 ± 0.6 | 33.3 ± 2.0 |
| Brown-CM-Freq | 32.7 ± 0.7 | 51.3 ± 1.3 | 54.8 ± 1.0 | 53.4 ± 0.8 | 38.1 ± 1.7 |
| K-Means-CM-Freq | 29.7 ± 2.3 | 54.1 ± 2.9 | 52.3 ± 1.2 | 54.9 ± 0.8 | 39.8 ± 1.8 |
| Brown-CM-IP | 29.6 ± 1.1 | 55.5 ± 3.7 | 55.6 ± 1.0 | 52.6 ± 0.9 | 37.3 ± 1.5 |
| K-Means-CM-IP | 33.4 ± 1.1 | 53.0 ± 4.0 | 56.3 ± 2.1 | 53.3 ± 0.5 | 36.0 ± 1.9 |
| Brown-CM-Freq-IP | **34.3 ± 1.4** | 51.4 ± 2.3 | **57.7 ± 2.4** | 53.1 ± 0.9 | **40.0 ± 1.3** |
| K-Means-CM-Freq-IP | 33.1 ± 2.1 | **57.6 ± 1.5** | 57.2 ± 1.3 | **55.2 ± 0.3** | 39.7 ± 1.0 |

Table 1: Results of the evaluation in low-resource settings with 1% of the original labeled training data averaged over six runs. We report the F1 scores (higher is better) on the complete test set, as well as the standard error.

confusion matrix for all noisy instances to model the noise as proposed by Hedderich and Klakow (2018) and presented in Section 3. The same architecture is used for **Global-ID-CM**, but the confusion matrix is initialized with the identity matrix (instead of Formula 4) and only adapted during training.

The cluster-based models we propose in Section 4 are **Brown-CM** and **K-Means-CM**. We experimented with numbers of clusters of 5, 10, 25 and 50. The models that select only the largest groups $G$ are marked as ***-Freq** and select either 30% or 50% of the clusters. The interpolation models have the postfix ***-IP** with $\lambda \in \{0.3, 0.5, 0.7\}$ . The combination of both is named ***-Freq-IP**. As for all other hyperparameters, the choice was taken on the development set.

We implemented the **Cleaning** (Veit et al., 2017) and **Dynamic-CM** (Luo et al., 2017) models. Both were not developed for sequence labeling tasks and therefore needed to be adapted. For the Cleaning model, we followed the instructions by Hedderich and Klakow (2018). The embedding and prediction components of the **Dynamic-CM** model were replaced according to our base model. The output of the dense layer was used as input to the dynamic matrix generation. We experimented with and without their proposed trace loss.

The training for all models was performed with labels in the IO format. The predicted labels for the test data were converted and evaluated in IOB2 with the official CoNLL evaluation script. The IOB2 format would increase matrix size making the confusion matrix estimation more difficult without adding much information in practice. In preliminary experiments, this decreased performance in particular for low-resource settings.

### 5.2 Data

The models were tested on the four CoNLL datasets for English, German, Spanish and Dutch (Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003) using the standard split, and the Estonian data from Tkachenko et al. (2013) using a 10/10/80 split for dev/test/train sets. For each language, the labels of 1% of the training data (ca. 2100 instances) were used to obtain a low-resource setting. We treat this as the clean data $C$. The rest of the (now unlabeled) training data was used for the automatic annotation which we treat as noisily labeled data $N$. We applied the distant supervision method by Dembowski et al. (2017), which uses lists and gazetteer information for NER labeling. As seen in Table 2, this method reaches rather high precision but has a poor recall. The development set of the original dataset is used for model-epoch and hyperparameter selection, and the results are reported on the complete, clean test set. The words were embedded with the pretrained fastText vectors (Grave et al., 2018). The clusters were calculated on the unlabeled version of the full training data. Additionally, the Brown clusters used the language-specific documents from the Europarl corpus (Koehn, 2005).

| | De | En | Es | Et | Nl |
|---|---|---|---|---|---|
| Precision | 23.2 | 39.9 | 51.0 | 59.7 | 32.4 |
| Recall | 9.2 | 30.1 | 24.7 | 49.3 | 21.1 |
| F1 | 13.2 | 34.3 | 33.3 | 54.0 | 25.5 |

Table 2: Results of the automatic labeling method proposed by Dembowski et al. (2017) on the test data.

## 6 Experimental Results

The results of all models are shown in Table 1. The newly proposed cluster-based models achieve the best performance across all languages and outperform all other models in particular for Dutch and English. The combination of interpolation with the global matrix and the selection of large clusters is almost always beneficial compared to the cluster-based models using only one of the methods. In general, both clustering methods achieve similar performance in combination with interpolation and selection, except for English, where Brown clustering performs worse than $k$-Means clustering. While the Brown clustering was trained on the relatively small Europarl corpus, $k$-Means clustering seems to benefit from the word embeddings trained on documents from the much larger common crawl.

## 7 Analysis

In the majority of cases, a cluster size of 10 or 25 was selected on the development set during the hyperparameter search. Increasing the number of clusters introduces smaller clusters for which it is difficult to estimate the noise matrix, due to the limited training resources. On the other hand, decreasing the number of clusters can generalize too much, resulting in loss of information on the noise distribution. For the $\lambda$ parameter, a value of either 0.3 or 0.5 was chosen on the development set giving the group clusters more or equal weight compared to the global confusion matrix. This shows that the feature dependent noise matrices are important and have a positive impact on performance.

Five confusion matrices for groups and the global matrix in the English data are shown as examples in Figure 2. One can see that the noise matrix can visibly differ depending on the cluster of the input word. Some of these differences can also be directly explained by flaws in the distant supervision method. The automatic annotation did not label any locations written in all upper-case letters as locations. Therefore, the noise distribution for all upper-cased locations differs from the distribution of other location names (cf. 2d and 2c). The words April and June are used both as names for a month and as first names in English. This results in a very specific noise distribution with many temporal expressions being annotated as person entities (cf. 2b). Similar to this, first-person names and also Asian words are likely to be labeled as persons by the automatic annotation method (cf. 2f and 2e).

All of these groups show traits that are not displayed in the global matrix, allowing the cluster-based models to outperform the other systems.

## 8 Conclusions

We have shown that the noise models with feature-dependent confusion matrices can be used effectively in practice. These models improve low-resource named entity recognition with noisy labels beyond all other tested baselines. Further, the feature-dependent confusion matrices are task-independent and could be used for other NLP tasks, which is one possible direction of future research.

## Acknowledgments

The authors would like to thank Heike Adel, Annemarie Friedrich, Jannik Strötgen and the anonymous reviewers for their helpful comments. This work has been partially funded by Deutsche Forschungsgemeinschaft (DFG) under grant SFB 1102: Information Density and Linguistic Encoding.

## References

Alan Joseph Bekker and Jacob Goldberger. 2016. Training deep neural-networks based on unreliable labels. In *Proceedings of the 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

A. P. Dawid and A. M. Skene. 1979. Maximum likelihood estimation of observer error-rates using the em algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):20–28.

Julia Dembowski, Michael Wiegand, and Dietrich Klakow. 2017. Language independent named entity recognition using distant supervision. In *Proceedings of Language and Technology Conference*.

Y. Dgani, H. Greenspan, and J. Goldberger. 2018. Training a neural network based on unreliable human annotation of medical images. In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pages 39–42.

Timothy Dozat. 2016. Incorporating nesterov momentum into adam. In *Workshop Track of the International Conference on Learning Representations (ICLR)*.

Meng Fang and Trevor Cohn. 2016. Learning when to trust distant supervision: An application to low-resource pos tagging using cross-lingual projection. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*.

B. Frenay and M. Verleysen. 2014. Classification in the presence of label noise: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 25(5):845–869.

Jacob Goldberger and Ehud Ben-Reuven. 2016. Training deep neural-networks using a noise adaptation layer. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Michael A. Hedderich and Dietrich Klakow. 2018. Training a neural network in a low-resource setting on automatically annotated noisy data. In *Proceedings of the Workshop on Deep Learning Approaches for Low-Resource NLP*. Association for Computational Linguistics.

Dan Hendrycks, Mantas Mazeika, Duncan Wilson, and Kevin Gimpel. 2018. Using trusted data to train deep networks on labels corrupted by severe noise. In *Advances in Neural Information Processing Systems 31*, pages 10477–10486. Curran Associates, Inc.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of the Tenth Machine Translation Summit*, volume 5, pages 79–86.

S. P. Lloyd. 1982. Least squares quantization in PCM. In *IEEE Transactions on Information Theory*.

Bingfeng Luo, Yansong Feng, Zheng Wang, Zhanxing Zhu, Songfang Huang, Rui Yan, and Dongyan Zhao. 2017. Learning with noise: Enhance distantly supervised relation extraction with dynamic transition matrix. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Volodymyr Mnih and Geoffrey Hinton. 2012. Learning to label aerial images from noisy data. In *Proceedings of the 29th International Conference on Machine Learning*, ICML'12.

Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. 2017. Making deep neural networks robust to label noise: A loss correction approach. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1944–1952.

Debjit Paul, Mittul Singh, Michael A. Hedderich, and Dietrich Klakow. 2019. Handling noisy labels for robustly learning from self-training data for low-resource sequence labeling. *CoRR*, abs/1903.12008.

Karl Pearson. 1901. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572.

Afshin Rahimi, Yuan Li, and Trevor Cohn. 2019. Multilingual NER transfer for low-resource languages. *CoRR*, abs/1902.00193.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the SIGNLL Conference on Computational Natural Language Learning (CoNLL 2009)*.

Sainbayar Sukhbaatar, Joan Bruna, Manohar Paluri, Lubomir Bourdev, and Rob Fergus. 2015. Training convolutional networks with noisy labels. In *Workshop Track of the International Conference on Learning Representations (ICLR)*.

Erik F. Tjong Kim Sang. 2002. Introduction to the conll-2002 shared task: Language-independent named entity recognition. In *Proceedings of the 6th Conference on Natural Language Learning - Volume 20*, COLING-02, pages 1–4, Stroudsburg, PA, USA. Association for Computational Linguistics.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 142–147, Stroudsburg, PA, USA. Association for Computational Linguistics.

Alexander Tkachenko, Timo Petmanson, and Sven Laur. 2013. Named entity recognition in estonian. In *Proceedings of the 4th Biennial International Workshop on Balto-Slavic Natural Language Processing, BSNLP@ACL*.

Andreas Veit, Neil Alldrin, Gal Chechik, Ivan Krasin, Abhinav Gupta, and Serge J. Belongie. 2017. Learning from noisy large-scale datasets with minimal supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. 2015. Learning from massive noisy labeled data for image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2691–2699.

Yaosheng Yang, Wenliang Chen, Zhenghua Li, Zhengqiu He, and Min Zhang. 2018. Distantly supervised ner with partial annotation learning and reinforcement learning. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*, pages 2159–2169.

# A Multi-Pairwise Extension of Procrustes Analysis for Multilingual Word Translation

**Hagai Taitelbaum**
Faculty of Engineering
Bar-Ilan University, Israel
hagait62@gmail.com

**Gal Chechik**
The Gonda Brain Research Center
Bar-Ilan University, NVIDIA
gal.chechik@biu.ac.il

**Jacob Goldberger**
Faculty of Engineering
Bar-Ilan University, Israel
jacob.goldberger@biu.ac.il

## Abstract

In this paper we present a novel approach to simultaneously representing multiple languages in a common space. Procrustes Analysis (PA) is commonly used to find the optimal orthogonal word mapping in the bilingual case. The proposed *Multi Pairwise Procrustes Analysis* (MPPA) is a natural extension of the PA algorithm to multilingual word mapping. Unlike previous PA extensions that require a $k$-way dictionary, this approach requires only pairwise bilingual dictionaries that are much easier to construct in either a supervised or an unsupervised way. The improved performance of the MPPA algorithm is demonstrated on two standard multilingual tasks.

## 1 Introduction

Continuous word embeddings have been proved effective in numerous NLP applications. In cross-language tasks, these vector-space representations have recently emerged as a tool to transfer knowledge from one language to another. Specifically, several studies have suggested forming cross-lingual embeddings by learning a linear mapping from a source-language embedding space to a target-language one and demonstrated the benefits of this approach for word translation (Mikolov et al., 2013; Klementiev et al., 2012). Xing et al. (2015) showed that imposing orthogonality constraints on the linear mapping between spaces can alleviate overfitting. Building on these concepts, several studies have aimed to improve these bilingual word embeddings using bilingual word dictionaries that are created in either a supervised or an unsupervised manner (Artetxe et al., 2017a). Bilingual word embedding were found to be useful in a number of monolingual and cross-lingual NLP tasks (Vulic and Moens, 2015; Tsai and Roth, 2016).

Bilingual embedding can be extended to a multilingual setup by jointly learning mappings from each monolingual word embedding to a shared vector space. Modeling multiple languages jointly has been shown to improve modeling accuracy on bilingual tasks because it can utilize knowledge learned from the other languages (Ammar et al., 2016; Duong et al., 2017; Taitelbaum et al., 2019).

Extending the bilingual setup to a multilingual setting poses new challenges. For bilingual embedding, the word-mapping problem has a closed-form solution known as Orthogonal Procrustes Analysis (PA), which can be computed using singular value decomposition (Schnemann, 1966). However, there is no similar closed-form solution for the multi-language case. The standard extension of PA to multi-set alignment is Generalized Procrustes Analysis (GPA) (Gower, 1975) which is an iterative greedy algorithm. GPA was recently used to jointly transform multiple languages into a shared vector space (Kementchedjhieva et al., 2018). However, GPA assumes that a multi-way word correspondence is available, which is often not the case. Building a multi-way dictionary is a challenging task in itself.

In this study, we propose a novel efficient approach for mapping multiple languages simultaneously into a shared vector space, while enforcing orthogonality constraints. This approach, *Multi Pairwise Procrustes Analysis* (MPPA) can be viewed as a multilingual extension of the Procrustes Analysis. Unlike GPA-based approaches, MPPA does not require a multi-way dictionary, but only bilingual dictionaries which are much easier to obtain even in an unsupervised manner. We evaluated MPPA on two standard multilingual tasks and report better results than GPA based methods and competitive results with gradient based methods.

Our main contribution is a new, efficient, and



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3560–3565,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

easy-to-use algorithm for solving the extension of the Orthogonal Procrustes problem to the multilingual case. Our project code will be publicly available.

## 2 A Multi-Pairwise Extension of Procrustes Analysis

We first briefly review Procrustes Analysis (PA), a procedure to find the best orthogonal mapping between two languages. We then describe our approach, *Multi-Pairwise Procrustes Analysis* (MPPA), which extends PA to the multilingual case.

Assume we are given $d$-dimensional word embedding data from two languages along with a dictionary consisting of pairs of corresponding words. Mikolov et al. (2013) showed that there is a strong linear correlation between the vector spaces of two languages and that learning a complex non-linear neural mapping does not yield better results than with a linear mapping. Xing et al. (2015) further showed that enforcing the linear mappings to be orthogonal matrices reduces overfitting and improves performance. We can learn the orthogonal mapping $T$ by minimizing the following cost function:

$$S(T) = \sum_{t=1}^{n} \|Tx_t - y_t\|^2 , \qquad (1)$$

where $x_t$ and $y_t$ are embeddings pf corresponding words from the two languages and $n$ is the dictionary size. Schnemann (1966) proved that the solution to Eq. (1), obtained as the result of a Procrustes Analysis algorithm, is $T = UV^\top$, where $U\Sigma V^\top$ is the singular value decomposition (SVD) of the $d \times d$ matrix $M = \sum_t y_t x_t^\top$. This method has been used in many recent cross-lingual studies (Xing et al., 2015; Artetxe et al., 2016, 2017a,b, 2018a,b; Hamilton et al., 2016; Conneau et al., 2017; Ruder et al., 2018).

Assume we are given $d$-dimensional word embedding data from $k$ languages and that each pair of languages is provided with a dictionary composed of pairs of corresponding words from the two languages. We could learn a mapping for each language pair independently as a solution to Eq. (1). However, this approach does not benefit from the multilingual setup. Another approach would be to choose one of the languages as a "pivot" and learn a mapping from each language to the pivot separately. A typical choice for the pivot, used in publicly available aligned vectors, is English (Conneau et al., 2017; Joulin et al., 2018). This strategy, however, does not guarantee that the indirect word translation between language pairs will have high quality. Alternatively, we can enforce a transitivity constraint by mapping all the embedding spaces to a shared vector space. Our goal in the multilingual case is thus to find the orthogonal matrices $T_1, ..., T_k$ such that pairs of corresponding words from different languages are mapped into close vectors in the shared space. Formally, we want to minimize the following mean-square error score:

$$S(T_1, ..., T_k) = \sum_{i<j} \sum_{t=1}^{n_{ij}} \|T_i x_{ij,t} - T_j x_{ji,t}\|^2 \quad (2)$$

where $(x_{ij,t}, x_{ji,t})$ is a pair of corresponding words in the $i$ and $j$ languages, respectively and $n_{ij}$ is the dictionary size. We use this notation to emphasize that the vocabularies of the same language in different dictionaries are not necessarily the same.

When more than two languages are involved there is no closed-form solution to the global minimum of (2). We propose an efficient algorithm for minimizing it. The basic step is optimizing the score (2) with respect to the mapping $T_i$ while keeping all other mappings fixed. Viewing the objective score (2) as a function of $T_i$ we obtain:

$$S(T_i) = \sum_{j \neq i} \sum_{t=1}^{n_{ij}} \|T_i x_{ij,t} - y_{j,t}\|^2 + \text{const.} \quad (3)$$

where $y_{j,t} = T_j x_{ji,t}$ is the representation of $x_{ji,t}$ in the common space. This is exactly the Orthogonal Procrustes problem (1) of finding a mapping from language $i$ into the common space. The optimal orthogonal matrix $T_i$ is thus obtained by $T_i = UV^\top$ s.t. $U\Sigma V^\top$ is the SVD of the matrix:

$$M_i = \sum_{j \neq i} \sum_{t} T_j x_{ji,t} x_{ij,t}^\top . \qquad (4)$$

Once $T_i$ is updated we move to the next language in a circular manner. At each step in the iterative algorithm, the score (2) is monotonically decreased until it converges to a local minimum point. Hence, we can stop the optimization procedure once there is no significant improvement in the objective score (2).

Each iteration is very costly since we need to go over all the dictionary words. To avoid this,

we can compute cross correlation $d \times d$ matrix for each pair of languages $i, j$ in a preprocessing step:

$$C_{ij} = C_{ji}^{\top} = \sum_{t} x_{ji,t} x_{ij,t}^{\top}. \quad (5)$$

Substituting (5) in (4) we obtain that

$$M_i = \sum_{j \neq i} T_j C_{ij}. \quad (6)$$

Therefore, updating the mapping $T_i$ can be done in a very efficient way without going over all the bilingual dictionaries of the $i$-th language.

**Algorithm 1** Multi Pairwise Procrustes Analysis

**Required:** A set of lexical of word pairs between each pair of languages.

**Task:** Find a set of orthogonal mappings $T_1, .., T_k$, to a common space.

Compute cross-correlation matrices:

$$C_{ij} = C_{ji}^{\top} = \sum_{t} x_{ji,t} x_{ij,t}^{\top} \qquad 1 \leq i < j \leq k$$

**Initialization:** $T_1 = I$,
**for** $i = 2, ..., k$ **do**
  $U\Sigma V^{\top}$ = SVD($\sum_{j<i} T_j C_{ij}$)
  $T_i \leftarrow UV^{\top}$
**end**
**Algorithm:**
**while** *not converged* **do**
  **for** $i = 1, ..., k$ **do**
    $U\Sigma V^{\top}$ = SVD($\sum_{j \neq i} T_j C_{ij}$)
    $T_i \leftarrow UV^{\top}$
  **end**
**end**

The proposed Multi Pairwise Procrustes Analysis (MPPA) word mapping training procedure is depicted in Algorithm box 1. The algorithm description also contains an initialization procedure that can help avoid getting stuck at local optima. The idea of the initialization is aligning each new language $i$ to the current common space which was built with languages $j < i$.

MPPA requires only pairwise bilingual dictionaries. It is applicable even if we only have dictionaries for a subset of all the language pairs such that each language under consideration is represented in at least one bilingual dictionary. Consider a graph whose vertices are the languages and an edge indicates the existence of a dictionary between two languages. It can easily be seen that if the graph is loop-free (as in the case where we only have dictionaries for a pivot language) the optimization of (2) is decoupled and each bilingual mapping can be learned separately. The task become really multi-lingual once the graph is loopy, where mapping transitivity implies that there is more than a single path between the source and target languages. We note in passing that we can consider the word representation in the common space as a latent variable and the mapping matrices as unknown parameters. The MPPA algorithm can be thus viewed as an instance of the EM algorithm (Dempster et al., 1977). Further discussion regarding the connection between MPPA algorithm and the EM algorithm can be found in (Goldberger, 1999).

| | he | af | oc | et | bs | Avg. |
|---|---|---|---|---|---|---|
| PA | 37.5 | 28.9 | 17.1 | 30.0 | 22.4 | 27.2 |
| MGPA | 37.5 | 28.9 | 23.8 | 30.7 | 21.0 | 28.4 |
| MPPA | 40.2 | 32.1 | 25.4 | 35.5 | 26.2 | **31.9** |

Table 1: p@1 for low resource languages: Hebrew, Afrikaans, Occitan, Estonian, and Bosnian, trained with multilingual algorithms over triplets.

## 3 Related work

The standard extension of PA to multi-set alignment is *Generalized Procrustes Analysis* (GPA) (Gower, 1975). Kementchedjhieva et al. (2018) recently proposed the Multi-support GPA (MGPA) algorithm for multilingual word translation which is based on the GPA. Their algorithm requires a $k$-way dictionary in the form of $(x_{it})$ where $(x_{1t}, ..., x_{kt})$ are representations of words that share the same semantic meaning across all the $k$ languages. This multi-way dictionary is constructed from the bilingual dictionaries (Kementchedjhieva et al., 2018). Whereas conflating multiple senses of a word is already problematic for bilingual dictionaries, this issue is amplified in a multilingual vocabulary. In our approach we avoid this form of error-prone data processing that consists of finding a joint translation of a single word across all the languages. Instead, the MPPA algorithm uses the bilingual dictionaries directly. Note that MPPA is an extension of the GPA algorithm. In case we are given a multi-way dictionary GPA and MPPA optimize the same cost function and MPPA can be viewed as an efficient alterna-

| | en-de | en-fr | en-es | en-it | en-pt | de-en | de-fr | de-es | de-it | de-pt | fr-en | fr-de | fr-es | fr-it | fr-pt | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PA | 73.5 | 81.1 | 81.4 | 77.3 | 79.9 | 72.4 | 73.3 | 67.7 | 69.5 | 59.1 | 82.4 | 69.5 | 82.6 | 83.2 | 78.1 | |
| MAT+MPPA | 74.5 | 82.7 | 82.2 | 78.5 | 81.3 | 72.9 | 75.2 | 68.0 | 70.1 | 61.1 | 82.2 | 69.0 | 83.6 | 83.1 | 78.7 | |
| MAT+MPSR | 74.8 | 82.4 | 82.5 | 78.8 | 81.5 | 72.9 | 76.7 | 69.6 | 72.0 | 63.2 | 81.8 | 71.2 | 83.9 | 83.5 | 79.3 | |
| UMH | 75.1 | 82.7 | 82.5 | 78.9 | 82.0 | 75.5 | 73.5 | 67.2 | 68.7 | 59.0 | 83.1 | 69.8 | 82.7 | 82.5 | 77.5 | |
| | es-en | es-de | es-fr | es-it | es-pt | it-en | it-de | it-fr | it-es | it-pt | pt-en | pt-de | pt-fr | pt-es | pt-it | Avg. |
| PA | 82.9 | 68.3 | 85.8 | 83.5 | 87.3 | 76.9 | 67.5 | 87.1 | 87.3 | 81.0 | 80.3 | 63.7 | 84.3 | 91.5 | 81.1 | 78.0 |
| MAT+MPPA | 83.5 | 66.5 | 85.9 | 83.7 | 86.8 | 77.7 | 67.1 | 87.7 | 87.5 | 81.2 | 80.2 | 63.7 | 84.6 | 92.2 | 82.6 | 78.5 |
| MAT+MPSR | 83.7 | 69.0 | 86.9 | 84.5 | 87.8 | 77.4 | 69.5 | 88.1 | 88.2 | 82.3 | 79.9 | 65.7 | 86.3 | 92.7 | 82.6 | **79.3** |
| UMH | 85.3 | 68.7 | 85.1 | 83.3 | 86.3 | 79.9 | 67.5 | 86.7 | 87.0 | 80.4 | 82.1 | 64.4 | 83.6 | 91.7 | 81.1 | 78.5 |

Table 2: Multilingual word translation results for English, German, French, Spanish, Italian and Portuguese. The reported numbers are precision@1 in percentage.

tive to the GPA optimization procedure.

Another line of research applies stochastic gradient-based optimization methods to minimize the mean-square error score (2) jointly with refinement of the bilingual dictionaries. The gradient is approximated by sampling word pairs from the bilingual dictionaries. Chen and Cardie (2018) proposed the *Multilingual Pseudo Supervised Refinement* (MPSR) for this minimization task that uses simple gradient methods in order to minimize (2). For unsupervised setup Chen and Cardie (2018) used an adversarial initialization step, *Multilingual Adversarial Training* (MAT). Alaux et al. (2019) presented, *Unsupervised Multilingual Hyperalignment* (UMH), a similar algorithm that extends the bilingual methods proposed by Grave et al. (2018); Alvarez-Melis and Jaakkola (2018), to multilingual setup.

A main difference between UMH (Alaux et al., 2019) and MAT+MPSR (Chen and Cardie, 2018) is how they treat orthogonality. The first is a stochastic gradient optimization followed by a projection on the set of orthogonal matrices. In the second method orthogonality is a regularization term that is optimized by gradient methods. The matrices are encouraged to be orthogonal by an orthogonalization update (Cisse et al., 2017) that yields matrices that are close to orthogonal but are not necessarily exactly orthogonal. In contrast to gradient based methods, our approach avoids word sampling and hyper-parameters that need to be tuned.

## 4 Experiments

**Datasets and embeddings** We used the MUSE benchmark (Conneau et al., 2017)[1], which consist of bilingual dictionaries of 5000 unique source word for training and 1500 for testing. The fastText embeddings (Bojanowski et al., 2017) trained on Wikipedia data, are available online[2]. Vectors were normalized to unit length and then zero centered (Artetxe et al., 2016).

**Compared methods** We compared MPPA to MGPA (Kementchedjhieva et al., 2018), MAT+MPSR (Chen and Cardie, 2018) and UMH (Alaux et al., 2019). We used the task and results reported in the corresponding paper (UMH results are from the appendix). All methods ran several refinement epochs (Artetxe et al., 2017a), where after each refinement iteration dictionaries were re-build, as described in Conneau et al. (2017). Model selection was done by the best validation criterion suggested in Conneau et al. (2017) and extended in Chen and Cardie (2018). All these methods retrieve word translation using the *Cross-domain Similarity Local Scaling* (CSLS) criterion (Lample et al., 2018).

**Results** The first experiment was conducted over language triplets (Kementchedjhieva et al., 2018). The goal is to translate from English to a low resource language (like Bosnian) using a high resource language (like Russian). As in Kementchedjhieva et al. (2018), 10 refinement epochs were used, and initial dictionaries for each language pair were generated by pairs of words with identical string matching.

Table 1 depicts precision@1 for the triplets task. MPPA outperformed MGPA and both outperform PA. Note that MGPA needs a multi-way dictionary constructed from the bilingual dictionaries. In contrast, MPPA uses directly the raw data (the bilingual dictionaries).

The second experiment involved multilingual word translation in six European languages: English, German, French, Spanish, Italian and Por-

[1] https://github.com/facebookresearch/MUSE

[2] https://github.com/facebookresearch/fastText

tuguese (Lample et al., 2018). We compared MPPA to MAT+MPSR (Chen and Cardie, 2018). MAT+MPSR is an unsupervised method, so for a fair comparison we replaced the MPSR algorithm with our MPPA algorithm, thus obtaining MAT+MPPA. We ran 5 refinement epochs, after the MAT step, as the default option in MAT+MPSR source code[3]. MPPA training phase is 10 times faster than MPSR equivalent phase, which also have hyper-parameters that needed to be tuned. UMH (Alaux et al., 2019), was also evaluated on this benchmark.

Table 2 shows precision@1 results. MPPA was comparable to UMH and MPSR performed slightly better. Note that the MPSR mapping matrices were not exactly orthogonal. They indeed achieved smaller mean-square error (2) on the training data than our solution, which was restricted to be orthogonal. This suggests that the orthogonality constraint, especially in the multilingual case where it is combined with transitivity constraints, can be too restrictive.

## 5 Conclusion

This paper presents a general approach to map word embeddings into a common space that can be viewed as an extension of PA to the multilingual case. The proposed algorithm efficiently avoids the need to go over the whole dictionary at each iteration. The optimization is done by enforcing both transitivity and orthogonal constraints. A possible future research direction would involve finding efficient optimization methods where the orthogonality constraint could be slightly relaxed.

## References


Jean Alaux, Edouard Grave, Marco Cuturi, and Armand Joulin. 2019. Unsupervised hyperalignment for multilingual word embeddings. In *International Conference on Learning Representations*.

David Alvarez-Melis and Tommi S Jaakkola. 2018. Gromov-wasserstein alignment of word embedding spaces. *arXiv preprint arXiv:1809.00013*.

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A Smith. 2016. Massively multilingual word embeddings. *arXiv preprint arXiv:1602.01925*.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2289–2294.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017a. Learning bilingual word embeddings with (almost) no bilingual data. In *Annual Meeting of the Association for Computational Linguistics*, pages 451–462. Association for Computational Linguistics.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018a. Generalizing and improving bilingual word embedding mappings with a multi-step framework of linear transformations. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018b. A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. *arXiv preprint arXiv:1805.06297*.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2017b. Unsupervised neural machine translation. *arXiv preprint arXiv:1710.11041*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Xilun Chen and Claire Cardie. 2018. Unsupervised multilingual word embeddings. In *Conference on Empirical Methods in Natural Language Processing*.

Moustapha Cisse, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. 2017. Parseval networks: Improving robustness to adversarial examples. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 854–863. JMLR. org.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B (Methodological)*, 39:1–38.

Long Duong, Hiroshi Kanayama, Tengfei Ma, Steven Bird, and Trevor Coh. 2017. Multilingual training of crosslingual word embeddings. In *The Conference of the European Chapter of the Association for Computational Linguistics*.

Jacob Goldberger. 1999. Registration of multiple point sets using the EM algorithm. In *IEEE International Conference on Computer Vision*.

John C Gower. 1975. Generalized procrustes analysis. *Psychometrika*, 40(1):33–51.


[3] https://github.com/ccsasuke/umwe

Edouard Grave, Armand Joulin, and Quentin Berthet. 2018. Unsupervised alignment of embeddings with wasserstein procrustes. *CoRR*, abs/1805.11222.

William L Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Diachronic word embeddings reveal statistical laws of semantic change. *arXiv preprint arXiv:1605.09096*.

Armand Joulin, Piotr Bojanowski, Tomas Mikolov, Herve Jegou, , and Edouard Grave. 2018. Loss in translation: Learning bilingual word mapping with a retrieval criterion. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Yova Kementchedjhieva, Sebastian Ruder, Ryan Cotterell, and Anders Søgaard. 2018. Generalizing procrustes analysis for better bilingual dictionary induction. In *CONLL*.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing crosslingual distributed representations of words. In *COLING*, pages 1459–1474.

Guillaume Lample, Alexis Conneau, Marc-Aurelio Ranzato, Ludovic Denoyer, and Herv Jgou. 2018. Word translation without parallel data. In *International Conference on Learning Representation*.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Sebastian Ruder, Ryan Cotterell, Yova Kementchedjhieva, and Anders Søgaard. 2018. A discriminative latent-variable model for bilingual lexicon induction. *arXiv preprint arXiv:1808.09334*.

Peter Schnemann. 1966. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10.

Hagai Taitelbaum, Gal Chechik, and Jacob Goldberger. 2019. Multilingual word translation using auxiliary languages. In *Conference on Empirical Methods in Natural Language Processing*.

Chen-Tse Tsai and Dan Roth. 2016. Cross-lingual wikification using multilingual embeddings. In *NAACL-HLT*.

Ivan Vulic and Marie-Francine Moens. 2015. Monolingual and cross-lingual information retrieval models based on (bilingual) word embedding. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, page 363372.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1006–1011.

# Out-of-Domain Detection for Low-Resource Text Classification Tasks

**Ming Tan** [*†] **Yang Yu** [*†] **Haoyu Wang** [*†] **Dakuo Wang** [‡]
**Saloni Potdar** [†] **Shiyu Chang** [‡] **Mo Yu** [‡]
[†] IBM Watson [‡] IBM Research

## Abstract

Out-of-domain (OOD) detection for low-resource text classification is a realistic but understudied task. The goal is to detect the OOD cases with limited in-domain (ID) training data, since we observe that training data is often insufficient in machine learning applications. In this work, we propose an *OOD-resistant Prototypical Network* to tackle this **zero-shot OOD detection and few-shot ID classification** task. Evaluation on real-world datasets show that the proposed solution outperforms state-of-the-art methods in zero-shot OOD detection task, while maintaining a competitive performance on ID classification task.

## 1 Introduction

Text classification tasks in real-world applications often consists of 2 components- In-Doman (ID) classification and Out-of-Domain (OOD) detection components (Liao et al., 2018; Kim and Kim, 2018; Shu et al., 2017; Shamekhi et al., 2018). ID classification refers to classifying a user's input with a label that exists in the training data, and OOD detection refers to designate a special OOD tag to the input when it does not belong to any of the labels in the ID training dataset (Dai et al., 2007). Recent state-of-the-art deep learning (DL) approaches for OOD detection and ID classification task often require massive amounts of ID or OOD labeled data (Kim and Kim, 2018). In reality, many applications have very limited ID labeled data (i.e., few-shot learning) and no OOD labeled data (i.e., zero-shot learning). Thus, existing methods for OOD detection do not perform well in this setting.

One such application is the intent classification for conversational AI services, such as IBM Watson Assistant[1]. For example, Table 1 shows some of the utterances a chat-bot builder provided for training. Each class may only have less than 20 training utterances, due to the high cost of manual labelling by domain experts. Meanwhile, the user also expects the service to effectively reject irrelevant queries (as shown at the bottom of Table 1). The challenge of OOD detection is reflected by the undefined in-domain boundary. Although one can provide a certain amount of OOD samples to build a binary classifier for OOD detection, such samples may not efficiently reflect the infinite OOD space. Recent approaches, such as (Shu et al., 2017), make remarkable progress on OOD detection with only ID examples. However, such condition on ID data cannot be satisfied by the few-shot scenario presented in Table 1.

| Intent Label | Example |
|---|---|
| Help_List | List what you can help me with. |
| | Watson, I need your help |
| Schedule_Appointment | Can you book a cleaning with my dentist for me? |
| | Can you schedule my dentist's appointment? |
| End_Meeting | You can end the meeting now |
| | Meeting is over |
| ... | ... |
| OOD utterances | My birthday is coming! |
| | blah blah... |

Table 1: A few-shot ID training set for a conversation service for teleconference management, with OOD testing examples.

This work aims to build a model that can detect OOD inputs with limited ID data and zero OOD training data, while classifying ID inputs with a high accuracy. Learning similarities with

[*] Equal contributions from the corresponding authors: {mingtan,yu,wanghaoy}@us.ibm.com.

[1] https://www.ibm.com/cloud/watson-assistant/



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3566–3572, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

the meta-learning strategy (Vinyals et al., 2016) has been proposed to deal with the problem of limited training examples for each label (few-shot learning). In this line of work, *Prototypical Networks* (Snell et al., 2017), which was originally introduced for few-shot image classification, has proven to be promising for few-shot ID text classification (Yu et al., 2018). However the usage of prototypical network for OOD detection is unexplored in this regard.

To the best of our knowledge, this work is the first one to adopt a meta-learning strategy to train an OOD-Resistant Prototypical Network for simultaneously detecting OOD examples and classifying ID examples. The contributions of this work are two-fold: 1) Unified solution using a prototypical network model which can detect OOD instances and classify ID instances in a real-world low-resource scenario. 2) Experiments and analysis on two datasets prove that the proposed model outperforms previous work on the OOD detection task, while maintaining a state-of-the-art ID classification performance.

## 2 Related Work

**Out-of-Domain Detection** Existing methods often formulate the OOD task as a one-class classification problem, then use appropriate methods to solve it (e.g., one-class SVM (Schölkopf et al., 2001) and one-class DL-based classifiers (Ruff et al., 2018; Manevitz and Yousef, 2007). A group of researchers also proposed an auto-encoder-based approach and its variation to tackle OOD tasks (Ryu et al., 2017, 2018). Recently, a few papers have investigated ID classification and OOD detection simultaneously (Kim and Kim, 2018; Shu et al., 2017), but they fail in a low resource setting.

**Few-Shot Learning** While few-shot learning approaches may help with this low-resource setting, some recent work is promising in this regard. For example, (Vinyals et al., 2016; Bertinetto et al., 2016; Snell et al., 2017) use metric learning by learning a good similarity metric between input examples; some other methods adapt a meta-learning framework, and train the model to quickly adapt to new tasks with gradients on small samples, e.g., learning the optimization step sizes (Ravi and Larochelle) or model initialization (Finn et al., 2017). Though most of these approaches are explored for computer vision, recent studies suggests that few-shot learning is promising in the text domain, including text classification (Yu et al., 2018; Jiang et al., 2018), relation extraction (Han et al., 2018), link prediction in knowledge bases (Xiong et al., 2018) and fine-grained entity typing (Xiong et al., 2019), and we put it to test with the OOD detection task.

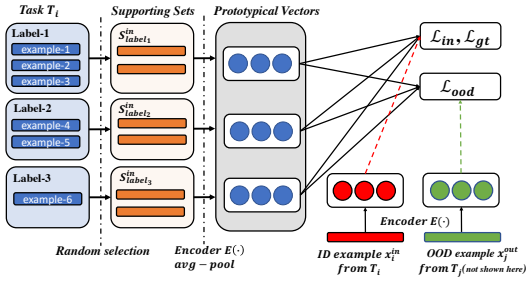


Figure 1: Model overview: the model maximizes likelihood of the ground-truth of ID example, minimizes distance between ID example and ground-truth, and maximizes the distance of OOD example and all ID labels.

## 3 Approach

In this paper, we target solving the zero-shot OOD detection problem for a few-shot **meta-test** dataset $D = (D^{train}, D^{test})$ by training a transferable prototypical network model from large-scale **independent source datasets** $\mathcal{T} = \{T_1, T_2, ..., T_N\}$ for dynamic construction of the **meta-train** set. Each task $T_i$ contains labeled training examples (note that a test set is not required in meta-train). $D$ is different from the traditional supervised close-domain classification dataset from two folds: 1) $D^{test}$ contains OOD testing examples, whereas $D^{train}$ only includes labeled examples for the target domain. 2) The training size for each label in $D^{train}$ is limited (e.g. less than 100 examples). Such limitations prevent existing methods from efficiently training a model for either ID classification or OOD detection using $D^{train}$ only.

We propose an OOD-resistant prototypical network for both OOD detection and few-shot ID classification. We follow (Snell et al., 2017) in few-shot image classification by training a prototypical network on $\mathcal{T}$ and directly perform prediction on $D$ without additional training. But our method is different from the prior work in that during the meta-training, while we maximize the likelihood of the true label for an example in $T_i$, we also sample an example from another meta-train task $T_j$ for the purpose of OOD training by maximizing the distance between the OOD instance and the prototypical vector of each ID label.

### 3.1 General Framework

As in Fig. 1, on a large-scale source dataset $\mathcal{T}$ with the following steps:

1. Sample a training task $T_i$ from $\mathcal{T}$ (e.g., the Book category of Amazon Review in Section 4), and another task $T_j$ from $\mathcal{T} - T_i$ (e.g. the Apps-for-Android category).
2. Sample an ID training example $x_i^{in}$ from $T_i$, and a simulated OOD example $x_j^{out}$ from $T_j$.
3. Sample $N$ labels ($N$=4) from $T_i$ in addition to the label of $x_i^{in}$. For the ground-truth label and $N$ negative labels, we select $K$ training examples for each label ($K$-shot learning, we set $K$=20). If a label has less than $K$ examples, we replicate the selected example to satisfy $K$. Therefore, $(N+1) \times K$ examples serve as a supporting set $\mathcal{S}^{in} = \{S_l^{in}\}_{l=1}^{N}$.
4. Given a batch of dynamically-constructed meta-train set $(x_i^{in}, x_j^{out}, \mathcal{S}^{in})$, $E(\cdot)$ encodes $x_i^{in}$, $x_j^{out}$ and the examples in $S_l^{in}$ using a deep network (Any DL structure can be used for the encoder, such as LSTM and CNN. Here we use a one-layer CNN with a mean pooling. The detailed CNN hyper-parameters are introduced in Section 5).
5. Following (Snell et al., 2017), a Prototypical Vector representation for each label is generated, by averaging all the examples' representations of that label.
6. The model is optimized by an objective function, defined by $x_i^{in}$ , $x_j^{out}$ and $\mathcal{S}^{in}$. Details in Section 3.2.
7. Repeat these steps for multiple epochs (5k in this paper) to train the model, and select the best model based on an independent meta-valid set $\mathcal{T}^{valid}$. $\mathcal{T}^{valid}$ contains tasks that are homogeneous to the meta-test task $D$.

The only trainable parameters of this model are in the encoder $E(\cdot)$. Therefore the trained model can be easily transferred to the few-shot target domain.

### 3.2 Training Objective and Runtime

Prototypical networks (Snell et al., 2017) minimize a cross-entropy loss defined on the distance metrics between $x_i^{in}$ and the supporting sets,

$$\mathcal{L}_{in} = -\log \frac{\exp \alpha F(x_i^{in}, S_{l_i}^{in})}{\sum_{l'} \exp \alpha F(x_i^{in}, S_{l'}^{in})} \quad (1)$$

where $l_i$ is the ground-truth label of $x_i$, $\alpha$ is a re-scaling factor. Here we define $F$ as a cosine similarity score (mapped to the range between 0 and 1)[2] between the $E(\cdot)$-encoded representations of $x$ and the prototypical vector of a label. Our experiments show this meta-learning approach is efficient for ID classification, but is not good enough for detecting the OOD examples.

We propose two more training losses in addition to the $L_{in}$ for OOD detection. The rationale behind this addition is to adopt the examples from other tasks as simulated OOD examples for the current meta-train tasks. Specifically, we first define a hinge loss on $x_j^{out}$ and the closest ID supporting set in $\mathcal{S}^{in}$, then we push the examples from another task away from the prototypical vectors of ID supporting sets.

$$\mathcal{L}_{ood} = max[0, \max_{l}(F(x_j^{out}, S_l^{in}) - \mathcal{M}_1)] \quad (2)$$

We expect optimizing only on $\mathcal{L}_{in}$ and $L_{ood}$ will lead to lower confidences on ID classification, because the system tends to mistakenly reduce the scale of $F$ in order to minimize the loss for OOD examples. Therefore we add another loss to improve the confidence of classifed ID labels.

$$\mathcal{L}_{gt} = max[0, \mathcal{M}_2 - F(x_i^{in}, S_{l_i}^{in}))] \quad (3)$$

The model is optimized on the three losses.

$$\mathcal{L} = \mathcal{L}_{in} + \beta \mathcal{L}_{ood} + \gamma \mathcal{L}_{gt} \quad (4)$$

where $\alpha$, $\beta$, $\gamma$, $\mathcal{M}_1$ and $\mathcal{M}_2$ are hyper-parameters, whose detailed values are shown in Section 5 .

During inference, the supporting set per label is generated by averaging the encoded representations of all instances of that label in $D^{train}$ and the prediction is based on $F(x, S_l^{in})$. OOD detection is decided with a *confidence* threshold.

## 4 Datasets

Our methods are evaluated on two datasets and each has many tasks and is divided into meta-train, meta-valid and meta-test sets, which are respectively used for background model training, evaluation and hyper-parameter selection.

**Amazon Review** [3]**:** We follow (Yu et al., 2018) to construct multiple tasks using the Amazon Review dataset (He and McAuley, 2016) . We convert it into a binary classification task of labeling the review sentiment (positive/negative). It has 21

[2]Following (Snell et al., 2017), we also tried squared Euclidean distance, but did not achieve better results.

[3]We will release Amazon data and our code at https://github.com/SLAD-ml/few-shot-ood

categories of products, each of which is treated as a task. We randomly picked 13 categories as meta-train, 4 as meta-test and 4 as meta-valid. (another 3 original categories are discarded due to not enough examples to make a dataset). We construct a 2-way 100-shot problem per meta-test task by sampling 100 reviews per label in a category. For the test examples in meta-test and meta-valid, we sample other categories' examples as OOD, merged with a equal number of ID instances. We used all available data for meta-train.

**Conversation Dataset:** An intent classification dataset for a AI conversational system. It has 539 categories/tasks. We allocate 497 tasks as meta-train, and 42 tasks as meta-test. This dataset is different and more difficult than the typical ID few-shot learning data: 1) Both the meta-test and meta-train tasks are not restricted to $N$-way $K$-shot classification, and the source dataset is highly imbalanced across labels; 2) Each task has a variety of labels (utterance intents), whereas Amazon data always has two labels. There are 29% OOD testing instances in meta-test, which are human-labeled and not generated from other tasks.

## 5 Experimental Results

**Baselines:** We compare our model **O-Proto** with 4 baselines: 1) **OSVM (Schölkopf et al., 2001)**: OSVM is trained on meta-test set, and learn a domain boundary by only examining ID examples. 2) **LSTM-AutoEncoder (Ryu et al., 2017)**: Recent work on OOD detection that uses only ID examples to train an autoencoder for OOD detection. 3) **Vanilla CNN**: A classifier with a typical CNN structure that uses a confidence threshold for OOD detection. 4) **Proto. Network (Snell et al., 2017)**: A native prototypical network trained on $\mathcal{T}$ with only the loss $\mathcal{L}_{in}$, which uses a confidence threshold for OOD detection. We test the Proto. Network with both CNN and bidirectional LSTM as the encoder $E(\cdot)$.

**Hyper Parameters:** We introduce the hyper-parameters of our model and all baselines below.

We use Python scikit-learn One-Class SVM as the basis of our OSVM implementation. We use Radial Basis Function (RBF) as the kernel and the gamma parameter is set to auto. We use squared hinge loss and L2 regularization.

We follow the same architecture as proposed in (Ryu et al., 2017) for the LSTM-Autoencoder. In LSTM, we set the input embedding dimension as 100 and hidden as 200. We use RMSprop as the optimizer with a learning rate of 0.001. We train the LSTM with a batch size of 32 and 100 epochs. For Autoencoder, we set the hidden size as 20. We use Adam as the optimizer with a learning rate of 0.001. We train the model with a batch size of 32 for 10 epochs.

For vanilla CNN, we use the most common CNN architecture used in NLP tasks, where the convolutional layer on top of word embedding has 128 filters followed by a ReLU and max pooling layer before the final softmax. We use Adam as the optimizer with a learning rate of 0.001. We train the model with a batch size 64 for 100 epochs.

Our proposed model O-Proto uses the similar CNN architecture, the optimizer and the learning rate in the previous Vanilla CNN. The input word embeddings are pre-trained by 1-billion-token Wikipedia corpus. We set the batch size as 10. In Eq. 1, 2, 3 and 4, $\alpha$, $\beta$, $\gamma$, $\mathcal{M}_1$ and $\mathcal{M}_2$ are hyper-parameters, which we fix $\beta$, $\gamma$ as 1.0 by default, and set $\alpha$, $\mathcal{M}_1$ and $\mathcal{M}_2$ as 10.0, 0.4 and 0.8 according to the meta-valid performance of Amazon dataset. The sentence encoder, CNN, has 200 filters, followed by a tanh and mean pooling layer before the final regression layer. The maximum length of tokens per example is 40, and any words out of this range will be discarded. During training, we set the size of sampled negative labels Step 3 (section 3.1) to at most four, so there will be maximum five labels involved in a training step (1 positive, 4 negative). The supporting set size for each label are 20.

To make a fair comparison, we follow the same hyper-parameters as O-Proto in Proto. Network, except that the weight of $\mathcal{L}_{ood}$ and $\mathcal{L}_{gt}$, $\beta$ and $\gamma$, are set to zero.

**Evaluation Metrics:** Following (Ryu et al., 2017; Lane et al., 2007), we use a commonly used OOD detection metric, equal error rate (**EER**), which is the error rate when the confidence threshold is located where false acceptance rate (**FAR**) is equivalent to false rejection rate (**FRR**).

$$FAR = \frac{Number\ of\ accepted\ OOD\ sentences}{Number\ of\ OOD\ sentences}$$
$$FRR = \frac{Number\ of\ rejected\ ID\ sentences}{Number\ of\ ID\ sentences}$$

We use class error rate **CER** to reflect ID performance. Lastly, we applied the threshold used in the EER to ID test examples to test how many ID

| (%) | Conversation EER | CER | Comb. | Amazon Review EER | CER | Comb. |
|---|---|---|---|---|---|---|
| OSVM | 63.6 | - | - | 47.6 | - | - |
| LSTM AutoEnc. | 48.0 | 78.4 | 79.5 | 45.4 | 29.3 | 38.6 |
| Vanilla CNN | 26.4 | 76.8 | 77.6 | 47.7 | 34.4 | 42.8 |
| Proto. Network | 26.9 | 32.5 | 44.5 | 46.5 | **7.3** | 47.6 |
| O-Proto ($\mathcal{L}_{in} + \mathcal{L}_{gt}$) | 27.6 | 33.3 | 46.2 | 47.8 | 7.4 | 48.9 |
| O-Proto ($\mathcal{L}_{in} + \mathcal{L}_{ood}$) | 24.5 | 30.1 | 41.2 | 24.7 | 9.7 | 30.1 |
| O-Proto (all) | **24.1** | **29.6** | **40.8** | **24.0** | 9.1 | **29.1** |
| Proto. with bilstm | 25.0 | 32.5 | 42.6 | 45.1 | **6.8** | 46.0 |
| O-Proto with bilstm | **22.0** | **30.5** | **39.8** | **21.9** | 9.0 | **27.1** |

Table 2: O-Proto is compared with other baselines for Conversation and Amazon data

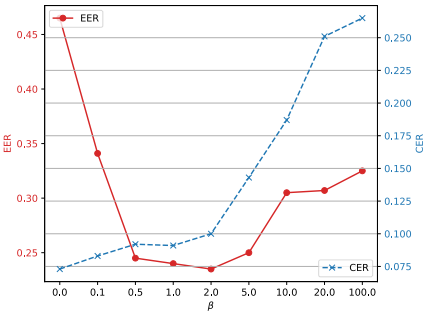


Figure 2: Various $\beta$ for Amazon data

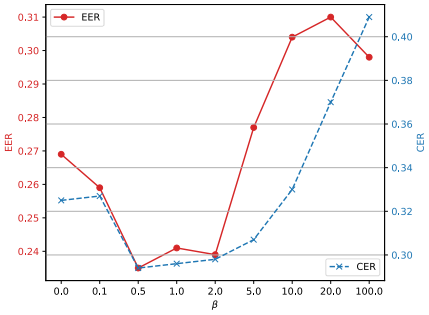


Figure 3: Various $\beta$ for Conversation data

| EER (%) | K=1 | =5 | =10 | =20 | =100 |
|---|---|---|---|---|---|
| Proto. | 53.9 | 46.4 | 44.5 | 41.0 | 46.5 |
| O-Proto. | 31.1 | 25.1 | 24.0 | 23.4 | 24.1 |

Table 3: Compare our model with Prototype Network on EER when choosing various $K$-shot values

examples are rejected by the OOD detection, as **Combined-CER** (Comb. in Table 2).

**Results:** Table 2 compares the proposed model and baselines on EER and CER on the two datasets. [4] For prototypical-network related models, we randomly initialize the model parameters by 10 times and report the averaged metrics. *OSVM, LSTM-AutoEncoder and Vanilla CNN* perform poorly on OOD detection as expected, as they require larger number of ID examples which is not a few-shot scenario. *Proto.Network* has a better performance on ID classification. But it is not designed for OOD, thus it does not perform well on OOD detection. *O-Proto* achieves significantly better EERs (2.8% improvement on Conversation, and 22.5% on Amazon), yielding competitive results on CER compared to *Proto.Network*. These lead to a remarkable improvement on Combined-CER. *O-Proto* improves less on EER in Conversation than Amazon, because some Conversation tasks actually come from the conversational service providers belonging to similar business domains. Our model is better even when meta-train datasets are from slightly different domains. Moreover, in Table 2 we show the ablation study by removing $\mathcal{L}_{ood}$ and $\mathcal{L}_{gt}$ from *O-Proto*, respectively. *O-Proto* without $\mathcal{L}_{ood}$ completely loses the ability of OOD detection. Compared to the one without $\mathcal{L}_{gt}$, *O-Proto* with all losses gives a mild improvement. We also observe a more stable testing performance among epochs during training. Finally, we replace the CNN encoders with bidirectional LSTMs (the bottom of Table 1), which yields the same dimension of sentence representations as CNN. For Conversation data, we achieve the best performance on validation set when $\alpha$ and $\gamma$ are 0.5. We observe comparable performances with respect to *Proto.Network* and *O-Proto*, showing that our proposed OOD approach is not limited to a specific sentence encoding architecture.

**Improvement in different K-shot settings:** On the Amazon data, we construct different $K$-shot tasks as meta-test (results shown in Table 3), and observe consistent improvements on EER.

**Effect of $\beta$:** Fig. 2 and 3 show EER and CER with different $\beta$ values on the two datasets. We observe within a proper range of $\beta$ (between 0.5 and 2.0), the model can provide stable improvement on EER, guaranteeing competitive CER results.

## 6 Conclusion

Inspired by the Prototypical Network, we propose a new method to tackle the OOD detection task in low-resource settings. Evaluation on real-world datasets demonstrates that our method performs favorably against state-of-the-art algorithms on the OOD detection, without adversely affecting performance on the few-shot ID classification.

[4] No CER reported for *OSVM* as it treats ID as one class and does not support ID classification.

## References


Luca Bertinetto, João F. Henriques, Jack Valmadre, Philip H. S. Torr, and Andrea Vedaldi. 2016. Learning feed-forward one-shot learners. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 523–531, USA. Curran Associates Inc.

Wenyuan Dai, Gui-Rong Xue, Qiang Yang, and Yong Yu. 2007. Co-clustering based classification for out-of-domain documents. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 210–219.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135, International Convention Centre, Sydney, Australia. PMLR.

Xu Han, Hao Zhu, Pengfei Yu, Ziyun Wang, Yuan Yao, Zhiyuan Liu, and Maosong Sun. 2018. Fewrel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4803–4809.

Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, pages 507–517.

Xiang Jiang, Mohammad Havaei, Gabriel Chartrand, Hassan Chouaib, Thomas Vincent, Andrew Jesson, Nicolas Chapados, and Stan Matwin. 2018. Attentive task-agnostic meta-learning for few-shot text classification. *The Second Workshop on Meta-Learning at NeurIPS*.

Joo-Kyung Kim and Young-Bum Kim. 2018. Joint learning of domain classification and out-of-domain detection with dynamic class weighting for satisficing false acceptance rates. In *Proc. Interspeech 2018*, pages 556–560.

I. Lane, T. Kawahara, T. Matsui, and S. Nakamura. 2007. Out-of-domain utterance detection using classification confidences of multiple topics. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(1):150–161.

Q. Vera Liao, Mas-ud Hussain, Praveen Chandar, Matthew Davis, Yasaman Khazaeni, Marco Patricio Crasso, Dakuo Wang, Michael Muller, N. Sadat Shami, and Werner Geyer. 2018. All work and no play? In *In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*.

Larry Manevitz and Malik Yousef. 2007. One-class document classification via neural networks. *Neurocomput.*, 70(7-9):1466–1481.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. *International Conference on Learning Representations, 2017*.

Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. 2018. Deep one-class classification. In *Proceedings of the 35th International Conference on Machine Learning*, pages 4393–4402, Stockholmsmssan, Stockholm Sweden. PMLR.

Seonghan Ryu, Seokhwan Kim, Junhwi Choi, Hwanjo Yu, and Gary Lee. 2017. Neural sentence embedding using only in-domain sentences for out-of-domain sentence detection in dialog systems. *Pattern Recognition Letters*, 88:26–32.

Seonghan Ryu, Sangjun Koo, Hwanjo Yu, and Gary Geunbae Lee. 2018. Out-of-domain detection based on generative adversarial network. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 714–718. Association for Computational Linguistics.

Bernhard Schölkopf, John C. Platt, John C. Shawe-Taylor, Alex J. Smola, and Robert C. Williamson. 2001. Estimating the support of a high-dimensional distribution. *Neural Comput.*, 13(7):1443–1471.

Ameneh Shamekhi, Q Vera Liao, Dakuo Wang, Rachel KE Bellamy, and Thomas Erickson. 2018. Face value? exploring the effects of embodiment for a group facilitation agent. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 391. ACM.

Lei Shu, Hu Xu, and Bing Liu. 2017. Doc: Deep open classification of text documents. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2911–2916, Copenhagen, Denmark. Association for Computational Linguistics.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4077–4087. Curran Associates, Inc.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. 2016. Matching networks for one shot learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 3637–3645, USA. Curran Associates Inc.

Wenhan Xiong, Jiawei Wu, Deren Lei, Mo Yu, Shiyu Chang, Xiaoxiao Guo, and William Yang Wang. 2019. Imposing label-relational inductive bias for extremely fine-grained entity typing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 773–784.

Wenhan Xiong, Mo Yu, Shiyu Chang, Xiaoxiao Guo, and William Yang Wang. 2018. One-shot relational learning for knowledge graphs. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1980–1990.

Mo Yu, Xiaoxiao Guo, Jinfeng Yi, Shiyu Chang, Saloni Potdar, Yu Cheng, Gerald Tesauro, Haoyu Wang, and Bowen Zhou. 2018. Diverse few-shot text classification with multiple metrics. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1206–1215. Association for Computational Linguistics.

# Harnessing Pre-Trained Neural Networks with Rules for Formality Style Transfer

**Yunli Wang**[†], **Yu Wu**[◇], **Lili Mou**[‡], **Zhoujun Li**[†]*, **Wenhan Chao**[†]
[†]State Key Lab of Software Development Environment, Beihang University, Beijing, China
[◇]Microsoft Research, Beijing, China
[‡]Dept. of Computing Science, University of Alberta, Edmonton, Canada
{wangyunli,lizj,chaowenhan}@buaa.edu.cn
Wu.Yu@microsoft.com doublepower.mou@gmail.com

## Abstract

Formality text style transfer plays an important role in various NLP applications, such as non-native speaker assistants and child education. Early studies normalize informal sentences with rules, before statistical and neural models become a prevailing method in the field. While a rule-based system is still a common preprocessing step for formality style transfer in the neural era, it could introduce noise if we use the rules in a naïve way such as data preprocessing. To mitigate this problem, we study how to harness rules into a state-of-the-art neural network that is typically pretrained on massive corpora. We propose three fine-tuning methods in this paper and achieve a new state-of-the-art on benchmark datasets.

## 1 Introduction

Text formality research is essential for a wide range of NLP applications, such as non-native speaker assistants and child education. Due to the progress of deep learning techniques, researchers make a step from formality understanding to formality-aware text generation. Recently, Rao and Tetreault (2018) published a dataset, the Grammarly's Yahoo Answers Formality Corpus (GYAFC), serving as a benchmark testbed for formality style transfer, which aims to generate a formal sentence given an informal one, while keeping its semantic meaning.

Since the GYAFC dataset is small, existing studies have realized the importance of rules as a preprocessing step of informal text, typically handling capitalization (e.g., "ARE YOU KIDDING ME?"), character repetition (e.g., "noooo"), slang words (e.g., "wanna"), etc. While rule-based preprocessing could largely simplify the formality style transfer task, we observe that it also introduces noise, with an example shown in Table 1. Given a sentence with all capital letters, a common rule-based method would lower all characters except the first one. Some entities, such as R & B, are changed to lower case incorrectly, especially if not recognized as a proper noun.

| **Original informal sentence:** |
|---|
| I LOVE HIP-HOP , RAP , ROCK & POP BUT MY FAV MUSIC IS R & B |
| **Output of a rule-based system:** |
| I love hip-hop , rap , rock and pop but my fav music is r and b |

Table 1: Example of informal sentence and the output of a rule-based system.

Another intuition of ours is that, due to the small size of the parallel corpus, it would be beneficial to leverage a large neural network, which is pretrained on a massive corpus and has learned general knowledge of language. Then, we could fine-tune it in the formality style transfer task.

To this end, we study in this paper how to effectively incorporate pretrained networks—particularly, the powerful GPT-2 model (Radford et al., 2019)—with simple rules for formality style transfer. We analyze three ways of harnessing rules in GPT-2: 1) We feed the concatenation of the original informal sentence and the preprocessed one to the encoder; 2) We ensemble two models at the inference stage: one takes the original informal text as input, while the other takes the rule-preprocessed text as input; and 3) We employ two encoders to encode original informal text and the rule-preprocessed text seperately, and then develop a hierarchical attention mechanism in both word- and sentence-levels to aggregate information. Our work differs from previous work, which only feeds preprocessed inputs to the encoder. Rather, we are able to preserve more information of the original sentence, and the rule-based system is harnessed in a learnable way.

* Corresponding author. Our code and output are available at: https://github.com/jimth001/formality_emnlp19.git



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3573–3578, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

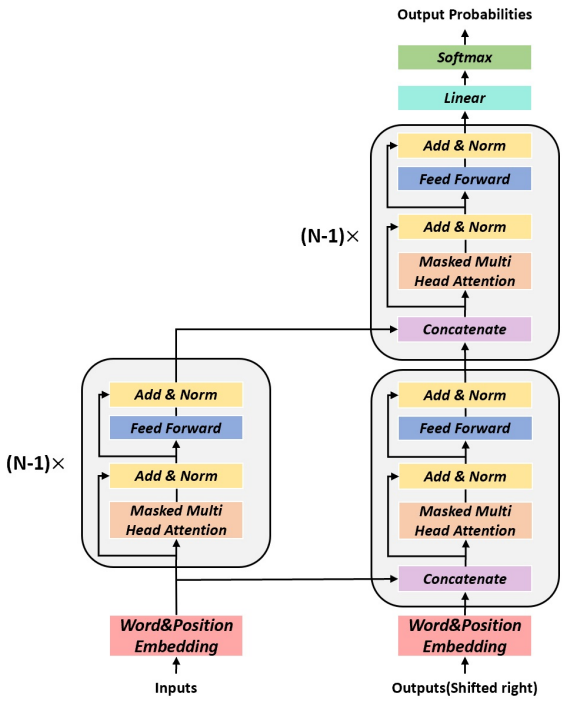


Figure 1: The architecture of our model. Both the encoder and decoder use masked multi-head attention blocks, but they do not share parameters. For the $i$th layer of the decoder (denoted as $l_i^{\text{dec}}$), it takes the concatenation of $l_{i-1}^{\text{enc}}$ and $l_{i-1}^{\text{dec}}$'s outputs as input. $N$ is the number of blocks in the decoder.

Experimental results show that our method outperforms direct fine-tuning of GPT-2 by 2 BLEU scores, and previous published results by 1.8–2.8 scores in different domains of the GYAFC dataset.

## 2 Related Work

In the past few years, style-transfer generation has attracted increasing attention in NLP research. Early work transfers between modern English and the Shakespeare style with a phrase-based machine translation system (Xu et al., 2012). Recently, style transfer is more recognized as a controllable text generation problem (Hu et al., 2017), where the style may be designated as sentiment (Fu et al., 2018), tense (Hu et al., 2017), or even general syntax (Bao et al., 2019; Chen et al., 2019). In the above approaches, the training sentences are labeled with style information, but no parallel data are given. Xu et al. (2019a) take one step further and capture the most salient style by detecting global variance in a purely unsupervised manner (i.e., style labels are unknown).

Formality style transfer is mostly driven by the GYAFC parallel corpus. Since a parallel corpus, albeit small, is available, formality style transfer usually takes a seq2seq-like approach (Rao and Tetreault, 2018; Niu et al., 2018a; Xu et al., 2019b). In particular, this paper focuses on harnessing pre-trained neural networks with rule-based systems.

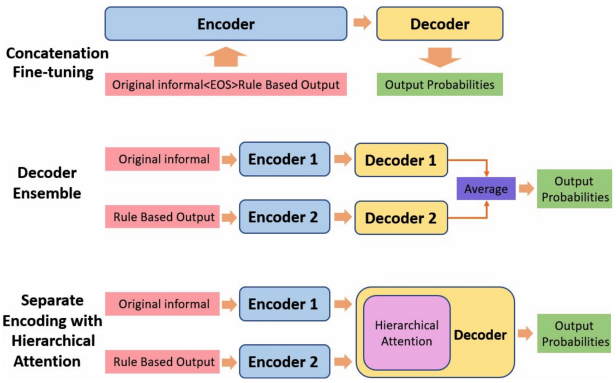


Figure 2: Three different methods to incorporate pre-trained models with rule-based systems.

## 3 Approach

We implement our encoder and decoder with GPT blocks, and initialize them with the pretrained GPT-2 parameters (Radford et al., 2019). The architecture of a decoder GPT block performs attention to the context words and previous words with the same multi-head attention layer, illustrated in Figure 1, which is slightly different with the classic Transformer (Vaswani et al., 2017). Formally, the output of the attention layer is[1]

$$\text{softmax}\left(\frac{\boldsymbol{Q}[\boldsymbol{K}_{\text{enc}};\boldsymbol{K}_{\text{dec}}]^T}{\sqrt{d_k}}\right)[\boldsymbol{V}_{\text{enc}};\boldsymbol{V}_{\text{dec}}] \quad (1)$$

where $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ are defined the same as the scaled dot-product attention in Transformer, and $d_k$ is a scaling factor. $[;]$ is a concatenation operation; it enables to consider context words and previous decoding results in the same layer. Such architecture enables to adapt GPT-2, a Transformer-based pretrained language model, to a sequence-to-sequence model without re-initializing the parameters.

In the following, we describe several methods combining the GPT-based encoder-decoder model with preprocessing rules and (limited) parallel data.

**Fine-Tuning with Prepocessed Text as Input.** Given an informal sentence $\mathbf{x}_i$ as input, the most

[1]We use bold italic letters to denote a vector or matrix.

straightforward method, perhaps, is to first convert $\mathbf{x}_i$ to $\mathbf{x}'_i$ by rules, and then fine-tune the pre-trained GPT model with parallel data $\{(\mathbf{x}'_i, \mathbf{y}_i)\}_{i=0}^{M}$ ($M$ is the number of samples). In this way, informal sentences can be normalized with rules before using a neural network. This simplifies the task and is standard in previous studies of formality style transfer (Rao and Tetreault, 2018).

However, the preprocessed sentence serves as a Markov blanket, i.e., the system is unaware of the original sentence, provided that the preprocessed one is given. This is in fact not desired, since the rule-based system could make mistakes and introduce noise (Table 1).

**Fine-Tuning with Concatenation.** To alleviate the above issue, we feed the encoder with both the original sentence $\mathbf{x}_i$ and the preprocessed one $\mathbf{x}'_i$. We concatenate the words of $\mathbf{x}_i$ and $\mathbf{x}'_i$ with a special token `EOS` in between, forming a long sequence $(x_{i,1}, \cdots, x_{i,s_i}, \texttt{EOS}, x'_{i,1}, \cdots, x'_{i,s'_i})$; after that, the concatenated sequence and the corresponding formal reference serve as a parallel text pair to fine-tune the GPT model. In this way, our model can make use of a rule-based system but also recognize its errors during the fine-tuning stage.

**Decoder Ensemble.** We investigate how the model performs if we train two GPTs with $\{\mathbf{x}'_i, \mathbf{y}_i\}_{i=0}^{M}$ and $\{\mathbf{x_i}, \mathbf{y_i}\}_{i=0}^{M}$ separately, but combine them by model ensemble in the decoding phase. We denote the generative probability of the $j$th word by $h(\mathbf{x}_i, \mathbf{y}_{i,<j})$ and $h'(\mathbf{x}'_i, \mathbf{y}_{i,<j})$. We apply "average voting" and the resulting predictive probability is $\frac{h'(\mathbf{x}'_i, \mathbf{y}_{i,<j}) + h(\mathbf{x}_i, \mathbf{y}_{i,<j})}{2}$.

**Hierarchical Attention.** In our final variant, we use two encoders to encode $\mathbf{x}_i$ and $\mathbf{x}'_i$ separately, but compute a hierarchical attention to aggregate information, given by

$$\alpha \cdot \text{softmax}\left(\frac{\boldsymbol{Q}[\boldsymbol{K}_{\text{enc1}}; \boldsymbol{K}_{\text{dec}}]^\top}{\sqrt{d_k}}\right)[\boldsymbol{V}_{\text{enc1}}; \boldsymbol{V}_{\text{dec}}] + \beta \cdot \text{softmax}\left(\frac{\boldsymbol{Q}[\boldsymbol{K}_{\text{enc2}}; \boldsymbol{K}_{\text{dec}}]^\top}{\sqrt{d_k}}\right)[\boldsymbol{V}_{\text{enc2}}; \boldsymbol{V}_{dec}] \quad (2)$$

where $\alpha$ and $\beta$ are sentence attention weights for each decoding step, computed by

$$\alpha = \frac{\exp(\boldsymbol{h}_l^T \boldsymbol{W} \boldsymbol{z}_1)}{\exp(\boldsymbol{h}_l^T \boldsymbol{W} \boldsymbol{z}_1) + \exp(\boldsymbol{h}_l^T \boldsymbol{W} \boldsymbol{z}_2)}, \quad \beta = 1 - \alpha \quad (3)$$

Here, $\boldsymbol{h}_l$ is the hidden state of the $l$th step at the decoder, $\boldsymbol{W}$ is a learnable parameter, $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ represent the last hidden state of the two encoders, respectively. We propose this variant in hopes of combining the information of $\mathbf{x}_i$ and $\mathbf{x}'_i$ in the training stage and in a learnable way, compared with the decoder ensemble.

## 4 Experiments

### 4.1 Setup

We evaluate our methods on the benchmark dataset, the Grammarly's Yahoo Answers Formality Corpus (GYAFC, Rao and Tetreault, 2018). It consists of handcrafted informal-formal sentence pairs in two domains, namely, Entertainment & Music (E&M) and Family & Relationship (F&R). Table 2 shows the statistics of the training, development, and test sets. In the development and test sets of GYAFC, each sentence has four references, against which evaluation metrics are computed.

| | Train | Dev | Test |
|---|---|---|---|
| Entertainment & Music | 52,595 | 2,877 | 1,416 |
| Family & Relationship | 51,967 | 2,788 | 1,332 |

Table 2: Corpus statistics.

We implement our model with Tensorflow 1.12.0 and take the pretrained GPT-2 model (117M) released by OpenAI[2] to initialize our encoder and decoder. We use the Adam algorithm (Kingma and Ba, 2015) to train our model with a batch size 128. We set the learning rate to 0.001 and stop training if validation loss increases in two successive epochs.

### 4.2 Competing Methods

We compare our model with the following state-of-the-art methods in previous studies.

**Rule-Based:** We follow Rao and Tetreault (2018) and create a set of rules to convert informal texts to formal ones. Due to the lack of industrial engineering, our rule-based system achieves a performance slightly lower than (but similar to) Rao and Tetreault (2018).

**NMT-Baseline:** An RNN-based Seq2Seq model with the attention mechanism (Bahdanau et al., 2015) is trained to predict formal texts, given rule-preproccessed informal text.

**PBMT-Combined:** Similar to NMT, this baseline trains a traditional phrase-based machine translation (PBMT) system, also taking the preprocessed text as input. Then, self-training (Ueff-

[2] https://github.com/openai/gpt-2

ing, 2006) is applied with an unlabeled in-domain dataset for further improvement.

**NMT-Combined:** This method uses back-translation (Sennrich et al., 2016) with the PBMT-Combined system to synthesize a pseudo-parallel corpora. Then a Seq2Seq model is trained on the combination of the pseudo-parallel and parallel corpora.

Note that the above baselines are reported by Rao and Tetreault (2018).

**Transformer-Combined:** This setting in Xu et al. (2019b) is the same as NMT-Combined, except that it employs Transformer (Vaswani et al., 2017) as the encoder and decoder.

**JTHTA:** Xu et al. (2019b) propose a bi-directional framework that can transfer formality from formal to informal or from informal to formal with one single encoder-decoder component. They jointly optimize the model against various losses and call it Joint Training with Hybrid Textual Annotation (JTHTA).

**Bi-directional-FT:** Niu et al. (2018b) merge the training data of two domains and leverage data borrowed from machine translation to train their models with a multi-task learning schema, and also apply model ensembles. For fairness, we also combine the two domains when comparing with Niu et al. (2018b).

Additionally, we also evaluate our model variants. We first apply the GPT based on the original parallel corpus without using the rule-based system, denoted as GPT-Orig. Then, we feed the rule-preprocessed text as input, denoted as GPT-Rule. Other variants in Section 3 are denoted as GPT-CAT, GPT-Ensemble, and GPT-HA, respectively.

### 4.3 Evaluation Metrics

To evaluate different models, we apply multiple automatic metrics, mostly following Rao and Tetreault (2018).

**Formality:** Rao and Tetreault (2018) train a feature-based model to evaluate the formality of sentences, requiring an extra labeled corpus for training, which is unfortunately not publicly available. As a replacement, we train an LSTM-based classifier using the training data of GYAFC. It achieves 93% accuracy in the development and test sets, and thus is an acceptable approximation.

**Meaning Preservation:** We evaluate whether the meaning of the source sentence is preserved with a model trained on the Semantic Textual Similarity (STS) dataset. We adopt the BERT-Base[3] model (Devlin et al., 2019) and use STS for fine-tuning.

**Overall:** We evaluate the overall quality of formality-transferred sentences with BLEU (Papineni et al., 2002) and PINC (Chen and Dolan, 2011). BLEU evaluates the $n$-gram overlap, and PINC is an auxiliary metric indicating the dissimilarity between an output sentence and an input. A PINC score of 0 indicates that the input and output sentences are the same. According to Rao and Tetreault (2018), BLEU correlates with human annotation best.

### 4.4 Results

We show results of the E&M and F&R domains in Tables 3 and 4, respectively. We see that, by using the GPT-2 pretrained model alone (GPT-Orig), we achieve close results to previous state-of-the-art models. It outperforms NMT-Combine and JTHTA, even without fine-tuning on pseudo-parallel data. Our method also significantly outperforms the Transformer-Combined model (without pretraining). The results suggest that the small GYAFC corpus does not suffice to fully train the Transformer model. The GPT-2 model, pretrained with massive unlabeled corpora, is able to capture the generic knowledge of language and can be adapted to formality style transfer.

We then evaluate our different methods of incorporating the rule-based system into the pretrained GPT-2 model. We see that GPT-CAT yields the best results, which is probably because the concatenation enables two input sentences interact with each other through a single self-attention mechanism, while other methods encode each input sentences (original and rule-preprocessed) separately.

When combining both domains as in Niu et al. (2018b), we also have better performance than the previous work. This further shows the robustness of our model.

Regarding formality, our model achieves a reasonably high accuracy, although combining domains is slightly worse (since cross-domain training may bring noise that hurts the output formality). The rule-based model itself shows the best performance on content preserving, but it does not

[3] https://github.com/google-research/bert

| | Formality | Meaning | BLEU | PINC |
|---|---|---|---|---|
| Original Informal | 20.05 | 4.85 | 50.30 | 0 |
| Formal Reference | 79.61 | 3.78 | 100.00 | 66.93 |
| In-domain data | | | | |
| Rule-based† | 48.69 | **4.37** | 60.35 | 28.26 |
| NMT-baseline† | **77.38** | 3.25 | 58.26 | 54.94 |
| NMT-Combined† | 73.81 | 3.88 | 67.55 | 43.45 |
| PBMT-Combined† | 66.94 | 4.00 | 66.87 | 43.27 |
| Transformer-Combined | - | - | 65.50 | - |
| JTHTA | - | - | 69.63 | - |
| GPT-Orig | 73.75 | 3.70 | 69.30 | 47.35 |
| GPT-Rule | 74.88 | 3.66 | 69.65 | 48.85 |
| GPT-Ensemble | 74.81 | 3.69 | 69.86 | 48.20 |
| GPT-CAT | 74.09 | 3.76 | **71.39** | 46.38 |
| GPT-HA | 74.37 | 3.67 | 69.03 | 47.82 |
| Combined domains | | | | |
| Bi-directional FT† | 70.61 | **3.98** | 72.01 | 41.74 |
| GPT-CAT | **71.45** | 3.81 | **72.70** | 44.07 |

Table 3: Test performance on the E&M domain. PINC reflects the dissimilarity to the original informal sentence. Neither a too high nor a too slow score is desired. † indicates that we evaluate the output (if available) given by each paper with our metrics. Othewise, we quote the BLEU score from respective papers.

| | Formality | Meaning | BLEU | PINC |
|---|---|---|---|---|
| Original Informal | 21.31 | 4.76 | 51.66 | 0 |
| Formal Reference | 81.53 | 3.20 | 100.00 | 65.59 |
| In domain data | | | | |
| Rule-based† | 57.50 | **4.24** | 66.36 | 27.75 |
| NMT-baseline† | **79.31** | 3.40 | 68.26 | 49.35 |
| NMT-Combined† | 76.75 | 3.77 | 73.78 | 41.76 |
| PBMT-Combined† | 77.45 | 3.82 | 72.40 | 44.02 |
| Transformer-Combined | - | - | 70.63 | - |
| JTHTA | - | - | 74.43 | - |
| GPT-Orig | 77.90 | 3.80 | 75.65 | 42.20 |
| GPT-Rule | 78.62 | 3.76 | 76.08 | 42.87 |
| GPT-Ensemble | 78.67 | 3.77 | 76.32 | 42.61 |
| GPT-CAT | 78.80 | 3.78 | **77.26** | 42.77 |
| GPT-HA | 77.31 | 3.79 | 76.31 | 41.86 |
| Combined domains | | | | |
| Bi-directional FT† | 74.54 | **3.97** | 75.33 | 39.39 |
| GPT-CAT | **76.84** | 3.81 | **76.87** | 42.44 |

Table 4: Test performance on the F&R domain.

change the input much (a low PINC score).

In summary, our models significantly outperform previous work in formality style transfer and achieve a state-of-the-art performance on the two domains of GYAFC, which credits to both the pretrained model and our fine-tuning methods in consideration of a rule-based system.

## 5 Conclusion

In this work, we study how to incorporate a pretrained neural network with a rule-based system for formality style transfer. We find that building a pretrained GPT-2 upon the concatenation of the original informal text and the rule-preprocessed text achieves the highest performance on benchmark datasets.


## Acknowledgments

The authors thank the anonymous reviewers for insightful comments. This work is supported in part by the National Natural Science Foundation of China (Grand Nos. U1636211, 61672081, and 61370126). Lili Mou is an Amii Fellow; he also thanks AltaML for support.


## References


Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*.

Yu Bao, Hao Zhou, Shujian Huang, Lei Li, Lili Mou, Olga Vechtomova, Xin-yu Dai, and Jiajun Chen. 2019. Generating sentences from disentangled syntactic and semantic spaces. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6008–6019.

David Chen and William B. Dolan. 2011. Collecting highly parallel data for paraphrase evaluation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 190–200.

Mingda Chen, Qingming Tang, Sam Wiseman, and Kevin Gimpel. 2019. Controllable paraphrase generation with a syntactic exemplar. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5972–5984.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2018. Style transfer in text: Exploration and evaluation. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 663–670.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1587–1596.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*.

Xing Niu, Sudha Rao, and Marine Carpuat. 2018a. Multi-task neural models for translating between

styles within and across languages. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1008–1021.

Xing Niu, Sudha Rao, and Marine Carpuat. 2018b. Multi-task neural models for translating between styles within and across languages. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1008–1021.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*.

Sudha Rao and Joel R. Tetreault. 2018. Dear Sir or Madam, may I introduce the GYAFC dataset: Corpus, benchmarks and metrics for formality style transfer. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 129–140.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Edinburgh neural machine translation systems for WMT 16. In *Proceedings of the 1st Conference on Machine Translation (WMT)*, pages 371–376.

Nicola Ueffing. 2006. Self-training for machine translation. In *NIPS Workshop on Machine Learning for Multilingual Information Access*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Peng Xu, Yanshuai Cao, and Jackie Chi Kit Cheung. 2019a. Unsupervised controllable text generation with global variation discovery and disentanglement. *arXiv preprint arXiv:1905.11975*.

Ruochen Xu, Tao Ge, and Furu Wei. 2019b. Formality style transfer with hybrid textual annotations. *arXiv preprint arXiv:1903.06353*.

Wei Xu, Alan Ritter, Bill Dolan, Ralph Grishman, and Colin Cherry. 2012. Paraphrasing for style. In *Proceedings of the 24th International Conference on Computational Linguistics*, pages 2899–2914.

# Multiple Text Style Transfer by using Word-level Conditional Generative Adversarial Network with Two-Phase Training

**Chih-Te Lai**[1], **Yi-Te Hong**[1,2], **Hong-You Chen**[1], **Chi-Jen Lu**[1], **Shou-De Lin**[2]
[1]Academina Sinica, Taipei, Taiwan
[2]National Taiwan University, Taipei, Taiwan
[1]{cloudylai, ted0504, cbbjames, cjlu}@iis.sinica.edu.tw
[2]sdlin@ntu.edu.tw

## Abstract

The objective of non-parallel text style transfer is to alter specific attributes (e.g. sentiment, mood, tense, politeness, etc) of a given text while preserving unrelated content. Adversarial training is a popular method to ensure the transferred sentences have the desired target styles. However, previous works often suffer from content leaking problem. In this paper, we propose a new adversarial training model with a word-level conditional architecture and a two-phase training procedure. By using a style-related condition architecture before generating a word, our model is able to maintain style-unrelated words while changing the others. By separating the training procedure into reconstruction and transfer phases, our model is able to balance the reconstruction and adversarial losses. We test our model on polarity sentiment transfer and multiple-attribute transfer tasks. The empirical results show that our model achieves comparable evaluation scores in both transfer accuracy and fluency but significantly outperforms other state-of-the-art models in content compatibility on three real-world datasets.

## 1 Introduction

Text style transfer is a challenging problem in natural language generation, whose objective is to alter specific attributes (e.g. sentiment, mood, tense, voice, politeness, etc (Hu et al., 2017; Shen et al., 2017; Sennrich et al., 2016; Logeswaran et al., 2018; Prabhumoye et al., 2018)) of a given text while preserving its remaining attributes and contents. This task has potential applications such as paraphrasing, summarizing articles, author obfuscation (Reddy and Knight, 2016), poems/lyrics rewriting, and scenario-adaptive machine translation (Michel and Neubig, 2018).

One major challenge for text style transfer is that parallel data across different text attributes is difficult to collect and label. Without parallel data, supervised deep-learning models are not applicable and the transfer rules among styles unclear. Unlike image style transfer, another main challenge for text style transfer is the difficulty to identify and disentangle neural feature representations for texts.

In previous models (Shen et al., 2017; Hu et al., 2017) the core idea for non-parallel text style transfer, is to training an auto-encoder with additional adversarial loss (Goodfellow et al., 2014), (or a VAE with a classifier), for the discriminator, (or classifier), to guide the decoder generate sentences to have a specific target style.

Despite previous success in attribute-conditioned text generation, several research questions remain, regarding to previous models, including: 1) previous modeling limitations to transformations between only a few attributes. 2) the issue of trade-off among text fluency, content preservation, and the accurate transfer with the desired attributes and 3) the unstability of adversarial training.

To address these issues, we make contribution by adopting an encoder-decoder framework and propose a novel conditional adversarial training, including several improvements as follows: (1) a word-level attribute condition architecture in both the decoder and discriminators to capture relations between styles and words; (2) we employ a seq-to-seq attention mechanism and (3) a two-phase training procedure (reconstruction/transfer phases) for better content preservation. It is trained with standard adversarial learning approach. We test our conditional adversarial training on two tasks: (a) polarity style transfer, and (b) multiple-attribute style transfer.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3579–3584,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

## 2 Related Works

Text style transfer without parallel data is an active research topic. Mueller et al. (2017) designed a variational auto-encoder (VAE) framework; Hu et al. (2017) used VAE with controllable attributes; Shen et al. (2017) proposed to adversarially train a Cross-Aligned Auto-Encoder (CAAE) to align two different styles. To improve performances, several works including, (Fu et al., 2017; Yang et al., 2018; dos Santos et al., 2018; Logeswaran et al., 2018) were proposed. Fu et al. (2017) suggested a multi-head decoder to generate sentences with different styles; Yang et al. (2018) utilized language models as discriminators to stabilize training; dos Santos et al. (2018) used a classifier to aid style transfer; Logeswaran et al. (2018) also made use of a conditional discriminator for multiple style transfer.

On the other hand, a few works including, Li et al. (2018), Xu et al. (2018) adopt an *erase-and-replace* approach and design their methods to erase the style-related words first and then fill in words of different style attributes. Non-parallel text style transfer is also relevant to unsupervised machine translation. Prabhumoye et al. (2018), Subramanian et al. (2018), Logeswaran et al. (2018) and dos Santos et al. (2018) apply *back-translation* technique from unsupervised machine translation for style transfer task. Our work follows the framework of CAAE, and we propose several adjustments to improve the performance.

## 3 Methodology

### 3.1 Model Architecture

As shown in Figure 1, our model contains an encoder-decoder $(E, G)$ and two discriminators $D_{cnn}, D_{rnn}$. We describe each model architecture in details.

#### 3.1.1 Encoder-Decoder

Following prior works (Hu et al., 2017; Shen et al., 2017; Yang et al., 2018; Logeswaran et al., 2018), we use a seq-to-seq de-noising encoder-decoder $(E, G)$ with attention mechanism (Bahdanau et al., 2014). For each input sentence $x$ and attribute $\tilde{y}$, the encoder $E$ encodes $x$ to a latent code $z = E(x)$, and the decoder $G$ decodes transferred sentence $\tilde{x} = G(z, \tilde{y})$, which can be further back-translated to reconstruct the original sentence. Based on this framework, we design *word-level condition* model for better results.

**Word-level Condition** The most unique component between our encoder-decoder and previous works is the condition architecture of attributes. Most of style transfer works (Hu et al., 2017; Shen et al., 2017; Logeswaran et al., 2018) treat attributes $y$ as part of the initial vector fed into the RNN cell in the decoder, and we argue that the conditioning structure is important to the model performance. In our decoder $G$, we embed $y$ to a vector and concatenate the vector with the output $h_t$ of GRU cell at each time step $t$. More formally, at each time step $t$, the hidden state $h_t$ and output probabilities $o_t$ are generated as follows:

$$h_t = GRU(h_{t-1}, x_t, c_t|z)$$
$$o_t = \sigma(W_p([y, h_t]) + b_p)$$

where $GRU$ denotes a Gated Recurrent Unit (Chung et al., 2015) in decoder $G$, and $c_t$ is the content vector from the attention mechanism; $W_p, b_p$ denote the projection matrix and bias to map a hidden state to an output vocabulary distribution; $\sigma$ is the softmax function.

#### 3.1.2 Discriminator

Our discriminators take output probability distributions $o$ as inputs along with attribute labels $y$. Formally, for each discriminator $D \in \{D_{cnn}, D_{rnn}\}$, the function of $D$ can be expressed as:

$$D(o, y) = f_{disc}(f_{trans}(f_{cond}(o, y))).$$

$f_{cond}$ is a multi-layer perceptron. $f_{trans}$ is either a bi-directional GRU or a CNN to perform a global feature transform in $D_{cnn}, D_{rnn}$ respectively. Finally, $f_{disc}$ is a fully-connected layer with the sigmoid function to output decisions. For simplicity, we substitute $x$ for $o$ as the input to discriminators in the following description.

### 3.2 Loss functions

We train our model with reconstruction loss $L_{rec}$, back-translation loss $L_{bt}$, discrimination loss $L_{disc}$, and adversarial loss $L_{adv}$. $L_{rec}$ and $L_{bt}$ force the decoder $G$ to reconstruct the input sentence $x$. $L_{rec}$, $L_{bt}$ are expressed as:
$L_{rec} = -\log p(x|z, y)$, $L_{bt} = -\log p(x|\tilde{z}, y)$.
$L_{disc}$ and $L_{adv}$ force the decoder $G$ to output the transferred sentence $\tilde{x}$ with correct attributes $\tilde{y}$. For $D \in \{D_{cnn}, D_{rnn}\}$, $L_{disc}$, $L_{adv}$ are listed as:
$L_{disc} = -\mathbb{E}_{x,y,\tilde{x},\tilde{y}}[\log D(x, y) + \log(1 - D(\tilde{x}, \tilde{y}))]$
$L_{adv} = -\sum_{D \in \{D_{cnn}, D_{rnn}\}} \{\mathbb{E}_{\tilde{x},\tilde{y}}[\log D(\tilde{x}, \tilde{y})]\}$.

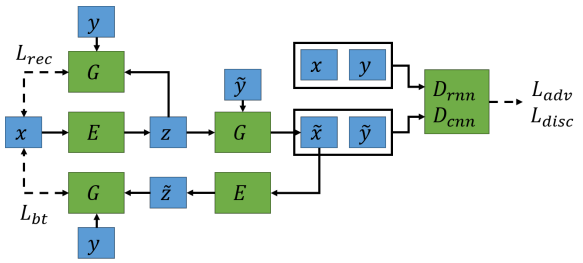


Figure 1: Overview of our model framework. Our model contains an encoder-decoder $(E, G)$, and two discriminators $D_{rnn}$ and $D_{cnn}$. We train our model with reconstruction loss $L_{rec}$, back-translation loss $L_{bt}$, adversarial loss $L_{adv}$, and discrimination loss $L_{disc}$. The dashed arrows describe where the losses are computed.

The total loss $L_{tot}$ for our encoder-decoder $(E, G)$ is a weighted sum of $L_{rec}$, $L_{bt}$, and $L_{adv}$.

### 3.3 Training

We split our training procedure into two phases: *reconstruction* and *transfer* phase. We first train our encoder-decoder $(E, G)$ with loss $L_{rec}$ in reconstruction phase, and then in transfer phase, we use total loss $L_{tot}$ and $L_{disc}$ to train our encoder-decoder $(E, G)$ and the discriminators $D_{cnn}, D_{rnn}$, respectively. Our experiments show this approach improves content preservation significantly. A diagram about this two-phase training is provided in Appendix A.1.

## 4 Experiments Setup

### 4.1 Datasets

**YelpSent Dataset** The preprocessed Yelp dataset (Shen et al., 2017) consists of sentences with length limit of 15 words, labeled with either positive or negative sentiment as attributes.

**AmaProd and AmaSent Dataset** Amazon product dataset (He and McAuley, 2016): consists of product reviews associated with ratings of the products. We select 4 product types: (books/ movies/ electronics/ CDs), following the approach in (Kim et al., 2017) to select relevant sentences. [1]

**YelpTense Dataset** We use similar approach to label sentences with sentiment and past/present tense from Yelp Dataset Challenge[2]. If a sentence contains at least one verb in past tense, we label it as 'past tense'; otherwise as 'present tense'.

All the data statistics are presented in Table 5 in Appendix A.2.

### 4.2 Evaluation Metrics

We follow previous works and apply three automatic evaluation metrics and one human evaluation for three indicative aspects: *attribute compatibility*, *content preservation*, and *fluency*.

**Attribute Compatibility** To measure how well our model transfers sentence attributes, we pre-train a CNN classifier (Kim, 2014) for each attribute category (i.e. product types, sentiments, tenses) on our training data, and use the classifiers to measure the accuracy of transferred sentences associated with the desired attribute. We report the accuracy of each attribute category separately.

**Content Preservation** Measuring content preservation is still an open research problem. Following the previous works, we compute the BLEU score (Papineni et al., 2002) used in machine translation. In our experiments, we use self BLEU to evaluate on transferred sentences as a measurement of content preservation.

**Fluency** To test fluency of the generated sentences, we train a bi-directional LSTM language model on our training data for each dataset. We regard the perplexity of generated sentences as a measure for fluency.

**Human Evaluation** We also evaluate the transferred sentences with human assessments. In the evaluation, 8 people are asked to rate sentences based on criteria associated with three dif: attribute compatibility, content preservation, and fluency. Each aspect is rated on a 5-point Likert scale. 20 sentences on *YelpSent* with corresponding transferred sentences are randomly selected as testing examples. More details about our human evaluation are provided in Appendix A.3.

### 4.3 Comparison with State-Of-The-Arts

We compare with four different State-of-the-Art models: **CAAE**, **DAR**, **MultiAttr**, **ContPrev**. **CAAE** (Shen et al., 2017) consists of an auto-encoder with discriminator networks to guide text generation. **DAR** (Li et al., 2018) uses a delete-and-retrieve approach. [3] **MultiAttr** (Subramanian et al., 2018) performs multiple-attributes style transfer using back-translation (Lample et al.,

[1] We refer to dataset labeled with both sentiment and product attributes as *AmaProd Dataset*, and dataset labeled only with sentiment attribute as *AmaSent Dataset*, respectively.

[2] https://ww.yelp.com/dataset/challenge

[3] We train DAR with the source code from https://github.com/rpryzant/delete_retrieve_generate

2018, 2017; Artetxe et al., 2017) and latent representation pooling. [4] **ContPrev** (Logeswaran et al., 2018) is an auto-encoder model with a conditional discriminator for multiple attributes transfer. Among these models, in our experiments, CAAE and DAR are compared only on polarity sentiment style transfer tasks. More details about our model and training settings are provided in Appendix A.4.

## 5 Experimental Results

### 5.1 Quantitative Result

**Automatic Evaluation Results** Table 1 and Table 2 show the automatic evaluation results. Our model achieves higher BLEU scores and comparable transfer accuracy. We also notice that our model generates sentences with higher perplexity, while other models produce sentences with perplexity lower than the real data.

**Human Evaluation Results** Table 3 exhibits the human evaluation results on *YelpSent*. The results show that our sentences are evaluated higher on content preservation, and share comparable scores with other models on attribute compatibility.

**Ablation test** We conduct ablation experiments on our model on *YelpSent* dataset. As the results shown at Table 4, removing the word-level condition architecture decreases transfer accuracy and BLEU scores. The two-phase training procedure can also ensure a much higher BLEU scores. Using both CNN and RNN discriminator sightly improves the performance on all metrics.

**Evaluation Curve** We also plot the curves of transfer accuracy, BLEU score and perplexity, respectively, on the validation set, as the number of training epochs increases, across different models in Appendix A.5. According to the curves, our model achieves higher BLEU scores and relatively stable performance on all three metrics.

### 5.2 Qualitative Results

Our model exhibits a tendency to follow the original sentence surface structure. With the help of word-level conditional architecture, the decoder learns to make word adjustments. Sample sentences are shown in Tables 7 and 8 in Appendix A.6. For frequent occurring sentence structures across different attribute domains, our

[4] We re-implement this model with comparable results to the original publication.

Table 1: Polarity sentiment transfer on *YelpSent* & *AmaSent*

| *YelpSent* | Sent. | BLEU | PPL |
|---|---|---|---|
| Real data | 96.9% | – | 20.3 |
| Our model | 87.8% | **35.5** | 34.5 |
| CAAE | 83.5% | 11.5 | 19.0 |
| DAR | **96.3%** | 0.1 | 20.6 |
| MultiAttr | 86.0% | 12.9 | **8.4** |
| ContPrev | 91.3% | 14.7 | 10.5 |
| *AmaSent* | Sent. | BLEU | PPL |
| Real data | 94.4% | – | 34.1 |
| Our model | 77.5% | **24.5** | 57.4 |
| CAAE | 75.4% | 5.3 | 23.7 |
| DAR | **83.4%** | 0.1 | 37.6 |
| MultiAttr | 76.6% | 12.7 | **6.0** |
| ContPrev | 72.1% | 12.7 | 17.0 |

**Sent.** represents the transfer accuracy measured by the pretrained sentiment-attribute classifier. **BLEU** and **PPL** stand for self BLEU and perplexity score, respectively.

Table 2: Multiple attribute transfer on *YelpTense* & *AmaProd*

| *YelpTense* | Sent. | Tense | BLEU | PPL |
|---|---|---|---|---|
| Real data | 95.4% | 99.9% | – | 24.5 |
| Our model | **79.9%** | **96.1%** | **32.2** | 40.0 |
| MultiAttr | 74.5% | 91.4% | 25.9 | **8.2** |
| ContPrev | 76.6% | 94.9% | 14.7 | 13.4 |
| *AmaProd* | Sent. | Prod. | BLEU | PPL |
| Real data | 94.4% | 95.6% | – | 34.1 |
| Our model | 76.0% | 87.4% | **22.4** | 26.8 |
| MultiAttr | **79.9%** | **90.4%** | 15.3 | **7.7** |
| ContPrev | 75.8% | 81.4% | 11.2 | 17.7 |

**Sent.**, **Tense** and **Prod.** represent the transfer accuracy measured by the pretrained sentiment-, tense- and product-attribute classifiers, respectively.

Table 3: Human evaluation on *YelpSent*

| *YelpSent* | Cont. | Flu. | Sent. |
|---|---|---|---|
| Our model | **4.01** | 3.54 | 3.17 |
| CAAE | 3.14 | 2.99 | 3.19 |
| DAR | 1.17 | **4.24** | 3.35 |
| MultiAttr | 3.26 | 3.91 | **3.59** |
| ContPrev | 3.02 | 3.76 | 3.56 |

Columns **Cont.**, **Flu.**, **Sent.** stand for human evaluation scores for *content preservation*, *fluency* and *sentiment compatibility*, each measured in a Likert scale from 1 through 5. (1: lowest and 5: highest)

Table 4: Model ablation test on *YelpSent*

| Model | Sent. | BLEU | PPL |
|---|---|---|---|
| Real data | 96.9% | – | 20.3 |
| Our model | 87.8% | 35.5 | 34.5 |
| w/o two-phase | 86.4% | 26.2 | 44.5 |
| w/o word-level | 91.5% | 3.5 | 34.2 |
| w/o CNN | 87.3% | 28.5 | 39.3 |
| w/o RNN | 86.1% | 31.7 | 42.7 |
| w/o CNN, RNN | 42.4% | 72.5 | 20.7 |

model shows good results. When the original sentence sometimes has a distinct sentence structure and would require multiple editing in a larger scale, our model may not find a correct similar sentence-level surface form adjustment in the target-attribute domain and the word-level local adjustments may sometimes not be coherent.

## 6 Conclusion

In this paper, we conduct non-parallel style transfer among multiple attributes. We propose a seq-to-seq model with word-level condition and two-phase training. The empirical results demonstrate that our model outperforms our competitors in the polarity sentiment transfer task on *YelpSent*. In multiple attribute transfer tasks, our model also achieves comparable results with the state-of-the-art MultiAttr on *YelpTense* and *AmaProd*. We also analyze our model with ablation tests.

Although our model achieves better content preservation, the general quality of our transferred sentences can be further improved. Also, designing proper evaluation metrics is still an open problem for text style transfer. We leave these two questions as the future works.

## References


Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2017. Unsupervised neural machine translation. *arXiv preprint arXiv:1710.11041*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2015. Gated feedback recurrent neural networks. In *International Conference on Machine Learning*, pages 2067–2075.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2017. Style transfer in text: Exploration and evaluation. *arXiv preprint arXiv:1711.06861*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, pages 507–517. International World Wide Web Conferences Steering Committee.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. *arXiv preprint arXiv:1703.00955*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Yoon Kim, Kelly Zhang, Alexander M Rush, Yann LeCun, et al. 2017. Adversarially regularized autoencoders. *arXiv preprint arXiv:1706.04223*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2017. Unsupervised machine translation using monolingual corpora only. *arXiv preprint arXiv:1711.00043*.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Phrase-based & neural unsupervised machine translation. *arXiv preprint arXiv:1804.07755*.

Juncen Li, Robin Jia, He He, and Percy Liang. 2018. Delete, retrieve, generate: A simple approach to sentiment and style transfer. *arXiv preprint arXiv:1804.06437*.

Lajanugen Logeswaran, Honglak Lee, and Samy Bengio. 2018. Content preserving text generation with attribute controls. In *Advances in Neural Information Processing Systems*, pages 5108–5118.

Paul Michel and Graham Neubig. 2018. Extreme adaptation for personalized neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 312–318.

Jonas Mueller, David Gifford, and Tommi Jaakkola. 2017. Sequence to better sequence: continuous revision of combinatorial structures. In *International Conference on Machine Learning*, pages 2536–2544.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W Black. 2018. Style transfer through back-translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 866–876.

Sravana Reddy and Kevin Knight. 2016. Obfuscating gender in social media writing. In *Proceedings of the First Workshop on NLP and Computational Social Science*, pages 17–26.

Cicero Nogueira dos Santos, Igor Melnyk, and Inkit Padhi. 2018. Fighting offensive language on social media with unsupervised text style transfer. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 189–194.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Controlling politeness in neural machine translation via side constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 35–40.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in Neural Information Processing Systems*, pages 6830–6841.

Sandeep Subramanian, Guillaume Lample, Eric Michael Smith, Ludovic Denoyer, Marc'Aurelio Ranzato, and Y-Lan Boureau. 2018. Multiple-attribute text style transfer. *arXiv preprint arXiv:1811.00552*.

Jingjing Xu, Xu Sun, Qi Zeng, Xuancheng Ren, Xiaodong Zhang, Houfeng Wang, and Wenjie Li. 2018. Unpaired sentiment-to-sentiment translation: A cycled reinforcement learning approach. *arXiv preprint arXiv:1805.05181*.

Zichao Yang, Zhiting Hu, Chris Dyer, Eric P Xing, and Taylor Berg-Kirkpatrick. 2018. Unsupervised text style transfer using language models as discriminators. *arXiv preprint arXiv:1805.11749*.

# Improved Differentiable Architecture Search for Language Modeling and Named Entity Recognition

**Yufan Jiang**[1], **Chi Hu**[1], **Tong Xiao**[12]*, **Chunliang Zhang**[12], **Jingbo Zhu**[12]
[1]NLP Lab, Northeastern University, Shenyang, China
[2]NiuTrans Research, Shenyang, China
jiangyufan2018@outlook.com, huchinlp@gmail.com
{xiaotong, zhangchunliang, zhujingbo}@mail.neu.edu.cn

## Abstract

In this paper, we study differentiable neural architecture search (NAS) methods for natural language processing. In particular, we improve differentiable architecture search by removing the softmax-local constraint. Also, we apply differentiable NAS to named entity recognition (NER). It is the first time that differentiable NAS methods are adopted in NLP tasks other than language modeling. On both the PTB language modeling and CoNLL-2003 English NER data, our method outperforms strong baselines. It achieves a new state-of-the-art on the NER task.

## 1 Introduction

Neural architecture search (NAS) has become popular recently in machine learning for their ability to find new models and to free researchers from the hard work of designing network architectures. The earliest of these approaches use reinforcement learning (RL) to learn promising architectures in a discrete space (Zoph and Le, 2016), whereas others have successfully modeled the problem in a continuous manner (Liu et al., 2019; Xie et al., 2019b; Huang and Xiang, 2019). As an instance of the latter, differentiable architecture search (DARTS) employs continuous relaxation to architecture representation and makes gradient descent straightforwardly applicable to search. This leads to an efficient search process that is orders of magnitude faster than the RL-based counterparts.

Like recent methods in NAS (Xie and Yuille, 2017; Zoph and Le, 2016; Baker et al., 2016), DARTS represents networks as a directed acyclic graph for a given computation cell (see Figure 1(a)). An edge between nodes performs a predefined operation to transform the input (i.e., tail) to the output (i.e., head). For a continuous network space, DARTS uses the softmax trick to relax the categorical choice of edges to soft decisions. Then, one can optimize over the graph using standard gradient descent methods. The optimized network is inferred by choosing the edges with maximum weights in softmax.

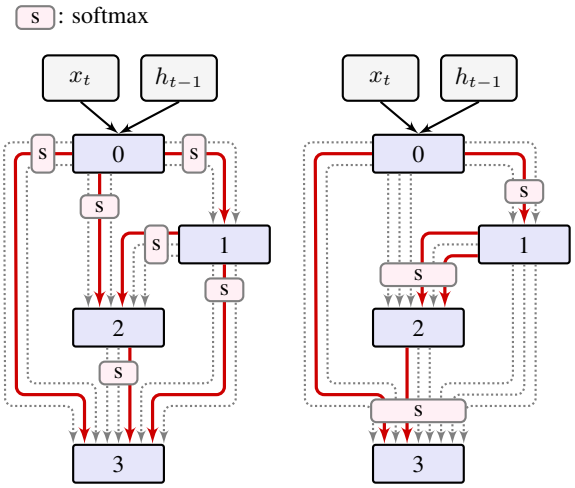


(a) DARTS cell (b) Our cell

Figure 1: An overview of DARTS cell and our cell

However, DARTS is a "local" model because the softmax-based relaxation is imposed on each bundle of edges between two nodes. This leads to a biased model in that edges coming from different nodes are not comparable. Such a constraint limits the inference space to sub-graphs with one edge between each pair of nodes. Also, the learned network might be redundant because every node has to receive edges from all predecessors no matter they are necessary or not. This problem is similar to the bias problem in other graph-based models where local decisions make the model non-optimal (Lafferty et al., 2001; Daphne Koller and Nir Friedman, 2009).

Here we present an improvement of DARTS,

*Corresponding author.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3585–3590, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

called I-DARTS, that further relaxes the softmax-local constraint. The idea is simple - we consider all incoming edges to a given node in a single softmax. This offers a broader choice of edges and enlarges the space we infer the network from. For example, one can simultaneously select multiple important edges between two nodes and leave some node pairs unlinked (see Figure 1(b)).

I-DARTS outperforms strong baselines on the PTB language modeling and CoNLL named entity recognition (NER) tasks. This gives a new state-of-the-art on the NER dataset. To our knowledge, it is the first time to apply differentiable architecture search methods to NLP tasks other than language modeling. More interestingly, we observe that our method is 1.4X faster than DARTS for convergence of architecture search. Also, we provide the architectures learned by I-DARTS, which can be referred for related tasks.

## 2 The Method

Although we will restrict ourselves to language modeling and NER for experiments, in the section, we discuss the more general case. We choose recurrent neural networks (RNNs) to model the sequence generation and tagging problems. Given a sequence of input vectors $\{x_1, ..., x_L\}$, we repeat applying RNN cells to generate the output $\{h_1, ..., h_L\}$. The RNN cell is defined as: $h_t = g(x_t, h_{t-1})$, where $t$ is the time step and $g(\cdot, \cdot)$ is the function of the cell. In NAS, the objective is to search for a good $g(\cdot, \cdot)$ in an automatic fashion.

### 2.1 Architecture Search Space

We follow the assumption that $g(\cdot, \cdot)$ is a DAG consisting of $N$ nodes and edges among them (Liu et al., 2019; Xie et al., 2019b; Pham et al., 2018). An edge $o^{i,j}(\cdot)$ between node pair $(i, j)$ indicates an activation function from node $j$ to node $i$. For node $i$, it simply sums over vectors from all predecessor nodes ($j < i$), followed by a linear transformation with a parameter matrix $W_i$. More formally, let $s_i$ be the state of node $i$. We define $s_i$ to be:

$$s_i = \sum_{j<i} o^{i,j}(s_j \cdot W_j) \tag{1}$$

See Figure 1 for an example network (red lines). Note that this model can encode an exponential number of graphs by choosing different sub-sets of edges (i.e., choosing $o^{i,j}(\cdot)$ for each $(i,j)$). The output of search is the optimal edge selection and the corresponding network.

### 2.2 Improved DARTS

Given a set of edges $\{o_k^{i,j}\}$, one can try each $o_k^{i,j}$ to induce a network, and then train and evaluate it. The optimal choice is the edge with highest accuracy on the validation set. In I-DARTS, we instead do this in a soft way. We re-define $s_i$ as:

$$s_i = \sum_{j<i} \sum_k \alpha_k^{i,j} \cdot o_k^{i,j}(s_j \cdot W_j) \tag{2}$$

where $\alpha_k^{i,j}$ is the weight indicating the importance of $o_k^{i,j}(\cdot)$. It is computed by the softmax normalization over edges between nodes $i$ and $j$, like this

$$\alpha_k^{i,j} = \frac{\exp(w_k^{i,j})}{\sum_{k'} \exp(w_{k'}^{i,j})} \tag{3}$$

where $w_k^{i,j}$ is the model parameter. This model reduces the architecture search problem to learn continuous variables $\{\alpha_k^{i,j}\}$, which can be implemented using efficient gradient descent methods. After training, the final architecture is encoded by the edges with largest weights.

Eq. (3) imposes a constraint that weights $\{\alpha_k^{i,j}\}$ are normalized for each $j$. Such a model in general faces the local decision and bias problems as pointed out in graph-based methods (Lafferty et al., 2001; Daphne Koller and Nir Friedman, 2009). Moreover, the inference has to be performed in a smaller space because we have to infer exactly one edge between each node pair and exclude networks violating this constraint.

Here we remove the constraint and system bias. To this end, we compute the softmax normalization over all incoming edges for node $i$:

$$\alpha_k^{i,j} = \frac{\exp(w_k^{i,j})}{\sum_{j<i} \sum_{k'} \exp(w_{k'}^{i,j})} \tag{4}$$

It provides us a way to compare all incoming edges in the same manner, rather than making a local decision via a bundle of edges from node $j$. As another bonus, this method can search for networks that are not covered by DARTS, e.g., networks that contain two edges between the same node pair.

See Figure 1(b) for an illustration of our method. To infer the optimal architecture, we basically do the same thing as in DARTS. The differ-

ence lies in that we select top-$n$ edges with respect to $\alpha_k^{i,j}$. Here $n$ is a hyper-parameter that controls the density of the network. E.g., $n = 1$ means a sparse net, and $n = \infty$ means a very dense net involving all those edges.

## 3 Experiments

We test our method on language modeling and named entity recognition tasks. Our experiments consist of two parts: recurrent neural architecture search and architecture evaluation. In architecture search, we search for good RNN cell architectures. Then, we train and evaluate the learned architecture.

### 3.1 Architecture Search

For language modeling, we run neural search on the PTB corpus. We use the standard preprocessed version of the dataset (Pham et al., 2018). To make it comparable with previous work, we copy the setup used in (Pham et al., 2018; Liu et al., 2019). The recurrent cell consist of 8 nodes. The candidate operation set of every edge contain 5 activation functions, including zeroize, tanh, relu, sigmoid, and identity. To learn architectures, we run the search system for 40 training epochs with a batch size of 256. We optimize models parameters $\{W_i\}$ using SGD with a learning rate of 20 and a weight decay rate of 5e-7, and optimized softmax relaxation parameters $\{w_k^{i,j}\}$ by Adam with a learning rate of 3e-3 and a weight decay rate of 1e-3. For RNN models, we use a single-layer recurrent network with embedding and hidden sizes = 300. It takes us 4 hours to learn the architecture on a single GPU of NVIDIA 1080Ti.

For named entity recognition, we choose the CONLL-2003 English dataset. We follow the same setup as in language modeling but with a different learning rate (0.1) and a different hidden layer size (256). It takes us 4 hours to learn the architecture on the same GPU.

### 3.2 Architecture Evaluation

Firstly, the discovered architecture is evaluated on the language modeling task. Before that, we train it on the same data used in architecture search. The size of hidden layers is set to 850. We use averaged SGD to train the model for 3,000 epochs, with a learning rate of 20 and a weight decay rate of 8e-7. For a fair comparison, we do not fine-tune the model at the end of the training.

| Architecture | Perplexity | | Search Cost |
|---|---|---|---|
| | val | test | (GPU days) |
| V-RHN | 67.9 | 65.4 | - |
| LSTM | 60.7 | 58.8 | - |
| LSTM + SC | 60.9 | 58.3 | - |
| LSTM + SE | 58.1 | 56.0 | - |
| ENAS | 60.8 | 58.6 | 0.50 |
| DARTS | 58.3 | 56.1 | 0.25 |
| Random RNNs | 63.7 | 61.2 | - |
| I-DARTS ($n = 1$) | 58.0 | 56.0 | 0.17 |
| I-DARTS ($n = 2$) | - | - | - |

Table 1: Perplexities on PTB (lower is better). V-RHN (Zilly et al., 2016) indicates Variational RHN. LSTM + SC (Yang et al., 2018) indicates LSTM with skip connection. LSTM + SE (Merity et al., 2018) indicates LSTM with mixture of softmax. Random RNNs indicates that the network generated by random initialized.

Table 1 shows the perplexities of different RNN models on PTB. We also report the results of previous systems. The model discovered by I-DARTS achieves a validation perplexity of 58.0 and a test perplexity of 56.0 when $n = 1$. It is on par with the state-of-the-art models that are designed either manually or automatically. However, we find that the model failed to optimize when $n = 2$. It might result from the complex interaction between operations. We leave this issue for future study.

Since architecture search is initialization-sensitive (Pham et al., 2018; Liu et al., 2019), we search the architectures for 4 times with different random seeds. We evaluate the architecture every 10 search epochs by retraining it on PTB for 500 epochs. We compare DARTS with our I-DARTS method with the same random seed. See Figure 2(b) for averaged validation perplexities over 4 different runs at different search epochs. We see that I-DARTS is easier to converge than DARTS (4 hours). It is 1.4X faster than that of DARTS. Another interesting finding is that I-DARTS achieves a lower validation perplexity than DARTS during architecture search. This may indicate better architectures found by I-DARTS because the search model is optimized with respect to validation perplexity.

Then, we test the learned architecture in a named entity recognition system on the English data from CoNLL-2003 shared task (Sang and Meulder, 2003). Following previous work (Akbik et al., 2018; Peters et al., 2017), we report the averaged F1 score over 5 runs on the test set. For modeling, we choose the single-layer RNN-CRF

model because it achieved state-of-the-art results on several sequence labeling tasks (Lample et al., 2016; Ma and Hovy, 2016). We use GloVe 100-dimensional word embeddings (Pennington et al., 2014) and pooled contextual embeddings (Akbik et al., 2019) as pre-trained word embeddings. We replace the standard bidirectional LSTMs with the discovered recurrent neural cells. Also, we set the hidden layer size to 512 and apply variational dropout to the input and output of the RNN layer. We train the network using SGD with a learning rate of 0.1 and a gradient clipping threshold of 5.0. We reduce the learning rate by a factor of 0.25 if the test error does not decrease for 2 epochs.

Table 2 shows a comparison of different methods. Our baseline uses RNN cells generated from random initialized whose F1-score varies greatly and is lower than that of the standard LSTMs. I-DARTS significantly outperforms Random RNNs and DARTS. The best score is achieved when $n = 1$. It indicates that the task prefers a sparse network. Also, we see that our model works with the advanced pre-trained language models in that we replace the LSTM cell to our cell. The I-DARTS architecture yields a new RNN-based state-of-the-art on this task (93.47 F1-score). In Table 2, We find it interesting that Random RNNs are good for NER task. This may result from the design of search space that fit for such tasks substantially. Search space is also a key factor in neural architecture search that new efforts should focus on (Xie et al., 2019a).

We visualize the discovered cells in Figure 3. Each cell is a directed acyclic graph consisting of an ordered sequence of 8 nodes with an activation function applied on each edge. These automatically discovered cells are complex and hard to be designed manually. An interesting phenomenon comes up that the best architecture on language modeling is different from that on name entity recognition. This might result from the fact that different tasks have different inductive bias. Also, this suggests the possibility of architecture selection from the top-$k$ search results on the target task.

## 4 Related Work

Neural architecture search has been proposed to automatically search for better architectures, showing competitive results on several tasks, e.g., image recognition and language modeling. A s-

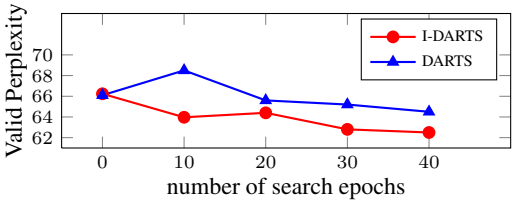


Figure 2: Perplexity vs. search epoch number.

| Model | F1 |
|---|---|
| *best published* | |
| BiLSTM-CRF (Lample et al., 2016) | 90.94 |
| BiLSTM-CRF+ELMo (Peters et al., 2018) | 92.22 |
| BERT Base (Devlin et al., 2018) | 92.40 |
| BERT Large (Devlin et al., 2018) | 92.80 |
| BiLSTM-CRF+PCE (Akbik et al., 2019) | 93.18 |
| Random RNNs w/o pre-trained LM | 90.64 |
| DARTS w/o pre-trained LM | 91.05 |
| I-DARTS ($n = 2$) w/o pre-trained LM | 90.96 |
| I-DARTS ($n = 1$) w/o pre-trained LM | 91.23 |
| Random RNNs | 92.89 |
| DARTS | 93.13 |
| I-DARTS ($n = 2$) | 93.14 |
| I-DARTS ($n = 1$) | 93.47 |

Table 2: F1 scores on the CoNLL-2003 English NER test set.

trand of NAS research focuses on reinforcement learning (Zoph and Le, 2016) and evolutionary algorithm-based (Xie and Yuille, 2017) methods. They are powerful but inefficient. Recent approaches speed up the search process by weight sharing (Pham et al., 2018) and differentiable architecture search (Liu et al., 2019). But there is no discussion on the softmax-local problem in previous work. Moreover, previous methods are often tested on language modeling. It is rare to see studies on these methods for other NLP tasks.

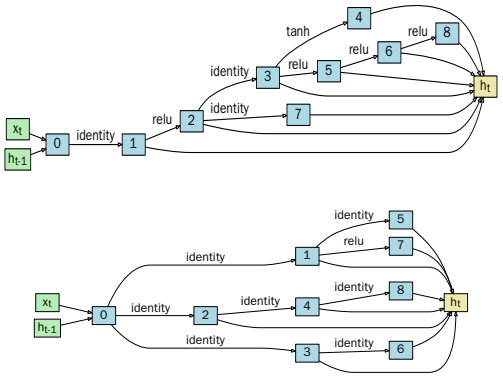


Figure 3: Cells discovered by I-DARTS for language modeling (top) and NER (bottom).

## 5 Conclusions

We improved the DARTS to address the bias problem by removing the softmax-local constraint. Our method is search efficient and discovers several better architectures for PTB language modeling and CoNLL named entity recognition (NER) tasks. We plan to consider the network density problem in search and apply I-DARTS to more tasks in our future study.

## 6 Acknowledgments

This work was supported in part by the National Science Foundation of China (Nos. 61876035, 61732005 and 61432013), the National Key R&D Program of China (No. 2019QY1801) and the Opening Project of Beijing Key Laboratory of Internet Culture and Digital Dissemination Research. We also thank the reviewers for their insightful comments.

## References

Alan Akbik, Tanja Bergmann, and Roland Vollgraf. 2019. Pooled contextualized embeddings for named entity recognition. In *NAACL*.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *COLING*.

Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. 2016. Designing neural network architectures using reinforcement learning. *arXiv preprint arXiv:1611.02167*.

Daphne Koller and Nir Friedman. 2009. *Probabilistic Graphical Models - Principles and Techniques*. MIT press.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Zhiheng Huang and Bing Xiang. 2019. Wenet: Weighted networks for recurrent network architecture search. *arXiv preprint arXiv:1904.03819*.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2019. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and optimizing LSTM language models. In *International Conference on Learning Representations*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1756–1765, Vancouver, Canada. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. 2018. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*.

Erik Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *CoNLL*.

Lingxi Xie and Alan Yuille. 2017. Genetic cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1379–1388.

Saining Xie, Alexander Kirillov, Ross Girshick, and Kaiming He. 2019a. Exploring randomly wired neural networks for image recognition. *arXiv preprint arXiv:1904.01569*.

Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. 2019b. SNAS: stochastic neural architecture search. In *International Conference on Learning Representations*.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. 2018. Breaking the softmax bottleneck: A high-rank RNN language model. In *International Conference on Learning Representations*.

Julian G. Zilly, Rupesh Kumar Srivastava, Jan Koutník, and Jürgen Schmidhuber. 2016. Recurrent highway networks. In *ICML*.

Barret Zoph and Quoc V Le. 2016. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*.

# Using Pairwise Occurrence Information to Improve Knowledge Graph Completion on Large-Scale Datasets

**Esma Balkır**[1,2*], **Masha Naslidnyk**[2], **Dave Palfrey**[2] and **Arpit Mittal**[2]

[1]University of Edinburgh, Scotland, UK
[2]Amazon Research, Cambridge, UK
[1]esma.balkir@ed.ac.uk
[2]{naslidny, dpalfrey, mitarpit}@amazon.co.uk

## Abstract

Bilinear models such as DistMult and ComplEx are effective methods for knowledge graph (KG) completion. However, they require large batch sizes, which becomes a performance bottleneck when training on large scale datasets due to memory constraints. In this paper we use occurrences of entity-relation pairs in the dataset to construct a joint learning model and to increase the quality of sampled negatives during training. We show on three standard datasets that when these two techniques are combined, they give a significant improvement in performance, especially when the batch size and the number of generated negative examples are low relative to the size of the dataset. We then apply our techniques to a dataset containing 2 million entities and demonstrate that our model outperforms the baseline by 2.8% absolute on hits@1.

## 1 Introduction

A Knowledge Graph (KG) is a collection of facts which are stored as triples, e.g. *Berlin is-capital-of Germany*. Even though knowledge graphs are essential for various NLP tasks, open domain knowledge graphs have missing facts. To tackle this issue, there has recently been considerable interest in KG completion methods, where the goal is to rank correct triples above incorrect ones.

Embedding methods such as DistMult (Yang et al., 2014) and ComplEx (Trouillon et al., 2016) are simple and effective methods for this task, but are known to be sensitive to hyperparameter and loss function choices (Kadlec et al., 2017; Lacroix et al., 2018). When paired with the right loss function, these methods need large minibatches and a large number of corrupted triples per each positive triple during training to reach peak performance. [1] This causes memory issues for KGs in the wild, which are several magnitudes bigger than the common benchmarking datasets.

To address the issue of scalability, we develop a framework that could be used with any bilinear KG embedding model. We name our model **JoBi** (**Jo**int model with **Bi**ased negative sampling). Our framework uses occurrences of entity-relation pairs to overcome data sparsity, and to bias the model to score plausible triples higher. The framework trains a base model jointly with an auxiliary model that uses occurrences of pairs within a given triple in the data as labels. For example, the auxiliary model would receive the label 1 for the triple (*Berlin is-capital-of France*) if the pairs (*Berlin is-capital-of*) and (*is-capital-of France*) are present in the training data, while the base model would receive the label 0.

The intuition for using bigram occurrences is to capture some information about restrictions on the set of entities that could appear as the object or subject of a given relation; that information should implicitly correspond to some underlying type constraints. For example, even if *(Berlin is-capital-of France)* is not a correct triple, *Berlin* is the right type for the subject of *is-capital-of*.

Our framework also utilizes entity-relation pair occurrences to improve the distribution of negative examples for contrastive training, by sampling a false triple e.g. (*Berlin is-capital-of France*) with higher probability if the pairs (*Berlin is-capital-of*) and (*is-capital-of France*) both occur in the dataset. This tunes the noise distribution for the task so that it is more challenging, and hence the model needs a fraction of the negative examples compared to a uniform distribution.

We show empirically that joint training is especially beneficial when the batch size is small,

*Work done while the author was an intern

[1]The choice of loss function has a very strong effect on the optimal negative ratio, but with any loss function, larger batches tend to improve the results.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3591–3596, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

and biased negative sampling helps model learn higher quality embeddings with much fewer negative samples. We show that the two techniques are complementary and perform significantly better when combined. We then test JoBi on a large-scale dataset, and demonstrate that JoBi learns better embeddings in very large KGs.

## 2 Background

Formally, given a set of entities $\mathcal{E} = \{e_0, \ldots, e_n\}$ and a set of relations $\mathcal{R} = \{r_0, \ldots, r_m\}$, a Knowledge Graph (KG) is a set triples in the form $\mathcal{G} = \{(h, r, t)\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, where if a triple $(h, r, t) \in \mathcal{G}$, then relation $r$ holds between entities $h$ and $t$. Given such a KG, the aim of KG completion is score each triple in $\mathcal{E} \times \mathcal{R} \times \mathcal{E}$, so that correct triples are assigned higher scores than the false ones. KG embedding methods achieve this by learning dense vector representations for entities and relations through optimizing a chosen scoring function. A class of KG completion models such as RESCAL (Nickel et al., 2012), DistMult (Yang et al., 2014), ComplEx (Trouillon et al., 2016), SimplE (Kazemi and Poole, 2018) and TUCKER (Balažević et al., 2019) define their scoring function to be a bilinear interaction of the embeddings of entities and relations in the triple. For this work we consider DistMult, ComplEx and SimplE as our baseline models due to their simplicity.

**DistMult.** (Yang et al., 2014) is a knowledge graph completion model that defines the scoring function for a triple as a simple bilinear interaction, where the entity has the same representation regardless of whether it appears as the head or the tail entity. For entities $h, t$, relation $r$, and the embeddings $\boldsymbol{h}, \boldsymbol{t}, \boldsymbol{r} \in \mathbb{R}^d$, the scoring function is defined as:

$$s(h, r, t) = \boldsymbol{h}^T \operatorname{diag}(\boldsymbol{r})\, \boldsymbol{t} \tag{1}$$

where $\operatorname{diag}(\boldsymbol{r})$ is a diagonal matrix with $\boldsymbol{r}$ on the diagonal.

**ComplEx.** (Trouillon et al., 2016) is a bilinear model similar to DistMult. Because of its symmetric structure, DistMult cannot model anti-symmetric relations. ComplEx overcomes this shortcoming by learning embeddings in a complex vector space, and defining the embedding of an entity in tail position as the complex conjugate of the embedding in the head position.

Let $\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t} \in \mathbb{C}^d$ be the embeddings for $h, r, t$. The score for ComplEx is defined as follows:

$$s(h, r, t) = \text{Re}\left(\boldsymbol{h}^T \operatorname{diag}(\boldsymbol{r}) \bar{\boldsymbol{t}}\right) \tag{2}$$

Where $\overline{\boldsymbol{a}}$ denotes the complex conjugate of $\boldsymbol{a}$, and $\text{Re}(\boldsymbol{a})$ denotes the real part of the complex vector $\boldsymbol{a}$.

**SimplE.** (Kazemi and Poole, 2018) is also a bilinear model similar to DistMult. Each entity has two associated embeddings $\boldsymbol{e}_1, \boldsymbol{e}_2 \in \mathbb{R}^d$, where one is the representation of $e$ as the head, and the other as the tail entity of the triple. Each relation also has two associated embeddings: $\boldsymbol{r}$ and $\boldsymbol{r}^{-1}$, where $\boldsymbol{r}^{-1}$ is the representation for the reverse of $r$. The score function is defined as:

$$\begin{aligned} s(h, r, t) = 1/2\,(\boldsymbol{h}_1)^T\ \operatorname{diag}(\boldsymbol{r})\boldsymbol{t}_1 \\ +1/2\,(\boldsymbol{t}_2)^T\ \operatorname{diag}(\boldsymbol{r}^{-1})\boldsymbol{h}_2 \end{aligned} \tag{3}$$

## 3 Joint framework

JoBi contains two copies of a bilinear model, where one is trained on labels of triples, and the other on occurrences of entity-relation pairs within the triples. For the pair module, we label a triple $(h, r, t)$ correct if there are triples $(h, r, t')$ and $(h', r, t)$ in the training set for some $t'$ and $h'$.

The scoring functions for the two models are $s_{\text{bi}}$ and $s_{\text{tri}}$, for the pair and the triple modules respectively. We tie the weights of the entity embeddings, but let the embeddings for the relations be optimized separately. The equations using ComplEx as the base model are as follows:

$$s_{\text{tri}}(h, r, t) = \text{Re}\left(\boldsymbol{h}^T \operatorname{diag}(\boldsymbol{r}_{\text{tri}}) \bar{\boldsymbol{t}}\right) \tag{4}$$

$$s_{\text{bi}}(h, r, t) = \text{Re}\left(\boldsymbol{h}^T \operatorname{diag}(\boldsymbol{r}_{\text{bi}}) \bar{\boldsymbol{t}}\right) \tag{5}$$

We define the framework for DistMult and SimplE analogously. During training, we optimize the two jointly, but use only $s_{\text{tri}}$ during test time. Hence, the addition of the auxiliary module has no effect on the number of final parameters of the trained model. Note that even during training, this doesn't increase model complexity in any significant way since the number of relations in KGs are often a fraction of the number of entities.

For each triple in the minibatch, we generate $n_{\text{neg}}$ negative examples per positive triple by randomly corrupting the head or the tail entity. For $s_{\text{tri}}$, we use the negative log-likelihood of softmax as the loss function, and for $s_{\text{bi}}$ we use binary cross

| Dataset | #entities | #relations | #train |
|---|---|---|---|
| FB15K | 14,951 | 1,345 | 483,142 |
| FB15K-237 | 14,541 | 237 | 272,115 |
| YAGO3-10 | 123,182 | 37 | 1,079,040 |
| FB1.9M | 1,892,241 | 3,247 | 19,323,513 |

Table 1: Statistics of datasets used in experiments.

entropy loss. We combine the two losses via a simple weighted addition with a tunable hyperparameter $\alpha$:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{tri}} + \alpha \mathcal{L}_{\text{bi}} \quad (6)$$

**Biased negative sampling.** We also examine the effect of using the pair cooccurrence information for making the contrastive training more challenging for the model. For this, we keep the model as is, but with probability $p$, instead of corrupting the head or the tail of the triple with an entity chosen uniformly at random, we corrupt it with an entity that is picked with uniform probability from the set of entities that occur as the head or tail entity of the relation in the given triple. To illustrate, when sampling a negative tail entity for the tuple *Berlin is-capital-of*, this method causes the model to pick *France* with higher probability than *George Orwell* if *France* but not *George Orwell* occurs as the head entity for the relation *is-capital-of* in the training data.

## 4 Experiments

We perform our experiments on standard datasets FB15K (Bordes et al., 2013), FB15K-237 (Toutanova et al., 2015), YAGO3-10 (Dettmers et al., 2018), and on a new large-scale dataset FB1.9M which we construced from FB3M (Xu and Barbosa, 2018). [2] We focus on YAGO3-10 since it is 10 times larger than the other two and better reflects how the performance of the models scale. We present the comparison of the sizes of these datasets in Table 1, and further details could be found in Appendix A.

For evaluation, we rank each triple $(h, r, t)$ in the test set against $(h', r, t)$ for all entities $h'$, and similarly against $(h, r, t')$ for all entities $t'$. We filter out the candidates that have occurred in training, validation or test set as described in Bordes et al. (2013), and we report average hits@1, 3, 10 and mean reciprocal rank (MRR).

We re-implement all our baselines and obtain very competitive results. In our preliminary experiments on baselines, we found that the choice of loss function had a large effect on performance, with negative log-likelihood (NLL) of softmax consistently outperforming both max-margin and logistic losses. Larger batch sizes lead to better performance. With NLL of sampled softmax, we found that increasing the number of generated negatives steadily increases performance[3], and state-of-the-art results could be reached by using the full softmax as used in Joulin et al. (2017) and Lacroix et al. (2018). This technique is possible for standard benchmarks but not for large KGs, and we report results in Appendix D for all datasets small enough to allow for full contrastive training. However, our main experiments use NLL of sampled softmax since our focus is on scalability. Note that results with full softmax (Appendix D) demonstrate that our implementation of baselines is very competitive.Our implementation of ComplEx performs significantly better than ConvE (Dettmers et al., 2018) on two out of the three datasets, and come close to results of Lacroix et al. (2018) who use extremely large embeddings as well as full softmax, thus cannot be scaled. Our code is publicly available. [4]

For most of our experiments, we choose to use ComplEx as the base for our model (**JoBi ComplEx**), since this configuration consistently outperformed others in preliminary experiments. To test the effect of our techniques on different bilinear models, we report results with DistMult (**JoBi DistMult**) and SimplE (**JoBi SimplE**) on FB15K-237.

**Discussion.** It could be seen in Table 2 that JoBi ComplEx outperforms both ComplEx and DistMult on all three standard datasets, on all the metrics we consider. For Hits@1, JoBi Complex outperforms baseline ComplEx by 4% on FB15K-237, 6.4% on FB15K and 5.6% on YAGO3-10.

Moreover, results in Table 2 demonstrate that JoBi improves performance on DistMult and SimplE. It should be noted that on FB15K-237, all JoBi models outperform all the baseline models, regardless of the base model used.

Lastly, results on FB1.9M (Table 3) demonstrate that JoBi improves performance on this very

[2] We do not perform experiments on WordNet derived datasets WN18 or WN18RR because bigram modelling would not provide any information – all entities are synsets and almost all can occur as an object or subject to all the possible relations.

[3] This effect is not observed when using logistic-loss or max-margin loss

[4] https://github.com/awslabs/joint_biased_embeddings

| FB15K-237 | *h@1* | *h@3* | *h@10* | *MRR* |
|---|---|---|---|---|
| SimplE | 0.160 | 0.268 | 0.430 | 0.248 |
| DistMult | 0.158 | 0.271 | 0.432 | 0.247 |
| ComplEx | 0.159 | 0.275 | 0.441 | 0.25 |
| JoBi SimplE | 0.188 | 0.301 | 0.461 | 0.277 |
| JoBi DistMult | **0.205** | 0.316 | 0.466 | 0.29 |
| JoBi ComplEx | 0.199 | **0.319** | **0.479** | **0.29** |
| **FB15K** | *h@1* | *h@3* | *h@10* | *MRR* |
| DistMult | 0.587 | 0.785 | 0.867 | 0.697 |
| ComplEx | 0.617 | 0.803 | 0.874 | 0.72 |
| JoBi ComplEx | **0.681** | **0.824** | **0.883** | **0.761** |
| **YAGO3-10** | *h@1* | *h@3* | *h@10* | *MRR* |
| DistMult | 0.252 | 0.407 | 0.568 | 0.357 |
| ComplEx | 0.277 | 0.44 | 0.589 | 0.383 |
| JoBi ComplEx | **0.333** | **0.477** | **0.617** | **0.428** |

Table 2: Performance on different datasets against baselines, where h@$k$ denotes hits at $k$. Results are reported on test sets with the best parameters found in grid search for each model.

| ComplEx | *h@1* | *h@3* | *h@10* | *MRR* |
|---|---|---|---|---|
| Baseline | 0.424 | 0.598 | 0.721 | 0.530 |
| JoBi | **0.452** | **0.615** | **0.726** | **0.550** |

Table 3: Performance on the large-scale FB1.9M dataset, measured against the best performing baseline.

| | # epochs | training time |
|---|---|---|
| ComplEx | 70 | 5 days 5 hours 8 minutes |
| JoBi ComplEx | 30 | 4 days 19 minutes |

Table 4: Runtimes of ComplEx and JoBi Complex on FB1.9M.

large dataset, where it is not possible to perform softmax over the entire set of entities, or have very large embedding sizes due to memory constraints.

Although one epoch for JoBi takes slightly longer than the baseline, JoBi converges in fewer epochs, resulting in shorter running time overall. We report running times on FB1.9M in Table 4.

**Comparison with TypeComplex** For results of TypeComplex, Jain et al. (2018) use a wider set of negative ratios in their grid search than we do. To isolate the effects of the different models from hyperparameter choices, we set the negative ratio for our model to be 400 to match the setting on their best performing models. We keep the other hyperparameters the same as the best performing models for the previous experiments.

Jain et al. (2018) use a modified version of the ranking evaluation procedure to report their results, where they only rank the tail entity against all other entities. To be able to compare our model to theirs, we also report the performance of our framework on this modified metric. The results

| FB15K-237 | *h@1* | *h@3* | *h@10* | *MRR* |
|---|---|---|---|---|
| ComplEx | 0.209 | 0.347 | 0.535 | 0.314 |
| TypeComplex | **0.296** | - | 0.575 | **0.389** |
| JoBi ComplEx | 0.276 | 0.416 | **0.587** | 0.377 |
| **FB15K** | *h@1* | *h@3* | *h@10* | *MRR* |
| ComplEx | 0.630 | 0.818 | 0.895 | 0.734 |
| TypeComplex | 0.663 | - | 0.885 | 0.754 |
| JoBi ComplEx | **0.702** | 0.847 | **0.906** | **0.782** |
| **YAGO3-10** | *h@1* | *h@3* | *h@10* | *MRR* |
| ComplEx | 0.412 | 0.587 | 0.701 | 0.516 |
| TypeComplex | **0.516** | - | 0.702 | 0.587 |
| JoBi ComplEx | 0.507 | 0.647 | **0.742** | **0.591** |

Table 5: Comparison with TypeComplex where the scores are calculated ranking only the tail entities. Results for TypeComplex are taken from Jain et al. (2018). h@$k$ denotes hits at $k$.

for these experiments can be found in Table 5.

Our model generally outperforms TypeComplex by a large margin on hits@10. It also outperforms TypeComplex on MRR by a moderate margin except on FB15K-237, the smallest dataset. On the other hand, TypeComplex outperforms our model on hits@1 in two out of the three datasets. In fact for FB15K, TypeComplex does worse on hits@10 compared to the baseline model. This suggests that TypeComplex may be compromising on hits@k where k is larger to improve the hits@1 metric, which might be undesirable depending on the application.

**Qualitative analysis.** We analyzed correct predictions made by JoBi ComplEx but not regular ComplEx. Among relations in YAGO3-10, major gains can be observed for *hasGender* (Appendix C). The improvement comes solely from tail-entity predictions, with hits@1 increasing from 0.22 to 0.86. Furthermore, we found that the errors made by ComplEx are exactly of the kind that can be mitigated by enforcing plausibility: ComplEx predicts an object that is not a gender (e.g. a sports team or a person) 65% of the time; JoBi makes such an obvious mistake only 2% of the time.

**Ablation studies.** We compare joint training without biased sampling (**Joint**) and biased sampling without joint training (**BiasedNeg**) to the full model JoBi on YAGO3-10. The results can be found in Table 6. We also conduct experiments to isolate the effect of our techniques on varying batch sizes and negative ratios. The results for this experiment are presented in Figures 1 and 2. Training details can be found in Appendix B.

| | *h@1* | *h@3* | *h@10* | *MRR* |
|---|---|---|---|---|
| Baseline | 0.277 | 0.44 | 0.589 | 0.383 |
| BiasedNeg | 0.276 | 0.427 | 0.568 | 0.375 |
| Joint | 0.287 | 0.447 | 0.601 | 0.392 |
| JoBi | **0.333** | **0.477** | **0.617** | **0.428** |

Table 6: Results of ablation study on ComplEx model.

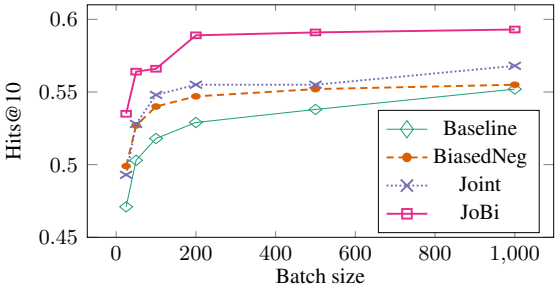


Figure 1: Performances on YAGO3-10 with different batch sizes

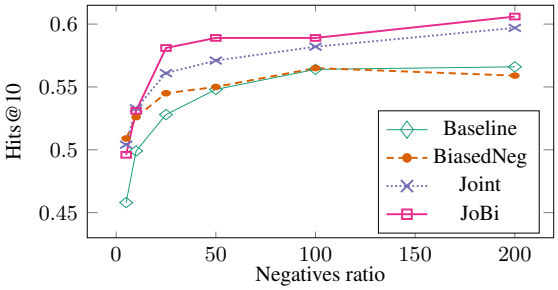


Figure 2: Performances on YAGO3-10 with different negative ratios

In Table 6 it can be seen that Joint on its own gives a slight performance boost over the baseline, and BiasedNeg performs slightly under the baseline on all measures. However, combining our two techniques in JoBi gives 5.6% points improvement on hits@1. This suggests that biased negative sampling increases the efficacy of joint training greatly, but is not very effective on its own.

Figure 1 and 2 shows that JoBi not only consistently performs the best over the entire range of parameters, but also delivers a performance improvement that is especially large when the batch size or the negative ratio is small. This setting was designed to reflect the training conditions on very large datasets. It can be seen that BiasedNeg is more robust to low values of negative ratios, and both BiasedNeg and Joint alone show less deterioration in performance as the batch size decreases. When these two methods are combined in JoBi, the training becomes more robust to different choices on both these parameters.

The reason behind BiasedNeg performing worse on its own but better with Joint could be the choice of binary cross entropy loss for the pair module. We speculate that as the negative ratio increases, the ratio of negative to positive examples for this module becomes more skewed. Biasing the negative triples in the training alleviates this problem by making the classes more balanced, and allows the joint training to be more effective.

### 4.1 Related work

Utilizing pair occurrences for embedding models have been considered before, both as explicit model choices and as negative sampling strategies. Chang et al. (2014) and Krompaß et al. (2015) use pair occurrences to constrain the set of triples to be used in the optimization procedure. For methods that rely on SGD with contrastive training, this translates to a special case of our biased sampling method where $p = 1$. Garcia-Durán et al. (2016) present *TATEC*, a model that combines bigram and trigram interactions. The trigram model uses a full matrix representation for relations, and hence has many more parameters compared to our model. Jain et al. (2018) present *JointDM* and *JointComplex*, which could be viewed as a simplification of *TATEC*. Unlike our model, both of these methods use the bigram terms both in training and evaluation, do not share any of the embeddings between two models, and do not provide supervision based on pair occurrences in the data. Other methods that have been considered for improving the negative sampling procedure includes adversarial (Cai and Wang, 2018) and self-adversarial (Sun et al., 2019) training. None of these methods focus on improving the models to scale to large KGs.

## 5 Conclusion

We have presented a joint framework for KG completion that utilizes entity-relation pair occurrences as an auxiliary task, and combined it with a technique to generate informative negative examples with higher probability. We have shown that joint training makes the model more robust to smaller batch sizes, and biased negative sampling to different values of the number of generated negative samples. Furthermore, these techniques perform well above baselines when combined, and are effective on a very large KG dataset. Applying JoBi to non-bilinear models is also possible, but left for future work.

## References


Ivana Balažević, Carl Allen, and Timothy M Hospedales. 2019. Tucker: Tensor factorization for knowledge graph completion. *arXiv preprint arXiv:1901.09590*.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, pages 2787–2795. Curran Associates Inc.

Liwei Cai and William Yang Wang. 2018. Kbgan: Adversarial learning for knowledge graph embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1470–1480.

Kai-Wei Chang, Scott Wen-tau Yih, Bishan Yang, and Chris Meek. 2014. Typed tensor decomposition of knowledge bases for relation extraction. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Tim Dettmers, Minervini Pasquale, Stenetorp Pontus, and Sebastian Riedel. 2018. Convolutional 2D Knowledge Graph Embeddings. In *Proceedings of the 32th AAAI Conference on Artificial Intelligence*, pages 1811–1818.

Alberto Garcia-Durán, Antoine Bordes, Nicolas Usunier, and Yves Grandvalet. 2016. Combining two and three-way embedding models for link prediction in knowledge bases. *Journal of Artificial Intelligence Research*, 55:715–742.

Prachi Jain, Pankaj Kumar, Soumen Chakrabarti, and others. 2018. Type-Sensitive Knowledge Base Inference Without Explicit Type Supervision. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 75–80.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Maximilian Nickel, and Tomas Mikolov. 2017. Fast Linear Model for Knowledge Graph Embeddings. *arXiv preprint arXiv:1710.10881*.

Rudolf Kadlec, Ondrej Bajgar, and Jan Kleindienst. 2017. Knowledge Base Completion: Baselines Strike Back. *arXiv preprint arXiv:1705.10744*.

Seyed Mehran Kazemi and David Poole. 2018. SimplE Embedding for Link Prediction in Knowledge Graphs. *arXiv preprint arXiv:1802.04868*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Denis Krompaß, Stephan Baier, and Volker Tresp. 2015. Type-constrained representation learning in knowledge graphs. In *International Semantic Web Conference*, pages 640–655.

Timothe Lacroix, Nicolas Usunier, and Guillaume Obozinski. 2018. Canonical Tensor Decomposition for Knowledge Base Completion. In *Proceedings of the 35th International Conference on Machine Learning*.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2012. Factorizing YAGO: scalable machine learning for linked data. In *Proceedings of the 21st International Conference on World Wide Web*, pages 271–280, New York, New York, USA. ACM.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. YAGO. In *Proceedings of the 16th International Conference on World Wide Web - WWW '07*, page 697, New York, New York, USA. ACM Press.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing Text for Joint Embedding of Text and Knowledge Bases. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1499–1509, Lisbon. ACM.

Tho Trouillon, Johannes Welbl, Sebastian Riedel, Eric Gaussier, and Guillaume Bouchard. 2016. Complex Embeddings for Simple Link Prediction. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 2071 – 2080.

Peng Xu and Denilson Barbosa. 2018. Investigations on Knowledge Base Embedding for Relation Prediction and Extraction. *arXiv preprint arXiv:1802.02114*.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. *arXiv preprint arXiv:1412.6575*.

# Single Training Dimension Selection for Word Embedding with PCA

**Yu Wang**
Apple
w.y@apple.com

## Abstract

In this paper, we present a fast and reliable method based on PCA to select the number of dimensions for word embeddings. First, we train one embedding with a generous upper bound (e.g. 1,000) of dimensions. Then we transform the embeddings using PCA and incrementally remove the lesser dimensions one at a time while recording the embeddings' performance on language tasks. Lastly, we select the number of dimensions while balancing model size and accuracy. Experiments using various datasets and language tasks demonstrate that we are able to train 10 times fewer sets of embeddings while retaining optimal performance. Researchers interested in training the best-performing embeddings for downstream tasks, such as sentiment analysis, question answering and hypernym extraction, as well as those interested in embedding compression should find the method helpful.

## 1 Introduction

Word embeddings constitute an integral part of the implementation of numerous NLP tasks ranging from sentiment classification (Lin et al., 2015), nationality classification (Ye et al., 2017), classification of behavior on Twitter (Wang et al., 2017), to measuring document similarity (Kusner et al., 2015). The strength of word embeddings stems from their embedding words into low dimensional continuous vector space (Mnih and Kavukcuoglu, 2013; Pennington et al., 2014; Tang et al., 2015; Raunak, 2017).

Various algorithms have been proposed for learning words' vector space representations, including most notably Mikolov et al. (2013b), Pennington et al. (2014), and Nickel and Kiela (2017). However, the exact definition of 'low' dimensionality is rarely explored. As pointed out by Yin and Shen (2018), the most frequently used dimensionality is 300, largely due to the fact that the early influential papers (Mikolov et al., 2013b; Pennington et al., 2014) used 300. Other often used dimensionalities include 200 (Tang et al., 2015; Ling et al., 2016; Nickel and Kiela, 2017) and 500 (Tang and Liu, 2009; Perozzi et al., 2014).

The impact of vector dimension on embeddings' performance is well known (Lai et al., 2016). With too few dimensions, the model will underfit; with too many dimensions the model will overfit. Both undercut the embeddings' performance (Yin and Shen, 2018). What is also known is that the size of the embeddings will grow linearly with the vector dimension (Ling et al., 2016; Raunak, 2017). What is less known is how to identify the optimal vector dimension given any dataset. The method we propose here helps fill this gap.

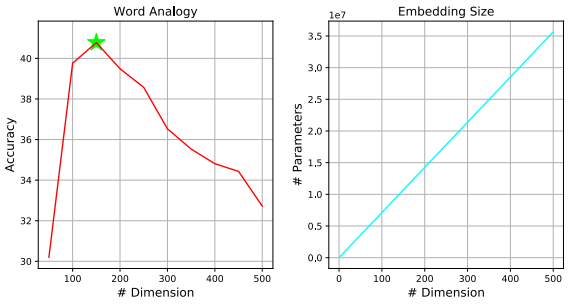


Figure 1: Left: Accuracy for the word analogy task as a function of vector dimension (dimension incremental is 50). Right: The number of embedding parameters as a function of vector dimension. Results are generated using the Text8 dataset with a vocabulary size of 71,291.

We offer a fast and reliable PCA-based method that (1) only needs to train the embeddings once and (2) is able to select vector dimension with competitive performance.[1] First, we train one embedding with a generous upper bound (e.g. 1,000)

[1] As such, the selected dimension is a Pareto-optimal dimension that balances embedding size and performance.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3597–3602,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

of dimensions. Then we transform the embeddings using PCA and incrementally remove the lesser dimensions one at a time while recording the embeddings' performance on language tasks. Lastly we calculate the best dimensionality and return the corresponding embeddings.

Experiments using various datasets and language tasks reveal three key observations:

- The optimal dimensionality calculated on the basis of PCA agrees with that by grid search.
- The resulting embedding is competitive against the one selected by grid search.
- Different upper-bound dimensionalities (e.g. 500, 1000) point to the same optimal dimensionality.

Researchers interested in downstream tasks, such as sentiment analysis (Lin et al., 2015), question answering (Devlin et al., 2018) and hypernym extraction (Chen et al., 2018), as well as those interested in embedding compression should find the method helpful.

## 2 Related Work

Our work draws inspiration from Yin and Shen (2018). The authors build on the Latent Semantics Analysis (LSA) approach and slide $k$ from a lower bound (e.g. 10) to a generous upper bound (e.g. 1,000) in $E = U_{1:k} D^{\alpha}_{1:k,1:k}$, where U and D come from the singular-value decomposition of the signal matrix and $\alpha$ is a hyperparameter to be tuned. For each $k$, the authors generate one corresponding embedding and compare it with the simulated oracle embedding. The $k$ that yields the smallest loss is selected. In a similar vein, our work bypasses the problem of training multiple embeddings, often necessitated by grid search, by sliding over all the k values of PCA. Compared with Yin and Shen (2018), our method is easier to implement, as we do not rely on, e.g, Monte Carlo simulations of the oracle embeddings.

At a deeper level, our work is also connected to Yin and Shen (2018) in terms of the trade-off between bias and variance. Yin and Shen (2018) propose pairwise inner product (PIP) loss to measure the quality of an embedding. They decompose the PIP loss into a bias term and a variance term, where reducing the dimension increases the bias term but reduces the variance. They show that the bias-variance trade-off reflects the signal-to-noise ratio in dimension selection. While there is no exact 1-1 mapping from their theorem to our work, we do have analogous discussion in Section 3. The PCA step in our algorithm enables us to identify and drop dimensions that (1) contribute less to the explained variance in the embedding and yet (2) contribute equally to cosine similarity. In essence, our PCA step is removing dimensions with low signal-to-noise ratios.

Our work also draws strength from the literature on post-processing embeddings. Mu and Viswanath (2018) demonstrate that removing the top dominating directions in the PCA-transformed embeddings helps improve the embeddings' performance in word analogy and similarity tasks. Building on that, Raunak (2017) shows that by performing another iteration of PCA and dropping the bottom directions, one can further improve a model's performance as well as reduce its size. Both works focus on improving pre-trained embeddings's performance in terms of accuracy and size. By contrast, our algorithm selects the optimal dimensionality before the actual training.

In addition, our work is related to a few recent studies on model compression (Faruqui et al., 2015; Ling et al., 2016; Shu and Nakayama, 2018). In particular, Ling et al. (2016) seek to drop the least significant digits to reduce the embeddings' size. By comparison, our method removes the least significant dimensions (in terms of explained variance (Bishop, 2006)). It should be noted that the two methods complement each other, as one focuses on dimension selection whereas the other on limited precision representation.

## 3 Algorithm

In this section, we formally describe how to select a competitive dimensionality by training one embedding. We state the proposed algorithm in Algorithm 1.

First, we note that the PCA transformation (when retaining all the N dimensions) does not affect embeddings' performance on word similarity tasks. Any potential performance gain should come from dropping the lesser dimensions. By "lesser dimensions," we mean the dimensions that contribute little to the explanation of variance (Bishop, 2006).

In Figure 2, we first transform an embedding

**Algorithm 1** Select the Optimal Dimensionality using PCA

Set dimension upper bound N; select language task from {word analogy, similarity}
Train embedding $E$ with N dimensions
Transform $E$ using PCA: $(u_1, u_2, ..., u_N; \tilde{E}) \leftarrow \text{PCA}(E)$, where $u_1, u_2, .., u_N$ are the new basis vectors, $\tilde{E}$ represents the transformed coefficients
**for** $i = N$ to 2 **do**
  $E$ = $E$-$\tilde{E}_{:,i} \cdot u_i$, where $\tilde{E}_{:,i}$ represents the $i$th column of $\tilde{E}$ and each scalar in $\tilde{E}_{:,i}$ scales vector $u_i$
  Evaluate $E$ on the selected language task, record (i, metric)
**end for**
Return the selected dimension: $[\underset{i}{\text{argmax}} f(\text{i, metric})]$-1, where $f$ is a score function that balances performance and model size and i is between 2 and N

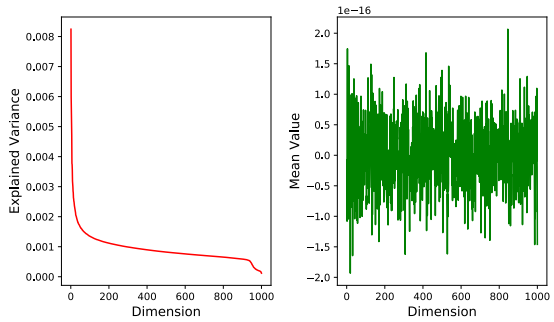


Figure 2: Left: explained variance drops sharply after the top 100 dimensions. Right: all the 1,000 dimensions have roughly the same mean value.

using PCA, so that each new dimension represents a principal component. We show that the explained variance goes drastically down for dimensions beyond the 100th and stays relatively stable for 100th-1,000th dimensions. In terms of magnitude, the first dimension explains 69.2 times more variance than the last dimension.

While different dimensions contribute differently to the explained variance, they nonetheless contribute equally to the calculation of inner product (Figure 2, right). Therefore, the lesser dimensions, with less variance but equal weighting, effectively decreases the discriminative power of the model. Removing these lesser dimensions enables us to focus on the more discriminative dimensions. To identify optimal dimensionality, beyond which all dimensions are considered lesser, we turn to experiments in Section 4.

## 4 Experiments

Given the popularity of the word2vec model (Mikolov et al., 2013b), we use Skip-gram as the embedding algorithm. Following Yin and Shen (2018) and Grave et al. (2017), we use the widely used benchmark datasets, Text8 (Mahoney, 2011) and WikiText-103 (Merity et al., 2017), as the training datasets. For ground truth, we train 20 embeddings, with dimensions ranging from 50 to 1,000 at an interval of 50 as well as two embeddings with dimensions of 5 and 25.[2] Here we have made the implicit assumption that 1,000 is an upper bound for the embedding space's dimensionality. For each embedding, we train 200 epochs (Pennington et al., 2014; Shazeer et al., 2016) and keep only the checkpoint that performs best on the word analogy task (Mikolov et al., 2013a). Our experiments focus on (1) comparing our method with grid-search based ground truth and (2) examining consistency between different upper bounds.

### 4.1 Performance Compared with Grid Search

In this subsection, we compare the optimal dimensionality that our method calculates with the ground truth (Figure 3). We perform the comparison across three testing datasets: Wordsim 353 (Finkelstein et al., 2002), RW Stanford (Luong et al., 2013) and MTurk 771 (Halawi et al., 2012). Figure 3 demonstrates that our PCA based method (with one training only) is able to uncover the optimal dimensionality. Table 1 reports the distance in selected dimensionalities.

We also observe that the optimal embedding that results from our method (with retraining) is competitive against the optimal embedding found using grid search. In Figure 3, we mark out the respective optimal performances of the two approaches in similarity tasks. In Table 2, we further

[2] A larger interval will save researchers more time, but may result in a sub-optimal dimensionality. While a smaller interval may give more accurate results, it requires training more sets of embeddings proportionally. Our proposed method, by contrast, can scan through all the dimensions at the finest granularity, 1, at virtually no extra cost.

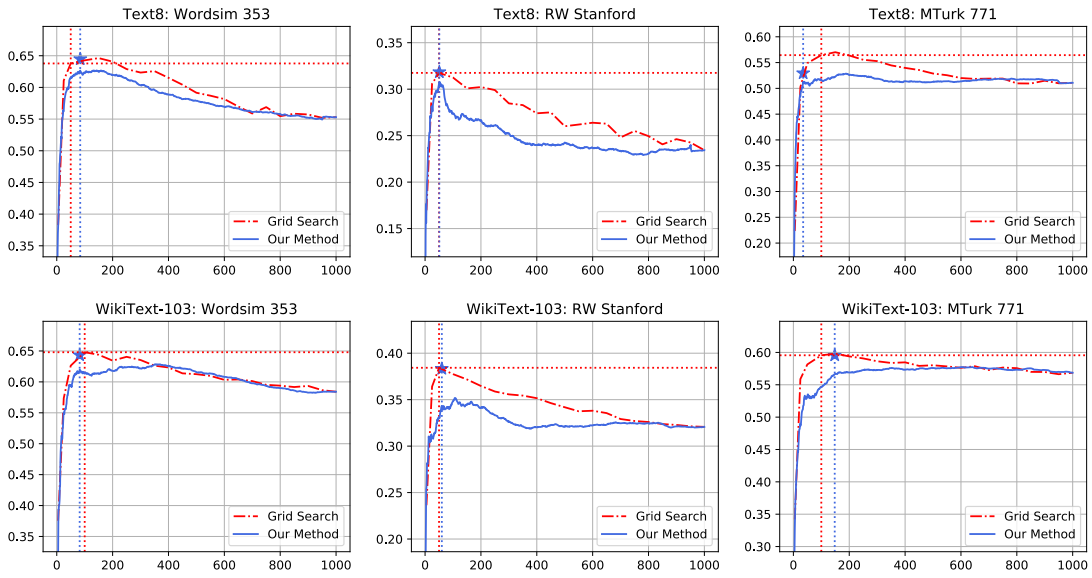


Figure 3: Across different benchmarks and different training datasets, the optimal dimensionality that our method (blue curve) identifies closely matches grid search (red curve). The top row is based on Text8. The bottom row is based on WikiText-103. The vertical dotted lines represent the optimal dimensionality for the respective curves. The horizontal dotted line represents the performance of grid search and the blue star marks the performance of our method. The score function is $f$(i, metric)= metric-50$\times$i. All the curves are averaged over 5 random runs.

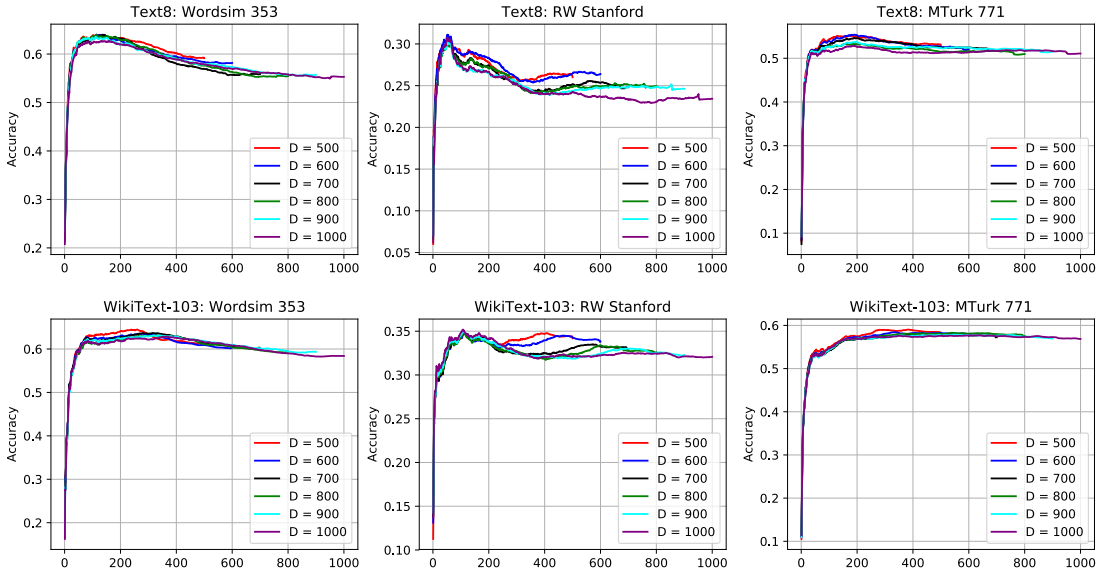


Figure 4: Across different benchmarks and different training datasets, consistency is observed when the upper bound is set to 500, 600, 700, 800, 900 and 1,000. The curves trace closely each other, in particular for WordSim 353 and MTurk 771. The top row is based on the Text8 dataset. The bottom row is based on the WikiText-103 dataset. All the curves are averaged over 5 random runs. Best viewed in color.

report the optimal performance achieved by grid search and our method as well as their relative performance. Even though we have only trained one embedding (and one retraining), our method, on average, is able to achieve 100.2% (WordSim 353) to 96.9% (MTurk 771) of the optimal performance by grid-searching through 22 sets of embeddings.

### 4.2 Consistency across Upper Bounds

One hyperparameter involved in our method is the upper bound. Intuitively, we expect the upper bound should be higher for larger datasets. In this

Table 1: Distance in Dimensionalities between Grid Search and Our Proposed Method (Numbers within the bucket size 50 are in bold)

| Datasets | WordSim | RW Stanford | MTurk |
|---|---|---|---|
| Text8 | **34** | **1** | 65 |
| WikiText | **18** | **10** | **48** |
| *Average* | **26** | **5.5** | 66.5 |

Table 2: Performance (*Correlation*) Comparison between Grid Search and Our Proposed Method

| Datasets | Method | WordSim | RW Stanford | MTurk |
|---|---|---|---|---|
| Text8 | G.S. | 63.8 | 31.8 | 56.4 |
| | Ours | 64.5 (101.1%) | 31.8(100.3%) | 52.9 (93.8%) |
| WikiText | G.S. | 64.8 | 38.4 | 59.6 |
| | Ours | 64.3 (99.3%) | 38.3 (99.7%) | 59.6 (100%) |
| *Average* | - | 100.2% | 100% | 96.9% |

subsection, we demonstrate our method is robust against different upper bounds.

In Figure 4, we vary the dimension from 500 to 1,000 at an increment of 100. We observe that the dimensionality our method selects is consistent across different upper bounds. Based on the demonstrated consistency, different upper bounds can be selected and still the optimal dimensionality can be uncovered as long as the chosen upper bound is larger than the optimal dimensionality.

### 4.3 Efficiency Compared with Grid Search

In this subsection, we report the running time of our algorithm and compare it with that of grid search. We have recorded the running time of our experiments and that of the PCA transformations. We average them over 5 random runs (Figure 5).

For Text8, grid search takes 22,801 minutes cumulatively. Training a 1,000-dimension embedding takes 1,724 minutes. The PCA step takes 22 minutes (note that for each embedding we only need one PCA operation). This represents a 13.1x speedup for our method. For WikiText-103, grid search takes 132,652 minutes. Training a 1,000-dimension embedding takes 10,448 minutes. The PCA step takes 34 minutes. This represents a 12.7x speedup.

We note that the comparison results are dependent on grid granularity. A coarser grid search could save researchers more time, at the cost of performance loss.

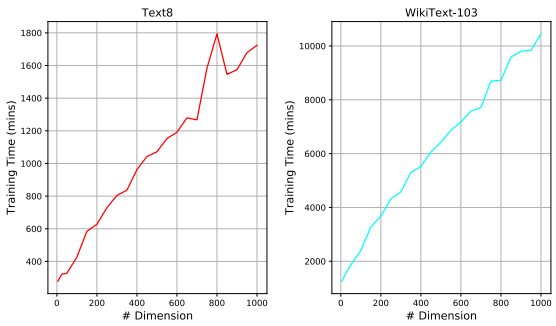


Figure 5: Left: training time for Text8 for embeddings of different dimensions for 200 epochs using 6 CPUs. Right: training time for WikiText-103 for embeddings of different dimensions for 200 epochs using 10 CPUs. All results are averaged over 5 random runs.

## 5 Conclusion

In this paper, we provided a fast and reliable method based on PCA to select the number of dimensions for training word embeddings. First, we train one embedding with a generous upper bound (e.g. 1,000) of dimensions. Then we transform the embedding using PCA and incrementally remove the lesser dimensions while recording the embeddings' performance on language tasks. Experiments demonstrate that (1) our method is able to identify the optimal dimensionality, (2) the resulting embedding has competitive performance against grid search, and (3) our method is robust against the selection of the upper bound.

## Acknowledgement

The authors would like to thank Hang Zhao and Srinivasan Venkatachary for supporting this project, would like to thank Russ Webb, Arnab Ghoshal, and Jaewook Chung for their insightful comments on early versions of this paper, and would like to thank the anonymous EMNLP-IJCNLP reviewers for their reviews and suggestions.

## References

Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer.

Hong-You Chen, Cheng-Syuan Lee, Keng-Te Liao, and Shou-De Lin. 2018. Word relation autoencoder for unseen hypernym extraction using word embeddings. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4834–4839.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint*.

Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah A. Smith. 2015. Sparse overcomplete word vector representations. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 1491–1500.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Gadi Wolfman Zach Solan, and Eytan Ruppin. 2002. Placing search in context: the concept revisited. *ACM Transactions on Information Systems*, 20(1):116–131.

Edouard Grave, Armand Joulin, and Nicolas Usunier. 2017. Improving neural language models with a continuous cache. *ICLR 2017*.

Guy Halawi, Gideon Dror, Evgeniy Gabrilovich, and Yehuda Koren. 2012. Large-scale larning of word relatedness with constraints. *KDD*.

Matt J. Kusner, Yu Sun, Nicholas I. Kolkin, and Kilian Q. Weinberger. 2015. From Word Embeddings To Document Distances. *Proceedings of the 32 nd International Conference on Machine Learning*.

Siwei Lai, Kang Liu, Shizhu He, and Jun Zhao. 2016. How to generate a good word embedding. *IEEE Intelligent Systems*, 31(6).

Yiou Lin, Hang Lei, Jia Wu, and Xiaoyu Li. 2015. An Empirical Study on Sentiment Classification of Chinese Review using Word Embedding. *29th Pacific Asia Conference on Language, Information and Computation*, pages 258 – 266.

Shaoshi Ling, Yangqiu Song, and Dan Roth. 2016. Word embeddings with limited memory. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 387–392.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*.

Matt Mahoney. 2011. Large text compression benchmark.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. *ICLR 2017*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient Estimation of Word Representations in Vector Space. *ICLR Workshop*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed Representations of Words and Phrases and their Compositionality. *NIPS'13 Proceedings of the 26th International Conference on Neural Information Processing Systems*, 2:3111–3119.

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. *NIPS'13 Proceedings of the 26th International Conference on Neural Information Processing Systems*, 2.

Jiaqi Mu and Pramod Viswanath. 2018. All-but-the-top: Simple and effective postprocessing for word representations. *Sixth International Conference on Learning Representations (ICLR 2018)*.

Maximillian Nickel and Douwe Kiela. 2017. Poincaré embeddings for learning hierarchical representations. *Advances in Neural Information Processing Systems 30 (NIPS 2017)*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online Learning of Social Representations. *KDD '14 Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 701–210.

Vikas Raunak. 2017. Simple and effective dimensionality reduction for word embeddings. *Advances in Neural Information Processing Systems 30 (NIPS 2017) LLD Workshop*.

Noam Shazeer, Ryan Doherty, Colin Evans, and Chris Waterson. 2016. Swivel: Improving Embeddings by Noticing What's Missing. *https://arxiv.org/abs/1602.02215*.

Raphael Shu and Hideki Nakayama. 2018. Compressing word embeddings via deep compositional code learning. *ICLR 2018 Conference*.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale Information Network Embedding. *International World Wide Web Conference*.

Lei Tang and Huan Liu. 2009. Relational learning via latent social dimensions. *KDD '09: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Yu Wang, Yang Feng, Zhe Hong Ryan Berger, and Jiebo Luo. 2017. How Polarized Have We Become? A Multimodal Classification of Trump Followers and Clinton Followers. *SocInfo 2017: Social Informatics*, pages 440–456.

Junting Ye, Shuchu Han, Yifan Hu, Baris Coskun, Meizhu Liu, Hong Qin, and Steven Skiena. 2017. Nationality Classification Using Name Embeddings. *CIKM'17*, pages 1897–1906.

Zi Yin and Yuanyuan Shen. 2018. On the dimensionality of word embedding. *Advances in Neural Information Processing Systems 31 (NIPS 2018)*.

# A Surprisingly Effective Fix for Deep Latent Variable Modeling of Text

**Bohan Li**[*1], **Junxian He**[*1], **Graham Neubig**[1], **Taylor Berg-Kirkpatrick**[2], **Yiming Yang**[1]
[1]Language Technologies Institute, Carnegie Mellon University
[2]Department of Computer Science and Engineering, University of California San Diego
{bohanl1,junxianh,gneubig,yiming}@cs.cmu.edu, tberg@eng.ucsd.edu

## Abstract

When trained effectively, the Variational Autoencoder (VAE) is both a powerful language model and an effective representation learning framework. In practice, however, VAEs are trained with the *evidence lower bound* (ELBO) as a surrogate objective to the intractable marginal data likelihood. This approach to training yields unstable results, frequently leading to a disastrous local optimum known as *posterior collapse*. In this paper, we investigate a simple fix for posterior collapse which yields surprisingly effective results. The combination of two known heuristics, previously considered only in isolation, substantially improves held-out likelihood, reconstruction, and latent representation learning when compared with previous state-of-the-art methods. More interestingly, while our experiments demonstrate superiority on these principle evaluations, our method obtains *a worse ELBO*. We use these results to argue that the typical surrogate objective for VAEs may not be sufficient or necessarily appropriate for balancing the goals of representation learning and data distribution modeling.[1]

## 1 Introduction

Latent variable models attempt to model observed data $\mathbf{x}$ given latent variables $\mathbf{z}$, both for purposes of modeling data distributions $p(\mathbf{x})$ (e.g. language modeling) and learning representations $\mathbf{z}$ for a particular $\mathbf{x}$ (e.g. sentence embedding). Variational Autoencoders (VAEs) (Kingma and Welling, 2014) are a powerful framework for learning latent variable models using neural networks. The generative model of VAEs first samples a latent vector $\mathbf{z}$ from a prior $p(\mathbf{z})$, then applies a neural decoder $p(\mathbf{x}|\mathbf{z})$ to produce $\mathbf{x}$ conditioned on the latent code $\mathbf{z}$. VAEs are trained to maximize the evidence lower bound (ELBO) of the intractable log marginal likelihood:

$$\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})] - D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})),$$

where $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ represents an approximate posterior distribution (i.e. the *encoder* or *inference network*) and $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ is the generative distribution (i.e. the *decoder*).

However, modeling text with VAEs has proven to be challenging, and is an open research problem (Yang et al., 2017; Xu and Durrett, 2018; Kim et al., 2018; Dieng et al., 2018; He et al., 2019; Pelsmaeker and Aziz, 2019). When a strong decoder (e.g. the LSTM (Hochreiter and Schmidhuber, 1997)) is employed, training often falls into a trivial local optimum where the decoder learns to ignore the latent variable and the encoder fails to encode any information. This phenomenon is referred to as "posterior collapse" (Bowman et al., 2016). Existing efforts tackling this problem include re-weighting the KL loss (Bowman et al., 2016; Kingma et al., 2016; Liu et al., 2019), changing the model (Yang et al., 2017; Semeniuta et al., 2017; Xu and Durrett, 2018), and modifying the training procedure (He et al., 2019).

After conducting an empirical examination of the state-of-the-art methods (Section 2), we find that they have difficulty striking a good balance between language modeling and representation learning. In this paper, we present a practically effective combination of two simple heuristic techniques for improving VAE learning: (1) pretraining the inference network using an autoencoder objective and (2) thresholding the KL term in the ELBO objective (also known as "free bits" (Kingma et al., 2016)). The former technique initializes VAE training with an inference network that encodes useful information about $\mathbf{x}$, biasing learning away from local optima where $\mathbf{x}$ is ignored (i.e. posterior collapse). The latter tech-

[*]Equal contribution.
[1]Code is available at https://github.com/bohanli/vae-pretraining-encoder.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3603–3614,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

nique modifies the ELBO objective to prevent the KL term from dominating the encoder's role in reconstruction (Alemi et al., 2018), again biasing learning to avoid posterior collapse (Chen et al., 2017; Zhao et al., 2019).

In experiments we find that these two techniques do not perform well in isolation. However, when combined, they substantially outperform all baselines across various metrics that evaluate both language modeling and representation learning capabilities. Finally, we find that our method tends to achieve a superior language modeling results in terms of perplexity but an inferior ELBO value, and we use these results to argue that ELBO is suboptimal for language modeling even though it provides a formal lower bound on log marginal likelihood. Thus, we suggest that future research in this direction should be careful to monitor the gap between ELBO and log marginal likelihood, and reconsider using ELBO as the surrogate, especially for evaluation.

## 2 Analysis of Existing Methods

In this section, we first analyze and compare state-of-the-art solutions to posterior collapse[2] to obtain a holistic view of current progress and remaining challenges. Based on these observations, we then propose and evaluate a method that demonstrates substantially improved performance.

### 2.1 Evaluation

In this paper we focus on the standard VAE setting where both the prior $p(\mathbf{z})$ and the posterior $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ are factorized Gaussians.[3] We conduct preliminary experiments on the English Penn Treebank (PTB) (Marcus and Marcinkiewicz), a standard dataset for benchmarking VAE that has been used extensively in previous work (Bowman et al., 2016; Xu and Durrett, 2018; Liu et al., 2019). We use 32-dimension latent codes (full setup details can be found in Appendix A). We evaluate using the following metrics:

**Perplexity (PPL).** To compute perplexity, we first estimate the log marginal likelihood with 1000 importance weighted samples (Burda et al., 2016). Note that it is inappropriate to estimate PPL with ELBO directly since the gap between ELBO and log marginal likelihood might be large, particularly when the posterior does not collapse.

**Reconstruction loss (Recon).** Reconstruction loss is equivalent to the negative reconstruction term in ELBO: $-\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]$. It characterizes how well the latent code can be used to recover the input.

**Number of active units (AU, Burda et al. (2016)).** Active units correspond to the dimensions of $\mathbf{z}$ that covary with observations after the model is trained. More active units usually indicates richer latent representations (Burda et al., 2016). Specifically, a dimension is "active" when it is sensitive to the change in observations $\mathbf{x}$. Here we follow (Burda et al., 2016) and classify a latent dimension $z$ as active if $\mathrm{Cov}(\mathbf{x}, E_{z\sim q(z|\mathbf{x})}[z])) > 0.01$.

In addition to the metrics above, we include KL between prior and posterior approximation, as well as the negative ELBO, for reference – though we find that these quantities are only partially descriptive of model quality. In Section 3.2 we also evaluate the latent space of learned models with specialized metrics such as reconstruction BLEU and classification accuracy.

### 2.2 Baselines

We experiment with several state-of-the-art techniques to mitigate posterior collapse, including several KL reweighting methods and the recently proposed aggressive training (He et al., 2019).

**KL annealing (Bowman et al., 2016).** KL annealing might be the most common method for reweighting. During annealing, the weight of the KL term is increased from a small value to 1.0 in the beginning of training.

**Cyclic annealing (Liu et al., 2019).** Cyclic annealing is another reweighting scheme proposed recently. It changes the weight of the KL term in a cyclic fashion, rather than monotonically increasing the weight.[4]

**KL Thresholding / Free Bits (FB) (Kingma et al., 2016).** FB replaces the KL term in ELBO with a hinge loss term that maxes each component

[2] Here we focus on different training techniques for existing models, not alternatives which require changing the underlying model (Yang et al., 2017; Xu and Durrett, 2018).

[3] Pelsmaeker and Aziz (2019) thoroughly investigate using more complicated priors/posteriors (Rezende and Mohamed, 2015) but find only marginal improvements.

[4] We use the default cyclic annealing schedule from https://github.com/haofuml/cyclic_annealing in our codebase for a fair comparison.

of the original KL with a constant:

$$\sum_i \max[\lambda, D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(z_i|\mathbf{x})\|p(z_i))] \quad (1)$$

Here, $\lambda$ denotes the *target rate*, and $z_i$ denotes the $i^{th}$ dimension in the latent variable $\mathbf{z}$. Using the FB objective causes learning to give up trying to drive down KL for dimensions of $\mathbf{z}$ that are already beneath the target rate. Pelsmaeker and Aziz (2019) conduct a comprehensive experimental evaluation of related methods and conclude that the FB objective is able to achieve comparable or superior performance (in terms of both language modeling and reconstruction) in comparison with other top-performing methods, many of which are substantially more complex. We notice that Pelsmaeker and Aziz (2019) experiment with a slightly different version of FB where the threshold is applied to the entire KL term directly, rather than on each dimension's KL separately. We examine both versions here and refer to the single threshold version as "FBP" and the multiple threshold version (Eq. 1) as "FB". For both FB and FBP, we vary the target rate $\lambda$ and report the setting with the best validation PPL and the setting with the best balance between PPL and reconstruction loss.[5]

**Aggressive training (He et al., 2019).** He et al. (2019) observe that when posterior collapse occurs, the inference network often lags behind the generator during training. In contrast with the KL reweighting methods described above, He et al. (2019) propose an aggressive training schedule which iterates between multiple encoder update steps and one decoder update step to mitigate posterior collapse.[6]

**Autoencoder (AE).** We also include an autoencoder[7] as a reference for reconstruction loss.

We show the results of these baselines trained on PTB in Table 1, where we find that it is difficult to balance language modeling (PPL) and representation learning (Recon and AU) – the systems with relatively good reconstruction (FBP, $\lambda = 7$) or higher AU (FB baselines) have suboptimal PPL, and the best PPL is achieved with sacrifice of Recon and AU. Note that good PPL indicates good LM, but good recon and AU indicates good representation learning. Without both, we do not really have a strong probabilistic model of language that captures latent factors.

[5]It is subjective to judge "balance", thus we also report complete results for different target rates in Appendix C.

[6]We use the public code at https://github.com/jxhe/vae-lagging-encoder.

[7]Here, AE denotes the VAE trained without the KL term.

Table 1: Results on PTB test set for various baselines.

| Method | PPL↓ | Recon↓ | AU↑ | KL | -ELBO |
|---|---|---|---|---|---|
| AE | - | 70.36 | 32 | - | - |
| VAE | 101.39 | 101.27 | 0 | 0.00 | 101.27 |
| + anneal | 101.40 | 101.28 | 0 | 0.00 | 101.28 |
| + cyclic | 108.97 | 101.85 | 5 | 1.37 | 103.22 |
| + aggressive | 99.83 | 100.26 | 4 | 0.93 | 101.19 |
| + FBP ($\lambda = 7$) | 102.82 | 95.63 | 4 | 7.05 | 102.67 |
| + FBP ($\lambda = 3$) | 99.62 | 98.52 | 3 | 2.95 | 101.48 |
| + FB ($\lambda = 7$) | 104.06 | 98.97 | 32 | 6.74 | 105.72 |
| + FB ($\lambda = 3$) | 100.50 | 99.94 | 32 | 2.96 | 102.90 |

Table 2: Results on PTB test with encoder pretraining.

| Method | PPL↓ | Recon↓ | AU↑ | KL | -ELBO |
|---|---|---|---|---|---|
| AE | - | 70.36 | 32 | - | - |
| VAE | 101.39 | 101.27 | 0 | 0.00 | 101.27 |
| + pretrain | 102.26 | 101.46 | 0 | 0.00 | 101.46 |
| + pretrain + anneal | 97.74 | 99.67 | 2 | 1.01 | 100.68 |

We make two observations from the results in Table 1. First, reconstruction loss for an AE is substantially better than all VAE methods, which is intuitive since reconstruction is the only goal of training an AE. Second, models with high ELBO do not necessarily have good PPL (e.g. VAE+anneal); ELBO is not an ideal surrogate for evaluating language modeling performance.

### 2.3 Autoencoder-based Initialization

Based on the observations above we hypothesize that VAEs might benefit from initialization with an non-collapsed encoder, trained via an AE objective. Intuitively, if the encoder is providing useful information from the beginning of training, the decoder is more likely to make use of the latent code. In Table 2 we show the results of exploring this hypothesis on PTB. Even with encoder pretraining, we see that posterior collapse occurs immediately after beginning to update both encoder and decoder using the full ELBO objective. This indicates that the gradients of ELBO point towards a collapsed local optimum, even with biased initialization. When pretraining is combined with annealing, PPL improves substantially. However, the pretraining and anneal combination only has 2 active units and has small KL value – the latent representation is likely unsatisfactory. We speculate that this is because the annealing schedule eventually returns to the full ELBO objective which guides learning towards a (nearly) collapsed latent space. In the next section, we present an alternate approach using the KL thresholding / free bits method.

### 2.4 Our Method

In our proposed method, we initialize the inference network with an encoder that is pretrained using an autoencoder objective, as described above. Then, we train the VAE using the FB objective, also described above. We use the original FB which thresholds along each dimension.[8] Thus, we combine two approaches so far considered independently: pretraining and KL thresholding. In this way, however, the VAE would start with a large KL and is thus trained with the full ELBO objective in the initial stage, which is bad as observed in Section 2.3. To remedy this, we apply an annealing weight to Eq. 1. We use the simplest linear annealing schedule that increases the weight from 0 to 1 in the first 10 epochs for all of our experiments. This approach can be viewed in connection with KL annealing: we train with zero KL weight until convergence, then reset the decoder and start increasing the KL weight with the free bits objective. Next we conduct comprehensive experiments across different datasets to validate our method.

## 3 Experiments

In this section we work with three text datasets: PTB (Marcus and Marcinkiewicz), Yahoo (Yang et al., 2017), and a downsampled version of SNLI (Bowman et al., 2015). We demonstrate the effectiveness of our method through language modeling, text reconstruction, and quality of the learned latent space. Complete experimental setup details can be found in Appendix A.

### 3.1 Language Modeling

For language modeling, we only report the results for PTB and Yahoo due to the space limit, but include the SNLI results in Appendix C. Since we have already shown that FBP outperforms FB on PPL in Section 2 (without pretraining), here we only include FBP as a baseline. For both FBP and our method we vary the target rate and report the settings that achieve competitive validation PPL.[9]

As shown in Table 3, our method with different target rates is able to consistently outperform all the baselines in terms of PPL. Meanwhile, for representation learning, our method beats all the baselines by a large margin on reconstruction loss with all latent units active. We also note that our method is not very sensitive to the target rate $\lambda$.

Table 3: Language modeling results on PTB and Yahoo test set. We bold the lines that represent the best average of language modeling and reconstruction loss. Cyclic$^\star$ is from (Liu et al., 2019).

| Method | PPL↓ | Recon↓ | AU↑ | KL | -ELBO |
|---|---|---|---|---|---|
| **PTB** | | | | | |
| LSTM-LM | 100.47 | - | - | - | - |
| VAE | 101.39 | 101.27 | 0 | 0.00 | 101.27 |
| + anneal | 101.40 | 101.28 | 0 | 0.00 | 101.28 |
| + cyclic$^\star$ | - | 100.51 | - | 1.96 | 102.46 |
| + cyclic | 108.97 | 101.85 | 5 | 1.37 | 103.22 |
| + aggressive | 99.83 | 100.26 | 4 | 0.93 | 101.19 |
| + FBP ($\lambda = 3$) | 99.62 | 98.52 | 3 | 2.95 | 101.48 |
| + FBP ($\lambda = 2$) | 100.96 | 99.37 | 2 | 1.99 | 101.36 |
| Ours ($\lambda = 8$) | 98.07 | 92.60 | 32 | 10.95 | 103.56 |
| Ours ($\lambda = 6$) | **96.35** | **94.52** | **32** | 8.15 | 102.67 |
| Ours ($\lambda = 4$) | 96.17 | 96.91 | 32 | 4.99 | 101.90 |
| **Yahoo** | | | | | |
| LSTM-LM | 60.75 | - | - | - | - |
| VAE | 61.52 | 329.10 | 0 | 0.00 | 329.10 |
| + anneal | 61.21 | 328.80 | 0 | 0.00 | 328.80 |
| + cyclic | 66.93 | 333.80 | 4 | 2.83 | 336.63 |
| + aggressive | 59.77 | 322.70 | 15 | 5.70 | 328.40 |
| + FBP ($\lambda = 9$) | 62.59 | 322.91 | 6 | 9.08 | 331.99 |
| + FBP ($\lambda = 7$) | 62.76 | 324.66 | 5 | 7.03 | 331.69 |
| + FBP ($\lambda = 5$) | 62.78 | 326.26 | 3 | 5.07 | 331.32 |
| + FBP ($\lambda = 3$) | 62.88 | 328.13 | 2 | 3.06 | 331.19 |
| Ours ($\lambda = 6$) | 59.23 | 317.39 | 32 | 12.09 | 329.48 |
| Ours ($\lambda = 8$) | **59.51** | **315.31** | **32** | 15.02 | 330.33 |
| Ours ($\lambda = 9$) | 59.60 | 315.09 | 32 | 15.49 | 330.58 |

It is worth noticing that in some cases our method (e.g. $\lambda = 8$ in PTB and $\lambda = 8, 9$ in Yahoo) is able to outperform all the baselines but produce a bad ELBO (actually the worst on PTB). This suggests that ELBO might be a suboptimal surrogate for the log marginal likelihood sometimes, especially when KL is large, where the gap between ELBO and the marginal tends to be large as well.

### 3.2 Probing the Latent Space

We assess quality of learned latent space with SNLI through several metrics. For our method and FBP, we use the target rate where the best reconstruction loss is achieved while maintaining comparable PPL with aggressive training. [10]

**Text Reconstruction** We use greedy decoding and compute the BLEU score of the reconstructed sentence with the original one as the reference. The result is shown in Table 4. Unsurprisingly,

[8] Note that we do not pretrain the decoder and rather initialize it randomly. In our preliminary experiments, pretraining the decoder produced worse PPL compared with encoder-only pretraining.

[9] Results of all target rates can be found in Appendix C.

[10] Basically we try to tie PPL of different models and compare their latent space. Specifically, aggressive training has PPL 32.83, and PPL of the selected model for FBP and our method are 33.07 and 32.88, respectively.

the autoencoder achieves the highest BLEU score, meanwhile our method beats other VAE baselines.

Table 4: Reconstruction

| Method | BLEU |
|---|---|
| AE | 60.80 |
| VAE | 1.82 |
| + anneal | 2.51 |
| + cyclic | 4.39 |
| + aggressive | 2.95 |
| + FBP ($\lambda = 7$) | 8.07 |
| Ours ($\lambda = 4$) | **8.62** |

Table 5: Smoothness

| Method | PCC |
|---|---|
| AE | 0.620 |
| VAE | 0.039 |
| + anneal | 0.009 |
| + cyclic | 0.482 |
| + aggressive | 0.209 |
| + FBP ($\lambda = 7$) | 0.242 |
| Ours ($\lambda = 4$) | **0.683** |

**Smoothness of Latent Space** A major difference between VAE and AE is that VAE can learn a smooth latent space through the regularization from the Gaussian prior. In a smooth latent space, latent codes of similar sentences should be close to each other and vice versa. Therefore, we randomly sample 100k sentence pairs and evaluate the Pearson Correlation Coefficient (PCC) between the $\ell_2$ distances of latent codes and edit word distances. As shown in Table 5, our method achieves a much higher PCC compared to the baselines.

Zhao et al. (2018) argue that a smooth latent space is beneficial for reconstructing noisy inputs. We follow their experiments and introduce noise to the input by randomly swapping words $k$ times. As shown in Table 6, while AE achieves the best reconstruction when the noise is small ($k = 1$), its reconstruction deteriorates dramatically when $k > 1$, which suggests AE fails to learn a smooth latent space. In contrast, our method outperforms all the baselines by a large margin when $k > 1$.

**Interpolation** As illustrated in (Bowman et al., 2016), linear interpolation between latent variables is an intuitive way to qualitatively evaluate the smoothness of the latent space. We sample two latent codes $\mathbf{z}_0$ and $\mathbf{z}_1$ from the prior $p(\mathbf{z})$ (Table 7) and do linear interpolation between the two with evenly divided intervals.[11]. For each interpolated point, we decode it greedily. Our method is able to generate grammatically plausible and semantically consistent interpolation in both cases.

### 3.3 Classification

To further evaluate the quality of the latent codes, we train a Gaussian mixture model (for unsupervised clustering) or a one-layer linear classifier (for supervised classification) on the pretrained latent codes. We work with a downsampled version of Yelp sentiment dataset collected by Shen et al. (2017). We vary the number of labeled data[12] and the results are shown in Table 8. For FBP and our method, the target rate is selected in terms of the average validation accuracy. Our method consistently yields the best results on all settings – remarkably, its performance with only 100 labeled samples already surpasses others with 10k labels.

[11] Interpolation examples from posterior samples $q_\phi(\mathbf{z}|\mathbf{x})$ are provided in Appendix 13.

Table 6: Noisy reconstruction loss (↓) on SNLI. #swap denotes the number of word swaps.

| #swap | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| AE | **26.05** | 40.46 | 52.77 | 63.07 |
| VAE | 33.10 | 33.11 | 33.11 | 33.12 |
| + anneal | 32.20 | 32.65 | 33.12 | 33.39 |
| + cyclic | 31.83 | 32.87 | 33.73 | 34.38 |
| + aggressive | 31.78 | 31.99 | 32.21 | 32.32 |
| + FBP ($\lambda = 7$) | 29.93 | 32.59 | 34.90 | 36.77 |
| Ours ($\lambda = 4$) | 27.92 | **29.12** | **30.03** | **30.85** |

Table 7: Interpolation between prior samples on SNLI.

| |
|---|
| **AE** |
| people on their ground and they sit towards each other . |
| girls riding their cellphones and other people sit near papers . |
| girls riding in an area while not talk to dishes . |
| person riding in an area while carrying bags and papers . |
| someone riding in an office , selling a button . |
| three kid riding in <unk> signs a brick advertisement area . |
| **Ours** |
| a man with a cane is walking down the street . |
| a man with a cane is walking down the street . |
| a man in a blue shirt is eating food . |
| people are eating food . |
| people walk in a city . |
| people are outside in a city . |

Table 8: Accuracy on Yelp of unsupervised and supervised classification. Evaluated via accuracy. #labeled denotes the number of labeled example during training.

| #labeled | 0 | 100 | 500 | 1k | 2k | 10k |
|---|---|---|---|---|---|---|
| AE | 52.0 | 78.4 | 81.1 | 83.5 | 83.8 | 83.8 |
| VAE | 56.5 | 58.9 | 62.3 | 62.5 | 62.9 | 64.0 |
| + anneal | 56.1 | 58.7 | 60.6 | 61.5 | 61.4 | 64.1 |
| + cyclic | 59.3 | 78.1 | 79.8 | 81.1 | 81.7 | 83.1 |
| + aggressive | 63.7 | 65.6 | 68.6 | 72.1 | 76.7 | 79.4 |
| + FBP ($\lambda = 9$) | 60.7 | 73.3 | 75.0 | 76.1 | 77.6 | 79.5 |
| Ours ($\lambda = 6$) | **67.2** | **83.8** | **88.3** | **89.1** | **89.5** | **89.5** |

## 4 Conclusion

In this paper, we propose a simple training fix to tackle posterior collapse in VAEs. Extensive experiments demonstrate the effectiveness of our method on both representation learning and language modeling.

[12] Specifically, we train a VAE model on the Yelp dataset to obtain the latent codes, then we we use the latent codes of labeled data to train the classifier.

## References


Alexander Alemi, Ben Poole, Ian Fischer, Joshua Dillon, Rif A Saurous, and Kevin Murphy. 2018. Fixing a broken ELBO. In *Proceedings of ICML*.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of EMNLP*.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of CoNLL*.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. 2016. Importance weighted autoencoders. In *Proceedings of ICLR*.

Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2017. Variational lossy autoencoder. In *Proceedings of ICLR*.

Adji B Dieng, Yoon Kim, Alexander M Rush, and David M Blei. 2018. Avoiding latent variable collapse with generative skip models. In *Proceedings of ICML workshop on Theoretical Foundations and Applications of Deep Generative Models*.

Junxian He, Daniel Spokoyny, Graham Neubig, and Taylor Berg-Kirkpatrick. 2019. Lagging inference networks and posterior collapse in variational autoencoders. In *Proceedings of ICLR*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Yoon Kim, Sam Wiseman, Andrew Miller, David Sontag, and Alexander Rush. 2018. Semi-amortized variational autoencoders. In *Proceedings of ICML*.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *Proceedings of ICLR*.

Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. 2016. Improved variational inference with inverse autoregressive flow. In *Proceedings of NeurIPS*.

Xiaodong Liu, Jianfeng Gao, Asli Celikyilmaz, Lawrence Carin, et al. 2019. Cyclical annealing schedule: A simple approach to mitigating KL vanishing. In *Proceedings of NAACL*.

Mitchell P Marcus and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2).

Tom Pelsmaeker and Wilker Aziz. 2019. Effective estimation of deep generative language models. *arXiv preprint arXiv:1904.08194*.

Danilo Rezende and Shakir Mohamed. 2015. Variational inference with normalizing flows. In *Proceedings of ICML*.

Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. 2017. A hybrid convolutional variational autoencoder for text generation. In *Proceedings of EMNLP*.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Proceedings of NeurIPS*.

Jiacheng Xu and Greg Durrett. 2018. Spherical latent spaces for stable variational autoencoders. In *Proceedings of EMNLP*.

Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. 2017. Improved variational autoencoders for text modeling using dilated convolutions. In *Proceedings of ICML*.

Junbo Zhao, Yoon Kim, Kelly Zhang, Alexander Rush, and Yann LeCun. 2018. Adversarially regularized autoencoders. In *Proceedings of ICML*.

Shengjia Zhao, Jiaming Song, and Stefano Ermon. 2019. InfoVAE: Information maximizing variational autoencoders. In *Proceedings of AAAI*.

## A Details of Experimental Setup

For SNLI, we randomly downsample a subset of it, which contains 100K/10K/10K sentences as training/validation/test. For the Yelp sentiment dataset, we also randomly downsample 100K/10K/10K sentences for training/validation/test, respectively. We use an one-layer LSTM for both the encoder and decoder and a latent vector size of 32.

We follow Kim et al. (2018); He et al. (2019) and use a single-layer LSTM the encoder and decoder in all of our experiments. The sizes of word embeddings and hidden states for different datasets are given in Table 9. We initialize the LSTM parameters with a uniform distribution $\mathcal{U}(-0.01, 0.01)$, and embeddings with another uniform distribution $\mathcal{U}(-0.1, 0.1)$.

Just as in Kim et al. (2018); He et al. (2019), for the decoder, we use dropout of 0.5 on both the input word embedding and the last dense before logits. During training, we use the SGD optimizer wihout momentum. Initialized with 0.5, the learning rate is decayed by $\times 0.5$ with a patience of 2 if the validation loss has not improved in the past 2 epochs. The maximum number of epochs is 100, but the training will stop early after 5 learning rate decays. For our VAE + anneal baseline, we use the simplest linear annealing schedule that increases the weight from 0 to 1 in the first 10 epochs, just the same as in our method (stated in Section 2.4).

Table 9: The sizes of word embeddings and hidden states for PTB, SNLI and Yahoo.

| | PTB | SNLI | Yahoo |
|---|---|---|---|
| Word Embedding Size | 256 | 128 | 512 |
| Hidden Size of Encoder | 256 | 512 | 1024 |
| Hidden Size of Decoder | 256 | 512 | 1024 |

## B Copying Behaviour

We check to make sure that our model is not simply learning to copy sentences from the training set. We test for copying behaviour on the PTB dataset. Specifically, we sampled 300 sentences from the prior and retrieved their nearest neighbors in the training set. The average edit distance between the samples and their nearest neighbors from our method is 14.11 (the average training sentence length is 22.10), versus 13.68 from the collapsed VAE. This means that there is no obvious copying behaviour when KL grows in our method.

## C Additional Results of Language Modeling

In Table 10, we provide a more detailed version of the language modeling results on PTB, SNLI and Yahoo. The full results of all the target rate trial for both FBP and our method are included. In addition to the metrics in Table 3, we also report NLL, Mutual Information $I_q$ between $\mathbf{z}$ and $\mathbf{x}$ under $q_{\phi}(\mathbf{z}|\mathbf{x})$ (MI)[13], and the perplexity computed by ELBO (ELBO PPL).

[13]For the estimation of $I_q$, we follow the same method as in He et al. (2019)

Table 10: Additional results of language modeling on PTB and SNLI.

| Dataset | Method | NLL↓ | PPL↓ | Recon↓ | MI↑ | AU↑ | -ELBO | ELBO PPL | KL |
|---|---|---|---|---|---|---|---|---|---|
| PTB | LSTM-LM | 101.04 | 100.47 | - | - | - | - | - | - |
| | AE | | | 70.36 | 8.22 | 32 | - | - | - |
| | VAE | 101.23 | 101.39 | 101.27 | 0.01 | 0 | 101.27 | 101.58 | 0.00 |
| | + anneal | 101.24 | 101.40 | 101.28 | 0.00 | 0 | 101.28 | 101.62 | 0.00 |
| | + cyclic (reported) | - | - | 100.51 | - | - | 102.46 | 107.25 | 1.96 |
| | + cyclic | 102.81 | 108.97 | 101.85 | 1.27 | 5 | 103.22 | 111.03 | 1.37 |
| | + cyclic (SGD) | 102.06 | 105.28 | 102.14 | 0.00 | 0 | 102.14 | 105.67 | 0.00 |
| | + aggressive | 100.89 | 99.83 | 100.26 | 0.83 | 4 | 101.19 | 101.17 | 0.93 |
| | + pretrained enc | 101.42 | 102.26 | 101.46 | 0.0 | 0 | 101.46 | 102.45 | 0.00 |
| | + anneal enc | 100.43 | 97.74 | 99.67 | 0.97 | 2 | 100.68 | 98.88 | 1.01 |
| | + FBP ($\lambda = 9$) | 101.95 | 104.73 | 94.66 | 7.22 | 6 | 103.59 | 112.88 | 8.93 |
| | + FBP ($\lambda = 8$) | 101.64 | 103.30 | 95.39 | 7.11 | 6 | 103.43 | 112.09 | 8.04 |
| | + FBP ($\lambda = 7$) | 101.54 | 102.82 | 95.63 | 6.52 | 4 | 102.67 | 108.26 | 7.05 |
| | + FBP ($\lambda = 6$) | 101.68 | 103.47 | 96.66 | 5.76 | 10 | 102.67 | 108.27 | 6.01 |
| | + FBP ($\lambda = 5$) | 101.24 | 101.42 | 97.12 | 4.80 | 4 | 102.21 | 106.01 | 5.10 |
| | + FBP ($\lambda = 4$) | 101.49 | 102.58 | 97.84 | 3.86 | 4 | 101.86 | 104.31 | 4.01 |
| | + FBP ($\lambda = 3$) | 100.85 | 99.62 | 98.52 | 2.86 | 3 | 101.48 | 102.52 | 2.95 |
| | + FBP ($\lambda = 2$) | 101.14 | 100.96 | 99.37 | 1.91 | 2 | 101.36 | 101.98 | 1.99 |
| | IWAE ($k = 10$) | 100.86 | 99.69 | 100.89 | 0.05 | 0 | 100.89 | 99.81 | 0.00 |
| | + pretrained enc | 100.92 | 99.95 | 100.96 | 0.01 | 0 | 100.96 | 100.15 | 0.00 |
| | Ours ($\lambda = 9$) | 101.09 | 100.71 | 92.00 | 7.78 | 32 | 104.43 | 117.32 | 12.44 |
| | Ours ($\lambda = 8$) | 100.51 | 98.07 | 92.60 | 7.49 | 32 | 103.56 | 112.72 | 10.95 |
| | Ours ($\lambda = 7$) | 101.06 | 100.60 | 93.25 | 7.46 | 32 | 103.74 | 113.68 | 10.49 |
| | Ours ($\lambda = 6$) | 100.12 | 96.35 | 94.52 | 6.30 | 32 | 102.67 | 108.25 | 8.15 |
| | Ours ($\lambda = 5$) | 100.23 | 96.86 | 95.87 | 5.31 | 32 | 102.41 | 106.97 | 6.54 |
| | Ours ($\lambda = 4$) | 100.08 | 96.17 | 96.91 | 4.08 | 32 | 101.90 | 104.51 | 4.99 |
| | Ours ($\lambda = 3$) | 100.21 | 96.75 | 97.71 | 3.19 | 32 | 101.56 | 102.90 | 3.85 |
| | Ours ($\lambda = 2$) | 100.41 | 97.65 | 98.73 | 2.21 | 32 | 101.38 | 102.07 | 2.65 |
| SNLI | LSTM-LM | 32.97 | 21.44 | - | - | - | - | - | - |
| | AE | - | - | 8.68 | 9.18 | 32 | - | - | - |
| | VAE | 33.09 | 21.67 | 33.08 | 0.03 | 1 | 33.12 | 21.73 | 0.04 |
| | + anneal | 33.01 | 21.50 | 31.66 | 1.45 | 2 | 33.07 | 21.63 | 1.42 |
| | + cyclic | 34.04 | 23.67 | 30.69 | 3.60 | 5 | 34.32 | 24.29 | 3.63 |
| | + cyclic (SGD) | 33.07 | 21.62 | 30.89 | 2.33 | 4 | 33.25 | 21.99 | 2.36 |
| | + aggressive | 32.83 | 21.16 | 31.53 | 1.38 | 5 | 32.95 | 21.39 | 1.42 |
| | + FBP ($\lambda = 9$) | 33.28 | 22.05 | 25.26 | 8.06 | 6 | 34.25 | 24.13 | 8.99 |
| | + FBP ($\lambda = 8$) | 33.26 | 22.02 | 26.07 | 7.35 | 7 | 34.08 | 23.75 | 8.01 |
| | + FBP ($\lambda = 7$) | 33.07 | 21.62 | 26.65 | 6.76 | 6 | 33.78 | 23.11 | 7.14 |
| | + FBP ($\lambda = 6$) | 33.09 | 21.68 | 27.54 | 5.95 | 5 | 33.59 | 22.71 | 6.06 |
| | + FBP ($\lambda = 5$) | 33.04 | 21.58 | 28.38 | 4.95 | 6 | 33.49 | 22.48 | 5.10 |
| | + FBP ($\lambda = 4$) | 33.04 | 21.57 | 29.25 | 4.06 | 4 | 33.36 | 22.22 | 4.11 |
| | + FBP ($\lambda = 3$) | 33.04 | 21.56 | 30.19 | 3.00 | 4 | 33.31 | 22.11 | 3.11 |
| | + FBP ($\lambda = 2$) | 32.99 | 21.46 | 31.04 | 2.11 | 3 | 33.16 | 21.80 | 2.12 |
| | Ours ($\lambda = 9$) | 33.42 | 22.33 | 22.30 | 8.80 | 32 | 35.70 | 27.61 | 13.40 |
| | Ours ($\lambda = 8$) | 33.47 | 22.45 | 22.65 | 8.76 | 32 | 35.62 | 27.40 | 12.96 |
| | Ours ($\lambda = 7$) | 33.25 | 22.00 | 23.36 | 8.48 | 32 | 35.11 | 26.14 | 11.75 |
| | Ours ($\lambda = 6$) | 33.17 | 21.84 | 24.06 | 8.24 | 32 | 34.83 | 25.47 | 10.77 |
| | Ours ($\lambda = 5$) | 33.07 | 21.64 | 24.94 | 7.71 | 32 | 34.47 | 24.63 | 9.53 |
| | Ours ($\lambda = 4$) | 32.88 | 21.24 | 26.52 | 6.65 | 32 | 34.11 | 23.83 | 7.60 |
| | Ours ($\lambda = 3$) | 32.87 | 21.23 | 28.02 | 5.25 | 32 | 33.87 | 23.31 | 5.86 |
| | Ours ($\lambda = 2$) | 32.79 | 21.07 | 29.75 | 3.32 | 32 | 33.42 | 22.35 | 3.67 |

Table 11: Additional results of language modeling on Yahoo.

| Dataset | Method | NLL↓ | PPL↓ | Recon↓ | MI↑ | AU↑ | -ELBO | ELBO PPL | KL |
|---|---|---|---|---|---|---|---|---|---|
| Yahoo | LSTM-LM | 328.00 | 60.75 | - | - | - | - | - | - |
| | AE | - | - | 278.76 | 9.26 | 32 | - | - | - |
| | VAE | 329.00 | 61.52 | 329.10 | 0.00 | 0 | 329.10 | 61.59 | 0.0 |
| | + anneal | 328.60 | 61.21 | 328.80 | 0.00 | 0 | 328.80 | 61.36 | 0.0 |
| | + cyclic | 335.74 | 66.93 | 333.80 | 2.77 | 4 | 336.63 | 67.69 | 2.83 |
| | + cyclic (SGD) | 332.48 | 64.26 | 332.65 | 0.00 | 1 | 332.68 | 64.42 | 0.03 |
| | + aggressive | 326.70 | 59.77 | 322.70 | 2.9 | 15 | 328.40 | 61.06 | 5.70 |
| | + FBP ($\lambda = 9$) | 330.38 | 62.59 | 322.91 | 8.21 | 6 | 331.99 | 63.86 | 9.08 |
| | + FBP ($\lambda = 8$) | 330.80 | 62.92 | 324.03 | 7.54 | 6 | 332.09 | 63.94 | 8.05 |
| | + FBP ($\lambda = 7$) | 330.60 | 62.76 | 324.66 | 6.76 | 5 | 331.69 | 63.62 | 7.03 |
| | + FBP ($\lambda = 6$) | 331.05 | 63.12 | 325.87 | 5.94 | 5 | 332.01 | 63.88 | 6.13 |
| | + FBP ($\lambda = 5$) | 330.62 | 62.78 | 326.26 | 5.00 | 3 | 331.32 | 63.33 | 5.07 |
| | + FBP ($\lambda = 4$) | 331.06 | 63.13 | 327.55 | 4.00 | 3 | 331.66 | 63.60 | 4.11 |
| | + FBP ($\lambda = 3$) | 330.75 | 62.88 | 328.13 | 2.99 | 2 | 331.19 | 63.23 | 3.06 |
| | + FBP ($\lambda = 2$) | 331.30 | 63.32 | 329.60 | 1.98 | 1 | 331.63 | 63.58 | 2.04 |
| | Ours ($\lambda = 2$) | 326.34 | 59.50 | 322.55 | 5.35 | 32 | 328.51 | 61.14 | 5.96 |
| | Ours ($\lambda = 3$) | 326.12 | 59.34 | 321.29 | 6.41 | 32 | 328.73 | 61.31 | 7.44 |
| | Ours ($\lambda = 4$) | 326.01 | 59.26 | 319.49 | 7.58 | 32 | 329.03 | 61.54 | 9.54 |
| | Ours ($\lambda = 5$) | 326.04 | 59.28 | 318.55 | 8.08 | 32 | 329.31 | 61.76 | 10.76 |
| | Ours ($\lambda = 6$) | 325.97 | 59.23 | 317.39 | 8.51 | 32 | 329.48 | 61.89 | 12.09 |
| | Ours ($\lambda = 7$) | 326.08 | 59.31 | 316.42 | 8.78 | 32 | 329.76 | 62.10 | 13.34 |
| | Ours ($\lambda = 8$) | 326.35 | 59.51 | 315.31 | 8.99 | 32 | 330.33 | 62.55 | 15.02 |
| | Ours ($\lambda = 9$) | 326.47 | 59.60 | 315.09 | 9.03 | 32 | 330.58 | 62.75 | 15.49 |

## D Additional Qualitative Examples

### D.1 Interpolation between Prior Samples

We randomly sample 10 pair of source and target latent code from the standard Gaussian prior and do linear interpolation. For the sampled and interpolated latent codes, we do greedy decoding. The results are shown in Table 12.

### D.2 Interpolation between Posterior Samples

We randomly sample 10 pairs of source and target input sentences from the test set of SNLI. For each input sentence, we randomly sample a latent code from the approximated posterior $q(\mathbf{z}|\mathbf{x})$. Then we linearly interpolate between each pair of sampled source/target latent code. For the sampled and interpolated latent codes, we do greedy decoding. The results are shown in Table 13.

Table 12: Interpolation between prior samples on SNLI

| EXAMPLE 1 | EXAMPLE 6 |
|---|---|
| the kids are playing hide and seek in the classroom | a man is jumping off a rock into the air . |
| the girl is about to play the drums | a man is sitting on a bench with a red umbrella . |
| the girl is about to play in the sandbox | a man is sitting on a bench with a red umbrella . |
| the girl is about to play in the sandbox | a man is sitting on a bench with a red umbrella . |
| the girls are watching tv in the classroom . | a woman is sitting on a bench in front of a building . |
| the girl is eating cake in the kitchen . | a young man is sitting on a bench outside . |
| the women are watching tv in the bar . | a young man is taking pictures of a building . |
| the women are eating lunch . | a large group of people are taking pictures of a building . |
| the women are eating lunch . | a large group of people are taking pictures in the street . |
| the women are eating dinner . | a large group of people are taking pictures in the street . |
| a woman is eating in a restaurant . | a large group of people are taking pictures in the street . |
| EXAMPLE 2 | EXAMPLE 7 |
| a man with a cane is walking down the street . | two men are swimming in the ocean . |
| a man with a cane is walking down the street . | two men are swimming in the ocean . |
| a man with a cane is walking down the street . | two men are on the beach . |
| a man with a cane is walking down a sidewalk . | two people are at the beach . |
| a man in a blue shirt is eating food . | two people are at the beach . |
| man in a hat and jeans is walking down a sidewalk . | the man is at the park . |
| people are eating food . | a man is at the beach . |
| people walk through a city street . | a man is at the beach . |
| people walk in a city . | a man is at the beach . |
| people walk in a city . | a man is at the beach . |
| people are outside in a city . | a man is taking pictures of the ocean . |
| EXAMPLE 3 | EXAMPLE 8 |
| the man is going to the bathroom . | two people in bathing suits are in a park . |
| the man is going to the bathroom . | two people in bathing suits are in a park . |
| the man is going to the bathroom . | two people in blue shirts are in a field . |
| the man is going to the bathroom . | two people in blue shirts are in a field . |
| the man is playing music in the living room . | two people in a field are playing soccer . |
| the man is playing with his dog . | two people in a field of flowers . |
| the man is playing with a cat . | two people are at a beach . |
| the man is playing with a ball . | a man in a blue shirt is looking at a camera . |
| the man is playing with a ball . | a man in a blue shirt is playing basketball . |
| the man is playing with a ball . | a man in a blue shirt is playing basketball . |
| the man is playing with a ball . | a man sits at a carnival . |
| EXAMPLE 4 | EXAMPLE 9 |
| a person is about to get a picture taken | two women are outside . |
| a person is about to get a picture taken | two women are outside . |
| a person is about to get a picture taken | two women are outside . |
| a person is waiting for a friend to come to work | two women are sitting outside . |
| a person is waiting for a friend to come | a couple is sitting outside . |
| a person is trying to find a speech | a couple is sitting outside . |
| the people are playing monopoly | a couple is sitting in a car . |
| there are people playing monopoly | a man is reading a newspaper in a park . |
| there are people performing | a man is reading a book in a park . |
| there are people performing surgery | a man is carrying a bag of food . |
| there are no people in this picture . | a man is carrying a bag of food . |
| EXAMPLE 5 | EXAMPLE 10 |
| a girl sits on a bench in front of a large crowd . | the man is climbing the mountain . |
| a girl sits on a bench in front of a large crowd . | the man is making a noise . |
| a man sits on a bench in front of a large crowd . | the man is making a noise . |
| a man sits on a bench in front of a large crowd . | the man is making a noise . |
| a woman wearing a blue shirt is walking on the beach . | the people are enjoying the sunshine . |
| a woman wearing a blue shirt is walking on the beach . | the people are watching a movie . |
| the woman is wearing a blue shirt . | the women are eating a meal . |
| the woman is wearing a blue shirt . | the men are watching a movie . |
| the woman is wearing a blue shirt . | the men are watching a movie . |
| the woman is wearing a blue shirt . | two women sit in a circle together . |
| the woman is moving her legs . | two women sit in a circle together . |

Table 13: Interpolation between posterior samples on SNLI.

| | EXAMPLE 1 | EXAMPLE 6 |
|---|---|---|
| SOURCE INPUT | a child is eating with utensils . | a middle-aged man with long , curly red-hair wearing a dark vest , shirt and pants is holding a microphone in front of a black backdrop . |
| TARGET INPUT | a youth wearing a blue and red jersey and yellow helmet is crouching in a football position | people are doing <unk> |
| POSTERIOR-SAMPLED SOURCE | a little girl is eating at a restaurant . | the two men are wearing jeans and a blue shirt , and a woman are holding a rope . |
| INTERPOLATION | a little girl is eating at a restaurant . | the two men are wearing jeans and a blue shirt . |
| | a little girl is eating at a table . | the two men are wearing jeans and a blue shirt . |
| | a little girl is playing with a toy . | the two men are wearing white shirts and are playing a sport . |
| | a little girl is playing with a ball . | the people are trying to find a cure for cancer . |
| | a little girl is wearing a pink shirt . | the people are playing soccer |
| | a little girl is wearing a pink shirt and holding a popsicle . | two people are playing baseball |
| | a man is wearing a blue shirt and a hat . | two people are playing baseball |
| | a man is wearing a blue shirt and a hat . | two people are playing baseball |
| | a man is wearing a black shirt and black pants | people are playing baseball |
| POSTERIOR-SAMPLED TARGET | a man wearing a black shirt and black pants is standing on a sidewalk | people are playing baseball |
| | EXAMPLE 2 | EXAMPLE 7 |
| SOURCE INPUT | the men are feeling competetive . | both men are wearing similar colors . |
| TARGET INPUT | a young woman is sitting in a field . | a huge animal surrounded |
| POSTERIOR-SAMPLED SOURCE | a man is climbing a tree . | a man is cutting a cake . |
| INTERPOLATION | a man is climbing a tree . | a man is cutting a cake . |
| | a man is eating a pizza . | a man is painting a portrait of a woman . |
| | a man is wearing a blue shirt . | a man is painting a portrait of a woman . |
| | a man is wearing a blue shirt . | a man with a beard is playing guitar . |
| | a man is wearing a blue shirt . | a young boy with a blue shirt . |
| | a man is standing in front of a large crowd . | a little girl with a pink shirt . |
| | a woman is standing in front of a large crowd . | a tall human with a shirt |
| | a woman is standing in front of a large crowd . | a tall human with a shirt |
| | a woman is standing in front of a large crowd . | a tall human with a shirt |
| POSTERIOR-SAMPLED TARGET | a woman is standing in front of a large crowd . | a tall human looking |
| | EXAMPLE 3 | EXAMPLE 8 |
| SOURCE INPUT | the animals are near the water . | a crowded city street in asia . |
| TARGET INPUT | a truck is going to tow an illegally parked white volkswagon . | a guy is in front of a business . a tree with flowers is in front . |
| POSTERIOR-SAMPLED SOURCE | two boys are at a beach . | a bunch of people are in a store . |
| INTERPOLATION | two boys are at a beach . | a bunch of people are in a park . |
| | two men are looking at a man in a wheelchair . | a few people are in a park . |
| | the children are at the beach . | a few people are in a park . |
| | the children are looking at the sky . | a few people are in a park . |
| | a woman is looking at a man in a wheelchair . | a few people are in front of a building . |
| | a woman is looking at a man in a wheelchair . | a lady is in a store with a man . |
| | a woman is looking at a map . | a lady in a blue shirt is looking at a man in a blue shirt . |
| | a woman is waiting for a bus to come out of the road . | a lady in a blue shirt is looking at a man in a blue shirt . |
| | a woman is waiting for a bus to come out of the city . | a lady in a blue shirt is looking at a man in a blue shirt . |
| POSTERIOR-SAMPLED TARGET | a woman is waiting for a bus . | a lady in a blue shirt and black pants is playing in a fountain with a small child in the background . |
| | EXAMPLE 4 | EXAMPLE 9 |
| SOURCE INPUT | one young child in a swimsuit jumping off a blue inflatable slide with water . | a woman with 5 small children . |
| TARGET INPUT | a girl swings from a rope swing in front . | a man works on <unk> a circuit as he monitors the progress on a tablet device . |
| POSTERIOR-SAMPLED SOURCE | a young man in a blue shirt and black pants is standing by a large rock formation . | man with blue shirt and blue shirt is playing basketball . |
| INTERPOLATION | a young girl in a pink shirt and blue shorts is jumping into a pool . | man with blue shirt and blue shirt is playing basketball . |
| | a woman in a pink shirt and black shorts is jumping into a pool . | man with blue shirt and blue shirt is playing basketball . |
| | a woman in a pink shirt and black shorts is playing a game of soccer . | man in blue shirt with a blue shirt on his head . |
| | a woman in a pink shirt and black shorts is playing a game of soccer . | a man with a hat is playing with a ball . |
| | a woman with a red shirt and a black shirt is sitting in a chair . | a man in a blue shirt is looking at a plant . |
| | a woman with a red shirt and a black shirt is sitting in a chair . | a man in a blue shirt is sitting on a bench with a shovel . |
| | a woman with a red shirt and a black shirt is sitting in a chair . | a man in a blue shirt is sitting on a bench with a shovel . |
| | a woman with a red shirt and a black shirt is looking at a camera . | a man is on a skateboard in front of a building . |
| | a woman watches a man play a game of soccer . | a man is on a skateboard in front of a building with graffiti on it . |
| POSTERIOR-SAMPLED TARGET | a woman watches a man play a guitar in front of a crowd . | a man works on a project on a stove . |
| | EXAMPLE 5 | EXAMPLE 10 |
| SOURCE INPUT | a girl with glasses next red white and blue flags . | a man walking across a bridge near a steak restaurant . |
| TARGET INPUT | three greyhounds are taking a walk with their owner . | a woman in a white shirt and shorts is playing a red guitar . |
| POSTERIOR-SAMPLED SOURCE | the girl is drinking milk with the camera . | people in a park |
| INTERPOLATION | the girl is drinking milk with the camera . | a woman in a blue shirt is eating a sandwich . |
| | the girl is drinking milk with her hands . | a woman in a blue shirt is playing a game of tennis . |
| | the girl is drinking water with a bucket . | a woman is playing a game of tennis . |
| | the girl is using a camera . | a woman is playing a game of tennis . |
| | two girls are outside with a blue umbrella . | a man is playing a guitar . |
| | two girls are outside with a blue umbrella . | a man is playing a guitar . |
| | two girls are outside with a dog . | a man is playing a guitar . |
| | two girls are taking a picture of a tree . | a man is playing a guitar . |
| | two guys are on a bench . | a man is playing a guitar . |
| POSTERIOR-SAMPLED TARGET | two guys are on a boat . | a man is playing a guitar . |

# SciBERT: A Pretrained Language Model for Scientific Text

**Iz Beltagy** **Kyle Lo** **Arman Cohan**
Allen Institute for Artificial Intelligence, Seattle, WA, USA
{beltagy,kylel,armanc}@allenai.org

## Abstract

Obtaining large-scale annotated data for NLP tasks in the scientific domain is challenging and expensive. We release SciBERT, a pretrained language model based on BERT (Devlin et al., 2019) to address the lack of high-quality, large-scale labeled scientific data. SciBERT leverages unsupervised pretraining on a large multi-domain corpus of scientific publications to improve performance on downstream scientific NLP tasks. We evaluate on a suite of tasks including sequence tagging, sentence classification and dependency parsing, with datasets from a variety of scientific domains. We demonstrate statistically significant improvements over BERT and achieve new state-of-the-art results on several of these tasks. The code and pretrained models are available at https://github.com/allenai/scibert/.

## 1 Introduction

The exponential increase in the volume of scientific publications in the past decades has made NLP an essential tool for large-scale knowledge extraction and machine reading of these documents. Recent progress in NLP has been driven by the adoption of deep neural models, but training such models often requires large amounts of labeled data. In general domains, large-scale training data is often possible to obtain through crowdsourcing, but in scientific domains, annotated data is difficult and expensive to collect due to the expertise required for quality annotation.

As shown through ELMo (Peters et al., 2018), GPT (Radford et al., 2018) and BERT (Devlin et al., 2019), unsupervised pretraining of language models on large corpora significantly improves performance on many NLP tasks. These models return contextualized embeddings for each token which can be passed into minimal task-specific neural architectures. Leveraging the success of unsupervised pretraining has become especially important especially when task-specific annotations are difficult to obtain, like in scientific NLP. Yet while both BERT and ELMo have released pretrained models, they are still trained on general domain corpora such as news articles and Wikipedia.

In this work, we make the following contributions:

(*i*) We release SciBERT, a new resource demonstrated to improve performance on a range of NLP tasks in the scientific domain. SciBERT is a pretrained language model based on BERT but trained on a large corpus of scientific text.

(*ii*) We perform extensive experimentation to investigate the performance of finetuning versus task-specific architectures atop frozen embeddings, and the effect of having an in-domain vocabulary.

(*iii*) We evaluate SciBERT on a suite of tasks in the scientific domain, and achieve new state-of-the-art (SOTA) results on many of these tasks.

## 2 Methods

**Background** The BERT model architecture (Devlin et al., 2019) is based on a multilayer bidirectional Transformer (Vaswani et al., 2017). Instead of the traditional left-to-right language modeling objective, BERT is trained on two tasks: predicting randomly masked tokens and predicting whether two sentences follow each other. SciBERT follows the same architecture as BERT but is instead pretrained on scientific text.

**Vocabulary** BERT uses WordPiece (Wu et al., 2016) for unsupervised tokenization of the input text. The vocabulary is built such that it contains the most frequently used words or subword units. We refer to the original vocabulary released with BERT as BaseVocab.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3615–3620, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

We construct SCIVOCAB, a new WordPiece vocabulary on our scientific corpus using the SentencePiece[1] library. We produce both cased and uncased vocabularies and set the vocabulary size to 30K to match the size of BASEVOCAB. The resulting token overlap between BASEVOCAB and SCIVOCAB is 42%, illustrating a substantial difference in frequently used words between scientific and general domain texts.

**Corpus** We train SCIBERT on a random sample of 1.14M papers from Semantic Scholar (Ammar et al., 2018). This corpus consists of 18% papers from the computer science domain and 82% from the broad biomedical domain. We use the full text of the papers, not just the abstracts. The average paper length is 154 sentences (2,769 tokens) resulting in a corpus size of 3.17B tokens, similar to the 3.3B tokens on which BERT was trained. We split sentences using ScispaCy (Neumann et al., 2019),[2] which is optimized for scientific text.

## 3 Experimental Setup

### 3.1 Tasks

We experiment on the following core NLP tasks:

1. Named Entity Recognition (NER)
2. PICO Extraction (PICO)
3. Text Classification (CLS)
4. Relation Classification (REL)
5. Dependency Parsing (DEP)

PICO, like NER, is a sequence labeling task where the model extracts spans describing the Participants, Interventions, Comparisons, and Outcomes in a clinical trial paper (Kim et al., 2011). REL is a special case of text classification where the model predicts the type of relation expressed between two entities, which are encapsulated in the sentence by inserted special tokens.

### 3.2 Datasets

For brevity, we only describe the newer datasets here, and refer the reader to the references in Table 1 for the older datasets. EBM-NLP (Nye et al., 2018) annotates PICO spans in clinical trial abstracts. SciERC (Luan et al., 2018) annotates entities and relations from computer science abstracts. ACL-ARC (Jurgens et al., 2018) and SciCite (Cohan et al., 2019) assign intent labels (e.g. Comparison, Extension, etc.) to sentences from scientific papers that cite other papers. The Paper Field dataset is built from the Microsoft Academic Graph (Sinha et al., 2015)[3] and maps paper titles to one of 7 fields of study. Each field of study (i.e. geography, politics, economics, business, sociology, medicine, and psychology) has approximately 12K training examples.

### 3.3 Pretrained BERT Variants

**BERT-Base** We use the pretrained weights for BERT-Base (Devlin et al., 2019) released with the original BERT code.[4] The vocabulary is BASEVOCAB. We evaluate both cased and uncased versions of this model.

**SCIBERT** We use the original BERT code to train SCIBERT on our corpus with the same configuration and size as BERT-Base. We train 4 different versions of SCIBERT: (*i*) cased or uncased and (*ii*) BASEVOCAB or SCIVOCAB. The two models that use BASEVOCAB are finetuned from the corresponding BERT-Base models. The other two models that use the new SCIVOCAB are trained from scratch.

Pretraining BERT for long sentences can be slow. Following the original BERT code, we set a maximum sentence length of 128 tokens, and train the model until the training loss stops decreasing. We then continue training the model allowing sentence lengths up to 512 tokens.

We use a single TPU v3 with 8 cores. Training the SCIVOCAB models from scratch on our corpus takes 1 week[5] (5 days with max length 128, then 2 days with max length 512). The BASEVOCAB models take 2 fewer days of training because they aren't trained from scratch.

All pretrained BERT models are converted to be compatible with PyTorch using the pytorch-transformers library.[6] All our models (Sections 3.4 and 3.5) are implemented in PyTorch using AllenNLP (Gardner et al., 2017).

[1] https://github.com/google/sentencepiece

[2] https://github.com/allenai/SciSpaCy

[3] https://academic.microsoft.com/

[4] https://github.com/google-research/bert

[5] BERT's largest model was trained on 16 Cloud TPUs for 4 days. Expected 40-70 days (Dettmers, 2019) on an 8-GPU machine.

[6] https://github.com/huggingface/pytorch-transformers

**Casing** We follow Devlin et al. (2019) in using the cased models for NER and the uncased models for all other tasks. We also use the cased models for parsing. Some light experimentation showed that the uncased models perform slightly better (even sometimes on NER) than cased models.

### 3.4 Finetuning BERT

We mostly follow the same architecture, optimization, and hyperparameter choices used in Devlin et al. (2019). For text classification (i.e. CLS and REL), we feed the final BERT vector for the `[CLS]` token into a linear classification layer. For sequence labeling (i.e. NER and PICO), we feed the final BERT vector for each token into a linear classification layer with softmax output. We differ slightly in using an additional conditional random field, which made evaluation easier by guaranteeing well-formed entities. For DEP, we use the model from Dozat and Manning (2017) with dependency tag and arc embeddings of size 100 and biaffine matrix attention over BERT vectors instead of stacked BiLSTMs.

In all settings, we apply a dropout of 0.1 and optimize cross entropy loss using Adam (Kingma and Ba, 2015). We finetune for 2 to 5 epochs using a batch size of 32 and a learning rate of 5e-6, 1e-5, 2e-5, or 5e-5 with a slanted triangular schedule (Howard and Ruder, 2018) which is equivalent to the linear warmup followed by linear decay (Devlin et al., 2019). For each dataset and BERT variant, we pick the best learning rate and number of epochs on the development set and report the corresponding test results.

We found the setting that works best across most datasets and models is 2 or 4 epochs and a learning rate of 2e-5. While task-dependent, optimal hyperparameters for each task are often the same across BERT variants.

### 3.5 Frozen BERT Embeddings

We also explore the usage of BERT as pretrained contextualized word embeddings, like ELMo (Peters et al., 2018), by training simple task-specific models atop frozen BERT embeddings.

For text classification, we feed each sentence of BERT vectors into a 2-layer BiLSTM of size 200 and apply a multilayer perceptron (with hidden size 200) on the concatenated first and last BiLSTM vectors. For sequence labeling, we use the same BiLSTM layers and use a conditional random field to guarantee well-formed predictions. For DEP, we use the full model from Dozat and Manning (2017) with dependency tag and arc embeddings of size 100 and the same BiLSTM setup as other tasks. We did not find changing the depth or size of the BiLSTMs to significantly impact results (Reimers and Gurevych, 2017).

We optimize cross entropy loss using Adam, but holding BERT weights frozen and applying a dropout of 0.5. We train with early stopping on the development set (patience of 10) using a batch size of 32 and a learning rate of 0.001.

We did not perform extensive hyperparameter search, but while optimal hyperparameters are going to be task-dependent, some light experimentation showed these settings work fairly well across most tasks and BERT variants.

## 4 Results

Table 1 summarizes the experimental results. We observe that SCIBERT outperforms BERT-Base on scientific tasks (+2.11 F1 with finetuning and +2.43 F1 without)[8]. We also achieve new SOTA results on many of these tasks using SCIBERT.

### 4.1 Biomedical Domain

We observe that SCIBERT outperforms BERT-Base on biomedical tasks (+1.92 F1 with finetuning and +3.59 F1 without). In addition, SCIBERT achieves new SOTA results on BC5CDR and ChemProt (Lee et al., 2019), and EBM-NLP (Nye et al., 2018).

SCIBERT performs slightly worse than SOTA on 3 datasets. The SOTA model for JNLPBA is a BiLSTM-CRF ensemble trained on multiple NER datasets not just JNLPBA (Yoon et al., 2018). The SOTA model for NCBI-disease is BIOBERT (Lee et al., 2019), which is BERT-Base finetuned on 18B tokens from biomedical papers. The SOTA result for GENIA is in Nguyen and Verspoor (2019) which uses the model from Dozat and Manning (2017) with part-of-speech (POS) features, which we do not use.

In Table 2, we compare SCIBERT results with reported BIOBERT results on the subset of datasets included in (Lee et al., 2019). Interesting, SCIBERT outperforms BIOBERT results on

[7]The SOTA paper did not report a single score. We compute the average of the reported results for each class weighted by number of examples in each class.

[8]For rest of this paper, all results reported in this manner are averaged over datasets excluding UAS for DEP since we already include LAS.

| Field | Task | Dataset | SOTA | BERT-Base | | SCIBERT | |
|---|---|---|---|---|---|---|---|
| | | | | Frozen | Finetune | Frozen | Finetune |
| Bio | NER | BC5CDR (Li et al., 2016) | 88.85[7] | 85.08 | 86.72 | 88.73 | **90.01** |
| | | JNLPBA (Collier and Kim, 2004) | **78.58** | 74.05 | 76.09 | 75.77 | 77.28 |
| | | NCBI-disease (Dogan et al., 2014) | **89.36** | 84.06 | 86.88 | 86.39 | 88.57 |
| | PICO | EBM-NLP (Nye et al., 2018) | 66.30 | 61.44 | 71.53 | 68.30 | **72.28** |
| | DEP | GENIA (Kim et al., 2003) - LAS | **91.92** | 90.22 | 90.33 | 90.36 | 90.43 |
| | | GENIA (Kim et al., 2003) - UAS | **92.84** | 91.84 | 91.89 | 92.00 | 91.99 |
| | REL | ChemProt (Kringelum et al., 2016) | 76.68 | 68.21 | 79.14 | 75.03 | **83.64** |
| CS | NER | SciERC (Luan et al., 2018) | 64.20 | 63.58 | 65.24 | 65.77 | **67.57** |
| | REL | SciERC (Luan et al., 2018) | n/a | 72.74 | 78.71 | 75.25 | **79.97** |
| | CLS | ACL-ARC (Jurgens et al., 2018) | 67.9 | 62.04 | 63.91 | 60.74 | **70.98** |
| Multi | CLS | Paper Field | n/a | 63.64 | 65.37 | 64.38 | **65.71** |
| | | SciCite (Cohan et al., 2019) | 84.0 | 84.31 | 84.85 | **85.42** | **85.49** |
| Average | | | | 73.58 | 77.16 | 76.01 | 79.27 |

Table 1: Test performances of all BERT variants on all tasks and datasets. **Bold** indicates the SOTA result (multiple results bolded if difference within 95% bootstrap confidence interval). Keeping with past work, we report macro F1 scores for NER (span-level), macro F1 scores for REL and CLS (sentence-level), and macro F1 for PICO (token-level), and micro F1 for ChemProt specifically. For DEP, we report labeled (LAS) and unlabeled (UAS) attachment scores (excluding punctuation) for the same model with hyperparameters tuned for LAS. All results are the average of multiple runs with different random seeds.

| Task | Dataset | BIOBERT | SCIBERT |
|---|---|---|---|
| NER | BC5CDR | 88.85 | 90.01 |
| | JNLPBA | 77.59 | 77.28 |
| | NCBI-disease | 89.36 | 88.57 |
| REL | ChemProt | 76.68 | 83.64 |

Table 2: Comparing SCIBERT with the reported BIOBERT results on biomedical datasets.

BC5CDR and ChemProt, and performs similarly on JNLPBA despite being trained on a substantially smaller biomedical corpus.

### 4.2 Computer Science Domain

We observe that SCIBERT outperforms BERT-Base on computer science tasks (+3.55 F1 with finetuning and +1.13 F1 without). In addition, SCIBERT achieves new SOTA results on ACL-ARC (Cohan et al., 2019), and the NER part of SciERC (Luan et al., 2018). For relations in SciERC, our results are not comparable with those in Luan et al. (2018) because we are performing relation classification given gold entities, while they perform joint entity and relation extraction.

### 4.3 Multiple Domains

We observe that SCIBERT outperforms BERT-Base on the multidomain tasks (+0.49 F1 with finetuning and +0.93 F1 without). In addition, SCIBERT outperforms the SOTA on SciCite (Cohan et al., 2019). No prior published SOTA results exist for the Paper Field dataset.

## 5 Discussion

### 5.1 Effect of Finetuning

We observe improved results via BERT finetuning rather than task-specific architectures atop frozen embeddings (+3.25 F1 with SCIBERT and +3.58 with BERT-Base, on average). For each scientific domain, we observe the largest effects of finetuning on the computer science (+5.59 F1 with SCIBERT and +3.17 F1 with BERT-Base) and biomedical tasks (+2.94 F1 with SCIBERT and +4.61 F1 with BERT-Base), and the smallest effect on multidomain tasks (+0.7 F1 with SCIBERT and +1.14 F1 with BERT-Base). On every dataset except BC5CDR and SciCite, BERT-Base with finetuning outperforms (or performs similarly to) a model using frozen SCIBERT embeddings.

### 5.2 Effect of SCIVOCAB

We assess the importance of an in-domain scientific vocabulary by repeating the finetuning experiments for SCIBERT with BASEVOCAB. We find the optimal hyperparameters for SCIBERT-BASEVOCAB often coincide with those of SCIBERT-SCIVOCAB.

Averaged across datasets, we observe +0.60 F1 when using SCIVOCAB. For each scientific do-

main, we observe +0.76 F1 for biomedical tasks, +0.61 F1 for computer science tasks, and +0.11 F1 for multidomain tasks.

Given the disjoint vocabularies (Section 2) and the magnitude of improvement over BERT-Base (Section 4), we suspect that while an in-domain vocabulary is helpful, SCIBERT benefits most from the scientific corpus pretraining.

## 6 Related Work

Recent work on domain adaptation of BERT includes BIOBERT (Lee et al., 2019) and CLINICALBERT (Alsentzer et al., 2019; Huang et al., 2019). BIOBERT is trained on PubMed abstracts and PMC full text articles, and CLINICALBERT is trained on clinical text from the MIMIC-III database (Johnson et al., 2016). In contrast, SCIBERT is trained on the full text of 1.14M biomedical and computer science papers from the Semantic Scholar corpus (Ammar et al., 2018). Furthermore, SCIBERT uses an in-domain vocabulary (SCIVOCAB) while the other abovementioned models use the original BERT vocabulary (BASEVOCAB).

## 7 Conclusion and Future Work

We released SCIBERT, a pretrained language model for scientific text based on BERT. We evaluated SCIBERT on a suite of tasks and datasets from scientific domains. SCIBERT significantly outperformed BERT-Base and achieves new SOTA results on several of these tasks, even compared to some reported BIOBERT (Lee et al., 2019) results on biomedical tasks.

For future work, we will release a version of SCIBERT analogous to BERT-Large, as well as experiment with different proportions of papers from each domain. Because these language models are costly to train, we aim to build a single resource that's useful across multiple domains.

## Acknowledgment

We thank the anonymous reviewers for their comments and suggestions. We also thank Waleed Ammar, Noah Smith, Yoav Goldberg, Daniel King, Doug Downey, and Dan Weld for their helpful discussions and feedback. All experiments were performed on beaker.org and supported in part by credits from Google Cloud.

## References

Emily Alsentzer, John R. Murphy, Willie Boag, Wei-Hung Weng, Di Jin, Tristan Naumann, and Matthew B. A. McDermott. 2019. Publicly available clinical bert embeddings. In *ClinicalNLP workshop at NAACL*.

Waleed Ammar, Dirk Groeneveld, Chandra Bhagavatula, Iz Beltagy, Miles Crawford, Doug Downey, Jason Dunkelberger, Ahmed Elgohary, Sergey Feldman, Vu Ha, Rodney Kinney, Sebastian Kohlmeier, Kyle Lo, Tyler Murray, Hsu-Han Ooi, Matthew Peters, Joanna Power, Sam Skjonsberg, Lucy Lu Wang, Chris Wilhelm, Zheng Yuan, Madeleine van Zuylen, and Oren Etzioni. 2018. Construction of the literature graph in semantic scholar. In *NAACL*.

Arman Cohan, Waleed Ammar, Madeleine van Zuylen, and Field Cady. 2019. Structural scaffolds for citation intent classification in scientific publications. In *NAACL-HLT*, pages 3586–3596, Minneapolis, Minnesota. Association for Computational Linguistics.

Nigel Collier and Jin-Dong Kim. 2004. Introduction to the bio-entity recognition task at jnlpba. In *NLPBA/BioNLP*.

Tim Dettmers. 2019. TPUs vs GPUs for Transformers (BERT). http://timdettmers.com/2018/10/17/tpus-vs-gpus-for-transformers-bert/. Accessed: 2019-02-22.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Rezarta Islamaj Dogan, Robert Leaman, and Zhiyong Lu. 2014. NCBI disease corpus: A resource for disease name recognition and concept normalization. *Journal of biomedical informatics*, 47:1–10.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. *ICLR*.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. Allennlp: A deep semantic natural language processing platform. In *arXiv:1803.07640*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *ACL*.

Kexin Huang, Jaan Altosaar, and Rajesh Ranganath. 2019. Clinicalbert: Modeling clinical notes and predicting hospital readmission. *arXiv:1904.05342*.

Alistair E. W. Johnson, Tom J. Pollard aand Lu Shen, Liwei H. Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, , and Roger G. Mark. 2016.

Mimic-iii, a freely accessible critical care database. In *Scientific Data, 3:160035*.

David Jurgens, Srijan Kumar, Raine Hoover, Daniel A. McFarland, and Daniel Jurafsky. 2018. Measuring the evolution of a scientific field through citation frames. *TACL*, 06:391–406.

Jin-Dong Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. 2003. GENIA corpus - a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19:i180i182.

Su Kim, David Martínez, Lawrence Cavedon, and Lars Yencken. 2011. Automatic classification of sentences to support evidence based medicine. In *BMC Bioinformatics*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *ICLR*.

Jens Kringelum, Sonny Kim Kjærulff, Søren Brunak, Ole Lund, Tudor I. Oprea, and Olivier Taboureau. 2016. ChemProt-3.0: a global chemical biology diseases mapping. In *Database*.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. In *arXiv:1901.08746*.

Jiao Li, Yueping Sun, Robin J. Johnson, Daniela Sciaky, Chih-Hsuan Wei, Robert Leaman, Allan Peter Davis, Carolyn J. Mattingly, Thomas C. Wiegers, and Zhiyong Lu. 2016. BioCreative V CDR task corpus: a resource for chemical disease relation extraction. *Database : the journal of biological databases and curation*.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *EMNLP*.

Mark Neumann, Daniel King, Iz Beltagy, and Waleed Ammar. 2019. ScispaCy: Fast and robust models for biomedical natural language processing. In *arXiv:1902.07669*.

Dat Quoc Nguyen and Karin M. Verspoor. 2019. From pos tagging to dependency parsing for biomedical event extraction. *BMC Bioinformatics*, 20:1–13.

Benjamin Nye, Junyi Jessy Li, Roma Patel, Yinfei Yang, Iain James Marshall, Ani Nenkova, and Byron C. Wallace. 2018. A corpus with multi-level annotations of patients, interventions and outcomes to support language processing for medical literature. In *ACL*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke S. Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Nils Reimers and Iryna Gurevych. 2017. Optimal hyperparameters for deep lstm-networks for sequence labeling tasks. In *EMNLP*.

Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June Paul Hsu, and Kuansan Wang. 2015. An overview of microsoft academic service (MAS) and applications. In *WWW*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Gregory S. Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. abs/1609.08144.

Wonjin Yoon, Chan Ho So, Jinhyuk Lee, and Jaewoo Kang. 2018. CollaboNet: collaboration of deep neural networks for biomedical named entity recognition. In *DTMBio workshop at CIKM*.

# Humor Detection: A Transformer Gets the Last Laugh

**Orion Weller**
Computer Science Department
Brigham Young University
orionw@byu.edu

**Kevin Seppi**
Computer Science Department
Brigham Young University
kseppi@byu.edu

## Abstract

Much previous work has been done in attempting to identify humor in text. In this paper we extend that capability by proposing a new task: assessing whether or not a joke is humorous. We present a novel way of approaching this problem by building a model that learns to identify humorous jokes based on ratings gleaned from Reddit pages, consisting of almost 16,000 labeled instances. Using these ratings to determine the level of humor, we then employ a Transformer architecture for its advantages in learning from sentence context. We demonstrate the effectiveness of this approach and show results that are comparable to human performance. We further demonstrate our model's increased capabilities on humor identification problems, such as the previously created datasets for short jokes and puns. These experiments show that this method outperforms all previous work done on these tasks, with an F-measure of 93.1% for the Puns dataset and 98.6% on the Short Jokes dataset.

## 1 Introduction

Recent advances in natural language processing and neural network architecture have allowed for widespread application of these methods in Text Summarization (Liu et al., 2018), Natural Language Generation (Bahuleyan, 2018), and Text Classification (Yang et al., 2016). Such advances have enabled scientists to study common language practices. One such area, humor, has garnered focus in classification (Zhang and Liu, 2014; Chen and Soo, 2018), generation (He et al., 2019; Valitutti et al., 2013), and in social media (Raz, 2012).

The next question then is, what makes a joke humorous? Although humor is a universal construct, there is a wide variety between what each individual may find humorous. We attempt to focus on a subset of the population where we can quantitatively measure reactions: the popular Reddit r/Jokes thread. This forum is highly popular - with tens of thousands of jokes being posted monthly and over 16 million members. Although larger joke datasets exist, the r/Jokes thread is unparalleled in the amount of rated jokes it contains. To the best of our knowledge there is no comparable source of rated jokes in any other language. These Reddit posts consist of the body of the joke, the punchline, and the number of reactions or *upvotes*. Although this type of humor may only be most enjoyable to a subset of the population, it is an effective way to measure responses to jokes in a large group setting.[1]

What enables us to perform such an analysis are the recent improvements in neural network architecture for natural language processing. These breakthroughs started with the Convolutional Neural Network (LeCun et al., 1998) and have recently included the inception (Bahdanau et al., 2015) and progress of the Attention mechanism (Luong et al., 2015; Xu et al., 2015), and the Transformer architecture (Vaswani et al., 2017).

## 2 Related Work

In the related work of joke identification, we find a myriad of methods employed over the years: statistical and N-gram analysis (Taylor and Mazlack, 2004), Regression Trees (Purandare and Litman, 2006), Word2Vec combined with K-NN Human Centric Features (Yang et al., 2015), and Convolutional Neural Networks (Chen and Soo, 2018).

This previous research has gone into many settings where humor takes place. Chen and Soo (2018) studied audience laughter compared to textual transcripts in order to identify jokes in conversation, while much work has also gone into us-

[1] See the thread (of varied and not safe for work content) at this link. We do not endorse these jokes.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3621–3625,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| Body | Punchline | Score |
|---|---|---|
| Man, I was so tired last night; I had a dream I was a muffler... | and I woke up exhausted | 276 |
| I told my teenage niece to go get me a newspaper... She laughed at me, and said, "Oh uncle you're so old. Just use my phone." | So I slammed her phone against the wall to kill a spider. | 28315 |

Table 1: Example format of the Reddit Jokes dataset

ing and creating datasets like the Pun of the Day (Yang et al., 2015), 16000 One-liners (Mihalcea and Strapparava, 2005), and even Ted Talks (Chen and Soo, 2018).

## 3 Data

We gathered jokes from a variety of sources, each covering a different type of humor. These datasets include jokes of multiple sentences (the Short Jokes dataset), jokes with only one sentence (the Puns dataset), and more mixed jokes (the Reddit dataset). We have made our code and datasets open source for others to use. [2]

### 3.1 Reddit

Our Reddit data was gathered using Reddit's public API, collecting the most recent jokes. Every time the scraper ran, it also updated the upvote score of the previously gathered jokes. This data collection occurred every hour through the months of March and April 2019. Since the data was already split into body and punchline sections from Reddit, we created separate datasets containing the body of the joke exclusively and the punchline of the joke exclusively. Additionally, we created a dataset that combined the body and punchline together.

Some sample jokes are shown in Table 1, above. The distribution of joke scores varies wildly, ranging from 0 to 136,354 upvotes. We found that there is a major jump between the 0-200 upvote range and the 200 range and onwards, with only 6% of jokes scoring between 200-20,000. We used this natural divide as the cutoff to decide what qualified as a funny joke, giving us 13884 not-funny jokes and 2025 funny jokes.

### 3.2 Short Jokes

The Short Jokes dataset, found on Kaggle, contains 231,657 short jokes scraped from various joke websites with lengths ranging from 10 to 200 characters. The previous work by Chen and Soo (2018) combined this dataset with the WMT162 English news crawl. Although their exact combined dataset is not publicly available, we used the same method and news crawl source to create a similar dataset. We built this new Short Jokes dataset by extracting sentences from the WMT162 news crawl that had the same distribution of words and characters as the jokes in the Short Jokes dataset on Kaggle[3]. This was in order to match the two halves (jokes and non-jokes) as closely as possible.

### 3.3 Pun of the Day

This dataset was scraped by Yang et al. (2015) and contains 16001 puns and 16002 not-punny sentences. We gratefully acknowledge their help in putting together and giving us use of this dataset. These puns were constructed from the Pun of the Day website while the negative samples were gathered from news websites.

## 4 Methods

In this section we will discuss the methods and model used in our experiments.

### 4.1 Our Model

We have chosen to use the pre-trained BERT (Devlin et al., 2018) as the base of our model. BERT is a multi-layer bidirectional Transformer encoder and was initially trained on a 3.3 billion word corpus. The model can be fined-tuned with another additional output layer for a multitude of other tasks. We chose to use this Transformer based model as our initial platform because of its success at recognizing and attending to the most important words in both sentence and paragraph structures.

In Figure 1, originally designed by Vaswani et al. (2017), we see the architecture of a Transformer model: the initial input goes up through an encoder, which has two parts: a multi-headed

[2]Our code and datasets are publicly available at this link.

[3]The Short Jokes dataset from Kaggle is available here.

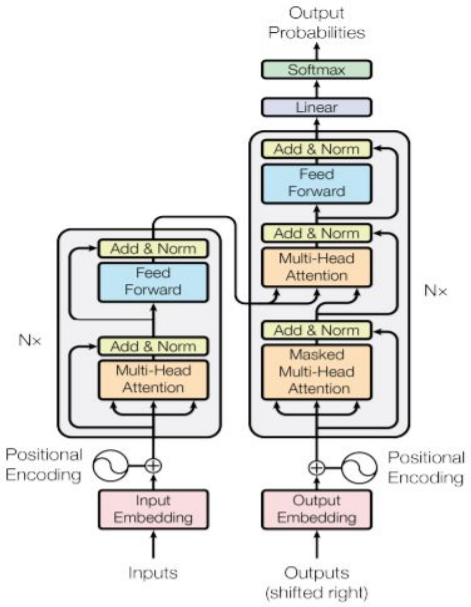


Figure 1: Transformer Model Architecture

self attention layer, followed by a feed-forward network. It then outputs the information into the decoder, which includes the previously mentioned layers, plus an additional masked attention step. Afterwords, it is transformed through a softmax into the output. This model's success is in large part due to the Transformer's self-attention layers.

We chose a learning rate of 2e-05 and a max sequence length of 128. We trained the model for a maximum of 7 epochs, creating checkpoints along the way.

### 4.2 Training

Since our data was unbalanced we decided to upsample the humorous jokes in training. We split the dataset into a 75/25 percent split, stratifying with the labels. We then upsampled the minority class in the training set until it reached an even 50 percent. This helped our model learn in a more balanced way despite the uneven amount of non-humorous jokes. Our validation and test sets were composed of the remaining 25%, downsampling the data into a 50/50 class split so that the accuracy metric could be balanced and easily understood.

To show how our model compares to the previous work done, we also test on the Short Joke and Pun datasets mentioned in the Data section. For these datasets we will use the metrics (Accuracy, Precision, Recall, and F1 Score) designated in Chen and Soo (2018) as a comparison. We use the same model format as previously mentioned, trained on the Reddit dataset. We then immediately apply the model to predict on the Short Joke and Puns dataset, without further fine-tuning, in order to compare the model. However, because both the Puns and Short Joke datasets have large and balanced labels, we do so without the upsampling and downsampling steps used for the Reddit dataset.

| Method | Body | Punchline | Full |
|---|---|---|---|
| CNN | 0.651 | 0.684 | 0.688 |
| Transformer | **0.661** | **0.692** | **0.724** |
| Human (General) | 0.493 | 0.592 | 0.663 |

Table 2: Results of Accuracy on Reddit Jokes dataset

## 5 Experiments

In this section we will introduce the baselines and models used in our experiments.

### 5.1 Baselines

In order to have fair baselines, we used the following two models: a CNN with Highway Layers as described by Chen and Soo (2018) and developed by Srivastava et al. (2015), and human performance from a study on Amazon's Mechanical Turk. We wanted to have the general population rate these same jokes, thus showing the difference between a general audience and a specific subset of the population, in particular, Reddit r/Jokes users. Since the Reddit users obviously found these jokes humorous, this experiment would show whether or not a more general population agreed with those labels.

We had 199 unique participants rate an average of 30 jokes each with the prompt *"do you find this joke humorous?"* If the participant was evaluating a sample from a body or punchline only dataset we prefaced our question with a sentence explaining that context, for example: *"Below is the punchline of a joke. Based on this punchline, do you think you would find this joke humorous?"* Taking these labels, we used the most frequently chosen tag from a majority vote to calculate the percentages found in the *Human* section of Table 2.

### 5.2 Results

In Table 2, we see the results of our experiment with the Reddit dataset. We ran our models on

| Previous Work: | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|
| Word2Vec+HCF | 0.797 | 0.776 | 0.836 | 0.705 |
| CNN | 0.867 | 0.880 | 0.859 | 0.869 |
| CNN+F | 0.892 | 0.886 | 0.907 | 0.896 |
| CNN+HN | 0.892 | 0.889 | 0.903 | 0.896 |
| CNN+F+HN | 0.894 | 0.866 | **0.940** | 0.901 |
| | | | | |
| Our Methods: | Accuracy | Precision | Recall | F1 |
| Transformer | **0.930** | **0.930** | 0.931 | **0.931** |

Table 3: Comparison of Methods on Pun of the Day Dataset. $HCF$ represents Human Centric Features, $F$ for increasing the number of filters, and $HN$ for the use of highway layers in the model. See (Chen and Soo, 2018; Yang et al., 2015) for more details regarding these acronyms.

the body of the joke exclusively, the punchline exclusively, and both parts together (labeled *full* in our table). On the full dataset we found that the Transformer achieved an accuracy of 72.4 percent on the hold out test set, while the CNN was in the high 60's. We also note that the general human classification found 66.3% of the jokes to be humorous.

In order to understand what may be happening in the model, we used the body and punchline only datasets to see what part of the joke was most important for humor. We found that all of the models, including humans, relied more on the punchline of the joke in their predictions (Table 2). Thus, it seems that although both parts of the joke are needed for it to be humorous, the punchline carries higher weight than the body. We hypothesize that this is due to the variations found in the different joke bodies: some take paragraphs to set up the joke, while others are less than a sentence.

Our experiment with the Short Jokes dataset found the Transformer model's accuracy and F1 score to be 0.986. This was a jump of 8 percent from the most recent work done with CNNs (Table 4).

The results on the Pun of the Day dataset are shown in Table 3 above. It shows an accuracy of 93 percent, close to 4 percent greater accuracy than the best CNN model proposed. Although the CNN model used a variety of techniques to extract the best features from the dataset, we see that the self-attention layers found even greater success in pulling out the crucial features.

## 6 Discussion

Considering that a joke's humor value is subjective, the results on the Reddit dataset are surprising. The model has used the context of the words to determine, with high probability, what an average Reddit r/Jokes viewer will find humorous. When we look at the general population's opinion as well, we find a stark difference between their preferences and those of the Reddit users (Table 2). We would hypothesize that our model is learning the specific type of humor enjoyed by those who use the Reddit r/Jokes forum. This would suggest that humor can be learned for a specific subset of the population.

| Method | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|
| CNN+F+HN | 0.906 | 0.902 | 0.946 | 0.924 |
| Transformer | **0.986** | 0.986 | 0.986 | 0.986 |

Table 4: Results on Short Jokes Identification

The model's high accuracy and F1 scores on the Short Jokes and Pun of the Day dataset show the effectiveness of the model for transfer learning. This result is not terribly surprising. If the model can figure out which jokes are funny, it seems to be an easier task to tell when something isn't a joke at all.

Although these results have high potential, defining the absolute truth value for a joke's humor is a challenging, if not impossible task. However, these results indicate that, at least for a subset of the population, we can find and identify jokes that will be most humorous to them.

## 7 Conclusion

In this paper, we showed a method to define the measure of a joke's humor. We explored the idea of using machine learning tools, specifically a Transformer neural network architecture, to discern what jokes are funny and what jokes are not. This proposed model does not require any human

interaction to determine, aside from the text of the joke itself, which jokes are humorous. This architecture can predict the level of humor for a specific audience to a higher degree than a general audience consensus. We also showed that this model has increased capability in joke identification as a result, with higher accuracy and F1 scores than previous work on this topic.

## References


Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. *International Conference on Learning Representations*.

Hareesh Bahuleyan. 2018. Natural Language Generation with Neural Variational Models. *arXiv e-prints*, page arXiv:1808.09012.

Peng-Yu Chen and Von-Wun Soo. 2018. Humor recognition using deep learning. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *North American Chapter of the Association for Computational Linguistics*.

He He, Nanyun Peng, and Percy Liang. 2019. Pun generation with surprise. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1734–1744, Minneapolis, Minnesota. Association for Computational Linguistics.

Yann LeCun, Leon Bottou, Y Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278 – 2324.

Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating Wikipedia by Summarizing Long Sequences. *International Conference on Learning Representations*.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages pages 1412 – 1421.

Rada Mihalcea and Carlo Strapparava. 2005. Making computers laugh: Investigations in automatic humor recognition. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 531–538, Stroudsburg, PA, USA. Association for Computational Linguistics.

Amruta Purandare and Diane Litman. 2006. Humor: Prosody analysis and automatic recognition for f*r*i*e*n*d*s*. *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 208–215.

Yishay Raz. 2012. Automatic humor classification on twitter. *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Student Research Workshop*, pages 66–70.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway Networks. *arXiv e-prints*, page arXiv:1505.00387.

Julia M. Taylor and Lawrence J. Mazlack. 2004. Computationally recognizing wordplay in jokes. *In Proceedings of CogSci 2004*.

Alessandro Valitutti, Hannu Toivonen, Antoine Doucet, and Jukka M Toivanen. 2013. let everything turn well in your wife: Generation of adult humor using lexical constraints. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 243–248.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *31st Conference on Neural Information Processing Systems*, pages 17–21.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. 2015. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. *arXiv e-prints*, page arXiv:1502.03044.

Diyi Yang, Alon Lavie, Chris Dyer, and Eduard Hovy. 2015. Humor recognition and humor anchor extraction. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2367–2376.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. 2016. Hierarchical attention networks for document classification. *HLT-NAACL*.

Renxian Zhang and Naishi Liu. 2014. Recognizing humor on twitter. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '14, pages 889–898, New York, NY, USA. ACM.

# Combining Global Sparse Gradients with Local Gradients in Distributed Neural Network Training

**Alham Fikri Aji** **Kenneth Heafield** **Nikolay Bogoychev**
School of Informatics, University of Edinburgh
10 Crichton Street
Edinburgh EH8 9AB
Scotland, European Union
{a.fikri, n.bogoych}@ed.ac.uk, kheafiel@inf.ed.ac.uk

## Abstract

One way to reduce network traffic in multi-node data-parallel stochastic gradient descent is to only exchange the largest gradients. However, doing so damages the gradient and degrades the model's performance. Tranformer models degrade dramatically while the impact on RNNs is smaller. We restore gradient quality by combining the compressed global gradient with the node's locally computed uncompressed gradient. Neural machine translation experiments show that Transformer convergence is restored while RNNs converge faster. With our method, training on 4 nodes converges up to 1.5x as fast as with uncompressed gradients and scales 3.5x relative to single-node training.

## 1 Introduction

In recent years, neural network models have grown dramatically in terms of number of parameters (Wen et al., 2017), so exchanging gradients during data-parallel training is costly in terms of both bandwidth and time, especially in a distributed setting. Communication can be reduced (possibly at the expense of convergence) by sending only the top 1% of largest gradients in terms of absolute values, a method known as gradient dropping (Strom, 2015; Dryden et al., 2016; Aji and Heafield, 2017; Lin et al., 2018). Related methods are synchronizing less often (McMahan et al., 2017; Ott et al., 2018; Bogoychev et al., 2018) and quantization (Seide et al., 2014; Alistarh et al., 2016).

As these compression methods are lossy, each node's locally computed gradient is not immediately reflected in the global gradient. Our experiments show that gradient compression damages the model's performance, especially in the case of a Transformer model (Vaswani et al., 2017), which is known to be sensitive to noisy gradients (Chen et al., 2018; Ott et al., 2018; Aji and Heafield, 2019). We aim to repair the compressed gradient by combining it with the local gradients to improve the trade-off between convergence and compression rates.

In this paper, we apply gradient dropping to reduce the inter-node communication during distributed neural network training, which leads to faster training speed but reduced model convergence rate. We find that combining the sparse global gradient with the dense local gradient improves convergence. However, adding local information means that nodes' parameters will diverge over time. We address this by periodically averaging the model (McMahan et al., 2017), achieving faster end-to-end training time.

## 2 Related Work

### 2.1 Sparse Gradient Compression

Gradients are skewed: most values are near zero while very few have large absolute value (Aji and Heafield, 2017). Formally, Pearson's skewness coefficient is typically 2–4, but up to 262 in embedding matrices where much of the parameters lie. Sparse gradient compression exploits this by rounding gradients below a threshold to zero, sending only a sparse matrix of large gradients (Strom, 2015). The threshold can be set dynamically to the top 1% of gradients, achieving constant compression (Dryden et al., 2016). Unsent gradients are added to the next gradient prior to compression (Seide et al., 2014).

Gradient dropping is outlined in Algorithm 1. At each time step $t$, each node $n$ computes a local gradient $L_t^n$ on its data. The error feedback mechanism adds unsent gradients from the previous step $E_{t-1}$ to the local gradient $L_t^n$. The combined gradient is then broken into sparse gradient $S_t^n$ and residual $E_t$. Although the gradient is sparse, all



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3626–3631,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

**Algorithm 1** Gradient dropping on node $n$

**procedure** SPARSESGD($L_t^n$)
  ▷ $L_t^n$ is local gradient of node $n$ at step $t$.
  $E_t^n \leftarrow E_{t-1}^n + L_t^n$
  $threshold \leftarrow K$-th largest of $|E_t^n|$
  $mask \leftarrow |E_t^n| \geq threshold$
  $S_t^n \leftarrow mask \odot E_t^n$
  $E_t^n \leftarrow \neg mask \odot E_t^n$
  $G_t \leftarrow AllReduce(S_t^n)$
  $ApplyOptimizer(G_t)$
**end procedure**

parameters are updated because Adam (Kingma and Ba, 2015) has momentum terms. Parameter updates run redundantly in all nodes so that only gradients are sent over the network.

The sum of sparse gradients is less sparse. We can send the summed gradient by itself (Lin et al., 2018) or again take the top 1% of summed gradients. Our cluster of 4 nodes is small enough that there was little speed difference, so we did not recompress the summed gradients.

### 2.2 Federated Averaging

Another way to reduce the bandwidth cost in multi-node training is by reducing the communication frequency (McMahan et al., 2017). In federated averaging, workers do not exchange gradients. Instead, each worker uses its local gradient to update its own local parameters. Each worker updates their local parameters by averaging across other nodes once every few steps.

In contrast with gradient dropping, federated averaging mainly uses the worker's local gradients for parameter updates. Gradients from other workers are not directly communicated and are therefore not taken into account by the optimizer.

## 3 Combining With Local Gradients

Recent work suggests that the Transformer is sensitive to noisy gradients, resulting in substantially worse models (Chen et al., 2018; Ott et al., 2018; Aji and Heafield, 2019). Consistent with these findings, both gradient sparsification and federated averaging yield low-quality Transformer models in our experiments. In gradient sparsification, noise comes from both thresholding and the error-feedback mechanism, which causes stale gradients. Federated averaging also introduces stale updates as this approach delays model synchronization. Previous work has shown that both noisy and stale gradients damage the model's quality (McMahan and Streeter, 2014; Ott et al., 2018; Dutta et al., 2018).

To address noisy updates in gradient sparsification, we combine the compressed global gradient and the uncompressed locally computed gradient in an effort to better approximate the true global gradient. Formally, let $G_t$ be the compressed global gradient at time $t$ and $L_t^n$ be the gradient computed locally on node $n$. These will be combined into $C_t^n$ that will be used to update the parameters.

An arguably naïve method sums the two gradients. With a scale-invariant optimizer like Adam, this is equivalent to averaging.

$$C_t^n = G_t + L_t^n$$

However, some of the locally-computed gradients were sent out and became part of the global gradient, so they will be double-counted by the sum. To compensate, we can subtract out the gradients $S_t^n$ sent by node $n$.

$$C_t^n = G_t - S_t^n + L_t^n$$

The term $G_t - S_t^n$ equals to the sum of all sparse gradients from other nodes (or approximates it when the all-reduce compresses the result). The local gradient $L_t^n$ used for updating does not include the error-feedback term $E_t^n$ to prevent applying gradients multiple times while they are pending in error-feedback.

### 3.1 Periodic Synchronization

Nodes will diverge because local gradients differ. Therefore, models are averaged periodically. We average parameters (McMahan et al., 2017) every 500 steps with minimal impact on speed.

In the limit, a gradient is applied twice. First, a local update eventually makes its way to the other nodes via periodic averaging. Second, it accumulates with enough other gradients to be selected for inclusion in the compressed gradient and applied as part of a global update.

## 4 Experimental Setup

We use Marian (Junczys-Dowmunt et al., 2018) to train on nodes with 4xP100s. Multi-node experiments use 4 of these nodes, each connected with 40Gb Mellanox Infiniband. These scenarios will be abbreviated as 1x4 (one node with four GPUs) and 4x4 (four nodes with four GPUs each).

### 4.1 Model and Dataset

We perform our neural machine translation experiments on the following architectures.

**Transformer:** We train a Transformer model with six encoder and six decoder layers with tied embeddings. The model has 62M parameters. We train the model on the WMT 2017 English to German dataset with back-translated monolingual corpora (Sennrich et al., 2016b) and byte-pair encoding (Sennrich et al., 2016c), consisting of 19.1M sentence pairs. Model performance is validated on newstest2016 and tested on newstest2017.

**Deep RNN:** We also train a deep RNN model Sennrich et al. (2017) with eight layers of bidirectional LSTM consisting of 225M parameters. We train the model with the same English to German dataset from the Transformer experiment.

**Shallow RNN:** Our shallow RNN model is based on the winning system by Sennrich et al. (2016a) and is a single layer bidirectional encoder-decoder LSTM with attention consisting of 119M parameters. We train this model on WMT 2016 Romanian to English dataset, consisting of 2.5M sentence pairs. We also apply byte-pair encoding to this dataset. Model performance is validated on newsdev2016 and tested on newstest2016.

We apply layer normalization (Lei Ba et al., 2016) and exponential smoothing to train the model for 8 epochs of training.

### 4.2 Scaling Hyperparameters

In all our experiments, we use a memory budget of 10GB per GPU to dynamically fit as many sentences as possible, corresponding to an average of 450 and 250 sentences per batch per GPU for Ro-En and En-De, respectively. Hence, we apply several adjustments to the hyperparameters to accommodate the larger effective batch size of multi-node synchronous SGD:

**Learning rate:** The Adam optimizer is scale-invariant, so the parameters move at the same magnitude regardless the gradient size. Therefore, we linearly scale the learning rate in all multi-node experiments, as suggested by Goyal et al. (2017). On one node, we use a learning rate of 0.0003 for Transformer and deep RNN models, and 0.001 for the shallow RNN model. These values multiplied by 4 for the 4-node setting. The single-node learning rates were optimized in the sense that further increasing them damages performance.

**Warm-up:** Learning rate warm-up helps overcome initial model instability when training with large mini-batches (Goyal et al., 2017). We add a linear learning rate warm-up for the Transformer, deep RNNs, and shallow RNNs for the first 16k, 4k, and 2k steps respectively. We apply inverse square root cool-down following Vaswani et al. (2017) for Transformer and deep RNN models.

We follow the rest of the hyperparameter settings as suggested in the papers (Vaswani et al., 2017; Sennrich et al., 2017, 2016a).

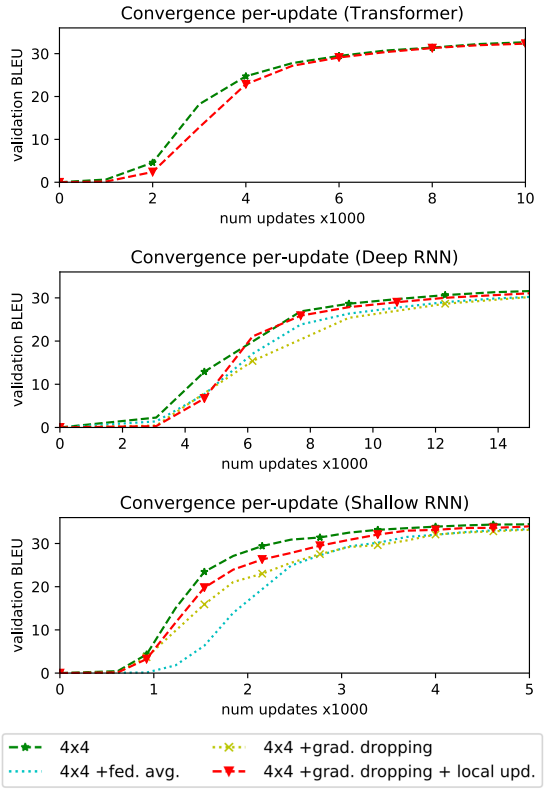


Figure 1: Model convergence per-updates on gradient dropping with local gradient update. We focus on the early stage of the training, before the BLEU converged. Training a Transformer with gradient dropping or federated averaging yielded 0 BLEU.

## 5 Results and Analysis

### 5.1 Restoring Quality

We approximate impact on quality by measuring the BLEU score (Papineni et al., 2002) obtained per update, experimenting with both RNN and Transformer systems. The baselines are vanilla 4-node synchronous SGD, gradient dropping (Aji and Heafield, 2017), and federated averaging (McMahan et al., 2017). For gradient dropping, we perform drop ratio warmup, gradually increasing it to 99% after 1000 steps. We average

| Model | Transformer dev | Transformer test | Deep RNN dev | Deep RNN test | Shallow RNN dev | Shallow RNN test |
|---|---|---|---|---|---|---|
| Multi-node (4x4) | 35.39 | 28.78 | 34.45 | 27.81 | 35.45 | 34.45 |
| 4x4 + gradient dropping | 0.00 | 0.00 | 34.38 | 27.50 | 35.20 | 33.89 |
| 4x4 + federated averaging | 0.00 | 0.00 | 34.33 | 27.42 | 35.25 | 33.93 |
| 4x4 + gradient dropping + local update | 35.07 | 28.50 | 34.52 | 27.68 | 35.35 | 34.45 |

Table 1: Training quality of multi-node training with gradient compression techniques, measured with BLEU.

the model every 20 steps in federated averaging experiment, and every 500 steps in our proposed method.

Figure 1 shows BLEU score per update. Gradient dropping and federated averaging reduce gradient quality and improvement per update is slower. In the Transformer case, the model is incapable of training at all. Local gradient incorporation improves the sparse gradient quality and improves convergence per-epoch over gradient dropping. In all architectures, the model achieved a comparable training curve compared to the uncompressed multi-node training.

Table 1 summarizes model's performance in terms of BLEU. With local gradient incorporation, the models obtained better final quality, performing closer to uncompressed multi-node training. Local gradient incorporation enables the Transformer to train with a sparse gradient, albeit with slight quality degradation (0.28–0.32%). This result confirms Transformer's sensitivity to noisy updates and the ability of local gradients to mostly repair them.

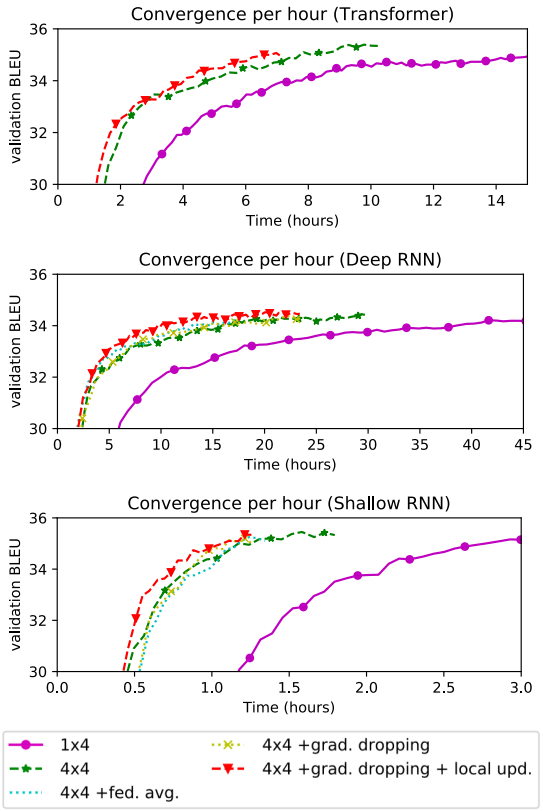


Figure 2: Convergence over time of gradient dropping with local gradient update.

## 5.2 Improving Training Speed

We measure the speed improvement of our proposed method by capturing the raw processing speed and time to reach certain BLEU. We compare it to both gradient dropping and federated averaging. We also measure the training efficiency by comparing the results with a single-node system. For the Transformer, we exclude gradient dropping and federated averaging as the models fail to train.

Table 2 summarizes our experiments. Gradient dropping reduces network traffic 50-fold and significantly improves the raw training speed in multi-node setting up to 3.4x over single-node, and up to 1.6x faster raw speed over uncompressed multi-node setting. Federated averaging is faster because there is no additional communication overhead for every step, and no extra computational cost for sparse gradients' compression. Finally, our method incurs the combined cost of gradient dropping, occasional federated averaging, and local updates so it is slower than gradient dropping at raw speed, but still substantially faster than uncompressed multi-node training.

While vanilla gradient dropping and federated averaging have better raw speed, there is no clear improvement on convergence speed as noisy gradients damage the convergence. Local gradient update restores the gradient and improves the convergence speed. In our RNN experiments, convergence speedup is closer to the raw speedup, up to 3.5x single-node performance.

The Transformer convergence rate increases more slowly than raw batch processing speed.

| Model | Words/ second | Raw speedup (1x4 / 4x4) | Time to conv. | conv. speedup (1x4 / 4x4) |
|---|---|---|---|---|
| **Transformer (En-De)** | | | | |
| Single-node (1x4) | 36029 | - | 7.61h | - |
| Multi-node (4x4) | 97111 | 2.7x / - | 4.76h | 1.6x / - |
| 4x4 + gradient dropping + local update | 122914 | 3.4x / 1.3x | 4.03h | 1.9x / 1.2x |
| **Deep RNN (En-De)** | | | | |
| Single-node (1x4) | 18205 | - | 23.68h | - |
| Multi-node (4x4) | 42930 | 2.4x / - | 10.59h | 2.2x / - |
| 4x4 + gradient dropping | 60090 | 3.3x / 1.4x | 8.94h | 2.6x / 1.2x |
| 4x4 + federated averaging | 66149 | 3.6x / 1.5x | 9.50h | 2.5x / 1.1x |
| 4x4 + gradient dropping + local update | 59747 | 3.3x / 1.4x | 6.80h | 3.5x / 1.5x |
| **Shallow RNN (Ro-En)** | | | | |
| Single-node (1x4) | 36466 | - | 2.37h | - |
| Multi-node (4x4) | 75641 | 2.1x / - | 1.05h | 2.3x / - |
| 4x4 + gradient dropping | 118189 | 3.2x / 1.6x | 0.94h | 2.5x / 1.1x |
| 4x4 + federated averaging | 124273 | 3.4x / 1.6x | 1.06h | 2.2x / 1.0x |
| 4x4 + gradient dropping + local update | 117756 | 3.2x / 1.6x | 0.85h | 2.8x / 1.2x |

Table 2: Speed performance of gradient dropping with local gradient update, compared to several baselines. Time to convergence is time needed to reach 34.5 BLEU (Transformer & Shallow RNN) or 33.5 BLEU (Deep RNN).

While the rule of thumb is to scale learning rate linearly with batch size (Goyal et al., 2017), the Transformer model is also sensitive to high learning rates (Aji and Heafield, 2019). We obtained 1.6x convergence speedup, instead of the expected 2.7x speedup. Scaling the learning rate sublinearly can be explored.

Compression results are of course dependent on the ratio between computation and network bandwidth in a system, as well as model size. Because the method reduces network load, we would expect to see even larger speed improvement with commodity hardware instead of the 40 gigabit Infiniband network used in our experiments.

## 6 Conclusion

We improve model convergence when training with sparse gradients by utilizing an additional locally-computed gradient, while also negates the quality loss in terms of BLEU caused by gradient dropping. With gradient dropping and local gradient incorporation, we improve the raw training speed in terms of word/second by up to 3.4x over single-node system, and up to 1.6x over uncompressed multi-node system. We also evaluate the training speed by the time needed to reach a near-convergence BLEU score. In this case, we improve the training speed by up to 3.5x over single-node system and up to 1.5x over uncompressed multi-node system.

## 7 Acknowledgements

Alham Fikri Aji is funded by the Indonesia Endowment Fund for Education scholarship scheme. This work was performed using resources provided by the Cambridge Service for Data Driven Discovery (CSD3) operated by the University of Cambridge Research Computing Service (http://www.csd3.cam.ac.uk/), provided by Dell EMC and Intel using Tier-2 funding from the Engineering and Physical Sciences Research Council (capital grant EP/P020259/1), and DiRAC funding from the Science and Technology Facilities Council (www.dirac.ac.uk).

## References

Alham Fikri Aji and Kenneth Heafield. 2017. Sparse communication for distributed gradient descent. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 440–445.

Alham Fikri Aji and Kenneth Heafield. 2019. Making asynchronous stochastic gradient descent work for transformers. *arXiv preprint arXiv:1906.03496*.

Dan Alistarh, Jerry Li, Ryota Tomioka, and Milan Vojnovic. 2016. QSGD: randomized quantization for communication-optimal stochastic gradient descent. *CoRR*, abs/1610.02132.

Nikolay Bogoychev, Kenneth Heafield, Alham Fikri Aji, and Marcin Junczys-Dowmunt. 2018. Accel-

erating asynchronous stochastic gradient descent for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2991–2996.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, et al. 2018. The best of both worlds: Combining recent advances in neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 76–86.

Nikoli Dryden, Sam Ade Jacobs, Tim Moon, and Brian Van Essen. 2016. Communication quantization for data-parallel training of deep neural networks. In *Proceedings of the Workshop on Machine Learning in High Performance Computing Environments*, pages 1–8. IEEE Press.

Sanghamitra Dutta, Gauri Joshi, Soumyadip Ghosh, Parijat Dube, and Priya Nagpurkar. 2018. Slow and stale gradients can win the race: Error-runtime trade-offs in distributed sgd. In *International Conference on Artificial Intelligence and Statistics*, pages 803–812.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. 2017. Accurate, large minibatch sgd: training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. 2018. Marian: Fast neural machine translation in C++. In *Proceedings of ACL 2018, System Demonstrations*, pages 116–121, Melbourne, Australia. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *3rd International Conference for Learning Representations*.

J. Lei Ba, J. R. Kiros, and G. E. Hinton. 2016. Layer Normalization. *ArXiv e-prints*.

Yujun Lin, Song Han, Huizi Mao, Yu Wang, and Bill Dally. 2018. Deep gradient compression: Reducing the communication bandwidth for distributed training. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282.

Brendan McMahan and Matthew Streeter. 2014. Delay-tolerant algorithms for asynchronous distributed online learning. In *Advances in Neural Information Processing Systems*, pages 2915–2923.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. 2018. Scaling neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 1–9.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evalution of machine translation. In *Proceedings 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, PA.

Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu. 2014. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns. In *Fifteenth Annual Conference of the International Speech Communication Association*.

Rico Sennrich, Alexandra Birch, Anna Currey, Ulrich Germann, Barry Haddow, Kenneth Heafield, Antonio Valerio Miceli Barone, and Philip Williams. 2017. The University of Edinburgh's neural mt systems for WMT17. In *Proceedings of the Second Conference on Machine Translation*, pages 389–399.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Edinburgh neural machine translation systems for WMT 16. In *Proceedings of the ACL 2016 First Conference on Machine Translation (WMT16)*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016c. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1715–1725.

Nikko Strom. 2015. Scalable distributed DNN training using commodity GPU cloud computing. In *INTERSPEECH*, volume 7, page 10.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Wei Wen, Cong Xu, Feng Yan, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. 2017. Terngrad: Ternary gradients to reduce communication in distributed deep learning. In *Advances in neural information processing systems*, pages 1509–1519.

# Small and Practical BERT Models for Sequence Labeling

**Henry Tsai, Jason Riesa, Melvin Johnson, Naveen Arivazhagan, Xin Li and Amelia Archer**

Google Research
San Francisco, CA, USA
{henrytsai,riesa,melvinp,navari,xinl,ameliaarcher}@google.com

## Abstract

We propose a practical scheme to train a single multilingual sequence labeling model that yields state of the art results and is small and fast enough to run on a single CPU. Starting from a public multilingual BERT checkpoint, our final model is 6x smaller and 27x faster, and has higher accuracy than a state-of-the-art multilingual baseline. We show that our model especially outperforms on low-resource languages, and works on codemixed input text without being explicitly trained on codemixed examples. We showcase the effectiveness of our method by reporting on part-of-speech tagging and morphological prediction on 70 treebanks and 48 languages.

## 1 Introduction

There have been many recent modeling improvements (Smith et al., 2018; Bohnet et al., 2018) on morphosyntactic tagging tasks. However, these models have largely focused on building separate models for each language or for a small group of related languages. In this paper, we consider the implications of training, evaluating, and deploying a single multilingual model for a diverse set of almost 50 languages, evaluating on both part-of-speech tagging and morphological attribute prediction data from the Universal Dependencies repository (Nivre et al., 2018).

There are several benefits of using one multilingual model over several language specific models.

- Parameter sharing among languages reduces model size and enables cross-lingual transfer learning. We show that this improves accuracy, especially for low-resource languages.
- No language identification model needed to decide which language-specific model to query. Critically, this reduces system complexity and prevents prediction errors from language identification from propagating into the downstream system.
- Multilingual models can be applied to multilingual or codemixed inputs without explicitly being trained on codemixed labeled examples. Otherwise, given, e.g. a mixed Hindi/English input, one must decide to query either the Hindi model or the English model, both of which are sub-optimal.

In this paper, we show that by finetuning a pretrained BERT (Devlin et al., 2019) model we can build a multilingual model that has comparable or better accuracy to state-of-the-art language-specific models, and outperforms the state-of-the-art on low-resource languages. Our model outperforms other multilingual model baselines by a large margin. We evaluate on both part-of-speech tagging and morphological attribute prediction tasks with data from the Universal Dependencies repository (Nivre et al., 2018). However, this model is slow, very large, and difficult to deploy in practice.

We describe our solution for making this model small and practical enough to use in practice on a single CPU, while preserving quality. The final model is 27x faster than a BERT-based baseline model and 7x faster than a state-of-the-art LSTM-based model on CPU. It is 6 times smaller than the BERT-based model. Furthermore, most of the quality gains are preserved in the small model.

## 2 Multilingual Models for Sequence Labeling

We discuss two core models for addressing sequence labeling problems and describe, for each, training them in a single-model multilingual setting: (1) the Meta-LSTM (Bohnet et al., 2018), an extremely strong baseline for our tasks, and (2)



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3632–3636, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| Model | Multilingual? | Part-of-Speech F1 | Morphology F1 |
|---|---|---|---|
| Meta-LSTM | No | 94.5 | 92.5 |
| BERT | No | **95.1** | **93.0** |
| Meta-LSTM | Yes | 91.1 | 82.9 |
| BERT | Yes | **94.5** | **91.0** |

Table 1: Macro-averaged F1 comparison of per-language models and multilingual models over 48 languages. For non-multilingual models, F1 is the average over each per-language model trained.

a multilingual BERT-based model (Devlin et al., 2019).

### 2.1 Meta-LSTM

The Meta-LSTM is the best-performing model of the CoNLL 2018 Shared Task (Smith et al., 2018) for universal part-of-speech tagging and morphological features. The model is composed of 3 LSTMs: a character-BiLSTM, a word-BiLSTM and a single joint BiLSTM which takes the output of the character and word-BiLSTMs as input. The entire model structure is referred to as Meta-LSTM.

To set up multilingual Meta-LSTM training, we take the union of all the word embeddings from the Bojanowski et al. (2017) embeddings model on Wikipedia[1] in all languages. For out-of-vocabulary words, a special unknown token is used in place of the word.

The model is then trained as usual with cross-entropy loss. The char-BiLSTM and word-biLSTM are first trained independently. And finally we train the entire Meta-LSTM.

### 2.2 Multilingual BERT

BERT is a transformer-based model (Vaswani et al., 2017) pretrained with a masked-LM task on millions of words of text. In this paper our BERT-based experiments make use of the cased multilingual BERT model available on GitHub[2] and pretrained on 104 languages.

Models fine-tuned on top of BERT models achieve state-of-the-art results on a variety of benchmark and real-world tasks.

To train a multilingual BERT model for our sequence prediction tasks, we add a softmax layer on top of the the first wordpiece (Schuster and Nakajima, 2012) of each token[3] and finetune on data from all languages combined. During training, we concatenate examples from all treebanks and randomly shuffle the examples.

[1] https://fasttext.cc/docs/en/pretrained-vectors.html

[2] https://github.com/google-research/bert/blob/master/multilingual.md

[3] We experimented with wordpiece-pooling (Lee et al., 2017) which we found to marginally improve accuracy but at a cost of increasing implementation complexity to maintain.

| Model | Embedding Size | Tokens | Hidden Units | Layers |
|---|---|---|---|---|
| Meta-LSTM | 300 | 1.2M | 8M | 3 |
| BERT | 768 | 120k | 87M | 12 |
| MiniBERT | 256 | 120k | 2M | 3 |

Table 2: The number of parameters of each model. *Tokens* refers to the number of tokens of the embedding rows. For the Meta-LSTM, a word-based model, this is the number of words in training. For BERT, this means the size of the Wordpiece vocabulary. And *Hidden Units* refers to all units that are not among the embedding layer or and output layer.

| Input | 32 words | 128 words |
|---|---|---|
| *Relative Speedup on GPU* | | |
| Meta-LSTM | 0.8x | 0.2x |
| MiniBERT | **4.3x** | **2.6x** |
| *Relative Speedup on CPU* | | |
| Meta-LSTM | 6.8x | 2.3x |
| MiniBERT | **27.7x** | **14.0x** |

Table 3: Relative inference speedup over BERT. We see MiniBERT is the fastest on both CPU and GPU. CPU is an Intel Xeon CPU E5-1650 v3 @3.50GHz. GPU is an Nvidia Titan V.

## 3 Small and Practical Models

The results in Table 1 make it clear that the BERT-based model for each task is a solid win over a Meta-LSTM model in both the per-language and multilingual settings. However, the number of parameters of the BERT model is very large (179M parameters), making deploying memory intensive and inference slow: 230ms on an Intel Xeon CPU. Our goal is to produce a model fast enough to run on a single CPU while maintaining the modeling capability of the large model on our tasks.

#### Size and speed

We choose a three-layer BERT, we call *MiniBERT*, that has the same number of layers as the Meta-

LSTM and has fewer embedding parameters and hidden units than both models. Table 2 shows the parameters of each model. The Meta-LSTM has the largest number of parameters dominated by the large embeddings. BERT's parameters are mostly in the hidden units. The MiniBERT has the fewest total parameters.

The inference-speed bottleneck for Meta-LSTM is the sequential character-LSTM-unrolling and for BERT is the large feedforward layers and attention computation that has time complexity quadratic to the sequence length. Table 3 compares the model speeds.

BERT is much slower than both MetaLSTM and MiniBERT on CPU. However, it is faster than Meta-LSTM on GPU due to the parallel computation of the transformer. The MiniBERT is significantly faster than the other models on both GPU and CPU.

**Distillation**

For model distillation (Hinton et al., 2015), we extract sentences from Wikipedia in languages for which public multilingual is pretrained. For each sentence, we use the open-source BERT wordpiece tokenizer (Schuster and Nakajima, 2012; Devlin et al., 2019) and compute cross-entropy loss for each wordpiece:

$$L(\mathbf{t}, \mathbf{s}) = H(\sigma(\mathbf{t}/T), \sigma(\mathbf{s}/T))$$

where $H$ is the cross-entropy function, $\sigma$ is the softmax function, $\mathbf{t}$ is the BERT model's logit of the current wordpiece, $\mathbf{s}$ is the small BERT model's logits and $T$ is a temperature hyperparameter, explained in Section 4.2.

To train the distilled multilingual model *m*MiniBERT, we first use the distillation loss above to train the student from scratch using the teacher's logits on unlabeled data. Afterwards, we finetune the student model on the labeled data the teacher is trained on.

# 4 Experiments

## 4.1 Data

We use universal part-of-speech tagging and morphology data from the The CoNLL 2018 Shared Task (Nivre et al., 2018; Zeman and Hajič, 2018). For comparison simplicity, we remove the languages that the multilingual BERT public checkpoint is not pretrained on.

| Model | Part-of-Speech F1 | Morphology F1 |
|---|---|---|
| *m*Meta-LSTM | 91.1 | 82.9 |
| *m*MiniBERT | 93.7 | 88.6 |
| *m*BERT | **94.5** | **91.0** |

Table 4: Macro-averaged F1 comparison of multilingual models. Multilingual models are prefixed with '*m*'.

For segmentation, we use a baseline segmenter (UDPipe v2.2)[4] provided by the shared task organizer to segment raw text. We train and tune the models on gold-segmented data and apply the segmenter on the raw test of test data before applying our models.

The part-of-speech tagging task has 17 labels for all languages. For morphology, we treat each morphological group as a class and union all classes as a output of 18334 labels.

## 4.2 Tuning

For Meta-LSTM, we use the public repository's hyperparameters[5].

Following Devlin et al. (2019), we use a smaller learning rate of 3e-5 for fine-tuning and a larger learning rate of 1e-4 when training from scratch and during distillation. Training batch size is set to 16 for finetuning and 256 for distillation.

For distillation, we try temperatures $T = 1, 2, 3$ and use the teacher-student accuracy for evaluation. We observe BERT is very confident on its predictions, and using a large temperature $T = 3$ to soften the distribution consistently yields the best result.

## 4.3 Multilingual Models

**Multilingual Modeling Results** We compare per-language models trained on single language treebanks with multilingual models in Table 1 and Table 4. In the experimental results we use a prefix $m$ to denote the model is a single multilingual model. We compare Meta-LSTM, BERT, and MiniBERT.

**Multilingual Models Comparison** *m*BERT performs the best among all multilingual models. The smallest and fastest model, *m*MiniBERT, performs comparably to *m*BERT, and outperforms *m*Meta-LSTM, a state-of-the-art model for this task.

[4]https://ufal.mff.cuni.cz/udpipe/models
[5]https://github.com/google/meta_tagger

| Languages | POS Tagging kk | hy | lt | be | mr | ta | Morphology kk | hy | lt | be | mr | ta |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Train Size | 31 | 50 | 153 | 260 | 373 | 400 | 31 | 50 | 153 | 260 | 373 | 400 |
| Meta-LSTM | 61.7 | 75.4 | 81.4 | 91.1 | 72.1 | 72.7 | 48.5 | 54.5 | 69.7 | 74.0 | 59.1 | 71.0 |
| BERT | 75.9 | 84.4 | 88.9 | 94.8 | **77.5** | **75.7** | 47.8 | 44.8 | **75.2** | 82.8 | 64.0 | 72.9 |
| *m*BERT | **81.4** | **86.6** | **90.0** | **95.0** | 75.9 | 74.3 | **64.6** | **51.1** | 73.6 | **87.5** | **64.2** | **73.8** |
| *m*Meta-LSTM | 52.9 | 63.8 | 65.6 | 87.6 | 65.5 | 61.5 | 25.6 | 36.6 | 42.5 | 59.2 | 33.6 | 46.9 |
| *m*MiniBERT | **76.6** | **86.0** | **86.9** | **95.0** | **75.4** | **74.6** | **59.7** | **47.6** | **64.8** | **81.6** | **59.4** | **71.7** |

Table 5: POS tagging and Morphology F1 for all models on low-resource languages. Multilingual models are prefixed with '*m*'.

**Multilingual Models vs Per-Language Models** When comparing with per-language models, the multilingual models have lower F1. Kondratyuk (2019) shows similar results. Meta-LSTM, when trained in a multilingual fashion, has bigger drops than BERT in general. Most of the Meta-LSTM drop is due to the character-LSTM, which drops by more than 4 points F1.

### 4.4 Low Resource Languages

We pick languages with fewer than 500 training examples to investigate the performance of low-resource languages: Tamil (ta), Marathi (mr), Belarusian (be), Lithuanian (lt), Armenian (hy), Kazakh (kk)[6]. Table 5 shows the performance of the models.

**BERT Cross-Lingual Transfer** While Wu and Dredze (2019) shows effective zero-shot crosslingual transfer from English to other high-resource languages, we show that cross-lingual transfer is even effective on low-resource languages when we train on all languages as *m*BERT is significantly better than BERT when we have fewer than 50 examples. In these cases, the *m*MiniBERT distilled from the multilingual *m*BERT yields results better than training individual BERT models. The gains becomes less significant when we have more training data.

***m*MiniBERT Effectiveness** The multilingual baseline *m*Meta-LSTM does not do well on low-resource languages. On the contrary, *m*MiniBERT performs well and outperforms the state-of-the-art Meta-LSTM on the POS tagging task and on four out of size languages of the Morphology task.

[6]The Universal Dependencies data does not have explicit tuning data for hy and kk.

| Model | F1 |
|---|---|
| BERT Supervised | 90.6 |
| Meta-LSTM English-Only | 47.7 |
| Meta-LSTM Hindi-Only | 53.8 |
| *m*Meta-LSTM | **83.4** |
| *m*BERT | 82.9 |
| *m*MiniBERT | 79.5 |

Table 6: F1 score on Hindi-English codemixed POS tagging task. Each multilingual model is within 10 points of the supervised BERT model without having explicitly seen code-mixed data.

### 4.5 Codemixed Input

We use the Universal Dependencies' Hindi-English codemixed data set (Bhat et al., 2017) to test the model's ability to label code-mixed data. This dataset is based on code-switching tweets of Hindi and English multilingual speakers. We use the Devanagari script provided by the data set as input tokens.

In the Universal Dependency labeling guidelines, code-switched or foreign-word tokens are labeled as X along with other tokens that cannot be labeled[7]. The trained model learns to partition the languages in a codemixed input by labeling tokens in one language with X, and tokens in the other language with any of the other POS tags. It turns out that the 2nd-most likely label is usually the correct label in this case; we evaluate on this label when the 1-best is X.

Table 6 shows that all multilingual models handle codemixed data reasonably well without supervised codemixed traininig data.

## 5 Conclusion

We have described the benefits of multilingual models over models trained on a single language for a single task, and have shown that it is possible to resolve a major concern of deploying large

[7]https://universaldependencies.org/u/pos/X.html

BERT-based models by distilling our multilingual model into one that maintains the quality wins with performance fast enough to run on a single CPU. Our distilled model outperforms a multilingual version of a very strong baseline model, and for most languages yields comparable or better performance to a large BERT model.

## References


Irshad Bhat, Riyaz A Bhat, Manish Shrivastava, and Dipti Sharma. 2017. Joining hands: Exploiting monolingual treebanks for parsing of code-mixing data. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 324–330.

Bernd Bohnet, Ryan McDonald, Gonçalo Simões, Daniel Andor, Emily Pitler, and Joshua Maynez. 2018. Morphosyntactic tagging with a meta-BiLSTM model over context sensitive token encodings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2642–2652, Melbourne, Australia. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Daniel Kondratyuk. 2019. 75 languages, 1 model: Parsing universal dependencies universally. *CoRR*, abs/1904.02099.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Joakim Nivre, Mitchell Abrams, Željko Agić, Lars Ahrenberg, and Lene Antonsen et al. 2018. Universal dependencies 2.2. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152. IEEE.

Aaron Smith, Bernd Bohnet, Miryam de Lhoneux, Joakim Nivre, Yan Shao, and Sara Stymne. 2018. 82 treebanks, 34 models: Universal dependency parsing with multi-treebank models. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 113–123, Brussels, Belgium. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Shijie Wu and Mark Dredze. 2019. Beto, bentz, becas: The surprising cross-lingual effectiveness of BERT. *CoRR*, abs/1904.09077.

Daniel Zeman and Jan Hajič, editors. 2018. *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics, Brussels, Belgium.

# Data Augmentation with Atomic Templates for Spoken Language Understanding

**Zijian Zhao**[*], **Su Zhu**[*] and **Kai Yu**[†]
MoE Key Lab of Artificial Intelligence
SpeechLab, Department of Computer Science and Engineering
Shanghai Jiao Tong University, Shanghai, China
{1248uu,paul2204,kai.yu}@sjtu.edu.cn

## Abstract

Spoken Language Understanding (SLU) converts user utterances into structured semantic representations. Data sparsity is one of the main obstacles of SLU due to the high cost of human annotation, especially when domain changes or a new domain comes. In this work, we propose a data augmentation method with atomic templates for SLU, which involves minimum human efforts. The atomic templates produce exemplars for fine-grained constituents of semantic representations. We propose an encoder-decoder model to generate the whole utterance from atomic exemplars. Moreover, the generator could be transferred from source domains to help a new domain which has little data. Experimental results show that our method achieves significant improvements on DSTC 2&3 dataset which is a domain adaptation setting of SLU.

## 1 Introduction

The SLU module is a key component of spoken dialogue system, parsing user utterances into corresponding semantic representations in a narrow domain. The typical semantic representation for SLU could be semantic frame (Tur and De Mori, 2011) or dialogue act (Young, 2007). In this paper, we focus on SLU with the dialogue act that a sentence is labelled as a set of *act-slot-value* triples. For example, the utterance *"Not Chinese but I want Thai food please"* has an annotation of *"deny(food=Chinese), inform(food=Thai)"*.

Deep learning has achieved great success in the SLU field (Mesnil et al., 2015; Liu and Lane, 2016; Zhao et al., 2019). However, it is notorious for requiring large labelled data, which limits the scalability of SLU models. Despite recent advancements and tremendous research activity in semi-supervised learning and domain adaptation, the deep SLU models still require massive amounts of labelled data to train.

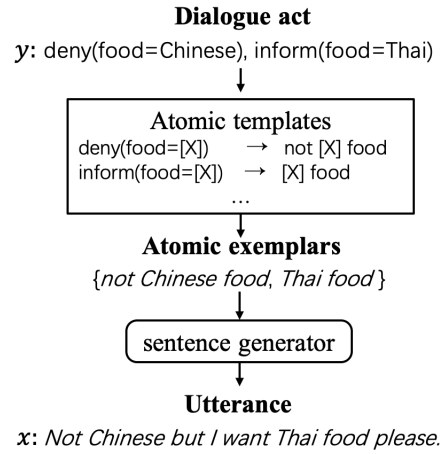


Figure 1: Workflow of the data augmentation with atomic templates for SLU.

Therefore, data augmentation for SLU becomes appealing, and it needs three kinds of capabilities:

- **Expression diversity**: There are always various expressions for the same semantic meaning. Lack of expression diversity is an obstacle of SLU.
- **Semantic diversity**: The data augmentation method should generate data samples with various semantic meanings.
- **Domain adaptation**: The method should be able to utilize data from other domains and adapt itself to a new domain rapidly.

In this work, we propose a new data augmentation method for SLU, which consists of atomic templates and a sentence generator (as shown in Figure 1). The method starts from dialogue acts

[*] Zijian Zhao and Su Zhu are co-first authors and contribute equally to this work.
[†] The corresponding author is Kai Yu.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3637–3643,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

which are well-structured and predefined by domain experts. Rich and varied dialogue acts can be created automatically, which is guided by a domain ontology.

To enhance the capability of domain adaptation for the sentence generator, we propose to first interpret dialogue acts in natural language so that new dialogue acts can be understood by the generator. However, interpreting a dialogue act at sentence level costs a lot, atomic templates are exploited to alleviate human efforts. The atomic templates are created at phrase level, which produce exemplars for fine-grained constituents (i.e. *act-slot-value* triple) of semantic representations (i.e. dialogue act). Thus, the sentence generator can be an encoder-decoder model, paraphrasing a set of atomic exemplars to an utterance.

We evaluate our approach on DSTC 2&3 dataset (Henderson et al., 2014a,b), a benchmark of SLU including domain adaptation setting. The results show that our method can obtain significant improvements over strong baselines with limited training data.

## 2 Related Work

**Data augmentation for SLU** Hou et al. (2018) use a sequence-to-sequence model to translate between sentence pairs with the same meaning. However, it cannot generate data for new semantic representations. Yoo et al. (2018) use a variational autoencoder to generate labelled language. It still lacks variety of semantic meanings. Let $x$ and $y$ denote an input sentence and the corresponding semantic representation respectively. These two works estimate $p(x, y)$ with generation models to produce more labelled samples, while they have two drawbacks. 1) The models cannot control which kinds of semantic meaning should be generated for supply. 2) The models cannot generate data for new semantic representations which may contain out-of-vocabulary (OOV) labels. We propose to generate utterance based on semantic representations by estimating $p(x|y)$, because $y$ is well-structured and easy to be synthesized. To overcome the OOV problem of semantic labels, we first map them to atomic exemplars with a little of human effort.

**Zero-shot learning of SLU** Besides data augmentation, zero-shot learning of SLU (Ferreira et al., 2015; Yazdani and Henderson, 2015) is also related, which can adapt to unseen semantic labels. Yazdani and Henderson (2015) exploit a binary classifier for each possible *act-slot-value* triple to predict whether it exists in the input sentence. Zhao et al. (2019) propose a hierarchical decoding model for SLU. However, they still have problem with new *act* and *slot*. Bapna et al. (2017); Lee and Jha (2018); Zhu and Yu (2018) try to solve it with textual slot descriptions. In this paper, we propose atomic templates to describe *act-slot-value* triples but not separate slots or acts.

## 3 Method

In this section, our data augmentation method with atomic templates will be introduced, which generates additional data samples for training SLU model.

### 3.1 SLU Model

For dialogue act based SLU, we adopt the hierarchical decoding (HD) model (Zhao et al., 2019) which dynamically parses *act*, *slot* and *value* in a structured way. The model consists of four parts:

- A shared utterance encoder, bidirectional LSTM recurrent neural network (BLSTM) (Graves, 2012);
- An *act* type classifier on the utterance;
- A *slot* type classifier with the utterance and an act type as inputs;
- A *value* decoder with the utterance and an *act-slot* type pair as inputs.

The *value* decoder generates word sequence of the value by utilizing a sequence-to-sequence model with attention (Luong et al., 2015) and pointer network (Vinyals et al., 2015a) which helps handling out-of-vocabulary (OOV) values.

### 3.2 Data Generation with Atomic Templates

As illustrated in Figure 1, the workflow of the data augmentation is broken down into: 1) mapping *act-slot-value* triples to exemplars with atomic templates, and 2) generating the corresponding utterance depending on the atomic exemplars.

Let $x = x_1 \cdots x_{|x|}$ denote the utterance (word sequence), and $y = \{y_1, \cdots, y_{|y|}\}$ denote the dialogue act (a set of *act-slot-value* triples). We wish to estimate $p(x|y)$, the conditional probability of utterance $x$ given dialogue act $y$.

However, there are also some disadvantages to directly using dialogue acts as inputs:

| Triple | Templates |
|---|---|
| bye() | goodbye<br>bye |
| request(addr) | the address<br>what's the address |
| inform(food=[food]) | [food]<br>[food] food |
| inform(hastv=true) | television<br>with a television |

Table 1: Examples of atomic templates in DSTC 2&3 dataset. "[food]" is an arbitrary value of slot "food".

- The dialogue acts from different narrow domains may conflict, e.g. different slot names for the same meaning.
- Act and slot types may not be defined in a literal way, e.g. using arbitrary symbols like "city_1" and "city_2" to represent city names in different contexts.

Thus, it is hard to adapt the model $p(x|y)$ to new act types, slot types, and domains.

We propose to interpret the dialogue act in short natural language, and then rephrase it to the corresponding user utterance. While interpreting the dialogue act at sentence level costs as much as building a rule-based SLU system, we choose to interpret *act-slot-value* triples with atomic templates which involve minimum human efforts.

#### 3.2.1 Atomic templates

Table 1 gives some examples of atomic templates used in DSTC 2&3 dataset. Atomic templates produce a simple description (atomic exemplar $e_i$) in natural language for each *act-slot-value* triple[1] $y_i$. If there are multiple templates for triple $y_i$, which generate multiple atomic exemplars $E(y_i)$, we choose the most similar one $e_i = \text{argmax}_{e \in E(y_i)} sim(e, x)$ in the training stage, and randomly select one $e_i$ from $E(y_i)$ in the data augmentation stage. The similarity function $sim(e, x)$ we used is Ratcliff-Obershelp algorithm (Black, 2004). Therefore,

$$p(x|y) = p(x|\{y_1 \cdots y_{|y|}\}) = p(x|\{e_1 \cdots e_{|y|}\})$$

#### 3.2.2 Sentence generator

An encoder-decoder model is exploited to generate the utterance based on the set of atomic exemplars by estimating $p(x|\{e_1 \cdots e_{|y|}\})$.

[1]It should be noted that an *act-slot-value* triple may have an empty value, e.g. "request(phone)" refers to asking for phone number. The triple may also need no slot, e.g. "bye()".

As the set of atomic exemplars is unordered, we encode them independently. For each atomic exemplar $e_i = w_{i1} \cdots w_{iT_i}$ (a sequence of words with length $T_i$), we use a BLSTM to encode it. The hidden vectors are recursively computed at the $j$-th time step via:

$$\begin{aligned}
\overrightarrow{\mathbf{h}}_{ij} &= \text{f}_{\text{LSTM}}(\psi(w_{ij}), \overrightarrow{\mathbf{h}}_{ij-1}), j = 1, \cdots, |T_i| \\
\overleftarrow{\mathbf{h}}_{ij} &= \text{b}_{\text{LSTM}}(\psi(w_{ij}), \overleftarrow{\mathbf{h}}_{ij+1}), j = |T_i|, \cdots, 1 \\
\mathbf{h}_{ij} &= [\overrightarrow{\mathbf{h}}_{ij}; \overleftarrow{\mathbf{h}}_{ij}]
\end{aligned}$$

where $[\cdot;\cdot]$ denotes vector concatenation, $\psi(\cdot)$ is a word embedding function, $\text{f}_{\text{LSTM}}$ is the forward LSTM function and $\text{b}_{\text{LSTM}}$ is the backward one. The final hidden vectors of the forward and backward passes are utilized to represent each atomic exemplar $e_i$, i.e. $\mathbf{c}_i = [\overrightarrow{\mathbf{h}}_{iT_i}; \overleftarrow{\mathbf{h}}_{i1}]$.

After encoding all the atomic exemplars, we have a list of hidden vectors $H(y) = [\mathbf{h}_{11} \cdots \mathbf{h}_{1T_1}; \cdots; \mathbf{h}_{|y|1} \cdots \mathbf{h}_{|y|T_{|y|}}]$. A LSTM model serves as the decoder (Vinyals et al., 2015b), generating the utterance $x$ word-by-word:

$$p(x|y) = p(x|H(y)) = \prod_{t=1}^{|x|} p(x_t|x_{<t}, H(y))$$

Before the generation, the hidden vector of the decoder is initialized as $1/|y| \sum_{i=1}^{|y|} \mathbf{c}_i$, the mean of representations of all the atomic exemplars. The pointer softmax (Gulcehre et al., 2016) enhanced by a trick of targeted feature dropout (Xu and Hu, 2018) is adopted to tackle OOV words, which will switch between generation and copying from the input source dynamically.

## 4 Experiment

### 4.1 Data

In our experiments, we use the dataset provided for the second and third Dialog State Tracking Challenge (DSTC 2&3) (Henderson et al., 2014a,b). DSTC2 contains a large number of training dialogues (∼15.6k utterances) related to restaurant search while DSTC3 is designed to address the problem of adaptation to a new domain (tourist information) with only a small amount of seed data (*dstc3_seed*, 109 utterances). The manual transcriptions are used as user utterances to eliminate the impact of speech recognition errors.

We follow the data partitioning policy as Zhu et al. (2014), which randomly selects one-half of

DSTC3 test data as the oracle training set (∼9.4k utterances) and leaves the other half as the evaluation set (∼9.2k utterances).

### 4.2 Experimental Setup

Delexicalized triples are used in case of non-enumerable slot, like "inform(food=[food])" in Table 1. There are 41 and 35 delexicalized triples for DSTC2 and DSTC3 respectively. For each triple, we prepare two short templates on average. Note that, compared to designing sentence-level templates, writing atomic templates for triples at phrase level requires much less human efforts.

For data augmentation of DSTC3, there are two ways to collect extra dialogue acts as inputs[2]:

- **Seed abridgement**: Abridging the delexicalized dialogue acts existing in *dstc3_seed* data, e.g. a dialogue act with 3 triples {A,B,C} can be abridged to obtain {A,B}, {B,C}, {A,C}, {A}, {B}, {C} and itself.

- **Combination**: As all possible triples are predefined in the domain ontology of DSTC3 by experts. We use a general policy of triples combination, which randomly selects at most $N_c$ triples to make up a dialogue act. ($N_c$ is set as 3 empirically.)

Then we fill up each non-enumerable slot in the dialogue act by randomly choosing a value of this slot, which ends when each value appears at least $N_v$ times in all the collected dialogue acts. $N_v$ is set as 3 empirically. After that, we have 1420 and 20670 dialogue acts from the seed abridgement and combination respectively. New data samples are generated starting from these dialogue acts, through the atomic templates and the sentence generator (e.g. 1-best output is kept).

The SLU model and sentence generator use the same hyper-parameters. The dimension of word embeddings (*Glove*[3] word vectors are used for initialization) is 100 and the number of hidden units is 128. Dropout rate is 0.5 and batch size is 20. Maximum norm for gradient clipping is set to 5 and Adam optimizer is used with an initial learning rate of 0.001. All training consists of 50 epochs with early stopping on the development set. We report F1-score of extracting *act-slot-value* triples by the official scoring script from http://camdial.org/~mh521/dstc/.

### 4.3 Systems

We first compare two SLU models to answer why the HD model is chosen for dialogue act based SLU:

- **ZS**: zero-shot learning of SLU (Yazdani and Henderson, 2015) which can adapt to unseen dialogue acts.

- **HD**: the hierarchical decoding model (Zhao et al., 2019) is adopted in our system.

We make comparisons of other data augmentation methods and the atomic templates (AT):

- **Naive**: We replace the value simultaneously existing in an utterance and its semantic labels of *dstc3_seed* by randomly selecting a value of the corresponding slot. It ends when each value appears at least $N_v$ times.

- **Human**: Zhu et al. (2014) proposed to design a large number of sentence-level templates for DSTC3 with lots of human efforts.

- **Oracle**: The oracle training set is used which simulates the perfect data augmentation.

Without data augmentation, the SLU models are pre-trained on DSTC2 dataset (source domain) and finetuned with *dstc3_seed* set. In our data augmentation method, the sentence generator based on atomic concepts is also pre-trained on DSTC2 dataset and finetuned with the *dstc3_seed*. The SLU model is first pre-trained on DSTC2 dataset, then finetuned with the augmented dataset and finally finetuned with the *dstc3_seed*.

### 4.4 Results and Analysis

The main results are illustrated in Table 2. We can see that:

1. The hierarchical decoding (HD) model gets better performance than the zero-shot learning (ZS) method of SLU.

2. The seed data $dstc3\_seed$ limits the power of the SLU model, and even the naive augmentation can enhance it.

[2]The data splits, atomic templates and extra dialogue acts will be available at https://github.com/sz128/DAAT_SLU

[3]http://nlp.stanford.edu/data/glove.6B.zip

| SLU | Augmentation | F1-score |
|---|---|---|
| ZS | w/o | 68.3 |
| HD | w/o | 78.5 |
| | Naive | 82.9 |
| | AT (seed abridgement) | 85.5 |
| | AT (combination) | 87.9 |
| | AT (seed abr. + comb.) | **88.6** |
| | Human (Zhu et al., 2014) | 90.4 |
| | Oracle | 96.9 |

Table 2: SLU performances of different systems on the DSTC3 evaluation set.

3. Our data augmentation method with atomic templates (AT) improves the SLU performance dramatically. One reason may be the generated data has higher variety of semantic meaning than the naive augmentation. *Combination* can make up more dialogue acts and shows better result than *Seed abridgement*, while *Seed abridgement* provides more realistic dialogue acts. Thus, their union gives the best result.

4. The best performance of our method is close to the human-designed sentence-level templates (Zhu et al., 2014), while our approach needs much less human efforts.

**Ablation Study** We conduct several ablation studies to analyze the effectiveness of different components of our method, as shown in Table 3. By removing SLU model pretraining on DSTC2 ("- *dstc2*") and finetuning on the seed data ("- *dstc3_seed*"), we can see a significant decrease in SLU performance. When we subsequently cast aside the sentence generator ("- *sentence generator*", i.e. using the atomic exemplars as inputs of SLU model directly), the SLU performance decreases by 10.3%. This shows that the sentence generator can produce more natural utterances. If we replace the atomic exemplars as the corresponding *act-slot-value* triples ("- *atomic templates*"), the SLU performance drops sharply. The reason may be that the atomic templates provide a better description of corresponding semantic meanings than the surfaces of the triples. Examples of generated data samples are in Appendix A.

**Number of seed samples** Figure 2 shows the number of seed samples used versus SLU performance on DSTC3 evaluation set. For zero-shot case (no seed samples), our method is much better than the baseline. When the number of seed samples increases, our method outperforms the baseline constantly.

| Method | F1-score |
|---|---|
| HD + AT(seed abr. + comb.) | 88.6 |
| - dstc2 | 86.2 |
| - dstc3_seed | 84.5 |
| - dstc2, - dstc3_seed | 84.3 |
| - sentence generator | 74.0 |
| - atomic templates | 70.5 |

Table 3: SLU performances on the DSTC3 evaluation set when removing different modules of our method.

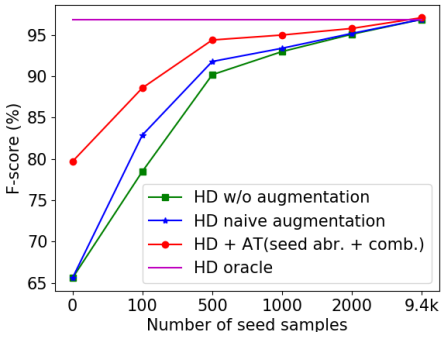


Figure 2: SLU performance of different methods with varying number of seed samples randomly selected from the oracle DSTC3 training set.

## 5 Conclusion

In this paper, we propose a new data augmentation method with atomic templates for SLU. The atomic templates provide exemplars in natural language for *act-slot-value* triples, involving minimum human efforts. An encoder-decoder model is exploited to generate utterances based on the atomic exemplars. We believe our method can also be applied to SLU systems with other semantic representations (e.g. semantic frame). Experimental results show that our method achieves significant improvements on DSTC 2&3 dataset, and it is is very effective for SLU domain adaptation with limited data.

## Acknowledgments

This work has been supported by the National Key Research and Development Program of China (Grant No. 2017YFB1002102) and the China NSFC projects (No. 61573241). We thank the anonymous reviewers for their thoughtful comments and efforts towards improving this manuscript.

## References


Ankur Bapna, Gökhan Tür, Dilek Hakkani-Tür, and Larry P. Heck. 2017. Towards zero-shot frame semantic parsing for domain scaling. In *18th Annual Conference of the International Speech Communication Association*, pages 2476–2480.

Paul E. Black. 2004. Ratcliff/Obershelp pattern recognition. In *Dictionary of Algorithms and Data Structures* [online], Paul E. Black, ed. Available from: https://www.nist.gov/dads/HTML/ratcliffObershelp.html (accessed 20 May 2019).

Emmanuel Ferreira, Bassam Jabaian, and Fabrice Lefèvre. 2015. Zero-shot semantic parser for spoken language understanding. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Alex Graves. 2012. *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer Berlin Heidelberg.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 140–149.

Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014a. The second dialog state tracking challenge. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 263–272.

Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014b. The third dialog state tracking challenge. In *2014 IEEE Spoken Language Technology Workshop (SLT)*, pages 324–329. IEEE.

Yutai Hou, Yijia Liu, Wanxiang Che, and Ting Liu. 2018. Sequence-to-sequence data augmentation for dialogue language understanding. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1234–1245.

Sungjin Lee and Rahul Jha. 2018. Zero-shot adaptive transfer for conversational language understanding. *arXiv preprint arXiv:1808.10059*.

Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. In *17th Annual Conference of the International Speech Communication Association (InterSpeech)*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, et al. 2015. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):530–539.

Gokhan Tur and Renato De Mori. 2011. *Spoken language understanding: Systems for extracting semantic information from speech*. John Wiley & Sons.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015a. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015b. Grammar as a foreign language. In *Advances in neural information processing systems*, pages 2773–2781.

Puyang Xu and Qi Hu. 2018. An end-to-end approach for handling unknown slot values in dialogue state tracking. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1448–1457.

Majid Yazdani and James Henderson. 2015. A model of zero-shot learning of spoken language understanding. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 244–249.

Kang Min Yoo, Youhyun Shin, and Sang-goo Lee. 2018. Data augmentation for spoken language understanding via joint variational generation. *arXiv preprint arXiv:1809.02305*.

Steve Young. 2007. CUED standard dialogue acts. *Report, Cambridge University Engineering Department, 14th October*.

Zijian Zhao, Su Zhu, and Kai Yu. 2019. A hierarchical decoding model for spoken language understanding from unaligned data. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7305–7309.

Su Zhu, Lu Chen, Kai Sun, Da Zheng, and Kai Yu. 2014. Semantic parser enhancement for dialogue domain extension with little data. In *2014 IEEE Spoken Language Technology Workshop (SLT)*, pages 336–341. IEEE.

Su Zhu and Kai Yu. 2018. Concept transfer learning for adaptive language understanding. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pages 391–399. Association for Computational Linguistics.

## A Examples of generated data

| Dialogue act | Atomic exemplars | Generated utterance |
|---|---|---|
| thankyou<br>bye | thank you<br>good bye | Thank you good bye. |
| request(addr)<br>request(phone) | the address<br>the phone number | The address and phone number. |
| request(addr)<br>request(phone) | whats the address<br>what is the phone number | What is the address and phone number? |
| inform(type=coffeeshop)<br>inform(hasinternet=true) | cafe<br>has internet connection | I'm looking for a cafe has the internet connection. |
| affirm<br>inform(type=restaurant)<br>inform(pricerange=moderate) | yes<br>looking for a restaurant<br>moderately priced | Yes, I'm looking for a moderately priced restaurant. |
| request(childrenallowed) | does it allow children | Does it have children? |
| request(hastv)<br>request(addr) | does it has a television<br>the address | Does it have the television address and address? |

Table 4: Examples of generated data samples for DSTC3. It shows that the generated utterances are associated with the atomic exemplars. For some frequent triples, well-formed utterances can be produced. We can find two bad cases at the end of the table. The reason may be that both "request(childrenallowed)" and "request(hastv);request(addr)" never appear in the *dstc3_seed*, thus the sentence generator doesn't fit well.

# PaLM: A Hybrid Parser and Language Model

**Hao Peng**[♠] **Roy Schwartz**[♠◇] **Noah A. Smith**[♠◇]
[♠]Paul G. Allen School of Computer Science & Engineering, University of Washington
[◇]Allen Institute for Artificial Intelligence
{hapeng,roysch,nasmith}@cs.washington.edu

## Abstract

We present PaLM, a hybrid **pa**rser and neural **l**anguage **m**odel. Building on an RNN language model, PaLM adds an attention layer over *text spans* in the left context. An unsupervised constituency parser can be derived from its attention weights, using a greedy decoding algorithm. We evaluate PaLM on language modeling, and empirically show that it outperforms strong baselines. If syntactic annotations *are* available, the attention component can be trained in a supervised manner, providing syntactically-informed representations of the context, and further improving language modeling performance.

## 1 Introduction

Recent language models have shown very strong data-fitting performance (Jozefowicz et al., 2016; Merity et al., 2018). They offer useful products including, most notably, contextual embeddings (Peters et al., 2018; Radford et al., 2019), which benefit many NLP tasks such as text classification (Howard and Ruder, 2018) and dataset creation (Zellers et al., 2018).

Language models are typically trained on large amounts of raw text, and therefore do not explicitly encode any notion of structural information. Structures in the form of syntactic trees have been shown to benefit both classical NLP models (Gildea and Palmer, 2002; Punyakanok et al., 2008; Das et al., 2012, *inter alia*) and recent state-of-the-art neural models (Dyer et al., 2016; Swayamdipta et al., 2018; Peng et al., 2018b; Strubell et al., 2018, *inter alia*). In this paper we show that LMs can benefit from syntactically-inspired encoding of the context.

We introduce PaLM (**pa**rser and **l**anguage **m**odel; Fig. 1), a novel hybrid model combining an RNN language model with a constituency parser. The LM in PaLM attends over *spans of tokens*, implicitly learning which syntactic constituents are likely. A span-based parser is then derived from the attention information (Stern et al., 2017).

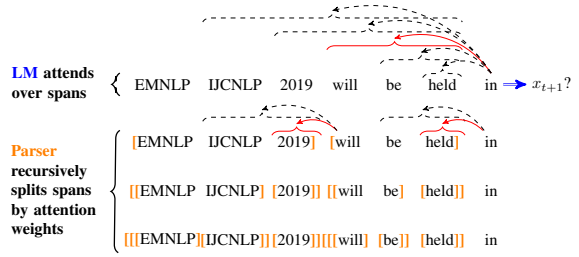


Figure 1: An illustration of PaLM. The LM (first line) predicts the next word ($x_{t+1}$, double blue arrow) by attending over previous spans ending in time $t-1$ (dashed lines). The parser (lines 2–4) splits the prefix into two spans (line 2) by the taking the top scoring attended span (red solid line) and the prefix leading to it. It then recursively splits the two sub-spans using the same procedure (line 3). Finally, spans of length two are trivially split into terminal nodes (line 4).

PaLM has several benefits. First, it is an intuitive and lightweight way of incorporating structural information (§2.1), requiring no marginal inference, which can be computationally expensive (Jelinek and Lafferty, 1991; Chelba and Jelinek, 1998; Roark, 2001; Dyer et al., 2016; Buys and Blunsom, 2018; Kim et al., 2019, *inter alia*). Second, the attention can be syntactically informed, in the sense that the attention component can optionally be supervised using syntactic annotations, either through pretraining or by joint training with the LM (§2.2). Last, PaLM can derive an unsupervised constituency parser (§2.2), whose parameters are estimated purely using the language modeling objective.

To demonstrate the empirical benefits of PaLM, we experiment with language modeling (§3). PaLM outperforms the AWD-LSTM model (Merity et al., 2018) on both the Penn Treebank



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3644–3651, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

(PTB; Marcus et al., 1993) and WikiText-2 (Merity et al., 2017) datasets by small but consistent margins in the unsupervised setup. When the parser is trained jointly with the language model, we see additional perplexity reductions in both cases. Our implementation is available at https://github.com/Noahs-ARK/PaLM.

## 2 PaLM—Parser and Language Model

We describe PaLM in detail. At its core is an attention component, gathering the representations of preceding *spans* at each time step. Similar to self-attention, PaLM can be implemented on top of RNN encoders (Parikh et al., 2016), or as it is (Vaswani et al., 2017). Here we encode the tokens using a left-to-right RNN, denoted with vectors $\mathbf{h}_t$.[1]

Below we describe the span-attention component and the parsing algorithm. We use $[i, j], i \leq j$ to denote text span $x_i \dots x_j$, i.e., inclusive on both sides. When $i = j$, it consists of a single token.

### 2.1 Span Attention

We want the language model attention to gather context information aware of syntactic structures. A constituency parse can be seen as a collection of syntactic constituents, i.e., token spans. Therefore we attend over preceding spans.[2]

At step $t$, PaLM attends over the spans ending at $t-1$, up to a maximum length $m$, i.e., $\{[i, t-1]\}_{i=t-m}^{t-1}$.[3] Essentially, this can be seen as splitting the prefix span $[t-m, t-1]$ into two, and attending over the one on the right. Such a span attention mechanism is inspired by the top-down greedy span parser of Stern et al. (2017), which recursively divides phrases. In §2.2, we will use a similar algorithm to derive a constituency parser from the span attention weights.

**Bidirectional span representation with rational RNNs.** Meaningful span representations are crucial in span-based tasks (Lee et al., 2017; Peng et al., 2018c; Swayamdipta et al., 2018, *inter alia*). Typical design choices are based on start and end token vectors contextualized by bidirectional RNNs. However, a language model does not have access to future words, and hence running a backward RNN from right to left is less straightforward: one will have to start an RNN running at each token, which is computationally daunting (Kong et al., 2016). To compute span representations efficiently, we use *rational* RNNs (RRNNs; Peng et al., 2018a).

RRNNs are a family of RNN models, where the recurrent function can be computed with weighted finite-state automata (WFSAs). We use the unigram WFSA–inspired RRNN (Peng et al., 2018a), where the cell state update is

$$\mathbf{f}_t = \boldsymbol{\sigma}\left(\mathbf{W}_f \mathbf{h}_t\right), \tag{1a}$$

$$\mathbf{u}_t = \left(\mathbf{1} - \mathbf{f}_t\right) \odot \tanh\left(\mathbf{W}_u \mathbf{h}_t\right), \tag{1b}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{u}_t. \tag{1c}$$

$\mathbf{f}_t$ is a forget gate implemented with the elementwise sigmoid function $\boldsymbol{\sigma}$, and $\odot$ denotes elementwise multiplication. $\mathbf{W}_u$ and $\mathbf{W}_f$ are learned matrices. Bias terms are suppressed for clarity.[4]

Slightly overloading the notation, let $\overrightarrow{\mathbf{c}}_{i,j}$ denote the encoding of span $[i, j]$ by running a forward RRNN in Eq. 1, from left to right. It can be efficiently computed by subtracting $\overrightarrow{\mathbf{c}}_{i-1}$ from $\overrightarrow{\mathbf{c}}_j$, weighted by a product of forget gates:

$$\overrightarrow{\mathbf{c}}_{i,j} = \overrightarrow{\mathbf{c}}_j - \overrightarrow{\mathbf{c}}_{i-1} \bigodot_{k=i}^{j} \overrightarrow{\mathbf{f}}_k. \tag{2}$$

$\overrightarrow{\mathbf{f}}_k$ vectors are the forget gates. See §B for a detailed derivation.

Using this observation, we now derive an efficient algorithm to calculate the span representations based on bidirectional RRNNs. In the interest of space, Alg. 1 describes the forward span representations. It takes advantage of the distributivity property of rational RNNs (Peng et al., 2018a), and the number of RNN function calls is linear in the input length.[5] Although overall asymptotic time complexity is still quadratic, Alg. 1 only involves elementwise operations, which can be eas-

[1] We experiment with a strong LSTM implementation for language modeling (Merity et al., 2018), see §3.

[2] Standard token-based self-attention naturally relates to dependency structures through head selection (Strubell et al., 2018). In a left-to-right factored language model, dependencies are less natural if we want to allow a child to precede its parent.

[3] $m$ is set to 20. This reduces the number of considered spans from $\mathcal{O}(n^2)$ to $\mathcal{O}(mn)$. Besides practical concerns, it makes less sense if a phrase goes beyond one single sentence (the average sentence length of WSJ training sentences is 21).

[4] Unlike other RNNs such as LSTM (Hochreiter and Schmidhuber, 1997) or GRU (Cho et al., 2014), RRNNs do not apply an affine transformation or a nonlinear dependency of $\mathbf{c}_t$ on $\mathbf{c}_{t-1}$.

[5] In contrast, the dynamic program of Kong et al. (2016) for segmental (span-scoring) RNNs requires a quadratic number of recurrent function calls, since they use LSTMs, where distributivity does not hold.

**Algorithm 1** RRNN-based span representation.[6]

**procedure** SPANREPR($\{\overrightarrow{\mathbf{c}}_t, \overrightarrow{\mathbf{f}}_t\}$)
  ▷ Accumulate forward forget gates
  **for** $i = 1, \ldots, n$ **do**
    $\overrightarrow{\mathbf{f}}_{1,i} = \overrightarrow{\mathbf{f}}_{1,i-1} \odot \overrightarrow{\mathbf{f}}_i$
  **end for**
  **for** $j = 1, \ldots, n$ **do**
    **for** $i = 1, \ldots, j$ **do**
      $\overrightarrow{\mathbf{c}}_{i,j} = \overrightarrow{\mathbf{c}}_j - \overrightarrow{\mathbf{c}}_{i-1} \odot \overrightarrow{\mathbf{f}}_{1,j} / \overrightarrow{\mathbf{f}}_{1,i-1}$
    **end for**
  **end for**
  **return** $\overrightarrow{\mathbf{c}}_{i,j}$ vectors
**end procedure**

ily parallelized on modern GPUs. The backward one (right to left) is analogous.

**Computing attention.** As in standard attention, we use a normalized weighted sum of the span representations. Let $\mathbf{g}([i,j]) = [\overrightarrow{\mathbf{c}}_{i,j}; \overleftarrow{\mathbf{c}}_{i,j}]$ denote the representation of span $[i,j]$, which concatenates the forward and backward representations calculated using Alg. 1. The context vector $\mathbf{a}_t$ is

$$\mathbf{a}_{t+1} = \sum_{i=0}^{m-1} \omega_{t,i}\ \mathbf{g}([t-i,t]), \tag{3a}$$

$$\omega_{t,i} = \frac{\exp s_{t,i}}{\sum_{j=0}^{m-1} \exp s_{t,j}}. \tag{3b}$$

Here $s_{t,i}$ is implemented as an MLP, taking as input the concatenation of $\mathbf{h}_{t+1}$ and $\mathbf{g}([t-i,t])$ and outputs the attention score. The context vector is then concatenated with the hidden state $\bar{\mathbf{h}}_{t+1} = [\mathbf{h}_{t+1}; \mathbf{a}_{t+1}]$, and fed into onward computation.

In summary, given an input sequence, PaLM:

1. First uses a standard left-to-right RNN to calculate the hidden states $\mathbf{h}_t$.
2. Feed $\mathbf{h}_t$ vectors into a one-layer bidirectional rational RNN (Eq. 1), using Alg. 1 to compute the span representations.
3. Attends over spans (Eq. 3b) to predict the next word.

## 2.2 Attention-Based Constituency Parsing

We next describe the other facet or PaLM: the constituency parser. Our parsing algorithm is similar to the greedy top-down algorithm proposed by Stern et al. (2017). It recursively divides a span into two smaller ones, until a single-token span, i.e., a leaf, is reached. The order of the partition specifies the tree structure.[7] Formally, for a maximum span length $m$, at each time step $j+1$, we split the span $[j-m+1, j]$ into two smaller parts $[j-m+1, k_0]$ and $[k_0+1, j]$. The partitioning point is greedily selected, maximizing the attention scores of spans ending at $j$:[8]

$$k_0 = \operatorname*{argmax}_{k \in \{0,\ldots,m-1\}} s_{j,k}. \tag{4}$$

The span is directly returned as a leaf if it contains a single token. A full parse is derived by running the algorithm recursively, starting with the input as a single span (with a special end-of-sentence mark at the end). The runtime is $\mathcal{O}(n^2)$, with $n-1$ partitioning points. See Fig. 1 for an illustration.

**Supervising the attention.** Now that we are able to derive phrase structures from attention weights, we can further inform the attention if syntactic annotations are available, using oracle span selections. For each token, the gold selection is a $m$-dimensional binary vector, and then normalized to sum to one, denoted $\mathbf{y}_t$.[9] We add a cross-entropy loss (averaged across the training data) to the language modeling objective, with $\lambda$ trading off between the two:

$$\mathcal{L} = \mathcal{L}_{\text{LM}} + \frac{\lambda}{N} \sum_{t=1}^{N} \mathcal{H}(\mathbf{y}_t, \boldsymbol{\omega}_t), \tag{5}$$

with $\boldsymbol{\omega}$ being the attention distribution at step $t$, and $N$ the length of the training corpus. As we will see in §3, providing syntactically guided span attention improves language modeling performance.

**Discussion.** PaLM provides an intuitive way to inject structural inductive bias into the language model—by supervising the attention distribution. This setting can be seen as a very lightweight multitask learning, where no actual syntactic tree is predicted during language modeling training or evaluation. The attention weight predictor (i.e., the $s$ scores in Eq. 3b) can be replaced with an off-the-shelf parser, or deterministically set (e.g., to simulate left/right-branching).

[6] / denotes elementwise division. Both elementwise product and division are implemented in log-space.

[7] It is only able to produce binarized unlabeled trees.

[8] Another natural choice is to maximize the sum of the scores of $[i, k_0]$ and $[k_0+1, j]$. The attention score of $[i, k_0]$ is computed at time step $k_0$, and hence does not know anything about the other span on the right. Therefore we consider only the score of the right span.

[9] Not necessarily one-hot: multiple spans can end at the same token.

## 3 Experiments

We evaluate PaLM on language modeling. We experiment with the Penn Treebank corpus (PTB) and WikiText-2 (WT2). We follow the preprocessing of Mikolov et al. (2010) for PTB and Merity et al. (2018) for WT2. More implementation details are described in Appendix A. We compare two configurations of PaLM:

- PaLM-U builds on top of AWD-LSTM (Merity et al., 2018), a state-of-the-art of LSTM implementation for language modeling. The span attention is included *before* the last layer.[10]
- PaLM-S is the same model as PaLM-U, but uses phrase syntax annotation to provide additional supervision to the attention component (§2.2).[11]

We compare against the AWD-LSTM baseline. On PTB, we also compare to two models using structural information in language modeling: parsing-reading-predict networks (PRPN; Shen et al., 2018a) predicts syntactic distance as structural features for language modeling; ordered-neuron LSTM (ON-LSTM; Shen et al., 2018b) posits a novel ordering on LSTM gates, simulating the covering of phrases at different levels in a constituency parse. On PTB we also compare to PaLM-RB, a baseline deterministically setting the attention scores (Eq. 3b) in decreasing order, such that the derived trees will be right-branching.[12]

Tables 1 and 2 summarize the language modeling results. On both datasets, the unsupervised configuration (PaLM-U) outperforms AWD-LSTM. On PTB, PaLM-U achieves similar performance to ON-LSTM and much better performance than PRPN. PaLM-S further reduces the perplexity by 1.6–3.4% (relative), showing that incorporating structural information with supervised span attention helps language modeling. Naively promoting right-branching attention (PaLM-RB) yields no improvement over the baseline.

**Unsupervised constituency parsing.** We evaluate the parser component of PaLM-U on WSJ-40. It uses the same data as in language modeling, but filters out sentences longer than 40 tokens after punctuation removal. The model is selected based on language modeling validation perplexity.

| Model | # Params. | Dev. | Test |
|---|---|---|---|
| AWD-LSTM | 24M | 60.0 | 57.3 |
| PRPN | - | - | 62.0 |
| ON-LSTM | 25M | 58.3 | 56.2 |
| PaLM-U | 24M | 58.6 | 56.4 |
| PaLM-RB | 24M | 60.1 | 57.5 |
| PaLM-S | 24M | **57.9** | **55.5** |

Table 1: PTB language modeling perplexity (lower is better). Bold fonts indicate best performance.[13]

| Model | # Params. | Dev. | Test |
|---|---|---|---|
| AWD-LSTM | 33M | 68.6 | 65.8 |
| PaLM-U | 36M | 68.4 | 65.4 |
| PaLM-S | 36M | **65.5** | **63.2** |

Table 2: WikiText-2 language modeling perplexity (lower is better). Bold fonts indicate best performance.

In addition to PRPN, we compare to DIORA (Drozdov et al., 2019), which uses an inside-outside dynamic program in an autoencoder. Table 3 shows the $F_1$ results. PaLM outperforms the right branching baseline, but is not as accurate as the other models.[14] This indicates that the type of syntactic trees learned by it, albeit useful to the LM component, do not correspond well to PTB-like syntactic trees.

**Discussion.** Despite its strong performance, the parsing algorithm used by Shen et al. (2018a) and Shen et al. (2018b) suffers from an incomplete support issue (Dyer et al., 2019). More precisely, it fails to produce "close-open-open," i.e., ) ( ( structures. As a result, the parser is intrinsically biased toward right-branching structures. PaLM, on the other hand, scores all the spans, and therefore can produce any binary tree spanning a given sentence: the algorithm recovers any given binary tree by letting $s_{j,j-i} = 1$ if the tree contains nonterminal $[i, j]$, and 0 otherwise.[15]

Is PaLM empirically biased toward any branching direction? In greedily selected trees, we measure the percentage of left-branching splits (dividing $[i, j]$ into $[i, j-1]$ and $j$) and right-branching

[10] Preliminary experiments show that including the span attention after the last layer yields similar empirical results, but is more sensitive to hyperparameters.

[11] We use the WSJ portion of PTB for parsing annotations.

[12] We set scores to $m, m-1, \ldots, 1$, before the softmax.

[13] Several recent works report better language modeling perplexity (Yang et al., 2019; Takase et al., 2018; Dai et al., 2019, *inter alia*). Their contribution is orthogonal to ours and not head-to-head comparable to the models in the table.

[14] Evaluation on WSJ-10, which contains sentences with 10 or less tokens, shows a similar trend.

[15] The maximum span length $m$ is only forced in language modeling training and evaluation.

| Model | Unlabeled $F_1$ |
|---|---|
| Right Branching | 40.7 |
| †DIORA | 60.6 |
| ‡PRPN | 52.4 |
| ‡PaLM-U | 42.0 |

Table 3: Unlabeled unsupervised parsing $F_1$ on WSJ-40. ‡ trains on the training split of WSJ, while † trains on AllNLI (Htut et al., 2018). The PRPN result is taken from Drozdov et al. (2019).

| | % Left Splits | | % Right Splits | |
|---|---|---|---|---|
| Random | 39.3 | ±10.5 | 41.2 | ±8.8 |
| PaLM-U | 1.1 | | 85.6 | |
| Gold | 6.5 | | 52.7 | |

Table 4: Percentage of left and right splits. The first row shows the numbers averaging over 25 differently randomly initialized PaLM models, without training. ± indicates standard deviation.

splits (dividing $[i, j]$ into $i$ and $[i+1, j]$).[16] Table 4 summarizes the results on WSJ-40 test set. The first row shows the results for randomly initialized models without training. We observe no significant trend of favoring one branching direction over the other. However, after training with the language modeling objective, PaLM-U shows a clear right-skewness more than it should: it produces much more right-branching structures than the gold annotation. This means that the span attention mechanism has learned to emphasize longer prefixes, rather than make strong Markov assumptions. More exploration of this effect is left to future work.

## 4 Conclusion

We present PaLM, a hybrid parser and language model. PaLM attends over the preceding text spans. From its attention weights phrase structures can be derived. The attention component can be separately trained to provide syntactically-informed context gathering. PaLM outperforms strong baselines on language modeling. Incorporating syntactic supervision during training leads to further language modeling improvements. Training our unsupervised model on large-scale corpora could result in both stronger language models and, potentially, stronger parsers. Our code is publicly available at https://github.com/Noahs-ARK/PaLM.

[16] We exclude trivial splits dividing a length-2 span into two tokens.


## Acknowledgments

We thank members of the ARK at the University of Washington, and researchers at the Allen Institute for Artificial Intelligence for their helpful comments on an earlier version of this work, and the anonymous reviewers for their insightful feedback. This work was supported in part by NSF grant 1562364.


## References


Jan Buys and Phil Blunsom. 2018. Neural syntactic generative models with exact marginalization. In *Proc. of NAACL*.

Ciprian Chelba and Frederick Jelinek. 1998. Exploiting syntactic structure for language modeling. In *Proc. of COLING*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proc. of EMNLP*.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proc. of ACL*.

Dipanjan Das, André F. T. Martins, and Noah A. Smith. 2012. An exact dual decomposition algorithm for shallow semantic parsing with constraints. In *Proc. of *SEM*.

Andrew Drozdov, Pat Verga, Mohit Yadav, Mohit Iyyer, and Andrew McCallum. 2019. Unsupervised latent tree induction with deep inside-outside recursive autoencoders. In *Proc. of NAACL*.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proc. of NAACL*.

Chris Dyer, Gbor Melis, and Phil Blunsom. 2019. A critical analysis of biased parsers in unsupervised parsing. arXiv:1909.09428.

Daniel Gildea and Martha Palmer. 2002. The necessity of parsing for predicate argument recognition. In *Proc. of ACL*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. *Proc. of CVPR*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proc. of ACL*.

Phu Mon Htut, Kyunghyun Cho, and Samuel Bowman. 2018. Grammar induction with neural language models: An unusual replication. In *Proc. of EMNLP*.

Frederick Jelinek and John D. Lafferty. 1991. Computation of the probability of initial substring generation by stochastic context-free grammars. *Computational Linguistics*, 17(3):315–353.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. arXiv:1602.02410.

Yoon Kim, Alexander M. Rush, Lei Yu, Adhiguna Kuncoro, Chris Dyer, and Gábor Melis. 2019. Unsupervised recurrent neural network grammars. In *Proc. of NAACL*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv:1412.6980.

Lingpeng Kong, Chris Dyer, and Noah A. Smith. 2016. Segmental recurrent neural networks. In *Proc. of ICLR*.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proc. of EMNLP*.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and optimizing LSTM language models. In *Proc. of ICLR*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *Proc. of ICLR*.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proc. of INTERSPEECH*.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proc. of EMNLP*.

Hao Peng, Roy Schwartz, Sam Thomson, and Noah A. Smith. 2018a. Rational recurrences. In *Proc. of EMNLP*.

Hao Peng, Sam Thomson, and Noah A. Smith. 2018b. Backpropagating through structured argmax using a spigot. In *Proc. of ACL*.

Hao Peng, Sam Thomson, Swabha Swayamdipta, and Noah A. Smith. 2018c. Learning joint semantic parsers from disjoint data. In *Proc. of NAACL*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

B. T. Polyak and A. B. Juditsky. 1992. Acceleration of stochastic approximation by averaging. *SIAM J. Control Optim.*, 30(4):838–855.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *American Journal of Computational Linguistics*, 34(2):257–287.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. OpenAI Blog.

Brian Roark. 2001. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249–276.

Yikang Shen, Zhouhan Lin, Chin wei Huang, and Aaron Courville. 2018a. Neural language modeling by jointly learning syntax and lexicon. In *Proc. of ICLR*.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron C. Courville. 2018b. Ordered neurons: Integrating tree structures into recurrent neural networks. In *Proc. of ICLR*.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proc. of ACL*.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proc. of EMNLP*.

Swabha Swayamdipta, Sam Thomson, Kenton Lee, Luke Zettlemoyer, Chris Dyer, and Noah A. Smith. 2018. Syntactic scaffolds for semantic structures. In *Proc. of EMNLP*.

Sho Takase, Jun Suzuki, and Masaaki Nagata. 2018. Direct output connection for a high-rank language model. In *Proc. of EMNLP*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proc. of NeurIPS*.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. 2019. Breaking the softmax bottleneck: A high-rank RNN language model. In *Proc. of ICLR*.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. SWAG: A large-scale adversarial dataset for grounded commonsense inference. In *Proc. of EMNLP*.

# Appendices

## A Implementation Details

**Neural Network Architecture** Our implementation is based on AWD-LSTM (Merity et al., 2018).[17] It uses a three-layer LSTM, with carefully designed regularization techniques. PaLM includes the span attention after the second layer. Preliminary results show that it yields similar results, but is less sensitive to hyperparameters, compared to adding it to the last layer.

The context is concatenated to the hidden state ($\bar{\mathbf{h}}_t = [\mathbf{h}_t; \mathbf{a}_t]$), and then fed to a $\tanh$-MLP controlled by a residual gate $\mathbf{g}_r$ (He et al., 2016), before fed onward into the next LSTM layer:

$$\hat{\mathbf{h}}_t = \mathbf{g}_r \odot \text{MLP}\left(\bar{\mathbf{h}}_t\right) + (\mathbf{1} - \mathbf{g}_r) \odot \mathbf{h}_t. \quad (6)$$

The rest of the architecture stays the same as AWD-LSTM. We refer the readers to Merity et al. (2018) for more details.

**More details on PaLM-S.** PaLM-S uses exactly the same architecture and hyperparameters as its unsupervised counterpart. We derive, from PTB training data, a $m$-dimensional 0-1 vector for each token. Each element specifies whether the corresponding span appears in the gold parse. Trivial spans (i.e., the ones over single tokens and full sentences) are ignored. The vector are normalized to sum to one, in order to facilitate the use of cross-entropy loss. $\lambda$ in Eq. 5 is set to 0.01.

**Hyperparameters.** The regularization and hyperparameters largely follow Merity et al. (2018). We only differ from them by using smaller hidden size (and hence smaller dropout rate) to control for the amount of parameters in the PTB experiments, summarized in Table 5 For the WikiText-2 experiments, we use 200 rational RNN size and 400 dimensional context vectors. Other hyperparameters follow Merity et al. (2018). The max span length $m$ is set to 20 for PTB experiments, and 10 for WikiText-2.

Merity et al. (2018) start by using SGD to train the model, and switch to averaged SGD (Polyak and Juditsky, 1992) after 5 nonimprovement-epochs. We instead use Adam (Kingma and Ba, 2014) with default PyTorch settings to train the model for 40 epochs, and then switch to ASGD, allowing for faster convergence.

[17] https://github.com/salesforce/awd-lstm-lm

| Type | Values |
|---|---|
| Rational RNN size | 200 |
| Context Vector Size | 400 |
| LSTM Hidden Size | 1020 |
| Weight Dropout | 0.45 |
| Vertical Dropout | 0.2 |

Table 5: The hyperparameters used in the PTB language modeling experiment.

## B Span Representations

Below is the derivation for Eq. 2.

$$\begin{aligned}
\overrightarrow{\mathbf{c}}_{i,j} &= \overrightarrow{\mathbf{u}}_j + \sum_{k=i}^{j-1} \overrightarrow{\mathbf{u}}_k \bigodot_{\ell=k+1}^{j} \overrightarrow{\mathbf{f}}_\ell \\
&= \overrightarrow{\mathbf{u}}_j + \sum_{k=1}^{j-1} \overrightarrow{\mathbf{u}}_k \bigodot_{\ell=k+1}^{j} \overrightarrow{\mathbf{f}}_\ell - \sum_{k=1}^{i-1} \overrightarrow{\mathbf{u}}_k \bigodot_{\ell=k+1}^{j} \overrightarrow{\mathbf{f}}_\ell \\
&= \overrightarrow{\mathbf{c}}_j - \left( \overrightarrow{\mathbf{u}}_{i-1} + \sum_{k=1}^{i-2} \overrightarrow{\mathbf{u}}_k \bigodot_{\ell=k+1}^{i-1} \overrightarrow{\mathbf{f}}_\ell \right) \bigodot_{\ell=i}^{j} \overrightarrow{\mathbf{f}}_\ell \\
&= \overrightarrow{\mathbf{c}}_j - \overrightarrow{\mathbf{c}}_{i-1} \bigodot_{k=i}^{j} \overrightarrow{\mathbf{f}}_k
\end{aligned}$$

# A Pilot Study for Chinese SQL Semantic Parsing

**Qingkai Min , Yuefeng Shi** and **Yue Zhang**
School of Engineering, Westlake University, China
Institute of Advanced Technology, Westlake Institute for Advanced Study
{minqingkai, shiyuefeng, zhangyue}@westlake.edu.cn

## Abstract

The task of semantic parsing is highly useful for dialogue and question answering systems. Many datasets have been proposed to map natural language text into SQL, among which the recent Spider dataset provides cross-domain samples with multiple tables and complex queries. We build a Spider dataset for Chinese, which is currently a low-resource language in this task area. Interesting research questions arise from the uniqueness of the language, which requires word segmentation, and also from the fact that SQL keywords and columns of DB tables are typically written in English. We compare character- and word-based encoders for a semantic parser, and different embedding schemes. Results show that word-based semantic parser is subject to segmentation errors and cross-lingual word embeddings are useful for text-to-SQL.

## 1 Introduction

The task of semantic parsing is highly useful for tasks such as dialogue (Chen et al., 2013; Gupta et al., 2018; Einolghozati et al., 2019) and question answering (Gildea and Jurafsky, 2002; Yih et al., 2015; Reddy et al., 2016). Among a wide range of possible semantic representations, SQL offers a standardized interface to knowledge bases across tasks (Astrova, 2009; Xu et al., 2017; Dong and Lapata, 2018; Lee et al., 2011). Recently, Yu et al. (2018b) released a manually labelled dataset for parsing natural language questions into complex SQL, which facilitates related research.

Yu et al. (2018b)'s dataset is exclusive for English questions. Intuitively, the same semantic parsing task can be applied cross-lingual, since SQL is a universal semantic representation and database interface. However, for languages other than English, there can be added difficulties parsing into SQL. Take Chinese for example, the additional challenges can be at least two-fold. First, structures of relational databases, in particular names and column names of DB tables, are typically represented in English. This adds to the challenges to question-to-DB mapping. Second, the basic semantic unit for denoting columns or cells can be words, but word segmentation can be erroneous. It is also interesting to study the influence of other linguistic characteristics of Chinese, such as zero-pronoun, on its SQL parsing.

We investigate parsing Chinese questions to SQL by creating a first dataset, and empirically evaluating a strong baseline model on the dataset. In particular, we translate the Spider (Yu et al., 2018b) dataset into Chinese. Using the model of Yu et al. (2018a), we compare several key model configurations.

Results show that our human-translated dataset is significantly more reliable compared to a dataset composed of machine-translated questions. In addition, the overall accuracy for Chinese SQL semantic parsing can be comparable to that for English. We found that cross-lingual word embeddings are useful for matching Chinese questions with English table columns and keywords and that language characteristics have a significant influence on parsing results. We release our dataset named CSpider and code at https://github.com/taolusi/chisp.

## 2 Related Work

Existing datasets for semantic parsing can be classified into two major categories. The first uses logic for semantic representation, including ATIS (Price, 1990; Dahl et al., 1994) and GeroQuery (Zelle and Mooney, 1996). The second and dominant category of datasets uses SQL, which includes Restaurants (Tang and Mooney, 2001; Popescu et al., 2003), Academic (Iyer et al., 2017), Yelp and IMDB (Yaghmazadeh et al., 2017), Ad-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3652–3658, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

vising (Finegan-Dollak et al., 2018) and the recently proposed WikiSQL (Zhong et al., 2017) and Spider (Yu et al., 2018b). One salient difference between Spider and prior work is that Spider uses different databases across domains for training and testing, which can verify the generalization power of a semantic parsing model. Compared with WikiSQL, Spider further has multiple tables in each database and correspondingly more complex queries. We thus consider Spider for sourcing our dataset. Existing semantic parsing datasets for Chinese include a small corpus for assigning semantic roles (Sun and Jurafsky, 2004) and SemEval-2016 Task 9 for Chinese semantic dependency parsing (Che et al., 2012), but these data are not related to SQL. To our knowledge, we are the first to release a Chinese SQL semantic parsing dataset.

There has been a line of work improving the model of Yu et al. (2018a) since the release of the Spider dataset (Guo et al., 2019; Bogin et al., 2019; Lin et al., 2019). At the time of our investigation, however, the models are not published. We thus chose the model of Yu et al. (2018a) as our baseline. The choice of more different neural models is orthogonal to our dataset contribution, but can empirically give more insights about the conclusions.

## 3 Dataset

We translate all English questions in the Spider dataset into Chinese.[1] The work is undertaken by 2 NLP researchers and 1 computer science student. Each question is first translated by one annotator, and then cross-checked and corrected by a second annotator. Finally, a third annotator verifies the original and corrected versions. Statistics of the dataset are shown in Table 1. There are originally 10181 questions from Spider, but only 9691 for the training and development sets are publicly available. We thus translated these sentences only. Following the database split setting of Yu et al. (2018b), we make training, development and test sets split in a way that no database overlaps in them as shown in Table 1.

The translation work is performed on a database to database basis. For each database, the same translator translates relevant inquiries sentence by sentence. The translator is asked to read the original question as well as the SQL query before making its Chinese translation. If the literal translation is possible, the translator is asked to stick to the original sentence style as much as feasible. For complex questions, the translator is allowed to rephrase the English question, so that the most natural Chinese translation is made. In addition, we keep the diversity of style in the English dataset by matching different English expressions to different Chinese expressions. A sample of our dataset is shown in Table 2. Our dataset is named CSpider.

[1]Note that we do not translate the database schema (i.e., column names) into Chinese because in practice databases have English schema and Chinese contents in the industry.

| | | # Q | # SQL | # DB | # Table/DB |
|---|---|---|---|---|---|
| English | all | 10181 | 5693 | 200 | 5.1 |
| Chinese | all | 9691 | 5263 | 166 | 5.28 |
| | train | 6831 | 3493 | 99 | 5.38 |
| | dev | 954 | 589 | 25 | 4.16 |
| | test | 1906 | 1193 | 42 | 5.69 |

Table 1: Comparisons between Spider and Chinese Spider datasets.

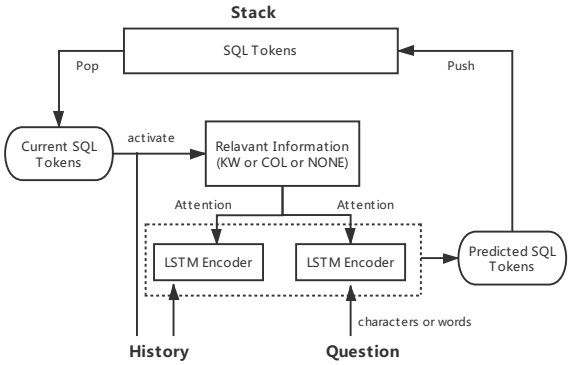


Figure 1: Overall structure of the Model.

## 4 Model

We use the neural semantic parsing method of Yu et al. (2018a) as the baseline model, which can be regarded as a sequence-to-tree model. In particular, the input question is encoded using an LSTM sequence encoder, and the output is a SQL query in its syntactic tree form. The tree is generated incrementally top-down, in a pre-order traversal sequence. Tree nodes include keyword nodes (e.g., SELECT, WHERE, EXCEPT) and table column name nodes (e.g., ID, City, Surname, which are defined in specific tables), which are represented in respective embedding spaces. Each keyword or column is generated by attention to the embedding space using the question representation as a key. A stack is used for incremental decoding, where

| **Sample 1: applying only one table in one database.** |
|---|
| **SQL Query** |
| SELECT area FROM state WHERE state_name = "New Mexico"; |
| **English Question** |
| What is the size of New Mexico? |
| **Translated Chinese Question** |
| 新墨西哥州的面积是多少? |
| **Sample 2: applying multiple tables in one database.** |
| **SQL Query** |
| SELECT T2.star_rating_description FROM HOTELS AS T1 JOIN Ref_Hotel_Star_Ratings AS T2 ON T1.star_rating_code = T2.star_rating_code WHERE T1.price_range >10000; |
| **English Question** |
| Give me the star rating descriptions of the hotels that cost more than 10000. |
| **Translated Chinese Question** |
| 给出费用超过10000的酒店星级的描述。 |
| **Sample 3:with a nested SQL query.** |
| **SQL Query** |
| SELECT T1.staff_name , T1.staff_id FROM Staff AS T1 JOIN Fault_Log AS T2 ON T1.staff_id = T2.recorded_by_staff_id EXCEPT SELECT T3.staff_name, T3.staff_id FROM Staff AS T3 JOIN Engineer_Visits AS T4 ON T3.staff_id = T4.contact_staff_id"; |
| **English Question** |
| What is the name and ID of the staff who recorded the fault log but has not contacted any visiting engineers? |
| **Translated Chinese Question** |
| 那些记录了错误报告但没有联系任何到访工程师的职工的姓名和ID是什么? |

Table 2: Example questions corresponding to SQL.

the whole output history is leveraged as a feature for deciding the next term. This method gives the current released state-of-the-art results while submitting this paper. We provide a visualization of the model in Figure 1.

## 5 Experiments

We focus on comparing different word segmentation methods and different embedding representations. As discussed above, column names are selected by attention over column embeddings using sentence representation as a key. Hence there must be a link between the embeddings of columns and those of the questions. Since columns are written in English and questions in Chinese, we consider two embedding methods. The first method is to use two separate sets of embeddings for Chinese and English, respectively. We use Glove (Pennington et al., 2014)[2] for embeddings of English keywords, column names etc., and Tencent embeddings (Song et al., 2018)[3] for Chinese. The second method is to directly use the cross-lingual word embeddings. To this end, the Tencent multi-lingual embeddings are chosen, which contain both Chinese and English words in a multi-lingual embedding matrix.

**Evaluation Metrics**. We follow Yu et al. (2018b), evaluating the results using two major types of metrics. The first is exact matching accuracy, namely the percentage of questions that have exactly the same SQL output as its reference. The second is component matching F1, namely the F1 scores for SELECT, WHERE, GROUP BY, ORDER BY and all keywords, respectively.

**Hyperparameters**. Our hyperparameters are mostly taken from Yu et al. (2018a), but tuned on the Chinese Spider development set. We use character and word embeddings from Tencent embedding; both of them are not fine-tuned during model training. Embedding sizes are set to 200 for both characters and words. For the different choices of keywords and column names embeddings, sizes are set to 200 and 300, respectively. Adam (Kingma and Ba, 2014) is used for optimization, with a learning rate of 1e-4. Dropout is used for the output of LSTM with a rate of 0.5.

For word-based models, segmentation is necessary. We take two segmentors with different performances, including the Jieba segmentor and the model of Yang et al. (2017), which we name Jieba and YZ, respectively. To verify their accuracy, we manually segment the first 100 sentences from the test set. Jieba and YZ give F1 scores of 89.8% and 91.7%, respectively.

### 5.1 Overall Results

The overall exact matching results are shown in Table 3. In this table, ENG represents the results of Yu et al. (2018a)'s model on their English dataset but under our split. HT and MT denote human translation and machine translation of questions, respectively. Both HT and MT results are evaluated on human translated questions. C-ML and C-S denote the results of our Chinese models based on characters with multi-lingual embeddings and monolingual embeddings, respectively, while WY-ML, WY-S denote the word-based models applying YZ segmentor with multi-lingual embeddings and monolingual embeddings, respectively. Finally, WJ-ML and WJ-S denote the word model with multi-lingual embeddings and monolingual embeddings with the Jieba segmentor, respectively.

First, compared to the best results of human translation (C-ML and WY-ML), machine translation results show a large disadvantage (e.g. 7.1% vs 12.1% using C-ML). We further did a manual inspection of 100 randomly picked machine-translated sentences. Out of the 100 translated

[2]https://nlp.stanford.edu/projects/glove/
[3]https://ai.tencent.com/ailab/nlp/embedding.html

| | | Easy | Medium | Hard | Extra Hard | All |
|---|---|---|---|---|---|---|
| ENG | | 31.8% | 11.3% | 9.5% | 2.7% | 14.1% |
| HT | C-ML | **27.3%** | **9.9%** | 7.5% | **2.3%** | **12.1%** |
| | C-S | 23.1% | 7.7% | 6.2% | 1.7% | 9.9% |
| | WY-ML | 21.4% | 8.1% | **8.0%** | 1.7% | 10.0% |
| | WY-S | 20.2% | 6.4% | 6.7% | 2.0% | 8.9% |
| | WJ-ML | 19.8% | 8.6% | 5.0% | 1.3% | 9.2% |
| | WJ-S | 20.1% | 5.0% | 5.7% | 1.7% | 8.2% |
| MT | C-ML | 18.1% | 4.6% | 5.2% | 0.3% | 7.9% |
| | WY-ML | 17.9% | 4.7% | 4.5% | 0.3% | 7.6% |

Table 3: Accuracy of Exact Matching on test set.

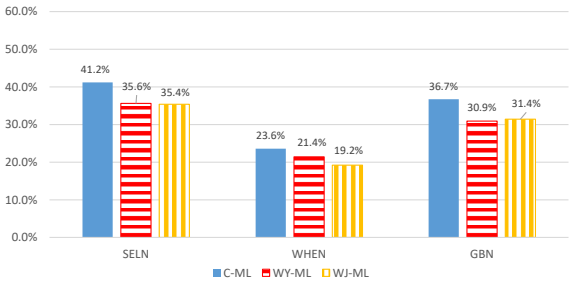


Figure 2: Component Matching Comparisons.

sentences, 42 have translation mistakes such as semantic changes (28 sentences) and grammar errors (14 sentences). Both of these facts indicate that data by machine-translation is not reliable for semantic parsing research.

Second, comparisons among C-ML, WY-ML and WJ-ML, and among C-S, WY-S and WJ-S show that multi-lingual embeddings give superior results compared to monolingual embeddings, which is likely because they bring a better connection between natural language questions and database columns.

Third, comparisons between WY-ML and WJ-ML, and WY-S and WJ-S indicate that better segmentation accuracy has a significant influence on question parsing. Word-based methods are subject to segmentation errors.

Moreover, with the current segmentation accuracy of 92%, a word-based model underperforms a character-based model. Intuitively, since words carry more direct semantic information as compared with database columns and keywords, improved segmentation may allow a word-based model to outperform a character-based model.

Finally, for easy questions, the character-based model shows strong advantages over the word-based models. However, for medium to extremely hard questions, the trend becomes less obvious, which is likely because the intrinsic semantic complexity overwhelms the encoding differences.

Our best Chinese system gives an overall accuracy of 12.1%, [4] which is less but comparable to the English results. This shows that Chinese semantic parsing may not be significantly more challenging compared to English with text to SQL.

**Component matching**. Figure 2 shows F1 scores of several typical components, including SELN (SELECT NO AGGREGATOR), WHEN (WHERE NO OPERATOR) and GBN (GROUP BY NO HAVING), applying the superior multi-lingual embeddings. The trends are consistent with the overall results.

The detailed results are shown in Table 4. Specifically, the char-based methods achieve around 41% on SELN and SEL (SELECT), which are about 5% higher compared to the word-based methods. This result may be due to the fact that word-based models are sensitive to the OOV words (Zhang and Yang, 2018; Li et al., 2019). Unlike other components, SEL and SELN are confronted with more severe OOV challenges caused by recognizing the unseen schema during testing.

In addition, the models using multi-lingual embedding overperform the models using separate embeddings on both WHEN and OB (ORDERBY), which further demonstrates that embeddings in the same dimension distribution benefit to strengthen the connection between the question and the schema.

Contrary to the overall result, the models employing the jieba segmentor perform better than those using the YZ segmentor on OB. The reason is that the jieba segmentor has different word segmentation results in terms of the superlative of adjectives. For example, the word "最高" (the highest) is segmented as "最"(most) and "高"(high) by YZ segmentor but "最高" in jieba segmentor. This again demonstrates the influence of word segmentation. Finally, for GB (GROUPBY) there is not a regular contrast pattern between different models, which can be likely because of the lack of sufficient training data.

## 5.2 Case study

Figure 3 shows the negative influence of segmentation errors. In particular, the incorrect segmentation of the word "店名" (shop name) leads to incorrect SQL for the whole sentence, since the

[4] Note that the results are lower than those reported by Yu et al. (2018a) under their split due to different training/test splits. Our split has less training data and more test instances in the "Hard" category and less in "Easy" and "Medium".

| | | SEL | SELN | WHE | WHEN | GB | GBN | OB |
|---|---|---|---|---|---|---|---|---|
| ENG | | 47.3% | 48.2% | 19.9% | 24.4% | 35.0% | 40.6% | 57.6% |
| HT | C-ML | 40.7% | 41.2% | 19.9% | 23.6% | 33.6% | 36.7% | 53.8% |
| | C-S | 40.6% | 41.0% | 15.3% | 17.3% | 29.2% | 32.9% | 51.7% |
| | WY-ML | 34.8% | 35.6% | 18.1% | 21.4% | 26.7% | 30.9% | 49.8% |
| | WY-S | 34.5% | 35.6% | 16.5% | 19.8% | 30.2% | 34.2% | 46.9% |
| | WJ-ML | 34.7% | 35.4% | 15.8% | 19.2% | 27.9% | 31.4% | 52.5% |
| | WJ-S | 35.7% | 36.8% | 15.9% | 19.6% | 24.4% | 26.8% | 48.0% |
| MT | C-ML | 36.5% | 37.2% | 11.3% | 14.2% | 29.1% | 33.4% | 50.7% |
| | WY-ML | 32.1% | 32.8% | 11.3% | 13.4% | 24.8% | 27.5% | 49.1% |

Table 4: F1 scores of Component Matching on test set.

| Word segmentation error | Predicted query |
|---|---|
| 哪些 商店 的 产品 数量 高于 平均 水平 ？ 把 店 名 给 我 。<br>Which shops' number products is above the average? Give me the shop names. | SELECT Manager_name FROM shop WHERE Number_products > (SELECT AVG(Number_products) FROM shop) |
| 哪些 商店 的 产品 数量 高于 平均 水平 ？ 把 店名 给 我 。<br>Which shops' number products is above the average? Give me the shop names. | SELECT name FROM shop WHERE Number_products > (SELECT AVG(Number_products) FROM shop) |

Figure 3: Word segmentation error.

| Sentence patterns | Predicted query |
|---|---|
| 最 常 见 的 教师 的 家乡 是 哪里 ?<br>What is the most common hometowns for teachers? | SELECT Hometown FROM teacher ORDER BY Age DESC LIMIT 1 |
| 列出 最 常 见 的 教师 的 家乡 。<br>List the most common hometown of teachers. | SELECT Hometown FROM teacher GROUP BY Hometown ORDER BY COUNT(*) DESC LIMIT 1 |

Figure 4: Sentence pattern.

| Chinese zero pronoun | Predicted query |
|---|---|
| 代表 的 不同 党派 是 什么 ？ 显示 各 党 的 党名 和 代表 人数 。<br>What are the different parties of representative? Show the party name and the number of representatives. | SELECT Date , COUNT(*) FROM election GROUP BY Seats |
| 代表 的 不同 党派 是 什么 ？ 显示 各 党 的 党名 和 各 党 的 代表 人数 。<br>What are the different parties of representative? Show the party name and the number of representatives in each party. | SELECT Party , COUNT(*) FROM representative GROUP BY Party |

Figure 5: Chinese zero pronoun.

character "店" (shop) can typically be associated with "店长" (shop manager).

Figure 4 shows the sensitivity of our model to sentence patterns. In particular, the word-based model gives incorrect predictions for many question sentences frequently. As shown in the first row, the word "where" confuses the system for making a choice between "ORDER BY" and "GROUP BY". When we manually change the sentence pattern into "List the most common hometown of teachers", the parser gives the correct keyword. In contrast, the character-based model is less sensitive to question sentences, which is likely because characters are less sparse compared with words. More training data or contextualized embeddings may alleviate the issue for the word-based method, which we leave for future work.

Figure 5 shows the sensitivity of the model to Chinese linguistic patterns. In particular, the first sentence has a zero pronoun "各党的" (in each party), which is omitted later. As a result, a semantic parser cannot tell the correct database columns from the sentence. We manually add the correct entity for the zero pronoun, resulting in the second sentence. Since zero-pronouns are frequent for Chinese (Chen and Ng, 2016), they give added difficulty for its semantic parsing.

## 6 Conclusion

We constructed a first resource named CSpider for Chinese sentence to SQL, evaluating the performance of a strong English model on this dataset. Results show that the input representation, embedding forms and linguistic factors all have the influence on the Chinese-specific task. Our dataset can serve as a starting point for further research on this task, which can be beneficial to the investigation of Chinese QA and dialogue models.

## Acknowledgments

We thank the anonymous reviewers for their detailed and constructive comments. Yue Zhang is the corresponding author.

## References


Irina Astrova. 2009. Rules for mapping sql relational databases to owl ontologies. *Metadata and Semantics*, pages 415–424.

Ben Bogin, Jonathan Berant, and Matt Gardner. 2019. Representing schema structure with graph neural networks for text-to-SQL parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4560–4565, Florence, Italy. Association for Computational Linguistics.

Wanxiang Che, Meishan Zhang, Yanqiu Shao, and Ting Liu. 2012. Semeval-2012 task 5: Chinese semantic dependency parsing. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 378–384. Association for Computational Linguistics.

Chen Chen and Vincent Ng. 2016. Chinese zero pronoun resolution with deep neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 778–788.

Yun-Nung Chen, William Yang Wang, and Alexander I Rudnicky. 2013. Unsupervised induction and filling of semantic slots for spoken dialogue systems using frame-semantic parsing. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 120–125. IEEE.

Deborah A Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. 1994. Expanding the scope of the atis task: The atis-3 corpus. In *Proceedings of the workshop on Human Language Technology*, pages 43–48. Association for Computational Linguistics.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 731–742.

Arash Einolghozati, Panupong Pasupat, Sonal Gupta, Rushin Shah, Mrinal Mohit, Mike Lewis, and Luke Zettlemoyer. 2019. Improving semantic parsing for task oriented dialog. *arXiv preprint arXiv:1902.06000*.

Catherine Finegan-Dollak, Jonathan K Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving text-to-sql evaluation methodology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 351–360.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational linguistics*, 28(3):245–288.

Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. 2019. Towards complex text-to-SQL in cross-domain database with intermediate representation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4524–4535, Florence, Italy. Association for Computational Linguistics.

Sonal Gupta, Rushin Shah, Mrinal Mohit, Anuj Kumar, and Mike Lewis. 2018. Semantic parsing for task oriented dialog using hierarchical representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2787–2792.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 963–973.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Rubao Lee, Tian Luo, Yin Huai, Fusheng Wang, Yongqiang He, and Xiaodong Zhang. 2011. Ysmart: Yet another sql-to-mapreduce translator. In *2011 31st International Conference on Distributed Computing Systems*, pages 25–36. IEEE.

Xiaoya Li, Yuxian Meng, Xiaofei Sun, Qinghong Han, Arianna Yuan, and Jiwei Li. 2019. Is word segmentation necessary for deep learning of chinese representations? In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 3242–3252.

Kevin Lin, Ben Bogin, Mark Neumann, Jonathan Berant, and Matt Gardner. 2019. Grammar-based neural text-to-sql generation. *arXiv preprint arXiv:1905.13326*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. 2003. Towards a theory of natural language interfaces to databases. In *Proceedings of the 8th international conference on Intelligent user interfaces*, pages 149–157. ACM.

Patti J Price. 1990. Evaluation of spoken language systems: The atis domain. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*.

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming dependency structures to logical forms for semantic parsing.

*Transactions of the Association for Computational Linguistics*, 4:127–140.

Yan Song, Shuming Shi, Jing Li, and Haisong Zhang. 2018. Directional skip-gram: Explicitly distinguishing left and right context for word embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 175–180.

Honglin Sun and Daniel Jurafsky. 2004. Shallow semantc parsing of chinese. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*.

Lappoon R Tang and Raymond J Mooney. 2001. Using multiple clause constructors in inductive logic programming for semantic parsing. In *European Conference on Machine Learning*, pages 466–477. Springer.

Xiaojun Xu, Chang Liu, and Dawn Song. 2017. Sqlnet: Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv:1711.04436*.

Navid Yaghmazadeh, Yuepeng Wang, Isil Dillig, and Thomas Dillig. 2017. Sqlizer: query synthesis from natural language. *Proceedings of the ACM on Programming Languages*, 1(OOPSLA):63.

Jie Yang, Yue Zhang, and Fei Dong. 2017. Neural word segmentation with rich pretraining. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 839–849.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1321–1331.

Tao Yu, Michihiro Yasunaga, Kai Yang, Rui Zhang, Dongxu Wang, Zifan Li, and Dragomir Radev. 2018a. Syntaxsqlnet: Syntax tree networks for complex and cross-domain text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1653–1663.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 2018b. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921.

John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the national conference on artificial intelligence*, pages 1050–1055.

Yue Zhang and Jie Yang. 2018. Chinese ner using lattice lstm. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1554–1564.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

# Global Reasoning over Database Structures for Text-to-SQL Parsing

**Ben Bogin**[1] **Matt Gardner**[2] **Jonathan Berant**[1,2]

[1]School of Computer Science, Tel-Aviv University
[2]Allen Institute for Artificial Intelligence
ben.bogin@cs.tau.ac.il, mattg@allenai.org, joberant@cs.tau.ac.il

## Abstract

State-of-the-art semantic parsers rely on auto-regressive decoding, emitting one symbol at a time. When tested against complex databases that are unobserved at training time (zero-shot), the parser often struggles to select the correct set of database constants in the new database, due to the local nature of decoding. In this work, we propose a semantic parser that globally reasons about the structure of the output query to make a more contextually-informed selection of database constants. We use message-passing through a graph neural network to softly select a subset of database constants for the output query, conditioned on the question. Moreover, we train a model to rank queries based on the global alignment of database constants to question words. We apply our techniques to the current state-of-the-art model for SPIDER, a zero-shot semantic parsing dataset with complex databases, increasing accuracy from 39.4% to 47.4%.

## 1 Introduction

The goal of zero-shot semantic parsing (Krishnamurthy et al., 2017; Xu et al., 2017; Yu et al., 2018b,a; Herzig and Berant, 2018) is to map language utterances into executable programs in a new environment, or database (DB). The key difficulty in this setup is that the parser must map new lexical items to DB constants that weren't observed at training time.

Existing semantic parsers handle this mostly through a *local* similarity function between words and DB constants, which considers each word and DB constant in isolation. This function is combined with an auto-regressive decoder, where the decoder chooses the DB constant that is most similar to the words it is currently attending to. Thus, selecting DB constants is done one at a time rather than as a set, and informative global considerations are ignored.

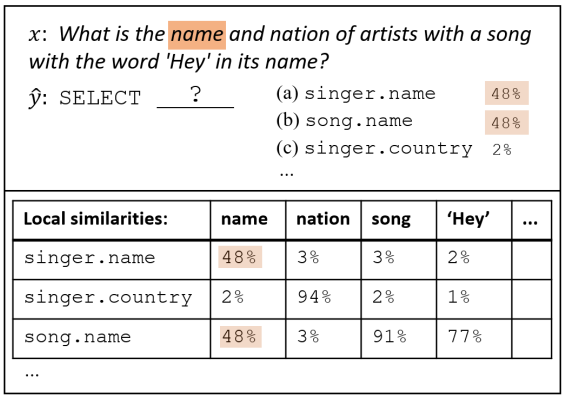


Figure 1: An example where choosing a DB constant based on local similarities is difficult, but the ambiguity can be resolved through global reasoning (see text).

Consider the example in Figure 1, where a question is mapped to a SQL query over a complex DB. After decoding `SELECT`, the decoder must now choose a DB constant. Assuming its attention is focused on the word *'name'* (highlighted), and given local similarities only, the choice between the lexically-related DB constants (`singer.name` and `song.name`) is ambiguous. However, if we globally reason over the DB constants and question, we can combine additional cues. First, a subsequent word *'nation'* is similar to the DB column `country` which belongs to the table `singer`, thus selecting the column `singer.name` from the same table is more likely. Second, the next appearance of the word *'name'* is next to the phrase *'Hey'*, which appears as the value in one of the cells of the column `song.name`. Assuming a one-to-one mapping between words and DB constants, again `singer.name` is preferred.

In this paper, we propose a semantic parser that reasons over the DB structure and question to make a *global* decision about which DB constants should be used in a query. We extend the parser of Bogin et al. (2019), which learns a representation for the



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3659–3664, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

DB schema at parsing time. First, we perform message-passing through a graph neural network representation of the DB schema, to softly select the set of DB constants that are likely to appear in the output query. Second, we train a model that takes the top-$K$ queries output by the auto-regressive model and re-ranks them based on a global match between the DB and the question. Both of these technical contributions can be applied to any zero-shot semantic parser.

We test our parser on SPIDER, a zero-shot semantic parsing dataset with complex DBs. We show that both our contributions improve performance, leading to an accuracy of 47.4%, well beyond the current state-of-the-art of 39.4%.

Our code is available at https://github.com/benbogin/spider-schema-gnn-global.

## 2 Schema-augmented Semantic Parser

**Problem Setup** We are given a training set $\{(x^{(k)}, y^{(k)}, S^{(k)})\}_{k=1}^{N}$, where $x^{(k)}$ is a question, $y^{(k)}$ is its translation to a SQL query, and $S^{(k)}$ is the schema of the corresponding DB. We train a model to map question-schema pairs $(x, S)$ to the correct SQL query. Importantly, the schema $S$ was not seen at training time.

A DB schema $S$ includes : (a) A set of DB tables, (b) a set of columns for each table, and (c) a set of foreign key-primary key column pairs where each pair is a relation from a foreign-key in one table to a primary-key in another. Schema tables and columns are termed *DB constants*, denoted by $\mathcal{V}$.

We now describe a recent semantic parser from Bogin et al. (2019), focusing on the components relevant for selecting DB constants.

**Base Model** The base parser is a standard top-down semantic parser with grammar-based decoding (Xiao et al., 2016; Yin and Neubig, 2017; Krishnamurthy et al., 2017; Rabinovich et al., 2017; Lin et al., 2019). The input question $(x_1, \ldots, x_{|x|})$ is encoded with a BiLSTM, where the hidden states $\boldsymbol{e}_i$ of the BiLSTM are used as contextualized representations for the word $x_i$. The output query $y$ is decoded top-down with another LSTM using a SQL grammar, where at each time step a grammar rule is decoded. Our main focus is decoding of DB constants, and we will elaborate on this part.

The parser decodes a DB constant whenever the previous step decoded the non-terminals `Table` or `Column`. To select the DB constant, it first computes an attention distribution over the question words $\{\alpha_i\}_{i=1}^{|x|}$ in the standard manner (Bahdanau et al., 2015). Then the score for a DB constant $v$ is $s_v = \sum_i \alpha_i s_{\text{link}}(v, x_i)$, where $s_{\text{link}}$ is a local similarity score, computed from learned embeddings of the word and DB constant, and a few manually-crafted features, such as the edit distance between the two inputs and the fraction of string overlap between them. The output distribution of the decoder is simply $\text{softmax}(\{s_v\}_{v \in \mathcal{V}})$. Importantly, the dependence between decoding decisions for DB constants is weak – the similarity function is independent for each constant and question word, and decisions are far apart in the decoding sequence, especially in a top-down parser.

**DB schema encoding** In the zero-shot setting, the schema structure of a new DB can affect the output query. To capture DB structure, Bogin et al. (2019) learned a representation $\boldsymbol{h}_v$ for every DB constant, which the parser later used at decoding time. This was done by converting the DB schema into a graph, where nodes are DB constants, and edges connect tables and their columns, as well as primary and foreign keys (Figure 2, left). A graph convolutional network (GCN) then learned representations $\boldsymbol{h_v}$ for nodes end-to-end (De Cao et al., 2019; Sorokin and Gurevych, 2018).

To focus the GCN's capacity on important nodes, a *relevance probability* $\rho_v$ was computed for every node, and used to "gate" the input to the GCN, conditioned on the question. Specifically, given a learned embedding $\boldsymbol{r}_v$ for every database constant, the GCN input is $\boldsymbol{h}_v^{(0)} = \rho_v \cdot \boldsymbol{r}_v$. Then, the GCN recurrence is applied for $L$ steps. At each step, nodes re-compute their representation based on the representation of their neighbors, where different edge types are associated with different learned parameters (Li et al., 2016). The final representation of each DB constant is $\boldsymbol{h}_v = \boldsymbol{h}_v^{(L)}$.

Importantly, the relevance probability $\rho_v$, which can be viewed as a soft selection for whether the DB constant should appear in the output, was computed based on local information only: First, a distribution $p_{\text{link}}(v \mid x_i) \propto \exp(s_{\text{link}}(v, x_i))$ was defined, and then $\rho_v = \max_i p_{\text{link}}(v \mid x_i)$ was computed deterministically. Thus, $\rho_v$ doesn't consider the full question or DB structure. We address this next.

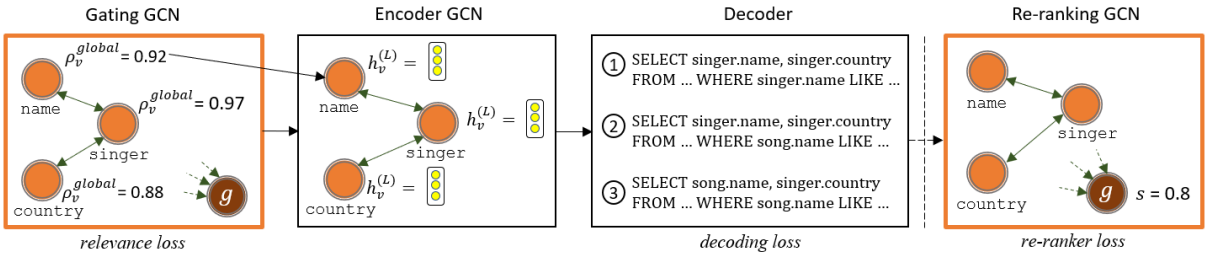


Figure 2: High-level overview, where our contributions are in thick orange boxes. First, a relevance score is predicted for each of the DB constants using the gating GCN. Then, a learned representation is computed for each DB constant using the encoder GCN, which is then used by the decoder to predict $K$ candidates queries. Finally, the re-ranking GCN scores each one of these candidates, basing its score only on the selected DB constants. The dashed line and arrow indicate no gradients are propagated from the re-ranking GCN to the decoder, as the decoder outputs SQL queries. Names of loss terms are written below models that are trained with a loss on their output.

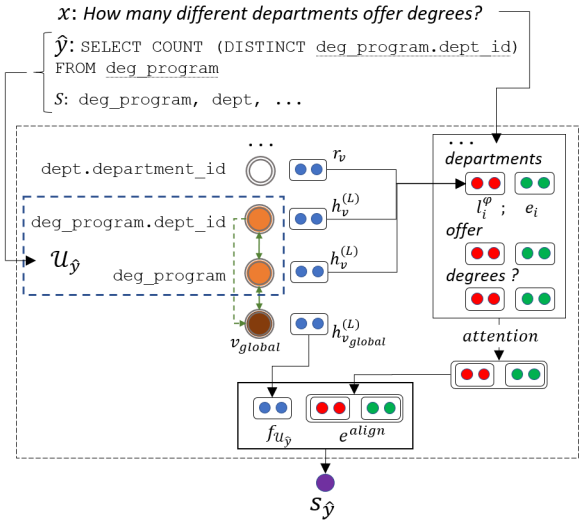


Figure 3: The re-ranking GCN architecture (see text).

## 3 Global Reasoning over DB Structures

Figure 2 gives a high-level view of our model, where the contributions of this paper are marked by thick orange boxes. First, the aforementioned relevance probabilities are estimated with a *learned* gating GCN, allowing global structure to be taken into account. Second, the model discriminatively re-ranks the top-$K$ queries output by the generative decoder.

**Global gating** Bogin et al. (2019) showed that an oracle relevance probability can increase model performance, but computed $\rho_v$ from local information only.

We propose to train a GCN to directly predict $\rho_v$ from the global context of the question and DB.

The input to the gating GCN is the same graph described in §2, except we add a new node $v_{\text{global}}$, connected to all other nodes with a special edge type. To predict the question-conditioned relevance of a node, we need a representation for both the DB constant and the question. Thus, we define the input to the GCN at node $v$ to be $\boldsymbol{g}_v^{(0)} = FF([\boldsymbol{r}_v; \bar{\boldsymbol{h}}_v; \rho_v])$, where ';' is concatenation, $FF(\cdot)$ is a feed-forward network, and $\bar{\boldsymbol{h}}_{\boldsymbol{v}} = \sum_i p_{\text{link}}(x_i \mid v) \cdot \boldsymbol{e}_i$ is a weighted average of contextual representations of question tokens. The initial embedding of $v_{\text{global}}$ is randomly initialized. A relevance probability is computed per DB constant based on the final graph representation: $\rho_v^{\text{global}} = \sigma(FF(\boldsymbol{g}_v^{(L)}))$. This probability replaces $\rho_v$ at the input to the encoder GCN (Figure 2).

Because we have the gold query $y$ for each question, we can extract the gold subset of DB constants $\mathcal{U}_y$, i.e., all DB constants that appear in $y$. We can now add a *relevance loss* term $-\sum_{v \in \mathcal{U}_y} \log \rho_v^{\text{global}} - \sum_{v \notin \mathcal{U}_y} \log(1 - \rho_v^{\text{global}})$ to the objective. Thus, the parameters of the gating GCN are trained from the relevance loss and the usual *decoding loss*, a ML objective over the gold sequence of decisions that output the query $y$.

**Discriminative re-ranking** Global gating provides a more accurate model for softly predicting the correct subset of DB constants. However, parsing is still auto-regressive and performed with a local similarity function. To overcome this, we separately train a discriminative model (Collins and Koo, 2005; Ge and Mooney, 2006; Lu et al., 2008; Fried et al., 2017) to re-rank the top-$K$ queries in the decoder's output beam. The re-ranker scores each candidate tuple $(x, S, \hat{y})$, and thus can globally reason over the entire candidate query $\hat{y}$.

We focus the re-ranker capacity on the main pain point of zero-shot parsing – the set of DB constants $\mathcal{U}_{\hat{y}}$ that appear in $\hat{y}$. At a high-level (Figure 3), for each candidate we compute a logit $s_{\hat{y}} = \boldsymbol{w}^\top FF(\boldsymbol{f}_{\mathcal{U}_{\hat{y}}}, \boldsymbol{e}^{\text{align}})$, where $\boldsymbol{w}$ is a learned

parameter vector, $\boldsymbol{f}_{\mathcal{U}_{\hat{y}}}$ is a representation for the set $\mathcal{U}_{\hat{y}}$, and $\boldsymbol{e}^{\text{align}}$ is a representation for the global alignment between question words and DB constants. The re-ranker is trained to minimize the *re-ranker loss*, the negative log probability of the correct query $y$. We now describe the computation of $\boldsymbol{f}_{\mathcal{U}_{\hat{y}}}$ and $\boldsymbol{e}^{\text{align}}$, based on a re-ranking GCN.

Unlike the gating GCN, the re-ranking GCN takes as input only the sub-graph induced by the selected DB constants $\mathcal{U}_{\hat{y}}$, and the global node $v_{\text{global}}$. The input is represented by $\boldsymbol{f}_v^{(0)} = FF(\boldsymbol{r}_v; \bar{\boldsymbol{h}}_v)$, and after L propagation steps we obtain $\boldsymbol{f}_{\mathcal{U}_{\hat{y}}} = \boldsymbol{f}^{(L)_{v_{\text{global}}}}$. Note that the global node representation is used to describe and score the question-conditioned sub-graph, unlike the gating GCN where the global node mostly created shorter paths between other graph nodes.

The representation $\boldsymbol{f}_{\mathcal{U}_{\hat{y}}}$ captures global properties of selected nodes but ignores nodes that were not selected and are possibly relevant. Thus, we compute a representation $\boldsymbol{e}^{\text{align}}$, which captures whether question words are aligned to selected DB constants. We define a representation for every node $v \in \mathcal{V}$:

$$\boldsymbol{\varphi}_v = \begin{cases} \boldsymbol{f}_v^{(L)} & \text{if } v \in \mathcal{U}_{\hat{y}} \\ \boldsymbol{r}_v & \text{otherwise} \end{cases}$$

Now, we compute for every question word $x_i$ a representation of the DB constants it aligns to: $\boldsymbol{l}_i^{\varphi} = \sum_{v \in \mathcal{V}} p_{\text{link}}(v \mid x_i) \cdot \boldsymbol{\varphi}_v$. We concatenate this representation to every word $\boldsymbol{e}_i^{\text{align}} = [\boldsymbol{e}_i; \boldsymbol{l}_i^{\varphi}]$, and compute the vector $\boldsymbol{e}^{\text{align}}$ using attention over the question words, where the attention score for every word is $\boldsymbol{e}_i^{\text{align}\top} \boldsymbol{w}_{\text{att}}$ for a learned vector $\boldsymbol{w}_{\text{att}}$. The goal of this term is to allow the model to recognize whether there are any attended words that are aligned with DB constants, but these DB constants were not selected in $\mathcal{U}_{\hat{y}}$.

In sum, our model adds a gating GCN trained to softly select relevant nodes for the encoder, and a re-ranking GCN that globally reasons over the subset of selected DB constants, and captures whether the query properly covers question words.

## 4 Experiments and Results

**Experimental setup** We train and evaluate on SPIDER (Yu et al., 2018b), which contains 7,000/1,034/2,147 train/development/test examples, using the same pre-processing as Bogin et al. (2019). To train the re-ranker, we take $K = 40$ candidates from the beam output of the decoder. At each training step, if the gold query is in the beam, we calculate the loss on the gold query and 10 randomly selected negative candidates. At test time, we re-rank the best $K = 10$ candidates in the beam, and break re-ranking ties using the auto-regressive decoder scores (ties happen since the re-ranker considers the DB constants only and not the entire query). We use the official SPIDER script for evaluation, which tests for loose exact match of queries.

| Model | Accuracy |
|---|---|
| SYNTAXSQLNET | 19.7% |
| GNN | 39.4% |
| **GLOBAL-GNN** | **47.4%** |

Table 1: Test set accuracy of GLOBAL-GNN compared to prior work on SPIDER.

| Model | Acc. | Beam | SINGLE | MULTI |
|---|---|---|---|---|
| SYNTAXSQLNET | 18.9% | | 23.1% | 7.0% |
| GNN | 40.7% | | 52.2% | 26.8% |
| + RE-IMPLEMENTATION | 44.1% | 62.2% | 58.3% | 27.6% |
| **GLOBAL-GNN** | **52.1%** | **65.9 %** | **61.6%** | **40.3%** |
| - NO GLOBAL GATING | 48.8% | 62.2% | 60.9% | 33.8% |
| - NO RE-RANKING | 48.3% | **65.9%** | 58.1% | 36.8% |
| - NO RELEVANCE LOSS | 50.1% | 64.8% | 60.9% | 36.6% |
| NO ALIGN REP. | 50.8% | **65.9%** | 60.7% | 38.3% |
| QUERY RE-RANKER | 47.8% | **65.9%** | 55.3% | 38.3% |
| ORACLE RELEVANCE | 56.4% | 73.5% | | |

Table 2: Development set accuracy for various experiments. The column 'Beam' indicates the fraction of examples where the gold query is in the beam ($K = 10$).

**Results** As shown in Table 1, the accuracy of our proposed model (GLOBAL-GNN) on the hidden test set is 47.4%, 8% higher than current state-of-the art of 39.4%. Table 2 shows accuracy results on the development set for different experiments. We perform minor modifications to the implementation of Bogin et al. (2019), improving the accuracy from 40.7% to 44.1% (details in appendix A). We follow Bogin et al. (2019), measuring accuracy on easier examples where queries use a single table (SINGLE) and those using more than one table (MULTI).

GLOBAL-GNN obtains 52.1% accuracy on the development set, substantially higher than all previous scores. Importantly, the performance increase comes mostly from queries that require more than one table, which are usually more complex.

Removing any of our two main contributions (NO GLOBAL GATING, NO RE-RANKING) leads to a 4% drop in performance. Training without the relevance loss (NO RELEVANCE LOSS) results in a 2% accuracy degrade. Omitting the representation $e^{\text{align}}$ from the re-ranker (NO ALIGN REP.)

reduces performance, showing the importance of identifying unaligned question words.

We also consider a model that ranks the entire query and not only the set of DB constants. We re-define $s_{\hat{y}} = \boldsymbol{w}^\top FF(\boldsymbol{f}_{\mathcal{U}_{\hat{y}}}, \boldsymbol{h}^{\text{align}}, \boldsymbol{h}^{\text{query}})$, where $\boldsymbol{h}^{\text{query}}$ is a concatenation of the last and first hidden states of a BiLSTM run over the output SQL query (QUERY RE-RANKER). We see performance is lower, and most introduced errors are minor mistakes such as `min` instead of `max`. This shows that our re-ranker excels at choosing DB constants, while the decoder is better at determining the SQL query structure and the SQL logical constants.

Finally, we compute two oracle scores to estimate future headroom. Assuming a perfect global gating, which gives probability 1.0 iff the DB constant is in the gold query, increases accuracy to 63.2%. Adding to that a perfect re-ranker leads to an accuracy of 73.5%.

**Qualitative analysis** Analyzing the development set, we find two main re-occurring patterns, where the baseline model is wrong, but our parser is correct. (a) *coverage*: when relevant question words are not covered by the query, which results in a missing joining of tables or selection of columns (b) *precision*: when unrelated tables are joined to the query due to high lexical similarity. Selected examples are in Appendix B.

**Error analysis** In 44.4% of errors where the correct query was in the beam, the selection of $\mathcal{U}$ was correct but the query was wrong. Most of these errors are caused by minor local errors, e.g., `min`/`max` errors, while the rest are due to larger structural mistakes, indicating that a global model that jointly selects both DB constants and SQL tokens might further improve performance. Other types of errors include missing or extra columns and tables, especially in complex queries.

## 5 Conclusion

In this paper, we demonstrate the importance of global decision-making for zero-shot semantic parsing, where selecting the relevant set of DB constants is challenging. We present two main technical contributions. First, we use a gating GCN that globally attends the input question and the entire DB schema to softly-select the relevant DB constants. Second, we re-rank the output of a generative semantic parser by globally scoring the set of selected DB-constants. Importantly, these contributions can be applied to any zero-shot semantic parser with minimal modifications. Empirically, we observe a substantial improvement over the state-of-the-art on the SPIDER dataset, showing the effectiveness of both contributions.

## Acknowledgments

This research was partially supported by The Yandex Initiative for Machine Learning. This work was completed in partial fulfillment for the Ph.D degree of the first author.

## References

D. Bahdanau, K. Cho, and Y. Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*.

B. Bogin, M. Gardner, and J. Berant. 2019. Representing schema structure with graph neural networks for text-to-sql parsing. In *Association for Computational Linguistics (ACL)*.

M. Collins and T. Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–70.

N. De Cao, W. Aziz, and I. Titov. 2019. Question answering by reasoning across documents with graph convolutional networks. In *North American Association for Computational Linguistics (NAACL)*.

D. Fried, M. Stern, and D. Klein. 2017. Improving neural parsing by disentangling model combination and reranking effects. In *Association for Computational Linguistics (ACL)*.

R. Ge and R. J. Mooney. 2006. Discriminative reranking for semantic parsing. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 263–270.

J. Herzig and J. Berant. 2018. Decoupling structure and lexicon for zero-shot semantic parsing. In *Empirical Methods in Natural Language Processing (EMNLP)*.

J. Krishnamurthy, P. Dasigi, and M. Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel. 2016. Gated graph sequence neural networks. In *International Conference on Learning Representations (ICLR)*.

K. Lin, B. Bogin, M. Neumann, J. Berant, and M. Gardner. 2019. Grammar-based neural text-to-sql generation. *arXiv preprint arXiv:1905.13326*.

W. Lu, H. T. Ng, W. S. Lee, and L. S. Zettlemoyer. 2008. A generative model for parsing natural language to meaning representations. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 783–792, Honolulu, Hawaii. Association for Computational Linguistics.

M. Rabinovich, M. Stern, and D. Klein. 2017. Abstract syntax networks for code generation and semantic parsing. In *Association for Computational Linguistics (ACL)*.

D. Sorokin and I. Gurevych. 2018. Modeling semantics with gated graph neural networks for knowledge base question answering. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3306–3317, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

C. Xiao, M. Dymetman, and C. Gardent. 2016. Sequence-based structured prediction for semantic parsing. In *Association for Computational Linguistics (ACL)*.

X. Xu, C. Liu, and D. Song. 2017. Sqlnet: Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv:1711.04436*.

P. Yin and G. Neubig. 2017. A syntactic neural model for general-purpose code generation. In *Association for Computational Linguistics (ACL)*, pages 440–450.

T. Yu, M. Yasunaga, K. Yang, R. Zhang, D. Wang, Z. Li, and D. Radev. 2018a. SyntaxSQLNet: Syntax tree networks for complex and cross-domaintext-to-SQL task. In *Empirical Methods in Natural Language Processing (EMNLP)*.

T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman, et al. 2018b. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. *arXiv preprint arXiv:1809.08887*.

# Transductive Learning of Neural Language Models for Syntactic and Semantic Analysis

**Hiroki Ouchi**[1,2] **Jun Suzuki**[2,1] **Kentaro Inui**[2,1]
[1] RIKEN Center for Advanced Intelligence Project [2] Tohoku University
hiroki.ouchi@riken.jp, {jun.suzuki,inui}@ecei.tohoku.ac.jp

## Abstract

In transductive learning, an unlabeled test set is used for model training. While this setting deviates from the common assumption of a completely unseen test set, it is applicable in many real-world scenarios, where the texts to be processed are known in advance. However, despite its practical advantages, transductive learning is underexplored in natural language processing. Here, we conduct an empirical study of transductive learning for neural models and demonstrate its utility in syntactic and semantic tasks. Specifically, we fine-tune language models (LMs) on an unlabeled test set to obtain test-set-specific word representations. Through extensive experiments, we demonstrate that despite its simplicity, transductive LM fine-tuning consistently improves state-of-the-art neural models in both in-domain and out-of-domain settings.

## 1 Introduction

In supervised learning, a model is trained on a training set and its generalization performance is evaluated on an unseen test set. In this setting, the model has no access to the test set during training. However, the assumption of a completely unseen test set is not always necessary. In many cases, certain aspects of the test set are already known at training time. For example, a company may want to annotate a large number of existing documents automatically (Section 3). In such a scenario, the texts to be processed are known in advance, and using the model trained on the texts themselves to process them can be more efficient. Using an unlabeled test set in this way is the key idea behind *transductive learning*.

In transductive learning (Vapnik, 1998), an unlabeled test set is given in the training phase. That is, the inputs of the test set, i.e., the raw texts, can be used during training, but the labels are never used. In the test phase, the trained model is evaluated on the same test set. Despite its practical advantages, transductive learning has received little attention in natural language processing (NLP). After the pioneering work of Joachims (1999), who proposed a transductive support vector machine for text classification, transductive methods for linear models have been investigated in only a few tasks, such as lexical acquisition (Duh and Kirchhoff, 2006) and machine translation (Ueffing et al., 2007). In particular, transductive learning with neural networks is underexplored.

Here, we investigate the impact of transductive learning on state-of-the-art neural models in syntactic and semantic tasks, namely syntactic chunking and semantic role labeling (SRL). Specifically, inspired by recent findings that language model (LM)-based word representations yield large performance improvement (Devlin et al., 2019), we fine-tune Embeddings from Language Models (ELMo) (Peters et al., 2018) on an unlabeled test set and use them in each task-specific model. Typically, LMs are trained on a large-scale corpus whose word distributions are different from the test set. By contrast, transductive learning allows us to fit LMs directly to the distributions of the test set. Our experiments show the effectiveness of transductive LM fine-tuning.

In summary, our main contributions are:

- This work is the first to introduce an LM fine-tuning method to transductive learning[1].
- Through extensive experiments in both in-domain and out-of-domain settings, we demonstrate that transductive LM fine-tuning consistently improves state-of-the-art neural models in syntactic and semantic tasks.

[1]Our code and scripts are publicly available at https://github.com/hiroki13/transductive-language-models.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3665–3671, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

## 2 Related Work

**Transductive learning.** Vapnik advocated and formalized transductive learning (Vapnik, 1998; Gammerman et al., 1998), which has been applied to text classification (Joachims, 1999; Ifrim and Weikum, 2006) and image processing (Bruzzone et al., 2006; Sener et al., 2016; Liu et al., 2019). Although some studies have presented transductive methods for linear models in other tasks (Duh and Kirchhoff, 2006; Ueffing et al., 2007; Chen et al., 2008; Alexandrescu and Kirchhoff, 2009), transductive methods for neural models are underexplored in NLP.

**Unsupervised domain adaptation.** Transductive learning is related to unsupervised domain adaptation, in which models are adapted to a target domain by using unlabeled target domain texts (Ben-David et al., 2010; Shi and Sha, 2012). This setting does not allow models to access the test set, which is the main difference between unsupervised domain adaptation and transductive learning. Various unsupervised adaptation methods have been proposed for linear models (Blitzer et al., 2006; Jiang and Zhai, 2007; Tsuboi et al., 2009; Søgaard, 2013). In the context of neural models, adversarial domain adaptation (Ganin and Lempitsky, 2015; Ganin et al., 2016; Guo et al., 2018), importance weighting (Wang et al., 2017), structural correspondence learning (Ziser and Reichart, 2017), self/tri/co-training (Saito et al., 2017; Ruder and Plank, 2018), and other techniques orthogonal to transductive LM fine-tuning have been applied successfully in unsupervised domain adaptation[2]. Integrating these methods with transductive LM fine-tuning is an interesting direction for future research.

**LM-based word representations.** Recently, LM-based word representations pre-trained on unlabeled data have gained considerable attention (Peters et al., 2018; Radford et al., 2018; Devlin et al., 2019). The most related method to ours is Universal Language Model Fine-tuning (ULMFiT), which pre-trains an LM on a large general-domain corpus and fine-tunes it on the target task (Howard and Ruder, 2018). Inspired by these studies, we introduce LM-based word representation in transductive learning.

[2]Feature augmentation is considered a *supervised* domain adaptation method (Daume III, 2007; Kim et al., 2016).

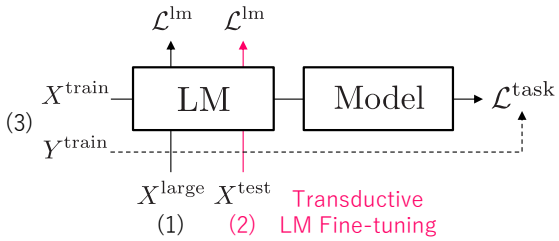


Figure 1: Training procedure. (1) LM pre-training: the LM is firstly pre-trained on the large-scale unlabeled corpus $\mathcal{D}^{\text{large}} = \{X_i^{\text{large}}\}_{i=1}^{N^{\text{large}}}$. (2) Transductive LM fine-tuning: the LM is then fine-tuned on the unlabeled test set $\mathcal{D}^{\text{test}} = \{X_i^{\text{test}}\}_{i=1}^{N^{\text{test}}}$. Note that the test set used for training is the identical one used in evaluation. (3) Task-specific model training: the task-specific model is trained on the training set $\mathcal{D}^{\text{train}} = \{(X_i^{\text{train}}, Y_i^{\text{train}})\}_{i=1}^{N^{\text{train}}}$. $\mathcal{L}$ denotes the loss function.

## 3 Neural Transductive Learning

**Motivation.** Suppose that a company has received a vast amount of customer reviews and wants to automatically process these reviews more accurately, even if it takes some time. For this purpose, they do not have to build a model that works well on new unseen reviews. Instead, they want a model that works well on only the reviews in hand. In this situation, using these reviews themselves to train a model can be more efficient. This is the key motivation for developing effective and practical transductive learning methods. Toward this goal, we develop transductive methods for state-of-the-art neural models.

**Problem formulation.** In the training phase, a training set $\mathcal{D}^{\text{train}} = \{(X_i^{\text{train}}, Y_i^{\text{train}})\}_{i=1}^{N^{\text{train}}}$ and an unlabeled test set $\mathcal{D}^{\text{test}} = \{X_i^{\text{test}}\}_{i=1}^{N^{\text{test}}}$ are used for model training, where $X_i$ is an input, e.g., a sentence, and $Y_i$ represents target labels, e.g., labels from a set of syntactic or semantic annotations. In the test phase, the trained model is used for predicting labels and is evaluated on the same test set $\mathcal{D}^{\text{test}}$.

**Method.** We present a simple transductive method for neural models. Specifically, we fine-tune an LM on an unlabeled test set. Figure 1 illustrates the training procedure that consists of the following steps: (1) LM pre-training, (2) Transductive LM fine-tuning and (3) task-specific model training. We first train an LM on a large-scale unlabeled corpus $\mathcal{D}^{\text{large}}$ and then fine-tune the LM on an unlabeled test set $\mathcal{D}^{\text{test}}$. Finally, we use the fine-tuned LM as the embedding layer of

| | Training | | Development | | Test | |
|---|---|---|---|---|---|---|
| | Sents | Preds | Sents | Preds | Sents | Preds |
| BC | 11.8k | 28.9k | 2.1k | 5.0k | 2,0k | 5.4k |
| BN | 10.6k | 3.1k | 1.2k | 3,9k | 1.2k | 3.7k |
| MZ | 6.9k | 2.4k | 0.6k | 2.1k | 0.7k | 2.6k |
| NW | 34.9k | 96.6k | 5.8k | 16.6k | 1.8k | 5.8k |
| PT | 21.5k | 34.9k | 1.7k | 2.5k | 1.2k | 2.8k |
| TC | 12.8k | 16.2k | 1.6k | 2.0k | 1.3k | 1.7k |
| WB | 16.9k | 20.0k | 2.3k | 2.8k | 0.9k | 2.2k |

Table 1: Dataset statistics on the CoNLL-2012 dataset. 1k = 1,000. Column "Sents" denotes the number of sentences in each dataset. Column "Preds" denotes the number of predicates in each dataset.

each task-specific model and train the model on a training set $\mathcal{D}^{\text{train}}$.

$$\Theta' \leftarrow \text{argmin}_{\Theta} \mathcal{L}^{\text{lm}}(\Theta|\mathcal{D}^{\text{large}}), \quad (1)$$

$$\Theta'' \leftarrow \text{argmin}_{\Theta'} \mathcal{L}^{\text{lm}}(\Theta'|\mathcal{D}^{\text{test}}), \quad (2)$$

$$\Phi' \leftarrow \text{argmin}_{\Phi} \mathcal{L}^{\text{task}}(\Phi|\Theta'', \mathcal{D}^{\text{train}}). \quad (3)$$

Here, $\mathcal{L}^{\text{lm}}$ and $\mathcal{L}^{\text{task}}$ are the loss functions for an LM and task-specific model, respectively.[3] In the LM pre-training and fine-tuning phases (Eqs. 1 and 2), we first train the initial LM parameters $\Theta$ and then fine-tune the pre-trained parameters $\Theta'$. In the task-specific training phase (Eq. 3), we fix the fine-tuned LM parameters $\Theta''$ used for the embedding layer of a task-specific model, and train only the task-specific model parameters $\Phi$.

## 4 Experiments

**Tasks.** To investigate the effectiveness of transductive LM fine-tuning for syntactic and semantic analysis, we conduct experiments in syntactic chunking (Ramshaw and Marcus, 1999; Sang and Buchholz, 2000; Ponvert et al., 2011) and SRL (Gildea and Jurafsky, 2002; Palmer et al., 2005; Carreras and Màrquez, 2005)[4]. The goal of syntactic chunking is to divide a sentence into non-overlapping phrases that consist of syntactically related words. The goal of SRL is to identify semantic arguments for each predicate. For example, consider the following sentence:

| | The | man | kept | a | cat |
|---|---|---|---|---|---|
| SynChunk | [ NP ] | | | [ NP ] | |
| SemRole | [ A0 ] | | | [ A1 ] | |

In syntactic chunking, given the input sentence, systems have to recognize "The man" and "a cat" as noun phrases (NP). In SRL, given the input sentence and the target predicate "kept", systems have to recognize "The man" as the A0 argument and "a cat" as the A1 argument. For syntactic chunking, we adopted the experimental protocol by Ponvert et al. (2011) and for SRL, we followed Ouchi et al. (2018) (details in Appendix A).

**Datasets.** We perform experiments using the CoNLL-2012 dataset[5]. To investigate the performances under in-domain and out-of-domain settings, we use each of the seven domains in the CoNLL-2012 dataset. Table 1 shows the data statistics. Each test set contains at most 2,000 sentences. Compared with previous studies, such as Xiao and Guo (2013) that used 570,000 sentences as unlabeled data for unsupervised domain adaptation of syntactic chunking, our transductive experiments can be regarded as a low-resource adaptation setting. As a large-scale unlabeled raw corpus for LM training, we use the 1B word benchmark corpus (Chelba et al., 2013).

**Model setup.** We use ELMo (Peters et al., 2018) as an LM. For syntactic chunking, we use a variant of the Reconciled Span Parser (Joshi et al., 2018). For SRL, we use the span selection model (BiLSTM-Span model) (Ouchi et al., 2018). Each model is trained on a source domain training set and was evaluated on a target domain test set[6]. The development set is also the source domain, and it is used for hyperparameter tuning[7]. Consider the case where NW $\rightarrow$ BC, i.e., the source domain is the newswire NW and the target domain is the broadcast conversation BC. We first train ELMo on the large-scale raw corpus (one billion word benchmark corpus) and fine-tune it on the BC test set. We then train syntactic and semantic models that use the fine-tuned ELMo on the NW training set. During the task-specific model training, we freeze the fine-tuned ELMo. We select hyperparameters by using the NW development set. Finally, we evaluate the trained model on the BC test set. In the same way, we conduct training and evaluation for each domain pair.

[3] In our experiments (Section 4), both losses were given by the negative log-likelihood (Appendix A).

[4] This paper addresses span-based, PropBank-style SRL. Detailed descriptions on other lines of SRL research (e.g. dependency-based SRL and FrameNet-based SRL) can be found in Baker et al. (1998); Das et al. (2014); Surdeanu et al. (2008); Hajič et al. (2009).

[5] We used the version of OntoNotes downloaded at: http://cemantix.org/data/ontonotes.html.

[6] We used the official evaluation scripts downloaded at https://www.clips.uantwerpen.be/conll2000/chunking/ and http://www.lsi.upc.edu/ srlconll/soft.html.

[7] All models and hyperparameters are described in Appendices B, C, and D.

| src → tgt | BC | BN | MZ | NW | PT | TC | WB | Averaged F1 |
|---|---|---|---|---|---|---|---|---|
| | | | | SYNTACTIC CHUNKING | | | | |
| BC | 93.0 / **93.5** | 92.9 / **93.0** | 90.0 / **90.6** | 88.1 / **88.7** | 94.1 / **94.9** | 84.5 / **85.1** | 89.4 / **89.8** | 90.3 / **90.8** |
| BN | 92.5 / **93.0** | 94.7 / **95.0** | 91.2 / **91.4** | 90.0 / **90.6** | 94.8 / **95.3** | 84.0 / **84.9** | 89.9 / **90.7** | 91.0 / **91.6** |
| MZ | 89.9 / **90.8** | 91.6 / **92.3** | 92.3 / **92.5** | 89.2 / **90.0** | 93.4 / **94.3** | 80.5 / **82.2** | 89.9 / **90.9** | 89.5 / **90.4** |
| NW | 91.4 / **92.0** | 93.7 / **93.9** | 92.2 / **92.6** | 94.2 / **94.5** | 95.7 / **96.1** | 83.3 / **84.2** | 92.3 / **92.9** | 91.8 / **92.3** |
| PT | 87.1 / **88.2** | 86.9 / **87.5** | 85.6 / **86.9** | 81.0 / **82.7** | 97.5 / **97.7** | 79.0 / **80.0** | 86.9 / **88.1** | 86.3 / **87.3** |
| TC | 87.3 / **88.2** | 87.2 / **87.5** | 84.1 / **85.4** | 80.8 / **82.3** | 93.0 / **93.7** | 89.3 / **89.5** | 85.4 / **86.5** | 86.7 / **87.6** |
| WB | 91.8 / **92.3** | 93.4 / **93.7** | 91.7 / **92.2** | 91.0 / **91.5** | 95.6 / **96.0** | 83.6 / **85.1** | 93.0 / **93.5** | 91.4 / **92.0** |
| | | | | SEMANTIC ROLE LABELING | | | | |
| BC | 83.3 / **83.9** | 78.9 / **79.3** | 74.2 / **74.8** | 71.0 / **72.4** | 82.8 / **84.4** | 80.2 / **80.6** | 78.7 / **79.8** | 78.4 / **79.3** |
| BN | 80.3 / **81.2** | 83.3 / **83.5** | 76.5 / **77.4** | 75.0 / **75.7** | 86.5 / **86.8** | 77.1 / **78.0** | 78.8 / **79.9** | 79.6 / **80.4** |
| MZ | 76.4 / **77.3** | 76.6 / **77.3** | 80.2 / **80.6** | 73.8 / **74.8** | 84.8 / **87.2** | 72.8 / **73.3** | 77.5 / **78.7** | 77.4 / **78.5** |
| NW | 79.2 / **80.1** | 79.8 / **80.0** | 79.5 / **80.0** | 83.8 / **84.4** | 88.3 / **89.0** | 75.5 / **76.5** | 81.1 / **81.8** | 81.0 / **81.7** |
| PT | 71.2 / **72.1** | 67.4 / **67.8** | 66.6 / **68.0** | 64.7 / **66.0** | 92.8 / **93.0** | 72.6 / **73.9** | 76.2 / **77.2** | 73.1 / **74.0** |
| TC | 73.8 / **74.1** | 67.6 / **67.8** | 64.5 / **64.9** | 59.2 / **60.2** | 79.0 / **80.4** | 83.3 / **83.6** | 71.3 / **72.5** | 71.2 / **71.9** |
| WB | 74.1 / **74.4** | 71.7 / **72.4** | 72.0 / **72.8** | 71.4 / **72.0** | 87.8 / **88.8** | 76.3 / **76.7** | 81.8 / **82.4** | 76.4 / **77.1** |

Table 2: Main results under cross-domain settings, src ("source", training set) → tgt ("target", test set). Cells show the F1 scores of the baseline model (before the slash) and the transductive model (after the slash). Column "Averaged F1" represents the F1 scores averaged across the target domains. Domains are as follows: BC = Broadcast Conversation, BN = Broadcast News, MZ = Magazine, NW = Newswire, PT = New Testament, TC = Telephone Conversation, and WB = Weblogs and Newsgroups.

**Results.** Table 2 shows the F1 scores on each test set. All reported F1 scores are the average of five distinct trials using different random seeds. In each cell, the left-hand side denotes the F1 score of the baseline (using a base LM without fine-tuning) and the right-hand side denotes F1 of the transductive models (using a fine-tuned LM on each test set). In in-domain (same source/target domains, e.g., BC→BC) and out-of-domain (different source/target domains, e.g., BC→NW) settings, all transductive models consistently outperformed the baselines, which suggests that transductive LM fine-tuning improves performance of neural models. Although the improvements were undramatic (around 1.0 F1 gain), these consistent improvements can be regarded as valuable empirical results because of the difficulty of *unsupervised* and *low-resource* adaptation settings.

## 5 Analysis

**Comparison between unsupervised domain adaptation and transduction.** In unsupervised domain adaptation, target domain unlabeled data (the texts whose domain is the same as that of a test set) is used for adaptation. Although the domain is identical between target domain data and a test set, their word distributions are somewhat different. In transductive learning, because an unlabeled test set can be used for training, it is possible to adapt LMs directly to the word distributions of the test set. Here, we investigate whether adapting LMs directly to each test set is more effective than adapting LMs to each target domain unlabeled data. Similarly to our transductive method shown in Figure 1, we first train LMs on the large-scale unlabeled corpus (the 1B word benchmark corpus) and then fine-tune them on the unlabeled target domain data[8]. In addition, we control the sizes of the target domain unlabeled data and test sets. That is, we use the same number of sentences in the unlabeled data of each target domain as in each test set. Table 3 shows the F1 scores averaged across all the target domains. The transductive models (T) consistently outperformed the domain-adapted models (CU). This demonstrates that adapting LMs directly to test sets is more effective than adapting them to target domain unlabeled data.

| | Syntactic chunking | | Semantic role labeling | |
|---|---|---|---|---|
| | CU | T | CU | T |
| BC | 90.4 | **90.8** | 78.6 | **79.3** |
| BN | 91.1 | **91.6** | 79.8 | **80.4** |
| MZ | 90.0 | **90.4** | 77.9 | **78.5** |
| NW | 92.1 | **92.3** | 81.1 | **81.7** |
| PT | 87.1 | **87.3** | 73.5 | **74.0** |
| TC | 87.1 | **87.6** | 71.3 | **71.6** |
| WB | 91.8 | **92.0** | 76.6 | **77.1** |

Table 3: Performance comparison between LM fine-tuning on target domain unlabeled data of the same size as each test set, "Controlled Unlabeled data (CU)," and transductive LM fine-tuning on each test set (T). Cells show the F1 scores averaged across the target domains.

[8] As target domain unlabeled data, we use the CoNLL-2012 training set of each domain.

| | Syntactic chunking | | Semantic role labeling | |
|---|---|---|---|---|
| | U | U + T | U | U + T |
| BC | 90.5 | **91.0** | 79.0 | **79.4** |
| BN | 91.3 | **91.6** | 80.1 | **80.6** |
| MZ | 90.2 | **90.6** | 78.3 | **78.7** |
| NW | 92.1 | **92.5** | 81.5 | **81.9** |
| PT | 87.3 | **87.7** | 73.6 | **74.3** |
| TC | 87.2 | **87.6** | 71.4 | **72.0** |
| WB | 91.8 | **92.2** | 76.8 | **77.2** |

Table 4: Performance comparison between LM fine-tuning on target domain unlabeled data (U) and on the combination of the unlabeled data and test sets (U + T). Cells show the F1 scores averaged across the target domains.

| CoNLL | 2000 | 2005 | | 2012 |
|---|---|---|---|---|
| | | WSJ | Brown | |
| BASE | 96.6 | 87.7 | 78.3 | 86.2 |
| TRANS | **96.7** | **87.9*** | **79.5*** | **86.6*** |
| Clark et al. (2018) | 97.0 | - | - | - |
| Peters et al. (2017) | 96.4 | - | - | - |
| Hashimoto et al. (2017) | 95.8 | - | - | - |
| Wang et al. (2019) | - | 88.2 | 79.3 | 86.4 |
| Li et al. (2019) | - | 87.7 | 80.5 | 86.0 |
| Ouchi et al. (2018) | - | 87.6 | 78.7 | 86.2 |
| He et al. (2018) | - | 87.4 | 80.4 | 85.5 |

Table 5: Standard benchmark results. Cells show the F1 scores on each test set. The CoNLL-2000 and CoNLL-2005/2012 datasets are used for syntactic chunking and SRL, respectively. Results of the transductive models (TRANS) marked with * are statistically significant compared to the baselines (BASE) using the permutation test ($p < 0.05$).

**Combination of unsupervised domain adaptation and transduction.** In real-world situations, large-scale unlabeled data of target domains is sometimes available. In such cases, LMs can be trained on both the target domain unlabeled data and the test sets. Here, we investigate the effectiveness of using both datasets. Table 4 shows the F1 scores averaged across all the target domains. Fine-tuning the LMs on the target domain unlabeled data as well as each test set (U + T) showed better performance than fine-tuning them only on the target domain unlabeled data (U). This combination of tranduction with unsupervised domain adaptation further improves performance.

**Effects in standard benchmarks.** Some studies indicated that when promising new techniques are only evaluated on very basic models, determining how much (if any) improvement will carry over to stronger models can be difficult (Denkowski and Neubig, 2017; Suzuki et al., 2018). Motivated by such studies, we provide the results in standard benchmark settings. For syntactic chunking, we use the CoNLL-2000 dataset (Sang and Buchholz, 2000) and follow the standard experimental protocol (Hashimoto et al., 2017). For SRL, we use the CoNLL-2005 (Carreras and Màrquez, 2005) and CoNLL-2012 datasets (Pradhan et al., 2012) and follow the standard experimental protocol (Ouchi et al., 2018). Table 5 shows the F1 scores of our models and those of existing models. The results of the baseline model were comparable with those of the state-of-the-art models, and the transductive model consistently outperformed the baseline model[9]. Note that we cannot fairly compare the transductive and existing models due to the difference in settings. These results, however, demonstrate that transductive LM fine-tuning improves state-of-the-art chunking and SRL models.

## 6 Conclusion

In this study, we investigated the impact of transductive learning on state-of-the-art neural models in syntactic and semantic tasks. Specifically, we fine-tuned an LM on an unlabeled test set. Through extensive experiments, we demonstrated that, despite its simplicity, transductive LM fine-tuning contributes to consistent performance improvement of state-of-the-art syntactic and semantic models in cross-domain settings. One interesting line of future work is to explore effective transductive methods for task-dependent (neural) layers. For instance, as some unsupervised domain adaptation methods can be applied to transductive learning, integrating them with transductive LM fine-tuning may further improve their performance. Another line of our future work is to apply these transductive methods to various NLP tasks and investigate their performance.

## Acknowledgments

This work was partially supported by JSPS KAKENHI Grant Number JP19H04162 and JP19K20351. We would like to thank Benjamin Heinzerling, Ana Brassard, Sosuke Kobayashi, Hitomi Yanaka, and the anonymous reviewers for their insightful comments.

[9]While the improvements in SRL were statistically significant compared to the baseline, the improvement in syntactic chunking was not. One reason for this is that the F1 score of the baseline in syntactic chunking is already high and there is less room for improvement. Since Clark et al. (2018) achieved 97.0 F1 with multi-task learning, missing information for further improvement might be derived from other tasks.

## References


Andrei Alexandrescu and Katrin Kirchhoff. 2009. Graph-based learning for statistical machine translation. In *Proceedings of HLT-NAACL*, pages 119–127.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The berkeley framenet project. In *Proceedings of ACL-COLING*, pages 86–90.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. 2010. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of ACL*, page 120.

Lorenzo Bruzzone, Mingmin Chi, and Mattia Marconcini. 2006. A novel transductive svm for semisupervised classification of remote-sensing images. *IEEE Transactions on Geoscience and Remote Sensing*, 44(11):3363–3373.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the conll-2005 shared task: Semantic role labeling. In *Proceedings of CoNLL*, pages 152–164.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Yaodong Chen, Ting Wang, Huowang Chen, and Xishan Xu. 2008. Semantic role labeling of chinese using transductive svm and semantic heuristics. In *Proceedings of IJCNLP*, pages 919–924.

Kevin Clark, Minh-Thang Luong, Christopher D Manning, and Quoc Le. 2018. Semi-supervised sequence modeling with cross-view training. In *Proceedings of EMNLP*, pages 1914–1925.

Dipanjan Das, Desai Chen, André FT Martins, Nathan Schneider, and Noah A Smith. 2014. Frame-semantic parsing. *Computational linguistics*, 40(1):9–56.

Hal Daume III. 2007. Frustratingly easy domain adaptation. In *Proceedings of ACL*, pages 256–263.

Michael Denkowski and Graham Neubig. 2017. Stronger baselines for trustable results in neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 18–27.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Kevin Duh and Katrin Kirchhoff. 2006. Lexicon acquisition for dialectal arabic using transductive learning. In *Proceedings of EMNLP*, pages 399–407.

Alexander Gammerman, Volodya Vovk, and Vladimir Vapnik. 1998. Learning by transduction. In *Proceedings of Uncertainty in artificial intelligence*, pages 148–155.

Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *Proceedings of ICML*, pages 1180–1189.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational linguistics*, 28(3):245–288.

Jiang Guo, Darsh Shah, and Regina Barzilay. 2018. Multi-source domain adaptation with mixture of experts. In *Proceedings of EMNLP*, pages 4694–4703.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of CoNLL*, pages 1–18.

Kazuma Hashimoto, Yoshimasa Tsuruoka, Richard Socher, et al. 2017. A joint many-task model: Growing a neural network for multiple nlp tasks. In *Proceedings of EMNLP*, pages 1923–1933.

Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2018. Jointly predicting predicates and arguments in neural semantic role labeling. In *Proceedings of ACL*, pages 364–369.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of ACL*, pages 328–339.

Georgiana Ifrim and Gerhard Weikum. 2006. Transductive learning for text classification using explicit knowledge models. In *Proceedings of European Conference on Principles of Data Mining and Knowledge Discovery*, pages 223–234.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in nlp. In *Proceedings of ACL*, pages 264–271.

Thorsten Joachims. 1999. Transductive inference for text classification using support vector machines. In *Proceedings of ICML*, pages 200–209.

Vidur Joshi, Matthew Peters, and Mark Hopkins. 2018. Extending a parser to distant domains using a few dozen partially annotated examples. In *Proceedings of ACL*, pages 1190–1199.

Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. 2016. Frustratingly easy neural domain adaptation. In *Proceedings of COLING*, pages 387–396.

Zuchao Li, Shexia He, Hai Zhano, Yiqing Zhang, Zhuosheng Zhang, Zhou. Xi, and Xiang Zhou. 2019. Dependency or span, end-to-end uniform semantic role labeling. In *Proceedings of AAAI*.

Yanbin Liu, Juho Lee, Minseop Park, Saehoon Kim, and Yi Yang. 2019. Transductive propagation network for few-shot learning. In *Proceedings of ICLR*.

Hiroki Ouchi, Hiroyuki Shindo, and Yuji Matsumoto. 2018. A span selection model for semantic role labeling. In *Proceedings of EMNLP*, pages 1630–1642.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106.

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of ACL*, pages 1756–1765.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL*, pages 2227–2237.

Elias Ponvert, Jason Baldridge, and Katrin Erk. 2011. Simple unsupervised grammar induction from raw text with cascaded finite state models. In *Proceedings of ACL*, pages 1077–1086.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Proceedings of CoNLL*, pages 1–40.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Lance A Ramshaw and Mitchell P Marcus. 1999. Text chunking using transformation-based learning. In *Natural language processing using very large corpora*, pages 157–176. Springer.

Sebastian Ruder and Barbara Plank. 2018. Strong baselines for neural semi-supervised learning under domain shift. In *Proceedings of ACL*, pages 1044–1054.

Kuniaki Saito, Yoshitaka Ushiku, and Tatsuya Harada. 2017. Asymmetric tri-training for unsupervised domain adaptation. In *Proceedings of ICML*, pages 2988–2997.

Erik F Sang and Sabine Buchholz. 2000. Introduction to the conll-2000 shared task: Chunking.

Ozan Sener, Hyun Oh Song, Ashutosh Saxena, and Silvio Savarese. 2016. Learning transferrable representations for unsupervised domain adaptation. In *Proceedings of NIPS*, pages 2110–2118.

Yuan Shi and Fei Sha. 2012. Information-theoretical learning of discriminative clusters for unsupervised domain adaptation. In *Proceedings of ICML*, pages 1275–1282.

Anders Søgaard. 2013. Semi-supervised learning and domain adaptation in natural language processing. *Synthesis Lectures on Human Language Technologies*, 6(2):1–103.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The conll 2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of CoNLL*, pages 159–177.

Jun Suzuki, Sho Takase, Hidetaka Kamigaito, Makoto Morishita, and Masaaki Nagata. 2018. An empirical study of building a strong baseline for constituency parsing. In *Proceedings of ACL*, pages 612–618.

Yuta Tsuboi, Hisashi Kashima, Shohei Hido, Steffen Bickel, and Masashi Sugiyama. 2009. Direct density ratio estimation for large-scale covariate shift adaptation. *Journal of Information Processing*, 17:138–155.

Nicola Ueffing, Gholamreza Haffari, and Anoop Sarkar. 2007. Transductive learning for statistical machine translation. In *Proceedings of ACL*, pages 25–32.

Vladimir Vapnik. 1998. *Statistical learning theory*. Wiley.

Rui Wang, Masao Utiyama, Lemao Liu, Kehai Chen, and Eiichiro Sumita. 2017. Instance weighting for neural machine translation domain adaptation. In *Proceedings of EMNLP*, pages 1482–1488.

Yufei Wang, Mark Johnson, Stephen Wan, Yifang Sun, and Wei Wang. 2019. How to best use syntax in semantic role labelling. In *Proceedings of ACL*, pages 5338–5343.

Min Xiao and Yuhong Guo. 2013. Domain adaptation for sequence labeling tasks with a probabilistic language adaptation model. In *Proceedings of ICML*, pages 293–301.

Yftah Ziser and Roi Reichart. 2017. Neural structural correspondence learning for domain adaptation. In *Proceedings of CoNLL*, pages 400–410.

# Efficient Sentence Embedding using Discrete Cosine Transform

**Nada Almarwani,**[*1,2] **Hanan Aldarmaki,**[*3] **Mona Diab** [1,4]
[1] Dept. of Computer Science, The George Washington University
[2] Dept. of Computer Science, Taibah University
[3] Computer Science & Software Engineering Dept., UAEU
[4] AWS, Amazon AI
nadaoh@gwu.edu, h-aldarmaki@uaeu.ac.ae, diabmona@amazon.com

## Abstract

Vector averaging remains one of the most popular sentence embedding methods in spite of its obvious disregard for syntactic structure. While more complex sequential or convolutional networks potentially yield superior classification performance, the improvements in classification accuracy are typically mediocre compared to the simple vector averaging. As an efficient alternative, we propose the use of discrete cosine transform (DCT) to compress word sequences in an order-preserving manner. The lower order DCT coefficients represent the overall feature patterns in sentences, which results in suitable embeddings for tasks that could benefit from syntactic features. Our results in semantic probing tasks demonstrate that DCT embeddings indeed preserve more syntactic information compared with vector averaging. With practically equivalent complexity, the model yields better overall performance in downstream classification tasks that correlate with syntactic features, which illustrates the capacity of DCT to preserve word order information.

## 1 Introduction

Modern NLP systems rely on word embeddings as input units to encode the statistical semantic and syntactic properties of words, ranging from standard context-independent embeddings such as word2vec (Mikolov et al., 2013) and Glove (Pennington et al., 2014) to contextualized embeddings such as ELMo (Peters et al., 2018) and BERT (Devlin et al., 2018). However, most applications operate at the phrase or sentence level. Hence, the word embeddings are averaged to yield sentence embeddings. Averaging is an efficient compositional operation that leads to good performance. In fact, averaging is difficult to beat by more complex compositional models as illustrated across several classification tasks: topic categorization, semantic textual similarity, and sentiment classification (Aldarmaki and Diab, 2018). Encoding sentences into fixed-length vectors that capture various full sentence linguistic properties leading to performance gains across different classification tasks remains a challenge. Given the complexity of most models that attempt to encode sentence structure, such as convolutional, recursive, or recurrent networks, the trade-off between efficiency and performance tips the balance in favor of simpler models like vector averaging. Sequential neural sentence encoders, like Skip-thought (Kiros et al., 2015) and InferSent (Conneau et al., 2017), can potentially encode rich semantic and syntactic features from sentence structures. However, for practical applications, sequential models are rather cumbersome and inefficient, and the gains in performance are typically mediocre compared with vector averaging (Aldarmaki and Diab, 2018). In addition, the more complex models typically don't generalize well to out-of-domain data (Wieting et al., 2015). FastSent (Hill et al., 2016) is an unsupervised alternative approach of lower computational cost, but similar to vector averaging, it disregards word order. Tensor-based composition can effectively capture word order, but current approaches rely on restricted grammatical constructs, such as transitive phrases, and cannot be easily extended to variable-length sequences of arbitrary structures (Milajevs et al., 2014). Therefore, despite its obvious disregard for structural properties, the efficiency and reasonable performance of vector averaging makes them more suitable for practical text classification.

In this work, we propose to use the Discrete Cosine Transform (DCT) as a simple and efficient way to model word order and structure in sentences while maintaining practical efficiency.

[*] Both authors contributed equally.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3672–3678, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

DCT is a widely-used technique in digital signal processing applications such as image compression (Watson, 1994) as well as speech recognition (Huang and Zhao, 2000) , but to our knowledge, this is the first successful application of DCT for NLP applications, and in particular sentence embedding. We use DCT to summarize the general feature patterns in word sequences and compress them into fixed-length vectors. Experiments in probing tasks demonstrate that our DCT embeddings preserve more syntactic and semantic features compared with vector averaging. Furthermore, the results indicate that DCT performance in downstream applications is correlated with these features.

## 2 Approach

### 2.1 Discrete Cosine Transform

Discrete Cosine Transform (DCT) is an invertible function that maps an input sequence of $N$ real numbers to the coefficients of $N$ orthogonal cosine basis functions. Given a vector of real numbers $\vec{v} = v_0, ..., v_{N-1}$, we calculate a sequence of DCT coefficients $c^0, ..., c^{N-1}$ as follows:[1]

$$c^0 = \sqrt{\frac{1}{N}} \sum_{n=0}^{N-1} v_n, \quad (1)$$

and

$$c^k = \sqrt{\frac{2}{N}} \sum_{n=0}^{N-1} v_n \cos \frac{\pi}{N}(n + \frac{1}{2})k, \quad (2)$$

for $1 \leq k < N$. Note that $c^0$ is the sum of the input sequence normalized by the square length, which is proportional to the average of the sequence. The $N$ coefficients can be used to reconstruct the original sequence exactly using the inverse transform. In practice, DCT is used for compression by preserving only the coefficients with large magnitudes. Lower-order coefficients represent lower signal frequencies which correspond to the overall patterns in the sequence (Ahmed et al., 1974).

### 2.2 DCT Sentence Embeddings

We apply DCT on the word vectors along the length of the sentence. Given a sentence of $N$ words $w_1, ..., w_N$, we stack the sequence of $d$-dimensional word embeddings in an $N \times d$ matrix, then apply DCT along the rows. In other words, each feature in the vector space is compressed independently, and the resultant DCT embeddings summarize the feature patterns along the word sequence. To get a fixed-length and consistent sentence vector, we extract and concatenate the first $K$ DCT coefficients and discard higher-order coefficients, which results in sentence vectors of size $Kd$. For cases where $N < K$, we pad the sentence with $K - N$ zero vectors. In image compression, the magnitude of the coefficients tends to decrease with increasing $k$, but we didn't observe this trend in text data except that $c^0$ tends to have larger absolute value than the remaining coefficients. Nonetheless, by retaining lower-order coefficients we get a consistent representation of overall feature patterns in the word sequence.

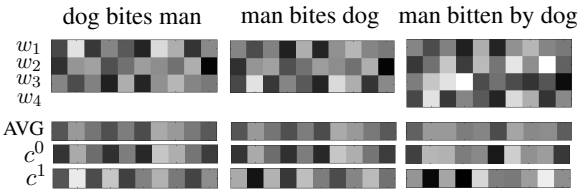


Figure 1: Illustration of word vector averaging vs. DCT using the first 2 DCT coefficients. The word vectors are generated randomly from a standard normal distribution with $d = 10$.

Figure 1 illustrates the properties of DCT embeddings compared to vector averaging (AVG). Notice that the first DCT coefficients, $c^0$, result in vectors that are independent of word order since the lowest frequency represents the average energy in the sequence. In this sense, $c^0$ is similar to AVG, where "dog bites man" and "man bites dog" have identical embeddings. The second-order coefficients, on the other hand, are sensitive to word order, which results in different representations for the above sentence pair. The counterexample "man bitten by dog" shows that $c^1$ embeddings are most sensitive to the overall patterns—in this case: "man ... dog"—which results in an embedding more similar to "man bites dog", than the semantically similar "dog bites man". However, there are still some variations in the final embeddings from the different word components ('bitten' vs. 'bite'), which can potentially be useful in downstream tasks. Since both DCT and AVG are un-

[1]There are several variants of DCT. We use DCT type II (Shao and Johnson, 2008) in our implementation

parameterized, the downstream classifiers can incorporate a hidden layer to learn these subtle variations in higher-order features depending on the learning objective.

### 2.3 A Note on Complexity

The cosine terms in Equation 2 can be precalculated for efficiency. For a maximum sentence length $\hat{N}$ and a given $K$, the total number of terms is $(K-1)\hat{N}$ for each feature. The run-time complexity is equivalent to calculating $K$ weighted averages, which is proportional to $KN$, where $K$ should be set to a small constant relative to the expected length.[2] Note also that the number of input parameters in downstream classification models will increase linearly with $K$. With parallel implementations however, the difference in run-time complexity between AVG and DCT is practically negligible.

## 3 Experiments and Results

### 3.1 Evaluation Framework

We use the SentEval toolkit[3] (Conneau and Kiela, 2018) to evaluate the sentence representations on different probing as well as downstream classification tasks. The probing benchmark set was designed to analyze the quality of sentence embeddings. It contains a set of 10 classification tasks, summarized in Table 1, that address varieties of linguistic properties including surface, syntactic, and semantic information (Conneau et al., 2018). The downstream set, on the other hand, includes the following standard classification tasks: binary and fine-grained sentiment classification (MR, SST2, SST5) (Pang and Lee, 2004; Socher et al., 2013), product reviews (CR) (Hu and Liu, 2004), opinion polarity (MPQA) (Wiebe et al., 2005), question type classification (TREC) (Voorhees and Tice, 2000), natural language inference (SICK-E) (Marelli et al., 2014), semantic relatedness (SICK-R, STSB) (Marelli et al., 2014; Cer et al., 2017), paraphrase detection (MRPC) (Dolan et al., 2004), and subjectivity/objectivity (SUBJ) (Pang and Lee, 2004).

### 3.2 Experimental setup

For the word embeddings, we use pre-trained FastText embeddings of size 300 (Mikolov et al., 2018) trained on Common-Crawl. We generate

[2]We experimented with $1 \leq K \leq 7$.

[3]https://github.com/facebookresearch/SentEval

| Task | Description |
|---|---|
| SentLen | Length prediction |
| WC | Word Content analysis |
| TreeDepth | Tree depth prediction |
| TopConst | Top Constituents prediction |
| BShift | Word order analysis |
| Tense | Verb tense prediction |
| SubjNum | Subject number prediction |
| ObjNum | Object number prediction |
| SOMO | Semantic odd man out |
| CoordInv | Coordination Inversion |

Table 1: Probing Tasks

DCT sentence vectors by concatenating the first $K$ DCT coefficients, which we denote by $\mathbf{c^{0:K}}$. We compare the performance against: vector averaging of the same word embeddings, denoted by **AVG**, and vector max pooling, denoted by **MAX**.[4]

For all tasks, we trained multi-layer perceptron (MLP) classifiers following the setup in SentEval. We tuned the following hyper-parameters on the validation sets: number of hidden states (in [0, 50, 100, 200, 512]) and dropout rate (in [0, 0.1, 0.2]). Note that the case with 0 hidden states corresponds to a Logistic Regression classifier.

### 3.3 Result & Discussion

We report the performance in probing tasks in Table 2. In general, DCT yields better performance compared to averaging on all tasks, and larger $K$ often yields improved performance in syntactic and semantic tasks. For the surface information tasks, SentLen and Word content (WC), $c^0$ significantly outperforms AVG. This is attributed to the non-linear scaling factor in DCT, where longer sentences are not discounted as much as in averaging. The performance decreased with increasing $K$ in $c^{0:K}$, which reflects the trade-off between deep and surface linguistic properties, as discussed in Conneau et al. (2018).

While increasing $K$ has no positive effect on surface information tasks, syntactic and semantic tasks demonstrate performance gains with larger $K$. This trend is clearly observed in all syntactic tasks and three of the semantic tasks, where DCT performs well above AVG and the performance improves with increasing $K$. The only exception

[4]To compare with other sentence embedding models, refer to the results in Conneau and Kiela (2018) and Conneau et al. (2018)

| | Surface | | Syntactic | | | Semantic | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Model | SentLen | WC | TreeDepth | TopConst | BShift | Tense | SubjNum | ObjNum | SOMO | CoordInv |
| Majority | 20.0 | 0.5 | 17.9 | 5.0 | 50.0 | 50.0 | 50.0 | 50.0 | 50.0 | 50.0 |
| Human | 100 | 100 | 84.0 | 84.0 | 98.0 | 85.0 | 88.0 | 86.5 | 81.2 | 85.0 |
| Length | 100 | 0.2 | 18.1 | 9.3 | 50.6 | 56.5 | 50.3 | 50.1 | 50.2 | 50.0 |
| AVG | 64.12 | 82.1 | 36.38 | 68.04 | 50.16 | 87.9 | 80.89 | 80.24 | 50.39 | 51.95 |
| MAX | 62.67 | 88.97 | 33.02 | 62.63 | 50.31 | 85.66 | 77.11 | 76.04 | 51.86 | 52.33 |
| $c^0$ | **98.67** | **91.11** | 38.6 | 70.54 | 50.42 | 88.25 | 80.88 | 80.56 | **55.6** | 55 |
| $c^{0:1}$ | 97.18 | 89.16 | 40.41 | 78.34 | 52.25 | 88.58 | 86.59 | 84.36 | 54.62 | 70.42 |
| $c^{0:2}$ | 95.84 | 86.77 | 43.01 | 80.41 | 54.84 | 88.87 | 88.06 | 86.26 | 53.07 | 71.87 |
| $c^{0:3}$ | 94.63 | 84.96 | **43.35** | 81.01 | 57.29 | 88.88 | 88.36 | 86.51 | 53.79 | **72.01** |
| $c^{0:4}$ | 93.25 | 83.24 | 43.26 | 81.49 | 60.31 | **88.91** | **88.65** | 87.15 | 52.77 | 71.91 |
| $c^{0:5}$ | 92.29 | 81.84 | 42.75 | **81.60** | 62.01 | 88.82 | 88.44 | 87.98 | 52.38 | 70.96 |
| $c^{0:6}$ | 91.56 | 79.83 | 43.05 | 81.41 | **62.59** | 88.87 | **88.65** | **88.28** | 52.07 | 70.63 |

Table 2: Probing tasks performance of vector averaging (AVG) and max pooling (MAX) vs. DCT embeddings with various $K$. Majority (baseline), Human (human-bound), and a linear classifier with sentence length as sole feature (Length) as reported in (Conneau et al., 2018), respectively.

| | Sentiment Analysis | | | | | SUBJ | Relatedness/Paraphrase | | | Inference | TREC |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Model | MR | SST2 | SST5 | CR | MPQA | | SICK-R | STSB | MRPC | SICK-E | |
| AVG | 78.3 | **84.13** | 44.16 | 79.6 | 87.94 | 92.33 | 81.95 | 69.26 | 74.43 | 79.5 | 83.2 |
| MAX | 73.31 | 79.24 | 41.86 | 73.35 | 86.54 | 88.02 | 81.93 | **71.57** | 72.5 | 77.98 | 76.2 |
| $c^0$ | **78.45** | 83.53 | 44.57 | 79.81 | **88.36** | **92.79** | 82.61 | 71.11 | 72.93 | 78.91 | 84.8 |
| $c^{0:1}$ | 78.15 | 83.47 | **46.06** | 79.84 | 87.76 | 92.61 | 82.73 | 70.82 | 72.81 | 79.64 | 88.2 |
| $c^{0:2}$ | 78.02 | 82.98 | 45.16 | 79.68 | 87.62 | 92.5 | **82.95** | 70.36 | 72.87 | 79.76 | **89.8** |
| $c^{0:3}$ | 77.81 | 83.8 | 45.79 | 79.66 | 87.54 | 92.4 | 82.93 | 69.79 | 73.57 | **80.56** | 88.2 |
| $c^{0:4}$ | 77.72 | 83.75 | 44.03 | **80.08** | 87.4 | 92.61 | 82.53 | 69.31 | 72.35 | 79.72 | **89.8** |
| $c^{0:5}$ | 77.42 | 82.43 | 43.3 | 78.6 | 87.21 | 92.19 | 82.36 | 68.9 | 73.91 | 79.89 | 88.8 |
| $c^{0:6}$ | 77.47 | 82.81 | 42.99 | 78.78 | 87.06 | 92.15 | 81.86 | 68.17 | **75.07** | 79.76 | 86.4 |

Table 3: DCT embedding Performance in SentEval downstream tasks compared to vector averaging (AVG) and max pooling (MAX).

is SOMO, where increasing $K$ actually results in lower performance, although all DCT results are still higher than AVG by about 1% to 34%.

The correlation between the performance in probing tasks and the standard text classification tasks is discussed in Conneau et al. (2018), where they show that most tasks are only positively correlated with a small subset of semantic or syntactic features, with the exception of TREC and some sentiment classification benchmarks. Furthermore, some tasks like SST and SICK-R are actually negatively correlated with the performance in some probing tasks like SubjNum, ObjNum, and BShift. This explains why simple averaging often outperforms more complex models in these tasks. Our results in Table 3 are consistent with these observations, where we see improvements in most tasks, but the difference is not as significant as the probing tasks, except in TREC question classification where increasing $K$ leads to much better performance. As discussed in Aldarmaki and Diab (2018), the ability to preserve word order leads to improved performance in TREC, which is exactly the advantage of using DCT instead of AVG. Note also that increasing $K$, while preserves more information, leads to increasing the number of model parameters, which in turn may negatively affect the generalization of the model by overfitting. In our experiments, $1 \leq k \leq 2$ yielded the best trade-off.

### 3.4 Comparison w. Related Methods

Spectral analysis is frequently employed in signal processing to decompose a signal into separate

| | 20-NG | | | R-8 | | | SST-5 | | |
|---|---|---|---|---|---|---|---|---|---|
| Model | P | R | F1 | P | R | F1 | P | R | F1 |
| PCA | 55.43 | 54.67 | 54.77 | 83.83 | 83.42 | 83.41 | 26.47 | 25.08 | 25.23 |
| DCT* | 61.07 | 59.16 | 59.78 | 90.41 | 90.78 | 90.38 | 30.11 | 30.09 | 29.53 |
| Avg. vec. | 68.72 | 68.19 | 68.25 | 96.34 | 96.30 | 96.27 | 27.88 | 26.44 | 24.81 |
| p-means | *72.20* | **71.65** | **71.79** | 96.69 | 96.67 | 96.65 | 33.77 | 33.41 | 33.26 |
| ELMo | 71.20 | 71.79 | 71.36 | 94.54 | 91.32 | 91.32 | *42.35* | *41.51* | *41.54* |
| BERT | 70.89 | 70.79 | 70.88 | 95.52 | 95.39 | 95.39 | 39.92 | 39.38 | 39.35 |
| EigenSent | 66.98 | 66.40 | 66.54 | 95.91 | 95.80 | 95.76 | 35.32 | 33.69 | 33.91 |
| EigenSent⊕Avg | **72.24** | *71.62* | *71.78* | **97.18** | **97.13** | **97.14** | **42.77** | **41.67** | **41.81** |
| $c^k$ | *72.20* | 71.58 | 71.73 | *96.98* | *96.98* | *96.94* | 37.67 | 34.47 | 34.54 |

Table 4: Performance in text classification (20-NG, R-8) and sentiment (SST-5) tasks of various models as reported in (Kayal and Tsatsaronis, 2019), where DCT* refers to the implementation in (Kayal and Tsatsaronis, 2019). Our DCT embeddings are denoted as $c^k$ in the bottom row. **Bold** indicates the best result, and *italic* indicates second-best.

frequency components, each revealing some information about the source signal, to enable analysis and compression. To the best of our knowledge, spectral methods have only been recently exploited to construct sentence embedding (Kayal and Tsatsaronis, 2019).[5].

Kayal and Tsatsaronis propose EigenSent that utilized Higher-Order Dynamic Mode Decomposition (HODMD) (Le Clainche and Vega, 2017) to construct sentence embedding. These embeddings summarize the dynamic properties of the sentence. In their work, they compare EigenSent with various sentence embedding models, including a different implementation of the Discrete Cosine Transform (DCT*). In contrast to our implementation described in section 2.2, DCT* is applied at the word level along the word embedding dimension.

For fair comparison, we use the same sentiment and text classification datasets, the SST-5, 20 newsgroups (20-NG) and Reuters-8 (R-8), as those used in Kayal and Tsatsaronis (2019). We also evaluate using the same pre-trained word embedding, framework and approaches as described in their work. Table 4 shows the best results for the various models as reported in Kayal and Tsatsaronis (2019), in addition to the best performance of our model denoted as $c^k$.[6]

Note that the DCT-based model, DCT*, described in Kayal and Tsatsaronis (2019) performed relatively poorly in all tasks, while our model achieved close to state-of-the-art performance in both the 20-NG and R-8 tasks. Our model outperformed EignSent on all tasks and generally performed better than or on par with p-means, ELMo, BERT, and EigenSent⊕Avg on both the 20-NG and R-8. On the other hand, both EigenSent⊕Avg and ELMo performed better than all other models on SST-5.

## 4 Conclusion

We proposed using the Discrete Cosine Transform (DCT) as a mechanism to efficiently compress variable-length sentences into fixed-length vectors in a manner that preserves some of the structural characteristics of the original sentences. By applying DCT on each feature along the word embedding sequence, we efficiently encode the overall feature patterns as reflected in the low-order DCT coefficients. We showed that these DCT embeddings reflect average semantic features, as in vector averaging but with a more suitable normalization, in addition to syntactic features like word order. Experiments using the SentEval suite showed that DCT embeddings outperform the commonly-used vector averaging on most tasks, particularly tasks that correlate with sentence structure and word order. Without compromising practical efficiency relative to averaging, DCT provides a suitable mechanism to represent both the average of the features and their overall syntactic patterns.

[5] independent from and in parallel with this work

[6] The best results were achieved with k=3 for SST-5 and k=2 for 20-NG and R-8.

## References


Nasir Ahmed, T_ Natarajan, and Kamisetty R Rao. 1974. Discrete cosine transform. *IEEE transactions on Computers*, 100(1):90–93.

Hanan Aldarmaki and Mona Diab. 2018. Evaluation of unsupervised compositional representations. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2666–2677.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14.

Alexis Conneau and Douwe Kiela. 2018. Senteval: An evaluation toolkit for universal sentence representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

Alexis Conneau, Germán Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $ &!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th international conference on Computational Linguistics*, page 350. Association for Computational Linguistics.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *Proceedings of NAACL-HLT*, pages 1367–1377.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Jun Huang and Yunxin Zhao. 2000. A dct-based fast signal subspace technique for robust speech recognition. *IEEE Transactions on Speech and Audio Processing*, 8(6):747–751.

Subhradeep Kayal and George Tsatsaronis. 2019. Eigensent: Spectral sentence embeddings using higher-order dynamic mode decomposition. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 4536–4546.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Soledad Le Clainche and José M Vega. 2017. Higher order dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems*, 16(2):882–925.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, pages 1–8.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the 11th Language Resources and Evaluation Conference*, Miyazaki, Japan. European Language Resource Association.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Dmitrijs Milajevs, Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Matthew Purver. 2014. Evaluating neural word representations in tensor-based compositional settings. *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke

Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237.

Xuancheng Shao and Steven G Johnson. 2008. Type-ii/iii dct/dst algorithms with reduced number of arithmetic operations. *Signal Processing*, 88(6):1553–1564.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Ellen M Voorhees and Dawn M Tice. 2000. Building a question answering test collection. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 200–207. ACM.

Andrew B Watson. 1994. Image compression using the discrete cosine transform. *Mathematica journal*, 4(1):81.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39(2-3):165–210.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Towards universal paraphrastic sentence embeddings. *CoRR*, abs/1511.08198.

# A Search-based Neural Model for Biomedical Nested and Overlapping Event Detection

**Kurt Espinosa**[1,4], **Makoto Miwa**[2,3], **Sophia Ananiadou**[1]

[1]National Centre for Text Mining, School of Computer Science, The University of Manchester, UK
[2]Toyota Technological Institute, Nagoya, 468-8511, Japan
[3]Artificial Intelligence Research Center (AIRC),
National Institute of Advanced Industrial Science and Technology (AIST), Japan
[4]Department of Computer Science, University of the Philippines Cebu, Cebu City, Philippines
{kurtjunshean.espinosa, sophia.ananiadou}@manchester.ac.uk
makoto-miwa@toyota-ti.ac.jp

## Abstract

We tackle the nested and overlapping event detection task and propose a novel search-based neural network (SBNN) structured prediction model that treats the task as a search problem on a relation graph of trigger-argument structures. Unlike existing structured prediction tasks such as dependency parsing, the task targets to detect DAG structures, which constitute events, from the relation graph. We define actions to construct events and use all the beams in a beam search to detect all event structures that may be overlapping and nested. The search process constructs events in a bottom-up manner while modelling the global properties for nested and overlapping structures simultaneously using neural networks. We show that the model achieves performance comparable to the state-of-the-art model Turku Event Extraction System (TEES) on the BioNLP Cancer Genetics (CG) Shared Task 2013 without the use of any syntactic and hand-engineered features. Further analyses on the development set show that our model is more computationally efficient while yielding higher F1-score performance.

## 1 Introduction

Nested and overlapping event structures, which occur widely in text, are important because they can capture relations between events such as causality, e.g., a "production" event is a consequence of a "discovery" event, which in turn is a result of an "exploration" event. Event extraction involves the identification of a trigger and a set of its arguments in a given text. Figure 1 shows an example of a nested and overlapping event structure in the biomedical domain. The relation graph (topmost) forms a directed acyclic graph (DAG) structure (McClosky et al., 2011) and it encapsulates 15 event structures. It contains nested event

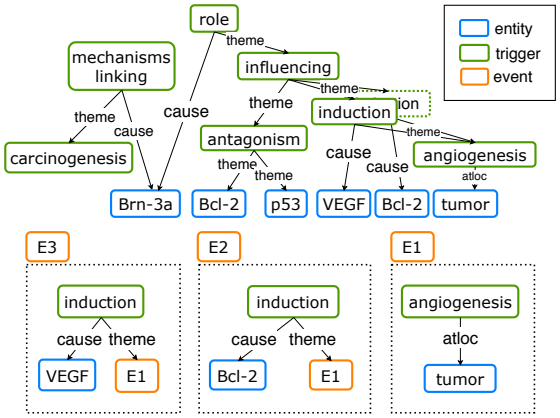


Figure 1: Top: A DAG-structured relation graph (topmost) from the sentence "*Looking for mechanisms linking Brn-3a to carcinogenesis, we discuss the role of this transcription factor in influencing Bcl-2/VEGF induction of tumor angiogenesis, ...*" from BioNLP'13 CG Shared Task (Pyysalo et al., 2015). Bottom: A pair of overlapping and nested events $(E2, E3)$ extracted from the graph with their shared argument event, a flat event $(E1)$.

structures such as *E2,E3* because one of their arguments, in this case *E1*, is an event. Specifically, *E1* is a flat event since its argument is an entity. Moreover, *E2* and *E3* are also overlapping events (explicitly shown in the relation graph having two *induction* triggers) because they share a common argument, *E1*.

State-of-the-art approaches to event extraction in the biomedical domain are pipeline systems (Björne and Salakoski, 2018; Miwa et al., 2013) that decompose event extraction into simpler tasks such as: i) *trigger/entity detection*, which determines which words and phrases in a sentence potentially constitute as participants of an event, ii) *relation detection*, which finds pairwise relations between triggers and arguments, and iii) *event detection*, which combines pairwise



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3679–3686,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

relations into complete event structures. Joint approaches have also been explored (Rao et al., 2017; Riedel and McCallum, 2011; Vlachos and Craven, 2012; Venugopal et al., 2014), but they focus on finding relation graphs and detect events with rules. McClosky et al. (2011) treats events as dependency structures by constraining event structures to map to trees, thus their method cannot represent overlapping event structures. Other neural models in event extraction are in the general domain (Feng et al., 2016; Nguyen and Grishman, 2015; Chen et al., 2015; Nguyen et al., 2016), but they used the ACE2005 corpus which does not have nested events (Miwa et al., 2014). Furthermore, there are some efforts on applying transition-based methods on DAG structures in dependency parsing, e.g., (Sagae and Tsujii, 2008; Wang et al., 2018), however, they do not consider overlapping and nested structures.

We present a novel search-based neural *event detection* model that detects overlapping and nested events with beam search by formulating it as a structured prediction task for DAG structures. Given a relation graph of trigger-argument relations, our model detects nested events by searching a sequence of actions that construct event structures incrementally in a bottom-up manner. We focus on *event detection* since the existing methods do not consider nested and overlapping structures as a whole during learning. Treating them simultaneously helps the model avoid inferring wrong causality relations between entities. Our model detects overlapping events by maintaining multiple beams and detecting events from all the beams, in contrast to existing transition-based methods (Nivre, 2003, 2006; Chen and Manning, 2014; Dyer et al., 2015; Andor et al., 2016; Vlachos and Craven, 2012). We define an LSTM-based neural network model that can represent nested event structure to choose actions.

We show that our event detection model achieves performance comparable to the event detection module of the state-of-the-art TEES system (Björne and Salakoski, 2018) on the BioNLP CG Shared Task 2013 (Pyysalo et al., 2013) without the use of any syntactic and hand-engineered features. Furthermore, analyses on the development set show our model performs fewer number of classifications in less time.

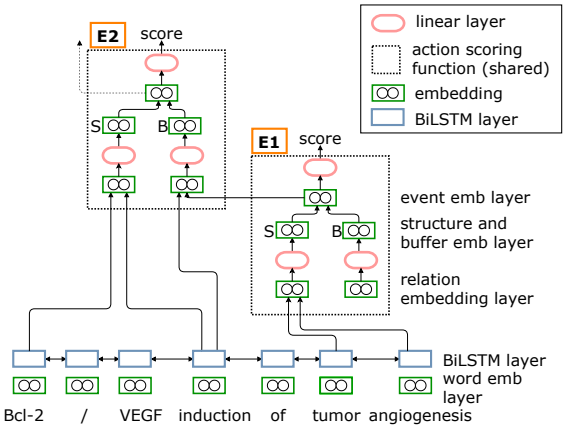


Figure 2: An illustration of the proposed neural model detecting event structures in a bottom-up manner, where E1 event representation becomes an argument to E2 event structure on the example sentence used in Figure 1.

## 2 Model

We describe our search-based neural network (SBNN) model that constitutes events from a relation graph by structured prediction. SBNN resembles an incremental transition-based parser (Nivre, 2006), but the search order, actions and representations are defined for DAG structures. We first discuss how we generate the relation graph in §2.1, then describe the structured prediction algorithm in §2.2 and the neural network in §2.3 and lastly, we explain the training procedure in §2.4.

### 2.1 Relation Graph Generation

To train our model, we use the predicted relations merged with pairwise relations decomposed from the gold events. We then generate a relation graph from the merged relations. During inference, the relation graph is generated only from the predicted relations. Figure 1 shows an example of the generated DAG-structured relation graph.

### 2.2 Structured Prediction for DAGs

We represent event structures with DAG structures and find them in a relation graph. Our model performs beam search on relation graphs by choosing actions to construct events. Unlike existing beam search usage where they only choose the best path in the beam, e.g., (Nivre, 2006), we use *all* the beams to predict event structures, which enables the model to predict overlapping and nested events.

Event structures are searched and fixed for each

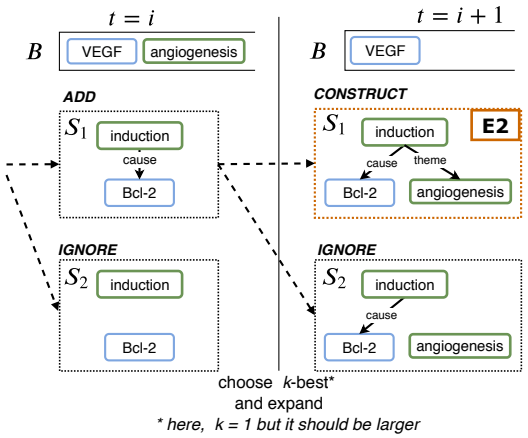


Figure 3: A snapshot of the search procedure within one time step (with $k = 1$) on the trigger *induction* which detects event $E2$ after a CONSTRUCT action.

trigger of the relation graph in a bottom-up manner. The model predicts the flat events first and then the representations of the flat events become the argument of the nested events. Figure 2 shows the proposed neural model (described in detail in §2.3), which illustrates that the event representation of event $E1$ becomes the argument to event $E2$. If the flat event is not detected, its nested event will not be detected consequently. When this happens, the search process stops.

Figure 3 shows an snapshot of the search procedure within one time step as applied to a relation graph with a trigger *induction* to detect event $E2$ (see Figure 1 and 2). To do the search, the model maintains two data structures: a buffer $B$ which holds a queue of arguments[1] to be processed and a structure $S$ which contains the partially built event structure. The initial state is composed of the buffer $B$ with all the arguments and a structure $S$ empty. At each succeeding time step, the model applies a set of predefined actions to each argument and uses a neural network to score those actions. Processing completes when $B$ is empty and $S$ contains all the arguments for the trigger along with the history of actions taken by the model.

We now define the three actions[2] that the model applies at each time step to each argument, namely: add the argument (ADD), ignore the argument (IGNORE) and add the argument and construct an event candidate (CONSTRUCT).

[1]A special NONE argument marks the first argument in the buffer to enable detection of no-argument events.

[2]Except for the NONE argument where only two actions are applied: IGNORE and CONSTRUCT.

We have chosen only three actions for simplicity. Figure 3 shows how actions are applied to the argument *angiogenesis* (due to space constraints we only show two actions indicated by the arrows). Concretely, we show two distinct structures $(S_1, S_2)$ that are created in time step $t + 1$ after *CONSTRUCT* and *IGNORE* actions were applied to the current structure $S_1$ in time step $i$.

The event candidate structures are fixed as events if the scores of the CONSTRUCT actions are above a certain threshold. The resulting state after a CONSTRUCT action is removed from the beams. We maintain multiple beams and use all of them to predict overlapping events.

### 2.3 Neural Network as Scoring Function

Figure 2 shows the proposed neural model. We employ a **BiLSTM** network to generate the word representations from pre-trained **word embeddings**. To represent phrases, we averaged the word representations. The LSTM network is shared among the states during search in the sentence. We build a relation embedding for each argument, which concatenates the information of the trigger $t$, the role $o$, the argument $a$ and the action $c$. We include both the type information $p$ and the word or phrase representation $w$ of the trigger or entity argument. Formally, each relation $r_i$ is represented as a **relation embedding**: $\mathbf{r}_i = [\mathbf{t}_p; \mathbf{t}_w; \mathbf{o}_p; \mathbf{a}_p; \mathbf{a}_w; \mathbf{c}]$, where $\mathbf{t}_p$ is the representation of the type of the trigger and so on. Each $\mathbf{r}_i$ is passed to a linear **hidden layer**, then to a rectified linear unit (ReLU) non-linearity and summed to produce the **structure** and **buffer** embeddings: $\mathbf{S}_t$ and $\mathbf{B}_t$.

We use a neural network as the **action scoring function** (indicated by the dotted box in Figure 2) defined as $\sigma(a_t|S_{t-1}, B_{t-1})$. We model the action scoring function using $S_t$ and $B_t$, which are composed by adding an action $a_t$ for a relation $r_t$ to $S_{t-1}$ and moving $r_t$ from $B_{t-1}$ to $S_{t-1}$. The state at any time step $t$ is composed of the buffer $B_t$ and the partially built structure $S_t$. Each of $S_t$ and $B_t$ contains a set of relations $\{r_1, r_2, r_3, \ldots, r_n\}$. In Figure 2, there is no arrow to *B* in event *E1* because the diagram only shows the model snapshot at a specific time step during the search process. In this particular time step *t*, the buffer *B* in *E1* is already empty and thus, it does not contain any relations $\mathbf{r}_i$.

$\mathbf{S}_t$ and $\mathbf{B}_t$ are then concatenated to form the

**event embedding**. The event embedding has the same dimension as the sum of argument type and word dimensions so that it can be used as argument representation in nested events as shown in Figure 2. Then, we passed the event embedding into a linear **hidden layer** and output $z_t$. Finally, the scoring function $\sigma$ is calculated as $\sigma(a_t|S_{t-1}, B_{t-1}) = \text{sigmoid}(z_t)$.

### 2.4 Training

From the relation graph generated in §2.1, we calculate gold action sequences that construct the gold event structures on the graph. The loss is summed over all actions and for all the events during the beam search and thus the objective function is to minimise their negative log-likelihood. We employ early updates (Collins and Roark, 2004): if the gold falls out of the beam, we stop searching and update the model immediately.

## 3 Experimental Settings

We applied our model to the BioNLP CG shared task 2013 (Pyysalo et al., 2015). We used the original data partition and employed the official evaluation metrics. We focussed on the CG task dataset over other BioNLP datasets because of its complexity and size (Nédellec et al., 2013; Björne and Salakoski, 2018; Pyysalo et al., 2013). The CG dataset has the most number of entity types and event types and thus is the most complex among the available (and accessible) BioNLP datasets. Furthermore, the CG dataset is the largest dataset in terms of the number of event instances and the proportion of nested and overlapping events. Evaluating our model extensively on other BioNLP tasks and datasets is part of our future work.

The development set contains 3,217 events of which 36.46% are nested events, 43.05% are overlapping events and 44.07% are flat events. Note that the total does not equal to 100% because nested and overlapping events may have intersection: a nested event can be overlapping and vice versa.

We compared our model with the event detection module of the state-of-the-art model TEES (Björne and Salakoski, 2018), which employs convolutional neural network and uses syntactic and hand-engineered features for event detection. Björne and Salakoski (2018) found that the dependency parse features increased the performance of the convolutional model. In contrast, we do not use these syntactic features nor hand-engineered features. Furthermore, instead of the ensemble methods, we used TEES's published *single* models, as this enables us to make a direct comparison with TEES in a minimal setting. We train our model using the predicted relations from TEES merged with the pairwise relations decomposed from the gold CG events. During inference, we predict event structures using only the predicted relations from TEES.

### 3.1 Nested and Overlapping Event Evaluation Process

Similarly, we used the official evaluation script to measure the performance of the model on nested, overlapping and flat events. We first separated the nested, overlapping and flat events, respectively. Then we compute the precision and recall for each category in the following way. For example, for nested events, to compute the precision, we compare the predicted nested events with all gold events and to compute recall, we compare gold nested events with all predicted events. The evaluation script detects nested events by comparing the whole tree structure down to its sub-events until it reaches the flat events. Hence, the performance scores of the nested events inevitably include the performance on flat events.

### 3.2 Training Details and Model Parameters

We implemented our model using the Chainer library (Tokui et al., 2015). We initialised the word embeddings using pre-trained embeddings (Chiu et al., 2016) while other embeddings are initialised using the normal distribution. All the embeddings and weight parameters were updated with mini-batch using the AMSGrad optimiser (Reddi et al., 2018). We also incorporated early stopping to choose the number of training epochs and tuned hyper-parameters (dropout, learning rate and weight decay rate) using grid search. The model parameters can be found in appendix A.

## 4 Results and Analyses

Table 1 shows the event detection performance of the models on the test set. Our model achieves performance comparable to the state-of-the-art TEES event detection module without the use of any syntactic and hand-engineered features, suggesting it can be applied to other domains with no need for feature engineering. We validated it to have no sig-

| Model | P | R | F1 (%) |
|---|---|---|---|
| TEES | 61.42 | **52.93** | 56.86 |
| SBNN | **63.67** | 51.43 | **56.90** |

Table 1: Event detection performance on the CG task 2013 test dataset.

| Model | P | R | F (%) |
|---|---|---|---|
| TEES | 56.81 | 48.21 | 52.16 |
| SBNN $k = 1$ | 48.95 | 43.79 | 46.23 |
| $k = 8$ | **63.60** | 47.46 | **54.36** |
| $k = 64$ | 60.30 | **49.16** | 54.17 |
| $k = 256$ | 60.91 | 48.53 | 54.02 |

Table 2: Event detection performance on CG task 2013 development set for SBNN with the top three performing $k$-values and when $k = 1$.

nificant statistical difference with the TEES model (the Approximate Randomisation test (Yeh, 2000; Noreen, 1989)).

To gain a deeper insight about the model, we performed analyses on the development set. Table 2 shows the performance of SBNN by varying the $k$-best parameter in beam search. We tested $2^i$ values for $i = 1, 2, 3, \ldots, 11$ and found that the best value was 8 with F1-score of 54.36%, which is 2.2 percentage points (pp) higher than TEES.

Table 3 shows the number of classifications (or action scoring function calls in our model) performed by each model with the corresponding actual running time. SBNN performs fewer classifications and in less time than TEES, implying it is more computationally efficient.

Table 4 shows the performance comparison of the models on nested, overlapping and flat event detection. Our model yields higher F1-scores than TEES which can be attributed to its ability to maintain multiple beams and to detect events from all these beams during search.

Finally, we computed the upper bound recall given the predicted relations from TEES. The upper bound is computed by setting the threshold parameter of our model to zero, which then constructs all gold events possible from the predicted relations of TEES. Since we evaluate our model on the output relations of TEES, the event detection performance is bounded or limited by these predicted relations. For instance, if one of the relations in an event was not predicted, the event structure will never be formed. We observe that this remains a challenging task since the upper bound recall is still at 53.47% (6.01pp higher than our current model's score). Closing this gap requires among others addressing inter-sentence and self-referential events, which account for 3.1% of the total events.

| Model | Number of Classifications | Running Time (s) |
|---|---|---|
| TEES | 6,141 | 155 |
| SBNN $k = 8$ | **4,093** | **131** |

Table 3: Comparison on computational efficiency on the CG task 2013 development dataset.

| Model | Nested | Overlapping | Flat | Overall F1 (%) |
|---|---|---|---|---|
| TEES | 42.70 | 34.49 | 56.81 | 52.16 |
| SBNN $k = 8$ | **45.24** | **36.92** | **60.5** | **54.36** |

Table 4: Nested and overlapping event detection F1 (%) score performance on the CG task 2013 development set.

## 5 Conclusions and Future Work

We presented a novel search-based neural model for nested and overlapping event detection by treating the task as structured prediction for DAGs. Our model achieves performance comparable to the state-of-the-art TEES event detection model without the use of any syntactic and hand-engineered features, suggesting the domain-independence of the model. Further analyses on the development set revealed some desirable characteristics of the model such as its computational efficiency while yielding higher F1-score performance. These results set the first focussed benchmark of our model and next steps include applying it to other event datasets in the biomedical and general domain. In addition, it can also be applied to other DAG structures such as nested/discontiguous entities (Muis and Lu, 2016; Ju et al., 2018).

## Acknowledgement

We thank our anonymous reviewers for their invaluable feedback. This research was supported by funding from BBSRC Japan Partnering Award, Text mining and bioinformatics platforms for metabolic pathway modelling [Grant ID: BB/P025684/1] and AIRC/AIST. The first author gratefully acknowledges financial support from the University of the Philippines System Doctoral Studies Fund.

## References


Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2442–2452, Berlin, Germany. Association for Computational Linguistics.

Jari Björne and Tapio Salakoski. 2018. Biomedical event extraction using convolutional neural networks and dependency parsing. In *Proceedings of the BioNLP 2018 workshop*, pages 98–108, Melbourne, Australia. Association for Computational Linguistics.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar. Association for Computational Linguistics.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 167–176, Beijing, China. Association for Computational Linguistics.

Billy Chiu, Gamal Crichton, Anna Korhonen, and Sampo Pyysalo. 2016. How to train good word embeddings for biomedical nlp. In *Proceedings of the 15th Workshop on Biomedical Natural Language Processing*, pages 166–174, Berlin, Germany. Association for Computational Linguistics.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 111–118, Barcelona, Spain.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343, Beijing, China. Association for Computational Linguistics.

Xiaocheng Feng, Lifu Huang, Duyu Tang, Heng Ji, Bing Qin, and Ting Liu. 2016. A language-independent neural network for event detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 66–71, Berlin, Germany. Association for Computational Linguistics.

Meizhi Ju, Makoto Miwa, and Sophia Ananiadou. 2018. A neural layered model for nested named entity recognition. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1446–1459, New Orleans, Louisiana. Association for Computational Linguistics.

David McClosky, Mihai Surdeanu, and Christopher Manning. 2011. Event extraction as dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1626–1635, Portland, Oregon, USA. Association for Computational Linguistics.

Makoto Miwa, Sampo Pyysalo, Tomoko Ohta, and Sophia Ananiadou. 2013. Wide coverage biomedical event extraction using multiple partially overlapping corpora. *BMC bioinformatics*, 14(1):175.

Makoto Miwa, Paul Thompson, Ioannis Korkontzelos, and Sophia Ananiadou. 2014. Comparable study of event extraction in newswire and biomedical domains. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2270–2279, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Aldrian Obaja Muis and Wei Lu. 2016. Learning to recognize discontiguous entities. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 75–84, Austin, Texas. Association for Computational Linguistics.

Claire Nédellec, Robert Bossy, Jin-Dong Kim, Jung-Jae Kim, Tomoko Ohta, Sampo Pyysalo, and Pierre Zweigenbaum. 2013. Overview of bionlp shared task 2013. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 1–7.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 300–309, San Diego, California. Association for Computational Linguistics.

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 365–371, Beijing, China. Association for Computational Linguistics.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pages 149–160, Nancy, France.

Joakim Nivre. 2006. *Inductive dependency parsing*, 1 edition, volume 34 of *Text, Speech and Language Technology*. Springer Netherlands.

Eric W Noreen. 1989. *Computer-intensive methods for testing hypotheses*. Wiley New York.

Sampo Pyysalo, Tomoko Ohta, and Sophia Ananiadou. 2013. Overview of the cancer genetics (cg) task of bionlp shared task 2013. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 58–66.

Sampo Pyysalo, Tomoko Ohta, Rafal Rak, Andrew Rowley, Hong-Woo Chun, Sung-Jae Jung, Sung-Pil Choi, Jun'ichi Tsujii, and Sophia Ananiadou. 2015. Overview of the cancer genetics and pathway curation tasks of bionlp shared task 2013. *BMC bioinformatics*, 16(10):S2.

Sudha Rao, Daniel Marcu, Kevin Knight, and Hal Daumé III. 2017. Biomedical event extraction using abstract meaning representation. In *BioNLP 2017*, pages 126–135, Vancouver, Canada,. Association for Computational Linguistics.

Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. 2018. On the convergence of adam and beyond. In *International Conference on Learning Representations*.

Sebastian Riedel and Andrew McCallum. 2011. Robust biomedical event extraction with dual decomposition and minimal domain adaptation. In *Proceedings of BioNLP Shared Task 2011 Workshop*, pages 46–50, Portland, Oregon, USA. Association for Computational Linguistics.

Kenji Sagae and Jun'ichi Tsujii. 2008. Shift-reduce dependency dag parsing. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 753–760. Coling 2008 Organizing Committee.

Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. 2015. Chainer: a next-generation open source framework for deep learning. In *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS)*.

Deepak Venugopal, Chen Chen, Vibhav Gogate, and Vincent Ng. 2014. Relieving the computational bottleneck: Joint inference for event extraction with high-dimensional features. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 831–843, Doha, Qatar. Association for Computational Linguistics.

Andreas Vlachos and Mark Craven. 2012. Biomedical event extraction from abstracts and full papers using search-based structured prediction. *BMC bioinformatics*, 13(11):S5.

Yuxuan Wang, Wanxiang Che, Jiang Guo, and Ting Liu. 2018. A neural transition-based approach for semantic dependency graph parsing. In *AAAI Conference on Artificial Intelligence*.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th conference on Computational linguistics-Volume 2*, pages 947–953. Association for Computational Linguistics.

## A Model Parameters

| Parameter | Value/Dimension |
|---|---|
| Mini-Batch Size | 100 |
| Word Embedding | 200 |
| BiLSTM Word Embedding | 100 |
| Role Type Embedding | 10 |
| Trigger/Argument Type Embedding | 20 |
| Early Stopping Patience | 5 |
| Dropout | 0.5 |
| Learning Rate | 0.001 |
| Hidden Layer Size | 60 |
| Event Embedding | 100 |
| Action Score Threshold | 0.5 |
| Beam Size | 8 |
| Action Embedding | 4 |
| Weight Decay Rate | 0.001 |

Table 5: Hyper-parameters used in our experiments.

# PAWS-X: A Cross-lingual Adversarial Dataset for Paraphrase Identification

**Yinfei Yang**,* **Yuan Zhang**,* **Chris Tar** and **Jason Baldridge**
Google Research, Mountain View, 94043
{yinfeiy, zhangyua, ctar, jasonbaldridge}@google.com

## Abstract

Most existing work on adversarial data generation focuses on English. For example, PAWS (Paraphrase Adversaries from Word Scrambling) (Zhang et al., 2019) consists of challenging English paraphrase identification pairs from Wikipedia and Quora. We remedy this gap with PAWS-X, a new dataset of 23,659 *human* translated PAWS evaluation pairs in six typologically distinct languages: French, Spanish, German, Chinese, Japanese, and Korean. We provide baseline numbers for three models with different capacity to capture non-local context and sentence structure, and using different multilingual training and evaluation regimes. Multilingual BERT (Devlin et al., 2019) fine-tuned on PAWS English plus machine-translated data performs the best, with a range of 83.1-90.8 accuracy across the non-English languages and an average accuracy gain of 23% over the next best model. PAWS-X shows the effectiveness of deep, multilingual pre-training while also leaving considerable headroom as a new challenge to drive multilingual research that better captures structure and contextual information.

## 1 Introduction

Adversarial examples have effectively highlighted the deficiencies of state-of-the-art models for many natural language processing tasks, e.g. question answering (Jia and Liang, 2017; Chen et al., 2018; Ribeiro et al., 2018), textual entailment (Zhao et al., 2018; Glockner et al., 2018), and text classification (Alzantot et al., 2018; Iyyer et al., 2018). Zhang et al. (2019) introduce PAWS, which has adversarial paraphrase identification pairs with high lexical overlap, like *flights from New York to Florida* and *flights from Florida to New York*. Such pairs stress the importance of modeling sentence structure and context because they have high word overlap ratio but different semantic meaning. In addition to revealing failures of state-of-the-art models, research on adversarial examples has generally shown that augmenting training data with good adversarial examples can boost performance for some models—providing greater clarity to the modeling landscape as well providing new headroom for further improvements.

Most previous work focuses only on English despite the fact that the problems highlighted by adversarial examples are shared by other languages. Existing multilingual datasets for paraphrase identification, e.g. Multi30k (Elliott et al., 2016) and Opusparcus (Creutz, 2018), lack challenging examples like PAWS. The lack of high-quality adversarial examples in other languages makes it difficult to benchmark model improvements. We bridge this gap by creating Cross-lingual PAWS (PAWS-X), an extension of the Wikipedia portion of the PAWS evaluation and test examples to six languages: Spanish, French, German, Chinese, Japanese, and Korean. This new corpus consists of 23,659 *human* translated example pairs with paraphrase judgments in each target language. Like previous work on multilingual corpus creation (Conneau et al., 2018), we machine translate the original PAWS English training set (49,401 pairs). Note that all translated pairs still have high word overlap and they inherit semantic similarity labels from the original PAWS examples; thus, the resulting dataset preserves the ability of probing structure and context sensitivity for models. We also machine translate the evaluation pairs of each language into English to establish the baseline performance of a translate-then-predict strategy. The PAWS-X dataset, including both the new human translated pairs and the machine translated examples, is available for download at https://github.com/google-research-datasets/paws.

*equal contribution



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3687–3692,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| Language | Text |
|---|---|
| *Original Pair (id: 000005309_9438, label: not-paraphrasing)* | |
| **en** | However, in order to defeat Slovak, Derek must become a vampire attacker.<br>However, in order to become Slovak, Derek must defeat a vampire assassin. |
| *Human Translated Pairs* | |
| **fr** | Toutefois, pour battre Slovak, Derek doit devenir un vampire attaquant.<br>Cependant, pour devenir Slovak, Derek doit vaincre un vampire assassin. |
| **es** | Sin embargo, para derrotar a Slovak, Derek debe convertirse en un atacante vampiro.<br>Sin embargo, para poder convertirse en Slovak, Derek debe derrotar a un asesino de vampiros. |
| **de** | Um Slovak zu besiegen, muss Derek jedoch zum Vampirjäger werden.<br>Um jedoch Slowake zu werden, muss Derek einen Vampirjäger besiegen. |
| **zh** | 但为击败斯洛伐克，德里克必须成为吸血鬼攻击者。<br>然而，为了成为斯洛伐克人，德里克必须击败吸血鬼刺客。 |
| **ja** | ただし、スロバークを倒すためには、デレクは吸血鬼アタッカーになる必要があります。<br>しかし、デレクがスロバックになるには、バンパイア・アサシンを倒さなければならない。 |
| **ko** | 하지만 Slovak이 되기 위해 Derek은 반드지 뱀파이어 암살자를 물리쳐야만 합니다.<br>하지만 Slovak을 물리치기 위해 Derek은 뱀파이어 사냥꾼이 되어야만 했습니다. |

Table 1: Examples of human translated pairs for each of the six languages.

Our experiments show that PAWS-X effectively measures the multilingual adaptability of models and how well they capture context and word order. The state-of-the-art multilingual BERT model (Devlin et al., 2019) obtains a 32% (absolute) accuracy improvement over a bag-of-words model. We also show that machine translation helps and works better than a zero-shot strategy. We find that performance on German, French, Spanish is overall better than Chinese, Japanese and Korean.

## 2 PAWS-X Corpus

The core of our corpus creation procedure is to translate the Wikipedia portion of the original PAWS corpus from English (en) to six languages: French (fr), Spanish (es), German (de), Chinese (zh), Japanese (ja), and Korean (ko). To this end, we hire human translators to translate the development and test sets, and use a neural machine translation (NMT) service[1] to translate the training set.

We choose translation instead of repeating the PAWS data generation approach (Zhang et al., 2019) to other languages. This has at least three advantages. First, human translation does not require high-quality multilingual part-of-speech taggers or named entity recognizers, which play a key role in the data generation process used in Zhang et al. (2019). Second, human translators are trained to produce the target sentence while preserving meaning, thereby ensuring high data quality. Third, the resulting data can provide a new testbed for cross-lingual transfer techniques because examples in all languages are translated from the same sources. For example, PAWS-X could be used to evaluate whether a German or French sentence is a paraphrase of a Chinese or Japanese one.

[1] https://cloud.google.com/translate/

| | fr | es | de | zh | ja | ko |
|---|---|---|---|---|---|---|
| dev | 1,992 | 1,962 | 1,932 | 1,984 | 1,980 | 1,965 |
| test | 1,985 | 1,999 | 1,967 | 1,975 | 1,946 | 1,972 |

Table 2: Examples translated per language.

**Translating Evaluation Sets** We obtain human translations on a random sample of 4,000 sentence pairs from the PAWS development set for each of the six languages (48,000 translations). The manual translation is performed by 10-20 in-house professionals that are native speakers of each language. A randomly sampled subset is presented and validated by a second worker. The final delivery is guaranteed to have less than 5% word level error rate. The sampled 4,000 pairs are split into new development and test sets, 2,000 pairs for each.

Due to time and cost constraints, we could not translate all 16,000 examples in both of original PAWS development and test set. Each sentence in a pair is presented independently so that translation is not affected by context. In our initial studies we noticed that sometimes it was difficult to translate an entity mention. We therefore ask translators to translate entity mentions, but different translators may have different preferences according to their background knowledge. Table 1 gives example translated pairs in each language.

**Resulting Corpus** Some sentences could not be be translated. Table 2 shows the final counts translated to each language. Most of the untranslated

sentences were due to incompleteness or ambiguities, such as *It said that Easipower was*, and *Park Green took over No*. These sentences are likely from the adversarial generation process when creating PAWS. On average less than 2% of the pairs are not translated, and we simply exclude them.

The authors further verified translation quality for a random sample of ten pairs in each language. PAWS-X includes 23,459 human-translated pairs, including 11,815 and 11,844 pairs in development and test, respectively. Finally, original PAWS labels (paraphrase or not paraphrase) are mapped to the translations. Positive pairs account for 44.0% of development sets and 45.4% of test respectively–close to the PAWS label distribution.

Translation brings new challenges to the paraphrasing identification task. An entity can be translated differently, such as *Slovak* and *Slowake* (Table 1) and models need to capture that these refer to the same entity. In a more challenging example, *Four Rivers*, *Audubon* and *Shawnee Trails* are translated in just one of the sentences:

**en** *s1* From the merger of the Four Rivers Council and the Audubon Council, the Shawnee Trails Council was born.

*s2* Shawnee Trails Council was formed from the merger of the Four Rivers Council and the Audubon Council.

**zh** *s1* Four Rivers 委员会与 Audubon 委员会合并后，Shawnee Trails 委员会得以问世.

*s2* 肖尼小径(Shawnee Trails) 委员会由合并四河 (Four Rivers) 委员会和 奥杜邦 (Audubon) 委员会成立.

In the zh-s2 example, the parentheses give English glosses of Chinese entity mentions.

## 3 Evaluated Methods

The goal of PAWS-X is to probe models' ability to capture structure and context in a multilingual setting. We consider three models with varied complexity and expressiveness. The first baseline is a simple bag-of-words (**BOW**) encoder with cosine similarity. It uses unigram to bigram token encoding as input features and takes a cosine value above 0.5 as a paraphrase. The second model is **ESIM**, Enhanced Sequential Inference Model (Chen et al., 2017). Following Zhang et al. (2019), ESIM encodes each sentence using a BiLSTM, and passes the concatenation of encodings through a feed-forward layer for classification. The additional layers allow ESIM to capture more complex sentence interaction than cosine similarity. Third, we evaluate **BERT**, Bidirectional Encoder Representations from Transformers (Devlin et al., 2019), which recently achieved state-of-the-art results on eleven natural language processing tasks.

| | BOW | ESIM | BERT |
|---|---|---|---|
| Non-local context | × | ✓ | ✓ |
| Word interaction | × | × | ✓ |
| Translate Train | ✓ | ✓ | ✓ |
| Translate Test | ✓ | ✓ | ✓ |
| Zero Shot | × | × | ✓ |
| Merged | × | × | ✓ |

Table 3: Complexity of each evaluated model and the training/evaluation strategies being tested.

We evaluate all models with two strategies (Conneau et al., 2018): (1) **Translate Train**: the English training data is machine-translated into each target language to provide data to train each model and (2) **Translate Test**: train a model using the English training data, and machine-translate all test examples to English for evaluation.

Multilingual BERT is a single model trained on 104 languages, which enables experiments with cross-lingual training regimes. (1) **Zero Shot**: the model is trained on the PAWS English training data, and then directly evaluated on all others. Machine translation is **not** involved in this strategy. (2) **Merged**: train a multilingual model on all languages, including the original English pairs and machine-translated data in all other languages.

Table 3 summarizes the models with respect to whether they represent non-local contexts or support cross-sentential word interaction, plus which strategies are evaluated for each model.

## 4 Experiments and Results

We use the latest public multilingual BERT base model with 12 layers[2] and apply the default fine-tuning strategy with batch size 32 and learning rate 1e-5. For BOW and ESIM, we use our own implementations and 300 dimensional multilingual word embeddings from fastText.[3] We allow fine-tuning word embeddings during training, which gives better empirical performance.

We use two metrics: classification accuracy and area-under-curve scores of precision-recall curves (AUC-PR). For BERT, probability scores for the positive class is used to compute AUC-PR. For BOW and ESIM a cosine threshold of 0.5 is used to compute accuracy. In all experiments, the best

[2] http://goo.gl/language/bert
[3] https://fasttext.cc/

| Method | Accuracy | | | | | | | AUC-PR | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | en | fr | es | de | zh | ja | ko | en | fr | es | de | zh | ja | ko |
| **BOW** | | | | | | | | | | | | | | |
| Translate Train | 55.8 | 51.7 | 47.9 | 50.2 | 54.5 | 55.1 | 56.7 | 41.1 | 48.9 | 46.8 | 46.4 | 50.0 | 48.7 | 49.3 |
| Translate Test | – | 54.9 | 54.7 | 55.2 | 55.3 | 55.9 | 55.2 | – | 46.3 | 45.5 | 45.8 | 50.9 | 46.8 | 48.5 |
| **ESIM** | | | | | | | | | | | | | | |
| Translate Train | 67.2 | 66.2 | 66.0 | 63.7 | 60.3 | 59.6 | 54.2 | 69.6 | 67.0 | 64.2 | 59.2 | 58.2 | 56.3 | 50.5 |
| Translate Test | – | 66.2 | 66.3 | 66.0 | 62.0 | 62.3 | 60.6 | – | 68.4 | 69.5 | 68.2 | 62.3 | 61.8 | 60.3 |
| **BERT** | | | | | | | | | | | | | | |
| Translate Train | 93.5 | 89.3 | 89.0 | 85.3 | 82.3 | 79.2 | 79.9 | **97.1** | 93.6 | 92.4 | 92.0 | 87.4 | 81.4 | 82.4 |
| Translate Test | – | 88.7 | 89.3 | 88.4 | 79.3 | 75.3 | 72.6 | – | 93.8 | 93.1 | 92.9 | 85.1 | 80.9 | 80.1 |
| Zero shot | – | 85.2 | 86.0 | 82.2 | 75.8 | 70.5 | 71.7 | – | 91.0 | 90.5 | 89.4 | 79.6 | 72.7 | 75.5 |
| Merged | **93.8** | **90.8** | **90.7** | **89.2** | **85.4** | **83.1** | **83.9** | 96.5 | **94.0** | **92.9** | **92.9** | **88.9** | **86.0** | **86.3** |

Table 4: Accuracy (%) and AUC-PR (%) of each approach. Best numbers in each column are marked in bold.

| Method | | Averaged | |
|---|---|---|---|
| | | Accuracy | AUC-PR |
| **BOW** | Translate Train | 52.7 | 48.4 |
| | Translate Test | 55.2 | 47.3 |
| **ESIM** | Translate Train | 61.7 | 59.2 |
| | Translate Test | 63.9 | 65.1 |
| **BERT** | Translate Train | 84.2 | 88.2 |
| | Translate Test | 82.3 | 87.6 |
| | Zero Shot | 78.6 | 83.1 |
| | Merged | **87.2** | **90.2** |

Table 5: Average Accuracy (%) and AUC-PR (%) over the six languages.

model checkpoint is chosen based on accuracy on development sets and report results on testing sets.

**Results** Table 4 shows the performance of all methods and languages. Table 5 summarizes the average results for the six non-English languages.

**Model Comparisons:** On both Translate Train and Translate Test, BERT consistently outperforms both BOW and ESIM by a substantial margin (>15% absolute accuracy gains) across all seven languages. BERT Translate Train achieves an average 20% accuracy gain. This result demonstrates that PAWS-X effectively measures models' sensitivity to word order and syntactic structure.

**Training/Evaluation Strategies:** As Table 4 and 5 show, the Zero Shot strategy yields the lowest performance compared to other strategies on BERT. This is evidence that machine-translated data helps in the multilingual scenario. Indeed, when training on machine-translated examples in all languages (Merged), the model achieves the best performance, with 8.6% accuracy and 7.1% AUC-PR average gains over Zero Shot.

BERT and ESIM show different performance patterns on Translate Train and Translate Test. Translate Test appears to give consistently better performance then Translate Train on ESIM, but not on BERT. This may be because multilingual BERT is pre-trained on over one hundred languages; hence BERT provides better initialization for non-English languages than ESIM (which relies on fastText embeddings). The gap between training on English and on other languages is therefore smaller on BERT than on ESIM, which makes Translate Train work better on BERT.

| | 0 | 1-2 | 3-4 | 5-6 | 7 |
|---|---|---|---|---|---|
| # | 32 | 52 | 140 | 542 | 1234 |
| % | 1.6 | 2.6 | 7.0 | 27.1 | 61.7 |

Table 6: The count of examples by number of languages (of 7) that agree with the gold label in test set.

**Language Difference:** Across all models and approaches, performance on Indo-European languages (German, French, Spanish) is consistently better than CJK (Chinese, Japanese, Korean). The performance difference is particularly noticeable on Zero Shot. This can be explained from two perspectives. First, the MT system we used works better on Indo-European languages than on CJK. Second, the CJK family is more typologically and syntactically different from English. For example, in table 1, *Slowake* in German is much closer to the original term *Slovak* in English, compared with its Chinese translation 斯洛伐克. This at least partly explains why performance on CJK is particularly poor in Zero Shot.

**Error Analysis:** To gauge the difficulty of each example for the best model (BERT-merged), Table 6 shows the count of examples based on how many languages for the same pair are assigned the correct label in test set. The majority of the examples are easy, with 61.7% correct in all languages. Of the 32 examples that failed in all languages, most are hard or highly ambiguous. Some have incorrect gold labels or were generated incorrectly in the original PAWS data.

The following is a sample of these.

a1 On July 29, 1791, Sarah married Lea Thomas Wright Hill (1765–1842) at St. Martin's Church in Birmingham and had 8 children.

a2 Thomas Wright Hill married Sarah Lea (1765–1842) on 29 July 1791 at St Martin's Church, Birmingham and had 8 children. **match**

b1 He established himself eventually in the northwest of Italy, apparently supported by Guy, where he probably comes "title".

b2 He eventually established himself in northwestern Italy, apparently supported by Guy, where he probably received the title of "comes". **not_match**

We also considered examples that are correctly predicted in just half of the languages. Some of these failed because of translation noise, e.g. inconsistent entity translations (as shown in §2).

## 5 Conclusion

We introduce PAWS-X, a challenging paraphrase identification dataset with 23,659 *human* translated evaluation pairs in six languages. Our experimental results showed that PAWS-X effectively measures sensitivity of models to word order and the efficacy of cross-lingual learning approaches. It also leaves considerable headroom as a new challenging benchmark to drive multilingual research on the problem of paraphrase identification.

## Acknowledgments

We would like to thank our anonymous reviewers and the Google AI Language team, especially Luheng He, for the insightful comments that contributed to this paper. Many thanks also to the translate team, especially Mengmeng Niu, for the help with the annotations.

## References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896, Brussels, Belgium. Association for Computational Linguistics.

Hongge Chen, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, and Cho-Jui Hsieh. 2018. Attacking visual language grounding with adversarial examples: A case study on neural image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2587–2597, Melbourne, Australia. Association for Computational Linguistics.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1657–1668, Vancouver, Canada. Association for Computational Linguistics.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485, Brussels, Belgium. Association for Computational Linguistics.

Mathias Creutz. 2018. Open subtitles paraphrase corpus for six languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*, Miyazaki, Japan. European Languages Resources Association (ELRA).

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Desmond Elliott, Stella Frank, Khalil Sima'an, and Lucia Specia. 2016. Multi30K: Multilingual English-German image descriptions. In *Proceedings of the 5th Workshop on Vision and Language*, pages 70–74, Berlin, Germany. Association for Computational Linguistics.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking NLI systems with sentences that require simple lexical inferences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 650–655, Melbourne, Australia. Association for Computational Linguistics.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885, New Orleans, Louisiana. Association for Computational Linguistics.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, Copenhagen, Denmark. Association for Computational Linguistics.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Semantically equivalent adversarial rules for debugging NLP models. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 856–865, Melbourne, Australia. Association for Computational Linguistics.

Yuan Zhang, Jason Baldridge, and Luheng He. 2019. PAWS: Paraphrase adversaries from word scrambling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1298–1308, Minneapolis, Minnesota. Association for Computational Linguistics.

Zhengli Zhao, Dheeru Dua, and Sameer Singh. 2018. Generating natural adversarial examples. In *International Conference on Learning Representations*.

# Pretrained Language Models for Sequential Sentence Classification

**Arman Cohan** [1★] **Iz Beltagy** [1★] **Daniel King** [1] **Bhavana Dalvi** [1] **Daniel S. Weld** [1,2]

[1]Allen Institute for Artificial Intelligence, Seattle, WA

[2]Allen School of Computer Science & Engineering, University of Washington, Seattle, WA

{armanc,beltagy,daniel,bhavanad,danw}@allenai.org

## Abstract

As a step toward better document-level understanding, we explore classification of a sequence of sentences into their corresponding categories, a task that requires understanding sentences in context of the document. Recent successful models for this task have used hierarchical models to contextualize sentence representations, and Conditional Random Fields (CRFs) to incorporate dependencies between subsequent labels. In this work, we show that pretrained language models, BERT (Devlin et al., 2018) in particular, can be used for this task to capture contextual dependencies without the need for hierarchical encoding nor a CRF. Specifically, we construct a joint sentence representation that allows BERT Transformer layers to directly utilize contextual information from all words in all sentences. Our approach achieves state-of-the-art results on four datasets, including a new dataset of structured scientific abstracts.

## 1 Introduction

Inspired by the importance of document-level natural language understanding, we explore classification of a sequence of sentences into their respective roles or functions. For example, one might classify sentences of scientific abstracts according to rhetorical roles (e.g., Introduction, Method, Result, Conclusion, etc). We refer to this task as Sequential Sentence Classification (SSC), because the meaning of a sentence in a document is often informed by context from neighboring sentences.

Recently, there have been a surge of new models for contextualized language representation, resulting in substantial improvements on many natural language processing tasks. These models use multiple layers of LSTMs (Hochreiter and Schmidhuber, 1997) or Transformers (Vaswani et al., 2017), and are pretrained on unsupervised text with language modeling objectives such as next word prediction (Peters et al., 2018; Radford et al., 2018) or masked token prediction (Devlin et al., 2018; Dong et al., 2019). BERT is among the most successful models for many token- and sentence-level tasks (Devlin et al., 2018; Liu et al., 2019). In addition to a masked token objective, BERT optimizes for next sentence prediction, allowing it to capture sentential context.

These objectives allow BERT to learn some document-level context through pretraining. In this work we explore the use of BERT for SSC. For this task, prior models are primarily based on hierarchical encoders over both words and sentences, often using a Conditional Random Field (CRF) (Lafferty et al., 2001) layer to capture document-level context (Cheng and Lapata, 2016; Jin and Szolovits, 2018; Chang et al., 2019). These models encode and contextualize sentences in two consecutive steps. In contrast, we propose an input representation which allows the Transformer layers in BERT to directly leverage contextualized representations of all words in all sentences, while still utilizing the pretrained weights from BERT. Specifically, we represent all the sentences in the document as one long sequence of words with special delimiter tokens in between them. We use the contextualized representations of the delimiter tokens to classify each sentence. The transformer layers allow the model to finetune the weights of these special tokens to encode contextual information necessary for correctly classifying sentences in context.

We apply our model to two instances of the SSC task in scientific text that can benefit from better contextualized representations of sentences: scientific abstract sentence classification and extractive summarization of scientific documents.

★Equal contribution.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3693–3699, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

Our contributions are as follows:

*(i)* We present a BERT-based approach for SSC that jointly encodes all sentences in the sequence, allowing the model to better utilize document-level context. *(ii)* We introduce and release CSABSTRUCT, a new dataset of manually annotated sentences from computer science abstracts. Unlike biomedical abstracts which are written with explicit structure, computer science abstracts are free-form and exhibit a variety of writing styles, making our dataset more challenging than existing datasets for this task. *(iii)* We achieve state-of-the-art (SOTA) results on multiple datasets of two SSC tasks: scientific abstract sentence classification and extractive summarization of scientific documents.[1]

## 2 Model

In Sequential Sentence Classification (SSC), the goal is to classify each sentence in a sequence of $n$ sentences in a document. We propose an approach for SSC based on BERT to encode sentences in context. The BERT model architecture consists of multiple layers of Transformers and uses a specific input representation, with two special tokens, `[CLS]` and `[SEP]`, added at the beginning of the input sentence pair and between the sentences (or bag of sentences) respectively. The pretrained multi-layer TRANSFORMER architecture allows the BERT model to contextualize the input over the entire sequence, allowing it to capture necessary information for correct classification. To utilize this for the SSC task, we propose a special input representation without any additional complex architecture augmentation. Our approach allows the model to better incorporate context from all surrounding sentences.

Figure 1 gives an overview of our model. Given the sequence of sentences $\mathbf{S} = \langle \mathbf{S_1}, ..., \mathbf{S_n} \rangle$ we concatenate the first sentence with BERT's delimiter, `[SEP]`, and repeat this process for each sentence, forming a large sequence containing all tokens from all sentences. After inserting the standard `[CLS]` token at the beginning of this sequence, we feed it into BERT. Unlike BERT, which uses the `[CLS]` token for classification, we use the encodings of the `[SEP]` tokens to classify each sentence. We use a multi-layer feedforward network on top of the `[SEP]` representations of each sentence to classify them to their corresponding categories.[2] Intuitively, through BERT's pretraining, the `[SEP]` tokens learn sentence structure and relations between continuous sentences (through the next sentence objective). The model is then finetuned on task-specific training data, where most of the model parameters are already pretrained using BERT and only a thin task-specific network on top is needed. During finetuning[3] the model learns appropriate weights for the `[SEP]` token to allow it to capture contextual information for classifying sentences in the sequence. This way of representing a sequence of sentences allows the self-attention layers of BERT to directly leverage contextual information from all words in all sentences, while still utilizing the pretrained weights from BERT. This is in contrast to existing hierarchical models which encode then contextualize sentences in two consecutive steps.[4]

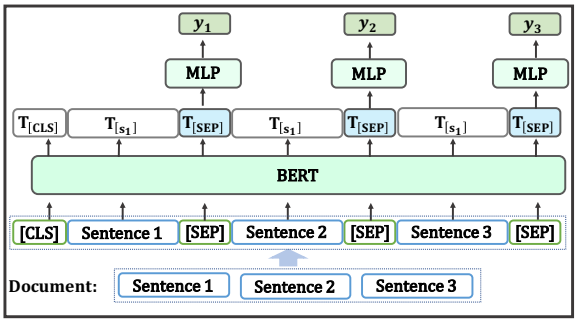


Figure 1: Overview of our model. Each `[SEP]` token is mapped to a contextualized representation of its sentence and then used to predict a label $y_i$ for sentence$_i$.

**Handling long sequences** Released BERT pretrained weights support sequences of up to 512 wordpieces (Wu et al., 2016). This is limiting for our model on datasets where the length of each document is large, as we represent all sentences in one single sequence. However, the semantics of a sentence are usually more dependent on local context, rather than all sentences in a long docu-

[1]Code & data: https://github.com/allenai/sequential_sentence_classification

[2]It is also possible to add another special token (e.g., `[CLS]`) at the beginning of each sentence and perform classification on that token. Empirically, we found the approaches to perform similarly.

[3] Following terminology from Howard and Ruder (2018), "finetuning" refers to "training" a model that was previously pretrained. We use both terms interchangeably.

[4]It is possible to add a CRF layer or another contextualizing layer on top of `[SEP]` tokens in our model, but empirically, we did not find this addition to be helpful. One explanation is that the self-attention layers of our model are already capturing necessary contextual information from the document.

| | PubMed | NICTA | CSAbstruct | CSPubSum |
|---|---|---|---|---|
| # docs | 20K | 1K | 2.2K | 21K |
| # sents | 225K | 21K | 15K | 601K |

Table 1: Statistics of the evaluation datasets. The first three datasets are for the abstract sentence classification task and the last dataset is for summarization.

| CSAbstruct characteristics | | |
|---|---|---|
| Doc length (sentences) | avg : 6.7 | std : 1.99 |
| Sentence length (words) | avg : 21.8 | std : 10.0 |
| Label distribution | BACKGROUND | 0.33 |
| | METHOD | 0.32 |
| | RESULT | 0.21 |
| | OBJECTIVE | 0.12 |
| | OTHER | 0.03 |

Table 2: Characteristics of our CSABSTRUCT dataset

ment. Therefore, we set a threshold on the number of sentences in each sequence. We recursively bisect the document until each split has less sentences than the specified threshold. At a limit of 10 sentences, only one division is needed to fit nearly all examples for the abstract sentence classification datasets. A limitation of this approach is that sentences on the edge of the splits could lose context from the previous(next) split. We leave this limitation to future work.

## 3 Tasks and Datasets

This section describes our tasks and datasets, and any model changes that are task-specific (see Table 1 for comparison of evaluation datasets).

### 3.1 Scientific abstract sentence classification

This task requires classifying sentences in scientific abstracts into their rhetorical roles (e.g., INTRODUCTION, METHOD, RESULTS, etc). We use the following three datasets in our experiments.

**PUBMED-RCT** (Dernoncourt and Lee, 2017) contains 20K biomedical abstracts from PubMed, with sentences classified as one of 5 categories {BACKGROUND, OBJECTIVE, METHOD, RESULT, CONCLUSION}. We use the preprocessed version of this dataset by Jin and Szolovits (2018).

**CSABSTRUCT** is a new dataset that we introduce. It has 2,189 manually annotated computer science abstracts with sentences annotated according to their rhetorical roles in the abstract, similar to the PUBMED-RCT categories. See §3.3 for details.

**NICTA** (Kim et al., 2011) contains 1,000 biomedical abstracts with sentences classified into PICO categories (Population, Intervention, Comparison, Outcome) (Richardson et al., 1995).

### 3.2 Extractive summarization of scientific documents

This task is to select a few text spans in a document that best summarize it. When the spans are sentences, this task can be viewed as SSC, classifying each sentence as a good summary sentence or not. Choosing the best summary sentences can benefit from context of surrounding sentences. We train on CSPUBSUMEXT (Collins et al., 2017), an extractive summarization dataset of 10k scientific papers, with sentences scored as good/bad summary sentences using ROUGE overlap scores with paper highlights. For evaluation, a separate test set, CSPUBSUM, of 150 publications and their paper highlights is used.[5]

A key difference between the training of our model and that of Collins et al. (2017) is that they use the ROUGE scores to label the top (bottom) 20 sentences as positive (negative), and the rest are neutral. However, we found it better to train our model to directly predict the ROUGE scores, and the loss function we used is Mean Square Error.

### 3.3 CSABSTRUCT construction details

CSABSTRUCT is a new dataset of annotated computer science abstracts with sentence labels according to their rhetorical roles. The key difference between this dataset and PUBMED-RCT is that PubMed abstracts are written according to a predefined structure, whereas computer science papers are free-form. Therefore, there is more variety in writing styles in CSABSTRUCT. CSABSTRUCT is collected from the Semantic Scholar corpus (Ammar et al., 2018). Each sentence is annotated by 5 workers on the Figure-eight platform,[6] with one of 5 categories {BACKGROUND, OBJECTIVE, METHOD, RESULT, OTHER}. Table 2 shows characteristics of the dataset. We use 8 abstracts (with 51 sentences) as test questions to train crowdworkers. Annotators whose accuracy is less than 75% are disqualified from doing the actual annotation job. The annotations are

[5] Dataset generated using author provided scripts: https://github.com/EdCo95/scientific-paper-summarisation

[6] http://figure-eight.com

| Model | PUBMED | CSABST. | NICTA |
|---|---|---|---|
| Jin and Szolovits (2018) | 92.6 | 81.3 | 84.7 |
| BERT +Transformer | 89.6 | 78.8 | 78.4 |
| BERT +Transformer+CRF | 92.1 | 78.5 | 79.1 |
| Our model | **92.9** | **83.1** | **84.8** |

Table 3: Abstract sentence classification (micro F1).

| Model | ROUGE-L |
|---|---|
| SAF + F Ens (Collins et al., 2017) | 0.313 |
| BERT +Transformer | 0.287 |
| Our model | 0.306 |
| Our model + ABSTRACTROUGE | **0.314** |

Table 4: Results on CSPUBSUM

aggregated using the agreement on a single sentence weighted by the accuracy of the annotator on the initial test questions. A confidence score is associated with each instance based on the annotator initial accuracy and agreement of all annotators on that instance. We then split the dataset 75%/15%/10% into train/dev/test partitions, such that the test set has the highest confidence scores. Agreement rate on a random subset of 200 sentences is 75%, which is quite high given the difficulty of the task. Compared with PUBMED-RCT, our dataset exhibits a wider variety of writing styles, since its abstracts are not written with an explicit structural template.

## 4 Experiments

**Training and Implementation** We implement our models using AllenNLP (Gardner et al., 2018). We use SCIBERT pretrained weights (Beltagy et al., 2019) in both our model and BERT-based baselines, because our datasets are from the scientific domain. As in prior work (Devlin et al., 2018; Howard and Ruder, 2018), for training, we use dropout of 0.1, the Adam (Kingma and Ba, 2015) optimizer for 2-5 epochs, and learning rates of $5e^6$, $1e^5$, $2e^5$, or $5e^5$. We use the largest batch size that fits in the memory of a Titan V GPU (between 1 to 4 depending on the dataset/model) and use gradient accumulation for effective batch size of 32. We report the average of results from 3 runs with different random seeds for the abstract sentence classification datasets to control potential non-determinism associated with deep neural models (Reimers and Gurevych, 2017). For summarization, we use the best model on the validation set. We choose hyperparameters based on the best performance on the validation set. We release our code and data to facilitate reproducibility.[7]

**Baselines** We compare our approach with two strong BERT-based baselines, finetuned for the task. The first baseline, BERT+Transformer, uses the `[CLS]` token to encode individual sentences as described in Devlin et al. (2018). We add an additional Transformer layer over the `[CLS]` vectors to contextualize the sentence representations over the entire sequence. The second baseline, BERT+Transformer+CRF, additionally adds a CRF layer. Both baselines split long lists of sentences into splits of length 30 using the method in §2 to fit into the GPU memory.

We also compare with existing SOTA models for each dataset. For the PUBMED-RCT and NICTA datasets, we report the results of Jin and Szolovits (2018), who use a hierarchical LSTM model augmented with attention and CRF. We also apply their model on our dataset, CSABSTRUCT, using the authors' original implementation.[8] For extractive summarization, we compare to Collins et al. (2017)'s model, `SAF+F Ens`, the model with highest reported results on this dataset. This model is an ensemble of an LSTM-based model augmented with global context and abstract similarity features, and a model trained on a set of hand-engineered features.

### 4.1 Results

Table 3 summarizes results for abstract sentence classification. Our approach achieves state-of-the-art results on all three datasets, outperforming Jin and Szolovits (2018). It also outperforms our BERT-based baselines. The performance gap between our baselines and our best model is large for small datasets (CSABSTRUCT, NICTA), and smaller for the large dataset (PUBMED-RCT). This suggests the importance of pretraining for small datasets.

Table 4 summarizes results on CSPUBSUM. Following Collins et al. (2017) we take the top 10 predicted sentences as the summary and use ROUGE-L scores for evaluation. It is clear that our approach outperforms BERT+TRANSFORMER. The BERT +TRANSFORMER+CRF baseline is not included

[7] https://github.com/allenai/sequential_sentence_classification

[8] https://github.com/jind11/HSLN-Joint-Sentence-Classification

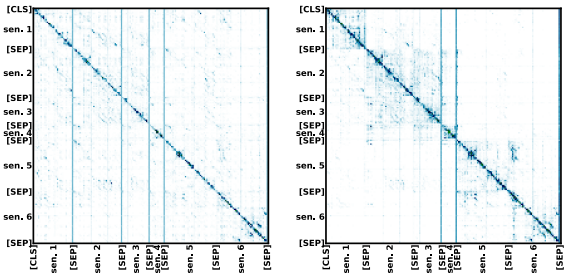


(a) Before finetuning (b) After finetuning

Figure 2: Self-attention weights of the top 2 layers of BERT for one abstract. Cell value in row i, column j, is the maximum attention weight of token i attending to token j across all 12 Transformer attention heads.

here because, as mentioned in section 3, we train our model to predict ROUGE, not binary labels as in Collins et al. (2017). As in Collins et al. (2017), we found the ABSTRACT-ROUGE feature to be useful. Our model augmented with this feature slightly outperforms Collins et al. (2017)'s model, which is a relatively complex ensemble model and uses a number of carefully engineered features for the task. Our model is a single model with only one added feature.

**Analysis** To better understand the advantage of our joint sentence encoding relative to the BERT+Transformer baseline, we qualitatively analyze examples from CSABSTRUCT that our model gets right and the baseline gets wrong. We found that 34/134 of such examples require context to classify correctly.[9]

For example, sentences 2 and 3 from one abstract are as follows: "*We present an improved oracle for the arc-eager transition system, which provides a set of optimal transitions [...].*", "*In such cases, the oracle provides transitions that will lead to the best reachable tree [...].*". In isolation, the label for sentence 3 is ambiguous, but with context from the previous sentence, it clearly falls under the METHOD category.

Figure 2 shows BERT self-attention weights for the above-mentioned abstract before and after finetuning. Before (Figure 2a), attention weights don't exhibit a clear pattern. After (Figure 2b), we observe blocks along the matrix diagonal of sentences attending to themselves, except for the block encompassing sentences 2 and 3. The words in these two sentences attend to each other, enabling the encoding of sentence 3 to capture the information needed from sentence 2 to predict its label (see Appendix A for additional patterns).

[9]Of the 1349 examples in the test set, our model gets 134 correct that the BERT +Transformer baseline gets wrong, and the baseline gets 79 correct that our model gets wrong.

## 5 Related Work

Prior work on scientific Sequential Sentence Classification datasets (e.g. PUBMED-RCT and NICTA) use hierarchical sequence encoders (e.g. LSTMs) to encode each sentence and contextualize the encodings, and apply CRF on top (Dernoncourt and Lee, 2017; Jin and Szolovits, 2018). Hierarchical models are also used for summarization (Cheng and Lapata, 2016; Nallapati et al., 2016; Narayan et al., 2018), usually trained in a seq2seq fashion (Sutskever et al., 2014) and evaluated on newswire data such as the CNN/Daily mail benchmark (Hermann et al., 2015). Prior work proposed generating summaries of scientific text by leveraging citations (Cohan and Goharian, 2015) and highlights (Collins et al., 2017). The highlights-based summarization dataset introduced by Collins et al. (2017) is among the largest extractive scientific summarization datasets. Prior work focuses on specific architectures designed for each of the tasks described in §3, giving them more power to model each task directly. Our approach is more general, uses minimal architecture augmentation, leverages language model pretraining, and can handle a variety of SSC tasks.

## 6 Conclusion and Future Work

We demonstrated how we can leverage pretrained language models, in particular BERT, for SSC without additional complex architectures. We showed that jointly encoding sentences in a sequence results in improvements across multiple datasets and tasks in the scientific domain. For future work, we would like to explore methods for better encoding long sequences using pretrained language models.

## Acknowledgments

We would like to thank Matthew Peters, Waleed Ammar and Hanna Hajishirzi for helpful discussions, Madeleine van Zuylen for help in crowdsourcing and data analysis, and the three anonymous reviewers for their comments and suggestions. Computations on beaker.org were supported in part by credits from Google Cloud. Other support includes ONR grant N00014-18-1-2193 and the WRF/Cable Professorship,

## References


Waleed Ammar, Dirk Groeneveld, Chandra Bhagavatula, Iz Beltagy, Miles Crawford, Doug Downey, Jason Dunkelberger, Ahmed Elgohary, Sergey Feldman, Vu Ha, Rodney Kinney, Sebastian Kohlmeier, Kyle Lo, Tyler C. Murray, Hsu-Han Ooi, Matthew E. Peters, Joanna Power, Sam Skjonsberg, Lucy Lu Wang, Christopher Wilhelm, Zheng Yuan, Madeleine van Zuylen, and Oren Etzioni. 2018. Construction of the literature graph in semantic scholar. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT))*.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. Scibert: A pretrained language model for scientific text. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Ming-Wei Chang, Kristina Toutanova, Kenton Lee, and Jacob Devlin. 2019. Language model pre-training for hierarchical document representations. *CoRR*, abs/1901.09128.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. *CoRR*, abs/1603.07252.

Arman Cohan and Nazli Goharian. 2015. Scientific article summarization using citation-context and article's discourse structure. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Ed Collins, Isabelle Augenstein, and Sebastian Riedel. 2017. A supervised approach to extractive summarisation of scientific papers. In *CoNLL*.

Franck Dernoncourt and Ji Young Lee. 2017. Pubmed 200k rct: a dataset for sequential sentence classification in medical abstracts. In *IJCNLP*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. *ArXiv*, abs/1905.03197.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew E. Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. *ArXiv*, abs/1803.07640.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in neural information processing systems*, pages 1693–1701.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9:1735–1780.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *ACL*.

Di Jin and Peter Szolovits. 2018. Hierarchical neural networks for sequential sentence classification in medical scientific abstracts. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Su Kim, David Martínez, Lawrence Cavedon, and Lars Yencken. 2011. Automatic classification of sentences to support evidence based medicine. In *BMC Bioinformatics*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

John D. Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint*.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2016. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *AAAI*.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Ranking sentences for extractive summarization with reinforcement learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke S. Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *Preprint*.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging. In *Empirical Methods in Natural Language Processing (EMNLP)*.

William S. Richardson, Mark C. Wilson, James A. Nishikawa, and Robert S. Hayward. 1995. The well-built clinical question: a key to evidence-based decisions. *ACP journal club*, 123 3:A12–3.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Gregory S. Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

# A Additional analysis

Figures 3 and 4 show attention weights of BERT before and after finetuning. We observe that before finetuning, the attention patterns on `[SEP]` tokens and periods is almost identical between sentences. However, after finetuning, the model attends to sentences differently, likely based on their different role in the sentence that requires different contextual information.

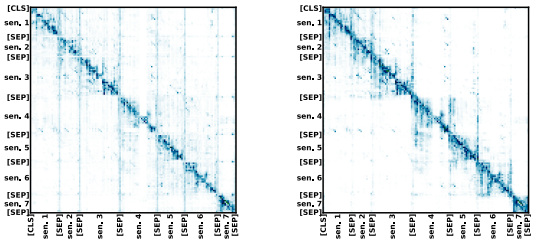

(a) Before finetuning (b) After finetuning

Figure 3: Visualization of attention weights for layer 8 of BERT before and after finetuning.

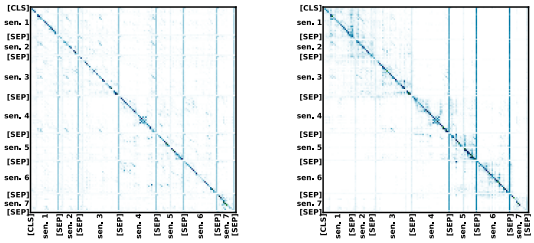

(a) Before finetuning (b) After finetuning

Figure 4: Visualization of attention weights in final layer (layer 12) of BERT before and after finetuning.

# Emergent Linguistic Phenomena in Multi-Agent Communication Games

**Laura Graesser**[†*], **Kyunghyun Cho**[†‡⋆], **Douwe Kiela**[‡]
[†] NYU; [*] Robotics at Google; [⋆] CIFAR Azrieli Global Scholar; [‡] Facebook AI Research
lauragraesser@google.com, kyunghyun.cho@nyu.edu, dkiela@fb.com

## Abstract

We describe a multi-agent communication framework for examining high-level linguistic phenomena at the community-level. We demonstrate that complex linguistic behavior observed in natural language can be reproduced in this simple setting: i) the outcome of contact between communities is a function of inter- and intra-group connectivity; ii) linguistic contact either converges to the majority protocol, or in balanced cases leads to novel creole languages of lower complexity; and iii) a linguistic continuum emerges where neighboring languages are more mutually intelligible than farther removed languages. We conclude that at least some of the intricate properties of language evolution need not depend on complex evolved linguistic capabilities, but can emerge from simple social exchanges between perceptually-enabled agents playing communication games.

## 1 Introduction

Contact linguistics (Myers-Scotton, 2002) studies what happens when two or more languages or language varieties interact. It poses several pertinent open questions that are difficult to answer: how does symmetric ("mutually intelligible") communication emerge; how do languages behave under contact; how does one language come to dominate another; how and why does extensive language contact tend to lead to simplification (e.g. in creoles); and how does a linguistic continuum come about, where neighboring languages are more intelligible than farther removed ones? In this work, we show that such linguistic phenomena emerge naturally given a few general assumptions about the organizational structure of networks of artificial agents equipped with a minimalistic form of learned communication.

We introduce a multi-agent framework for studying the emergence and evolution of language, where agents are neural networks endowed with the ability to exchange messages about their perceptual input. The advantage of this approach is that one can precisely control linguistic, environmental and algorithmic variables.

First, we investigate linguistic behavior at the agent-level, and examine when symmetric communication emerges within a linguistic community, as well as how the topological organization of communities—i.e., which other agents an agent comes into contact with and how frequently that happens—impacts convergence and learning. We then examine the behavior of communities of such agents when they come into contact, as well as how community-level topology impacts convergence, success rate and mutual intelligibility.

We demonstrate that the following linguistic behaviors emerge, which correspond to known linguistic phenomena in natural languages: 1) the outcome of contact is a function of inter- and intra-group connectivity, i.e. that languages become mutually intelligible through contact, even for agents that have not themselves been exposed to the other language, provided there is sufficient connectivity between communities; 2) linguistic contact over time either converges to the dominant majority protocol, leading to the extinction of the other language, or if the communities are balanced, gives rise to an original "creole" protocol that has lower complexity than the original languages; 3) a linguistic continuum emerges, where neighboring languages are more mutually intelligible than farther removed languages and the topology of the continuum governs its behavior.

To our knowledge, this work constitutes the first attempt at studying contact linguistic phenomena using communities of deep neural agents. Our findings indicate that intricate properties of language evolution need not depend on intrinsic properties of highly complex evolved linguistic capa-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3700–3710, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

bilities, but instead can emerge purely from social exchanges between perceptually-enabled agents with simple communicative capabilities.

## 2 Related Work

Studying language change *in vivo* is challenging, since it requires simultaneous observation of speaker and community interactions (Brooks and Ragir, 2008; Trudgill, 1974; Joseph, 2017; Christiansen and Kirby, 2003), while carefully controlling for purposes and goals (Winograd, 1971; Flores and Winograd, 1987; Nowak and Krakauer, 1999). Studies of language emergence and evolution must furthermore be conducted over a long period of time, spanning decades or even centuries. Even the Nicaraguan sign language, which emerged remarkably rapidly, took several decades to develop fully (Senghas et al., 2005). Language itself also never ceases to evolve (Fishman, 1964).

Advances in computer science have provided us with opportunities for instead investigating the emergence and evolution of languages *in vitro* using computational and mathematical models (R. Hurford, 1989; Briscoe, 2002; Kirby, 2002; Christiansen and Kirby, 2003; Kirby et al., 2008; Lewis, 2008; Skyrms, 2010). In computational approaches, communities of agents, equipped with the ability to communicate, are deployed in a simulated environment. Their communication protocol is either evolved or learned, in order to maximize some reward provided by the environment. The agents' behavior and communication are observed and used to compare against linguistic phenomena or hypothesized linguistic theories.

Computational multi-agent models are characterized by the complexity of the agents, the choice of learning algorithm, and the design of the environment and reward structure. The complexity of an artificial agent ranges from a set of simple difference equations (Grouchy et al., 2016), to a CPU-like architecture with an instruction set and registers (Knoester et al., 2007), to a co-occurrence matrix between objects and symbols (Nowak and Krakauer, 1999), to a simple single-layer neural network (Trianni and Dorigo, 2006), to a deep neural network (Lazaridou et al., 2016; Foerster et al., 2016; Jorge et al., 2016). The learning algorithm is either a variant of evolutionary algorithms (Nowak and Krakauer, 1999; Kirby, 2002; Grouchy et al., 2016), often used in the framework of Artificial Life (Bedau, 2003), or a gradient-based optimization algorithm, as is often used for training deep neural networks with a supervised or reinforcement learning objective function. The former simulates generations developing complex behavior over time, while the latter enables more sophisticated agents thanks to the recent advances in deep learning (LeCun et al., 2015). Recent years have seen intriguing new results in emergent communication, starting with Lazaridou et al. (2016) and Foerster et al. (2016), using deep neural agents (Lewis et al., 2017; Havrylov and Titov, 2017; Jorge et al., 2016; Evtimova et al., 2018; Das et al., 2018; Cao et al., 2018). Often, these approaches could be framed as special or generalized cases of Lewis's signalling game (Lewis, 2008), in which agents exchange signals to achieve a common goal. In this work, deep neural agents play games within communities of similar agents, where the aim is for agents to communicate about their perceptual input.

## 3 Multi-agent communication

In order to study emergent linguistic phenomena in a simplified but realistic setting, the communication game needs to have several properties. First, it should be *symmetric*, in that all agents should be able to act as "speaker" and "listener". Second, the agents should communicate about something *external* to themselves, i.e., about the sensory experience of something in their environment. Third, the world should be *partially observable*, implying that communication is required for solving the game successfully. **Fig. 1** shows example training data, the game setup and agent design. See the supplementary material for details.

**Reference Game** Let $G = \langle A, O, M, W, R \rangle$ be a multi-agent communication game, consisting of communities of agents $A$ communicating across the bidirectional message channel $M$ given environmental observations $O$, where the community membership of agents is defined as a graph with the agents $A$ as vertices and weighted edges $W$ that determine whether two agents are connected, i.e., $W$ specifies the topology of the network.

Each agent is a deep neural network with perceptual inputs and given a language description, as illustrated in **Fig.1**. All agents have the same network architecture because of the requirement that the game be symmetric, but each agent has its own parameters which are learned during training. Pairs of connected agents learn to play a game

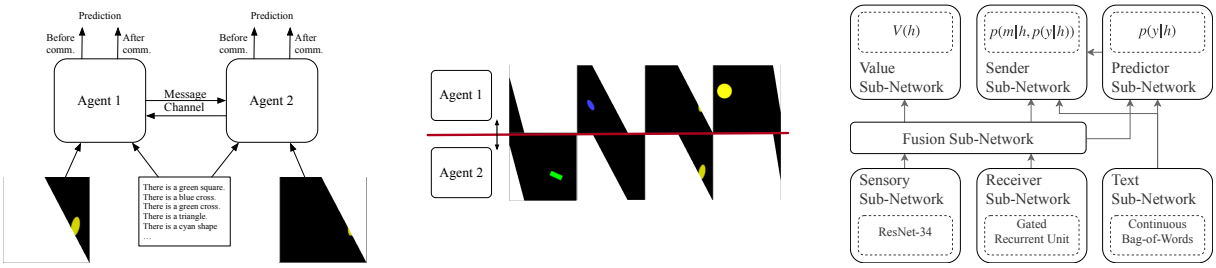


Figure 1: Overview of the communication game and agent structure. **Left**: A graphical illustration of the proposed game. Each of two agents observes a partition of an input image and decides which of ten textual captions best describes the entire image before and after exchanging messages with the other agent. **Middle**: Example training data. Only a random part of each image (dark background) is presented to one agent, necessitating communication in order to solve the game. **Right**: The modular structure of an agent.

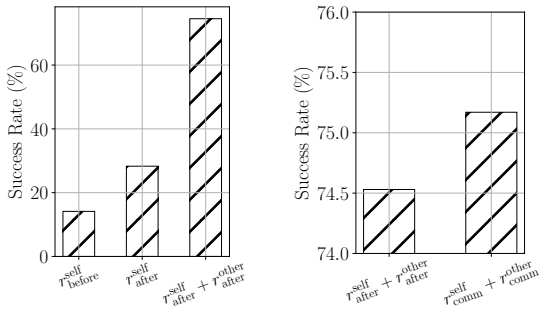


Figure 2: The chance of both agents correctly guessing the answer depends on the reward (averaged over three runs): (**left**) it is important to reward the collective behaviour in order for two agents to collaborate; (**right**) the success rate goes up when we encourage agents to maximize the accuracy *improvement* after communication.

with a reward structure $R$, designed specifically to require communication-based collaboration. The exchange of information through the communication channel $M$ can take any form. By learning to play the game, agents develop a communication protocol about the observations. This framework allows us to control for proximity constraints, population size and degree of interaction, through the underlying graph structure.

Throughout the game, each agent observes one part of an image that contains an object of a specific shape and color and is given a set of $n_{\text{cap}}$ synthetic compositional captions (e.g. "there is a green triangle") of which only one correctly describes the image[1].

The goal is for the agents to identify the correct caption $y^*$ for the image. Since each agent only has partial information, the pair of agents must cooperate via communication to be effective at solving the problem *together*.

At the beginning of the game, each agent makes an initial guess $\hat{y}_0$ of the correct answer, followed by $k$ rounds of communication in which the agents take turns transmitting a binary message to the other agent. In the experiments discussed in this paper, agents communicate using 8-bit binary message vectors. Binary message vectors have been used before for studying the emergence and evolution of language (Kirby and Hurford, 2002). While we selected this type of communication for efficiency reasons, it is straightforward to replace it with sequences of discrete symbols (Jorge et al., 2016; Havrylov and Titov, 2017; Lee et al., 2017; Cao et al., 2018; Lazaridou et al., 2018), continuous vectors or larger binary vectors. After several rounds of communication, each agent makes their final guess $\hat{y}_1$. The game is considered successful if *both* of the agents correctly guessed the answer.

During training, a pair of agents corresponding to adjacent vertices, $a_i \in A$ and $a_j \in A$, is selected at random according to interaction weights $w_{ij} \in W$. One of them is randomly picked as the starting agent. The agents then play one instance of the reference game and their parameters are updated accordingly. The community structure can change during training. For instance, we are able to merge separately trained linguistic communities into a single community and fine-tune the agents from both communities (i.e. bring the communities into contact) to investigate linguistic contact. Once training is done, we can test pairs of distant agents to understand what changed in the protocol.

**Reward** The reward structure for each agent is designed as follows. First, we reward the agent

[1] ShapeWorld (Kuhnle and Copestake, 2017) was used to derive the data set which includes both the images and captions. See the supplementary material for details.

when it correctly guesses the answer *after* communication $r^{\text{self}}_{\text{after}} = \mathbb{1}_{y^*=\hat{y}_1}$, where $\mathbb{1}$ is an indicator function, in order to encourage it to incorporate information received from the other agent. Second, we reward cooperation by giving each agent a shared reward composed of both its own and the other agent's rewards, i.e., $r_{\text{after}} = r^{\text{self}}_{\text{after}} + r^{\text{other}}_{\text{after}}$. As Fig. 2 shows, we empirically validate the importance of rewarding after-communication behavior and observe that the cooperation reward significantly boosts the success rate. Lastly, we explicitly encourage the agents to rely on communication by rewarding them for *relative improvement from communication*, rather than the success after communication: $r = r^{\text{self}}_{\text{comm}} + r^{\text{other}}_{\text{comm}}$, where $r^{\text{self}}_{\text{comm}} = r^{\text{self}}_{\text{after}} - r^{\text{self}}_{\text{before}}$ and $r^{\text{other}}_{\text{comm}} = r^{\text{other}}_{\text{after}} - r^{\text{other}}_{\text{before}}$. This final reward, which encourages both cooperation and explicit reliance on communication, reaches the highest success rate.

**Training** Each agent is trained using a hybrid of supervised and reinforcement learning. Each agent computes two predictive distributions before and after message exchange, $p^{\text{before}}(y|h)$ and $p^{\text{after}}(y|h)$, where $y$ refers to one of the $n_{\text{cap}}$ image captions and $h$ is the hidden state of the agent network, which fuses perceptual input with the last received message. Since we know the correct caption $y^*$ during training, we use supervised learning to train these two predictive distribution, $\max \log p^{\text{before}}(y^*|h) + \log p^{\text{after}}(y^*|h)$, using backpropagation and stochastic gradient descent (Rumelhart et al., 1986). Since messages are discrete, the message generating process cannot be backpropagated through. Instead, we use REINFORCE (Williams, 1992) to maximize the reward $r$; $\max \mathbb{E}_{m|h,p(y|h)}[r]$. We add the entropy of the message distribution as a regularization term, encouraging the exploration of various communication strategies in the early stage of learning.

**Loss Functions** There are four loss functions involved in each game. The first one is a prediction loss function. Given the index $y^*$ of a correct caption, the prediction loss function is

$$\mathcal{L}_{\text{pred}} = -\log p(y = y^*|h).$$

This loss is used twice based on the predictions before and after the message exchange; denoted as $\mathcal{L}^{\text{before}}_{\text{pred}}$ and $\mathcal{L}^{\text{after}}_{\text{pred}}$ respectively.

The second loss is a value loss function. After playing a game, the agent receives a reward $r$. The agent's value sub-network (see **Fig.1**) learns to predict this reward:

$$\mathcal{L}_{\text{value}} = (r - V(h))^2.$$

The third loss is a message loss function. During training, we sample one message $\tilde{m}$ from the message distribution generated by the agent $p(m|h, p(y|h))$. If this message led to a success, we increase the probability of the sampled message. Otherwise, we decrease it. The success is measured relative to the predicted value. The cost function is then:

$$\mathcal{L}_{\text{msg}} = -(r - \hat{V}(h)) \log p(m = \tilde{m}|h, p(y|h)),$$

where $\hat{V}(h)$ refers to using the predicted value but not updating the value sub-network according to this loss function. The gradient of this message loss function with respect to $p(m = \tilde{m}|h, p(y|h))$ corresponds to REINFORCE (Williams, 1992).

Lastly, we include an entropy penalty. Following (Evtimova et al., 2018), we encourage the entropy of the message distribution to be higher to facilitate exploration:

$$\mathcal{L}_{\text{entropy}} = -\mathcal{H}(p(m|h, p(y|h))).$$

The overall loss function is then the weighted sum of the four loss functions:

$$\mathcal{L} = \alpha_{\text{pred}}\mathcal{L}_{\text{pred}} + \alpha_{\text{value}}\mathcal{L}_{\text{value}} + \alpha_{\text{msg}}\mathcal{L}_{\text{msg}} + \alpha_{\text{entropy}}\mathcal{L}_{\text{entropy}},$$

where we set $\alpha_{\text{pred}} = 1.0$, $\alpha_{\text{value}} = 1.0$, $\alpha_{\text{msg}} = 1.0$ and $\alpha_{\text{entropy}} = 0.01$.

## 4 Experiments

**When Are Protocols Symmetric?** We first examine whether it is sufficient to have just two autonomous agents to develop a common communication protocol. That is, we ask whether a *symmetric* language emerges when there are only two agents in a linguistic community, or that they learn to speak distinct idiolects. We formally define "mutual intelligibility" in the communication protocol as the ability for each agent to play the game against itself. This is an elegant solution: if a shared communication protocol has emerged, the agent would not have any trouble playing a game with itself during test time (she "understands" what she "says" and "says" what she "understands"). We run five simulations with

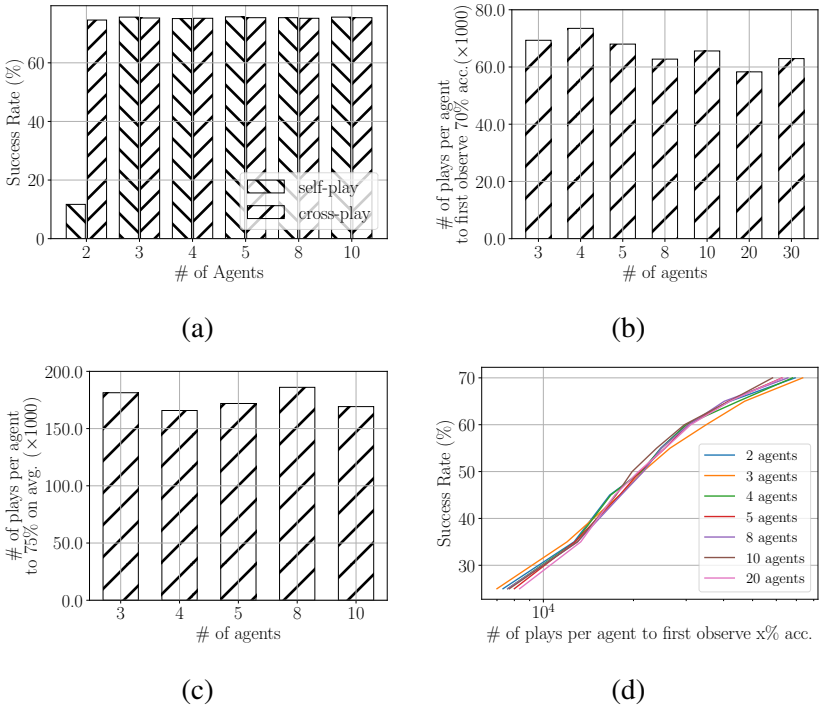


Figure 3: Emergence of symmetric linguistic protocols (averaged over five runs). **(a)** Six communities were trained: we show the success rates under self-play and cross-play. We observe that at least three agents are necessary for the emergent protocol to be symmetric—without any specialized mechanism that enforces the symmetry of emergent protocol. **(b)** The average number of plays per agent to the first observed success rate $\geq 70\%$ between a pair of agents in a linguistic community approximately stays constant with respect to the community size. **(c)** The number of plays required for each agent in a linguistic community to reach the success rate of 75% on average across all agents pairs in the community approximately stays constant with respect to the community size. **(d)** Each agent learns at approximately the same rate regardless of community size, which suggests that the emergent protocol emerges incrementally in a distributed manner rather than in a centralized way.

random initialization and examine the success rate between two agents averaged over all the test examples, where the agents succeed on each example if both of them correctly guess the answer after communication.

As shown in **Fig. 3** (a), two agents can play the game with a high success rate when they play with each other (cross-play). However, the success rate drops to random chance (10%) when each agent plays against itself (self-play). This suggests that the emergent communication protocol is not symmetric: each agent has developed its own protocol, to which the other has adapted, leading to two incompatible idiolects, similar to previous observations in both fully (Matignon et al., 2012) and partially observable settings (Lanctot et al., 2017).

We then run additional experiments having more than two agents, where every pair of agents $(v_i, v_j)$ interacts with an equal interaction intensity $w_{ij} = c$ (the success rate is averaged over all possible pairs of agents, unless stated otherwise). We notice that the success rates between self-play and cross-play are indistinguishable, strongly implying that a common, shared language emerges as a social convention if and only if we have more than two language users. This finding demonstrates that it is not strictly necessary to specifically equip an agent with an innate mechanism that ties listening and speaking, such as the obverter technique (Oliphant and Batali, 1997; Choi et al., 2018), nor any explicit community-wide coordination. All that is needed in order for a common language to emerge, at least within this framework, is a minimum number of agents.

We observe no detrimental effect to increasing the number of agents per linguistic community, even though more agents have to come to agree on a single protocol. As shown in **Fig. 3** (b–c), it takes approximately 60–65,000 plays per agent for us to observe the first instance of a pair of agents reaching the success rate of 70% regardless of the community size. Similarly, it takes approximately

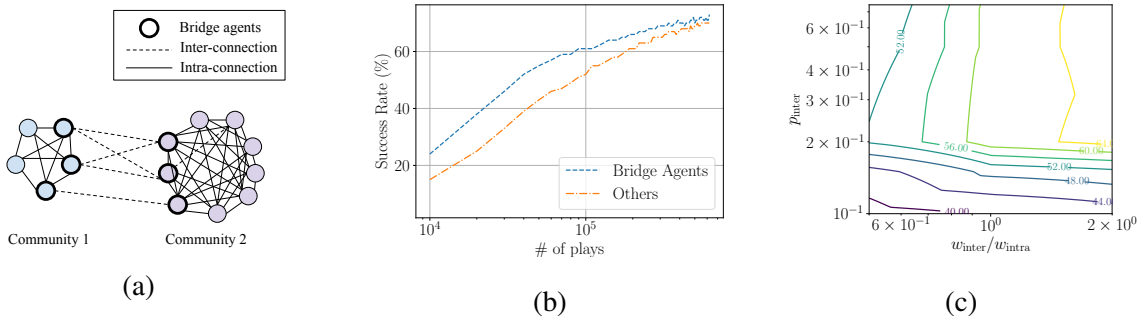


Figure 4: The consequences of contact between two linguistic communities with initially distinct protocols. **(a)** Two linguistic communities come in contact. **(b)** Two communities of population ten each came in contact with the ratio of learning frequencies $(K\ w_{\text{inter}})/(L\ w_{\text{intra}}) = 1$ and the inter-group connectivity $p_{\text{inter}} = 0.2$, after being separately trained in isolation (averaged over five runs). The bridge agents, who interact with the agents from the other community, learn faster and better the new, shared emergent protocol. All the other agents however also rapidly learn to communicate with the agents from the other community, although they never interact directly with them. **(c)** Contour plot visualization of the success rate after 200,000 plays after the contact by two linguistic communities while varying the ratio of learning frequencies $(Kw_{\text{inter}})/(Lw_{\text{intra}})$ and the inter-group connectivity $p_{\text{inter}}$ (linearly interpolated from 15 experiments.) We observe that the success rate, which measures the level of convergence of two protocols, requires a certain level of the inter-group connectivity ($p_{\text{inter}} > 0.2$). Even when the inter-group connectivity is high enough, we further see that the bridge agents must interact with the agents from the other community enough ($(Kw_{\text{inter}})/(Lw_{\text{intra}}) \geq 1$) for the converged protocol to be well understood by the agents from both communities.

150-200,000 plays per agent for the success rate averaged over all pairs of agents in a community to reach 75%, again, regardless of the size of the community. Surprisingly, the speed at which each agent learns stays constant with respect to the community size, as in **Fig. 3** (d). That is, we did not observe any correlation between community size and linguistic convergence of the entire community, which would probably only come into play with orders of magnitude more agents or for more complex reference games.

**Understanding Convergence** We next examine what happens when we expose different linguistic communities *to each other*. Specifically, we consider two linguistic communities of population sizes $N_1$ and $N_2$, which are trained independently from each other as fully-connected communities and have developed separate communication protocols. We bring these two communities into contact with each other by introducing a new set of inter-community edges with probability $p_{\text{inter}}$ to form a new linguistic community. We assign a weight $w_{\text{inter}}$ to all the inter-group edges and another weight $w_{\text{intra}}$ to all the intra-group edges. We then examine how "interaction intensity" relates to language shift. See **Fig. 4** (a).

We first investigate two communities of identical population sizes ($N_1 = N_2$) with the ratio of the learning frequencies of the intra-group pairs and inter-group pairs set to $(Kw_{\text{inter}})/(Lw_{\text{intra}}) = 1$, where $K$ is the number of inter-community connections and $L$ is the number of intra-community connections, and the inter-group connectivity chance set to $p_{\text{inter}} = 0.2$. We notice in **Fig. 4** (b) that the bridge agents learn to communicate better more rapidly (evident from the higher success rate among themselves), but the other agents quickly catch up (according to the success rate among themselves excluding the bridge agents), although these other agents never directly interact with agents from the other community. This finding demonstrates the rapid shift toward a common protocol in both groups where all agents learn to speak a shared language, regardless of whether they actually interact with agents from the other group.

Having established that linguistic contact leads to convergence of the communication protocol, we delve deeper into the impact of two major parameters, the ratio $(Kw_{\text{inter}})/(Lw_{\text{intra}})$ and the connectivity probability $p_{\text{inter}}$. We vary the ratio $(Kw_{\text{inter}})/(Lw_{\text{intra}})$ of the learning frequencies of the intra-group pairs and inter-group pairs between $2/1$, $1/1$ and $1/2$, while fixing the inter-group connectivity to $p_{\text{inter}} = 0.2$. After 200,000 plays, the former ($(Kw_{\text{inter}})/(Lw_{\text{intra}}) = 2/1$) converges to a more tightly coupled linguistic community,

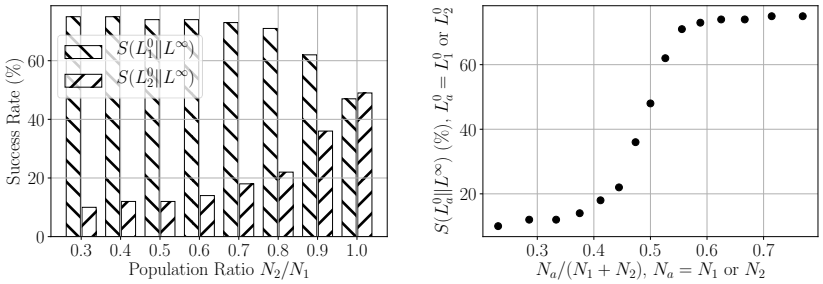


(a) (b)

Figure 5: Relationship between the common emergent protocol and the original protocols after linguistic contact between two communities (averaged over five runs). **(a)** Divergence of the common emergent protocol from the original protocols. Agents converge either to the majority protocol or to one in-between the two originals. **(b)** By varying the population ratio, it becomes clear that a near-even balance between two communities is necessary for a novel, contact protocol to emerge rather than the domination by a majority protocol.

achieving 65.6% success rate between agents that never interacted with each other. On the other hand, when the inter-group interaction occurred only half as frequently as the intra-group interaction, the agents from the two groups can play together with a much lower 52.4% success rate. We observed similar patterns over many different combinations of the ratio and inter-group connectivity. For example, we varied the inter-group connectivity $p_{\text{inter}}$ between 0.1, 0.15, 0.2, 0.5 and 0.75 while the interaction ratio was fixed to $(Kw_{\text{inter}})/(Lw_{\text{intra}}) = 2/1$. After 200,000 plays, we observed the success rates, averaged over all possible inter-group pairs, reach 42.1%, 51.1%, 65.55%, 66.65% and 66.3%, respectively. This implies that there is a critical level of inter-group connectivity (around 0.2 in this specific case) after which language propagation saturates.

In **Fig. 4** (c), we plot the interplay between the ratio $(Kw_{\text{inter}})/(Lw_{\text{intra}})$ and the inter-group connectivity $p_{\text{inter}}$ after interpolating from the fifteen experiments varying these parameters. This demonstrates that both parameters are important in determining the level of linguistic convergence.

**Birth of a New Language: Emergence of a Contact Language** We investigate the effect of the population size ratio $N_1/N_2$ between two linguistic communities when they come in contact. We study how population size is a factor in one language coming to "dominate" another language upon contact. We vary the ratio by fixing $N_1 = 10$ and varying $N_2 \in \{3, \ldots, 10\}$. Each community is pretrained in isolation to develop its own protocol before being exposed to the other. We set the interaction ratio $(Kw_{\text{inter}})/(Lw_{\text{intra}})$ to 1 and the inter-group connectivity chance $p_{\text{inter}}$ to 0.2.

We refer to the original protocols of the communities right after pretraining by $L_1^0$ and $L_2^0$. Each of these is then evolved further after these two communities come in contact, resulting in $L_1^\infty$ and $L_2^\infty$. The previous experiment on linguistic contact suggests that $L_1^\infty \approx L_2^\infty$ based on the fact that the agents from both communities can successfully play the game after coming in contact, so we refer to the final protocol as $L^\infty$. We examine how similar $L^\infty$ is to either of the original protocols, $L_1^0$ or $L_2^0$. This similarity is measured by letting the agent using $L_1^0$ or $L_2^0$ play against the one using $L^\infty$, which is naturally facilitated by the proposed framework. This "historical self-play" accuracy $S(L_\cdot^0\|L^\infty)$ reflects the similarly of the original and final protocols.

When the population ratio deviates from $N_1/N_2 = 1$, we observe that the final protocol rapidly converges to the majority protocol ($L_1^0$), evident from the near-perfect $S(L_1^0\|L^\infty)$ and the near-chance $S(L_2^0\|L^\infty)$ in **Fig. 5**. This results from the fact that members of both communities are rewarded for cooperating and playing the game well (via the bridge agents). In other words, the agents prefer to integrate or assimilate rather than segregate, similar to how it has been found that minority groups shift toward the use of dominant language (Fase et al., 1992). On the other hand, we observe $S(L_1^0\|L^\infty) \approx S(L_2^1\|L^\infty)$ and that both of these historical self-play accuracies are significantly above chance, when the population ratio is closer to or exactly 1. It is impossible to identify either $L_1^0$ or $L_2^0$ as an ancestor of $L^\infty$, but

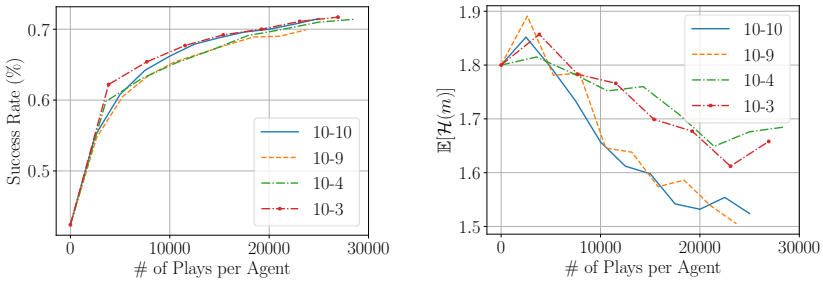


(a) Success Rate

(b) $\mathbb{E}[\mathcal{H}(p(m|h, p(y|h)))]$

Figure 6: Evolution of success rate and protocol complexity as two communities of varying populations come into contact (averaged over three runs). Agents were finetuned until the average success rate reached at least 70%. The success rate evolves similarly in terms of the number of plays per agent. **(a)** We observe significantly different levels of complexity dependent on the population ratio. **(b)** The complexity is generally lower when the sizes of two communities are more balanced (10-10 and 10-9), while the complexity does not decrease as much when there is a significant imbalance in sizes between two communities (10-4 and 10-3).

$L^{\infty}$ is rather a combination of these two original protocols, which is a key characteristic of contact languages (Matras, 2009). Both of these observations suggest the potential of the proposed framework for simulating and understanding the birth and death of new languages via linguistic contact.

We further investigate the complexity of the contact language arising from two linguistic communities coming into contact. We define complexity as the uncertainty of an agent when generating a message, and measure the entropy of the message distribution $\mathcal{H}(p(m|h, p(y|h)))$. Higher entropy indicates that agents can express states in many different ways: in other words, the more complex a language, the higher the degree of freedom. For each linguistic community, we then compute the average of their message distributions in order to characterize the complexity of a learned communication protocol.

We observe in **Fig. 6** (b) that the complexity decreases when two communities come into contact. This observation agrees with a similar phenomenon of structural simplification in creole languages which are understood to arise from the contact of two or more languages (Parkvall, 2008; Bakker et al., 2011). We also observe that the complexity plateaus earlier when there is a larger imbalance between two communities' population (10-3 and 10-4), while it drops further with more balanced communities (10-9 and 10-10). This implies that the new-born contact languages arising from the contact of two similarly-sized communities tend to be substantially simpler.

**A Linguistic Continuum of Contact** We generalize the previous setting to having $M > 2$ linguistic communities in various topologies. We start by pretraining $M$ linguistic communities of populations $N_1, N_2, \ldots, N_M$ respectively, evolving $M$ distinct communication protocols. We then chain them such that each consecutive pair, $C_i$ and $C_{i+1}$, comes in contact with a pre-specified intergroup connectivity chance $p_{\text{inter}}$ and interaction ratio $(Kw_{\text{inter}})/(Lw_{\text{intra}})$, and begin training all of the communities jointly. We study the emergence of a linguistic continuum, similar to the dialect continuums that can be found in natural languages such as the Nordic Germanic dialects of Scandinavia (Chrystal, 1987). Often, speakers on the border are mutually intelligible, while those from communities geographically separated by many intermediate ones cannot communicate.

We start by considering a chain of five communities of equal population ($M = 5$). As plotted in **Fig. 7** (a), we clearly observe the emergence of a linguistic continuum. The agents from a pair of adjacent communities can communicate with each other almost as well as those within a single community, while communicability rapidly degrades as the distance between a pair of communities grows (off-diagonal). The agents from $C_1$ and $C_5$ cannot understand each other at all, achieving the near-chance success rate. A similar continuum is observed when we increased the population of the center community two-fold (5→10). This continuum however exhibits properties different from the original chain of equal-sized communities: the center communities $C_2$, $C_3$ and $C_4$

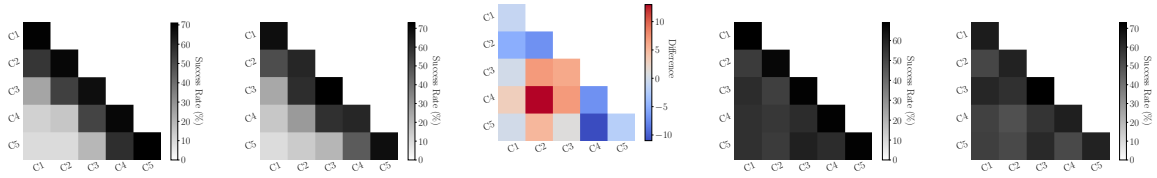

(a) 5-5-5-5-5 (b) 5-5-10-5-5 (c) Difference (b)-(a) (d) Dense (a) (e) Dense (b)

Figure 7: We plot the protocol similarities among five communities in a chain (averaged over three runs). **(a)** Neighbouring communities exhibit higher protocol similarities, but distant communities are not mutually intelligible. **(b)** With a larger community in the middle of a chain, we observe higher levels of protocol similarities among the communities in the chain. **(c)** Differences between the two, showing that the protocols near the center become more similar to each other when the center community dominates, at the expense of intelligibility between further-removed communities. **(d, e)** We do not observe a continuum when the communities are connected densely.

become more tightly coupled, as evident from the higher success rate among those in **Fig. 7** (b-c). This however happens at the cost of communicability between the agents from furthest-removed communities.

To see if the emergence of such a continuum is due to topological properties of communities, we show similarities $S(L_i^\infty \| L_j^\infty)$ among the five communities when densely connected in **Fig. 7** (d-e). Unlike for chaining, we ensure that every pair of communities comes in contact with each other in a densely connected topology. All communities are uniformly similar, confirming that the linguistic continuum arises from the topology.

## 5 Discussion and Conclusion

We have described a framework for the large-scale investigation of complex linguistic phenomena via multi-agent communication games. We started by observing that a symmetric communication protocol emerges without any innate, explicit mechanism built in an agent, when there were three or more of them in a community. We then demonstrated the emergence of several complex linguistic phenomena in this simple framework.

First, the result of linguistic contact between communities is determined by inter- and intra-group connectivity patterns. Given sufficient inter-group connectivity, languages become mutually intelligible through contact, even for agents that have not been exposed to the other language. Second, linguistic contact over time either converges to the majority protocol, leading to the extinction of the other language, or gives rise to an original "creole" protocol that has lower complexity than the original languages, if the communities are balanced. Third, a linguistic continuum emerges, where neighboring languages are more mutually intelligible than farther removed languages. The topology of the continuum governs its behavior, and a very dominant central language causes its neighbors to lose mutual intelligibility with communities that are not directly exposed to that central language.

We conclude that intricate properties of language evolution need not depend on complex evolved linguistic capabilities, but can emerge from simple social exchanges between perceptually-enabled agents playing communication games. Language evolution and its properties can be effectively and efficiently investigated in-depth under the proposed framework.

In future work, it would be useful to investigate more complicated environments and more complex agent interactions. The setting in this paper included a very simple vision task, and we suspect emergent linguistic phenomena would be more pronounced and even more interesting to study in more sophisticated settings.

## Acknowledgments

We thank the anonymous reviewers for providing helpful feedback. We thank Marco Baroni, Alex Peysakhovich and Adina Williams for their comments on an earlier draft of this work.

## References

Peter Bakker, Aymeric Daval-Markussen, Mikael Parkvall, and Ingo Plag. 2011. Creoles are typologically distinct from non-creoles. *Journal of Pidgin and Creole Languages*, 26:5–42.

Mark A Bedau. 2003. Artificial life: organization, adaptation and complexity from the bottom up. *Trends in Cognitive Sciences*, 7(11):505–512.

Ted Briscoe. 2002. *Linguistic evolution through language acquisition*. Cambridge UP.

Patricia J Brooks and Sonia Ragir. 2008. Prolonged plasticity: Necessary and sufficient for language-ready brains. *Behavioral and Brain Sciences*, 31(5):514–515.

Kris Cao, Angeliki Lazaridou, Marc Lanctot, Joel Z Leibo, Karl Tuyls, and Stephen Clark. 2018. Emergent communication through negotiation. In *ICLR*.

Edward Choi, Angeliki Lazaridou, and Nando de Freitas. 2018. Compositional obverter communication learning from raw visual input. *ICLR*.

Morten H Christiansen and Simon Kirby. 2003. Language evolution: Consensus and controversies. *Trends in Cognitive Sciences*, 7(7):300–307.

David Chrystal. 1987. The cambridge encyclopedia of language. *Cambridge ua*.

Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. 2018. Embodied question answering. In *CVPR*.

Katrina Evtimova, Andrew Drozdov, Douwe Kiela, and Kyunghyun Cho. 2018. Emergent communication in a multi-modal, multi-step referential game. In *ICLR*.

Willem Fase, Koen Jaspaert, and Sjaak Kroon. 1992. *Maintenance and loss of minority languages*. John Benjamins Publishing.

Joshua A Fishman. 1964. Language maintenance and language shift as a field of inquiry. a definition of the field and suggestions for its further development. *Linguistics*, 2(9):32–70.

F. Flores and T. Winograd. 1987. *Understanding Computers and Cognition: A New Foundation for Design*. Addison-Wesley Longman.

Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson. 2016. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2137–2145.

Paul Grouchy, Gabriele MT DEleuterio, Morten H Christiansen, and Hod Lipson. 2016. On the evolutionary origin of symbolic communication. *Scientific Reports*, 6:34615.

Serhii Havrylov and Ivan Titov. 2017. Emergence of language with multi-agent games: learning to communicate with sequences of symbols. In *Advances in Neural Information Processing Systems*, pages 2149–2159.

Emilio Jorge, Mikael Kågebäck, Fredrik D Johansson, and Emil Gustavsson. 2016. Learning to play guess who? and inventing a grounded language as a consequence. *Deep Reinforcement Learning Workshop at Advances in Neural Information Processing Systems*.

Brian D. Joseph. 2017. Historical linguistics. In Mark Aronoff, editor, *The handbook of linguistics*, chapter 15. John Wiley & Sons.

Simon Kirby. 2002. Natural language from artificial life. *Artificial Life*, 8(2):185–215.

Simon Kirby, Hannah Cornish, and Kenny Smith. 2008. Cumulative cultural evolution in the laboratory: An experimental approach to the origins of structure in human language. *Proceedings of the National Academy of Sciences*, 105(31):10681–10686.

Simon Kirby and James R. Hurford. 2002. The emergence of linguistic structure: An overview of the iterated learning model. In Angelo Cangelosi and Domenico Parisi, editors, *Simulating the Evolution of Language*, pages 121–147. Springer London, London.

David B Knoester, Philip K McKinley, Benjamin Beckmann, and Charles Ofria. 2007. Directed evolution of communication and cooperation in digital organisms. In *European Conference on Artificial Life*, pages 384–394. Springer.

Alexander Kuhnle and Ann Copestake. 2017. Shapeworld-a new test methodology for multimodal language understanding. *arXiv preprint arXiv:1704.04517*.

Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Pérolat, David Silver, and Thore Graepel. 2017. A unified game-theoretic approach to multiagent reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 4190–4203.

Angeliki Lazaridou, Karl Moritz Hermann, Karl Tuyls, and Stephen Clark. 2018. Emergence of linguistic communication from referential games with symbolic and pixel input. *ICLR*.

Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. 2016. Multi-agent cooperation and the emergence of (natural) language. *ICLR*.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature*, 521(7553):436.

Jason Lee, Kyunghyun Cho, Jason Weston, and Douwe Kiela. 2017. Emergent translation in multi-agent communication. *ICLR*.

David Lewis. 2008. *Convention: A philosophical study*. John Wiley & Sons.

Mike Lewis, Denis Yarats, Yann Dauphin, Devi Parikh, and Dhruv Batra. 2017. Deal or no deal? end-to-end learning of negotiation dialogues. In *EMNLP*, pages 2443–2453.

Laetitia Matignon, Guillaume J Laurent, and Nadine Le Fort-Piat. 2012. Independent reinforcement learners in cooperative markov games: a survey regarding coordination problems. *The Knowledge Engineering Review*, 27(1):1–31.

Yaron Matras. 2009. *Language contact*. Cambridge UP.

Carol Myers-Scotton. 2002. *Contact linguistics: Bilingual encounters and grammatical outcomes*. Oxford University Press on Demand.

Martin A. Nowak and David C. Krakauer. 1999. The evolution of language. *Proceedings of the National Academy of Sciences*, 96(14):8028–8033.

Michael Oliphant and John Batali. 1997. Learning and the emergence of coordinated communication. *Center for Research on Language Newsletter*, 11(1):1–46.

Mikael Parkvall. 2008. The simplicity of creoles in a cross-linguistic perspective. *Language Complexity: Typology, contact, change*, pages 265–285.

James R. Hurford. 1989. Biological evolution of the saussurean sign as a component of the language acquisition device. *Lingua*, 77:187–222.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *Nature*, 323(6088):533.

Richard J Senghas, Ann Senghas, and Jennie E Pyers. 2005. The emergence of nicaraguan sign language: Questions of development, acquisition, and evolution. *Biology and knowledge revisited: From neurogenesis to psychogenesis*, pages 287–306.

Brian Skyrms. 2010. *Signals: Evolution, learning, and information*. Oxford UP.

Vito Trianni and Marco Dorigo. 2006. Self-organisation and communication in groups of simulated and physical robots. *Biological Cybernetics*, 95(3):213–231.

Peter Trudgill. 1974. Linguistic change and diffusion: Description and explanation in sociolinguistic dialect geography. *Language in Society*, 3(2):215–246.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pages 5–32. Springer.

Terry Winograd. 1971. Procedures as a representation for data in a computer program for understanding natural language. Technical report, MIT.

# TalkDown: A Corpus for Condescension Detection in Context

**Zijian Wang**
Symbolic Systems Program
Stanford University
zijwang@stanford.edu

**Christopher Potts**
Department of Linguistics
Stanford University
cgpotts@stanford.edu

## Abstract

Condescending language use is caustic; it can bring dialogues to an end and bifurcate communities. Thus, systems for condescension detection could have a large positive impact. A challenge here is that condescension is often impossible to detect from isolated utterances, as it depends on the discourse and social context. To address this, we present TALKDOWN, a new labeled dataset of condescending linguistic acts in context. We show that extending a language-only model with representations of the discourse improves performance, and we motivate techniques for dealing with the low rates of condescension overall. We also use our model to estimate condescension rates in various online communities and relate these differences to differing community norms.

## 1 Introduction

Condescending language use can derail conversations and, over time, disrupt healthy communities. The caustic nature of this language traces in part to the ways that it keys into differing social roles and levels of power (Fournier et al., 2002). It is common for people to be condescending without realizing it (Wong et al., 2014), but a lack of intent only partly mitigates the damage it can cause. Thus, condescension detection is a potentially high-impact NLP task that could open the door to many applications and future research directions, including, for example, supporting productive interventions in online communities (Spertus, 1997), educating people who use condescending language in writing, helping linguists to understand the implicit linguistic acts associated with condescension, and helping social scientists to study the relationship between condescension and other variables like gender or socioeconomic status.

Progress on this task is currently limited by a lack of high-quality labeled data. A deeper challenge is that condescension is often impossible to detect from isolated utterances. First, a characteristic of condescending language is that it is not overtly negative or critical – it might even include (insincere) praise (Huckin, 2002). Second, condescension tends to rest on a pair of conflicting pragmatic presuppositions: a speaker presumption that the speaker has higher social status than the listener, and a listener presumption that this is incorrect. For example, an utterance that is entirely friendly if said by one friend to another might be perceived as highly condescending if said by a customer to a store clerk. In such cases, the social roles of the participants shape the language in particular ways to yield two very different outcomes.

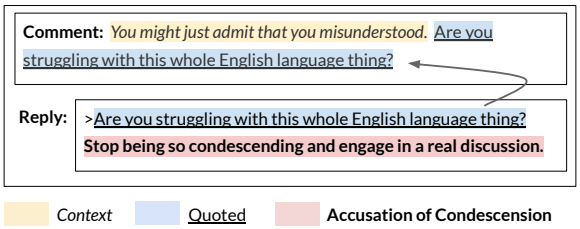


Figure 1: In this example, the REPLY quotes from part of the COMMENT and says that this QUOTED text is condescending.

In this paper, we seek to facilitate the development of models for condescension detection by introducing TALKDOWN, a new labeled dataset of condescending acts in context. The dataset is derived from Reddit, a thriving set of online communities that is diverse in content and tone. We focus on COMMENT and REPLY pairs of the sort given in Figure 1, in which the REPLY targets a specific quoted span (QUOTED) in the COMMENT as being condescending. The examples were multiply-labeled by crowdsourced workers, which ensures high-quality labels and allows us to include nu-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3711–3719,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

anced examples that require human judgment.

Our central hypothesis is that context is decisive for condescension detection. To test this, we evaluate models that seek to make this classification based only on the QUOTED span in the COMMENT as well as extensions of those models that include summary representations of the preceding linguistic context, which we treat as an approximation of the discourse context in which the condescension accusation was made. Models with contextual representations are far superior, bolstering the original hypothesis. In addition, we show that these models are robust to highly imbalanced testing scenarios that approximate the true low rate of condescension in the wider world. Such robustness to imbalanced data is an important prerequisite for deploying models like this. Finally, we apply our model to a wide range of subreddits, arguing that our estimated rates of condescension are related to different community norms.

## 2 The TALKDOWN Corpus

We chose Reddit as the basis for our corpus for a few key reasons. First, it is a large, publicly available dataset from an active set of more than one million user-created online communities (subreddits).[1] Second, it varies in both content and tone. Third, users can develop strong identities on the site, which could facilitate user-level modeling, but these identities are generally pseudonymous, which is useful when studying charged social phenomena (Hamilton et al., 2017; Wang and Jurgens, 2018). Fourth, the subreddit structure of the site creates opportunities to study the impact of condescension on community structure and norms (Buntain and Golbeck, 2014; Lin et al., 2017; Zhang et al., 2017; Chandrasekharan et al., 2018).

The basis for our work is the Reddit data dump 2006–2018.[2] We first extracted COMMENT/REPLY pairs in which the REPLY contains a condescension-related word. After further filtering out self-replies and moderator posts, and normalizing links and references, we obtain 2.62M COMMENT/REPLY pairs.

Not all of these examples truly involve the REPLY saying that the QUOTED span is condescending. Our simple pattern-based extraction method is not sufficiently sensitive. To address this, we conducted an annotation project on Amazon Mechanical Turk. Our own initial assessment of 200 examples using a five-point Likert scale revealed two things that informed this project.

| | Median | Mean | Std. | Max |
|---|---|---|---|---|
| QUOTED | 18 | 22.79 | 18.40 | 399 |
| REPLY | 67 | 100.92 | 112.34 | 1,921 |
| CONTEXT | 47 | 98.40 | 154.59 | 2,136 |

Table 1: Basic statistics of the length of the examples in the corpus. CONTEXT is everything in the COMMENT before the QUOTED span.

First, we saw a clear split in the positive instances of condescension. In some, specific linguistic acts are labeled as condescending ("This is really condesending"), whereas others involve general user-level accusations that are not tied to specific acts ("You're so condescending"). We chose to focus on the specific linguistic acts. They provide a more objective basis for annotation, and they can presumably be aggregated to provide an empirically grounded picture of user-level behavior (or others' reactions to such behavior). Thus, for positive instances of condescension, we further limited our attention to COMMENT/REPLY pairs in which the REPLY contains a direct quotation from the COMMENT, using fuzzy match based on Levenshtein distance (Navarro, 2001), as illustrated in Figure 1. We extracted 66K such examples. Some statistics on these examples is given in Table 1.

Second, with the above ambiguity addressed, the signal of condescending or not is mostly clear. Thus, we designed the annotation project around a three-way multiple choice question: *condescending*, *not condescending*, and *cannot decide*. Each task began with instructions and two training questions, following by 10 different COMMENT/REPLY pairs to be labeled. Appendix A.2 provides screenshots of the annotation interface.

To process the annotations, we filtered out the work of annotators who did not correctly answer the training questions. The remaining annotators have moderate to substantial agreement (Fleiss $\kappa = 0.593$; Fleiss 1971; Landis and Koch 1977). We then used Expectation–Maximization, as in Dempster et al. 1977, to assign labels. This yields slightly better quality in our hand-inspection than labels by majority vote, presumably because it factors individual worker reliability into the decision making. In the end, we obtained 4,992 valid labeled instances: 65.2% labeled as *conde-*

[1] http://redditmetrics.com/history
[2] https://files.pushshift.io/reddit

| | Positive | Negative |
|---|---|---|
| Balanced (1:1) | 3,255 | 3,255 |
| Imbalanced (1:20) | 3,255 | 65,100 |

Table 2: Basic statistics of our dataset.

*scending* (henceforth, *positive*), and 34.8% as *non-condescending* (henceforth, *negative*).[3]

To fully balance the dataset, we pulled out one random month's data for each year in 2011 to 2017. We extracted instances using the same methods as described above, but we filtered out COMMENT/REPLY pairs in which a condescension-related word appeared. Our final dataset thus consists of annotated positive and negative instances, with supplemental randomly-sampled negative instances. For our experiments, we partitioned the data into 80% train, 10% development, and 10% test splits. In addition, to simulate real-world situations, we built a dataset with a 1:20 ratio of positive to negative instances.[4] The basic statistics of the dataset are shown in Table 2.

## 3 Experiments

We now establish some baselines for the TALKDOWN Corpus and begin to test the hypothesis that contextual representations are valuable for this task. To do this, we use the BERT model of Devlin et al. (2019), which uses a Transformer-based encoder architecture (Vaswani et al., 2017) to learn word representations by training against a masked language modeling task and a next-sentence prediction task. Our models are initialized with the pretrained representations released by the BERT team and a fully connected layer on the top (Figure 4 in Devlin et al. 2019), which is then fine-tuned to our dataset (Peters et al., 2019). We explore both BERT Base ($BERT_B$) and BERT Large ($BERT_L$),[5] to determine whether the added expense of using $BERT_L$ is justified. Appendix B provides details on our process of hyperparameter tuning and optimization.

[3]There was just one case where *cannot decide* was the chosen label; it was in Spanish, so we excluded it and added a language classification step to our preprocessing pipeline.

[4]To the best of our knowledge, there is no prior work on what percentage of conversations on Reddit (or, more broadly, in daily conversations) are condescending. Thus, we chose the ratio based on informal observations on Reddit.

[5]The whole-word masking model was used as it performs better than the original one in multiple benchmarks.

| Input 1 | Input 2 | Model | Imb. F1 | Bal. F1 |
|---|---|---|---|---|
| QUOTED ∧ CONTEXT | | ∧ $BERT_L$ | **0.684** | **0.654** |
| QUOTED ∧ CONTEXT | | ∧ $BERT_B$ | 0.657 | 0.596 |
| QUOTED | | ∧ $BERT_L$ | 0.650 | 0.640 |
| | CONTEXT | ∧ $BERT_L$ | 0.611 | 0.513 |
| | | *random* | 0.371 | 0.500 |
| | | *majority* | 0.488 | 0.333 |

Table 3: Performance (macro-F1) for predicting condescension on balanced and imbalanced versions of TALKDOWN. Model selection was done according to the procedure described in Appendix B.

| | QUOTED + CONTEXT | | QUOTED | |
|---|---|---|---|---|
| Ratio | Imb. F1 | Bal. F1 | Imb. F1 | Bal. F1 |
| 1:1 | 0.542 | **0.708** | 0.554 | **0.690** |
| 1:20 | 0.670 | 0.574 | 0.620 | 0.518 |
| 2:20 | 0.682 | 0.619 | 0.640 | 0.554 |
| 3:20 | **0.684** | 0.654 | 0.646 | 0.585 |
| 4:20 | 0.678 | 0.632 | **0.650** | 0.640 |
| 5:20 | 0.668 | 0.626 | 0.645 | 0.582 |
| 10:20 | 0.672 | 0.656 | 0.641 | 0.640 |
| 15:20 | 0.665 | 0.641 | 0.641 | 0.593 |
| 20:20 | 0.674 | 0.621 | 0.645 | 0.597 |

Table 4: The impact of different train-set positive:negative ratios. All the models are $BERT_L$. The first row is based on the balanced dataset, and the rest are based on the imbalanced dataset with different oversampling ratios. Model selection again used the procedure in Appendix B.

### 3.1 Predicting Condescension

Table 3 summarizes the results of our core experiments. Input 1 and Input 2 describe the basis for the feature representations. Thus, for example, QUOTED∧CONTEXT is a model that uses both the quoted span and the preceding linguistic context. We report two testing scenarios: *Balanced* and *Imbalanced*, in which there are 20 negative examples for each positive example.

The results clearly support our hypothesis that context matters; using the QUOTED part and CONTEXT together give us 3–4% boost in macro-F1 using the same model architecture. In addition, we see that increasing the capacity of the model also helps, though more modestly. It's noteworthy that the performance of using the QUOTED part is better than that of using CONTEXT alone, though the QUOTED part is roughly three times shorter. Thus, there is a strong signal in the QUOTED part – the replier chose this span for a reason – but the context contains a signal as well.

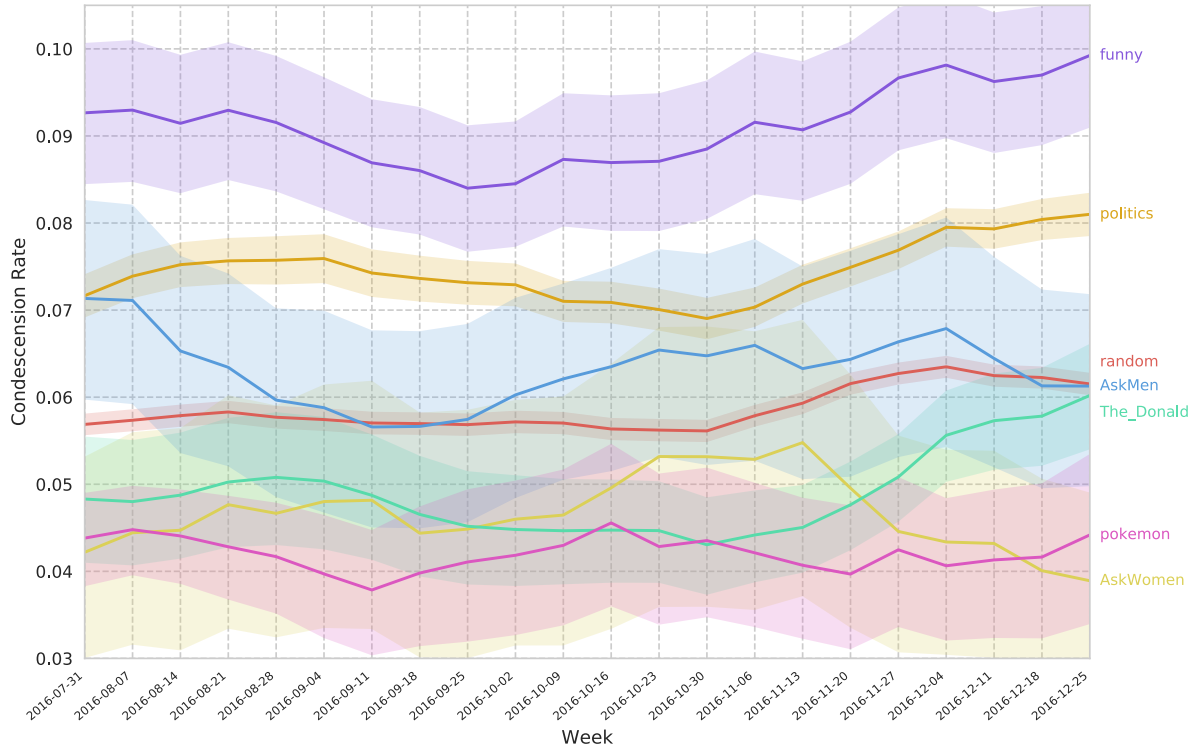


Figure 2: Condescension rates over time in a selection of subreddits, as predicted by our best model. The time window is centered around the 2016 U.S. Presidential Elections. We calculate a rolling mean with a window size of 5 and use July 2016 only for smoothing. We obtain 95% confidence intervals via bootstrapping.

### 3.2 Imbalanced Testing Scenarios

Imbalanced testing scenarios are more challenging, but they also better reflect usage rates of condescending language in public forums like Reddit. To further understand how best to get traction on this problem, we explored a range of different methods for creating training data. Our results are summarized in Table 4. As expected, the balanced problem is best addressed with a balanced dataset. For the imbalanced problem, we found that an oversampling ratio of 2 to 4 yielded the best performance. Our full QUOTED ∧ CONTEXT model is again clearly superior in these scenarios.

### 3.3 Condescension Rates Across Subreddits

Our hope for TALKDOWN is that it will play a role in developing systems that can help identify condescending acts on social media. This will depend on models trained on TALKDOWN being able to get an accurate read on condescension at scale. As a first step towards assessing this capability, we ran our models on 14 subreddits, over the time period of July 2016 to December 2016, which covers the 2016 U.S. Presidential Election, an event that we expect to influence condescension rates in various ways across Reddit. Appendix C lists these subreddits along with their post counts and estimated average rates of condescension. Figure 2 highlights a selection of them.

As a baseline, we include a 10% random sample from the top 100 most active subreddits.[6] Consistently above this baseline are 'politics' and 'funny'. It makes sense that an overtly political subreddit would show a high rate of condescension (as do 'news' and 'worldnews'; Appendix C): it's a contentious topic in a contentious time period; see also the rising rate for 'The_Donald' in the post-election period. It is more surprising that 'funny' shows the highest rates. We do not have a deep understanding of why this is, but it could trace to our model confusing irony and sarcasm with condescension.

Below the baseline are 'AskWomen' and 'pokemon'. We expect 'pokemon' to have low rates of condescension, as it strikes us as a supportive community. However, one might be surprised to see 'AskWomen' so low, especially as compared with 'AskMen', which has high rates in general.

[6]This is derived from the 'subscribers' section in http://redditlist.com/all, excluding 'announcements'.

There is wide support for the idea that women experience more condescension than men do (Hall and Braunwald, 1981; Harris, 1993; McKechnie et al., 1998; Cortina et al., 2002; Trix and Psenka, 2003), as reflected in the recent lexical innovation *mansplaining*, which can be roughly paraphrased as 'a man condescending to a woman'.[7] However, community norms on 'AskWomen' and 'AskMen' are likely shaping these outcomes. Whereas the description for 'AskWomen' says it is "curated to promote respectful and on-topic discussions, and not serve as a debate subreddit", the description for 'AskMen' ends with "And don't be an asshole. Also, go away."

## 4 Conclusion

We introduced TALKDOWN, a new annotated Reddit corpus of condescending linguistic acts in context. Using BERT, we established baseline models that suggest this is a challenging task, and one that benefits from rich contextual representations. Finally, in qualitative analyses on diverse subreddits, we offered initial evidence that models trained on TALKDOWN generalize to new data, a prerequisite for using them to help improve online communities via condescension detection. The full dataset with the pretrained BERT model is available at `http://github.com/zijwang/talkdown`.


## Acknowledgements

We would like to thank David Jurgens for his help with the Reddit dataset, Yulia Tsvetkov and Dan Jurafsky for their insightful early-stage suggestions, Lucy Li for detailed comments on an early draft of the paper, and our anonymous EMNLP–IJCNLP reviewers for their helpful feedback. We would also like to thank our Mechanical Turk contributors for their help in annotating the TALKDOWN Corpus. This research is based in part upon work supported by a Google Faculty Research Award and by the NSF under Grant No. BCS-1456077.


[7] https://en.wikipedia.org/wiki/Mansplaining

## References


Cody Buntain and Jennifer Golbeck. 2014. Identifying social roles in Reddit using network structure. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 615–620, New York, NY, USA. ACM.

Eshwar Chandrasekharan, Mattia Samory, Shagun Jhaver, Hunter Charvat, Amy Bruckman, Cliff Lampe, Jacob Eisenstein, and Eric Gilbert. 2018. The internet's hidden rules: An empirical study of Reddit norm violations at micro, meso, and macro scales. *Proceedings of the ACM on Human-Computer Interaction*, 2(CSCW):32:1–32:25.

Lilia M. Cortina, Kimberly A. Lonsway, Vicki J. Magley, Leslie V. Freeman, Linda L. Collinsworth, Mary Hunter, and Louise F. Fitzgerald. 2002. What's gender got to do with it? Incivility in the federal courts. *Law & Social Inquiry*, 27(2):235–270.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Joseph L. Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378.

Marc A. Fournier, D.S. Moskowitz, and David C. Zuroff. 2002. Social rank strategies in hierarchical relationships. *Journal of Personality and Social Psychology*, 83(2):425.

Judith A. Hall and Karen G. Braunwald. 1981. Gender cues in conversations. *Journal of Personality and Social Psychology*, 40(1):99.

William L. Hamilton, Justine Zhang, Cristian Danescu-Niculescu-Mizil, Dan Jurafsky, and Jure Leskovec. 2017. Loyalty in online communities. In *Eleventh International AAAI Conference on Web and Social Media*.

Mary B. Harris. 1993. How provoking! What makes men and women angry? *Aggressive Behavior*, 19(3):199–211.

Thomas Huckin. 2002. Critical discourse analysis and the discourse of condescension. *Discourse Studies in Composition*, 155:176.

J. Richard Landis and Gary G. Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*, pages 159–174.

Zhiyuan Lin, Niloufar Salehi, Bowen Yao, Yiqi Chen, and Michael S. Bernstein. 2017. Better when it was smaller? community content and behavior after massive growth. In *Eleventh International AAAI Conference on Web and Social Media*.

Sally A. McKechnie, Christine T. Ennew, and Lauren H. Read. 1998. The nature of the banking relationship: A comparison of the experiences of male and female small business ownersi. *International Small Business Journal*, 16(3):39–55.

Gonzalo Navarro. 2001. A guided tour to approximate string matching. *ACM computing surveys (CSUR)*, 33(1):31–88.

Matthew Peters, Sebastian Ruder, and Noah A. Smith. 2019. To tune or not to tune? Adapting pretrained representations to diverse tasks. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 7–14.

Ellen Spertus. 1997. Smokey: Automatic recognition of hostile messages. In *Innovative Applications of Artificial Intelligence*, pages 1058–1065.

Frances Trix and Carolyn Psenka. 2003. Exploring the color of glass: Letters of recommendation for female and male medical faculty. *Discourse & Society*, 14(2):191–220.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008.

Zijian Wang and David Jurgens. 2018. It's going to be okay: Measuring access to support in online communities. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 33–45, Brussels, Belgium. Association for Computational Linguistics.

Gloria Wong, Annie O. Derthick, E.J.R. David, Anne Saw, and Sumie Okazaki. 2014. The what, the why, and the how: A review of racial microaggressions research in psychology. *Race and Social Problems*, 6(2):181–200.

Justine Zhang, William L. Hamilton, Cristian Danescu-Niculescu-Mizil, Dan Jurafsky, and Jure Leskovec. 2017. Community identity and user engagement in a multi-community landscape. In *Eleventh International AAAI Conference on Web and Social Media*.

# A Data

## A.1 In-house Annotation Analysis

Figure 3 shows the five-point Likert scale annotations between two in-house annotators. It can be seen that the signal of condescending or not is clear, and the agreement level between the two annotations is substantial: the Fleiss' $\kappa$ is $0.613$ for the five-point scale and $0.732$ when normalized to three-point scale used in the paper (Fleiss, 1971; Landis and Koch, 1977).

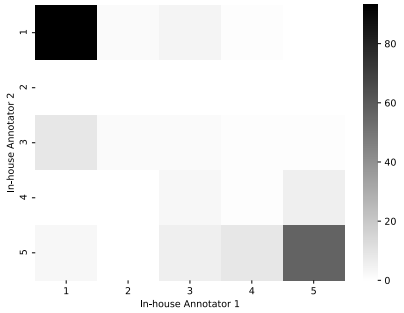


Figure 3: Heatmap for in-house initial assessment with a five-point Likert scale.

## A.2 Annotation Interface

In this section, we show examples of the annotation interface we used on Amazon Mechanical Turk: Figure 4 and Figure 5.

Annotators were presented with the task name, the instructions, and two simple training questions, followed by a warning in red saying they needed to pass the training questions to proceed (Figure 4). They had unlimited trials for the training questions, and explanations (for both correct and incorrect answers) were presented directly after each trial. This helped the annotators learn how to approach the task.

After they passed the training questions, they were prompted that they could start to do the test questions (Figure 5). The interface of the test questions was similar to that of the training questions, but without explanations after selections. We explicitly checked that the annotators had made selections on each test question before submission, while this was not forced for training questions. This was to filter out possibly low-quality annotations, where the annotators did not pay attention to the instructions.

## B Model Hyperparameters

Our BERT models were trained using a set of hyperparameters based on the recommendations in Devlin et al. 2019. Specifically, we set:

- Model Architecture:
  - $BERT_B$: Bert Base, Cased
  - $BERT_L$: Bert Large, Cased, with whole-word masking
- Learning rate: $\{0.5, 0.8, 1, 2, 3, 5\} \cdot 10^{-5}$
- Epoch: 2, 3
- Batch size: 32
- Max sequence length: 512

When optimizing these models, we set the batch size to 32 in order to ensure there was at least one positive instance per mini-batch. Grid search was performed with different learning rates and oversampling ratios, and best models were selected based on the best performance on the development set under the imbalanced setting. We found that oversampling 2 to 4 times the positive class (i.e., 10%–20% of the number of instances in the negative class) generally yielded good performance in all the experiments we ran. For all experiments, we used the HuggingFace PyTorch implementation of BERT.[8]

## C Subreddit Condescension Rates

Table 5 shows basic statistics for all the subreddits we analyzed.

| Subreddit | #Pairs | Mean rate | Std. err |
|---|---|---|---|
| 4chan | 6,099 | 0.072 | 0.189 |
| AskMen | 19,406 | 0.064 | 0.164 |
| AskReddit | 256,181 | 0.049 | 0.147 |
| AskWomen | 9,021 | 0.046 | 0.129 |
| The_Donald | 57,429 | 0.051 | 0.149 |
| aww | 8,727 | 0.059 | 0.168 |
| funny | 71,875 | 0.092 | 0.214 |
| gaming | 64,881 | 0.064 | 0.175 |
| news | 396,710 | 0.066 | 0.179 |
| pokemon | 38,353 | 0.043 | 0.142 |
| politics | 583,033 | 0.075 | 0.186 |
| stopdrinking | 910 | 0.038 | 0.112 |
| tifu | 35,443 | 0.068 | 0.177 |
| worldnews | 164,302 | 0.064 | 0.176 |
| *random* | 1,700,192 | 0.059 | 0.166 |

Table 5: Subreddit experiment statistics. The raw data are from the Reddit dump from July 2016 to December 2016. 'Pairs' are COMMENT/REPLY pairs as defined in the paper. 'Mean rate' is the mean rate of condescension as estimated by our best model, and 'Std. err' gives the associated standard error. '*random*' is a 10% random sample from the top 100 active subreddits over the same time period.

[8] https://github.com/huggingface/pytorch-transformers/

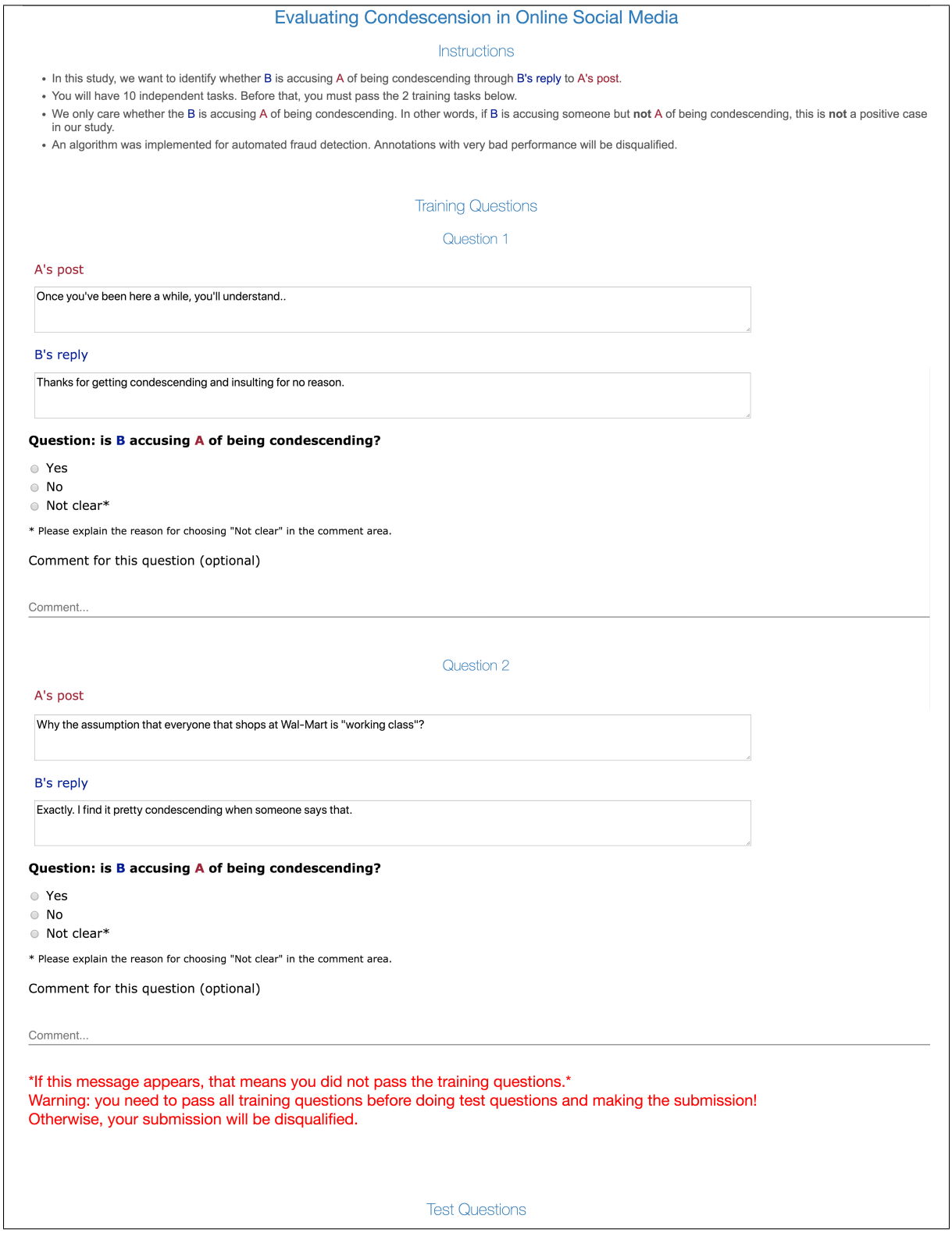


Figure 4: The initial view of instructions and training questions in the annotation interface.

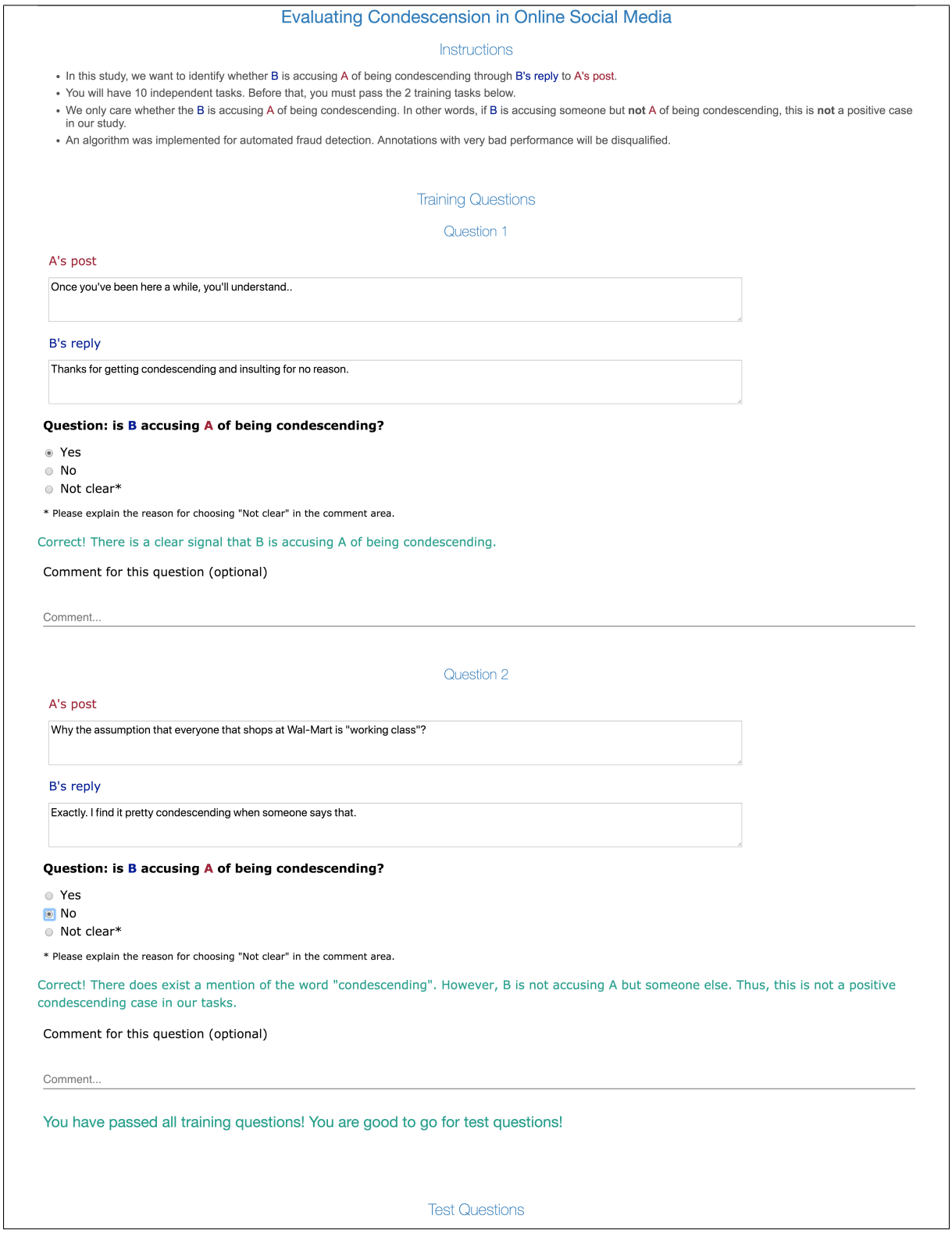


Figure 5: The view after the annotator passed the training questions.

# Summary Cloze: A New Task for Content Selection in Topic-Focused Summarization

**Daniel Deutsch and Dan Roth**
Department of Computer and Information Science
University of Pennsylvania
{ddeutsch,danroth}@seas.upenn.edu

## Abstract

A key challenge in topic-focused summarization is determining what information should be included in the summary, a problem known as content selection. In this work, we propose a new method for studying content selection in topic-focused summarization called the *summary cloze* task. The goal of the summary cloze task is to generate the next sentence of a summary conditioned on the beginning of the summary, a topic, and a reference document(s). The main challenge is deciding what information in the references is relevant to the topic and partial summary and should be included in the summary. Although the cloze task does not address all aspects of the traditional summarization problem, the more narrow scope of the task allows us to collect a large-scale datset of nearly 500k summary cloze instances from Wikipedia. We report experimental results on this new dataset using various extractive models and a two-step abstractive model that first extractively selects a small number of sentences and then abstractively summarizes them. Our results show that the topic and partial summary help the models identify relevant content, but the task remains a significant challenge.

## 1 Introduction

Topic-focused multi-document summarization (MDS) has long been a goal for natural language processing (Dang, 2005, 2006). In contrast to generic summarization, topic-focused summarization systems attempt to summarize a set of reference documents with respect to a specific topic or information need.

Recent research on summarization has mostly focused on generic summarization, in part due to the size of the available datasets. Work on topic-focused summarization (Lin and Bilmes, 2011; Ma et al., 2016; Feigenblat et al., 2017) is largely based on the DUC 2005 and 2006 datasets (Dang, 2005, 2006), which are orders of magnitude smaller than comparable generic summarization datasets (Nallapati et al., 2016; Grusky et al., 2018). Because the available corpora are so small, it is difficult to use them to train recent state-of-the-art summarization systems, which require large amounts of training data.

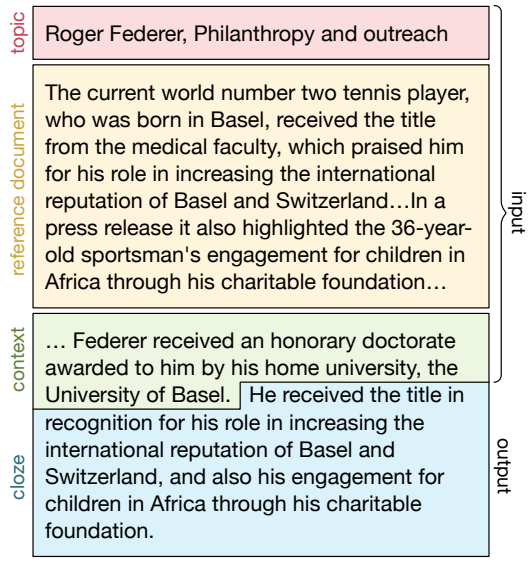


Figure 1: Given a topic, reference document, and a partial summary (the context), the objective of the summary cloze task is to predict the next sentence of the summary, known as the cloze.

Instead of focusing on the full topic-focused MDS problem, we address one aspect of the task known as *content selection*, the problem of deciding what information should be included in the summary (Nenkova, 2006b). Narrowing the scope of the problem makes it easier to collect a large-scale dataset that is tailored to the specific task and thus build upon recent work on summarization.

To that end, we formulate a new task for content selection in topic-focused summarization which



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3720–3729,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

we call the *summary cloze* task (§2). Given a topic, a partial summary, and a reference document(s), the goal of the summary cloze task is to produce the next sentence in the summary (referred to as the *cloze*). The underlying assumption of the task is that the content of the cloze is known to come from the reference document, and thus the primary challenge of the task is to decide what information in the reference document(s) is relevant to the topic and partial summary. An example instance of the summary cloze task is presented in Figure 1.

Then, we collected a new large-scale summary cloze dataset from Wikipedia, called the WIKICITE dataset (§3). Each paragraph in Wikipedia can be viewed as a topic-focused summary of the references cited within the paragraph, where the topic is defined as the article title and section headings. The citations provide supervision at the sentence-level that indicates where the content of the preceding sentence came from. We scraped hundreds of thousands of Wikipedia articles and corresponding references to collect nearly 500k summary cloze instances.[1]

With a new task definition and corresponding large-scale dataset, we then propose an extractive model and a two-step abstractive model, both of which are based on recent successful work on generic summarization (§4). The extractive model combines representations of the topic and partial summary with representations of the document sentences through an attention mechanism to extract one reference sentence. The two-step model first reduces the length of the input data by extractively selecting a small number of sentences, then abstractively summarizes them using a decoder that has an initial hidden state which depends on the partial summary. Our experimental results (§6) show that the topic and context help the models identify relevant information, but the task remains a significant challenge.

The contributions of this work are three-fold: (1) We formalize the summary cloze task as a method to develop content selection models for topic-focused summarization; (2) we release the WIKICITE dataset, which contains nearly 500k summary cloze instances collected from Wikipedia; and (3) we propose sensible baseline extractive and abstractive models for content selection in topic-focused summarization.

[1]Available for download at http://cogcomp.org/page/publication_view/880

## 2 The Summary Cloze Task

One of the main challenges for a topic-focused MDS system is selecting what content from the reference documents to include in the summary. Intuitively, this decision largely depends on the topic itself and the content of the summary generated thus far. Therefore, being able to predict the next sentence of a summary is an important component of a full topic-focused MDS system and a worthwhile task on its own. With this motivation in mind, we now formally define the summary cloze task.

**Task Definition** Given a partial summary (the *context*) and a reference document or documents (the *references*), the goal of the summary cloze task is to generate the next sentence of the summary (the cloze). It is assumed that the information in the cloze comes from the provided reference documents. This is the most general definition of the summary cloze task. However, because we are interested in topic-focused summarization, in this work, it is also assumed that a high-level topic is provided as input, such as in Figure 1.

One advantage that the summary cloze task has over the traditional topic-focused MDS problem is the availability of data. Topic-focused MDS requires collecting example summaries of a set of documents about a particular topic. This data is hard to collect from human annotators and is difficult to find occurring naturally. The cloze task requires sentence-level supervision for which we were able to collect a large-scale dataset using Wikipedia (§3). By conditioning the cloze generation on a partial summary and working at the sentence-level, we are able to get around the problem of data scarcity.

**Evaluation** Evaluating summary cloze models is challenging; The difficulties of evaluating summarization systems (Nenkova, 2006a) still persist for the summary cloze task. We propose for the main evaluation metric to be ROUGE (Lin, 2004), which has been shown to be an effective metric for summarization (Owczarzak et al., 2012). Additionally, for abstractive models we use perplexity to quantify how likely the ground-truth cloze is according to the model.

Because the cloze is only one sentence long, the chance of the ground-truth and model clozes being equally valid but very different to each other is likely higher than for other summarization tasks.

It is desirable to have a good automatic evaluation metric that could capture the coherence of the model's cloze with the context in such situations. However, developing such an automatic metric for measuring summary coherence that correlates well with human judgments is an open problem for summarization in general, and thus we leave it for future work.

## 3 The WikiCite Dataset

In order to build content selection models, we collected a large-scale dataset of summary cloze instances using Wikipedia.

**Wikipedia as a Summary** Each paragraph in a Wikipedia article can be viewed as a topic-focused multi-document summary. Many Wikipedia articles have citations to external sources that provide supporting evidence for the information within the article. The articles' paragraphs can be viewed as summarizing the documents cited within the paragraph with respect to the page entity and section headings, which together form the topic of the summary. Since the citations are at the sentence-level, they provide supervision that identifies where the information in the preceding sentence came from, and thus can be used to generate summary cloze instances.

For example, consider a paragraph from the "Family and personal life" section of Barack Obama's Wikipedia page, presented in Figure 2. Nearly all of the sentences in the paragraph are supported with citations, and these citations contain the same information written in the paragraph. Each sentence with a citation can be used to produce a summary cloze instance where the preceding sentences are the context and the topic is "Barack Obama, Family and personal life." Although not all paragraphs on Wikipedia are as well-cited as the example in Figure 2, only one citation is necessary to create a cloze instance.

**Dataset Collection** To collect a summary cloze dataset, we first parsed the articles and citations from the January 2019 Wikipedia dump. Only articles marked with the category "Living people" were kept because these articles are more likely to have reference documents which can be scraped automatically. The reference document HTMLs were collected from 12 months of Common Crawl data.[2] Then, the text of each document was extracted from the HTML using unfluff.[3]

| |
|---|
| In June 1989, Obama met Michelle Robinson when he was employed as a summer associate at the Chicago law firm of Sidley Austin.[62] Robinson was assigned for three months as Obama's adviser at the firm, and she joined him at several group social functions but declined his initial requests to date.[63] They began dating later that summer, became engaged in 1991, and were married on October 3, 1992.[64] The couple's first daughter, Malia Ann, was born in 1998,[65] followed by a second daughter, Natasha ("Sasha"), in 2001.[66] The Obama daughters attended the University of Chicago Laboratory Schools. When they moved to Washington, D.C., in January 2009, the girls started at the Sidwell Friends School.[67] The Obamas have two Portuguese Water Dogs; the first, a male named Bo, was a gift from Senator Ted Kennedy.[68] In 2013, Bo was joined by Sunny, a female.[69] |
| 62: [Michelle] Obama was assigned to mentor Barack, who was a summer intern at Sidley & Austin… |
| 67: … Barack Obama and his wife have chosen Sidwell Friends School for their two daughters |
| 68: … the Obamas have chosen a 6-month-old Portuguese water dog — a gift from Senator Edward M. Kennedy… The girls have named the dog Bo… |

Figure 2: A paragraph from the "Family and personal life" section of Barack Obama's Wikipedia page and selected excerpts from the cited documents which provide supporting evidence.

In order to filter low-quality instances (e.g., HTML was a 404 page, the body text was incorrectly extracted, etc.), we labeled around 500 instances based on whether or not the reference document provided supporting evidence for the corresponding cloze. We trained a simple linear classifier with length and BM25 features that achieved an F1 score of 85 to filter the data. The remaining data was split between training, 10k validation, and 10k testing instances, ensuring that no reference document or Wikipedia article appears in more than one set.

The dataset statistics compared to those of the DUC 2005 and 2006 datasets are presented in Table 1. Other than the number of instances, the biggest differences between the two datasets are the length of the reference documents and topics. The WIKICITE documents are significantly longer, over double the length DUC documents at one standard deviation above the mean. The DUC topics, which are generally text that describe an information need, are much longer than the WIKICITE topics, which are high-level topic keywords.

Examples from the dataset are provided in Appendix A.

[2] http://commoncrawl.org/

[3] https://github.com/ageitgey/node-unfluff

| Statistic | WikiCite | DUC 2005 + DUC 2006 |
|---|---|---|
| #Instances | 486,015 | 100 |
| #Document Tokens | 1083.2 (2105.0) | 723.3 (629.0) |
| #Document Sents | 51.4 (137.1) | 30.7 (30.1) |
| #Topic Tokens | 5.7 (3.4) | 29.5 (11.3) |
| #Context Tokens | 65.5 (56.1) | - |
| #Context Sents | 2.8 (2.3) | - |
| #Cloze/Summ. Tokens | 24.6 (12.0) | 277.8 (19.9) |
| #Cloze/Summ. Sents | 1 | 13.5 (3.4) |

Table 1: The WIKICITE and DUC 2005 and 2006 dataset statistics. The average values are reported with standard deviations in parentheses.

## 4 Content Selection Models

We propose two sensible extensions to successful single-document summarization systems for the summary cloze task, an extractive model and a two-step abstractive model.

### 4.1 Extractive Model

The extractive model is based on those presented in Kedzie et al. (2018). The base model works as follows. For every sentence $\mathbf{x}_i$ in the reference document, a sentence-level representation $\mathbf{h}_i$ is created by a sentence encoder (e.g., by averaging word vectors, by using a recurrent neural network, etc.). Then, a sentence extractor creates a document-level representation for each sentence $\mathbf{d}_i$, for example, by running a second RNN over the sentence-level representations. The document-level representation is passed through a multilayer perceptron and a sigmoid unit to calculate the probability of extracting a particular sentence under the model.

**Adding Topics and Context** We extended the standard extractive model by incorporating the topic and context information through an attention mechanism. First, each of the individual topics is encoded into a vector representation $\mathbf{t}_j$ by averaging the word vectors for each word in topic $j$. Then, a bidirectional RNN encodes the context tokens into a vector representation $\mathbf{c}_k$ for each token.

The topic and context token representations are combined with the sentence-level sentence representations through an attention mechanism. Specifically, an attention score is computed between each $\mathbf{h}_i$ and the topic and context representations. Then, the subsequent attention scores are used to create a weighted sum of the topic and context representations, $\mathbf{a}_i$.

Finally, the sentence representation $\tilde{\mathbf{h}}_i$ that includes the attention mechanism is computed as

$$\tilde{\mathbf{h}}_i = \mathbf{W}_a[\mathbf{h}_i; \mathbf{a}_i] \quad (1)$$

where $\mathbf{W}_a$ is a matrix of learned parameters.

In this extended version of the model, $\tilde{\mathbf{h}}_i$ is used to compute the extraction probabilities instead of $\mathbf{h}_i$. A graphical representation of the full extractive model is presented in Figure 3.

**Training & Inference** To convert the abstractive clozes into extractive reference sentence labels, we follow the procedure of Nallapati et al. (2017) and greedily select reference sentences that maximize ROUGE-1 recall until no more sentences improve the score. The training objective is the weighted negative log-likelihood of the labels with the same weighting as Kedzie et al. (2018). At inference, the sentence with the highest probability of extraction is selected.

### 4.2 Two-Step Abstractive Model

Since the reference documents in the WIKICITE dataset are rather long (see Table 1), we chose to use a two-step approach to build an abstractive system for the summary cloze task.

**Extractive Step** The first step uses the same extractive model from the previous subsection to significantly reduce the amount of input text. Instead of selecting just one sentence at inference, the extractive model repeatedly selects sentences until a pre-specified number of words has been met. Then, only the sentences which were extracted are passed as input to the abstractive step.

**Abstractive Step** For the abstractive step, we extended the Pointer-Generator + Coverage network (See et al., 2017) for single-document summarization to include the context.

The Pointer-Generator network is built on a sequence-to-sequence model with attention. The reference document is encoded using an RNN, and the summary is produced using a second RNN. The model is augmented with a copy mechanism by including a soft switch that allows the attention distribution to influence the decoder's probability distribution over the vocabulary, thus making it easier to copy words from the input.

Then, the coverage mechanism discourages the attention weights from repeatedly assigning high values to the same input tokens across decoding time steps. This is accomplished by adding a new

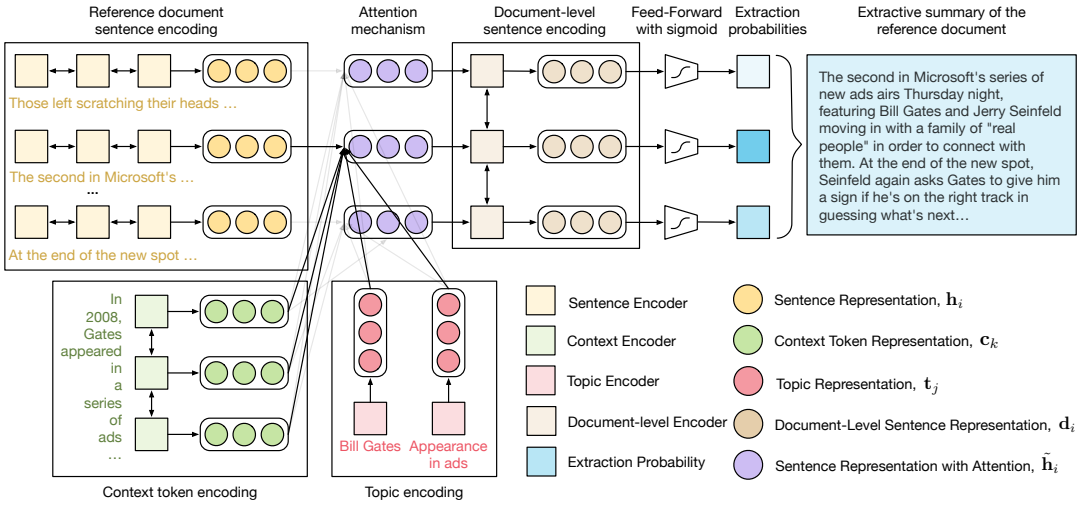


Figure 3: The extractive model uses three separate encoders create representations for the reference document sentences, context tokens, and topics. These are combined through an attention mechanism, encoded at a document-level, and passed through a feed-forward layer to compute an extraction probability for each reference sentence.

term to the loss function which penalizes such behavior and a new term to the attention score calculation that informs the model about the previous steps' attention weights. The coverage mechanism has the effect of decreasing the amount of redundancy in the generated summary.

**Conditioning on Context** In order to condition the generation of the cloze on the context, we use the context tokens to initialize the decoder's hidden state representation by first forcing the tokens through the decoder.

Concretely, let $\mathbf{s}_t$ be the decoder's hidden state for decoding step $t$ (with $\mathbf{s}_0$ initialized to the final encoder hidden state), $d_\theta(\cdot)$ be the decoder function, and $c_1, \ldots, c_T$ be the context tokens. The context tokens are passed through the decoder to create a new decoder hidden state representation using the following recursive formula

$$\mathbf{s}_t = d_\theta(c_t, \mathbf{s}_{t-1}) \qquad \forall t = 1, \ldots, T \qquad (2)$$

Then, $\mathbf{s}_T$ is used as the initial hidden state that the decoder uses to generate the cloze tokens.

Priming the decoder with the context tokens has the benefit of allowing the model to condition the cloze generation on a representation of the context. Additionally, because the context will impact the decoder's coverage calculation, it will discourage the cloze from repeating information already in the context.

**Training & Inference** Following See et al. (2017), the training objective is the negative log-likelihood of the ground-truth cloze tokens with the added coverage penalty. Inference is done using beam search.

## 5 Experimental Setup

Our experiments aim to better understand the WIKICITE dataset, evaluation metric, and performance of the extractive and abstractive models. Numbers in bold indicate statistical significance based on bootstrap resampling with $p = 0.05$.

### 5.1 Dataset Ablation & Human Performance

In order to better understand the WIKICITE dataset, we estimate how well models can perform without the reference documents and context and establish human performance on the task.

**No Reference Document** We used the publicly available large-scale language model from Radford et al. (2019) (the 345M parameter version) with the default parameters to generate the cloze. The language model (which we call NO REFERENCE) was conditioned on the context tokens and one full sentence was generated from the model without access to the reference document.

**No Context** We trained an abstractive model (§4.2) to summarize the first 200 tokens of the reference documents in one sentence, using the cloze as ground-truth, without access to the context. This is equivalent to training a single-document summarization system on the input text. This model is referred to as NO CONTEXT.

**Human Performance** We manually wrote clozes for 50 different instances with and without access to the topic and context to establish human performance on this task. We also had each cloze judged by 5 crowd workers to rate from 1 to 5 how coherent the topic, context, and cloze are. We report this average quality score alongside ROUGE with the HUMAN label.

### 5.2 Extractive Models

In addition to an ablation study on the extractive model (referred to as CONTENTSELECTOR; §4.1) where we remove the topic and context, we also evaluate several different extractive models on this task, described below.

**Lead & Oracle** The LEAD-1 chooses the first sentence from the reference documents as the cloze, a common baseline for summarization.

The ORACLE-1 model selects the one sentence from the reference set which maximizes the ROUGE score using the ground-truth. A different oracle model was created for each individual metric. The performance of the oracle represents the upper-bound ROUGE score that a system which extracts exactly one sentence can achieve.

**BM25** BM25 (Robertson et al., 1995) is an information retrieval-based score used to rank documents in search results for an input query. We selected the reference document sentence with the highest BM25 score for the context, treating the context as the query, the sentences as the documents, and computing the document frequency scores over the corpus of reference documents.

**SumFocus** SumFocus (Vanderwende et al., 2007) is an unsupervised model for topic-focused extractive summarization. The model constructs a unigram distribution over tokens by interpolating the unigram distributions of the documents and topic. Then, the model iteratively selects a sentence that has the highest probability token, discounts the probabilities of tokens in the selected sentence, and repeats. We extend SumFocus by additionally interpolating between the context unigram distribution in addition to the document and topic distributions.

### 5.3 Abstractive Models

Then, we measured how well the two-step abstractive system performs with and without access to the context and with different extractive models.

**Extractive Step Evaluation** We use the extractive models above to preprocess the reference set to select the top scoring sentences, up to 200 tokens. The preprocessed documents are evaluated against the ground-truth cloze by calculating the recall variant of ROUGE. This will measure what percent of the cloze $n$-grams are contained within the text input to the abstractive step.

In addition to the extractive models, we also report lead and extractive oracle scores. The LEAD-200 extractive step selects the first 200 tokens of the reference document. The extractive oracle model preprocesses the reference sentences by selecting only those which were labeled positively in the heuristically-generated extractive labels, limited to 200 tokens. This model, called HEURISTIC LABELS, serves to demonstrate how well the abstractive model would perform with a perfect extractive step.

**Abstractive Step Evaluation** Using the preprocessed documents from various extractive models, we trained abstractive models (§4.2) to produce the ground-truth cloze, both with and without access to the context. These systems were evaluated with the F1 variant of ROUGE and perplexity.

### 5.4 Implementation Details

Our models were implemented using PyTorch (Paszke et al., 2017) in the AllenNLP framework (Gardner et al., 2017).[4] We used fixed 200-dimensional GloVe embeddings (Pennington et al., 2014). The RNNs were implemented with LSTMs (Hochreiter and Schmidhuber, 1997) of size 256. The models were trained with Adam (Kingma and Ba, 2014) using a learning rate of 1e-4 and 1e-3 and batch sizes of 32 and 16 for 290k and 62.5k iterations for the extractive and abstractive models, respectively. Following See et al. (2017), we trained the abstractive models without the coverage loss until convergence, then by 3k iterations with a coverage loss weight of 1.

## 6 Experimental Results

### 6.1 Human Performance & Evaluation

Table 2 shows the ROUGE and quality scores for the human-written clozes with and without access to the topic and context. For an additional comparison, we included a random human baseline that randomly shuffled the human-written clozes.

[4] All code available at http://cogcomp.org/page/publication_view/880

| Model | R1 | R2 | RL | Qual. |
|---|---|---|---|---|
| GROUND-TRUTH | 100 | 100 | 100 | 4.0 |
| HUMAN | 31.78 | 16.94 | 28.32 | 3.9 |
| -TOPIC,-CONTEXT | 24.49 | 7.55 | 20.11 | 3.7 |
| RANDOM HUMAN | 9.96 | 0.08 | 8.24 | 2.3 |

Table 2: The ROUGE F1 scores for the human clozes with and without access to the topic and context and crowdsourced quality judgments (on a 1-5 scale) of how well the cloze continues the corresponding text. The standard deviations for the quality judgments (omitted for space) were each around 0.9.

| Model | R1 | R2 | RL |
|---|---|---|---|
| NO REFERENCE | 14.47 | 1.43 | 11.41 |
| NO CONTEXT | 21.79 | 6.18 | 17.83 |

Table 3: The ROUGE F1 results of the baseline models that do not have access to the references or context.

The ROUGE scores for when the human cloze writers had access to the topic and context are higher by 7.3 R1 points, a significant margin. From this observation, we can infer that, when provided with the topic and context, humans do a better job at identifying the same information present in the ground-truth cloze. This is indicative that the topic and context should also be necessary for a system to do the same.

However, the crowd worker quality judgments are roughly equal for the ground-truth and both human-written clozes, and human performance is at 31.78 R1, which is somewhat low on an absolute scale. We believe this reflects the difficulty of evaluating this task and summarization in general. Because there are many possible valid clozes (or valid summaries of documents in the more general case) and only one ground-truth cloze, evaluation with automatic metrics is difficult. Although ROUGE is imperfect, it does positively correlate with the crowd worker quality scores, with a Pearson's coefficient of 0.44.

### 6.2 Dataset Ablation

The two baseline models that do not have access to the references or context, respectively, are evaluated in Table 3. Although the NO REFERENCE model is a high-quality language model, it performs significantly worse than the NO CONTEXT model. This is likely due to the fact that the NO CONTEXT model was trained for this task.

| Model | R1 | R2 | RL |
|---|---|---|---|
| ORACLE-1 | 44.32 | 26.17 | 37.62 |
| LEAD-1 | 17.50 | 4.70 | 13.45 |
| BM25 | 21.33 | 5.76 | 16.18 |
| SUMFOCUS | 17.78 | 4.61 | 13.86 |
| CONTENTSELECTOR | **25.13** | **9.48** | **19.65** |
| -TOPIC | **25.21** | **9.49** | **19.68** |
| -CONTEXT | 22.11 | 6.53 | 16.66 |
| -TOPIC,-CONTEXT | 21.29 | 6.01 | 16.05 |

Table 4: The ROUGE F1 scores for the extractive models, all of which are significantly lower than the oracle model, indicating there is room for improvement.

### 6.3 Extractive Models

The performance of the different extractive models on the summary cloze task is presented in Table 4.

The CONTENTSELECTOR model achieves the best score overall with around 25 R1. Providing CONTENTSELECTOR with the topic and context improves the R1 score by nearly 4 points, however, this improvement is mostly due to the context. The topic does not change the model's performance by a statistically significant amount, which is an indication that it is not being properly utilized.

Among the symbolic models, BM25 outperforms SUMFOCUS by 3.55 R1. One reason for this could be that BM25 uses a token weighting to score each sentence based on how frequently the token appears in the entire reference corpora, whereas the standard SUMFOCUS implementation treats all tokens equally.

Although the models perform well, they are still significantly below the oracle model by at least 19 R1 and 16 R2 points. Such a large gap indicates that there is some signal in the data that is not yet captured by the extractive models.

### 6.4 Abstractive Models

**Extractive Step** The ROUGE recall scores for the different extractive models used to preprocess the input to the abstractive model are shown in Table 5. The results are largely consistent with the extractive model scores, with CONTENTSELECTOR performing the best, the topic causing no statistically significant difference, and a large gap remaining between current models and a perfect system (the heuristic labels).

Table 5 also contains an ablation study for the SUMFOCUS in which the topic did make a significant difference. This suggests that it may be easier to use the topic in a symbolic model and a model

| Model | R1-R | R2-R | RL-R |
|---|---|---|---|
| HEURISTIC LABELS | 80.25 | 35.75 | 55.11 |
| LEAD-200 | 60.89 | 22.31 | 44.97 |
| BM25 | 64.19 | 25.04 | 56.24 |
| SUMFOCUS | 61.79 | 22.77 | 54.15 |
| -TOPIC | 60.95 | 22.38 | 53.61 |
| -CONTEXT | 61.03 | 22.08 | 53.32 |
| -TOPIC,-CONTEXT | 59.11 | 20.62 | 51.93 |
| CONTENTSELECTOR | **66.27** | **27.59** | **58.25** |
| -TOPIC | **66.32** | **27.63** | **58.32** |
| -CONTEXT | 63.86 | 24.48 | 55.80 |
| -TOPIC,-CONTEXT | 63.11 | 23.76 | 55.09 |

Table 5: An evaluation using the recall variant of ROUGE of the different extractive preprocessing steps.

| Ext. Model | Abs. | R1 | R2 | RL | PPL |
|---|---|---|---|---|---|
| HEUR. LAB. | +C | 34.30 | 16.34 | 28.73 | 18.08 |
| | –C | 33.62 | 15.80 | 27.96 | 18.71 |
| LEAD-200 | +C | 23.56 | 7.15 | 19.3 | 39.73 |
| | –C | 21.79 | 6.18 | 17.83 | 40.90 |
| CONTSEL. | +C | **24.67** | **8.55** | **20.44** | 33.22 |
| | –C | 22.71 | 7.33 | 18.71 | 32.77 |

Table 6: The ROUGE F1 and perplexity results for abstractive models with and without the context (+/–C) with heuristic labels, lead, and CONTENTSELECTOR extractive preprocessing steps.

may benefit from combining symbolic and continuous representations.

**Abstractive Step** The ROUGE scores and perplexities for the abstractive models that use three different extractive steps are presented in Table 6. We observe that the CONTENTSELECTOR extractive step outperforms LEAD-200 in both ROUGE and perplexity. Additionally, including the context improves performance across all models, with nearly a 2 R1 gain with the CONTENTSELECTOR. This result is a clear signal that priming the decoder with the context produces a better cloze.

Example output from the CONTENTSELECTOR model that uses the context can been seen in Figure 4. In general, the model tends to copy long sequences from the reference document, often preferring somewhat formulaic sentences that begin with phrases such as "he became," "she graduated," or "she received" and are similar in style to Wikipedia articles.

Similar to the extractive results, there is a sizable gap between the performance of the abstractive model when it uses the heuristic labels versus the extractive model for preprocessing. This indicates that improving the extractive model will provide large downstream abstractive improvements.

## 7 Related Work

**Summarization** Recent work on generic summarization has focused on single-document models on the CNN/DailyMail dataset (Nallapati et al., 2016), focusing on different neural architectures and loss functions for extractive and abstractive summarization (Cheng and Lapata, 2016; Nallapati et al., 2017; Chopra et al., 2016; Nallapati et al., 2016; See et al., 2017, inter alia). Our models were based on the successful approaches of See et al. (2017) and those described in Kedzie et al. (2018).

Most work on topic-focused (also referred to as "query-focused") summarization uses the DUC 2005 and 2006 datasets (Dang, 2005, 2006). Approaches are mostly extractive, with some work selecting sentences based on token frequency/salience (Otterbacher et al., 2005; Vanderwende et al., 2007), diversity with submodular functions (Lin and Bilmes, 2011), or using Bayesian networks (Daumé III and Marcu, 2006). To the best of our knowledge, ours is the first work which builds end-to-end models on a large-scale dataset for topic-focused summarization.

**Generating Wikipedia** In contrast to our work focusing on content selection for topic-focused summaries, there have been previous work interested in generating Wikipedia articles. Sauper and Barzilay (2009) try to generate articles from a small set of domains by learning to generate templates from similar articles using an integer linear program. Similarly, Banerjee and Mitra (2016) build topic classifiers based on sections on Wikipedia to identify content in search results for topic keywords as part of a pipeline to generate full articles. Like us, Liu et al. (2018) also employ a two-step abstractive approach, but their goal is to generate the lead Wikipedia paragraph from all of the citations and web search results.

## 8 Conclusion

In this work, we propose the summary cloze task, a new task for studying content selection in topic-focused summarization. By narrowing the scope from the full topic-focused summarization task to one at the sentence-level, we are able to collect a large-scale summary cloze dataset from Wikipedia. Our experimental results demonstrate

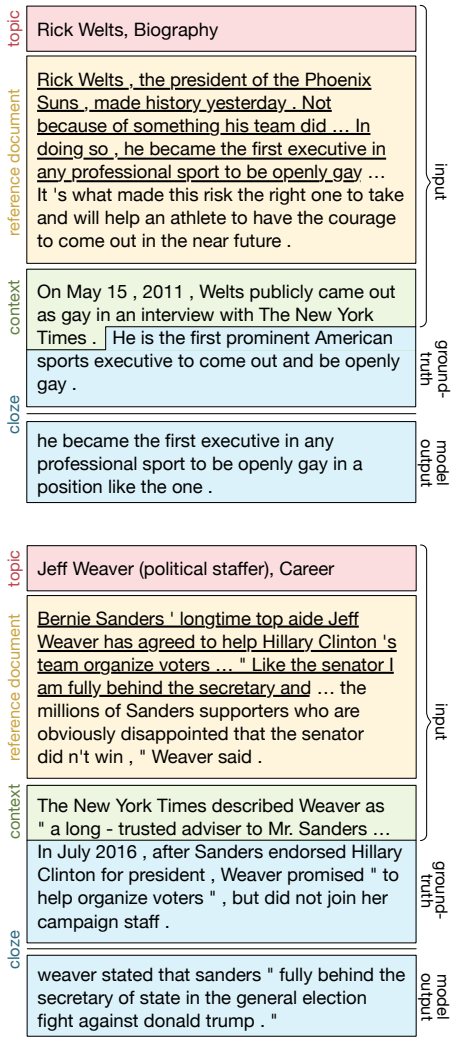


Figure 4: Example outputs from the abstractive model that uses the context. The model often copies sequences from the references which are sometimes correct (top) or incorrect but sensible (bottom), highlighting the difficulty of automatic evaluation. (Documents shortened for space. Sentences which are underlined were selected by the extraction step.)

that the topic and partial summary help the extractive and abstractive models, but the task remains a significant challenge with room for improvement in future work.

## Acknowledgements

The authors would like to thank the anonymous reviewers for their helpful feedback and suggestions.

Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053 (the ARL Network Science CTA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

## References

Siddhartha Banerjee and Prasenjit Mitra. 2016. WikiWrite: Generating Wikipedia Articles Automatically. In *IJCAI*, pages 2740–2746.

Jianpeng Cheng and Mirella Lapata. 2016. Neural Summarization by Extracting Sentences and Words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 484–494. Association for Computational Linguistics.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive Sentence Summarization with Attentive Recurrent Neural Networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98. Association for Computational Linguistics.

Hoa Trang Dang. 2005. Overview of DUC 2005. In *Proceedings of the document understanding conference*, volume 2005, pages 1–12. Citeseer.

Hoa Trang Dang. 2006. Overview of DUC 2006. In *Document Understanding Conference*.

Hal Daumé III and Daniel Marcu. 2006. Bayesian Query-Focused Summarization. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 305–312, Sydney, Australia. Association for Computational Linguistics.

Guy Feigenblat, Haggai Roitman, Odellia Boni, and David Konopnicki. 2017. Unsupervised Query-Focused Multi-Document Summarization Using the Cross Entropy Method. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 961–964, New York, NY, USA. ACM.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. AllenNLP: A Deep Semantic Natural Language Processing Platform.

Max Grusky, Mor Naaman, and Yoav Artzi. 2018. NEWSROOM: A Dataset of 1.3 Million Summaries with Diverse Extractive Strategies. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational*

*Linguistics: Human Language Technologies*, pages 708–719, New Orleans, Louisiana. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Chris Kedzie, Kathleen McKeown, and Hal Daume III. 2018. Content Selection in Deep Learning Models of Summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1818–1828, Brussels, Belgium. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Hui Lin and Jeff Bilmes. 2011. A Class of Submodular Functions for Document Summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 510–520, Portland, Oregon, USA. Association for Computational Linguistics.

Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating Wikipedia by Summarizing Long Sequences. *International Conference on Learning Representations 2018*.

Shulei Ma, Zhi-Hong Deng, and Yunlun Yang. 2016. An Unsupervised Multi-Document Summarization Framework Based on Neural Document Model. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1514–1523, Osaka, Japan. The COLING 2016 Organizing Committee.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A Recurrent Neural Network based Sequence Model for Extractive Summarization of Documents. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany. Association for Computational Linguistics.

Ani Nenkova. 2006a. Summarization evaluation for text and speech: issues and approaches. In *INTERSPEECH*.

Ani Nenkova. 2006b. *Understanding the process of multi-document summarization: content selection, rewriting and evaluation*. Columbia University.

Jahna Otterbacher, Gunes Erkan, and Dragomir Radev. 2005. Using Random Walks for Question-focused Sentence Retrieval. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 915–922, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Karolina Owczarzak, John M. Conroy, Hoa Trang Dang, and Ani Nenkova. 2012. An Assessment of the Accuracy of Automatic Evaluation in Summarization. In *Proceedings of Workshop on Evaluation Metrics and System Comparison for Automatic Summarization*, pages 1–9, Montréal, Canada. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS-W*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners.

Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. 1995. Okapi at TREC-3. *Nist Special Publication Sp*, 109:109.

Christina Sauper and Regina Barzilay. 2009. Automatically Generating Wikipedia Articles: A Structure-Aware Approach. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 208–216, Suntec, Singapore. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get To The Point: Summarization with Pointer-Generator Networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Lucy Vanderwende, Hisami Suzuki, Chris Brockett, and Ani Nenkova. 2007. Beyond SumBasic: Task-focused summarization with sentence simplification and lexical expansion. *Information Processing & Management*, 43(6):1606–1618.

# Text Summarization with Pretrained Encoders

**Yang Liu** and **Mirella Lapata**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
yang.liu2@ed.ac.uk, mlap@inf.ed.ac.uk

## Abstract

Bidirectional Encoder Representations from Transformers (BERT; Devlin et al. 2019) represents the latest incarnation of pretrained language models which have recently advanced a wide range of natural language processing tasks. In this paper, we showcase how BERT can be usefully applied in text summarization and propose a general framework for both extractive and abstractive models. We introduce a novel document-level encoder based on BERT which is able to express the semantics of a document and obtain representations for its sentences. Our extractive model is built on top of this encoder by stacking several inter-sentence Transformer layers. For abstractive summarization, we propose a new fine-tuning schedule which adopts different optimizers for the encoder and the decoder as a means of alleviating the mismatch between the two (the former is pretrained while the latter is not). We also demonstrate that a two-staged fine-tuning approach can further boost the quality of the generated summaries. Experiments on three datasets show that our model achieves state-of-the-art results across the board in both extractive and abstractive settings.[1]

## 1 Introduction

Language model pretraining has advanced the state of the art in many NLP tasks ranging from sentiment analysis, to question answering, natural language inference, named entity recognition, and textual similarity. State-of-the-art pretrained models include ELMo (Peters et al., 2018), GPT (Radford et al., 2018), and more recently Bidirectional Encoder Representations from Transformers (BERT; Devlin et al. 2019). BERT combines both word and sentence representations in a single very large Transformer (Vaswani et al., 2017); it is pretrained on vast amounts of text, with an unsupervised objective of masked language modeling and next-sentence prediction and can be fine-tuned with various task-specific objectives.

In most cases, pretrained language models have been employed as encoders for sentence- and paragraph-level natural language understanding problems (Devlin et al., 2019) involving various classification tasks (e.g., predicting whether any two sentences are in an entailment relationship; or determining the completion of a sentence among four alternative sentences). In this paper, we examine the influence of language model pretraining on text summarization. Different from previous tasks, summarization requires wide-coverage natural language understanding going beyond the meaning of individual words and sentences. The aim is to condense a *document* into a shorter version while preserving most of its meaning. Furthermore, under abstractive modeling formulations, the task requires language generation capabilities in order to create summaries containing novel words and phrases not featured in the source text, while extractive summarization is often defined as a binary classification task with labels indicating whether a text span (typically a sentence) should be included in the summary.

We explore the potential of BERT for text summarization under a general framework encompassing both extractive and abstractive modeling paradigms. We propose a novel document-level encoder based on BERT which is able to encode a document and obtain representations for its sentences. Our extractive model is built on top of this encoder by stacking several inter-sentence Transformer layers to capture document-level features for extracting sentences. Our abstractive model adopts an encoder-decoder architecture, combining the same pretrained BERT encoder with a randomly-initialized Transformer de-

[1] Our code is available at https://github.com/nlpyang/PreSumm.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3730–3740, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

coder (Vaswani et al., 2017). We design a new training schedule which separates the optimizers of the encoder and the decoder in order to accommodate the fact that the former is pretrained while the latter must be trained from scratch. Finally, motivated by previous work showing that the combination of extractive and abstractive objectives can help generate better summaries (Gehrmann et al., 2018), we present a two-stage approach where the encoder is fine-tuned twice, first with an extractive objective and subsequently on the abstractive summarization task.

We evaluate the proposed approach on three single-document news summarization datasets representative of different writing conventions (e.g., important information is concentrated at the beginning of the document or distributed more evenly throughout) and summary styles (e.g., verbose vs. more telegraphic; extractive vs. abstractive). Across datasets, we experimentally show that the proposed models achieve state-of-the-art results under both extractive and abstractive settings. Our contributions in this work are threefold: a) we highlight the importance of document encoding for the summarization task; a variety of recently proposed techniques aim to enhance summarization performance via copying mechanisms (Gu et al., 2016; See et al., 2017; Nallapati et al., 2017), reinforcement learning (Narayan et al., 2018b; Paulus et al., 2018; Dong et al., 2018), and multiple communicating encoders (Celikyilmaz et al., 2018). We achieve better results with a minimum-requirement model without using any of these mechanisms; b) we showcase ways to effectively employ pretrained language models in summarization under both extractive and abstractive settings; we would expect any improvements in model pretraining to translate in better summarization in the future; and c) the proposed models can be used as a stepping stone to further improve summarization performance as well as baselines against which new proposals are tested.

## 2 Background

### 2.1 Pretrained Language Models

Pretrained language models (Peters et al., 2018; Radford et al., 2018; Devlin et al., 2019; Dong et al., 2019; Zhang et al., 2019) have recently emerged as a key technology for achieving impressive gains in a wide variety of natural language tasks. These models extend the idea of word embeddings by learning contextual representations from large-scale corpora using a language modeling objective. Bidirectional Encoder Representations from Transformers (BERT; Devlin et al. 2019) is a new language representation model which is trained with a masked language modeling and a "next sentence prediction" task on a corpus of 3,300M words.

The general architecture of BERT is shown in the left part of Figure 1. Input text is first preprocessed by inserting two special tokens. [CLS] is appended to the beginning of the text; the output representation of this token is used to aggregate information from the whole sequence (e.g., for classification tasks). And token [SEP] is inserted after each sentence as an indicator of sentence boundaries. The modified text is then represented as a sequence of tokens $X = [w_1, w_2, \cdots, w_n]$. Each token $w_i$ is assigned three kinds of embeddings: *token embeddings* indicate the meaning of each token, *segmentation embeddings* are used to discriminate between two sentences (e.g., during a sentence-pair classification task) and *position embeddings* indicate the position of each token within the text sequence. These three embeddings are summed to a single input vector $x_i$ and fed to a bidirectional Transformer with multiple layers:

$$\tilde{h}^l = \mathrm{LN}(h^{l-1} + \mathrm{MHAtt}(h^{l-1})) \quad (1)$$

$$h^l = \mathrm{LN}(\tilde{h}^l + \mathrm{FFN}(\tilde{h}^l)) \quad (2)$$

where $h^0 = x$ are the input vectors; LN is the layer normalization operation (Ba et al., 2016); MHAtt is the multi-head attention operation (Vaswani et al., 2017); superscript $l$ indicates the depth of the stacked layer. On the top layer, BERT will generate an output vector $t_i$ for each token with rich contextual information.

Pretrained language models are usually used to enhance performance in language understanding tasks. Very recently, there have been attempts to apply pretrained models to various generation problems (Edunov et al., 2019; Rothe et al., 2019). When fine-tuning for a specific task, unlike ELMo whose parameters are usually fixed, parameters in BERT are *jointly* fine-tuned with additional task-specific parameters.

### 2.2 Extractive Summarization

Extractive summarization systems create a summary by identifying (and subsequently concatenating) the most important sentences in a document. Neural models consider extractive sum-

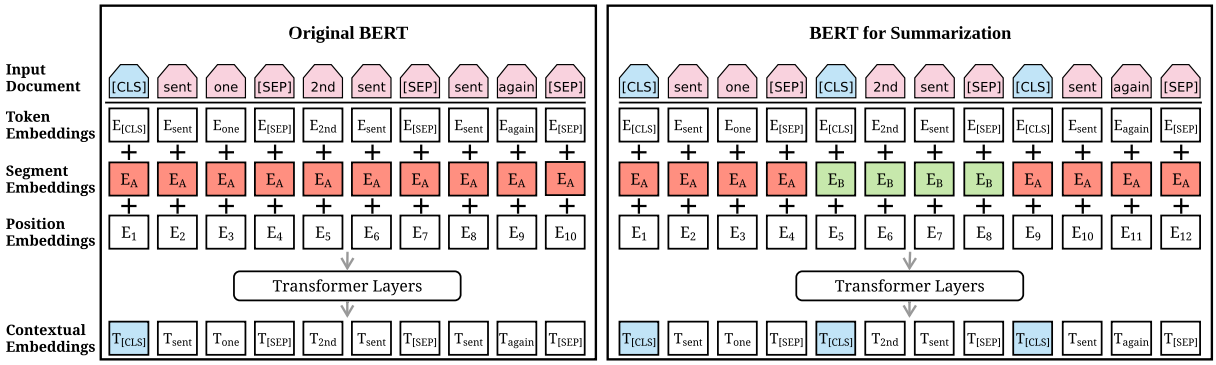


Figure 1: Architecture of the original BERT model (left) and BERTSUM (right). The sequence on top is the input document, followed by the summation of three kinds of embeddings for each token. The summed vectors are used as input embeddings to several bidirectional Transformer layers, generating contextual vectors for each token. BERTSUM extends BERT by inserting multiple [CLS] symbols to learn sentence representations and using interval segmentation embeddings (illustrated in red and green color) to distinguish multiple sentences.

marization as a sentence classification problem: a neural encoder creates sentence representations and a classifier predicts which sentences should be selected as summaries. SUMMARUNNER (Nallapati et al., 2017) is one of the earliest neural approaches adopting an encoder based on Recurrent Neural Networks. REFRESH (Narayan et al., 2018b) is a reinforcement learning-based system trained by globally optimizing the ROUGE metric. More recent work achieves higher performance with more sophisticated model structures. LATENT (Zhang et al., 2018) frames extractive summarization as a latent variable inference problem; instead of maximizing the likelihood of "gold" standard labels, their latent model directly maximizes the likelihood of human summaries given selected sentences. SUMO (Liu et al., 2019) capitalizes on the notion of structured attention to induce a multi-root dependency tree representation of the document while predicting the output summary. NEUSUM (Zhou et al., 2018) scores and selects sentences jointly and represents the state of the art in extractive summarization.

### 2.3 Abstractive Summarization

Neural approaches to abstractive summarization conceptualize the task as a sequence-to-sequence problem, where an encoder maps a sequence of tokens in the source document $\boldsymbol{x} = [x_1, ..., x_n]$ to a sequence of continuous representations $\boldsymbol{z} = [z_1, ..., z_n]$, and a decoder then generates the target summary $\boldsymbol{y} = [y_1, ..., y_m]$ token-by-token, in an auto-regressive manner, hence modeling the conditional probability: $p(y_1, ..., y_m|x_1, ..., x_n)$.

Rush et al. (2015) and Nallapati et al. (2016) were among the first to apply the neural encoder-decoder architecture to text summarization. See et al. (2017) enhance this model with a pointer-generator network (PTGEN) which allows it to copy words from the source text, and a coverage mechanism (COV) which keeps track of words that have been summarized. Celikyilmaz et al. (2018) propose an abstractive system where multiple agents (encoders) represent the document together with a hierarchical attention mechanism (over the agents) for decoding. Their Deep Communicating Agents (DCA) model is trained end-to-end with reinforcement learning. Paulus et al. (2018) also present a deep reinforced model (DRM) for abstractive summarization which handles the coverage problem with an intra-attention mechanism where the decoder attends over previously generated words. Gehrmann et al. (2018) follow a bottom-up approach (BOTTOMUP); a content selector first determines which phrases in the source document should be part of the summary, and a copy mechanism is applied only to preselected phrases during decoding. Narayan et al. (2018a) propose an abstractive model which is particularly suited to extreme summarization (i.e., single sentence summaries), based on convolutional neural networks and additionally conditioned on topic distributions (TCONVS2S).

## 3 Fine-tuning BERT for Summarization

### 3.1 Summarization Encoder

Although BERT has been used to fine-tune various NLP tasks, its application to summarization

is not as straightforward. Since BERT is trained as a masked-language model, the output vectors are grounded to tokens instead of sentences, while in extractive summarization, most models manipulate sentence-level representations. Although segmentation embeddings represent different sentences in BERT, they only apply to sentence-pair inputs, while in summarization we must encode and manipulate multi-sentential inputs. Figure 1 illustrates our proposed BERT architecture for SUMmarization (which we call BERTSUM).

In order to represent *individual* sentences, we insert external [CLS] tokens at the start of each sentence, and each [CLS] symbol collects features for the sentence preceding it. We also use *interval segment* embeddings to distinguish multiple sentences within a document. For $sent_i$ we assign segment embedding $E_A$ or $E_B$ depending on whether $i$ is odd or even. For example, for document $[sent_1, sent_2, sent_3, sent_4, sent_5]$, we would assign embeddings $[E_A, E_B, E_A, E_B, E_A]$. This way, document representations are learned hierarchically where lower Transformer layers represent adjacent sentences, while higher layers, in combination with self-attention, represent multi-sentence discourse.

Position embeddings in the original BERT model have a maximum length of 512; we overcome this limitation by adding more position embeddings that are initialized randomly and fine-tuned with other parameters in the encoder.

### 3.2 Extractive Summarization

Let $d$ denote a document containing sentences $[sent_1, sent_2, \cdots, sent_m]$, where $sent_i$ is the $i$-th sentence in the document. Extractive summarization can be defined as the task of assigning a label $y_i \in \{0, 1\}$ to each $sent_i$, indicating whether the sentence should be included in the summary. It is assumed that summary sentences represent the most important content of the document.

With BERTSUM, vector $t_i$ which is the vector of the $i$-th [CLS] symbol from the top layer can be used as the representation for $sent_i$. Several inter-sentence Transformer layers are then stacked on top of BERT outputs, to capture document-level features for extracting summaries:

$$\tilde{h}^l = \mathrm{LN}(h^{l-1} + \mathrm{MHAtt}(h^{l-1})) \quad (3)$$

$$h^l = \mathrm{LN}(\tilde{h}^l + \mathrm{FFN}(\tilde{h}^l)) \quad (4)$$

where $h^0 = \mathrm{PosEmb}(T)$; $T$ denotes the sentence vectors output by BERTSUM, and function PosEmb adds sinusoid positional embeddings (Vaswani et al., 2017) to $T$, indicating the position of each sentence.

The final output layer is a sigmoid classifier:

$$\hat{y}_i = \sigma(W_o h_i^L + b_o) \quad (5)$$

where $h_i^L$ is the vector for $sent_i$ from the top layer (the $L$-th layer ) of the Transformer. In experiments, we implemented Transformers with $L = 1, 2, 3$ and found that a Transformer with $L = 2$ performed best. We name this model BERTSUMEXT.

The loss of the model is the binary classification entropy of prediction $\hat{y}_i$ against gold label $y_i$. Inter-sentence Transformer layers are jointly fine-tuned with BERTSUM. We use the Adam optimizer with $\beta_1 = 0.9$, and $\beta_2 = 0.999$). Our learning rate schedule follows (Vaswani et al., 2017) with warming-up ($\mathrm{warmup} = 10,000$):

$$lr = 2e^{-3} \cdot \min(\mathrm{step}^{-0.5}, \mathrm{step} \cdot \mathrm{warmup}^{-1.5})$$

### 3.3 Abstractive Summarization

We use a standard encoder-decoder framework for abstractive summarization (See et al., 2017). The encoder is the pretrained BERTSUM and the decoder is a 6-layered Transformer initialized randomly. It is conceivable that there is a mismatch between the encoder and the decoder, since the former is pretrained while the latter must be trained from scratch. This can make fine-tuning unstable; for example, the encoder might overfit the data while the decoder underfits, or vice versa. To circumvent this, we design a new fine-tuning schedule which separates the optimizers of the encoder and the decoder.

We use two Adam optimizers with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ for the encoder and the decoder, respectively, each with different warmup-steps and learning rates:

$$lr_{\mathcal{E}} = \tilde{lr}_{\mathcal{E}} \cdot \min(step^{-0.5}, \mathrm{step} \cdot \mathrm{warmup}_{\mathcal{E}}^{-1.5}) \quad (6)$$

$$lr_{\mathcal{D}} = \tilde{lr}_{\mathcal{D}} \cdot \min(step^{-0.5}, \mathrm{step} \cdot \mathrm{warmup}_{\mathcal{D}}^{-1.5}) \quad (7)$$

where $\tilde{lr}_{\mathcal{E}} = 2e^{-3}$, and $\mathrm{warmup}_{\mathcal{E}} = 20,000$ for the encoder and $\tilde{lr}_{\mathcal{D}} = 0.1$, and $\mathrm{warmup}_{\mathcal{D}} = 10,000$ for the decoder. This is based on the assumption that the pretrained encoder should be fine-tuned with a smaller learning rate and smoother decay (so that the encoder can be trained with more accurate gradients when the decoder is becoming stable).

| Datasets | # docs (train/val/test) | avg. doc length | | avg. summary length | | % novel bi-grams |
|---|---|---|---|---|---|---|
| | | words | sentences | words | sentences | in gold summary |
| CNN | 90,266/1,220/1,093 | 760.50 | 33.98 | 45.70 | 3.59 | 52.90 |
| DailyMail | 196,961/12,148/10,397 | 653.33 | 29.33 | 54.65 | 3.86 | 52.16 |
| NYT | 96,834/4,000/3,452 | 800.04 | 35.55 | 45.54 | 2.44 | 54.70 |
| XSum | 204,045/11,332/11,334 | 431.07 | 19.77 | 23.26 | 1.00 | 83.31 |

Table 1: Comparison of summarization datasets: size of training, validation, and test sets and average document and summary length (in terms of words and sentences). The proportion of novel bi-grams that do not appear in source documents but do appear in the gold summaries quantifies corpus bias towards extractive methods.

In addition, we propose a two-stage fine-tuning approach, where we first fine-tune the encoder on the extractive summarization task (Section 3.2) and then fine-tune it on the abstractive summarization task (Section 3.3). Previous work (Gehrmann et al., 2018; Li et al., 2018) suggests that using extractive objectives can boost the performance of abstractive summarization. Also notice that this two-stage approach is conceptually very simple, the model can take advantage of information shared between these two tasks, without fundamentally changing its architecture. We name the default abstractive model BERTSUMABS and the two-stage fine-tuned model BERTSUMEXTABS.

## 4 Experimental Setup

In this section, we describe the summarization datasets used in our experiments and discuss various implementation details.

### 4.1 Summarization Datasets

We evaluated our model on three benchmark datasets, namely the CNN/DailyMail news highlights dataset (Hermann et al., 2015), the New York Times Annotated Corpus (NYT; Sandhaus 2008), and XSum (Narayan et al., 2018a). These datasets represent different summary styles ranging from highlights to very brief one sentence summaries. The summaries also vary with respect to the type of rewriting operations they exemplify (e.g., some showcase more cut and paste operations while others are genuinely abstractive). Table 1 presents statistics on these datasets (test set); example (gold-standard) summaries are provided in the supplementary material.

**CNN/DailyMail** contains news articles and associated highlights, i.e., a few bullet points giving a brief overview of the article. We used the standard splits of Hermann et al. (2015) for training, validation, and testing (90,266/1,220/1,093 CNN documents and 196,961/12,148/10,397 DailyMail documents). We did not anonymize entities. We first split sentences with the Stanford CoreNLP toolkit (Manning et al., 2014) and pre-processed the dataset following See et al. (2017). Input documents were truncated to 512 tokens.

**NYT** contains 110,540 articles with abstractive summaries. Following Durrett et al. (2016), we split these into 100,834/9,706 training/test examples, based on the date of publication (the test set contains all articles published from January 1, 2007 onward). We used 4,000 examples from the training as validation set. We also followed their filtering procedure, documents with summaries less than 50 words were removed from the dataset. The filtered test set (NYT50) includes 3,452 examples. Sentences were split with the Stanford CoreNLP toolkit (Manning et al., 2014) and pre-processed following Durrett et al. (2016). Input documents were truncated to 800 tokens.

**XSum** contains 226,711 news articles accompanied with a one-sentence summary, answering the question "What is this article about?". We used the splits of Narayan et al. (2018a) for training, validation, and testing (204,045/11,332/11,334) and followed the pre-processing introduced in their work. Input documents were truncated to 512 tokens.

Aside from various statistics on the three datasets, Table 1 also reports the proportion of novel bi-grams in gold summaries as a measure of their abstractiveness. We would expect models with extractive biases to perform better on datasets with (mostly) extractive summaries, and abstractive models to perform more rewrite operations on datasets with abstractive summaries. CNN/DailyMail and NYT are somewhat extractive, while XSum is highly abstractive.

### 4.2 Implementation Details

For both extractive and abstractive settings, we used PyTorch, OpenNMT (Klein et al., 2017) and the 'bert-base-uncased'[2] version of BERT to implement BERTSUM. Both source and target texts

[2] https://git.io/fhbJQ

were tokenized with BERT's subwords tokenizer.

**Extractive Summarization** All extractive models were trained for 50,000 steps on 3 GPUs (GTX 1080 Ti) with gradient accumulation every two steps. Model checkpoints were saved and evaluated on the validation set every 1,000 steps. We selected the top-3 checkpoints based on the evaluation loss on the validation set, and report the averaged results on the test set. We used a greedy algorithm similar to Nallapati et al. (2017) to obtain an oracle summary for each document to train extractive models. The algorithm generates an oracle consisting of multiple sentences which maximize the ROUGE-2 score against the gold summary.

When predicting summaries for a new document, we first use the model to obtain the score for each sentence. We then rank these sentences by their scores from highest to lowest, and select the top-3 sentences as the summary.

During sentence selection we use *Trigram Blocking* to reduce redundancy (Paulus et al., 2018). Given summary $S$ and candidate sentence $c$, we skip $c$ if there exists a trigram overlapping between $c$ and $S$. The intuition is similar to Maximal Marginal Relevance (MMR; Carbonell and Goldstein 1998); we wish to minimize the similarity between the sentence being considered and sentences which have been already selected as part of the summary.

**Abstractive Summarization** In all abstractive models, we applied dropout (with probability 0.1) before all linear layers; label smoothing (Szegedy et al., 2016) with smoothing factor 0.1 was also used. Our Transformer decoder has 768 hidden units and the hidden size for all feed-forward layers is 2,048. All models were trained for 200,000 steps on 4 GPUs (GTX 1080 Ti) with gradient accumulation every five steps. Model checkpoints were saved and evaluated on the validation set every 2,500 steps. We selected the top-3 checkpoints based on their evaluation loss on the validation set, and report the averaged results on the test set.

During decoding we used beam search (size 5), and tuned the $\alpha$ for the length penalty (Wu et al., 2016) between 0.6 and 1 on the validation set; we decode until an end-of-sequence token is emitted and repeated trigrams are blocked (Paulus et al., 2018). It is worth noting that our decoder applies neither a copy nor a coverage mechanism (See et al., 2017), despite their popularity in abstractive summarization. This is mainly because we focus on building a minimum-requirements model and these mechanisms may introduce additional hyper-parameters to tune. Thanks to the subwords tokenizer, we also rarely observe issues with out-of-vocabulary words in the output; moreover, trigram-blocking produces diverse summaries managing to reduce repetitions.

| Model | R1 | R2 | RL |
|---|---|---|---|
| ORACLE | 52.59 | 31.24 | 48.87 |
| LEAD-3 | 40.42 | 17.62 | 36.67 |
| Extractive | | | |
| SUMMARUNNER (Nallapati et al., 2017) | 39.60 | 16.20 | 35.30 |
| REFRESH (Narayan et al., 2018b) | 40.00 | 18.20 | 36.60 |
| LATENT (Zhang et al., 2018) | 41.05 | 18.77 | 37.54 |
| NEUSUM (Zhou et al., 2018) | 41.59 | 19.01 | 37.98 |
| SUMO (Liu et al., 2019) | 41.00 | 18.40 | 37.20 |
| TransformerEXT | 40.90 | 18.02 | 37.17 |
| Abstractive | | | |
| PTGEN (See et al., 2017) | 36.44 | 15.66 | 33.42 |
| PTGEN+COV (See et al., 2017) | 39.53 | 17.28 | 36.38 |
| DRM (Paulus et al., 2018) | 39.87 | 15.82 | 36.90 |
| BOTTOMUP (Gehrmann et al., 2018) | 41.22 | 18.68 | 38.34 |
| DCA (Celikyilmaz et al., 2018) | 41.69 | 19.47 | 37.92 |
| TransformerABS | 40.21 | 17.76 | 37.09 |
| BERT-based | | | |
| BERTSUMEXT | 43.25 | 20.24 | 39.63 |
| BERTSUMEXT w/o interval embeddings | 43.20 | 20.22 | 39.59 |
| BERTSUMEXT (large) | 43.85 | 20.34 | 39.90 |
| BERTSUMABS | 41.72 | 19.39 | 38.76 |
| BERTSUMEXTABS | 42.13 | 19.60 | 39.18 |

Table 2: ROUGE F1 results on **CNN/DailyMail** test set (R1 and R2 are shorthands for unigram and bigram overlap; RL is the longest common subsequence). Results for comparison systems are taken from the authors' respective papers or obtained on our data by running publicly released software.

## 5 Results

### 5.1 Automatic Evaluation

We evaluated summarization quality automatically using ROUGE (Lin, 2004). We report unigram and bigram overlap (ROUGE-1 and ROUGE-2) as a means of assessing informativeness and the longest common subsequence (ROUGE-L) as a means of assessing fluency.

Table 2 summarizes our results on the CNN/DailyMail dataset. The first block in the table includes the results of an extractive ORACLE system as an upper bound. We also present the LEAD-3 baseline (which simply selects the first three sentences in a document).

The second block in the table includes various extractive models trained on the CNN/DailyMail dataset (see Section 2.2 for an overview). For

| Model | R1 | R2 | RL |
|---|---|---|---|
| ORACLE | 49.18 | 33.24 | 46.02 |
| LEAD-3 | 39.58 | 20.11 | 35.78 |
| Extractive | | | |
| COMPRESS (Durrett et al., 2016) | 42.20 | 24.90 | — |
| SUMO (Liu et al., 2019) | 42.30 | 22.70 | 38.60 |
| TransformerEXT | 41.95 | 22.68 | 38.51 |
| Abstractive | | | |
| PTGEN (See et al., 2017) | 42.47 | 25.61 | — |
| PTGEN + COV (See et al., 2017) | 43.71 | 26.40 | — |
| DRM (Paulus et al., 2018) | 42.94 | 26.02 | — |
| TransformerABS | 35.75 | 17.23 | 31.41 |
| BERT-based | | | |
| BERTSUMEXT | 46.66 | 26.35 | 42.62 |
| BERTSUMABS | 48.92 | 30.84 | 45.41 |
| BERTSUMEXTABS | 49.02 | 31.02 | 45.55 |

Table 3: ROUGE Recall results on **NYT** test set. Results for comparison systems are taken from the authors' respective papers or obtained on our data by running publicly released software. Table cells are filled with — whenever results are not available.

| Model | R1 | R2 | RL |
|---|---|---|---|
| ORACLE | 29.79 | 8.81 | 22.66 |
| LEAD | 16.30 | 1.60 | 11.95 |
| Abstractive | | | |
| PTGEN (See et al., 2017) | 29.70 | 9.21 | 23.24 |
| PTGEN+COV (See et al., 2017) | 28.10 | 8.02 | 21.72 |
| TCONVS2S (Narayan et al., 2018a) | 31.89 | 11.54 | 25.75 |
| TransformerABS | 29.41 | 9.77 | 23.01 |
| BERT-based | | | |
| BERTSUMABS | 38.76 | 16.33 | 31.15 |
| BERTSUMEXTABS | 38.81 | 16.50 | 31.27 |

Table 4: ROUGE F1 results on the **XSum** test set. Results for comparison systems are taken from the authors' respective papers or obtained on our data by running publicly released software.

comparison to our own model, we also implemented a non-pretrained Transformer baseline (TransformerEXT) which uses the same architecture as BERTSUMEXT, but with fewer parameters. It is randomly initialized and only trained on the summarization task. TransformerEXT has 6 layers, the hidden size is 512, and the feed-forward filter size is 2,048. The model was trained with same settings as in Vaswani et al. (2017).

The third block in Table 2 highlights the performance of several abstractive models on the CNN/DailyMail dataset (see Section 2.3 for an overview). We also include an abstractive Transformer baseline (TransformerABS) which has the same decoder as our abstractive BERTSUM models; the encoder is a 6-layer Transformer with 768 hidden size and 2,048 feed-forward filter size.

The fourth block reports results with fine-tuned BERT models: BERTSUMEXT and its two variants (one without interval embeddings, and one with the large version of BERT), BERTSUMABS, and BERTSUMEXTABS. BERT-based models outperform the LEAD-3 baseline which is not a strawman; on the CNN/DailyMail corpus it is indeed superior to several extractive (Nallapati et al., 2017; Narayan et al., 2018b; Zhou et al., 2018) and abstractive models (See et al., 2017). BERT models collectively outperform all previously proposed extractive and abstractive systems, only falling behind the ORACLE upper bound. Among BERT variants, BERTSUMEXT performs best which is not entirely surprising; CNN/DailyMail summaries are somewhat extractive and even abstractive models are prone to copying sentences from the source document when trained on this dataset (See et al., 2017). Perhaps unsurprisingly we observe that larger versions of BERT lead to performance improvements and that interval embeddings bring only slight gains.

Table 3 presents results on the NYT dataset. Following the evaluation protocol in Durrett et al. (2016), we use limited-length ROUGE Recall, where predicted summaries are truncated to the length of the gold summaries. Again, we report the performance of the ORACLE upper bound and LEAD-3 baseline. The second block in the table contains previously proposed extractive models as well as our own Transformer baseline. COMPRESS (Durrett et al., 2016) is an ILP-based model which combines compression and anaphoricity constraints. The third block includes abstractive models from the literature, and our Transformer baseline. BERT-based models are shown in the fourth block. Again, we observe that they outperform previously proposed approaches. On this dataset, abstractive BERT models generally perform better compared to BERTSUMEXT, almost approaching ORACLE performance.

Table 4 summarizes our results on the XSum dataset. Recall that summaries in this dataset are highly abstractive (see Table 1) consisting of a single sentence conveying the gist of the document. Extractive models here perform poorly as corroborated by the low performance of the LEAD baseline (which simply selects the leading sentence from the document), and the ORACLE (which selects a single-best sentence in each document) in Table 4. As a result, we do not report results for extractive models on this dataset. The second

| $\tilde{lr}_{\mathcal{E}}$ \ $\tilde{lr}_{\mathcal{D}}$ | 1 | 0.1 | 0.01 | 0.001 |
|---|---|---|---|---|
| 2e-2 | 50.69 | 9.33 | 10.13 | 19.26 |
| 2e-3 | 37.21 | 8.73 | 9.52 | 16.88 |

Table 5: Model perplexity (CNN/DailyMail; validation set) under different combinations of encoder and decoder learning rates.

block in Table 4 presents the results of various abstractive models taken from Narayan et al. (2018a) and also includes our own abstractive Transformer baseline. In the third block we show the results of our BERT summarizers which again are superior to all previously reported models (by a wide margin).

### 5.2 Model Analysis

**Learning Rates** Recall that our abstractive model uses separate optimizers for the encoder and decoder. In Table 5 we examine whether the combination of different learning rates ($\tilde{lr}_{\mathcal{E}}$ and $\tilde{lr}_{\mathcal{D}}$) is indeed beneficial. Specifically, we report model perplexity on the CNN/DailyMail validation set for varying encoder/decoder learning rates. We can see that the model performs best with $\tilde{lr}_{\mathcal{E}} = 2e-3$ and $\tilde{lr}_{\mathcal{D}} = 0.1$.

**Position of Extracted Sentences** In addition to the evaluation based on ROUGE, we also analyzed in more detail the summaries produced by our model. For the extractive setting, we looked at the position (in the source document) of the sentences which were selected to appear in the summary. Figure 2 shows the proportion of selected summary sentences which appear in the source document at positions 1, 2, and so on. The analysis was conducted on the CNN/DailyMail dataset for Oracle summaries, and those produced by BERTSUMEXT and the TransformerEXT. We can see that Oracle summary sentences are fairly smoothly distributed across documents, while summaries created by TransformerEXT mostly concentrate on the first document sentences. BERTSUMEXT outputs are more similar to Oracle summaries, indicating that with the pretrained encoder, the model relies less on shallow position features, and learns deeper document representations.

**Novel N-grams** We also analyzed the output of abstractive systems by calculating the proportion of novel n-grams that appear in the summaries but not in the source texts. The results are shown in Figure 3. In the CNN/DailyMail dataset, the proportion of novel n-grams in automatically generated summaries is much lower compared to reference summaries, but in XSum, this gap is much smaller. We also observe that on CNN/DailyMail, BERTEXTABS produces less novel n-ngrams than BERTABS, which is not surprising. BERTEXTABS is more biased towards selecting sentences from the source document since it is initially trained as an extractive model.

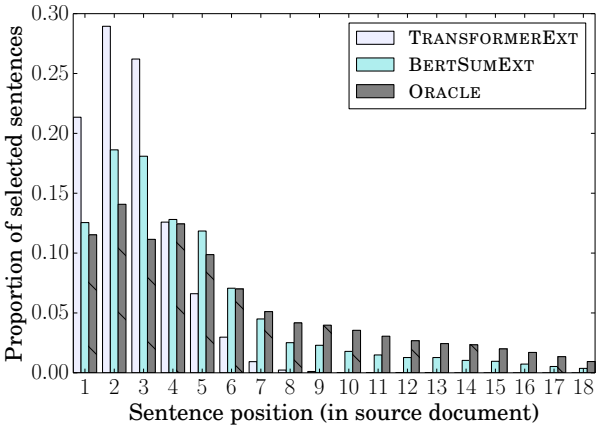


Figure 2: Proportion of extracted sentences according to their position in the original document.

The supplementary material includes examples of system output and additional ablation studies.

### 5.3 Human Evaluation

In addition to automatic evaluation, we also evaluated system output by eliciting human judgments. We report experiments following a question-answering (QA) paradigm (Clarke and Lapata, 2010; Narayan et al., 2018b) which quantifies the degree to which summarization models retain key information from the document. Under this paradigm, a set of questions is created based on the gold summary under the assumption that it highlights the most important document content. Participants are then asked to answer these questions by reading system summaries alone without access to the article. The more questions a system can answer, the better it is at summarizing the document as a whole.

Moreover, we also assessed the overall quality of the summaries produced by abstractive systems which due to their ability to rewrite content may produce disfluent or ungrammatical output. Specifically, we followed the Best-Worst Scaling (Kiritchenko and Mohammad, 2017) method where participants were presented with the output of two systems (and the original document) and

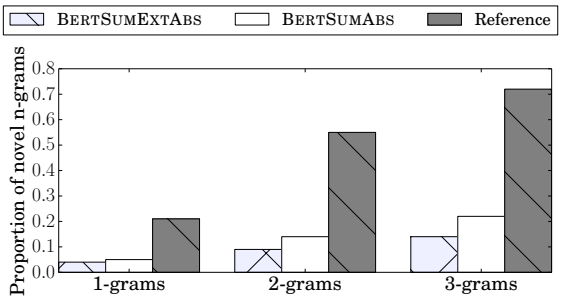


(a) CNN/DailyMail Dataset

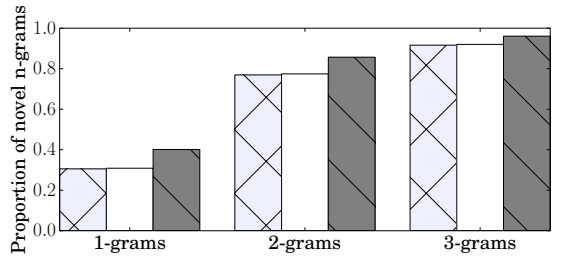


(b) XSum dataset

Figure 3: Proportion of novel n-grams in model generated summaries.

| Extractive | CNN/DM | NYT |
|---|---|---|
| LEAD | 42.5† | 36.2† |
| NEUSUM | 42.2† | — |
| SUMO | 41.7† | 38.1† |
| Transformer | 37.8† | 32.5† |
| BERTSUM | 58.9 | 41.9 |

Table 6: QA-based evaluation. Models with † are significantly different from BERTSUM (using a paired student t-test; $p < 0.05$). Table cells are filled with — whenever system output is not available.

asked to decide which one was better according to the criteria of *Informativeness*, *Fluency*, and *Succinctness*.

Both types of evaluation were conducted on the Amazon Mechanical Turk platform. For the CNN/DailyMail and NYT datasets we used the same documents (20 in total) and questions from previous work (Narayan et al., 2018b; Liu et al., 2019). For XSum, we randomly selected 20 documents (and their questions) from the release of Narayan et al. (2018a). We elicited 3 responses per HIT. With regard to QA evaluation, we adopted the scoring mechanism from Clarke and Lapata (2010); correct answers were marked with a score of one, partially correct answers with 0.5, and zero otherwise. For quality-based evaluation, the rating of each system was computed as the percentage of times it was chosen as better minus the times it was selected as worse. Ratings thus range from -1 (worst) to 1 (best).

| Abstractive | CNN/DM | | NYT | | XSum | |
|---|---|---|---|---|---|---|
| | QA | Rank | QA | Rank | QA | Rank |
| LEAD | 42.5† | — | 36.2† | — | 9.20† | — |
| PTGEN | 33.3† | -0.24† | 30.5† | -0.27† | 23.7† | -0.36† |
| BOTTOMUP | 40.6† | -0.16† | — | — | — | — |
| TCONVS2S | — | — | — | — | 52.1 | -0.20† |
| GOLD | — | 0.22† | — | 0.33† | — | 0.38† |
| BERTSUM | 56.1 | 0.17 | 41.8 | -0.07 | 57.5 | 0.19 |

Table 7: QA-based and ranking-based evaluation. Models with † are significantly different from BERTSUM (using a paired student t-test; $p < 0.05$). Table cells are filled with — whenever system output is not available. GOLD is not used in QA setting, and LEAD is not used in Rank evaluation.

Results for extractive and abstractive systems are shown in Tables 6 and 7, respectively. We compared the best performing BERTSUM model in each setting (extractive or abstractive) against various state-of-the-art systems (whose output is publicly available), the LEAD baseline, and the GOLD standard as an upper bound. As shown in both tables participants overwhelmingly prefer the output of our model against comparison systems across datasets and evaluation paradigms. All differences between BERTSUM and comparison models are statistically significant ($p < 0.05$), with the exception of TCONVS2S (see Table 7; XSum) in the QA evaluation setting.

## 6 Conclusions

In this paper, we showcased how pretrained BERT can be usefully applied in text summarization. We introduced a novel document-level encoder and proposed a general framework for both abstractive and extractive summarization. Experimental results across three datasets show that our model achieves state-of-the-art results across the board under automatic and human-based evaluation protocols. Although we mainly focused on document encoding for summarization, in the future, we would like to take advantage the capabilities of BERT for language generation.

## Acknowledgments

This research is supported by a Google PhD Fellowship to the first author. We gratefully acknowledge the support of the European Research Council (Lapata, award number 681760, "Translating Multiple Modalities into Text"). We would also like to thank Shashi Narayan for providing us with the XSum dataset.

## References


Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Jaime G Carbonell and Jade Goldstein. 1998. The use of MMR and diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACL SIGIR Conference on Research and Development in Information Retrieval*, pages 335–336, Melbourne, Australia.

Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep communicating agents for abstractive summarization. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1662–1675, New Orleans, Louisiana.

James Clarke and Mirella Lapata. 2010. Discourse constraints for document compression. *Computational Linguistics*, 36(3):411–441.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. *arXiv preprint arXiv:1905.03197*.

Yue Dong, Yikang Shen, Eric Crawford, Herke van Hoof, and Jackie Chi Kit Cheung. 2018. BanditSum: Extractive summarization as a contextual bandit. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3739–3748, Brussels, Belgium.

Greg Durrett, Taylor Berg-Kirkpatrick, and Dan Klein. 2016. Learning-based single-document summarization with compression and anaphoricity constraints. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1998–2008, Berlin, Germany.

Sergey Edunov, Alexei Baevski, and Michael Auli. 2019. Pre-trained language model representations for language generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4052–4059, Minneapolis, Minnesota.

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109, Brussels, Belgium.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany. Association for Computational Linguistics.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Svetlana Kiritchenko and Saif Mohammad. 2017. Best-worst scaling more reliable than rating scales: A case study on sentiment intensity annotation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 465–470, Vancouver, Canada.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada.

Wei Li, Xinyan Xiao, Yajuan Lyu, and Yuanzhuo Wang. 2018. Improving neural abstractive document summarization with explicit information selection modeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1787–1796, Brussels, Belgium.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain.

Yang Liu, Ivan Titov, and Mirella Lapata. 2019. Single document summarization as tree induction. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1745–1755, Minneapolis, Minnesota.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. SummaRuNNer: A recurrent neural network based sequence model for extractive summarization of

documents. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, pages 3075–3081, San Francisco, California.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gülçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018a. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018b. Ranking sentences for extractive summarization with reinforcement learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1747–1759, New Orleans, Louisiana.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *Proceedings of the 6th International Conference on Learning Representations*, Vancouver, Canada.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. In *CoRR, abs/1704.01444, 2017*.

Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2019. Leveraging pre-trained checkpoints for sequence generation tasks. *arXiv preprint arXiv:1907.12461*.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal.

Evan Sandhaus. 2008. The New York Times Annotated Corpus. *Linguistic Data Consortium, Philadelphia*, 6(12).

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, Las Vegas, Nevada.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. In *arXiv preprint arXiv:1609.08144*.

Xingxing Zhang, Mirella Lapata, Furu Wei, and Ming Zhou. 2018. Neural latent extractive document summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 779–784, Brussels, Belgium.

Xingxing Zhang, Furu Wei, and Ming Zhou. 2019. HIBERT: Document level pre-training of hierarchical bidirectional transformers for document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5059–5069, Florence, Italy. Association for Computational Linguistics.

Qingyu Zhou, Nan Yang, Furu Wei, Shaohan Huang, Ming Zhou, and Tiejun Zhao. 2018. Neural document summarization by jointly learning to score and select sentences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 654–663, Melbourne, Australia.

# How to Write Summaries with Patterns? Learning towards Abstractive Summarization through Prototype Editing

**Shen Gao** [♠,*], **Xiuying Chen** [♠♣,*], **Piji Li** [♢], **Zhangming Chan** [♠♣], **Dongyan Zhao** [♠♣], **Rui Yan** [♠♣,†]
[♠]Wangxuan Institute of Computer Technology, Peking University, Beijing, China,
[♢]Tencent AI Lab, Shenzhen, China
[♣]Center for Data Science, Peking University, Beijing, China
{shengao, xy-chen, zhangming.chan, zhaody, ruiyan}@pku.edu.cn
pijili@tencent.com

## Abstract

Under special circumstances, summaries should conform to a particular style with patterns, such as court judgments and abstracts in academic papers. To this end, the prototype document-summary pairs can be utilized to generate better summaries. There are two main challenges in this task: (1) the model needs to incorporate learned patterns from the prototype, but (2) should avoid copying contents other than the patternized words—such as irrelevant facts—into the generated summaries. To tackle these challenges, we design a model named *Prototype Editing based Summary Generator* (PESG). PESG first **learns summary patterns** and prototype facts by analyzing the correlation between a prototype document and its summary. Prototype facts are then utilized to help **extract facts** from the input document. Next, an editing generator generates new summary based on the **summary pattern** or **extracted facts**. Finally, to address the second challenge, a fact checker is used to estimate mutual information between the input document and generated summary, providing an additional signal for the generator. Extensive experiments conducted on a large-scale real-world text summarization dataset[1] show that PESG achieves the state-of-the-art performance in terms of both automatic metrics and human evaluations.

## 1 Introduction

Abstractive summarization can be regarded as a sequence mapping task that maps the source text to the target summary (Rush et al., 2015; Li et al., 2017; Cao et al., 2018; Gao et al., 2019a). It has drawn significant attention since the introduction of deep neural networks to natural language processing. Under special circumstances, the generated summaries are required to conform to a specific pattern, such as court judgments, diagnosis certificates, abstracts in academic papers, etc. Take the court judgments for example, there is always a statement of the crime committed by the accused, followed by the motives and the results of the judgment. An example case is shown in Table 1, where the summary shares the same writing style and has words in common with the prototype summary (retrieved from the training dataset).

| | |
|---|---|
| prototype summary | The court held that the defendant Wang had stolen the property of others for the purpose of illegal possession. The amount was large, and his behavior constituted the crime of theft. The accusation of the public prosecution agency was established. The defendant Wang has a criminal record and will be considered when sentencing. Since the defendant Wang did not succeed because of reasons other than his will, he could be punished lightly. After the defendant confessed his crimes to the case, he was given a lighter punishment according to law. |
| summary | The court held that the accused Zhang and Fan stole property and the amount was large. Their actions constituted the crime of theft. The accusation of the public prosecution agency was established and supported. This crime was committed within two years after the release of the defendants Zhang and Fan. Thus they are recidivists and this situation will be considered when sentencing. The fact that defendants Zhang and Fan surrendered themselves and pleaded guilty in court gives a lighter punishment according to law. |

Table 1: An example of patternized summary generation. The text in red denotes patternized words shared in different summaries, and text in blue denotes specific facts.

Existing prototype based generation models such as (Wu et al., 2018) are all applied on short text, thus, cannot handle long documents summarization task. Another series of works focus on template-based methods such as (Oya et al., 2014). However, template-based methods are too rigid for our patternized summary generation task. Hence, in this paper, we propose a summarization framework named *Prototype Editing based Summary Generator* (PESG) that incorporates prototype document-summary pairs to improve summa-

[*] Equal contribution. Ordering is decided by a coin flip.
[†] Corresponding author.
[1] https://github.com/gsh199449/proto-summ



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3741–3751, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

rization performance when generating summaries with pattern. First, we calculate the cross dependency between the prototype document-summary pair to obtain a summary pattern and prototype facts (explained in § 4.2). Then, we extract facts from the input document with the help of the prototype facts (explained in § 4.3). Next, a recurrent neural network (RNN) based decoder is used to generate a new summary, incorporating both the summary pattern and extracted facts (explained in § 4.4). Finally, a fact checker is designed to provide mutual information between the generated summary and the input document to prevent the generator from copying irrelevant facts from the prototype (explained in § 4.5). To evaluate PESG, we collect a large-scale court judgment dataset, where each judgment is a summary of the case description with a patternized style. Extensive experiments conducted on this dataset show that PESG outperforms the state-of-the-art summarization baselines in terms of ROUGE metrics and human evaluations by a large margin.

Our contributions can be summarized as follows:

• We propose to use prototype information to help generate better summaries with patterns.

• Specifically, we propose to generate the summary incorporating the prototype summary pattern and extracted facts from input document.

• We provide mutual information signal for the generator to prevent copying irrelevant facts from the prototype.

• We release a large-scale prototype based summarization dataset that is beneficial for the community.

## 2 Related Work

We detail related work on text summarization and prototype editing.

Text summarization can be classified into extractive and abstractive methods. Extractive methods (Narayan et al., 2018b; Chen et al., 2018) directly select salient sentences from an article to compose a summary. One shortcoming of these models is that they tend to suffer from redundancy. Recently, with the emergence of neural network models for text generation, a vast majority of the literature on summarization (Ma et al., 2018; Zhou et al., 2018; Gao et al., 2019a; Chen et al., 2019) is dedicated to abstractive summarization, which aims to generate new content that concisely paraphrases a document from scratch.

Another line of research focuses on prototype editing. (Guu et al., 2018) proposed the first prototype editing model, which samples a prototype sentence from training data and then edits it into a new sentence. Following this work, (Wu et al., 2018) proposed a new paradigm for response generation, which first retrieves a prototype response from a pre-defined index and then edits the prototype response. (Cao et al., 2018) applied this method on summarization, where they employed existing summaries as soft templates to generate new summary without modeling the dependency between the prototype document, summary and input document. Different from these soft attention methods, (Cai et al., 2018) proposed a hard-editing skeleton-based model to promote the coherence of generated stories. Template-based summarization is also a hard-editing method (Oya et al., 2014), where a multi-sentence fusion algorithm is extended in order to generate summary templates.

Different from all above works, our model focuses on patternized summary generation, which is more challenging than traditional news summarization and short sentence prototype editing.

## 3 Problem Formulation

For an input document $X = \{x_1, x_2, \ldots, x_{T_m}\}$, we assume there is a ground truth summary $Y = \{y_1, y_2, \ldots, y_{T_n}\}$. In our prototype summarization task, a retrieved prototype document $\hat{X} = \{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_{T_m}\}$ with a corresponding prototype summary $\hat{Y} = \{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{T_n}\}$ is also attached according to their similarities with $X$.

For a given document $X$, our model extracts salient facts from $X$ guided by a prototype document $\hat{X}$, and then generates the summary $Y^{'}$ by referring to the prototype summary $\hat{Y}$. The goal is to generate a summary $Y^{'}$ that not only follows a patternized style (as defined by prototype summary $\hat{Y}$) but also is consistent with the facts in document $X$.

## 4 Model

### 4.1 Overview

In this section, we propose our *prototype editing based summary generator*, which can be split into two main parts, as shown in Figure 1:

• *Summary Generator.* (1) **Prototype Reader** analyzes the dependency between $\hat{X}$ and $\hat{Y}$ to de-

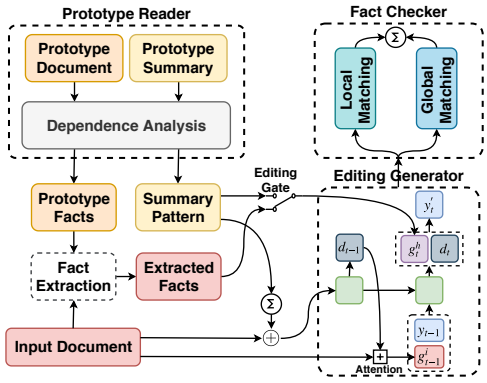


Figure 1: Overview of PESG. We divide our model into four parts: (1) *Prototype Reader*; (2) *Fact Extraction*; (3) *Editing Generator*; (4) *Fact Checker*.

termine the summary pattern and prototype facts. (2) **Fact Extraction** module extracts facts from the input document under the guidance of the prototype facts. (3) **Editing Generator** module generates the summary $Y^{'}$ of document $X$ by incorporating summary pattern and facts.

• *Fact Checker* estimates the mutual information between the generated summary $Y^{'}$ and input document $X$. This information provides an additional signal for the generation process, preventing irrelevant facts from being copied from the prototype document.

### 4.2 Prototype Reader

To begin with, we use an embedding matrix $e$ to map a one-hot representation of each word in $X$, $\hat{X}$, $\hat{Y}$ into a high-dimensional vector space. We then employ a bi-directional recurrent neural network (Bi-RNN) to model the temporal interactions between words:

$$h_t^x = \text{Bi-RNN}_x(e(x_t), h_{t-1}^x), \quad (1)$$

$$\hat{h}_t^x = \text{Bi-RNN}_x(e(\hat{x}_t), \hat{h}_{t-1}^x), \quad (2)$$

$$\hat{h}_t^y = \text{Bi-RNN}_y(e(\hat{y}_t), \hat{h}_{t-1}^y), \quad (3)$$

where $h_t^x$, $\hat{h}_t^x$ and $\hat{h}_t^y$ denote the hidden state of $t$-th step in Bi-RNN for $X$, $\hat{X}$ and $\hat{Y}$, respectively. Following (Tao et al., 2018; Gao et al., 2019b; Hu et al., 2019), we choose long short-term memory (LSTM) as the cell for Bi-RNN.

On one hand, the sections in the prototype summary that are not highly related to the prototype document are the universal patternized words and should be emphasized when generating the new summary. On the other hand, the sections in the prototype document that are highly related to the prototype summary are useful facts that can guide the process of extracting facts from input document. Hence, we employ a bi-directional attention mechanism between a prototype document and summary to analyze the cross-dependency, that is, from document to summary and from summary to document. Both of these attentions are derived from a shared similarity matrix, $S \in \mathbb{R}^{T_m \times T_n}$, calculated by the hidden states of prototype document $\hat{X}$ and prototype summary $\hat{Y}$. $S_{ij}$ indicates the similarity between the $i$-th document word $\hat{x}_i$ and $j$-th summary word $\hat{y}_j$ and is computed as:

$$\begin{aligned} S_{ij} &= \alpha(\hat{h}_i^x, \hat{h}_j^y), \\ \alpha(x, y) &= w^\intercal[x \oplus y \oplus (x \otimes y)], \end{aligned} \quad (4)$$

where $\alpha$ is a trainable scalar function that calculates the similarity between two input vectors. $\oplus$ denotes a concatenation operation and $\otimes$ is an element-wise multiplication.

We use $a_t^s = \text{mean}(S_{:t}) \in \mathbb{R}$ to represent the attention weight on the $t$-th prototype summary word by document words, which will learn to assign high weights to highly related universal patternized words when generating a summary. From $a_t^s$, we obtain the weighted sum of the hidden states of prototype summary as "**summary pattern**" $l = \{l_1, \ldots, l_{T_n}\}$, where $l_i$ is:

$$l_i = a_i^s \hat{h}_i^y. \quad (5)$$

Similarly, $a_t^d = \text{mean}(S_{t:}) \in \mathbb{R}$ assigns high weights to the words in a prototype document that are relevant to the prototype summary. A convolutional layer is then applied to extract "**prototype facts**" $\hat{r}_t$ from the prototype document:

$$\hat{r}_t = \text{CNN}(a_t^d \hat{h}_t^x). \quad (6)$$

We sum the prototype facts to obtain the overall representation of these facts:

$$q = \textstyle\sum_t^{T_m} \hat{r}_t. \quad (7)$$

### 4.3 Fact Extraction

In this section, we discuss how to extract useful facts from an input document with the help of prototype facts.

We first extract the facts from an input document by calculating their relevance to prototype facts. The similarity matrix $E$ is then calculated between the weighted prototype document $a_i^d \hat{h}_i^x$ and input document representation $h_j^x$:

$$E_{ij} = \alpha(a_i^d \hat{h}_i^x, h_j^x), \quad (8)$$

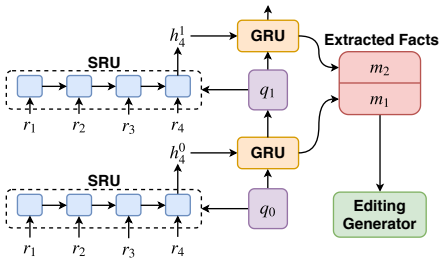


Figure 2: Framework of fact extraction module.

where $\alpha$ is the similarity function introduced in Equation 4. Then, we sum up $E_{ij}$ along the length of the prototype document to obtain the weight $E_j = \sum_{ti}^{T_m} E_{tj}$ for $j$-th word in the document. Next, similar to Equation 6, a convolutional layer is applied on the weighted hidden states $E_t h_t^x$ to obtain the fact representation $r_t$ from the input document:

$$r_t = \text{CNN}(E_t h_t^x). \tag{9}$$

Inspired by the polishing strategy in extractive summarization (Chen et al., 2018), we propose to use the prototype facts to polish the extracted facts $r_t$ and obtain the final fact representation $m_\cdot$, as shown in Figure 2. Generally, the polishing process consists of two hierarchical recurrent layers. The first recurrent layer is made up of Selective Recurrent Units (SRUs), which take facts $r_\cdot$ and polished fact $q_k$ as input, outputting the hidden state $h_{T_m}^k$. The second recurrent layer consists of regular Gated Recurrent Units (GRUs), which are used to update the polished fact from $q_k$ to $q_{k+1}$ using $h_{T_m}^k$.

SRU is a modified version of the original GRU introduced in (Chen et al., 2018), details of which can be found in Appendix A. Its difference from GRU lies in that the update gate in SRU is decided by both the polished fact $q_k$ and original fact $r_t$ together. The $t$-th hidden state of SRU is calculated as:

$$h_t^k = \text{SRU}(r_t, q_k). \tag{10}$$

We take $h_{T_m}^k$ as the overall representation of all input facts $r_\cdot$. In this way, SRU can decide to which degree each unit should be updated based on its relationship with the polished fact $q_k$.

Next, $h_{T_m}^k$ is used to update the polished fact $q_k$ using the second recurrent layer, consisting of GRUs:

$$m_{k+1}, q_{k+1} = \text{GRU}(h_{T_m}^k, q_k), \tag{11}$$

where $q_k$ is the cell state, $h_{T_m}^k$ is the input and $m_{k+1}$ is the output hidden state. $q_0$ is initialized using $q$ in Equation 7. This iterative process is conducted $K$ times, and each output $m_k$ is stored as **extracted facts** $M = \{m_1, m_2, \ldots, m_K\}$. In this way, $M$ stores facts with different polished levels.

### 4.4 Editing Generator

The editing generator aims to generate a summary based on the input document, prototype summary and extracted facts. As with the settings of prototype reader, we use LSTM as the RNN cell. We first apply a linear transformation on the summation of the summary pattern $l' = \sum_i^{T_n} l_i$ and input document representations $h_{T_m}^x$, and then employ this vector as the initial state $d_0$ of the RNN generator as shown in Equation 12. The procedure of $t$-th generation is shown in Equation 13:

$$d_0 = W_e[h_{T_m}^x \oplus l'] + b_e, \tag{12}$$

$$d_t = \text{LSTM}(d_{t-1}, [g_{t-1}^i \oplus e(y_{t-1})]), \tag{13}$$

where $W_e, b_e$ are trainable parameters, $d_t$ is the hidden state of the $t$-th generating step, and $g_{t-1}^i$ is the context vector produced by the standard attention mechanism (Bahdanau et al., 2015).

To take advantage of the extracted facts $M$ and prototype summary $l$, we incorporate them both into summary generation using a dynamic attention. More specifically, we utilize a matching function $f$ to model the relationship between the current decoding state $d_t$ and each $v_i$ ($v_i$ can be a extracted fact $m_i$ or summary pattern $l_i$):

$$\delta_{it} = \frac{\exp(f(v_i, d_t))}{\sum_j^K \exp(f(v_j, d_t))}, \tag{14}$$

$$g_t^* = \sum_i^K \delta_{it} v_i, \tag{15}$$

where $g_t^*$ can be $g_t^m$ or $g_t^s$ for attending to extracted facts or a summary pattern, respectively. We use a simple but efficient bi-linear layer as the matching function $f = m_i W_f d_t$. As for combining $g_t^m$ and $g_t^s$, we propose to use an "editing gate" $\gamma$, which is determined by the decoder state $d_t$, to decide the importance of the summary pattern and extracted facts at each decoding step.

$$\gamma = \sigma\left(W_g d_t + b_g\right), \tag{16}$$

where $\sigma$ denotes the sigmoid function. Using the editing gate, we obtain $g_t^h$ which dynamically

combines information from the extracted facts and summary pattern with the editing gate $\gamma$, as:

$$g_t^h = [\gamma g_t^m \oplus (1-\gamma)\, g_t^s]\,. \tag{17}$$

Finally, the context vector $g_t^h$ is concatenated with the decoder state $d_t$ and fed into a linear layer to obtain the generated word distribution $P_v$:

$$d_t^o = W_o[d_t \oplus g_t^h] + b_o, \tag{18}$$

$$P_v = \text{softmax}\left(W_v d_t^o + b_v\right). \tag{19}$$

The loss is the negative log likelihood of the target word $y_t$:

$$\mathcal{L}_s = -\sum_{t=1}^{T_n} \log P_v(y_t). \tag{20}$$

In order to handle the out-of-vocabulary (OOV) problem, we equip our decoder with a pointer network (Gu et al., 2016; Vinyals et al., 2015; See et al., 2017). This process is the same as the model described in (See et al., 2017), thus, is omit here due to limited space.

What's more, previous work (Holtzman et al., 2018) has found that using a cross entropy loss alone is not enough for generating coherent text. Similarly, in our task, using $\mathcal{L}_s$ alone is not enough to distinguish a good summary with accurate facts from a bad summary with detailed facts from the prototype document (see § 6.2). Thus, we propose a fact checker to determine whether the generated summary is highly related to the input document.

### 4.5 Fact Checker

To generate accurate summaries that are consistent with the detailed facts from the input document rather than facts from the prototype document, we add a fact checker to provide additional training signals for the generator. Following (Hjelm et al., 2018), we employ the neural mutual information estimator to estimate the mutual information between the generated summary $Y^{'}$ and its corresponding document $X$, as well as the prototype document $\hat{X}$. Generally, mutual information is estimated from a local and global level, and we expect the matching degree to be higher between the generated summary and input document than the prototype document. An overview of the fact checker is shown in Figure 3.

To begin, we use a local matching network to calculate the matching degree, for local features, between the generated summary and the input, as well as prototype document. Remember that, in

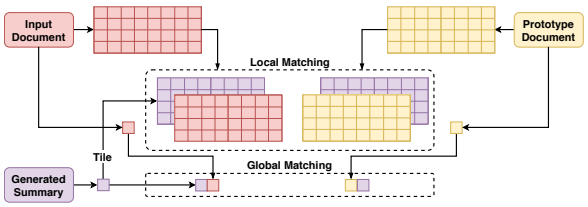


Figure 3: Framework of fact checker module.

§ 4.3, we obtain the fact representation of an input document $r_i$ and prototype facts $\hat{r}_i$. Combining these with the final hidden state $d_{T_n}$ of the generator RNN (in Equation 13), yields the local features of input extracted facts and the prototype facts:

$$C^r = \{d_{T_n} \oplus r_1, \ldots, d_{T_n} \oplus r_{T_m}\}, \tag{21}$$

$$C^f = \{d_{T_n} \oplus \hat{r}_1, \ldots, d_{T_n} \oplus \hat{r}_{T_m}\}. \tag{22}$$

A $1 \times 1$ convolutional layer and a fully-connected layer are applied to score these two features:

$$\tau_l^r = \text{CNN}_l(C^r), \quad \tau_l^f = \text{CNN}_l(C^f), \tag{23}$$

where $\tau_l^r \in \mathbb{R}$, $\tau_l^f \in \mathbb{R}$ represent the local matching degree between the generated summary and input document and prototype document, respectively. We want the generated summary to be more similar to the input document than the prototype document. Thus, the optimization objective of the local matching network is to minimize $\mathcal{L}_l$:

$$\mathcal{L}_l = -\left(\log(\tau_l^r) + \log(1-\tau_l^f)\right). \tag{24}$$

We also have a global matching network to measure the matching degree, for global features, between the generated summary and the input document, as well as prototype document. To do so, we concatenate the representation of the generated summary with the final hidden state of the input document $h_{T_m}^x$ and final state of the prototype document $\hat{h}_{T_m}^x$, respectively, and apply a linear layer to these:

$$\tau_g^r = \text{relu}(W_m[d_{T_n} \oplus h_{T_m}^x] + b_m), \tag{25}$$

$$\tau_g^f = \text{relu}(W_m[d_{T_n} \oplus \hat{h}_{T_m}^x] + b_m), \tag{26}$$

where $W_m, b_m$ are trainable parameters and $\tau_g^r \in \mathbb{R}$ and $\tau_g^f \in \mathbb{R}$ represent the matching degree between the generated summary and the input document, and prototype document, respectively. The objective of this global matching network, similar to the local matching network, is to minimize:

$$\mathcal{L}_g = -\left(\log(\tau_g^r) + \log(1-\tau_g^f)\right). \tag{27}$$

| Acronym | Gloss |
|---|---|
| PESG-FC | PESG w/o Fact Checker |
| PESG-PR | PESG w/o Prototype Reader |
| PESG-SS | PESG w/o Summary Pattern |
| PESG-FG | PESG w/o FGRU |

Table 2: Ablation models for comparison.

Finally, we combine the local and global loss functions to obtain the final loss $\mathcal{L}$, which we use $\mathcal{L}$ to calculate the gradients for all parameters:

$$\mathcal{L} = \epsilon\mathcal{L}_g + \eta\mathcal{L}_l + \mathcal{L}_s, \tag{28}$$

where $\epsilon, \eta$ are both hyper parameters. To optimize the trainable parameters, we employ the gradient descent method Adagrad (Duchi et al., 2010) to update all parameters.

## 5 Experimental Setup

### 5.1 Dataset

We collect a large-scale prototype based summarization dataset[2], which contains 2,003,390 court judgment documents. In this dataset, we use a case description as an input document and the court judgment as the summary. The average lengths of the input documents and summaries are 595.15 words and 273.57 words respectively. The percentage of words common to a prototype summary and the reference summary is 80.66%, which confirms the feasibility and necessity of prototype summarization. Following other summarization datasets (Grusky et al., 2018; Kim et al., 2019; Narayan et al., 2018a), we also count the novel n-grams in a summary compared with the n-grams in the original document, and the percentage of novel n-grams are 51.21%, 84.59%, 91.48%, 94.83% for novel 1-grams to 4-grams respectively. The coverage, compression and density (Grusky et al., 2018) are commonly used as metrics to evaluate the abstractness of a summary. For the summaries in our dataset, the coverage percentage is 48.78%, compression is 2.28 and density is 1.31. We anonymize entity tokens into special tags, such as using "PERS" to replace a person's name.

### 5.2 Comparisons

In order to prove the effectiveness of each module of PESG, we conduct several ablation studies, shown in Table 2. We also compare our model with the following baselines: (1) `Lead-3` is a commonly used summarization baseline (Nallapati et al., 2017; See et al., 2017), which selects the first three sentences of document as the summary. (2) `S2S` is a sequence-to-sequence framework with a pointer network, proposed by (See et al., 2017). (3) `Proto` is a context-aware prototype editing dialog response generation model proposed by (Wu et al., 2018). (4) `Re`$^3$`Sum`, proposed by (Cao et al., 2018), uses an IR platform to retrieve proper summaries and extends the seq2seq framework to jointly conduct template-aware summary generation. (5) `Uni-model` was proposed by (Hsu et al., 2018), and is the current state-of-the-art abstractive summarization approach on the CNN/DailyMail dataset. (6) We also directly concatenate the prototype summary with the original document as input for S2S and Uni-model, named as `Concat-S2S` and `Concat-Uni`, respectively.

### 5.3 Evaluation Metrics

For the court judgment dataset, we evaluate standard ROUGE-1, ROUGE-2 and ROUGE-L (Lin, 2004) on full-length F1 following previous works (Nallapati et al., 2017; See et al., 2017; Paulus et al., 2018), where ROUGE-1 (R1), ROUGE-2 (R2), and ROUGE-L (RL) refer to the matches of unigram, bigrams, and the longest common subsequence respectively.

(Schluter, 2017) notes that only using the ROUGE metric to evaluate summarization quality can be misleading. Therefore, we also evaluate our model by human evaluation. Three highly educated participants are asked to score 100 randomly sampled summaries generated by three models: `Uni-model`, `Re`$^3$`Sum` and PESG. The statistical significance of observed differences between the performance of two runs is tested using a two-tailed paired t-test and is denoted using $^{\blacktriangle}$(or $^{\blacktriangledown}$) for strong (or weak) significance for $\alpha = 0.01$.

### 5.4 Implementation Details

We implement our experiments in TensorFlow (Abadi et al., 2016) on an NVIDIA GTX 1080 Ti GPU. The word embedding dimension is 256 and the number of hidden units is 256. The batch size is set to 64. We padded or cut input document to contain exactly 250 words, and the decoding length is set to 100. $\epsilon$ and $\eta$ from the Equation 28 are both set to 1.0. We initialize all of the parameters randomly using a Gaussian distribution. We use Adagrad optimizer (Duchi

[2] https://github.com/gsh199449/proto-summ

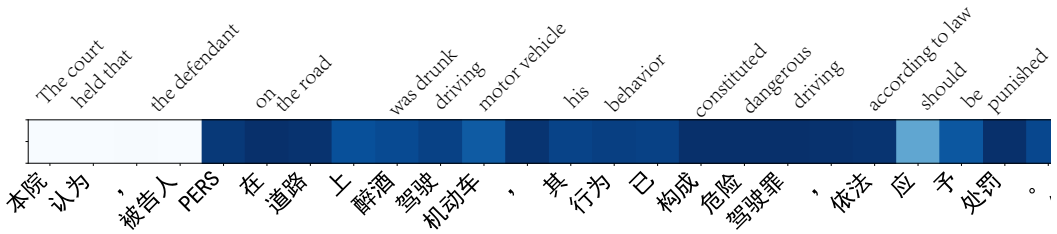


Figure 4: Visualizations of editing gate.

| | R1 | R2 | RL |
|---|---|---|---|
| Lead-3 | 32.9 | 13.3 | 30.0 |
| Re$^3$Sum | 36.3 | 24.0 | 36.0 |
| S2S | 37.6 | 24.6 | 37.3 |
| Uni-model | 37.9 | 25.0 | 37.6 |
| Concat-S2S | 34.2 | 20.3 | 34.3 |
| Concat-Uni | 37.4 | 24.3 | 36.9 |
| PESG | **40.2** | **28.1** | **39.9** |
| Proto | – | – | – |

Table 3: ROUGE scores comparison with baselines. `Proto` directly copies from the prototype summary as generated summary.

| | Fluency | Consistency |
|---|---|---|
| Uni-model | 1.61 | 1.53 |
| Re$^3$Sum | 1.53 | 1.14 |
| PESG | **1.86**$^\blacktriangle$ | **1.73**$^\blacktriangle$ |

Table 4: Fluency and consistency comparison by human evaluation.

et al., 2010) as our optimizing algorithm and employ beam search with size 5 to generate more fluency summary sentence. We also apply gradient clipping (Pascanu et al., 2013) with range $[-5, 5]$ during training. We use dropout (Srivastava et al., 2014) as regularization with keep probability $p = 0.7$.

## 6 Experimental Result

### 6.1 Overall Performance

We compare our model with the baselines listed in Table 3. Our model performs consistently better than other summarization models including the state-of-the-art model with improvements of 6%, 12% and 6% in terms of ROUGE-1, ROUGE-2 and ROUGE-L. This demonstrates that prototype document-summary pair provides strong guidance for summary generation that cannot be replaced by other complicated baselines without prototype information. Meanwhile, directly concatenating the prototype summary with the original input does not increase performance, instead leading to drops of 9%, 17%, 8% and 1%, 3%, 2% in terms of ROUGE 1,2,L on the S2S and Unified models, respectively. As for the baseline model `Proto`, we found that it directly **copies** from the prototype summary as generated summary, which leads to a totally useless and incorrect summary.

For the human evaluation, we asked annotators to rate each summary according to its consistency and fluency. The rating score ranges from 1 to 3, with 3 being the best. Table 4 lists the average scores of each model, showing that PESG outperforms the other baseline models in both fluency and consistency. The kappa statistics are 0.33 and 0.29 for fluency and consistency respectively, and that indicates the moderate agreement between annotators. To prove the significance of these results, we also conduct the paired student t-test between our model and `Re`$^3$`Sum` (row with shaded background). We obtain a p-value of $2 \times 10^{-7}$ and $9 \times 10^{-12}$ for fluency and consistency, respectively.

We also analyze the effectiveness of performance by the two hyper-parameters: $\eta$ and $\epsilon$. It turns out that our model has a consistently good performance, with ROUGE-1, ROUGE-2, ROUGE-L scores above 39.5, 27.5, 39.4, which demonstrates that our model is very robust.

### 6.2 Ablation Study

The ROUGE scores of different ablation models are shown in Table 5. All ablation models perform worse than PESG in terms of all metrics, which demonstrates the preeminence of PESG. More importantly, by this controlled experiment, we can verify the contributions of each modules in PESG.

### 6.3 Analysis of Editing Generator

We visualize the editing gate (illustrated in Equation 16) of two **randomly sampled** examples,

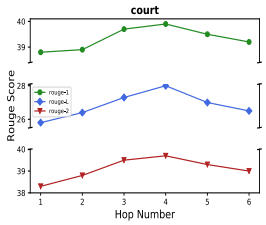


Figure 5: Visualizations of ROUGE score with different hop numbers.

| | R1 | R2 | RL |
|---|---|---|---|
| PESG-FC | 38.7 | 26.2 | 38.6 |
| PESG-PR | 37.3 | 24.6 | 37.0 |
| PESG-SS | 38.6 | 25.6 | 38.3 |
| PESG-FG | 38.8 | 25.9 | 38.5 |

Table 5: ROUGE scores of different ablation models of PESG.

shown in Figure 4. A lower weight (lighter color) means that the word is more likely to be copied from the summary pattern; that is to say, this word is a universal patternized word. We can see that the phrase 本院认为 (the court held that) has a lower weight than the name of the defendant (PERS), which is consistent with the fact that (the court held that) is a patternized word and the name of the defendant is closely related to the input document.

We also show a case study in Table 6, which includes the input document and reference summary with the generated summaries. Underlined text denotes a grammar error and a strike-through line denotes a fact contrary to the input document. We only show part of the document and summary due to limited space; the full version is shown in Appendix. As can be seen, the summary generated by `Uni-model` faces an inconsistency problem and the summary generated by `Re`$^3$`Sum` is contrary to the facts described in the input document. However, PESG overcomes both of these problems and generates an accurate summary with good grammar and logic.

### 6.4 Analysis of Fact Extraction Module

We investigate the influence of the iteration number when facts are extracted. Figure 5 illustrates the relationship between iteration number and the f-value of the ROUGE score. The results show that the ROUGE scores first increases with the number of hops. After reaching an upper limit it then begins to drop. This phenomenon demonstrates that the fact extraction module is effective by polishing the facts representation.

## 7 Conclusion

In this paper, we propose a framework named *Prototype Editing based Summary Generator* (PESG), which aims to generate summaries in formal writing scenarios, where summaries should conform to a patternized style. Given a prototype document-summary pair, our model first calculates the cross dependency between the prototype document-summary pair. Next, a fact extraction module is employed to extract facts from the document, which are then polished. Finally, we design an editing-based generator to produce a summary by incorporating the polished fact and summary pattern. To ensure that the generated summary is consistent with the input document, we propose a fact checker to estimate the mutual information between the input document and generated summary. Our model outperforms state-of-the-art methods in terms of ROUGE scores and human evaluations by a large margin, which demonstrates the effectiveness of PESG.

| | |
|---|---|
| | 安徽省人民法院作出刑事判决，以斗殴罪判处被告人PERS 有期徒刑YEARS 。合肥市监狱提出减刑意见。执行机关以PERS 在服刑期间确有悔改表现，已获得奖励二个，建议对其减刑。经审理查明，罪犯PERS 在服刑期间，能认罪服法，努力参加技术学习。(Anhui Provincial People's Court made a criminal judgment and sentenced the defendant PERS to YEARS imprisonment for the crime of fighting. Hefei Prison proposed a commutation sentence since PERS had repentance and received two awards during his sentence. It was found through trial that the criminal PERS pleaded guilty and strived to participate in technical learning while serving his sentence.) |
| reference | 本院认为，罪犯PERS 在服刑期间确有悔改，表现符合法定减刑条件。依据《中华人民共和国刑法》第NUM 款之规定，对罪犯PERS 减去自本裁定送达之日起至DATE 止尚未执行完毕的刑罚。本裁定送达后即发生法律效力。(The court believed that the criminal PERS did have repentance during his sentence, and his performance was in line with the statutory commutation conditions. According to the provisions of Section NUM of the Criminal Law of the People's Republic of China, the court decided to exempt PERS from penalty that had not been executed since the date of delivery of this ruling until DATE. Legal effect would occur upon the delivery of this ruling.) |
| Uni-model | 本院认为，被告人PERS ~~以非法占有为目的，秘密窃取他人财物，数额较大，其行为已构成盗窃罪~~。公诉机关指控被告人PERS勋斗殴罪的事实清楚，证据确实充分，本院予以支持。(The court held that the defendant PERS ~~secretly stole the property of others for the purpose of illegal possession, the stolen amount was relatively large, and his behavior had constituted theft.~~ The fact that defendant PERS committed the crime of fighting was clear, the evidence was indeed sufficient, and the court supported it.) |
| Re$^3$Sum | 本院认为，PERS ~~在道路上驾驶机动车~~,其行为已构成上诉人服刑确系罪. 公诉机关指控的犯罪事实清楚，证据确实充分。(The court held that PERS was ~~driving a motor vehicle on the road~~, and his behavior constituted that the appellant was guilty of serving a sentence. The criminal facts accused by the public prosecution agency were clear and the evidence was indeed sufficient.) |
| PESG | 本院认为，罪犯PERS 在服刑期间，确有悔改表现。符合法定减刑条件，依照《中华人民共和国刑法》第NUM 条、第NUM 条之规定，裁定如下对罪犯PERS 减去有期徒刑MONTHS。本裁定送达后即发生法律效力。(The court held that the criminal PERS did have repentance during his sentence. In accordance with the provisions of NUM and NUM of the Criminal Law of the People's Republic of China, the ruling was as follows: exempted PERS from the MONTHS penalty. Legal effect would occur upon the delivery of this ruling.) |

Table 6: Examples of the generated natural answers by PESG and other models.

## Acknowledgments

We would like to thank the anonymous reviewers for their constructive comments. We would also

like to thank Anna Hennig in Inception Institute of Artificial Intelligence for her help on this paper. This work was supported by the National Key Research and Development Program of China (No. 2017YFC0804001), the National Science Foundation of China (NSFC No. 61876196 and NSFC No. 61672058)

## A SRU Cell

Gated recurrent unit (GRU) (Cho et al., 2014) is a gating mechanism in recurrent neural networks, which incorporate an update gate in an RNN. We first give the details of the original GRU here.

$$u_i = \sigma(W^{(u)}x_i + U^{(u)}h_{i-1} + b^{(u)}), \quad (29)$$

$$r_i = \sigma(W^{(r)}x_i + U^{(r)}h_{i-1} + b^{(r)}), \quad (30)$$

$$\tilde{h_i} = \tanh(W^{(h)}x_i + r_i \circ Uh_{i-1} + b^{(h)}), \quad (31)$$

$$h_i = u_i \circ \tilde{h_i} + (1 - u_i) \circ h_{i-1}, \quad (32)$$

where $\sigma$ is the sigmoid activation function, $W^{(u)}, W^{(r)}, W^{(h)} \in \mathbb{R}^{n_H \times n_I}, U^{(u)}, U^{(r)}, U \in \mathbb{R}^{n_H \times n_H}$, $n_H$ is the hidden size, and $n_I$ is the size of input $x_i$. In the original version of GRU, the update gate $u_i$ in Equation 29 is used to decide how much of the hidden state should be retained and how much should be updated. In our case, we want to decide which facts are salient according to the polished facts $q_{k-1}$ at the $k$-th hop. To achieve this, we replace the calculation of $u_i$ with the newly computed update gate $g_i$:

$$f_i = [x_i \circ q_{k-1}; x_i; q_{k-1}], \quad (33)$$

$$z_i = W^{(2)} \tanh(W^{(1)}f_i + b^{(1)}) + b^{(2)}, \quad (34)$$

$$g_i = \frac{\exp(z_i)}{\sum_{j=1}^{n_s} \exp(z_j)}, \quad (35)$$

where $W^{(2)}, W^{(1)}, b^{(1)}, b^{(2)}$ are all trainable parameters and $k$ is the hop number in the multi-hop situation which is a hyper-parameter manually set. The effectiveness of this hyper-parameter is verified in the experimental results shown in § 6.4. Equation 32 now becomes:

$$h_i = g_i \circ \tilde{h_i} + (1 - g_i) \circ h_{i-1}. \quad (36)$$

We use the name "SRU" to denote this modified version of an GRU cell.

## B Case Study

| | |
|---|---|
| | 安徽省合肥市包河区人民法院作出刑事判决，以斗殴罪判处被告人PERS 有期徒刑YEARS 。案经本院二审，准许上诉人撤回上诉。判决生效后交付执行。执行机关合肥市义城监狱于DATE 提出减刑意见，报送本院审理。本院依法组成合议庭进行了审理，现已审理终结。执行机关以罪犯PERS 在服刑期间确有悔改表现，已获得奖励二个，建议对其减刑。经审理查明，罪犯PERS 在服刑期间，能认罪服法，努力参加技术学习，积极完成生产任务。(The People's Court of Baohe District, Hefei City, Anhui Province made a criminal judgment and sentenced the defendant PERS to YEARS imprisonment for the crime of fighting. In the second instance of the court, the appellant was allowed to withdraw the appeal. The judgment takes effect after being delivered. The enforcement agency Hefei City Yicheng Prison put forward a commutation opinion on DATE and submitted it to the court for trial. The court had formed a collegiate bench in accordance with the law and had now end the trial. Criminal PERS had repentance during his sentence and received two awards, thus the enforcement agency suggested he should be commuted. Investigation confirmed that the criminal PERS was able to plead guilty while serving his sentence, strived to participate in technical studies, and completed production tasks.) |
| reference | 本院认为，罪犯PERS 在服刑期间确有悔改表现符合法定减刑条件。依据《中华人民共和国刑法》第NUM 款之规定，裁定如下对罪犯PERS 减去自本裁定送达之日起至DATE 止尚未执行完毕的刑罚。本裁定送达后即发生法律效力。(The court held that the criminal PERS did have repentance during the period of serving his sentence in accordance with the statutory commutation conditions. According to the provisions of NUM Section of the Criminal Law of the People's Republic of China, the court decided to exempt PERS from penalty that had not been executed since the date of delivery of this ruling until DATE. Legal effect would occur upon the delivery of this ruling.) |
| Unified | 本院认为，被告人PERS 以非法占有为目的，秘密窃取他人财物，数额较大，其行为已构成盗窃罪，依法应予惩处。公诉机关指控被告人PERS 聚众斗殴罪的事实清楚，证据确实充分，指控罪名成立，本院予以支持。被告人PERS 曾因故意犯罪被判处有期徒刑，在刑罚执行完毕后YEARS 内再犯，应当判处有期徒刑以上刑罚之罪，是累犯，依法应当从重处罚。(The court held that the defendant PERS secretly stole the property of others for the purpose of illegal possession, and the amount was relatively large. His behavior had constituted theft and should be punished according to law. The fact that the defendant PERS commiting the crimes of the fighting was clear, the evidence was indeed sufficient, thus the charges were established and the court supported it. The defendant PERS was sentenced to fixed-term imprisonment for intentional crimes. If he committed another crime in YEARS after the execution of the penalty, he should be sentenced to a fixed-term imprisonment or more. He is a recidivist and should be severely punished according to law.) |
| Re$^3$Sum | 本院认为NUM PERS 在道路上驾驶机动车其行为已构成上诉人服刑确系罪公诉机关指控的犯罪事实清楚证据确实充分指控的罪名成立本院予以支持被告人PERS 归案后如实供述自己的罪行依法予以从轻处罚公诉机关的量刑建议适当本院予以采纳此依照《中华人民共和国刑法》第NUM 款之规定判决如下被告人PERS 犯服刑判处有期徒刑MONTHS并处罚金人民币MONEY (The court held that NUM PERS was driving a motor vehicle on the road, and his behavior constituted that the appellant was guilty of serving a sentence. The criminal facts accused by the public prosecution agency were clear, the evidence was indeed sufficient, and the charges were supported by the court. After confessing his crimes, he should be given a lighter punishment according to law. The public prosecution agency's sentencing recommendations were appropriate and adopted by the court. According to the provisions of Section NUM of the Criminal Law of the People's Republic of China, the judgment was as follows, the defendant PERS was sentenced to imprisonment and sentenced to fixed-term MONTHS imprisonment, and fined the penalty RMB MONEY.) |
| PESG | 本院认为，罪犯PERS 在服刑期间，确有悔改表现。符合法定减刑条件，依照《中华人民共和国刑法》第NUM 条、第NUM 条之规定，裁定如下：对罪犯PERS 减去有期徒刑MONTHS。本裁定送达后即发生法律效力。(The court believed that the criminal PERS did have repentance during his sentence. In accordance with the statutory commutation conditions and NUM and NUM of the Criminal Law of the People's Republic of China, the ruling was as follows: exempted PERS from the MONTHS penalty. Legal effect would occur upon the delivery of this ruling.) |

Table 7: Examples of the generated natural answers by PESG and other models.

## References


Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Deng Cai, Yan Wang, Victoria Bi, Zhaopeng Tu, Xiaojiang Liu, Wai Lam, and Shuming Shi. 2018. Skeleton-to-response: Dialogue generation guided by retrieval memory. *arXiv preprint arXiv:1809.05296*.

Ziqiang Cao, Wenjie Li, Sujian Li, and Furu Wei. 2018. Retrieve, rerank and rewrite: Soft template based neural summarization. In *ACL*, pages 152–161. Association for Computational Linguistics.

Xiuying Chen, Zhangming Chan, Shen Gao, Meng-Hsuan Yu, Dongyan Zhao, and Rui Yan. 2019. Learning towards abstractive timeline summarization. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 4939–4945. International Joint Conferences on Artificial Intelligence Organization.

Xiuying Chen, Shen Gao, Chongyang Tao, Yan Song, Dongyan Zhao, and Rui Yan. 2018. Iterative document representation learning towards summarization with polishing. In *EMNLP*, pages 4088–4097.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*.

John C. Duchi, Elad Hazan, and Yoram Singer. 2010. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159.

Shen Gao, Xiuying Chen, Piji Li, Zhaochun Ren, Lidong Bing, Dongyan Zhao, and Rui Yan. 2019a. Abstractive text summarization by incorporating reader comments. In *AAAI*.

Shen Gao, Zhaochun Ren, Yihong Eric Zhao, Dongyan Zhao, Dawei Yin, and Rui Yan. 2019b. Product-aware answer generation in e-commerce question-answering. In *WSDM*.

Max Grusky, Mor Naaman, and Yoav Artzi. 2018. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In *NAACL-HLT*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *CoRR*, abs/1603.06393.

Kelvin Guu, Tatsunori B. Hashimoto, Yonatan Oren, and Percy Liang. 2018. Generating sentences by editing prototypes. *TACL*, 6:437–450.

R. Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Adam Trischler, and Yoshua Bengio. 2018. Learning deep representations by mutual information estimation and maximization. *CoRR*, abs/1808.06670.

Ari Holtzman, Jan Buys, Maxwell Forbes, Antoine Bosselut, David Golub, and Yejin Choi. 2018. Learning to write with cooperative discriminators. In *ACL*.

Wan-Ting Hsu, Chieh-Kai Lin, Ming-Ying Lee, Kerui Min, Jing Tang, and Min Sun. 2018. A unified model for extractive and abstractive summarization using inconsistency loss. In *ACL*.

Wenpeng Hu, Zhangming Chan, Bing Liu, Dongyan Zhao, Jinwen Ma, and Rui Yan. 2019. Gsn: A graph-structured network for multi-party dialogues. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 5010–5016. International Joint Conferences on Artificial Intelligence Organization.

Byeongchang Kim, Hyunwoo Kim, and Gunhee Kim. 2019. Abstractive Summarization of Reddit Posts with Multi-level Memory Networks. In *NAACL-HLT*.

Piji Li, Wai Lam, Lidong Bing, and Zihao Wang. 2017. Deep recurrent generative decoder for abstractive text summarization. In *EMNLP*, pages 2091–2100.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Shuming Ma, Xu Sun, Junyang Lin, and Houfeng Wang. 2018. Autoencoder as assistant supervisor: Improving text representation for chinese social media text summarization. In *ACL*.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *AAAI*.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018a. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *EMNLP*.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018b. Ranking sentences for extractive summarization with reinforcement learning. In *NAACL-HLT*.

Tatsuro Oya, Yashar Mehdad, Giuseppe Carenini, and Raymond Ng. 2014. A template-based abstractive meeting summarization: Leveraging summary and source text relationships. In *INLG*, pages 45–53.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *ICML*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *ICLR*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *EMNLP*, pages 379–389.

Natalie Schluter. 2017. The limits of automatic summarisation according to rouge. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 41–45. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *ACL*.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Chongyang Tao, Shen Gao, Mingyue Shang, Wei Chung Wu, Dongyan Zhao, and Rui Yan. 2018. Get the point of my utterance! learning towards effective responses with multi-head attention mechanism. In *IJCAI*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *NIPS*, pages 2692–2700.

Yu Ping Wu, Furu Wei, Shaohan Huang, Zhoujun Li, and Ming Zhou. 2018. Response generation by context-aware prototype editing. *CoRR*, abs/1806.07042.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2018. Sequential copying networks. In *AAAI*.

# *BottleSum*: Unsupervised and Self-supervised Sentence Summarization using the Information Bottleneck Principle

**Peter West**[1] **Ari Holtzman**[1,2] **Jan Buys**[1] **Yejin Choi**[1,2]
[1]Paul G. Allen School of Computer Science & Engineering, University of Washington
[2]Allen Institute for Artificial Intelligence
{pawest, ahai, jbuys, yejin}@cs.washington.edu

## Abstract

The principle of the Information Bottleneck (Tishby et al., 1999) is to produce a summary of information $X$ optimized to predict some other relevant information $Y$. In this paper, we propose a novel approach to unsupervised sentence summarization by mapping the Information Bottleneck principle to a conditional language modelling objective: given a sentence, our approach seeks a compressed sentence that can best predict the *next* sentence. Our iterative algorithm under the Information Bottleneck objective searches gradually shorter subsequences of the given sentence while maximizing the probability of the next sentence conditioned on the summary. Using only pretrained language models with no direct supervision, our approach can efficiently perform extractive sentence summarization over a large corpus.

Building on our *unsupervised extractive* summarization (*BottleSum*$^{Ex}$), we then present a new approach to *self-supervised abstractive* summarization (*BottleSum*$^{Self}$), where a transformer-based language model is trained on the output summaries of our unsupervised method. Empirical results demonstrate that our extractive method outperforms other unsupervised models on multiple automatic metrics. In addition, we find that our self-supervised abstractive model outperforms unsupervised baselines (including our own) by human evaluation along multiple attributes.

## 1 Introduction

Recent approaches based on neural networks have brought significant advancements for both extractive and abstractive summarization (Rush et al., 2015; Nallapati et al., 2016). However, their success relies on large-scale parallel corpora of input text and output summaries for direct supervision. For example, there are ~280,000 training instances in the CNN/Daily Mail dataset (Nallapati et al., 2016; Hermann et al., 2015), and ~4,000,000 instances in the sentence summarization dataset of Rush et al. (2015). Because it is too costly to have humans write gold summaries at this scale, existing large-scale datasets are based on naturally occurring pairs of summary-like text paired with source text, for instance using news titles or highlights as summaries for news-text. A major drawback to this approach is that these pairs must already exist in-domain, which is often not true.

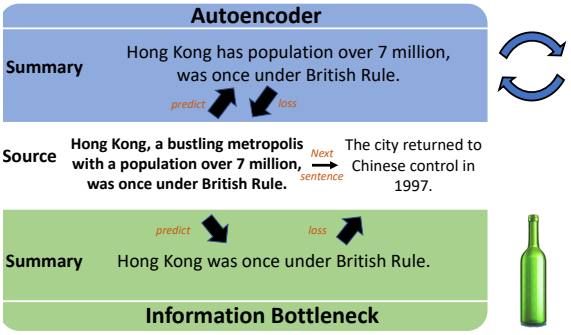


Figure 1: Example contrasting the Autoencoder (AE) and Information Bottleneck (IB) approaches to summarization. While AE (top) preserves any detail that helps to reconstruct the original, such as population size in this example, IB (bottom) uses context to determine which information is relevant, which results in a more appropriate summary.

The sample inefficiency of current neural approaches limits their impact across different tasks and domains, motivating the need for unsupervised or self-supervised alternatives (Artetxe et al., 2017; LeCun, 2018; Schmidhuber, 1990). Further, for summarization in particular, the current paradigm requiring millions of supervision examples is almost counter-intuitive; after all, humans don't need to see a million summaries to know how to summarize, or what information to include.

In this paper, we present *BottleSum*, consisting



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3752–3761, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

of a pair of novel approaches, *BottleSum*$^{Ex}$ and *BottleSum*$^{Self}$ for *unsupervised extractive* and *self-supervised abstractive* summarization, respectively. Core to our approach is the principle of the Information Bottleneck (Tishby et al., 1999), producing a summary for information X optimized to predict some other relevant information Y. In particular, we map (conditional) language modeling objectives to the Information Bottleneck principle to guide the unsupervised model on what to keep and what to discard.

The key intuition of our bottleneck-based summarization is that a good sentence summary contains information related to the broader context while discarding less significant details. Figure 1 demonstrates this intuition. Given input sentence "Hong Kong, a bustling metropolis with a population over 7 million, ...", which is followed by the next sentence "The city returned to Chinese control in 1997", the information bottleneck would suggest that minute details such as the city's population being over 7 million are relatively less important to keep. In contrast, the continued discussion of the city's governance in the next sentence suggests its former British rule is important here.

This intuition contrasts with that of autoencoder-based approaches where the goal is to minimize the reconstruction loss of the input sentence when constructing the summary (Miao and Blunsom, 2016; Wang and Lee, 2018; Fevry and Phang, 2018; Baziotis et al., 2019). Under the reconstruction loss, minute but specific details such as the city's population being over 7 million will be difficult to discard from the summary, because they are useful for reconstruction.

Concretely, *BottleSum*$^{Ex}$ is an extractive and unsupervised sentence summarization method using the next sentence, a sample of nearby context, as guidance to *relevance*, or what information to keep. We capture this with a conditional language modelling objective, allowing us to benefit from powerful deep neural language models that are pre-trained over an extremely large-scale corpus. Under the Information Bottleneck objective, we present an iterative algorithm that searches gradually shorter subsequences of the source sentence while maximizing the probability of the next sentence conditioned on the summary. The benefit of this approach is that it requires no domain-specific supervision or fine-turning.

Building on our unsupervised extractive summarization, we then present *BottleSum*$^{Self}$, a new approach to self-supervised abstractive summarization. This method also uses a pretrained language model, but turns it into an abstractive summarizer by fine-tuning on the output summaries generated by *BottleSum*$^{Ex}$ paired with their original input sentences. The goal is to generalize the summaries generated by an extractive method by training a language model on them, which can then produce abstractive summaries as its generation is not constrained to be extractive.

Together, *BottleSum*$^{Ex}$ and *BottleSum*$^{Self}$ are *BottleSum* methods for unsupervised sentence summarization. Empirical results demonstrate that *BottleSum*$^{Ex}$ outperforms other unsupervised methods on multiple automatic metrics, closely followed by *BottleSum*$^{Self}$. Furthermore, testing on a large unsupervised corpus, we find *BottleSum*$^{Self}$ outperforms unsupervised baselines (including our own *BottleSum*$^{Ex}$) on human evaluation along multiple attributes.

## 2 The Information Bottleneck Principle

Unsupervised summarization requires formulating an appropriate learning objective that can be optimized without supervision (example summaries). Recent work has treated unsupervised summarization as an autoencoding problem with a reconstruction loss (Miao and Blunsom, 2016; Baziotis et al., 2019). The goal is then to produce a compressed summary from which the source sentence can be accurately predicted, i.e. to maximize:

$$\mathbb{E}_{p(\tilde{s}|s)} \ \log p(s|\tilde{s}), \tag{1}$$

where $s$ is the source sentence, $\tilde{s}$ is the generated summary and $p(\tilde{s}|s)$ the learned summarization model. The exact form of this loss may be more elaborate depending on the system, for example including an auxiliary language modeling loss, but the main aim is to produce a summary from which the source can be reconstructed.

The intuitive limitation of this approach is that it will always prefer to retain all informative content from the source. This goes against the fundamental goal of summarization, which crucially needs to forget all but the "relevant" information. It should be detrimental to keep tangential information, as illustrated by the example in Figure 1. As a result, autoencoding systems need to introduce additional loss terms to augment the reconstruction loss (e.g. length penalty, or the topic loss

of Baziotis et al. (2019)).

The premise of our work is that the Information Bottleneck (IB) principle (Tishby et al., 1999) is a more natural fit for summarization. Unlike reconstruction loss, which requires augmentative terms to summarize, IB naturally incorporates a tradeoff between information selection and pruning. These approaches are compared directly in section 5.

At its core, IB is concerned with the problem of maximal compression while defining a formal notion of information relevance. This is introduced with an external variable $Y$. The key is that $\tilde{S}$, the summary of source $S$, contains only information useful for predicting $Y$. This can be posed formally as learning a conditional distribution $p(\tilde{S}|S)$ minimizing:

$$I(\tilde{S};S) - \beta I(\tilde{S};Y), \tag{2}$$

where $I$ denotes mutual information between these variables.

A notion of information relevance comes from the second term, the *relevance term*: with a positive coefficient $\beta$, this is encouraging summaries $\tilde{S}$ to contain information shared with $Y$. The first term, or *pruning term*, ensures that irrelevant information is discarded. By minimizing the mutual information between summary $\tilde{S}$ and source $S$, any information about the source that is not credited by the *relevance term* is thrown away. The statistical structure of IB makes this compressive by forcing the summary to only contain information shared with the source.[1]

In sum, IB relies on 3 principles:

1. Encouraging relevant information with a *relevance term*.

2. Discouraging extra information with a *pruning term*.

3. Strictly summarizing the source.

To clarify the difference from a reconstructive loss, suppose there is irrelevant information in $S$ (i.e. unrelated to relevance variable $Y$), call this $Z$. With the IB objective (eq 3), there is no benefit to keeping any information from $Z$, which strictly makes the first term worse (more mutual information between source and summary) and does not affect the second ($Z$ is unrelated to $Y$). In contrast, because $Z$ contains information about $S$, including it in $\tilde{S}$ could easily benefit the reconstructive loss (eq. 1) despite being irrelevant.

As a relevance variable we will use the sentence following the source in the document in which it occurs. This choice is motivated by linguistic cohesion, in which we expect more broadly relevant information to be common between consecutive sentences, while less relevant information and details are often not carried forward.

We use these principles to derive two methods for sentence summarization. Our first method (§3) enforces strict summarization through being extractive. Additionally, it does not require any training, so can be applied directly without the availability of domain-specific data. The second method (§4) generalizes IB-based summarization to *abstractive* summarization that can be trained on large unsupervised datasets, learning an explicit summarization function $p(\tilde{s}|s)$ over a distribution of inputs.

## 3 Unsupervised Extractive Summarization

We now use the Information Bottleneck principle to propose *BottleSum*$^{Ex}$, an unsupervised extractive approach to sentence summarization. Our approach does not require any training; only a pretrained language model is required to satisfy the IB principles of (2), and the stronger the language model, the stronger our approach will be. In section 5 we demonstrate the effectiveness of this method using GPT-2, the pretrained language model of Radford et al. (2019). [2]

### 3.1 IB for Extractive Summarization

Here, we take advantage of the natural parallel between the Information Bottleneck and summarization developed in section 2. Working from the 3 IB principles stated there, we derive a set of actionable principles for a concrete sentence summarization method.

We approach the task of summarizing a single sentence $s$ using the following sentence $s_{next}$ as the relevance variable. The method will be a deterministic function mapping $s$ to the summary $\tilde{s}$, so instead of learning a distribution over summaries, we take $p(\tilde{s}|s) = 1$ for the summary we arrive at. Our goal is then to optimize the IB equation (Eq 2)

[1] In IB, this is a strict statistical relationship.

[2] We use the originally released "small" 117M parameter version.

for a single example rather a distribution of inputs (as in the original IB method).

In this setting, to minimize equation 2 we can equivalently minimize:

$$-\log p(\tilde{s}) - \beta_1 p(s_{next}|\tilde{s}) p(\tilde{s}) \log p(s_{next}|\tilde{s}), \quad (3)$$

where coefficient $\beta_1 > 0$ controls the trade-off between keeping relevant information and pruning. See appendix A for the derivation of this equation. Similar to eq 2, the first term encourages pruning, while the second encourages information about the relevance variable, $s_{next}$. Both unique values in eq 3 ($p(\tilde{s})$ and $p(s_{next}|\tilde{s})$) can be estimated directly by a pretrained language model, a result of the summary being natural language as well as our choice of relevance variable. This will give us a direct path to enforcing IB principles 1 and 2 from section 2.

To interpret principle 3 for text, we consider what attributes are important to strict textual summarization. Simply, a strict textual summary should be shorter than the source, while agreeing semantically. The first condition is straightforward but the second is currently infeasible to ensure with automatic systems, and so we instead enforce extractive summarization to ensure the first and encourage the second.

Without a supervised validation set, there is no clear way to select a value for $\beta_1$ in Eq 3 and so no way to optimize this directly. Instead, we opt to ensure both terms improve as our method proceeds. Thus, we are not comparing the pruning and relevance terms directly (only ensuring mutual progress), and so we optimize simpler quantities monotonic in the two terms instead: $p(\tilde{s})$ for pruning and $p(y|\tilde{s})$ for relevance.

We perform extractive summarization by iteratively deleting words or phrases, starting with the original sentence. At each elimination step, we only consider candidate deletions which decrease the value of the pruning term, i.e., increase the language model score of the candidate summary. This ensures progress on the pruning term, and also enforces the notion that word deletion should reduce the information content of the summary. The relevance term is optimized through only expanding candidates that have the highest relevance scores at each iteration, and picking the candidate with the highest relevance score as final summary.

Altogether, this gives 3 principles for extractive summarization with IB.

1. Maximize *relevance term* by maximizing $p(s_{next}|\tilde{s})$.
2. Prune information and enforce compression by bounding: $p(\tilde{s}_{i+1}) > p(\tilde{s}_i)$.
3. Enforce strict summarization by extractive word elimination.

### 3.2 Method

**Algorithm 1** *BottleSum*$^{Ex}$ method

**Require:** sentence $s$ and context $s_{next}$

1: $C \leftarrow \{s\}$ ▷ set of summary candidates
2: **for** $l$ in $length(s)...1$ **do**
3: $C_l \leftarrow \{s' \in C | len(s') = l\}$
4: sort $C_l$ descending by $p(s_{next}|s')$
5: **for** $s'$ in $C_l[1:k]$ **do**
6: $l' \leftarrow length(s')$
7: **for** $j$ in $1...m$ **do**
8: **for** $i$ in $1...(l'-j)$ **do**
9: $s'' \leftarrow s'[1:i-1] \circ s'[i+j:l']$
10: **if** $p(s'') > p(s')$ **then**
11: $C \leftarrow C + \{s''\}$
12: **return** $\arg\max_{s' \in C} p(s_{next}|s')$

We turn these principles into a concrete method which iteratively produces summaries of decreasing length by deleting consecutive words in candidate summaries (Algorithm 1). The relevance term is optimized in two ways: first, only the top-scoring summaries of each length are used to generate new, shorter summaries (line 5). Second, the final summary is chosen explicitly by this measure (line 12).

In order to satisfy the second condition, each candidate must contain less self-information (i.e., have higher probability) than the candidate that derives it. This ensures that each deletion (line 9) strictly removes information. The third condition, strict extractiveness, is satisfied per definition.

The algorithm has two parameters: $m$ is the max number of consecutive words to delete when producing new summary candidates (line 9), and $k$ is the number of candidates at each length used to generate shorter candidates by deletion (line 5).

## 4 Abstractive Summarization with Extractive Self-Supervision

Next, we extend the unsupervised summarization of *BottleSum*$^{Ex}$ to abstractive summarization with

*BottleSum*$^{Self}$, based on a straightforward technique for self-supervision. Simply, a large corpus of unsupervised summaries is generated with *BottleSum*$^{Ex}$ using a strong language model, then the same language model is tuned to produce summaries from source sentences on that dataset.

The conceptual goal of *BottleSum*$^{Self}$ is to use *BottleSum*$^{Ex}$ as a guide to learn the notion of information relevance as expressed through IB, but in a way that (a) removes the restriction of extractiveness, to produce more natural outputs and (b) learns an explicit compression function not requiring a next sentence for decoding.

### 4.1 Extractive Dataset

The first step of *BottleSum*$^{Self}$ is to produce a large-scale dataset for self-supervision using the *BottleSum*$^{Ex}$ method set out in §3.2. The only requirement for the input corpus is that next sentences need to be available.

In our experiments, we generate a corpus of 100,000 sentence-summary pairs with *BottleSum*$^{Ex}$, using the same parameter settings as in section 3. The resulting summaries have an average compression ratio (by character length) of approximately 0.55.

### 4.2 Abstractive Fine-tuning

The second step of *BottleSum*$^{Self}$ is fine-tuning the language model on its extractive summary dataset. The tuning data is formed by concatenating source sentences with generated summaries, separated by a delimiter and followed by an end token. The model (GPT-2) is fine-tuned with a simple language modeling objective over the full sequence.

As a delimiter, we use `TL;DR:` , following Radford et al. (2019) who found that this induces summarization behavior in GPT-2 even without tuning. We use a tuning procedure closely related to Radford et al. (2018), training for 10 epochs. We take the trained model weights that minimize loss on a held-out set of 7000 extractive summaries.

To generate from this model, we use a standard beam search decoder, keeping the top candidates at each iteration. Unless otherwise specified, assume we use a beam size of 5. We restrict produced summaries to be at least 5 tokens long, and no longer than the source sentence.

## 5 Experiments

We evaluate our *BottleSum* methods using both automatic metrics and human evaluation. We find our methods dominant over a range of baselines in both categories.

### 5.1 Setup

We evaluate our methods and baselines using automatic ROUGE metrics (1,2,L) on the DUC-2003 and DUC-2004 datasets (Over et al., 2007), similar to the evaluation used by Baziotis et al. (2019). DUC-2003 and DUC-2004 consist of 624 and 500 sentence-summary pairs respectively. Sentences are taken from newstext, and each summary consists of 4 human-written reference summaries capped at 75 bytes. We recover next-sentences from DUC articles for *BottleSum*$^{Ex}$.

We also employ human evaluation as a point of comparison between models. This is both to combat known issues with ROUGE metrics (Schluter, 2017) and to experiment beyond limited supervised domains. Studying unsupervised methods allows for comparison over a much wider range of data where training summary pairs are not available, which we take advantage of here by summarizing sentences from the non-anonymized CNN corpus (Hermann et al., 2015; Nallapati et al., 2016; See et al., 2017).

We use Amazon Mechanical Turk (AMT) for human evaluation, summarizing on 100 sentences sampled from a held out set. Evaluation between systems is primarily done as a pairwise comparison between *BottleSum* models and baselines, over 3 attributes: coherence, conciseness, and agreement with the input. AMT workers are then asked to make a final judgement of which summary has higher overall quality. Each comparison is done by 3 different workers. Results are aggregated across workers and examples.

### 5.2 Models

In both experiments, *BottleSum*$^{Ex}$ is executed as described in section 3.2. In experiments on DUC datasets, next-sentences are recovered from original news sources, while we limit test sentences in the CNN dataset to those with an available next-sentence (this includes over 95% of sentences). We set parameter $k = 1$ (i.e. expand a single candidate at each step) with up to $m = 3$ consecutive words deleted per expansion. GPT-2 (small) is used as the method's pretrained language model,

with no task-specific tuning. To clarify, the only difference between how *BottleSum*$^{Ex}$ runs on the datasets tested here is the input sentences; no data-specific learning is required.

As with *BottleSum*$^{Ex}$, we use GPT-2 (small) as the base for *BottleSum*$^{Self}$. To produce source-summary pairs for self supervision, we generate over 100,000 summaries using *BottleSum*$^{Ex}$ with the parameters above, on both the Gigaword sentence dataset (for automatic evaluation) and CNN training set (for human evaluation). *BottleSum*$^{Self}$ is tuned on the respective set for 10 epoch with a procedure similar to Radford et al. (2019). When generating summaries, *BottleSum*$^{Self}$ uses beam-search with beam size of 5, and outputs constrained to be at least 5 tokens long.

We include a related model, Recon$^{Ex}$ as a simple autoencoding baseline comparable in setup to *BottleSum*$^{Ex}$. Recon$^{Ex}$ follows the procedure of *BottleSum*$^{Ex}$, but replaces the next-sentence with the source sentence. This aims to take advantage of the tendency of language models to semantically repeat in to substitute the Information Bottleneck objective in *BottleSum*$^{Ex}$ with a reconstruction-inspired loss. While this is not a perfect autoencoder by any means, we include it to probe the role of the next-sentence in the success of *BottleSum*$^{Ex}$, particularly compared to a reconstructive method. As Recon$^{Ex}$ tends to have a best reconstructive loss by retaining the entire source as its summary, we constrain its length to be as close as possible to the *BottleSum*$^{Ex}$ summary for the same sentence.

As an unsupervised neural baseline, we include SEQ$^3$ (Baziotis et al., 2019), which is trained with an autoencoding objective paired with a topic loss and language model prior loss. SEQ$^3$ had the highest comparable unsupervised results on the DUC datasets that we are aware of, which we cite directly. For human evaluation, we retrained the model with released code on the training portion of the CNN corpus.

We use the ABS model of Rush et al. (2015) as a baseline for automatic and human evaluation. For automatic evaluation, this model is the best published supervised result we are aware of on the DUC-2003 dataset, and we include it as a point of reference for the gap between supervised and unsupervised performance. We cite their results directly. For human evaluation, this model demonstrates the performance gap for out-of-domain summarization. Specifically, it requires supervision (unavailable for the CNN dataset), and so we use the model as originally trained on the Gigaword sentence dataset. This constitutes a significant domain-shift from the first-sentences of articles with limited vocabulary to arbitrary article sentences with diverse vocabulary.

| Method | R-1 | R-2 | R-L |
|---|---|---|---|
| Supervised | | | |
| ABS | 28.18 | 8.49 | 23.81 |
| Li et al. (2017) | 31.79 | 10.75 | 27.48 |
| Unsupervised | | | |
| PREFIX | 20.91 | 5.52 | 18.20 |
| INPUT | 22.18 | **6.30** | 19.33 |
| SEQ$^3$ | 22.13 | 6.18 | 19.3 |
| Recon$^{Ex}$ | 21.97 | 5.70 | 18.81 |
| *BottleSum*$^{Ex}$ | **22.85** | 5.71 | **19.87** |
| *BottleSum*$^{Self}$ | 22.30 | 5.84 | 19.60 |

Table 1: Averaged ROUGE scores on the DUC-2004 dataset

We include the result of Li et al. (2017) on DUC-2004, who achieved the best supervised performance we are aware of. This is intended as a point of reference for supervised performance.

Finally, for automatic metrics we include common baseline PREFIX, the first 75 bytes of the source sentence. To take into account lack of strict length constraints and possible bias of ROUGE towards longer sequences, we include INPUT, the full input sentence. Because our model is extractive, we know its outputs will be no longer than the input, but may exceed the length of other methods/baselines.

### 5.3 Results

In automatic evaluation, we find *BottleSum*$^{Ex}$ achieves the highest R-1 and R-L scores for unsupervised summarization on both datasets. This is promising in terms of the effectiveness of the Information Bottleneck (IB) as a framework. *BottleSum*$^{Self}$ achieves the second highest scores in both of these categories, further suggesting that the tuning process used here is able to capture some of this benefit. The superiority of *BottleSum*$^{Ex}$ suggests possible benefit to having access to a relevance variable (next-sentence) to the effectiveness of IB on these datasets.

The R-2 scores for *BottleSum*$^{Ex}$ on both bench-

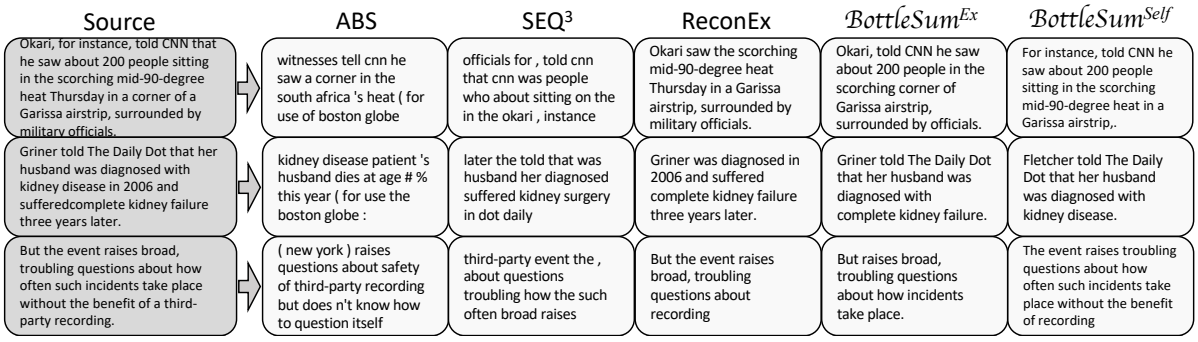


Figure 2: Representative example generations from the summarization systems compared

| Method | R-1 | R-2 | R-L |
|---|---|---|---|
| Supervised | | | |
| ABS | 28.48 | 8.91 | 23.97 |
| Unsupervised | | | |
| PREFIX | 21.14 | **6.35** | 18.74 |
| INPUT | 20.83 | 6.15 | 18.44 |
| SEQ$^3$ | 20.90 | 6.08 | 18.55 |
| Recon$^{Ex}$ | 21.11 | 5.77 | 18.33 |
| *BottleSum*$^{Ex}$ | **21.80** | 5.63 | **19.19** |
| *BottleSum*$^{Self}$ | 21.54 | 5.93 | 18.96 |

Table 2: Averaged ROUGE scores on the DUC-2003 dataset

mark sets were lower than baselines, possibly due to a lack of fluency in the outputs of the extractive approach used. PREFIX and INPUT both copy human text directly and so should be highly fluent, while Rush et al. (2015) and Baziotis et al. (2019) have the benefit of abstractive summarization, which is less restrictive in word order. Further, the fact that *BottleSum*$^{Self}$ is abstractive and surpasses R-2 scores of both new extractive methods tested here (*BottleSum*$^{Ex}$, Recon$^{Ex}$) supports this idea. Recon$^{Ex}$, also extractive, has similar R-2 scores to *BottleSum*$^{Ex}$.

The performance of Recon$^{Ex}$, our simple reconstructive baseline, is mixed. It does succeed to some extent (e.g. surpassing R-1 for all other baselines but PREFIX on DUC-2003) but not as consistently as either *BottleSum* method. This suggests that while some benefit may come from the extractive process of *BottleSum*$^{Ex}$ alone (which Recon$^{Ex}$ shares), there is significant benefit to using a strong relevance variable (specifically in contrast to a reconstructive loss).

Next, we consider model results on human evaluation. *BottleSum*$^{Self}$ and *BottleSum*$^{Ex}$ both show reliably stronger performance compared to models from related work (ABS and SEQ$^3$ in Table 3). While *BottleSum*$^{Self}$ seems superior to Recon$^{Ex}$ other than in conciseness (in accordance with their compression ratios in Table 4), *BottleSum*$^{Ex}$ appears roughly comparable to Recon$^{Ex}$ and slightly inferior to *BottleSum*$^{Self}$.

The inversion of dominance between *BottleSum*$^{Ex}$ and *BottleSum*$^{Self}$ on automatic and human evaluation may cast light on competing advantages. *BottleSum*$^{Ex}$ captures reference summaries more effectively, while *BottleSum*$^{Self}$, through a combination of abstractivness and learning a cohesive underlying mechanism of summarization, writes more favorable summaries for a human audience. Further analysis and accounting for known limitations of ROUGE metrics may clarify these competing advantages.

In comparing these models, there are also practical considerations (summarized in table 5). ABS can be quite effective, but requires learning on a large supervised training set (as demonstrated by its poor out-of-domain performance in Table 3). While SEQ$^3$ is unsupervised, it still needs extensive training on a large corpus of in-domain text. *BottleSum*$^{Ex}$, whose outputs were preferred over both by humans, requires neither of these. Given a strong pretrained language model (GPT-2 small is used here) it only requires a source and next-sentence to summarize. *BottleSum*$^{Self}$ requires in-domain text for self-supervision, but its superior performance by human evaluation and summarization without next-sentence are clear advantages. Further, its beam-search decoding is more computationally efficient than *BottleSum*$^{Ex}$, which requires evaluating conditional next-sentence perplexity over a large grid of extractive summary candidates.

| Models | | Attributes | | | Overall | | |
|---|---|---|---|---|---|---|---|
| Model | Comparison | coherence | conciseness | agreement | better | equal | worse |
| *BottleSum*$^{Ex}$ vs. | ABS | +0.45 | +0.48 | +0.52 | 60% | 31% | 9% |
| | SEQ$^3$ | +0.61 | +0.57 | +0.56 | 61% | 34% | 5% |
| | Recon$^{Ex}$ | -0.05 | +0.01 | -0.05 | 37% | 22% | 41% |
| *BottleSum*$^{Self}$ vs. | ABS | +0.47 | +0.39 | +0.48 | 62% | 26% | 12% |
| | SEQ$^3$ | +0.56 | +0.45 | +0.53 | 65% | 26% | 9% |
| | Recon$^{Ex}$ | +0.11 | -0.05 | +0.09 | 47% | 14% | 39% |
| | *BottleSum*$^{Ex}$ | +0.14 | +0.06 | +0.11 | 43 % | 27% | 30% |

Table 3: Human evaluation on 100 CNN test sentences (pairwise comparison of model outputs). Attribute scores are averaged over a scale of 1 (better), 0 (equal) and -1 (worse). We also report the overall preferences as percentages.

Another difference from *BottleSum*$^{Ex}$ is the ability of *BottleSum*$^{Self}$ to be abstractivene (Table 4). Other baselines have a higher degree of abstractiveness than *BottleSum*$^{Self}$, but this can be misleading. Consider the examples in figure 2. While many of the phrases introduced by other models are technically abstractive, they are often off-topic and confusing.

This hints at an advantage of *BottleSum* methods. In only requiring the base model to be a (tunable) language model, they are architecture-agnostic and can incorporate as powerful a language model as is available. Here, incorporating GPT-2 (small) carries benefits like strong pretrained weights and robust vocabulary handling by byte pair encoding, allowing them to process the diverse language of the non-anonymized CNN corpus with ease. The specific benefits of GPT-2 are less central, however; any such language model could be used for *BottleSum*$^{Ex}$ immediately, and *BottleSum*$^{Self}$ with some tuning. This is in contrast architecture-specific models like ABS and SEQ$^3$, which would require significant restructuring to fully incorporate a new model.

As a first work to study the Information Bottleneck principle for unsupervised summarization, our results suggest this is a promising direction for the field. It yielded two methods with unique performance benefits (Table 1, 2, 3) and practical advantages (table 5). We believe this concept warrants further exploration in future work.

| Model | Abstractive Tokens % | Compression Ratio % |
|---|---|---|
| *BottleSum*$^{Ex}$ | - | 51 |
| Recon$^{Ex}$ | - | 52 |
| *BottleSum*$^{Self}$ | 5.8 | 56 |
| SEQ$^3$ | 12.6 | 58 |
| ABS | 60.4 | 64 |

Table 4: Abstractiveness and compression of CNN summaries. Abstractiveness is omitted for strictly extractive approaches

## 6 Related Work

### 6.1 Sentence Compression and Summarization

Rush et al. (2015) first proposed abstractive sentence compression with neural sequence to sequence models, trained on a large corpus of headlines with the first sentences of newspaper articles as supervision. This followed early work on approaching headline generation as statistical machine translation (Banko et al., 2000). Subsequently, recurrent neural networks with pointer-generator decoders became the standard model for this task, and focus shifted to document-level summarization (Nallapati et al., 2016; See et al., 2017).

Pointer-based neural models have also been proposed for extractive summarization (Cheng and Lapata, 2016). The main limitations of this approach stem from the fact that the training data is constructed heuristically, covering a very specific type of sentence summarization (headline generation). Thus, these supervised models do not generalize well to other kinds of sentence summarization or domains. In contrast, our method is applicable to any domain for which examples of the inputs to be summarized are available in context.

### 6.2 Unsupervised Summarization

Miao and Blunsom (2016) framed sentence compression as an autoencoder problem, where the compressed sentence is a latent variable from

| Model Name | Model architecture | Data for training | Data for summarizing | When to use |
|---|---|---|---|---|
| ABS (Rush et al., 2015) | Seq2Seq | Large scale, paired source-summaries (supervised) | source sentence | Large scale supervised training set available |
| SEQ$^3$ (Baziotis et al., 2019) | Seq2Seq2Seq | Large scale, unsupervised source sentences | source sentence | Large scale unsupervised (no summaries) data available, *without* next-sentences |
| *BottleSum*$^{Ex}$ | Pre-trained LMs | No training data needed | source sentence and next-sentence | No training data available, and next-sentences *are* available for sentences to summarize. |
| *BottleSum*$^{Self}$ | Pre-trained LMs (fined-tuned on data from *BottleSum*$^{Ex}$) | Large scale, unsupervised source sentences with next sentences | source sentence | Large scale unsupervised (no summaries) data available, *with* next-sentences and/or no next-sentences available for sentences to summarize |

Table 5: Comparison of sentence summarization methods.

which the input sentence can be reconstructed. They proposed extractive and pointer-generator models, regularizing the autoencoder with a language model to encourage compression and optimizing the variational objective with the REINFORCE algorithm. While their extractive model can be trained without supervision, results are only reported for semi-supervised training, which requires less supervised data than purely supervised training. Fevry and Phang (2018) applied denoising autoencoders to fully unsupervised summarization, while Wang and Lee (2018) proposed an autoencoder with a discriminator for distinguising well-formed and ill-formed compressions in a Generative Adversarial Network (GAN) setting, as an alternative to using a language model. However, their discriminator was trained using unpaired summaries, so while they obtain better results than purely unsupervised approaches like ours their results are not directly comparable. Recently Baziotis et al. (2019) proposed a differentiable autoencoder using a gumbel-softmax to represent the distribution over summaries. The model is trained with a straight-through estimator as an alternative to reinforcement learning, obtaining better results on unsupervised summarization. All of these approaches have in common autoencoder-based training, which we argue does not naturally capture an appropriate notion of relevance for summarization.

Recently, Zhou and Rush (2019) introduced a promising method for summarization using contextual matching with pretrained language models. While contextual matching requires pretrained language models to generate contextual vectors, *BottleSum* methods do not have specific architectural constraints. Also, like Wang and Lee (2018) it trains with unpaired summaries and so is not directly comparable to us.

## 7 Conclusion

We have presented *BottleSum*$^{Ex}$, an unsupervised extractived approach to sentence summarization, and extended this to *BottleSum*$^{Self}$, a self-supervised abstractive approach. *BottleSum*$^{Ex}$, which can be applied without any training, achieves competitive performance on automatic and human evaluations, compared to unsupervised baselines. *BottleSum*$^{Self}$, trained on a new domain, obtains stronger performance by human evaluation than unsupervised baselines as well as *BottleSum*$^{Ex}$. Our results show that the Information Bottleneck principle, by encoding a more appropriate notion of relevance than autoencoders, offers a promising direction for progress on unsupervised summarization.

## 8 Acknowledgments

We thank anonymous reviewers for many helpful comments. This research is supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) (funding reference number 401233309), NSF (IIS-1524371), DARPA CwC through ARO (W911NF15-1-0543), Darpa MCS program N66001-19-2-4031 through NIWC Pacific (N66001-19-2-4031), Samsung AI Research, and Allen Institute for AI.

## References

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2017. Unsupervised neural ma-

chine translation. *CoRR*, abs/1710.11041.

Michele Banko, Vibhu O. Mittal, and Michael J. Witbrock. 2000. Headline generation based on statistical translation. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 318–325, Hong Kong. Association for Computational Linguistics.

Christos Baziotis, Ion Androutsopoulos, Ioannis Konstas, and Alexandros Potamianos. 2019. SEQ$^3$: Differentiable sequence-to-sequence-to-sequence autoencoder for unsupervised abstractive sentence compression. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL:HLT)*, Minneapolis, USA. To appear.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 484–494, Berlin, Germany. Association for Computational Linguistics.

Thibault Fevry and Jason Phang. 2018. Unsupervised sentence compression using denoising auto-encoders. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 413–422, Brussels, Belgium. Association for Computational Linguistics.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in neural information processing systems*, pages 1693–1701.

Yann LeCun. 2018. Self-supervised learning: could machines learn like humans? `https://www.youtube.com/watch?v=7I0Qt7GALVk`.

Piji Li, Wai Lam, Lidong Bing, and Zihao Wang. 2017. Deep recurrent generative decoder for abstractive text summarization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2091–2100, Copenhagen, Denmark. Association for Computational Linguistics.

Yishu Miao and Phil Blunsom. 2016. Language as a latent variable: Discrete generative models for sentence compression. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 319–328, Austin, Texas. Association for Computational Linguistics.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290.

Paul Over, Hoa Dang, and Donna Harman. 2007. Duc in context. *Information Processing & Management*, 43(6):1506–1520.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. Unpublished manuscript.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. Unpublished manuscript.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389.

Natalie Schluter. 2017. The limits of automatic summarisation according to rouge. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 41–45.

Jurgen Schmidhuber. 1990. Making the world differentiable: On using self-supervised fully recurrent neural networks for dynamic reinforcement learning and planning in non-stationary environments (tr fki-126-90).

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1073–1083.

Naftali Tishby, Fernando C. Pereira, and William Bialek. 1999. The information bottleneck method. In *Proc. of the 37-th Annual Allerton Conference on Communication, Control and Computing*, pages 368–377.

Yaushian Wang and Hung-yi Lee. 2018. Learning to encode text as human-readable summaries using generative adversarial networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4187–4195, Brussels, Belgium. Association for Computational Linguistics.

Jiawei Zhou and Alexander M Rush. 2019. Simple unsupervised summarization by contextual matching. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 5101–5106.

# Improving Latent Alignment in Text Summarization by Generalizing the Pointer Generator

**Xiaoyu Shen**[1,2]*, **Yang Zhao**[3], **Hui Su**[4] **and Dietrich Klakow**[1]
[1]Spoken Language Systems (LSV), Saarland University, Germany
[2]Max Planck Institute for Informatics, Saarland Informatics Campus, Germany
[3]The University of Tokyo, Japan
[4]Pattern Recognition Center, Wechat AI, Tencent Inc, China

## Abstract

Pointer Generators have been the de facto standard for modern summarization systems. However, this architecture faces two major drawbacks: Firstly, the pointer is limited to copying the exact words while ignoring possible inflections or abstractions, which restricts its power of capturing richer latent alignment. Secondly, the copy mechanism results in a strong bias towards extractive generations, where most sentences are produced by simply copying from the source text. In this paper, we address these problems by allowing the model to "edit" pointed tokens instead of always hard copying them. The editing is performed by transforming the pointed word vector into a target space with a learned relation embedding. On three large-scale summarization dataset, we show the model is able to (1) capture more latent alignment relations than exact word matches, (2) improve word alignment accuracy, allowing for better model interpretation and controlling, (3) generate higher-quality summaries validated by both qualitative and quantitative evaluations and (4) bring more abstraction to the generated summaries.[1]

## 1 Introduction

Modern state-of-the-art (SOTA) summarization models are built upon the pointer generator architecture (See et al., 2017). At each decoding step, the model generates a sentinel to decide whether to sample words based on the neural attention (generation mode), or directly copy from an aligned source context (point mode) (Gu et al., 2016; Merity et al., 2017; Yang et al., 2017). Though outperforming the vanilla attention models, the pointer generator only captures exact word matches. As shown in Fig. 1, for abstractive summarization, there exists a large number of syntactic inflections (escalated → escalates) or semantic transformations (military campaign → war), where the target word also has an explicit grounding in the source context but changes its surface. In standard pointer generators, these words are not covered by the point mode. This largely restricts the application of the pointer generator, especially on highly abstractive dataset where only a few words are exactly copied. Moreover, the hard copy operation biases the model towards extractive summarization, which is undesirable for generating more human-like summaries (Kryściński et al., 2018).

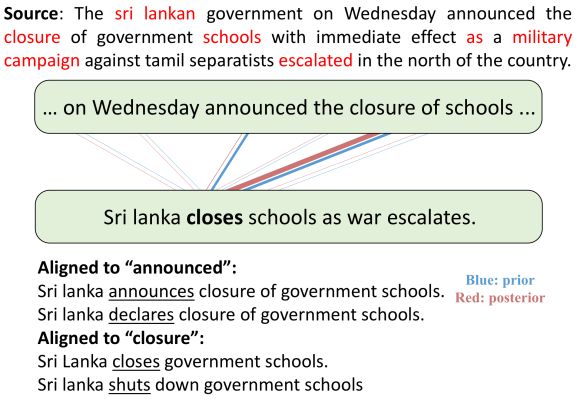


Figure 1: Alignment visualization of our model when decoding "closes". Posterior alignment is more accurate for model interpretation. In contrast, the prior alignment probability is spared to "announced" and "closure", which can be manually controlled to generate desired summaries. Decoded samples are shown when aligned to "announced" and "closure" respectively. Highlighted source words are those that can be directly aligned to a target token in the gold summary.

To solve this problem, we propose **G**eneralized **P**ointer **G**enerator (**GPG**) which replaces the hard copy component with a more general soft "editing" function. We do this by learning a relation embedding to transform the pointed word into a

*Correspondence to xshen@mpi-inf.mpg.de
[1]The source code is available at https://github.com/chin-gyou/generalized-PG.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3762–3773, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

target embedding. For example, when decoding "closes" in Figure 1, the model should first point to "closure" in the source, predict a relation to be applied (noun → third person singular verb), then transform "closure" into "closes" by applying the relation transformation. The generalized point mode is encouraged to capture such latent alignment which cannot be identified by the standard pointer generator.

This improved alignment modelling is intriguing in that (a) people can better control the generation by manipulating the alignment trajectory, (b) posterior alignment can be inferred by Bayes' theorem (Deng et al., 2018; Shankar and Sarawagi, 2019) to provide a better tool for interpretation[2] and finally (c) explicitly capturing the alignment relation should improve generation performance. (Figure 1 shows an example of how latent alignment can improve the controllability and interpretation. Pointer generators fail to model such alignment relations that are not exact copies.) To eliminate the OOV problem, we utilize the byte-pair-encoding (BPE) segmentation (Sennrich et al., 2016) to split rare words into sub-units, which has very few applications in summarization so far (Fan et al., 2018; Kiyono et al., 2018), though being a common technique in machine translation (Wu et al., 2016; Vaswani et al., 2017; Gehring et al., 2017).

Our experiments are conducted on three summarization datasets: CNN/dailymail (Hermann et al., 2015), English Gigaword (Rush et al., 2015) and XSum (Narayan et al., 2018) (a newly collected corpus for extreme summarization). We further perform human evaluation and examine the word alignment accuracy on the manually annotated DUC 2004 dataset. Overall we find our model provides the following benefits:

1. It can capture richer latent alignment and improve the word alignment accuracy, enabling better controllability and interpretation.

2. The generated summaries are more faithful to the source context because of the explicit alignment grounding.

3. It improves the abstraction of generations because our model allows editing the pointed token instead of always copying exactly.

In the next section, we will first go over the background, then introduce our model and finally present the experiment results and conclusion.

## 2 Background

Let $X, Y$ denote a source-target pair where $X$ corresponds to a sequence of words $x_1, x_2, \ldots, x_n$ and $Y$ is its corresponding summary $y_1, y_2, \ldots, y_m$. In this section, we introduce two baseline models for automatically generating $Y$ from $X$:

### 2.1 Seq2seq with Attention

In a seq2seq model, each source token $x_i$ is encoded into a vector $h_i$. At each decoding step $t$, the decoder computes an attention distribution $a_t$ over the encoded vectors based on the current hidden state $d_t$ (Bahdanau et al., 2015):

$$a_t = \text{softmax}(f(h_i, d_t)) \tag{1}$$

$f$ is a score function to measure the similarity between $h_i$ and $d_t$. The context vector $c_t$ and the probability of next token are computed as below.

$$\begin{aligned} c_t &= \sum_i h_i a_{t,i} \\ y_t^* &= [d_t \circ c_t] L \\ p_{vocab} &= \text{softmax}(y_t^* W^T) \end{aligned} \tag{2}$$

$\circ$ means concatenation and $L, W$ are trainable parameters. We tie the parameters of $W$ and the word embedding matrix as in Press and Wolf (2017); Inan et al. (2017). Namely, a target vector $y_t^*$ is predicted, words having a higher inner product with $y_t^*$ will have a higher probability.

### 2.2 Pointer Generator

The pointer generator extends the seq2seq model to support copying source words (Vinyals et al., 2015). At each time step $t$, the model first computes a generation probability $p_{gen} \in [0, 1]$ by:

$$p_{gen} = \sigma(\text{MLP}_g([d_t \circ c_t])) \tag{3}$$

$\sigma$ is a sigmoid function and $\text{MLP}_g$ is a learnable multi-layer perceptron. $p_{gen}$ is the probability of enabling the generation mode instead of the point

[2]The induced alignment offers useful annotations for people to identify the source correspondence for each target word. News editors can post-edit machine-generated summaries more efficiently with such annotation. For summary readers, it also helps them track back the source context when they are interested in some specific details. Derivation for inducing the posterior alignment is in the appendix A.1.

mode. In the generation mode, the model computes the probability over the whole vocabulary as in Eq. 2. In the point mode, the model computes which source word to copy based on the attention distribution $a_t$ from Eq.1. The final probability is marginalized over $a_{t,i}$:

$$p(y_t) = p_{gen}p_{vocab}(y_t) + (1 - p_{gen}) \sum_i a_{t,i}\delta(y_t|x_i)$$

$$\delta(y_t|x_i) = \begin{cases} 1, & \text{if } y_t = x_i. \\ 0, & \text{otherwise.} \end{cases} \quad (4)$$

If we know exactly from which mode each word comes from, *e.g.*, by assuming all co-occurred words are copied, then the marginalization can be omitted (Gulcehre et al., 2016; Wiseman et al., 2017), but normally $p_{gen}$ is treated as a latent variable (Gu et al., 2016; See et al., 2017).

## 3 Generalized Pointer Generator (GPG)

As seen in Eq .4, $\delta(y_t|x_i)$ is a 0-1 event that is only turned on when $y_t$ is exactly the same word as $x_i$. This restricts the expressiveness of the point mode, preventing it from paying attention to inflections, POS transitions or paraphrases. This section explains how we generalize pointer networks to cover these conditions.

**Redefine $\boldsymbol{\delta(y_t|x_i)}$**: We extend $\delta(y_t|x_i)$ by defining it as a smooth probability distribution over the whole vocabulary. It allows the pointer to edit $x_i$ to a different word $y_t$ instead of simply copying it. Following Eq. 2, we derive $\delta(y_t|x_i)$ by first predicting a target embedding $y^*_{t,i}$, then applying the softmax. The difference is that we derive $y^*_{t,i}$ as the summation of the pointed word embedding $\overrightarrow{x_i}$ and a relation embedding $r(d_t, h_t)$:

$$\begin{aligned} y^*_{t,i} &= r(d_t, h_i) + \overrightarrow{x_i} \\ \delta(y_t|x_i) &= \text{softmax}(y^*_{t,i}W^T) \end{aligned} \quad (5)$$

$\overrightarrow{x_i}$ denotes the embedding of the word $x_i$. $r(d_t, h_t)$ can be any function conditioning on $d_t$ and $h_t$, which we parameterize with a multi-layer-perceptron in our experiments. The computation of $y^*_{t,i}$ is similar to the classical TransE model (Bordes et al., 2013) where an entity vector is added by a relation embedding to translate into the target entity. The intuition is straightforward: After pointing to $x_t$, humans usually first decide which relation should be applied (inflection, hypernym, synonym, etc) based on the context $[d_t, h_i]$, then transform $x_i$ to the proper target word $y_t$. Using addition transformation is backed by the observation that vector differences often reflect meaningful word analogies (Mikolov et al., 2013; Pennington et al., 2014) ("$man$" − "$king$" ≈ "$woman$" − "$queen$") and they are effective at encoding a great amount of word relations like hypernym, meronym and morphological changes (Vylomova et al., 2016; Hakami et al., 2018; Allen and Hospedales, 2019). These word relations reflect most alignment conditions in text summarization. For example, humans often change the source word to its hypernym (boy → child), to make it more specific (person → man) or apply morphological transformations (liked → like). Therefore, we assume $\delta(y_t|x_i)$ can be well modelled by first predicting a relation embedding to be applied, then added to $\overrightarrow{x_i}$. If $x_i$ should be exactly copied like in standard pointer generators, the relation embedding is a zero vector meaning an identity transition. We also tried applying more complex transitions to $\overrightarrow{x_i}$ like diagonal mapping (Trouillon et al., 2016), but did not observe improvements. Another option is to estimate $\delta(y_t|x_i)$ directly from $(d_t, h_i)$ by an MLP regardless of $\overrightarrow{x_i}$. However, this leads to poor alignment and performance drop because $y_t$ is not explicitly grounded on $x_i$[3]. A comparison of different choices can be found in the appendix A.4. In this paper, we stick to Eq. 5 to compute $\delta(y_t|x_i)$.

**Estimate Marginal Likelihood**: Putting Eq. 5 back to Eq. 4, the exact marginal likelihood is too expensive for training. The complexity grows linearly with the source text length $n$ and each computation of $\delta(y_t|x_i)$ requires a separate softmax operation. One option is to approximate it by sampling like in hard attention models (Xu et al., 2015; Deng et al., 2017), but the training becomes challenging due to the non-differentiable sampling process. In our work, we take an alternative strategy of marginalizing only over $k$ most likely aligned source words. This top-$k$ approximation is widely adopted when the target distribution is expected to be sparse and only a few modes dominate (Britz et al., 2017; Ke et al., 2018; Shankar et al., 2018). We believe this is a valid assumption in text summarization since most source

[3] $h_i$ can contain context information from surrounding words and thus not necessarily relates to word $x_i$. It is part of the reason that neural attention has a poor alignment (Koehn and Knowles, 2017). Making it grounded on $x_i$ improves alignment and performance is also observed in machine translation (Nguyen and Chiang, 2018; Kuang et al., 2018)

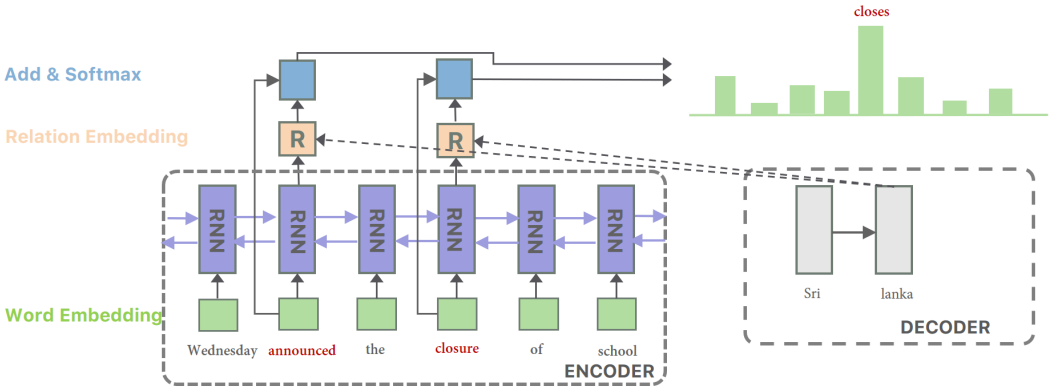


Figure 2: Architecture of the generalized pointer. The same encoder is applied to encode the source and target. When decoding "closes", we first find top-k source positions with the most similar encoded state. For each position, the decoding probability is computed by adding its word embedding and a predicted relation embedding.

tokens have a vanishingly small probability to be transferred into a target word.

For each target word, how to determine the $k$ most likely aligned source words is crucial. An ideal system should always include the gold aligned source word in the top-$k$ selections. We tried several methods and find the best performance is achieved when encoding each source/target token into a vector, then choosing the $k$ source words that are closest to the target word in the encoded vector space. The closeness is measured by the vector inner product[4]. The encoded vector space serves like a contextualized word embedding (McCann et al., 2017; Peters et al., 2018). Intuitively if a target word can be aligned to a source word, they should have similar semantic meaning and surrounding context thus should have similar contextualized word embeddings. The new objective is then defined as in Eq. 6:

$$
\begin{aligned}
p(y_t) &= p_{gen} p_{vocab}(y_t) + (1 - p_{gen}) p_{point}(y_t) \\
p_{point}(y_t) &= \sum_i a_{t,i} \delta(y_t | x_i) \\
&\approx \sum_{i; h_i^T e(y_t) \in \text{TopK}} a_{t,i} \delta(y_t | x_i)
\end{aligned} \tag{6}
$$

$e(y_t)$ is the encoded vector for $y_t$. The marginalization is performed only over the $k$ chosen source words. Eq. 6 is a lower bound of the data likelihood because it only marginalizes over a subset of $X$. In general a larger $k$ can tighten the bound to get a more accurate estimation and we analyze the effect of $k$ in Section 5.2. Note that the only extra parameters introduced by our model are the multi-layer-perceptron to compute the relation embedding $r(d_t, h_i)$. The marginalization in Eq. 6 can also be efficiently parallelized. An illustration of the generalized pointer is in Figure 2.

[4]We compared several strategies for choosing the top-k words and report it in Appendix A.5. Note that the top-k approximation is only used for training, so we can spot the whole target text to decide top-k candidates.

## 4 Related Work

Neural attention models (Bahdanau et al., 2015) with the seq2seq architecture (Sutskever et al., 2014) have achieved impressive results in text summarization tasks. However, the attention vector comes from a weighted sum of source information and does not model the source-target alignment in a probabilistic sense. This makes it difficult to interpret or control model generations through the attention mechanism. In practice, people do find the attention vector is often blurred and suffers from poor alignment (Koehn and Knowles, 2017; Kiyono et al., 2018; Jain and Wallace, 2019). Hard alignment models, on the other hand, explicitly models the alignment relation between each source-target pair. Though theoretically sound, hard alignment models are hard to train. Exact marginalization is only feasible for data with limited length (Yu et al., 2016; Aharoni and Goldberg, 2017; Deng et al., 2018; Backes et al., 2018), or by assuming a simple copy generation process (Vinyals et al., 2015; Gu et al., 2016; See et al., 2017). Our model can be viewed as a combination of soft attention and hard alignment, where a simple top-k approximation is used to train the alignment part (Shankar et al., 2018; Shankar and Sarawagi, 2019). The hard alignment generation probability is designed as a relation summation operation to better fit the sum-

marization task. In this way, the generalized copy mode acts as a hard alignment component to capture the direct word-to-word transitions. On the contrary, the generation mode is a standard soft-attention structure to only model words that are purely functional, or need fusion, high-level inference and can be hardly aligned to any specific source context (Daumé III and Marcu, 2005).

## 5 Experiments and Results

In the experiment, we compare seq2seq with attention, standard pointer generators and the proposed generalized pointer generator (GPG). To further analyze the effect of the generalized pointer, we implement a GPG model with only the point mode (GPG-ptr) for comparison. We first introduce the general setup, then report the evaluation results and analysis.

### 5.1 General Setup

**Dataset**: We perform experiments on the CNN/dailymail (Hermann et al., 2015), English Gigaword (Rush et al., 2015) and XSum (Narayan et al., 2018) dataset. We put statistics of the datasets in Appendix A.2. CNN/DM contains online news with multi-sentence summaries (We use the non-anonymized version from See et al. (2017)). English Gigaword paired the first sentence of news articles with its headline. XSum corpus provides a single-sentence summary for each BBC long story. We pick these three dataset as they have different properties for us to compare models. CNN/DM strongly favors extractive summarization (Kryściński et al., 2018). Gigaword has more one-to-one word direct mapping (with simple paraphrasing) (Napoles et al., 2012) while XSum needs to perform more information fusion and inference since the source is much longer than the target (Narayan et al., 2018).

**Model**: We use single-layer bi-LSTM encoders for all models. For comparison, hidden layer dimensions are set the same as in Zhou et al. (2017) for Gigaword and See et al. (2017) for CNN/DM and XSum. We train with batch size 256 for gigaword and 32 for the other two. The vocabulary size is set to 30k for all dataset. Word representations are shared between the encoder and decoder. We tokenize words with WordPiece segmentation (Wu et al., 2016) to eliminate the OOV problem. More details are in Appendix A.3

**Inference**: We decode text using beam search (Graves, 2012) with beam size 10. We apply length normalization to rescale the score. Unlike See et al. (2017); Gehrmann et al. (2018), we do not explicitly impose coverage penalty since it brings extra hyper-parameters. Instead, for CNN/Dailymail, we use a simple tri-gram penalty (Paulus et al., 2018) to prevent repeated generations. GPG models use an exact marginalization for testing and decoding, while for training and validation we use the top-$k$ approximation mentioned above. The decoder will first decode sub-word ids then map them back to the normal sentence. All scores are reported on the word level and thus comparable with previous results. When computing scores for multi-sentence summaries. The generations are split into sentences with the NLTK sentence tokenizer.

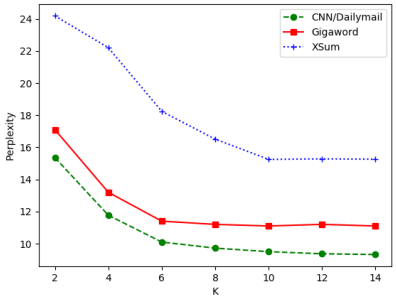


Figure 3: Test perplexity when increasing $k$

### 5.2 Results and Analysis

The results are presented in the following order: We first study the effect of the hyperparameter $k$, then evaluate model generations by automatic metrics and look into the generation's level of abstraction. Finally, we report the human evaluation and word alignment accuracy.

**Effect of K**: $k$ is the only hyperparameter introduced by our model. Figure 3 visualizes the effect of $k$ on the test perplexity. As mentioned in Sec 3, a larger $k$ is expected to tighten the estimation bound and improve the performance. The figure shows the perplexity generally decreases as increasing $k$. The effect on Gigaword and XSum saturates at $k = 6$ and 10 respectively, so we fix such $k$ value for later experiments. For the longer dataset CNN/Dailymail, the perplexity might still decrease a bit afterwards, but the improvement is marginal, so we set $k = 14$ for the memory limit.

**Automatic Evaluation**: The accuracy is evaluated based on the standard metric ROUGE (Lin,

| Method | R-1 | R-2 | R-L | PPL |
|---|---|---|---|---|
| Point.Gen.+Cov* | 39.53 | 17.28 | 36.38 | |
| Bottom Up† | **41.22** | **18.68** | **38.34** | |
| seq2seq | 39.79 | 17.37 | 36.34 | 17.49 |
| Point.Gen. | 40.03 | 17.52 | 36.77 | 12.36 |
| GPG-ptr | 40.54 | **18.05** | 37.19 | 10.23 |
| GPG | **40.95** | 18.01 | **37.46** | **9.37** |

Table 1: ROUGE score on CNN/Dailymail. * marks results from See et al. (2017), and † from Gehrmann et al. (2018). Underlined values are significantly better than Point.Gen. with $p = 0.05$.

| Method | R-1 | R-2 | R-L | PPL |
|---|---|---|---|---|
| seq2seq* | 34.04 | 15.95 | 31.68 | |
| DRGD† | **36.27** | **17.57** | **33.62** | |
| seq2seq | 36.01 | 17.52 | 33.60 | 18.92 |
| Point.Gen. | 36.14 | 17.68 | 33.56 | 14.90 |
| GPG-ptr | 37.14 | **19.05** | **34.67** | 12.32 |
| GPG | **37.23** | 19.02 | 34.66 | **11.41** |

Table 2: ROUGE score on Gigaword. * marks results from the word-based seq2seq implementation of Zhou et al. (2017), and † from Li et al. (2017).

2004) and the word perplexity on the test data. We report the ROUGE-1, ROUGE-2 and ROUGE-L F-score measured by the official script. Table 1, 2 and 3 lists the results for CNN/Dailymail, Gigaword and XSum respectively. Statistically significant results are underlined[5]. On the top two rows of each table, we include two results taken from current state-of-the-art word-based models. They are incomparable with our model because of the different vocabulary, training and decoding process, but we report them for completeness. Lower rows are results from our implemented models. Pointer generators bring only slight improvements over the seq2seq baseline. This suggests that after eliminating the OOV problem, the naive seq2seq with attention model can already implicitly learn most copy operations by itself. GPG models outperform seq2seq and pointer generators on all dataset. The improvement is more significant for more abstractive corpus Gigaword and XSum, indicating our model is effective at identifying more latent alignment relations.

Notably, even the pure pointer model (GPG-ptr) without the generation mode outperforms standard pointer generators in CNN/DM and Gigaword, implying most target tokens can be generated by aligning to a specific source word. The finding is consistent with previous research claiming CNN/DM summaries are largely extractive (Zhang et al., 2018; Kryściński et al., 2018). Though the Gigaword headline dataset is more abstractive, most words are simple paraphrases of some specific source word, so pure pointer GPG-ptr can work well. This is different from the XSum story summarization dataset where many target words require high-level abstraction or inference and cannot be aligned to a single source word, so combining the point and generation mode is necessary for a good performance.

[5]Results on the Gigaword test set is not significant due to the smalle test size (1951 article-summary pairs).

| Method | R-1 | R-2 | R-L | PPL |
|---|---|---|---|---|
| Point.Gen* | 29.70 | 9.21 | 23.24 | |
| T-CONVS2S* | **31.89** | **11.54** | **25.75** | |
| seq2seq | 31.90 | 11.15 | 25.48 | 22.87 |
| Point.Gen. | 31.87 | 11.20 | 25.42 | 17.83 |
| GPG-ptr | 31.49 | 11.02 | 25.37 | 18.62 |
| GPG | **33.11** | **12.55** | **26.57** | **15.28** |

Table 3: ROUGE score on XSum. * marks results from Narayan et al. (2018). Underlined values are significantly better than Point.Gen. with $p = 0.05$.

The word perplexity results are consistent over all dataset (GPG $<$ GPG-ptr $<$ Point.Gen. $<$ seq2seq). The reduction of perplexity does not necessarily indicate an increase for the ROUGE score, especially for pointer generators. This might attribute to the different probability computation of pointer generators, where the probability of copied words are only normalized over the source words. This brings it an inherent advantage over other models where the normalization is over the whole 30k vocabularies.

**Level of Abstraction**: In Tab. 4, we look into how abstractive the generated summaries are by calculating the proportion of novel unigram, bigram and trigrams that do not exist in the corresponding source text. On CNN/DM, as the generations contain multiple sentences, we further report the proportion of novel sentences (obtained with NLTK sent_tokenize).

Tab. 4 reflects the clear difference in the level of abstraction (seq2seq $>$ GPG $>$ GPG-ptr $>$ Point.Gen.). Though the seq2seq baseline generated most novel words, many of them are hallu-

| Models | % of NNs in CNN/Dailymail | | | | % of NNs in Gigaword | | | % of NNs in XSum | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **NN-1** | **NN-2** | **NN-3** | **NN-S** | **NN-1** | **NN-2** | **NN-3** | **NN-1** | **NN-2** | **NN-3** |
| Seq2seq | 0.38 | 3.56 | 7.98 | 54.97 | 16.15 | 52.84 | 73.76 | 27.05 | 76.54 | 92.07 |
| Point.Gen. | 0.04 | 1.51 | 4.29 | 35.82 | 13.99 | 47.79 | 68.53 | 19.45 | 66.68 | 84.59 |
| GPG-ptr | 0.17 | 2.05 | 5.08 | 41.64 | 14.05 | 48.09 | 70.70 | 20.03 | 69.54 | 87.14 |
| GPG | 0.35 | 2.91 | 5.66 | 49.24 | 15.14 | 52.07 | 72.73 | 24.16 | 71.93 | 87.94 |
| Reference | 9.08 | 46.71 | 67.99 | 97.78 | 48.26 | 84.53 | 94.43 | 32.24 | 84.12 | 95.92 |

Table 4: Proportion of novel n-grams (NN-1,2,3) and sentences (NN-S) on generated summaries. GPG generate more novel words compared with standard pointer generators, though still slightly lower than seq2seq.

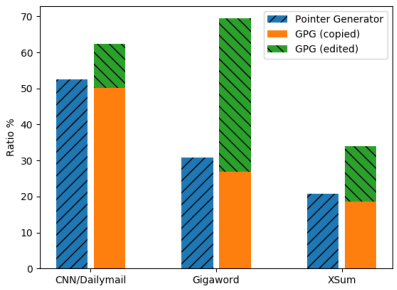


Figure 4: Pointing Ratio of the standard pointer generator and GPG (evaluated on the test data). GPG enables the point mode more often, but quite a few pointed tokens are edited rather than simply copied.

**Article:** (...) and bild **reported** that the (...)
**Summary:** (...) and bild **report** that the (...) [0.964]

**Article:** (...) thousands more **death** row prisoner (...)
**Summary:** (...) thousands more **deaths** row (...) [0.981]

**Article:** (...) surviving **relatives** of a woman who (...)
**Summary:** (...) **family** of woman who (...) [0.984]

**Article:** **jordan** 's crown prince (...)
**Summary:** **jordanian** crown prince (...) [0.993]

**Article:** a middle-aged man and a young **girl** (...) [0.814]
**Summary:** a man and a **child** died (...)

**Article:** (...) , was **abducted** in 2012 (...)
**Summary:** (...) was **kidnapped** in (...) [0.924]

Figure 5: Examples of summaries produced by GPG. Each two samples from CNN/DM, Gigaword and XSum (up to down). **bold** denotes novel words and their pointed source tokens. Bracketed numbers are the pointing probability $(1 - p_{gen})$ during decoding.

cinated facts (see Fig 6), as has also been noted in See et al. (2017). The abstraction of GPG model is close to seq2seq and much higher than copy-based pointer generators. We believe it comes from their ability of "editing" pointed tokens rather than simple copying them.

To examine the pointing behavior of the GPG model, we visualize the average pointing ratio on three dataset in Fig. 4. The pointing ratio can be considered as the chance that a word is generated from the point mode instead of the generation mode. We compute it as $(1 - p_{gen}) \sum_i a_{t,i} \delta(y_t | x_i) / p(y_t)$, averaged over all target tokens in the test data. For the GPG model, we further split it into the copy ratio (words that are exactly copied) and the editing ratio (words that are edited). We find the GPG model enables the point mode more frequently than standord pointer generators, especially on the Gigaword dataset (40% more). This also explains why a pure pointer model is more effective for Gigaword and CNN/DM. More than 60% target tokens can be generated from the point mode, while for XSum the ratio is less than 40%. Quite a few pointing operation includes text rewriting (green bar in Fig. 4). This could explain why our model is able to generate more novel words.

A few examples are displayed in Fig 5. We find our model frequently changes the tense (reported → report), singular/plural (death → deaths) or POS tag (jordan → jordanian) of words. Sometimes it also paraphrases (relatives → family) or abstracts a noun to its hypernym (girl → child). The word editing might be wrong though. For example, "death row prisoner" is wrongly changed to "deaths row prisoner" in the second example, possibly because this phrase is rarely seen so that the model made an error by mistaking "death" as the main subject after "thousands more"[6].

**Human evaluation**: We further perform a human evaluation to assess the model generations. We focus on evaluating the fluency and faithfulness since the ROUGE score often fails to quan-

[6]It also reveals a limit of GPG model in that it only models token-level alignment. For phrases like death row prisoner, it cannot align it based on its compositional meaning.

| | Fluency | Faithfulness | 0/1 |
|---|---|---|---|
| seq2seq | **0.83** | 0.61 | 0.53 |
| Point.Gen. | 0.78 | 0.65 | 0.55 |
| GPG-ptr | 0.79 | **0.78** | 0.67 |
| GPG | 0.82 | 0.74 | **0.69** |
| Gold | 0.96 | 0.92 | 0.96 |

Table 5: Human evaluation results on DUC 2004. 0/1 is the score for the 0/1 Turing test.

tify them (Schluter, 2017; Cao et al., 2018). 100 random source-target pairs are sampled from the human-written DUC 2004 data for task 1&2 (Over et al., 2007). Models trained on Gigaword are applied to generate corresponding summaries. The gold targets, together with the model generations are randomly shuffled then assigned to 10 human annotators. Each pair is evaluated by three different people and the most agreed score is adopted. Each pair is assigned a 0-1 score to indicate (1) whether the target is fluent in grammar, (2) whether the target faithfully conveys the source information without hallucination and (3) whether the target is considered human-generated or machine-generated (like a 0/1 Turing test). The averaged score is reported in Table 5. All models generally achieve high scores in fluency, but generations from GPG models are more faithful to the source information thereby have a larger chance of fooling people into believe they're human-generated (over 0.1 higher score on the 0/1 Turing test). This can be explained by GPG's capability at capturing more latent alignments. As shown in Figure 4, GPG generates over half of the target words by its point mode. Words are generated by explicitly grounding on some source context instead of fabricating freely.

Fig. 6 compares some generation snippets. As can be observed, seq2seq models tend to freely synthesize wrong facts not grounded on the source text, especially on the more difficult XSum dataset. In the last example, seq2seq only capture the subject "odom" and some keywords "police", "basketball" then start to freely fabricate random facts. Pointer generators are slightly better as it is trained to directly copy keyword from the source. However, once it starts to enter the generation mode ("of british" in example 2 and "has been arrested" in example 3), the generation also loses control. GPG largely alleviates the problems because it can point to an aligned source word, then transform it by a learned relation embedding. The explicit alignment modelling encourages the model to stay close to the source information.

| | seq2seq | Point.Gen. | GPG |
|---|---|---|---|
| Prec | 0.361 | 0.435 (0.512) | 0.533 (0.628) |

Table 6: Word Alignment Precision on DUC 2004. Number in bracket is the posterior alignment precision.

**Alignment Accuracy**: We also manually annotate the word alignment on the same 100 DUC 2004 pairs. Following Daumé III and Marcu (2005), words are allowed to be aligned with a specific source word, phrase or a "null" anchor meaning that it cannot be aligned with any source word. The accuracy is only evaluated on the target words with a non-null alignment. For each target token, the most attended source word is considered as alignment (Ghader and Monz, 2017). For the pointer generator and GPG, we also induce the posterior alignment by applying the Bayes' theorem (derivation in appendix A.1). We report the alignment precision (Och and Ney, 2000) in Table 6, i.e., an alignment is considered as valid if it matches one of the human annotated ground truth.

The results show that GPG improves the alignment precision by 0.1 compared with the standard pointer generator. The posterior alignment is more accurate than the prior one (also reflected in Figure 1), enabling better human interpretation.

## 6 Conclusion

In this work, we propose generalizing the pointer generator to go beyond exact copy operation. At each decoding step, the decoder can either generate from the vocabulary, copy or edit some source words by estimating a relation embedding. Experiments on abstractive summarization show the generalized model generates more abstract summaries yet faithful to the source information. The generalized pointer is able to capture richer latent alignment relationship beyond exact copies. This helps improve the alignment accuracy, allowing better model controllability and interpretation.

We believe the generalized pointer mechanism could have potential applications in many fields where tokens are not exactly copied. By integrating off-the-shelf knowledge bases to clearly model the transition relation embedding, it should further improve the interpretability and might be espe-

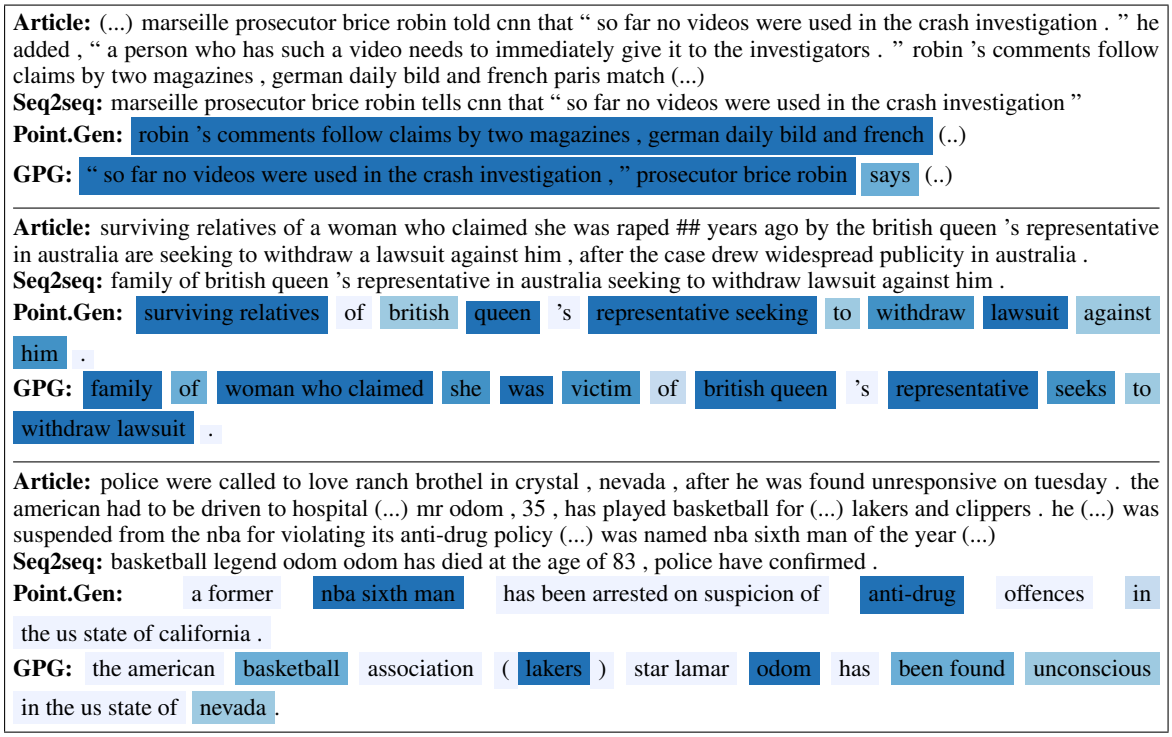


Figure 6: Examples of generated summaries. Examples are taken from CNN/DM, Gigaword and XSum (from up to down). Darker means higher pointing probability.

cially helpful under low-resource settings, which we leave for future work.

## Acknowledgments

We thank anonymous reviewers for valuable comments, thank Marius Mosbach for the proof reading. Xiaoyu Shen is supported by IMPRS-CS fellowship. The work is partially funded by DFG collaborative research center SFB 1102

## References

Roee Aharoni and Yoav Goldberg. 2017. Morphological inflection generation with hard monotonic attention. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2004–2015.

Carl Allen and Timothy Hospedales. 2019. Analogies explained: Towards understanding word embeddings. *ICML*.

Michael Backes, Pascal Berrang, Mathias Humbert, Xiaoyu Shen, and Verena Wolf. 2018. Simulating the large-scale erosion of genomic privacy over time. *IEEE/ACM transactions on computational biology and bioinformatics*, 15(5):1405–1412.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *ICLR*.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

Denny Britz, Melody Guan, and Minh-Thang Luong. 2017. Efficient attention using a fixed-size memory representation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 392–400.

Ziqiang Cao, Furu Wei, Wenjie Li, and Sujian Li. 2018. Faithful to the original: Fact aware neural abstractive summarization. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Sumit Chopra, Michael Auli, and Alexander M Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98.

Hal Daumé III and Daniel Marcu. 2005. Induction of word and phrase alignments for automatic document summarization. *Computational Linguistics*, 31(4):505–530.

Yuntian Deng, Anssi Kanervisto, Jeffrey Ling, and Alexander M Rush. 2017. Image-to-markup generation with coarse-to-fine attention. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 980–989. JMLR. org.

Yuntian Deng, Yoon Kim, Justin Chiu, Demi Guo, and Alexander Rush. 2018. Latent alignment and variational attention. In *Advances in Neural Information Processing Systems*, pages 9735–9747.

Angela Fan, David Grangier, and Michael Auli. 2018. Controllable abstractive summarization. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 45–54. Association for Computational Linguistics.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *International Conference on Machine Learning*, pages 1243–1252.

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109.

Hamidreza Ghader and Christof Monz. 2017. What does attention in neural machine translation pay attention to? In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 30–39.

Alex Graves. 2012. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1631–1640.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 140–149.

Huda Hakami, Kohei Hayashi, and Danushka Bollegala. 2018. Why does pairdiff work?-a mathematical analysis of bilinear relational compositional operators for analogy detection. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2493–2504.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying word vectors and word classifiers: A loss framework for language modeling. *ICLR*.

Sarthak Jain and Byron C Wallace. 2019. Attention is not explanation. *NAACL*.

Nan Rosemary Ke, Anirudh Goyal ALIAS PARTH GOYAL, Olexa Bilaniuk, Jonathan Binas, Michael C Mozer, Chris Pal, and Yoshua Bengio. 2018. Sparse attentive backtracking: Temporal credit assignment through reminding. In *Advances in Neural Information Processing Systems*, pages 7640–7651.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *ICLR*.

Shun Kiyono, Sho Takase, Jun Suzuki, Naoaki Okazaki, Kentaro Inui, and Masaaki Nagata. 2018. Unsupervised token-wise alignment to improve interpretation of encoder-decoder models. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 74–81.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39.

Wojciech Kryściński, Romain Paulus, Caiming Xiong, and Richard Socher. 2018. Improving abstraction in text summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1808–1817.

Shaohui Kuang, Junhui Li, António Branco, Weihua Luo, and Deyi Xiong. 2018. Attention focusing for neural machine translation by bridging source and target embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1767–1776.

Piji Li, Wai Lam, Lidong Bing, and Zihao Wang. 2017. Deep recurrent generative decoder for abstractive text summarization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2091–2100.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6294–6305.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. *ICLR*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 95–100. Association for Computational Linguistics.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807.

Toan Nguyen and David Chiang. 2018. Improving lexical choice in neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 334–343.

Franz Josef Och and Hermann Ney. 2000. A comparison of alignment models for statistical machine translation. In *COLING 2000 Volume 2: The 18th International Conference on Computational Linguistics*.

Paul Over, Hoa Dang, and Donna Harman. 2007. Duc in context. *Information Processing & Management*, 43(6):1506–1520.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. *ICLR*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 2227–2237.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389.

Natalie Schluter. 2017. The limits of automatic summarisation according to rouge. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 41–45.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1073–1083.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1715–1725.

Shiv Shankar, Siddhant Garg, and Sunita Sarawagi. 2018. Surprisingly easy hard-attention for sequence to sequence learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 640–645.

Shiv Shankar and Sunita Sarawagi. 2019. Posterior attention models for sequence to sequence learning. In *International Conference on Learning Representations*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, pages 2071–2080.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Ekaterina Vylomova, Laura Rimell, Trevor Cohn, and Timothy Baldwin. 2016. Take and took, gaggle and goose, book and read: Evaluating the utility of vector differences for lexical relation learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1671–1682.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057.

Zichao Yang, Phil Blunsom, Chris Dyer, and Wang Ling. 2017. Reference-aware language models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1850–1859.

Lei Yu, Jan Buys, and Phil Blunsom. 2016. Online segment to segment neural transduction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1307–1316.

Fangfang Zhang, Jin-ge Yao, and Rui Yan. 2018. On the abstractiveness of neural document summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 785–790.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2017. Selective encoding for abstractive sentence summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1095–1104.

# Learning Semantic Parsers from Denotations with Latent Structured Alignments and Abstract Programs

**Bailin Wang, Ivan Titov** and **Mirella Lapata**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
bailin.wang@ed.ac.uk, {ititov,mlap}@inf.ed.ac.uk

## Abstract

Semantic parsing aims to map natural language utterances onto machine interpretable meaning representations, aka programs whose execution against a real-world environment produces a denotation. Weakly-supervised semantic parsers are trained on utterance-denotation pairs treating programs as latent. The task is challenging due to the large search space and spuriousness of programs which may execute to the correct answer but do not generalize to unseen examples. Our goal is to instill an inductive bias in the parser to help it distinguish between spurious and correct programs. We capitalize on the intuition that correct programs would likely respect certain structural constraints were they to be aligned to the question (e.g., program fragments are unlikely to align to overlapping text spans) and propose to model alignments as structured latent variables. In order to make the latent-alignment framework tractable, we decompose the parsing task into (1) predicting a partial "abstract program" and (2) refining it while modeling structured alignments with differential dynamic programming. We obtain state-of-the-art performance on the WIKITABLEQUESTIONS and WIKISQL datasets. When compared to a standard attention baseline, we observe that the proposed structured-alignment mechanism is highly beneficial.

## 1 Introduction

Semantic parsing is the task of translating natural language to machine interpretable meaning representations. Typically, it requires mapping a natural language utterance onto a program, which is executed against a knowledge base to obtain an answer or a denotation. Most previous work (Zettlemoyer and Collins, 2005; Wong and Mooney, 2007; Lu et al., 2008; Jia and Liang, 2016) has focused on the supervised setting where a model is learned from question-program pairs. Weakly supervised semantic parsing (Berant et al., 2013; Liang et al., 2011) reduces the burden of annotating programs by learning from questions paired with their answers (or denotations).

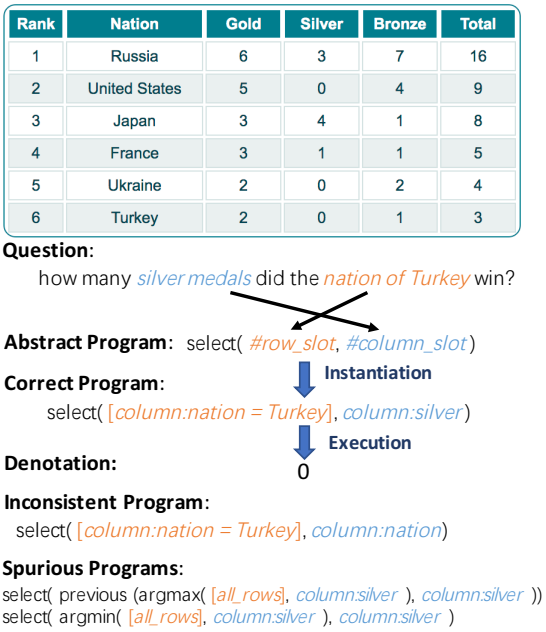


Figure 1: After generating an abstract program for a question, our parser finds alignments between slots (with prefix #) and question spans. Based on the alignment, it instantiates each slot and a complete program is executed against a table to obtain a denotation.

Two major challenges arise when learning from denotations: 1) training of the semantic parser requires exploring a large search space of possible programs to find those which are *consistent*, and execute to correct denotations; 2) the parser should be robust to *spurious* programs which accidentally execute to correct denotations, but do not reflect the semantics of the question. In this paper, we propose a weakly-supervised neural semantic parser that features structured latent alignments to bias learning towards *correct* programs which are consistent but not spurious.

Our intuition is that correct programs should re-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3774–3785, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

spect certain constraints were they to be aligned to the question text, while spurious and inconsistent programs do not. For instance, in Figure 1, the answer to the question ("0") can be obtained by executing the correct program which selects the number of Turkey's silver medals. However, the same answer can be also obtained by the spurious programs shown in the figure.[1] The spurious programs differ from the correct one in that they repeatedly use the column "silver". Whereas, in the question, the word "silver" only refers to the target column containing the answer; it also mistakenly triggers the appearance of the column "silver" in the row selection condition. This constraint, i.e., that a text span within a question cannot trigger two semantically distinct operations (e.g., selecting target rows and target columns) can provide a useful inductive bias. We propose to capture structural constraints by modeling the alignments between programs and questions *explicitly* as structured latent variables.

Considering the large search space of possible programs, an alignment model that takes into account the full range of correspondences between program operations and question spans would be very expensive. To make the process tractable, we introduce a two-stage approach that features *abstract* programs. Specifically, we decompose semantic parsing into two steps: 1) a natural language utterance is first mapped to an abstract program which is a composition of high-level operations; and 2) the abstract program is then instantiated with low-level operations that usually involve relations and entities specific to the knowledge base at hand. This decomposition is motivated by the observation that only a small number of sensible abstract programs can be instantiated into consistent programs. Similar ideas of using abstract meaning representations have been explored with fully-supervised semantic parsers (Dong and Lapata, 2018; Catherine Finegan-Dollak and Radev, 2018) and in other related tasks (Goldman et al., 2018; Herzig and Berant, 2018; Nye et al., 2019).

For a knowledge base in tabular format, we abstract two basic operations of row selection and column selection from programs: these are handled in the second (instantiation) stage. As shown in Figure 1, the question is first mapped to the abstract program "select (#*row_slot*, #*column_slot*)" whose two slots are subsequently instantiated with filter conditions (row slot) and a column name (column slot). During the instantiation of abstract programs, each slot should refer to the question to obtain its specific semantics. In Figure 1, *row_slot* should attend to "nation of Turkey" while *column_slot* needs to attend to "silver medals". The structural constraint discussed above now corresponds to assuming that each span in a question can be aligned to a *unique* row or column slot. Under this assumption, the instantiation of spurious programs will be discouraged. The uniqueness constraint would be violated by both spurious programs in Figure 1, since "column:silver" appears in the program twice but can be only aligned to the span "silver medals" once.

The first stage (i.e., mapping a question onto an abstract program) is handled with a sequence-to-sequence model. The second stage (i.e., program instantation) is approached with local classifiers: one per slot in the abstract program. The classifiers are conditionally independent given the abstract program and a latent alignment. Instead of marginalizing out alignments, which would be intractable, we use structured attention (Kim et al., 2017), i.e., we compute the marginal probabilities for individual span-slot alignment edges and use them to weight the input to the classifiers. As we discuss below, the marginals in our constrained model are computed with dynamic programming.

We perform experiments on two open-domain question answering datasets in the setting of learning from denotations. Our model achieves an execution accuracy of 44.5% in WIKITABLEQUESTIONS and 79.3% in WIKISQL, which both surpass previous state-of-the-art methods in the same weakly-supervised setting. In WIKISQL, our parser is better than recent supervised parsers that are trained on question-program pairs. Our contributions can be summarized as follows:

- we introduce an alignment model as a means of differentiating between correct and spurious programs;
- we propose a neural semantic parser that performs tractable alignments by first mapping questions to abstract programs;
- we achieve state-of-the-art performance on two semantic parsing benchmarks.[2]

[1] The first program can be paraphrased as: find the row with the largest number of silver medals and then select the number of silver medals from the previous row.

[2] Our code is available at https://github.com/berlino/weaksp_em19.

Although we use structured alignments to mostly enforce the uniqueness constraint described above, other types of inductive biases can be useful and could be encoded in our two-stage framework. For example, we could replace the uniqueness constraint with modeling the number of slots aligned to a span, or favor sparse alignment distributions. Crucially, the two-stage framework makes it easier to inject prior knowledge about datasets and formalisms while maintaining efficiency.

## 2 Background

Given knowledge base $t$, our task is to map a natural utterance $x$ to program $z$, which is then executed against a knowledge base to obtain denotation $[[z]]_t = d$. We train our parser only based on $d$ without access to correct programs $z^*$. Our experiments focus on two benchmarks, namely WIKITABLEQUESTIONS (Pasupat and Liang, 2015) and WIKISQL (Zhong et al., 2017) where each question is paired with a Wikipedia table and a denotation. Figure 1 shows a simplified example taken from WIKITABLEQUESTIONS.

### 2.1 Grammars

Executable programs $z$ that can query tables are defined according to a language. Specifically, the search space of programs is constrained by grammar rules so that it can be explored efficiently. We adopt the variable-free language of Liang et al. (2018) and define an *abstract* grammar and an *instantiation* grammar which decompose the generation of a program in two stages.[3]

The first stage involves the generation of an abstract version of a program which, in the second stage, gets instantiated. Abstract programs only consider compositions of high-level functions, such as superlatives and aggregation, while low-level functions and arguments, such as filter conditions and entities, are taken into account in the next step. In our table-based datasets, abstract programs do not include two basic operations of querying tables: row selection and column selection. These operations are handled at the instantiation stage. In Figure 1 the abstract program has two slots for row and column selection, which are filled with the conditions "column:nation = Turkey" and "column:silver" at the instantiation stage. The two stages can be easily merged into one step when conducting symbolic combinatorial search. The motivation for the decomposition is to facilitate the learning of our neural semantic parser and the handling of structured alignments.

[3]We also extend their grammar to additionally support operations of conjunction and disjunction. More details are provided in the Appendix.

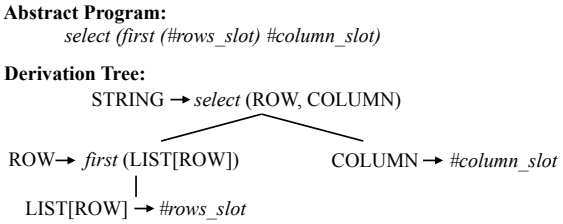


Figure 2: An abstract program and its derivation tree. Capitalized words indicate types of function arguments and their return value.

**Abstract Grammar** Our abstract grammar has five basic types: ROW, COLUMN, STRING, NUMBER, and DATE; COLUMN is further sub-typed into STRING_COLUMN, NUMBER_COLUMN, and DATE_COLUMN; other basic types are augmented with LIST to represent a list of elements like LIST[ROW]. Arguments and return values of functions are typed using these basic types.

Function composition can be defined recursively based on a set of production rules, each corresponding to a function type. For instance, function ROW → *first*(LIST[ROW]) selects the first row from a list of rows and corresponds to production rule "ROW → *first*".

The abstract grammar has two additional types for slots (aka terminal rules) which correspond to row and column selection:

$$\text{LIST[ROW]} \rightarrow \textit{\#row\_slot}$$
$$\text{COLUMN} \rightarrow \textit{\#column\_slot}$$

An example of an abstract program and its derivation tree is shown in Figure 2. We linearize the derivation by traversing it in a left-to-right depth-first manner. We represent the tree in Figure 2 as a sequence of production rules: "ROOT → STRING, STRING → *select*, ROW → *first*", LIST[ROW] → *#row_slot*, COLUMN → *#column_slot*". The first action is always to select the return type for the root node.

Given a specific table $t$, the abstract grammar $\mathcal{H}_t$ will depend on its column types. For example, if the table does not have number cells, "max/min" operations will not be executable.

**Instantiation Grammar** A column slot is directly instantiated by selecting a column; a row

slot is filled with one or multiple conditions (COND) which are joined together with conjunction (*OR*) and disjunction (*AND*) operators:

COND → COLUMN OPERATOR VALUE
*#row_slot* → COND (*AND* COND)*
*#row_slot* → COND (*OR* COND)*

where OPERATOR $\in [>, <, =, \geq, \leq]$ and VALUE is a string, a number, or a date. A special condition *#row_slot* → $all\_rows$ is defined to signify that a program queries all rows.

### 2.2 Search for Consistent Programs

A problematic aspect of learning from denotations is that, since annotated programs are not available (e.g., for WIKITABLEQUESTIONS), we have no means to directly evaluate a proposed grammar. As an evaluation proxy, we measure the coverage of our grammar in terms of consistent programs. Specifically, we exhaustively search for all consistent programs for each question in the training set. While the space of programs is exponential, we observed that abstract programs which are instantiated into correct programs are not very complex in terms of the number of production rules used to generate them. As a result, we impose restrictions on the number of production rules which can abstract programs, and in this way the search process becomes tractable.[4]

We find that 83.6% of questions in WIKITABLEQUESTIONS are covered by at least one consistent program. However, each question eventually has 200 consistent programs on average and most of them are spurious. Treating them as ground truth poses a great challenge for learning a semantic parser. The coverage for WIKISQL is 96.6% and each question generates 84 consistent programs.

Another important observation is that there is only a limited number of abstract programs that can be instantiated into consistent programs. The number of such abstract programs is 23 for WIKITABLEQUESTIONS and 6 for WIKISQL, suggesting that there are a few patterns underlying several utterances. This motivates us to design a semantic parser that first maps utterances to abstract programs. For the sake of generality, we do not restrict our parser to abstract programs in the training set. We elaborate on this below.

[4]Details are provided in the Appendix.

## 3 Model

After obtaining consistent programs $z$ for each question via offline search, we next show how to learn a parser that can generalize to unseen questions and tables.

### 3.1 Training and Inference

Our learning objective $\mathcal{J}$ is to maximize the log-likelihood of the marginal probability of all consistent programs, which are generated by mapping an utterance $x$ to an interim abstract program $h$:

$$\mathcal{J} = \log\{\sum_{h \in \mathcal{H}_t} p(h|x,t) \sum_{[[z]]=d} p(z|x,t,h)\}. \quad (1)$$

During training, our model only needs to focus on abstract programs that have successful instantiations of consistent programs and it does not have to explore the whole space of possible programs.

At test time, the parser chooses the program $\hat{z}$ with the highest probability:

$$\hat{h}, \hat{z} = \arg\max_{h \in \mathcal{H}_t,\ z} p(h|x,t)p(z|x,t,h). \quad (2)$$

For efficiency, we only choose the top-$k$ abstract programs to instantiate through beam search. $\hat{z}$ is then executed to obtain its denotation as the final prediction.

Next, we will explain the basic components of our neural parser. Basically, our model first encodes a question and a table with an input encoder; it then generates abstract programs with a seq2seq model; and finally, these abstract programs are instantiated based on a structured alignment model.

### 3.2 Input Encoder

Each word in an utterance is mapped to a distributed representation through an embedding layer. Following previous work (Neelakantan et al., 2017; Liang et al., 2018), we also add an indicator feature specifying whether the word appears in the table. This feature is mapped to a learnable vector. Additionally, in WIKITABLEQUESTIONS, we use POS tags from the CoreNLP annotations released with the dataset and map them to vector representations. The final representation for a word is the concatenation of the vectors above. A bidirectional LSTM (Hochreiter and Schmidhuber, 1997) is then used to obtain a contextual representation $\boldsymbol{l}_i$ for the $i_{th}$ word.

A table is represented by a set of columns. Each column is encoded by averaging the embeddings

of words under its column name. We also have a column type feature (i.e., number, date, or string) and an indicator feature signaling whether at least one entity in the column appears in the utterance.

### 3.3 Generating Abstract Programs

Instead of extracting abstract programs as templates, similarly to Xu et al. (2017) and Catherine Finegan-Dollak and Radev (2018), we generate them with a seq2seq model. Although template-based approaches would be more efficient in practice, a seq2seq model is more general since it could generate unseen abstract programs which fixed templates could not otherwise handle.

Our goal here is to generate a sequence of production rules that lead to abstract programs. During decoding, the hidden state $\boldsymbol{g}_j$ of the $j$-th timestep is computed based on the previous production rule, which is mapped to an embedding $\boldsymbol{a}_{j-1}$. We also incorporate an attention mechanism (Luong et al., 2015) to compute a contextual vector $\boldsymbol{b}_j$. Finally, a score vector $\boldsymbol{s}_j$ is computed by feeding the concatenation of the hidden state and context vector to a multilayer perceptron (MLP):

$$\begin{aligned} \boldsymbol{g}_j &= \text{LSTM}(\boldsymbol{g}_{j-1}, \boldsymbol{a}_{j-1}) \\ \boldsymbol{b}_j &= \text{Attention}(\boldsymbol{g}_j, \boldsymbol{l}) \\ \boldsymbol{s}_j &= \text{MLP}_1([\boldsymbol{g}_j; \boldsymbol{b}_j]) \\ p(a_j|x,t,a_{<j}) &= \text{softmax}_{a_j}(\boldsymbol{s}_j) \end{aligned} \tag{3}$$

where the probability of production rule $a_j$ is computed by the softmax function. According to our abstract grammar, only a subset of production rules will be valid at the $j$-th time step. For instance, in Figure 2, production rule "STRING $\rightarrow$ *select*" will only expand to rules whose left-hand side is ROW, which is the type of the first argument of *select*. In this case, the next production rule is "ROW $\rightarrow$ *first*". We thus restrict the normalization of softmax to only focus on these valid production rules. The probability of generating an abstract program $p(h|x,t)$ is simply the product of the probability of predicting each production rule $\prod_j p(a_j|x,t,a_{<j})$.

After an abstract program is generated, we need to instantiate slots in abstract programs. Our model first encodes the abstract program using a bi-directional LSTM. As a result, the representation of a slot is contextually aware of the entire abstract program (Dong and Lapata, 2018).

### 3.4 Instantiating Abstract Programs

To instantiate an abstract program, each slot must obtain its specific semantics from the question. We model this process by an alignment model which learns the correspondence between slots and question spans. Formally, we use a binary alignment matrix $\boldsymbol{A}$ with size $m \times n \times n$, where $m$ is the number of slots and $n$ is the number of tokens. In Figure 1, the alignment matrix will only have $\boldsymbol{A}_{0,6,8} = 1$ and $\boldsymbol{A}_{1,2,3} = 1$ which indicates that the first slot is aligned with "nation of Turkey", and the second slot is aligned with "silver medals". The second and third dimension of the matrix represent the start and end position of a span.

We model alignments as discrete latent variables and condition the instantiation process on the alignments as follows:

$$\begin{aligned} &\sum_{[z]=d} p(z|x,t,h) = \\ &\quad \sum_{\boldsymbol{A}} p(\boldsymbol{A}|x,t,h) \sum_{[z]=d} p(z|x,t,h,\boldsymbol{A}). \end{aligned} \tag{4}$$

We will first discuss the instantiation model $p(z|x,t,h,\boldsymbol{A})$ and then elaborate on how to avoid marginalization in the next section. Each slot in an abstract program can be instantiated by a set of candidates following the instantiation grammar. For efficiency, we use local classifiers to model the instantiation of each slot independently:

$$p(z|x,t,h,\boldsymbol{A}) = \prod_{s \in S} p(s \rightarrow c|x,t,h,\boldsymbol{A}), \tag{5}$$

where $S$ is the set of slots and $c$ is a candidate following our instantiation grammar. "$s \rightarrow c$" represents the instantiation of slot $s$ into candidate $c$.

Recall that there are two types of slots, one for rows and one for columns. All column names in the table are potential instantiations of column slots. We represent each column slot candidate by the average of the embeddings of words in the column name. Based on our instantiation grammar in Section 2.1, candidates for row slots are represented as follows: 1) each condition is represented with the concatenation of the representations of a column, an operator, and a value. For instance, condition "string_column:nation = Turkey" in Figure 1 is represented by vector representations of the column 'nation', the operator '=', and the entity 'Turkey'; 2) multiple conditions are encoded by averaging the representations of all conditions and adding a vector representation of *AND*/*OR* to indicate the relation between them.

For each slot, the probability of generating a candidate is computed with softmax normalization on a score function:

$$p(s \to c|x,t,h,\boldsymbol{A}) \propto \exp\{\text{MLP}([\boldsymbol{s};\boldsymbol{c}])\}, \quad (6)$$

where $\boldsymbol{s}$ is the representation of the span that slot $s$ is aligned with, and $\boldsymbol{c}$ is the representation of candidate $c$. The representations $\boldsymbol{s}$ and $\boldsymbol{c}$ are concatenated and fed to a MLP. We use the same MLP architecture but different parameters for column and row slots.

### 3.5 Structured Attention

We first formally define a few structural constraints over alignments and then explain how to incorporate them efficiently into our parser.

The intuition behind our alignment model is that row and column selection operations represent distinct semantics, and should therefore be expressed by distinct natural language expressions. Hence, we propose the following constraints:

**Unique Span** In most cases, the semantics of a row selection or a column selection is expressed uniquely with a single contiguous span:

$$\forall k \in [1, |S|], \quad \sum_{i,j} \boldsymbol{A}_{k,i,j} = 1, \quad (7)$$

where $|S|$ is the number of slots.

**No Overlap** Spans aligned to different slots should not overlap. Formally, at most one span that contains word $i$ can be aligned to a slot:

$$\forall i \in [1, n], \quad \sum_{k,j} \boldsymbol{A}_{k,i,j} \leq 1. \quad (8)$$

As an example, the alignments in Figure 1 follow the above constraints. Intuitively, the one-to-one mapping constraint aims to assign distinct and non-overlapping spans to slots of abstract programs. To further bias the alignments and improve efficiency, we impose additional restrictions: (1) a row slot must be aligned to a span that contains an entity since conditions that instantiate the slot would require entities for filtering; (2) a column slot must be aligned to a span with length 1 since most column names only have one word.

Marginalizing out all $\boldsymbol{A}$ in Equation (4) would be very expensive considering the exponential number of possible alignments. We approximate the marginalization by moving the outside expectation directly inside over $\boldsymbol{A}$. As a result, we instead optimize the following objective:

$$\mathcal{J} \approx \log\Big\{ \sum_{h \in \mathcal{H}_t} p(h|x,t) \sum_{[[z]]=d} p(z|x,t,h,\mathbb{E}[\boldsymbol{A}]) \Big\}, \quad (9)$$

where $\mathbb{E}[\boldsymbol{A}]$ are the marginals of $\boldsymbol{A}$ with respect to $p(\boldsymbol{A}|x,t,h)$.

The idea of using differentiable surrogates for discrete latent variables has been used in many other works like differentiable data structures (Grefenstette et al., 2015; Graves et al., 2014) and attention-based networks (Bahdanau et al., 2015; Kim et al., 2017). Using marginals $\mathbb{E}[\boldsymbol{A}]$ can be viewed as *structured attention* between slots and question spans.

The marginal probability of the alignment matrix $\boldsymbol{A}$ can be computed efficiently using dynamic programming (see Täckström et al. 2015 for details). An alignment is encoded into a path in a weighted lattice where each vertex has $2^{|S|}$ states to keep track of the set of covered slots. The marginal probability of edges in this lattice can be computed by the forward-backward algorithm (Wainwright et al., 2008). The lattice weights, represented by a scoring matrix $\boldsymbol{M} \in \mathbb{R}^{m \times n \times n}$ for all possible slot-span pairs, are computed using the following scoring function:

$$\boldsymbol{M}_{k,i,j} = \text{MLP}_2([\boldsymbol{r}(k); \mathbf{span}[i:j]]), \quad (10)$$

where $\boldsymbol{r}(k)$ represents the $k_{th}$ slot and $\mathbf{span}[i:j]$ represents the span from word $i$ to $j$. Recall that we obtain $\boldsymbol{r}(k)$ by encoding a generated abstract program. A span is represented by averaging the representations of the words therein. These two representations are concatenated and fed to a MLP to obtain a score. Since $\mathbb{E}[\boldsymbol{A}]$ is not discrete anymore, the aligned representation of slot $\boldsymbol{s}$ in Equation (6) becomes the weighted average of representations of all spans in the set.

## 4 Experiments

We evaluated our model on two semantic parsing benchmarks, WIKITABLEQUESTIONS and WIKISQL. We compare against two common baselines to demonstrate the effectiveness of using abstract programs and alignment. We also conduct detailed analysis which shows that structured attention is highly beneficial, enabling our parser to differentiate between correct and spurious programs. Finally, we break down the errors of our parser so as

to examine whether structured attention is better at instantiating abstract programs.

### 4.1 Experimental Setup

**Datasets** WIKITABLEQUESTIONS contains 2,018 tables and 18,496 utterance-denotation pairs. The dataset is challenging as 1) the tables cover a wide range of domains and unseen tables appear at test time; and 2) the questions involve a variety of operations such as superlatives, comparisons, and aggregation (Pasupat and Liang, 2015). WIKISQL has 24,241 tables and 80,654 utterance-denotation pairs. The questions are logically simpler and only involve aggregation, column selection, and conditions. The original dataset is annotated with SQL queries, but we only use the execution result for training. In both datasets, tables are extracted from Wikipedia and cover a wide range of domains.

Entity extraction is important during parsing since entities are used as values in filter conditions during instantiation. String entities are extracted by string matching utterance spans and table cells. In WIKITABLEQUESTIONS, numbers and dates are extracted from the CoreNLP annotations released with the dataset. WIKISQL does not have entities for dates, and we use string-based normalization to deal with numbers.

**Implementation** We obtained word embeddings by a linear projection of GloVe pre-trained embeddings (Pennington et al., 2014) which were fixed during training. Attention scores were computed based on the dot product between two vectors. Each MLP is a one-hidden-layer perceptron with ReLU as the activation function. Dropout (Srivastava et al., 2014) was applied to prevent overfitting. All models were trained with Adam (Kingma and Ba, 2015). Implementations of abstract and instantiation grammars were based on AllenNLP (Gardner et al., 2017).[5]

### 4.2 Baselines

Aside from comparing our model against previously published approaches, we also implemented the following baselines:

**Typed Seq2Seq** Programs were generated using a sequence-to-sequence model with attention (Dong and Lapata, 2016). Similarly to Krishnamurthy et al. (2017), we constrained the decoding process so that only well-formed programs are predicted. This baseline can be viewed as merging the two stages of our model into one stage where generation of abstract programs and their instantiations are performed with a shared decoder.

[5]Please refer to the Appendix for the full list of hyperparameters used in our experiments.

| **Supervised by Denotations** | Dev. | Test |
|---|---|---|
| Pasupat and Liang (2015) | 37.0 | 37.1 |
| Neelakantan et al. (2017) | 34.1 | 34.2 |
| Haug et al. (2018) | — | 34.8 |
| Zhang et al. (2017) | 40.4 | 43.7 |
| Liang et al. (2018) | 42.3 | 43.1 |
| Dasigi et al. (2019) | 42.1 | 43.9 |
| Agarwal et al. (2019) | 43.2 | 44.1 |
| Typed Seq2Seq | 37.3 | 38.3 |
| Abstract Programs | | |
| *f.w.* standard attention | 39.4 | 41.4 |
| *f.w.* structured attention | **43.7** | **44.5** |

Table 1: Results on WIKITABLEQUESTIONS. *f.w.* stands for slots filled with.

**Standard Attention** The aligned representation of slot $s$ in Equation (6) is computed by a standard attention mechanism: $\boldsymbol{s} = \text{Attention}(\boldsymbol{r}(s), \boldsymbol{l})$ where $\boldsymbol{r}(s)$ is the representation of slot $s$ from abstract programs. Each slot is aligned independently with attention, and there are no global structural constraints on alignments.

### 4.3 Main Results

For all experiments, we report the mean accuracy of 5 runs. Results on WIKITABLEQUESTIONS are shown in Table 1. The structured-attention model achieves the best performance, compared against the two baselines and previous approaches. The standard attention baseline with abstract programs is superior to the typed Seq2Seq model, demonstrating the effectiveness of decomposing semantic parsing into two stages. Results on WIKISQL are shown in Table 2. The structured-attention model is again superior to our two baseline models. Interestingly, its performance surpasses previously reported weakly-supervised models (Liang et al., 2018; Agarwal et al., 2019) and is on par even with fully supervised ones (Dong and Lapata, 2018).

The gap between the standard attention baseline and the typed Seq2Seq model is not very large on WIKISQL, compared to WIKITABLEQUESTIONS. Recall from Section 2.2 that WIK-

| Supervised by Programs | Dev. | Test |
|---|---|---|
| Zhong et al. (2017) | 60.8 | 59.4 |
| Wang et al. (2017) | 67.1 | 66.8 |
| Xu et al. (2017) | 69.8 | 68.0 |
| Huang et al. (2018) | 68.3 | 68.0 |
| Yu et al. (2018) | 74.5 | 73.5 |
| Sun et al. (2018) | 75.1 | 74.6 |
| Dong and Lapata (2018) | 79.0 | 78.5 |
| Shi et al. (2018) | **84.0** | **83.7** |
| **Supervised by Denotations** | Dev. | Test |
| Liang et al. (2018) | 72.2 | 72.1 |
| Agarwal et al. (2019) | 74.9 | 74.8 |
| Typed Seq2Seq | 74.5 | 74.7 |
| Abstract Programs | | |
| *f.w.* standard attention | 75.2 | 75.3 |
| *f.w.* structured attention | 79.4 | 79.3 |

Table 2: Results on WIKISQL. *f.w.*: slots filled with.

| Models | WTQ | WIKISQL |
|---|---|---|
| Liang et al. (2018) | 46.3 (10) | 74.2 (5) |
| Agarwal et al. (2019) | 46.9 (10) | 76.9 (5) |
| Our model | **47.3** (10) | **81.7** (5) |

Table 3: Results of ensembled models on the test set; ensemble sizes are shown within parentheses.

| Error Types | standard | structured |
|---|---|---|
| Abstraction Error | 19.2 | 20.0 |
| Instantiation Error | 41.5 | 36.2 |
| Coverage Error | 39.2 | 43.8 |

Table 4: Proportion of errors on the development set in WIKITABLEQUESTIONS.

ISQL only has 6 abstract programs that can be successfully instantiated. For this reason, our decomposition alone may not be very beneficial if coupled with standard attention. In contrast, our structured-attention model consistently performs much better than both baselines.

We report scores of ensemble systems in Table 3. We use the best model which relies on abstract programs and structured attention as a base model in our ensemble. Our ensemble system achieves better performance than Liang et al. (2018) and Agarwal et al. (2019), while using the same ensemble size.

### 4.4 Analysis of Spuriousness

To understand how well structured attention can help a parser differentiate between correct and spurious programs, we analyzed the posterior distribution of consistent programs given a denotation: $p(z|x,t,d)$ where $[[z]] = d$.

WIKISQL includes gold-standard SQL annotations, which we do not use in our experiments but exploit here for analysis. Specifically, we converted the annotations released with WIKISQL to programs licensed by our grammar. We then computed the log-probability of these programs according to the posterior distribution as a measure of how well a parser can identify them amongst all consistent programs $\log \sum_{z^*} p(z^*|x,t,d)$, where $z^*$ denotes correct programs. The average log-probability assigned to correct programs by structured and standard attention is -0.37 and -0.85, respectively. This gap confirms that structured attention can bias our parser towards correct programs during learning.

### 4.5 Error Analysis

We further manually inspected the output of our structured-attention model and the standard attention baseline in WIKITABLEQUESTIONS. Specifically, we randomly sampled 130 error cases independently from both models and classified them into three categories.

**Abstraction Errors** If a parser fails to generate an abstract program, then it is impossible for it to instantiate a consistent complete program.

**Instantiation Errors** These errors arise when abstract programs are correctly generated, but are mistakenly instantiated either by incorrect column names or filter conditions.

**Coverage Errors** These errors arise from implicit assumptions made by our parser: a) there is a long tail of unsupported operations that are not covered by our abstract programs; b) if entities are not correctly identified and linked, abstract programs cannot be correctly instantiated.

Table 4 shows the proportion of errors attested by the two attention models. We observe that structured attention suffers less from instantiation errors compared against the standard attention baseline, which points to the benefits of the structured alignment model.

## 5 Related Work

**Neural Semantic Parsing** We follow the line of work that applies sequence-to-sequence models (Sutskever et al., 2014) to semantic parsing (Jia and Liang, 2016; Dong and Lapata, 2016). Our work also relates to models which enforce type constraints (Yin and Neubig, 2017; Rabinovich et al., 2017; Krishnamurthy et al., 2017) so as to restrict the vast search space of potential programs. We use both methods as baselines to show that the structured bias introduced by our model can help our parser handle spurious programs in the setting of learning from denotations. Note that our alignment model can also be applied in the supervised case in order to help the parser rule out incorrect programs.

Earlier work has used lexicon mappings (Zettlemoyer and Collins, 2007; Wong and Mooney, 2007; Lu et al., 2008; Kwiatkowski et al., 2010) to model correspondences between programs and natural language. However, these methods cannot generalize to unseen tables where new relations and entities appear. To address this issue, Pasupat and Liang (2015) propose a floating parser which allows partial programs to be generated without being anchored to question tokens. In the same spirit, we use a sequence-to-sequence model to generate abstract programs while relying on explicit alignments to instantiate them. Besides semantic parsing, treating alignments as discrete latent variables has proved effective in other tasks like sequence transduction (Yu et al., 2016) and AMR parsing (Lyu and Titov, 2018).

**Learning from Denotations** To improve the efficiency of searching for consistent programs, Zhang et al. (2017) use a macro grammar induced from cached consistent programs. Unlike Zhang et al. (2017) who abstract entities and relations from logical forms, we take a step further and abstract the computation of row and column selection. Our work also differs from Pasupat and Liang (2016) who resort to manual annotations to alleviate spuriousness. Instead, we equip our parser with an inductive bias to rule out spurious programs during training. Recently, reinforcement learning based methods address the computational challenge by using a memory buffer (Liang et al., 2018) which stores consistent programs and an auxiliary reward function (Agarwal et al., 2019) which provides feedback to deal with spurious programs. Guu et al. (2017) employ various strategies to encourage even distributions over consistent programs in cases where the parser has been misled by spurious programs. Dasigi et al. (2019) use coverage of lexicon-like rules to guide the search of consistent programs.

## 6 Conclusions

In this paper, we proposed a neural semantic parser that learns from denotations using abstract programs and latent structured alignments. Our parser achieves state-of-the-art performance on two benchmarks, WIKITABLEQUESTIONS and WIKISQL. Empirical analysis shows that the inductive bias introduced by the alignment model helps our parser differentiate between correct and spurious programs. Alignments can exhibit different properties (e.g., monotonicity or bijectivity), depending on the meaning representation language (e.g., logical forms or SQL), the definition of abstract programs, and the domain at hand. We believe that these properties can be often captured within a probabilistic alignment model and hence provide a useful inductive bias to the parser.

## Acknowledgements

We would like to thank the anonymous reviewers for their valuable comments. We gratefully acknowledge the support of the European Research Council (Titov: ERC StG BroadSem 678254; Lapata: ERC CoG TransModal 681760) and the Dutch National Science Foundation (NWO VIDI 639.022.518).

## References

Rishabh Agarwal, Chen Liang, Dale Schuurmans, and Mohammad Norouzi. 2019. Learning to generalize from sparse and underspecified rewards. In *Proc. of ICML*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proc. of EMNLP*.

Li Zhang Karthik Ramanathan Sesh Sadasivam Rui Zhang Catherine Finegan-Dollak, Jonathan K. Kummerfeld and Dragomir Radev. 2018. Improving text-to-sql evaluation methodology. In *Proc. of ACL*.

Pradeep Dasigi, Matt Gardner, Shikhar Murty, Luke Zettlemoyer, and Eduard Hovy. 2019. Iterative

search for weakly supervised semantic parsing. In *Proc. of NAACL*.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proc. of ACL*.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In *Proc. of ACL*.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. Allennlp: A deep semantic natural language processing platform.

Omer Goldman, Veronica Latcinnik, Ehud Nave, Amir Globerson, and Jonathan Berant. 2018. Weakly supervised semantic parsing with abstract examples. In *Proc. of ACL*.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Edward Grefenstette, Karl Moritz Hermann, Mustafa Suleyman, and Phil Blunsom. 2015. Learning to transduce with unbounded memory. In *Proc. of NeurIPS*, pages 1828–1836.

Kelvin Guu, Panupong Pasupat, Evan Zheran Liu, and Percy Liang. 2017. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In *Proc. of ACL*.

Till Haug, Octavian-Eugen Ganea, and Paulina Grnarova. 2018. Neural multi-step reasoning for question answering on semi-structured tables. *ECIR*.

Jonathan Herzig and Jonathan Berant. 2018. Decoupling structure and lexicon for zero-shot semantic parsing. In *Proc. of EMNLP*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Po-Sen Huang, Chenglong Wang, Rishabh Singh, Wen tau Yih, and Xiaodong He. 2018. Natural language to structured query generation via meta-learning. In *Proc. of NAACL*.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proc. of ACL*.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M Rush. 2017. Structured attention networks. In *Proc. of ICLR*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proc. of ICLR*.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proc. of EMNLP*.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic ccg grammars from logical form with higher-order unification. In *Proc. of EMNLP*.

Chen Liang, Mohammad Norouzi, Jonathan Berant, Quoc V Le, and Ni Lao. 2018. Memory augmented policy optimization for program synthesis and semantic parsing. In *Proc. of NeurIPS*.

P. Liang, M. I. Jordan, and D. Klein. 2011. Learning dependency-based compositional semantics. In *Proc. of ACL*.

Wei Lu, Hwee Tou Ng, Wee Sun Lee, and Luke S Zettlemoyer. 2008. A generative model for parsing natural language to meaning representations. In *Proc. of EMNLP*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proc. of EMNLP*.

Chunchuan Lyu and Ivan Titov. 2018. Amr parsing as graph prediction with latent alignment. In *Proc. of ACL*.

Arvind Neelakantan, Quoc V Le, Martin Abadi, Andrew McCallum, and Dario Amodei. 2017. Learning a natural language interface with neural programmer. In *Proc. of ICLR*.

Maxwell Nye, Luke Hewitt, Joshua Tenenbaum, and Armando Solar-Lezama. 2019. Learning to infer program sketches. *Proc. of ICML*.

P. Pasupat and P. Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proc. of ACL*.

Panupong Pasupat and Percy Liang. 2016. Inferring logical forms from denotations. In *Proc. of ACL*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proc. of EMNLP*.

Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract syntax networks for code generation and semantic parsing. In *Proc. of ACL*.

Tianze Shi, Kedar Tatwawadi, Kaushik Chakrabarti, Yi Mao, Oleksandr Polozov, and Weizhu Chen. 2018. Incsql: Training incremental text-to-sql parsers with non-deterministic oracles. *arXiv preprint arXiv:1809.05054*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958.

Yibo Sun, Duyu Tang, Nan Duan, Jianshu Ji, Guihong Cao, Xiaocheng Feng, Bing Qin, Ting Liu, and Ming Zhou. 2018. Semantic parsing with syntax- and table-aware sql generation. In *Proc. of ACL*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proc. of NeurIPS*.

Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Efficient inference and structured learning for semantic role labeling. *TACL*.

Martin J Wainwright, Michael I Jordan, et al. 2008. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305.

Chenglong Wang, Marc Brockschmidt, and Rishabh Singh. 2017. Pointing out SQL queries from text. Technical report, MSR.

Yuk Wah Wong and Raymond Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proc. of ACL*.

Xiaojun Xu, Chang Liu, and Dawn Song. 2017. Sqlnet: Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv:1711.04436*.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *Proc. of ACL*.

Lei Yu, Jan Buys, and Phil Blunsom. 2016. Online segment to segment neural transduction. In *Proc. of EMNLP*.

Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. 2018. Typesql: Knowledge-based type-aware neural text-to-sql generation. In *Proc. of NAACL*.

Luke Zettlemoyer and Michael Collins. 2007. Online learning of relaxed ccg grammars for parsing to logical form. In *Proc. of EMNLP/CoNLL*.

Luke S Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: structured classification with probabilistic categorial grammars. In *Proc. of UAI*.

Yuchen Zhang, Panupong Pasupat, and Percy Liang. 2017. Macro grammars and holistic triggering for efficient semantic parsing. In *Proc. of EMNLP*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.

## A Grammars

We created our grammars following Zhang et al. (2017) and Liang et al. (2018). Compared with Liang et al. (2018), we additionally support disjunction(*OR*) and conjunction(*AND*). Some functions are pruned based on their effect on coverage, which is the proportion of questions that obtain at least one consistent program. "same_as" function (Liang et al., 2018) is excluded since it introduces too many spurious programs while contributing little to the coverage. For the same reason, conjunction(*AND*) is not used in WIKITABLEQUESTIONS and disjunction(*OR*) is not used in WIKISQL.

We also include non-terminals of function types in production rules (Krishnamurthy et al., 2017). For instance, function "ROW → *first*(LIST[ROW])" selects the first row from a list of rows and will lead to the production rule "ROW → *first*". In the paper, we eliminate the function type for simplicity. Practically, we use two production rules to represent the function: ROW → <ROW: LIST[ROW] > and <ROW:LIST[ROW]> → *first* , where < ROW: LIST[ROW] > is an abstract function type.

## B Search for Consistent Programs

We enumerate all possible programs in two stages using the abstract and instantiation grammars. To constrain the space and make the search process tractable, we restrict the maximal number of production rules for generating abstract programs during the first stage. It is based on the observation that the abstract programs which can be successfully instantiated into *correct* programs are usually not very complex. In other words, the *consistent* programs that are instantiated by long abstract programs are very likely to be spurious. For instance, programs like "select( previous( next( previous(argmax [all_rows], column:silver) column:bronze)" are unlikely to have a corresponding question. Specifically, we set the maximal number of production rules for generating abstract programs to 6 and 9, which leads to search time of around 7 and 10 hours for WIKISQL and WIKITABLEQUESTIONS respectively, using a single CPU. Note that this needs to be done only once.

## C Hyperparameters

Models used in WIKITABLEQUESTIONS and WIKISQL share similar hyperparameters which are listed in Table 5. Our input embeddings are obtained by a linear projection from the fixed pre-trained embedding (Pennington et al., 2014). Word Indicator refers to the indicator feature of whether a word appears in the table; Column Indicator refers to the indicator feature of whether at least one entity in a column appears in the question. All MLPs mentioned in the paper have the

| Hyperparameters | WTQ | WIKISQL |
|---|---|---|
| Input Fixed Embedding Size | 300 | 300 |
| Input Linear Projection Size | 256 | 256 |
| POS Embedding Size | 64 | - |
| Production Rule Embedding Size | 436 | 328 |
| Column Type Embedding Size | 16 | 16 |
| Word Indictor Embedding Size | 16 | 16 |
| Column Indictor Embedding Size | 16 | 16 |
| Operator Embedding Size | 128 | 128 |
| Encoder Hidden Size | 256 | 256 |
| Encoder Dropout | 0.45 | 0.35 |
| Decoder Hidden Size | 218 | 164 |
| AP Encoder Hidden Size | 218 | 164 |
| AP Encoder Dropout | 0.25 | 0.25 |
| MLP Hidden Size | 436 | 328 |
| MLP Dropout | 0.25 | 0.2 |

Table 5: Hyperparameters for WTQ (WIKITABLEQUESTIONS) and WIKISQL. AP Encoder is the encoder representing the abstract programs we generate.

.

same hidden size and dropout rate. During decoding, we choose the top-6 abstract programs to instantiate via beam search.

## D Alignments

If a row slot is instantiated with the special condition 'all_rows', then it is possible that the semantics of this slot is implicit. For instance, the question "which driver completed the least number of laps? " should first be mapped to the abstract program "select (argmin (#*row_slot*, #*column_slot*), #*column_slot* )" which is then instantiated to the correct program "select (argmin (all_rows, column:laps) column:driver)". The row slot in the abstract program is instantiated with 'all_rows', but this is not explicitly expressed in the question.

To make it compatible with our constraints in Equation (6) and (7), we add a special token "ALL_ROW" at the end of each question. If a row slot is aligned with this token, then it is expected to be instantiated with the special condition 'all_rows'. Specifically, this special token is mapped to a learnable vector during instantiations. Our alignment needs to learn to align this special token with a row slot if this row slot should be instantiated with the condition 'all_rows'.

# Broad-Coverage Semantic Parsing as Transduction

**Sheng Zhang** **Xutai Ma** **Kevin Duh** **Benjamin Van Durme**
Johns Hopkins University
{zsheng2, xutai_ma}@jhu.edu
{kevinduh, vandurme}@cs.jhu.edu

## Abstract

We unify different broad-coverage semantic parsing tasks under a transduction paradigm, and propose an attention-based neural framework that *incrementally* builds a meaning representation via a sequence of semantic relations. By leveraging multiple attention mechanisms, the transducer can be effectively trained without relying on a pre-trained aligner. Experiments conducted on three separate broad-coverage semantic parsing tasks – AMR, SDP and UCCA – demonstrate that our attention-based neural transducer improves the state of the art on both AMR and UCCA, and is competitive with the state of the art on SDP.

## 1 Introduction

Broad-coverage semantic parsing aims at mapping *any* natural language text, regardless of its domain, genre, or even the language itself, into a general-purpose meaning representation. As a long-standing topic of interest in computational linguistics, broad-coverage semantic parsing has targeted a number of meaning representation frameworks, including CCG (Steedman, 1996, 2001), DRS (Kamp and Reyle, 1993; Bos, 2008), AMR (Banarescu et al., 2013), UCCA (Abend and Rappoport, 2013), SDP (Oepen et al., 2014, 2015), and UDS (White et al., 2016).[1] Each of these frameworks has their specific formal and linguistic assumptions. Such framework-specific "*balkanization*" results in a variety of framework-specific parsing approaches, and the state-of-the-art semantic parser for one framework is not always applicable to another. For instance, the state-of-the-art approaches to SDP parsing (Dozat and Manning, 2018; Peng et al., 2017a) are not directly transferable to AMR and UCCA because of the lack of explicit alignments between tokens in the sentence and nodes in the semantic graph.

While transition-based approaches are adaptable to different broad-coverage semantic parsing tasks (Wang et al., 2018; Hershcovich et al., 2018; Damonte et al., 2017), when it comes to representations such as AMR whose nodes are *unanchored* to tokens in the sentence, a pre-trained aligner has to be used to produce the reference transition sequences (Wang et al., 2015; Damonte et al., 2017; Peng et al., 2017b). In contrast, there are attempts to develop attention-based approaches in a graph-based parsing paradigm (Dozat and Manning, 2018; Zhang et al., 2019), but they lack parsing incrementality, which is advocated in terms of computational efficiency and cognitive modeling (Nivre, 2004; Huang and Sagae, 2010).

In this paper, we approach different broad-coverage semantic parsing tasks under a unified framework of transduction. We propose an attention-based neural transducer that extends the two-stage semantic parser of Zhang et al. (2019) to directly transduce input text into a meaning representation in *one* stage. This transducer has properties of both transition-based approaches and graph-based approaches: on the one hand, it builds a meaning representation incrementally via a sequence of semantic relations, similar to a transition-based parser; on the other hand, it leverages multiple attention mechanisms used in recent graph-based parsers, thereby removing the need for pre-trained aligners.

Requiring only minor task-specific adaptations, we apply this framework to three separate broad-coverage semantic parsing tasks: AMR, SDP, and UCCA. Experimental results show that our neural transducer outperforms the state-of-the-art parsers on AMR (77.0% F1 on LDC2017T10 and 71.3%

[1] Abbreviations respectively denote: Combinatory Categorical Grammar, Discourse Representation Theory, Abstract Meaning Representation, Universal Conceptual Cognitive Annotation, Semantic Dependency Parsing, and Universal Decompositional Semantics.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3786–3798, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

F1 on LDC2014T12) and UCCA (76.6% F1 on the English-Wiki dataset v1.2), and is competitive with the state of the art on SDP (92.2% F1 on the English DELPH-IN MRS dataset).

## 2 Background and Related Work

We provide summary background on the meaning representations we target, and review related work on parsing for each.

**Abstract Meaning Representation** (AMR; Banarescu et al., 2013) encodes sentence-level semantics, such as predicate-argument information, reentrancies, named entities, negation and modality, into a rooted, directed, and usually acyclic graph with node and edge labels. AMR graphs abstract away from syntactic realizations, i.e., there is no explicit correspondence between elements of the graph and the surface utterance. Fig. 1(a) shows an example AMR graph.

Since its first general release in 2014, AMR has been a popular target of data-driven semantic parsing, notably in two SemEval shared tasks (May, 2016; May and Priyadarshi, 2017). Graph-based parsers build AMRs by identifying concepts and scoring edges between them, either in a pipeline (Flanigan et al., 2014), or jointly (Zhou et al., 2016; Lyu and Titov, 2018; Zhang et al., 2019). This two-stage parsing process limits the parser incrementality. Transition-based parsers either transform dependency trees into AMRs (Wang et al., 2015, 2016; Goodman et al., 2016), or employ transition systems specifically tailored to AMR parsing (Damonte et al., 2017; Ballesteros and Al-Onaizan, 2017). Transition-based parsers rely on pre-trained aligner produce the reference transitions. Grammar-based parsers leverage external semantic resources to derive AMRs compositionally based on CCG rules (Artzi et al., 2015), or SHRG rules (Peng et al., 2015). Another line of work uses neural model translation models to convert sentences into *linearized* AMRs (Barzdins and Gosko, 2016; Peng et al., 2017b), but has relied on data augmentation to produce effective parsers (van Noord and Bos, 2017; Konstas et al., 2017). Our parser differs from the previous ones in that it has incrementality without relying on pre-trained aligners, and can be effectively trained without data augmentation.

**Semantic Dependency Parsing** (SDP) was introduced in 2014 and 2015 SemEval shared tasks (Oepen et al., 2014, 2015). It is centered around three semantic formalisms – DM (DELPH-IN MRS; Flickinger et al., 2012; Oepen and Lønning, 2006), PAS (Predicate-Argument Structures; Miyao and Tsujii, 2004), and PSD (Prague Semantic Dependencies; Hajič et al., 2012) – representing predicate-argument relations between content words in a sentence. Their annotations have been converted into bi-lexical dependencies, forming directed graphs whose nodes injectively correspond to surface lexical units, and edges represent semantic relations between nodes. In this work, we focus on only the DM formalism. Fig. 1(b) shows an example DM graph.

Most recent parsers for SDP are graph-based: Peng et al. (2017a, 2018) use a max-margin classifier on top of a BiLSTM, with the factored score for each graph over predicates, unlabeled arcs, and arc labels. Multi-task learning approaches and disjoint data have been used to improve the parser performance. Dozat and Manning (2018) extend an LSTM-based syntactic dependency parser to produce graph-structured dependencies, and carefully tune it to state of the art performance. Wang et al. (2018) extend the transition system of Choi and McCallum (2013) to produce non-projective trees, and use improved versions of stack-LSTMs (Dyer et al., 2015) to learn representation for key components. All of these are specialized for bi-lexical dependency parsing, whereas our parser can effectively produce both bi-lexical semantics graphs, and graphs that are less anchored to the surface utterance.

**Universal Conceptual Cognitive Annotation** (UCCA; Abend and Rappoport, 2013) targets a level of semantic granularity that abstracts away from syntactic paraphrases in a typologically-motivated, cross-linguistic fashion. Sentence representations in UCCA are directed acyclic graphs (DAG), where terminal nodes correspond to surface lexical tokens, and non-terminal nodes to semantic units that participate in super-ordinate relations. Edges are labeled, indicating the role of a child in the relation the parent represents. Fig. 1(c) shows an example UCCA DAG.

The first UCCA parser is proposed by Hershcovich et al. (2017), where they extend a transition system to produce DAGs. To leverage other semantic resources, Hershcovich et al. (2018) is one of the few attempts to present (lossy) conversion from AMR, SDP and Universal Dependencies (UD; Nivre et al., 2016) to a unified UCCA-

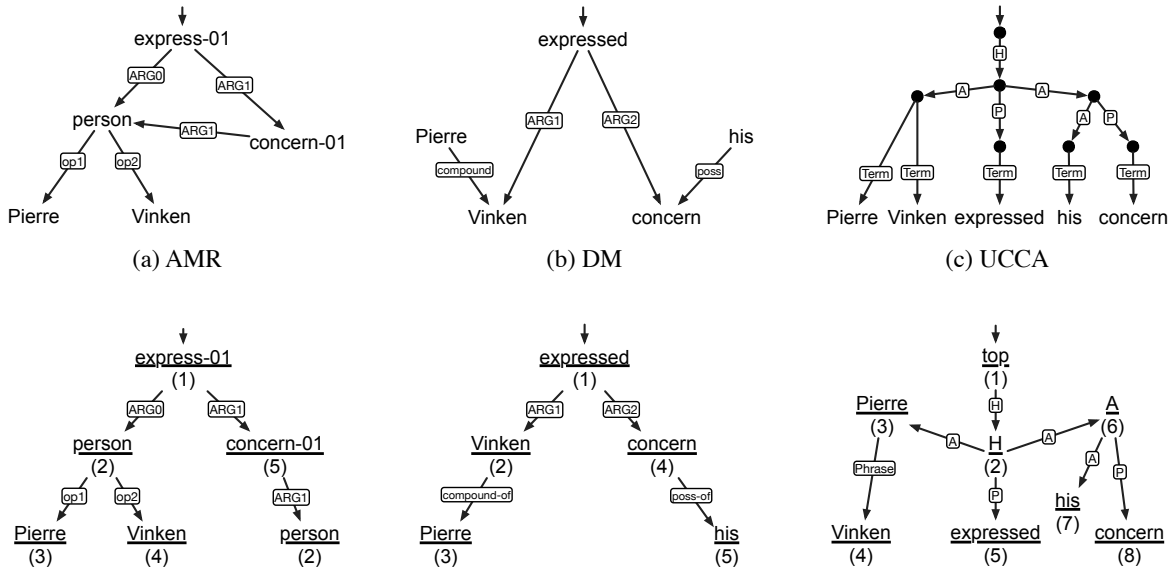


(a) AMR (b) DM (c) UCCA

(d) AMR arborescence (e) DM arborescence (f) UCCA arborescence

Figure 1: Meaning representation in the task-specific format – (a) AMR, (b) DM, and (c) UCCA – for an example sentence "*Pierre Vinken expressed his concern*". Meaning representation (d), (e) and (f) are in the unified arborescence format, which are converted from (a), (b) and (c) respectively.

based DAG format. They explore multi-task learning under the unified format. While multi-task learning improves UCCA parsing results, it shows poor performance on AMR, SDP and UD parsing. In contrast, different semantic parsing tasks are formalized in our unified transduction paradigm with no loss, and our approach achieves state-of-the-art or competitive performance on each task, using only single-task data.

## 3 Unified Transduction Problem

### 3.1 Unified Arborescence Format

We first introduce a unified target format for different broad-coverage semantic parsing tasks. Meaning representation in the unified format is an arborescence (aka, a directed rooted tree), which is converted from its corresponding task-specific semantic graph via the following *reversible* steps:

**AMR** Reentrancy is what can make an AMR graph not an arborescence (it introduces cycles). Following Zhang et al. (2019), we convert an AMR graph into an arborescence by duplicating nodes that have reentrant relations; that is, whenever a node has a reentrant relation, we make a copy of that node and use the copy to participate in the relation, thereby resulting in an arborescence. Next, in order to preserve the reentrancy information, we assign a node index to each node. Duplicated nodes are assigned the same index as the original node. Fig. 1(d) shows an AMR arborescence converted from Fig. 1(a): two "*person*" nodes have the same node index 2. The original AMR graph can be recovered by merging identically indexed nodes.

**DM** We first break the DM graph into a set of *weakly* connected subgraphs. For each subgraph, if it has the *top* node, we treat *top* as root; otherwise, we treat the node with the max outdegree as root. We then run depth-first traversal over each subgraph from its root to yield an arborescence, and repeat the following three steps until no more edges can be added to the arborescence: (1) we run breadth-first traversal over the arborescence from the root until we find a node that has an incoming edge not belonging to the arborescence; (2) we reverse the edge and add a `-of` suffix to the edge label; (3) we run depth-first search from that node to include more edges to the arborescence. During the whole process, we add node indices and duplicate reentrant nodes in the same way as AMR conversion. Finally, we connect arborescences by adding a `null` edge from *top* to other arborescence roots. Fig. 1(e) shows a DM arborescence converted from Fig. 1(b). The original DM graph can be recovered by removing `null` edges, merging identically indexed nodes, and reversing edges with `-of` suffix.

**UCCA** To date, official UCCA evaluation only considers UCCA's *foundational* layer, which is al-

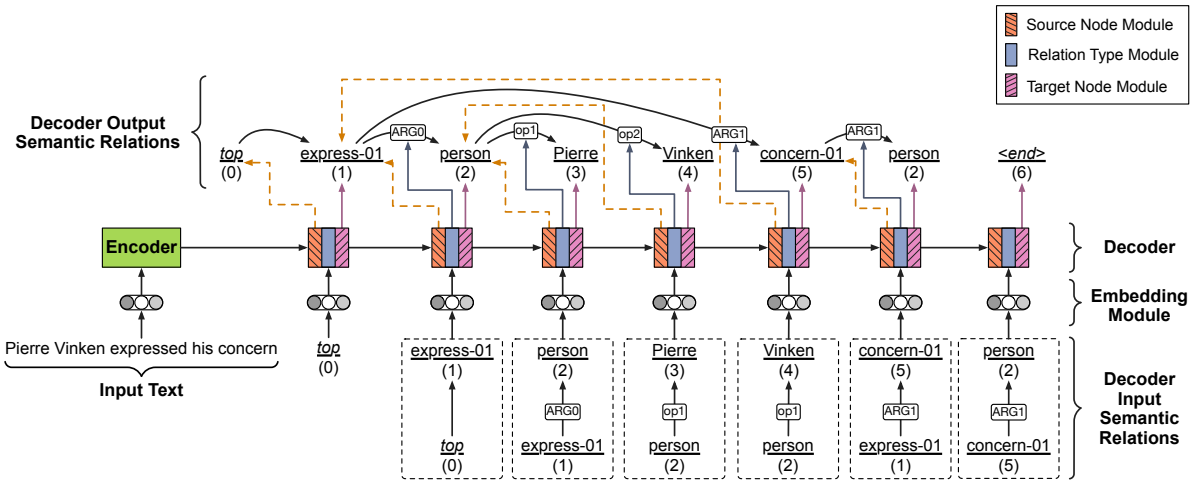


Figure 2: The encoder-decoder architecture of our attention-based neural transducer. An encoder encodes the input text into hidden states. A decoder is composed by three modules: a target node module, a relation type module, and a source node module. At each decoding time step, the decoder takes the previous semantic relation as input, and outputs a new semantic relation in a *factorized* way: firstly, the target node module produces a new target node; secondly, the source node module *points* to a preceding node as a new source node; finally, the relation type module predicts the relation type between source and target nodes.

ready an arborescence. We convert it to the unified arborescence format by first collapsing subgraphs of pre-terminal nodes: we replace each pre-terminal node with its first terminal node; if the pre-terminal node has other terminals, we add a special `phrase` edge from the first terminal node to other terminal nodes. The collapsing step largely reduces the number of terminal nodes in UCCA. We then add labels to the remaining non-terminal nodes. Each node label is simply the same as its incoming edge label. We find that adding node labels improves performance of our neural transducer (See Section 6.2 for the experimental results). Lastly, we add node indices in the same way as AMR conversion. Fig. 1(f) shows a DM arborescence converted from Fig. 1(c). The original UCCA DAG can be recovered by expanding pre-terminal subgraphs, and removing non-terminal node labels.

### 3.2 Problem Formalization

For any broad-coverage semantic parsing task, we denote the input text by $X$, and the output meaning representation in the unified arborescence format by $Y$, where $X$ is a sequence of tokens $\langle x_1, x_2, ..., x_n \rangle$ and $Y$ can be decomposed as a sequence of semantic relations $\langle y_1, y_2, ..., y_m \rangle$. A relation $y$ is a tuple $\langle u, d^u, r, v, d^v \rangle$, consisting of a source node label $u$, a source node index $d^u$, a relation type $r$, a target node label $v$, and a target node index $d^v$.

Let $\mathcal{Y}$ be the *output space*. The unified transduction problem is to seek the most-likely sequence of semantic relations $\hat{Y}$ given $X$:

$$\begin{aligned}\hat{Y} &= \arg\max_{Y \in \mathcal{Y}} \mathrm{P}(Y \mid X) \\ &= \arg\max_{Y \in \mathcal{Y}} \prod_{i}^{m} \mathrm{P}(y_i \mid y_{<i}, X)\end{aligned}$$

## 4 Transducer

To tackle the unified transduction problem, we introduce an attention-based neural transducer that extends Zhang et al. (2019)'s attention-based parser. Their attention-based parser addresses semantic parsing in a two-stage process: it first employs an extended variant of pointer-generator network (See et al., 2017) to convert the input text into a list of nodes, and then uses a deep biaffine graph-based parser (Dozat and Manning, 2016) with a maximum spanning tree (MST) algorithm to create edges. In contrast, our attention-based neural transducer directly transduces the input text into a meaning representation in *one* stage via a sequence of semantic relations. A high-level model architecture of our transducer is depicted in Fig. 2: an *encoder* first encodes the input text into hidden states; and then conditioned on the hidden states, at each decoding time step, a *decoder* takes the previous semantic relation as input, and outputs a new semantic relation, which includes a target node, a relation type, and a source node.

Specifically, there a significant difference between Zhang et al. (2019) and our model: Zhang et al. (2019) first predicts nodes, and then edges. These two stages are done *separately* (except that a shared encoder is used). At the node prediction stage, their model has no knowledge of edges, and therefore node prediction is performed purely based previous nodes. At the edge prediction stage, their model predicts the head of each node in parallel. Head prediction of one node has no constrains or impact on another. As a result, MST algorithms have to be used to search for a valid prediction. In comparison, our model does not have two separate stages for node and edge prediction. At each decoding step, our model predicts not only a node, but also the incoming edge to the node, which includes a source and a relation type. See Fig. 2 for an example. The predicted node and incoming edge together with previous predictions form a partial semantic graph, which is used as input of the next decoding step for the next node and incoming edge prediction. Our model therefore makes predictions based on the partial semantic graph, which helps prune the output space for both nodes and edges. Since at each decoding step, we assume the incoming edge is always from a preceding node (see Section 4.3 for the details), the predicted semantic graph is guaranteed to be a valid arborescence, and a MST algorithm is no longer needed.

### 4.1 Encoder

At the encoding stage, we employ an encoder embedding module to convert the input text into vector representations, and a BiLSTM is used to encode vector representations into hidden states.

**Encoder Embedding Module** concatenates word-level embeddings from GloVe (Pennington et al., 2014) and BERT[2] (Devlin et al., 2018), char-level embeddings from CharCNN (Kim et al., 2016), and randomly initialized embeddings for POS tags.

For AMR, it includes extra randomly initialized embeddings for anonymization indicators that tell the encoder whether a token is an anonymized token from preprocessing.

For UCCA, it includes extra randomly initialized embeddings for NER tags, syntactic dependency labels, punctuation indicators, and shapes that are provided in the UCCA official dataset.

**Multi-layer BiLSTM** (Hochreiter and Schmidhuber, 1997) is defined as:

$$\mathbf{s}_t^l = \begin{bmatrix} \overrightarrow{\mathbf{s}}_t^l \\ \overleftarrow{\mathbf{s}}_t^l \end{bmatrix} = \begin{bmatrix} \overrightarrow{\text{LSTM}}(\mathbf{s}_t^{l-1}, \mathbf{s}_{t-1}^l) \\ \overleftarrow{\text{LSTM}}(\mathbf{s}_t^{l-1}, \mathbf{s}_{t+1}^l) \end{bmatrix}, \quad (1)$$

where $\mathbf{s}_t^l$ is the $l$-th layer hidden state at time step $t$; $\mathbf{s}_i^t$ is the embedding module output for token $x_t$.

### 4.2 Decoder

**Decoder Embedding Module** at decoding time step $i$ converts elements in the input semantic relation $\langle u_i, d_i^u, r_i, v_i, d_i^v \rangle$ into vector representations $\langle \mathbf{u}_i, \mathbf{d}_i^u, \mathbf{r}_i, \mathbf{v}_i, \mathbf{d}_i^v \rangle$:[3]

$\mathbf{u}_i$ and $\mathbf{v}_i$ are concatenations of word-level embeddings from GloVe, char-level embeddings from CharCNN, and randomly initialized embeddings for POS tags. POS tags for source and target nodes are inferred at runtime: if a node is copied from input text, the POS tag of the corresponding token is used; if it is copied from a preceding node, the POS tag of the preceding node is used; otherwise, an `UNK` tag is used.

$\mathbf{d}_i^u, \mathbf{d}_i^v$ and $\mathbf{r}_i$ are randomly initialized embeddings for source node index, target node index, and relation type.

Next, the decoder outputs a new semantic relation in a *factorized* way depicted in Fig. 2: First, a target node module takes vector representations of the previous semantic relation, and predicts a target node label as well as its index. Then, a source node module predicts a source node via *pointing* to a preceding node. Lastly, a relation type module takes the predicted source and target nodes, and predicts the relation type between them.

**Target Node Module** converts vector representations of the input semantic relation into a hidden state $\mathbf{z}_i$ in the following way:

$$\mathbf{z}_i = \text{FFN}^{(\text{relation})}([\mathbf{h}_i^l; \mathbf{c}_i; \mathbf{r}_i; \mathbf{u}_i; \mathbf{d}_i^u]) \quad (2)$$

$$\mathbf{h}_i^l = \text{LSTM}(\mathbf{h}_i^{l-1}, \mathbf{h}_{i-1}^l) \quad (3)$$

$$\text{FFN}(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b} \quad (4)$$

where an $l$-layer LSTM generates contextual representation $\mathbf{h}_i^l$ for *target* node $v_i$ (for initialization, $\mathbf{h}_i^0 = [\mathbf{v}_i; \mathbf{d}_i^v]$, $\mathbf{h}_0^l = [\overleftarrow{\mathbf{s}}_1^l; \overrightarrow{\mathbf{s}}_n^l]$). A feed-forward neural network $\text{FFN}^{(\text{relation})}$ generates the hidden

[2] We use average pooling in the same way as Zhang et al. (2019) to get word-level embeddings from BERT.

[3] While training, the input semantic relation is from the reference sequence of relations; at test time, it is the previous decoder output semantic relation.

state $\mathbf{z}_i$ of the input semantic relation by combining contextual representation $\mathbf{h}_i^l$ for target node $v_i$, encoder context vector $\mathbf{c}_i$, and vector representations $\mathbf{r}_i, \mathbf{u}_i, \mathbf{d}_i^u$ for relation type $r_i$, source node label $u_i$ and source node index $d_i^u$.

Encoder context vector $\mathbf{c}_i$ is a weighted-sum of encoder hidden states $\mathbf{s}_{1:n}^l$. The weight is attention $\mathbf{a}_i^{(\text{enc})}$ from the decoder at decoding step $i$ to encoder hidden states:

$$\mathbf{a}_i^{(\text{enc})} = \text{softmax}\big(\text{MLP}^{(\text{enc})}([\mathbf{h}_i^l; \mathbf{s}_{1:n}^l])\big) \quad (5)$$

$$\text{MLP}(\mathbf{x}) = \text{ELU}(\mathbf{W}\mathbf{x} + \mathbf{b}) \quad (6)$$

Given the hidden state $\mathbf{z}_i$ for input semantic relation, we use an extended variant of pointer-generator network to compute the probability distribution of next target node label $v_{i+1}$:

$$\text{P}(v_{i+1}) = p_{\text{gen}}\mathbf{p}_i^{(\text{vocab})} \oplus p_{\text{enc}}\mathbf{a}_i^{(\text{enc})} \oplus p_{\text{dec}}\mathbf{a}_i^{(\text{dec})} \quad (7)$$

$$\mathbf{p}_i^{(\text{vocab})} = \text{softmax}\big(\text{FFN}^{(\text{vocab})}(\mathbf{z}_i)\big) \quad (8)$$

$$\mathbf{a}_i^{(\text{dec})} = \text{softmax}\big(\text{MLP}^{(\text{dec})}([\mathbf{z}_i; \mathbf{z}_{1:i-1}])\big) \quad (9)$$

$$[p_{\text{gen}}, p_{\text{enc}}, p_{\text{dec}}] = \text{softmax}\big(\text{FFN}^{(\text{switch})}(\mathbf{z}_i)\big) \quad (10)$$

$\text{P}(v_{i+1})$ is a hybrid of three parts: (1) emitting a new node label from a pre-defined vocabulary via probability distribution $\mathbf{p}_i^{(\text{vocab})}$; (2) copying a token from the encoder input text as node label via encoder-side attention $\mathbf{a}_i^{(\text{enc})}$; and (3) copying a node label from preceding target nodes via decoder-side attention $\mathbf{a}_i^{(\text{dec})}$. Scalars $p_{\text{gen}}, p_{\text{enc}}$ and $p_{\text{dec}}$ act as a soft switch to control the production of target node label from different sources.

The next target node index $d_{i+1}^v$ is assigned based on the following rule:

$$d_{i+1}^v = \begin{cases} d_j^v, \text{ if } v_{i+1} \text{ copies its antecedent } v_j. \\ i+1, \text{ otherwise.} \end{cases}$$

**Source Node Module** produces the next source node label $u_{i+1}$ via *pointing* to a node label among preceding *target* node labels (the dotted arrows shown in Fig. 2). The probability distribution of next source node label $u_{i+1}$ is defined as

$$\text{P}(u_{i+1}) = \text{softmax}\big(\text{BIAFFINE}(\mathbf{h}_{i+1}^{(\text{start})}, \mathbf{h}_{1:i}^{(\text{end})})\big) \quad (11)$$

where BIAFFINE is a biaffine function (Dozat and Manning, 2016). $\mathbf{h}_{i+1}^{(\text{start})}$ is the vector representation for the *start* of the pointer. $\mathbf{h}_{1:i}^{(\text{end})}$ are vector representations for possible *ends* of the pointer. They are computed by two multi-layer perceptrons:

$$\mathbf{h}_{i+1}^{(\text{start})} = \text{MLP}^{(\text{start})}(\mathbf{h}_{i+1}^l) \quad (12)$$

$$\mathbf{h}_{1:i}^{(\text{end})} = \text{MLP}^{(\text{end})}(\mathbf{h}_{1:i}^l) \quad (13)$$

Note that $\mathbf{h}_{i+1}^l$ is the LSTM hidden state for target node $v_{i+1}$, generated by Equation (3) in the target node module. We reuse LSTM hidden states from the target node module such that we can train the decoder modules jointly.

Then, the next source node index $d_{i+1}^u$ is the same as the target node the module points to.

**Relation Type Module** also reuses LSTM hidden states from the target node module to compute the probability distribution of next relation type $r_{i+1}$. Assuming that the source node module points to target node label $v_j$ as the next source node label, The next relation type probability distribution is computed by:

$$\text{P}(r_{i+1}) = \text{softmax}\big(\text{BILINEAR}(\mathbf{h}_{i+1}^{(\text{rel-src})}, \mathbf{h}_{i+1}^{(\text{rel-tgt})})\big) \quad (14)$$

$$\mathbf{h}_{i+1}^{(\text{rel-src})} = \text{MLP}^{(\text{rel-src})}(\mathbf{h}_j^l) \quad (15)$$

$$\mathbf{h}_{i+1}^{(\text{rel-tgt})} = \text{MLP}^{(\text{rel-tgt})}(\mathbf{h}_{i+1}^l) \quad (16)$$

### 4.3 Training

To ensure that at each decoding step, the source node can be found in the preceding nodes, we create the reference sequence of semantic relations by running a *pre-order* traversal over the reference arborescence. The pre-order traversal only determines the order between a node and its children. As for the order of its children, we sort them in alphanumerical order in the case of AMR, following Zhang et al. (2019). In the case of SDP, we sort the children based on their order in the input text. In the case of UCCA, we sort the children based on their UCCA node ID.

Given a training pair $\langle X, Y \rangle$, the optimization objective is to maximize the decomposed conditional log likelihood $\sum_i \log\big(\text{P}(y_i \mid y_{<i}, X)\big)$, which is approximated by:

$$\sum_i \log\big(\text{P}(u_i)\big) + \log\big(\text{P}(r_i)\big) + \log\big(\text{P}(v_i)\big) \quad (17)$$

We also employ label smoothing (Szegedy et al., 2016) to prevent overfitting, and include a coverage loss (See et al., 2017) to penalize repetitive nodes: $\text{covloss}_i = \sum_t \min(\mathbf{a}_i^{(\text{enc})}[t], \mathbf{cov}_i[t])$, where $\mathbf{cov}^i = \sum_{j=0}^{i-1} \mathbf{a}_j^{(\text{enc})}$.

### 4.4 Prediction

Our transducer at each decoding time step looks for the source node from the preceding nodes, which ensures that the output of a greedy search is already a valid arborescence $\hat{Y}$:

$$\mathrm{P}(\hat{Y} \mid X) = \prod_i \max_{u_i} \mathrm{P}(u_i) \max_{r_i} \mathrm{P}(r_i) \max_{v_i} \mathrm{P}(v_i)$$

Therefore, a MST algorithm such as the Chu-Liu-Edmonds algorithm at $\mathcal{O}(EV)$ used in Zhang et al. (2019) is no longer needed,[4] and the decoding speed of our transducer is $\mathcal{O}(V)$. Moreover, since our transducer builds the meaning representation via a sequence of semantic relations, we implement a beam search over relation in Algo. 1. Compared to the beam search of Zhang et al. (2019) that only returns top-$k$ nodes, our beam search finds the top-$k$ relation scores, which includes source nodes, relation types and target nodes.

**Algorithm 1:** Beam Search over Semantic Relations.

**Input :** The input text $X$.
**Output:** A sequence of relations $Y = \{y_1, ...y_m\}$.

```
// Initialization.
i, score ← 0, 0;
Y, finished ← {}, {};
beam← {{Y, score}};
// Encoding.
encode(X);
// Decoding.
for i ← 1 to MaxLength do
    new_beam ← {};
    {Y, score} = beam.pop();
    for v_i in topK(P(v_i)) do
        if v_i = EOS then
            finished.push({Y, score});
        else
            for u_i ← v_0 to v_{i-1} do
                for r_i in RelationTypeSet do
                    Y ← Y ∪ {⟨u_i, r_i, v_i⟩};
                    score ← score + log(P(u_i)) +
                      log(P(r_i)) + log(P(v_i));
                    new_beam.push({Y, score});
                end
            end
        end
    end
    beam ← new_beam.topK();
end
// Finishing.
while beam.not_empty() do
    {Y, score} ← beam.pop();
    finished.push({Y, score});
end
{Y, score} ← finished.topK(k=1);
return Y;
```

[4] $E$ denotes the number of edges. $V$ the number of nodes.

## 5 Data Pre- and Post-processing

**AMR** Pre- and post-processing steps are similar to those of Zhang et al. (2019): in preprocessing, we anonymize subgraphs of entities, remove senses, and convert resultant AMR graphs into the unified format; in post-processing, we assign the most frequent sense for nodes, restore Wikipedia links using the DBpedia Spotlight API (Daiber et al., 2013), add polarity attributes based on rules observed from training data, and recover the original AMR format from the unified format.

**DM** No pre- or post-processing is done to DM except converting them into the unified format, and recovering them from predictions.

**UCCA** During training, multi-sentence input text and its corresponding DAG are split into single-sentence training pairs based on rules observed from training data. At test time, we split multi-sentence input text, and join the predicted graphs into one. We also convert the original format to the unified format in preprocessing, and recover the original DAG format in post-processing.

| | | |
|---|---|---|
| **Hidden Size** | | |
| Glove | | 300 |
| BERT | | 1024 |
| POS / NER / Dep / Shapes | | 100 |
| Anonymization / Node index | | 50 |
| CharCNN kernel size | | 3 |
| CharCNN channel size | | 100 |
| Encoder | | 2@512 |
| Decoder | | 2@1024 |
| Biaffine input size | | 256 |
| Bilinear input size | AMR | 128 |
| | DM | 256 |
| | UCCA | 128 |
| **Optimizer** | | |
| Type | | ADAM |
| Learning rate | | 0.001 |
| Maximum gradient norm | | 5.0 |
| Coverage loss weight $\lambda$ | | 1.0 |
| Label smoothing $\epsilon$ | | 0.1 |
| Beam size | | 5 |
| Batch size | | 64 |
| Dropout rate | AMR | 0.33 |
| | DM | 0.2 |
| | UCCA | 0.33 |
| **Vocabulary** | | |
| Encoder-side vocab size | AMR 1.0 | 9200 |
| | AMR 2.0 | 18000 |
| | DM | 11000 |
| | UCCA | 10000 |
| Decoder-side vocab size | AMR 1.0 | 7300 |
| | AMR 2.0 | 12200 |
| | DM | 11000 |
| | UCCA | 10000 |

Table 1: Hyperparameter settings

## 6 Experiments

### 6.1 Data and Setup

We evaluate our approach on three separate broad-coverage semantic parsing tasks: (1) AMR 2.0 (LDC2017T10) and 1.0 (LDC2014T12); (2) the English DM dataset from SemEval 2015 Task 18 (LDC2016T10); (3) the UCCA English Wikipedia Corpus v1.2 (Abend and Rappoport, 2013). The train/dev/test split follows the official setup. Our model is trained on two GeForce GTX TITAN X GPUs with early stop based on the dev set. We fix BERT parameters similar to Zhang et al. (2019) due to the limited GPU memory. Hyperparameter setting for each task is provided in Table 1.

### 6.2 Results

| Data | Parser | F1(%) |
|---|---|---|
| AMR 2.0 | Cai and Lam (2019) | 73.2 |
| | Lyu and Titov (2018) | 74.4±0.2 |
| | Lindemann et al. (2019) | 75.3±0.1 |
| | Naseem et al. (2019) | 75.5 |
| | Zhang et al. (2019) | 76.3±0.1 |
| | - w/o beam search | 75.3±0.1 |
| | Ours | **77.0**±0.1 |
| | - w/o beam search | 76.4±0.1 |
| AMR 1.0 | Flanigan et al. (2016) | 66.0 |
| | Pust et al. (2015) | 67.1 |
| | Wang and Xue (2017) | 68.1 |
| | Guo and Lu (2018) | 68.3±0.4 |
| | Zhang et al. (2019) | 70.2±0.1 |
| | - w/o beam search | 69.2±0.1 |
| | Ours | **71.3**±0.1 |
| | - w/o beam search | 70.4±0.1 |

Table 2: SMATCH F1 on AMR 2.0 and 1.0 test sets. Standard deviation is computed over 3 runs.

**AMR** Table 2 compares our neural transducer to the previous best results (SMATCH F1, Cai and Knight, 2013) on AMR test sets. The transducer improves the state of the art on AMR 2.0 by 0.7% F1. On AMR 1.0 where training data is much smaller than AMR 2.0, it shows a larger improvement (1.1% F1) over the state of the art.

In Table 2, we also conduct ablation study on beam search to investigate contributions from the model architecture itself and the beam search algorithm. The transducer model without beam search is already better than the previous best parser that is equipped with beam search. When compared with the previous best parser without beam search, our model still has around 1.0% F1 improvement.

| Metric | L'18 | N'19 | Z'19 | Ours |
|---|---|---|---|---|
| SMATCH | 74 | 75 | 76 | **77** |
| Unlabeled | 77 | **80** | 79 | **80** |
| No WSD | 76 | 76 | 77 | **78** |
| Reentrancies | 52 | 56 | 60 | **61** |
| Concepts | **86** | **86** | 85 | **86** |
| Named Ent. | **86** | 83 | 78 | 79 |
| Wikification | 76 | 80 | **86** | **86** |
| Negation | 58 | 67 | 75 | **77** |
| SRL | 70 | **72** | 70 | 71 |

Table 3: Fine-grained F1 scores on the AMR 2.0 test set. L'18 is Lyu and Titov (2018); N'19 is Naseem et al. (2019); Z'19 is Zhang et al. (2019).

Table 3 summarizes the parser performance on each subtask using Damonte et al. (2017) evaluation tool. Our transducer outperforms Zhang et al. (2019) on all subtasks, but is still not close to Lyu and Titov (2018) on named entities due to the different preprocessing methods for anonymization.

| Parser | ID | OOD |
|---|---|---|
| Du et al. (2015) | 89.1 | 81.8 |
| Almeida and Martins (2015)$^{(open)}$ | 89.4 | 83.8 |
| Wang et al. (2018) | 90.3 | 84.9 |
| Peng et al. (2017a): BASIC | 89.4 | 84.5 |
| Peng et al. (2017a): FREDA3 | 90.4 | 85.3 |
| Peng et al. (2018) | 91.2 | 86.6 |
| Dozat and Manning (2018) | **93.7** | **88.9** |
| Ours | 92.2 | 87.1 |

Table 4: Labeled F1 (%) scores on the English DM in-domain (WSJ) and out-of-domain (Brown corpus) test sets. $^{(open)}$ denotes results from the open track.

**DM** Table 4 compares our neural transducer to the state of the art (labeled F1) on the English DM in-domain (ID) and out-of-domain (OOD) data. Except Dozat and Manning (2018), our transducer outperforms all other baselines, including FREDA3 of Peng et al. (2017a) and Peng et al. (2018), which leverage multi-task learning from different datasets. The best parser (Dozat and Manning, 2018) is specifically designed for bi-lexical dependencies, and is not directly applicable to other se-

mantic parsing tasks such as AMR and UCCA. In contrast, our transducer is more general, and is competitive to the best SDP parser.

| Parser | F1 (%) |
|---|---|
| Hershcovich et al. (2017) | 71.1 |
| Hershcovich et al. (2018): single | 71.2 |
| Hershcovich et al. (2018): MTL | 74.3 |
| Ours | **76.6**±0.1 |
| - w/o non-terminal node labels | 75.7±0.1 |

Table 5: Labeled F1 (%) scores for all edges including primary edges and remote edges. Standard deviation is computed over 3 runs.

**UCCA** Table 5 compares our results to the previous best published results (labeled F1 for all edges) on the English Wiki test set. Hershcovich et al. (2018) explore multi-task learning (MTL) to improve UCCA parsing, using AMR, DM and UD parsing as auxiliaries. While improvement is achieved UCCA parsing, their MTL model shows poor results on the auxiliary tasks: 64.7% unlabeled F1 on AMR, 27.2% unlabeled F1 on DM, and 4.9% UAS on UD. In comparison, our transducer improves the state of the art on AMR, and shows competitive results on DM. At the same time, it also outperforms the best published UCCA results by 2.3% F1. When converting UCCA DAGs to the unified format, we adopt a simple rule (Section 3.1) to add node labels to non-terminals. Table 5 shows that these node labels do improve the parsing performance from 75.7% to 76.6%.

### 6.3 Analysis

**Validity** Graph-based parsers like Dozat and Manning (2018); Zhang et al. (2019) make independent decisions on edge types. As a result, the same outgoing edge type can appear multiple times to a node. For instance, a node can have more than one `ARG1` outgoing edge. Although F1 scores can be computed for graphs with such kind of nodes, these graphs are in fact invalid mean representations. Our neural transducer incrementally builds meaning representations: at each decoding step, it takes a semantic relation as input, and has memory of preceding edge type information, which implicitly places constraints on edge type prediction. We compute the number of invalid graphs predicted by the parser of Zhang et al. (2019) and our neural transducer on the AMR 2.0 test set, and find that our neural transducer reduces the number of invalid graphs by 8%.

**Speed** Besides the improvement on parsing accuracy, we also significantly speed up parsing. Table 6 compares the parsing speed of our transducer and Zhang et al. (2019) on the AMR 2.0 test set, under the same environment setup. Without relying on MST algorithms to produce a valid arborescence, our transducer is able to parse at 1.7x speed.

| | Speed (tokens/sec) |
|---|---|
| Zhang et al. (2019) | 617 |
| Ours | **1076** |

Table 6: Parsing speed on the AMR 2.0 test set.

## 7 Conclusion

We cast three broad-coverage semantic parsing tasks into a unified transduction framework, and propose a neural transducer to tackle the problem. Given the input text, the transducer incrementally builds a meaning representation via a sequence of semantic relations. Experiments conducted on three tasks show that our approach improves the state of the art in both AMR and UCCA, and is competitive to the best parser in SDP.

This work can be viewed as a starting point for cross-framework semantic parsing. Also, compared with transition-based parsers (e.g. Damonte et al., 2017) and graph-based parsers (e.g. Dozat and Manning, 2018), our transductive framework does not require a pre-trained aligner, and it is capable of building a meaning representation that is less anchored to the input text. These advantages make it well suited to semantic parsing in cross-lingual settings (Zhang et al., 2018). In the future, we hope to explore its potential in cross-framework and cross-lingual semantic parsing.

## Acknowledgments

We thank the anonymous reviewers for their valuable feedback. This work was supported in part by the JHU Human Language Technology Center of Excellence, and DARPA LORELEI and AIDA. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes. The views and conclusions contained in this publication are those of the authors and should not be interpreted as representing official policies or endorsements of DARPA or the U.S. Government.

## References


Omri Abend and Ari Rappoport. 2013. Universal conceptual cognitive annotation (ucca). In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 228–238. Association for Computational Linguistics.

Mariana S. C. Almeida and André F. T. Martins. 2015. Lisbon: Evaluating TurboSemanticParser on multiple languages and out-of-domain data. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 970–973, Denver, Colorado. Association for Computational Linguistics.

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage ccg semantic parsing with amr. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1699–1710. Association for Computational Linguistics.

Miguel Ballesteros and Yaser Al-Onaizan. 2017. AMR parsing using stack-LSTMs. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1269–1275, Copenhagen, Denmark. Association for Computational Linguistics.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186. Association for Computational Linguistics.

Guntis Barzdins and Didzis Gosko. 2016. RIGA at SemEval-2016 task 8: Impact of Smatch extensions and character-level neural translation on AMR parsing accuracy. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1143–1147, San Diego, California. Association for Computational Linguistics.

Johan Bos. 2008. Wide-coverage semantic analysis with Boxer. In *Semantics in Text Processing. STEP 2008 Conference Proceedings*, pages 277–286. College Publications.

Deng Cai and Wai Lam. 2019. Core Semantic First: A Top-down Approach for AMR Parsing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing*, Hong Kong, China. Association for Computational Linguistics.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 748–752. Association for Computational Linguistics.

Jinho D. Choi and Andrew McCallum. 2013. Transition-based dependency parsing with selectional branching. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1052–1062, Sofia, Bulgaria. Association for Computational Linguistics.

Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N. Mendes. 2013. Improving efficiency and accuracy in multilingual entity extraction. In *Proceedings of the 9th International Conference on Semantic Systems (I-Semantics)*.

Marco Damonte, Shay B. Cohen, and Giorgio Satta. 2017. An incremental parser for abstract meaning representation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 536–546. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Timothy Dozat and Christopher D Manning. 2016. Deep biaffine attention for neural dependency parsing. *arXiv preprint arXiv:1611.01734*.

Timothy Dozat and Christopher D. Manning. 2018. Simpler but more accurate semantic dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 484–490. Association for Computational Linguistics.

Yantao Du, Fan Zhang, Xun Zhang, Weiwei Sun, and Xiaojun Wan. 2015. Peking: Building semantic dependency graphs with a hybrid parser. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 927–931, Denver, Colorado. Association for Computational Linguistics.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343, Beijing, China. Association for Computational Linguistics.

Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime Carbonell. 2016. Cmu at semeval-2016 task 8: Graph-based amr parsing with infinite ramp loss. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1202–1206. Association for Computational Linguistics.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436. Association for Computational Linguistics.

Dan Flickinger, Valia Kordoni, and Zhang Yi. 2012. DeepBank: A Dynamically Annotated Treebank of the Wall Street. In *Proceedings of the 11th International Workshop on Treebanks and Linguistic Theories*, pages 85–86, Lisbon, Portugal.

James Goodman, Andreas Vlachos, and Jason Naradowsky. 2016. Ucl+sheffield at semeval-2016 task 8: Imitation learning for amr parsing with an alpha-bound. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1167–1172. Association for Computational Linguistics.

Zhijiang Guo and Wei Lu. 2018. Better transition-based amr parsing with a refined search space. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1712–1722. Association for Computational Linguistics.

Jan Hajič, Eva Hajičová, Jarmila Panevová, Petr Sgall, Ondřej Bojar, Silvie Cinková, Eva Fučíková, Marie Mikulová, Petr Pajas, Jan Popelka, Jiří Semecký, Jana Šindlerová, Jan Štěpánek, Josef Toman, Zdeňka Urešová, and Zdeněk Žabokrtský. 2012. Announcing Prague Czech-English dependency treebank 2.0. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*, pages 3153–3160, Istanbul, Turkey. European Language Resources Association (ELRA).

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2017. A transition-based directed acyclic graph parser for UCCA. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1127–1138, Vancouver, Canada. Association for Computational Linguistics.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2018. Multitask parsing across semantic representations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 373–385, Melbourne, Australia. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Liang Huang and Kenji Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 1077–1086, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hans Kamp and Uwe Reyle. 1993. *From Discourse to Logic*. Dordrecht: Kluwer Academic Publishers.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 2741–2749. AAAI Press.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural amr: Sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 146–157. Association for Computational Linguistics.

Matthias Lindemann, Jonas Groschwitz, and Alexander Koller. 2019. Compositional semantic parsing across graphbanks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4576–4585, Florence, Italy. Association for Computational Linguistics.

Chunchuan Lyu and Ivan Titov. 2018. Amr parsing as graph prediction with latent alignment. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 397–407. Association for Computational Linguistics.

Jonathan May. 2016. SemEval-2016 task 8: Meaning representation parsing. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1063–1073, San Diego, California. Association for Computational Linguistics.

Jonathan May and Jay Priyadarshi. 2017. SemEval-2017 task 9: Abstract meaning representation parsing and generation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 536–545, Vancouver, Canada. Association for Computational Linguistics.

Yusuke Miyao and Jun'ichi Tsujii. 2004. Deep linguistic analysis for the accurate identification of predicate-argument relations. In *Proceedings of Coling 2004*, pages 1392–1398, Geneva, Switzerland. COLING.

Tahira Naseem, Abhishek Shah, Hui Wan, Radu Florian, Salim Roukos, and Miguel Ballesteros. 2019. Rewarding Smatch: Transition-based AMR parsing with reinforcement learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4586–4592, Florence, Italy. Association for Computational Linguistics.

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*, IncrementParsing '04,

pages 50–57, Stroudsburg, PA, USA. Association for Computational Linguistics.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*.

Rik van Noord and Johan Bos. 2017. Neural semantic parsing by character-based translation: Experiments with abstract meaning representations. *Computational Linguistics in the Netherlands Journal*, 7:93–108.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinkova, Dan Flickinger, Jan Hajic, and Zdenka Uresova. 2015. SemEval 2015 task 18: Broad-coverage semantic dependency parsing. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 915–926, Denver, Colorado. Association for Computational Linguistics.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajic, Angelina Ivanova, and Yi Zhang. 2014. SemEval 2014 task 8: Broad-coverage semantic dependency parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 63–72, Dublin, Ireland. Association for Computational Linguistics.

Stephan Oepen and Jan Tore Lønning. 2006. Discriminant-based MRS banking. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy. European Language Resources Association (ELRA).

Hao Peng, Sam Thomson, and Noah A. Smith. 2017a. Deep multitask learning for semantic dependency parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2037–2048. Association for Computational Linguistics.

Hao Peng, Sam Thomson, Swabha Swayamdipta, and Noah A. Smith. 2018. Learning joint semantic parsers from disjoint data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1492–1502, New Orle/ans, Louisiana. Association for Computational Linguistics.

Xiaochang Peng, Linfeng Song, and Daniel Gildea. 2015. A synchronous hyperedge replacement grammar based approach for AMR parsing. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 32–41, Beijing, China. Association for Computational Linguistics.

Xiaochang Peng, Chuan Wang, Daniel Gildea, and Nianwen Xue. 2017b. Addressing the data sparsity issue in neural amr parsing. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 366–375. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics.

Michael Pust, Ulf Hermjakob, Kevin Knight, Daniel Marcu, and Jonathan May. 2015. Parsing english into abstract meaning representation using syntax-based machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1143–1154. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083. Association for Computational Linguistics.

M. Steedman. 1996. *Surface Structure and Interpretation*. Linguistic inquiry monographs. MIT Press.

M. Steedman. 2001. *The Syntactic Process*. A Bradford book. MIT Press.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Chuan Wang, Sameer Pradhan, Xiaoman Pan, Heng Ji, and Nianwen Xue. 2016. Camr at semeval-2016 task 8: An extended transition-based amr parser. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1173–1178. Association for Computational Linguistics.

Chuan Wang and Nianwen Xue. 2017. Getting the most out of amr parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1257–1268. Association for Computational Linguistics.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015. A transition-based algorithm for amr parsing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 366–375. Association for Computational Linguistics.

Yuxuan Wang, Wanxiang Che, Jiang Guo, and Ting Liu. 2018. A neural transition-based approach for semantic dependency graph parsing. In *AAAI Conference on Artificial Intelligence*.

Aaron Steven White, Drew Reisinger, Keisuke Sakaguchi, Tim Vieira, Sheng Zhang, Rachel Rudinger, Kyle Rawlins, and Benjamin Van Durme. 2016. Universal decompositional semantics on universal dependencies. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1713–1723, Austin, Texas. Association for Computational Linguistics.

Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019. AMR parsing as sequence-to-graph transduction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 80–94, Florence, Italy. Association for Computational Linguistics.

Sheng Zhang, Xutai Ma, Rachel Rudinger, Kevin Duh, and Benjamin Van Durme. 2018. Cross-lingual decompositional semantic parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1664–1675, Brussels, Belgium. Association for Computational Linguistics.

Junsheng Zhou, Feiyu Xu, Hans Uszkoreit, Weiguang QU, Ran Li, and Yanhui Gu. 2016. Amr parsing with an incremental joint model. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 680–689. Association for Computational Linguistics.

# Core Semantic First: A Top-down Approach for AMR Parsing*

**Deng Cai**
The Chinese University of Hong Kong
thisisjcykcd@gmail.com

**Wai Lam**
The Chinese University of Hong Kong
wlam@se.cuhk.edu.hk

## Abstract

We introduce a novel scheme for parsing a piece of text into its Abstract Meaning Representation (AMR): Graph Spanning based Parsing (GSP). One novel characteristic of GSP is that it constructs a parse graph incrementally in a top-down fashion. Starting from the root, at each step, a new node and its connections to existing nodes will be jointly predicted. The output graph spans the nodes by the distance to the root, following the intuition of first grasping the main ideas then digging into more details. The *core semantic first* principle emphasizes capturing the main ideas of a sentence, which is of great interest. We evaluate our model on the latest AMR sembank and achieve the state-of-the-art performance in the sense that no heuristic graph re-categorization is adopted. More importantly, the experiments show that our parser is especially good at obtaining the core semantics.

## 1 Introduction

Abstract Meaning Representation (AMR) (Banarescu et al., 2013) is a semantic formalism that encodes the meaning of a sentence as a rooted labeled directed graph. As illustrated by an example in Figure 1, AMR abstracts away from the surface forms in text, where the root serves as a rudimentary representation of the overall focus while the details are elaborated as the depth of the graph increases. AMR has been proved useful for many downstream NLP tasks, including text summarization (Liu et al., 2015; Hardy and Vlachos, 2018) and question answering (Mitra and Baral, 2016).

The task of AMR parsing is to map natural language strings to AMR semantic graphs automatically. Compared to constituent parsing (Zhang and Clark, 2009) and dependency parsing (Kübler et al., 2009), AMR parsing is considered more challenging due to the following characteristics: (1) The nodes in AMR have no explicit alignment to text tokens; (2) The graph structure is more complicated because of frequent reentrancies and non-projective arcs; (3) There is a large and sparse vocabulary of possible node types (concepts).

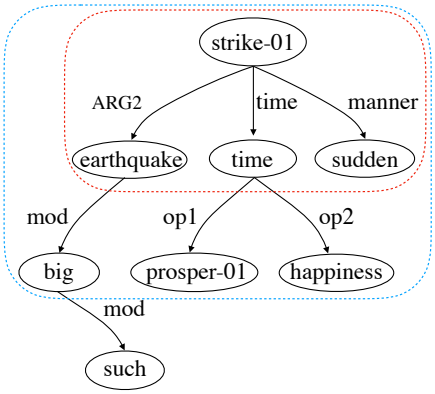


Figure 1: AMR for the sentence "*During a time of prosperity and happiness, such a big earthquake suddenly struck.*", where the subgraphs close to the root represent the core semantics.

Many methods for AMR parsing have been developed in the past years, which can be categorized into three main classes: Graph-based parsing (Flanigan et al., 2014; Lyu and Titov, 2018) uses a pipeline design for concept identification and relation prediction. Transition-based parsing (Wang et al., 2016; Damonte et al., 2017; Ballesteros and Al-Onaizan, 2017; Guo and Lu, 2018; Liu et al., 2018; Wang and Xue, 2017) processes a sentence from left-to-right and constructs the graph incrementally. The third class is seq2seq-based parsing (Barzdins and Gosko, 2016; Konstas et al., 2017; van Noord and Bos, 2017), which views parsing as sequence-to-sequence transduction by a linearization (depth-first traversal) of the AMR graph.

*The work described in this paper is substantially supported by a grant from the Research Grant Council of the Hong Kong Special Administrative Region, China (Project Code: 14204418). The first author is grateful for the discussions with Zhisong Zhang and Zhijiang Guo.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3799–3809,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

While existing graph-based models cannot sufficiently model the interactions between individual decisions, the autoregressive nature of transition-based and seq2seq-based models makes them suffer from error propagation, where later decisions can easily go awry, especially given the complexity of AMR. Since capturing the core semantics of a sentence is arguably more important and useful in practice, it is desirable for a parser to have a global view and a priority for capturing the main ideas first. In fact, AMR graphs are organized in a hierarchy that the core semantics stay closely to the root, for which a top-down parsing scheme can fulfill the desiderata. For example, in Figure 1, the subgraph in the red box already conveys the core meaning "*an earthquake suddenly struck at a particular time*", and the subgraph in the blue box further informs that "*the earthquake was big*" and "*the time was of prosperity and happiness*".

We propose a novel framework for AMR parsing known as Graph Spanning based Parsing (GSP). One novel characteristic of GSP is that, to our knowledge, it is the first top-down AMR parser.[1] GSP performs parsing in an incremental, root-to-leaf fashion, but still maintains a global view of the sentence and the previously derived graph. At each step, it generates the connecting arcs between the existing nodes and the coming new node, upon which the type of the new node (concept) is jointly decided. The output graph spans the nodes by the distance to the root, following the intuition of first grasping the main ideas then digging into more details. Compared to previous graph-based methods, our model is capable of capturing more complicated intra-graph interactions, while reducing the number of parsing steps to be linear in the sentence length.[2] Compared to transition-based methods, our model removes the left-to-right restriction and avoids sophisticated oracle design for handling the complexity of AMR graphs.

Notably, most existing methods including the state-the-of-art parsers often rely on heavy graph re-categorization for reducing the complexity of the original AMR graphs. For graph re-categorization, specific subgraphs of AMR are grouped together and assigned to a single node with a new compound category (Werling et al., 2015; Wang and Xue, 2017; Foland and Martin, 2017; Lyu and Titov, 2018; Groschwitz et al., 2018; Guo and Lu, 2018). The hand-crafted rules for re-categorization are often non-trivial, requiring exhaustive screening and expert-level manual efforts. For instance, in the re-categorization system of Lyu and Titov (2018), the graph fragment "`temporal-quantity` $\xrightarrow{:ARG3-of}$ `rate-entity-91` $\xrightarrow{:unit}$ `year` $\xrightarrow{:quant}$ `1`" will be replaced by one single nested node "`rate-entity-3(annual-01)`". There are hundreds of such manual heuristic rules. This kind of re-categorization has been shown to have considerable effects on the performance (Wang and Xue, 2017; Guo and Lu, 2018). However, one issue is that the precise set of re-categorization rules differs among different models, making it difficult to distinguish the performance improvement from model optimization or carefully designed rules. In fact, some work will become totally infeasible when removing this re-categorization step. For example, the parser of Lyu and Titov (2018) requires tight integration with this step as it is built on the assumption that an injective alignment exists between sentence tokens and graph nodes.

We evaluate our parser on the latest AMR sembank and achieve competitive results to the state-of-the-art models. The result is remarkable since our parser directly operates on the original AMR graphs and requires no manual efforts for graph re-categorization. The contributions of our work are summarized as follows:

- We propose a new method for learning AMR parsing that produces high-quality core semantics.
- Without the help of heuristic graph re-categorization which requires expensive expert-level manual efforts for designing re-categorization rules, our method achieves state-of-the-art performance.

## 2 Related Work

Currently, most AMR parsers can be categorized into three classes: (1) *Graph-based* methods (Flanigan et al., 2014, 2016; Werling et al., 2015; Foland and Martin, 2017; Lyu and Titov, 2018; Zhang et al., 2019) adopt a pipeline approach for graph construction. It first maps continuous text spans into AMR concepts, then calculates the scores of possible edges and uses a max-

[1] Depth-first traversal in seq2seq models does not produce a strictly top-down order due to the reentrancies in AMR.

[2] Since the size of AMR graph is approximately linear in the length of sentence.

imum spanning connected subgraph algorithm to select the final graph. The major deficiency is that the concept identification and relation prediction are strictly performed in order, yet the interactions between them should benefit both sides (Zhou et al., 2016). In addition, for computational efficacy, usually only first-order information is considered for edge scoring. (2) *Transition-based* methods (Wang et al., 2016; Damonte et al., 2017; Wang and Xue, 2017; Ballesteros and Al-Onaizan, 2017; Liu et al., 2018; Peng et al., 2018; Guo and Lu, 2018; Naseem et al., 2019) borrow techniques from shift-reduce dependency parsing. Yet the non-trivial nature of AMR graphs (e.g., reentrancies and non-projective arcs) makes the transition system even more complicated and difficult to train (Guo and Lu, 2018). (3) *Seq2seq-based* methods (Barzdins and Gosko, 2016; Peng et al., 2017; Konstas et al., 2017; van Noord and Bos, 2017) treat AMR parsing as sequence-to-sequence problem by linearizing AMR graphs, thus existing seq2seq models (Bahdanau et al., 2014; Luong et al., 2015) can be readily utilized. Despite its simplicity, the performance of the current seq2seq models lag behind when the training data is limited. The first reason is that seq2seq models are often not as effective on smaller datasets. The second reason is that the linearized AMRs add the challenges of making use of the graph structure information.

There are also some notable exceptions. Peng et al. (2015) introduce a synchronous hyperedge replacement grammar solution. Pust et al. (2015) regard the task as a machine translation problem, while Artzi et al. (2015) adapt combinatory categorical grammar. Groschwitz et al. (2018); Lindemann et al. (2019) view AMR graphs as the structure AM algebra.

Most AMR parsers require an explicit alignment between tokens in the sentences and nodes in the AMR graph during training. Since such information is not annotated, a pre-trained aligner (Flanigan et al., 2014; Pourdamghani et al., 2014; Liu et al., 2018) is often required. More recently, Lyu and Titov (2018) demonstrate that the alignments can be treated as latent variables in a joint probabilistic model.

## 3 Background and Overview

### 3.1 Background of Multi-head Attention

The multi-head attention mechanism introduced by Vaswani et al. (2017) is used as a basic building block in our framework. The multi-head attention consists of $H$ attention heads, and each of which learns a distinct attention function. Given a query vector $x$ and a set of vectors $\{y_1, y_2, \ldots, y_m\}$ or in short $y_{1:m}$, for each attention head, we project $x$ and $y_{1:m}$ into distinct query, key, and value representations $q \in \mathbb{R}^d$, $K \in \mathbb{R}^{m \times d}$ and $V \in \mathbb{R}^{m \times d}$ respectively, where $d$ is the dimension of the vector space. Then we perform scaled dot-product attention (Vaswani et al., 2017):

$$a = \text{softmax}\frac{(Kq)}{\sqrt{d}}$$
$$attn = aV$$

where $a \in \mathbb{R}^m$ is the attention vector (a distribution over all input $y_{1:m}$) and $attn$ is the weighted sum of the value vectors. Finally, the outputs of all attention heads are concatenated and projected to the original dimension of $x$. For brevity, we will denote the whole attention procedure described above as a function $T(x, y_{1:m})$.

Based on the multi-head attention, the Transformer encoder (Vaswani et al., 2017) uses self-attention for context information aggregation when given a set of vectors (e.g., word embeddings in a sentence or node embeddings in a graph).

### 3.2 Overview

Figure 2 depicts the major neural components in our proposed framework: The Sentence Encoder component and the Graph Encoder component are designed for token-level sentence representation and node-level graph representation respectively. Given an input sentence $\mathbf{w} = (w_1, w_2, \ldots, w_n)$, where $n$ is the sentence length, the Sentence Encoder component will first read the whole sentence and encode each word $w_i$ into the hidden state $s_i$. The initial graph $G_0$ is always initialized with one dummy node $d^*$ and a previously generated concept $c_j$ is encoded into the hidden state $v_j$ by the Graph Encoder component.

At each time step $t$, the Focus Selection component reads both the sentence representation $s_{1:n}$ and the graph representation $v_{0:t-1}$ of $G_{t-1}$ repeatedly, generates the initial parser state $h_t$. The

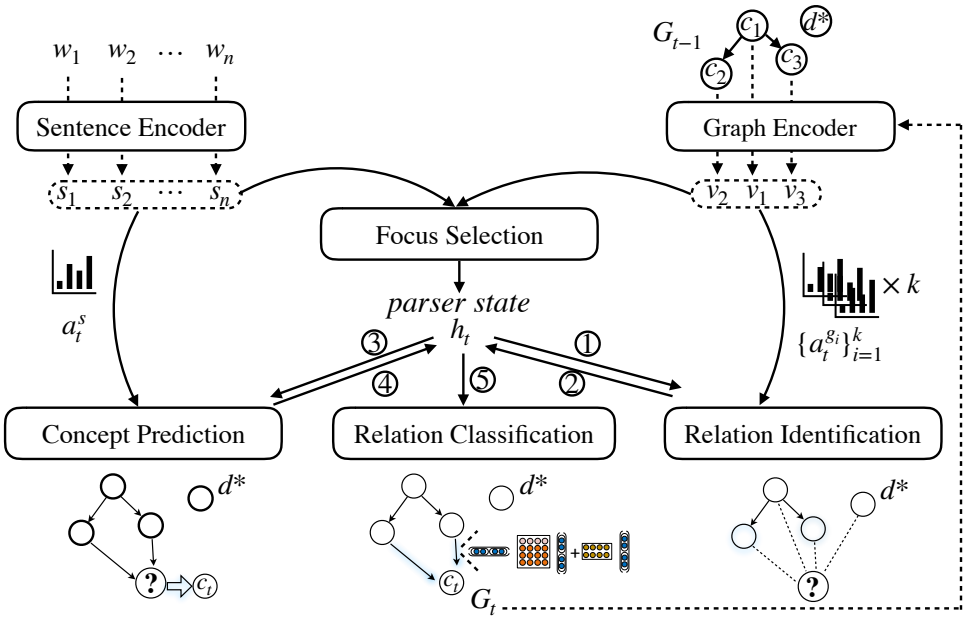


Figure 2: Model architecture of GSP, together with the decoding procedure at the time step $t$, where the read and write operations around the parser state $h_t$ follow the order ① → ② → ③ → ④ → ⑤.

parser state carries the most useful information and serves as a writable memory during the expansion step. Next, the Relation Identification component decides which specific head nodes to expand by computing the multiple attention scores $\{a_t^{g_i}\}_{i=1}^k$ over the existing nodes. New arcs are generated according to the attention scores. Then the Concept Prediction component updates the parser state $h_t$ with arc information, computes the attention vector $a_t^s$ over the sentence and accordingly chooses a specific part to generate the new concept $c_t$. Finally, the Relation Classification component is used to predict the relation labels between the newly generated concept and its predecessors. Consequently an updated graph $G_t$ is produced and $G_t$ will be processed for the next time step. The whole decoding procedure is terminated if the newly generated concept is the special stop concept $\oslash$.

Our method expands the graph in a root-to-leaf fashion, nodes with shorter distances to the root will be introduced first. It follows a similar way that humans grasp the meaning: first seeking the main concepts then proceeding to the substructures governed by certain head concepts (Banarescu et al., 2013).

During training, we use breadth-first search to decide the order of nodes. However, for nodes with multiple children, there still exist multiple valid selections. In order to define a deterministic decoding process, we sort sibling nodes by their relations to the head node. We will present more discussions on the choice of sibling order in § 5.3.

## 4 Framework Description

### 4.1 Sentence & Graph Representation

Transformer encoder architecture is employed for both the Sentence Encoder and the Graph encoder components. For sentence encoding, a special token ($\triangleright$) is prepended to the input word sequence, whose final hidden state $s_0$ is regarded as an aggregated summary of the whole sentence and used as the initial state in parsing steps.

The Graph Encoder component takes previously generated concept sequence $(c_0, c_1, \ldots, c_{t-1})$ ($c_0$ is the dummy node $d^*$) as input. For computation efficiency and reducing error propagation, instead of encoding the edge information explicitly, we use the Transformer encoder to capture the interactions between nodes. Finally, the encoder outputs a sequence of node representations $(v_0, v_1, \ldots, v_{t-1})$.

### 4.2 Focus Selection

At each time step $t$, the Focus Selection component will read the sentence and the partially constructed graph repeatedly for gradually locating and collecting the most relevant information for the next expansion. We simulate the repeated reading by multiple levels of attention. Formally,

the following recurrence is applied by $L$ times:

$$x_{t,1}^{l+1} = \text{LN}(h_t^l + T_1^{l+1}(h_t^l, s_{1:n}))$$
$$x_{t,2}^{l+1} = \text{LN}(x_{t,1}^{l+1} + T_2^{l+1}(x_{t,1}^{l+1}, v_{0:t-1}))$$
$$h_t^{l+1} = \max(x_{t,2}^{l+1} W_1^{l+1} + b_1^{l+1}) W_2^{l+1} + b_2^{l+1}$$

where $T(\cdot,\cdot)$ is the multi-head attention function. LN is the layer normalization (Ba et al., 2016) and $h_t^0$ is always initialized with $s_0$. For clarity, we denote the last hidden state $h_t^L$ as $h_t$, as the parser state at the time step $t$. We now proceed to present the details of each decision stage of one parsing step, which is also illustrated in Figure 2.

### 4.3 Relation Identification

Our Relation Identification component is inspired by a recent attempt of exposing auxiliary supervision on attention mechanism (Strubell et al., 2018). It can be considered as another attention layer over the existing graph, yet the attention weights explicitly indicate the likelihood of the new node being attached to a specific node. In other words, its aim is to answer the question of where to expand. Since a node can be attached to multiple nodes by playing different semantic roles, we utilize multi-head attention and take the maximum over different heads as the final arc probabilities.

Formally, through a multi-head attention mechanism taking $h_t$ and $v_{0:t-1}$ as input, we obtain a set of attention weights $\{a_t^{g_i}\}_{i=1}^k$, where $k$ is the number of attention heads and $a_t^{g_i}$ is the $i$-th probability vector. The probability of the arc between the new node and the node $v_j$ is then computed by $a_{t,j}^g = max_i(a_{t,j}^{g_i})$. Intuitively, each head is in charge of a set of possible relations (though not explicitly specified). If certain relations do not exist between the new node and any existing node, the probability mass will be assigned to the dummy node $d^*$. The maximum pooling reflects that the arc should be built once one relation is activated.[3]

The attention mechanism passes the arc decisions to later layers by the update of the parser state as follows:

$$h_t = LN(h_t + W^{\text{arc}} \sum_{j=0}^{t-1} a_{t,j}^g v_j)$$

[3]We also found that there may exist more than one relation between two distinct nodes, however, it rarely happens.

### 4.4 Concept Prediction

Our Concept Prediction component uses a soft alignment between words and the new concept. Concretely, a single-head attention $a_t^s$ is computed based on the parser state $h_t$ and the sentence representation $s_{1:n}$, where $a_{t,i}^s$ denotes the attention weight of the word $w_i$ in the current time step. This component then updates the parser state with the alignment information via the following equation:

$$h_t = LN(h_t + W^{\text{conc}} \sum_{i=1}^{n} a_{t,i}^s s_i)$$

The probability of generating a specific concept $c$ from the concept vocabulary $\mathcal{V}$ is calculated as $gen(c|h_t) = \exp({x_c}^T h_t)/\sum_{c' \in \mathcal{V}} \exp({x_{c'}}^T h_t)$, where $x_c$ (for $c \in \mathcal{V}$) denotes the model parameters. To address the data sparsity issue in concept prediction, we introduce a copy mechanism in similar spirit to Gu et al. (2016). Besides generation, our model can either directly copy an input token $w_i$ (e.g, for entity names) or map $w_i$ to one concept $m(w_i)$ according to the alignment statistics[4] in the training data (e.g., for "went", it would propose *go*). Formally, the prediction probability of a concept $c$ is given by:

$$\begin{aligned} P(c|h_t) =& P(copy|h_t) \sum_{i=1}^{n} a_{t,i}^s [[w_i = c]] \\ &+ P(map|h_t) \sum_{i=1}^{n} a_{t,i}^s [[m(w_i) = c]] \\ &+ P(gen|h_t) gen(c|h_t) \end{aligned}$$

where $[[\ldots]]$ is the indicator function. $P(copy|h_t)$, $P(map|h_t)$ and $P(gen|h_t)$ are the probabilities of three prediction modes respectively, computed by a single layer neural network with softmax activation.

### 4.5 Relation Classification

Lastly, the Relation Classification component employs a multi-class classifier for labeling the arcs detected in the Relation Identification component. The classifier uses a biaffine function to score each label, given the head concept representation $v_i$ and the child vector $h_t$ as input:

$$e_t^i = h_t^T W v_i + U^T h_t + V^T v_i + b$$

[4]Based on the alignments provided by Liu et al. (2018), for each word, the most frequently aligned concept (or its lemma if it has empty alignment) is used for direct mapping.

where $W, U, V, b$ are model parameters. As suggested by Dozat and Manning (2016), we project $v_i$ and $h_t$ to a lower dimension for reducing the computation cost and avoiding the overfitting of the model. The label probabilities are computed by a softmax function over all label scores.

### 4.6 Reentrancies

AMR reentrancy is employed when a node participates in multiple semantic relations (with multiple parent nodes), and that is why AMRs are graphs, rather than trees. The reentrancies are often hard to treat. While previous work often either remove them (Guo and Lu, 2018) or relies on rule-based restoration in the postprocessing stage (Lyu and Titov, 2018; van Noord and Bos, 2017), our model provides a new and principled way to deal with reentrancies. In our approach, when a new node is generated, all its connections to already existing nodes are determined by the multi-head attention. For example, for a node with $k$ parent nodes, $k$ different heads will point to the those parent nodes respectively. For a better understanding of our model, a pseudocode is presented in Algorithm 1.

### 4.7 Training and Inference

Our model is trained to maximize the log likelihood of the gold AMR graphs given sentences, i.e. $\log P(G|\mathbf{w})$, which can be factorized as:

$$\log P(G|\mathbf{w}) = \sum_{t=1}^{m} \Bigg( \log P(c_t|G_{t-1}, \mathbf{w}) + \sum_{i \in pred(t)} \log P(arc_{it}|G_{t-1}, \mathbf{w}) + \sum_{i \in pred(t)} \log P(rel_{arc_{it}}|G_{t-1}, \mathbf{w}) \Bigg)$$

where $m$ is the total number of vertices. The set of predecessor nodes of $c_t$ is denoted as $pred(t)$. $arc_{it}$ denotes the arc between $c_i$ and $c_t$, and $rel_{arc_{it}}$ indicates the arc label (relation type).

As mentioned, GSP is an autoregressive model, such as seq2seq models and transition models, but it factors the distribution according to a top-down graph structure rather than a depth-first traversal or a left-to-right chain. Meanwhile, GSP has a clear separation of node, arc and relation label probabilities, interacting in a more interpretable and tighten manner.

**Algorithm 1** Graph Spanning based Parsing

**Input:** the input sentence $\mathbf{w} = (w_1, w_2, \ldots, w_n)$
**Output:** the AMR graph $G$ corresponds to $\mathbf{w}$.

◇ Learning Sentence Representation
$\mathbf{w} = (w_0 = \triangleright) + (w_1, w_2, \ldots, w_n)$
$s_0, s_1, s_2, \ldots, s_n =$ Transformer($\mathbf{w}$)
◇ Initialization
initialize the graph $G_0$ ($c_0 = d^*$)
initialize time step $t = 1$
◇ Entering Main Spanning Loop
**while** True **do**
  $h_0, \ldots, v_{t-1}$ = Transformer($c_{0:t-1}$)
  $h_t$ = Focus_Selection $(s_0, v_{0:t-1}, s_{1:n})$
  $h_t$ = Relation_Identification $(h_t, v_{0:t-1})$
  *decide the parents nodes* $pred(t)$ *of* $c_t$
  $h_t$ = Concept_Prediction $(h_t, s_{1:n})$
  *decide the node type of* $c_t$
  **if** $c_t == \oslash$ **then**
    **break**
  **end if**
  **for** $i \in pred(t)$ **do**
    Relation_Classification $(h_t, v_i)$
    *decide the edge type between* $c_t$ *and* $c_i$
  **end for**
  update $G_{t-1}$ to $G_t$
**end while**
**return** $G_{t-1}$

At the operational or testing time, the prediction for the input $\mathbf{w}$ is obtained via $\hat{G} = \arg\max_{G'} P(G'|\mathbf{w})$. Rather than iterating over all possible graphs, we adopt a beam search to approximate the best graph. Specifically, for each partially constructed graph, we only consider the top-$K$ concepts obtaining the best single-step probability (a product of the corresponding concept, arc, and relation label probability), where $K$ is the beam size. Only the best $K$ graphs at each time step are kept for the next expansion.

## 5 Experiments

### 5.1 Setup

We focus on the most recent LDC2017T10 dataset, as it is the largest AMR corpus. It consists of 36521, 1368, and 1371 sentences in the training, development, and testing sets respectively.

We use Stanford CoreNLP (Manning et al., 2014) for text preprocessing, including tokenization, lemmatization, part-of-speech, and named-entity tagging. The input for sentence encoder

consists of the randomly initialized lemma, part-of-speech tag, and named-entity tag embeddings, as well as the output from a learnable CNN with character embeddings as inputs. The graph encoder uses randomly initialized concept embeddings and another char-level CNN. Model hyper-parameters are chosen by experiments on the development set. The details of the hyper-parameter settings are provided in the Appendix. During testing, we use a beam size of 8 for generating graphs.[5]

Conventionally, the quality of AMR parsing results is evaluated using the Smatch tool (Cai and Knight, 2013), which seeks for the maximum number of overlaps between two AMR annotations after decomposing AMR graphs into triples. However, the ordinary Smatch metric treats all triples equally regardless of their roles in the composition of the whole sentence meaning. We refine the ordinary Smatch metric to take into consideration the notion of core semantics. Specifically, we compute:

- Smatch-weighted: This metric weights different triples by their importance of composing the core ideas. The root distance $d$ of a triple is defined as the minimum root distance of its involving nodes, the weight of the triple is then computed as:

$$w = \min(-d + d_{thr}, 1)$$

  In other words, the weight has a linear decay in root distance until $d_{thr}$. If two triples are matched, the minimum importance score of them is obtained. In our experiments, $d_{thr}$ is set to 5.

- Smatch-core: This metric only compares the subgraphs representing the main meaning. Precisely, we cut down AMR graphs by setting a maximum root distance $d_{max}$ and only keep the nodes and edges within the threshold. $d_{max}$ is set to 4 in our experiments, of which the remaining subgraphs still have a broad coverage of the original meaning, as illustrated by the distribution of root distance in Figure 3.

Besides, we also evaluate the quality by computing the following metrics.

[5]Our code can be found at https://github.com/jcyk/AMR-parser.

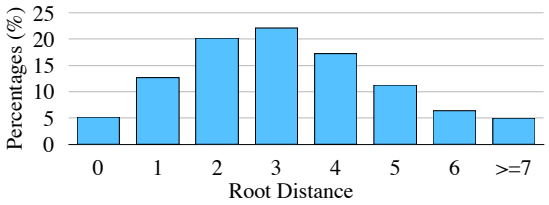


Figure 3: The distribution of root distance of concepts in the test set.

- complete-match (CM): This metric counts the number of parsing results that are completely correct.

- root-accuracy (RA): This metric measures the accuracy of the root concept identification.

### 5.2 Main Results and Case Study

The main result is presented in Table 1. We compare our method with the best-performing models in each category as discussed in § 2.

Concretely, van Noord and Bos (2017) is a character-level seq2seq model that achieves very competitive result. However, their model is very data demanding as it requires to train on additional 100K sentence-AMR pairs generated by other parsers. Guo and Lu (2018) is a transition-based parser with refined search space for AMR. Certain concepts and relations (e.g., reentrancies) are removed to reduce the burdens during training. Lyu and Titov (2018) is a graph-based method that achieves the best-reported result evaluated by the ordinary Smatch metric. Their parser uses different LSTMs for concept prediction, relation identification, and root identification sequentially. Also, the relation identification stage has the time complexity of $O(m^2 \log m)$ where $m$ is the number of concepts. Groschwitz et al. (2018) views AMR as terms of the AM algebra (Groschwitz et al., 2017), which allows standard tree-based parsing techniques to be applicable. The complexity of their projective decoder is $O(m^5)$. Last but not least, all these models except for that of van Noord and Bos (2017) require hand-crafted heuristics for graph re-categorization.

We consider the Smatch-weighted metric as the most suitable metric for measuring the parser's quality on capturing core semantics. The comparison shows that our method significantly outperforms all other methods. The Smatch-core metric also demonstrates the advantage of our

| Model | Graph Re-ca. | Smatch(%) | | | RA(%) | CM(%) |
|---|---|---|---|---|---|---|
| | | weighted | core | ordinary | | |
| Buys and Blunsom (2017) | No | - | - | 61.9 | - | - |
| van Noord and Bos (2017) + 100K | No | 68.8 | 67.6 | 71.0 | 75.8 | 10.2 |
| Guo and Lu (2018) | Yes | 63.5 | 62.3 | 69.8 | 63.6 | 9.4 |
| Lyu and Titov (2018) | Yes | 66.6 | 67.1 | **74.4** | 59.1 | 10.2 |
| Groschwitz et al. (2018) | Yes | - | - | 71.0 | - | - |
| Ours | No | **71.3** | **70.2** | 73.2 | **76.9** | **11.6** |

Table 1: Comparison with state-of-the-art methods (results on the test set). Results relying on heuristic rules for graph re-categorization are marked "Yes" in the Graph Re-ca. column.

**Input:** *The solution is functional human patterns and a balance between transport system capacity and land use generated travel demand.*

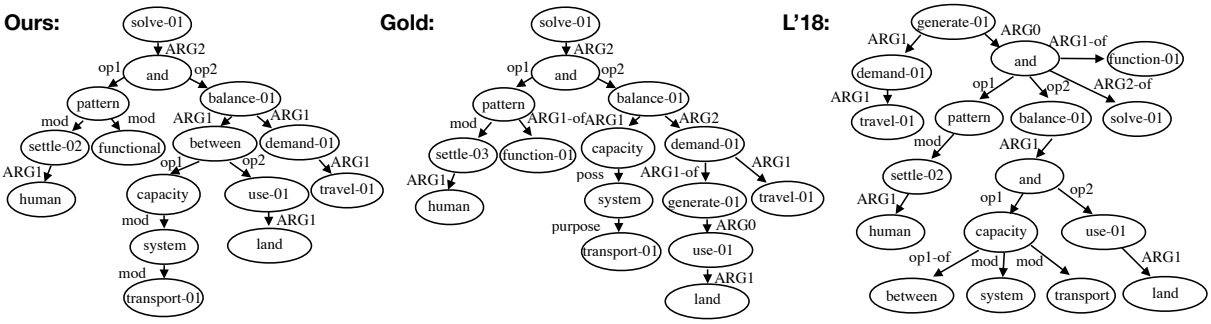


Figure 4: Case study.

method in capturing the core ideas. Besides, our model achieves the highest root-accuracy (RA) and complete-match (CM), which further confirms the usefulness of a global view and the core semantic first principle.

Even evaluated by the ordinary Smatch metric, our model yields better results than all previously reported models with the exception of Lyu and Titov (2018), which relies on a tremendous amount of manual heuristics for designing rules for graph re-categorization and adopts a pipeline approach. Note that our parser constructs the AMR graph in an end-to-end fashion with a better (quadratic) time complexity.

We present a case study in Figure 4 with comparison to the output of Lyu and Titov (2018)'s parser. As seen, both parsers make some mistakes. Specifically, our method fails to identify the concept `generated-01`. While Lyu and Titov (2018)'s parser successfully identifies it, their parser mistakenly treats it as the root of the whole AMR. It leads to a serious drawback of making the sentence meaning be interpreted in a wrong way. In contrast, our method shows a strong capacity in capturing the main idea "*the solution is about some patterns and a balance*". However, on the ordinary Smatch metric, their graph obtains a higher score (68% vs. 66%), which indicates that the ordinary Smatch is not a proper metric for evaluating the quality of capturing core semantics. If we adopt the Smatch-weighted metric, our method achieves a better score i.e. 74% vs. 61%.

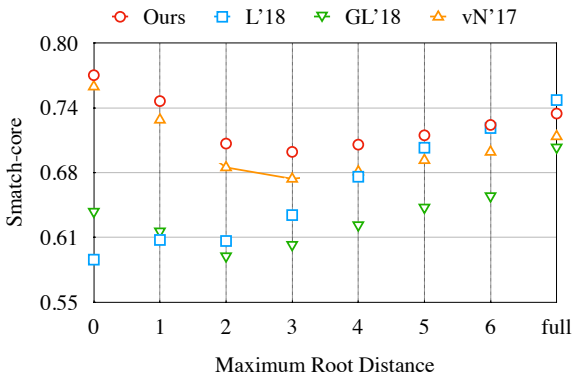


Figure 5: Smatch scores with different root distances. vN'17 is van Noord and Bos (2017)'s parser with 100K additional training pairs. GL'18 is Guo and Lu (2018)'s parser. L'18 is Lyu and Titov (2018)'s parser.

### 5.3 More Results

To reveal our parser's ability for grasping meanings at different levels of granularity, we plot the Smatch-core scores in Figure 5 by varying

the maximum root distance $d_{max}$, compared with several strong baselines and the state-of-the-art model. It demonstrates that our method is better at abstracting the core ideas of a sentence.

As discussed in § 3, there could be multiple valid generation orders for sibling nodes in an AMR graph. We experiment with the following traversal variants: (1) *random*, which sorts the sibling nodes in completely random order. (2) *relation freq.*, which sorts the sibling nodes according to their relations to the head node. We assign higher priorities to relations that occur more frequently, which drives our parser always to seek for the most common relation first. (3) *combined*, which combines the above two strategies by using *random* and *relation freq.* with equal chance. As seen in Table 2, the deterministic order strategy for training (*relation freq.*) achieves better performance than random order. Interestingly, the combined strategy significantly boosts the performance.[6] The reason is that the random order potentially produces a larger set of training pairs since each random order strategy can be considered as a different training pair. On the other hand, the deterministic order stabilizes the maximum likelihood estimate training. Therefore, the combined strategy benefits from both worlds.

| Order | Smatch | | |
|---|---|---|---|
| | weighted | core | ordinary |
| random | 68.2 | 67.4 | 70.4 |
| relation freq. | 69.9 | 68.3 | 70.9 |
| combined | 71.3 | 70.2 | 73.2 |

Table 2: The effect of different sibling orders.

## 6 Conclusion and Future Work

We presented the first top-down AMR parser. Our proposed parser builds a AMR graph incrementally in a root-to-leaf manner. Experiments show that our method has a better capability of capturing the core semantics in a sentence compared with previous state-of-the-art methods. In addition, we overcome the need of heuristics for graph recategorization employed in most previous work, which makes our method much more transferable to other semantic representations or languages.

Our methods follows the intuition that humans tend to grasp the core meaning of a sentence first. However, some cognitive theories (Langacker, 2008) also suggest that human language understanding is often presented as a circular, abductive process (hermeneutic circle). It is interesting to explore the use of some revision mechanisms when the initial steps go wrong.

[6] We note there are many other ways to generate a deterministic order. For example, van Noord and Bos (2017) uses the order of aligned words in the sentence. However, we use the relation frequency method for its simplicity and not relying on external resources (e.g, an aligner).

## References


Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage ccg semantic parsing with amr. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1699–1710.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Miguel Ballesteros and Yaser Al-Onaizan. 2017. AMR parsing using stack-LSTMs. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1269–1275.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186.

Guntis Barzdins and Didzis Gosko. 2016. RIGA at SemEval-2016 task 8: Impact of Smatch extensions and character-level neural translation on AMR parsing accuracy. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1143–1147.

Jan Buys and Phil Blunsom. 2017. Oxford at semeval-2017 task 9: Neural amr parsing with pointer-augmented attention. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 914–919.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 748–752.

Marco Damonte, Shay B. Cohen, and Giorgio Satta. 2017. An incremental parser for abstract meaning representation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 536–546.

Timothy Dozat and Christopher D Manning. 2016. Deep biaffine attention for neural dependency parsing. *arXiv preprint arXiv:1611.01734*.

Jeffrey Flanigan, Chris Dyer, Noah A Smith, and Jaime Carbonell. 2016. Cmu at semeval-2016 task 8: Graph-based amr parsing with infinite ramp loss. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1202–1206.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1426–1436.

William Foland and James H Martin. 2017. Abstract meaning representation parsing using lstm recurrent neural networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 463–472.

Jonas Groschwitz, Meaghan Fowlie, Mark Johnson, and Alexander Koller. 2017. A constrained graph algebra for semantic parsing with AMRs. In *IWCS 2017 - 12th International Conference on Computational Semantics - Long papers*.

Jonas Groschwitz, Matthias Lindemann, Meaghan Fowlie, Mark Johnson, and Alexander Koller. 2018. AMR dependency parsing with a typed semantic algebra. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1831–1841.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640.

Zhijiang Guo and Wei Lu. 2018. Better transition-based amr parsing with refined search space. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1712–1722.

Hardy Hardy and Andreas Vlachos. 2018. Guided neural language generation for abstractive summarization using abstract meaning representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 768–773.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: Sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 146–157.

Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. Dependency parsing. *Synthesis Lectures on Human Language Technologies*, 1(1):1–127.

Ronald W Langacker. 2008. *Cognitive Grammar: A Basic Introduction*. Oxford University Press.

Matthias Lindemann, Jonas Groschwitz, and Alexander Koller. 2019. Compositional semantic parsing across graphbanks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4576–4585.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A. Smith. 2015. Toward abstractive summarization using semantic representations. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1077–1086.

Yijia Liu, Wanxiang Che, Bo Zheng, Bing Qin, and Ting Liu. 2018. An AMR aligner tuned by transition-based parser. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2422–2430.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Chunchuan Lyu and Ivan Titov. 2018. AMR parsing as graph prediction with latent alignment. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 397–407.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

Arindam Mitra and Chitta Baral. 2016. Addressing a question answering challenge by combining statistical methods with inductive rule learning and reasoning. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Tahira Naseem, Abhishek Shah, Hui Wan, Radu Florian, Salim Roukos, and Miguel Ballesteros. 2019. Rewarding Smatch: Transition-based AMR parsing with reinforcement learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4586–4592.

Rik van Noord and Johan Bos. 2017. Neural semantic parsing by character-based translation: Experiments with abstract meaning representations. *arXiv preprint arXiv:1705.09980*.

Xiaochang Peng, Daniel Gildea, and Giorgio Satta. 2018. Amr parsing with cache transition systems. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Xiaochang Peng, Linfeng Song, and Daniel Gildea. 2015. A synchronous hyperedge replacement grammar based approach for amr parsing. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 32–41.

Xiaochang Peng, Chuan Wang, Daniel Gildea, and Nianwen Xue. 2017. Addressing the data sparsity issue in neural AMR parsing. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 366–375.

Nima Pourdamghani, Yang Gao, Ulf Hermjakob, and Kevin Knight. 2014. Aligning english strings with abstract meaning representation graphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 425–429.

Michael Pust, Ulf Hermjakob, Kevin Knight, Daniel Marcu, and Jonathan May. 2015. Parsing english into abstract meaning representation using syntax-based machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1143–1154.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5027–5038.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Chuan Wang, Sameer Pradhan, Xiaoman Pan, Heng Ji, and Nianwen Xue. 2016. Camr at semeval-2016 task 8: An extended transition-based amr parser. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1173–1178.

Chuan Wang and Nianwen Xue. 2017. Getting the most out of amr parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1257–1268.

Keenon Werling, Gabor Angeli, and Christopher D. Manning. 2015. Robust subgraph generation improves abstract meaning representation parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 982–991.

Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019. AMR parsing as sequence-to-graph transduction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 80–94.

Yue Zhang and Stephen Clark. 2009. Transition-based parsing of the chinese treebank using a global discriminative model. In *Proceedings of the 11th International Conference on Parsing Technologies*, pages 162–171.

Junsheng Zhou, Feiyu Xu, Hans Uszkoreit, Weiguang Qu, Ran Li, and Yanhui Gu. 2016. AMR parsing with an incremental joint model. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 680–689.

# *Don't paraphrase, detect!* Rapid and Effective Data Collection for Semantic Parsing

**Jonathan Herzig**[1] **Jonathan Berant**[1,2]

[1]School of Computer Science, Tel-Aviv University
[2]Allen Institute for Artificial Intelligence
{jonathan.herzig,joberant}@cs.tau.ac.il

## Abstract

A major hurdle on the road to conversational interfaces is the difficulty in collecting data that maps language utterances to logical forms. One prominent approach for data collection has been to automatically generate pseudo-language paired with logical forms, and paraphrase the pseudo-language to natural language through crowdsourcing (Wang et al., 2015). However, this data collection procedure often leads to low performance on real data, due to a mismatch between the true distribution of examples and the distribution induced by the data collection procedure. In this paper, we thoroughly analyze two sources of mismatch in this process: the mismatch in *logical form distribution* and the mismatch in *language distribution* between the true and induced distributions. We quantify the effects of these mismatches, and propose a new data collection approach that mitigates them. Assuming access to unlabeled utterances from the true distribution, we combine crowdsourcing with a paraphrase model to detect correct logical forms for the unlabeled utterances. On two datasets, our method leads to 70.6 accuracy on average on the *true distribution*, compared to 51.3 in paraphrasing-based data collection.

## 1 Introduction

Conversing with a virtual assistant in natural language is one of the most exciting current applications of semantic parsing, the task of mapping natural language utterances to executable logical forms (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Liang et al., 2011). Semantic parsing models rely on supervised training data that pairs natural language utterances with logical forms. Alas, such data does not occur naturally, especially in virtual assistants that are meant to support thousands of different applications and use-cases. Thus, efficient data collection is perhaps the most pressing problem for scalable conversational interfaces.

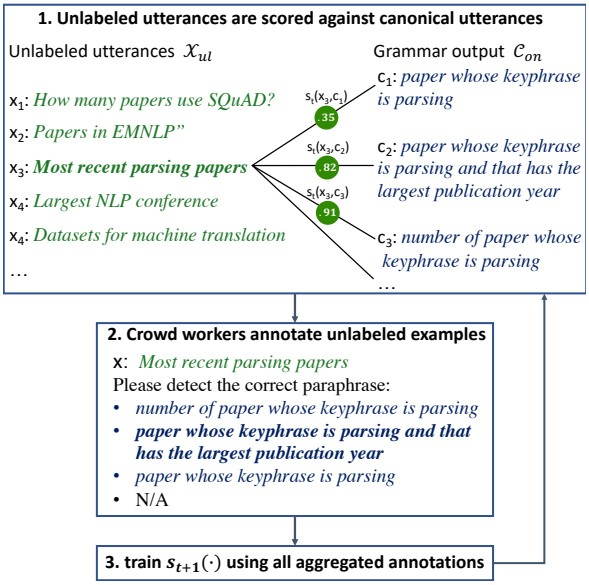


Figure 1: An overview of GRANNO, a method for annotating unlabeled utterances with their logical forms.

In recent years, many attempts aim to reduce the burden of data collection for semantic parsing, including training from denotations (Kwiatkowski et al., 2013; Artzi and Zettlemoyer, 2013), semi-supervised learning (Kociský et al., 2016; Yin et al., 2018), human in the loop (Iyer et al., 2017; Lawrence and Riezler, 2018), and training on examples from other domains (Herzig and Berant, 2017, 2018; Su and Yan, 2017). One prominent approach for data collection was introduced by Wang et al. (2015), termed OVERNIGHT. In this method, one automatically generates logical forms for an application from a grammar, paired with pseudo-language utterances. These pseudo-language utterances are then shown to crowd workers, who are able to understand them and paraphrase them into natural language, resulting



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3810–3820, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

in a supervised dataset.

The OVERNIGHT procedure has been adopted and extended both within semantic parsing (Locascio et al., 2016; Ravichander et al., 2017; Zhong et al., 2017; Cheng et al., 2018), in dialogue (Shah et al., 2018; Damonte et al., 2019) and in visual reasoning (Johnson et al., 2017; Hudson and Manning, 2019).

While the OVERNIGHT approach is appealing since it generates training data from scratch, it suffers from a major drawback – training a parser on data generated from OVERNIGHT and testing on utterances collected from a target distribution results in a significant drop in performance (Wang et al., 2015; Ravichander et al., 2017).

In this paper, we thoroughly analyze the sources of mismatch between a target distribution and the distribution induced by OVERNIGHT, and propose a new method for overcoming this mismatch. We decouple the mismatch into two terms: the *logical form mismatch*, i.e., the difference between the target distribution of logical forms and the distribution obtained when generating from a grammar, and the *language mismatch*, i.e., the difference between the natural language obtained when paraphrasing a pseudo-language utterance and the language obtained by real users of an application.

We find that the most severe problem arising from the *logical form mismatch* is insufficient coverage of logical form templates that occur in the true distribution, when generating from a grammar. We also isolate the *language mismatch* effect by paraphrasing logical forms sampled from the true logical form distribution, and find that the language mismatch alone results in a decrease of 9 accuracy points on average on two datasets.

To overcome these mismatches, we propose an alternative method to OVERNIGHT, that utilizes unlabeled utterances. Our method, named GRANNO (grammar-driven annotation), allows crowd workers to iteratively annotate unlabeled utterances by *detecting* their correct grammar generated paraphrase. Figure 1 illustrates a single iteration of this approach. An unlabeled utterance is matched using a paraphrase model against candidate pseudo-language utterances generated by the grammar (step 1). The unlabeled utterance and its top candidate paraphrases are presented to a crowd worker that detects the correct paraphrase (step 2). The paraphrase model is re-trained given all annotated utterances thus far (step 3), and is used to score the remaining unlabeled utterances.

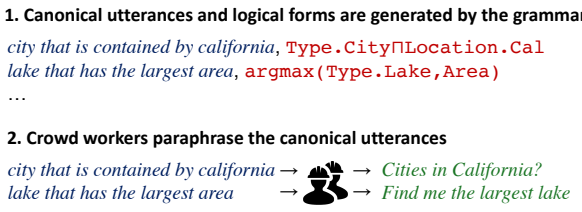


Figure 2: OVERNIGHT: Canonical utterances are generated by a grammar and paraphrased by crowd workers.

On two semantic parsing datasets, we show our procedure leads to annotation of 89% of the original training data. The accuracy of the resulting parser is 70.6 on average, well beyond the accuracy obtained through the original OVERNIGHT procedure at 51.3. This substantially closes the gap to a fully-supervised semantic parser, which obtains 84.9 accuracy. All our code and collected data is available at https://github.com/jonathanherzig/semantic-parsing-annotation.

## 2 The OVERNIGHT Framework

We now describe the OVERNIGHT framework for data collection, which we investigate and improve upon in this work. The starting point is a user who needs a semantic parser for some domain, but has *no data*. OVERNIGHT describes a procedure for generating training data from scratch, which comprises two steps (see Figure 2). First, a synchronous grammar is used to generate logical forms paired with *canonical utterances*, which are pseudo-language utterances that are understandable to people, but do not sound natural. Second, crowd workers paraphrase these canonical utterances into natural language utterances. This results in a training set of utterances paired with logical forms that is used to train the semantic parser. We now briefly elaborate on these two steps.

**Grammar** The grammar in OVERNIGHT generates logical forms paired with canonical utterances, which are understandable to crowd workers (e.g., *"number of state that borders california"*). The grammar has two parts: the *domain-*

*general* part contains domain-independent rules that cover logical operators (e.g., comparatives, superlatives, negation etc.). In addition, a *seed lexicon* specifies a canonical phrase (*"publication year"*) for each knowledge-base (KB) constant (`PublicationYear`), in each particular domain.

While there are many possible ways to sample data from the grammar, in OVERNIGHT logical forms and canonical utterances are exhaustively generated up to a certain maximal depth, hereafter denoted $D$. This follows the assumption that the semantic parser, trained from this data, should generalize to logical forms that correspond to deeper trees (we re-examine this assumption in Section 3). In addition, a typing system is used during generation, thus, semantically vacuous logical forms are not generated (e.g., `PublicationYear.Parsing`), which substantially reduces the number of generated examples.

**Crowdsourcing** After the above generation procedure terminates, crowd workers paraphrase each canonical utterance into a natural language utterance (e.g., the canonical utterance *"paper that has the largest publication year"* can be paraphrased to *"what is the most recent paper?"*). Finally, the framework yields a training dataset $\mathcal{D}_{\text{on}} = \{(x_i, z_i)\}_{i=1}^{N_{\text{on}}}$ consisting of pairs of natural language utterances and logical forms, which can be used to train a semantic parser.

## 3 Mismatch Analysis

In supervised semantic parsing, we assume access to training data of the form $\mathcal{D}_{\text{nat}} = \{(x_i, z_i)\}_{i=1}^{N_{\text{nat}}}$ sampled from the true target distribution $p_{\text{nat}}(x, z)$. Conversely, in the OVERNIGHT framework, we train from $\mathcal{D}_{\text{on}}$, which is sampled from a different distribution $p_{\text{on}}(x, z)$. Training on data sampled from $p_{\text{on}}(x, z)$ and testing on data sampled from $p_{\text{nat}}(x, z)$ leads to a substantial decrease in performance (compared to training on data sampled from $p_{\text{nat}}(x, z)$). In this section, we will analyze and quantify the causes for this degradation in performance.

By writing $p_{\text{nat}}(x, z) = p_{\text{nat}}(z)p_{\text{nat}}(x \mid z)$ and $p_{\text{on}}(x, z) = p_{\text{on}}(z)p_{\text{on}}(x \mid z)$ (mirroring the data generation procedure of OVERNIGHT), we can decouple the distribution mismatch into two terms: The first is the *logical form mismatch*, i.e., the difference between the natural distribution of logical forms $p_{\text{nat}}(z)$ and the distribution $p_{\text{on}}(z)$ of logical forms induced by OVERNIGHT. The second is the *language mismatch*, i.e., the difference between the conditional distribution of natural language $p_{\text{nat}}(x \mid z)$ and the conditional distribution $p_{\text{on}}(x \mid z)$ of natural language when performing crowdsourcing with the OVERNIGHT procedure. We will now investigate these two types of mismatch and their interaction with neural semantic parsers in the context of two popular semantic parsing datasets, GEOQUERY (Zelle and Mooney, 1996) and SCHOLAR (Iyer et al., 2017), which focus on the domains of geography and publications, respectively. For this analysis, we replicated the OVERNIGHT procedure and generated a dataset $\mathcal{D}_{\text{on}}$ for these two domains (details in Section 5).

### 3.1 Logical Form Mismatch

The first source of mismatch is that the natural distribution of logical forms $p_{\text{nat}}(z)$ can be quite different from the distribution $p_{\text{on}}(z)$ induced by the ad-hoc procedure for generating logical forms in OVERNIGHT.

**Logical Operators Frequency** A simple way to quantify the logical form mismatch, is to look whether the proportion of different logical operators, such as superlatives (e.g., `argmax`) and comparatives (e.g., `>`), substantially differs between $\mathcal{D}_{\text{on}}$ and $\mathcal{D}_{\text{nat}}$. Table 1 examines this, where hereafter we use the original training and development sets of GEOQUERY and SCHOLAR as $\mathcal{D}_{\text{nat}}$. The table shows that the frequency of logical operators is different between the two training sets $\mathcal{D}_{\text{nat}}$ and $\mathcal{D}_{\text{on}}$. For example, in $\mathcal{D}_{\text{nat}}$ of GEOQUERY, only $1\%$ of the examples involve negation, compared to $55\%$ of the examples in $\mathcal{D}_{\text{on}}$. Such differences may present generalization difficulties for a model trained from $\mathcal{D}_{\text{on}}$, as the real distribution we care about is $\mathcal{D}_{\text{nat}}$.

**Coverage** Wang et al. (2015) proposed to exhaustively generate all logical forms corresponding to a derivation tree of maximal depth $D$, assuming that the model will generalize to logical forms that require deeper trees. However, recent work from Finegan-Dollak et al. (2018) showed this might not be the case. In their work, Finegan-Dollak et al. (2018) consider logical form *templates*, i.e., logical forms where the KB constants are abstracted to their semantic type, and split the data such that templates in the test set are disjoint from templates in the training set. They show that neural semantic parsers struggle to generalize in this setup. Thus, *covering* the logical form

| operator | GEOQUERY | | SCHOLAR | | example |
|---|---|---|---|---|---|
| | $\mathcal{D}_{\text{nat}}$ | $\mathcal{D}_{\text{on}}$ | $\mathcal{D}_{\text{nat}}$ | $\mathcal{D}_{\text{on}}$ | |
| argmax | .27 | .14 | .18 | .08 | *"the longest river"* |
| argmin | .10 | .14 | .02 | .08 | *"earliest paper by..."* |
| larger | .01 | .06 | .02 | .10 | *"high point higher than that of..."* |
| smaller | .00 | .06 | .02 | .10 | *"paper with less than 10 citations"* |
| $\text{conj}_1$ | .81 | .20 | .50 | .22 | *"parsing papers"* |
| $\text{conj}_2$ | .19 | .80 | .42 | .78 | *"2019 parsing papers"* |
| $\text{conj}_3$ | .00 | .00 | .06 | .00 | *"2019 EMNLP parsing papers"* |
| $\text{conj}_4$ | .00 | .00 | .01 | .00 | *"2019 EMNLP parsing papers by..."* |
| negation | .01 | .55 | .01 | .57 | *"state with no rivers"* |
| aggregation | .01 | .04 | .01 | .03 | *"total population of all states"* |
| count | .10 | .02 | .16 | .02 | *"number of papers published by..."* |

Table 1: Logical operators frequency within the total number of examples. $\text{conj}_n$ refers to a logical from that contains a conjunction of $n$ clauses.

| Dataset | $D$ | $\lvert\mathcal{D}_{\text{on}}\rvert$ | $\mathcal{D}_{\text{nat}}$ coverage |
|---|---|---|---|
| SCHOLAR | 7 | 655,516 | 95.16 |
| | 6 | 91,739 | 90.85 |
| | 5 | 2,283 | 63.73 |
| | 4 | 91 | 17.44 |
| GEOQUERY | 7 | 2,086,830 | 94.13 |
| | 6 | 249,617 | 89.43 |
| | 5 | 4,480 | 76.52 |
| | 4 | 166 | 51.68 |

Table 2: Logical form coverage of $\mathcal{D}_{\text{nat}}$ by $\mathcal{D}_{\text{on}}$, after converting logical forms to their templates, for different values of $D$.

templates that appear in $\mathcal{D}_{\text{nat}}$ is important in the OVERNIGHT procedure.

Table 2 shows the number of examples generated by OVERNIGHT for different values of $D$, along with the proportion of examples in $\mathcal{D}_{\text{nat}}$ whose logical form template is covered by $\mathcal{D}_{\text{on}}$. Because OVERNIGHT requires paraphrasing each generated example, a reasonable value for $D$ is 5. We observe that in this case only $63.73\%$ of the examples in SCHOLAR are covered, as well as $76.52\%$ of the examples in GEOQUERY.

To verify whether coverage is indeed important in our setup, we performed the following experiment. We trained a semantic parser (detailed in Section 5) on $\mathcal{D}_{\text{on}}$ for GEOQUERY and SCHOLAR (with D$=5$), and evaluated its performance on the training set of $\mathcal{D}_{\text{nat}}$ (which serves as a development set in this context). Then, we calculated the accuracy of the model with respect to examples for which their template appears in $\mathcal{D}_{\text{on}}$ ($acc_{\text{cov}}$), and for those their template does not appear ($acc_{\text{disj}}$). Table 3 shows that $acc_{\text{disj}}$ is substantially lower

| Dataset | $acc_{\text{cov}}$ | $acc_{\text{disj}}$ |
|---|---|---|
| GEOQUERY | 78.73 | 10.71 |
| SCHOLAR | 65.58 | 5.24 |

Table 3: Denotation accuracy for the set of examples in $\mathcal{D}_{\text{nat}}$ covered by $\mathcal{D}_{\text{on}}$ ($acc_{\text{cov}}$), and for the set of uncovered examples ($acc_{\text{disj}}$).

than $acc_{\text{cov}}$ for both datasets. This shows the importance of generating logical form templates that are likely to appear in the target distribution. Thus, our finding reinforces the conclusions in (Finegan-Dollak et al., 2018), and shows that while neural semantic parsers obtain good performance on existing benchmarks, their generalization to new compositional structures is limited.

**Unlikely Logical Forms** In OVERNIGHT, logical forms that are unlikely to appear in a natural interaction could be generated. For instance, the canonical utterance *"total publication year of paper whose keyphrase is semantic parsing"*, which refers to the sum of publication years of all semantic parsing papers. Although such logical forms are valid with respect to type matching, users are unlikely to produce them, and they are hard to prune automatically. To estimate the logical form mismatch caused by unlikely logical forms, we manually inspected 100 random examples from $\mathcal{D}_{\text{on}}$ with D$=5$ for GEOQUERY, and found that 31% of the examples are unlikely. While these examples are not necessarily harmful for a model, they are difficult to paraphrase, and may introduce noisy paraphrases that hurt performance.

**Example 1**
$c$: *"river that traverses california and that has the largest length"*
$x_{\text{on}}$: *"What river that traverses California has the largest length?"*
$x_{\text{nat}}$: *"What is the longest river in California?"*

**Example 2**
$c$: *"paper whose keyphrase is deep learning and that has the largest publication year"*
$x_{\text{on}}$: *"Name the paper with a deep learning keyphrase that has the most recent publication year."*
$x_{\text{nat}}$: *"Most recent deep learning papers."*

Figure 3: Examples for canonical utterances ($c$) generated by the grammar, their paraphrase by crowd workers ($x_{\text{on}}$), and their natural utterance in $\mathcal{D}_{\text{nat}}$ ($x_{\text{nat}}$). While the paraphrases are correct, they are biased towards the language style in $c$.

### 3.2 Language mismatch

The second mismatch in OVERNIGHT, demonstrated in Figure 3, is between the language used when naturally interacting with a conversational interface, and the language crowd workers use when elicited to generate paraphrases conditioned on a canonical utterance.

To directly measure the language mismatch, we performed the following experiment. We generated a dataset $\mathcal{D}_{\text{lang}}$ by taking the examples from $\mathcal{D}_{\text{nat}}$ and paraphrasing their logical forms using the OVERNIGHT procedure. This ensures that $p_{\text{lang}}(z) = p_{\text{nat}}(z)$, and thus the only difference between $\mathcal{D}_{\text{nat}}$ and $\mathcal{D}_{\text{lang}}$ is due to *language mismatch*. Then we measured the difference in performance when training on these two datasets.

In more detail, for every example in $\mathcal{D}_{\text{nat}}$, we extracted the corresponding canonical utterance template (since the examples are covered by the OVERNIGHT grammar), denoted by $\mathcal{C}_{\text{nat}}$. This list of canonical utterances can be viewed as an oracle output of the OVERNIGHT grammar generation procedure, since they do not exhibit any logical form mismatch. Next, 12 NLP graduate students paraphrased the examples in $\mathcal{C}_{\text{nat}}$. The students were presented with guidelines and examples for creating paraphrases, similar to the original guidelines in Wang et al. (2015). Moreover, we explicitly asked to paraphrase the canonical utterances such that the output is significantly different from the input (while preserving the meaning). After the paraphrase process was completed, we manually fixed typos in all generated paraphrases. Thus, our paraphrasing procedure yields high quality paraphrases and is an upper bound to what can be achieved by crowd workers.

| Model | GEOQUERY | SCHOLAR |
|---|---|---|
| SUPERVISED+ELMO | 86.3 | 83.4 |
| OVERNIGHT-ORACLE-LF+ELMO | 77.7 | 73.5 |
| SUPERVISED+GLOVE | 84.9 | 78.2 |
| OVERNIGHT-ORACLE-LF+GLOVE | 71.6 | 67.2 |

Table 4: Denotation accuracy on the test set, comparing a semantic parser trained on $\mathcal{D}_{\text{nat}}$ (SUPERVISED) and parser trained on $\mathcal{D}_{\text{lang}}$ (OVERNIGHT-ORACLE-LF).

We trained a neural semantic parser on both $\mathcal{D}_{\text{nat}}$ (named SUPERVISED) and $\mathcal{D}_{\text{lang}}$ (named OVERNIGHT-ORACLE-LF) using the original train/development split. Results in Table 4 show that for each domain, a decrease of approximately 9 points in accuracy occurs only due to the language mismatch, even with high-quality workers. This gap is likely to grow when workers are not experienced in paraphrasing, or are unfamiliar with the domain they generate paraphrases for.

We additionally observe that when we use GloVe embeddings (Pennington et al., 2014) instead of contextual ELMo embeddings (Peters et al., 2018), the gap is even higher. This shows that better representations reduce the language gap, though it is still substantial.

## 4 Grammar-driven Annotation

To overcome the logical form and language mismatches discussed, we propose in this section a new data generation procedure that does not suffer from these shortcomings. Our procedure, named GRANNO, relies on two assumptions. First, unlike OVERNIGHT, we assume access to unlabeled utterances $\mathcal{X}_{\text{ul}} = \{x_i\}_{i=1}^{N_{\text{ul}}}$. These can typically be found in query logs, or generated by users experimenting with a prototype. Second, we assume a scoring function $s_0(x, c)$, which provides a reasonable initial similarity score between a natural language utterance $x$ and a canonical utterance $c$.

The goal of our procedure is to iteratively label $\mathcal{X}_{\text{ul}}$ with crowd workers, aided by the OVERNIGHT grammar. If we manage to label most of the dataset $\mathcal{X}_{\text{ul}}$, which is sampled from the true target distribution, we will end up with a labeled dataset $\mathcal{D}_{\text{ga}}$ that has very little distribution mismatch.

Figure 1 illustrates the procedure. First, we generate all canonical utterances $\mathcal{C}_{\text{on}}$ up to a depth $D$ from the OVERNIGHT grammar. Because we

**Algorithm 1** GRANNO

**Input:** unlabeled utterances $\mathcal{X}_{\text{ul}}$, grammar $G$
**Output:** $\mathcal{D}_{\text{ga}}$ - training data for a semantic parser

generate $\mathcal{C}_{\text{on}}$ from $G$
$\mathcal{D}_{\text{ga}} \leftarrow \emptyset$, $s_0(\cdot) \leftarrow -WMD(\cdot)$, $t \leftarrow 0$
*converge* $\leftarrow False$, $\mathcal{S}_{\text{pos}} \leftarrow \emptyset$, $\mathcal{S}_{\text{neg}} \leftarrow \emptyset$
**while not** *converge* **do**
  **for** $x$ **in** $\mathcal{X}_{\text{ul}}$ **do**
    calculate $\mathcal{C}_x^K$ using $s_t(\cdot)$ ▷ top $K$ candidates
    crowd workers annotate $c_x$ given $x$ and $\mathcal{C}_x^K$
    **if** $c_x \neq N/A$ **then**
      $\mathcal{D}_{\text{ga}} \leftarrow \mathcal{D}_{\text{ga}} \cup (x, c_x)$
      $\mathcal{X}_{\text{ul}} \leftarrow \mathcal{X}_{\text{ul}} \setminus x$
      $\mathcal{S}_{\text{pos}} \leftarrow \mathcal{S}_{\text{pos}} \cup (x, c_x)$ ▷ positive examples
      **for** $c$ **in** $\mathcal{C}_x^M \setminus \{c_x\}$ **do**
        $\mathcal{S}_{\text{neg}} \leftarrow \mathcal{S}_{\text{neg}} \cup (x, c)$ ▷ negative examples
  *converge* $\leftarrow$ check for convergence, $t \leftarrow t + 1$
  train $s_t(\cdot)$ over $\mathcal{S}_{\text{pos}}$ and $\mathcal{S}_{\text{neg}}$
**return** $\mathcal{D}_{\text{ga}}$

do not paraphrase all the canonical utterances, we can generate to a higher depth $D$ compared to OVERNIGHT and cover more of the examples in $\mathcal{X}_{\text{ul}}$ (see Table 2). Then, we iteratively label the utterances in $\mathcal{X}_{\text{ul}}$. At each iteration $t$, we use a paraphrase detection model $s_t(x, c)$ to present promising canonical utterances to crowd workers for each unlabeled utterance $x$, who label the dataset. Importantly, crowd workers in our setup do not *generate paraphrases*, they only *detect* them. We now describe GRANNO in more detail.

**Iterative Annotation** At each iteration $t$, we rely on a trained scoring function $s_t(x, c)$ that estimates the similarity between an unlabeled utterance $x \in \mathcal{X}_{\text{ul}}$ and a generated canonical utterance $c \in \mathcal{C}_{\text{on}}$. We follow the procedure described in Algorithm 1. For an utterance $x \in \mathcal{X}_{\text{ul}}$, we calculate the top-$K$ ($= 5$) most similar canonical utterances in $\mathcal{C}_{\text{on}}$, denoted by $\mathcal{C}_x^K$. We then present $x$ along with its candidate paraphrases $\mathcal{C}_x^K$ to a worker, and ask her to choose the correct candidate paraphrase. If a paraphrase does not appear in the top-$K$ candidates, the worker selects no candidates.

These annotations are then used to train $s_{t+1}(x, c)$, which will be used in the next iteration. For each $x \in \mathcal{X}_{\text{ul}}$ for which a worker detected a paraphrase $c_x$, we label $(x, c_x)$ as a positive example. We use the top-$M$ ($= 100$) other most similar canonical utterances $\mathcal{C}_x^M \setminus \{c_x\}$ (according to $s_t(\cdot)$) as negative examples. We train $s_{t+1}$ from all the examples generated in iterations $0 \cdots t$. Thus, in every iteration more and more examples are labeled, and a better scoring function is trained. We stop when we meet convergence, i.e., when no new unlabeled utterances are labeled. We then output the dataset $\mathcal{D}_{\text{ga}}$ that contains every utterance $x$ from $\mathcal{X}_{\text{ul}}$ that is now labeled, paired with the logical form that corresponds to the detected canonical utterance paraphrase $c_x$.

We note that an alternative modeling choice was to use the semantic parser itself as the scorer for candidate canonical utterances. However, decoupling the parser and the scorer is beneficial, as the discriminative scoring function $s_t(\cdot)$ benefits from negative examples (incorrect paraphrases), unlike the generative semantic parser.

The success of our procedure depends on a good initial scoring function $s_0(x, c)$, to be used in the first iteration, that we next describe.

**Initial Scoring Function** We implement $s_0(x, c)$ in an unsupervised manner, as no labeled examples are available in the first iteration. Formally, we take $s_0(x, c) = -WMD(x, c)$, where $WMD(x, c)$ is the Word Mover's Distance (WMD) (Kusner et al., 2015) between $x$ and $c$, which is the minimum amount of distance that the embedded words of one utterance need to travel to reach those of the other utterance. We found WMD to perform better than cosine similarity over averaged pre-trained embeddings, as WMD performs word-level alignment, and shared words (such as entities) encourage small distance.

**Implementation details** We take the unlabeled utterance set to be all utterances in $\mathcal{D}_{\text{nat}}$, when ignoring logical forms: $\mathcal{X}_{\text{ul}} = \{x \mid (x, \cdot) \in \mathcal{D}_{\text{nat}}\}$. We generate all canonical utterances up to depth D= 6, resulting in roughly 350K canonical utterances in SCHOLAR and GEOQUERY, and coverage of 90% of the examples in $\mathcal{X}_{\text{ul}}$ (Table 2).

Our binary classifier $s_t(\cdot)$ is trained from paraphrases detected by workers. We utilize the ESIM neural architecture originally used for natural language inference (Chen et al., 2017). We also employ ELMo contextualized embeddings (Peters et al., 2018), and minimize the binary cross-entropy loss for each example.

## 5 Experiments

### 5.1 Experimental Setup

**Datasets** We experiment with two popular semantic parsing datasets: GEOQUERY (Zettlemoyer and Collins, 2005) and SCHOLAR (Iyer et al., 2017), that contain questions about US geography and academic publications, respectively.

Because we utilize the original grammar from Wang et al. (2015) that generates logical forms in lambda-DCS (Liang, 2013), we first manually annotated GEOQUERY and SCHOLAR with lambda-DCS logical forms that are translations of the original logical forms (Prolog for GEOQUERY and SQL for SCHOLAR). We only convert examples that are covered by the OVERNIGHT grammar, which results in annotating 99.3% of the examples in GEOQUERY (874 in total), and 96.7% of the examples in SCHOLAR (790 in total).

**Grammar Generation** To generate the training data $\mathcal{D}_{\text{on}}$ for our OVERNIGHT baseline and GRANNO, we first exhaustively generated logical form and canonical utterance pairs from the grammar up to depth $\text{D} = 5$ for OVERNIGHT and $\text{D} = 6$ for GRANNO, using type matching rules such that vacuous logical forms are not generated. Then, we further pruned unlikely logical forms that can be automatically detected (e.g., contradictions such as *"state that borders california and that not borders california"*). We additionally pruned equivalent examples: if we generated a logical form with the structure `A ⊓ B`, we pruned `B ⊓ A`.

**Crowd Sourcing** We gathered annotations from crowd workers by running a qualification task where we manually verified workers (annotators with at least 85% success rate were qualified). Then, qualified workers performed the tasks for the remaining examples, and for those where unqualified workers failed.

For OVERNIGHT, we gathered a single paraphrase per canonical utterance in $\mathcal{D}_{\text{on}}$ with a cost of 0.06$. For GRANNO, we detected a single paraphrase for each unlabeled utterance in $\mathcal{X}_{\text{ul}}$ that was not annotated in previous iterations, with a cost of 0.05$.

In total, we gathered 7,140 paraphrases for OVERNIGHT with a total cost of 515$, and 2,594 detections for GRANNO with a total cost of 155$. Thus, GRANNO had lower cost per task, and given the results below, benefits more from fewer tasks.

**Neural Semantic Parser** We use a standard semantic parser provided by AllenNLP (Gardner et al., 2017), based on a sequence-to-sequence model (Sutskever et al., 2014). The encoder is a BiLSTM that receives ELMo pre-trained embeddings (Peters et al., 2018) as input. The attention-based decoder (Bahdanau et al., 2015) is an LSTM that also employs copying (Jia and Liang, 2016).

| Model | GEOQUERY | SCHOLAR |
|---|---|---|
| SUPERVISED | 86.3 | 83.4 |
| GRANNO | 72.0 | 69.2 |
| OVERNIGHT | 61.9 | 40.8 |
| OVERNIGHT-ORACLE-LF | 77.7 | 73.5 |
| GRANNO-ORCALE | 79.1 | 72.5 |

Table 5: Denotation accuracy on the test set.

**Training Scheme** For both the semantic parser and the paraphrase detection model $s_t(\cdot)$ we take 10% of the training data for validation (or the official development set, if it exists). When training both models, we abstract examples to their templates, as in Dong and Lapata (2016) and Finegan-Dollak et al. (2018) inter alia. For our semantic parser, we tune the learning rate and dropout over the development set, and for $s_t(\cdot)$ we use the same hyper-parameter values as in Chen et al. (2017). We use early-stopping, and choose the model with the highest denotation accuracy and $F_1$ measure for the semantic parser and $s_t(\cdot)$, respectively.

### 5.2 Results

**Main Results** Table 5 shows the denotation accuracy for all experiments, when training from: $\mathcal{D}_{\text{nat}}$ (SUPERVISED); $\mathcal{D}_{\text{on}}$ (OVERNIGHT); $\mathcal{D}_{\text{lang}}$, as described in Section 3.2 (OVERNIGHT-ORACLE-LF); $\mathcal{D}_{\text{ga}}$, described in Section 4 (GRANNO); and $\mathcal{D}_{\text{ga}}$ where we simulate a perfect worker that always detects the gold paraphrase if it appears in the top-$K$ candidates (GRANNO-ORCALE).

Results show that OVERNIGHT achieves substantially lower accuracy compared to training with examples from the target distribution (SUPERVISED), inline with the analyses presented in Section 3. For instance, SCHOLAR accuracy more than halves (40.8 for OVERNIGHT in comparison to 83.4 for SUPERVISED). Conversely, our suggested method GRANNO that directly annotates unlabeled utterances achieves much higher accuracy than OVERNIGHT. Utilizing crowd workers for detection leads to 70.6 accuracy on average, and to 75.8 when simulating perfect crowd workers (GRANNO-ORACLE).

All models performed well on the development set that was sampled from the same distribution as the training set ($> 80\%$ accuracy), and thus differences in performance are due to generalization to the true distribution. Moreover, the accuracy of the paraphrase detection model $s_t(\cdot)$ was very high ($> 95\%$ $F_1$ measure) on the develop-

| Case | % | Natural language utterance | Gold canonical utterance | Detected canonical utterance |
|---|---|---|---|---|
| *us* | .48 | *"what is the highest point in california?"* | *"place that is high point of california"* | *"high point of california"* |
| *os* | .13 | *"how many citizens in california?"* | *"population of california"* | *"total population of california"* |
| *w* | .08 | *"which state has the greatest density?"* | *"state that has the largest density"* | *"total density of state"* |
| *pw* | .31 | *"what is the lowest elevation in california?"* | *"place that is low point of california"* | *"elevation of low point of california"* |

Table 6: Examples of false positive detections by crowd workers for different cases: *under specification* (*us*), *over specification* (*os*), *wrong* (*w*) and *partially wrong* (*pw*).

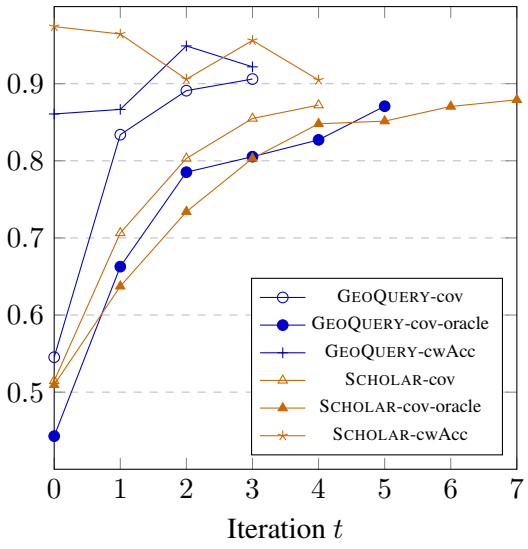


Figure 4: Analysis of GRANNO across iterations.

ment set, showing that detection is easier to model compared to generation.

We note in passing that we also implemented a baseline that uses unlabeled examples in conjunction with OVERNIGHT through self-training (Konstas et al., 2017). We trained a model with OVERNIGHT and iteratively labeled unlabeled utterances for which model confidence was high. However, we were unable to obtain good performance with this method.

**GRANNO Analysis** GRANNO iteratively annotates unlabeled utterances utilizing crowd workers. Figure 4 reports several metrics for GRANNO for each iteration: *cov* details the fraction of annotated utterances in $\mathcal{X}_{\text{ul}}$, where *cov-oracle* corresponds to the annotation coverage by GRANNO-ORACLE. In addition, cwAcc is the crowd workers' detection accuracy per iteration, with respect to the gold canonical utterances. The figure shows that in both datasets, GRANNO converges in a few iterations, and that workers' detection accuracy is high across all iterations ($> 85\%$).

An interesting phenomenon is that crowd workers (GRANNO) cover the unlabeled utterances faster than oracle workers (GRANNO-ORACLE). To analyze this, we inspect false positive, i.e., cases where the gold canonical utterance does *not* appear in the top-$K$ candidates of $s_t(\cdot)$, but the crowd worker detects some candidate as the paraphrase. Table 6 presents examples for these cases and their fraction within all false positives for iteration $t = 0$, where workers cover unlabeled utterances faster than the oracle. We find that in 61% of the cases, the choice of the workers was equivalent to the gold candidate. This is due to *under specification*, when the gold paraphrase is more specific than the detected one, or *over specification*, which is the opposite case. The other 39% are indeed errors, which we split into *wrong* detections, and *partially wrong* detections, where the detected paraphrase is different than the gold one, but is reasonable choice the phrasing of the question. E.g., for *"what is the lowest elevation in California"* it is unclear whether the answer should be a location or the elevation. This shows that many false positives are in fact correct.

**Limitations** GRANNO relies on the unsupervised function $s_0(\cdot)$ to bootstrap the annotation procedure. In both datasets, $s_0(\cdot)$ managed to rank gold paraphrases within its top-5 candidates for roughly half of the unlabeled utterances in $\mathcal{X}_{\text{ul}}$, but this is not guaranteed.

During each iteration in GRANNO, the function $s_t(\cdot)$ is applied on all pairs of an unlabeled utterance and candidate canonical utterances, thus $s_t(\cdot)$ is applied $O(|\mathcal{X}_{\text{ul}}| \cdot |\mathcal{D}_{\text{on}}|)$ times. Empirically, we find that computation time is manageable when limiting the application of $s_t(\cdot)$ to candidates that share the same entities as the unlabeled utterance. However, this might not suffice for KBs with large schemas. In such cases, an information retrieval module could retrieve a small number of candidates, similar to Yang et al. (2019).

## 6 Related Work

Several works used a human in the loop for training a semantic parser. Iyer et al. (2017) incorporated user feedback to detect wrong parses and

sent them to expert annotation. Lawrence and Riezler (2018) and Berant et al. (2019) improved a supervised parser by showing its predictions to users and training from this feedback. Gupta et al. (2018) built a hierarchical annotation scheme for annotating utterances with multiple intents. Labutov et al. (2019) trained a semantic parser through interactive dialogues. Comparing to these works, our proposed method requires no supervised data or expert annotators, and is suitable for rapid development of parsers in multiple domains.

In semi-supervised learning, Konstas et al. (2017) used self-training to improve an existing AMR parser. Others used a variational autoencoder by modeling unlabeled utterances (Kociský et al., 2016) or logical forms (Yin et al., 2018) as latent variables. However, empirical gains from the unlabeled data were relatively small compared to annotating more examples.

Finally, several papers extended the OVERNIGHT procedure. Ravichander et al. (2017) replaced phrases in the lexicon with images to elicit more natural language from workers. (Cheng et al., 2018) generated more complex compositional structures by splitting the canonical utterances into multiple steps. Such work relies on workers to *generate* paraphrases, while we propose to simply *detect* them.

## 7 Conclusion

We address the challenge of generating data for training semantic parsers from scratch in multiple domains. We thoroughly analyze the OVERNIGHT procedure, and shed light on the factors that lead to poor generalization, namely logical form mismatch and language mismatch. We then propose GRANNO, a method that directly annotates unlabeled utterances with their logical form, by letting crowd workers detect automatically-generated canonical utterances. We demonstrate our method's success on two popular datasets, and find it substantially improves generalization to real data, compared to OVERNIGHT.

## Acknowledgments

We thank the anonymous reviewers for their constructive feedback. This work was completed in partial fulfillment for the PhD degree of the first author, which was also supported by a Google PhD fellowship. This research was partially supported by The Israel Science Foundation grant 942/16, The Blavatnik Computer Science Research Fund and The Yandex Initiative for Machine Learning.

## References

Y. Artzi and L. Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics (TACL)*, 1:49–62.

D. Bahdanau, K. Cho, and Y. Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*.

Jonathan Berant, Daniel Deutch, Amir Globerson, Tova Milo, and Tomer Wolfson. 2019. Explaining queries over web tables to non-experts. *International Conference on Data Engineering (ICDE)*.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1657–1668, Vancouver, Canada. Association for Computational Linguistics.

Jianpeng Cheng, Siva Reddy, and Mirella Lapata. 2018. Building a neural semantic parser from a domain ontology. *arXiv preprint arXiv:1812.10037*.

Marco Damonte, Rahul Goel, and Tagyoung Chung. 2019. Practical semantic parsing for spoken language understanding. In *Human Language Technology and North American Association for Computational Linguistics (HLT/NAACL)*.

L. Dong and M. Lapata. 2016. Language to logical form with neural attention. In *Association for Computational Linguistics (ACL)*.

Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving text-to-sql evaluation methodology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 351–360, Melbourne, Australia. Association for Computational Linguistics.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. Allennlp: A deep semantic natural language processing platform.

Sonal Gupta, Rushin Shah, Mrinal Mohit, Anuj Kumar, and Mike Lewis. 2018. Semantic parsing for task oriented dialog using hierarchical representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2787–2792, Brussels, Belgium. Association for Computational Linguistics.

J. Herzig and J. Berant. 2017. Neural semantic parsing over multiple knowledge-bases. In *Association for Computational Linguistics (ACL)*.

J. Herzig and J. Berant. 2018. Decoupling structure and lexicon for zero-shot semantic parsing. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Drew A Hudson and Christopher D Manning. 2019. Gqa: a new dataset for compositional question answering over real-world images. *arXiv preprint arXiv:1902.09506*.

S. Iyer, I. Konstas, A. Cheung, J. Krishnamurthy, and L. Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. In *Association for Computational Linguistics (ACL)*.

R. Jia and P. Liang. 2016. Data recombination for neural semantic parsing. In *Association for Computational Linguistics (ACL)*.

J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. L. Zitnick, and R. Girshick. 2017. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Computer Vision and Pattern Recognition (CVPR)*.

T. Kociský, G. Melis, E. Grefenstette, C. Dyer, W. Ling, P. Blunsom, and K. M. Hermann. 2016. Semantic parsing with semi-supervised sequential autoencoders. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1078–1087.

I. Konstas, S. Iyer, M. Yatskar, Y. Choi, and L. Zettlemoyer. 2017. Neural AMR: sequence-to-sequence models for parsing and generation. *CoRR*, 0.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In *International Conference on Machine Learning*, pages 957–966.

T. Kwiatkowski, E. Choi, Y. Artzi, and L. Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Igor Labutov, Bishan Yang, and Tom Mitchell. 2019. Learning to learn semantic parsers from natural language supervision. *arXiv preprint arXiv:1902.08373*.

Carolin Lawrence and Stefan Riezler. 2018. Improving a neural semantic parser by counterfactual learning from human bandit feedback. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1820–1830, Melbourne, Australia. Association for Computational Linguistics.

P. Liang. 2013. Lambda dependency-based compositional semantics. *arXiv preprint arXiv:1309.4408*.

P. Liang, M. I. Jordan, and D. Klein. 2011. Learning dependency-based compositional semantics. In *Association for Computational Linguistics (ACL)*, pages 590–599.

Nicholas Locascio, Karthik Narasimhan, Eduardo De Leon, Nate Kushman, and Regina Barzilay. 2016. Neural generation of regular expressions from natural language with minimal domain knowledge. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1918–1923, Austin, Texas. Association for Computational Linguistics.

J. Pennington, R. Socher, and C. D. Manning. 2014. GloVe: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. 2018. Deep contextualized word representations. In *North American Association for Computational Linguistics (NAACL)*.

Abhilasha Ravichander, Thomas Manzini, Matthias Grabmair, Graham Neubig, Jonathan Francis, and Eric Nyberg. 2017. How would you say it? eliciting lexically diverse dialogue for supervised semantic parsing. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 374–383, Saarbrücken, Germany. Association for Computational Linguistics.

Pararth Shah, Dilek Hakkani-Tür, Gokhan Tür, Abhinav Rastogi, Ankur Bapna, Neha Nayak, and Larry Heck. 2018. Building a conversational agent overnight with dialogue self-play. *arXiv preprint arXiv:1801.04871*.

Y. Su and X. Yan. 2017. Cross-domain semantic parsing via paraphrasing. In *Empirical Methods in Natural Language Processing (EMNLP)*.

I. Sutskever, O. Vinyals, and Q. V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3104–3112.

Y. Wang, J. Berant, and P. Liang. 2015. Building a semantic parser overnight. In *Association for Computational Linguistics (ACL)*.

Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019. End-to-end open-domain question answering with BERTserini. *arXiv preprint arXiv:1902.01718*.

Pengcheng Yin, Chunting Zhou, Junxian He, and Graham Neubig. 2018. StructVAE: Tree-structured latent variable models for semi-supervised semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 754–765, Melbourne, Australia. Association for Computational Linguistics.

M. Zelle and R. J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Association for the Advancement of Artificial Intelligence (AAAI)*, pages 1050–1055.

L. S. Zettlemoyer and M. Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Uncertainty in Artificial Intelligence (UAI)*, pages 658–666.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.

# Improving Distantly-Supervised Relation Extraction with Joint Label Embedding

**Linmei Hu[1], Luhao Zhang[1], Chuan Shi[1]*, Liqiang Nie[2], Weili Guan[3], Cheng Yang[1]**
[1]Beijing University of Posts and Telecommunications, China
[2]Shan Dong University, China
[3]Hewlett Packard Enterprise Singapore, Singapore
{hulinmei,zhangluhao,shichuan}@bupt.edu.cn
{nieliqiang, honeyguan, albertyang33}@gmail.com

## Abstract

Distantly-supervised relation extraction has proven to be effective to find relational facts from texts. However, the existing approaches treat labels as independent and meaningless one-hot vectors, which cause a loss of potential label information for selecting valid instances. In this paper, we propose a novel multi-layer attention-based model to improve relation extraction with joint label embedding. The model makes full use of both structural information from Knowledge Graphs and textual information from entity descriptions to learn label embeddings through gating integration, while avoiding the imposed noise with an *attention mechanism*. Then the learned label embeddings are used as *another attention* over the instances (whose embeddings are also enhanced with the entity descriptions) for improving relation extraction. Extensive experiments demonstrate that our model significantly outperforms state-of-the-art methods.

## 1 Introduction

Knowledge Graphs (KGs) such as Freebase and DBpedia have shown their strong power in many natural language processing tasks including question answering (Zhang et al., 2018) and dialog generation (Zhou et al., 2018). However, these KGs are far from complete. Relation extraction, which aims to fill this gap by extracting semantic relationships between entity pairs from plain texts, is thus of great importance.

Most existing supervised relation extraction methods require a large number of labeled training data, which is time-consuming and laborious. Distant supervision has been proposed by (Mintz et al., 2009) to address the challenge. It assumes that if two entities have a relation in KGs, then all sentences mentioning the two entities express this relation. Thus, distant supervision can automatically generate a large number of labeled data without labor cost. Simultaneously, it often suffers from wrong labeling problem (Surdeanu et al., 2012; Zeng et al., 2015).

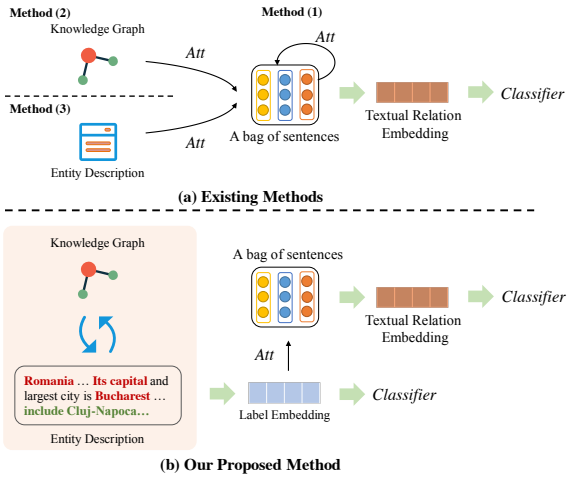


Figure 1: Illustration of comparison of existing methods and our proposed method.

Recently, significant progress has been made in applying deep neural networks for relation extraction under distant supervision (Zeng et al., 2014, 2015; Feng et al., 2017). To alleviate the wrong labeling problem in distant supervision, attention models have been proposed to select valid instances (Ji et al., 2017). As shown in Figure 1 (a), they can be divided into three categories: (1) typical attention models without external information (Lin et al., 2016; Du et al., 2018), (2) attention models using KGs (Han et al., 2018a), (3) attention models using side information such as entity descriptions (Vashishth et al., 2018; Ji et al., 2017). However, they all have flaws in selecting valid instances due to failing to exploit potential label information. They treat labels as independent and meaningless one-hot vectors, which cause a loss

*Corresponding author: Chuan Shi.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3821–3829,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

of potential label information. Label embeddings aim to learn representations of labels based on related information. The label embeddings can be used to attend over the bag of instances for relation classification. Additionally, they don't take advantage of both structural information from KGs and textual information from entity descriptions and ignore the imposed noise.

In this paper, we propose a novel multi-layer attention-based model RELE (**R**elation **E**xtraction with Joint **L**abel **E**mbedding) to improve relation extraction. Our model integrates both structural information from KGs and textual information from entity descriptions with a gating mechanism to learn label embeddings, while avoiding the imposed noise (highlighted in green) in entity descriptions with an *attention mechanism*. Then the label embeddings are used as *another antention* over the bag of instances to select valid ones for improving relation extraction. Note that we also enhance the instance embedding with entity descriptions. The contributions of this paper can be summarized as follows:

- We propose a novel multi-layer attention-based model RELE to improve distantly supervised relation extraction with joint label embedding. The label embeddings can be used to attend over the bag of instances for relation classification.
- RELE makes full use of both structural information from KGs and textual information of entity descriptions to learn label embeddings through gating integration, while avoiding the imposed noise with attention.
- Extensive experiments on two benchmark datasets have demonstrated that our model significantly outperforms state-of-the-art methods on distantly-supervised relation extraction.

## 2 Related Work

Our work is mainly related to distant supervision, neural relation extraction and label embedding.

**Distant Supervision.** Most supervised relation extraction methods require large-scale labeled training data which are expensive. Distant Supervision proposed by (Mintz et al., 2009) is an effective method to automatically label large-scale training data under the assumption that if two entities have a relation in a KG, then all sentences mentioning those entities express this relation. The assumption does not work in all cases and causes the mislabeling problem.

MultiR (Hoffmann et al., 2011) and MIMLRE (Surdeanu et al., 2012) introduce multi-instance learning where the instances mentioning the same entity pair are processed at a bag level. However, these methods rely heavily on handcrafted features.

**Neural Relation Extraction.** With the development of deep learning, neural networks have proven to be efficient to automatically extract valid features from sentences in recent years. Some researches (Zeng et al., 2014, 2015) adopt Convolution Neural Networks (CNN) to learn sentence representations automatically. To alleviate the mislabeling problem, attention mechanisms (Lin et al., 2016; Du et al., 2018) have been employed. Apart from that, some studies apply other relevant information to improve relation extraction (Zeng et al., 2017; Vashishth et al., 2018; Han et al., 2018b,a; Ji et al., 2017). For example, RESIDE (Vashishth et al., 2018) utilizes the available side information from knowledge bases, including entity types and relation alias information. Han et al. (2018a) proposed a joint representation learning framework of KGs and instances, which leverages the KG embeddings to select valid instances. APCNN+D (Ji et al., 2017) exploits the entity descriptions as background knowledge for selection of valid instances, and ignores the imposed noise.

Different from the existing works, we propose a novel multi-layer attention-based model RELE with joint label embedding. Our model makes full use of both structural information from KGs and textual information from entity descriptions to learn label embeddings through gating integration, while avoiding the imposed noise with an *attention mechanism*. The label embeddings are then used as *another attention* over the bag of instances to select valid instances for relation classification.

**Label Embedding.** Label embedding has been widely exploited in computer vision including image classification (Akata et al., 2016) and text recognition (Rodriguez-Serrano et al., 2015). Recently, LEAM (Wang et al., 2018) successfully applies label embedding in text classification, which obtains each label's embedding by its corresponding text descriptions. In this work, we are the first to apply it for relation extraction. We propose

a novel multi-layer attention-based model to improve relation extraction with joint label embedding.

## 3 Preliminaries

In this section, we briefly introduce some notations and concepts used in this paper.

For convenience, we denote a KG as $G = \{(h, r, t)\}$, which contains considerable triplets $(h, r, t)$ where $h$ and $t$ are respectively head entity and tail entity, and $r$ denotes the relation. Their embeddings are denoted as $(\mathbf{h}, \mathbf{r}, \mathbf{t})$.

Formally, given a pair of entities $(h, t)$ in a KG $G$ and a bag of instances (sentences) $B = \{s_1, s_2, \cdots, s_m\}$, where each instance $s_i$ contains $(h, t)$, the task of *relation extraction* is to train a classifier based on $B$ to predict the relation label $y$ of $(h, t)$ from a predefined relation set. If no relation exists, we simply assign NA to it.

To improve relation extraction, we make full use of both the KG $G$ and entity descriptions $D = \{d_1, d_2, \cdots, d_n\}$ to learn label embeddings which can benefit selection of valid instances. For each entity $e_i$, we take the first paragraph of its corresponding Wikipedia page as its description text $d_i = \{w_1, w_2, \cdots, w_l\}$, where $w_i \in V$ denotes the description word, $l$ is the length and $V$ is the vocabulary.

## 4 Our Proposed Model

In this section, we will detail our proposed multi-layer attention-based model RELE for relation extraction with joint label embedding. Existing methods for relation extraction take labels as independent and meaningless one-hot vectors, which cause a loss of potential label information for selecting valid instances. Additionally, they don't take full advantage of both structural information from KGs and textual information from entity descriptions and ignore the imposed noisy information.

As shown in Figure 2, our model is based on a multi-layer attention, containing two parts: 1) label embedding (shown in the right) and 2) neural relation classification (shown in the left) . The former makes full use of structural information from KGs and textual information from entity descriptions to learn label embeddings through gating mechanism, while avoiding the imposed noise with an attention mechanism. The latter leverages the label embeddings as another attention over the

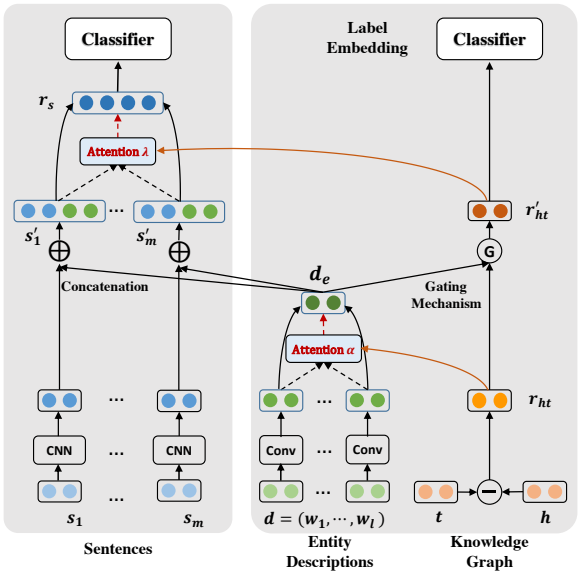


Figure 2: Illustration of our multi-layer attention based model RELE.

instances to select valid instances for improving neural relation extraction. Note that we also use the entity descriptions to enhance the representations of instances. We detail the two parts as follows.

### 4.1 Joint Label Embedding

Label information plays a vital role in selecting valid instances for improving relation extraction. We make full use of both structural information from KGs (Han et al., 2018a) and textual information from entity descriptions (Ji et al., 2017) to learn label embeddings via gating integration. Entity descriptions provide rich background knowledge for entities (Newman-Griffis et al., 2018) and are supposed to benefit the label embedding and relation extraction. Nevertheless, as shown in Figure 1, entity descriptions may also contain irrelevant and even misleading information. Therefore, we propose to use KG embeddings to attend over the entity descriptions, alleviating the imposed noise. Then a gating mechanism is used to integrate both the KGs and entity descriptions for learning label embeddings.

**KG Embedding**. We use TransE (Bordes et al., 2013) for KG embedding. Given a triplet $(h, r, t)$, the model aims to learn low-dimensional representations vector for entities $h$, $t$ and the relationship $r$ into the same vector space, and regards a relationship $r$ as a translation from the head entity $h$ to tail entity $t$, assuming the embedding $\mathbf{t}$ should

be close to $\mathbf{h}+\mathbf{r}$ if $(h, r, t)$ exists. The score function is defined as :

$$f(h, r, t) = -\|\mathbf{h}+\mathbf{r}-\mathbf{t}\|_2^2. \quad (1)$$

Note that since the true relations in test set are unknown, we simply represent the relation by:

$$\mathbf{r}_{ht} = \mathbf{t} - \mathbf{h}. \quad (2)$$

In this way, we can also get the relation embeddings given the entity pairs during testing.

**Entity Description Embedding**. Then we use the representations of relations $\mathbf{r}_{ht}$ as attention over the words of an entity description to reduce the weights of noisy words. Formally, for each entity $e$, we learn the representation of its description $d = (w_1, w_2, \cdots, w_l)$ as follows:

$$\mathbf{x}_i = \text{CNN}(\mathbf{w}_{i-\frac{c-1}{2}}, \cdots, \mathbf{w}_{i+\frac{c-1}{2}}), \quad (3)$$

$$\hat{\mathbf{x}}_i = \tanh(\mathbf{W}_x \cdot \mathbf{x}_i + \mathbf{b}_x), \quad (4)$$

$$\alpha_i = \frac{\exp(\hat{\mathbf{x}}_i \cdot \mathbf{r}_{ht})}{\sum_{i=1} \exp(\hat{\mathbf{x}}_i \cdot \mathbf{r}_{ht})}, \quad (5)$$

$$\mathbf{d}_e = \sum_{i=1}^{l} \alpha_i \mathbf{x}_i, \quad (6)$$

where CNN$(\cdot)$ denotes a convolution layer with window size $c$ over the word sequence. $\mathbf{x}_i \in \mathbb{R}^{D_h}$ is the hidden representation of the word $w_i$. $\mathbf{W}_x$ is the weight matrix and $\mathbf{b}_x$ is the bias vector. $\alpha_i$ is the attention weight of the word $w_i$, which is computed based on the relation embedding $\mathbf{r}_{ht}$. Finally, the text description embedding $\mathbf{d}_e$ is computed by the weighted average of words.

**Gating Integration**. We apply a gating mechanism (Xu et al., 2016) to integrate the textual entity description embedding $\mathbf{d}_e$ and the structural information (entity embedding $\mathbf{e}$) from KGs:

$$\mathbf{e}' = \mathbf{g} \odot \mathbf{e} + (1-\mathbf{g}) \odot \mathbf{d}_e, \quad (7)$$

where $\mathbf{g} \in \mathbb{R}^{D_w}$ is a gating vector for integration, $\mathbf{e}' \in \mathbb{R}^{D_w}$ represents the final integrated entity embedding and $\odot$ represents Hadamard product. Consequently, we compute the final label embedding $\mathbf{l}$:

$$\mathbf{l} = \mathbf{t}' - \mathbf{h}'. \quad (8)$$

**Label Classifier**. Ideally, each label embedding is supposed to act as an "anchor" for each relation class. To achieve this goal, we consider to train the learned label embeddings $\mathbf{l}$ to be easily classified as the correct relation class. Therefore, we use softmax to get the predicted probabilities of the relation classes:

$$\text{P}(y|D, G) = \text{Softmax}(\mathbf{M}_k\mathbf{l} + \mathbf{b}_k), \quad (9)$$

where $\mathbf{M}_k$ is the transformation matrix, $\mathbf{b}_k$ is the bias.

### 4.2 Neural Relation Extraction

After obtaining the embeddings of labels and entity descriptions, we leverage them to advance the neural relation extraction. We first use the entity descriptions to enhance instance embeddings. Then we leverage the label embeddings to attend over the instances to select valid instances for relation classification.

**Instance Embedding**. We enrich the representation of an instance with the pair of entity descriptions. Firstly, for each word $w \in s = \{w_1, \cdots, w_n\}$, its embedding $\hat{\mathbf{w}}_i$ is initialized as follows:

$$\hat{\mathbf{w}}_i = \mathbf{w}_i \oplus \mathbf{p}_{i1} \oplus \mathbf{p}_{i2}, \quad (10)$$

where $\mathbf{w}_i$ is the pre-trained word vector of $w_i$, $\mathbf{p}_{i1}$ and $\mathbf{p}_{i2}$ are its position embeddings to incorporate relative distances to two target entities into two $D_p$-dimensional vectors respectively (Zeng et al., 2014). The symbol $\oplus$ represents concatenation operator.

Then, we choose CNN (Zeng et al., 2014) with window size $c$ as our encoder to learn the instance embedding considering the text of the instance itself.

$$\mathbf{z}_i = \text{CNN}(\hat{\mathbf{w}}_{i-\frac{c-1}{2}}, \cdots, \hat{\mathbf{w}}_{i+\frac{c-1}{2}}), \quad (11)$$

$$[\mathbf{s}]_j = \max\{[\mathbf{z}_1]_j, \cdots, [\mathbf{z}_n]_j\}, \quad (12)$$

where $\mathbf{s} \in \mathbb{R}^{D_h}$ is the sentence (instance) embedding, $[\cdot]_j$ is the $j$-th value of a vector and function max denotes max-pooling.

Finally, we concatenate the original instance embedding $\mathbf{s}$ with the entity descriptions $(\mathbf{d}_h, \mathbf{d}_t)$, obtaining the new instance representation $\mathbf{s}'$. Formally,

$$\mathbf{s}' = \mathbf{s} \oplus \mathbf{d}_h \oplus \mathbf{d}_t. \quad (13)$$

**Attention over Instances**. To alleviate the wrong labeling problem of distant supervision, we leverage the label embedding $\mathbf{l}$ as attention over

instances to reduce the weights of noisy instances in the sentence bag $B = \{s_1, \cdots, s_m\}$. Then the representation of textual relation feature $\bar{\mathbf{s}}$ from the bag $B$ can be calculated as weighted average of the instance embeddings $\mathbf{s}'_i$:

$$\hat{\mathbf{s}}_i = \tanh(\mathbf{W}_s \mathbf{s}'_i + \mathbf{b}_s), \tag{14}$$

$$\lambda_i = \frac{\exp(\mathbf{l} \cdot \hat{\mathbf{s}}_i)}{\sum_{j=1}^{m} \exp(\mathbf{l} \cdot \hat{\mathbf{s}}_i)}, \tag{15}$$

$$\bar{\mathbf{s}} = \sum_{i=1}^{m} \lambda_i \mathbf{s}'_i, \tag{16}$$

where $\mathbf{W}_s$ is the weight matrix and $\mathbf{b}_s$ is the bias vector. $\lambda_i$ is the attention score of the instance $s_i$, computed based on the label embedding $\mathbf{l}$.

**Relation Classifier**. Finally, to compute the confidence of each relation class, we feed the representation of the textual relation $\bar{\mathbf{s}}$ into a softmax classifier after being processed by a linear transformation. Formally,

$$\mathrm{P}(y|B) = \mathrm{Softmax}(\mathbf{M}_s \bar{\mathbf{s}} + \mathbf{b}_s), \tag{17}$$

where $\mathbf{M}_s$ is the transformation matrix, and $\mathbf{b}_s$ is the bias.

### 4.3 Model Training

The objective function of our joint model RELE includes two parts, the loss of label classifier $L_1$ and the loss of relation classifier $L_2$.

Assuming that there are $N$ bags in training set $\{B_1, B_2, \cdots, B_N\}$, and their corresponding labels $\{y_1, y_2, \cdots, y_N\}$, we exploit cross entropy for the loss function of label classifier $L_1$:

$$L_1 = -\sum_{i=1}^{N} \log \mathrm{P}(y_i|D, G), \tag{18}$$

The loss $L_1$ aims to train the label embedding to be classified to the correct relation class.

Similarly, for the loss of relation classifier $L_2$, we also exploit cross entropy and get:

$$L_2 = -\sum_{i=1}^{N} \log \mathrm{P}(y_i|B_i), \tag{19}$$

Finally, we aim to minimize the loss function $L$ with L2-norm:

$$\min L = L_1 + L_2 + \eta \|\Theta\|^2, \tag{20}$$

where $\eta$ is the regularization coefficient and $\Theta$ denotes the parameters in our model. Stochastic gradient descent (SGD) is used to optimize our model.

## 5 Experiments

### 5.1 Datasets

In our experiments, we evaluate our model over the NYT-FB60K dataset (Han et al., 2018a) and GIDS-FB8K dataset (Jat et al., 2018). In the following, we detail each dataset.

- NYT-FB60K. The dataset NYT-FB60K (Han et al., 2018a) includes three parts: knowledge graphs (FB60K extended from Riedel dataset (Riedel et al., 2010), containing 1,324 relations, 69,512 entities, and 335,350 facts), text corpus (whose sentences are from Riedel dataset, containing 570,088 sentences, 63,696 entities, and 56 relations) and entity descriptions (which are the first paragraphs of the entities' Wikipedia pages, containing around 80 words on average).

- GIDS-FB8K. We construct the dataset GIDS-FB8K based on GIDS dataset (Jat et al., 2018). It also contains three parts: knowledge graphs (FB8K extended from GIDS dataset, containing 208 relations, 8,917 entities, and 38,509 facts), text corpus (whose sentences are from GIDS dataset, containing 16,960 sentences, 14,261 entities, and 5 relations) and entity descriptions (which are the first paragraphs of the entities' Wikipedia pages, containing around 80 words on average).

### 5.2 Baselines

We compare our model with the state-of-the-art baselines:

- Mintz (Mintz et al., 2009). A multi-class logistic regression model under distant supervision.

- MultiR (Hoffmann et al., 2011). A probabilistic graphical model for multi-instance learning.

- MIMLRE (Surdeanu et al., 2012). A graphical model for multi-instance multi-label learning.

- CNN+ATT (Lin et al., 2016). A CNN model with instance-level attention.

- APCNN+D (Ji et al., 2017) A Piecewise CNN model with instance-level attention using entity descriptions. As no code available, we implemented it by ourselves.

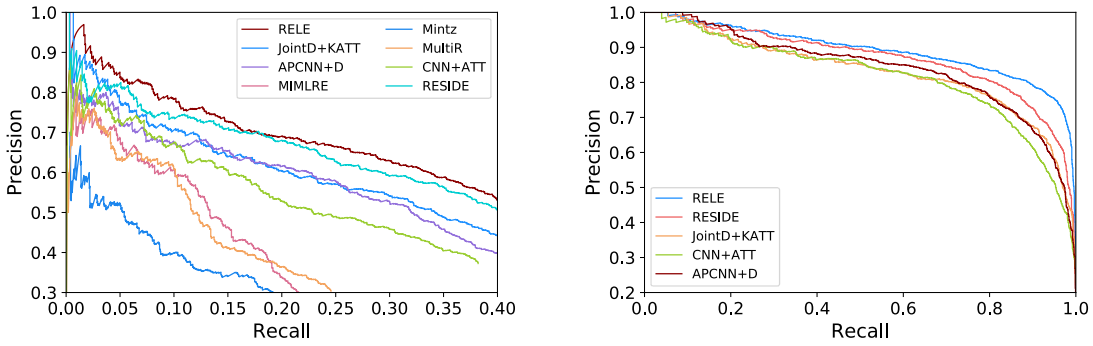


(a) NYT-FB60K Dataset (b) GIDS-FB8K Dataset

Figure 3: Precision-recall curves for different methods. RELE achieves higher precision over the entire range of recall compared to all the baselines on both datasets.

| Parameter | Value |
|---|---|
| Word/Entity/Relation Dimension $D_w$ | 50 |
| Position Dimension $D_p$ | 5 |
| Hidden Layer Dimension $D_h$ | 230 |
| Kernel Size $c$ | 3 |
| Learning Rate $\alpha$ | 0.5 |
| Regularization Coefficient $\eta$ | 0.0001 |
| Dropout Probability $p$ | 0.5 |

Table 1: Parameter Settings.

- JointD+KATT (Han et al., 2018a). A joint model for knowledge graph embedding and relation extraction.

- Reside (Vashishth et al., 2018). A neural network based model which makes use of relevant side information and employs Graph Convolution Networks for encoding syntactic information of instances.

### 5.3 Evaluation Metrics

Following previous studies (Lin et al., 2016), our model is evaluated held-out, which compares the relations discovered from test corpus with those in Freebase. We report the Precision-Recall curve and top-N precision (P@N) metric for NYT-FB60K dataset.

For GIDS-FB8K dataset, we report the Precision-Recall curve, F1 score and Mean Average Precision (MAP). We do not use the top-N precision (P@N) metric for the dataset which is small containing only 5 relation classes.

### 5.4 Parameter Settings

For all the models, we use the pre-trained word embeddings with word2vec tool* on NYT corpus for initialization. The embeddings of entities mentioned in datasets are pre-trained through TransE model. We select the learning rate $\alpha$ among $\{0.1, 0.01, 0.005, 0.001\}$ for minimizing the loss. For other parameters, we simply follow the settings used in (Lin et al., 2016; Han et al., 2018a) so that it can be fairly compared with these baselines. Table 1 shows all the parameters used in our experiment.

### 5.5 Experiment Results

#### 5.5.1 Precision-Recall Curves on Both Datasets

Figure 3 shows the comparison results in terms of Precision-Recall Curves on NYT-FB60K and GIDS-FB8K datasets. Overall, we observe that: (1) As shown in Figure 3(a), the neural network based approaches have more obvious advantages than Mintz, MultiR, and MIMLRE, illustrating the limitation of human-designed features and the advancement of neural networks in relation extraction. (2) APCNN+D using entity descriptions, and JointD+KATT exploiting KGs both outperform CNN+ATT on both datasets, showing that entity descriptions and KGs are both useful for improving the performance of relation extraction under distant supervision. RESIDE achieves better performance than APCNN+D and JointD+KATT. It is probably because that RESIDE utilizes more available side information, including entity types and relation alias information. (3) Our model

*https://code.google.com/p/word2vec/

| Test Setttings | One | | | | Two | | | | All | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| P@N(%) | 100 | 300 | 500 | Mean | 100 | 300 | 500 | Mean | 100 | 300 | 500 | Mean |
| CNN+ATT | 67.3 | 61.1 | 56.5 | 61.6 | 73.3 | 62.1 | 53.5 | 63.0 | 74.2 | 62.4 | 55.9 | 64.1 |
| APCNN+D | 78.0 | 67.7 | 60.6 | 68.7 | 77.0 | 63.3 | 55.6 | 65.3 | 77.0 | 66.7 | 63.0 | 68.9 |
| JointD+KATT | 80.0 | 70.0 | 64.0 | 71.3 | 81.0 | 67.0 | 59.6 | 69.2 | 84.0 | 68.7 | 63.7 | 71.0 |
| RESIDE | 79.0 | 67.6 | 57.4 | 68.0 | 83.0 | 70.6 | 62.6 | 72.0 | 84.0 | 78.5 | 69.8 | 77.4 |
| RELE | **87.0** | **76.6** | **67.0** | **76.8** | **88.0** | **73.3** | **63.2** | **78.7** | **88.0** | **78.6** | **69.8** | **78.8** |

Table 2: Evaluation results P@N of different models using different number of sentences in bags on NYT-FB60K dataset. Here, **One**, **Two** and **All** represent the number of sentences randomly selected from a bag.

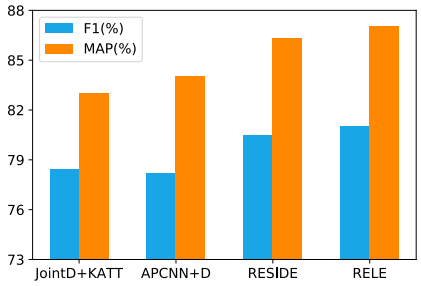


Figure 4: Comparison on GIDS-FB8K dataset.

| P@N(%) | 100 | 300 | 500 | Mean |
|---|---|---|---|---|
| RELE w/o LE | 74.2 | 62.4 | 55.9 | 64.1 |
| RELE w/o ATT$_e$ | 82.0 | 75.6 | 69.6 | 75.7 |
| RELE w/o LC | 81.0 | 74.0 | 69.0 | 74.7 |
| RELE | **88.0** | **78.6** | **69.8** | **78.8** |

Table 3: Evaluation results P@N of variant models on NYT-FB60K dataset.

RELE achieves the best performance compared to all the baselines. We believe the reason is that we make use of potential label information through joint label embedding. The learned label embeddings are of high quality since we fully exploit both the structural information from KGs and textual information from texts via gating integration while avoiding the imposed noise by an attention mechanism.

#### 5.5.2 P@N Evaluation on NYT-FB60K Dataset

As shown in Table 2, we report Precision@N of different neural network based approaches on NYT-FB60K dataset. To verify the performance of our model on those entity pairs with few instances, we randomly select one, two and all instances for each entity pair, following previous studies (Du et al., 2018; Vashishth et al., 2018). As we can observe: (1) APCNN+D and JointD+KATT both outperform CNN+ATT in all cases, demonstrating the effectiveness of entity descriptions and KGs. RESIDE achieves better performance than APCNN+D and JointD+KATT by incorporating more side information (e.g., entity types). (2) Our model RELE significantly outperforms all the baselines. The reason is that our model learns high quality of label embeddings which play a critical role in relation extraction.

#### 5.5.3 Results on GIDS-FB8K Dataset

Based on the results on the NYT-FB60K dataset, we choose APCNN+D, JointD+KATT and RESIDE as representative baselines to compare against our models on the GIDS-FB8K dataset in terms of F1 and MAP. As shown in Figure 4, our model consistently achieves better performance, which verifies the effectiveness of our model with joint label embedding. The overall results on GIDS-FB8K dataset show that our model can be well applied to smaller-scale datasets.

#### 5.5.4 Comparison of Variant Models

In order to verify the effectiveness of different modules of our model, we design three variant models:

- **RELE w/o LE** removes the label embedding from RELE, which degenerates to CNN+ATT.
- **RELE w/o ATT$_e$** removes the attention of the KG over entity descriptions during label embedding. We use max-pooling instead, which does not consider the noise in the entity descriptions.

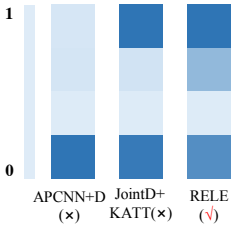


| |
|---|
| $s_1$: In **Bucharest** , **Romania** , president Traian Basescu said Sunday night that the entry into … |
| $s_2$: Last year, it opened offices in Warsaw and **Bucharest** , the capital of **Romania**. |
| $s_3$: … by the presence of two films from **Romania** , the way I spent … east of **Bucharest** … |
| $s_4$:… Ervin a of New York City , on April 18th , age 98 , formerly of **Bucharest** , **Romania** . |
| Triplet(*Romania, Bucharest, /location/country/capital*) |

Figure 5: A case study for predicting the relation between "Bucharest" and "Romania". The baselines all predict wrongly while our model gives the right result. The left shows the weights assigned to different sentences by different models. Our model always gives higher weights to correct sentences (shown in red).

Romania is a country located at the crossroads of Central , Eastern , and Southeastern Europe . It borders the Black Sea to the southeast , … , Romania is the 12th largest country and also the 7th most populous ... Its capital and largest city is Bucharest , and other major urban areas include Cluj-Napoca ...

(a) Description text of "Romania"

Bucharest is the capital and largest city of Romania , as well as its cultural , industrial , and financial centre . It is located in the southeast of the country , … ,on the banks of the Dambovita River , less than 60 km north of the Danube River and the Bulgarian border.

(b) Description text of "Bucharest"

Figure 6: Visualization of attention values of words in the descriptions of the entities "Romania" and "Bucharest".

- **RELE w/o LC** removes the label classifier. It does not train the label embeddings to be classified to the correct classes.

As shown in Table 3, without label embedding, the performance of RELE w/o LE drops significantly (more than 10%). It demonstrates the effectiveness of our joint label embedding. If we do not consider the imposed noise in the entity descriptions, the performance of RELE w/o $\text{ATT}_e$ decreases by around 3% on mean P@N. It demonstrates that the attention of KGs over entity descriptions is important for learning high-quality label embeddings. We also explore the performance of RELE w/o LC which does not train the label embeddings to be classified to the correct classes and find that label classifier plays a critical role in label embedding. The performance decreased by around 4% on mean P@N, without label classifier.

#### 5.5.5 Case Study

Figure 5 illustrates an example from the test set of NYT-FB60K. As we can observe, representative attention-based baselines APCNN+D and JointD+KATT both predict wrong relation labels for the entities, while our model RELE correctly predicts the relation label "*/location/country/capital*". That is because the baselines assign relatively low weights to correct sentences ($s_1$ and $s_2$) and achieve inferior representations of the textual relation for classification. Our model RELE assigns higher weights to all the correct sentences $s_1$, $s_2$ and $s_4$, demonstrating that RELE learns high-quality label embeddings.

We also provide the insights of the attention of KGs over entity descriptions about "Romania" and "Bucharest". As shown in Figure 6, the words "capital", "largest". and "centre" which are related to the relation are given higher weights. It demonstrates that the attention of KGs over entity descriptions can help reduce the noise in entity descriptions and thus improve the label embeddings.

## 6 Conclusion

In this work, we consider leveraging potential label information to select valid instances for distantly-supervised relation extraction. We propose a novel multi-layer attention-based model RELE to improve relation extraction with joint label embedding. Our model takes full advantage of both structural information from KGs and textual information from entity descriptions to learn label embeddings, while avoiding the imposed noise with an attention mechanism. The label embeddings are trained to be classified to the correct relation classes. Then, the learned label embeddings are used as another attention over the bag to select valid instances for relation extraction. Extensive experiments have demonstrated that our model significantly outperforms state-of-the-art methods.

In the future, we will explore other useful information (e.g., correlations among the relations from KGs) available to improve the label embeddings.

## Acknowledgement

This work is supported by the National Natural Science Foundation of China (No. 61806020, 61772082, 61702296), the National Key Research and Development Program of China (2017YFB0803304), the Beijing Municipal Natural Science Foundation (4182043), the CCF-Tencent Open Fund, and the Fundamental Research Funds for the Central Universities.

## References

Zeynep Akata, Florent Perronnin, Zaïd Harchaoui, and Cordelia Schmid. 2016. Label-embedding for image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38:1425–1438.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*, pages 2787–2795.

Jinhua Du, Jingguang Han, Andy Way, and Dadong Wan. 2018. Multi-level structured self-attentions for distantly supervised relation extraction. In *EMNLP*, pages 2216–2225.

Xiaocheng Feng, Jiang Guo, Bing Qin, Ting Liu, and Yongjie Liu. 2017. Effective deep memory networks for distant supervised relation extraction. In *IJCAI*, pages 4002–4008.

Xu Han, Zhiyuan Liu, and Maosong Sun. 2018a. Neural knowledge acquisition via mutual attention between knowledge graph and text. In *AAAI*, pages 4832–4839.

Xu Han, Pengfei Yu, Zhiyuan Liu, Maosong Sun, and Peng Li. 2018b. Hierarchical relation extraction with coarse-to-fine grained attention. In *EMNLP*, pages 2236–2245.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *ACL*, pages 541–550.

Sharmistha Jat, Siddhesh Khandelwal, and Partha Talukdar. 2018. Improving distantly supervised relation extraction using word and entity based attention. *arXiv preprint arXiv:1804.06987*.

Guoliang Ji, Kang Liu, Shizhu He, Jun Zhao, et al. 2017. Distant supervision for relation extraction with sentence-level attention and entity descriptions. In *AAAI*, pages 3060–3066.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *ACL*, volume 1, pages 2124–2133.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACL/IJCNLP*, pages 1003–1011.

Denis Newman-Griffis, Albert M Lai, and Eric Fosler-Lussier. 2018. Jointly embedding entities and text with distant supervision. *ACL 2018*, page 195.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *ECML/PKDD*, pages 148–163.

Jose A Rodriguez-Serrano, Albert Gordo, and Florent Perronnin. 2015. Label embedding: A frugal baseline for text recognition. *International Journal of Computer Vision*, 113(3):193–207.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *EMNLP-CoNLL*, pages 455–465.

Shikhar Vashishth, Rishabh Joshi, Sai Suman Prayaga, Chiranjib Bhattacharyya, and Partha Talukdar. 2018. Reside: Improving distantly-supervised neural relation extraction using side information. In *EMNLP*, pages 1257–1266.

Guoyin Wang, Chunyuan Li, Wenlin Wang, Yizhe Zhang, Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. 2018. Joint embedding of words and labels for text classification. In *ACL*, volume 1, pages 2321–2331.

Jiacheng Xu, Kan Chen, Xipeng Qiu, and Xuanjing Huang. 2016. Knowledge graph representation with jointly structural and textual encoding. In *IJCAI*, pages 1318–1324.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *EMNLP*, pages 1753–1762.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via conlutional deep neural network. In *COLING*, pages 2335–2344.

Wenyuan Zeng, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2017. Incorporating relation paths in neural relation extraction. In *EMNLP*, pages 1768–1777.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J Smola, and Le Song. 2018. Variational reasoning for question answering with knowledge graph. In *AAAI*, pages 6069–6076.

Hao Zhou, Tom Young, Minlie Huang, Haizhou Zhao, Jingfang Xu, and Xiaoyan Zhu. 2018. Commonsense knowledge aware conversation generation with graph attention. In *IJCAI*, pages 4623–4629.

# Leverage Lexical Knowledge for Chinese Named Entity Recognition via Collaborative Graph Network

**Dianbo Sui**[1,2], **Yubo Chen**[1], **Kang Liu**[1,2], **Jun Zhao**[1,2], **Shengping Liu**[3]
[1] National Laboratory of Pattern Recognition, Institute of Automation,
Chinese Academy of Sciences, Beijing, 100190, China
[2] University of Chinese Academy of Sciences, Beijing, 100049, China
[3] Beijing Unisound Information Technology Co., Ltd, Beijing, 100028, China
{dianbo.sui, yubo.chen, kliu, jzhao}@nlpr.ia.ac.cn
liushengping@unisound.com

## Abstract

The lack of word boundaries information has been seen as one of the main obstacles to develop a high performance Chinese named entity recognition (NER) system. Fortunately, the automatically constructed lexicon contains rich word boundaries information and word semantic information. However, integrating lexical knowledge in Chinese NER tasks still faces challenges when it comes to self-matched lexical words as well as the nearest contextual lexical words. We present a Collaborative Graph Network to solve these challenges. Experiments on various datasets show that our model not only outperforms the state-of-the-art (SOTA) results, but also achieves a speed that is six to fifteen times faster than that of the SOTA model.[1]

## 1 Introduction

Named entity recognition (NER) aims to locate and classify certain occurrences of words or expressions in unstructured text into predefined semantic categories such as the person names, locations, organizations, etc. NER is an essential pre-processing step for many natural language processing (NLP) applications, such as relation extraction (Bunescu and Mooney, 2005), event extraction (Chen et al., 2015), question answering (Mollá et al., 2006) etc. In English NER, LSTM-CRF models (Lample et al., 2016; Ma and Hovy, 2016; Chiu and Nichols, 2016; Liu et al., 2018) leveraging word-level representations and character-level representations achieve the state-of-the-art results.

In this paper, we focus on Chinese NER. Compared with English, Chinese has no obvious word boundaries. Since without word boundaries information, it is intuitive to use character information

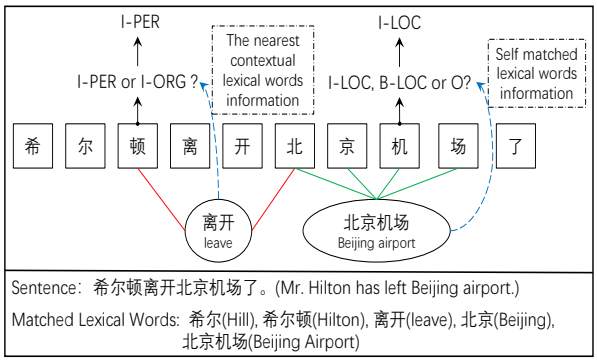


Figure 1: An example sentence integrating the nearest contextual lexical words (red line) and self-matched lexical words (green line)

only for Chinese NER (He and Wang, 2008; Liu et al., 2010; Li et al., 2014), although such methods could result in the disregard of word information. However, word information is very useful in Chinese NER, because word boundaries are usually the same as named entity boundaries. For example, as shown in Figure 1, the boundaries of the word "北京机场" (Beijing airport) are the same as the boundaries of the named entity " 北京机场" (Beijing airport). Therefore, making full use of word information would help to improve the Chinese NER performance.

There are three main ways to incorporate word information in NER. The first one is the pipeline method. The way of pipeline method is to apply Chinese Word Segmentation (CWS) first, and then to use a word-based NER model. However, the pipeline method suffers from error propagation, since the error of CWS may affect the performance of NER. The second one is to learn CWS and NER tasks jointly (Xu et al., 2013; Peng and Dredze, 2016; Cao et al., 2018; Wu et al., 2019). However, the joint models must rely on CWS annotation datasets, which are costly and are annotated under many diverse segmentation criteria (Chen

[1]The code is available at https://github.com/DianboWork/Graph4CNER



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3830–3840,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

et al., 2017). The third one is to leverage an automatically constructed lexicon, which is pre-trained on large automatically segmented texts. Lexical knowledge includes boundaries and semantic information. Boundaries information is provided by the lexicon word itself, and semantic information is provided by pre-trained word embeddings (Bengio et al., 2003; Mikolov et al., 2013). Compared with joint methods, a lexicon is easy to obtain and additional annotation CWS datasets are not required. Recently, Zhang and Yang (2018) propose a lattice LSTM to integrate lexical knowledge in NER. However, integrating lexical knowledge into sentences still faces two challenges.

The first challenge is to integrate self-matched lexical words. A self-matched lexical word of a character is the lexical word that contains this character. For instance, "北京**机**场" (Beijing Airport) and "**机**场" (Airport) are the self-matched words of the character "**机**" (airplane). "离开" (leave) is not the self-matched word of the character "机" (airplane), since " 机" (airplane) is not contained in the word " 离开" (leave). The lexical knowledge of self-matched word is useful in Chinese NER. For example, as shown in Figure 1, the boundaries and semantic knowledge of the self-matched word "北京机场" (Beijing Airport) can help the character "机"(airplane) to predict an "I-LOC" tag, instead of "O" or "B-LOC" tags. However, due to the limits of the word-character lattice, the lattice LSTM (Zhang and Yang, 2018) fails to integrate the self-matched word "北京机场" (Beijing Airport) into the character "机" (airplane).

The second challenge is to integrate the nearest contextual lexical words directly. The nearest contextual lexical word of a character is the word that matches the nearest past or future subsequence in the given sentence of this character. For instance, the lexical word "离开" (leave) is the nearest contextual word of the character "顿" (-ton), since the word matches the nearest future subsequence "离开" of the character, while "北京" (Beijing) is not the nearest contextual lexical word of this character. The nearest contextual lexical words are beneficial for Chinese NER. For example, as shown in Figure 1, by directly using the semantic knowledge of the nearest contextual words "离开" (leave), an "I-PER" tag can be predicted instead of an "I-ORG" tag, since "希尔顿" (Hilton Hotels) cannot be taken as the subject of the verb "离开" (leave). However, a lattice model (Zhang and Yang, 2018) only implicitly integrate the knowledge of the nearest contextual lexical words via the previous hidden state. The information of the nearest contextual lexical word may be disturbed by other information.

To solve the above challenges, we propose a character-based Collaborative Graph Network, including an encoding layer, a graph layer, a fusion layer and a decoding layer. Specifically, there are three word-character interactive graphs in the graph layer. The first one is the Containing graph (C-graph), which is designed for integrating self-matched lexical words. It models the connection between characters and self-matched lexical words. The second one is the Transition graph (T-graph), which builds the direct connection between characters and the nearest contextual matched words. It helps to handle the challenge of integrating the nearest contextual words directly. The third one is the Lattice graph (L-graph), which is inspired by the lattice LSTM (Zhang and Yang, 2018). L-graph captures partial information of self-matched lexical words and the nearest contextual lexical words implicitly by multiple hops. These graphs are built without external NLP tools, which can avoid error propagation problem. Besides, these graphs complement each other nicely and a fusion layer is designed for collaboration between these graphs.

We test our model with various Chinese NER datasets. our model not only significantly outperforms the existing state-of-the-art (SOTA) model but also is six to fifteen times faster than the speed of the SOTA model.

In summary, our main contributions are as follows:

- We propose a Collaborative Graph Network to integrate lexical knowledge directly and efficiently for Chinese NER.
- To solve the challenges of integrating self-matched lexical words and the nearest contextual lexical words, we propose three word-character interactive graphs. These interactive graphs can capture different lexical knowledge and are built without external NLP tools.
- We achieve the state-of-the-art results in various popular Chinese NER datasets, and our model achieves a 6-15x speedup over the existing SOTA model.

## 2 Related Work

**NER**. There is rich literature on NER. This includes statistic methods, such as SVM (Isozaki and Kazawa, 2002), HMMs (Bikel et al., 1997) and CRF (Lafferty et al., 2001), suffering from feature engineering. There are also a number of recent neural network approaches applied to NER, such as (Collobert et al., 2011; Huang et al., 2015; Lample et al., 2016; Ma and Hovy, 2016; Chiu and Nichols, 2016; Liu et al., 2018; Akbik et al., 2018; Jie et al., 2019; Akbik et al., 2019). Compared with English, Chinese is not featured with obvious word boundaries, but it is important to leverage word boundaries and semantic information in Chinese NER. Many works use word segmentation information as extra features for Chinese NER, such as (Peng and Dredze, 2015; He and Sun, 2017a; Zhu and Wang, 2019). Peng and Dredze (2016), Cao et al. (2018) and Wu et al. (2019) propose joint models to train NER together with CWS. Our work is inspired by lattice LSTM (Zhang and Yang, 2018), which can integrate lexicon in NER.

**Graph convolutional networks**. There are a number of recent graph convolutional network (GCN) architectures (Kipf and Welling, 2017; Hamilton et al., 2017; Veličković et al., 2018; Qu et al., 2019) for learning over graphs. Our work is closely related to the graph attention networks (GAT), introduced by Veličković et al. (2018), leveraging masked self-attention layers to assign different importance to neighbouring nodes. In recent years, there is more and more literature about the application of GCN in NLP (Bastings et al., 2017; Marcheggiani and Titov, 2017; Zhang et al., 2018; Yao et al., 2019; Wang et al., 2018; Mishra et al., 2019; Cao et al., 2019; Zhang et al., 2019). Cetoli et al. (2017) use GCN to investigate the role of the dependency tree in English named entity recognition. However, most of the works (Bastings et al., 2017; Marcheggiani and Titov, 2017; Cetoli et al., 2017; Zhang et al., 2018) heavily rely on the dependency tree to construct a single graph, which suffer from error propagation. To capture different semantic and boundaries information, we propose a Collaborative Graph Network consisting of three automatically constructed graphs, which can avoid error propagation problem naturally. To our best knowledge, we are the first to introduce GAT and automatically constructed semantic graphs to Chinese NER tasks.

## 3 Approach

In this section, we first introduce the construction of graphs to integrate self-matched lexical words and the nearest contextual lexical words into sentences. We then introduce the architecture of Collaborative Graph Network as a core for solving Chinese NER tasks.

### 3.1 The Construction of Graphs

To integrate self-matched lexical words and the nearest contextual lexical words, we propose three word-character interactive graphs. The first is the word-character Containing graph (C-graph), which is to assist the character to capture the boundaries and semantic information of self-matched lexical words. The second is the word-character Transition graph (T-graph). The function of T-graph is to assist the character to capture the semantic information of the nearest contextual lexical words. The third is the Lattice graph (L-graph). Zhang and Yang (2018) propose a lattice structure, nested in the LSTM (Hochreiter and Schmidhuber, 1997), to integrate lexical knowledge. We free the lattice structure from the LSTM and adopt it as the third graph.

These three graphs share the same vertex set, but the edge sets of the three graphs are completely different. The vertex set is made up of the characters in the sentence and the matched lexical words, for example, as shown in Figure 1, the vertex set is V={希, 尔,..., 了, 希尔, 希尔顿, ..., 北京机场}. To represent the edge set, adjacency matrix needs to be introduced. The elements of the adjacency matrix indicate whether pairs of vertices are adjacent or not in the graph. Since the edge sets of the three graphs are totally different, the adjacency matrices of these three graphs are introduced below:

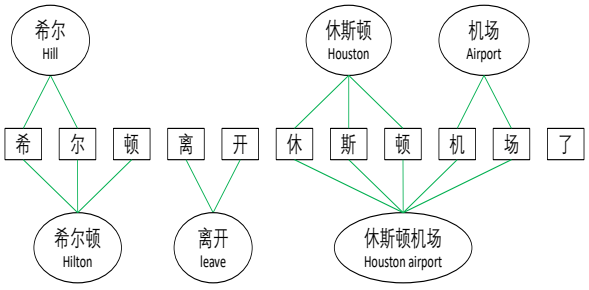


Figure 2: Word-Character Containing graph

**Word-Character Containing graph**

With the C-graph, the characters in the sentence can capture the boundaries and semantic information of self-matched lexical words. As shown in Figure 2, if a lexical word $i$ contains a character $j$, the $(i, j)$-entry of the C-graph corresponding adjacency matrix $\mathbf{A}^C$ will be assigned a value of 1.

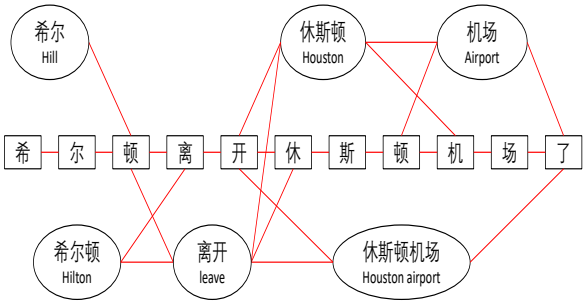


Figure 3: Word-Character Transition graph

**Word-Character Transition graph**

The T-graph is to assist the character to capture the semantic information of the nearest contextual lexical words. As shown in Figure 3, if a lexical word $i$ or a character $m$ matches the nearest preceding or following subsequence of a character $j$, the $(i, j)$ or $(m, j)$-entry of the T-graph corresponding adjacency matrix $\mathbf{A}^T$ will be assigned a value of 1. Moreover, for capturing the context relation between lexical words, if a lexical word $i$ is the preceding or following context of another lexical word $k$, we will assign "$\mathbf{A}^T_{ik} = 1$". Note that the T-graph is the same with the word cutting graph which is used in Chinese Word Segmentation.

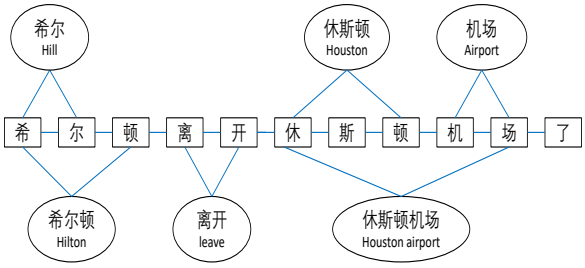


Figure 4: Word-Character Lattice graph

**Word-Character Lattice graph**

Zhang and Yang (2018) propose a lattice structure LSTM to exploit lexical knowledge for Chinese NER. A lattice structure can capture the information of the nearest contextual lexical words implicitly and capture some information of self-matched lexical words. As shown in Figure 4, if a character $m$ is the nearest preceding or following character of a character $j$, the $(m, j)$-entry of the L-graph corresponding adjacency matrix $\mathbf{A}^L$ will be assigned a value of 1. Moreover, if a character $j$ matches the lexical word $i$ first character or end character, we will assign "$\mathbf{A}^L_{ij} = 1$".

### 3.2 Model

A character-based Collaborative Graph Network includes an encoding layer, a graph layer, a fusion layer, and a decoding layer. The encoding layer is to capture contextual information of the sentence and to represent the semantic information of lexical words. The graph layer is based on GAT (Veličković et al., 2018) for modeling over three word-character interactive graphs. A fusion layer is used for fusing different lexical knowledge captured by these three graphs. Finally, a standard CRF (Lafferty et al., 2001) model is used for decoding labels.

**Encoding**

The input of the model is a sentence and all lexical words that match consecutive subsequences of the sentence. We denote the sentence as $s = \{c_1, c_2, ..., c_n\}$, where $c_i$ is the $i$-th character, and denote the matched lexical words as $l = \{l_1, l_2, ..., l_m\}$. By looking up the embedding vector from a pre-train character embedding matrix, each character $c_i$ is represented as a vector, which denotes as $\mathbf{x}_i$.

$$\mathbf{x}_i = e^c(c_i) \tag{1}$$

$e^c$ is a character embedding lookup table.

To capture contextual information, A bidirectional LSTM (Hochreiter and Schmidhuber, 1997) is applied to $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$. By concatenating the left-to-right and right-to-left LSTM hidden states, we obtain the contextual representation $\mathbf{H} = \{\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_n\}$.

$$\overrightarrow{\mathbf{h}_i} = \overrightarrow{LSTM}(\mathbf{x}_i, \overrightarrow{\mathbf{h}_{i-1}}) \tag{2}$$

$$\overleftarrow{\mathbf{h}_i} = \overleftarrow{LSTM}(\mathbf{x}_i, \overleftarrow{\mathbf{h}_{i+1}}) \tag{3}$$

$$\mathbf{h}_i = \overrightarrow{\mathbf{h}_i} \oplus \overleftarrow{\mathbf{h}_i} \tag{4}$$

To represent the semantic information of lexical words, we look up word embeddings from a

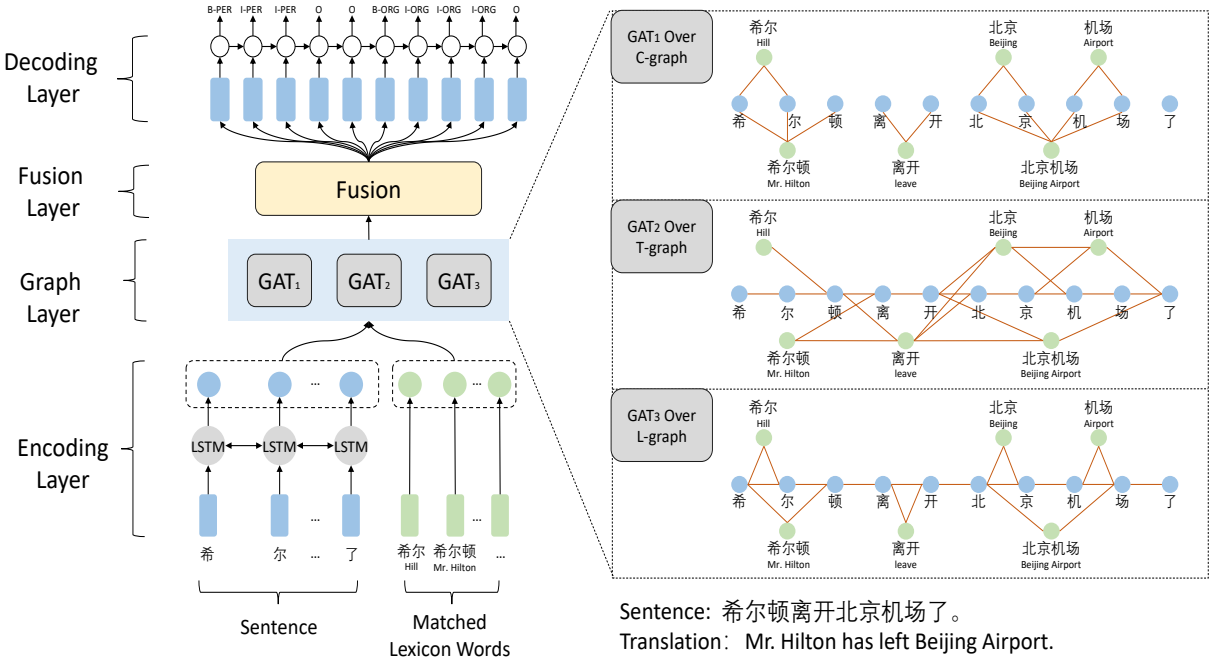


Figure 5: Main architecture of a Collaborative Graph Network for integrating lexical knowledge in Chinese NER. The left side shows the overall architecture, including an encoding layer, a graph layer, a fusion layer, and a decoding layer. On the right side, we show the details of graph attention networks over three word-character interactive graphs. We use blue to denote the characters in the sentence and use green to denote the matched lexicon words.

pre-train word embedding matrix, and each lexical words $l_i$ is represented as a semantic vector, which denotes as $\mathbf{wv}_i$.

$$\mathbf{wv}_i = e^w(l_i) \tag{5}$$

$e^w$ is a word embedding lookup table. We concatenate the contextual representation and the word embeddings as the output of this layer, denoting it as $\mathbf{Node}_f$.

$$\mathbf{Node}_f = [\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_n, \mathbf{wv}_1, \mathbf{wv}_2, ..., \mathbf{wv}_m] \tag{6}$$

### Graph Attention Networks over Word-Character Interactive Graphs

We use Graph Attention Networks (GAT) to model over three interactive graphs. In an M-layer GAT, the input of $j$-th layer is a set of node features, $\mathbf{NF}^j = \{\mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_N\}$, together with an adjacency matrix $\mathbf{A}$ , $\mathbf{f}_i \in \mathbb{R}^F$, $\mathbf{A} \in \mathbb{R}^{N \times N}$, where N denotes the number of the nodes and F is the the dimension of features at $j$-th layer. The output of $j$-th layer is a new set of node features, $\mathbf{NF}^{(j+1)} = \{\mathbf{f}'_1, \mathbf{f}'_2, ..., \mathbf{f}'_N\}$. A GAT operation with K independent attention head can be written as :

$$\mathbf{f}'_i = \Big\Vert_{k=1}^{K} \sigma\Big(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \mathbf{f}_j\Big) \tag{7}$$

$$\alpha_{ij}^k = \frac{\exp(\mathrm{LeakyReLU}(\mathbf{a}^{\mathrm{T}}[\mathbf{W}^{\mathrm{k}}\mathbf{f}_{\mathrm{i}} \| \mathbf{W}^{\mathrm{K}}\mathbf{f}_{\mathrm{j}}]))}{\Sigma_{k \in \mathcal{N}_i} \exp(\mathrm{LeakyReLU}(\mathbf{a}^{\mathrm{T}}[\mathbf{W}^{\mathrm{k}}\mathbf{f}_{\mathrm{i}} \| \mathbf{W}^{\mathrm{K}}\mathbf{f}_{\mathrm{k}}]))} \tag{8}$$

where $\|$ denotes concatenation operation, $\sigma$ is a nonlinear activation function, $\mathcal{N}_i$ is the neighborhood of node $i$ in the graph, $\alpha_{ij}^k$ are the attention coefficients, $\mathbf{W}^k \in \mathbb{R}^{F' \times F}$, and $\mathbf{a} \in \mathbb{R}^{2F'}$ is a single-layer feed-forward neural network. Note that, the dimension of the output $\mathbf{f}'_i$ is $KF'$. At the last layer, averaging will be adopted, and the dimension of final output features is $F'$.

$$\mathbf{f}_i^{final} = \sigma\Big(\frac{1}{K} \sum_{k=1}^{K} \sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \mathbf{f}_j\Big) \tag{9}$$

To model three totally different word-character interactive graphs, We build three independent graph attention networks, which are denoted as $GAT_1$, $GAT_2$, and $GAT_3$. Since three word-character interactive graphs share the same vertex set, the input node features of all GAT are matrix $\mathbf{Node}_f$, which is shown in Equation 6. The output node features are denoted as $\mathbf{G}_1$, $\mathbf{G}_2$ and $\mathbf{G}_3$,

$$\mathbf{G}_1 = GAT_1(\mathbf{Node}_f, A^C) \tag{10}$$

$$\mathbf{G}_2 = GAT_2(\mathbf{Node}_f, A^T) \tag{11}$$

$$\mathbf{G}_3 = GAT_3(\mathbf{Node}_f, A^L) \tag{12}$$

| Extra Resource | Models | Named Entity | | | Named Mention | | | Overall |
|---|---|---|---|---|---|---|---|---|
| | | P(%) | R(%) | F1(%) | P(%) | R(%) | F1(%) | F1(%) |
| Automatic word seg | Peng and Dredze (2015) | 74.78 | 39.81 | 51.96 | 71.92 | 53.03 | 61.05 | 56.05 |
| Word Seg Data | Peng and Dredze (2016) | 66.67 | 47.22 | 55.28 | 74.48 | 54.55 | 62.97 | 58.99 |
| Automatic word seg | He and Sun (2017a) | 66.93 | 40.67 | 50.60 | 66.46 | 53.57 | 59.32 | 54.82 |
| Other data | He and Sun (2017b) | 61.68 | 48.82 | 54.50 | 74.13 | 53.54 | 62.17 | 58.32 |
| Word Seg Data | Cao et al. (2018) | 59.51 | 50.00 | 54.43 | 71.43 | 47.90 | 57.53 | 58.70 |
| Automatic word seg | Zhu and Wang (2019) | - | - | 55.38 | - | - | 62.98 | 59.31 |
| Lexicon | Zhang and Yang (2018) | - | - | 53.04 | - | - | 62.25 | 58.79 |
| Lexicon | **Ours** | **67.31** | **48.61** | **56.45** | **75.15** | **62.63** | **68.32** | **63.09** |

Table 1: Main results on Weibo NER

where $\mathbf{G}_k \in \mathbb{R}^{F' \times (n+m)}$, $k \in \{1, 2, 3\}$. We keep the first $n$ columns of these matrices and discard the last $m$ columns, because only character representations are used to decode labels.

$$\mathbf{Q}_k = \mathbf{G}_k[:, 0{:}n], \ k \in \{1, 2, 3\} \quad (13)$$

**Fusion Layer**

A fusion layer is used to fuse different lexical knowledge captured by word-character interactive graphs. The input of the fusion layer is the contextual representation $\mathbf{H}$ and the output of the graph layer $\mathbf{Q}_i$, $i \in \{1, 2, 3\}$. The equation of the fusion layer is introduced below:

$$\mathbf{R} = \mathbf{W}_1\mathbf{H} + \mathbf{W}_2\mathbf{Q}_1 + \mathbf{W}_3\mathbf{Q}_2 + \mathbf{W}_4\mathbf{Q}_3 \quad (14)$$

where $\mathbf{W}_1$, $\mathbf{W}_2$, $\mathbf{W}_3$ and $\mathbf{W}_4$ are trainable matrices. Via a fusion layer, we obtain a matrix $\mathbf{R}$, $\mathbf{R} \in \mathbb{R}^{F' \times n}$, which is a new sentence representation integrating the contextual information as well as the lexical knowledge of self-matched lexical words and the nearest contextual lexical words.

**Decoding and Training**

We use a standard CRF (Lafferty et al., 2001) layer to capture the dependencies between successive labels. Given a sentence $s = \{c_1, c_2, ..., c_n\}$, the input of the CRF layer is $\mathbf{R} = \{\mathbf{r}_1, \mathbf{r}_2, ..., \mathbf{r}_n\}$, and the probability of the ground-truth tag sequence $y = \{y_1, y_2, ..., y_n\}$ is

$$p(y|s) = \frac{\exp(\sum_i (\mathbf{W}^{y_i}\mathbf{r}_i + \mathbf{T}_{(y_{i-1}, y_i)}))}{\sum_{y'} \exp(\sum_i (\mathbf{W}^{y'_i} r_i + \mathbf{T}_{(y'_{i-1}, y'_i)}))} \quad (15)$$

Here $y'$ is an arbitrary label sequence, $\mathbf{W}^{y_i}$ is used for modeling emission potential for the $i$-th character in the sentence, and $\mathbf{T}$ is the transition matrix storing the score of transferring from one tag to another. Viterbi algorithm (Viterbi, 1967) is used to get the label sequence with the highest score. Given a manually annotated training data $\{(s_i, y_i)\}|_{i=1}^N$, we optimize the model by minimizing the negative log-likelihood loss with $L_2$ regularization. The loss function is defined as:

$$L = -\sum_{i=1}^{N} log(P(y_i|s_i)) + \frac{\lambda}{2}\|\Theta\|^2 \quad (16)$$

where $\lambda$ denotes the $L_2$ regularization parameter and $\Theta$ is the all trainable parameters set

| Resource | Models | P(%) | R(%) | F1(%) |
|---|---|---|---|---|
| Gold Seg | Che et al. (2013) | 77.71 | 72.51 | 75.02 |
| | Wang et al. (2013) | 76.43 | 72.32 | 74.32 |
| | Yang et al. (2016) | 65.59 | 71.84 | 68.57 |
| | Yang et al. (2016) | 72.98 | 80.15 | 76.40 |
| | Zhu and Wang (2019) | 75.05 | 72.29 | 73.64 |
| Lexicon | Zhang and Yang (2018) | 76.35 | 71.56 | 73.88 |
| | **Ours** | **75.06** | **74.52** | **74.79** |

Table 2: Main results on OntoNotes. Gold seg means gold-standard segmentation, which is not available in the real world.

## 4 Experiments

In this section, we carry out extensive experiments to investigate the effectiveness of the Collaborative Graph Network.

### 4.1 Datasets

We evaluate our model on Weibo NER (Peng and Dredze, 2015; He and Sun, 2017a), OntoNotes 4 (Weischedel et al., 2011), and MSRA (Levow, 2006), where Weibo NER is in social domain, OntoNotes and MSRA are in the news domain. On Weibo NER, we use the same training, development and test split as Peng and Dredze (2015). On OntoNotes, we use the same data split as Che et al. (2013). Since the MSRA dataset does not have a development set, we randomly select 10% samples from the training set as the development set.

| Models | P(%) | R(%) | F1(%) |
|---|---|---|---|
| Chen et al. (2006) | 91.22 | 81.71 | 86.20 |
| Zhang et al. (2006) | 92.20 | 90.18 | 91.18 |
| Zhou et al. (2013) | 91.86 | 88.75 | 90.28 |
| Lu et al. (2016) | - | - | 87.94 |
| Dong et al. (2016) | 91.28 | 90.62 | 90.95 |
| Cao et al. (2018) | 91.73 | 89.58 | 90.64 |
| Zhu and Wang (2019) | 93.53 | 92.42 | 92.97 |
| Zhang and Yang (2018) | 93.57 | 92.79 | 93.18 |
| **Ours** | **94.01** | **92.93** | **93.47** |

Table 3: Main results on MSRA

| | Dataset | Ours(s) | Lattice(s) | Speedup |
|---|---|---|---|---|
| Training | MSRA | 344 | 13723 | ×**15** |
| | OntoNotes | 188 | 2561 | ×**13** |
| | Weibo NER | 64 | 458 | ×**7** |
| Testing | MSRA | 52 | 344 | ×**6** |
| | OntoNotes | 27.1 | 386 | ×**14** |
| | Weibo NER | 2.2 | 23 | ×**10** |

Table 4: The performance of models in training and testing time. Time is measured in seconds. Lattice means the lattice LSTM (Zhang and Yang, 2018).

### 4.2 Experimental Settings

In our experiments, We use the same character embeddings as Zhang and Yang (2018), which is pretrained on Chinese Giga-Word. We use the lexicon provided by Li et al. (2018), including 1.3 million Chinese words. We set the dimensionality of LSTM hidden states to 300 and set the initial learning rate to 0.001. Since the scale of each dataset varies, we set different training batch size for different datasets. Specifically, we set batch sizes of MSRA, OntoNotes and Weibo NER as 64, 20 and 10. We use stochastic gradient Descent (SGD) algorithm to optimize parameters in OntoNotes and WeiboNER, and use Adam (Kingma and Ba, 2014) algorithm to optimize parameters in MSRA. We stop the training when we find the best result in the development set.

### 4.3 Overall Performance

**Weibo NER**. Table 1 shows the results on Weibo NER. Zhu and Wang (2019) propose a Convolutional Attention Network using segmentation information, which is the existing state-of-the-art (SOTA) model. Our model outperforms SOTA model by 3.78%, 1.07% and 5.34% in F1 score on Overall, Named Entity, and Nominal Mention. Zhang and Yang (2018) propose a lattice LSTM to integrate lexical knowledge. Our model outperforms the lattice LSTM by 4.3%, 3.41% and 6.07% in F1 score on Overall, Named Entity, and Nominal Mention.

**OntoNotes**. Table 2 shows the results on OntoNotes. Compared with lattice LSTM (Zhang and Yang, 2018), Our model gains a 0.91% improvement in F1 score. Compared with the best result (Yang et al., 2016), our model doesn't rely on gold-standard segmentation, which is not available in the real world. Note that our model even outperforms the model proposed by (Wang et al., 2013; Yang et al., 2016; Zhu and Wang, 2019), which uses the information of gold-standard segmentation.

**MSRA**. Results on the MSRA dataset are shown in Table 3. By leveraging hand crafted features (Chen et al., 2006; Zhang et al., 2006; Zhou et al., 2013) and character embeddings (Lu et al., 2016), statistical models achieve good results on MSRA dataset. Dong et al. (2016) integrate LSTM-CRF with radical features and Zhang and Yang (2018) propose a lattice LSTM to integrate lexical knowledge. Our model outperforms the lattice LSTM by 0.29% in F1 score on MSRA datasets.

**Speed**. As an essential preprocessing NLP tool, NER tasks require high speeds of both training and testing. Since aligning word-character lattice structure for batch training is usually non-trivial, the lattice LSTM (Zhang and Yang, 2018) suffers from slow speeds in training and testing. However, both LSTM and GAT in our model can compute efficiently by batch training.

For fair comparison, both the lattice LSTM and our model are implemented under PyTorch[2]. By using a single NVIDIA GeForce GTX 1080 Ti GPU, We randomly select 10 training and testing epoch as samples. The average time of training and testing is shown in Table 4. Our model can achieve a 6-15x speedup over the lattice LSTM.

### 4.4 Effectiveness of Three Word-Character Interactive Graphs

We conduct ablation experiments to demonstrate the effectiveness of these three word-character interactive graphs.

**Comparison Setting**. We design ablation studies as follow: 1) w/o C: without word-character Containing graph(C-graph). 2) w/o T: without word-character Transition graph (T-graph). 3) w/o L: without word-character Lattice graph (L-graph). 4)w/o C & T: without C-graph and T-graph, only keep L-graph. 5)w/o C & L : without C-graph and L-graph, only keep T-graph. 6) w/o

[2] https://pytorch.org/

| | | | | | |
|---|---|---|---|---|---|
| Case1 | Sentence | …//@西安电子科技大学:#早安…<br>…// @Xidian University:#good morning… | Case2 | Sentence | 腾讯联想联合发起电脑清理日…<br>Tencent and Lenovo jointly launched a computer cleaning day… |
| | Matched lexical word | … 西安电子科技大学(XidianUniversity)，西安(Xi’ an)，<br>电子科技大学(UESTC)，安电(An Dian)，电子科技(Electronics Technology)，<br>早安(Good Morning)… | | Matched lexical word | 腾讯(Tencent), 联想(Lenovo), 联合(Joint),<br>发起(Launch), 电脑(Computer), 清理(Clean)… |
| | Sentence with gold label | …// (O)@(O)西(B-ORG) 安(I-ORG) 电(I-ORG) 子(I-ORG)<br>科(I-ORG)技(I-ORG) 大(I-ORG)学(I-ORG):(O)#(O) 早(O)安(O) … | | Sentence with gold label | 腾(B-ORG)讯(I-ORG)联(B-ORG) 想(I-ORG) 联(O)合(O)<br>发(O)起(O)电(O)脑(O)清(O)理(O)日(O)… |
| | w/o C-graph predicted label | …// (O)@(O) 西(B-LOC) 安(I-LOC) 电(B-ORG) 子(I-ORG)<br>科(I-ORG)技(I-ORG) 大(I-ORG) 学(I-ORG):(O)#(O)早(O)安(O) … | | w/o T-graph predicted label | 腾(B-ORG)讯(I-ORG)联(O)想(O)联(O)合(O)<br>发(O) 起(O)电(O)脑(O)清(O)理(O)日(O)… |
| | with C-graph predicted label | …// (O)@(O)西(B-ORG) 安(I-ORG) 电(I-ORG) 子(I-ORG)<br>科(I-ORG)技(I-ORG) 大(I-ORG)学(I-ORG):(O)#(O)早(O)安(O)… | | with T-graph predicted label | 腾(B-ORG)讯(I-ORG)联(B-ORG) 想(I-ORG) 联(O) 合(O)<br>发(O) 起(O)电(O)脑(O)清(O) 理(O)日(O)… |

Table 6: Case study. w/o C-graph predicted label means without C-graph predicted label, and w/o T-graph predicted label means without T-graph predicted label. We use green to denote the correct labels and use red to denote the wrong labels.

| Models | Dataset | | |
|---|---|---|---|
| | OntoNotes | Weibo NER | MSRA |
| Complete model | **74.79** | **63.09** | **93.47** |
| w/o C | 72.24 | 60.75 | 93.35 |
| w/o T | 71.57 | 60.94 | 93.02 |
| w/o L | 72.87 | 60.69 | 93.21 |
| w/o C & T | 70.53 | 58.51 | 92.72 |
| w/o C & L | 65.81 | 58.65 | 91.98 |
| w/o T & L | 71.41 | 58.72 | 92.80 |
| BiLSTM+CRF | 61.84 | 52.77 | 88.05 |

Table 5: Ablation study on reducing word-character interactive graphs, For example, ”w/o C” means removing word-character containing graph from the complete model.

T & L : without T-graph and L-graph, only keep L-graph. 7) BiLSTM+CRF: baseline model.

**Comparison Results**. Table 5 shows the results of ablation experiments. We can clearly see that removing any graph causes obvious performance degradation, but the importance of different graphs varies from dataset to dataset. Specifically, on OntoNotes and MSRA, ’w/o T-graph’ obtains worse performance than others, showing that T-graph is important. However, T-graph performs poorly without cooperating with other graphs. We guess that “T-graph” graph can only capture the information of the nearest contextual lexical words, and it is not enough to rely solely on T-graph. On Weibo NER, these graphs show equal importance. Since dialects slangs and irregular phrases are very common in social domain, we must rely on C-graph, T-graph, and L-graph jointly to handle the informal and complex contexts. In conclusion, from ablation experiments, we can find that each graph can be implemented independent of the other, but together they can achieve the best result, showing that all these three graphs are essential to our model.

## 5 Case Study

To show visually that our model can solve the challenges when integrating self-matched lexical words and the nearest contextual lexical words, a case study comparing without C-graph, without T-graph and the complete model is shown in Table 6. In the first case, there is an entity “ 西安电子科技大学”(Xidian University) with nested “西安” (Xi’an) and “ 电子科技大学” (UESTC). These common entities are all in the lexicon. Without C-graph, the model can’t integrate the information of the self-matched lexical word ‘西安电子科技大学” (Xidian University) into the characters “ 电” and “安”. Influenced by another lexical word “ 电子科技大学” (UESTC), the predicted label of the character “ 电” is “B-ORG”, and the label of the character “安” is predicted to be “I-ORG”, affect by the lexical word “西安” (Xi’an). In the second case, there is an entity “ 联想” (Lenovo), which can also be a common verb (“Associate”) in Chinese. Without T-graph, the model can’t integrate the information of the nearest contextual lexical words “ 腾讯” (Tencent) and “联合” (Joint) into the characters “ 联” and “ 想”, so the predicted labels of the characters “联” and “ 想” are ”O”s. However, with the help of T-graph, the model can use the information of the nearest contextual lexical words “腾讯” (Tencent) and “ 联合” (Joint) to predict the correct labels.

## 6 Conclusion

In this paper, we propose a Collaborative Graph Network for integrating lexical knowledge in Chinese NER. The core of the network is three lexical word-character interactive graphs. These interactive graphs can capture different lexical knowledge and are built without external NLP tools. We show through various experiments that our model has complementary strengths to the SOTA model and these interactive graphs are effective.

## Acknowledgments

This work is supported by the National Natural Science Foundation of China (No.61533018), the Natural Key R&D Program of China (No.2017YFB1002101), the National Natural Science Foundation of China (No.61806201) and the independent research project of National Laboratory of Pattern Recognition. This work is also supported by the CCF-Tencent Open Research Fund. We thank the anonymous reviewers for their insightful comments. We also thank Xiangrong Zeng, Pengfei Cao and Yushan Xie for helpful comments and suggestions.

## References

Alan Akbik, Tanja Bergmann, and Roland Vollgraf. 2019. Pooled contextualized embeddings for named entity recognition. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 724–728.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *COLING 2018: The 27th International Conference on Computational Linguistics*, pages 1638–1649.

Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Simaan. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1957–1967.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Daniel M. Bikel, Scott Miller, Richard Schwartz, and Ralph Weischedel. 1997. Nymble: a high-performance learning name-finder. In *Fifth Conference on Applied Natural Language Processing*.

Razvan C Bunescu and Raymond J Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 724–731.

Pengfei Cao, Yubo Chen, Kang Liu, Jun Zhao, and Shengping Liu. 2018. Adversarial transfer learning for Chinese named entity recognition with self-attention mechanism. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 182–192.

Yu Cao, Meng Fang, and Dacheng Tao. 2019. BAG: Bi-directional attention entity graph convolutional network for multi-hop reasoning question answering. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 357–362.

Alberto Cetoli, Stefano Bragaglia, Andrew O'Harney, and Marc Sloan. 2017. Graph convolutional networks for named entity recognition. In *Proceedings of the 16th International Workshop on Treebanks and Linguistic Theories*, pages 37–45.

Wanxiang Che, Mengqiu Wang, Christopher D. Manning, and Ting Liu. 2013. Named entity recognition with bilingual constraints. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 52–62.

Aitao Chen, Fuchun Peng, Roy Shan, and Gordon Sun. 2006. Chinese named entity recognition with conditional probabilistic models. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 173–176.

Xinchi Chen, Zhan Shi, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-criteria learning for Chinese word segmentation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1193–1203.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 167–176.

Jason PC Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(Aug):2493–2537.

Chuanhai Dong, Jiajun Zhang, Chengqing Zong, Masanori Hattori, and Hui Di. 2016. Character-based lstm-crf with radical-level features for chinese named entity recognition. In *Natural Language Understanding and Intelligent Applications*, pages 239–250.

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034.

Hangfeng He and Xu Sun. 2017a. F-score driven max margin neural network for named entity recognition in Chinese social media. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 713–718.

Hangfeng He and Xu Sun. 2017b. A unified model for cross-domain and semi-supervised named entity recognition in chinese social media. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Jingzhou He and Houfeng Wang. 2008. Chinese named entity recognition and word segmentation based on character. In *Proceedings of the Sixth SIGHAN Workshop on Chinese Language Processing*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Hideki Isozaki and Hideto Kazawa. 2002. Efficient support vector classifiers for named entity recognition. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Zhanming Jie, Pengjun Xie, Wei Lu, Ruixue Ding, and Linlin Li. 2019. Better modeling of incomplete annotations for named entity recognition. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 729–734.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270.

Gina-Anne Levow. 2006. The third international chinese language processing bakeoff: Word segmentation and named entity recognition. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 108–117.

Haibo Li, Masato Hagiwara, Qi Li, and Heng Ji. Ji. 2014. Comparison of the impact of word segmentation on name tagging for chinese and japanese. In *LREC*, page 2532 ¨ C2536.

Shen Li, Zhe Zhao, Renfen Hu, Wensi Li, Tao Liu, and Xiaoyong Du. 2018. Analogical reasoning on chinese morphological and semantic relations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 138–143.

Liyuan Liu, Jingbo Shang, Frank Xu, Xiang Ren, Jian Gui, Huan Peng, and Jiawei. Han. 2018. Empower sequence labeling with task-aware neural language model. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Zhangxun Liu, Conghui Zhu, and Tieju Zhao. 2010. Chinese named entity recognition with a sequence labeling approach: based on characters, or based on words? In *Advanced intelligent computing theories and applications. With aspects of artificial intelligence*, page 634 ¨ C640.

Yanan Lu, Yue Zhang, and Donghong Ji. 2016. Multi-prototype Chinese character embedding. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 855–859.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1506–1515.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Pushkar Mishra, Marco Del Tredici, Helen Yannakoudakis, and Ekaterina Shutova. 2019. Abusive Language Detection with Graph Convolutional Networks. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2145–2150.

Diego Mollá, Menno van Zaanen, and Daniel Smith. 2006. Named entity recognition for question answering. In *Proceedings of the Australasian Language Technology Workshop 2006*, pages 51–58, Sydney, Australia.

Nanyun Peng and Mark Dredze. 2015. Named entity recognition for Chinese social media with jointly trained embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 548–554.

Nanyun Peng and Mark Dredze. 2016. Improving named entity recognition for Chinese social media with word segmentation representation learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 149–155.

Meng Qu, Yoshua Bengio, and Jian Tang. 2019. Gmnn: Graph markov neural networks. In *International Conference on Machine Learning*, pages 5241–5250.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. *International Conference on Learning Representations*.

Andrew Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*, 13(2):260–269.

Mengqiu Wang, Wanxiang Che, and Christopher D Manning. 2013. Effective bilingual constraints for semi-supervised learning of named entity recognizers. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*.

Zhichun Wang, Qingsong Lv, Xiaohan Lan, and Yu Zhang. 2018. Cross-lingual knowledge graph alignment via graph convolutional networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 349–357.

Ralph Weischedel, Sameer Pradhan, Lance Ramshaw, Martha Palmer, Nianwen Xue, Mitchell Marcus, Ann Taylor, Craig Greenberg, Eduard Hovy, Robert Belvin, et al. 2011. Ontonotes release 4.0. *LDC2011T03, Philadelphia, Penn.: Linguistic Data Consortium*.

Fangzhao Wu, Junxin Liu, Chuhan Wu, Yongfeng Huang, and Xing Xie. 2019. Neural chinese named entity recognition via cnn-lstm-crf and joint training with word segmentation. In *The World Wide Web Conference*, WWW '19, pages 3342–3348.

Yan Xu, Yining Wang, Tianren Liu, Jiahua Liu, Yubo Fan, Yi Qian, Junichi Tsujii, and Eric I Chang. 2013. Joint segmentation and named entity recognition using dual decomposition in chinese discharge summaries. *Journal of the American Medical Informatics Association*, 21(e1):e84–e92.

Jie Yang, Zhiyang Teng, Meishan Zhang, and Yue Zhang. 2016. Combining discrete and neural features for sequence labeling. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 140–154.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In *Thirty-Third AAAI Conference on Artificial Intelligence*.

Ningyu Zhang, Shumin Deng, Zhanlin Sun, Guanying Wang, Xi Chen, Wei Zhang, and Huajun Chen. 2019. Long-tail relation extraction via knowledge graph embeddings and graph convolution networks. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3016–3025.

Suxiang Zhang, Ying Qin, Juan Wen, and Xiaojie Wang. 2006. Word segmentation and named entity recognition for sighan bakeoff3. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 158–161.

Yue Zhang and Jie Yang. 2018. Chinese NER using lattice LSTM. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1554–1564.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2205–2215.

Junsheng Zhou, Weiguang Qu, and Fen Zhang. 2013. Chinese named entity recognition via joint identification and categorization. *Chinese journal of electronics*, 22(2):225–230.

Yuying Zhu and Guoxin Wang. 2019. CAN-NER: Convolutional Attention Network for Chinese Named Entity Recognition. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3384–3393.

# Looking Beyond Label Noise: Shifted Label Distribution Matters in Distantly Supervised Relation Extraction

**Qinyuan Ye***
University of Southern California
qinyuany@usc.edu

**Liyuan Liu***
University of Illinois, Urbana-Champaign
ll2@illinois.edu

**Maosen Zhang**
Purdue University
maosenzhang.milo@gmail.com

**Xiang Ren**
University of Southern California
xiangren@usc.edu

## Abstract

In recent years there is a surge of interest in applying distant supervision (DS) to automatically generate training data for relation extraction (RE). In this paper, we study the problem *what limits the performance of DS-trained neural models*, conduct thorough analyses, and identify a factor that can influence the performance greatly, *shifted label distribution*. Specifically, we found this problem commonly exists in real-world DS datasets, and without special handing, typical DS-RE models cannot automatically adapt to this shift, thus achieving deteriorated performance. To further validate our intuition, we develop a simple yet effective adaptation method for DS-trained models, *bias adjustment*, which updates models learned over the source domain (*i.e.*, DS training set) with a label distribution estimated on the target domain (*i.e.*, test set). Experiments demonstrate that bias adjustment achieves consistent performance gains on DS-trained models, especially on neural models, with an up to 23% relative F1 improvement, which verifies our assumptions. Our code and data can be found at https://github.com/INK-USC/shifted-label-distribution.

## 1 Introduction

Aiming to identify the relation among an entity pair, relation extraction (RE) serves as an important step towards text understanding and is a long-standing pursuit by many researchers. To reduce the reliance on human-annotated data, especially for data-hungry neural models (Zeng et al., 2014; Zhang et al., 2017), there have been extensive studies on leveraging *distant supervision* (DS) in conjunction with external knowledge bases to automatically generate large-scale training data (Mintz et al., 2009; Zeng et al., 2015).

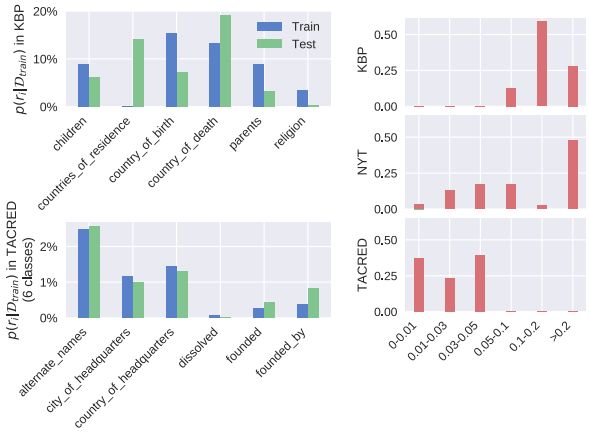


**Figure 1: Left:** Label distributions of KBP (distantly supervised dataset) are shifted, while those of TACRED (human-annotated dataset, 6 classes for illustration) are consistent. Full distributions for NYT and TACRED can be found in Appendix B. **Right:** Each relation $r_i$ is categorized into intervals along x-axis according to $|p(r_i|\mathcal{D}_{train}) - p(r_i|\mathcal{D}_{test})|$; height of the bars indicates proportion of test instances that fall into each category, revealing that the shift is severe in DS datasets such as KBP and NYT.

While recent DS-based relation extraction methods focus on handling *label noise* (Riedel et al., 2010; Hoffmann et al., 2011; Lin et al., 2016), *i.e.*, false labels introduced by error-prone DS process, other factors may have been overlooked. Here, we observe model behaviors to be different on DS datasets and clean dataset, which implies existence of other challenges that restrict performance of DS-RE models. In this paper, we conduct thorough analyses over both real-world and synthetic datasets to explore the question — *what limits the performance of DS-trained neural models*.

Our analysis starts with a performance comparison among recent relation extraction methods on both DS datasets (*i.e.*, KBP (Ellis et al., 2012), NYT (Riedel et al., 2010)) and human-annotated dataset (*i.e.*, TACRED (Zhang et al., 2017)), with the goal of seeking models that can consistently yield strong results. We observe that, on human-

*Equal contribution.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3841–3850, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

annotated dataset, neural relation extraction models outperform feature-based models by notable gaps, but these gaps diminish when the same models are applied to DS datasets—neural models merely achieve performance comparable with feature-based models. We endeavor to analyze the underlying problem that leads to this unexpected "diminishing" phenomenon.

Inspired by two heuristic threshold techniques that prove to be effective on DS datasets, and further convinced by comprehensive analysis on synthetic datasets, we reveal an important characteristic of DS datasets—*shifted label distribution*, the issue that the label distribution of training set does not align with that of test set. There often exists a large margin between label distributions of *distantly-supervised training set* and that of *human-annotated test set*, as shown in Fig 1. Intuitively, this is mainly caused by "false positive" and "false negative" labels generated by error-prone DS processes, and the imbalanced data distribution of external knowledge bases.

To some extent, such distortion is a special case of domain shift — *i.e.*, training the model on a source domain and applying the learned model to a different target domain. To further verify our assumption, we develop a simple domain adaption method, *bias adaptation*, to address the shifted label distribution issue. It modifies the bias term in softmax classifiers and explicitly fits models along the shift. Specifically, the proposed method estimates the label distribution of target domain with a small development set sampled from test set, and derives the adapted predictions under reasonable assumptions. In our experiments, we observe consistent performance improvement, which validates that model performance may be severely hindered by label distribution shift.

In the rest of the paper, we first introduce the problem setting in Section 2 and report the inconsistency of model performance with human annotations and DS in Section 3. Then, we present two threshold techniques which are found to be effective on DS datasets and further lead us to the discovery of shifted label distribution. We explore its impact on synthetic datasets in Section 4, and introduce the bias adjustment method in Section 5. In addition, comparison of denoising method, heuristic threshold and bias adjustment is conducted in Section 6. We discuss related work in Section 7 and conclude our findings in Section 8.

| Dataset | Distantly Supervised | | Human-annotated |
|---|---|---|---|
| | KBP | NYT | TACRED |
| #Relation Types | 7 | 25 | 42 |
| #Train Sentences | 23,784 | 235,982 | 37,311 |
| #Test Sentences | 289 | 395 | 6,277 |

**Table 1:** Statistics of Datasets Used in Our Study.

## 2 Experiment Setup

In this paper, we conduct extensive empirical analyses on distantly supervised relation extraction (DS-RE). For a meaningful comparison, we ensure the same setup in all experiments. In this section, we provide a brief introduction on the setting, while more details could be found in Appendix A. All implementations are available at https://github.com/INK-USC/shifted-label-distribution.

### 2.1 Problem Setting

Following previous works (Ren et al., 2017; Liu et al., 2017), we conduct relation extraction at *sentence level*. Formally speaking, the basic unit is the relation mention, which is composed of one sentence and one ordered entity pair within the sentence. The relation extraction task is to categorize each relation mention into a given set of relation types, or a Not-Target-Type (NONE).

### 2.2 Datasets

We select three popular relation extraction datasets as benchmarks. Specifically, two of them are distantly supervised and one is human-annotated.

**KBP** (Ling and Weld, 2012) uses Wikipedia articles annotated with Freebase entries as train set, and manually-annotated sentences from 2013 KBP slot filling assessment results (Ellis et al., 2012) as test set.

**NYT** (Riedel et al., 2010) contains New York Times news articles and has been already heuristically annotated. Test set is constructed manually by (Hoffmann et al., 2011).

**TACRED** (Zhang et al., 2017) is a large-scale crowd-sourced dataset, and is sufficiently larger than previous manually annotated datasets.

### 2.3 Pre-processing

We leverage pre-trained GloVe (Pennington et al., 2014) embedding[1], and use the StanfordNLP toolkit (Manning et al., 2014) to get part of speech

[1] http://nlp.stanford.edu/data/glove.840B.300d.zip

| | Method / Dataset | Distantly-supervised Wiki-KBP | Distantly-supervised NYT | Human-annotated TACRED |
|---|---|---|---|---|
| Feature-based | CoType-RM (Ren et al., 2017) | 28.98 ± 0.76 | 40.26 ± 0.51 | 45.97 ± 0.34 |
| | ReHession (Liu et al., 2017) | 36.07 ± 1.06 | 46.79 ± 0.75 | 58.06 ± 0.54 |
| | Logistic (Mintz et al., 2009) | 37.58 ± 0.27 | 47.33 ± 0.44 | 51.67 ± 0.03 |
| Neural | CNN (Zeng et al., 2014) | 30.53 ± 2.26 | 46.75 ± 2.79 | 56.96 ± 0.43 |
| | PCNN (Zeng et al., 2015) | 31.58 ± 0.35 | 44.63 ± 2.70 | 58.39 ± 0.71 |
| | Bi-GRU | **37.77 ± 0.18** | 47.88 ± 0.85 | 65.38 ± 0.60 |
| | Bi-LSTM | 34.51 ± 0.99 | 48.15 ± 0.87 | 62.74 ± 0.23 |
| | PA-LSTM (Zhang et al., 2017) | 37.28 ± 0.81 | 46.33 ± 0.64 | **65.69 ± 0.48** |
| | Bi-GRU-ATT (Lin et al., 2016) | 37.50 ± 1.21 | **49.67 ± 1.06** | - |
| | PCNN-ATT (Lin et al., 2016) | 33.74 ± 2.19 | 46.82 ± 0.82 | - |

**Table 2: Performance Comparison of RE Models.** 5-time average and standard deviation of F1 scores are reported.

(POS) tags, named-entity recognition (NER) tags, and dependency parsing trees.

For the development sets, we use the provided development set on TACRED, and randomly split 10% of training set on KBP and NYT. This development set is also refered to as *noisy dev*, as we will introduce a *clean dev* to deal with shifted label distribution in the later parts.

### 2.4 Models

We consider two classes of relation extraction methods, *i.e.*, feature-based and neural models.

Specifically, Feature-based models include Logistic Regression (Mintz et al., 2009), CoType-RM (Ren et al., 2017) and ReHession (Liu et al., 2017). Neural models include Bi-LSTMs and Bi-GRUs (Zhang et al., 2017), Position-Aware LSTM (Zhang et al., 2017), CNNs and PCNNs (Zeng et al., 2014, 2015).

For each relation mention, these models will first construct a representation vector $\mathbf{h}$, and then make predictions with softmax based on $\mathbf{h}$[2]:

$$p(y = r_i|\mathbf{h}) = \frac{\exp(\mathbf{r}_i^T\mathbf{h} + b_i)}{\sum_{r_j}\exp(\mathbf{r}_j^T\mathbf{h} + b_j)}, \quad (1)$$

where $\mathbf{r}_i$ and $b_i$ are the parameters corresponding to *i*-th relation type. More details on these models can be found in Appendix A.

## 3 The Diminishing Phenomenon

Neural models alleviate the reliance on handcrafted features and have greatly advanced the state-of-the-art, especially on datasets with human annotations. Meanwhile, we observe such performance boost starts to diminish on distantly supervised datasets. Specifically, we list the performance of all tested models in Table 2 and summarize three popular models in Figure 2.

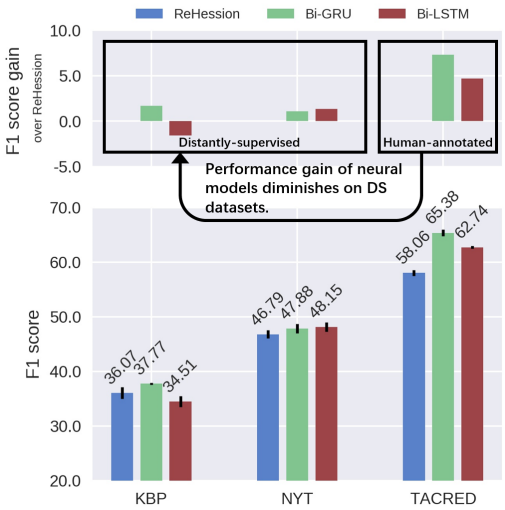


**Figure 2: F1 score comparison among three models.** Bi-GRU outperforms ReHession with a significant gap on TACRED, but only has comparable performance (with ReHession) on KBP and NYT. Similar gap diminishing phenomenon happens to Bi-LSTM.

On TACRED, a human annotated dataset, complex neural models like Bi-LSTM and Bi-GRU significantly outperform feature-based ReHession, with an up to 13% relative F1 improvement. On the other hand, on distantly supervised datasets (KBP and NYT), the performance gaps between the aforementioned methods diminishes to within 5% (relative improvement). We refer to this observation as "diminishing" phenomenon. Such observation implies a lack of handling of the underlying difference between human annotations and distant supervision.

After a broad exploration, we found two heuristic techniques that we believe capture problems exclusive to distantly supervised RE, and are potentially related to "diminishing" phenomenon. We found they can greatly boost the performance on distantly supervised datasets, but fail to do so

[2]CoType-RM and logistic regression are exceptions as they don't adopt softmax to generate output.

on human-annotated dataset. To get deeper insights, we analyze the diminishing phenomenon and the two heuristic methods.

### 3.1 Heuristic Threshold Techniques

Max threshold and entropy threshold are designed to identify the "ambiguous" relation mentions (*i.e.*, predicted with a low confidence) and label them as the NONE type (Ren et al., 2017; Liu et al., 2017). In particular, referring to the original predictions as $r^* = \arg\max_{r_i} p(y = r_i|\mathbf{h})$, we formally introduce these two threshold techniques:

- **Max Threshold** introduces an additional hyper-parameter $T_m$, and adjusts the prediction as (Ren et al., 2017):

$$predict(\mathbf{h}) = \begin{cases} r^*, & p(y = r^*|\mathbf{h}) > T_m \\ \text{NONE}, & \text{Otherwise} \end{cases}.$$

- **Entropy Threshold** introduces an additional hyper-parameter $T_e$. It first calculates the entropy of prediction:

$$e(\mathbf{h}) = -\sum_{r_k} p(y = r_k|\mathbf{h}) \log p(y = r_k|\mathbf{h}),$$

then it adjusts prediction as (Liu et al., 2017):

$$predict(\mathbf{h}) = \begin{cases} r^*, & e(\mathbf{h}) < T_e \\ \text{NONE}, & \text{Otherwise} \end{cases}.$$

To estimate $T_e$ or $T_m$, 20% instances are sampled from the test set as an additional development set, and used to tune the value of $T_e$ and $T_m$ with grid search. After that, we evaluate the model performance on the remaining 80% of the test set. We refer to this new dev set as *clean dev* and refer to the original dev set used for tuning other parameters as *noisy dev*. We would like to highlight that tuning threshold on this clean dev is necessary as it acts as a bridge between distantly supervised train set and human-annotated test set.

### 3.2 Results and Discussion

Results of three representative models (ReHession, Bi-GRU and Bi-LSTM) using threshold techniques are summarized in Figure 3. Full results are listed in Table 3. We observe significant improvements on distantly supervised datasets (*i.e.*, KBP and NYT), with a up to 19% relative F1 improvement (Bi-GRU from 37.77% to 45.01% on KBP). However, on the human-annotated corpus, the performance gain can be hardly noticed. Such inconsistency implies that these heuristics may capture some important but overlooked factors for distantly supervised relation extraction, while we are still unclear about their underlying mechanisms.

Intuitively, annotations provided by distant supervision differ from human annotations in two ways: (1) **False Positive:** falsely annotating unrelated entities in one sentence as a certain relation type; (2) **False Negative:** neglecting related entities by marking their relationship as NONE. The terms, false positive and false negative, are commonly used to describe wrongful predictions. Here we only borrow the terms for describing label noises. These label noises distort the true label distribution of train corpora, creating a gap between the label distribution of train and test set (*i.e.*, shifted label distribution). With existing denoising methods, the effect of noisy training instances may be reduced; still, it would be infeasible to recover the original label, and thus label distribution shift remains an unsolved problem.

In our experiments, we notice that in distantly supervised datasets, instances labeled as NONE have a larger portion in test set than in train set. It is apparent that the strategy of rejecting "ambiguous" predictions would guide the model to predict more NONE types, leading the predicted label distribution towards a favorable direction. Specifically, in the train set of KBP, 74.25% instances are annotated as NONE and 85.67% in the test set. The original Bi-GRU model would annotate 75.72% instances to be NONE, which is close to 74.25%; after applying the max-threshold and entropy-threshold, this proportion becomes 86.18% and 88.30%, which are close to 85.67%.

Accordingly, we believe part of the underlying mechanism of heuristic threshold is to better handle the label distribution difference between train and test set. We try to further verify this hypothesis with experiments in next section.

## 4 Shifted Label Distribution

In this section, we first summarize our observation on shifted label distribution, and then conduct empirical analysis to study its impact on model performance using synthetic datasets. These datasets are carefully created so that label distribution is the only variable and other factors are controlled.

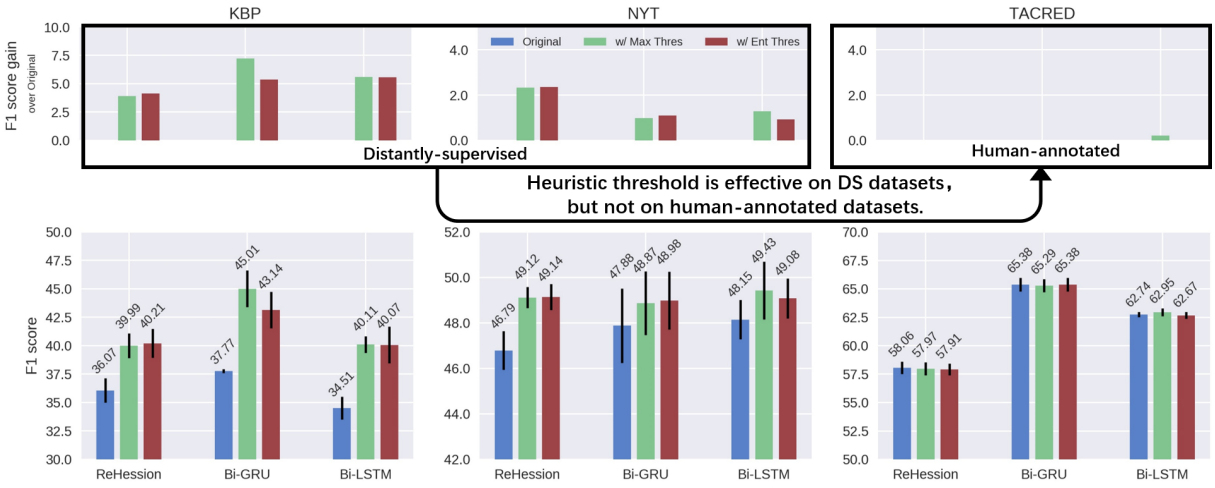


**Figure 3: F1 scores on three datasets when (a) no threshold (original) (b) max threshold (c) entropy threshold are applied, respectively.** A clear boost is observed on distantly supervised datasets (*i.e.*KBP and NYT) after applying threshold; however, performance n human-annotated dataset (*i.e.*TACRED) remains almost the same with thresholding. Heuristic threshold techniques may capture some important but overlooked problems in distantly supervised relation extraction.

### 4.1 Shifted Label Distribution

Shifted label distribution refers to the problem that the label distribution of train set does not align with the test set. This problem is related to but different from "learning from imbalanced data", where the data distribution is imbalanced, but consistent across train and test set. Admittedly, one relation may appear more or less than another in natural language, creating distribution skews; however, this problem widely occurs in both supervised and distantly supervised settings, and is not our focus in this paper.

Our focus is the *label distribution difference between train and test set*. This problem is critical to distantly supervised relation extraction, where the train set is annotated with distant supervision and the test set is manually annotated. As previously mentioned in Section 3.2, distant supervision differs from human annotations by introducing "false positive" and "false negative" labels. The label distribution of train set is subject to existing entries in KBs, and thus there exists a gap between label distributions of train and test set.

We visualize the distribution of KBP (DS dataset) and a truncated 6-class version of TACRED (human-annotated dataset) in Fig 1. Also, proportions of instances categorized into $\delta = |p(r_i|\mathcal{D}_{train}) - p(r_i|\mathcal{D}_{test})|$ bins are shown. It is observed that KBP and NYT both have shifted label distributions (most instances fall into $\delta > 0.05$ bins); while TACRED has consistent label distributions (all instances fall into $\delta < 0.05$ bins).

### 4.2 Impact of Shifted Label Distribution

In order to quantitatively study the impact of label distribution shift, we construct synthetic datasets by sub-sampling instances from the human-annotated TACRED dataset. In this way, the only variable is the label distribution of synthetic datasets, and the impact of other factors such as label noise is excluded.

**Synthetic Dataset Generation.** We create five synthetic train sets, by sampling sentence-label pairs from original TACRED train set with label distributions S1-S5. S5 is a randomly generated label distribution (see Fig 7 in Appendix B). S0 is TACRED's original train set label distribution. S1 – S4 are created with linear interpolation between S0 and S5, *i.e.*, $\text{Si} = \frac{5-i}{5}\text{S0} + \frac{i}{5}\text{S5}$. We apply disproportionate stratification to control the label distribution of synthetic datasets. In this way, we ensure that the number of sampled instances in each synthetic train set is kept constant within $10000 \pm 3$, and the label distribution of each set satisfies S1-S5 respectively.

**Results and Discussion.** We conduct experiments with three typical models (*i.e.*, Bi-GRU, Bi-LSTM and ReHession) and summarize their results in Fig 4. We observe that, from S1 to S5, the performance of all models consistently drops. This phenomenon verifies that shifted label distribution is making a negative influence on model performance. The negative effect expands as the train set label distribution becomes more twisted.

At the same time, we observe that feature-based ReHession is more robust to such shift. The gap

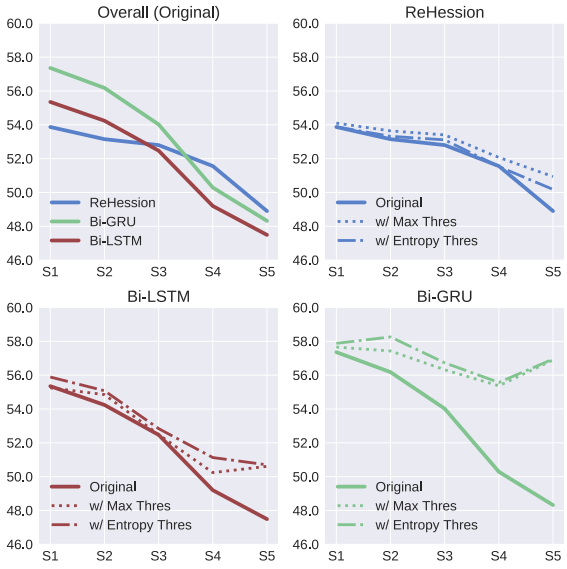


**Figure 4: F1 scores on synthesized datasets S1-S5.** We observe that (a) performance consistently drops from S1 to S5, demonstrating the impact of shifted label distributions; (b) ReHession is more robust to such distribution shift, outperforming Bi-LSTM and Bi-GRU on S4 and S5; (c) threshold is an effective way to handle such shift.

between ReHession and Bi-GRU stably decreases, and eventually ReHession starts to outperform the other two at S4. This could be the reason accounting for "diminishing" phenomenon — neural models such as Bi-GRU is supposed to outperform ReHession by a huge gap (as with S1); however on distantly supervised datasets, shifted label distribution seriously interfere the performance (as with S4 and S5), and therefore it appears as the performance gap diminishes.

**Applying Threshold Techniques.** We also applied the two threshold techniques on the synthetic datasets and summarize their performance in Fig 4. The three models become more robust to the label distribution shift when threshold is applied. Threshold techniques are consistently making improvements, and the improvements are more significant with S5 (most shifted) than with S0 (least shifted). This observation verifies that the underlying mechanism of threshold techniques help the model better handle label distribution shift.

## 5 Bias Adjustment: An Adaptation Method for Label Distribution Shift

Investigating the probabilistic nature of softmax classifier, we present a principled domain adaptation approach, *bias adjustment* (BA), to deal with label distribution shift. This approach explicitly fits the model along such shift by adjusting the bias term in softmax classifier.

### 5.1 Bias Adjustment

We view corpora with different label distributions as different domains. Denoting distantly supervised corpus (train set) as $\mathcal{D}_d$ and human-annotated corpus (test set) as $\mathcal{D}_m$, our task becomes to calculate $p(y = r_i|\mathbf{h}, \mathcal{D}_m)$ based on $p(y = r_i|\mathbf{h}, \mathcal{D}_d)$.

We assume the only difference between $p(y = r_i|\mathbf{h}, \mathcal{D}_m)$ and $p(y = r_i|\mathbf{h}, \mathcal{D}_d)$ to be the label distribution, and the semantic meaning of each label is unchanged. Accordingly, we assume $p(\mathbf{h}|r_i)$ is universal, *i.e.*,

$$p(\mathbf{h}|r_i, \mathcal{D}_m) = p(\mathbf{h}|r_i, \mathcal{D}_d) = p(\mathbf{h}|r_i). \quad (2)$$

As distantly supervised relation extraction models are trained with $\mathcal{D}_d$, its prediction in Equation 1 can be viewed as $p(y = r_i|\mathbf{h}, \mathcal{D}_d)$, *i.e.*,

$$\begin{aligned} p(y = r_i|\mathbf{h}, \mathcal{D}_d) &= p(y = r_i|\mathbf{h}) \\ &= \frac{\exp(\mathbf{r}_i^T\mathbf{h} + b_i)}{\sum_{r_j} \exp(\mathbf{r}_j^T\mathbf{h} + b_j)}. \end{aligned} \quad (3)$$

Based on the Bayes Theorem, we have:

$$p(y = r_i|\mathbf{h}, \mathcal{D}_m) = \frac{p(\mathbf{h}|r_i, \mathcal{D}_m)p(r_i|\mathcal{D}_m)}{\sum_{r_j} p(\mathbf{h}|r_j, \mathcal{D}_m)p(r_j|\mathcal{D}_m)}. \quad (4)$$

Based on the definition of conditional probability and Equation 2, we have:

$$\begin{aligned} p(\mathbf{h}|r_i, \mathcal{D}_m) &= p(\mathbf{h}|r_i) = p(\mathbf{h}|r_i, \mathcal{D}_d) \\ &= \frac{p(\mathbf{h}, r_i, \mathcal{D}_d)}{p(r_i, \mathcal{D}_d)} \\ &= \frac{p(r_i|\mathbf{h}, \mathcal{D}_d) \cdot p(\mathbf{h}|\mathcal{D}_d)}{p(r_i|\mathcal{D}_d)}. \end{aligned} \quad (5)$$

With Equation 3, 4 and 5, we can derive that

$$\begin{aligned} & p(y = r_i|\mathbf{h}, \mathcal{D}_m) \\ =& \frac{p(y = r_i|\mathbf{h}, \mathcal{D}_d)\frac{p(r_i|\mathcal{D}_m)}{p(r_i|\mathcal{D}_d)}}{\sum_{r_j} p(y = r_j|\mathbf{h}, \mathcal{D}_d)\frac{p(r_j|\mathcal{D}_m)}{p(r_j|\mathcal{D}_d)}} \\ =& \frac{\exp(\mathbf{r}_i^T\mathbf{h} + b_i')}{\sum_j \exp(\mathbf{r}_j^T\mathbf{h} + b_j')}, \end{aligned} \quad (6)$$

where

$$b_i' = b_i + \ln p(r_i|\mathcal{D}_m) - \ln p(r_i|\mathcal{D}_d). \quad (7)$$

| Dataset | Model | Original | Max-thres | Δ | Ent-thres | Δ | BA-Set | Δ | BA-Fix | Δ |
|---|---|---|---|---|---|---|---|---|---|---|
| KBP | ReHession | 36.07 ± 1.06 | 39.99 ± 1.09 | 3.92 | 40.21 ± 1.28 | 4.14 | 37.18 ± 1.59 | 1.11 | 37.86 ± 1.64 | 1.79 |
| | Bi-GRU | 37.77 ± 0.18 | 45.01 ± 1.61 | 7.24 | 43.14 ± 1.59 | 5.37 | 40.63 ± 1.79 | 2.86 | 42.28 ± 2.05 | 4.51 |
| | Bi-LSTM | 34.51 ± 0.99 | 40.11 ± 0.73 | 5.60 | 40.07 ± 1.61 | 5.56 | 37.43 ± 1.72 | 2.92 | 38.28 ± 1.66 | 3.77 |
| | PA-LSTM | 37.28 ± 0.81 | 44.04 ± 1.09 | 6.76 | 43.17 ± 1.25 | 5.89 | 40.47 ± 2.55 | 3.19 | 42.44 ± 1.32 | 5.16 |
| | CNN | 30.53 ± 2.26 | 31.23 ± 1.68 | 0.70 | 31.39 ± 2.43 | 0.86 | 30.73 ± 3.96 | 0.20 | 36.12 ± 2.65 | 5.59 |
| | PCNN | 31.58 ± 0.35 | 32.98 ± 0.89 | 1.40 | 32.19 ± 0.76 | 0.61 | 32.79 ± 1.95 | 1.21 | 38.78 ± 2.63 | 7.20 |
| NYT | ReHession | 46.79 ± 0.75 | 49.12 ± 0.47 | 2.33 | 49.14 ± 0.57 | 2.35 | 48.50 ± 1.23 | 1.71 | 48.14 ± 1.08 | 1.35 |
| | Bi-GRU | 47.88 ± 0.85 | 48.87 ± 1.40 | 0.99 | 48.98 ± 1.27 | 1.10 | 48.40 ± 1.52 | 0.52 | 48.74 ± 1.94 | 0.86 |
| | Bi-LSTM | 48.15 ± 0.87 | 49.43 ± 1.27 | 1.28 | 49.08 ± 0.88 | 0.93 | 49.72 ± 1.24 | 1.57 | 49.90 ± 1.44 | 1.75 |
| | PA-LSTM | 46.33 ± 0.64 | 46.70 ± 1.43 | 0.37 | 48.65 ± 1.24 | 2.32 | 47.77 ± 1.43 | 1.44 | 48.65 ± 1.24 | 2.32 |
| | CNN | 46.75 ± 2.79 | 47.87 ± 1.89 | 1.12 | 46.54 ± 2.70 | -0.21 | 48.83 ± 1.66 | 2.08 | 48.17 ± 1.88 | 1.42 |
| | PCNN | 44.63 ± 2.70 | 48.31 ± 0.40 | 3.68 | 47.04 ± 2.60 | 2.41 | 49.08 ± 0.88 | 4.45 | 48.72 ± 1.72 | 4.09 |
| TACRED | ReHession | 58.06 ± 0.54 | 57.97 ± 0.57 | -0.09 | 57.91 ± 0.52 | -0.15 | 58.59 ± 0.66 | 0.53 | 58.61 ± 0.99 | 0.55 |
| | Bi-GRU | 65.38 ± 0.60 | 65.29 ± 0.58 | 0.01 | 65.38 ± 0.60 | 0.10 | 63.72 ± 0.64 | -1.56 | 64.70 ± 0.46 | -0.68 |
| | Bi-LSTM | 62.74 ± 0.23 | 62.95 ± 0.35 | 0.21 | 62.67 ± 0.29 | -0.07 | 62.73 ± 0.60 | -0.01 | 63.44 ± 0.54 | 0.70 |

**Table 3: F1 score of RE Models with Threshold and Bias Adaptation.** 5-time average and standard deviation of F1 scores are reported. Δ denotes the F1 improvement over original. On DS datasets, the four methods targeting label distribution shift achieve consistent performance improvement, with averagely 3.83 F1 improvement on KBP and 1.72 on NYT. However, the same four methods fail to improve performance on human-annotated TACRED.

With this derivation we now know that, under certain assumptions (Equation 2), we can adjust the prediction to fit a target label distribution given $p(r_i|\mathcal{D}_d)$ and $p(r_i|\mathcal{D}_m)$. Accordingly, we use Equation 6 and 7 to calculate the adjusted prediction as:

$$r^* = \arg\max_{r_i} \exp(\mathbf{r}_i^T \mathbf{h} + b'_i). \quad (8)$$

**Label distribution estimation.** The source domain (train) label distribution, $p(r_i|\mathcal{D}_d)$ can be easily estimated on train set. As for target domain (test) distribution $p(r_i|\mathcal{D}_m)$, we use maximum likelihood estimation on a held-out *clean dev* set, which is a 20% sample from test set. This setting is similar to heuristic threshold techniques.

**Implementation details.** The bias adjustment model are implemented in two ways:

- **BA-Set** directly replaces the bias term in Equation 1 with $b'_i$ in Equation 7 during evaluation. No modification to model training is required.
- **BA-Fix** fixes the bias term in Equation 1 as $b_i = \ln p(r_i|\mathcal{D}_d)$ during training and replaces it with $b'_i = \ln p(r_i|\mathcal{D}_m)$ during evaluation. Intuitively, BA-Fix would encourage the model to fit our assumption better (Equation 2); still, it needs special handling during model training, which is a minor disadvantage of BA-Fix compared with BA-Set.

## 5.2 Results and Discussion

We conduct experiments to explore the effectiveness of BA-Set and BA-Fix and summarize their performance in Table 3. Also, F1 improvements

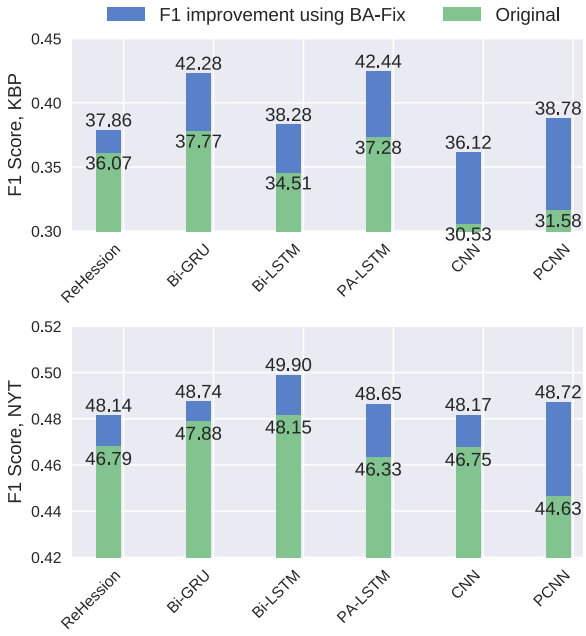


**Figure 5: F1 Improvement using BA-Fix.** BA-Fix consistently improves performance in compared models.

when BA-Fix is used is shown in Fig 5. We find that these two technologies bring consistent improvements to all RE models on both distantly supervised datasets. Especially, in the case of PCNN on KBP dataset, a 23% relative F1 improvement is observed. At the same time, the same technology fails to achieve performance improvements on TACRED, the human annotated dataset. In other words, we found that, by explicitly adapting the model along label distribution shift, consistent improvements can be achieved on distant supervision but not on human annotations. This observation again supports our assumption that shifted label distribution is an unique factor for distantly super-

vised RE that needs special handling.

**Summary of all four methods.** Compared with heuristic threshold techniques, bias adjustment methods directly address shifted label distribution issue by adjusting bias term in classifier, and is more principled and explainable. Though both threshold and bias adjustment methods require an extra clean dev sampled from test set, threshold techniques require predicting on the instances in the clean dev and tune $T_m$ or $T_e$ based on the output probability. Bias adjustment only needs the label distribution estimated on clean dev, and is more efficient in computation. As for performance, there is no clear indication that one method is consistently stronger than another. However, all four methods (2 threshold and 2 bias adjustment) achieve similar results when applied to RE models – we observe relatively significant performance improvements on DS datasets, but only marginal improvements on human-annotated dataset.

**Comments on neural RE models.** On average, bias adjustment methods results in 3.66 F1 improvement on KBP and 2.05 on NYT for neural models. BA-Fix gains a suprising 7.20 with PCNN on KBP. Noting that only bias terms in softmax classifier are modified and only a small piece of extra information is used, it is implied that shifted label distribution is severely hindering model performance and capabilities of state-of-the-art neural models are not fully described with traditional evaluation. Hidden representations $\mathbf{h}$ learned by neural models indeed capture semantic meanings more accurately than feature-based model, while the bias in classifier becomes the major obstacle towards better performance.

## 6 Comparison with Denoising Methods

In this section, we conduct analyses about shifted label distribution and its relation with label noise. We apply a popular label noise reduction method– selective attention (Lin et al., 2016), which groups all sentences with the same entity pair into one bag, conducts multi-instance training and tries to place more weight on high-quality sentences within the bag. This method, along with threshold techniques and bias adjustment introduced in previous sections, is applied to two different models (*i.e.*, PCNN and Bi-GRU).

We summarize their improvements over the original model in Figure 6. We find that selective attention is indeed effective and improves the performance; meanwhile, heuristic threshold and bias adaption approaches boost the performance, and in some cases the boost is even more significant than that of selective attention. This observation is reasonable since both heuristics and bias adaption approaches are able to access additional information from clean dev (20% of test set). Still, it is surprising that such small piece of information brings about huge difference, demonstrating the importance of handling shifted label distribution. It also shows that there exists much space for improving distantly supervised RE models from a shifted label distribution perspective.

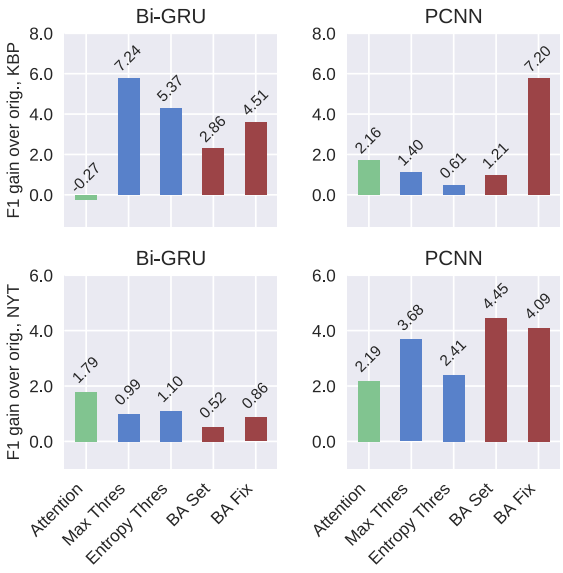


**Figure 6: Comparison among selective attention, threshold heuristics and bias adaption approaches.** Threshold heuristics and bias adaption approaches bring more significant improvements in some cases, indicating that shifted label distribution is a non-negligible problem.

## 7 Related Work

### 7.1 Relation Extraction

Relation extraction is to identify the relationship between a pair of entity mentions. The task setting slightly varies as the relation can be either extracted from a bag of sentences (corpus-level) or one single sentence (sentence-level). In this paper, we focus on sentence-level RE. That is, prediction should be purely based on the information provided within the sentence, instead of external knowledge or commonsense.

Recent approaches follow the supervised learning paradigm and rely on considerable amounts of labeled instances to train effective models. Zeng et al. (2014) proposed using CNN for relation ex-

traction, which could automatically capture features from texts. Zeng et al. (2015) further extended it with piecewise max-pooling, *i.e.*, splitting the sentence into three pieces with the object and subject entity, doing max-pooling over the three pieces separately, and finally concatenating the hidden representations. Lin et al. (2016) applied selective attention over sentences for learning from multiple instances. This method was originally designed for corpus-level setting. It organizes sentences into bags and assign lower weights to those less relevant sentences in the bag. Zhang et al. (2017) proposed position-aware LSTM network which incorporates entity position information into encoding and enables attention mechanism to simultaneously exploit semantic and positional information.

### 7.2 Distant Supervision

In supervised relation extraction paradigm, one longstanding bottleneck is the lack of large-scale labeled training data. In order to alleviate the dependency on human supervision, Mintz et al. (2009) proposed distant supervision, namely constructing large datasets automatically by aligning text to an existing knowledge bases (*e.g.*, Freebase). Also, distantly supervised relation extraction is formulated into a reinforcement learning problem by Feng et al. (2018) for selecting high-quality instances. Similar annotation generation strategy using distant supervision has also been used for other NLP tasks, such as named entity recognition (Shang et al., 2018) and sentiment classification (Go et al., 2009).

Though this strategy lightens annotation burdens, distant supervision inevitably introduces label noises. As the relation types are annotated merely according to entity mentions in the sentence, the local context may be annotated with labels that are not expressed in the sentence, In recent years, researchers mainly focus on dealing with label noises, and proposed the following methods: Riedel et al. (2010) use multi-instance single-label learning paradigm; Hoffmann et al. (2011); Surdeanu et al. (2012) propose multi-instance multi-label learning paradigm. Recently, with the advance of neural network techniques, deep learning methods (Zeng et al., 2015; Lin et al., 2016) are applied to distantly supervised datasets, with powerful automatic feature extraction and advanced label noised reducing techniques such as selective attention. Liu et al. (2017) proposed a general framework to consolidate heterogeneous information and refine the true labels from noisy labels.

Label noise is certainly an important factor limiting the performance of DS-RE models. Meanwhile, we argue that shifted label distribution is also a performance-limiting aspect. It is long-overlooked and should be handled properly.

## 8 Conclusion and Future Work

In this paper, we first present the observation of inconsistent performance when models are trained with human annotations and distant supervision in the task of relation extraction. It leads us to explore the underlying challenges for distantly supervised relation extraction. Relating two effective threshold techniques to label distribution, we reveal an important yet long-overlooked issue – *shifted label distribution*. The impact of this issue is further demonstrated with experiments on five synthetic train sets. We also consider this issue from a domain adaptation perspective, introducing a theoretically-sound bias adjustment method to recognize and highlight label distribution shift. The bias adjustment methods achieve significant performance improvement on distantly-supervised datasets. All of these findings support our argument that shifted label distribution can severely hinder model performance and should be handled properly in future research.

Based on these observations, we suggest that in addition to label noise, more attention be paid to the shifted label distribution in distantly supervised relation extraction research. We hope that the analysis presented will provide new insights into this long-overlooked factor and encourage future research of creating models robust to label distribution shift. We also hope that methods such as threshold techniques and bias adjustment become useful tools in future research.

## Acknowledgments

This work has been supported in part by National Science Foundation SMA 18-29268, DARPA MCS and GAILA, IARPA BETTER, Schmidt Family Foundation, Amazon Faculty Award, Google Research Award, Snapchat Gift and JP Morgan AI Research Award. We would like to thank all the collaborators in INK research lab for their constructive feedback on the work.

## References


Joe Ellis, Xuansong Li, Kira Griffitt, Stephanie Strassel, and Jonathan Wright. 2012. Linguistic resources for 2013 knowledge base population evaluations. In *TAC*.

Jun Feng, Minlie Huang, Li Zhao, Yang Yang, and Xiaoyan Zhu. 2018. Reinforcement learning for relation classification from noisy data. In *Proceedings of AAAI*.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12):2009.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 541–550. Association for Computational Linguistics.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2124–2133.

Xiao Ling and Daniel S Weld. 2012. Fine-grained entity recognition. In *AAAI*, volume 12, pages 94–100.

Liyuan Liu, Xiang Ren, Qi Zhu, Shi Zhi, Huan Gui, Heng Ji, and Jiawei Han. 2017. Heterogeneous supervision for relation extraction: A representation learning approach. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 46–56.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Gábor Melis, Chris Dyer, and Phil Blunsom. 2017. On the state of the art of evaluation in neural language models. *arXiv preprint arXiv:1707.05589*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Xiang Ren, Zeqiu Wu, Wenqi He, Meng Qu, Clare R Voss, Heng Ji, Tarek F Abdelzaher, and Jiawei Han. 2017. Cotype: Joint extraction of typed entities and relations with knowledge bases. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1015–1024. International World Wide Web Conferences Steering Committee.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer.

Jingbo Shang, Liyuan Liu, Xiaotao Gu, Xiang Ren, Teng Ren, and Jiawei Han. 2018. Learning named entity tagger using domain-specific dictionary. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2054–2064.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, pages 455–465. Association for Computational Linguistics.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1753–1762.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D Manning. 2017. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 35–45.

# Easy First Relation Extraction with Information Redundancy

**Shuai Ma**[1,2] **Gang Wang**[1,2] **Yansong Feng**[3] **Jinpeng Huai**[1,2]
[1]SKLSDE Lab, Beihang University, China
[2] Beijing Advanced Innovation Center for Big Data and Brain Computing, Beijing, China
[3]Peking University, China
{mashuai, iegwang, huaijp}@buaa.edu.cn fengyansong@pku.edu.cn

## Abstract

Many existing relation extraction (RE) models make decisions globally using integer linear programming (ILP). However, it is nontrivial to make use of integer linear programming as a blackbox solver for RE. Its cost of time and memory may become unacceptable with the increase of data scale, and redundant information needs to be encoded cautiously for ILP. In this paper, we propose an easy first approach for relation extraction with information redundancies, embedded in the results produced by local sentence level extractors, during which conflict decisions are resolved with domain and uniqueness constraints. Information redundancies are leveraged to support both easy first collective inference for easy decisions in the first stage and ILP for hard decisions in a subsequent stage. Experimental study shows that our approach improves the efficiency and accuracy of RE, and outperforms both ILP and neural network-based methods.

## 1 Introduction

Relation extraction (RE) has been extensively studied due to its crucial role for many knowledge based applications (Fan et al., 2012; Zhang et al., 2012; Chen et al., 2014a), such as question answering and knowledge graph. There are two types of relation extractors: local and global. The former identify relationships between entity pairs individually according to the local features of sentences, *e.g.,* lexical and syntactic features (Bunescu and Mooney, 2007; Mintz et al., 2009; Surdeanu et al., 2010; Hoffmann et al., 2011; Surdeanu et al., 2012; Zeng et al., 2015; Lin et al., 2016). The latter make decisions by further considering joint features of the entire corpus (Yao et al., 2010; Li et al., 2011, 2013; de Lacalle and Lapata, 2013; Chen et al., 2014a). Global relation extractors are able to resolve conflict decisions, and to utilize the dependencies among extracted facts to improve the performance (Li et al., 2011; Chen et al., 2014a), commonly by formalizing RE as a constrained optimization problem and solving RE with integer linear programming (ILP) (Roth and Yih, 2004; Choi et al., 2006; Roth and Yih, 2007; Li et al., 2011; Ji et al., 2013; Chen et al., 2014a,b). However, global optimization remains dominated by the ILP solvers, suffering from heavy time expenses.

Using integer linear programming for RE as a blackbox solver is a challenging task. First, with the increase of entity pairs and candidate relations, the variables and constraints encoded for ILP increase dramatically, which in return may consume too much computing time and memory. Second, redundant information needs to be encoded cautiously. Consider <United States, *capital : 0.6*, New York>, <United States, *capital : 0.7*, New York> and <United States, *capital : 0.96*, Washington D.C.> in Table 1. Simple statistical methods, such as confidence summation, may easily lead to a wrong decision in this case.

To address the above issues, we propose to make easy (most confident) decisions first, and then to make hard decisions with ILP. The rationale lies in that easy decisions should be made early, and eliminating conflicts with constraints aids hard decision making. We leverage information redundancies, embedded in the results produced by sentence level extractors, to compute the confidences of candidate relations for entity pairs. Redundancy commonly exists in various corpora, *e.g.,* subject "United States" and object "New York" appear multiple times in Table 1. Even if there exists a specific relation between a subject and an object, the relation may not be reflected in certain mentions due to the lack of embedded evidence in single mentions, and *information redundancies are essentially to alleviate such insufficiency for decision making*.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3851–3861, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| ID | Entity Pair | Top-3 Candidate Relations | | |
|---|---|---|---|---|
| 1 | Australia, Canberra | *capital : 0.7* | *locationCity : 0.2* | *nationality : 0.1* |
| 2 | United States, New York | *capital : 0.6* | *birthPlace : 0.35* | *nationality : 0.05* |
| 3 | United States, New York | *capital : 0.7* | *country : 0.23* | *deathPlace : 0.07* |
| 4 | United States, Washington D.C. | *capital : 0.96* | *country : 0.03* | *birthPlace : 0.01* |
| 5 | Australia, Sydney | *locationCountry : 0.6* | *capital : 0.2* | *largestCity : 0.2* |
| 6 | University of Sydney, Sydney | *locationCity : 0.7* | *locationCountry : 0.2* | *city : 0.1* |
| 7 | Geoff Ogilvy, Australia | *nationality : 0.4* | *locationCountry : 0.3* | *city : 0.3* |

Table 1: Example entity pairs and their candidate relations.

When making easy decisions, we keep the consistency among candidate relations by making use of constraints (*i.e.,* domain and uniqueness constraints) to eliminate conflicts directly, instead of implicitly encoding constraints in ILP (Yao et al., 2010; de Lacalle and Lapata, 2013; Chen et al., 2014a; Koch et al., 2014; Chen et al., 2014b). As a result, the number of variables and constraints to be encoded in ILP is significantly reduced, which speeds up the entire decision making process.

In short, our approach employs an easy-first strategy with information redundancies to make most confident decisions first during which conflict candidate relations are resolved directly by domain and uniqueness constraints, and it only makes the remaining hard decisions with ILP. Our approach is an important improvement of ILP-models, and it is not a new RE model, but an efficient and effective approach to fully exploiting the results of (any) local RE extractors. As shown in the experiments, our approach improves the performance, on average (4.58%, 17.90%) more accurate and (21, 37) times faster than existing ILP and neural network-based methods, respectively.

In the rest, we first introduce constraints and redundancies in Section 2, then present our detailed approach in Section 3, followed by experimental study in Section 4, related work in Section 5, and conclusions in Section 6.

## 2 Constraints and Redundancies

Consider a set $M$ of mentions and a set $R$ of predefined relations $\{r_1, \ldots, r_k\}$. For each mention $m$, a sentence level extractor produces an entity pair (subject $s$ and object $o$) and a confidence score $c_i$ for each relation $r_i$ ($i \in [1, k]$), which represents the probability that $s$ and $o$ have relation $r_i$ (Berger et al., 1996; Hoffmann et al., 2011). Here an NA (unknown) relation is typically included in $R$.

A mention $m$ preprocessed by a sentence level extractor is essentially a $k+2$ tuple $(s, o, r_1 : c_1, \ldots, r_k : c_k)$, as illustrated in Table 1, where $1 \geq c_i \geq c_j \geq 0$ for any $i > j \in [1, k]$ and $\sum_{i=1}^{k} c_i = 1$. Observe that an entity pair (subject $s$ and object $o$) with a relation $r \in R$ can be treated as an SRO triple $(s, r, o)$, and a mention $m$ contains $k$ SRO triples. For convenience, given a mention $m$ and a relation $r \in R$, we also denote the score of relation $r$ in $m$ as $m[r].c$.

Given a set $M$ of mentions preprocessed by a sentence level extractor, our task for relation extraction is to determine the set of correct relations for each entity pair in $M$, and our approach adopts an easy-first strategy to make fast and accurate decisions, by leveraging constraints and redundancies that are to be introduced below.

In the following, a mention is referred to the one preprocessed by a sentence level extractor.

### 2.1 Constraints

We consider two types of constraints: domain and uniqueness constraints, commonly used to enforce agreements on decisions for RE (Yao et al., 2010; de Lacalle and Lapata, 2013; Chen et al., 2014a; Koch et al., 2014; Chen et al., 2014b).

**Domain constraints**. This type of constrains enforces constraints among the subject and object domains of relations. Given relations $r_i$ and $r_j$ ($1 \leq i, j \leq k$), (1) an *S-S domain constraint* ensures that $r_i$ and $r_j$ share no common entities between their subjects, *i.e.,* $r_i.subject \cap r_j.subject = \emptyset$, (2) an *O-O domain constraint* ensures that $r_i$ and $r_j$ share no common entities between their objects, *i.e.,* $r_i.object \cap r_j.object = \emptyset$, and (3) an *S-O domain constraint* ensures that $r_i$ and $r_j$ share no common entities between the subject of $r_i$ and the object of $r_j$, *i.e.,* $r_i.subject \cap r_j.object = \emptyset$.

For example, (1) relations *largestCity* and *locationCity* have their subjects as countries (*e.g.,* "Australia") and organizations (*e.g.,* "University of Sydney"), respectively. They hold an *S-S* domain constraint; (2) relations *locationCity* and *locationCountry* have their objects as cities and countries, respectively. They hold an *O-O* domain constraint; (3) *locationCity* has its subjects as or-

| # of mentions | # of entity pairs | # of subjects | # of objects | # / ratio of entity pairs mentioned multiple times | | # / ratio of subjects mentioned multiple times | | # / ratio of objects mentioned multiple times | | Top 3 relation precision (%) |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | $\geq 3$ | $\geq 5$ | $\geq 3$ | $\geq 5$ | $\geq 3$ | $\geq 5$ | |
| 53162 | 30864 | 11360 | 9709 | 2754/ 8.92% | 1225/ 3.97% | 2914/ 25.65% | 1591/ 14.01% | 2442/ 25.15% | 1351/ 13.91% | DB_me: 77.3/ 11.8/ 5.0<br>DB_nn: 76.5/ 10.6/ 5.0 |

Table 2: Redundancy statistics of DBpedia - ( see Section 4 for details about the datasets).

ganizations and *nationality* has its objects as countries. It holds an *S-O* domain constraint.

**Uniqueness constraints**. This type of constrains enforces the uniqueness of the subjects and objects of relations. Given a relation $r_i$ $(1 \leq i \leq k)$, (1) an *S uniqueness constraint* ensures that there exist no subject $s$ and objects $o_h \neq o_l$ such that both $(s, r_i, o_h)$ and $(s, r_i, o_l)$ hold, and (2) an *O uniqueness constraint* ensures that there exist no object $o$ and subjects $s_h \neq s_l$ such that both $(s_h, r_i, o)$ and $(s_l, r_i, o)$ hold, respectively.

For example, relation *capital* holds both *S* and *O* uniqueness constraints, since a country as the subject of *capital* has only one capital, and a city as the object of *capital* can only be the capital of one country.

We refer to the total set of relations with *S-S* (resp. *O-O* and *S-O*) domain constraints as $DC_{ss}$ (resp. $DC_{oo}$ and $DC_{so}$). We also refer to the total set of relations with *S* (resp. *O*) uniqueness constraints as $UC_s$ (resp. $UC_o$).

## 2.2 Information Redundancies

Redundancies are used to pick up hidden information (Downey et al., 2005; Banko et al., 2007; Li et al., 2011), and are very common in the corpus, as revealed by the statistics in Table 2. They are essentially the statistical characteristics (knowledge) of the results produced by local sentence level extractors, and are very important for global predictions. In this study, we introduce and leverage four classes of information redundancies: S-O, S-R, O-R and R redundancies.

**S-O redundancies** are introduced to aid the decision making of the top-one relations of mentions with the same subjects and objects. For a mention $m = (s, o, r_1 : c_1, \ldots, r_k : c_k)$ preprocessed by a sentence level extractor, its certainty degree $\mathsf{ent}(\mathsf{m})$ is defined as follows.

$$\mathsf{ent}(\mathsf{m}) = -\sum_{i=1}^{k} c_i \ln c_i. \quad (1)$$

The redundancy score $\mathsf{RC}(\mathsf{s}, \mathsf{r}_1, \mathsf{o})$ of the top-one relation $r_1$ for subject $s$ and object $o$ in $m$, based on its S-O redundancies, is defined as

$$\mathsf{RC}(\mathsf{s}, \mathsf{r}_1, \mathsf{o}) = \sum_{m' \in M_{s,o,r_1}} \alpha^{\mathsf{ent}(\mathsf{m}')}, \quad (2)$$

where $M_{s,o,r_1}$ is the set of mentions in $M$ whose subjects are $s$, objects are $o$, and top-one relations are $r_1$, and $\alpha$ is a small positive number in $(0, 1)$ (set to 0.05 by default) to enforce that $\alpha^{\mathsf{ent}(\mathsf{m}')}$ falls into $(0, 1)$. Informally, $\mathsf{RC}(\mathsf{s}, \mathsf{r}_1, \mathsf{o})$ is a collective score based on its S-O redundancies, which makes use of *information entropy* to judge the confidence, and considers both the relative confidence scores and repeated times.

**S-R redundancies** are introduced to aid the decision making whether a subject meets the domain requirement of a relation. The likelihood score $\mathsf{LC}(\mathsf{s}, \mathsf{r})$ for subject $s$ and relation $r$, based on its S-R redundancies, is

$$\mathsf{LC}(\mathsf{s}, \mathsf{r}) = \frac{\sum_{m \in M_s} m[r].c}{\sum_{m \in M_s} \sum_{r' \in R \setminus \{NA\}} m[r'].c}, \quad (3)$$

where $M_s$ is the set of mentions with the same subject $s$. Informally, $\mathsf{LC}(\mathsf{s}, \mathsf{r})$ measures the probability that relation $r$ has a subject $s$ among all relations except NA.

**O-R redundancies** are introduced to aid the decision making whether an object meets the domain requirement of a relation. The likelihood score $\mathsf{LC}(\mathsf{o}, \mathsf{r})$ for object $o$ and relation $r$, based on its O-R redundancies, is

$$\mathsf{LC}(\mathsf{o}, \mathsf{r}) = \frac{\sum_{m \in M_o} m[r].c}{\sum_{m \in M_o} \sum_{r' \in R \setminus \{NA\}} m[r'].c}, \quad (4)$$

where $M_o$ is the set of mentions with the same object $o$. Informally, $\mathsf{LC}(\mathsf{o}, \mathsf{r})$ measures the probability that relation $r$ has an object $o$ among all relations except NA.

**R redundancies** are introduced to aid the decision making whether a subject and an object have a non-NA relation. The likelihood score $\mathsf{LC}(\mathsf{s}, \mathsf{o})$ for subject $s$ and object $o$, based on its R redundancies, is

$$\mathsf{LC}(\mathsf{s}, \mathsf{o}) = \max_{m \in M_{s,o}} \sum_{r \in R \setminus \{NA\}} m[r].c, \quad (5)$$

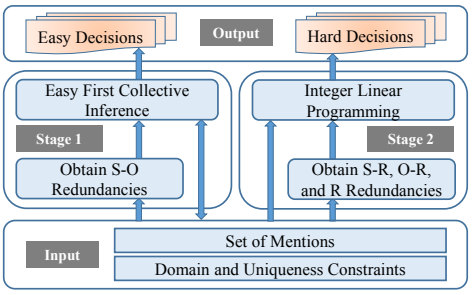


Figure 1: Framework of our approach eFIRE.

*Input:* a set $M$ of mentions, a set of domain and uniqueness constraints, and a threshold $\epsilon$.
*Output:* a set $E$ of decision and updated $M$.
1. $E := \emptyset$;
2. Compute redundancy scores and sort with a max-heap MH;
3. **while** MH.Max() $> \epsilon$ **do**
4. $\quad (s, r, o, c) :=$ MH.popMax();
5. $\quad E := E \cup \{(s, r, o, c)\}$ ;
6. $\quad$ Conflict resolving by updating MH and $M$;
7. **return** $E$, $M$.

Figure 2: Algorithm for making easy decisions.

where $M_{s,o}$ is the set of mentions with the same subject $s$ and object $o$. Informally, $\mathsf{LC(s, o)}$ selects prominent information from local decisions to measure the probability of having at least one non-NA relation between $s$ and $o$.

As will be seen in the next section, redundancy scores $\mathsf{RC(s, r_1, o)}$ are used for making easy decisions, and likelihood scores $\mathsf{LC(s, r)}$, $\mathsf{LC(o, r)}$ and $\mathsf{LC(s, o)}$ are used for hard decisions, respectively.

## 3 Easy First Relation Extraction

We propose a novel easy FIrst approach for Relation Extraction with information redundancies, referred to as eFIRE. As shown by the framework in Figure 1, our approach obtains S-O redundancies, and makes easy decisions with easy first collective inference in the first stage. Then it derives S-R, O-R and R redundancies, and makes hard decisions with ILP in the second stage. We next introduce our approach in detail.

### 3.1 Easy First Collective Inference

In the first stage, decisions must be highly accurate to avoid error propagation. As revealed by Table 2, the decisions produced by local extractors are only reliable for top-one relations. Hence, eFIRE first makes decisions for entity pairs using their top-one relations only. The confidences are the redundancy scores obtained with the S-O redundancies (Section 2.2). Once a decision is made, disagreements are resolved immediately with domain and uniqueness constraints directly (Section 2.1).

**Confidence computing and ordering**. It computes the redundancy scores $\mathsf{RC(s, r_1, o)}$ of all the entity pairs $(s, o)$ with their top-one relations $r_1$ in mentions $M$, using S-O redundancies. The higher the redundancy scores $\mathsf{RC(s, r_1, o)}$ are, the more confident subject $s$ and object $o$ have a relation $r_1$. Hence, entity pairs with their top-one relations are sorted in a descending order of their redundancy scores. As updating operations happen very often during the process of decision making, we introduce a max-heap MH to maintain all the entity pairs with their top-one relations and redundancy scores for the sake of efficiency.

**Decision making and conflict resolving**. It first makes a decision for the entity pair $(s, o)$ by choosing its top-one relation $r_1$ such that the redundancy score $\mathsf{RC(s, r_1, o)}$ is the highest in MH.

Then it resolves conflicts accordingly. (1) For any relation $r \in R$ having an S-S domain constraint with $r_1$, all entity pairs with subject $s$ and top-one relation $r$ are deleted from MH, and for entity pairs with subject $s$ in $M$, their scores of relation $r$ are set to zeroes. It is similar for O-O and S-O domain constraints. (2) If relation $r_1$ has an S uniqueness constraint, all entity pairs with subject $s$ and top-one relation $r_1$ are deleted from MH, and for entity pairs with subject $s$ and object $o' \neq o$ in $M$, their scores of relation $r_1$ are set to zeroes. It is similar for O uniqueness constraints.

The above process is repeated until the highest redundancy score in the max-heap MH is less than a predefined threshold $\epsilon$, which is to ensure the correctness of decisions. For the benefit of efficiency, we also index mentions by subjects, by objects and by entity pairs, separately.

The intention of threshold $\epsilon$ is to distinguish easy decisions from hard ones based on S-O redundancies. This indeed can be reflected from the redundancy scores. Consider an extreme case when there is only one mention $m$ in $M$ with subject $s$, object $o$ and top-one relation $r_1$ whose score is 1.0, *i.e.,* there are no S-O redundancies for mention $m$. In this case, we have $\mathsf{RC(s, r_1, o)} = 1.0$. However, for cases when multiple mentions with the same subject $s$, object $o$ and top-one relation $r_1$, it is likely that the redundancy score is less than 1.0. Hence, threshold $\epsilon$ is typically set to a value a

little less than 1.0.

Our approach makes a better use of information redundancies in the corpus to aid the decision making process. Recall the example on determining whether the capital of "United States" is "New York" or "Washington D.C." in Section 1. With the S-O redundancies, "Washington D.C." is chosen as the capital of "United States", as $\mathsf{RC}(\mathsf{United\ States}, \textit{capital}, \mathsf{New\ York}) = 0.18$ and $\mathsf{RC}(\mathsf{United\ States}, \textit{capital}, \mathsf{Washington\ D.C.}) = 0.57$, which justifies the rationale of introducing the certainty degree $\mathsf{ent}(\mathsf{m})$ for mentions $m$.

**Complete algorithm for easy first inference**. The complete algorithm is presented in Figure 2.

It first initializes the set $E$ of easy decisions empty (line 1). It then computes the redundancy scores of all the entity pairs with their top-one relations in mentions $M$, using S-O redundancies, and these entity pairs with their top-one relations are sorted in a descending order of their redundancy scores with a max heap MH (line 2). It repeatedly deals with entity pairs and their top-one relations in MH one by one until the highest redundancy score in MH is no more than $\epsilon$ (lines 3-6). Once a decision is made (lines 4, 5), conflicts are resolved immediately by updating MH and $M$ (line 6). Finally, the modified mentions $M$ and a set $E$ of easy decisions are returned (line 7).

**Time and space complexity analyses**. The algorithm runs in $O(|M|^2(|R| + \log|M|))$ time, where $|M|$ and $|R|$ are the numbers of mentions and predefined relations, respectively. Observe the following. For a mention, (1) it takes $O(|R| + \log|M|)$ time to compute the redundancy scores for all entity pairs with their top-one relations, (2) maintaining MH can be done in $O(\log|M|)$ time, and (3) decision making and conflict resolving take $O(|M|(|R|+\log|M|))$ time. As there are in total $|M|$ mentions, we have the conclusion.

It is easy to see that the algorithm takes a space in the linear size of the set $M$ of mentions.

### 3.2 Integer Linear Programming

In the second stage, our goal is to find an optimal configuration for the remaining mentions, making use of S-R, O-R and R redundancies, solving the disagreements by domain and uniqueness constraints, and maximizing the overall confidence of the made decisions. This is an NP-hard optimization problem, and many optimization models can be used to obtain approximate solutions. The tricky part is the design of the objective functions. Here, we adopt the ILP tool "IBM ILOG CPLEX".

More specifically, for each mention $m$ and each of its candidate relations $r$ in the set $M$ of remaining mentions returned in the first stage, we define a binary decision variable $v_m^r$ indicating whether relation $r$ is chosen for the entity pair $(s, o)$ in $m$ by the solver. For each mention $m$ in $M$, we choose its top three relations with scores no less than $0.1$ as the candidate relations. As revealed by Table 2, candidates beyond top-3 are very unreliable.

Our objective is to maximize the total confidence of all the selected candidates based on the S-R, O-R and R redundancies, and the objective function can be written as:

$$\max \sum_{m\in M, r\in R^m} \big(\mathsf{LC}(\mathsf{s},\mathsf{r})+\mathsf{LC}(\mathsf{o},\mathsf{r})+\mathsf{LC}(\mathsf{s},\mathsf{o}) + \sum_{m'\in M_m} m'[r].c\big) v_m^r, \tag{6}$$

where $s$ and $o$ are the subject and object in $m$, $R^m$ is the set of candidate relations for $s$ and $o$ in $m$, and $M_m$ is the set of mentions in $M$ having the same subject and object as $m$. The first component is the sum of S-R, O-R, and R redundancies of the selected candidates, and the second one is the sum of the original confidence scores of the selected candidates. The former is designed to encourage the model to select candidates meeting the domain requirements of relations, and the latter is designed to give consideration to decisions produced by sentence level extractors. That is, although the sentence level extractors may make wrong decisions, the global statistics of their decisions are reliable, and should be preserved.

The domain and uniqueness constraints are encoded to avoid conflict decisions as follows.

$$v_{m_i}^{r^{m_i}} + v_{m_j}^{r^{m_j}} \le 1, \tag{7}$$

where $m_i$ and $m_j$ have the same subject, $r^{m_i} \in R^{m_i}$, $r^{m_j} \in R^{m_j}$, and $r^{m_i}$ and $r^{m_j}$ have an S-S domain constraint in $DC_{ss}$.

$$v_{m_i}^{r^{m_i}} + v_{m_j}^{r^{m_j}} \le 1, \tag{8}$$

where $m_i$ and $m_j$ have the same object, $r^{m_i} \in R^{m_i}$, $r^{m_j} \in R^{m_j}$, and $r^{m_i}$ and $r^{m_j}$ have an O-O domain constraint in $DC_{oo}$.

$$v_{m_i}^{r^{m_i}} + v_{m_j}^{r^{m_j}} \le 1, \tag{9}$$

| Datasets | Methods | Running time(s) | # of ILP variables | # of ILP constraints |
|---|---|---|---|---|
| DB_me | baseline | 52.70 | 12353 | 293361 |
| | eFIRE | **1.88** | **8185** | **34931** |
| | eFIRE-1S | 1.90 | 8185 | 34931 |
| | eFIRE-2S | 52.35 | 12353 | 293361 |
| | CNN+ATT | 69.49 | - | - |
| | PCNN+ATT | 34.55 | - | - |
| DB_nn | baseline | 15.82 | 11770 | 94498 |
| | eFIRE | **1.11** | **9678** | **24314** |
| | eFIRE-1S | 1.20 | 9678 | 24314 |
| | eFIRE-2S | 15.81 | 11770 | 94498 |
| | CNN+ATT | 69.49 | - | - |
| | PCNN+ATT | 34.55 | - | - |

Table 3: Efficiency of eFIRE and comparison methods.

where the subject of $m_i$ is the object of $m_j$, $r^{m_i} \in R^{m_i}$, $r^{m_j} \in R^{m_j}$, and $r^{m_i}$ and $r^{m_j}$ have an S-O domain constraint in $DC_{so}$.

$$\sum_{m \in M_{r,s}} v_m^r \leq 1, \tag{10}$$

where $M_{r,s}$ is the set of mentions with candidate relation $r$, subject $s$ and pairwise distinct objects, and $r$ has an S uniqueness constraint in $UC_s$.

$$\sum_{m \in M_{r,o}} v_m^r \leq 1, \tag{11}$$

where $M_{r,o}$ is the set of mentions with candidate relation $r$, object $o$ and pairwise distinct subjects, and $r$ has an O uniqueness constraint in $UC_o$.

By adopting ILP, eFIRE combines the scores refined in the first stage and the constraints to make hard decisions. After the optimization problem is solved, together with the easy decisions obtained in the first stage, eFIRE finally produces a list of selected candidate relations for each entity pair.

## 4 Experimental Study

In this section, we present an extensive experimental study of our easy first approach eFIRE.

### 4.1 Experimental Settings

We first present our experimental settings.

**Datasets**. The two datasets, DB_me and DB_nn, stem from DBpedia (Bizer et al., 2009), mapping the triples in DBpedia to sentences in the New York Time corpus. We map 51 different relations to the dataset. We use both a maximum entropy model MaxEnt (Chen et al., 2014a) and neural network model NN (Lin et al., 2016) as the sentence level extractors to output confidence scores, denoted as DB_me and DB_nn, respectively. The features of these two datasets are reported in Table 2. There are 53162 mentions in each dataset, including 38654 mentions with NA relations. We learn domain and uniqueness constraints from Freebase as knowledge base (KB).

**Algorithms for comparison**. To evaluate our approach, we compare with three methods: the ILP based global method for RE in (Chen et al., 2014a) as baseline that use the global clues to help resolve the disagreements, and CNN+ATT and PCNN+ATT in (Lin et al., 2016) that are neural network-based methods with attention mechanism to use all informative sentences.

**Implementation**. All algorithms were implemented with C++. All experiments were run on a PC with 2 Intel(R) Xeon(R) E5ÍC2640 2.6GHz CPUs and 64 GB of memory. When running time is measured, the test was repeated over 5 times and the average is reported.

### 4.2 Experimental Results

We next present our findings of the effectiveness and efficiency of our easy first approach eFIRE. Following previous work, we also use the precision in the low recall portion of the P-R curve as the effectiveness criterion.

**Exp-1: Overall performance**. In the first set of tests, we evaluated the effectiveness and efficiency, and the results are reported in Figures 3(a), 3(b) and Table 3, respectively.

Our approach eFIRE outperforms the methods for comparison. eFIRE is on average (4.80%, 4.36%), (17.99%, 28.10%) and (7.69%, 17.82%) more accurate than baseline, CNN+ATT and PCNN+ATT on (DB_me, DB_nn) in the low-recall portion $[0, 0.25]$ of the P-R curves, respectively. eFIRE consistently outperforms CNN+ATT and PCNN+ATT over the entire range of recall. While baseline tends to have results with a higher recall, it has a weakness of low precision, which is alleviated by eFIRE that is able to obtain more correct decisions. It is difficult to guarantee high precision and recall at the same time. For most KB population applications, only the high precision part is considered for the effectiveness evaluation. It is worth pointing out that we only compare the testing time of CNN+ATT and PCNN+ATT here.

Our method eFIRE also reduces the running time on all datasets. eFIRE is on average (28, 14), (37, 63) and (18, 31) times faster than baseline, CNN+ATT and PCNN+ATT on (DB_me, DB_nn), respectively. This is because the easy-first strategy of eFIRE significantly re-

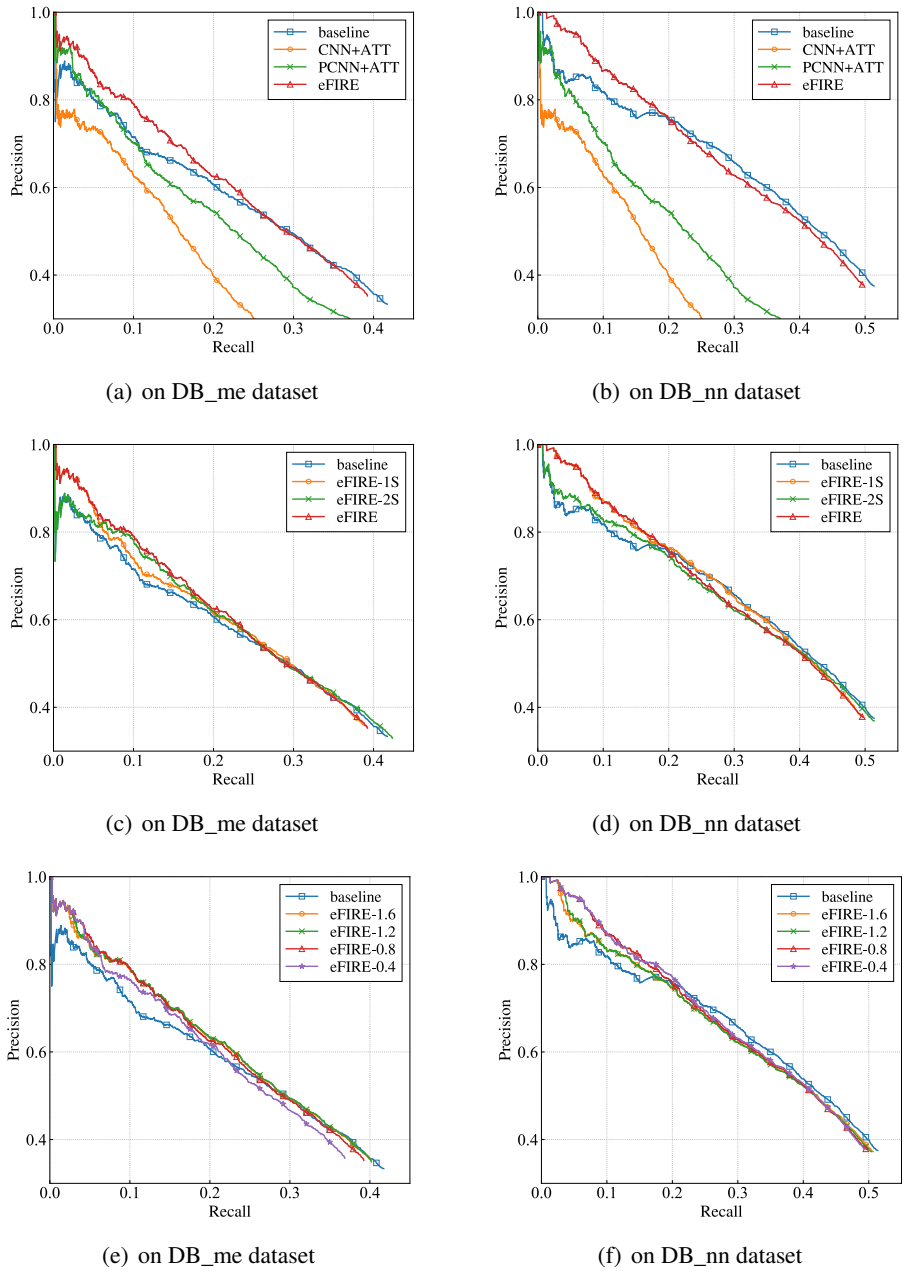


(a) on DB_me dataset (b) on DB_nn dataset

(c) on DB_me dataset (d) on DB_nn dataset

(e) on DB_me dataset (f) on DB_nn dataset

Figure 3: Overall performances: eFIRE vs. its variants vs. baseline vs. CNN+ATT vs. PCNN+ATT.

duces the number of variables and constraints encoded in the ILP solver, as shown in Table 3.

Note that the running time has no obvious linear relationships with the number of variables and constraints among different datasets. In addition to the number of variables and constraints, objective functions have a impact on the running time of ILP too. Further, CPLEX is used as a black box, which makes it hard to have a precise analysis.

These results tell us that the easy-first strategy for RE, by making use of the redundancy information embedded in the local results of sentence level extractors, is an effective and efficient complement for RE using ILP solvers.

**Exp-2: Performance of easy first collective inference with S-O redundancies**. In the second set of tests, in order to evaluate the impacts of S-O redundancies, we implemented a variant of our approach, referred to as eFIRE-1S, that makes easy decisions with the easy first collective inference, and then adopts the same ILP method in baseline for making the rest decisions. The results are reported in Figures 3(c), 3(d), and Table 3.

Method eFIRE-1S outperforms baseline in the

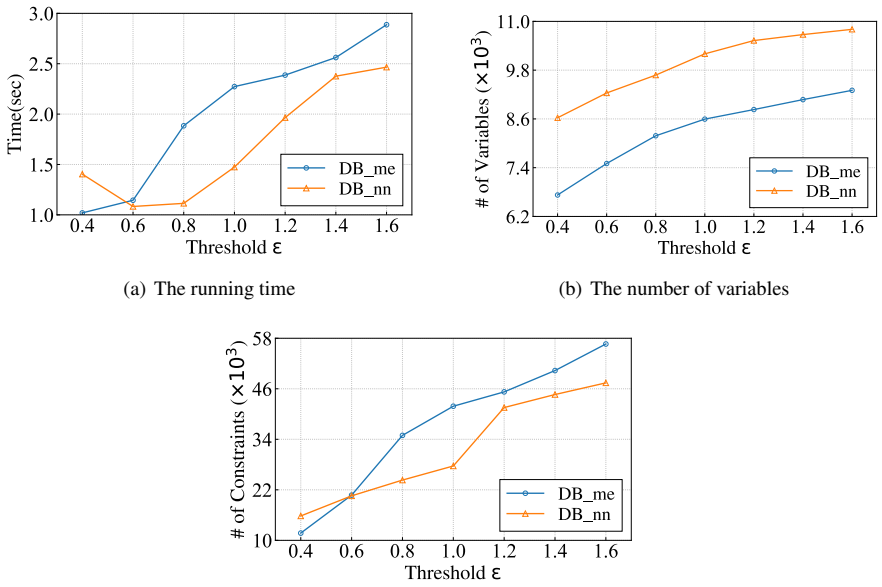


(a) The running time

(b) The number of variables

(c) The number of constraints

Figure 4: Impacts of threshold $\epsilon$ on time, variables and constraints.

low-recall portions of the P-R curves on all two datasets. These results tell us that the easy first collective inference using S-O redundancies can not only improve the effectiveness of decision making for RE, but also improve the efficiency, as it significantly reduces the constraints and variables encoded in the ILP solver.

**Exp-3: Performance of ILP with S-R, O-R and R redundancies**. In the third set of tests, to evaluate the impacts of S-R, O-R and R redundancies, we implemented a variant of eFIRE, referred to as eFIRE-2S, that only consists of the second stage of eFIRE. That is, all decisions of eFIRE-2S are made by the ILP solver. The results are reported in Figures 3(c), 3(d) and Table 3.

Method eFIRE-2S outperforms baseline on all two datasets. It essentially improves the precision without sacrificing the recall. For ILP based methods, their key differences are the objective functions. eFIRE-2S incorporates more reliable statistics of sentence level extractors, *i.e.,* S-R, O-R and R redundancies, while baseline only uses maximal scores to encourage choosing the candidates with higher individual sentence level confidence scores. So, S-R, O-R and R redundancies benefit the decision making of ILP solvers. The efficiency of eFIRE-2S and baseline are comparable, which implies that the efficiency benefit of eFIRE comes from its first stage easy first collective inference.

**Exp-4: Setting verification of threshold** $\epsilon$. In the last set of tests, to verify the setting of threshold $\epsilon$, we varied its values to $[0.4, 1.6]$. The results of effectiveness and running time are reported in Figures 3(e), 3(f) and Figure 4, respectively.

The results show that our approach eFIRE is effective and efficient when $\epsilon$ falls into $[0.5, 0.9]$, during which eFIRE outperforms baseline in the low-recall portion. This justifies our setting for threshold $\epsilon$, which is typically a little less than 1.0 to distinguish easy decisions from hard ones (Section 3.1). Threshold $\epsilon$ obviously has an impact on the running time, as the smaller $\epsilon$ is, the more running time eFIRE has in the first stage, and the less it has in the second stage.

**Summary.** From these tests, we find followings.

(1) Our approach eFIRE is both effective and efficient. It is more accurate than baseline in the low-recall portion $[0, 0.25]$ and CNN+ATT and PCNN+ATT consistently. eFIRE is on average (4.58%, 23.05%, 12.76%) more accurate in the low-recall portion and (21, 50, 25) times faster than baseline, CNN+ATT and PCNN+ATT, respectively. (2) The use of the easy-first strategy and S-O redundancies both improves the accuracy of RE and reduces the running time, and the use of S-R, O-R and R redundancies improves the accuracy of RE. (3) The setting of threshold $\epsilon$ is flexible in a range of $[0.5, 0.9]$ for our approach eFIRE.

## 5 Related Work

Relation extraction has been studied extensively in recent years, and can be divided into local re-

lation extractors (Mintz et al., 2009; Surdeanu et al., 2010; Hoffmann et al., 2011; Surdeanu et al., 2012; Søgaard et al., 2015) using the lexical features, syntactic features, and other local features of sentences, and global relation extractors utilizing the corpus features and relationships among local decisions (Kate and Mooney, 2010; Yao et al., 2010; Li et al., 2011; Singh et al., 2013; Li and Ji, 2014; Nguyen et al., 2017; Su et al., 2017). Global relation extractors leverage more information to resolve conflict decisions, which typically leads to more accurate decisions than local ones.

Recently, neural network-based models (Zeng et al., 2014; Ji et al., 2017) have been proposed for RE. Lin et al. (2016) proposes an attention mechanism to calculate weights for all sentences of one entity pair and selects plausible sentences from noisy sentences. Different from these are supervised methods that need label data for training, we propose an unsupervised method in this study.

Disagreements among decisions can be resolved by constraints. Yao et al. (2010) propose a relation extraction model that captures selectional preferences and functionality constraints to integrate information across documents. Fader et al. (2011) implement syntactic and lexical constraints on binary relations expressed by verbs in Open IE systems. Koch et al. (2014) impose type (or domain) constraints to only allow relations over appropriately typed mentions for relation extraction. Similar to (Chen et al., 2014a), our approach utilizes both domain and uniqueness constraints to resolve disagreements.

Many global relation extractors use integer linear programming as a blackbox solver (Roth and Yih, 2004; Choi et al., 2006; Roth and Yih, 2007; Li et al., 2011; Ji et al., 2013; Chen et al., 2014a,b; Wang et al., 2015). The ILP solver derives decisions through a well designed objective function, and resolves conflict decisions by encoding constraints into ILP. Our easy first approach is complimentary to these methods with each other, as these methods can take the easy first collective inference of our approach for making easy decisions as a first step, and our approach can make use of any of these methods as its solution for making hard decisions in its second stage.

Redundancies in the corpus have been used to pick up hidden information. Downey et al. (2005) consider redundant extractions for judging the correctness of extractions. Li et al. (2011) take advantage of redundant information to conduct reasoning across documents based on the information network structure. We introduce four classes of information redundancies: S-O, S-R, O-R and R redundancies, and we leverage S-O redundancies for making easy decisions, and the others to aid hard decision making.

Easy-first strategy relies on the intuition that "easy decisions should be made early, while harder decisions should be left for later when more information is available (Stoyanov and Eisner, 2012)". Their method makes easy decisions first for coreference resolution, and further makes use of the information from coreference clusters in the form of features to make later decisions. In this study, we propose to make easy (most confident) decisions first for relation extraction, and then to make hard decisions with ILP, where easy decisions are distinguished from hard ones with redundance scores based on S-O redundancies.

Data dependencies have well studied for improving data quality (Ma, 2011; Ma et al., 2015; Fan and Geerts, 2012), which essentially make use of data redundancies and dependencies. Our approach is partially inspired by these studies. Indeed, the uniqueness constraints defined in this study can be treated as functional dependencies (Abiteboul et al., 1995).

## 6 Conclusions

In this paper, we have proposed a fast easy first approach for relation extraction by making use of information redundancies, embedded in the results produced by local sentence level extractors, under domain and uniqueness constraints. We have introduced four classes of information redundancies to aid both easy first collective inference for easy decisions in the first stage and ILP for hard decisions in the second stage. Finally, using datasets processed by sentence level extractors with different models, we have experimentally verified that our easy first approach is both effective and efficient compared with state-of-the-art both ILP and neural network-based methods.

## Acknowledgments

This work is supported in part by National Key R&D Program of China 2018YFB1700403, NSFC U1636210&61421003, and Shenzhen Institute of Computing Sciences.

## References


Serge Abiteboul, Richard Hull, and Victor Vianu. 1995. *Foundations of Databases*. Addison-Wesley.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 2670–2676.

Adam L. Berger, Stephen Della Pietra, and Vincent J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. 2009. Dbpedia - A crystallization point for the web of data. *Journal of Web Semantics*, 7(3):154–165.

Razvan C. Bunescu and Raymond J. Mooney. 2007. Learning to extract relations from the web using minimal supervision. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 576–583. Association for Computational Linguistics.

Liwei Chen, Yansong Feng, Songfang Huang, Yong Qin, and Dongyan Zhao. 2014a. Encoding relation requirements for relation extraction via joint inference. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 818–827. Association for Computational Linguistics.

Liwei Chen, Yansong Feng, Jinghui Mo, Songfang Huang, and Dongyan Zhao. 2014b. Joint inference for knowledge base population. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1912–1923. Association for Computational Linguistics.

Yejin Choi, Eric Breck, and Claire Cardie. 2006. Joint extraction of entities and relations for opinion recognition. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 431–439. Association for Computational Linguistics.

Doug Downey, Oren Etzioni, and Stephen Soderland. 2005. A probabilistic model of redundancy in information extraction. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, pages 1034–1041. Professional Book Center.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545. Association for Computational Linguistics.

James Fan, Raphael Hoffman, Aditya Kalyanpur, Sebastian Riedel, Fabian M. Suchanek, and Partha Pratim Talukdar, editors. 2012. *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction, AKBC-WEKEX@NAACL-HLT 2012, Montrèal, Canada, June 7-8, 2012*. Association for Computational Linguistics.

Wenfei Fan and Floris Geerts. 2012. *Foundations of Data Quality Management*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke S. Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of ACL 2011, the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 541–550. Association for Computational Linguistics.

Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. 2017. Distant supervision for relation extraction with sentence-level attention and entity descriptions. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 3060–3066. AAAI Press.

Heng Ji, Benoît Favre, Wen-Pin Lin, Dan Gillick, Dilek Hakkani-Tür, and Ralph Grishman. 2013. Open-domain multi-document summarization via information extraction: Challenges and prospects. In *Multi-source, Multilingual Information Extraction and Summarization*, pages 177–201. Springer.

Rohit J. Kate and Raymond J. Mooney. 2010. Joint entity and relation extraction using card-pyramid parsing. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 203–212. Association for Computational Linguistics.

Mitchell Koch, John Gilmer, Stephen Soderland, and Daniel S. Weld. 2014. Type-aware distantly supervised relation extraction with linked arguments. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1891–1901. Association for Computational Linguistics.

Oier Lopez de Lacalle and Mirella Lapata. 2013. Unsupervised relation extraction with general domain knowledge. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 415–425. Association for Computational Linguistics.

Qi Li, Sam Anzaroot, Wen-Pin Lin, Xiang Li, and Heng Ji. 2011. Joint inference for cross-document information extraction. In *Proceedings of the 20th ACM Conference on Information and Knowledge Management*, pages 2225–2228. Association for Computing Machinery.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 402–412. Association for Computational Linguistics.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, (Volume 1: Long Papers)*, pages 73–82. Association for Computational Linguistics.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, (Volume 1: Long Papers)*, pages 2124–2133. Association for Computational Linguistics.

Shuai Ma. 2011. *Extending dependencies for improving data quality*. Ph.D. thesis, University of Edinburgh, UK.

Shuai Ma, Liang Duan, Wenfei Fan, Chunming Hu, and Wenguang Chen. 2015. Extending conditional dependencies with built-in predicates. *IEEE Transactions on Knowledge and Data Engineering*, 27(12):3274–3288.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011. Association for Computational Linguistics.

Dat Ba Nguyen, Martin Theobald, and Gerhard Weikum. 2017. J-REED: joint relation extraction and entity disambiguation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 2227–2230. Association for Computing Machinery.

Dan Roth and Wen-tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proceedings of the Eighth Conference on Computational Natural Language Learning at HLT-NAACL 2004*, pages 1–8. Association for Computational Linguistics.

Dan Roth and Wen-tau Yih. 2007. Global inference for entity and relation identification via a linear programming formulation. *Introduction to statistical relational learning*, pages 553–580.

Sameer Singh, Sebastian Riedel, Brian Martin, Jiaping Zheng, and Andrew McCallum. 2013. Joint inference of entities, relations, and coreference. In *Proceedings of the 2013 workshop on Automated knowledge base construction, AKBC@CIKM*, pages 1–6. Association for Computing Machinery.

Anders Søgaard, Barbara Plank, and Héctor Martínez Alonso. 2015. Using frame semantics for knowledge extraction from twitter. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2447–2452. AAAI Press.

Veselin Stoyanov and Jason Eisner. 2012. Easy-first coreference resolution. In *Proceedings of COLING 2012, the 24th International Conference on Computational Linguistics: Technical Papers*, pages 2519–2534. The COLING 2012 Organizing Committee.

Yu Su, Honglei Liu, Semih Yavuz, Izzeddin Gur, Huan Sun, and Xifeng Yan. 2017. Global relation embedding for relation extraction. *CoRR*, abs/1704.05958.

Mihai Surdeanu, David McClosky, Julie Tibshirani, John Bauer, Angel X. Chang, Valentin I. Spitkovsky, and Christopher D. Manning. 2010. A simple distant supervision approach for the TAC-KBP slot filling task. In *Proceedings of the Third Text Analysis Conference*. National Institute of Standards and Technology.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 455–465. Association for Computational Linguistics.

Quan Wang, Bin Wang, and Li Guo. 2015. Knowledge base completion using embeddings and rules. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, pages 1859–1866. AAAI Press.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2010. Collective cross-document relation extraction without labelled data. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1013–1023. Association for Computational Linguistics.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1753–1762. Association for Computational Linguistics.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344. Dublin City University and Association for Computational Linguistics.

Congle Zhang, Raphael Hoffmann, and Daniel S. Weld. 2012. Ontological smoothing for relation extraction with minimal supervision. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*. AAAI Press.

# Dependency-Guided LSTM-CRF for Named Entity Recognition

**Zhanming Jie** and **Wei Lu**
StatNLP Research Group
Singapore University of Technology and Design
zhanming_jie@mymail.sutd.edu.sg, luwei@sutd.edu.sg

## Abstract

Dependency tree structures capture long-distance and syntactic relationships between words in a sentence. The syntactic relations (e.g., *nominal subject*, *object*) can potentially infer the existence of certain named entities. In addition, the performance of a named entity recognizer could benefit from the long-distance dependencies between the words in dependency trees. In this work, we propose a simple yet effective *dependency-guided* LSTM-CRF model to encode the complete dependency trees and capture the above properties for the task of named entity recognition (NER). The data statistics show strong correlations between the entity types and dependency relations. We conduct extensive experiments on several standard datasets and demonstrate the effectiveness of the proposed model in improving NER and achieving state-of-the-art performance. Our analysis reveals that the significant improvements mainly result from the dependency relations and long-distance interactions provided by dependency trees.

## 1 Introduction

Named entity recognition (NER) is one of the most important and fundamental tasks in natural language processing (NLP). Named entities capture useful semantic information which was shown helpful for downstream NLP tasks such as coreference resolution (Lee et al., 2017), relation extraction (Miwa and Bansal, 2016) and semantic parsing (Dong and Lapata, 2018). On the other hand, dependency trees also capture useful semantic information within natural language sentences. Currently, research efforts have derived useful discrete features from dependency structures (Sasano and Kurohashi, 2008; Cucchiarelli and Velardi, 2001; Ling and Weld, 2012) or structural constraints (Jie et al., 2017) to help the NER task. However, how to make good use of the rich relational information as well as complex long-distance interactions among words as conveyed by the complete dependency structures for improved NER remains a research question to be answered.

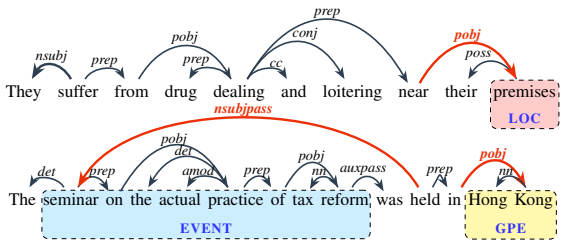


Figure 1: Example sentences annotated with named entiteis and dependencies in the OntoNotes 5.0 dataset.

The first example in Figure 1 illustrates the relationship between a dependency structure and a named entity. Specifically, the word "*premises*", which is a named entity of type LOC (location), is characterized by the incoming arc with label "*pobj*" (prepositional object). This arc reveals a certain level of the semantic role that the word "*premises*" plays in the sentence. Similarly, the two words "*Hong Kong*" in the second example that form an entity of type GPE are also characterized by a similar dependency arc towards them.

The long-distance dependencies capturing non-local structural information can also be very helpful for the NER task (Finkel et al., 2005). In the second example of Figure 1, the long-distance dependency from "*held*" to "*seminar*" indicates a direct relation "*nsubjpass*" (passive subject) between them, which can be used to characterize the existence of an entity. However, existing NER models based on linear-chain structures would have difficulties in capturing such long-distance relations (i.e., non-local structures).

One interesting property, as highlighted in the work of Jie et al. (2017), is that most of the en-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3862–3872,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

tities form subtrees under their corresponding dependency trees. In the example of the EVENT entity in Figure 1, the entity itself forms a subtree and the words inside have rich complex dependencies among themselves. Exploiting such dependency edges within the subtrees allows a model to capture non-trivial semantic-level interactions between words within long entities. For example, "*practice*" is the prepositional object (*pobj*) of "*on*" which is a preposition (*prep*) of "*seminar*" in the EVENT entity. Modeling these grandchild dependencies (GD) (Koo and Collins, 2010) requires the model to capture some higher-order long-distance interactions among different words in a sentence.

Inspired by the above characteristics of dependency structures, in this work, we propose a simple yet effective dependency-guided model for NER. Our neural network based model is able to capture both contextual information and rich long-distance interactions between words for the NER task. Through extensive experiments on several datasets on different languages, we demonstrate the effectiveness of our model, which achieves the state-of-the-art performance. To the best of our knowledge, this is the first work that leverages the complete dependency graphs for NER. We make our code publicly available at `http://www.statnlp.org/research/information-extraction`.

## 2 Related Work

NER has been a long-standing task in the field of NLP. While many recent works (Peters et al., 2018a; Akbik et al., 2018; Devlin et al., 2019) focus on finding good contextualized word representations for improving NER, our work is mostly related to the literature that focuses on employing dependency trees for improving NER.

Sasano and Kurohashi (2008) exploited the syntactic dependency features for Japanese NER and achieved improved performance with a support vector machine (SVM) (Cortes and Vapnik, 1995) classifier. Similarly, Ling and Weld (2012) included the head word in a dependency edge as features for fine-grained entity recognition. Their approach is a pipeline where they extract the entity mentions with linear-chain conditional random fields (CRF) (Lafferty et al., 2001) and used a classifier to predict the entity type. Liu et al. (2010) proposed to link the words that are associated with selected typed dependencies (e.g., "*nn*", "*prep*") using a skip-chain CRF (Sutton and McCallum, 2004) model. They showed that some specific relations between the words can be exploited for improved NER. Cucchiarelli and Velardi (2001) applied a dependency parser to obtain the syntactic relations for the purpose of unsupervised NER. The resulting relation information serves as the features for potential existence of named entities. Jie et al. (2017) proposed an efficient dependency-guided model based on the semi-Markov CRF (Sarawagi and Cohen, 2004) for NER. The purpose is to reduce time complexity while maintaining the non-Markovian features. They observed certain relationships between the dependency edges and the named entities. Such relationships are able to define a reduced search space for their model. While these previous approaches do not make full use of the dependency tree structures, we focus on exploring neural architectures to exploit the complete structural information conveyed by the dependency trees.

## 3 Model

Our dependency-guided model is based on the state-of-the-art BiLSTM-CRF model proposed by Lample et al. (2016). We first briefly present their model as background and next present our dependency-guided model.

### 3.1 Background: BiLSTM-CRF

In the task of named entity recognition, we aim to predict the label sequence $\mathbf{y} = \{y_1, y_2, \cdots, y_n\}$ given the input sentence $\mathbf{x} = \{x_1, x_2, \cdots, x_n\}$ where $n$ is the number of words. The labels in $\mathbf{y}$ are defined by a label set with the standard IOBES[1] labeling scheme (Ramshaw and Marcus, 1999; Ratinov and Roth, 2009). The CRF (Lafferty et al., 2001) layer defines the probability of the label sequence $\mathbf{y}$ given $\mathbf{x}$:

$$P(\mathbf{y}|\mathbf{x}) = \frac{\exp(score(\mathbf{x}, \mathbf{y}))}{\sum_{\mathbf{y}'} \exp(score(\mathbf{x}, \mathbf{y}'))} \quad (1)$$

Following Lample et al. (2016), the score is defined as the sum of transitions and emissions from the bidirectional LSTM (BiLSTM):

$$score(\mathbf{x}, \mathbf{y}) = \sum_{i=0}^{n} A_{y_i, y_{i+1}} + \sum_{i=1}^{n} F_{\mathbf{x}, y_i} \quad (2)$$

[1] "S-" indicates the entity with a single word and "E-" indicates the end of an entity.

where $\mathbf{A}$ is a transition matrix in which $A_{y_i,y_{i+1}}$ is the transition parameter from the label $y_i$ to the label $y_{i+1}$[2]. $\mathbf{F}_{\mathbf{x}}$ is an emission matrix where $F_{\mathbf{x},y_i}$ represents the scores of the label $y_i$ at the $i$-th position. Such scores are provided by the parameterized LSTM (Hochreiter and Schmidhuber, 1997) networks. During training, we minimize the negative log-likelihood to obtain the model parameters including both LSTM and transition parameters.

### 3.2 Dependency-Guided LSTM-CRF

**Input Representations** The word representation $\mathbf{w}$ in the BiLSTM-CRF (Lample et al., 2016; Ma and Hovy, 2016; Reimers and Gurevych, 2017) model consists of the concatenation of the word embedding as well as the corresponding character-based representation. Inspired by the fact that each word (except the root) in a sentence has exactly one *head* (i.e., *parent*) word in the dependency structure, we can enhance the word representations with such dependency information. Similar to the work by Miwa and Bansal (2016), we concatenate the word representation together with the corresponding head word representation and dependency relation embedding as the input representation. Specifically, given a dependency edge $(x_h, x_i, r)$ with $x_h$ as parent, $x_i$ as child and $r$ as dependency relation, the representation at position $i$ can be denoted as:

$$\mathbf{u}_i = [\mathbf{w}_i; \mathbf{w}_h; \mathbf{v}_r], \quad x_h = parent(x_i) \qquad (3)$$

where $\mathbf{w}_i$ and $\mathbf{w}_h$ are the word representations of the word $x_i$ and its parent $x_h$, respectively. We take the final hidden state of character-level BiLSTM as the character-based representation (Lample et al., 2016). $\mathbf{v}_r$ is the embedding for the dependency relation $r$. These relation embeddings are randomly initialized and fine-tuned during training. The above representation allows us to capture the direct long-distance interactions at the input layer. For the word that is a root of the dependency tree, we treat its parent as itself[3] and create a root relation embedding. Additionally, contextualized word representations (e.g., ELMo) can also be concatenated into $\mathbf{u}$.

**Neural Architecture** Given the dependency-encoded input representation $\mathbf{u}$, we apply the LSTM to capture the contextual information and model the interactions between the words and their corresponding parents in the dependency trees. Figure 2 shows the proposed dependency-guided LSTM-CRF (**DGLSTM-CRF**) with 2 LSTM layers for the example sentence "*Abramov had an accident in Moscow*" and its dependency structure. The corresponding label sequence is {S-PER, O, O, O, O, S-GPE}. Followed by the first BiLSTM, the hidden states at each position will propagate to the next BiLSTM layer and its child along the dependency trees. For example, the hidden state of the word "*had*", $\mathbf{h}_2^{(1)}$, will propagate to its child "*Abramov*" at the first position. For the word that is root, the hidden state at that specific position will propagate to itself. We use an *interaction function* $g(\mathbf{h}_i, \mathbf{h}_{p_i})$ to capture the interaction between the child and its parent in a dependency. Such an interaction function can be concatenation, addition or a multilayer perceptron (MLP). We further apply another BiLSTM layer on top of the interaction functions to produce the context representation for the final CRF layer.

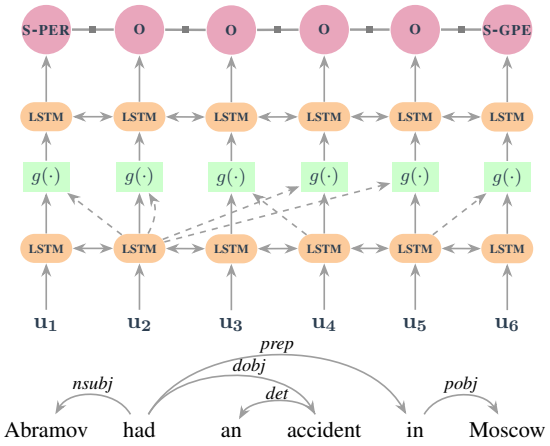


Figure 2: Dependency-guided LSTM-CRF with 2 LSTM Layers. Dashed connections mimic the dependency edges. "$g(\cdot)$" represents the interaction function.

The architecture shown in Figure 2 with a 2-layer BiLSTM can effectively encode the grandchild dependencies because the input representations encode the parent information and the interaction function further propagate the grandparent information. Such propagations allow the model to capture the indirect long-distance interactions from the grandchild dependencies between the words in the sentence as mentioned in Section 1. In general, we can stack more interaction functions and BiLSTMs to enable deeper reasoning over the dependency trees. Specifically, the hid-

[2] $y_0$ and $y_{n+1}$ are `start` and `end` labels.

[3] We also tried using a root word embedding but the performance is similar.

| Interaction Function | $g(\mathbf{h}_i, \mathbf{h}_{p_i})$ |
|---|---|
| Self connection | $\mathbf{h}_i$ |
| Concatenation | $\mathbf{h}_i \bigoplus \mathbf{h}_{p_i}$ |
| Addition | $\mathbf{h}_i + \mathbf{h}_{p_i}$ |
| MLP | $\text{ReLU}(\mathbf{W}_1\mathbf{h}_i + \mathbf{W}_2\mathbf{h}_{p_i})$ |

Table 1: List of interaction functions.

den states of the $(l+1)$-th layer $\mathbf{H}^{(l+1)}$ can be calculated from the hidden state of the previous layer $\mathbf{H}^{(l)}$:

$$\mathbf{H}^{(l+1)} = \text{BiLSTM}\Big(f\big(\mathbf{H}^{(l)}\big)\Big)$$
$$\mathbf{H}^{(l)} = \Big[\mathbf{h}_1^{(l)}, \mathbf{h}_2^{(l)}, \cdots, \mathbf{h}_n^{(l)}\Big]$$
$$f\big(\mathbf{H}^{(l)}\big) = \Big[g(\mathbf{h}_1^{(l)}, \mathbf{h}_{p_1}^{(l)}), \cdots, g(\mathbf{h}_n^{(l)}, \mathbf{h}_{p_n}^{(l)})\Big]$$

where $p_i$ indicates the parent index of the word $x_i$. $g(\mathbf{h}_i^{(l)}, \mathbf{h}_{p_i}^{(l)})$ represents the interaction functions between the hidden state at the $i$-th and $p_i$-th positions under the dependency edges $(x_{p_i}, x_i)$. The number of layers $L$ can be chosen according to the performance on the development set.

**Interaction Function** The interaction function between the parent and child representations can be defined in various ways. Table 1 shows the list of interaction function considered in our experiments. The first one returns the hidden state itself, which is equivalent to stacking the LSTM layers. The concatenation and addition involve no parameter, which are straightforward ways to model the interactions. The last one applies an MLP to model the interaction between parent and child representations. With the rectified linear unit (ReLU) as activation function, the $g(\mathbf{h}_i, \mathbf{h}_{p_i})$ function is analogous to a graph convolutional network (GCN) (Kipf and Welling, 2017) formulation. In such a graph, each node has a self connection (i.e., $\mathbf{h}_i$) and a dependency connection with parent (i.e., $\mathbf{h}_{p_i}$). Similar to the work by Marcheggiani and Titov (2017), we adopt different parameters $\mathbf{W}_1$ and $\mathbf{W}_2$ for self and dependency connections.

## 4 Experiments

**Datasets** The main experiments are conducted on the large-scale OntoNotes 5.0 (Weischedel et al., 2013) English and Chinese datasets. We chose these datasets because they contain both constituency tree and named entity annotations. There are 18 types of entities defined in the OntoNotes dataset. We convert the constituency trees into the Stanford dependency (De Marneffe and Manning, 2008) trees using the rule-based tool (De Marneffe et al., 2006) by Stanford CoreNLP (Manning et al., 2014). For English, Pradhan et al. (2013) provided the train/dev/test split[4] and the split has been used by several previous works (Chiu and Nichols, 2016; Li et al., 2017; Ghaddar and Langlais, 2018). For Chinese, we use the official splits provided by Pradhan et al. (2012)[5].

| Dataset | Train | | Dev | | Test | | ST | GD |
|---|---|---|---|---|---|---|---|---|
| | # Sent. | # Entity | # Sent. | # Entity | # Sent. | # Entity | (%) | (%) |
| OntoNotes 5.0 - English | 59,924 | 81,828 | 8,528 | 11,066 | 8,262 | 11,057 | 98.5 | 41.1 |
| OntoNotes 5.0 - Chinese | 36,487 | 62,543 | 6,083 | 9,104 | 4,472 | 7,494 | 92.9 | 49.1 |
| SemEval2010T1 - Catalan | 8,709 | 15,278 | 1,445 | 2,431 | 1,698 | 2,910 | 100.0 | 28.6 |
| SemEval2010T1 - Spanish | 9,022 | 17,297 | 1,419 | 2,615 | 1,705 | 3,046 | 100.0 | 29.8 |

Table 2: Dataset statistics. "ST" is the ratio of entities that form subtrees. "GD" is the ratio of entities that have grandchild dependencies within their subtrees.

Besides, we also conduct experiments on the Catalan and Spanish datasets from the SemEval-2010 Task 1[6] (Recasens et al., 2010)[7]. The SemEval-2010 task was originally designed for the task of coreference resolution in multiple languages. Again, we chose these corpora primarily because they contain both dependency and named entity annotations. Following Finkel and Manning (2009) and Jie et al. (2017), we select the most dominant three entity types and merge the rest into one general a entity type "*misc*". Table 2 shows the statistics of the datasets used in main experiments. To further evaluate the effectiveness of the dependency structures, we also conduct additional experiments under a low-resource setting for NER (Cotterell and Duh, 2017).

The last two columns of Table 2 show the relationships between the dependency trees and named entities with length larger than 2 for the complete dataset. Specifically, the penultimate column shows the percentage of entities that can form a complete subtree (ST) under their dependency tree structures. Apparently, most of the entities form subtrees, especially for the Catalan and Spanish datasets where 100% entities form subtrees. This observation is consistent with the findings reported in Jie et al. (2017). The last column in Table 2 shows the percentage of the grandchild

[4]http://cemantix.org/data/ontonotes.html
[5]http://conll.cemantix.org/2012/data.html
[6]http://stel.ub.edu/semeval2010-coref/download
[7]This dataset also has English portion but it is a subset of the OntoNotes English.

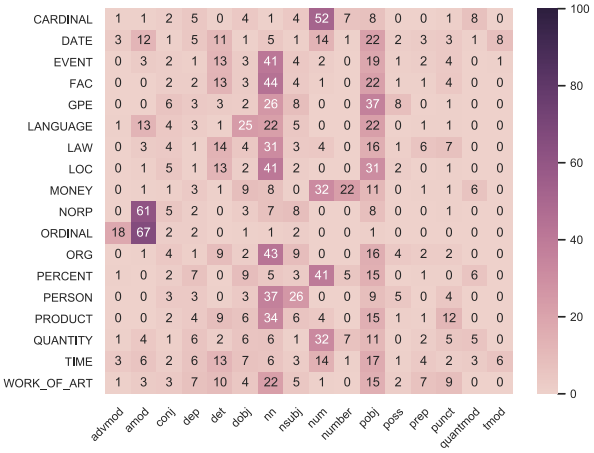


Figure 3: Percentage of entity words ($y$ axis) with respect to dependency relations ($x$ axis) in the OntoNotes English dataset. Columns with percentage less than 5% are ignored for brevity.

dependencies (Koo and Collins, 2010) (GD) that exist in these subtrees (i.e., entities). Such grandchild dependencies could be useful for detecting certain named entities, especially for long entities. As we will see later in Section 5, the performance of long entities can be significantly improved with our dependency-guide model.

The heatmap table in Figure 3 shows the correlation between the entity types and the dependency relations in the OntoNotes English dataset. Specifically, each entry denotes the percentage of the entities that have a parent dependency with a specific dependency relation. For example, at the row with GPE entity, 37% of the entity words[8] have a dependency edge whose label is "*pobj*". When looking at column of "*pobj*" and "*nn*", we can see that most of the entities relate to the prepositional object (*pobj*) and noun compound modifier (*nn*) dependencies. Especially for the NORP (i.e., *nationalities or religious or political groups*) and ORDINAL (e.g., "*first*", "*second*") entities, more than 60% of the entity words have the dependency with adjectival modifier (*amod*) relation. Furthermore, every entity type (i.e., row) has a most related dependency relation (with more than 17% occurrences). Such observations present useful information that can be used to categorize named entities with different types.

**Baselines** We implemented the state-of-the-art NER model **BiLSTM-CRF** (Lample et al., 2016) as the first baseline with different number of LSTM layers ($L = \{0, 1, 2, 3\}$). $L = 0$ indicates the model only relies on the input representation. Following Zhang et al. (2018), the complete dependency trees are considered bidirectional and encoded with a contextualized GCN (BiLSTM-GCN). We further add the relation-specific parameters (Marcheggiani and Titov, 2017) and a CRF layer for the NER task. The resulting baseline is **BiLSTM-GCN-CRF** [9]. We use the bootstrapping paired t-test (Berg-Kirkpatrick et al., 2012) for significance test when comparing the results of different models.

**Experimental Setup** We choose MLP as the interaction function in our DGLSTM-CRF according to performance on the development set. The hidden size of all models (i.e., LSTM, GCN) is set to 200. We use the Glove (Pennington et al., 2014) 100-d word embeddings, which was shown to be effective in English NER task (Ma and Hovy, 2016; Peters et al., 2018a). We use the publicly available FastText (Grave et al., 2018) word embeddings for Chinese, Catalan and Spanish. The ELMo (Peters et al., 2018a), deep contextualized word representations[10] are used for all languages in our experiments since Che et al. (2018) provides ELMo for many other languages[11], including Chinese, Catalan and Spanish. We use the average weights over all layers of the ELMo representations and concatenate them with the input representation $\mathbf{u}$. Our models are optimized by mini-batch stochastic gradient descent (SGD) with learning rate 0.01 and batch size 10. The $L_2$ regularization parameter is 1e-8. The hyperparameters are selected according to the performance on the OntoNotes English development set.

### 4.1 Main Results

**OntoNotes English** Table 3 shows the performance comparison between our work and previous work on the OntoNotes English dataset. Without the LSTM layers (i.e., $L = 0$), the proposed model with dependency information significantly improves the NER performance with more than 2 points in $F_1$ compared to the baseline BiLSTM-CRF ($L = 0$), which demonstrate the effective-

[8] The words that are annotated with entity labels.

[9] Detailed description of this baseline can also be found in the supplementary material.

[10] We also tried BERT (Devlin et al., 2019) in preliminary experiments and obtained similar performance as ELMo. The NER performance using BERT without fine-tuning reported in Peters et al. (2019) is consistent with the one reported by ELMo (Peters et al., 2018a).

[11] https://github.com/HIT-SCIR/ELMoForManyLangs

| Model | Prec. | Rec. | $F_1$ |
|---|---|---|---|
| Chiu and Nichols (2016) | 86.04 | 86.53 | 86.28 |
| Li et al. (2017) | 88.00 | 86.50 | 87.21 |
| Ghaddar and Langlais (2018) | - | - | 87.95 |
| Strubell et al. (2017) | - | - | 86.84 |
| BiLSTM-CRF ($L = 0$) | 82.03 | 80.78 | 81.40 |
| BiLSTM-CRF ($L = 1$) | 87.21 | 86.93 | 87.07 |
| BiLSTM-CRF ($L = 2$) | 87.89 | 87.68 | 87.78 |
| BiLSTM-CRF ($L = 3$) | 87.81 | 87.50 | 87.65 |
| BiLSTM-GCN-CRF | 88.30 | 88.06 | 88.18 |
| DGLSTM-CRF ($L = 0$) | 85.31 | 82.19 | 84.09 |
| DGLSTM-CRF ($L = 1$) | **88.78** | 87.29 | 88.03 |
| DGLSTM-CRF ($L = 2$) | 88.53 | **88.50** | **88.52** |
| DGLSTM-CRF ($L = 3$) | 87.59 | 88.93 | 88.25 |
| **Contextualized Word Representation** | | | |
| Akbik et al. (2018) (Flair) | - | - | 89.30 |
| BiLSTM-CRF ($L = 0$) + ELMo | 85.44 | 84.41 | 84.92 |
| BiLSTM-CRF ($L = 1$) + ELMo | 89.14 | 88.59 | 88.87 |
| BiLSTM-CRF ($L = 2$) + ELMo | 88.25 | 89.71 | 88.98 |
| BiLSTM-CRF ($L = 3$) + ELMo | 88.03 | 89.04 | 88.53 |
| BiLSTM-GCN-CRF + ELMo | 89.40 | 89.71 | 89.55 |
| DGLSTM-CRF ($L = 0$) + ELMo | 86.87 | 85.12 | 85.99 |
| DGLSTM-CRF ($L = 1$) + ELMo | 89.40 | 89.96 | 89.68 |
| DGLSTM-CRF ($L = 2$) + ELMo | **89.59** | **90.17** | **89.88** |
| DGLSTM-CRF ($L = 3$) + ELMo | 89.43 | 90.15 | 89.79 |

Table 3: Performance comparison on the OntoNotes 5.0 English dataset.

| Model | Prec. | Rec. | $F_1$ |
|---|---|---|---|
| Pradhan et al. (2013) | 78.20 | 66.45 | 71.85 |
| Lattice LSTM (Z&Y, 2018) | 76.34 | 77.01 | 76.67 |
| BiLSTM-CRF ($L = 0$) | 76.67 | 67.79 | 71.95 |
| BiLSTM-CRF ($L = 1$) | **78.45** | 74.59 | 76.47 |
| BiLSTM-CRF ($L = 2$) | 77.94 | 75.33 | 76.61 |
| BiLSTM-CRF ($L = 3$) | 76.17 | 75.23 | 75.70 |
| BiLSTM-GCN-CRF | 76.35 | 75.89 | 76.12 |
| DGLSTM-CRF ($L = 0$) | 76.91 | 70.65 | 73.65 |
| DGLSTM-CRF ($L = 1$) | 77.79 | 75.29 | 76.52 |
| DGLSTM-CRF ($L = 2$) | 77.40 | **77.41** | **77.40** |
| DGLSTM-CRF ($L = 3$) | 77.01 | 74.90 | 75.94 |
| **Contextualized Word Representation** | | | |
| BiLSTM-CRF ($L = 0$) + ELMo | 75.20 | 73.39 | 74.28 |
| BiLSTM-CRF ($L = 1$) + ELMo | **79.20** | 79.21 | 79.20 |
| BiLSTM-CRF($L = 2$) + ELMo | 78.49 | 79.44 | 78.96 |
| BiLSTM-CRF ($L = 3$) + ELMo | 78.54 | 79.76 | 79.14 |
| BiLSTM-GCN-CRF + ELMo | 78.71 | 79.29 | 79.00 |
| DGLSTM-CRF ($L = 0$) + ELMo | 76.27 | 74.61 | 75.43 |
| DGLSTM-CRF ($L = 1$) + ELMo | 78.91 | 80.22 | 79.56 |
| DGLSTM-CRF ($L = 2$) + ELMo | 78.86 | **81.00** | **79.92** |
| DGLSTM-CRF ($L = 3$) + ELMo | 79.30 | 79.86 | 79.58 |

Table 4: Performance comparison on the OntoNotes 5.0 Chinese Dataset.

ness of dependencies for the NER task. Our best performing BiLSTM-CRF baseline (with Glove) achieves a $F_1$ score of 87.78 which is better than or on par with previous works (Chiu and Nichols, 2016; Li et al., 2017; Ghaddar and Langlais, 2018) with extra features. This baseline also outperforms the CNN-based models (Strubell et al., 2017; Li et al., 2017). The BiLSTM-GCN-CRF model outperforms the BiLSTM-CRF model but achieves inferior performance compared to the proposed DGLSTM-CRF model. We believe it is challenging to preserve the surrounding context information with stacking GCN layers while contextual information is important for NER (Peters et al., 2018b). Overall, the 2-layer DGLSTM-CRF model significantly (with $p < 0.01$) outperforms the best BiLSTM-CRF baseline and the BiLSTM-GCN-CRF model. As we can see from the table, increasing the number of layers (e.g., $L$ = 3) does not give us further improvements for both BiLSTM-CRF and DGLSTM-CRF because such third-order information (e.g., the relationship among a words parent, its grandparent, and great-grandparent) does not play an important role in indicating the presence of named entities.

We further compare the performance of all models with ELMo (Peters et al., 2018a) representations to investigate whether the effect of dependency would be diminished by the contextualized word representations. With $L = 0$, the ELMo representations largely improve the performance of BiLSTM-CRF compared to the BiLSTM-CRF model with word embeddings only but is still 1 point below our DGLSTM-CRF model. The 2-layer DGLSTM-CRF model outperforms the best BilSTM-CRF baseline with 0.9 points in $F_1$ ($p < 0.001$). Empirically, we found that among the entities that are correctly predicted by DGLSTM-CRF but wrongly predicted by BiLSTM-CRF, 47% of them are with length more than 2. Our finding shows the 2-layer DGLSTM-CRF model is able to accurately recognize long entities, which can lead to a higher precision. In addition, 51.9% of the entities that are correctly retrieved by DGLSTM-CRF have the dependency relations "*pobj*", "*nn*" and "*nsubj*", which have strong correlations with certain named entity types (Figure 3). Such a result demonstrates the effectiveness of dependency relations in improving the recall of NER.

**OntoNotes Chinese** Table 4 shows the performance comparison on the Chinese datasets. We compare our models against the state-of-the-art

| Model | Catalan | | | Spanish | | |
|---|---|---|---|---|---|---|
| | Prec. | Rec. | $F_1$ | Prec. | Rec. | $F_1$ |
| BiLSTM-CRF ($L = 0$) | 65.91 | 49.90 | 56.80 | 65.97 | 52.63 | 58.55 |
| BiLSTM-CRF ($L = 1$) | 76.83 | 63.47 | 69.51 | 78.33 | 69.89 | 73.87 |
| BiLSTM-CRF ($L = 2$) | 73.79 | 67.63 | 70.58 | 77.73 | 70.91 | 74.16 |
| BiLSTM-CRF ($L = 3$) | 74.75 | 67.35 | 70.86 | 76.41 | 72.95 | 74.64 |
| BiLSTM-GCN-CRF | 81.25 | 75.22 | 78.12 | 84.10 | 79.88 | 81.93 |
| DGLSTM-CRF ($L = 0$) | 73.42 | 61.79 | 67.10 | 74.90 | 61.21 | 67.38 |
| DGLSTM-CRF ($L = 1$) | 81.87 | 79.28 | 80.55 | 83.21 | 81.19 | 82.19 |
| DGLSTM-CRF ($L = 2$) | **83.35** | **80.00** | **81.64** | 84.05 | 82.90 | 83.47 |
| DGLSTM-CRF ($L = 3$) | 81.87 | 80.21 | 81.03 | **84.12** | **83.45** | **83.78** |
| **Contextualized Word Representation** | | | | | | |
| BiLSTM-CRF ($L = 0$) + ELMo | 67.53 | 64.47 | 65.96 | 73.16 | 69.01 | 71.03 |
| BiLSTM-CRF ($L = 1$) + ELMo | 77.85 | 76.22 | 77.03 | 81.72 | 79.09 | 80.38 |
| BiLSTM-CRF($L = 2$) + ELMo | 78.61 | 78.32 | 78.46 | 80.89 | 80.30 | 80.59 |
| BiLSTM-CRF($L = 3$) + ELMo | 79.11 | 77.32 | 78.21 | 80.48 | 79.45 | 79.96 |
| BiLSTM-GCN-CRF + ELMo | 83.68 | 83.16 | 83.42 | 85.31 | 85.19 | 85.25 |
| DGLSTM-CRF ($L = 0$) + ELMo | 70.87 | 65.81 | 68.25 | 75.96 | 72.52 | 74.20 |
| DGLSTM-CRF ($L = 1$) + ELMo | 82.29 | 82.37 | 82.33 | 84.05 | 84.77 | 84.41 |
| DGLSTM-CRF ($L = 2$) + ELMo | **84.71** | 83.75 | **84.22** | **87.79** | **87.33** | **87.56** |
| DGLSTM-CRF ($L = 3$) + ELMo | 84.50 | **83.92** | **84.21** | 86.74 | 86.57 | 86.66 |

Table 5: Results on the SemEval-2010 Task 1 datasets.

NER model on this dataset, Lattice LSTM (Zhang and Yang, 2018)[12]. Our implementation of the strong BiLSTM-CRF baseline achieves comparable performance against the Lattice LSTM. Similar to the English dataset, our model with $L$ = 0 significantly improves the performance compared to the BiLSTM-CRF ($L$ = 0) model. Our DGLSTM-CRF model achieves the best performance with $L$ = 2 and is consistently better ($p < 0.02$) than the strong BiLSTM-CRF baselines. As we can see from the table, the improvements of the DGLSTM-CRF model mainly come from recall ($p < 0.001$) compared to the BiLSTM model, especially in the scenario with word embeddings only. Empirically, we also found that those correctly retrieved entities of the DGLSTM-CRF (compared against the baseline) mostly correlate with the following dependency relations: "*nn*", "*nsubj*", "*nummod*". However, DGLSTM-CRF achieves lower precisions against BiLSTM-CRF, which indicates that the DGLSTM-CRF model makes more false-positive predictions. The reason could be the relatively lower ratio of ST(%)[13] as shown in Table 2, which means some of the entities do not form subtrees under the complete dependency trees. In such a scenario, the model may not correctly identify the boundary of the entities, which results in lower precision.

[12]We run their code on the OntoNotes 5.0 Chinese dataset.

[13]Percentage of entities that can form a subtree.

**SemEval-2010** Table 5 shows the results of our models on the SemEval-2010 Task 1 datasets. Overall, we observe substantial improvements of the DGLSTM-CRF on the Catalan and Spanish datasets (with $p < 0.001$ marked in bold against the best performing BiLSTM-CRF baseline), especially for DGLSTM-CRF with ELMo and $L$ larger than 1. With word embeddings, the best DGLSTM-CRF model outperforms the best performing BiLSTM-CRF baseline with more than 10 and 9 points in $F_1$ on the Catalan and Spanish datasets, respectively. The BiLSTM-GCN-CRF model also performs much better than the BiLSTM-CRF baselines but is worse than the DGLSTM-CRF model with $L \geq 2$. Both precision and recall significantly improve with a large margin compared to the best performing BiLSTM-CRF, especially for the recall (with more than 10 points improvement) on these two datasets. With ELMo, the best performing DGLSTM-CRF model outperforms the BiLSTM-CRF baseline with about 6 and 7 points in $F_1$ on these two datasets, respectively. The substantial improvements show that the structural dependency information is extremely helpful for these two datasets.

With ELMo representations, we observe about 2 and 3 points improvements in $F_1$ compared with the 1-layer DGLSTM-CRF model on these two datasets, respectively. Empirically, more than 50% of the entities that are correctly predicted by the

| Model | Prec. | Rec. | $F_1$ |
|---|---|---|---|
| Peters et al. (2018a) ELMo | - | - | 92.2 |
| BiLSTM-CRF + ELMo ($L = 2$) | 92.1 | 92.3 | 92.2 |
| DGLSTM-CRF + ELMo ($L = 2$) | 92.2 | 92.5 | 92.4 |

Table 6: Performance on the CoNLL-2003 English dataset.

| Model | Catalan | | | Spanish | | |
|---|---|---|---|---|---|---|
| | Prec. | Rec. | $F_1$ | Prec. | Rec. | $F_1$ |
| BiLSTM-CRF ($L = 1$) | 47.88 | 18.59 | 26.78 | 40.77 | 19.01 | 25.93 |
| DGLSTM-CRF ($L = 1$) | 47.71 | 31.55 | 37.98 | 49.39 | 31.91 | 38.77 |
| – with gold dependency | **52.13** | **33.26** | **40.61** | **52.14** | **35.59** | **42.30** |

Table 7: Low-resource NER performance on the SemEval-2010 Task 1 datasets.

2-layer model but not the 1-layer model are with length larger than 2. Also, most of these entities contain the grandchild dependencies "(*sn, sn*)" and "(*spec, sn*)" where *sn* represents noun phrase and *spec* represents specifier (e.g., determiner, quantifier) in both datasets. Such a finding shows that the 2-layer model is able to capture the interactions given by the grandchild dependencies.

### 4.2 Additional Experiments

**CoNLL-2003 English** Table 6 shows the performance on the CoNLL-2003 English dataset. The dependencies are predicted from Spacy (Honnibal and Montani, 2017). With the contextualized word representations, DGLSTM-CRF outperforms BiLSTM-CRF with 0.2 points in $F_1$ ($p < 0.09$). The improvement is not significant due to the relatively lower equality of the dependency trees. To further study the effect of the dependencies, we modified the predicted dependencies to ensure each entity form a subtree in the complete dataset. Such modification improves the $F_1$ to 92.7, which is significantly better ($p < 0.05$) than the BiLSTM-CRF.

**Low-Resource NER** Following Cotterell and Duh (2017), we emulate truly low-resource condition with 100 sentences for training. We assume that the contextualized word representations are not available and dependencies are predicted. Table 7 shows the NER performance on the SemEval-2010 Task 1 datasets under the low-resource setting. With limited amount of training data, BiLSTM-CRF suffers from low recall and the DGLSTM-CRF largely improves it on these two datasets. Using gold dependencies further significantly improves the precision and recall.

| | English | Chinese | Catalan | Spanish |
|---|---|---|---|---|
| BiLSTM-CRF | 88.98 | 79.20 | 78.46 | 80.59 |
| (Dependency LAS)† | (94.89) | (89.28) | (93.25) | (93.35) |
| DGLSTM-CRF (Predicted) | **89.64** | **79.59** | **82.37** | **83.92** |
| Improvement Δ | +0.66 | +0.39 | +3.91 | +3.33 |
| DGLSTM-CRF (Gold) | 89.88 | 79.92 | 84.22 | 87.56 |

Table 8: $F_1$ performance of DGLSTM-CRF with predicted dependencies against the best performing BiLSTM-CRF. †: LAS is label attachment score which is the metric for dependency evaluation.

| Model | Prec. | Rec. | $F_1$ |
|---|---|---|---|
| BiLSTM-CRF + ELMo ($L = 2$) | 89.14 | 88.59 | 88.87 |
| DGLSTM-CRF + ELMo ($L = 2$) | **89.59** | **90.17** | **89.88** |
| $-g(\cdot)$ = self connection | 89.17 | 90.08 | 89.62 |
| $-g(\cdot)$ = Concatenation | 89.43 | 90.09 | 89.76 |
| $-g(\cdot)$ = Addition | 89.24 | 89.78 | 89.50 |
| –w/o dependency relation | 88.92 | 89.99 | 89.46 |

Table 9: Ablation study of the DGLSTM-CRF model on the OntoNotes English dataset.

**Effect of Dependency Quality** To evaluate how the quality of dependency trees affect the performance, we train a state-of-the-art dependency parser (Dozat and Manning, 2017) using our training set and make prediction on the development/test set. We implemented the dependency parser using the AllenNLP package (Gardner et al., 2017). Table 8 shows the performance (LAS) of the dependency parser on four languages (i.e., OntoNotes English, OntoNotes Chinese, Catalan and Spanish) and the performance of DGLSTM-CRF against the best performing BiLSTM-CRF with ELMo. DGLSTM-CRF even with predicted dependencies is able to consistently outperform the BiLSTM-CRF on four languages. However, the performance is still worse than the DGLSTM-CRF with gold dependencies, especially on the Catalan and Spanish. Such results suggest that it is essential to have high-quality dependency annotations available for the proposed model.

**Ablation Study** Table 9 shows the ablation study of the 2-layer DGLSTM-CRF model on the OntoNotes English dataset. With self connection as interaction function, the $F_1$ drops 0.3 points. The model achieves comparable performance with concatenation as interaction function but $F_1$ drops about 0.4 points with the addition interaction function. We believe that the addition potentially leads to certain information loss. Without the depen-

| Dataset | Model | Entity Length | | | | | |
|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | ≥6 |
| **English** | BiLSTM-CRF | **91.8** | 88.5 | 83.4 | 84.0 | 75.4 | 76.0 |
| | DGLSTM-CRF | **91.8** | **90.1** | **85.4** | **87.0** | **80.8** | **78.7** |
| **Chinese** | BiLSTM-CRF | 81.2 | 74.3 | **73.1** | 62.8 | **70.3** | **57.5** |
| | DGLSTM-CRF | **82.2** | **75.5** | 71.8 | **64.1** | 58.5 | 41.1 |
| **Catalan** | BiLSTM-CRF | 80.5 | 81.0 | 75.8 | 56.1 | 45.0 | 38.4 |
| | DGLSTM-CRF | **85.4** | **85.1** | **84.1** | **78.9** | **60.9** | **59.3** |
| **Spanish** | BiLSTM-CRF | 84.2 | 81.1 | 81.0 | 53.3 | 53.3 | 37.1 |
| | DGLSTM-CRF | **89.3** | **87.4** | **90.8** | **74.1** | **67.7** | **64.4** |

Table 10: Performance of entities with different lengths on the four datasets: OntoNotes (English), OntoNotes Chinese, Catalan and Spanish.

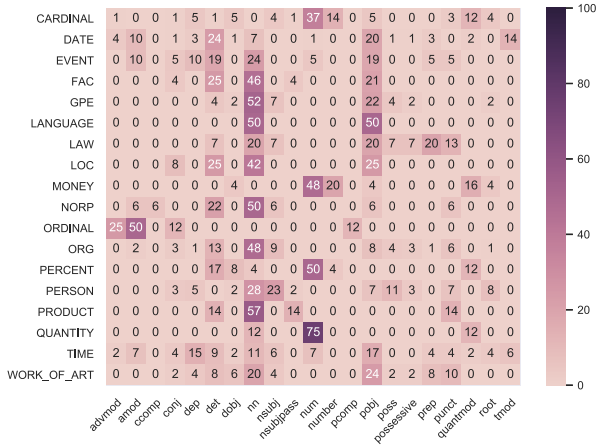


Figure 4: Correlations between the correctly predicted entities and the dependency relations.

dency relation embedding $\mathbf{v}_r$ in the input representation, the $F_1$ drops about 0.4 points.

## 5 Analysis

### 5.1 Effectiveness of Dependency Relations

To demonstrate whether the model benefits from the dependency relations, we first select the entities that are correctly predicted by the 2-layer DGLSTM-CRF model but not by the best performing baseline 2-layer BiLSTM-CRF on the OntoNotes English dataset. We draw the heatmap in Figure 4 based on these entities. Comparing Figure 3 and 4, we can see that they are similar in terms of the density. Both of them show consistent relationships between the entity types and the dependency relations. The comparison shows that the improvements partially result from the effect of dependency relations. We also found from our model's predictions that some entity types have strong correlations with the relation pairs on grandchild dependencies[14].

### 5.2 Entity with Different Lengths

Table 10 shows the performance comparison with different entity lengths on all datasets. As mentioned earlier, the dependencies as well as the grandchild relations allow our models to capture the long-distance interactions between the words. As shown in the table, the performance of entities with lengths more than 1 consistently improves with DGLSTM-CRF for all languages except Chinese. As we pointed out in the dataset statistics (Table 2), the number of entities that form subtrees in OntoNotes Chinese is relatively smaller compared to other datasets. The performance gain is more significant for entities with longer length on the other three languages. We found that, among the improvements of entities with length larger than 2 in English, 85% of them have long-distance dependencies and 30% of them have grandchild dependencies within the entity boundary. The analysis shows that our model that exploits the dependency tree structures is helpful for recognizing long entities.

[14]The corresponding heatmap visualization is provided in supplementary material.

## 6 Conclusions and Future Work

Motivated by the relationships between the dependency trees and named entities, we propose a dependency-guided LSTM-CRF model to encode the complete dependency tree and capture such relationships for the NER task. Through extensive experiments on several datasets, we demonstrate the effectiveness of the proposed model in improving the NER performance. Our analysis shows that NER benefits from the dependency relations and long-distance dependencies, which are able to capture the non-local interactions between the words.

As statistics shows that most of the entities form subtrees under the dependency trees, future work includes building a model for joint NER and dependency parsing which regards each entity as a single unit in a dependency tree.

## Ackowledgements

We would like to thank the anonymous reviewers for their constructive comments on this work. We would also like to thank Zhijiang Guo and Yan Zhang for the fruitul discussion. This work is supported by Singapore Ministry of Education Academic Research Fund (AcRF) Tier 2 Project MOE2017-T2-1-156.

## References


Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of COLING*.

Taylor Berg-Kirkpatrick, David Burkett, and Dan Klein. 2012. An empirical investigation of statistical significance in nlp. In *Proceedings of EMNLP-CoNLL*.

Wanxiang Che, Yijia Liu, Yuxuan Wang, Bo Zheng, and Ting Liu. 2018. Towards better UD parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*.

Jason P. C. Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association of Computational Linguistics*, 4:357–370.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

Ryan Cotterell and Kevin Duh. 2017. Low-resource named entity recognition with cross-lingual, character-level neural conditional random fields. In *Proceedings of IJCNLP*.

Alessandro Cucchiarelli and Paola Velardi. 2001. Unsupervised named entity recognition using syntactic and semantic contextual evidence. *Computational Linguistics*, 27(1):123–131.

Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*.

Marie-Catherine De Marneffe and Christopher D Manning. 2008. Stanford typed dependencies manual. Technical report, Stanford University.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL*.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of ACL*.

Timothy Dozat and Christopher D Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proceedings of ICLR*.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of ACL*.

Jenny Rose Finkel and Christopher D Manning. 2009. Joint parsing and named entity recognition. In *Proceedings of NAACL*.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. Allennlp: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP-OSS*.

Abbas Ghaddar and Phillippe Langlais. 2018. Robust lexical features for improved neural network named-entity recognition. In *Proceedings of COLING*.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of LREC*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Zhanming Jie, Aldrian Obaja Muis, and Wei Lu. 2017. Efficient dependency-guided named entity recognition. In *Proceedings of AAAI*.

Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of ICLR*.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of ACL*.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT*.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of EMNLP*.

Peng-Hsuan Li, Ruo-Ping Dong, Yu-Siang Wang, Ju-Chieh Chou, and Wei-Yun Ma. 2017. Leveraging linguistic structures for named entity recognition with bidirectional recursive neural networks. In *Proceedings of EMNLP*.

Xiao Ling and Daniel S Weld. 2012. Fine-grained entity recognition. In *Proceedings of AAAI*.

Jingchen Liu, Minlie Huang, and Xiaoyan Zhu. 2010. Recognizing biomedical named entities using skip-chain conditional random fields. In *Proceedings of the Workshop on BioNLP*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of ACL*.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of ACL: System Demonstrations*.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of EMNLP*.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *Proceedings of ACL*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018a. Deep contextualized word representations. In *Proceedings of NAACL*.

Matthew Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. 2018b. Dissecting contextual word embeddings: Architecture and representation. In *Proceedings of EMNLP*.

Matthew Peters, Sebastian Ruder, and Noah A Smith. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks. *arXiv preprint arXiv:1903.05987*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using ontonotes. In *Proceedings of CoNLL*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Proceedings of Joint EMNLP-CoNLL: Shared Task*.

Lance A Ramshaw and Mitchell P Marcus. 1999. Text chunking using transformation-based learning. In *Proceedings of Third Workshop on Very Large Corpora*.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of CoNLL*.

Marta Recasens, Lluís Màrquez, Emili Sapena, M Antònia Martí, Mariona Taulé, Véronique Hoste, Massimo Poesio, and Yannick Versley. 2010. Semeval-2010 task 1: Coreference resolution in multiple languages. In *Proceedings of the 5th International Workshop on Semantic Evaluation*.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging. In *Proceedings of EMNLP*.

Sunita Sarawagi and William W Cohen. 2004. Semi-markov conditional random fields for information extraction. In *Proceedings of NeurIPS*.

Ryohei Sasano and Sadao Kurohashi. 2008. Japanese named entity recognition using structural natural language processing. In *Proceedings of IJCNLP*.

Emma Strubell, Patrick Verga, David Belanger, and Andrew McCallum. 2017. Fast and accurate entity recognition with iterated dilated convolutions. In *Proceedings of EMNLP*.

Charles Sutton and Andrew McCallum. 2004. Collective segmentation and labeling of distant entities in information extraction. In *Proceedings of the ICML Workshop on Statistical Relational Learning and Its Connections to Other Fields*.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al. 2013. Ontonotes release 5.0 ldc2013t19. *Linguistic Data Consortium*.

Yue Zhang and Jie Yang. 2018. Chinese ner using lattice lstm. In *Proceedings of ACL*.

Yuhao Zhang, Peng Qi, and Christopher D Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of EMNLP*.

# Cross-Cultural Transfer Learning for Text Classification

**Dor Ringel**[1] **Gal Lavee**[2] **Ido Guy**[2] **Kira Radinsky**[1]
[1]Technion - Israel Institute of Technology
[2]eBay Research
{dorringel,kirar}@cs.technion.ac.il
idoguy@acm.org glavee@ebay.com

## Abstract

Large training datasets are required to achieve competitive performance in most natural language tasks. The acquisition process for these datasets is labor intensive, expensive, and time consuming. This process is also prone to human errors. In this work, we show that cross-cultural differences can be harnessed for natural language text classification. We present a transfer-learning framework that leverages widely-available unaligned bilingual corpora for classification tasks, using no task-specific data. Our empirical evaluation on two tasks – formality classification and sarcasm detection – shows that the cross-cultural difference between German and American English, as manifested in product review text, can be applied to achieve good performance for formality classification, while the difference between Japanese and American English can be applied to achieve good performance for sarcasm detection – both without any task-specific labeled data.

## 1 Introduction

The collection of large, task-specific labeled datasets poses a major challenge to the application of supervised text classification in many domains. Acquiring these datasets is expensive, time-consuming, and error-prone.

In this work, we propose leveraging bilingual datasets for classification tasks. The large-scale availability of such datasets counteracts the low availability of task-specific labeled data. The cross-cultural differences expressed in these bilingual datasets can be learned and then transferred for text classification tasks with no labeled data. Our work, thus, extends the idea of *distant supervision* (Mintz et al., 2009) to tasks for which no relevant large-scale curated datasets exist.

Culture was defined by Hofstede et al. (2010) as "*the collective programming of the mind, which distinguishes the members of one group of people from another*." Shweder et al. (2006) defines membership in a cultural group as "*thinking and acting in a certain way, in the light of particular goals, values, pictures of the world*." Based on these definitions we can reasonably expect members of a specific cultural group to think and act in ways that are distinct from others. For example, Hedderich (2010), who investigated cross-cultural differences at the workplace, found that the overall interaction of employees in Germany is more *formal* than their American counterparts. House (1997) found that "*German subjects tended to interact in ways that were more direct, more explicit, more self-referenced, and more content-oriented*".

Similarly, several works have studied cross-cultural differences between Japanese and English (both American and British variants) (Koga and Pearson, 1992; Minami, 1994; Martinsons, 2001; Adachi, 2010). Specifically, Adachi (1996) and Ziv (1988) investigated sarcasm in Japanese and the latter showed that American students are more sarcastic than their Japanese counterparts.

The main thesis underlying this work is that texts composed by members of a specific cultural group are distinct from those composed by members of another. Observational studies on cultural differences, such as those described in the examples above, allow us to identify the high-level semantics of these differences.

We present a transfer learning algorithm which learns a model encapsulating cross-cultural differences from bilingual data, and applies the learned model to text classification tasks. We study this idea using two natural language classification tasks: (1) *Formality classification* – by learning the difference between the writing style of Germans compared to Americans; and (2) *Sarcasm detection* – by learning the difference between the writing style of Japanese compared to Americans.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3873–3883,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

Our approach requires only a bilingual dataset without the need for any alignments, special labels, or task-specific training data. Our empirical evaluation demonstrates that such cross-cultural distinction can be successfully transferred to those tasks. We present an algorithm that, given two unaligned corpora of texts written in two languages, transforms the two document classes into a common representation. A binary classifier is then trained to distinguish between representation of documents originating from one language as compared to the other. The classifier can be subsequently applied to a binary text classification task. For example, if the document is deemed by the American-Japanese classifier as American, we will infer that the text is sarcastic.

We study various representations of both words and documents and present a novel document representation algorithm adapted to our task. Our empirical results suggest that our transfer-learning approach based on cross-cultural differences achieves comparable performance to direct learning approaches trained on task-specific labeled data. Additional results demonstrate the contribution of our proposed representation approach.

The contributions of this work can be summarized as follows: (1) We propose a transfer learning framework to enable text classification using cross-cultural differences learned on bilingual data; (2) We propose a representation scheme for documents, designed for the task of text classification based on bilingual data; (3) We perform an empirical evaluation of our approach and contribute our labeled datasets to the community.

## 2 Related Work

Cross-cultural differences have been studied extensively in the social science literature (Shweder, 1991; Weber and Hsee, 1998; Liu and McClure, 2001; Boroditsky et al., 2003; Yin et al., 2011; Hong et al., 2012). Specifically, Paul and Girju (2009); Garimella et al. (2016) explored differences through language analysis.

Text classification approaches have traditionally used distributed representation of texts (e.g. TF-IDF) and applied supervised models to these representations (Joachims, 1998; Wang and Manning, 2012). More recent work has pursued improved representations (Joulin et al., 2017) and novel neural architectures (Conneau et al., 2017).

**Algorithm 1** Cross-Cultural Transfer Learning

**Input:** $C_0$ (documents from language $L_0$)
**Input:** $C_1$ (documents from language $L_1$)
**Input:** $T$ (transformation method)
**Input:** $D_t$ (binary classification task dataset)
**Output:** $\hat{\mathbf{y}}_t$ (predictions)

$C_0^T, C_1^T \leftarrow \text{TRANSFORM\_DOCUMENTS}(T, C_0, C_1)$
$h \leftarrow \text{TRAIN\_CLASSIFIER}(C_0^T, C_1^T)$
$D_t^T \leftarrow \text{TRANSFORM\_DOCUMENTS}(T, D_t)$
$\hat{\mathbf{y}}_t \leftarrow \text{TRANSFER\_LEARNING}(h, D_t^T)$

Transfer learning in natural language processing has recently moved beyond pre-trained word embeddings (Howard and Ruder, 2018) to more advanced approaches (Peters et al., 2018; Radford et al., 2018).

Cross-lingual word embedding seeks a mapping between embedding spaces representing different languages (Ruder et al., 2019). This goal is most often achieved by training monolingual word embeddings for multiple languages independently, and then learning a transformation between them using either supervised or unsupervised methods.

Cross-lingual language understanding (XLU) is an area of research that has recently gained much attention. Related methods seek to learn a joint-embedding space for multiple languages (Conneau et al., 2018; Devlin et al., 2019) .

This work, to the best of our knowledge, is the first to leverage cross-cultural differences in bilingual text data to perform inference on monolingual data.

## 3 Cross-Cultural Transfer Learning

In this section we describe our approach for text classification tasks such as formality classification and sarcasm detection, using transfer learning from bilingual data. A supervised learning model is trained to differentiate between a pair of languages based on their cross-cultural differences, as manifested in the available text data. Specifically, the training data contains a corpus of text documents collected from two distinct languages, with the assumption that the language of each document is known. Note that, the bilingual data needed for this approach is coarsely taken from the same domain. However, the texts need not be aligned beyond this coarse level. That is, full alignment between specific documents across the two languages is not required. Algorithm 1 summarizes the steps of our method.

### 3.1 Formal Framework

Given two collections of documents from languages $L_0$ and $L_1$ denoted as $C_0$ and $C_1$, respectively, the following are the steps to applying our framework:

1. Using a transformation, denoted $T$, transform the two document classes into a common representation. The resulting transformed collections of documents are denoted $C_0^T$ and $C_1^T$, respectively.

2. Train a binary classifier, denoted $h$, to discriminate document representations in $C_0^T$ from those in $C_1^T$. That is, the classifier is trained to classify representations of documents as originating from either $L_0$ or $L_1$.

3. Given a binary classification task dataset, denoted $D_t$, which consists of text documents (with a known language), transform all texts in $D_t$ using transformation $T$, to obtain a set of document representations denoted $D_t^T$.

4. Finally, apply the classifier $h$, trained on the bilingual data, to the document representations in $D_t^T$ to obtain predictions for each member of this set, denoted $\hat{\mathbf{y}}_t$.

Figure 1 summarizes the process.

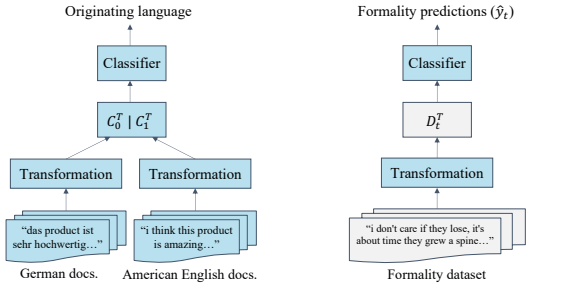


(a) Train on transformed bilingual data (b) Apply to a binary classification task

**Figure 1: Illustration of the cross-cultural transfer learning framework through the example of the formality classification task.** (a) The classifier is trained on document representations of the German-American bilingual data ($C_0^T \mid C_1^T$) to distinguish documents as originating from either German or American English ($L_0$ or $L_1$). (b) The trained classifier is applied to a formality task.

### 3.2 Transformations

The effectiveness of the framework described above depends in large part on the choice of transformation $T$. In general, transformation $T$ transforms documents from languages $L_0$ and $L_1$ into a common representation. A well chosen transformation should enable the supervised learning in subsequent steps to focus on cross-cultural differences between the original documents rather than superficial differences between the texts.

In our construction, we used compound transformations composed of the following parts: (1) Translation of the texts to a common language (Section 3.2.1); (2) Mapping the tokens in the text to word embeddings (Section 3.2.2); (3) Combining the word embeddings to form document-level vector representations (Section 3.2.3).

#### 3.2.1 Translation

The first part of our document transformation involves translating the documents. We chose to translate documents from $L_0$ to $L_1$, so that documents authored in language $L_1$ (those documents denoted $C_1$ above), require no translation.

The motivation for choosing $L_1$ as the "target" language may involve the reliability of the translation to this language, the availability of high quality word embeddings in this language, or the fore-knowledge that downstream tasks will involve datasets written in this language.

Under this framework we evaluate two different translation approaches: (1) A state of the art machine translation system we denote as **MT**. Following preliminary experiments with both the method by Vaswani et al. (2017) and with Google cloud translation[1], we focused on the latter as it achieved better results. (2) A word-by-word nearest-neighbor search, denoted as **NN**, where we use the method of Lample et al. (2018), adapted to our task. These choices represent a complexity-accuracy trade-off. We expect the **MT** system to give high-quality translation, but at the cost of higher complexity. Conversely, the **NN** approach is simple to implement, but may produce lower-quality translations.

In order to implement **NN** some notion of distance between words in language $L_0$ and words in $L_1$ is needed. We used cross-lingual word embeddings for this purpose. For a given language pair, the embedding of the joint vector space representation of many words from both languages is known. For implementing nearest neighbor search for a particular word in $L_0$, we simply compare the embedding vector representation of this word

[1] https://cloud.google.com/translate/docs

with all known embedding vectors of words in language $L_1$, and select the one that maximizes their similarity, as suggested by Lample et al. (2018) (and done efficiently by employing the method of Johnson et al. (2019)). For translating German to English such an embedding dataset is available for download.[2] For Japanese to English, we trained our own such model (following Lample et al. (2018)), as this language pair is not available for download. Note that the embeddings used in this step may or may not be related to the embeddings described in the next section.

#### 3.2.2 Word Embeddings

Once our documents are converted to a single language using one of the translation methods above, we represent each word in the document as an embedding vector, which is a dense distributed version of the corresponding word.

We experiment with several types of word embeddings, including: pre-trained embedding models trained on large corpora of general English language text, publicly available for download (Word2Vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014) and FastText (Bojanowski et al., 2017)).

#### 3.2.3 Document Representation

After applying translation and mapping tokens to word embeddings, our documents are represented as a variable-length list of vectors (each corresponding to word). In the following we examine various methods for combining these vectors into a single uniform length representation, and derive a document representation scheme that uses label information to achieve better discrimination, which we conjecture increases the ability to encode the cross-cultural difference.

The methods we explore produce weighted combinations of the word vectors in the document, normalized to unit length. A model that captures the intuition that text is composed of both local and global context is proposed by Arora et al. (2017). Applying maximum likelihood estimation to this model obtains the representation:

$$\mathbf{c}_d = \sum_i^n \frac{a}{\mathcal{U}(w_i) + a} \mathbf{v}_{w_i} \tag{1}$$

where $\mathbf{c}_d \in \mathbb{R}^k$ is the resulting document representation vector and $\mathbf{v}_{w_i} \in \mathbb{R}^k$ is the embedding vector corresponding to the $i$-th of $n$ words in document $d$; $\mathcal{U}(w)$ denotes the unigram distribution; and $a>0$ is a free parameter that controls smoothing. This equation expresses the intuitive approach of averaging the word embeddings, each weighted inversely to the word's frequency. Thus, very popular words are down-weighted. This is also related to an inverse document frequency (IDF) weighting proposed in De Boom et al. (2016).

In our application we seek a document embedding to assist in solving a binary classification task. In other words, we seek a context vector $\mathbf{c}_d$ that is most discriminative, taking advantage of additional information about some binary label. More specifically, we assume that we have a conditional model for the probability of each token given the class.

Formalizing this idea, we seek an embedding that will maximize the *expected log-probability of a document*:

$$\mathbb{E}\left[\log\left(P\left(L|w_1, w_2, \ldots, w_n\right)\right)\right] = \mathbb{E}\left[\log \textstyle\prod_{i=1}^n P\left(w_i \mid \mathbf{c}_d, L\right)\right] + C$$

where we have used Bayes rule and $C$ is constant w.r.t $\mathbf{c}_d$. As the notation suggests, we now have a generative model for each class:

$$P\left(w_i \mid \mathbf{c}_d, L_0\right) = \alpha \cdot P\left(w_i \mid L_0\right) + (1-\alpha) \cdot \left(\frac{\exp\left(\mathbf{v}_{w_i}^\top \mathbf{c}_d\right)}{Z}\right)$$
$$P\left(w_i \mid \mathbf{c}_d, L_1\right) = \alpha \cdot P\left(w_i \mid L_1\right) + (1-\alpha) \cdot \left(\frac{\exp\left(\mathbf{v}_{w_i}^\top \mathbf{c}_d\right)}{Z}\right)$$

where $P\left(w_i \mid L_0\right)$ and $P\left(w_i \mid L_1\right)$ are the conditional unigram models, and $Z$ denotes a normalization constant.

Focusing on the probability of a single token:

$$\begin{aligned} f\left(\mathbf{c}_d\right) &= \mathbb{E}\left[\log P\left(w_i \mid \mathbf{c}_d, L\right)\right] \\ &= \frac{1}{2}\left[\log P\left(w_i \mid \mathbf{c}_d, L_0\right) + \log P\left(w_i \mid \mathbf{c}_d, L_1\right)\right] \end{aligned} \tag{2}$$

under the assumption that $P\left(L_0\right) = P\left(L_1\right) = \frac{1}{2}$.

This expression has the following gradient (w.r.t. $\mathbf{c}_d$):

$$\begin{aligned} &\nabla f\left(\mathbf{c}_d\right) = \\ &\frac{1}{2} a \cdot e_{i,d} \left[\frac{P\left(w_i \mid L_0\right) + P\left(w_i \mid L_1\right) + 2a e_{i,d}}{\left(P\left(w_i \mid L_0\right) + a e_{i,d}\right) \cdot \left(P\left(w_i \mid L_1\right) + a e_{i,d}\right)}\right] \mathbf{v}_{w_i} \end{aligned} \tag{3}$$

where $a = \frac{(1-\alpha)}{\alpha Z}$ and $e_{i,d} = \exp\left(\mathbf{v}_{w_i}^\top \mathbf{c}_d\right)$. Note that when we evaluate Equation (3) with $\mathbf{c}_d = 0$ the expression becomes:

$$\nabla f\left(0\right) = \frac{1}{2} a \left[\frac{P(w_i|y) + P(w_i|L_1) + 2a}{(P(w_i|L_0)+a)\cdot(P(w_i|L_1)+a)}\right] \mathbf{v}_{w_i}$$

[2] https://github.com/facebookresearch/MUSE

Thus, maximizing the Taylor approximation $f(\mathbf{c}_d) \approxeq f(0) + \nabla f(0)^\top \mathbf{c}_d$ w.r.t. $\mathbf{c}_d$ (following Arora et al. (2017)) yields the following estimator:

$$\mathbf{c}_d = \frac{1}{2} \sum_{i=1}^{n} a \left[ \frac{P(w_i|L_0)+P(w_i|L_1)+2a}{(P(w_i|L_0)+a)\cdot(P(w_i|L_1)+a)} \right] \mathbf{v}_{w_i} \quad (4)$$

Examining the expression above, we can see that for a fixed value of $a$, the numerator of the of the ratio grows with the frequency of $w_i$ in *either* language $L_0$ or $L_1$. The denominator of the ratio grows when $w_i$ is frequent in *both* languages. Hence, the weighting scheme gives more weight to vectors corresponding to words that are frequent in either language, but not both. Further, similarly to Equation 1, the expression gives a larger weight to words that have low frequency in either language.

Note that the representations above define an *unnormalized* quantity, so that the absolute values of the vector weights are less important than their relative values. The final document representation is given by $\frac{\mathbf{c}_d}{||\mathbf{c}_d||}$.

## 4 Evaluation

In order to evaluate our setup, we consider the application of the above framework to two binary text classification tasks: formality classification and sarcasm detection. For each such task we require a bilingual dataset containing texts drawn from two languages for training and another (one or more) dataset to use for evaluation. Table 1 summarizes the characteristics of both the bilingual and evaluation datasets.

### 4.1 Formality Classification Task

The formality classification task (Heylighen and Dewaele, 1999) aims to classify a document as either formal or informal. For evaluating this task we used the following datasets:

**Amazon Product Descriptions and Reviews** Product descriptions and their corresponding user reviews from the Motors and Fashion categories of Amazon e-commerce website (He and McAuley, 2016). We consider the description texts to be examples of formal writing and the reviews to be examples of informal writing (Novgorodov et al., 2019). We filter documents that are not in English or shorter than 10 terms. We then sample one review per description document to result with a balanced dataset in terms of class labels.

**New York Times Article snippets and Comments** Article-snippets and their corresponding user comments from the New York Times (Kesarwani, 2018). We consider article-snippet texts to be examples of formal writing and the comment texts to be examples of informal writing. We focus on the news section, and filter documents shorter than 10 terms. We then sample one comment per article-snippet to result with a balanced dataset in terms of class labels.

**Formality in Online Communication** Texts annotated for formality, originating from four types of online communication: News, Blog, Email, and community question answering forums (denoted as Answers) (Lahiri, 2015; Pavlick and Tetreault, 2016). The mean formality score for each document (across all annotators) ranges from -3 (very informal) and 3 (very formal). To make this dataset suitable for our binary classification framework, we only consider documents with a mean formality score in the highest and lowest $20^{th}$ percentile per communication type as formal and informal, respectively.

**Surrogate Bilingual Dataset for Formality** Based on prior research by Hedderich (2010), who investigated cross-cultural differences between Germans and Americans and identified the former to be more formal, we selected German and American English (Ger-Am) as the language pair to serve as a surrogate for the formality task. The German language is chosen to represent the formal class and the American English language is chosen to represent the informal class. We use the eBay Fashion product review bilingual data to learn the cross-culture differences. The data is extracted from the German and American sites of eBay e-commerce website, and the reviews originate from 24 sub-categories (e.g, "Jewelry").

### 4.2 Sarcasm Detection Task

The sarcasm detection task (Tepperman et al., 2006) aims to classify a document as either non-sarcastic or sarcastic. For evaluating the sarcasm detection task, we used the following dataset:

**Sarcasm Corpus V2** The Sarcasm Corpus version 2 (Oraby et al., 2016)[3] includes response text from quote-response pairs annotated for sarcasm. Each sample consists of the form of sarcasm (Gen for general sarcasm, RQ for rhetorical question or

[3] https://nlds.soe.ucsc.edu/sarcasm2

| Dataset | size | mean | std. | min. | median | max |
|---|---|---|---|---|---|---|
| *formality classification* | | | | | | |
| Amazon Motors | 786 | 78.48 | 42.98 | 10 | 72 | 197 |
| Amazon Fashion | 866 | 68.98 | 45.07 | 10 | 58 | 258 |
| New York Times | 3,226 | 27.38 | 5.370 | 10 | 27 | 49 |
| Answers | 858 | 12.91 | 2.585 | 10 | 13 | 48 |
| Blog | 480 | 29.05 | 16.72 | 11 | 25 | 201 |
| Email | 366 | 26.46 | 18.92 | 10 | 22 | 230 |
| News | 622 | 28.73 | 14.82 | 11 | 26 | 248 |
| **eBay Ger-Am** | 11,576 | 33.71 | 25.47 | 10 | 26 | 188 |
| *Sarcasm detection* | | | | | | |
| Sarcasm Gen | 3,260 | 47.54 | 34.72 | 10 | 35 | 241 |
| Sarcasm RQ | 786 | 79.82 | 42.45 | 13 | 76 | 188 |
| Sarcasm Hyp | 582 | 62.76 | 40.87 | 11 | 53 | 195 |
| **Amazon Jp-Am** | 37,048 | 57.61 | 36.13 | 10 | 45 | 224 |

**Table 1:** The size of the datasets and the characteristics of the length of documents in the datasets. The bilingual datasets used to learn the cross-cultural difference during training for each of the tasks are marked in bold.

Hyp for hyperbole), a class label (either sarcastic or non-sarcastic), a quote text, and a response text (the text annotated for sarcasm).

**Surrogate Bilingual Dataset for Sarcasm** Based on prior research on cross-cultural differences between Japanese and American (Koga and Pearson, 1992; Minami, 1994; Martinsons, 2001; Adachi, 2010), and specifically Adachi (1996) and Ziv (1988) who addressed sarcasm in Japanese, we selected Japanese and American English (Jp-Am) as the language pair to serve as a surrogate for the sarcasm task. The Japanese language is chosen to represent the non-sarcastic class and the American English language is chosen to represent the sarcastic class. We use the Amazon Japanese and American reviews as the bilingual data to learn the cross-culture difference. The data originates from the Japanese and American marketplaces of the Amazon Customer Reviews Dataset [4], and the reviews originate from 39 categories (e.g. "Books").

### 4.3 Surrogate Bilingual Datasets Pre-processing

To allow the classifier to learn cross-cultural differences during training, and before transferring to the specific task, we seek to remove from the bilingual datasets used for training any other sources of information that may incidentally distinguish between the two language corpora: (1) We filter documents containing fewer than 10 or more than 250 terms; (2) We apply language detection (Shuyo, 2010) to remove documents in languages outside the surrogate languages; (3) We sample the same number of documents per each sub-category, document length and review-rating. This enables us to remove any bias towards a topic; (4) We sample the same number of documents from both languages to result with a balanced dataset in terms of class labels. We draw the reader's attention to the fact that the documents in the bilingual datasets are plain reviews which have not gone through any formality classification or sarcasm detection as part of their pre-processing. We publicly release the surrogate datasets.[5]

### 4.4 Classification Algorithms

We experiment with both Logistic Regression and XGBoost classifiers (McCullagh, 2018; Chen and Guestrin, 2016), and report the results of the former as it yielded superior performance. For Logistic Regression, we tune both the regularization norm ($L1$ or $L2$), and the regularization coefficient ($C$). For XGBoost, we tune the learning-rate, maximum tree depth, number of rounds (trees), minimum loss reduction required for partitioning, sub-sample ratio of features per tree, and both the $L1$ and $L2$ regularization coefficients. The configurations empirically chosen (based on grid search experiments on the validation set) were (1) $L2$ regularization with $C = 10^4$ for models trained on bilingual data; (2) $L2$ regularization with $C = 1$ for models trained on task-specific data. We also experimented with the following neural network architectures for the classification: LSTM-RNN (Hochreiter and Schmidhuber, 1997), HAN (Yang et al., 2016), QRNN (Bradbury et al., 2017), and VDCNN (Conneau et al., 2017). However, these models did not achieve any substantial performance gain to justify their additional complexity.

### 4.5 Evaluation Metrics

We report the AUC scores of stratified, five-fold cross-validation experiments on the formality classification and sarcasm detection tasks. For each fold of each experiment, the evaluation dataset is partitioned into train, validation and test splits. Task-specific models use all three splits for training, hyper-parameter tuning and evaluation, respectively, while models trained on bilingual data only use the test split for evaluation (so

[4] https://s3.amazonaws.com/amazon-reviews-pds/readme.html

[5] https://github.com/dorringel/CCTL

| Dataset | task-specific | our-approach | Δ(%) |
|---|---|---|---|
| *formality classification* | | | |
| Amazon Motors | 98.75 | 95.61 | -3.14 |
| Amazon Fashion | 97.97 | 91.67 | -6.30 |
| New York Times | 90.57 | 84.30 | -6.27 |
| Answers | 95.11 | 91.16 | -3.95 |
| Blog | 89.24 | 79.29 | -9.95 |
| Email | 96.16 | 88.91 | -7.25 |
| News | 85.42 | 78.04 | -7.38 |
| *Sarcasm detection* | | | |
| Sarcasm Gen | 80.38 | 76.99 | -3.39 |
| Sarcasm RQ | 81.04 | 77.34 | -3.70 |
| Sarcasm Hyp | 71.03 | 65.32 | -5.71 |

**Table 2:** AUC performance for our approach in comparison to the task-specific models directly trained on the evaluation datasets (results are comparable to supervised state-of-the-art). The final column reports the percent difference between the two methods.

| Dataset | NN | MT | Δ(%) |
|---|---|---|---|
| *formality classification* | | | |
| Amazon Motors | 91.76 | **95.61*** | 3.85 |
| Amazon Fashion | 84.59 | **91.67*** | 7.08 |
| New York Times | 81.02 | **84.30*** | 3.28 |
| Answers | 88.60 | **91.16*** | 2.56 |
| Blog | 78.01 | **79.29*** | 1.28 |
| Email | 88.30 | **88.91*** | 0.61 |
| News | 77.70 | **78.04*** | 0.34 |
| *Sarcasm detection* | | | |
| Sarcasm Gen | 65.00 | **76.99*** | 11.99 |
| Sarcasm RQ | 63.70 | **77.34*** | 13.64 |
| Sarcasm Hyp | 54.13 | **65.32*** | 11.19 |

**Table 3:** AUC performance for **MT** translation method in comparison to **NN** translation method. The best result per dataset is marked in bold, "*" indicates statistically significant difference of the leading method from the other method. The final column reports the percent difference between the two methods.

that both types of models are evaluated on the same test set). Statistical significance is measured using one-way paired $t$-test (Casella and Berger, 2002) with $p<0.01$.

## 5 Results and Discussion

Our empirical evaluation explores the following main research questions: (1) How effective is the approach of using bilingual data as described in Section 3 on our chosen tasks? (2) How does the choice of transformation impact the performance of the approach? Specifically, we consider the choice of machine translation and document representation.

**Comparison to Models Trained on Task-specific Data** In order to determine the overall effectiveness of our approach as described in Section 3, we consider the result of training a model using task-specific labeled data. We expect that if our approach is effective, we can achieve performance close to that of a model trained on this task-specific data. Table 2 reports the cross-validation AUC comparing our approach to the model trained on task-specific data. The table considers both tasks across the various evaluation datasets described in Section 4. The final column reports the percent difference between the two methods.

Examining the table, we observe that our method is within 10% of the AUC of the model trained on task-specific data, on all datasets and tasks, and as close as 4% on four of the datasets. Our model was trained entirely on bilingual data with no examples from the task it was evaluated on. The comparison with a model that was trained on data specific to the target task yields only a small improvement over our method. The comparable performance supports our thesis that cross-cultural information which exists within the bilingual data can be leveraged to achieve performance that is nearly equal to that achieved by collecting expensive task-specific labels.

**Effect of Translation Component** Table 3 considers the effect of the translation component of the compound transformation described in Section 3.2.1. Recall we considered two approaches for translation which represent a complexity-accuracy trade-off. We expect the **MT** approach to provide a higher quality translation, but it is substantially more complex to train and deploy. Conversely, the **NN** approach is simple to deploy at the cost of overall translation quality. The table illustrates that the difference in translation quality does have an impact on the down-stream task performance. However, the magnitude of the impact varies widely across the datasets in our experiments. For the formality task, the <8% gap between the methods may justify the reduced complexity of the **NN** approach. It is interesting to note that the model performs well even with a very simple word-by-word translation scheme.

**Effect of Document Representation** Table 4 considers the effect of the document representation component of the transformation discussed in Section 3.2.3 across the tasks, datasets, and translation methods. The first result column shows the

| Dataset | AVG | IDF | LANG |
|---|---|---|---|
| *formality classification* | | | |
| Amazon Motors$_{MT}$ | 95.26 | 95.47 | **95.61** |
| Amazon Motors$_{NN}$ | 76.93 | 86.46 | **91.76*** |
| Amazon Fashion$_{MT}$ | 90.31 | 91.19 | **91.67** |
| Amazon Fashion$_{NN}$ | 61.30 | 75.83 | **84.59*** |
| New York Times $_{MT}$ | 82.45 | 82.27 | **84.30** |
| New York Times $_{NN}$ | 70.64 | 75.14 | **81.02*** |
| Answers $_{MT}$ | 87.88 | 87.82 | **91.16*** |
| Answers $_{NN}$ | 80.10 | 80.60 | **88.60*** |
| Blog $_{MT}$ | 78.63 | 77.89 | **79.29** |
| Blog $_{NN}$ | 65.56 | 68.80 | **78.01*** |
| Email $_{MT}$ | 87.22 | 88.21 | **88.91** |
| Email $_{NN}$ | 73.80 | 77.70 | **88.30*** |
| News $_{MT}$ | 77.71 | 77.84 | **78.04** |
| News $_{NN}$ | 65.30 | 69.40 | **77.70*** |
| *Sarcasm detection* | | | |
| Sarcasm Gen $_{MT}$ | 76.68 | 76.82 | **76.99** |
| Sarcasm Gen $_{NN}$ | 61.40 | **65.00*** | 62.10 |
| Sarcasm RQ $_{MT}$ | 76.81 | **77.34** | 77.16 |
| Sarcasm RQ $_{NN}$ | 61.13 | **63.70*** | 60.85 |
| Sarcasm Hyp $_{MT}$ | 64.31 | 65.12 | **65.32** |
| Sarcasm Hyp $_{NN}$ | 50.94 | 53.32 | **54.13** |

**Table 4:** AUC performance for various representation methods. AVG refers to a simple unweighted average of word vectors, IDF refers to a document-frequency-based weighting according to equation 1. LANG refers to a weighting scheme that takes the language of origin into consideration, based on Equation 4. The best result per dataset is marked in bold, "*" indicates statistically significant difference of the leading method from both other methods.

performance of a simple unweighted average of the word vectors. The second result column shows a document-frequency-based weighting according to Equation 1. The third result column shows the performance of a weighting scheme that takes the language of origin into consideration, based on Equation 4.

Examining the table, we observe that using a non-uniform weighting scheme generally gives improved performance over the naive unweighted baseline. The effect of the document representation method is significant when paired with the **NN** translation approach. Thus, the translation and document representation components of the compound transformation are complementary in the sense that when translation is of high quality, a naive document representation suffices. Conversely, when translation quality is sub-optimal, the choice of document representation can significantly impact performance.

We also studied the effect of the choice of word embeddings as discussed in Section 3.2.2, but there were no statistical significant differences.

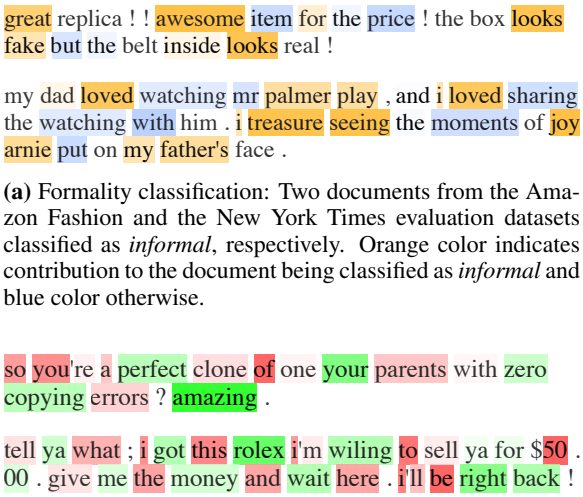


**(a)** Formality classification: Two documents from the Amazon Fashion and the New York Times evaluation datasets classified as *informal*, respectively. Orange color indicates contribution to the document being classified as *informal* and blue color otherwise.

**(b)** Sarcasm detection: Two documents from the Sarcasm Corpus evaluation dataset classified as *sarcastic*. Green color indicates contribution to the document being classified as *sarcastic* and red color otherwise.

**Figure 2:** Documents from the evaluation datasets of both the formality classification, and sarcasm detection tasks and their corresponding LIME interpretation (Ribeiro et al., 2016). Color intensity is in proportion with the word's contribution to the classification according to the LIME algorithm.

**Qualitative Examples** To better understand what the models trained on bilingual data actually learn, we utilize the LIME algorithm (Ribeiro et al., 2016) to attain each word's gravity to the classification of the document. Figure 2 provides visual explanations of documents from the evaluation datasets of both the formality classification, and sarcasm detection tasks and their corresponding LIME interpretation. Specifically, Figure 2a showcases documents from the Amazon Fashion and the New York Times datasets classified as *informal*, respectively, while Figure 2b showcases documents from the Sarcasm Corpus dataset classified as *sarcastic*. Examining the formality visualization (2a), we observe that words that affect the formality of the document, such as `awesome` in the first example and `treasure` in the second one, are indeed assigned with a higher weight by the LIME algorithm. Similarly for the sarcasm visualization (2b), we observe that words that contribute to the document being sarcastic, such as `perfect` and `amazing` in the first example, and `rolex` and `right` followed immediately by `back` in the second example, are indeed assigned with a higher weight by the LIME algorithm.

**Most Discriminating Words** To further demonstrate qualitatively the formality discriminating in-

| German (Translated) | | American English | |
|---|---|---|---|
| long | original | love | cute |
| high | broken | one | another |
| well | material | loved | going |
| good | handy | excellent | lovely |
| even | narrow | amazing | overpowering |
| cut | time | perfect | wanted |
| quality | clear | wearing | wonderful |
| cheap | ok | favorite | incredible |
| chic | flowery | like | overwhelming |
| hand | fake | comfortable | back |
| alternative | optimal | new | tiny |
| woody | waterproof | terrible | glad |

**Table 5:** Most discriminative unigrams between German and American English according to their $wc(w_i, L_j)$ scores, based on the eBay Ger-Am product reviews dataset.

formation latent in German and American data, we seek the words that are most helpful in discriminating between the two languages. This notion is made concrete using the relative contribution of word $w_i$ to the Kullback-Leibler divergence (Berger and Lafferty, 2017) between the languages, applied to all words, separately for each language:

$$wc\left(w_i, L_j\right) = P\left(w_i \mid L_j\right) \log\left(\frac{P(w_i|L_j)}{P\left(w_i|L_{(1-j)}\right)}\right)$$

Table 5 presents the unigrams that achieve the highest values for the expression above as computed on the eBay bilingual dataset of German and American English reviews. We can see that terms conveying information, such as `long`, `high`, `quality`, `cheap`, `woody`, and `clear` are more common in translated German documents, while terms conveying emotion, such as `love`, `amazing`, `favorite`, `wonderful`, `terrible`, and `overwhelming` are more common in American English documents.

## 6 Conclusions

In this work, we show that cross-cultural differences can be harnessed for natural language text classification. We present a transfer-learning framework that leverages bilingual corpora for classification tasks using no task-specific data, and evaluate its performance on formality classification and sarcasm detection tasks. We show that our approach achieves comparable performance to task-specific methods directly trained on the two tasks, and propose a document representation scheme designed for bilingual training data. Such a representation can improve performance substantially when a low-quality translation method is used. In future work, we would like to generalize our approach to the multilingual case of multiple languages and a multi-class target task, explore applications beyond text classification, and study transformations that eliminate the need for a translation system.

## References


Takanori Adachi. 1996. Sarcasm in japanese. *Studies in Language. International Journal sponsored by the Foundation "Foundations of Language"*, 20(1):1–36.

Yumi Adachi. 2010. Business negotiations between the americans and the japanese. *Global Business Languages*, 2(1):4.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *Proceedings of the International Conference on Learning Representations*.

Adam Berger and John Lafferty. 2017. Information retrieval as statistical translation. In *ACM SIGIR Forum*, pages 219–226. Association for Computing Machinery.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Lera Boroditsky, Lauren A Schmidt, and Webb Phillips. 2003. Sex, syntax, and semantics. *Language in mind: Advances in the study of language and thought*, pages 61–79.

James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. 2017. Quasi-recurrent neural networks. *International Conference on Learning Representations (ICLR 2017)*.

George Casella and Roger L Berger. 2002. *Statistical inference*, volume 2. Duxbury Pacific Grove, CA.

Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. Association for Computational Linguistics.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. Xnli: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485. Association for Computational Linguistics.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2017. Very deep convolutional networks for natural language processing. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics:*

*Volume 1, Long Papers*, pages 1107–1116. Association for Computational Linguistics.

Cedric De Boom, Steven Van Canneyt, Thomas Demeester, and Bart Dhoedt. 2016. Representation learning for very short texts using weighted word embedding aggregation. *Pattern Recognition Letters*, 80:150–156.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Aparna Garimella, Rada Mihalcea, and James Pennebaker. 2016. Identifying cross-cultural differences in word usage. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 674–683.

Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web*, pages 507–517. International World Wide Web Conferences Steering Committee.

Norbert Hedderich. 2010. German-american intercultural differences at the workplace: A survey. *Global Business Languages*, 2(1):14.

Francis Heylighen and Jean-Marc Dewaele. 1999. Formality of language: definition, measurement and behavioral determinants. *Interner Bericht, Center "Leo Apostel", Vrije Universiteit Brüssel*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Geert Hofstede, Gert Jan Hofstede, and Michael Minkov. 2010. *Cultures and Organizations - Software of the Mind: Intercultural Cooperation and its Importance for Survival (3. ed.)*. McGraw-Hill.

Liangjie Hong, Amr Ahmed, Siva Gurumurthy, Alexander J Smola, and Kostas Tsioutsiouliklis. 2012. Discovering geographical topics in the twitter stream. In *Proceedings of the 21st international conference on World Wide Web*, pages 769–778. Association for Computing Machinery.

Juliane House. 1997. *Translation quality assessment: A model revisited*, volume 410. Gunter Narr Verlag.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339. Association for Computational Linguistics.

Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer.

J. Johnson, M. Douze, and H. Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics.

Aashita Kesarwani. 2018. New York Times Comments. Online; accessed 27 August 2018.

Yago Koga and Bethyl A Pearson. 1992. Cross-cultural advertising strategies in japanese vs. american women's magazines. *Intercultural Communication Studies II*, 1(1):1–18.

Shibamouli Lahiri. 2015. SQUINKY! A corpus of sentence-level formality, informativeness, and implicature. *arXiv preprint*, abs/1506.02306.

Guillaume Lample, Alexis Conneau, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. In *International Conference on Learning Representations*.

Raymond R Liu and Peter McClure. 2001. Recognizing cross-cultural differences in consumer complaint behavior and intentions: an empirical examination. *Journal of consumer marketing*, 18(1):54–75.

Maris G Martinsons. 2001. Comparing the decision styles of american, chinese and japanese business leaders. In *Best Paper Proceedings of Academy of Management Meetings, Washington, DC*.

Peter McCullagh. 2018. *Generalized linear models*. Routledge.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, pages 3111–3119. Curran Associates Inc.

Masahiko Minami. 1994. English and japanese: A cross-cultural comparison of parental styles of narrative elicitation. *Issues in Applied Linguistics*, 5(2):383–407.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Slava Novgorodov, Ido Guy, Guy Elad, and Kira Radinsky. 2019. Generating product descriptions from user reviews. In *The World Wide Web Conference*, pages 1354–1364. ACM.

Shereen Oraby, Vrindavan Harrison, Lena Reed, Ernesto Hernandez, Ellen Riloff, and Marilyn Walker. 2016. Creating and characterizing a diverse corpus of sarcasm in dialogue. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 31–41. Association for Computational Linguistics.

Michael Paul and Roxana Girju. 2009. Cross-cultural analysis of blogs and forums with mixed-collection topic models. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3 - Volume 3*, pages 1408–1417. Association for Computational Linguistics.

Ellie Pavlick and Joel Tetreault. 2016. An empirical analysis of formality in online communication. *Transactions of the Association for Computational Linguistics*, 4:61–74.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 2227–2237. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. Unpublished manuscript accessible via the OpenAI Blog.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144. Association for Computing Machinery.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2019. A survey of cross-lingual word embedding models. *Journal of Artificial Intelligence Research*.

Nakatani Shuyo. 2010. Language detection library for java.

Richard A. Shweder. 1991. *Thinking through cultures: Expeditions in cultural psychology*. Harvard University Press.

Richard A Shweder, Jacqueline J Goodnow, Giyoo Hatano, Robert A LeVine, Hazel R Markus, and Peggy J Miller. 2006. The cultural psychology of development: One mind, many mentalities. pages 716–792.

Joseph Tepperman, David Traum, and Shrikanth Narayanan. 2006. "yeah right": Sarcasm recognition for spoken dialogue systems. In *Ninth International Conference on Spoken Language Processing*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 90–94. Association for Computational Linguistics.

Elke U Weber and Christopher Hsee. 1998. Cross-cultural differences in risk perception, but cross-cultural similarities in attitudes towards perceived risk. *Management science*, 44(9):1205–1217.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489. Association for Computational Linguistics.

Zhijun Yin, Liangliang Cao, Jiawei Han, Chengxiang Zhai, and Thomas Huang. 2011. Geographical topic discovery and comparison. In *Proceedings of the 20th international conference on World wide web*, pages 247–256. Association for Computing Machinery.

Avner Ziv. 1988. Teaching and learning with humor: Experiment and replication. *The Journal of Experimental Education*, 57(1):4–15.

# Combining Unsupervised Pre-training and Annotator Rationales to Improve Low-shot Text Classification

**Oren Melamud**
IBM Research
Yorktown Heights, NY, USA
oren.melamud@ibm.com

**Mihaela Bornea**
IBM Research
Yorktown Heights, NY, USA
mabornea@us.ibm.com

**Ken Barker**
IBM Research
Yorktown Heights, NY, USA
kjbarker@us.ibm.com

## Abstract

Supervised learning models often perform poorly at *low-shot* tasks, i.e. tasks for which little labeled data is available for training. One prominent approach for improving low-shot learning is to use unsupervised pre-trained neural models. Another approach is to obtain richer supervision by collecting annotator *rationales* (explanations supporting label annotations). In this work, we combine these two approaches to improve low-shot text classification with two novel methods: a simple bag-of-words embedding approach; and a more complex context-aware method, based on the BERT model. In experiments with two English text classification datasets, we demonstrate substantial performance gains from combining pre-training with rationales. Furthermore, our investigation of a range of train-set sizes reveals that the simple bag-of-words approach is the clear top performer when there are only a few dozen training instances or less, while more complex models, such as BERT or CNN, require more training data to shine.

## 1 Introduction

When trained on large amounts of labeled data, supervised machine learning models demonstrate impressive performance in various domains ranging from text classification to image recognition (Yogatama et al., 2017; Krizhevsky et al., 2012). In many domains, however, labeled data is difficult to obtain. This might be due to annotation cost, or simply because there are no available instances to annotate before the system has to make the next classification decision. Imagine, for example, a personalized email tagger, where new tag types can be created and then used for annotation by the user at any time. When only little training is available, model performance drops dramatically.

Various methods have been proposed to optimize machine learning models trained on little data with the goal of improving performance for a given train set size or reducing the number of training instances that need to be collected. One of the most prominent for text classification (and other NLP tasks) is transfer learning from unsupervised data, or unsupervised pre-training. This approach has been widely adopted with the introduction of pre-trained word embeddings (Mikolov et al., 2013; Joulin et al., 2017). Word type representations have subsequently been extended with pre-trained, context-dependent representations achieving strong results across a wide range of NLP tasks (Melamud et al., 2016; Peters et al., 2018; Howard and Ruder, 2018). A recent strong performer in this line of research is BERT (Devlin et al., 2018), a multi-layer attention-based neural model that is pre-trained on large amounts of plain text and then fine tuned to specific tasks. Systems using BERT have achieved state-of-the-art performance on various NLP tasks, including short text classification.

Another approach for training text classifiers with few labeled instances is to supplement the class labels with annotation rationales. Annotation rationales for text are usually specific subsequences of words within a text instance that the annotator considers to be evidence supporting the label assignment. Zaidan et al. (2007) were the first to investigate the use of annotator rationales for training text classifiers. They obtained manual rationales for an IMDB movie review dataset (Pang et al., 2002) and showed that an SVM classifier can be modified to obtain improved accuracy using these rationales. The bold text in the movie review snippet "*perhaps **the greatest element to the film** is its constant surprises and unpredictability*" is one example of an annotated rationale in this dataset. Abedin et al. (2011) showed similar results using SVM and rationales for classifying



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3884–3893,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

cause identifiers in aviation safety reports from the Aviation Safety Reporting System (ASRS).

In this work, we combine the power of unsupervised pre-training with the fine-grained guidance from rationales to propose two novel low-shot text classification methods. Both methods promote rationale-like features in the input to improve the classification outcome. The first method uses rationale supervision to bias a simple bag-of-words text embedding towards the more discriminative features. In the second method, we show how to fine-tune a pre-trained BERT model with both instance labels and rationales jointly. An investigation across a range of train-set sizes on the IMDB and ASRS text classification datasets reveals that our bag-of-words approach substantially outperforms all baselines for train sets of up to a few dozen instances. For larger sizes, our BERT-based model and a CNN baseline perform the best. Our code is publicly available. [1]

## 2 Related Work

As mentioned in the Introduction, Zaidan et al. (2007) was the first work to show how manual rationales annotated for movie reviews (Pang et al., 2002) can be used to improve classifier accuracy. They also investigated the annotation effort ramifications and found that the additional annotation of a few rationales per instance on top of the standard label annotation roughly doubled annotation time. When enough instances are readily available for annotation, this information is important to assess the trade-off between obtaining annotator rationales and obtaining standard labeled instances. However, we note that there are cases where more unlabeled instances may not be available and then obtaining annotator rationales may be the only way to improve the classifier accuracy.

The model most closely related to our rationale-biased bag-of-words is the one proposed by Sharma and Bilgic (2018) (denoted RA-SVM in Section 5.2). It biases the input representation by adjusting feature weights in a way that is agnostic to the type of classifier used. More specifically, it uses one-hot word features and performs simple fixed discounting of non-rationale word feature weights during training only. In contrast, our model learns a rationale-biased embedded representation of texts and this representation is used both at train and test time.

The model most closely related to our rationale-biased BERT model is the one proposed by Zhang et al. (2016) (denoted RA-CNN in Section 5.2). This model uses a CNN sentence encoder with pre-trained word embeddings. It follows a two-step approach, where one sentence classifier is trained to identify sentences containing rationales and weigh them higher in an overall 'weighted-average-of-sentences' document representation. Then a second classifier is trained to make the final label prediction based on that representation. They showed improved accuracy compared to non-rationale-augmented CNNs as well as to rationale-augmented SVM models. Our main contributions relative to that work are the introduction of the joint-learning technique, which learns text classification and rationale words identification concurrently, and the adaptation of the entire approach to BERT. The ability of our model to identify rationale spans within sentences also seems useful for interpretability. Tepper et al. (2013) also chose a two-step approach similar to Zhang et al. (2016) using more traditional classifiers with sparse features. They train one classifier to identify rationale-like pieces of text and then a second classifier is trained to label input instances taking only the rationale-like texts into account.

Finally, *few-shot* learning methods attempt to learn how to effectively train classifiers for new classes with only few labeled instances, using techniques commonly referred to as *meta-learning* (Snell et al., 2017; Yu et al., 2018). Snell et al. (2017), for example, learn an embedding function that is used to represent the input instances such that simple nearest-prototype classifiers can be trained effectively to recognise instances of a new target class after being exposed to a small number of labeled examples. Although a new class is learned from few labeled instances as in our work, the preceding meta-learning process relies on a large number of labeled instances aggregated over many closely related classes or tasks. This is an important distinction from our low-shot setting.

## 3 Rationale-biased Bag-of-words

In this section, we describe our proposed rationale-biased bag-of-words method for performing low-shot text classification with rationales.

[1] https://github.com/mihaela-bornea/low-shot-text-classification

### 3.1 Rationale-biased representation

Let $\vec{v}$ be a vector representation of word $v$ (either one-hot or a low-dimensional embedding). We represent a text instance $t = (v_1, .., v_n)$ as an L2-normalized, weighted average of its words giving the text embedding function:

$$emb(t) = \vec{t} = \frac{1}{Z} \sum_{v_i \in t} w_i \vec{v_i} \quad (1)$$

where $Z$ is the normalization factor.

This embedding is a standard bag-of-words approach to representing texts, where weights, such as TF-IDF, are typically used to bias the representation towards words that carry more important information. We hypothesize that since rationales highlight support for annotator classification decisions, biasing this text embedding function towards *rationale-like* words would expose the more discriminative features. To this end, given a rationale-bias function $f_{rb}(v)$ that estimates the similarity between a word $v$ and the annotated rationale texts in the training dataset, the rationale biased representation could be computed using Eq. (1) with $w_i = f_{rb}(v_i)$. The remainder of this section describes how we compute $f_{rb}(v)$.

Let the *prototype* representation of a collection of text instances $T = (t_1, .., t_m)$ be the L2-normalized centroid of its text instance vectors:

$$prot(T) = \frac{1}{Z} \sum_{t_i \in T} \vec{t_i} \quad (2)$$

where $Z$ is the normalization factor.

A rationale for a text instance $t$ and class label $c$ is any word subsequence $r$ of $t$ that was annotated as evidence for the assignment of $c$ to $t$. A *class rationale prototype* $\vec{R_i} = prot(R_i)$ is the prototype representation of the rationales $R_i = (r_1, .., r_l)$ annotated for class $c_i$, where each rationale instance $r$ is embedded using Eq. (1) with uniform weights. We define the following rationale-bias function $f_{rb}(v)$:

$$f_{rb}(v) = (\exp sim_{rb}(v))^{\alpha} \quad (3)$$

$$sim_{rb}(v) = \max_{i \in \{1..k\}} cosine(\vec{v}, \vec{R_i})$$

That is, $sim_{rb}(v)$ is the maximum cosine similarity of the word vector to any of the class rationale prototypes, and $f_{rb}(v)$ is an always positive derivative of $sim_{rb}(v)$ that is controlled by the hyperparameter $\alpha \geq 0$. Higher values of $\alpha$ mean stronger magnification of the impact of the rationale biases; $\alpha = 0$ means that the rationale bias is completely ignored. With this rationale-bias function, we can now compute rationale-biased representations for text instances using Eq. (1), and these representations can be used as inputs to any classifier.

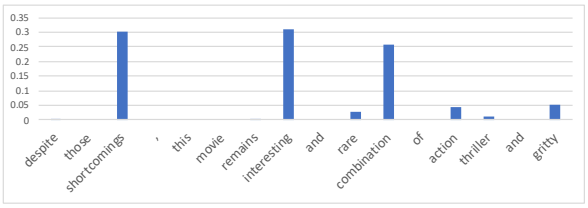


Figure 1: The bias weights generated by a rationale-biased model trained on 20 instances, to a snippet from an example movie review.

In early experiments, we noticed that many *non-discriminating*, common words in the target domain (e.g., *movie*, *certainly*, *guy*) tended to get high rationale biases. This is presumably due to the fact that rationales in these datasets often have spans that include multiple rationale words along with other words. To compensate, we adjusted the class rationale prototype representation to:

$$adjusted(R_i) = prot(R_i) - prot(T_{all}) \quad (4)$$

where $T_{all}$ is the collection of all training instances across all classes. That is, we subtract the entire training-set prototype vector from each class rationale prototype. We L2-normalize this representation as well.

Figure 1 is an example illustrating actual bias weights generated by our model. As can be seen in this case, the most positive and negative indicative words are 'picked up' as indicated by the high respective bias. However, it should also be noted that this model misses the sentiment polarity inversion expressed by the word *despite*, due to its simple sequence-agnostic nature.

### 3.2 Classifiers

Various types of classifiers can be trained to perform text classification on top of the rationale-biased text representations described in the previous section. We chose to experiment with a standard SVM classifier as well as a simple, nearest-prototype classifier similar to the approach taken in prior few-shot learning work (Snell et al., 2017). Before training and applying these classifiers, we compute the class rationale prototypes to determine the rationale-bias function $f_{rb}$. We then use

that function to embed all of the dataset text instances and use these text embeddings as inputs to the classifiers. More specifically, for our nearest-prototype classifier, given a training set of text instance collections $T_1, .., T_k$ labeled with classes $c_1, .., c_k$, we represent each class $c_i$ by the *class prototype* $\vec{T_i} = prot(T_i)$. To classify a new text instance $t'$ we simply choose the closest class prototype using cosine similarity:

$$class(t') = \arg\max_{i \in \{1..k\}} cosine(\vec{t'}, \vec{T_i}) \quad (5)$$

Our hypothesis is that this proposed method would do comparatively well with very little training data because (a) with the guidance of the rationale bias it could pick up individual discriminative words more quickly (e.g., *good* or *bad* for sentiment tasks); (b) combining the use of pre-trained word embeddings as the underlying word representation with rationale bias would help generalize to words unseen in the training data (e.g. to *best* from *good*) and would facilitate an effective rationale bias similarity function $sim_{rb}(v)$; and (c) the training regimen is simple with a relatively small number of parameters to be learned compared to some more complex neural models such as the one discussed in the following section.

## 4 Rationale-biased BERT

BERT (Devlin et al., 2018) is a context-aware neural network model that has shown excellent results when used as a basis for various supervised NLP systems. Its power comes from transferring context-aware language modeling information learned from unsupervised training over very large amounts of plain text. In this section, we show how we adapt BERT to handle long texts and then to incorporate rationale supervision together with standard instance labels.

### 4.1 Fine-tuning BERT

Presented with a sequence of word tokens[2] $s = (a_1, ..a_n)$ as input, a BERT model outputs contextualized vector encodings for each token $B(s) = (h_0, h_1, .., h_n)$,[3] where $h_0$ is a special encoding for the entire sequence. Pre-trained versions of this model were trained by Devlin et al. (2018) against language modeling and sentence prediction objectives on large corpora of text allowing them to capture valuable contextualized information in these encodings. Devlin et al. (2018) applied dropout followed by a simple linear layer, $L_{tok}(drop(B(s)_i))$ where $i > 0$ to perform token-level classification and $L_{text}(drop(B(s)_0))$ for sentence-level classification. They fine-tuned the entire model on task-specific labels using a standard cross-entropy loss function. With this approach, they achieved state-of-the-art results on various NLP tasks, such as Named Entity Recognition (token-level) and single-sentence sentiment classification (sentence-level).

### 4.2 Averaged BERT classifier for long texts

While BERT proved to be very effective for classifying short texts such as individual sentences, its computational complexity is quadratic with the length of the input text yielding long run times for texts longer than a few sentences. Since our input texts may be long, we split them into sentences, apply BERT to each of the sentences separately and average the outputs as follows.

Given a text instance comprising the set of sentences $t = (s_1, ..., s_m)$, we encode it as the weighted average of the individual sentences:

$$B_{avg}(t) = \vec{t} = \sum_{i \in 1..m} w_i \cdot drop(B(s_i)_0) \quad (6)$$

where $w_i$ are the weights used in the weighted average. We experiment with uniform weights and also with learning weights as learned attention, using a linear layer $L_{attn}$:

$$w_i = softmax(L_{attn}(drop(B(s_i)_0))) \quad (7)$$

Finally, similar to the original BERT approach, to fine tune a classifier for long text instances, we apply a linear layer to the text encoding $L_{text}(B_{avg}(t))$ and train the whole model to predict the target labels with the loss function:

$$S_{target}(t, l_t) = CE(L_{text}(B_{avg}(t)), l_t) \quad (8)$$

where $l_t$ is the training label for text instance $t$ and $CE$ is the cross-entropy loss function. Figure 2 illustrates our architecture.

### 4.3 Rationale-biased BERT classifiers

We propose two methods to incorporate rationale supervision into our averaged BERT text classifier. The first is based on straightforward multi-task learning, where rationale word prediction is considered an auxiliary task that is jointly trained

[2] BERT breaks words into subword units called *word pieces* (Wu et al., 2016).

[3] We refer here to the top-most output layer of BERT.

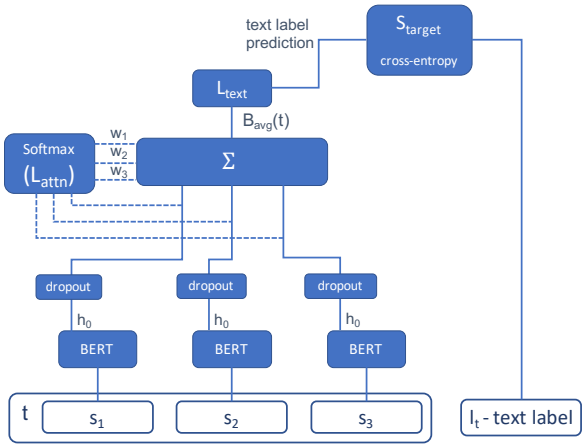


Figure 2: Averaged BERT model architecture. The three BERT boxes refer to the same single instance of the model. $L_{attn}$ is the optional sentence attention learning component from Eq. (7).

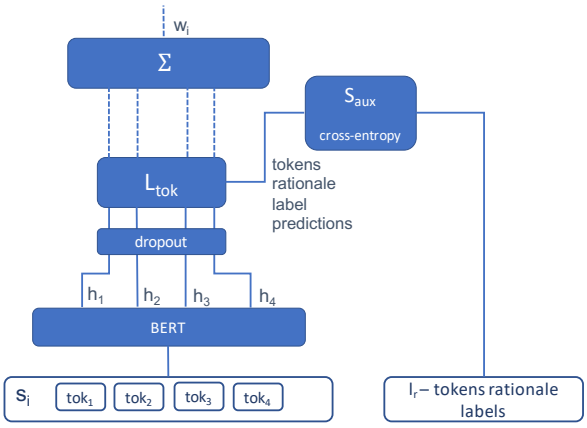


Figure 3: Rationale-bias auxiliary model component. The BERT model here is the same one used for the main target task. $w_i$ is the average probability that a token in the sentence is part of a rationale, which can be used as sentence attention in the averaged BERT.

with the main text label prediction task. The second extends the first by using rationale supervision to estimate the attention to each sentence. It then uses that attention (or importance measure) to inform the weighted average from Eq. (6).

**Simple joint learning.** Figure 3 illustrates the auxiliary model added to the averaged BERT. The same BERT model is trained to perform two classification tasks: (1) predicting the label of each text instance (target text classification task); and (2) predicting for every input token in every input sentence whether it is part of an annotated rationale subsequence (auxiliary task). This is achieved by introducing the auxiliary loss function:

$$S_{aux}(t, l_r) = \frac{1}{n} \sum_{i,j} CE(L_{tok}(drop(B(s_i)_j)), l_r^{i,j}) \quad (9)$$

where $l_r^{i,j}$ is the rationale label of token $j$ in sentence $i$, $L_{tok}$ is a linear layer for learning to classify tokens and $n$ is the total number of tokens in text $t$. Finally, the multi-task objective function per input text $t$ is:

$$S_{joint}(t, l_t, l_r) = S_{target}(t, l_t) + S_{aux}(t, l_r) \quad (10)$$

Our hypothesis is that the auxiliary task will infuse lower-level information regarding the relevance of input words into the BERT model, which will help boost the main task classifier.[4]

[4]In early experiments, we also tried infusing this information to lower layers of the BERT model (i.e., to an internal layer rather than the output layer) following Søgaard and Goldberg (2016), but that did not improve the performance.

**Rationale-biased Attention.** The dotted lines in Figure 3 illustrate our proposed approach to extend simple joint-learning by using the rationale signal to learn how to identify the more important sentences in the input text. To train this model, we use the same multi-task loss function in Eq. (10). However, this time, instead of using uniform weights in the averaged BERT, we consider the attention weight of a sentence to be the average of the probabilities that each of its tokens is in a rationale span. These weights are used to compute the final weighted average representation of the text input in Eq. (6). In our experiments, we found that it was better not to backpropagate the error from $S_{target}$ towards the token-level rationale classifier (i.e. along the dotted lines). Therefore, the rationale classifier is still trained only based on $S_{aux}$.

The hypothesis here is that this model could be successful by infusing more explicitly the inductive bias that sentences with rationale words are more important than others for the final classification task. Incorporating more inductive bias is particularly important with little training data. Compared to the bag-of-words approach presented in the previous section, however, we presume that both BERT-based methods presented here might be more challenged with extremely little training data because of the more complex underlying models and training regimen.

Figure 4 shows an example of sentence attentions and word rationale probabilities computed by our rationale-biased attention BERT model.

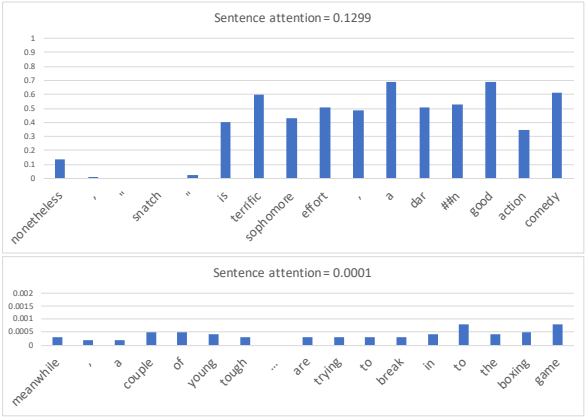


Figure 4: Excerpts from the sentence that got the highest attention (top) and lowest attention (bottom) in an IMDB movie review, by our rationale-biased attention BERT model. Bars indicate the probability the model assigns for every word of being part of a rationale.

Being context-aware, this model captures relevant sequences of words rather than just spotting key words. In addition to being useful for the classification task, these sentence and token weights also seem useful for model interpretability.

## 5 Experimental Settings

We conducted experiments with two English text classification datasets. Zaidan et al. (2007) added rationales supporting the positive/negative class labels on 1800 movie reviews from the Internet Movie Database (IMDB) dataset (Pang et al. (2002)). We used 900 reviews for training, 450 for development and 450 for testing, and maintained the original even positive/negative balance in every split.

The Aviation Safety Reporting System (ASRS) dataset contains reports of aviation safety incidents. We used the version from Abedin et al. (2011), which has 1233 reports in the training set, 100 for development and 1000 for testing. Reports are labeled with one or more of 14 safety incident cause categories. To create a binary classification dataset comparable to the IMDB dataset, we extracted a balanced subset of reports that were labeled with the 'Proficiency' label or the 'Physical Environment' label (but not both). We chose these particular two labels because they have a similar number of instances and are among the top most frequent labels in the dataset. This resulted in 386 reports for training and 392 reports for testing. Table 1 provides some additional statistics regarding these two datasets.

| | IMDB | ASRS |
|---|---|---|
| sentences per instance | 33.8 | 15.3 |
| words per instance | 736.6 | 262.1 |
| rationales per instance | 8.6 | 3.0 |
| words per rationale | 7.2 | 2.2 |

Table 1: Dataset statistics

### 5.1 Experimental protocol

To test the hypothesis that our proposed rationale-biased methods can improve classification with little training data, we ran experiments varying the amount of training data used from 2 instances up to roughly 400. For each training size $n$, we sampled $n$ training instances to train the models and tested on the entire dev/test set. We repeated this experiment 30 times (for each $n$) and report average accuracies. We fixed all parameters based on tuning on only the IMDB development set prior to running on the IMDB and ASRS test data (no tuning was done on ASRS data).

### 5.2 Compared methods

***RA-SVM*** is the Rationale-Augmented SVM by Sharma and Bilgic (2018) that biases the text representation to rationales by discounting non-rationale word features by a fixed discount ratio during training. All words are TF-IDF weighted prior to discounting. We experimented with different discounts on the IMDB development set and found that 0.1 gave the best results. And while Sharma and Bilgic (2018) use a one-hot word representation, we also experimented with giving it dense word embeddings, but this yielded degraded performance. We also report results with no rationale discount - a plain ***SVM*** baseline.

***RA-CNN*** is the Rationale-Augmented CNN by Zhang et al. (2016), described in section 2. We used the authors' implementation with pre-trained word2vec Google News word embeddings with 300 dimensions for 3 million words.[5] We tuned batch size, dropout, learning rate and number of epochs on our development set. We also report results with no rationale weighting - a plain ***CNN*** baseline.

***RB-BOW-SVM*** and ***RB-BOW-PROTO*** are the SVM and nearest-prototype classifiers running on top of our proposed rationale-biased bag-of-words (BOW) text representation (Section 3). We tuned

[5] https://code.google.com/archive/p/word2vec/

the hyperparameter $\alpha$ on the dev set, obtaining the best results with $\alpha = 6$ for both classifiers. ***BOW-PROTO*** is the prototype classifier based on a BOW with no rationale bias ($\alpha = 0$). We tried both one-hot word representations and word embeddings using the same word2vec embeddings as for CNN and RA-CNN.

***AVG-BERT*** and ***ATTN-WAVG-BERT*** are our proposed BERT-based models trained without rationales using a uniform-weight average and an attention-based weighted average of sentences, respectively (Section 4.2). ***RB-AVG-BERT*** is the joint model that uses rationale prediction as an auxiliary task, and ***RB-WAVG-BERT*** is the method that also uses rationales to estimate sentence-level attention (Section 4.3). For all of the above, we used the pre-trained models 'bert-base-uncased' and 'bert-base-cased' for the IMDB and ASRS datasets, respectively. We used default hyperparameters, except for using learning rate of 5e-6 and 10 epochs for fine-tuning, which we found to work better for AVG-BERT with light tuning. To fit the computation graphs into the GPU memory (NVIDIA Tesla V100) we trimmed sentences longer than 48 word pieces, and texts containing more than 64 sentences. We implemented our models using a PyTorch implementation of BERT.[6]

***ULMFiT*** (Howard and Ruder, 2018) is a recent method achieving state-of-the-art text classification results on several datasets. It is trained in three steps: (1) training a general-domain recurrent neural network language model on a large corpus (WikiText-103); (2) fine-tuning the language model to the domain data of the target task disregarding class labels; and (3) fine-tuning a classifier for the task using the encoder of the learned fine-tuned language model as a starting point. We used the *fast.ai*[7] implementation of ULMFiT, tuning batch size, dropout multiplication factor, learning rates and number of epochs on our development set. ULMFiT does not currently have a rationale-augmented version.

## 6 Results

In this section, we start by investigating the performance of different variants of our proposed models on the IMDB dev set. The most promising configurations are then evaluated on the IMDB and ASRS test sets. Since both of our datasets are binary and balanced we note that the random baseline accuracy is 50%.

### 6.1 Development set investigation

To demonstrate the contribution of various components of our rationale-biased bag-of-words approach, we performed several ablation experiments on the development set, shown in Figure 5. The compared variants include: ***RB-BOW-PROTO***: our full method with pre-trained word embeddings; ***BOW-PROTO***: no rationale-bias; ***OH***: RB-BOW-PROTO with one-hot word representations instead of word embeddings; ***NO-ADJUST***: RB-BOW-PROTO without discounting common words by subtracting the general prototype vector (Eq. (4)); ***SINGLE***: RB-BOW-PROTO with just a single rationale prototype representing all rationales instead of one rationale prototype for each class (Eq. (3)); and ***INSTANCE***: RB-BOW-PROTO with a rationale prototype for every individual text instance instead of one per class. As can be seen, combining the use of pre-training and rationales is critical to the success of our full method. First, the importance of using pre-trained word embeddings is evident by the significant drop in performance when using one-hot encodings with train-sets with 60 instances or fewer. Second, the importance of rationale-biasing can be seen by the large performance drop with BOW-PROTO across all train-set sizes. With those two components in place, we see that the RB-BOW-PROTO is able to perform significantly above the 50% random baseline with very few training instances. We note that RB-BOW-SVM demonstrated similar performance to RB-BOW-PROTO. For brevity, we don't include a detailed report here, but do include RB-BOW-SVM in the test results section.

To illustrate the potential merit in using our proposed rationale-biased embedding function we show 2-dimensional plots of text embeddings with and without rationale-bias in Figure 7. We randomly sampled a train-set of size 60 from the IMDB dataset and used it to learn the rationale-biased embedding function. We plotted the positive and negative class prototypes as learned from the training set as well as the embeddings of the IMDB dev set instances. We used T-SNE to reduce those 300-dimensional embeddings down to

[6] https://github.com/huggingface/pytorch-pretrained-BERT

[7] https://www.fast.ai/

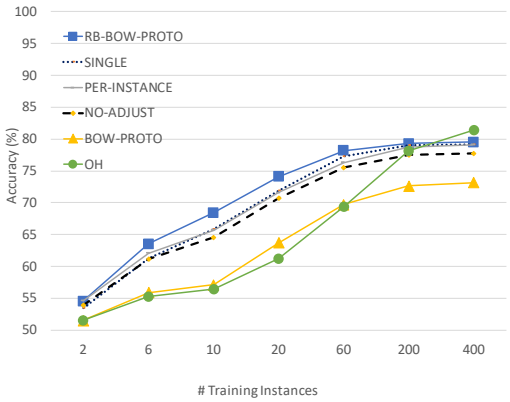


Figure 5: Ablation experiments on the IMDB dev set.

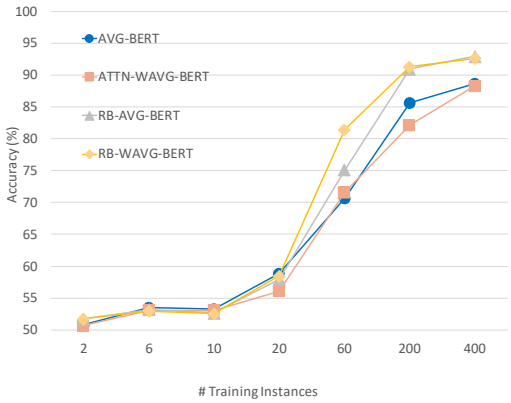


Figure 6: BERT-based models on the IMDB dev set.

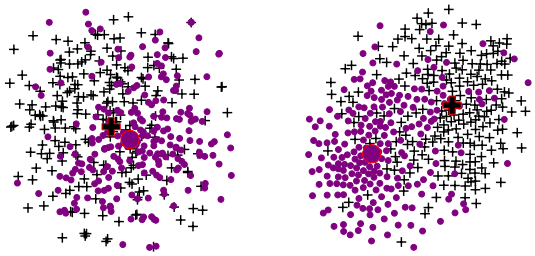

Figure 7: Bag-of-words text embedding without rationale bias (left) and with (right). +/o markers are positive/negative dev text instances. Large markers are positive/negative class prototypes learned during training.

two dimensions for visualization. The rationale-biased representation appears to discriminate better between positive and negative instances. In particular, we notice an increase in the distance between the representations of class prototypes and a clearer clustering of the development set instances around their respective class prototypes. For this data subset, the nearest prototype classifier achieves accuracy of 78.1% when using the rationale-biased embedding compared to 69.7% with an unbiased representation.

Next, we investigate the performance of the BERT-based models on the IMDB dataset. Figure 6 shows the results on the dev set. First we note that unlike the simpler bag-of-words methods, all of these models require at least 20 training instances before they meaningfully improve on the random 50% baseline. On the other hand, with 200 training instances or more, all of the BERT-based models significantly outperform the bag-of-words methods. More specifically, we see that RB-AVG-BERT and RB-WAVG-BERT, which use rationale supervision, significantly outperform AVG-BERT and ATTN-WAVG-BERT, which only use text label supervision, by up to 10 accuracy points. Between RB-AVG-BERT and RB-WAVG-BERT, the latter, which incorporates more inductive bias, performs more robustly.

### 6.2 Test set results

Figure 8 and Table 2 show the IMDB test-set results of our best rationale-biased methods and their non-rationale counterparts (chosen based on dev set performance), as well as all of the baselines.[8] Our bag-of-words models, RB-BOW-PROTO and RB-BOW-SVM, which combine rationale-biasing with pre-trained word embeddings, outperform all other baselines by a large margin of up to more than 20 absolute accuracy points for training sets of size 20 or smaller. On these particularly small datasets, the other rationale-aware baselines, RA-SVM and RA-CNN, do not perform as well. For larger training sets of size 60 and more, all systems perform better, as expected. In particular, the more complex CNN and BERT-based models have significantly improved performance with more supervision. Both RA-CNN and our RB-BERT, which combine pre-training with rationales, benefit substantially from rationale supervision, with up to more than 30 absolute accuracy points improvement. In absolute terms, our BERT models outperform their CNN counterparts and every other baseline in this training size range.

Finally, Figure 9 and Table 3 show results on the ASRS test set. The trends here are similar, showing the robust merit in our rationale-bias approach. The only striking difference is that in this dataset, the CNN methods outperform our BERT-based

[8] ULMFiT is not included in Figures 8 and 9 for better readability.

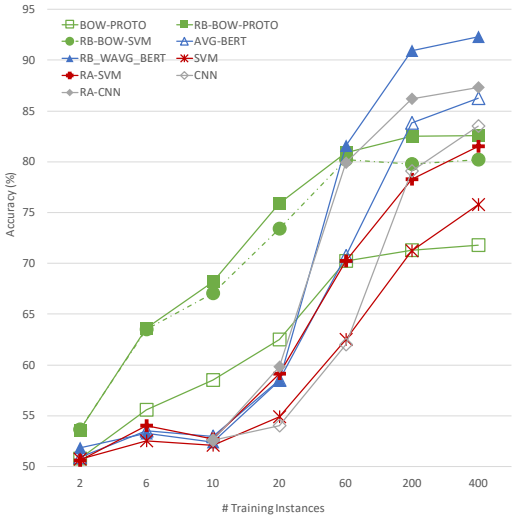


Figure 8: IMDB test set results

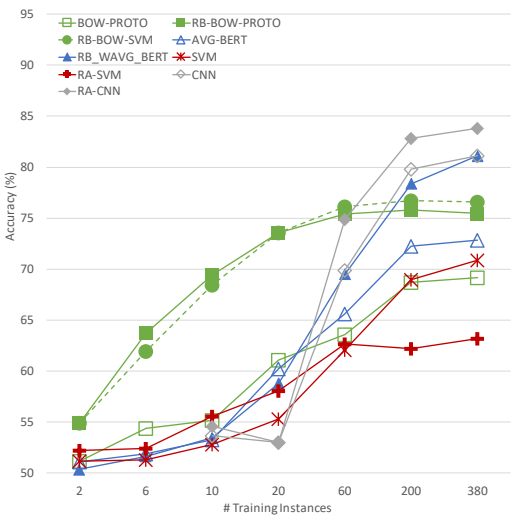


Figure 9: ASRS test set results

| | Number of Training Instances | | | | | | |
|---|---|---|---|---|---|---|---|
| | **2** | **6** | **10** | **20** | **60** | **200** | **400** |
| **RB-BOW-PROTO** | 53.6 | 63.6 | 68.2 | 75.9 | 80.9 | 82.5 | 82.6 |
| **RB-BOW-SVM** | 53.6 | 63.5 | 67.0 | 73.4 | 80.2 | 79.7 | 80.2 |
| **BOW-PROTO** | 50.8 | 55.6 | 58.5 | 62.5 | 70.2 | 71.3 | 71.8 |
| **RB-WAVG-BERT** | 51.8 | 53.2 | 52.3 | 58.4 | 81.6 | 90.9 | 92.2 |
| **AVG-BERT** | 50.9 | 53.5 | 52.9 | 58.5 | 70.7 | 83.8 | 86.2 |
| **RA-SVM** | 50.6 | 54 | 52.7 | 59.1 | 70.2 | 78.3 | 81.5 |
| **SVM** | 50.7 | 52.5 | 52.1 | 54.9 | 62.5 | 71.2 | 75.8 |
| **RA-CNN** | * | * | 52.6 | 59.8 | 79.9 | 86.2 | 87.3 |
| **CNN** | * | * | 52.6 | 54 | 62 | 79.1 | 83.5 |
| **ULMFiT** | * | * | * | 54.7 | 62 | 71.3 | 78.8 |

Table 2: IMDB test set results.

| | Number of Training Instances | | | | | | |
|---|---|---|---|---|---|---|---|
| | **2** | **6** | **10** | **20** | **60** | **200** | **380** |
| **RB-BOW-PROTO** | 54.9 | 63.7 | 69.4 | 73.6 | 75.4 | 75.8 | 75.5 |
| **RB-BOW-SVM** | 54.9 | 61.9 | 68.4 | 73.5 | 76.1 | 76.7 | 76.5 |
| **BOW-PROTO** | 51.2 | 54.4 | 55.2 | 61.1 | 63.6 | 68.7 | 69.2 |
| **RB-WAVG-BERT** | 50.3 | 51.6 | 53.4 | 58.8 | 69.5 | 78.3 | 81.0 |
| **AVG-BERT** | 51.1 | 51.9 | 53.2 | 60.2 | 65.6 | 72.2 | 72.8 |
| **RA-SVM** | 52.2 | 52.4 | 55.6 | 58.1 | 62.7 | 62.2 | 63.2 |
| **SVM** | 51.2 | 51.3 | 52.8 | 55.3 | 62.1 | 69 | 70.9 |
| **RA-CNN** | * | * | 54.6 | 53 | 74.9 | 82.8 | 83.8 |
| **CNN** | * | * | 53.7 | 53 | 69.9 | 79.8 | 81.1 |
| **ULMFiT** | * | * | * | 53.7 | 59.9 | 65.9 | 69.3 |

Table 3: ASRS test set results.

methods both in their rationale and non-rationale versions. We note that the RA-CNN baseline takes a similar approach to ours in the sense that it also combines pre-training (pre-trained word embeddings) with rationales. The reasons for one method being better than the other on a particular dataset could be related to the compatibility between the pre-training corpus and the test dataset or sensitivity to hyperparameter tuning (we did not do any tuning on the ASRS dev set).

## 7 Conclusions

In this work, we addressed an important challenge in supervised machine learning, namely the dependency on large amounts of labeled training data. We demonstrated that substantial performance gains in low-shot text classification can be obtained by combining unsupervised pre-training with annotator rationales across various methods. To this end, we presented two novel methods that together provide strong results on a range of train set sizes. We performed experiments using various baselines, data sizes and ablations to help understand what works best for varying amounts of available training data. Most notably, we showed that simple bag-of-words methods with pre-trained word embeddings work best for very small train sets, while more complex methods based on pre-trained language models excel when more data is available.

## References

Muhammad Arshad Ul Abedin, Vincent Ng, and Latifur Khan. 2011. Learning cause identifiers from annotator rationales. In *IJCAI*, pages 1758–1763.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *ACL*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and

Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *EACL*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional lstm. In *CoNLL*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *EMNLP*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*.

Manali Sharma and Mustafa Bilgic. 2018. Learning with rationales for document classification. *Machine Learning*, 107(5):797–824.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *ACL*.

Michael Tepper, Heather L Evans, Fei Xia, and Meliha Yetisgen-Yildiz. 2013. Modeling annotator rationales with application to pneumonia classification. In *Workshops at the Twenty-Seventh AAAI Conference on Artificial Intelligence*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Dani Yogatama, Chris Dyer, Wang Ling, and Phil Blunsom. 2017. Generative and discriminative text classification with recurrent neural networks. *arXiv preprint arXiv:1703.01898*.

Mo Yu, Xiaoxiao Guo, Jinfeng Yi, Shiyu Chang, Saloni Potdar, Yu Cheng, Gerald Tesauro, Haoyu Wang, and Bowen Zhou. 2018. Diverse few-shot text classification with multiple metrics. In *NAACL*.

Omar Zaidan, Jason Eisner, and Christine Piatko. 2007. Using annotator rationales to improve machine learning for text categorization. In *NAACL*.

Ye Zhang, Iain Marshall, and Byron C Wallace. 2016. Rationale-augmented convolutional neural networks for text classification. In *EMNLP*.

# ProSeqo: Projection Sequence Networks for On-Device Text Classification

**Zornitsa Kozareva**
Google
Mountain View, CA, USA
zornitsa@kozareva.com

**Sujith Ravi**
Google Research
Mountain View, CA, USA
sravi@google.com

## Abstract

We propose a novel on-device sequence model for text classification using recurrent projections. Our model ProSeqo uses dynamic recurrent projections without the need to store or look up any pre-trained embeddings. This results in fast and compact neural networks that can perform on-device inference for complex short and long text classification tasks.

We conducted exhaustive evaluation on multiple text classification tasks. Results show that ProSeqo outperformed state-of-the-art neural and on-device approaches for short text classification tasks such as dialog act and intent prediction. To the best of our knowledge, ProSeqo is the first on-device long text classification neural model. It achieved comparable results to previous neural approaches for news article, answers and product categorization, while preserving small memory footprint and maintaining high accuracy.

## 1 Introduction

In the last decade, research in deep neural networks has lead to tremendous advances and state-of-the-art performance on wide range of Natural Language Processing (NLP), speech and vision applications. Coupled with the tremendous growth and adoption of smart devices such as mobile phones, watches, Internet of Things, conversational AI devices like Alexa and Google Home, these deep learning improvements have resulted in text and speech becoming the primary modes of communication (Cohen, 2008). To date, the majority of the models run on the server side by taking user input, sending the data to the server (or cloud) for processing and returning the results back to the user. This often results in latency delays, inconsistent user experiences and concerns over user privacy.

One way to surmount these challenges is to develop and deploy text and speech models that run inference entirely on-device and return accurate predictions in real-time. To make this work, the on-device models have to be very small in size (few kilobytes or megabytes) to fit on-devices like mobile phone, watch and IoT; to have low latency and be as accurate as server side models. These on-device challenges opened up new active area of research, which recently has shown promising results for speech (Lin et al., 2018), wake word detection (He et al., 2017), dialog act (Ravi and Kozareva, 2018) and intent prediction short text classification (Ravi and Kozareva, 2019).

Previous attempts for on-device text classification used hashed <*input_text*, *output_class*> pairs. Unfortunately, such models cannot handle examples that are not part of the lookup table and cannot generalize to more complex tasks like the ones we solve in this paper – long text classification, conversational intent prediction. (Bui et al., 2018) combined graphs with neural networks to improve the robustness of the model, but this resulted in large model sizes that cannot fit on-device. The most successful work is the self-governing neural network (SGNN) from (Ravi and Kozareva, 2018) and (SGNN++) from (Ravi and Kozareva, 2019), which use locality-sensitive projections (Ravi, 2017, 2019) to encode short text into low-dimensional representations. SGNN reached state-of-the-art results on dialog acts outperforming recurrent networks. While successful, SGNN's static projections make it hard to generalize to longer sequences.

In this work, we take one step further by developing a novel embedding free on-device neural model that combines *dynamic projections* with recurrent operations to learn powerful representations that can more powerfully encode text representations and can be equally efficient at short and long text classification tasks. The main contributions of our work are:



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3894–3903,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

- We propose novel on-device dynamic projection sequence network ProSeqo for short and long text classification; To the best of our knowledge, this is the first on-device neural network for long text classification;
- Unlike SGNN which uses static projections and fully connected networks, ProSeqo is the first work to introduce embedding-free LSTMs that combine *dynamic projections* with recurrent operations to learn powerful yet compact neural networks applicable for on-device scenarios;
- We conduct exhaustive evaluations and comparisons against state-of-the-art on-device and deep learning models for short and long text classification;
- Our results show that ProSeqo outperformed state-of-the-art on-device SGNN network (Ravi and Kozareva, 2018) with up to +8.9% accuracy on short text classification and +35.9% accuracy on long document classification. This shows that ProSeqo's dynamic recurrent projections are more powerful than the static ones used in SGNN;
- Similarly, our results show that ProSeqo outperformed the non-on-device neural models like RNN (Khanpour et al., 2016), CNN (Lee and Dernoncourt, 2016) for dialog acts and intent prediction; as well as LSTMs (Zhang et al., 2015), character CNNs (Zhang et al., 2015; Bui et al., 2018) for long document classification of news, answers and product reviews. This is a significant advancement given ProSeqo's small model size compared to these widely used approaches;
- Finally, we conduct a series of ablation studies to demonstrate the effect of ProSeqo's projection size on accuracy and the ability of ProSeqo to produce small model sizes while retaining high accuracy.

## 2 ProSeqo: Projection Sequence Networks for On-device Text Classification

We introduce a neural architecture, *ProSeqo*, a projection-based sequence network, that is designed to (a) be efficient and learn compact *embedding-free* neural representations of text at the semantic level, (b) capture contextual and morphological information at the character-level, and (c) model longer context across words and sentences that is suited for both short and long-text classification tasks.

The key differences to prior work (e.g., RNNs, CNNs) are that we use context information to *dynamically* compute text representations and learn compact neural networks for text classification that are suited for on-device applications. Our work departs significantly from recent on-device work (Ravi and Kozareva, 2018) and (Ravi and Kozareva, 2019) which leverage projection operations to learn efficient networks that can even be transferred to other tasks (Sankar et al., 2019b). However, they are limited to short-text classification tasks. In contrast, our model effectively uses contextual information and combines *recurrent* and *projection* operations to achieve efficiency and enable learning more powerful neural networks that generalize well and can solve more complex language classification tasks. As a result, our projection sequence network is able to dynamically project and learn efficient neural classifier models that are competitive with state-of-the-art RNNs and CNNs without the need to store or lookup any pre-trained embeddings.

### 2.1 Model Overview

The overall architecture of ProSeqo is shown in Figure 1. It consists of multiple parts: a word context projector, a recurrent projector, a projection sequence encoder, and a classification layer. Next, we describe in details each component.

### 2.2 Word Context Projector

Given an input text $x_i$ (sentence or document) with sequence of words $w_{it}$, where $w_{it}$ refers to $t$-th word in the input, we first project the input $x_i$ to a vector representation $\mathbb{P}(x_i)$. Majority of the neural models used in NLP rely on embedding matrices (word or character level) to achieve this. Instead, we dynamically compute word vector representations using *context* and locality-sensitive projections (Ravi, 2017).

#### 2.2.1 Character and Morphology Projection

Each word contains characters and morphological information that can be informative for the task. For example, the words `amaze` and `amazing`, may appear slightly different but capture similar semantics for a review classification task. It

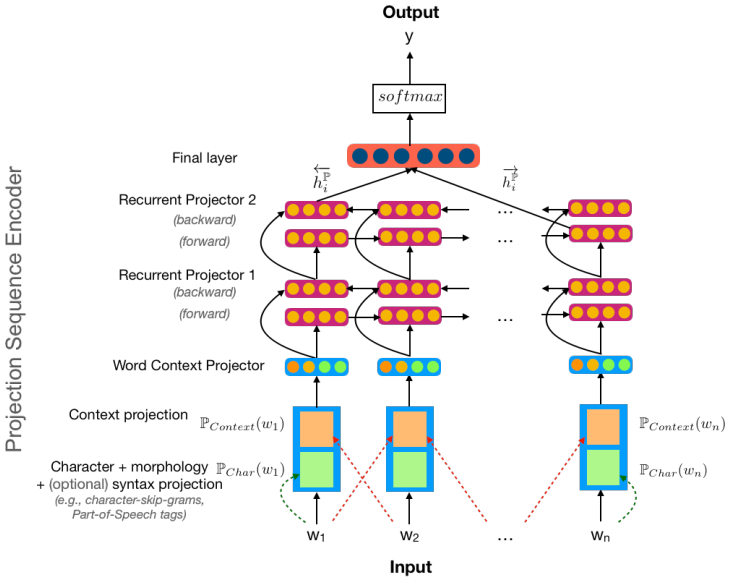


Figure 1: Model architecture for ProSeqo: On-device Projection Sequence Neural Networks.

is particularly challenging for NLP tasks to handle large vocabularies and model *unknown* words missing from training data. Previous methods have proposed to overcome this by relying on character-level embeddings and other neural models like character-CNNs. However, these methods are often complex and slow to compute for long text (e.g., convolution kernels on devices without significant computational capacity) and still require explicitly storing character or sub-word sequences. For example, even simple character trigrams over 26 characters in the English alphabet requires storing and looking up $V = 26^3 = 18K$ entries in character embedding table and complexity of $O(V \cdot d)$, where $d$ is the dimensionality of the embedding. For word embeddings, $V$ can be as large as 100K to millions of entries.

We extract character-level sequence information (i.e., character sequences) and use locality-sensitive projections (Ravi, 2017, 2019) to dynamically project them to a vector representation $\mathbb{P}_{Char}(w_{it})$ with just $O(T \cdot d)$ complexity, where $T << V$.

$$\mathbb{P}_{Char}(w_{it}) = \mathbb{P}(w_{itc_{1\ldots C}}) \quad (1)$$
$$= \mathbb{P}(\overrightarrow{w_{it}}) \quad (2)$$

where, $w_{itc_{1\ldots C}}$ is the sequence of characters $c \in [1, C]$ in word $w_{it}$ and $\overrightarrow{w_{it}}$ is a sparse vector comprised of character-level features (character skip-grams) extracted from the word $w_{it}$. We can also extend this to include other morphological (e.g., stemming) or syntax features (e.g., Part-of-Speech) but this adds further complexity and requires lexicons or further pre-processing steps (e.g., stemmers, taggers) which may not be available on device during inference. Hence, we only use simple raw features like character skip-grams in the projector.

$$\mathbb{P}(\overrightarrow{w_{it}}) = \texttt{LSH}_{proj}(\overrightarrow{w_{it}}, \overrightarrow{\mathbb{P}_{T,d}}) \quad (3)$$
$$= [\ sgn(\overrightarrow{w_{it}} \cdot \overrightarrow{\mathbb{P}_{1,1}}), \ldots, \quad (4)$$
$$\ldots, sgn(\overrightarrow{w_{it}} \cdot \overrightarrow{\mathbb{P}_{T,d}})\ ]$$

We then apply LSH projections that dynamically compute locality-sensitive projections and transform the extracted sparse vector $\overrightarrow{w_{it}}$ (character sequence features) into a compact, low-dimensional binary representation $\mathbb{P}(\overrightarrow{w_{it}})$. The projection functions $\overrightarrow{\mathbb{P}}$ used to compute the binary vector are dynamically generated from observed features in $\overrightarrow{w_{it}}$ and a random seed. $T, d$ represents the number of projection functions and dimensonality of each binary vector produced by each projection function. So, the output of the character-level projector is a $T \cdot d$ binary vector.

The locality-sensitive nature of the projections help learn a compact representation that capture semantic similarity (Sankar et al., 2019a) (i.e., words with similar character or morphology are mapped to the same binary representation) with a small memory footprint. For more details on LSH projections, refer to (Ravi, 2017).

#### 2.2.2 Context Projection

In addition to characters, the surrounding context also contains important information useful to represent in the word vector. Unlike characters though, word context is typically not fixed in language and changes across sentences and documents. To address this, we extend the projection mechanism to capture word meaning using contextual information.

$$\begin{aligned}
\mathbb{P}_{Context}(w_{it}) &= \mathbb{P}(context(w_{it})) && (5)\\
&= \mathbb{P}(w_{it-\Delta}...w_{it-1}, && (6)\\
&\quad\ w_{it+1}...w_{it+\Delta})\\
&= \mathbb{P}(\overrightarrow{w_{it,\Delta}}) && (7)\\
&= \texttt{LSH}_{proj}(\overrightarrow{w_{it,\Delta}}, \overrightarrow{\mathbb{P}_{T,d}}) && (8)
\end{aligned}$$

where, $\Delta$ represents the context size around the current word $w_{it}$, $\overrightarrow{w_{it,\Delta}}$ is the context features extracted from neighboring context. We use simple raw features like word unigrams to model the context and apply dynamic projections on the sparse context feature vector to transform this to a binary context projection vector.

### 2.3 Recurrent Projector

We use LSTM to model the state of word projection sequences within a sentence or document. This allows the model to capture words appearing in different parts of the input text, and use the projection space to learn an efficient neural representation for the task. At each time-step $t$, we compute the word projection by combining both character and context information, and transforming them via projections.

$$\mathbb{P}(x_t) = [\ \mathbb{P}_{Char}(x_t),\ \mathbb{P}_{Context}(x_t)\ ] \quad (9)$$

The LSTM-projector $\mathbb{P}_{\text{RNN}}$ then computes the new state $h_t^{\mathbb{P}}$ using the projector output as follows

$$\begin{aligned}
f_t^{\mathbb{P}} &= \sigma(W_f \cdot [h_{t-1}, \mathbb{P}(x_t)] + b_f) && (10)\\
i_t^{\mathbb{P}} &= \sigma(W_i \cdot [h_{t-1}, \mathbb{P}(x_t)] + b_i) && (11)\\
\widetilde{C_t^{\mathbb{P}}} &= \tanh(W_C \cdot [h_{t-1}, \mathbb{P}(x_t)] + b_C) && (12)\\
C_t^{\mathbb{P}} &= f_t^{\mathbb{P}} * C_{t-1}^{\mathbb{P}} + i_t^{\mathbb{P}} * \widetilde{C_t^{\mathbb{P}}} && (13)\\
o_t^{\mathbb{P}} &= \sigma(W_o \cdot [h_{t-1}, \mathbb{P}(x_t)] + b_o) && (14)\\
h_t^{\mathbb{P}} &= o_t^{\mathbb{P}} * \tanh(C_t^{\mathbb{P}}) && (15)
\end{aligned}$$

where, $f^{\mathbb{P}}, i^{\mathbb{P}}, o^{\mathbb{P}}$ represent the projection forget, input and output gates respectively, $\sigma$ is a non-linearity (ReLu) and $W, b$ are the weight and bias parameters. $h_t^{\mathbb{P}}$ is the new state computed from previous state $h_{t-1}^{\mathbb{P}}$. Note that we can replace LSTM with GRU (Cho et al., 2014) or other variants to control the gating mechanism and derive other recurrent projector model variants.

### 2.4 Projection Sequence Encoder

We use a bidirectional recurrent projector $\mathbb{P}_{\text{RNN}}$ that encodes the input word sequence by summarizing information from both directions for word projections. The sequence encoder captures both character-level and contextual information across time steps through the projection space $\Omega_{\mathbb{P}}$.

The projection sequence encoder contains a forward encoder $\overrightarrow{\mathbb{P}_{\text{RNN}}}$ that processes the sentence (or document) from $w_{i1}$ to $w_{iT}$ and the backward encoder $\overleftarrow{\mathbb{P}_{\text{RNN}}}$ which reads the input text in reverse.

$$\begin{aligned}
\overrightarrow{xh_i^{\mathbb{P}}} &= \overrightarrow{\mathbb{P}_{\text{RNN}}}(x_i) && (16)\\
\overleftarrow{h_i^{\mathbb{P}}} &= \overleftarrow{\mathbb{P}_{\text{RNN}}}(x_i) && (17)\\
h_i &= [\ \overrightarrow{h_i^{\mathbb{P}}},\ \overleftarrow{h_i^{\mathbb{P}}}\ ] && (18)
\end{aligned}$$

We concatenate the two states to model the state $h_i$ and transition of word projection sequences within a sentence or document.

**Stacked Recurrent Projections:** We also stack the recurrent projections to create deeper, multi-layered ProSeqo. Each layer is a bidirectional recurrent projection encoder with information from the forward and backward states passed to the next layer. In our work, we train ProSeqo with 2 layers.

### 2.5 Text Classifier

The final state of the projection sequence encoder $h_{iF}$ is combined with an output layer to learn a classifier for a given task.

$$y_i = \texttt{softmax}(W_F \cdot h_{iF} + b_F) \quad (19)$$

We train ProSeqo via backpropagation using stochastic gradient descent with negative log likelihood of correct labels as the loss.

## 3 Text Classification Tasks

In this section, we describe the tasks and datasets for our experimental evaluation. We choose the datasets to compare against prior state-of-the-art on-device work (Ravi and Kozareva, 2018), and also to test the generalizability of our on-device model on short and long texts.

| Text Classification Task | | #Classes | Vocabulary | Avg. Length | Train | Test |
|---|---|---|---|---|---|---|
| **SWDA** | Dialog Act | 42 | 20,000 | 7 | 193,000 | 5,000 |
| **MRDA** | Dialog Act | 6 | 12,000 | 8 | 78,000 | 15,000 |
| **ATIS** | Intent Prediction | 21 | 950 | 11 | 4,478 | 893 |
| **SNIPS** | Intent Prediction | 7 | 11,241 | 10 | 13,084 | 700 |
| **AG** | News Categorization | 4 | 156,000 | 38 | 120,000 | 7,600 |
| **Y!A** | Yahoo! Answers Categorization | 10 | 1,554,607 | 108 | 1,400,000 | 60,000 |
| **AMZN** | Product Review Prediction | 5 | 1,919,336 | 92 | 3,000,000 | 650,000 |

Table 1: Text Classification Tasks and Dataset Characteristics

### 3.1 Dataset Description

• **SWDA:** Switchboard Dialog Act Corpus is a popular open domain dialog corpus between two speakers with 42 dialog acts (Godfrey et al., 1992; Jurafsky et al., 1997). The dataset is used in the on-device work of (Ravi and Kozareva, 2018).

• **MRDA:** Meeting Recorder Dialog Act is a dialog corpus of multiparty meetings annotated with 6 dialog acts (Adam et al., 2003; Shriberg et al., 2004). The dataset was also used by on-device work of (Ravi and Kozareva, 2018). We used the same data split as (Lee and Dernoncourt, 2016; Ortega and Vu, 2017; Ravi and Kozareva, 2018).

• **ATIS:** The Airline Travel Information Systems dataset (Tür et al., 2010) is widely used in dialog and speech research. The dataset contains audio recordings of people making flight reservations (Tür et al., 2010; Goo et al., 2018).

• **SNIPS:** To test the generalizability of our model, we use another NLU dataset with custom intent-engines collected by Snips personal voice assistant. We used the data from (Goo et al., 2018). Compared to the single-domain ATIS dataset, Snips is more complicated mostly because of the rich and diverse intent repository combined with the larger vocabulary.

• **AG:** AG News corpus is a collection of news articles on the web, where each document has a title and a description field. We used the dataset from (Zhang et al., 2015).

• **Y!A:** Yahoo! Answers is a text classification task with 10 diverse classes: *Society & Culture*, *Science & Mathematics*, *Health*, *Education & Reference*, *Computers & Internet*, *Sports*, *Business & Finance*, *Entertainment & Music*, *Family & Relationships* and *Politics & Government*. Each document contains a question title, question context and best answers. We obtained the data from (Zhang et al., 2015). Note that (Yang et al., 2016) used a smaller test sample for evaluation. To present fair results, we will not compare to (Yang et al., 2016).

• **AMZN:** Amazon review dataset is obtained from (Zhang et al., 2015). Resolving this task can be very helpful for product categorization (Kozareva, 2015). The corpora has reviews with ratings ranging from 1 to 5. Following prior work, we use 3,000,000 reviews for training and 650,000 reviews for testing.

### 3.2 Dataset Characteristics

Table 1 shows the datasets characteristics such as the type of the task, number of classes, vocabulary size, average text length, train and test sizes. As shown in Table 1, the datasets are very diverse, some have few thousand training examples, while others have millions. The test sizes also differ. Similarly, there are tasks with small and huge vocabularies. These diverse characteristics help validate and capture the ability of our on-device model ProSeqo to generalize to different tasks, data constraints, settings and vocabulary. With respect to the text length, we have two major categories: short text tasks such as dialog act and intent prediction, and long text tasks such as news, product reviews and answer categorization.

### 3.3 Experimental Setup & Model Parameters

We setup our experiments as given a classification task and dataset, we apply ProSeqo to train an on-device model. For each task, we report *Accuracy* on the test set. Unlike typical NLP approaches including baseline neural methods, we do **not** apply any vocabulary pruning or pre-processing at all to the input sentences and documents, except for splitting tokens by space.

We use a 2-layer *ProSeqo* neural network with recurrent projections of size $T = 60, d = 14$ and 256 hidden dimensions to represent state in each bidirectional LSTM-projection layer. We use character 7-grams (with 1-skip) and context size of 1 to model the projector described in Section 2.2. ProSeqo network is trained with SGD and Adam

| Model | SWDA | MRDA | ATIS | SNIPS |
|---|---|---|---|---|
| **ProSeqo (our on-device model)** | **88.3** | **90.1** | **97.8** | **97.9** |
| **SGNN**(Ravi and Kozareva, 2018)(**on-device**) | 83.1 | 86.7 | *88.9* | *93.4* |
| RNN(Khanpour et al., 2016) | 80.1 | 86.8 | - | - |
| RNN+Attention(Ortega and Vu, 2017) | 73.8 | 84.3 | - | - |
| CNN(Lee and Dernoncourt, 2016) | 73.1 | 84.6 | - | - |
| GatedIntentAtten.(Goo et al., 2018) | - | - | 94.1 | 96.8 |
| GatedFullAtten.(Goo et al., 2018) | - | - | 93.6 | 97.0 |
| JointBiLSTM(Hakkani-Tur et al., 2016) | - | - | 92.6 | 96.9 |
| Atten.RNN(Liu and Lane, 2016) | - | - | 91.1 | 96.7 |

Table 2: Short Text Classification On-device Results & Comparisons to Prior Work

optimizer (Kingma and Ba, 2014) over shuffled mini-batches of size 100. We did not do any additional dataset-specific tuning or processing.

## 4 STC: Short Text Classification Results

This section focuses on the multiple short text classification experiments we have conducted. Table 2 shows the results on the dialog act and intent prediction tasks for SWDA, MRDA, ATIS and SNIPS datasets. Overall, ProSeqo reached the highest performance and significantly improved upon prior state-of-the-art on-device neural approaches (Ravi and Kozareva, 2018). Similarly, ProSeqo outperformed non-on-device neural models like RNN (Khanpour et al., 2016), CNN (Lee and Dernoncourt, 2016) and joint neural approaches (Goo et al., 2018; Hakkani-Tur et al., 2016).

### 4.1 STC: Comparison with On-Device Work

We compare our on-device model against state-of-the-art on-device short text classification approach SGNN (Ravi and Kozareva, 2018). SGNN was evaluated only on the SWDA and MRDA dialog act tasks (Ravi and Kozareva, 2018) and reached state-of-the-art performance against prior non-on-device neural approaches (Khanpour et al., 2016). We directly compare performance on the same SWDA and MRDA datasets. As shown in Table 2 ProSeqo reaches +5.3% improvement for SWDA and +3.4% accuracy improvement for MRDA.

Since we want to compare on-device performance on a wider spectrum of tasks and datasets, we re-implemented SGNN with the same parameters (Ravi and Kozareva, 2018). We run experiments on ATIS and SNIPS intent prediction tasks and reported results in Table 2 in italic to indicate that this is a re-implementation of (Ravi and Kozareva, 2018) and previously these on-device results were not reported. As shown in Table 2, SGNN consistently performs well on dialog act and intent prediction tasks. Overall, ProSeqo reached +8.9% accuracy improvements on ATIS and +4.5% accuracy improvements on SNIPS compared to SGNN. This shows that ProSeqo's recurrent dynamic projections learn more powerful representations than the static SGNN ones, this also leads to significant performance improvements on multiple tasks.

### 4.2 STC: Comparison with Non-On-Device

The main objective of on-device work is to develop small and efficient models that fit on devices with limited memory and capacity. In contrast, the non-on-device models do not have any constraints and can use all resources available on server side. Therefore, it is not fair to directly compare the on-device and non-on-device models. Taking these differences in consideration, we show in Table 2 results from non-on-device models with the objective to highlight the power of on-device models and their capability to produce small memory footprint models, which are competitive in accuracy to those using pre-trained embeddings and unconstrained resources.

As shown in Table 2, prior work focused on developing the best approach for a specific task and dataset, resulting in not having a single model across all tasks. ProSeqo is the only approach spanning across all tasks and datasets. In terms of SWDA and MRDA, ProSeqo significantly improves with +8.2% accuracy and +3.3% accuracy the best performing non-on-device neural approach (Khanpour et al., 2016). For ATIS and SNIPs, the most recent state-of-art approaches use joint intent and slot prediction model (Hakkani-Tur et al., 2016; Liu and Lane, 2016; Goo et al., 2018), where the slot model recognizes the enti-

| Model | AG | Y!A | AMZN |
|---|---|---|---|
| **ProSeqo (our on-device model)** | **91.5** | **72.4** | **62.3** |
| **SGNN** (Ravi and Kozareva, 2018)(**on-device**) | *57.6* | *36.5* | *39.3* |
| FastText-full(Joulin et al., 2016) | **92.5** | 72.3 | 60.2 |
| CharCNNLargeWithThesau.(Zhang et al., 2015) | 90.6 | 71.2 | 59.6 |
| CNN+NGM(Bui et al., 2018) | 86.9 | - | - |
| LSTM-full(Zhang et al., 2015) | 86.1 | 70.8 | 59.4 |
| Hier.-Attention(Yang et al., 2016) | - | - | **63.6** |
| Hier.-AVE(Yang et al., 2016) | - | - | **62.9** |

Table 3: Long Text Classification On-device Results & Comparisons to Prior Work

ties and their semantic categories in the text, and the intent prediction model uses the slot information. (Liu and Lane, 2016) showed that joint intent and slot model improves upon the individual ones. While the entities and their semantic types are useful, ProSeqo does not rely on such additional information. ProSeqo simply uses the text and different context to learn compact semantic projection representations on-the-fly to make the intent prediction. ProSeqo improves with +3.7% the best approach for ATIS, which is a gated intent attention (Goo et al., 2018), and with +1% the best approach for SNIPS, which is gated full attention (Goo et al., 2018). (Goo et al., 2018) developed two complimentary approaches one with full attention and one with gated intent attention, yet they do not have consistent improvements on both datasets and tasks. In our case, our on-device ProSeqo model consistently improved upon both tasks, and reached state-of-the-art performance over prior non-on-device neural approaches. This is a significant improvement given the small size of ProSeqo and the fact that it does not store or lookup any pre-trained embedding matrices.

## 5 LDC: Long Document Classification Results

This section focuses on long document classification results. Table 3 shows the results on three tasks and datasets (AG, Y!A, AMZN). Overall, ProSeqo significantly improved upon the on-device neural model SGNN (Ravi and Kozareva, 2018) with +23% to +35.9% accuracy. ProSeqo also reached comparable performance to prior non-on-device neural LSTMs and character CNNs approaches (Zhang et al., 2015; Bui et al., 2018).

### 5.1 LDC: Comparison with On-Device Work

To the best of our knowledge, ProSeqo is the first on-device neural approach for long document classification. Since prior on-device work focused mostly on short text, we re-implemented the on-device state-of-the-art neural approach SGNN (Ravi and Kozareva, 2018) and run it on multiple long text tasks and datasets. Similarly to the short text section, we reported SGNN results in Table 3 in italic to indicate that this is a re-implementation of (Ravi and Kozareva, 2018) and previously results on these datasets and tasks were not reported.

As shown in Table 3, SGNN performed well on short texts, but it did not reach high performance on long documents. It it important to highlight that ProSeqo maintained consistent performance across all short and long document classification tasks. For long texts, ProSeqo improved SGNN with +33.9% accuracy on AG news, with +35.9% accuracy on Y!A and with +23% accuracy on Amazon product review classification. These are significant improvements and further show the importance of ProSeqo's recurrent dynamic projections for short and long text classification.

### 5.2 LDC: Comparison with Non-On-Device

Similarly to the short text classification, we also conduct a comparison of non-on-device model for long document classification. As highlighted before, such comparison is not fair due to the fact that non-on-device models do not have to comply with small memory, small size and low latency constraints. Bearing this in mind, we show results in Table 3. As it can be seen, only FastText (Joulin et al., 2016), CharCNN with Thesaurus (Zhang et al., 2015) and LSTM-full (Zhang et al., 2015) have been evaluated on all datasets, while CNN+NGM (Bui et al., 2018) focused on reaching the best performance for AG news. ProSeqo outperforms (Joulin et al., 2016; Zhang et al., 2015,?; Bui et al., 2018), and reaches comparable results to FastText (Joulin et al., 2016) on AG news. The hierarchical attention and hierarchi-

cal average models of (Yang et al., 2016) were evaluated only on the Amazon reviews and Y!A datasets. However, we noticed that the test set of the Y!A dataset in (Yang et al., 2016) was smaller compared to the data used in (Zhang et al., 2015; Joulin et al., 2016) and our experiments. To make consistent and fair comparisons, we did not include (Yang et al., 2016). For Amazon reviews, ProSeqo reached comparable performance to hierarchical average and hierarchical attention models (Yang et al., 2016). Overall, ProSeqo reached similar performance to non-on-device neural approaches, which use pre-trained embeddings while ProSeqo does not store or lookup any pre-trained embedding matrices.

## 6 ProSeqo Ablation Studies

This section shows two ablation studies focusing on the: (1) effect of the projection size on accuracy and (2) model size on accuracy.

### 6.1 Effect of Projection Size on Accuracy

During series of evaluations on multiple short and long text classification tasks, and comparing results against prior state-of-the-art neural networks and on-device neural models, ProSeqo achieved robust performance irrespective of the task, vocabulary size, training and test datasets. This is significant improvement given the small memory size of the model and the fact that we used the same model, configuration and model parameters across all seven NLP tasks unlike prior work which fine-tuned on specific data.

We also evaluated ProSeqo with different number of parameters $T$, $d$ used in the recurrent projection. Table 4 shows the performance of ProSeqo on the AG News classification task and indicates that even if we reduce the projection size by 3x, we still achieve 90% accuracy with a small drop in performance of 1.4% (91.5 $\rightarrow$ 90.09).

| Projection Size ($T \cdot d$) | 300 | 600 | 800 |
|---|---|---|---|
| Compression Ratio ($\frac{S_1}{S_2}$) | $15 \times 10^4$ | $7.5 \times 10^4$ | $5.6 \times 10^4$ |
| Accuracy | 90.0 | 91.0 | **91.5** |

Table 4: Effect of ProSeqo's Recurrent Projection Size on Accuracy for AG News Classification. *Compression Ratio* is computed wrt ProSeqo model size ($S_1$) vs. LSTM baseline ($S_2$) with word embeddings (vocabulary size=150K, embedding size=300).

### 6.2 ProSeqo's Model Size

In addition to the quality evaluations, comparisons and results, we also highlight how compact and fast our on-device model is, which is very important for fitting the model on small memory-constrained devices such as mobile phones and watches. For this purpose, we measure the current size of our ProSeqo model. We do not have the exact parameter counts and model sizes from the previous neural approaches such as RNNs, BiLSTMs, and Hierarchical Attention models. But we make an observation that number of parameters in the recurrent layers of these baseline networks are similar to the parameters encoded in the recurrent projector layers in the ProSeqo model. Hence we can approximately compute the difference in model sizes between ProSeqo and other neural models by taking into consideration the size of embeddings fed into the recurrent layers along with the training data vocabulary size.

As highlighted in the main contributions of our approach ProSeqo is an embedding-free method, which means that it does not have to store or look up any embedding matrices. This makes the model small in comparison to existing neural approaches. For instance, to calculate the model size of ProSeqo as explained in Section 2 we need to consider $(T \cdot d)$ where $d$ is the number of LSH bits specified for each projection vector, and $T$ is the number of projection functions used resulting in $(T \cdot d) = 60 \times 14$ parameters shared across all time-steps. Independent of the task, length of the text (*short* vs *long*) and vocabulary sizes, ProSeqo's model size remains constant $(T \cdot d)$ across all seven tasks and datasets. Comparing ProSeqo's model size against a neural approach that uses Glove (or word2vec) word embeddings with 300 dimensions and vocabulary size $V$ for the datasets, we can easily see that the neural models will have bigger sizes and their model size also varies depending on the task and vocabulary size. For instance, the vocabulary size in AG dataset is $150K$ resulting in $150K \times 300$ more model parameters than ProSeqo. For Yahoo this is $1.5M \times 300$ and for Amazon model size is $1.9M \times 300$. As it can be seen, ProSeqo produces models that are orders of magnitude smaller compared to prior neural approaches. On AG dataset, ProSeqo model has $5 \cdot 10^4$x fewer parameters for the first layer which is equivalent to just 3.3KB in size using `float32` parameters compared to 180MB for other neural methods. ProS-

eqo's size remains constant independent of the text lengths, vocabulary size and classification tasks. Even with neural models like charCNN that use character-level embeddings which is typically applied in conjunction with word embeddings, ProSeqo still yields an order or two magnitude smaller model sizes. The computation within ProSeqo is fast, $O(T \cdot d \cdot n)$ `multiply-add` operations for $n$-length words regardless of word/character vocabulary sizes.

## 7 Conclusion

We proposed a novel Projection Sequence Neural Network (ProSeqo) for on-device text classification. Evaluations on multiple text classification tasks show that ProSeqo significantly improved accuracy compared to state-of-the-art on-device neural network SGNN (Ravi and Kozareva, 2018) on short text with +3.4% for MRDA, +4.5% on SNIPS, +5.3% on SWDA and +8.9% on ATIS; and long documents with +23% on Amazon product reviews, +33.9% on AG news, +35.9% on Y!A. Similarly, ProSeqo improved non-on-device neural RNN, CNN and joint approaches (Khanpour et al., 2016; Lee and Dernoncourt, 2016; Hakkani-Tur et al., 2016; Goo et al., 2018) on short text and reached comparable results on long documents (Joulin et al., 2016; Zhang et al., 2015; Bui et al., 2018). This is significant improvement given ProSeqo's small network size, and the fact that we used the same architecture and parameters across all seven NLP tasks, while prior work tuned depending on the task and dataset.

## References


Janin Adam, Don Baron, Jane Edwards, Dan Ellis, David Gelbart, Nelson Morgan, Barbara Peskin, Thilo Pfau, Elizabeth Shriberg, Andreas Stolcke, and Chuck Wooters. 2003. The icsi meeting corpus. In *Proceedings of the 5TH SIGdial Workshop on Discourse and Dialogue*, pages 364–367.

Thang D. Bui, Sujith Ravi, and Vivek Ramavajjala. 2018. Neural graph learning: Training neural networks using graphs. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, WSDM '18, pages 64–71.

Kyunghyun Cho, Bart van Merriënboer, Çalar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Jordan Cohen. 2008. Embedded speech recognition applications in mobile phones: Status, trends, and challenges. pages 5352 – 5355.

John J. Godfrey, Edward C. Holliman, and Jane McDaniel. 1992. Switchboard: Telephone speech corpus for research and development. In *Proceedings of the 1992 IEEE International Conference on Acoustics, Speech and Signal Processing - Volume 1*, ICASSP'92, pages 517–520. IEEE Computer Society.

Chih-Wen Goo, Guang Gao, Yun-Kai Hsu, Chih-Li Huo, Tsung-Chieh Chen, Keng-Wei Hsu, and Yun-Nung Chen. 2018. Slot-gated modeling for joint slot filling and intent prediction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 753–757. Association for Computational Linguistics.

Dilek Hakkani-Tur, Gokhan Tur, Asli Celikyilmaz, Yun-Nung Vivian Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In *Proceedings of The 17th Annual Meeting of the International Speech Communication Association (INTERSPEECH 2016)*.

Yanzhang He, Rohit Prabhavalkar, Kanishka Rao, Wei Li, Anton Bakhtin, and Ian McGraw. 2017. Streaming small-footprint keyword spotting using sequence-to-sequence models. *CoRR*, abs/1710.09617.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hervé Jégou, and Tomas Mikolov. 2016. Fasttext.zip: Compressing text classification models. *CoRR*, abs/1612.03651.

Daniel Jurafsky, Rebecca Bates, Rachel Martin Noah Coccaro, Marie Meteer, Klaus Ries, Elizabeth Shriberg, Audreas Stolcke, Paul Taylor, and Van Ess-Dykema. 1997. Automatic detection of discourse structure for speech recognition and understanding. In *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 88–95.

Hamed Khanpour, Nishitha Guntakandla, and Rodney Nielsen. 2016. Dialogue act classification in domain-independent conversations using a deep recurrent neural network. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2012–2021.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Zornitsa Kozareva. 2015. Everyone likes shopping! multi-class product categorization for e-commerce. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1329–1333.

Ji Young Lee and Franck Dernoncourt. 2016. Sequential short-text classification with recurrent and convolutional neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 515–520.

Zhong Qiu Lin, Audrey G. Chung, and Alexander Wong. 2018. Edgespeechnets: Highly efficient deep neural networks for speech recognition on the edge. *CoRR*, abs/1810.08559.

Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. *Proceedings of The 17th Annual Meeting of the International Speech Communication Association (INTERSPEECH 2016)*.

Daniel Ortega and Ngoc Thang Vu. 2017. Neural-based context representation learning for dialog act classification. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 247–252.

Sujith Ravi. 2017. Projectionnet: Learning efficient on-device deep networks using neural projections. *CoRR*, abs/1708.00630.

Sujith Ravi. 2019. Efficient on-device models using neural projections. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5370–5379.

Sujith Ravi and Zornitsa Kozareva. 2018. Self-governing neural networks for on-device short text classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 804–810.

Sujith Ravi and Zornitsa Kozareva. 2019. On-device structured and context partitioned projection networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3784–3793.

Chinnadhurai Sankar, Sujith Ravi, and Zornitsa Kozareva. 2019a. On the robustness of projection neural networks for efficient text representation: An empirical study. *ArXiv*, abs/1908.05763.

Chinnadhurai Sankar, Sujith Ravi, and Zornitsa Kozareva. 2019b. Transferable neural projection representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3355–3360.

Elizabeth Shriberg, Raj Dhillon, Sonali Bhagat, Jeremy Ang, and Hannah Carvey. 2004. The icsi meeting recorder dialog act (mrda) corpus. In *Proceedings of the 5th SIGdial Workshop on Discourse and Dialogue*, pages 97–100, Cambridge, Massachusetts, USA. Association for Computational Linguistics.

Gökhan Tür, Dilek Hakkani-Tür, and Larry P. Heck. 2010. What is left to be understood in atis? In *Proceedings of 2010 IEEE Spoken Language Technology Workshop (SLT)*, pages 19–24.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489. Association for Computational Linguistics.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 649–657. MIT Press.

# Induction Networks for Few-Shot Text Classification

**Ruiying Geng[1,2], Binhua Li[2], Yongbin Li[2]* , Xiaodan Zhu[3], Ping Jian[1]*, Jian Sun[2]**
[1]School of Computer Science and Technology, Beijing Institute of Technology
[2] Alibaba Group, Beijing
[3] ECE, Queen's University
{ruiying.gry,binhua.lbh,shuide.lyb,jian.sun}@alibaba-inc.com
zhu2048@gmail.com pjian@bit.edu.cn

## Abstract

Text classification tends to struggle when data is deficient or when it needs to adapt to unseen classes. In such challenging scenarios, recent studies have used meta-learning to simulate the few-shot task, in which new queries are compared to a small support set at the sample-wise level. However, this sample-wise comparison may be severely disturbed by the various expressions in the same class. Therefore, we should be able to learn a general representation of each class in the support set and then compare it to new queries. In this paper, we propose a novel Induction Network to learn such a generalized class-wise representation, by innovatively leveraging the dynamic routing algorithm in meta-learning. In this way, we find the model is able to induce and generalize better. We evaluate the proposed model on a well-studied sentiment classification dataset (English) and a real-world dialogue intent classification dataset (Chinese). Experiment results show that on both datasets, the proposed model significantly outperforms the existing state-of-the-art approaches, proving the effectiveness of class-wise generalization in few-shot text classification.

## 1 Introduction

Deep learning has achieved a great success in many fields such as computer vision, speech recognition and natural language processing (Kuang et al., 2018). However, supervised deep learning is notoriously greedy for large labeled datasets, which limits the generalizability of deep models to new classes due to annotation cost. Humans on the other hand are readily capable of rapidly learning new classes of concepts with few examples or stimuli. This notable gap provides a fertile ground for further research.

Few-shot learning is devoted to resolving the data deficiency problem by recognizing novel classes from very few labeled examples. The limitation of only one or very few examples challenges the standard fine-tuning method in deep learning. Early studies (Salamon and Bello, 2017) applied data augmentation and regularization techniques to alleviate the overfitting problem caused by data sparseness, only to a limited extent. Instead, researchers have explored meta-learning (Finn et al., 2017) to leverage the distribution over similar tasks, inspired by human learning. Contemporary approaches to few-shot learning often decompose the training procedure into an auxiliary meta-learning phase, which includes many meta-tasks, following the principle that the testing and training conditions must match. They extract some transferable knowledge by switching the meta-task from mini-batch to mini-batch. As such, few-shot models can classify new classes with just a small labeled support set.

However, existing approaches for few-shot learning still confront many important problems, including the imposed strong priors (Fei-Fei et al., 2006), complex gradient transfer between tasks (Munkhdalai and Yu, 2017), and fine-tuning the target problem (Qi et al., 2018). The approaches proposed by Snell et al. (2017) and Sung et al. (2018), which combine non-parametric methods and metric learning, provide potential solutions to some of those problems. The non-parametric methods allow novel examples to be rapidly assimilated, without suffering from catastrophic overfitting. Such non-parametric models only need to learn the representation of the samples and the metric measure. However, instances in the same class are interlinked and have their uniform fraction and their specific fractions. In previous studies, the class-level representations are calculated by simply summing or averaging represen-

*Corresponding authors: Y.Li and P.Jian.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3904–3913,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

tations of samples in the support set. In doing so, essential information may be lost in the noise brought by various forms of samples in the same class. Note that few-shot learning algorithms do not fine-tune on the support set. When increasing size of the support set, the improvement brought by a bigger data size will also be diminished by more sample level noises.

Instead, we explore a better approach by performing induction at the class-wise level: ignoring irrelevant details and encapsulating general semantic information from samples with various linguistic forms in the same class. As a result, there is a need for a perspective architecture that can reconstruct hierarchical representations of support sets and dynamically induce sample representations to class representations.

Recently, capsule network (Sabour et al., 2017) has been proposed, which possesses the exciting potential to address the aforementioned issue. A capsule network uses "capsules" that perform dynamic routing to encode the intrinsic spatial relationship between parts and whole that constitutes viewpoint invariant knowledge. Following a similar spirit, we can regard samples as parts and class as a *whole*. We propose the Induction Networks, which aims to model the ability of learning generalized class-level representation from samples in a small support set, based on the dynamic routing process. First, an Encoder Module generates representations for a query and support samples. Next, an Induction Module executes a dynamic routing procedure, in which the matrix transformation can be seen as a map from the sample space to the class space, and then the generation of the class representation is all depending on the routing-by-agreement procedure other than any parameters, which renders a robust induction ability to the proposed model to deal with unseen classes. By regarding the samples' representations as input capsules and the classes' as output capsules, we expect to recognize the semantics of classes that is invariant to sample-level noise. Finally, the interaction between queries and classes is modelled—their representations are compared by a Relation Module to determine if the query matches the class or not. Defining an episode-based meta-training strategy, the holistic model is meta-trained end-to-end with the generalizability and scalability to recognize unseen classes.

The specific contributions of our work are listed as follows:

- We propose the Induction Networks for few-shot text classification. To deal with sample-wise diversity in the few-shot learning task, our model is the first, to the best of our knowledge, that explicitly models the ability to induce class-level representations from small support sets.
- The proposed Induction Module combines the dynamic routing algorithm with typical meta-learning frameworks. The matrix transformation and routing procedure enable our model to generalize well to recognize unseen classes.
- Our method outperforms the current state-of-the-art models on two few-shot text classification datasets, including a well-studied sentiment classification benchmark and a real-world dialogue intent classification dataset.

## 2 Related Work

### 2.1 Few-Shot Learning

The seminal work on few-shot learning dates back to the early 2000s (Fe-Fei et al., 2003; Fei-Fei et al., 2006). The authors combined generative models with complex iterative inference strategies. More recently, many approaches have used a meta-learning (Finn et al., 2017; Mishra et al., 2018) strategy in the sense that they extract some transferable knowledge from a set of auxiliary tasks, which then helps them to learn the target few-shot problem well without suffering from overfitting. In general, these approaches can be divided into two categories.

**Optimization-based Methods** This type of approach aims to learn to optimize the model parameters given the gradients computed from the few-shot examples. Munkhdalai and Yu (2017) proposed the Meta Network, which learnt the meta-level knowledge across tasks and shifted its inductive biases via fast parameterization for rapid generalization. Mishra et al. (2018) introduced a generic meta-learning architecture called SNAIL which used a novel combination of temporal convolutions and soft attention.

**Distance Metric Learning** These approaches are different from the above approaches that entail some complexity when learning the target few-

shot problem. The core idea in metric-based few-shot learning is similar to nearest neighbours and kernel density estimation. The predicted probability over a set of known labels is a weighted sum of labels of support set samples. Vinyals et al. (2016) produced a weighted K-nearest neighbour classifier measured by the cosine distance, which was called Matching Networks. Snell et al. (2017) proposed the Prototypical Networks which learnt a metric space where classification could be performed by computing squared Euclidean distances to prototype representations of each class. Different from fixed metric measures, the Relation Network learnt a deep distance metric to compare the query with given examples (Sung et al., 2018).

Recently, some studies have been presented focusing specifically on few-shot text classification problems. Xu et al. (2018) studied lifelong domain word embeddings via meta-learning. Yu et al. (2018) argued that the optimal meta-model may vary across tasks, and they employed the multi-metric model by clustering the meta-tasks into several defined clusters. Rios and Kavuluru (2018) developed a few-shot text classification model for multi-label text classification where there was a known structure over the label space. Xu et al. (2019) proposed a open-world learning model to deal with the unseen classes in the product classification problem. We solve the few-shot learning problem from a different perspective and propose a dynamic routing induction method to encapsulate the abstract class representation from samples, achieving state-of-the-art performances on two datasets.

### 2.2 Capsule Network

The Capsule Network was first proposed by Sabour et al. (2017), which allowed the network to learn robustly the invariants in part-whole relationships. Lately, Capsule Network has been explored in the natural language processing field. Yang et al. (2018) successfully applied Capsule Network to fully supervised text classification problem with large labeled datasets. Unlike their work, we study few-shot text classification. Xia et al. (2018) reused the supervised model similar to that of Yang et al. (2018) for intent classification, in which a capsule-based architecture is extended to compute similarity between the target intents and source intents. Unlike their work, we propose Induction Networks for few-shot learning, in which we propose to use capsules and dynamic routing to learn generalized class-level representation from samples based. The dynamic routing method makes our model generalize better in the few-shot text classification task.

## 3 Problem Definition

### 3.1 Few-Shot Classification

Few-shot classification (Vinyals et al., 2016; Snell et al., 2017) is a task in which a classifier must be adapted to accommodate new classes not seen in training, given only a few examples of each of these new classes. We have a large labeled training set with a set of classes $C_{train}$. However, after training, our ultimate goal is to produce classifiers on the testing set with a disjoint set of new classes $C_{test}$, for which only a small labeled support set will be available. If the support set contains $K$ labeled examples for each of the $C$ unique classes, the target few-shot problem is called a $C$-way $K$-shot problem. Usually, the $K$ is too small to train a supervised classification model. Therefore, we aim to perform meta-learning on the training set, and extract transferable knowledge that will allow us to deliver better few-shot learning on the support set and thus classify the test set more accurately.

### 3.2 Training Procedure

The training procedure has to be chosen carefully to match inference at test time. An effective way to exploit the training set is to decompose the training procedure into an auxiliary meta-learning phase and mimic the few-shot learning setting via episode-based training, as proposed in Vinyals et al. (2016). We construct an meta-episode to compute gradients and update our model in each training iteration. The meta-episode is formed by randomly selecting a subset of classes from the training set first, and then choosing a subset of examples within each selected class to act as the support set $S$ and a subset of the remaining examples to serve as the query set $Q$. The meta-training procedure explicitly learns to learn from the given support set $S$ to minimise a loss over the query set $Q$. We call this strategy as episode-based meta training, and the details are shown in Algorithm 1. It is worth noting that there are exponentially many possible meta tasks to train the model on, making it hard to overfit. For example, if a dataset contains 159 training classes, this leads to

**Algorithm 1** Episode-Based Meta Training

1: **for** each *episode iteration* **do**
2: Randomly select $C$ classes from the class space of the training set;
3: Randomly select $K$ labeled samples from each of the $C$ classes as support set $S = \{(x_s, y_s)\}_{s=1}^{m}$ $(m = K \times C)$, and select a fraction of the reminder of those $C$ classes' samples as query set $Q = \{(x_q, y_q)\}_{q=1}^{n}$;
4: Feed the support set $S$ to the model and update the parameters by minimizing the loss in the query set $Q$;
5: **end for**

$$\begin{pmatrix} 159 \\ 5 \end{pmatrix} = 794,747,031 \text{ possible } 5-way \text{ tasks.}$$

## 4 The Models

Our Induction Networks, depicted in Figure 1 (the case of 3-way 2-shot model), consists of three modules: Encoder Module, Induction Module and Relation Module. In the rest of this section, we will show how these modules work in each meta-episode.

### 4.1 Encoder Module

This module is a bi-direction recurrent neural network with self-attention as shown in Lin et al. (2017). Given an input text $x = (w_1, w_2, ..., w_T)$, represented by a sequence of word embeddings. We use a bidirectional LSTM to process the text:

$$\overrightarrow{h_t} = \overrightarrow{LSTM}(w_t, h_{t-1}) \quad (1)$$

$$\overleftarrow{h_t} = \overleftarrow{LSTM}(w_t, h_{t+1}) \quad (2)$$

And we concatenate $\overrightarrow{h_t}$ with $\overleftarrow{h_t}$ to obtain a hidden state $h_t$. Let the hidden state size for each unidirectional LSTM be $u$. For simplicity, we note all the $T$ $h_t s$ as $H = (h_1, h_2, ..., h_T)$. Our aim is to encode a variable length of text into a fixed size embedding. We achieve that by choosing a linear combination of the $T$ $LSTM$ hidden vectors in $H$. Computing the linear combination requires the self-attention mechanism, which takes the whole LSTM hidden states $H$ as input, and outputs a vector of weights $a$:

$$a = softmax(W_{a2} tanh(W_{a1} H^T)) \quad (3)$$

here $W_{a1} \in R^{d_a \times 2u}$ and $W_{a2} \in R^{d_a}$ are weight matrices and $d_a$ is a hyperparameter. The final representation $e$ of the text is the weighted sum of $H$:

$$e = \sum_{t=1}^{T} a_t \cdot h_t \quad (4)$$

### 4.2 Induction Module

This section introduces the proposed dynamic routing induction algorithm. We regard these vectors $e$ obtained from the support set $S$ by Eq 4 as sample vectors $e^s$, and the vectors $e$ from the query set $Q$ as query vectors $e^q$. The most important step is to extract the representation for each class in the support set. The main purpose of the induction module is to design a non-linear mapping from sample vector $e_{ij}^s$ to class vector $c_i$:

$$\left\{e_{ij}^s \in R^{2u}\right\}_{i=1,...C, j=1...K} \mapsto \left\{c_i \in R^{2u}\right\}_{i=1}^{C}.$$

We apply the dynamic routing algorithm (Sabour et al., 2017) in this module, in the situation where the number of the output capsule is one. In order to accept $any$-way $any$-shot inputs in our model, a weight-sharable transformation across all sample vectors in the support set is employed. All of the sample vectors in the support set share the same transformation weights $W_s \in R^{2u \times 2u}$ and bias $b_s$, so that the model is flexible enough to handle the support set at any scale. Each sample prediction vector $\hat{e}_{ij}^s$ is computed by:

$$\hat{e}_{ij}^s = squash(W_s e_{ij}^s + b_s) \quad (5)$$

where $squash$ is a non-linear squashing function through the entire vector, which leaves the direction of the vector unchanged but decreases its magnitude. Given input vector $x$, $squash$ is defined as:

$$squash(x) = \frac{\|x\|^2}{1 + \|x\|^2} \frac{x}{\|x\|} \quad (6)$$

Eq 5 encodes important invariant semantic relationships between lower level sample features and higher level class features (Hinton et al., 2011). To ensure the class vector encapsulates the sample feature vectors of this class automatically, dynamic routing is applied iteratively. In each iteration, the process dynamically amends the connection strength and makes sure that the coupling coefficients $d_i$ sum to 1 between class $i$ and all support samples in this class by a "routing softmax":

$$d_i = softmax\left(b_i\right) \quad (7)$$

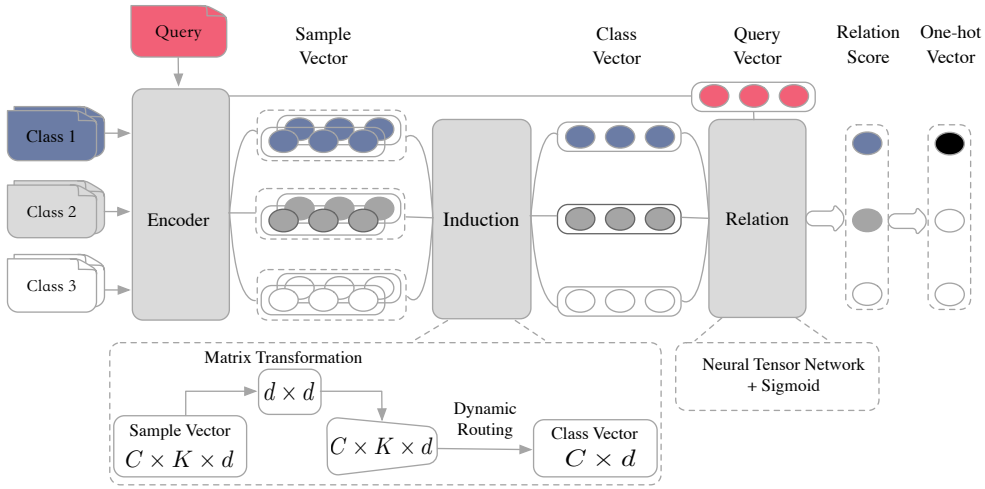


Figure 1: Induction Networks architecture for a $C$-way $K$-shot ($C = 3$, $K = 2$) problem with one query example

where $b_i$ is the logits of coupling coefficients, and initialized by 0 in the first iteration. Given each sample prediction vector $\hat{e}^s_{ij}$, each class candidate vector $\hat{c}_i$ is a weighted sum of all sample prediction vectors $\hat{e}^s_{ij}$ in class $i$:

$$\hat{c}_i = \sum_j d_{ij} \cdot \hat{e}^s_{ij} \tag{8}$$

then a non-linear "squashing" function is applied to ensure that the length of the vector output of the routing process will not exceed 1:

$$c_i = squash(\hat{c}_i) \tag{9}$$

The last step in every iteration is to adjust the logits of coupling coefficients $b_{ij}$ by a "routing by agreement" method. If the produced class candidate vector has a large scalar output with one sample prediction vector, there is a top-down feedback which increases the coupling coefficient for that sample and decreases it for other samples. This type of adjustment is very effective and robust for the few-shot learning scenario because it does not need to restore any parameters. Each $b_{ij}$ is updated by:

$$b_{ij} = b_{ij} + \hat{e}^s_{ij} \cdot c_i \tag{10}$$

Formally, we call our induction method as dynamic routing induction and summarize it in Algorithm 2.

**Algorithm 2** Dynamic Routing Induction

**Require:** sample vector $e^s_{ij}$ in support set $S$ and initialize the logits of coupling coefficients $b_{ij} = 0$

**Ensure:** class vector $c_i$

for all samples $j = 1, ..., K$ in class $i$:
  $\hat{e}^s_{ij} = squash(W_s e^s_{ij} + b_s)$
**for** *iter* iterations **do**
  $d_i = softmax(b_i)$
  $\hat{c}_i = \sum_j d_{ij} \cdot \hat{e}^s_{ij}$
  $c_i = squash(\hat{c}_i)$
  for all samples $j = 1, ..., K$ in class $i$:
    $b_{ij} = b_{ij} + \hat{e}^s_{ij} \cdot c_i$
**end for**
**Return** $c_i$

### 4.3 Relation Module

After the class vector $c_i$ is generated by the Induction Module and each query text in the query set is encoded to a query vector $e^q$ by the Encoder Module, the next essential procedure is to measure the correlation between each pair of query and class. The output of the Relation Module is called the relation score, representing the correlation between $c_i$ and $e^q$, which is a scalar between 0 and 1. Specifically, we use the neural tensor layer (Socher et al., 2013) in this module, which has shown great advantages in modeling the relationship between two vectors (Wan et al., 2016; Geng et al., 2017). We choose it as an interaction function in this paper. The tensor layer outputs a relation vector as follows:

$$v(c_i, e^q) = f\left({c_i}^T M^{[1:h]} e^q\right) \tag{11}$$

where $M^k \in R^{2u \times 2u}, k \in [1, ..., h]$ is one slice of the tensor parameters and $f$ is a non-linear activation function called RELU (Glorot et al., 2011).

The final relation score $r_{iq}$ between the $i$-th class and the $q$-th query is calculated by a fully connected layer activated by a sigmoid function.

$$r_{iq} = sigmoid(W_r v(c_i, e^q) + b_r) \quad (12)$$

### 4.4 Objective Function

We use the mean square error (MSE) loss to train our model, regressing the relation score $r_{iq}$ to the ground truth $y_q$: matched pairs have similarity 1 and the mismatched pair have similarity 0. Given the support set $S$ with $C$ classes and query set $Q = \{(x_q, y_q)\}_{q=1}^{n}$ in an episode, the loss function is defined as:

$$L(S, Q) = \sum_{i=1}^{C} \sum_{q=1}^{n} (r_{iq} - \mathbf{1}(y_q == i))^2 \quad (13)$$

conceptually we are predicting relation scores, which can be considered as a regression problem and the ground truth is within the space $\{0, 1\}$.

All parameters of the three modules are trained jointly by backpropagation. The Adagrad (Duchi et al., 2011) is used on all parameters in each training episode. Our model does not need any fine-tuning on the classes it has never seen due to its generalization nature. The induction and comparison ability are accumulated in the model along with the training episodes.

## 5 Experiments

We evaluate our model by conducting experiments on two few-shot text classification datasets. All the experiments are implemented with Tensorflow.

### 5.1 Datasets

**Amazon Review Sentiment Classification (ARSC)** Following Yu et al. (2018), we use the multiple tasks with the multi-domain sentiment classification (Blitzer et al., 2007) dataset. The dataset comprises English reviews for 23 types of products on Amazon. For each product domain, there are three different binary classification tasks. These buckets then form $23 \times 3 = 69$ tasks in total. Following Yu et al. (2018), we select $12(4 \times 3)$ tasks from 4 domains (Books, DVD, Electronics and Kitchen) as the test set, and there are only five examples as support set for each label in the test set. We create 5-shot learning models on this dataset.

| | Training Set | Testing Set |
|---|---|---|
| Class Num | 159 | 57 |
| Data Num | 195,775 | 2,279 |
| Data Num/Class | $\geq 77$ | $20 \sim 77$ |

Table 1: Details of ODIC

| Model | Mean Acc |
|---|---|
| Matching Networks (Vinyals et al., 2016) | 65.73 |
| Prototypical Networks (Snell et al., 2017) | 68.17 |
| Graph Network (Garcia and Bruna, 2017) | 82.61 |
| Relation Network (Sung et al., 2018) | 83.07 |
| SNAIL (Mishra et al., 2018) | 82.57 |
| ROBUSTTC-FSL (Yu et al., 2018) | 83.12 |
| Induction Networks (ours) | **85.63** |

Table 2: Comparison of mean accuracy (%) on ARSC

**Open Domain Intent Classification for Dialog System (ODIC)** We create this dataset by fetching the log data on a real-world conversational platform. The enterprises submit various dialogue tasks with a great number of intents, but many intents have only a few labeled samples, which is a typical few-shot classification application. Following the definition of the few-shot learning task, we divide the ODIC into a training set and a testing set and ensure that the labels of the two sets have no intersection. The details of the set partition are shown in Table 1.

### 5.2 Experiment Setup

**Baselines** In this section, the baseline models in our experiments are introduced as follows.

- Matching Networks: a few-shot learning model using a metric-based attention method (Vinyals et al., 2016).
- Prototypical Networks: a deep metric-based method using sample average as class prototypes (Snell et al., 2017).
- Graph Network: a graph-based few-shot learning model that implements a task-driven message passing algorithm on the sample-wise level (Garcia and Bruna, 2017).
- Relation Network: a few-shot learning model which uses a neural network as the distance metric and sums up sample vectors in the support set as class vectors (Sung et al., 2018).
- SNAIL: a class of simple and generic meta-learner architectures that use a novel combi-

| Model | 5-way Acc. | | 10-way Acc. | |
|---|---|---|---|---|
| | 5-shot | 10-shot | 5-shot | 10-shot |
| Matching Networks (Vinyals et al., 2016) | 82.54±0.12 | 84.63±0.08 | 73.64±0.15 | 76.72±0.07 |
| Prototypical Networks (Snell et al., 2017) | 81.82±0.08 | 85.83±0.06 | 73.31±0.14 | 75.97±0.11 |
| Graph Network (Garcia and Bruna, 2017) | 84.15±0.16 | 87.24±0.09 | 75.58±0.12 | 78.27±0.10 |
| Relation Network (Sung et al., 2018) | 84.41±0.14 | 86.93±0.15 | 75.28±0.13 | 78.61±0.06 |
| SNAIL (Mishra et al., 2018) | 84.62±0.16 | 87.31±0.11 | 75.74±0.07 | 79.26±0.09 |
| Induction Networks (ours) | **87.16**±0.09 | **88.49**±0.17 | **78.27**±0.14 | **81.64**±0.08 |

Table 3: Comparison of mean accuracy (%) on ODIC

nation of temporal convolutions and soft attention (Mishra et al., 2018).

- ROBUSTTC-FSL: This approach combines several metric-based methods by clustering the tasks (Yu et al., 2018).

The baseline results on ARSC are reported in Yu et al. (2018) and we implemented the baseline models on ODIC with the same text encoder module.

**Implementation Details** We use 300-dimension Glove embeddings (Pennington et al., 2014) for ARSC dataset and 300-dimension Chinese word embeddings trained by Li et al. (2018) for ODIC. We set the hidden state size of LSTM $u = 128$ and the attention dimension $d_a = 64$. The iteration number $iter$ used in dynamic routing algorithm is 3. The relation module is a neural tensor layer with $h = 100$ followed by a fully connected layer activated by sigmoid. We build 2-way 5-shot models on ARSC following Yu et al. (2018), and build episode-based meta training with $C = [5, 10]$ and $K = [5, 10]$ for comparison on ODIC. In addition to $K$ sample texts as support set, the query set has 20 query texts for each of the $C$ sampled classes in every training episode. This means, for example, that there are $20 \times 5 + 5 \times 5 = 125$ texts in one training episode for the 5-way 5-shot experiments.

**Evaluation Methods** We evaluate the performance by few-shot classification accuracy following previous studies in few-shot learning (Snell et al., 2017; Sung et al., 2018). To evaluate the proposed model with the baselines objectively, we compute mean few-shot classification accuracies on ODIC over 600 randomly selected episodes from the testing set. We sample 10 test texts per class in each episode for evaluation in both 5-shot and 10-shot scenarios. Note that for ARSC, the support set for testing is fixed by Yu et al. (2018). Consequently, we just need to run the test episode once for each of the target tasks. The mean accuracy of the 12 target task is compared to the baseline models following Yu et al. (2018).

### 5.3 Experiment Results

**Overall Performance** Experiment results on ARSC are presented in Table 2. The proposed Induction Networks achieves a 85.63% accuracy, outperforming the existing state-of-the-art model, ROBUSTTC-FSL, by a notable 3% improvement. We due the improvement to the fact that ROBUSTTC-FSL builds a general metric method by integrating several metrics at the sample level, which faces the difficulty of getting rid of the noise among different expressions in the same class. In addition to that, the task-clustering-based method used by ROBUSTTC-FSL must be found on the relevance matrix, which is inefficient when applied to real-world scenarios where the tasks change rapidly. Our Induction Networks, however, is trained in the meta-learning framework with more flexible generalization and its induction ability can hence be accumulated through different tasks.

We also evaluate our method with a real-world intent classification dataset ODIC. The experiment results are listed in Table 3. We can see that our proposed Induction Networks achieves best classification performances on all of the four experiments. In the distance metric learning models (Matching Networks, Prototypical Networks, Graph Network and Relation Network), all the learning occurs in representing features and measuring distances at the sample-wise level. Our work builds an induction module focusing on the class-wise level of representation, which we claim to be more robust to variation of samples in the support set. Our model also outperforms the latest optimization-based method—SNAIL. The difference between Induction Networks and SNAIL shown in Table 3 is statistically significant under the paired at the 99% significance level. In

| | Iteration | Accuracy |
|---|---|---|
| Routing+Relation | 3 | **85.63** |
| Routing+Relation | 2 | 85.41 |
| Routing+Relation | 1 | 85.06 |
| Routing + Cosine | 3 | 84.67 |
| Sum +Relation | - | 83.07 |
| Attention + Relation | - | 85.15 |

Table 4: Ablation study of Induction Networks on ARSC dataset

addition, the performance difference between our model and other baselines in the 10-shot scenario is more significant than in the 5-shot scenario. This is because in the 10-shot scenario, for the baseline models the improvement brought by a bigger data size is also diminished by more sample level noises.

**Ablation Study** To analyze the effect of varying different components of the Induction Module and Relation Module, we further report the ablation experiments on the ARSC dataset as shown in Table 4. We can see that the best performance is achieved when we used 3 iterations, corresponding to the best result reported in Table 2 (more rounds of iterations did not further improve the performance), and the table shows the effectiveness of the routing component. We also changed the Induction Module with sum and self-attention and changed Relation Module with cosine distance. Changes in the performances validate the benefit of both the Relation Module and Induction Module. The Attention+Relation models the induction ability by self-attention mechanism, but the ability is limited by the learnt attention parameters. Conversely, the proposed dynamic routing induction method captures class-level information by automatically adjusting the coupling coefficients according to inputted support sets, which is more suitable for the few-shot learning task.

### 5.4 Further Analysis

We further analyze the effect of transformation and visualize query text vectors to show the advantage of the Induction Networks.

**Effect of Transformation** Figure 2 shows the t-SNE (Maaten and Hinton, 2008) visualization of support sample vectors before and after matrix transformation under the 5-way 10-shot scenario. We randomly select a support set with 50 texts (10 texts per class) from the ODIC testing set, and obtain the sample vectors $\left\{e^s_{ij}\right\}_{i=1,...5,j=1...10}$ after the encoder module and the sample prediction vector $\left\{\hat{e}^s_{ij}\right\}_{i=1,...5,j=1...10}$ after transformation. We can see that the vectors after matrix transformation are more separable, demonstrating the effectiveness of matrix transformation to encode semantic relationships between lower-level sample features and higher-level class features.




(a) Before transformation (b) After transformation

Figure 2: Effect of Transformation under the 5-way 10-shot scenario. (a) The support sample vectors before matrix transformation. (b) The support sample vectors after matrix transformation.

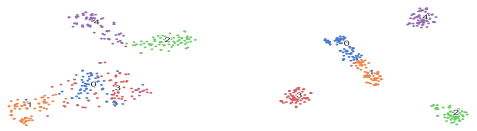

(a) Relation Network (b) Induction Networks

Figure 3: Query text vector visualization learnt by (a) Relation Network and (b) Induction Networks.

**Query Text Vector Visualization** We also find out that our induction module does not only work well in generating effective class-level features, but also helps the encoders to learn better text vectors, as it can give different weights to instances and features during backpropagation. Figure 3 shows the t-SNE (Maaten and Hinton, 2008) visualization of text vectors from the same randomly selected five classes, learnt by the Relation Network and our Induction Networks. It is clear that the text vectors learnt by Induction Networks are better separated semantically than those of Relation Network.

## 6 Conclusion

In this paper, we propose the Induction Networks, a novel neural model for few-shot text classification. We propose to induce the class-level representations from support sets to deal with sample-wise diversity in few-shot learning tasks. The Induction Module combines the dynamic routing algorithm with a meta-learning framework, and the

routing mechanism makes our model more general to recognize unseen classes. The experiment results show that the proposed model outperforms the existing state-of-the-art few-shot text classification models. We found that both the matrix transformation and routing procedure contribute consistently to the few-shot learning tasks.

## Acknowledgments

The authors would like to thank the organizers of EMNLP-IJCNLP2019 and the reviewers for their helpful suggestions. This research work is supported by the National Key Research and Development Program of China under Grant No. 2017YFB1002103, the National Natural Science Foundation of China under Grant No. 61751201, and the Research Foundation of Beijing Municipal Science & Technology Commission under Grant No. Z181100008918002.

## References

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th annual meeting of the association of computational linguistics*, pages 440–447.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Li Fe-Fei et al. 2003. A bayesian approach to unsupervised one-shot learning of object categories. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1134–1141. IEEE.

Li Fei-Fei, Rob Fergus, and Pietro Perona. 2006. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):594–611.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org.

Victor Garcia and Joan Bruna. 2017. Few-shot learning with graph neural networks. *CoRR*, abs/1711.04043.

Ruiying Geng, Ping Jian, Yingxue Zhang, and Heyan Huang. 2017. Implicit discourse relation identification based on tree structure neural network. In *2017 International Conference on Asian Language Processing (IALP)*, pages 334–337. IEEE.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323.

Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. 2011. Transforming auto-encoders. In *International Conference on Artificial Neural Networks*, pages 44–51. Springer.

Shaohui Kuang, Junhui Li, António Branco, Weihua Luo, and Deyi Xiong. 2018. Attention focusing for neural machine translation by bridging source and target embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1767–1776.

Shen Li, Zhe Zhao, Renfen Hu, Wensi Li, Tao Liu, and Xiaoyong Du. 2018. Analogical reasoning on chinese morphological and semantic relations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 138–143.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. 2018. A simple neural attentive meta-learner. *In Proceedings of ICLR*.

Tsendsuren Munkhdalai and Hong Yu. 2017. Meta networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2554–2563. JMLR. org.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Hang Qi, Matthew Brown, and David G Lowe. 2018. Low-shot learning with imprinted weights. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5822–5830.

Anthony Rios and Ramakanth Kavuluru. 2018. Few-shot and zero-shot multi-label learning for structured label spaces. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3132–3142.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. 2017. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems*, pages 3856–3866.

Justin Salamon and Juan Pablo Bello. 2017. Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal Processing Letters*, 24(3):279–283.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems*, pages 926–934.

Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. 2018. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1199–1208.

Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638.

Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. 2016. A deep architecture for semantic matching with multiple positional sentence representations. In *AAAI*, volume 16, pages 2835–2841.

Congying Xia, Chenwei Zhang, Xiaohui Yan, Yi Chang, and Philip Yu. 2018. Zero-shot user intent detection via capsule neural networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3090–3099.

Hu Xu, Bing Liu, Lei Shu, and P Yu. 2019. Open-world learning and application to product classification. In *The World Wide Web Conference*, pages 3413–3419. ACM.

Hu Xu, Bing Liu, Lei Shu, and Philip S Yu. 2018. Lifelong domain word embedding via meta-learning. *arXiv preprint arXiv:1805.09991*.

Min Yang, Wei Zhao, Jianbo Ye, Zeyang Lei, Zhou Zhao, and Soufei Zhang. 2018. Investigating capsule networks with dynamic routing for text classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3110–3119.

Mo Yu, Xiaoxiao Guo, Jinfeng Yi, Shiyu Chang, Saloni Potdar, Yu Cheng, Gerald Tesauro, Haoyu Wang, and Bowen Zhou. 2018. Diverse few-shot text classification with multiple metrics. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1206–1215.

# Benchmarking Zero-shot Text Classification: Datasets, Evaluation and Entailment Approach

**Wenpeng Yin, Jamaal Hay, Dan Roth**
Cognitive Computation Group
Department of Computer and Information Science, University of Pennsylvania
{wenpeng,jamaalh,danroth}@seas.upenn.edu

## Abstract

Zero-shot text classification (0SHOT-TC) is a challenging NLU problem to which little attention has been paid by the research community. 0SHOT-TC aims to associate an appropriate label with a piece of text, irrespective of the text domain and the aspect (e.g., topic, emotion, event, etc.) described by the label. And there are only a few articles studying 0SHOT-TC, all focusing only on topical categorization which, we argue, is just the tip of the iceberg in 0SHOT-TC. In addition, the chaotic experiments in literature make no uniform comparison, which blurs the progress.

This work benchmarks the 0SHOT-TC problem by providing unified datasets, standardized evaluations, and state-of-the-art baselines. Our contributions include: i) The datasets we provide facilitate studying 0SHOT-TC relative to conceptually different and diverse aspects: the "topic" aspect includes "sports" and "politics" as labels; the "emotion" aspect includes "joy" and "anger"; the "situation" aspect includes "medical assistance" and "water shortage". ii) We extend the existing evaluation setup (*label-partially-unseen*) – given a dataset, train on some labels, test on all labels – to include a more challenging yet realistic evaluation *label-fully-unseen* 0SHOT-TC (Chang et al., 2008), aiming at classifying text snippets without seeing task specific training data at all. iii) We unify the 0SHOT-TC of diverse aspects within a textual entailment formulation and study it this way. [1]

## 1 Introduction

Supervised text classification has achieved great success in the past decades due to the availability of rich training data and deep learning techniques. However, zero-shot text classification (0SHOT-TC) has attracted little attention despite its great potential in real world applications, e.g., the intent recognition of bank consumers. 0SHOT-TC is challenging because we often have to deal with classes that are compound, ultra-fine-grained, changing over time, and from different aspects such as topic, emotion, etc.

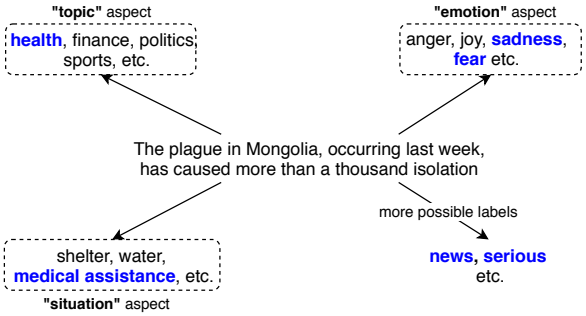


Figure 1: A piece of text can be assigned labels which describe the different aspects of the text. Positive labels are in blue.

Existing 0SHOT-TC studies have mainly the following three problems.

**First problem.** The 0SHOT-TC problem was modeled in a too restrictive vision. Firstly, most work only explored a single task, which was mainly topic categorization, e.g., (Pushp and Srivastava, 2017; Yogatama et al., 2017; Zhang et al., 2019). We argue that this is only the tiny tip of the iceberg for 0SHOT-TC. Secondly, there is often a precondition that a part of classes are seen and their labeled instances are available to train a model, as we define here as *Definition-Restrictive*:

*Definition-Restrictive (*0SHOT-TC*).* Given labeled instances belonging to a set of seen classes $S$, 0SHOT-TC aims at learning a classifier $f(\cdot) : X \rightarrow Y$, where $Y = S \cup U$; $U$ is a set of unseen classes and belongs to the same aspect as $S$.

In this work, we formulate the 0SHOT-TC in a

[1] https://cogcomp.seas.upenn.edu/page/publication_view/883



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3914–3923,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

broader vision. As Figure 1 demonstrates, a piece of text can be assigned labels which interpret the text in different aspects, such as the "topic" aspect, the "emotion" aspect, or the "situation" aspect described in the text. Different aspects, therefore, differ in interpreting the text. For instance, by "topic", it means "this text is about {health, finance $\cdots$}"; by "emotion", it means "this text expresses a sense of {joy, anger, $\cdots$}"; by "situation", it means "the people there need {shelter, medical assistance, $\cdots$}". Figure 1 also shows another essential property of 0SHOT-TC – the applicable label space for a piece of text has no boundary, e.g., "this text is news", "the situation described in this text is serious", etc. Therefore, we argue that we have to emphasize a more challenging scenario to satisfy the real-world problems: seeing no labels, no label-specific training data. Here is our new 0SHOT-TC definition:

*Definition-Wild (*0SHOT-TC*).* 0SHOT-TC aims at learning a classifier $f(\cdot) : X \rightarrow Y$, where classifier $f(\cdot)$ never sees $Y$-specific labeled data in its model development.

**Second problem.** Usually, conventional text classification denotes labels as indices {0,1,2, $\cdots$, $n$} without understanding neither the aspect's specific interpretation nor the meaning of the labels. This does not apply to 0SHOT-TC as we can not pre-define the size of the label space anymore, and we can not presume the availability of labeled data. Humans can easily decide the truth value of any upcoming labels because humans can interpret those aspects correctly and understand the meaning of those labels. The ultimate goal of 0SHOT-TC should be to develop machines to catch up with humans in this capability. To this end, making sure the system can understand the described aspect and the label meanings plays a key role.

**Third problem.** Prior work is mostly evaluated on different datasets and adopted different evaluation setups, which makes it hard to compare them fairly. For example, Rios and Kavuluru (2018) work on medical data while reporting R@K as metric; Xia et al. (2018) work on SNIPS-NLU intent detection data while only unseen intents are in the label-searching space in evaluation.

In this work, we benchmark the datasets and evaluation setups of 0SHOT-TC. Furthermore, we propose a textual entailment approach to handle the 0SHOT-TC problem of diverse aspects in a unified paradigm. To be specific, we contribute in the following three aspects:

**Dataset.** We provide datasets for studying three aspects of 0SHOT-TC: topic categorization, emotion detection, and situation frame detection – an event level recognition problem. For each dataset, we have standard split for *train*, *dev*, and *test*, and standard separation of seen and unseen classes.

**Evaluation.** Our standardized evaluations correspond to the *Definition-Restrictive* and *Definition-Wild*. i) *Label-partially-unseen evaluation*. This corresponds to the commonly studied 0SHOT-TC defined in *Definition-Restrictive*: for the set of labels of a specific aspect, given training data for a part of labels, predicting in the full label set. This is the most basic setup in 0SHOT-TC. It checks whether the system can generalize to some labels in the same aspect. To satisfy *Definition-Wild*, we define a new evaluation: ii) *Label-fully-unseen evaluation*. In this setup, we assume the system is unaware of the upcoming aspects and can not access any labeled data for task-specific training.

**Entailment approach.** Our *Definition-Wild* challenges the system design – how to develop a 0SHOT-TC system, without accessing any task-specific labeled data, to deal with labels from diverse aspects? In this work, we propose to treat 0SHOT-TC as a textual entailment problem. This is to imitate how humans decide the truth value of labels from any aspects. Usually, humans understand the problem described by the aspect and the meaning of the label candidates. Then humans mentally construct a hypothesis by filling a label candidate, e.g., "sports", into the aspect-defined problem "the text is about ?", and ask ourselves if this hypothesis is true, given the text. We treat 0SHOT-TC as a textual entailment problem so that our model can gain knowledge from entailment datasets, and we show that it applies to both *Definition-Restrictive* and *Definition-Wild*.

Overall, this work aims at benchmarking the research of 0SHOT-TC by providing standardized datasets, evaluations, and a state-of-the-art entailment system. All datasets and codes are released.

## 2 Related Work

ZERO-STC was first explored by the paradigm "Dataless Classification" (Chang et al., 2008).

Dataless classification first maps the text and labels into a common space by Explicit Semantic Analysis (ESA) (Gabrilovich and Markovitch, 2007), then picks the label with the highest matching score. Dataless classification emphasizes that the representation of labels takes the equally crucial role as the representation learning of text. Then this idea was further developed in (Song and Roth, 2014; Chen et al., 2015; Li et al., 2016a,b; Song et al., 2016).

With the prevalence of word embeddings, more and more work adopts pretrained word embeddings to represent the meaning of words, so as to provide the models with the knowledge of labels (Sappadla et al., 2016; Yogatama et al., 2017; Rios and Kavuluru, 2018; Xia et al., 2018). Yogatama et al. (2017) build generative LSTM to generate text given the embedded labels. Rios and Kavuluru (2018) use label embedding to attend the text representation in the developing of a multi-label classifier. But they report R@K, so it is unclear whether the system can really predict unseen labels. Xia et al. (2018) study the zero-shot intent detection problem. The learned representations of intents are still the sum of word embeddings. But during testing, the intent space includes only new intents; seen intents are not covered. All of these studies can only meet the definition in *Definition-Restrictive*, so they do not really generalize to open aspects of 0SHOT-TC.

Zhang et al. (2019) enrich the embedding representations by incorporating class descriptions, class hierarchy, and the word-to-label paths in ConceptNet. Srivastava et al. (2018) assume that some natural language explanations about new labels are available. Then those explanations are parsed into formal constraints which are further combined with unlabeled data to yield new label oriented classifiers through posterior regularization. However, those explanatory statements about new labels are collected from crowd-sourcing. This limits its application in real world 0SHOT-TC scenarios.

There are a few works that study a specific zero-shot problem by indirect supervision from other problems. Levy et al. (2017) and Obamuyide and Vlachos (2018) study zero-shot relation extraction by converting it into a machine comprehension and textual entailment problem respectively. Then, a supervised system pretrained on an existing machine comprehension dataset or textual entailment dataset is used to do inference. Our work studies the 0SHOT-TC by formulating a broader vision: datasets of multiple apsects and evaluations.

Other zero-shot problems studied in NLP involve entity typing (Zhou et al., 2018), sequence labeling (Rei and Søgaard, 2018), etc.

## 3 Benchmark the dataset

In this work, we standardize the datasets for 0SHOT-TC for three aspects: topic detection, emotion detection, and situation detection.

For each dataset, we insist on two principles: i) **Label-partially-unseen**: A part of labels are unseen. This corresponds to *Definition-Restrictive*, enabling us to check the performance of unseen labels as well as seen labels. ii) **Label-fully-unseen**: All labels are unseen. This corresponds to *Definition-Wild*, enabling us to check the system performance in test-agnostic setups.

### 3.1 Topic detection

**Yahoo.** We use the large-scale Yahoo dataset released by Zhang et al. (2015). Yahoo has 10 classes: {"Society & Culture", "Science & Mathematics", "Health", "Education & Reference", "Computers & Internet", "Sports", "Business & Finance", "Entertainment & Music", "Family & Relationships", "Politics & Government"}, with original split: 1.4M/60k in train/test (all labels are balanced distributed).

We reorganize the dataset by first fixing the *dev* and *test* sets as follows: for *dev*, all 10 labels are included, with 6k labeled instances for each; For *test*, all 10 labels are included, with 10k instances for each. Then training sets are created on remaining instances as follows.

For *label-partially-unseen*, we create two versions of Yahoo *train* for 0SHOT-TC:

- *Train-v0*: 5 classes: {"Society & Culture", "Health", "Computers & Internet", "Business & Finance", "Family & Relationships"} are included; each is equipped with 130k labeled instances.
- *Train-v1*: 5 classes: { "Science & Mathematics", "Education & Reference", "Sports", "Entertainment & Music", "Politics & Government"} are included; each is equipped with 130k labeled instances.

We always create two versions of *train* with non-overlapping labels so as to get rid of the

| | | emotions | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | sad | joy | anger | disgust | fear | surp. | shame | guilt | love | none | sum |
| domains | tweets | 1,500 | 2,150 | 1,650 | 50 | 2,150 | 880 | | | 1,100 | 1,000 | 10,480 |
| | events | 300 | 200 | 400 | 400 | 200 | | 300 | 300 | | | 2,100 |
| | fairytales | 300 | 500 | 250 | 120 | 250 | 220 | | | | 1,000 | 2,640 |
| | arti. sent. | 200 | 150 | 200 | 30 | 100 | 100 | | | | | 780 |
| sum | | 2,300 | 3,100 | 2,500 | 600 | 2,700 | 1,200 | 300 | 300 | 1,100 | 2,000 | 16,000 |

Table 1: Emotion *test* in 0SHOT-TC

| | | emotions | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | sad | joy | anger | disgust | fear | surp. | shame | guilt | love | none | sum |
| domains | tweets | 900 | 1,050 | 400 | 40 | 1,200 | 370 | | | 400 | 500 | 4,860 |
| | events | 150 | 150 | 150 | 150 | 150 | | 100 | 100 | | | 950 |
| | fairytales | 150 | 300 | 150 | 90 | 150 | 80 | | | | 500 | 1,420 |
| | arti. sent. | 100 | 100 | 100 | 20 | 100 | 50 | | | | | 470 |
| sum | | 1,300 | 1,600 | 800 | 300 | 1,600 | 500 | 100 | 100 | 400 | 1,000 | 7,700 |

Table 2: Emotion *dev* in 0SHOT-TC

model's over-fitting on one of them.

*Label-fully-unseen* share the same *test* and *dev* with the *label-partially-unseen* except that it has no training set. It is worth mentioning that our setup of *label-partially-unseen* and *label-fully-unseen* enables us to compare the performance mutually; it can show the system's capabilities while seeing different sizes of classes.

### 3.2 Emotion detection

**UnifyEmotion.** This emotion dataset was released by Bostan and Klinger (2018). It was constructed by unifying the emotion labels of multiple public emotion datasets[2]. This dataset consists of text from multiple domains: tweet, emotional events, fairy tale and artificial sentences, and it contains 9 emotion types {"sadness", "joy", "anger", "disgust", "fear", "surprise", "shame", "guilt", "love"} and "none" (if no emotion applies). We remove the multi-label instances (appro. 4k) so that the remaining instances always have a single positive label. The official evaluation metric is *label-weighted* F1.

Since the labels in this dataset has unbalanced distribution. We first directly list the fixed *test* and *dev* in Table 1 and Table 2, respectively. They are shared by following *label-partial-unseen* and *label-fully-unseen* setups of *train*.

*Label-partial-unseen* has the following two versions of *train*:

- *Train-v0*: 5 classes: {"sadness", "anger", "fear", "shame", "love"} are included.
- *Train-v1*: 4 classes: { "joy", "disgust", "surprise", "guilt"} are included.

For *label-fully-unseen*, no training set is provided.

### 3.3 Situation detection

The situation frame typing is one example of an event-type classification task. A situation frame studied here is a *need* situation such as the need for water or medical aid, or an *issue* situation such as crime violence (Strassel et al., 2017; Muis et al., 2018). It was originally designed for low-resource situation detection, where annotated data is unavailable. This is why it is particularly suitable for 0SHOT-TC.

We use the Situation Typing dataset released by Mayhew et al. (2019). It has 5,956 labeled instances. Totally 11 situation types: "food supply", "infrastructure", "medical assistance", "search/rescue", "shelter", "utilities, energy, or sanitation", "water supply", "evacuation", "regime change", "terrisms", "crime violence" and an extra type "none" – if none of the 11 types applies. This dataset is a *multi-label* classification, and *label-wise weighted F1* is the official evaluation.

The *train*, *test* and *dev* are listed in Table 3.

[2]Please refer to (Bostan and Klinger, 2018) for more details about the constituent datasets.

**Summary of 0SHOT-TC datasets.** Our three datasets covers single-label classification (i.e., "topic" and "emotion") and multi-label classification (i.e., "situation"). In addition, a "none" type is adopted in "emotion" and "situation" tasks if no predefined types apply – this makes the problem more realistic.

## 4 Benchmark the evaluation

How to evaluate a 0SHOT-TC system? This needs to review the original motivation of doing 0SHOT-TC research. As we discussed in Introduction section, ideally, we aim to build a system that works like humans – figuring out if a piece of text can be assigned with an open-defined label, without any constrains on the domains and the aspects described by the labels. Therefore, we challenge the system in two setups: *label-partially-unseen* and *label-fully-unseen*.

**Label-partially-unseen.** This is the most common setup in existing 0SHOT-TC literature: for a given dataset of a specific problem such as topic categorization, emotion detection, etc, train a system on a part of the labels, then test on the whole label space. Usually all labels describe the same aspect of the text.

**Label-fully-unseen.** In this setup, we push "zero-shot" to the extreme – no annotated data for any labels. So, we imagine that learning a system through whatever approaches, then testing it on 0SHOT-TC datasets of open aspects.

This *label-fully-unseen* setup is more like the dataless learning principle (Chang et al., 2008), in which no task-specific annotated data is provided for training a model (since usually this kind of model fails to generalize in other domains and other tasks), therefore, we are encouraged to learn models with open-data or test-agnostic data. In this way, the learned models behave more like humans.

## 5 An entailment model for 0SHOT-TC

As one contribution of this work, we propose to deal with 0SHOT-TC as a textual entailment problem. It is inspired by: i) text classification is essentially a textual entailment problem. Let us think about how humans do classification: we mentally think "whether this text is about sport?", or "whether this text expresses a specific feeling?", or "whether the people there need water supply?" and so on. The reason that conventional text classification did not employ entailment approach is it always has pre-defined, fixed-size of classes equipped with annotated data. However, in 0SHOT-TC, we can neither estimate how many and what classes will be handled nor have annotated data to train class-specific parameters. Textual entailment, instead, does not preordain the boundary of the hypothesis space. ii) To pursue the ideal generalization of classifiers, we definitely need to make sure that the classifiers understand the problem encoded in the aspects and understand the meaning of labels. Conventional supervised classifiers fail in this aspect since label names are converted into indices – this means the classifiers do not really understand the labels, let alone the problem. Therefore, exploring 0SHOT-TC as a textual entailment paradigm is a reasonable way to achieve generalization.

**Convert labels into hypotheses.** The first step of dealing with 0SHOT-TC as an entailment problem is to convert labels into hypotheses. To this end, we first convert each aspect into an *interpretation* (we discussed before that generally one aspect defines one interpretation). E.g., "topic" aspect to interpretation "the text is about the topic". Table 4 lists some examples for the three aspects: "topic", "emotion" and "situation".

In this work, we just explored two simple methods to generate the hypotheses. As Table 4 shows, one is to use the label name to complete the interpretation, the other is to use the label's definition in WordNet to complete the interpretation. In testing, once one of them results in an "entailment" decision, then we decide the corresponding label is positive. We can definitely create more natural hypotheses through crowd-sourcing, such as "food" into "the people there are starving". Here we just set the baseline examples by automatic approaches, more explorations are left as future work, and we welcome the community to contribute.

**Convert classification data into entailment data.** For a data split (*train*, *dev* and *test*), each input text, acting as the premise, has a positive hypothesis corresponding to the positive label, and all negative labels in the data split provide negative hypotheses. Note that unseen labels do not provide negative hypotheses for instances in *train*.

| | | situations | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | search | evac | infra | utils | water | shelter | med | food | reg. | terr. | crim. | none |
| | total size | 327 | 278 | 445 | 412 | 492 | 659 | 1,046 | 810 | 80 | 348 | 983 | 1,868 |
| split | test | 190 | 166 | 271 | 260 | 289 | 396 | 611 | 472 | 51 | 204 | 590 | 1,144 |
| | dev | 137 | 112 | 174 | 152 | 203 | 263 | 435 | 338 | 29 | 144 | 393 | 724 |
| | train-v0 | 327 | – | 445 | – | 492 | – | 1,046 | – | 80 | – | 983 | – |
| | train-v1 | – | 278 | – | 412 | – | 659 | – | 810 | – | 348 | – | – |

Table 3: Situation *train*, *dev* and *test* split for 0SHOT-TC.

| aspect | labels | interpretation | example hypothesis | |
|---|---|---|---|---|
| | | | word | wordnet definition |
| topic | sports etc. | this text is about ? | "?"= sports | "?" = an active diversion requiring physical exertion and competition |
| emotion | anger etc. | this text expresses ? | "?"= anger | "?" = a strong emotion; a feeling that is oriented toward some real or supposed grievance |
| situation | shelter etc. | The people there need ? | "?"= shelter | "?" = a structure that provides privacy and protection from danger |

Table 4: Example hypotheses we created for modeling different aspects of 0SHOT-TC.

**Entailment model learning.** In this work, we make use of the widely-recognized state of the art entailment technique – BERT (Devlin et al., 2019), and train it on three mainstream entailment datasets: MNLI (Williams et al., 2018), GLUE RTE (Dagan et al., 2005; Wang et al., 2019) and FEVER[3] (Thorne et al., 2018), respectively. We convert all datasets into binary case: "entailment" vs. "non-entailment", by changing the label "neutral" (if exist in some datasets) into "non-entailment".

For our *label-fully-unseen* setup, we directly apply this pretrained entailment model on the test sets of all 0SHOT-TC aspects. For *label-partially-unseen* setup in which we intentionally provide annotated data, we first pretrain BERT on the MNLI/FEVER/RTE, then fine-tune on the provided training data.

**Harsh policy in testing.** Since seen labels have annotated data for training, we adopt different policies to pick up seen and unseen labels. To be specific, we pick a seen label with a harsher rule: i) In single-label classification, if both seen and unseen labels are predicted as positive, we pick the seen label only if its probability of being positive is higher than that of the unseen label by a hyper-parameter $\alpha$. If only seen or unseen labels are predicted as positive, we pick the one with the highest probability; ii) In multi-label classification, if both seen and unseen labels are predicted as positive, we change the seen labels into "negative" if their probability of being positive is higher than that of the unseen label by less than $\alpha$. Finally, all labels labeled positive will be selected. If no positive labels, we choose "none" type.

$\alpha = 0.05$ in our systems, tuned on *dev*.

## 6 Experiments

### 6.1 Label-partially-unseen evaluation

In this setup, there is annotated data for partial labels as *train*. So, we report performance for unseen classes as well as seen classes. We compare our entailment approaches, trained separately on MNLI, FEVER and RTE, with the following baselines.

**Baselines.**

- *Majority*: the text picks the label of the largest size.
- *ESA*: A dataless classifier proposed in (Chang et al., 2008). It maps the words (in text and label names) into the title space of Wikipedia articles, then compares the text with label names. This method does not rely on *train*.

[3]FEVER is an evidential claim verification problem: given a hypothesis, the system needs to identify evidence sentences as premise, then gives the entailment decision. We use the ground truth evidence as premises in this work.

| | | topic | | | | emotion | | | | situation | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | v0 | | v1 | | v0 | | v1 | | v0 | | v1 | |
| | | s | u | s | u | s | u | s | u | s | u | s | u |
| w/o *train* | Majority | 0.0 | 10.0 | 10.0 | 0.0 | 0.0 | 13.3 | 18.5 | 0.0 | 0.0 | 19.7 | 0.0 | 16.4 |
| | Word2Vec | 28.1 | 43.3 | 43.3 | 28.1 | 8.1 | 5.4 | 6.2 | 7.3 | 10.3 | 24.7 | 8.6 | 23.1 |
| | ESA | 27.5 | 30.1 | 30.1 | 27.5 | 6.7 | 9.7 | 5.5 | 9.2 | 22.8 | 28.5 | 22.4 | 27.7 |
| supervised train | Binary-BERT | **72.6** | 44.3 | **80.6** | 34.9 | **35.6** | 17.5 | **37.1** | 14.2 | 72.4 | 48.4 | 63.8 | 42.9 |
| | our entail | | | | | | | | | | | | |
| | MNLI | 70.9 | **52.1** | 77.3 | **45.3** | 33.4 | **26.6** | 33.9 | 21.4 | **74.8** | **53.4** | **68.4** | **47.8** |
| | FEVER | 70.2 | 51.7 | 77.2 | 42.7 | 31.9 | 24.5 | 26.0 | **22.5** | 73.5 | 47.6 | 65.7 | 43.6 |
| | RTE | 71.5 | 45.3 | 78.6 | 40.6 | 32.0 | 21.8 | 32.7 | 21.1 | 72.8 | 52.0 | 65.0 | 45.2 |

Table 5: Label-partially-unseen evaluation. "v0/v1" means the results in that column are obtained by training on train-v0/v1. "s": seen labels; "u": unseen labels. "Topic" uses *acc.*, both "emotion" and "situation" use *label-wised weighted F1*. Note that for baselines "Majority", "Word2Vec" and "ESA", they do not have *seen* labels; we just separate their numbers into *seen* and *unseen* subsets of supervised approaches for clear comparison.

| | topic | emotion | situation | sum |
|---|---|---|---|---|
| Majority | 10.0 | 5.9 | 11.0 | 26.9 |
| Word2Vec | 35.7 | 6.9 | 15.6 | 58.2 |
| ESA | 28.6 | 8.0 | 26.0 | 62.6 |
| Wiki-based | **52.1** | 21.2 | 27.7 | 101.0 |
| our entail. | | | | |
| MNLI | 37.9 | 22.3 | 15.4 | 75.6 |
| FEVER | 40.1 | 24.7 | 21.0 | 85.8 |
| RTE | 43.8 | 12.6 | 37.2 | 93.6 |
| ensemble | 45.7 | **25.2** | **38.0** | **108.9** |

Table 6: Label-fully-unseen evaluation.

We implemented ESA based on 08/01/2019 Wikipedia dump[4]. There are about 6.1M words and 5.9M articles.

- *Word2Vec*[5] (Mikolov et al., 2013): Both the representations of the text and the labels are the addition of word embeddings element-wisely. Then cosine similarity determines the labels. This method does not rely on *train* either.

- *Binary-BERT*: We fine-tune BERT[6] on *train*, which will yield a binary classifier for entailment or not; then we test it on *test* – picking the label with the maximal probability in single-label scenarios while choosing all the labels with "entailment" decision in multi-label cases.

[4] https://dumps.wikimedia.org/enwiki/

[5] https://code.google.com/archive/p/word2vec/

[6] We always use "bert-base-uncased" version.

**Discussion.** The results of *label-partially-unseen* are listed in Table 5. "ESA" performs slightly worse than "Word2Vec" in topic detection, mainly because the label names, i.e., topics such as "sports", are closer than some keywords such as "basketball" in Word2Vec space. However, "ESA" is clearly better than "Word2Vec" in situation detection; this should be mainly due to the fact that the label names (e.g., "shelter", "evaculation", etc.) can hardly find close words in the text by Word2Vec embeddings. Quite the contrary, "ESA" is easier to make a class such as "shelter" closer to some keywords like "earthquake". Unfortunately, both Word2Vec and ESA work poorly for emotion detection problem. We suspect that emotion detection requires more entailment capability. For example, the text snippet "when my brother was very late in arriving home from work", its gold emotion "fear" requires some common-knowledge inference, rather than just word semantic matching through Word2Vec and ESA.

The supervised method "Binary-BERT" is indeed strong in learning the seen-label-specific models – this is why it predicts very well for seen classes while performing much worse for unseen classes.

Our entailment models, especially the one pretrained on MNLI, generally get competitive performance with the "Binary-BERT" for *seen* (slightly worse on "topic" and "emotion" while clearly better on "situation") and improve the performance regarding *unseen* by large margins. At this stage, fine-tuning on an MNLI-based pre-

| | topic | | | | emotion | | | | situation | | | | sum | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RTE | FEV. | MN. | ens. | RTE | FEV. | MN. | ens. | RTE | FEV. | MN. | ens. | RTE | FEV. | MN. | ens. |
| word | 44.9 | 42.0 | 43.4 | 48.4 | 12.4 | 26.7 | 21.2 | 18.3 | 37.7 | 24.5 | 14.7 | 38.3 | 95.0 | 93.2 | 79.3 | 105.0 |
| def | 14.5 | 25.3 | 17.2 | 26.0 | 3.4 | 18.7 | 16.8 | 9.0 | 14.1 | 19.2 | 11.8 | 14.4 | 32.0 | 63.2 | 45.8 | 49.4 |
| comb. | 43.8 | 40.1 | 37.9 | 45.7 | 12.6 | 24.7 | 22.3 | 25.2 | 37.2 | 21.0 | 15.4 | 38.0 | 93.6 | 85.8 | 81.2 | **108.9** |

Table 7: Fine-grained *label-fully-unseen* performances of different hypothesis generation approaches "word", "def" (definition) and "comb" (word&definition) on the three tasks ("topic", "emotion" and "situation") based on three pretrained entailment models (RTE, FEVER, MNLI) and the ensemble approach (ens.). The last column *sum* contains the addition of its preceding three blocks element-wisely.

trained entailment model seems more powerful.

### 6.2 Label-fully-unseen evaluation

Regarding this *label-fully-unseen* evaluation, apart from our entailment models and three unsupervised baselines "Majority", "Word2Vec" and "ESA", we also report the following baseline:

**Wikipedia-based**: We train a *binary* classifier based on BERT on a dataset collected from Wikipedia. Wikipedia is a corpus of general purpose, without targeting any specific 0SHOT-TC task. Collecting categorized articles from Wikipedia is popular way of creating training data for text categorization, such as (Zhou et al., 2018). More specifically, we collected 100K articles along with their categories in the bottom of each article. For each article, apart from its attached positive categories, we randomly sample three negative categories. Then each article and its positive/negative categories act as training pairs for the binary classifier.

We notice "Wikipedia-based" training indeed contributes a lot for the topic detection task; however, its performances on emotion and situation detection problems are far from satisfactory. We believe this is mainly because the Yahoo-based topic categorization task is much closer to the Wikipedia-based topic categorization task; emotion and situation categorizations, however, are relatively further.

Our entailment models, pretrained on MNLI/FEVER/RTE respectively, perform more robust on the three 0SHOT-TC aspects (except for the RTE on emotion). Recall that they are not trained on any text classification data, and never know the domain and the aspects in the *test*. This clearly shows the great promise of developing textual entailment models for 0SHOT-TC. Our ensemble approach[7] further boosts the performances on all three tasks.

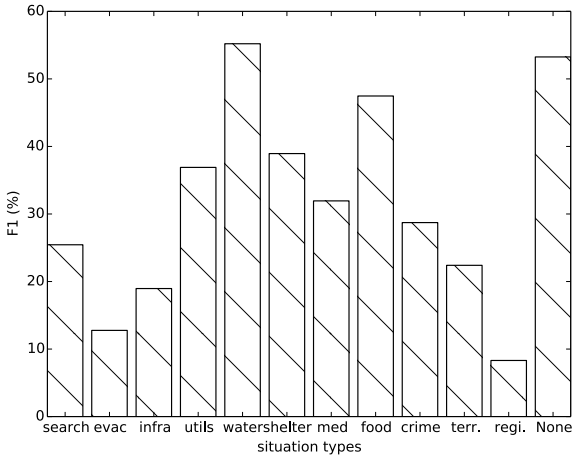


Figure 2: Performance of different situation classes in *label-fully-unseen, predicted by the ensemble model.*

An interesting phenomenon, comparing the *label-partially-unseen* results in Table 5 and the *label-fully-unseen* results in Table 6, is that the pretrained entailment models work in this order for *label-fully-unseen* case: RTE > FEVER >MNLI; on the contrary, if we fine-tune them on the label-partially-unseen case, the MNLI-based model performs best. This could be due to a possibility that, on one hand, the constructed situation entailment dataset is closer to the RTE dataset than to the MNLI dataset, so an RTE-based model can generalize well to situation data, but, on the other hand, it could also be more likely to over-fit the training set of "situation" during fine-tuning. A deeper exploration of this is left as future work.

### 6.3 How do the generated hypotheses influence

In Table 4, we listed examples for converting class names into hypotheses. In this work, we only tried to make use of the class names and their defini-

[7]For each input pair of the entailment model, we sum up their probabilities after softmax, then do softmax to get new probabilities.

tions in WordNet. Table 7 lists the fine-grained performance of three ways of generating hypotheses: "word", "definition", and "combination" (i.e., word&definition).

This table indicates that: i) Definition alone usually does not work well in any of the three tasks, no matter which pretrained entailment model is used; ii) Whether "word" alone or "word&definition" works better depends on the specific task and the pretrained entailment model. For example, the pretrained MNLI model prefers "word&definition" in both "emotion" and "situation" detection tasks. However, the other two entailment models (RTE and FEVER) mostly prefer "word". iii) Since it is unrealistic to adopt only one entailment model, such as from {RTE, FEVER, MNLI}, for any open 0SHOT-TC problem, an ensemble system should be preferred. However, the concrete implementation of the ensemble system also influences the strengths of different hypothesis generation approaches. In this work, our ensemble method reaches the top performance when combining the "word" and "definition". More ensemble systems and hypothesis generation paradigms need to be studied in the future.

To better understand the impact of generated hypotheses, we dive into the performance of each labels, taking "situation detection" as an example. Figure 2 illustrates the separate F1 scores for each situation class, predicted by the ensemble model for *label-fully-unseen* setup. This enables us to check in detail how easily the constructed hypotheses can be understood by the entailment model. Unfortunately, some classes are still challenging, such as "evacuation", "infrastructure", and "regime change". This should be attributed to their over-abstract meaning. Some classes were well recognized, such as "water", "shelter", and "food". One reason is that these labels mostly are common words – systems can more easily match them to the text; the other reason is that they are situation classes with higher frequencies (refer to Table 3) – this is reasonable based on our common knowledge about disasters.

## 7 Summary

In this work, we analyzed the problems of existing research on zero-shot text classification (0SHOT-TC): restrictive problem definition, the weakness in understanding the problem and the labels' meaning, and the chaos of datasets and evaluation setups. Therefore, we are benchmarking 0SHOT-TC by standardizing the datasets and evaluations. More importantly, to tackle the broader-defined 0SHOT-TC, we proposed a textual entailment framework which can work with or without the annotated data of *seen* labels.

## Acknowledgments

The authors would like to thank Jennifer Sheffield and the anonymous reviewers for insightful comments and suggestions. This work was supported by Contracts HR0011-15-C-0113 and HR0011-18-2-0052 with the US Defense Advanced Research Projects Agency (DARPA). Approved for Public Release, Distribution Unlimited. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

## References

Laura Ana Maria Bostan and Roman Klinger. 2018. An analysis of annotated corpora for emotion classification in text. In *COLING*, pages 2104–2119.

Ming-Wei Chang, Lev-Arie Ratinov, Dan Roth, and Vivek Srikumar. 2008. Importance of semantic representation: Dataless classification. In *AAAI*, pages 830–835.

Xingyuan Chen, Yunqing Xia, Peng Jin, and John A. Carroll. 2015. Dataless text classification with descriptive LDA. In *AAAI*, pages 2224–2231.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The PASCAL recognising textual entailment challenge. In *Machine Learning Challenges, Evaluating Predictive Uncertainty, Visual Object Classification and Recognizing Textual Entailment, First PASCAL Machine Learning Challenges Workshop*, pages 177–190.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJCAI*, pages 1606–1611.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. In *CoNLL*, pages 333–342.

Chenliang Li, Jian Xing, Aixin Sun, and Zongyang Ma. 2016a. Effective document labeling with very few

seed words: A topic model approach. In *CIKM*, pages 85–94.

Yuezhang Li, Ronghuo Zheng, Tian Tian, Zhiting Hu, Rahul Iyer, and Katia P. Sycara. 2016b. Joint embedding of hierarchical categories and entities for concept categorization and dataless classification. In *COLING*, pages 2678–2688.

Stephen Mayhew, Tatiana Tsygankova, Francesca Marini, Zihan Wang, Jane Lee, Xiaodong Yu, Xingyu Fu, Weijia Shi, Zian Zhao, Wenpeng Yin, Karthikeyan K, Jamaal Hay, Michael Shur, Jennifer Sheffield, and Dan Roth. 2019. University of pennsylvania lorehlt 2019 submission.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Aldrian Obaja Muis, Naoki Otani, Nidhi Vyas, Ruochen Xu, Yiming Yang, Teruko Mitamura, and Eduard Hovy. 2018. Low-resource cross-lingual event type detection via distant supervision with minimal effort. In *COLING*, pages 70–82.

Abiola Obamuyide and Andreas Vlachos. 2018. Zero-shot relation classification as textual entailment. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 72–78.

Pushpankar Kumar Pushp and Muktabh Mayank Srivastava. 2017. Train once, test anywhere: Zero-shot learning for text classification. *CoRR*, abs/1712.05972.

Marek Rei and Anders Søgaard. 2018. Zero-shot sequence labeling: Transferring knowledge from sentences to tokens. In *NAACL-HLT*, pages 293–302.

Anthony Rios and Ramakanth Kavuluru. 2018. Few-shot and zero-shot multi-label learning for structured label spaces. In *EMNLP*, pages 3132–3142.

Prateek Veeranna Sappadla, Jinseok Nam, Eneldo Loza Menc'ia, and Johannes Fürnkranz. 2016. Using semantic similarity for multi-label zero-shot classification of text documents. In *24th European Symposium on Artificial Neural Networks*, pages 423–428.

Yangqiu Song and Dan Roth. 2014. On dataless hierarchical text classification. In *AAAI*, pages 1579–1585.

Yangqiu Song, Shyam Upadhyay, Haoruo Peng, and Dan Roth. 2016. Cross-lingual dataless classification for many languages. In *IJCAI*, pages 2901–2907.

Shashank Srivastava, Igor Labutov, and Tom M. Mitchell. 2018. Zero-shot learning of classifiers from natural language quantification. In *ACL*, pages 306–316.

Stephanie M. Strassel, Ann Bies, and Jennifer Tracey. 2017. Situational awareness for low resource languages: the LORELEI situation frame annotation task. In *SMERP@ECIR*, pages 32–41.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and verification. In *NAACL-HLT*, pages 809–819.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR*.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL-HLT*, pages 1112–1122.

Congying Xia, Chenwei Zhang, Xiaohui Yan, Yi Chang, and Philip S. Yu. 2018. Zero-shot user intent detection via capsule neural networks. In *EMNLP*, pages 3090–3099.

Dani Yogatama, Chris Dyer, Wang Ling, and Phil Blunsom. 2017. Generative and discriminative text classification with recurrent neural networks. *CoRR*, abs/1703.01898.

Jingqing Zhang, Piyawat Lertvittayakumjorn, and Yike Guo. 2019. Integrating semantic knowledge to tackle zero-shot text classification. In *NAACL*.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NIPS*, pages 649–657.

Ben Zhou, Daniel Khashabi, Chen-Tse Tsai, and Dan Roth. 2018. Zero-shot open entity typing as type-compatible grounding. In *EMNLP*, pages 2065–2076.

# A Logic-Driven Framework for Consistency of Neural Models

**Tao Li, Vivek Gupta, Maitrey Mehta, Vivek Srikumar**
School of Computing, University of Utah
{tli,vgupta,maitrey,svivek}@cs.utah.edu

## Abstract

While neural models show remarkable accuracy on individual predictions, their internal beliefs can be inconsistent *across* examples. In this paper, we formalize such inconsistency as a generalization of prediction error. We propose a learning framework for constraining models using logic rules to regularize them away from inconsistency. Our framework can leverage both labeled and unlabeled examples and is directly compatible with off-the-shelf learning schemes without model redesign. We instantiate our framework on natural language inference, where experiments show that enforcing invariants stated in logic can help make the predictions of neural models both accurate and consistent.

## 1 Introduction

Recent NLP advances have been powered by improved representations (e.g., ELMo, BERT— Peters et al., 2018; Devlin et al., 2019), novel neural architectures (e.g., Cheng et al., 2016; Seo et al., 2017; Parikh et al., 2016; Vaswani et al., 2017), and large labeled corpora (e.g., Bowman et al., 2015; Rajpurkar et al., 2016; Williams et al., 2018). Consequently, we have seen progressively improving performances on benchmarks such as GLUE (Wang et al., 2018). But, are models really becoming better? We take the position that, while tracking performance on a leaderboard is necessary to characterize model quality, it is not sufficient. Reasoning about language requires that a system has the ability not only to draw correct inferences about textual inputs, but also to be consistent its beliefs across various inputs.

To illustrate this notion of consistency, let us consider the task of natural language inference (NLI) which seeks to identify whether a premise entails, contradicts or is unrelated to a hypothesis (Dagan et al., 2013). Suppose we have three sentences $P$, $H$ and $Z$, where $P$ entails $H$ and $H$ contradicts $Z$. Using these two facts, we can infer that $P$ contradicts $Z$. In other words, these three decisions are not independent of each other. Any model for textual inference should not violate this invariant defined over any three sentences, *even if they are not labeled*.

Neither are today's models trained to be consistent in this fashion, nor is consistency evaluated. The decomposable attention model of Parikh et al. (2016) updated with ELMo violates the above constraint for the following sentences:[1]

$P$: John is on a train to Berlin.
$H$: John is traveling to Berlin.
$Z$: John is having lunch in Berlin.

Highly accurate models can be inconsistent in their beliefs over groups of examples. For example, using a BERT-based NLI model that achieves about 90% F-score on the SNLI test set (Bowman et al., 2015), we found that in about 46% of *unlabeled* sentence triples where $P$ entails $H$ and $H$ contradicts $Z$, the first sentence does not contradict the third. Observations of a similar spirit were also made by Minervini and Riedel (2018), Glockner et al. (2018) and Nie et al. (2018).

To characterize and eliminate such errors, first, we define a method to measure the inconsistency of models with respect to invariants stated as first-order logic formulas over model predictions. We show that our definition of inconsistency strictly generalizes the standard definition of model error.

Second, we develop a systematic framework for mitigating inconsistency in models by compiling the invariants into a differentiable loss functions using t-norms (Klement et al., 2013; Gupta and Qi, 1991) to soften logic. This allows us

[1] We used the model available through the Allen NLP online demo: http://demo.allennlp.org/textual-entailment.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3924–3935, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

to take advantage of unlabeled examples and enforce consistency of model predictions over them. We show that the commonly used cross-entropy loss emerges as a specific instance of our framework. Our framework can be easily instantiated with modern neural network architectures.

To show the effectiveness of our approach, we instantiate it on the NLI task. We show that even state-of-the-art models can be highly inconsistent in their predictions, but our approach significantly reduces inconsistency.

In summary, our contributions are:

1. We define a mechanism to measure model inconsistency with respect to declaratively specified invariants.
2. We present a framework that compiles knowledge stated in first-order logic to loss functions that mitigate inconsistency.
3. We show that our learning framework can reduce prediction inconsistencies even with small amount of annotated examples without sacrificing predictive accuracy.[2]

## 2 A Framework for (In)consistency

In this section, we will present a systematic approach for measuring and mitigating inconsistent predictions. A prediction is *incorrect* if it disagrees with what is known to be true. Similarly, predictions are *inconsistent* if they do not follow a known rule. Therefore, a model's errors can be defined by their concordance with declarative knowledge. We will formalize this intuition by first developing a uniform representation for both labeled examples and consistency constraints (§2.1). Then, we will present a general definition of errors in the context of such a representation (§2.2). Finally, we will show a logic-driven approach for designing training losses (§ 2.3).

As a running example, we will use the NLI task whose goal is to predict one of three labels: *Entailment* ($E$), *Contradiction* ($C$), or *Neutral* ($N$).

### 2.1 Representing Knowledge

Suppose $x$ is a *collection* of examples (perhaps labeled). We write constraints about them as a conjunction of statements in logic:

$$\bigwedge_{(L,R)} L(x) \rightarrow R(x) \tag{1}$$

Here, $L$ and $R$ are Boolean formulas, *i.e.* antecedents and consequents, constructed from model predictions on examples in $x$.

One example of such an invariant is the constraint from §1, which can be written as $E(P,H) \wedge C(H,Z) \rightarrow C(P,Z)$, where, e.g., predicate $E(P,H)$ denotes that model predicted label $E$. We can also represent labeled examples as constraints: *"If an example is annotated with label $Y^\star$, then model should predict so."* In logic, we write $\top \rightarrow Y^\star(x)$.[3] Seen this way, the expression (1) could represent labeled data, unlabeled groups of examples with constraints between them, or a combination.

### 2.2 Generalizing Errors as Inconsistencies

Using the representation defined above, we can define how to evaluate predictors. We seek two properties of an evaluation metric: It should 1) quantify the inconsistency of predictions, and 2) also generalize classification error. To this end, we define two types of errors: *global* and *conditional* violation. Both are defined for a dataset $D$ consisting of example collections $x$ as described above.

**Global Violation ($\rho$)** The global violation is the fraction of examples in a dataset $D$ where any constraint is violated. We have:

$$\rho = \frac{\sum_{x \in D} \left[ \bigvee_{(L,R)} \neg \left( L(x) \rightarrow R(x) \right) \right]}{|D|} \tag{2}$$

Here, $[\cdot]$ is the indicator function.

**Conditional Violation ($\tau$)** For a conditional statement, if the antecedent is not satisfied, the statement becomes trivially true. Thus, with complex antecedents, the number of examples where the constraint is true can be trivially large. To only consider those examples where the antecedent holds, we define the conditional violation as:

$$\tau = \frac{\sum_{x \in D} \left[ \bigvee_{(L,R)} \neg \left( L(x) \rightarrow R(x) \right) \right]}{\sum_{x \in D} \left[ \bigvee_{(L,R)} L(x) \right]} \tag{3}$$

**Discussion** The two metrics are complementary to each other. On one hand, to lower the global metric $\rho$, a model could avoid satisfying the antecedents. In this case, the conditional metric $\tau$ is

[2] Our code to replay our experiments is archived at https://github.com/utahnlp/consistency.

[3] The symbol $\top$ denotes the Boolean `true`.

more informative. On the other hand, the global metric reflects the impact of domain knowledge in a given dataset, while the conditional one does not. Ideally, both should be low.

Both violations strictly generalize classification error. If all the knowledge we have takes the form of labeled examples, as exemplified at the end of §2.1, both violation metrics are identical to model error. The appendix formally shows this.

### 2.3 Learning by Minimizing Inconsistencies

With the notion of errors, we can now focus on how to train models to minimize them. A key technical challenge involves the unification of discrete declarative constraints with the standard loss-driven learning paradigm.

To address this, we will use relaxations of logic in the form of t-norms to deterministically compile rules into differentiable loss functions.[4] We treat predicted label probabilities as soft surrogates for Boolean decisions. In the rest of the paper, we will use lower case for model probabilities—e.g., $e(P, H)$, and upper case—e.g., $E(P, H)$—for Boolean predicates.

Different t-norms map the standard Boolean operations into different continuous functions. Table 1 summarizes this mapping for three t-norms: product, Gödel, and Łukasiewicz. Complex Boolean expressions can be constructed from these four operations. Thus, with t-norms to relax logic, we can systematically convert rules as in (1) into differentiable functions, which in turn serve as learning objectives to minimize constraint violations. We can use any off-the-shelf optimizer (e.g., ADAM Kingma and Ba, 2015). We will see concrete examples in the NLI case study in §3.

Picking a t-norm is both a design choice and an algorithmic one. Different t-norms have different numerical characteristics and their comparison is a question for future research.[5] Here, we will focus on the product t-norm to allow comparisons to previous work: as we will see in the next section, the product t-norm strictly generalizes the widely used cross entropy loss.

[4] A full description of t-norms is beyond the scope of this paper; we refer the interested reader to Klement et al. (2013).

[5] For example, the Gödel t-norm, used by Minervini and Riedel (2018), has a discountinuous but semi-differentiable residuum. The Łukasiewicz t-norm can lead to zero gradients for large disjunctions, rendering learning difficult.

## 3 Case Study: NLI

We study our framework using the NLI task as a case study. First, in §3.1, we will show how to represent a training set as in (1). We will also introduce two classes of domain constraints that apply to groups of premise-hypothesis pairs. Next, we will show how to compile these declaratively stated learning objectives to loss functions (§3.2). Finally, we will end this case study with a discussion about practical issues (§3.3).

### 3.1 Learning Objectives in Logic

Our goal is to build models that minimize inconsistency with domain knowledge stated in logic. Let us look at three such consistency requirements.

**Annotation Consistency** For labeled examples, we expect that a model should predict what an annotator specifies. That is, we require

$$\forall(P, H), Y^\star \in D, \quad \top \rightarrow Y^\star(P, H) \qquad (4)$$

where $Y^\star$ represents the ground truth label for the example $(P, H)$. As mentioned at the end of §2.2, for the annotation consistency, both global and conditional violation rates are the same, and minimizing them is maximizing accuracy. In our experiments, we will report accuracy instead of violation rate for annotation consistency (to align with the literature).

**Symmetry Consistency** Given any premise-hypothesis pair, the grounds for a model to predict *Contradiction* is that the events in the premise and the hypothesis cannot coexist simultaneously. That is, a $(P, H)$ pair is a contradiction if, and only if, the $(H, P)$ pair is also a contradiction:

$$\forall(P, H) \in D, \quad C(P, H) \leftrightarrow C(H, P) \qquad (5)$$

**Transitivity Consistency** This constraint is applicable to any three related sentences $P$, $H$ and $Z$. If we group the sentences into three pairs, namely $(P, H)$, $(H, Z)$ and $(P, Z)$, the label definitions mandate that not all of the $3^3 = 27$ assignments to these three pairs are allowed. The example in §1 is an allowed label assignment. We can enumerate all such valid labels as the conjunction:

$$\begin{aligned}
&\forall(P, H, Z) \in D, \\
&\quad (E\,(P, H) \wedge E\,(H, Z) \rightarrow E\,(P, Z)) \\
&\wedge\,(E\,(P, H) \wedge C\,(H, Z) \rightarrow C\,(P, Z)) \\
&\wedge\,(N\,(P, H) \wedge E\,(H, Z) \rightarrow \neg C\,(P, Z)) \\
&\wedge\,(N\,(P, H) \wedge C\,(H, Z) \rightarrow \neg E\,(P, Z))
\end{aligned} \qquad (6)$$

| Name | Boolean Logic | Product | Gödel | Łukasiewicz |
|---|---|---|---|---|
| Negation | $\neg A$ | $1-a$ | $1-a$ | $1-a$ |
| T-norm | $A \wedge B$ | $ab$ | $\min(a,b)$ | $\max(0, a+b-1)$ |
| T-conorm | $A \vee B$ | $a+b-ab$ | $\max(a,b)$ | $\min(1, a+b)$ |
| Residuum | $A \rightarrow B$ | $\min\left(1, \frac{b}{a}\right)$ | $\begin{cases} 1, \text{ if b} \geq \text{a}, \\ b, \text{ else} \end{cases}$ | $\min(1, 1-a+b)$ |

Table 1: Mapping discrete statements to differentiable functions using t-norms. Literals are upper-cased (e.g. $A$) while real-valued probabilities are lower-cased (e.g. $a$). In this paper, we focus on the product t-norm.

### 3.2 Inconsistency Losses

Using the consistency constraints stated in §3.1, we can now derive the inconsistency losses to minimize. For brevity, we will focus on the annotation and symmetry consistencies.

First, let us examine annotation consistency. We can write the universal quantifier in (4) as a conjunction to get:

$$\bigwedge_{(P,H),Y^\star \in D} \top \rightarrow Y^\star(P,H) \tag{7}$$

Using the product t-norm from Table 1, we get the learning objective of maximizing the probability of the true labels:

$$\prod_{(P,H),Y^\star \in D} y^\star_{(P,H)} \tag{8}$$

Or equivalently, by transforming to the negative log space, we get the annotation loss:

$$L_{ann} = \sum_{(P,H),Y^\star \in D} -\log y^\star_{(P,H)}. \tag{9}$$

We see that we get the familiar cross-entropy loss function using the definition of inconsistency with the product t-norm!

Next, let us look at symmetry consistency:

$$\bigwedge_{(P,H) \in D} C(P,H) \leftrightarrow C(H,P). \tag{10}$$

Using the product t-norm, we get:

$$\prod_{(P,H) \in D} \min\left(1, \frac{c_{(H,P)}}{c_{(P,H)}}\right) \min\left(1, \frac{c_{(P,H)}}{c_{(H,P)}}\right) \tag{11}$$

Transforming to the negative log space as before, we get the symmetry loss:

$$L_{sym} = \sum_{(P,H) \in D} |\log c_{(P,H)} - \log c_{(H,P)}| \tag{12}$$

The loss for transitivity $L_{tran}$ can also be similarly derived. We refer the reader to the appendix for details.

The important point is that we can systematically convert logical statements to loss functions and cross-entropy is only one of such losses. To enforce some or all of these constraints, we add their corresponding losses. In our case study, with all constraints, the goal of learning is to minimize:

$$L = L_{ann} + \lambda_{sym} L_{sym} + \lambda_{tran} L_{tran} \tag{13}$$

Here, the $\lambda$'s are hyperparameters to control the influence of each loss term.

### 3.3 Training Constrained Models

The derived loss functions are directly compatible with off-the-shelf optimizers. The symmetry/transitivity consistencies admit using unlabeled examples, while annotation consistency requires labeled examples. Thus, in §4, we will use both labeled and unlabeled data to power training.

Ideally, we want the unlabeled dataset to be absolutely informative, meaning a model learns from *every* example. Unfortunately, obtaining such a dataset remains an open question since new examples are required to be both linguistically meaningful and difficult enough for the model. Minervini and Riedel (2018) used a language model to generate unlabeled adversarial examples. Another way is via pivoting through a different language, which has a long history in machine translation (e.g., Kay, 1997; Mallinson et al., 2017).

Since our focus is to study inconsistency, as an alternative, we propose a simple method to create unlabeled examples: we randomly sample sentences from the same topic. In §4, we will show that even random sentences can be surprisingly informative because the derived losses operate in real-valued space instead on discrete decisions.

| Config | 5% | | | | 100% | | | |
|---|---|---|---|---|---|---|---|---|
| | $\rho_S$ | $\tau_S$ | $\rho_T$ | $\tau_T$ | $\rho_S$ | $\tau_S$ | $\rho_T$ | $\tau_T$ |
| BERT w/ SNLI | 26.3 | 64.4 | 4.9 | 14.8 | 18.6 | 60.3 | 4.7 | 14.9 |
| BERT w/ MultiNLI | 28.4 | 69.3 | 7.0 | 18.5 | 20.6 | 58.9 | 5.6 | 17.5 |
| BERT w/ SNLI+MultiNLI | 25.3 | 62.4 | 4.8 | 14.8 | 18.1 | 59.6 | 4.5 | 14.8 |
| BERT w/ SNLI+MultiNLI$^2$ | 22.1 | 67.1 | 4.1 | 13.7 | 19.3 | 59.7 | 4.5 | 15.2 |
| LSTM w/ SNLI+MultiNLI | 25.8 | 69.5 | 9.9 | 21.0 | 16.8 | 53.6 | 5.3 | 16.0 |

Table 2: Inconsistencies (%) of models on our 100k evaluation dataset. Each number represents the average of three random runs. Models are trained using 5% and 100% of the train sets. SNLI+MultiNLI$^2$: finetuned twice. $\rho_S$ and $\tau_S$: symmetry consistency violations. $\rho_T$ and $\tau_T$: transitivity consistency violations.

## 4 Experiments

In this section, we evaluate our framework using (near) state-of-the-art approaches for NLI, primarily based on BERT, and also compare to an LSTM model. We use the SNLI and MultiNLI (Wang et al., 2018) datasets to define annotation consistency. Our LSTM model is based on the decomposable attention model with a BiLSTM encoder and GloVe embeddings (Pennington et al., 2014). Our BERT model is based on the pretrained BERT$_{base}$, finetuned on SNLI/MultiNLI. The constrained models are initialized with the finetuned BERT$_{base}$ and finetuned again with inconsistency losses.[6] For fair comparison, we also show results of BERT$_{base}$ models finetuned twice.

Our constrained models are trained on both labeled and unlabeled examples. We expect that the different inconsistencies do not conflict with each other. Hence, we select hyperparameters (e.g., the $\lambda$'s) using development accuracy only (*i.e.*, annotation consistency). We refer the reader to the appendix for details of our experimental setup.

### 4.1 Datasets

To be comprehensive, we will use both of the SNLI and MultiNLI to train our models, but we also show individual results.

We study the impact of the amount of label supervision by randomly sampling different percentages of labeled examples. For each case, we also sample the same percentages from the corresponding development sets for model selection. For the MultiNLI dataset, we use the `matched` dev for validation and `mismatched` dev for evaluation.

**Mirrored Instances (M)** Given a labeled example, we construct its mirrored version by swapping the premise and the hypothesis. This results in the same number of unlabeled sentence pairs as the annotated dataset. When sampling by percentage, we will only use the sampled examples to construct mirrored examples. We use this dataset for symmetry consistency.

**Unlabeled Instance Triples (T)** For the transitivity constraint, we sample 100k sentence triples from MS COCO (Lin et al., 2014) captions. From these, we construct three examples as in §3.1: sentences $(P, H, Z)$ gives the pairs $(P, H)$, $(H, Z)$, and $(P, Z)$. In all, we have 100k example *unlabeled* triples for the transitivity constraint.

**Unlabeled Instance Pairs (U)** For each sentence triple in the dataset T, we take the first example $(P, H)$ and construct mirrored examples, *i.e.* $(H, P)$. This yields 100k *unlabeled* instance pairs for training with the symmetry loss.

**Evaluation Dataset** We sample a different set of 100k example triples for measuring transitivity consistency. For symmetry consistency, we follow the above procedure for the dataset U to construct evaluation instance pairs. Recall that the definition of inconsistency allows measuring model quality with unlabeled data.

### 4.2 Inconsistency of Neural Models

In Table 2, we report the impact of the amount of annotated data on symmetry/transitivity consistencies by using different percentages of labeled examples. We see that both LSTM and BERT models have symmetry consistency violations, while the transitivity consistency has lower violations. Surprisingly, the LSTM model performed on par with BERT in terms of symmetry/transitivity consistency; stronger representations does not necessarily mean more consistent models.

The table shows that, given an example and its mirrored version, if the BERT baseline predicts a

[6] This is critical when label supervision is limited.

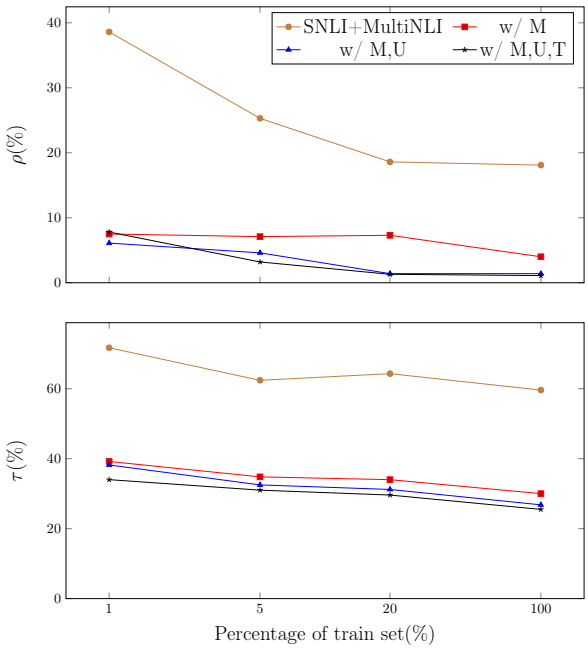


Figure 1: Symmetry inconsistencies on the 100k evaluation example pairs. Each point represents the average of three random runs. M, U, and T are our unlabeled datasets with corresponding losses.

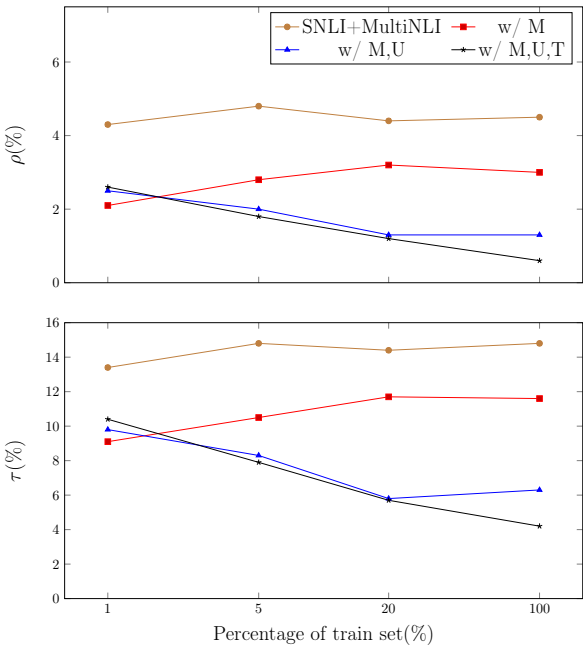


Figure 2: Transitivity inconsistencies on the 100k evaluation example pairs. Each point represents the average of three random runs. M, U, and T are our unlabeled datasets with corresponding losses.

*Contradiction* on one, it has about 60% chance ($\tau_S$) to make an inconsistent judgement on the other. Further, we see that the inconsistencies are not affected much by different datasets. Models trained on the SNLI are as inconsistent as ones trained on MultiNLI. Combining them only gives slight improvements. Also, finetuning twice does not improve much over models finetuned once.

Finally, with more annotation, a model has fewer symmetry consistency violations. However, the same observation does not apply to the transitivity consistency. In the following sections, we will show that we can almost annihilate these inconsistencies using the losses from §3.2.

### 4.3 Reducing Inconsistencies

We will study the effect of symmetry and transitivity consistency losses in turn using the BERT models. To the baseline models, we incrementally include the M, U, and T datasets. We expect that the constrained models should have accuracies at least on par with the baseline (though one of the key points of this paper is that accuracy by itself is not a comprehensive metric).

In Fig. 1, we present both of the global and conditional violation rates of baselines and the constrained models. We see that mirrored examples (*i.e.*, the w/ M curve) greatly reduced the symmetry inconsistency. Further, with 100k unlabeled example pairs (the w/ M,U curve), we can further reduce the error rate. The same observation also applies when combining symmetry with transitivity constraint.

Fig. 2 shows the results for transitivity inconsistency. The transitivity loss is, again, greatly reduced both for the global and conditional violations. We refer the reader to the appendix for exact numbers.

We see that with our augmented losses, even a model using 1% label supervision can be much more consistent than the baselines trained on 100% training set! This suggests that label supervision does not explicitly encode the notion of consistency, and consequently models do not get this information from the training data.

With the simultaneous decline in global and conditional violation rate, the constrained models learn to agree with the consistency requirements specified declaratively. We will see in the next section, doing so does not sacrifice model accuracies.

### 4.4 Interaction of Losses

In Table 3, we show the impact of symmetry and transitivity consistency on test accuracy. And the interaction between symmetry and transitivity consistency is covered in Fig 1 and 2.

| | 1% | | 5% | | 20% | | 100% | |
|---|---|---|---|---|---|---|---|---|
| Config | SNLI | MultiNLI | SNLI | MultiNLI | SNLI | MultiNLI | SNLI | MultiNLI |
| SNLI+MultiNLI | 79.7 | 70.1 | 84.6 | 77.2 | 87.8 | 80.6 | 90.1 | 83.5 |
| SNLI+MultiNLI$^2$ | 80.3 | 71.0 | 85.3 | 77.4 | 87.9 | 80.7 | 90.3 | 84.0 |
| w/ M | 80.1 | 71.0 | 85.3 | 77.8 | 88.1 | 80.6 | 90.3 | 84.1 |
| w/ M,U | 80.2 | 71.0 | 85.4 | 77.2 | 88.1 | 80.9 | 90.5 | 84.3 |
| w/ M,U,T | 80.6 | 71.1 | 85.4 | 77.2 | 88.1 | 80.9 | 90.2 | 84.2 |

Table 3: Impact of symmetry/transitivity consistencies on test set accuracies. Each number represents the average of three random runs of BERT$_{base}$. Columns are accuracies on our SNLI/MultiNLI test sets. SNLI+MultiNLI$^2$: finetuned twice. M, U, and T are unlabeled datasets with respective inconsistency losses.

Our goal is to minimize all inconsistencies without sacrificing one for another. In Table 3, we see that lower symmetry/transitivity inconsistency generally does not reduce test accuracy, but we do not observe substantial improvement either. In conjunction with the observations from above, this suggests that test sets do not explicitly measure symmetry/transitivity consistency.

From Fig 1 and 2, we see that models constrained by both symmetry and transitivity losses are generally more consistent than models using symmetry loss alone. Further, we see that in Fig. 2, using mirrored dataset alone can even mitigate the transitivity errors. With dataset P, the transitivity inconsistency is strongly reduced by the symmetry inconsistency loss. These observations suggest that the compositionality of constraints does not pose internal conflict to the model. They are in fact beneficial to each other.

Interestingly, in Fig 2, the models trained with mirrored dataset (w/ M) become more inconsistent in transitivity measurement when using more training data. We believe there are two factors causing this. Firstly, there is a vocabulary gap between SNLI/MultiNLI data and our unlabeled datasets (U and T). Secondly, the w/ M models are trained with symmetry consistency but evaluated with transitivity consistency. The slightly rising inconsistency implies that, without vocabulary coverage, training with one consistency might not always benefit another consistency, even using more training data.

When label supervision is limited (*i.e.* 1%), the models can easily overfit via the transitivity loss. As a result, models trained on the combined losses (*i.e.* w/ M,U,T) have slightly larger transitivity inconsistency than models trained with mirrored data (*i.e.* w/ M) alone. In fact, if we use no label supervision at all, the symmetry and transitivity losses can push every prediction towards label *Neutral*. But such predictions sacrifice annotation consistency. Therefore, we believe that some amount of label supervision is necessary.

| Data | M | U | T |
|---|---|---|---|
| 5% w/ M,U,T | 99.8 | 99.4 | 12.0 |
| 100% w/ M,U,T | 98.7 | 97.6 | 6.8 |

Table 4: Coverage (%) of unlabeled training sentences during the first epoch of training. Percentages are calculated from models with random seed 1.

## 5 Analysis

In this section, we present an analysis of how the different losses affect model prediction and how informative they are during training.

### 5.1 Coverage of Unlabeled Dataset

Table 4 shows the *coverage* of the three unlabeled datasets during the first training epoch. Specifically, we count the percentage of unlabeled examples where the symmetry/transitivity loss is positive. The coverage decreases in subsequent epochs as the model learns to minimize constraint violations. We see that both datasets M and U have high coverage. This is because that, as mentioned in §2, our loss function works in real-valued space instead of discrete decisions. The coverage of the dataset T is much lower because the compositional antecedent in transitivity statements holds less often, which naturally leads to smaller coverage, unlike the unary antecedent for symmetry.

### 5.2 Distribution of Predictions

In Table 5, we present the distribution of model predictions on the 100k evaluation example pairs for symmetry consistency. Clearly, the number of

| | | BERT $(H, P)$ | | | w/ M,U,T $(H, P)$ | | |
|---|---|---|---|---|---|---|---|
| | | E | C | N | E | C | N |
| $(P, H)$ | E | 4649 | **1491** | 14708 | 2036 | **29** | 9580 |
| | C | **1508** | 10712 | **6459** | **33** | 4025 | **627** |
| | N | 14609 | **6633** | 39231 | 9632 | **613** | 73425 |

Table 5: Distribution of predictions on the 100k evaluation data using BERT trained on 100% SNLI+MultiNLI data with random seed 1. Bold entries are symmetrically inconsistent.

| Model | Example | E | C | N |
|---|---|---|---|---|
| BERT | $(P, H)$ | 20848 | 18679 | 60473 |
| | $(H, Z)$ | 20919 | 18768 | 60313 |
| | $(P, Z)$ | 20779 | 18721 | 60500 |
| w/ M,U,T | $(P, H)$ | 11645 | 4685 | 83670 |
| | $(H, Z)$ | 11671 | 4703 | 83626 |
| | $(P, Z)$ | 11585 | 4597 | 83818 |

Table 6: Distribution of predictions on the 100k evaluation example triples. BERT: trained on the full SNLI+MultiNLI data. Predictions are from random run with seed 1.

constraint-violating (off-diagonal) predictions significantly dropped. Also note that the number of *Neutral* nearly doubled in our constrained model. This meets our expectation because the example pairs are constructed from randomly sampled sentences under the same topic.

We also present the distribution of predictions on example triples for the transitivity consistency in Table 6. As expected, with our transitivity consistency, the distribution of the label *Neutral* gets significantly higher as well. Further, in Table 7, we show the error rates of each individual transitivity consistencies. Clearly our framework mitigated the violation rates on all four statements.

While the logic-derived regularization pushes model prediction on unlabeled datasets towards *Neutral*, the accuracies on labeled test sets are not compromised. We believe this relates to the design of current NLI datasets where the three labels are balanced. But in the real world, neutrality represents potentially infinite negative space while entailments and contradictions are rarer. The total number of neutral examples across both the SNLI and MultiNLI test sets is about 7k. *Can we use these 7k examples to evaluate the nearly infinite negative space?* We believe not.

| Transitivity | BERT $\rho_T$ | BERT $\tau_T$ | w/ M,U,T $\rho_T$ | w/ M,U,T $\tau_T$ |
|---|---|---|---|---|
| $E \wedge E \rightarrow E$ | 0.7 | 16.0 | 0.2 | 15.1 |
| $E \wedge C \rightarrow C$ | 1.8 | 49.6 | 0.2 | 46.5 |
| $N \wedge E \rightarrow \neg C$ | 1.2 | 9.0 | 0.2 | 1.8 |
| $N \wedge C \rightarrow \neg E$ | 1.0 | 9.3 | 0.1 | 4.8 |

Table 7: Individual transitivity inconsistency (%) on the 100k evaluation example triples. BERT: trained on the full SNLI+MultiNLI data. Predictions are from random run with seed 1.

## 6 Related Works and Discussion

**Logic, Knowledge and Statistical Models** Using soft relaxations of Boolean formulas as loss functions has rich history in AI. The Łukasiewicz t-norm drives knowledge-driven learning and inference in probabilistic soft logic (Kimmig et al., 2012). Li and Srikumar (2019) show how to augment existing neural network architectures with domain knowledge using the Łukasiewicz t-norm. Xu et al. (2018) proposed a general framework for designing a semantically informed loss, without t-norms, for constraining a complex output space. In the same vein, Fischer et al. (2019) also proposed a framework for designing losses with logic, but using a bespoke mapping of the Boolean operators.

Our work is also conceptually related to posterior regularization (Ganchev et al., 2010) and constrained conditional models (Chang et al., 2012), which integrate knowledge with statistical models. Using posterior regularization with imitation learning, Hu et al. (2016) transferred knowledge from rules into neural parameters. Rocktäschel et al. (2015) embedded logic into distributed representations for entity relation extraction. Alberti et al. (2019) imposed answer consistency over generated questions for machine comprehension. Ad-hoc regularizers have been proposed for process comprehension (Du et al., 2019), semantic role labeling (Mehta et al., 2018), and summarization (Hsu et al., 2018).

**Natural Language Inference** In the literature, it has been shown that even highly accurate models show a decline in performance with perturbed examples. This lack of robustness of NLI models has been shown by comparing model performance on pre-defined propositional rules for swapped datasets (Wang et al., 2019) or outlining large-

scale stress tests to measure stability of models to semantic, lexical and random perturbations (Naik et al., 2018). Moreover, adversarial training examples produced by paraphrasing training data (Iyyer et al., 2018) or inserting additional seemingly important, yet unrelated, information to training instances (Jia and Liang, 2017) have been used to show model inconsistency. Finally, adversarially labeled examples have been shown to improve prediction accuracy (Kang et al., 2018) . Also related in this vein is the idea of dataset inoculation (Liu et al., 2019), where models are finetuned by exposing them to a challenging dataset.

The closest related work to this paper is probably that of Minervini and Riedel (2018), which uses the Gödel t-norm to discover adversarial examples that violate constraints. There are three major differences compared to this paper: 1) our definition of inconsistency is a strict generalization of errors of model predictions, giving us a unified framework for that includes cross-entropy as a special case, 2) our framework does not rely on the construction of adversarial datasets, and 3) we studied the interaction of annotated examples vs. unlabeled examples via constraint, showing that our constraints can yield strongly consistent model with even a small amount of label supervision.

## 7 Conclusion

In this paper, we proposed a general framework to measure and mitigate model inconsistencies. Our framework systematically derives loss functions from domain knowledge stated in logic rules to constrain model training. As a case study, we instantiated the framework on a state-of-the-art model for the NLI task, showing that models can be highly accurate and consistent at the same time. Our framework is easily extensible to other domains with rich output structure, e.g., entity relation extraction, and multilabel classification.

## Acknowledgements

We thank members of the NLP group at the University of Utah for their valuable insights and suggestions; and reviewers for pointers to related works, corrections, and helpful comments. We also acknowledge the support of NSF SaTC-1801446, and gifts from Google and NVIDIA.

## References

Chris Alberti, Daniel Andor, Emily Pitler, Jacob Devlin, and Michael Collins. 2019. Synthetic QA Corpora Generation with Roundtrip Consistency.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A Large Annotated Corpus for Learning Natural Language Inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Ming-Wei Chang, Lev Ratinov, and Dan Roth. 2012. Structured Learning with Constrained Conditional Models. *Machine learning*, 88(3):399–431.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long Short-Term Memory-Networks for Machine Reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto. 2013. Recognizing Textual Entailment: Models and Applications. *Synthesis Lectures on Human Language Technologies*, 6(4):1–220.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Xinya Du, Bhavana Dalvi, Niket Tandon, Antoine Bosselut, Wen tau Yih, Peter Clark, and Claire Cardie. 2019. Be Consistent! Improving Procedural Text Comprehension using Label Consistency. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Marc Fischer, Mislav Balunovic, Dana Drachsler-Cohen, Timon Gehr, Ce Zhang, and Martin Vechev. 2019. DL2: Training and Querying Neural Networks with Logic. In *International Conference on Learning Representations*.

Kuzman Ganchev, Jennifer Gillenwater, Ben Taskar, et al. 2010. Posterior Regularization for Structured Latent Variable Models. *Journal of Machine Learning Research*.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking NLI Systems with Sentences that Require Simple Lexical Inferences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.

Madan M Gupta and J Qi. 1991. Theory of T-norms and fuzzy inference methods. *Fuzzy Sets and Systems*, 40.

Wan-Ting Hsu, Chieh-Kai Lin, Ming-Ying Lee, Kerui Min, Jing Tang, and Min Sun. 2018. A Unified Model for Extractive and Abstractive Summarization using Inconsistency Loss. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. 2016. Harnessing Deep Neural Networks with Logic Rules. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial Example Generation with Syntactically Controlled Paraphrase Networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Dongyeop Kang, Tushar Khot, Ashish Sabharwal, and Eduard Hovy. 2018. AdvEntuRe: Adversarial Training for Textual Entailment with Knowledge-Guided Examples. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Martin Kay. 1997. The Proper Place of Men and Machines in Language Translation. *Machine Translation*, 12.

Angelika Kimmig, Stephen Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. 2012. A short Introduction to Probabilistic Soft Logic. In *Proceedings of the NIPS Workshop on Probabilistic Programming: Foundations and Applications*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations*.

Erich Peter Klement, Radko Mesiar, and Endre Pap. 2013. *Triangular Norms*. Springer Science & Business Media.

Tao Li and Vivek Srikumar. 2019. Augmenting Neural Networks with First-order Logic. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft COCO: Common Objects in Context. In *13th European Conference on Computer Vision*. Springer.

Nelson F Liu, Roy Schwartz, and Noah A Smith. 2019. Inoculation by Fine-Tuning: A Method for Analyzing Challenge Datasets. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Jonathan Mallinson, Rico Sennrich, and Mirella Lapata. 2017. Paraphrasing Revisited with Neural Machine Translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*.

Sanket Vaibhav Mehta, Jay Yoon Lee, and Jaime Carbonell. 2018. Towards Semi-Supervised Learning for Deep Semantic Role Labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Pasquale Minervini and Sebastian Riedel. 2018. Adversarially Regularising Neural NLI Models to Integrate Logical Background Knowledge. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*.

Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Rose, and Graham Neubig. 2018. Stress Test Evaluation for Natural Language Inference. In *Proceedings of the 27th International Conference on Computational Linguistics*.

Yixin Nie, Yicheng Wang, and Mohit Bansal. 2018. Analyzing Compositionality-Sensitivity of NLI Models. *The Thirty-Second AAAI Conference on Artificial Intelligence*.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A Decomposable Attention Model for Natural Language Inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Tim Rocktäschel, Sameer Singh, and Sebastian Riedel. 2015. Injecting Logical Background Knowledge into Embeddings for Relation Extraction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional Attention Flow for Machine Comprehension. *International Conference on Learning Representations*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. In *Advances in Neural Information Processing Systems*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. GLUE: A Multi-task Benchmark and Analysis Platform for Natural Language Understanding. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Haohan Wang, Da Sun, and Eric P Xing. 2019. What If We Simply Swap the Two Text Fragments? A Straightforward yet Effective Way to Test the Robustness of Methods to Confounding Signals in Nature Language Inference Tasks. *The Thirty-Third AAAI Conference on Artificial Intelligence*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck. 2018. A Semantic Loss Function for Deep Learning with Symbolic Knowledge. In *International Conference on Machine Learning*.

# A Appendices

## A.1 Violations as Generalizing Errors

Both global and conditional violations defined in the body of the paper generalize classifier error. In this section, we will show that for a dataset with only labeled examples, and no additional constraints, both are identical to error.

Recall that an example $x$ annotated with label $Y^\star$ can be written as $\top \rightarrow Y^\star(x)$. If we have a dataset $D$ of such examples and no constraints, in our unified representation of examples, we can write this as the following conjunction:

$$\forall x \in D, \quad \top \rightarrow Y^\star(x).$$

We can now evaluate the two definitions of violation for this dataset.

First, note that the denominator in the definition of the conditional violation $\tau$ counts the number of examples because the antecedent for all examples is always true. This makes $\rho$ and $\tau$ equal. Moreover, the numerator is the number of examples where the label for an example is not $Y^\star$. In other words, the value of $\rho$ and $\tau$ represents the fraction of examples in $D$ that are mislabeled.

The strength of the unified representation and the definition of violation comes from the fact that they apply to arbitrary constraints.

## A.2 Loss for Transitivity Consistency

This section shows the loss associated with the transitivity consistency in the NLI case study. For an individual example $(P, H, Z)$, applying the product t-norm to the definition of the transitivity consistency constraint, we get the loss

$$\begin{aligned}
&\text{ReLU}\left(\log e(P,H) + \log e(H,Z) - \log e(P,Z)\right) \\
+&\text{ReLU}\left(\log e(P,H) + \log c(H,Z) - \log c(P,Z)\right) \\
+&\text{ReLU}\left(\log n(P,H) + \log e(H,Z) - \log\left(1 - c(P,Z)\right)\right) \\
+&\text{ReLU}\left(\log n(P,H) + \log c(H,Z) - \log\left(1 - e(P,Z)\right)\right)
\end{aligned} \tag{14}$$

That is, the total transitivity loss $L_{tran}$ is the sum of this expression over the entire dataset.

## A.3 Details of Experiments

### A.3.1 Setup

For BERT$_{base}$ baselines, we finetune them for 3 epochs with learning rate $3 \times 10^{-5}$, warmed up for all gradient updates. For constrained models, we further finetune them for another 3 epochs with lowered learning rate $1 \times 10^{-5}$. When dataset $U$ is present, we further lower the learning rate to $5 \times 10^{-6}$. Optimizer is Adam across all runs. During training, we adopt Dropout rate (Srivastava et al., 2014) 0.1 inside of BERT transformer encoder while 0 at the final linear layer of classification.

For different types of data and different consistency constraints, we used different weighting factors $\lambda$'s. In general, we found that the smaller amount of labeled examples, the smaller $\lambda$ for the symmetry and transitivity consistency. In Table 9, we see that the $\lambda$'s for U and T grows exponentially with the size of annotated examples. In contrast, the $\lambda$ for M dataset can be much higher. We found a good value for M is 1. This is because the size of dataset $U$ and $T$ are fixed to be 100k, while the size of dataset $M$ is the same as the amount of labeled examples.

| | 1% | | | | | | 5% | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Train | SNLI | MultiNLI | $\rho_S$ | $\tau_S$ | $\rho_T$ | $\tau_T$ | SNLI | MultiNLI | $\rho_S$ | $\tau_S$ | $\rho_T$ | $\tau_T$ |
| SNLI | 79.3 | na | 36.7 | 70.6 | 6.1 | 17.1 | 84.5 | na | 26.3 | 64.4 | 4.9 | 14.8 |
| MultiNLI | na | 69.0 | 29.1 | 83.1 | 8.2 | 18.4 | na | 76.1 | 28.4 | 69.3 | 7.0 | 18.5 |
| SNLI+MultiNLI | 79.7 | 70.1 | 38.6 | 71.7 | 4.3 | 13.4 | 84.6 | 77.2 | 25.3 | 62.4 | 4.8 | 14.8 |
| SNLI+MultiNLI$^2$ | 80.3 | 71.0 | 32.4 | 75.0 | 3.9 | 12.8 | 85.3 | 77.4 | 22.1 | 67.1 | 4.1 | 13.7 |
| w/ M | 80.1 | 71.0 | 7.5 | 39.2 | 2.1 | 9.1 | 85.3 | 76.8 | 7.1 | 34.8 | 2.8 | 10.5 |
| w/ M,U | 80.2 | 71.0 | 6.1 | 38.2 | 2.5 | 9.8 | 85.4 | 77.2 | 4.6 | 32.5 | 2.0 | 8.3 |
| w/ M,U,T | 80.6 | 71.1 | 7.8 | 34.0 | 2.6 | 10.4 | 85.4 | 77.2 | 3.2 | 31.0 | 1.8 | 7.9 |
| | 20% | | | | | | 100% | | | | | |
| Train | SNLI | MultiNLI | $\rho_S$ | $\tau_S$ | $\rho_T$ | $\tau_T$ | SNLI | MultiNLI | $\rho_S$ | $\tau_S$ | $\rho_T$ | $\tau_T$ |
| SNLI | 87.5 | na | 21.2 | 63.0 | 4.1 | 13.6 | 90.1 | na | 18.6 | 60.3 | 4.7 | 14.9 |
| MultiNLI | na | 80.4 | 25.8 | 58.1 | 5.1 | 16.5 | na | 83.7 | 20.6 | 58.9 | 5.6 | 17.5 |
| SNLI+MultiNLI | 87.8 | 80.6 | 18.6 | 64.3 | 4.4 | 14.4 | 90.1 | 83.5 | 18.1 | 59.6 | 4.5 | 14.8 |
| SNLI+MultiNLI$^2$ | 87.9 | 80.7 | 19.0 | 64.0 | 4.3 | 14.5 | 90.3 | 84.0 | 19.3 | 59.7 | 4.5 | 15.2 |
| w/ M | 88.1 | 80.6 | 7.3 | 34.0 | 3.2 | 11.7 | 90.3 | 84.1 | 6.2 | 28.1 | 3.0 | 11.6 |
| w/ M,U | 88.1 | 80.9 | 1.4 | 31.2 | 1.3 | 5.8 | 90.5 | 84.3 | 1.4 | 26.8 | 1.3 | 6.3 |
| w/ M,U,T | 88.1 | 80.9 | 1.3 | 29.6 | 1.2 | 5.7 | 90.2 | 84.2 | 1.1 | 25.5 | 0.6 | 4.2 |

Table 8: Symmetry/Transitivity inconsistencies (%) for models using 1%, 5%, 20%, and 100% training data. Each number represents the average of three random runs. SNLI+MultiNLI$^2$: BERT$_{base}$ finetuned twice for fair comparison. SNLI/MultiNLI column: accuracies on corresponding text sets. M: mirrored labeled examples. U: unlabeled instance pairs. T: unlabeled instance triples.

| Data | 1% | 5% | 20% | 100% |
|---|---|---|---|---|
| M | 1 | 1 | 1 | 1 |
| U | $10^{-5}$ | $10^{-4}$ | $10^{-3}$ | $10^{-1}$ |
| T | $10^{-6}$ | $10^{-5}$ | $10^{-4}$ | $10^{-3}$ |

Table 9: Choice of $\lambda$'s for different consistency and corresponding unlabeled datasets. For different sizes of annotation and different types of data, we adopt different $\lambda$'s.

Having larger $\lambda$ leads to significantly worse accuracy on the development set, especially that of SNLI. Therefore we did not select such models for evaluation. We hypothesize that it is because the SNLI and MultiNLI are crowdsourced from different domains while the MS COCO shares the same domain as the SNLI. Larger scaling factor could push unlabeled examples towards *Neutral*, thus sacrificing the annotation consistency on SNLI examples.

### A.3.2 Results

We present the full experiment results on the natural language inference task in Table 8. Note that the accuracies of baselines finetuned twice are slightly better than models only finetuned once, while their symmetry/transitivity consistencies are roughly on par. We found such observation is consistent with different finetuning hyperparameters (e.g. warming, epochs, learning rate).

# Style Transfer for Texts: Retrain, Report Errors, Compare with Rewrites

**Alexey Tikhonov***
Yandex
Berlin, Germany
altsoph@gmail.com

**Viacheslav Shibaev***
Ural Federal University
Ekaterinburg, Russia

**Aleksander Nagaev**
Ural Federal University/Sberbank
Ekaterinburg, Russia

**Aigul Nugmanova**
Speech Technology Center
St. Petersburg, Russia

**Ivan P. Yamshchikov***
Max Planck Institute
for Mathematics in the Sciences
Leipzig, Germany
ivan@yamshchikov.info

## Abstract

This paper shows that standard assessment methodology for style transfer has several significant problems. First, the standard metrics for style accuracy and semantics preservation vary significantly on different re-runs. Therefore one has to report error margins for the obtained results. Second, starting with certain values of bilingual evaluation understudy (BLEU) between input and output and accuracy of the sentiment transfer the optimization of these two standard metrics diverge from the intuitive goal of the style transfer task. Finally, due to the nature of the task itself, there is a specific dependence between these two metrics that could be easily manipulated. Under these circumstances, we suggest taking BLEU between input and human-written reformulations into consideration for benchmarks. We also propose three new architectures that outperform state of the art in terms of this metric.

## 1 Introduction

Deep generative models attract a lot of attention in recent years (Hu et al., 2017b). Such methods as variational autoencoders (Kingma and Welling, 2013) or generative adversarial networks (Goodfellow et al., 2014) are successfully applied to a variety of machine vision problems including image generation (Radford et al., 2017), learning interpretable image representations (Chen et al., 2016) and style transfer for images (Gatys et al., 2016). However, natural language generation is more challenging due to many reasons, such as the discrete nature of textual information (Hu et al., 2017a), the absence of local information continuity and non-smooth disentangled representations (Bowman et al., 2015). Due to these difficulties, text generation is mostly limited to specific narrow applications and is usually working in supervised settings.

Content and style are deeply fused in natural language, but style transfer for texts is often addressed in the context of disentangled latent representations (Hu et al., 2017a; Shen et al., 2017; Fu et al., 2018; John et al., 2018; Romanov et al., 2018; Tian et al., 2018). Intuitive understanding of this problem is apparent: if an input text has some attribute $A$, a system generates new text similar to the input on a given set of attributes with only one attribute $A$ changed to the target attribute $\tilde{A}$. In the majority of previous works, style transfer is obtained through an encoder-decoder architecture with one or multiple style discriminators to learn disentangled representations. The encoder takes a sentence as an input and generates a style-independent content representation. The decoder then takes the content representation and the target style representation to generate the transformed sentence. In (Subramanian et al., 2018) authors question the quality and usability of the disentangled representations for texts and suggest an end-to-end approach to style transfer similar to an end-to-end machine translation.

Contribution of this paper is three-fold: 1) we show that different style transfer architectures have varying results on test and that reporting error margins for various training re-runs of the same model is especially important for adequate assessment of the models accuracy, see Figure 1; 2) we show that BLEU (Papineni et al., 2002) between input and output and accuracy of style transfer measured in terms of the accuracy of a pre-trained external style classifier can be manipulated and naturally diverge from the intuitive goal of the style transfer task starting from a certain threshold; 3) new architectures that perform style transfer using improved latent representations are shown to outperform state of the art in terms of BLEU between output and

* Equal contribution



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3936–3945,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

human-written reformulations.

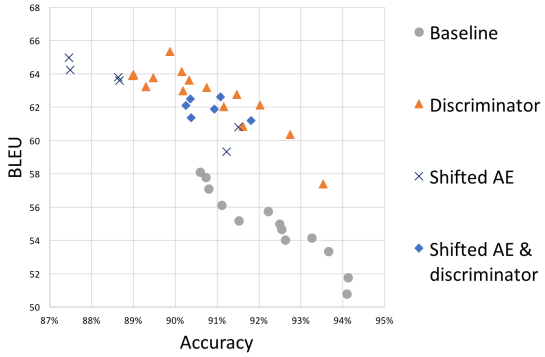


Figure 1: Test results of multiple runs for four different architectures retrained several times from scratch. In-depth description of the architectures can be found in Section 3.

## 2 Related Work

Style of a text is a very general notion that is hard to define in rigorous terms (Xu, 2017). However, the style of a text can be characterized quantitatively (Hughes et al., 2012); stylized texts could be generated if a system is trained on a dataset of stylistically similar texts (Potash et al., 2015); and author-style could be learned end-to-end (Tikhonov and Yamshchikov, 2018b,c; Vechtomova et al., 2018). A majority of recent works on style transfer focus on the sentiment of text and use it as a target attribute. For example, in (Li et al., 2018; Kabbara and Cheung, 2016; Xu et al., 2018) estimate the quality of the style transfer with binary sentiment classifier trained on the corpora further used for the training of the style-transfer system. (Ficler and Goldberg, 2017) and especially (Fu et al., 2018) generalize this ad-hoc approach defining a style as a set of arbitrary quantitively measurable categorial or continuous parameters. Such parameters could include the *'style of the time'* (Hughes et al., 2012), author-specific attributes (see (Xu et al., 2012) or (Jhamtani et al., 2017) on 'shakespearization'), politeness (Sennrich et al., 2016), formality of speech (Rao and Tetreault, 2018), and gender or even political slant (Prabhumoye et al., 2018).

A significant challenge associated with narrowly defined style-transfer problems is that finding a good solution for one aspect of a style does not guarantee that you can use the same solution for a different aspect of it. For example, Guu et al. (2018) build a generative model for sentiment transfer with a retrieve-edit approach. In (Li et al., 2018) a delete-retrieve model shows good results for sentiment transfer. However, it is hard to imagine that these retrieval approaches could be used, say, for the style of the time or formality, since in these cases the system is often expected to paraphrase a given sentence to achieve the target style.

In (Hu et al., 2017a) the authors propose a more general approach to the controlled text generation combining variational autoencoder (VAE) with an extended wake-sleep mechanism in which the sleep procedure updates both the generator and external classifier that assesses generated samples and feedbacks learning signals to the generator. Authors had concatenated labels for style with the text representation of the encoder and used this vector with "hard-coded" information about the sentiment of the output as the input of the decoder. This approach seems promising, and some other papers either extend it or use similar ideas. Shen et al. (2017) applied a GAN to align the hidden representations of sentences from two corpora using an adversarial loss to decompose information about the form. In (Zhao et al., 2017) model learns a smooth code space and can be used as a discrete GAN with the ability to generate coherent discrete outputs from continuous samples. Authors use two different generators for two different styles. In (Fu et al., 2018) an adversarial network is used to make sure that the output of the encoder does not have style representation. (Hu et al., 2017a) also uses an adversarial component that ensures there is no stylistic information within the representation. Fu et al. (2018) do not use a dedicated component that controls the semantic component of the latent representation. Such a component is proposed by John et al. (2018) who demonstrate that decomposition of style and content could be improved with an auxiliary multi-task for label prediction and adversarial objective for bag-of-words prediction. Romanov et al. (2018) also introduces a dedicated component to control semantic aspects of latent representations and an adversarial-motivational training that includes a special motivational loss to encourage a better decomposition. Speaking about preservation of semantics one also has to mention works on paraphrase systems, see, for example (Prakash et al., 2016; Gupta et al., 2018; Roy and Grangier, 2019). The methodology described in this paper could be extended to paraphrasing systems in terms of semantic preservation measurement, however,

this is the matter of future work.

Subramanian et al. (2018) state that learning a latent representation, which is independent of the attributes specifying its style, is rarely attainable. There are other works on style transfer that are based on the ideas of neural machine translation with (Carlson et al., 2018) and without parallel corpora (Zhang et al., 2018) in line with (Lample et al., 2017) and (Artetxe et al., 2017).

It is important to underline here that majority of the papers dedicated to style transfer for texts treat sentiment of a sentence as a stylistic rather than semantic attribute despite particular concerns (Tikhonov and Yamshchikov, 2018a). It is also crucial to mention that in line with (Fu et al., 2018) majority of the state of the art methods for style transfer use an external pre-trained classifier to measure the accuracy of the style transfer. BLEU computes the harmonic mean of precision of exact matching n-grams between a reference and a target sentence across the corpus. It is not sensitive to minute changes, but BLEU between input and output is often used as the coarse measure of the semantics preservation. For the corpora that have human written reformulations, BLEU between the output of the model and human text is used. These metrics are used alongside with a handful of others such as PINC (Paraphrase In N-gram Changes) score (Carlson et al., 2018), POS distance (Tian et al., 2018), language fluency (John et al., 2018), etc. Figure 2 shows self-reported results of different models in terms of two most frequently measured performance metrics, namely, BLEU and Accuracy of the style transfer.

This paper focuses on Yelp! reviews dataset[1] that was lately enhanced with human written reformulations by (Li et al., 2018). These are Yelp! reviews, where each short English review of a place is labeled as a negative or as a positive once. This paper studies three metrics that are most common in the field at the moment and questions to which extent can they be used for the performance assessment. These metrics are the accuracy of an external style classifier that is trained to measure the accuracy of the style transfer, BLEU between input and output of a system, and BLEU between output and human-written texts.

[1]https://www.yelp.com/dataset

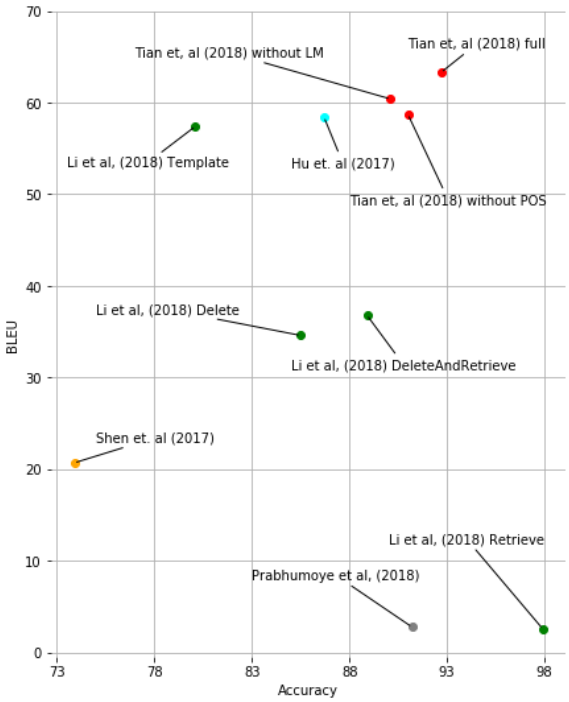


Figure 2: Overview of the self-reported results for sentiment transfer on Yelp! reviews. Results of (Romanov et al., 2018) are not displayed due to the absence of self-reported BLEU scores. Later in the paper we show that on different reruns BLEU and accuracy can vary from these self-reported single results.

## 3 Style transfer

In this work we experiment with extensions of a model, described in (Hu et al., 2017a), using Texar (Hu et al., 2018) framework. To generate plausible sentences with specific semantic and stylistic features every sentence is conditioned on a representation vector $z$ which is concatenated with a particular code $c$ that specifies desired attribute, see Figure 3. Under notation introduced in (Hu et al., 2017a) the base autoencoder (AE) includes a conditional probabilistic encoder $E$ defined with parameters $\theta_E$ to infer the latent representation $z$ given input $x$

$$z \sim E(x) = q_E(z, c|x).$$

Generator $G$ defined with parameters $\theta_G$ is a GRU-RNN for generating and output $\hat{x}$ defined as a sequence of tokens $\hat{x} = \hat{x}_1, ..., \hat{x}_T$ conditioned on the latent representation $z$ and a stylistic component $c$ that are concatenated and give rise to a generative distribution

$$\hat{x} \sim G(z, c) = p_G(\hat{x}|z, c).$$

These encoder and generator form an AE with the following loss

$$\mathcal{L}_{ae}(\theta_G, \theta_E; x, c) = -\mathbb{E}_{q_E(z,c|x)} \left[\log q_G(x|z,c)\right]. \quad (1)$$

This standard reconstruction loss that drives the generator to produce realistic sentences is combined with two additional losses. The first discriminator provides extra learning signals which enforce the generator to produce coherent attributes that match the structured code in $c$. Since it is impossible to propagate gradients from the discriminator through the discrete sample $\hat{x}$, we use a deterministic continuous approximation a "soft" generated sentence, denoted as $\tilde{G} = \tilde{G}_\tau(z, c)$ with "temperature" $\tau$ set to $\tau \to 0$ as training proceeds. The resulting soft generated sentence is fed into the discriminator to measure the fitness to the target attribute, leading to the following loss

$$\mathcal{L}_c(\theta_G, \theta_E; x) = -\mathbb{E}_{q_E(z,c|x)} \left[\log q_D(c|\tilde{G})\right]. \quad (2)$$

Finally, under the assumption that each structured attribute of generated sentences is controlled through the corresponding code in $c$ and is independent from $z$ one would like to control that other not explicitly modelled attributes do not entangle with $c$. This is addressed by the dedicated loss

$$\mathcal{L}_z(\theta_G; x) = -\mathbb{E}_{q_E(z,c|x)q_D(c|x)} \left[\log q_E(z|\tilde{G})\right]. \quad (3)$$

The training objective for the baseline, shown in Figure 3, is therefore a sum of the losses from Equations (1) – (3) defined as

$$min_{\theta_G} \mathcal{L}_{baseline} = \mathcal{L}_{ae} + \lambda_c \mathcal{L}_c + \lambda_z \mathcal{L}_z, \quad (4)$$

where $\lambda_c$ and $\lambda_z$ are balancing parameters.

Let us propose two further extensions of this baseline architecture. To improve reproducibility of the research the code of the studied models is open[2]. Both extensions aim to improve the quality of information decomposition within the latent representation. In the first one, shown in Figure 4, a special dedicated discriminator is added to the model to control that the latent representation does not contain stylistic information. The loss of this discriminator is defined as

$$\mathcal{L}_{D_z}(\theta_G; x, c) = -\mathbb{E}_{q_E(z|x)} \left[\log q_{D_z}(c|z)\right]. \quad (5)$$

[2]https://github.com/VAShibaev/text_style_transfer

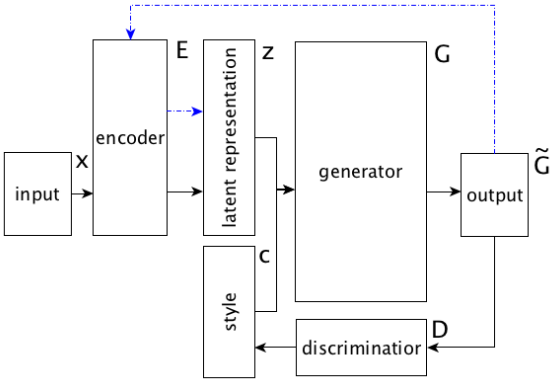


Figure 3: The generative model, where style is a structured code targeting sentence attributes to control. Blue dashed arrows denote the proposed independence constraint of latent representation and controlled attribute, see (Hu et al., 2017a) for the details.

Here a discriminator denoted as $D_z$ is trying to predict code $c$ using representation $z$. Combining the loss defined by Equation (4) with the adversarial component defined in Equation (5) the following learning objective is formed

$$min_{\theta_G} \mathcal{L} = \mathcal{L}_{baseline} - \lambda_{D_z} \mathcal{L}_{Dz}, \quad (6)$$

where $\mathcal{L}_{baseline}$ is a sum defined in Equation (4), $\lambda_{D_z}$ is a balancing parameter.

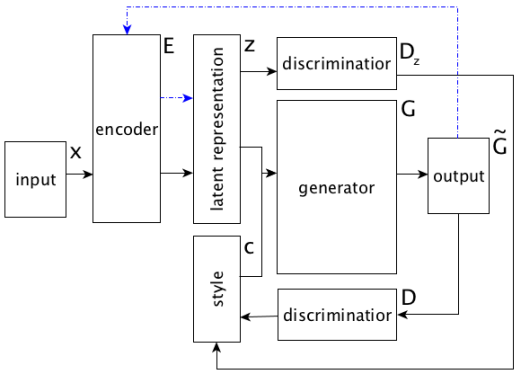


Figure 4: The generative model with dedicated discriminator introduced to ensure that semantic part of the latent representation does not have information on the style of the text.

The second extension of the baseline architecture does not use an adversarial component $D_z$ that is trying to eradicate information on $c$ from component $z$. Instead, the system, shown in Figure 5 feeds the "soft" generated sentence $\tilde{G}$ into encoder $E$ and checks how close is the representation $E(\tilde{G})$

to the original representation $z = E(x)$ in terms of the cosine distance. We further refer to it as *shifted autoencoder* or SAE. Ideally, both $E(\tilde{G}(E(x), c))$ and $E(\tilde{G}(E(x), \bar{c}))$, where $\bar{c}$ denotes an inverse style code, should be both equal to $E(x)$[3]. The loss of the shifted autoencoder is

$$min_{\theta_G}\mathcal{L} = \mathcal{L}_{baseline} + \lambda_{cos}\mathcal{L}_{cos} + \lambda_{cos^-}\mathcal{L}_{cos^-}, \tag{7}$$

where $\lambda_{cos}$ and $\lambda_{cos^-}$ are two balancing parameters, with two additional terms in the loss, namely, cosine distances between the softened output processed by the encoder and the encoded original input, defined as

$$\mathcal{L}_{cos}(x, c) = \cos\left(E(\tilde{G}(E(x), c)), E(x)\right),$$
$$\mathcal{L}_{cos^-}(x, c) = \cos\left(E(\tilde{G}(E(x), \bar{c})), E(x)\right). \tag{8}$$

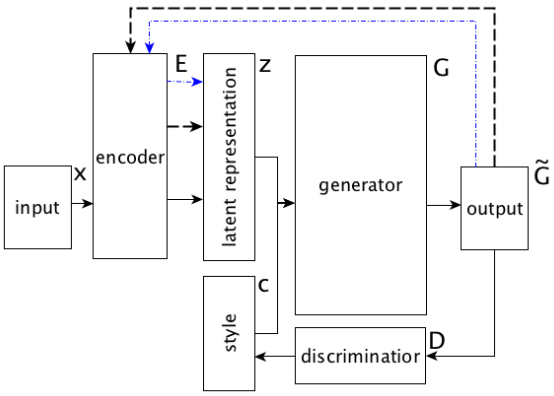


Figure 5: The generative model with a dedicated loss added to control that semantic representation of the output, when processed by the encoder, is close to the semantic representation of the input.

We also study a combination of both approaches described above, shown on Figure 6.

In Section 4 we describe a series of experiments that we have carried out for these architectures using Yelp! reviews dataset.

## 4 Experiments

We have found that the baseline, as well as the proposed extensions, have noisy outcomes, when retrained from scratch, see Figure 1. Most of the papers mentioned in Section 2 measure the performance of the methods proposed for the sentiment transfer with two metrics: accuracy of the external sentiment classifier measured on test data, and BLEU between the input and output that is regarded as a coarse metric for semantic similarity.

[3]This notation is valid under the assumption that every stylistic attribute is a binary feature

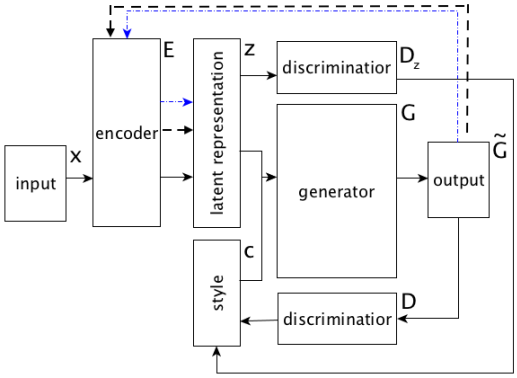


Figure 6: A combination of an additional discriminator used in Figure 4 with a shifted autoencoder shown in Figure 5

In the first part of this section, we demonstrate that reporting error margins is essential for the performance assessment in terms that are prevalent in the field at the moment, i.e., BLEU between input and output and accuracy of the external sentiment classifier. In the second part, we also show that both of these two metrics after a certain threshold start to diverge from an intuitive goal of the style transfer and could be manipulated.

### 4.1 Error margins matter

On Figure 1 one can see that the outcomes for every single rerun differ significantly. Namely, accuracy can change up to 5 percentage points, whereas BLEU can vary up to 8 points. This variance can be partially explained with the stochasticity incurred due to sampling from the latent variables. However, we show that results for state of the art models sometimes end up within error margins from one another, so one has to report the margins to compare the results rigorously. More importantly, one can see that there is an inherent trade-off between these two performance metrics. This trade-off is not only visible across models but is also present for the same retrained architecture. Therefore, improving one of the two metrics is not enough to confidently state that one system solves the style-transfer problem better than the other. One has to report error margins after several consecutive retrains and instead of comparing one of the two metrics has to talk about Pareto-like optimization that would show confident improvement of both.

To put obtained results into perspective, we have retrained every model from scratch five times in a row. We have also retrained the models of Tian et al. (2018) five times since their code is published online. Figure 7 shows the results of all models with error margins. It is also enhanced with other self-reported results on the same Yelp! review dataset for which no code was published.

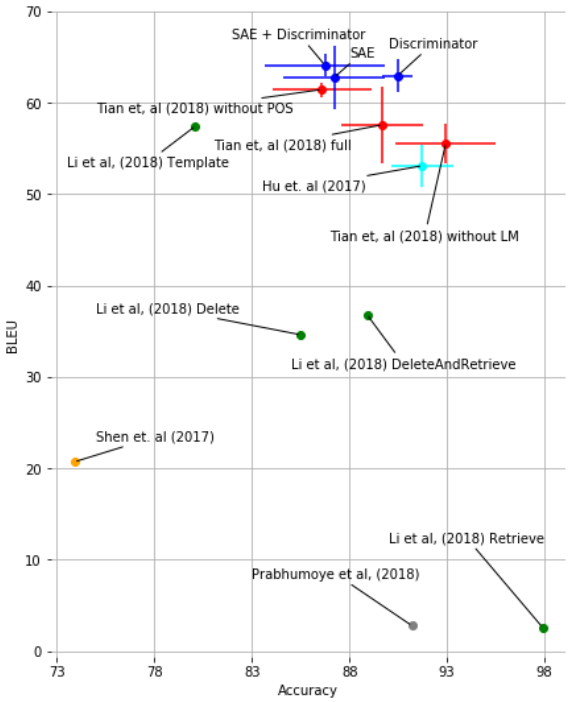


Figure 7: Overview of the self-reported results for sentiment transfer on Yelp! reviews alongside with the results for the baseline model (Hu et al., 2017a), architecture with additional discriminator, shifted autoencoder (SAE) with additional cosine losses, and a combination of these two architectures averaged after five re-trains alongside with architectures proposed by (Tian et al., 2018) after five consecutive re-trains. Results of (Romanov et al., 2018) are not displayed due to the absence of self-reported BLEU scores.

One can see that error margins of the models, for which several reruns could be performed, overlap significantly. In the next subsection, we carefully study BLEU and accuracy of the external classifier and discuss their aptness to measure style transfer performance.

### 4.2 Delete, duplicate and conquer

One can argue that as there is an inevitable entanglement between semantics and stylistics in natural language, there is also an apparent entanglement between BLEU of input and output and accuracy estimation of the style. Indeed, the output that copies input gives maximal BLEU yet clearly fails in terms of the style transfer. On the other hand, a wholly rephrased sentence could provide a low BLEU between input and output but high accuracy. These two issues are not problematic when both BLEU between input and output and accuracy of the transfer are relatively low. However, since style transfer methods have significantly evolved in recent years, some state of the art methods are now sensitive to these issues. The trade-off between these two metrics can be seen in Figure 1 as well as in Figure 7.

As we have mentioned above, the accuracy of an external classifier and BLEU between output and input are the most widely used methods to assess the performance of style transfer at this moment. However, both of these metrics can be manipulated in a relatively simple manner. One can extend the generative architecture with internal pre-trained classifier of style and then perform the following heuristic procedure:

- measure the style accuracy on the output for a given batch;
- choose the sentences that style classifier labels as incorrect;
- replace them with duplicates of sentences from the given batch that have correct style according to the internal classifier and show the highest BLEU with given inputs.

This way One can replace all sentences that push measured accuracy down and boost reported accuracy to 100%. To see the effect that this manipulation has on the key performance metric we split all sentences with wrong style in 10 groups of equal size and replaces them with the best possible duplicates of the stylistically correct sentences group after group. The results of this process are shown in Figure 8.

This result is disconcerting. Simply replacing part of the output with duplicates of the sentences that happen to have relatively high BLEU with given inputs allows to "boost" accuracy to 100% and "improve" BLEU. The change of BLEU during such manipulation stays within error margins of the architecture, but accuracy is significantly manipulated. What is even more disturbing is that BLEU between such manipulated output of the batch and human-written reformulations provided in (Tian

et al., 2018) also grows. Figure 8 shows that for SAE but all four architectures described in Section 3 demonstrate similar behavior.

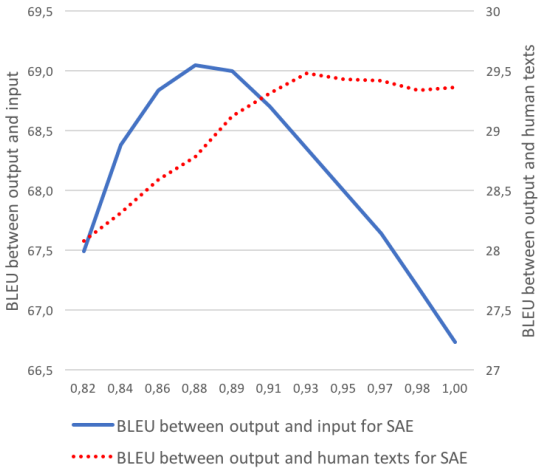


Figure 8: Manipulating the generated output in a way that boosts accuracy one can change BLEU between output and input. Moreover, such manipulation increases BLEU between output and human written reformulations. The picture shows behavior of SAE, but other architectures demonstrate similar behavior. The results are an average of four consecutive retrains of the same architecture.

Our experiments show that though we can manipulate BLEU between output and human-written text, it tends to change monotonically. That might be because of the fact that this metric incorporates information on stylistics and semantics of the text at the same time, preserving inevitable entanglement that we have mentioned earlier. Despite being costly, human-written reformulations are needed for future experiments with style transfer. It seems that modern architectures have reached a certain level of complexity for which naive proxy metrics such as accuracy of an external classifier or BLEU between output and input are already not enough for performance estimation and should be combined with BLEU between output and human-written texts. As the quality of style transfer grows further one has to improve the human-written data sets: for example, one would like to have data sets similar to the ones used for machine translation with several reformulations of the same sentence.

On Figure 9 one can see how new proposed architectures compare with another state of the art approaches in terms of BLEU between output and human-written reformulations.

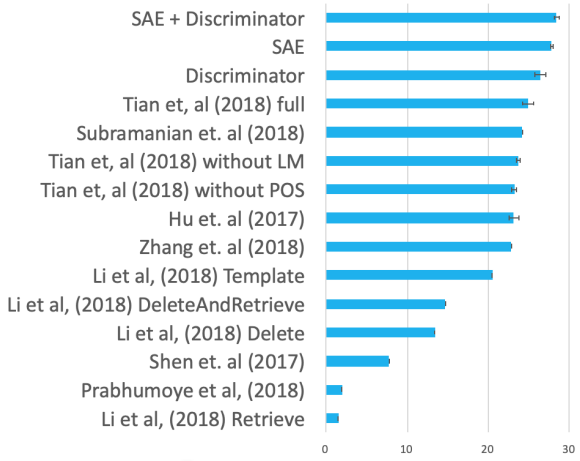


Figure 9: Overview of the BLEU between output and human-written reformulations of Yelp! reviews. Architecture with additional discriminator, shifted autoencoder (SAE) with additional cosine losses, and a combination of these two architectures measured after five re-runs outperform the baseline by (Hu et al., 2017a) as well as other state of the art models. Results of (Romanov et al., 2018) are not displayed due to the absence of self-reported BLEU scores

## 5 Conclusion

Style transfer is not a rigorously defined NLP problem. Starting from definitions of style and semantics and finishing with metrics that could be used to evaluate the performance of a proposed system. There is a surge of recent contributions that work on this problem. This paper highlights several issues connected with this lack of rigor. First, it shows that the state of the art algorithms are inherently noisy on the two most widely accepted metrics, namely, BLEU between input and output and accuracy of the external style classifier. This noise can be partially attributed to the adversarial components that are often used in the state of the art architectures and partly due to certain methodological inconsistencies in the assessment of the performance. Second, it shows that reporting error margins of several consecutive retrains for the same model is crucial for the comparison of different architectures, since error margins for some of the models overlap significantly. Finally, it demonstrates that even BLEU on human-written reformulations can be manipulated in a relatively simple way.

## References


Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2017. Unsupervised neural machine translation. In *arXiv preprint*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642.

Keith Carlson, Allen Riddell, and Daniel Rockmore. 2018. Evaluating prose style transfer with the bible. *Royal Society open science*, 5(10):171920.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *Advances in neural information processing systems*, pages 2172–2180.

Jessica Ficler and Yoav Goldberg. 2017. Controlling linguistic style aspects in neural language generation. In *Proceedings of the Workshop on Stylistic Variation*, volume 94-104.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2018. Style transfer in text: Exploration and evaluation. *AAAI*.

Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. 2016. Image style transfer using convolutional neural network. *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference*, pages 2414–2423.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *Advances in neural information processing systems*, pages 2672–2680.

Ankush Gupta, Arvind Agarwal, Prawaan Singh, and Piyush Rai. 2018. A deep generative framework for paraphrase generation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Kelvin Guu, Tatsunori B. Hashimoto, Yonatan Oren, and Percy Liang. 2018. Generating sentences by editing prototypes. *Transactions of the Association of Computational Linguistics*, 6:437–450.

Zhiting Hu, Haoran Shi, Zichao Yang, Bowen Tan, Tiancheng Zhao, Junxian He, Wentao Wang, Xingjiang Yu, Lianhui Qin, Di Wang, Xuezhe Ma, Hector Liu, Xiaodan Liang, Wanrong Zhu, Devendra Singh Sachan, and Eric P. Xing. 2018. Texar: A modularized, versatile, and extensible toolkit for text generation. In *arXiv preprint*.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017a. Toward controlled generation of text. In *International Conference on Machine Learning*, pages 1587–1596.

Zhiting Hu, Zichao Yang, Ruslan Salakhutdinov, and Eric Xing. 2017b. On unifying deep generative models. In *arXiv preprint*.

James M. Hughes, Nicholas J. Foti, David C. Krakauer, and Daniel N. Rockmore. 2012. Quantitative patterns of stylistic influence in the evolution of literature. *Proceedings of the National Academy of Sciences*, 109(20):7682–7686.

Harsh Jhamtani, Varun Gangal, Eduard Hovy, and Eric Nyberg. 2017. Shakespearizing modern language using copy-enriched sequence-to-sequence models. In *Proceedings of the Workshop on Stylistic Variation*, pages 10–19.

Vineet John, Lili Mou, Hareesh Bahuleyan, and Olga Vechtomova. 2018. Disentangled representation learning for text style transfer. In *arXiv preprint*.

Jad Kabbara and Jackie Chi Kit Cheung. 2016. Stylistic transfer in natural language generation systems using recurrent neural networks. *Proceedings of the Workshop on Uphill Battles in Language Processing: Scaling Early Achievements to Robust Methods*, pages 43–47.

Diederik P. Kingma and Max Welling. 2013. Auto-encoding variational bayes. In *arXiv preprint*.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2017. Unsupervised machine translation using monolingual corpora only. In *arXiv preprint*.

Juncen Li, Robin Jia, He He, and Percy Liang. 2018. Delete, retrieve, generate: A simple approach to sentiment and style transfer. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 865–1874.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Gbleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 311–318.

Peter Potash, Alexey Romanov, and Anna Rumshisky. 2015. Ghostwriter: Using an lstm for automatic rap lyric generation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1919–1924. Association for Computational Linguistics.

Shrimai Prabhumoye, Yulia Tsvetkov, Alan W. Black, and Ruslan Salakhutdinov. 2018. Style transfer through back-translation. In *arXiv preprint*.

Aaditya Prakash, Sadid A. Hasan, Kathy Lee, Vivek Datla, Ashequl Qadir, Joey Liu, and Oladimeji Farri. 2016. Neural paraphrase generation with stacked residual lstm networks. In *arXiv preprint*.

Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. 2017. Learning to generate reviews and discovering sentiment. In *arXiv preprint*.

Sudha Rao and Joel Tetreault. 2018. Dear sir or madam, may i introduce the gyafc dataset: Corpus, benchmarks and metrics for formality style transfer. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 129–140.

Alexey Romanov, Anna Rumshisky, Anna Rogers, and David Donahue. 2018. Adversarial decomposition of text representation. In *arXiv preprint*.

Aurko Roy and David Grangier. 2019. Unsupervised paraphrasing without translation. In *arXiv preprint*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Controlling politeness in neural machine translation via side constraints. *In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 35–40.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. *31st Conference on Neural Information Processing Systems*, pages 6833–6844.

Sandeep Subramanian, Guillaume Lample, Eric M. Smith, Ludovic Denoyer, Marc Aurelio Ranzato, and Y-Lan Boureau. 2018. Multiple-attribute text style transfer. In *arXiv preprint*.

Youzhi Tian, Zhiting Hu, and Zhou Yu. 2018. Structured content preservation for unsupervised text style transfer. In *arXiv preprint*.

Alexe Tikhonov and Ivan P. Yamshchikov. 2018a. What is wrong with style transfer for texts? In *arXiv preprint*.

Alexey Tikhonov and Ivan P. Yamshchikov. 2018b. Guess who? Multilingual approach for the automated generation of author-stylized poetry. In *IEEE Spoken Language Technology Workshop (SLT)*, pages 787–794.

Alexey Tikhonov and Ivan P. Yamshchikov. 2018c. Sounds Wilde. Phonetically extended embeddings for author-stylized poetry generation. In *Proceedings of the Fifteenth Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 117–124.

Olga Vechtomova, Hareesh Bahuleyan, Amirpasha Ghabussi, and Vineet John. 2018. Generating lyrics with variational autoencoder and multi-modal artist embeddings. In *arXiv preprint*.

Jingjing Xu, Xu Sun, Qi Zeng, Xuancheng Ren, Xiaodong Zhang, Houfeng Wang, and Wenjie Li. 2018. Unpaired sentiment-to-sentiment translation: A cycled reinforcement learning approach. In *arXiv preprint*.

Wei Xu. 2017. From shakespeare to twitter: What are language styles all about? *Proceedings of the Workshop on Stylistic Variation*, pages 1–9.

Wei Xu, Alan Ritter, William B. Dolan, Ralph Grishman, and Colin Cherry. 2012. Paraphrasing for style. *Proceedings of COLING*, pages 2899–2914.

Zhirui Zhang, Shuo Ren, Shujie Liu, Jianyong Wang, Peng Chen, Mu Li, Ming Zhou, and Enhong Chen. 2018. Style transfer as unsupervised machine translation. In *arXiv preprint*.

Junbo Jake Zhao, Yoon Kim, Kelly Zhang, Alexander M. Rush, and Yann LeCun. 2017. Adversarially regularized autoencoders for generating discrete structures. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. CoRR.

## A Supplemental Material

Here are some examples characteristic for different systems. An output of a system follows the input. Here are some successful examples produced by the system with additional discriminator:

- it's not much like an actual irish pub, which is depressing. → it's definitely much like an actual irish pub, which is grateful.
- i got a bagel breakfast sandwich and it was delicious! → i got a bagel breakfast sandwich and it was disgusting!
- i love their flavored coffee. → i dumb their flavored coffee.
- i got a bagel breakfast sandwich and it was delicious! → i got a bagel breakfast sandwich and it was disgusting!
- i love their flavored coffee. → i dumb their flavored coffee.
- nice selection of games to play. → typical selection of games to play.
- i'm not a fan of huge chain restaurants. → i'm definitely a fan of huge chain restaurants.

Here are some examples of typical faulty reformulations:

- only now i'm really hungry, and really pissed off. → kids now i'm really hungry, and really extraordinary off.
- what a waste of my time and theirs. → what a wow. of my time and theirs.
- cooked to perfection and very flavorful. → cooked to pain and very outdated.

- the beer was nice and cold! → the beer was nice and consistant!
- corn bread was also good! → corn bread was also unethical bagged

Here are some successful examples produced by the SAE:

- our waitress was the best, very accommodating. → our waitress was the worst, very accommodating.
- great food and awesome service! → horrible food and nasty service!
- their sandwiches were really tasty. → their sandwiches were really bland.
- i highly recommend the ahi tuna. → i highly hated the ahi tuna.
- other than that, it's great! → other than that, it's horrible!

Here are some examples of typical faulty reformulations by SAE:

- good drinks, and good company. → 9:30 drinks, and 9:30 company.
- like it's been in a fridge for a week. → like it's been in a fridge for a true.
- save your money & your patience. → save your smile & your patience.
- no call, no nothing. → deliciously call, deliciously community.
- sounds good doesn't it? → sounds good does keeps it talented

Here are some successful examples produced by the SAE with additional discriminator:

- best green corn tamales around. → worst green corn tamales around.
- she did the most amazing job. → she did the most desperate job.
- very friendly staff and manager. → very inconsistent staff and manager.
- even the water tasted horrible. → even the water tasted great.
- go here, you will love it. → go here, you will avoid it.

Here are some examples of typical faulty reformulations by the SAE with additional discriminator:

- _num_ - _num_ % capacity at most , i was the only one in the pool. → sweetness - stylish % fountains at most, i was the new one in the
- this is pretty darn good pizza! → this is pretty darn unsafe pizza misleading
- enjoyed the dolly a lot. → remove the shortage a lot.
- so, it went in the trash. → so, it improved in the hooked.
- they are so fresh and yummy. → they are so bland and yummy.

# Implicit Deep Latent Variable Models for Text Generation

**Le Fang**[†], **Chunyuan Li**[§], **Jianfeng Gao**[§], **Wen Dong**[†], **Changyou Chen**[†]
[†]University at Buffalo, [§]Microsoft Research, Redmond
{lefang, wendong, changyou}@buffalo.edu
{chunyl, jfgao}@microsoft.com

## Abstract

Deep latent variable models (LVM) such as variational auto-encoder (VAE) have recently played an important role in text generation. One key factor is the exploitation of smooth latent structures to guide the generation. However, the representation power of VAEs is limited due to two reasons: (1) the Gaussian assumption is often made on the variational posteriors; and meanwhile (2) a notorious "posterior collapse" issue occurs. In this paper, we advocate sample-based representations of variational distributions for natural language, leading to implicit latent features, which can provide flexible representation power compared with Gaussian-based posteriors. We further develop an LVM to directly match the aggregated posterior to the prior. It can be viewed as a natural extension of VAEs with a regularization of maximizing mutual information, mitigating the "posterior collapse" issue. We demonstrate the effectiveness and versatility of our models in various text generation scenarios, including language modeling, unaligned style transfer, and dialog response generation. The source code to reproduce our experimental results is available on GitHub[1].

## 1 Introduction

Deep latent variable models (LVM) such as variational auto-encoder (VAE) (Kingma and Welling, 2013; Rezende et al., 2014) are successfully applied for many natural language processing tasks, including language modeling (Bowman et al., 2015; Miao et al., 2016), dialogue response generation (Zhao et al., 2017b), controllable text generation (Hu et al., 2017) and neural machine translation (Shah and Barber, 2018) etc. One advantage of VAEs is the flexible distribution-based latent representation. It captures holistic properties of input, such as style, topic, and high-level linguistic/semantic features, which further guide the generation of diverse and relevant sentences.

However, the representation capacity of VAEs is restrictive due to two reasons. The first reason is rooted in the assumption of variational posteriors, which usually follow spherical Gaussian distributions with diagonal co-variance matrices. It has been shown that an approximation gap generally exists between the true posterior and the best possible variational posterior in a restricted family (Cremer et al., 2018). Consequently, the gap may militate against learning an optimal generative model, as its parameters may be always updated based on sub-optimal posteriors (Kim et al., 2018). The second reason is the so-called *posterior collapse* issue, which occurs when learning VAEs with an auto-regressive decoder (Bowman et al., 2015). It produces undesirable outcomes: the encoder yields meaningless posteriors that are very close to the prior, while the decoder tends to ignore the latent codes in generation (Bowman et al., 2015). Several attempts have been made to alleviate this issue (Bowman et al., 2015; Higgins et al., 2017; Zhao et al., 2017a; Fu et al., 2019; He et al., 2019).

These two seemingly unrelated issues are studied independently. In this paper, we argue that the posterior collapse issue is partially due to the restrictive Gaussian assumption, as it limits the optimization space of the encoder/decoder in a given distribution family. $(i)$ To break the assumption, we propose to use sample-based representations for natural language, thus leading to implicit latent features. Such a representation is much more expressive than Gaussian-based posteriors. $(ii)$ This implicit representation allows us to extend VAE and develop new LVM that further mitigate the posterior collapse issue. It represents all the sentences in the dataset as posterior samples in the latent space, and matches the aggregated posterior samples to the prior distribution. Consequently,

[1] https://github.com/fangleai/Implicit-LVM



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3946–3956, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

latent features are encouraged to cooperate and behave diversely to capture meaningful information for each sentence.

However, learning with implicit representations faces one challenge: it is intractable to evaluate the KL divergence term in the objectives. We overcome the issue by introducing a conjugate-dual form of the KL divergence (Rockafellar et al., 1966; Dai et al., 2018). It facilitates learning via training an auxiliary dual function. The effectiveness of our models is validated by producing consistently state-of-the-art results on a broad range of generation tasks, including language modeling, unsupervised style transfer, and dialog response generation.

## 2 Preliminaries

When applied to text generation, VAEs (Bowman et al., 2015) consist of two parts, a generative network (decoder) and an inference network (encoder). Given a training dataset $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^{|\mathcal{D}|}$, where $\boldsymbol{x}_i = [x_{1i}, \cdots, x_{Ti}]$ represents $i$th sentence of length $T$. Starting from a prior distribution $p(\boldsymbol{z})$, VAE generates a sentence $\boldsymbol{x}$ using the deep generative network $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$, where $\boldsymbol{\theta}$ is the network parameter. Therefore, the joint distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})$ is defined as $p(\boldsymbol{z})p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$. The prior $p(\boldsymbol{z})$ is typically assumed as a standard multivariate Gaussian. Due to the sequential nature of natural language, the decoder $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ takes an auto-regressive form $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) = \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(x_t|x_{<t}, \boldsymbol{z})$. The goal of model training is to maximize the marginal data log-likelihood $\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\log p_{\boldsymbol{\theta}}(\boldsymbol{x})$.

However, it is intractable to perform posterior inference. An $\boldsymbol{\phi}$-parameterized encoder is introduced to approximate $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}) \propto p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})$ with a variational distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$. Variational inference is employed for VAE learning, yielding following evidence lower bound (ELBO):

$$\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) \geq \mathcal{L}_1 = \mathcal{L}_2 \text{ with}$$

$$\mathcal{L}_1 = \mathcal{L}_E + \mathcal{L}_R \text{ where} \quad (1)$$

$$\mathcal{L}_E = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[\mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\right] \quad (2)$$

$$\mathcal{L}_R = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[-\mathsf{KL}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \parallel p(\boldsymbol{z})\right)\right] \quad (3)$$

$$\mathcal{L}_2 = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) + \mathcal{L}_G \text{ where} \quad (4)$$

$$\mathcal{L}_G = -\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\mathsf{KL}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})\right) \quad (5)$$

Note that $\mathcal{L}_1$ and $\mathcal{L}_2$ provide two different views for the VAE objective:

- $\mathcal{L}_1$ consists of a reconstruction error term $\mathcal{L}_E$ and a KL divergence regularization term $\mathcal{L}_R$. With a strong auto-regressive decoder $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$, the objective tends to degenerate all encoding distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ to the prior, causing $\mathcal{L}_R \to 0$, *i.e.,* the posterior collapse issue.
- $\mathcal{L}_2$ indicates that VAE requests a flexible encoding distribution family to minimize the approximation gap $\mathcal{L}_G$ between the true posterior and the best possible encoding distribution. This motivates us to perform more flexible posterior inference with implicit representations.

## 3 The Proposed Models

We introduce a sample-based latent representation for natural language, and develop two models that leverage its advantages. (1) When replacing the Gaussian variational distributions with sample-based distributions in VAEs, we derive *implicit VAE* (iVAE). (2) We further extend VAE to maximize mutual information between latent representations and observed sentences, leading to a variant termed as $\text{iVAE}_{\text{MI}}$.

### 3.1 Implicit VAE

**Implicit Representations** Instead of assuming an explicit density form such as Gaussian, we define a sampling mechanism to represent $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ as a set of sample $\{\boldsymbol{z}_{\boldsymbol{x},i}\}_{i=1}^{M}$, through the encoder as

$$\boldsymbol{z}_{\boldsymbol{x},i} = G_{\boldsymbol{\phi}}(\boldsymbol{x}, \epsilon_i), \epsilon_i \sim q(\epsilon) \quad (6)$$

where the $i$-th sample is drawn from a neural network $G_{\boldsymbol{\phi}}$ that takes $(\boldsymbol{x}, \epsilon_i)$ as input; $q(\epsilon)$ is a simple distribution such as standard Gaussian. It is difficult to naively combine the random noise $\epsilon$ with the sentence $\boldsymbol{x}$ (a sequence of discrete tokens) as the input of $G_{\boldsymbol{\phi}}$. Our solution is to concatenate noise $\epsilon_i$ with hidden representations $\boldsymbol{h}$ of $\boldsymbol{x}$. $\boldsymbol{h}$ is generated using a LSTM encoder, as illustrated in Figure 1.

**Dual form of KL-divergence** Though theoretically promising, the implicit representations in (6) render difficulties in optimizing the KL term $\mathcal{L}_R$ in (3), as the functional form is no longer tractable with implicit $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$. We resort to evaluating its dual form based on Fenchel duality theorem (Rockafellar et al., 1966; Dai et al., 2018):

$$\mathsf{KL}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \parallel p(\boldsymbol{z})\right) \quad (7)$$

$$= \max_{\nu} \mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\nu_{\boldsymbol{\psi}}(\boldsymbol{x}, \boldsymbol{z}) - \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})}\exp(\nu_{\boldsymbol{\psi}}(\boldsymbol{x}, \boldsymbol{z})),$$

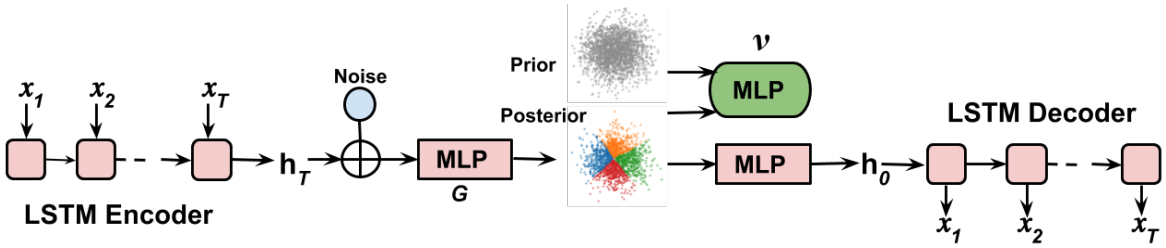


Figure 1: Illustration of the proposed implicit LVMs. $\nu(\boldsymbol{x}, \boldsymbol{z})$ is used in iVAE, and $\nu(\boldsymbol{z})$ is used in iVAE$_{\text{MI}}$. In this example, the piror is $p(\boldsymbol{z}) = \mathcal{N}(0, 1)$, the sample-based aggregated posterior $q(\boldsymbol{z}) = \int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})q(\boldsymbol{x})d\boldsymbol{x}$ for four observations is shown, where the posterior $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ for each observation is visualized in a different color.

where $\nu_{\boldsymbol{\psi}}(\boldsymbol{x}, \boldsymbol{z})$ is an auxiliary dual function, parameterized by a neural network with weights $\boldsymbol{\psi}$. By replacing the KL term with this dual form, the implicit VAE has the following objective:

$$\begin{aligned}\mathcal{L}_{\text{iVAE}} = &\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) \\ &- \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\nu_{\boldsymbol{\psi}}(\boldsymbol{x}, \boldsymbol{z}) \\ &+ \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})}\exp(\nu_{\boldsymbol{\psi}}(\boldsymbol{x}, \boldsymbol{z})), \end{aligned} \tag{8}$$

**Training scheme** Implicit VAE inherits the end-to-end training scheme of VAEs with extra work on training the auxiliary network $\nu_{\boldsymbol{\psi}}(\boldsymbol{x}, \boldsymbol{z})$:

- Sample a mini-batch of $\boldsymbol{x}_i \sim \mathcal{D}$, $\epsilon_i \sim q(\epsilon)$, and generate $\boldsymbol{z}_{\boldsymbol{x}_i,\epsilon_i} = G(\boldsymbol{x}_i, \epsilon_i; \boldsymbol{\phi})$; Sample a mini-batch of $\boldsymbol{z}_i \sim p(\boldsymbol{z})$.
- Update $\boldsymbol{\psi}$ in $\nu_{\boldsymbol{\psi}}(\boldsymbol{x}, \boldsymbol{z})$ to maximize

$$\sum_i \nu_{\boldsymbol{\psi}}(\boldsymbol{x}_i, \boldsymbol{z}_{\boldsymbol{x}_i,\epsilon_i}) - \sum_i \exp(\nu_{\boldsymbol{\psi}}(\boldsymbol{x}_i, \boldsymbol{z}_i)) \tag{9}$$

- Update parameters $\{\boldsymbol{\phi}, \boldsymbol{\theta}\}$ to maximize

$$\sum_i \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_i|\boldsymbol{z}_{\boldsymbol{x}_i,\epsilon_i}) - \sum_i \nu_{\boldsymbol{\psi}}(\boldsymbol{x}_i, \boldsymbol{z}_{\boldsymbol{x}_i,\epsilon_i}) \tag{10}$$

In practice, we implement $\nu_{\boldsymbol{\psi}}(\boldsymbol{x}, \boldsymbol{z})$ with a multi-layer perceptron (MLP), which takes the concatenation of $\boldsymbol{h}$ and $\boldsymbol{z}$. In another word, the auxiliary network distinguishes between $(\boldsymbol{x}, \boldsymbol{z}_{\boldsymbol{x}})$ and $(\boldsymbol{x}, \boldsymbol{z})$, where $\boldsymbol{z}_{\boldsymbol{x}}$ is drawn from the posterior and $\boldsymbol{z}$ is drawn from the prior, respectively. We found the MLP-parameterized auxiliary network converges faster than LSTM encoder and decoder (Hochreiter and Schmidhuber, 1997). This means that the auxiliary network practically provides an accurate approximation to the KL regularization $\mathcal{L}_R$.

### 3.2 Mutual Information Regularized iVAE

It is noted the inherent deficiency of the original VAE objective in (3): the KL divergence regularization term matches each posterior distribution independently to the same prior. This is prone to posterior collapse in text generation, due to a strong auto-regressive decoder $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$. When sequentially generating $x_t$, the model learns to solely rely on the ground-truth $[x_1, \cdots, x_{t-1}]$, and ignore the dependency from $\boldsymbol{z}$ (Fu et al., 2019). It results in the learned variational posteriors $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ to exactly match $p(\boldsymbol{z})$, without discriminating data $\boldsymbol{x}$.

To better regularize the latent space, we propose to replace $\mathcal{L}_R = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[-\mathsf{KL}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \parallel p(\boldsymbol{z})\right)\right]$ in (3), with the following KL divergence:

$$-\mathsf{KL}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z}) \parallel p(\boldsymbol{z})\right), \tag{11}$$

where $q_{\boldsymbol{\phi}}(\boldsymbol{z}) = \int q(\boldsymbol{x})q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})d\boldsymbol{x}$ is the *aggregated posterior*, $q(\boldsymbol{x})$ is the empirical data distribution for the training dataset $\mathcal{D}$. The integral is estimated by ancestral sampling in practice, *i.e.* we first sample $\boldsymbol{x}$ from dataset and then sample $\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$.

In (11), variational posterior is regularized as a whole $q_{\boldsymbol{\phi}}(\boldsymbol{z})$, encouraging posterior samples from different sentences to cooperate to satisfy the objective. It implies a solution that each sentence is represented as a local region in the latent space, the aggregated representation of all sentences match the prior; This avoids the degenerated solution from (3) that the feature representation of individual sentence spans over the whole space.

**Connection to mutual information** The proposed latent variable model coincides with (Zhao et al., 2018, 2017a) where mutual information is introduced into the optimization, based on the following decomposition result (Please see detailed

proof in Appendix A):

$$-\mathsf{KL}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z}) \parallel p(\boldsymbol{z})\right) = I(\boldsymbol{x}, \boldsymbol{z}) - \mathbb{E}_{\boldsymbol{x}}\mathsf{KL}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \parallel p(\boldsymbol{z})\right),$$

where $I(\boldsymbol{x}, \boldsymbol{z})$ is the mutual information between $\boldsymbol{z}$ and $\boldsymbol{x}$ under the joint distribution $q_{\boldsymbol{\phi}}(\boldsymbol{x}, \boldsymbol{z}) = q(\boldsymbol{x})q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$. Therefore, the objective in (11) also maximizes the mutual information between individual sentences and their latent features. We term the new LVM objective as iVAE$_{\text{MI}}$:

$$\mathcal{L}_{\text{iVAE}_{\text{MI}}} = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) - \mathsf{KL}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z}) \parallel p(\boldsymbol{z})\right) \tag{12}$$

**Training scheme** Note that the aggregated posterior $q_{\boldsymbol{\phi}}(\boldsymbol{z})$ is also a sample-based distribution. Similarly, we evaluate (12) through its dual form:

$$\begin{aligned}&\mathsf{KL}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z}) \parallel p(\boldsymbol{z})\right) \\ =&\max_{\nu}\mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z})}\nu_{\boldsymbol{\psi}}(\boldsymbol{z}) - \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})}\exp(\nu_{\boldsymbol{\psi}}(\boldsymbol{z})).\end{aligned} \tag{13}$$

Therefore, iVAE$_{\text{MI}}$ in (12) can be written as:

$$\begin{aligned}\mathcal{L}_{\text{iVAE}_{\text{MI}}} =& \mathbb{E}_{\boldsymbol{x}\sim D}\mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) \\ &- \mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z})}\nu_{\boldsymbol{\psi}}(\boldsymbol{z}) + \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})}\exp(\nu_{\boldsymbol{\psi}}(\boldsymbol{z})),\end{aligned} \tag{14}$$

where the auxiliary network $\nu_{\boldsymbol{\psi}}(\boldsymbol{z})$ is parameterized as a neural network. Different from iVAE, $\nu_{\boldsymbol{\psi}}(\boldsymbol{z})$ in iVAE$_{\text{MI}}$ only takes posterior samples as input. The training algorithm is similar to iVAE in Section 3.1, except a different auxiliary network $\nu_{\boldsymbol{\psi}}(\boldsymbol{z})$. In Appendix B, we show the full algorithm of iVAE$_{\text{MI}}$.

We illustrate the proposed methods in Figure 1. Note that both iVAE and iVAE$_{\text{MI}}$ share the same model architecture, except a different auxiliary network $\nu$.

# 4 Related Work

## 4.1 Solutions to posterior collapse

Several attempts have been made to alleviate the posterior collapse issue. The KL annealing scheme in language VAEs has been first used in (Bowman et al., 2015). An effective cyclical KL annealing schedule is used in (Fu et al., 2019), where the KL annealing process is repeated multiple times. KL term weighting scheme is also adopted in $\beta$-VAE (Higgins et al., 2017) for disentanglement. On model architecture side, dilated CNN was considered to replace auto-regressive LSTMs for decoding (Yang et al., 2017). The bag-of-word auxiliary loss was proposed to improve the dependence on latent representations in generation (Zhao et al., 2017b). More recently, lagging inference proposes to aggressively update encoder multiple times before a single decoder update (He et al., 2019). Semi-amortized VAE refines variational parameters from an amortized encoder per instance with stochastic variational inference (Kim et al., 2018).

All these efforts utilize the Gaussian-based forms for posterior inference. Our paper is among the first ones to attribute posterior collapse issue to the restrictive Gaussian assumption, and advocate more flexible sample-based representations.

## 4.2 Implicit Feature Learning

Sample-based distributions, as well as implicit features, have been widely used in representation learning (Donahue et al., 2017; Li et al., 2017a). Vanilla autoencoders learn point masses of latent features rather than their distributions. Adversarial variational Bayes introduces an auxiliary discriminator network like GANs (Goodfellow et al., 2014; Makhzani et al., 2015) to learn almost arbitrarily distributed latent variables (Mescheder et al., 2017; Pu et al., 2017b). We explore the similar spirit in the natural language processing (NLP) domain. Amortized MCMC and particle based methods are introduced for LVM learning in (Li et al., 2017d; Pu et al., 2017a; Chen et al., 2018). Coupled variational Bayes (Dai et al., 2018) emphasizes an optimization embedding, *i.e.*, a flow of particles, in a general setting of non-parametric variational inference. It also utilizes similar dual form with auxiliary function $\nu_{\boldsymbol{\psi}}(\boldsymbol{x}, \boldsymbol{z})$ to evaluate KL divergence. Adversarially regularized autoencoders (Makhzani et al., 2015; Kim et al., 2017) use similar objectives with iVAEs, in the form of a reconstruction error plus a specific regularization evaluated with implicit samples. Mutual information has also been considered into regularization in (Zhao et al., 2018, 2017a) to obtain more informative representations.

Most previous works focus on image domain. It is largely unexplored in NLP. Further, the auto-regressive decoder renders an additional challenge when applying implicit latent representations. Adversarial training with samples can be empirically unstable, and slow even applying recent stabilization techniques in GANs (Arjovsky et al., 2017; Gulrajani et al., 2017). To the best of our knowledge, this paper presents the first to effectively apply implicit feature representations, to NLP.

| Dataset | PTB | Yahoo | Yelp |
|---|---|---|---|
| num. sents. | 42,000 | 100,000 | 100,000 |
| min. len. | 2 | 21 | 21 |
| max. len. | 78 | 201 | 201 |
| avg. len. | 21.9 | 79.9 | 96.7 |

Table 1: Statistics of datasets for language modeling.

## 5 Experiments

In this section, the effectiveness of our methods is validated by largely producing state-of-the-art metrics on a broad range of text generation tasks under various scenarios.

### 5.1 Language Modeling

**Datasets.** We consider three public datasets, the Penn Treebank (PTB) (Marcus et al., 1993; Bowman et al., 2015), Yahoo, and Yelp corpora (Yang et al., 2017; He et al., 2019). PTB is a relatively small dataset with sentences of varying lengths, whereas Yahoo and Yelp contain larger amounts of with long sentences. Detailed statistics of these datasets are shown in Table 1.

**Settings** We implement both encoder and decoder as one-layer LSTM. As illustrated in Figure 1, concatenation of the last hidden state of the encoder LSTM and a Gaussian noise is fed into a MLP to draw a posterior sample. When decoding, a latent sample is fed into a MLP to produce the initial hidden state of the decoder LSTM. A KL-term annealing scheme is also used (Bowman et al., 2015). We list more detailed hyper-parameters and architectures in Appendix C.1.2.

**Baselines.** We compare with several state-of-the-art VAE language modeling methods, including (1) VAEs with a monotonically KL-annealing schedule (Bowman et al., 2015); (2) $\beta$-VAE (Higgins et al., 2017), VAEs with a small penalty on KL term scaled by $\beta$; (3) SA-VAE (Kim et al., 2018), mixing instance-specific variational inference with amortized inference; (4) Cyclical VAE (Fu et al., 2019) that periodically anneals the KL term; (5) Lagging VAE, which aggressively updates encoder multiple times before a single decoder update.

**Evaluation metrics.** Two categories of metrics are used to study VAEs for language modeling:

- To characterize the modeling ability of the observed sentences, we use the negative ELBO as the sum of reconstruction loss and KL term, as well as perplexity (PPL).
- Compared with traditional neural language models, VAEs has its unique advantages in feature learning. To measure the quality of learned features, we consider (1) KL: $\mathsf{KL}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \parallel p(\boldsymbol{z})\right)$; (2) Mutual information (MI) $I(\boldsymbol{x}, \boldsymbol{z})$ under the joint distribution $q_{\boldsymbol{\phi}}(\boldsymbol{x}, \boldsymbol{z})$; (3) Number of active units (AU) of latent representation. The activity of a latent dimension $z$ is measured as $A_{\boldsymbol{z}} = \mathrm{Cov}_{\boldsymbol{x}}\left(\mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x})}[z]\right)$, which is defined as active if $A_{\boldsymbol{z}} > 0.01$.

| Methods | -ELBO↓ | PPL↓ | KL↑ | MI↑ | AU↑ |
|---|---|---|---|---|---|
| Dataset: PTB | | | | | |
| VAE | 102.6 | 108.26 | 1.08 | 0.8 | 2 |
| $\beta$(0.5)-VAE | 104.5 | 117.92 | **7.50** | 3.1 | 5 |
| SA-VAE | 102.6 | 107.71 | 1.23 | 0.7 | 2 |
| Cyc-VAE | 103.1 | 110.50 | 3.48 | 1.8 | 5 |
| iVAE | **87.6** | **54.46** | 6.32 | **3.5** | **32** |
| iVAE$_{\text{MI}}$ | **87.2** | **53.44** | **12.51** | **12.2** | **32** |
| Dataset: Yahoo | | | | | |
| VAE | 328.6 | 61.21 | 0.0 | 0.0 | 0 |
| $\beta$(0.4)-VAE | 328.7 | 61.29 | 6.3 | 2.8 | 8 |
| SA-VAE | 327.2 | 60.15 | 5.2 | 2.7 | 10 |
| Lag-VAE | 326.7 | 59.77 | 5.7 | 2.9 | 15 |
| iVAE | **309.5** | **48.22** | **8.0** | **4.4** | **32** |
| iVAE$_{\text{MI}}$ | **309.1** | **47.93** | **11.4** | **10.7** | **32** |
| Dataset: Yelp | | | | | |
| VAE | 357.9 | 40.56 | 0.0 | 0.0 | 0 |
| $\beta$(0.4)-VAE | 358.2 | 40.69 | 4.2 | 2.0 | 4 |
| SA-VAE | 355.9 | 39.73 | 2.8 | 1.7 | 8 |
| Lag-VAE | 355.9 | 39.73 | 3.8 | 2.4 | 11 |
| iVAE | **348.2** | **36.70** | **7.6** | **4.6** | **32** |
| iVAE$_{\text{MI}}$ | **348.7** | **36.88** | **11.6** | **11.0** | **32** |

Table 2: Language modeling on three datasets.

The evaluation of implicit LVMs is unexplored in language models, as there is no analytical forms for the KL term. We consider to evaluate both $\mathsf{KL}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z}) \parallel p(\boldsymbol{z})\right)$ and $\mathsf{KL}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \parallel p(\boldsymbol{z})\right)$ by training a fully connected $\nu$ network in Eq. (7) and (13). To avoid the inconsistency between $\nu(\boldsymbol{x}, \boldsymbol{z})$ and $\nu(\boldsymbol{z})$ networks due to training, we train them using the same data and optimizer in every iteration. We evaluate each distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ with 128 code samples per $\boldsymbol{x}$.

We report the results in Table 2. A better language model would pursue a lower negative ELBO (also lower reconstruction errors, lower PPL), and make sufficient use of the latent space (*i.e.*, maintain relatively high KL term, higher mutual infor-

| Dataset | PTB | | Yahoo | | Yelp | |
|---|---|---|---|---|---|---|
| | Re.↓ | Abs.↓ | Re.↓ | Abs.↓ | Re.↓ | Abs.↓ |
| VAE/$\beta$- | 1.0 | 1.3 | 1.0 | 5.3 | 1.0 | 5.7 |
| SA-VAE | 5.5 | 7.1 | 9.9 | 52.9 | 10.3 | 59.3 |
| Lag-VAE | - | - | 2.2 | 11.7 | 3.7 | 21.4 |
| iVAEs | 1.4 | 1.8 | 1.3 | 6.9 | 1.3 | 7.5 |

Table 3: Total training time in hours: absolute time and relative time versus VAE.

| | |
|---|---|
| $t$ =0 | in new york the company declined comment |
| $t$ =0.1 | in new york the company declined comment |
| $t$ =0.2 | in new york the transaction was suspended |
| $t$ =0.3 | in the securities company said yesterday |
| $t$ =0.4 | in other board the transaction had disclosed |
| $t$ =0.5 | other of those has been available |
| $t$ =0.6 | both of companies have been unchanged |
| $t$ =0.7 | both men have received a plan to restructure |
| $t$ =0.8 | and to reduce that it owns |
| $t$ =0.9 | and to continue to make prices |
| $t$ =1 | and they plan to buy more today |

Table 4: Interpolating latent representation.

mation and more active units). Under all these metrics, the proposed iVAEs achieve much better performance. The posterior collapse issue is largely alleviated as indicated by the improved KL and MI values, especially with iVAE$_{\text{MI}}$ which directly takes mutual information into account.

The comparison on training time is shown in Table 3. iVAE and iVAE$_{\text{MI}}$ requires updating an auxiliary network, it spends 30% more time than traditional VAEs. This is more efficient than SA-VAE and Lag-VAE.

**Latent space interpolation.** One favorable property of VAEs (Bowman et al., 2015; Zhao et al., 2018) is to provide smooth latent representations that capture sentence semantics. We demonstrate this by interpolating two latent feature, each of which represents a unique sentence. Table 4 shows the generated examples. We take two sentences $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, and obtain their latent features as the sample-averaging results for $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$, respectively, from the implicit encoder, and then greedily decode conditioned on the interpolated feature $\boldsymbol{z}_t = \boldsymbol{z}_1 \cdot (1-t) + \boldsymbol{z}_2 \cdot t$ with $t$ increased from 0 to 1 by a step size of 0.1. It generates sentences with smooth semantic evolution.

**Improved Decoder.** One might wonder whether the improvements come from simply having a more flexible encoder during evaluation, rather than from utilizing high-quality latent features, and learning a better decoder. We use PTB to confirm our findings. We draw samples from the prior $p(\boldsymbol{z})$, and greedily decode them using the trained decoder. The quality of the generated text is evaluated by an external library "KenLM Language Model Toolkit" (Heafield et al., 2013) with two metrics (Kim et al., 2017): (1) Forward PPL: the fluency of the generated text based on language models derived from the PTB training corpus; (2) Reverse PPL: the fluency of PTB corpus based on language model derived from the generated text, which measures the extent to which the generations are representative of the PTB underlying language model. For both the PPL numbers, the lower the better. We use n=5 for n-gram language models in "KenLM".

| Model | Forward↓ | Reverse↓ |
|---|---|---|
| VAE | 18494 | 10149 |
| Cyc-VAE | 3390 | 5587 |
| AE | 672 | 2589 |
| $\beta$(0)-VAE | 625 | 1897 |
| $\beta$(0.5)-VAE | 939 | 4078 |
| SA-VAE | 341 | 10651 |
| iVAE | **116** | **1520** |
| iVAE$_{\text{MI}}$ | **134** | **1137** |

Table 5: Forward and reverse PPL on PTB.

As shown in Table 5, implicit LVMs outperform others in both PPLs, which confirms that the implicit representation can lead to better decoders. The vanilla VAE model performs the worst. This is expected, as the posterior collapse issue results in poor utilization of a latent space. Besides, we can see that iVAE$_{\text{MI}}$ generates comparably fluent but more diverse text than pure iVAE, from the lower reverse PPL values. This is reasonable, due to the ability of iVAE$_{\text{MI}}$ to encourage diverse latent samples per sentence with the aggregated regularization in the latent space.

## 5.2 Unaligned style transfer

We next consider the task of unaligned style transfer, which represents a scenario to generate text with desired specifications. The goal is to control one style aspect of a sentence that is independent of its content. We consider non-parallel corpora of sentences, where the sentences in two corpora have the same content distribution but with different styles, and no paired sentence is provided.

**Model Extension.** The success of this task depends on the exploitation of the distributional equivalence of content to learn sentiment independent content code and decoding it to a different style. To ensure such independence, we extend iVAE$_{\text{MI}}$ by adding a sentiment classifier loss to its objective (14), similar to the previous style transfer methods (Shen et al., 2017; Kim et al., 2017). Let $y$ be the style attribute, $\boldsymbol{x}_p$ and $\boldsymbol{x}_n$ (with corresponding features $\boldsymbol{z}_p$, $\boldsymbol{z}_n$) be sentences with positive and negative sentiments , respectively. The style classifier loss $\mathcal{L}_{\text{class}}(\boldsymbol{z}_p, \boldsymbol{z}_n)$ is the cross entropy loss of a binary classifier.

The classifier and encoder are trained adversarially: (1) the classifier is trained to distinguish latent features with different sentiments; (2) the encoder is trained to fool the classifier in order to remove distinctions of content features from sentiments. In practice, the classifier is implemented as a MLP. We implement two separate decoder LSTMs for clean sentiment decoding: one for positive $p(\boldsymbol{x}|\boldsymbol{z}, y = 1)$, and one for negative sentiment $p(\boldsymbol{x}|\boldsymbol{z}, y = 0)$. The prior $p(\boldsymbol{z})$ is also implemented as an implicit distribution, via transforming noise from a standard Gaussian through a MLP. Appendix C.2.2 lists more details.

**Baseline.** We compare iVAE$_{\text{MI}}$ with a state-of-the-art unaligned sentiment transferer: the adversarially regularized autoencoder (ARAE) (Kim et al., 2017), which directly regularizes the latent space with Wasserstein distance measure.

**Datasets.** Following (Shen et al., 2017), the `Yelp` restaurant reviews dataset is processed from the original `Yelp` dataset in language modeling. Reviews with user rating above three are considered *positive*, and those below three are considered *negative*. The pre-processing allows sentiment analysis on sentence level with feasible sentiment, ending up with shorter sentences with each at most 15 words than those in language modeling. Finally, we get two sets of unaligned reviews: 250K negative sentences, and 350K positive ones. Other dataset details are shown in Appendix C.2.1.

**Evaluation Metrics.** (1) Acc: the accuracy of transferring sentences into another sentiment measured by an automatic classifier: the "fasttext" library (Joulin et al., 2017); (2) BLEU: the consistency between the transferred text and the original; (3) PPL: the reconstruction perplexity of original sentences without altering sentiment; (4) RPPL: the reverse perplexity that evaluates the training corpus based on language model derived from the generated text, which measures the extent to which the generations are representative of the training corpus; (5) Flu: human evaluated index on the fluency of transferred sentences when read alone (1-5, 5 being most fluent as natural language); (6) Sim: the human evaluated similarity between the original and the transferred sentences in terms of their contents (1-5, 5 being most similar). Note that the similarity measure doesn't care sentiment but only the topic covered by sentences. For human evaluation, we show 1000 randomly selected pairs of original and transferred sentences to crowdsourcing readers, and ask them to evaluate the "Flu" and "Sim" metrics stated above. Each measure is averaged among crowdsourcing readers.

| | |
|---|---|
| **Input**: | it was super dry and had a weird taste to the entire slice . |
| **ARAE**: | it was super nice and the owner was super sweet and helpful . |
| **iVAE$_{\text{MI}}$**: | it was super tasty and a good size with the best in the burgh . |
| **Input**: | so i only had half of the regular fries and my soda . |
| **ARAE**: | it 's the best to eat and had a great meal . |
| **iVAE$_{\text{MI}}$**: | so i had a huge side and the price was great . |
| **Input**: | i am just not a fan of this kind of pizza . |
| **ARAE**: | i am very pleased and will definitely use this place . |
| **iVAE$_{\text{MI}}$**: | i am just a fan of the chicken and egg roll . |

| | |
|---|---|
| **Input**: | i have eaten the lunch buffet and it was outstanding ! |
| **ARAE**: | once again , i was told by the wait and was seated . |
| **iVAE$_{\text{MI}}$**: | we were not impressed with the buffet there last night . |
| **Input**: | my favorite food is kung pao beef , it is delicious . |
| **ARAE**: | my husband was on the phone , which i tried it . |
| **iVAE$_{\text{MI}}$**: | my chicken was n't warm , though it is n't delicious . |
| **Input**: | overall , it was a very positive dining experience . |
| **ARAE**: | overall , it was very rude and unprofessional . |
| **iVAE$_{\text{MI}}$**: | overall , it was a nightmare of terrible experience . |

Table 6: Sentiment transfer on `Yelp`. (Up: From negative to positive, Down: From positive to negative.)

| Model | Acc↑ | BLEU↑ | PPL↓ | RPPL↓ | Flu↑ | Sim↑ |
|---|---|---|---|---|---|---|
| ARAE | **95** | 32.5 | 6.8 | 395 | 3.6 | 3.5 |
| iVAE$_{\text{MI}}$ | 92 | **36.7** | **6.2** | **285** | **3.8** | **3.9** |

Table 7: Sentiment Transfer on `Yelp`.

As shown in Table 7, iVAE$_{\text{MI}}$ outperforms ARAE in metrics except Acc, showing that iVAE$_{\text{MI}}$ captures informative representations, generates consistently opposite sentences with similar grammatical structure and reserved semantic meaning. Both methods perform successful sentiment transfer as shown by Acc values. iVAE$_{\text{MI}}$ achieves a little lower Acc due to much more content reserving, even word reserving, of the source sentences.

Table 6 presents some examples. In each box, we show the source sentence, the transferred sentence by ARAE and iVAE$_{\text{MI}}$, respectively. We observe that ARAE usually generates new sentences that miss the content of the source, while iVAE$_{\text{MI}}$ shows better content-preserving.

### 5.3 Dialog response generation

We consider the open-domain dialog response generation task, where we need to generate a natural language response given a dialog history. It is crucial to learn a meaningful latent feature representation of the dialog history in order to generate a consistent, relevant, and contentful response that is likely to drive the conversation (Gao et al., 2019).

**Datasets.** We consider two mainstream datasets in recent studies (Zhao et al., 2017b, 2018; Fu et al., 2019; Gu et al., 2018): `Switchboard` (Godfrey and Holliman, 1997) and `Dailydialog` (Li et al., 2017c). `Switchboard` contains 2,400 two-way telephone conversations under 70 specified topics. `Dailydialog` has 13,118 daily conversations for a English learner. We process each utterance as the response of previous 10 context utterances from both speakers. The datasets are separated into training, validation, and test sets as convention: 2316:60:62 for `Switchboard` and 10:1:1 for `Dailydialog`, respectively.

**Model Extension.** We adapt iVAE$_{\text{MI}}$ by integrating the context embedding $\boldsymbol{c}$ into all model components. The prior $p(\boldsymbol{z}|\boldsymbol{c})$ is defined as an implicit mapping between context embedding $\boldsymbol{c}$ and prior samples, which is not pre-fixed but learned together with the variational posterior for more modeling flexibility. The encoder $q(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{c})$, auxiliary dual function $\nu_{\psi}(\boldsymbol{z}, \boldsymbol{c})$ and decoder $p(\boldsymbol{x}|\boldsymbol{c}, \boldsymbol{z})$ depend on context embedding $\boldsymbol{c}$ as well. Both encoder and decoder are implemented as GRUs. The utterance encoder is a bidirectional GRU with 300 hidden units in each direction. The context encoder and decoder are both GRUs with 300 hidden units. Appendix C.3.1 presents more training details.

**Baseline.** We benchmark representative baselines and state-of-the-art approaches, include: SeqGAN, a GAN based model for sequence generation (Li et al., 2017b); CVAE baseline (Zhao et al., 2017b); dialogue WAE, a conditional Wasserstein auto-encoder for response generation (Gu et al., 2018).

| Metrics | SeqGAN | CVAE | WAE | iVAE$_{\text{MI}}$ |
|---|---|---|---|---|
| Dataset: `Switchboard` | | | | |
| BLEU-R↑ | 0.282 | 0.295 | 0.394 | **0.427** |
| BLEU-P↑ | **0.282** | 0.258 | 0.254 | 0.254 |
| BLEU-F1↑ | 0.282 | 0.275 | 0.309 | **0.319** |
| BOW-A↑ | 0.817 | 0.836 | 0.897 | **0.930** |
| BOW-E↑ | 0.515 | 0.572 | 0.627 | **0.670** |
| BOW-G↑ | 0.748 | 0.846 | 0.887 | **0.900** |
| Intra-dist1↑ | 0.705 | 0.803 | 0.713 | **0.828** |
| Intra-dist2↑ | 0.521 | 0.415 | 0.651 | **0.692** |
| Inter-dist1↑ | 0.070 | 0.112 | 0.245 | **0.391** |
| Inter-dist2↑ | 0.052 | 0.102 | 0.413 | **0.668** |
| Dataset: `Dailydialog` | | | | |
| BLEU-R↑ | 0.270 | 0.265 | 0.341 | **0.355** |
| BLEU-P↑ | 0.270 | 0.222 | **0.278** | 0.239 |
| BLEU-F1↑ | 0.270 | 0.242 | **0.306** | 0.285 |
| BOW-A↑ | 0.907 | 0.923 | 0.948 | **0.951** |
| BOW-E↑ | 0.495 | 0.543 | 0.578 | **0.609** |
| BOW-G↑ | 0.774 | 0.811 | 0.846 | **0.872** |
| Intra-dist1↑ | 0.747 | **0.938** | 0.830 | 0.897 |
| Intra-dist2↑ | 0.806 | 0.973 | 0.940 | **0.975** |
| Inter-dist1↑ | 0.075 | 0.177 | 0.327 | **0.501** |
| Inter-dist2↑ | 0.081 | 0.222 | 0.583 | **0.868** |

Table 8: Dialog response generation on two datasets.

**Evaluation Metrics.** We adopt several widely used numerical metrics to systematically evaluate the response generation, including BLEU score (Papineni et al., 2002), BOW Embedding (Liu et al., 2016) and Distinct (Li et al., 2015), as used in (Gu et al., 2018). For each testing context, we sample 10 responses from each model.

- BLEU measures how much a generated response contains n-gram overlaps with the references. We compute 4-gram BLEU. For each test context, we sample 10 responses from the models and compute their BLEU scores. BLEU precision and recall is defind as the average and maximum scores, respectively (Zhao et al., 2017b).
- BOW embedding is the cosine similarity of bag-of-words embeddings between the generations and the reference. We compute three different BOW embedding: greedy, average, and extreme.
- Distinct evaluates the diversity of the generated responses: dist-n is defined as the ratio of unique n-grams (n=1,2) over all n-grams in the generated responses. We evaluate diversities for both within each sampled response and among all responses as intra-dist and inter-dist, respectively.

Tables 8 show the performance comparison. iVAE$_{\text{MI}}$ achieves consistent improvement on a majority of the metrics. Especially, the BOW embeddings and Distinct get significantly improvement, which implies that iVAE$_{\text{MI}}$ produces both meaningful and diverse latent representations.

## 6 Conclusion

We present two types of implicit deep latent variable models, iVAE and iVAE$_{\text{MI}}$. Core to these two models is the sample-based representation of the latent features in LVM, in replacement of traditional Gaussian-based distributions. Extensive experiments show that the proposed implicit LVM models consistently outperform the vanilla VAEs on three tasks, including language modeling, style transfer and dialog response generation.

### Acknowledgments

We acknowledge Yufan Zhou for helpful discussions and the reviewers for their comments.

## References

Martín Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein gan. *CoRR*, abs/1701.07875.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*.

Changyou Chen, Chunyuan Li, Liqun Chen, Wenlin Wang, Yunchen Pu, and Lawrence Carin. 2018. Continuous-time flows for efficient inference and density estimation. *ICML*.

Chris Cremer, Xuechen Li, and David Duvenaud. 2018. Inference suboptimality in variational autoencoders. *arXiv preprint arXiv:1801.03558*.

Bo Dai, Hanjun Dai, Niao He, Weiyang Liu, Zhen Liu, Jianshu Chen, Lin Xiao, and Le Song. 2018. Coupled variational bayes via optimization embedding. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 9713–9723. Curran Associates, Inc.

Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. 2017. Adversarial feature learning. *ICLR*.

Hao Fu, Chunyuan Li, Xiaodong Liu, Jianfeng Gao, Asli Celikyilmaz, and Lawrence Carin. 2019. Cyclical annealing schedule: A simple approach to mitigating kl vanishing. *NAACL*.

Jianfeng Gao, Michel Galley, and Lihong Li. 2019. Neural approaches to conversational ai. *Foundations and Trends® in Information Retrieval*, 13(2-3):127–298.

John J Godfrey and Edward Holliman. 1997. Switchboard-1 release 2. *Linguistic Data Consortium, Philadelphia*, 926:927.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc.

Xiaodong Gu, Kyunghyun Cho, Jungwoo Ha, and Sunghun Kim. 2018. Dialogwae: Multimodal response generation with conditional wasserstein autoencoder. *arXiv preprint arXiv:1805.12352*.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. 2017. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems 30*, pages 5769–5779. Arxiv: 1704.00028.

Junxian He, Daniel Spokoyny, Graham Neubig, and Taylor Berg-Kirkpatrick. 2019. Lagging inference networks and posterior collapse in variational autoencoders. *arXiv preprint arXiv:1901.05534*.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 690–696, Sofia, Bulgaria.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, volume 3.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1587–1596. JMLR. org.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics.

Yoon Kim, Sam Wiseman, Andrew Miller, David Sontag, and Alexander Rush. 2018. Semi-amortized variational autoencoders. In *International Conference on Machine Learning*, pages 2683–2692.

Yoon Kim, Kelly Zhang, Alexander M Rush, Yann LeCun, et al. 2017. Adversarially regularized autoencoders. *arXiv preprint arXiv:1706.04223*.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Chunyuan Li, Hao Liu, Changyou Chen, Yuchen Pu, Liqun Chen, Ricardo Henao, and Lawrence Carin. 2017a. Alice: Towards understanding adversarial learning for joint distribution matching. In *Advances in Neural Information Processing Systems*, pages 5495–5503.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2015. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055*.

Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. 2017b. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547*.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017c. Dailydialog: A manually labelled multi-turn dialogue dataset. *arXiv preprint arXiv:1710.03957*.

Yingzhen Li, Richard E Turner, and Qiang Liu. 2017d. Approximate inference with amortised mcmc. *arXiv preprint arXiv:1702.08343*.

Chia-Wei Liu, Ryan Lowe, Iulian V Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *arXiv preprint arXiv:1603.08023*.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. 2015. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*.

Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. 2017. Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2391–2400. JMLR. org.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *International conference on machine learning*, pages 1727–1736.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Yuchen Pu, Zhe Gan, Ricardo Henao, Chunyuan Li, Shaobo Han, and Lawrence Carin. 2017a. Vae learning via stein variational gradient descent. In *Advances in Neural Information Processing Systems*, pages 4236–4245.

Yuchen Pu, Weiyao Wang, Ricardo Henao, Liqun Chen, Zhe Gan, Chunyuan Li, and Lawrence Carin. 2017b. Adversarial symmetric variational autoencoder. In *Advances in Neural Information Processing Systems*, pages 4330–4339.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pages 1278–1286.

R Tyrrell Rockafellar et al. 1966. Extension of fenchel'duality theorem for convex functions. *Duke mathematical journal*, 33(1):81–89.

Harshil Shah and David Barber. 2018. Generative neural machine translation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 1346–1355. Curran Associates, Inc.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in neural information processing systems*, pages 6830–6841.

Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. 2017. Improved variational autoencoders for text modeling using dilated convolutions. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3881–3890. JMLR. org.

Shengjia Zhao, Jiaming Song, and Stefano Ermon. 2017a. InfoVAE: Information maximizing variational autoencoders. *arXiv preprint arXiv:1706.02262*.

Tiancheng Zhao, Kyusong Lee, and Maxine Eskenazi. 2018. Unsupervised discrete sentence representation learning for interpretable neural dialog generation. *arXiv preprint arXiv:1804.08069*.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017b. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. *arXiv preprint arXiv:1703.10960*.

# Text Emotion Distribution Learning from Small Sample: A Meta-Learning Approach

**Zhenjie Zhao** **Xiaojuan Ma**
Department of Computer Science and Engineering
Hong Kong University of Science and Technology
Clear Water Bay, Hong Kong
{zzhaoao, mxj}@cse.ust.hk

## Abstract

Text emotion distribution learning (EDL) aims to develop models that can predict the intensity values of a sentence across a set of emotion categories. Existing methods based on supervised learning require a large amount of well-labelled training data, which is difficult to obtain due to inconsistent perception of fine-grained emotion intensity. In this paper, we propose a meta-learning approach to learn text emotion distributions from a small sample. Specifically, we propose to learn low-rank sentence embeddings by tensor decomposition to capture their contextual semantic similarity, and use K-nearest neighbors (KNNs) of each sentence in the embedding space to generate sample clusters. We then train a meta-learner that can adapt to new data with only a few training samples on the clusters, and further fit the meta-learner on KNNs of a testing sample for EDL. In this way, we effectively augment the learning ability of a model on the small sample. To demonstrate the performance, we compare the proposed approach with state-of-the-art EDL methods on a widely used EDL dataset: SemEval 2007 Task 14 (Strapparava and Mihalcea, 2007). Results show the superiority of our method on small-sample emotion distribution learning.

## 1 Introduction

Analyzing emotions in text automatically is an important topic (Yadollahi et al., 2017) with widely used applications such as classifying e-commerce products reviews (Rao et al., 2016) and developing emotionally intelligent chatbots for healthcare (Fadhil and Gabrielli, 2017), to name a few. Text emotion analysis aims to recognize writers' emotional states towards particular topics or subjects (Yadollahi et al., 2017), and has attracted considerable research efforts in the last few decades (Canales and Martínez-Barco, 2014; Abdul-Mageed and Ungar, 2017; Yu et al., 2018; Zhang et al., 2018).

Existing works cast text emotion analysis into three types of tasks: single label learning (SLL), multi-label learning (MLL), and label distribution learning (LDL). In SLL, one particular name in an emotion category (referred as label in the following) is predicted (Abdul-Mageed and Ungar, 2017), e.g., joyful, angry, or sad. However, due to the correlation among different emotions, one sentence can potentially contain multiple different emotions (Yu et al., 2018). Therefore, a more practical way is to assign multiple labels to a sentence (Yu et al., 2018), namely, MLL. Label distribution learning, which is called emotion distribution learning (EDL) in emotion mining (Zhou et al., 2016), goes a step further and assigns one intensity value to each emotion, which is necessary to encode fine-grained information for comparing strength of different emotions, etc.

In general, there are three approaches to text emotion analysis tasks: lexicon-based (Staiano and Guerini, 2014), learning-based (Zhang et al., 2018), and the combination of both methods (Mudinas et al., 2012). In the lexicon-based case, one needs to collect an emotion lexicon corpus, e.g., WordNet-Affect (Strapparava and Valitutti, 2004), SentiWordNet (Esuli and Sebastiani, 2006), and applies different counting methods to aggregate the occurrences of words associated with various emotions (Staiano and Guerini, 2014). Learning-based methods, in contrast, mostly frame emotion analysis as a supervised learning problem, which requires annotated text data with emotion labels and extraction of proper sentence features. Learning-based approaches alone, or combined with lexicon-based methods, usually produce state-of-the-art (SOTA) performance (Mudinas et al., 2012; Zhang et al., 2018), and thus are widely used nowadays.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3957–3967,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

However, learning-based methods usually demand a large amount of annotated data to train models, which has become one of the performance bottlenecks. EDL in particular aims to decode fine-grained composition and magnitude of emotions in text, the human perception of which can be highly subtle and personal (Volkova et al., 2010). It is difficult, if not impossible, to collect a large-scale emotion distribution dataset with consistent, clean human labels. Therefore, developing techniques to learn from a small sample is critical for the practicality of emotion distribution analysis.

In this paper, based on meta-learning (Vilalta and Drissi, 2002; Finn et al., 2017), we propose an efficient approach to learn text emotion distribution from a small sample. To make the most of a small labeled dataset, we propose to use the K-nearest semantically similar neighbors (KNNs) of each training sample to cluster the training data, and train a meta-learner that can adapt to new testing data with only a few samples on the clusters. We can then fit the meta-learner on KNNs of each testing sample. Learning semantic similarity of sentences usually requires a large amount of data (Le and Mikolov, 2014). We propose to learn low-rank embeddings of sentences by tensor decomposition to capture their contextual semantic similarity, which works well regardless of the size of documents (Hosseinimotlagh and Papalexakis, 2018). We evaluate the proposed approach on a widely used text emotion distribution dataset: SemEval 2007 Task 14 (Strapparava and Mihalcea, 2007), and show that it outperforms the existing SOTA methods for small sample EDLs.

The contributions of this paper are: 1) we propose a novel meta-learning framework to learn text emotion distribution from a small sample; 2) we propose to learn low-rank embeddings of sentences by tensor decomposition to find semantically similar neighbors for training and adapting a meta-learner.

## 2 Related Works

We briefly review three related areas that motivate this work, including emotion distribution learning, text representation learning, and learning from a small sample.

### 2.1 Emotion Distribution Learning

Emotion distribution learning of text (Zhou et al., 2016; Zhang et al., 2018) is a recently proposed task that tries to predict the intensity values of a sentence across a set of emotion categories. Such information is important for understanding the fine-grained emotion information (Zhou et al., 2016; Zhang et al., 2018). For example, one sentence can usually invoke multiple emotional states with different levels, and existing approaches of SLL and MLL are inadequate for capturing multi-label, multi-intensity information. LDL is more suitable for such scenarios. In general, LDL methods can be classified into three categories: problem transformation, algorithm adaption, and specialized algorithms (Geng and Ji, 2013). PT-Bayes and PT-SVM are typical problem transformation methods that transform the LDL problem into an SLL problem, and use Bayes classifier and SVM to predict label distribution, respectively. AA-KNN and AA-BP are algorithm adaption methods that extend K-nearest neighbors (KNNs) and back-propagation (BP) neural networks. SA-LDSVR, SA-IIS, SA-BFGS, SA-CPNN are specialized LDL algorithms that directly parametrize and optimize LDL objectives (Geng and Ji, 2013). In terms of EDL, two different models have been proposed recently, including a maximum entropy model (Zhou et al., 2016) and a convolutional neural network model (Zhang et al., 2018), and the latter one is the SOTA.

### 2.2 Text Representation Learning

Learning numerical representation of text is usually the first step for learning-based emotion analysis, which can be categorized at two levels: word/phrase and sentence/document. At the word/phrase level, for example, Mikolov et al. (2013) propose word2vec which uses a three-layer neural network, i.e., the input layer, the projection layer, and the output layer, to learn word vectors from a large corpora in an unsupervised manner. The projection layer maps one-hot encoded input to a low-dimension vector, which is used to predict a word in the output layer. The learned word vectors of the projection layer are shown to be an effective word representation. At the sentence/document level, similar to word2vec, doc2vec (Le and Mikolov, 2014) treats sentences and context words in the same way in the input layer, and uses a three-layer neural network to learn the vector representation of sentences. Similarly, the skip-thought vector method (Kiros et al., 2015) uses sentences directly for prediction tasks,

and ignores context words used in doc2vec. Recent works show that using attention mechanism, transformer in particular (Vaswani et al., 2017), is an effective way to learn universal sentence representation with unlabeled text, such as BERT (Devlin et al., 2019) and XLNet (Yang et al., 2019). There are other ways to improve the discriminative ability of the learned features through supervised learning, such as task transfer methods like InferSent (Conneau et al., 2017).

### 2.3 Learning from A Small Sample

The success of machine learning models, especially deep learning models, heavily depends on a large amount of labeled data. But acquiring training data is usually difficult and expensive (Johnson et al., 2018). Therefore, considerable research efforts on small sample learning are emerging recently (Lake et al., 2015; Shu et al., 2018). There are generally two approaches for learning from a small sample (Shu et al., 2018): concept learning and experience learning. Concept learning means analyzing the structure of example samples by imitating human abilities like imagination, synthesis, and analysis (Lake et al., 2015; Shu et al., 2018). Experience learning means augmenting or transferring learning experience from other data or models, including data augmentation, model fine-tuning, model compression, and meta-learning (Shu et al., 2018). Meta-learning in particular is an effective way of augmenting the experience learning ability (Thrun and Pratt, 1998; Vilalta and Drissi, 2002; Finn et al., 2017).

#### 2.3.1 Meta-Learning

Meta-learning aims to increase the learning performance through experience sharing among tasks (Thrun and Pratt, 1998; Vilalta and Drissi, 2002; Finn et al., 2017). One kind of meta-learning is to train a meta-learner to update the parameters or rules of the learning models (Bengio et al., 1991). However, the complexity of this approach is usually high (Bengio et al., 1991; Finn et al., 2017). Another approach is to train a memory neural network, such as recurrent neural networks (Santoro et al., 2016), to keep a record of the different experiences among different tasks. In particular, Finn et al. (2017) find that it is possible to optimize a learner to adapt new tasks quickly only by gradient descent. In this approach, no additional parameters are needed. Recently, researchers have also extended this method to unsupervised learning (Hsu et al., 2019) and online learning scenarios (Finn et al., 2019). Meta-learning has been used for few-shot image recognition (Lee et al., 2019), agent navigation (Finn et al., 2017), low-resource machine translation (Gu et al., 2018), query generation (Huang et al., 2018), and so on.

Existing EDL methods are designed with traditional machine learning paradigms (Geng and Ji, 2013; Zhou et al., 2016; Zhang et al., 2018), which may not work well on a small sample. Due to the difficulty of annotating emotion distribution data, developing techniques of small sample EDLs is thus critical for text emotion analysis. Transferring experience with pre-trained models (Shu et al., 2018) can be an effective way to boost natural language tasks. But such methods may need an additional large training corpora, and fine-tuning models on specific domains is non-trivial (Le and Mikolov, 2014). In this paper, we are interested in how to only use the labeled data for small sample EDLs (Shu et al., 2018).

## 3 Problem Formulation

### 3.1 Emotion Distribution Learning (EDL)

We denote one sample as $(s, \mathbf{y})$, where $s$ is a sentence or document, $\mathbf{y} = (y^1, y^2, \ldots, y^C)$ is its emotion distribution label, $y^j$ denotes the intensity value of the $j$-th emotion, $C$ is the number of discrete emotions, and $\sum_j^C y^j = 1$. For EDL, we expect to train a model $f : \mathcal{S} \mapsto \mathcal{Y}$ that can accurately map any sentence $s$ in semantic space $\mathcal{S}$ to its entailed emotion distribution $\mathbf{y}$ in emotion distribution space $\mathcal{Y}$.

### 3.2 EDL from A Small Sample

Given a set of training samples $\mathcal{D}_{\text{train}} = \{(s_i, \mathbf{y}_i)\}_{i=1}^M$, we are interested in how to train a model $f$ that can predict emotion distribution effectively on a set of testing samples $\mathcal{D}_{\text{test}} = \{(s_i, \mathbf{y}_i)\}_{i=1}^N$, where $M$ is the number of training samples and is a small number (e.g., 50), $N$ is the number of testing samples.

For EDL from a small sample, we adopt the experience learning approach (Shu et al., 2018) by considering: 1) how to augment learning ability on a small number of training samples, and 2) how to learn semantic similarity on $\mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{test}}$ so that we can use K-nearest neighbors in $\mathcal{D}_{\text{train}}$ to predict the emotion distribution of a testing sample in $\mathcal{D}_{\text{test}}$.

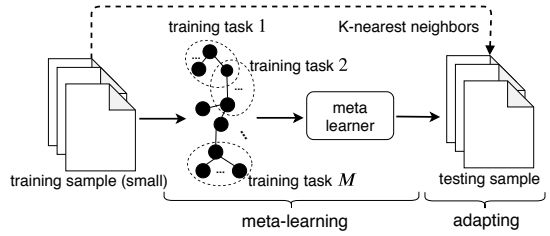


Figure 1: Illustration diagram of the proposed EDL method.

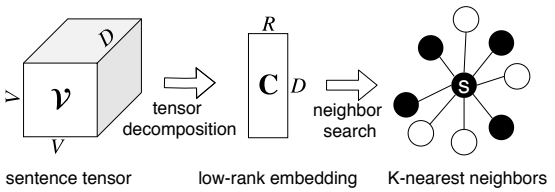


Figure 2: Tensor decomposition to find KNNs of each sentence.

## 4 Method

As shown in Figure 1, to augment the learning ability on a small sample, we first partition the training data by finding K-nearest neighbors (KNNs) of each sample based on semantic similarity, and treat each $K$ samples as a cluster. We then train a meta-learner that is optimized to adapt to new data with only a few training samples on the clusters. During testing, we fit the meta-learner with KNNs of a testing sample, where KNNs are found in the training data.

### 4.1 Meta-Learning Preliminary

A meta-learning algorithm consists of a set of training tasks $\mathcal{T}_{\text{train}} = \{T_i\}_{i=1}^{M}$, a set of testing tasks $\mathcal{T}_{\text{test}} = \{T_i\}_{i=1}^{N}$, and a model $f_\theta$ [1], where $M$ and $N$ are the numbers of the training and testing tasks separately, and $\theta$ is the parameters of $f$. Each task $T$ consists of a set of training samples called supporting set $S$ and a set of testing samples called querying set $Q$, namely, $T = \{S, Q\}$. The goal of meta-learning is to train a learner $f_\theta$ using $\mathcal{T}_{\text{train}}$ so that given a new testing task $T$ in $\mathcal{T}_{\text{test}}$, it can perform well on the query set $Q$ of $T$ by fine-tuning it with only a few samples in the support set $S$.

### 4.2 EDL via Meta-Learning

We propose a meta-learning framework for EDL, including task generation from a small sample, meta-learner training, and meta-learner adaption for predicting testing samples.

#### 4.2.1 Task Generation

For each sample in $\mathcal{D}_{\text{train}}$, we first find its KNNs by semantic similarity, and treat the $K$ samples as a training task. Therefore, samples in each task can be overlapped, and we use this way to increase the number of tasks. For each task, we randomly select $K/2$ samples as the support set, and use the remaining ones as the query set, i.e., the sizes of $S$ and $Q$ are both $K/2$. Therefore, for $M$ samples in $\mathcal{D}_{\text{train}}$, we can generate $M$ training tasks $\mathcal{T}_{\text{train}}$. The intuition is that semantically similar sentences are more likely to have similar emotion distribution patterns.

[1] To be consistent with the existing literature, we refer to the *model* as the *learner* in the rest of the paper.

**Algorithm 1:** Text EDL via Meta-Learning

**Data:** training sample $\mathcal{D}_{\text{train}}$ and testing sample $\mathcal{D}_{\text{test}}$
**Result:** learner $f_{\theta_s}$ for each sentence $s$ in $\mathcal{D}_{\text{test}}$

initialize hyper-parameters $\alpha, \beta, \gamma, H, R, K, niter, L$;

// Task Generation
calculate sentence embeddings $\mathbf{C}$ with formula (1), and for each sample in $\mathcal{D}_{\text{train}}$, find its KNNs to get $\mathcal{T}_{\text{train}}$;

// Meta-Learner Training
initialize the meta-learner $f$ with parameter $\theta$;
**for** *iter in* $1, 2, \ldots, niter$ **do**
  sample $L$ tasks from $\mathcal{T}_{train}$ randomly;
  **for** $l$ *in* $1, 2, \ldots, L$ **do**
    $\theta_l' \leftarrow \theta - \alpha \bigtriangledown_\theta \mathcal{L}_{T_l(S)}(f_\theta)$
  **end**
  $\theta \leftarrow \theta - \beta \bigtriangledown_\theta \sum_l \mathcal{L}_{T_l(Q)}(f_{\theta_l'})$;
**end**

// Adapting Meta-Learner
**for** *each sentence* $s$ *in* $\mathcal{D}_{test}$ **do**
  find KNNs $\mathcal{K}_s$ of $s$ in $\mathcal{D}_{\text{train}}$;
  $\theta_s \leftarrow \theta - \bigtriangledown_\theta \mathcal{L}_{\mathcal{K}_s}(f_\theta)$
**end**

**Low-rank embedding by tensor decomposition** Contextual patterns of words can be used to measure semantic similarity for emotion analysis (Staiano and Guerini, 2014; Mikolov et al., 2013). Traditional embedding approaches, e.g., doc2vec, usually require a large amount of data (Le and

Mikolov, 2014). We propose to use low-rank embeddings of sentences mined by tensor decomposition, which can obtain text embeddings regardless of the corpus size (Hosseinimotlagh and Papalexakis, 2018). As shown in Figure 2, considering $\mathcal{D} = \mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{test}}$, we first build a vocabulary for it, namely, $w_1, w_2, \ldots, w_V$, where $V$ is the number of words. For each sentence $s$ in $\mathcal{D}$, we count the word-word co-occurrence in a small window $H$, and build a binary matrix $\mathbf{V}_s \in [0,1]^{V \times V}$. In particular, $\mathbf{V}_s(i,j) = 1$ indicates word $w_i$ and $w_j$ co-occurs in $s$ within a small window $H$ at least once. In this way, we can capture the semantic patterns of a sentence in $\mathbf{V}_s$. In addition, this approach can also deal with negation issue to some extent, which is important for sentiment analysis (Reitan et al., 2015). Because negation sentences usually have negation words, they will have a different word-word co-occurrence pattern with that of normal sentences, which will result in different embeddings. Therefore, the tensor decomposition method may put sentences with and without negation words into different clusters. Afterwards, we stack all $\mathbf{V}_s$ as a three-dimensional tensor $\boldsymbol{\mathcal{V}} \in [0,1]^{V \times V \times D}$, where $D = M + N$. We adopt the CANDECOMP/PARAFAC tensor decomposition method (Sidiropoulos et al., 2017) to find an approximation $\hat{\boldsymbol{\mathcal{V}}}$ of $\boldsymbol{\mathcal{V}}$:

$$\hat{\boldsymbol{\mathcal{V}}} = \sum_{r=1}^{R} \mathbf{w}_r \otimes \mathbf{w}_r \otimes \mathbf{s}_r, \tag{1}$$

such that the Frobenius norm $\|\hat{\boldsymbol{\mathcal{V}}} - \boldsymbol{\mathcal{V}}\|_F$ is minimal, where $\mathbf{w}_r \in \mathbb{R}^V$, $\mathbf{s}_r \in \mathbb{R}^D$, $R$ is the rank, and $\otimes$ is the outer product, namely, $\mathbf{w}_r \otimes \mathbf{w}_r \otimes \mathbf{s}_r$ being a three-dimensional tensor, and $\mathbf{w}_r \otimes \mathbf{w}_r \otimes \mathbf{s}_r(i,j,k) = \mathbf{w}_r(i) \cdot \mathbf{w}_r(j) \cdot \mathbf{s}_r(k)$. We use both training and testing datasets for embedding, which follows the general practice of previous literature (Zhou et al., 2016; Zhang et al., 2018). In such a case, it is possible to infer a complete vocabulary of the corpus by making use of both the training and testing dataset. If some testing data are not available when the tensor is built, there exist tensor decomposition methods for dealing with streaming data scenarios (Gujral et al., 2018), namely, the testing data can be updated continuously over time. Our framework can easily utilize such advanced methods to generalize to sentences with out-of-vocabulary words.

With the tensor decomposition, we can find low-rank embeddings of sentences that capture the similarity of contextual patterns (Hosseinimotlagh and Papalexakis, 2018). In particular, $\mathbf{C} = [\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_R] \in \mathbb{R}^{D \times R}$, where the $s$-row of $\mathbf{C} = [\mathbf{c}_1^T, \mathbf{c}_2^T, \ldots, \mathbf{c}_D^T]^T$, $\mathbf{c}_s^T$, is the embedding vector of the sentence $s$. We measure the similarity of two sentences $i$, $j$ by Euclidean distance: $d(i,j) = \sqrt{\sum_{r=1}^{R} (\mathbf{c}_i(r) - \mathbf{c}_j(r))^2}$, where $\mathbf{c}(r)$ denotes the $r$-th element of $\mathbf{c}$.

#### 4.2.2 Meta-Learner Training

We can train a meta-learner on the generated $\mathcal{T}_{\text{train}}$.

**Learner.** We use a Convolutional Neural Network (CNN) model as the basic learner, which has shown good performance on text classification (Kim, 2014) and text emotion distribution learning (Zhang et al., 2018). The CNN learner has the same architecture with (Zhang et al., 2018). In particular, given the input sentence $s$, we first stack a matrix $\mathbf{X}$ with the word vector of each word (Mikolov et al., 2013). Then three convolution layers with kernel sizes $3, 4, 5$ are performed on $\mathbf{X}$ separately, and we then concatenate them together. A fully connected layer is then used, following with a soft-max operation to get the final emotion probability prediction. We use Kullback-Leibler (K-L) divergence $\mathcal{L}_{\text{K-L}}$ as the objective to measure the distance of predicted distribution $\hat{\mathbf{y}}$ and the true distribution $\hat{\mathbf{y}}$: $\mathcal{L}_{\text{K-L}} = \sum_{i=1}^{K} \frac{1}{K} \sum_{j=1}^{C} y_i^j \log \frac{y_i^j}{\hat{y}_i^j}$, where $K$ is the number of training samples. Similarly to (Zhang et al., 2018), we also optimize the classification accuracy of the dominant emotion by a cross-entropy (CE) objective, which is shown to improve EDL performance. In particular, $\mathcal{L}_{\text{CE}} = -\sum_{i=1}^{K} \frac{1}{K} \sum_{j=1}^{C} \mathbb{1}(y_i^j) \log \hat{y}_i^j$, where $\mathbb{1}(y^j) = 1$ if $y^j$ is the maximal value of $\mathbf{y}$, otherwise $\mathbb{1}(y^j) = 0$. The final objective is a weighted sum of $\mathcal{L}_{\text{KL}}$ and $\mathcal{L}_{\text{CE}}$: $\mathcal{L} = \gamma \mathcal{L}_{\text{K-L}} + (1-\gamma)\mathcal{L}_{\text{CE}}$, where $\gamma$ is the weight factor, $0 \le \gamma \le 1$.

**Training meta-learner.** To train the learner $f_\theta$, we optimize the parameters $\theta$ of the learner $f$ such that a small number of gradient steps on a new task will produce maximally effective behavior on that task, namely,

$$\min_{\theta} \sum_{T_i \sim \mathcal{T}_{\text{train}}} \mathcal{L}_{T_i(Q)}(f_{\theta_i'}), \tag{2}$$

where $T_i$ is a randomly sampled task in $\mathcal{T}_{\text{train}}$, $\theta_i'$ is the trained optimal parameters on task $T_i$. $T_i(Q)$ denotes that the loss is computed on the query set

$Q$. Equation (3) is optimized via stochastic gradient descent (SGD),

$$\theta \leftarrow \theta - \beta(\nabla_\theta \sum_{T_i \sim \mathcal{T}_{\text{train}}} \mathcal{L}_{T_i(Q)}(f_{\theta_i'})), \quad (3)$$

where $\beta$ is the meta-learning rate. Similarly, SGD is used to compute $\theta_i'$, $\theta_i' \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_{T_i(S)}(f_\theta)$, where $T_i(S)$ denotes that the loss is computed on $S$, and $\alpha$ is the learning rate.

#### 4.2.3 Adapting Meta-Learner

The meta-learner $f_\theta$ is trained to adapt to new data with only a few semantically similar training samples. Therefore, given a testing sample $s$, we first find its K-nearest neighbors $\mathcal{K}_s$ in the training data $\mathcal{D}_{\text{train}}$, and then adapt $f_\theta$ on $\mathcal{K}_s$ by SGD:

$$\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{K}_s}(f_\theta). \quad (4)$$

Here we use the learning rate $\alpha$ to be consistent with the meta-training procedure.

The overall algorithm is shown in Algorithm 1, where $\alpha$ is the learning rate, $\beta$ is the meta-learning rate, $\gamma$ is the weight value of the distribution and classification losses, $H$ is the window size for word-word co-occurrence counting, $R$ is the rank of tensor decomposition, $K$ is the number of nearest neighbors, $niter$ is the number of meta-training iterations, and $L$ is the number of tasks for each round of meta-training.

## 5 Experiment

To evaluate the proposed approach for learning emotion distribution from a small sample, we conduct intensive experiments on SemEval 2007 Task 14 (Strapparava and Mihalcea, 2007). To the best of our knowledge, this is the only publicly available English dataset with emotion distributions labeled by humans.

### 5.1 Experiment Setup

#### 5.1.1 Dataset

SemEval 2007 Task 14 contains 1250 sentences of news headlines with 6 emotion intensities (anger, disgust, fear, joy, sadness, and surprise) labeled by humans. Each intensity value is a non-negative value. We normalize the annotated scores to get emotion distribution labels using the same procedure with Zhang et al. (2018). In particular, given an original intensity tuple $(l_1, l_2, \ldots, l_6)$, we first calculate the sum of all values, $l = \sum_{k=1}^{6} l_k$, and then normalize it to get a distribution as $(l_1/l, l_2/l, \ldots, l_6/l)$. If $l = 0$, then the distribution will be $(1/6, 1/6, \ldots, 1/6)$. It is possible to add another intensity value $l_7$ denoting the overall level of emotion to the original label; for example, if $l_7$ is 1, there is no emotion in the text, and if $l_7$ is 0, there is strong emotions. Similar to $l_1 \sim l_6$, $l_7$ can be annotated manually. We can then normalize $(l_1, l_2, \ldots, l_7)$ to get a new distribution. In this way, we can better model the cases of a sentence expressing very little emotion overall. For simplicity and fair comparison, we only use $l_1 \sim l_6$ in this paper.

#### 5.1.2 Experiment Protocal

Because there is no publicly available small dataset for text emotion distribution evaluation, we follow existing practices (Hosseinimotlagh and Papalexakis, 2018) to simulate a small training set by randomly selecting $10\%$ of the samples and using the remaining ones for testing. We run 10-fold cross validation, and report the averaged results.

**Baselines.** We compare the proposed method (denoted as EDL-Meta) with several baseline methods, including the SOTA method EDL-CNN (Zhang et al., 2018), LDL methods, and document vectorization methods. EDL-CNN uses the basic learner of EDL-Meta, and can be seen as a special case of EDL-Meta when $K = 0$. Similar to Zhang et al. (2018), we also extract the penultimate layer of EDL-CNN as features, and fit several typical LDL methods: PT-Bayes, PT-SVM, AA-KNN, AA-BP, SA-LDSVR, SA-IIS, SA-BFGS, SA-CPNN for comparison. The basic principles of each method are briefly summarized in subsection 2.1. In addition, we apply three SOTA document vectorization methods – doc2vec (Le and Mikolov, 2014), InferSent (Conneau et al., 2017), and BERT (Devlin et al., 2019) – and use the extracted vectors to train a linear regressor to predict emotion distributions.

#### 5.1.3 Evaluation Metrics

Evaluating the performance of distribution learning is challenging, because we need to measure the prediction results of fine-grained intensity values. In (Geng and Ji, 2013), the authors propose six metrics, and suggest that each metric may reflect certain aspects of an algorithm. A good algorithm should perform well on most of them. The six metrics contain Euclidean$^{\downarrow}$, Sørensen$^{\downarrow}$, Squared$\chi^{2\downarrow}$,

| Method | Euclidean$^{\downarrow}$ | Sørensen$^{\downarrow}$ | Squared$\chi^{2\downarrow}$ | K-L$^{\downarrow}$ | Fidelity$^{\uparrow}$ | Intersection$^{\uparrow}$ |
|---|---|---|---|---|---|---|
| PT-Bayes | 0.4870 | 0.4889 | 0.6707 | 0.7574 | 0.7330 | 0.5111 |
| PT-SVM | 0.5748 | 0.5706 | 0.8578 | 1.1327 | 0.6639 | 0.4294 |
| AA-KNN | 0.4248 | 0.4054 | 0.5181 | 0.8325 | 0.7986 | 0.5946 |
| AA-BP | 0.4620 | 0.4468 | 0.5945 | 0.7410 | 0.7662 | 0.5532 |
| SA-LDSVR | 0.4336 | 0.4238 | 0.5301 | 0.5828 | 0.7927 | 0.5762 |
| SA-IIS | 0.4402 | 0.4305 | 0.5456 | 0.5983 | 0.7842 | 0.5695 |
| SA-BFGS | 0.8403 | 0.6909 | 1.1563 | 7.7697 | 0.5056 | 0.3091 |
| SA-CPCNN | 0.4779 | 0.4600 | 0.6108 | 0.7365 | 0.7660 | 0.5400 |
| BERT | 0.4498 | 0.4378 | 0.5620 | 0.6207 | 0.7770 | 0.5622 |
| InferSent | 0.4630 | 0.4518 | 0.5882 | 0.6488 | 0.7662 | 0.5482 |
| doc2vec | 0.4458 | 0.4334 | 0.5552 | 0.6133 | 0.7799 | 0.5666 |
| EDL-CNN | 0.4450 | 0.4345 | 0.5553 | 0.6105 | 0.7794 | 0.5655 |
| EDL-Meta | **0.3949** | **0.3747** | **0.4478** | **0.5148** | **0.8319** | **0.6253** |

Table 1: Evaluation results on SemEval 2007 Task 14.

| | | |
|---|---|---|
| Distance | Euclidean$^{\downarrow}$ | $\sqrt{\sum_{i=1}^{C}(p_i - q_i)^2}$ |
| | Sørensen$^{\downarrow}$ | $\frac{\sum_{i=1}^{C}\lvert p_i - q_i\rvert}{\sum_{i=1}^{C}(p_i + q_i)}$ |
| | Squared$\chi^{2\downarrow}$ | $\sum_{i=1}^{C}\frac{(p_i - q_i)^2}{p_i + q_i}$ |
| | K-L$^{\downarrow}$ | $\sum_{i=1}^{C} p_i \log \frac{p_i}{q_i}$ |
| Similarly | Fidelity$^{\uparrow}$ | $\sum_{i=1}^{C}\sqrt{p_i q_i}$ |
| | Intersection$^{\uparrow}$ | $\sum_{i=1}^{C}\min(p_i, q_i)$ |

Table 2: Metrics for evaluating the performance of emotion distribution learning.

K-L$^{\downarrow}$, Fidelity$^{\uparrow}$, Intersection$^{\uparrow}$, and can be classified into two categories: distance metrics and similarity metrics (Geng and Ji, 2013). Distance metrics measure the distance between the predicted distribution and the true one, the smaller the better. Similarity metrics measure the similarity between the predicted distribution and the true one, the bigger the better. Here, $^{\uparrow}$ means the bigger the better, $^{\downarrow}$ means the smaller the better. The formulas for calculating all metrics are summarized in Table 2, where $(q_1, q_2, \ldots, q_C)$ is the predicted distribution, $(p_1, p_2, \ldots, p_C)$ is the true distribution, $C$ is the number of emotions.

#### 5.1.4 Implementation

We use 300 dimension word vectors trained on Google News from (Mikolov et al., 2013) as initial input. Because SemEval 2007 Task 14 is mostly News headlines, the pre-trained word vectors are closer to it. We implement our algorithm by PyTorch [2]. Owing to EDL-CNN not being open-sourced, we also implement it by PyTorch. The implementation of doc2vec is adopted from gemsim [3]. BERT embedding (uncased base model, without fine-tuning) is adopted from an open-source implementation (Xiao, 2018). The implementations of other methods are downloaded from the original paper (Geng and Ji, 2013; Conneau et al., 2017). For EDL-CNN, we use the network architecture in the original paper, and set the learning rate 0.1, epoch number 25 (Kim, 2014). In the same experiment setting with (Zhang et al., 2018), namely, 90% training data and 10% testing data of SemEval 2007, we can obtain 0.344 Euclidean$^{\downarrow}$, 0.330 Sørensen$^{\downarrow}$, 0.344 Squared$\chi^{2\downarrow}$, 0.437 K-L$^{\downarrow}$, 0.853 Fidelity$^{\uparrow}$, 0.670 Intersection$^{\uparrow}$, which are slightly better than the results reported in (Zhang et al., 2018). For fair comparison, we use the default parameters of all other methods, which are reported to get the best performance. For EDL-Meta, we empirically set $\alpha = 0.01$, $\beta = 0.1$, $L = 5$, $H = 5$. $\gamma$ is set to be 0.7 according to (Zhang et al., 2018), and we do not further tune it. 1000 epoches are used to train the meta-learner for both datasets. To find optimal $K$ and $R$, we run a 5-fold cross validation grid search, and set $K = 20$, $R = 5$. The alternating least squares method is used to calculate the tensor decomposition, and we use the implementation in the Matlab tensor toolbox [4]. The k-d tree method is used to find KNNs of a sample efficiently. We run all experiments on a desktop with Intel(R) Core(TM) i9-7900X CPU 3.30GHz, 64GB RAM, Nvidia GeForce GTX 1080 Ti (×2).

### 5.2 Results

The comparison results of SemEval 2007 Task 14 are shown in Table 1. We can see that EDL-Meta shows better results than other methods on all metrics. In particular, EDL-Meta can usually outper-

[2] https://pytorch.org

[3] https://radimrehurek.com/gensim

[4] http://www.tensortoolbox.org

| Method | Euclidean$^{\downarrow}$ | Sørensen$^{\downarrow}$ | Squared$\chi^{2\downarrow}$ | K-L$^{\downarrow}$ | Fidelity$^{\uparrow}$ | Intersection$^{\uparrow}$ |
|---|---|---|---|---|---|---|
| doc2vec | 0.3987 | 0.3795 | 0.4563 | 0.5231 | 0.8279 | 0.6205 |
| InferSent | 0.3993 | 0.3799 | 0.4567 | 0.5228 | 0.8277 | 0.6201 |
| random | 0.3966 | 0.3763 | 0.4494 | 0.5179 | 0.8317 | 0.6237 |
| tensor | **0.3949** | **0.3747** | **0.4478** | **0.5148** | **0.8319** | **0.6253** |

Table 3: Comparison results of different embedding methods.

| Meta. | Adap. | Euclidean$^{\downarrow}$ | Sørensen$^{\downarrow}$ | Squared$\chi^{2\downarrow}$ | K-L$^{\downarrow}$ | Fidelity$^{\uparrow}$ | Intersection$^{\uparrow}$ |
|---|---|---|---|---|---|---|---|
| No | No | 0.4450 | 0.4345 | 0.5553 | 0.6105 | 0.7794 | 0.5655 |
| No | Yes | 0.4588 | 0.4489 | 0.5827 | 0.6403 | 0.7680 | 0.5511 |
| Yes | No | 0.3954 | 0.3752 | 0.4482 | 0.5154 | 0.8318 | 0.6248 |
| Yes | Yes | **0.3949** | **0.3747** | **0.4478** | **0.5148** | **0.8319** | **0.6253** |

Table 4: Comparison results of different combinations of training and adapting procedures.

form other methods by about $5\%$ on most metrics. Although EDL-CNN can perform well with more training samples (Zhang et al., 2018), on the small sample, its performance is not as stable as EDL-Meta. Even simple KNN can get slightly better results than EDL-CNN on most metrics. Likewise, other methods cannot get stable results on all metrics. This indicates that EDL-Meta is more suitable for small sample EDLs.

### 5.3 Discussion

#### 5.3.1 Low-rank Embedding

To investigate the effectiveness of the tensor decomposition method, we use doc2vec (Le and Mikolov, 2014) and InferSent (Conneau et al., 2017) to train sentence vectors on $\mathcal{D}$, and find KNNs of a sample by cosine similarity, denoted as doc2vec and InferSent, respectively. In addition, we also use randomly selected K-neighbors to train and test EDL-Meta, denoted as random. We denote our tensor decomposition method as tensor. The results are shown in Table 3. Compared to doc2vec, InferSent, and random, our method can get better results on all metrics. The doc2vec and InferSent embedding methods show a slightly lower performance, even worse than the random version. doc2vec usually assumes a large pool of training data (Le and Mikolov, 2014), while InferSent also does not work well for transferring to a task with a small training sample. The tensor decomposition method is more suitable for small sample learning.

#### 5.3.2 Meta-Training and Adapting

To investigate the effect of the proposed meta-training and adapting procedures, we conduct another experiment. In particular, with the same experiment setting, we replace the meta-training procedure (denoted as Meta.) with a normal batch training (batch size 50) and the adaption procedure (denoted as Adap.) with a normal evaluation procedure, respectively. We can then get four versions of EDL-Meta, namely, *without Meta. + without Adap.*, *without Meta. + with Adap.*, *with Meta. + without Adap.*, and *with Meta. + with Adap*. The first one is EDL-CNN in particular, and the last one is the normal EDL-Meta. In this way, we can further examine the importance of the proposed meta-training and adapting procedures. The results are shown in Table 4. We can see that with both the meta-training and adapting procedures, EDL-Meta performs more stably than other methods. In addition, the meta-training procedure plays a crucial role on boosting the performance, as the results of the third version are similar to the fourth version. Finally, without meta-training, only using

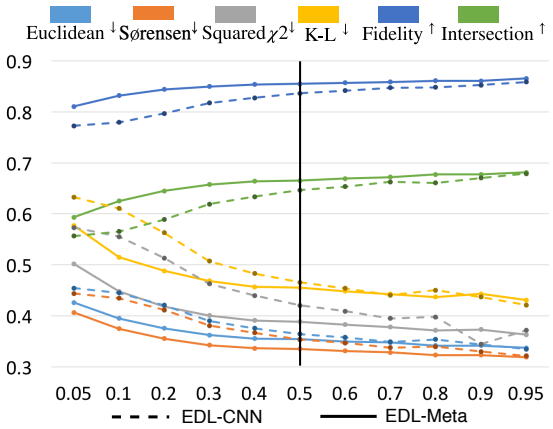


Figure 3: The performance of EDL-CNN and EDL-Meta on different training percentages. The horizontal axis denotes different training percentages, and the vertical axis denotes the performance on different metrics.

adapting is meaningless, as the performance of the second version decreases a lot.

#### 5.3.3 Label Percentage

In the same experiment setting, we further investigate the influence of different training/testing partition. We compare the performance of EDL-CNN and EDL-Meta under different training/testing partition. The results are shown in Figure 3. We can find that on small training sample, there is a clear gap between EDL-CNN and EDL-Meta. But with more training sample (>50%), EDL-CNN and EDL-Meta converge to similar results. Therefore, it is likely that the learning ability of EDL-Meta is limited by its basic learner. Further experiments on different learners deserve to investigate.

## 6 Conclusion

In this paper, we propose an efficient meta-learning approach to learn text emotion distribution from a small sample. In addition, to find the K-nearest semantically similar neighbors, we propose to learn sentence embedding by tensor decomposition. Experiments on SemEval 2007 Task 14 shows that the proposed approach outperforms existing EDL methods on small sample emotion distribution learning. We further investigate the low-rank embedding method, the meta-training procedure, and the adaption procedure. We find that leveraging nearest semantically similar neighbors is an effective way for small sample EDLs. We also find that the meta-training procedure can be a good alternative to the normal batch training procedure, especially for small sample EDLs.

## Acknowledgments

The authors would like to thank Lanqing Xue, Zhenhui Peng, and Prof. Evangelos E. Papalexakis for insightful discussions, as well as all anonymous reviewers for constructive comments. This work is supported by the Research Grants Council of the Hong Kong Special Administrative Region, China under Grant No.: C6030-18G.

## References

Muhammad Abdul-Mageed and Lyle Ungar. 2017. EmoNet: Fine-grained emotion detection with gated recurrent neural networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 718–728, Vancouver, Canada. Association for Computational Linguistics.

Yoshua Bengio, Samy Bengio, and Jocelyn Cloutier. 1991. Learning a synaptic learning rule. In *International Joint Conference on Neural Networks (IJCNN)*, page 969.

Lea Canales and Patricio Martínez-Barco. 2014. Emotion detection from text: A survey. In *Proceedings of the Workshop on Natural Language Processing in the 5th Information Systems Research Working Days (JISIC)*, pages 37–43, Quito, Ecuador. Association for Computational Linguistics.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Andrea Esuli and Fabrizio Sebastiani. 2006. SENTIWORDNET: A publicly available lexical resource for opinion mining. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy. European Language Resources Association (ELRA).

Ahmed Fadhil and Silvia Gabrielli. 2017. Addressing challenges in promoting healthy lifestyles: The AI-chatbot approach. In *Proceedings of the 11th EAI International Conference on Pervasive Computing Technologies for Healthcare*, pages 261–265. ACM.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135, International Convention Centre, Sydney, Australia. PMLR.

Chelsea Finn, Aravind Rajeswaran, Sham Kakade, and Sergey Levine. 2019. Online meta-learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1920–1930, Long Beach, California, USA. PMLR.

Xin Geng and Rongzi Ji. 2013. Label distribution learning. *Proceedings of the 13th IEEE International Conference on Data Mining Workshops*, pages 377–383.

Jiatao Gu, Yong Wang, Yun Chen, Victor O. K. Li, and Kyunghyun Cho. 2018. Meta-learning for low-resource neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods*

*in Natural Language Processing*, pages 3622–3631, Brussels, Belgium. Association for Computational Linguistics.

Ekta Gujral, Ravdeep Pasricha, and Evangelos E. Papalexakis. 2018. Sambaten: Sampling-based batch incremental tensor decomposition. In *Proceedings of the 2018 SIAM International Conference on Data Mining*, pages 387–395.

Seyedmehdi Hosseinimotlagh and Evangelos E Papalexakis. 2018. Unsupervised content-based identification of fake news articles with tensor decomposition ensembles. In *WSDM 2018 Workshop on Misinformation and Misbehavior Mining on the Web (MIS2)*.

Kyle Hsu, Sergey Levine, and Chelsea Finn. 2019. Unsupervised learning via meta-learning. In *International Conference on Learning Representations*.

Po-Sen Huang, Chenglong Wang, Rishabh Singh, Wen-tau Yih, and Xiaodong He. 2018. Natural language to structured query generation via meta-learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 732–738, New Orleans, Louisiana. Association for Computational Linguistics.

Mark Johnson, Peter Anderson, Mark Dras, and Mark Steedman. 2018. Predicting accuracy on large datasets from smaller pilot data. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 450–455, Melbourne, Australia. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3294–3302. Curran Associates, Inc.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. 2015. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1188–1196, Bejing, China. PMLR.

Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. 2019. Meta-learning with differentiable convex optimization. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Andrius Mudinas, Dell Zhang, and Mark Levene. 2012. Combining lexicon and learning based approaches for concept-level sentiment analysis. In *Proceedings of the First International Workshop on Issues of Sentiment Discovery and Opinion Mining*, WISDOM '12, pages 5:1–5:8, New York, NY, USA. ACM.

Yanghui Rao, Haoran Xie, Jun Li, Fengmei Jin, Fu Lee Wang, and Qing Li. 2016. Social emotion classification of short text via topic-level maximum entropy model. In *Information & Management*, volume 53, pages 978 – 986.

Johan Reitan, Jørgen Faret, Björn Gambäck, and Lars Bungum. 2015. Negation scope detection for twitter sentiment analysis. In *Proceedings of the 6th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 99–108, Lisboa, Portugal. Association for Computational Linguistics.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. 2016. Meta-learning with memory-augmented neural networks. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 1842–1850. JMLR.org.

Jun Shu, Zongben Xu, and Deyu Meng. 2018. Small sample learning in big data era. *arXiv:1808.04572*.

Nicholas D. Sidiropoulos, Lieven De Lathauwer, Xiao Fu, Kejun Huang, Evangelos E. Papalexakis, and Christos Faloutsos. 2017. Tensor decomposition for signal processing and machine learning. *IEEE Trans. Signal Process.*, 65(13):3551–3582.

Jacopo Staiano and Marco Guerini. 2014. Depeche mood: a lexicon for emotion analysis from crowd annotated news. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 427–433, Baltimore, Maryland. Association for Computational Linguistics.

Carlo Strapparava and Rada Mihalcea. 2007. Semeval-2007 task 14: Affective text. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, SemEval '07, pages 70–74.

Carlo Strapparava and Alessandro Valitutti. 2004. WordNet affect: an affective extension of WordNet. In *Proceedings of the Fourth International*

*Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal. European Language Resources Association (ELRA).

Sebastian Thrun and Lorien Pratt, editors. 1998. *Learning to Learn*. Kluwer Academic Publishers, Norwell, MA, USA.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Ricardo Vilalta and Youssef Drissi. 2002. A perspective view and survey of meta-learning. *Artificial Intelligence Review*, 18(2):77–95.

Ekaterina P. Volkova, Betty J. Mohler, Detmar Meurers, Dale Gerdemann, and Heinrich H. Bülthoff. 2010. Emotional perception of fairy tales: Achieving agreement in emotion annotation of text. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, CAAGET '10, pages 98–106, Stroudsburg, PA, USA. Association for Computational Linguistics.

Han Xiao. 2018. bert-as-service. https://github.com/hanxiao/bert-as-service.

Ali Yadollahi, Ameneh Gholipour Shahraki, and Osmar R. Zaiane. 2017. Current state of text sentiment analysis from opinion to emotion mining. *ACM Comput. Surv.*, 50(2):25:1–25:33.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Jianfei Yu, Luis Marujo, Jing Jiang, Pradeep Karuturi, and William Brendel. 2018. Improving multi-label emotion classification via sentiment classification with dual attention transfer network. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1097–1102, Brussels, Belgium. Association for Computational Linguistics.

Yuxiang Zhang, Jiamei Fu, Dongyu She, Ying Zhang, Senzhang Wang, and Jufeng Yang. 2018. Text emotion distribution learning via multi-task convolutional neural network. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, pages 4595–4601. International Joint Conferences on Artificial Intelligence Organization.

Deyu Zhou, Xuan Zhang, Yin Zhou, Quan Zhao, and Xin Geng. 2016. Emotion distribution learning from texts. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 638–647, Austin, Texas. Association for Computational Linguistics.

# Judge the Judges: A Large-Scale Evaluation Study of Neural Language Models for Online Review Generation

**Cristina Gârbacea**[1], **Samuel Carton**[2], **Shiyan Yan**[2], **Qiaozhu Mei**[1,2]
[1]Department of EECS, University of Michigan, Ann Arbor, MI, USA
[2]School of Information, University of Michigan, Ann Arbor, MI, USA
{garbacea, scarton, shiyansi, qmei}@umich.edu

## Abstract

We conduct a large-scale, systematic study to evaluate the existing evaluation methods for natural language generation in the context of generating online product reviews. We compare human-based evaluators with a variety of automated evaluation procedures, including discriminative evaluators that measure how well machine-generated text can be distinguished from human-written text, as well as word overlap metrics that assess how similar the generated text compares to human-written references. We determine to what extent these different evaluators agree on the ranking of a dozen of state-of-the-art generators for online product reviews. We find that human evaluators do not correlate well with discriminative evaluators, leaving a bigger question of whether adversarial accuracy is the correct objective for natural language generation. In general, distinguishing machine-generated text is challenging even for human evaluators, and human decisions correlate better with lexical overlaps. We find lexical diversity an intriguing metric that is indicative of the assessments of different evaluators. A post-experiment survey of participants provides insights into how to evaluate and improve the quality of natural language generation systems [1].

## 1 Introduction

Recent developments in neural language models (Mikolov and Zweig, 2012), (Reiter and Belz, 2009), (Mikolov et al., 2011b), (Mikolov et al., 2011a) have inspired the use of neural network based architectures for the task of natural language generation (NLG). Despite fast development of algorithms, there is an urgency to fill the huge gap in evaluating NLG systems. On one hand, a rigorous, efficient, and reproducible evaluation procedure is critical for the development of any machine learning technology and for correct interpretation of the state-of-the-art. On the other hand, evaluating the quality of language generation is inherently difficult due to the special properties of text, such as *subjectivity* and *non-compositionality*. Indeed, *"there is no agreed objective criterion for comparing the goodness of texts"* (Dale and Mellish, 1998), and there lacks a clear model of text quality (Hardcastle and Scott, 2008).

Conventionally, most NLG systems have been evaluated in a rather informal manner. (Reiter and Belz, 2009) divide existing evaluation methods commonly employed in text generation into three categories: *i)* evaluations based on task performance, *ii)* human judgments and ratings, where human subjects are recruited to rate different dimensions of the generated texts, and *iii)* evaluations based on comparison to a reference corpus using automated metrics. *Task based evaluation* considers that the value of a piece of functional text lies in how well it serves the user to fulfill a specific application. It can be expensive, time-consuming, and often dependent on the good will of participants in the study. Besides that, it is hard to toss out the general quality of text generation from the special context (and confounds) of the task, or to generalize the evaluation conclusions across tasks. *Human annotation* is able to assess the quality of text more directly than task based evaluation. However, rigorously evaluating NLG systems with real users can be expensive and time consuming, and it does not scale well (Reiter et al., 2001). Human assessments also need to be consistent and repeatable for a meaningful evaluation (Lopez, 2012). Alternative strategies which are more effective in terms of cost and time are used more frequently.

*Automated evaluation* compares texts generated by the candidate algorithms to human-written texts.

[1] The experimental setup, data, and annotations are publicly available at: https://github.com/Crista23/JudgeTheJudges



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3968–3981,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

Word overlap metrics and more recent automated adversarial evaluators are widely employed in NLG as they are cheap, quick, repeatable, and do not require human subjects when a reference corpus is already available. In addition, they allow developers to make rapid changes to their systems and automatically tune parameters without human intervention. Despite the benefits, however, the use of automated metrics in the field of NLG is controversial (Reiter and Belz, 2009), and their results are often criticized as not meaningful for a number of reasons. First, these automatic evaluations rely on a high-quality corpus of references, which is not often available. Second, comparisons with a reference corpus do not assess the usefulness and the impact of the generated text on the readers as in human-based evaluations. Third, creating human written reference texts specifically for the purpose of evaluation could still be expensive, especially if these reference texts need to be created by skilled domain experts. Finally and most importantly, using automated evaluation metrics is sensible only if they correlate with results of human-based evaluations and if they are accurate predictors of text quality, which is never formally verified at scale.

We present a large-scale, systematic experiment that evaluates the *evaluators* for NLG. We compare three types of evaluators including human evaluators, automated adversarial evaluators trained to distinguish human-written from machine-generated product reviews, and word overlap metrics (such as BLEU and ROUGE) in a particular scenario, generating online product reviews. The preferences of different evaluators on a dozen representative deep-learning based NLG algorithms are compared with human assessments of the quality of the generated reviews. Our findings reveal significant differences among the evaluators and shed light on the potential factors that contribute to these differences. The analysis of a post experiment survey also provides important implications on how to guide the development of new NLG algorithms.

## 2 Related Work

### 2.1 Deep Learning Based NLG

Recently, a decent number of deep learning based models have been proposed for text generation. Recurrent Neural Networks (RNNs) and their variants, such as Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) models, Google LM (Jozefowicz et al., 2016), and Scheduled Sampling (SS) (Bengio et al., 2015) are widely used for generating textual data.

Generative Adversarial Networks (Goodfellow et al., 2014), or GANs, train generative models through an adversarial process. Generating text with GANs is challenging due to the discrete nature of text data. SeqGAN (Yu et al., 2017) is one of the earliest GAN-based model for sequence generation, which treats the procedure as a sequential decision making process. RankGAN (Lin et al., 2017) proposes a framework that addresses the quality of a set of generated sequences collectively. Many GAN-based models (Yu et al., 2017), (Lin et al., 2017), (Rajeswar et al., 2017), (Che et al., 2017), (Li et al., 2017), (Zhang et al., 2017) are only capable of generating short texts. LeakGAN (Guo et al., 2018) is proposed for generating longer texts.

Deep learning architectures other than LSTM or GAN have also been proposed for text generation. (Tang et al., 2016) study NLG given particular contexts or situations and proposes two approaches based on the encoder-decoder framework. (Dong et al., 2017) address the same task and employ an additional soft attention mechanism. Pre-training enables better generalization in deep neural networks (Erhan et al., 2010), especially when combined with supervised discriminative fine-tuning to learn universal robust representations (Radford et al., 2018), (Devlin et al., 2018), (Radford et al., 2019). (Guu et al., 2018) use a prototype-then-edit generative language model for sentences.

### 2.2 Automated Evaluation Metrics

The variety of NLG models are also evaluated with various approaches. Arguably, the most natural way to evaluate the quality of a generator is to involve humans as judges, either through some type of Turing test (Machinery, 1950) to distinguish generated text from human input texts, or to directly compare the texts generated by different generators (Mellish and Dale, 1998). Such approaches are hard to scale and have to be redesigned whenever a new generator is included. Practically, it is critical to find automated metrics to evaluate the quality of a generator independent of human judges or an exhaustive set of competing generators.

**Perplexity** (Jelinek et al., 1977) is commonly used to evaluate the quality of a language model, which has also been employed to evaluate generators (Yarats and Lewis, 2018), (Ficler and Goldberg, 2017), (Gerz et al., 2018) even though it is com-

monly criticized for not being a direct measure of the quality of generated text (Fedus et al., 2018). Perplexity is a model dependent metric, and "how likely a sentence is generated by a given model" is not comparable across different models. Therefore we do not include perplexity in this study.

**Discriminative Evaluation** is an alternative way to evaluate a generator, which measures how likely its generated text can fool a classifier that aims to distinguish the generated text from human-written texts. In a way, this is an automated approximation of the Turing test, where machine judges are used to replace human judges. Discriminative machine judges can be trained either using a data set with explicit labels (Ott et al., 2011), or using a mixture of text written by real humans and those generated by the model being evaluated. The latter is usually referred to as *adversarial evaluation*. (Bowman et al., 2016) proposes one of the earliest studies that uses adversarial error to assess the quality of generated sentences. Notably, maximizing the adversarial error is consistent to the objective of the generator in generative adversarial networks. (Kannan and Vinyals, 2017) propose an adversarial loss to discriminate a dialogue model's output from human output. The discriminator prefers longer output and rarer language instead of the common responses generated. There however lacks evidence that a model that obtains a lower adversarial loss is better according to human evaluations.

Automatic dialogue evaluation is formulated as a learning problem in (Lowe et al., 2017), who train an RNN to predict the scores a human would assign to dialogue responses. RNN predictions correlate with human judgments at the utterance and system level, however each response is evaluated in a very specific context and the system requires substantial human judgments for training. (Li et al., 2017) employ a discriminator (analogous to the human evaluator in the Turing test) both in training and testing and define adversarial success. Other work finds the performance of a discriminative agent (e.g., attention-based bidirectional LSTM binary classifier) is comparable with human judges at distinguishing between real and fake dialogue excerpts (Bruni and Fernández, 2017). However, results show there is limited consensus among humans on what is considered as coherent dialogue passages.

**Word Overlap Metrics**, such as BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), and METEOR (Banerjee and Lavie, 2005), are commonly used to measure the similarity between the generated text and human written references. (Liu et al., 2016) find that word overlap metrics present weak or no correlation with human judgments in non-task oriented dialogue systems and thus should be used with caution or in combination with user studies. In contrary, it is reported in (Sharma et al., 2017) that text overlap metrics are indicative of human judgments in task-oriented dialogue settings, when used on datasets which contain multiple ground truth references. (Dai et al., 2017) find text overlap metrics too restrictive as they focus on fidelity of wording instead of fidelity of semantics. (Callison-Burch et al., 2006) consider an increase in BLEU insufficient for an actual improvement in the quality of a system and posit in favor of human evaluation.

BLEU and its variants (e.g., Self-BLEU) are used to evaluate GAN models (Caccia et al., 2018; Zhu et al., 2018). (Shi et al., 2018) compare frameworks for text generation including MLE, SeqGAN, LeakGAN and Inverse Reinforcement Learning using a simulated Turing test. A benchmarking experiment with GAN models is conducted in (Lu et al., 2018); results show LeakGAN presents the highest BLEU scores on the test data. Similarly, BLEU and METEOR present highest correlations with human judgements (Callison-Burch et al., 2008), (Graham and Baldwin, 2014). However, evaluation metrics are not robust across conditions, and no single metric consistently outperforms other metrics across all correlation levels (Przybocki et al., 2009).

Conventional neural language models trained with maximum likelihood can be on par or better than GANs (Caccia et al., 2018), (Semeniuta et al., 2018), (Tevet et al., 2018). However, log-likelihood is often computationally intractable (Theis et al., 2016). Models with good likelihood can produce bad samples, and vice-versa (Goodfellow, 2016). Generative models should be evaluated with regards to the task they are intended for over the full quality-diversity spectrum (Cífka et al., 2018), (Hashimoto et al., 2019), (Montahaei et al., 2019).

While many generators are proposed and evaluated with various metrics, no existing work has systematically evaluated the different evaluators at scale, especially in the context of online review generation. Our work fills in this gap.

## 3 Experiment Design

We design a large-scale experiment to systematically analyze the procedures and metrics used for

evaluating NLG models. To test the different *evaluators*, the experiment carefully chooses a particular application context and a variety of natural language generators in this context. Ideally, a sound automated evaluator should be able to distinguish good generators from suboptimal ones. Its preferences (on ordering the generators) should be consistent to humans in the exact application context.

### 3.1 Experiment Context and Procedure

We design the experiment in the context of generating online product reviews. There are several reasons why review generation is a desirable task for the experiment: 1) online product reviews are widely available, and it is easy to collect a large number of examples for training/ testing the generators; 2) Internet users are used to reading online reviews, and it is easy to recruit capable human judges to assess the quality of reviews; and 3) comparing to tasks like image caption generation or dialogue systems, review generation has minimal dependency on the conversation context or on non-textual data, which reduces possible confounds.

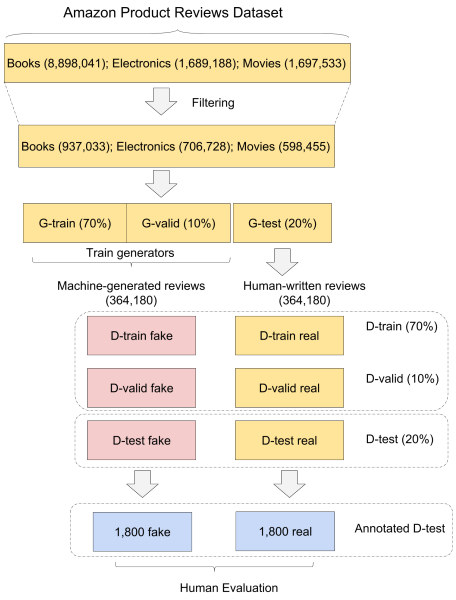


Figure 1: Overview of the Experiment Procedure.

The general experiment procedure is presented in Figure 1. We start from the publicly available Amazon Product Reviews dataset [2] and select three most popular domains: *books*, *electronics*, and *movies*. After filtering rare products, inactive users, and overly long reviews, the dataset is randomly split into three parts, to train, to validate, and to test the candidate review generators (denoted as *G-train*, *G-valid*, and *G-test*). Every generative model is trained and validated using the same datasets, and then charged to generate a number of product reviews (details are included in the next section). These generated reviews, mixed with the real reviews in *G-test*, are randomly split into three new subsets for training, validating, and testing candidate (discriminative) evaluators, denoted as *D-train*, *D-valid*, and *D-test*. Finally, a random sample of reviews from *D-test* are sent for human evaluation.

[2] http://jmcauley.ucsd.edu/data/amazon/

### 3.2 Review Generators

Although our goal is to evaluate the evaluators, it is critical to include a wide range of text generators with various degrees of quality. A good evaluator should be able to distinguish the high-quality generators from the low-quality ones. We select a diverse set of generative models from recent literature. The goal of this study is *not* to name the best generative model, and it is unfeasible to include all existing models. Our criteria are: (1) the models are published before 2018, when our experiment is conducted; (2) the models represent different learning strategies and quality levels; (3) the models have publicly available implementations, for reproducibility purposes. In Table 1 we list the candidate generators. It is not an exhaustive list of what are currently available. For implementation details of these models please see Appendix A.1.

Table 1: Candidate models for review generation.

| Generative Model | Adversarial Framework |
|---|---|
| Word LSTM temp 1.0 (Hochreiter and Schmidhuber, 1997) | No |
| Word LSTM temp 0.7 (Hochreiter and Schmidhuber, 1997) | No |
| Word LSTM temp 0.5 (Hochreiter and Schmidhuber, 1997) | No |
| Scheduled Sampling (Bengio et al., 2015) | No |
| Google LM (Jozefowicz et al., 2016) | No |
| Attention Attribute to Sequence* (Dong et al., 2017) | No |
| Contexts to Sequences* (Tang et al., 2016) | No |
| Gated Contexts to Sequences* (Tang et al., 2016) | No |
| MLE SeqGAN (Yu et al., 2017) | Yes |
| SeqGAN (Yu et al., 2017) | Yes |
| RankGAN (Lin et al., 2017) | Yes |
| LeakGAN (Guo et al., 2018) | Yes |

* indicates that review generation using these models are conditional on context information such as product ids; other models are context independent.

Every generator (except Google LM) is trained and validated on *G-train* and *G-valid* datasets, and used to generate the same number of machine-generated (a.k.a., fake) reviews (see Table 2). We follow the best practice in literature to train these models, although it is possible that the performance of models might not be optimal due to various con-

straints. This will not affect the validity of the experiment as our goal is to evaluate the **evaluators** instead of the individual generators. Google LM was not trained on reviews, but it provides a sanity check for the experiment - a reasonable evaluator should not rank it higher than those trained for generating reviews.

Table 2: Number of generated reviews by each model.

| Generative Model | Total | D-Train | D-Valid | D-Test |
|---|---|---|---|---|
| ∀ model in Table 1 except Google LM | 32,500 | 22,750 | 3,250 | 6,500 |
| Google LM | 6,680 | 4,676 | 668 | 1,336 |

### 3.3 Evaluators

We include a comprehensive set of evaluators for the quality of the aforementioned generators: *i)* human evaluators, *ii)* discriminative evaluators, and *iii)* text overlap evaluators. The evaluators are the main subjects of the experiment.

#### 3.3.1 Human evaluators

We conduct a careful power analysis (Christensen, 2007), which suggests that at least 111 examples per generative model should be human annotated to infer that the machine-generated reviews are comparable in quality to human-written reviews, at a minimal statistically significance level of 0.05. Per this calculation, we sample 150 examples for each of the 12 generators for human evaluation. This totals 1,800 machine-generated reviews, to which we add 1,800 human-written reviews, or a total of 3,600 product reviews sent for human annotation. We markup out-of-vocabulary words in *both* human-written and machine-generated reviews to control for confounds of using certain rare words. There is no significant difference in proportion of the markup token between the two classes (2.5%-real vs. 2.2%-fake). We recruit 900 human annotators through the Amazon Mechanical Turk (AMT) platform. Each annotator is presented 20 reviews, a mixture of 10 real (i.e., human written) and 10 fake (i.e., machine generated), and they are charged to label each review as real or fake based on their own judgment. Clear instructions are presented to the workers that markup tokens are present in both classes and cannot be used to decide whether a review is real or fake. Each page is annotated by 5 distinct human evaluators. The 5 judgments on every review are used to assemble two distinct **human evaluators**: *H1* - **individual votes**, treating all human annotations independently, and *H2* - **majority votes** of the 5 human judgments. For every *annotated* review, the human evaluator ($H1$ or $H2$) makes a call which can be either right or wrong with regard to the ground truth. A generator is high quality if the human evaluator achieves low accuracy identifying the reviews as fake.

#### 3.3.2 Discriminative evaluators

The inclusion of multiple generators provides the opportunity of creating **meta-adversarial evaluators**, trained using a *pool* of generated reviews by *many* generators, mixed with a larger number of "real" reviews (*D-train* and *D-valid* datasets). Such a "pooling" strategy is similar to the standard practice used by the TREC conferences to evaluate different information retrieval systems (Harman and Voorhees, 2006). Comparing to individual adversarial evaluators, a meta-evaluator is supposed to be more robust and fair, and it can be applied to evaluate new generators without being retrained. In our experiment, we find that the meta-adversarial evaluators rank the generators in similar orders to the best individual adversarial evaluators.

We employ a total of 7 meta-adversarial evaluators: 3 deep, among which one using LSTM (Hochreiter and Schmidhuber, 1997), one using Convolutional Neural Network (CNN) (LeCun et al., 1998), and one using a combination of LSTM and CNN architectures; 4 shallow, based on Naive Bayes (NB) (Rish, 2001), Random Forest (RF) (Liaw et al., 2002), Support Vector Machines (SVM) (Cortes and Vapnik, 1995), and XGBoost (Chen and Guestrin, 2016), with unigrams, bigrams, and trigrams as features and on balanced training sets. We find the best hyper-parameters using random search and prevent the models from overfitting by using early stopping. For every review in *D-test* (either annotated or not), a meta-adversarial evaluator makes a judgment call. A generator is considered high quality if the meta-adversarial evaluator makes more mistakes on reviews it generated.

#### 3.3.3 Word-overlap evaluators

We include a set of 4 text-overlap metrics used for NLG evaluation: BLEU and METEOR (specific to machine translation), ROUGE (used in text summarization), and CIDEr (Vedantam et al., 2015) (used in image description evaluation). These metrics rely on matching $n$-grams in the target text (i.e., generated reviews) to the "references" (i.e., human-written reviews). The higher the overlap (similarity), the higher the quality of generated text. For

every generated review in *D-test Fake*, we assemble the set of references by retrieving the top-10 most similar human-written reviews in *D-test Real* using a simple vector space model. We compute 600-dimensional vector representation of reviews using Sent2Vec (Pagliardini et al., 2018), pretrained on English Wikipedia, and retrieve the top-k nearest neighbors for each review based on cosine similarity of the embedding vectors. The rationale of using nearest neighbors of each generated review as references is that to appear "real", a generated review just need to be similar to *some* real reviews instead of *all*. A generator is considered high quality if its generated reviews obtain a high average score by a text overlap evaluator. In total, we analyze and compare 13 candidate evaluators (2 human evaluators, 7 discriminative evaluators, and 4 text-overlap metrics), based on the *D-test* dataset.

## 4 Results

First, we are interested in the accuracy of individual evaluators - how well they can distinguish "fake" (machine-generated) reviews from "real" (human-written) reviews. Second, we are interested in how an evaluator assesses the quality of the 12 generators instead of individual reviews. The absolute scores an evaluator gives to the generators are not as informative as how it ranks them: a good evaluator should be able to rank good generators above bad generators. Last but not least, we are interested in how the rankings by different evaluators correlate with each other. Intuitively, an automated evaluator that ranks the generators in similar orders as the human evaluators is more reasonable and can potentially be used as the surrogate of humans.

### 4.1 Results of Individual Evaluators

#### 4.1.1 Human evaluators

Every review is annotated by 5 human judges as either "fake" or "real." The inter-annotator agreement (Fleiss-Kappa score (Fleiss et al., 2013)) is $k = 0.2748$. This suggests that *distinguishing machine-generated reviews from real in general is a hard task even for humans*; there is limited consensus on what counts as a realistic review. The low agreement also implies that any automated evaluator that mimics human judges is not necessarily the most "accurate."

In Figure 2 we present the accuracy of two human evaluators on individual annotated reviews, based on either all 5 annotations or their majority votes for each review. Comparing to the ground-truth (of whether a review is machine-generated or collected from Amazon), individual human decisions are 66.61% accurate, while their majority votes can reach 72.63%. Neither of them is close to perfect. *We observe that human evaluators generally do better at correctly labelling human-written reviews as real (true positive rate of 78.96% for $H1$ and 88.31% for $H2$), and they are confused by machine-generated reviews in close to half of the cases (true negative rate of 54.26% for $H1$ and 56.95% for $H2$).* This trend reassures previous observations (Tang et al., 2016).

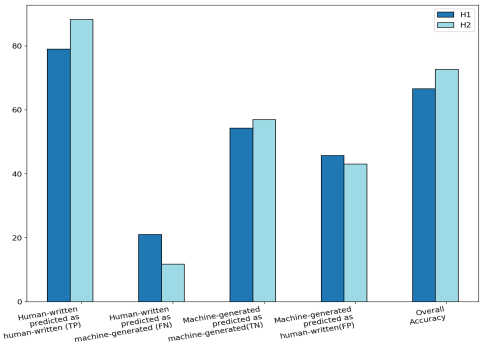


Figure 2: Accuracy of human evaluators on individual reviews: *H1* - individual votes; *H2* - majority votes.

We then look at how the human evaluators rank the 12 generators, according to the accuracy of human evaluators on all (fake) reviews generated by each of the generators. The lower the accuracy, the more likely the human evaluator is confused by the generated reviews, and thus the better the generator. We observe a substantial variance in the accuracy of both human evaluators on different generators, which suggests that human evaluators are able to distinguish between generators. The generator ranked the highest by both human evaluators is *Gated Contexts to Sequences*. Google LM is ranked on the lower side, which makes sense as the model is not trained to generate reviews. Interestingly, humans tend not to be fooled by reviews generated by the GAN-based models (MLE SeqGAN, SeqGAN, RankGAN and LeakGAN), even though their objective is to confuse fake from real. GAN-generated reviews tend to be easily distinguishable from the real reviews by human judges.

#### 4.1.2 Discriminative evaluators

We then analyze the 7 meta-adversarial evaluators. Different from human evaluators that are applied

to the 3,600 annotated reviews, the discriminative evaluators are applied to *all* reviews in *D-test*.

**Meta-adversarial Evaluators.** On individual reviews, the three deep learning based and the one SVM based evaluators achieve higher accuracy than the two human evaluators, indicating that adversarial evaluators can distinguish a single machine-generated review from human-written better than humans (Figure 3 and Table 4 in Appendix A.3.2). Their true positive rates and true negative rates are more balanced than human evaluators. Meta-discriminators commonly rank GAN-based generators the highest. This makes sense as the objective of GAN is consistent to the (reversed) evaluator accuracy. Interestingly, by simply setting the temperature parameter of Word LSTM to 1.0, it achieves comparable performance to the GANs.

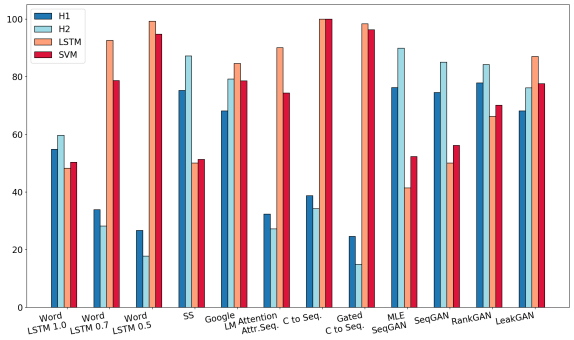


Figure 3: Accuracy of human (H1, H2) and meta-adversarial evaluators (LSTM, SVM) on reviews generated by individual generators. **The lower the accuracy, the better the generator.**

### 4.1.3 Word-Overlap Evaluators

The generators are ranked based on the average scores of their generated reviews. In Figure 4 we present the average scores of the 12 generators by each evaluator. Different word-overlap evaluators also tend to rank the generators in similar orders. Interestingly, the top-ranked generator according to three evaluators is *Contexts to Sequences*, while CIDEr scores highest the *Gated Contexts to Sequences* model. GAN-based generators are generally ranked low; please also see Appendix A.3.3.

## 4.2 Comparing Evaluators

To what degree do the evaluators agree on the ranking of generators? We are more interested in how the automated evaluators compare to the human evaluators, and whether there is any suitable automated surrogate for human judges at all. To do

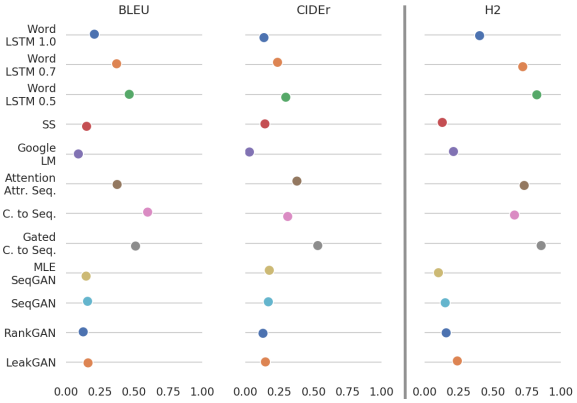


Figure 4: Text-Overlap Evaluators (BLEU and CIDEr) scores for individual generators. **The higher the better.** The rankings are overall similar, as GAN-based generators are ranked low.

this, we compute the correlations between $H1$, $H2$ and each discriminative evaluator and correlations between $H1$, $H2$ and the text-overlap evaluators, based on either their decisions on individual reviews, their scores of the generators (by Pearson's coefficient (Fieller et al., 1957)), and their rankings of the generators (by Spearman's $\rho$ (Spearman, 1904) and Kendall's $\tau$ (Daniel et al., 1978)). Patterns of the three correlation metrics are similar; please see Figure 5 and Table 5 in Appendix A.3.4.

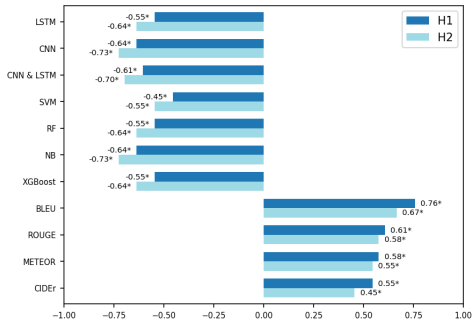


Figure 5: Kendall $\tau$-b between human and automated evaluators. Human's ranking is positively correlated to text-overlap evaluators and negatively correlated to adversarial evaluators ($^*$ is $p \leq 0.05$).

Surprisingly, none of the discriminative evaluators have a positive correlation with the human evaluators. That says, *generators that fool machine judges easily are less likely to confuse human judges, and vice versa. Word-overlap evaluators tend to have a positive correlation with the human evaluators in ranking the generators*. Among them, BLEU appears to be closer to humans. This pattern is consistent in all three types of correlations. These two observations are intriguing, which indi-

cates that when identifying fake reviews, humans might focus more on word usage rather than trying to construct a "decision boundary" mentally.

In summary, we find that 1) human evaluators cannot distinguish machine-generated reviews from real reviews perfectly, with significant bias between the two classes; 2) meta-adversarial evaluators can better distinguish individual fake reviews, but their rankings at the generator level tend to be negatively correlated with human evaluators; and 3) text-overlap evaluators are highly correlated with human evaluators in ranking generators.

## 5 Discussion

We carried a systematic experiment that evaluates the evaluators for NLG. Results have intriguing implications to both the evaluation and the construction of natural language generators. We conduct in-depth analysis to discover possible explanations.

### 5.1 Granularity of Judgments

We charged the Turkers to label individual reviews as either fake or real instead of evaluating each generator as a whole. Comparing to an adversarial discriminator, a human judge has not seen many "training" examples of *fake* reviews or generators. That explains why the meta-adversarial evaluators are better at identifying fake reviews. In this context, humans are likely to judge whether a review is real based on how "similar" it appears to the true reviews they are used to seeing online.

This finding provides interesting implications to the selection of evaluation methods for different tasks. In tasks that are set up to judge individual pieces of generated text (e.g., reviews, translations, summaries, captions, fake news) where there exists human-written ground-truth, it is better to use word-overlap metrics instead of adversarial evaluators. When judgments are made on the agent/system level (e.g., whether a Twitter account is a bot), signals like how similar the agent outputs are or how much the agent memorizes the training examples may become more useful than word usage, and a discriminative evaluator may be more effective than word-overlap metrics. Our finding also implies that adversarial accuracy might not be the optimal objective for NLG if the goal is to generate documents that humans consider as real. Indeed, a fake review that fools humans does not necessarily need to fool a machine that has seen everything. In Appendix B.2 we provide more details.

### 5.2 Imperfect Ground Truth

One important thing to note is that all discriminative evaluators are trained using natural labels (i.e., treating all examples from the Amazon review dataset as positive and examples generated by the candidate models as negative) instead of human-annotated labels. Some reviews posted on Amazon may have been generated by bots, and if that is the case, treating them as human-written examples may bias the discriminators. To verify this, we apply the already trained meta-discriminators to the human-annotated subset (3,600 reviews) instead of the full *D-test* set, and we use the majority vote of human judges (whether a review is fake or real) to surrogate the natural "ground-truth" labels (whether a review is generated or sampled from Amazon).

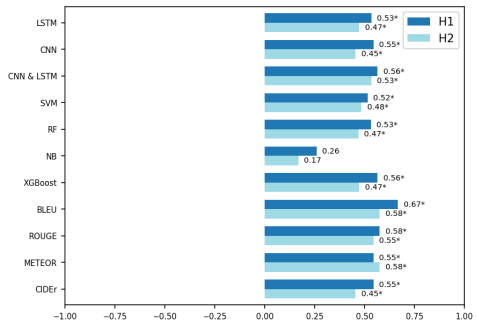


Figure 6: Kendall $\tau$-b correlation coefficients between human evaluators and automated evaluators, tested on the **annotated subset of *D-test*** with *majority votes* as ground-truth (* denotes $p \leq 0.05$).

When the meta-adversarial evaluators are tested using human majority-votes as ground-truth, the scores and rankings of these discriminative evaluators are more inline with the human evaluators, although still not as highly correlated as BLEU; please see Figure 6. Indeed, discriminative evaluators suffer the most from low-quality labels, as they were directly trained using the imperfect ground-truth. Word-overlap evaluators are more robust, as they only take the most relevant parts of the test set as references (more likely to be high quality). Our results also suggest that when adversarial training is used, selection of training examples must be done with caution. If the "ground-truth" is hijacked by low quality or "fake" examples, models trained by GAN may be significantly biased. This finding is related to the recent literature of the robustness and security of machine learning models (Papernot et al., 2017). Appendix B.3 contains further details.

### 5.3 Role of Diversity

We assess the role diversity plays in rankings the generators. Diversity of a generator is measured by either the lexical diversity (Bache et al., 2013) or Self-BLEU (Zhu et al., 2018) of the samples produced by the generator. Results obtained (see Figure 7) indicate generators that produce the least diverse samples are easily distinguished by the meta-discriminators, while they confuse humans the most. This confirms that adversarial discriminators capture limitations of generative models in lack of diversity (Kannan and Vinyals, 2017).

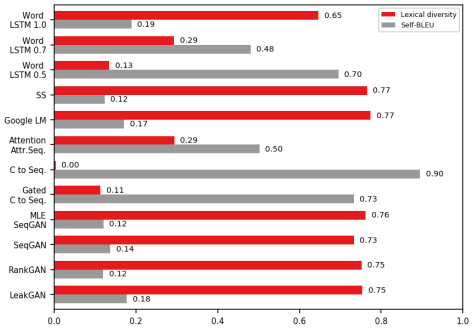


Figure 7: Self-BLEU scores (the lower the more diverse) and lexical diversity scores (the higher the more diverse) are highly correlated in ranking the generators.

Similarly, we measure to what extent the generators are memorizing the training set *G-train* as the average BLEU scores of generated reviews using their nearest neighbors in *G-train* as references. We observe the generators do not memorize the training set, and GAN models generate reviews that have fewer overlap with *G-train*; this finding is in line with recent theoretical studies on memorization in GANs (Nagarajan et al.).

The effects of diversity indicate that when humans are distinguishing individual reviews as real or fake, whether or not a fake review is sufficiently different from the other generated reviews is not a major factor for their decision. Instead, they tend to focus on whether the review looks similar to the reviews they have seen in reality. A discriminative evaluator is more powerful in making decisions at a system level (e.g., a dialog system or a bot account), where diversity becomes a major factor. In Appendix B.4 we provide more details.

### 5.4 User Study

Finally, we are interested in the reasons why human annotators label certain reviews as fake (machine-written). After annotating a batch of reviews, workers are asked to explain their decisions by filling in an optional free-text comment. This enables us to have a better understanding of what differentiates machine-generated from human-written reviews from human's perspective. Analyzing their comments, we identify the main reasons why human evaluators annotate a review as machine-written. These are mainly related to the presence of grammatical errors in the review text, wrong wording or inappropriate choice of expressions, redundant use of specific phrases or contradictory arguments in the review. Interestingly, human evaluators' innate biases are also reflected in their decisions: they are likely to categorize a review as fake if it is too formal, lacks emotion and personal pronouns, or is too vague and generic. Please see Appendix B.1.

### 5.5 Summary

In summary, our findings represent a preliminary foundation for proposing more solid and robust evaluation metrics and objectives of natural language generation. The low inter-rater agreement we observe suggests that judging *individual* pieces of text as machine- or human-generated is a difficult task even for humans. In this context, discriminative evaluators are not as correlated with human judges as word-overlap evaluators. That implies that adversarial accuracy might not be the optimal objective for generating individual documents when realism is the main concern. In contrast, GAN based models may more easily pass a Turing test on a *system* level, or in a conversational context. When the judges have seen enough examples from the same generator, the next example had better be somewhat different.

Our results also suggest that when adversarial evaluation is used, the training examples must be carefully selected to avoid false-positives. We also find that when humans are distinguishing fake reviews from real ones, they tend to focus more on the usage of words, expressions, emotions, and other details. This may affect the design of objectives for the next generation of NLG models.

## Acknowledgement

We thank Wei Ai for his help on the power analysis, and Yue Wang and Teng Ye for helpful discussions. This work is in part supported by the National Science Foundation under grant numbers 1633370 and 1620319 and by the National Library of Medicine under grant number 2R01LM010681-05.

## References


Kevin Bache, David Newman, and Padhraic Smyth. 2013. Text-based measures of document diversity. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 23–31. ACM.

Philip Bachman and Doina Precup. 2015. Data generation as sequential decision making. In *Advances in Neural Information Processing Systems*, pages 3249–3257.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21.

Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. 2016. Neural photo editing with introspective adversarial networks. *arXiv preprint arXiv:1609.07093*.

Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. 2012. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43.

Elia Bruni and Raquel Fernández. 2017. Adversarial evaluation for open-domain dialogue generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 284–288.

Massimo Caccia, Lucas Caccia, William Fedus, Hugo Larochelle, Joelle Pineau, and Laurent Charlin. 2018. Language gans falling short. *arXiv preprint arXiv:1811.02549*.

Chris Callison-Burch, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. 2008. Further meta-evaluation of machine translation. In *Proceedings of the third workshop on statistical machine translation*, pages 70–106. Association for Computational Linguistics.

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. 2006. Re-evaluation the role of bleu in machine translation research. In *11th Conference of the European Chapter of the Association for Computational Linguistics*.

Tong Che, Yanran Li, Ruixiang Zhang, R Devon Hjelm, Wenjie Li, Yangqiu Song, and Yoshua Bengio. 2017. Maximum-likelihood augmented discrete generative adversarial networks. *arXiv preprint arXiv:1702.07983*.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM.

Erik Christensen. 2007. Methodology of superiority vs. equivalence trials and non-inferiority trials. *Journal of hepatology*, 46(5):947–954.

Ondřej Cífka, Aliaksei Severyn, Enrique Alfonseca, and Katja Filippova. 2018. Eval all, trust a few, do wrong to none: Comparing sentence generation models. *arXiv preprint arXiv:1804.07972*.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

Bo Dai, Sanja Fidler, Raquel Urtasun, and Dahua Lin. 2017. Towards diverse and natural image descriptions via a conditional gan. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2970–2979.

Robert Dale and Chris Mellish. 1998. Towards evaluation in natural language generation. In *In Proceedings of First International Conference on Language Resources and Evaluation*.

Wayne W Daniel et al. 1978. *Applied nonparametric statistics*. Houghton Mifflin.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Li Dong, Shaohan Huang, Furu Wei, Mirella Lapata, Ming Zhou, and Ke Xu. 2017. Learning to generate product reviews from attributes. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 623–632.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660.

William Fedus, Ian Goodfellow, and Andrew M Dai. 2018. Maskgan: Better text generation via filling in the _. *arXiv preprint arXiv:1801.07736*.

Jessica Ficler and Yoav Goldberg. 2017. Controlling linguistic style aspects in neural language generation. *EMNLP 2017*, page 94.

Edgar C Fieller, Herman O Hartley, and Egon S Pearson. 1957. Tests for rank correlation coefficients. i. *Biometrika*, 44(3/4):470–481.

Joseph L Fleiss, Bruce Levin, and Myunghee Cho Paik. 2013. *Statistical methods for rates and proportions*. John Wiley & Sons.

Daniela Gerz, Ivan Vulić, Edoardo Ponti, Jason Naradowsky, Roi Reichart, and Anna Korhonen. 2018. Language modeling for morphologically rich languages: Character-aware modeling for word-level prediction. *Transactions of the Association of Computational Linguistics*, 6:451–465.

Ian Goodfellow. 2016. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Yvette Graham and Timothy Baldwin. 2014. Testing for significance of increased correlation with human judgment. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 172–176.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. 2018. Long text generation via adversarial training with leaked information. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Kelvin Guu, Tatsunori B Hashimoto, Yonatan Oren, and Percy Liang. 2018. Generating sentences by editing prototypes. *Transactions of the Association of Computational Linguistics*, 6:437–450.

David Hardcastle and Donia Scott. 2008. Can we evaluate the quality of generated text? In *LREC*. Citeseer.

Donna K Harman and Ellen M Voorhees. 2006. Trec: An overview. *Annual review of information science and technology*, 40(1):113–155.

Tatsunori B Hashimoto, Hugh Zhang, and Percy Liang. 2019. Unifying human and statistical evaluation for natural language generation. *arXiv preprint arXiv:1904.02792*.

Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. 2001. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Ferenc Huszár. 2015. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? *arXiv preprint arXiv:1511.05101*.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Fred Jelinek, Robert L Mercer, Lalit R Bahl, and James K Baker. 1977. Perplexity?a measure of the difficulty of speech recognition tasks. *The Journal of the Acoustical Society of America*, 62(S1):S63–S63.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

Anjuli Kannan and Oriol Vinyals. 2017. Adversarial evaluation of dialogue models. *arXiv preprint arXiv:1701.08198*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Alex M Lamb, Anirudh Goyal ALIAS PARTH GOYAL, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. 2016. Professor forcing: A new algorithm for training recurrent networks. In *Advances In Neural Information Processing Systems*, pages 4601–4609.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2157–2169.

Andy Liaw, Matthew Wiener, et al. 2002. Classification and regression by randomforest. *R news*, 2(3):18–22.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Kevin Lin, Dianqi Li, Xiaodong He, Zhengyou Zhang, and Ming-Ting Sun. 2017. Adversarial ranking for language generation. In *Advances in Neural Information Processing Systems*, pages 3155–3165.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2122–2132.

Adam Lopez. 2012. Putting human assessments of machine translation systems in order. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 1–9. Association for Computational Linguistics.

Ryan Lowe, Michael Noseworthy, Iulian Vlad Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. 2017. Towards an automatic turing test: Learning to evaluate dialogue responses. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1116–1126.

Sidi Lu, Yaoming Zhu, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Neural text generation: Past, present and beyond. *arXiv preprint arXiv:1803.07133*.

Fujun Luan, Sylvain Paris, Eli Shechtman, and Kavita Bala. 2017. Deep photo style transfer. *CoRR, abs/1703.07511*.

Pauline Luc, Camille Couprie, Soumith Chintala, and Jakob Verbeek. 2016. Semantic segmentation using adversarial networks. *arXiv preprint arXiv:1611.08408*.

Computing Machinery. 1950. Computing machinery and intelligence-am turing. *Mind*, 59(236):433.

Chris Mellish and Robert Dale. 1998. Evaluation in the context of natural language generation. *Computer Speech & Language*, 12(4):349–373.

Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2011a. Extensions of recurrent neural network language model. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5528–5531. IEEE.

Tomas Mikolov, Stefan Kombrink, Anoop Deoras, Lukar Burget, and Jan Cernocky. 2011b. Rnnlm-recurrent neural network language modeling toolkit. In *Proc. of the 2011 ASRU Workshop*, pages 196–201.

Tomas Mikolov and Geoffrey Zweig. 2012. Context dependent recurrent neural network language model. *SLT*, 12:234–239.

Ehsan Montahaei, Danial Alihosseini, and Mahdieh Soleymani Baghshah. 2019. Jointly measuring diversity and quality in text generation models. *arXiv preprint arXiv:1904.03971*.

Vaishnavh Nagarajan, Colin Raffel, and Ian J. Goodfellow. Theoretical Insights into Memorization in GANs.

Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T Hancock. 2011. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 309–319. Association for Computational Linguistics.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2018. Unsupervised learning of sentence embeddings using compositional n-gram features. In *Proceedings of NAACL-HLT*, pages 528–540.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2017. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519. ACM.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Mark A. Przybocki, Kay Peterson, Sebastien Bronsart, and Gregory A. Sanders. 2009. The NIST 2008 metrics for machine translation challenge - overview, methodology, metrics, and results. *Machine Translation*, 23(2-3):71–103.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1:8.

Sai Rajeswar, Sandeep Subramanian, Francis Dutil, Christopher Pal, and Aaron Courville. 2017. Adversarial generation of natural language. *arXiv preprint arXiv:1705.10929*.

Ehud Reiter and Anja Belz. 2009. An investigation into the validity of some metrics for automatically evaluating natural language generation systems. *Computational Linguistics*, 35(4):529–558.

Ehud Reiter, Roma Robertson, A Scott Lennox, and Liesl Osman. 2001. Using a randomised controlled clinical trial to evaluate an nlg system. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 442–449. Association for Computational Linguistics.

Irina Rish. 2001. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46. IBM.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242.

Jürgen Schmidhuber and Sepp Hochreiter. 1997. Long short-term memory. *Neural Comput*, 9(8):1735–1780.

Stanislau Semeniuta, Aliaksei Severyn, and Sylvain Gelly. 2018. On accurate evaluation of gans for language generation. *arXiv preprint arXiv:1806.04936*.

Shikhar Sharma, Layla El Asri, Hannes Schulz, and Jeremie Zumer. 2017. Relevance of unsupervised metrics in task-oriented dialogue for evaluating natural language generation. *arXiv preprint arXiv:1706.09799*.

Zhan Shi, Xinchi Chen, Xipeng Qiu, and Xuanjing Huang. 2018. Toward diverse text generation with inverse reinforcement learning. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 4361–4367. AAAI Press.

Nasim Souly, Concetto Spampinato, and Mubarak Shah. 2017. Semi and weakly supervised semantic segmentation using generative adversarial network. *arXiv preprint arXiv:1703.09695*.

Charles Spearman. 1904. The proof and measurement of association between two things. *The American journal of psychology*, 15(1):72–101.

Akash Srivastava, Lazar Valkoz, Chris Russell, Michael U Gutmann, and Charles Sutton. 2017. Veegan: Reducing mode collapse in gans using implicit variational learning. In *Advances in Neural Information Processing Systems*, pages 3310–3320.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *arXiv preprint arXiv:1505.00387*.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063.

Yaniv Taigman, Adam Polyak, and Lior Wolf. 2016. Unsupervised cross-domain image generation. *arXiv preprint arXiv:1611.02200*.

Jian Tang, Yifan Yang, Sam Carton, Ming Zhang, and Qiaozhu Mei. 2016. Context-aware natural language generation with recurrent neural networks. *arXiv preprint arXiv:1611.09900*.

Guy Tevet, Gavriel Habib, Vered Shwartz, and Jonathan Berant. 2018. Evaluating text gans as language models. *arXiv preprint arXiv:1810.12686*.

L Theis, A van den Oord, and M Bethge. 2016. A note on the evaluation of generative models. In *International Conference on Learning Representations (ICLR 2016)*, pages 1–10.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pages 5–32. Springer.

Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057.

Denis Yarats and Mike Lewis. 2018. Hierarchical text generation and planning for strategic dialogue. In *International Conference on Machine Learning*, pages 5587–5595.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*, pages 2852–2858.

Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. 2017. Adversarial feature matching for text generation. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 4006–4015. JMLR. org.

Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. 2016. Generative visual manipulation on the natural image manifold. In *European Conference on Computer Vision*, pages 597–613. Springer.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593*.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1097–1100. ACM.

# Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks

**Nils Reimers and Iryna Gurevych**
Ubiquitous Knowledge Processing Lab (UKP-TUDA)
Department of Computer Science, Technische Universität Darmstadt
www.ukp.tu-darmstadt.de

## Abstract

BERT (Devlin et al., 2018) and RoBERTa (Liu et al., 2019) has set a new state-of-the-art performance on sentence-pair regression tasks like semantic textual similarity (STS). However, it requires that both sentences are fed into the network, which causes a massive computational overhead: Finding the most similar pair in a collection of 10,000 sentences requires about 50 million inference computations (~65 hours) with BERT. The construction of BERT makes it unsuitable for semantic similarity search as well as for unsupervised tasks like clustering.

In this publication, we present Sentence-BERT (SBERT), a modification of the pretrained BERT network that use siamese and triplet network structures to derive semantically meaningful sentence embeddings that can be compared using cosine-similarity. This reduces the effort for finding the most similar pair from 65 hours with BERT / RoBERTa to about 5 seconds with SBERT, while maintaining the accuracy from BERT.

We evaluate SBERT and SRoBERTa on common STS tasks and transfer learning tasks, where it outperforms other state-of-the-art sentence embeddings methods.[1]

## 1 Introduction

In this publication, we present Sentence-BERT (SBERT), a modification of the BERT network using siamese and triplet networks that is able to derive semantically meaningful sentence embeddings[2]. This enables BERT to be used for certain new tasks, which up-to-now were not applicable for BERT. These tasks include large-scale semantic similarity comparison, clustering, and information retrieval via semantic search.

BERT set new state-of-the-art performance on various sentence classification and sentence-pair regression tasks. BERT uses a cross-encoder: Two sentences are passed to the transformer network and the target value is predicted. However, this setup is unsuitable for various pair regression tasks due to too many possible combinations. Finding in a collection of $n = 10\,000$ sentences the pair with the highest similarity requires with BERT $n \cdot (n-1)/2 = 49\,995\,000$ inference computations. On a modern V100 GPU, this requires about 65 hours. Similar, finding which of the over 40 million existent questions of Quora is the most similar for a new question could be modeled as a pair-wise comparison with BERT, however, answering a single query would require over 50 hours.

A common method to address clustering and semantic search is to map each sentence to a vector space such that semantically similar sentences are close. Researchers have started to input individual sentences into BERT and to derive fixed-size sentence embeddings. The most commonly used approach is to average the BERT output layer (known as BERT embeddings) or by using the output of the first token (the `[CLS]` token). As we will show, this common practice yields rather bad sentence embeddings, often worse than averaging GloVe embeddings (Pennington et al., 2014).

To alleviate this issue, we developed SBERT. The siamese network architecture enables that fixed-sized vectors for input sentences can be derived. Using a similarity measure like cosine-similarity or Manhatten / Euclidean distance, semantically similar sentences can be found. These similarity measures can be performed extremely efficient on modern hardware, allowing SBERT to be used for semantic similarity search as well as for clustering. The complexity for finding the

[1] Code available: https://github.com/UKPLab/sentence-transformers

[2] With *semantically meaningful* we mean that semantically similar sentences are close in vector space.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3982–3992,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

most similar sentence pair in a collection of 10,000 sentences is reduced from 65 hours with BERT to the computation of 10,000 sentence embeddings (~5 seconds with SBERT) and computing cosine-similarity (~0.01 seconds). By using optimized index structures, finding the most similar Quora question can be reduced from 50 hours to a few milliseconds (Johnson et al., 2017).

We fine-tune SBERT on NLI data, which creates sentence embeddings that significantly outperform other state-of-the-art sentence embedding methods like InferSent (Conneau et al., 2017) and Universal Sentence Encoder (Cer et al., 2018). On seven Semantic Textual Similarity (STS) tasks, SBERT achieves an improvement of 11.7 points compared to InferSent and 5.5 points compared to Universal Sentence Encoder. On SentEval (Conneau and Kiela, 2018), an evaluation toolkit for sentence embeddings, we achieve an improvement of 2.1 and 2.6 points, respectively.

SBERT can be adapted to a specific task. It sets new state-of-the-art performance on a challenging argument similarity dataset (Misra et al., 2016) and on a triplet dataset to distinguish sentences from different sections of a Wikipedia article (Dor et al., 2018).

The paper is structured in the following way: Section 3 presents SBERT, section 4 evaluates SBERT on common STS tasks and on the challenging Argument Facet Similarity (AFS) corpus (Misra et al., 2016). Section 5 evaluates SBERT on SentEval. In section 6, we perform an ablation study to test some design aspect of SBERT. In section 7, we compare the computational efficiency of SBERT sentence embeddings in contrast to other state-of-the-art sentence embedding methods.

## 2 Related Work

We first introduce BERT, then, we discuss state-of-the-art sentence embedding methods.

BERT (Devlin et al., 2018) is a pre-trained transformer network (Vaswani et al., 2017), which set for various NLP tasks new state-of-the-art results, including question answering, sentence classification, and sentence-pair regression. The input for BERT for sentence-pair regression consists of the two sentences, separated by a special `[SEP]` token. Multi-head attention over 12 (base-model) or 24 layers (large-model) is applied and the output is passed to a simple regression function to derive the final label. Using this setup, BERT set a new state-of-the-art performance on the Semantic Textual Semilarity (STS) benchmark (Cer et al., 2017). RoBERTa (Liu et al., 2019) showed, that the performance of BERT can further improved by small adaptations to the pre-training process. We also tested XLNet (Yang et al., 2019), but it led in general to worse results than BERT.

A large disadvantage of the BERT network structure is that no independent sentence embeddings are computed, which makes it difficult to derive sentence embeddings from BERT. To bypass this limitations, researchers passed single sentences through BERT and then derive a fixed sized vector by either averaging the outputs (similar to average word embeddings) or by using the output of the special `CLS` token (for example: May et al. (2019); Zhang et al. (2019); Qiao et al. (2019)). These two options are also provided by the popular bert-as-a-service-repository[3]. Up to our knowledge, there is so far no evaluation if these methods lead to useful sentence embeddings.

Sentence embeddings are a well studied area with dozens of proposed methods. Skip-Thought (Kiros et al., 2015) trains an encoder-decoder architecture to predict the surrounding sentences. InferSent (Conneau et al., 2017) uses labeled data of the Stanford Natural Language Inference dataset (Bowman et al., 2015) and the Multi-Genre NLI dataset (Williams et al., 2018) to train a siamese BiLSTM network with max-pooling over the output. Conneau et al. showed, that InferSent consistently outperforms unsupervised methods like SkipThought. Universal Sentence Encoder (Cer et al., 2018) trains a transformer network and augments unsupervised learning with training on SNLI. Hill et al. (2016) showed, that the task on which sentence embeddings are trained significantly impacts their quality. Previous work (Conneau et al., 2017; Cer et al., 2018) found that the SNLI datasets are suitable for training sentence embeddings. Yang et al. (2018) presented a method to train on conversations from Reddit using siamese DAN and siamese transformer networks, which yielded good results on the STS benchmark dataset.

Humeau et al. (2019) addresses the run-time overhead of the cross-encoder from BERT and present a method (poly-encoders) to compute a score between $m$ context vectors and pre-

[3] https://github.com/hanxiao/bert-as-service/

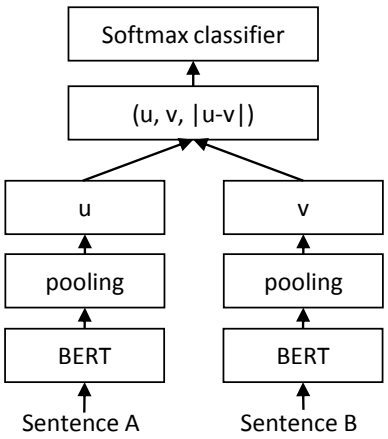


Figure 1: SBERT architecture with classification objective function, e.g., for fine-tuning on SNLI dataset. The two BERT networks have tied weights (siamese network structure).

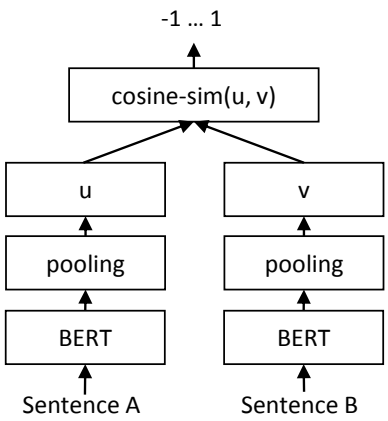


Figure 2: SBERT architecture at inference, for example, to compute similarity scores. This architecture is also used with the regression objective function.

computed candidate embeddings using attention. This idea works for finding the highest scoring sentence in a larger collection. However, poly-encoders have the drawback that the score function is not symmetric and the computational overhead is too large for use-cases like clustering, which would require $O(n^2)$ score computations.

Previous neural sentence embedding methods started the training from a random initialization. In this publication, we use the pre-trained BERT and RoBERTa network and only fine-tune it to yield useful sentence embeddings. This reduces significantly the needed training time: SBERT can be tuned in less than 20 minutes, while yielding better results than comparable sentence embedding methods.

## 3 Model

SBERT adds a pooling operation to the output of BERT / RoBERTa to derive a fixed sized sentence embedding. We experiment with three pooling strategies: Using the output of the `CLS`-token, computing the mean of all output vectors (`MEAN`-strategy), and computing a max-over-time of the output vectors (`MAX`-strategy). The default configuration is `MEAN`.

In order to fine-tune BERT / RoBERTa, we create siamese and triplet networks (Schroff et al., 2015) to update the weights such that the produced sentence embeddings are semantically meaningful and can be compared with cosine-similarity.

The network structure depends on the available training data. We experiment with the following structures and objective functions.

**Classification Objective Function.** We concatenate the sentence embeddings $u$ and $v$ with the element-wise difference $|u-v|$ and multiply it with the trainable weight $W_t \in \mathbb{R}^{3n \times k}$:

$$o = \text{softmax}(W_t(u, v, |u - v|))$$

where $n$ is the dimension of the sentence embeddings and $k$ the number of labels. We optimize cross-entropy loss. This structure is depicted in Figure 1.

**Regression Objective Function.** The cosine-similarity between the two sentence embeddings $u$ and $v$ is computed (Figure 2). We use mean-squared-error loss as the objective function.

**Triplet Objective Function.** Given an anchor sentence $a$, a positive sentence $p$, and a negative sentence $n$, triplet loss tunes the network such that the distance between $a$ and $p$ is smaller than the distance between $a$ and $n$. Mathematically, we minimize the following loss function:

$$max(||s_a - s_p|| - ||s_a - s_n|| + \epsilon, 0)$$

with $s_x$ the sentence embedding for $a/n/p$, $||\cdot||$ a distance metric and margin $\epsilon$. Margin $\epsilon$ ensures that $s_p$ is at least $\epsilon$ closer to $s_a$ than $s_n$. As metric we use Euclidean distance and we set $\epsilon = 1$ in our experiments.

### 3.1 Training Details

We train SBERT on the combination of the SNLI (Bowman et al., 2015) and the Multi-Genre NLI

| Model | STS12 | STS13 | STS14 | STS15 | STS16 | STSb | SICK-R | Avg. |
|---|---|---|---|---|---|---|---|---|
| Avg. GloVe embeddings | 55.14 | 70.66 | 59.73 | 68.25 | 63.66 | 58.02 | 53.76 | 61.32 |
| Avg. BERT embeddings | 38.78 | 57.98 | 57.98 | 63.15 | 61.06 | 46.35 | 58.40 | 54.81 |
| BERT CLS-vector | 20.16 | 30.01 | 20.09 | 36.88 | 38.08 | 16.50 | 42.63 | 29.19 |
| InferSent - Glove | 52.86 | 66.75 | 62.15 | 72.77 | 66.87 | 68.03 | 65.65 | 65.01 |
| Universal Sentence Encoder | 64.49 | 67.80 | 64.61 | 76.83 | 73.18 | 74.92 | **76.69** | 71.22 |
| SBERT-NLI-base | 70.97 | 76.53 | 73.19 | 79.09 | 74.30 | 77.03 | 72.91 | 74.89 |
| SBERT-NLI-large | 72.27 | **78.46** | **74.90** | 80.99 | 76.25 | **79.23** | 73.75 | 76.55 |
| SRoBERTa-NLI-base | 71.54 | 72.49 | 70.80 | 78.74 | 73.69 | 77.77 | 74.46 | 74.21 |
| SRoBERTa-NLI-large | **74.53** | 77.00 | 73.18 | **81.85** | **76.82** | 79.10 | 74.29 | **76.68** |

Table 1: Spearman rank correlation $\rho$ between the cosine similarity of sentence representations and the gold labels for various Textual Similarity (STS) tasks. Performance is reported by convention as $\rho \times 100$. STS12-STS16: SemEval 2012-2016, STSb: STSbenchmark, SICK-R: SICK relatedness dataset.

(Williams et al., 2018) dataset. The SNLI is a collection of 570,000 sentence pairs annotated with the labels *contradiction*, *eintailment*, and *neutral*. MultiNLI contains 430,000 sentence pairs and covers a range of genres of spoken and written text. We fine-tune SBERT with a 3-way softmax-classifier objective function for one epoch. We used a batch-size of 16, Adam optimizer with learning rate $2e{-}5$, and a linear learning rate warm-up over 10% of the training data. Our default pooling strategy is `MEAN`.

## 4 Evaluation - Semantic Textual Similarity

We evaluate the performance of SBERT for common Semantic Textual Similarity (STS) tasks. State-of-the-art methods often learn a (complex) regression function that maps sentence embeddings to a similarity score. However, these regression functions work pair-wise and due to the combinatorial explosion those are often not scalable if the collection of sentences reaches a certain size. Instead, we always use cosine-similarity to compare the similarity between two sentence embeddings. We ran our experiments also with negative Manhatten and negative Euclidean distances as similarity measures, but the results for all approaches remained roughly the same.

### 4.1 Unsupervised STS

We evaluate the performance of SBERT for STS without using any STS specific training data. We use the STS tasks 2012 - 2016 (Agirre et al., 2012, 2013, 2014, 2015, 2016), the STS benchmark (Cer et al., 2017), and the SICK-Relatedness dataset (Marelli et al., 2014). These datasets provide labels between 0 and 5 on the semantic relatedness of sentence pairs. We showed in (Reimers et al., 2016) that Pearson correlation is badly suited for STS. Instead, we compute the Spearman's rank correlation between the cosine-similarity of the sentence embeddings and the gold labels. The setup for the other sentence embedding methods is equivalent, the similarity is computed by cosine-similarity. The results are depicted in Table 1.

The results shows that directly using the output of BERT leads to rather poor performances. Averaging the BERT embeddings achieves an average correlation of only 54.81, and using the `CLS`-token output only achieves an average correlation of 29.19. Both are worse than computing average GloVe embeddings.

Using the described siamese network structure and fine-tuning mechanism substantially improves the correlation, outperforming both InferSent and Universal Sentence Encoder substantially. The only dataset where SBERT performs worse than Universal Sentence Encoder is SICK-R. Universal Sentence Encoder was trained on various datasets, including news, question-answer pages and discussion forums, which appears to be more suitable to the data of SICK-R. In contrast, SBERT was pre-trained only on Wikipedia (via BERT) and on NLI data.

While RoBERTa was able to improve the performance for several supervised tasks, we only observe minor difference between SBERT and SRoBERTa for generating sentence embeddings.

### 4.2 Supervised STS

The STS benchmark (STSb) (Cer et al., 2017) provides is a popular dataset to evaluate supervised STS systems. The data includes 8,628 sentence pairs from the three categories *captions*, *news*, and *forums*. It is divided into train (5,749), dev (1,500) and test (1,379). BERT set a new state-of-the-art performance on this dataset by passing both sentences to the network and using a simple regres-

sion method for the output.

| Model | Spearman |
|---|---|
| *Not trained for STS* | |
| Avg. GloVe embeddings | 58.02 |
| Avg. BERT embeddings | 46.35 |
| InferSent - GloVe | 68.03 |
| Universal Sentence Encoder | 74.92 |
| SBERT-NLI-base | 77.03 |
| SBERT-NLI-large | 79.23 |
| *Trained on STS benchmark dataset* | |
| BERT-STSb-base | 84.30 ± 0.76 |
| SBERT-STSb-base | 84.67 ± 0.19 |
| SRoBERTa-STSb-base | **84.92** ± 0.34 |
| BERT-STSb-large | **85.64** ± 0.81 |
| SBERT-STSb-large | 84.45 ± 0.43 |
| SRoBERTa-STSb-large | 85.02 ± 0.76 |
| *Trained on NLI data + STS benchmark data* | |
| BERT-NLI-STSb-base | **88.33** ± 0.19 |
| SBERT-NLI-STSb-base | 85.35 ± 0.17 |
| SRoBERTa-NLI-STSb-base | 84.79 ± 0.38 |
| BERT-NLI-STSb-large | **88.77** ± 0.46 |
| SBERT-NLI-STSb-large | 86.10 ± 0.13 |
| SRoBERTa-NLI-STSb-large | 86.15 ± 0.35 |

Table 2: Evaluation on the STS benchmark test set. BERT systems were trained with 10 random seeds and 4 epochs. SBERT was fine-tuned on the STSb dataset, SBERT-NLI was pretrained on the NLI datasets, then fine-tuned on the STSb dataset.

We use the training set to fine-tune SBERT using the regression objective function. At prediction time, we compute the cosine-similarity between the sentence embeddings. All systems are trained with 10 random seeds to counter variances (Reimers and Gurevych, 2018).

The results are depicted in Table 2. We experimented with two setups: Only training on STSb, and first training on NLI, then training on STSb. We observe that the later strategy leads to a slight improvement of 1-2 points. This two-step approach had an especially large impact for the BERT cross-encoder, which improved the performance by 3-4 points. We do not observe a significant difference between BERT and RoBERTa.

### 4.3 Argument Facet Similarity

We evaluate SBERT on the Argument Facet Similarity (AFS) corpus by Misra et al. (2016). The AFS corpus annotated 6,000 sentential argument pairs from social media dialogs on three controversial topics: *gun control*, *gay marriage*, and *death penalty*. The data was annotated on a scale from 0 ("different topic") to 5 ("completely equivalent"). The similarity notion in the AFS corpus is fairly different to the similarity notion in the STS datasets from SemEval. STS data is usually descriptive, while AFS data are argumentative excerpts from dialogs. To be considered similar, arguments must not only make similar claims, but also provide a similar reasoning. Further, the lexical gap between the sentences in AFS is much larger. Hence, simple unsupervised methods as well as state-of-the-art STS systems perform badly on this dataset (Reimers et al., 2019).

We evaluate SBERT on this dataset in two scenarios: 1) As proposed by Misra et al., we evaluate SBERT using 10-fold cross-validation. A drawback of this evaluation setup is that it is not clear how well approaches generalize to different topics. Hence, 2) we evaluate SBERT in a cross-topic setup. Two topics serve for training and the approach is evaluated on the left-out topic. We repeat this for all three topics and average the results.

SBERT is fine-tuned using the Regression Objective Function. The similarity score is computed using cosine-similarity based on the sentence embeddings. We also provide the Pearson correlation $r$ to make the results comparable to Misra et al. However, we showed (Reimers et al., 2016) that Pearson correlation has some serious drawbacks and should be avoided for comparing STS systems. The results are depicted in Table 3.

Unsupervised methods like tf-idf, average GloVe embeddings or InferSent perform rather badly on this dataset with low scores. Training SBERT in the 10-fold cross-validation setup gives a performance that is nearly on-par with BERT.

However, in the cross-topic evaluation, we observe a performance drop of SBERT by about 7 points Spearman correlation. To be considered similar, arguments should address the same claims and provide the same reasoning. BERT is able to use attention to compare directly both sentences (e.g. word-by-word comparison), while SBERT must map individual sentences from an unseen topic to a vector space such that arguments with similar claims and reasons are close. This is a much more challenging task, which appears to require more than just two topics for training to work on-par with BERT.

### 4.4 Wikipedia Sections Distinction

Dor et al. (2018) use Wikipedia to create a thematically fine-grained train, dev and test set for sentence embeddings methods. Wikipedia articles are separated into distinct sections focusing on certain aspects. Dor et al. assume that sen-

| Model | $r$ | $\rho$ |
|---|---|---|
| *Unsupervised methods* | | |
| tf-idf | 46.77 | 42.95 |
| Avg. GloVe embeddings | 32.40 | 34.00 |
| InferSent - GloVe | 27.08 | 26.63 |
| *10-fold Cross-Validation* | | |
| SVR (Misra et al., 2016) | 63.33 | - |
| BERT-AFS-base | 77.20 | 74.84 |
| SBERT-AFS-base | 76.57 | 74.13 |
| BERT-AFS-large | 78.68 | 76.38 |
| SBERT-AFS-large | 77.85 | 75.93 |
| *Cross-Topic Evaluation* | | |
| BERT-AFS-base | 58.49 | 57.23 |
| SBERT-AFS-base | 52.34 | 50.65 |
| BERT-AFS-large | 62.02 | 60.34 |
| SBERT-AFS-large | 53.82 | 53.10 |

Table 3: Average Pearson correlation $r$ and average Spearman's rank correlation $\rho$ on the Argument Facet Similarity (AFS) corpus (Misra et al., 2016). Misra et al. proposes 10-fold cross-validation. We additionally evaluate in a cross-topic scenario: Methods are trained on two topics, and are evaluated on the third topic.

tences in the same section are thematically closer than sentences in different sections. They use this to create a large dataset of weakly labeled sentence triplets: The anchor and the positive example come from the same section, while the negative example comes from a different section of the same article. For example, from the Alice Arnold article: Anchor: *Arnold joined the BBC Radio Drama Company in 1988.*, positive: *Arnold gained media attention in May 2012.*, negative: *Balding and Arnold are keen amateur golfers.*

We use the dataset from Dor et al. We use the Triplet Objective, train SBERT for one epoch on the about 1.8 Million training triplets and evaluate it on the 222,957 test triplets. Test triplets are from a distinct set of Wikipedia articles. As evaluation metric, we use accuracy: Is the positive example closer to the anchor than the negative example?

Results are presented in Table 4. Dor et al. fine-tuned a BiLSTM architecture with triplet loss to derive sentence embeddings for this dataset. As the table shows, SBERT clearly outperforms the BiLSTM approach by Dor et al.

## 5 Evaluation - SentEval

SentEval (Conneau and Kiela, 2018) is a popular toolkit to evaluate the quality of sentence embeddings. Sentence embeddings are used as features for a logistic regression classifier. The logistic regression classifier is trained on various tasks in a 10-fold cross-validation setup and the prediction accuracy is computed for the test-fold.

| Model | Accuracy |
|---|---|
| mean-vectors | 0.65 |
| skip-thoughts-CS | 0.62 |
| Dor et al. | 0.74 |
| SBERT-WikiSec-base | 0.8042 |
| SBERT-WikiSec-large | **0.8078** |
| SRoBERTa-WikiSec-base | 0.7945 |
| SRoBERTa-WikiSec-large | 0.7973 |

Table 4: Evaluation on the Wikipedia section triplets dataset (Dor et al., 2018). SBERT trained with triplet loss for one epoch.

The purpose of SBERT sentence embeddings are not to be used for transfer learning for other tasks. Here, we think fine-tuning BERT as described by Devlin et al. (2018) for new tasks is the more suitable method, as it updates all layers of the BERT network. However, SentEval can still give an impression on the quality of our sentence embeddings for various tasks.

We compare the SBERT sentence embeddings to other sentence embeddings methods on the following seven SentEval transfer tasks:

- **MR**: Sentiment prediction for movie reviews snippets on a five start scale (Pang and Lee, 2005).
- **CR**: Sentiment prediction of customer product reviews (Hu and Liu, 2004).
- **SUBJ**: Subjectivity prediction of sentences from movie reviews and plot summaries (Pang and Lee, 2004).
- **MPQA**: Phrase level opinion polarity classification from newswire (Wiebe et al., 2005).
- **SST**: Stanford Sentiment Treebank with binary labels (Socher et al., 2013).
- **TREC**: Fine grained question-type classification from TREC (Li and Roth, 2002).
- **MRPC**: Microsoft Research Paraphrase Corpus from parallel news sources (Dolan et al., 2004).

The results can be found in Table 5. SBERT is able to achieve the best performance in 5 out of 7 tasks. The average performance increases by about 2 percentage points compared to InferSent as well as the Universal Sentence Encoder. Even though transfer learning is not the purpose of SBERT, it outperforms other state-of-the-art sentence embeddings methods on this task.

| Model | MR | CR | SUBJ | MPQA | SST | TREC | MRPC | Avg. |
|---|---|---|---|---|---|---|---|---|
| Avg. GloVe embeddings | 77.25 | 78.30 | 91.17 | 87.85 | 80.18 | 83.0 | 72.87 | 81.52 |
| Avg. fast-text embeddings | 77.96 | 79.23 | 91.68 | 87.81 | 82.15 | 83.6 | 74.49 | 82.42 |
| Avg. BERT embeddings | 78.66 | 86.25 | 94.37 | 88.66 | 84.40 | 92.8 | 69.45 | 84.94 |
| BERT CLS-vector | 78.68 | 84.85 | 94.21 | 88.23 | 84.13 | 91.4 | 71.13 | 84.66 |
| InferSent - GloVe | 81.57 | 86.54 | 92.50 | **90.38** | 84.18 | 88.2 | 75.77 | 85.59 |
| Universal Sentence Encoder | 80.09 | 85.19 | 93.98 | 86.70 | 86.38 | **93.2** | 70.14 | 85.10 |
| SBERT-NLI-base | 83.64 | 89.43 | 94.39 | 89.86 | 88.96 | 89.6 | **76.00** | 87.41 |
| SBERT-NLI-large | **84.88** | **90.07** | **94.52** | 90.33 | **90.66** | 87.4 | 75.94 | **87.69** |

Table 5: Evaluation of SBERT sentence embeddings using the SentEval toolkit. SentEval evaluates sentence embeddings on different sentence classification tasks by training a logistic regression classifier using the sentence embeddings as features. Scores are based on a 10-fold cross-validation.

It appears that the sentence embeddings from SBERT capture well sentiment information: We observe large improvements for all sentiment tasks (MR, CR, and SST) from SentEval in comparison to InferSent and Universal Sentence Encoder.

The only dataset where SBERT is significantly worse than Universal Sentence Encoder is the TREC dataset. Universal Sentence Encoder was pre-trained on question-answering data, which appears to be beneficial for the question-type classification task of the TREC dataset.

Average BERT embeddings or using the CLS-token output from a BERT network achieved bad results for various STS tasks (Table 1), worse than average GloVe embeddings. However, for SentEval, average BERT embeddings and the BERT CLS-token output achieves decent results (Table 5), outperforming average GloVe embeddings. The reason for this are the different setups. For the STS tasks, we used cosine-similarity to estimate the similarities between sentence embeddings. Cosine-similarity treats all dimensions equally. In contrast, SentEval fits a logistic regression classifier to the sentence embeddings. This allows that certain dimensions can have higher or lower impact on the classification result.

We conclude that average BERT embeddings / CLS-token output from BERT return sentence embeddings that are infeasible to be used with cosine-similarity or with Manhatten / Euclidean distance. For transfer learning, they yield slightly worse results than InferSent or Universal Sentence Encoder. However, using the described fine-tuning setup with a siamese network structure on NLI datasets yields sentence embeddings that achieve a new state-of-the-art for the SentEval toolkit.

## 6 Ablation Study

We have demonstrated strong empirical results for the quality of SBERT sentence embeddings. In this section, we perform an ablation study of different aspects of SBERT in order to get a better understanding of their relative importance.

We evaluated different pooling strategies (MEAN, MAX, and CLS). For the classification objective function, we evaluate different concatenation methods. For each possible configuration, we train SBERT with 10 different random seeds and average the performances.

The objective function (classification vs. regression) depends on the annotated dataset. For the classification objective function, we train SBERT-base on the SNLI and the Multi-NLI dataset. For the regression objective function, we train on the training set of the STS benchmark dataset. Performances are measured on the development split of the STS benchmark dataset. Results are shown in Table 6.

| | NLI | STSb |
|---|---|---|
| *Pooling Strategy* | | |
| MEAN | **80.78** | **87.44** |
| MAX | 79.07 | 69.92 |
| CLS | 79.80 | 86.62 |
| *Concatenation* | | |
| $(u, v)$ | 66.04 | - |
| $(\|u - v\|)$ | 69.78 | - |
| $(u * v)$ | 70.54 | - |
| $(\|u - v\|, u * v)$ | 78.37 | - |
| $(u, v, u * v)$ | 77.44 | - |
| $(u, v, \|u - v\|)$ | **80.78** | - |
| $(u, v, \|u - v\|, u * v)$ | 80.44 | - |

Table 6: SBERT trained on NLI data with the classification objective function, on the STS benchmark (STSb) with the regression objective function. Configurations are evaluated on the development set of the STSb using cosine-similarity and Spearman's rank correlation. For the concatenation methods, we only report scores with MEAN pooling strategy.

When trained with the classification objective function on NLI data, the pooling strategy has a rather minor impact. The impact of the concatenation mode is much larger. InferSent (Conneau

et al., 2017) and Universal Sentence Encoder (Cer et al., 2018) both use $(u, v, |u - v|, u * v)$ as input for a softmax classifier. However, in our architecture, adding the element-wise $u * v$ decreased the performance.

The most important component is the element-wise difference $|u - v|$. Note, that the concatenation mode is only relevant for training the softmax classifier. At inference, when predicting similarities for the STS benchmark dataset, only the sentence embeddings $u$ and $v$ are used in combination with cosine-similarity. The element-wise difference measures the distance between the dimensions of the two sentence embeddings, ensuring that similar pairs are closer and dissimilar pairs are further apart.

When trained with the regression objective function, we observe that the pooling strategy has a large impact. There, the `MAX` strategy perform significantly worse than `MEAN` or `CLS`-token strategy. This is in contrast to (Conneau et al., 2017), who found it beneficial for the BiLSTM-layer of InferSent to use `MAX` instead of `MEAN` pooling.

## 7 Computational Efficiency

Sentence embeddings need potentially be computed for Millions of sentences, hence, a high computation speed is desired. In this section, we compare SBERT to average GloVe embeddings, InferSent (Conneau et al., 2017), and Universal Sentence Encoder (Cer et al., 2018).

For our comparison we use the sentences from the STS benchmark (Cer et al., 2017). We compute average GloVe embeddings using a simple for-loop with python dictionary lookups and NumPy. InferSent[4] is based on PyTorch. For Universal Sentence Encoder, we use the TensorFlow Hub version[5], which is based on TensorFlow. SBERT is based on PyTorch. For improved computation of sentence embeddings, we implemented a smart batching strategy: Sentences with similar lengths are grouped together and are only padded to the longest element in a mini-batch. This drastically reduces computational overhead from padding tokens.

Performances were measured on a server with Intel i7-5820K CPU @ 3.30GHz, Nvidia Tesla V100 GPU, CUDA 9.2 and cuDNN. The results are depicted in Table 7.

| Model | CPU | GPU |
|---|---|---|
| Avg. GloVe embeddings | 6469 | - |
| InferSent | 137 | 1876 |
| Universal Sentence Encoder | 67 | 1318 |
| SBERT-base | 44 | 1378 |
| SBERT-base - smart batching | 83 | 2042 |

Table 7: Computation speed (sentences per second) of sentence embedding methods. Higher is better.

On CPU, InferSent is about 65% faster than SBERT. This is due to the much simpler network architecture. InferSent uses a single BiLSTM layer, while BERT uses 12 stacked transformer layers. However, an advantage of transformer networks is the computational efficiency on GPUs. There, SBERT with smart batching is about 9% faster than InferSent and about 55% faster than Universal Sentence Encoder. Smart batching achieves a speed-up of 89% on CPU and 48% on GPU. Average GloVe embeddings is obviously by a large margin the fastest method to compute sentence embeddings.

## 8 Conclusion

We showed that BERT out-of-the-box maps sentences to a vector space that is rather unsuitable to be used with common similarity measures like cosine-similarity. The performance for seven STS tasks was below the performance of average GloVe embeddings.

To overcome this shortcoming, we presented Sentence-BERT (SBERT). SBERT fine-tunes BERT in a siamese / triplet network architecture. We evaluated the quality on various common benchmarks, where it could achieve a significant improvement over state-of-the-art sentence embeddings methods. Replacing BERT with RoBERTa did not yield a significant improvement in our experiments.

SBERT is computationally efficient. On a GPU, it is about 9% faster than InferSent and about 55% faster than Universal Sentence Encoder. SBERT can be used for tasks which are computationally not feasible to be modeled with BERT. For example, clustering of 10,000 sentences with hierarchical clustering requires with BERT about 65 hours, as around 50 Million sentence combinations must be computed. With SBERT, we were able to reduce the effort to about 5 seconds.

[4] https://github.com/facebookresearch/InferSent

[5] https://tfhub.dev/google/universal-sentence-encoder-large/3

## Acknowledgments

This work has been supported by the German Research Foundation through the German-Israeli Project Cooperation (DIP, grant DA 1600/1-1 and grant GU 798/17-1). It has been co-funded by the German Federal Ministry of Education and Research (BMBF) under the promotional references 03VP02540 (ArgumenText).

## References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Inigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 2015. SemEval-2015 Task 2: Semantic Textual Similarity, English, Spanish and Pilot on Interpretability. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 252–263, Denver, Colorado. Association for Computational Linguistics.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. SemEval-2014 Task 10: Multilingual Semantic Textual Similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 81–91, Dublin, Ireland. Association for Computational Linguistics.

Eneko Agirre, Carmen Banea, Daniel M. Cer, Mona T. Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016. SemEval-2016 Task 1: Semantic Textual Similarity, Monolingual and Cross-Lingual Evaluation. In *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2016, San Diego, CA, USA, June 16-17, 2016*, pages 497–511.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *SEM 2013 shared task: Semantic Textual Similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 32–43, Atlanta, Georgia, USA. Association for Computational Linguistics.

Eneko Agirre, Mona Diab, Daniel Cer, and Aitor Gonzalez-Agirre. 2012. SemEval-2012 Task 6: A Pilot on Semantic Textual Similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics - Volume 1: Proceedings of the Main Conference and the Shared Task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, SemEval '12, pages 385–393, Stroudsburg, PA, USA. Association for Computational Linguistics.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Daniel Cer, Mona Diab, Eneko Agirre, Iigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Crosslingual Focused Evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal Sentence Encoder. *arXiv preprint arXiv:1803.11175*.

Alexis Conneau and Douwe Kiela. 2018. SentEval: An Evaluation Toolkit for Universal Sentence Representations. *arXiv preprint arXiv:1803.05449*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised Construction of Large Paraphrase Corpora: Exploiting Massively Parallel News Sources. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Liat Ein Dor, Yosi Mass, Alon Halfon, Elad Venezian, Ilya Shnayderman, Ranit Aharonov, and Noam Slonim. 2018. Learning Thematic Similarity Metric from Article Sections Using Triplet Networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 49–54, Melbourne, Australia. Association for Computational Linguistics.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning Distributed Representations of Sentences from Unlabelled Data. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1367–1377, San Diego, California. Association for Computational Linguistics.

Minqing Hu and Bing Liu. 2004. Mining and Summarizing Customer Reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 168–177, New York, NY, USA. ACM.

Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. 2019. Real-time Inference in Multi-sentence Tasks with Deep Pretrained Transformers. *arXiv preprint arXiv:1905.01969*, abs/1905.01969.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with GPUs. *arXiv preprint arXiv:1702.08734*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-Thought Vectors. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3294–3302. Curran Associates, Inc.

Xin Li and Dan Roth. 2002. Learning Question Classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, COLING '02, pages 1–7, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692*.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 216–223, Reykjavik, Iceland. European Language Resources Association (ELRA).

Chandler May, Alex Wang, Shikha Bordia, Samuel R. Bowman, and Rachel Rudinger. 2019. On Measuring Social Biases in Sentence Encoders. *arXiv preprint arXiv:1903.10561*.

Amita Misra, Brian Ecker, and Marilyn A. Walker. 2016. Measuring the Similarity of Sentential Arguments in Dialogue. In *Proceedings of the SIGDIAL 2016 Conference, The 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 13-15 September 2016, Los Angeles, CA, USA*, pages 276–287.

Bo Pang and Lillian Lee. 2004. A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 271–278, Barcelona, Spain.

Bo Pang and Lillian Lee. 2005. Seeing Stars: Exploiting Class Relationships for Sentiment Categorization with Respect to Rating Scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 115–124, Ann Arbor, Michigan. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Yifan Qiao, Chenyan Xiong, Zheng-Hao Liu, and Zhiyuan Liu. 2019. Understanding the Behaviors of BERT in Ranking. *arXiv preprint arXiv:1904.07531*.

Nils Reimers, Philip Beyer, and Iryna Gurevych. 2016. Task-Oriented Intrinsic Evaluation of Semantic Textual Similarity. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING)*, pages 87–96.

Nils Reimers and Iryna Gurevych. 2018. Why Comparing Single Performance Scores Does Not Allow to Draw Conclusions About Machine Learning Approaches. *arXiv preprint arXiv:1803.09578*, abs/1803.09578.

Nils Reimers, Benjamin Schiller, Tilman Beck, Johannes Daxenberger, Christian Stab, and Iryna Gurevych. 2019. Classification and Clustering of Arguments with Contextualized Word Embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 567–578, Florence, Italy. Association for Computational Linguistics.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. FaceNet: A Unified Embedding for Face Recognition and Clustering. *arXiv preprint arXiv:1503.03832*, abs/1503.03832.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating Expressions of Opinions and Emotions in Language. *Language Resources and Evaluation*, 39(2):165–210.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Yinfei Yang, Steve Yuan, Daniel Cer, Sheng-Yi Kong, Noah Constant, Petr Pilar, Heming Ge, Yun-hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Learning Semantic Textual Similarity from Conversations. In *Proceedings of The Third Workshop on Representation Learning for NLP*, pages 164–174, Melbourne, Australia. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. *arXiv preprint arXiv:1906.08237*, abs/1906.08237.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2019. BERTScore: Evaluating Text Generation with BERT. *arXiv preprint arXiv:1904.09675*.

# Learning Only from Relevant Keywords and Unlabeled Documents

**Nontawat Charoenphakdee**[1,3] **Jongyeong Lee**[1,3] **Yiping Jin**[2]
**Dittaya Wanvarie**[2] **Masashi Sugiyama**[3,4,1]
[1]Department of Computer Science, The University of Tokyo
[2]Department of Mathematics and Computer Science, Faculty of Science, Chulalongkorn University
[3]RIKEN Center for Advanced Intelligence Project
[4]Department of Complexity Science and Engineering, The University of Tokyo
{nontawat, lee}@ms.k.u-tokyo.ac.jp
jinyiping@knorex.com.
Dittaya.W@chula.ac.th
sugi@k.u-tokyo.ac.jp

## Abstract

We consider a document classification problem where document labels are absent but only relevant keywords of a target class and unlabeled documents are given. Although heuristic methods based on pseudo-labeling have been considered, theoretical understanding of this problem has still been limited. Moreover, previous methods cannot easily incorporate well-developed techniques in supervised text classification. In this paper, we propose a theoretically guaranteed learning framework that is simple to implement and has flexible choices of models, e.g., linear models or neural networks. We demonstrate how to optimize the area under the receiver operating characteristic curve (AUC) effectively and also discuss how to adjust it to optimize other well-known evaluation metrics such as the accuracy and $F_1$-measure. Finally, we show the effectiveness of our framework using benchmark datasets.

## 1 Introduction

Supervised text classification is a traditional problem in natural language processing that has been studied extensively (Nigam et al., 2000; Sebastiani, 2002; Forman, 2003; Joulin et al., 2016). In this problem, we are given a set of labeled text documents and the goal is to construct a classifier that can classify an unseen document effectively. There are many useful applications for text classification, e.g., sentiment classification (Liu and Zhang, 2012; Medhat et al., 2014), biomedical text mining (Cohen and Hersh, 2005; Huang and Lu, 2015), and social media monitoring (Zeng et al., 2010; Hu and Liu, 2012).

In the real-world, it is impractical to expect that labeled data can always be obtained abundantly. For example, given big data of unlabeled texts, it can be very time-consuming and costly for the labeling task so that we can apply a supervised text classification method. Another example is when the label information is protected due to privacy concerns. These bottlenecks motivate researches on weakly-supervised learning (Zhou, 2017), which focuses on devising a machine learning method that can learn effectively although a lot of labeled data are not accessible.

An alternative is to provide keywords as a hint for classifying a document to a target class. Intuitively, this approach can be much cheaper when the number of documents is huge since the number of keywords does not necessarily grow linearly with the number of documents. *Dataless classification* is a text classification problem where we are given keywords for each class and unlabeled documents (Chang et al., 2008; Song and Roth, 2014; Chen et al., 2015; Li and Yang, 2018). Note that we do not have access to *any* labeled documents.

In this paper, we investigate dataless classification in the situation where only the keywords of the target class are given. We regard documents of a class that we have relevant keywords as positive and others as negative. Unlike dataless classification, we have access to only the keywords of the positive class. This problem setting can be considered more difficult than the traditional one since one can always reduce dataless classification to this problem by ignoring negative keywords. This scenario is also highly relevant when the information about negative classes cannot be explicitly described, e.g., in the information retrieval task, we may only have information about the target class of interest.

This problem setting has already been considered by Jin et al. (2017), where they called it *lightly-supervised one-class classification*. To the best of our knowledge, existing work in dataless classification and lightly-supervised one-class classification neither provide a theoretical guarantee, nor have flexible choices of models and opti-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3993–4002,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

mization algorithms for this problem.

The goal of this paper is to formalize lightly-supervised one-class classification and develop a reliable and flexible framework to handle this problem effectively. To achieve this goal, we propose a framework that has a theoretical guarantee and allows practitioners to have flexible choices to maximize the performance, e.g., neural network architectures such as convolutional neural networks (Zhang et al., 2015) or recurrent neural networks (Lai et al., 2015). They can also pick an optimization method that is suitable for their model such as Adam (Kingma and Ba, 2014) or AMSGrad (Reddi et al., 2018). These advantages allow us to utilize well-developed supervised-learning text classification methods in our framework.

We point out that the problem of lightly-supervised one-class classification is highly related to binary classification from corrupted labels. Intuitively, the common idea to solve this problem is to use keywords to pseudo-label given documents then performs classification (Jin et al., 2017). However, pseudo-positive and pseudo-negative labels are unreliable since they can be incorrectly labeled. Thus, the learning method should be robust against label corruption. In our framework, we also use relevant keywords to split unlabeled data into two sets. Then, the key idea is to employ a method based on an empirical risk minimization framework (Vapnik, 1998) with a loss function that benefits from a symmetric condition to maximize AUC (Charoenphakdee et al., 2019). We justify the soundness of our method by proving that our proposed framework gives a consistent estimation of the true expected AUC risk as long as the divided two sets have different proportions of positive data. We elucidate this theoretical result by providing an estimation error bound for AUC maximization using a Rademacher complexity measure. Furthermore, we discuss how to adjust our method to optimize other evaluation metrics, which are the accuracy and $\mathrm{F}_1$-measure. We also illustrate that label corruption may cause a classifier to pick a wrong decision boundary and suggest to adjust the threshold on the basis of the proportion of positive data in unlabeled data.

## 2 Preliminaries

In this section, we introduce the notation, review AUC, and an empirical risk minimization framework for optimizing AUC.

Table 1: Examples of loss functions. A symmetric loss is a loss function such that $\ell(z) + \ell(-z) = K$, where $K$ is a positive constant.

| Loss name | $\ell(z)$ | Symmetric |
|---|---|---|
| Logistic | $\log(1 + e^{-z})$ | $\times$ |
| Squared | $(1 - z)^2$ | $\times$ |
| Zero-one | $-\frac{1}{2}\mathrm{sign}(z) + \frac{1}{2}$ | ✓ |
| Sigmoid | $\frac{1}{1+\exp(z)}$ | ✓ |

### 2.1 Notation

Let $\boldsymbol{x} \in \mathcal{X}$ be a pattern in the input space $\mathcal{X}$ and $y \in \{-1, 1\}$ be a label. We denote $g : \mathcal{X} \to \mathbb{R}$ as a prediction function. Let $\mathbb{E}_{\mathrm{P}}[\cdot]$ and $\mathbb{E}_{\mathrm{N}}[\cdot]$ denote the expectation over the class-conditional probability $p(\boldsymbol{x}|y = 1)$ and $p(\boldsymbol{x}|y = -1)$, respectively. Moreover, let $\pi$ denote a class prior $p(y = 1)$, i.e., the proportion of positive data.

Loss functions that we will discuss in this paper are in the family of margin losses, where a loss receives one argument $\ell : \mathbb{R} \to \mathbb{R}$. This loss family covers a lot of well-known losses (Bartlett et al., 2006), e.g., the logistic loss, squared loss, and hinge loss. We define the zero-one loss as $\ell_{0\text{-}1}(z) = -\frac{1}{2}\mathrm{sign}(z) + \frac{1}{2}$, where $\mathrm{sign}(g(\boldsymbol{x})) = 1$ if $g(\boldsymbol{x}) > 0$, $-1$ if $g(\boldsymbol{x}) < 0$, and 0 otherwise. Finally, we consider a symmetric loss as a margin loss $\ell_{\mathrm{sym}}$ that satisfies the symmetric condition $\ell_{\mathrm{sym}}(z) + \ell_{\mathrm{sym}}(-z) = K$, where $K$ is a positive constant. Examples of such losses are the zero-one loss and sigmoid loss (Ghosh et al., 2015). Table 1 provides examples of margin losses in binary classification.

### 2.2 Area under the Receiver Operating Characteristic Curve (AUC)

AUC is an evaluation metric for a bipartite ranking task (Cortes and Mohri, 2004; Menon and Williamson, 2016), where we want to find a function that outputs a higher value for positive data than negative data. Moreover, AUC is also a popular metric for a classifier under class imbalance (Menon et al., 2015). It is also known in the literature of statistics as the Wilcoxon-Mann-Whitney statistic (Mann and Whitney, 1947; Hanley and McNeil, 1982).

Let us consider the AUC risk, i.e., *bipartite ranking risk* (Narasimhan and Agarwal, 2013):

$$R^{\ell_{0\text{-}1}}_{\mathrm{AUC}}(g) = \mathbb{E}_{\mathrm{P}}\left[\mathbb{E}_{\mathrm{N}}\left[\ell_{0\text{-}1}(g(\boldsymbol{x}^{\mathrm{P}}_i) - g(\boldsymbol{x}^{\mathrm{N}}_j))\right]\right]. \quad (1)$$

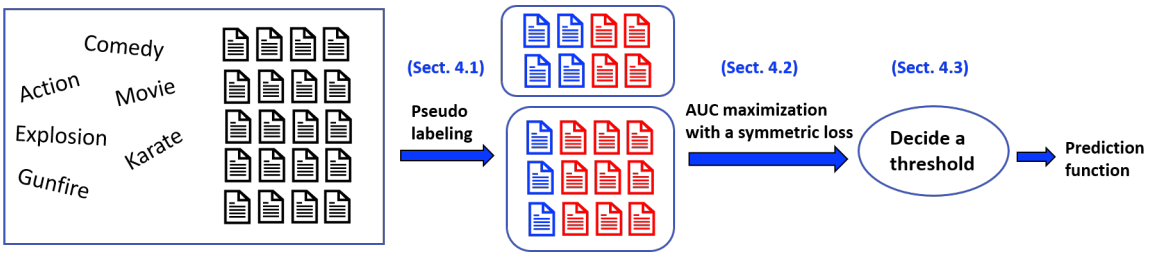


Figure 1: An overview of the framework. Blue documents indicates clean positive data and red documents denote clean negative data in the two sets of data divided by the pseudo-labeling algorithm. Note that labels are not observed by the framework. Sect. denotes a section that describes the procedure for each step.

Then, AUC is defined as $\mathrm{AUC}(g) = 1 - R^{\ell_{0\text{-}1}}_{\mathrm{AUC}}(g)$. It is important to note that unlike many other evaluation metrics, we can evaluate AUC without deciding the threshold for a function $g$. In this paper, we will show that optimizing AUC in our problem setting has a great advantage when no information about the threshold is given since we only have keywords and unlabeled documents.

### 2.3 Empirical Risk Minimization (ERM) for AUC Maximization

Here, we review a widely used framework in machine learning called the ERM framework (Vapnik, 1998) for AUC maximization.

The goal of AUC maximization is to minimize the AUC risk in (1) with respect to the zero-one loss $R^{\ell_{0\text{-}1}}_{\mathrm{AUC}}(g)$ (Menon and Williamson, 2016).

Although the goal is to minimize the bipartite ranking risk with respect to the zero-one loss $\ell_{0\text{-}1}$, a surrogate loss $\ell$, e.g., the logistic loss or the squared loss, is instead applied in practice since minimizing $\ell_{0\text{-}1}$ is computationally infeasible (Ben-David et al., 2003; Zhang, 2004; Bartlett et al., 2006; Feldman et al., 2012). Note that we cannot directly minimize the AUC risk in our setting since we do not have access to the positive and negative distributions.

To explain the ERM framework for bipartite ranking, in this section, let us assume that we are given positive data $\{\boldsymbol{x}^{\mathrm{P}}_i\}_{i=1}^{n_{\mathrm{P}}}$ and negative data $\{\boldsymbol{x}^{\mathrm{N}}_j\}_{j=1}^{n_{\mathrm{N}}}$ drawn from the class-conditional probability densities $p(\boldsymbol{x}|y=1)$ and $p(\boldsymbol{x}|y=-1)$, respectively. Having access to positive and negative data, the ERM framework suggests us to minimize the following empirical risk (Yan et al., 2003; Cortes and Mohri, 2004):

$$\hat{R}^{\ell}_{\mathrm{AUC}}(g) = \frac{1}{n_{\mathrm{P}} n_{\mathrm{N}}} \sum_{i=1}^{n_{\mathrm{P}}} \sum_{j=1}^{n_{\mathrm{N}}} \ell(g(\boldsymbol{x}^{\mathrm{P}}_i) - g(\boldsymbol{x}^{\mathrm{N}}_j)). \tag{2}$$

With the given training data and a surrogate loss $\ell$, we can minimize $\hat{R}^{\ell}_{\mathrm{AUC}}(g)$. This risk estimator $\hat{R}^{\ell}_{\mathrm{AUC}}(g)$ is an unbiased and consistent estimator of the true AUC risk (1), which means it converges to the true AUC risk as the number of data increases (Yan et al., 2003; Herschtal and Raskutti, 2004; Gao and Zhou, 2015).

Note that in practice, it is common to apply a regularization method for controlling the bias-variance trade-off of the estimator to avoid overfitting. In this paper, we adopt the ERM framework for minimizing the AUC risk. Note that one challenge of our problem is that we cannot apply the standard AUC risk directly since we have no access to any labeled data.

## 3 Problem Formulation

In this section, we formulate the problem of learning from keywords and unlabeled data. We are given a set of relevant keywords $W := \{w_j\}_{j=1}^{n_{\mathrm{W}}}$, and unlabeled documents drawn from the following distribution:

$$X_{\mathrm{U}} := \{\boldsymbol{x}^{\mathrm{U}}_i\}_{i=1}^{n_{\mathrm{U}}} \overset{\text{i.i.d.}}{\sim} p_{\pi}(\boldsymbol{x}),$$

where

$$p_{\pi}(\boldsymbol{x}) = \pi p(\boldsymbol{x}|y=1) + (1-\pi)\, p(\boldsymbol{x}|y=-1).$$

The evaluation metric defines the goal of this problem. For example, if we want a ranking function that outputs a large value for positive data, and small otherwise, the well-known evaluation metric is AUC (Agarwal et al., 2005). We will discuss other evaluation metrics, which are the accuracy and $\mathrm{F}_1$-measure in Section 4.3.

## 4 Proposed Framework

In this section, we propose a novel framework for handling this problem systematically. The framework consists of three parts. First, we use relevant

keywords to extract pseudo-positive data from the unlabeled data. Then, we employ a symmetric loss function for AUC optimization within the empirical risk minimization framework. Finally, we adjust the threshold of a trained AUC maximizer to further optimize other evaluation metrics. Figure 1 shows an overview of our framework.

### 4.1 Pseudo-labeling

Here, we propose a simple pseudo-labeling method. We want to emphasize that we do not expect that a pseudo-labeling algorithm is perfect, i.e., pseudo-positive documents may contain negative documents. In Section 4.2, we show that it is possible to maximize AUC effectively even though the labels are highly corrupted.

Since we focus on proposing a general framework that can be used in a wide range of text classification tasks, we use a simple pseudo-labeling method. In practice, one can incorporate prior knowledge such as the specific characteristic of documents to develop a sophisticated and potentially better pseudo-labeling algorithm to further enhance the performance. Our simple pseudo-labeling method is a ranking score based on the document similarities with the cosine similarity between unlabeled documents and keywords. Although we assume some keywords are given, it is hard to guarantee that the given keywords are sufficient to learn effectively. Hence, it is reasonable to sample more similar words to given keywords. In this paper, we used the GloVe model to find similar words (Pennington et al., 2014). With a sufficient number of keywords, we calculate the cosine similarity between the set of extended keywords set and each unlabeled document by simply merging all keywords and extended keywords into one document and measure the similarity between this keyword document and each unlabeled document. Finally, we pseudo-label documents to positive if their cosine similarities are in the top-$\phi$%.

**Algorithm 1:** A pseudo-labeling algorithm

**Input:** Unlabeled documents $\{\boldsymbol{x}_i^{\mathrm{U}}\}_{i=1}^{n_{\mathrm{U}}}$, keyword set $W := \{w_j\}_{j=1}^{n_{\mathrm{W}}}$, threshold $\phi \in [0, 100]$, weight of original keywords $\alpha \in \mathbb{N}$, sampling factor $\gamma \in \mathbb{N}$.

**Output:** two sets of pseudo-labeled documents $X_{\mathrm{CP}}$ and $X_{\mathrm{CN}}$

$X_{\mathrm{CP}} := \{\}$; $X_{\mathrm{CN}} := \{\}$

**foreach** $w_j$ **do**
  Add additional ($\alpha$-1) of $w_j$ to $W$;
  Find top-$\gamma$ similar words to $w_j$ $W_{sim} := \{w_j^i\}_{i=1}^{\gamma}$;
  Add $W_{sim}$ to $W$;
**end**

Compute similarity $s_i$ between one document that merges all keywords $W = \{w_j\}_{j=1}^{(\alpha+\gamma)n_{\mathrm{W}}}$ and each document $\boldsymbol{x}_i^{\mathrm{U}}$;

**foreach** $\boldsymbol{x}_i^{\mathrm{U}}$ **do**
  **if** $s_i > 0$ *and in the top-$\phi$(%)* **then**
    Add $\boldsymbol{x}_i^{\mathrm{U}}$ to $X_{\mathrm{CP}}$;
  **else**
    Add $\boldsymbol{x}_i^{\mathrm{U}}$ to $X_{\mathrm{CN}}$;
  **end**
**end**

**return** $X_{\mathrm{CP}}$, $X_{\mathrm{CN}}$

### 4.2 AUC Optimization from Corrupted Labels

After applying Algorithm 1, we can obtain two sets of data, which we call them a corrupted positive set and a corrupted negative set. Let us define $\theta$, $\theta' \in [0, 1]$, which indicate the proportion of positive data for the corrupted positive and negative sets, respectively. We assume a mild condition that our pseudo-labeling method can split the data such that the corrupted positive set has a higher proportion of positive data than another set, i.e., $\theta > \theta'$. Then, we formulate the data from the two sets as

$$X_{\mathrm{CP}} := \{\boldsymbol{x}_i^{\mathrm{CP}}\}_{i=1}^{n_{\mathrm{CP}}} \overset{\text{i.i.d.}}{\sim} p_\theta(\boldsymbol{x}),$$
$$X_{\mathrm{CN}} := \{\boldsymbol{x}_j^{\mathrm{CN}}\}_{j=1}^{n_{\mathrm{CN}}} \overset{\text{i.i.d.}}{\sim} p_{\theta'}(\boldsymbol{x}),$$

where

$$p_\theta(\boldsymbol{x}) := \theta p(\boldsymbol{x}|y=1) + (1-\theta)p(\boldsymbol{x}|y=-1),$$
$$p_{\theta'}(\boldsymbol{x}) := \theta' p(\boldsymbol{x}|y=1) + (1-\theta')p(\boldsymbol{x}|y=-1).$$

$X_{\mathrm{CP}}$ and $X_{\mathrm{CN}}$ are the sets of corrupted positive and corrupted negative data, respectively. We utilize the fact that optimizing AUC can be performed effectively even when the data is highly corrupted by using a symmetric loss function (Charoenphakdee et al., 2019). More specifically, let us consider a corrupted AUC risk. First, let us define $\mathbb{E}_\theta$ and $\mathbb{E}_{\theta'}$ as the expectation over $p_\theta$ and $p_{\theta'}$, respectively. A corrupted AUC risk is defined as

$$R^{\ell}_{\text{AUC-Corr}}(g) = \mathbb{E}_\theta[\mathbb{E}_{\theta'}[\ell(g(\boldsymbol{x}) - g(\boldsymbol{x}'))]]. \quad (3)$$

Intuitively, the risk (3) is minimized by a function $g$ that gives a higher value on corrupted positive data over corrupted negative data. The following theorem establishes the relationship between the clean AUC risk and corrupted AUC risk.

**Theorem 1** (Charoenphakdee et al. (2019))**.** *Let* $\varphi^{\ell}(\boldsymbol{x}, \boldsymbol{x}') = \ell(g(\boldsymbol{x}) - g(\boldsymbol{x}')) + \ell(g(\boldsymbol{x}') - g(\boldsymbol{x}))$. *Then* $R^{\ell}_{\text{AUC-Corr}}(g)$ *can be expressed as*

$$
\begin{aligned}
R^{\ell}_{\text{AUC-Corr}}(g) &= (\pi - \pi') R^{\ell}_{\text{AUC}}(g) \\
&+ \underbrace{(1-\pi)\pi' \mathbb{E}_{\mathrm{P}}[\mathbb{E}_{\mathrm{N}}[\varphi^{\ell}(\boldsymbol{x}^{\mathrm{P}}, \boldsymbol{x}^{\mathrm{N}})]]}_{\text{Excessive term}} \\
&+ \underbrace{\frac{\pi\pi'}{2} \mathbb{E}_{\mathrm{P}'}[\mathbb{E}_{\mathrm{P}}[\varphi^{\ell}(\boldsymbol{x}^{\mathrm{P}'}, \boldsymbol{x}^{\mathrm{P}})]]}_{\text{Excessive term}} \\
&+ \underbrace{\frac{(1-\pi)(1-\pi')}{2} \mathbb{E}_{\mathrm{N}'}[\mathbb{E}_{\mathrm{N}}[\varphi^{\ell}(\boldsymbol{x}^{\mathrm{N}'}, \boldsymbol{x}^{\mathrm{N}})]]}_{\text{Excessive term}}.
\end{aligned}
$$

Theorem 1 shows that minimizing the corrupted risk $R^{\ell}_{\text{AUC-Corr}}(g)$ implies minimizing both the clean risk $R^{\ell}_{\text{AUC}}(g)$ and excessive terms. However, since we only aim to minimize the clean risk, minimizing the corrupted risk may not effectively minimize the clean risk since excessive terms can be minimized instead and lead to overfitting. Charoenphakdee et al. (2019) showed that with a specific class of losses, excessive terms become constant. The following theorem suggests that by using a family of symmetric losses, the corrupted AUC risk can be expressed as an affine transformation of the clean AUC risk.

**Theorem 2** (Charoenphakdee et al. (2019))**.** *Let* $\ell_{\text{sym}}$ *be a symmetric loss such that* $\ell_{\text{sym}}(z) + \ell_{\text{sym}}(-z) = K$, *where* $K$ *is a positive constant. Then,* $R^{\ell_{\text{sym}}}_{\text{AUC-Corr}}(g)$ *can be expressed as*

$$
\begin{aligned}
R^{\ell_{\text{sym}}}_{\text{AUC-Corr}}(g) &= (\theta - \theta') R^{\ell_{\text{sym}}}_{\text{AUC}}(g) \\
&+ \frac{K(1 - \theta + \theta')}{2}.
\end{aligned}
$$

According to Theorem 2, we can optimize AUC by using a symmetric loss *without estimating* $\theta$ *and* $\theta'$ and do not suffer from excessive terms. This insight was also suggested by Menon et al. (2015), but they focused on more general losses, while it has been shown that using a symmetric loss can be preferable (Charoenphakdee et al., 2019). Our experiments also support that using a symmetric loss is preferable, while using a non-symmetric loss still provided reasonable performance. Similarly to AUC, we can also maximize the balanced accuracy effectively with a symmetric loss (see van Rooyen et al. (2015); Charoenphakdee et al. (2019) for more details on the balanced accuracy). Since the result of Charoenphakdee et al. (2019) only suggests that the minimizer of the corrupted risk and clean risk are identical, we provide a theory to explain why a more accurate pseudo-labeling algorithm can improve the performance by proving an estimation error bound in Section 5.

### 4.3 Optimizing Accuracy and $\mathrm{F}_1$-measure

Here, we review other evaluation metrics and discuss how to adjust our framework for optimizing other evaluation metrics. Many evaluation metrics can be optimized if a suitable threshold and $p(y = 1|\boldsymbol{x})$ are known (Yan et al., 2018), e.g., $\operatorname{sign}[p(y = 1|x) - \frac{1}{2}]$ is the Bayes-optimal solution for the accuracy. It is known that the Bayes-optimal solution of AUC maximization is any function that has a strictly monotonic relationship with $p(y = 1|\boldsymbol{x})$ (Menon and Williamson, 2016). Therefore, finding an appropriate threshold with an AUC maximizer can give an effective classifier (Narasimhan and Agarwal, 2013).

For the accuracy and $\mathrm{F}_1$-measure, we propose to optimize AUC first and then adjust the threshold to optimize other metrics. It is important to emphasize that a big challenge behind this problem is that *there is no information about the threshold from given data since all given data are unlabeled.*

Without additional assumptions, we argue that it is *theoretically impossible* to draw an optimal threshold to optimize the accuracy and $\mathrm{F}_1$-measure for this problem. Unlike supervised-learning, where we have positive and negative data, the threshold information is provided from labels and we can draw the decision boundary accordingly. On the other hand, one can learn a bipartite ranking function effectively by AUC optimization in our setting, which suggests the idea to learn a reliable ranking function first, and then adjust the threshold.

Suppose that we know the proportion of positive data $\pi$ in the training data and the class prior of the training and test data are identical. A reasonable threshold $\beta \in \mathbb{R}$ can be given as

$$
\pi = \int \operatorname{sign}(g(\boldsymbol{x}) - \beta) p_{\pi}(\boldsymbol{x}) d\boldsymbol{x}. \tag{4}
$$

Intuitively, we use a threshold $\beta$ that decides the top proportion $\pi$ as positives and negative oth-

erwise, which can optimize the accuracy effectively. This threshold is known as a precision-recall breakeven point, where it is the point where the precision equals to recall (see Kato et al. (2019) for its proof). Therefore, it is a reasonable threshold for the $\mathrm{F}_1$-measure since it is a harmonic mean of precision and recall. With unlabeled documents and $\pi$, we can decide $\beta$ that satisfies the empirical version of Eq. (4).

Existing methods, which may learn a threshold without knowing $\pi$, may actually learn a wrong threshold. Table 3 illustrates the failure of a default threshold when learning a classifier in noisy environments and Table 4 shows that much better performance on $\mathrm{F}_1$-measure and accuracy can be recover by adjusting the threshold with Eq. (4). Regarding the assumption that training and test class priors are identical, it is the common assumption if we want to use these metrics for evaluation and is implicitly assumed in the existing work (Jin et al., 2017). If the class priors are not identical, we suggest not to use these metrics since it can be misleading because a good $\mathrm{F}_1$-measure in training data cannot guarantee a good $\mathrm{F}_1$-measure in the test data. In real-world applications, such situations may occur when we do not collect unlabeled data directly from the test environment or the test environment is prone to be changed over time. If it is impossible to estimate $\pi$ or the test condition can be changed, we suggest to use other evaluation metrics such as AUC or precision at $k$.

## 5 Theoretical Analysis

In this section, we provide a theoretical guarantee for our method. Although it has been shown that the risk minimizer of corrupted data is identical to that of clean data (Charoenphakdee et al., 2019), the existing work cannot explain why a good pseudo-labeling algorithm can give a better performance. Throughout this section, we assume that an output of a loss function $\ell_{\mathrm{sym}}$ is bounded by $[0, K]$ without loss of generality for a nonnegative symmetric loss. Here, we prove an estimation error bound, which suggests that a larger gap between $\theta - \theta'$ leads to a tighter bound and thus faster convergence. Note that the gap is largest when we have clean positive data (i.e., $\theta = 1$) and clean negative data (i.e., $\theta' = 0$). Therefore, a good pseudo-labeling is an algorithm that can divide the data with a large gap (i.e., $\theta - \theta'$ is large). All proofs can be found in Appendix A.

Let $\hat{g} \in \mathcal{G}$ be a minimizer of the empirical corrupted AUC risk $\widehat{R}^{\ell_{\mathrm{sym}}}_{\text{AUC-Corr}}$ in the hypothesis class $\mathcal{G}$. Let $g^* \in \mathcal{G}$ be a minimizer of the expected risk of clean AUC risk $R^{\ell_{\mathrm{sym}}}_{\mathrm{AUC}}$. Then, the following lemma establishes the relationship between an estimation error bound of the AUC risk and the uniform deviation bound of the corrupted AUC risk.

**Lemma 3.** *An estimation error bound of the clean AUC risk $R^{\ell}_{\mathrm{AUC}}$ can be given as follows.*

$$R^{\ell_{\mathrm{sym}}}_{\mathrm{AUC}}(\hat{g}) - R^{\ell_{\mathrm{sym}}}_{\mathrm{AUC}}(g^*) \leq \frac{2}{\theta - \theta'} \sup_{g \in \mathcal{G}} |R^{\ell_{\mathrm{sym}}}_{\text{AUC-Corr}}(g) - \widehat{R}^{\ell_{\mathrm{sym}}}_{\text{AUC-Corr}}(g)|.$$

Next, let $\mathcal{Q}^{\ell_{\mathrm{sym}}}_{\mathcal{G}}$ be a class of functions mapping $\mathcal{X}^2$ to $[0, K]$ such that $\mathcal{Q}^{\ell_{\mathrm{sym}}}_{\mathcal{G}} = \{Q : (\boldsymbol{x}, \boldsymbol{x}') \to \ell_{\mathrm{sym}}(g(\boldsymbol{x}) - g(\boldsymbol{x}')), g \in \mathcal{G}\}$. Given samples $(x_1, \ldots, x_n) \in \mathcal{X}^n$ and $(x'_1, \ldots, x'_m) \in \mathcal{X}^m$ independently and identically drawn from a distribution with densities $\mu$ and $\mu'$, respectively. The empirical bipartite Rademacher complexity (Usunier et al., 2005) of a function class $\mathcal{Q}^{\ell_{\mathrm{sym}}}_{\mathcal{G}}$ is defined as

$$\hat{\mathfrak{R}}_{n,m}(\mathcal{Q}^{\ell_{\mathrm{sym}}}_{\mathcal{G}}) = 2\mathbb{E}_{\sigma,\upsilon} \sup_{Q \in \mathcal{Q}^{\ell_{\mathrm{sym}}}_{\mathcal{G}}} \frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{\sigma_i + \upsilon_j}{2} \left[Q(\boldsymbol{x}_i, \boldsymbol{x}'_j)\right],$$

where the inner expectation is taken over $\sigma = (\sigma_1, \ldots, \sigma_n)$ and $\upsilon = (\upsilon_1, \ldots, \upsilon_m)$ which are independent random variables taking values in $\{-1, +1\}$ uniformly.

Then, the bipartite Rademacher complexity can be defined as

$$\mathfrak{R}_{n,m}(\mathcal{Q}^{\ell_{\mathrm{sym}}}_{\mathcal{G}}) = \mathbb{E}_{x_1,\ldots,x_n,x'_1,\ldots,x'_m} \hat{\mathfrak{R}}_{n,m}(\mathcal{Q}^{\ell_{\mathrm{sym}}}_{\mathcal{G}}).$$

Let $\mathfrak{R}^{\mathrm{Corr}}_{n_{\mathrm{CP}},n_{\mathrm{CN}}}$ be the bipartite Rademacher complexities of pseudo-labeled data where $\mu = p_\theta$ and $\mu' = p'_\theta$. Next, we provide a uniform deviation bound of the corrupted AUC risk.

**Lemma 4** (Uniform deviation bound)**.** *For all $Q \in \mathcal{Q}^{\ell_{\mathrm{sym}}}_{\mathcal{G}}$ and $\delta \in (0, 1)$, with probability at least $1 - \delta$, we have*

$$\sup_{g \in \mathcal{G}} \left| R^{\ell_{\mathrm{sym}}}_{\text{AUC-Corr}}(g) - \widehat{R}^{\ell_{\mathrm{sym}}}_{\text{AUC-Corr}}(g) \right| \leq K \sqrt{\frac{(n_{\mathrm{CP}} + n_{\mathrm{CN}}) \log \frac{1}{\delta}}{2 n_{\mathrm{CP}} n_{\mathrm{CN}}}} + 2\mathfrak{R}^{\mathrm{Corr}}_{n_{\mathrm{CP}},n_{\mathrm{CN}}}(\mathcal{Q}^{\ell_{\mathrm{sym}}}_{\mathcal{G}}).$$

*where the probability is over repeated sampling of data for evaluating $\widehat{R}^{\ell}_{\text{AUC-Corr}}(g)$.*

Table 2: Keywords for each dataset

| Dataset | Keywords |
|---|---|
| Subj | wonderful terrible feel happy ugly even horrible interesting funny dramatic romantic compassionate |
| Custrev | easy excellent nice great good love amazing best awesome perfect definitely better happy compassionate |
| MPQA | support hope help good great love |
| AYI | great best excellent friendly awesome nice amazing |
| 20NG | sports baseball hockey |

Then, by combining Lemma 3 and Lemma 4, we obtain the following theorem.

**Theorem 5** (Estimation error bound)**.** *For all* $Q \in \mathcal{Q}_{\mathcal{G}}^{\ell_{\mathrm{sym}}}$ *and* $\delta \in (0,1)$*, with probability at least* $1-\delta$*, we have*

$$R_{\mathrm{AUC}}^{\ell_{\mathrm{sym}}}(\hat{g}) - R_{\mathrm{AUC}}^{\ell_{\mathrm{sym}}}(g^*) \leq \frac{1}{\theta-\theta'}\left[K\sqrt{\frac{2(n_{\mathrm{CP}}+n_{\mathrm{CN}})\log\frac{1}{\delta}}{n_{\mathrm{CP}}n_{\mathrm{CN}}}}\right] + \frac{4\mathfrak{R}^{\mathrm{Corr}}_{n_{\mathrm{CP}},n_{\mathrm{CN}}}(\mathcal{Q}_{\mathcal{G}}^{\ell_{\mathrm{sym}}})}{\theta-\theta'},$$

*where the probability is over repeated sampling of* $X_{\mathrm{CP}}$ *and* $X_{\mathrm{CN}}$ *for training* $\hat{g}$*.*

Theorem 5 shows that a gap $\frac{1}{\theta-\theta'}$ affects the tightness of the bound. When $\theta$ and $\theta'$ are close to each other, i.e., when the data is highly corrupted, the bound becomes loose. This illustrates the difficulty of the task when pseudo-labeling algorithm performs poorly and we may need more data. Nevertheless, as long as $\theta > \theta'$, with all parametric models with their norm is bounded such as neural networks with weight decay or kernel model, our learning framework is consistent, i.e., the estimation error converges to zero as $n_{\mathrm{CP}}, n_{\mathrm{CN}} \to \infty$.

## 6 Experimental Results

We present experimental results in this section with evaluation metrics include AUC, macro $\mathrm{F}_1$-measure, accuracy (ACC) and precision at 100 (Prec@100). Prec@100 is the ratio of the true positive data over in the top-100 ranking score.

### 6.1 Experiment Setup

**Datasets:** We used five datasets, namely the Subjectivity dataset (Pang and Lee, 2004) (Subj), Customer reviews (Hu and Liu, 2004) (Custrev), Opinion mining in MPQA corpus (MPQA), Product reviews from Amazon, Yelp, and IMDb (AYI), 20 Newsgroups (Lang, 1995) (20NG) with *baseball* and *hockey* groups as positive. More information on the datasets can be found in Appendix B.

Table 3: Failure of optimizing $\mathrm{F}_1$-measure if thresholds are not adjusted. Full results can be found in Appendix C. By adjusting the threshold in Table 4 can substantially improve the performance.

| Methods | Subj | MPQA | AYI | 20NG |
|---|---|---|---|---|
| Maxent | 63.4 (0.31) | 50.1 (0.22) | 42.5 (0.35) | 47.4 (0.05) |
| NB | 73.7 (0.23) | 53.8 (0.22) | 65.8 (0.42) | 23.7 (0.25) |
| RandomForest | 33.3 (0.00) | 43.5 (0.20) | 35.0 (0.20) | 47.2 (0.00) |
| KNN | 43.6 (0.23) | 51.0 (0.16) | 61.6 (0.43) | 84.3 (0.26) |

Keywords for each dataset are shown in Table 2.

**Common setup:** First, we need to pseudo-label documents. We used Algorithm 1 to feed pseudo-labeled documents to all methods, where $\phi$ is set to 90. ($\alpha$, $\gamma$) were set to (3,5) for all datasets except Subj, which was (1,50). We used 50-dimensional features for GloVe word embeddings. For ACC and $\mathrm{F}_1$, we gave true thresholds $\pi$ to all methods to see the top performance of each method. We also provide the results with varying thresholds and heuristic thresholds in Appendix C, where the trends of performance for each method do not differ much from Table 4. Note that AUC and Prec@100 do not use a threshold to evaluate. The experimental results are reported in the mean value and standard error of 20 trials. "N/A" indicates that an algorithm is not terminated due to too many number of vocabularies.

**Baselines:** We compared our method with three categories of baselines: the text-feature, GloVe-feature, and zero-shot baselines. For the text-feature baselines, we use naive Bayes (NB), the maximum entropy model (Maxent), and naive Bayes that is implemented for learning from positive and unlabeled data (PU-NB). All implementations were from Natural Language Toolkit (NLTK) (Loper and Bird, 2002). For the GloVe-feature baselines, we used mean word vectors as features and employed a random forest (RandomForest) and K-nearest neighbors (KNN), which were implemented by Scikit-learn (Pedregosa et al., 2011). Finally, the zero-shot baselines did not use unlabeled data but simply GloVe to rank the score of a document (GloVeRanking) and keyword voting (Voting). We also showed the performance when fully-labeled data are given as references. O-Maxent is a maxent model with fully-labeled data and O-Sigmoid is our framework that skips the pseudo-labeling step and uses fully-labeled data for AUC optimization.

**Proposed methods:** For the AUC optimization part, we used the recurrent convolutional

Table 4: Mean value and standard error of 20 trials for the AUC, $F_1$-measure, accuracy (ACC), and precision at 100 (Prec@100) of learning from relavant keywords and unlabeled documents. Outperforming methods are highlighted in boldface using one-sided t-test with the significance level of 5%. The sigmoid loss is symmetric while the logistic loss is non-symmetric.

| Dataset | Evaluation | Proposed framework | | Text-feature baselines | | | GloVe-feature baselines | | Zero-shot baselines | | Oracle | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Sigmoid | Logistic | PU-NB | NB | Maxent | RandomForest | KNN | GloVeRanking | Voting | O-Maxent | O-Sigmoid |
| Subj | AUC | **88.1 (0.35)** | 84.1 (0.30) | 55.4 (0.13) | 85.0 (0.18) | 84.6 (0.20) | 82.4 (0.27) | 73.6 (0.29) | 81.7 (0.19) | 70.2 (0.24) | 97.4 (0.06) | 93.6 (0.11) |
| | $F_1$ | **80.1 (0.38)** | 76.0 (0.32) | 47.1 (0.20) | 76.3 (0.16) | 76.3 (0.24) | 75.1 (0.27) | 63.6 (0.32) | 74.5 (0.13) | 63.5 (0.18) | 92.0 (0.13) | 86.4 (0.14) |
| | ACC | **80.1 (0.38)** | 76.0 (0.32) | 55.0 (0.13) | 76.3 (0.16) | 76.3 (0.24) | 75.1 (0.27) | 65.0 (0.28) | 74.5 (0.13) | 64.1 (0.18) | 92.0 (0.13) | 86.4 (0.14) |
| | Prec@100 | **96.3 (0.60)** | **95.1 (0.60)** | 0.9 (0.09) | **95.9 (0.33)** | 94.7 (0.39) | 93.2 (0.50) | 91.5 (0.59) | **95.2 (0.54)** | 85.8 (0.93) | 99.3 (0.15) | 97.8 (0.27) |
| Custrev | AUC | 71.2 (0.34) | 70.7 (0.34) | 59.7 (0.32) | **74.2 (0.35)** | 69.9 (0.46) | 62.6 (0.47) | 67.9 (0.40) | 55.3 (0.39) | 67.4 (0.32) | 75.0(0.38) | 78.5(0.30) |
| | $F_1$ | **63.6 (0.41)** | **63.1 (0.31)** | 58.4 (0.35) | **63.3 (0.49)** | 62.1 (0.43) | 58.3 (0.44) | 60.8 (0.33) | 53.0 (0.40) | 38.9 (0.00) | 68.5 (0.42) | 70.2 (0.29) |
| | ACC | 66.5 (0.40) | 66.0 (0.31) | 64.8 (0.32) | **69.8 (0.37)** | 65.8 (0.39) | 61.9 (0.40) | 66.6 (0.29) | 56.6 (0.37) | 63.7 (0.00) | 71.4 (0.41) | 72.5 (0.28) |
| | Prec@100 | 91.2 (0.46) | 91.5 (0.49) | **99.2 (0.18)** | 89.8 (0.55) | 91.2 (0.57) | 80.9 (0.49) | 86.3 (0.85) | 75.1 (0.81) | 87.3 (0.66) | 82.2 (0.35) | 86.9 (0.28) |
| MPQA | AUC | **80.4 (0.44)** | 78.7 (0.37) | 52.1 (0.27) | 56.4 (0.31) | 56.7 (0.23) | 69.1 (0.55) | 60.1 (0.23) | 63.6 (0.26) | 56.0 (0.12) | 78.3 (0.25) | 86.8 (0.18) |
| | $F_1$ | **71.7 (0.44)** | 69.8 (0.31) | 46.7 (0.23) | 54.3 (0.28) | 53.1 (0.20) | 62.4 (0.45) | 23.8 (0.00) | 57.5 (0.17) | 23.8 (0.00) | 69.8 (0.19) | 77.9 (0.22) |
| | ACC | **75.6 (0.39)** | 74.0 (0.27) | 47.1 (0.24) | 62.4 (0.28) | 58.4 (0.17) | 67.4 (0.39) | 31.2 (0.00) | 63.3 (0.17) | 31.2 (0.00) | 72.8 (0.20) | 81.0 (0.19) |
| | Prec@100 | **81.5 (0.97)** | 77.5 (1.02) | 10.8 (3.37) | 69.5 (0.86) | 63.8 (1.93) | 76.9 (1.06) | 78.7 (0.80) | 50.6 (0.60) | 74.7 (0.69) | 94.8 (0.46) | 90.5 (0.52) |
| AYI | AUC | **76.0 (0.41)** | **75.6 (0.43)** | 60.5 (0.39) | 71.2 (0.41) | 60.7 (0.46) | 70.1 (0.55) | 72.5 (0.39) | 62.4 (0.53) | 61.0 (0.33) | 84.6 (0.32) | 81.1 (0.40) |
| | $F_1$ | **69.3 (0.36)** | **68.8 (0.40)** | 58.9 (0.47) | 61.6 (0.38) | 56.6 (0.36) | 64.5 (0.53) | 65.5 (0.52) | 58.7 (0.51) | 33.5 (0.00) | 76.8 (0.37) | 73.0 (0.39) |
| | ACC | **69.3 (0.36)** | **68.8 (0.40)** | 60.1 (0.41) | 62.5 (0.35) | 56.8 (0.36) | 64.6 (0.53) | 65.8 (0.44) | 58.7 (0.51) | 50.5 (0.00) | 76.9 (0.37) | 73.0 (0.39) |
| | Prec@100 | **87.5 (0.55)** | **87.5 (0.62)** | 74.5 (2.20) | 85.1 (0.71) | 70.2 (1.00) | 77.2 (0.99) | 82.5 (0.69) | 72.4 (0.91) | 79.2 (0.87) | 95.6 (0.47) | 90.1 (0.73) |
| 20NG | AUC | 96.4 (0.12) | 96.0 (0.15) | N/A | 77.1 (0.21) | 57.6 (0.32) | **96.8 (0.16)** | 94.7 (0.16) | 95.0 (0.17) | 62.9 (0.22) | 65.5 (0.46) | 99.0 (0.05) |
| | $F_1$ | **90.8 (0.20)** | **90.6 (0.21)** | N/A | 58.4 (0.22) | 52.4 (0.25) | 89.6 (0.28) | 86.7 (0.59) | **90.5 (0.18)** | 9.6 (0.00) | 56.8 (0.29) | 94.1 (0.14) |
| | ACC | **96.5 (0.08)** | **96.4 (0.08)** | N/A | 70.2 (0.31) | 81.6 (0.11) | 96.1 (0.10) | 94.5 (0.35) | **96.4 (0.07)** | 10.6 (0.00) | 83.5 (0.11) | 97.8 (0.05) |
| | Prec@100 | **99.5 (0.15)** | **99.1 (0.24)** | N/A | 0.4 (0.11) | 17.6 (0.77) | **99.5 (0.15)** | 97.6 (0.38) | 97.5 (0.28) | 85.2 (1.03) | 32.0 (1.31) | 99.9 (0.07) |

Table 5: Mean $F_1$-measure and standard error of 20 trials with varying thresholds with different $\hat{\pi}$ in Eq. (4).

| Dataset | Methods | 0.05 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Subj | Sigmoid | 43.3 (0.17) | 51.2 (0.24) | 63.9 (0.34) | 72.5 (0.27) | 77.9 (0.37) | **80.1 (0.38)** | 78.7 (0.32) | 73.7 (0.29) | 64.7 (0.32) | 51.8 (0.30) |
| | Maxent | 37.1 (0.15) | 41.9 (0.18) | 52.5 (0.27) | 62.1 (0.19) | 70.6 (0.27) | **76.3 (0.24)** | 74.5 (0.25) | 64.1 (0.25) | 50.3 (0.22) | 39.3 (0.13) |
| | RandomForest | 42.1 (0.22) | 49.9 (0.15) | 61.6 (0.22) | 69.1 (0.26) | 73.5 (0.29) | **75.1 (0.27)** | 73.5 (0.17) | 69.0 (0.21) | 61.2 (0.24) | 49.6 (0.23) |
| | GloVe Ranking | 43.1 (0.24) | 50.5 (0.25) | 61.9 (0.25) | 69.3 (0.27) | 73.6 (0.18) | **74.5 (0.13)** | 72.8 (0.18) | 68.0 (0.16) | 60.6 (0.23) | 49.4 (0.22) |
| 20NG | Sigmoid | 79.9 (0.23) | **91.2 (0.18)** | 78.4 (0.20) | 68.2 (0.15) | 60.0 (0.14) | 52.7 (0.13) | 45.3 (0.15) | 37.8 (0.13) | 29.7 (0.13) | 20.5 (0.13) |
| | Maxent | 51.5 (0.19) | **52.3 (0.24)** | 51.5 (0.21) | 49.5 (0.16) | 46.6 (0.15) | 43.0 (0.16) | 38.1 (0.14) | 32.9 (0.15) | 26.3 (0.16) | 19.0 (0.13) |
| | RandomForest | 79.4 (0.33) | **89.8 (0.29)** | 78.5 (0.17) | 68.2 (0.15) | 59.8 (0.16) | 52.6 (0.13) | 45.4 (0.12) | 37.7 (0.16) | 29.6 (0.13) | 20.3 (0.10) |
| | GloVe Ranking | 79.2 (0.34) | **90.7 (0.17)** | 78.2 (0.22) | 67.9 (0.19) | 59.9 (0.14) | 52.3 (0.11) | 45.0 (0.10) | 37.6 (0.10) | 29.3 (0.11) | 20.1 (0.13) |

neural networks (RCNN) model (Lai et al., 2015) with two layer long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997). We used Adam (Kingma and Ba, 2014) as an optimization method. We used the symmetric sigmoid loss (Sigmoid). We also show the results of the non-symmetric logistic loss (Logistic) to validate the improvement from using a symmetric loss. Implementation details are provided in Appendix B.

### 6.2 End-to-end Classification Performance

Table 4 shows the classification performance. It can be observed that our proposed framework outperforms other baselines in many cases. Moreover, the sigmoid loss (Sigmoid) outperforms the logstic loss (Logistic), which agrees with Theorem 2. Nevertheless, the performance of the logistic loss is desirable compared with baselines. It can be observed that GloveRanking, which does not use unlabeled data can outperform many baselines in 20NG. This can be due to the fact that pseudo-labeling data were corrupted and degraded the performance of a classifier heavily. In Table 4, for MPQA and 20NG, our proposed framework with symmetric losses is able to outperform O-Maxent in the $F_1$-measure, AUC score, and accuracy without having access to labels. One possible reason that O-Maxent does not perform well on 20NG is class imbalance.

Table 5 shows the accuracy and $F_1$-measure with varying thresholds. The true class prior of Subj and 20NG were 0.5 and 0.11, respectively. It can be observed that the threshold based on Eq. (4) gives a desirable performance when $\hat{\pi}$ is close to $\pi$. For Prec@100, we can see that Prec@100 for 20NG for the sigmoid loss in our framework and RandomForest is 99.5, without any labeled data, which indicates that we may pick top-100 documents as positive with almost perfect precision.

## 7 Conclusion

We proposed a theoretically-grounded framework for learning from relevant keywords and unlabeled documents. Our framework is highly flexible and can guarantee any heuristic pseudo-labeling method as long as an algorithm can divide unlabeled documents into two sets of data with different proportions of positive data. Experiments showed the usefulness of the proposed method.

## Acknowledgement

NC was supported by MEXT scholarship and MS was supported by JST CREST Grant Number JPMJCR1403. Part of the research was done while NC was visiting Chulalongkorn University with the support of U. Tokyo IST Research Internship Program.

## References

Shivani Agarwal, Thore Graepel, Ralf Herbrich, Sariel Har-Peled, and Dan Roth. 2005. Generalization bounds for the area under the roc curve. *JMLR*, 6(Apr):393–425.

Peter L Bartlett, Michael I Jordan, and Jon D McAuliffe. 2006. Convexity, classification, and risk bounds. *JASA*, 101(473):138–156.

Shai Ben-David, Nadav Eiron, and Philip M Long. 2003. On the difficulty of approximately maximizing agreements. *Journal of Computer and System Sciences*, 66(3):496–514.

Olivier Bousquet, Stéphane Boucheron, and Gábor Lugosi. 2003. Introduction to statistical learning theory. In *Summer School on Machine Learning*, pages 169–207. Springer.

Ming-Wei Chang, Lev-Arie Ratinov, Dan Roth, and Vivek Srikumar. 2008. Importance of semantic representation: Dataless classification. In *AAAI*, volume 2, pages 830–835.

Nontawat Charoenphakdee, Jongyeong Lee, and Masashi Sugiyama. 2019. On symmetric losses for learning from corrupted labels. *ICML*.

Xingyuan Chen, Yunqing Xia, Peng Jin, and John Carroll. 2015. Dataless text classification with descriptive lda. In *AAAI*.

Aaron M Cohen and William R Hersh. 2005. A survey of current work in biomedical text mining. *Briefings in bioinformatics*, 6(1):57–71.

Corinna Cortes and Mehryar Mohri. 2004. Auc optimization vs. error rate minimization. In *NeurIPS*, pages 313–320.

Vitaly Feldman, Venkatesan Guruswami, Prasad Raghavendra, and Yi Wu. 2012. Agnostic learning of monomials by halfspaces is hard. *SIAM Journal on Computing*, 41(6):1558–1590.

George Forman. 2003. An extensive empirical study of feature selection metrics for text classification. *JMLR*, 3(Mar):1289–1305.

Wei Gao and Zhi-Hua Zhou. 2015. On the consistency of auc pairwise optimization. In *IJCAI*, pages 939–945.

Aritra Ghosh, Naresh Manwani, and PS Sastry. 2015. Making risk minimization tolerant to label noise. *Neurocomputing*, 160:93–107.

James A Hanley and Barbara J McNeil. 1982. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36.

Alan Herschtal and Bhavani Raskutti. 2004. Optimising area under the roc curve using gradient descent. In *ICML*, page 49. ACM.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, pages 1735–1780.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *KDD*, pages 168–177. ACM.

Xia Hu and Huan Liu. 2012. Text analytics in social media. In *Mining text data*, pages 385–414. Springer.

Chung-Chi Huang and Zhiyong Lu. 2015. Community challenges in biomedical text mining over 10 years: success, failure and the future. *Briefings in bioinformatics*, 17(1):132–144.

Yiping Jin, Dittaya Wanvarie, and Phu Le. 2017. Combining lightly-supervised text classification models for accurate contextual advertising. In *IJCNLP*, volume 1, pages 545–554.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Masahiro Kato, Takeshi Teshima, and Junya Honda. 2019. Learning from positive and unlabeled data with a selection bias. *ICLR*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *ICLR*.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *AAAI*.

Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *ICML*, pages 331–339.

Ximing Li and Bo Yang. 2018. A pseudo label based dataless naive bayes algorithm for text classification with seed words. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1908–1917.

Bing Liu and Lei Zhang. 2012. A survey of opinion mining and sentiment analysis. In *Mining text data*, pages 415–463. Springer.

Edward Loper and Steven Bird. 2002. Nltk: the natural language toolkit. *arXiv preprint cs/0205028*.

Henry B Mann and Donald R Whitney. 1947. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, pages 50–60.

Colin McDiarmid. 1989. On the method of bounded differences. *Surveys in Combinatorics*, 141(1):148–188.

Walaa Medhat, Ahmed Hassan, and Hoda Korashy. 2014. Sentiment analysis algorithms and applications: A survey. *Ain Shams engineering journal*, 5(4):1093–1113.

Aditya Menon, Brendan van Rooyen, Cheng Soon Ong, and Bob Williamson. 2015. Learning from corrupted binary labels via class-probability estimation. In *ICML*, pages 125–134.

Aditya Krishna Menon and Robert C Williamson. 2016. Bipartite ranking: a risk-theoretic perspective. *JMLR*, 17(1):6766–6867.

Harikrishna Narasimhan and Shivani Agarwal. 2013. On the relationship between binary classification, bipartite ranking, and binary class probability estimation. In *NeurIPS*, pages 2913–2921.

Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. 2000. Text classification from labeled and unlabeled documents using em. *Machine learning*, 39(2-3):103–134.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity. In *ACL*, pages 271–278.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in Python. *JMLR*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. 2018. On the convergence of Adam and beyond. In *ICLR*.

Brendan van Rooyen, Aditya Krishna Menon, and Robert C Williamson. 2015. An average classification algorithm. *arXiv preprint arXiv:1506.01520*.

Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47.

Yangqiu Song and Dan Roth. 2014. On dataless hierarchical text classification. In *AAAI*.

Nicolas Usunier, Massih-Reza Amini, and Patrick Gallinari. 2005. A data-dependent generalisation error bound for the auc. In *ICML Workshop*.

Vladimir Vapnik. 1998. *Statistical learning theory. 1998*, volume 3. Wiley, New York.

Bowei Yan, Oluwasanmi Koyejo, Kai Zhong, and Pradeep Ravikumar. 2018. Binary classification with karmic, threshold-quasi-concave metrics. *arXiv preprint arXiv:1806.00640*.

Lian Yan, Robert H Dodier, Michael Mozer, and Richard H Wolniewicz. 2003. Optimizing classifier performance via an approximation to the wilcoxon-mann-whitney statistic. In *ICML*, pages 848–855.

Daniel Zeng, Hsinchun Chen, Robert Lusch, and Shu-Hsing Li. 2010. Social media analytics and intelligence. *IEEE Intelligent Systems*, 25(6):13–16.

Tong Zhang. 2004. Statistical behavior and consistency of classification methods based on convex risk minimization. *Annals of Statistics*, pages 56–85.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NeurIPS*, pages 649–657.

Zhi-Hua Zhou. 2017. A brief introduction to weakly supervised learning. *National Science Review*, 5(1):44–53.

# Denoising based Sequence-to-Sequence Pre-training for Text Generation

**Liang Wang**[1], **Wei Zhao**[1], **Ruoyu Jia**[1], **Sujian Li**[2], **Jingming Liu**[1]
[1]Yuanfudao AI Lab, Beijing, China
[2]Key Laboratory of Computational Linguistics, Peking University, MOE, China
{wangliang01,zhaowei01,jiary,liujm}@fenbi.com
lisujian@pku.edu.cn

## Abstract

This paper presents a new sequence-to-sequence (seq2seq) pre-training method PoDA (**P**re-training **o**f **D**enoising **A**utoencoders), which learns representations suitable for text generation tasks. Unlike encoder-only (e.g., BERT) or decoder-only (e.g., OpenAI GPT) pre-training approaches, PoDA jointly pre-trains both the encoder and decoder by denoising the noise-corrupted text, and it also has the advantage of keeping the network architecture unchanged in the subsequent fine-tuning stage. Meanwhile, we design a hybrid model of Transformer and pointer-generator networks as the backbone architecture for PoDA. We conduct experiments on two text generation tasks: abstractive summarization, and grammatical error correction. Results on four datasets show that PoDA can improve model performance over strong baselines without using any task-specific techniques and significantly speed up convergence. [1]

## 1 Introduction

Methods based on unsupervised pre-training and supervised fine-tuning for NLP have achieved phenomenal successes in the last two years. Most of the proposed methods in the literature choose language modeling or its variant as the pre-training task. After the pre-training stage, ELMo (Peters et al., 2018) and CoVe (McCann et al., 2017) directly use the learned representations as additional features for downstream tasks, while BERT (Devlin et al., 2018), ULMFiT (Howard and Ruder, 2018), XLM (Lample and Conneau, 2019), and OpenAI GPT (Radford et al., 2018, 2019) require fine-tuning both pre-trained parameters and task-specific parameters on labeled data. The state-of-the-art performances have been significantly advanced for classification and sequence labeling tasks, such as natural language inference (Bowman et al., 2015), named-entity recognition, SQuAD question answering (Rajpurkar et al., 2016) etc.

However, little attention has been paid to pre-training for seq2seq text generation (Sutskever et al., 2014). A typical seq2seq network consists of a bidirectional encoder, a unidirectional decoder and attention between the encoder and decoder. Previous work mainly focuses on encoder-only or decoder-only pre-training. For example, BERT pre-trains a bidirectional encoder, and OpenAI GPT pre-trains a language model which is essentially a unidirectional decoder. Ramachandran et al. (2016) propose to train two independent language models for the encoder and decoder respectively. All of the aforementioned methods are only able to partially pre-train the seq2seq networks, and therefore are unable to unleash the full potential of transfer learning for text generation.

In this paper, we present PoDA, a denoising based pre-training method that is able to jointly pre-train all components of seq2seq networks. Like denoising autoencoders, PoDA works by denoising the noise-corrupted text sequences. Any noising function that fits in the seq2seq framework can be used. We experiment with three types of noises: randomly shuffle, delete or replace the words in a given sequence. It is noted PoDA is simple, easy-to-implement and applicable to virtually all seq2seq architectures, including ConvS2S (Gehring et al., 2017) and Transformer (Vaswani et al., 2017). Here, we adopt the hybrid architecture of Transformer and pointer-generator networks (See et al., 2017). Transformer is effective at modeling long-distance dependencies, highly parallelizable and demonstrates good performance empirically. Pointer-generator network incorporates copying mechanism (Gu et al., 2016; Gulcehre et al., 2016) which is helpful for most text

[1]The code and pre-trained models are available at https://github.com/yuantiku/PoDA.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4003–4015,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

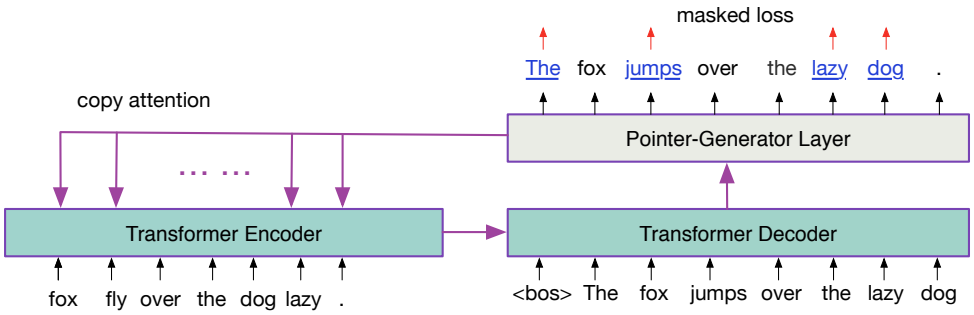


Figure 1: PoDA model architecture. The masked loss is calculated only for the blue underlined words. "<bos>" is a special begin-of-sequence padding symbol. The example input-output pair is explained in Section 2.2.

generation tasks.

The text corpora used for pre-training are the Billion Word Benchmark (Chelba et al., 2013) and English Wikipedia, both of which are publicly available and consists of nearly 2.99 billion words in total. We conduct experiments on two abstractive summarization datasets (CNN/Daily Mail (See et al., 2017) and Gigaword (Rush et al., 2015)), and two grammatical error correction datasets (CoNLL-2014 (Ng et al., 2014) and JFLEG (Napoles et al., 2017)). With simple maximum likelihood training and no task-specific techniques, PoDA achieves superior or comparable performance against state-of-the-art systems and speeds up convergence for all four datasets.

## 2 Method

### 2.1 Model Architecture

First, we design a seq2seq model as the backbone architecture of our proposed pre-training method, which is a combination of Transformer and pointer-generator networks, as shown in Figure 1.

The input representations are the sum of word embeddings and sinusoidal positional encodings. Both the Transformer encoder and the decoder consist of 6 layers of transformer blocks, and each block is a multi-head self-attention layer followed by one layer of positionwise feedforward network.

For the output layer, we use a pointer-generator layer to allow both copying from the input sequence and generation from a fixed vocabulary. The implementation is detailed in Appendix.

As a side note, we want to point out that the seq2seq architecture is not limited to the one we propose and other networks such as ConvS2S, RNN-based seq2seq models are also applicable. Pointer-generator networks are also not the only solution for handling out-of-vocabulary(OOV) words, and subword-based methods such as sentencepiece (Kudo and Richardson, 2018) can be used at the cost of making the input and output sequences longer.

### 2.2 Noising and Denoising

Similar to denoising autoencoders, PoDA involves two parts: noising and denoising. The noising part corrupts a given word sequence $\mathbf{x} = \{\mathbf{x}_i\}_{i=1}^{n}$ and gets a noisy word sequence $\mathbf{x}' = \{\mathbf{x}'_i\}_{i=1}^{n'}$. The denoising part tries to recover $\mathbf{x}$ given $\mathbf{x}'$ using a seq2seq model.

We use three noising functions: randomly shuffle, delete or replace the words in $\mathbf{x}$. The details are shown in Algorithm 1, where $N(0, \sigma)$ is a gaussian distribution with mean 0 and variance $\sigma$. $B(p)$ is a Bernoulli distribution, and $Beta(\alpha, \beta)$ is a beta distribution serving as the prior for $B(p)$. Take function `DELETE` (line 10 to line 15 in Algorithm 1) as an example, it first samples a Bernoulli distribution with expectation $p$ from $Beta(\alpha, \beta)$, then each word is deleted with probability $p$. The usage of $Beta(\alpha, \beta)$ prior can make the model robust to different degrees of noises.

We exemplify the operations above in Figure 1. The original word sequence $\mathbf{x}$ = *"The fox jumps over the lazy dog ."*, after three noising operations: delete *"The"*, replace *"jumps"* with *"fly"* and swap *"lazy"* and *"dog"*, we get the noisy word sequence $\mathbf{x}'$ = *"fox fly over the dog lazy ."*.

The denoising part maximizes the conditional probability $p(\mathbf{x}|\mathbf{x}')$, which can be factorized as:

$$p(\mathbf{x}|\mathbf{x}') = \Pi_{i=1}^{n} p(\mathbf{x}_i|\mathbf{x}', \mathbf{x}_{<i}) \quad (1)$$

When predicting $\mathbf{x}_i$, it is conditioned on the noise-corrupted full context $\mathbf{x}'$ and the clean left

**Algorithm 1** The Noising Algorithm

**Input**: $\mathbf{x}$ is a sequence of words
$\alpha, \beta, \sigma$ are hyperparameters

**function** NOISING($\mathbf{x}$)
  Apply SHUFFLE, DELETE, REPLACE to $\mathbf{x}$ in random order
**end function**

**function** SHUFFLE($\mathbf{x}$)
  **for** $i \leftarrow 1$ *to* $len(\mathbf{x})$ **do**
    $indices[i] \leftarrow i + \delta_i \sim N(0, \sigma)$
  **end for**
  Rearrange $\mathbf{x}$ based on $argsort(indices)$
**end function**
**function** DELETE($\mathbf{x}$)
  Sample $p \sim Beta(\alpha, \beta)$
  **for** $w$ *in* $\mathbf{x}$ **do**
    Delete $w$ if $\mu \sim B(p)$ is 1
  **end for**
**end function**
**function** REPLACE($\mathbf{x}$)
  Sample $p \sim Beta(\alpha, \beta)$
  **for** $w$ *in* $\mathbf{x}$ **do**
    Replace $w$ with $w'$ sampled from unigram distribution if $\mu \sim B(p)$ is 1
  **end for**
**end function**

context $\mathbf{x}_{<i}$. This shows that our seq2seq formulation is capable of unifying both encoder-only pre-training and decoder-only pre-training methods, since a bidirectional language model used by BERT can be seen as simulating $p(\mathbf{x}_i|\mathbf{x}')$ , while a traditional unidirectional language model used by OpenAI GPT as resembling $p(\mathbf{x}_i|\mathbf{x}_{<i})$.

Like BERT, we add a mask to the target sequence when computing the loss function. To force the model to learn meaningful representations, instead of copying from the input most of the time, the positions where the corresponding words are corrupted in the input are kept. We also keep a small percentage (3%) of positions where the words are not corrupted, so that the model can learn to copy from the input when appropriate. Then, the training loss with mask is as follows ($\boldsymbol{\Theta}$ is model parameters):

$$L = -\sum_{i=1}^{n} mask(i) \cdot \log p(\mathbf{x}_i|\mathbf{x}', \mathbf{x}_{<i}, \boldsymbol{\Theta}) \quad (2)$$

Empirically, we set $\sigma = 0.5$ for Gaussian distribution. $\alpha$ and $\beta$ are chosen to have a Beta distribution with mean 0.15 and standard deviation 0.03.

### 2.3 Pre-training Procedure

| Corpus | #words |
|---|---|
| English Wikipedia | 2.22B |
| Billion Word Benchmark | 0.76B |
| Total | 2.99B |

Table 1: Text corpora used for pre-training.

For pre-training, we use two text corpora: the full dump of English Wikipedia[2] and the Billion Word Benchmark[3], as shown in Table 1. For English Wikipedia, we remove paragraphs with less than 3 words or more than 30% OOV words, and each paragraph is split into text segments with no more than 128 words for each segment. The Billion Word Benchmark is a sentence-level corpus. Sentences with more than 500 words are ignored during training.

The pre-training is performed on 4 GPUs using synchronous data parallelism, gradients are averaged across different GPUs. Each batch on a single GPU consists of at most 3000 tokens. We pre-train the network for 5 million iterations, which is roughly 14 epochs over the entire dataset. The final perplexity on the validation set is about 6.8. Each epoch takes approximately 2 days. Details on the network hyperparameters and optimizers are given in Section 3.1.

### 2.4 Fine-tuning Procedure

With our pre-training method, we do not need to change the network architecture during the fine-tuning stage, since both the pre-training task and text generation tasks take a source sequence as input and return a target sequence as output. The network is initialized with pre-trained parameter values. For fine-tuning, the preprocessing is dataset-specific, but the learning rate scheduling, dropout, early stopping, and gradient clipping are exactly the same as pre-training.

The objective function for fine-tuning is the word-level negative log-likelihood. Here we do not use reinforcement learning to tune towards the automatic evaluation metrics such as ROUGE

[2] https://dumps.wikimedia.org/

[3] http://www.statmt.org/lm-benchmark/, we do not use BooksCorpus (Zhu et al., 2015) used by BERT, because it is not publicly available now.

(Lin, 2004) or BLEU (Papineni et al., 2002), because it may overfit evaluation metrics and barely show improvements in human evaluations (Wu et al., 2016).

## 3 Experiments

### 3.1 Setup

The network architecture used by our experiments has 97 million parameters. It consists of 6 layers of encoder blocks, 6 layers of decoder blocks, and 1 pointer-generator layer. The hidden size of each positionwise feedforward layer is 4096. We use 8 heads for all multi-head attention layers. The vocabulary consists of the top $50k$ most frequent words (case sensitive), and the dimension of word embedding is 512. We tie the parameters of encoder word embeddings, decoder word embeddings, and the output softmax layer. NAG (Nesterov Accelerated Gradient) optimizer is used with initial learning rate $2 \times 10^{-3}$. Dropout of 0.2 is applied for all self-attention layers, positionwise feedforward layers and input embedding layers. The gradient norm is clipped to have a maximum value of 2. We follow the Transformer implementation from *fairseq*[4].

For task-specific fine-tuning, unless explicitly specified, we reuse the hyperparameters from the pre-training stage. After each training epoch, we compute the validation loss and halve the learning rate whenever the validation loss stops decreasing. The training procedure terminates if the learning rate drops below $10^{-4}$. Exponential moving average (EMA) with decay rate 0.9995 is used to make the training stabilized. At inference time, we use standard beam search decoding based on the length-normalized log-likelihood. For ensemble models, we use different random seeds and pre-trained checkpoints for fine-tuning. Ensemble decoding is used by averaging the output probabilities from different models at every decoding step.

When reporting experimental results, *"PoDA w/o pre-training"* refers to the proposed architecture in Section 2.1 trained only on the supervised data, and *"PoDA w/o fine-tuning"* only pre-trains on unlabeled data. *PoDA* first pre-trains a denoising autoencoder and then fine-tunes on the supervised data.

[4] https://github.com/pytorch/fairseq

### 3.2 Abstractive Summarization

**Datasets** We use two summarization datasets: CNN/Daily Mail[5] (See et al., 2017) and Gigaword (Rush et al., 2015) dataset. The official split for training, validation, and test is shown in Table 2.

| Corpus | # of examples | | |
|---|---|---|---|
| | train | valid | test |
| CNN/Daily Mail | 287,113 | 13,368 | 11,490 |
| Gigaword | 3,803,957 | 189,651 | 1,951 |

Table 2: Dataset statistics for abstractive summarization.

The CNN/Daily Mail dataset contains approximately $300k$ news articles with an average of 781 words for each article, and each article is paired with summaries with 56 words on average. We use the preprocessing script[6] provided by See et al. (2017). The articles are truncated to 800 words for both training and testing. The summaries are truncated to 130 words for training.

The Gigaword is a headline-generation dataset consisting of nearly 4 million examples. Headline generation can be seen as a sentence summarization task. Each example in Gigaword consists of one sentence with an average length of 31.3 words, which is much shorter than CNN/Daily Mail, and one short headline with an average length of 8.3 words. The Gigaword dataset provided by Rush et al. (2015) is already tokenized and lower-cased. Since our vocabulary is case-sensitive, such inconsistency is expected to hurt our system's performance.

**Evaluation** We report evaluation results in terms of of ROUGE-1, ROUGE-2 and ROUGE-L (Lin, 2004) using the *pyrouge*[7] package. For the CNN/Daily Mail dataset, PGNet (See et al., 2017), Lead3 (See et al., 2017), rnn-ext + RL (**?**), NeuSum (Zhou et al., 2018) are used as baselines. For the Gigaword dataset, ABS+ (Rush et al., 2015), CGU (Lin et al., 2018), FTSum (Cao et al., 2018b), and Re$^3$Sum (Cao et al., 2018a) are used as baselines.

**Results for CNN/Daily Mail** Considering the characteristics of news articles, baselines such as Lead3 (simply choose the first 3 sentences) can achieve strong performance in terms of ROUGE

[5] We use the non-anonymized version, which is considered to be more realistic.

[6] https://github.com/abisee/cnn-dailymail

[7] https://github.com/andersjo/pyrouge

| System | ROUGE | | |
|---|---|---|---|
| | 1 | 2 | L |
| Lead3 | 40.34 | 17.70 | 36.57 |
| PGNet | 36.44 | 15.66 | 33.42 |
| rnn-ext + RL | 41.47 | 18.72 | 37.76 |
| NeuSum | 41.59 | 19.01 | 37.98 |
| **PoDA w/o pre-training** | 40.82 | 18.46 | 37.61 |
| **PoDA** | **41.87** | **19.27** | **38.54** |

Table 3: ROUGE scores for CNN/Daily Mail dataset.

scores, as shown in Table 3. "rnn-ext+RL" combines both extractive and abstractive methods and achieves performance improvements (**?**). PoDA is a purely abstractive summarization system and performs stably better than all the methods. "PoDA w/o pre-training" only has moderate success with ROUGE-1 40.82, ROUGE-2 18.46 and ROUGE-L 37.61. When combined with pre-training, PoDA establishes new state-of-the-art on CNN/Daily Mail dataset.

| System | ROUGE | | |
|---|---|---|---|
| | 1 | 2 | L |
| ABS+ | 29.76 | 11.88 | 26.96 |
| CGU | 36.3 | 18.0 | 33.8 |
| FTSum | 37.27 | 17.65 | 34.24 |
| Re$^3$Sum | 37.04 | 19.03 | 34.46 |
| **PoDA w/o pre-training** | 37.24 | 18.28 | 34.53 |
| **PoDA** | **38.29** | **19.06** | **35.45** |

Table 4: ROUGE scores for Gigaword dataset.

**Results for Gigaword** The Gigaword dataset is much larger than CNN/Daily Mail, and this enables "PoDA w/o pre-training" to have competitive performance even without pre-training. As in Table 4, PoDA substantially improves ROUGE-1 from 37.24 to 38.29(+1.05), ROUGE-2 from 18.28 to 19.06(+0.78), and ROUGE-L from 34.53 to 35.45(+0.92), with pre-training added. We can see that PoDA performs favorably over the state-of-the-art Re$^3$Sum method, which utilizes unlabeled text corpora by first retrieving and then rewriting relevant snippets.

### 3.3 Grammatical Error Correction (GEC)

**Datasets** GEC can also be seen as a text generation task, where the input sequence is a sentence possibly containing some grammatical errors, and the output is a clean and grammatical sentence. We experiment PoDA on two GEC datasets: CoNLL-2014 (Ng et al., 2014) and JFLEG (Napoles et al., 2017). We use three public datasets for training: Lang-8 NAIST (Mizumoto et al., 2011), NUCLE (Dahlmeier et al., 2013) and CLC FCE (Felice et al., 2014). The test set of CoNLL-2013 shared task is used as validation set for the CoNLL-2014 task. JFLEG has its own validation set. For preprocessing, we use NLTK[8] to tokenize sentences, and remove all sentence pairs without any edits in Lang-8 NAIST. Simple spelling errors are corrected based on edit distance. The dataset statistics are shown in Table 5.

| Corpus | #Sent Pairs | Split |
|---|---|---|
| Lang-8 NAIST | 1,097,274 | train |
| NUCLE | 57,113 | train |
| CLC FCE | 32,073 | train |
| CoNLL-2013 test set | 1,381 | valid |
| JFLEG valid set | 754 | valid |
| CoNLL-2014 test set | 1,312 | test |
| JFLEG test set | 747 | test |

Table 5: Dataset statistics for grammatical error correction. Due to the lack of standard preprocessing script, the number of sentence pairs in the training set are slightly different from previous work.

**Evaluation** To compare with previous work, we use the official evaluation metrics: MaxMatch ($M^2$) $F_{0.5}$ (Dahlmeier and Ng, 2012) for CoNLL-2014 and GLEU (Napoles et al., 2015) for JFLEG dataset. Both metrics are shown to correlate well with human evaluation scores. ML-Conv (Chollampatt and Ng, 2018a), char-seq2seq (Xie et al., 2016), dual-boost (Ge et al., 2018a), Hybrid SMT-NMT (Grundkiewicz and Junczys-Dowmunt, 2018), NQE (Chollampatt and Ng, 2018b), and NRL (Sakaguchi et al., 2017) are used as baselines.

**Results** As shown in Table 6 and Table 7, when trained only on the supervised data, "PoDA w/o pre-training" can still achieve an impressive performance with $F_{0.5}$ score 54.01 on CoNLL-2014 test set and GLEU score 56.52 on JFLEG test set, surpassing previous state-of-the-art single model results. This once again shows the effectiveness of the Transformer architecture. For GEC task, most words in the output sequence also appear in the input sequence, pointer-generator makes it easier to learn such prior. With denoising based pre-training, PoDA greatly boosts the $F_{0.5}$ score from 54.01 to 59.40(+5.39) for CoNLL-2014 dataset,

[8] https://www.nltk.org/

| System (single) | $P$ | $R$ | $F_{0.5}$ |
|---|---|---|---|
| char-seq2seq | 49.24 | 23.77 | 40.56 |
| MLConv | 60.90 | 23.74 | 46.38 |
| dual-boost | 62.70 | 27.69 | 50.04 |
| **PoDA w/o fine-tuning** | 19.22 | 31.73 | 20.86 |
| **PoDA w/o pre-training** | 65.63 | 31.62 | 54.01 |
| **PoDA** | **70.10** | **36.88** | **59.40** |
| Ensemble | | | |
| MLConv(+rerank) | 65.49 | 33.14 | 54.79 |
| SMT-NMT(+rerank) | 66.77 | 34.49 | 56.25 |
| NQE | - | - | 56.52 |
| **PoDA** | **71.01** | **37.68** | **60.34** |

Table 6: Precision ($P$), recall ($R$) and $F_{0.5}$ scores for CoNLL-2014 test set. We only list systems trained on public data. Ge et al. (2018b) reported better performance with additional 4 million non-public sentence pairs.

| System (single) | valid | test |
|---|---|---|
| MLConv | 47.71 | 51.34 |
| NRL | 49.82 | 53.98 |
| dual-boost | 51.35 | 56.33 |
| **PoDA w/o fine-tuning** | 34.43 | 36.83 |
| **PoDA w/o pre-training** | 51.57 | 56.52 |
| **PoDA** | **53.16** | **59.02** |
| Ensemble | | |
| MLConv(+rerank) | 52.48 | 57.47 |
| SMT-NMT(+rerank) | - | **61.50** |
| **PoDA** | **53.29** | 59.48 |
| Human | - | 62.38 |

Table 7: GLEU scores for JFLEG dataset.

and GLEU score from 56.52 to 59.02(+2.50) for JFLEG. By ensembling 4 models initialized with different pre-trained checkpoints and trained with different random seeds, the performance can be further boosted on both datasets (+0.94 for CoNLL-2014 and +0.46 for JFLEG), outperforming the other ensemble models such as "Hybrid SMT-NMT".

We also report the performance of "PoDA w/o fine-tuning" which does not conduct fine-tuning. The $F_{0.5}$ score only reaches 20.86 on CoNLL-2014 dataset and the GLEU score is 36.83 on JFLEG. These results are even worse than the weakest baselines in Table 6 and Table 7. The denoising based pre-training and the GEC task share some similarities in the sense that both of them attempt to convert noisy texts to clean and grammatical texts. However, the poor results of "PoDA w/o fine-tuning" show that PoDA cannot be seen as a simple data augmentation method for GEC. Instead, PoDA learns generic text representations and requires task-specific fine-tuning.

Techniques from previous work for GEC such as language model based rerank (Chollampatt and Ng, 2018a), data augmentation (Ge et al., 2018a), and domain adaptation (Junczys-Dowmunt et al., 2018) can be easily incorporated. A parallel work (Zhao et al., 2019) observes similar gain by combining simpler pre-training strategy and various GEC-specific techniques.

## 4 Analysis

In the following analysis, we only choose one task (summarization or GEC) to analyze each aspect due to space limitation. Similar conclusions also hold for the other task.

### 4.1 Linguistic Quality Analysis

In Table 8, we show some generated summaries by PoDA from Gigaword dataset. In the first example, PoDA successfully deletes the relatively unimportant modifier "ivory coast striker", and keeps the picture complete by including both "bremen" and "saint-etienne" in the summary. In the second example, "PoDA w/o pre-training" misses an important date ("first quarter") of the event "economic crisis". Both examples show our model is able to identify the important text snippets in the source sequence and organize them into a short and fluent summary.

More example outputs by PoDA are listed in Appendix.

### 4.2 Effects of the Number of Pre-training Iterations

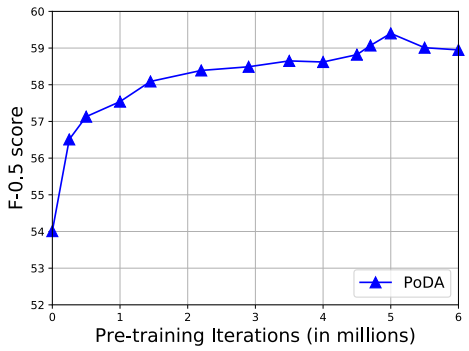


Figure 2: $F_{0.5}$ score on CoNLL-2014 test set with respect to the number of pre-training iterations.

| | |
|---|---|
| Source | ivory coast striker boubacar sanogo is set to leave werder bremen for french first division side saint-etienne. |
| Target | sanogo set to sign for saint-etienne |
| PoDA w/o pre-training | ivory coast striker sanogo set to join saint-etienne |
| PoDA | sanogo set to leave bremen for saint-etienne |
| Source | thailand's battered economy should start to bottom out in the first quarter of #### provided the government's efforts are n't neutralized by outside developments, a news report said monday. |
| Target | economic crisis to bottom out early next year minister says |
| PoDA w/o pre-training | thai economy expected to start to bottom out in UNK |
| PoDA | thai economy to start to bottom out in first quarter |

Table 8: Examples of generated summaries from Gigaword dataset.

In Figure 2, we show the $F_{0.5}$ score on CoNLL-2014 dataset when the model is initialized with different pre-trained checkpoints. Though the $F_{0.5}$ score has some fluctuations due to the random factors in training neural networks and the limited size of the test set, the overall trend is very clear: the $F_{0.5}$ score first improves greatly and then keeps a slow improvement after about 1 million iterations, from 54 at the very beginning to 59 after convergence.

### 4.3 Effects of Pre-trained Encoder and Decoder

To show the effectiveness of the pre-trained encoder and decoder, we train the model by only using the encoder-side pre-trained parameters ("w/o pre-trained decoder") or decoder-side pre-trained parameters ("w/o pre-trained encoder") We do not compare with pre-trained encoder from BERT or pre-trained decoder from OpenAI GPT, mainly because the corresponding model capacity, tokenization and text corpora used for pre-training are very different.

| System | $P$ | $R$ | $F_{0.5}$ |
|---|---|---|---|
| Fully pre-trained | **70.10** | **36.88** | **59.40** |
| w/o pre-trained encoder | 66.14 | 34.67 | 55.98 |
| w/o pre-trained decoder | 66.62 | 36.10 | 56.98 |
| w/o pre-training | 65.63 | 31.62 | 54.01 |

Table 9: Ablations for pre-trained encoder and decoder on CoNLL-2014 test set.

Table 9 shows that the performance degrades by a large margin if the network is only partially pre-trained. The pre-trained encoder ($-3.42$ drop in $F_{0.5}$) is more important than the pre-trained decoder ($-2.42$ drop in $F_{0.5}$).

### 4.4 Effects of Dataset Size

We also conduct ablations in few-shot learning settings to see how the performance changes when the model only accesses a small percentage of labeled data. We randomly sample $10^3$ to $10^5$ training examples from the Gigaword dataset and train "PoDA w/o pre-training" and PoDA (with pre-training) using exactly the same hyperparameters.

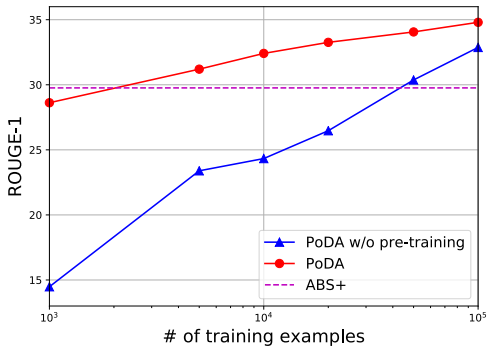


Figure 3: ROUGE-1 on Gigaword test set with respect to the number of training examples. ABS+ is a baseline method from Rush et al. (2015) using attention. The x-axis is in log scale.

Figure 3 shows the ROUGE-1 score on Gigaword test set. With only $10^3$ training examples, PoDA reaches a reasonably good performance comparable to ABS+ (an attention-based system trained on nearly 4 million examples). With more labeled data available, the performance gap between "PoDA w/o pre-training" and PoDA slowly decreases from 15 to 2 in Figure 3. However, the pre-training still helps even when the models are trained on the full dataset (shown in Table 4).

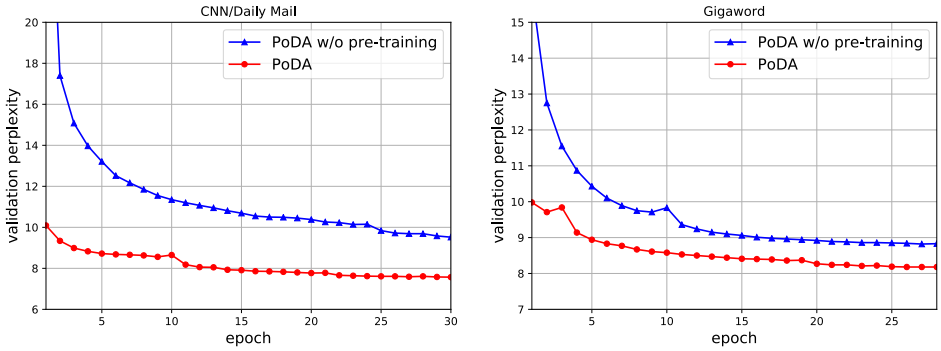


Figure 4: Validation perplexity with respect to the training epochs on CNN/Daily Mail and Gigaword datasets. Perplexity is related to the vocabulary size, so the values here are not comparable with previous work.

### 4.5 Convergence Analysis

Pre-training can not only achieve better final performance but also helps the model converge faster. In Figure 4, we show the validation perplexity after each training epoch for both "PoDA w/o pre-training" and PoDA.

We can clearly see that the validation perplexity of PoDA is consistently lower than that of "PoDA w/o pre-training", especially at the first few epochs. After 5 epochs, PoDA can arrive at a validation perplexity that "PoDA w/o pre-training" usually takes 30 or more epochs for both CNN/Daily Mail and Gigaword datasets. This nice property is particularly helpful when the computational resources are limited. Other pre-training methods such as BERT also demonstrate similar behaviors.

## 5 Related Work

**Network Pre-training** The idea of pre-training neural networks dates back to the early days of deep learning. Bengio et al. (2007) proposed layer-wise pre-training for deep belief networks (DBN) to tackle the difficulty of training deep neural networks based on a reconstruction objective. (Erhan et al., 2010; Dahl et al., 2012) showed the effectiveness of pre-training for tasks such as speech recognition. In the area of computer vision, using ImageNet pre-trained models have become a standard practice. In NLP community, using pre-trained word embeddings is the most popular way to transfer knowledge from the unlabeled corpus. There are also work on semi-supervised sequence learning (Dai and Le, 2015; Peters et al., 2017) attempting to incorporate language modeling as an auxiliary task. Recently, several pre-training methods based on language models are presented, such as ELMo (Peters et al., 2018), OpenAI GPT (Radford et al., 2018), BERT (Devlin et al., 2018), XLM (Lample and Conneau, 2019) etc. The combination of more compute, larger model capacity and large-scale text corpora lead to significant improvements on NLP benchmarks (Wang et al., 2018).

**Autoencoders** have long been used for representation learning of images (Vincent et al., 2010) and text (Li et al., 2015). However, precisely reconstructing the clean input is probably too easy for high-capacity models. Sparse autoencoders (Deng et al., 2013), contractive autoencoders (Rifai et al., 2011), and denoising autoencoders (Vincent et al., 2010) are several popular variants. Denoising autoencoders (DA) are shown to be able to learn better representations for downstream tasks (Vincent et al., 2010, 2008; Hill et al., 2016). Freitag and Roy (2018) use seq2seq DAs for unsupervised natural language generation in dialogue, and (Kim et al., 2018) propose to improve the quality of machine translation with DAs.

**Text Generation** covers a wide spectrum of NLP tasks, including machine translation (Wu et al., 2016), summarization (See et al., 2017), response generation (Vinyals and Le, 2015), paraphrase generation, grammatical error correction etc. Early studies on text generation mainly adopt template-based (Reiter and Dale, 2000) or example-based (Watanabe and Takeda, 1998) methods. With the emergence of deep learning for NLP, seq2seq models (Sutskever et al., 2014) become a popular choice for text generation tasks and show better performance in terms of both

automatic evaluation metrics and human evaluations (Wu et al., 2016). There are also studies focusing on text generation from structured data such as SQL-to-text (Xu et al., 2018). Previous pre-training for text generation is usually done by independently pre-training encoder-side or decoder-side language models (Ramachandran et al., 2016). Concurrent to our work, Edunov et al. augment encoder representation with ELMo-style models, MASS (Song et al., 2019) masks continuous text fragments for pre-training, and UNILM (Dong et al., 2019) proposes to pre-train for both language understanding and generation tasks.

## 6 Conclusion

This paper presents a new transfer learning approach for seq2seq text generation named PoDA. It involves two steps: first, pre-train a customized seq2seq denoising autoencoder on large-scale unlabeled text corpora; then, fine-tune on in-domain labeled data. The pre-training step is independent of downstream tasks and jointly learns both encoder and decoder representations. PoDA is simple, intuitive and doesn't require changing network architecture during the fine-tuning stage. Experiments on several abstractive summarization and grammatical error correction datasets demonstrate that PoDA leads to better performance and faster convergence.

For future work, we would like to validate our model on other tasks such as response generation, explore more effective unsupervised sequence-to-sequence pre-training methods, and handle cross-lingual tasks such as machine translation.

## Acknowledgments

We want to thank three anonymous reviewers for their valuable comments, and EMNLP-IJCNLP 2019 organizers for their efforts.

## References

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. 2007. Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, pages 153–160.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642.

Ziqiang Cao, Wenjie Li, Sujian Li, and Furu Wei. 2018a. Retrieve, rerank and rewrite: Soft template based neural summarization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 152–161.

Ziqiang Cao, Furu Wei, Wenjie Li, and Sujian Li. 2018b. Faithful to the original: Fact aware neural abstractive summarization. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Shamil Chollampatt and Hwee Tou Ng. 2018a. A multilayer convolutional encoder-decoder neural network for grammatical error correction. *arXiv preprint arXiv:1801.08831*.

Shamil Chollampatt and Hwee Tou Ng. 2018b. Neural quality estimation of grammatical error correction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2528–2539.

George E Dahl, Dong Yu, Li Deng, and Alex Acero. 2012. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on audio, speech, and language processing*, 20(1):30–42.

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 568–572. Association for Computational Linguistics.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. 2013. Building a large annotated corpus of learner english: The nus corpus of learner english. In *Proceedings of the eighth workshop on innovative use of NLP for building educational applications*, pages 22–31.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Advances in neural information processing systems*, pages 3079–3087.

Jun Deng, Zixing Zhang, Erik Marchi, and Bjorn Schuller. 2013. Sparse autoencoder-based feature transfer learning for speech emotion recognition. In *2013 Humaine Association Conference on Affective Computing and Intelligent Interaction*, pages 511–516. IEEE.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. *arXiv preprint arXiv:1905.03197*.

Sergey Edunov, Alexei Baevski, and Michael Auli. 2019. Pre-trained language model representations for language generation. *arXiv preprint arXiv:1903.09722*.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660.

Mariano Felice, Zheng Yuan, Øistein E Andersen, Helen Yannakoudakis, and Ekaterina Kochmar. 2014. Grammatical error correction using hybrid systems and type filtering. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 15–24.

Markus Freitag and Scott Roy. 2018. Unsupervised natural language generation with denoising autoencoders. *arXiv preprint arXiv:1804.07899*.

Tao Ge, Furu Wei, and Ming Zhou. 2018a. Fluency boost learning and inference for neural grammatical error correction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1055–1065.

Tao Ge, Furu Wei, and Ming Zhou. 2018b. Reaching human-level performance in automatic grammatical error correction: An empirical study. *arXiv preprint arXiv:1807.01270*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

Roman Grundkiewicz and Marcin Junczys-Dowmunt. 2018. Near human-level performance in grammatical error correction with hybrid machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 284–290.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393*.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 140–149.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *Proceedings of NAACL-HLT*, pages 1367–1377.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 328–339.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Shubha Guha, and Kenneth Heafield. 2018. Approaching neural grammatical error correction as a low-resource machine translation task. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 595–606.

Yunsu Kim, Jiahui Geng, and Hermann Ney. 2018. Improving unsupervised word-by-word translation with language model and denoising autoencoder. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 862–868.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*.

Jiwei Li, Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1106–1115.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Junyang Lin, Xu Sun, Shuming Ma, and Qi Su. 2018. Global encoding for abstractive summarization. *arXiv preprint arXiv:1805.03989*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6294–6305.

Tomoya Mizumoto, Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. 2011. Mining revision log of language learning sns for automated japanese error correction of second language learners. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 147–155.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2015. Ground truth for grammatical error correction metrics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 588–593.

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2017. Jfleg: A fluency corpus and benchmark for grammatical error correction. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 229–234.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The conll-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1756–1765.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 2227–2237.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/language-unsupervised/language_ understanding_paper. pdf*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Prajit Ramachandran, Peter J Liu, and Quoc V Le. 2016. Unsupervised pretraining for sequence to sequence learning. *arXiv preprint arXiv:1611.02683*.

Ehud Reiter and Robert Dale. 2000. *Building natural language generation systems*. Cambridge university press.

Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. 2011. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pages 833–840. Omnipress.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389.

Keisuke Sakaguchi, Matt Post, and Benjamin Van Durme. 2017. Grammatical error correction with neural reinforcement learning. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 366–372.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. Mass: Masked sequence to sequence pre-training for language generation. In *International Conference on Machine Learning*, pages 5926–5936.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. Stacked denoising autoencoders: Learning

useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(Dec):3371–3408.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Alex Wang, Amapreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Hideo Watanabe and Koichi Takeda. 1998. A pattern-based machine translation system extended by example-based processing. In *Proceedings of the 17th international conference on Computational linguistics-Volume 2*, pages 1369–1373. Association for Computational Linguistics.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Ziang Xie, Anand Avati, Naveen Arivazhagan, Dan Jurafsky, and Andrew Y Ng. 2016. Neural language correction with character-based attention. *arXiv preprint arXiv:1603.09727*.

Kun Xu, Lingfei Wu, Zhiguo Wang, Yansong Feng, and Vadim Sheinin. 2018. Sql-to-text generation with graph-to-sequence model. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 931–936.

Wei Zhao, Liang Wang, Kewei Shen, Ruoyu Jia, and Jingming Liu. 2019. Improving grammatical error correction via pre-training a copy-augmented architecture with unlabeled data. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 156–165.

Qingyu Zhou, Nan Yang, Furu Wei, Shaohan Huang, Ming Zhou, and Tiejun Zhao. 2018. Neural document summarization by jointly learning to score and select sentences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 654–663.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

# A Supplemental Material

## A.1 Implementation of Pointer-Generator Layer

The pointer-generator layer calculates a probability distribution over the union of a fixed vocabulary $\mathbf{V}$ and the words in the input sequence $\{\mathbf{x}_i\}_{i=1}^{n}$.

For a word $w$, the probability $p(w)$ can be calculated as follows:

$$\begin{aligned} \boldsymbol{\alpha} &= MultiHeadAttention(\mathbf{h}_{dec}^{t}, \mathbf{h}_{enc}) \\ p_{gen} &= \sigma(\mathbf{W}_1 AttPooling(\mathbf{h}_{enc}, \boldsymbol{\alpha})) \\ p(w) &= p_{gen} p_v(w) + (1 - p_{gen}) \sum_{i:\mathbf{x}'_i = w} \boldsymbol{\alpha}_i \end{aligned} \tag{3}$$

$\mathbf{h}_{enc}$ is the top layer output of the Transformer encoder, and $\mathbf{h}_{dec}^{t}$ is the output of the Transformer decoder at timestep $t$. $p_v(w)$ is the standard softmax probability for word $w$ over the vocabulary ($p_v(w) = 0$ if $w$ is an OOV word). $\boldsymbol{\alpha}$ is the copy attention over the input sequence, *AttPooling* is the attentive pooling of encoder outputs $\mathbf{h}_{enc}$ with distribution $\boldsymbol{\alpha}$, and $p_{gen}$ denotes the probability of generating from the fixed vocabulary. $p(w)$ is a linear interpolation of the generation and copying probabilities.

| | |
|---|---|
| Source | I do think there is difference in it and I believe many of us will agree. |
| Target | I do think there is **a difference** and I believe many of us will agree. |
| PoDA w/o pre-training | I do think there is **difference in it** and I believe many of us will agree. |
| PoDA | I do think there is **a difference in it** and I believe many of us will agree. |
| Source | Almost all students and young adults possess the Facebook or Twitter account. |
| Target | Almost all students and young adults possess **a Facebook or Twitter account**. |
| PoDA w/o pre-training | Almost all students and young adults possess **Facebook or Twitter accounts**. |
| PoDA | Almost all students and young adults possess **a Facebook or Twitter account**. |

Table 10: Examples of corrected sentences from CoNLL-2014 dataset. The important text snippets are highlighted with bold font.

| | |
|---|---|
| Source | But, on the contrary, he argues that fluoride also some disadvantage. |
| Target | But, on the contrary, he argues that fluoride also **has some disadvantages**. |
| PoDA w/o pre-training | But, on the contrary, he argues that **there are also some disadvantages**. |
| PoDA | But, on the contrary, he argues that fluoride also **has some disadvantages**. |
| Source | Such people impressed other people through their strong well and divoution to duty. |
| Target | Such people impressed others through their **strong will and devotion to duty**. |
| PoDA w/o pre-training | Such people impressed other people through their **strong and divoution to duty**. |
| PoDA | Such people impressed other people through their **strong will and devotion to duty**. |

Table 11: Examples of corrected sentences from JFLEG dataset. The important text snippets are highlighted with bold font.

| Input | Output samples |
|---|---|
| the best university in the world. | Harvard is considered the 10th best university in the world.<br>I believe that's the best university in the world.<br>Nevada offers the best continuing education in the whole world.<br>Greek is becoming the best university in the world. |
| The meaning of life is. | The meaning of daily life is unclear.<br>"The real meaning of daily life is peaceful."<br>The underlying meaning of your life is lost forever.<br>The immediate meaning of our political life is undeniable. |

Table 12: Some input-output samples by our pre-trained PoDA models. We input some incomplete sentences and output the generated sentences by sampling the output distribution of pre-trained PoDA models at each timestep. We can see that PoDA successfully transforms the inputs to coherent and grammatical sentences, though the statements entailed by these output sentences are not always correct.

# Dialog Intent Induction with Deep Multi-View Clustering

**Hugh Perkins** and **Yi Yang**
ASAPP Inc.
New York, NY 10007
{hp+yyang}@asapp.com

## Abstract

We introduce the dialog intent induction task and present a novel deep multi-view clustering approach to tackle the problem. Dialog intent induction aims at discovering user intents from user query utterances in human-human conversations such as dialogs between customer support agents and customers.[1] Motivated by the intuition that a dialog intent is not only expressed in the user query utterance but also captured in the rest of the dialog, we split a conversation into two independent views and exploit multi-view clustering techniques for inducing the dialog intent. In particular, we propose alternating-view k-means (AV-KMEANS) for joint multi-view representation learning and clustering analysis. The key innovation is that the instance-view representations are updated iteratively by predicting the cluster assignment obtained from the alternative view, so that the multi-view representations of the instances lead to similar cluster assignments. Experiments on two public datasets show that AV-KMEANS can induce better dialog intent clusters than state-of-the-art unsupervised representation learning methods and standard multi-view clustering approaches.[2]

## 1 Introduction

Goal-oriented dialog systems assist users to accomplish well-defined tasks with clear intents within a limited number of dialog turns. They have been adopted in a wide range of applications, including booking flights and restaurants (Hemphill et al., 1990; Williams, 2012), providing tourist information (Kim et al., 2016), aiding in the customer support domain, and powering intelligent virtual assistants such as Apple Siri, Amazon Alexa, or Google Assistant. The first step towards building such systems is to determine the target tasks and construct corresponding ontologies to define the constrained set of dialog states and actions (Henderson et al., 2014b; Mrkšić et al., 2015).

**Customer 1**: *A wireless charging case is fancy and all but can we get a "find my airpod" feature going?*
**Agent 1**: *If you have lost your AirPods, Find My iPhone can help you locate them.*

**Customer 2**: *hey man I lost and miss my airpods plz help me!*
**Agent 2**: *Hi there! With iOS 10.3 or later, Find My iPhone can help you locate missing AirPods.*

Figure 1: Two dialogs with the `FindAirPods` user intent. The user query utterances of the two dialogs are lexically and syntactically dissimilar, while the rests of the dialogs are similar.

Existing work assumes the target tasks are given and excludes dialog intent discovery from the dialog system design pipeline. Because of this, most of the works focus on few simple dialog intents and fail to explore the realistic complexity of user intent space (Williams et al., 2013; Budzianowski et al., 2018). The assumption puts a great limitation on adapting goal-oriented dialog systems to important but complex domains like customer support and healthcare where having a complete view of user intents is impossible. For example, as shown in Fig. 1, it is non-trivial to predict user intents for troubleshooting a newly released product in advance. To address this problem, we propose to employ data-driven approaches to automatically discover user intents in dialogs from human-human conversations. Follow-up analysis can then be performed to identify the most valuable dialog intents and design dialog systems to automate the conversations accordingly.

Similar to previous work on user question/query intent induction (Sadikov et al., 2010; Haponchyk

[1] We focus on inducing abstract intents like `BookFlight` and ignore detailed arguments such as *departure date* and *destination*.

[2] When ready, the data and code will be published at https://github.com/asappresearch/dialog-intent-induction.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4016–4025,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

et al., 2018), we can induce dialog intents by clustering user query utterances[3] in human-human conversations. The key is to learn discriminative query utterance representations in the user intent semantic space. Unsupervised learning of such representations is challenging due to the semantic shift across different domains (Nida, 2015). We propose to overcome this difficulty by leveraging the rest of a conversation in addition to the user query utterance as a weak supervision signal. Consider the two dialogs presented in Fig. 1 where both of the users are looking for how to find their AirPods. Although the user query utterances vary in the choice of lexical items and syntactic structures, the human agents follow the same workflow to assist the users, resulting in similar conversation structures.[4]

We present a deep multi-view clustering approach, alternating-view k-means (AV-KMEANS), to leverage the weak supervision for the semantic clustering problem. In this respect, we partition a dialog into two independent views: the user query utterance and the rest of the conversation. AV-KMEANS uses different neural encoders to embed the inputs corresponding to the two views and to encourage the representations learned by the encoders to yield similar cluster assignments. Specifically, we alternatingly perform k-means-style updates to compute the cluster assignment on one view and then train the encoder of the other view by predicting the assignment using a metric learning algorithm (Snell et al., 2017). Our method diverges from previous work on multi-view clustering (Bickel and Scheffer, 2004; Chaudhuri et al., 2009; Kumar et al., 2011), as it is able to learn robust representations via neural networks that are in clustering-analysis-friendly geometric spaces. Experimental results on a dialog intent induction dataset and a question intent clustering dataset show that AV-KMEANS significantly outperforms multi-view clustering algorithms without joint representation learning by 6–20% absolute F1 scores. It also gives rise to better F1 scores than quick thoughts (Logeswaran and Lee, 2018), a state-of-the-art unsupervised representation learning method.

[3]We treat the initial user utterances of the dialogs as user query utterances.

[4]Note this is not always the case. For the same dialog intent, the agent treatments may differ depending on the user profiles. The user may also change intent in the middle of a conversation. Thus, the supervision is often very noisy.

Our contributions are summarized as follows:

- We introduce the dialog intent induction task and present a multi-view clustering formulation to solve the problem.
- We propose a novel deep multi-view clustering approach that jointly learns cluster-discriminative representations and cluster assignments.
- We derive and annotate a dialog intent induction dataset obtained from a public Twitter corpus and process a duplicate question detection dataset into a question intent clustering dataset.
- The presented algorithm, AV-KMEANS, significantly outperforms previous state-of-the-art multi-view clustering algorithms as well as two unsupervised representation learning methods on the two datasets.

## 2 Deep Multi-View Clustering

In this section, we present a novel method for joint multi-view representation learning and clustering analysis. We consider the case of two independent views, in which the first view corresponds to the user query utterance (query view) and the second one corresponds to the rest of the conversation (content view).

Formally, given a set of $n$ instances $\{x_i\}$, we assume that each data point $x_i$ can be naturally partitioned into two independent views $x_i^{(1)}$ and $x_i^{(2)}$. We further use two neural network encoders $f_{\phi_1}$ and $f_{\phi_2}$ to transform the two views into vector representations $\mathbf{x}_i^{(1)}, \mathbf{x}_i^{(2)} \in \mathbb{R}^D$. We are interested in grouping the data points into $K$ clusters using the multi-view feature representations. In particular, the neural encoders corresponding to the two views are jointly optimized so that they would commit to similar cluster assignments for the same instances.

In this work, we implement the query-view encoder $f_{\phi_1}$ with a bi-directional LSTM (BiLSTM) network (Hochreiter and Schmidhuber, 1997) and the content-view encoder $f_{\phi_2}$ with a hierarchical BiLSTM model that consists of a utterance-level BiLSTM encoder and a content-level BiLSTM encoder. The concatenations of the hidden representations from the last time steps are adopted as the query or content embeddings.

### 2.1 Alternating-view k-means clustering

In this work, we propose alternating-view k-means (AV-KMEANS) clustering, a novel method for deep multi-view clustering that iteratively updates neural encoders corresponding to the two views by encouraging them to yield similar cluster assignments for the same instances. In each semi-iteration, we perform k-means-style updates to compute a cluster assignment and centroids on feature representations corresponding to one view, and then project the cluster assignment to the other view where the assignment is used to train the view encoder in a supervised learning fashion.

**Algorithm 1:** alternating-view k-means

**Input** : two-view inputs $\{(x_i^{(1)}, x_i^{(2)})\}$; numbers of iterations $T$, $M$; number of clusters $K$
**Output** : final cluster assignment $\{z_i^{(1)}\}$
**Parameter:** encoders $f_{\phi_1}$ and $f_{\phi_2}$
Initialize $f_{\phi_1}$ and $f_{\phi_2}$ (§ 2.3)
$\{z_i^{(1)}\} \leftarrow$ K-MEANS$(\{f_{\phi_1}(x_i^{(1)})\}, K)$
**for** $t = 1, \cdots, T$ **do**
  // project cluster assignment from view 1 to view 2
  Update $f_{\phi_2}$ with pseudo training instances $\{(x_i^{(2)}, z_i^{(1)})\}$ (§ 2.2)
  Encode view-2 inputs: $\{\mathbf{x}_i^{(2)} \leftarrow f_{\phi_2}(x_i^{(2)})\}$
  $\{z_i^{(2)}\} \leftarrow$ K-MEANS$(\{\mathbf{x}_i^{(2)}\}, K, M, \{z_i^{(1)}\})$

  // project cluster assignment from view 2 to view 1
  Update $f_{\phi_1}$ with pseudo training instances $\{(x_i^{(1)}, z_i^{(2)})\}$ (§ 2.2)
  Encode view-1 inputs: $\{\mathbf{x}_i^{(1)} \leftarrow f_{\phi_1}(x_i^{(1)})\}$
  $\{z_i^{(1)}\} \leftarrow$ K-MEANS$(\{\mathbf{x}_i^{(1)}\}, K, M, \{z_i^{(2)}\})$
**end**

The full training algorithm is presented in Alg. 1, where K-MEANS$(\{\mathbf{x}_i\}, K, M, \{z_i'\})$ is a function that runs k-means clustering on inputs $\{\mathbf{x}_i\}$. $K$ is the number of clusters. $M$ and $\{z_i'\}$ are optional arguments that represent the number of k-means iterations and the initial cluster assignment. The function returns cluster assignment $\{z_i\}$. A visual demonstration of one semi-iteration of AV-KMEANS is also available in Fig. 2.

In particular, we initialize the encoders randomly or by using pretrained encoders (§ 2.3). Then, we can obtain the initial cluster assignment by performing k-means clustering on vector representations encoded by $f_{\phi_1}$. During each AV-KMEANS iteration, we first project cluster assignment from view 1 to view 2 and update the neural encoder for view 2 by formulating a supervised learning problem (§ 2.2). Then we perform $M$ vanilla k-means steps to adjust the cluster assignment in view 2 based on the updated encoder. We repeat the procedure for view 2 in the same iteration. Note that in each semi-iteration, the initial centroids corresponding to a view are calculated based on the cluster assignment obtained from the other view. The algorithm runs a total number of $T$ iterations.

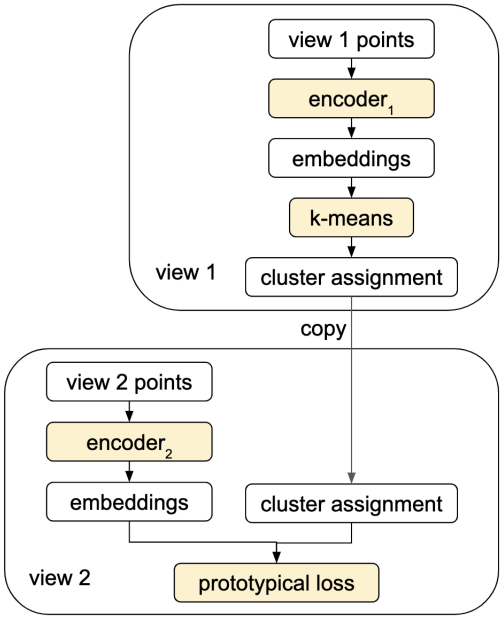


Figure 2: A depiction of a semi-iteration of the alternating-view k-means algorithm. k-means clustering and prototypical classification are performed for view 1 and view 2 respectively. The view 1 encoder is frozen and the view 2 encoder is updated in this semi-iteration.

### 2.2 Prototypical episode training

In each AV-KMEANS iteration, we need to solve two supervised classification problems using the pseudo training datasets $\{(x_i^{(2)}, z_i^{(1)})\}$ and $\{(x_i^{(1)}, z_i^{(2)})\}$ respectively. A simple way to do so is putting a softmax classification layer on top of each encoder network. However, we find that it is beneficial to directly perform classification in the k-means clustering space. To this end, we adopt prototypical networks (Snell et al., 2017), a metric learning approach, to solely rely on the encoders to form the classifiers instead of introducing additional classification layers.

Given input data $\{(x_i, z_i)\}$ and a neural network encoder $f_\phi$, prototypical networks compute

a $D$-dimensional representation $\mathbf{c}_k$, or prototype, of each class by averaging the vectors of the embedded support points belonging to its class:

$$\mathbf{c}_k = \frac{1}{|S_k|} \sum_{(x_i, z_i) \in S_k} f_\phi(x_i) \quad (1)$$

here we drop the view superscripts for simplicity. Conceptually, the prototypes $\{\mathbf{c}_k\}$ are similar to the centroids in the k-means algorithm, except that a prototype is computed on a subset of the instances of a class (the support set) while a centroid is computed based on all instances of a class.

Given a sampled query data point $x$, prototypical networks produce a distribution over classes based on a softmax over distances to the prototypes in the embedding space:

$$p(y = k|x) = \frac{\exp(-d(f_\phi(x), \mathbf{c}_k))}{\sum_{k'} \exp(-d(f_\phi(x), \mathbf{c}_{k'}))}, \quad (2)$$

where the distance function is the squared Euclidean distance $d(\mathbf{x}, \mathbf{x}') = ||\mathbf{x} - \mathbf{x}'||^2$.

The model minimizes the negative log-likelihood of the data: $L(\phi) = -\log p(y = k|x)$. Training episodes are formed by randomly selecting a subset of classes from the training set, then choosing a subset of examples within each class to act as the support set and a subset of the remainder to serve as query points. We refer to the original paper (Snell et al., 2017) for more detailed description of the model.

### 2.3 Parameter initialization

Although AV-KMEANS can effectively work with random parameter initializations, we do expect that it will benefit from initializations obtained from pretrained models with some well-studied unsupervised learning objectives. We present two methods to initialize the utterance encoders for both the query and content views. The first approach is based on recurrent autoencoders. We embed an utterance using a BiLSTM encoder. The utterance embedding is then concatenated with every word vector corresponding to the decoder inputs that are fed into a uni-directional LSTM decoder. We use the neural encoder trained with the autoencoding objective to initialize the two utterance encoders in AV-KMEANS.

Recurrent autoencoders independently reconstruct an input utterance without capturing semantic dependencies across consecutive utterances. We consider a second initialization method, quick thoughts (Logeswaran and Lee, 2018), that addresses the problem by predicting a context utterance from a set of candidates given a target utterance. Here, the target utterances are sampled randomly from the corpus, and the context utterances are sampled from within each pair of adjacent utterances. We use two separate BiLSTM encoders to encode utterances, which are named as the target encoder $f$ and the context encoder $g$. To score the compatibility of a target utterance $\mathbf{s}$ and a candidate context utterance $\mathbf{t}$, we simply use the inner product of the two utterance vectors $f(\mathbf{s})^\top \cdot g(\mathbf{t})$. The training objective maximizes the log-likelihood of the context utterance given the target utterance and the candidate utterance set. After pretraining, we adopt the target encoder to initialize the two utterance encoders in AV-KMEANS.

## 3 Data

As discussed in the introduction, existing goal-oriented dialog datasets mostly concern predefined dialog intents in some narrow domains such as restaurant or travel booking (Henderson et al., 2014a; Budzianowski et al., 2018; Serban et al., 2018). To carry out this study, we adopt a more challenging corpus that consists of human-human conversations for customer service and manually annotate the user intents of a small number of dialogs. We also build a question intent clustering dataset to assess the generalization ability of the proposed method on the related problem.

### 3.1 Twitter airline customer support

We consider the customer support on Twitter corpus released by Kaggle,[5] which contains more than three million tweets and replies in the customer support domain. The tweets constitute conversations between customer support agents of some big companies and their customers. As the conversations regard a variety of dynamic topics, they serve as an ideal testbed for the dialog intent induction task. In the customer service domain, different industries generally address unrelated topics and concerns. We focus on dialogs in the airline industry,[6] as they represent the largest number of conversations in the corpus. We name

[5] https://www.kaggle.com/thoughtvector/customer-support-on-twitter

[6] We combined conversations involved the following Twitter handles: @Delta, @British_Airways, @SouthwestAir, and @AmericanAir.

| Dialog intent | # Dialogs | Query utterance example |
|---|---|---|
| Baggage | 40 | hi, do suit bags count as a personal items besides carry on baggage? |
| BookFlight | 27 | trying all day to book an international flight, only getting error msg. |
| ChangeFlight | 16 | can i request to change my flight from lax to msy on 10/15? |
| CheckIn | 21 | hy how can i have some help... having some problems with a check in |
| CustomerService | 19 | 2 hour wait time to talk to a customer service agent?!? |
| FlightDelay | 85 | delay... detroit < orlando |
| FlightEntertainment | 40 | @airline is killing it with these inflight movie options |
| FlightFacility | 32 | just flew @airline economy... best main cabin seat ive ever sat in. |
| FlightStaff | 30 | great crew on las vegas to baltimore tonight. |
| Other | 116 | hi, i have a small question! |
| RequestFeature | 10 | when are you going to update your app for iphone x? |
| Reward | 17 | need to extend travel funds that expire tomorrow! |
| TerminalFacility | 13 | thx for the new digital signs at dallas lovefield. well done!! |
| TerminalOperation | 34 | would be nice if you actually announced delays |

Table 1: Statistics of the labeled Twitter airline customer support (TwACS) dataset and the corresponding user query utterance examples. The Twitter handles of the airlines are replaced by @airline.

the resulting dataset the *Twitter airline customer support (TwACS)* corpus. We rejected any conversation that redirects the customer to a URL or another communication channel, e.g., direct messages. We ended up with a dataset of $43,072$ dialogs. The total numbers of dialog turns and tokens are $63,147$ and $2,717,295$ respectively.

After investigating $500$ randomly sampled conversations from TwACS, we established an annotation task with $14$ dialog intents and hired two annotators to label the sampled dialogs based on the user query utterances. The Cohen's kappa coefficient was $0.75$, indicating a substantial agreement between the annotators. The disagreed items were resolved by a third annotator. To our knowledge, this is the first dialog intent induction dataset. The data statistics and user query utterance examples corresponding to different dialog intents are presented in Table 1.

### 3.2 AskUbuntu

*AskUbuntu* is a dataset collected and processed by Shah et al. (2018) for the duplicate question detection task. The dataset consists of technical support questions posted by users on AskUbuntu website with annotations indicating that two questions are semantically equivalent. For instance,

$q_1$ : *how to install ubuntu w/o removing windows*
$q_2$ : *installing ubuntu over windows 8.1*

are duplicate and they can be resolved with similar answers. A total number of $257,173$ questions are included in the dataset and $27,289$ pairs of questions are labeled as duplicate ones. In addition, we obtain the top rated answer for each question from the AskUbuntu website dump.[7]

In this work, we reprocess the data and build a question intent clustering dataset using an automatic procedure. Following Haponchyk et al. (2018), we transform the duplicate question annotations into the question intent cluster annotations with a simple heuristic: for each question pair $q_1$, $q_2$ annotated as a duplicate, we assigned $q_1$ and $q_2$ to the same cluster. As a result, the question intent clusters correspond to the connected components in the duplicate question graph. There are $7,654$ such connected components. However, most of the clusters are very small: $91.7\%$ of the clusters contain only 2–5 questions. Therefore, we experiment with the largest $20$ clusters that contain $4,692$ questions in this work. The sizes of the largest and the smallest clusters considered in this study are $1,364$ and $71$ respectively.

## 4 Experiments

In this section, we evaluate AV-KMEANS on the TwACS and AskUbuntu datasets as described in § 3. We compare AV-KMEANS with competitive systems for representation learning or multi-view clustering and present our main findings in § 4.2. In addition, we examine the output clusters obtained from AV-KMEANS on the TwACS dataset to perform a thoughtful error analysis.

### 4.1 Experimental settings

We train the models on all the instances of a dataset and evaluate on the labeled instances. We employ the publicly available 300-dimensional

[7] https://archive.org/details/stackexchange

| Clustering algorithm | Pretraining method | TwACS | | | | AskUbuntu | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Prec | Rec | F1 | ACC | Prec | Rec | F1 | ACC |
| *Baseline systems* | | | | | | | | | |
| k-means | PCA | 28.1 | 28.3 | 28.2 | 19.8 | 35.1 | 27.8 | 31.0 | 22.0 |
| | autoencoders | 34.4 | 25.9 | 29.5 | 23.2 | 27.3 | 20.1 | 23.1 | 14.6 |
| | quick thoughts | 46.7 | 38.3 | 42.1 | 35.4 | 42.9 | 39.4 | 41.1 | 33.1 |
| MVSC | PCA | 32.4 | 24.2 | 27.8 | 22.6 | 40.4 | 27.7 | 32.9 | 25.5 |
| | autoencoders | 36.1 | 27.7 | 31.3 | 24.9 | 36.7 | 22.8 | 28.2 | 21.4 |
| | quick thoughts | 45.8 | 35.4 | 40.0 | 32.6 | 35.1 | 23.4 | 28.1 | 22.3 |
| *Our approach* | | | | | | | | | |
| AV-KMEANS | no pretraining | 37.5 | 33.6 | 35.4 | 29.5 | 52.0 | 51.9 | 51.9 | **44.0** |
| | autoencoders | 44.4 | 34.6 | 38.9 | 31.6 | 50.6 | 46.1 | 48.2 | 39.7 |
| | quick thoughts | **48.9** | **43.8** | **46.2** | **39.9** | **53.8** | **52.7** | **53.3** | 41.1 |

Table 2: Evaluation results on the TwACS and AskUbuntu datasets for different systems. MVSC is short for the multi-view spectral clustering algorithm proposed by Kanaan-Izquierdo et al. (2018). The pretrained representations are fixed during k-means and MVSC clustering and they are fine-tuned during AV-KMEANS clustering. The best results are in **bold**.

GloVe vectors (Pennington et al., 2014) pretrained with 840 billion tokens to initialize the word embeddings for all the models.

**Competitive systems** We consider state-of-the-art methods for representation learning and/or multi-view clustering as our baseline systems. We formulate the dialog induction task as an unsupervised clustering task and include two popular clustering algorithms *k-means* and *spectral clustering*. *multi-view spectral clustering (MVSC)* (Kanaan-Izquierdo et al., 2018) is a competitive standard multi-view clustering approach.[8] In particular, we carry out clustering using the query-view and content-view representations learned by the representation learning methods (k-means only requires query-view representations). In the case where a content-view input corresponds to multiple utterances, we take the average of the utterance vectors as the content-view output representation for autoencoders and quick thoughts.

AV-KMEANS is a joint representation learning and multiview clustering method. Therefore, we compare with SOTA representation learning methods *autoencoders*, and *quick thoughts* (Logeswaran and Lee, 2018). Quick thoughts is a strong representation learning baseline that is adopted in BERT (Devlin et al., 2019). We also include *principal component analysis (PCA)*, a classic representation learning and dimensionality reduction method, since bag-of-words representations are too expensive to work with for clustering analysis.

We compare three variants of AV-KMEANS that differ in the pretraining strategies. In addition to the AV-KMEANS systems pretrained with autoencoders and quick thoughts, we also consider a system whose encoder parameters are randomly initialized (*no pretraining*).

**Metrics** We compare the competitive approaches on a number of standard evaluation measures for clustering analysis. Following prior work (Kumar et al., 2011; Haponchyk et al., 2018; Xie et al., 2016), we set the number of clusters to the number of ground truth categories and report precision, recall, F1 score, and unsupervised clustering accuracy (ACC). To compute precision or recall, we assign each predicted cluster to the most frequent gold cluster or assign each gold cluster to the most frequent predicted cluster respectively. The F1 score is the harmonic average of the precision and recall. ACC uses a one-to-one assignment between the gold standard clusters and the predicted clusters. The assignment can be efficiently computed by the Hungarian algorithm (Kuhn, 1955).

**Parameter tuning** We empirically set both the dimension of the LSTM hidden state and the number of principal components in PCA to 300. The number of AV-KMEANS iterations $T$ and the number of k-means steps in a AV-KMEANS semi-iteration $M$ are set to 50 and 10 respectively, as we find that more iterations lead to similar cluster assignments. We adopt the same set of hyper-parameter values as used by Snell et al. (2017)

[8] We use the scikit-learn k-means implementation and the MVSC implementation available at: https://pypi.org/project/multiview/.

for training the prototypical networks. Specifically, we fix the number of query examples and the number of support examples to 15 and 5. The networks are trained for 100 episodes per AV-KMEANS semi-iteration. The number of sampled classes per episode is chosen to be 10, as it has to be smaller than the number of ground truth clusters. Adam (Kingma and Ba, 2015) is utilized to optimize the models and the initial learning rate is 0.001. During autoencoders or quick thoughts pretraining, we check the performance on the development set after each epoch to perform early stopping, where we randomly sample 10% unlabeled instances as the development data.

### 4.2 Results

Our main empirical findings are presented in Table 2, in which we compare AV-KMEANS with standard single-view and multi-view clustering algorithms. We also evaluate classic and neural approaches for representation learning, where the pretrained representations are fixed during k-means and MVSC clustering and they are fine-tuned during AV-KMEANS clustering. We analyze the empirical results in details in the following paragraphs.

**Utilizing multi-view information** Among all the systems, k-means clustering on representations trained with PCA or autoencoders only employs single-view information encoded in user query utterances. They clearly underperform the rest of the systems that leverage multi-view information of the entire conversations. Quick thoughts infuses the multi-view knowledge through the learning of the query-view vectors that are aware of the content-view semantics. In contrast, multi-view spectral clustering can work with representations that are separately learned for the individual views and the multi-view information is aggregated using the common eigenvectors of the data similarity Laplacian matrices. As shown, k-means clustering on quick thoughts vectors gives superior results than MVSC pretrained with PCA or autoencoders by more than 10% F1 or ACC, which indicates that multi-view representation learning is effective for problems beyond simple supervised learning tasks. Combining representation learning and multi-view clustering in a static way seems to be less ideal—MVSC performs worse than k-means using the quick thoughts vectors as clustering inputs. Multi-view representation learning breaks the independent-view assumption that is critical for classic multi-view clustering algorithms.

**Joint representation learning and clustering** We now investigate whether joint representation learning and clustering can reconcile the conflict between cross-view representation learning and classic multi-view clustering. AV-KMEANS outperforms k-means and MVSC baselines by considerable margins. It achieves 46% and 53% F1 scores and 40% and 44% ACC scores on the TwACS and AskUbuntu datasets, which are 5–30 percent higher than competitive systems. Compared to alternative methods, AV-KMEANS is able to effectively seek clustering-friendly representations that also encourage similar cluster assignments for different views of the same instances. With the help of quick thoughts pretraining, AV-KMEANS improves upon the strongest baseline, k-means clustering on quick thoughts vectors, by 4.5% ACC on the TwACS dataset and 12.2% F1 on the AskUbuntu dataset.

**Model pretraining for AV-KMEANS** Evaluation results on AV-KMEANS with different parameter initialization strategies are available in Table 2. As suggested, pretraining neural encoders is important for obtaining competitive results on the TwACS dataset, while its impact on the AskUbuntu dataset is less pronounced. AskUbuntu is six times larger than TwACS and models trained on AskUbuntu are less sensitive to their parameter initializations. This observation is consistent with early research on unsupervised pretraining, where Schmidhuber et al. (2012) argue that unsupervised initialization/pretraining is not necessary if a large amount of training data is available. Between the two pretraining methods, quick thoughts is much more effective than autoencoders—it improves upon no pretraining and autoencoders by 10.4% and 8.3% ACC scores on the TwACS dataset.

### 4.3 Error analysis

Our best performed system still fails to hit 50% F1 or ACC score on the TwACS dataset. We examine the outputs of the quick thoughts pretrained AV-KMEANS on TwACS, focusing on investigating the most frequent errors made by the system. To this end, we compute the confusion matrix based on the one-to-one assignment between the gold clusters and the predicted clusters used by ACC.

| Ground truth | Prediction | # Instances |
|---|---|---|
| Other | CustomerService | 21 |
| TerminalOp. | FlightDelay | 10 |
| FlightDelay | ChangeFlight | 10 |
| Other | FlightStaff | 10 |
| FlightEnter. | Other | 8 |

Table 3: The top 5 most frequent errors made by the quick thoughts pretrained AV-KMEANS on the TwACS dataset. The one-to-one assignment between the gold clusters and the predicted clusters is computed by the Hungarian algorithm.

The top 5 most frequent errors are presented in Table 3. As shown, three of the five errors involve Other. Instances under the Other category correspond to miscellaneous dialog intents, thereby they are less likely to be grouped together based on the semantic meaning representations.

The other two frequent errors confuse FlightDelay with TerminalOperation and ChangeFlight respectively. Poor terminal operations often incur unexpected customer delays. Two example query utterances are shown as follows,

$q_1$ : *who's running operation at mia flight 1088 been waiting for a gate.*
$q_2$ : *have been sitting in the plane waiting for our gate for 25 minutes.*

Sometimes, a user may express more than one intents in a single query utterance. For example, in the following query utterance, the user complaints about the delay and requests for an alternative flight:

$q$ : *why is ba flight 82 from abuja to london delayed almost 24 hours? and are you offering any alternatives?*

We leave multi-intent induction to future work.

## 5 Related Work

**User intent clustering** Automatic discovery of user intents by clustering user utterances is a critical task in understanding the dynamics of a domain with user generated content. Previous work focuses on grouping similar web queries or user questions together using supervised or unsupervised clustering techniques. Kathuria et al. (2010) perform simple k-means clustering on a variety of query traits to understand user intents. Cheung and Li (2012) present an unsupervised method for query intent clustering that produces a pattern consisting of a sequence of semantic concepts and/or lexical items for each intent. Jeon et al. (2005) use machine translation to estimate word translation probabilities and retrieve similar questions from question archives. A variation of k-means algorithm, MiXKmeans, is presented by Deepak (2016) to cluster threads that present on forums and Community Question Answering websites. Haponchyk et al. (2018) propose to cluster questions into intents using a supervised learning method that yields better semantic similarity modeling. Our work focuses on a related but different task that automatically induces user intents for building dialog systems. Two sources of information are naturally available for exploring our deep multi-view clustering approach.

**Multi-view clustering** Multi-view clustering (MVC) aims at grouping similar subjects into the same cluster by combining the available multi-view feature information to search for consistent cluster assignments across different views (Chao et al., 2017). Generative MVC approaches assume that the data is drawn from a mixture model and the membership information can be inferred using the multi-view EM algorithm (Bickel and Scheffer, 2004). Most of the works on MVC employ discriminative approaches that directly optimize an objective function that involves pairwise similarities so that the average similarity within clusters can be minimized and the average similarity between clusters can be maximized. In particular, Chaudhuri et al. (2009) propose to exploit canonical correlation analysis to learn multi-view representations that are then used for downstream clustering. Multi-view spectral clustering (Kumar et al., 2011; Kanaan-Izquierdo et al., 2018) constructs a similarity matrix for each view and then iteratively updates a matrix using the eigenvectors of the similarity matrix computed on another view. Standard MVC algorithms expect multi-view feature inputs that are fixed during unsupervised clustering. AV-KMEANS works with raw multi-view text inputs and learns representations that are particularly suitable for clustering.

**Joint representation learning and clustering** Several recent works propose to jointly learn feature representations and clustering via neural networks. Xie et al. (2016) present the deep embedded clustering (DEC) method that learns a map-

ping from the data space to a lower-dimensional feature space where it iteratively optimizes a KL divergence based clustering objective. Deep clustering network (DCN) (Yang et al., 2016) is a joint dimensional reduction and k-means clustering framework, in which the dimensional reduction model is implemented with a deep neural network. These methods focus on the learning of single-view representations and the multi-view information is under-explored. Lin et al. (2018) present a joint framework for deep multi-view clustering (DMJC) that is the closest work to ours. However, DMJC only works with single-view inputs and the feature representations are learned using a multi-view fusion mechanism. In contrast, AV-KMEANS assumes that the inputs can be naturally partitioned into multiple views and carry out learning with the multi-view inputs directly.

## 6 Conclusion

We introduce the novel task of dialog intent induction that concerns automatic discovery of dialog intents from user query utterances in human-human conversations. The resulting dialog intents provide valuable insights in helping design goal-oriented dialog systems. We propose to leverage the dialog structure to divide a dialog into two independent views and then present AV-KMEANS, a deep multi-view clustering algorithm, to jointly perform multi-view representation learning and clustering on the views. We conduct extensive experiments on a Twitter conversation dataset and a question intent clustering dataset. The results demonstrate the superiority of AV-KMEANS over competitive representation learning and multi-view clustering baselines. In the future, we would like to abstract multi-view data from multi-lingual and multi-modal sources and investigate the effectiveness of AV-KMEANS on a wider range of tasks in the multi-lingual or multi-modal settings.

## Acknowledgments

We thank Shawn Henry and Ethan Elenberg for their comments on an early draft of the paper. We also thank the conversational AI and the general research teams of ASAPP for their support throughout the project. We thank the EMNLP reviewers for their helpful feedback.

## References

Steffen Bickel and Tobias Scheffer. 2004. Multi-view clustering. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. 2018. Multiwoz-a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*.

Guoqing Chao, Shiliang Sun, and Jinbo Bi. 2017. A survey on multi-view clustering. *arXiv preprint arXiv:1712.06246*.

Kamalika Chaudhuri, Sham M Kakade, Karen Livescu, and Karthik Sridharan. 2009. Multi-view clustering via canonical correlation analysis. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Jackie Chi Kit Cheung and Xiao Li. 2012. Sequence clustering and labeling for unsupervised query intent discovery. In *Proceedings of the ACM international conference on Web search and data mining (WSDM)*.

Padmanabhan Deepak. 2016. Mixkmeans: Clustering question-answer archives. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Iryna Haponchyk, Antonio Uva, Seunghak Yu, Olga Uryupina, and Alessandro Moschitti. 2018. Supervised clustering of questions into intents for dialog system applications. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*.

Charles T Hemphill, John J Godfrey, and George R Doddington. 1990. The atis spoken language systems pilot corpus. In *Proceedings of the Workshop on Speech and Natural Language*.

Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014a. The second dialog state tracking challenge. In *Proceedings of the Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*.

Matthew Henderson, Blaise Thomson, and Steve Young. 2014b. Word-based dialog state tracking with recurrent neural networks. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*.

Jiwoon Jeon, W Bruce Croft, and Joon Ho Lee. 2005. Finding similar questions in large question and answer archives. In *Proceedings of the ACM international conference on Information and knowledge management (CIKM)*.

Samir Kanaan-Izquierdo, Andrey Ziyatdinov, and Alexandre Perera-Lluna. 2018. Multiview and multifeature spectral clustering using common eigenvectors. *Pattern Recognition Letters*, 102.

Ashish Kathuria, Bernard J Jansen, Carolyn Hafernik, and Amanda Spink. 2010. Classifying the user intent of web queries using k-means clustering. *Internet Research*, 20.

Seokhwan Kim, Luis Fernando D'Haro, Rafael E Banchs, Jason D Williams, Matthew Henderson, and Koichiro Yoshino. 2016. The fifth dialog state tracking challenge. In *Proceedings of the IEEE Spoken Language Technology Workshop (SLT)*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Harold W Kuhn. 1955. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2.

Abhishek Kumar, Piyush Rai, and Hal Daume. 2011. Co-regularized multi-view spectral clustering. In *Neural Information Processing Systems (NIPS)*.

Bingqian Lin, Yuan Xie, Yanyun Qu, Cuihua Li, and Xiaodan Liang. 2018. Jointly deep multi-view learning for clustering analysis. *arXiv preprint arXiv:1808.06220*.

Lajanugen Logeswaran and Honglak Lee. 2018. An efficient framework for learning sentence representations. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gasic, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2015. Multi-domain dialog state tracking using recurrent neural networks. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Eugene A Nida. 2015. *A componential analysis of meaning: An introduction to semantic structures*, volume 57. Walter de Gruyter GmbH & Co KG.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*.

Eldar Sadikov, Jayant Madhavan, Lu Wang, and Alon Halevy. 2010. Clustering query refinements by user intent. In *Proceedings of the international conference on World wide web*.

J Schmidhuber, U Meier, and D Ciresan. 2012. Multi-column deep neural networks for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Iulian Vlad Serban, Ryan Lowe, Peter Henderson, Laurent Charlin, and Joelle Pineau. 2018. A survey of available corpora for building data-driven dialogue systems: The journal version. *Dialogue & Discourse*, 9(1).

Darsh Shah, Tao Lei, Alessandro Moschitti, Salvatore Romeo, and Preslav Nakov. 2018. Adversarial domain adaptation for duplicate question detection. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *Neural Information Processing Systems (NIPS)*.

Jason Williams. 2012. A belief tracking challenge task for spoken dialog systems. In *Proceedings of the NAACL-HLT Workshop on Future directions and needs in the Spoken Dialog Community*.

Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. 2013. The dialog state tracking challenge. In *Proceedings of the SIGDIAL Conference*.

Junyuan Xie, Ross Girshick, and Ali Farhadi. 2016. Unsupervised deep embedding for clustering analysis. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Bo Yang, Xiao Fu, Nicholas D Sidiropoulos, and Mingyi Hong. 2016. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *Proceedings of the International Conference on Machine Learning (ICML)*.

# Nearly-Unsupervised Hashcode Representations for Biomedical Relation Extraction

**Sahil Garg**
USC ISI
sahilgar@usc.edu

**Aram Galstyan**
USC ISI
galstyan@isi.edu

**Greg Ver Steeg**
USC ISI
gregv@isi.edu

**Guillermo Cecchi**
IBM Research, NY
gcecchi@us.ibm.com

## Abstract

Recently, *kernelized locality sensitive hashcodes* have been successfully employed as representations of natural language text, especially showing high relevance to biomedical relation extraction tasks. In this paper, we propose to optimize the hashcode representations in a *nearly unsupervised* manner, in which we only use data points, but not their class labels, for learning. The optimized hashcode representations are then fed to a supervised classifier following the prior work. This nearly unsupervised approach allows fine-grained optimization of each hash function, which is particularly suitable for building hashcode representations generalizing from a training set to a test set. We empirically evaluate the proposed approach for biomedical relation extraction tasks, obtaining significant accuracy improvements w.r.t. state-of-the-art supervised and semi-supervised approaches.

## 1 Introduction

In natural language processing, one important but a highly challenging task is of identifying biological entities and their relations from biomedical text, as illustrated in Fig. 1, relevant for real world problems, such as personalized cancer treatments (Rzhetsky, 2016; Hahn and Surdeanu, 2015; Cohen, 2015). In the previous works, the task of biomedical relation extraction is formulated as of binary classification of natural language structures; one of the primary challenges to solve the problem is that the number of data points annotated with class labels (in a training set) is small due to high cost of annotations by biomedical domain experts, and further, bio-text sentences in a test set can vary significantly w.r.t. the ones from a training set due to practical aspects, such as high diversity of research topics or writing styles, hedging, etc. Considering such challenges for the task, many classification models based on kernel similarity functions have been proposed (Garg et al., 2016; Chang et al., 2016; Tikk et al., 2010; Miwa et al., 2009; Airola et al., 2008; Mooney and Bunescu, 2005), and recently, many neural networks based classification models have also been explored (Kavuluru et al., 2017; Peng and Lu, 2017; Hsieh et al., 2017; Rao et al., 2017; Nguyen and Grishman, 2015), including the ones doing adversarial learning using the knowledge of data points (excluding class labels) from a test set, or semi-supervised variational autoencoders (Rios et al., 2018; Ganin et al., 2016; Zhang and Lu, 2019; Kingma et al., 2014).

In a very recent work, kernelized locality sensitive hashcodes based representation learning approach has been proposed that has shown to be the most successful in terms of accuracy and computational efficiency for the task (Garg et al., 2019). The model parameters, shared between all the hash functions, are optimized in a supervised manner, whereas an individual hash function is constructed in a *randomized* fashion. The authors suggest to obtain thousands of (randomized) semantic features extracted from natural language data points into binary hashcodes, and then making classification decision as per the *features* using hundreds of decision trees, which is the core of their robust classification approach. Even if we extract thousands of semantic features using the hashing approach, it is difficult to ensure that the features extracted from training data points would generalize to a test set. While the inherent randomness in construct-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4026–4036,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

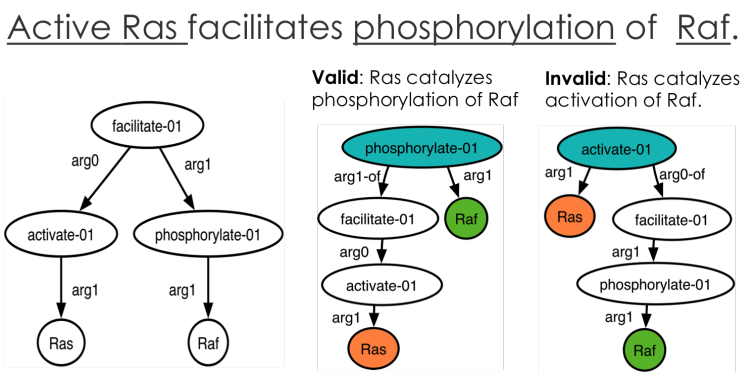


Figure 1: On the left, we show an abstract meaning representation (AMR) of a sentence. As per the semantics of the sentence, there is a valid biomedical relationship between the two proteins, Ras and Raf, i.e. Ras catalyzes phosphorylation of Raf; the relation corresponds to a subgraph extracted from the AMR. On the other hand, one of the many invalid biomedical relationships that one could infer is, Ras catalyzes activation of Raf, for which we show the corresponding subgraph too. A given candidate relation automatically hypothesized from the sentence, is binary classified, as valid or invalid, using the subgraph as features.

ing hash functions from a training set can help towards generalization in the case of absence of a test set, there should be better alternatives if we do have the knowledge of a test set of data points, or a subset of a training set treated as a *pseudo-test set*. What if we construct hash functions in an intelligent manner via exploiting the additional knowledge of unlabeled data points in a test/pseudo-test set, performing *fine-grained optimization of each hash function* rather than relying upon randomness, so as to extract semantic features which generalize?

Along these lines, we propose a new framework for learning hashcode representations accomplishing two important (inter-related) extensions w.r.t. the previous work:

(a) We propose to use a *nearly unsupervised* hashcode representation learning setting, in which we use only the knowledge of which set a data point comes from, a training set or a test/pseudo-test set, along with the data point itself, whereas the actual class labels of data points from a training set are input only to the final supervised-classifier, such as a Random Forest, which takes input of the learned hashcodes as representation (feature) vectors of data points along with their class labels;

(b) We introduce multiple concepts for *fine-grained (discrete) optimization* of hash functions, employed in our novel *information-theoretic algorithm* that constructs hash functions greedily one by one. In supervised settings, fine-grained (greedy) optimization of hash functions could lead to overfitting whereas, in our proposed nearly-unsupervised framework, it allows *flexibility for explicitly maximizing the generalization capabilities* of hash functions.

For a task of biomedical relation extraction, we evaluate our approach on four public datasets, and obtain significant gains in F1 scores w.r.t. state-of-the-art models including kernel-based approaches as well the ones based on semi-supervised learning of neural networks. We also show how to employ our framework for learning locality sensitive hashcode representations using neural networks.[1]

## 2 Problem Formulation & Background

In Fig. 1, we demonstrate how biomedical relations between entities are extracted from the semantic (or syntactic) parse of a sentence. As we see, the task is formulated as of binary classification of natural language substructures extracted from the semantic parse. Suppose we have natural language structures, $\boldsymbol{S} = \{S_i\}_1^N$, such as parse trees, shortest paths, text sentences, etc, with corresponding class labels, $\boldsymbol{y} = \{y_i\}_1^N$. For the data points coming from a training set and a test set, we use notations, $\boldsymbol{S}_T$, and $\boldsymbol{S}_*$, respectively; same applies for the class labels. In addition, we define

[1] Code: https://github.com/sgarg87/nearly_unsupervised_hashcode_representations

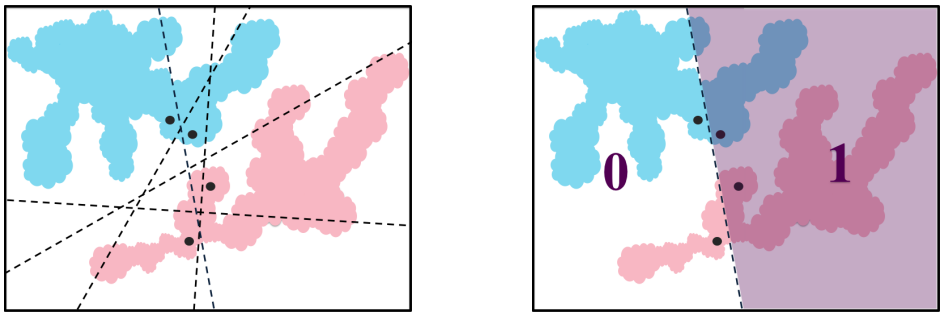


(a) Hash function choices (b) Generalizing hash function

Figure 2: In this figure, we illustrate how to construct hash functions which generalize across a training and a test set. Blue and red colors denote training and test data points respectively. A hash function is constructed by splitting a very small subset of data points into two parts. In Fig. 2(a), given the set of four data points selected randomly, there are many choices possible for splitting the set, corresponding to difference choices of hash functions which are denoted with dashed lines. The optimal choice of hash function is shown in Fig. 2(b) since it generalizes well, assigning same value to many data points from both training & test sets.

indicator variable, $\boldsymbol{x} = \{x_i\}_{i=1}^N$, for $\boldsymbol{S}$, with $x_i \in \{0, 1\}$ denoting if a data point $S_i$ is coming from a test/pseudo-test set or a training set. Our goal is to infer the class labels of the data points from a test set, $\boldsymbol{y}_*$.

### 2.1 Hashcode Representations

As per the hashcode representation approach, $\boldsymbol{S}$ is mapped to a set of *locality sensitive* hashcodes, $\boldsymbol{C} = \{\boldsymbol{c}_i\}_1^N$, using a set of $H$ binary hash functions, i.e. $\boldsymbol{c}_i = \boldsymbol{h}(S_i) = \{h_1(S_i), \cdots, h_H(S_i)\}$. $h_l(.;\boldsymbol{\theta})$ is constructed such that it splits a set of data points, $\boldsymbol{S}_l^R$, into two subsets as shown in Fig. 2(a), while choosing the set as a small random subset of size $\alpha$ from the superset $\boldsymbol{S}$, i.e. $\boldsymbol{S}_l^R \subset \boldsymbol{S}$ s.t. $|\boldsymbol{S}_l^R| = \alpha \ll N$. In this manner, we can construct a large number of hash functions, $\{h_l(.;\boldsymbol{\theta})\}_{l=1}^H$, from a *reference set* of size $M$, $\boldsymbol{S}^R = \{\boldsymbol{S}_1^R \cup \cdots \cup \boldsymbol{S}_H^R\}, |\boldsymbol{S}^R| = M \leq \alpha H \ll N$.

While, mathematically, a locality sensitive hash function can be of any form, kernelized hash functions (Garg et al., 2019, 2018; Joly and Buisson, 2011), rely upon a convolution kernel similarity function $K(S_i, S_j; \boldsymbol{\theta})$ defined for any pair of structures $S_i$ and $S_j$ with kernel-parameters $\boldsymbol{\theta}$ (Haussler, 1999). To construct $h_l(.)$, a kernel-trick based model, such as kNN, SVM, is fit to $\{\boldsymbol{S}_l^R, \boldsymbol{z}_l\}$, with a randomly sampled binary vector, $\boldsymbol{z}_l \in \{0, 1\}^\alpha$, that defines the split of $\boldsymbol{S}_l^R$. For computing hashcode $\boldsymbol{c}_i$ for $S_i$, it requires only $M$ number of convolution-kernel similarities of $S_i$ w.r.t. the data points in $\boldsymbol{S}^R$, which makes this approach highly scalable in compute cost terms.

In the previous work (Garg et al., 2019), it is proposed to optimize all the hash functions jointly by learning only the *parameters which are shared amongst all the functions*, i.e. learning kernel parameters, $\boldsymbol{\theta}$ and the choice of reference set, $\boldsymbol{S}^R \subset \boldsymbol{S}_T$. This optimization is performed in a supervised manner via maximization of the mutual information between hashcodes of data points and their class labels, using $\{\boldsymbol{S}_T, \boldsymbol{y}_T\}$ for training.

## 3 Nearly-Unsupervised Hashcode Representations

Our key insight in regards to limitation of the previous approach for supervised learning of hashcode representations, is that, to avoid overfitting, learning is intentionally restricted only to the optimization of shared parameters whereas each hash function $h_l(.)$ is constructed in a randomized manner, i.e. random sub-sampling of a subset, $\boldsymbol{S}_l^R \subset \boldsymbol{S}$, and a random split of the subset. On the other hand, in a nearly-unsupervised hashcode learning settings as we introduce next, we can have the additional knowledge of data points from a test/pseudo-test set which can be leveraged to extend the optimization from the shared (global) parameters to fine-grained optimization of hash functions, not only to avoid overfitting but for higher generalization of hash-

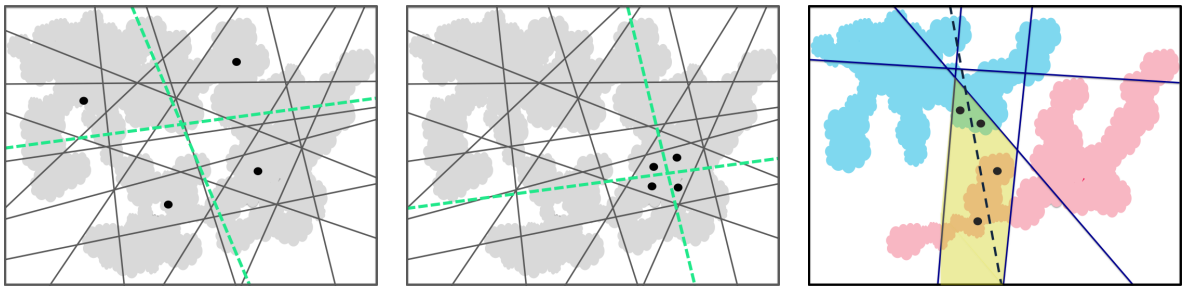

(a) Global sampling (b) Local sampling from a cluster (c) Local sampling from a high-entropy cluster

Figure 3: In all the three figures, dark-gray lines denote hash functions optimized previously, and intersections of the lines give us 2-D cells denoting hashcodes as well as clusters; black dots represent a set of data points which are used to construct a hash function, and some of the choices to split the subset are shown as green dashed lines. The procedure of sampling the subset varies across the three figures. In Fig. 3(a), since the four data points from global sampling have already unique hashcodes (2-D cells), the newly constructed hash function (one of the green dashed-lines) adds little information to their representations. On the other hand in Fig. 3(b), a hash function constructed from a set of data points, which are sampled locally from within a cluster, puts some of the data points in the sampled set into two different cells (hashcodes), so adding more fine-grained information to their representations, hence more advantageous from representation learning perspective. In Fig. 3(c), training and test data points are denoted with blue and red colors respectively, and the data points to construct a new hash function are sampled locally from within a high entropy cluster, i.e. the one containing a balanced proportion of the training & test data points.

codes across training & test sets.

**Nearly unsupervised learning settings**
We propose to learn hash functions, $\boldsymbol{h}(.)$, using $\boldsymbol{S} = \{\boldsymbol{S}_T, \boldsymbol{S}_*\}$, $\boldsymbol{x}$, and optionally, $\boldsymbol{y}_T$. Herein, $\boldsymbol{S}_*$ is a test set, or a pseudo-test set that can be a random subset of the training set or a large set of unlabeled data points outside the training set.

### 3.1 Basic Concepts for Fine-Grained Optimization

In the prior works on kernel-similarity based locality sensitive hashing, the first step for constructing a hash function is to randomly sample a small subset of data points, from a superset $\boldsymbol{S}$, and in the second step, the subset is split into two parts using a *kernel-trick* based model (Garg et al., 2019, 2018; Joly and Buisson, 2011), serving as the hash function, as described in Sec. 2.

In the following, we introduce basic concepts for improving upon these two key aspects of constructing a hash function, while later, in Sec. 3.2, these concepts are incorporated in a unified manner in our proposed *information-theoretic algorithm that greedily optimizes hash functions* one by one.

**Informative split**
In Fig. 2(a), construction of a hash function is pictorially illustrated, showing multiple possible splits, as dotted lines, of a small set of four data points (black dots). (Note that a hash function is shown to be a linear hyperplane only for simplistic explanations of the basic concepts.) While in the previous works, one of the many choices for splitting the set is chosen randomly, we propose to optimize upon this choice. Intuitively, one should choose a split of the set, corresponding to a hash function, such that it gives a balanced split for the whole set of data points, and it should also generalize across training & test sets. In reference to the figure, one simple way to analyze the generalization of a split (so the hash function) is to see if there are training as well as test data points (or pseudo-test data points) on either side of the dotted line. As per this concept, an optimal split of the set of four data points is shown in Fig. 2(b).

Referring back to Sec. 2, clearly, this is a combinatorial optimization problem, where we need to choose an optimal choice of $\boldsymbol{z}_l \in \{0,1\}^\alpha$ for set, $\boldsymbol{S}_l^R$, to construct $h_l(.)$. For a small value of $\alpha$, one can either go through

all the possible combinations in a brute force manner, or use Markov Chain Monte Carlo sampling. It is interesting to note that, even though a hash function is constructed from a very small set of data points (of size $\alpha \not\gg 1$), the generalization criterion, formulated in our info-theoretic objective introduced in Sec. 3.2, is computed using all the data points available for the optimization, $\boldsymbol{S}$.

**Local sampling from a cluster**

Another aspect of constructing a hash function, having a scope for improvement, is sampling of a small subset of data points, $\boldsymbol{S}_l^R \subset \boldsymbol{S}$, that is used to construct a hash function. In the prior works, the selection of such a subset is purely random, i.e. random selection of data points globally from $\boldsymbol{S}$. In Fig. 3, we illustrate that, it is wiser to (randomly) *select data points locally from one of the clusters* of the data points in $\boldsymbol{S}$, rather than sampling globally from $\boldsymbol{S}$. Here, we propose that clustering of all the data points in $\boldsymbol{S}$ can be obtained using the hash functions itself, due to their locality sensitive property. While using a large number of locality sensitive hash functions give us fine-grained representations of data points, *a small subset of the hash functions, of size $\zeta$, defines a valid clustering* of the data points, since data points which are similar to each other should have same *hashcodes serving as cluster labels*.

From this perspective, we can construct first few hash functions from global sampling of data points, what we refer as global hash functions. These global hash functions should serve to provide hashcode representations as well as clusters of data points. Then, via local sampling from the clusters, we can also construct *local hash functions* to capture more finer details of data points. As per this concept, we can learn *hierarchical (multi-scale) hashcode representations* of data points, capturing differences between data points from coarser (global hash functions) to finer scales (local hash functions).

Further, we suggest to choose a cluster that has a balanced proportion of training & test (pseudo-test) data points, which is desirable from the perspective of having generalized hashcode representations; see Fig. 3(c).

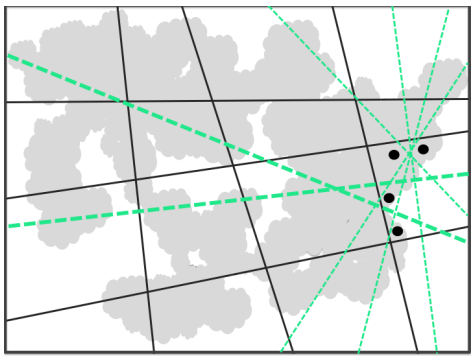

Figure 4: Dark-gray lines denote hash functions optimized previously, and gray color represents data points used for the optimization. The intersections of the lines give us 2-D cells, corresponding to hashcodes as well as clusters. For the set of four data points sampled within a cluster, there are many choices to split it (corresponding to hash functions choices), shown with dashed lines. We propose to choose the one which also splits the sets of data points in other (neighboring) clusters in a balanced manner, i.e. having data points in a cluster on either side of the dashed line and cutting through as many clusters as possible. As per this criterion, the thick-green dashed lines are superior choices w.r.t. the thin-green ones.

**Splitting other clusters**

In reference to Fig. 4, *non-redundancy* of a hash function w.r.t. the other hash functions can be characterized in terms of how well the hash function splits the clusters defined as per the other hash functions.

Next we mathematically formalize all the concepts introduced above for fine-grained optimization of hash functions into an information-theoretic objective function.

### 3.2 Information-Theoretic Learning

We optimize hash functions greedily, one by one. Referring back to Sec. 2, we define binary random variable $x$ denoting if a data point comes from a training set or a test/pseudo-test set. In a greedy step of optimizing a hash function, random variable, $\boldsymbol{c}$, represents the hashcode of a data point $S$, as per the previously optimized hash functions $\boldsymbol{h}_{l-1}(.)$. Along same lines, $c$ denotes the binary random variable corresponding from the present hash function under optimization, $h_l(.)$. We maximize the information-theoretic objective as below.

**Algorithm 1** Nearly-Unsupervised Hashcode Representation Learning

**Require:** Dataset $\{\boldsymbol{S}, \boldsymbol{x}\}$, and parameters, $H, \boldsymbol{\alpha}, \zeta$.

$\boldsymbol{h}(.) \leftarrow \{\}$, $\boldsymbol{C} \leftarrow \{\}$, $\boldsymbol{f} \leftarrow \{\}$
*%Greedy step for optimizing hash function, $h_l(.)$*
**while** $|\boldsymbol{h}(.)| < H$ **do**
  $\alpha_l \leftarrow$ sampleSubsetSize$(\boldsymbol{\alpha})$ *%Sample a size for the subset of data points, $\boldsymbol{S}_l^R$, to construct a hash function*
  **if** $|\boldsymbol{h}(.)| < \zeta$ **then**
    $\boldsymbol{S}_l^R \leftarrow$ randomSampleGlobally$(\boldsymbol{S}, \alpha_l)$ *%Randomly sample data points, globally from $\boldsymbol{S}$, for constructing a global hash function.*
  **else**
    $\boldsymbol{S}_l^R \leftarrow$ randomSampleLocally$(\boldsymbol{S}, \boldsymbol{C}, \boldsymbol{x}, \alpha_l, \zeta)$ *%Sample data points randomly from, a high entropy cluster, for constructing a local hash function.*
  **end if**
  $h_l(.), f_l \leftarrow$ optimizeSplit$\big(\boldsymbol{S}_l^R, \boldsymbol{S}, \boldsymbol{C}, \boldsymbol{x}\big)$ *%Optimize split of $\boldsymbol{S}_l^R$.*
  $\boldsymbol{c} \leftarrow$ computeHash$(\boldsymbol{S}, h_l(.))$
  $\boldsymbol{C} \leftarrow \boldsymbol{C} \cup \boldsymbol{c}$, $\boldsymbol{h}(.) \leftarrow \boldsymbol{h(.)} \cup h_l(.)$, $\boldsymbol{f} \leftarrow \boldsymbol{f} \cup f_l$.
  $\boldsymbol{h}(.), \boldsymbol{C}, \boldsymbol{f} \leftarrow$ deleteLowInfoFunc$(\boldsymbol{h}(.), \boldsymbol{C}, \boldsymbol{f})$ *%delete hash functions from the set with lower objective values*
**end while**
**Return** $\boldsymbol{h}(.), \boldsymbol{C}$.

$$\underset{h_l()}{\operatorname{argmax}}\, \mathcal{H}(x, c) - \mathcal{I}(c : \boldsymbol{c}) + \mathcal{H}(x|\boldsymbol{c}); \qquad (1)$$
$$\boldsymbol{c} = \boldsymbol{h}_{l-1}(S), c = h_l(S)$$

Herein the optimization of a hash function, $h_{(.)}$, involves intelligent selection of $\boldsymbol{S}_l^R$, an informative split of $\boldsymbol{S}_l^R$, i.e. optimizing $\boldsymbol{z}_l$ for $\boldsymbol{S}_l^R$, and learning of the parameters $\boldsymbol{\theta}$ of a (kernel or neural) model, which is fit on $\{\boldsymbol{S}_l^R, \boldsymbol{z}_l\}$, acting as the hash function.

In the objective function above, maximizing the first term, $\mathcal{H}(x, c)$, i.e. joint entropy on $x$ and $c$, corresponds to the concept of *informative split* described above in Sec. 3.1; see Fig. 2. This term is cheap to compute since $x$ and $c$ are both 1-dimensional binary variables.

The second term in the objective, the mutual information term, ensures minimal redundancies between hash functions. This is related to the concept of constructing a hash function such that it splits many of the existing clusters, as mentioned above in Sec. 3.1; see Fig. 4. This mutual information function can be computed using the approximation in the previous work by (Garg et al., 2019).

The last quantity in the objective is $\mathcal{H}(x|\boldsymbol{c})$, conditional entropy on $x$ given $\boldsymbol{c}$. We propose to maximize this term indirectly via *choosing a cluster informatively, from which to randomly select data points for constructing the hash function*, such that it contains a balanced ratio of the count of training & test data points, i.e. a cluster with high entropy on $x$, which we refer as a *high entropy cluster*. In reference to Fig. 3(c), the new clusters emerging from a split of a high entropy cluster should have higher chances to be high entropy clusters themselves, thus maximizing the last term indirectly. We compute marginal entropy on $x$ for each cluster, and an explicit computation of $H(x|\boldsymbol{c})$ is not required.

Optionally one may extend the objective to include the term, $-\mathcal{H}(y|\boldsymbol{c}, c)$, with $y$ denoting the random variable for a class label.

The above described learning framework is summarized in Alg. 1.

Besides kernelized locality sensitive hashcodes, the above framework allows neural locality sensitive hashing. One can fit any (regularized) neural model on $\{\boldsymbol{S}_l^R, \boldsymbol{z}_l\}$, acting as a *neural locality sensitive hash function*. We expect that some of the many possible choices for a split of $\boldsymbol{S}_l^R$ should lead to natural semantic categorizations of the data points. For such a natural split choice, even a parameterized

model can act as a good hash function without overfitting as we observed empirically.

In the algorithm, we also propose to *delete some of the hash functions* from the set of optimized ones, the ones which have low objective function values w.r.t. the rest. This step provides robustness against an arbitrarily bad choice of *randomly* selected subset, $\boldsymbol{S}_l^R$.

Our algorithm allows parallel computing, as in the previous hashcode learning approach.

## 4 Experiments

We demonstrate the applicability of our approach for a challenging task of biomedical relation extraction, using four public datasets.

**Dataset details**

For AIMed and BioInfer, cross-corpus evaluations have been performed in many previous works (Airola et al., 2008; Tikk et al., 2010; Peng and Lu, 2017; Hsieh et al., 2017; Rios et al., 2018; Garg et al., 2019). These datasets have annotations on pairs of interacting proteins (PPI) in a sentence while ignoring the interaction type. Following the previous works, for a given pair of proteins mentioned in a text sentence from a training or a test set, we obtain the corresponding undirected shortest path from a Stanford dependency parse of the sentence, that serves as a data point.

We also use PubMed45 and BioNLP datasets which have been used for extensive evaluations in recent works (Garg et al., 2019, 2018; Rao et al., 2017; Garg et al., 2016). These two datasets consider a relatively more difficult task of inferring an interaction between two or more bio-entities mentioned in a sentence, along with the inference of their interaction-roles, and the type of interaction from an unrestricted list. As in the previous works, we use abstract meaning representation (AMR) to obtain shortest path-based data points (Banarescu et al., 2013); same bio-AMR parser (Pust et al., 2015) is employed as in the previous works. PubMed45 dataset has 11 subsets, with evaluation performed for each of the subsets as a test set leaving the rest for training (not to be confused with cross-validation). For BioNLP dataset (Kim et al., 2009, 2011; Nédellec et al., 2013), the training set contains annotations from years 2009, 2011, 2013, and the test set contains development set from year 2013. Overall, for a fair comparison of the models, we keep same experimental setup as followed in (Garg et al., 2019), for all the four datasets, so as to avoid any bias due to engineering aspects; evaluation metrics for the relation extraction task are, f1 score, precision, recall.

**Baseline methods**

The most important baseline method for the comparison is the recent work of supervised hashcode representations (Garg et al., 2019). Their model is called as *KLSH-RF*, with KLSH referring to kernelized locality sensitive hashcodes, and RF denotes Random Forest. Our approach differs from their work in the sense that our hashcode representations are nearly unsupervised, whereas their approach is purely supervised, while both approaches use a supervised RF. We refer to our model as *KLSH-NU-RF*. Within the nearly unsupervised learning setting, we consider transductive setting by default, i.e. using data points from both training and test sets. Later, we also show results for inductive settings, i.e. using a random subset of training data points, as a pseudo-test set. In both scenarios, we do not use class labels for learning hashcodes, but only for training RF.

For AIMed and BioInfer datasets, adversarial learning based four neural network models had been evaluated in the prior works (Rios et al., 2018; Ganin et al., 2016), referred as *CNN-RevGrad, Bi-LSTM-RevGrad, Adv-CNN, Adv-CNN*. Like our model KLSH-NU-RF, these four models are also learned in transductive settings of using data points from the test set in addition to the training set. Semi-supervised Variational Autoencoders (*SSL-VAE*) have also been explored for biomedical relation extraction (Zhang and Lu, 2019), which we evaluate ourselves for all the four datasets considered in this paper.

**Parameter settings**

We use path kernels with word vectors & kernel parameter settings as in the previous work (Garg et al., 2019). From a preliminary tuning, we set the number of hash functions, $H = 100$, and the number of decision trees in a Random Forest classifier, $R = 100$; these parameters are not sensitive, requiring mini-

| Models | (A, B) | (B, A) |
|---|---|---|
| SVM (Airola08) | 0.47 | 0.47 |
| SVM (Miwa09) | 0.53 | 0.50 |
| SVM (Tikk10) | 0.41 (0.67, 0.29) | 0.42 (0.27, 0.87) |
| CNN (Nguyen15) | 0.37 | 0.45 |
| Bi-LSTM (Kavuluru17) | 0.30 | 0.47 |
| CNN (Peng17) | 0.48 (0.40, 0.61) | 0.50 (0.40, 0.66) |
| RNN (Hsieh17) | 0.49 | 0.51 |
| CNN-RevGrad (Ganin16) | 0.43 | 0.47 |
| Bi-LSTM-RevGrad (Ganin16) | 0.40 | 0.46 |
| Adv-CNN (Rios18) | 0.54 | 0.49 |
| Adv-Bi-LSTM (Rios18) | **0.57** | 0.49 |
| KLSH-kNN (Garg18) | 0.51 (0.41, 0.68) | 0.51 (0.38, 0.80) |
| KLSH-RF (Garg19) | **0.57** (0.46, 0.75) | 0.54 (0.37, 0.95) |
| SSL-VAE (Zhang19) | 0.50 (0.38, 0.72) | 0.46 (0.39, 0.57) |
| **KLSH-NU-RF** | **0.57** (0.44, 0.81) | **0.57** (0.44, 0.81) |

Table 1: Evaluation results from cross-corpus evaluation for (train, test) pairs of datasets, AIMed (A) and BioInfer (B). For each model, we report F1 score, and if available, precision, recall scores are also shown in brackets. For the adversarial neural models by (Ganin et al., 2016), evaluation on the datasets was provided by (Rios et al., 2018).

| Models | PubMed45 | BioNLP |
|---|---|---|
| SVM (Garg16) | 0.45±0.25 (0.58, 0.43) | 0.46 (0.35, 0.67) |
| LSTM (Rao17) | N.A. | 0.46 (0.51, 0.44) |
| LSTM (Garg19) | 0.30±0.21 (0.38, 0.28) | 0.59 (0.89, 0.44) |
| Bi-LSTM (Garg19) | 0.46±0.26 (0.59, 0.43) | 0.55 (0.92, 0.39) |
| LSTM-CNN (Garg19) | 0.50±0.27 (0.55, 0.50) | 0.60 (0.77, 0.49) |
| CNN (Garg19) | 0.51±0.28 (0.46, 0.46) | 0.60 (0.80, 0.48) |
| KLSH-kNN (Garg18) | 0.46±0.21 (0.44, 0.53) | 0.60 (0.63, 0.57) |
| KLSH-RF (Garg19) | 0.57±0.25 (0.63, 0.55) | 0.63 (0.78, 0.53) |
| SSL-VAE (Zhang19) | 0.40 ± 0.16 (0.33, 0.69) | 0.48 (0.43, 0.56) |
| **KLSH-NU-RF** | **0.61±0.23 (0.61, 0.62)** | **0.67 (0.73, 0.61)** |

Table 2: Evaluation results for PubMed45 and BioNLP datasets. We report F1 score (mean ± standard deviation), and mean-precision & mean-recall numbers in brackets. For BioNLP, we standard deviation numbers are not provided as there is one fixed test subset.

mal tuning. For any other parameters which may require fine-grained tuning, we use 10% of training data points, selected randomly, for validation. Within kernel locality sensitive hashing, we choose between Random Maximum Margin and Random k-Nearest Neighbors techniques, and for neural locality sensitive hashing, we use a simple 2-layer LSTM model with 8 units per layer. In our nearly unsupervised learning framework, we use subsets of the hash functions, of size 10, to obtain clusters ($\zeta = 10$). We employ 8 cores on an i7 processor, with 32GB memory, for all the computations.

### 4.1 Experimental Results

In summary, our model KLSH-NU-RF significantly outperforms its purely supervised counterpart, KLSH-RF, and also semi-supervised neural network models.

In reference to Tab. 1, we first discuss results for the evaluation setting of using AIMed dataset as a test set, and BioInfer as a training set. We observe that our model, KLSH-NU-RF, obtains F1 score, 3 pts higher w.r.t. the most recent baseline, KLSH-RF. In comparison to the semi-supervised neural neural models, CNN-RevGrad, Bi-LSTM-RevGrad, Adv-CNN, Adv-Bi-LSTM, SSL-VAE, which use the knowledge of a test set just like our model, we gain 8-11 pts in F1 score. On the other hand, when evaluating on BioInfer dataset as a test set and AIMed as a training set, our model is in tie w.r.t. the adversarial neural model, Adv-Bi-LSTM, though outperforming the other three adversarial models and SSL-VAE, by large margins in F1 score. In comparison to KLSH-RF, we retain same F1 score, while gaining in recall by 6 pts at the cost of losing 2 pts in precision.

For PubMed and BioNLP datasets, there is no prior evaluation of adversarial models. Nevertheless, in Tab. 2, we see that our model significantly outperforms SSL-VAE, and it also outperforms the most relevant baseline, KLSH-RF, gaining F1 score by 4 pts for both the datasets.[2] Note that, high standard deviations reported for PubMed45 dataset are due to high diversity across the 11 test sets,

[2]These two datasets have high importance to gauge practical relevance of a model for the task of biomedical relation extraction.

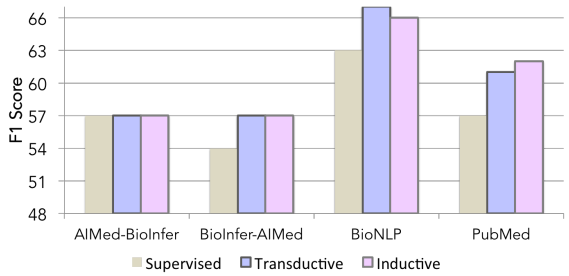


Figure 5: Comparison across the three types of learning settings, supervised, transductive, and inductive.

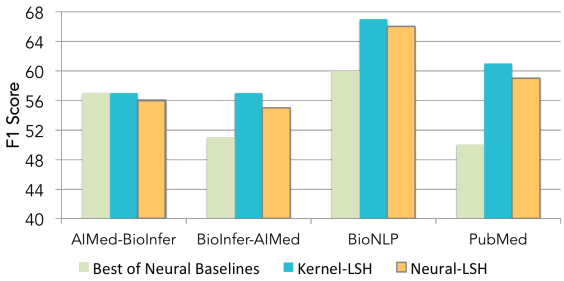


Figure 6: Comparison of neural hashing w.r.t. kernel hashing, and the best of neural baselines.

while the performance of our model for a given test set is highly stable (improvements are statistically significant with p-value: 6e-3).

One general trend to observe is that the proposed nearly unsupervised learning approach leads to a significantly higher recall, at the cost of marginal drop in precision, w.r.t. its supervised baseline.

Further, note that the number of hash functions used in the prior work is 1000 whereas we use only 100 hash functions. Compute time is same as for their model.

Our approach is easily extensible for other modeling aspects such as non-stationary kernel functions, document level inference, joint use of semantic & syntactic parses, ontology or database usage (Garg et al., 2019, 2016; Alicante et al., 2016), though we refrain from presenting system level evaluations, and have focused only upon analyzing improvements from our principled extension of the recently proposed technique that has already been shown to be successful for the task.

**Transductive vs inductive settings**

In the above discussed results, hashcode representations in our models are learned in transductive setting. For inductive settings, we randomly select 25% of the training data points for use as a pseudo-test set instead of the test set. In Fig. 5, we observe that both inductive and transductive settings are more favorable w.r.t. the supervised one, KLSH-RF, which is the baseline in this paper. F1 scores obtained from the inductive setting are on a par with the transductive settings. It is worth noting that, in inductive settings, our model is trained on *information even lesser than the baseline model* KLSH-RF, yet it obtains F1 scores significantly higher.

**Neural hashing**

In Fig. 6, we show results for neural locality sensitive hashing within our proposed framework, and observe that neural hashing is a little worse than its kernel based counterpart, however it performs significantly superior w.r.t. the best of other neural models.

## 5 Conclusions

We proposed a nearly-unsupervised framework for learning of kernelized locality sensitive hashcode representations, a recent technique, that was originally supervised, which has shown state-of-the-art results for a difficult task of biomedical relation extraction. Within our proposed framework, we use the additional knowledge of test/pseudo-test data points for fine-grained optimization of hash functions so as to obtain hashcode representations generalizing across training & test sets. Our experiment results show significant improvements in accuracy numbers w.r.t. the supervised baseline, as well as semi-supervised neural network models, for the same task of bio-medical relation extraction across four public datasets.

## 6 Acknowledgments

This work was sponsored by the DARPA Big Mechanism program (W911NF-14-1-0364). It is our pleasure to acknowledge fruitful discussions with Shushan Arakelyan, Amit Dhurandhar, Sarik Ghazarian, Palash Goyal, Karen Hambardzumyan, Hrant Khachatrian, Kevin Knight, Daniel Marcu, Dan Moyer, Kyle Reing, and Irina Rish. We are also grateful to anonymous reviewers for their valuable feedback.

## References


Antti Airola, Sampo Pyysalo, Jari Björne, Tapio Pahikkala, Filip Ginter, and Tapio Salakoski. 2008. All-paths graph kernel for protein-protein interaction extraction with evaluation of cross-corpus learning. *BMC Bioinformatics*.

Anita Alicante, Massimo Benerecetti, Anna Corazza, and Stefano Silvestri. 2016. A distributed architecture to integrate ontological knowledge into information extraction. *International Journal of Grid and Utility Computing*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the Linguistic Annotation Workshop and Interoperability with Discourse*.

Yung-Chun Chang, Chun-Han Chu, Yu-Chen Su, Chien Chin Chen, and Wen-Lian Hsu. 2016. PIPE: a protein–protein interaction passage extraction module for biocreative challenge. *Database*.

Paul R Cohen. 2015. DARPA's big mechanism program. *Physical biology*.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, Francois Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *Journal of Machine Learning*.

Sahil Garg, Aram Galstyan, Ulf Hermjakob, and Daniel Marcu. 2016. Extracting biomolecular interactions using semantic parsing of biomedical text. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Sahil Garg, Aram Galstyan, Greg Ver Steeg, Irina Rish, Guillermo Cecchi, and Shuyang Gao. 2019. Kernelized hashcode representations for relation extraction. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Sahil Garg, Greg Ver Steeg, and Aram Galstyan. 2018. Stochastic learning of nonstationary kernels for natural language modeling. *arXiv preprint arXiv:1801.03911*.

Marco A Valenzuela-Escárcega Gus Hahn and Powell Thomas Hicks Mihai Surdeanu. 2015. A domain-independent rule-based framework for event extraction. In *Proceedings of ACL-IJCNLP*.

David Haussler. 1999. Convolution kernels on discrete structures. Technical report.

Yu-Lun Hsieh, Yung-Chun Chang, Nai-Wen Chang, and Wen-Lian Hsu. 2017. Identifying protein-protein interactions in biomedical literature using recurrent neural networks with long short-term memory. In *Proceedings of the International Joint Conference on Natural Language Processing*.

Alexis Joly and Olivier Buisson. 2011. Random maximum margin hashing. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*.

Ramakanth Kavuluru, Anthony Rios, and Tung Tran. 2017. Extracting drug-drug interactions with word and character-level recurrent neural networks. In *IEEE International Conference on Healthcare Informatics*.

Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. 2009. Overview of BioNLP'09 shared task on event extraction. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing: Shared Task*.

Jin-Dong Kim, Sampo Pyysalo, Tomoko Ohta, Robert Bossy, Ngan Nguyen, and Jun'ichi Tsujii. 2011. Overview of BioNLP shared task 2011. In *Proceedings of the BioNLP Shared Task Workshop*.

Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. 2014. Semi-supervised learning with deep generative models. In *Proceedings of the Neural Information Processing Systems Conference*.

Makoto Miwa, Rune Sætre, Yusuke Miyao, and Jun'ichi Tsujii. 2009. Protein–protein interaction extraction by leveraging multiple kernels and parsers. *International Journal of Medical Informatics*.

Raymond J Mooney and Razvan C Bunescu. 2005. Subsequence kernels for relation extraction. In *Proceedings of the Neural Information Processing Systems Conference*.

Claire Nédellec, Robert Bossy, Jin-Dong Kim, Jung-Jae Kim, Tomoko Ohta, Sampo Pyysalo, and Pierre Zweigenbaum. 2013. Overview of BioNLP shared task 2013. In *Proceedings of the BioNLP Shared Task Workshop*.

Thien Huu Nguyen and Ralph Grishman. 2015. Relation extraction: Perspective from convolutional neural networks. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*.

Yifan Peng and Zhiyong Lu. 2017. Deep learning for extracting protein-protein interactions from biomedical literature. In *Proceedings of BioNLP Workshop*.

Michael Pust, Ulf Hermjakob, Kevin Knight, Daniel Marcu, and Jonathan May. 2015. Parsing english into abstract meaning representation using syntax-based machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Sudha Rao, Daniel Marcu, Kevin Knight, and Hal Daumé III. 2017. Biomedical event extraction using abstract meaning representation. In *Workshop on Biomedical Natural Language Processing*.

Anthony Rios, Ramakanth Kavuluru, and Zhiyong Lu. 2018. Generalizing biomedical relation classification with neural adversarial domain adaptation. *Bioinformatics*.

Andrey Rzhetsky. 2016. The big mechanism program: Changing how science is done. In *Proceedings of DAMDID/RCDL*.

Domonkos Tikk, Philippe Thomas, Peter Palaga, Jörg Hakenberg, and Ulf Leser. 2010. A comprehensive benchmark of kernel methods to extract protein–protein interactions from literature. *PLoS Computational Biology*.

Yijia Zhang and Zhiyong Lu. 2019. Exploring semi-supervised variational autoencoders for biomedical relation extraction. *Methods*.

# Auditing Deep Learning processes through Kernel-based Explanatory Models

**Danilo Croce** and **Daniele Rossini** and **Roberto Basili**
Department of Enterprise Engineering
University of Roma, Tor Vergata
{croce,basili}@info.uniroma2.it
rossini.danie@gmail.com

## Abstract

While NLP systems become more pervasive, their accountability gains value as a focal point of effort. Epistemological opaqueness of non-linear learning methods, such as deep learning models, can be a major drawback for their adoptions. In this paper, we discuss the application of *Layerwise Relevance Propagation* over a linguistically motivated neural architecture, the *Kernel-based Deep Architecture*, in order to trace back connections between linguistic properties of input instances and system decisions. Such connections then guide the construction of argumentations on the network's inferences, i.e., explanations based on real examples that are semantically related to the input. We also propose here a methodology to evaluate the transparency and coherence of analogy-based explanations modeling an $audit$ stage for the system. Quantitative analysis on two semantic tasks, i.e., question classification and semantic role labeling, shows that the explanatory capabilities (native in KDAs) are effective and they pave the way to more complex argumentation methods.

## 1 Introduction

AI systems are currently used in a wide variety of applications, with several levels of societal impact, and are expected to be soon deployed in safety-critical fields, e.g., autonomous driving. The definition of codes of conduct in the development of AI applications to ensure their ethical sustainability across dimensions, such as fairness, reliability and beneficialness (Kroll et al., 2016; Garfinkel et al., 2017; Dignum, 2017) is becoming a crucial issue. Hence, a natural need for the ethical accountability of such systems is gaining importance.

A central issue lies in designing systems whose decisions are *transparent* (Ribeiro et al., 2016; Doshi-Velez et al., 2017), i.e., they must be easily interpretable by humans, as users must be able to suitably weight and trust the assistance of such systems.

Deep neural networks are clearly problematic in this regard: their high non-linearity, despite allowing for state-of-the-art performances in several challenging problems, amplifies the epistemological opaqueness of the decision-flow and limits its interpretability. The concept of transparency of a machine learning model spans multiple definitions, focusing on different aspects, from the simplicity of the model, e.g., the number of nodes in a decision tree, to the intuitiveness of its parameters and computations (Chakraborty et al., 2017).

In this context, an important capability of an AI system is the ability to provide *post-hoc explanations* in terms of evidences supporting the produced decisions: although they usually do not formally elucidate how a model works, they often have the property of being quite intuitive, conveying useful information also to end-users without any AI or machine learning expertise (Lipton, 2018). In semantic inference tasks (e.g., text classification), an *explanation model* generating post-hoc explanations should hence be able to trace back connections between the output categories and the semantic and syntactic properties of the input texts. Such models should have three desired properties: *semantic transparency*, *informativeness* w.r.t. the system decision and *effectiveness* in enabling auditing processes against the system.

In this work we focus on a specific post-hoc mechanism, which is to provide, along with the prediction, a comparison with one or more other examples, namely *landmarks*, that share task-relevant linguistic properties with the input. From an argument theory perspective, this corresponds to supporting decisions through an "argument by analogy" schema (Walton et al., 2008): a user ex-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4037–4046,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

posed to such a kind of argument will endow a different level of trust into the machine decision according to the linguistic plausibility of the analogy. He will implicitly gauge the evidence from the linguistic properties shared between the input sentence (or its parts) and the one(s) used for comparison as well their importance with respect to the output decision. Let us consider, for example, the following prediction in a question classification (QC) task (Li and Roth, 2006): *"What is the capital of Zimbabwe?" refers to a* `Location`. We would like the system to motivate its decision with an argument such as: *...since it recalls me of "What is the capital of California?" which also refers to a* `Location`. Notice that explanation of a decision is quite different from sentence or document ranking in Information Retrieval so that semantic similarity plays only a minor role: clear and trustful analogies may exist with semantically different training examples that imply similar relationships between the input and the decision.

Recent work has been inspired by efforts in improving model's interpretability in image processing tasks, in particular by the *Layerwise Relevance Propagation* (LRP) (Bach et al., 2015). In LRP the classification decision of a deep neural network is decomposed backward across the network layers and evidence about the contribution to the final decision brought by individual input fragments (i.e., pixels of the input image) is gathered.

In this paper, we propose to extend the LRP application to the linguistically motivated network architectures, known as Kernel-Based Deep Architectures (KDAs) (Croce et al., 2017), which frames semantic information captured by linguistic Tree Kernel methods (Collins and Duffy, 2001) within the neural-based learning paradigm. The result is a mechanism that, for each system's prediction such as in question classification, generates an argument-by-analogy explanation based on real training examples, not necessarily similar to the input.

We also propose a novel approach to evaluate numerically the interpretability of any explanation-enriched model applied in semantic inference tasks. By defining a specific audit process, we derive a synthetic metric, i.e. *Auditing Accuracy*, that takes into account the properties of transparency, informativeness and effectiveness. The evaluation of the proposed methodology shows the meaningful impact of LRP-based explanation models: users faced with explanations are systematically oriented to accept (or reject) the system decisions, so that *post hoc* judgments may even improve the overall application accuracy.

In the rest of the paper, section 2 reports related works, while Section 3 describes the LRP and its extension to KDAs. In Section 4, we propose three explanation models and illustrate a novel evaluation methodology, commenting on the audit process and deriving quantitative notions such as the *Auditing Accuracy* measure. Section 5 presents and discusses the system effectiveness against two semantic tasks, i.e., question classification and frame-based argument classification in a semantic role labeling chain. Finally, in Section 6 conclusions are derived.

## 2 Related Work

In recent years, research communities showed great interest in improving neural models' interpretability, as testified by the effort of defining the concept of interpretability itself and the development of a variety of approaches to the problem. In (Chakraborty et al., 2017) and (Lipton, 2018), the authors examine the different notions of interpretability found in literature and categorize techniques according to the transparency properties they confer to decision models. Common approaches to improve the *readability* of a neural model in image-related tasks are based on backward algorithms that reuse arc weights to propagate the prediction down to the input (Erhan et al., 2010; Zeiler and Fergus, 2013), thus leading to the re-creation of *meaningful* patterns in the input space. Typical examples are deconvolution heatmaps, used to approximate through Taylor series the partial derivatives at each layer (Simonyan et al., 2013), or the so-called Layer-wise Relevance Propagation (LRP), that redistributes back positive and negative evidence across the layers (Bach et al., 2015).

Local explanation approaches focus on highlighting a handful of crucial features (Baehrens et al., 2010) or deriving simpler, more readable models from a complex one, e.g., a binary decision tree (Frosst and Hinton, 2017), or by local approximation with linear models (Ribeiro et al., 2016). However, although they can explicitly show the representations learned in the specific hidden neurons (Frosst and Hinton, 2017), these approaches base their effectiveness on the user ability to study

the quality of the reasoning and of the accountability as a side effect of the quality and the coherence of the features selection: this can be very hard in tasks where boundaries between classes are not well defined. Another strategy is pairing the decision model with a generative model to produce verbose explanations (Krening et al., 2017). Sometimes explanations are associated to vector representations as in (Ribeiro et al., 2016), i.e., bag-of-words in case of text classification, that are clearly weak at capturing significant linguistic abstractions, such as the involved syntactic relations. In this work, we systematically extend the model presented in (Croce et al., 2018) which allows to provide explanations that are easily interpretable even by non-expert users, as they are expressed in natural language. Moreover, the investigated approach is is computationally affordable, as it roughly corresponds to a forward pass across the network. In addition, we also provide a systematic way to evaluate the provided explanations with a methodology able to support the audit of the targeted AI systems.

## 3 Layer-wise Relevance Propagation in Kernel-based Deep Architectures

In this section, we will review the Layer-wise Relevance Propagation technique (LRP, as in (Bach et al., 2015)), usually applied in image processing, and show how it can be naturally extended to Kernel-based Deep Architectures (KDA, as in (Croce et al., 2017)) in order to select real examples useful to support the network decisions.

LRP is mainly a framework which allows to decompose the prediction of a deep neural network computed over a sample, usually an image, down to relevance scores for the single input dimensions of the sample, such as sub-pixels of the image itself. More formally, let $f : \mathbb{R}^d \rightarrow \mathbb{R}^+$ be a function that quantifies, for example, the probability of $x \in \mathbb{R}^d$ being in a certain class. The Layer-wise Relevance Propagation assigns to each dimension, or feature, $x_d$ a relevance score $R_d^{(1)}$ such that $f(x) \approx \sum_d R_d^{(1)}$. Features whose score is $R_d^{(1)} > 0$ or $R_d^{(1)} < 0$ correspond to evidence in favor or against, respectively, the output classification. In other words, LRP allows to identify fragments of the input playing key roles in the decision, by propagating relevance backwards. Let us suppose to know the relevance score $R_j^{(l+1)}$ of a neuron $j$ at network layer $l+1$. This can be decomposed into messages $R_{i \leftarrow j}^{(l,l+1)}$ sent from $j$ to neurons $i$ in layer $l$ according to $R_j^{(l+1)} = \sum_{i \in (l)} R_{i \leftarrow j}^{(l,l+1)}$. Then ti directly follows that the relevance of a neuron $i$ at layer $l$, that is the quantity of information travelling through $i$, can be defined as $R_i^{(l)} = \sum_{j \in (l+1)} R_{i \leftarrow j}^{(l,l+1)}$. In this work, we adopted the $\epsilon$-rule defined in (Bach et al., 2015) to compute the messages $R_{i \leftarrow j}^{(l,l+1)} = \frac{z_{ij}}{z_j + \epsilon \cdot \text{sign}(z_j)} R_j^{(l+1)}$, where $z_{ij} = x_i w_{ij}$ and $\epsilon > 0$ is a small numerical stabilizing term. The informative value is justified by the fact that the weights $z_{ij}$ are linked to the activation weights $w_{ij}$ of the input neurons.

Given the capability of computing relevance scores for input dimensions, we now summarize the KDA to motivate how LRP can be applied also to tasks other than image classification. In a nutshell, the KDA is a neural network trained in low-dimensional spaces which approximate a generic Reproducing Kernel Hilbert Space (RKHS) (Shawe-Taylor and Cristianini, 2004). These low-dimensional approximations are derived as a reconstruction from a set of real reference training examples, called *landmarks*, which can be used to compile the representation of any unseen test instance. As a consequence, the ability of making connections between the KDA decisions and the landmarks corresponds to locating the candidate training examples that justify (in the LRP sense) decisions and trigger meaningful linguistic explanations.

More formally, given an input dataset $\mathcal{D}$, a kernel $K(o_i, o_j)$ is a function over $\mathcal{D}^2$ operating dot-products, i.e., similarity scores, in an projection space, given by the mapping $\Phi$ over the input instances $o_i$, which is implicit in the sense that the kernel never explicitly accesses the representation of projections $\Phi(o_i)$. Here a RKHS corresponds to the Gram Matrix $G = XX^\top$, whose element $G_{i,j}$ is $K(o_i, o_j) = \Phi(o_i) \cdot \Phi(o_j)$. The Nyström method (Williams and Seeger, 2001) can be applied to derive the approximating matrix $\tilde{G} = (CUS^{-\frac{1}{2}})(CUS^{-\frac{1}{2}})^\top \approx G$, where $U$, $S$ are obtained by applying the Singular Value Decomposition to the matrix $W \in \mathbb{R}^{l \times l}$, a submatrix of $G$ containing the kernel evaluations of $l$ sampled instances (namely, the *landmarks*) and $C \in \mathbb{R}^{|D| \times l}$, whose row $\vec{c_i}$ corresponds to the similarity scores between $o_i \in \mathcal{D}$ and the landmarks. Hence, a mapping from $\mathcal{D}$ to a $l$-dimensional embedding, with $l \ll n$, is naturally provided by the **projection**

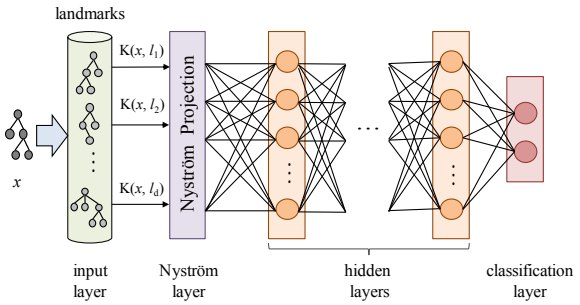


Figure 1: Kernel-based Deep Architecture.

**function** $\tilde{\vec{x}} = \vec{c}\,US^{-\frac{1}{2}}$. Therefore, the method produces $l$-dimensional vectors[1].

In (Croce et al., 2017), the Nyström representation $\tilde{\vec{x}}$ has been used to map semantically annotated grammatical trees to the linear input of a Multi-Layer Perceptron (MLP). In fact, given a dataset $\mathcal{L}$, with $o \in \mathcal{L}$ denoting a generic instance, the MLP architecture is defined with a specific Nyström input layer based on the Nyström embeddings. The resulting Kernel-based Deep Architecture (KDA) includes an *input layer*, the *Nyström layer*, a sequence of *hidden layers* and the final *classification layer*, which produces the output. The *input* layer corresponds to the input vector $\vec{c_i}$, i.e., the row of the $C$ matrix associated to an example $o_i$. The input layer is mapped to the *Nyström* layer, through the Nyström projection. Notice that the embedding provides also the proper weights, defined by $US^{-\frac{1}{2}}$, so that the mapping can be expressed through the Nyström matrix $H_{Ny} = US^{-\frac{1}{2}}$. The resulting $\tilde{\vec{x}}$ is the input to one or more *hidden* layers. Clearly, the first hidden layer receives in input $\tilde{\vec{x}} = \vec{c}H_{Ny}$. Finally, the *classification layer* computes a linear classification function with a softmax operator, as shown in Figure 1. A KDA optimizes the standard cross-entropy function with $L_2$ regularization.

It is worth recalling that the network is triggered by an input vector $\vec{c}$ expressing the kernel evaluations $K(x, l_i)$ between the example and the landmarks. When using linguistic kernels (such as Semantic Tree Kernels, (Croce et al., 2011)), this measure corresponds to the grammatical and lexical semantic similarity between $x$ and the subset of landmarks. The expected explanation is obtained from the network output by applying LRP to revert the propagation process, thus linking the output back to the input. In a KDA that models linguistic instances, LRP implicitly traces back the syntactic, semantic and lexical relations between the example and the landmarks across the Nyström layer: the side effect is to select those real examples that mostly influenced the identification of the predicted structure in the sentence.

[1]Note that, while any sampling policy for landmarks is admissible, in (Kumar et al., 2012) it is demonstrated that uniform sampling without replacement achieves results comparable with alternative, more resource-consuming policies.

## 4 Generating explanations in Kernel-based Deep Architectures

Justifications for the KDA emissions can be obtained by exploiting landmarks $\{\ell\}$ as the evidence in favour or against a class. The idea is to select those $\{\ell\}$ that the LRP highlights as the most active elements in layer 0. Once such active landmarks are detected, an *Explanatory Model* is a function in charge to compile a linguistically fluent explanation by comparing the input case with such selection.

The semantic expressiveness of such analogies makes the resulting explanation clear and increases the user confidence on the system reliability. When a sentence $s$ is classified, LRP assigns activation scores $r^s_\ell$ to each individual landmark $\ell$: let $\mathcal{L}^{(+)}$ (or $\mathcal{L}^{(-)}$) denote the set of landmarks with positive (or negative) activation scores. Formally, each explanation is characterized by a triple $e = \langle s, C, \tau \rangle$ where $s$ is the input sentence, $C$ is a target label and $\tau$ is the modality of the explanation: $\tau = +1$ for positive (i.e., acceptance) statements while $\tau = -1$ correspond to rejections of $C$. A landmark $\ell$ is *positively activated* for a given sentence $s$ if there are at most $k-1$ other active landmarks $\ell'$ with activation value higher than the one for $\ell$, i.e.,

$$|\{\ell' \in \mathcal{L}^{(+)} : \ell' \neq \ell \wedge r^s_{\ell'} \geq r^s_\ell > 0\}| < k$$

Similarly, a landmark is *negatively activated* when:

$$|\{\ell' \in \mathcal{L}^{(-)} : \ell' \neq \ell \wedge r^s_{\ell'} \leq r^s_\ell < 0\}| < k$$

where $k$ is a fixed parameter used to make the explanation depending on not more than $k$ landmarks, denoted as a set by $\mathcal{L}_k$. Positively (or negative) active landmarks in $\mathcal{L}_k$ are assigned an activation value $a(\ell, s) = +1$ $(-1)$, while $a(\ell, s) = 0$ for not active landmarks. Given the explanation $e = \langle s, C, \tau \rangle$, a landmark $\ell$, whose known class is $C_\ell$, is called *consistent* (or *inconsistent*) with $e$ if the function

$\delta(C_\ell, C) \cdot a(\ell, q) \cdot \tau$ is positive (or negative, respectively), where $\delta(C', C) = 2\delta_{kron}(C', C) - 1$ and $\delta_{kron}$ is the Kronecker delta.

We can thus partition such landmarks into the set of positively consistent landmarks $\mathcal{L}_k^{c,+} \subseteq \mathcal{L}_k^c \subseteq \mathcal{L}_k$ and negatively consistent ones $\mathcal{L}_k^{c,-} \subseteq \mathcal{L}_k^c \subseteq \mathcal{L}_k$, with $\mathcal{L}_k^{c,+} \cup \mathcal{L}_k^{c,-} = \mathcal{L}_k^c$, that aggregates all the consistent landmarks.

An *explanatory model* is then a function $M(e, \mathcal{L}_k^c)$ which maps an explanation $e$ and the set $\mathcal{L}_k^c$ for $e$ into a sentence $f$ in natural language. Of course several definitions of $M(e, \mathcal{L}_k^c)$ are possible, e.g.,

$$M(e, \mathcal{L}_k^c) = \begin{cases} \text{“}s \text{ is } C \text{ since it recalls me of } \ell\text{”} \\ \forall \ell \in \mathcal{L}_k^{c,+} \quad \text{if } \tau > 0 \\ \\ \text{“}s \text{ is not } C \text{ since it does not} \\ \text{recall me of } \ell \text{ which is } C\text{”} \\ \forall \ell \in \mathcal{L}_k^{c,-} \quad \text{if } \tau < 0 \\ \\ \text{“}s \text{ is } C \text{ but I don't know why”} \\ \text{if } \mathcal{L}^c \equiv \emptyset \end{cases}$$

Here we introduce three explanatory models used during experimental evaluation:

- **Singleton Model.** The first model is the simplest as it returns a single analogy with the consistent landmark with the highest positive score, if $\tau = 1$, or lowest negative score, when $\tau = -1$. As an example, the explanation of an accepted decision in a semantic argument classification task, described by the triple $e_1 = \langle$'Put *this plate* in the center of the table', $\text{THEME}_{\text{PLACING}}, 1\rangle$, would be mapped by the model into: *I think "this plate" is* THEME *of* PLACING *in "Robot* PUT *this plate in the center of the table" since it recalls me of "the soap" in "Can you* PUT *the soap in the washing machine?"*.

- **Conjunctive Model.** In a second model, denoted as *Conjunctive*, the system makes reference to up to $k_1 \leq k$ analogies with positively (or negatively) active and consistent landmarks. Given the above explanation $e_1$, and $k_1 = 2$, it would return: *I think "this plate" is* THEME *of* PLACING *in "Robot* PUT *this plate in the center of the table" since it recalls me of "the soap" in "Can you* PUT *"the soap" in the washing machine?" and also of "my coat" in "*HANG *my coat in the closet in the bedroom"*.

- **Contrastive Model.** The last proposed model is more complex since it returns both a positive and a negative analogy by selecting, respectively, the most positively relevant and the most negatively relevant consistent landmark. Given $e1$, it would return: *I think "this plate" is* THEME *of* PLACING *in "Robot* PUT *this plate in the center of the table" since it recalls me of "the soap" in "Can you* PUT *the soap in the washing machine" and it is not the* GOAL *of* PLACING *since it does not recall me of "on the counter" in "*PUT *the plate on the counter"*.

In case no active and consistent landmark can be found, the models return a phrase stating only the predicted class, with no explanation.

## 5 Experimental Evaluation

Evaluating the explanatory quality of an inductive model is still an open problem and universally recognized gold standards are not available for comparative analysis. In order to rely on a quantitative analysis, we assume that an explanation to be effective should assist a human user to ascertain whether the proposed classification is correct or not. Plausible and coherent explanations should thus be generated from correct system's decisions, while bad decisions should correspond to ambiguous or plainly fallacious arguments.

Hence, the evaluation of an explanatory model should reflect the model's adherence to three desired properties: **semantic transparency**, i.e., argument's linguistic grounding should be clear and straightforward, requiring as less knowledge on the system's functioning and on the specific task as possible; **informativeness** with respect to the system's decision, i.e., the explanation's generating process should be highly dependent on how the system processes input information; **effectiveness w.r.t an audit against the system**, i.e., the explanation should convey enough meaningful information so that a human can correctly decide whether to trust the system prediction or not.

Consequently, we define a auditing task in which annotators are required to judge if a proposed explanation would commit them to trust the system decision. This judgment is discretized

within five possible labels: *Very Good* if the analogy is strongly convincing and linguistically clear; *Good* if the explanation is still accepted but the pertinence is slacker; *Uncertain* if the annotator gains no meaningful information from the explanation or no explanation is provided at all; *Bad* if some connections can be detected between the input sentence and the one used as a comparison but they are so ambiguous that the explanation is rejected; *Incoherent* if the argument appears totally inconsistent and meaningless. Given the nature of the argument by analogy schema (Walton et al., 2008), it follows that annotators assigning a *Very Good* or *Good* label to an explanation are also implicitly accepting the system decision as correct, whereas they are rejecting it as wrong in the other cases.

Given an explanation dataset $E = \{(e, c, x_C)\}$ where $e$ is an explanation, $c \in \{1, -1\}$ expresses if the explanation was generated from a correct ($c = 1$) or incorrect ($c = -1$) classification, and $x_C$ is the numerical value corresponding to one of the five labels categories $C$ above[2], we can define the set $A_c$ of accepted explanations generated from correct predictions and the set $R_{nc}$ of rejected explanations generated from not correct predictions as follows:

$$A_c = \{(e, c, l) \in E | c = 1 \wedge x_C > 0\}$$

$$R_{nc} = \{(e, c, l) \in E | c = -1 \wedge x_C \leq 0\}$$

Accordingly, the *Audit Accuracy (AuAcc)*

$$AuAcc = \frac{|A_c| + |R_{nc}|}{|E|}$$

measures the ratio between correct acceptance/rejection decisions and the total number of decisions made by the human auditor.

Additionally, the Pearson Correlation between the system classification accuracy and the human judgment of an explanation can be interpreted as a concrete measure of the consistency of an explanatory model: an ideal model should map correct classifications to convincing explanations and incorrect classification to implausible explanations. It will be thus exploited to compare alternative explanatory models. To test an explanation approach as well as of the proposed evaluation metrics, we will address two different semantic processing tasks, i.e., question classification (QC) and argument classification (AC) in semantic role labeling.

**Experimental Setup.** The Nyström projection has been implemented in the KeLP framework[3], while the LRP-integrated KDA in Tensorflow, with 1 and 2 hidden layers, respectively, whose layer-size is equal to the number of randomly selected Nyström landmarks (500 and 200, in QC and AC respectively). For both tasks, training have been executed in 500 epochs, using the Adam optimizer and adopting early-stop and dropout strategy while selecting the best model according to performances over the development set. We conducted preliminary evaluations on small samples of the dataset and set the parameter $k = 5$, which defines the cardinality of the active landmarks $\mathcal{L}_k$[4]. The remaining hyper-parameters were tuned via grid-search.

A group of human annotators was asked to rate each explanation with one out of the five labels described early in this section, basing their judgment only on the perceived level of trust w.r.t. the explanations. Each annotator was exposed to explanations derived from a perfectly balanced set of correct and incorrect classifications, so that annotators are not biased by the (possibly high) quality of the classifier when judging the explanations.

### 5.1 Question Classification

We replicated the experiments reported in (Croce et al., 2017) with respect to the question classification task, using the UIUC dataset (Li and Roth, 2006), including a training and test set of 5452 and 500 questions, respectively, organized in 6 coarse-grained classes (as `ENTITY` or `HUMAN`). We generated Nyström representation of the Compositionally Smoothed Partial Tree Kernel function (Annesi et al., 2014) consistently with (Croce et al., 2017). Using 500 landmarks, the KDA accuracy was 93.6%, which is comparable with state-of-the-art neural models, as discussed in (Croce et al., 2017). The audit manual task was independently performed by 3 annotators[5]: each annotator

[2] We assigned values $x_C \in [1, -1]$: $x_{Very\ Good} = 1$, $x_{Good} = 0.8$, $x_{Uncertain} = 0$, $x_{Bad} = -0.8$, $x_{Incoherent} = -1$

[3] http://www.kelp-ml.org, presented in (Filice et al., 2018).

[4] No particular differences have emerged in the generation of explanations when slightly different values of $k$ where considered.

[5] Annotators have different levels of fluency in English (even though no one is a native speaker) and different levels of expertise: one has no specific technical knowledge, one is a graduate student with advanced knowledge in NLP, while

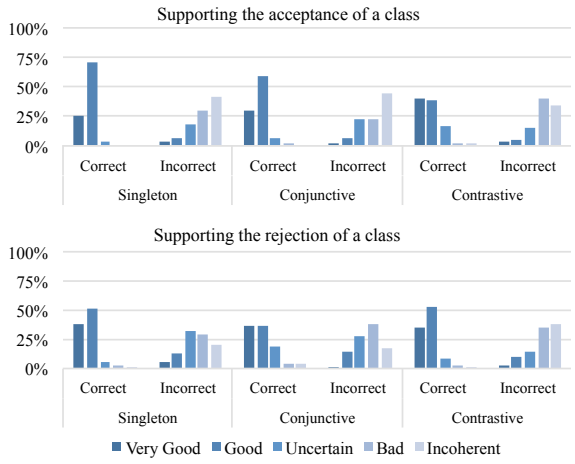


Figure 2: Results of the audit process in the QC task for the three explanatory models separating system acceptance (top) from rejection (bottom). Individual columns represent the percentage of quality label across explanations. Left columns describe correct classifications, while right ones are derived from incorrect classifications.

evaluated 300 explanations (100 for each model), reaching an inter-annotation agreement if 0.82 on these data.

Results in Figure 2 suggest that the annotators were able to properly discriminate correct from incorrect decisions, just through the exposure to the explanations: in both acceptance or rejection cases, all models tend to assign positive labels (Very Good and Good) to explanations of correct decisions and negative ones (Uncertain, Bad and Incoherent) to explanations of incorrect decisions instead. Note that an explanation rejecting a class should be labeled as positive, if the landmark used for *negative* analogy is actually *not* recalling the input sentence. The graphical intuition in Figure 2 is confirmed by the metrics: the Singleton, Conjunctive and Contrastive models reach an *Audit Accuracy* of 89.3%, 84.7% and 86.3%, respectively. The *Pearson Correlation* between acceptance and correctness is 78.9%, 69.4% and 72.8%, while if we measure the correlation between the explanation quality score and the decision correctness, the Pearson coefficients become 76.1%, 71.2% and 77.2%: these are slightly lower basically for the lower reward assigned to $x_{Good}$ w.r.t. $x_{Very\ Good}$. Small numerical differences among models emerge: it seems that the Conjunctive and Contrastive models are not always able to retrieve meaningful additional information, while the Singleton model is simpler and more direct. An example of output analogies is given by *I think "How many Admirals are there in the U.S. Navy?" refers to a* `NUMBER` *since it recalls me of both "How many words are there in the Spanish language?" and "How many sides does an obelisk have?"*, generated by the Conjunctive model. Here the semantic hint corresponds to the discriminative fragment *"How many"*. However meaningful connections between the input and landmarks are also traced against poor overlaps in syntactic and lexical information as in: *I think "Where is the Mall of the America?" refers to a* `LOCATION` *since it recalls me of "What town was the setting for The Music Man?"*.

Table 1 reports question-explanation pairs with similarity estimates based on the adopted CSPTK kernel function. It is clear from the examples that similarity alone is not able to correlate with classification decisions: questions in different classes (e.g. first two rows in the table) may have very high similarity scores. Second, landmarks correlate with decisions in interesting ways that do not depend on strict lexical and grammatical similarity. Conceptually more grounded associations seem to emerge: e.g., explaining *"What was J.F.K.'s wife's name"* by the analogy with *"What was Darth Vader's son named?"* is abstracted across a conceptual relation (e.g. `has_name`) and the derived analogy is quite clear. Notice that active landmarks are independent from similar questions, as landmarks triggered by similar questions are not similar to each other.

Interestingly, explanations of ambiguous instances are harmonic with human uncertainty. The explanation *I think "What is the sales tax in Minnesota?" refers to a* `NUMBER` *since it recalls me of "What is the population of Mozambique?" and does not refer to a* `ENTITY` *since it does not recall me of "What is a fear of slime?"* is convincing, but incorrect. Here, the lack of context impacts on the disambiguation of two plausible interpretations that are (1) the definition of the notion of "sales tax" (`ENTITY`), w.r.t (2) its current value (`NUMBER`): the gold standard suggests `ENTITY` as the correct category.

### 5.2 Argument Classification

Semantic role labeling (SRL (Palmer et al., 2010)) consists in detecting the semantic arguments associated with the predicate of a sentence and their

the last one is an expert in the field.

| Class | Questions ($q_i$) | $k(q_1, q_2)$ | Activated Landmarks ($l_i$) | $k(l_1, l_2)$ |
|---|---|---|---|---|
| LOC | *"What is the capital of Ethiopia?"* | 0.98 | | |
| NUM | *"What is the population of Nigeria?"* | | | |
| ENTY | *"What was FDR 's dog 's name?"* | 0.97 | *"What is the name of David Letterman's dog?"* | 0.49 |
| HUM | *"What was J.F.K.'s wife 's name?"* | | *"What was Darth Vader 's son named?"* | |
| ENTY | *"What is the Ohio state bird?"* | 0.90 | *"What is the name of David Letterman 's dog?"* | 0.61 |
| ENTY | *"What is the pH scale?"* | | *"What is viscosity?"* | |
| ENTY | *"What was the first satellite to go into space?"* | 0.83 | *"What was the first TV set to include a remote control?"* | 0.61 |
| HUM | *"Who was the first American to walk in space?* | | *"What 's the name of the actress who starred in the movie, Silence of the Lambs?"* | |
| NUM | *"What was the last year that the Chicago Cubs won the World Series?"* | 0.73 | *"The film Jaws was made in what year?"* | 0.31 |
| NUM | *"What is the average speed of the horses at the Kentucky Derby?"* | | *"What is average salary of restaurant manager in United States?"* | |

Table 1: Examples of semantically similar questions in the same or different classes, with the corresponding landmarks activated during the classification.

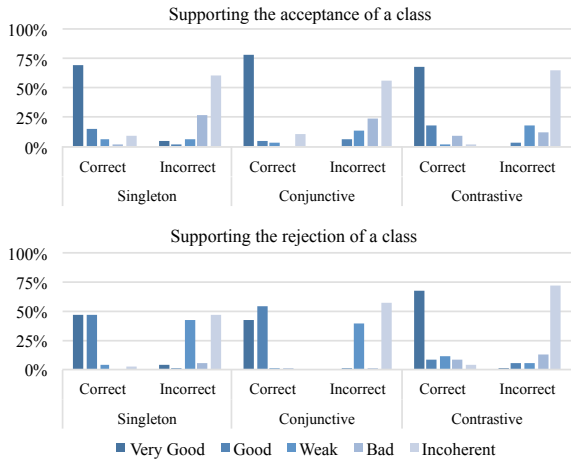


Figure 3: Results of the audit process in the SRL task for the three explanatory models: system acceptance (top), rejection (bottom).

classification into their specific roles ((Fillmore, 1985)). For example, given the sentence "*Bring the fruit onto the dining table*", the task would be to recognize the verb "*bring*" as evoking the BRINGING frame, with its roles, THEME for "*the fruit*" and GOAL for "*onto the dining table*". Argument classification corresponds to the subtask of assigning labels to the sentence fragments spanning individual roles.

As proposed in (Moschitti et al., 2008), SRL can be modeled as a multi-classification task over each parse tree node $n$, where argument spans reflect sub-sentences covered by the tree rooted at $n$. Consistently with (Croce et al., 2011), in our experiments the KDA has been empowered with a Smoothed Partial Tree Kernel, operating over Grammatical Relation Centered Trees (GRCT) derived from dependency grammar. The reference benchmark, i.e., the HuRIC dataset related to an Interactive Robotics task (Bastianelli et al., 2016), includes about 650 annotated transcriptions of spoken robotic commands, organized in 18 frames and about 60 arguments. Individual arguments extracted amount to 1,300 examples. Experimental setup was similar to that of Section 5.1, but due to the limited data size we applied 10-fold cross-validation, optimizing network hyper-parameters via grid-search for each fold. We generated the Nyström representations of a SPTK function with default parameters $\mu = \lambda = 0.4$ as in (Croce et al., 2011). With these settings, the KDA accuracy was 96.1%. Due to the slightly higher complexity of the task, w.r.t. QC, in the case the two independent auditors had at least graduate-level knowledge in NLP. They were requested to judge about 700 generated explanations with an inter-annotation agreement of 0.89. For the Singleton, Conjunctive and Contrastive model, respectively, the *Audit Accuracy* is 91.6%, 93.4%, 88.4% while Pearson Coefficients between acceptance/rejection and correctness are 83.3% (80.1% for quality-correctness correlation), 86.9% (81.9%), 77.3% (78.7%): this suggests an higher annotator's sensitivity to the explanations' plausibility, as reflected also by the charts in Figure 3, probably due to the task itself being more challenging for humans.

As in QC, the system can convey semantically transparent and useful information without relying on lexical similarity alone; e.g., consider *I think "is hot" is* DESIRED STATE *of* INSPECTING *in "Robot* CHECK *whether the oven is hot?" since it recalls me of "is empty" in "*SEE *if the washing machine is empty"*. In this task, the Contrastive model could also to produce explanations exemplifying differences between separate roles in the same frame, for example: *I think "to me" is not*

GOAL *of* BRINGING *in "Can you go to the kitchen find a glass and* BRING *it to me?" since it does not recall me of "to the bedroom" in "*BRING *the phone to the bedroom" and it's the* BENEFICIARY *of* BRINGING *since it recalls me of "to me" in "can you please search the book and* BRING *it to me".*

## 6 Conclusions

This paper proposes a quantitative evaluation of the automatic generation of epistemologically transparent and linguistically fluid explanations for neural inferences. The proposed approach applies LRP to a Kernel-based Deep Architecture (KDA) that redistributes the prediction value to training entries (i.e., annotated landmarks). The resulting sentence exploits analogies with training instances, according to different explanatory strategies. Given that KDAs (based on Nyström embeddings) can be flexibly adopted in neural learning for NLP, we show how the auditing mechanism outlined in the paper is epistemologically very effective and emphasizes the neural embeddings with a strong impact on explainability. First, language semantics is promoted *by design* and associations generated between input instances and decisions are obtained without ever leaving the language level. Second, different and mathematically solid models for different levels of language semantics can be obtained by modifying the adopted kernel formulations. In this way, a unique general auditing mechanism is able to support fine tuning towards very different tasks, without changes in the learning architecture. Finally, as Table 1 shows, explanations are strictly dependent on the induced neural model and are not just triggered by text similarity metrics: they are epistemologically principled evidences about the neural learning stages, based on the observed examples and the selected landmarks.

Empirical investigations carried out against the QC and AC tasks also confirm that the good explanatory models strongly correlate with consistent decisions and effectively contribute to increase the user confidence in the neural inference consistency. This make an auditing activity for human users viable. On one side, it allows to limit the impact of machine mistakes, in a natural and portable manner. Moreover, it can also serve as a novel comparative evaluation paradigm. The reachable auditing accuracy thus measures the explanatory power of different models and can be employed as a comparative benchmark. While the methods proposed in this paper stem just from early explorations, the ways activated landmarks can be made useful to meaningful explanations stimulate further research, involving feature based analysis such as suggested in (Ribeiro et al., 2016) or the application of LRP to architectures more complex than a MLP. Argumentation theory, applied to the active landmark semantics and the source input example as captured by the kernel, provides a very rich framework to design future and more complex justification mechanisms.

## References


Paolo Annesi, Danilo Croce, and Roberto Basili. 2014. Semantic compositionality in tree kernels. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014*, pages 1029–1038.

Sebastian Bach, Alexander Binder, Gregoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLOS ONE*, 10(7).

David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert Müller. 2010. How to explain individual classification decisions. *J. Mach. Learn. Res.*, 11:1803–1831.

Emanuele Bastianelli, Danilo Croce, Andrea Vanzo, Roberto Basili, and Daniele Nardi. 2016. A discriminative approach to grounded spoken language understanding in interactive robotics. In *Proceedings of IJCAI 2016, New York, NY, USA*, pages 2747–2753.

Supriyo Chakraborty, Richard Tomsett, Ramya Raghavendra, Daniel Harborne, and Moustafa Alzantot et al. 2017. Interpretability of deep learning models: A survey of results. *2017 SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI*, pages 1–6.

Michael Collins and Nigel Duffy. 2001. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of ACL '02, July 7-12, 2002, Philadelphia, PA, USA*, pages 263–270.

Danilo Croce, Simone Filice, Giuseppe Castellucci, and Roberto Basili. 2017. Deep learning in semantic kernel spaces. In *Proceedings of ACL 2017*, pages 345–354, Vancouver, Canada.

Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2011. Structured lexical similarity via convolution kernels on dependency trees. In *Proceedings of EMNLP '11*, pages 1034–1046.

Danilo Croce, Daniele Rossini, and Roberto Basili. 2018. Explaining non-linear classifier decisions within kernel-based deep architectures. In *Proceedings of the Workshop: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@EMNLP 2018, Brussels, Belgium, November 1, 2018*, pages 16–24.

Virginia Dignum. 2017. Responsible autonomy. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 4698–4704.

Finale Doshi-Velez, Mason Kortz, Ryan Budish, Chris Bavitz, Sam Gershman, David O'Brien, Stuart Schieber, James Waldo, David Weinberger, and Alexandra Wood. 2017. Accountability of ai under the law: The role of explanation. *CoRR*, abs/1711.01134.

Dumitru Erhan, Aaron Courville, and Yoshua Bengio. 2010. Understanding representations learned in deep architectures. Technical Report 1355, Université de Montréal/DIRO.

Simone Filice, Giuseppe Castellucci, Giovanni Da San Martino, Alessandro Moschitti, Danilo Croce, and Roberto Basili. 2018. Kelp: a kernel-based learning platform. *Journal of Machine Learning Research*, 18(191):1–5.

Charles J. Fillmore. 1985. Frames and the semantics of understanding. *Quaderni di Semantica*, 6(2):222–254.

Nicholas Frosst and Geoffrey Hinton. 2017. Distilling a neural network into a soft decision. *CEUR Workshop Proceedings*, 2071.

Simson Garfinkel, Jeanna Matthews, Stuart S. Shapiro, and Jonathan M. Smith. 2017. Toward algorithmic transparency and accountability. *Commun. ACM*, 60(9):5–5.

S. Krening, B. Harrison, K. M. Feigh, C. L. Isbell, M. Riedl, and A. Thomaz. 2017. Learning from explanations using sentiment and advice in rl. *IEEE Transactions on Cognitive and Developmental Systems*, 9(1):44–55.

Kroll, Huey, BAROCAS, W. Isaac Edward, and R. REIDENBERG. 2016. Accountable algorithms. *University of Pennsylvania Law Review*, 16.

Sanjiv Kumar, Mehryar Mohri, and Ameet Talwalkar. 2012. Sampling methods for the nyström method. *J. Mach. Learn. Res.*, 13:981–1006.

Xin Li and Dan Roth. 2006. Learning question classifiers: the role of semantic information. *Natural Language Engineering*, 12(3):229–249.

Zachary C. Lipton. 2018. The mythos of model interpretability. *Queue*, 16(3):30:31–30:57.

Alessandro Moschitti, Daniele Pighin, and Robert Basili. 2008. Tree kernels for semantic role labeling. *Computational Linguistics*, 34.

M.S. Palmer, D. Gildea, and N. Xue. 2010. *Semantic Role Labeling*. Online access: IEEE (Institute of Electrical and Electronics Engineers) IEEE Morgan & Claypool Synthesis eBooks Library. Morgan & Claypool Publishers.

Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "why should I trust you?": Explaining the predictions of any classifier. *CoRR*, abs/1602.04938.

John Shawe-Taylor and Nello Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press, Cambridge, UK.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034.

Douglas Walton, Christopher Reed, and Fabrizio Macagno. 2008. *Argumentation Schemes*. Cambridge University Press.

Christopher K. I. Williams and Matthias Seeger. 2001. Using the nyström method to speed up kernel machines. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 682–688. MIT Press.

Matthew D. Zeiler and Rob Fergus. 2013. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901.

# Enhancing Variational Autoencoders with Mutual Information Neural Estimation for Text Generation

**Dong Qian, William K. Cheung**
Department of Computer Science
Hong Kong Baptist University
{dongqian,william}@comp.hkbu.edu.hk

## Abstract

While broadly applicable to many natural language processing (NLP) tasks, variational autoencoders (VAEs) are hard to train due to the posterior collapse issue where the latent variable fails to encode the input data effectively. Various approaches have been proposed to alleviate this problem to improve the capability of the VAE. In this paper, we propose to introduce a mutual information (MI) term between the input and its latent variable to regularize the objective of the VAE. Since estimating the MI in the high-dimensional space is intractable, we employ neural networks for the estimation of the MI and provide a training algorithm based on the convex duality approach. Our experimental results on three benchmark datasets demonstrate that the proposed model, compared to the state-of-the-art baselines, exhibits less posterior collapse and has comparable or better performance in language modeling and text generation. We also qualitatively evaluate the inferred latent space and show that the proposed model can generate more reasonable and diverse sentences via linear interpolation in the latent space.

## 1 Introduction

Deep learning architectures are parameterized by families of non-linear functions, which learn multiple levels of more abstract representations (Bengio, 2009; Bengio et al., 2013). Recently, there has been a surge of interest in deep generative models for unsupervised learning, such as variational autoencoders (VAEs) (Kingma and Welling, 2014; Rezende et al., 2014), generative adversarial networks (GANs) (Goodfellow et al., 2014), and autoregressive models (van den Oord et al., 2016). The goal is to learn a compact representation to capture the salient structure in a given highly complex high-dimensional unlabelled data so that new data with some variations can be generated. They have been widely applied to a range of NLP tasks, such as language modeling (Bowman et al., 2016; Zhao et al., 2018a), dialog generation (Zhao et al., 2017, 2018b), etc. In this paper, we focus on the VAE with recurrent neural networks as its encoder and decoder for text generation.

Recurrent neural networks (RNNs) are one of the state-of-the-art autoregressive models for text modeling (Melis et al., 2018; Trinh et al., 2018). Training RNNs involves factorizing the joint distribution over a set of random variables into a series of conditional distributions. Yet, the one-step-ahead predictions force RNNs to learn local correlations, rather than global coherence. This is insufficient to capture high-level abstractions which characterize text sequences.

One approach to circumvent this challenge is to introduce variational autoencoders (VAEs) (Bowman et al., 2016), where an encoder learns a latent variable from text sequences and a decoder takes advantage of the variable to reconstruct word-level details. The inferred latent variable in the continuous space captures the semantics and syntactics of text sequences and text generation can be performed in the latent space. Although the approach is theoretically elegant, training the VAE often suffers from the well-known *posterior collapse* issue. It tends to ignore the latent variable and the resulting model reduces to a language model. The issue is especially challenging when a powerful autoregressive decoder (e.g., RNNs) is adopted (Bowman et al., 2016). Many recent efforts have been made to address this problem by modifying the objective (Zhao et al., 2019; Dieng et al., 2019; Wang and Wang, 2019), the decoder architecture (Yang et al., 2017), and the variational inference procedure (He et al., 2019; Fu et al., 2019).

The posterior collapse issue can also be understood from the fact that word-level details tend to carry more entropy bits than semantically-relevant concepts. It is well known that most of the changes in the word level to ensure grammatical coherence



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4047–4057,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

do not quickly change the underlying semantics. The maximum-likelihood objective prefers to capture entropy bits at the word level, rather than how well high-level semantics are encoded in the latent space. Training the VAE thus often ends up with relying solely on the autoregressive properties of text sequences, leaving the latent variable unused. Some previous studies (Chen et al., 2017; Alemi et al., 2018) have showed that optimizing the original objective of the VAE is deviated from the goal of learning a good representation.

We consider to address the posterior collapse issue by regularizing the latent space so that the inferred posterior of the latent variable is more specific to its own input, instead of only sticking to the prior. This can be done by maximizing the mutual information (MI) between the input and its latent variable. The MI not only measures how accurate the output agrees with its input, but also whether a meaningful latent variable can be learned in the latent space. However, the estimation and maximization of the MI in the high-dimensional space are difficult. A recent line of work (van den Oord et al., 2018; Belghazi et al., 2018; Hjelm et al., 2019; Veličković et al., 2019) tries to estimate and maximize the MI using the GAN-based approach, where the MI is defined as the Jensen-Shannon divergence (JSD) between the joint distribution and marginals. In this paper, we propose to add a mutual information regularization term, defined as the Kullback-Leibler (KL) divergence, into the objective of the VAE to alleviate the posterior collapse. Different from the models proposed in (Zhao et al., 2019; Dieng et al., 2019), we make effective use of the expressive power of neural networks for the tractable estimation and maximization of the MI in the high-dimensional space. Our empirical results demonstrate that the proposed model can achieve better modeling quality.

Our contributions are summarized as follows:

- We introduce a mutual information term into the objective of the VAE. The latent space can be learned by promoting the posteriors to be more specific to its own inputs, thus leading to low-redundant and diverse representations.
- We establish a lower bound on the mutual information via introducing an energy function paramterized by neural networks. An effective learning algorithm is derived for the unbiased estimation of the gradients.
- We empirically show that the proposed model improves the performance in language modeling and text generation.

## 2 Related Work

It has been observed that VAEs tend to suffer from the posterior collapse issue, especially when powerful autoregressive decoders are used for modeling text sequences. A common solution is to warm up the KL term in the objective of the VAE by gradually increasing the KL weight from 0 to 1 during training (Bowman et al., 2016). Other proposed methods include randomly dropping words during decoding (Bowman et al., 2016), thresholding the KL term to retain free bits (Kingma et al., 2016; Razavi et al., 2019; Pelsmaeker and Aziz, 2019), adding auxiliary objectives to ensure the effective latent variable for decoding (Goyal et al., 2017; Zhao et al., 2017; Dieng et al., 2019), imposing conditional independence assumptions on the inputs of autoregressive decoders to limit the contextual capacity (Yang et al., 2017; Semeniuta et al., 2017), replacing the Gaussian distribution with the von Mises-Fisher distribution to obtain a fixed KL term (Xu and Durrett, 2018). Also, the semi-amortized approach (Kim et al., 2018) is proposed to perform stochastic variational inference on top of amortized variational inference. Yet, an effective latent variable is learned at the cost of the computational complexity. Some recent work (He et al., 2019; Fu et al., 2019) attempts to improve the training procedure of the VAE without changing its original objective.

Another recent thread of research studies has focused on enhancing the dependency between the input and its latent variable. The skip connection was introduced in (Dieng et al., 2019) to enforce stronger links between the latent variable and the likelihood. InfoVAE (Zhao et al., 2019) introduces a mutual information term, approximated by maximum mean discrepancy, into the objective of the VAE. A similar approach was proposed in (Zhao et al., 2018b) for learning discrete representations.

Recent years have seen some attempts for unsupervised learning through the use of the mutual information. Neural estimation for the MI was proposed in (Brakel and Bengio, 2017), where an encoder and a discriminator are trained to minimize the JSD-based MI. This framework has been further improved and extended, including DeepInfoMax (Hjelm et al., 2019) for image classification

and Deep Graph InfoMax (Veličković et al., 2019) for graph-structured data. Contrastive predicting coding (van den Oord et al., 2018) is an architecture to learn global representations by maximizing the mutual information between the past and future information. Other than the neural estimation approach, the Monte Carlo method can also be used to estimate the MI (Zhao et al., 2018b; Dieng et al., 2019). Yet it suffers from biased estimation and high computational cost.

## 3 Background

### 3.1 Variational Autoencoders (VAEs)

Deep generative models define a joint distribution over a set of random variables composed in multiple layers of hierarchies. The goal is to learn the model distribution $p_\theta(\mathbf{x})$ to fit the true data distribution $q(\mathbf{x})$ as well as possible. This can be done by minimizing the Kullback-Leibler (KL) divergence between two distributions, and it is equivalent to the maximum-likelihood objective:

$$\min_\theta \mathrm{KL}[q(\mathbf{x})||p_\theta(\mathbf{x})] = \max_\theta \mathbb{E}_{q(\mathbf{x})}[\log p_\theta(\mathbf{x})]. \tag{1}$$

In this paper, we focus on latent variable models, which define the marginal log-likelihood via a latent variable $\mathbf{z}$:

$$\log p_\theta(\mathbf{x}) = \log \int p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}. \tag{2}$$

Since exactly estimating $\log p_\theta(\mathbf{x})$ is typically intractable, the VAE instead optimizes a tractable Evidence Lower BOund (ELBO):

$$\mathbb{E}_{q(\mathbf{x})}[\log p_\theta(\mathbf{x})] \geq \mathcal{L}_{\text{ELBO}} = \mathbb{E}_{q(\mathbf{x})}\{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \mathrm{KL}[q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})]\}, \tag{3}$$

where the variational posterior $q_\phi(\mathbf{z}|\mathbf{x})$ and the generative distribution $p_\theta(\mathbf{x}|\mathbf{z})$ are parameterized by neural networks (also known as the encoder and the decoder). The prior $p(\mathbf{z})$ is assumed a standard Gaussian distribution $\mathcal{N}(0, \mathbf{I})$.

The ELBO consists of a reconstruction likelihood term that ensures $q_\phi(\mathbf{z}|\mathbf{x})$ to encode *enough* information to generate $\mathbf{x}$ and a KL regularizer that ensures $q_\phi(\mathbf{z}|\mathbf{x})$ to encode *little* information in the posterior by matching it to the prior. Ideally, the encoder would embed high-level abstractions of $\mathbf{x}$ into the latent variable $\mathbf{z}$ and guide the decoder to recover low-level details based on $\mathbf{z}$.

To generate text sequences $\mathbf{x} = \{x_1, \ldots, x_T\}$ with length $T$, the VAE encodes holistic properties of text sequences $\mathbf{x}$ into the latent variable $\mathbf{z}$. RNNs are used for both the encoder and decoder. For encoding, the last hidden state of the encoder RNN is mapped into the latent variable $\mathbf{z}$. For decoding, $\mathbf{z}$ is fed as an additional input to the decoder RNN at each step, and then the next word $x_t$ is generated conditional on the latent variable $\mathbf{z}$ and all preceding information $x_{<t}$.

### 3.2 Posterior Collapse

The ELBO will be tighter via optimizing parameters $\{\phi, \theta\}$. The optimal ELBO should be equal to the true data distribution, and thus

$$\mathcal{L}_{\text{ELBO}} \leq \mathbb{E}_{q(\mathbf{x})}[\log p_\theta(\mathbf{x})] \leq \mathbb{E}_{q(\mathbf{x})}[\log q(\mathbf{x})]. \tag{4}$$

If the autoregressive decoder $p_\theta(\mathbf{x}|\mathbf{z})$ is powerful enough to approximate well the true data distribution $q(\mathbf{x})$, the ELBO would be

$$\mathcal{L}_{\text{ELBO}} = \mathbb{E}_{q(\mathbf{x})}\{\log q(\mathbf{x}) - \mathrm{KL}[q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})]\}. \tag{5}$$

The optimal ELBO forces the variational posterior to be like the prior, regardless of how expressively $q_\phi(\mathbf{z}|\mathbf{x})$ is parameterized. The zero-forcing effect of the KL term causes two undesirable outcomes. (*i*) The latent variable $\mathbf{z}$ is independent of the input $\mathbf{x}$, meaning that it contains no information about $\mathbf{x}$. (*ii*) The reconstruction of $\mathbf{x}$ cannot benefit from the encoder at all, meaning that the VAE generates sequences without making use of the latent variable $\mathbf{z}$. This phenomenon is called *posterior collapse*.

It has been proposed in (Hoffman and Johnson, 2016) that decomposing the ELBO can produce an equivalent way to define the ELBO:

$$\mathcal{L}_{\text{ELBO}} = \mathbb{E}_{q_\phi(\mathbf{x},\mathbf{z})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \mathbb{I}_q[\mathbf{x}, \mathbf{z}] - \mathrm{KL}[q_\phi(\mathbf{z})||p(\mathbf{z})], \tag{6}$$

where $q_\phi(\mathbf{x}, \mathbf{z})$ denotes the *variational joint distribution* induced by the posterior $q_\phi(\mathbf{z}|\mathbf{x})$, $\mathbb{I}_q[\mathbf{x}, \mathbf{z}]$ denotes the mutual information between $\mathbf{x}$ and $\mathbf{z}$ under $q_\phi(\mathbf{x}, \mathbf{z})$, and $q_\phi(\mathbf{z}) = \frac{1}{N}\sum_{n=1}^{N} q_\phi(\mathbf{z}_n|\mathbf{x}_n)$ denotes the *aggregated posterior* (Makhzani et al., 2016). The zero-forcing effect of the KL indicates

$$\mathbb{I}_q[\mathbf{x}, \mathbf{z}] = \mathrm{KL}[q_\phi(\mathbf{z})||p(\mathbf{z})] = 0. \tag{7}$$

However, if the latent variable $\mathbf{z}$ is a good representation of $\mathbf{x}$, the mutual information between $\mathbf{x}$ and $\mathbf{z}$ should take a large value and the KL term

in Equation (3) would be non-zero. As the ELBO objective contains the negative term of the mutual information, achieving higher mutual information between $\mathbf{x}$ and $\mathbf{z}$ is in fact opposite to maximizing the ELBO. This provides another way to explain the posterior collapse issue.

### 3.3 Mutual Information

Mutual information measures the non-linear relationship between two variables $\mathbf{x}$ and $\mathbf{y}$:

$$\mathbb{I}[\mathbf{x}, \mathbf{y}] = \mathbb{E}_{p(\mathbf{x},\mathbf{y})}\left[\log \frac{p(\mathbf{x}, \mathbf{y})}{p(\mathbf{x})p(\mathbf{y})}\right], \tag{8}$$

where the MI is minimum if two random variables are statistically independent, or maximum when two variables contain identical information. So if the MI is high, the variables are highly predictive of each other.

This hints that we can learn a stochastic encoder $q_\phi(\mathbf{z}|\mathbf{x})$ to maximize the MI between the input $\mathbf{x}$ and its latent variable $\mathbf{z}$ under the variational joint distribution $q_\phi(\mathbf{x}, \mathbf{z})$, given by

$$\max_\phi \mathbb{I}_q[\mathbf{x}, \mathbf{z}] = \max_\phi \mathbb{E}_{q_\phi(\mathbf{x},\mathbf{z})}\left[\log \frac{q_\phi(\mathbf{x}, \mathbf{z})}{q(\mathbf{x})q_\phi(\mathbf{z})}\right], \tag{9}$$

where $q_\phi(\mathbf{z}|\mathbf{x})$ and $q_\phi(\mathbf{z})$ are distributions over the latent variable. This leads to a decoder-free model for the maximization of the MI. Yet the MI is hard to compute, as $q_\phi(\mathbf{z})$ involves a mixture of the data points. The Monte Carlo approach has been used to approximate the MI in the VAE (Dieng et al., 2019), which however incurs a high computational cost and easily suffers from biased estimation.

## 4 Proposed Approach

### 4.1 Model Formulation

As shown in Equation (6), maximizing the ELBO penalizes the mutual information between the input $\mathbf{x}$ and its latent variable $\mathbf{z}$. Due to the posterior collapse issue, the latent variable $\mathbf{z}$ does not represent high-level abstractions of $\mathbf{x}$, making the representation learning even harder. We explicitly introduce a mutual information regularization term into the original ELBO objective that prefers high mutual information between $\mathbf{x}$ and $\mathbf{z}$. This encourages the model to make effective use of the latent variable and alleviate the posterior collapse issue. More formally, we arrive at the following training objective:

$$\begin{aligned}\mathcal{L}_{\text{MI-ELBO}} =& \mathbb{E}_{q(\mathbf{x})}\{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z}) \\ &- \text{KL}[q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})]\} + \alpha\mathbb{I}_q[\mathbf{x}, \mathbf{z}],\end{aligned} \tag{10}$$

where the first two terms can be optimized through the reparameterization trick as in the ELBO and $\alpha$ is a hyperparameter, which balances the trade-off between the inference and generation. We choose to compute the mutual information under the variational joint distribution, as it focuses on the latent space and thus allows us to organize the latent space. The consequence is that the variational posteriors would be more diverse in the latent space characterizing different input sequences, while the KL regularizer restricts the posteriors to match the Gaussian prior, with less "holes" in between where the decoder cannot be trained.

There has been prior work leveraging the mutual information for improving the inference and generation in deep generative models. Some studies (Chen et al., 2016; Dieng et al., 2019) have attempted to maximize the mutual information under the *generative joint distribution* $p_\theta(\mathbf{x}, \mathbf{z})$. In this paper, we focus on explicitly maximizing the mutual information under the variational joint distribution $q_\phi(\mathbf{x}, \mathbf{z})$, which encourages the VAE to learn a useful latent variable $\mathbf{z}$ for improving word generation. Our proposed model is similar in the spirit to the InfoVAE (Zhao et al., 2019). Instead, we adopt neural networks to estimate the MI accurately in the high-dimensional space.

### 4.2 Mutual Information Formulation

The Jensen-Shannon divergence was employed to define the MI between two variables (Brakel and Bengio, 2017; Hjelm et al., 2019; Veličković et al., 2019). In this paper, we instead maximize the MI, defined by the KL divergence between the joint distribution and marginals. Due to the asymmetric discrepancy, the dependencies between the input and its latent variable can be enhanced. More formally, we define an energy function over the variational joint distribution to estimate the probability of each configuration between two variables (Bengio, 2009), given by:

$$q_\phi(\mathbf{x}, \mathbf{z}) = e^{f_\psi(\mathbf{x},\mathbf{z})}q(\mathbf{x})q_\phi(\mathbf{z})/Z_\psi, \tag{11}$$

where the energy function $f_\psi(\mathbf{x}, \mathbf{z})$ is parameterized by neural networks and the partition function $Z_\psi$ is defined as $\mathbb{E}_{q(\mathbf{x})q_\phi(\mathbf{z})}[e^{f_\psi(\mathbf{x},\mathbf{z})}]$.

Based on Equations (9) and (11), a lower bound on the mutual information $\mathbb{I}_q[\mathbf{x}, \mathbf{z}]$ is given by:

$$\mathbb{I}_q[\mathbf{x}, \mathbf{z}] = \mathbb{E}_{q_\phi(\mathbf{x},\mathbf{z})}[f_\psi(\mathbf{x}, \mathbf{z})] - \log\Big[\mathbb{E}_{q(\mathbf{x})q_\phi(\mathbf{z})}[e^{f_\psi(\mathbf{x},\mathbf{z})}]\Big] \geq \mathbb{E}_{q_\phi(\mathbf{x},\mathbf{z})}[f_\psi(\mathbf{x}, \mathbf{z})] - \xi \cdot \mathbb{E}_{q(\mathbf{x})q_\phi(\mathbf{z})}\Big[e^{f_\psi(\mathbf{x},\mathbf{z})}\Big] + \log \xi + 1, \tag{12}$$

where the concave function $\log[x]$ is approximated by a tangent line with the scope $\xi$ according to the Taylor expansion. The approximation is $\log[x] \leq \xi \cdot x - \log[\xi] - 1$, where different values of $\xi$ correspond to different tangent lines. The role of the energy function $f_\psi(\mathbf{x}, \mathbf{z})$ is similar to the *discriminator* in GANs, which learns to distinguish pairs that are sampled from $q(\mathbf{x})q_\phi(\mathbf{z}|\mathbf{x})$ or pairs that are independently sampled from $q(\mathbf{x})q_\phi(\mathbf{z})$. The maximization of the lower bound suggests that the energy function learns to assign higher values to the samples from $q(\mathbf{x})q_\phi(\mathbf{z}|\mathbf{x})$, rather than those from $q(\mathbf{x})q_\phi(\mathbf{z})$, thus enhancing the dependencies.

Since after computing the gradient derivative of $\log[\cdot]$ with respect to parameters $\psi$, the denominator would contain the expectation of the gradients, leading to biased estimation of the full batch gradients. To solve this, we approximate $\log[\cdot]$ with a tangent line for unbiased estimation of the gradients with respect to the optimal function $f^*_\psi(\mathbf{x}, \mathbf{z})$.

In order to achieve the tighter lower bound on the MI and unbiased estimation for the gradients, we use the convex duality approach (Jordan et al., 1999). The update procedure of model parameters $\{\phi, \psi\}$ can be described as follows.

(*i*). Fixing $\phi$ and $\psi$, we optimize Equation (12) with respect to $\xi$, and the optimal value $\xi^*$ can thus be obtained by

$$\xi^* = \underset{\xi>0}{\operatorname{argmax}}\Big\{-\xi \cdot \mathbb{E}_{q(\mathbf{x})q_\phi(\mathbf{z})}[e^{f_\psi(\mathbf{x},\mathbf{z})}] + \log \xi\Big\} = \mathbb{E}_{q(\mathbf{x})q_\phi(\mathbf{z})}\Big[e^{f_\psi(\mathbf{x},\mathbf{z})}\Big]. \tag{13}$$

(*ii*). Fixing $\xi$, we optimize Equation (12) with respect to $\phi$ and $\psi$:

$$\max_{\phi,\psi}\Big\{\mathbb{E}_{q_\phi(\mathbf{x},\mathbf{z})}[f_\psi(\mathbf{x}, \mathbf{z})] - \xi^* \cdot \mathbb{E}_{q(\mathbf{x})q_\phi(\mathbf{z})}[e^{f_\psi(\mathbf{x},\mathbf{z})}]\Big\}. \tag{14}$$

Here, we first consider the optimal energy function $f^*_\psi(\mathbf{x}, \mathbf{z})$ for any given posterior $q_\phi(\mathbf{z}|\mathbf{x})$ and tangent scope $\xi$.

**Proposition 1.** With the fixed $\phi$ and $\xi$, the optimal energy function $f^*_\psi(\mathbf{x}, \mathbf{z})$ according to the objective in Equation (12) is given by

$$f^*_\psi(\mathbf{x}, \mathbf{z}) = \log q_\phi(\mathbf{x}, \mathbf{z}) - \log[q(\mathbf{x})q_\phi(\mathbf{z})] - \log \xi, \tag{15}$$

where $f^*_\psi(\mathbf{x}, \mathbf{z})$ becomes the *pointwise mutual information* when $\xi = 1$. This suggests that the energy function assigns zero probability to the samples independently drawn from $q(\mathbf{x})q_\phi(\mathbf{z})$.

With the optimal energy function $f^*_\psi(\mathbf{x}, \mathbf{z})$, the max-max objective in Equation (12) can be reformulated as:

$$C(\phi, \psi^*) = \mathbb{E}_{q_\phi(\mathbf{x},\mathbf{z})}[f^*_\psi(\mathbf{x}, \mathbf{z})] - \xi \cdot \mathbb{E}_{q(\mathbf{x})q_\phi(\mathbf{z})}\Big[e^{f^*_\psi(\mathbf{x},\mathbf{z})}\Big] + \log \xi + 1 = \mathrm{KL}[q_\phi(\mathbf{x}, \mathbf{z})||q(\mathbf{x})q_\phi(\mathbf{z})] = \mathbb{I}_q[\mathbf{x}, \mathbf{z}]. \tag{16}$$

We show that maximizing the lower bound on the MI with respect to $\{\phi, \psi, \xi\}$ is equivalent to maximizing the KL divergence between the joint distribution and their marginals. Together with Equations (10) and (16) as well as Proposition 1, the optimization objective becomes:

$$\mathcal{L}_{\text{MI-ELBO}} = \mathbb{E}_{q(\mathbf{x})}\{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z}) - \mathrm{KL}[q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})]\} + \alpha C(\phi, \psi^*), \tag{17}$$

where $C(\phi, \psi^*)$ is defined as a function that maximizes the lower bound on the MI with the optimal $\psi^*$. Then, we compute the gradients of Equation (17) with respect to $\{\phi, \theta\}$. While it is easy to compute the gradient with respect to $\theta$, the gradient with respect to $\phi$ is hard to compute since $C(\phi, \psi^*)$ itself depends on $\phi$. Actually, when the function $f_\psi(\mathbf{x}, \mathbf{z})$ is optimal, the expectation of the gradients becomes zero, that is

$$\mathbb{E}_{q_\phi(\mathbf{x},\mathbf{z})}[\nabla_\phi f^*_\psi(\mathbf{x}, \mathbf{z})] - \mathbb{E}_{q(\mathbf{x})q_\phi(\mathbf{z})}\Big[\nabla_\phi e^{f^*_\psi(\mathbf{x},\mathbf{z})}\Big] = 0. \tag{18}$$

In practice, we can ignore the gradients once $f_\psi(\mathbf{x}, \mathbf{z})$ achieves the optimality. We refer the proposed model as *VAE-MINE*. The overall learning procedure is summarized in Algorithm 1.

## 5 Experiments and Results

### 5.1 Datasets

We adopt three benchmark datasets, Penn Treebank (Marcus et al., 1993), Stanford Natural Language Inference (Bowman et al., 2015) and Yahoo Answers (Yang et al., 2017) to evaluate whether the inferred latent variable can give better performance in language modeling and text generation. For data pre-processing, we set the maximum

**Algorithm 1:** VAE with Mutual Information Neural Estimation (VAE-MINE)

Initialize the parameters of the encoder, decoder, energy function $\{\phi, \theta, \psi\}$ and tangent scope $\xi$;
**repeat**
  Sample mini-batch of $M$ sentences $\{\mathbf{x}_1, \ldots, \mathbf{x}_M\}$ from the data distribution $q(\mathbf{x})$;
  Sample mini-batch of $M$ latent variables $\{\mathbf{z}_1, \ldots, \mathbf{z}_M\}$ from the encoder $q_\phi(\mathbf{z}|\mathbf{x})$;
  Shuffle $M$ latent variables $\{\mathbf{z}'_1, \ldots, \mathbf{z}'_M\}$;
  Update $\psi$ by ascending its stochastic gradient:
  $\frac{1}{M}\sum_{i=1}^{M}\nabla_\psi\Big[f_\psi(\mathbf{x}_i, \mathbf{z}_i) - \xi \cdot e^{f_\psi(\mathbf{x}_i, \mathbf{z}'_i)}\Big]$;
  Compute the optimal tangent scope $\xi^*$:
  $\xi^* = \frac{1}{M}\sum_{i=1}^{M} e^{f_\psi(\mathbf{x}_i, \mathbf{z}'_i)}$;
  Update $\phi$ by ascending its stochastic gradient:
  $\frac{1}{M}\sum_{i=1}^{M}\nabla_\phi[\log p_\theta(\mathbf{x}_i|\mathbf{z}_i) - \mathrm{KL}[q_\phi(\mathbf{z}_i|\mathbf{x}_i)||p(\mathbf{z})]]$
  $+\frac{1}{M}\sum_{i=1}^{M}\nabla_\phi\Big[f_\psi(\mathbf{x}_i, \mathbf{z}_i) - \xi \cdot e^{f_\psi(\mathbf{x}_i, \mathbf{z}'_i)}\Big]$;
  Update $\theta$ by ascending its stochastic gradient:
  $\frac{1}{M}\sum_{i=1}^{M}\nabla_\theta \log p_\theta(\mathbf{x}_i|\mathbf{z}_i)$;
  Perform SGD-updates for parameters $\{\phi, \theta, \psi\}$;
**until** *convergence*;

| Dataset | #Train | #Valid | #Test | Length | Vocab |
|---|---|---|---|---|---|
| PTB | 42K | 3.3K | 3.7K | 21 | 10K |
| SNLI | 1.1M | 13K | 13K | 20 | 20K |
| Yahoo | 101K | 10K | 10K | 78 | 20K |

Table 1: Statistics of three benchmark datasets used in the experiments. Length denotes the average length of the sentences in the dataset and Vocab denotes the vocabulary size.

length of sentences to 200 and the maximum vocabulary size to 20K across all the datasets. Statistics of three datasets are summarized in Table 1.

### 5.2 Experimental Setup

We implement the VAE using a one-layer unidirectional LSTM (Hochreiter and Schmidhuber, 1997) with 512 hidden units and 128-dimensional word embedding for the encoder and decoder. The latent variable is 64-dimensional for all the models. We adopt the standard Gaussian prior and the diagonal Gaussian posterior. The last hidden state of the encoder RNN is fed into an MLP to estimate the mean and variance of the Gaussian posterior. Then, we sample the latent variable $\mathbf{z}$ and feed it to the decoder RNN at each step. The initial hidden state of the decoder RNN is obtained by feeding $\mathbf{z}$ to another MLP with the $\tanh()$ activation function. In order to further regularize the decoder RNN, dropout with a keep probability of 0.5 is applied to the inputs and outputs. We adopt SGD to optimize the model with a decayed learning rate and a gradient clipping.

Our architecture adopts an encoder RNN and an energy function parameterized by neural networks to maximize the lower bound on the MI. The energy function is trained by samples from the variational joint distribution and marginals. In practice, samples from the marginals are obtained by pairing one sentence $\mathbf{x}$ and another latent variable $\mathbf{z}'$ inferred by another sentence $\mathbf{x}'$. We adopt an MLP with two 512-unit hidden layers and ReLU activation to parameterize the energy function. As the encoder RNN and energy function optimize the same MI objective, the lower layers are shared between two modules. We set the value of the hyperparameter $\alpha$ within $[2, 5]$ with step $= 0.5$ for VAE-MINE and select the best $\alpha^*$ based on the validation dataset.

We compare the proposed VAE-MINE with the state-of-the-art baselines. LSTM-LM is an unconditional language model. For VAE-0.5 (Bowman et al., 2016), we implement KL annealing by increasing the KL weight linearly from 0.1 to 1.0 in the first 10 epochs and adopt word dropout rate of 0.5 to alleviate the posterior collapse. For VAE-BOW (Zhao et al., 2017), we replace word dropout with the bag-of-word (BOW) loss. For InfoVAE (Zhao et al., 2019), CyclicalVAE (Fu et al., 2019), and LaggingVAE (He et al., 2019), we follow the reported implementations.

### 5.3 Performance Evaluation

**Posterior Collapse.** We apply three metrics to evaluate the severity of the posterior collapse: KL divergence between the variational posterior and the prior, mutual information (MI), and the number of active units (AU) of the latent variable $\mathbf{z}$. If the KL in Equation (3) becomes zero, the posterior collapses, meaning that the variational posterior is equal to the prior. For the MI, we use the Monte Carlo approach to approximate Equation (6),

$$\mathrm{KL}[q_\phi(\mathbf{z})||p(\mathbf{z})] = \frac{1}{S}\sum_{s=1}^{S}[\log q_\phi(\mathbf{z}_s) - \log p(\mathbf{z}_s)], \tag{19}$$

where $S$ is the sample size.

The number of active units (Burda et al., 2015) is computed by

$$\mathrm{AU} = \sum_{d=1}^{D}\mathbb{I}\{\mathrm{Cov}_{q(\mathbf{x})}(\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[z_d]) \geq \epsilon\}, \tag{20}$$

where $z_d$ is the $d$-th dimension of $\mathbf{z}$ and the threshold $\epsilon$ is set to 0.01. $\mathbb{I}\{\cdot\}$ is an indicator giving 1 when its statement is true and 0 otherwise.

**Forward and Reverse Perplexity.** A common measure for the quality of the generated sentences is to evaluate the perplexity of a language model trained on the training or generated sentences. We fit a non-parametric Kneiser-Ney (KN) smoothed 5-gram language model (LM) (Heafield, 2011) on the sentences sampled from the model distribution $p_\theta(\mathbf{x})$ and the true data distribution $q(\mathbf{x})$. Two evaluation criteria are defined as *Forward Perplexity* (FPPL) and *Reverse Perplexity* (RPPL) (Zhao et al., 2018a; Cífka et al., 2018). Computing the FPPL is equivalent to training a KN 5-gram LM on the training sentences and reporting the perplexity on the 100K sentences generated from the model, while the RPPL involves training a KN 5-gram LM on the 100K generated sentences and reporting the perplexity on the test sentences.

### 5.4 Language Modeling Results

The results for language modeling are shown in Table 2. We report negative log-likelihood (NLL), KL divergence (KL), perplexity (PPL), mutual information (MI), the number of active units (AU), forward perplexity (FPPL) and reverse perplexity (RPPL).

According to Table 2, we observe that the KL value for VAE-0.5 is almost zero, indicating that the model suffers from posterior collapse. Similar situations can also be observed for InfoVAE. When the bag-of-word loss is used, the KL value increases, yet making VAE-BOW produce a larger NLL. This suggests that the latent variable does not contain useful information to reduce the reconstruction error. For VAE-MINE, we see that it can produce either comparable or better results over the evaluation metrics when compared with other state-of-the-art baselines. With the mutual information term considered for the optimization, we believe that VAE-MINE allows more reasonable correspondence patterns between the input and its inferred latent variable so as to better alleviate posterior collapse. Together with the original ELBO objective, the inferred posteriors of the different input sentences give an appropriate level of overlapping among them in the latent space, instead of sticking to the Gaussian prior. To summarize, the latent variable with non-trivial KL value helps the decoder RNN for better word-level generation.

Regarding the generation capability, we observe that VAE-MINE and LaggingVAE exhibit similar performance in terms of PPL and FPPL (sample quality) and outperform the other baselines by a significant margin. Particularly, VAE-MINE performs significantly better than others in terms of RPPL (sample diversity). We believe that this is due to the explicit use of the MI term to guide the learning process. It is consistent with our motivation that the MI-regularized objective can alleviate the posterior collapse to improve VAEs' capability of generating fluent and diverse sentences with the effective use of the latent variable. In contrast, LaggingVAE treats the MI as a stopping criterion for training the encoder RNN at the beginning of optimization. Even though the model can achieve good performance on alleviating the posterior collapse and generate fluent sentences, the generated sentences lack diversity. VAE-MINE can achieve relatively lower RPPL values, which suggests that it can achieve a good balance between sample diversity and training quality. This is not surprising that maximizing the MI would produce more diverse sentences. Also, the encoder parameters can benefit from the gradients from the energy function during training.

### 5.5 Reconstruction, Interpolation and Generation

We qualitatively evaluate the sentences generated using VAE-MINE as well as other baselines. The VAE variants can reconstruct input sentences by encoding them into the latent space and then decoding the means or samples drawn from the variational posteriors. Table 3 shows two sets of sentences generated using VAE-MINE with a greedy decoding based on two particular inputs. It shows that sentences involving similar underlying structures and concepts can be generated.

In order to evaluate the quality of the inferred latent space, we interpolate two points in the latent space and then decode the interpolated points for generating sentences. Table 4 shows the transitions of the generated sentences between two particular sentences in the SNLI dataset using different models. LaggingVAE turns out to be performing the worst by generating repetitive sentences. This explains why it gives large RPPL values as shown in Table 2. For VAE-MINE, we find that it can generate smoothly transitioned sentences in term of sentence semantics and syntactics via linear interpolation in the latent space. This demonstrates that the proposed VAE-MINE can learn the latent representations which smoothly fill up the

| Dataset | Model | NLL (KL) | PPL | MI | AU | FPPL | RPPL |
|---|---|---|---|---|---|---|---|
| PTB | LSTM-LM | 102.85 (–) | 95.63 | – | – | 461.44 | 472.86 |
| | VAE-0.5 | 104.06 (0.03) | 100.90 | 0.07 | 2 | 339.32 | 444.81 |
| | VAE-BOW | 105.58 (4.35) | 107.94 | 1.65 | 4 | 385.77 | 463.34 |
| | InfoVAE | 108.37 (0.02) | 122.15 | 0.06 | 0 | 483.71 | 515.09 |
| | CyclicalVAE | 104.36 (4.93) | 102.25 | 2.84 | 10 | 419.61 | 464.78 |
| | LaggingVAE | 100.93 (4.84) | 87.82 | 2.84 | **13** | **12.64** | 842.02 |
| | VAE-MINE | **100.91** (4.86) | **87.75** | **3.01** | **13** | 13.34 | **400.21** |
| SNLI | LSTM-LM | 33.41 (–) | 18.97 | – | – | 49.99 | 289.30 |
| | VAE-0.5 | 33.55 (0.01) | 19.20 | 0.01 | 0 | 45.04 | 281.15 |
| | VAE-BOW | 37.00 (13.57) | 26.06 | 2.07 | 23 | 44.72 | 278.31 |
| | InfoVAE | 33.88 (0.02) | 19.78 | 0.09 | 0 | 45.34 | 295.38 |
| | CyclicalVAE | 34.84 (6.39) | 21.52 | 3.12 | 28 | 58.16 | 291.53 |
| | LaggingVAE | **33.10** (1.34) | **18.03** | 0.73 | 3 | **4.02** | 173.47 |
| | VAE-MINE | 33.13 (4.05) | 18.09 | **3.15** | **30** | 6.21 | **152.63** |
| Yahoo | LSTM-LM | 348.95 (–) | 78.99 | – | – | 461.44 | 472.85 |
| | VAE-0.5 | 340.64 (0.63) | 71.17 | 0.44 | 3 | 178.97 | 252.51 |
| | VAE-BOW | 343.39 (10.13) | 73.66 | 1.99 | 7 | 182.95 | 256.85 |
| | InfoVAE | 344.84 (0.03) | 74.21 | 0.01 | 1 | 161.93 | 260.77 |
| | CyclicalVAE | 335.40 (7.51) | 75.56 | 3.29 | 12 | 193.79 | 256.28 |
| | LaggingVAE | **328.69** (6.65) | **61.29** | 2.99 | 17 | **5.60** | 843.56 |
| | VAE-MINE | 328.70 (6.70) | 61.31 | **4.34** | **18** | 6.72 | **185.33** |

Table 2: Language modeling results on the PTB, SNLI and Yahoo datasets. We report negative log-likelihood (NLL), KL divergence (KL), perplexity (PPL), mutual information (MI), the number of active units (AU) of the latent variable $\mathbf{z}$, forward perplexity (FPPL) and reverse perplexity (RPPL). We express the NLL as the ELBO. **Bold** numbers indicate the best performance and underlined numbers indicate the second best performance.

| Input: | **the man is at home sleeping.** | **three girls are sitting at desks and appear to be working intently.** |
|---|---|---|
| Output1: | the man is at home sleeping. | three women are sitting at desks and looking to be working together. |
| Output2: | the woman is sleeping at home. | Two ladies are standing at desks and appear to be playing together. |
| Output3: | the woman is working at home. | three women are sitting at desks and looking to be playing together. |

Table 3: Sentences generated from the variational posteriors of the latent variable based on SNLI dataset.

latent space.

Next, we compare generated sentences by sampling latent variables from the Gaussian prior and then greedy decoding. In Table 5, we observe that sentences generated from LaggingVAE are not as diverse as the ones generated from VAE-MINE.

### 5.6 Nearest Neighbour Analysis

To better understand the learned representations, we did a nearest-neighbour analysis by choosing a sentence from the training set, ordering the training set in terms of the cosine distance computed in the latent space, then selecting the top three with the highest similarity. Our results in Table 6 show that VAE-MINE learns a good latent variable such that novel sentences can be chosen, due to the nature of MI-regularized objective.

## 6 Conclusion

Variational autoencoders suffer from the posterior collapse issue. In this paper, we propose to alleviate this problem by adding a mutual information regularization into the objective of the VAE. Explicitly optimizing the proposed objective allows to encode high-level abstractions effectively. We use an energy function parameterzied by neural networks and the convex duality approach to maximize the lower bound on the MI. Experimental results on three benchmark datasets show that the proposed model VAE-MINE in general outperforms other state-of-the-art baselines and can generate more reasonable sentences via linear interpolation in the latent space. One research direction is to explore how to generate long-form text with deep generative models organized in multiple layers of latent variables, due to natural language characterizing a hierarchical structure.

## Acknowledgments

The authors would like to thank the anonymous reviewers for their insightful comments.

## References

Alexander A. Alemi, Ben Poole, Ian Fischer, Joshua V. Dillon, Rif A. Saurous, and Kevin Murphy. 2018. Fixing a broken ELBO. In *Proceedings of the 35th International Conference on Machine Learning*, pages 159–168.

| | |
|---|---|
| InfoVAE | two men are sitting on a couch while they wait for their parents to work.<br>the woman is the world war.<br>a man telling his hand.<br>this man is sitting on bike.<br>man jet ski is catching colorful waves.<br>two little girl are happily not about to play twister. |
| CyclicalVAE | the men are ready to fight in the streets in front of the crowd with a sheet.<br>a woman is standing with the canister in a band.<br>a female age matured on the stage is looking in a microscope.<br>a man is eating a small meal at a nightclub.<br>the performer is taking his photos by a statue.<br>there is a crowd of people, and several women posing for a picture. |
| LaggingVAE | this church choir sings to the masses as they sing joyous songs from the book at a church.<br>this church choir sings to the masses as they sing joyous songs from the book at a church.<br>this church choir sings to the masses as they sing joyous songs from the book at a church.<br>a woman gets picture taken in front of the masses.<br>a woman gets picture taken in front of the masses.<br>a woman gets picture taken in front of the masses. |
| VAE-MINE | a choir including three people sing and dance on the stage in front of the masses.<br>two musicians are playing the drums and a girl sits on a piano.<br>a young family sits while waiting a girl at the bottom of a church.<br>a man takes a photo of the girl.<br>a little girl has a toy and a digital camera.<br>the woman with red shirt is smiling to the girl. |

Table 4: Sentences generated by interpolating between the encodings of "**this church choir sings to the masses as they sing joyous songs from the book at a church**." and "**a woman with a green headscarf, blue shirt and a very big grin**.".

| |
|---|
| the company said it will be able to sell its n stake in the u.s. |
| the company said it will be to pay $ n million in the debt |
| the company said it will be able to replace its existing UNK and UNK |
| the company said it would buy n shares of its common shares for $ n each |
| the stock market's plunge was a UNK shot of the company's stock market yesterday |
| mr. UNK said the agreement has n't yet to be determined by the company |
| but the japanese are struggling to the UNK of the u.s. government |
| the company also said it will sell its boston corp. unit to the company's UNK group |
| how do i get rid of UNK UNK? i have a UNK UNK and i have a UNK UNK. i have tried UNK and UNK. |
| how do you say "UNK"? i'm not sure what you mean. i'm not sure what you mean. |
| how do i get a UNK? i am a UNK student in college and i want to know how to get a job in the UNK area. |
| how do i get a job in a bank? i am a student. i have a UNK what is the best way to study in a UNK. |
| what is the difference between a democrat and republican? i think it is a UNK, but i do n't think UNK is a UNK. |
| what is the best city in the world? i'm looking for a good place to start a UNK, but i'm not sure if this is the best way. |
| how do i get a free music download? i want to download it from my computer and i want to know the name of the song. |
| who is the best player in the nba? i think UNK is a great player. |

Table 5: Qualitative comparisons on the generated sentences. First row: PTB samples generated from the Gaussian prior by LaggingVAE (upper half) and VAE-MINE (lower half). Second row: Yahoo samples generated from the Gaussian prior by LaggingVAE (upper half) and VAE-MINE (lower half).

| Query | **but the japanese are struggling to the UNK of the u.s. government** |
|---|---|
| LaggingVAE | the japanese UNK openly about the u.s. public's UNK<br>many UNK regard a u.s. presence as a desirable UNK to japanese influence<br>the u.s. government in recent years has accused japanese companies of UNK slashing prices |
| VAE-MINE | the u.s. government in recent years has accused japanese companies of UNK slashing prices<br>while the small deals are far less UNK they add to japanese UNK of the u.s. market<br>continuing demand for dollars from japanese investors boosted the u.s. currency |
| Query | **a person on a horse jumps over a broken down airplane.** |
| LaggingVAE | a horse jockey is jumping a horse over an obstacle on a coarse.<br>a person aquestrian horse jumping over a wooden fence.<br>a jockey riding a horse prepares to jump over an obstacle. |
| VAE-MINE | a horse jockey is jumping a horse over an obstacle on a coarse.<br>a jockey riding a horse prepares to jump over an obstacle.<br>a small child plays with her airplane as a cat looks on. |

Table 6: Sentences selected from the PTB and SNLI training set in terms of the cosine distance between the inferred latent variables.

Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeswar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and R. Devon Hjelm. 2018. Mutual information neural estimation. In *Proceedings of the 35th International Conference on Machine Learning*, pages 530–539.

Yoshua Bengio. 2009. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21.

Philemon Brakel and Yoshua Bengio. 2017. Maximizing independence with GANs for non-linear ICA. In *ICML Workshop on Implicit Models*.

Yuri Burda, Roger B. Grosse, and Ruslan Salakhutdinov. 2015. Importance weighted autoencoders. In *3rd International Conference on Learning Representations*.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180.

Xi Chen, Diederik P. Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2017. Variational lossy autoencoder. In *5th International Conference on Learning Representations*.

Ondřej Cífka, Aliaksei Severyn, Enrique Alfonseca, and Katja Filippova. 2018. Eval all, trust a few, do wrong to none: Comparing sentence generation models. *Computing Research Repository*, arXiv:1804.07972.

Adji B. Dieng, Yoon Kim, Alexander M. Rush, and David M. Blei. 2019. Avoiding latent variable collapse with generative skip models. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics*, pages 2397–2405.

Hao Fu, Chunyuan Li, Xiaodong Liu, Jianfeng Gao, Asli Çelikyilmaz, and Lawrence Carin. 2019. Cyclical annealing schedule: A simple approach to mitigating KL vanishing. In *Proceedings of the 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680.

Anirudh Goyal, Alessandro Sordoni, Marc-Alexandre Côté, Nan Rosemary Ke, and Yoshua Bengio. 2017. Z-forcing: Training stochastic recurrent networks. In *Advances in Neural Information Processing Systems*, pages 6716–6726.

Junxian He, Daniel Spokoyny, Graham Neubig, and Taylor Berg-Kirkpatrick. 2019. Lagging inference networks and posterior collapse in variational autoencoders. In *7th International Conference on Learning Representations*.

Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *Proceedings of the 6th Workshop on Statistical Machine Translation*, pages 187–197.

R. Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. 2019. Learning deep representations by mutual information estimation and maximization. In *7th International Conference on Learning Representations*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Matthew D. Hoffman and Matthew J. Johnson. 2016. ELBO surgery: yet another way to carve up the variational evidence lower bound. In *NIPS Workshop in Advances in Approximate Bayesian Inference*.

Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. 1999. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233.

Yoon Kim, Sam Wiseman, Andrew C. Miller, David Sontag, and Alexander M. Rush. 2018. Semi-amortized variational autoencoders. In *Proceedings of the 35th International Conference on Machine Learning*, pages 2683–2692.

Diederik P. Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. 2016. Improving variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, pages 4743–4751.

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational Bayes. In *2nd International Conference on Learning Representations*.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, and Ian Goodfellow. 2016. Adversarial autoencoders. In *4th International Conference on Learning Representations*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Gábor Melis, Chris Dyer, and Phil Blunsom. 2018. On the state of the art of evaluation in neural language models. In *6th International Conference on Learning Representations*.

Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. 2016. Pixel recurrent neural networks. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 1747–1756.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *Computing Research Repository*, arXiv:1807.03748.

Tom Pelsmaeker and Wilker Aziz. 2019. Effective estimation of deep generative language models. *Computing Research Repository*, arXiv:1904.08194.

Ali Razavi, Aäron van den Oord, Ben Poole, and Oriol Vinyals. 2019. Preventing posterior collapse with $\delta$-VAEs. In *7th International Conference on Learning Representations*.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1278–1286.

Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. 2017. A hybrid convolutional variational autoencoder for text generation. In *Empirical Methods in Natural Language Processing*, pages 627–637.

Trieu H. Trinh, Andrew M. Dai, Minh-Thang Luong, and Quoc V. Le. 2018. Learning longer-term dependencies in RNNs with auxiliary losses. In *Proceedings of the 35th International Conference on Machine Learning*, pages 4972–4981.

Petar Veličković, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. 2019. Deep graph Infomax. In *7th International Conference on Learning Representations*.

Prince Zizhuang Wang and William Yang Wang. 2019. Riemannian normalizing flow on variational wasserstein autoencoder for text modeling. In *Proceedings of the 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Jiacheng Xu and Greg Durrett. 2018. Spherical latent spaces for stable variational autoencoders. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4503–4513.

Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. 2017. Improved variational autoencoders for text modeling using dilated convolutions. In *Proceedings of the 34th International Conference on Machine Learning*, pages 3881–3890.

Junbo Zhao, Yoon Kim, Kelly Zhang, Alexander M. Rush, and Yann LeCun. 2018a. Adversarially regularized autoencoders. In *Proceedings of the 35th International Conference on Machine Learning*, pages 5897–5906.

Shengjia Zhao, Jiaming Song, and Stefano Ermon. 2019. InfoVAE: Balancing learning and inference in variational autoencoders. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*.

Tiancheng Zhao, Kyusong Lee, and Maxine Eskenazi. 2018b. Unsupervised discrete sentence representation learning for interpretable neural dialog generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 1098–1107.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 654–664.

# Sampling Bias in Deep Active Classification: An Empirical Study

**Ameya Prabhu**[1*†], **Charles Dognin**[2*], **Maneesh Singh**[2]
University of Oxford[1] Verisk | AI, Verisk Analytics[2]
ameya.prabhu@mailfence.com, {charles.dognin, maneesh.singh}@verisk.com

## Abstract

The exploding cost and time needed for data labeling and model training are bottlenecks for training DNN models on large datasets. Identifying smaller representative data samples with strategies like active learning can help mitigate such bottlenecks. Previous works on active learning in NLP identify the problem of sampling bias in the samples acquired by uncertainty-based querying and develop costly approaches to address it. Using a large empirical study, we demonstrate that active set selection using the posterior entropy of deep models like FastText.zip (FTZ) is robust to sampling biases and to various algorithmic choices (query size and strategies) unlike that suggested by traditional literature. We also show that FTZ based query strategy produces sample sets similar to those from more sophisticated approaches (e.g ensemble networks). Finally, we show the effectiveness of the selected samples by creating tiny high-quality datasets, and utilizing them for fast and cheap training of large models. Based on the above, we propose a simple baseline for deep active text classification that outperforms the state-of-the-art. We expect the presented work to be useful and informative for dataset compression and for problems involving active, semi-supervised or online learning scenarios. Code and models are available at: https://github.com/drimpossible/Sampling-Bias-Active-Learning

## 1 Introduction

Deep neural networks (DNNs) trained on large datasets provide state-of-the-art results on various NLP problems (Devlin et al., 2019) including text classification (Howard and Ruder, 2018). However, the cost and time needed to get labeled data and to train models is a serious impediment to creating new and/or better models. This problem can be mitigated by creating smaller representative datasets with active learning which can be used for training DNNs to achieve similar test accuracy as that using the full training dataset . In other words, the smaller sample can be considered a surrogate for the full data.

However, there is lack of clarity in the active learning literature regarding sampling bias in such surrogate datasets created using active learning (Settles, 2009): its dependence on models, functions and parameters used to acquire the sample. Indeed, what constitutes a good sample? In this paper, we perform an empirical investigation using active text classification as the application.

Early work in active text classification (Lewis and Gale, 1994) suggests that greedy query generation using label uncertainty may lead to efficient representative samples (Nonetheless, the same test accuracy). Subsequent concerns regarding sampling bias has lead to explicit use of expensive diversity measures (Brinker, 2003; Hoi et al., 2006) in acquisition functions or using ensemble approaches (Liere and Tadepalli, 1997; McCallum and Nigam, 1998) to improve diversity implicitly.

Deep active learning approaches adapt the discussed framework above to train DNNs on large data. However, it is not clear if the properties of deep approaches mirror those of their shallow counterparts and if the theory and the empirical evidence regarding sampling efficiency and bias translates from shallow to deep models. For example, (Sener and Savarese, 2018) and (Ducoffe and Precioso, 2018) find that uncertainty based strategies perform no better than random sampling even if ensembles are used and using diversity measures outperform both. On the other hand, (Beluch et al., 2018; Gissin and Shalev-Shwartz, 2019) find that uncertainty measures computed with ensembles outperform diversity based approaches while (Gal et al., 2017; Beluch et al., 2018; Siddhant and Lipton, 2018) find them to outperform uncertainty measures computed using single models. A recent empirical study (Siddhant and Lipton, 2018) investigating active learning in NLP suggests that Bayesian active learning outperforms classical uncertainty sampling across all settings. However, the approaches have been limited to rel-

[*] indicates equal contribution
[†] Work done at Verisk | AI



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4058–4068, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

atively small datasets.

### 1.1 Sampling Bias in Active Classification

In this paper, we investigate the issues of sampling bias and sample efficiency, the stability of the actively collected query and train sets and the impact of algorithmic factors - i.e. the setup chosen while training the algorithm, in the context of deep active text classification on large datasets. In particular, we consider two sampling biases: label and distributional bias, three algorithmic factors: initial set selection, query size and query strategy along with two trained models and four acquisition functions on eight large datasets.

To isolate and evaluate the impact of the above (combinatorial) factors, a large experimental study was necessary. Consequently, we conducted over 2.3K experiments on 8 popular, large, datasets of sizes ranging from 120K-3.6M. Note that the current trend in deep learning is to train large models on very large datasets. However, the aforementioned issues have not yet been investigated in the literature in such a setup. As shown in Table 1, the datasets used in latest such analysis on active text classification by (Siddhant and Lipton, 2018) are quite small in comparison. The datasets used by us are two orders of magnitude larger, our query samples often being the size of the entire datasets used, and the presented empirical study is more extensive (20x experiments).

Our findings are as follows:

(i) We find that utilizing the uncertainty query strategy using a deep model like FastText.zip (FTZ)[1] to actively construct a representative sample provides query and train sets with remarkably good sampling properties.

(ii) We finds that a single deep model (FTZ) used for querying provides a sample set similar to more expensive approaches using ensemble of models. Additionally, the sample set has a large overlap with support vectors of an SVM trained on the entire dataset largely invariant to a variety of algorithmic factors, thus indicating the robustness of the acquired sample set.

(iii) We demonstrate that the actively acquired training datasets can be utilized as small, surrogate training sets with a 5x-40x compression for training large, deep text classification models. In particular, we can train the ULMFiT (Howard and Ruder, 2018) model to state of the art accuracy at 25x-200x speedups.

(iv) Finally, we create a novel, state-of-the-art baseline for active text classification which outperforms recent work (Siddhant and Lipton, 2018), using Bayesian dropout, utilizing 4x less training data. We also outperform (Sener and Savarese, 2018) at all training data sizes. The latter uses an expensive diversity based query strategy (coreset sampling).

The rest of the paper is organized as follows: in Section 2, the experimental methodology and setup are described. Section 3 presents the experimental study on sampling biases as well as the impact of various algorithmic factors. In Section 4, we compare with prior literature in active text classification. Section 5 presents a downstream use case - fast bootstrapping of the training of very large models like ULMFiT. Finally, we discuss the current literature in light of our work in Section 6 and summarize the conclusions in Section 7.

## 2 Methodology

This section describes the experimental approach and the setup used to empirically investigate the issues of (i) sampling bias and (ii) sampling efficiency in creating small samples to train deep models.

### 2.1 Approach

A labelled training set is incrementally built from a pool of unlabeled data by selecting & acquiring labels from an oracle in sequential increments. In this, we follow the standard approach found in the active learning literature. We use the following terminology:

**Queries & Query Strategy:** We refer to the (incremental) set of points selected to be labeled and added to the training as the query and the (acquisition) function used to select the samples as the query strategy.

**Pool & Train Sets:** The pool is the unlabeled data from which queries are iteratively selected, labeled and added to the (labeled) train set.

Let $\mathcal{D}_S = (\mathbf{x}_i, y_i)$ denote a dataset consisting of $|\mathcal{S}| = n$ i.i.d samples of data/label pairs, where $|.|$ denotes the cardinality. Let $\mathcal{S}_0 \subset \mathcal{S}$ denote an initial randomly drawn sample from the initial pool. At each iteration, we train the model on the current train set and use a model-dependent query strategy

[1] We use FastText.zip (FTZ) to optimize the time and resources needed for this study.

to acquire new samples from the pool, get them labeled by an oracle and add them to the train set. Thus, a sequence of training sets: $[\mathcal{S}_1, \mathcal{S}_2 \ldots, \mathcal{S}_b]$ is created by sampling $b$ queries from the pool set, each of size $K$. The $b$ queries are given by $[\mathcal{S}_1 - \mathcal{S}_0, \mathcal{S}_2 - \mathcal{S}_1 \ldots, \mathcal{S}_b - \mathcal{S}_{b-1}]$. Note that $|\mathcal{S}_i| = (|\mathcal{S}_0| + i \times K)$ and $\mathcal{S}_1 \subset \mathcal{S}_2 \ldots \subset \mathcal{S}_b \subset \mathcal{S}$.

In this paper, we investigate the efficiency and bias of sample sets $\mathcal{S}_b^1, \mathcal{S}_b^2, \ldots, \mathcal{S}_b^t$ obtained by different query strategies $Q^1, Q^2, \ldots Q^t$. We exclude the randomly acquired initial set and perform comparisons on the actively acquired sample sets defined as $\hat{\mathcal{S}}_j^i = (\mathcal{S}_j^i - \mathcal{S}_0^i)$.

### 2.2 Experimental Setup

In this section, we share details of the experimental setup, and present and explain the choice of the datasets, models and query strategies used.

**Datasets:** We used eight, large, representative datasets widely used for text classification: AGNews (AGN), DBPedia (DBP), Amazon Review Polarity (AMZP), Amazon Review Full (AMZF), Yelp Review Polarity (YRP), Yelp Review Full (YRF), Yahoo Answers (YHA) and Sogou News (SGN). Please refer to Section 4 of (Zhang et al., 2015) for details regarding the collection and characteristics of these datasets. Table 1 provides a comparison regarding the choice of datasets, models and number of experiments between our study and (Siddhant and Lipton, 2018) which investigates a variety of NLP tasks including text classification while we focus only on the latter.

**Models:** We reported two text classification models as representatives of classical and deep learning approaches respectively which were fast to train and also had good performance on text classification: Multinomial Naive Bayes (MNB) with TF-IDF (Wang and Manning, 2012) and FastText.zip (FTZ) (Joulin et al., 2016). The FTZ model provides results competitive with VDCNNs (a 29 layer CNN) (Conneau et al., 2017) but with over 15,000$\times$ speedup (Joulin et al., 2017). This allowed us to conduct a thorough empirical study on large datasets. Multinomial Naive Bayes (MNB) with TF-IDF features is a popularly claimed baseline for text classification (Wang and Manning, 2012).

**Query Strategies:** Uncertainty based query strategies are widely used and well studied in the active learning literature. Those strategies typically use a scoring function on the (softmax) output of a single model. We evaluate the following ones: Least Confidence (LC) and Entropy (Ent). Independently training ensembles of models (Lakshminarayanan et al., 2017) is another principled approach to obtain uncertainties associated with the output estimate.Then, we tried four query strategies - LC and Ent computed using single and ensemble models and evaluated them against random sampling (chance) as a baseline. For ensembles, we used five FTZ ensembles (Lakshminarayanan et al., 2017). In contrast, (Siddhant and Lipton, 2018) used Bayesian ensembles using Dropout, proposed in (Gal et al., 2017). Please refer to Section 4 for a comparison.

| Paper | #Exp | Datasets (#Train) | Models (Full Acc) |
|---|---|---|---|
| DAL | 120 | TREC-QA (6k), MAReview (10.5k) | SVM (89%), CNN (91%), LSTM (92%) |
| Ours | 2.3K | AGN (120k), SGN (450k), DBP (560k), YRP (560k), YRF (650k), YHA (1400k), AMZP (3600k), AMZF (3000k) | FTZ (97%), MNB (90%) |

Table 1: Comparison of active text classification datasets and models (Acc on Trec-QA) used in (Siddhant and Lipton, 2018) and our work. We use significantly larger datasets (two orders larger), perform 20x more experiments, and use more efficient and accurate models.

**Implementation Details:** We performed 2304 active learning experiments. We obtained our results on three random initial sets and three runs per seed (to account for stochasticity in FTZ) for each of the eight datasets. The query sizes were $0.5\%$ of the dataset for AGN, AMZF, YRF and YHA and $0.25\%$ for SGN, DBP, YRP and AMZP respectively for $b = 39$ sequential, active queries. We also experimented with different query sizes keeping the size of the final training data $b \times K$ constant. The default query strategy uses a single model with output Entropy (Ent) unless explicitly stated otherwise. Results in the chance column are obtained using random query strategy.

We used Scikit-Learn (Pedregosa et al., 2011) implementation for MNB and original implementation for FastText.zip (FTZ) [2]. We required 3 weeks of running time for all FTZ experiments on a x1.16xlarge AWS instance with Intel Xeon E7-8880 v3 processors and 1TB RAM to obtain results presented in this work. The experiments are deterministic beyond the stochasticity involved in training the FTZ model, random initialization and

[2] https://github.com/facebookresearch/fastText

| Dsets | Limit | FTZ ($\cap Q$) | MNB ($\cap Q$) | FTZ ($\cap S$) | MNB ($\cap S$) |
|---|---|---|---|---|---|
| SGN | 1.61 | $1.56 \pm 0.03$ | $1.15 \pm 0.32$ | $1.59 \pm 0.01$ | $1.57 \pm 0.01$ |
| DBP | 2.64 | $2.50 \pm 0.02$ | $2.27 \pm 0.11$ | $2.51 \pm 0.0$ | $2.58 \pm 0.01$ |
| YHA | 2.30 | $2.25 \pm 0.01$ | $2.22 \pm 0.02$ | $2.25 \pm 0.0$ | $2.28 \pm 0.0$ |
| YRP | 0.69 | $0.69 \pm 0.0$ | $0.56 \pm 0.13$ | $0.69 \pm 0.0$ | $0.69 \pm 0.01$ |
| YRF | 1.61 | $1.56 \pm 0.02$ | $1.42 \pm 0.21$ | $1.56 \pm 0.0$ | $1.57 \pm 0.01$ |
| AGN | 1.39 | $1.33 \pm 0.04$ | $1.13 \pm 0.17$ | $1.33 \pm 0.0$ | $1.35 \pm 0.01$ |
| AMZP | 0.69 | $0.69 \pm 0.0$ | $0.69 \pm 0.0$ | $0.69 \pm 0.0$ | $0.69 \pm 0.0$ |
| AMZF | 1.61 | $1.58 \pm 0.02$ | $1.6 \pm 0.01$ | $1.59 \pm 0.0$ | $1.61 \pm 0.0$ |

Table 2: Label entropy with a large query size ($b = 9$ queries). $\cap Q$ denotes averaging across queries of a single run, $\cap S$ denotes the label entropy of the final collected samples, averaged across seeds. Naive Bayes ($\cap Q$) has biased (inefficient) queries while FastText ($\cap Q$) shows stable, high label entropy showing a rich diversity in classes despite the large query size. Overall, the resultant sample ($\cap S$) becomes balanced in both cases.

SGD updates. The entire list of hyperparameters and metrics affecting uncertainty such as calibration error (Guo et al., 2017) is given in the supplementary material. The experimental logs and models are available on our github link[3].

## 3 Results

In this section, we study several aspects of sampling bias (class bias, feature bias) and the impact of relevant algorithmic factors (initial set selection, query size and query strategy.

We evaluated the actively acquired queries and sample set for sampling bias, and for the stability as measured by %intersection of collected sets across a critical influencing factor. Higher sample intersections indicate more stability increase to the chosen influencing factor.

### 3.1 Aspects of Sampling Bias

We study two types of sampling biases: (a) Class Bias and (b) Feature Bias.

#### 3.1.1 Class Bias

Greedy uncertainty based query strategies are said to pick disproportionately from a subset of classes per query (Sener and Savarese, 2018; Ebert et al., 2012), developing a lopsided representation in each query. However, its effect on the resulting sample set is not clear. We test this by measuring the Kullback-Leibler (KL) divergence between the ground-truth label distribution and the distribution obtained per query as one experiment ($\cap Q$), and over the resulting sample ($\cap S$) as the second. Let us denote $P$ as the true distribution of labels, $\hat{P}$ the sample distribution and $C$ the total number of classes. Since $P$ follows a uniform distribution, we can use Label entropy instead ($L = -KL(P||\hat{P}) + log(C)$). Label entropy $L$ is an intuitive measure. The maximum label entropy is reached when sampling is uniform, $\hat{P}(x) = P(x)$, i.e. $L = log(C)$.

We present our results in Table 2. We observe that across queries ($\cap Q$), FTZ with entropy strategy has a balanced representation from all classes (high mean) with a high probability (low std) while Multinomial Naive Bayes (MNB) results in more biased queries (lower mean) with high probability (high std) as studied previously. However, we did not find evidence of class bias in the resulting sample ($\cap S$) in both models: FastText and Naive Bayes (column 5 and 6 from Table 2).

We conclude that entropy as a query strategy can be robust to class bias even with large query sizes.

#### 3.1.2 Feature Bias

Uncertainty sampling can lead to undesirable sampling bias in feature space (Settles, 2009) by repeating redundant samples and picking outliers (Zhu et al., 2008). Diversity-based query strategies (Sener and Savarese, 2018) are used to address this issue, by selecting a representative subset of the data. In the context of active classification, it is good to pick the most informative samples to be the ones closer to class boundaries[4].

Indeed, recent work suggests that the learning in deep classification networks may focus on small part of the data closer to class boundaries, thus resembling support vectors (Xu et al., 2018; Toneva et al., 2019). To investigate whether uncertainty sampling also exhibits this behavior, we perform below a direct comparison with support vectors from a SVM. For this, we train a FTZ model on the full training data and train a SVM on the resulting features (sentence embeddings) to obtain the support vectors and compute the intersection of support vectors with each selected set. The percentage intersections are shown in Table 3. The high percentage overlap is a surprising result which shows that the sampling is indeed biased but in

[3] https://github.com/drimpossible/Sampling-Bias-Active-Learning

[4] In this work, we assume ergodicity in the setup. We do not consider incremental, online modeling scenarios where new modes or new classes are sequentially encountered.

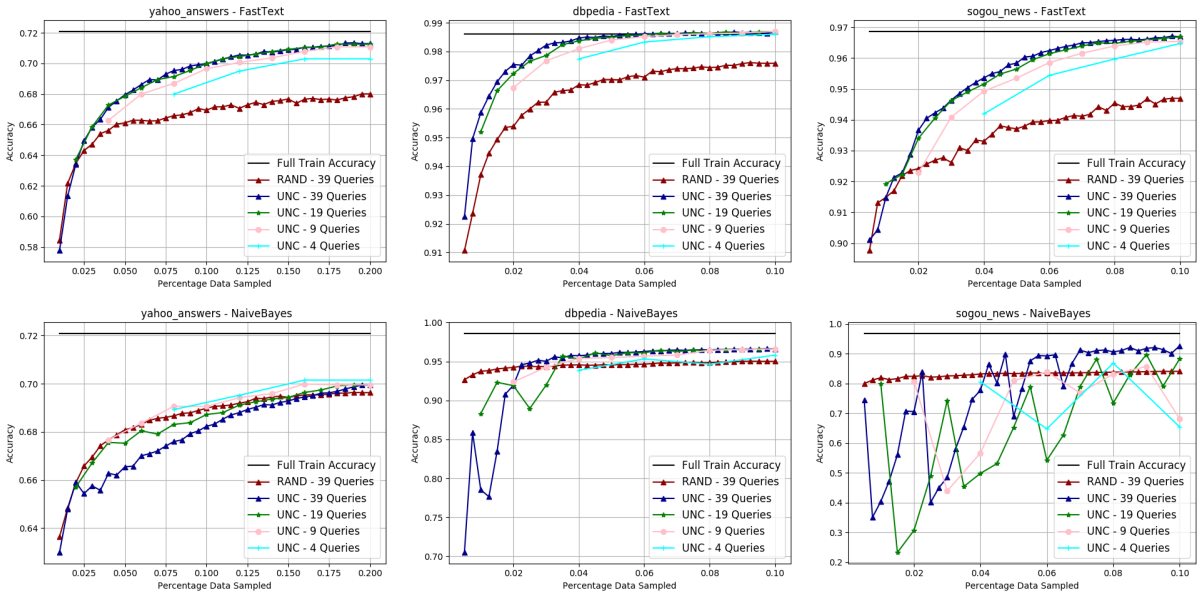


Figure 1: Accuracy across different number of queries $b$ for FastText and Naive Bayes, with $b \times K$ constant. FastText is robust to increase in query size and significantly outperforms random in all cases. Naive Bayes: (Left) All including $b$=39 perform worse than random, (Center) All including b=9 eventually perform better than random (Right) $b = 39$ performs better than random but larger query sizes perform worse than random. Uncertainty sampling with Naive Bayes suffers from sampling size bias.

| Dsets | Common% | Chance% | $\#SV$ |
|---|---|---|---|
| SGN | $71.3 \pm 0.5$ | $9.3 \pm 0.5$ | 13184 |
| DBP | $86.3 \pm 0.5$ | $9.7 \pm 0.5$ | 1479 |
| YRP | $57.3 \pm 0.5$ | $9.7 \pm 0.5$ | 31750 |
| AGN | $45.0 \pm 0.8$ | $21.0 \pm 1.6$ | 1032 |

Table 3: Proportion of Support Vectors intersecting with our actively selected set calculated by $\frac{|\mathcal{S}_{SV} \cap \hat{\mathcal{S}}_b|}{|\mathcal{S}_{SV}|}$. Actively selected sets share large overlap with supports of an SVM (critical for classification).

a desirable way. Since the support vectors represent the class boundaries, a large percentage of selected data consists of samples around the class boundaries. This overlap indicates that the actively acquired training sample covers the support vectors well which are important for good classification performance. The overlap with the support vectors of an SVM (a fixed algorithm) also suggests that uncertainty sampling using deep models might generalize beyond FastText, to other learning algorithms.

**Experimental Details:** We used a fast GPU implementation for training an SVM with a linear kernel (Wen et al., 2018) with default hyperparameters. Please refer to supplementary material for additional details. We ensured the SVM achieves similar accuracies as original FTZ model.

| Dsets | Chance | FTZD | FTZS | MNBD | MNBS |
|---|---|---|---|---|---|
| SGN | 0.8 | 77.8 | 81.0 | 55.5 | 100.0 |
| DBP | 0.9 | 79.7 | 81.3 | 79.7 | 100.0 |
| YHA | 3.7 | 69.0 | 73.6 | 89.5 | 100.0 |
| YRP | 0.9 | 42.9 | 43.7 | 16.0 | 100.0 |
| YRF | 3.6 | 67.7 | 71.6 | 13.6 | 100.0 |
| AGN | 3.7 | 68.7 | 70.1 | 79.8 | 100.0 |
| AMZP | 0.9 | 48.4 | 48.8 | 15.0 | 100.0 |
| AMZF | 3.6 | 56.8 | 63.1 | 57.8 | 100.0 |

Table 4: % Intersection of samples obtained with different seeds (ModelD) compared to same seeds (ModelS) and chance intersection for $b = 39$ queries. We see that FastText is initialization independent (FTZD $\approx$ FTZS $\gg$ Chance). NaiveBayes shows significant dependency on the initial set sometimes, while other times performs comparable to FastText.

### 3.2 Algorithmic Factors

We analyze three algorithmic factors of relevance to sampling bias: (a) Initial set selection (b) Query size, and, (c) Query strategy.

#### 3.2.1 Initial Set Selection

To investigate the dependence of the actively acquired train set on the initial set, we compare the overlap (intersection) of the incrementally constructed sets from different random initial sets versus the same initial set. The results are shown in Table 4. We first observe that chance overlaps (column 2) are very low - less than 4%. Columns

| Dsets | Chance | FTZ $9 \cap 19 \cap 39$ | FTZ $39 \cap 39 \cap 39$ | MNB $9 \cap 19 \cap 39$ | MNB $39 \cap 39 \cap 39$ |
|---|---|---|---|---|---|
| SGN | $0.83 \pm 0.0$ | $77.0 \pm 0.5$ | $77.9 \pm 0.2$ | $31.9 \pm 0.0$ | $55.5 \pm 0.0$ |
| DBP | $0.9 \pm 0.0$ | $80.0 \pm 0.1$ | $79.6 \pm 0.2$ | $82.3 \pm 0.0$ | $79.7 \pm 0.0$ |
| YHA | $3.7 \pm 0.0$ | $68.3 \pm 0.1$ | $69.0 \pm 0.0$ | $92.1 \pm 0.0$ | $89.5 \pm 0.0$ |
| YRP | $0.9 \pm 0.0$ | $46.0 \pm 0.9$ | $42.7 \pm 1.0$ | $10.8 \pm 0.0$ | $16.0 \pm 0.0$ |
| YRF | $3.6 \pm 0.0$ | $68.4 \pm 0.2$ | $67.6 \pm 0.1$ | $14.2 \pm 0.0$ | $13.6 \pm 0.0$ |
| AGN | $3.7 \pm 0.0$ | $70.3 \pm 0.2$ | $68.7 \pm 0.1$ | $81.6 \pm 0.0$ | $79.8 \pm 0.0$ |
| AMZP | $0.9 \pm 0.0$ | $45.8 \pm 0.1$ | $48.2 \pm 0.2$ | $11.5 \pm 0.0$ | $15.0 \pm 0.0$ |
| AMZF | $3.6 \pm 0.0$ | $55.2 \pm 0.4$ | $57.0 \pm 0.2$ | $28.4 \pm 0.0$ | $57.8 \pm 0.0$ |

Table 5: Intersection of samples obtained with different values of $b$. We see the intersection of samples selected with different number of intersections comparable to highest possible (different seeds) in FastText, far higher compared to chance intersection. This indicates similar samples are selected regardless of sample size. NaiveBayes does not show clear trends but occasionally the queried percentage drops significantly when increasing iterations, occasionally it remains unaffected.

| Dsets | Chance | FTZ Ent-Ent | FTZ Ent-LC | FTZ Ent-DelEnt | FTZ DelEnt-DelLC | FTZ DelEnt-DelEnt |
|---|---|---|---|---|---|---|
| SGN | $9.4 \pm 0.0$ | $84.6 \pm 0.2$ | $83.1 \pm 0.3$ | $81.7 \pm 0.1$ | $82.6 \pm 0.1$ | $84.2 \pm 0.1$ |
| DBP | $9.3 \pm 0.0$ | $85.7 \pm 0.2$ | $85.5 \pm 0.3$ | $83.3 \pm 0.1$ | $83.0 \pm 0.4$ | $83.2 \pm 0.2$ |
| YHA | $19.0 \pm 0.0$ | $79.0 \pm 0.0$ | $71.6 \pm 0.2$ | $76.3 \pm 0.1$ | $69.6 \pm 0.7$ | $75.6 \pm 3.9$ |
| YRP | $9.3 \pm 0.0$ | $58.4 \pm 0.6$ | $59.0 \pm 0.3$ | $59.0 \pm 0.6$ | $61.6 \pm 0.7$ | $62.1 \pm 0.1$ |
| YRF | $19.0 \pm 0.0$ | $77.8 \pm 0.2$ | $66.6 \pm 0.3$ | $75.8 \pm 0.1$ | $65.4 \pm 0.3$ | $80.1 \pm 0.2$ |
| AGN | $19.1 \pm 0.0$ | $78.3 \pm 0.1$ | $77.3 \pm 0.1$ | $77.1 \pm 0.3$ | $78.2 \pm 0.4$ | $79.0 \pm 0.3$ |
| AMZP | $9.5 \pm 0.0$ | $63.5 \pm 0.2$ | $63.5 \pm 0.3$ | $66.1 \pm 0.4$ | $70.0 \pm 0.1$ | $70.0 \pm 0.1$ |
| AMZF | $19.0 \pm 0.0$ | $70.3 \pm 0.1$ | $64.3 \pm 0.2$ | $69.6 \pm 0.1$ | $65.6 \pm 0.2$ | $72.6 \pm 0.2$ |

Table 6: Intersection of query strategies across acquisition functions. We observe that the % intersection among samples in the Ent-LC is comparable to those Ent-Ent. Similarly, the Ent-DelEnt (entropy with deletion) is comparable to both DelEnt-DelLC and DelEnt-DelEnt showing robustness of FastText to query functions (beyond minor variation). DelEnt-DelEnt obtains similar intersections as compared to Ent-Ent, showing the robustness of the acquired samples to deletion.

3 and 5 present overlaps from different initial sets while 4 and 6 from same initial sets. We note from column 4 and 6 that due to the stochasticity of training in FTZ, we expect non-identical final sets even with same initial samples as well. The results demonstrate that samples obtained using FastText are largely initialization independent (low variation between columns 3 and 4) consistently across datasets while the samples obtained with Naive Bayes can be vastly different showing relatively heavy dependence on the initial seed. This indicates the relative stability of train set obtained with the posterior uncertainty of the actively trained FTZ as an acquisition function.

#### 3.2.2 Query size

Since the sampled data is sequentially constructed by training models on previously sampled data, large query sizes were expected to impact samples collected by uncertainty sampling and the performance thereof (Hoi et al., 2006). We experiment with various query sizes - (0.25%, 0.5%, 1%) for DBP, SGN, YRP and AMZP and (0.5%, 1%, 2%) for the rest corresponding to 9, 19 and 39 iterations. Figure 1 shows that FastText (top row) has very stable performance across sample sizes while MNB (bottom row) show more erratic performance. Table 5 presents the intersection of samples obtained with different query sizes across multiple runs. We observe a high overlap of the acquired samples across different query sizes indicating that the performance is independent of the query size (compare column 3 to column 4 where the size is held constant) while MNB results in lower overlap with more erratic behavior due to change in the query size (compare column 5 compared to column 6).

#### 3.2.3 Query strategy

We now investigate the impact of various query strategies using FastText by evaluating and comparing the correlation between the respective actively selected sample sets.

**Acquisition Functions:** We compare four uncertainty query strategies: Least Confidence (LC) and Entropy (Ent), with and without deletion of least uncertain samples from the training set. Deletion of least uncertain samples reduces the dependence on the initial randomly selected set. The results are presented in Table 6. We present five of

| Dsets | Chance | FTZ-FTZ Ent | FTZ-5F TZ Ent | 5FTZ-5FTZ Ent-LC | 5FTZ-5FTZ Ent-Ent |
|---|---|---|---|---|---|
| SGN | $9.4 \pm 0.0$ | $84.6 \pm 0.2$ | $86.3 \pm 0.2$ | $85.4 \pm 0.4$ | $85.8 \pm 0.0$ |
| DBP | $9.3 \pm 0.0$ | $85.7 \pm 0.2$ | $86.6 \pm 0.3$ | $86.78 \pm 0.1$ | $87.8 \pm 0.2$ |
| YRP | $9.3 \pm 0.0$ | $58.4 \pm 0.6$ | $58.1 \pm 0.7$ | $58.3 \pm 0.3$ | $58.2 \pm 0.2$ |
| YRF | $19.0 \pm 0.0$ | $77.8 \pm 0.2$ | $79.0 \pm 0.3$ | $68.5 \pm 1.1$ | $77.6 \pm 0.3$ |
| AGN | $19.1 \pm 0.0$ | $78.3 \pm 0.1$ | $79.0 \pm 0.2$ | $79.1 \pm 0.2$ | $77.9 \pm 0.2$ |

Table 7: Intersection of query strategies across single and ensemble of 5FTZ models. We observe that the % intersection of samples selected by ensembles and single models is comparable to intersection among either. The 5-model committee does not seem to add any additional value over selection by a single model.

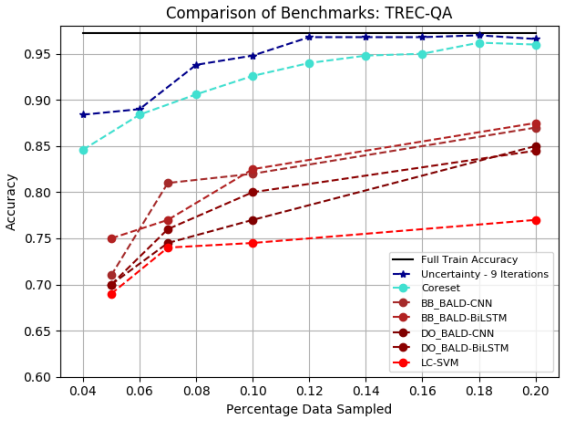


Figure 2: Active text classification: Comparison with K-Center Coreset, BALD and SVM algorithms. Accuracy is plotted against percentage data sampled. We reach full-train accuracy using 12% of the data, compared to BALD which requires 50% data and perform significantly worse in terms of accuracy. We also outperform K-center greedy Coreset at all sampling percentages without utilizing additional diversity-based augmentation.

the ten possible combinations and again observe the high degree of overlap in the collected samples. It can be concluded that the approach is fairly robust to these variations in the query strategy.

**Ensembles versus Single Models:** A similar experiment was conducted to investigate the overlap between a single FTZ model and a probabilistic committee of models (5-model ensemble with FTZ (Lakshminarayanan et al., 2017)) to identify comparative advantages of using ensemble methods. The results are presented in Table 7 showing little to no difference in sample overlaps. [5] We conclude that more expensive sampling strategies commonly used, like ensembling, may offer little benefit compared to using a single FTZ model with posterior uncertainty as a query function.

The experiments in this section demonstrate that uncertainty based sampling using deep models like FTZ show no class bias or an undesirable feature bias (and favorable bias to class boundaries). There is also a high degree of robustness to algorithmic factors, especially query size, a surprisingly high degree of overlap in the resulting training samples and stable performances (classification accuracy). Additionally, all uncertainty query strategies perform well, and expensive sampling strategies like ensembling offer little benefit. We conclude that sampling biases demonstrated in active learning literature do hold well with traditional models, however, they do not seem to translate to deep models like FTZ using (posterior) uncertainty.

## 4 Application: Active Text Classification

Experimental results from the previous sections suggest that entropy function with a single FTZ model would be a good baseline for active text classification. We compare our baseline with the latest work in deep active learning for text classification - BALD (Siddhant and Lipton, 2018) and with the recent diversity based Coreset query function (Sener and Savarese, 2018) which uses a costly K-center algorithm to build the query. Experiments are performed on TREC-QA for a fair comparison (used by (Siddhant and Lipton, 2018)). Table 8 shows that the results of our study generalize to small datasets like TREC-QA.

The results are shown in Figure 2 using the baseline with the query size of 2% of the full dataset (b=9 queries). Note that uncertainty sampling converges to full accuracy using just 12% of the data, whereas (Siddhant and Lipton, 2018) required 50% of the data. There is also a remarkable accuracy improvement over (Siddhant and Lipton, 2018) which can be largely attributed to the models used (FastText versus 1-layer CNN/BiLSTM). Also, uncertainty sampling outperforms diversity-based augmentations like Coreset Sampling (Sener and Savarese, 2018) before convergence. Thus, we establish a new state-of-the-art baseline for further research in deep active text classification.

## 5 Application: Training of Large Models

The cost and time needed to get and label vast amounts of data to train large DNNs is a serious

[5] The ensembles were too costly to run on larger datasets, so the results for YHA, AMZP and AMZF could not be obtained.

| Dsets | Chance | FTZ-Ent-Ent | FTZ Ent-LC | SV Chce% | SV Com% |
|---|---|---|---|---|---|
| TQA | $15.1 \pm 0.0$ | $\mathbf{59.7 \pm 0.5}$ | $\mathbf{56.3 \pm 1.4}$ | $18.7 \pm 6.1$ | $\mathbf{79.0 \pm 3.6}$ |

Table 8: Results of sample selection from previous investigations on small datasets (Trec-QA).

| Model | AGN | DBP | SGN | YRF | YRP | YHA | AMZP | AMZF |
|---|---|---|---|---|---|---|---|---|
| VDCNN (Conneau et al., 2017) | 91.3 | 98.7 | 96.8 | 64.7 | 95.7 | 73.4 | 95.7 | 63.0 |
| DPCNN (Johnson and Zhang, 2017) | 93.1 | **99.1** | **98.1** | **69.4** | **97.3** | **76.1** | **96.7** | **65.2** |
| WC-Reg (Qiao et al., 2018) | 92.8 | 98.9 | **97.6** | 64.9 | 96.4 | 73.7 | 95.1 | 60.9 |
| DC+MFA (Wang et al., 2018) | 93.6 | **99.2** | - | 66.0 | 96.5 | - | - | 63.0 |
| DRNN (Wang, 2018) | **94.5** | **99.2** | - | **69.1** | **97.3** | 70.3 | **96.5** | **64.4** |
| ULMFiT (Howard and Ruder, 2018) | **95.0** | **99.2** | - | **70.0** | **97.8** | - | - | - |
| EXAM (Du et al., 2019) | 93.0 | 99.0 | - | - | - | 74.8 | 95.5 | 61.9 |
| Ours: ULMFiT (Small data) | 93.7 (20) | **99.2 (10)** | 97.0 (10) | 67.6 (20) | **97.1 (10)** | 74.3 (20) | **96.1 (10)** | **64.1 (20)** |
| Ours: ULMFiT (Tiny data) | 91.7 (8) | 98.6 (2.3) | **97.4 (6.3)** | 66.3 (8) | 96.7 (4) | 73.3 (8) | 95.8 (4) | 62.9 (8) |

Table 9: Comparison of accuracies with state-of-the-art approaches (earliest-latest) for text classification (%dataset in brackets). We are competitive with state-of-the-art models while using 5x-40x compressed datasets.

| ULMFiT | AGN | DBP | YRP | YRF |
|---|---|---|---|---|
| Full | **95.0** | **99.2** | **97.8** | **70.0** |
| Ours-Small | 93.7 (20) | **99.2 (10)** | **97.1 (10)** | 67.6 (20) |
| Ours-Tiny | 91.7 (8) | **98.6 (2.3)** | 96.7 (4) | 66.3(8) |

Table 10: ULMFiT: Resulting sample $\hat{\mathcal{S}}_b$ compared to reported accuracies in (Howard and Ruder, 2018) (%dataset in brackets). We observe that using our cheaply obtained compressed datasets, we can achieve similar accuracies with 25x-200x speedup (5x less epochs, 5x-40x less data). Transferability to other models is evidence of the generalizability of the subset collected using FTZ to other deep models.

impediment to creating new and/or better models. Our study suggests that the training samples collected with uncertainty sampling (entropy) on a single model FTZ may provide a good representation (surrogate) for the entire dataset. Buoyed by this, we investigate if we can speedup training of ULMFiT (Howard and Ruder, 2018) using the surrogate dataset. We show these results in Table 10. We achieve 25x-200x speedup[6] (5x fewer epochs, 5x-40x smaller training size). We also benchmark the performance against the state-of-the-art on text classification as shown in Table 9. We conclude that we can significantly compress the training datasets and speedup classifier training time with little tradeoff in accuracy.

**Implementation Details:** We use the official github repository for ULMFiT[7], use default hyperparameters and train on one NVIDIA Tesla V100 16GB GPU. Further details are provided in supplementary material.

[6]The cost of acquiring the training data using FTZ-Ent is negligible in comparison.

[7]https://github.com/fastai/fastai/tree/master/courses/dl2/imdb_scripts

## 6 Related Work

We now expand on the brief literature review in Section 1 to better contextualize our work. We divide the past works into (i) Traditional Models and (ii) Deep Models.

**Sampling Bias in Classical AL in NLP:** Active learning (AL) in text classification started with greedy uncertainty query strategy from a pool using decision trees (Lewis and Gale, 1994), which was shown to be effective and led to widespread adoption with classifiers like SVMs (Tong and Koller, 2001), Naive Bayes (Roy and McCallum, 2001) and KNN (Fujii et al., 1998). This strategy was also applied to other NLP tasks like parse selection (Baldridge and Osborne, 2004), sequence labeling (Settles and Craven, 2008) and information extraction (Thompson and Mooney, 1999). These early papers popularized two greedy uncertainty query methods: Least Confident and Entropy.

Issues of lack of diversity (large redudancy in sampling) (Zhang and Oles, 2000) and lack of robustness (high variance in sample quality)(Krogh and Vedelsby, 1994) guided subsequent efforts. The two most popular directions were: (i) augmenting uncertainty with diversity measures (Hoi et al., 2006; Brinker, 2003; Tang et al., 2002) and (ii) using query-by-committee (McCallum and Nigam, 1998; Liere and Tadepalli, 1997). For a comprehensive survey of classical AL methods for NLP, please refer to (Settles, 2009).

**Sampling Bias in Deep AL:** Deep active learning approach adapt the above framework to the

training of DNNs on large data. Two main query strategies are used: (i) ensemble based greedy uncertainty, which represents a probabilistic query-by-committee paradigm (Gal et al., 2017; Beluch et al., 2018), and (ii) diversity based measures (Sener and Savarese, 2018; Ducoffe and Precioso, 2018). Papers proposing diversity based approaches find that greedy uncertainty based sampling (using ensemble and single model) perform significantly worse than random (See Figures 4 and 2 respectively in (Sener and Savarese, 2018; Ducoffe and Precioso, 2018)). They attribute the poor performance to redundant, highly correlated sampling selected using uncertainty based methods and justify the need for prohibitively expensive diversity-based approaches (Refer section 2 of (Sener and Savarese, 2018) for details on the expensiveness of various diversity sampling methods). However, K-center greedy coreset sampling scales poorly: we were only able to use it on TREC-QA (a small dataset). On the other hand, ensemble-based greedy uncertainty methods find that probabilistic averaging from a committee (Gal et al., 2017; Beluch et al., 2018) performs better than single model as with on diversity based methods like coreset(Gissin and Shalev-Shwartz, 2019; Beluch et al., 2018). Current approaches in text classification literature mostly adopt the ensemble based greedy uncertainty framework (Siddhant and Lipton, 2018; Lowell et al., 2018; Zhang et al., 2017).

However, our work demonstrates the problems of sampling bias and efficiency may not translate from shallow to deep approaches. Recent evidence from image domain (Gissin and Shalev-Shwartz, 2019) demonstrates atleast a subset of our findings generalize to other DNNs (class bias and query functions). Uncertainty sampling using a deep model like FTZ demonstrates surprisingly good sampling properties without using ensembles or bayesian methods. Ensembles do not seem to significantly affect sampling. Whether this behavior generalizes to other deep models and tasks is yet to be seen.

**Other Related Works:** An interesting set of papers (Soudry et al., 2018; Xu et al., 2018) show that deep neural networks trained with SGD converge to the maximum margin solution in the linearly separable case. Several works investigate the possibility that deep networks give high importance to a subset of the training dataset (Toneva et al., 2019; Vodrahalli et al., 2018; Birodkar et al., 2019), resembling supports in support vector machines. In our experiments, we find that active learning with uncertainty sampling with deep models like FTZ has a (surprisingly) large overlap with the support vectors of an SVM. Thus, it seems to have a inductive bias for class boundaries, similar to the above works. Whether this property generalizes to other deep models is yet to be seen.

## 7 Conclusion

We conducted a large empirical study of sampling bias and efficiency, along with algorithmic factors which impacting active text classification. We conclude that uncertainty sampling with deep models like FastText.zip exhibits negligible class bias, seems to be favorably biased to sampling data points near class boundaries, is robust to various algorithmic factors and expensive sampling strategies like ensembling offer little benefit. Also, we find a surprisingly large overlap of actively acquired points with supports of a SVM. We additionally show that uncertainty sampling can be effectively used to bootstrap the training of large DNN models by generating compact surrogate datasets (5x-40x compression). Finally, FTZ-Ent provides a strong baseline for deep active text classification, outperforming previous results by a margin of 4x less data.

The current work opens up several directions for future investigations. To list a few: (a) a deeper look into the nature of sampled data - their distribution in the feature space, as well as their importance for the task at hand; (b) the creation of surrogate datasets for a variety of applications, including hyperparameter and architecture search, etc; (c) an extension to other deep models (beyond FTZ) and beyond classification models; and, (d) an extension to semi-supervised, online and continual learning.

**Acknowledgements**: We thank Prof. Vineeth Balasubramium, IIT Hyderabad, India for the many helpful suggestions and discussions.

## References


Jason Baldridge and Miles Osborne. 2004. Active learning and the total cost of annotation. In *EMNLP*.

William H. Beluch, Tim Genewein, Andreas Nurnberger, and Jan M. Kohler. 2018. The power of ensembles for active learning in image classification. In *CVPR*.

Vighnesh Birodkar, Hossein Mobahi, and Samy Bengio. 2019. Semantic redundancies in image-classification datasets: The 10% you don't need. *arXiv preprint arXiv:1901.11409*.

Klaus Brinker. 2003. Incorporating diversity in active learning with support vector machines. In *ICML*.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2017. Very deep convolutional networks for text classification. In *EACL*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Cunxiao Du, Zhaozheng Chin, Fuli Feng, Lei Zhu, Tian Gan, and Liqiang Nie. 2019. Explicit interaction model towards text classification. In *AAAI*.

Melanie Ducoffe and Frederic Precioso. 2018. Adversarial active learning for deep networks: a margin based approach. In *ICML*.

Sandra Ebert, Mario Fritz, and Bernt Schiele. 2012. Ralf: A reinforced active learning formulation for object class recognition. In *CVPR*.

Atsushi Fujii, Takenobu Tokunaga, Kentaro Inui, and Hozumi Tanaka. 1998. Selective sampling for example-based word sense disambiguation. In *Computational Linguistics*.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. 2017. Deep bayesian active learning with image data. In *ICML*.

Daniel Gissin and Shai Shalev-Shwartz. 2019. Discriminative active learning. *ArXiv preprint arxiv:1907.06347v1*.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. On calibration of modern neural networks. In *ICML*.

Steven C. H. Hoi, Rong Jin, and Michael R. Lyu. 2006. Large-scale text categorization by batch mode active learning. In *WWW*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *ACL*.

Rie Johnson and Tong Zhang. 2017. Deep pyramid convolutional neural networks for text categorization. In *ACL*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. 2016. Fasttext. zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *EACL*.

Anders Krogh and Jesper Vedelsby. 1994. Neural network ensembles, cross validation and active learning. In *NeurIPS*.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. In *NeurIPS*.

David D. Lewis and William A. Gale. 1994. A sequential algorithm for training text classifiers. In *SIGIR*.

Ray Liere and Prasad Tadepalli. 1997. Active learning with committees for text categorization. In *AAAI*.

David Lowell, Zachary C Lipton, and Byron C Wallace. 2018. How transferable are the datasets collected by active learners? *arXiv preprint arXiv:1807.04801*.

Andrew McCallum and Kamal Nigam. 1998. Employing em and pool-based active learning for text classification. In *ICML*.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. In *JMLR*.

Chao Qiao, Bo Huang, Guocheng Niu, Daren Li, Daxiang Dong, Wei He, Dianhai Yu, and Hua Wu. 2018. A new method of region embedding for text classification. In *ICLR*.

Nicholas Roy and Andrew McCallum. 2001. Toward optimal active learning through sampling estimation of error reduction. In *ICML*.

Ozan Sener and Silvio Savarese. 2018. Active learning for convolutional neural networks: A core-set approach. In *ICLR*.

Burr Settles. 2009. Active learning literature survey. Technical report, University of Wisconsin-Madison.

Burr Settles and Mark Craven. 2008. An analysis of active learning strategies for sequence labeling tasks. In *EMNLP*.

Aditya Siddhant and Zachary C Lipton. 2018. Deep bayesian active learning for natural language processing: Results of a large-scale empirical study. In *EMNLP*.

Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. 2018. The implicit bias of gradient descent on separable data. *JMLR*.

Min Tang, Xiaoqiang Luo, and Salim Roukos. 2002. Active learning for statistical natural language parsing. In *ACL*.

Cynthia A Thompson and Raymond J Mooney. 1999. Active learning for natural language parsing and information extraction. In *ICML*.

Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J. Gordon. 2019. An empirical study of example forgetting during deep neural network learning. In *ICLR*.

Simon Tong and Daphne Koller. 2001. Support vector machine active learning with applications to text classification. *JMLR*.

Kailas Vodrahalli, Ke Li, and Jitendra Malik. 2018. Are all training examples created equal? an empirical study. *arXiv preprint arXiv:1811.12569*.

Baoxin Wang. 2018. Disconnected recurrent neural networks for text categorization. In *ACL*.

Shiyao Wang, Minlie Huang, and Zhidong Deng. 2018. Densely connected CNN with multi-scale feature attention for text classification. In *IJCAI*.

Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *ACL*.

Zeyi Wen, Jiashuai Shi, Qinbin Li, Bingsheng He, and Jian Chen. 2018. ThunderSVM: A fast SVM library on GPUs and CPUs. In *JMLR*.

Tengyu Xu, Yi Zhou, Kaiyi Ji, and Yingbin Liang. 2018. Convergence of SGD in learning ReLU models with separable data. *arXiv preprint arXiv:1806.04339*.

Tong Zhang and F Oles. 2000. The value of unlabeled data for classification problems. In *ICML*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NeurIPS*.

Ye Zhang, Matthew Lease, and Byron C Wallace. 2017. Active discriminative text representation learning. In *AAAI*.

Jingbo Zhu, Huizhen Wang, Tianshun Yao, and Benjamin K Tsou. 2008. Active learning with sampling by uncertainty and density for word sense disambiguation and text classification. In *ACL*.

# Don't Take the Easy Way Out: Ensemble Based Methods for Avoiding Known Dataset Biases

**Christopher Clark***, **Mark Yatskar**†, **Luke Zettlemoyer***

*Paul G. Allen School of CSE, University of Washington
{csquared, lsz}@cs.uw.edu

†Allen Institute for Artificial Intelligence, Seattle WA
marky@allenai.org

## Abstract

State-of-the-art models often make use of superficial patterns in the data that do not generalize well to out-of-domain or adversarial settings. For example, textual entailment models often learn that particular key words imply entailment, irrespective of context, and visual question answering models learn to predict prototypical answers, without considering evidence in the image. In this paper, we show that if we have prior knowledge of such biases, we can train a model to be more robust to domain shift. Our method has two stages: we (1) train a naive model that makes predictions exclusively based on dataset biases, and (2) train a robust model as part of an ensemble with the naive one in order to encourage it to focus on other patterns in the data that are more likely to generalize. Experiments on five datasets with out-of-domain test sets show significantly improved robustness in all settings, including a 12 point gain on a changing priors visual question answering dataset and a 9 point gain on an adversarial question answering test set.

## 1 Introduction

While recent neural models have shown remarkable results, these achievements have been tempered by the observation that they are often exploiting dataset-specific patterns that do not generalize well to out-of-domain or adversarial settings. For example, entailment models trained on MNLI (Bowman et al., 2015) will guess an answer based solely on the presence of particular keywords (Gururangan et al., 2018) or whether sentences pairs contain the same words (McCoy et al., 2019), while QA models trained on SQuAD (Rajpurkar et al., 2016) tend to select text near question-words as answers, regardless of context (Jia and Liang, 2017).

We refer to these kinds of superficial patterns as bias. Models that rely on bias can perform well on in-domain data, but are brittle and easy to fool (e.g., SQuAD models are easily distracted by irrelevant sentences that contain many question words). Recent concern about dataset bias has led researchers to re-examine many popular datasets, resulting in the discovery of a wide variety of biases (Agrawal et al., 2018; Anand et al., 2018; Min et al., 2019; Schwartz et al., 2017).

In this paper, we build on these works by showing that, once a dataset bias has been identified, we can improve the out-of-domain performance of models by preventing them from making use of that bias. To do this, we use the fact that these biases can often be explicitly modelled with simple, constrained baseline methods to factor them out of a final model through ensemble-based training.

Our method has two stages. First, we build a bias-only model designed to capture a naive solution that performs well on the training data, but generalizes poorly to out-of-domain settings. Next, we train a second model in an ensemble with the pre-trained bias-only model, which incentivizes the second model to learn an alternative strategy, and use the second model alone on the test set. We explore several different ensembling methods, building on product-of-expert style approaches (Hinton, 2002; Smith et al., 2005). Figure 1 shows an example of applying this procedure to prevent a visual question answering (VQA) model from guessing answers because they are typical for the question, a flaw observed in VQA models (Goyal et al., 2018; Agrawal et al., 2018).

We evaluate our approach on a diverse set of tasks, all of which require models to overcome a challenging domain-shift between the train and test data. First, we build a set of synthetic datasets that contain manually constructed biases by adding artificial features to MNLI. We then consider three challenge datasets proposed by prior work (Agrawal et al., 2018; McCoy et al.,



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4069–4082, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

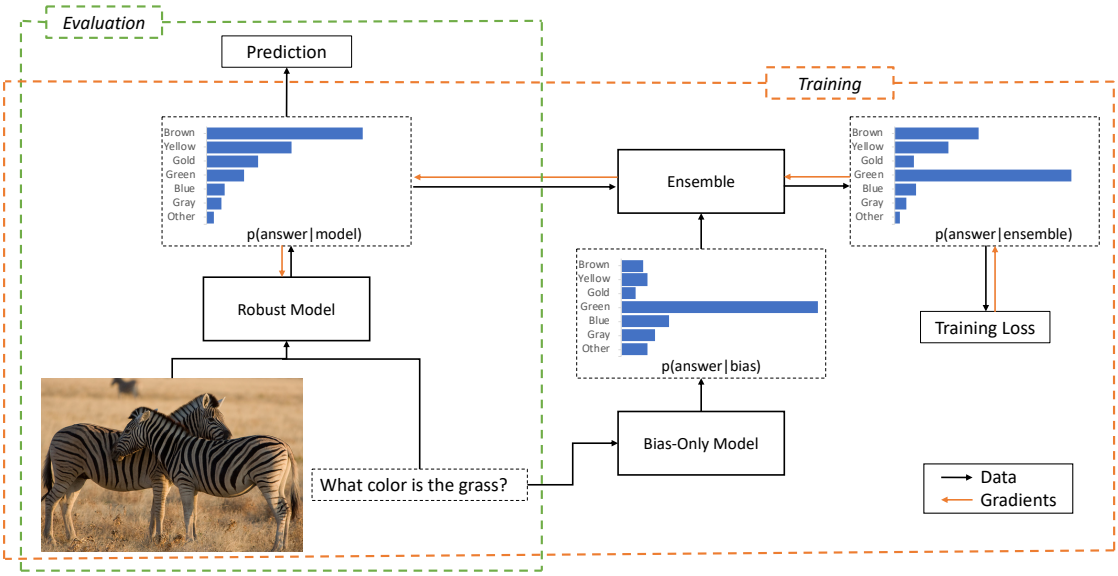


Figure 1: An example of applying our method to a Visual Question Answering (VQA) task. We assume predicting green for the given question is almost always correct on the training data. To prevent a model from learning this bias, we first train a bias-only model that only uses the question as input, and then train a robust model in an ensemble with the bias-only model. Since the bias-only model will have already captured the target pattern, the robust model has no incentive to learn it, and thus does better on test data where the pattern is not reliable.

2019; Jia and Liang, 2017), which were designed to break models that adopt superficial strategies on well known textual entailment (Bowman et al., 2015), reading comprehension (Rajpurkar et al., 2016), and VQA (Antol et al., 2015) datasets.

We additionally construct a new QA challenge dataset, TriviaQA-CP (for TriviaQA changing priors). This dataset was built by holding out questions from TriviaQA (Joshi et al., 2017) that ask about particular kinds of entities from the train set, and evaluating on those questions in the dev set, in order to challenge models to generalize between different types of questions.

We are able to improve out-of-domain performance in all settings, including a 6 and 9 point gain on the two QA datasets. On the VQA challenge set, we achieve a 12 point gain, compared to a 3 point gain from prior work. In general, we find using an ensembling method that can dynamically choose when to trust the bias-only model is the most effective, and we present synthetic experiments and qualitative analysis to illustrate the advantages of that approach. We release our datasets and code to facilitate future work.[1]

## 2 Related Work

Researchers have raised concerns about bias in many datasets. For example, many joint natural language processing and vision datasets can be partially solved by models that ignore the vision aspect of the task (Jabri et al., 2016; Zhang et al., 2016; Anand et al., 2018; Caglayan et al., 2019). Some questions in recent multi-hop QA datasets (Yang et al., 2018; Welbl et al., 2018) can be solved by single-hop models (Chen and Durrett, 2019; Min et al., 2019). Additional examples include story completion (Schwartz et al., 2017) and multiple choice questions (Clark et al., 2016, 2018). Recognizing that bias is a concern in diverse domains, our work is the first to perform an evaluation across multiple datasets spanning language and vision.

Recent dataset construction protocols have tried to avoid certain kinds of bias. For example, both CoQA (Reddy et al., 2019) and QuAC (Choi et al., 2018) take steps to prevent annotators from using words that occur in the context passage, VQA 2.0 (Goyal et al., 2018) selects examples to limit the effectiveness of question-only models, and others have filtered examples solvable by simple baselines (Yang et al., 2018; Zhang et al., 2018b; Clark et al., 2018; Zellers et al., 2018). While reducing bias is important, developing ways to prevent models from using known biases will allow us to continue to leverage existing datasets, and update our methods as our understanding of what biases we want to avoid evolve.

[1] github.com/chrisc36/debias

Recent work has focused on biases that come from ignoring parts of the input (e.g., guessing the answer to a question before seeing the evidence). Solutions include generative objectives to force models to understand all the input (Lewis and Fan, 2019), carefully designed model architecture (Agrawal et al., 2018; Zhang et al., 2016), or adversarial removal of class-indicative features from model's internal representations (Ramakrishnan et al., 2018; Zhang et al., 2018a; Belinkov et al., 2019; Grand and Belinkov, 2019). In contrast, we consider biases beyond partial-input cases (Feng et al., 2019), and show our method is superior on VQA-CP. Concurrently, He et al. (2019) also suggested using a product-of-experts ensemble to train unbiased models, but we consider a wider variety of ensembling approaches and test on additional domains.

A related task is preventing models from using particular problematic dataset features, which is often studied from the perspective of fairness (Zhao et al., 2017; Burns et al., 2018). A popular approach is to use an adversary to remove information about a target feature, often gender or ethnicity, from a model's internal representations (Edwards and Storkey, 2016; Wang et al., 2018; Kim et al., 2019). In contrast, the biases we consider are related to features that are essential to the overall task, so they cannot simply be ignored.

Evaluating models on out-of-domain examples built by applying minor perturbations to existing examples has also been the subject of recent study (Szegedy et al., 2014; Belinkov and Bisk, 2018; Carlini and Wagner, 2018; Glockner et al., 2018). The domain shifts we consider involve larger changes to the input distribution, built to uncover higher-level flaws in existing models.

# 3 Methods

This section describes the two stages of our method, (1) building a bias-only model and (2) using it to train a robust model through ensembling.

## 3.1 Training a Bias-Only Model

The goal of the first stage is to build a model that performs well on training data, but is likely to perform very poorly on the out-of-domain test set. Since we assume we do not have access to examples from the test set, we must apply *a-priori* knowledge to meet this goal.

The most straightforward approach is to identify a set of features that are correlated with the class label during training, but are known to be uncorrelated or anticorrelated with the label on the test set, and then train a classifier on those features.[2] For example, our VQA-CP (Agrawal et al., 2018) bias-only model (see Section 5.2) uses the question type as input, because the correlations between question types and answers is very different in the train set than the test set (e.g., 2 is a common answer to "How many..." questions on the train set, but is rare for such questions on the test set).

However, a benefit of our method is that the bias can be modelled using any kind of predictor, giving us a way to capture more complex intuitions. For example, on SQuAD our bias-only model operates on a view of the input built from TF-IDF scores (see Section 5.4), and on our changing prior TriviaQA dataset our bias-only model makes use of a pre-trained named entity recognition (NER) tagger (see Section 5.5).

## 3.2 Training a Robust Model

This stage trains a robust model that avoids using the method learned by the bias-only model.

### 3.2.1 Problem Definition

We assume $n$ training examples $\langle x_1, x_2, \ldots, x_n \rangle$, each of which has an integer label $y_i$, where $y_i \in \{1, 2, \ldots, C\}$ and $C$ is the number of classes. We additionally assume a pre-trained bias-only predictor, $h$, where $h(x_i) = b_i = \langle b_{i1}, b_{i2}, ..b_{iC} \rangle$ and $b_{ij}$ is the bias-only model's predicted probability of class $j$ for example $i$. Finally we have a second predictor function, $f$, with parameters $\theta$, where $f(x_i, \theta) = p_i$ and $p_i$ is a similar probability distribution over the classes. Our goal is to construct a training objective to optimize $\theta$ so that $f$ will learn to select the correct class without using the strategy captured by the bias-only model.

### 3.2.2 General Approach

We train an ensemble of $h$ and $f$. In particular, for each example, a new class distribution, $\hat{p_i}$, is computed by combining $p_i$ and $b_i$. During training, the loss is computed using $\hat{p_i}$ and the gradients are backproped through $f$. During evaluation $f$ is used alone. We propose several different ensembling methods.

[2] Since the bias-only model is trained on the same trainset as the robust model care should also be taken to minimize overfitting, although the bias-only model is typically simple enough that this is not an issue.

#### 3.2.3 Bias Product

Our simplest ensemble is a product of experts (Hinton, 2002):

$$\hat{p_i} = softmax(\log(p_i) + \log(b_i))$$

Equivalently, $\hat{p_i} \propto p_i \circ b_i$, where $\circ$ is element-wise multiplication.

**Probabilistic Justification**: For a given example, $x$, let $x^b$ be the bias of the example. That is, it is the features we will use in our bias-only model. Let $x^{-b}$ be a view of the example that captures all information about that example except the bias. Assume that $x^{-b}$ and $x^b$ are conditionally independent given the label, $c$. Then to compute $p(c|x)$ we have:

$$\begin{aligned} p(c|x) &= p(c|x^b, x^{-b}) && (1)\\ &\propto p(c|x^{-b})p(x^b|c, x^{-b}) && (2)\\ &= p(c|x^{-b})p(x^b|c) && (3)\\ &= p(c|x^{-b})\frac{p(c|x^b)p(x^b)}{p(c)} && (4)\\ &\propto p(c|x^{-b})\frac{p(c|x^b)}{p(c)} && (5) \end{aligned}$$

Where 2 is from applying Bayes Rule while conditioning on $x^{-b}$, 3 follows from the conditional independence assumption, and 4 applies Bayes Rule a second time to $p(x^b|c)$.

We cannot directly model $p(c|x^{-b})$ because it is usually not possible to create a view of the data that excludes the bias. Instead, with the goal of encouraging the model to fall into the role of computing $p(c|x^{-b})$, we compute $p(c|x^b)/p(c)$ using the bias-only model, and train the product of the two models to compute $p(c|x)$.

In practice, we ignore the $p(c)$ factor because, on our datasets, either the classes are uniformly distributed (MNLI), the bias-only model cannot easily capture a class prior since it is using a pointer network (QA), or because we want to remove class priors from model anyway (VQA).

#### 3.2.4 Learned-Mixin

The assumption of conditional independence (Equation 3) will often be too strong. For example, in some cases the robust model might be able to predict the bias-only model will be unreliable for certain kinds of training examples. We find that this can cause the robust model to selectively adjust its behavior in order to compensate for the inaccuracy of the bias-only model, leading to errors in the out-of-domain setting (see Section 5.1).

Instead we allow the model to explicitly determine how much to trust the bias given the input:

$$\hat{p_i} = softmax(\log(p_i) + g(x_i)\log(b_i))$$

where $g$ is a learned function. We compute $g$ as $softplus(w \cdot h_i)$ where $w$ is a learned vector, $h_i$ is the last hidden layer of the model for example $x_i$, and the $softplus(x) = \log(1+e^x)$ function is used to prevent the model reversing the bias by multiplying it by a negative weight. $w$ is trained with the rest of the model parameters. This reduces to bias product when $g(x_i) = 1$.

A difficulty with this method is that the model could learn to integrate the bias into $p_i$ and set $g(x_i) = 0$. We find this does sometimes occurs in practice, and our next method alleviates this challenge.

#### 3.2.5 Learned-Mixin +H

To prevent the learned-mixin ensemble from ignoring $b_i$, we add an entropy penalty to the loss:

$$R = wH(softmax(g(x_i)\log(b_i)))$$

Where $H(z) = -\sum_j z_j \log(z_j)$ is the entropy and $w$ is a hyperparameter. Penalizing the entropy encourages the bias component to be non-uniform, and thus have a greater impact on the ensemble.

## 4 Evaluation Methodology

We evaluate our methods on several datasets that have out-of-domain test sets. Some of these tasks, such as HANS (McCoy et al., 2019) or Adversarial SQuAD (Jia and Liang, 2017), can be solved easily by generating additional training examples similar to the ones in the test set (e.g., Wang and Bansal (2018)). We, instead, demonstrate that it is possible to improve performance on these tasks by exploiting knowledge of general, biased strategies the model is likely to adopt.

Our evaluation setup consists of a training set, an out-of-domain test set, a bias-only model, and a main model. To run an evaluation we train the bias-only model on the train set, train the main model on the train set while employing one of the methods in Section 3, and evaluate the main model on the out-of-domain test set. We also report performance on the in-domain test set, when available. We use models that are known to work well for their respective tasks for the main model, and

| Task | Dataset | Domain Shift | Bias-Only Model | Main Model |
|---|---|---|---|---|
| NLI | Synthetic MNLI | Synthetic indicator features are randomized | Indicator features | Co-Attention |
| VQA | VQA-CP v2.0 | Correlations between question-types and answers are altered | Question-type | BottomUpTopDown |
| NLI | HANS | Sentence pairs always contain the same words | Shared word features | BERT & Co-Attention |
| QA | Adv. SQuAD | Distractor sentences are added to the context | TF-IDF sentence selector | Modified BiDAF |
| QA | TriviaQA-CP | Questions ask about different kinds of entities | NER answer detector | Modified BiDAF |

Table 1: Summary of the evaluations we perform, Domain Shift refers to what changes between the train and test data, and Bias-Only Model specifies how the bias model we use was constructed. See the main text for details.

do not further tune their hyperparameters or perform early stopping.

We consider two extractive QA datasets, which we treat as a joint classification task where the model must select the start and end answer token (Wang and Jiang, 2017). For these datasets, we build independent bias-only models for selecting the start and end token, and separately ensemble those biases with the classifier's start token and end token output distributions. We apply a ReLU layer to the question and passage embeddings, followed by max-pooling, to construct a hidden state for computing the learned-mixin weights.

We compare our methods to a reweighting baseline described below, and to training the main model without any modifications. On VQA we also compare to the adversarial methods from Ramakrishnan et al. (2018) and Grand and Belinkov (2019). The other biases we consider are not based on observing only part of the input, so these adversarial methods cannot be directly applied.

### 4.1 Reweight Baseline

As a non-ensemble baseline, we train the main model on a weighted version of the data, where the weight of example $x_i$ is $1 - b_{iy_i}$ (i.e., we weigh examples by one minus the probability the bias-only model assigns the correct label). This encourages the main model to focus on examples the bias-only model gets wrong.

### 4.2 Hyperparameters

One of our methods (Learned-Mixin +H) requires hyperparameter tuning. However hyperparameter tuning is challenging in our setup since our assumption is that we have no access to out-of-domain test examples during training. A plausible option would be to tune hyperparameters on a dev set that exhibits a related, but not identical, domain shift to the test set, but unfortunately none of our datasets have such dev sets. Instead we follow prior work (Grand and Belinkov, 2019; Ramakrishnan et al., 2018) and perform model selection on the test set. Although this presents an important caveat to the results of this method, we think it is still of interest to observe that the entropy regularizer can be very impactful. Future work may be able to either construct suitable development sets, or propose other hyperparameter-tuning methods to relieve this issue. The hyperparameters selected are shown in Appendix A.

## 5 Experiments

We provide experiments on five different domains, summarized in Table 1, each of which requires models to overcome a challenging domain-shift between train and test data. In the following sections we provide summaries of the datasets, main models and bias-only models, but leave low-level details to the appendix.

### 5.1 Synthetic Data

**Data**: We experiment with a synthetic dataset built by modifying MNLI (Bowman et al., 2015). In particular, we add a feature that is correlated with the class label to the train set, and build an out-of-domain test set by adding a randomized version of that feature to the MNLI matched dev set. We additionally construct an in-domain test set by modifying the matched dev set in the same way as was done in the train set. We build three variations of this dataset:

*Indicator*: Adds the token "0", "1", or "2" to the start of the hypothesis, such that 80% of the

time the token corresponds to the example's label (i.e., "0" if the class is "entailment", "1" if the class is contradiction, ect.). In the out-of-domain test set, the token is selected randomly.

*Excluder*: The same as Indicator, but with a 3% chance the added token corresponds to the example's label, meaning the token can usually be used to eliminate one of the three output classes.

*Dependent*: In the previous two settings, the added bias is independent of the example given the example's label. To simulate a case where this independence is broken, we experiment with adding an additional feature that is correlated with the bias feature, but is not treated as being part of the bias (i.e., it is not used by the bias-only model). In particular, 80% of the time a token is added to the start of the hypothesis that matches the label with 90% probability, and the "0" token is appended to the end of the hypothesis. The other 20% of the time a random token is prepended and "1" is appended.

**Bias-Only Model**: The bias-only model predicts the label using the first token of the hypothesis.

**Main Model**: We use a recurrent co-attention model, similar to ESIM (Chen et al., 2017). Details are given in Appendix B.

**Results**: Table 2 shows the results. All ensembling methods work well on the Indicator bias. The reweight method performs poorly on the Excluder bias, likely because the bias-only model assigns the correct class approximately 50% probability for almost all the training examples, making the weights mostly uniform. This illustrates a general weakness with reweighting methods: they require at least a small number of bias-free examples for the model to learn from.

The bias product method performs poorly on the Dependent bias. Inspection shows that, when the indicator is 1, the bias product model is anti-correlated with the bias. In particular, it assigns an average of 22.5% probability to the class indicated by the bias, where an unbaised model would assign an average of 33% since the bias is random. The root cause is that, if the indicator is 1, the model knows the bias is likely to be wrong, so it learns to subtract the value the bias-only model will produce from its own output in order to cancel out the bias-only model's effect on the ensemble's output.

The learned-mixin model does not suffer from this issue, and assigns the class indicated by the bias an average of 34.5% probability. Analysis shows that $g(x_i)$ is set to $0.00 \pm 0.0001$ when the indicator is turned off, and to $1.91 \pm 0.285$ otherwise, showing that the model learns to turn off the bias-only component of the ensemble as needed, thus avoiding this over-compensating issue. The entropy regularizer appears to be unnecessary on this dataset because $g(x_i)$ does not go to zero.

### 5.2 VQA-CP

**Data**: We evaluate on the VQA-CP v2 (Agrawal et al., 2018) dataset, which was constructed by re-splitting the VQA 2.0 (Goyal et al., 2018) train and validation sets into new train and test sets such that the correlations between question types and answers differs between each split. For example, "tennis" is the most common answer for questions that start with "What sport..." in the train set, whereas "skiing" is the most common answers for those questions in the test set. Models that choose answers because they are typical in the training data will perform poorly on this test set.

**Bias-Only Model**: VQA-CP comes with questions annotated with one of 65 question types, corresponding to the first few words of the question (e.g., "What color is"). The bias-only model uses this categorical label as input, and is trained on the same multi-label objective as the main model.

**Main Model**: We use a popular implementation[3] of the BottomUpToDown (Anderson et al., 2018) VQA model. This model uses a multi-label objective, so we apply our ensemble methods by treating each possible answer as a two-class classification problem.[4]

**Results:** Table 3 shows the results. The learned-mixin method was highly effective, boosting performance on VQA-CP by about 9 points, and the entropy regularizer can increase this by another 3 points, significantly surpassing

[3] github.com/hengyuan-hu/bottom-up-attention-vqa

[4] Since the bias sometimes assigns a zero probability to an answer, we additionally add $\sigma(\alpha)$ to the bias probabilities where $\alpha$ is learned parameter to allow the model to soften the bias as needed

| Debiasing Method | Indicator | | Excluder | | Dependent | |
|---|---|---|---|---|---|---|
| | Acc. | w/Bias | Acc. | w/Bias | Acc. | w/Bias |
| None | 69.36 | 86.49 | 68.06 | 83.56 | 63.23 | 87.90 |
| Reweight | 75.44 | 82.74 | 70.36 | 83.29 | 69.81 | 85.50 |
| Bias Product | 76.27 | 81.32 | 77.33 | 80.41 | 71.85 | 84.98 |
| Learned-Mixin | 76.29 | 81.35 | 77.80 | 78.86 | 75.75 | 77.70 |
| Learned-Mixin +H | 76.77 | 77.65 | 77.90 | 78.57 | 75.79 | 76.65 |
| Unbiased Training | 78.94 | 78.94 | 78.94 | 78.94 | 78.86 | 78.86 |

Table 2: Results on MNLI with different kinds of synthetic bias. The Acc columns show the accuracy on the out-of-domain test set, and the w/Bias columns show accuracy on the in-domain test. Unbiased Training is an upper bound constructed by training a model with the same randomized features that are used at test time.

| Debiasing Method | Acc. |
|---|---|
| None | 39.18 |
| Reweight | 40.06 |
| Bias Product | 39.93 |
| Learned-Mixin | 48.69 |
| Learned-Mixin +H | 52.05 |
| Ramakrishnan et al. (2018) | 41.17 |
| Grand and Belinkov (2019) | 42.33 |

Table 3: Results on the VQA-CP v2.0 test set.

| Debiasing Method | Co-Attention | | BERT | |
|---|---|---|---|---|
| | HANS | MNLI | HANS | MNLI |
| None | 50.58 | 78.73 | 62.40 | 84.24 |
| Reweight | 52.85 | 77.03 | 69.19 | 83.54 |
| Bias Product | 53.69 | 76.63 | 67.92 | 82.97 |
| Learned-Mixin | 51.65 | 78.05 | 64.00 | 84.29 |
| Learned-Mixin +H | 53.35 | 74.50 | 66.15 | 83.97 |

Table 4: Accuracy on the adversarial MNLI dataset, HANS, and the MNLI matched dev set.

prior work. For the learned-mixin ensemble, we find $g(x_i)$ is strongly correlated with the bias's expected accuracy[5], with a spearmanr correlation of 0.77 on the test data. Qualitative examples (Figure 2) further suggest the model increases $g(x_i)$ when it knows if can rely on the bias-only model.

### 5.3 HANS

**Data**: We evaluate on the HANS adversarial MNLI dataset (McCoy et al., 2019). This dataset was built by constructing templated examples of entailment and non-entailment, such that the hypothesis sentence only includes words that are also in the premise sentence. Naively trained models tend to classify all such examples as "entailment" because detecting the presence of many shared words is an effective tactic on MNLI.

**Bias-Only Model**: The bias-only model is a shallow linear classifier using the following features: (1) whether the hypothesis is a sub-sequence of the premise, (2) whether all words in the hypothesis appear in the premise, (3) the percent of words from the hypothesis that appear in the premise, (4) the average of the minimum distance between each premise word with each hypothesis word, measured using cosine distance with the fasttext (Mikolov et al., 2018) word vectors, and (5) the max of those same distances. We constrain the bias-only model to put the same amount of probability mass on the neutral and contradiction classes so it focuses on distinguishing entailment and non-entailment, and reweight the dataset so that the entailment and non-entailment examples have an equal total weight to prevent a class prior from being learned.

**Main Models**: We experiment with both the uncased BERT base model (Devlin et al., 2019), and the same recurrent model used for the synthetic data (see Appendix B). We use the default hyperparameters for BERT since they work well for MNLI.

**Results**: Table 4 shows the results. We show scores for individual heuristics used in HANS in Appendix C. For the recurrent method, both the bias product and learned-mixin +H methods result in about a three point gain. However, for the BERT model, the simpler reweight method is more effective. We noticed high variance in performance between runs in this setting, and speculate the ensemble methods might be compounding this instability by introducing additional complexity.

[5]Computed as $\sum_j s_{ij} b_{ij} / \sum_j b_{ij}$ where $s_{ij}$ is the score for class $j$ on example $i$

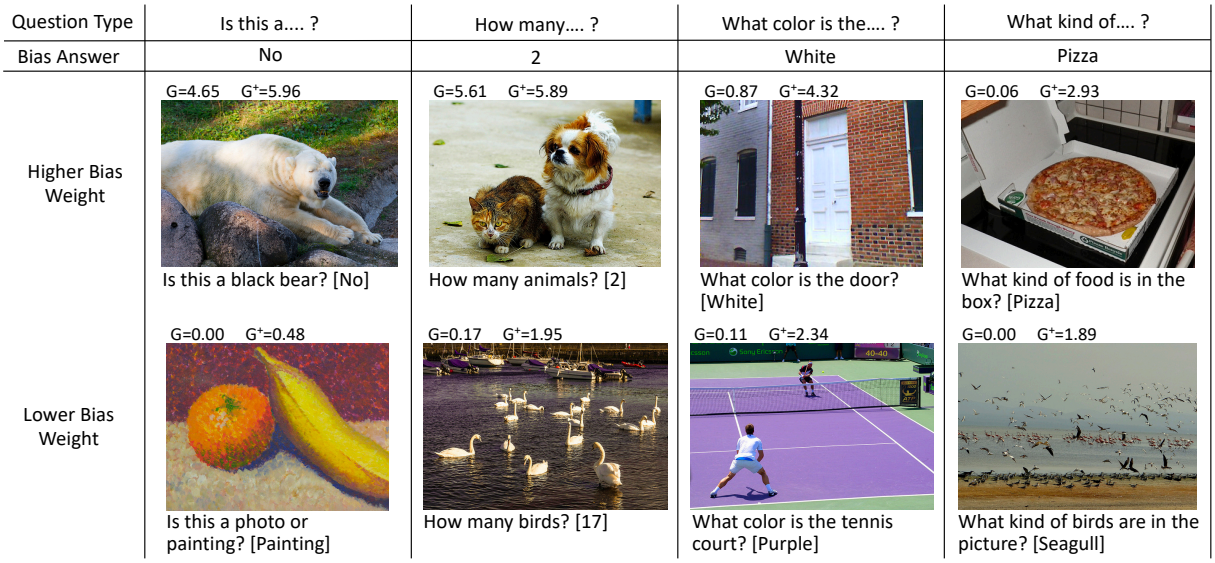


Figure 2: Qualitative examples of the values of $g(x_i)$ on the VQA-CP training data for the learned-mixin model (labelled "G") and learned-mixin +H model (labelled "G+"). The question type and the bias model's highest ranked answer for that type are shown above. We find $g(x_i)$ is larger when the bias answers are likely to be correct.

### 5.4 Adversarial SQuAD

**Data**: We evaluate on the Adversarial SQuAD (Jia and Liang, 2017) dataset, which was built by adding distractor sentences to the passages in SQuAD (Rajpurkar et al., 2016). The sentences are built to closely resemble the question and contain a plausible answer candidate, but with a few key semantic changes to ensure they do not incidentally answer the question. Models that naively focus on sentences that contain many question words are often fooled by the new sentence.

**Bias-Only Models**: We consider two bias-only models: (1) TF-IDF: the TF-IDF score between each sentence and question is used to select an answer (meaning tokens within the same sentence all get the same score) and (2) TF-IDF Filtered: the same but excluding pronouns and numbers from the words used to compute the TF-IDF scores. The second model is motivated by the fact distractor sentences never include numbers or pronouns that occur in the question.

**Main Model**: We use an updated version of BiDAF (Seo et al., 2017), that uses the fasttext words vectors (Mikolov et al., 2018), includes an additional recurrent layer, and simplifies the prediction stage (see Appendix D).

**Results**: Table 5 shows the results. We find the bias product method improves performance by up to 3 points, and the learned-mixin +H model achieves up to a 9 point gain. The importance of including the entropy penalty is explained by the fact that, without the penalty, the model learns to ignore the bias by settings $g(x_i)$ close to zero. For example, on the AddSent dataset with the TF-IDF filtered bias, the learned-mixin ensemble sets $g(x_i)$ to an average of 0.13, while the learned-mixin +H ensemble increases that to 5.16. The high values are likely caused by the fact the bias-only model is very weak, since it assigns the same score to each token in each sentence, so the model can often scale it by large values. As expected, we get better results using the TF-IDF Filtered bias which is more closely tailored to how the test set was constructed.

### 5.5 TriviaQA-CP

**Data**: We construct a changing-prior QA dataset from TriviaQA (Joshi et al., 2017) by categorizing questions into three classes, Person, Location, and Other, based on what kind of entity they are asking about. During training, we hold out all the person questions or all the location questions from the train set, and evaluate on the person or location questions in the TriviaQA dev set. Details can be found in Appendix E.

**Bias-Only Model**: The bias-only model uses NER tags, identified by running the Stanford NER Tagger (Finkel et al., 2005) on the passage, as input.

| Debiasing Method | TF-IDF Filtered | | | TF-IDF | | |
|---|---|---|---|---|---|---|
| | AddSent | AddSentOne | Dev | AddSent | AddSentOne | Dev |
| None | 42.54 | 53.91 | 80.61 | 42.54 | 53.91 | 80.61 |
| Reweight | 41.55 | 53.06 | 80.59 | 42.74 | 53.83 | 80.51 |
| Bias Product | 47.17 | 57.74 | 78.63 | 44.41 | 55.73 | 78.22 |
| Learned-Mixin | 42.25 | 53.51 | 80.39 | 42.00 | 53.46 | 80.46 |
| Learned-Mixin +H | 51.84 | 60.66 | 75.94 | 48.30 | 58.26 | 74.14 |

Table 5: F1 scores on Adversarial SQuAD and the standard SQuAD dev set using two different bias-only models.

| Debiasing Method | Location | | Person | |
|---|---|---|---|---|
| | CP | Dev | CP | Dev |
| None | 41.23 | 59.27 | 39.69 | 55.26 |
| Reweight | 40.14 | 59.18 | 39.96 | 55.38 |
| Bias Product | 44.42 | 60.02 | 40.58 | 55.20 |
| Learned-Mixin | 41.15 | 61.64 | 41.31 | 56.08 |
| Learned-Mixin +H | 47.77 | 57.74 | 44.37 | 54.83 |

Table 6: EM scores on two changing priors TriviaQA datasets. The CP column shows scores on the changing priors test set, and Dev shows in-domain scores.

We only apply the model to tokens that have a NER tag, and assign all other tokens the average score given to the tokens with NER tags in order to prevent the model from reflecting a preference for entity tokens in general.

**Main Model**: We use a larger version of the model used for Adversarial SQuAD (see Appendix D), to account for the larger dataset.

**Results**: Table 6 shows the results. Similar to adversarial SQuAD, the bias product method is moderately effective, and the ensemble method is superior as long as a suitable regularizer is applied. We again observe that the learned-mixin method tends to push $g(x_i)$ close to zero without the entropy penalty (average of $0.25$ without the penalty vs. $5.01$ with the penalty on the Location dev set). We see smaller gains on the person dataset. One possible cause is that differentiating between people and other named entities, such as organizations or groups, is difficult for the main model, and as a result it does not learn a strong non-person prior even without the use of a debiasing method.

### 5.6 Discussion

Despite tackling a diverse range of problems, we were able to improve out-of-domain performance in all settings. The bias product method works consistently, but can almost always be significantly out-performed by the learned-mixin method with an appropriate entropy penalty. The reweight baseline improved performance on HANS, but was relatively ineffective in other cases.

Increasing the out-of-domain performance usually comes at the cost of losing some in-domain performance, which is unsurprising since the biased approaches we are removing are helpful on the in-domain data. TriviaQA-CP stands out as a case where this trade-off is minimal.

A possible issue is that our methods reduce the need for the model to solve examples the bias-only model works well on (since the ensemble's prediction will already be mostly correct for those examples), which effectively reduces the amount of training data. An ideal approach would be to block the model from using the bias-only method, and require it to solve examples the bias-only method solves through other means. We suspect this will necessitate a more clear-box method since it requires doing fine-grained regularization of how the model is solving individual examples.

## 6 Conclusion

Our key contribution is a method of using human knowledge about what methods will not generalize well to improve model robustness to domain-shift. Our approach is to train a robust model in an ensemble with a pre-trained naive model, and then use the robust model alone at test time. Extensive experiments show that our method works well on two adversarial datasets, and two changing-prior datasets, including a 12 point gain on VQA-CP. Future work includes learning to automatically detect dataset bias, which would allow our method to be applicable with less specific prior knowledge.

## Acknowledgements

This work was supported in part by the ARO (ARO-W911NF-16-1-0121) and the NSF (IIS-1252835, IIS-1562364). We thank Sewon Min, Gabriel Stanovsky, Mandar Joshi and the anonymous reviewers for their helpful comments.

## References


Aishwarya Agrawal, Dhruv Batra, Devi Parikh, and Aniruddha Kembhavi. 2018. Don't Just Assume; Look and Answer: Overcoming Priors for Visual Question Answering. In *CVPR*.

Ankesh Anand, Eugene Belilovsky, Kyle Kastner, Hugo Larochelle, and Aaron Courville. 2018. Blindfold Baselines for Embodied QA. *Computing Research Repository*, arXiv:1811.05013. Version 1.

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering. In *CVPR*.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. VQA: Visual Question Answering. In *ICCV*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *ICLR*.

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and Natural Noise Both Break Neural Machine Translation. In *ICLR*.

Yonatan Belinkov, Adam Poliak, Stuart M Shieber, Benjamin Van Durme, and Alexander M Rush. 2019. On Adversarial Removal of Hypothesis-only Bias in Natural Language Inference. In *StarSem*.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A Large Annotated Corpus for Learning Natural Language Inference. In *EMNLP*.

Kaylee Burns, Lisa Anne Hendricks, Kate Saenko, Trevor Darrell, and Anna Rohrbach. 2018. Women Also Snowboard: Overcoming Bias in Captioning Models. In *ECCV*.

Ozan Caglayan, Pranava Madhyastha, Lucia Specia, and Loïc Barrault. 2019. Probing the Need for Visual Context in Multimodal Machine Translation. In *NAACL*.

Nicholas Carlini and David Wagner. 2018. Audio Adversarial Examples: Targeted Attacks on Speech-to-Text. In *2018 IEEE Security and Privacy Workshops*.

Jifan Chen and Greg Durrett. 2019. Understanding Dataset Design Choices for Multi-hop Reasoning. In *NAACL*.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for Natural Language Inference. In *ACL*.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wentau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. QuAC: Question Answering in Context. In *EMNLP*.

Christopher Clark and Matt Gardner. 2018. Simple and Effective Multi-Paragraph Reading Comprehension. In *ACL*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge. *Computing Research Repository*, arXiv:1803.05457. Version 1.

Peter Clark, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Turney, and Daniel Khashabi. 2016. Combining Retrieval, Statistics, and Inference to Answer Elementary Science Questions. In *AAAI*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*.

Harrison Edwards and Amos Storkey. 2016. Censoring Representations with an Adversary. In *ICLR*.

Shi Feng, Eric Wallace, and Jordan Boyd-Graber. 2019. Misleading Failures of Partial-input Baselines. In *ACL*.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *ACL*.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking NLI Systems with Sentences that Require Simple Lexical Inferences. In *ACL*.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2018. Making the V in VQA Matter: Elevating the Role of Image Understanding in Visual Question Answering. *IJCV*.

Gabriel Grand and Yonatan Belinkov. 2019. Adversarial Regularization for Visual Question Answering: Strengths, Shortcomings, and Side Effects. In *Proceedings of the Second Workshop on Shortcomings in Vision and Language*.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R Bowman, and Noah A Smith. 2018. Annotation Artifacts in Natural Language Inference Data. In *NAACL*.

He He, Sheng Zha, and Haohan Wang. 2019. Unlearn Dataset Bias in Natural Language Inference by Fitting the Residual. *Computing Research Repository*, arXiv:1908.10763. Version 1.

Geoffrey E. Hinton. 2002. Training Products of Experts by Minimizing Contrastive Divergence. *Neural Computation*.

Allan Jabri, Armand Joulin, and Laurens Van Der Maaten. 2016. Revisiting Visual Question Answering Baselines. In *European conference on computer vision*.

Robin Jia and Percy Liang. 2017. Adversarial Examples for Evaluating Reading Comprehension Systems. In *EMNLP*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. In *ACL*.

Byungju Kim, Hyunwoo Kim, Kyungsu Kim, Sungjin Kim, and Junmo Kim. 2019. Learning Not to Learn: Training Deep Neural Networks with Biased Data. In *CVPR*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

Mike Lewis and Angela Fan. 2019. Generative Question Answering: Learning to Answer the Whole Question. In *ICLR*.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference. In *ACL*.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in Pre-Training Distributed Word Representations. In *LREC*.

Sewon Min, Eric Wallace, Sameer Singh, Matt Gardner, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2019. Compositional Questions Do Not Necessitate Multi-hop Reasoning. In *ACL*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100, 000+ Questions for Machine Comprehension of Text. In *EMNLP*.

Sainandan Ramakrishnan, Aishwarya Agrawal, and Stefan Lee. 2018. Overcoming Language Priors in Visual Question Answering with Adversarial Regularization. In *NeurIPS*.

Siva Reddy, Danqi Chen, and Christopher D. Manning. 2019. CoQA: A Conversational Question Answering Challenge. In *TACL*.

Roy Schwartz, Maarten Sap, Ioannis Konstas, Li Zilles, Yejin Choi, and Noah A Smith. 2017. The Effect of Different Writing Tasks on Linguistic Style: A Case Study of the ROC Story Cloze Task. In *CoNLL*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. BiDirectional Attention Flow for Machine Comprehension. In *ICLR*.

Andrew Smith, Trevor Cohn, and Miles Osborne. 2005. Logarithmic Opinion Pools for Conditional Random Fields. In *ACL*.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. In *Deep Learning Workshop (ICML)*.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: A Core of Semantic Knowledge. In *WWW*.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing Properties of Neural Networks. In *ICLR*.

Shuohang Wang and Jing Jiang. 2017. Machine Comprehension Using Match-LSTM and Answer Pointer. In *ICLR*.

Tianlu Wang, Jieyu Zhao, Kai-Wei Chang, Mark Yatskar, and Vicente Ordonez. 2018. Adversarial Removal of Gender from Deep Image Representations. *Computing Research Repository*, arXiv:1811.08489. Version 2.

Yicheng Wang and Mohit Bansal. 2018. Robust Machine Comprehension Models via Adversarial Training. In *NAACL*.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing Datasets for Multi-hop Reading Comprehension across Documents. *TACL*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A Dataset for Diverse, Explainable Multi-hop Question Answering. In *EMNLP*.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. Swag: A Large-Scale Adversarial Dataset for Grounded Commonsense Inference. In *EMNLP*.

Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. 2018a. Mitigating Unwanted Biases with Adversarial Learning. In *AIES*.

Peng Zhang, Yash Goyal, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2016. Yin and Yang: Balancing and Answering Binary Visual Questions. In *CVPR*.

Sheng Zhang, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, Kevin Duh, and Benjamin Van Durme. 2018b. ReCoRD: Bridging the Gap between Human and Machine Commonsense Reading Comprehension. *Computing Research Repository*, arXiv:1810.12885. Version 1.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2017. Men Also Like Shopping: Reducing Gender Bias Amplification using Corpus-level Constraints. In *EMNLP*.

## A Entropy Penalty Weights

The strength of the entropy penalty used for the learned-mixin +H model can be found in Table 7.

| Dataset | Experiment | Penalty |
|---|---|---|
| Synthetic | Indicator | 0.01 |
| Synthetic | Excluder | 0.005 |
| Synthetic | Dependent | 0.005 |
| VQA-CP | - | 0.36 |
| HANS | Recurrent | 0.03 |
| HANS | BERT | 0.03 |
| Adver. SQuAD | TF-IDF Filtered | 2.0 |
| Adver. SQuAD | TF-IDF | 2.0 |
| TriviaQA-CP | Location | 0.4 |
| TriviaQA-CP | Person | 0.2 |

Table 7: Entropy penalty weight for the learned-mixin +H ensemble on all our experiments.

## B Co-Attention NLI Model

The model we use for NLI is based on ESIM (Chen et al., 2017). It has the following stages:

**Embed**: Embed the words using a character CNN, following what was done by Seo et al. (2017), and the fasttext crawl word embeddings (Mikolov et al., 2018), then run a shared BiLSTM over the results.

**Co-Attention**: Compute an attention matrix using the formulation from Seo et al. (2017), and use it to compute a context vector for each premise word (Bahdanau et al., 2015). Then build an augmented vector for each premise word by concatenating the word's embedding, the context vector, and the elementwise product of the two. Augmented vectors for the hypothesis are built in the same way using the transpose of the attention matrix.

**Pool**: Run another shared BiLSTM over the augmented vectors, and max-pool the results. The max-pooled vectors from the premise and hypothesis are fed into a fully-connected layer, and then into a softmax layer with three outputs to compute class probabilities.

We apply variational dropout at a rate of 0.2 between all layers, and to the recurrent states of the LSTM, and train the model for 30 epochs using the Adam optimizer (Kingma and Ba, 2015) with a batch size of 32. The learning rate is decayed by 0.999 every 100 steps. We use 200 dimensional LSTMs and a 50 dimensional fully connected layer.

## C Fine-Grained HANS Results

We show the scores our methods achieve for the various heuristics used in HANS in Table 8. Our methods reduce the extent to which models naively guess entailment in all cases. Interestingly, the BERT model shows significantly degraded performance on the entailment examples when using the reweight and bias product method, but largely maintains its performance on those examples when using the learned-mixin method.

## D Modified BiDAF QA Model

The model we use for QA is based on BiDAF (Seo et al., 2017). It has the following stages:

**Embed**: Embed the words using a character CNN following Seo et al. (2017) and the fasttext crawl word embeddings (Mikolov et al., 2018). Then run a BiLSTM over the results to get context-aware question embeddings and passage embeddings.

**Bi-Attention**: Apply the bi-directional attention mechanism from Seo et al. (2017) to produce question-aware passage embeddings.

**Predict**: Apply a fully connected layer, then two more BiLSTM layers, then a two dimensional linear layer to produce start and end scores for each token.

We apply variational dropout at a rate of 0.2 between all layers. We use the Adam optimizer (Kingma and Ba, 2015) with a batch size of 45, while decaying the learning rate by 0.999 every 100 steps.

For SQuAD, we use a 200 dimensional fully connected layer and 100 dimensional LSTMs.

For TriviaQA we use a 256 dimensional fully connected layer and 128 dimensional LSTMs, with highway connections between each BiLSTM (Srivastava et al., 2015) and a recurrent dropout rate of 0.2.

## E TriviaQA-CP

In this section we discuss our changing-prior TriviaQA dataset, TriviaQA-CP. This dataset was built by training a classifier to identify TriviaQA (Joshi et al., 2017) questions as being about people,

| Model | Debiasing Method | MNLI | Correct: *Entailment* | | | Correct: *Non-entailment* | | |
|---|---|---|---|---|---|---|---|---|
| | | | Lexical | Subseq. | Const | Lexical | Subseq. | Const |
| Co-Attention | None | 78.73 | 97.83 | 99.67 | 97.28 | 1.37 | 3.68 | 3.68 |
| | Reweight | 77.03 | 80.10 | 77.84 | 73.76 | 15.68 | 34.27 | 35.44 |
| | Bias Product | 76.63 | 77.89 | 76.61 | 70.95 | 17.89 | 35.11 | 43.71 |
| | Learned-Mixin | 78.05 | 94.84 | 97.25 | 91.19 | 3.69 | 9.57 | 13.37 |
| | Learned-Mixin +H | 74.50 | 67.18 | 61.05 | 47.13 | 27.41 | 56.82 | 60.53 |
| BERT | None | 84.24 | 96.30 | 99.58 | 99.30 | 49.03 | 7.88 | 22.30 |
| | Reweight | 83.54 | 67.93 | 84.34 | 80.97 | 77.44 | 44.87 | 59.57 |
| | Bias Product | 82.97 | 53.67 | 69.47 | 70.88 | 81.34 | 62.93 | 69.23 |
| | Learned-Mixin | 84.29 | 95.64 | 99.52 | 99.14 | 55.41 | 8.34 | 25.96 |
| | Learned-Mixin +H | 83.97 | 91.98 | 98.20 | 97.98 | 64.99 | 13.25 | 30.48 |

Table 8: Scores on individual heuristics in HANS, with scores on the MNLI matched dev set for reference. Results are an average of 8 runs.

| Statistic | Location | Person |
|---|---|---|
| Num Train | 60,133 | 52,953 |
| Num Test | 1,992 | 2,865 |
| Avg. Passage Length | 318 | 317 |
| Avg. Question Length | 16.7 | 16.0 |

Table 9: Statistics for the TriviaQA-CP datasets.

locations, or other topics, and then selecting an answer-containing passage for each question as context. There are two versions of this dataset: a person changing-priors dataset that was built by removing the person questions from the train set and using only person questions from the dev set for evaluation, and a location changing-priors dataset that was built by repeating this process for location questions. Statistics for these two sets are shown in Table 9. We review the three-step procedure we used to construct this dataset below.

**Distantly Supervised Classification**: We first train a preliminary question-type classifier using distant supervision. We noisily label person and location questions using a manually constructed set of patterns (e.g., questions with the phrase "What is the family name of..." are almost always about people), and by attempting to look up the answers in the Yago database (Suchanek et al., 2007) and checking if the answer belongs to a person or location category. Questions that did not match either of these heuristics are labelled as other.

We use these labels to train a simple recurrent model that embeds the question using the fasttext words vectors, applies a 100 dimensional BiLSTM, max-pools, and then applies a softmax layer with 3 outputs. We train the model for 3 epochs using the Adam optimizer (Kingma and Ba, 2015), and apply 0.5 dropout to the embeddings and 0.2 dropout to the recurrent states and the output of the max-pooling layer.

**Supervised Classification**: Next we use higher quality labels to train a second linear classifier to re-calibrate the recurrent model's predictions, and to integrate its predictions with the distantly supervised heuristics. An author manually labelled 1,100 questions, then a classifier was trained on those questions using the predictions from the recurrent model as features, as well as two additional features built from looking up the category of the answer in Yago as before. This classifier was then used to decide the final question classifications.

Table 10 shows the accuracy of these classifiers. The final model achieves about 95% accuracy. We find about 25% of the questions are about people and about 20% of the questions are about locations.

**Paragraph Selection**: In TriviaQA, each question is paired with multiple documents. We simplify the task by selecting a single answer-containing paragraph for each question. We use the approach of Clark and Gardner (2018) to break up the documents into passages of at most 400 tokens, and rank the passages for each question using their linear paragraph ranker. Each question is then paired with the highest ranking paragraph that contains an answer.

| Method | Accuracy | Location | | | Person | | |
|---|---|---|---|---|---|---|---|
| | | Precision | Recall | F1 | Precision | Recall | F1 |
| Patterns | 72.84 | 98.30 | 70.61 | 82.19 | 99.12 | 33.43 | 50.00 |
| Yago | 88.56 | 95.87 | 85.31 | 90.28 | 94.70 | 80.00 | 86.73 |
| Yago + Patterns | 91.73 | 95.44 | 93.88 | 94.65 | 94.70 | 85.37 | 89.80 |
| Dist. Supervised Model | 92.73 | 96.60 | 92.65 | 94.58 | 98.28 | 85.37 | 91.37 |
| Supervised Model | 94.46 | 95.08 | 94.69 | 94.89 | 93.31 | 95.82 | 94.55 |

Table 10: Accuracy, and per-class scores, on the manually annotated questions for the various question classification methods we used when building TriviaQA-CP.

# Achieving Verified Robustness to Symbol Substitutions via Interval Bound Propagation

**Po-Sen Huang**[†] **Robert Stanforth**[†§] **Johannes Welbl**[‡§] **Chris Dyer**[†]
**Dani Yogatama**[†] **Sven Gowal**[†] **Krishnamurthy Dvijotham**[†] **Pushmeet Kohli**[†]

[†]DeepMind [‡]University College London
{posenhuang, stanforth, cdyer, dyogatama, sgowal, dvij, pushmeet}@google.com
{j.welbl}@cs.ucl.ac.uk

## Abstract

Neural networks are part of many contemporary NLP systems, yet their empirical successes come at the price of vulnerability to adversarial attacks. Previous work has used adversarial training and data augmentation to partially mitigate such brittleness, but these are unlikely to find worst-case adversaries due to the complexity of the search space arising from discrete text perturbations. In this work, we approach the problem from the opposite direction: to formally verify a system's robustness against a predefined class of adversarial attacks. We study text classification under synonym replacements or character flip perturbations. We propose modeling these input perturbations as a simplex and then using Interval Bound Propagation – a formal model verification method. We modify the conventional log-likelihood training objective to train models that can be efficiently verified, which would otherwise come with exponential search complexity. The resulting models show only little difference in terms of nominal accuracy, but have much improved verified accuracy under perturbations and come with an efficiently computable formal guarantee on worst case adversaries.

## 1 Introduction

Deep models have been shown to be vulnerable against adversarial input perturbations (Szegedy et al., 2013; Kurakin et al., 2016). Small, semantically invariant input alterations can lead to drastic changes in predictions, leading to poor performance on adversarially chosen samples. Recent work (Jia and Liang, 2017; Belinkov and Bisk, 2018; Ettinger et al., 2017) also exposed the vulnerabilities of neural NLP models, e.g. with small character perturbations (Ebrahimi et al., 2018) or paraphrases (Ribeiro et al., 2018; Iyyer et al., 2018). These adversarial attacks highlight often unintuitive model failure modes and present a challenge to deploying NLP models.

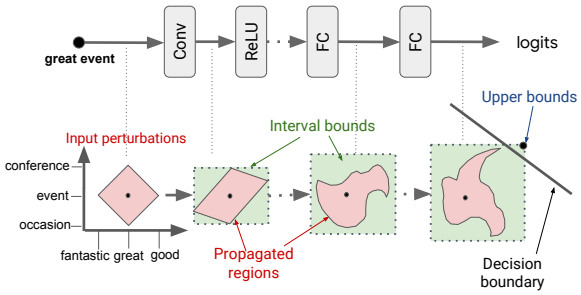


Figure 1: Illustration of verification with the input simplex and Interval Bound Propagation. From the left, input perturbations define the extreme points of a simplex (in red, projected to 2D here) around the statement "great event" that is propagated through a model. At each layer, this shape deforms itself, but can be bounded by axis-parallel bounding boxes, which are propagated similarly. Finally, in logit space, we can compute an upper bound on the worst-case specification violation (e.g., prediction changes).

Common attempts to mitigate the issue are adversarial training (Ebrahimi et al., 2018) and data augmentation (Belinkov and Bisk, 2018; Li et al., 2017), which lead to improved accuracy on adversarial examples. However, this might cause a false sense of security, as there is generally no guarantee that stronger adversaries could not circumvent defenses to find other successful attacks (Carlini and Wagner, 2017; Athalye et al., 2018; Uesato et al., 2018). Rather than continuing the race with adversaries, *formal verification* (Baier and Katoen, 2008; Barrett and Tinelli, 2018; Katz et al., 2017) offers a different approach: it aims at providing provable guarantees to a given model specification. In the case of adversarial robustness, such a specification can be formulated as prediction consistency under

[‡]Work done during an internship at DeepMind.
[§]Equal contribution.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4083–4093, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

*any* altered – but semantically invariant – input change.

In this paper, we study verifiable robustness, i.e., providing a certificate that for a given network and test input, no attack or perturbation under the specification can change predictions, using the example of text classification tasks, Stanford Sentiment Treebank (SST) (Socher et al., 2013) and AG News (Zhang et al., 2015). The *specification* against which we verify is that a text classification model should preserve its prediction under character (or synonym) substitutions in a character (or word) based model. We propose modeling these input perturbations as a simplex and then using Interval Bound Propagation (IBP) (Gowal et al., 2018; Mirman et al., 2018; Dvijotham et al., 2018) to compute worst case bounds on specification satisfaction, as illustrated in Figure 1. Since these bounds can be computed efficiently, we can furthermore derive an auxiliary objective for models to *become* verifiable. The resulting classifiers are efficiently verifiable and improve robustness on adversarial examples, while maintaining comparable performance in terms of nominal test accuracy.

The contributions of this paper are twofold:

- To the best of our knowledge, this paper is the first to introduce verification and verifiable training for neural networks in natural language processing (§3).
- Through a series of experiments (§4), we demonstrate (a) the effectiveness of modeling input perturbations as a simplex and using simplex bounds with IBP for training and testing, (b) the weakness of adversarial training under exhaustive verification, (c) the effects of perturbation space on the performance of different methods, and (d) the impact of using GloVe and counter-fitted embeddings on the IBP verification bounds.

## 2 Related Work

**Adversarial Examples in NLP.** Creating adversarial examples for NLP systems requires identifying semantically invariant text transformations to define an input perturbation space. In this paper, given our specification, we study word- and character-level *HotFlip* attacks (Ebrahimi et al., 2018) – which consist of character and synonym replacements – on text classification tasks. We compare our verifiable approach to other defenses including adversarial training (Goodfellow et al., 2014) and data augmentation (Li et al., 2017; Belinkov and Bisk, 2018). Note that some existing adversarial perturbations such as syntactically controlled paraphrasing (Iyyer et al., 2018), exploiting backtranslation systems (Ribeiro et al., 2018), or using targeted keyword attack (Cheng et al., 2018) are beyond the specification in this paper.

**Formal Verification of Neural Networks.** Formal verification provides a provable guarantee that models are consistent with a *specification* for all possible model inputs. Previous work can be categorised into complete methods that use Mixed-Integer Programming (MIP) (Bunel et al., 2017; Cheng et al., 2017) or Satisfiability Modulo Theory (SMT) (Katz et al., 2017; Carlini et al., 2017), and incomplete methods that solve a convex relaxation of the verification problem (Weng et al., 2018; Wong and Kolter, 2018; Wang et al., 2018). Complete methods perform exhaustive enumeration to find the worst case. Hence, complete methods are expensive and difficult to scale, though they provide exact robustness bounds. Incomplete methods provide loose robustness bounds, but can be more scalable and used inside the training loop for training models to be robust and verifiable (Raghunathan et al., 2018; Wong and Kolter, 2018; Dvijotham et al., 2018; Gowal et al., 2018). Our work is the first to extend incomplete verification to text classification, considering input perturbations on a simplex and minimising worst case bounds to adversarial attacks in text classification. We highlight that the verification of neural networks is an extremely challenging task, and that scaling complete and incomplete methods to large models remains an open challenge.

**Representations of Combinatorial Spaces.** Word lattices and hypergraphs are data structures that have often been used to efficiently represent and process exponentially large numbers of sentences without exhaustively enumerating them. Applications include automatic speech recognition (ASR) output rescoring (Liu et al., 2016), machine translation of ASR outputs (Bertoldi et al., 2007), paraphrase variants (Onishi et al., 2010), and word segmentation alternatives (Dyer et al., 2008). The specifications used to characterise the space of adversarial attacks are likewise a compact representation, and the algorithms discussed below operate on them without exhaustive enumeration.

## 3 Methodology

We assume a fixed initial vector representation $\mathbf{z}_0$ of a given input sentence $z$[1] (e.g. the concatenation of pretrained word embeddings) and use a neural network model, i.e. a series of differentiable transformations $h_k$:

$$\mathbf{z}_k = h_k(\mathbf{z}_{k-1}) \quad k = 1, \ldots, K \tag{1}$$

where $\mathbf{z}_k$ is the vector of activations in the $k$-th layer and the final output $\mathbf{z}_K$ consists of the logits for each class. Typically each $h_k$ will be an affine transformation followed by an activation function (e.g. ReLU or sigmoid). The affine transformation can be a convolution (with the inputs and outputs having an implied 2D structure) of a vector of activations at each point in a sequence; in what follows these activations will be concatenated along the sequence to form a vector $\mathbf{z}_k$.

### 3.1 Verification

Verification is the process of examining whether the output of a model satisfies a given specification. Formally, this means establishing whether the following holds true for a given *normal* model input $\mathbf{x}_0$: $\forall \mathbf{z}_0 \in \mathcal{X}_{\text{in}}(\mathbf{x}_0) : \mathbf{z}_K \in \mathcal{X}_{\text{out}}$, where $\mathcal{X}_{\text{out}}$ characterizes a constraint on the outputs, and $\mathcal{X}_{\text{in}}(\mathbf{x}_0)$ defines a neighbourhood of $\mathbf{x}_0$ throughout which the constraint should be satisfied.

In our concrete use case, we consider a specification of robustness against adversarial attacks which are defined by bounded input perturbations (synonym flips up to $\delta$ words, or character flips up to $\delta$ characters) of the original sentence $x$. The attack space $\mathcal{X}_{\text{in}}(\mathbf{x}_0)$ is the set of vector representations (embeddings) of all such perturbed sentences. Denoting by $z_{K,y}$ the logit of label $y$, we formulate the output constraint that for all classes $y : z_{K,y_{\text{true}}} \geq z_{K,y}$. This specification establishes that the prediction of *all* perturbed sentences $\mathbf{z}_0 \in \mathcal{X}_{\text{in}}(\mathbf{x}_0)$ should correspond to the correct label $y_{\text{true}}$. This specification may equivalently be formulated as a set of half-space constraints on the logits: for each class $y$

$$(\mathbf{e}_y - \mathbf{e}_{y_{\text{true}}})^\top \mathbf{z}_K \leq 0 \quad \forall \mathbf{z}_0 \in \mathcal{X}_{\text{in}}(\mathbf{x}_0) \tag{2}$$

where $\mathbf{e}_i$ is a one-hot vector with 1 in the $i$-th position. In other words, the true class logit should be greater or equal than those for all other classes $y$, which means the prediction remains constant.

[1]For brevity, we will refer both to the original symbol sequence and its corresponding vector representation with the same variable name, distinguishing them by styling.

### 3.2 Verification as Optimisation

Verifying the specification in Eq. (2) can be done by solving the following constrained optimisation problem to find the input that would most strongly violate it:

$$\begin{aligned} \underset{\mathbf{z}_0 \in \mathcal{X}_{\text{in}}(\mathbf{x}_0)}{\text{maximize}} \quad & \mathbf{c}^\top \mathbf{z}_K \\ \text{subject to} \quad & \mathbf{z}_k = h_k(\mathbf{z}_{k-1}) \quad k = 1, \ldots, K \end{aligned} \tag{3}$$

where $\mathbf{c}$ is a vector with entries $c_y = 1$, $c_{y_{\text{true}}} = -1$ and 0 everywhere else. If the optimal value of the above optimisation problem is smaller than 0, then the specification in Eq. (2) is satisfied, otherwise a counter-example has been found. In our case, this corresponds to a successful adversarial attack.

### 3.3 Modeling Input Perturbations using Simplices

In the interests of computational feasibility, we will actually attempt to verify the specification on a larger, but more tractable input perturbation space $\bar{\mathcal{X}}_{\text{in}} \supseteq \mathcal{X}_{\text{in}}$. Any data point that is verifiable on this larger input perturbation space is necessarily verifiable with respect to the original specification.

In the domain of image classification, $\mathcal{X}_{\text{in}}$ is often modeled as an $L_\infty$-ball, corresponding to input perturbations in which each pixel may be independently varied within a small interval. However, using such interval bounds is unsuitable for our situation of perturbations consisting of a small number $\delta$ of symbol substitutions. Although we could construct an axis-aligned bounding box $\bar{\mathcal{X}}_{\text{in}}$ in embedding space that encompasses all of $\mathcal{X}_{\text{in}}$, it would over-approximate the perturbation space to such an extent that it would contain perturbations where *all* symbols in the sentence have been substituted simultaneously.

To remedy this, we propose a tighter over-approximation in the form of a 'simplex' in embedding space. We first define this for the special case $\delta = 1$, in which $\mathcal{X}_{\text{in}} = \{\mathbf{x}_0\} \cup \{\mathbf{p}_0^{(m)} : 1 \leq m \leq M\}$ consists of the representations of all $M$ sentences $p^{(m)}$ derived from $x$ by performing a *single* synonym (or character) substitution, together with the unperturbed sentence $x$ itself. In this case we define $\bar{\mathcal{X}}_{\text{in}}$ to be the convex hull $\mathcal{S}_1$ of $\mathcal{X}_{\text{in}}$. Note we are not considering contextual embeddings (Peters et al., 2018) here. Each 'vertex' $\mathbf{p}_0^{(m)}$ is a sequence of embedding vectors that differs from $\mathbf{x}_0$ at only one word (or character) position.

For a larger perturbation radius $\delta > 1$, the cardinality of $\mathcal{X}_{\text{in}}$ grows exponentially, so manipulating

its convex hull becomes infeasible. However, dilating $\mathcal{S}_1$ centered at $\mathbf{x}_0$, scaling it up by a factor of $\delta$, yields a simplex $\mathcal{S}_\delta$ with $M+1$ vertices that contains $\mathcal{X}_{\text{in}}$.

More formally, we define a region in the input embedding space based on the $M$ 'elementary' perturbations $\{\mathbf{p}_0^{(m)} : m = 1 \dots M\}$ of $\mathbf{x}_0$ defined earlier for the $\delta = 1$ case. For perturbations of up to $\delta$ substitutions, we define $\bar{\mathcal{X}}_{\text{in}}(\mathbf{x}_0)$ as the convex hull of $\{\mathbf{z}_0^{(m)} : m = 0 \dots M\}$, where $\mathbf{z}_0^{(0)} = \mathbf{x}_0$ denotes the original (unperturbed) sentence representation and, for $m \geq 1$, $\mathbf{z}_0^{(m)} = \mathbf{x}_0 + \delta \cdot (\mathbf{p}_0^{(m)} - \mathbf{x}_0)$. The convex hull is an over-approximation of $\mathcal{X}_{\text{in}}(\mathbf{x}_0)$: it contains the representations of all sentences derived from $x$ by performing up to $\delta$ substitutions at distinct word (or character) positions.

### 3.4 Interval Bound Propagation

To estimate the optimal value of the problem (3), given an input $\mathbf{z}_0$, we can propagate the upper/lower bounds on the activations $\mathbf{z}_k$ of each layer using interval arithmetic (Gowal et al., 2018).

We begin by computing interval bounds on the first layer's activations. Recall that any input $\mathbf{z}_0 \in \mathcal{X}_{\text{in}}$ will lie within the convex hull of certain vertices $\{\mathbf{z}_0^{(m)} : m = 0 \dots M\}$. Then, assuming that the first layer $h_1$ is an affine transformation (e.g. linear or convolutional) followed by a monotonic activation function, the lower and upper bounds on the components $z_{1,i}$ of the first layer's activations $\mathbf{z}_1$ are as follows:

$$\begin{aligned} \underline{z}_{1,i}(\delta) &= \min_{m=0,\dots,M} \mathbf{e}_i^\top h_1(\mathbf{z}_0^{(m)}) \\ \overline{z}_{1,i}(\delta) &= \max_{m=0,\dots,M} \mathbf{e}_i^\top h_1(\mathbf{z}_0^{(m)}) \end{aligned} \tag{4}$$

Note that these bounds are efficient to compute (by passing each perturbation $\mathbf{z}_0^{(m)}$ through the first layer); in particular there is no need to compute the convex hull polytope.

For subsequent layers $k > 1$, the bounds on the components $z_{k,i}$ of $\mathbf{z}_k$ are:

$$\begin{aligned} \underline{z}_{k,i}(\delta) &= \min_{\underline{\mathbf{z}}_{k-1}(\delta) \leq \mathbf{z}_{k-1} \leq \overline{\mathbf{z}}_{k-1}(\delta)} \mathbf{e}_i^\top h_k(\mathbf{z}_{k-1}) \\ \overline{z}_{k,i}(\delta) &= \max_{\underline{\mathbf{z}}_{k-1}(\delta) \leq \mathbf{z}_{k-1} \leq \overline{\mathbf{z}}_{k-1}(\delta)} \mathbf{e}_i^\top h_k(\mathbf{z}_{k-1}) \end{aligned} \tag{5}$$

The above optimisation problems can be solved in closed form quickly for affine layers and monotonic activation functions, as illustrated in Gowal et al. (2018). Finally, the lower and upper bounds of the output logits $\mathbf{z}_K$ can be used to construct an upper bound on the solution of (3):

$$\underset{\underline{\mathbf{z}}_K(\delta) \leq \mathbf{z}_K \leq \overline{\mathbf{z}}_K(\delta)}{\text{maximize}} \; \mathbf{c}^\top \mathbf{z}_K \tag{6}$$

**Verifiable Training.** The upper bound in (6) is fast to compute (only requires two forward passes for upper and lower bounds through the network). Hence, we can define a loss to optimise models such that the models are trained to be verifiable. Solving (6) is equivalent to finding the worst-case logit difference, and this is achieved when the logit of the true class is equal to its lower bound, and all other logits equal to their upper bounds. Concretely, for each class $y \neq y_{\text{true}}$: $\hat{\mathbf{z}}_{K,y}(\delta) = \overline{\mathbf{z}}_{K,y}(\delta)$, and $\hat{\mathbf{z}}_{K,y_{\text{true}}}(\delta) = \underline{\mathbf{z}}_{K,y_{\text{true}}}(\delta)$. The training loss can then be formulated as

$$L = \kappa \underbrace{\ell(\mathbf{z}_K, y_{\text{true}})}_{L_{\text{normal}}} + (1-\kappa) \underbrace{\ell(\hat{\mathbf{z}}_K(\delta), y_{\text{true}})}_{L_{\text{spec}}} \tag{7}$$

where $\ell$ is the cross-entropy loss, $\kappa$ a hyperparameter that controls the relative weights between the classification loss $L_{\text{normal}}$ and specification loss $L_{\text{spec}}$. If $\delta = 0$ then $\mathbf{z}_K = \hat{\mathbf{z}}_K(\delta)$, and thus $L$ reduces to a standard classification loss. Empirically, we found that a curriculum-based training, starting with $\kappa$=1 and linearly decreasing to 0.25, is effective for verifiable training.

## 4 Experiments

We conduct verification experiments on two text classification datasets, Stanford Sentiment Treebank (SST) (Socher et al., 2013) and AG News corpus, processed in (Zhang et al., 2015). We focus on word-level and character-level experiments on SST and character-level experiments on AG News. Our specification is that models should preserve their prediction against up to $\delta$ synonym substitutions or character typos, respectively.

### 4.1 A Motivating Example

We provide an example from Table 2 to highlight different evaluation metrics and training methods. Given a sentence, "you ' ve seen them a million times .", that is predicted correctly (called *Nominal Accuracy*[2]) by a classification model, we want to further examine whether the model is robust against character typos (e.g., up to $\delta = 3$ typos) to

[2]We use the term "nominal accuracy" to indicate the accuracy under various adversarial perturbations is much lower.

this example. One way is to use some heuristic to search for a valid example with up to 3 typos that can change the prediction the most (called *adversarial example*). We evaluate the model using this adversarial example and report the performance (called *Adversarial Accuracy*). However, even if the adversarial example is predicted correctly, one can still ask: is the model truly robust against *any* typos (up to 3) to this example? In order to have a certificate that the prediction will not change under any $\delta = 3$ character typos (called *verifiably robust*), we could in theory exhaustively search over all possible cases and check whether any of the predictions is changed (called *Oracle Accuracy*). If we only allow a character to be replaced by another character nearby on the keyboard, already for this short sentence we need to exhaustively search over 2,951 possible perturbations. To avoid this combinatorial growth, we can instead model all possible perturbations using the proposed simplex bounds and propagate the bounds through IBP at the cost of two forward passes. Following Eq. (3), we can check whether this example can be verified to be robust against all perturbations (called *IBP-Verified Accuracy*).

There are also a number of ways in which the training procedure can be enhanced to improve the verifiable robustness of a model against typos to the sentence. The baseline is to train the model with the original/normal sentence directly (called *Normal Training*). Another way is to randomly sample typo sentences among the 2,951 possible perturbations and add these sentences to the training data (called *Data Augmentation Training*). Yet another way is to find, at each training iteration, the adversarial example among the (subset of) 2,951 possible perturbations that can change the prediction the most; we then use the adversarial example alongside the training example (called *Adversarial Training*). Finally, as simplex bounds with IBP is efficient to run, we can train a model to be verifiable by minimising Eq. (7) (called *Verifiable Training*).

### 4.2 Baselines

In this section we detail our baseline models.

**Adversarial Training.** In adversarial training (Madry et al., 2018; Goodfellow et al., 2014), the goal is to optimise the following saddle point problem:

$$\min_{\theta} \mathop{\mathbb{E}}_{(\mathbf{x}_0, y)} \left[ \max_{\mathbf{z}_0 \in \mathcal{X}_{\text{in}}(\mathbf{x}_0)} \ell_\theta(\mathbf{z}_0, y) \right] \tag{8}$$

where the inner maximisation problem is to find an adversarial perturbation $\mathbf{z}_0 \in \mathcal{X}_{\text{in}}(\mathbf{x}_0)$ that can maximise the loss. In the inner maximisation problem, we use HotFlip (Ebrahimi et al., 2018) with perturbation budget $\delta$ to find the adversarial example. The outer minimisation problem aims to update model parameters such that the adversarial risk of (8) is minimised. To balance between the adversarial robustness and nominal accuracy, we use an interpolation weight of 0.5 between the original cross-entropy loss and the adversarial risk.

**Data Augmentation Training.** In the data augmentation setup, we randomly sample a valid perturbation $z$ with perturbation budget $\delta$ from a normal input $x$, and minimise the cross-entropy loss given the perturbed sample $z$ (denoted as data augmentation loss). We also set the interpolation weight between the data augmentation loss and the original normal cross-entropy loss to 0.5.

**Normal Training.** In normal training, we use the likelihood-based training using the normal training input $x$.

### 4.3 Setup

We use a shallow convolutional network with a small number of fully-connected layers for SST and AG News experiments. The detailed model architectures and hyperparameter details are introduced in the supplementary material. Although we use shallow models for ease of verifiable training, our nominal accuracy is on par with previous work such as Socher et al. (2013) (85.4%) and Shen et al. (2018) (84.3%) in SST and Zhang et al. (2015) (87.18%) in AG News. During training, we set the maximum number of perturbations to $\delta = 3$, and evaluate performance with the maximum number of perturbations from $\delta = 1$ to 6 at test time.

For word-level experiments, we construct the synonym pairs using the PPDB database (Ganitkevitch et al., 2013) and filter the synonyms with fine-grained part-of-speech tags using Spacy (Honnibal and Montani, 2017). For character-level experiments, we use synthetic keyboard typos from Belinkov and Bisk (2018), and allow one possible alteration per character that is adjacent to it on an American keyboard. The allowable input perturbation space is much larger than for word-level synonym substitutions, as shown in Table 3.

| | SST-Char-Level | | | SST-Word-Level | | | AG-Char-Level | | |
|---|---|---|---|---|---|---|---|---|---|
| **Training** | **Acc.** | **Adv. Acc.** | **Oracle** | **Acc.** | **Adv. Acc.** | **Oracle** | **Acc.** | **Adv. Acc.** | **Oracle** |
| Normal | 79.8 | 36.5 | 10.3 | 84.8 | 71.3 | 69.8 | 89.5 | 75.4 | 65.1 |
| Adversarial | 79.0 | **74.9** | 25.8 | **85.0** | 76.8 | 74.6 | **90.5** | 85.5 | 81.6 |
| Data aug. | 79.8 | 37.8 | 13.7 | 85.4 | 72.7 | 71.6 | 88.4 | 77.5 | 72.0 |
| Verifiable (IBP) | 74.2 | 73.1 | **73.1** | 81.7 | **77.2** | **76.5** | 87.6 | **87.1** | **87.1** |

Table 1: Experimental results for changes up to $\delta$=3 and $\delta$=2 symbols on SST and AG dataset, respectively. We compare normal training, adversarial training, data augmentation and IBP-verifiable training, using three metrics on the test set: the nominal accuracy, adversarial accuracy, and exhaustively verified accuracy (Oracle) (%).

| Prediction | SST word-level examples (by exhaustive verification, not by adversarial attack) |
|---|---|
| + | it ' s the kind of pigeonhole-resisting romp that hollywood too rarely provides . |
| - | it ' s the kind of pigeonhole-resisting romp that hollywood too rarely **gives** . |
| - | sets up a nice concept for its fiftysomething leading ladies , but fails loudly in execution . |
| + | sets up a nice concept for its fiftysomething leading ladies , but fails **aloud** in execution . |
| Prediction | SST character level examples (by exhaustive verification, not by adversarial attack) |
| - | you ' ve seen them a million times . |
| + | you ' ve se**r**n them a million times . |
| + | choose your reaction : a. ) that sure is funny ! |
| - | choose **t**our reaction : a. ) that sure is funny ! |

Table 2: Pairs of original inputs and adversarial examples for SST sentiment classification found via an exhaustive verification oracle, but not found by the HotFlip attack (i.e., the HotFlip attack does not change the model prediction). The bold words/characters represent the flips found by the adversary that change the predictions.

### 4.4 Evaluation Metrics

We use the following four metrics to evaluate our models: i) test set accuracy (called Acc.), ii) adversarial test accuracy (called Adv. Acc.), which uses samples generated by HotFlip attacks on the original test examples, iii) verifiable accuracy under IBP verification (called IBP-verified), that is, the ratio of test samples for which IBP can verify that the specification is not violated, and iv) exhaustively verified accuracy (called Oracle), computed by enumerating all possible perturbations given the perturbation budget $\delta$, where a sample is verifiably robust if the prediction is unchanged under all valid perturbations.

### 4.5 Results

Table 1 shows the results of IBP training and baseline models under $\delta = 3$ and $\delta = 2$[3] perturbations on SST and AG News, respectively. Figures 2 and 3 show the character- and word-level results with $\delta$ between 1 and 6 under four metrics on the SST test set; similar figures for SST word-level (adversarial training, data augmentation) models and AG News dataset can be found in the supplementary material.

**Oracle Accuracy and Adversarial Accuracy.** In Table 1, comparing adversarial accuracy with exhaustive verification accuracy (oracle), we observe that although adversarial training is effective at defending against HotFlip attacks (74.9 / 76.8 / 85.5%), the oracle adversarial accuracy under exhaustive testing (25.8 / 74.6 / 81.6%) is much lower in SST-character / SST-word / AG-character level, respectively. For illustration, we show some concrete adversarial examples from the HotFlip attack in Table 2. For some samples, even though the model is robust with respect to HotFlip attacks, its predictions are incorrect for stronger adversarial examples obtained using the exhaustive verification oracle. This underscores the need for verification, as robustness with respect to suboptimal adversarial attacks alone might give a false sense of security.

**Effectiveness of Simplex Bounds with IBP.** Rather than sampling individual points from the perturbation space, IBP training covers the full space at once. The resulting models achieve the highest exhaustively verified accuracy at the cost of only moderate deterioration in nominal accuracy (Table 1). At test time, IBP allows for constant-time verification with arbitrary $\delta$, whereas exhaustive verification requires evaluation over an exponentially growing search space.

**Perturbation Space Size.** In Table 1, when the perturbation space is larger (SST character-level vs. SST word-level), (a) across models, there is a larger gap in adversarial accuracy and true robustness

[3]Note that the exhaustive oracle is not computationally feasible beyond $\delta = 2$ on AG News.

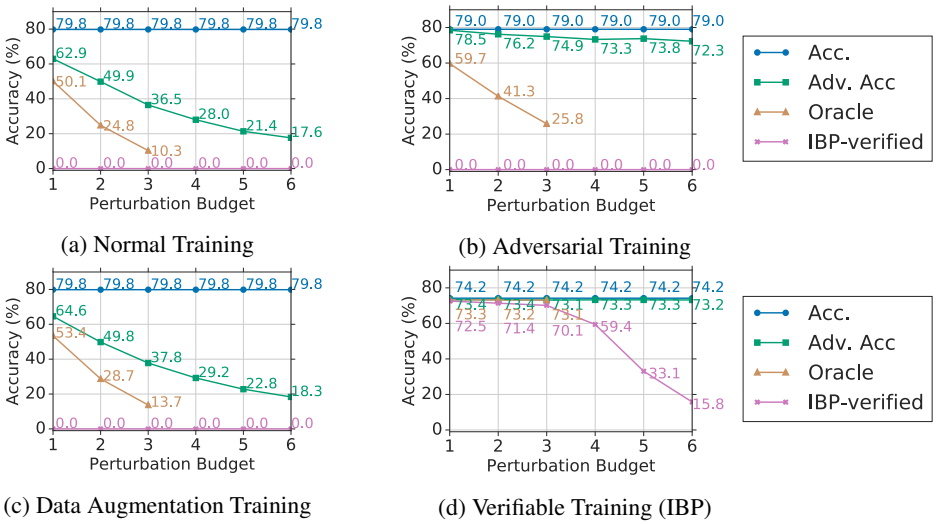


(a) Normal Training (b) Adversarial Training

(c) Data Augmentation Training (d) Verifiable Training (IBP)

Figure 2: SST character-level models with different training objectives (trained at $\delta$=3) against different perturbation budgets in nominal accuracy, adversarial accuracy, exhaustively verified accuracy (Oracle), and IBP verified accuracy. Note that exhaustive verification is not scalable to perturbation budget 4 and beyond.

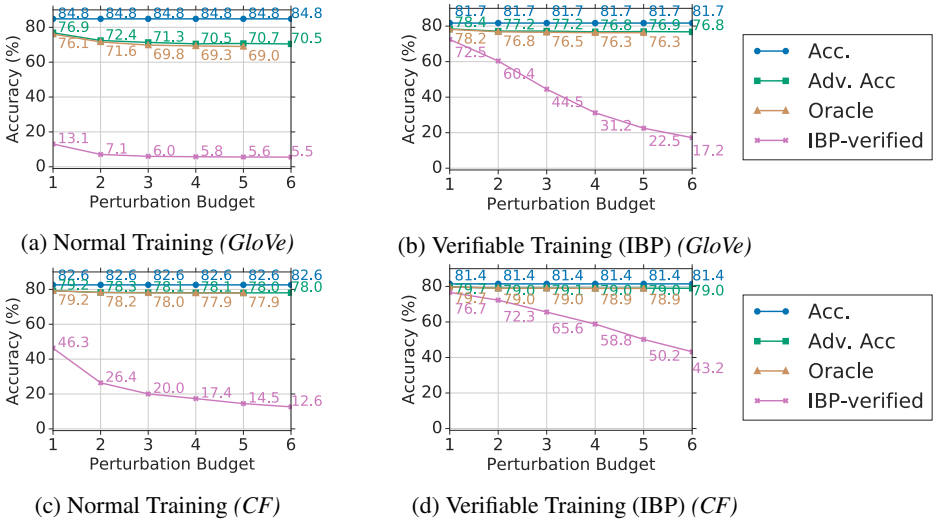


(a) Normal Training *(GloVe)* (b) Verifiable Training (IBP) *(GloVe)*

(c) Normal Training *(CF)* (d) Verifiable Training (IBP) *(CF)*

Figure 3: SST word-level models with normal and verifiable training objectives (trained at $\delta$=3) using *GloVe* and *counter-fitted (CF)* embeddings against different perturbation budgets in nominal accuracy, adversarial accuracy, exhaustively verified accuracy (Oracle), and IBP verified accuracy. Note that exhaustive verification is not scalable to perturbation budget 6 and beyond.

(oracle); (b) the difference in oracle robustness between IBP and adversarial training is even larger (73.1% vs. 25.8% and 76.5% vs. 74.6%).

**Perturbation Budget.** In Figures 2 and 3, we compare normal training, adversarial training, data augmentation, and verifiable training models with four metrics under various perturbation budgets on the SST dataset. Overall, as the perturbation budget increases, the adversarial accuracy, oracle accuracy, and IBP-verified accuracy decrease. We can observe that even for large perturbation budgets, verifiably trained models are still able to verify a sizable number of samples. Again, although adversarial accuracy flattens for larger perturbation budgets in the word level experiments, oracle verification can further find counterexamples to change the prediction. Note that exhaustive verification becomes intractable with large perturbation sizes.

**Computational Cost of Exhaustive Verification.** The perturbation space in NLP problems is discrete and finite, and a valid option to verify the specification is to exhaustively generate predictions for all $\mathbf{z}_0 \in \mathcal{X}_{\text{in}}(\mathbf{x}_0)$, and then check if at least one does not match the correct label. Conversely, such an

| Perturbation radius | $\delta = 1$ | $\delta = 2$ | $\delta = 3$ |
|---|---|---|---|
| SST-word | 49 | 674 | 5,136 |
| SST-character | 206 | 21,116 | 1,436,026 |
| AG-character | 722 | 260,282 | - |

Table 3: Maximum perturbation space size in the SST and AG News test set using word / character substitutions, which is the maximum number of forward passes per sentence to evaluate in the exhaustive verification.

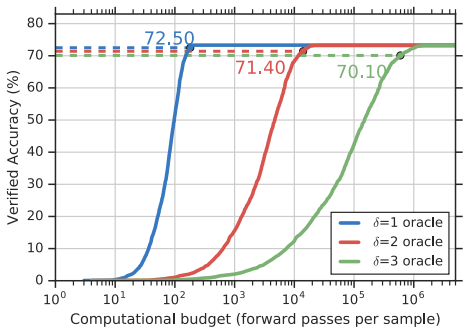


Figure 4: Verified accuracy vs. computation budget for exhaustive verification oracles on the SST character-level test set, for an IBP-trained model trained with $\delta$=3. Solid lines represent the number of forward passes required to verify a given proportion of the test set using exhaustive search. Dashed lines indicate verification levels achievable using IBP verification, which comes at small and constant cost, and is thus orders of magnitude more efficient.

exhaustive (oracle) approach can also identify the strongest possible attack. But the size of $\mathcal{X}_{\text{in}}$ grows exponentially with $\delta$, and exhaustive verification quickly becomes prohibitively expensive.

In Table 3, we show the maximum perturbation space size in the SST and AG News test set for different perturbation radii $\delta$. This number grows exponentially as $\delta$ increases. To further illustrate this, Figure 4 shows the number of forward passes required to verify a given proportion of the SST test set for an IBP-trained model using exhaustive verification and IBP verification. IBP reaches verification levels comparable to an exhaustive verification oracle, but requires only two forward passes to verify any sample – one pass for computing the upper, and one for the lower bounds. Exhaustive verification, on the other hand, requires several orders of magnitude more forward passes, and there is a tail of samples with extremely large attack spaces.

### 4.6 Counter-Fitted Embeddings

As shown in Figures 2 and 3a, although IBP can verify arbitrary networks in theory, the verification bound is very loose except for models trained to be IBP-verifiable. One possible reason is the potentially large volume of the perturbation simplex. Since representations of substitution words/characters are not necessarily close to those of synonyms/typos in embedding space, the vertices of the simplex could be far apart, and thus cover a large area in representation space. Therefore, when propagating the interval bounds through the network, the interval bounds become too loose and fail to verify most of the examples if the models are not specifically trained. To test this hypothesis, we follow Mrkšić et al. (2016) and use fine-tuned GloVe embeddings trained to respect linguistic constraints; these representations (called counter-fitted embeddings) force synonyms to be closer and antonyms to be farther apart using word pairs from the PPDB database (Ganitkevitch et al., 2013) and WordNet (Miller, 1995). We repeat the word level experiments with these counter-fitted embeddings, Figures 3c and 3d show the experimental results. We observe that IBP verified accuracy is now substantially higher across models, especially for $\delta = 1, 2, 3$. The examples which IBP can verify increase by up to 33.2% when using the counter-fitted embeddings (normal training, $\delta = 1$). Moreover, adversarial and exhaustively verified accuracy are also improved, at the cost of a mild deterioration in nominal test accuracy. The IBP-trained model also further improves both its oracle accuracy and IBP verified accuracy. These results validate our hypothesis that reducing the simplex volume via soft linguistic constraints can provide even tighter bounds for IBP, resulting in larger proportions of verifiable samples.

## 5 Discussion

Our experiments indicate that adversarial attacks are not always the *worst* adversarial inputs, which can only be revealed via verification. On the other hand, exhaustive verification is computationally very expensive. Our results show that using the proposed simplex bounds with IBP can verify a sizable amount of test samples, and can be considered a potent verification method in an NLP context. We note however two limitations within the scope of this work: i) limited model depth: we only investigated models with few layers. IBP bounds are likely to become looser as the number of layers increases. ii) limited model types: we only studied models with CNN and fully connected layers.

We focused on the HotFlip attack to showcase specification verification in the NLP context, with the goal of understanding factors that impact its effectiveness (e.g. the perturbation space volume, see Section 4.6). It is worth noting that symbol substitution is general enough to encompass other threat models such as lexical entailment perturbations (Glockner et al., 2018), and could potentially be extended to the addition of pre/postfixes (Jia and Liang, 2017; Wallace et al., 2019).

Interesting directions of future work include: tightening IBP bounds to allow applicability to deeper models, investigating bound propagation in other types of neural architectures (e.g. those based on recurrent networks or self-attention), and exploring other forms of specifications in NLP.

## 6 Conclusion

We introduced formal verification of text classification models against synonym and character flip perturbations. Through experiments, we demonstrated the effectiveness of the proposed simplex bounds with IBP both during training and testing, and found weaknesses of adversarial training compared with exhaustive verification. Verifiably trained models achieve the highest exhaustive verification accuracy on SST and AG News. IBP verifies models in constant time, which is exponentially more efficient than naive verification via exhaustive search.

## References


Anish Athalye, Nicholas Carlini, and David A. Wagner. 2018. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *ICML*, pages 274–283.

Christel Baier and Joost-Pieter Katoen. 2008. *Principles of Model Checking*. MIT press.

Clark Barrett and Cesare Tinelli. 2018. Satisfiability modulo theories. In *Handbook of Model Checking*, pages 305–343. Springer.

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and natural noise both break neural machine translation. In *International Conference on Learning Representations*.

Nicola Bertoldi, Richard Zens, and Marcello Federico. 2007. Speech translation by confusion network decoding. In *Proc. ICASSP*.

Rudy Bunel, Ilker Turkaslan, Philip HS Torr, Pushmeet Kohli, and M Pawan Kumar. 2017. Piecewise linear neural network verification: a comparative study. *arXiv preprint arXiv:1711.00455*.

Nicholas Carlini, Guy Katz, Clark Barrett, and David L Dill. 2017. Ground-truth adversarial examples. *arXiv preprint arXiv:1709.10207*.

Nicholas Carlini and David Wagner. 2017. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 3–14. ACM.

Chih-Hong Cheng, Georg Nührenberg, and Harald Ruess. 2017. Maximum resilience of artificial neural networks. In *International Symposium on Automated Technology for Verification and Analysis*, pages 251–268. Springer.

Minhao Cheng, Jinfeng Yi, Huan Zhang, Pin-Yu Chen, and Cho-Jui Hsieh. 2018. Seq2Sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. *CoRR*, abs/1803.01128.

Krishnamurthy Dvijotham, Sven Gowal, Robert Stanforth, Relja Arandjelovic, Brendan O'Donoghue, Jonathan Uesato, and Pushmeet Kohli. 2018. Training verified learners with learned verifiers. *arXiv preprint arXiv:1805.10265*.

Christopher Dyer, Smaranda Muresan, and Philip Resnik. 2008. Generalizing word lattice translation. In *Proceedings of ACL-08: HLT*, pages 1012–1020, Columbus, Ohio. Association for Computational Linguistics.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. HotFlip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 31–36. Association for Computational Linguistics.

Allyson Ettinger, Sudha Rao, Hal Daumé III, and Emily M. Bender. 2017. Towards linguistically generalizable NLP systems: A workshop and shared task. In *Proceedings of the First Workshop on Building Linguistically Generalizable NLP Systems*.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764, Atlanta, Georgia. Association for Computational Linguistics.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking NLI systems with sentences that require simple lexical inferences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 650–655, Melbourne, Australia. Association for Computational Linguistics.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy A. Mann, and Pushmeet Kohli. 2018. On the effectiveness of interval bound propagation for training verifiably robust models. *CoRR*, abs/1810.12715.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885, New Orleans, Louisiana. Association for Computational Linguistics.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. 2017. Reluplex: An efficient SMT solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, pages 97–117. Springer.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2016. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*.

Yitong Li, Trevor Cohn, and Timothy Baldwin. 2017. Robust training under linguistic adversity. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 21–27.

Xunying Liu, Xie Chen, Yongqiang Wang, Mark J. F. Gales, and Philip C. Woodland. 2016. Two efficient lattice rescoring methods using recurrent neural network language models. *IEEE/ACM Trans. Audio, Speech & Language Processing*, 24(8):1438–1449.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*.

George A. Miller. 1995. WordNet: A lexical database for English. *Commun. ACM*, 38(11):39–41.

Matthew Mirman, Timon Gehr, and Martin Vechev. 2018. Differentiable abstract interpretation for provably robust neural networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 3578–3586.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 142–148, San Diego, California. Association for Computational Linguistics.

Takashi Onishi, Masao Utiyama, and Eiichiro Sumita. 2010. Paraphrase lattice for statistical machine translation. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 1–5, Uppsala, Sweden. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. 2018. Certified defenses against adversarial examples. In *International Conference on Learning Representations*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Semantically equivalent adversarial rules for debugging NLP models. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 856–865. Association for Computational Linguistics.

Dinghan Shen, Guoyin Wang, Wenlin Wang, Martin Renqiang Min, Qinliang Su, Yizhe Zhang, Chunyuan Li, Ricardo Henao, and Lawrence Carin. 2018. Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 440–450, Melbourne, Australia. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642. Association for Computational Linguistics.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. 2019. Robustness may be at odds with accuracy. In *International Conference on Learning Representations*.

Jonathan Uesato, Brendan O'Donoghue, Pushmeet Kohli, and Aron van den Oord. 2018. Adversarial risk and the dangers of evaluating against weak attacks. In *ICML*, pages 5032–5041.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal trigger sequences for attacking and analyzing NLP. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. 2018. Formal security analysis of neural networks using symbolic intervals. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1599–1614, Baltimore, MD. USENIX Association.

Lily Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Luca Daniel, Duane Boning, and Inderjit Dhillon. 2018. Towards fast computation of certified robustness for ReLU networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5276–5285, Stockholmsmssan, Stockholm Sweden. PMLR.

Eric Wong and Zico Kolter. 2018. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pages 5283–5292.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, pages 649–657.

# Rethinking Cooperative Rationalization: Introspective Extraction and Complement Control

**Mo Yu**[*] [♡] **Shiyu Chang**[*] [♣♡] **Yang Zhang**[*] [♣♡] **Tommi S. Jaakkola** [♠]
[♡] IBM Research [♣] MIT-IBM Watson AI Lab [♠] MIT
yum@us.ibm.com {shiyu.chang, yang.zhang2}@ibm.com tommi@csail.mit.edu

## Abstract

Selective rationalization has become a common mechanism to ensure that predictive models reveal how they use any available features. The selection may be soft or hard, and identifies a subset of input features relevant for prediction. The setup can be viewed as a cooperate game between the selector (aka rationale generator) and the predictor making use of only the selected features. The co-operative setting may, however, be compromised for two reasons. First, the generator typically has no direct access to the outcome it aims to justify, resulting in poor performance. Second, there's typically no control exerted on the information left outside the selection. We revise the overall co-operative framework to address these challenges. We introduce an introspective model which explicitly predicts and incorporates the outcome into the selection process. Moreover, we explicitly control the rationale complement via an adversary so as not to leave any useful information out of the selection. We show that the two complementary mechanisms maintain both high predictive accuracy and lead to comprehensive rationales.[1]

## 1 Introduction

Rapidly expanding applications of complex neural models also bring forth criteria other than mere performance. For example, medical (Yala et al., 2019) and other high-value decision applications require some means of verifying reasons for the predicted outcomes. This area of self-explaining models in the context of NLP applications has primarily evolved along two parallel tracks. On one hand, we can design neural architectures that expose more intricate mechanisms of reasoning such as module networks (Andreas et al., 2016a,b; Johnson et al., 2017). While important, such approaches may still require adopting specialized designs and architectural choices that do not yet reach accuracies comparable to black-box approaches. On the other hand, we can impose limited architectural constraints in the form of selective rationalization (Lei et al., 2016; Li et al., 2016b; Chen et al., 2018a,b) where the goal is to only expose the portion of the text relevant for prediction. The selection is done by a separately trained model called rationale generator. The resulting text selection can be subsequently used as an input to an unconstrained, complex predictor, *i.e.*, architectures used in the absence of any rationalization.[2] The main challenge of this track is how to properly coordinating the rationale generator with the powerful predictor operating on the selected information during training.

| |
|---|
| **Label:** negative<br>**Original Text:** *really cloudy , lots of sediment , washed out yellow color . looks pretty gross , actually , like swamp water . no head , no lacing .*<br>**Rationale from (Lei et al., 2016)**:<br>[*"really cloudy lots", "yellow", "no", "no"*]<br>**Rationale from cooperative introspection model**:<br>[*". looks", "no", "no"*]<br>**Rationale from our full model**:<br>[*"cloudy", "lots", "pretty gross", "no lacing"*] |

Table 1: An example of the rationales extracted by different models on the sentiment analysis task of beer reviews (appearance aspect). *Red* words are human-labeled rationales. Details of the experiments can be found in Appendix B.

In this paper, we build on and extend selective rationalization. The selection process can be thought of as a cooperative game between the generator and the predictor operating on the selected,

[*]Authors contributed equally to this work.

[1]The code and data for our method is publicly available at https://github.com/Gorov/three_player_for_emnlp.

[2]Therefore the selective rationalization approach is easier to be adapted to new applications, such as question answering (Yu et al., 2018), image classification (Chen et al., 2018a) and medical image analysis (Yala et al., 2019).



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4094–4103, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

partial input text. The two players aim for the shared goal of achieving high predictive accuracy, just having to operate within the confines imposed by rationale selection (a small, concise portion of input text). The rationales are learned entirely in an unsupervised manner, without any guidance other than their size, form. An example of ground-truth and learned rationales are given in Table 1.

The key motivation for our work arises from the potential failures of cooperative selection. Since the generator typically has no direct access to the outcome it aims to justify, the learning process may converge to a poorly performing solution. Moreover, since only the selected portion is evaluated for its information value (via the predictor), there is typically no explicit control over the remaining portion of the text left outside the rationale. These two challenges are complementary and should be addressed jointly.

**Performance** The clues in text classification tasks are typically short phrases (Zaidan et al., 2007). However, diverse textual inputs offer a sea of such clues that may be difficult to disentangle in a manner that generalizes to evaluation data. Indeed, the generator may fail to disentangle the information about the correct label, offering misleading rationales instead. Moreover, as confirmed by the experiments presented in this paper, the collaborative nature of the game may enable the players to select a sub-optimal communication code that does not generalize, rather overfits the training data. Regression tasks considered in prior work typically offer greater feedback for the generator, making it less likely that such communication patterns would arise.

We address these concerns by proposing an *introspective rationale generator*. The key idea is to force the generator to explicitly understand what to generate rationales for. Specifically, we make the label that would be predicted with the full text as an additional input to the generator thereby ensuring better overall performance.

**Rationale quality** The cooperative game setup does not explicitly control the information left out of the rationale. As a result, it is possible for the rationales to degenerate as in containing only select words without the appropriate context. In fact, the introspective generator proposed above can aggravate this problem. With access to the predicted label as input, the generator and the predictor can find a communication scheme by encoding the predicted label with special word patterns (*e.g.* highlighting "." for positive examples and "," negative ones). Table 1 shows such degenerate cases for the two cooperative methods.

In order to prevent degenerate rationales, we propose a *three-player game* that renders explicit control over the unselected parts. In addition to the generator and the predictor as in conventional cooperative rationale selection schemes, we add a third adversarial player, called the *complement predictor*, to regularize the cooperative communication between the generator and the predictor. The goal of the complement predictor is to predict the correct label using only words left out of the rationale. During training, the generator aims to fool the complement predictor while still maintaining high accuracy for the predictor. This ensures that the selected rationale must contain all/most of the information about the target label, leaving out irrelevant parts, within size constraints imposed on the rationales.

We also theoretically show that the equilibrium of the three-player game guarantees good properties for the extracted rationales. Moreover, we empirically show that (1) the three-player framework on its own helps cooperative games such as (Lei et al., 2016) to improve both predictive accuracy and rationale quality; (2) by combining the two solutions – introspective generator and the three player game – we can achieve high predictive accuracy and non-degenerate rationales.

## 2 Problem Formulation

This section formally defines the problem of rationalization, and then proposes a set of conditions that desirable rationales should satisfy, which addresses problems of previous cooperative frameworks. Here are some notations throughout this paper. Bolded upper-cased letters, *e.g.* $\boldsymbol{X}$, denote random vectors; unbolded upper-cased letters, *e.g.* $X$, denote random scalar variables; bolded lower-cased letters, *e.g.* $\boldsymbol{x}$, denote deterministic vectors or vector functions; unbolded lower-cased letters, *e.g.* $x$, denote deterministic scalars or scalar functions. $p_X(\cdot|Y)$ denotes conditional probability density/mass function conditional on $Y$. $H(\cdot)$ denotes Shannon entropy. $\mathbb{E}[\cdot]$ denotes expectation.

### 2.1 Problem Formulation of Rationalization

The target application here is text classification on data tokens in the form of $\{(\boldsymbol{X}, Y)\}$. Denote

$\boldsymbol{X} = \boldsymbol{X}_{1:L}$ as a sequence of words in an input text with length $L$. Denote $Y$ as a label. Our goal is to generate a rationale, denoted as $\boldsymbol{r}(\boldsymbol{X}) = \boldsymbol{r}_{1:L}(\boldsymbol{X})$, which is a selection of words in $\boldsymbol{X}$ that accounts for $Y$. Formally, $\boldsymbol{r}(\boldsymbol{X})$ is a hard-masked version of $\boldsymbol{X}$ that takes the following form at each position $i$:

$$\boldsymbol{r}_i(\boldsymbol{X}) = \boldsymbol{z}_i(\boldsymbol{X}) \cdot \boldsymbol{X}_i, \tag{1}$$

where $\boldsymbol{z}_i \in \{0,1\}^N$ is the binary mask. Many previous works (Lei et al., 2016; Chen et al., 2018a) follows the above definition of rationales. In this work, we further define the complement of rationale, denoted as $\boldsymbol{r}^c(\boldsymbol{X})$, as

$$\boldsymbol{r}_i^c(\boldsymbol{X}) = (1 - \boldsymbol{z}_i(\boldsymbol{X})) \cdot \boldsymbol{X}_i. \tag{2}$$

For notational ease, define

$$\boldsymbol{R} = \boldsymbol{r}(\boldsymbol{X}), \quad \boldsymbol{R}^c = \boldsymbol{r}^c(\boldsymbol{X}), \quad \boldsymbol{Z} = \boldsymbol{z}(\boldsymbol{X}). \tag{3}$$

### 2.2 Rationale Conditions

An ideal rationale should satisfy the following conditions.

**Sufficiency:** $\boldsymbol{R}$ is sufficient to predict $Y$, *i.e.*

$$p_Y(\cdot|\boldsymbol{R}) = p_Y(\cdot|\boldsymbol{X}). \tag{4}$$

**Comprehensiveness:** $\boldsymbol{R}^c$ does not contain sufficient information to predict $Y$, *i.e.*

$$H(Y|\boldsymbol{R}^c) \geq H(Y|\boldsymbol{R}) + h, \tag{5}$$

for some constant $h$.

**Compactness:** the segments in $\boldsymbol{X}$ that are included in $\boldsymbol{R}$ should be sparse and consecutive, *i.e.,*

$$\sum_i \boldsymbol{Z}_i \leq s, \quad \sum_i |\boldsymbol{Z}_i - \boldsymbol{Z}_{i-1}| \leq c, \tag{6}$$

for some constants $s$ and $c$.

Here is an explanation of what each of these conditions means. The sufficiency condition is the core one of a legitimate rationale, which essentially stipulates that the rationale maintains the same predictive power as $\boldsymbol{X}$ to predict $Y$. The compactness condition stipulates that the rationale should be continuous and should not contain more words than necessary. For example, without the compactness condition, a trivial solution to Eq. (4) would be $\boldsymbol{X}$ itself. The first inequality in Eq. (6) constrains the sparsity of rationale, and the second one constrains the continuity. The comprehensiveness condition requires some elaboration, which we will discuss in the next subsection.

### 2.3 Comprehensiveness and Degeneration

There are two justifications of the comprehensiveness condition. First, it regulates the information outside the rationale, so that the rationale contains all the relevant and useful information, hence the name comprehensiveness. Second and more importantly, we will show there exists a failure case, called degeneration, which can only be prevented by the comprehensiveness condition.

Degeneration refers to the situation where, rather than finding words in $\boldsymbol{X}$ that explains $Y$, $\boldsymbol{R}$ attempts to encode the probability of $Y$ using trivial information, *e.g.* punctuation and position. Consider the following toy example of binary classification ($Y \in \{0,1\}$), where $\boldsymbol{X}$ can always perfectly predict $Y$. Then the following rationale satisfies the sufficiency and compactness: $\boldsymbol{R}$ includes only the first word of $\boldsymbol{X}$ when $Y = 0$, and only the last word when $Y = 1$. It is obvious that this $\boldsymbol{R}$ is sufficient to predict $\boldsymbol{Y}$ (by looking at if the first word or the last word is chosen), and thus satisfies the sufficiency condition. Apparently this $\boldsymbol{R}$ is perfectly compact (only one word). However, this rationale does not provide a valid explanation.

Theoretically, any previous cooperative framework may suffer from the above ill-defined problem, if the generator has the potential to accurately guess $Y$ with sufficient capacity.[3] This problem happens because they have no control of the words unselected by $\boldsymbol{R}$. Intuitively, in the presence of degeneration, some key predictors in $\boldsymbol{X}$ will be left unselected by $\boldsymbol{R}$. Thus by looking at the predictive power of $\boldsymbol{R}^c$, we can determine if degeneration occurs. Specifically, when degeneration is present, all the useful information is left unselected by $\boldsymbol{R}$, and so $H(Y|\boldsymbol{R}^c)$ is low. That is why the lower bound in Eq. (5) rules out the degeneration cases.

## 3 The Proposed Three-Player Models

This section introduces our new rationalization solutions, which is theoretically guaranteed to be able to find the rationales that satisfy Eqs. (4-6).

### 3.1 The Basic Three-Player Model

This section introduces our basic three-player model. The model consists of three players: a **rationale generator** that generates the rationale $\boldsymbol{R}$ and its complement $\boldsymbol{R}^c$ from text, a **predictor** that predicts the probability of $Y$ based on $\boldsymbol{R}$, a **com-**

[3] We show such cases of (Lei et al., 2016) in Appendix B.

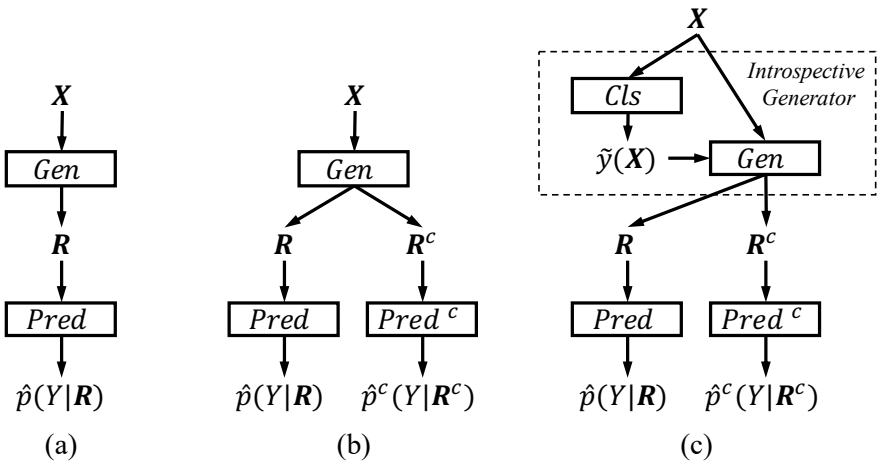


Figure 1: Illustration of different rationalization frameworks. (a) The cooperative framework from (Lei et al., 2016), which consists of two players, *i.e.,* a generator (*Gen*) and a predictor (*Pred*). (b) A straightforward extension of the model (a) with a compliment predictor (*Pred* $^c$). The generator plays a cooperative game with the predictor and plays a mini-max game with the compliment predictor. (c) The introspective three-player framework. The introspective module first predicts the possible outcome $\tilde{y}$ based on the full texts $x$ and then generate rationales using both $x$ and $\tilde{y}$. The third framework (c) is a special case of (b). Such an inductive bias in model design preserves the predictive performance.

**plement predictor** that predicts the probability of $Y$ based on $\boldsymbol{R}^c$.

Figure 1(b) illustrates the basic three-player model. Compared with (Lei et al., 2016), as shown in Figure 1(a), the three-player model introduces an additional complement predictor, which plays a minimax game in addition to the cooperative game in (Lei et al., 2016). For clarity, we will describe the game backward, starting from the two predictors followed by the generator.

**Predictors:** The predictor estimates probability of $Y$ conditioned on $\boldsymbol{R}$, denoted as $\hat{p}(Y|\boldsymbol{R})$. The complement predictor estimates probability of $Y$ conditioned on $\boldsymbol{R}^c$, denoted as $\hat{p}^c(Y|\boldsymbol{R})$. Both predictors are trained using the cross entropy loss, *i.e.*

$$\begin{aligned}\mathcal{L}_p &= \min_{\hat{p}(\cdot,\cdot)} -H(p(Y|\boldsymbol{R}); \hat{p}(Y|\boldsymbol{R})) \\ \mathcal{L}_c &= \min_{\hat{p}^c(\cdot,\cdot)} -H(p(Y|\boldsymbol{R}^c); \hat{p}^c(Y|\boldsymbol{R}^c)),\end{aligned} \tag{7}$$

where $H(p;q)$ denotes the cross entropy between $p$ and $q$. $p(\cdot|\cdot)$ denotes the empirical distribution. It is worth emphasizing that $\mathcal{L}_p$ and $\mathcal{L}_c$ are both functions of the generator.

**Generator:** The generator extracts $\boldsymbol{R}$ and $\boldsymbol{R}^c$ by generating the rationale mask, $\boldsymbol{z}(\cdot)$, as shown in Eqs. (1-2). Specifically, $\boldsymbol{z}(\cdot)$ is determined by minimizing the weighted combination of four losses:

$$\min_{\boldsymbol{z}(\cdot)} \mathcal{L}_p + \lambda_g \mathcal{L}_g + \lambda_s \mathcal{L}_s + \lambda_c \mathcal{L}_c, \tag{8}$$

where $\mathcal{L}_g$ encourages the gap between $\mathcal{L}_p$ and $\mathcal{L}_c$ to be large, *i.e.*

$$\mathcal{L}_g = \max\{\mathcal{L}_p - \mathcal{L}_c + h, 0\}. \tag{9}$$

It stipulates the comprehensiveness property of the rationale (Eq. (5)). Intuitively, if the complement rationale is less informative of $Y$ than the rationale, then $\mathcal{L}_c$ should be larger than $\mathcal{L}_p$.

$\mathcal{L}_s$ and $\mathcal{L}_c$ impose the sparsity and continuity respectively, which correspond to Eq. (6):

$$\begin{aligned}\mathcal{L}_s &= \max\Big\{\sum_i \boldsymbol{Z}_i - s, 0\Big\}, \\ \mathcal{L}_c &= \sum_i \max\{||\boldsymbol{Z}_i - \boldsymbol{Z}_{i-1}|| - c, 0\}.\end{aligned} \tag{10}$$

From Eq. (7), we can see that the generator plays a *cooperative game* with the predictor, because both tries to maximize the predictive performance of $\boldsymbol{R}$. On the other hand, the generator plays an *adversarial game* with the complement predictor, because the latter tries to maximize the predictive performance of $\boldsymbol{R}^c$, but the former tries to reduce it. Without the complement predictor, and thus the loss $\mathcal{L}_g$, the framework reduces to the method in (Lei et al., 2016).

**Training** During training, the three players perform gradient descent steps with respect to their own losses. For the generator, Since $\boldsymbol{z}(\boldsymbol{X})$ is a set of binary variables, we cannot apply the regular gradient descent algorithm. Instead we will use policy gradient (Williams, 1992) to optimize the models. We maximize the reward that is defined as the negative loss in Eq. (8). In order to have bounded rewards for training stability, the negative losses $\mathcal{L}_p$ and $\mathcal{L}_c$ are replaced with accuracy.

**Theoretical guarantees** The proposed framework is able to obtain a rationale that simultane-

ously satisfies the conditions in Eqs. (4) to (6), as stated in the following theorem:

**Theorem 1.** *A rationalization scheme $\boldsymbol{z}(\boldsymbol{X})$ that simultaneously satisfies Eqs.* (4)-(6) *is the global optimizer of Eq.* (8)*.*

The proof is given in Appendix A. The basic idea is that there is a correspondence between each term in Eq. (8) and each of the properties Eqs. (4)-(6). The minimization of each loss term is equivalent to satisfying the corresponding property.

### 3.2 The Introspection Generator

As discussed in Section 1, in the existing generator-predictor framework, the generator may fail to disentangle the information about the correct label, offering misleading rationales instead. To address this problem, we propose a new generator module, called the **Introspective Generator**, which explicitly predicts the label before making rationale selections.

Figure 1(c) illustrates the model with the introspection generator. Specifically, the improved generator still fits into the basic three-player framework in Section 3.1. The only difference is in how the generator generates the mask $\boldsymbol{z}(\boldsymbol{X})$, which now breaks down into two steps.

First, the module uses a *regular classifier* that takes the input $\boldsymbol{X}$ and predicts the label, denoted $\tilde{y}(\boldsymbol{X})$. For classification tasks, we use the maximum likelihood estimate, *i.e.*

$$\tilde{y}(\boldsymbol{X}) = \underset{y}{\operatorname{argmax}}\ \tilde{p}(Y = y|\boldsymbol{X}), \tag{11}$$

where $\tilde{p}(Y = y|\boldsymbol{X})$ is the predicted probability by maximizing the cross entropy, which is pretrained.

Second, a label-aware rationale generator generates the binary mask of the rationales, *i.e.*

$$z(\boldsymbol{X}) = \tilde{z}(\boldsymbol{X}, \tilde{y}(\boldsymbol{X})). \tag{12}$$

Note that $\tilde{y}$ is a function of $\boldsymbol{X}$, so the entire introspective generator is essentially a function of $\boldsymbol{X}$. In this way, all the formulations in Section 3.1 and the Theorem 1 still hold for the three-player game with the introspective generator.

In the implementation, the classifier can use the same architecture like the predictor and the complement predictor. The generator is of the same architecture in Section 3.1, but with the additional input of $\hat{y}(\boldsymbol{X})$.

**Remark on degeneration** Obviously, when working in a cooperative game, the introspection generator will make the degeneration problem more severe: when the classifier $\tilde{p}(\cdot|\boldsymbol{X})$ becomes sufficiently accurate during training, the generator only needs to encode the information of $\tilde{y}$ into $\boldsymbol{R}$. Therefore our three-player game, while improving over any existing generator-predictor frameworks on its own, is critical for the introspective model.

## 4 Experimental Settings

### 4.1 Datasets

We construct three text classification tasks, including two sentiment classification tasks from the BeerAdvocate review dataset (McAuley et al., 2012)[4], and a more complex relation classification task from SemEval 2010 Task 8 (Hendrickx et al., 2009). Table 2 gives examples of the above tasks. Finally, as suggested by an anonymous reviewer, we evaluate on the text matching benchmark AskUbuntu, following Lei et al. (2016). The experimental setting and results are reported in Appendix F.

**Multi-aspect beer review** This is the same data used in (Lei et al., 2016). Each review evaluates multiple aspects of a kind of beer, including appearance, smell, palate, and an overall score. For each aspect, a rating $\in$ [0,1] is labeled. We limit ourselves to the appearance aspect only and use a threshold of 0.6 to create balanced binary classification tasks for each aspect. Then, the task becomes to predict the appearance aspect of a beer based on multi-aspect text inputs. The advantage of this dataset is that it enables automatic evaluation of rationale extraction. The dataset provides sentence-level annotations on about 1,000 reviews, where each sentence is labeled by the aspect it covers. Note that in this dataset, each aspect is often described by a single sentence with clear polarity. Thus, a generator can select a sentence based on the topic distribution of words. The selected sentence often has very high overlap with the ground truth annotations and also contains sufficient information for predicting the sentiment. This characteristic of the dataset makes it a relatively easy task, and thus we further consider two more challenging tasks.

**Single-aspect beer review** To construct a more challenging task, for each review, we extract the

[4]http://snap.stanford.edu/data/web-BeerAdvocate.html

| Task | Label | Input Texts |
|---|---|---|
| Multi-Aspect Beer Review | positive (regarding the aspect of *appearance*) | ***Clear, burnished copper-brown topped by a large beige head that displays impressive persistance and leaves a small to moderate amount of lace in sheets when it eventually departs***. *The nose is sweet and spicy and the flavor is malty sweet, accented nicely by honey and by abundant caramel/toffee notes. There's some spiciness (especially cinnamon), but it's not overdone. The finish contains a moderate amount of bitterness and a trace of alcohol. The mouthfeel is exemplary* $\cdots$ |
| Single-Aspect Beer Review | positive | *appearance : dark-brown/black color with a* ***huge tan head*** *that gradually collapses , leaving* ***thick lacing*** . |
| Relation Classification | Message-Topic(e1,e2) | *It was a friendly* ***call***$_{e_1}$ ***to remind*** *them* ***about the bill***$_{e_2}$ *and make sure they have a copy of the invoice* |

Table 2: Example of the three tasks used for evaluation. The ***red bold*** words are sample rationales (created by the authors for illustration purpose only for the last two tasks).

sentences that are specifically about the appearance aspect from the aforementioned BeerAdvocate dataset[5], and the task is to predict the sentiment of the appearance only on the extracted sentences. The details of the dataset construction can be found in Appendix C. This new dataset is obviously more challenging in terms of generating meaningful rationales. This is because the generator is required to select more find-grained rationales. Since there are no rationale annotations, we rely on subjective evaluations to test the quality of the extracted rationales.

**Multi-class relation classification** To show the generalizability of our proposed approaches to other NLP applications, we further consider the SemEval 2010 Task 8 dataset (Hendrickx et al., 2009). Given two target entities $e_1$ and $e_2$ in a sentence, the goal is to classify the relation type (with directions) between the two entities (including None if there is no relation). Similar to the single-aspect beer review, it is also a fine-grained rationalization task. The major difference is that the number of class labels in this task is much larger, and we hope to investigate its effects on degeneration and performance downgrade.

### 4.2 Implementation Details

For both the generators and the two predictors, we use bidirectional LSTMs with hidden dimension 400. In the introspection generator, the classifier consists of the same bidirectional LSTM, and $\boldsymbol{z}(\boldsymbol{X}, \tilde{y})$ is implemented as an LSTM sequential labeler with the label $\tilde{y}$ transformed to an embedding vector that serves as the initial hidden states of the LSTM. For the relation classification task, since the model needs to be aware of the two target entities, we add the relative position features following Nguyen and Grishman (2015). We map the relative position features to learnable embedding vectors and concatenate them with word embeddings as the inputs to the LSTM encoder of each player. All hyper-parameters are tuned on the development sets according to predictive accuracy. In other words, all the models are tuned **without** seeing any rationale annotations.

[5]We will release the single-aspect dataset.

| Model | 10% Highlighting Acc | 10% Highlighting Prec | 20% Highlighting Acc | 20% Highlighting Rec |
|---|---|---|---|---|
| Random | 65.70 | 17.75 | 72.31 | 19.84 |
| Lei2016 | 82.05 | 86.14 | 83.88 | 79.98 |
| +minimax | 83.45 | 86.54 | 84.25 | 85.16 |
| Intros | 87.10 | 68.37 | 87.50 | 59.63 |
| +minimax | 86.16 | 85.67 | 86.86 | 79.40 |

Table 3: Main results of binary classification on multi-aspect beer reviews. The *Acc* column shows the accuracy of prediction. The *Prec* and *Rec* are precision and recall on the extracted rationales compared to the human annotations. The accuracy of sentiment prediction with the full texts is **87.59**.

## 5 Experiments

### 5.1 Multi-Aspect Beer Review

Table 3 summarizes the main results on the multi-aspect beer review task. In this task, a desirable rationale should be both appearance-related and sufficient for sentiment predictions. Since the average sparsity level of human-annotated rationales is about 18%, we consider the following evaluation settings. Specifically, we compare the generated rationales to human annotations by measuring the precision when extracting 10% of words and the

recall for 20%[6]. In addition to the precision/recall, we also report the predictive accuracy of the extracted rationales on predicting the sentiment.

When only 10% of the words are used, Lei et al. (2016) has a significant performance downgrade compared to the accuracy when using the whole passage (82.05 v.s. 87.59). With the additional third player added, the accuracy is slightly improved, which validates that controlling the unselected words improves the robustness of rationales. On the other hand, our introspection models are able to maintain higher predictive accuracy (86.16 v.s. 82.05) compared to (Lei et al., 2016), while only sacrificing a little loss on highlighting precision (0.47% drop).

Similar observations are made when 20% of the words required to highlight with one exception. Comparing the model of (Lei et al., 2016) with and without the proposed mini-max module, there is a huge gap of more than 5% on recall of generated rationales. This confirms the motivation that the original cooperative game tends to generate less comprehensive rationales, where the three-player framework controls the unselected words to be less informative so the recall is significantly improved.

It is worth mentioning that when a classifier is trained with randomly highlighted rationales (*i.e.* random dropout (Srivastava et al., 2014) on the inputs), it performs significantly worse on both predictive accuracy and highlighting qualities. This confirms that extracting concise and sufficient rationales is not a trivial task. Moreover, our re-implementation of (Lei et al., 2016) for the original regression task achieves 90.1% precision when highlighting 10% words, which suggest that rationalization for binary classification is more challenging compared to the regression where finer supervision is available.

In summary, our three-player framework consistently improves the quality of extracted rationales on both of the original (Lei et al., 2016) and the introspective framework. Particularly, without controlling the unselected words, the introspection model experiences a serious degeneration problem as expected. With the three-player framework, it manages to maintain both high predictive accuracy and high quality of rationalization.

[6]Previous work in (Lei et al., 2016) only evaluates the precision for the 10% extraction.

| Model | Single-Aspect Beer | | Relation | |
|---|---|---|---|---|
| | Acc | $\text{Acc}^c$ | Acc | $\text{Acc}^c$ |
| All (100%) | 87.1 | 51.3 | 77.8 | 4.9 |
| Random | 74.3 | 83.1 | 51.8 | 64.9 |
| Lei2016 | 81.4 | 74.6 | 73.5 | 36.8 |
| +minimax | 82.9 | 73.2 | 75.0 | 30.9 |
| Intros | 87.0 | 85.8 | 75.2 | 56.6 |
| +minimax | 85.3 | 73.6 | 76.2 | 29.1 |

Table 4: Predictive accuracy on the single-aspect beer appearance review (left) and the relation classification (right). $\text{Acc}^c$ refers to the accuracy of the complement predictor. We restrict the extracted rationales to be on average eight words and four continuous pieces for the beer review and six words and three pieces for relation classification. A desired rationalization method will have high Acc and low $\text{Acc}^c$.

### 5.2 Single-Aspect Beer Review

In this section, we evaluate the proposed methods on the more challenging single-aspect beer dataset. Similar to previous experiments, we force all the methods to have comparable highlighting ratio and continuity constraints for fair evaluation. The highlighting ratio is determined from human estimation on a small set of data. We report two classification results, which are the accuracy of the predictor and complement predictor. For both cooperative methods, *i.e.* (Lei et al., 2016) with and without introspection , we train an independent extra predictor on the unselected words from the generator, which does not affect the training of the generator-predictor framework.

From the left part of Table 4, we observe that the original model of (Lei et al., 2016) has a hard time maintaining the accuracy compared to the classifier trained with full texts. Transforming it into a three-player game helps to improve the performance of the evaluation set while lower the accuracy of the complement predictor. Since the extracted rationales are in a similar length, these results suggest that learning with a three-player game successfully enforces the generator not to leave informative words unselected. Similar to the multi-aspect results, the cooperative introspection model suffers from the degeneration problem, which produces a high complement accuracy. The three-player game yields a more comprehensive rationalization with small losses in accuracy.

**Human evaluation** We further conduct subjective evaluations by comparing the original model of (Lei et al., 2016) with our introspective three-player model. We mask the extracted rationales and present the unselected words only to the hu-

| Model | %UNK | Acc | $\text{Acc}_{\text{w/o UNK}}$ |
|---|---|---|---|
| Lei2016 | 43.5 | 63.5 | 69.0 |
| Intros+minimax | **54.0**$^{\dagger}$ | **58.0** | **66.3** |

Table 5: Subjective evaluations on the task of controlling the unselected rationale words. Acc denotes the accuracy in guessing sentiment labels. $\text{Acc}_{\text{w/o UNK}}$ denotes the sentiment accuracy for these samples that are not selected as "UNK" for the secondary task. † denotes p-value $< 0.005$ in t-test. A desired rationalization method achieves high "UNK" rate and performance randomly for the Acc predictions.

man evaluators. The evaluators are asked to predict the sentiment label as their first task. If a rationalizing method successfully includes all informative pieces in the rationale, subjects should have around 50% of accuracy in guessing the label. In addition, after providing the sentiment label, subjects are then asked to answer a secondary question, which is whether the provided text spans are sufficient for them to predict the sentiment. If they believe there are not enough clues and their sentiment classification is based on a random guess, they are instructed to select unknown (denoted as "UNK") as the answer to the second question. Appendix E elaborates why we design subjective evaluations in such a way in more details.

Table 5 shows the performance of subjective evaluations. Looking at the first column of the table, our model is better in confusing human, which gives a higher rate in selecting "UNK". It confirms that the three-player introspective model selects more comprehensive rationales and leave less informative texts unattended. Furthermore, the results also show that human evaluators offer worse sentiment predictions on the proposed approach, which is also desired and expected.

### 5.3 Relation Classification

The predictive performances on the relation classification task are shown in the right part of the Table 4. We observe consistent results as in previous datasets. Clearly, the introspective generator helps the accuracy and the three-player game regularize the complement of the rationale selections.

**Examples of the extracted rationales** For relation classification, it is difficult to conduct subjective evaluations because the task requires people to have sufficient knowledge of the schema of relation annotation. To further demonstrate the quality of generated rationales, we provide some illustrative examples. Since there is a rich form of supervised signal, *i.e.,* the number of class labels is large, the chance of any visible degeneration of the Lei et al. (2016)'s model should be low. However, we still spot quite a few cases. In the first example, Lei et al. (2016) fails to highlight the second entity while ours does. In the second example, the introspective three-player model selects more words than (Lei et al., 2016). In this case, the two entities themselves suffice to serve as the rationales. However, our model preserves the words like "working". This problem might due to the bias of the dataset. For example, some words that are not relevant to the target entities may still correlate with the labels. In the case, our model will pick these words as a part of the rationale.

| |
|---|
| **Original Text**: *of the hundreds of strains of avian influenza a* ***viruses***$_{e_1}$ *, only four have caused human* ***infections***$_{e_2}$ *: h5n1 , h7n3 , h7n7 , and h9n2*<br>**Label:** *Cause-Effect*($e_1$,$e_2$) |
| **Lei et al. (2016)**:<br>[***viruses***$_{e_1}$ *, only four have caused*]<br>**Our Intros+minimax**:<br>[***viruses***$_{e_1}$], [*four have caused human* ***infections***$_{e_2}$] |
| **Original Text**: *i spent a year working for a* ***software***$_{e_1}$ ***company***$_{e_2}$ *to pay off my college loans*<br>**Label:** *Product-Producer*($e_1$,$e_2$) |
| **Lei et al. (2016)**:<br>[*a* ***software***$_{e_1}$ ***company***$_{e_2}$ *to*]<br>**Our Intros+minimax**:<br>[*working for a* ***software***$_{e_1}$ ***company***$_{e_2}$], [*loans*] |

Table 6: Illustrative examples of generated rationales on the relation classification task. Entities are shown in ***bold***.

## 6 Related Work

**Model interpretability** Besides the two major categories of self-explaining models discussed in Section 1, model interpretability is widely studied in the general machine learning field. For example, evaluating feature importance with gradient information (Simonyan et al., 2013; Li et al., 2016a; Sundararajan et al., 2017) or local perturbations (Kononenko et al., 2010; Lundberg and Lee, 2017); and interpreting deep networks by locally fitting interpretable models (Ribeiro et al., 2016; Alvarez-Melis and Jaakkola, 2018).

Besides selective rationalization, the cooperative game has been studied in the latter direction above (Lee et al., 2018). It has also been applied to a relevant problem on summarization (Arumae and Liu, 2019), where the selected summary should be sufficient for answering questions related to the document. In this problem, the sum-

mary is a special type of rationale of a document. Another related concurrent work (Bastings et al., 2019) proposes differentiable solution to optimize the cooperative rationalization method.

**Game-theoretical methods** Though not having been explored much for self-explaining models, the minimax game setup has been widely used in many machine learning problems, such as self-playing for chess games (Silver et al., 2017), generative models (Goodfellow et al., 2014) and many tasks that can be formulated as multi-agent reinforcement learning (Busoniu et al., 2006). Our three-player game framework also shares a similar idea with (Zhang et al., 2017; Zhao et al., 2017), which aim to learn domain-invariant representations with both cooperative and minimax games.

## 7 Conclusion

We proposed a novel framework for improving the predictive accuracy and comprehensiveness of the selective rationalization methods. This framework (1) addresses the degeneration problem in previous cooperative frameworks by regularizing the unselected words via a three-player game; and (2) augments the conventional generator with introspection, which can better maintain the performance for down-stream tasks. Experiments with both automatic evaluation and subjective studies confirm the advantage of our proposed framework.

## References


David Alvarez-Melis and Tommi S Jaakkola. 2018. Towards robust interpretability with self-explaining neural networks. *arXiv preprint arXiv:1806.07538*.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016a. Learning to compose neural networks for question answering. In *Proceedings of NAACL-HLT*, pages 1545–1554.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016b. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 39–48.

Kristjan Arumae and Fei Liu. 2019. Guiding extractive summarization with question-answering rewards. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2566–2577.

Joost Bastings, Wilker Aziz, and Ivan Titov. 2019. Interpretable neural predictions with differentiable binary variables. *arXiv preprint arXiv:1905.08160*.

Lucian Busoniu, Robert Babuska, and Bart De Schutter. 2006. Multi-agent reinforcement learning: A survey. In *2006 9th International Conference on Control, Automation, Robotics and Vision*, pages 1–6. IEEE.

Jianbo Chen, Le Song, Martin Wainwright, and Michael Jordan. 2018a. Learning to explain: An information-theoretic perspective on model interpretation. In *International Conference on Machine Learning*, pages 882–891.

Jianbo Chen, Le Song, Martin J Wainwright, and Michael I Jordan. 2018b. L-Shapley and C-Shapley: Efficient model interpretation for structured data. *arXiv preprint arXiv:1808.02610*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2009. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, pages 94–99. Association for Computational Linguistics.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Judy Hoffman, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017. Inferring and executing programs for visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2989–2998.

Igor Kononenko et al. 2010. An efficient explanation of individual classifications using game theory. *Journal of Machine Learning Research*, 11(Jan):1–18.

Guang-He Lee, David Alvarez-Melis, and Tommi S Jaakkola. 2018. Game-theoretic interpretability for temporal modeling. *arXiv preprint arXiv:1807.00130*.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 107–117.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016a. Visualizing and understanding neural models in NLP. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 681–691.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016b. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220*.

Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pages 4765–4774.

Julian McAuley, Jure Leskovec, and Dan Jurafsky. 2012. Learning attitudes and attributes from multi-aspect reviews. In *2012 IEEE 12th International Conference on Data Mining*, pages 1020–1025. IEEE.

Thien Huu Nguyen and Ralph Grishman. 2015. Combining neural networks and log-linear models to improve relation extraction. *arXiv preprint arXiv:1511.05926*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why should I trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144. ACM.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. 2017. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3319–3328. JMLR. org.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*.

Adam Yala, Constance Lehman, Tal Schuster, Tally Portnoi, and Regina Barzilay. 2019. A deep learning mammography-based model for improved breast cancer risk prediction. *Radiology*, page 182716.

Mo Yu, Shiyu Chang, and Tommi S Jaakkola. 2018. Learning corresponded rationales for text matching.

Omar Zaidan, Jason Eisner, and Christine Piatko. 2007. Using annotator rationales to improve machine learning for text categorization. In *Human language technologies 2007: The conference of the North American chapter of the association for computational linguistics; proceedings of the main conference*, pages 260–267.

Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2017. Aspect-augmented adversarial networks for domain adaptation. *Transactions of the Association for Computational Linguistics*, 5:515–528.

Mingmin Zhao, Shichao Yue, Dina Katabi, Tommi S Jaakkola, and Matt T Bianchi. 2017. Learning sleep stages from radio signals: A conditional adversarial architecture. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 4100–4109. JMLR. org.

# Experimenting with Power Divergences for Language Modeling

**Matthieu Labeau** **Shay B. Cohen**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
{mlabeau,scohen}@inf.ed.ac.uk

## Abstract

Neural language models are usually trained using Maximum-Likelihood Estimation (MLE). The corresponding objective function for MLE is derived from the Kullback-Leibler (KL) divergence between the empirical probability distribution representing the data and the parametric probability distribution output by the model. However, the word frequency discrepancies in natural language make performance extremely uneven: while the perplexity is usually very low for frequent words, it is especially difficult to predict rare words.

To address that, we experiment with several families ($\alpha$, $\beta$ and $\gamma$) of *power divergences*, generalized from the KL divergence, for learning language models with an objective different than standard MLE. Intuitively, these divergences should affect the way the probability mass is spread during learning, notably by prioritizing performance on high or low-frequency words. In addition, we implement and experiment with various sampling-based objectives, where the computation of the output layer is only done on a small subset of the vocabulary.

They are derived as *power* generalizations of a softmax approximated via Importance Sampling, and Noise Contrastive Estimation, for accelerated learning. Our experiments on the Penn Treebank and Wikitext-2 show that these power divergences can indeed be used to prioritize learning on the frequent or rare words, and lead to general performance improvements in the case of sampling-based learning.

## 1 Introduction

Language models are an important component in many NLP tasks, where they provide prior knowledge on the language used. They are conditional models that aim to predict the next token in a sequence: they can be applied to basic units ranging from individual characters to full words, each approach coming with its own benefits and limitations (Merity et al., 2018a). Word-level language models have traditionally been based on $n$-gram counts, obtaining good performance with smoothing techniques (Kneser and Ney, 1995; Goodman, 2001). Recently, neural networks have shown strong results in language modeling (Bengio et al., 2001), especially recurrent neural networks (Mikolov et al., 2011b). As previous approaches, like maximum entropy models (Berger et al., 1996), neural language models are trained via Maximum Likelihood Estimation (MLE). Thus, their training cost grows linearly with the number of words in the vocabulary, often making it prohibitively slow. This motivated a large amount of research work, bringing a variety of solutions (Chen et al., 2016).

The large vocabulary sizes encountered in training corpora arguably stem from the fact that the frequency distribution of words in a corpus of natural language follows Zipf's law (Powers, 1998). This also implies that the discrepancy between counts of high-frequency and low-frequency words increases with the size of the corpus, as well as the number of those low-frequency words. As a consequence, distributed word representations of low-frequency words are difficult to learn. Numerous approaches use decomposition of words with various sub-word units (Sennrich et al., 2016; Kim et al., 2016), but the same phenomenon exists for low-frequency subwords. In order to deal with this issue and to accelerate training, Grave et al. (2017a) implement a dependency between embedding sizes and word frequencies in the output layer, while Baevski and Auli (2019) apply it to the input layer, comparing the possible choices of which units to model. Using a different approach, Gong et al. (2018) attempt to learn word representations that are less affected by these large discrepancies in word frequencies, with an adver-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4104–4114,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

sarial training method to force the model to make frequent and rare word embeddings hard to differentiate based on word frequency alone.

These improvements have been obtained by explicitly incorporating in the model different ways of treating words according to their frequency. However, learning is always made via (or approximating) Maximum Likelihood Estimation, which finds the distribution that maximizes entropy subject to the constraints given by training examples. In this work, we explore the possibility of affecting how words are learned depending on their frequency by using alternative loss functions. We specifically explore *power divergences*, obtained through various generalizations of the Kullback-Leibler divergence, which is traditionally used to obtain the MLE objective function. This is motivated by the intuition that a well-suited power factor may direct learning towards prioritizing high or low-frequency words, instead of learning uniformally.

In this paper, we derive and experiment with the objective functions obtained from three power-divergences: the $\alpha$, $\beta$ and $\gamma$ divergences. We also derive objective functions for the corresponding generalizations of two sampling-based methods: an Approximated Softmax obtained with importance sampling, and Noise Contrastive Estimation. We conduct a set of experiments comparing these objectives and their effect of various parts of the word frequency distribution, by training and evaluating models on two corpora: the Penn Treebank and Wikitext-2. Our experiments show that depending on the vocabulary used and the choice of power divergence, it is indeed possible to gear learning to focus on the most frequent or infrequent words. We also observe that, while the MLE gives the best overall performance for exact objectives, derived from the KL-divergence, our generalized objectives yield perplexity improvements compared to baselines for both sampling-based methods, up to 1 point in perplexity on both corpora.

## 2 Background

Language modeling aims to learn a probability distribution over a sequence of tokens from a finite target vocabulary $\mathcal{Y}$. Such a distribution is decomposed into a product of conditional distributions of tokens over $\mathcal{Y}$ given the previous tokens in the sequence. Hence, we learn a parametric model of the form $p_\theta(y|x)$, where $x \in \mathcal{X}$ represents the sequence of previous tokens, $y$ is a target label belonging to $\mathcal{Y}$, and $\theta$ is the set of model parameters. They are obtained via maximum likelihood estimation (MLE), which consists in minimizing the *negative likelihood* objective function:

$$\mathrm{NLL}(\theta) = -\sum_{(x,y)\in\mathcal{D}} \log p_\theta(y|x)$$

over examples $(x, y)$ corresponding to sequences of tokens drawn from the data $\mathcal{D}$. This can be seen as minimizing the *Kullback-Leibler* divergence between the parametrized probability distribution $p_\theta$ that we are learning and the distribution $p_\mathcal{D}$ described by our training data:

$$D_{KL}(p_\mathcal{D}||p_\theta) = \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} p_\mathcal{D}(y|x) \log \frac{p_\mathcal{D}(y|x)}{p_\theta(y|x)},$$

since $p_\mathcal{D}(y|x) = 1$ if the sequence $(x, y)$ appears in the training data, and equals 0 otherwise. Hence, the set of parameters $\theta^*$ minimizing $D_{KL}(p_\mathcal{D}||p_\theta)$ is the maximum likelihood distribution on the training data $\mathcal{D}$.

In this work, we will use several classes of divergences. A measure of discrepancy $D$ between two probability densities $p$ and $q$ is a divergence if $D(p||q) \geq 0$ with equality if and only if $p = q$.[1] In the following, we will derive an objective function from a divergence $D$ with the data distribution as first argument, and the second being the distribution of the parametric model:

$$Obj(\theta) = D(p_\mathcal{D}||p_\theta).$$

## 3 Power Divergences

A large number of divergence measures has been introduced for a variety of applications (Basseville, 2013). Several families of divergences are notably obtained from generalizing the Kullback-Leibler divergence by using a *generalized logarithm* function, which is a power function:

$$\log_\alpha(x) = \frac{1}{1-\alpha}(x^{1-\alpha} - 1), \qquad (1)$$

defined by a parameter $\alpha$, and that converges to the logarithm as $\alpha \to 1$. In this section, we will consider three families of divergences that can be generated from this function; see Cichocki and Amari

[1] Divergences are not necessarily metrics: they are defined only by non-negativity and positive definiteness.

(2010) for a review. For now, they are to be applied only to *normalized* probability densities, implying that $\sum_{y \in \mathcal{Y}} p_\theta(y|x) = 1$.

### 3.1 $\alpha$-divergences

The notion of $\alpha$-divergence was introduced by Csiszar (1967). The full expression of $D_\alpha(p||q)$ is shown in Table 1. It is a special case of $f$-divergence (Ali and Silvey, 1966), derived from a standardized version of the generalized logarithm[2] (Eq. 1). Applying $\alpha$-divergences to parameter estimation generalizes MLE, and can be shown to have similar properties, as consistency and asymptotic normality of the estimation error (Keziou, 2003). Intuitively, the choice of $\alpha$ will impact the importance of the likelihood ratio $p/q$: while the limiting cases $\alpha \to 1$ and $\alpha \to 0$ are the Kullback-Leibler $D_{KL}(p||q)$ and the reverse Kullback-Leibler divergences $D_{KL}(q||p)$, having $\alpha \geq 1$ will make learning zero-avoiding[3] for $q$, and $\alpha \leq 0$ zero-forcing[4] (Minka, 2005). We should also note that $D_\alpha(p||q) = D_{1-\alpha}(q||p)$. When working with normalized densities, the $\alpha$-divergence is linked to the Rényi divergence (Rényi, 1961), which has been used to measure domain similarity (Van Asch and Daelemans, 2010). Given that we are trying to learn from conditional distributions which are zero everywhere except for one target token, we are interested in experimenting with values of $\alpha > 1$, which should push the model towards generalizing, while having $\alpha \in (0.5, 1)$ should force the model to concentrate probability mass on training examples. Since we are here working with normalized distribution, we obtain the following objective:

$$Obj_\alpha(\theta) = \frac{1}{\alpha(\alpha-1)} \sum_{(x,y)\in\mathcal{D}} (p_\theta(y|x))^{1-\alpha}.$$

### 3.2 $\beta$-divergences

The $\beta$-divergence, also called *density power divergence* was introduced by Basu et al. (1998) as a robust estimation method, which showed it to be consistent for parameter estimation, with asymptotic normality of the estimation error (Basu et al., 1998, Theorem 2). The full expression of $D_\beta(p||q)$ is shown in Table 1.[5] It can be seen as a *Bregman* divergence (Bregman, 1967) also derived using the generalized logarithm[2] (Eq. 1). The motivation is to obtain divergences that are robust to outliers, which is the case for values of $\beta > 1$; choosing $\beta = 2$ gives us the $L_2$-loss, while $\beta \to 1$ gives the Kullback-Leibler divergence $D_{KL}(p||q)$ as a limiting case. Hence, choosing a value close to 1 while larger is supposed to provide a compromise between the efficiency of the Kullback-Leibler divergence and the robustness of the $L_2$-loss. Similarly, we can expect to give more importance to outliers (which we suppose to be the low-frequency tokens) by choosing $\beta < 1$. The objective becomes:

$$Obj_\beta(\theta) = \frac{1}{\beta(\beta-1)} \times \sum_{(x,y)\in\mathcal{D}} \left[ (\beta-1) \sum_{y'\in\mathcal{Y}} (p_\theta(y'|x))^\beta - \beta (p_\theta(y|x))^{\beta-1} \right].$$

### 3.3 $\gamma$-divergences

Eguchi and Kato (2010) introduced the $\gamma$-divergence as a modification of the $\beta$-divergence, with the specific goal of obtaining a *scale-invariant* version of the robust $\beta$-divergence,[6] also showing it to be consistent for parameter estimation, with asymptotic normality of the estimation error (Eguchi and Kato, 2010, Section 3). This divergence has notably been used for the estimation of parameters without normalizing the output probability distribution (Takenouchi and Kanamori, 2015).[7] The general expression of the $\gamma$-divergence is shown in Table 1. While the use of the log non-linearity makes this divergence non-separable, which at first glance could be thought to complicate computation in practice, our motivation for using it is its scale-invariance, which we will discuss in Section 4.3. Applied to our prob-

[2] $\alpha$ and $\beta$-divergences are related to the *Tsallis* entropy, which can be seen as a deformation of the Shannon entropy using this same generalized logarithm: see Appendix A from Cichocki and Amari (2010) . Recently, Blondel et al. (2018) showed how to build Fenchel-Young Losses from the same Tsallis entropies.

[3] Having $p_i > 0$ will enforce $q_i > 0$.

[4] Having $p_i = 0$ will enforce $q_i = 0$.

[5] For readability, we keep the notation and values used in Cichocki and Amari (2010) instead of Basu et al. (1998).

[6] 'Scale-invariance' here means that the divergence should remain unchanged when one or both of the measures it is applied to are multiplied by scalars.

[7] However, this particular method is not suitable in our case because of the sparsity of our data.

lem, the objective becomes:

$$Obj_\gamma(\theta) = \sum_{(x,y)\in\mathcal{D}} \left[ \log p_\theta(y|x) - \frac{1}{\gamma} \log \sum_{y'\in\mathcal{Y}} (p_\theta(y'|x))^\gamma \right]. \tag{2}$$

# 4 Accelerating Learning

In order to use the previously defined objectives, we need to compute the model probabilities $p_\theta(y|x)$, which are usually obtained using a softmax function:

$$p_\theta(y|x) = \frac{\exp\left(s_\theta(x,y)\right)}{\sum_{y'\in\mathcal{Y}} \exp\left(s_\theta(x,y')\right)}$$

applied on scores (or logits) $s_\theta(x,y')$ output by the model for every possible target token $y' \in \mathcal{Y}$. However, as explained earlier, $\mathcal{Y}$ can be very large, hence computing all the scores and summing them is extremely slow. We choose to follow the strategies employed by Jozefowicz et al. (2016): approximating the softmax using importance sampling (Bengio and Sénécal, 2003, 2008), and Noise Contrastive Estimation (NCE; Gutmann and Hyvärinen 2010; Mnih and Teh 2012). Besides, the divergences presented in Section 3 can be applied to *positive measures*. Hence, a possible third direction is to instead approximate the objectives to the *un-normalized* model distributions. All the objectives presented in this section are explained in Appendix A.4.

## 4.1 Approximated softmax

Plugging the first solution into our objectives is straightforward: using self-importance sampling to approximate directly the multinomial classification probability, we compute $p_\theta$ via an *approximated softmax*:[8]

$$p_\theta(y|x) \approx \frac{\frac{\exp\left(s_\theta(x,y)\right)}{p_n(y)}}{\frac{\exp\left(s_\theta(x,y)\right)}{p_n(y)} + \sum\limits_{\substack{i=1 \\ \hat{y}_i\sim p_n}}^{k} \frac{\exp\left(s_\theta(x,\hat{y}_i)\right)}{p_n(\hat{y}_i)}}$$

where $p_n$ is an auxiliary distribution chosen to reflect the training data while still being easy to sample from, and $k \ll |\mathcal{Y}|$ is the number of samples drawn.

[8]The same objective is called 'Ranking NCE' by Ma and Collins (2018).

## 4.2 Adapting Noise-Contrastive Estimation

Noise-Contrastive Estimation consists of training the model for a surrogate binary classification task where, given examples from the mixture:

$$\frac{1}{k+1} p_\mathcal{D} + \frac{k}{k+1} p_n$$

we learn to discriminate between examples coming from data and samples coming from the auxiliary noise distribution $p_n$. Minimizing the NCE loss function has been shown to be equivalent to minimizing a Bregman divergence (Gutmann and Hirayama, 2011); however, the transformation that we need to apply to the associated generating function (Pihlaja et al., 2012) in order to obtain a power divergence is not straigthforward. Instead, with the posterior classification probability:

$$p_\theta(C = \text{True}|y,x) = \frac{p_\theta(y|x)}{p_\theta(y|x) + k p_n(y)} \tag{3}$$

Noting $p_\theta^C$ the positive measure on $\mathcal{X}\times\mathcal{Y}$:

$$p_\theta^C : (x,y) \to p_\theta(C = \text{True}|y,x)$$

we can show that the NCE objective function can be derived from the following expression:[9]

$$D_{KL}(p_\mathcal{D}^C || p_\theta^C) + D_{KL}(1 - p_\mathcal{D}^C || 1 - p_\theta^C)$$

Knowing this, we simply need to replace $D_{KL}$ by the other divergences to derive the $\alpha, \beta$ and $\gamma$ objective functions associated to this surrogate classification task. It is interesting to note that the power transformations will here be applied on the posterior classification probabilities $p_\theta^C$ instead of categorical probabilities $p_\theta$.

## 4.3 Working With Positive Measures

The three divergences presentend in Section 3 are defined on positive measures: in theory, we can simply use the exp function on the scores $s_\theta$ and do not need to normalize them:

$$Obj(\theta) = D(p_\mathcal{D} || \exp\left(s_\theta\right))$$

However, neither the $\alpha$ and $\beta$ divergences are scale invariant (see right column of Table 1 and Cichocki and Amari 2010). We can show that working with an un-normalized model distribution will, in both those cases, give an objective proportional to the scale of the model, allowing the divergence

[9]The full derivation is given in Appendix A.2.

| | Expression | Scaling properties |
|---|---|---|
| $\alpha \in \mathbb{R} \setminus \{0, 1\}$ | $\frac{1}{\alpha(\alpha-1)} \sum_{i=1}^{n} \left[p_i^\alpha q_i^{1-\alpha} - \alpha p_i + (\alpha - 1) q_i\right]$ | $D_\alpha(c \times p \| \| c \times q) = c \times D_\alpha(p \| \| q)$ |
| $\beta \in \mathbb{R} \setminus \{0, 1\}$ | $\frac{1}{\beta(\beta-1)} \sum_{i=1}^{n} \left[p_i^\beta + (\beta - 1) q_i^\beta - \beta p_i q_i^{\beta-1}\right]$ | $D_\alpha(c \times p \| \| c \times q) = c^\beta \times D_\alpha(p \| \| q)$ |
| $\gamma \in \mathbb{R} \setminus \{0, 1\}$ | $\frac{1}{\gamma(\gamma-1)} \log \sum_{i=1}^{n} p_i^\gamma + \frac{1}{\gamma} \log \sum_{i=1}^{n} q_i^\gamma - \frac{1}{\gamma-1} \log \sum_{i=1}^{n} p_i q_i^{\gamma-1}$ | $D_\gamma(c_1 \times p \| \| c_2 \times q) = D_\gamma(p \| \| q)$ |

Table 1: Complete expressions of $\alpha$, $\beta$, and $\gamma$ divergences between two positive measures $p = (p_i)_{i=1}^n$ and $q = (q_i)_{i=1}^n$ on $\mathbb{R}_+^n$, as well as their scaling properties. Note that the $\alpha$-divergence simplifies if we restrict them to the set of normalized probability densities (if $\sum_{i=1}^n p_i = \sum_{i=1}^n q_i = 1$).

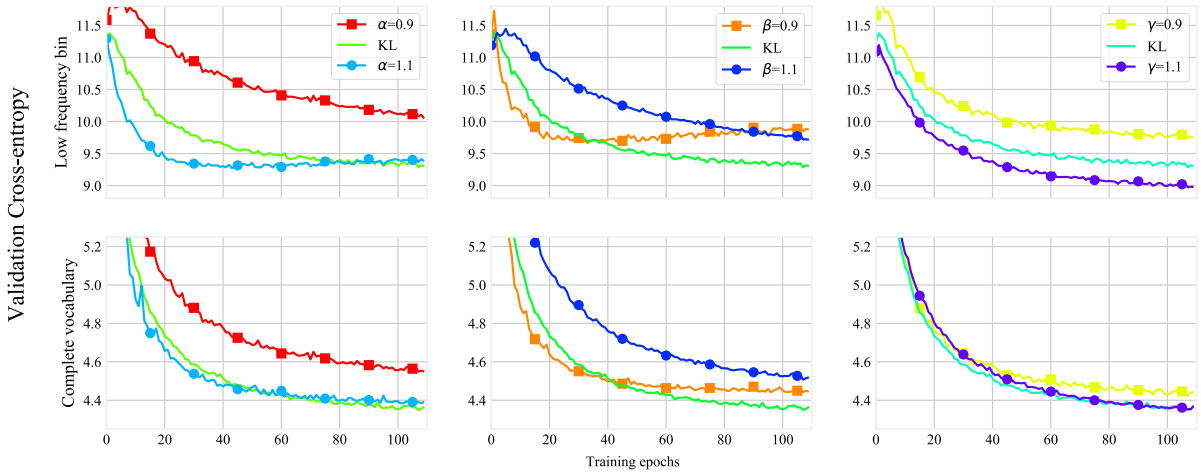


Figure 1: Validation cross-entropy values obtained during the beginning of training models with $Obj_\alpha$ (left), $Obj_\beta$ (center) and $Obj_\gamma$ (right) on the PTB with a full vocabulary. Words are grouped into 5 buckets of equal size, following their frequencies. On the top is shown the cross-entropy obtained on the bucket of lowest frequency words, while the global cross-entropy is displayed on the bottom.

to be minimized by minimizing the scale alone, with no actual learning.[10] This is not an issue for the $\gamma$-divergence, since $D_\gamma(p_\mathcal{D} || \exp(s_\theta)) = D_\gamma(p_\mathcal{D} || p_\theta)$. Hence, the objective of Eq. 2 may be simplified as following:

$$Obj_\gamma(\theta) = \sum_{(x,y)\in\mathcal{D}} \left[ s_\theta(x, y) - \frac{1}{\gamma} \log \sum_{y' \in \mathcal{Y}} \exp\left(\gamma s_\theta(x, y')\right) \right].$$

We can further accelerate learning by using importance sampling on the second term, and because of the logarithm applied to the sum, we obtain an objective that is equivalent to composing the approximated softmax with $\gamma$-divergence, as can be seen in Appendix A.4.

[10] The proof is given in Appendix A.1.

## 5 Experiments

Our goal is to compare the effect of the various objective functions we proposed in Sections 3 and 4, and especially study how the values of $\alpha$, $\beta$ and $\gamma$ affect learning and performance, overall as well as on low-frequency words. Since each model is trained with a different objective function, the training scores are not comparable. Hence, we use the validation cross-entropy and perplexity at each epoch as a way to track progress during training.

### 5.1 Datasets

We perform our experiments on two widely used, reasonably sized datasets: the Penn Treebank (PTB; Mikolov et al. 2011a) and WikiText-2 (WT2; Merity et al. 2017). The PTB, heavily preprocessed, retains the $10k$ most frequent words in its vocabulary, while the others tokens are replaced by a common $<unk>$ token. The WT2,

about two times larger, retains words that appear at least three times in the training data in its vocabulary, which makes it $33,278$ words. To study the impact of our various objective functions on a vocabulary containing very rare words, we also experiment with a version on the PTB to which we only apply limited preprocessing, allowing us to keep its full training vocabulary, which contains 39030 words.

### 5.2 Experimental setup

We based our setup on the ASGD weight-dropped LSTM (AWD-LSTM) models of Merity et al. (2018b), since to the best of our knowledge they give state-of-the-art results on the PTB and WT2 for models that are build with a softmax output layer and do not use any adaptative method (as the pointer sentinel LSTM (Merity et al., 2017), the neural cache (Grave et al., 2017b) and the dynamic evaluation (Krause et al., 2018)). Our models[11], replications of their 3-layer LSTMs with tied input and output representations, were built using their implementation.[12] For each dataset, we follow their choice of hyperparameters, only modifying the objective functions.

For our sampling-based objectives, we use $k = 1024$ samples, and the unigram distribution as $p_n$. In the case of objectives derived from binary NCE, to avoid issues with the phenomenon of *self-normalization* and consistency issues (Ma and Collins, 2018) we chose to use *blackout* (Ji et al., 2015). Indeed, this method amounts to using a noise distribution which depends on the model probabilities, making the normalization term disappear from the posterior classification probabilities of Eq. 3. Hence, we have an objective function that is fast to compute without any supplementary assumption, and the negative samples are still drawn from the unigram distribution. For both AS and NCE, in our setting, the computation time is reduced to about $40\%$ of the time taken by MLE on the WT2, and $50\%$ on the PTB.

## 6 Results and discussion

We turn to describe the results of our experiments.

[11] https://github.com/MatthieuLabeau/power-divergences-LM

[12] https://github.com/salesforce/awd-lstm-lm.

### 6.1 Qualitative results on the full vocabulary PTB

To observe the behavior of the proposed objectives for various choices of $\alpha$, $\beta$ and $\gamma$, we plot the validation cross-entropy at the beginning of learning of models on PTB equipped with a full training vocabulary. We choose values of $0.9$ and $1.1$ for the power parameters, to experiment with an objective on each side of the baseline MLE objective. We split the words according to frequency into $5$ buckets, in order for the buckets to represent have equal size based on word counts, and display both the global cross-entropy and the cross-entropy on the lowest frequency bin in Figure 1. A value of $\alpha > 1$ seems to initially behave better, especially for rare words. This could be expected: intuitively, these values of $\alpha$ should make the model 'stretch' the probability mass. However, as learning progresses, this phenomenon lessens, and the performance on rare words gets worse. A value of $\beta < 1$, supposed to make the model less robust to outliers, seems to make learning faster initially, particularly on rare words, but again improvements lessen. Choosing $\alpha < 1$ or $\beta > 1$ gives a worse cross-entropy overall. This 'inverted' similarity between the behaviors induced by choices of $\alpha$ and $\beta$, and the links between the two divergences, have been explored by Patra et al. (2013). Finally, choosing $\gamma > 1$ gives very interesting results, allowing for a better cross-entropy on rare words while retaining the same overall performance than MLE.

### 6.2 Penn Treebank and WikiText-2

In this section, we present the results of exploratory experiments. We fully train models with the proposed objectives, for a variety of power parameters. For the PTB, the final validation perplexities are presented in Table 2, while we present the final validation cross-entropies for the objectives derived from MLE, on $5$ frequency buckets, in Figure 2. For the WT2, they are shown in Table 3 and Figure 3.

**With exact objectives:** for both corpora, the results for the high and low-frequency buckets seem to confirm that values of $\alpha, \gamma$ that are smaller and larger than $1$ can help prioritizing learning towards the frequent and rare words. The effect of $\beta$ depending on frequency seems lighter, especially on the PTB: we hypothesize that this is due to the vocabulary being cut short, and containing no very

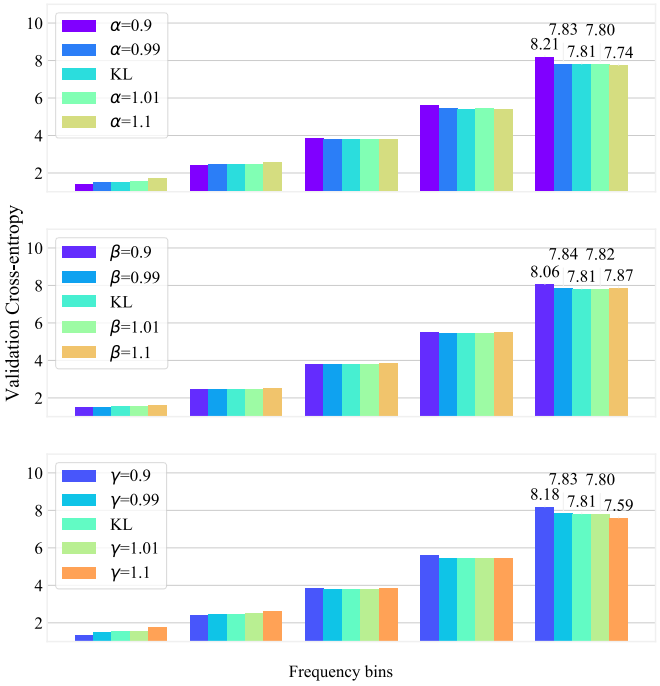


Figure 2: Validation cross-entropy values for the best epoch obtained for models trained with $Obj_\alpha$ (top), $Obj_\beta$ (middle) and $Obj_\gamma$ (bottom) on the PTB. Words are grouped into 5 buckets of equal size, following their frequencies. We display values for each bucket from the most frequent words (left) to less frequent ones (right).

| Objective | | $Obj_\alpha$ | $Obj_\beta$ | $Obj_\gamma$ |
|---|---|---|---|---|
| MLE | 0.9 | 65.8 | 63.3 | 62.3 |
| | 0.99 | 60.7 | 61.0 | 60.6 |
| | 1.0 | | 60.1 | |
| | 1.01 | 60.8 | 61.1 | 61.0 |
| | 1.1 | 63.0 | 63.9 | 63.3 |
| AS | 0.5 | 147.0 | 87.1 | 113.8 |
| | 0.9 | 61.3 | 62.8 | 61.0 |
| | 0.99 | 61.7 | 61.7 | 61.6 |
| | 1.0 | | 61.7 | |
| | 1.01 | 61.8 | 61.6 | 62.0 |
| | 1.1 | 65.6 | 61.6 | 64.1 |
| | 1.5 | 97.7 | 127.6 | 81.7 |
| NCE | 0.5 | 1407.5 | 88.0 | 117.2 |
| | 0.9 | 61.4 | 62.6 | 61.8 |
| | 0.99 | 61.1 | 61.4 | 61.7 |
| | 1.0 | | 61.2 | |
| | 1.01 | 61.3 | 61.1 | 61.8 |
| | 1.1 | 64.4 | 61.0 | 64.4 |
| | 1.5 | 97.9 | 123.7 | 81.2 |

Table 2: Best validation perplexities obtained on the PTB with $Obj_\alpha$, $Obj_\beta$, and $Obj_\gamma$, derived from MLE, and approximated objectives AS and NCE, on single models with the same intialization.

| Objective | $Obj_\alpha$ | | $Obj_\beta$ | | $Obj_\gamma$ | |
|---|---|---|---|---|---|---|
| Dataset | PTB | WT2 | PTB | WT2 | PTB | WT2 |
| 0.9 | 65.8 | 75.8 | 63.3 | 68.3 | 63.2 | 70.1 |
| | 39.7 | 42.4 | 107.9 | 130.0 | 62.4 | 69.0 |
| 0.99 | 60.7 | 66.5 | 61.0 | 65.6 | 60.6 | 65.9 |
| | 57.9 | 63.0 | 64.0 | 69.3 | 60.6 | 65.9 |
| 1.0 | 60.1 | 64.9 | 60.1 | 64.9 | 60.1 | 64.9 |
| 1.01 | 60.8 | 65.6 | 61.1 | 65.4 | 61.0 | 66.0 |
| | 63.8 | 69.3 | 58.3 | 61.8 | 61.0 | 66.0 |
| 1.1 | 63.0 | 66.7 | 63.9 | 68.3 | 63.3 | 68.1 |
| | 99.8 | 114.3 | 40.7 | 45.2 | 62.5 | 67.0 |

Table 4: Final validation results obtained on the PTB and WT2, with the exact objectives $Obj_\alpha$, $Obj_\beta$, and $Obj_\gamma$, derived from MLE. In each cell we give on top the validation perplexity, and below, the 'counterpart' to perplexity corresponding to the training objective — which is the value being optimized. Each color corresponds to a different objective: values cannot be compared for varying $\alpha$, $\beta$ and $\gamma$.

rare words, which reduces the discrepancies between higher and lower scoring examples. However, no objective seems to be able to improve on MLE for overall performance.

To explain these results, we may argue that perplexity is a biased measure as MLE directly optimizes it. Hence, we compute the 'counterparts' to perplexity corresponding to each objective. Equivalently to perplexity for MLE, they are directly optimized by their respective objectives, and should decrease to 1 as the model distribution gets closer to the data distribution. Again, since they vary for each objective, they are not comparable between themselves and with perplexity. We display these values for our models trained with exact objectives in Table 4. As $\alpha$, $\beta$ and $\gamma$ are closer to 1, these values get closer to the perplexity of the model. Besides, they are especially close across all values of $\gamma$: we can assume that this indicates that the corresponding $Obj_\gamma$ are 'closer' to the MLE objective. However, tracking these values during training[13] shows that they all behave very similarly to perplexity.

**With approximated objectives:** for objectives derived from AS and NCE, we observe far less impact of the choice of the power parameter on frequent or rare words,[14] which is probably due to the fact that only a small subset of the vocab-

[13] See Appendix A.3.

[14] See figures in Appendix A.5.

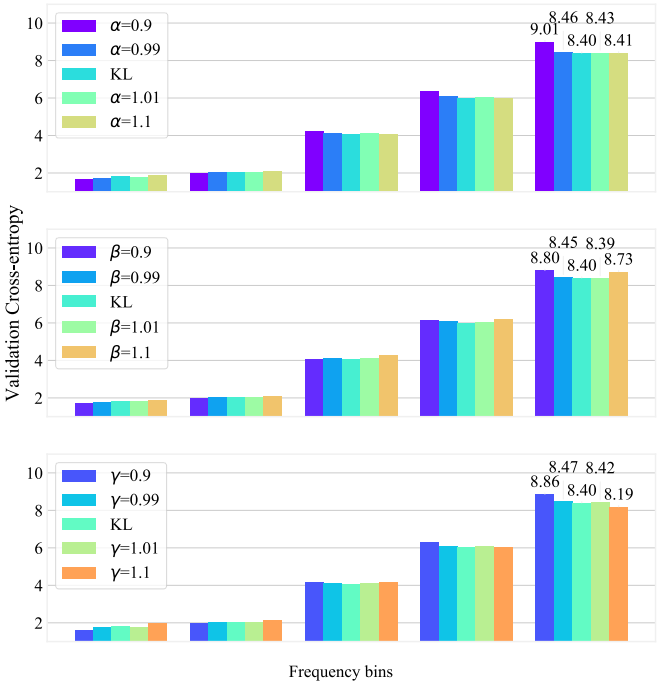


Figure 3: Validation cross-entropy values for the best epoch obtained for models trained with $Obj_\alpha$ (top), $Obj_\beta$ (middle) and $Obj_\gamma$ (bottom) on the WT2. Words are grouped into 5 buckets of equal size, following their frequencies. We display values for each bucket from the most frequent words (left) to less frequent ones (right).

| Objective | | $Obj_\alpha$ | $Obj_\beta$ | $Obj_\gamma$ |
|---|---|---|---|---|
| MLE | 0.9 | 75.8 | 68.3 | 70.1 |
| | 0.99 | 66.5 | 65.6 | 65.9 |
| | 1.0 | | 64.9 | |
| | 1.01 | 65.6 | 65.4 | 66.0 |
| | 1.1 | 66.7 | 68.3 | 68.1 |
| AS | 0.5 | 203.6 | 82.1 | 191.6 |
| | 0.9 | 69.9 | 65.6 | 68.0 |
| | 0.99 | 67.4 | 65.0 | 65.1 |
| | 1.0 | | 65.3 | |
| | 1.01 | 67.5 | 65.1 | 65.0 |
| | 1.1 | 69.8 | 67.8 | 64.8 |
| | 1.5 | 100.7 | 211.1 | 78.1 |
| NCE | 0.5 | 2401.2 | 84.1 | 193.1 |
| | 0.9 | 70.6 | 66.6 | 69.7 |
| | 0.99 | 66.8 | 65.7 | 67.5 |
| | 1.0 | | 65.8 | |
| | 1.01 | 65.8 | 65.9 | 65.8 |
| | 1.1 | 64.7 | 68.3 | 65.4 |
| | 1.5 | 83.8 | 196.4 | 77.3 |

Table 3: Best validation perplexities obtained on the WT2 with $Obj_\alpha$, $Obj_\beta$, and $Obj_\gamma$, derived from MLE, and approximated objectives AS and NCE, on single models with the same initialization.

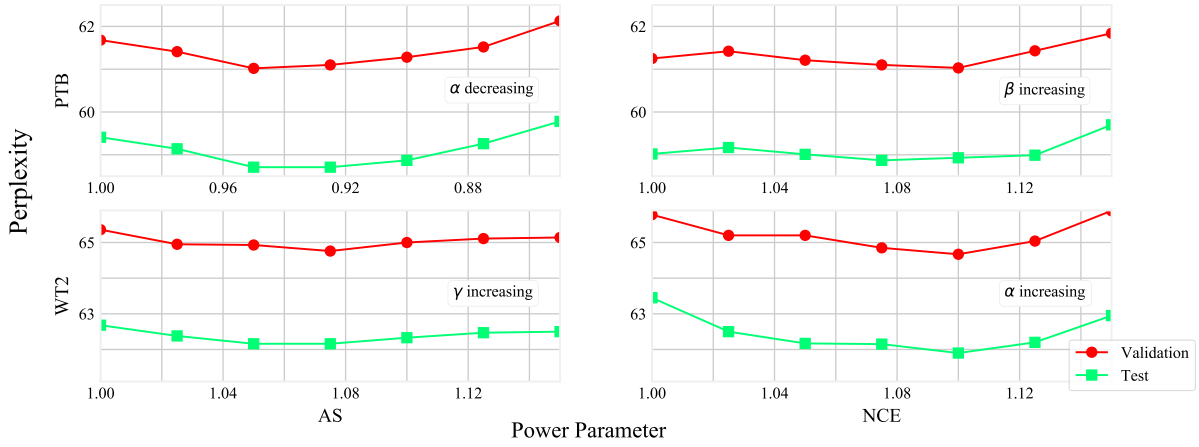


Figure 4: Validation and test perplexities obtained for particular values of $\alpha$, $\beta$ or $\gamma$ with sampling-based objectives in 4 possible pairings of AS and NCE-based objectives trained on the PTB and WT2, on single models with the same initialization.

ulary is updated during learning — and this subset is drawn according to word frequency. However, some objectives provide modest improvements over the corresponding KL-based baseline.

### 6.3 Searching for the Optimal Power Parameter

In order to verify the potential benefits of our generalization of sampling-based objectives, we use these preliminary results to search for the 'best' power parameter, and check that improvements are consistent for several versions of the model,

| | | Objective | Validation | Test |
|---|---|---|---|---|
| PTB | | MLE | 60.7 | 58.6 |
| | AS | KL | 62.2 | 59.8 |
| | | $\alpha = 0.95$ | 61.2 | 58.9 |
| | NCE | KL | 61.5 | 59.2 |
| | | $\beta = 1.1$ | 61.2 | 59.0 |
| WT2 | | MLE | 65.5 | 62.8 |
| | AS | KL | 65.1 | 62.6 |
| | | $\gamma = 1.075$ | 64.8 | 62.1 |
| | NCE | KL | 65.8 | 63.4 |
| | | $\alpha = 1.1$ | 64.7 | 62.0 |

Table 5: Best validation and test perplexities obtained on the best performing configurations of power parameter for both corpora and each category of objectives, averaged over 5 models with different initializations.

initialized with different seeds. This search is shown in Figure 4 for 4 different configurations chosen using the preliminary results presented earlier. In each case, we can see that the perplexity seems to decrease, reach a minimum, and then increase monotonously. While verifying this would require a deeper investigation, we can make the hypothesis that the sampling procedure induces noise that some of our proposed objectives can be better suited for. In Table 5, we present perplexity results averaged for different seeds, in order to check that model initialization does not discernibly affect these improvements. They show gains of up to 1 point in perplexity, and a slight improvement over the MLE baseline for WT2, confirming the results obtained on a single model. However, it is tedious to find the optimal power parameter for a given configuration, and we can expect it to shift sensibly when hyper-parameters are modified. The size of the vocabulary and the distribution of word frequencies should intuitively have the biggest influence, as we can see that the behavior of $\alpha$, $\beta$ and $\gamma$ is quite different on the PTB and WT2. In general, finding the optimal divergence with the optimal parameter for a given problem is very difficult. Methods for doing so have only been recently explored; for example, Dikmen et al. (2015) use the Tweedie Likelihood to find the optimal $\beta$, which can lead to the optimal $\alpha$ and $\gamma$.

## 7 Conclusion

We explore the use of generalizations of KL-divergence into power ($\alpha$, $\beta$ and $\gamma$) divergences for training language models. We derive exact but also approximated objectives, based on an approximated softmax using importance sampling, and noise contrastive estimation. We show that in the case of exact objectives, a well-chosen $\gamma$-divergence can be used to prioritize learning low or high-frequency words. In the case of approximated objective, we show that our proposed objectives can improve on perplexity, and demonstrate it is the case for several configurations. Further research should investigate the potential gains of using $\gamma$-divergences for appropriate downstream tasks.

## Acknowledgments

We thank the anonymous reviewers for helpful feedback. We gratefully acknowledge the support of Huawei Technologies.

## References

S. M. Ali and S. D. Silvey. 1966. A general class of coefficients of divergence of one distribution from another. *Journal of the Royal Statistical Society. Series B (Methodological)*, 28(1):131–142.

Alexei Baevski and Michael Auli. 2019. Adaptive input representations for neural language modeling. In *International Conference on Learning Representations*.

MichèLe Basseville. 2013. Review: Divergence measures for statistical data processing-an annotated bibliography. *Signal Process.*, 93(4):621–633.

Ayanendranath Basu, Ian R. Harris, Nils L. Hjort, and M. C. Jones. 1998. Robust and efficient estimation by minimising a density power divergence. *Biometrika*, 85(3):549–559.

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. 2001. A neural probabilistic language model. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 932–938. MIT Press.

Yoshua Bengio and Jean-Sébastien Sénécal. 2003. Quick training of probabilistic neural nets by importance sampling. In *Proceedings of the conference on Artificial Intelligence and Statistics (AISTATS)*.

Yoshua Bengio and Jean-Sébastien Sénécal. 2008. Adaptive importance sampling to accelerate training of a neural probabilistic language model. *IEEE Trans. Neural Networks*, 19(4):713–722.

Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Comput. Linguist.*, 22(1):39–71.

Mathieu Blondel, André F. T. Martins, and Vlad Niculae. 2018. Learning classifiers with fenchel-young losses: Generalized entropies, margins, and algorithms. Cite arxiv:1805.09717.

L.M. Bregman. 1967. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, 7(3):200 – 217.

Wenlin Chen, David Grangier, and Michael Auli. 2016. Strategies for training large vocabulary neural language models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1975–1985, Berlin, Germany. Association for Computational Linguistics.

Andrzej Cichocki and Shun Amari. 2010. Families of alpha- beta- and gamma- divergences: Flexible and robust measures of similarities. *Entropy*, 12(6):1532–1568.

I. Csiszar. 1967. Information-type measures of difference of probability distributions and indirect observation. *Studia Scientiarum Mathematicarum Hungarica*, 2:229–318.

Onur Dikmen, Zhirong Yang, and Erkki Oja. 2015. Learning the information divergence. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(7):1442–1454.

Shinto Eguchi and Shogo Kato. 2010. Entropy and divergence associated with power function and the statistical application. *Entropy*, 12(2):262–274.

Chengyue Gong, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2018. Frage: Frequency-agnostic word representation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 1334–1345. Curran Associates, Inc.

Joshua T. Goodman. 2001. A bit of progress in language modeling. *Comput. Speech Lang.*, 15(4):403–434.

Edouard Grave, Armand Joulin, Moustapha Cissé, David Grangier, and Hervé Jégou. 2017a. Efficient softmax approximation for gpus. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 1302–1310.

Edouard Grave, Armand Joulin, and Nicolas Usunier. 2017b. Improving neural language models with a continuous cache. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Michael Gutmann and Junichiro Hirayama. 2011. Bregman divergence as general framework to estimate unnormalized statistical models. In *UAI*.

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, pages 297–304.

Shihao Ji, S. V. N. Vishwanathan, Nadathur Satish, Michael J. Anderson, and Pradeep Dubey. 2015. Blackout: Speeding up recurrent neural network language models with very large vocabularies. *CoRR*, abs/1511.06909.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling.

Amor Keziou. 2003. Dual representation of phi-divergences and applications. *Comptes Rendus Mathematique*, 336(10):857 – 862.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *30th AAAI Conference on Artificial Intelligence, AAAI 2016*, pages 2741–2749. AAAI press.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, volume I, pages 181–184, Detroit, Michigan.

Ben Krause, Emmanuel Kahembwe, Iain Murray, and Steve Renals. 2018. Dynamic evaluation of neural sequence models. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2766–2775, Stockholmsmässan, Stockholm Sweden. PMLR.

Zhuang Ma and Michael Collins. 2018. Noise contrastive estimation and negative sampling for conditional models: Consistency and statistical efficiency. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3698–3707. Association for Computational Linguistics.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018a. An analysis of neural language modeling at multiple scales. *CoRR*, abs/1803.08240.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018b. Regularizing and optimizing LSTM language models. In *International Conference on Learning Representations*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Tomas Mikolov, , Stefan Kombrink, Lukas Burget, and Jan Honza Cernocky. 2011a. Empirical evaluation and combination of advanced language modeling techniques. In *Interspeech*. ISCA.

Tomas Mikolov, Stefan Kombrink, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2011b. Extensions of recurrent neural network language model. In *ICASSP*, pages 5528–5531. IEEE.

Thomas Minka. 2005. Divergence measures and message passing. Technical report.

Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*.

Sujayendu Patra, Avijit Maji, Ayanendranath Basu, and Leandro Pardo. 2013. The power divergence and the density power divergence families: The mathematical connection. *Sankhya B*, 75.

Miika Pihlaja, Michael Gutmann, and Aapo Hyvärinen. 2012. A family of computationally efficient and simple estimators for unnormalized statistical models. *CoRR*, abs/1203.3506.

David M. W. Powers. 1998. Applications and explanations of Zipf's law. In *New Methods in Language Processing and Computational Natural Language Learning*.

Alfréd Rényi. 1961. On measures of entropy and information. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*, pages 547–561, Berkeley, Calif. University of California Press.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Takashi Takenouchi and Takafumi Kanamori. 2015. Empirical localization of homogeneous divergences on discrete sample spaces. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 820–828. Curran Associates, Inc.

Vincent Van Asch and Walter Daelemans. 2010. Using domain similarity for performance estimation. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, DANLP 2010, pages 31–36, Stroudsburg, PA, USA. Association for Computational Linguistics.

# Hierarchically-Refined Label Attention Network for Sequence Labeling

**Leyang Cui**[1,2,3] and **Yue Zhang**[2,3]
[1]Zhejiang University
[2]School of Engineering, Westlake University
[3]Institute of Advanced Technology, Westlake Institute for Advanced Study
cuileyang@westlake.edu.cn, yue.zhang@wias.org.cn

## Abstract

CRF has been used as a powerful model for statistical sequence labeling. For neural sequence labeling, however, BiLSTM-CRF does not always lead to better results compared with BiLSTM-softmax local classification. This can be because the simple Markov label transition model of CRF does not give much information gain over strong neural encoding. For better representing label sequences, we investigate a hierarchically-refined label attention network, which explicitly leverages label embeddings and captures potential long-term label dependency by giving each word incrementally refined label distributions with hierarchical attention. Results on POS tagging, NER and CCG supertagging show that the proposed model not only improves the overall tagging accuracy with similar number of parameters, but also significantly speeds up the training and testing compared to BiLSTM-CRF.

## 1 Introduction

Conditional random fields (CRF) (Lafferty et al., 2001) is a state-of-the-art model for statistical sequence labeling (Toutanova et al., 2003; Peng et al., 2004; Ratinov and Roth, 2009). Recently, CRF has been integrated with neural encoders as an output layer to capture label transition patterns (Zhou and Xu, 2015; Ma and Hovy, 2016). This, however, sees mixed results. For example, previous work (Reimers and Gurevych, 2017; Yang et al., 2018) has shown that BiLSTM-softmax gives better accuracies compared to BiLSTM-CRF for part-of-speech (POS) tagging. In addition, the state-of-the-art neural Combinatory Categorial Grammar (CCG) supertaggers do not use CRF (Xu et al., 2015; Lewis et al., 2016).

One possible reason is that the strong representation power of neural sentence encoders such as BiLSTMs allow models to capture implicit long-range label dependencies from input word sequences alone (Kiperwasser and Goldberg, 2016; Dozat and Manning, 2016; Teng and Zhang, 2018), thereby allowing the output layer to make local predictions. In contrast, though explicitly capturing output label dependencies, CRF can be limited by its Markov assumptions, particularly when being used on top of neural encoders. In addition, CRF can be computationally expensive when the number of labels is large, due to the use of Viterbi decoding.

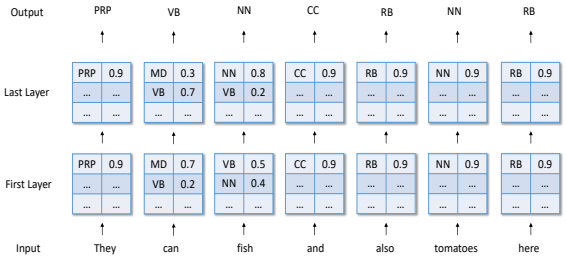


Figure 1: Visualization of hierarchically-refined Label Attention Network for POS tagging. The numbers indicate the label probability distribution for each word.

One interesting research question is whether there is neural alternative to CRF for neural sequence labeling, which is both faster and more accurate. To this question, we investigate a neural network model for output label sequences. In particular, we represent each possible label using an embedding vector, and aim to encode sequences of label distributions using a recurrent neural network. One main challenge, however, is that the number of possible label sequences is exponential to the size of input. This makes our task essentially to represent a full-exponential search space without making Markov assumptions.

We tackle this challenge using a hierarchically-refined representation of marginal label distributions. As shown in Figure 1, our model consists of a multi-layer neural network. In each layer, each input words is represented together with its



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4115–4128,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

marginal label probabilities, and a sequence neural network is employed to model unbounded dependencies. The marginal distributions space are refined hierarchically bottom-up, where a higher layer learns a more informed label sequence distribution based on information from a lower layer.

For instance, given a sentence "They$_1$ can$_2$ fish$_3$ and$_4$ also$_5$ tomatoes$_6$ here$_7$", the label distributions of the words *can*$_2$ and *fish*$_3$ in the first layer of Figure 1 have higher probabilities on the tags MD (modal verb) and VB (base form verb), respectively, though not fully confidently. The initial label distributions are then fed as the inputs to the next layer, so that long-range label dependencies can be considered. In the second layer, the network can learn to assign a noun tag to *fish*$_3$ by taking into account the highly confident tagging information of *tomatoes*$_6$ *(NN)*, resulting in the pattern "*can*$_2$ *(VB) fish*$_3$ *(NN)*".

As shown in Figure 2, our model consists of stacked attentive BiLSTM layers, each of which takes a sequence of vectors as input and yields a sequence of hidden state vectors together with a sequence of label distributions. The model performs attention over label embeddings (Wang et al., 2015; Zhang et al., 2018a) for deriving a marginal label distributions, which are in turn used to calculate a weighted sum of label embeddings. Finally, the resulting packed label vector is used together with input word vectors as the hidden state vector for the current layer. Thus our model is named *label attention network* (LAN). For sequence labeling, the input to the whole model is a sentence and the output is the label distributions of each word in the final layer.

BiLSTM-LAN can be viewed as a form of multi-layered BiLSTM-softmax sequence labeler. In particular, a single-layer BiLSTM-LAN is identical to a single-layer BiLSTM-softmax model, where the label embedding table serves as the softmax weights in BiLSTM-softmax, and the label attention distribution is the softmax distribution in BiLSTM-softmax. The traditional way of making a multi-layer extention to BiLSTM-softmax is to stack multiple BiLSTM encoder layers before the softmax output layer, which learns a deeper input representation. In contrast, a multi-layer BiLSTM-LAN stacks both the BiLSTM encoder layer and the softmax output layer, learning a deeper representation of both the input and candidate output sequences.

On standard benchmarks for POS tagging, NER and CCG supertagging, our model achieves significantly better accuracies and higher efficiencies than BiLSTM-CRF and BiLSTM-softmax with similar number of parameters. It gives highly competitive results compared with top-performance systems on WSJ, OntoNotes 5.0 and CCGBank without external training. In addition to accuracy and efficiency, BiLSTM-LAN is also more interpretable than BiLSTM-CRF thanks to visualizable label embeddings and label distributions. Our code and models are released at `https://github.com/Nealcly/LAN`.

## 2 Related Work

**Neural Attention.** Attention has been shown useful in neural machine translation (Bahdanau et al., 2014), sentiment classification (Chen et al., 2017; Liu and Zhang, 2017), relation classification (Zhou et al., 2016), read comprehension (Hermann et al., 2015), sentence summarization (Rush et al., 2015), parsing (Li et al., 2016), question answering (Wang et al., 2018b) and text understanding (Kadlec et al., 2016). Self-attention network (SAN) (Vaswani et al., 2017) has been used for semantic role labeling (Strubell et al., 2018), text classification (Xu et al., 2018; Wu et al., 2018) and other tasks. Our work is similar to Vaswani et al. (2017) in the sense that we also build a hierarchical attentive neural network for sequence representation. The difference lies in that our main goal is to investigate the encoding of exponential *label* sequences, whereas their work focuses on encoding of a *word* sequence only.

**Label Embeddings.** Label embedding was first used in the field of computer vision for facilitating zero-shot learning (Palatucci et al., 2009; Socher et al., 2013; Zhang and Saligrama, 2016). The basic idea is to improve the performance of classifying previously unseen class instances by learning output label knowledge. In NLP, label embeddings have been exploited for better text classification (Tang et al., 2015; Nam et al., 2016; Wang et al., 2018a). However, relatively little work has been done investigating label embeddings for sequence labeling. One exception is Vaswani et al. (2016b), who use supertag embeddings in the output layer of a CCG supertagger, through a combination of local classification model rescored using a supertag language model. In contrast, we model deep label interactions using a dynamically refined

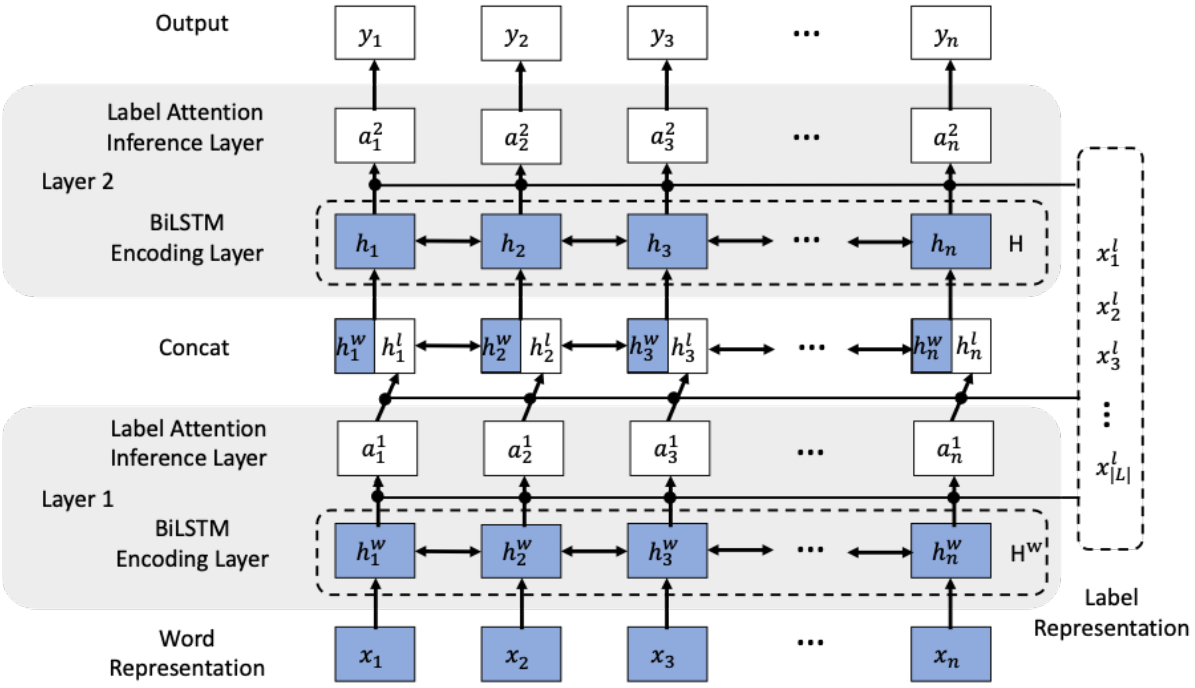


Figure 2: Architecture of hierarchically-refined label attention network.

sequence representation network. To our knowledge, we are the first to investigate a hierarchical attention network over a label space.

**Neural CRF**. There has been methods that aim to speed up neural CRF (Tu and Gimpel, 2018), and to solve the Markov constraint of neural CRF. In particular, Zhang et al. (2018b) predicts a sequence of labels as a sequence to sequence problem; Guo et al. (2019) further integrates global input information in encoding. Capturing non-local dependencies between labels, these methods, however, are slower compared with CRF. In contrast to these lines of work, our method is both asymptotically faster and empirically more accurate compared with neural CRF.

## 3 Baseline

We implement BiLSTM-CRF and BiLSTM-softmax baseline models with character-level features (Dos Santos and Zadrozny, 2014; Lample et al., 2016), which consist of a word representation layer, a sequence representation layer and a local softmax or CRF inference layer.

### 3.1 Word Representation Layer

Following Dos Santos and Zadrozny (2014) and Lample et al. (2016), we use character information for POS tagging. Given a word sequence $w_1, w_2, ..., w_n$ , where $c_{ij}$ denotes the $j$th character in the $i$th word, each $c_{ij}$ is represented using

$$\mathbf{x}^c_{ij} = \mathbf{e}^c(c_{ij})$$

where $\mathbf{e}^c$ denotes a character embedding lookup table.

We adopt BiLSTM for character encoding. $x^c_i$ denotes the output of character-level encoding.

A word is represented by concatenating its word embedding and its character representation:

$$\mathbf{x}_i = [\mathbf{e}^w(w_i); \mathbf{x}^c_i]$$

where $\mathbf{e}^w$ denotes a word embedding lookup table.

### 3.2 Sequence Representation Layer

For sequence encoding, the input is a sentence $\mathbf{x} = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$. Word representations are fed into a BiLSTM layer, yielding a sequence of forward hidden states $\{\overrightarrow{\mathbf{h}^w_1}, \cdots, \overrightarrow{\mathbf{h}^w_n}\}$ and a sequence of backward hidden states $\{\overleftarrow{\mathbf{h}^w_1}, \cdots, \overleftarrow{\mathbf{h}^w_n}\}$, respectively. Finally, the two hidden states are concatenated for a final representation

$$\mathbf{h}^w_i = [\overrightarrow{\mathbf{h}^w_i}; \overleftarrow{\mathbf{h}^w_i}]$$
$$\mathbf{H}^w = \{\mathbf{h}^w_1, \cdots, \mathbf{h}^w_n\}$$

### 3.3 Inference Layer

**CRF.** A CRF layer is used on top of the hidden vectors $\mathbf{H}^w$. The conditional probabilities of label distribution sequences $y = \{y_1, \cdots, y_n\}$ is

$$P(y|x) = \frac{\exp(\sum_i (\mathbf{W}_{CRF}^{l_i} \mathbf{h_i}^w + b_{CRF}^{(l_{i-1}, l_i)}))}{\sum_{y'} \exp(\sum_i (\mathbf{W}_{CRF}^{l_i'} \mathbf{h_i}^w + b_{CRF}^{(l'_{i-1}, l_i')}))}$$

Here $y'$ represents an arbitrary label distribution sequence, $\mathbf{W}_{CRF}^{l_i}$ is a model parameter specific to $l_i$, and $b_{CRF}^{(l'_{i-1}, l_i')}$ is a bias specific to $l_{i-1}$ and $l_i$.

The first-order Viterbi algorithm is used to find the highest scored label sequence over an input word sequence during decoding.

**Softmax.** Independent local softmax classification can give competitive result on sequence labeling (Reimers and Gurevych, 2017). For each node, $\mathbf{h}_i^w$ is fed to a softmax layer to find

$$\hat{y}_i = \text{softmax}(\mathbf{W}\mathbf{h}_i^w + \mathbf{b}) \quad (1)$$

where $\hat{y}_i$ is the predicted label for $w_i$; $\mathbf{W}$ and $\mathbf{b}$ are the parameters for softmax layer.

## 4 Label Attention Network

The structure of our refined label attention network is shown in Figure 2. We denote our model as BiLSTM-LAN (namely BiLSTM-label attention network) for the remaining of this paper. We use the same input word representations for BiLSTM-softmax, BiLSTM-CRF and BiLSTM-LAN, as described in Section 3.1. Compared with the baseline models, BiLSTM-LAN uses a set of BiLSTM-LAN layers for both encoding and label prediction, instead of a set of traditional sequence encoding layers and an inference layer.

### 4.1 Label Representation.

Given the set of candidates output labels $L = \{l_1, \cdots, l_{|L|}\}$, each label $l_k$ is represented using an embedding vector:

$$\mathbf{x}_k^l = \mathbf{e}^l(l_k)$$

where $\mathbf{e}^l$ denotes a label embedding lookup table. Label embeddings are randomly initialized and tuned during model training.

### 4.2 BiLSTM-LAN Layer

The model in Figure 2 consists of 2 BiLSTM-LAN layers. As discussed earlier, each BiLSTM-LAN layer is composed of a BiLSTM encoding sublayer and a label-attention inference sublayer. In paticular, the former is the same as the BiLSTM layer in the baseline model, while the latter uses multi-head attention (Vaswani et al., 2017) to jointly encode information from the word representation subspace and the label representation subspace. For each head, we apply a dot-product attention with a scaling factor to the inference component, deriving label distributions for BiLSTM hidden states and label embeddings.

**BiLSTM Encoding Sublayer.** Denote the input to each layer as $\mathbf{x} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$. BiLSTM (Section 3.2) is used to calculate $\mathbf{H}^w \in \mathbb{R}^{n \times d_h}$, where $n$ and $d_h$ denote the word sequence length and BiLSTM hidden size (same as the dimension of label embedding), respectively.

**Label-Attention Inference Sublayer.** For the label-attention inference sublayer, the attention mechanism produces an attention matrix $\boldsymbol{\alpha}$ consisting of a potential label distribution for each word. We define $\mathbf{Q} = \mathbf{H}^w$, $\mathbf{K} = \mathbf{V} = \mathbf{x}^l$. $\mathbf{x}^l \in \mathbb{R}^{|L| \times d_h}$ is the label set representation, where $|L|$ is the total number of labels. As shown in Figure 2, outputs are calculated by

$$\mathbf{H}^l = \text{attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \boldsymbol{\alpha}\mathbf{V}$$
$$\boldsymbol{\alpha} = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_h}})$$

Instead of the standard attention mechanism above, it can be beneficial to use multi-head for capturing multiple possible of potential label distributions in parallel.

$$\mathbf{H}^l = \text{concat}(\text{head}, \ldots, \text{head}_k) + \mathbf{H}^w$$
$$\text{head}_i = \text{attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V)$$

where $\mathbf{W}_i^Q \in \mathbb{R}^{d_h \times \frac{d_h}{k}}$, $\mathbf{W}_i^K \in \mathbb{R}^{d_h \times \frac{d_h}{k}}$ and $\mathbf{W}_i^V \in \mathbb{R}^{d_h \times \frac{d_h}{k}}$ are parameters to be learned during the training, $k$ is the number of parallel heads.

The final representation for each BiLSTM-LAN layer is the concatenation of BiLSTM hidden states and attention outputs :

$$\mathbf{H} = [\mathbf{H}^w; \mathbf{H}^l]$$

$\mathbf{H}$ is then fed to a subsequent BiLSTM-LAN layer as input, if any.

**Output.** In the last layer, BiLSTM-LAN directly predicts the label of each word based on the attention weights.

$$\begin{bmatrix} \hat{y_1}^1 \cdots \hat{y_1}^{|L|} \\ \cdots\cdots \\ \cdots\cdots \\ \cdots\cdots \\ \hat{y_i}^1 \cdots \hat{y_i}^{|L|} \end{bmatrix} = \boldsymbol{\alpha}$$

$$\hat{y_i} = \text{argmax}_j(\hat{y_i}^1, ..., \hat{y_i}^n)$$

where $\hat{y_i}^j$ denotes the $j$th candidate label for the $i$th word and $\hat{y_i}$ is the predicted label for $i$th word in the sequence.

### 4.3 Training

BiLSTM-LAN can be trained by standard back-propagation using the log-likelihood loss. The training object is to minimize the cross-entropy between $y_i$ and $\hat{y_i}$ for all labeled gold-standard sentences. For each sentence,

$$L = -\sum_i \sum_j y_i^j \log \hat{y}_i^j$$

where $i$ is the word index, $j$ is the index of labels for each word.

### 4.4 Complexity

For decoding, the asymptotic time complexity is $O(|L|^2 n)$ and $O(|L|n)$ for BiLSTM-CRF and BiLSTM-LAN, respectively, where $|L|$ is the total number of labels and $n$ is the sequence length. Compared with BiLSTM-CRF, BiLSTM-LAN can significantly increase the speed for sequence labeling, especially for CCG supertagging, where the sequence length can be much smaller than the total number of labels.

### 4.5 BiLSTM-LAN and BiLSTM-softmax

As mentioned in the introduction, a single-layer BiLSTM-LAN is identical to a single-layer BiLSTM-softmax sequence labeling model. In particular, the BiLSTM-softmax model is given by Eq 1. In a BiLSTM-LAN model, we arrange the set of label embeddings into a matrix as follows:

$$\mathbf{x}^l = [x_1^l; x_2^l; ..., x_{|L|}^l]$$

A naive attention model over X has:

$$\boldsymbol{\alpha} = \text{softmax}(\mathbf{H}\mathbf{x}^l)$$

It is thus straightforward to see that the label embedding matrix $\mathbf{x}^l$ corresponds to the weight matrix $\mathbf{W}$ is Eq 1, and the distribution $\boldsymbol{\alpha}$ corresponds to y in Eq 1.

## 5 Experiments

We empirically compare BiLSTM-CRF, BiLSTM-softmax and BiLSTM-LAN using different sequence labelling tasks, including English POS tagging, multilingual POS tagging, NER and CCG supertagging.

| Data | | training | dev | test |
|---|---|---|---|---|
| **WSJ** | #l | 45 | 45 | 45 |
| | #s | 38,219 | 5,527 | 5,462 |
| | #t | 912,344 | 131,768 | 129,654 |
| **UD_en** | #l | 50 | 50 | 49 |
| | #s | 12,544 | 2,003 | 2,078 |
| | #t | 204,607 | 25,150 | 25,097 |
| **OntoNotes** | #l | 18 | 18 | 18 |
| | #s | 59,924 | 8,528 | 8,262 |
| | #t | 1,088,503 | 147,724 | 152,728 |
| **CCGBank** | #l | 426 | 323 | 348 |
| | #s | 39,604 | 1,913 | 2,407 |
| | #t | 929,552 | 45,422 | 55,371 |

Table 1: Data statistics. l:label, s:sentence, t:tokens.

| Model | # E/# H | # L | Acc | # Param |
|---|---|---|---|---|
| BiLSTM-CRF | 200 | 1 | 97.56 | 5.1M |
| | 400 | 1 | 97.57 | 5.5M |
| | 400 | 2 | 97.57 | 6.4M |
| | 400 | 3 | 97.52 | 7.4M |
| | 600 | 2 | 97.57 | 8.2M |
| | 600 | 3 | 97.50 | 10.4M |
| BiLSTM-LAN | 200 | 3 | 97.53 | 5.7M |
| | 400 | 2 | 97.57 | 8.1M |
| | 400 | 3 | 97.63 | 10.0M |
| | 400 | 4 | 97.60 | 12.2M |
| | 600 | 3 | 97.62 | 16.5M |

Table 2: WSJ development set. E: label embedding size, H: hidden size, L: number of layers.

| Model | Accuracy (%) |
|---|---|
| BiLSTM-CRF | 97.57 |
| BiLSTM-softmax | 97.58 |
| BiLSTM-LAN w/o attention† | 97.59 |
| BiLSTM-LAN | 97.65 |

Table 3: Effect of attention layer. † denotes the model without attention sublayers except for the last layer.

### 5.1 Dataset

For English POS tagging, we use the Wall Street Journal (WSJ) portion of the Penn Treebank (PTB) (Marcus et al., 1993), which has 45 POS tags. We adopt the same partitions used by previous work (Manning, 2011; Yang et al., 2018), selecting sections 0-18 as the training set, sections 19-21 as the development set and sections 22-24 as the test set. For multilingual POS tagging, we use treebanks from Universal Dependencies(UD) v2.2 (Silveira et al., 2014; Nivre et al., 2018) with the standard splits, choosing 8 resource-rich languages in our experiments. For NER, we use the OntoNotes 5.0 (Hovy et al., 2006; Pradhan et al., 2013). Following previous work,we adopt the official OntoNotes data split used for co-reference resolution in the CoNLL-2012 shared task (Pradhan et al., 2012). We train our CCG supertagging

| Model | train (s) | test (st/s) |
|---|---|---|
| BiLSTM-CRF (POS) | 181.32 | 781.90 |
| BiLSTM-LAN (POS) | 128.75 | 805.32 |
| BiLSTM-CRF (CCG) | 884.67 | 599.18 |
| BiLSTM-LAN (CCG) | 369.98 | 713.70 |

Table 4: Comparison of the training time for one iteration and decoding speed. st indicates sentences

model on CCGBank (Hockenmaier and Steedman, 2007). Sections 2-21 are used for training, section 0 for development and section 23 as the test set. The performance is calculated on the 425 most frequent labels. Table 1 shows the numbers of sentences, tokens and labels for training, development and test, respectively.

### 5.2 Settings

**Hyper-Parameters.** We use 100-dimensional GloVe (Pennington et al., 2014) word embeddings for English POS tagging (WSJ, UD v2.2 EN) and name entity recognition, 300-dimensional Glove word embeddings for CCG supertagging and 64-dimensional Polyglot (Al-Rfou et al., 2013) word embeddings for multilingual POS tagging. The detail values of the hyper-parameters for all experiments are summarized in Appendix A.

**Evaluation.** F-1 score are used for NER. Other tasks are evaluated based on the accuracy. We repeat the same experiment five times with the same hyperparameters and report the max accuracy, average accuracy and standard deviation for POS tagging. For fair comparison, all experiments are implemented in NCRF++ (Yang and Zhang, 2018) and conducted using a GeForce GTX 1080Ti with 11GB memory.

### 5.3 Development Experiments

We report a set of WSJ development experiments to show our investigation on key configurations of BiLSTM-LAN and BiLSTM-CRF.

**Label embedding size.** Table 2 shows the effect of the label embedding size. Notable improvement can be achieved when the label embedding size increases from 200 to 400 in our model, but the accuracy does not further increase when the size increases beyond 400. We fix the label embedding size to 400 for our model.

**Number of Layers.** For BiLSTM-CRF, previous work has shown that one BiLSTM layer is the most effecitve for POS tagging (Ma and Hovy, 2016; Yang et al., 2018). Table 2 compares different numbers of BiLSTM layers and hidden sizes.

| Model | Mean±std | Max |
|---|---|---|
| BiLSTM-CRF† | 97.47±0.02 | 97.49 |
| BiLSTM-CRF‡ | 97.50±0.03 | 97.51 |
| BiLSTM-softmax† | 97.48±0.02 | 97.51 |
| BiLSTM-LAN | **97.58±0.04** | **97.65** |

Table 5: Result for POS tagging on WSJ. † Yang et al. (2018) and ‡ Yasunaga et al. (2018) are baseline models re-implemented in NCRF++ (Yang and Zhang, 2018). Our results are same as Table 6 of Yang et al. (2018).

| Model | Accuracy |
|---|---|
| Plank et al. (2016) | 97.22 |
| Huang et al. (2015) | 97.55 |
| Ma and Hovy (2016) | 97.55 |
| Liu et al. (2017) | 97.53 |
| Yang et al. (2018) | 97.51 |
| Zhang et al. (2018c) | 97.55 |
| Yasunaga et al. (2018) | 97.58 |
| Xin et al. (2018) | 97.58 |
| Transformer-softmax (Guo et al., 2019) | 97.04 |
| BiLSTM-softmax (Yang et al., 2018) | 97.51 |
| BiLSTM-CRF (Yang et al., 2018) | 97.51 |
| BiLSTM-LAN | **97.65** |

Table 6: Main results on WSJ.

As can be seen, a multi-layer model with larger hidden sizes does not give significantly better results compared to a 1-layer model with a hidden size of 400. We thus chose the latter for the final model.

For BiLSTM-LAN, each layer learns a more abstract representation of word and label distribution sequences. As shown in Table 2, for POS tagging, it is effective to capture label dependencies using two layers. More layers do not empirically improve the performance. We thus set the final number of layers to 3.

**The effectiveness of Model Structure.** To evaluate the effect of BiLSTM-LAN layers, we conduct ablation experiments as shown in Table 3. In BiLSTM-LAN w/o attention, we remove the attention inference sublayers from BiLSTM-LAN except for the last BiLSTM-LAN layer. This model is reminiscent to BiLSTM-softmax except that the output is based on label embeddings. It gives an accuracy slightly higher than that of LSTM-softmax, which demonstrates the advantage of label embeddings. On the other hand, it significantly underperforms BiLSTM-LAN ($p$-value$<$0.01), which shows the advantage of hierarchically-refined label distribution sequence encoding.

**Model Size vs CRF.** Table 2 also compares the effect of model sizes. We observe that: (1) As

| | | cs | da | en | fr | nl | no | pt | sv |
|---|---|---|---|---|---|---|---|---|---|
| BiLSTM-CRF | mean | 98.42 | 95.77 | 95.41 | 96.94 | 94.65 | 97.07 | 97.78 | 96.06 |
| | ± std | 0.03 | 0.12 | 0.06 | 0.08 | 0.11 | 0.11 | 0.04 | 0.07 |
| (Yasunaga et al., 2018) | training(s) | 268.74 | 18.17 | 58.20 | 70.10 | 44.49 | 56.06 | 51.59 | 15.54 |
| BiLSTM-softmax | mean | 98.48 | 95.90 | 95.36 | 97.01 | 94.76 | 97.26 | 97.78 | 95.98 |
| | ± std | 0.04 | 0.09 | 0.17 | 0.09 | 0.17 | 0.03 | 0.05 | 0.08 |
| | training(s) | 129.14 | 9.27 | 25.02 | 33.65 | 23.00 | 28.84 | 23.13 | 8.16 |
| BiLSTM-LAN | mean | **98.75** | **96.26** | **95.59** | **97.28** | **94.94** | **97.59** | **98.04** | **96.55** |
| | ± std | 0.02 | 0.12 | 0.13 | 0.08 | 0.11 | 0.04 | 0.04 | 0.01 |
| | training(s) | 165.64 | 11.32 | 33.04 | 40.48 | 29.71 | 37.06 | 27.48 | 10.40 |

Table 7: Multilingual POS tagging result on UD v2.2 treebanks, compared on 8 resource-rich languages.

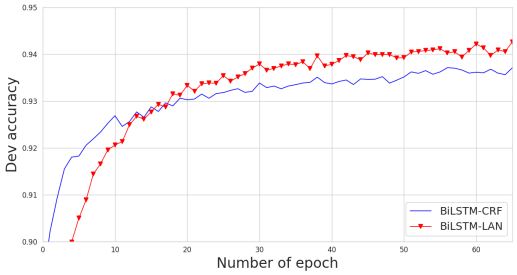


Figure 3: Training on the WSJ development set.

the model size increases, both BiLSTM-CRF and BiLSTM-LAN see a peak point beyond which further increase of model size does not bring better results, which is consistent with observations from prior work, demonstrating that the number of parameters is *not* the decisive factor to model accuracy; and (2) the best-performing BiLSTM-LAN model size is comparable to that of the BiLSTM-CRF model size, which indicates that the model structure is more important for the better accuracy of BiLSTM-LAN.

**Speed vs CRF.** Table 4 shows a comparison of training and decoding speeds. BiLSTM-LAN processes 805 and 714 sentences per second on the WSJ and CCGBank development data, respectively, outperforming BiLSTM-CRF by 3% and 19%, respectively. The larger speed improvement on CCGBank shows the benefit of lower asymptotic complexity, as discussed in Section 4.

**Training vs CRF.** Figure 3 presents the training curves on the WSJ development set. At the beginning, BiLSTM-LAN converges slower than BiLSTM-CRF, which is likely because BiLSTM-LAN has more complex layer structures for label embedding and attention. After around 15 training iterations, the accuracy of BiLSTM-LAN on the development sets becomes increasingly higher than BiLSTM-CRF. This demonstrates the effect of label embeddings, which allows more structured knowledge to be learned in modeling.

### 5.4 Final Results

**WSJ.** Table 5 shows the final POS tagging results on WSJ. Each experiment is repeated 5 times. BiLSTM-LAN gives significant accuracy improvements over both BiLSTM-CRF and BiLSTM-softmax ($p$ <0.01), which is consistent with observations on development experiments.

Table 6 compares our model with top-performing methods reported in the literature. In particular, Huang et al. (2015) use BiLSTM-CRF. Ma and Hovy (2016), Liu et al. (2017) and Yang et al. (2018) explore character level representations on BiLSTM-CRF. Zhang et al. (2018c) use S-LSTM-CRF, a graph recurrent network encoder. Yasunaga et al. (2018) demonstrate that adversarial training can improve the tagging accuracy. Xin et al. (2018) proposed a compositional character-to-word model combined with LSTM-CRF. BiLSTM-LAN gives highly competitive result on WSJ without training on external data.

**Universal Dependencies(UD) v2.2.** We design a multilingual experiment to compare BiLSTM-softmax, BiLSTM-CRF (strictly following Yasunaga et al. (2018) [1], which is the state-of-the-art on multi-lingual POS tagging) and BiLSTM-LAN. The accuracy and training speeds are shown in Table 7. Our model outperforms all the baselines on all the languages. The improvements are statistically significant for all the languages ($p$ <0.01), suggesting that BiLSTM-LAN is generally effective across languages.

**OntoNotes 5.0.** In NER, BiLSTM-CRF is widely used, because local dependencies between neighboring labels relatively more important that POS tagging and CCG supertagging. BiLSTM-LAN also significantly outperforms BiLSTM-CRF by 1.17 F1-score ($p$ <0.01). Table 8 compares BiLSTM-LAN to other published results on

[1] Note that our results are different from Table 2 of Yasunaga et al. (2018), since they reported results on UD v1.2

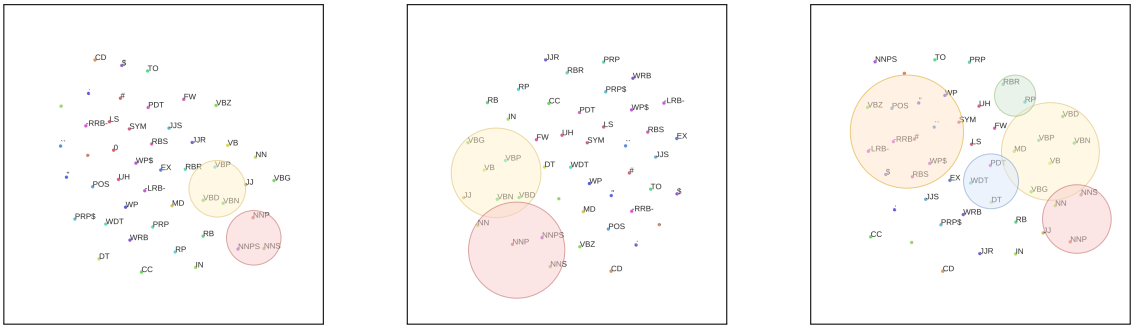

(a) 5 iterations (b) 15 iterations (c) 38 iterations

Figure 4: t-SNE plot of label embeddings after different numbers of training iterations.

OntoNotes 5.0. Durrett and Klein (2014) propose a joint model for coreference resolution, entity linking and NER. Chiu and Nichols (2016) use a BiLSTM with CNN character encoding. Shen et al. (2017) introduce active learning to get better performance. Strubell et al. (2017) present an iterated dilated convolutions, which is a faster alternative to BiLSTM. Ghaddar and Langlais (2018) demonstrate that lexical features are actually quite useful for NER. Clark et al. (2018) present a cross-view training for neural sequence models. BiLSTM-LAN obtains highly competitive results compared with various of top-performance models without training on external data.

**CCGBank.** In CCG supertagging, the major challenge is a larger set of lexical tags $|L|$ and supertag constraints over long distance dependencies. As shown in Table 9, BiLSTM-LAN significantly outperforms both BiLSTM-softmax and BiLSTM-CRF ($p$ <0.01), showing the advantage of LAN. Xu et al. (2015) and Vaswani et al. (2016a) explore BiRNN-softmax and BiLSTM-softmax, respectively. Søgaard and Goldberg (2016) present a multi-task learning architecture with BiRNN to improve the performance. Lewis et al. (2016) train BiLSTM-softmax using tri-training. Vaswani et al. (2016b) combine a LSTM language model and BiLSTM over the supertags. Tu and Gimpel (2019) introduce the inference network (Tu and Gimpel, 2018) in CCG supertagging to speed up the training and decoding for BiLSTM-CRF. Compared with these methods, BiLSTM-LAN obtains new state-of-the-art results on CCGBank, matching the tri-training performance of Lewis et al. (2016), without training on external data.

| Model | F1 |
|---|---|
| Durrett and Klein (2014) | 84.04 |
| Chiu and Nichols (2016) | 86.28 |
| Shen et al. (2017) | 86.52 |
| Strubell et al. (2017) | 86.84 |
| Ghaddar and Langlais (2018) | 87.95 |
| Clark et al. (2018)* | **88.81** |
| BiLSTM-softmax (Strubell et al., 2017) | 83.76 |
| BiLSTM-CRF (Strubell et al., 2017) | 86.99 |
| BiLSTM-LAN | 88.16 |

Table 8: F1-scores on the OntoNotes 5.0 test set. * denotes semi-supervised and multi-task learning.

| Model | Accuracy (%) |
|---|---|
| Xu et al. (2015) | 93.0 |
| Søgaard and Goldberg (2016) | 93.3 |
| Vaswani et al. (2016a) | 94.2 |
| Lewis et al. (2016) | 94.3 |
| Lewis et al. (2016) | 94.7* |
| Vaswani et al. (2016b) | 94.5 |
| Tu and Gimpel (2019) | 94.4 |
| BiLSTM-softmax | 94.1 |
| BiLSTM-CRF | 94.1 |
| BiLSTM-LAN | **94.7** |

Table 9: Supertagging accuracy on CCGbank test set. * indicates that further gains follow from semi-supervised tri-training (improving the accuracy from 94.3% to 94.7%).

## 6 Discussion

**Visualization.** A salient advantage of LAN is more interpretable models. We visualize the label embeddings as well as label attention weights for POS tagging. We use t-SNE to visualize the 45 different English POS tags (WSJ) on a 2D map after 5, 15, 38 training iteration, respectively. Each dot represents a label embedding. As can be seen in Figure 4, label embeddings are increasingly more meaningful during training. Initially, the vectors sit in random locations in the space. After 5 iterations, small clusters emerge, such as

| Sentence | Gold Standard | LSTM-softmax | LSTM-CRF | LSTM-LAN |
|---|---|---|---|---|
| it | NP | NP | NP | NP |
| settled | (S[dcl]\NP)/PP | (S[dcl]\NP)/PP | (S[dcl]\NP)/PP | (S[dcl]\NP)/PP |
| with | ((S\NP)\(S\NP))/NP | PP/NP | PP/NP | ((S\NP)\(S\NP))/NP |
| a | NP[nb]/N | NP[nb]/N | NP[nb]/N | NP[nb]/N |
| loss | N | N | N | N |
| of | (NP\NP)/NP | (NP\NP)/NP | (NP\NP)/NP | (NP\NP)/NP |
| 4.95 | N / N | N / N | N / N | N / N |
| cents | N | N | N | N |
| at | PP/NP | (NP\NP)/NP | PP/NP | PP/NP |
| $ | N / N[num] | N / N[num] | N / N[num] | N / N[num] |
| 1.3210 | N[num] | N[num] | N[num] | N[num] |
| a | (NP\NP) / N | (NP\NP) / N | (NP\NP) / N | (NP\NP) / N |
| pound | N | N | N | N |
| . | . | . | . | . |

Table 10: CCG case analysis. The error are in yellow.

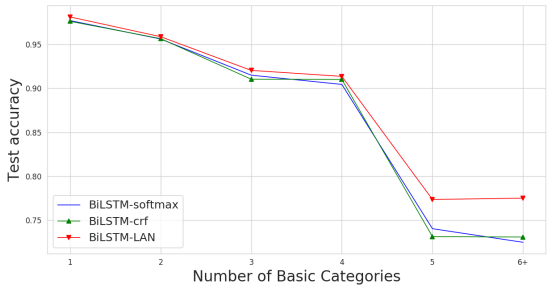


Figure 5: Accuracy against supercategory complexity.

"NNP" and "NNPS", "VBD" and "VBN", "JJS" and "JJR" etc. The clusters grow absorbing more related tags after more training iterations. After 38 training iterations, most similar POS tags are grouped together, such as "VB", "VBD", "VBN", "VBG" and "VBP". More attention visualization are shown in Appendix B.

**Supercategory Complexity.** We also measure the complexity of supercategories by the number of basic categories that they contain. According to this definition, "S", "S/NP" and "(S \NP)/NP" have complexities of 1, 2 and 3, respectively. Figure 5 shows the accuracy of BiLSTM-softmax, BiLSTM-CRF and BiLSTM-LAN against the supertag complexity. As the complexity increases, the performance of all the models decrease, which conforms to intuition. BiLSTM-CRF does not show obvious advantages over BiLSTM-softmax on complex categories. In contrast, BiLSTM-LAN outperforms both models on complex categories, demonstrating its advantage in capturing more sophisticated label dependencies.

**Case Study.** Some predictions of BiLSTM-softmax, BiLSTM-CRF and BiLSTM-LAN are shown in Table 10. The sentence contains two prepositional phrases "with ..." and "at ...", thus exemplifies the PP-attachment problem, one of the hardest sub-problems in CCG supertagging. As can be seen, BiLSTM-softmax fails to learn the long-range relation between "settled" and "at". BiLSTM-CRF strictly follows the hard constraint between neighbor categories thanks to Markov label transition. However, it predicts "with" incorrectly as "PP/NP" with the former supertag ending with "/PP". In contrast, BiLSTM-LAN can capture potential long-term dependency and better determine the supertags based on global label information. In this case, our model can effectively represent a full label search space without making Markov assumptions.

## 7 Conclusion

We investigate a hierarchically-refined label attention network (LAN) for sequence labeling, which leverages label embeddings and captures potential long-range label dependencies by deep attentional encoding of label distribution sequences. Both in theory and empirical results prove that BiLSTM-LAN effective solve label bias issue. Results on POS tagging, NER and CCG supertagging show that BiLSTM-LAN outperforms BiLSTM-CRF and BiLSTM-softmax.

## Acknowledgments

We thank Zhiyang Teng and Junchi Zhang for insightful discussions. We thank Chenhua Chen for proofreading the paper. We also thank all anonymous reviewers for their constructive comments. This work is supported by National Science Foundation of China (Grant No. 61976180). The corresponding author is Yue Zhang.

## References


Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual nlp. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 183–192, Sofia, Bulgaria. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. 2017. Recurrent attention network on memory for aspect sentiment analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 452–461. Association for Computational Linguistics.

Jason P.C. Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Kevin Clark, Minh-Thang Luong, Christopher D. Manning, and Quoc Le. 2018. Semi-supervised sequence modeling with cross-view training. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1914–1925, Brussels, Belgium. Association for Computational Linguistics.

Cícero Nogueira Dos Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–1818–II–1826. JMLR.org.

Timothy Dozat and Christopher D Manning. 2016. Deep biaffine attention for neural dependency parsing. *arXiv preprint arXiv:1611.01734*.

Greg Durrett and Dan Klein. 2014. A joint model for entity analysis: Coreference, typing, and linking. *Transactions of the Association for Computational Linguistics*, 2:477–490.

Abbas Ghaddar and Phillippe Langlais. 2018. Robust lexical features for improved neural network named-entity recognition. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1896–1907, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Qipeng Guo, Xipeng Qiu, Pengfei Liu, Yunfan Shao, Xiangyang Xue, and Zheng Zhang. 2019. Star-transformer. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1315–1325, Minneapolis, Minnesota. Association for Computational Linguistics.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 1693–1701, Cambridge, MA, USA. MIT Press.

Julia Hockenmaier and Mark Steedman. 2007. Ccgbank: A corpus of ccg derivations and dependency structures extracted from the penn treebank. *Comput. Linguist.*, 33(3):355–396.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: The 90% solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, NAACL-Short '06, pages 57–60, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *CoRR*, abs/1508.01991.

Rudolf Kadlec, Martin Schmid, Ondřej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 908–918. Association for Computational Linguistics.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *TACL*, 4:313–327.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270. Association for Computational Linguistics.

Mike Lewis, Kenton Lee, and Luke Zettlemoyer. 2016. Lstm ccg parsing. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 221–231. Association for Computational Linguistics.

Qi Li, Tianshi Li, and Baobao Chang. 2016. Discourse parsing with attention-based hierarchical neural networks. In *Proceedings of the 2016 Conference on*

*Empirical Methods in Natural Language Processing*, pages 362–371. Association for Computational Linguistics.

Jiangming Liu and Yue Zhang. 2017. Attention modeling for targeted sentiment. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 572–577. Association for Computational Linguistics.

Liyuan Liu, Jingbo Shang, Frank Xu, Xiang Ren, Huan Gui, Jian Peng, and Jiawei Han. 2017. Empower sequence labeling with task-aware neural language model. *arXiv preprint arXiv:1709.04109*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074. Association for Computational Linguistics.

Christopher D. Manning. 2011. Part-of-speech tagging from 97% to 100%: Is it time for some linguistics? In *Computational Linguistics and Intelligent Text Processing*, pages 171–189, Berlin, Heidelberg. Springer Berlin Heidelberg.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19(2):313–330.

Jinseok Nam, Eneldo Loza Mencía, and Johannes Fürnkranz. 2016. All-in text: Learning document, label, and word representations jointly. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pages 1948–1954. AAAI Press.

Joakim Nivre, Mitchell Abrams, Željko Agić, Lars Ahrenberg, Lene Antonsen, Maria Jesus Aranzabe, Gashaw Arutie, Masayuki Asahara, Luma Ateyah, Mohammed Attia, Aitziber Atutxa, Liesbeth Augustinus, Elena Badmaeva, Miguel Ballesteros, Esha Banerjee, Sebastian Bank, Verginica Barbu Mititelu, John Bauer, Sandra Bellato, Kepa Bengoetxea, Riyaz Ahmad Bhat, Erica Biagetti, Eckhard Bick, Rogier Blokland, Victoria Bobicev, Carl Börstell, Cristina Bosco, Gosse Bouma, Sam Bowman, Adriane Boyd, Aljoscha Burchardt, Marie Candito, Bernard Caron, Gauthier Caron, Gülşen Cebiroğlu Eryiğit, Giuseppe G. A. Celano, Savas Cetin, Fabricio Chalub, Jinho Choi, Yongseok Cho, Jayeol Chun, Silvie Cinková, Aurélie Collomb, Çağrı Çöltekin, Miriam Connor, Marine Courtin, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Arantza Diaz de Ilarraza, Carly Dickerson, Peter Dirix, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Marhaba Eli, Ali Elkahky, Binyam Ephrem, Tomaž Erjavec, Aline Etienne, Richárd Farkas, Hector Fernandez Alcalde, Jennifer Foster, Cláudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Moa Gärdenfors, Kim Gerdes, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökırmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Céline Guillot-Barbance, Nizar Habash, Jan Hajič, Jan Hajič jr., Linh Hà Mỹ, Na-Rae Han, Kim Harris, Dag Haug, Barbora Hladká, Jaroslava Hlaváčová, Florinel Hociung, Petter Hohle, Jena Hwang, Radu Ion, Elena Irimia, Tomáš Jelínek, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşıkara, Sylvain Kahane, Hiroshi Kanayama, Jenna Kanerva, Tolga Kayadelen, Václava Kettnerová, Jesse Kirchner, Natalia Kotsyba, Simon Krek, Sookyoung Kwak, Veronika Laippala, Lorenzo Lambertino, Tatiana Lando, Septina Dian Larasati, Alexei Lavrentiev, John Lee, Phng Lê H`ông, Alessandro Lenci, Saran Lertpradit, Herman Leung, Cheuk Ying Li, Josie Li, Keying Li, KyungTae Lim, Nikola Ljubešić, Olga Loginova, Olga Lyashevskaya, Teresa Lynn, Vivien Macketanz, Aibek Makazhanov, Michael Mandl, Christopher Manning, Ruli Manurung, Cătălina Mărănduc, David Mareček, Katrin Marheinecke, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Gustavo Mendonça, Niko Miekka, Anna Missilä, Cătălin Mititelu, Yusuke Miyao, Simonetta Montemagni, Amir More, Laura Moreno Romero, Shinsuke Mori, Bjartur Mortensen, Bohdan Moskalevskyi, Kadri Muischnek, Yugo Murawaki, Kaili Müürisep, Pinkey Nainwani, Juan Ignacio Navarro Horñiacek, Anna Nedoluzhko, Gunta Nešpore-Bērzkalne, Lng Nguy˜ên Thị, Huy`ên Nguy˜ên Thị Minh, Vitaly Nikolaev, Rattima Nitisaroj, Hanna Nurmi, Stina Ojala, Adédayọ̀ Olúòkun, Mai Omura, Petya Osenova, Robert Östling, Lilja Øvrelid, Niko Partanen, Elena Pascual, Marco Passarotti, Agnieszka Patejuk, Siyao Peng, Cenel-Augusto Perez, Guy Perrier, Slav Petrov, Jussi Piitulainen, Emily Pitler, Barbara Plank, Thierry Poibeau, Martin Popel, Lauma Pretkalniņa, Sophie Prévost, Prokopis Prokopidis, Adam Przepiórkowski, Tiina Puolakainen, Sampo Pyysalo, Andriela Rääbis, Alexandre Rademaker, Loganathan Ramasamy, Taraka Rama, Carlos Ramisch, Vinit Ravishankar, Livy Real, Siva Reddy, Georg Rehm, Michael Rießler, Larissa Rinaldi, Laura Rituma, Luisa Rocha, Mykhailo Romanenko, Rudolf Rosa, Davide Rovati, Valentin Roca, Olga Rudina, Shoval Sadde, Shadi Saleh, Tanja Samardžić, Stephanie Samson, Manuela Sanguinetti, Baiba Saulīte, Yanin Sawanakunanon, Nathan Schneider, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Mo Shen, Atsuko Shimada, Muh Shohibussirri, Dmitry Sichinava, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Isabela Soares-Bastos, Antonio Stella, Milan Straka, Jana Strnadová, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Dima Taji, Yuta Takahashi, Takaaki Tanaka, Isabelle Tellier, Trond Trosterud, Anna Trukhina, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Zdeňka Urešová, Larraitz Uria, Hans Uszkoreit, Sowmya Vajjala, Daniel van Niekerk, Gertjan

van Noord, Viktor Varga, Veronika Vincze, Lars Wallin, Jonathan North Washington, Seyi Williams, Mats Wirén, Tsegay Woldemariam, Tak-sum Wong, Chunxiao Yan, Marat M. Yavrumyan, Zhuoran Yu, Zdeněk Žabokrtský, Amir Zeldes, Daniel Zeman, Manying Zhang, and Hanzhi Zhu. 2018. Universal dependencies 2.2. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Mark Palatucci, Dean Pomerleau, Geoffrey E Hinton, and Tom M Mitchell. 2009. Zero-shot learning with semantic output codes. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1410–1418. Curran Associates, Inc.

Fuchun Peng, Fangfang Feng, and Andrew McCallum. 2004. Chinese segmentation and new word detection using conditional random fields. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 412–418. Association for Computational Linguistics.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using OntoNotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152, Sofia, Bulgaria. Association for Computational Linguistics.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 1–40, Jeju Island, Korea. Association for Computational Linguistics.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, CoNLL '09, pages 147–155, Stroudsburg, PA, USA. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2017. Optimal hyperparameters for deep lstm-networks for sequence labeling tasks. *arXiv preprint arXiv:1707.06799*.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389. Association for Computational Linguistics.

Yanyao Shen, Hyokun Yun, Zachary Lipton, Yakov Kronrod, and Animashree Anandkumar. 2017. Deep active learning for named entity recognition. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 252–256, Vancouver, Canada. Association for Computational Linguistics.

Natalia Silveira, Timothy Dozat, Marie-Catherine de Marneffe, Samuel Bowman, Miriam Connor, John Bauer, and Christopher D. Manning. 2014. A gold standard dependency corpus for English. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*.

Richard Socher, Milind Ganjoo, Christopher D. Manning, and Andrew Y. Ng. 2013. Zero-shot learning through cross-modal transfer. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'13, pages 935–943, USA. Curran Associates Inc.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 231–235. Association for Computational Linguistics.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5027–5038. Association for Computational Linguistics.

Emma Strubell, Patrick Verga, David Belanger, and Andrew McCallum. 2017. Fast and accurate entity recognition with iterated dilated convolutions. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2670–2680, Copenhagen, Denmark. Association for Computational Linguistics.

Jian Tang, Meng Qu, and Qiaozhu Mei. 2015. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1165–1174. ACM.

Zhiyang Teng and Yue Zhang. 2018. Two local models for neural constituent parsing. In *Proceedings*

*of the 27th International Conference on Computational Linguistics*, pages 119–132. Association for Computational Linguistics.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 173–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Lifu Tu and Kevin Gimpel. 2018. Learning approximate inference networks for structured prediction. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Lifu Tu and Kevin Gimpel. 2019. Benchmarking approximate inference methods for neural structured prediction. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3313–3324, Minneapolis, Minnesota. Association for Computational Linguistics.

A. Vaswani, Y. Bisk, K. Sagae, and R. Musa. 2016a. Supertagging with LSTMs. In *Proc. NAACL*.

Ashish Vaswani, Yonatan Bisk, Kenji Sagae, and Ryan Musa. 2016b. Supertagging with lstms. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 232–237. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Guoyin Wang, Chunyuan Li, Wenlin Wang, Yizhe Zhang, Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. 2018a. Joint embedding of words and labels for text classification. *arXiv preprint arXiv:1805.04174*.

Lu Wang, Shoushan Li, Changlong Sun, Luo Si, Xiaozhong Liu, Min Zhang, and Guodong Zhou. 2018b. One vs. many qa matching with both word-level and sentence-level attention network. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2540–2550. Association for Computational Linguistics.

Xun Wang, Katsuhito Sudoh, and Masaaki Nagata. 2015. Empty category detection with joint context-label embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 263–271. Association for Computational Linguistics.

Wei Wu, Houfeng Wang, Tianyu Liu, and Shuming Ma. 2018. Phrase-level self-attention networks for universal sentence encoding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3729–3738. Association for Computational Linguistics.

Yingwei Xin, Ethan Hart, Vibhuti Mahajan, and Jean-David Ruvini. 2018. Learning better internal structure of words for sequence labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2584–2593, Brussels, Belgium. Association for Computational Linguistics.

Chang Xu, Cecile Paris, Surya Nepal, and Ross Sparks. 2018. Cross-target stance classification with self-attention networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 778–783. Association for Computational Linguistics.

Wenduan Xu, Michael Auli, and Stephen Clark. 2015. Ccg supertagging with a recurrent neural network. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 250–255. Association for Computational Linguistics.

Jie Yang, Shuailong Liang, and Yue Zhang. 2018. Design challenges and misconceptions in neural sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*.

Jie Yang and Yue Zhang. 2018. Ncrf++: An open-source neural sequence labeling toolkit. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Michihiro Yasunaga, Jungo Kasai, and Dragomir Radev. 2018. Robust multilingual part-of-speech tagging via adversarial training. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 976–986. Association for Computational Linguistics.

Honglun Zhang, Liqiang Xiao, Wenqing Chen, Yongkun Wang, and Yaohui Jin. 2018a. Multi-task label embedding for text classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4545–4553. Association for Computational Linguistics.

Yuan Zhang, Hongshen Chen, Yihong Zhao, Qun Liu, and Dawei Yin. 2018b. Learning tag dependencies

for sequence tagging. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 4581–4587. International Joint Conferences on Artificial Intelligence Organization.

Yue Zhang, Qi Liu, and Linfeng Song. 2018c. Sentence-state lstm for text representation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 317–327. Association for Computational Linguistics.

Z. Zhang and V. Saligrama. 2016. Zero-shot learning via joint latent similarity embedding. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6034–6042.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1127–1137. Association for Computational Linguistics.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 207–212. Association for Computational Linguistics.

# Certified Robustness to Adversarial Word Substitutions

**Robin Jia** **Aditi Raghunathan** **Kerem Göksel** **Percy Liang**
Computer Science Department, Stanford University
{robinjia,aditir,kerem,pliang}@cs.stanford.edu

## Abstract

State-of-the-art NLP models can often be fooled by adversaries that apply seemingly innocuous label-preserving transformations (e.g., paraphrasing) to input text. The number of possible transformations scales exponentially with text length, so data augmentation cannot cover all transformations of an input. This paper considers one exponentially large family of label-preserving transformations, in which every word in the input can be replaced with a similar word. We train the first models that are provably robust to *all* word substitutions in this family. Our training procedure uses Interval Bound Propagation (IBP) to minimize an upper bound on the worst-case loss that any combination of word substitutions can induce. To evaluate models' robustness to these transformations, we measure accuracy on adversarially chosen word substitutions applied to test examples. Our IBP-trained models attain 75% adversarial accuracy on both sentiment analysis on IMDB and natural language inference on SNLI. In comparison, on IMDB, models trained normally and ones trained with data augmentation achieve adversarial accuracy of only 8% and 35%, respectively.

## 1 Introduction

Machine learning models have achieved impressive accuracy on many NLP tasks, but they are surprisingly brittle. Adding distracting text to the input (Jia and Liang, 2017), paraphrasing the text (Iyyer et al., 2018; Ribeiro et al., 2018), replacing words with similar words (Alzantot et al., 2018), or inserting character-level "typos" (Belinkov and Bisk, 2017; Ebrahimi et al., 2017) can significantly degrade a model's performance. Such perturbed inputs are called *adversarial examples*, and have shown to break models in other domains as well, most notably in vision (Szegedy et al., 2014; Goodfellow et al., 2015). Since humans are not fooled by the same perturbations, the widespread existence of adversarial examples exposes troubling gaps in models' understanding.

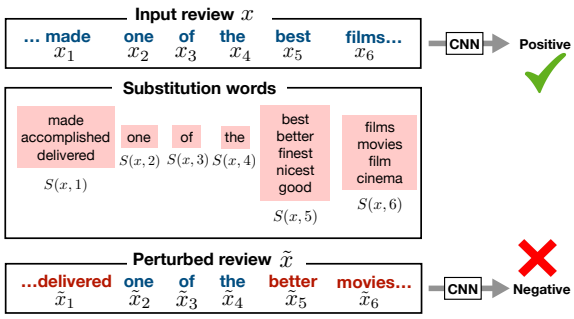


Figure 1: Word substitution-based perturbations in sentiment analysis. For an input $x$, we consider perturbations $\tilde{x}$, in which *every* word $x_i$ can be replaced with any similar word from the set $S(x, i)$, without changing the original sentiment. Models can be easily fooled by adversarially chosen perturbations (e.g., changing "*best*" to "*better*", "*made*" to "*delivered*", "*films*" to "*movies*"), but the ideal model would be robust to all combinations of word substitutions.

In this paper, we focus on the word substitution perturbations of Alzantot et al. (2018). In this setting, an attacker may replace every word in the input with a similar word (that ought not to change the label), leading to an exponentially large number of possible perturbations. Figure 1 shows an example of these word substitutions. As demonstrated by a long line of work in computer vision, it is challenging to make models that are robust to very large perturbation spaces, even when the set of perturbations is known at training time (Goodfellow et al., 2015; Athalye et al., 2018; Raghunathan et al., 2018; Wong and Kolter, 2018).

Our paper addresses two key questions. First, is it possible to guarantee that a model is robust against *all* adversarial perturbations of a given in-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4129–4142, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

put? Existing methods that use heuristic search to attack models (Ebrahimi et al., 2017; Alzantot et al., 2018) are slow and cannot provide guarantees of robustness, since the space of possible perturbations is too large to search exhaustively. We obtain guarantees by leveraging Interval Bound Propagation (IBP), a technique that was previously applied to feedforward networks and CNNs in computer vision (Dvijotham et al., 2018). IBP efficiently computes a tractable *upper bound* on the loss of the worst-case perturbation. When this upper bound on the worst-case loss is small, the model is guaranteed to be robust to all perturbations, providing a *certificate* of robustness. To apply IBP to NLP settings, we derive new interval bound formulas for multiplication and softmax layers, which enable us to compute IBP bounds for LSTMs (Hochreiter and Schmidhuber, 1997) and attention layers (Bahdanau et al., 2015). We also extend IBP to handle discrete perturbation sets, rather than the continuous ones used in vision.

Second, can we train models that are robust in this way? Data augmentation can sometimes mitigate the effect of adversarial examples (Jia and Liang, 2017; Belinkov and Bisk, 2017; Ribeiro et al., 2018; Liu et al., 2019), but it is insufficient when considering very large perturbation spaces (Alzantot et al., 2018). Adversarial training strategies from computer vision (Madry et al., 2018) rely on gradient information, and therefore do not extend to the discrete perturbations seen in NLP. We instead use *certifiably robust training*, in which we train models to optimize the IBP upper bound (Dvijotham et al., 2018).

We evaluate certifiably robust training on two tasks—sentiment analysis on the IMDB dataset (Maas et al., 2011) and natural language inference on the SNLI dataset (Bowman et al., 2015). Across various model architectures (bag-of-words, CNN, LSTM, and attention-based), certifiably robust training consistently yields models which are provably robust to all perturbations on a large fraction of test examples. A normally-trained model has only $8\%$ and $41\%$ accuracy on IMDB and SNLI, respectively, when evaluated on adversarially perturbed test examples. With certifiably robust training, we achieve $75\%$ adversarial accuracy for both IMDB and SNLI. Data augmentation fares much worse than certifiably robust training, with adversarial accuracies falling to $35\%$ and $71\%$, respectively.

## 2 Setup

We consider tasks where a model must predict a label $y \in \mathcal{Y}$ given textual input $x \in \mathcal{X}$. For example, for sentiment analysis, the input $x$ is a sequence of words $x_1, x_2, \ldots, x_L$, and the goal is to assign a label $y \in \{-1, 1\}$ denoting negative or positive sentiment, respectively. We use $z = (x, y)$ to denote an example with input $x$ and label $y$, and use $\theta$ to denote parameters of a model. Let $f(z, \theta) \in \mathbb{R}$ denote some loss of a model with parameters $\theta$ on example $z$. We evaluate models on $f^{\text{0-1}}(z, \theta)$, the zero-one loss under model $\theta$.

### 2.1 Perturbations by word substitutions

Our goal is to build models that are robust to label-preserving perturbations. In this work, we focus on perturbations where words of the input are substituted with similar words. Formally, for every word $x_i$, we consider a set of allowed substitution words $S(x, i)$, including $x_i$ itself. We use $\tilde{x}$ to denote a perturbed version of $x$, where each word $\tilde{x}_i$ is in $S(x, i)$. For an example $z = (x, y)$, let $B_{\text{perturb}}(z)$ denote the set of *all* allowed perturbations of $z$:

$$B_{\text{perturb}}(z) = \{(\tilde{x}, y) : \tilde{x}_i \in S(x, i) \ \ \forall i\}. \quad (1)$$

Figure 1 provides an illustration of word substitution perturbations. We choose $S(x, i)$ so that $\tilde{x}$ is likely to be grammatical and have the same label as $x$ (see Section 5.1).

### 2.2 Robustness to all perturbations

Let $\mathcal{F}(z, \theta)$ denote the set of losses of the network on the set of perturbed examples defined in (1):

$$\mathcal{F}(z, \theta) = \{f(\tilde{z}, \theta) : \tilde{z} \in B_{\text{perturb}}(z)\}. \quad (2)$$

We define the *robust loss* as $\max \mathcal{F}(z, \theta)$, the loss due to worst-case perturbation. A model is robust at $z$ if it classifies all inputs in the perturbation set correctly, i.e., the robust zero-one loss $\max \mathcal{F}^{\text{0-1}}(z, \theta) = 0$. Unfortunately, the robust loss is often intractable to compute, as each word can be perturbed independently. For example, reviews in the IMDB dataset (Maas et al., 2011) have a median of $10^{31}$ possible perturbations and max of $10^{271}$, far too many to enumerate. We instead propose a tractable *upper bound* by constructing a set $\mathcal{O}(z, \theta) \supseteq \mathcal{F}(z, \theta)$. Note that

$$\max \mathcal{O}^{\text{0-1}}(z, \theta) = 0 \Rightarrow \max \mathcal{F}^{\text{0-1}}(z, \theta) = 0 \Leftrightarrow \text{robust at } z. \quad (3)$$

Therefore, whenever $\max \mathcal{O}^{0\text{-}1}(z, \theta) = 0$, this fact is sufficient to *certify* robustness to all perturbed examples $B_{\text{perturb}}(z)$. However, since $\mathcal{O}^{0\text{-}1}(z, \theta) \supseteq \mathcal{F}^{0\text{-}1}(z, \theta)$, the model could be robust even if $\max \mathcal{O}^{0\text{-}1}(z, \theta) \neq 0$.

## 3 Certification via Interval Bound Propagation

We now show how to use Interval Bound Propagation (IBP) (Dvijotham et al., 2018) to obtain a superset $\mathcal{O}(z, \theta)$ of the losses of perturbed inputs $\mathcal{F}(z, \theta)$, given $z$, $\theta$, and $B_{\text{perturb}}(z)$. For notational convenience, we drop $z$ and $\theta$. The key idea is to compute upper and lower bounds on the activations in each layer of the network, in terms of bounds computed for previous layers. These bounds *propagate* through the network, as in a standard forward pass, until we obtain bounds on the final output, i.e., the loss $f$. While IBP bounds may be loose in general, Section 5.2 shows that training networks to minimize the upper bound on $f$ makes these bounds much tighter (Gowal et al., 2018; Raghunathan et al., 2018).

Formally, let $g^i$ denote a scalar-valued function of $z$ and $\theta$ (e.g., a single activation in one layer of the network) computed at node $i$ of the computation graph for a given network. Let $\text{dep}(i)$ be the set of nodes used to compute $g^i$ in the computation graph (e.g., activations of the previous layer). Let $\mathcal{G}^i$ denote the set of possible values of $g^i$ across all examples in $B_{\text{perturb}}(z)$. We construct an interval $\mathcal{O}^i = [\ell^i, u^i]$ that contains all these possible values of $g^i$, i.e., $\mathcal{O}^i \supseteq \mathcal{G}^i$. $\mathcal{O}^i$ is computed from the intervals $\mathcal{O}^{\text{dep}(i)} = \{\mathcal{O}^j : j \in \text{dep}(i)\}$ of the dependencies of $g^i$. Once computed, $\mathcal{O}^i$ can then be used to compute intervals on nodes that depend on $i$. In this way, bounds propagate through the entire computation graph in an efficient forward pass.

We now discuss how to compute interval bounds for NLP models and word substitution perturbations. We obtain interval bounds for model inputs given $B_{\text{perturb}}(z)$ (Section 3.1), then show how to compute $\mathcal{O}^i$ from $\mathcal{O}^{\text{dep(i)}}$ for elementary operations used in standard NLP models (Section 3.2). Finally, we use these bounds to certify robustness and train robust models.

### 3.1 Bounds for the input layer

Previous work (Gowal et al., 2018) applied IBP to continuous image perturbations, which are naturally represented with interval bounds (Dvijotham et al., 2018). We instead work with discrete word substitutions, which we must convert into interval bounds $\mathcal{O}^{\text{input}}$ in order to use IBP. Given input words $x = x_1, \ldots, x_L$, we assume that the model embeds each word as $g^{\text{input}} = [\phi(x_1), \ldots, \phi(x_L)] \in \mathbb{R}^{L \times d}$, where $\phi(x_i) \in \mathbb{R}^d$ is the word vector for word $x_i$. To compute $\mathcal{O}^{\text{input}} \supseteq \mathcal{G}^{\text{input}}$, recall that each input word $x_i$ can be replaced with any $\tilde{x}_i \in S(x, i)$. So, for each coordinate $j \in \{1, \ldots, d\}$, we can obtain an interval bound $\mathcal{O}^{\text{input}}_{ij} = [\ell^{\text{input}}_{ij}, u^{\text{input}}_{ij}]$ for $g^{\text{input}}_{ij}$ by computing the smallest axis-aligned box that contains all the word vectors:

$$\ell^{\text{input}}_{ij} = \min_{w \in S(x,i)} \phi(w)_j, \quad u^{\text{input}}_{ij} = \max_{w \in S(x,i)} \phi(w)_j. \tag{4}$$

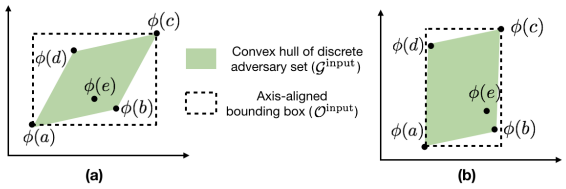


Figure 2: Bounds on the word vector inputs to the neural network. Consider a word (sentence of length one) $x = a$ with the set of substitution words $S(x, 1) = \{a, b, c, d, e\}$. (a) IBP constructs axis-aligned bounds around a set of word vectors. These bounds may be loose, especially if the word vectors are pre-trained and fixed. (b) A different word vector space can give tighter IBP bounds, if the convex hull of the word vectors is better approximated by an axis-aligned box.

Figure 2 illustrates these bounds. We can view this as relaxing a set of discrete points to a convex set that contains all of the points. Section 4.2 discusses modeling choices to make this box tighter.

### 3.2 Interval bounds for elementary functions

Next, we describe how to compute the interval of a node $i$ from intervals of its dependencies. Gowal et al. (2018) show how to efficiently compute interval bounds for affine transformations (i.e., linear layers) and monotonic elementwise nonlinearities (see Appendix 3). This suffices to compute interval bounds for feedforward networks and CNNs. However, common NLP model components like LSTMs and attention also rely on softmax (for attention), element-wise multiplication (for LSTM gates), and dot product (for computing attention scores). We show how to compute interval bounds for these new operations. These building blocks can be used to compute interval bounds

not only for LSTMs and attention, but also for any model that uses these elementary functions.

For ease of notation, we drop the superscript $i$ on $g^i$ and write that a node computes a result $z^{\text{res}} = g(z^{\text{dep}})$ where $z^{\text{res}} \in \mathbb{R}$ and $z^{\text{dep}} \in \mathbb{R}^m$ for $m = |\text{dep}(i)|$. We are given intervals $\mathcal{O}^{\text{dep}}$ such that $z_j^{\text{dep}} \in \mathcal{O}_j^{\text{dep}} = [\ell_j^{\text{dep}}, u_j^{\text{dep}}]$ for each coordinate $j$ and want to compute $\mathcal{O}^{\text{res}} = [\ell^{\text{res}}, u^{\text{res}}]$.

**Softmax layer.** The softmax function is often used to convert activations into a probability distribution, e.g., for attention. Gowal et al. (2018) uses unnormalized logits and does not handle softmax operations. Formally, let $z^{\text{res}}$ represent the normalized score of the word at position $c$. We have $z^{\text{res}} = \frac{\exp(z_c^{\text{dep}})}{\sum_{j=1}^m \exp(z_j^{\text{dep}})}$. The value of $z^{\text{res}}$ is largest when $z_c^{\text{dep}}$ takes its largest value and all other words take the smallest value:

$$u^{\text{res}} = \frac{\exp(u_c^{\text{dep}})}{\exp(u_c^{\text{dep}}) + \sum_{j \neq c} \exp(\ell_j^{\text{dep}})}. \quad (5)$$

We obtain a similar expression for $\ell^{\text{res}}$. Note that $\ell^{\text{res}}$ and $u^{\text{res}}$ can each be computed in a forward pass, with some care taken to avoid numerical instability (see Appendix A.2).

**Element-wise multiplication and dot product.** Models like LSTMs incorporate gates which perform element-wise multiplication of two activations. Let $z^{\text{res}} = z_1^{\text{dep}} z_2^{\text{dep}}$ where $z^{\text{res}}, z_1^{\text{dep}}, z_2^{\text{dep}} \in \mathbb{R}$. The extreme values of the product occur at one of the four points corresponding to the products of the extreme values of the inputs. In other words,

$$\begin{aligned} \mathcal{C} = \{&\ell_1^{\text{dep}} \ell_2^{\text{dep}}, \quad \ell_1^{\text{dep}} u_2^{\text{dep}} \\ &u_1^{\text{dep}} \ell_2^{\text{dep}}, \quad u_1^{\text{dep}} u_2^{\text{dep}}\} \\ \ell^{\text{res}} = \min(\mathcal{C}) \quad & u^{\text{res}} = \max(\mathcal{C}). \end{aligned} \quad (6)$$

Propagating intervals through multiplication nodes therefore requires four multiplications.

Dot products between activations are often used to compute attention scores.[1] The dot product $(z_1^{\text{dep}})^\top z_2^{\text{dep}}$ is just the sum of the element-wise product $z_1^{\text{dep}} \odot z_2^{\text{dep}}$. Therefore, we can bound the dot product by summing the bounds on each element of $z_1^{\text{dep}} \odot z_2^{\text{dep}}$, using the formula for element-wise multiplication.

[1]This is distinct from an affine transformation, because both vectors have associated bounds; in an affine layer, the input has bounds, but the weight matrix is fixed.

### 3.3 Final layer

Classification models typically output a single logit for binary classification, or $k$ logits for $k$-way classification. The final loss $f(z, \theta)$ is a function of the logits $s(x)$. For standard loss functions, we can represent this function in terms of element-wise monotonic functions (Appendix 3) and the elementary functions described in Section 3.2.

1. Zero-one loss: $f(z, \theta) = \mathbb{I}[\max(s(x)) = y]$ involves a max operation followed by a step function, which is monotonic.

2. Cross entropy: For multi-class, $f(z, \theta) = \text{softmax}(s(x))$. In the binary case, $f(z, \theta) = \sigma(s(x))$, where the sigmoid function $\sigma$ is monotonic.

Thus, we can compute bounds on the loss $\mathcal{O}(z, \theta) = [\ell^{\text{final}}, u^{\text{final}}]$ from bounds on the logits.

### 3.4 Certifiably Robust Training with IBP

Finally, we describe certifiably robust training, in which we encourage robustness by minimizing the upper bound on the worst-case loss (Dvijotham et al., 2018; Gowal et al., 2018). Recall that for an example $z$ and parameters $\theta$, $u^{\text{final}}(z, \theta)$ is the upper bound on the loss $f(z, \theta)$. Given a dataset $D$, we optimize a weighted combination of the normal loss and the upper bound $u^{\text{final}}$,

$$\min_\theta \sum_{z \in D} (1 - \kappa) f(z, \theta) + \kappa\, u^{\text{final}}(z, \theta), \quad (7)$$

where $0 \leq \kappa \leq 1$ is a scalar hyperparameter.

As described above, we compute $u^{\text{final}}$ in a modular fashion: each layer has an accompanying function that computes bounds on its outputs given bounds on its inputs. Therefore, we can easily apply IBP to new architectures. Bounds propagate through layers via forward passes, so the entire objective (7) can be optimized via backpropagation.

Gowal et al. (2018) found that this objective was easier to optimize by starting with a smaller space of allowed perturbations, and make it larger during training. We accomplish this by artificially shrinking the input layer intervals $\mathcal{O}_{ij}^{\text{input}} = [\ell_{ij}^{\text{input}}, u_{ij}^{\text{input}}]$ towards the original value $\phi(x_i)_j$ by a factor of $\epsilon$:

$$\begin{aligned} \ell_{ij}^{\text{input}} &\leftarrow \phi(x_i)_j - \epsilon(\phi(x_i)_j - \ell_{ij}^{\text{input}}) \\ u_{ij}^{\text{input}} &\leftarrow \phi(x_i)_j + \epsilon(u_{ij}^{\text{input}} - \phi(x_i)_j). \end{aligned}$$

Standard training corresponds to $\epsilon = 0$. We train for $T^{\text{init}}$ epochs while linearly increasing $\epsilon$ from 0

to 1, and also increasing $\kappa$ from 0 up to a maximum value of $\kappa^\star$, We then train for an additional $T^{\text{final}}$ epochs at $\kappa = \kappa^\star$ and $\epsilon = 1$.

To summarize, we use IBP to compute an upper bound on the model's loss when given an adversarially perturbed input. This bound is computed in a modular fashion. We efficiently train models to minimize this bound via backpropagation.

## 4 Tasks and models

Now we describe the tasks and model architectures on which we run experiments. These models are all built from the primitives in Section 3.

### 4.1 Tasks

Following Alzantot et al. (2018), we evaluate on two standard NLP datasets: the IMDB sentiment analysis dataset (Maas et al., 2011) and the Stanford Natural Language Inference (SNLI) dataset (Bowman et al., 2015). For IMDB, the model is given a movie review and must classify it as positive or negative. For SNLI, the model is given two sentences, a premise and a hypothesis, and is asked whether the premise entails, contradicts, or is neutral with respect to the hypothesis. For SNLI, the adversary is only allowed to change the hypothesis, as in Alzantot et al. (2018), though it is possible to also allow changing the premise.

### 4.2 Models

**IMDB.** We implemented three models for IMDB. The bag-of-words model (BOW) averages the word vectors for each word in the input, then passes this through a two-layer feedforward network with 100-dimensional hidden state to obtain a final logit. The other models are similar, except they run either a CNN or bidirectional LSTM on the word vectors, then average their hidden states. All models are trained on cross entropy loss.

**SNLI** We implemented two models for SNLI. The bag-of-words model (BOW) encodes the premise and hypothesis separately by summing their word vectors, then feeds the concatenation of these encodings to a 3-layer feedforward network. We also reimplement the Decomposable Attention model (Parikh et al., 2016), which uses attention between the premise and hypothesis to compute richer representations of each word in both sentences. These context-aware vectors are used in the same way BOW uses the original word vectors to generate the final prediction. Both models are trained on cross entropy loss. Implementation details are provided in Appendix A.4.

**Word vector layer.** The choice of word vectors affects the tightness of our interval bounds. We choose to define the word vector $\phi(w)$ for word $w$ as the output of a feedforward layer applied to a fixed pre-trained word vector $\phi^{\text{pre}}(w)$:

$$\phi(w) = \text{ReLU}(g^{\text{word}}(\phi^{\text{pre}}(w))), \qquad (8)$$

where $g^{\text{word}}$ is a learned linear transformation. Learning $g^{\text{word}}$ with certifiably robust training encourages it to orient the word vectors so that the convex hull of the word vectors is close to an axis-aligned box. Note that $g^{\text{word}}$ is applied *before* bounds are computed via (4).[2] Applying $g^{\text{word}}$ after the bound calculation would result in looser interval bounds, since the original word vectors $\phi^{\text{pre}}(w)$ might be poorly approximated by interval bounds (e.g., Figure 2a), compared to $\phi(w)$ (e.g., Figure 2b). Section 5.7 confirms the importance of adding $g^{\text{word}}$. We use 300-dimensional GloVe vectors (Pennington et al., 2014) as our $\phi^{\text{pre}}(w)$.

## 5 Experiments

### 5.1 Setup

**Word substitution perturbations.** We base our sets of allowed word substitutions $S(x, i)$ on the substitutions allowed by Alzantot et al. (2018). They demonstrated that their substitutions lead to adversarial examples that are qualitatively similar to the original input and retain the original label, as judged by humans. Alzantot et al. (2018) define the neighbors $N(w)$ of a word $w$ as the $n = 8$ nearest neighbors of $w$ in a "counter-fitted" word vector space where antonyms are far apart (Mrkšić et al., 2016).[3] The neighbors must also lie within some Euclidean distance threshold. They also use a language model constraint to avoid nonsensical perturbations: they allow substituting $x_i$ with $\tilde{x}_i \in N(x_i)$ if and only if it does not decrease the log-likelihood of the text under a pre-trained language model by more than some threshold.

We make three modifications to this approach. First, in Alzantot et al. (2018), the adversary applies substitutions one at a time, and the neighborhoods and language model scores are computed

[2] Equation (4) must be applied before the model can combine information from multiple words, but it can be delayed until after processing each word independently.

[3] Note that the model itself classifies using a different set of pre-trained word vectors; the counter-fitted vectors are only used to define the set of allowed substitution words.

relative to the current altered version of the input. This results in a hard-to-define attack surface, as changing one word can allow or disallow changes to other words. It also requires recomputing language model scores at each iteration of the genetic attack, which is inefficient. Moreover, the same word can be substituted multiple times, leading to semantic drift. We define allowed substitutions relative to the original sentence $x$, and disallow repeated substitutions. Second, we use a faster language model that allows us to query longer contexts; Alzantot et al. (2018) use a slower language model and could only query it with short contexts. Finally, we use the language model constraint only at test time; the model is trained against all perturbations in $N(w)$. This encourages the model to be robust to a larger space of perturbations, instead of specializing for the particular choice of language model. See Appendix A.3 for further details.

**Analysis of word neighbors.** One natural question is whether we could guarantee robustness by having the model treat all neighboring words the same. We could construct equivalence classes of words from the transitive closure of $N(w)$, and represent each equivalence class with one embedding. We found that this would lose a significant amount of information. Out of the 50,000 word vocabulary, 19,122 words would be in the same equivalence class, including the words "good", "bad", "excellent", and "terrible." Of the remaining words, 24,389 (79%) have no neighbors.

**Baseline training methods.** We compare certifiably robust training (Section 3) with both standard training and data augmentation, which has been used in NLP to encourage robustness to various types of perturbations (Jia and Liang, 2017; Belinkov and Bisk, 2017; Iyyer et al., 2018; Ribeiro et al., 2018). In data augmentation, for each training example $z$, we augment the dataset with $K$ new examples $\tilde{z}$ by sampling $\tilde{z}$ uniformly from $B_{\text{perturb}}(z)$, then train on the normal cross entropy loss. For our main experiments, we use $K = 4$. We do not use adversarial training (Goodfellow et al., 2015) because it would require running an adversarial search procedure at each training step, which would be prohibitively slow.

**Evaluation of robustness.** We wish to evaluate robustness of models to all word substitution perturbations. Ideally, we would directly measure *robust accuracy*, the fraction of test examples $z$ for which the model is correct on all $\tilde{z} \in B_{\text{perturb}}(z)$. However, evaluating this exactly involves enumerating the exponentially large set of perturbations, which is intractable. Instead, we compute tractable upper and lower bounds:

1. Genetic attack accuracy: Alzantot et al. (2018) demonstrate the effectiveness of a genetic algorithm that searches for perturbations $\tilde{z}$ that cause model misclassification. The algorithm maintains a "population" of candidate $\tilde{z}$'s and repeatedly perturbs and combines them. We used a population size of 60 and ran 40 search iterations on each example. Since the algorithm does not exhaustively search over $B_{\text{perturb}}(z)$, accuracy on the perturbations it finds is an *upper bound* on the true robust accuracy.

2. Certified accuracy: To complement this upper bound, we use IBP to obtain a tractable lower bound on the robust accuracy. Recall from Section 3.3 that we can use IBP to get an upper bound on the zero-one loss. From this, we obtain a *lower bound* on the robust accuracy by measuring the fraction of test examples for which the zero-one loss is guaranteed to be 0.

**Experimental details.** For IMDB, we split the official train set into train and development subsets, putting reviews for different movies into different splits (matching the original train/test split). For SNLI, we use the official train/development/test split. We tune hyperparameters on the development set for each dataset. Hyperparameters are reported in Appendix A.4.

### 5.2 Main results

Table 1 and Table 2 show our main results for IMDB and SNLI, respectively. We measure accuracy on perturbations found by the genetic attack (upper bound on robust accuracy) and IBP-certified accuracy (lower bound on robust accuracy) on 1000 random test examples from IMDB,[4] and all 9824 test examples from SNLI. Across many architectures, our models are more robust to perturbations than ones trained with data augmentation. This effect is especially pronounced on IMDB, where inputs can be hundreds of words long, so many words can be perturbed. On IMDB, the best IBP-trained model gets 75.0% accuracy on perturbations found by the genetic at-

[4] We downsample the test set because the genetic attack is slow on IMDB, as inputs can be hundreds of words long.

| System | Genetic attack (Upper bound) | IBP-certified (Lower bound) |
|---|---|---|
| **Standard training** | | |
| BOW | 9.6 | 0.8 |
| CNN | 7.9 | 0.1 |
| LSTM | 6.9 | 0.0 |
| **Robust training** | | |
| BOW | 70.5 | 68.9 |
| CNN | **75.0** | **74.2** |
| LSTM | 64.7 | 63.0 |
| **Data augmentation** | | |
| BOW | 34.6 | 3.5 |
| CNN | 35.2 | 0.3 |
| LSTM | 33.0 | 0.0 |

Table 1: Robustness of models on IMDB. We report accuracy on perturbations obtained via the genetic attack (upper bound on robust accuracy), and certified accuracy obtained using IBP (lower bound on robust accuracy) on 1000 random IMDB test set examples. For all models, robust training vastly outperforms data augmentation ($p < 10^{-63}$, Wilcoxon signed-rank test).

| System | Genetic attack (Upper bound) | IBP-certified (Lower bound) |
|---|---|---|
| **Normal training** | | |
| BOW | 40.5 | 2.3 |
| DECOMPATTN | 40.3 | 1.4 |
| **Robust training** | | |
| BOW | **75.0** | **72.7** |
| DECOMPATTN | 73.7 | 72.4 |
| **Data augmentation** | | |
| BOW | 68.5 | 7.7 |
| DECOMPATTN | 70.8 | 1.4 |

Table 2: Robustness of models on the SNLI test set. For both models, robust training outperforms data augmentation ($p < 10^{-10}$, Wilcoxon signed-rank test).

tack, whereas the best data augmentation model gets 35.2%. Normally trained models are even worse, with adversarial accuracies below 10%.

**Certified accuracy.** Certifiably robust training yields models with tight guarantees on robustness—the upper and lower bounds on robust accuracy are close. On IMDB, the best model is *guaranteed* to be correct on all perturbations of 74.2% of test examples, very close to the 75.0% accuracy against the genetic attack. In contrast, for data augmentation models, the IBP bound cannot guarantee robustness on almost all examples. It is possible that a stronger attack (e.g., exhaustive search) could further lower the accuracy of these models, or that the IBP bounds are loose.

LSTM models can be certified with IBP, though they fare worse than other models. IBP bounds may be loose for RNNs because of their long computation paths, along which looseness of bounds can get amplified. Nonetheless, in Appendix A.7, we show on synthetic data that robustly trained LSTMs can learn long-range dependencies.

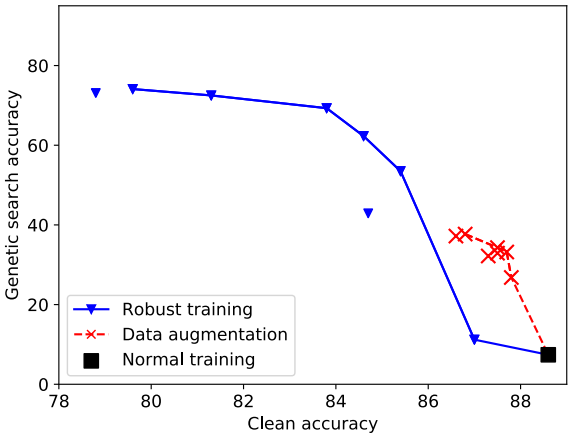


Figure 3: Trade-off between clean accuracy and genetic attack accuracy for CNN models on IMDB. Data augmentation cannot achieve high robustness. Certifiably robust training yields much more robust models, though at the cost of some clean accuracy. Lines connect Pareto optimal points for each training strategy.

### 5.3 Clean versus robust accuracy

Robust training does cause a moderate drop in clean accuracy (accuracy on unperturbed test examples) compared with normal training. On IMDB, our normally trained CNN model gets 89% clean accuracy, compared to 81% for the robustly trained model. We also see a drop on SNLI: the normally trained BOW model gets 83% clean accuracy, compared to 79% for the robustly trained model. Similar drops in clean accuracy are also seen for robust models in vision (Madry et al., 2017). For example, the state-of-the-art robust model on CIFAR10 (Zhang et al., 2019) only has 85% clean accuracy, but comparable normally-trained models get $> 96\%$ accuracy.

We found that the robustly trained models tend to underfit the training data—on IMDB, the CNN model gets only 86% clean training accuracy, lower than the *test* accuracy of the normally trained model. The model continued to underfit when we increased either the depth or width of the network. One possible explanation is that the attack surface adds a lot of noise, though a large enough model should still be able to overfit the training set. Better optimization or a tighter way to compute bounds could also improve training accuracy. We leave further exploration to future work.

Next, we analyzed the trade-off between clean and robust accuracy by varying the importance

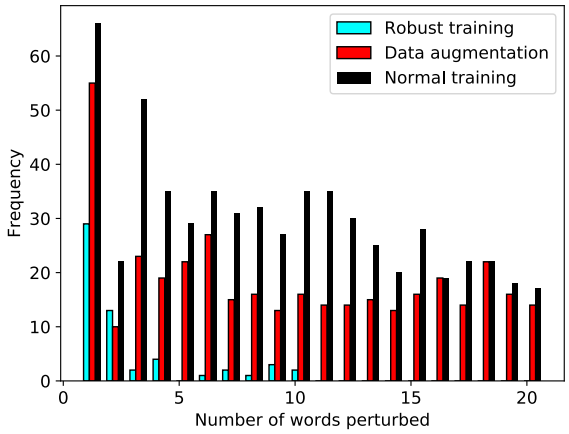


Figure 4: Number of words perturbed by the genetic attack to cause errors by CNN models on 1000 IMDB development set examples. Certifiably robust training reduces the effect of many simultaneous perturbations.

placed on perturbed examples during training. We use accuracy against the genetic attack as our proxy for robust accuracy, rather than IBP-certified accuracy, as IBP bounds may be loose for models that were not trained with IBP. For data augmentation, we vary $K$, the number of augmented examples per real example, from 1 to 64. For certifiably robust training, we vary $\kappa^\star$, the weight of the certified robustness training objective, between 0.01 and 1.0. Figure 3 shows trade-off curves for the CNN model on 1000 random IMDB development set examples. Data augmentation can increase robustness somewhat, but cannot reach very high adversarial accuracy. With certifiably robust training, we can trade off some clean accuracy for much higher robust accuracy.

### 5.4 Runtime considerations

IBP enables efficient computation of $u^{\text{final}}(z, \theta)$, but it still incurs some overhead. Across model architectures, we found that one epoch of certifiably robust training takes between $2\times$ and $4\times$ longer than one epoch of standard training. On the other hand, IBP certificates are much faster to compute at test time than genetic attack accuracy. For the robustly trained CNN IMDB model, computing certificates on 1000 test examples took 5 seconds, while running the genetic attack on those same examples took over 3 hours.

### 5.5 Error analysis

We examined development set examples on which models were correct on the original input but incorrect on the perturbation found by the genetic attack. We refer to such cases as *robustness errors*. We focused on the CNN IMDB models trained normally, robustly, and with data augmentation. We found that robustness errors of the robustly trained model mostly occurred when it was not confident in its original prediction. The model had $> 70\%$ confidence in the correct class for the original input in only $14\%$ of robustness errors. In contrast, the normally trained and data augmentation models were more confident on their robustness errors; they had $> 70\%$ confidence on the original example in $92\%$ and $87\%$ of cases, respectively.

We next investigated how many words the genetic attack needed to change to cause misclassification, as shown in Figure 4. For the normally trained model, some robustness errors involved only a couple changed words (e.g., "*I've finally found a movie worse than . . .*" was classified negative, but the same review with "*I've finally* ***discovered*** *a movie worse than. . .*" was classified positive), but more changes were also common (e.g., part of a review was changed from "*The creature looked very cheesy*" to "*The creature* ***seemed supremely dorky***", with 15 words changed in total). Surprisingly, certifiably robust training nearly eliminated robustness errors in which the genetic attack had to change many words: the genetic attack either caused an error by changing a couple words, or was unable to trigger an error at all. In contrast, data augmentation is unable to cover the exponentially large space of perturbations that involve many words, so it does not prevent errors caused by changing many words.

### 5.6 Training schedule

We investigated the importance of slowly increasing $\epsilon$ during training, as suggested by Gowal et al. (2018). Fixing $\epsilon = 1$ during training led to a 5 point reduction in certified accuracy for the CNN. On the other hand, we found that holding $\kappa$ fixed did not hurt accuracy, and in fact may be preferable. More details are shown in Appendix A.5.

### 5.7 Word vector analysis

We determined the importance of the extra feed-forward layer $g^{\text{word}}$ that we apply to pre-trained word vectors, as described in Section 4.2. We compared with directly using pre-trained word vectors, i.e. $\phi(w) = \phi^{\text{pre}}(w)$. We also tried using $g^{\text{word}}$ but applying interval bounds on $\phi^{\text{pre}}(w)$, then computing bounds on $\phi(w)$ with the IBP for-

mula for affine layers. In both cases, we could not train a CNN to achieve more than $52.2\%$ certified accuracy on the development set. Thus, transforming pre-trained word vectors and applying interval bounds *after* is crucial for robust training. In Appendix A.6, we show that robust training makes the intervals around transformed word vectors smaller, compared to the pre-trained vectors.

## 6 Related Work and Discussion

Recent work on adversarial examples in NLP has proposed various classes of perturbations, such as insertion of extraneous text (Jia and Liang, 2017), word substitutions (Alzantot et al., 2018), paraphrasing (Iyyer et al., 2018; Ribeiro et al., 2018), and character-level noise (Belinkov and Bisk, 2017; Ebrahimi et al., 2017). These works focus mainly on demonstrating models' lack of robustness, and mostly do not explore ways to increase robustness beyond data augmentation. Data augmentation is effective for narrow perturbation spaces (Jia and Liang, 2017; Ribeiro et al., 2018), but only confers partial robustness in other cases (Iyyer et al., 2018; Alzantot et al., 2018). Ebrahimi et al. (2017) tried adversarial training (Goodfellow et al., 2015) for character-level perturbations, but could only use a fast heuristic attack at training time, due to runtime considerations. As a result, their models were still be fooled by running a more expensive search procedure at test time.

Provable defenses have been studied for simpler NLP models and attacks, particularly for tasks like spam detection where real-life adversaries try to evade detection. Globerson and Roweis (2006) train linear classifiers that are robust to adversarial feature deletion. Dalvi et al. (2004) analyzed optimal strategies for a Naive Bayes classifier and attacker, but their classifier only defends against a fixed attacker that does not adapt to the model.

Recent work in computer vision (Szegedy et al., 2014; Goodfellow et al., 2015) has sparked renewed interest in adversarial examples. Most work in this area focuses on $L_\infty$-bounded perturbations, in which each input pixel can be changed by a small amount. The word substitution attack model we consider is similar to $L_\infty$ perturbations, as the adversary can change each input word by a small amount. Our work is inspired by work based on convex optimization (Raghunathan et al., 2018; Wong and Kolter, 2018) and builds directly on interval bound propagation (Dvijotham et al., 2018; Gowal et al., 2018), which has certified robustness of computer vision models to $L_\infty$ attacks. Adversarial training via projected gradient descent (Madry et al., 2018) has also been shown to improve robustness, but assumes that inputs are continuous. It could be applied in NLP by relaxing sets of word vectors to continuous regions.

This work provides certificates against word substitution perturbations for particular models. Since IBP is modular, it can be extended to other model architectures on other tasks. It is an open question whether IBP can give non-trivial bounds for sequence-to-sequence tasks like machine translation (Belinkov and Bisk, 2017; Michel et al., 2019). In principle, IBP can handle character-level typos (Ebrahimi et al., 2017; Pruthi et al., 2019), though typos yield more perturbations per word than we consider in this work. We are also interested in handling word insertions and deletions, rather than just substitutions. Finally, we would like to train models that get state-of-the-art clean accuracy while also being provably robust; achieving this remains an open problem.

In conclusion, state-of-the-art NLP models are accurate on average, but they still have significant blind spots. Certifiably robust training provides a general, principled mechanism to avoid such blind spots by encouraging models to make correct predictions on all inputs within some known perturbation neighborhood. This type of robustness is a necessary (but not sufficient) property of models that truly understand language. We hope that our work is a stepping stone towards models that are robust against an even wider, harder-to-characterize space of possible attacks.

## Acknowledgments

This work was supported by NSF Award Grant no. 1805310 and the DARPA ASED program under FA8650-18-2-7882. R.J. is supported by an NSF Graduate Research Fellowship under Grant No. DGE-114747. A.R. is supported by a Google PhD Fellowship and the Open Philanthropy Project AI Fellowship. We thank Allen Nie for providing the pre-trained language model, and thank Peng Qi, Urvashi Khandelwal, Shiori Sagawa, and the anonymous reviewers for their helpful comments.

## Reproducibility

All code, data, and experiments are available on Codalab at `https://bit.ly/2KVxIFN`.

## References


M. Alzantot, Y. Sharma, A. Elgohary, B. Ho, M. Srivastava, and K. Chang. 2018. Generating natural language adversarial examples. In *Empirical Methods in Natural Language Processing (EMNLP)*.

A. Athalye, N. Carlini, and D. Wagner. 2018. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*.

D. Bahdanau, K. Cho, and Y. Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*.

Y. Belinkov and Y. Bisk. 2017. Synthetic and natural noise both break neural machine translation. *arXiv preprint arXiv:1711.02173*.

S. Bowman, G. Angeli, C. Potts, and C. D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Empirical Methods in Natural Language Processing (EMNLP)*.

C. Chelba, T. Mikolov, M. Schuster, Q. Ge, T. Brants, P. Koehn, and T. Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma. 2004. Adversarial classification. In *International Conference on Knowledge Discovery and Data Mining (KDD)*.

K. Dvijotham, S. Gowal, R. Stanforth, R. Arandjelovic, B. O'Donoghue, J. Uesato, and P. Kohli. 2018. Training verified learners with learned verifiers. *arXiv preprint arXiv:1805.10265*.

J. Ebrahimi, A. Rao, D. Lowd, and D. Dou. 2017. Hotflip: White-box adversarial examples for text classification. *arXiv preprint arXiv:1712.06751*.

A. Globerson and S. Roweis. 2006. Nightmare at test time: robust learning by feature deletion. In *International Conference on Machine Learning (ICML)*, pages 353–360.

I. J. Goodfellow, J. Shlens, and C. Szegedy. 2015. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR)*.

S. Gowal, K. Dvijotham, R. Stanforth, R. Bunel, C. Qin, J. Uesato, T. Mann, and P. Kohli. 2018. On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv preprint arXiv:1810.12715*.

S. Hochreiter and J. Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

M. Iyyer, J. Wieting, K. Gimpel, and L. Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *North American Association for Computational Linguistics (NAACL)*.

R. Jia and P. Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Empirical Methods in Natural Language Processing (EMNLP)*.

D. Kingma and J. Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

N. F. Liu, R. Schwartz, and N. A. Smith. 2019. Inoculation by fine-tuning: A method for analyzing challenge datasets. In *North American Association for Computational Linguistics (NAACL)*.

A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. 2011. Learning word vectors for sentiment analysis. In *Association for Computational Linguistics (ACL)*.

A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. 2017. Towards deep learning models resistant to adversarial attacks (published at ICLR 2018). *arXiv*.

A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. 2018. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations (ICLR)*.

P. Michel, X. Li, G. Neubig, and J. M. Pino. 2019. On evaluation of adversarial perturbations for sequence-to-sequence models. In *North American Association for Computational Linguistics (NAACL)*.

N. Mrkšić, D. Ó Séaghdha, B. Thomson, M. Gašić, L. Rojas-Barahona, P. Su, D. Vandyke, T. Wen, and S. Young. 2016. Counter-fitting word vectors to linguistic constraints. In *North American Association for Computational Linguistics (NAACL)*.

A. Parikh, O. Täckström, D. Das, and J. Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Empirical Methods in Natural Language Processing (EMNLP)*.

J. Pennington, R. Socher, and C. D. Manning. 2014. GloVe: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

D. Pruthi, B. Dhingra, and Z. C. Lipton. 2019. Combating adversarial misspellings with robust word recognition. In *Association for Computational Linguistics (ACL)*.

A. Raghunathan, J. Steinhardt, and P. Liang. 2018. Certified defenses against adversarial examples. In *International Conference on Learning Representations (ICLR)*.

M. T. Ribeiro, S. Singh, and C. Guestrin. 2018. Semantically equivalent adversarial rules for debugging NLP models. In *Association for Computational Linguistics (ACL)*.

C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. 2014. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*.

E. Wong and J. Z. Kolter. 2018. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning (ICML)*.

H. Zhang, Y. Yu, J. Jiao, E. P. Xing, L. E. Ghaoui, and M. I. Jordan. 2019. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning (ICML)*.

# A Supplemental material

## A.1 Additional interval bound formulas

Gowal et al. (2018) showed how to compute interval bounds for affine transformations and monotonic element-wise functions. Here, we review their derivations, for completeness.

**Affine transformations.** Affine transformations are the building blocks of neural networks. Suppose $z^{\text{res}} = a^\top z^{\text{dep}} + b$ for weight $a \in \mathbb{R}^m$ and bias $b \in \mathbb{R}$. $z^{\text{res}}$ is largest when positive entries of $a$ are multiplied with $u^{\text{dep}}$ and negative with $\ell^{\text{dep}}$:

$$\begin{aligned} u^{\text{res}} &= \underbrace{0.5(a+|a|)^\top}_{\text{positive}} u^{\text{dep}} + \underbrace{0.5(a-|a|)^\top}_{\text{negative}} \ell^{\text{dep}} + b \\ &= \mu + r, \end{aligned} \tag{9}$$

where $\mu = 0.5a^\top(\ell^{\text{dep}} + u^{\text{dep}}) + b$ and $r = 0.5|a|^\top(u-l)$. A similar computation yields that $\ell^{\text{res}} = \mu - r$. Therefore, the interval $\mathcal{O}^{\text{res}}$ can be computed using two inner product evaluations: one with $a$ and one with $|a|$.

**Monotonic scalar functions.** Activation functions such as ReLU, sigmoid and tanh are monotonic. Suppose $z^{\text{res}} = \sigma(z^{\text{dep}})$ where $z^{\text{res}}, z^{\text{dep}} \in \mathbb{R}$, i.e. the node applies an element-wise function to its input. The intervals can be computed trivially since $z^{\text{res}}$ is minimized at $\ell^{\text{dep}}$ and maximized at $u^{\text{dep}}$.

$$\ell^{\text{res}} = \sigma(\ell^{\text{dep}}), \;\; u^{\text{res}} = \sigma(u^{\text{dep}}). \tag{10}$$

## A.2 Numerical stability of softmax

In this section, we show how to compute interval bounds for softmax layers in a numerically stable way. We will do this by showing how to handle log-softmax layers. Note that since softmax is just exponentiated log-softmax, and exponentiation is monotonic, bounds on log-softmax directly yield bounds on softmax.

Let $z^{\text{dep}}$ denote a vector of length $m$, let $c$ be an integer $\in \{1, \ldots, m\}$, and let $z^{\text{res}}$ represent the log-softmax score of index $c$, i.e.

$$z^{\text{res}} = \log \frac{\exp(z_c^{\text{dep}})}{\sum_{j=1}^m \exp(z_j^{\text{dep}})} \tag{11}$$

$$= z_c^{\text{dep}} - \log \sum_{j=1}^m \exp(z_j^{\text{dep}}). \tag{12}$$

Given interval bounds $\ell_j \le z_j^{\text{dep}} \le u_j$ for each $j$, we show how to compute upper and lower bounds on $z^{\text{res}}$. For any vector $v$, we assume access to a subroutine that computes

$$\text{logsumexp}(v) = \log \sum_i \exp(v_i)$$

stably. The standard way to compute this is to normalize $v$ by subtracting $\max_i(v_i)$ before taking exponentials, then add it back at the end. logsumexp is a standard function in libraries like PyTorch. We will also rely on the fact that if $v$ is the concatenation of vectors $u$ and $w$, then $\text{logsumexp}(v) = \text{logsumexp}([\text{logsumexp}(u), \text{logsumexp}(w)])$.

**Upper bound.** The upper bound $u^{\text{res}}$ is achieved by having the maximum value of $z_c^{\text{dep}}$, and minimum value of all others. This can be written as:

$$u^{\text{res}} = u_c^{\text{dep}} - \log \left( \exp(u_c^{\text{dep}}) + \sum_{1 \le j \le m, j \ne c} \exp(\ell_j^{\text{dep}}) \right). \tag{13}$$

While we could directly compute this expression, it is difficult to vectorize. Instead, with some rearranging, we get

$$u^{\text{res}} = u_c^{\text{dep}} - \log \left( \exp(u_c^{\text{dep}}) - \exp(\ell_c^{\text{dep}}) + \sum_{j=1}^m \exp(\ell_j^{\text{dep}}) \right). \tag{14}$$

The second term is the logsumexp of

$$\log \left( \exp(u_c^{\text{dep}}) - \exp(\ell_c^{\text{dep}}) \right) \tag{15}$$

and

$$\text{logsumexp}(\ell^{\text{dep}}). \tag{16}$$

Since we know how to compute logsumexp, this reduces to computing (15). Note that (15) can be rewritten as

$$u_c^{\text{dep}} + \log\left(1 - \exp(\ell_c^{\text{dep}} - u_c^{\text{dep}})\right) \tag{17}$$

by adding and subtracting $u_c^{\text{dep}}$. To compute this quantity, we consider two cases:

1. $u_c^{\text{dep}} \gg \ell_c^{\text{dep}}$. Here we use the fact that stable methods exist to compute $\log1\text{p}(x) = \log(1+x)$ for $x$ close to 0. We compute the desired value as

$$u_c^{\text{dep}} + \log 1p(-\exp(\ell_c^{\text{dep}} - u_c^{\text{dep}})),$$

since $\exp(\ell_c^{\text{dep}} - u_c^{\text{dep}})$ will be close to 0.

2. $u_c^{\text{dep}}$ close to $\ell_c^{\text{dep}}$. Here we use the fact that stable methods exist to compute $\text{expm1}(x) = \exp(x) - 1$ for $x$ close to 0. We compute the desired value as

$$u_c^{\text{dep}} + \log(-\text{expm1}(\ell_c^{\text{dep}} - u_c^{\text{dep}})),$$

since $\ell_c^{\text{dep}} - u_c^{\text{dep}}$ may be close to 0.

We use case 1 if $u_c^{\text{dep}} - \ell_c^{\text{dep}} > \log 2$, and case 2 otherwise.[5]

**Lower bound.** The lower bound $\ell^{\text{res}}$ is achieved by having the minimum value of $z_c^{\text{dep}}$, and the maximum value of all others. This can be written as:

$$\ell^{\text{res}} = \ell_c^{\text{dep}} - \log\left(\exp(\ell_c^{\text{dep}}) + \sum_{1\le j\le m, j\neq c} \exp(u_j^{\text{dep}})\right). \tag{18}$$

The second term is just a normal logsumexp, which is easy to compute. To vectorize the implementation, it helps to first compute the logsumexp of everything except $\ell_c^{\text{dep}}$, and then logsumexp that with $\ell_c^{\text{dep}}$.

### A.3 Attack surface differences

In Alzantot et al. (2018), the adversary applies replacements one at a time, and the neighborhoods and language model scores are computed relative to the current altered version of the input. This results in a hard-to-define attack surface, as the same word can be replaced many times, leading to semantic drift. We instead pre-compute the allowed substitutions $S(x, i)$ at index $i$ based on the original $x$. We define $S(x, i)$ as the set of $\tilde{x}_i \in N(x_i)$ such that

$$\log P(x_{i-W:i-1}, \tilde{x}_i, x_{i+1:i+W}) \geq \log P(x_{i-W:i+W}) - \delta \tag{19}$$

where probabilities are assigned by a pre-trained language model, and the window radius $W$ and threshold $\delta$ are hyperparameters. We use $W = 6$ and $\delta = 5$. We also use a different language model[6] from Alzantot et al. (2018) that achieves perplexity of 50.79 on the One Billion Word dataset (Chelba et al., 2013). Alzantot et al. (2018) use a different, slower language model, which compels them to use a smaller window radius of $W = 1$.

### A.4 Experimental details

We do not run training for a set number of epochs but do early stopping on the development set instead. For normal training, we early stop on normal development set accuracy. For training with data augmentation, we early stop on the accuracy on the augmented development set. For certifiably robust training, we early stop on the certifiably robust accuracy on the development set. We use the Adam optimizer (Kingma and Ba, 2014) to train all models.

On IMDB, we restrict the model to only use the 50,000 words that are in the vocabulary of the counter-fitted word vector space of Mrkšić et al. (2016). This is because perturbations are not allowed for any words not in this vocabulary, i.e. $N(w) = \{w\}$ for $w \notin V$. Therefore, the model is strongly incentivized to predict based on words outside of this set. While this is a valid way to achieve high certified accuracy, it is not a valid robustness strategy in general. We simply delete all words that are not in the vocabulary before feeding the input to the model.

For SNLI, we use 100-dimensional hidden state for the BOW model and a 3-layer feedforward network. These values were chosen by a hyperparameter search on the dev set. For DECOMPATTN, we use a 300-dimensional hidden state and a 2-layer feedforward network on top of the context-aware vectors. These values were chosen to match Parikh et al. (2016).

[5] See https://cran.r-project.org/web/packages/Rmpfr/vignettes/log1mexp-note.pdf for further explanation.

[6] https://github.com/windweller/l2w

| System | $\kappa$ | Learning Rate | Dropout Prob. | Weight Decay | Gradient Norm Clip Val. | $T^{init}$ |
|---|---|---|---|---|---|---|
| IMDB, BOW | 0.8 | $1 \times 10^{-3}$ | 0.2 | $1 \times 10^{-4}$ | 0.25 | 40 |
| IMDB, CNN | 0.8 | $1 \times 10^{-3}$ | 0.2 | $1 \times 10^{-4}$ | 0.25 | 40 |
| IMDB, LSTM | 0.8 | $1 \times 10^{-3}$ | 0.2 | $1 \times 10^{-4}$ | 0.25 | 20 |
| SNLI, BOW | 0.5 | $5 \times 10^{-4}$ | 0.1 | $1 \times 10^{-4}$ | 0.25 | 35 |
| SNLI, DECOMPATTN | 0.5 | $1 \times 10^{-4}$ | 0.1 | 0 | 0.25 | 50 |

Table 3: Training hyperparameters for training the models. The same hyperparameters were used for all training settings(plain, data augmentation, robust training)

Our implementation of the Decomposable Attention follows the original described in (Parikh et al., 2016) except for a few differences listed below;

- We do not normalize GloVe vectors to have norm 1.
- We do not hash out-of-vocabulary words to randomly generated vectors that we train, instead we omit them.
- We do randomly generate a null token vector that we then train. (Whether the null vector is trained is unspecified in the original paper).
- We use the Adam optimizer (with a learning rate of $1 \times 10^{-4}$) instead of AdaGrad.
- We use a batch size of 256 instead of 4.
- We use a dropout probability of 0.1 instead of 0.2
- We do not use the intra-sentence attention module.

### A.5 Training schedule

In Table 4, we show the effect of holding $\epsilon$ or $\kappa$ fixed during training, as described in Section 5.6. All numbers are on 1000 randomly chosen examples from the IMDB development set. Slowly increasing $\epsilon$ is important for good performance. Slowly increasing $\kappa$ is actually slightly worse than holding $\kappa = \kappa^*$ fixed during training, despite earlier experiments we ran suggesting the opposite. Here we only report certified accuracy, as all models are trained with certifiably robust training, and certified accuracy is much faster to compute for development purposes.

### A.6 Word vector bound sizes

To better understand the effect of $g^{\text{word}}$, we checked whether $g^{\text{word}}$ made interval bound boxes around neighborhoods $N(w)$ smaller. For each word $w$ with $|N(w)| > 1$, and for both the pre-trained vectors $\phi^{\text{pre}}(\cdot)$ and transformed vectors $\phi(\cdot)$, we compute

$$\frac{1}{d}\sum_{i=1}^{d}\frac{1}{\sigma_i}\left(u_w^{\text{word}} - \ell_w^{\text{word}}\right)$$

where $\ell_w^{\text{word}}$ and $u_w^{\text{word}}$ are the interval bounds around either $\{\phi^{\text{pre}}(\tilde{w}) : \tilde{w} \in N(w)\}$ or $\{\phi(\tilde{w}) : \tilde{w} \in N(w)\}$, and $\sigma_i$ is the standard deviation across the vocabulary of the $i$-th coordinate of the embeddings. This quantity measures the average width of the IBP bounds for the word vectors of $w$ and its neighbors, normalized by the standard deviation in each coordinate. On 78.2% of words with $|N(w)| > 1$, this value was smaller for the transformed vectors learned by the CNN on IMDB with robust training, compared to the GloVe vectors. For same model with normal training, the value was smaller only 54.5% of the time, implying that robust training makes the transformation produce tighter bounds. We observed the same pattern for other model architectures as well.

| System | IBP-certified (Lower bound) |
|---|---|
| BOW | 68.8 |
| $\rightarrow$ Fixed $\epsilon$ | 46.6 |
| $\rightarrow$ Fixed $\kappa$ | 69.8 |
| $\rightarrow$ Fixed $\epsilon$ and $\kappa$ | 66.3 |
| CNN | 72.5 |
| $\rightarrow$ Fixed $\epsilon$ | 67.6 |
| $\rightarrow$ Fixed $\kappa$ | 74.5 |
| $\rightarrow$ Fixed $\epsilon$ and $\kappa$ | 65.3 |
| LSTM | 62.5 |
| $\rightarrow$ Fixed $\epsilon$ | 43.7 |
| $\rightarrow$ Fixed $\kappa$ | 63.0 |
| $\rightarrow$ Fixed $\epsilon$ and $\kappa$ | 62.0 |

Table 4: Effects of holding $\epsilon$ and $\kappa$ fixed during training. All numbers are on 1000 randomly chosen IMDB development set examples.

### A.7 Certifying long-term memory

We might expect that LSTMs are difficult to certify with IBP, due to their long computation paths. To test whether robust training can learn recurrent models that track state across many time steps, we created a toy binary classification task where the input is a sequence of words $x_1, \ldots, x_L$, and the label $y$ is 1 if $x_1 = x_L$ and 0 otherwise. We trained an LSTM model that reads the input left-to-right, and tries to predict $y$ with a two-layer feedforward network on top of the final hidden state. To do this task, the model must encode the first word in its state and remember it until the final timestep; a bag of words model cannot do this task. For perturbations, we allow replacing every middle word $x_2, \ldots, x_{L-1}$ with any word in the vocabulary. We use robust training on $4000$ randomly generated examples, where the length of each example is sampled uniformly between 3 and 10. The model obtains $100\%$ certified accuracy on a test set of $1000$ examples, confirming that robust training can learn models that track state across many time steps.

For this experiment, we found it important to first train for multiple epochs with no certified objective, before increasing $\epsilon$ and $\kappa$. Otherwise, the model gets stuck in bad local optima. We trained for $50$ epochs using the normal objective, $50$ epochs increasing $\epsilon$ towards $1$ and $\kappa$ towards $0.5$, then $17$ final epochs (determined by early stopping) with these final values of $\epsilon$ and $\kappa$.[7] We leave further exploration of these learning schedule tactics to future work. We also found it necessary to use a larger LSTM—we used one with 300-dimensional hidden states.

[7] Note that this dataset is much smaller than IMDB and SNLI, so each epoch corresponds to many fewer parameter updates.

# Visualizing and Understanding the Effectiveness of BERT

**Yaru Hao**[†]*, **Li Dong**[‡], **Furu Wei**[‡], **Ke Xu**[†]
[†]Beihang University
[‡]Microsoft Research
{haoyaru@,kexu@nlsde.}buaa.edu.cn
{lidong1,fuwei}@microsoft.com

## Abstract

Language model pre-training, such as BERT, has achieved remarkable results in many NLP tasks. However, it is unclear why the pre-training-then-fine-tuning paradigm can improve performance and generalization capability across different tasks. In this paper, we propose to visualize loss landscapes and optimization trajectories of fine-tuning BERT on specific datasets. First, we find that pre-training reaches a good initial point across downstream tasks, which leads to wider optima and easier optimization compared with training from scratch. We also demonstrate that the fine-tuning procedure is robust to overfitting, even though BERT is highly over-parameterized for downstream tasks. Second, the visualization results indicate that fine-tuning BERT tends to generalize better because of the flat and wide optima, and the consistency between the training loss surface and the generalization error surface. Third, the lower layers of BERT are more invariant during fine-tuning, which suggests that the layers that are close to input learn more transferable representations of language.

## 1 Introduction

Language model pre-training has achieved strong performance in many NLP tasks (Peters et al., 2018; Howard and Ruder, 2018a; Radford et al., 2018; Devlin et al., 2018; Baevski et al., 2019; Dong et al., 2019). A neural encoder is trained on a large text corpus by using language modeling objectives. Then the pre-trained model either is used to extract vector representations for input, or is fine-tuned on the specific datasets.

Recent work (Tenney et al., 2019b; Liu et al., 2019a; Goldberg, 2019; Tenney et al., 2019a) has shown that the pre-trained models can encode syntactic and semantic information of language. However, it is unclear why pre-training is effective on downstream tasks in terms of both trainability and generalization capability. In this work, we take BERT (Devlin et al., 2018) as an example to understand the effectiveness of pre-training. We visualize the loss landscapes and the optimization procedure of fine-tuning on specific datasets in three ways. First, we compute the one-dimensional (1D) loss curve, so that we can inspect the difference between fine-tuning BERT and training from scratch. Second, we visualize the two-dimensional (2D) loss surface, which provides more information about loss landscapes than 1D curves. Third, we project the high-dimensional optimization trajectory of fine-tuning to the obtained 2D loss surface, which demonstrate the learning properties in an intuitive way.

The main findings are summarized as follows. First, visualization results indicate that BERT pre-training reaches a good initial point across downstream tasks, which leads to wider optima on the 2D loss landscape compared with random initialization. Moreover, the visualization of optimization trajectories shows that pre-training results in easier optimization and faster convergence. We also demonstrate that the fine-tuning procedure is robust to overfitting. Second, loss landscapes of fine-tuning partially explain the good generalization capability of BERT. Specifically, pre-training obtains more flat and wider optima, which indicates the pre-trained model tends to generalize better on unseen data (Chaudhari et al., 2017; Li et al., 2018; Izmailov et al., 2018). Additionally, we find that the training loss surface correlates well with the generalization error. Third, we demonstrate that the lower (i.e., close to input) layers of BERT are more invariant across tasks than the higher layers, which suggests that the lower layers learn transferable representations of language. We verify the point by visualizing the loss landscape with respect to different groups of layers.

*Contribution during internship at Microsoft Research.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4143–4152,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

## 2 Background: BERT

We use BERT (Bidirectional Encoder Representations from Transformers; Devlin et al. 2018) as an example of pre-trained language models in our experiments. BERT is pre-trained on a large corpus by using the masked language modeling and next-sentence prediction objectives. Then we can add task-specific layers to the BERT model, and fine-tune all the parameters according to the downstream tasks.

BERT employs a Transformer (Vaswani et al., 2017) network to encode contextual information, which contains multi-layer self-attention blocks. Given the embeddings $\{\mathbf{x}_i\}_{i=1}^{|x|}$ of input text, we concatenate them into $\mathbf{H}^0 = [\mathbf{x}_1, \cdots, \mathbf{x}_{|x|}]$. Then, an $L$-layer Transformer encodes the input: $\mathbf{H}^l = \mathrm{Transformer\_block}_l(\mathbf{H}^{l-1})$, where $l = 1, \cdots, L$, and $\mathbf{H}^L = [\mathbf{h}_1^L, \cdots, \mathbf{h}_{|x|}^L]$. We use the hidden vector $\mathbf{h}_i^L$ as the contextualized representation of the input token $x_i$. For more implementation details, we refer readers to Vaswani et al. (2017).

## 3 Methodology

We employ three visualization methods to understand why fine-tuning the pre-trained BERT model can achieve better performance on downstream tasks compared with training from scratch. We plot both one-dimensional and two-dimensional loss landscapes of BERT on the specific datasets. Besides, we project the optimization trajectories of the fine-tuning procedure to the loss surface. The visualization algorithms can also be used for the models that are trained from random initialization, so that we can compare the difference between two learning paradigm.

### 3.1 One-dimensional Loss Curve

Let $\boldsymbol{\theta}_0$ denote the initialized parameters. For fine-tuning BERT, $\boldsymbol{\theta}_0$ represents the the pre-trained parameters. For training from scratch, $\boldsymbol{\theta}_0$ represents the randomly initialized parameters. After fine-tuning, the model parameters are updated to $\boldsymbol{\theta}_1$. The one-dimensional (1D) loss curve aims to quantify the loss values along the optimization direction (i.e., from $\boldsymbol{\theta}_0$ to $\boldsymbol{\theta}_1$).

The loss curve is plotted by linear interpolation between $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$ (Goodfellow and Vinyals, 2015). The curve function $f(\alpha)$ is defined as:

$$f(\alpha) = \mathcal{J}(\boldsymbol{\theta}_0 + \alpha\boldsymbol{\delta}_1) \tag{1}$$

where $\alpha$ is a scalar parameter, $\boldsymbol{\delta}_1 = \boldsymbol{\theta}_1 - \boldsymbol{\theta}_0$ is the optimization direction, and $\mathcal{J}(\boldsymbol{\theta})$ is the loss function under the model parameters $\boldsymbol{\theta}$. In our experiments, we set the range of $\alpha$ to $[-4, 4]$ and sample 40 points for each axis. Note that we only consider the parameters of BERT in $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$, so $\boldsymbol{\delta}_1$ only indicates the updates of the original BERT parameters. The effect of the added task-specific layers is eliminated by keeping them fixed to the learned values.

### 3.2 Two-dimensional Loss Surface

The one-dimensional loss curve can be extended to the two-dimensional (2D) loss surface (Li et al., 2018). Similar as in Equation (1), we need to define two directions ($\boldsymbol{\delta}_1$ and $\boldsymbol{\delta}_2$) as axes to plot the loss surface:

$$f(\alpha, \beta) = \mathcal{J}(\boldsymbol{\theta}_0 + \alpha\boldsymbol{\delta}_1 + \beta\boldsymbol{\delta}_2) \tag{2}$$

where $\alpha, \beta$ are scalar values, $\mathcal{J}(\cdot)$ is the loss function, and $\boldsymbol{\theta}_0$ represents the initialized parameters. Similar to Section 3.1, we are only interested in the parameter space of the BERT encoder, without taking into consideration task-specific layers. One of the axes is the optimization direction $\boldsymbol{\delta}_1 = \boldsymbol{\theta}_1 - \boldsymbol{\theta}_0$ on the target dataset, which is defined in the same way as in Equation (1). We compute the other axis direction via $\boldsymbol{\delta}_2 = \boldsymbol{\theta}_2 - \boldsymbol{\theta}_0$, where $\boldsymbol{\theta}_2$ represents the fine-tuned parameters on another dataset. So the other axis is the optimization direction of fine-tuning on another dataset. Even though the other dataset is randomly chosen, experimental results confirm that the optimization directions $\boldsymbol{\delta}_1, \boldsymbol{\delta}_2$ are divergent and orthogonal to each other because of the high-dimensional parameter space.

The direction vectors $\boldsymbol{\delta}_1$ and $\boldsymbol{\delta}_2$ are projected onto a two-dimensional plane. It is beneficial to ensure the scale equivalence of two axes for visualization purposes. Similar to the filter normalization approach introduced in (Li et al., 2018), we address this issue by normalizing two direction vectors to the same norm. We re-scale $\boldsymbol{\delta}_2$ to $\frac{\|\boldsymbol{\delta}_1\|}{\|\boldsymbol{\delta}_2\|}\boldsymbol{\delta}_2$, where $\|\cdot\|$ denotes the Euclidean norm. We set the range of both $\alpha$ and $\beta$ to $[-4, 4]$ and sample 40 points for each axis.

### 3.3 Optimization Trajectory

Our goal is to project the optimization trajectory of the fine-tuning procedure onto the 2D loss surface obtained in Section 3.2. Let $\{(d_i^\alpha, d_i^\beta)\}_{i=1}^T$ denote the projected optimization trajectory, where

$(d_i^\alpha, d_i^\beta)$ is a projected point in the loss surface, and $i = 1, \cdots, T$ represents the $i$-th epoch of fine-tuning.

As shown in Equation (2), we have known the optimization direction $\boldsymbol{\delta}_1 = \boldsymbol{\theta}_1 - \boldsymbol{\theta}_0$ on the target dataset. We can compute the deviation degrees between the optimization direction and the trajectory to visualize the projection results. Let $\boldsymbol{\theta}^i$ denote the BERT parameters at the $i$-th epoch, and $\boldsymbol{\delta}^i = \boldsymbol{\theta}^i - \boldsymbol{\theta}_0$ denote the optimization direction at the $i$-th epoch. The point $(d_i^\alpha, d_i^\beta)$ of the trajectory is computed via:

$$v_{cos} = \frac{\boldsymbol{\delta}^i \times \boldsymbol{\delta}_1}{\|\boldsymbol{\delta}^i\| \cdot \|\boldsymbol{\delta}_1\|} \tag{3}$$

$$d_i^\alpha = v_{cos} \frac{\left\|\boldsymbol{\delta}^i\right\|}{\|\boldsymbol{\delta}_1\|} = \frac{\boldsymbol{\delta}^i \times \boldsymbol{\delta}_1}{\|\boldsymbol{\delta}_1\|^2} \tag{4}$$

$$d_i^\beta = \sqrt{(\frac{\|\boldsymbol{\delta}^i\|}{\|\boldsymbol{\delta}_1\|})^2 - (d_i^\alpha)^2} \tag{5}$$

where $\times$ denotes the cross product of two vectors, and $\|\cdot\|$ denotes the Euclidean norm. To be specific, we first compute cosine similarity between $\boldsymbol{\delta}^i$ and $\boldsymbol{\delta}_1$, which indicates the angle between the current optimization direction and the final optimization direction. Then we get the projection values $d_i^\alpha$ and $d_i^\beta$ by computing the deviation degrees between the optimization direction $\boldsymbol{\delta}^i$ and the axes.

## 4 Experimental Setup

We conduct experiments on four datasets: Multi-genre Natural Language Inference Corpus (MNLI; Williams et al. 2018), Recognizing Textual Entailment (RTE; Dagan et al. 2006; Bar-Haim et al. 2006; Giampiccolo et al. 2007; Bentivogli et al. 2009), Stanford Sentiment Treebank (SST-2; Socher et al. 2013), and Microsoft Research Paraphrase Corpus (MRPC; Dolan and Brockett 2005). We use the same data split as in (Wang et al., 2019). The accuracy metric is used for evaluation.

We employ the pre-trained BERT-large model in our experiments. The cased version of tokenizer is used. We follow the settings and the hyper-parameters suggested in (Devlin et al., 2018). The Adam (Kingma and Ba, 2015) optimizer is used for fine-tuning. The number of fine-tuning epochs is selected from $\{3, 4, 5\}$. For RTE and MRPC, we set the batch size to 32, and the learning rate to 1e-5. For MNLI and SST-2, the batch size is 64, and the learning rate is 3e-5.

For the setting of training from scratch, we use the same network architecture as BERT, and randomly initialize the model parameters. Most hyper-parameters are kept the same. The number of training epochs is larger than fine-tuning BERT, because training from scratch requires more epochs to converge. The number of epochs is set to 8 for SST-2, and 16 for the other datasets, which is validated on the development set.

## 5 Pre-training Gets a Good Initial Point Across Downstream Tasks

Fine-tuning BERT on the usually performs significantly better than training the same network with random initialization, especially when the data size is small. Results indicate that language model pre-training objectives learn good initialization for downstream tasks. In this section, we inspect the benefits of using BERT as the initial point from three aspects.

### 5.1 Pre-training Leads to Wider Optima

As described in Section 3.2, we plot 2D training loss surfaces on four datasets in Figure 1. The figures in the top row are the models trained from scratch, while the others in bottom are based on fine-tuning BERT. The start point indicates the training loss of the initialized model, while the end point indicates the final training loss. We observe that the optima obtained by fine-tuning BERT are much wider than training from scratch. A wide optimum of fine-tuning BERT implicates that the small perturbations of the model parameters cannot hurt the final performance seriously, while a thin optimum is more sensitive to these subtle changes. Moreover, in Section 6.1, we further discuss about the width of the optima can contribute to the generalization capability.

As shown in Figure 1, the fine-tuning path from the start point to the end point on the loss landscape is more smooth than training from scratch. In other words, the training loss of fine-tuning BERT tends to monotonously decrease along the optimization direction, which eases optimization and accelerates training convergence. In contrast, the path from random initial point to the end point is more rough, which requires a more carefully tweaked optimizer to obtain reasonable performance.

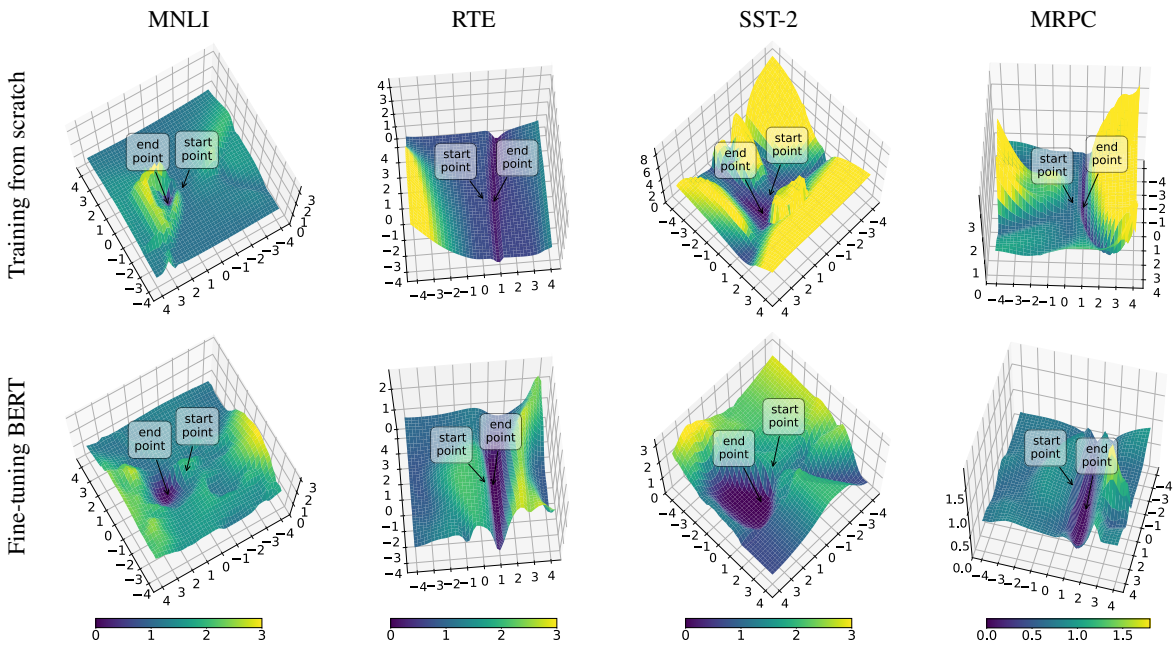


Figure 1: Training loss surfaces of training from scratch (top) and fine-tuning BERT (bottom) on four datasets. We annotate the start point (i.e., initialized model) and the end point (i.e., estimated model) in the loss surfaces. Pre-training leads to wider optima, and eases optimization compared with random initialization.

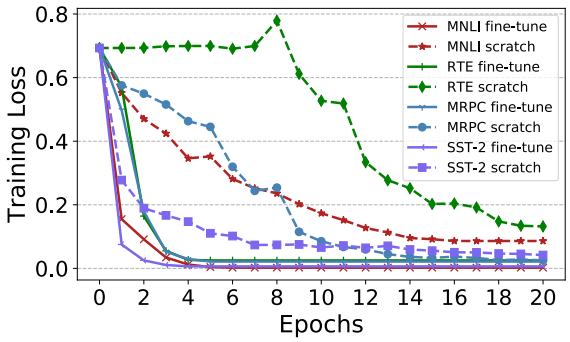


Figure 2: Training loss of fine-tuning BERT and training from scratch on four datasets.

### 5.2 Pre-training Eases Optimization on Downstream Tasks

We fine-tune BERT and train the same network from scratch on four datasets. The learning curves are shown in Figure 2. We find that training from scratch requires more iterations to converge on the datasets, while pre-training-then-fine-tuning converges faster in terms of training loss. We also notice that the final loss of training from scratch tends to be higher than fine-tuning BERT, even if it undergoes more epochs. On the RTE dataset, training the model from scratch has a hard time decreasing the loss in the first few epochs.

In order to visualize the dynamic convergence process, we plot the optimization trajectories using the method described in Section 3.3. As shown in Figure 3, for training from scratch, the optimization directions of the first few epochs are divergent from the final optimization direction. Moreover, the loss landscape from the initial point to the end point is more rough than fine-tuning BERT, we can see that the trajectory of training from scratch on the MRPC dataset crosses an obstacle to reach the end point.

Compared with training from scratch, fine-tuning BERT finds the optimization direction in a more straightforward way. The optimization process also converges faster. Besides, the fine-tuning path is unimpeded along the optimization direction. In addition, because of the wider optima near the initial point, fine-tuning BERT tends to reach the expected optimal region even if it optimizes along the direction of the first epoch.

### 5.3 Pre-training-then-fine-tuning is Robust to Overfitting

The BERT-large model has 345M parameters, which is over-parameterized for the target datasets. However, experimental results show fine-tuning BERT is robust to over-fitting, i.e., the generalization error (namely, the classification error rate on the development set) does not dramatically increase for more training epochs, despite the huge number of model parameters.

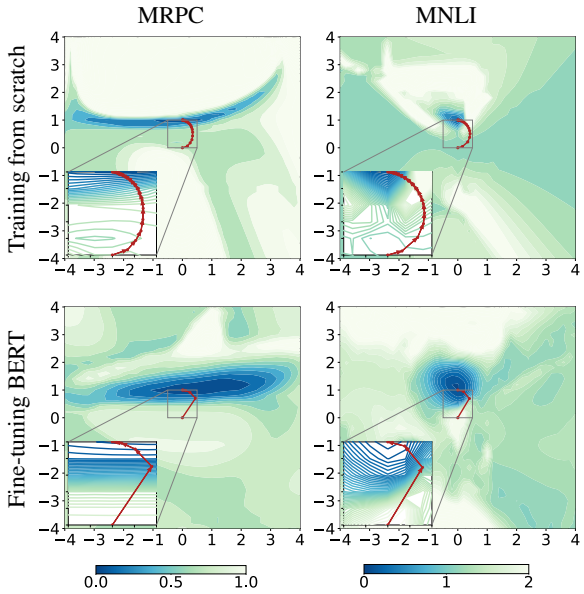


Figure 3: The optimization trajectories on the training loss surfaces of training from scratch (top) and fine-tuning BERT (bottom) on the MRPC and MNLI datasets. Darker color represents smaller training loss.

We use the MRPC dataset as a case study, because its data size is relatively small, which is prone to overfitting if we train the model from scratch. As shown in Figure 4, we plot the optimization trajectory of fine-tuning on the generalization error surface. We first fine-tune the BERT model for five epochs as suggested in (Devlin et al., 2018). Then we continue fine-tuning for another twenty epochs, which still obtains comparable performance with the first five epochs. Figure 4 shows that even though we fine-tune the BERT model for twenty more epochs, the final estimation is not far away from its original optimum. Moreover, the optimum area is wide enough to avoid the model from jumping out the region with good generalization capability, which explains why the pre-training-then-fine-tuning paradigm is robust to overfitting.

## 6 Pre-training Helps to Generalize Better

Although training from scratch can achieve comparable training losses as fine-tuning BERT, the model with random initialization usually has poor performance on the unseen data. In this section, we use visualization techniques to understand why the model obtained by pre-training-then-fine-tuning tends to have better generalization capability.

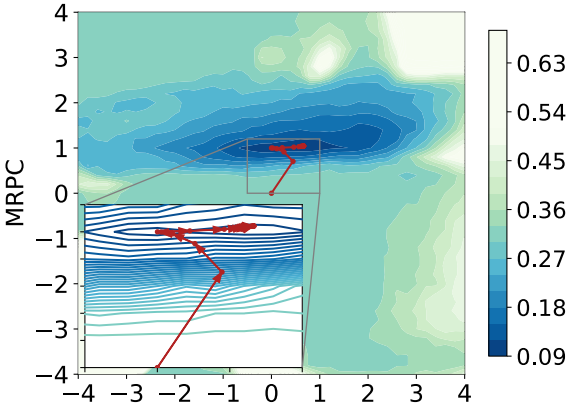


Figure 4: The optimization trajectory of fine-tuning $5 + 20$ epochs on MRPC. The two-dimensional generalization error surface is presented. We find that pre-training-then-fine-tuning is robust to overfitting.

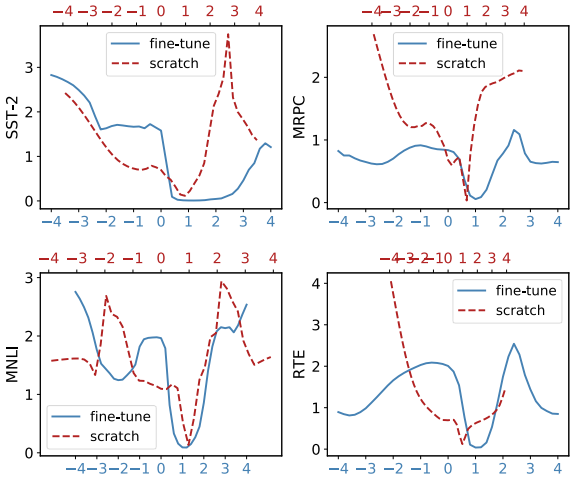


Figure 5: One-dimensional training loss curves. Dash lines represent training from scratch, and solid lines represent fine-tuning BERT. The scales of axes are normalized for comparison. The optima of training from scratch are sharper than fine-tuning BERT.

### 6.1 Wide and Flat Optima Lead to Better Generalization

Previous work (Hochreiter and Schmidhuber, 1997; Keskar et al., 2016; Li et al., 2018) shows that the flatness of a local optimum correlates with the generalization capability, i.e., more flat optima lead to better generalization. The finding inspires us to inspect the loss landscapes of BERT fine-tuning, so that we can understand the generalization capability from the perspective of the flatness of optima.

Section 5.1 presents that the optima obtained by fine-tuning BERT are wider than training from scratch. As shown in Figure 5, we further plot one-dimensional training loss curves of

both fine-tuning BERT and training from scratch, which represents the transverse section of two-dimensional loss surface along the optimization direction. We normalize the scale of axes for flatness comparison as suggested in (Li et al., 2018). Figure 5 shows that the optima of fine-tuning BERT are more flat, while training from scratch obtains more sharp optima. The results indicate that pre-training-then-fine-tuning tends to generalize better on unseen data.

### 6.2 Consistency Between Training Loss Surface and Generalization Error Surface

To further understand the effects of geometry of loss functions to the generalization capability, we make comparisons between the training loss surfaces and the generalization error surfaces on different datasets. The classification error rate on the development set is used as an indicator of the generalization capability.

As shown in Figure 6, we find the end points of fine-tuning BERT fall into the wide areas with smaller generalization error. The results show that the generalization error surfaces are consistent with the corresponding training loss surfaces on the datasets, i.e., smaller training loss tends to decrease the error on the development set. Moreover, the fine-tuned BERT models tend to stay approximately optimal under subtle perturbations. The visualization results also indicate that it is preferred to converge to wider and more flat local optima, as the training loss surface and the generalization error surface are shifted with respect to each other (Izmailov et al., 2018). In contrast, training from scratch obtains thinner optimum areas and poorer generalization than fine-tuning BERT, especially on the datasets with relatively small data size (such as MRPC, and RTE). Intuitively, the thin and sharp optima on the training loss surfaces are hard to be migrated to the generalization surfaces.

For training from scratch, it is not surprising that on larger datasets (such as MNLI, and SST-2) the generalization error surfaces are more consistent with the training loss surfaces. The results suggest that training the model from scratch usually requires more training examples to generalize better compared with fine-tuning BERT.

## 7 Lower Layers of BERT are More Invariant and Transferable

The BERT-large model has 24 layers. Different layers could have learned different granularities or perspectives of language during the pre-training procedure. For example, Tenney et al. (2019a) observe that most local syntactic phenomena are encoded in lower layers while higher layers capture more complex semantics. They also show that most examples can be classified correctly in the first few layers. From above, we conjecture that lower layers of BERT are more invariant and transferable across tasks.

We divide the layers of the BERT-large model into three groups: low layers (0th-7th layer), middle layers (8th-15th layer), and high layers (16th-23rd layer). As shown in Figure 7, we plot the two-dimensional loss surfaces with respect to different groups of layers (i.e., parameter subspace instead of all parameters) around the fine-tuned point. To be specific, we modify the loss surface function in Section 3.2 to $f(\alpha, \beta) = \mathcal{J}(\boldsymbol{\theta}_1 + \alpha\boldsymbol{\delta}_1^G + \beta\boldsymbol{\delta}_2^G)$, where $\boldsymbol{\theta}_1$ represents the fine-tuned parameters, $G \in \{\text{low layers, middle layers, high layers}\}$, and the optimization direction of the layer group is used as the axis. On the visualized loss landscapes, $f(0, 0)$ corresponds to the loss value at the fine-tuned point. Besides, $f(-1, 0)$ corresponds to the loss value with the corresponding layer group rollbacked to its original values in the pre-trained BERT model.

Figure 7 shows that the loss surface with respect to lower layers has the wider local optimum along the optimization direction. The results demonstrate that rollbacking parameters of lower layers to their original values (the star-shaped points in Figure 7) does not dramatically hurt the model performance. In contrast, rollbacking high layers makes the model fall into the region with high loss. This phenomenon indicates that the optimization of high layers is critical to fine-tuning whereas lower layers are more invariant and transferable across tasks.

In order to make a further verification, we rollback different layer groups of the fine-tuned model to the parameters of the original pre-trained BERT model. The accuracy results on the development set are presented in Table 1. Similar to Figure 7, the generalization capability does not dramatically decrease after rollbacking low layers or middle layers. Rollbacking low layers (0th-7th layer)

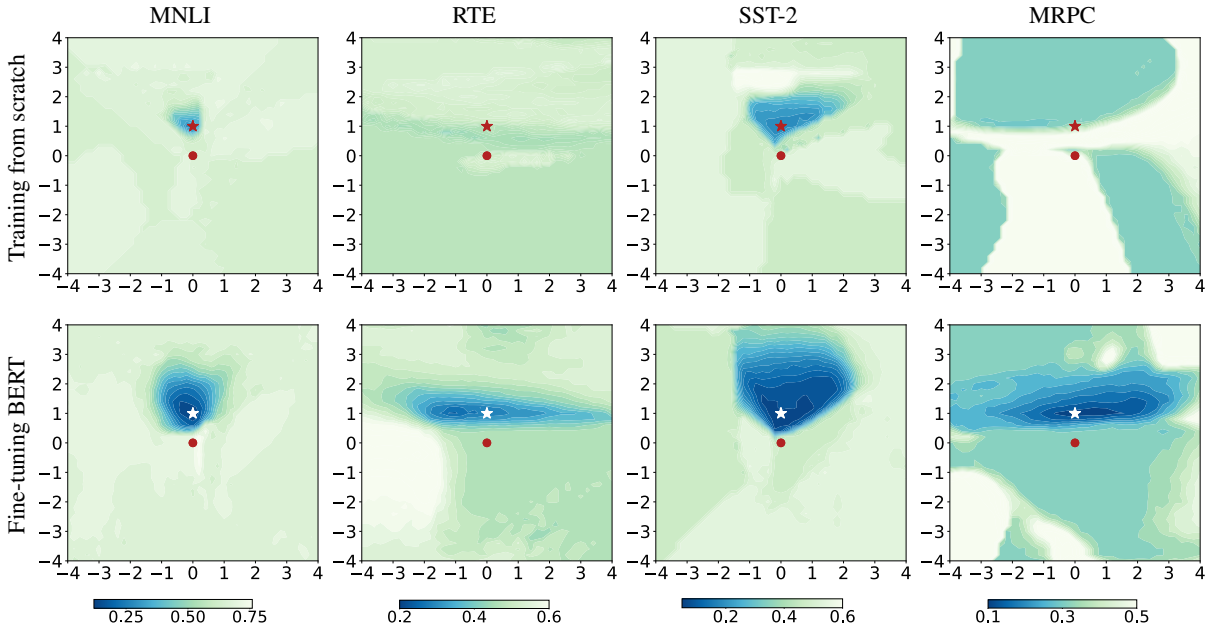


Figure 6: Two-dimensional generalization error surfaces of training from scratch (top) and fine-tuning BERT (bottom). The dot-shaped point and the star-shaped point indicate the generalization errors at the beginning and at the end of the training procedure, respectively.

| Dataset | BERT | Layer Rollback | | |
|---|---|---|---|---|
| | | 0-7 | 8-15 | 16-23 |
| MNLI | 86.54 | 86.73 (+0.19) | 84.71 (-1.83) | 32.85 (-53.88) |
| RTE | 75.45 | 73.29 (-2.16) | 70.04 (-5.41) | 47.29 (-28.16) |
| SST-2 | 94.04 | 93.69 (-0.35) | 93.12 (-0.92) | 59.29 (-34.75) |
| MRPC | 90.20 | 87.99 (-2.21) | 80.15 (-10.05) | 78.17 (-12.03) |

Table 1: Accuracy on the development sets. The second column represents the fine-tuned BERT models on the specific datasets. The last column represents the fine-tuned models with rollbacking different groups of layers.

even improves the generalization capability on the MNLI dataset. By contrast, rollbacking high layers hurts the model performance. Evaluation results suggest that low layers that are close to input learn more transferable representations of language, which makes them more invariant across tasks. Moreover, high layers seem to play a more important role in learning task-specific information during fine-tuning.

## 8 Related Work

Pre-trained contextualized word representations learned from language modeling objectives, such as CoVe (McCann et al., 2017), ELMo (Peters et al., 2018), ULMFit (Howard and Ruder, 2018b), GPT (Radford et al., 2018, 2019), and BERT (Devlin et al., 2018), have shown strong performance on a variety of natural language processing tasks.

Recent work of inspecting the effectiveness of the pre-trained models (Linzen et al., 2016; Kuncoro et al., 2018; Tenney et al., 2019b; Liu et al., 2019a) focuses on analyzing the syntactic and semantic properties. Tenney et al. (2019b) and Liu et al. (2019a) suggest that pre-training helps the models to encode much syntactic information and many transferable features through evaluating models on several probing tasks. Goldberg (2019) assesses the syntactic abilities of BERT and draws the similar conclusions. Our work explores the effectiveness of pre-training from another angle. We propose to visualize the loss landscapes and optimization trajectories of the BERT fine-tuning procedure. The visualization results help us to understand the benefits of pre-training in a more intuitive way. More importantly, the geometry of loss landscapes partially explains why fine-tuning

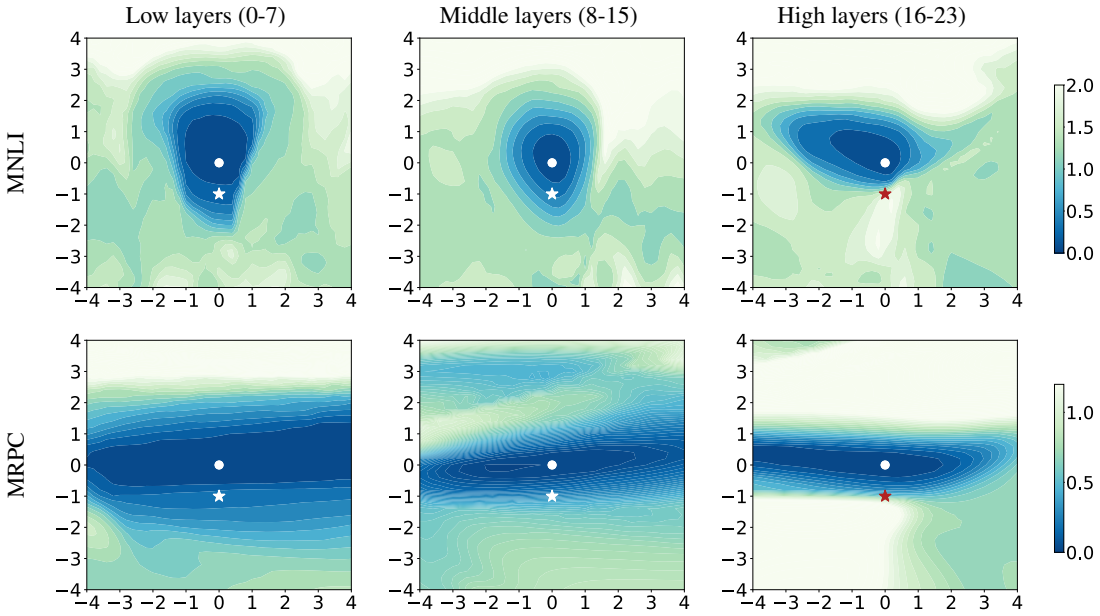


Figure 7: Layer-wise training loss surfaces on the MNLI dataset (top) and the MRPC dataset (bottom). The dot-shaped point represents the fine-tuned model. The star-shaped point represents the model with rollbacking different layer groups. Low layers correspond to the 0th-7th layers of BERT, middle layers correspond to the 8th-15th layers, and high layers correspond to the 16th-23rd layers.

BERT can achieve better generalization capability than training from scratch.

Liu et al. (2019a) find that different layers of BERT exhibit different transferability. Peters et al. (2019) show that the classification tasks build up information mainly in the intermediate and last layers of BERT. Tenney et al. (2019a) observe that low layers of BERT encode more local syntax, while high layers capture more complex semantics. Zhang et al. (2019) also show that not all layers of a deep neural model have equal contributions to model performance. We draw the similar conclusion by visualizing layer-wise loss surface of BERT on downstream tasks. Besides, we find that low layers of BERT are more invariant and transferable across datasets.

In the computer vision community, many efforts have been made to visualize the loss function, and figure out how the geometry of a loss function affects the generalization (Goodfellow and Vinyals, 2015; Im et al., 2016; Li et al., 2018). Hochreiter and Schmidhuber (1997) define the flatness as the size of the connected region around a minimum. Keskar et al. (2016) characterize the definition of flatness using eigenvalues of the Hessian, and conclude that small-batch training converges to flat minima, which leads to good generalization. Li et al. (2018) propose a filter normalization method to reduce the influence of parameter scale, and show that the sharpness of a minimum correlates well with generalization capability. The assumption is also used to design optimization algorithms (Chaudhari et al., 2017; Izmailov et al., 2018), which aims at finding broader optima with better generalization than standard SGD.

## 9 Conclusion

We visualize the loss landscapes and optimization trajectories of the BERT fine-tuning procedure, which aims at inspecting the effectiveness of language model pre-training. We find that pre-training leads to wider optima on the loss landscape, and eases optimization compared with training from scratch. Moreover, we give evidence that the pre-training-then-fine-tuning paradigm is robust to overfitting. We also demonstrate the consistency between the training loss surfaces and the generalization error surfaces, which explains why pre-training improves the generalization capability. In addition, we find that low layers of the BERT model are more invariant and transferable across tasks.

All our experiments and conclusions were derived from BERT fine-tuning. A further understanding of how multi-task training with BERT (Liu et al., 2019b) improves fine-tuning, and how it affects the geometry of loss surfaces are worthy of exploration, which we leave to future

work. Moreover, the results motivate us to develop fine-tuning algorithms that converge to wider and more flat optima, which would lead to better generalization on unseen data. In addition, we would like to apply the proposed methods for other pre-trained models.

## Acknowledgements

The work was partially supported by National Natural Science Foundation of China (NSFC) [Grant No. 61421003].

## References

Alexei Baevski, Sergey Edunov, Yinhan Liu, Luke Zettlemoyer, and Michael Auli. 2019. Cloze-driven pretraining of self-attention networks. *arXiv preprint arXiv:1903.07785*.

Roy Bar-Haim, Ido Dagan, Bill Dolan, Lisa Ferro, and Danilo Giampiccolo. 2006. The second PASCAL recognising textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*.

Luisa Bentivogli, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, and Bernardo Magnini. 2009. The fifth PASCAL recognizing textual entailment challenge. In *In Proc Text Analysis Conference (TAC09*.

Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. 2017. Entropy-SGD: Biasing gradient descent into wide valleys.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Proceedings of the First International Conference on Machine Learning Challenges: Evaluating Predictive Uncertainty Visual Object Classification, and Recognizing Textual Entailment*, MLCW'05, pages 177–190, Berlin, Heidelberg. Springer-Verlag.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. *arXiv preprint arXiv:1905.03197*.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 1–9, Prague. Association for Computational Linguistics.

Yoav Goldberg. 2019. Assessing BERT's syntactic abilities. *CoRR*, abs/1901.05287.

Ian J. Goodfellow and Oriol Vinyals. 2015. Qualitatively characterizing neural network optimization problems. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Flat minima. *Neural Comput.*, 9(1):1–42.

Jeremy Howard and Sebastian Ruder. 2018a. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Jeremy Howard and Sebastian Ruder. 2018b. Universal language model fine-tuning for text classification. In *ACL*. Association for Computational Linguistics.

Daniel Jiwoong Im, Michael Tao, and Kristin Branson. 2016. An empirical analysis of deep network loss surfaces. *CoRR*, abs/1612.04010.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. 2018. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 2016. On large-batch training for deep learning: Generalization gap and sharp minima. *CoRR*, abs/1609.04836.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*, San Diego, CA.

Adhiguna Kuncoro, Chris Dyer, John Hale, Dani Yogatama, Stephen Clark, and Phil Blunsom. 2018. LSTMs can learn syntax-sensitive dependencies well, but modeling structure makes them better. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436, Melbourne, Australia. Association for Computational Linguistics.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. 2018. Visualizing the loss landscape of neural nets. In *Neural Information Processing Systems*.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *CoRR*, abs/1611.01368.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew Peters, and Noah A. Smith. 2019a. Linguistic knowledge and transferability of contextual representations. *CoRR*, abs/1903.08855.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019b. Multi-task deep neural networks for natural language understanding. *CoRR*, abs/1901.11504.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6297–6308.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Matthew Peters, Sebastian Ruder, and Noah A. Smith. 2019. To tune or not to tune? Adapting pretrained representations to diverse tasks.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019a. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R. Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R. Bowman, Dipanjan Das, and Ellie Pavlick. 2019b. What do you learn from context? Probing for sentence structure in contextualized word representations. In *International Conference on Learning Representations*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Chiyuan Zhang, Samy Bengio, and Yoram Singer. 2019. Are all layers created equal? *arXiv preprint arXiv:1902.01996*.

# Topics to Avoid: Demoting Latent Confounds in Text Classification

**Sachin Kumar**[◇] **Shuly Wintner**[♣] **Noah A. Smith**[♡♠] **Yulia Tsvetkov**[◇]
[◇]Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA, USA
[♣]Department of Computer Science, University of Haifa, Haifa, Israel
[♡]Paul G. Allen School of Computer Science & Engineering,
University of Washington, Seattle, WA, USA
[♠]Allen Institute for Artificial Intelligence, Seattle, WA, USA
sachink@cs.cmu.edu, shuly@cs.haifa.ac.il
nasmith@cs.washington.edu, ytsvetko@cs.cmu.edu

## Abstract

Despite impressive performance on many text classification tasks, deep neural networks tend to learn frequent superficial patterns that are specific to the training data and do not always generalize well. In this work, we observe this limitation with respect to the task of *native language identification*. We find that standard text classifiers which perform well on the test set end up learning topical features which are confounds of the prediction task (e.g., if the input text mentions Sweden, the classifier predicts that the author's native language is Swedish). We propose a method that represents the latent topical confounds and a model which "unlearns" confounding features by predicting both the label of the input text and the confound; but we train the two predictors adversarially in an alternating fashion to learn a text representation that predicts the correct label but is less prone to using information about the confound. We show that this model generalizes better and learns features that are indicative of the writing style rather than the content.[1]

## 1 Introduction

Text classification systems based on neural networks are biased towards learning frequent spurious correlations in the training data that may be confounds in the actual classification task (Leino et al., 2019). A major challenge in building such systems is to discover features that are not just correlated with the signals in the training data, but are true indicators of these signals, and therefore generalize well.

For example, Kiritchenko and Mohammad (2018) found that sentiment analysis systems implicitly overfit to demographic confounds, systematically amplifying the intensity ratings of posts written by women. Zhao et al. (2017) showed that visual semantic role labeling models implicitly capture actions stereotypically associated with men or women (e.g., *women are cooking* and *men are fixing a faucet*), and in cases of higher model uncertainty assign stereotypical labels to actions and objects, thereby amplifying social biases found in the training data.

We focus on the task of *native language identification* (L1ID), which aims at automatically identifying the native language (L1) of an individual based on their language production in a second language (L2, English in this work). The aim of this task is to discover stylistic features present in the input that are indicative of the author's L1. However, a model trained to predict L1 is likely to predict that a person is, say, a native Greek speaker, if the texts authored by that person mention Greece, because the training data exhibits such topical correlations (§2).

This problem is the focus of our work, and we address it in two steps. First, we introduce a novel method for representing *latent* confounds. Recent relevant work in the area of domain adaptation (Ganin et al., 2016) and deconfounding for text classification (Pryzant et al., 2018; Elazar and Goldberg, 2018) assumes that the set of confounds is known a priori, and their values are given as part of the training data. This is an unrealistic setting that limits the applicability of such models in real world scenarios. In contrast, we introduce a new method, based on log-odds ratio with Dirichlet prior (Monroe et al., 2008), for identifying and representing latent confounds as probability distributions (§3). Second, we propose a novel alternating learning procedure with multiple adversarial discriminators, inspired by adversarial learning (Goodfellow et al., 2014), that demotes latent confounds and results in textual representations that are invariant to the confounds (§4).

[1]The code is available at: https://github.com/Sachin19/adversarial-classify



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4153–4163,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

Note that these two proposals are task-independent and can be extended to a vast array of text classification tasks where confounding factors are not known a priori. For concreteness, however, we evaluate our approach on the task of L1ID (§5). We experiment with two different datasets: a small corpus of student written essays (Malmasi et al., 2017) and a large and noisy dataset of Reddit posts (Rabinovich et al., 2018). We show that classifiers trained on these datasets without any intervention learn spurious topical correlations that are not indicative of style, and that our proposed deconfounded classifiers alleviate this problem (§6). We present an analysis of the features discovered after demoting these confounds in §7.

The main contributions of this work are:

1. We introduce a novel method for representing and identifying variables which are confounds in text classification tasks.

2. We propose a classification model and an algorithm aimed at learning textual representations that are invariant to the confounding variable.

3. We introduce a novel approach to adversarial training with multiple adversaries, to alleviate the problem of drifting parameters during alternating classifier–adversary optimization.

4. Finally, we analyze some linguistic features that are not only predictive of the author's L1 but are also devoid of topical bias.

## 2 Motivation

We study the general effect of *topical* confounds in text classification. To motivate the need to demote them, we introduce as a case study the L1ID task, in which the goal is to predict the native language of a writer given their texts in L2.

We begin with a subset of the L2-Reddit corpus (Rabinovich et al., 2018), consistsing of Reddit posts by authors with 23 different L1s, most of them European languages. Some of the posts come from Europe-related forums (e.g. r/Europe, r/AskEurope, r/EuropeanCulture), whereas others are from unrelated forums. We view the latter as out-of-domain data and use them to evaluate the generalization of our models. We use a subset of this corpus, with only the 10 most frequent L1s, to guarantee a large enough balanced training set. We remove all the posts with fewer than 50 words and sample the dataset to obtain a balanced distribution of labels: from this balanced dataset, we randomly sample 20% of examples from each class and divide them equally to create development and test sets. In total, there are around 260,000 examples in the training set and 32,000 examples each in the development, the in-domain test set, and the out-of-domain test set.

We trained a standard (non-adversarial) classifier, with a bidirectional LSTM encoder followed by two feedforward layers with a $\tanh$ activation function and a softmax in the final layer (full experimental details are given in §5.2). We refer to this model as NO-ADV. The results are shown in Table 1. Notice the huge drop in accuracy on the out-of-domain data, which indicates that the model is learning topical features.

To further verify this claim, we used *log-odds ratio with Dirichlet prior* (Monroe et al., 2008)—a common way to identify words that are statistically overrepresented in a particular population compared to others—to identify the top-$K$ words that were most strongly associated with a specific L1 in the training set. (We refer the reader to (Monroe et al., 2008) for the details about the algorithm.) We experimented with $K \in \{20, 50, 100, 200\}$. Table 2 shows the top-10 words in each class; observe that almost all of these words are geographical (hence, topical) terms that have nothing to do with the L1.

Next, we masked such topical words (by replacing them with a special token) and evaluate the trained classifier on masked test sets. Accuracy (Table 1) degrades on both the in-domain and out-of-domain sets, even when only 20 words are removed. The drop in accuracy with the out-of-domain dataset is smaller since these data do not include many instances where the presence of topical words would help in identifying the label. These experiments confirm our hypothesis that the baseline classifier is primarily learning topical correlations, and motivate the need for a deconfounded classification approach which we describe next.

| | **In-Domain** | **Out-of-Domain** |
|---|---|---|
| **NO-ADV** | 52.5 | 25.7 |
| **+MASK TOP-20** | 32.8 | 21.0 |
| **+MASK TOP-50** | 31.6 | 20.4 |
| **+MASK TOP-100** | 30.1 | 19.7 |
| **+MASK TOP-200** | 28.5 | 18.7 |

Table 1: Motivation: accuracy (%) of L1ID on the L2-Reddit dataset.

## 3 Representing Confounds

Latent Dirichlet allocation (LDA; Blei et al., 2003) is a probabilistic generative model for discovering abstract topics that occur in a collection of documents. Under LDA, each document can be considered a mixture of a small (fixed) number of topics—each represented as a distribution over words—and each word's presence is assumed to be attributed to one of the document's topics. More precisely, LDA assigns each document a probability distribution over a fixed number of topics $K$.

LDA topics are known to be poor features for classification (McAuliffe and Blei, 2008), indicating that they do not encode all the topical information. Moreover, they can encode information which is not actually topical and can be a useful L1 marker. Motivated by our case study (§2), we propose a novel method to represent topic distributions, based on log-odds scores (Monroe et al., 2008), and compare it to LDA as a baseline.

For each class label $y$ and each word type $w$, we calculate a log-odds score $lo(w, y) \in \mathbb{R}$. The higher this score, the stronger the association between the class and the word. As we saw in §2, the highest scored words are mostly topical and hence constitute superficial features which we want the classification model to "unlearn." We therefore define a distribution which assigns high probability to a document containing these high scoring words. For a label $y \in \mathcal{Y}$ and an input document $x = \langle w_1, \ldots, w_n \rangle$, we define $p(y \mid x)$:

$$p(y \mid x) \propto p(y) \cdot p(x \mid y) = p(y) \cdot \prod_{i=1}^{n} p(w_i \mid y)$$

The above expansion assumes a bag of words representation. When the dataset is balanced, $p(y)$ is equal for each label and can be omitted. Finally, we define $p(w_i \mid y) \propto \sigma(lo(w_i, y))$, where $\sigma(.)$ is the sigmoid function, which squashes the log-odds scores (whose values are in $\mathbb{R}$) to the range $[0, 1]$. We normalize the sigmoid values over the vocabulary to convert them to a probability distribution. In this distribution, the number of "topics" equals the number of labels, $m$.

## 4 Deconfounded Text Classification

We now formalize the task setup and the classification model. We are given $N$ labeled documents in the training set $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$, where $x_i$ is a document with label $y_i \in \mathcal{Y}$, where $m = |\mathcal{Y}|$ is the number of labels. For each document $x_i$, we represent latent (topical) confounds—domain-specific and superficial document features—as a $K$-dimensional multinomial distribution $t_i \in \{(t_1, \ldots, t_K) \mid \sum_{j=1}^{K} t_j = 1\}$. In our task, the confounds are *topics*, so that each $t_j$ represents the proportion of document $i$ associated with topic $j$ but these topics are not given a priori. In this work, the number of topics $K$, equals $m$, but the methods presented in this work are valid for any number of topics.

Our goal is to train a classifier $f$, parameterized by $\theta$, which learns to accurately predict the target label, while ignoring superficial topical correlations present in the training set. That is, for a text $x$ we wish to predict $\hat{y} = f_\theta(x)$ which doesn't encode any information about $t$. Following Ganin et al. (2016), Pryzant et al. (2018), and Elazar and Goldberg (2018), we input $x$ to an encoder neural network $h(x; \theta_h)$ to obtain a hidden representation $\mathbf{h}_x$ (see Figure 1), followed by two feedforward networks: (1) $c(h(x); \theta_c)$ to predict the label $y$; and (2) an adversary network $\text{adv}(h(x); \theta_a)$ to predict the topics. Departing from prior work which used predefined binary confounds, our adversary predicts the topic distribution $t$. If $\mathbf{h}_x$ does not encode any information to predict $t$, then $c(h(x))$ will not depend on $t$. Concretely, we want to optimize the following quantity:

$$\min_{c,h} \frac{1}{N} \sum_{i=1}^{N} \text{CE}(c(h(x_i)), y_i) + \text{CE}(\text{adv}_h^*(h(x_i)), \mathbb{U}_K)$$

where CE denotes cross-entropy loss, and

$$\text{adv}_h^* = \arg\min_{\text{adv}} \frac{1}{N} \sum_{i=1}^{N} \text{CE}(\text{adv}(h(x_i)), t_i),$$

and $\mathbb{U}_K = (\frac{1}{K}, \ldots, \frac{1}{K})$. This objective seeks a representation $\mathbf{h}_x$ which is maximally predictive of the class label but not of the topical distribution (ideally, it should output a uniform topic distribution for every input).

### 4.1 Learning Schedule: Alternating Optimization of Classifier and Adversary

In practice, this optimization is done in an alternating fashion by minimizing the following two

| | |
|---|---|
| **English** | ireland irish british britain russia scotland england states american london brexit |
| **Finnish** | finland finnish finns helsinki swedish finn nordic sweden sauna nokia estonian |
| **French** | french france paris sarkozy macron fillon hollande gaulle hamon marine valls breton |
| **German** | german germany austria merkel refugees asylum germans bavaria austrian berlin also |
| **Greece** | greek greece greeks syriza macedonia athens turkey macedonians fyrom turkish ancient |
| **Dutch** | dutch netherlands amsterdam wilders rotterdam holland rutte belgium bike hague |
| **Polish** | poland polish poles warsaw lithuanian lithuania judges jews ukranians imho tusk |
| **Romanian** | romania romanian romanians moldova bucharest hungarian hungarians transistria |
| **Spanish** | spain catalan spanish catalonia catalans madrid barcelona independence spaniards |
| **Swedish** | sweden swedish swedes stockholm swede malmo danish nordic denmark finland |

Table 2: Top words based on log-odds scores for each label in the L2-Reddit dataset.

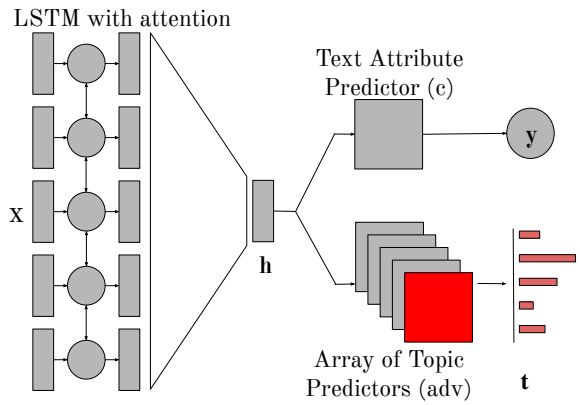


(a) Weights of the LSTM and of the discriminator are fixed. A new topic predictor is trained by minimizing the cross entropy of the output and the distribution of the input document over latent topics as described in §3.

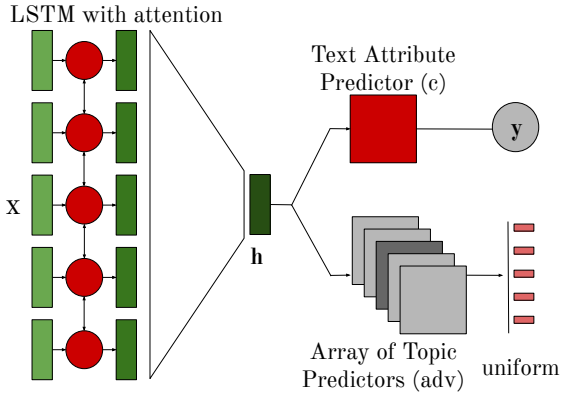


(b) Weights of all the topic predictors are fixed, but the encoder is trained. The model is jointly minimizing the cross-entropy of the classifier and encouraging the topic predictor toward uniformity.

Figure 1: We alternate between training the topic predictor (left; (1)) and the deconfounded classifier/encoder (right; (2)). Pretraining is not shown in the figure.

quantities:

$$\min_{\text{adv}} \frac{1}{N} \sum_{i=1}^{N} \text{CE}(\text{adv}(h(x_i)), t_i) \quad (1)$$

$$\min_{c,h} \frac{1}{N} \sum_{i=1}^{N} \text{CE}(c(h(x_i)), y_i) + \text{CE}(\text{adv}(h(x_i)), \mathbb{U}_K) \quad (2)$$

The training schedule is critical in adversarial setups where the loss has two competing terms (Mescheder et al., 2018; Arjovsky and Bottou, 2017; Roth et al., 2017); here, these terms minimize classification loss while maximizing the topic prediction loss. Algorithm 1 details our proposed alternating learning procedure.

Inspired by generative adversarial networks (GANs; Goodfellow et al., 2014), the training procedure alternates between training the classifier and the adversary (see Figure 1). First (*pretraining*), we train the encoder along with the classifier using only classification loss, until convergence. After pretraining, $\mathbf{h}_x$ has encoded topical information which it uses for classification (as shown in our analysis in §2). Now, we train only $\text{adv}(h(x))$ to (accurately) predict $t$, keeping the parameters of $h(.)$ fixed. Once $\text{adv}(.)$ is trained, it should be able to successfully extract a topic distribution from $\mathbf{h}_x$ (topic training, see Figure 1a). The goal now is to modify $\mathbf{h}_x$ in such a way that $\text{adv}(\mathbf{h}_x)$ produces a uniform distribution (that is, fooling the adversary; similar to fooling the discriminator in GANs). We do that by keeping the weights of $\text{adv}(.)$ fixed, and training the network to produce the class label and a uniform topic distribution (*topic forgetting*, see Figure 1b). We then repeat this procedure for a fixed number of steps which was tuned using the validation set.

**Algorithm 1:** Alternating optimization of classifier and adversary.

**Result:** $\theta_h, \theta_c, \theta_{a_1}, \ldots, \theta_{a_T}$
Randomly initialize $\theta_h, \theta_c$;
**while** *not converged* **do**
  Sample a minibatch of $b$ training samples;
  Update $\theta_h$ and $\theta_c$ using gradients with respect to $\frac{1}{b}\sum_{i=1}^{b} \mathrm{CE}(c(h(x_i)), y_i)$.;
**end**
$j = 1$;
**for** *number of training iterations* $T$ **do**
  Randomly initialize $\theta_{a_j}$;
**end**
**for** $t$ *steps* **do**
  Sample a minibatch of $b$ training samples;
  Fix $\theta_h$ and $\theta_c$, update $\theta_{a_j}$ using gradients with respect to $\frac{1}{b}\sum_{i=1}^{b} \mathrm{CE}(\mathrm{adv}_{\theta_{a_j}}(h(x_i)), t_i)$;
**end**
**for** $c$ *steps* **do**
  Sample a minibatch of $b$ training samples;
  Fix $\theta_{a_u}$ for $u \in_R \{1, \ldots, j\}$ and update $\theta_c$ and $\theta_h$ using gradients with respect to $\frac{1}{b}\sum_{i=1}^{b} \mathrm{CE}(c(h(x_i)), y_i) + \mathrm{CE}(\mathrm{adv}_{\theta_{a_u}}(h(x_i)), \mathbb{U}_K)$;
**end**
$j \longleftarrow j + 1$;

### 4.2 Multiple Adversaries

In our experiments, we observe that after every "topic forgetting" stage, $\mathrm{adv}(.)$ does end up producing a uniform distribution, but in the next "topic training" phase, $\mathrm{adv}(.)$ is able to reproduce the topical distribution accurately. This is because, during "topic forgetting," the classifier does not really forget the topics in $\mathbf{h}_x$; it just encodes them in a different way.[2] This is a general problem in setups with alternating classifier-adversary optimization. To solve this issue, we propose using *multiple adversaries*, inspired by the "experience replay" approach used in reinforcement learning (O'Neill et al., 2010; Mnih et al., 2015). During the $i$th "topic training" phase, we train a new adversary $\mathrm{adv}_i$ (with parameters $\theta_{a_i}$ instead of retraining only one adversary over and over again. In the next "topic forgetting" phase, at each training step we pick $\mathrm{adv}_j$ at random from the pool of previously learned adversaries, $j \in_R \{1, \ldots, i\}$. By using multiple adversaries, we make it difficult for the classifier to encode topical information anywhere.

[2] We observe this in our analysis where the most salient features encoded after pretraining and topic forgetting phrase are the same.

## 5 Experimental Setup

### 5.1 Datasets

We evaluate our topical confound demotion method on the L1ID task. We show experiments with two datasets where L2 is English: the L2-Reddit dataset described in §2, and TOEFL17, a collection of essays authored by non-native English speakers who apply for academic studies in the US (Malmasi et al., 2017). This corpus reflects eleven L1s: Arabic, Chinese, French, German, Hindi, Italian, Japanese, Korean, Spanish, Telugu, and Turkish. The training data include 11,000 authors (1,000 per L1) and the development set has 1,100 essays per L1. We evaluate on the development set. Each essay is also marked with a prompt ID which was given to the authors to write the essay. There are 8 prompts in total, based on which we construct 8 versions of train and test set. In each version, we remove essays marked with one of the prompts from both the train and the development sets, and consider the removed essays from the development set an "out-of-domain" test set. We refer to the version where prompt "P$K$" is out-of-domain as "–P$K$" in the results (Table 3), $K \in \{0, \ldots, 7\}$.

### 5.2 Implementation Details

We tokenized and lowercased all the text using spaCy. Limiting our vocabulary to the most frequent 30,000 words in the training data, we replaced all out-of-vocabulary words with "UNK." We encoded each word using a word embedding layer (initialized at random and learned) and passed these embeddings to a bidirectional LSTM encoder (one layer for each direction) with attention ($h(x)$; Pryzant et al., 2018). Each LSTM layer had a hidden dimension of 128. We used two layered feed forward networks with a $\tanh$ activation function in the middle layer (of size 256), followed by a softmax in the final layer, as $c(.)$ and $\mathrm{adv}(.)$.

### 5.3 Baselines

We consider several baselines that are intended to capture the stylistic features of the texts, explicitly avoiding content.

**Linear classifier with content-independent features (LR)** Replicating Goldin et al. (2018), we trained a logistic regression classifier with three types of features: function words, POS trigrams, and sentence length, all of which are reflective of the style of writing. We deliberately avoided using content features (e.g., word frequencies).

**Classification with no adversary on masked texts (LO-TOP-$K$)** We mask the top-$K$ words (based on log-odds scores) in *both* the train and the test sets (as in §2); we train the classification model again without training $\mathrm{adv}(.)$. After masking the top words, we expect patterns of writing style (and, therefore, L1) to become more apparent.

**Adversarial training with gradient reversal (GR-LO)** A common method of learning a confound-invariant representations is to use a gradient reversal layer (Beutel et al., 2017; Ganin et al., 2016; Pryzant et al., 2018; Elazar and Goldberg, 2018). The output of the encoder, $\mathbf{h}_x$, is passed through this layer before applying $\mathrm{adv}(.)$. This training setup usually proves too difficult to optimize, and often results in poor performance. That is, even if the performance of $\mathrm{adv}(.)$ is weak, $\mathbf{h}_x$ still ends up leaking information about the confound (Lample et al., 2019; Elazar and Goldberg, 2018). In the forward pass, this layer acts as identity whereas in the backward pass it multiplies the gradient values by $-\lambda$, essentially reversing the gradients before they go into the encoder. $\lambda$ controls the intensity of the reversal (we used $\lambda = 0.2$).

**LDA topics as confounds (ALT-LDA)** We trained LDA on the training set and for each example in the training set, generated a probability distribution (over 50 topics), and used it as topical confound with our proposed learning setup, alternating classifier-adversary training.

## 6 Results

### 6.1 TOEFL17 Dataset

We begin with experiments on the TOEFL17 dataset, where predicting L1 is an easier task due to the lower proficiency of the authors. Table 3 reports the accuracy of our proposed model, denoted **ALT-LO**, compared to the logistic regression baseline (**LR**), and two adversarial baselines: one demotes latent log-odds-based topics via gradient reversal (**GR-LO**), and another uses our proposed novel learning procedure but demotes baseline LDA topics (**ALT-LDA**). We report both in-domain accuracy and out-of-domain results; the latter is obtained by averaging the accuracy of each set "–P$K$" over $K \in \{0, \ldots, 7\}$.

| | In-Domain | Out-of-Domain |
|---|---|---|
| **LR** | 55.3 | 50.9 |
| **GR-LO** | 12.7 | 13.6 |
| **ALT-LDA** | 59.1 | 50.1 |
| **ALT-LO** | **61.9** | **60.4** |

Table 3: Classification accuracy with topic-demoting methods, TOEFL dataset.

Our model strongly outperforms all baselines that demote confounds, in both classification setups. We observe in our experiments that gradient reversal is especially unstable and hyperparameter sensitive: it has been shown to work well with categorical confounds like domain type or binary gender, but in demoting continuous outputs like a topic distribution, we observe it is not effective. The proposed alternating training with multiple discriminators obtains better results, and replacing LDA with log-odds-based topics also improves both in-domain and (much more substantially) out-of-domain predictions, confirming the effectiveness of our proposed innovations.

A vanilla classifier without demoting confounds (denoted in §2 as NO-ADV) yields in-domain and out-of-domain accuracies of 62.0 and 58.3, respectively. We would expect that the better generalization power of our proposed model would come at a price of lower accuracy in-domain. Our goal is to capture the true signals of L1, rather than superficial patterns that are more frequent in the data and artificially boost the performance in NO-ADV settings. This is indeed what we observe.

For example, the text "…i agree with you on the prolonged war if the plc heartland (poland proper) was not as rich as it was i dont really see how we would been …" in the dataset is labeled as "Polish" instead of the gold label "Swedish" by the NO-ADV classifier, likely because of the mention of the term "poland", but the ADV-LO model predicts it correctly since it likely picks on other features that indicate non-fluency, like "we would been". Such naive classification errors become especially costly in making predictions about peo-

ple's demographic attributes: ethnicity, which often correlates with L1, but also gender, race, religion, and others (Hardt et al., 2016; Beutel et al., 2017).

### 6.2 L2-Reddit Dataset

Next, we experiment with L2-Reddit, a larger and more challenging dataset (since many speakers in the dataset are highly fluent, and the signal of their native language is weaker). The performance of the simple baselines on this dataset is shown in Table 4. The accuracy of the linear classifier is poor (compared to Table 1), perhaps because it fails to capture some contextual features learned by the neural network models. With LO-TOP-20, the performance on both test sets improves. It slightly degrades when more words are removed, perhaps because some words indicative of L1 are also removed.

| | In-Domain | Out-of-Domain |
|---|---|---|
| **LR** | 21.2 | 18.5 |
| **LO-TOP-20** | 38.7 | 21.9 |
| **LO-TOP-50** | 36.4 | 21.4 |
| **LO-TOP-100** | 35.8 | 21.2 |
| **LO-TOP-200** | 34.7 | 20.8 |

Table 4: Baseline classification accuracy on L2-Reddit.

Finally, we evaluate the impact of our novel training procedure and the quality of our proposed topical confound identification method. We compare our proposed solution, denoted ALT-LO, with two alternatives, as before, one with a different learning setup (GR-LO) and one with a different confound representation (ALT-LDA). Table 5 summarizes the results: our proposed learning procedure ALT-LO performs better than both the alternatives. Unsurprisingly, the model trained with gradient reversal (GR-LO) performs particularly poorly; this was our primary motivation to explore better learning techniques.

| | In-Domain | Out-of-Domain |
|---|---|---|
| **GR-LO** | 22.5 | 15.7 |
| **ALT-LDA** | 46.2 | 21.9 |
| **ALT-LO** | **48.8** | **22.9** |

Table 5: Classification accuracy with topic-demoting methods, L2-Reddit dataset.

To further confirm that the ALT-LO model is not learning topical features, we repeat the experiment presented in Table 1—masking the top $K$ topical words (based on log-odds scores) from the test sets, but not retraining the models—now, with our proposed model ALT-LO. Table 6 shows that in contrast to standard models that do not demote topical confounds (as in Table 1), there is less degradation in the performance of ALT-LO. We conjecture that our model is stable to demoting topics because it learns relevant stylistic features, rather than spurious correlations.

| | In-Domain | Out-of-Domain |
|---|---|---|
| **ALT-LO** | 48.8 | 22.9 |
| **+MASK TOP-20** | 38.7 | 21.6 |
| **+MASK TOP-50** | 36.2 | 21.5 |
| **+MASK TOP-100** | 33.5 | 21.2 |
| **+MASK TOP-200** | 31.9 | 20.4 |

Table 6: Accuracy on the L2-Reddit dataset; the proposed model (ALT-LO) with different settings of the test sets.

## 7 Analysis

We present an analysis of what the models are learning, based on words they attend to for classification. We focus on the L2-Reddit dataset.

Following Pryzant et al. (2018), we generated a lexicon of most attended words by (1) running the model on the test set and saving the attention score for each word; and (2) for each word, computing its average attentional score and selecting the top-$k$ words based on this score.

What emerges from this lexicon (Table 7) is a dramatic difference between the top indicative words in the various models. Whereas in the baseline model *all* the most indicative words are proper nouns, the ALT-LO model highlights exclusively function words. The proper nouns in the baseline model are all geographical terms directly associated with the L1s reflected in the L2-Reddit dataset: they are easy giveaways of the authors' L1s, but they are meaningless linguistically. In contrast, the function words highlighted in the ALT-LO model are mostly prepositions and determiners; it is well known that nonnative speakers are challenged by the use of prepositions (in any L2, English included). The distribution of determiners is also a challenge for nonnatives, and

the correct usage of *the* in particular is quite hard for learners to master. These challenges are evident from the most indicative words of our model. Observe also that the LO-TOP-50 model is somewhere in the middle: it includes some proper nouns (including geographical terms such as *eu* or *us*) but also several function words. A more detailed analysis of these observations is left for future work.

Recently, there has been a debate on whether attention can be used to explain model decisions (Serrano and Smith, 2019; Jain and Wallace, 2019; Wiegreffe and Pinter, 2019), we thus present additional analysis of our proposed method based on saliency maps (Ding et al., 2019). Saliency maps have been shown to better capture word alignment than attention probabilities in neural machine translation. This method is based on computing the gradient of the probability of the predicted label with respect to each word in the input text and normalizing the gradient to obtain probabilities. We use saliency maps to generate lexicons similar to the ones generated using attention. As shown in table 8, the top indicative words for baseline and LO-TOP-50 follow a similar pattern as the ones obtained with attention scores. In line with results in Table 7, salient words for ALT-LO are determiners and prepositions. However, saliency maps also reveal that our proposed approach still attends to some geographical terms that were not demoted by our classifier.

## 8 Related Work

**Controlling for confounds in text** Controlling for confounds is an active field of research, especially in the medical domain, where the common solution is to do random trials or propensity score matching (Rosenbaum and Rubin, 1985). Paul (2017) tackled the problem of learning causal associations between word features and class labels using propensity matching for the task of sentiment analysis. This method is not scalable to large text datasets as it involves training a logistic regression model for every word type. Tan et al. (2014) built models to estimate the number of retweets of Twitter messages and addressed confounding factors by matching tweets of the same author and topic. Reis and Culotta (2018) proposed a statistical technique called Pearl's back-door adjustment for text classification (Pearl, 2009). All these works focused on a bag-of-words model with lexical features only.

**Adversarial training in text** Much recent work focuses on learning textual representations that are invariant to selective properties of the text. This work used domain adaptation and transfer learning (Ganin et al., 2016; Tzeng et al., 2014; Xie et al., 2017), either to remove sensitive attributes such as demographic information (Li et al., 2018; Elazar and Goldberg, 2018; Beutel et al., 2017), or to understand costumer behavior for social science applications (Pryzant et al., 2018). Most of the work in this area, however, focuses on cases where these confounds are known in advance and their values are given along with the training data.

**Native language identification** The L1ID task was introduced by Koppel et al. (2005), who worked on the International Corpus of Learner English (Granger, 2003). The same experimental setup was adopted by several other authors (Tsur and Rappoport, 2007; Wong and Dras, 2009, 2011). Since the release of nonnative *TOEFL* essays by the Educational Testing Service (Blanchard et al., 2013), the task gained popularity and this dataset has been used for two L1ID Shared Tasks (Tetreault et al., 2013; Malmasi et al., 2017).

Malmasi and Dras (2017) report that the state of the art is a linear classifier with character $n$-grams and lexical and morphosyntactic features.

The best accuracy under cross-validation on the TOEFL17 dataset, which includes 11 native languages (with a rather diverse distribution of language families), was 85.2%.

The above works all identify the L1 of *learners*. Identifying the native language of advanced, fluent speakers is a much harder task. Goldin et al. (2018) addressed this task, using the L2-Reddit dataset with as many as 23 different L1s, all of them European and many which are typologically close, which makes the task even harder. They experimented with a variety of features, using logistic regression as the classifier, and achieved results as high as 69% accuracy with cross-validation; however, when testing their classifier outside the domain it was trained on (Reddit forums focusing on European issues), accuracy dropped to 36%.

## 9 Conclusion

We introduced a method to represent unknown confounds in text classification using topic models and log-odds scores, and a new general method

| | |
|---|---|
| **NO-ADV** | sweden france greece finland poland spain greek germany french eu romania polish dutch german spanish swedish netherlands finnish |
| **LO-TOP-50** | eu 's 're 'm ' & uk us because 've am its nt english these usa nt here 'll especially correct pis de within |
| **ALT-LO** | the in to of that a i is and 't as from with by ? on but & they are about at because like was would have you |

Table 7: The highest scoring words in lexicons generated using attention scores.

| | |
|---|---|
| **NO-ADV** | poland greek romania greece france spain french sweden finland polish dutch spanish netherlands finnish german |
| **LO-TOP-50** | on 're even 'd up less things 'll doesn living majority sense talk level 've rights took number north |
| **ALT-LO** | the of to i a in greece romania france finland that for is french & you 't finnish |

Table 8: The highest scoring words in lexicons generated using saliency maps.

with alternating optimization to learn textual representations which are invariant of confounds. We evaluated the proposed solution on the task of native language identification, and showed that it learns to make predictions using stylistic features, rather than focus on topical information.

The learning procedure we presented is general and applicable to other tasks that require learning invariant representations with respect to some attribute of text (some of which are discussed in §8). We plan to evaluate our proposed solution on other tasks where topics can be latent confounds, like predicting gender bias (Voigt et al., 2018). We leave this exploration for future work.

## Acknowledgments

The authors acknowledge helpful input from the anonymous reviewers. This work was supported in part by NSF grants IIS-1812327 and IIS-1813153, by grant no. 2017699 from the United States-Israel Binational Science Foundation (BSF), and by grant no. LU 856/13-1 from the Deutsche Forschungsgemeinschaft. Finally, the authors also thank Anjalie Field, Biswajit Paria, Ella Rabinovich, and Gili Goldin for helpful discussions.

## References

Martin Arjovsky and Léon Bottou. 2017. Towards principled methods for training generative adversarial networks. In *Proc. ICLR*.

Alex Beutel, Jilin Chen, Zhe Zhao, and Ed H Chi. 2017. Data decisions and theoretical implications when adversarially learning fair representations. In *Proc. FATML*.

Daniel Blanchard, Joel Tetreault, Derrick Higgins, Aoife Cahill, and Martin Chodorow. 2013. TOEFL11: A corpus of non-native English. *ETS Research Report Series*.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *JMLR*.

Shuoyang Ding, Hainan Xu, and Philipp Koehn. 2019. Saliency-driven word alignment interpretation for neural machine translation. In *Proceedings of the Fourth Conference on Machine Translation*.

Yanai Elazar and Yoav Goldberg. 2018. Adversarial removal of demographic attributes from text data. In *Proc. EMNLP*.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *JMLR*.

Gili Goldin, Ella Rabinovich, and Shuly Wintner. 2018. Native language identification with user generated content. In *Proc. EMNLP*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *NeurIPS*.

Sylviane Granger. 2003. The International Corpus of Learner English: a new resource for foreign language learning and teaching and second language acquisition research. *Tesol Quarterly*.

Moritz Hardt, Eric Price, Nati Srebro, et al. 2016. Equality of opportunity in supervised learning. In *NeurIPS*.

Sarthak Jain and Byron C. Wallace. 2019. Attention is not explanation. In *Proc. NAACL*.

Svetlana Kiritchenko and Saif M. Mohammad. 2018. Examining gender and race bias in two hundred sentiment analysis systems. In *Proc. NAACL*.

Moshe Koppel, Jonathan Schler, and Kfir Zigdon. 2005. Automatically determining an anonymous author's native language. *Intelligence and Security Informatics*.

Guillaume Lample, Sandeep Subramanian, Eric Smith, Ludovic Denoyer, Marc'Aurelio Ranzato, and Y-Lan Boureau. 2019. Multiple-attribute text rewriting. In *Proc. ICLR*.

Klas Leino, Matt Fredrikson, Emily Black, Shayak Sen, and Anupam Datta. 2019. Feature-wise bias amplification. In *Proc. ICLR*.

Yitong Li, Timothy Baldwin, and Trevor Cohn. 2018. Towards robust and privacy-preserving text representations. In *Proc. ACL*.

Shervin Malmasi and Mark Dras. 2017. Native language identification using stacked generalization. ArXiv:1703.06541.

Shervin Malmasi, Keelan Evanini, Aoife Cahill, Joel Tetreault, Robert Pugh, Christopher Hamill, Diane Napolitano, and Yao Qian. 2017. A report on the 2017 native language identification shared task. In *Proc. Workshop on Innovative Use of NLP for Building Educational Applications*.

Jon D. McAuliffe and David M. Blei. 2008. Supervised topic models. In *NeurIPS*.

Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. 2018. Which training methods for GANs do actually converge? In *Proc. ICML*.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature*.

Burt L. Monroe, Michael P. Colaresi, and Kevin M. Quinn. 2008. Fightin' words: Lexical feature selection and evaluation for identifying the content of political conflict. *Political Analysis*.

Joseph O'Neill, Barty Pleydell-Bouverie, David Dupret, and Jozsef Csicsvari. 2010. Play it again: reactivation of waking experience and memory. *Trends in Neurosciences*.

Michael J. Paul. 2017. Feature selection as causal inference: Experiments with text classification. In *Proc. CoNLL*.

Judea Pearl. 2009. *Causality: Models, Reasoning and Inference*. Cambridge University Press.

Reid Pryzant, Kelly Wang, Dan Jurafsky, and Stefan Wager. 2018. Deconfounded lexicon induction for interpretable social science. In *Proc. NAACL*.

Ella Rabinovich, Yulia Tsvetkov, and Shuly Wintner. 2018. Native language cognate effects on second language lexical choice. *TACL*.

Virgile Landeiro Dos Reis and Aron Culotta. 2018. Robust text classification under confounding shift. *Journal of Artificial Intelligence Research*.

Paul R. Rosenbaum and Donald B. Rubin. 1985. Constructing a control group using multivariate matched sampling methods that incorporate the propensity score. *The American Statistician*.

Kevin Roth, Aurelien Lucchi, Sebastian Nowozin, and Thomas Hofmann. 2017. Stabilizing training of generative adversarial networks through regularization. In *NeurIPS*.

Sofia Serrano and Noah A. Smith. 2019. Is attention interpretable? In *Proc. ACL*.

Chenhao Tan, Lillian Lee, and Bo Pang. 2014. The effect of wording on message propagation: Topic- and author-controlled natural experiments on Twitter. In *Proc. ACL*.

Joel Tetreault, Daniel Blanchard, and Aoife Cahill. 2013. A report on the first native language identification shared task. In *Proc. Workshop on Building Educational Applications Using NLP*.

Oren Tsur and Ari Rappoport. 2007. Using classifier features for studying the effect of native language on the choice of written second language words. In *Proc. Workshop on Cognitive Aspects of Computational Language Acquisition*.

Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. 2014. Deep domain confusion: Maximizing for domain invariance. ArXiv:1412.3474.

Rob Voigt, David Jurgens, Vinodkumar Prabhakaran, Dan Jurafsky, and Yulia Tsvetkov. 2018. RtGender: A corpus for studying differential responses to gender. In *Proc. LREC*.

Sarah Wiegreffe and Yuval Pinter. 2019. Attention is not not explanation. In *Proc. EMNLP*.

Sze-Meng Jojo Wong and Mark Dras. 2009. Contrastive analysis and native language identification. In *Proc. Australasian Language Technology Association Workshop*.

Sze-Meng Jojo Wong and Mark Dras. 2011. Exploiting parse structures for native language identification. In *Proc. EMNLP*.

Qizhe Xie, Zihang Dai, Yulun Du, Eduard H. Hovy, and Graham Neubig. 2017. Controllable invariance through adversarial feature learning. In *NeurIPS*.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2017. Men also like shopping: Reducing gender bias amplification using corpus-level constraints. In *Proc. of EMNLP*.

# Learning to Ask for Conversational Machine Learning

**Shashank Srivastava**[1*] **Igor Labutov**[2] **Tom Mitchell**[3]
[1]UNC Chapel Hill, [2]LAER AI
[3]Machine Learning Department, Carnegie Mellon University
ssrivastava@cs.unc.edu, igor.labutov@laer.ai, tom.mitchell@cmu.edu

## Abstract

Natural language has recently been increasingly explored as a medium of supervision for training machine learning models. Here, we explore learning classification tasks using language in a conversational setting – where the automated learner does not simply receive language input from a teacher, but can proactively engage the teacher by asking template-based questions. We experiment with a reinforcement learning framework, where the learner's actions correspond to question types and the reward for asking a question is based on how the teacher's response changes performance of the resulting machine learning model on the learning task. In this framework, learning good question-asking strategies corresponds to asking sequences of questions that maximize the cumulative (discounted) reward, and hence quickly lead to effective classifiers. Empirical analysis shows that learned question-asking strategies can expedite classifier training by asking appropriate questions at different points in the learning process. The approach allows learning using a blend of strategies, including learning from observations, explanations and clarifications.

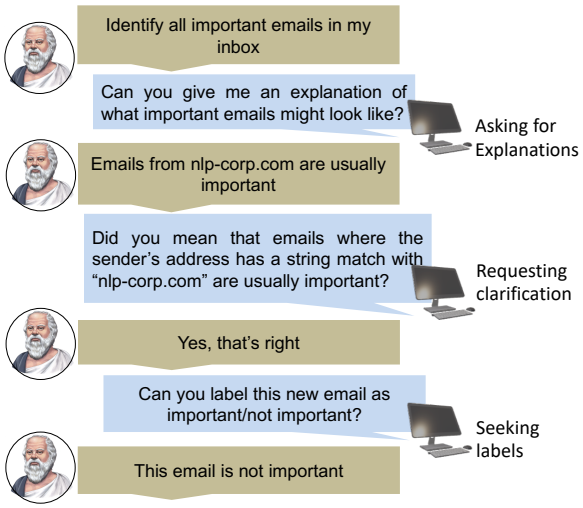


Figure 1: Question-Answer dialog can enable learning from a mix of strategies, including label observations (traditional supervised learning), explanations and clarifications (to overcome parsing limitations). The output from the teacher-learner interaction is a classification model (here, for important emails). We present a framework that (a) enables learning classifiers from a mix of such supervision; (b) learns to ask appropriate sequences of questions to accelerate this.

## 1 Introduction

The ability to learn new tasks and behaviors from language is characteristic of human intelligence. In recent years, the fields of machine learning and NLP have seen an renewed interest in incorporating natural language supervision in models of machine intelligence (Narasimhan et al., 2015; Elhoseiny et al., 2013; Goldwasser and Roth, 2014; Andreas et al., 2018; Fried et al., 2018; Wang et al., 2016). In particular, methods such as BabbleLabble (Hancock et al., 2018) and LNL (Srivastava et al., 2017) show progress towards realistic applications of supervised learning from language on tasks such as information extraction and email categorization. However, until now, such methods have been limited in two ways.

First, despite a body of work on leveraging language for tasks involving human robot interaction (She and Chai, 2017; Cakmak and Thomaz, 2012; Krishnamurthy and Kollar, 2013) and interactive learning in non-linguistic settings (see Section 2), existing approaches for training machine learning models from language are largely non-interactive, i.e. the learner agent receives statically collected text-based advice from a teacher as input, but does not directly engage with the

* Work done while the first and second authors were at CMU



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4164–4174, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

teacher.[1] In comparison, when humans learn, they do not rely only on passively receiving instruction from a teacher. Rather, the interaction takes the form of a mixed-initiative dialog, where they ask questions and proactively seek clarifications to simplify learning. These questions can generalize learning to novel situations, explore hypotheses, or fill information gaps. The ability to ask questions can, thus, fundamentally facilitate learning.

Second, existing approaches have focused on using language either as a standalone replacement for labeled data (Hancock et al., 2018), or to drive learning such as through specifying features for learning tasks (Eisenstein et al., 2009). In contrast, many realistic scenarios of learning from language would involve not learning from language alone, but learning from a mix of supervision, including both traditional labeled data, and natural language advice. Thus, automated learners should be capable of learning from a blend of observations, explanations and clarification.

In this work, we introduce a framework for learning from language in a conversational setting (*LiD*, for *Learning with Interactive Dialog*), which is a step towards alleviating these shortcomings. Language provides a natural medium for conversational interactions between a learner and a teacher, specifically in the form of question-answer dialog. The premise driving our work is that the ability to ask questions can be leveraged by an automated learner to accelerate its learning. We explore a data-driven approach for learning effective question-asking strategies in the specific context of learning classification tasks. The signal for learning to ask questions is grounded in the learning task itself. i.e., the value of a question is evaluated in utilitarian terms of how it affects performance on a downstream classification task. This follows a Wittgensteinian view of language as a cooperative game (Wittgenstein, 1953) between agents (here, the teacher and a learner) with a shared goal (here, building an effective classifier). While the space of questions that an interactive learner can ask can be vast in general, here we specifically focus on leveraging interactivity for three specific aspects (highlighted in Figure 1):

1. Seeking labels for specific examples.
2. Asking for explanations of a concept.
3. Requesting clarifications about explanations.

As illustrated in Figure 1, these dimensions can facilitate multiple aspects of the learning process: including learning from labeled examples (similar to traditional supervised learning), learning from natural language explanations (similar to recent work on learning from explanations) and alleviating limitations in the learner's semantic parsing abilities (in vein with work such as Labutov et al. (2018)). Learning systems that reify these abilities can enable users to interactively teach new concepts using a blend of traditional and language-based supervision. Our contributions are:

- A reinforcement learning formulation to guide question-asking strategies for learning from language.
- A method for interactively training classifiers using a mix of labeled data, natural language explanations and clarifications. Our exploration highlights some of the challenges involved in interactive learning from language.

## 2 Challenges in Relation to Previous Work

From the perspective of traditional supervised learning, the problem of asking questions can be seen as cognate with active learning. Methods in active learning have explored various criteria for choosing which of a set of unlabeled examples to label next while training supervised machine learning models (Settles, 2012; Collins et al., 2008). This can be seen as asking a specific kind of question (as illustrated in Figure 1). Learning to ask questions generalizes active learning in multiple ways by possibly soliciting a wider range of data measurements. These include feature labels ('Are emails with subject "urgent" usually important?'), label proportions ('Around what fraction of emails are important?'), constraints on model expectations ('Are you more likely to reply to important emails?'), etc. Approaches such as Srivastava et al. (2018) map such language to data measurements that computational models can reason over.[2] Statistical frameworks such as Generalized Expectation (Druck et al., 2008), Posterior Regularization (Ganchev et al., 2010) and Bayesian

[1] Zhang et al. (2018) diverge from prior work in this respect, and model language games between teachers and learners. However, their learning tasks are toylike, and the method does not generalize to realistic scenarios.

[2] For example, a statement such as 'Emails from my boss are usually important' may be mapped to a data measurement of form $P(y = important|sender = boss) \approx P_{usually}$.

Measurements (Liang et al., 2009) then allow for model training from a broad range of such data measurements in conjunction with unlabeled data, rather than using labeled examples. Other recent approaches such as Huang et al. (2015) and Siddiquie and Gupta (2010) have expanded pool-based active learning to learning from multiple types of queries, especially in the context of multi-label and multi-class learning. Similarly, Parikh and Grauman (2011) explore feature space construction for visual tasks in an interactive setting. Although in principle soliciting different types of data measurements can help learning, each type requires its own interface. The advantage of using natural language as a medium is that it allows us to unify the different modes of supervision into a single, familiar user interface. However, using natural language as a medium of supervision comes with its own set of challenges, as we discuss next.

### 2.1 Dependence on Language Interpreter

Since both generation and transmission of language advice can be noisy, the optimal question asking strategy may depend not only on the information content of data measurements, but also factors such as the quality of the learner's semantic parsing model and the teacher's skill.[3] To explain, while useful from an information theoretic sense, a teacher's explanations may be too complex to handle for the learning agent's parser, in which case it might be preferable to stick to asking about the teacher about instance labels (which would require minimal parsing). Thus, question-asking strategies need to be sensitive to the learner's own semantic parsing ability, which may also change during the course of interactions with users.

### 2.2 Context Dependence

Rather than learning a static criterion for choosing what question to ask (as in active learning), our focus is on asking questions in conversational settings, which are inherently dynamic processes. To explain, asking a teacher to rephrase an explanation only makes sense in specific contexts (when the interpretation of something said previously is unclear). Further, the question to ask can depend on factors such as the task domain, supervision previously received, etc. These factors motivated our choice of a reinforcement learning approach for learning question-asking strategies.

**Relation to Question Generation approaches:** The problem of learning to ask questions has previously been explored by several approaches. Vanderwende (2008) and Olney et al. (2012) explore generating reading comprehension questions conditioned on a given text. More recently, Romeo et al. (2016) and Rao and Daumé III (2018) present neural network models that rank questions in community QA forums, whereas Misra et al. (2018) generate questions for visual scene understanding. Our framework significantly differs from these in its sequential framework, and that the questions to be asked are grounded in quantitative performance on a downstream task.

## 3 Approach

In this section, we describe our framework for interactive learning from question-answer dialog. We first describe an approach to learn classifiers using a mix of explanations and labeled examples in Section 3.1. This is a preliminary towards question-asking strategies that subsume active learning as well as language advice; and constitutes a subroutine that is repeatedly invoked in our approach. Section 3.2 describes our reinforcement learning formulation for learning question-asking strategies in simulated conversational settings. The space of actions consists of a vocabulary of question types that a learner can ask, and reward is based on improvements in the classification model that the teacher's response to an asked question leads to.

### 3.1 Learning classifiers from a mix of observations and explanations

We base our learning framework on previous work by Srivastava et al. (2018), who train loglinear classifiers (with parameters $\theta$) using natural language explanations of the individual classes and unlabeled data. Further, they use the semantics of linguistic quantifiers (such as 'usually','always', etc.) as priors in a Posterior Regularization objective to drive the model training. In particular, their training objective takes the following form:

$$J_Q(\theta) = \underbrace{\mathcal{L}(\theta)}_{\text{Explain data}} \underbrace{- \min_{q \in Q} KL(q \mid p_\theta(Y|X))}_{\text{Emulate human advice}} \quad (1)$$

[3] In this work, we are not interested in learning semantic parsing models. We presume the existence of pretrained semantic parsers for learning agents. Our focus is rather on whether some question asking strategies may be more effective than others for a learning agent with given capabilities.

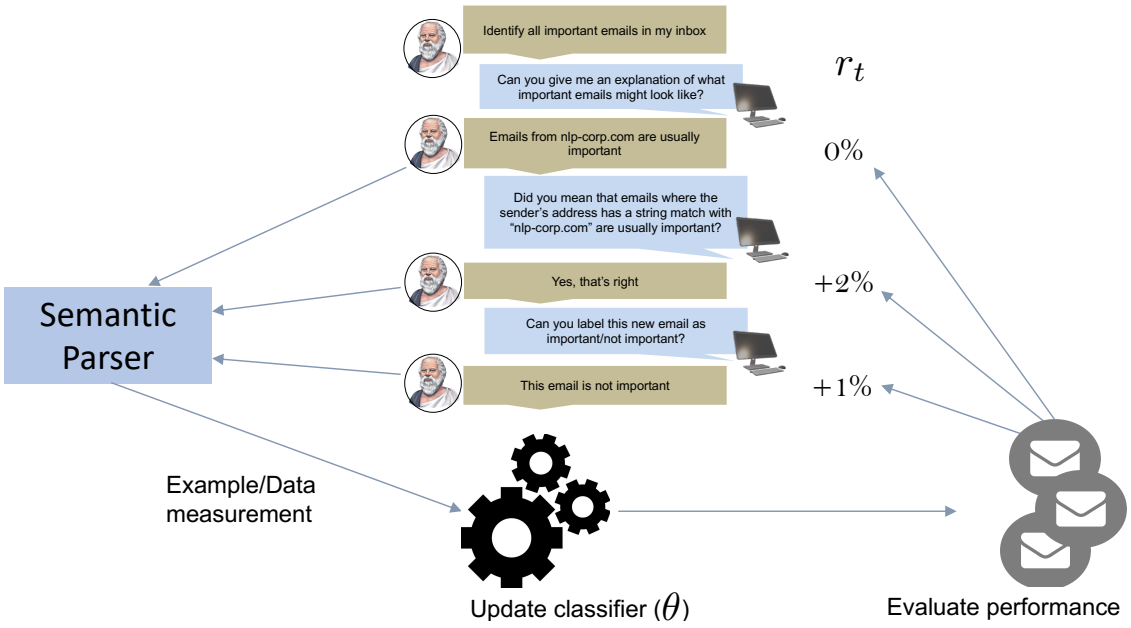


Figure 2: We assume that the dialog between the learner and the teacher is in the form of turn-wise conversations – consisting of a sequence of questions asked by the learner, and the teacher's responses to those questions. At each step in this process, the teacher's response is parsed by the learner (using a pre-trained semantic parser), and can be incorporated into the learner's concept model as either a labeled example or a data measurement (the learner can also choose to seek a clarification). A reward (denoted by $r_t$) can be computed at each step, which denotes the marginal change in classification performance on a held-out set of examples due to the last response. In this framework, learning good question-asking strategies corresponds to asking sequences of questions that maximize the cumulative (discounted) reward, and hence quickly lead to effective concept models. The framework also allows for asking sequences of multiple questions before seeing a major jump in model performance.

The objective reflects a tension between explaining the unlabeled data (likelihood term) and emulating the natural language advice provided by a teacher. The KL divergence represents difference between predictions from the trained model on unlabeled data $p_\theta(Y|X)$ and language advice (each explanation is incorporated as a data measurement; the conjunction of these defines the 'valid set' of posterior distributions $Q$ that perfectly concur with the natural language advice). The second term essentially computes the minimum distance between the model posterior and the set $Q$.

Here, we show that we can naturally extend this approach to learn classifiers from a mix of both labeled and unlabeled data, and natural language explanations. To do this, we simply append a log-likelihood term for the labeled examples to the objective in Equation 1. The updated objective is:

$$J_Q(\theta) = \mathcal{L}_{labeled}(\theta) + \mu\Big(\mathcal{L}_{unlabeled}(\theta) - \min_{q \in Q} KL(q \mid p_\theta(Y|X))\Big) \quad (2)$$

Here, $\mathcal{L}_{labeled}(\theta)$ denotes the log-likelihood term for a set of $n_{labeled}$ labeled examples $\mathcal{X}_{labeled} = \{(x_k, y_k)\}_{labeled}^{n}$ (normalized by $n_{labeled}$), whereas the other two terms are as before: $\mathcal{L}_{unlabeled}(\theta)$ denoting log-likelihood over a set of $n_{unlabeled}$ unlabeled examples, and a posterior regularizer term (KL-divergence) penalizing violations of the parse natural language advice. In the E-step of the Posterior Regularization training (Ganchev et al., 2010), the computation of the posterior regularizer remains unchanged. However, the M-step is modified so that the classifier parameters $\theta$ are learned using both the inferred labels for the unlabeled data, and provided labels for the labeled examples.

In Equation 2, $\mu > 0$ determines the relative weights of provided example labels and natural language advice in the optimization objective and is a hyper-parameter for the method. In learning scenarios where there is little labeled data, we would like to rely primarily on constraints specified from natural language explanations, and unlabeled data. On the other hand, in scenarios where there is a lot of labeled data available enabling robust inductive inference, we would like

to primarily rely on it rather than explanations.[4] While setting up the optimization problem, the value of $\mu$ can be adapted to reflect this intuition. In our experiments, we found setting $\mu = 1/\max(n_{labeled}, 1)$ to work well across settings.

### 3.2 RL formulation for learning to ask

Figure 2 illustrates our framework for learning classification tasks in a question-answer dialog setting. We assume the presence of a teacher to answer questions posed by the learner. We restrict the structure of dialog to a sequence of questions $(q_1 \dots q_T)$ asked by the learner, and the teacher's responses $(u_1 \dots u_T)$ to them. We further assume the presence of a held-out set of labeled examples of the concept, which can be used to evaluate the learner's classification performance as the dialog progresses. At each step $t$, the learner's action $a_t$ consists of choosing a question to ask the teacher. The teacher's response to the learner's question is parsed (in the form of a labeled example, or a data measurement), which is then incorporated into the learner's concept model (by retraining with the additional labeled example or the new data measurement). The classification performance, $c_t$, of the updated model is evaluated on the held-out set. The change in classification performance from the previous step, $r_t$ approximates the marginal value of the question in learning the task, and constitutes the learner's reward at that step.

Our approach for learning question asking strategies models the dialog as a simple Markov Decision Process. Since our state and action spaces are discrete (as described in the following sections), we can use a table-based SARSA-learning procedure (which allows for on-policy learning over Q-learning) to estimate the state-action values $Q(s, a)$ of different question types in different contexts. We next describe the state-space, actions and rewards, and the learning procedure.

#### 3.2.1 Action Space

As mentioned earlier, there can be a multitude of questions that a learner can ask a teacher. Here, we are interested in exploring three specific types of questions that are specially germane for facilitating learning from a mix of labeled examples and explanations. These consist of the following:

1. **Seeking labels for specific examples:** This is similar to traditional active learning. In particular, we can have a different action corresponding to every active learning criterion, which chooses which example to label next. In our experiments, we use two active learning techniques (with a corresponding action for each):
   - *Random:* Ask for class label for a randomly chosen unlabeled instance.
   - *Maximum Uncertainty:* Ask for class label for the instance in the data for which the current concept model is most uncertain (highest entropy). If there are multiple such instances, randomly pick one among them.
2. **Asking for an explanation for the concept:** This action seeks out from the teacher a short natural language explanation of the concept. This is then incorporated in the concept model as a quantitative constraint. In general, this can encompass several types of questions:
   - Asking for probability estimates about specific labels and features. e.g., 'How often are emails about meetings important?'
   - Asking for discriminative features for particular concept labels. e.g., 'Can you think of a feature that if present always denotes that an email is important?'
   - Asking about class probabilities. e.g., 'Around what fraction of emails in your inbox are important?'

   In principle, each of the above provide admissible constraints which the classifier training procedure (from Section 3.1) can handle. However, to simplify analysis, in our experiments, we conflate these actions into one category, and ask for general explanations of the form '*Can you give me an explanation of the concept?*', which could return a variety of data measurements (subsuming $P(y)$,$P(y|x)$ and $P(x|y)$, previously explored in Srivastava et al. (2018)).
3. **Requesting clarification about the previous explanation:** This action asks for a clarification about the interpretation of a previous explanation (which can be helpful in cases when the learner is uncertain about the interpretation). For this, the learner verifies if the interpretation of the previous explanation (using

[4] In the asymptotic case of infinite data with labels, an inductively learned Bayes Classifier would be optimal.

the learner's semantic parser) was correct or not. To do this, we generate a question of the form '*Did you mean ...*' using a synchronous grammar which deterministically maps logical forms to natural language descriptions (see Figure 1 for an example). The teacher responds with yes, if the parsed logical form matches the gold annotated logical form, and with no otherwise. In case the teacher responds with no, the current explanation is discarded (not used in model training), and the learner moves ahead to ask for a new label or explanation.

**Simulating Interactions:** We note that each of the question types described above – (1) asking for labels for examples, (2) asking for concept explanations, and (3) verifying interpretations of language explanations – can be simulated with corresponding statically collected data – consisting of (1) labeled examples for classification, (2) natural language explanations of classes, and (3) annotations of those explanations with logical forms. This has a significant implication: rather than relying on questioning human users in real-time, we can simulate the conversational exchange by asking questions to an oracle, which has access to previously pre-collected data of the above-described form for each classification task. While this is a coarse approximation of actual dialog between an automated learner and human teachers, it can serve as a useful proof-of-concept, and allows for quick experimentation. We rely on this simulated setting for learning policies for question asking.

#### 3.2.2 Rewards

The reward, $r_t$, evaluates the change in classification performance due to an asked question at each step $t$ of the dialog. The performance of the classification model is evaluated on a held out set of $n_{heldout} = 50$ labeled examples for each learning task. In our experiments, we use the model's F1 score as the metric for classification performance, $c_t$. We define the reward as the absolute change in model performance from the previous step:

$$r_t = c_t - c_{t-1} \tag{3}$$

#### 3.2.3 State-space

Next, we describe the featurized state space for our reinforcement learning formulation. The best question to ask at a particular point can likely depend on the state of the conversation. This could include factors such as the pedagogical phase in the learning process (exploratory vs confirmatory), previous questions asked, etc. Thus, defining a rich enough state space is an important consideration for a formulation of conversational learning. In our treatment, we assume a discrete state-space, which is defined as the cross product of the following (also discrete) features.

- **Curricular stage in the Learning process:** We use a discrete variable to model the curricular stage of the learner, approximating it by the number of steps (questions previously asked) in the interaction at any point. We cluster the number of steps in the following five bins of values: BEGINNER (0 steps), NOVICE (1-5 steps), INTERMEDIATE (6-10 steps), ADVANCED (11-15 steps) and MATURE ($>$ 15 steps).[5]
- **Reward in the previous state:** We discretize the value of reward as belonging to one of four ranges (abstractly named GOOD, INCREASING, FLAT, and DECREASING), with thresholds chosen to correspond to the inter-quartile ranges for the value of rewards observed in evaluating a random policy.
- **Velocity of reward:** This is a ternary variable indicating whether the value of the discrete variable for the reward (above) is BETTER,WORSE or the SAME than the previous step.
- **Type of the previous two actions:** As mentioned in the description of the action space.
- **Domain of learning task:** Indicates the domain of the current classification task. We use datasets corresponding to three domains: EMAIL CATEGORIZATION, SHAPE CLASSIFICATION and BIRD SPECIES IDENTIFICATION; hence this variable can take these three values.
- **Confidence of previous parse:** We model the learner's confidence in parsing the response $u_t$ from a teacher as the ratio of the probability of the highest probability (predicted) logical form from the learner's semantic parser and the next best logical form. We discretize this ratio into three values, corresponding to the upper (HIGH), lower (LOW) and middle two inter-quartile ranges (MEDIUM) for the value of the ratio over all explanations in our data.

[5]These threshold values were heuristically chosen based on the observation that for many datasets that we experiment with, classifier performance roughly begins to taper at around 20 training examples.

While our state space captures several facets, it does not model some other important factors:

- **Teacher behavior:** Whether the teacher provides correct information, and uses easily interpretable language.
- **Task difficulty:** This refers to how expressible a classification task is using language explanations. For example, some concept maybe significantly easier to explain using language than others, depending on the logical language available to the semantic parser. For example, it maybe impossible to explain digit recognition using pixel level features using language.

### 3.3 Model training

Since the action and state-spaces are discrete and not prohibitively large, we use the on-policy SARSA learning algorithm for policy control (Rummery and Niranjan, 1994), where we represent the state-action Q-values for pairs of states $s$ and actions $a$ as a table. We use an $\epsilon$-greedy strategy ($\epsilon = 0.20$). i.e., the strategy balances between exploitation and exploration by picking the next action to be the estimated optimal one (having the maximum estimated Q-value for a state) with probability $1 - \epsilon$, and choosing the next action randomly with a probability of $\epsilon$. The initial policy is defined by uniform randomly initializing $Q(s, a)$ values between 0 and 1.

## 4 Data

Our empirical analysis uses existing datasets for learning classifiers from language. These are datasets for email categorization from natural language explanations from Srivastava et al. (2017); and bird species classification and synthetic shape classification tasks from Srivastava et al. (2018). In all, thesee consist of 67 classification tasks belonging to these three domains.[6] For each task, the corresponding data consisting of natural language explanations of classes as well as annotated logical forms for these explanations are available. For each task, we hold out a random sample of 50 examples for evaluating classifier performance. The rest of the examples (ranging between 50 and 100 for individual tasks) are considered as unlabeled data at the start of each interaction. At each step in a simulated interaction, either (1) the label for a new example from this set is revealed by the teacher, (2) a new data measurement is provided as a natural language explanation by the teacher, or (3) the learner seeks a clarification about the parse of the previous explanation. Thus, from the learner's perspective, over time the size of the unlabeled data decreases, and the labeled examples increase (see Equation 2).

[6]The complexity of these problems varies considerably, ranging from marginally above chance to perfectly learnable.

## 5 Experiments

In this section, we evaluate *LiD*'s performance for three domains of classification tasks. As previously mentioned, for policy learning we simulate conversational interactions between teachers and learners by asking questions to a oracle, which has access to previously collected data about each classification task. One limitation of learning question-asking strategies from simulated interactions is that for some classification tasks, we may run out of explanations in the course of model training (since the number of explanations of a concept are limited). In these cases, we end the interaction as soon as all explanations are already provided to the learner.

### 5.1 Learned vs Random policy evaluation

First, we compare learned policies for question asking with a naive policy that randomly takes a new action at each step in the learning process.

Figure 3 shows averaged cumulative reward for question asking strategies on 20 unseen classification tasks, after SARSA learning for 10 epochs on the remaining 37 classification tasks. In the figure, the x-axis corresponds to the number of steps in a dialog, and the y-axis denotes the cumulative reward (averaged over 20 tasks) for a learned policy vs a random policy. We note that policy learning leads to consistently superior performance (on average, *LiD* achieves any given level of classification performance in fewer steps), which unambiguously indicates value in asking the right sequences of questions. We observed that the trend was also similar in most individual learning tasks. For example, the cumulative reward after 10 steps was higher for the learned policy than the random policy for 18 of the 20 learning tasks. The difference in performance was statistically significant at $p < 0.05$ using a signed permutation test. We characterize some learned behaviors that drive this improved performance in Section 5.4.

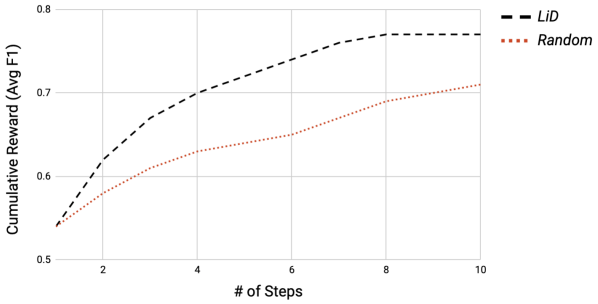


Figure 3: Cumulative Reward (averaged over 20 tasks) for interactive concept learning for learned question asking policy vs random policy.

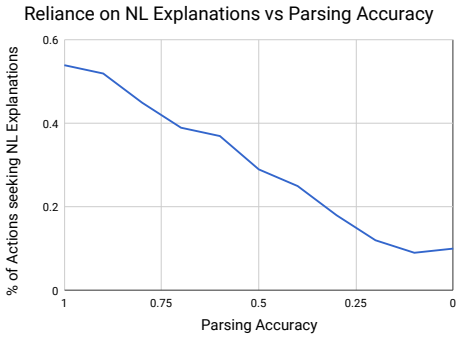


Figure 4: Fraction of actions seeking Natural Language Explanations vs competence of the learner's semantic parser.

## 5.2 Reliance on NL vs Parsing accuracy

Intuitively, semantic parsing competence of a learner should be a significant consideration in whether it should rely on explanations. To test this, we simulate scenarios of learners with different levels of semantic parsing ability by choosing the true logical form for any explanation with the corresponding probability, and choosing an alternative logical form from the remaining candidates in the $k$-best list from the semantic parser otherwise. Figure 4 depicts the effect of parsing competence on learned question-asking strategies. The empirical behavior largely corroborates our expectation, as the learned strategies increasingly avoid seeking natural language explanations of concepts as the parsing performance worsens. In the base case where the learner has no parsing competence, the model learns to exclusively ask for labeled examples only (In the figure, the fraction is seen to converge close to 0.1 rather than 0 due to the $\epsilon$-greedy nature of the policy).

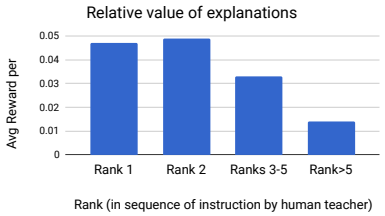


Figure 5: Average increase in classification performance from explanations of different positional orders.

## 5.3 Differential value of explanations

A notable issue in learning from explanations, which we do not model here is that a teacher's multiple explanations of a concept can have a large variance in their utility to a learner. In particular, we might expect that teachers would be more likely to provide the most useful explanations first, and minor explanations subsequently. From an ablation study, we observe that this is indeed a valid concern. Figure 5 shows the average marginal increase in classification performance over 50 visual shape classification tasks from explanations with different rank (based on the actual order of providing from human users). This indicates that explanations from teachers provided later contribute significantly less towards classification performance.

## 5.4 Examples of Learned behavior

From a qualitative analysis, learned policies are seen to be intuitive and interpretable. In particular, the policies overwhelmingly seek clarifications when confidence in the parser is low. On the other hand, there are strong inclinations to continue using an action type as long as it yields high returns. Interestingly, the optimal policy differs significantly in behavior across domains. As example, learned policies rely nearly twice as much on explanations for bird species identification as for email classification tasks. The probable reason is that parsing is harder for email explanations, as features in this domain are often compositional.

## 5.5 User study

We perform a small user-study to also evaluate performance of the learned policies on an email categorization task with actual human teachers. We still train the question-asking strategy using the simulated teacher framework (since learning the policy from crowdsourced human users would be expensive and slow). 20 users were asked to in-

| | Avg. Reward | Natural | Avg. Rew (simulated) |
|---|---|---|---|
| *LiD* | 0.524 | 3.2 | 0.607 |
| *Random* | 0.493 | 2.9 | 0.551 |

Table 1: Human teacher evaluation for learned and random question asking policy.

teract with the learned *LiD* policy to teach a chosen email-classification task. For each task, the system asked a sequence of 10 questions, and the human teacher's responses were incorporated into the system to update the classification model. The users were also asked to teach another task with questions asked through a random policy. Table 1 shows the average cumulative reward for humans interacting with *LiD* vs a random policy for this experiment. We note that *LiD* leads to better performance on average. This trend is the same as in the simulated analysis, although we note that the learning is slower with real teachers than in the simulated setting on the same tasks, and the gain in performance is substantially smaller. A contributing reason for this is likely annotator bias (Geva et al., 2019), since in the simulated testing scenarios, the teacher's explanations can often likely come from a small set of turkers whose language explanations for teaching other tasks were used for training the learner's semantic parsing model. We note that the learned policy was rated by human users as more natural than a random policy on a Likert scale (with range 1-5).

## 6 Conclusion

In this paper, we have provided a reinforcement learning formulation for learning to ask questions for interactive training of machine learning models. This framework is attractive in grounding the value of questions asked in a measurable downstream task. Further, change in model performance is a natural reward to drive this learning. While this provides a conceptually useful framework for framing question generation, in its current form the approach makes simplistic assumptions on the types of questions that can be asked, as well as on the structure of the dialog between the teacher and the learner. While the system outperforms a random policy on learning classification tasks, the dialog looks contrived from a human perspective. An interesting direction could be to pair the framework with neural text generation methods to model fine-grained question types, and generate more natural-looking interactions through dialog. An important scientific question is to characterize learning tasks for which learning from language is likely to outperform pure inductive learning. Future work can also extend the approach to other supervised learning tasks, as well as bootstrap from natural dialog data.

## References


Jacob Andreas, Dan Klein, and Sergey Levine. 2018. Learning with latent language. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2166–2179. Association for Computational Linguistics.

Maya Cakmak and Andrea L. Thomaz. 2012. Designing robot learners that ask good questions. In *Proceedings of the Seventh Annual ACM/IEEE International Conference on Human-Robot Interaction*, HRI '12, pages 17–24, New York, NY, USA. ACM.

Brendan Collins, Jia Deng, Kai Li, and Li Fei-Fei. 2008. Towards scalable dataset construction: An active learning approach. In *European conference on computer vision*, pages 86–98. Springer.

Gregory Druck, Gideon Mann, and Andrew McCallum. 2008. Learning from labeled features using generalized expectation criteria. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 595–602. ACM.

Jacob Eisenstein, James Clarke, Dan Goldwasser, and Dan Roth. 2009. Reading to learn: Constructing features from semantic abstracts. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 958–967. Association for Computational Linguistics.

Mohamed Elhoseiny, Babak Saleh, and Ahmed Elgammal. 2013. Write a classifier: Zero-shot learning using purely textual descriptions. In *The IEEE International Conference on Computer Vision (ICCV)*.

Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. 2018. Speaker-follower models for vision-and-language navigation. *CoRR*, abs/1806.02724.

Kuzman Ganchev, João Graça, Jennifer Gillenwater, and Ben Taskar. 2010. Posterior regularization for structured latent variable models. *The Journal of Machine Learning Research*, 11:2001–2049.

Mor Geva, Yoav Goldberg, and Jonathan Berant. 2019. Are we modeling the task or the annotator? an investigation of annotator bias in natural language understanding datasets. *arXiv preprint arXiv:1908.07898*.

Dan Goldwasser and Dan Roth. 2014. Learning from natural instructions. *Mach. Learn.*, 94(2):205–232.

Braden Hancock, Paroma Varma, Stephanie Wang, Martin Bringmann, Percy Liang, and Christopher Ré. 2018. Training classifiers with natural language explanations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1884–1895. Association for Computational Linguistics.

Sheng-Jun Huang, Songcan Chen, and Zhi-Hua Zhou. 2015. Multi-label active learning: Query type matters. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 946–952. AAAI Press.

Jayant Krishnamurthy and Thomas Kollar. 2013. Jointly learning to parse and perceive: Connecting natural language to the physical world. *Transactions of Association for Computational Linguistics*.

Igor Labutov, Bishan Yang, and Tom Mitchell. 2018. Learning to learn semantic parsers from natural language supervision. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1676–1690.

Percy Liang, Michael I Jordan, and Dan Klein. 2009. Learning from measurements in exponential families. In *Proceedings of the 26th annual international conference on machine learning*, pages 641–648. ACM.

Ishan Misra, Ross Girshick, Rob Fergus, Martial Hebert, Abhinav Gupta, and Laurens van der Maaten. 2018. Learning by asking questions. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11–20. IEEE.

Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. 2015. Language understanding for text-based games using deep reinforcement learning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1–11.

Andrew M Olney, Arthur C Graesser, and Natalie K Person. 2012. Question generation from concept maps. *Dialogue & Discourse*, 3(2):75–99.

Devi Parikh and Kristen Grauman. 2011. Interactively building a discriminative vocabulary of nameable attributes. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1681–1688. IEEE Computer Society.

Sudha Rao and Hal Daumé III. 2018. Learning to ask good questions: Ranking clarification questions using neural expected value of perfect information. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2737–2746.

Salvatore Romeo, Giovanni Da San Martino, Alberto Barrón-Cedeno, Alessandro Moschitti, Yonatan Belinkov, Wei-Ning Hsu, Yu Zhang, Mitra Mohtarami, and James Glass. 2016. Neural attention for learning to rank questions in community question answering. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1734–1745.

G. A. Rummery and M. Niranjan. 1994. On-line Q-learning using connectionist systems. Technical Report TR 166, Cambridge University Engineering Department, Cambridge, England.

Burr Settles. 2012. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114.

Lanbo She and Joyce Chai. 2017. Interactive learning of grounded verb semantics towards human-robot communication. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1634–1644.

Behjat Siddiquie and Abhinav Gupta. 2010. Beyond active noun tagging: Modeling contextual interactions for multi-class active learning. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2979–2986. IEEE.

Shashank Srivastava, Igor Labutov, and Tom Mitchell. 2017. Joint concept learning and semantic parsing from natural language explanations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1527–1536.

Shashank Srivastava, Igor Labutov, and Tom Mitchell. 2018. Zero-shot learning of classifiers from natural language quantification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 306–316. Association for Computational Linguistics.

Lucy Vanderwende. 2008. The importance of being important: Question generation. In *Proceedings of the 1st Workshop on the Question Generation Shared Task Evaluation Challenge, Arlington, VA*.

Sida I. Wang, Percy Liang, and Christopher D. Manning. 2016. Learning language games through interaction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Ludwig Wittgenstein. 1953. *Philosophical investigations. Philosophische Untersuchungen.* Macmillan.

Haichao Zhang, Haonan Yu, and Wei Xu. 2018. Interactive language acquisition with one-shot visual concept learning through a conversational game. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2609–2619. Association for Computational Linguistics.

# Language Modeling for Code-Switching: Evaluation, Integration of Monolingual Data, and Discriminative Training

**Hila Gonen**[1] and **Yoav Goldberg**[1,2]
[1]Department of Computer Science, Bar-Ilan University
[2]Allen Institute for Artificial Intelligence
{hilagnn,yoav.goldberg}@gmail.com

## Abstract

We focus on the problem of language modeling for code-switched language, in the context of automatic speech recognition (ASR). Language modeling for code-switched language is challenging for (at least) three reasons: (1) lack of available large-scale code-switched data for training; (2) lack of a replicable evaluation setup that is ASR directed yet isolates language modeling performance from the other intricacies of the ASR system; and (3) the reliance on generative modeling. We tackle these three issues: we propose an ASR-motivated evaluation setup which is decoupled from an ASR system and the choice of vocabulary, and provide an evaluation dataset for English-Spanish code-switching. This setup lends itself to a discriminative training approach, which we demonstrate to work better than generative language modeling. Finally, we explore a variety of training protocols and verify the effectiveness of training with large amounts of monolingual data followed by fine-tuning with small amounts of code-switched data, for both the generative and discriminative cases.

## 1 Introduction

This work deals with neural language modeling of code-switched language, motivated by an application to speech recognition. Code-switching (CS) is a linguistic phenomenon defined as "the alternation of two languages within a single discourse, sentence or constituent." (Poplack, 1980). Since CS is widely used in spoken platforms, dealing with code-switched language becomes an important challenge for automatic speech recognition (ASR) systems. To get a feeling how an ASR system trained on monolingual data performs on code-switched language, we fed the IBM English and Spanish systems[1] with audio files of code-switched conversations from the Bangor Miami Corpus (see Section 7). The results (examples available in Table 1) exhibit two failure modes: (1) Words and sentences in the opposite language are not recognized correctly; and (2) Code-switch points also hurt recognition of words from the main language. This demonstrates the need for designated speech recognition systems for CS. A crucial component in such a CS ASR system is a strong CS language model, which is used to rank the bilingual candidates produced by the ASR system for a given acoustic signal.

Language models are traditionally evaluated with perplexity. However, this measure suffers from several shortcomings, in particular strong dependence on vocabulary size and lack of ability to directly evaluate scores assigned to malformed sentences. We address these deficiencies by presenting a new evaluation scheme for LM that simulates an ASR setup. Rather than requiring the LM to produce a probability for a gold sentence, we instead present the LM with a set of alternative sentences, including the gold one, and require it to rank the gold one higher. This evaluation is more realistic since it simulates the role a language model plays in the process of converting audio into text (Section 3). We create such an evaluation dataset for English-Spanish CS – SpanglishEval (Section 4).

Additionally, LM for CS poses a unique challenge: while data for training monolingual LMs is easy to obtain in large quantities, CS occurs primarily in spoken language. This severely limits its data availability, thus, small amounts of training data are an intrinsic property of CS LM. A natural approach to overcome this problem is to train monolingual language models—for which

[1]https://www.ibm.com/watson/services/speech-to-text/



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4175–4185,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| Original sentence (audio) | English model output | Spanish model output |
|---|---|---|
| no pero vino porque he came to check on my machine | No but we no longer he came to check on my machine | Cual provino de que en dicha cámara chino |
| yo le invité pa Easter porque me daba pena el pobre aquí solo sin familia ni nada | feeling betrayed by eastern lap and I bought a new solo seem funny any now | y el envite país tampoco nada pero por aquí solo sin familia ni nada |

Table 1: Examples of the output of IBM's English and Spanish speech recognition systems on code-switched audio.

we have huge amounts of data—for each language separately, and combine them somehow into a code-switched LM. While this is relatively straightforward to do in an n-gram language model, it is not obvious how to perform such an LM combination in a non-markovian, RNN-based language model. We use a protocol for LSTM-based CS LM training which can take advantage of monolingual data (Baheti et al., 2017) and verify its effectiveness (Section 5).

Based on the new evaluation scheme we present, we further propose to learn a model for this ranking task using discriminative training. This model, as opposed to LMs, no longer depends on estimating next-word probabilities for the entire vocabulary. Instead, during training the model is introduced with positive and negative examples and is encouraged to prefer the positive examples over the negative ones. This model gives significantly better results (Section 6).

Our contributions in this work are four-fold: (a) We propose a new, vocabulary-size independent evaluation scheme for LM in general, motivated by ASR. This evaluation scheme is ranking-based and also suits CS LM; (b) We describe a process for automatically creating such datasets, and provide a concrete evaluation dataset for English-Spanish CS (SpanglishEval); (c) We present a model for this new ranking task that uses discriminative training and is decoupled of probability estimations, this model surpasses the standard LMs; (d) We verify the effectiveness of pretraining CS LM for this ranking task with monolingual data and show significant improvement over various baselines. The CS LM evaluation dataset and the code for the model are available at `https://github.com/gonenhila/codeswitching-lm`.

## 2 Background

**Code-Switching** Code-switching (CS) is defined as the use of two languages at the same discourse (Poplack, 1980). The mixing of different languages in various levels has been widely studied from social and linguistic point of view (Auer, 1999; Muysken, 2000; Bullock and Toribio, 2009), and started getting attention also in the NLP community in the past few years (Solorio and Liu, 2008; Adel et al., 2013a; Cotterell et al., 2014).

Below is an example of code-switching between Spanish and English (taken from the Bangor Miami corpus described in Section 7). Translation to English follows:

- "that *es su tío* that has lived with him like I don't know how like *ya* several years..."
  that his uncle who has lived with him like, I don't know how, like several years already...

Code-switching is becoming increasingly popular, mainly among bilingual communities. Yet, one of the main challenges when dealing with CS is the limited data and its unique nature: it is usually found in non standard platforms such as spoken data and social media and accessing it is not trivial (Çetinoğlu et al., 2016).

**Shortcomings of Perplexity-based Evaluation for LM** The most common evaluation measure of language models is perplexity. Given a language model $M$ and a test sequence of words $w_1, ..., w_N$, the perplexity of $M$ over the sequence is defined as:

$$2^{-\frac{1}{N}\sum_{i=1}^{N}\log_2 M(w_i)}$$

where $M(w_i)$ is the probability the model assigns to $w_i$.

A better model is expected to give higher probability to sentences in the test set, that is, lower perplexity. However, this measure is not always well aligned with the quality of a language model as it should be. For example, Tran et al. (2018) show that RNNs capture long-distance syntactic dependencies better than attention-only LMs, despite having higher (worse) perplexity. Similarly, better perplexities often do not translate to better word-error-rate (WER) scores in an ASR system (Huang et al., 2018).

This highlights a shortcoming of perplexity-based evaluation: the method is rewarded for assigning high probability to gold sentences, but is not directly penalized for assigning high probability to highly implausible sentences. When used in a speech recognition setup, the LM is expected to do just that: score correct sentences above incorrect hypotheses.

Another shortcoming of perplexity-based evaluation is that it requires the compared models to have the same support (in other words, the same vocabulary). Simply adding words to the vocabulary, even if no additional change is done to the model, will likely result in higher perplexity for the same dataset. It is also sketchy to compare perplexities of word-based LMs and character-based LMs for the same reason.

**Problem with WER-based evaluation** Evaluating LM models based on the final WER of an ASR system side-steps these two issues: it evaluates the LM on incorrect sentences, and seamlessly compares LMs with different support. However, this makes the evaluation procedure highly dependent on a particular ASR system. This is highly undesirable, as ASR systems are hard to set up and tune and are not standardized. This both conflates the LM performance with performance of other aspects of the ASR system, and makes it hard to replicate the evaluation procedure and fairly compare results across different publications. Indeed, as discussed in Section 10, most previous works on CS LM use an ASR system as part of their evaluation setup, and none of them compares to any other work. Moreover, no standard benchmark or evaluation setup exists for CS LM.

## 3 An Alternative Evaluation Method

We propose an evaluation technique for language models which is ASR-system-independent, that does take into account incorrect sentences and allows to compare LMs with different support or OOV handling techniques.

We seek a method that meets the following requirements: (1) Prefers language models that prioritize correct sentences over incorrect ones; (2) Does not depend on the support (vocabulary) of the language model; (3) Is independent of and not coupled with a speech recognition system.

To meet these criteria, we propose to assemble a dataset comprising of gold sentences, where each gold sentence is associated with a set of alternative sentences, and require the LM to identify the gold sentence in each set. The alternative sentences in a set should be related to the gold sentence. Following the ASR motivation, we take the alternative set to contain sentences that are phonetically related (similar-sounding) to the gold sentence. This setup simulates the task an LM is intended to perform as a part of an ASR system: given several candidates, all originating from the same acoustic signal, the LM should choose the correct sentence over the others.

**A New Evaluation metric** Given this setup, we propose to use the *accuracy* metric: the percentage of sets in which the LM (or other method) successfully identified the gold sentence among the alternatives.[2] The natural way of using an LM for identifying the gold sentences is assigning a probability to each sentence in the set, and choosing the one with highest probability. Yet, the scoring mechanism is independent of perplexity, and addresses the two deficiencies of perplexity based evaluation discussed above.

Our proposed evaluation method is similar in concept to the NMT evaluation proposed by Sennrich (2016). There, the core idea is to measure whether a reference translation is more probable under an NMT model than a contrastive translation which introduces a specific type of error.

## 4 Evaluation Dataset

We now turn to construct such an evaluation dataset. One method of obtaining sentence-sets is feeding acoustic waves into an ASR system and tracking the resulting lattices. However, this requires access to the audio signals, as well as a trained system for the relevant languages. We propose instead a generation method that does not require access to an ASR system.

The dataset we create is designed for evaluating English-Spanish CS LMs, but the creation process can be applied to other languages and language pairs.[3]

[2] We chose accuracy over WER as the default metric since in our case, WER should be treated with caution: the alternatives created might be "too close" to the gold sentence (e.g. when only a small fraction of the gold sentence is sampled and replaced) or "too far" (e.g. a Spanish alternative for an English sentence), thus affecting the WER.

[3] The requirements for creating an evaluation dataset for a language pair L1,L2 is to have access to code-switched sentences where each word is tagged with a language ID (lan-

Example 1:
Gold: *pero* i thought *ella fue* like very good
Alt-CS: *pero* i thought a *llevó* alike very good
Alt-EN: peru i thought a of alike very could
Alt-SP: *pero azote la fue la que rico*
Example 2:
Gold: *vamos a ser juntos* twenty four seven
Alt-CS: *vamos a ser juntos* when de for *saben*
Alt-EN: follows a santos twenty for seven
Alt-SP: *vamos hacer junto sentí for saben*

Figure 1: Examples from SpanglishEval – the English-Spanish evaluation dataset. The first sentence in each example is the gold sentence, followed by a generated code-switched alternative, a generated English alternative, and a generated Spanish one. Blue (normal) marks English, while red (italic) marks Spanish.

The process of alternative sentence creation is as follows: (1) Convert a sequence of language-tagged words (either CS or monolingual) into a sequence of the matching phonemes (using pronunciation dictionaries); (2) Decode the sequence of phonemes into new sentences, which include words from either language (possibly both); (3) When decoding, allow minor changes in the sequence of phonemes to facilitate the differences between the languages.

These steps can be easily implemented using composition of finite-state transducers (FSTs).[4]

For each gold sentence (which can be either code-switched or monolingual) we create alternatives of all three types: (a) code-switched sentences, (b) sentences in L1 only, (c) sentences in L2 only.

We created such a dataset for English-Spanish with gold sentences from the Bangor Miami Corpus (Section 7). Figure 1 shows two examples from the dataset. In each, the gold sentence is followed by a single code-switched alternative, a single English alternative, and a single Spanish one (a subset of the full set).

guage ID is not mandatory, but helps in cases in which a word is found in the vocabulary of both languages and is pronounced differently), compatible pronunciation dictionaries and unigram word probabilities for each of the languages.

[4]Specifically, we compose the following FSTs: (a) an FST for converting a sentence into a sequence of phonemes, (b) an FST that allows minor changes in the phoneme sequence, (c) an FST for decoding a sequence of phonemes into a sentence, the inverse of (a).

| | dev set | test set |
|---|---|---|
| no. of sets | 1000 | 1000 |
| no. of sentences | 30884 | 30811 |
| no. of CS alternatives | 10000 | 10000 |
| no. of English alternatives | 9999 | 9993 |
| no. of Spanish alternatives | 9885 | 9818 |
| no. of CS gold sentences | 250/1000 | 250/1000 |

Table 2: Statistics of the dataset.

### 4.1 Technical details

When creating code-switched alternatives, we want to encourage the creation of sentences that include both languages, and that differ from each other. This is done with scores determined by some heuristics, such as preferring sentences that include more words from the language that was less dominant in the original one and vice versa. As part of this we also try to avoid noise from addition of short frequent words, by encouraging the average word length to be high[5]. We create 1000-best alternatives from the FST, re-score them according to the heuristic and keep the top 10.

We discard sets in which the gold sentence has less than three words (excluding punctuation), and also sets with less than 5 alternatives in one of the three types.

We randomly choose 250 sets in which the gold sentence is code-switched, and 750 sets in which the gold sentence is monolingual, both for the development set and for the test set. This percentage of CS sentences is higher than in the underlying corpus in order to aid a meaningful comparison between the models regarding their ability to prefer gold CS sentences over alternatives. The statistics of the dataset are detailed in Table 2.

Further details regarding the implementation can be found in the Appendix.

## 5 Using Monolingual Data for CS LM

Data for code-switching is relatively scarce, while monolingual data is easy to obtain. The question then is *how do we efficiently incorporate monolingual data when training a CS LM?*

We use an effective training protocol (FINETUNED) for incorporating monolingual data into the language model, similar to the best protocol introduced in (Baheti et al., 2017). We first pre-train a model with monolingual sentences

[5]The implementation of these heuristics is part of our code that is available online.

from both English and Spanish (with shared vocabulary for both languages). This essentially trains two monolingual models, one for English and one for Spanish, but with full sharing of parameters. Note that in this pre-training phase, the model is not exposed to any code-switched sentence.

We then use the little amount of available code-switched data to further train the model, making it familiar with code-switching examples that mix the two languages. This fine-tunning procedure enables the model to learn to correctly combine the two languages in a single sentence.

We show in Section 8 that adding the CS data only at the end, in the described manner, works substantially better than several alternatives, verifying the results from (Baheti et al., 2017).

## 6 Discriminative Training

Our new evaluation method gives rise to training models that are designated to the ranking task. As our main goal is to choose a single best sentence out of a set of candidate sentences, we can focus on training models that score whole sentences with discriminative training, rather than using the standard probability setting of LMs. Discriminative approach unbinds us from the burden of estimating probability distributions over all the words in the vocabulary and allows us to simply create representations of sentences and match them with scores.

Using negative examples is not straight forward in LM training, but is essential for speech recognition, where the language model needs to distinguish between "good" and "bad" sentences. Our discriminative training is based on the idea of using both positive and negative examples during training. The training data we use is similar in nature to that of our test data: sets of sentences in which only a single one is a genuine example collected from a real CS corpus, while all the others are synthetically created.

During training, we aim to assign the gold sentences with a higher score than the scores of the others. For each sentence, we require the difference between its score and the score of the gold sentence to be as large as its WER with respect to the gold sentence. This way, the farther a sentence is from the gold one, the lower its score is.

Formally, let $s_1$ be the gold sentence, and $s_2, ..., s_m$ be the other sentences in that set. The loss of the set is the sum of losses over all sentences, except for the gold one:

$$\sum_{i=2}^{m} \max(0, \text{WER}(s_1, s_i) - [\text{score}(s_1) - \text{score}(s_i)])$$

where $\text{score}(s_i)$ is computed by the multiplication of the representation of $s_i$ and a learned vector $w$:

$$\text{score}(s_i) = w \cdot \text{repr}(s_i)$$

A sentence is represented with its BiLSTM representation – the concatenation of the final states of forward and backward LSTMs. Formally, a sentence $s = w_1, ..., w_n$ is represented as follows:

$$\text{repr}(s) = LSTM(w_1, ..., w_n) \circ LSTM(w_n, ..., w_1)$$

where $\circ$ is the concatenation operation.

**Incorporating monolingual data** In order to use monolingual data in the case of discriminative training, we follow the same protocol: as a first step, we create alternative sentences for each monolingual sentence from the monolingual corpora. We train a model using this data, and as a next step, we fine-tune this model with the sets of sentences that are created from the CS corpus.

## 7 Empirical Experiments

**Models and Baselines** We report results on two models that use discriminative training: CS-ONLY-DISCRIMINATIVE only trains on data that is created based on the small code-switched corpus, while FINE-TUNED-DISCRIMINATIVE first trains on data created based on the monolingual corpora and is then fine-tuned using the data created from the code-switched corpus.

We compare our models to several baselines, all of which use standard LM training: ENGLISHONLY-LM and SPANISHONLY-LM train on the monolingual data only. Two additional models train on a combination of the code-switched corpus and the two monolingual corpora: the first (ALL:SHUFFLED-LM) trains on all sentences (monolingual and code-switched) presented in a random order. The second (ALL:CS-LAST-LM) trains each epoch on the monolingual datasets followed by a pass on the small code-switched corpus. The models CS-ONLY-LM and FINE-TUNED-LM are the equivalents of CS-ONLY-DISCRIMINATIVE and FINE-TUNED-DISCRIMINATIVE but with standard LM training.

**Code-switching corpus** We use the Bangor Miami Corpus, consisting of transcripts of conversations by Spanish-speakers in Florida, all of whom are bilingual in English.[6]

We split the sentences (45,621 in total) to train/dev/test with ratio of 60/20/20 respectively, and evaluate perplexity on the dev and test sets.

The dataset described in Section 4 is based on sentences from the dev and test sets, and serves as our main evaluation method.

**Monolingual corpora** The monolingual corpora used for training the English and Spanish monolingual models are taken from the OpenSubtitles2018 corpus (Tiedemann, 2009),[7] of subtitles of movies and TV series.

We use 1M lines from each language, with a split of 60/20/20 for train/dev/test, respectively. The test set is reserved for future use. For discriminative training we use 1/6 of the monolingual training data (as creating the data results in roughly 30 sentences per gold sentence).

Additional details on preprocessing and statistics of the data can be found in the Appendix.

**Training** We implement our language models in DyNet (Neubig et al., 2017). Our basic configuration is similar to that of Gal and Ghahramani (2016) with minor changes. It has a standard architecture of a 2-layer LSTM followed by a softmax layer, and the optimization is done with SGD (see Appendix for details).

Tuning of hyper-parameters was done on the PTB corpus, in order to be on par with state-of-the-art models such as that of Merity et al. (2017). We then trained the same LM on our CS corpus with no additional tuning and got perplexity of 44.06, better than the 52.99 of Merity et al. (2017) when using their default parameters on the CS corpus.[8] We thus make no further tuning of hyper-parameters.

When changing to discriminative training, we perform minimal necessary changes: discarding weight decay and reducing the learning rate to 1. The weight vector in the discriminative setting is learned jointly with the other parameters of the network.

[6] http://bangortalk.org.uk/speakers.php?c=miami

[7] http://opus.nlpl.eu/OpenSubtitles2018.php http://www.opensubtitles.org/

[8] https://github.com/salesforce/awd-lstm-lm

| | dev | | | test | | |
|---|---|---|---|---|---|---|
| | perp ↓ | acc ↑ | wer ↓ | perp ↓ | acc ↑ | wer ↓ |
| SPANISH-ONLY-LM | 329.68 | 26.6 | 30.47 | 322.26 | 25.1 | 29.62 |
| ENGLISH-ONLY-LM | 320.92 | 29.3 | 32.02 | 314.04 | 30.3 | 32.51 |
| ALL:CS-LAST-LM | 76.64 | 47.8 | 14.56 | 76.97 | 49.2 | 14.13 |
| ALL:SHUFFLED-LM | 68.00 | 51.8 | 13.64 | 68.72 | 51.4 | 13.89 |
| CS-ONLY-LM | 43.20 | 60.7 | 12.60 | 43.42 | 57.9 | 12.18 |
| CS-ONLY+VOCAB-LM | 45.61 | 61.0 | 12.56 | 45.79 | 58.8 | 12.49 |
| FINE-TUNED-LM | 39.76 | 66.9 | 10.71 | 40.11 | 65.4 | 10.17 |
| CS-ONLY-DISC | – | 72.0 | 6.35 | – | 70.5 | 6.70 |
| FINE-TUNED-DISC | – | **74.2** | **5.85** | – | **75.5** | **5.59** |

Table 3: Results on the dev set and on the test set. "perp" stands for perplexity, "acc" stands for accuracy (in percents), and "wer" stands for word-error-rate.

As done in previous work, in order to be able to give a reliable probability to every next-token in the test set, we include all the tokens from the test set in our vocabulary and we do not use the "<unk>" token. We only train those that appear in the train set. Handling of unknown tokens becomes easier with discriminative training, as scores are assigned to full sentences. However, we leave this issue for future work.

## 8 Results

The results of the different models are presented in Table 3. For each model we report both perplexity and accuracy (except for discriminative training, where perplexity is not valid), where each of them is reported according to the best performing model on that measure (on the dev set). We also report the WER of all models, which correlates perfectly with the accuracy measure.

### 8.1 Using monolingual data

As mentioned above, both in standard LM and in discriminative training, using monolingual data in a correct manner (FINE-TUNED-LM and FINE-TUNED-DISCRIMINATIVE) significantly improves over using solely the code-switching data. In standard LM, adding monolingual data results in an improvement of 7.5 points (improving from an accuracy of 57.9% to 65.4%), and in the discriminative training it results in an improvement of 5 points (improving from an accuracy of 70.5% to 75.5%). Even though both ALL:SHUFFLED-LM and ALL:CS-LAST-LM use the same data as the FINE-TUNED-LM model, they perform even worse than CS-ONLY-LM that does not use the monolingual data at all. This emphasizes that the manner of integration of the monolingual data has a very strong influence.

Note that the FINE-TUNED-LM model also im-

| | 25% train | | 50% train | | 75% train | | full train | |
|---|---|---|---|---|---|---|---|---|
| | dev | test | dev | test | dev | test | dev | test |
| CS-ONLY | 58.4 | 58.9 | 65.2 | 63.6 | 70.8 | 68.8 | 72.0 | 70.5 |
| FINE-TUNED | **68.4** | **67.7** | **71.9** | **70.1** | **72.8** | **73.0** | **74.2** | **75.5** |

Table 4: Results on the dev set and on the test set using discriminative training with only subsets of the code-switched data.

proves perplexity. As perplexity is significantly affected by the size of the vocabulary—and to ensure fair comparison—we also add the additional vocabulary from the monolingual data to CS-ONLY-LM (CS-ONLY+VOCAB-LM). Extending the vocabulary without training those additional words, results in a 2.37-points loss on the perplexity measure, while our evaluation metric (accuracy) stays essentially the same. This demonstrates the utility of our proposed evaluation compared to using perplexity, allowing it to fairly compare models with different vocabulary sizes.

In order to examine the contribution of the monolingual data, we also experimented with subsets of the code-switching training data. Table 4 depicts the results when using subsets of the CS training data with discriminative training. The less code-switching data we use, the more the effect of using the monolingual data is significant: we gain 8.8, 6.5, 4.2 and 5 more accuracy points with 25%, 50%, 75% and 100% of the data, respectively. In the case of 25% of the data, the FINE-TUNED-DISCRIMINATIVE model improves over CS-ONLY-DISCRIMINATIVE by 17 relative percents.

### 8.2 Standard LMs vs. Discriminative Training

In the standard training setting, The FINE-TUNED-LM baseline is the strongest baseline, outperforming all others with an accuracy of 65.4%. Similarly, when using discriminative training, the FINE-TUNED-DISCRIMINATIVE model outperforms the CS-ONLY-DISCRIMINATIVE model. Note that using discriminative training, even with no additional monolingual data, leads to better performance than that of the best language model: the CS-ONLY-DISCRIMINATIVE model achieves an accuracy of 70.5%, 5.1 points more than the accuracy of the FINE-TUNED-LM model. We gain further improvement by adding monolingual data and get an even higher accuracy of 75.5%, which is 10.1 points higher than the best language model.

**Limitations** Our evaluation setup in the discriminative training case is not ideal: the negative samples in both the train and test sets are artificially created by the same mechanism. Thus, high performance on the test set in the discriminative case may result from "leakage" in which the model learns to rely on idiosyncrasies and artifcats of the data generation mechanism. As such, these results may not transfer as is to a real-world ASR scenario, and should be considered as an optimistic estimate of the actual gains.[9] Nevertheless, we do believe that the accuracy improvements of the discriminative setup are real, and should translate to improvements also in the real-world scenario.

To alleviate the concerns to some extent, we consider the case in which the leakage is in the induced negative words distribution.[10] We re-train the discriminative model using a BOW representation of the sentence. This results in accuracies of 52.1% and 47.4% for dev and test, respectively, far below those of the basic model. A model that leaks in word distribution would score much higher in this evaluation, indicating that the improvement is likely not due to this form of leakage, and that most of the improvement of the discriminative training is likely to translate to a real ASR scenario.

## 9 Analysis

Table 5 breaks down the results of the different models according to two conditions: when the gold sentence is code-switched, and when the gold sentence is monolingual.

As expected, the FINE-TUNED-DISCRIMINATIVE model is able to prioritize the gold sentence better than all other models, under both conditions. The improvement we get is most significant when the gold sentence is CS: in those cases we get a dramatic improvement of 27.73 accuracy points (a relative improvement of 58%). Note that for the standard LMs, the cases

[9] An ideal evaluation setup will use an artificial dataset for training, and a dataset obtained from an acoustic model of a code-switched ASR system for testing. However, such an evaluation setup requires access to a high-quality code-switched acoustic model, which we do not posses nor have the means to obtain. Furthermore, it would also tie the evaluation to a specific ASR system, which we aim to avoid.

[10] As our generation process, much like an acoustic model, makes only local word replacements, the choice of replacement words is the most salient difference from a real acoustic model, and has the largest leakage potential.

| | dev | | test | |
|---|---|---|---|---|
| | CS | mono | CS | mono |
| CS-ONLY-LM | 45.20 | 65.87 | 43.20 | 62.80 |
| FINE-TUNED-LM | 49.60 | 72.67 | 47.60 | 71.33 |
| CS-ONLY-DISC | **75.60** | 70.40 | 70.80 | 70.53 |
| FINE-TUNED-DISC | 70.80 | **74.40** | **75.33** | **75.87** |

Table 5: Accuracy on the dev set and on the test set, according to the type of the gold sentence in the set: code-switched (CS) vs. monolingual (mono).

in which the gold sentence is CS are much harder, and they perform badly on this portion of the test set. However, using discriminative learning enable us to get improved performance on both portions of the test set and to get comparable results on both parts.

A closer examination of the mistakes of the FINE-TUNED-LM and FINE-TUNED-DISCRIMINATIVE models reveals the superiority of the discriminative training in various cases. Table 6 presents several examples in which FINE-TUNED-LM prefers a wrong sentence whereas FINE-TUNED-DISCRIMINATIVE identifies the gold one. In examples 1–4 the gold sentence was code-switched but FINE-TUNED-LM forced an improbable monolingual one. Examples 5 and 6 show mistakes in monolingual sentences.

While discriminative training is significantly better than the standard LM training, it can still be improved quite a bit. Table 7 lists some of the mistakes of the FINE-TUNED-DISCRIMINATIVE model: in examples 1, 2 and 3, the gold sentence was code-switched but the model preferred a monolingual one, in example 4 the model prefers a wrong CS sentence over the gold monolingual one, and in 5 and 6 the model makes mistakes in monolingual sentences.

## 10 Related Work

**CS** Most prior work on CS focused on Language Identification (LID) (Solorio et al., 2014; Molina et al., 2016) and POS tagging (Solorio and Liu, 2008; Vyas et al., 2014; Ghosh et al., 2016; Barman et al., 2016). In this work we focus on language modeling, which we find more challenging.

**LM** Language models have been traditionally created by using the n-grams approach (Brown et al., 1992; Chen and Goodman, 1996). Recently, neural models gained more popularity, both using a feed-forward network for an n-gram language model (Bengio et al., 2003; Morin and Bengio, 2005) and using recurrent architectures that are fed with the sequence of words, one word at a time (Mikolov et al., 2010; Zaremba et al., 2014; Gal and Ghahramani, 2016; Foerster et al., 2017; Melis et al., 2017).

Some work has been done also on optimizing LMs for ASR purposes, using discriminative training. Kuo et al. (2002), Roark et al. (2007) and Dikici et al. (2013) all improve LM for ASR by maximizing the probability of the correct candidates. All of them use candidates of ASR systems as "negative" examples and train n-gram LMs or use linear models for classifying candidates. A closer approach to ours is used by Huang et al. (2018). There, they optimize an RNNLM with a discriminative loss as part of training an ASR system. Unlike our proposed model, they still use the standard setting of LM. In addition, their training is coupled with an end-to-end ASR system, in particular, as in previous works, the "negative" examples they use are candidates of that ASR system.

**LM for CS** Some work has been done also specifically on LM for code-switching. In Chan et al. (2009), the authors compare different n-gram language models, Vu et al. (2012) suggest to improve language modeling by generating artificial code-switched text. Li and Fung (2012) propose a language model that incorporates a syntactic constraint and combine both a code-switched LM and a monolingual LM in the decoding process of an ASR system. Later on, they also suggest to incorporate a different syntactic constraint and to learn the language model from bilingual data using it (Li and Fung, 2014). Pratapa et al. (2018) also use a syntactic constraint to improve LM by augmenting synthetically created CS sentences in which this constraint is not violated. Adel et al. (2013a) introduce an RNN based LM, where the output layer is factorized into languages, and POS tags are added to the input. In Adel et al. (2013b), they further investigate an n-gram based factorized LM where each word in the input is concatenated with its POS tag and its language identifier. Adel et al. (2014; 2015) also investigate the influence of syntactic and semantic features in the framework of factorized language models. Sreeram and Sinha (2017) also use a factorize LM with the addition of POS tags. Baheti et al. (2017) explore several different training protocols for CS LM and find that fine-tuning with CS data after pretrain-

| no. | type | Gold sentence | Choice of FINE-TUNED-LM model |
|---|---|---|---|
| 1 | CS → mono | and in front of everybody *me saltó* . | and in front of everybody muscle too . |
| 2 | CS → mono | *porque* the sewer system has them in there *porque* . | *porque de ser sistemas de mil de porque* . |
| 3 | CS → mono | *entonces* what i had done is gone ahead and printed it out . | and all says what i had done is gone ahead and printed it out . |
| 4 | CS → mono | *es que creo que quedan como novecientos oportunidades para* beta testers . | *es que creo que que del como novecientos oportunidad esperaba de estar* . |
| 5 | mono → mono | we we stop beyond getting too cruel . | we we stop be and getting to cruel . |
| 6 | mono → mono | *en mi casa tengo tanto huevo duro* . | *en mi casa tengo tanto a futuro* . |

Table 6: Examples of sentences the FINE-TUNED-DISCRIMINATIVE model identifies correctly while the FINE-TUNED-LM model does not.

| no. | type | Gold sentence | Choice of FINE-TUNED-DISCRIMINATIVE model |
|---|---|---|---|
| 1 | CS → mono | *que* by the way *se vino ilegal* . | *que va de esa vino ilegal* . |
| 2 | CS → mono | *son un* website there . | so noon website there . |
| 3 | CS → mono | no i have never felt nothing close to the *espíritu santo* never . | no i have never felt nothing close to the a spirits on to never . |
| 4 | mono → CS | *Tú sabes que que el cuerpo empieza a sentirse raro* . | *Tú sabes que que el cuerpo empieza sentir ser* arrow . |
| 5 | mono → mono | *bueno ya son las las y cuarenta casi casi* . | *bueno ya son las lassie cuarenta que si casi* . |
| 6 | mono → mono | we have a peter lang . | we have a bitter lung . |

Table 7: Examples of sentences that the FINE-TUNED-DISCRIMINATIVE model fails to identify.

ing on monolingual data works best. Finally, another line of works suggests using a dual language model, where two monolingual LMs are combined by a probabilistic model (Garg et al., 2017, 2018).

No standard benchmark or evaluation setup exists for CS LM, and most previous works use an ASR system as part of their evaluation setup. This makes comparison of methods very challenging. Indeed, all the works listed above use different setups and don't compare to each other, even for works coming from the same group. We believe the evaluation setup we propose in this work and our English-Spanish dataset, which is easy to replicate and decoupled from an ASR system, is a needed step towards meaningful comparison of CS LM approaches.

## 11 Conclusions and Future Work

We consider the topic of language modeling for code-switched data. We propose a novel ranking-based evaluation method for language models, motivated by speech recognition, and create an evaluation dataset for English-Spanish code-switching LM (SpanglishEval).

We further present a discriminative training for this ranking task that is intended for ASR purposes. This training procedure is not bound to probability distributions, and uses both positive and negative training sentences. This significantly improves performance. Such discriminative training can also be applied to monolingual data.

Finally, we verify the effectiveness of the training protocol for CS LM presented in Baheti et al. (2017): pre-training on a mix of monolingual sentences, followed by fine-tuning on a code-switched dataset. This protocol significantly outperforms various baselines. Moreover, we show that the less code-switched training data we use, the more effective it is to incorporate the monolingual data.

Our proposed evaluation framework and dataset will facilitate future work by providing the ability to meaningfully compare the performance of different methods to each other, an ability that was sorely missing in previous work.

## Acknowledgments

The work was supported by The Israeli Science Foundation (grant number 1555/15).

## References

Heike Adel, Katrin Kirchhoff, Dominic Telaar, Ngoc Thang Vu, Tim Schlippe, and Tanja Schultz. 2014. Features for factored language models for code-switching speech. In *Proceedings of SLTU*.

Heike Adel, Ngoc Thang Vu, Katrin Kirchhoff, Dominic Telaar, and Tanja Schultz. 2015. Syntactic and semantic features for code-switching factored language models. *IEEE Transactions on Audio, Speech, and Language Processing*, 23(3).

Heike Adel, Ngoc Thang Vu, Franziska Kraus, Tim Schlippe, Haizhou Li, and Tanja Schultz. 2013a. Recurrent neural network language modeling for code switching conversational speech. In *Proceedings of ICASSP, IEEE International Conference*.

Heike Adel, Ngoc Thang Vu, and Tanja Schultz. 2013b. Combination of recurrent neural networks and factored language models for code-switching language modeling. In *Proceedings of ACL*.

Peter Auer. 1999. From codeswitching via language mixing to fused lects: Toward a dynamic typology of bilingual speech. *International journal of bilingualism*, 3(4):309–332.

Ashutosh Baheti, Sunayana Sitaram, Monojit Choudhury, and Kalika Bali. 2017. Curriculum design for code-switching: Experiments with language identification and language modeling with deep neural networks. In *Proceedings of the 14th International Conference on Natural Language Processing (ICON-2017)*.

Utsab Barman, Joachim Wagner, and Jennifer Foster. 2016. Part-of-speech tagging of code-mixed social media content: Pipeline, stacking and joint modelling. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3:1137–1155.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

Barbara E. Bullock and Almeida Jacqueline Toribio. 2009. *The Cambridge handbook of linguistic code-switching*. Cambridge University Press Cambridge.

Özlem Çetinoğlu, Sarah Schulz, and Ngoc Thang Vu. 2016. Challenges of computational processing of code-switching. In *Proceedings of the 2nd Workshop on Computational Approaches to Linguistic Code Switching, EMNLP*.

Joyce YC Chan, Houwei Cao, PC Ching, and Tan Lee. 2009. Automatic recognition of cantonese-english code-mixing speech. *Computational Linguistics and Chinese Language Processing*, 14(3):281–304.

Stanley F Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of ACL*, pages 310–318.

Ryan Cotterell, Adithya Renduchintala, Naomi Saphra, and Chris Callison-Burch. 2014. An algerian arabic-french code-switched corpus. In *Workshop on Free/Open-Source Arabic Corpora and Corpora Processing Tools Workshop Programme*.

Erinç Dikici, Murat Semerci, Murat Saraclar, and Ethem Alpaydin. 2013. Classification and ranking approaches to discriminative language modeling for asr. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(2):291–300.

Jakob N Foerster, Justin Gilmer, Jan Chorowski, Jascha Sohl-Dickstein, and David Sussillo. 2017. Intelligible language modeling with input switched affine networks. *arXiv preprint arXiv:1611.09434*.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Proceedings of NIPS*.

Saurabh Garg, Tanmay Parekh, and Preethi Jyothi. 2017. Dual language models for code switched speech recognition. *arXiv preprint arXiv:1711.01048*.

Saurabh Garg, Tanmay Parekh, and Preethi Jyothi. 2018. Code-switched language models using dual rnns and same-source pretraining. *arXiv preprint arXiv:1809.01962*.

Souvick Ghosh, Satanu Ghosh, and Dipankar Das. 2016. Part-of-speech tagging of code-mixed social media text. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching*.

Jiaji Huang, Yi Li, Wei Ping, and Liang Huang. 2018. Large margin neural language model. In *Proceedings of EMNLP*.

Hong-Kwang Jeff Kuo, Eric Fosler-Lussier, Hui Jiang, and Chin-Hui Lee. 2002. Discriminative training of language models for speech recognition. In *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, volume 1. IEEE.

Ying Li and Pascale Fung. 2012. Code-switch language model with inversion constraints for mixed language speech recognition. In *Proceedings of COLING*, pages 1671–1680.

Ying Li and Pascale Fung. 2014. Language modeling with functional head constraint for code switching speech recognition. In *Proceedings of EMNLP*, pages 907–916.

Gábor Melis, Chris Dyer, and Phil Blunsom. 2017. On the state of the art of evaluation in neural language models. *arXiv:1707.05589*.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017. Regularizing and optimizing lstm language models. *arXiv:1708.02182*.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of Interspeech*.

Giovanni Molina, Fahad AlGhamdi, Mahmoud Ghoneim, Abdelati Hawwari, Nicolas Rey-Villamizar, Mona Diab, and Thamar Solorio. 2016. Overview for the second shared task on language identification in code-switched data. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching*.

Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *Proceedings of the International Workshop on Artificial Intelligence and Statistics*, volume 5.

Pieter Muysken. 2000. *Bilingual speech: A typology of code-mixing*, volume 11. Cambridge University Press.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, et al. 2017. Dynet: The dynamic neural network toolkit. *arXiv:1701.03980*.

Shana Poplack. 1980. Sometimes i'll start a sentence in spanish y termino en español: toward a typology of code-switching. *Linguistics*, 18(7-8):581–618.

Adithya Pratapa, Gayatri Bhat, Monojit Choudhury, Sunayana Sitaram, Sandipan Dandapat, and Kalika Bali. 2018. Language modeling for code-mixing: The role of linguistic theory based synthetic data. In *Proceedings of ACL*.

Brian Roark, Murat Saraclar, and Michael Collins. 2007. Discriminative n-gram language modeling. *Computer Speech & Language*, 21(2):373–392.

Rico Sennrich. 2016. How grammatical is character-level neural machine translation? assessing MT quality with contrastive translation pairs. *arXiv:1612.04629*.

Thamar Solorio, Elizabeth Blair, Suraj Maharjan, Steven Bethard, Mona Diab, Mahmoud Ghoneim, Abdelati Hawwari, Fahad AlGhamdi, Julia Hirschberg, Alison Chang, and Pascale Fung. 2014. Overview for the first shared task on language identification in code-switched data. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*.

Thamar Solorio and Yang Liu. 2008. Part-of-speech tagging for english-spanish code-switched text. In *Proceedings of EMNLP*, pages 1051–1060.

Ganji Sreeram and Rohit Sinha. 2017. Language modeling for code-switched data: Challenges and approaches. *arXiv:1711.03541*.

Jörg Tiedemann. 2009. News from OPUS - A collection of multilingual parallel corpora with tools and interfaces. In N. Nicolov, K. Bontcheva, G. Angelova, and R. Mitkov, editors, *Recent Advances in Natural Language Processing*, pages 237–248.

Ke Tran, Arianna Bisazza, and Christof Monz. 2018. The importance of being recurrent for modeling hierarchical structure. *arXiv:1803.03585*.

Ngoc Thang Vu, Dau-Cheng Lyu, Jochen Weiner, Dominic Telaar, Tim Schlippe, Fabian Blaicher, Eng-Siong Chng, Tanja Schultz, and Haizhou Li. 2012. A first speech recognition system for mandarin-english code-switch conversational speech. In *Proceedings of ICASSP, IEEE International Conference*, pages 4889–4892.

Yogarshi Vyas, Spandana Gella, Jatin Sharma, Kalika Bali, and Monojit Choudhury. 2014. POS tagging of English-Hindi code-mixed social media content. In *Proceedings of EMNLP*.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv:1409.2329*.

# Using Local Knowledge Graph Construction to Scale Seq2Seq Models to Multi-Document Inputs

**Angela Fan**
FAIR / LORIA

**Claire Gardent**
CNRS / LORIA

**Chloé Braud**
CNRS / LORIA

**Antoine Bordes**
FAIR

angelafan,abordes@fb.com
claire.gardent,chloe.braud@loria.fr

## Abstract

Query-based open-domain NLP tasks require information synthesis from long and diverse web results. Current approaches extractively select portions of web text as input to Sequence-to-Sequence models using methods such as TF-IDF ranking. We propose constructing a local graph structured knowledge base for each query, which compresses the web search information and reduces redundancy. We show that by linearizing the graph into a structured input sequence, models can encode the graph representations within a standard Sequence-to-Sequence setting. For two generative tasks with very long text input, long-form question answering and multi-document summarization, feeding graph representations as input can achieve better performance than using retrieved text portions.

## 1 Introduction

Effective information synthesis is at the core of many Natural Language Processing applications, such as open-domain question answering and multi-document summarization. In such tasks, a fundamental challenge is the ability to distill relevant knowledge from hundreds of thousands of tokens of noisy and redundant input such as webpages. Current approaches predominantly conduct TF-IDF-based information extraction to identify key portions of the information, and then provide this as sequential input to a Sequence-to-Sequence (Seq2Seq) model. The sub-selected portions are limited to a few thousand words, as models often struggle to encode much longer sequences.

In this work, we propose a mechanism to restructure free text into local knowledge graphs that are then linearized into sequences, creating a canonical form in which information is presented to models. By constructing a graph intermediary, redundant information can be merged or discarded, producing substantially compressed input

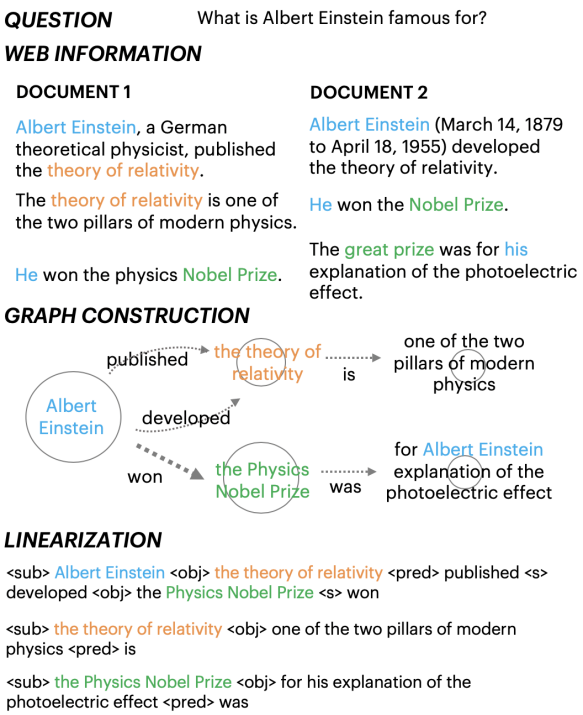


Figure 1: **Multi-Document Input to Linearized Graph** Multi-document input resulting from web search queries are converted to a graph structured knowledge base using coreference resolution and information extraction, then linearized into a sequence for Seq2Seq models. Color indicates coreference resolution. Node weight is indicated by circle radius and edge weight by line thickness.

— small enough to be fully encoded by Seq2Seq models. Such a method can be seen as merging previous work on symbolic knowledge bases for information extraction with newer approaches using deep neural networks to encode knowledge.

Our approach, shown in Figure 1, takes a query and its corresponding multi-document web search results and builds for each query a specific local knowledge graph. We present several modeling contributions to effectively encode the entire graph as a sequence and attend to the most relevant portions within this linearization. We demonstrate the effectiveness of this approach on two large-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4186–4196,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

scale generative tasks with both long and noisy multi-document web input and paragraph length output: long-form question answering on the ELI5 dataset (Fan et al., 2019) and Wikipedia lead paragraph generation as a multi-document summarization problem (Liu et al., 2018b).

## 2 Related Work

Interest in generative sequence modeling has intensified due to recent improvements (Peters et al., 2018; Devlin et al., 2018; Radford et al., 2019), making the challenge of information synthesis more relevant. In contrast to extractive tasks which only require models to identify spans and can do so effectively on long documents by looking at the paragraphs independently, generative sequence models must combine multiple pieces of evidence from long and noisy multi-document input to generate correct and convincing responses.

### 2.1 Multi-Document Input

Previous work in multi-document summarization (Barzilay et al., 1999) applies various techniques to handle long input, including clustering to find similar information (Honarpisheh et al., 2008), extractive methods to select relevant sentences (Daumé III and Marcu, 2002; Gillick and Favre, 2009; Berg-Kirkpatrick et al., 2011; Di Fabbrizio et al., 2014; Bing et al., 2015; Cao et al., 2017) including maximal marginal relevance (Fabbri et al., 2019), and incorporating queries (Baumel et al., 2018) and graphs (Ganesan et al., 2010; Yasunaga et al., 2017). However, there are few large scale multi-document summarization datasets and many approaches have focused on extractive selection or hybrid extractive-abstractive models. In this work, we use graph construction to re-structure multi-document input for abstractive generation.

Advancements in question answering have examined performance on datasets with multi-document input, such as TriviaQA (Joshi et al., 2017). Various approaches have been proposed, including leveraging TF-IDF and bigram hashing with an RNN to find relevant information (Chen et al., 2017), models that score individual paragraphs for sub-selection (Clark and Gardner, 2017), and nearest neighbor search with paragraph re-ranking (Das et al., 2018a). However, these approaches have been applied to extractive question answering tasks that require span identification, rather than abstractive text generation in an information synthesis setting.

### 2.2 Using Knowledge Bases

Previous work has explored various ways of representing information in knowledge bases (Bordes et al., 2011) and improving these representations (Chen et al., 2013). Knowledge bases have been leveraged to improve performance on various tasks, from coreference resolution (Ng and Cardie, 2002) and question answering (Zheng, 2003; Bao et al., 2014; Cui et al., 2017; Sun et al., 2018) to signal processing (Bückner et al., 2002). Various works convert text into Abstract Meaning Representations (Liu et al., 2018a) for domains such as news (Vossen et al., 2015; Rospocher et al., 2016) and link nodes to large knowledge bases such as DBPedia (Auer et al., 2007). Wities et al. (2017) combine open information extraction with coreference and lexical inference to build knowledge representations. They apply this to tweets and analyze the accuracy on various aspects of graph construction. Das et al. (2018b) construct graphs from procedural text to track entity position to answer when and if entities are created, destroyed, or moved. In contrast, we build graphs from substantially longer multi-document input and use them for multi-sentence text generation.

Recently, many have explored neural architectures that can encode graph structured input (Bruna et al., 2013; Kipf and Welling, 2016; Beck et al., 2018; Zhou et al., 2018; Xu et al., 2018; Lai et al., 2019). These models often represent graphs as adjacency matrices to generalize architectures such as convolutional networks to graph inputs. Rather than encoding a static knowledge graph or leveraging external knowledge graphs, we build a local graph for each query and model these using standard Seq2Seq models. We leave the incorporation of graph networks for future work.

## 3 Graph Construction

We describe how symbolic graph representations of knowledge can be constructed from text. Our approach assumes a multi-document input (such as web pages) resulting from the execution of a query. The graph construction process (1) compresses the web search input to a significantly smaller size, allowing models to encode the entirety of the compression, and (2) reduces redundancy through merge operations, allowing relevant information to be more easily identified.

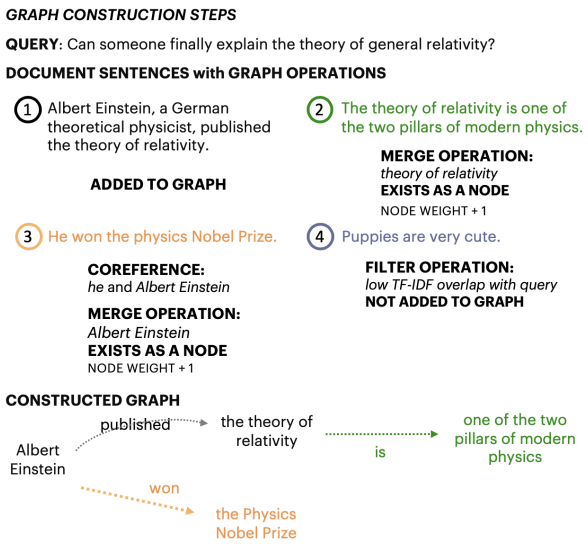


Figure 2: Steps of **Graph Construction**. Color relates the document sentence used to produce the graph output.

**Text to Triples to Graph** Graph construction proceeds in several steps outlined in Figure 2. We apply *Coreference Resolution* (Clark and Manning, 2016a,b)[1] and *Open Information Extraction* (Stanovsky et al., 2018)[2] to convert sentences into a *Triple* of the form (subject, predicate, object). The sentence *Albert Einstein, a German theoretical physicist, won the Nobel Prize* would become (*Albert Einstein, won, the Nobel Prize*).

A graph is constructed using the triples by representing subjects and objects as nodes connected by predicates as directed edges. For example, the triple would become *Albert Einstein* $\xrightarrow[\text{won}]{}$ *the Nobel Prize*. Nodes and edges have a *name* property that is the text they represent. They also have a *weight* property that denotes the number of times that node or edge has appeared. For example, in Figure 1, the node with name *Albert Einstein* has weight 4 and edge with name *won* has weight 2.

**Merging Nodes and Edges** When subsequent triples are added to the graph, they are merged with the existing graph if they already exist to reduce information replication. To merge nodes, the TF-IDF overlap of the new node's name is calculated with the existing graph node names, and the new node is merged into an existing node if the TF-IDF is higher than some threshold (see Figure 2, steps 2 and 3 for example merge operations). Edges are merged similarly with existing edges between the same two nodes. Such merge operations allow strings such as *the Nobel Prize* and *Nobel Prize* to be represented as one node rather than separately. Similarly, coreference resolution aids in merging — by identifying that *Albert Einstein* and *He* refer to the same entity and thus merging them, the construction of the graph reduces redundancy. The size of the graph can be modified by controlling which triples are added using TF-IDF overlap (see Figure 2, step 4). TF-IDF overlap of the triple with the question can be used to determine if the triple contains relevant information. This improves robustness to noisy web search input and helps filter entirely irrelevant portions, such as scraped HTML tags.

[1]We use the implementation available here: https://github.com/huggingface/neuralcoref

[2]We use the implementation available here: https://github.com/gabrielStanovsky/supervised-oie

## 4 Modeling Graphs as Sequences

Current models for text generation often use Seq2Seq architectures such as the Transformer (Vaswani et al., 2017). These models are designed to encode sequences rather than graphs. We describe now how to convert a graph into a structured input sequence. Our complete model will take as input a linearized graph by encoding graph attributes such as node and edge weight as embeddings. We add hierarchical and memory-compressed attention mechanisms to scale Seq2Seq models to encode the full graph and attend to the most relevant information within it (Figure 4), and finally we integrate the benefits of language modeling using multi-task training.

### 4.1 Graph to Sequence

**Linearization** To represent the graph as a sequence for Seq2Seq, it is linearized into a structured standard form of subject node, object node, and predicate edge, separated by indicator tokens, as shown in Figure 1. For example, the linearization *<sub> Albert Einstein <obj> the Nobel Prize <pred> won* would be created. The linearization is accomplished through graph traversal in a breadth-first manner following the directed edges formed by predicates and starting from the node with the largest weight as the root. For two nodes that are connected by multiple predicates, the predicates are concatenated (shown in Figure 1), so a linearization such as *<pred> won <s> received* would indicate that Albert Einstein both won and received the Nobel Prize.

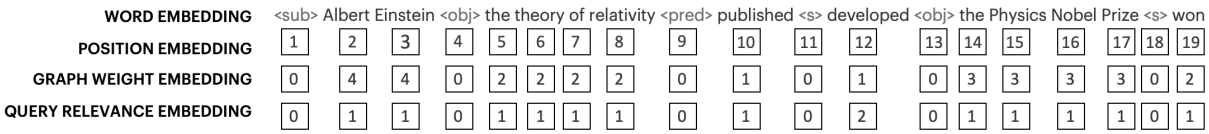


Figure 3: **Graph Attribute Embeddings**. In addition to word and position embeddings, models receive a Graph Weight embedding to encode node and edge weight and a Query Relevance embedding that encodes search result rank.

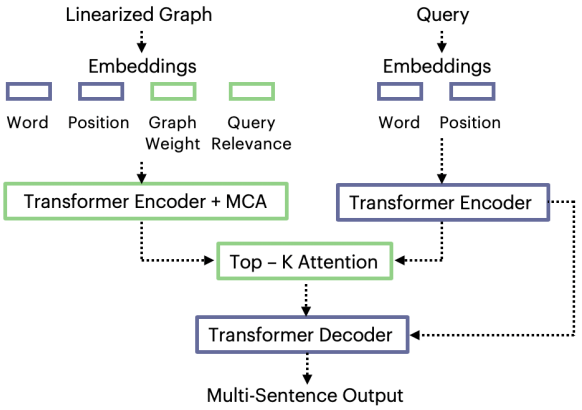


Figure 4: **Model Architecture**. Gray indicates standard Transformer elements, Green indicates modification.

**Encoding Graph Information** Transformer Seq2Seq models have two embeddings: a word embedding and a position embedding. Simply linearizing the graph, however, loses attribute information such as node and edge weight. Instead, we encode these attributes as embeddings in addition to the word and position embeddings.

To represent *Graph Weight* (GW), node and edge weight is provided as an embedding for each token. The node weight and edge weight are equivalent to the number of merge operations + 1. For example, if *Albert Einstein* occurred 4 times in the text, the GW embedding for the tokens *Albert* and *Einstein* would be 4, as shown in Figure 3.

We encode a *Query Relevance* (QR) embedding to represent the relevance of the web search to the query as ranked by the information retrieval system (e.g. search engine). Information from the top web search results is likely more relevant than information from the last web search results, so providing this embedding allows the model to distinguish between these different information sources. In Figure 3, tokens representing sentences from the first document have QR embedding 1, and tokens from the second document have value 2.

Models now have access to several different types of embeddings, but all embedding information contributes equally as there is no mechanism to distinguish between them. We introduce a mechanism for models to scale the graph embeddings. We denote the embedding for position $t$ as $e_t$, such that $e_t^{\text{word}}$ is the word embedding.

For the GW embedding, models learn a gating function $g$ based on the word and GW embeddings. Such a mechanism provides capacity for the model to decide when the additional embeddings are useful based on the words in the input. The gate is calculated by applying an MLP $W$ to the concatenation of the word and GW embeddings. The learned gate is then applied to GW embeddings to create the output $h$:

$$g_t^{\text{GW}} = W[e_t^{\text{GW}}; e_t^{\text{word}}]$$
$$h_t^{\text{GW}} = g_t^{\text{GW}} \circ e_t^{\text{GW}}$$

Models learn a gating mechanism for the QR embedding in a similar manner. The full embedding the model receives is as follows:

$$e_t^{\text{word}} + e_t^{\text{pos}} + [h_t^{\text{GW}}; h_t^{\text{QR}}]$$

### 4.2 Hierarchical Attention

One challenge in modeling long sequences is that attention mechanisms struggle to make sharp selections when softmax-ing over long sequences (Fan et al., 2018). When attention is blurry, there lacks a strong distinction between noise and relevant information.

We assume that graphs are constructed from query-based web search input and thus leverage this query to learn a subselection operation using *hierarchical top-k attention*, depicted in Figure 4. The query is encoded with a Transformer encoder and the linearized graph with another Transformer encoder. We model the interaction between the query and the input sequence (e.g. web search results or linearized graph) by computing an attention distribution between the question and the input, then selecting the top $k$ positions with the most attention weight. Such a mechanism can be thought of as building a query-dependent representation of the most relevant knowledge, which is commonly done in question answering architectures like BiDAF (Seo et al., 2017). The top $k$

operation limits the number of tokens, making the attention mechanism sharper.

### 4.3 Scaling to Encode the Graph

Recent progress has improved the ability of language models to process longer sequences (Sukhbaatar et al., 2019; Dai et al., 2019), but models remain limited in their capacity to encode long documents. The multi-document results of query-based web search have hundreds of thousands of tokens, beyond the limit of current Seq2Seq models to handle. For example, the ELI5 dataset provides an average of 200K tokens of web search input. However, by compressing the web search results into a knowledge graph, we significantly reduce the number of tokens by an order of magnitude and make it possible for a model to access the entirety of the search information.

To represent the full graph, models must scale to encode around 10K input tokens. The attention mechanism in Transformer architectures becomes computationally expensive for sequences of this length. Instead, we experiment with the *Memory-Compressed Attention* (MCA) mechanism proposed for language models in (Liu et al., 2018b)[3] and apply it to the encoder side of Seq2Seq models. At each self-attention layer, MCA alternates between (1) local attention, computed between smaller chunks of tokens and (2) strided convolutions to reduce the number of keys and values used in attention computation. By adding the MCA mechanism to the encoder (*E-MCA*), we are able to encode the complete linearized graph.

### 4.4 Multi-tasking with KB Completion

Fan et al. (2019) used multi-task training on language modeling and various Seq2Seq tasks to incorporate the benefits of language modeling in Seq2Seq models. We extend this by training additionally on *knowledge graph completion*. Models receive at training time sequences of a linearized graph with nodes, edges, or both selectively masked and must predict the missing content words. For example, models might receive as input *<sub> Albert Einstein <obj> [mask] <pred> won* and need to predict *the Nobel Prize*. This can be seen as both a multi-word extension of the masked language model training proposed in (Devlin et al., 2018) and as learning the task of knowledge base completion (Lacroix et al., 2018; Bordes et al., 2011). At training time, the tasks are distinguished using an indicator token in the input.

## 5 Experimental Setup

We evaluate our approach on two datasets with multi-document web input and multi-sentence abstractive output. We use Seq2Seq models that leverage a query concatenated with web search results that have been processed into a supporting document — e.g. TF-IDF subselection, linearized graph, etc. — to generate long output.

### 5.1 Datasets and Evaluation

**ELI5** First, we experiment with the *Explain Like I'm Five* (ELI5) (Fan et al., 2019) question answering dataset of 270K complex questions paired with multi-sentence, explanatory answers (130 words on average). To facilitate question answering, the dataset provides the top 100 web search hits from querying the question, which results in 200K words on average. See Appendix for examples.

Previous work (Fan et al., 2019) used TF-IDF to find sentences in the web search that have the largest overlap with the question and created a *TF-IDF extraction* of about 850 words as input for Seq2Seq models. Instead, we construct a local knowledge graph for each question from the 100 web search hits. Following the average length of the TF-IDF support document constructed in (Fan et al., 2019), we experiment with modeling the first $N = 850$ tokens of the linearized graph, then scale to encode the entire graph using E-MCA.

**WikiSum** Second, we experiment on the *WikiSum CommonCrawl* (Liu et al., 2018b) summarization dataset[4] with 1.5 million examples. This task formulates Wikipedia lead paragraph generation as a multi-document summarization problem, where the paragraph must be generated using the cited article references and other queried content from web search. The query used is the title of the Wikipedia article. See Appendix for examples.

Previous work (Liu et al., 2018b) applied *TF-IDF Ranking* to order the paragraphs of web search given a query. Models receive the reordered paragraphs ranked by TF-IDF as input. Liu et al. (2018b) model the first $N = 500$ words of this re-ranking and then $N = 11,000$ using

[3] In (Liu et al., 2018b), the mechanism is termed *DMCA* as it is applied on the decoder side

[4] https://github.com/tensorflow/tensor2tensor/tree/master/tensor2tensor/data_generators/wikisum

MCA. We construct the knowledge graph for each Wikipedia article from the first 200K words of the ranked web search results[5], and experiment with encoding 500 and 11,000 tokens.

**Evaluation** Both tasks evaluate the multi-sentence generation output against the gold output using F1 ROUGE. On WikiSum, we report only ROUGE-L following (Liu et al., 2018b).

We conduct a comparative human evaluation on the ELI5 dataset. We use crowdworkers to examine the responses of two models on 300 different questions from the test set. For each question, 3 evaluators are shown two answers and asked to choose the one they prefer. To reduce variance, answers are standardized at 150 words each.

### 5.2 Training and Generation

To reduce the vocabulary size of varied web document content, we apply byte-pair encoding (Sennrich et al., 2016) to generate 40K codes for each dataset. We implement our models in fairseq-py (Ott et al., 2019) using the Transformer Big architecture and training schedule described in (Vaswani et al., 2017). Detailed parameters are listed in the Appendix. For generation, we use beam search with beam size 5 and tune a minimum and maximum length on the validation set.

### 5.3 Baselines

We compare our results to the Transformer sequence models presented in (Fan et al., 2019) for ELI5 and (Liu et al., 2018b) for WikiSum.

We evaluate three additional baseline models:

- *Sentence Selection with Maximal Marginal Relevance:* (Fan et al., 2019) used TF-IDF to identify relevant sentences in the web documents to form a support document of around 850 words. However, recent work (Fabbri et al., 2019) has shown that using maximal marginal relevance is an effective strategy for selecting relevant information while reducing redundancy. We explore using MMR to select sentences from the web text to concatenate to form a support document.
- *Seq2Seq Multi-task Triples:* To examine the impact of solely restructuring the input into Open IE Triples but not leveraging graph construction to reduce redundancy, we experiment with a Triples Only baseline. The triples are concatenated to form the input.
- *Seq2Seq Multi-task Top 20 Triples:* As an alternative to using graph construction to compress the full set of Open IE Triples, we explore using TF-IDF overlap with the query to find the most relevant information. We select the top 20 triples to concatenate as input.

[5]Average length of provided web input is around 50K words, and maximum length is around 900K words

| Model | Input Length | ROUGE 1 | 2 | L |
|---|---|---|---|---|
| Q + D to A*, TF-IDF | avg 850 | 28.3 | 5.1 | 22.8 |
| Q + D to A, MMR | avg 850 | 28.1 | 5.0 | 22.9 |
| Multi-task* | avg 850 | 28.9 | 5.4 | 23.1 |
| Multi-task Triples | 850 | 29.0 | 5.2 | 23.2 |
| Multi-task Top20 Trip. | avg 570 | 28.8 | 5.3 | 23.2 |
| Q + D to A Graph | 850 | 28.8 | 5.3 | 23.3 |
| Multi-task Graph | 850 | 29.5 | 5.6 | 23.6 |
| + Top-100 Attention | 850 | 29.7 | 5.7 | 23.8 |
| + E-MCA | 11K | 30.0 | 5.8 | 24.0 |

Table 1: **Answer Generation on ELI5** using Seq2Seq models receiving the **Q**uestion and a support **D**ocument (e.g. TF-IDF selection, Triples, Linearized Graph) to produce the **A**nswer. * denotes results from (Fan et al., 2019).

| Model | InputLen | ROUGE-L |
|---|---|---|
| T + D to P* | 500 | 34.2 |
| LM + D-MCA* | 11K | 36.2 |
| T + D to P | 500 | 33.8 |
| Multi-task | 500 | 34.4 |
| Multi-task Graph | 500 | 34.9 |
| + Top-100 Attention | 500 | 35.2 |
| + E-MCA | 11K | 36.5 |
| LM + D-MCA + MoE-256* | 7.5K | 38.8 |

Table 2: **Lead Paragraph Generation on WikiSum CommonCrawl** using Seq2Seq models receiving the **T**itle and a support **D**ocument (e.g. TF-IDF ranking, Linearized Graph) to produce the Lead **P**aragraph. * denotes results from (Liu et al., 2018b) that use data scraped from unrestricted web search, not the static CommonCrawl version.

## 6 Results

We examine the performance of our proposed approach and the choices made in graph construction and modeling. We analyze the quality of the compression created by graph construction and the robustness and interpretability of this process.

### 6.1 Linearized Graph Improves Performance

In Table 1, we compare our methods to various baselines on the ELI5 dataset. Using MMR to select the most relevant non-redundant input is similar to the TF-IDF baseline from Fan et al. (2019).

The Seq2Seq Multi-task Triples baseline standardizes the input by forming triples but does not remove redundant triples. It produces marginally better results compared to the baseline Multi-Task model. Sub-selecting to the Top 20 Triples is harmful, as similar text has high TF-IDF overlap with the query so redundant information is selected. Creating the graph structure brings an improvement of around 0.6 ROUGE-1.

Similar trends are seen for the WikiSum dataset in Table 2, where graph construction improves the Multi-task model by 0.5 ROUGE-1. These improvements are statistically significant at the 95% confidence level.

For both datasets, a further improvement is seen by using the hierarchical attention mechanism to attend to only the most relevant information in the linearized graph input. For ELI5, it brings an additional 0.2 ROUGE-1 improvement and on WikiSum a 0.3 ROUGE-1 improvement.

By using MCA to scale Seq2Seq models to encode the entire graph, further gains can be seen. Particularly in information synthesis tasks, prior work has shown the importance of reading more information. Liu et al. (2018b) achieved a 2-point ROUGE improvement by reading 11K tokens instead of 500. In our setting, E-MCA improves our results around 0.3 ROUGE on ELI5 and 1.3 ROUGE on WikiSum. We display random generations from both datasets in the Appendix.

We use human evaluation to compare the Multi-task baseline to the Multi-task Graph + Top-$k$ Attention model. **57.4%** of evaluations prefer the Multi-task Graph + Top-$k$ Attention model. We conduct a two-tailed binomial test and find this result is statistically significant with $p = 0.003$.

### 6.2 Analysis of Modeling Choices

**Ablation on Model Components** Table 3(a) sequentially removes the graph embeddings, the knowledge-base completion multi-tasking, and the multi-tasking from (Fan et al., 2019) and reveals that each of these is important for performance.

**Graph Attribute Embeddings** Table 3(b) displays the effect of removing the graph attribute embeddings and gating mechanism. Removing each is slightly harmful, and the combination of all three together provide the best performance.

**More Web Search Documents** Figure 7 (right) shows that graph construction with more web search information is important for answer token coverage. The graph on the top search hit alone is missing 64% of the answer tokens, but this decreases as more search hits are used. Table 3(c) indicates that this lack of coverage of the answer tokens correlates with generation quality. Models receiving a graph built on the first 5 search hits alone are substantially worse than all 100 hits.

**Top-$k$ Attention** Table 3(d) shows the effect of the Top-$k$ Hierarchical Attention mechanism for various values of $k$. Attending to too many tokens lowers ROUGE — for the ELI5 task of writing approximately 130 word answers, attending to 1000 input tokens likely means the model is focusing on irrelevant information and 50 tokens is too few.

| Model | ROUGE-1 |
|---|---|
| **(a) Iterative Removal of Model Components** | |
| Multi-task Graph | 29.4 |
| - Graph Embeddings | 29.1 |
| - KB Completion Multi-tasking | 28.9 |
| - LM Multi-tasking from (Fan et al., 2019) | 28.4 |
| **(b) Removing Graph Embedding Components** | |
| Graph | |
| + Gated Graph Weight + Query Relevance | 28.6 |
| No Graph Weight Embedding | 28.4 |
| No Query Relevance Embedding | 28.3 |
| No Gating | 28.4 |
| **(c) Varying Number of Hits in Graph** | |
| Multi-task Graph + Top-$k$ Attention + E-MCA | |
| with Graph on 5 Search Hits | 28.8 |
| with Graph on 10 Search Hits | 29.3 |
| with Graph on 50 Search Hits | 29.6 |
| with Graph on 100 Search Hits | 29.9 |
| **(d) Varying $k$ in Hierarchical Top-$k$ Atttention** | |
| Multi-task Graph + E-MCA + | |
| Top-$k = 50$ | 29.1 |
| Top-$k = 100$ | 29.5 |
| Top-$k = 250$ | 29.4 |
| Top-$k = 500$ | 29.3 |
| Top-$k = 1000$ | 29.2 |

Table 3: **Ablations** on the ELI5 Validation Set

| Model | Input | ROUGE-1 |
|---|---|---|
| Seq2Seq Q + D to A | TF-IDF | 28.3 |
| Seq2Seq Q + D to A | Web | 25.9 |
| + Shuffle | Web Shuffle | 24.4 |
| Seq2Seq Q + D to A Graph | Web | 28.5 |
| + Shuffle | Web Shuffle | 28.2 |

Table 4: Importance of **Web Search Relevance** on Validation for ELI5, modeling 850 input words.

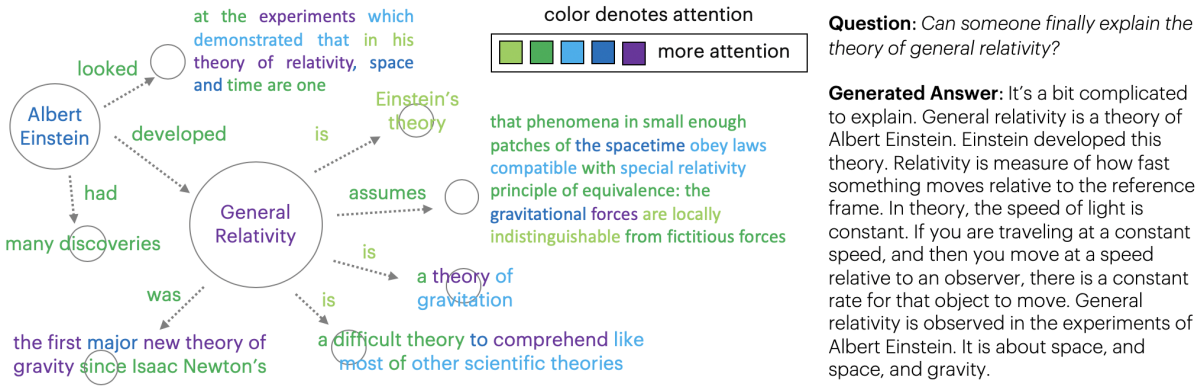


Figure 5: **Interpretable Attention** of Seq2Seq models on a subgraph when answering a question in ELI5

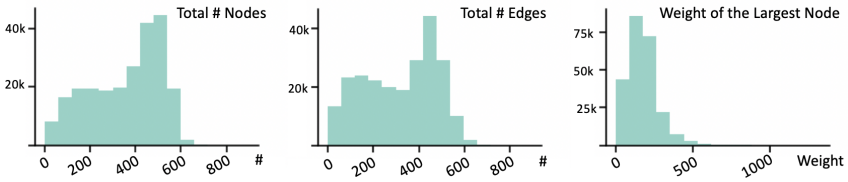


Figure 6: (Left) Distribution of Number of **Nodes**, (Middle) Number of **Edges**, (Right) **Weight** of the Largest Node in graph construction on the ELI5 training set.

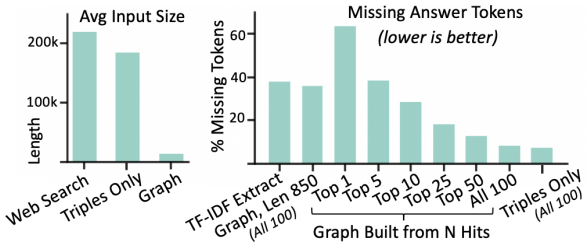


Figure 7: (Left) Graph construction drastically reduces input size by an order of magnitude. (Right) Graph construction encodes more tokens present in the answer compared to TF-IDF extraction and building the graph from more search hits increases answer token coverage. Analysis on ELI5 for both plots.

### 6.3 Graph Improves Answer Token Coverage Despite Compression

Figure 6 displays the distribution of the number of nodes, edges, and the largest node weight for each local graph built on the ELI5 dataset. The 100 web search results are compressed to a few hundred nodes. By merging redundancy and trimming irrelevant triples from the graph, the input is reduced by an order of magnitude (Figure 7, left).

Despite compression, the graph retains more answer tokens than TF-IDF subselection. Figure 7 (right) displays the percentage of answer tokens not present in the input. The TF-IDF Extraction from (Fan et al., 2019) is missing 38% of tokens. The graph constructed on all 100 web search results is only missing 8.7% of tokens, but has around 10K words. When analyzing just the first 850 tokens to match the average length of the TF-IDF extraction, the graph is better (only missing 35% of tokens). Further, the merging and discarding operations done during graph construction do not have a large effect on answer token coverage — the full set of triples marginally reduces the percentage of answer tokens missing to 7.3% instead of 8.7%. This indicates that much of the information in the full set triples is redundant and unnecessary for good answer token coverage.

### 6.4 Graph Representation is More Robust to Poor Search Relevance Ordering

We analyze the robustness of our approach to the ordering of web search results in Table 4. Instead of constructing the graph from the first web search result to the last, we shuffle the web search results and construct the graph on this shuffled input. We compare this to modeling the web search results directly (no TF-IDF retrieval) and a model that receives this shuffled web search input. The graph is more robust to shuffling — as more information can be encoded in the graph due to its compression effect, the search hit ordering is less critical.

### 6.5 Interpretable Attention on Subgraphs

Figure 5 shows an example of the nodes and edges the model focuses upon most when answering a question on ELI5. To construct this visualization, we calculate the top nodes the model attends to and then their top edges. The model attention on a sub-portion of the linearized input can be visualized as an interpretable graph that corresponds well to the model's generated answer. For example, the relationship *General Relativity* $\xrightarrow[\text{is}]{}$ *Einstein's theory* becomes the generated sentence *General Relativity is a theory of Albert Einstein*.

## 7 Conclusion

Many open-domain NLP tasks rely upon multi-document input from the web to facilitate tasks such as answering questions or writing summaries, but current approaches struggle to encode the entirely of this information. We propose constructing one knowledge graph per query and show that this method compresses information and reduces redundancy. We show on two abstractive generation tasks that using the linearized graph achieves better performance than TF-IDF retrieval.

## References


Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer.

Junwei Bao, Nan Duan, Ming Zhou, and Tiejun Zhao. 2014. Knowledge-based question answering as machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 967–976.

Regina Barzilay, Kathleen R McKeown, and Michael Elhadad. 1999. Information fusion in the context of multi-document summarization. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics*.

Tal Baumel, Matan Eyal, and Michael Elhadad. 2018. Query focused abstractive summarization: Incorporating query relevance, multi-document coverage, and summary length constraints into seq2seq models. *arXiv preprint arXiv:1801.07704*.

Daniel Beck, Gholamreza Haffari, and Trevor Cohn. 2018. Graph-to-sequence learning using gated graph neural networks. *arXiv preprint arXiv:1806.09835*.

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 481–490. Association for Computational Linguistics.

Lidong Bing, Piji Li, Yi Liao, Wai Lam, Weiwei Guo, and Rebecca J Passonneau. 2015. Abstractive multi-document summarization via phrase selection and merging. *arXiv preprint arXiv:1506.01597*.

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2013. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*.

Jürgen Bückner, Martin Pahl, O Stahlhut, and C-E Liedtke. 2002. A knowledge-based system for context dependent evaluation of remote sensing data. In *Joint Pattern Recognition Symposium*, pages 58–65. Springer.

Ziqiang Cao, Wenjie Li, Sujian Li, and Furu Wei. 2017. Improving multi-document summarization via text classification. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *ACL*.

Danqi Chen, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2013. Learning new facts from knowledge bases with neural tensor networks and semantic word vectors. *arXiv preprint arXiv:1301.3618*.

Christopher Clark and Matt Gardner. 2017. Simple and effective multi-paragraph reading comprehension. *arXiv preprint arXiv:1710.10723*.

Kevin Clark and Christopher D Manning. 2016a. Deep reinforcement learning for mention-ranking coreference models. *arXiv preprint arXiv:1609.08667*.

Kevin Clark and Christopher D Manning. 2016b. Improving coreference resolution by learning entity-level distributed representations. *arXiv preprint arXiv:1606.01323*.

Wanyun Cui, Yanghua Xiao, Haixun Wang, Yangqiu Song, Seung-won Hwang, and Wei Wang. 2017. Kbqa: learning question answering over qa corpora and knowledge bases. *Proceedings of the VLDB Endowment*, 10(5):565–576.

Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, and Andrew McCallum. 2018a. Multi-step retriever-reader interaction for scalable open-domain question answering.

Rajarshi Das, Tsendsuren Munkhdalai, Xingdi Yuan, Adam Trischler, and Andrew McCallum. 2018b. Building dynamic knowledge graphs from text using machine reading comprehension. *arXiv preprint arXiv:1810.05682*.

Hal Daumé III and Daniel Marcu. 2002. A noisy-channel model for document compression. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 449–456. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Giuseppe Di Fabbrizio, Amanda Stent, and Robert Gaizauskas. 2014. A hybrid approach to multi-document summarization of opinions in reviews. In *Proceedings of the 8th International Natural Language Generation Conference (INLG)*, pages 54–63.

Alexander R Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir R Radev. 2019. Multi-news: a large-scale multi-document summarization dataset and abstractive hierarchical model. *arXiv preprint arXiv:1906.01749*.

Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. Eli5: Long form question answering. In *Proceedings of ACL 2019*.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. *arXiv preprint arXiv:1805.04833*.

Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. 2010. Opinosis: A graph based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 340–348.

Dan Gillick and Benoit Favre. 2009. A scalable global model for summarization. In *Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing*, pages 10–18. Association for Computational Linguistics.

Mohamad Ali Honarpisheh, Gholamreza Ghassem-Sani, and Ghassem Mirroshandel. 2008. A multi-document multi-lingual automatic summarization system. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-II*.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Timothée Lacroix, Nicolas Usunier, and Guillaume Obozinski. 2018. Canonical tensor decomposition for knowledge base completion. *arXiv preprint arXiv:1806.07297*.

Yuxuan Lai, Yansong Feng, Xiaohan Yu, Zheng Wang, Kun Xu, and Dongyan Zhao. 2019. Lattice cnns for matching based chinese question answering. *arXiv preprint arXiv:1902.09087*.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A Smith. 2018a. Toward abstractive summarization using semantic representations. *arXiv preprint arXiv:1805.10399*.

Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018b. Generating wikipedia by summarizing long sequences. In *ICLR*.

Vincent Ng and Claire Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 104–111. Association for Computational Linguistics.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Marco Rospocher, Marieke van Erp, Piek Vossen, Antske Fokkens, Itziar Aldabe, German Rigau, Aitor Soroa, Thomas Ploeger, and Tessel Bogaard. 2016. Building event-centric knowledge graphs from news. *Journal of Web Semantics*, 37:132–151.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *ICLR*.

Gabriel Stanovsky, Julian Michael, Luke Zettlemoyer, and Ido Dagan. 2018. Supervised open information extraction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 885–895.

Sainbayar Sukhbaatar, Edouard Grave, Piotr Bojanowski, and Armand Joulin. 2019. Adaptive attention span in transformers.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William W Cohen. 2018. Open domain question answering using early fusion of knowledge bases and text. *arXiv preprint arXiv:1809.00782*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *NIPS*.

Piek Vossen, Tommaso Caselli, and Yiota Kontzopoulou. 2015. Storylines for structuring massive streams of news. In *Proceedings of the First Workshop on Computing News Storylines*, pages 40–49.

Rachel Wities, Vered Shwartz, Gabriel Stanovsky, Meni Adler, Ori Shapira, Shyam Upadhyay, Dan Roth, Eugenio Martínez-Cámara, Iryna Gurevych, and Ido Dagan. 2017. A consolidated open knowledge representation for multiple texts. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, pages 12–24.

Kun Xu, Lingfei Wu, Zhiguo Wang, Yansong Feng, Michael Witbrock, and Vadim Sheinin. 2018. Graph2seq: Graph to sequence learning with attention-based neural networks. *arXiv preprint arXiv:1804.00823*.

Michihiro Yasunaga, Rui Zhang, Kshitijh Meelu, Ayush Pareek, Krishnan Srinivasan, and Dragomir Radev. 2017. Graph-based neural multi-document summarization. *arXiv preprint arXiv:1706.06681*.

Zhiping Zheng. 2003. Question answering using web news as knowledge base. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 2*, pages 251–254. Association for Computational Linguistics.

Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. 2018. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*.

# Fine-grained Knowledge Fusion for Sequence Labeling Domain Adaptation

**Huiyun Yang**[1,2] **Shujian Huang**[1,2] **Xinyu Dai**[1,2] **Jiajun Chen**[1,2]
National Key Laboratory for Novel Software Technology, Nanjing, China[1]
Nanjing University, Nanjing, China[2]
yanghy@smail.nju.edu.cn
{huangsj, daixinyu, chenjj}@nju.edu.cn

## Abstract

In sequence labeling, previous domain adaptation methods focus on the adaptation from the source domain to the entire target domain without considering the diversity of individual target domain samples, which may lead to negative transfer results for certain samples. Besides, an important characteristic of sequence labeling tasks is that different elements within a given sample may also have diverse domain relevance, which requires further consideration. To take the multi-level domain relevance discrepancy into account, in this paper, we propose a fine-grained knowledge fusion model with the domain relevance modeling scheme to control the balance between learning from the target domain data and learning from the source domain model. Experiments on three sequence labeling tasks show that our fine-grained knowledge fusion model outperforms strong baselines and other state-of-the-art sequence labeling domain adaptation methods. [1]

## 1 Introduction

Sequence labeling tasks, such as Chinese word segmentation (CWS), POS tagging (POS) and named entity recognition (NER), are fundamental tasks in natural language processing. Recently, with the development of deep learning, neural sequence labeling approaches have achieved pretty high accuracy (Chen et al., 2017; Zhang and Yang, 2018), relying on large-scale annotated corpora. However, most of the standard annotated corpora belong to the news domain, and models trained on these corpora will get sharp declines in performance when applied to other domains like social media, forum, literature or patents (Daume III, 2007; Blitzer et al., 2007), which limits their application in the real world. Domain adaptation aims to exploit the abundant information of well-studied source domains to improve the performance in target domains (Pan and Yang, 2010), which is suitable to handle this issue. Following Daume III (2007), we focus on the supervised domain adaptation setting, which utilizes large-scale annotated data from the source domain and small-scale annotated data from the target domain.

| Types | Cases |
|---|---|
| Strongly | *Ops* **Steve Jobs resigned as CEO of Apple. Share prices are rising** *soooo* **fast**! |
| Weakly | *Alas* **as time goes by**, *hair's gone. Rock to 204 Section* **next week**! |

Table 1: Tweets from the social media domain have different degrees of relevance to the source domain (news). Within each case, the bold part is strongly relevant and the italic part is weakly relevant.

For sequence labeling tasks, each sample is usually a sentence, which consists of a sequence of words/Chinese characters, denoted as the element. We notice an interesting phenomenon: different target domain samples may have varying degrees of domain relevance to the source domain. As depicted in Table 1, there are some tweets similar to the news domain (i.e. strongly relevant). But there are also some tweets of their own style, which only appear in the social media domain (i.e. weakly relevant). The phenomenon can be more complicated for the cases where the whole sample is strongly relevant while contains some target domain specific elements, or vice versa, showing the diversity of relevance at the element-level. In the rest of this paper, we use 'domain relevance' to refer to the domain relevance to the source domain, unless specified otherwise.

Conventional neural sequence labeling domain adaptation methods (Liu and Zhang, 2012; Liu et al., 2014; Zhang et al., 2014; Chen et al., 2017; Peng and Dredze, 2017; Lin and Lu, 2018) mainly

[1]Our code is available at https://github.com/yhy1117/FGKF-DA.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4197–4206, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

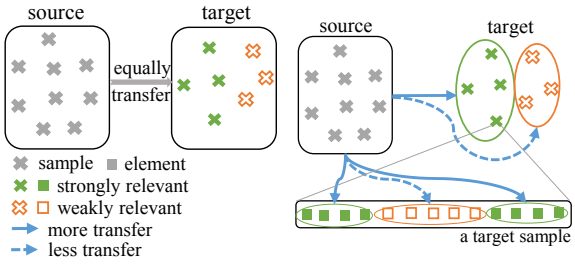


(a) Previous methods (b) Our method

Figure 1: Previous methods transfer knowledge by the whole sample set, while our method consider diverse domain relevance within the target domain set and within every target sample to transfer knowledge respectively.

focus on reducing the discrepancy between the sets of source domain samples and target domain samples. However, they neglect the diverse domain relevance of individual target domain samples, let alone the element-level domain relevance. As depicted in Figure 1, obviously, strongly relevant samples/elements should learn more knowledge from the source domain, while weakly relevant samples/elements should learn less and keep their characteristics.

In this paper, we propose a fine-grained knowledge fusion model to control the balance between learning from the target domain data and learning from the source model, inspired by the knowledge distillation method (Bucila et al., 2006; Hinton et al., 2015). With both the sample-level and element-level domain relevance modeling and incorporating, the fine-grained knowledge fusion model can alleviate the *negative transfer* (Rosenstein et al., 2005) in sequence labeling domain adaptation.

We verify the effectiveness of our method on six domain adaptation experiments of three different tasks, i.e. CWS, POS and NER, in two different languages, i.e. Chinese and English, respectively. Experiments show that our method achieves better results than previous state-of-the-art methods on all tasks. We also provide detailed analyses to study the knowledge fusion process.

Contributions of our work are summarized as follows:

- We propose a fine-grained knowledge fusion model to balance the learning from the target data and learning from the source model.
- We also propose multi-level relevance modeling schemes to model both the sample-level and element-level domain relevance.
- Empirical evidences and analyses are provided on three different tasks in two different languages, which verify the effectiveness of our method.

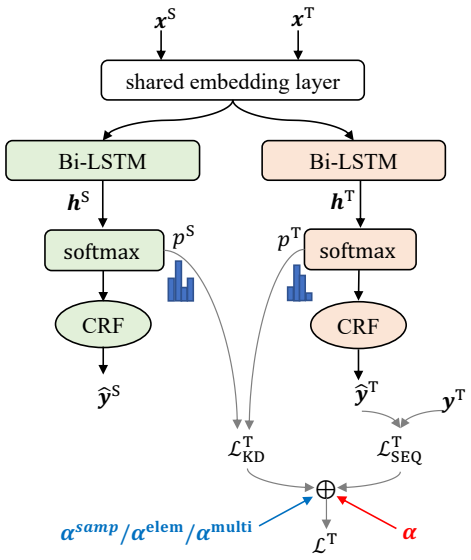


Figure 2: The architecture of basicKD (with the red $\alpha$, see §2) or fine-grained knowledge fusion model (with the blue $\alpha$, see §4), where the green part belongs to the source model, the orange part belongs to the target model and the white part is common. Better viewed in color.

## 2 Knowledge Distillation for Adaptation

Knowledge distillation (KD), which distills the knowledge from a sophisticated model to a simple model, has been employed in domain adaptation (Bao et al., 2017; Meng et al., 2018). Recently, online knowledge distillation(Furlanello et al., 2018; Zhou et al., 2018) is shown to be more effective, which shares lower layers between the two models and trains them simultaneously.

For sequence labeling domain adaptation, we utilize the online knowledge distillation method to distill knowledge from the source model to improve the target model, denoted as basicKD, which is depicted in Figure 2. We use the Bi-LSTM-CRF architecture (Huang et al., 2015), for both the source model and the target model, and share the embedding layer between them.

**Notations** For the rest of the paper, we use the superscript $S$ and $T$ to denote the source domain and the target domain, respectively. Source domain data is a set of $m$ samples with gold la-

bel sequences, denoted as $(\mathbf{x}_j^S, \mathbf{y}_j^S)_{j=1}^m$. Similarly, target domain data has $n$ samples, denoted as $(\mathbf{x}_i^T, \mathbf{y}_i^T)_{i=1}^n$, where $n \ll m$.

The training loss of the source model is the cross entropy between the predicted label distribution $\hat{\mathbf{y}}$ and the gold label $\mathbf{y}$:

$$\mathcal{L}^S = -\frac{1}{m}\sum_{j=1}^{m} \mathbf{y}_j^S \log \hat{\mathbf{y}}_j^S \tag{1}$$

The training loss of the target model is composed of two parts, namely the sequence labeling loss $\mathcal{L}_{\text{SEQ}}^T$ and the knowledge distillation loss $\mathcal{L}_{\text{KD}}^T$:

$$\mathcal{L}^T = (1-\alpha)\mathcal{L}_{\text{SEQ}}^T + \alpha\mathcal{L}_{\text{KD}}^T \tag{2}$$

$$\mathcal{L}_{\text{SEQ}}^T = -\frac{1}{n}\sum_{i=1}^{n} \mathbf{y}_i^T \log \hat{\mathbf{y}}_i^T \tag{3}$$

$$\mathcal{L}_{\text{KD}}^T = -\frac{1}{n}\sum_{i=1}^{n} \mathbf{p}_i^S \log \mathbf{p}_i^T \tag{4}$$

where $\mathcal{L}_{\text{SEQ}}^T$ is similar to $\mathcal{L}^S$, while $\mathcal{L}_{\text{KD}}^T$ is the cross entropy between the probability distributions predicted by the source model and the target model. $\alpha$ is a hyper-parameter scalar, which is used to balance the learning from the target domain data and the learning from the source model.

## 3 Relevance Modeling

BasicKD provides individual learning goals for every sample and element of the target domain, using a scalar $\alpha$ to weight. As a result, the source model has the same influence on all target samples, in which the diversity of domain relevance is neglected.

Here we present methods to model the domain relevance of target samples and elements, which could then be used to guide the knowledge fusion process (see §4). The overall architecture is shown in Figure 3. The relevance of each sample is a scalar, denoted as the sample-level relevance weight, $\mathbf{w}_i^{\text{samp}}$ for the $i^{th}$ sample, which can be obtained by the sample-level domain classification. The relevance of each element is also a scalar, while the relevance weights of all elements within a sample form a weight vector $\mathbf{w}^{\text{elem}}$, which can be obtained by the similarity calculation.

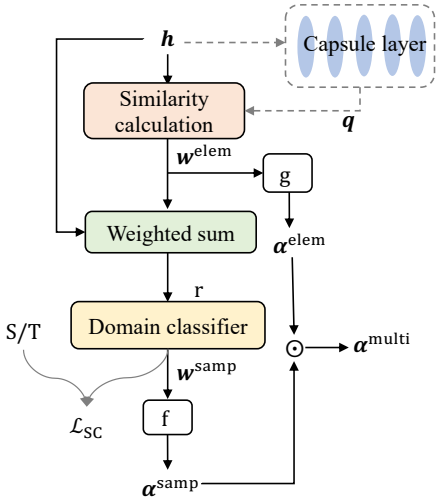


Figure 3: The relevance modeling process (see §3), where the block f denotes Eq.(10) and the block g denotes Eq.(14).

### 3.1 Element-level Relevance

To acquire the element-level relevance, we employ the domain representation $\boldsymbol{q} \in \mathbb{R}^{2d_h}$ ($d_h$ is the dimension of the Bi-LSTM) and calculate the similarity between the element representation and the domain representation. We incorporate two methods to get $\boldsymbol{q}$: (1) Domain-q: $\boldsymbol{q}$ is a trainable domain specific vector, where every element within a domain share the same $\boldsymbol{q}$; (2) Sample-q: $\boldsymbol{q}$ is the domain relevant feature extracted from each sample, where every element within a sample share the same $\boldsymbol{q}$. Because of the superiority of the capsule network modeling abstract features (Gong et al., 2018; Yang et al., 2018), we use it to capture the domain relevant features within a sample. We incorporate the same bottom-up aggregation process as Gong et al. (2018) and the encoded vector is regarded as $\boldsymbol{q}$:

$$\mathbf{q} = \text{Capsule}(\mathbf{h}) \tag{5}$$

where $\mathbf{h}$ is the hidden state matrix of a sample.

The similarity calculation formula is the matrix dot [2]:

$$\mathbf{w}_j^{\text{elem}} = \mathbf{q}^\top \mathbf{B}\mathbf{h}_j \tag{6}$$

where $\boldsymbol{h}_j$ is the hidden states of the $j^{\text{th}}$ element and $\mathbf{w}_j^{\text{elem}}$ is the relevance weight of it. $\mathbf{B} \in \mathbb{R}^{2d_h \times 2d_h}$ is a trainable matrix.

[2]We also try dot and MLP, while matrix dot get better performance with fewer parameters.

### 3.2 Sample-level Relevance

To acquire the sample-level domain relevance, we make use of the domain label to carry out sample-level text classification (two class, source domain or target domain). The weight $\mathbf{w}^{\text{elem}}$ is normalized across the sample length using the softmax function, then the sample representation can be obtained by the weighted sum of hidden states. The process can be expressed as:

$$\hat{\mathbf{w}}_j^{\text{elem}} = \frac{\exp(\mathbf{w}_j^{\text{elem}})}{\sum_k \exp(\mathbf{w}_k^{\text{elem}})} \tag{7}$$

$$\mathbf{r} = \sum_{j=1}^{L} \hat{\mathbf{w}}_j{}^{\text{elem}} \cdot \mathbf{h}_j \tag{8}$$

$\mathbf{r} \in \mathbb{R}^{2d_h}$ is the sample representation and $L$ is the sample length.

Once the sample representation is obtained, the multi-layer perceptron (MLP) and softmax do sample classification next:

$$[w^{\text{samp}}, 1 - w^{\text{samp}}] = [\text{softmax}(\text{MLP}(\mathbf{r}))]^\top \tag{9}$$

where $w^{\text{samp}}$ is the sample relevance weight.

## 4 Fine-grained Knowledge Fusion for Adaptation

With the relevance modeling, the fine-grained knowledge fusion model is proposed to fusion the knowledge from the source domain and the target domain at different levels. The overall architecture is shown in Figure 2.

### 4.1 Sample-level Knowledge Fusion

Different samples of target domain tend to show different domain relevance, and as a result, they need to acquire different amount of knowledge from the source domain. Different $\alpha$ is assigned to each target sample based on its domain relevance to achieve the sample-level knowledge fusion. The new $\alpha$ can be computed as:

$$\boldsymbol{\alpha}_i^{\text{samp}} = \sigma(\tau \cdot \mathbf{w}_i^{\text{samp}} + \gamma) \tag{10}$$

where $\boldsymbol{\alpha}_i^{\text{samp}}$ is the $\alpha$ of the $i^{\text{th}}$ sample and $\mathbf{w}_i^{\text{samp}}$ is the relevance weight of it; $\sigma$ denotes the sigmoid function; $\tau$ is temperature and $\gamma$ is bias.

The loss functions of the target model can be computed as:

$$\mathcal{L}^T = \mathcal{L}^T_{\text{SEQ}} + \mathcal{L}^T_{\text{KD}} \tag{11}$$

$$\mathcal{L}^T_{\text{SEQ}} = -\frac{1}{n}\sum_{i=1}^{n}(1 - \boldsymbol{\alpha}_i^{\text{samp}})\mathbf{y}_i^T \log \hat{\mathbf{y}}_i^T \tag{12}$$

$$\mathcal{L}^T_{\text{KD}} = -\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{\alpha}_i^{\text{samp}}\mathbf{p}_i^S \log \mathbf{p}_i^T \tag{13}$$

The sample classification losses of the source model $\mathcal{L}^S_{sc}$ and target model $\mathcal{L}^T_{sc}$ are both cross entropy.

### 4.2 Element-level Knowledge Fusion

Besides the sample-level domain relevance, different elements within a sample tend to present diverse domain relevance. In this method, we assign different $\alpha$ to each element based on its domain relevance weight to achieve the element-level knowledge fusion. The new $\alpha$ can be computed as:

$$\boldsymbol{\alpha}_i^{\text{elem}} = \sigma(\mathbf{W}_\alpha \mathbf{w}_i^{\text{elem}} + \mathbf{b}_\alpha)] \tag{14}$$

where $\boldsymbol{\alpha}_i^{\text{elem}} \in \mathbb{R}^L$ is a vector, in which $\boldsymbol{\alpha}_{ij}^{\text{elem}}$ denotes the $\alpha$ of the $j^{\text{th}}$ element in the $i^{\text{th}}$ sample. $\mathbf{w}_i^{\text{elem}}$ is the relevance weight of the $i^{\text{th}}$ sample. $\mathbf{W}_\alpha$ and $\mathbf{b}_\alpha$ are trainable parameters.

The loss functions of the target model can be expressed as:

$$\mathcal{L}^T_{\text{SEQ}} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{L}(1 - \boldsymbol{\alpha}_{ij}^{\text{elem}})\mathbf{y}_{ij}^T \log \hat{\mathbf{y}}_{ij}^T \tag{15}$$

$$\mathcal{L}^T_{\text{KD}} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{L}\boldsymbol{\alpha}_{ij}^{\text{elem}}\mathbf{p}_{ij}^S \log \mathbf{p}_{ij}^T \tag{16}$$

where $*_{ij}$ denotes the $*$ of the $j^{\text{th}}$ element in the $i^{\text{th}}$ sample, and the final loss function is the same with Eq.(11).

### 4.3 Multi-level Knowledge Fusion

In this method, we take both the sample-level and element-level relevance diversities into account to implement the multi-level knowledge fusion, and the multi-level $\alpha$ can be computed as:

$$\boldsymbol{\alpha}^{\text{multi}} = \boldsymbol{\alpha}^{\text{samp}} \odot \boldsymbol{\alpha}^{\text{elem}} \tag{17}$$

where $\odot$ denotes the element-wise product. $\boldsymbol{\alpha}^{\text{multi}} \in \mathbb{R}^{n \times L}$ is a matrix as well.

The loss functions of the target model can be obtained by replacing $\boldsymbol{\alpha}_{ij}^{\text{elem}}$ with $\boldsymbol{\alpha}_{ij}^{\text{multi}}$ in Eq.(15) and Eq.(16).

| Task | Language | Source | Target | Domain |
|---|---|---|---|---|
| CWS | Chinese | CTB6 (Xue et al., 2005) | Zhuxian (Zhang et al., 2014) | news → novels |
| POS | Chinese | CTB6 (Xue et al., 2005) | Weibo (Qiu et al., 2016) | news → social media |
| | English | PTB (Marcus et al., 1993) | Twitter (Ritter et al., 2011) | |
| NER | Chinese | MSRA (Levow, 2006) | WeiboNER (Peng and Dredze, 2015) | |
| | English | Ontonotes (Ralph et al., 2013) | Twitter (Ritter et al., 2011) | |

Table 2: Datasets used in this paper.

**Algorithm 1** Training Process of Knowledge Fusion

1. **Input**: source data, target data
2. **Hyper − parameters**: batch size $b$, teach step $I$
3. Initialize parameters of the source and target model
4. (optional) Use the source data to pre-train $\boldsymbol{\theta}^S$ and $\boldsymbol{\theta}^T$
5. **repeat**
6. **for** $i = 1$ to $I$ **do**
7. Sample $b$ samples from the source data
8. Compute $\mathcal{L}^S$, and update $\boldsymbol{\theta}^S$
9. Compute $\mathcal{L}^S_{sc}$, and update $\boldsymbol{\theta}^S$
10. **end for**
11. Use $\boldsymbol{\theta}^S$ to test $\mathbf{x}^T_{train}$ and get $\mathbf{p}^S$
12. **while** in an episode:
13. Sample $b$ samples from the target data
14. Use relevance modeling to get $\mathbf{w}^{\text{samp}}$,$\mathbf{w}^{\text{elem}}$
15. Compute $\boldsymbol{\alpha}^{\text{samp}}$/$\boldsymbol{\alpha}^{\text{elem}}$/$\boldsymbol{\alpha}^{\text{multi}}$ and $\mathcal{L}^T_{\text{SEQ}}$
16. Use $\boldsymbol{\theta}^T$ to predict $\mathbf{p}^T$, and compute $\mathcal{L}^T_{\text{KD}}$
17. Compute $\mathcal{L}^T$, and update $\boldsymbol{\theta}^T$
18. Compute $\mathcal{L}^T_{sc}$, and update $\boldsymbol{\theta}^T$
19. **end while**
20. **until** converge

### 4.4 Training Process

Both the source model and the target model can be pre-trained on the source domain data (warm up, optional). In the fine-grained knowledge fusion method, the source model and the target model are trained alternately. Within an episode, we use $I$ steps to train the source model ahead, then the soft target ($\mathbf{p}^S$) can be obtained and the target model will be trained. During the training of the target model, the parameters of the source model are fixed (gradient block). Every training step includes the sequence labeling training and the sample classification training. We conduct early stopping according to the performance of the target model. The whole training process is shown in Algorithm 1.

## 5 Experiments

### 5.1 Datasets

We conduct three sequence labeling tasks: CWS, POS and NER, and the latter two tasks containing both Chinese and English settings. Detailed datasets are shown in Table 2. There are two kinds of source-target domain pairs: news-novels and news-social media. To be consistent with the setting where there is only small-scale target domain data, we use 5% training data of Weibo for both CWS and POS. For the different NER tag sets, we only focus on three types of entities: Person (PER), Location (LOC) and Organization (ORG) and regard other types as Other (O).

### 5.2 Settings

For each task, hyper-parameters are set via grid search on the target domain development set. Embedding size and the dimension of LSTM hidden states is set to 100. Batch size is set to 64. Learning rate is set to 0.01. We employ the dropout strategy on the embedding and MLP layer with the rate of 0.2. The $l_2$ regularization term is set to 0.1. The gradient clip is set to 5. The teach step $I$ is set to 100. The routing iteration is set to 3 and the number of the output capsules is set to 60. The temperature $\tau$ is initialized to 1 and the probability bias $\gamma$ is initialized to 0.5. We set the $\alpha$ of the basicKD method to 0.5 according to Hinton et al. (2015). We randomly initialize the embedding matrix without using extra data to pre-train, unless specified otherwise.

### 5.3 Baselines

We implement several baseline methods, including: **source only** (training with only source domain data), **target only** (training with only target domain data) and **basicKD** (see §2).

We also re-implement state-of-the-art sequence labeling domain adaptation methods, following their settings except for unifying the embedding size and the dimension of LSTM hidden states:

- **Pre-trained methods: Pre-trained embedding** incorporates source domain data with its gold label to pre-train context-aware character embedding (Zhou et al., 2017), which is used to initialize the target model; **Pre-trained model** trains the model on the source domain and then finetune it on the target domain.

| Methods | CWS | | | | POS | | NER | |
|---|---|---|---|---|---|---|---|---|
| | Zhuxian | | 5% Weibo | | | | | |
| | F | $R_{OOV}$ | F | $R_{OOV}$ | zh | en | zh | en |
| Target only | 92.80 | 65.81 | 84.01 | 64.12 | 93.03 | 86.83 | 46.49 | 59.58 |
| BasicKD | 94.23 | 74.08 | 89.21 | 76.26 | 95.69 | 89.96 | 49.92 | 62.15 |
| Pre-trained embedding | 93.70 | 70.44 | 87.62 | 72.27 | 94.96 | 89.70 | 52.53 | 61.36 |
| Pre-trained model | 94.43 | 74.30 | 89.50 | 76.27 | 96.10 | 90.05 | 54.25 | 62.88 |
| Linear projection | 94.14 | 72.75 | 88.77 | 75.85 | 95.92 | 89.36 | 52.71 | 62.27 |
| Domain mask | 94.30 | 75.20 | 88.84 | 75.03 | 96.01 | 89.81 | 54.12 | 62.64 |
| NAL | 94.47 | 74.62 | 88.63 | 75.77 | 96.19 | 90.48 | 54.70 | 63.32 |
| AMCL | 94.62 | 74.46 | 89.42 | 76.16 | 94.13 | 89.12 | 51.47 | 61.57 |
| FGKF | 95.01 | 77.26 | 90.45 | 77.27 | **96.60** | 91.33 | 55.60 | 63.81 |
| + Pre-trained embedding | **95.09** | **77.56** | **90.73** | **77.87** | 96.36 | **91.66** | **57.57** | **65.51** |

Table 3: Results of domain adaptation on three tasks, where zh denotes the Weibo datasets (in Chinese), and en denotes the Twitter dataset (in English).

- **Projection methods: Linear projection** (Peng and Dredze, 2017) uses the domain-relevant matrix to transform the learned representation from different domains into the shared space; **Domain mask** (Peng and Dredze, 2017) masks the hidden states of Bi-LSTM to split the representations into private and public regions to do the projection; **Neural adaptation layer** (NAL) (Lin and Lu, 2018) incorporates adaptation layers at the input and output to conduct private-public-private projections.

- **Adversarial method: Adversarial multi-criteria learning** (AMCL) (Chen et al., 2017) uses the shared-private architecture with the adversarial strategy to learn the shared representations across domains.

### 5.4 Overall Results on CWS

We use the F1-score (F) and the recall of out-of-vocabulary words ($R_{oov}$) to evaluate the domain adaptation performance on CWS. We compare methods with different relevance modeling schemes and different levels of knowledge fusion, without warm up. And we denote our final model as **FGKF**, which is the multi-level knowledge fusion with the sample-q relevance modeling and warm up.

The results in Table 4 show that both the basicKD method and fine-grained methods achieve performance improvements through domain adaptation. Compared with the basicKD method, FGKF behaves better (+1.1% F and +2.8% $R_{oov}$ v.s. basicKD on average), as it takes multi-level relevance discrepancies into account. The sample-q method performs better than the domain-q method, which shows the domain feature is better represented at the sample level, not at the domain level. As for the granularity of $\alpha$, the performances of $\boldsymbol{\alpha}^{\text{elem}}$ is better than $\boldsymbol{\alpha}^{\text{samp}}$, showing the necessity of modeling element-level relevance. And there isn't a distinct margin between $\boldsymbol{\alpha}^{\text{elem}}$ and $\boldsymbol{\alpha}^{\text{multi}}$ as most of the multi-level domain relevance can be included by the element level. Results of FGKF with warm up indicate that starting from sub-optimal point is better than starting from scratch for the target model.

| Methods | Zhuxian | | 5% Weibo | |
|---|---|---|---|---|
| | F | $R_{OOV}$ | F | $R_{OOV}$ |
| Source only | 83.86 | 62.40 | 83.75 | 70.74 |
| Target only | 92.80 | 65.81 | 84.01 | 64.12 |
| BasicKD | 94.23 | 74.08 | 89.21 | 76.26 |
| Domain-q $\boldsymbol{\alpha}^{\text{samp}}$ | 94.55 | 74.02 | 89.63 | 75.93 |
| Domain-q $\boldsymbol{\alpha}^{\text{elem}}$ | 94.81 | 74.75 | 89.99 | **77.59** |
| Domain-q $\boldsymbol{\alpha}^{\text{multi}}$ | 94.75 | 74.96 | 90.06 | 77.25 |
| Sample-q $\boldsymbol{\alpha}^{\text{samp}}$ | 94.57 | 74.47 | 89.77 | 76.81 |
| Sample-q $\boldsymbol{\alpha}^{\text{elem}}$ | 94.78 | 74.52 | 90.07 | 76.94 |
| Sample-q $\boldsymbol{\alpha}^{\text{multi}}$ | 94.91 | 75.56 | 90.20 | 77.46 |
| FGKF | **95.01** | **77.26** | **90.45** | 77.27 |

Table 4: Results of baselines and fine-grained knowledge fusion methods on CWS.

Among related works (Table 3), AMCL and Pre-trained model methods have better performances in CWS. Compared with other methods, FGKF achieves the best results in both F and $R_{OOV}$. Results demonstrate the effectiveness of our fine-grained knowledge fusion architecture for domain adaptation, and also show the significance of considering sample-level and element-level relevance discrepancies.

### 5.5 Overall Results on POS and NER

To further verify the effectiveness of FGKF, we conduct experiments on POS and NER tasks, using F1-score as the evaluation criterion. Detailed results are shown in Table 3. In these tasks, FGKF

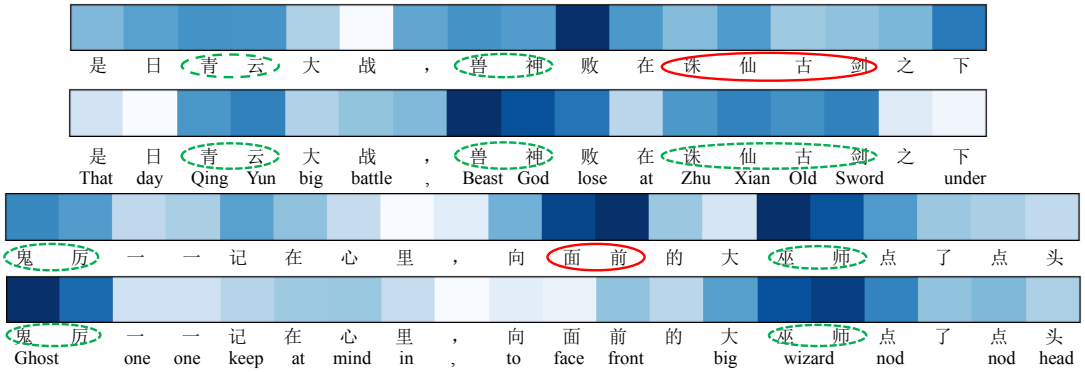


Figure 4: Two cases of the element-level relevance modeling visualization, where the upper one belongs to the domain-q method and the lower one belongs to the sample-q method. The green dotted circle indicates the correct domain relevant element and the red solid circle indicates the ignored or mistaken extracted element.

achieves better results than other adaptation methods. Extra gain could be obtained by using pre-trained embedding. These results also verify the generalization of our method over different tasks and languages.

## 6 Analysis

In this section, we will display and discuss the domain adaptation improvements provided by our fine-grained knowledge fusion method.

### 6.1 Performances of Elements with Different Relevance

To further probe into the experimental results of the fine-grained knowledge fusion, we classify the target test data (in element level) into two classes: strongly relevant and weakly relevant, based on their relevance degrees to the source domain. The partition threshold is according to the average relevance score of the target training data. Detailed results on Twitter are depicted in Table 5.

| Methods | POS | | NER | |
|---|---|---|---|---|
| | Strong | Weak | Strong | Weak |
| Source only | 87.47 | 82.48 | 68.27 | 46.30 |
| Target only | 86.46 | 87.41 | 62.01 | 56.29 |
| BasicKD | 91.92 | 83.82 | 70.20 | 52.63 |
| FGKF | **92.55** | **89.93** | **71.81** | **57.92** |

Table 5: Results of the strongly/weakly relevant elements on the Twitter test set.

It is reasonable that both the basicKD and FGKF enhance the performance of the strongly relevant part, while FGKF get larger improvements because it is able to enhance the knowledge fusion by learning more from the source model. For the weakly relevant part, the basicKD method damages the performance on it (from 87.41 to 83.82 for POS and from 56.29 to 52.63 for NER), which indicate the negative transfer. On the contrary, FGKF improves the performance of the weakly relevant part compared with the target only baseline with a large margin. It is shown that the fine-grained domain adaptation method can reduce the negative transfer on the weakly relevant part and contribute to the transfer on the strongly relevant one.

### 6.2 Relevance Weight Visualization

We carry out the visualization of the element-level relevance weight to illustrate the effects of the two relevance modeling schemes (domain-q and sample-q). Figure 4 exhibits two cases of element-level relevance modeling results, from which we can explicitly observe that the two schemes capture different domain relevance within a sample. In the first case, the sample-q method extracts more domain relevant elments, like “Qingyun”, “Beast God” and “Zhuxian Old Sword”, while the domain-q method ignores the last one. In the second case, the domain-q method extracts “front” incorrectly. These results indicate that the sample-q method can implement better relevance modeling than the domain-q method to some extent, and prove that the domain relevant feature is better represented at the sample level, not at the domain level.

### 6.3 Case Study

We take two samples in Twitter test set as examples to show how the element-level relevance affects the adaptation. Results in Table 6 show that both basicKD and FGKF can improve the performance of strongly relevant elements, e.g. “got (VBD)”, “Lovis (B-PER)”. However, only FGKF

| Tasks | POS | | | | | NER | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Sentence | I | got | u | next | week | Louis | interview | with | The | Sun |
| Source only | PN | VBD | NN | JJ | NN | B-PER | O | O | O | O |
| Target only | PN | VBZ | PN | JJ | NN | O | O | O | B-ORG | I-ORG |
| BasicKD | PN | VBD | NN | JJ | NN | B-PER | O | O | O | O |
| FGKF | PN | VBD | PN | JJ | NN | B-PER | O | O | B-ORG | I-ORG |

Table 6: Two cases of domain adaptation, where the underlined tags are wrong.

reduces the transfer of source domain errors, e.g. "u (NN)", "The (B-ORG) Sun (I-ORG)".

### 6.4 Ablation Study

We conduct the ablation study on Twitter dataset (Table 7). Results show the gradient block and the multi-level knowledge fusion are of vital importance to FGKF. The embedding sharing and warm up also make contributions.

| Methods | POS | | NER | |
|---|---|---|---|---|
| | F | Δ | F | Δ |
| FGKF | 91.33 | - | 63.81 | - |
| w/o share embedding | 90.75 | -0.58 | 62.47 | -1.34 |
| w/o gradient block | 88.48 | -2.85 | 58.83 | -4.98 |
| w/o $\alpha^{\text{samp}}$ | 90.94 | -0.39 | 63.52 | -0.30 |
| w/o $\alpha^{\text{elem}}$ | 90.23 | -1.10 | 62.43 | -1.38 |
| w/o $\alpha^{\text{multi}}$ | 90.12 | -1.21 | 62.32 | -1.49 |
| w/o warm up | 90.89 | -0.44 | 63.17 | -0.64 |

Table 7: Ablation results of the Twitter test set.

### 6.5 Influence of Target Data Size

Here we investigate the impact of the target domain data size on FGKF. As is depicted in Figure 5, when the size is small (20%), the gap is pretty huge between FGKF and basicKD, which verifies the significance of fine-grained knowledge fusion in the low-resource setting. Even with the size of target data increasing, there are still stable margins between the two methods.

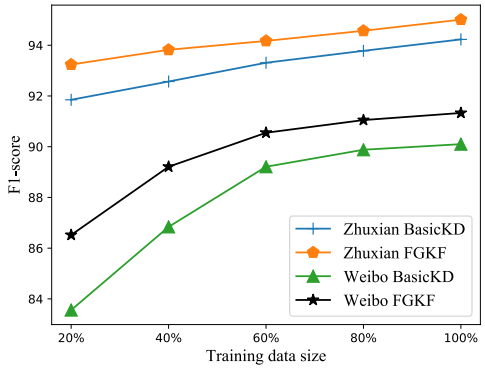


Figure 5: Results of CWS target test set with varying target training data size. Only 10% training data of Weibo is utilized.

## 7 Related Work

Besides the source domain data, some methods utilize the target domain lexicons (Liu et al., 2014; Zhang et al., 2014), unlabeled (Liu and Zhang, 2012) or partial-labeled target domain data (Liu et al., 2014) to boost the sequence labeling adaptation performance, which belong to unsupervised or semi-supervised domain adaptation. However, we focus on supervised sequence labeling domain adaptation, where huge improvement can be achieved by utilizing only small-scale annotated data from the target domain.

Previous works in domain adaptation often try to find a subset of source domain data to align with the target domain data (Chopra et al., 2013; Ruder and Plank, 2017) which realizes a kind of source data sample or construct a common feature space, while those methods may wash out informative characteristics of target domain samples. Instance-based domain adaptation (Jiang and Zhai, 2007; Zhang and Xiong, 2018) implement the source sample weighting by assigning higher weights to source domain samples which are more similar to the target domain. There are also some methods (Guo et al., 2018; Kim et al., 2017; Zeng et al., 2018) explicitly weighting multiple source domain models for target samples in multi-source domain adaptation. However, our work focuses on the supervised single source domain adaptation, which devote to implementing the knowledge fusion between the source domain and the target domain, not within multiple source domains. Moreover, considering the important characteristics of sequence labeling tasks, we put more attention to the finer-grained adaptation, considering the domain relevance in sample level and element level.

## 8 Conclusion

In this paper, we propose a fine-grained knowledge fusion model for sequence labeling domain adaptation to take the domain relevance diversity of target data into account. With the relevance modeling on both the sample level and element

level, the knowledge of the source model and target data can achieve multi-level fusion. Experimental results on different tasks demonstrate the effectiveness of our approach, and show the potential of our approach in a broader range of domain adaptation applications.

## Acknowledgements

We would like to thank the anonymous reviewers for their insightful comments. Shujian Huang is the corresponding author. This work is supported by National Science Foundation of China (No. U1836221, No. 61772261), National Key RD Program of China (No. 2019QY1806).

## References

Zuyi Bao, Si Li, Weiran Xu, and Sheng Gao. 2017. Neural regularized domain adaptation for Chinese word segmentation. In *AFNLP*.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*.

Cristian Bucila, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *SIGKDD*.

Xinchi Chen, Zhan Shi, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-criteria learning for Chinese word segmentation. In *ACL*.

Sumit Chopra, Suhrid Balakrishnan, and Raghuraman Gopalan. 2013. Dlid: Deep learning for domain adaptation by interpolating between domains. In *ICML*.

Hal Daume III. 2007. Frustratingly easy domain adaptation. In *ACL*.

Tommaso Furlanello, Zachary Chase Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. 2018. Born-again neural networks. In *ICML*.

Jingjing Gong, Xipeng Qiu, Shaojing Wang, and Xuanjing Huang. 2018. Information aggregation via dynamic routing for sequence encoding. In *COLING*.

Jiang Guo, Darsh Shah, and Regina Barzilay. 2018. Multi-source domain adaptation with mixture of experts. In *EMNLP*.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. In *NIPS*.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. arXiv:1508.01991. Version 1.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in nlp. In *ACL*.

Young-Bum Kim, Karl Stratos, and Dongchan Kim. 2017. Domain attention with an ensemble of experts. In *ACL*.

Gina-Anne Levow. 2006. The third international Chinese language processing bakeoff: Word segmentation and named entity recognition. In *AFNLP*.

Bill Yuchen Lin and Wei Lu. 2018. Neural adaptation layers for cross-domain named entity recognition. In *EMNLP*.

Yang Liu and Yue Zhang. 2012. Unsupervised domain adaptation for joint segmentation and pos-tagging. In *COLING*.

Yijia Liu, Yue Zhang, Wanxiang Che, Ting Liu, and Fan Wu. 2014. Domain adaptation for CRF-based Chinese word segmentation using free annotations. In *EMNLP*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The penn treebank. *Computational Linguistics*.

Zhong Meng, Jinyu Li, Yifan Gong, and Biing-Hwang Juang. 2018. Adversarial teacher-student learning for unsupervised domain adaptation. In *ICASSP*.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.*

Nanyun Peng and Mark Dredze. 2015. Named entity recognition for Chinese social media with jointly trained embeddings. In *EMNLP*.

Nanyun Peng and Mark Dredze. 2017. Multi-task domain adaptation for sequence tagging. In *Repl4NLP*.

Xipeng Qiu, Peng Qian, and Zhan Shi. 2016. Overview of the NLPCC-ICCPOL 2016 shared task: Chinese word segmentation for micro-blog texts. In *ICCPOL*.

Weischedel Ralph, Palmer Martha, and Marcus et al. Mitchell. 2013. Ontonotes release 5.0 ldc2013t19. Linguistic Data Consortium.

Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. 2011. Named entity recognition in tweets: An experimental study. In *EMNLP*.

Michael T Rosenstein, Zvika Marx, Leslie Pack Kaelbling, and Thomas G Dietterich. 2005. To transfer or not to transfer. *NIPS*.

Sebastian Ruder and Barbara Plank. 2017. Learning to select data for transfer learning with bayesian optimization. In *EMNLP*.

Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The Penn Chinese Treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering*.

Min Yang, Wei Zhao, Jianbo Ye, Zeyang Lei, Zhou Zhao, and Soufei Zhang. 2018. Investigating capsule networks with dynamic routing for text classification. In *EMNLP*.

Jiali Zeng, Jinsong Su, Huating Wen, Yang Liu, Jun Xie, Yongjing Yin, and Jianqiang Zhao. 2018. Multi-domain neural machine translation with word-level domain context discrimination. In *EMNLP*.

Meishan Zhang, Yue Zhang, Wanxiang Che, and Ting Liu. 2014. Type-supervised domain adaptation for joint segmentation and pos-tagging. In *ACL*.

Shiqi Zhang and Deyi Xiong. 2018. Sentence weighting for neural machine translation domain adaptation. In *COLING*.

Yue Zhang and Jie Yang. 2018. Chinese ner using lattice lstm. In *ACL*.

Guorui Zhou, Ying Fan, Runpeng Cui, Weijie Bian, Xiaoqiang Zhu, and Kun Gai. 2018. Rocket launching: A universal and efficient framework for training well-performing light net. In *AAAI*.

Hao Zhou, Zhenting Yu, Yue Zhang, Shujian Huang, XIN-YU DAI, and Jiajun Chen. 2017. Word-context character embeddings for Chinese word segmentation. In *EMNLP*.

# Exploiting Monolingual Data at Scale for Neural Machine Translation

**Lijun Wu**[1,*], **Yiren Wang**[2,*], **Yingce Xia**[3,†], **Tao Qin**[3], **Jianhuang Lai**[1], **Tie-Yan Liu**[3]
[1]School of Data and Computer Science, Sun Yat-sen University;
[2]University of Illinois at Urbana-Champaign;
[3]Microsoft Research Asia;
[1]{wulijun3@mail2, stsljh@mail}.sysu.edu.cn;
[2]yiren@illinois.edu;
[3]{Yingce.Xia, taoqin, tyliu}@microsoft.com

## Abstract

While target-side monolingual data has been proven to be very useful to improve neural machine translation (briefly, NMT) through back translation, source-side monolingual data is not well investigated. In this work, we study how to use both the source-side and target-side monolingual data for NMT, and propose an effective strategy leveraging both of them. First, we generate synthetic bitext by translating monolingual data from the two domains into the other domain using the models pre-trained on genuine bitext. Next, a model is trained on a noised version of the concatenated synthetic bitext where each source sequence is randomly corrupted. Finally, the model is fine-tuned on the genuine bitext and a clean version of a subset of the synthetic bitext without adding any noise. Our approach achieves state-of-the-art results on WMT16, WMT17, WMT18 English↔German translations and WMT19 German→French translations, which demonstrate the effectiveness of our method. We also conduct a comprehensive study on how each part in the pipeline works.

## 1 Introduction

Neural machine translation (briefly, NMT) (Bahdanau et al., 2014; Gehring et al., 2017; Wu et al., 2016; Vaswani et al., 2017) is well-known for its outstanding performance (Hassan et al., 2018), which usually relies on large-scale bitext for training. However, high quality bitext is always limited and costly to collect. In contrast, there exists large amount of monolingual data which can be leveraged to augment the training corpus. How to effectively leverage monolingual data is an important research topic for NMT and there are plenty of works studying this problem (Gulcehre et al., 2015, 2017; Sennrich et al., 2016a; He et al., 2016; Zhang and Zong, 2016; Zhang et al., 2018; Cheng et al., 2016; Wang et al., 2018; Chinea-Rios et al., 2017; Wu et al., 2018).

Among them, one of the most cited approach is back translation (briefly, BT) (Sennrich et al., 2016a), which leverages the target-side monolingual data. Specifically, a target-to-source translation model (trained on the genuine bitext) is used to translate target-side monolingual sentences into the source domain to generate a set of synthetic bitext, which is then used together with the genuine bitext to train a source-to-target NMT model. While target-side monolingual data is well utilized by NMT through BT and its variants (Sennrich et al., 2016a; Edunov et al., 2018; Hassan et al., 2018; He et al., 2016), the investigation on source-side monolingual data is very limited. Only few attempts (Zhang and Zong, 2016; Ueffing et al., 2007) are made to explore source-side monolingual data, with a common high-level idea that a source-to-target translation model is trained to translate the source-side monolingual data into target domain, the resulted synthetic data is then used for further training. In this work, we study how to leverage both source-side and target-side monolingual data to boost the accuracy of NMT.

We propose a simple yet effective strategy to leverage two-side monolingual data for NMT, which consists of three steps:
(1) *Preparation*: We pretrain a source-to-target and a target-to-source NMT models on the genuine bitext, and use them to generate synthetic bitext by translating the monolingual data from the source/target domain to the other domain respectively.
(2) *Large-scale Noised training*: The source sentences of the synthetic parallel corpus (including both the genuine source-side monolingual sentences and the synthetic source sentences back-

[*]The first two authors contributed equally to this work. This work was conducted at Microsoft Research Asia.
[†]Corresponding author.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2964–2975, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

translated from the target domain) are first corrupted. We then train an NMT model on this noised dataset together with the genuine bitext. We find that this step benefits from a large amount of monolingual data. The NMT model obtained from noised training can be further improved in the next finetune step.
(3) *Clean training*: We randomly generate a subset of the clean synthetic bitext without adding any noise, and leverage them together with the genuine bitext to finetune the output model of step (2). This step only needs a small set of synthetic bitext.

We conduct a comprehensive study of our proposed method on WMT English↔German translation and German↔French translation and have the following empirical observations:

- Using both source-side and target-side monolingual data is better than using monolingual data from only one domain (see Section 5.1).
- Adding noise to large-scale synthetic bitext improves the accuracy of NMT (see Section 5.2 and Section 5.3).
- Clean training/tuning of the model obtained from noised training further improves its accuracy (see Section 5.4).
- Our method achieves state-of-the-art results on English↔German newstest 2016, 2017 and 2018 and German→French newstest 2019 (see Section 4.2).

## 2 Related Work

Our work is related to several important research directions of NMT, and we describe the previous relevant works in this section.

### 2.1 Neural Machine Translation

NMT adopts the sequence-to-sequence framework, which consists of an encoder and a decoder in the network architecture. The encoder and decoder are usually built upon deep neural networks, which can be recurrent neural network (Sutskever et al., 2014), convolutional neural network (Gehring et al., 2017) or simple self-attention based transformer network (Vaswani et al., 2017). The encoder encodes the source sentence into a continuous representation space, and the decoder will decodes the target sentence based on the encoder representations word-by-word. The objective of the NMT model training is to maximize the conditional probability of the target sentence given the source sequence. Different model architectures and modifications have been proposed to improve the training efficiency and NMT accuracy (Hassan et al., 2018; Luong et al., 2015; Gu et al., 2016).

Different from the view of model architecture, in this paper, we study the NMT training from the data aspect. Specifically, we study the effect of both source-side and target-side monolingual data at scale and investigate how to make the best utilization of the monolingual data.

### 2.2 Improving NMT by Monolingual Data

NMT heavily relies on large-scale parallel dataset for training. To augment the limited bilingual data, there are plenty of works attempt to leverage the monolingual data to help the training, which includes the language model fusion (Gulcehre et al., 2015), back translation (Sennrich et al., 2016a), dual learning (He et al., 2016; Wang et al., 2018, 2019) and self learning (Zhang and Zong, 2016).

Gulcehre et al. (2017) integrates the hidden states from the target language model into the NMT decoder to improve the accuracy. Sennrich et al. (2016a) propose the back translation (BT) approach to leverage the target-side monolingual data, which is simple and effective. BT requires training an additional target-to-source NMT model given the bilingual dataset, the model will be used to back translate the target-side monolingual data. The translation output and the target-side monolingual data then paired as synthetic parallel corpus to augment the original bilingual dataset in order to further train the source-to-target NMT model. Dual learning (He et al., 2016) extends the BT approach to train NMT systems in both translation directions. When jointly training the source-to-target and target-to-source NMT models, the two models can provide back translated data for each other direction and perform multi-rounds BT. This strategy is also successfully adopted to build the unsupervised translation system (Lample et al., 2017).

There exists few attempts working on using the source-side monolingual data. Zhang and Zong (2016) propose self learning approach to generate the synthetic data for the source-side monolingual data, which is a semi-supervised method. Wu et al. (2017) leverage the source-side monolingual data to train the NMT system by learning reward func-

tion in a reinforcement learning framework.

### 2.3 Study of the Back Translation

Since BT is widely acknowledged and effective to improve the NMT model, there has been several works investigating back translation from different views. Poncelas et al. (2018) study on how using the back translated data as a training corpus (separately usage or combined with bilingual data) affects the performance of an NMT model. Burlot and Yvon (2019) analyze the accuracy impact from the quality of the BT data, the alternative uses of BT data to give explanations of why BT works. Cotterell and Kreutzer (2018) provide an interpretation of back translation as approximate inference in a generative model of bitext and give a new algorithm. However, above studies are all based on the small-scale monolingual data, therefore it remains unclear in the large-scale setting.

Edunov et al. (2018) firstly provide an extensive analysis of the back translation at scale. They investigate the different synthetic data generation methods, and also compare the different combination of the synthetic data with bitext. They finally build a strong system with millions of target-side monolingual sentences.

## 3 Training Strategy

In this section, we give a detailed introduction of our training strategy.

**Notations:** Let $X$ and $Y$ denote two languages, and let $\mathcal{X}$ and $\mathcal{Y}$ denote two corresponding language sentence domains, which are the collection of all sentences. Let $\mathcal{B} = \{(x_i, y_i)\}_{i=1}^{N}$ denote the bilingual training pairs, where $x_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$, $N$ is the number of sentence pairs. Let $\mathcal{M}_x = \{x_j\}_{j=1}^{M_x}$ and $\mathcal{M}_y = \{y_j\}_{j=1}^{M_y}$ denote the collections of monolingual sentences, where $M_x$ and $M_y$ are the sizes of the two sets, $x_j \in \mathcal{X}$, $y_j \in \mathcal{Y}$. Our objective is to obtain a translation model $f : \mathcal{X} \mapsto \mathcal{Y}$, that can translate sequences from language $X$ to language $Y$.

There are three steps of our training strategy:

**Step-1: Preparation.** We first train two translation models $f_b : \mathcal{X} \mapsto \mathcal{Y}$ and $g_b : \mathcal{Y} \mapsto \mathcal{X}$ on the given bilingual data $\mathcal{B}$ by minimizing the negative log-likelihood. After that, we build the following two synthetic datasets through the trained models:

$$\begin{aligned} \bar{\mathcal{B}}_s &= \{(x, f_b(x)) | x \in \mathcal{M}_x\}, \\ \bar{\mathcal{B}}_t &= \{(g_b(y), y) | y \in \mathcal{M}_y\}, \end{aligned} \tag{1}$$

where $\bar{\mathcal{B}}_s$ and $\bar{\mathcal{B}}_t$ can be seen the forward-translation (i.e., self learning) of source-side monolingual data (Zhang and Zong, 2016) and back-translation of target-side monolingual data (Sennrich et al., 2016a). In practice, $f_b$ and $g_b$ will output one translation by either beam search or random sampling according to the model output distribution. Random sampling explores more possibility for the unknown generated sentence, which may benefit the data augmentation. In this work, we mainly adopt beam search to generate the sentences, but we will also compare different sequence generation methods in Section 5.2.

**Step-2: Large-scale noised training.** Inspired from the noised back translation (Edunov et al., 2018) and denoising auto-encoder (Vincent et al., 2008), we add noise to the source-side data of both $\bar{\mathcal{B}}_s$ and $\bar{\mathcal{B}}_t$ for training instead of directly using them to train models. We build two following noised versions of the two augmented datasets,

$$\begin{aligned} \bar{\mathcal{B}}_s^n &= \{(\sigma(x), y) | (x, y) \in \bar{\mathcal{B}}_s\}, \\ \bar{\mathcal{B}}_t^n &= \{(\sigma(x), y) | (x, y) \in \bar{\mathcal{B}}_t\}, \end{aligned} \tag{2}$$

where $\sigma$ is the operator of adding noise. We follow Edunov et al. (2018) to design $\sigma$. Specifically,
(1) we randomly replace a word in the sentence to be a special token "<UNK>" (representing unknown words) with probability $0.1$;
(2) we randomly drop the words in each position with probability $0.1$;
(3) we randomly shuffle (swap) the words in the sentence, with constraint that the words will not be shuffled further than three positions distance.

We then train an NMT model $f_n$ for $X$ to $Y$ translation on $\mathcal{B} \cup \bar{\mathcal{B}}_s^n \cup \bar{\mathcal{B}}_t^n$ by minimizing the negative log-likelihood. Compared with Edunov et al. (2018), we enlarge the noised data from $\bar{\mathcal{B}}_t^n$ to $\bar{\mathcal{B}}_s^n \cup \bar{\mathcal{B}}_t^n$, instead of using the target-side monolingual data only. The intuition behind noised training is to force the encoder to discover more robust features and thus improve the generalization ability (Vincent et al., 2008). Besides, the output model of noised training has great potential to be improved in further finetune. Adding noise is widely acknowledged in other NLP tasks, like unsupervised NMT (Lample et al., 2017), BERT pretaining (Devlin et al., 2019), etc. There are some

other ways to augment/add noise to the training data (Artetxe et al., 2017) and we leave the combination with those approaches as future work.

**Step-3: Clean data tuning.** After obtaining noised training model from step-2, we further finetune it on the clean version of the synthetic data without adding noise manually. For the efficiency, we can randomly subsample $\bar{\mathcal{B}}_s$ and $\bar{\mathcal{B}}_t$ to form the clean $\bar{\mathcal{B}}_s^s$ and $\bar{\mathcal{B}}_t^s$ dataset. Then we continue tuning $f_n$ on the new datasets:

$$\min \sum_{(x,y)\in\mathcal{B}\cup\bar{\mathcal{B}}_s^s\cup\bar{\mathcal{B}}_t^s} -\log P(y|x;f), \quad (3)$$

where $f$ is initialized by $f_n$.

There are different ways to obtain $\bar{\mathcal{B}}_s^s$ and $\bar{\mathcal{B}}_t^s$, such as the subset of $\bar{\mathcal{B}}_s$ and $\bar{\mathcal{B}}_t$ we described here. A more effective way is to train another $\bar{f}_b$ for $X$ to $Y$ translation and another $\bar{g}_b$ for $Y$ to $X$ translation, and use them to build new synthetic data for step-3. In this way, more diverse samples are included and we are able to achieve better results. We will provide more discussions in Section 4.1 and study in Section 5.4.

## 4 Experiments

We verify the effectiveness of our proposed training strategy in this section. We conduct experiments on four different translation tasks. We also make a comprehensive study about the effect of the monolingual data usage in our approach from various aspects.

### 4.1 Experimental Setup

**Datasets** We carry out experiments on four different translation tasks from WMT19 competition[1], including En→De, De→En, De→Fr and Fr→De, where En, De and Fr are short for English, German and French.

For En↔De translation tasks, the bilingual data consists of two parts: (1) We concatenate "Europarl v9", "News Commentary v14", "Common Crawl corpus" and "Document-split Rapid corpus", remove the empty and duplicate lines and eventually get a clean dataset of news domain. (2) We also use the Paracrawl dataset to extend the bilingual corpus. Consider Paracrawl is noisy, we apply a series of filtration rules to this dataset and remove the low-quality sentences, including sentences with too many punctuation marks or invalid characters, and those with too many or too few words, etc. All the rules are available in the `preprocess.py` in the supplementary document. These two parts of data are then merged together to get the bilingual dataset. We eventually get a clean corpus with about $5M$ clean data and $18M$ Paracrawl data, which are denoted as `WMT` and `WMTPC` respectively for ease of reference. The monolingual data we use is from newscrawl released by WMT19. We combine the newscrawl data from year 2016 to 2018 for the English and German monolingual corpus. After filtering with similar rules applied in Paracrawl (`preprocess_mono.py` in the supplementary document), an additional step is that we further perform language detection (Lui and Baldwin, 2012) on each side monolingual data. Finally, we randomly select about 120M sentences for each of English and German language. We choose newstest2015 as the validation set and use newstest2016 to newstest2019 as test sets.

For De↔Fr translation data, we follow the same process as that used in En↔De. Eventually, the `WMT` and ParaCrawl data of De↔Fr contains about 2M and 4.8M sentence pairs respectively. As for the French monolingual data, we use all the available newscrawl from previous years due to its small size compared to En and De. After filtering, we keep 60M monolingual data. We use the validation and test set provided by WMT19 for De↔Fr translation.

All datasets are tokenized with Moses (Koehn et al., 2007) toolkit[2]. The vocabulary is built based on the Byte-Pair-Encoding (BPE) (Sennrich et al., 2016b) with 35K merge operations[3] for both En↔De and De↔Fr datasets. We learn the BPE vocabulary jointly on the source and target language sentences.

**Models** We choose the state-of-the-art Transformer network as our model structure, which consists of an encoder with 6 layers and a decoder with 6 layers. We use `transformer_big` configuration for all experiments: the dimension of word embedding and the inner feed-forward layer is 1024 and 4096 respectively. The parameters of

[1] http://www.statmt.org/wmt19/translation-task.html

[2] https://github.com/moses-smt/mosesdecoder/tree/master

[3] https://github.com/rsennrich/subword-nmt

source and target word embeddings, as well as the projection layer before softmax are shared. The number of attention heads is 16. The dropout is fixed as 0.2 due to validation performance. We conduct experiments on the fairseq (Ott et al., 2019) codebase[4].

**Training** The models are trained by the Adam optimizer (Kingma and Ba, 2014) with $\beta_1 = 0.9$, $\beta_2 = 0.98$. We use the default learning rate schedule used in Vaswani et al. (2017) with the initial learning rate $5 \times 10^{-4}$. Label smoothing (Szegedy et al., 2016) is adopted with value 0.1. The batch size is set to be 4096 tokens per GPU. We use 4 P40 GPUs for training and update the parameters every 16 minibatches. The pretraining and the noised training both take about one week, and the finetune process takes about one day training.

To effectively leverage monolingual data, as mentioned in Section 3, we use two different groups of models to generate synthetic data for noised training and finetuning. For noised training, the models used for translating monolingual data are trained on WMT; for finetuning, the models are trained on WMTPC. The intuition behind using different generation models is that each optimization stage can benefit from the new/unseen generated synthetic training data. Therefore, we adopt this way in our experiments. As for the data size, in noised training, $|\bar{\mathcal{B}}_s| = |\bar{\mathcal{B}}_t| = 60M$, and during finetuning, $|\bar{\mathcal{B}}_s^s| = |\bar{\mathcal{B}}_t^s| = 20M$.

**Evaluation** To evaluate the model performance, we use beam search generation with beam width 5 and without length penalty. The BLEU score is measured by the de-tokenized case-sensitive SacreBLEU (Post, 2018), which is widely adopted in the WMT competitions[5].

### 4.2 Results

We summarize the results of our training strategy for En↔De and De↔Fr translations in Table 1 and Table 2.

As can be seen from Table 1, the Paracrawl dataset improves the model accuracy by a large margin. Compared with the baseline of using WMT only, on average, Paracrawl brings more than 3.0 and 4.0 BLEU improvements to En→De and De→En tasks respectively, indicating the effectiveness of leveraging more data. After large-scale noised training, the accuracy of each task is further boosted. On average, we improve the BLEU scores of En→De and De→En by 1.6 and 2.2 points, demonstrating the effectiveness of noised training. At last, the output models are tuned on the clean synthetic data. This step further brings 1.7 and 1.5 points gain over the previous step. For En↔De translation tasks, we finally achieve 40.9, 32.9, 49.2 and 43.8 BLEU scores from newstest2016 to newstest2019; for De→En, the eventual BLEU scores of the four test sets end up at 47.5, 41.0, 49.5 and 41.9. Such strong improvements over the baseline reveal the great potential of our proposed strategy.

| | De→Fr | Fr→De |
|---|---|---|
| WMT | 31.2 | 26.1 |
| WMTPC | 34.2 | 28.6 |
| +*Noised Training* | 36.1 | 30.8 |
| **+*Clean Tuning*** | **37.3** | **33.1** |

Table 2: De-tokenized case-sensitive SacreBLEU on WMT De↔Fr newstest2019. "+" is conducted upon WMTPC dataset.

Prior to this work, the most common way to leverage monolingual data is BT (Sennrich et al., 2016a). We also compare our strategy to the vanilla BT, which consists of $20M$ synthetic data and WMTPC. The results are shown in the last row of Table 1. Without finetuning, our noised training strategy surpasses the BT by 0.92 and 0.55 points respectively. This shows that our strategy is more effective than standard BT.

In Table 2, we show the results of De↔Fr translation tasks. We have similar observation as that for En↔De translations. Specifically, the noised training can improve WMTPC by 1.9 and 2.2 points on De→Fr and Fr→De. When turning to clean tuning, we can obtain another 1.2 and 2.3 points improvement. The results on De↔Fr demonstrate that our strategy works across different languages.

Our method achieves state-of-the-art results on En↔De newstest2016, newstest2017 and newstest2018 and De→Fr newstest2019. We list several widely acknowledged systems on En→De translation in Table 3: MS-Marian (Junczys-Dowmunt, 2018), which is the champion of

[4] https://github.com/pytorch/fairseq

[5] https://github.com/mjpost/sacreBLEU. SacreBLEU signatures: BLEU+case.mixed+lang.$task +numrefs.1+smooth.exp+test.$SET+tok.13a+version.1.2.12, where $task are en-de, de-en, de-fr and fr-de; $SET are 16, 17, 18, 19.

| Model | En→De | | | | | De→En | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **2016** | **2017** | **2018** | **2019** | **Avg** | **2016** | **2017** | **2018** | **2019** | **Avg** |
| WMT | 34.0 | 28.0 | 41.3 | 37.3 | 35.15 | 38.6 | 34.3 | 41.1 | 34.5 | 37.13 |
| WMTPC | 37.1 | 30.5 | 45.6 | 40.3 | 38.38 | 41.9 | 37.5 | 45.4 | 40.1 | 41.23 |
| +*Noised Training* | 39.3 | 32.0 | 47.5 | 41.2 | 40.00 | 46.1 | 39.8 | 47.7 | 40.2 | 43.45 |
| ***+Clean Tuning*** | **40.9** | **32.9** | **49.2** | **43.8** | **41.70** | **47.5** | **41.0** | **49.5** | **41.9** | **44.98** |
| WMTPC+BT | 38.7 | 31.8 | 46.0 | 39.8 | 39.08 | 45.8 | 39.8 | 47.2 | 38.6 | 42.90 |

Table 1: De-tokenized case-sensitive SacreBLEU on WMT En↔De newstest2016, newstest2017, newstest2018, newstest2019 and the average score. "Avg" means the average BLEU score. "+" is conducted upon WMTPC dataset.

| Model (En→De) | **2016** | **2017** | **2018** |
|---|---|---|---|
| FAIR (ensemble) | 38.0 | 32.8 | 46.1 |
| MS-Marian (ensemble) | 39.6 | 31.9 | 48.3 |
| **Ours (single)** | **40.9** | **32.9** | **49.2** |

Table 3: De-tokenized case-sensitive SacreBLEU on WMT En→De newstest2016, newstest2017 and newstest2018. MS-Marian and FAIR are ensemble results while ours are single-model results.

| Model (De→En) | **2016** | **2017** | **2018** |
|---|---|---|---|
| UCAM (ensemble) | 45.1 | 38.7 | 48.0 |
| RWTH (ensemble) | 46.0 | 39.9 | 48.4 |
| **Ours (single)** | **47.5** | **41.0** | **49.5** |

Table 4: De-tokenized case-sensitive SacreBLEU on WMT De→En newstest2016, newstest2017 and newstest2018. UCAM and RWTH are ensemble results while ours are single-model results.

WMT18 En→De competition, and FAIR's model (Edunov et al., 2018) which leverages a large amount of monolingual data. Note that results of MS-Marian and FAIR are from ensemble models, while ours are from a single model. We find that our single model successfully surpasses the previous best systems in all test sets. Especially, for newstest2016 and newstest2018, we achieve 1.0 BLEU score improvement over the MS-Marian and set new records for these tasks. We also list the WMT18 top-2 systems for De→En translation in Table 4: RWTH (Graça et al., 2018) and UCAM (Stahlberg et al., 2018) systems, which are both ensemble models. Similarly, our single model surpasses these ensemble systems by a large margin.

## 5 Analysis

In this section, we provide a comprehensive study on the effect of monolingual data from different aspects, including data combination ways, data scale and the data generation methods. The experiments are conducted on the En→De translation and evaluated on newstest2016, newstest2017 and newstest2018. In this section, without specific clarification, the training data of each setting contains WMTPC.

### 5.1 Source or Target Monolingual Data

We first investigate the effect about different combinations of monolingual data. Specifically, we compare three different ways to use monolingual data, including leveraging source-side monolingual data only (i.e., $\bar{\mathcal{B}}_s$), target-side monolingual data only (i.e., $\bar{\mathcal{B}}_t$), and the monolingual data from both two sides (i.e., $\bar{\mathcal{B}}_s$ and $\bar{\mathcal{B}}_t$).

We keep the number of total synthetic data to be same across the three settings, that is, (i) 120M $\bar{\mathcal{B}}_s$; (ii) 120M $\bar{\mathcal{B}}_t$, and (iii) the combination of 60M $\bar{\mathcal{B}}_s$ and 60M $\bar{\mathcal{B}}_t$. We conduct this study on the clean synthetic data without noise, in order to verify which kind of data is more helpful for boosting performances.

The results are shown in Figure 1. From the figure, we can clearly observe that the best configuration is the combination of $\bar{\mathcal{B}}_s$ and $\bar{\mathcal{B}}_t$. With the same data size, on average, the model trained with combined synthetic dataset outperforms those trained with $\bar{\mathcal{B}}_s$ only and $\bar{\mathcal{B}}_t$ only by 2.6 and 3.5 points respectively. In particular, on newstest2018, the advantage of the combined dataset is nearly 5 points compared to $\bar{\mathcal{B}}_t$, which is extremely significant. This result strongly supports us that our training strategy by leveraging both the source-side and target-side monolingual

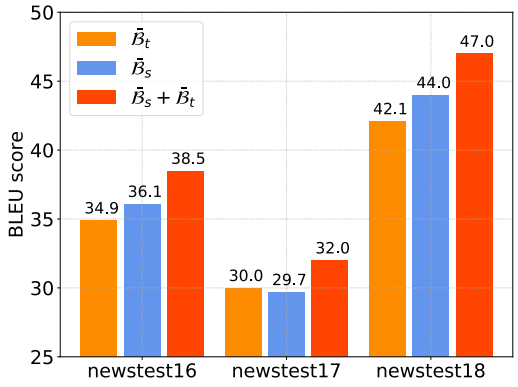


Figure 1: The de-tokenized SacreBLEU scores on En→De newstest2016, newstest2017 and newstest2018 of the models trained by different synthetic data: (1) $\bar{\mathcal{B}}_s$ from source-side monolingual data only, (2) $\bar{\mathcal{B}}_t$ from target-side monolingual data only and (3) the combination of $\bar{\mathcal{B}}_s$ and $\bar{\mathcal{B}}_t$.

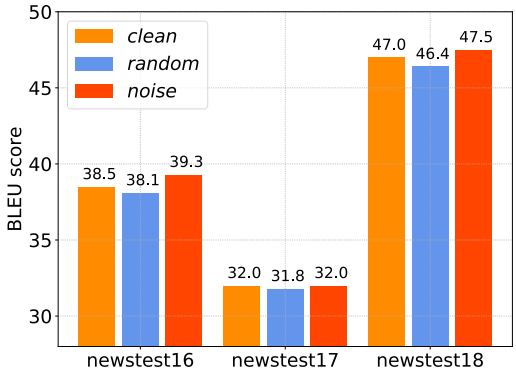


Figure 2: The de-tokenized SacreBLEU scores on En→De newstest2016, newstest2017 and newstest2018 of the models trained by synthetic data generated in different ways: (1) clean $\bar{\mathcal{B}}_s$ and $\bar{\mathcal{B}}_t$ data, (2) $\bar{\mathcal{B}}_s^r$ and randomly sampled $\bar{\mathcal{B}}_t^r$ data, and (3) noised $\bar{\mathcal{B}}_s^n$ and $\bar{\mathcal{B}}_t^n$ data.

data is helpful. Another point is that on such large-scale dataset, the $\bar{\mathcal{B}}_t$ data from back translation seems to be worse than the $\bar{\mathcal{B}}_s$ data generated by source-side monolingual data, which also supports the point that source-side monolingual is helpful.

### 5.2 Synthetic Data Generation

We conduct experiments on different generated synthetic data, to verify whether adding noise is essential. We use the same combined dataset (i.e., $60M$ $\bar{\mathcal{B}}_s$ and $60M$ $\bar{\mathcal{B}}_t$) as that used in Section 5.1 since the effectiveness has been verified.

We compare our noised training data $\bar{\mathcal{B}}_s^n$ and $\bar{\mathcal{B}}_t^n$ with another two baselines: 1) $\bar{\mathcal{B}}_s$ and $\bar{\mathcal{B}}_t$ without any transformation; 2) $\bar{\mathcal{B}}_s$ and the randomly sampled synthetic data $\bar{\mathcal{B}}_t^r$, where each token in the translation is sampled from the multinomial distribution determined by the NMT model. Random sampling is an alternative way to introduce noise, where the resulted synthetic dataset is more diverse but the translation quality is relatively poor. The data sizes for all settings are 120M.

We present the result in Figure 2. Overall, noised traininig outperforms the clean training, e.g., 0.8 and 0.5 points advantage on newstest2016 and newstest2018 respectively. Compared with randomly sampled data, our noised data training achieves more than 1.0 BLEU improvement on newstest2016 and newstest2018, and similar performance on newstest2017. The above results suggest that our noised training is effective.

### 5.3 Scale of Monolingual Data

In this section, we give a comparison of different data scales for each kind of synthetic data.

**(a) Scale of Target-side (BT) Data** We first look into the widely adopted back translation data $\bar{\mathcal{B}}_t$. We vary the data scale from 20M to 60M, which are obtained by random selection from the 120M corpus, and then to 120M.

From Figure 3(a), we can observe that model performance is improved when we add 20M $\bar{\mathcal{B}}_t$ data. However, adding more clean back translated data starts to hurt the model performances. We suspect the reason is that the data distribution of $\bar{\mathcal{B}}_t$ shifts from the groundtruth distribution. Adding too much $\bar{\mathcal{B}}_t$ will enforce the model training to fit a biased distribution, making the generalization ability drop and thus, we observe that the BLEU scores drop. This result implies that we should choose a proper data size when using $\bar{\mathcal{B}}_T$ only, such as same size of bitext (e.g., 20M as we used).

**(b) Scale of Source-side Data** We then investigate the effect of data scale of source-side monolingual data, i.e., $\bar{\mathcal{B}}_s$ data. We also set the data sizes as {20M,60M,120M}.

The results are shown in Figure 3(b). We do not observe any improvement over the bilingual system when adding $\bar{\mathcal{B}}_s$ data, and with more data added, the performance slightly drops. It seems to be contrary to the experimental results in Zhang and Zong (2016) which are conducted on RNN models. We speculate the self learning approach

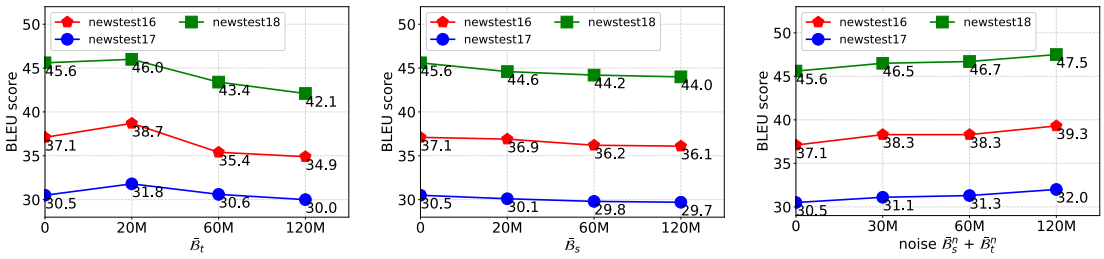


(a) Different scales of $\bar{\mathcal{B}}_t$ data. (b) Different scales of $\bar{\mathcal{B}}_s$ data. (c) Different scales of noised $\bar{\mathcal{B}}_s + \bar{\mathcal{B}}_t$ data.

Figure 3: The de-tokenized SacreBLEU scores on newstest2016, newstest2017, newstest2018 of the models trained with varied data scales of (a) $\bar{\mathcal{B}}_t$ data, (b) $\bar{\mathcal{B}}_s$ data, and (c) combined $\bar{\mathcal{B}}_s$ and $\bar{\mathcal{B}}_t$ data.

is actually hard for the model to boost itself, since there are no other signals to help the learning. The results demonstrate that leveraging source-side monolingual data alone is not a good choice.

**(c) Scale of Noised Synthetic Data** We finally study our noised training on different sizes of noised synthetic data. The experiments are conducted on the combination of noised 30M, 60M and 120M $\bar{\mathcal{B}}_s^n$ and $\bar{\mathcal{B}}_t^n$ with the same number of source-side and target-side monolingual sentences[6]. We choose the maximum data size as 120M due to GPU memory limitation. The results are shown in Figure 3(c).

We can see that the performances are consistently improved as the number of the noised synthetic data increases on all test sets. The result again proves that the hybrid usage of source-side and target-side monolingual data is an effective approach which outperforms using the two kinds of data individually.

In summary, first, we verify that source-side monolingual data is helpful and the best way to use it is to combine with target-side monolingual data. Next, we show that adding noise to synthetic data outperforms that without noise. Finally, we empirically prove that our strategy benefits from more monolingual data, while BT does not, demonstrating our strategy has great potential of utilizing more data.

### 5.4 Synthetic Tuning

We further study the clean tuning step in our training strategy. Two questions remain to be answered: (1) Is it helpful to use two groups of models of building synthetic data for noised training and clean training as we discussed before? (2) Is it helpful to use noised training first regarding future BLEU score achieved by the finetune step?

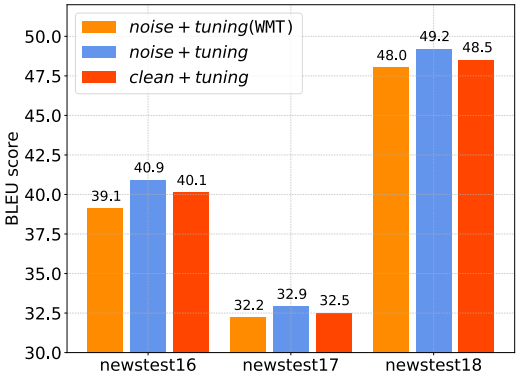


Figure 4: The de-tokenized SacreBLEU scores on En→De newstest2016, newstest2017 and newstest2018 of the models tuned by different synthetic data and pretrained models: (1) noised training model tuned on the subset of $\bar{\mathcal{B}}_s$ and $\bar{\mathcal{B}}_t$ (WMT in the figure), (2) noised training model tuned on the synthetic data as introduced in Section 4.1, and (3) clean training model tuned on the same synthetic data generated as (2).

To answer these questions, we conduct another two experiments: (1) At finetuning step, use a subsample of synthetic data of noised training step, which is $\bar{\mathcal{B}}_s^s \subset \bar{\mathcal{B}}_s, \bar{\mathcal{B}}_t^s \subset \bar{\mathcal{B}}_t$. (2) Training without noise first and then finetuning using clean data.

We present the results in Figure 4. Obviously, we can see that tuning on the subset of $\bar{\mathcal{B}}_s$ and $\bar{\mathcal{B}}_t$ is worse than tuning on another set of synthetic data used in our experiments (e.g., 49.2 v.s. 48.0 on newstest2018). In addition, first training on the clean synthetic data and then tuning on other clean synthetic data also makes improvement, but similar to the results shown in Section 5.2, it is still not as good as our noised pretraining. Above results

[6]For example, same as combined 120M data, 60M combined data contains 30M $\bar{\mathcal{B}}_s^n$ and 30M $\bar{\mathcal{B}}_t^n$ data.

again prove the importance of our noised training and tuning on more diverse dataset is effective.

## 6 Conclusion

In this work, we exploit the monolingual data at scale for the neural machine translation. Different from previous works which usually adopt back translation on target-side monolingual data only, we propose an effective training strategy to boost the NMT performance by leveraging both source-side and target-side monolingual data. Our approach contains three steps: synthetic data generation, large-scale noised training on synthetic data and clean data training/tuning. We verify our approach on the widely acknowledged WMT English↔German translation tasks and achieve state-of-the-art results, as well as that for the WMT German↔French translations. We also make a comprehensive study on the monolingual data utilization.

For future work, we would like to verify our training strategy on more language pairs and other sequence-to-sequence tasks. Furthermore, we are interested in studying our noised training with other data augmentation approaches.

## References


Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2017. Unsupervised neural machine translation. *arXiv preprint arXiv:1710.11041*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Franck Burlot and François Yvon. 2019. Using monolingual data in neural machine translation: a systematic study. *arXiv preprint arXiv:1903.11437*.

Yong Cheng, Wei Xu, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Semi-supervised learning for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1965–1974.

Mara Chinea-Rios, Alvaro Peris, and Francisco Casacuberta. 2017. Adapting neural machine translation with parallel synthetic data. In *Proceedings of the Second Conference on Machine Translation*, pages 138–147.

Ryan Cotterell and Julia Kreutzer. 2018. Explaining and generalizing back-translation through wake-sleep. *arXiv preprint arXiv:1806.04402*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *NAACL*.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1243–1252. JMLR. org.

Miguel Graça, Yunsu Kim, Julian Schamper, Jiahui Geng, and Hermann Ney. 2018. The rwth aachen university english-german and german-english unsupervised neural machine translation systems for wmt 2018. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 377–385.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1631–1640.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. *arXiv preprint arXiv:1503.03535*.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, and Yoshua Bengio. 2017. On integrating a language model into neural machine translation. *Computer Speech & Language*, 45:137–148.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, et al. 2018. Achieving human parity on automatic chinese to english news translation. *arXiv preprint arXiv:1803.05567*.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In *Advances in Neural Information Processing Systems*, pages 820–828.

Marcin Junczys-Dowmunt. 2018. Microsoft's submission to the wmt2018 news translation task: How i learned to stop worrying and love the data. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 425–430.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the association for computational linguistics companion volume proceedings of the demo and poster sessions*, pages 177–180.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2017. Unsupervised machine translation using monolingual corpora only. *arXiv preprint arXiv:1711.00043*.

Marco Lui and Timothy Baldwin. 2012. langid. py: An off-the-shelf language identification tool. In *Proceedings of the ACL 2012 system demonstrations*, pages 25–30. Association for Computational Linguistics.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Alberto Poncelas, Dimitar Shterionov, Andy Way, Gideon Maillette de Buy Wenniger, and Peyman Passban. 2018. Investigating backtranslation in neural machine translation. In *Proceedings of the 21st Annual Conference of the European Association for Machine Translation: 28-30 May 2018, Universitat d'Alacant, Alacant, Spain*, pages 249–258. European Association for Machine Translation.

Matt Post. 2018. A call for clarity in reporting bleu scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. In *ACL*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1715–1725.

Felix Stahlberg, Adrià de Gispert, and Bill Byrne. 2018. The university of cambridges machine translation systems for wmt18. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 504–512.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Nicola Ueffing, Gholamreza Haffari, and Anoop Sarkar. 2007. Transductive learning for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 25–32.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM.

Yijun Wang, Yingce Xia, Li Zhao, Jiang Bian, Tao Qin, Guiquan Liu, and Tie-Yan Liu. 2018. Dual transfer learning for neural machine translation with marginal distribution regularization. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Yiren Wang, Yingce Xia, Tianyu He, Fei Tian, Tao Qin, ChengXiang Zhai, and Tie-Yan Liu. 2019. Multi-agent dual learning. In *International Conference on Learning Representations*.

Lijun Wu, Fei Tian, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2018. A study of reinforcement learning for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3612–3621.

Lijun Wu, Li Zhao, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2017. Sequence prediction with unlabeled data by reward function learning. In *IJCAI*, pages 3098–3104.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Jiajun Zhang and Chengqing Zong. 2016. Exploiting source-side monolingual data in neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545.

Zhirui Zhang, Shujie Liu, Mu Li, Ming Zhou, and Enhong Chen. 2018. Joint training for neural machine translation models with monolingual data. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

# Meta Relational Learning for Few-Shot Link Prediction in Knowledge Graphs

**Mingyang Chen**[1,*], **Wen Zhang**[1,*], **Wei Zhang**[2], **Qiang Chen**[2], **Huajun Chen**[1,3,†]

[1]College of Computer Science and Technology, Zhejiang University
[2]Alibaba Group
[3]AZFT Joint Lab for Knowledge Engine
{mingyangchen, wenzhang2015, huajunsir}@zju.edu.cn
{lantu.zw, lapu.cq}@alibaba-inc.com

## Abstract

Link prediction is an important way to complete knowledge graphs (KGs), while embedding-based methods, effective for link prediction in KGs, perform poorly on relations that only have a few associative triples. In this work, we propose a Meta Relational Learning (MetaR) framework to do the common but challenging few-shot link prediction in KGs, namely predicting new triples about a relation by only observing a few associative triples. We solve few-shot link prediction by focusing on transferring relation-specific meta information to make model learn the most important knowledge and learn faster, corresponding to relation meta and gradient meta respectively in MetaR. Empirically, our model achieves state-of-the-art results on few-shot link prediction KG benchmarks.

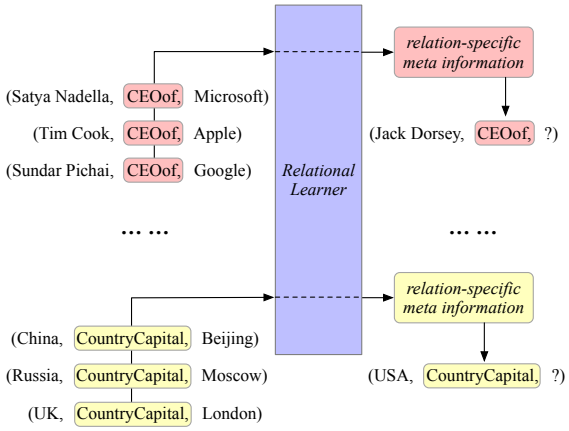


Figure 1: An example of 3-shot link prediction in KGs. One task represents observing only three instances of one specific relation and conducting link prediction on this relation. Our model focuses on extracting relation-specific meta information by a kind of relational learner which is shared across tasks and transferring this meta information to do link prediction within one task.

## 1 Introduction

A knowledge graph is composed by a large amount of triples in the form of $(head\ \ entity, relation, tail\ \ entity)$ ($(h, r, t)$ in short), encoding knowledge and facts in the world. Many KGs have been proposed (Vrandei and Krtzsch, 2014; Bollacker et al., 2008; Carlson et al., 2010) and applied to various applications (Bordes et al., 2014; Zhang et al., 2016, 2019a).

Although with huge amount of entities, relations and triples, many KGs still suffer from incompleteness, thus knowledge graph completion is vital for the development of KGs. One of knowledge graph completion tasks is link prediction, predicting new triples based on existing ones. For link prediction, KG embedding methods (Bordes et al., 2013; Nickel et al., 2011; Trouillon et al., 2016; Yang et al., 2015) are promising ways. They learn latent representations, called embeddings, for entities and relations in continuous vector space and accomplish link prediction via calculation with embeddings.

The effectiveness of KG embedding methods is promised by sufficient training examples, thus results are much worse for elements with a few instances during training (Zhang et al., 2019c). However, few-shot problem widely exists in KGs. For example, about 10% of relations in Wikidata (Vrandei and Krtzsch, 2014) have no more than 10 triples. Relations with a few instances are called few-shot relations. In this paper, we devote to discuss *few-shot link prediction in knowledge graphs*, predicting tail entity $t$ given head entity $h$ and relation $r$ by only observing $K$ triples about $r$, usually $K$ is small. Figure 1 depicts an example of 3-shot link prediction in KGs.

To do few-shot link prediction, Xiong et al.

* Equal contribution.
† Corresponding author.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4217–4226, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

(2018) made the first trial and proposed GMatching, learning a matching metric by considering both learned embeddings and one-hop graph structures, while we try to accomplish few-shot link prediction from another perspective based on the intuition that *the most important information to be transferred from a few existing instances to incomplete triples should be the common and shared knowledge within one task.* We call such information *relation-specific meta information* and propose a new framework Meta Relational Learning (MetaR) for few-shot link prediction. For example, in Figure 1, relation-specific meta information related to the relation *CEOof* or *CountryCapital* will be extracted and transferred by MetaR from a few existing instances to incomplete triples.

The relation-specific meta information is helpful in the following two perspectives: 1) transferring common relation information from observed triples to incomplete triples, 2) accelerating the learning process within one task by observing only a few instances. Thus we propose two kinds of relation-specific meta information: *relation meta* and *gradient meta* corresponding to afore mentioned two perspectives respectively. In our proposed framework MetaR, relation meta is the high-order representation of a relation connecting head and tail entities. Gradient meta is the loss gradient of relation meta which will be used to make a rapid update before transferring relation meta to incomplete triples during prediction.

Compared with GMatching (Xiong et al., 2018) which relies on a background knowledge graph, our MetaR is independent with them, thus is more robust as background knowledge graphs might not be available for few-shot link prediction in real scenarios.

We evaluate MetaR with different settings on few-shot link prediction datasets. MetaR achieves state-of-the-art results, indicating the success of transferring relation-specific meta information in few-shot link prediction tasks. In summary, main contributions of our work are three-folds:

- Firstly, we propose a novel meta relational learning framework (MetaR) to address few-shot link prediction in knowledge graphs.
- Secondly, we highlight the critical role of relation-specific meta information for few-shot link prediction, and propose two kinds of relation-specific meta information, *relation meta* and *gradient meta*. Experiments show that both of them contribute significantly.
- Thirdly, our MetaR achieves state-of-the-art results on few-shot link prediction tasks and we also analyze the facts that affect MetaR's performance.

## 2 Related Work

One target of MetaR is to learn the representation of entities fitting the few-shot link prediction task and the learning framework is inspired by knowledge graph embedding methods. Furthermore, using loss gradient as one kind of meta information is inspired by MetaNet (Munkhdalai and Yu, 2017) and MAML (Finn et al., 2017) which explore methods for few-shot learning by meta-learning. From these two points, we regard knowledge graph embedding and meta-learning as two main kinds of related work.

### 2.1 Knowledge Graph Embedding

Knowledge graph embedding models map relations and entities into continuous vector space. They use a score function to measure the truth value of each triple $(h, r, t)$. Same as knowledge graph embedding, our MetaR also need a score function, and the main difference is that representation for $r$ is the learned relation meta in MetaR rather than embedding of $r$ as in normal knowledge graph embedding methods.

One line of work is started by TransE (Bordes et al., 2013) with distance score function. TransH (Wang et al., 2014) and TransR (Lin et al., 2015b) are two typical models using different methods to connect head, tail entities and their relations. DistMult (Yang et al., 2015) and ComplEx (Trouillon et al., 2016) are derived from RESCAL (Nickel et al., 2011), trying to mine latent semantics in different ways. There are also some others like ConvE (Dettmers et al., 2018) using convolutional structure to score triples and models using additional information such as entity types (Xie et al., 2016) and relation paths (Lin et al., 2015a). Wang et al. (2017) comprehensively summarize the current popular knowledge graph embedding methods.

Traditional embedding models are heavily rely on rich training instances (Zhang et al., 2019b; Xiong et al., 2018), thus are limited to do few-shot link prediction. Our MetaR is designed to fill this vulnerability of existing embedding models.

### 2.2 Meta-Learning

Meta-learning seeks for the ability of learning quickly from only a few instances within the same concept and adapting continuously to more concepts, which are actually the rapid and incremental learning that humans are very good at.

Several meta-learning models have been proposed recently. Generally, there are three kinds of meta-learning methods so far: (1) *Metric-based* meta-learning (Koch et al., 2015; Vinyals et al., 2016; Snell et al., 2017; Xiong et al., 2018), which tries to learn a matching metric between query and support set generalized to all tasks, where the idea of matching is similar to some nearest neighbors algorithms. Siamese Neural Network (Koch et al., 2015) is a typical method using symmetric twin networks to compute the metric of two inputs. GMatching (Xiong et al., 2018), the first trial on one-shot link prediction in knowledge graphs, learns a matching metric based on entity embeddings and local graph structures which also can be regarded as a metric-based method. (2) *Model-based* method (Santoro et al., 2016; Munkhdalai and Yu, 2017; Mishra et al., 2018), which uses a specially designed part like memory to achieve the ability of learning rapidly by only a few training instances. MetaNet (Munkhdalai and Yu, 2017), a kind of memory augmented neural network (MANN), acquires meta information from loss gradient and generalizes rapidly via its fast parameterization. (3) *Optimization-based* approach (Finn et al., 2017; Lee and Choi, 2018), which gains the idea of learning faster by changing the optimization algorithm. Model-Agnostic Meta-Learning (Finn et al., 2017) abbreviated as MAML is a model-agnostic algorithm. It firstly updates parameters of task-specific learner, and meta-optimization across tasks is performed over parameters by using above updated parameters, it's like "a gradient through a gradient".

As far as we know, work proposed by Xiong et al. (2018) is the first research on few-shot learning for knowledge graphs. It's a metric-based model which consists of a neighbor encoder and a matching processor. Neighbor encoder enhances the embedding of entities by their one-hop neighbors, and matching processor performs a multi-step matching by a LSTM block.

| Training | |
|---|---|
| Task #1 (CountryCapital) | |
| Support | (China, CountryCapital, Beijing) |
| Query | (France, CountryCapital, Paris) |
| Task #2 (CEOof) | |
| Support | (Satya Nadella, CEOof, Microsoft) |
| Query | (Jack Dorsey, CEOof, Twitter) |
| **Testing** | |
| Task #1 (OfficialLanguage) | |
| Support | (Japan, OfficialLanguage, Japanese) |
| Query | (Spain, OfficialLanguage, Spanish) |

Table 1: The training and testing examples of 1-shot link prediction in KGs.

## 3 Task Formulation

In this section, we present the formal definition of a knowledge graph and few-shot link prediction task. A knowledge graph is defined as follows:

**Definition 3.1** *(Knowledge Graph $\mathcal{G}$) A knowledge graph $\mathcal{G} = \{\mathcal{E}, \mathcal{R}, \mathcal{TP}\}$. $\mathcal{E}$ is the entity set. $\mathcal{R}$ is the relation set. And $\mathcal{TP} = \{(h, r, t) \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}\}$ is the triple set.*

And a few-shot link prediction task in knowledge graphs is defined as:

**Definition 3.2** *(Few-shot link prediction task $\mathcal{T}$) With a knowledge graph $\mathcal{G} = \{\mathcal{E}, \mathcal{R}, \mathcal{TP}\}$, given a support set $\mathcal{S}_r = \{(h_i, t_i) \in \mathcal{E} \times \mathcal{E} | (h_i, r, t_i) \in \mathcal{TP}\}$ about relation $r \in \mathcal{R}$, where $|\mathcal{S}_r| = K$, predicting the tail entity linked with relation $r$ to head entity $h_j$, formulated as $r : (h_j, ?)$, is called K-shot link prediction.*

As defined above, a few-shot link prediction task is always defined for a specific relation. During prediction, there usually is more than one triple to be predicted, and with *support set* $\mathcal{S}_r$ , we call the set of all triples to be predicted as *query set* $\mathcal{Q}_r = \{r : (h_j, ?)\}$.

The goal of a few-shot link prediction method is to gain the capability of predicting new triples about a relation $r$ with only observing a few triples about $r$. Thus its training process is based on a set of tasks $\mathcal{T}_{train} = \{\mathcal{T}_i\}_{i=1}^{M}$ where each task $\mathcal{T}_i = \{\mathcal{S}_i, \mathcal{Q}_i\}$ corresponds to an individual few-shot link prediction task with its own support and query set. Its testing process is conducted on a set of new tasks $\mathcal{T}_{test} = \{\mathcal{T}_j\}_{j=1}^{N}$ which is similar to $\mathcal{T}_{train}$, other than that $\mathcal{T}_j \in \mathcal{T}_{test}$ should be about relations that have never been seen in $\mathcal{T}_{train}$.

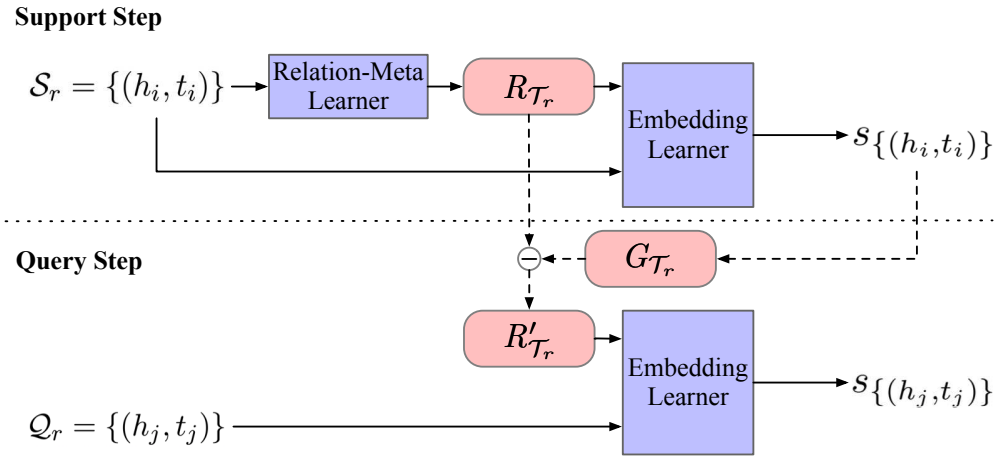


Figure 2: Overview of MetaR. $\mathcal{T}_r = \{\mathcal{S}_r, \mathcal{Q}_r\}$, $R_{\mathcal{T}_r}$ and $R'_{\mathcal{T}_r}$ represent relation meta and updated relation meta, and $G_{\mathcal{T}_r}$ represents gradient meta.

Table 1 gives a concrete example of the data during learning and testing for few-shot link prediction.

## 4 Method

To make one model gain the few-shot link prediction capability, the most important thing is transferring information from support set to query set and there are two questions for us to think about: (1) what is the most transferable and common information between support set and query set and (2) how to learn faster by only observing a few instances within one task. For question (1), within one task, all triples in support set and query set are about the same relation, thus it is naturally to suppose that relation is the key common part between support and query set. For question (2), the learning process is usually conducted by minimizing a loss function via gradient descending, thus gradients reveal how the model's parameters should be changed. Intuitively, we believe that gradients are valuable source to accelerate learning process.

Based on these thoughts, we propose two kinds of meta information which are shared between support set and query set to deal with above problems:

- **Relation Meta** represents the relation connecting head and tail entities in both support and query set and we extract relation meta for each task, represented as a vector, from support set and transfer it to query set.
- **Gradient Meta** is the loss gradient of relation meta in support set. As gradient meta shows how relation meta should be changed in order to reach a loss minima, thus to accelerate the learning process, relation meta is updated through gradient meta before being transferred to query set. This update can be viewed as the rapid learning of relation meta.

In order to extract relation meta and gradient mate and incorporate them with knowledge graph embedding to solve few-shot link prediction, our proposal, **MetaR**, mainly contains two modules:

- **Relation-Meta Learner** generates relation meta from heads' and tails' embeddings in the support set.
- **Embedding Learner** calculates the truth values of triples in support set and query set via entity embeddings and relation meta. Based on the loss function in embedding learner, gradient meta is calculated and a rapid update for relation meta will be implemented before transferring relation meta to query set.

The overview and algorithm of MetaR are shown in Figure 2 and Algorithm 1. Next, we introduce each module of MetaR via one few-shot link prediction task $\mathcal{T}_r = \{\mathcal{S}_r, \mathcal{Q}_r\}$.

### 4.1 Relation-Meta Learner

To extract the relation meta from support set, we design a relation-meta learner to learn a mapping from head and tail entities in support set to relation meta. The structure of this relation-meta learner can be implemented as a simple neural network.

In task $\mathcal{T}_r$, the input of relation-meta learner is head and tail entity pairs in support set $\{(h_i, t_i) \in \mathcal{S}_r\}$. We firstly extract entity-pair specific relation

**Algorithm 1** Learning of MetaR

**Require:** Training tasks $\mathcal{T}_{train}$
**Require:** Embedding layer $emb$; Parameter of relation-meta learner $\phi$
**while** not done **do**
  Sample a task $\mathcal{T}_r = \{\mathcal{S}_r, \mathcal{Q}_r\}$ from $\mathcal{T}_{train}$
  Get $R$ from $\mathcal{S}_r$ (Equ. 1, Equ. 2)
  Compute loss in $\mathcal{S}_r$ (Equ. 4)
  Get $G$ by gradient of $R$ (Equ. 5)
  Update $R$ by $G$ (Equ. 6)
  Compute loss in $\mathcal{Q}_r$ (Equ. 8)
  Update $\phi$ and $emb$ by loss in $\mathcal{Q}_r$
**end while**

meta via a $L$-layers fully connected neural network,

$$\begin{aligned} \mathbf{x}^0 &= \mathbf{h}_i \oplus \mathbf{t}_i \\ \mathbf{x}^l &= \sigma(\mathbf{W}^l \mathbf{x}^{l-1} + b^l) \\ R_{(h_i,t_i)} &= \mathbf{W}^L \mathbf{x}^{L-1} + b^L \end{aligned} \tag{1}$$

where $\mathbf{h}_i \in \mathbb{R}^d$ and $\mathbf{t}_i \in \mathbb{R}^d$ are embeddings of head entity $h_i$ and tail entity $t_i$ with dimension $d$ respectively. $L$ is the number of layers in neural network, and $l \in \{1, \ldots, L-1\}$. $\mathbf{W}^l$ and $\mathbf{b}^l$ are weights and bias in layer $l$. We use LeakyReLU for activation $\sigma$. $\mathbf{x} \oplus \mathbf{y}$ represents the concatenation of vector $\mathbf{x}$ and $\mathbf{y}$. Finally, $R_{(h_i,t_i)}$ represent the relation meta from specific entity pare $h_i$ and $t_i$.

With multiple entity-pair specific relation meta, we generate the final relation meta in current task via averaging all entity-pair specific relation meta in current task,

$$R_{\mathcal{T}_r} = \frac{\sum_{i=1}^{K} R_{(h_i,t_i)}}{K} \tag{2}$$

### 4.2 Embedding Learner

As we want to get gradient meta to make a rapid update on relation meta, we need a score function to evaluate the truth value of entity pairs under specific relations and also the loss function for current task. We apply the key idea of knowledge graph embedding methods in our embedding learner, as they are proved to be effective on evaluating truth value of triples in knowledge graphs.

In task $\mathcal{T}_r$, we firstly calculate the score for each entity pairs $(h_i, t_i)$ in support set $\mathcal{S}_r$ as follows:

$$s_{(h_i,t_i)} = \|\mathbf{h}_i + R_{\mathcal{T}_r} - \mathbf{t}_i\| \tag{3}$$

where $\|\mathbf{x}\|$ represents the L2 norm of vector $\mathbf{x}$. We design the score function inspired by TransE (Bordes et al., 2013) which assumes the head entity embedding $\mathbf{h}$, relation embedding $\mathbf{r}$ and tail entity embedding $\mathbf{t}$ for a true triple $(h, r, t)$ satisfying $\mathbf{h}+\mathbf{r} = \mathbf{t}$. Thus the score function is defined according to the distance between $\mathbf{h} + \mathbf{r}$ and $\mathbf{t}$. Transferring to our few-show link prediction task, we replace the relation embedding $\mathbf{r}$ with relation meta $R_{\mathcal{T}_r}$ as there is no direct general relation embeddings in our task and $R_{\mathcal{T}_r}$ can be regarded as the relation embedding for current task $\mathcal{T}_r$.

With score function for each triple, we set the following loss,

$$L(\mathcal{S}_r) = \sum_{(h_i,t_i) \in \mathcal{S}_r} [\gamma + s_{(h_i,t_i)} - s_{(h_i,t_i')}]_+ \tag{4}$$

where $[x]_+$ represents the positive part of $x$ and $\gamma$ represents margin which is a hyperparameter. $s_{(h_i,t_i')}$ is the score for negative sample $(h_i, t_i')$ corresponding to current positive entity pair $(h_i, t_i) \in \mathcal{S}_r$, where $(h_i, r, t_i') \notin \mathcal{G}$.

$L(\mathcal{S}_r)$ should be small for task $\mathcal{T}_r$ which represents the model can properly encode truth values of triples. Thus gradients of parameters indicate how should the parameters be updated. Thus we regard the gradient of $R_{\mathcal{T}_r}$ based on $L(\mathcal{S}_r)$ as gradient meta $G_{\mathcal{T}_r}$:

$$G_{\mathcal{T}_r} = \nabla_{R_{\mathcal{T}_r}} L(\mathcal{S}_r) \tag{5}$$

Following the gradient update rule, we make a rapid update on relation meta as follows:

$$R'_{\mathcal{T}_r} = R_{\mathcal{T}_r} - \beta G_{\mathcal{T}_r} \tag{6}$$

where $\beta$ indicates the step size of gradient meta when operating on relation meta.

When scoring the query set by embedding learner, we use updated relation meta. After getting the updated relation meta $R'$, we transfer it to samples in query set $\mathcal{Q}_r = \{(h_j, t_j)\}$ and calculate their scores and loss of query set, following the same way in support set:

$$s_{(h_j,t_j)} = \|\mathbf{h}_j + R'_{\mathcal{T}_r} - \mathbf{t}_j\| \tag{7}$$

$$L(\mathcal{Q}_r) = \sum_{(h_j,t_j) \in \mathcal{Q}_r} [\gamma + s_{(h_j,t_j)} - s_{(h_j,t_j')}]_+ \tag{8}$$

where $L(\mathcal{Q}_r)$ is our training objective to be minimized. We use this loss to update the whole model.

| Dataset | Fit | # Train | # Dev | # Test |
|---|---|---|---|---|
| NELL-One | Y | 321 | 5 | 11 |
| Wiki-One | Y | 589 | 16 | 34 |
| NELL-One | N | 51 | 5 | 11 |
| Wiki-One | N | 133 | 16 | 34 |

Table 2: Statistic of datasets. Fit denotes fitting background into training tasks (Y) or not (N), # Train, # Dev and # Test denote the number of relations in training, validation and test set.

| Background Conf. | Description |
|---|---|
| BG:Pre-Train | Use background to train entity embedding in advance. |
| BG:In-Train | Fit background graph into training tasks. |
| BG:Discard | Discard the background graph. |

Table 3: Three forms of datasets in our experiments.

### 4.3 Training Objective

During training, our objective is to minimize the following loss $L$ which is the sum of query loss for all tasks in one minibatch:

$$L = \sum_{(\mathcal{S}_r, \mathcal{Q}_r) \in \mathcal{T}_{train}} L(\mathcal{Q}_r) \tag{9}$$

## 5 Experiments

With MetaR, we want to figure out following things: 1) can MetaR accomplish few-shot link prediction task and even perform better than previous model? 2) how much relation-specific meta information contributes to few-shot link prediction? 3) is there any requirement for MetaR to work on few-shot link prediction? To do these, we conduct the experiments on two few-shot link prediction datasets and deeply analyze the experiment results [1].

### 5.1 Datasets and Evaluation Metrics

We use two datasets, NELL-One and Wiki-One which are constructed by Xiong et al. (2018). NELL-One and Wiki-One are derived from NELL (Carlson et al., 2010) and Wikidata (Vrandei and Krtzsch, 2014) respectively. Furthermore, because these two benchmarks are firstly tested on GMatching which consider both learned embeddings and one-hop graph structures, a background graph is constructed with relations out of training/validation/test sets for obtaining the pre-train entity embeddings and providing the local graph for GMatching.

Unlike GMatching using background graph to enhance the representations of entities, our MetaR can be trained without background graph. For NELL-One and Wiki-One which have background graph originally, we can make use of such background graph by fitting it into training tasks or using it to train embeddings to initialize entity representations. Overall, we have three kinds of dataset settings, shown in Table 3. For setting of *BG:In-Train*, in order to make background graph included in training tasks, we sample tasks from triples in background graph and original training set, rather than sampling from only original training set.

Note that these three settings don't violate the task formulation of few-shot link prediction in KGs. The statistics of NELL-One and Wiki-One are shown in Table 2.

We use two traditional metrics to evaluate different methods on these datasets, MRR and Hits@N. MRR is the mean reciprocal rank and Hits@N is the proportion of correct entities ranked in the top N in link prediction.

### 5.2 Implementation

During training, mini-batch gradient descent is applied with batch size set as 64 and 128 for NELL-One and Wiki-One respectively. We use Adam (Kingma and Ba, 2015) with the initial learning rate as 0.001 to update parameters. We set $\gamma = 1$ and $\beta = 1$. The number of positive and negative triples in query set is 3 and 10 in NELL-One and Wiki-One. Trained model will be applied on validation tasks each 1000 epochs, and the current model parameters and corresponding performance will be recorded, after stopping, the model that has the best performance on Hits@10 will be treated as final model. For number of training epoch, we use early stopping with 30 patient epochs, which means that we stop the training when the performance on Hits@10 drops 30 times continuously. Following GMatching, the embedding dimension of NELL-One is 100 and Wiki-One is 50. The

[1]The source code of experiments is available at https://github.com/AnselCmy/MetaR

| | MRR | | Hits@10 | | Hits@5 | | Hits@1 | |
|---|---|---|---|---|---|---|---|---|
| **NELL-One** | 1-shot | 5-shot | 1-shot | 5-shot | 1-shot | 5-shot | 1-shot | 5-shot |
| GMatching_RESCAL | .188 | – | .305 | – | .243 | – | .133 | – |
| GMatching_TransE | .171 | – | .255 | – | .210 | – | .122 | – |
| GMatching_DistMult | .171 | – | .301 | – | .221 | – | .114 | – |
| GMatching_ComplEx | .185 | .201 | .313 | .311 | .260 | .264 | .119 | .143 |
| GMatching_Random | .151 | – | .252 | – | .186 | – | .103 | – |
| MetaR (BG:Pre-Train) | .164 | .209 | .331 | .355 | .238 | .280 | .093 | .141 |
| MetaR (BG:In-Train) | **.250** | **.261** | **.401** | **.437** | **.336** | **.350** | **.170** | **.168** |
| **Wiki-One** | 1-shot | 5-shot | 1-shot | 5-shot | 1-shot | 5-shot | 1-shot | 5-shot |
| GMatching_RESCAL | .139 | – | .305 | – | .228 | – | .061 | – |
| GMatching_TransE | .219 | – | .328 | – | .269 | – | .163 | – |
| GMatching_DistMult | .222 | – | .340 | – | .271 | – | .164 | – |
| GMatching_ComplEx | .200 | – | .336 | – | .272 | – | .120 | – |
| GMatching_Random | .198 | – | .299 | – | .260 | – | .133 | – |
| MetaR (BG:Pre-Train) | **.314** | **.323** | **.404** | **.418** | **.375** | **.385** | **.266** | **.270** |
| MetaR (BG:In-Train) | .193 | .221 | .280 | .302 | .233 | .264 | .152 | .178 |

Table 4: Results of few-shot link prediction in NELL-One and Wiki-One. **Bold** numbers are the best results of all and underline numbers are the best results of GMatching. The contents of (bracket) after MetaR illustrate the form of datasets we use for MetaR.

sizes of two hidden layers in relation-meta learner are 500, 200 and 250, 100 for NELL-One and Wiki-One.

### 5.3 Results

The results of two few-shot link prediction tasks, including 1-shot and 5-shot, on NELL-One and Wiki-One are shown in Table 4. The baseline in our experiment is GMatching (Xiong et al., 2018), which made the first trial on few-shot link prediction task and is the only method that we can find as baseline. In this table, results of GMatching with different KG embedding initialization are copied from the original paper. Our MetaR is tested on different settings of datasets introduced in Table 3.

In Table 4, our model performs better with all evaluation metrics on both datasets. Specifically, for 1-shot link prediction, MetaR increases by 33%, 28.1%, 29.2% and 27.8% on MRR, Hits@10, Hits@5 and Hits@1 on NELL-One, and 41.4%, 18.8%, 37.9% and 62.2% on Wiki-One, with average improvement of 29.53% and 40.08% respectively. For 5-shot, MetaR increases by 29.9%, 40.5%, 32.6% and 17.5% on MRR, Hits@10, Hits@5 and Hits@1 on NELL-One with average improvement of 30.13%.

Thus for the first question we want to explore, the results of MetaR are no worse than GMatching, indicating that MetaR has the capability of accomplishing few-shot link prediction. In parallel, the impressive improvement compared with GMatching demonstrates that the key idea of MetaR, transferring relation-specific meta information from support set to query set, works well on few-shot link prediction task.

| Ablation Conf. | BG:Pre-Train | BG:In-Train |
|---|---|---|
| *standard* | .331 | .401 |
| *-g* | .234 | .341 |
| *-g -r* | .052 | .052 |

Table 5: Results of ablation study on Hits@10 of 1-shot link prediction in NELL-One.

Furthermore, compared with GMatching, our MetaR is independent with background knowledge graphs. We test MetaR on 1-shot link prediction in partial NELL-One and Wiki-One which discard the background graph, and get the results of 0.279 and 0.348 on Hits@10 respectively. Such results are still comparable with GMatching in fully datasets with background.

### 5.4 Ablation Study

We have proved that relation-specific meta information, the key point of MetaR, successfully contributes to few-shot link prediction in previous section. As there are two kinds of relation-specific meta information in this paper, relation meta and gradient meta, we want to figure out how these two kinds of meta information contribute to the performance. Thus, we conduct an ablation study with three settings. The first one is our complete MetaR method denoted as *standard*. The second one is removing the gradient meta by transferring un-updated relation meta directly from support set to query set without updating it via gradient meta, denoted as *-g*. The third one is removing the relation meta further which makes the model rebase to a simple TransE embedding model, denoted as *-g -r*. The result under the third setting is copied from Xiong et al. (2018). It uses the triples from background graph, training tasks and one-shot training triples from validation/test set, so it's neither *BG:Pre-Train* nor *BG:In-Train*. We conduct the ablation study on NELL-one with metric Hit@10 and results are shown in Table 5.

Table 5 shows that removing gradient meta decreases 29.3% and 15% on two dataset settings, and further removing relation meta continuous decreases the performance with 55% and 72% compared to the *standard* results. Thus both relation meta and gradient meta contribute significantly and relation meta contributes more than gradient meta. Without gradient meta and relation meta, there is no relation-specific meta information transferred in the model and it almost doesn't work. This also illustrates that relation-specific meta information is important and effective for few-shot link prediction task.

### 5.5 Facts That Affect MetaR's Performance

We have proved that both relation meta and gradient meta surely contribute to few-shot link prediction. But is there any requirement for MetaR to ensure the performance on few-shot link prediction? We analyze this from two points based on the results, one is the sparsity of entities and the other is the number of tasks in training set.

**The sparsity of entities** We notice that the best result of NELL-One and Wiki-One appears in different dataset settings. With NELL-One, MetaR performs better on *BG:In-Train* dataset setting, while with Wiki-One, it performs better on *BG:Pre-Train*. Performance difference between two dataset settings is more significant on Wiki-One.

Most datasets for few-shot task are sparse and the same with NELL-One and Wiki-One, but the entity sparsity in these two datasets are still significantly different, which is especially reflected in the proportion of entities that only appear in one triple in training set, 82.8% and 37.1% in Wiki-One and NELL-One respectively. Entities only have one triple during training will make MetaR unable to learn good representations for them, because entity embeddings heavily rely on triples related to them in MetaR. Only based on one triple, the learned entity embeddings will include a lot of bias. Knowledge graph embedding method can learn better embeddings than MetaR for those one-shot entities, because entity embeddings can be corrected by embeddings of relations that connect to it, while they can't in MetaR. This is why the best performance occurs in *BG:Pre-train* setting on Wiki-One, pre-train entity embeddings help MetaR overcome the low-quality on one-shot entities.

**The number of tasks** From the comparison of MetaR's performance between with and without background dataset setting on NELL-One, we find that the number of tasks will affect MetaR's performance significantly. With *BG:In-Train*, there are 321 tasks during training and MetaR achieves 0.401 on Hits@10, while without background knowledge, there are 51, with 270 less, and MetaR achieves 0.279. This makes it reasonable that why MetaR achieves best performance on *BG:In-Train* with NELL-One. Even NELL-One has 37.1% one-shot entities, adding background knowledge into dataset increases the number of training tasks significantly, which complements the sparsity problem and contributes more to the task.

Thus we conclude that both the sparsity of entities and number of tasks will affect performance of MetaR. Generally, with more training tasks, MetaR performs better and for extremely sparse dataset, pre-train entity embeddings are preferred.

## 6 Conclusion

We propose a meta relational learning framework to do few-shot link prediction in KGs, and we design our model to transfer relation-specific meta information from support set to query set. Specif-

ically, using relation meta to transfer common and important information, and using gradient meta to accelerate learning. Compared to GMatching which is the only method in this task, our method MetaR gets better performance and it is also independent with background knowledge graphs. Based on experimental results, we analyze that the performance of MetaR will be affected by the number of training tasks and sparsity of entities. We may consider obtaining more valuable information about sparse entities in few-shot link prediction in KGs in the future.

## Acknowledgments

We want to express gratitude to the anonymous reviewers for their hard work and kind comments, which will further improve our work in the future. This work is funded by NSFC 91846204/61473260, national key research program YS2018YFB140004, and Alibaba CangJingGe(Knowledge Engine) Research Plan.

## References

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1247–1250. ACM.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

Antoine Bordes, Jason Weston, and Nicolas Usunier. 2014. Open question answering with weakly supervised embedding models. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 165–180. Springer.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka, and Tom M Mitchell. 2010. Toward an architecture for never-ending language learning. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. 2015. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2.

Yoonho Lee and Seungjin Choi. 2018. Gradient-based meta-learning with learned layerwise metric and subspace. In *International Conference on Machine Learning*, pages 2933–2942.

Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015a. Modeling relation paths for representation learning of knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 705–714.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015b. Learning entity and relation embeddings for knowledge graph completion. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. 2018. A simple neural attentive meta-learner. In *International Conference on Learning Representations*.

Tsendsuren Munkhdalai and Hong Yu. 2017. Meta networks. In *International Conference on Machine Learning*, pages 2554–2563.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *International Conference on Machine Learning*, volume 11, pages 809–816.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. 2016. Meta-learning with memory-augmented neural networks. In *International Conference on Machine Learning*, pages 1842–1850.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, pages 2071–2080.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638.

Denny Vrandei and Markus Krtzsch. 2014. Wikidata: A free collaborative knowledge base. *Communications of the ACM*, 57:78–85.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.

Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2016. Representation learning of knowledge graphs with hierarchical types. In *IJCAI*, pages 2965–2971.

Wenhan Xiong, Mo Yu, Shiyu Chang, Xiaoxiao Guo, and William Yang Wang. 2018. One-shot relational learning for knowledge graphs. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1980–1990. Association for Computational Linguistics.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *International Conference on Learning Representations*.

Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 353–362. ACM.

Ningyu Zhang, Shumin Deng, Zhanlin Sun, Guanying Wang, Xi Chen, Wei Zhang, and Huajun Chen. 2019a. Long-tail relation extraction via knowledge graph embeddings and graph convolution networks. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3016–3025.

Wen Zhang, Bibek Paudel, Liang Wang, Jiaoyan Chen, Hai Zhu, Wei Zhang, Abraham Bernstein, and Huajun Chen. 2019b. Iteratively learning embeddings and rules for knowledge graph reasoning. In *The World Wide Web Conference*, pages 2366–2377. ACM.

Wen Zhang, Bibek Paudel, Wei Zhang, Abraham Bernstein, and Huajun Chen. 2019c. Interaction embeddings for prediction and explanation in knowledge graphs. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 96–104. ACM.

# Distributionally Robust Language Modeling

**Yonatan Oren*[1] Shiori Sagawa*[1] Tatsunori B. Hashimoto*[1,2] Percy Liang[1]**
**(* equal contribution)**
[1]Stanford Computer Science [2]Stanford Statistics
{yonatano,thashim}@stanford.edu {ssagawa,pliang}@cs.stanford.edu

## Abstract

Language models are generally trained on data spanning a wide range of topics (e.g., news, reviews, fiction), but they might be applied to an a priori unknown target distribution (e.g., restaurant reviews). In this paper, we first show that training on text outside the test distribution can degrade test performance when using standard maximum likelihood (MLE) training. To remedy this without the knowledge of the test distribution, we propose an approach which trains a model that performs well over a wide range of potential test distributions. In particular, we derive a new distributionally robust optimization (DRO) procedure which minimizes the loss of the model over the *worst-case* mixture of topics with sufficient overlap with the training distribution. Our approach, called *topic conditional value at risk (topic CVaR)*, obtains a 5.5 point perplexity reduction over MLE when the language models are trained on a mixture of Yelp reviews and news and tested only on reviews.

## 1 Introduction

Large-scale language modeling plays a central role in both text generation (Sordoni et al., 2015; Nallapati et al., 2016) and unsupervised pre-training (Vaswani et al., 2013; Dai and Le, 2015; McCann et al., 2017; Peters et al., 2018; Devlin et al., 2018; Radford et al., 2018). In both settings, a single language model is trained on a large corpus containing a range of topics (e.g. news, fiction, and reviews). This language model is then applied in many different tasks, each with a specific test distribution (e.g., analyzing the sentiment of restaurant reviews). Can we train a single general-purpose language model that works across a wide range of potential test distributions?

In this work, we first demonstrate that standard maximum likelihood training on a large, heterogeneous dataset can fail to achieve this goal.

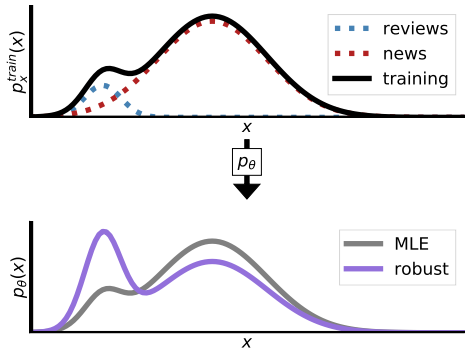


**Figure 1.** Illustration of a training corpus as a density (black) with mostly news stories (red) and a small number of restaurant reviews (blue). The standard MLE model (gray) reflects the underlying data and assigns little weight to reviews, and thus performs poorly on reviews. A more robust model should try to equalize the weight across all topics so that it can perform well regardless of which topics appear at test time.

While more data is generally better, the presence of text outside the target distribution actually *degrades* performance on a target test distribution. For example, a language model trained on Yelp reviews achieves a perplexity of 32, and this perplexity increases to 43 when trained on a mixture of 10% Yelp and 90% newswire sentences from the One Billion Word Benchmark (Chelba et al., 2013). Performance degrades because existing maximum likelihood estimation (MLE) objectives tend to emphasize model performance on more common sentences and topics at the expense of infrequent ones (Figure 1).

While the above performance degradation can be mitigated by fine-tuning and domain adaptation techniques (Shimodaira, 2000; Quiñonero-Candela et al., 2009; Daume III, 2007; Ben-David et al., 2010; Blitzer et al., 2011; Pryzant et al., 2017; Ganin and Lempitsky, 2015; Tzeng et al., 2014), these methods require knowing the test distribution and training a separate model specific to each target distribution. Instead, we aim to train a



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4227–4237, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

*single* model that performs well across many unknown test distributions.

In order to do this, we will train a model that performs uniformly well over an entire family of potential test distributions. Since we cannot expect to do well on all possible test distributions, we consider the *subpopulation shift* setting, in which the test distribution is a subpopulation of the training distribution, and seek good performance across all such test distributions (e.g. Yelp reviews in a Yelp-newswire mixture). [1] In other words, adding data from topics outside the test topics should not hurt. It seems reasonable to protect against subpopulation shifts, intuitively because large-scale data collection schemes are designed to cover a diverse array of topics as a way to generalize to potential test distributions.

We train a model that performs well over all subpopulations by minimizing the risk for the *worst-case* subpopulation, following the distributionally robust optimization (DRO) literature (Ben-Tal et al., 2013). While an existing DRO framework called the conditional value at risk (CVaR) ensures uniformly good performance across subpopulations (Rockafellar and Uryasev, 2000; Duchi and Namkoong, 2018), we demonstrate that naïvely applying this approach to language modeling fails due to three challenges. First, the existing CVaR approach is too conservative because it considers robustness to *arbitrary* subpopulations. Such worst-case subpopulations are attained by adversarially choosing the hardest, most unusual sentences. Instead, we propose to consider *meaningful* subpopulations, defined by topics in a corpus (Hu et al., 2018). Second, applying CVaR directly to log loss results in a loss which is biased towards topics with high entropy, instead of those for which the model performs poorly relative to what is possible. We correct this by introducing a new *baselined* loss function which measures losses relative to the entropy of each topic. Finally, existing optimization algorithms for CVaR are either inapplicable to topic-based robustness sets or unscalable because they require batch optimization. We develop a scalable online algorithm which identifies the worst-performing topics at each iteration and upweights examples from those topics.

With these methodological improvements, we demonstrate that our approach, *topic CVaR*, improves robustness against subpopulation shifts. Topic CVaR reduces perplexity on the Yelp review corpus by 5.5 points compared to MLE when trained on the Yelp-One Billion Word Benchmark mixture from before. We also show improved robustness even when the shift is not strictly a subpopulation shift. Topic CVaR also achieves a 4 point perplexity reduction on a test distribution (TripAdvisor hotel reviews) that is similar to, but not strictly a subpopulation of the training distribution (Yelp and newswire text).

[1] The subpopulation assumption refers to overlaps in *distributions*, rather than individual examples. Our assumptions do not require overlap in the training and test data.

## 2 Problem Statement

Our goal is to learn a language model $p_\theta$ based on sentences sampled from the training distribution $x \sim p_{\text{x}}^{\text{train}}$, such that $p_\theta$ performs well on unknown test distributions $p_{\text{x}}^{\text{test}}$.

Language models $p_\theta$ are generally trained to approximate $p_{\text{x}}^{\text{train}}$ by minimizing the KL divergence $\text{KL}\big(p_{\text{x}}^{\text{train}} \,\big\|\, p_\theta\big)$ via maximum likelihood estimation (MLE),

$$\inf_\theta \mathbb{E}\left[-\log p_\theta(x)\right]. \tag{1}$$

When $p_{\text{x}}^{\text{test}} = p_{\text{x}}^{\text{train}}$, classical statistical theory guarantees that a model trained via MLE performs well on the test distribution given sufficient data. However, when $p_{\text{x}}^{\text{test}}$ is not identical to $p_{\text{x}}^{\text{train}}$, MLE can perform poorly no matter how much data is observed. This is because the test set might consist solely of sentences from topics that are infrequent during training, to which MLE would assign low probabilities.

To illustrate this point, consider the toy example drawn in Figure 2. In this example, the training distribution $p_{\text{x}}^{\text{train}}$ is a multinomial distribution over six possible sentences A–F, with two from reviews and four from news. Sentence F is ungrammatical and thus has an extremely low probability. The training distribution includes $10\%$ reviews and $90\%$ news, whereas the test distribution could be all reviews, all news, or a mixture. MLE assigns low probabilities to any review and thus performs poorly when evaluated solely on reviews. To be robust, we intuitively need a more conservative objective that encourages models to assign higher probabilities to rare but valid sentences.

In order to achieve this, we want to learn a model $p_\theta$ which performs well in situations where $p_{\text{x}}^{\text{train}} \neq p_{\text{x}}^{\text{test}}$ for a large set of potential test distributions $\mathcal{P}$, termed the uncertainty set. By training

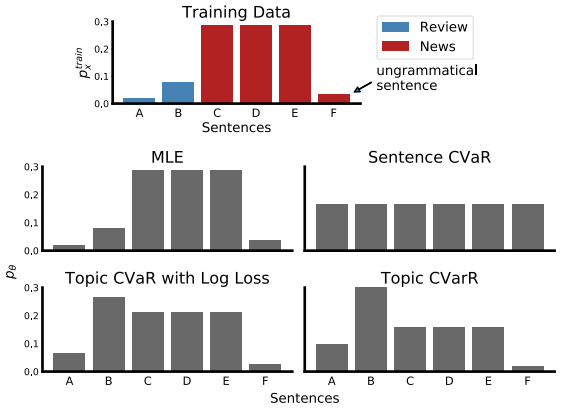


**Figure 2.** Toy example of a multinomial distribution over six sentences (top). Different panels illustrate models learned by different training procedures. MLE fits common topics (news) at the expense of rare ones (reviews). Sentence CVaR is too conservative, overemphasizing the ungrammatical sentence. Topic CVaR with log loss overemphasizes difficult topics (news) over easy ones (review). Topic CVaR (with baselining) balances the weight assigned to each topics, as desired.

a model that performs well on all distributions in the uncertainty set $\mathcal{P}$, we can ensure good test performance as long as $p_{\text{x}}^{\text{test}} \in \mathcal{P}$.

More formally, this approach falls under the framework of distributionally robust optimization (DRO) (Ben-Tal et al., 2013). With DRO, we optimize a model for loss $\ell$ and a set of potential test distributions $\mathcal{P}$ by minimizing the risk under the *worst-case* distribution in $\mathcal{P}$,

$$\sup_{p_{\text{x}} \in \mathcal{P}} \mathbb{E}_{p_{\text{x}}}[\ell(x;\theta)]. \tag{2}$$

Observe that the above worst-case objective does not depend on the unknown quantity $p_{\text{x}}^{\text{test}}$. The objective also upper bounds the test risk for all $p_{\text{x}}^{\text{test}} \in \mathcal{P}$ as

$$\mathbb{E}_{p_{\text{x}}^{\text{test}}}[\ell(x;\theta)] \leq \sup_{p_{\text{x}} \in \mathcal{P}} \mathbb{E}_{p_{\text{x}}}[\ell(x;\theta)], \tag{3}$$

so optimizing the above objective gives guarantees on test performance whenever $p_{\text{x}}^{\text{test}} \in \mathcal{P}$.

DRO provides a conceptually appealing framework for learning under train-test mismatch. However, it crucially depends on both the choice of uncertainty set $\mathcal{P}$ and loss $\ell$, and we will discuss these choices in the next section.

## 3 Robust Language Modeling

We will begin by applying standard distributionally robust optimization approaches to the log loss (Section 3.1), and showing that this naïve approach suffers from two drawbacks:

1. Existing DRO uncertainty sets $\mathcal{P}$ are too conservative.
2. The log loss overemphasizes topics with inherently high entropy.

These drawbacks will motivate our development of a new approach we call topic CVaR, which addresses these two problems (Sections 3.2 and 3.3).

### 3.1 Robustness to arbitrary subpopulations

Observing that MLE is not robust because it assigns low probabilities (i.e. incurs high losses) on rare sentences, we might initially try to define $\mathcal{P}$ as individual training examples to ensure low loss on *all* data points. However, this is far too conservative, since the worst-case distribution would consist of exactly one data point. Therefore, we may want to optimize a slightly more realistic uncertainty set consisting of all sufficiently large subpopulations of the training distribution.

Minimizing losses over all subpopulations of the training distribution can be formulated as a type of distributionally robust optimization (DRO) problem (Duchi and Namkoong, 2018), which has been used to regularize models (Duchi and Namkoong, 2016), defend against adversarial examples (Sinha et al., 2018), and improve the fairness of models (Hashimoto et al., 2018).

One type of distributionally robust loss is known as conditional value at risk (CVaR) which guarantees low losses on all $\alpha$-fraction subpopulations of the training distribution (Rockafellar and Uryasev, 2000). This corresponds to defining the uncertainty set $\mathcal{P}$ as all sentence distributions that are $\alpha$-*covered* by $p_{\text{x}}^{\text{train}}$,

$$\mathcal{P}_{\text{x}}^{\alpha} := \{p_{\text{x}} : \alpha p_{\text{x}}(x) \leq p_{\text{x}}^{\text{train}}(x) \ \ \forall x\}. \tag{4}$$

This is equivalent to defining $\mathcal{P}_{\text{x}}^{\alpha}$ as the set of $p_{\text{x}}$ which fulfills $p_{\text{x}}^{\text{train}} = \alpha p_{\text{x}} + (1-\alpha) p_{\text{x}}^{\text{other}}$ for some distribution $p_{\text{x}}^{\text{other}}$.

To achieve low loss on all possible test distributions in $\mathcal{P}_{\text{x}}^{\alpha}$, we minimize the expected loss under the worst-case distribution,

$$\sup_{p_{\text{x}} \in \mathcal{P}_{\text{x}}^{\alpha}} \mathbb{E}_{x \sim p_{\text{x}}}[\ell(x;\theta)]. \tag{5}$$

For the remainder of the paper, we will refer to this approach as *sentence CVaR*, highlighting the fact that it considers robustness over arbitrary sets of sentences. It intuitively encourages uniform performance across all subpopulations of sentences

by downweighting sentences with low loss, and upweighting sentences with high loss.

Because sentence CVaR considers *arbitrary* groups of examples, it can be too conservative in our problem setting. While sentence CVaR can prevent modeling common sentences at the cost of rare ones, it can also encourage modeling invalid sentences at the expense of valid ones. Returning to our example in Figure 2 with $\ell(x;\theta) = -\log p_\theta(x)$ , sentence CVaR with for sufficiently low $\alpha$ achieves perfectly uniform performance. It equalizes likelihoods across all sentences, which unfortunately also results in high probabilities assigned to the ungrammatical sentence F.

### 3.2 Robustness over Topics

Sentence CVaR is too conservative since it allows for arbitrary groups — including ones consisting of purely invalid sentences. To remedy this, we will optimize models for all *meaningful* subpopulations instead of *arbitrary* ones.

One way to achieve this is through robustness over topics, rather than individual examples. For example, a news corpus often contains a variety of topics (politics, business, opinion, food) and a test corpus may contain these topics with different proportions. A robust language model should perform well on a wide range of topic mixtures without taking the topic identity as an input.

Formally, we posit that each sentence $x$ belongs to some latent topic $z$, which has a sentence distribution $p_{\mathrm{x}|z}$. We want our models to be robust to shifts in the topic distribution, where we have $z \sim p_{\mathrm{z}}^{\text{train}}$ and $z \sim p_{\mathrm{z}}^{\text{test}}$. In this case, we can define a natural uncertainty set for CVaR, defined over latent topics rather than individual examples. Extending the definition of $\alpha$-covered distributions to topics, we have the set

$$\mathcal{P}_{\mathrm{z}}^{\alpha} := \{p_{\mathrm{z}} : \alpha p_{\mathrm{z}}(z) \le p_{\mathrm{z}}^{\text{train}}(z) \ \ \forall z\} \tag{6}$$

and the objective is the expected loss under the worst-case topic distribution,

$$\sup_{p_{\mathrm{z}} \in \mathcal{P}_{\mathrm{z}}^{\alpha}} \mathbb{E}_{z\sim p_{\mathrm{z}}} \left[ \mathbb{E}_{x \sim p_{\mathrm{x}|z}} \left[ \ell(x;\theta) \right] \right]. \tag{7}$$

This objective intuitively encourages uniform loss across topics by upweighting topics incurring high losses and downweighting topics with low losses, while keeping the conditional distribution of sentences given a topic constant.

### 3.3 Baselined Loss Function

Recall that DRO depends critically on the choice of uncertainty set and loss function. Having specified the uncertainty set, we now turn to the choice of loss $\ell(x;\theta)$. While the log loss $\ell(x;\theta) = -\log p_\theta(x)$ is the standard choice in language modeling, we show that this approach has a flaw in the robust setting and propose a corrected loss.

**Log Loss.** Using log loss on CVaR encourages uniform *absolute* log-likelihoods across topics even if some topics are much harder than others. For example, consider a model which performs nearly optimally on difficult topics and highly suboptimally on easy topics. Since log loss measures absolute performance, it would force the model to focus on the difficult topic *even if the model can't improve further on this topic*. In the example in Figure 2, news is emphasized over reviews because news has higher entropy and thus higher difficulty. Empirically, we observe that log loss with CVaR forces the models to focus almost entirely on the difficult topics such as long news stories.

**Baselined Loss.** We now propose a new baselined loss, which encourages uniform *relative* performance across topics. We refer to our approach with the baselined loss as *topic CVaR*.

The baselined loss function $\ell(x, z; \theta) = \log p_{\mathrm{x}|z}(x \mid z) - \log p_\theta(x)$ evaluates the performance of the model relative to the best possible model for the topic, $\log p_{\mathrm{x}|z}(x \mid z)$. Although we do not observe $\log p_{\mathrm{x}|z}(x \mid z)$, we will show later in section 4.2 that we can estimate sufficient statistics of $\log p_{\mathrm{x}|z}(x \mid z)$ that allow us to compute the baselined loss. By using baselined loss, we intuitively encourage models to perform as well as it can on each topic while making optimal trade-offs among topics.

Plugging the baselined loss into the robust objective (7), the optimization problem is

$$\sup_{p_{\mathrm{z}} \in \mathcal{P}_{\mathrm{z}}^{\alpha}} \mathbb{E}_{z\sim p_{\mathrm{z}}} \left[ \mathbb{E}_{x \sim p_{\mathrm{x}|z}} \left[ \log p_{\mathrm{x}|z}(x \mid z) - \log p_\theta(x) \right] \right], \tag{8}$$

which can be simplified to

$$\sup_{p_{\mathrm{z}} \in \mathcal{P}_{\mathrm{z}}^{\alpha}} \mathbb{E}_{z\sim p_{\mathrm{z}}} \left[ \mathrm{KL}\big(p_{\mathrm{x}|z} \,\big\|\, p_\theta\big) \right]. \tag{9}$$

Topic CVaR thus minimizes the per-topic KL divergences, and this interpretation fits nicely with

a general goal of training $p_\theta$ that matches the test distribution. Unlike in the MLE case, minimizing the KL is not equivalent to minimizing the log loss. In MLE, minimizing $\mathrm{KL}(p_{\mathrm{x}}^{\text{train}} \| p_\theta) = \mathbb{E}\left[\log p_{\mathrm{x}}^{\text{train}}(x) - \log p_\theta(x)\right]$ is equivalent to minimizing the log loss because $\log p_{\mathrm{x}}^{\text{train}}(x)$ can be treated as a constant. However, in topic CVaR, the analogous baseline entropy term $\log p_{\mathrm{x}|z}(x \mid z)$ depends on $z$ and thus is not a constant with respect to the outer supremum.

In the running toy example (Figure 2), topic CVaR results in robust models that perform relatively well on both news and reviews. The resulting model is a mixture of news and review distribution with equal weights on the two topics.

In summary, topic CVaR contains two improvements over existing DRO approaches: using the latent topic distribution $p_{\mathrm{z}}^{\text{train}}$ to specify the uncertainty set and defining the baselined loss. In the following section, we will describe an algorithm which optimizes this topic CVaR objective.

## 4 Algorithm

We now operationalize the principles in the previous section, specifying (i) how we choose topics (Section 4.1), (ii) how we estimate the baseline (Section 4.2), and (iii) how to efficiently optimize the robust objective (7) (Section 4.3).

### 4.1 Identifying Topics

The topic CVaR objective requires topic assignments $z$ for each sentence in order to define the uncertainty set $\mathcal{P}$. Since the topics determine the set of $p_{\mathrm{x}}^{\text{test}}$ distribution for which the model performs well, we seek topics whose subpopulation shifts capture realistic potential test settings.

We use latent Dirichlet allocation (LDA) (Blei et al., 2003) to cluster the sentences into latent topics. LDA assigns each word in a sentence to a topic, and we assign each sentence to the topic with highest total posterior probability.

### 4.2 Estimating Baselined Losses

Recall that topic CVaR uses KL-divergence as the loss term (Eq. (9)),

$$\mathrm{KL}\big(p_{\mathrm{x}|z} \,\big\|\, p_\theta\big) := \mathbb{E}_{p_{\mathrm{x}|z}}[\log p_{\mathrm{x}|z}(x \mid z)] - \mathbb{E}_{p_{\mathrm{x}|z}}[\log p_\theta(x)].$$

While we can estimate the log loss term $\mathbb{E}[\log p_\theta(x)]$ from samples, the entropy term $H(X \mid Z = z) := \mathbb{E}_{p_{\mathrm{x}|z}}[-\log p_{\mathrm{x}|z}(x \mid z)]$ is not something we can easily estimate.

We thus propose to estimate the entropies $H(X \mid Z = z)$ by fitting a *baseline model* $p_\beta$ for each topic, and computing $H_\beta(X \mid Z = z) := \mathbb{E}_{p_{\mathrm{x}|z}}[-\log p_\beta(x \mid z)]$.[2] In practice, we use a bigram model, which was fast enough to scale and worked sufficiently well in experiments.

### 4.3 Online Optimization of topic CVaR

No scalable, online algorithm exists for optimizing the topic CVaR objective. Many DRO problems admit efficient batch optimization procedures based on Lagrangian duality (Duchi et al., 2016). However, this approach fails for topic CVaR, since the dual form requires exact computations rather than stochastic estimates of $\mathbb{E}_{p_{\mathrm{x}|z}}[-\log p_\theta(x)]$. Online algorithms for DRO exist (Namkoong and Duchi, 2016), but do not handle the nested maximization-expectation structure arising in topic CVaR (Eq. (7)).

Because of this, we develop an online optimization procedure for topic CVaR compatible with stochastic gradient descent methods. The topic CVaR problem is a two-player minimax game between the model parameter $\theta$ and the potential test distribution $p_{\mathrm{z}}$. Intuitively, $p_{\mathrm{z}}$ attempts to be the worst-case distribution and maximize the robust objective, while $\theta$ attempts to minimize the robust objective. The precise two-player minimax game is

$$\inf_\theta \sup_{p_{\mathrm{z}} \in \mathcal{P}_z^\alpha} \mathbb{E}_{z \sim p_{\mathrm{z}}}[L(z;\theta)], \tag{10}$$

where the expected loss for each $z$ (inner expectation) is $L(z;\theta) := \mathbb{E}_{x \sim p_{\mathrm{x}|z}}[\ell(x;\theta)]$.

In the above two-player game, the game proceeds in multiple rounds $t = 1, 2, \ldots$. At each round, the players select $p_{\mathrm{z}}{}^{(t)}$ and $\theta^{(t)}$. It is standard to interleave parameter updates between the two players in minimax optimization, and we describe the precise update rules in subsequent paragraphs. To carry out these updates, we keep track of an empirical estimate of the probability $p_{\mathrm{z}}^{\text{train}}(z)$ at each iteration $t$, which we refer to as $\hat{p}_{\mathrm{z}}^{\text{train}(t)}(z)$. We also keep track of the historical average of losses incurred for each topic so far, up to the current round $t$, which we call $\hat{L}^{(t)}(z;\theta^{(1:t)})$. Con-

[2] $H_\beta$ yields accurate solutions to the topic CVaR problem as long as they capture the entropy up to a constant (i.e. $H_\beta(X \mid Z = z) \approx H(X \mid Z = z) + c$)

cretely, $\hat{L}^{(t)}(z;\theta^{(1:t)})$ is computed as an average of $\{\ell(x^{(t')};\theta^{(t')}) : t' \in [t], z^{(t')} = z\}$.

At each iteration $t$, $p_{\mathrm{z}}$ is updated by selecting an optimal value with respect to historical losses up to the current iteration, loosely inspired by the "Be The Leader" algorithm. This results in the following update rule to $p_{\mathrm{z}}$,

$$p_{\mathrm{z}}{}^{(t)} = \operatorname*{argmax}_{p_z \in \mathcal{P}_z^{\alpha}} \mathbb{E}_{z \sim p_z}\left[\hat{L}^{(t)}(z;\theta^{(1:t)})\right]. \quad (11)$$

The above argmax can computed efficiently by ordering topics in the order of decreasing average loss, and assigning each topic either $\frac{\hat{p}_{\mathrm{z}}^{\mathrm{train}}(z)}{\alpha}$ or the probability left to be assigned, whichever is lower.[3]

We update $\theta$ with online gradient descent,

$$\theta^{(t)} = \theta^{(t-1)} - \epsilon \frac{p_{\mathrm{z}}{}^{(t)}(z^{(t)})}{\hat{p}_{\mathrm{z}}^{\mathrm{train}(t)}(z^{(t)})} \nabla \ell(x^{(t)};\theta^{(t-1)}),$$

where $\epsilon$ is the learning rate.

To give intuition for the two updates, first note that $\frac{p_{\mathrm{z}}{}^{(t)}(z)}{p_{\mathrm{z}}^{\mathrm{train}(t)}(z)} = \frac{1}{\alpha}$ on approximately $\alpha$ fraction of the data and this ratio acts as an indicator function which determines if an example is part of the worst-case set or not. If it is, we update the model and otherwise we ignore it.

## 5 Experiments

We demonstrate that topic CVaR improves maximum likelihood language models when $p_{\mathrm{x}}^{\mathrm{train}} \neq p_{\mathrm{x}}^{\mathrm{test}}$. Section 5.1 outlines the experimental setup while Section 5.2 shows the robustness improvements and analysis of topic CVaR.

### 5.1 Evaluation Details

**Datasets.** We use the following three corpora: the Yelp review corpus (YELP, (2017)), One Billion Word benchmark corpus (ONEBWORD), and the TripAdvisor Annotated Dataset (TRIPADV, Marcheggiani et al. (2014)).

We preprocess the corpora using SPACY (Honnibal and Johnson (2015)) by removing sentences with fewer than 10 characters, segmenting sentences, tagging named-entities, and replacing each entity with its corresponding OntoNotes tag.

[3]For example with $\alpha = 0.2$, $\hat{L}^{(t)} = [40, 30, 60]$, and $\hat{p}_{\mathrm{z}}^{\mathrm{train}} = [0.2, 0.8, 0.1]$, then $p_{\mathrm{z}}{}^{(t)} = [0.5, 0, 0.5]$.

**Vocabulary.** Our experiments will evaluate models using perplexity, which depends on the choice of vocabulary. To make perplexity comparable for models trained on different datasets, we use a single, fixed vocabulary formed by combining the most frequently occurring $10,000$ words in each corpus. All words in the mixtures which are not in the vocabulary ($1-3\%$ in our experiments) are replaced with a special *unk* token.

**Clustering.** To cluster sentences in the training set, we ran LightLDA (Yuan et al. (2015)) for 100 iterations with prior hyperparameters $\alpha = 0.1$, $\beta = 1.0$ and 2 Metropolis-Hastings steps. We set the model to find 10 topics, as this resulted in stable clusters consisting of semantically similar sentences.

**Models.** Our models are Transformer (Vaswani et al., 2017) based language models trained using the FAIRSEQ sequence-to-sequence toolkit (2017). We use the same model architecture, optimizers, and hyperparameters for both MLE and CVaR. For both models, we use Nesterov's accelerated gradient descent, a fixed learning rate of 0.01, minibatch size of 500 sentences, and 30k minibatches (corresponding to 100 epochs on the YELP corpus). These values were derived by tuning a MLE model trained on the YELP data and tested on the YELP dev set.

**Hyperparameters** Topic CVaR can be unstable at small $\alpha$ values due to the fact that we are optimizing for worst-case errors. Because of this, we make three small but important modifications to the algorithm. (i) We use $\alpha = 0.2$ to estimate models for $\alpha^* < 0.2$, as small $\alpha$s can cause gradients to become unstable; (ii) we set a minimum $p_{\mathrm{z}}(z)/p_{\mathrm{z}}^{\mathrm{train}}(z)$ value of 0.1; and (iii) we compute historical losses using exponentially weighted moving averages. With these modifications, the model reliably converges to similar validation losses.

### 5.2 Language Model Robustness

We seek to assess the performance of MLE and CVaR models under subpopulation shift. In order to do this, we train language models on various mixtures of YELP and ONEBWORD corpora and evaluate the models on a held-out set of YELP sentences.

We will construct a training corpus, whose distribution $\alpha^*$-*covers* the test distribution (i.e. $\alpha^*$

fraction of the training distribution corresponds to the Yelp distribution). In this case, we expect that topic CVaR with $\alpha = \alpha^*$ to perform well since the test set exactly fulfills the subpopulation shift assumption.

To form a training corpus, whose distribution $\alpha^*$-covers the YELP distribution, we mix a fixed set of 500,000 sentences from YELP training subset with $500,000(1 - \alpha^*)/\alpha^*$ sentences from ONEBWORD. This results in a dataset where $\alpha^*$ of the training data comes from YELP. The test corpus is composed of sentences from the YELP test subset, with no sentence overlap with the training corpora. Since the absolute number of YELP samples in the training corpora remains constant across different values of $\alpha^*$, we expect that a model which is robust to added nuisance data will perform equally well on a YELP-only test set, even as the mixture proportion of ONEBWORD samples in the training corpus increases.

**Oracle model.** We estimate the *oracle* performance of a robust language model as running topic CVaR where the topic $z = \{\text{YELP}, \text{ONEBWORD}\}$ and the topic assignments use the ground truth corpus identities rather than a clustering algorithm. In this case, when $\alpha^* = \alpha$ we are directly minimizing the worst-case baselined test loss over both YELP and ONEBWORD.

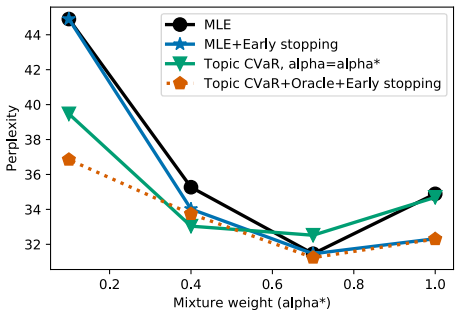


**Figure 3.** Topic CVaR (green) provides substantial improvements in perplexity compared to MLE (black and blue) as the amount of train-test mismatch increases ($\alpha^* \to 0$). This performance is close to the oracle performance which uses ground truth corpus labels and early stopping (orange).

**Topic CVaR improves robustness over MLE.** Using the YELP-ONEBWORD mixtures, we evaluate the robustness of topic CVaR and MLE to added nuisance data. We find that with no nuisance data, the MLE model matches the topic CVaR model (Figure 3 $\alpha^* = 1.0$). As we add data from ONEBWORD and $\alpha^*$ decreases to 0.7, we find some *positive transfer* effects where the increased data from the ONEBWORD corpus improves the performance on Yelp. However, as the fraction of nuisance data grows further and $\alpha^*$ drops below 0.4 the MLE models suffer large increases in perplexity, incurring up to 10 additional points of perplexity. Early stopping according to validation perplexity on YELP does not improve this substantially beyond the basic MLE model (blue star). On the other hand, applying topic CVaR with $\alpha^* = \alpha$ provides substantial boosts to language model performance for small $\alpha^*$, with nearly no loss of performance for large $\alpha^*$ (green triangle). Finally, we find that the topic CVaR method we propose is close to the best possible *oracle* performance.

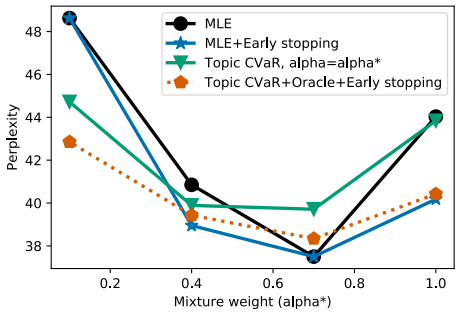


**Figure 4.** The robustness improvements from topic CVaR (black vs green and orange) apply even when the test set (TRIPADV reviews) is not a subpopulation shift from the training set (YELP and ONEBWORD).

**Topic CVaR robustness beyond subpopulation shift.** The prior YELP-ONEBWORD experiment showed that topic CVaR is more robust than MLE under subpopulation shift.

We now explore the more realistic setting in which the test distribution is not a subpopulation shift, but merely "similar" to the training distribution. We do this by testing the same model on the TRIPADV hotel review corpus. The hotel and restaurant review distributions are similar (i.e. they both frequently mention service) but differ in that hotels reviews often mention the location and room, while restaurant reviews often mention food items.

We find a similar result consistent with the earlier subpopulation shift experiment (Figure 4). The MLE model performance degrades rapidly between $\alpha^* = 0.7$ and 0.1, while topic CVaR substantially reduces this degradation. This suggests that topic CVaR models provide robustness bene-

fits in real-world settings where the topic overlaps are not exact, and the subpopulation shift assumption no longer holds.

**Ablations.** Topic CVaR extends the standard CVaR objective in two ways: the use of topics and the use of a baseline. We investigate the effect of these choices via an ablation experiment. Removing the topic structure results in dramatic loss of performance for our models: the perplexity exceeds 80 with $\alpha = 0.2$ for all $\alpha^*$. This is because the worst case group can consist of solely of disfluent sentences that do not match any real test distribution. If we remove the baseline, the resulting model is not completely degenerate, but it is not as robust as $\alpha^*$ decreases (Figure 5, teal). This is because ONEBWORD is a higher entropy corpus than YELP, and forcing the model to achieve equal *absolute* losses causes the model to focus nearly entirely on ONEBWORD, resulting in low YELP performance.

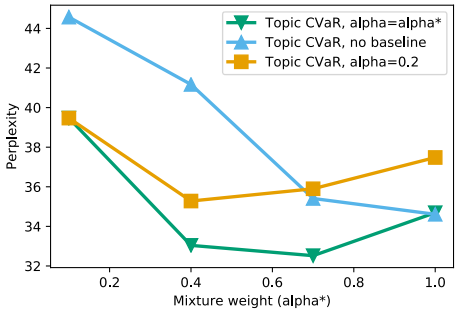


**Figure 5.** The robustness of topic CVaR degrades when the baseline is removed (teal), but is resistant to being over-conservative in choosing $\alpha$ (yellow).

**Choice of $\alpha$.** Since the true train-test overlap $\alpha^*$ is not always known, we cannot always set our hyperparameter $\alpha$ equal to $\alpha^*$. We find that selecting suboptimal values of $\alpha$ degrades perplexity between 2–3 points depending on $\alpha^*$. Figure 5 shows that setting $\alpha$ to the most conservative choice of 0.2 outperforms MLE on small $\alpha^*$ while incurring only 2 points of perplexity loss over MLE at $\alpha^* = 1.0$. Figure 6 further demonstrates that when $\alpha^* = 0.1$, any choice of $\alpha$ outperforms MLE, and incorrectly selecting $\alpha$ seems to incur a linear penalty in perplexity.

**Error analysis and examples.** Evaluating both models trained with $\alpha^* = 0.1$ on both the YELP and ONEBWORD test sets, we find that topic CVaR assigns higher probabilities (and therefore

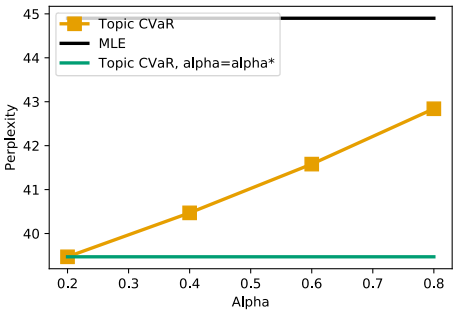


**Figure 6.** Topic CVaR outperforms MLE in the $p_x^{\text{train}} \neq p_x^{\text{test}}$ setting ($\alpha^* = 0.1$) for any small $\alpha$ (x-axis). The performance degradation is linear, implying topic CVaR is robust to small errors in the choice of $\alpha$.

incurs lower losses) on sentences from Yelp (Figure 7, top right). We also see that MLE does particularly well on low loss examples (bottom left) while topic CVaR does well on high-loss ones (top right) as we might expect from optimizing the worst-case losses.

Examining examples from the YELP test set (Table 1), we identify examples which have substantially higher probabilities under MLE than topic CVaR (left column) and vice versa (right column). These examples show that topic CVaR performs well by assigning high probabilities to stereotypical YELP sentences that discuss food and service, while MLE performs better on sentences about accidents and locations. These examples are consistent with the observation that topic CVaR assigns higher probabilities to typical YELP sentences and thus has lower perplexity, while the MLE model has high perplexity since it assigns probabilities to YELP sentences primarily based on their similarity to examples from ONEBWORD.

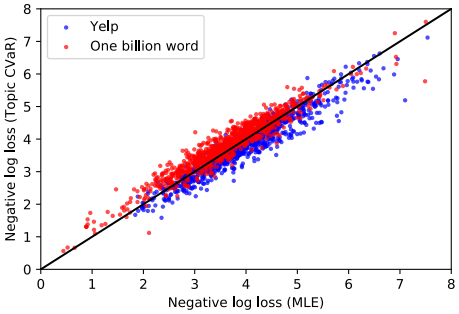


**Figure 7.** Log losses for sentences (points) from YELP (blue) and ONEBWORD (red) under topic CVaR (y-axis) and MLE (x-axis). Topic CVaR performs well on YELP and infrequent sentences (top right). MLE performs better on frequent sentences from ONEBWORD (bottom left).

| $p_{\text{MLE}} > p_{\text{CVaR}}$ | $p_{\text{CVaR}} > p_{\text{MLE}}$ |
|---|---|
| my girlfriend had an awful accident that hurt her leg & ankle which resulted in a fire and rescue ride | huge servings, so plenty for leftovers. |
| the address [PERSON] has listed is their old address | it tastes the way food should taste! |
| wonderful location in a up and coming part of [GPE]. | every single person we spoke to on staff was absolutely incredible. |

**Table 1.** Examples from the YELP corpus for which MLE outperforms topic CVaR (left column) and vice versa. Brackets indicate ONTONOTES named-entity tags. The examples preferred by topic CVaR are stereotypical Yelp sentences, while those preferred by MLE refer to locations and accidents.

## 6 Related Work

**Domain Adaptation:** In the case of known source (train) and target (test) domains, there exist a variety of techniques to learn robust models (Shimodaira, 2000; Quiñonero-Candela et al., 2009; Daume III, 2007; Ben-David et al., 2010; Blitzer et al., 2011; Pryzant et al., 2017) or domain-invariant features (Ganin and Lempitsky, 2015; Tzeng et al., 2014). However, such methods require accurate domain membership annotations.

In the absence of domain membership annotations, prior multi-source domain adaptation (Mansour et al., 2009) approaches propose the use of clustering to identify candidate domains. For instance, Hoffman et al. (2012) and Xiong et al. (2014) discover latent domains in classification by clustering data using class labels. Gong et al. (2013) extend this work by identifying subsets which are distinct and learnable. More recent work consider errors in estimating the target domain (Hoffman et al., 2018) and derive learning bounds with respect to such errors. While these approaches make use of cluster and topic structures as prior, they still require some knowledge of the target distribution and train a model tailored to the target distribution. Instead, we assume no knowledge on the target distribution and train a single model by considering the worst case.

In *conditional* settings such as machine translation, prior works connect topic modeling and domain adaptation (Hu et al., 2014; Eidelman et al., 2012). However, unlike our work, these approaches use topics at *test time* by inferring the domain from the input variable $x$. In language modeling, we have no inputs and thus must find models robust to unknown domain shifts at test time. In addition, it can be difficult to infer the test distribution as the distribution can rapidly change across users and time.

**Distributional Robustness:** Our approach is based upon existing work in the distributionally robust optimization (DRO) literature. Optimizing on a ball of distributions around the empirical distribution has been considered in prior work (Ben-Tal et al., 2013; Namkoong and Duchi, 2017; Duchi and Namkoong, 2016; Sinha et al., 2018). Using DRO to minimize losses over subpopulations was proposed earlier in Hashimoto et al. (2018) and Duchi and Namkoong (2018), and Hu et al. (2018) proposed incorporating problem structure via class labels. Our work derives an efficient optimization procedure for DRO with topic-based uncertainty sets, and demonstrates that naively applying DRO to log losses fails to provide robustness due to the lack of baselining.

## 7 Discussion

In this work, we show that the performance of language models degrade as the amount of text from outside the test distribution grows. We hypothesize that this problem arises from the tendency of MLE to optimize for common sentences in the corpus, and we propose a solution based on distributionally robust optimization.

Empirically, we demonstrate that the DRO-based topic CVaR is more robust than MLE to subpopulation shifts and similar shifts. While this work focuses on DRO for language modeling, train-test mismatches under subpopulation shifts are more broadly applicable to any task where there are trade-offs between potential test distributions, and potential test distributions can be described with topics. Our work shows that topics are an effective way to encode prior information about test distributions, and baselines can properly normalize for the difficulty across these topics.

**Acknowledgments.** This work was supported by a PECASE Award and DARPA CwC program under ARO prime contract no. W911NF-15-1-0462. SS was supported by a Herbert Kunzel Stanford Graduate Fellowship.

**Reproducibility.** Code and data is available on CodaLab: https://worksheets.codalab.org/worksheets/0xf8122ebd24e94209a2a1764007509098.

## References


S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan. 2010. A theory of learning from different domains. *Machine Learning*, 79(1):151–175.

A. Ben-Tal, D. den Hertog, A. D. Waegenaere, B. Melenberg, and G. Rennen. 2013. Robust solutions of optimization problems affected by uncertain probabilities. *Management Science*, 59:341–357.

D. Blei, A. Ng, and M. I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research (JMLR)*, 3:993–1022.

J. Blitzer, S. Kakade, and D. P. Foster. 2011. Domain adaptation with coupled subspaces. In *Artificial Intelligence and Statistics (AISTATS)*, pages 173–181.

C. Chelba, T. Mikolov, M. Schuster, Q. Ge, T. Brants, P. Koehn, and T. Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

A. M. Dai and Q. V. Le. 2015. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

H. Daume III. 2007. Frustratingly easy domain adaptation. In *Association for Computational Linguistics (ACL)*.

J. Devlin, M. Chang, K. Lee, and K. Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

J. Duchi, P. Glynn, and H. Namkoong. 2016. Statistics of robust optimization: A generalized empirical likelihood approach. *arXiv*.

J. Duchi and H. Namkoong. 2016. Variance-based regularization with convex objectives. *arXiv preprint arXiv:1610.02581*.

J. Duchi and H. Namkoong. 2018. Learning models with uniform performance via distributionally robust optimization. *arXiv preprint arXiv:1810.08750*.

V. Eidelman, J. Boyd-Graber, and P. Resnik. 2012. Topic models for dynamic translation model adaptation. In *Association for Computational Linguistics (ACL)*, pages 115–119.

Y. Ganin and V. Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning (ICML)*, pages 1180–1189.

J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

B. Gong, K. Grauman, and F. Sha. 2013. Reshaping visual datasets for domain adaptation. In *Advances in Neural Information Processing Systems (NeurIPS)*.

T. B. Hashimoto, M. Srivastava, H. Namkoong, and P. Liang. 2018. Fairness without demographics in repeated loss minimization. In *International Conference on Machine Learning (ICML)*.

J. Hoffman, B. Kulis, T. Darrell, and K. Saenko. 2012. Discovering latent domains for multisource domain adaptation. In *European Conference on Computer Vision (ECCV)*, pages 702–715.

J. Hoffman, M. Mohri, and N. Zhang. 2018. Algorithms and theory for multiple-source adaptation. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 8256–8266.

M. Honnibal and M. Johnson. 2015. An improved non-monotonic transition system for dependency parsing. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1373–1378.

W. Hu, G. Niu, I. Sato, and M. Sugiyama. 2018. Does distributionally robust supervised learning give robust classifiers? In *International Conference on Machine Learning (ICML)*.

Y. Hu, K. Zhai, V. Eidelman, and J. Boyd-Graber. 2014. Polylingual tree-based topic models for translation domain adaptation. In *Association for Computational Linguistics (ACL)*, pages 1166–1176.

Y. Mansour, M. Mohri, and A. Rostamizadeh. 2009. Domain adaptation with multiple sources. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1041–1048.

D. Marcheggiani, O. Täckström, A. Esuli, and F. Sebastiani. 2014. Hierarchical multi-label conditional random fields for aspect-oriented opinion mining. In *ECIR*.

B. McCann, J. Bradbury, C. Xiong, and R. Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 6297–6308.

R. Nallapati, B. Zhou, C. Gulcehre, B. Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*.

H. Namkoong and J. Duchi. 2016. Stochastic gradient methods for distributionally robust optimization with f-divergences. In *Advances in Neural Information Processing Systems (NeurIPS)*.

H. Namkoong and J. Duchi. 2017. Variance regularization with convex objectives. In *Advances in Neural Information Processing Systems (NeurIPS)*.

M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. 2018. Deep contextualized word representations. In *North American Association for Computational Linguistics (NAACL)*.

R. Pryzant, D. Britz, and Q. V. Le. 2017. Effective domain mixing for neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 118–126.

J. Quiñonero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence. 2009. *Dataset shift in machine learning*. The MIT Press.

A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. 2018. Improving language understanding by generative pre-training. Technical report, OpenAI.

R. T. Rockafellar and S. Uryasev. 2000. Optimization of conditional value-at-risk. *Journal of Risk*, 2:21–41.

H. Shimodaira. 2000. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90:227–244.

A. Sinha, H. Namkoong, and J. Duchi. 2018. Certifiable distributional robustness with principled adversarial training. In *International Conference on Learning Representations (ICLR)*.

A. Sordoni, M. Galley, M. Auli, C. Brockett, Y. Ji, M. Mitchell, J. Nie, J. Gao, and B. Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *North American Association for Computational Linguistics (NAACL)*.

E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell. 2014. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

A. Vaswani, Y. Zhao, V. Fossum, and D. Chiang. 2013. Decoding with large-scale neural language models improves translation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1387–1392.

C. Xiong, S. McCloskey, S. Hsieh, and J. J. Corso. 2014. Latent domains modeling for visual domain adaptation. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

Yelp. 2017. Yelp Dataset Challenge, Round 8. https://www.yelp.com/dataset_challenge.

J. Yuan, F. Gao, Q. Ho, W. Dai, J. Wei, X. Zheng, E. P. Xing, T. Liu, and W. Ma. 2015. Lightlda: Big topic models on modest compute clusters. In *World Wide Web (WWW)*.

# Unsupervised Domain Adaptation of Contextualized Embeddings for Sequence Labeling

**Xiaochuang Han** and **Jacob Eisenstein**
Georgia Institute of Technology*
hxc@cmu.edu, me@jacob-eisenstein.com

## Abstract

Contextualized word embeddings such as ELMo and BERT provide a foundation for strong performance across a wide range of natural language processing tasks by pretraining on large corpora of unlabeled text. However, the applicability of this approach is unknown when the target domain varies substantially from the pretraining corpus. We are specifically interested in the scenario in which labeled data is available in only a canonical source domain such as newstext, and the target domain is distinct from both the labeled and pretraining texts. To address this scenario, we propose *domain-adaptive fine-tuning*, in which the contextualized embeddings are adapted by masked language modeling on text from the target domain. We test this approach on sequence labeling in two challenging domains: Early Modern English and Twitter. Both domains differ substantially from existing pretraining corpora, and domain-adaptive fine-tuning yields substantial improvements over strong BERT baselines, with particularly impressive results on out-of-vocabulary words. We conclude that domain-adaptive fine-tuning offers a simple and effective approach for the unsupervised adaptation of sequence labeling to difficult new domains.[1]

## 1 Introduction

Contextualized word embeddings are becoming a ubiquitous component of natural language processing (Dai and Le, 2015; Devlin et al., 2019; Howard and Ruder, 2018; Radford et al., 2018; Peters et al., 2018). Pretrained contextualized word embeddings can be used as feature for downstream tasks; alternatively, the contextualized word embedding module can be incorporated into an end-to-end system, allowing the embeddings to be fine-tuned from task-specific labeled data. In either case, a primary benefit of contextualized word embeddings is that they seed the learner with distributional information from large unlabeled datasets.

However, the texts used to build pretrained contextualized word embedding models are drawn from a narrow set of domains:

- **Wikipedia** in BERT (Devlin et al., 2019) and ULMFiT (Howard and Ruder, 2018);
- **Newstext** (Chelba et al., 2013) in ELMo (Peters et al., 2018);
- **BooksCorpus** (Zhu et al., 2015) in BERT (Devlin et al., 2019) and GPT (Radford et al., 2018).

All three corpora consist exclusively of text written since the late 20th century; furthermore, Wikipedia and newstext are subject to restrictive stylistic constraints (Bryant et al., 2005).[2] It is therefore crucial to determine whether these pretrained models are transferable to texts from other periods or other stylistic traditions, such as historical documents, technical research papers, and social media.

This paper offers a first empirical investigation of the applicability of domain adaptation to pretrained contextualized word embeddings. As a case study, we focus on sequence labeling in historical texts. Historical texts are of increasing interest in the computational social sciences and digital humanities, offering insights on patterns of

*XH is now at Carnegie Mellon University and JE is now at Google Research. Some of the work was performed while JE was visiting Facebook AI Research.

[1]Trained models for Early Modern English and Twitter are available at https://github.com/xhan77/AdaptaBERT

[2]While it might be desirable to completely retrain contextualized word embedding models in the target domain (e.g., Lee et al., 2019), this requires data and computational resources that are often unavailable.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4238–4248,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

language change (Hilpert and Gries, 2016), social norms (Garg et al., 2018), and the history of ideas and culture (Michel et al., 2011). Syntactic analysis can play an important role: researchers have used part-of-speech tagging to identify syntactic changes (Degaetano-Ortlieb, 2018) and dependency parsing to quantify gender-based patterns of adjectival modification and possession in classic literary texts (Vuillemot et al., 2009; Muralidharan and Hearst, 2013). But despite the appeal of using NLP in historical linguistics and literary analysis, there is relatively little research on how performance is impacted by diachronic transfer. Indeed, the evidence that does exist suggests that accuracy degrades significantly, especially if steps are not taken to adapt: for example, Yang and Eisenstein (2015) compare the accuracy of tagging 18th century and 16th century Portuguese text (using a model trained on 19th century text), and find that the error rate is twice as high for the older text.

We evaluate the impact of diachronic transfer on a contemporary pretrained part-of-speech tagging system. First, we show that a BERT-based part-of-speech tagger outperforms the state-of-the-art unsupervised domain adaptation method (Yang and Eisenstein, 2016), without taking any explicit steps to adapt to the target domain of Early Modern English. Next, we propose a simple unsupervised domain-adaptive fine-tuning step, using a masked language modeling objective over unlabeled text in the target domain. This yields significant further improvements, with more than 50% error reduction on out-of-vocabulary words. We also present a secondary evaluation on named entity recognition in contemporary social media (Strauss et al., 2016). This evaluation yields similar results: a direct application of BERT outperforms most of the systems from the 2016 WNUT shared task, and domain-adaptive fine-tuning yields further improvements. In both Early Modern English and social media, domain-adaptive fine-tuning does not decrease performance in the original source domain; the adapted tagger performs well in both settings. In contrast, fine-tuning BERT on *labeled* data in the target domain causes catastrophic forgetting (McCloskey and Cohen, 1989), with a significant deterioration in tagging performance in the source domain.

## 2 Tagging Historical Texts

Before describing our modeling approach, we highlight some of the unique aspects of tagging historical texts, focusing on the target dataset of the Penn-Helsinki Corpus of Early Modern English (Kroch et al., 2004).

### 2.1 Early Modern English

Early Modern English (EME) refers to the dominant language spoken in England during the period spanning the 15th-17th centuries, which includes the time of Shakespeare. While the English of this period is more comprehensible to contemporary readers than the Middle English that immediately preceded it, EME nonetheless differs from the English of today in a number of respects.

**Orthography.** A particularly salient aspect of EME is the variability of spelling and other orthographic conventions (Baron and Rayson, 2008). For example:

(1) If this marsch waulle (marsh wall) were not kept, and the canales of eche partes of Sowey river kept from abundance of wedes, al the plaine marsch ground at sodaine raynes (sudden rains) wold be overflowen, and the profite of the meade lost.

While these differences are not too difficult for fluent human readers of English, they affect a large number of tokens, resulting in a substantial increase in the out-of-vocabulary rate (Yang and Eisenstein, 2016). Some of the spelling differences are purely typographical, such as the substitution of *v* for *u* in words like *vnto*, and the substitution of *y* for *i* in words like *hym*. These are common sources of errors for baseline models. Another source of out-of-vocabulary words is the addition of a silent *e* to the end of many words. This generally did not cause errors for wordpiece-based models (like BERT), perhaps because the final 'e' is segmented as a separate token, which does not receive a tag. Capitalization is also used inconsistently, making it difficult to distinguish proper and common nouns, as in the following examples:

(2) And that those **Writs** which shall be awarded and directed for returning of **Juryes** ...

(3) ... shall not then have **Twenty** pounds or **Eight** pounds respectively ...

**Morphosyntax.** Aside from orthography, EME is fairly similar to contemporary English, with a few notable exceptions. EME includes several inflections that are rare or nonexistent today, such as the *-th* suffix for third person singular conjugation, as in *hath* (has) and *doth* (does). Another difference is in the system of second-person pronouns. EME includes the informal second person *thou* with declensions *thee, thine, thy*, and the plural second-person pronoun *ye*. These pronouns are significant sources of errors for baseline models: for example, a BERT-based tagger makes 216 errors on 272 occurrences of the pronoun *thee*.

### 2.2 Part-of-Speech Tags in the Penn Parsed Corpora of Historical English

The Penn Parsed Corpora of Historical English (PPCHE) include part-of-speech annotations for texts from several historical periods (Kroch et al., 2004). We focus on the corpus covering Early Modern English, which we refer to as PPCEME. As discussed in § 7, prior work has generally treated tagging the PPCEME as a problem of *domain adaptation*, with a post-processing stage to map deterministically between the tagsets. Specifically, we train on the Penn Treebank (PTB) corpus of 20th century English (Marcus et al., 1993), and then evaluate on the PPCEME test set, using a mapping between the PPCHE and PTB tagsets defined by Moon and Baldridge (2007).

Unfortunately, there are some fundamental differences in the approaches to verbs taken by each tagset. Unlike the PTB, the PPCHE has distinct tags for the modal verbs *have*, *do*, and *be* (and their infections); unlike the PPCHE, the PTB has distinct tags for third-person singular present indicative (VBZ) and other present indicative verbs (VBP). Moon and Baldridge map only to VBP, and Yang and Eisenstein report an error when VBZ is predicted, even though the corresponding PPCHE tag would be identical in both cases. We avoid this issue by focusing most of our evaluations on a coarse-grained version of the PTB tagset, described in § 4.3.

## 3 Adaptation with Contextualized Embeddings

The problem of processing historical English can be treated as one of unsupervised domain adaptation. Specifically, we assume that labeled data is available only in the "source domain" of contemporary modern English, and adapt to the "target domain" of historical text by using unlabeled data. We now explain how contextualized word embeddings can be applied in this setting.

Contextualized word embeddings provide a generic framework for semi-supervised learning across a range of natural language processing tasks, including sequence labeling tasks like part-of-speech tagging and named entity recognition (Peters et al., 2018; Devlin et al., 2019). Given a sequence of tokens $w_1, w_2, \ldots, w_T$, these methods return a sequence of vector embeddings $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$. The embeddings are *contextualized* in the sense that they reflect not only each token but also the context in which each token appears. The embedding function is trained either from a language modeling task (Peters et al., 2018) or a related task of recovering masked tokens (Devlin et al., 2019); these methods can be seen as performing semi-supervised learning, because they benefit from large amounts of unlabeled data.

### 3.1 Task-specific fine-tuning

Contextualized embeddings are powerful features for a wide range of downstream tasks. Of particular relevance for our work, Devlin et al. (2019) show that a state-of-the-art named entity recognition system can be constructed by simply feeding the contextualized embeddings into a linear classification layer. The log probability can then be computed by the log softmax,

$$\log p(y_t \mid \mathbf{w}_{1:T}) = \beta_{y_t} \cdot \mathbf{x}_t - \log \sum_{y \in \mathcal{Y}} \exp\left(\beta_y \cdot \mathbf{x}_t\right), \tag{1}$$

where the contextualized embedding $\mathbf{x}_t$ captures information from the entire sequence $\mathbf{w}_{1:T} = (w_1, w_2, \ldots, w_T)$, and $\beta_y$ is a vector of weights for each tag $y$.

To fine-tune the contextualized word embeddings, the model is trained by minimizing the negative conditional log-likelihood of the labeled data. This involves backpropagating from the tagging loss into the network that computes contextualized word embeddings. We refer to this procedure as *task-tuning*.

To borrow from the terminology of domain adaptation (Daumé III and Marcu, 2006), a *direct transfer* of contextualized word embeddings to the problem of tagging historical text works as follows:

1. Fine-tune BERT for the part-of-speech tagging task, using the Penn Treebank (PTB) corpus of 20th century English;

2. Apply BERT and the learned tagger to the test set of the Penn Parsed Corpus of Early Modern English (PPCEME).

We evaluate this approach in § 5.

### 3.2 Domain-adaptive fine-tuning

When the target domain differs substantially from the pretraining corpus, the contextualized word embeddings may be ineffective for the tagging task. This risk is particularly serious in unsupervised domain adaptation, because the labeled data may also differ substantially from the target text. In this case, task-specific fine-tuning may help adapt the contextualized embeddings to the labeling task, but not to the domain. To address this issue, we propose the *AdaptaBERT* model for unsupervised domain adaptation, which adds an additional fine-tuning objective: masked language modeling in the target domain. Specifically, we apply a simple two-step approach:

1. **Domain tuning**. In the first step, we fine-tune the contextualized word embeddings by backpropagating from the BERT objective, which is to maximize the log-probability of randomly masked tokens.

   We apply this training procedure to a dataset that includes all available target domain data, and an equal amount of unlabeled data in the source domain.[3] We create ten random maskings of each instance; in each masking, 15% of the tokens are randomly masked out, following the original BERT training procedure. We then perform three training iterations over this masked data.

2. **Task tuning**. In the second step, we fine-tune the contextualized word embeddings and learn the prediction model by backpropagating from the labeling objective on the source domain labeled data (Equation 1). This step fine-tunes the contextualized embeddings for the desired labeling task.

Attempts to interleave these two steps did not yield significant improvements in performance. While Peters et al. (2019) report good results on named entity recognition without task tuning, we found this step to be essential in our transfer applications.

[3] If the source domain dataset is smaller than the target domain data, then all of the unlabeled source domain data is included.

| | Domain tuning | Task tuning |
|---|---|---|
| Prediction | masked tokens | tags |
| Data | source + target | source |

Table 1: Overview of domain tuning and task tuning

## 4 Evaluation Setting

We evaluate on the task of part-of-speech tagging in the Penn Parsed Corpus of Early Modern English (PPCEME). There is no canonical training/test split for this data, so we follow Moon and Baldridge in randomly selecting 25% of the documents for the test set. Details of this split are described in supplement.

### 4.1 Systems

We evaluate the following systems:

**Frozen BERT.** This baseline applies the pretrained "BERT-base" contextualized embeddings, and then learns a tagger from the top-level embeddings, supervised by the PTB labeled data. The embeddings are from the pretrained case-sensitive BERT model, and are not adjusted during training. Peters et al. (2019) refer to this as a "feature extraction" application of pretrained embeddings. [4]

**Task-tuned BERT.** This baseline starts with pretrained BERT contextualized embeddings, and then fine-tunes them for the part-of-speech tagging task, using the PTB labeled data. This directly follows the methodology for named entity recognition proposed by Devlin et al. (2019) in the original BERT paper.

**AdaptaBERT.** Here we fine-tune the BERT contextualized embedding first on unlabeled data as described in § 3, and then on source domain labeled data. The target domain data is the unlabeled PPCEME training set.

**Fine-tuned BERT.** In *supervised* learning, we fine-tune the BERT contextualized embeddings on the labeled PPCEME training set.

[4] Note that this baseline learns only a linear final layer over the pretrained embeddings. We do not adopt weighted combination of internal contextual layers or any non-linear final layers in Peters et al. (2019).

Performance of this method should be viewed as an upper bound, because large-scale labeled data is not available in many domains of interest.

All BERT systems use the pretrained models from Google and the PyTorch implementation from huggingface.[5] Fine-tuning was performed using one NVIDIA GeForce RTX 2080 TI GPU. Domain-adaptive fine-tuning took 12 hours, and task tuning took an additional 30 minutes.

### 4.2 Previous results

We compare the above systems against prior published results from three feature-based taggers:

**SVM** A support vector machine baseline tagger, using the surface features described by Yang and Eisenstein (2015).

**FEMA** This is a feature-based unsupervised domain adaptation method for structured prediction (Yang and Eisenstein, 2015), which has the best reported performance on tagging the PPCEME. Unlike AdaptaBERT, the reported results for this system are based on feature induction from the entire PPCEME, including the (unlabeled) test set.

### 4.3 Tagset mappings

Because we focus on unsupervised domain adaptation, it is not possible to produce tags in the historical English (PPCHE) tagset, which is not encountered at training time. Following Moon and Baldridge (2007), we evaluate on a coarsened version of the PTB tagset, using the first letter of each tag (e.g., VBD → V). For comparison with Yang and Eisenstein (2016), we also report results on the full PTB tagset. In these evaluations, the ground truth is produced by applying the mapping of Moon and Baldridge to the PPCEME tags.

## 5 Results

Fine-tuning to the task and domain each yield significant improvements in performance over the Frozen BERT baseline (Table 2, line 1). Task-tuning improves accuracy by 7.6% on the coarse-grained tagset (line 2), and domain-adaptive fine-tuning yields a further 4.5% in accuracy (line 3). AdaptaBERT's performance gains are almost entirely due to the improvement on out-of-vocabulary terms, as discussed below.

The rightmost column of the table shows performance on the Penn Treebank test set. Interestingly, domain-adaptive fine-tuning has no impact on the performance on the original tagging task. This shows that adapting the pretrained BERT embeddings to the target domain does not make them less useful for tagging in the source domain, as long as task-tuning is performed after domain-adaptive tuning. In contrast, *supervised* fine-tuning in the target domain causes performance on the PTB to decrease significantly, as shown in line 4. This can be viewed as a form of *catastrophic forgetting* (McCloskey and Cohen, 1989).

As a secondary evaluation, we measure performance on the full PTB tagset in Table 3, thereby enabling direct comparison with prior work (Yang and Eisenstein, 2016). AdaptaBERT outperforms task-tuned BERT by 3.9%, again due to improvements on OOV terms. Task-tuned BERT is on par with the best previous unsupervised domain adaptation result (FEMA), showing the power of contextualized word embeddings, even across disparate domains. Note also that FEMA's representation was trained on the entire PPCEME, including the unlabeled test set, while the AdaptaBERT model uses the test set only for evaluation.

### 5.1 Out-of-vocabulary terms

We define out-of-vocabulary terms as those that are not present in the PTB training set. Of the out-of-PTB-vocabulary words in the PPCEME test set, 52.7% of the types and 82.2% of the tokens appear in the PPCEME training set. This enables domain-adaptive fine-tuning to produce better representations for these terms, making it possible to tag them correctly. Indeed, AdaptaBERT's gains come almost entirely from these terms, with an improvement in OOV accuracy of 25.8% over the frozen BERT baseline and 15.7% over task-tuned BERT. Similarly, on the full PTB tagset, AdaptaBERT attains an improvement in OOV accuracy of 11.3% over FEMA, which was the previous state-of-the-art.

### 5.2 Errors on in-vocabulary terms

The final two lines of Table 2 indicate that there remains a significant gap between AdaptaBERT and the performance of taggers trained with in-domain data: fine-tuning BERT on the PPCEME train-

[5] Models retrieved from https://github.com/google-research/bert on March 14, 2019; implementation retrieved from https://github.com/huggingface/pytorch-pretrained-bert also on March 14, 2019.

| | Early Modern English | | | PTB |
|---|---|---|---|---|
| System | Accuracy | In-vocab | Out-of-vocab | Accuracy |
| *Unsupervised domain adaptation* | | | | |
| 1. Frozen BERT | 77.7 | 83.7 | 61.0 | 91.4 |
| 2. Task-tuned BERT | 85.3 | 90.4 | 71.1 | 98.2 |
| 3. AdaptaBERT (this work) | **89.8** | **90.8** | **86.8** | **98.2** |
| *Supervised in-domain training* | | | | |
| 4. Fine-tuned BERT | 98.8 | 99.0† | 93.2† | 92.4 |

Table 2: Tagging accuracy on PPCEME, using the coarse-grained tagset. The unsupervised systems never see labeled data in the target domain of Early Modern English. † in line 4, "in-vocab" and "out-of-vocab" refer to the PPCEME training set vocabulary; for lines 1-3, this refers the PTB training set.

| System | Accuracy | In-vocab | Out-of-vocab |
|---|---|---|---|
| 1. SVM | 74.2 | 81.7 | 49.9 |
| 2. FEMA (Yang and Eisenstein, 2016) | 77.9 | 82.3 | 63.2 |
| 3. Task-tuned BERT | 78.4 | 84.4 | 58.4 |
| 4. AdaptaBERT (this work) | **82.3** | **84.7** | **74.5** |

Table 3: Tagging accuracy on PPCEME, using the full PTB tagset to compare with Yang and Eisenstein (2016).

ing set reduces the error rate to 1.2%. This improvement is largely attributable to *in-vocabulary* terms: although fine-tuned BERT does better than AdaptaBERT on both IV and OOV terms, the IV terms are far more frequent. The most frequent errors for AdaptaBERT are on tags for *to* (5337), *all* (2054), and *that* (1792). The OOV term with the largest number of errors is *al* (306), which is a shortening of *all*.

These errors on in-vocabulary terms can be explained by inconsistencies in annotation across the two domains:

- In the PPCEME, *to* may be tagged as either infinitival (to, e.g., *I am going to study*) or as a preposition (p, e.g., *I am going to Italy*). However, in the PTB, *to* is tagged exclusively as TO, which is a special tag reserved for this word.[6] Unsupervised domain adaptation generally fails predict the preposition tag for *to* when it appears in the PPCEME.
- In the PPCEME, *all* is often tagged as a quantifier (q), which is mapped to adjective (JJ) in the PTB. However, in the PTB, these cases are tagged as determiners (DT), and as a result, the domain adaptation systems always tag *all* as a determiner.
- In the PTB, the word *that* is sometimes tagged as a wh-determiner (WDT), in cases such as *symptoms that showed up decades later*. In the PPCEME, all such cases are tagged as complementizers (c), and this tag is then mapped to the preposition IN. AdaptaBERT often incorrectly tags *that* as WDT, when IN would be correct.

These examples point to the inherent limitations of unsupervised domain adaptation when there are inconsistencies in the annotation protocol.

## 6 Social Media Microblogs

As an additional evaluation, we consider social media microblogs, of which Twitter is the best known example in English. Twitter poses some of the same challenges as historical text: orthographic variation leads to a substantial mismatch in vocabulary between the target domain and source training documents such as Wikipedia (Baldwin et al., 2013; Eisenstein, 2013). We hypothesize that domain-adaptive fine-tuning can help to produce better contextualized word embeddings for microblog text.

Our evaluation is focused on the problem of identifying named entity spans in Tweets, which was the shared task of the 2016 Workshop on Noisy User Text (WNUT; Strauss et al., 2016). In

[6] In this discussion, sans-serif is used for PPCEME tags, and SMALL CAPS is used for the PTB tags.

the shared task, systems were given access to labeled data in the target domain; in contrast, we are interested to measure whether it is possible to perform this task without access to such data. As training data, we use the canonical CoNLL 2003 shared task dataset, in which named entity spans were annotated on a corpus of newstext (Tjong Kim Sang and De Meulder, 2003). Because the entity types are different in the WNUT and CoNLL corpora, we focus on the *segmentation task* of identifying named entity spans. Participants in the 2016 WNUT shared task competed on this metric, enabling a direct comparison with the performance of these supervised systems.

Results are shown in Table 4. A baseline system using task-tuned BERT (trained on the CoNLL labeled data) achieves an F1 of 57.7 (line 1). This outperforms six of the ten submissions to the WNUT shared task, even though these systems are trained on in-domain data. AdaptaBERT yields marginal improvements when domain-adaptive fine-tuning is performed on the WNUT training set (line 2); expanding the target domain data with an additional million unlabeled tweets yields a 2.3% improvement over the BERT baseline (line 3). Performance improves considerably when the domain-adaptive fine-tuning is performed on the combined WNUT training and test sets (line 4).

Test set adaptation is controversial: it is not realistic for deployed systems that are expected to perform well on unseen instances without retraining, but it may be applicable in scenarios in which we desire good performance on a predefined set of documents. The WNUT test set was constructed by searching for tweets in a narrow time window on two specific topics: shootings and cybersecurity events. It is therefore unsurprising that test set adaptation yields significant improvements, since it can yield useful representations of the names of the relevant entities, which might not appear in a random sample of tweets. This is a plausible approach for researchers who are interested in finding the key entities participating in such events in an pre-selected corpus of text.

When BERT is fine-tuned on labeled data in the target domain, the performance of the resulting tagger improves to 64.3% (line 5). This would have achieved second place in the 2016 WNUT shared task. The state-of-the-art system makes use of character-level information that is not available to our models (Limsopatham and Collier, 2016). As with the evaluation on Early Modern English, we find that domain-adaptive fine-tuning does not impair performance on the source domain (CoNLL), but supervised in-domain training increases the source domain error rate dramatically.

## 7 Related Work

**Adaptation in neural sequence labeling.** Most prior work on adapting neural networks for NLP has focused on supervised domain adaptation, in which a labeled data is available in the target domain (Mou et al., 2016). RNN-based models for sequence labeling can be adapted across domains by manipulating the input or output layers individually (e.g., Yang et al., 2016) or simultaneously (Lin and Lu, 2018). Unlike these approaches, we tackle unsupervised domain adaptation, which assumes only unlabeled instances in the target domain. In this setting, prior work has focused on *domain-adversarial* objectives, which construct an auxiliary loss based on the capability of an adversary to learn to distinguish the domains based on a shared encoding of the input (Ganin et al., 2016; Purushotham et al., 2017). However, adversarial methods require balancing between at least two and as many as six different objectives (Kim et al., 2017), which can lead to instability (Arjovsky et al., 2017) unless the objectives are carefully balanced (Alam et al., 2018).

In addition to supervised and unsupervised domain adaptation, there are "distantly supervised" methods that construct noisy target domain instances, e.g., by using a bilingual dictionary (Fang and Cohn, 2017). Normalization dictionaries exist for Early Modern English (Baron and Rayson, 2008) and social media (Han et al., 2012), but we leave their application to distant supervision for future work.

Finally, language modeling objectives have previously been used for domain adaptation of text classifiers (Ziser and Reichart, 2018), but this prior work has focused on representation learning from scratch, rather than adaptation of a pretrained contextualized embedding model. Our work shows that models that are pretrained on large-scale data yield very strong performance even when applied out-of-domain, making them a natural starting point for further adaptation.

**Semi-supervised learning.** Universal Language Model Fine-tuning (ULMFiT) also involves fine-tuning on a language modeling task on the target

| | WNUT | | | | CoNLL |
|---|---|---|---|---|---|
| System | domain adaptation data | Precision | Recall | **F1** | F1 |
| *Unsupervised domain adaptation* | | | | | |
| 1. Task-tuned BERT | n/a | 50.9 | 66.6 | 57.7 | **97.8** |
| 2. AdaptaBERT | WNUT training | 52.8 | 66.7 | 58.9 | 97.6 |
| 3. AdaptaBERT | + 1M tweets | 53.6 | 68.3 | 60.0 | 97.8 |
| 4. AdaptaBERT | WNUT train+test | 57.7 | 68.9 | **62.8** | 97.8 |
| *Supervised in-domain training* | | | | | |
| 5. Fine-tuned BERT | n/a | 66.3 | 62.3 | 64.3 | 80.9 |
| 6. Limsopatham and Collier (2016) | n/a | 73.5 | 59.7 | **65.9** | |

Table 4: Named entity segmentation performance on the WNUT test set and CoNLL test set A. Limsopatham and Collier (2016) had the winning system at the 2016 WNUT shared task. Their results are reprinted from their paper, which did not report performance on the CoNLL dataset.

text (Howard and Ruder, 2018), but the goal is semi-supervised learning: prior work shows that accurate text classification can be achieved with fewer labeled examples, but does not consider the issue of domain shift. ULMFiT involves a training regime in which the layers of the embedding model are gradually "unfrozen" during task-tuning, to avoid catastrophic forgetting. We do not employ this approach, nor did we experiment with ULMFiT's elaborate set of learning rate schedules. Finally, contemporaneous unpublished work has applied the BERT objective to target domain data, but like ULMFiT, this work focuses on semi-supervised classification rather than cross-domain sequence labeling (Xie et al., 2019).

**Tagging historical texts.** Moon and Baldridge (2007) approach part-of-speech tagging of Early Modern English by projecting annotations from labeled out-of-domain data to unlabeled in-domain data. The general problem of adapting part-of-speech tagging over time was considered by Yang and Eisenstein (2015). Their approach projected source (contemporary) and target (historical) training instances into a shared space, by examining the co-occurrence of hand-crafted features. This was shown to significantly reduce the transfer loss in Portuguese, and later in English (Yang and Eisenstein, 2016). However, this approach relies on hand-crafted features, and does not benefit from contemporary neural pretraining architectures. We show that pretrained contextualized embeddings yield significant improvements over feature-based methods.

**Normalization.** Another approach to historical texts and social media is spelling normalization (e.g., Baron and Rayson, 2008; Han et al., 2012), which has been shown to offer improvements in tagging historical texts (Robertson and Goldwater, 2018). In Early Modern English, Yang and Eisenstein (2016) found that domain adaptation and normalization are complementary. In this paper, we have shown that domain-adaptive fine-tuning (and wordpiece segmentation) significantly improves the OOV tagging accuracy from FEMA, so future research must explore whether normalization is still necessary for state-of-the-art tagging of historical texts.

**Domain-specific pretraining.** Given sufficient unlabeled data in the target domain, the simplest approach may be to retrain a BERT-like model from scratch. BioBERT is an application of BERT to the biomedical domain, which was achieved by pretraining on more than 10 billion tokens of biomedical abstracts and full-text articles from PubMed (Lee et al., 2019); SciBERT is a similar approach for scientific texts (Beltagy et al., 2019). Data on this scale is not available in Early Modern English, but retraining might be applicable to social media text, assuming the user has access to both large-scale data and sufficient computing power. The mismatch between pretrained contextualized embeddings and technical NER corpora was explored in recent work by Dai et al. (2019).

## 8 Conclusion

This paper demonstrates the applicability of contextualized word embeddings to two difficult un-

supervised domain adaptation tasks. On the task of adaptation to historical text, BERT works relatively well out-of-the-box, yielding equivalent performance to the best prior unsupervised domain adaptation approach. Domain-adaptive fine tuning on unlabeled target domain data yields significant further improvements, especially on OOV terms. On the task of adaptation to contemporary social media, a straightforward application of BERT yields competitive results, and domain-adaptive fine tuning again offers improvements.

A potentially interesting side note is that while supervised fine-tuning in the target domain results in *catastrophic forgetting* of the source domain, unsupervised target domain tuning does not. This suggests the intriguing possibility of training a single contextualized embedding model that works well across a wide range of domains, genres, and writing styles. However, further investigation is necessary to determine whether this finding is dependent on specific details of the source and target domains in our experiments, or whether it is a general difference between unsupervised domain tuning and supervised fine-tuning. We are also interested to more thoroughly explore how to combine domain-adaptive and task-specific fine-tuning within the framework of continual learning (Yogatama et al., 2019), with the goal of balancing between these apparently conflicting objectives.

**Acknowledgments.** Thanks to the anonymous reviewers and to Ross Girshick, Omer Levy, Michael Lewis, Yuval Pinter, Luke Zettlemoyer, and the Georgia Tech Computational Linguistics Lab for helpful discussions of this work. The research was supported by the National Science Foundation under award RI-1452443.

## References

Firoj Alam, Shafiq Joty, and Muhammad Imran. 2018. Domain adaptation with adversarial training and graph embeddings. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 1077–1087.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein generative adversarial networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 214–223.

Timothy Baldwin, Paul Cook, Marco Lui, Andrew MacKinlay, and Li Wang. 2013. How noisy social media text, how diffrnt social media sources. In *Proceedings of the 6th International Joint Conference on Natural Language Processing (IJCNLP 2013)*, pages 356–364.

Alistair Baron and Paul Rayson. 2008. Vard2: A tool for dealing with spelling variation in historical corpora. In *Postgraduate conference in corpus linguistics*.

Iz Beltagy, Arman Cohan, and Kyle Lo. 2019. SciBERT: Pretrained contextualized embeddings for scientific text. *arXiv preprint arXiv:1903.10676*.

Susan L Bryant, Andrea Forte, and Amy Bruckman. 2005. Becoming Wikipedian: transformation of participation in a collaborative online encyclopedia. In *Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work*, pages 1–10. ACM.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Neural Information Processing Systems (NIPS)*, pages 3079–3087.

Xiang Dai, Sarvnaz Karimi, Ben Hachey, and Cecile Paris. 2019. Using similarity measures to select pretraining data for NER. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Hal Daumé III and Daniel Marcu. 2006. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26:101–126.

Stefania Degaetano-Ortlieb. 2018. Stylistic variation over 200 years of court proceedings according to gender and social class. In *Proceedings of the Second Workshop on Stylistic Variation*, pages 1–10.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Jacob Eisenstein. 2013. What to do about bad language on the internet. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 359–369.

Meng Fang and Trevor Cohn. 2017. Model transfer for tagging low-resource languages using a bilingual dictionary. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(59):1–35.

Nikhil Garg, Londa Schiebinger, Dan Jurafsky, and James Zou. 2018. Word embeddings quantify 100 years of gender and ethnic stereotypes. *Proceedings of the National Academy of Sciences*, 115(16):E3635–E3644.

Bo Han, Paul Cook, and Timothy Baldwin. 2012. Automatically constructing a normalisation dictionary for microblogs. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*, pages 421–432.

Martin Hilpert and Stefan Th Gries. 2016. Quantitative approaches to diachronic corpus linguistics. *The Cambridge handbook of English historical linguistics*, pages 36–53.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 328–339.

Young-Bum Kim, Karl Stratos, and Dongchan Kim. 2017. Adversarial adaptation of synthetic or stale data. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 1297–1307.

Anthony Kroch, Beatrice Santorini, and Ariel Diertani. 2004. Penn-Helsinki Parsed Corpus of Early Modern English. http://www.ling.upenn.edu/hist-corpora/PPCEME-RELEASE-2/index.html.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. BioBERT: pre-trained biomedical language representation model for biomedical text mining. *arXiv preprint arXiv:1901.08746*.

Nut Limsopatham and Nigel Collier. 2016. Bidirectional LSTM for named entity recognition in twitter messages. In *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*, pages 145–152, Osaka, Japan. The COLING 2016 Organizing Committee.

Bill Yuchen Lin and Wei Lu. 2018. Neural adaptation layers for cross-domain named entity recognition. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier.

Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K Gray, Joseph P Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig, Jon Orwant, et al. 2011. Quantitative analysis of culture using millions of digitized books. *Science*, 331(6014):176–182.

Taesun Moon and Jason Baldridge. 2007. Part-of-speech tagging for middle English through alignment and projection of parallel diachronic texts. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*, pages 390–399.

Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2016. How transferable are neural networks in NLP applications? In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*, pages 479–489.

Aditi Muralidharan and Marti A Hearst. 2013. Supporting exploratory text analysis in literature study. *Literary and linguistic computing*, 28(2):283–295.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Matthew E. Peters, Sebastian Ruder, and Noah A. Smith. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 7–14, Florence, Italy. Association for Computational Linguistics.

Sanjay Purushotham, Wilka Carvalho, Tanachat Nilanon, and Yan Liu. 2017. Variational recurrent adversarial deep domain adaptation. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. Technical report, OpenAI.

Alexander Robertson and Sharon Goldwater. 2018. Evaluating historical text normalization systems: How well do they generalize? In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 720–725.

Benjamin Strauss, Bethany Toma, Alan Ritter, Marie-Catherine De Marneffe, and Wei Xu. 2016. Results of the wnut16 named entity recognition shared task. In *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*, pages 138–144.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Conference on Natural Language Learning (CoNLL)*, pages 142–147.

Romain Vuillemot, Tanya Clement, Catherine Plaisant, and Amit Kumar. 2009. What's being said near "Martha"? Exploring name entities in literary text collections. In *Symposium on Visual Analytics Science and Technology*, pages 107–114. IEEE.

Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V. Le. 2019. Unsupervised data augmentation. *arXiv preprint arXiv:1904.12848*.

Yi Yang and Jacob Eisenstein. 2015. Unsupervised multi-domain adaptation with feature embeddings. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Yi Yang and Jacob Eisenstein. 2016. Part-of-speech tagging for historical English. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Zhilin Yang, Ruslan Salakhutdinov, and William W Cohen. 2016. Transfer learning for sequence tagging with hierarchical recurrent networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Dani Yogatama, Cyprien de Masson d'Autume, Jerome Connor, Tomas Kocisky, Mike Chrzanowski, Lingpeng Kong, Angeliki Lazaridou, Wang Ling, Lei Yu, Chris Dyer, et al. 2019. Learning and evaluating general linguistic intelligence. *arXiv preprint arXiv:1901.11373*.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 19–27.

Yftah Ziser and Roi Reichart. 2018. Pivot based language modeling for improved neural domain adaptation. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 1241–1251.

# Learning Latent Parameters without Human Response Patterns: Item Response Theory with Artificial Crowds

**John P. Lalor**[1]*, **Hao Wu**[2], **Hong Yu**[1,3,4]
[1]College of Information and Computer Sciences, UMass Amherst
[2]Department of Psychology and Human Development, Vanderbilt University
[3]Department of Computer Science, UMass Lowell
[4]Center for Healthcare Organization & Implementation Research, Bedford VA
john.lalor@nd.edu, hao.wu.1@vanderbilt.edu,
hong.yu@umassmed.edu

## Abstract

Incorporating Item Response Theory (IRT) into NLP tasks can provide valuable information about model performance and behavior. Traditionally, IRT models are learned using human response pattern (RP) data, presenting a significant bottleneck for large data sets like those required for training deep neural networks (DNNs). In this work we propose learning IRT models using RPs generated from artificial crowds of DNN models. We demonstrate the effectiveness of learning IRT models using DNN-generated data through quantitative and qualitative analyses for two NLP tasks. Parameters learned from human and machine RPs for natural language inference and sentiment analysis exhibit medium to large positive correlations. We demonstrate a use-case for latent difficulty item parameters, namely training set filtering, and show that using difficulty to sample training data outperforms baseline methods. Finally, we highlight cases where human expectation about item difficulty does not match difficulty as estimated from the machine RPs.

## 1 Introduction

What is the most difficult example in the Stanford Natural Language Inference (SNLI) data set (Bowman et al., 2015) or in the Stanford Sentiment Treebank (SSTB) (Socher et al., 2013)? *A priori* the answer is not clear. How does one quantify the difficulty of an example and does it pertain to a specific model, or more generally?

There has been much recent work trying to assess the quality of data sets used for NLP tasks, (e.g. Lalor et al., 2016; Sakaguchi and Van Durme, 2018; Kaushik and Lipton, 2018). In particular, a common finding is that different examples within the same class have very different qualities such as difficulty, and these differences affect models' performance. For example, one study found that a subset of reading comprehension questions were so difficult as to be unanswerable (Kaushik and Lipton, 2018). In another work, the difficulty of specific items was found to be a significant predictor of whether a model would classify the item correctly (Lalor et al., 2018).

While a number of methods exist for estimating difficulty, in this work we focus on Item Response Theory (IRT) (Baker, 2001; Baker and Kim, 2004), a widely used method in psychometrics. IRT models fit parameters of data points (called "items") such as difficulty based on a large number of annotations ("response patterns" or RPs), typically gathered from a human population ("subjects"). It has been shown to be an effective way to evaluate and analyze NLP models with respect to human populations (Lalor et al., 2016, 2018).

While IRT models are designed to be learned with human RPs for at most 100 items, data sets used in machine learning, particularly for training deep neural networks (DNNs), are on the order of tens or hundreds of thousands of examples or more. It is not possible to ask humans to label every example in a data set of that size. In this work we hypothesize that IRT models can be fit using RPs from artificial crowds of DNNs as inputs, thereby removing the expense of gathering human RPs. Recent work has shown that DNNs encode linguistic knowledge (Tenney et al., 2019b,a) and can reach or surpass human-level performance on classification tasks (Lake et al., 2015). In addition, generating IRT data with deep learning models

*Current affiliation: Mendoza College of Business, University of Notre Dame



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4249–4259,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

is much cheaper compared to employing human annotators.

We demonstrate that learned parameters from IRT models fit with artificial crowd data are positively correlated with parameters learned with human data for small data sets. We then use variational inference (VI) methods (Jordan et al., 1999; Hoffman et al., 2013) to fit a large-scale IRT model. Using VI allows us to scale IRT models to deep-learning-sized data sets. Finally, we show why learning such models is useful by demonstrating how learned difficulties can improve training set subsampling.

Our contributions are as follows: (1) We show that IRT models can be fit using machine RPs by comparing item parameters learned from human and from machine RPs for two NLP tasks; (2) we show that RPs from more complex models lead to higher correlations between parameters from human and machine RPs; (3) we demonstrate a use-case for latent difficulty item parameters, namely training set filtering, and show that using difficulty to sample training data outperforms baseline methods; (4) we provide a qualitative analysis of items with the largest human-machine disagreement in terms of difficulty to highlight cases where human intuition is inconsistent with model behavior.

These results provide a direct comparison between humans and machine learning models in terms of identifying easy and difficult items. They also provide a foundation for large-scale IRT models to be fit by using ensembles of machine learning models to obtain RPs instead of humans, greatly reducing the cost of data-collection.[1]

## 2 Fitting Item Response Theory Models

### 2.1 Traditional Item Response Theory

Here we briefly describe IRT and the specific model under consideration, the Rasch model (also known as the one-parameter logistic or 1PL model) (Rasch, 1960).

We refer the reader to (Baker, 2001; Baker and Kim, 2004) for additional details on IRT, and to (Martınez-Plumed et al., 2016; Lalor et al., 2016, 2018) for more details on previous applications of IRT to machine learning.

IRT models are designed to estimate latent ability parameters ($\theta$) of subjects and latent item parameters such as difficulty of items ($b$). For a 1PL model, the probability that subject $j$ will answer item $i$ correctly is a function of the subject's latent ability $\theta_j$ and the item's latent difficulty $b_i$:

$$p(y_{ij} = 1|\theta_j, b_i) = \frac{1}{1 + e^{-(\theta_j - b_i)}} \quad (1)$$

The probability that subject $j$ will answer item $i$ incorrectly is:

$$p(y_{ij} = 0|\theta_j, b_i) = 1 - p(y_{ij} = 1|\theta_j, b_i) \quad (2)$$

The likelihood of a data set of RPs $Y$ from $J$ subjects to a set of $I$ items is:

$$p(Y|\theta, b) = \prod_{j=1}^{J} \prod_{i=1}^{I} p(Y_{ij} = y_{ij}|\theta_j, b_i) \quad (3)$$

For the 1PL model, the difficulty parameter represents the ability level at which the probability of an individual answering an item correctly is 50%. This occurs when item difficulty is equal to subject ability ($\theta_j = b_i$ in Eq. 1).

The item parameters are typically estimated by marginal maximum likelihood (MML) via an Expectation-Maximization (EM) algorithm (Bock and Aitkin, 1981), in which subject parameters are considered random effects $\theta_i \sim N(0, \sigma_\theta^2)$ and marginalized out. Once item parameters are learned, subjects' $\theta$ parameters are scored typically with maximum a posteriori (MAP) estimation. IRT models are usually fitted to RPs of hundreds or thousands of human subjects, who usually answer at most 100 questions. Therefore the methods for fitting these models have not been scaled to huge data sets and large numbers of subjects (e.g. tens of thousands of machine learning models).

### 2.2 IRT with Variational Inference

VI is a model fitting method that approximates an intractable posterior distribution in Bayesian inference by a simpler variational distribution. Prior work has compared VI methods with traditional IRT methods (Natesan et al., 2016) and found it effective, but was primarily concerned with fitting IRT models for human-scale data.

Bayesian methods in IRT assume that the individual $\theta$ and $b$ parameters in Eq. (2) both follow Gaussian prior distributions and make inference through the resultant joint posterior distribution $\pi(\theta, b|Y)$. As this posterior is usually intractable, VI approximates it by the variational distribution:

$$q(\theta, b) = \prod_{j=1}^{J} \pi_j^{\theta}(\theta_j) \prod_{i=1}^{I} \pi_i^{b}(b_i) \quad (4)$$

[1] Code for IRT model fitting is available at `https://github.com/jplalor/py-irt`.

Where $\pi_j^{\theta}()$ and $\pi_i^{b}()$ denotes different Gaussian densities for different parameters whose means and variances are determined by minimizing the KL-Divergence between $q(\theta, b)$ and $\pi(\theta, b|Y)$.

The choice of priors in Bayesian IRT can vary. Prior work has shown that vague and hierarchical priors are both effective (Natesan et al., 2016). We experiment with both in this work. A vague prior assumes $\theta_j \sim N(0, 1)$ and $b_i \sim N(0, 10^3)$, where the large variance indicates a lack of information on the difficulty parameters. A hierarchical Bayesian model assumes

$$\begin{aligned} \theta_j \mid m_\theta, u_\theta &\sim N(m_\theta, u_\theta^{-1}) \\ b_i \mid m_b, u_b &\sim N(m_b, u_b^{-1}) \\ m_\theta, m_b &\sim N(0, 10^6) \\ u_\theta, u_b &\sim \Gamma(1, 1) \end{aligned}$$

Our results for these two options were very similar, so we only report those for hierarchical priors.

## 3 Data and Models

Here we describe the data sets used to conduct our experiments, as well as the DNN model architectures for both generating response patterns and conducting our training set filtering experiment.

**SNLI** The SNLI data set (Bowman et al., 2015) is a popular data set for the natural language inference task. Briefly, each example in the data set consists of two sentences in English: the premise and the hypothesis, and a corresponding label. The correct label is "entailment" if the premise implies the hypothesis, "contradiction" if the premise implies that the hypothesis must be false, and "neutral" if the premise implies neither the hypothesis nor its negation. SNLI consists of 550k/10k/10k training/validation/testing examples examples.

**SSTB** The Stanford Sentiment Treebank (SSTB) (Socher et al., 2013) is a collection of English phrases extracted from movie reviews with fine-grained sentiment annotations (very negative, negative, neutral, positive, very positive). In this work we focus on binary sentiment classification, using the SST-2 split of the data set, where neutral examples have been removed. The data set consists of 67k/873/1.8k training/validation/testing examples.

**Human RP Data** The human RP data sets for SNLI and SSTB were previously collected from Amazon Mechanical Turk (AMT) workers (Lalor et al., 2016, 2018). For a randomly selected sample of items from SNLI and SSTB, new labels were gathered from 1000 AMT workers (Turkers). Each Turker labeled each item, so that for each item there were 1000 new labels. For each Turker, a RP was generated by grading the provided labels against the known gold-standard label.

**Building an Artificial Crowd** As mentioned earlier, it is not feasible to have humans provide RPs for data sets used to train DNN models. Can we instead use RPs from DNNs? We trained an ensemble of DNN models with varying amount of training data to simulate an artificial crowd so that enough responses were obtained to fit the IRT models. The goal here is not to build an ensemble of DNNs to surpass current classification state of the art results, but instead to test our hypothesis to determine if machine RPs can fit IRT models that can benefit NLP tasks.

Specifically, we trained 1000 LSTM models for NLI classification using the SNLI data set and 1000 LSTM models for binary SA classification using the SSTB data set (Bowman et al., 2015; Socher et al., 2013). The SNLI model consists of two LSTM sequence-embedding models (Hochreiter and Schmidhuber, 1997), one to encode the premise and another to encode the hypothesis. The two sentence encodings are then concatenated and passed through three tanh layers. Finally, the output is passed to a softmax classifier layer to output class probabilities. For SSTB, we used a single LSTM model without the concatenation step. The models were implemented in DyNet (Neubig et al., 2017). Models were trained with SGD for 100 epochs with a learning rate of 0.1, and validation set accuracy was used for early stopping.

For each model $m_i$, we randomly sampled a subset of the task training set, $x^i_{\text{train}}$. We corrupted a random selection of training labels by replacing the gold standard label with an incorrect label. For each model-training set pair, we trained the model, used the held out validation set for early stopping, and wrote the model's graded (correct/incorrect) outputs to disk as that model's RP. The set of RPs for all models is our input data for the IRT models.

We also looked at a more complex model to determine if the learned parameters would differ given the different model architectures. For our more complex model we used the Neural Semantic Encoder model (NSE), a memory-augmented recur-

rent neural network (Munkhdalai and Yu, 2017):

$$
\begin{aligned}
o_t &= f_{\mathrm{r}}^{LSTM}(x^t) \\
z_t &= softmax(o_t^\top M_{t-1}) \\
m_{r,t} &= z_t^\top M_{t-1} \\
c_t &= f_{\mathrm{c}}^{MLP}(o_t, m_{r,t}) \\
h_t &= f_{\mathrm{w}}^{LSTM}(c_t) \\
M_t = M_{t-1}(\mathbf{1} &- (z_t \otimes e_k)^\top) \\
&+ (h_t \otimes e_t)(z_t \otimes e_k)^\top
\end{aligned}
$$

where $f_r^{LSTM}$ is the read function, $f_c^{MLP}$ is the composition function, $f_w^{LSTM}$ is the write function, $M_t$ is the external memory at time $t$, and $e_l \in R^l$ and $e_k \in R^k$ are vectors of ones.

The goal with the data set restriction and label corruption was to build an ensemble of models with widely varying performance on the SNLI test set. Training with different training set sizes and levels of noise corruption means that certain models will perform very well on the test set (large training sets and low label corruption) while others will perform poorly (small training sets and high label corruption). This way we will get a variety of response patterns to simulate performance on the task across a spectrum of ability levels. While we could have modified the networks in any number of ways (e.g. changing layer sizes, learning rates, etc.), modifying the training data is a straightforward method for generating a variety of response patterns, and has been shown to have an impact on performance in terms of item difficulty (Lalor et al., 2018). Further investigations of network modifications is left for future work.

## 4 Methods

We conduct the following experiments: (i) a comparison of IRT parameters learned from human and machine RP data, using existing IRT data sets (Lalor et al., 2016, 2018) as the baseline for comparison, (ii) a comparison between MML and VI parameter estimates, and (iii) a demonstration of the effectiveness of learned IRT parameters via training data set selection experiments.

### 4.1 Validating Variational Inference

Before using VI to fit IRT models for DNN data, we must first show that VI produces estimates similar to traditional methods. This was established in prior work on synthetic data (Natesan et al., 2016). Here we compare them on an existing human data set (Lalor et al., 2016).

A traditional Rasch model was fit with both MML and VI. MML was implemented in the R package mirt (Chalmers et al., 2015) and VI in Pyro (Bingham et al., 2018), a probabilistic programming language built on PyTorch (Paszke et al., 2017) that implements typical VI model fitting and variance reduction (Kingma and Welling, 2014; Ranganath et al., 2014). We calculate the root mean squared difference (RMSD) between MML and VI estimates for subject and item parameters. Our expectation is that the RMSD will be sufficiently small to confirm that the VI parameters are similar enough to those learned by MML, since we will not be able to use MML when we attempt to scale up to larger data sets.

### 4.2 Human Machine Correlation

We further compare item difficulty parameters learned from machine RPs to those learned from human RPs. These two sets of parameters cannot be compared directly as they can only be interpreted in reference to their respective subject populations. Instead, we compute the correlation between these two sets of parameters to see whether items that are easy for humans are also easy for machines. We fit two Rasch models, one with existing human RPs (Lalor et al., 2016, 2018). and one with the machine RPs. Both models were fit with MML using the mirt R package (Chalmers et al., 2015). Learned item difficulty parameters were extracted and compared via Spearman $\rho$ rank order correlations.

### 4.3 Training Set Subsampling

To demonstrate the usefulness of the learned IRT parameters, we next describe a downstream use case: training set filtering for more efficient learning. Can we maintain model performance by removing the easiest and/or hardest items from the training set? Once difficulty parameters for each data set were learned, we trained a new DNN model using only a subset of the original training data. We trained a number of models, each with a different cutoff in terms of training data to observe how generalization was impacted in each case.

We looked at 4 filtering strategies (in each case $d$ is the item difficulty threshold): (i) absolute value inner (AVI), where all training items with $|b_i| < d$ were retained, (ii) absolute value outer (AVO), where all training items with $|b_i| > d$

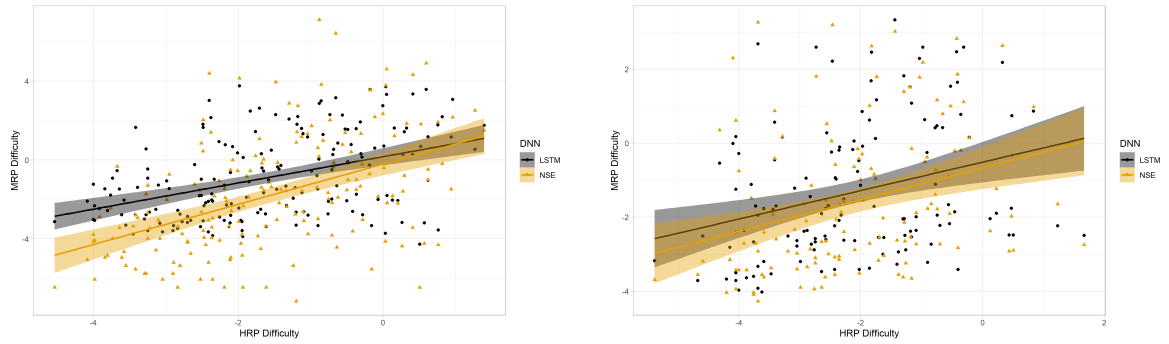


(a) NLI (b) SA

Figure 1: Comparison of learned item difficulty parameters for human (x-axis) and machine data (y-axis) for NLI (Fig. 1a) and SA (Fig. 1b). Spearman $\rho$ (NLI): 0.409 (LSTM) and 0.496 (NSE). Spearman $\rho$ (SA): 0.332 (LSTM) and 0.392 (NSE).

were retained, (iii) an upper bound (UB), where items with $b_i < d$ were retained, and (iv) a lower bound (LB), where items with $b_i > d$ were retained. These methods were compared against two baselines that consider the percentage of models that label an item correctly ($0 \leq pc \leq 1$) as an inexpensive proxy for difficulty: (i) percent-correct upper bound (PCUB), where items with $pc_i < d$ were retained, and (ii) percent-correct lower bound (PCLB), where items with $pc_i > d$ were retained. Setting an upper bound on difficulty (UB) is similar to setting a lower bound on percent correct (PCLB) (i.e., we are excluding the hardest items from training). Similarly, setting a lower bound on difficulty (LB) is analogous to setting an upper bound on percent correct (PCUB) in that they both exclude the easiest items from training.

Each of the filtering strategies have arguments in favor of their potential effectiveness. AVI includes "average" items in terms of training examples, none that are too easy or too difficulty. AVO is the opposite, where only the easiest and most difficult examples are retained, so that the extremes for each class can be learned. UB ensures that those examples that are too difficult are not included, and LB ensures that the examples that are too easy are not included so that the model doesn't spend time learning very easy examples.

## 5 Results

### 5.1 Human Machine Model Correlations

We first look at the results of our human-machine model comparison (Figures 1a and 1b). As an upper bound for correlations, we split the human annotation data in half for both SNLI and SSTB, fit two IRT Rasch models, and calculated the correlation between the learned parameters. Spearman $\rho$ values were 0.992 and 0.987 for SNLI and SSTB items, respectively.

For both SNLI and SSTB, we find a positive correlation between the item difficulties of IRT models fit using human and machine RPs. In addition, the more complex NSE model has consistently a higher correlation with the human-learned difficulty parameters than the LSTM model. This suggests that creating more complex DNN architectures has bearing on how the model identifies difficult items with regards to human expectations.

The correlation is not perfect, and we would argue that this is an expected and encouraging result. A close to perfect correlation would indicate that the DNN models and the human population agree closely on the difficulty ranking for the data sets and would be an incredible finding and evidence for the argument that DNN models encode human knowledge well, at least with respect to the difficulty of specific items. This of course is not true, and the positive but not perfect correlation coefficients indicate this as such. That said, it is encouraging that the positive correlation exists. One would expect that training ensembles of more sophisticated NLP models such as BERT (Devlin et al., 2018) would further increase correlation scores.

### 5.2 Learning IRT Models with VI

Our next goal was to determine if VI could be used to fit IRT models and confirm prior work to that effect (Natesan et al., 2016). The RMSDs between MML and VI estimates were 0.158 and 0.154, respectively, for the difficulty and ability

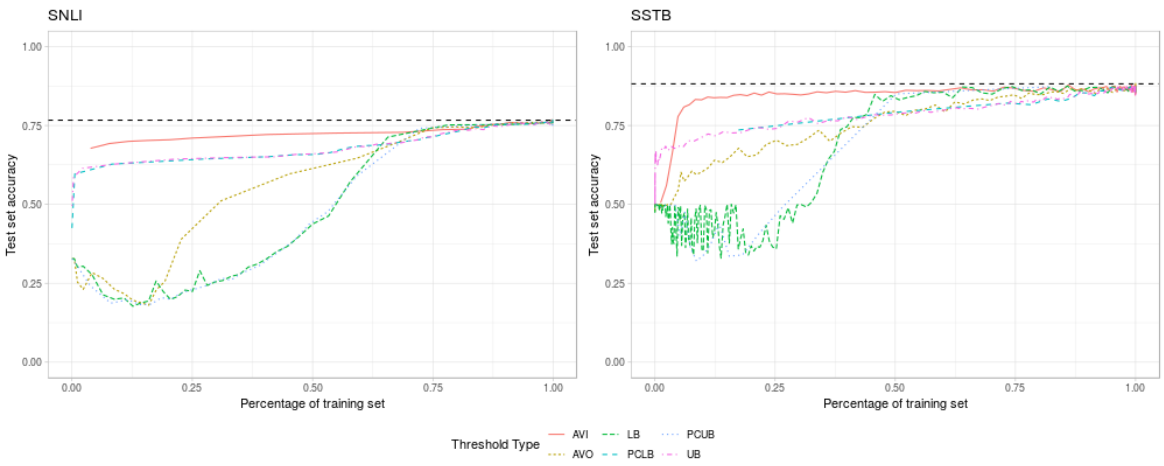


Figure 2: Test set accuracy by filtering strategy for NLI (left) and SA (right) plotted against percentage of training data retained. In both tasks filtering using the AVI strategy is most efficient in terms of high accuracy for small training set sizes.

parameters. Learned parameters are very similar between the two methods, which is to be expected. This echos the results of prior work showing that VI is a good alternative to traditional MML methods for learning IRT models (Natesan et al., 2016). This result holds not only with synthetic data, as was used in the prior work, but also with human data collected for the development of an actual IRT test (Lalor et al., 2016).

### 5.3 Data Filtering

Finally we consider training new DNN models on the filtered training data sets, restricted according to latent difficulty and the strategies described above (Figure 2). The horizontal dotted lines in each plot represent the test set accuracy for a model trained with the full training data set. For both SNLI and SSTB, the AVI strategy of selecting "average" examples leads to very good test set accuracy scores with less than 25% of the original training data. This shows that the strategy of selecting training data in terms of average difficulty, and gradually adding easier and harder examples at the same time provides examples that allows trained models to generalize well. For both tasks, there is a large number of examples that are very easy in terms of latent difficulty (Figure 3). Sampling with AVI avoids selecting too many examples that are too easy and instead selects examples that are of average difficulty for the task, which may be better for learning. In both cases LB and PCUB are the least effective strategies, indicating that it is not enough to only include the most difficult examples.

The plots show that PCUB and LB provide very

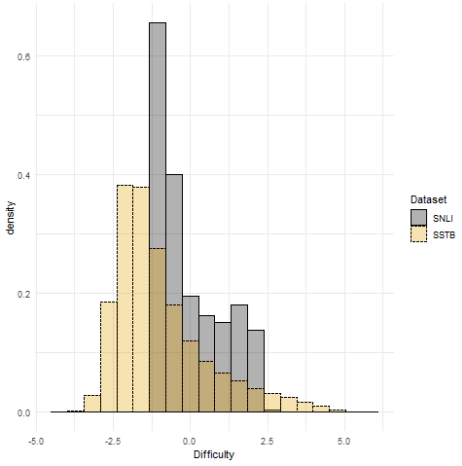


Figure 3: Density plot of learned difficulties for SNLI and SSTB data sets.

similar results, as do PCLB and UB, which is to be expected. Difficulty parameters learned from IRT are very similar to metrics such as percent correct, but as the plots show are not exactly the same. Differences in RPs (i.e. which specific items were answered correctly/incorrectly) have an effect on item difficulty that is not captured by calculating percent correct.

It is worth noting here that the filtering strategy we used did not take class labels into consideration.[2] The only determining factor as to whether a training item was included was the learned difficulty parameter $b_i$, which led to class imbalances in the training set. This imbalance, however did

[2]This is true for only the filtering step. Class labels are needed for learning the difficulty parameters needed for filtering (§2).

| Strategy | % of Training Data | | |
|---|---|---|---|
| | 0.1% | 1% | 10% |
| Random (reported) | 82.1 | 85.2 | **88.4** |
| Random (small batch) | 81.79 | 84.90 | 88.32 |
| Lower-bound | 43.68 | 41.56 | 39.89 |
| Upper-bound | 81.62 | 80.46 | 79.06 |
| AVI | **82.44** | **85.44** | 86.73 |
| AVO | 43.60 | 42.05 | 40.81 |

Table 1: Dev accuracy results for MT-DNN model with different training set sampling strategies.

not seem to have a significant negative effect in terms of performance. More advanced sampling strategies that maintain training set distribution or sample data using a Bayesian approach are left for future work.

As an additional experiment, we used the learned difficulty parameters to compare data sampling strategies for a state-of-the-art NLI model, MT-DNN (Liu et al., 2019). We sampled training data for SNLI at several intervals (0.1%, 1%, 10%) and trained the MT-DNN model with the sampled data. We trained each model, as well as the random sample baseline, using the publicly available MT-DNN code.[3] Results are reported in Table 1. Note that we report two random baselines: (i) those reported in the original work, which were obtained by training the MT-DNN model with a batch size of 32. Due to GPU resource constraints we had to train each MT-DNN model with a batch size of 8, and therefore report our reproduced random baseline results that we obtained as well ("Random (small batch)"). For very small samples of data, the AVI strategy outperforms random sampling and all other methods as well. As more data is sampled, the random models perform better. This indicates that a more advanced sampling strategy that starts with AVI then incorporates outliers (very easy/hard examples) at certain thresholds may improve learning as well.

## 6 Analysis

**Qualitative Evaluation of Difficulty** Table 2 shows examples of premise-hypothesis sentence pairs from SNLI with the learned difficulty parameter from the machine RP IRT model. The easy sentence pairs for each class seem to be very obvious, whereas the most difficult examples are difficult due to ambiguity. For example, the hardest contradiction example could be classified as neutral instead of contradiction. It could be the case that the man is sweeping while on vacation, though it isn't likely. The hypothesis doesn't directly contradict the premise like the easy example does (cats instead of dogs, sleeping instead of playing).

**Analysis of Differences** An interesting question comes up as a result of the less-than-perfect correlation scores (§5.1): Where are the differences? To examine these more closely we identified those examples from the data sets where the rank order was most different between the human- and machine-response pattern models (Table 3). That is, we calculated the absolute difference in ranking between the human model and the DNN model, and selected those where that value was highest. The average absolute difference in ranking was around 40 for the SNLI task and around 30 for SSTB, for both the LSTM and NSE ensembles.

We can see interesting patterns in the discrepancies. For SNLI, the easiest sentence pair for the LSTM model (which is also very easy for the NSE model) is one of the hardest for humans (Table 3, row 1). Upon inspection of the gathered labels, the high difficulty comes from the fact that there were many Turkers who labeled the data as neutral and also many who labeled it as contradiction.

On the other hand, an example that is easy for humans but difficult for the DNN models (Table 2, row 2) requires more abstract thinking than the earlier example. The humans are able to infer that because the girl is unwrapping an item, she will discover what is under the wrapping paper when the unwrapping is complete. The models find this pair to be one of the most difficult in the data set.

For SSTB, we see similar patterns (Table 3, rows 3-4). For humans, one of the easiest review snippets is clearly positive (row 3), mainly because we know who Anthony Hopkins is and know how to rate his quality as an actor. However for the DNN models, the text itself does not have a lot of positive or negative signal and therefore the item is considered very difficult. On the other hand, the last example is very difficult for humans (row 4), possibly due to the relatively neutral text. However, for the DNN models certain terms such as "stultifyingly contrived" may signal a more negative review and lead to the item being easier.

In both cases, it is not clear if there is a "gold standard" for difficulty. Estimating difficulty using IRT relies on responses from a group of humans or an ensemble of models, and the resulting difficulty estimates may be biased based on who or what

[3] https://github.com/namisan/mt-dnn

| Premise | Hypothesis | Label | Difficulty |
|---|---|---|---|
| Two men and a woman are inspecting the front tire of a bicycle. | There are a group of people near a bike. | Entailment | -3.7 |
| A girl in a newspaper hat with a bow is unwrapping an item. | The girl is going to find out what is under the wrapping paper. | Entailment | 3.1 |
| Two dogs playing in snow. | A cat sleeps on floor | Contradiction | -4.0 |
| Man sweeping trash outside a large statue. | A man is on vacation. | Contradiction | 3.8 |
| People sitting in chairs with a row flags hanging over them. | A family reunion for Fourth of July | Neutral | -3.6 |
| A group of dancers are performing. | The audience is silent. | Neutral | 3.8 |

Table 2: The easiest and hardest items judged by machine responses for each class in the SNLI test data set.

| Task | Label | Item Text | Difficulty ranking | | |
|---|---|---|---|---|---|
| | | | Humans | LSTM | NSE |
| SNLI | Contradiction | *P:* Two dogs playing in snow.<br>*H:* A cat sleeps on floor | 168 | 1 | 5 |
| | Entailment | *P:* A girl in a newspaper hat with a bow is unwrapping an item.<br>*H:* The girl is going to find out what is under the wrapping paper. | 55 | 172 | 176 |
| SSTB | Positive | Only two words will tell you what you know when deciding to see it: Anthony. Hopkins. | 9 | 103 | 110 |
| | Negative | ...are of course stultifyingly contrived and too stylized by half. Still, it gets the job done–a sleepy afternoon rental. | 128 | 46 | 41 |

Table 3: Examples from the SNLI and SSTB data sets where the ranking in terms of difficulty varies widely between human and DNN models. In all cases difficulty is ranked from easy to hard (1=easiest).

provides the labels. Human intuitions or model architecture decisions impact the response patterns collected, which in turn affect the learned parameters. An investigation into what upstream information drives downstream effects such as learned difficulty is an interesting and important direction for future work.

## 7 Related Work

Prior work has considered IRT in the context of evaluating ML models using human (Lalor et al., 2016) and machine-generated (Martınez-Plumed et al., 2016) response patterns. Martınez-Plumed et al. (2016) attempted to fit IRT models using machine generated response patterns on small data sets (i.e. 200-300 items), but obtain results that are difficult to interpret using the existing IRT assumptions. Lalor et al. (2016) develop new IRT test sets for NLI using human-generated data, and present new ways to interpret and understand model performance beyond raw accuracy. Due to the need for human annotations the resulting tests are short (i.e. 124 examples). To the best of our knowledge no one has attempted to fit IRT models using DNN-generated response patterns on large data sets.

There have been a number of studies on modeling latent traits of data to identify a correct label, (e.g. Bruce and Wiebe, 1999). There has also been work in modeling individuals to identify poor annotators (Hovy et al., 2013), but neither jointly model the ability of individuals and data points, nor apply the resulting metrics to interpret DNN models. Other work has modeled the probability a label is correct along with the probability of an annotator to label an item correctly according to the (Dawid and Skene, 1979) model, but do not consider difficulty or discriminatory ability of the data points (Passonneau and Carpenter, 2014). In the above models an annotator's response depends on an item only through its correct label. IRT assumes a more sophisticated response mechanism involving both annotator qualities and item characteristics. The DARE model (Bachrach et al., 2012) jointly estimates ability, difficulty and response using probabilistic inference. It was evaluated on an intelligence test of 60 multiple choice questions administered to 120 individuals.

There are several other areas of study regarding how best to use training data that are related to this work. Re-weighting or re-ordering training examples is a well-studied and related area of supervised learning. Often examples are re-weighted according to some notion of difficulty, or model uncertainty (Chang et al., 2017). In particular, the

internal uncertainty of the model is used as the basis for selecting how training examples are weighted. However, model uncertainty depends upon the original training data the model was trained on, while here we use an external measure of uncertainty.

Curriculum learning (CL) is a training procedure where models are trained to learn simple concepts before more complex concepts are introduced (Bengio et al., 2009). CL training for neural networks can improve generalization and speed up convergence. In curriculum learning the difficulty of items is typically assigned based on heuristics of the data (e.g. the number of sides of a shape). IRT models directly estimate difficulty from the responses of human or machine test-takers themselves instead of relying on heuristics. Self-paced learning and the Leitner method use model performance to estimate difficulties, but are restricted to a single model's performance, not a more global notion of difficulty (Kumar et al., 2010; Amiri et al., 2018).

## 8 Conclusion

In this work we have described how large-scale IRT models can be trained with DNN response patterns using VI. Learning the difficulty parameters of items and the ability parameters of DNN models allows for more nuanced interpretation of model performance and enables us to filter training data so that DNN models can be trained on less data while maintaining generalization as measured by test set performance. IRT models with machine RPs can be fit not only for NLP data sets but also data sets in other machine learning domains such as computer vision (additional results on two computer vision data sets are included in Appendix A).

One limitation of this work is the up-front cost of generating RPs from the DNN ensemble. However, the cost of running a large number of DNN models to generate response pattern data is significantly less than the cost of obtaining those labels from human annotators in two ways. First, the monetary cost of asking thousands of humans to label tens or hundreds of thousands of images or sentence pairs is prohibitive. Second, since the response patterns require that a single individual provide labels for all (or most) of the data set, each individual would need to label a huge number of items. Each individual would most likely get bored or burned out and the quality of the labels would suffer.

That said, consider for example a large company (or research lab) that runs hundreds or thousands of experiments each day on some internal data set. Many of the experiments would not lead to significant improvements in model performance, and the outputs from those experiments would be discarded. With the methods proposed here those outputs can be used to learn the latent parameters of the data to focus in on what exactly is working well and what isn't with respect to the models being tested and the data used to train them. Using the previously discarded data to learn IRT models and estimate latent difficulty and ability parameters can be used to improve a variety of tasks such as model selection, data selection, and curriculum learning strategies.

IRT models assume difficulty is a latent parameter of the items and can be estimated from response pattern data. Difficulty is directly linked to subject ability, in contrast to heuristics such as sentence length or word rarity. Certain items may be easy or difficult for a variety of reasons. With the methods presented here, an interesting direction for future work is to further examine why certain examples are more difficult than others.

We have shown that it is possible to fit IRT models using RPs from DNN models. Prior work relied on human RPs to investigate the impact of difficulty on model performance (Lalor et al., 2018), but it is now possible to conduct similar IRT analyses with machine RPs. This work also opens the possibility of fitting IRT models on much larger data sets. By removing the human bottleneck, we can use ensembles of DNN models to generate RPs for large data sets (e.g. all of SNLI or SSTB instead of a sample). Having difficulty and ability estimates for machine learning data sets and models can lead to very interesting work around such areas as active learning, curriculum learning, and meta learning.

## Acknowledgements

We thank the anonymous reviewers for their comments and suggestions. This work was supported in part by the HSR&D award IIR 1I01HX001457 from the United States Department of Veterans Affairs (VA). We also acknowledge the support of LM012817 from the National Institutes of Health. This work was also supported in part by the Center for Intelligent Information Retrieval. The contents of this paper do not represent the views of CIIR, NIH, VA, or the United States Government.

## References


Hadi Amiri, Timothy Miller, and Guergana Savova. 2018. Spotting spurious data with neural networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 2006–2016.

Yoram Bachrach, Thore Graepel, Tom Minka, and John Guiver. 2012. How to grade a test without knowing the answers - A bayesian graphical model for adaptive crowdsourcing and aptitude testing. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. icml.cc / Omnipress.

Frank B Baker. 2001. *The basics of item response theory*. ERIC.

Frank B. Baker and Seock-Ho Kim. 2004. *Item Response Theory: Parameter Estimation Techniques, Second Edition*. CRC Press.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th International Conference on Machine Learning*, pages 41–48. ACM.

E. Bingham, J. P. Chen, M. Jankowiak, F. Obermeyer, N. Pradhan, T. Karaletsos, R. Singh, P. Szerlip, P. Horsfall, and N. D. Goodman. 2018. Pyro: Deep Universal Probabilistic Programming. *ArXiv e-prints*.

R Darrell Bock and Murray Aitkin. 1981. Marginal maximum likelihood estimation of item parameters: Application of an em algorithm. *Psychometrika*, 46(4):443–459.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and D. Christopher Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642. Association for Computational Linguistics.

Rebecca F. Bruce and Janyce M. Wiebe. 1999. Recognizing subjectivity: A case study in manual tagging. *Nat. Lang. Eng.*, 5(2):187–205.

Phil Chalmers, Joshua Pritikin, Alexander Robitzsch, and Mateusz Zoltak. 2015. mirt: Multidimensional Item Response Theory.

Haw-Shiuan Chang, Erik Learned-Miller, and Andrew McCallum. 2017. Active bias: Training a more accurate neural network by emphasizing high variance samples. In *Advances in Neural Information Processing Systems*.

A. Philip Dawid and Allan M. Skene. 1979. Maximum likelihood estimation of observer error-rates using the em algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):20–28.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

S Hochreiter and J Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. 2013. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. Learning whom to trust with mace. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1120–1130. Association for Computational Linguistics.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. 1999. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233.

Divyansh Kaushik and Zachary C Lipton. 2018. How much reading does reading comprehension require? a critical investigation of popular benchmarks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing*, volume 2018.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *International Conference on Learning Representations*.

M Pawan Kumar, Benjamin Packer, and Daphne Koller. 2010. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, pages 1189–1197.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. 2015. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338.

John P Lalor, Hao Wu, Tsendsuren Munkhdalai, and Hong Yu. 2018. Understanding deep learning performance through an examination of test set difficulty: A psychometric case study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing*, volume 2018.

John P. Lalor, Hao Wu, and Hong Yu. 2016. Building an evaluation scale using item response theory. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 648–657. Association for Computational Linguistics.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*.

Fernando Martınez-Plumed, Ricardo BC Prudêncio, Adolfo Martınez-Usó, and José Hernández-Orallo. 2016. Making sense of item response theory in machine learning. In *Proceedings of 22nd European Conference on Artificial Intelligence (ECAI), Frontiers in Artificial Intelligence and Applications*, volume 285, pages 1140–1148.

Tsendsuren Munkhdalai and Hong Yu. 2017. Neural semantic encoders. *EACL 2017*.

Prathiba Natesan, Ratna Nandakumar, Tom Minka, and Jonathan D Rubright. 2016. Bayesian prior choice in irt estimation using mcmc and variational bayes. *Frontiers in psychology*, 7:1422.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Rebecca J. Passonneau and Bob Carpenter. 2014. The benefits of a model of annotation. *Transactions of the Association of Computational Linguistics*, 2:311–326.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

Rajesh Ranganath, Sean Gerrish, and David Blei. 2014. Black box variational inference. In *Artificial Intelligence and Statistics*, pages 814–822.

Georg Rasch. 1960. Studies in mathematical psychology: I. probabilistic models for some intelligence and attainment tests.

Keisuke Sakaguchi and Benjamin Van Durme. 2018. Efficient online scalar annotation with bounded support. In *Proceedings of the 56st Annual Meeting of the Association for Computational Linguistics*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, D. Christopher Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642. Association for Computational Linguistics.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019a. Bert rediscovers the classical nlp pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R Bowman, Dipanjan Das, et al. 2019b. What do you learn from context? probing for sentence structure in contextualized word representations. In *International Conference on Learning Representations*.

# Parallel Iterative Edit Models for Local Sequence Transduction

**Abhijeet Awasthi***, **Sunita Sarawagi** , **Rasna Goyal** , **Sabyasachi Ghosh** , **Vihari Piratla**
Department of Computer Science and Engineering, IIT Bombay

## Abstract

We present a Parallel Iterative Edit (PIE) model for the problem of local sequence transduction arising in tasks like Grammatical error correction (GEC). Recent approaches are based on the popular encoder-decoder (ED) model for sequence to sequence learning. The ED model auto-regressively captures full dependency among output tokens but is slow due to sequential decoding. The PIE model does parallel decoding, giving up the advantage of modelling full dependency in the output, yet it achieves accuracy competitive with the ED model for four reasons: 1. predicting edits instead of tokens, 2. labeling sequences instead of generating sequences, 3. iteratively refining predictions to capture dependencies, and 4. factorizing logits over edits and their token argument to harness pre-trained language models like BERT. Experiments on tasks spanning GEC, OCR correction and spell correction demonstrate that the PIE model is an accurate and significantly faster alternative for local sequence transduction. The code and pre-trained models for GEC are available at https://github.com/awasthiabhijeet/PIE.

## 1 Introduction

In local sequence transduction (LST) an input sequence $x_1, \ldots, x_n$ needs to be mapped to an output sequence $y_1, \ldots, y_m$ where the $\mathbf{x}$ and $\mathbf{y}$ sequences differ only in a few positions, $m$ is close to $n$, and $x_i, y_j$ come from the same vocabulary $\Sigma$. An important application of local sequence transduction that we focus on in this paper is Grammatical error correction (GEC). We contrast local transduction with more general sequence transduction tasks like translation and paraphrasing which might entail different input-output vocabulary and non-local alignments. The general sequence transduction task is cast as sequence to sequence (seq2seq) learning and modeled popularly using an attentional encoder-decoder (ED) model. The ED model auto-regressively produces each token $y_t$ in the output sequence conditioned on all previous tokens $y_1, \ldots, y_{t-1}$. Owing to the remarkable success of this model in challenging tasks like translation, almost all state-of-the-art neural models for GEC use it (Zhao et al., 2019; Lichtarge et al., 2019; Ge et al., 2018b; Chollampatt and Ng, 2018b; Junczys-Dowmunt et al., 2018).

We take a fresh look at local sequence transduction tasks and present a new parallel-iterative-edit (PIE) architecture. Unlike the prevalent ED model that is constrained to sequentially generating the tokens in the output, the PIE model generates the output in parallel, thereby substantially reducing the latency of sequential decoding on long inputs. However, matching the accuracy of existing ED models without the luxury of conditional generation is highly challenging. Recently, parallel models have also been explored in tasks like translation (Stern et al., 2018; Lee et al., 2018; Gu et al., 2018; Kaiser et al., 2018) and speech synthesis (van den Oord et al., 2018), but their accuracy is significantly lower than corresponding ED models. The PIE model incorporates the following four ideas to achieve comparable accuracy on tasks like GEC in spite of parallel decoding.

**1. Output edits instead of tokens:** First, instead of outputting tokens from a large vocabulary, we output edits such as copy, appends, deletes, replacements, and case-changes which generalize better across tokens and yield a much smaller vocabulary. Suppose in GEC we have an input sentence: `fowler fed dog`. Existing seq2seq learning approaches would need to output the four tokens `Fowler`, `fed`, `the`, `dog` from a word vocabulary whereas we

*Correspondence to: awasthi@cse.iitb.ac.in



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4260–4270, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

would predict the edits {`Capitalize token 1, Append(the) to token 2, Copy token 3`}.

**2. Sequence labeling instead of sequence generation:** Second we perform in-place edits on the source tokens and formulate local sequence transduction as *labeling* the input tokens with edit commands, rather than solving the much harder whole *sequence generation* task involving a separate decoder and attention module. Since input and output lengths are different in general such formulation is non-trivial, particularly due to edits that insert words. We create special compounds edits that merge token inserts with preceding edits that yield higher accuracy than earlier methods of independently predicting inserts (Ribeiro et al., 2018; Lee et al., 2018).

**3. Iterative refinement:** Third, we increase the inference capacity of the parallel model by iteratively inputting the model's own output for further refinement. This handles dependencies implicitly, in a way reminiscent of Iterative Conditional Modes (ICM) fitting in graphical model inference (Koller and Friedman, 2009). Lichtarge et al. (2019) and Ge et al. (2018b) also refine iteratively but with ED models.

**4. Factorize pre-trained bidirectional LMs:** Finally, we adapt recent pre-trained bidirectional models like BERT (Devlin et al., 2018) by factorizing the logit layer over edit commands and their token argument. Existing GEC systems typically rely on conventional forward directional LM to pretrain their decoder, whereas we show how to use a bi-directional LM in the encoder, and that too to predict edits.

Novel contributions of our work are as follows:

- Recognizing GEC as a local sequence transduction (LST) problem, rather than machine translation. We then cast LST as a fast non-autoregressive, sequence labeling model as against existing auto-regressive encoder-decoder model.
- Our method of reducing LST to non-autoregressive sequence labeling has many novel elements: outputting edit operations instead of tokens, append operations instead of insertions in the edit space, and replacements along with custom transformations.
- We show how to effectively harness a pre-trained language model like BERT using our factorized logit architecture with edit-specific attention masks.
- The parallel inference in PIE is 5 to 15 times faster than a competitive ED based GEC model like (Lichtarge et al., 2019) which performs sequential decoding using beam-search. PIE also attains close to state of the art performance on standard GEC datasets. On two other local transduction tasks, viz., OCR and spell corrections the PIE model is fast and accurate w.r.t. other existing models developed specifically for local sequence transduction.

## 2 Our Method

We assume a fully supervised training setup where we are given a parallel dataset of incorrect, correct sequence pairs: $D = \{(\mathbf{x}^i, \mathbf{y}^i) : i = 1 \dots N\}$, and an optional large corpus of unmatched correct sequences $\mathcal{L} = \{\tilde{\mathbf{y}}^1, \dots, \tilde{\mathbf{y}}^U\}$. In GEC, this could be a corpus of grammatically correct sentences used to pre-train a language model.

**Background: existing ED model** Existing seq2seq ED models factorize $\Pr(\mathbf{y}|\mathbf{x})$ to capture the full dependency between a $y_t$ and all previous $\mathbf{y}_{<t} = y_1, \dots, y_{t-1}$ as $\prod_{t=1}^{m} \Pr(y_t|\mathbf{y}_{<t}, \mathbf{x})$. An encoder converts input tokens $x_1, \dots, x_n$ to contextual states $\mathbf{h}_1, \dots, \mathbf{h}_n$ and a decoder summarizes $\mathbf{y}_{<t}$ to a state $\mathbf{s}_t$. An attention distribution over contextual states computed from $\mathbf{s}_t$ determines the relevant input context $\mathbf{c}_t$ and the output token distribution is calculated as $\Pr(y_t|\mathbf{y}_{<t}, \mathbf{x}) = \Pr(y_t|\mathbf{c}_t, \mathbf{s}_t)$. Decoding is done sequentially using beam-search. When a correct sequence corpus $\mathcal{L}$ is available, the decoder is pre-trained on a next-token prediction loss and/or a trained LM is used to re-rank the beam-search outputs (Zhao et al., 2019; Chollampatt and Ng, 2018a; Junczys-Dowmunt et al., 2018).

### 2.1 Overview of the PIE model

We move from generating tokens in the output sequence $\mathbf{y}$ using a separate decoder, to labelling the input sequence $x_1, \dots, x_n$ with edits $e_1, \dots, e_n$. For this we need to design a function Seq2Edits that takes as input an $(\mathbf{x}, \mathbf{y})$ pair in $D$ and outputs a sequence $\mathbf{e}$ of edits from an edit space $\mathcal{E}$ where $\mathbf{e}$ is of the same length as $\mathbf{x}$ in spite of $\mathbf{x}$ and $\mathbf{y}$ being of different lengths. In Section 2.2 we show how we design such a function.

**Training:** We invoke Seq2Edits on $D$ and learn the parameters of a probabilistic model $\Pr(\mathbf{e}|\mathbf{x}, \theta)$ to assign a distribution over the edit labels on tokens of the input sequence. In Section 2.3 we describe the PIE architecture in more detail. The correct corpus $\mathcal{L}$ , when available, is used to pre-train the encoder to predict an arbitrarily masked token $y_t$ in a sequence $\mathbf{y}$ in $\mathcal{L}$, much like in BERT. Unlike in existing seq2seq systems where $\mathcal{L}$ is used to pre-train the decoder that only captures forward dependencies, in our pre-training the predicted token $y_t$ is dependent on both forward and backward contexts. This is particularly useful for GEC-type tasks where future context $y_{t+1} \dots y_m$ can be approximated by $x_{t+1}, \dots, x_n$.

**Inference:** Given an input $\mathbf{x}$, the trained model predicts the edit distribution for each input token independent of others, that is $\Pr(\mathbf{e}|\mathbf{x}, \theta) = \prod_{t=1}^{n} \Pr(e_t|\mathbf{x}, t, \theta)$, and thus does not entail the latency of sequential token generation of the ED model. We output the most probable edits $\hat{\mathbf{e}} = \text{argmax}_{\mathbf{e}} \Pr(\mathbf{e}|\mathbf{x}, \theta)$. Edits are designed so that we can easily get the edited sequence $\hat{\mathbf{y}}$ after *applying* $\hat{\mathbf{e}}$ on $\mathbf{x}$. $\hat{\mathbf{y}}$ is further refined by iteratively applying the model on the generated outputs $\hat{\mathbf{y}}$ until we get a sequence identical to one of the previous sequences upto a maximum number of iterations $I$.

## 2.2 The Seq2Edits Function

Given a $\mathbf{x} = x_1, \dots, x_n$ and $\mathbf{y} = y_1, \dots, y_m$ where $m$ may not be equal to $n$, our goal is to obtain a sequence of edit operations $\mathbf{e} = (e_1, \dots, e_n) : e_i \in \mathcal{E}$ such that applying edit $e_i$ on the input token $x_i$ at each position $i$ reconstructs the output sequence $\mathbf{y}$. Invoking an off-the-shelf edit distance algorithm between $\mathbf{x}$ and $\mathbf{y}$ can give us a sequence of copy, delete, replace, and insert operations of arbitrary length. The main difficulty is converting the insert operations into in-place edits at each $x_i$. Other parallel models (Ribeiro et al., 2018; Lee et al., 2018) have used methods like predicting insertion slots in a pre-processing step, or predicting zero or more tokens in-between any two tokens in $\mathbf{x}$. We will see in Section 3.1.4 that these options do not perform well. Hence we design an alternative edit space $\mathcal{E}$ that merges inserts with preceding edit operations creating compound append or replace operations. Further, we create a dictionary $\Sigma_a$ of common q-gram insertions or replacements observed in the training data.

Our edit space ($\mathcal{E}$) comprises of copy (C) $x_i$, delete (D) $x_i$, append (A) a q-gram $w \in \Sigma_a$ after copying $x_i$, replace (R) $x_i$ with a q-gram $w \in \Sigma_a$. For GEC, we additionally use transformations denoted as $\text{T}_1, \dots, \text{T}_k$ which perform word-inflection (e.g. `arrive` to `arrival`). The space of all edits is thus:

$$\mathcal{E} = \{\text{C}, \text{D}, \text{T}_1, \dots, \text{T}_k\} \cup \{\text{A}(w) : w \in \Sigma_a\} \cup \{\text{R}(w) : w \in \Sigma_a\} \quad (1)$$

**Require:** $\mathbf{x} = (x_1, \dots, x_n), \mathbf{y} = (y_1, \dots, y_m)$ and $\mathcal{T}$ : List of Transformations
diffs $\leftarrow$ LEVENSHTEIN-DIST($\mathbf{x}, \mathbf{y}$) with modified cost.
diffs $\leftarrow$ In diffs break substitutions, merge inserts into q-grams
$\Sigma_a \leftarrow M$ most common inserts in training data
$\mathbf{e} \leftarrow$ EMPTYARRAY($n$)
**for** $t \leftarrow 1$ to LENGTH(diffs) **do**
  **if** diffs[$t$] = (C, $x_i$) or (D, $x_i$) **then**
    $e_i \leftarrow$ diffs[$t$].$op$
  **else if** diffs[$t$] = (I, $x_i$, $w$) **then**
    **if** $e_i$ = D **then**
      **if** ($x_i$, $w$) match transformation T $\in \mathcal{T}$ **then**
        $e_i \leftarrow$ T
      **else**
        $e_i \leftarrow$ R($w$) **if** $w \in \Sigma_a$ else C
    **else if** $e_i$ = C **then**
      $e_i \leftarrow$ A($w$) **if** $w \in \Sigma_a$ else C
**return** $\mathbf{e}$

Figure 1: The Seq2Edits function used to convert a sequence pair $\mathbf{x}, \mathbf{y}$ into in-place edits.

| | Example 1 |
|---|---|
| $\mathbf{x}$ | [ Bolt can have run race ] |
| $\mathbf{y}$ | [ Bolt could have run the race ] |
| `diff` | (C,[) (C,Bolt) (D,can) (I,can,could) (C,have) (C,run) (I,run,the) (C,race) (C,]) |
| $\mathbf{e}$ | C C R(could) C A(the) C C<br>↑ ↑ ↑ ↑ ↑ ↑ ↑<br>[ Bolt can have run race ] |
| | **Example 2** |
| $\mathbf{x}$ | [ He still won race ! ] |
| $\mathbf{y}$ | [ However , he still won ! ] |
| `diff` | (C,[) (I,[,However,) (D,He) (I,He,he) (C,still) (C,won) (D,-race) (C,!) (C,]) |
| $\mathbf{e}$ | A(However,) T_case C C D C C<br>↑ ↑ ↑ ↑ ↑ ↑ ↑<br>[ He still won race ! ] |

Table 1: Two example sentence pairs converted into respective edit sequences.

We present our algorithm for converting a sequence $\mathbf{x}$ and $\mathbf{y}$ into in-place edits on $\mathbf{x}$ using the above edit space in Figure 1. Table 1 gives examples of converting ($\mathbf{x}$, $\mathbf{y}$) pairs to edit sequences. We first invoke the Levenshtein distance algorithm (Levenshtein, 1966) to obtain `diff` between $\mathbf{x}$ and $\mathbf{y}$ with delete and insert cost as 1 as usual, but

with a modified substitution cost to favor matching of related words. We detail this modified cost in the Appendix and show an example of how this modification leads to more sensible edits. The `diff` is post-processed to convert substitutions into deletes followed by inserts, and consecutive inserts are merged into a q-gram. We then create a dictionary $\Sigma_a$ of the $M$ most frequent q-gram inserts in the training set. Thereafter, we scan the `diff` left to right: a copy at $x_i$ makes $e_i$ = C, a delete at $x_i$ makes $e_i$ = D, an insert $w$ at $x_i$ and a $e_i$ = C flips the $e_i$ into a $e_i$ = A$(w)$ if $w$ is in $\Sigma_a$, else it is dropped, an insert $w$ at $x_i$ and a $e_i$ = D flips the $e_i$ into a $e_i$ = T$(w)$ if a match found, else a replace R$(w)$ if $w$ is in $\Sigma_a$, else it is dropped.

The above algorithm does not guarantee that when $\mathbf{e}$ is applied on $\mathbf{x}$ we will recover $\mathbf{y}$ for all sequences in the training data. This is because we limit $\Sigma_a$ to include only the $M$ most frequently inserted q-grams. For local sequence transduction tasks, we expect a long chain of consecutive inserts to be rare, hence our experiments were performed with $q = 2$. For example, in NUCLE dataset (Ng et al., 2014) which has roughly 57.1K sentences, less than 3% sentences have three or more consecutive inserts.

### 2.3 The Parallel Edit Prediction Model

We next describe our model for predicting edits $\mathbf{e}$ : $e_1, \ldots, e_n$ on an input sequence $\mathbf{x}$ : $x_1, \ldots, x_n$. We use a bidirectional encoder to provide a contextual encoding of each $x_i$. This can either be multiple layers of bidirectional RNNs, CNNs or deep bidirectional transformers. We adopt the deep bidirectional transformer architecture since it encodes the input in parallel. We pre-train the model using $\mathcal{L}$ much like in BERT pre-training recently proposed for language modeling(Devlin et al., 2018). We first give an overview of BERT and then describe our model.

**Background: BERT** The input is token embedding $\mathbf{x}_i$ and positional embedding $\mathbf{p}_i$ for each token $x_i$ in the sequence $\mathbf{x}$. Call these jointly as $\mathbf{h}_i^0 = [\mathbf{x}_i, \mathbf{p}_i]$. Each layer $\ell$ produces a $\mathbf{h}_i^\ell$ at each position $i$ as a function of $\mathbf{h}_i^{\ell-1}$ and self-attention over all $\mathbf{h}_j^{\ell-1}$, $j \in [1, n]$. The BERT model is pre-trained by masking out a small fraction of the input tokens by a special MASK, and predicting the masked word from its bidirectional context captured as the last layer output: $\mathbf{h}_1, \ldots, \mathbf{h}_n$.

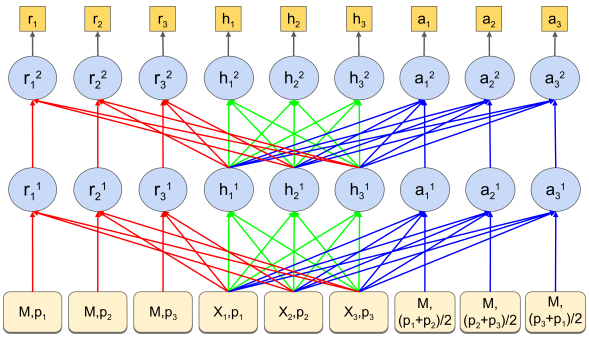


Figure 2: A Parallel Edit Architecture based on a 2-layer bidirectional transformer. Input sequence length is 3. Arrows indicate attention mask for computation of $\mathbf{h}_i^l, \mathbf{r}_i^l, \mathbf{a}_i^l$ at position i for layer l.

#### 2.3.1 A Default use of BERT

Since we have cast our task as a sequence labeling task, a default output layer would be to compute $\Pr(e_i|\mathbf{x})$ as a softmax over the space of edits $\mathcal{E}$ from each $\mathbf{h}_i$. If $W_e$ denotes the softmax parameter for edit $e$, we get:

$$\Pr(e_i = e|\mathbf{x}) = \text{softmax}(W_e^\intercal \mathbf{h}_i) \qquad (2)$$

The softmax parameters, $W_e$ in Equation 2, have to be trained from scratch. We propose a method to exploit token embeddings of the pre-trained language model to warm-start the training of edits like appends and replaces which are associated with a token argument. Furthermore, for appends and replaces, we provide a new method of computing the hidden layer output via alternative input positional embeddings and self-attention. We do so without introducing any new parameters in the hidden layers of BERT.

#### 2.3.2 An Edit-factorized BERT Architecture

We adapt a pre-trained BERT-like bidirectional language model to learn to predict edits as follows. For suitably capturing the contexts for replace edits, for each position $i$ we create an additional input comprising of $\mathbf{r}_i^0 = [\text{M}, \mathbf{p}_i]$ where M is the embedding for MASK token in the LM. Likewise for a potential insert between $i$ and $i+1$ we create an input $\mathbf{a}_i^0 = [\text{M}, \frac{\mathbf{p}_i + \mathbf{p}_{i+1}}{2}]$ where the second component is the average of the position embedding of the $i$th and $i+1$th position. As shown in Figure 2, at each layer $\ell$ we compute self-attention for the $i$th replace unit $\mathbf{r}_i^\ell$ over $\mathbf{h}_j^\ell$ for all $j \neq i$ and itself. Likewise, for the append unit $\mathbf{a}_i^\ell$ the self-attention is over all $\mathbf{h}_j^\ell$ for all $j$s and itself. At the last layer we have $\mathbf{h}_1, \ldots, \mathbf{h}_n, \mathbf{r}_1, \ldots, \mathbf{r}_n, \mathbf{a}_1, \ldots, \mathbf{a}_n$. Using these we compute logits factorized over edits

and their token argument. For an edit $e \in \mathcal{E}$ at position $i$, let $w$ denote the argument of the edit (if any). As mentioned earlier, $w$ can be a q-gram for append and replace edits. Embedding of $w$, represented by $\phi(w)$ is obtained by summing up individual output embeddings of tokens in $w$. Additionally, in the outer layer we allocate edit-specific parameters $\boldsymbol{\theta}$ corresponding to each distinct command in $\mathcal{E}$. Using these, various edit logits are computed as follows:

$$\Pr(e_i|\mathbf{x}) = \text{softmax}(\text{logit}(e_i|\mathbf{x})) \quad \text{where}$$
$$\text{logit}(e_i|\mathbf{x}) = \tag{3}$$
$$\begin{cases} \theta_{\text{C}}^{\intercal}\mathbf{h}_i + \phi(x_i)^{\intercal}\mathbf{h}_i + 0 & \text{if } e_i = \text{C} \\ \theta_{\text{A}(w)}^{\intercal}\mathbf{h}_i + \phi(x_i)^{\intercal}\mathbf{h}_i + \phi(w)^{\intercal}\mathbf{a}_i & \text{if } e_i = \text{A}(w) \\ \theta_{\text{R}(w)}^{\intercal}\mathbf{h}_i + 0 + (\phi(w) - \phi(x_i))^{\intercal}\mathbf{r}_i & \text{if } e_i = \text{R}(w) \\ \theta_{\text{D}}^{\intercal}\mathbf{h}_i + 0 + 0 & \text{if } e_i = \text{D} \\ \theta_{T_k}^{\intercal}\mathbf{h}_i + \phi(x_i)^{\intercal}\mathbf{h}_i + 0 & \text{if } e_i = T_k \end{cases}$$

The first term in the RHS of above equations captures edit specific score. The second term captures the score for copying the current word $x_i$ to the output. The third term models the influence of a new incoming token in the output obtained by a replace or append edit. For replace edits, score of the replaced word is subtracted from score of the incoming word. For transformation we add the copy score because they typically modify only the word forms, hence we do not expect meaning of the transformed word to change significantly.

The above equation provides insights on why predicting independent edits is easier than predicting independent tokens. Consider the append edit ($\text{A}(w)$). Instead of independently predicting $x_i$ at $i$ and $w$ at $i+1$, we jointly predict the tokens in these two slots and contrast it with not inserting any new $w$ after $x_i$ in a single softmax. We will show empirically (Sec 3.1.4) that such selective joint prediction is key to obtaining high accuracy in spite of parallel decoding.

Finally, loss for a training example $(\mathbf{e}, \mathbf{x})$ is obtained by summing up the cross-entropy associated with predicting edit $e_i$ at each token $x_i$.

$$\mathcal{L}(\mathbf{e}, \mathbf{x}) = -\Sigma_i \log(\Pr(e_i|\mathbf{x})) \tag{4}$$

# 3 Experiments

We compare our parallel iterative edit (PIE) model with state-of-the-art GEC models that are all based on attentional encoder-decoder architectures. In Section 3.2 we show that the PIE model is also effective on two other local sequence transduction tasks: spell and OCR corrections. Hyperparameters for all the experiments are provided in Table 12, 13 of the Appendix.

## 3.1 Grammatical error correction (GEC)

We use Lang-8 (Mizumoto et al., 2011), NUCLE (Ng et al., 2014) and FCE (Yannakoudakis et al., 2011) corpora, which jointly comprise 1.2 million sentence pairs in English. The validation dataset comprises of 1381 sentence pairs from CoNLL-13 (Ng et al., 2013) test set. We initialize our GEC model with the publicly available BERT-LARGE[1] model that was pre-trained on Wikipedia (2500M words) and Book corpus (Zhu et al., 2015) (800M words) to predict 15% randomly masked words using its deep bidirectional context. Next, we perform 2 epochs of training on a synthetically perturbed version of the One-Billion-word corpus (Chelba et al., 2013). We refer to this as synthetic training. Details of how we create the synthetic corpus appear in Section A.3 of the appendix. Finally we fine-tune on the real GEC training corpus for 2 epochs. We use a batch size of 64 and learning rate 2e-5.

The edit space consists of copy, delete, 1000 appends, 1000 replaces and 29 transformations and their inverse. Arguments of Append and Replace operations mostly comprise punctuations, articles, pronouns, prepositions, conjunctions and verbs. Transformations perform inflections like add suffix `s, d, es, ing, ed` or replace suffix `s to ing`, `d to s`, etc. These transformations were chosen out of common replaces in the training data such that many replace edits map to only a few transformation edits, in order to help the model better generalize replaces across different words. In Section 3.1.4 we see that transformations increase the model's recall. The complete list of transformations appears in Table 10 of the Appendix. We evaluate on $F_{0.5}$ score over span-level corrections from the MaxMatch (M2) scorer (Dahlmeier and Ng, 2012) on CONLL-2014-test. Like most existing GEC systems, we invoke a spell-checker on the test sentences before applying our model. We also report GLEU$^+$ (Napoles et al., 2016) scores on JFLEG corpus (Napoles et al., 2017) to evaluate fluency.

[1] https://github.com/google-research/bert

| Work | CONLL-14 | | | JFLEG |
|---|---|---|---|---|
| | P | R | $F_{0.5}$ | GLEU$^+$ |
| Zhao et al. (2019) | 67.7 | 40.6 | 59.8 | - |
| Lichtarge et al. (2019) | 65.5 | 37.1 | 56.8 | 61.6 |
| Chollampatt and Ng (2018a) | 69.9 | 23.7 | 46.4 | 51.3 |
| PIE (This work) | 66.1 | 43.0 | 59.7 | 60.3 |

Table 2: Comparison of non-ensemble model numbers for various GEC models. Best non-ensemble results are reported for each model.

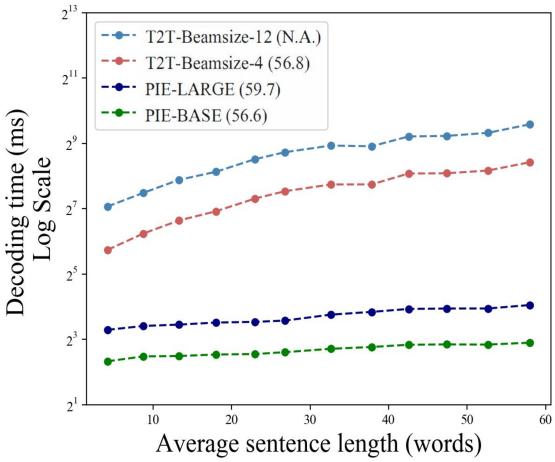


Figure 3: Comparing average decoding time in milliseconds (log scale) of PIE Vs transformer based ED model with two beam widths. Numbers in legend denote M2 score.

#### 3.1.1 Overall Results

Table 3 compares ensemble model results of PIE and other state of the art models, which all happen to be seq2seq ED models and also use ensemble decoding. For PIE, we simply average the probability distribution over edits from 5 independent ensembles. In Table-2 we compare non-ensemble numbers of PIE with the best available non-ensemble numbers of competing methods. On CoNLL-14 test-set our results are very close to the highest reported by Zhao et al. (2019). These results show that our parallel prediction model is competitive without incurring the overheads of beam-search and slow decoding of sequential models. GLEU$^+$ score, that rewards fluency, is somewhat lower for our model on the JFLEG test set because of parallel predictions. We do not finetune our model on the JFLEG dev set. We expect these to improve with re-ranking using a LM. All subsequent ablations and timing measurements are reported for non-ensemble models.

#### 3.1.2 Running Time Comparison

Parallel decoding enables PIE models to be considerably faster than ED models. In Figure 3 we compare wall-clock decoding time of PIE with 24 encoder layers (PIE-LARGE, $F_{0.5} = 59.7$), PIE with 12 encoder layers (PIE-BASE, $F_{0.5} = 56.6$) and competitive ED architecture by Lichtarge et al. (2019) with 6 encoder and 6 decoder layers (T2T, $F_{0.5} = 56.8$ ) on CoNLL-14 test set. All decoding experiments were run and measured on a n1-standard-2[2] VM instance with a single TPU shard (v-2.8). We observe that even PIE-LARGE is between a factor of 5 to 15 faster than an equivalent transformer-based ED model (T2T) with beam-size 4. The running time of PIE-LARGE increases sub-linearly with sentence length whereas the ED model's decoding time increases linearly.

#### 3.1.3 Impact of Iterative Refinement

We next evaluate the impact of iterative refinements on accuracy in Table 4. Out of 1312 sentences in the test set, only 832 sentences changed in the first round which were then fed to the second round where only 156 sentences changed, etc. The average number of refinement rounds per example was 2.7. In contrast, a sequential model on this dataset would require 23.2 steps corresponding to the average number of tokens in a sentence. The $F_{0.5}$ score increases from 57.9 to 59.5 at the end of the second iteration.

Table 5 presents some sentences corrected by PIE. We see that PIE makes multiple parallel edits in a round if needed. Also, we see how refinement over successive iterations captures output space dependency. For example, in the second sentence `interact` gets converted to `interacted` followed by insertion of `have` in the next round.

#### 3.1.4 Ablation study on the PIE Architecture

In this section we perform ablation studies to understand the importance of individual features of the PIE model.

**Synthetic Training** We evaluate the impact of training on the artifically generated GEC corpus in row 2 of Table 6. We find that without it the $F_{0.5}$ score is 3.4 points lower.

**Factorized Logits** We evaluate the gains due to

[2] https://cloud.google.com/compute/docs/machine-types#standard_machine_types

| Work | Method | CONLL-14 | | | JFLEG |
|---|---|---|---|---|---|
| | | P | R | $F_{0.5}$ | GLEU$^+$ |
| Zhao et al. (2019) | Transformer + Pre-training + LM + Spellcheck + Ensemble Decoding (4) | 71.6 | 38.7 | 61.2 | 61.0 |
| Lichtarge et al. (2019) | Transformer + Pre-training + Iterative refinement + Ensemble Decoding (8) | 66.7 | 43.9 | 60.4 | 63.3 |
| Grundkiewicz and Junczys-Dowmunt (2018) | SMT + NMT (Bi-GRU) + LM + Ensemble Decoding (4) + Spellcheck | 66.8 | 34.5 | 56.3 | 61.5 |
| Chollampatt and Ng (2018a) | CNN + LM + Ensemble Decoding (4) + Spellcheck | 65.5 | 33.1 | 54.8 | 57.5 |
| PIE (This work) | Transformer + LM + Pre-training + Spellcheck + Iterative refinement + Ensembles (5) | 68.3 | 43.2 | 61.2 | 61.0 |

Table 3: Comparison of recent GEC models trained using publicly available corpus. All the methods here except ours perform sequential decoding. Precision and Recall are represented by P and R respectively.

| | R1 | R2 | R3 | R4 |
|---|---|---|---|---|
| #edited | 832 | 156 | 20 | 5 |
| P | 64.5 | 65.9 | 66.1 | 66.1 |
| R | 41.0 | 42.9 | 43.0 | 43.0 |
| $F_{0.5}$ | 57.9 | 59.5 | 59.7 | 59.7 |

Table 4: Statistics on successive rounds of iterative refinement. First row denotes number changed sentences (out of 1312) with each round on CONLL-14 (test).

our edit-factorized BERT model (Section 2.3.2) over the default BERT model (Section 2.3.1). In Table 6 (row 3) we show that compared to the factorized model (row 2) we get a 1.2 point drop in $F_{0.5}$ score in absence of factorization.

**Inserts as Appends** on the preceding word was another important design choice. The alternative of predicting `insert` independently at each gap with a null token added to $\Sigma_a$ performs 2.7 $F_{0.5}$ points poorly (Table 6 row 4 vs row 2).

**Transformation edits** are significant as we observe a 6.3 drop in recall without them (row 5).

**Impact of Language Model** We evaluate the benefit of starting from BERT's pre-trained LM by reporting accuracy from an un-initialized network (row 6). We observe a 20 points drop in $F_{0.5}$ establishing the importance of LMs in GEC.

**Impact of Network Size** We train the BERT-Base model with one-third fewer parameters than BERT-LARGE. From Table 6 (row 7 vs 1) we see once again that size matters in deep learning!

| | |
|---|---|
| x | I started invoving into Facebook one years ago . |
| PIE1 | I started ***involving in*** Facebook one ***year*** ago . |
| PIE2 | I started ***involved*** in Facebook one year ago . |
| PIE3 | I started ***getting*** involved in Facebook one year ago . |
| x | Since ancient times , human interact with others face by face . |
| PIE1 | Since ancient times , humans ***interacted*** with others face ***to*** face . |
| PIE2 | Since ancient times , humans ***have*** interacted with others face to face . |
| x | However , there are two sides of stories always . |
| PIE1 | However , there are ***always*** two sides ***to*** stories ~~always~~ . |
| PIE2 | However , there are always two sides to ***the*** stories . |
| PIE3 | However , there are always two sides to the ***story*** . |

Table 5: Parallel and Iterative edits done by PIE.

### 3.2 More Sequence Transduction Tasks

We demonstrate the effectiveness of PIE model on two additional local sequence transduction tasks recently used in (Ribeiro et al., 2018).

**Spell Correction** We use the twitter spell correction dataset (Aramaki, 2010) which consists of 39172 pairs of original and corrected words obtained from twitter. We use the same train-dev-valid split as (Ribeiro et al., 2018) (31172/4000/4000). We tokenize on characters, and our vocabulary $\Sigma$ and $\Sigma_a$ comprises the 26 lower cased letters of English.

**Correcting OCR errors** We use the Finnish OCR data set[3] by (Silfverberg et al., 2016) comprising words extracted from Early Modern

[3] https://github.com/mpsilfve/ocrpp

| # | Methods | P | R | $F_{0.5}$ |
|---|---|---|---|---|
| 1 | PIE | 66.1 | 43.0 | 59.7 |
| 2 | − Synthetic training | 67.2 | 34.2 | 56.3 |
| 3 | −Factorized-logits | 66.4 | 32.8 | 55.1 |
| 4 | −Append +Inserts | 57.4 | 42.5 | 53.6 |
| 5 | −Transformations | 63.6 | 27.9 | 50.6 |
| 6 | −LM Pre-init | 48.8 | 18.3 | 36.6 |
| 7 | PIE on BERT-Base | 67.8 | 34.0 | 56.6 |

Table 6: Ablation study on the PIE model.

| Methods | Spell | OCR |
|---|---|---|
| Seq2Seq (Soft, lstm) | 46.3 | 79.9 |
| Seq2Seq (Hard,lstm) | 52.2 | 58.4 |
| Seq2Seq (Soft-T2T) | 67.6 | 84.5 |
| Ribeiro2018 (best model) | 54.1 | 81.8 |
| PIE | 67.0 | 87.6 |

Table 7: Comparing PIE with Seq2Seq models (top-part) and Ribeiro's parallel model on two other local sequence transduction tasks.

Finnish corpus of OCR processed newspaper text. We use the same train-dev-test splits as provided by (Silfverberg et al., 2016). We tokenize on characters in the word. For a particular split, our vocabulary $\Sigma$ and $\Sigma_a$ comprises of all the characters seen in the training data of the split.

**Architecture** For all the tasks in this section, PIE is a 4 layer self-attention transformer with 200 hidden units, 400 intermediate units and 4 attention heads. No $\mathcal{L}$ pre-initialization is done. Also, number of iterations of refinements is set to 1.

**Results** Table 7 presents whole-word 0/1 accuracy for these tasks on PIE and the following methods: Ribeiro et al. (2018)'s local transduction model (described in Section 4), and LSTM based ED models with hard monotonic attention (Aharoni and Goldberg, 2017) and soft-attention (Bahdanau et al., 2015) as reported in (Ribeiro et al., 2018). In addition, for a fair decoding time comparison, we also train a transformer-based ED model referred as Soft-T2T with 2 encoder, 2 decoder layers for spell-correction and 2 encoder, 1 decoder layer for OCR correction. We observe that PIE's accuracy is comparable with ED models in both the tasks. Table 8 compares decoding speed of PIE with Soft-T2T in words/second. Since more than 90% of words have fewer than 9 tokens and the token vocabulary $\Sigma$ is small, decoding speed-ups of PIE over ED model on these tasks is modest compared to GEC.

| | PIE | Soft-T2T |
|---|---|---|
| Spell | 80.43 | 36.62 |
| OCR | 65.44 | 43.02 |

Table 8: Wall-clock decoding speed in words/second of PIE and a comparable Seq2Seq T2T model.

## 4 Related Work

**Grammar Error Correction (GEC)** is an extensively researched area in NLP. See Ng et al. (2013) and Ng et al. (2014) for past shared tasks on GEC, and this[4] website for current progress. Approaches attempted so far include rules (Felice et al., 2014), classifiers (Rozovskaya and Roth, 2016), statistical machine translation (SMT) (Junczys-Dowmunt and Grundkiewicz, 2016), neural ED models (Chollampatt and Ng, 2018a; Junczys-Dowmunt et al., 2018; Ge et al., 2018a), and hybrids (Grundkiewicz and Junczys-Dowmunt, 2018). All recent neural approaches are sequential ED models that predict either word sequences (Zhao et al., 2019; Lichtarge et al., 2019) or character sequences (Xie et al., 2016) using either multi-layer RNNs (Ji et al., 2017; Grundkiewicz and Junczys-Dowmunt, 2018) or CNNs(Chollampatt and Ng, 2018a; Ge et al., 2018a) or Transformers (Junczys-Dowmunt et al., 2018; Lichtarge et al., 2019). Our sequence labeling formulation is similar to (Yannakoudakis et al., 2017) and (Kaili et al., 2018) but the former uses it to only detect errors and the latter only corrects five error-types using separate classifiers. Edits have been exploited in earlier GEC systems too but very unlike our method of re-architecting the core model to label input sequence with edits. Schmaltz et al. (2017) interleave edit tags in target tokens but use seq2seq learning to predict the output sequence. Chollampatt and Ng (2018a) use edits as features for rescoring seq2seq predictions. Junczys-Dowmunt et al. (2018) use an edit-weighted MLE objective to emphasise corrective edits during seq2seq learning. Stahlberg et al. (2019) use finite state transducers, whose state transitions denote possible edits, built from an unlabeled corpus to constrain the output of a neural beam decoder to a small GEC-feasible space.

[4] https://nlpprogress.com/english/grammatical_error_correction.html

**Parallel decoding in neural machine translation** Kaiser et al. (2018) achieve partial parallelism by first generating latent variables sequentially to model dependency. Stern et al. (2018) use a parallel generate-and-test method with modest speed-up. Gu et al. (2018) generate all tokens in parallel but initialize decoder states using latent fertility variables to determine number of replicas of an encoder state. We achieve the effect of fertility using delete and append edits. Lee et al. (2018) generate target sequences iteratively but require the target sequence length to be predicted at start. In contrast our in-place edit model allows target sequence length to change with appends.

**Local Sequence Transduction** is handled in Ribeiro et al. (2018) by first predicting insert slots in $\mathbf{x}$ using learned insertion patterns and then using a sequence labeling task to output tokens in $\mathbf{x}$ or a special token `delete`. Instead, we output edit operations including word transformations. Their pattern-based insert pre-slotting is unlikely to work for more challenging tasks like GEC. Koide et al. (2018) design a special edit-invariant neural network for being robust to small edit changes in input biological sequences. This is a different task than ours of edit prediction. Yin et al. (2019) is about neural representation of edits specifically for structured objects like source code. This is again a different problem than ours.

## 5 Conclusion

We presented a parallel iterative edit (PIE) model for local sequence transduction with a focus on the GEC task. Compared to the popular encoder-decoder models that perform sequential decoding, parallel decoding in the PIE model yields a factor of 5 to 15 reduction in decoding time. The PIE model employs a number of ideas to match the accuracy of sequential models in spite of parallel decoding: it predicts in-place edits using a carefully designed edit space, iteratively refines its own predictions, and effectively reuses state-of-the-art pre-trained bidirectional language models. In the future we plan to apply the PIE model to more ambitious transduction tasks like translation.

**Acknowledgements** This research was partly sponsored by a Google India AI/ML Research Award and Google PhD Fellowship in Machine Learning. We gratefully acknowledge Google's TFRC program for providing us Cloud-TPUs. We thank Varun Patil for helping us improve the speed of pre-processing and synthetic-data generation pipelines.

## References

Roee Aharoni and Yoav Goldberg. 2017. Morphological inflection generation with hard monotonic attention. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 2004–2015.

Eiji Aramaki. 2010. Typo corpus. `http://luululu.com/tweet/`.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *ICLR*.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Shamil Chollampatt and Hwee Tou Ng. 2018a. A multilayer convolutional encoder-decoder neural network for grammatical error correction. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

Shamil Chollampatt and Hwee Tou Ng. 2018b. Neural quality estimation of grammatical error correction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2528–2539.

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 568–572. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Mariano Felice, Zheng Yuan, Øistein E Andersen, Helen Yannakoudakis, and Ekaterina Kochmar. 2014. Grammatical error correction using hybrid systems and type filtering. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 15–24.

Tao Ge, Furu Wei, and Ming Zhou. 2018a. Fluency boost learning and inference for neural grammatical error correction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1055–1065.

Tao Ge, Furu Wei, and Ming Zhou. 2018b. Reaching human-level performance in automatic grammatical error correction: An empirical study. *arXiv preprint arXiv:1807.01270*.

Roman Grundkiewicz and Marcin Junczys-Dowmunt. 2018. Near human-level performance in grammatical error correction with hybrid machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 284–290.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O.K. Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *International Conference on Learning Representations (ICLR)*.

Jianshu Ji, Qinlong Wang, Kristina Toutanova, Yongen Gong, Steven Truong, and Jianfeng Gao. 2017. A nested attention neural hybrid model for grammatical error correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 753–762, Vancouver, Canada. Association for Computational Linguistics.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2016. Phrase-based machine translation is state-of-the-art for automatic grammatical error correction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1546–1556.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Shubha Guha, and Kenneth Heafield. 2018. Approaching neural grammatical error correction as a low-resource machine translation task. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 595–606.

Zhu Kaili, Chuan Wang, Ruobing Li, Yang Liu, Tianlei Hu, and Hui Lin. 2018. A simple but effective classification model for grammatical error correction. *arXiv preprint arXiv:1807.00488*.

Lukasz Kaiser, Samy Bengio, Aurko Roy, Ashish Vaswani, Niki Parmar, Jakob Uszkoreit, and Noam Shazeer. 2018. Fast decoding in sequence models using discrete latent variables. In *Proceedings of the 35th International Conference on Machine Learning*, pages 2390–2399.

Satoshi Koide, Keisuke Kawano, and Takuro Kutsuna. 2018. Neural edit operations for biological sequences. In *Advances in Neural Information Processing Systems 31*.

D. Koller and N. Friedman. 2009. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710.

Jared Lichtarge, Chris Alberti, Shankar Kumar, Noam Shazeer, Niki Parmar, and Simon Tong. 2019. Corpora generation for grammatical error correction. *arXiv preprint arXiv:1904.05780*.

Tomoya Mizumoto, Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. 2011. Mining revision log of language learning sns for automated japanese error correction of second language learners. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 147–155.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2016. Gleu without tuning. *arXiv preprint arXiv:1605.02592*.

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2017. Jfleg: A fluency corpus and benchmark for grammatical error correction. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 229–234, Valencia, Spain. Association for Computational Linguistics.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14.

Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. 2013. The conll-2013 shared task on grammatical error correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–12, Sofia, Bulgaria. Association for Computational Linguistics.

Aaron van den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George van den Driessche, Edward Lockhart, Luis Cobo, Florian Stimberg, Norman Casagrande, Dominik Grewe, Seb Noury, Sander Dieleman, Erich Elsen, Nal Kalchbrenner, Heiga Zen, Alex Graves, Helen King, Tom Walters, Dan Belov, and Demis Hassabis. 2018. Parallel WaveNet: Fast high-fidelity speech synthesis. In *Proceedings of the 35th International Conference on Machine Learning*, pages 3918–3926.

Joana Ribeiro, Shashi Narayan, Shay B. Cohen, and Xavier Carreras. 2018. Local string transduction as sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics (ACL-COLING)*.

Alla Rozovskaya and Dan Roth. 2016. Grammatical error correction: Machine translation and classifiers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2205–2215.

Allen Schmaltz, Yoon Kim, Alexander M. Rush, and Stuart M. Shieber. 2017. Adapting sequence models for sentence correction. In *EMNLP*, pages 2807–2813.

Miikka Silfverberg, Pekka Kauppinen, Krister Lindén, et al. 2016. Data-driven spelling correction using weighted finite-state methods. In *The 54th Annual Meeting of the Association for Computational Linguistics Proceedings of the SIGFSM Workshop on Statistical NLP and Weighted Automata*. ACL.

Felix Stahlberg, Christopher Bryant, and Bill Byrne. 2019. Neural grammatical error correction with finite state transducers. *arXiv preprint arXiv:1903.10625*.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. 2018. Blockwise parallel decoding for deep autoregressive models. In *Advances in Neural Information Processing Systems 31*.

Ziang Xie, Anand Avati, Naveen Arivazhagan, Dan Jurafsky, and Andrew Y. Ng. 2016. Neural language correction with character-based attention. *CoRR*, abs/1603.09727.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading esol texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 180–189. Association for Computational Linguistics.

Helen Yannakoudakis, Marek Rei, Øistein E Andersen, and Zheng Yuan. 2017. Neural sequence-labelling models for grammatical error correction. In *EMNLP*, pages 2795–2806.

Pengcheng Yin, Graham Neubig, Miltiadis Allamanis, Marc Brockschmidt, and Alexander L. Gaunt. 2019. Learning to represent edits. In *International Conference on Learning Representations*.

Wei Zhao, Liang Wang, Kewei Shen, Ruoyu Jia, and Jingming Liu. 2019. Improving grammatical error correction via pre-training a copy-augmented architecture with unlabeled data. *arXiv preprint arXiv:1903.00138*.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

# ARAML: A Stable Adversarial Training Framework for Text Generation

**Pei Ke**[*], **Fei Huang**[*], **Minlie Huang**[†], **Xiaoyan Zhu**
Institute for Artificial Intelligence, State Key Lab of Intelligent Technology and Systems
Beijing National Research Center for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
kepei1106@outlook.com, f-huang18@mails.tsinghua.edu.cn
aihuang@tsinghua.edu.cn, zxy-dcs@tsinghua.edu.cn

## Abstract

Most of the existing generative adversarial networks (GAN) for text generation suffer from the instability of reinforcement learning training algorithms such as policy gradient, leading to unstable performance. To tackle this problem, we propose a novel framework called Adversarial Reward Augmented Maximum Likelihood (ARAML). During adversarial training, the discriminator assigns rewards to samples which are acquired from a stationary distribution near the data rather than the generator's distribution. The generator is optimized with maximum likelihood estimation augmented by the discriminator's rewards instead of policy gradient. Experiments show that our model can outperform state-of-the-art text GANs with a more stable training process.

## 1 Introduction

Natural text generation, as a key task in NLP, has been advanced substantially thanks to the flourish of neural models (Bengio et al., 2003; Mikolov et al., 2010). Typical frameworks such as sequence-to-sequence (seq2seq) have been applied to various generation tasks, including machine translation (Sutskever et al., 2014) and dialogue generation (Vinyals and Le, 2015). The standard paradigm to train such neural models is maximum likelihood estimation (MLE), which maximizes the log-likelihood of observing each word in the text given the ground-truth proceeding context (Graves, 2013).

Although widely used, MLE suffers from the exposure bias problem (Bengio et al., 2015; Ranzato et al., 2016): during test, the model sequentially predicts the next word conditioned on its previous generated words while during training conditioned on ground-truth words. To tackle this problem, generative adversarial networks (GAN) with reinforcement learning (RL) training approaches have been introduced to text generation tasks (Yu et al., 2017; Che et al., 2017; Lin et al., 2017; Fedus et al., 2018; Guo et al., 2018; Shi et al., 2018; Xu et al., 2018), where the discriminator is trained to distinguish real and generated text samples to provide reward signals for the generator, and the generator is optimized via policy gradient (Yu et al., 2017).

However, recent studies have shown that potential issues of training GANs on discrete data are more severe than exposure bias (Semeniuta1 et al., 2018; Caccia et al., 2018). One of the fundamental issues when generating discrete text samples with GANs is training instability. Updating the generator with policy gradient always leads to an unstable training process because it's difficult for the generator to derive positive and stable reward signals from the discriminator even with careful pretraining (Che et al., 2017). As a result, the generator gets lost due to the high variance of reward signals and the training process may finally collapse (Li et al., 2017).

In this paper, we propose a novel adversarial training framework called Adversarial Reward Augmented Maximum Likelihood (ARAML) to deal with the instability issue of training GANs for text generation. At each iteration of adversarial training, we first train the discriminator to assign higher rewards to real data than to generated samples. Then, inspired by reward augmented maximum likelihood (RAML) (Norouzi et al., 2016), the generator is updated on the samples acquired from a stationary distribution with maximum likelihood estimation (MLE), weighted by the discriminator's rewards. This stationary distribution is designed to guarantee that training samples are surrounding the real data, thus the exploration space of our generator is indeed restricted by the

[*] Equal contribution
[†] Corresponding author: Minlie Huang



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4271–4281,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

MLE training objective, resulting in more stable training. Compared to other text GANs with RL training techniques, our framework acquires samples from the stationary distribution rather than the generator's distribution, and uses RAML training paradigm to optimize the generator instead of policy gradient. Our contributions are mainly as follows:

- We analyze the fundamental issue of current GANs for text generation from the perspectives of training instability.
- We propose a novel framework called Adversarial Reward Augmented Maximum Likelihood (ARAML), which incorporates stable RAML training into adversarial training paradigm. Experimental results on three text generation tasks show the effectiveness of our method.

## 2 Related Work

Recently, text generation has been widely studied with neural models trained with maximum likelihood estimation (Graves, 2013). However, MLE tends to generate universal text (Li et al., 2016). Various methods have been proposed to enhance the generation quality by refining the objective function (Li et al., 2016; Mou et al., 2016) or modifying the generation distribution with external information like topic (Xing et al., 2017), sentence type (Ke et al., 2018), emotion (Zhou et al., 2018a) and knowledge (Zhou et al., 2018b).

As mentioned above, MLE suffers from the exposure bias problem (Bengio et al., 2015; Ranzato et al., 2016). Thus, reinforcement learning has been introduced to text generation tasks such as policy gradient (Ranzato et al., 2016) and actor-critic (Bahdanau et al., 2017). (Norouzi et al., 2016) proposed an efficient and stable approach called Reward Augmented Maximum Likelihood (RAML), which connects the log-likelihood and expected rewards to incorporate MLE training objective into RL framework.

Since some text generation tasks have no explicit metrics to be directly optimized, adversarial training has been applied to generating discrete text samples with a discriminator to learn a proper reward. For instance, SeqGAN (Yu et al., 2017) devised a discriminator to distinguish the real data and generated samples, and a generator to maximize the reward from the discriminator via policy gradient. Other variants of GANs have been proposed to improve the generator or the discriminator. To improve the generator, MaliGAN (Che et al., 2017) developed a normalized maximum likelihood optimization target for the generator to stably model the discrete sequences. LeakGAN (Guo et al., 2018) guided the generator with reward signals leaked from the discriminator at all generation steps to deal with long text generation task. MaskGAN (Fedus et al., 2018) employed an actor-critic architecture to make the generator fill in missing text conditioned on the surrounding context, which is expected to mitigate the problem of mode collapse. As for the discriminator, RankGAN (Lin et al., 2017) replaced traditional discriminator with a ranker to learn the relative ranking information between the real texts and generated ones. Inverse reinforcement learning (Shi et al., 2018) used a trainable reward approximator as the discriminator to provide dense reward signals at each generation step. DPGAN (Xu et al., 2018) introduced a language model based discriminator and regarded cross-entropy as rewards to promote the diversity of generation results.

The most similar works to our model are RAML (Norouzi et al., 2016) and MaliGAN (Che et al., 2017): 1) Compared with RAML, our model adds a discriminator to learn the reward signals instead of choosing existing metrics as rewards. We believe that our model can adapt to various text generation tasks, particularly those without explicit evaluation metrics. 2) Unlike MaliGAN, we acquire samples from a fixed distribution near the real data rather than the generator's distribution, which is expected to make the training process more stable.

## 3 Model

### 3.1 Task Definition and Model Overview

Text generation can be formulated as follows: given the real data distribution $P_{\text{data}}(X)$, the task is to train a generative model $G_\theta$ where $P_{G_\theta}(X)$ can fit $P_{\text{data}}(X)$ well. In this formulation, $X = x_1 x_2 \cdots x_m$ and $x_t (1 \leq t \leq m)$ denotes a word in the vocabulary $\mathcal{V}$.

Figure 1 shows the overview of our model ARAML. This adversarial training framework consists of two phases: 1) The discriminator is trained to assign higher rewards to real data than to generated data. 2) The generator is trained on the samples acquired from a stationary distribu-

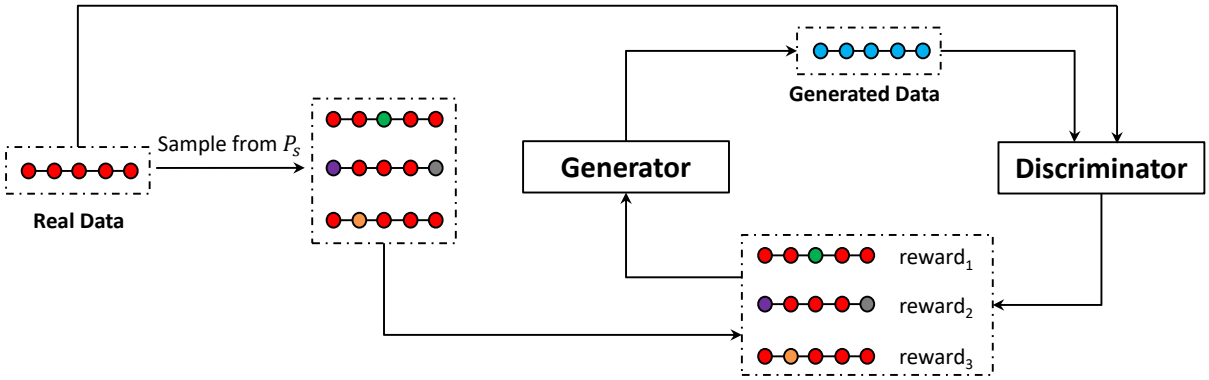


Figure 1: Overview of ARAML. The training samples are acquired from a stationary distribution $P_s$ based on the real data. The generator is then trained on the samples augmented by the discriminator's rewards. The discriminator is trained to distinguish real data and generated data.

tion with reward augmented MLE training objective. This training paradigm of the generator indeed constrains the search space with the MLE training objective, which alleviates the issue of unstable training.

### 3.2 Discriminator

The discriminator $D_\phi$ aims to distinguish real data and generated data like other GANs. Inspired by Least-Square GAN (Mao et al., 2017), we devise the loss function as follows:

$$\mathcal{L}_{D_\phi} = \frac{1}{2}\mathbb{E}_{X\sim P_{\text{data}}(X)}\left[(D_\phi(X)-1)^2\right] + \frac{1}{2}\mathbb{E}_{X\sim P_{G_\theta}(X)}\left[(D_\phi(X))^2\right] \quad (1)$$

This loss function forces the discriminator to assign higher rewards to real data than to generated data, so the discriminator can learn to provide more proper rewards as the training proceeds.

### 3.3 Generator

The training objective of our generator $G_\theta$ is derived from the objective of other discrete GANs with RL training method:

$$\mathcal{L}_{\text{RL},\theta} = -\mathbb{E}_{X\sim P_{G_\theta}(X)}[r_\phi(X)] - \tau\mathbb{H}(P_{G_\theta}(X)) \quad (2)$$

where $r_\phi(X)$ denotes the rewards from the discriminator $D_\phi$ and the entropy regularized term $\mathbb{H}(P_{G_\theta}(X))$ encourages $G_\theta$ to generate diverse text samples. $\tau$ is a temperature hyper-parameter to balance these two terms.

As mentioned above, discrete GANs suffer from the instability issue due to policy gradient, thus they are consequently difficult to train. Inspired by RAML (Norouzi et al., 2016), we introduce an exponential payoff distribution $Q_\phi(X)$ to connect RL loss with RAML loss:

$$Q_\phi(X) = \frac{1}{Z}\exp(r_\phi(X)/\tau) \quad (3)$$

where $Z = \sum_X \exp(r_\phi(X)/\tau)$. Thus, we can rewrite $\mathcal{L}_{\text{RL},\theta}$ with $P_{G_\theta}(X)$ and $Q_\phi(X)$ as follows:

$$\mathcal{L}_{\text{RL},\theta} = \tau KL(P_{G_\theta}(X)||Q_\phi(X)) + constant \quad (4)$$

Following RAML, we remove the constant term and optimize the KL divergence in the opposite direction:

$$\begin{aligned}\mathcal{L}_{\text{RAML},\theta} &= KL(Q_\phi(X)||P_{G_\theta}(X)) \\ &= -\mathbb{E}_{X\sim Q_\phi(X)}[\log P_{G_\theta}(X)] - \mathbb{H}(Q_\phi(X)) \\ &= -\mathbb{E}_{X\sim Q_\phi(X)}[\log P_{G_\theta}(X)] + constant \quad (5)\end{aligned}$$

where $\mathbb{H}(Q_\phi(X))$ is a constant in the training phase of the generator. It has been proved that $\mathcal{L}_{\text{RL},\theta}$ and $\mathcal{L}_{\text{RAML},\theta}$ are equivalent up to their first order Taylor approximations, and they have the same global optimum (Norouzi et al., 2016). $\mathcal{L}_{\text{RAML},\theta}$ can be trained in a MLE-like fashion but sampling from the distribution $Q_\phi(X)$ is intractable in the adversarial setting, because $Q_\phi(X)$ varies with the discriminator $D_\phi$. Thus, we introduce importance sampling to separate sampling process from $D_\phi$ and obtain the final loss function:

$$\mathcal{L}_{G_\theta} = -\mathbb{E}_{X\sim P_s(X)}[W_\phi(X)\log P_{G_\theta}(X)] \quad (6)$$

where $P_s(X)$ denotes a stationary distribution and $W_\phi(X) \propto Q_\phi(X)/P_s(X)$. To optimize this loss function, we first construct the fixed distribution $P_s(X)$ to get samples, and devise the proper reward function $r_\phi(X)$ to train the generator in a stable and effective way.

#### 3.3.1 Sampling

We construct the distribution $P_s$ based on $P_{\text{data}}$:

$$P_s(X) = \mathbb{E}_{X \sim P_{\text{data}}(X)}[P_s(X_s|X)] \quad (7)$$

In this way, $P_s(X_s|X)$ can be designed to guarantee that $P_s(X)$ is near $P_{\text{data}}(X)$, leading to a more stable training process. To obtain a new sample $X_s$ from a real data sample $X$, we can design three steps which contain sampling an edit distance $d$, the positions $\{p_1, p_2, \cdots, p_d\}$ for substitution and the new words $\{w_1, w_2, \cdots, w_d\}$ filled into the corresponding positions. Thus, $P_s(X_s|X)$ can be decomposed into three terms:

$$\begin{aligned} P_s(X_s|X) &= P(d, p, w|X) \\ &= P(d|X)P(p|X, d)P(w|X, d, p) \end{aligned} \quad (8)$$

The first step is to sample an edit distance based on a real data sample $X$, where $X = x_1 x_2 \cdots x_m$ is a sequence of length $m$. The number of sentences which have the edit distance $e$ to some input sentence can be computed approximately as below:

$$c(e, m) = \binom{m}{e} \cdot (|\mathcal{V}| - 1)^e \quad (9)$$

where $c(e, m)$ denotes the number of sentences which have an edit distance $e (e \in \{0, 1, ..., m\})$ to a sentence of length $m$, and $|\mathcal{V}|$ indicates the size of vocabulary. We then follow (Norouzi et al., 2016) to re-scale the counts by $\exp\{-e/\tau\}$ and do normalization, so that we can sample an edit distance $d^*$ from:

$$P(d = d^*|X) = \frac{\exp\{-d^*/\tau\}c(d^*, m)}{\sum_{e=0}^{m} \exp\{-e/\tau\}c(e, m)} \quad (10)$$

where $\tau$, as a temperature hyper-parameter, restricts the search space surrounding the original sentence. Larger $\tau$ brings more samples with long edit distances.

The next step is to select positions for substitution based on the sampled edit distance $d^*$. Intuitively, we can randomly choose $d^*$ distinct positions in $X$ to be replaced by new words. The probability of choosing the position $p^*$ is calculated as follows:

$$P(p = p^*|X, d = d^*) = \frac{d^*}{m} \quad (11)$$

Following this sampling strategy, we can obtain the position set $\{p_1, p_2, \cdots, p_{d^*}\}$. This strategy approximately guarantees that the edit distance between a new sentence and the original sentence is $d^*$.

At the final step, our model determines new words for substitution at each sampled position $p_j (j = 1, 2, ..., d^*)$. We can formulate this sampling process from the original sequence $X$ to a new sample $X_s$ as a sequential transition $X = X_0 \to X_1 \to \cdots \to X_{d^*} = X_s$. At each step from $X_{j-1}$ to $X_j$ $(j = 1, \cdots, d^*)$, we first sample a new word $w_j$ from the distribution $P(w|X_{j-1}, p = p_j)$, then replace the old word at position $p_j$ of $X_{j-1}$ to obtain $X_j$. The whole sampling process can be decomposed as follows:

$$\begin{aligned} &P(w|X, d = d^*, p = \{p_1, p_2, \cdots, p_{d^*}\}) \\ &\qquad = \prod_{j=1}^{d^*} P(w_j|X_{j-1}, p = p_j) \end{aligned} \quad (12)$$

There are two common sampling strategies to model $P(w|X_{j-1}, p = p_j)$, i.e. random sampling and constrained sampling. Random sampling strategy samples a new word $w_j$ according to the uniform distribution over the vocabulary $\mathcal{V}$ (Norouzi et al., 2016), while constrained sampling strategy samples $w_j$ to maximize the language model score of the target sentence $X_j$ (Su et al., 2018; Miao et al., 2019). Here, we adopt constrained sampling in our model and compare the performances of two strategies in the experiment.

#### 3.3.2 Training

We devise the reward function $r_\phi(X)$ according to the discriminator's output $D_\phi(X)$ and the stationary distribution $P_s(X)$:

$$r_\phi(X) = \tau \cdot [\log P_s(X) + D_\phi(X)] \quad (13)$$

Intuitively, this reward function encourages the generator to generate sentences with large sampling probability and high rewards from the discriminator. Thus, the weight of samples $W_\phi(X)$ can be calculated as follows:

$$W_\phi(X) \propto \frac{Q_\phi(X)}{P_s(X)} \propto \exp\{D_\phi(X)\} \quad (14)$$

So far, we can successfully optimize the generator's loss $\mathcal{L}_{G_\theta}$ via Equation 6. This training paradigm makes our generator avoid possible variances caused by policy gradient and get more stable reward signals from the discriminator, because our generator is restricted to explore the training samples near the real data.

**Algorithm 1** Adversarial Reward Augmented Maximum Likelihood

**Require:**
Total adversarial training iterations: $N\_iters$
Steps of training generator: $G\_steps$
Steps of training discriminator: $D\_steps$

Pre-train the generator $G_\theta$ with MLE loss
Generate samples from $P_{G_\theta}$
Pre-train the discriminator $D_\phi$ via Eq.(1)
Construct $P_s(X)$ via Eq.(7) - Eq.(12)
**for** each $s = 1, 2, ..., N\_iters$ **do**
  **for** each $j = 1, 2, ..., G\_steps$ **do**
    Update $G_\theta$ via Eq.(6)
  **end for**
  **for** each $k = 1, 2, ..., D\_steps$ **do**
    Update $D_\phi$ via Eq.(1)
  **end for**
**end for**

### 3.4 Extension to Conditional Text Generation

We have shown our adversarial training framework for text generation tasks without an input. Actually, it can also be extended to conditional text generation tasks like dialogue generation. Given the data distribution $P_{\text{data}}(C, X)$ where $C, X$ denote contexts and responses respectively, the objective function of ARAML's generator can be modified as below:

$$\mathcal{L}_{G_\theta} = -\mathbb{E}_{(C,X)\sim P_{\text{data}}(C,X)}\big[ \mathbb{E}_{X_s\sim P_s(X_s|C,X)}\left[W_\phi(C, X_s)\log P_{G_\theta}(X_s|C)\right]\big] \tag{15}$$

where $W_\phi(C, X_s) \propto \exp\{D_\phi(C, X_s)\}$ and $D_\phi(C, X_s)$ is trained to distinguish whether $X_s$ is the true response to $C$.

### 3.5 Comparison with RAML and MaliGAN

The most similar works to our framework are RAML (Norouzi et al., 2016) and MaliGAN (Che et al., 2017). The main difference among them is the training objective of their generators. We have shown different objective functions in Table 1. For comparison, we use the form with no input for all the three models.

Our model is greatly inspired by RAML, which gets samples from a non-parametric distribution $Q(X)$ constructed based on a specific reward. Compared to RAML, our reward comes from a learnable discriminator which varies as the adversarial training proceeds rather than a specific reward function. This difference equips our framework with the ability to adapt to the text generation tasks with no explicit evaluation metrics as rewards.

Our model is also similar to MaliGAN, which gets samples from the generator's distribution. In MaliGAN's training objective, $G_{\theta'}$ also indicates the generator's distribution but it's used in the sampling phase and fixed at each optimization step. The weight of samples $W'_\phi(X) \propto \frac{D_\phi(X)}{1-D_\phi(X)}$. Different from our model, MaliGAN acquires samples from the generator's distribution $P_{G_{\theta'}}$, which usually brings samples with low rewards even with careful pre-training for the generator, leading to training instability. Instead, our framework gets samples from a stationary distribution $P_s$ around real data, thus our training process is more stable.

| Model | Training Objective of Generator |
|---|---|
| RAML | $\mathcal{L}_{G_\theta} = -\mathbb{E}_{X\sim Q(X)}[\log P_{G_\theta}(X)]$ |
| MaliGAN | $\mathcal{L}_{G_\theta} = -\mathbb{E}_{X\sim P_{G_{\theta'}}(X)}[W'_\phi(X)\log P_{G_\theta}(X)]$ |
| ARAML | $\mathcal{L}_{G_\theta} = -\mathbb{E}_{X\sim P_s(X)}[W_\phi(X)\log P_{G_\theta}(X)]$ |

Table 1: Training objectives of generators for RAML, MaliGAN and ARAML.

## 4 Experiment

### 4.1 Datasets

| Dataset | Amount(Train/Test) | Vocabulary | Length |
|---|---|---|---|
| COCO | 80,000/5,000 | 4,839 | 12.8 |
| EMNLP2017 | 49,996/10,000 | 5,721 | 27.8 |
| WeiboDial | 100,000/5,000 | 7,998 | 7.3/10.8 |

Table 2: Statistics of COCO, EMNLP2017 WMT and WeiboDial. The average lengths 7.3/10.8 of WeiboDial indicate the lengths of posts and responses, respectively.

We evaluated ARAML on three datasets: COCO image caption dataset (Chen et al., 2015), EMNLP2017 WMT dataset[1] and WeiboDial single-turn dialogue dataset (Qian et al.,

[1] http://statmt.org/wmt17/translation-task.html

2018). COCO and EMNLP2017 WMT are the common benchmarks with no input to evaluate the performance of discrete GANs, and we followed the existing works to preprocess these datasets (Shi et al., 2018; Guo et al., 2018). WeiboDial, as a dialogue dataset, was applied to test the performance of our model with input trigger. We simply removed post-response pairs containing low-frequency words and randomly selected a subset for our training/test set. The statistics of three datasets are presented in Table 2.

### 4.2 Baselines

We compared our model with MLE, RL and GAN baselines. Since COCO and EMNLP2017 WMT don't have input while WeiboDial regards posts as input, we chose the following baselines respectively:

**MLE**: a RNN model trained with MLE objective (Graves, 2013). Its extension, Seq2Seq, can work on the dialogue dataset (Sutskever et al., 2014).

**SeqGAN**: The first text GAN model that updates the generator with policy gradient based on the rewards from the discriminator (Yu et al., 2017).

**LeakGAN**: A variant of SeqGAN that provides rewards based on the leaked information of the discriminator for the generator (Guo et al., 2018).

**MaliGAN**: A variant of SeqGAN that optimizes the generator with a normalized maximum likelihood objective (Che et al., 2017).

**IRL**: This inverse reinforcement learning method replaces the discriminator with a reward approximator to provide dense rewards (Shi et al., 2018).

**RAML**: A RL approach to incorporate MLE objective into RL training framework, which regards BLEU as rewards (Norouzi et al., 2016).

**DialogGAN**: An extension of SeqGAN tuned to dialogue generation task with MLE objective added to the adversarial objective (Li et al., 2017).

**DPGAN**: A variant of DialogGAN which uses a language model based discriminator and regards cross-entropy as rewards (Xu et al., 2018).

Note that MLE, SeqGAN, LeakGAN, MaliGAN and IRL are the baselines on COCO and EMNLP2017 WMT, while MLE, RAML, DialogGAN, and DPGAN on WeiboDial. The original codes are used to test the baselines.

### 4.3 Implementation Details

The implementation details of our model are shown in Table 3. For COCO / EMNLP2017, the generator is a LSTM unit (Hochreiter and Schmidhuber, 1997) with 128 cells, and the discriminator is implemented based on (Yu et al., 2017). For WeiboDial, the generator is an encoder-decoder structure with attention mechanism, where both the encoder and the decoder consist of a two-layer GRU (Cho et al., 2014) with 128 cells. The discriminator is implemented based on (Tao et al., 2018). The language model used in the constrained sampling of ARAML is implemented in the same setting as the generators, and is pre-trained on the training set of each dataset. The codes and the datasets are available at `https://github.com/kepei1106/ARAML`.

| Dataset | COCO / EMNLP2017 | WeiboDial |
|---|---|---|
| Generator | LSTM | GRU |
| Discriminator | GRU & CNN | GRU & MLP |
| Temperature | 0.85 (COCO)<br>0.9 (EMNLP2017) | 0.95 |
| Sampling size | 5 | 5 |
| Dimension of word embedding | 128 | 100 |
| Batch size | 100 | 100 |
| Pretraining epochs (G/D/LM) | 50 / 15 / 50 | 50 / 10 / 30 |
| Optimizer | Adam | Adam |
| Learning rate(G/D) | 0.001 / 0.0001 | 0.001 / 0.0001 |

Table 3: Implementation details of ARAML. G/D/LM indicates the generator / discriminator / language model used in constrained sampling, respectively.

As for the details of the baselines, the generators of all the baselines except LeakGAN are the same as ours. Note that the generator of LeakGAN consists of a hierarchical LSTM unit, thus we followed the implementation in the original paper. In terms of the differences, the discriminators of GAN baselines are implemented based on the original papers. Other hyper-parameters of baselines including batch size, learning rate, and pre-training epochs, were set based on the original codes, because the convergence of baselines is sensitive to these hyper-parameters.

### 4.4 Language Generation on COCO and EMNLP2017 WMT

We adopted forward/reverse perplexity (Zhao et al., 2018) and Self-BLEU (Zhu et al., 2018) to evaluate the quality of generated texts. Forward perplexity (PPL-F) indicates the perplexity on the generated data provided by a language model trained on real data to measure the fluency of generated samples. Reverse perplexity (PPL-R) switches the roles of generated data and real data

| Model | COCO | | | EMNLP2017 WMT | | |
|---|---|---|---|---|---|---|
| | PPL-F | PPL-R | S-BLEU-2/3/4 | PPL-F | PPL-R | S-BLEU-2/3/4 |
| MLE | 18.83 ± 0.51 | 38.81 ± 0.97 | 0.790 ± 0.006 / 0.598 ± 0.009 / 0.419 ± 0.010 | 55.64 ± 1.03 | 192.33 ± 9.51 | 0.771 ± 0.005 / 0.505 ± 0.009 / 0.304 ± 0.008 |
| SeqGAN | 33.07 ± 1.98 | 49.24 ± 2.36 | 0.814 ± 0.005 / 0.619 ± 0.008 / 0.430 ± 0.007 | 67.60 ± 3.48 | 276.82 ± 5.12 | 0.786 ± 0.019 / 0.500 ± 0.029 / 0.276 ± 0.023 |
| LeakGAN | **11.43 ± 2.74** | 87.54 ± 6.42 | 0.877 ± 0.032 / 0.762 ± 0.045 / 0.645 ± 0.049 | **17.92 ± 1.77** | 491.70 ± 18.29 | 0.890 ± 0.013 / 0.751 ± 0.011 / 0.604 ± 0.009 |
| MaliGAN | 47.16 ± 2.94 | 54.40 ± 1.29 | 0.786 ± 0.005 / 0.572 ± 0.008 / 0.370 ± 0.007 | 126.84 ± 11.18 | 288.20 ± 16.48 | 0.780 ± 0.019 / 0.494 ± 0.032 / 0.265 ± 0.028 |
| IRL | 41.86 ± 11.82 | 84.23 ± 7.02 | 0.857 ± 0.014 / 0.687 ± 0.031 / 0.499 ± 0.062 | 285.20 ± 36.47 | 1124.57 ± 109.80 | 0.890 ± 0.008 / 0.656 ± 0.052 / 0.406 ± 0.077 |
| ARAML | 26.97 ± 0.55 | **35.79 ± 0.49** | **0.777 ± 0.005 / 0.560 ± 0.006/ 0.366 ± 0.008** | 77.90 ± 0.70 | **188.41 ± 3.18** | **0.745 ± 0.002 / 0.455 ± 0.006 / 0.257 ± 0.006** |

Table 4: Automatic evaluation on COCO and EMNLP2017 WMT. Each metric is presented with mean and standard deviation.

to reflect the discrepancy between the generated distribution and the data distribution. Self-BLEU (S-BLEU) regards each sentence in the generated collection as hypothesis and the others as reference to obtain BLEU scores, which evaluates the diversity of generated results.

Results are shown in Table 4. LeakGAN performs best on forward perplexity because it can generate more fluent samples. As for reverse perplexity, our model ARAML beats other baselines, showing that our model can fit the data distribution better. Other GANs, particularly LeakGAN, obtain high reverse perplexity due to mode collapse (Shi et al., 2018), thus they only capture limited fluent expressions, resulting in large discrepancy between the generated distribution and data distribution. ARAML also outperforms the baselines in terms of Self-BLEU, indicating that our model doesn't fall into mode collapse with the help of the MLE training objective and has the ability to generate more diverse sentences.

We also provide standard deviation of each metric in Table 4, reflecting the stability of each model's performance. Our model ARAML nearly achieves the smallest standard deviation in all the metrics, indicating that our framework outperforms policy gradient in the stability of adversarial training.

### 4.5 Dialogue Generation on WeiboDial

Dialogue evaluation is an open problem and existing works have found that automatic metrics have low correlation to human evaluation (Liu et al., 2016; Novikova et al., 2017; Chaganty et al., 2018). Thus, we resorted to manual evaluation to assess the generation quality on WeiboDial. We randomly sampled 200 posts from the test set and collected the generated results from all the models. For each pair of responses (one from ARAML and the other from a baseline, given the same input post), five annotators were hired to label which response is better (i.e. win, lose or tie) in terms of *grammaticality* (whether a response itself is grammatical and logical) and *relevance* (whether a response is appropriate and relevant to the post). The two metrics were evaluated independently.

The evaluation results are shown in Table 5. To measure the inter-annotator agreement, we calculated Fleiss' kappa (Fleiss, 1971) for each pairwise comparison where results show *moderate agreement* ($0.4 \leq \kappa \leq 0.6$). We also conducted sign test to check the significance of the differences.

As shown in Table 5, ARAML performs significantly better than other baselines in all the cases. This result indicates that the samples surrounding true responses provide stable rewards for the generator, and stable RAML training paradigm significantly enhances the performance in both metrics.

### 4.6 Further Analysis on Stability

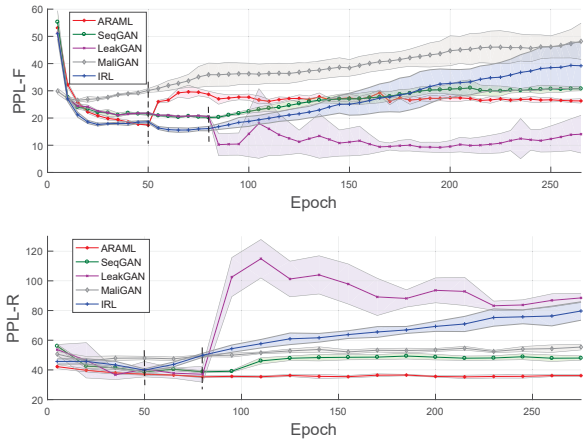


Figure 2: PPL-F/PPL-R curves of ARAML, SeqGAN, LeakGAN, MaliGAN and IRL in the training process. The shade area indicates the standard deviation at each data point. The dotted vertical lines separate pre-training and adversarial training phases (50 for ARAML, IRL and MaliGAN, 80 for SeqGAN and LeakGAN).

To verify the training stability, we conducted experiments on COCO many times and chose the best 5 trials for SeqGAN, LeakGAN, IRL, MaliGAN and ARAML, respectively. Then, we presented the forward/reverse perplexity in the train-

| Model | Grammaticality | | | $\kappa$ | Relevance | | | $\kappa$ |
|---|---|---|---|---|---|---|---|---|
| | Win (%) | Lose (%) | Tie (%) | | Win (%) | Lose (%) | Tie (%) | |
| ARAML vs. MLE | 56.5** | 36.5 | 7.0 | 0.456 | 50.5** | 37.5 | 12.0 | 0.465 |
| ARAML vs. RAML | 54.5** | 37.5 | 8.0 | 0.416 | 47.0* | 40.5 | 12.5 | 0.480 |
| ARAML vs. DialogGAN | 75.5** | 13.5 | 11.0 | 0.445 | 73.0** | 11.0 | 16.0 | 0.460 |
| ARAML vs. DPGAN | 57.5** | 36.0 | 6.5 | 0.435 | 56.5** | 30.5 | 13.0 | 0.529 |

Table 5: Human evaluation on WeiboDial. The scores represent the percentages of *Win*, *Lose* or *Tie* when our model is compared with a baseline. $\kappa$ denotes Fleiss' kappa (all are *moderate agreement*). The scores marked with * mean *p-value*$< 0.05$ and ** indicates *p-value*$< 0.01$ in sign test.

ing process in Figure 2. We can see that our model with smaller standard deviation is more stable than other GAN baselines in both metrics. Although LeakGAN reaches the best forward perplexity, its standard deviation is extremely large and it performs badly in reverse perplexity, indicating that it generates limited expressions that are grammatical yet divergent from the data distribution.

## 4.7 Ablation Study

### 4.7.1 Impact of Temperature

The temperature $\tau$ controls the search space surrounding the real data as we analyze in Section 3.3.1. To investigate its impact on the performance of our model, we fixed all the other hyperparameters and test ARAML with different temperatures on COCO.

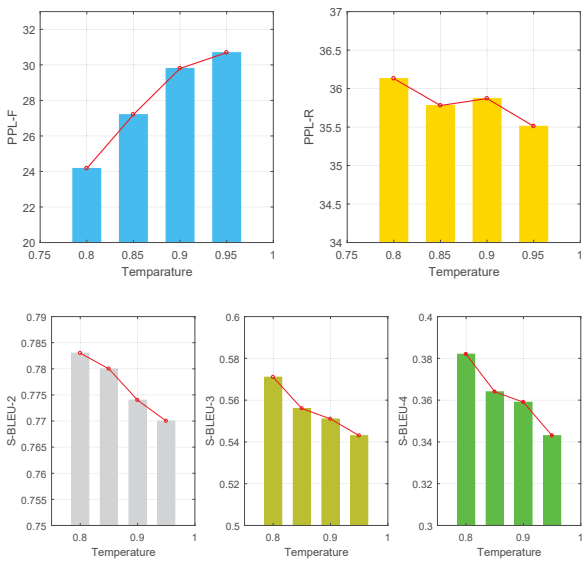


Figure 3: PPL-F, PPL-R and S-BLEU of ARAML with different temperatures $\tau \in \{0.8, 0.85, 0.9, 0.95\}$ on COCO.

The experimental results are shown in Figure 3. We can see that as the temperature becomes larger, forward perplexity increases gradually while Self-BLEU decreases. As mentioned in Section 3.3.1, large temperatures encourage our generator to explore the samples that are distant from real data distribution, thus the diversity of generated results will be improved. However, these samples distant from the data distribution are more likely to be poor in fluency, leading to worse forward perplexity. Reverse perplexity is influenced by both generation quality and diversity, so the correlation between temperature and reverse perplexity is not intuitive. We can observe that the model with $\tau = 0.95$ reaches the best reverse perplexity.

### 4.7.2 Impact of Sampling Strategy

We have mentioned two common sampling strategies in Section 3.3.1, i.e. random sampling and constrained sampling. To analyze their impact, we keep all the model structures and hyperparameters fixed and test ARAML with these two strategies on COCO.

| Model | PPL-F | PPL-R | S-BLEU-2/3/4 |
|---|---|---|---|
| ARAML-R | 37.48±0.53 | 37.44±0.56 | **0.752**/0.571/0.384 |
| ARAML-C | **26.97±0.55** | **35.79±0.49** | 0.777/**0.560/0.366** |

Table 6: PPL-F, PPL-R and S-BLEU of ARAML with random sampling (ARAML-R) and constrained sampling (ARAML-C) on COCO.

Table 6 shows the results. It's obvious that random sampling hurts the model performance except Self-BLEU-1, because it indeed allows low-quality samples available to the generator. Exploring these samples degrades the quality and diversity of generated results. Despite the worse performance on automatic metrics, random sampling doesn't affect the training stability of our framework. The standard deviation of ARAML-R is still smaller than other GAN baselines.

## 4.8 Case Study

Table 7 presents the examples generated by the models on COCO. We can find that other baselines suffer from grammatical errors (e.g. *"in front of flying her kite"* from MLE), repetitive expressions

| Model | Generated Samples |
|---|---|
| MLE | A little girl sitting on a beach in front of flying her kite at the beach.<br>A little boy standing in a room next to a desk. |
| SeqGAN | A man sitting on a bench with snow board in the background.<br>A brown gray cat is in the corner of a street. |
| LeakGAN | A person that is holding something while another kid is standing in the water.<br>A room with a television, mantle, and a chair. |
| MaliGAN | A man with a shirt on holding one large pink giant and white kite.<br>A couple and vases are outside on the bed. |
| IRL | A group of people wearing helmet sitting down on a cell phone.<br>A group of people sitting in the middle of tracks. |
| ARAML | A man is wearing a hat and holding a toothbrush as he stands on the grass of a field.<br>A boy reading a book on a sofa in a room. |

Table 7: Examples of generated sentences on COCO. Grammatical errors are in red, while blue text represents repetitive expressions and green part indicates incoherent statements.

(e.g. *"A group of people"* from IRL) and incoherent statements (e.g. *"A group of people sitting on a cell phone"* from IRL). By contrast, our model performs well in these sentences and has the ability to generate grammatical and coherent results.

| | |
|---|---|
| Post: | 你还迟到哪？你没这习惯啊<br>Are you late? You don't have this habit. |
| MLE: | 你是不是要好好照顾自己的！！！<br>Would you take care of yourself !!! |
| RAML: | 这是我的生活，我也不想干<br>This is my life. I don't want to do this. |
| DialogGAN: | 这社会的温度快到了。周末，要躺床上！<br>The temperature of the society has nearly reached. I'll lie on my bed on the weekend! |
| DPGAN: | 我在上班，我就不习惯了<br>I'm working and I'm used to doing this. |
| ARAML: | 嗯，周末就不迟到了！<br>Hum. I won't be late on the weekend! |

Table 8: Examples of generated responses on WeiboDial.

Table 8 shows the generated examples on WeiboDial. It's obvious that other baselines don't capture the topic word "late" in the post, thus generate irrelevant responses. ARAML can provide a response that is grammatical and closely relevant to the post.

## 5 Conclusion

We propose a novel adversarial training framework to deal with the instability problem of current GANs for text generation. To address the instability issue caused by policy gradient, we incorporate RAML into the adversarial training paradigm to make our generator acquire stable rewards. Experiments show that our model performs better than several state-of-the-art GAN baselines with lower training variance, yet producing better performance on three text generation tasks.

## Acknowledgments

This work was supported by the National Science Foundation of China (Grant No. 61936010/61876096) and the National Key R&D Program of China (Grant No. 2018YFC0830200). We would like to thank THUNUS NExT Joint-Lab for the support.

## References

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. An actor-critic algorithm for sequence prediction. In *Proceedings of International Conference on Learning Representations*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Massimo Caccia, Lucas Caccia, William Fedus, Hugo Larochelle, Joelle Pineau, and Laurent Charlin. 2018. Language gans falling short. *arXiv preprint arXiv: 1811.02549*.

Arun Tejasvi Chaganty, Stephen Mussmann, and Percy Liang. 2018. The price of debiasing automatic metrics in natural language evaluation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 643–653.

Tong Che, Yanran Li, Ruixiang Zhang, R Devon Hjelm, Wenjie Li, Yangqiu Song, and Yoshua Bengio. 2017. Maximum-likelihood augmented discrete generative adversarial networks. *arXiv preprint arXiv: 1702.07983*.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollar, and C. Lawrence Zitnick. 2015. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv: 1504.00325*.

Kyunghyun Cho, Bart Van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder

approaches. In *Proceedings of Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111.

William Fedus, Ian J. Goodfellow, and Andrew M. Dai. 2018. Maskgan: Better text generation via filling in the _____. In *Proceedings of International Conference on Learning Representations*.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv: 1308.0850*.

Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. 2018. Long text generation via adversarial training with leaked information. In *Proceedings of AAAI conference on Artificial Intelligence*, pages 5141–5148.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Pei Ke, Jian Guan, Minlie Huang, and Xiaoyan Zhu. 2018. Generating informative responses with controlled sentence function. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 1499–1508.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119.

Jiwei Li, Will Monroe, Tianlin Shi, Sebastien Jean, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2157–2169.

Kevin Lin, Dianqi Li, Xiaodong He, Zhengyou Zhang, and Ming-Ting Sun. 2017. Adversarial ranking for language generation. In *Advances in Neural Information Processing Systems*, pages 3155–3165.

Chia Wei Liu, Ryan Lowe, Iulian V. Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2122–2132.

Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, Zhen Wang, and Stephen Paul Smolley. 2017. Least squares generative adversarial networks. In *International Conference on Computer Vision*, pages 2813–2821.

Ning Miao, Hao Zhou, Lili Mou, Rui Yan, and Lei Li. 2019. Cgmh: Constrained sentence generation by metropolis-hastings sampling. In *Proceedings of AAAI conference on Artificial Intelligence*.

Tomas Mikolov, Martin Karafiat, Lukas Burget, Jan Honza Cernock, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of the 11st Annual Conference of the International Speech Communication Association*, pages 1045–1048.

Lili Mou, Yiping Song, Rui Yan, Ge Li, Lu Zhang, and Zhi Jin. 2016. Sequence to backward and forward sequences: A content-introducing approach to generative short-text conversation. In *Proceedings of 26th International Conference on Computational Linguistics*, pages 3349–3358.

Mohammad Norouzi, Samy Bengio, Zhifeng Chen, Navdeep Jaitly, Mike Schuster, Yonghui Wu, and Dale Schuurmans. 2016. Reward augmented maximum likelihood for neural structured prediction. In *Advances in Neural Information Processing Systems*, pages 1723–1731.

Jekaterina Novikova, Ondrej Dusek, Amanda Cercas Curry, and Verena Rieser. 2017. Why we need new evaluation metrics for NLG. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2241–2252.

Qiao Qian, Minlie Huang, Haizhou Zhao, Jingfang Xu, and Xiaoyan Zhu. 2018. Assigning personality/profile to a chatting machine for coherent conversation generation. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, pages 4279–4285.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *Proceedings of International Conference on Learning Representations*.

Stanislau Semeniuta1, Aliaksei Severyn, and Sylvain Gelly. 2018. On accurate evaluation of gans for language generation. *arXiv preprint arXiv: 1806.04936*.

Zhan Shi, Xinchi Chen, Xipeng Qiu, and Xuanjing Huang. 2018. Toward diverse text generation with inverse reinforcement learning. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, pages 4361–4367.

Jinyue Su, Jiacheng Xu, Xipeng Qiu, and Xuanjing Huang. 2018. Incorporating discriminator in sentence generation: A gibbs sampling method. In *Proceedings of AAAI conference on Artificial Intelligence*, pages 5496–5503.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Chongyang Tao, Lili Mou, Dongyan Zhao, and Rui Yan. 2018. Ruber: An unsupervised method for automatic evaluation of open-domain dialog systems. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 722–729.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. In *International Conference on Machine Learning Deep Learning Workshop*.

Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2017. Topic aware neural response generation. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 3351–3357.

Jingjing Xu, Xuancheng Ren, Junyang Lin, and Xu Sun. 2018. Diversity-promoting gan: A cross-entropy based generative adversarial network for diversified text generation. In *Conference on Empirical Methods in Natural Language Processing*, page 3940–3949.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of AAAI conference on Artificial Intelligence*, pages 2852–2858.

Junbo Jake Zhao, Yoon Kim, Kelly Zhang, Alexander M. Rush, and Yann LeCun. 2018. Adversarially regularized autoencoders. In *Proceedings of the 35th International Conference on Machine Learning*, pages 5897–5906.

Hao Zhou, Minlie Huang, Tianyang Zhang, Xiaoyan Zhu, and Bing Liu. 2018a. Emotional chatting machine: Emotional conversation generation with internal and external memory. In *Proceedings of AAAI conference on Artificial Intelligence*.

Hao Zhou, Tom Young, Minlie Huang, Haizhou Zhao, Jingfang Xu, and Xiaoyan Zhu. 2018b. Commonsense knowledge aware conversation generation with graph attention. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, pages 4623–4629.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. In *Proceedings of the 41st International ACM SIGIR Conference on Research Development in Information Retrieval*, pages 1097–1100.

# FlowSeq: Non-Autoregressive Conditional Sequence Generation with Generative Flow

**Xuezhe Ma**[*,1] **Chunting Zhou**[*,1] **Xian Li**[2] **Graham Neubig**[1] **Eduard Hovy**[1]
[1]Language Technologies Institute, Carnegie Mellon University
[2]Facebook AI
{xuezhem, chuntinz, gneubig, ehovy}@cs.cmu.edu xianl@fb.com

## Abstract

Most sequence-to-sequence (seq2seq) models are *autoregressive*; they generate each token by conditioning on previously generated tokens. In contrast, non-autoregressive seq2seq models generate all tokens in one pass, which leads to increased efficiency through parallel processing on hardware such as GPUs. However, directly modeling the joint distribution of all tokens simultaneously is challenging, and even with increasingly complex model structures accuracy lags significantly behind autoregressive models. In this paper, we propose a simple, efficient, and effective model for non-autoregressive sequence generation using latent variable models. Specifically, we turn to generative flow, an elegant technique to model complex distributions using neural networks, and design several layers of flow tailored for modeling the conditional density of sequential latent variables. We evaluate this model on three neural machine translation (NMT) benchmark datasets, achieving comparable performance with state-of-the-art non-autoregressive NMT models and almost constant decoding time w.r.t the sequence length.[1]

## 1 Introduction

Neural sequence-to-sequence (seq2seq) models (Bahdanau et al., 2015; Rush et al., 2015; Vinyals et al., 2015; Vaswani et al., 2017) generate an output sequence $\mathbf{y} = \{y_1, \ldots, y_T\}$ given an input sequence $\mathbf{x} = \{x_1, \ldots, x_{T'}\}$ using conditional probabilities $P_\theta(\mathbf{y}|\mathbf{x})$ predicted by neural networks (parameterized by $\theta$).

Most seq2seq models are *autoregressive*, meaning that they factorize the joint probability of the output sequence given the input sequence $P_\theta(\mathbf{y}|\mathbf{x})$ into the product of probabilities over the next token in the sequence given the input sequence and previously generated tokens:

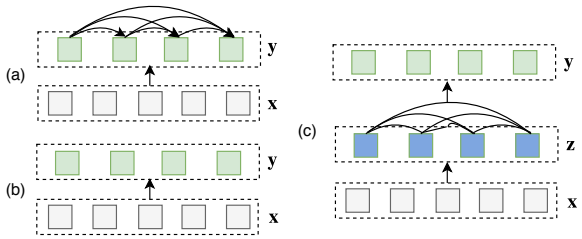


Figure 1: (a) Autoregressive (b) non-autoregressive and (c) our proposed sequence generation models. $\mathbf{x}$ is the source, $\mathbf{y}$ is the target, and $\mathbf{z}$ are latent variables.

$$P_\theta(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^{T} P_\theta(y_t|y_{<t}, \mathbf{x}). \quad (1)$$

Each factor, $P_\theta(y_t|y_{<t}, \mathbf{x})$, can be implemented by function approximators such as RNNs (Bahdanau et al., 2015) and Transformers (Vaswani et al., 2017). This factorization takes the complicated problem of *joint estimation* over an exponentially large output space of outputs $\mathbf{y}$, and turns it into a *sequence of tractable multi-class classification problems* predicting $y_t$ given the previous words, allowing for simple maximum log-likelihood training. However, this assumption of left-to-right factorization may be sub-optimal from a modeling perspective (Gu et al., 2019; Stern et al., 2019), and generation of outputs must be done through a linear left-to-right pass through the output tokens using beam search, which is not easily parallelizable on hardware such as GPUs.

Recently, there has been work on non-autoregressive sequence generation for neural machine translation (NMT; Gu et al. (2018); Lee et al. (2018); Ghazvininejad et al. (2019)) and language modeling (Ziegler and Rush, 2019). Non-autoregressive models attempt to model the joint distribution $P_\theta(\mathbf{y}|\mathbf{x})$ directly, decoupling the dependencies of decoding history during generation.

[*] Equal contribution, in alphabetical order.
[1] https://github.com/XuezheMax/flowseq



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4282–4292, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

A naïve solution is to assume that each token of the target sequence is independent given the input:

$$P_\theta(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^{T} P_\theta(y_t|\mathbf{x}). \quad (2)$$

Unfortunately, the performance of this simple model falls far behind autoregressive models, as seq2seq tasks usually do have strong conditional dependencies between output variables (Gu et al., 2018). This problem can be mitigated by introducing a latent variable $\mathbf{z}$ to model these conditional dependencies:

$$P_\theta(\mathbf{y}|\mathbf{x}) = \int_{\mathbf{z}} P_\theta(\mathbf{y}|\mathbf{z},\mathbf{x}) p_\theta(\mathbf{z}|\mathbf{x}) d\mathbf{z}, \quad (3)$$

where $p_\theta(\mathbf{z}|\mathbf{x})$ is the prior distribution over latent $\mathbf{z}$ and $P_\theta(\mathbf{y}|\mathbf{z},\mathbf{x})$ is the "generative" distribution (a.k.a decoder). Non-autoregressive generation can be achieved by the following independence assumption in the decoding process:

$$P_\theta(\mathbf{y}|\mathbf{z},\mathbf{x}) = \prod_{t=1}^{T} P_\theta(y_t|\mathbf{z},\mathbf{x}). \quad (4)$$

Gu et al. (2018) proposed a $\mathbf{z}$ representing fertility scores specifying the number of output words each input word generates, significantly improving the performance over Eq. (2). But the performance still falls behind state-of-the-art autoregressive models due to the limited expressiveness of fertility to model the interdependence between words in $\mathbf{y}$.

In this paper, we propose a simple, effective, and efficient model, **FlowSeq**, which models expressive prior distribution $p_\theta(\mathbf{z}|\mathbf{x})$ using a powerful mathematical framework called generative flow (Rezende and Mohamed, 2015). This framework can elegantly model complex distributions, and has obtained remarkable success in modeling continuous data such as images and speech through efficient density estimation and sampling (Kingma and Dhariwal, 2018; Prenger et al., 2019; Ma and Hovy, 2019). Based on this, we posit that generative flow also has potential to introduce more meaningful latent variables $\mathbf{z}$ in the non-autoregressive generation in Eq. (3).

FlowSeq is a *flow-based sequence-to-sequence* model, which is (to our knowledge) the first non-autoregressive seq2seq model utilizing generative flows. It allows for efficient parallel decoding while modeling the joint distribution of the output sequence. Experimentally, on three benchmark datasets for machine translation – WMT2014, WMT2016 and IWSLT-2014, FlowSeq achieves comparable performance with state-of-the-art non-autoregressive models, and almost constant decoding time w.r.t. the sequence length compared to a typical left-to-right Transformer model, which is super-linear.

## 2 Background

As noted above, incorporating expressive latent variables $\mathbf{z}$ is essential to decouple the dependencies between tokens in the target sequence in non-autoregressive models. However, in order to model all of the complexities of sequence generation to the point that we can read off all of the words in the output in an independent fashion (as in Eq. (4)), the prior distribution $p_\theta(\mathbf{z}|\mathbf{x})$ will necessarily be quite complex. In this section, we describe generative flows (Rezende and Mohamed, 2015), an effective method for arbitrary modeling of complicated distributions, before describing how we apply them to sequence-to-sequence generation in §3.

### 2.1 Flow-based Generative Models

Put simply, flow-based generative models work by transforming a simple distribution (e.g. a simple Gaussian) into a complex one (e.g. the complex prior distribution over $\mathbf{z}$ that we want to model) through a chain of invertible transformations.

Formally, a set of latent variables $\boldsymbol{\upsilon} \in \Upsilon$ are introduced with a simple prior distribution $p_\Upsilon(\upsilon)$. We then define a bijection function $f : \mathcal{Z} \to \Upsilon$ (with $g = f^{-1}$), whereby we can define a generative process over variables $\mathbf{z}$:

$$\begin{aligned} \boldsymbol{\upsilon} &\sim p_\Upsilon(\boldsymbol{\upsilon}) \\ \mathbf{z} &= g_\theta(\boldsymbol{\upsilon}). \end{aligned} \quad (5)$$

An important insight behind flow-based models is that given this bijection function, the change of variable formula defines the model distribution on $\mathbf{z} \in \mathcal{Z}$ by:

$$p_\theta(\mathbf{z}) = p_\Upsilon(f_\theta(\mathbf{z})) \left| \det\left(\frac{\partial f_\theta(\mathbf{z})}{\partial \mathbf{z}}\right) \right|. \quad (6)$$

Here $\frac{\partial f_\theta(\mathbf{z})}{\partial \mathbf{z}}$ is the Jacobian matrix of $f_\theta$ at $\mathbf{z}$.

Eq. (6) provides a way to calculate the (complex) density of $\mathbf{z}$ by calculating the (simple) density of $\upsilon$ and the Jacobian of the transformation from $\mathbf{z}$ to $\upsilon$. For efficiency purposes, flow-based models generally use certain types of transformations $f_\theta$ where both the inverse functions

$g_\theta$ and the Jacobian determinants are tractable to compute. A stacked sequence of such invertible transformations is also called a (normalizing) *flow* (Rezende and Mohamed, 2015):

$$\mathbf{z} \underset{g_1}{\overset{f_1}{\longleftrightarrow}} H_1 \underset{g_2}{\overset{f_2}{\longleftrightarrow}} H2 \underset{g_3}{\overset{f_3}{\longleftrightarrow}} \cdots \underset{g_K}{\overset{f_K}{\longleftrightarrow}} \boldsymbol{\upsilon},$$

where $f = f_1 \circ f_2 \circ \cdots \circ f_K$ is a flow of $K$ transformations (omitting $\theta$s for brevity).

### 2.2 Variational Inference and Training

In the context of maximal likelihood estimation (MLE), we wish to minimize the negative log-likelihood of the parameters:

$$\min_{\theta \in \Theta} \frac{1}{N} \sum_{i=1}^{N} -\log P_\theta(\mathbf{y}^i|\mathbf{x}^i), \tag{7}$$

where $D = \{(\mathbf{x}^i, \mathbf{y}^i)\}_{i=1}^N$ is the set of training data. However, the likelihood $P_\theta(\mathbf{y}|\mathbf{x})$ after marginalizing out latent variables $\mathbf{z}$ (LHS in Eq. (3)) is intractable to compute or differentiate directly. Variational inference (Wainwright et al., 2008) provides a solution by introducing a parametric *inference model* $q_\phi(\mathbf{z}|\mathbf{y}, \mathbf{x})$ (a.k.a posterior) which is then used to approximate this integral by sampling individual examples of $\mathbf{z}$. These models then optimize the *evidence lower bound* (ELBO), which considers both the "reconstruction error" $\log P_\theta(\mathbf{y}|\mathbf{z}, \mathbf{x})$ and KL-divergence between the posterior and the prior:

$$\begin{aligned}\log P_\theta(\mathbf{y}|\mathbf{x}) \geq \mathrm{E}_{q_\phi(\mathbf{z}|\mathbf{y},\mathbf{x})}[\log P_\theta(\mathbf{y}|\mathbf{z}, \mathbf{x})] \\ - \mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{y}, \mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})).\end{aligned} \tag{8}$$

Both inference model $\phi$ and decoder $\theta$ parameters are optimized according to this objective.

## 3 FlowSeq

We first overview FlowSeq's architecture (shown in Figure 2) and training process here before detailing each component in following sections. Similarly to classic seq2seq models, at both training and test time FlowSeq first reads the whole input sequence $\mathbf{x}$ and calculates a vector for each word in the sequence, the source encoding.

At training time, FlowSeq's parameters are learned using a variational training paradigm overviewed in §2.2. First, we draw samples of latent codes $\mathbf{z}$ from the current posterior $q_\phi(\mathbf{z}|\mathbf{y}, \mathbf{x})$. Next, we feed $\mathbf{z}$ together with source encodings into the decoder network and the prior flow to compute the probabilities of $P_\theta(\mathbf{y}|\mathbf{z}, \mathbf{x})$ and $p_\theta(\mathbf{z}|\mathbf{x})$ for optimizing the ELBO (Eq. (8)).

At test time, generation is performed by first sampling a latent code $\mathbf{z}$ from the prior flow by executing the generative process defined in Eq. (5). In this step, the source encodings produced from the encoder are used as conditional inputs. Then the decoder receives both the sampled latent code $\mathbf{z}$ and the source encoder outputs to generate the target sequence $\mathbf{y}$ from $P_\theta(\mathbf{y}|\mathbf{z}, \mathbf{x})$.

### 3.1 Source Encoder

The source encoder encodes the source sequences into hidden representations, which are used in computing attention when generating latent variables in the posterior network and prior network as well as the cross-attention with decoder. Any standard neural sequence model can be used as its encoder, including RNNs (Bahdanau et al., 2015) or Transformers (Vaswani et al., 2017).

### 3.2 Posterior

**Generation of Latent Variables.** The latent variables $\mathbf{z}$ are represented as a sequence of continuous random vectors $\mathbf{z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_T\}$ with the same length as the target sequence $\mathbf{y}$. Each $\mathbf{z}_t$ is a $d_\mathrm{z}$-dimensional vector, where $d_\mathrm{z}$ is the dimension of the latent space. The posterior distribution $q_\phi(\mathbf{z}|\mathbf{y}, \mathbf{x})$ models each $\mathbf{z}_t$ as a diagonal Gaussian with learned mean and variance:

$$q_\phi(\mathbf{z}|\mathbf{y}, \mathbf{x}) = \prod_{t=1}^{T} \mathcal{N}(\mathbf{z}_t|\mu_t(\mathbf{x}, \mathbf{y}), \sigma_t^2(\mathbf{x}, \mathbf{y})) \tag{9}$$

where $\mu_t(\cdot)$ and $\sigma_t(\cdot)$ are neural networks such as RNNs or Transformers.

**Zero initialization.** While we perform standard random initialization for most layers of the network, we initialize the last linear transforms that generate the $\mu$ and $\log \sigma^2$ values with zeros. This ensures that the posterior distribution as a simple normal distribution, which we found helps train very deep generative flows more stably.

**Token Dropout.** The motivation of introducing the latent variable $\mathbf{z}$ into the model is to model the uncertainty in the generative process. Thus, it is preferable that $\mathbf{z}$ capture contextual interdependence between tokens in $\mathbf{y}$. However, there is an obvious local optimum where the posterior network generates a latent vector $\mathbf{z}_t$ that only encodes the information about the corresponding target token $y_t$, and the decoder simply generates the

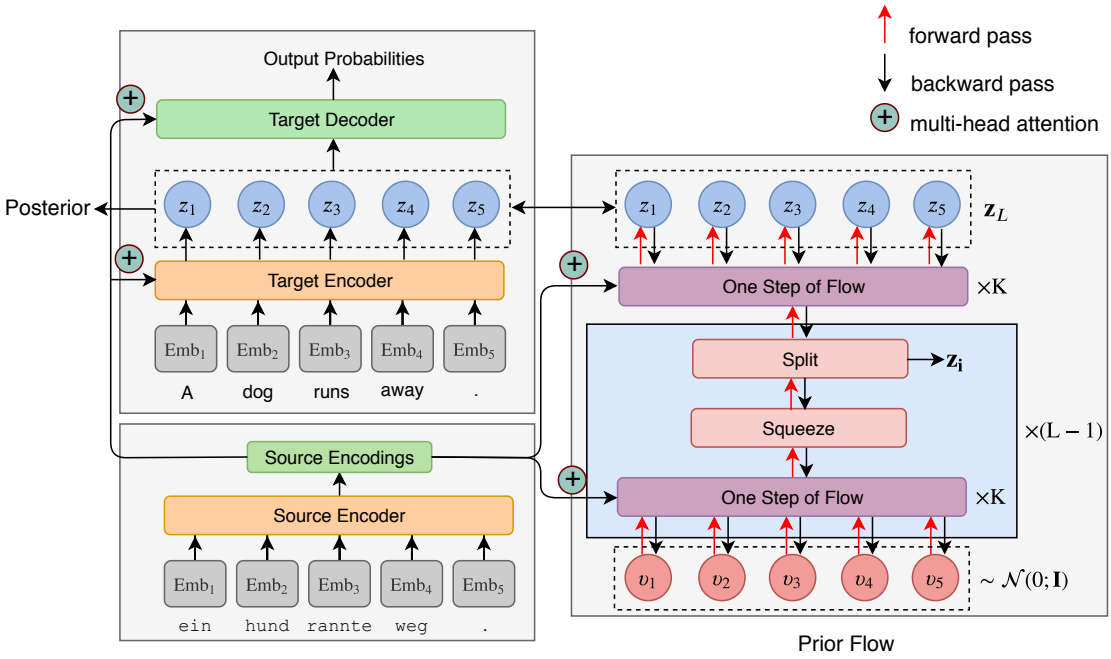


Figure 2: Neural architecture of FlowSeq, including the encoder, the decoder and the posterior networks, together with the multi-scale architecture of the prior flow. The architecture of each flow step is in Figure 3.

"correct" token at each step $t$ with $\mathbf{z}_t$ as input. In this case, FlowSeq reduces to the baseline model in Eq. (2). To escape this undesired local optimum, we apply token-level dropout to randomly drop an entire token when calculating the posterior, to ensure the model also has to learn how to use contextual information. This technique is similar to the "masked language model" in previous studies (Melamud et al., 2016; Devlin et al., 2018; Ma et al., 2018).

### 3.3 Decoder

As the decoder, we take the latent sequence $\mathbf{z}$ as input, run it through several layers of a neural sequence model such as a Transformer, then directly predict the output tokens in $\mathbf{y}$ individually and independently. Notably, unlike standard seq2seq decoders, we do not perform causal masking to prevent attending to future tokens, making the model fully non-autoregressive.

### 3.4 Flow Architecture for Prior

The flow architecture is based on Glow (Kingma and Dhariwal, 2018). It consists of a series of steps of flow, combined in a multi-scale architecture (see Figure 2.) Each step of flow consists three types of elementary flows – actnorm, invertible multi-head linear, and coupling. Note that all three functions are invertible and conducive to calculation of log determinants (details in Appendix A).

**Actnorm.** The activation normalization layer (actnorm; Kingma and Dhariwal (2018)) is an alternative for batch normalization (Ioffe and Szegedy, 2015), that has mainly been used in the context of image data to alleviate problems in model training. Actnorm performs an affine transformation of the activations using a scale and bias parameter per feature for sequences:

$$\mathbf{z}'_t = \mathbf{s} \odot \mathbf{z}_t + \mathbf{b}. \tag{10}$$

Both $\mathbf{z}$ and $\mathbf{z}'$ are tensors of shape $[T \times d_z]$ with time dimension $t$ and feature dimension $d_z$. The parameters are initialized such that over each feature $\mathbf{z}'_t$ has zero mean and unit variance given an initial mini-batch of data.

**Invertible Multi-head Linear Layers.** To incorporate general permutations of variables along the feature dimension to ensure that each dimension can affect every other ones after a sufficient number of steps of flow, Kingma and Dhariwal (2018) proposed a trainable invertible $1 \times 1$ convolution layer for 2D images. It is straightforward to apply similar transformations to sequential data:

$$\mathbf{z}'_t = \mathbf{z}_t \mathbf{W}, \tag{11}$$

where $\mathbf{W}$ is the weight matrix of shape $[d_z \times d_z]$. The log-determinant of this transformation is:

$$\log \left| \det \left( \frac{\partial \text{linear}(\mathbf{z}; \mathbf{W})}{\partial \mathbf{z}} \right) \right| = T \cdot \log |\det(\mathbf{W})|$$

The cost of computing $\det(\mathbf{W})$ is $O(d_z^3)$.

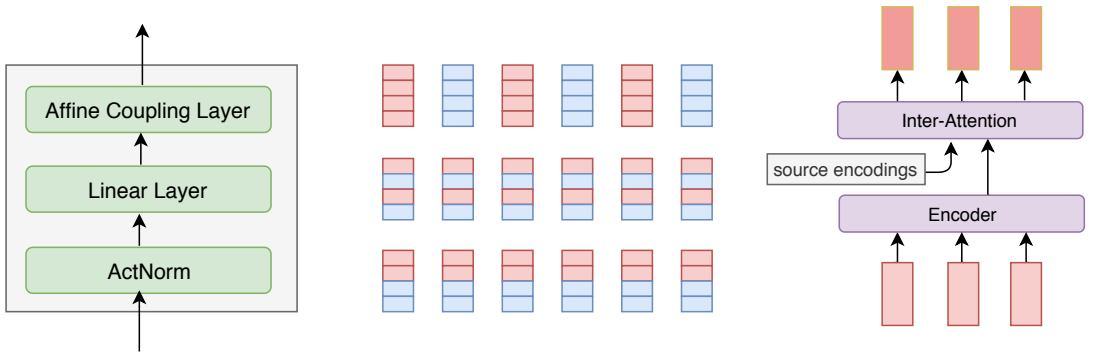


(a) One step of flow. (b) Coupling layer splits. (c) NN function on the split of the coupling layer.

Figure 3: (a) The architecture of one step of our flow. (b) The visualization of three split pattern for coupling layers, where the red color denotes $\mathbf{z}_a$ and the blue color denotes $zv_b$. (c) The attention-based architecture of the NN function in coupling layers.

Unfortunately, $d_z$ in Seq2Seq generation is commonly large, e.g. 512, significantly slowing down the model for computing $\det(\mathbf{W})$. To apply this to sequence generation, we propose a multi-head invertible linear layer, which first splits each $d_z$-dimensional feature vector into $h$ heads with dimension $d_h = d_z/h$. Then the linear transformation in (11) is applied to each head, with $d_h \times d_h$ weight matrix $\mathbf{W}$, significantly reducing the dimension. For splitting of heads, one step of flow contains one linear layer with either row-major or column-major splitting format, and these steps with different linear layers are composed in an alternating pattern.

**Affine Coupling Layers.** To model interdependence across time steps, we use affine coupling layers (Dinh et al., 2016):

$$\begin{aligned} \mathbf{z}_a, \mathbf{z}_b &= \text{split}(\mathbf{z}) \\ \mathbf{z}'_a &= \mathbf{z}_a \\ \mathbf{z}'_b &= \text{s}(\mathbf{z}_a, \mathbf{x}) \odot \mathbf{z}_b + \text{b}(\mathbf{z}_a, \mathbf{x}) \\ \mathbf{z}' &= \text{concat}(\mathbf{z}'_a, \mathbf{z}'_b), \end{aligned}$$

where $\text{s}(\mathbf{z}_a, \mathbf{x})$ and $\text{b}(\mathbf{z}_a, \mathbf{x})$ are outputs of two neural networks with $\mathbf{z}_a$ and $\mathbf{x}$ as input. These are shown in Figure 3 (c). In experiments, we implement $\text{s}(\cdot)$ and $\text{b}(\cdot)$ with one Transformer decoder layer (Vaswani et al., 2017): multi-head self-attention over $\mathbf{z}_a$, followed by multi-head inter-attention over $\mathbf{x}$, followed by a position-wise feed-forward network. The input $\mathbf{z}_a$ is fed into this layer in one pass, without causal masking.

As in Dinh et al. (2016), the $\text{split}()$ function splits $\mathbf{z}$ the input tensor into two halves, while the $\text{concat}$ operation performs the corresponding reverse concatenation operation. In our architecture, three types of split functions are used, based on the split dimension and pattern. Figure 3 (b) illustrates the three splitting types. The first type of split groups $\mathbf{z}$ along the time dimension on alternate indices. In this case, FlowSeq mainly models the interactions between time-steps. The second and third types of splits perform on the feature dimension, with continuous and alternate patterns, respectively. For each type of split, we alternate $\mathbf{z}_a$ and $\mathbf{z}_b$ to increase the flexibility of the split function. Different types of affine coupling layers alternate in the flow, similar to the linear layers.

**Multi-scale Architecture.** We follow Dinh et al. (2016) in implementing a multi-scale architecture using the squeezing operation on the feature dimension, which has been demonstrated helpful for training deep flows. Formally, each scale is a combination of several steps of the flow (see Figure 3 (a)). After each scale, the model drops half of the dimensions with the third type of split in Figure 3 (b) to reduce computational and memory cost, outputting the tensor with shape $[T \times \frac{d}{2}]$. Then the squeezing operation transforms the $T \times \frac{d}{2}$ tensor into an $\frac{T}{2} \times d$ one as the input of the next scale. We pad each sentence with EOS tokens to ensure $T$ is divisible by 2. The right component of Figure 2 illustrates the multi-scale architecture.

### 3.5 Predicting Target Sequence Length

In autoregressive seq2seq models, it is natural to determine the length of the sequence dynamically by simply predicting a special EOS token. However, for FlowSeq to predict the entire sequence in parallel, it needs to know its length in advance to generate the latent sequence $\mathbf{z}$. Instead of predicting the absolute length of the target sequence, we predict the length difference between source

and target sequences using a classifier with a range of $[-20, 20]$. Numbers in this range are predicted by max-pooling the source encodings into a single vector,[2] running this through a linear layer, and taking a softmax. This classifier is learned jointly with the rest of the model.

### 3.6 Decoding Process

At inference time, the model needs to identify the sequence with the highest conditional probability by marginalizing over all possible latent variables (see Eq. (3)), which is intractable in practice. We propose three approximating decoding algorithms to reduce the search space.

**Argmax Decoding.** Following Gu et al. (2018), one simple and effective method is to select the best sequence by choosing the highest-probability latent sequence $\mathbf{z}$:

$$\begin{aligned} \mathbf{z}^* &= \operatorname*{argmax}_{\mathbf{z}\in\mathcal{Z}} p_\theta(\mathbf{z}|\mathbf{x}) \\ \mathbf{y}^* &= \operatorname*{argmax}_{\mathbf{y}} P_\theta(\mathbf{y}|\mathbf{z}^*, \mathbf{x}) \end{aligned}$$

where identifying $\mathbf{y}^*$ only requires independently maximizing the local probability for each output position (see Eq. 4).

**Noisy Parallel Decoding (NPD).** A more accurate approximation of decoding, proposed in Gu et al. (2018), is to draw samples from the latent space and compute the best output for each latent sequence. Then, a pre-trained autoregressive model is adopted to rank these sequences. In FlowSeq, different candidates can be generated by sampling different target lengths or different samples from the prior, and both of the strategies can be batched via masks during decoding. In our experiments, we first select the top $l$ length candidates from the length predictor in §3.5. Then, for each length candidate we use $r$ random samples from the prior network to generate output sequences, yielding a total of $l \times r$ candidates.

**Importance Weighted Decoding (IWD)** The third approximating method is based on the *lower bound of importance weighted estimation* (Burda et al., 2015). Similarly to NPD, IWD first draws samples from the latent space and computes the best output for each latent sequence. Then, IWD ranks these candidate sequences with $K$ importance samples:

$$\begin{aligned} \mathbf{z}_i &\sim p_\theta(\mathbf{z}|\mathbf{x}), \forall i = 1, \ldots, N \\ \hat{\mathbf{y}}_i &= \operatorname*{argmax}_{\mathbf{y}} P_\theta(\mathbf{y}|\mathbf{z}_i, \mathbf{x}) \\ \mathbf{z}_i^{(k)} &\sim q_\phi(\mathbf{z}|\hat{\mathbf{y}}_i, \mathbf{x}), \forall k = 1, \ldots, K \\ P(\hat{\mathbf{y}}_i|\mathbf{x}) &\approx \frac{1}{K}\sum_{k=1}^{K} \frac{P_\theta(\hat{\mathbf{y}}_i|\mathbf{z}_i^{(k)}, \mathbf{x}) p_\theta(\mathbf{z}_i^{(k)}|\mathbf{x})}{q_\phi(\mathbf{z}_i^{(k)}|\hat{\mathbf{y}}_i, \mathbf{x})} \end{aligned}$$

IWD does not rely on a separate pre-trained model, though it significantly slows down the decoding speed. The detailed comparison of these three decoding methods is provided in §4.2.

### 3.7 Discussion

Different from the architecture proposed in Ziegler and Rush (2019), the architecture of FlowSeq is not using any autoregressive flow (Kingma et al., 2016; Papamakarios et al., 2017), yielding a truly non-autoregressive model with efficient generation. Note that the FlowSeq remains non-autoregressive even if we use an RNN in the architecture because RNN is only used to encode a complete sequence of codes and all the input tokens can be fed into the RNN in parallel. This makes it possible to use highly-optimized implementations of RNNs such as those provided by cuDNN.[3] Thus while RNNs do experience some drop in speed, it is less extreme than that experienced when using autoregressive models.

## 4 Experiments

### 4.1 Experimental Setups

**Translation Datasets** We evaluate FlowSeq on three machine translation benchmark datasets: WMT2014 DE-EN (around 4.5M sentence pairs), WMT2016 RO-EN (around 610K sentence pairs) and a smaller dataset IWSLT2014 DE-EN (around 150K sentence pairs). We use scripts from fairseq (Ott et al., 2019) to preprocess WMT2014 and IWSLT2014, where the preprocessing steps follow Vaswani et al. (2017) for WMT2014. We use the data provided in Lee et al. (2018) for WMT2016. For both WMT datasets, the source and target languages share the same set of BPE embeddings while for IWSLT2014 we use separate embeddings. During training, we filter out sentences longer than 80 for WMT dataset and 60 for IWSLT, respectively.

[2] We experimented with other methods such as mean-pooling or taking the last hidden state and found no major difference in our experiments

[3] https://devblogs.nvidia.com/optimizing-recurrent-neural-networks-cudnn-5/

| Models | WMT2014 EN-DE | WMT2014 DE-EN | WMT2016 EN-RO | WMT2016 RO-EN | IWSLT2014 DE-EN |
|---|---|---|---|---|---|
| Raw Data | | | | | |
| CMLM-base | 10.88 | – | 20.24 | – | – |
| LV NAR | 11.80 | – | – | – | – |
| FlowSeq-base | 18.55 | 23.36 | 29.26 | 30.16 | **24.75** |
| FlowSeq-large | **20.85** | **25.40** | **29.86** | **30.69** | – |
| Knowledge Distillation | | | | | |
| NAT-IR | 13.91 | 16.77 | 24.45 | 25.73 | 21.86 |
| CTC Loss | 17.68 | 19.80 | 19.93 | 24.71 | – |
| NAT w/ FT | 17.69 | 21.47 | 27.29 | 29.06 | 20.32 |
| NAT-REG | 20.65 | 24.77 | – | – | 23.89 |
| CMLM-small | 15.06 | 19.26 | 20.12 | 20.36 | – |
| CMLM-base | 18.12 | 22.26 | 23.65 | 22.78 | – |
| FlowSeq-base | 21.45 | 26.16 | 29.34 | 30.44 | **27.55** |
| FlowSeq-large | **23.72** | **28.39** | **29.73** | **30.72** | – |

Table 1: BLEU scores on three MT benchmark datasets for FlowSeq with argmax decoding and baselines with purely non-autoregressive decoding method. The first and second block are results of models trained w/w.o. knowledge distillation, respectively.

**Modules and Hyperparameters** We implement the encoder, decoder and posterior networks with standard (unmasked) Transformer layers (Vaswani et al., 2017). For WMT datasets, the encoder consists of 6 layers, and the decoder and posterior are composed of 4 layers, and 8 attention heads. and for IWSLT, the encoder has 5 layers, and decoder and posterior have 3 layers, and 4 attention heads. The prior flow consists of 3 scales with the number of steps $[48, 48, 16]$ from bottom to top. To dissect the impact of model dimension on translation quality and speed, we perform experiments on two versions of FlowSeq with $d_{model}/d_{hidden} = 256/512$ (base) and $d_{model}/d_{hidden} = 512/1024$ (large). More model details are provided in Appendix B.

**Optimization** Parameter optimization is performed with the Adam optimizer (Kingma and Ba, 2014) with $\beta = (0.9, 0.999)$ and $\epsilon = 1e-6$. Each mini-batch consist of 2048 sentences. The learning rate is initialized to $5e-4$, and exponentially decays with rate $0.999995$. The gradient clipping cutoff is $1.0$. For all the FlowSeq models, we apply $0.1$ label smoothing and averaged the 5 best checkpoints to create the final model.

At the beginning of training, the posterior network is randomly initialized, producing noisy supervision to the prior. To mitigate this issue, we first set the weight of the KL term in ELBO to zero for 30,000 updates to train the encoder, decoder and posterior networks. Then the KL weight linearly increases to one for another 10,000 updates, which we found essential to accelerate training and achieve stable performance.

| Models | WMT2014 EN-DE | WMT2014 DE-EN | WMT2016 EN-RO | WMT2016 RO-EN |
|---|---|---|---|---|
| Autoregressive Methods | | | | |
| Transformer-base | 27.30 | – | – | – |
| Our Implementation | 27.16 | 31.44 | 32.92 | 33.09 |
| Raw Data | | | | |
| CMLM-base (refinement 4) | 22.06 | – | 30.89 | – |
| CMLM-base (refinement 10) | **24.65** | – | **32.53** | – |
| FlowSeq-base (IWD $n = 15$) | 20.20 | 24.63 | 30.61 | 31.50 |
| FlowSeq-base (NPD $n = 15$) | 20.81 | 25.76 | 31.38 | 32.01 |
| FlowSeq-base (NPD $n = 30$) | 21.15 | 26.04 | 31.74 | 32.45 |
| FlowSeq-large (IWD $n = 15$) | 22.94 | 27.16 | 31.08 | 32.03 |
| FlowSeq-large (NPD $n = 15$) | 23.14 | 27.71 | 31.97 | 32.46 |
| FlowSeq-large (NPD $n = 30$) | 23.64 | **28.29** | 32.35 | **32.91** |
| Knowledge Distillation | | | | |
| NAT-IR (refinement 10) | 21.61 | 25.48 | 29.32 | 30.19 |
| NAT w/ FT (NPD $n = 10$) | 18.66 | 22.42 | 29.02 | 31.44 |
| NAT-REG (NPD $n = 9$) | 24.61 | 28.90 | – | – |
| LV NAR (refinement 4) | 24.20 | – | – | – |
| CMLM-small (refinement 10) | 25.51 | 29.47 | 31.65 | 32.27 |
| CMLM-base (refinement 10) | **26.92** | **30.86** | **32.42** | **33.06** |
| FlowSeq-base (IWD $n = 15$) | 22.49 | 27.40 | 30.59 | 31.58 |
| FlowSeq-base (NPD $n = 15$) | 23.08 | 28.07 | 31.35 | 32.11 |
| FlowSeq-base (NPD $n = 30$) | 23.48 | 28.40 | 31.75 | 32.49 |
| FlowSeq-large (IWD $n = 15$) | 24.70 | 29.44 | 31.02 | 31.97 |
| FlowSeq-large (NPD $n = 15$) | 25.03 | 30.48 | 31.89 | 32.43 |
| FlowSeq-large (NPD $n = 30$) | 25.31 | 30.68 | 32.20 | 32.84 |

Table 2: BLEU scores on two WMT datasets of models using advanced decoding methods. The first block are Transformer-base (Vaswani et al., 2017). The second and the third block are results of models trained w/w.o. knowledge distillation, respectively. $n = l \times r$ is the total number of candidates for rescoring.

**Knowledge Distillation** Previous work on non-autoregressive generation (Gu et al., 2018; Ghazvininejad et al., 2019) has used translations produced by a pre-trained autoregressive NMT model as the training data, noting that this can significantly improve the performance. We analyze the impact of distillation in § 4.2.

## 4.2 Main Results

We first conduct experiments to compare the performance of FlowSeq with strong baseline models, including NAT w/ Fertility (Gu et al., 2018), NAT-IR (Lee et al., 2018), NAT-REG (Wang et al., 2019), LV NAR (Shu et al., 2019), CTC Loss (Libovický and Helcl, 2018), and CMLM (Ghazvininejad et al., 2019).

Table 1 provides the BLEU scores of FlowSeq with argmax decoding, together with baselines with purely non-autoregressive decoding methods that generate output sequence in one parallel pass. The first block lists results of models trained on raw data, while the second block are results using

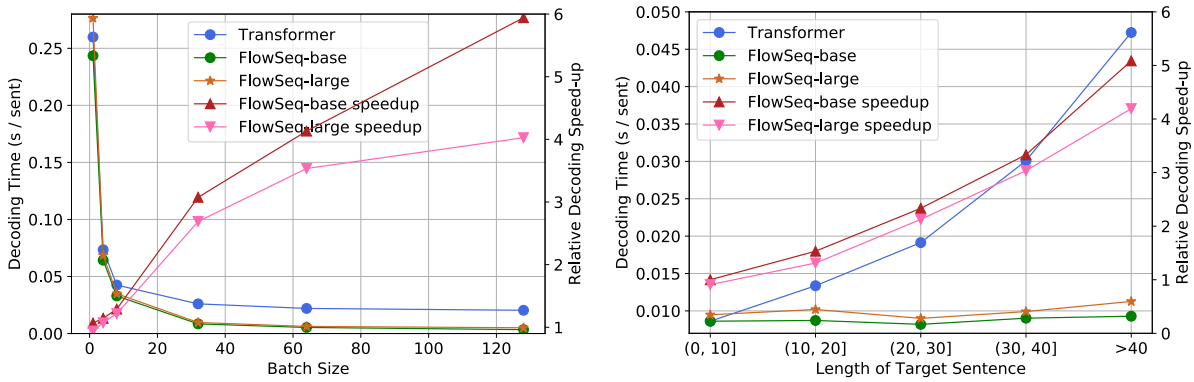


(a) batch size

(b) target length

Figure 4: The decoding speed of Transformer (batched, beam size 5) and FlowSeq on WMT14 EN-DE test set (a) w.r.t different batch sizes (b) bucketed by different target sentence lengths (batch size 32).

knowledge distillation. Without using knowledge distillation, FlowSeq base model achieves significant improvement (more than 9 BLEU points) over CMLM-base and LV NAR. It demonstrates the effectiveness of FlowSeq on modeling the complex interdependence in target languages.

Towards the effect of knowledge distillation, we can mainly obtain two observations: i) Similar to the findings in previous work, knowledge distillation still benefits the translation quality of FlowSeq. ii) Compared to previous models, the benefit of knowledge distillation on FlowSeq is less significant, yielding less than 3 BLEU improvement on WMT2014 DE-EN corpus, and even no improvement on WMT2016 RO-EN corpus. The reason might be that FlowSeq does not rely much on knowledge distillation to alleviate the multi-modality problem.

Table 2 illustrates the BLEU scores of FlowSeq and baselines with advanced decoding methods such as iterative refinement, IWD and NPD rescoring. The first block in Table 2 includes the baseline results from autoregressive Transformer. For the sampling procedure in IWD and NPD, we sampled from a reduced-temperature model (Kingma and Dhariwal, 2018) to obtain high-quality samples. We vary the temperature within $\{0.1, 0.2, 0.3, 0.4, 0.5, 1.0\}$ and select the best temperature based on the performance on development sets. The analysis of the impact of sampling temperature and other hyper-parameters on samples is in § 4.4. For FlowSeq, NPD obtains better results than IWD, showing that FlowSeq still falls behind auto-regressive Transformer on model data distributions. Comparing with CMLM (Ghazvininejad et al., 2019) with 10 iterations of refinement, which is a contemporaneous work that achieves state-of-the-art translation performance, FlowSeq obtains competitive performance on both WMT2014 and WMT2016 corpora, with only slight degradation in translation quality. Leveraging iterative refinement to further improve the performance of FlowSeq has been left to future work.

### 4.3 Analysis on Decoding Speed

In this section, we compare the decoding speed (measured in average time in seconds required to decode one sentence) of FlowSeq at test time with that of the autoregressive Transformer model. We use the test set of WMT14 EN-DE for evaluation and all experiments are conducted on a single NVIDIA TITAN X GPU.

**How does batch size affect the decoding speed?** First, we investigate how different decoding batch size can affect the decoding speed. We vary the decoding batch size within $\{1, 4, 8, 32, 64, 128\}$. Figure. 4a shows that for both FlowSeq and Transformer decoding is faster when using a larger batch size. However, FlowSeq has much larger gains in the decoding speed w.r.t. the increase in batch size, gaining a speed up of 594% of base model and 403% of large model when using a batch size of 128. We hypothesize that this is because the operations in FlowSeq are more friendly to batching while the Transformer model with beam search at test time is less efficient in benefiting from batching.

**How does sentence length affect the decoding speed?** Next, we examine if sentence length is a major factor affecting the decoding speed. We bucket the test data by the target sentence length.

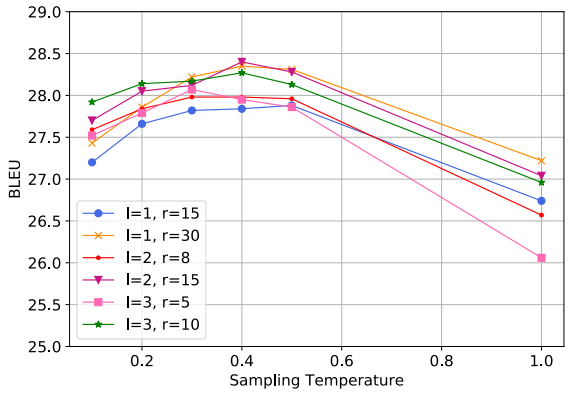


Figure 5: Impact of sampling hyperparameters on the rescoring BLEU on the dev set of WMT14 DE-EN. Experiments are performed with FlowSeq-base trained with distillation data. $l$ is the number of length candidates. $r$ is the number of samples for each length.

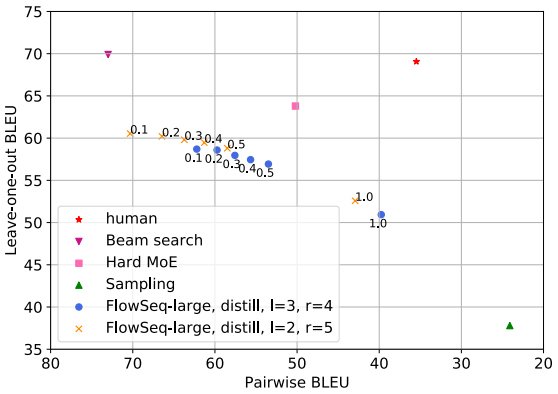


Figure 6: Comparisons of FlowSeq with human translations, beam search and sampling results of Transformer-base, and mixture-of-experts model (Hard MoE (Shen et al., 2019)) on the averaged leave-one-out BLEU score v.s pairwise-BLEU in descending order.

From Fig. 4b, we can see that as the sentence length increases, FlowSeq achieves almost constant decoding time while Transformer has a linearly increasing decoding time. The relative decoding speed up of FlowSeq versus Transformer linearly increases as the sequence length increases. The potential of decoding long sequences with constant time is an attractive property of FlowSeq.

### 4.4 Analysis of Rescoring Candidates

In Fig. 5, we analyze how different sampling hyperparameters affect the performance of rescoring. First, we observe that the number of samples $r$ for each length is the most important factor. The performance is always improved with a larger sample size. Second, a larger number of length candidates does not necessarily increase the rescoring performance. Third, we find that a larger sampling temperature (0.3 - 0.5) can increase the diversity of translations and leads to better rescoring BLEU. However, the latent samples become noisy when a large temperature (1.0) is used.

### 4.5 Analysis of Translation Diversity

Following (Shen et al., 2019), we analyze the output diversity of FlowSeq. Shen et al. (2019) proposed pairwise-BLEU and BLEU computed in a leave-one-out manner to calibrate the diversity and quality of translation hypotheses. A lower pairwise-BLEU score implies a more diverse hypothesis set. And a higher BLEU score implies a better translation quality. We experiment on a subset of test set of WMT14-ENDE with ten references each sentence (Ott et al., 2018). In Fig. 6, we compare FlowSeq with other multi-hypothesis generation methods (ten hypotheses each sentence) to analyze how well the generation outputs of FlowSeq are in terms of diversity and quality. The right corner area of the figure indicates the ideal generations: high diversity and high quality. While FlowSeq still lags behind the autoregressive generations, by increasing the sampling temperature it provides a way of generating more diverse outputs while keeping the translation quality almost unchanged. More analysis of translation outputs and detailed results are provided in the Appendix D and E.

## 5 Conclusion

We propose FlowSeq, an efficient and effective model for non-autoregressive sequence generation by using generative flows. One potential direction for future work is to leverage iterative refinement techniques such as masked language models to further improve translation quality. Another exciting direction is to, theoretically and empirically, investigate the latent space in FlowSeq, hence providing deep insights of the model, even enhancing controllable text generation.

## Acknowledgments

This work was supported in part by DARPA grant FA8750-18-2-0018 funded under the AIDA program and grant HR0011-15-C-0114 funded under the LORELEI program. Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of DARPA. The authors thank Amazon for their gift of AWS cloud credits and anonymous reviewers for their helpful suggestions.

## References


Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *In International Conference on Learning Representations (ICLR)*.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. 2015. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. 2016. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Constant-time machine translation with conditional masked language models. *arXiv preprint arXiv:1904.09324*.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. *In International Conference on Learning Representations (ICLR)*.

Jiatao Gu, Qi Liu, and Kyunghyun Cho. 2019. Insertion-based decoding with automatically inferred generation order. *arXiv preprint arXiv:1902.01370*.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. 2016. Improving variational inference with inverse autoregressive flow. *The 29th Conference on Neural Information Processing Systems*.

Durk P Kingma and Prafulla Dhariwal. 2018. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pages 10215–10224.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1173–1182.

Jindřich Libovický and Jindřich Helcl. 2018. End-to-end non-autoregressive neural machine translation with connectionist temporal classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3016–3021.

Xuezhe Ma and Eduard Hovy. 2019. Macow: Masked convolutional generative flow. *arXiv preprint arXiv:1902.04208*.

Xuezhe Ma, Zecong Hu, Jingzhou Liu, Nanyun Peng, Graham Neubig, and Eduard Hovy. 2018. Stack-pointer networks for dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1403–1414.

Xuezhe Ma, Chunting Zhou, and Eduard Hovy. 2019. Mae: Mutual posterior-divergence regularization for variational autoencoders. In *Proceedings of the 7th International Conference on Learning Representations (ICLR-2019)*, New Orleans, Louisiana, USA.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional LSTM. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 51–61, Berlin, Germany. Association for Computational Linguistics.

Myle Ott, Michael Auli, David Grangier, et al. 2018. Analyzing uncertainty in neural machine translation. In *International Conference on Machine Learning*, pages 3953–3962.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

George Papamakarios, Theo Pavlakou, and Iain Murray. 2017. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pages 2338–2347.

Ryan Prenger, Rafael Valle, and Bryan Catanzaro. 2019. Waveglow: A flow-based generative network for speech synthesis. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3617–3621. IEEE.

Danilo Jimenez Rezende and Shakir Mohamed. 2015. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning-Volume 37*, pages 1530–1538. JMLR. org.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389.

Tianxiao Shen, Myle Ott, Michael Auli, et al. 2019. Mixture models for diverse machine translation: Tricks of the trade. In *International Conference on Machine Learning*, pages 5719–5728.

Raphael Shu, Jason Lee, Hideki Nakayama, and Kyunghyun Cho. 2019. Latent-variable non-autoregressive neural machine translation with deterministic inference using a delta posterior. *arXiv preprint arXiv:1908.07181*.

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. 2019. Insertion transformer: Flexible sequence generation via insertion operations. *arXiv preprint arXiv:1902.03249*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164.

Martin J Wainwright, Michael I Jordan, et al. 2008. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305.

Yiren Wang, Fei Tian, Di He, Tao Qin, ChengXiang Zhai, and Tie-Yan Liu. 2019. Non-autoregressive machine translation with auxiliary regularization. *arXiv preprint arXiv:1902.10245*.

Zachary Ziegler and Alexander Rush. 2019. Latent normalizing flows for discrete sequences. In *International Conference on Machine Learning*, pages 7673–7682.

# Compositional Generalization for Primitive Substitutions

**Yuanpeng Li, Liang Zhao, Jianyu Wang, Joel Hestness**
Baidu Research
{yuanpeng16,jthestness}@gmail.com
{zhaoliang07,wangjianyu02}@baidu.com

## Abstract

Compositional generalization is a basic mechanism in human language learning, but current neural networks lack such ability. In this paper, we conduct fundamental research for encoding compositionality in neural networks. Conventional methods use a single representation for the input sentence, making it hard to apply prior knowledge of compositionality. In contrast, our approach leverages such knowledge with two representations, one generating attention maps, and the other mapping attended input words to output symbols. We reduce the entropy in each representation to improve generalization. Our experiments demonstrate significant improvements over the conventional methods in five NLP tasks including instruction learning and machine translation. In the SCAN domain, it boosts accuracies from 14.0% to 98.8% in Jump task, and from 92.0% to 99.7% in TurnLeft task. It also beats human performance on a few-shot learning task. We hope the proposed approach can help ease future research towards human-level compositional language learning. Source code is available online[1].

## 1 Introduction

Humans learn language in a flexible and efficient way by leveraging *systematic compositionality*, the algebraic capacity to understand and produce large amount of novel combinations from known components (Chomsky, 1957; Montague, 1970). For example, if a person knows how to "step", "step twice" and "jump", then it is natural for the person to know how to "jump twice" (Figure 1). This compositional generalization is critical in human cognition (Minsky, 1986; Lake et al., 2017), and it helps humans learn language from a limited amount of data, and extend to unseen sentences.

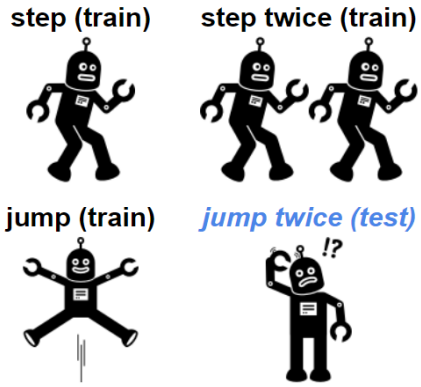


Figure 1: Problem illustration. If a person knows how to "step", "step twice" and "jump", then the person should know how to "jump twice". Although this is straightforward for humans, it remains an open problem to equip machines with such ability.

In recent years, deep neural networks have made many breakthroughs in various problems (LeCun et al., 2015; Krizhevsky et al., 2012; Yu and Deng, 2012; He et al., 2016; Wu and et al, 2016). However, there have been critiques that neural networks do not have compositional generalization ability (Fodor and Pylyshyn, 1988; Marcus, 1998; Fodor and Lepore, 2002; Marcus, 2003; Calvo and Symons, 2014).

Our observation is that conventional methods use a single representation for the input sentence, which makes it hard to apply prior knowledge of compositionality. In contrast, our approach leverages such knowledge with two representations, one generating attention maps, and the other mapping attended input words to output symbols. We reduce each of their entropies to improve generalization. This mechanism equips the proposed approach with the ability to understand and produce novel combinations of known words and to achieve compositional generalization.

In this paper, we focus on compositinal gener-

[1] https://github.com/yli1/CGPS



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4293–4302,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| | |
|---|---|
| **jump** | JUMP |
| run after run left | LTURN RUN RUN |
| look left twice and look opposite right | LTURN LOOK LTURN LOOK RTURN RTURN LOOK |
| **jump** twice after look | LOOK JUMP JUMP |
| turn left after **jump** twice | JUMP JUMP LTURN |
| **jump** right twice after **jump** left twice | LTURN JUMP LTURN JUMP RTURN JUMP RTURN JUMP |

(a) Jump task. In training, "jump" only appears as a single command. In test, it appears with other words.

| | |
|---|---|
| **turn left** | LTURN |
| run thrice and jump right | RUN RUN RUN RTURN JUMP |
| look left thrice after run left twice | LTURN RUN LTURN RUN LTURN LOOK LTURN LOOK LTURN LOOK |
| look twice and **turn left** twice | LOOK LOOK LTURN LTURN |
| **turn left** thrice and **turn left** | LTURN LTURN LTURN LTURN |
| **turn left** twice after look opposite right twice | RTURN RTURN LOOK RTURN RTURN LOOK LTURN LTURN |

(b) TurnLeft task. In training, "turn left" only appears as a single command. In test, it appears with other words.

Table 1: Examples of SCAN input commands (left) and output action sequences (right). Upper section is for training, and lower section is for testing. These tasks are difficult because "jump" (or "turn left") does not appear with other words in training.

alization for primitive substitutions. We use Primitive tasks in the SCAN (Lake and Baroni, 2018) dataset—a set of command-action pairs—for illustrating how the proposed algorithm works. In a Primitive task, training data include a single primitive command, and other training commands do not contain the primitive. In test data, the primitive appears together with other words. A primitive can be "jump" (Jump, see Table 1a for examples) or "turn left" (TurnLeft, see Table 1b for examples). These tasks are difficult because "jump" (or "turn left") does not appear with other words in training. A model without compositionality handling will easily learn a wrong rule such as "any sentence with more than one word should not contain the 'jump' action". This rule fits the training data perfectly, and may reduce training loss quickly, but it does not work for test data. In contrast, our algorithm can process a sentence such as "jump twice after look" correctly by generating a sequence of three attention maps from function representation and then a sequence of actions "LOOK JUMP JUMP" by decoding the primitive representation from the attended command words (see the example in Figure 2).

Besides Primitive tasks, we experiment on more tasks in SCAN and other datasets for instruction learning and machine translation. We also introduce an extension of SCAN dataset, dubbed as SCAN-ADJ, which covers additional compositional challenges. Our experiments demonstrate that the proposed method achieves significant improvements over the conventional methods. In the SCAN domain, it boosts accuracies from 14.0% to 98.8% in Jump task, and from 92.0% to 99.7% in TurnLeft task. It also beats human performance on a few-shot learning task.

## 2 Approach

In compositional generalization tasks, we consider both input $X$ and output $Y$ have multiple components that are not labeled in data. Suppose input has two components $X_1$ and $X_2$, and output has two components $Y_1$ and $Y_2$, then we can define compositional generalization probabilistically as follows. In training,

$$P(X_1) > 0 \text{ and } P(X_2) > 0,$$
$$P(X_1, X_2) = 0,$$
$$P(Y_1|X_1) \text{ and } P(Y_2|X_2) \text{ are high.}$$

In test,

$$P(X_1, X_2) > 0,$$
$$P(Y_1, Y_2|X_1, X_2) \text{ is predicted high.}$$

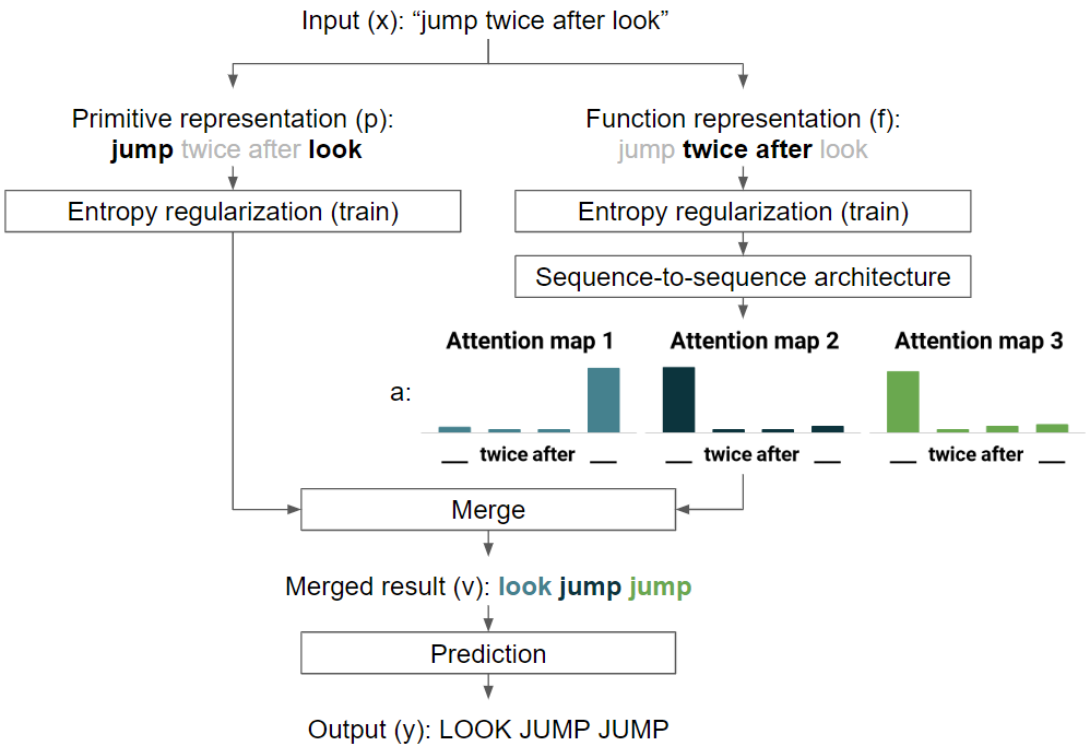


Figure 2: Illustration of network structure and flowchart for the proposed approach. Input is a sequence of words. Output is a sequence of actions. An input sentence is converted to primitive representation (left) and function representation (right). In training, we reduce entropy of two representations. Function representation is used to generate a sequence of attention maps. Each attention map is used to select an embedding from primitive representation to output an action.

We expect this is possible when we have prior knowledge that $Y_1$ depends only on $X_1$ and $Y_2$ depends only on $X_2$.

$$P(Y_1|X_1, X_2, Y_2) = P(Y_1|X_1),$$
$$P(Y_2|X_1, X_2, Y_1) = P(Y_2|X_2).$$

Therefore, with chain rule,

$$P(Y_1, Y_2|X_1, X_2) = P(Y_1|X_1)P(Y_2|X_2).$$

Since $P(Y_1|X_1)$ and $P(Y_2|X_2)$ are both high in training, if they generalize well, they should also be high in test, so that their product should be high in test. To make $P(Y_1|X_1)$ and $P(Y_2|X_2)$ generalize well at test time, we hope to remove unnecessary information from $X_1$ and $X_2$. In other words, we want $X_1$ ($X_2$) to be a *minimal sufficient statistic* for $Y_1$ ($Y_2$).

In this task, input $X$ is a word sequence, and output $Y$ is an action sequence. We consider $X$ has two types of information: which actions are present ($X_1$), and how the actions should be ordered ($X_2$). $Y$ is constructed by the output action types ($Y_1$), and output action order ($Y_2$). $Y_1$ depends only on $X_1$, and $Y_2$ depends only on $X_2$.

Based on these arguments, we design the algorithm with the following objectives:

(i) Learn two representations for input.

(ii) Reduce entropy in each representation.

(iii) Make output action type depend only on one representation, and output action order depend only on the other.

Figure 2 shows an illustration of flowchart for the algorithm design (see Algorithm 1 for details). We describe it further here.

### 2.1 Input and Output

Input $x$ contains a sequence of $n$ words, where each input word is from an input vocabulary of size $U$. Output $y$ contains a sequence of $m$ action symbols, where each output symbol is from an output vocabulary of size $V$. Both vocabularies contain an end-of-sentence symbol which appears at the end of $x$ and $y$, respectively. The model output $\hat{y}$ is a prediction for $y$. Both input words and

**Algorithm 1** Proposed approach. Decoder in Seq2seq architecture is autoregressive model here.

**Require:** $V$: output vocabulary size
$k_p, k_f$: embedding sizes
$\alpha$: 1 for training, 0 for inference
Seq2seq: seq2seq architecture
**Input:** $x$: sequence of one-hot representations
**Output:** $\hat{y}$: sequence of distributions

**for** $i = 1, \ldots, n$ **do**
  $p_i = \text{Emb}_p(x_i) \in \mathbb{R}^{k_p}$
  $f_i = \text{Emb}_f(x_i) \in \mathbb{R}^{k_f}$
**end for**
$p' = p + \alpha\epsilon_p \in \mathbb{R}^{n \times k_p}, \epsilon_p \sim \mathcal{N}(0, I)$
$f' = f + \alpha\epsilon_f \in \mathbb{R}^{n \times k_f}, \epsilon_f \sim \mathcal{N}(0, I)$
decoder = Seq2seq($f'$)
**for** $j = 1, \ldots,$ max length **do**
  $b_j = \text{decoder.next}() \in \mathbb{R}^n$
  $a_j = \text{Softmax}(b_j) \in \mathbb{R}^n$
  $v_j = a_j p' \in \mathbb{R}^{k_p}$
  $l_j = f_{\text{predict}}(v_j) \in \mathbb{R}^V$
  $\hat{y}_j = \text{Softmax}(l_j) \in (0, 1)^V$
  **break if** $\operatorname{argmax} \hat{y}_j$ is <EOS>
**end for**

output symbols are in one-hot representation, i.e.,

$$x = [x_1, \ldots, x_n] \in \{0, 1\}^{U \times n},$$
$$y = [y_1, \ldots, y_m] \in \{0, 1\}^{V \times m}.$$

### 2.2 Primitive Representation and Function Representation

To achieve objective (i), we use two representations for an input sentence. For each word, we use two word embeddings for primitive and functional information.

$$p_i = \text{Emb}_p(x_i) \in \mathbb{R}^{k_p},$$
$$f_i = \text{Emb}_f(x_i) \in \mathbb{R}^{k_f}.$$

We concatenate $p_i$ and $f_i$ to form the primitive representation and functional representation for the entire input sequence, i.e.,

$$p = [p_1, \ldots, p_n] \in \mathbb{R}^{k_p \times n},$$
$$f = [f_1, \ldots, f_n] \in \mathbb{R}^{k_f \times n}.$$

### 2.3 Entropy Regularization

To achieve objective (ii), we reduce the amount of information, or entropy of the representations. We do this by regularizing the $L_2$ norm of the representations, and then adding noise to them. This reduces channel capacities, and thus entropy for the representations.

$$p' = p + \alpha\epsilon_p \in \mathbb{R}^{k_p \times n}, \epsilon_p \sim \mathcal{N}(0, I),$$
$$f' = f + \alpha\epsilon_f \in \mathbb{R}^{k_f \times n}, \epsilon_f \sim \mathcal{N}(0, I),$$
$$\mathcal{L}_{\text{regularize}} = L_2(p) + L_2(f).$$

Here, $\alpha$ is the weight for noise. Noise is helpful in training, but is not necessary in inference. Therefore, we set $\alpha$ as a hyper-parameter for training and $\alpha = 0$ for inference.

### 2.4 Combine Representations and Generate Output

To achieve objective (iii), we want to make the output action types depend only on primitive representation, and the output action order depend only on function representation. Attention mechanism is a good fit for this purpose. We generate a sequence of attention maps from function representation, and use each attention map to get the weighted average of primitive representation to predict the corresponding action.

We use attention mechanism with sequence-to-sequence architectures to aggregate representations in the following ways: (1) Instead of outputting a sequence of logits on vocabulary, we emit a sequence of logits for attention maps on input positions (see Figure 2 for an example). (2) During decoding for autoregressive models, we do not use the previous output symbol as input for the next symbol prediction, because output symbols contain information for primitives. We still use the last decoder hidden state as input. Non-autoregressive models should not have this problem. (3) We end decoding when the last output is end-of-sentence symbol.

More specifically, we feed the function representation, $f'$, to a sequence-to-sequence neural network for decoding. At each step $j$, we generate $b_j \in \mathbb{R}^n$ from the decoder, and use Softmax to obtain the attention map, $a_j$. With attention maps, we extract primitives and generate output symbols. To do that, we compute the weighted average, $v_j$, on noised primitive representations, $p'$, with attention, $a_j$. Then we feed it to a one-layer fully connected network, $f_{\text{predict}}$. We use Softmax to compute the output distribution, $\hat{y}_j$. The decoding ends when

$\text{argmax}\, \hat{y}_j$ is end-of-sentence symbol.

$$
\begin{aligned}
a_j &= \text{Softmax}(b_j) \in \mathbb{R}^n, \\
v_j &= a_j p' \in \mathbb{R}^{k_p}, \\
l_j &= f_{\text{predict}}(v_j) \in \mathbb{R}^V, \\
\hat{y}_j &= \text{Softmax}(l_j) \in (0,1)^V.
\end{aligned}
$$

### 2.5 Loss

We use the cross entropy of $y$ and $\hat{y}$ as the prediction loss, $\mathcal{L}_{\text{prediction}}$, and the final loss, $\mathcal{L}$, is the combination of prediction loss and entropy regularization loss (Section 2.3). $\lambda$ is the regularization weight.

$$
\begin{aligned}
\mathcal{L}_{\text{prediction}} &= \sum_{j=1}^{m} \text{CrossEntropy}(y_j, \hat{y}_j), \\
\mathcal{L} &= \mathcal{L}_{\text{prediction}} + \lambda \mathcal{L}_{\text{regularize}}.
\end{aligned}
$$

## 3 Experiments

We ran experiments on instruction learning and machine translation tasks. Following Lake and Baroni (2018), we used sequence-level accuracy as the evaluation metric. A prediction is considered correct if and only if it perfectly matches the reference. We ran all experiments five times with different random seeds for the proposed approach, and report mean and standard deviation (note that their sum may exceed 100%). The Appendix contains details of models and experiments.

### 3.1 SCAN Primitive Tasks

We used the SCAN dataset (Lake and Baroni, 2018), and targeted the Jump and TurnLeft tasks. For the Jump and TurnLeft tasks, "jump" and "turn left" appear only as a single command in their respective training sets, but they appear with other words at test time (see Table 1a and Table 1b, respectively).

We used multiple state-of-the-art or representative methods as baselines. L&B methods are from Lake and Baroni (2018), RNN and GRU are best models for each task from Bastings et al. (2018). Ptr-Bahdanau and Ptr-Luong are best methods from Kliegl and Xu (2018).

We applied the same configuration for all tasks (see Appendix A for details). The result in Table 2 shows that the proposed method boosts accuracies from 14.0% to 98.8% in Jump task, and from 92.0% to 99.7% in TurnLeft task.

| Method | Jump | TurnLeft |
|---|---|---|
| L&B best overall | 0.1 | 90.0 |
| L&B best | 1.2 | 90.3 |
| RNN +Attn -Dep | 2.7 ± 1.7 | 92.0 ± 5.8 |
| GRU +Attn | 12.5 ± 6.6 | 59.1 ± 16.8 |
| Ptr-Bahdanau | 3.3 ± 0.7 | 91.7 ± 3.6 |
| Ptr-Loung | 14.0 ± 2.8 | 66.0 ± 5.6 |
| Proposed | **98.8** ± **1.4** | **99.7** ± **0.4** |

Table 2: Test accuracy (mean ± std %) for SCAN Primitive tasks. L&B is from Lake and Baroni (2018). RNN and GRU are best models for each task from Bastings et al. (2018). Ptr-Bahdanau and Ptr-Loung are from Kliegl and Xu (2018). The proposed method achieves significantly higher performance than baselines.

### 3.2 SCAN Template-matching

We also experimented with the SCAN template-matching task. The task and dataset descriptions can be found in Loula et al. (2018b). There are four tasks in this dataset: *jump around right*, *primitive right*, *primitive opposite right* and *primitive around right*. Each requires compositional generalization. Please see Appendix B for more details.

Table 3 summarizes the results on SCAN template-matching tasks. The baseline methods are from Loula et al. (2018b). The proposed method achieves significantly higher performance than baselines in all tasks. This experiment demonstrates that the proposed method has better compositional generalization related to template matching.

| Condition | Baseline | Proposed |
|---|---|---|
| Jump around | 98.4 ± 0.5 | **100.0** ± **0.0** |
| Prim. | 23.5 ± 8.1 | **99.7** ± **0.5** |
| Prim. opposite | 47.6 ± 17.7 | **89.3** ± **5.5** |
| Prim. around | 2.5 ± 2.7 | **83.2** ± **13.2** |

Table 3: Test accuracy (mean ± std %) for SCAN template-matching tasks. The baseline methods are from Loula et al. (2018b). The proposed method achieves significantly higher performance than baselines.

### 3.3 Primitive and Functional Information Exist in One Word

One limitation of SCAN dataset is that each word contains either primitive or functional information, but not both. For example, "jump" contains only primitive information, but no functional information. On the other hand, "twice" contains only

functional information, but no primitive information. However, in general cases, each word may contain both primitive and functional information. The functional information of a word usually appears as a word class, such as part of speech. For example, in translation, a sequence of part-of-speech tags determines the output sentence structure (functional), and primitive information of each input word determines the corresponding output word.

To explore whether the proposed method works when a word contains both primitive and functional information, we introduced an extension of SCAN dataset for this purpose. We designed a task SCAN-ADJ to identify adjectives in a sentence, and place them according to their word classes. The adjectives have three word classes: color, size, and material. The word classes appear in random order in input sentences, but they should be in a fixed order in the expected outputs. For example, an input sentence is "push blue metal small cube", and the expected output is "PUSH CUBE SMALL BLUE METAL". For another input "push small metal blue cube", the expected output is the same. Please see Table 9 in Appendix for more details. To evaluate compositional generalization, similarly to the SCAN setting, the material, rubber, appears only with other fixed words in training "push small yellow rubber sphere". However, it appears with other combinations of words at test time. To solve this compositionality problem, a model needs to capture functional information of adjectives to determine their word classes, and primitive information to determine the specific value of the word in its class.

The experiment results are shown in Table 4. The proposed method achieves 100.0% ($\pm$0.0%) accuracy, and as a comparison, a standard LSTM model can only achieves 2.1% ($\pm$2.2%) accuracy. Please refer to Appendix C for more details. This indicates that the proposed method can extend beyond simple structures in the SCAN dataset to flexibly adapt for words containing both primitive and functional information.

| Method | Accuracy |
|---|---|
| Baseline | 2.1 $\pm$ 2.2 |
| Proposed | **100.0** $\pm$ **0.0** |

Table 4: Test accuracy (mean $\pm$ std %) for SCAN-ADJ task.

### 3.4 Few-shot Learning Task

We also probed the few-shot learning ability of the proposed approach using the dataset from Lake et al. (2019). The dataset is shown in Appendix Figure 4. In the few-shot learning task, human participants learned to produce abstract outputs (colored circles) from instructions in pseudowords. Some pseudowords are primitive corresponding to a single output symbol, while others are function words that process items. A primitive ("zup") is presented only in isolation during training and evaluated with other words during test. Participants are expected to learn each function from limited number of examples, and to generalize in a compositional way.

We trained a model from all training samples and evaluate them on both Primitive test samples and all test samples. We used a similar configuration to SCAN task (Appendix D) to train the model, and compared the results with baselines of human performance and conventional sequence to sequence model (Lake et al., 2019). In the Primitive task, we used the average over sample accuracies to estimate the accuracy of human participants, and the upper bound accuracy of the sequence-to-sequence model.

Table 5 shows the results that the proposed method achieves high accuracy on both test sets. In the Primitive task, the proposed method beats human performance. This indicates that the proposed method is able to learn compositionality from only a few samples. To our best knowledge, this is the first time machines beat humans in a few-shot learning task, and it breaks the common sense that humans are better than machines in learning from few samples.

| Method | Primitive | All |
|---|---|---|
| Human | 82.8 | 84.3 |
| Baseline | $\leq$ 3.1 | 2.5 |
| Proposed | **95.0** $\pm$ **6.8** | 76.0 $\pm$ 5.5 |

Table 5: Test accuracy (mean $\pm$ std %) for few-shot learning task. The baseline methods are from Lake et al. (2019). The proposed method achieves performance superior to humans in Primitive task.

### 3.5 Compositionality in Machine Translation

We also investigated whether the proposed approach is applicable to other sequence-to-sequence problems. As an example, we ran a proof-of-

concept machine translation experiment. We consider the English-French translation task from Lake and Baroni (2018). To evaluate compositional generalization, for a word "dax", the training data contains only one pattern of sentence pair ("I am daxy", "je suis daxiste"), but test data contains other patterns. Appendix E provides more details on dataset and model configuration.

Compared to the baseline method, the proposed method increases sentence accuracy from 12.5% to 62.5% (±0.0%). Further, we find that other predicted outputs are even correct translations though they are different from references (Table 10 in Appendix). If we count them as correct, the adjusted accuracy for the proposed approach is 100.0% (±0.0%). This experiment shows that the proposed approach has promise to be applied to real-world tasks.

| Method | Accuracy |
|---|---|
| Baseline | 12.5 |
| Proposed | **62.5** |

Table 6: Test accuracy (%) for machine translation task. The baseline method is from Lake and Baroni (2018).

## 4 Discussion

Our experiments show that the proposed approach results in significant improvements on many tasks. To better understand why the approach achieves these gains, we designed the experiments on SCAN domain to address the following questions: (1) Does the proposed model work in the expected way that humans do (i.e., visualization)? (2) What factors of the proposed model contribute to the high performance (i.e., ablation study)? (3) Does the proposed method influence other tasks?

### 4.1 Visualization of Attention Maps

To investigate whether the proposed method works in the expected way, we visualize the model's attention maps. These maps indicate whether the sequence-to-sequence model produces correct position sequences.

We visualized these activations for one sample from the Jump task and one from the TurnLeft task (Figure 3). In both figures, the horizontal dimension is the input sequence position, and the vertical dimension is the output position. Each figure is an $m \times n$ matrix, where $m$ and $n$ are output and input lengths, including end-of-sentence symbols. The rows correspond to $m$ attention maps, where the $i$th row corresponds to the attention map for the $i$th output ($i = 1, ..., m$). The values in each row sum to 1.

We expect that each attention map attends on the corresponding primitive input word. For the output end-of-sentence symbol, we expect that the attention is on the input end-of-sentence symbol. The visualization in both figures align with our expectation. When the model should replicate some input word multiple times, it correctly attends on that word in a manner similar to human understanding. In the Jump example, "jump" appears twice in the input. As expected, the generated attentions are on the second instance for the early part of the output, and on the first instance for the later part of the output. Similarly, the attentions are correct in the TurnLeft example with two words of "left".

These visualizations demonstrate that the models work in the way we expected, correctly parsing the organization of complex sentences. We have successfully encoded our prior knowledge to the network, and enabled compositional generalization.

### 4.2 Ablation Study

We conducted an ablation study to find out what factors of the proposed approach contribute to its high performance. The main idea of the proposed approach is to use two representations and corresponding entropy regularization. Also, for autoregressive decoders, the approach does not use the last output rather than feeding it as input for decoding. Based on that, we designed the following ablation experiments.

(A) Use only one representation.

(B) Do not use primitive entropy regularization.

(C) Do not use function entropy regularization.

(D) Both B and C.

(E) Use the last output as input in decoding.

The results are summarized in Table 7. Experiment A shows that the accuracy drops in both tasks, indicating that two representations are important factor in the proposed approach. When there are two representations, we find the following from experiments B, C and D. For Jump task, the accuracy drops in these three experiments, indicating both primitive and function entropy regularization are necessary. For TurnLeft task, the accuracy drops only in D, but not in B or C, indicating at least one of entropy regularization is necessary. The dif-

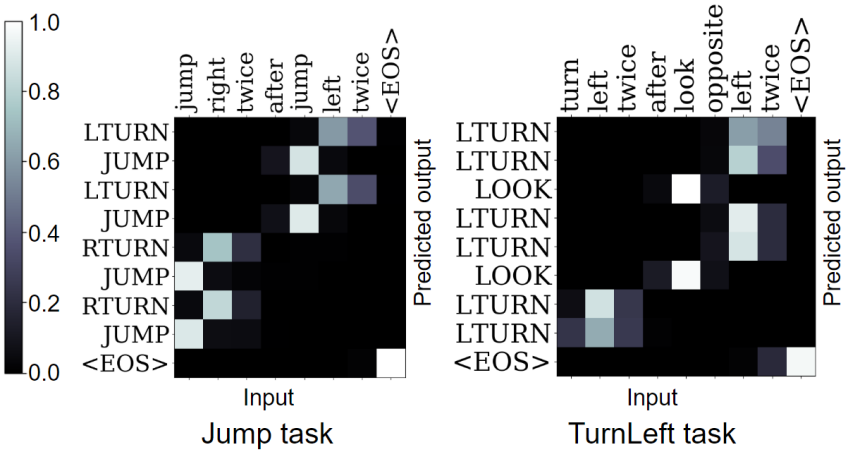


Figure 3: Visualization of attention maps. The figures show that the model solves the problem in a manner similar to humans, identifying the appropriate input to map to each output position. This indicates that the generalizable prior knowledge is successfully encoded in the network.

ference between Jump and TurnLeft tasks may be because TurnLeft is relatively easier than the Jump task. Both "turn" and "left" individually appear with other words in training data, but "jump" does not. Experiment E shows that we should not use last output as next input during decoding.

| | Experiment | Jump | TurnLeft |
|---|---|---|---|
| A | one rep. | 63.4 ± 36.3 | 55.7 ± 31.5 |
| B | no prim reg. | 79.8 ± 11.3 | 99.9 ± 0.1 |
| C | no func reg. | 29.5 ± 36.3 | 99.6 ± 1.0 |
| D | both B and C | 14.1 ± 23.2 | 64.0 ± 40.0 |
| E | decoder input | 68.8 ± 15.0 | 50.3 ± 48.8 |
| | Proposed | 98.8 ± 1.4 | 99.7 ± 0.4 |

Table 7: Ablation: Test accuracy (mean ± std %) for SCAN Primitive tasks.

### 4.3 Influence on Other Tasks

To find whether the proposed method affects other tasks, we conducted experiments on Simple and Length tasks in SCAN dataset. In Simple task, test data follows the same distribution as the training data. In Length task, test commands have longer action sequences than training. Simple task does not require compositional generalization, and Length task requires syntactic generalization, so that they are beyond the scope of this paper. However, we still hope to avoid big performance drop from previous methods. We used the same configuration (Appendix A) as SCAN Primitive tasks. The results in Table 8 confirms that the approach does not significantly reduce performance in other tasks.

| Methods | Simple | Length |
|---|---|---|
| L&B best overall | 99.7 | 13.8 |
| L&B best | 99.8 | 20.8 |
| RNN +Attn -Dep | 100.0 ± 0.0 | 11.7 ± 3.2 |
| GRU +Attn | 100.0 ± 0.0 | 18.1 ± 1.1 |
| Ptr-Bahdanau | - | 13.4 ± 0.8 |
| Ptr-Loung | - | 16.8 ± 0.9 |
| Proposed | 99.9 ± 0.0 | 20.3 ± 1.1 |

Table 8: Test accuracy (mean ± std %) for other SCAN tasks. L&B is from Lake and Baroni (2018). RNN and GRU are best models for each task from Bastings et al. (2018). Ptr-Bahdanau and Ptr-Loung are from Kliegl and Xu (2018). The proposed method does not significantly reduce performance.

## 5 Related Work

Human-level compositional learning has been an important open challenge (Yang et al., 2019), although there is a long history of studying compositionality in neural networks. Classic view (Fodor and Pylyshyn, 1988; Marcus, 1998; Fodor and Lepore, 2002) considers conventional neural networks lack systematic compositionality. With the breakthroughs in sequence to sequence neural networks for NLP and other tasks, such as RNN (Sutskever et al., 2014), Attention (Xu et al., 2015), Pointer Network (Vinyals et al., 2015), and Transformer (Vaswani et al., 2017), there are more contemporary attempts to encode compositionality in sequence to sequence neural networks.

There has been exploration on compositionality in neural networks for systematic behaviour (Wong and Wang, 2007; Brakel and Frank, 2009), count-

ing ability (Rodriguez and Wiles, 1998; Weiss et al., 2018) and sensitivity to hierarchical structure (Linzen et al., 2016). Recently, many related tasks (Lake and Baroni, 2018; Loula et al., 2018a; Liška et al., 2018; Bastings et al., 2018; Lake et al., 2019) and methods (Bastings et al., 2018; Loula et al., 2018a; Kliegl and Xu, 2018; Chang et al., 2018) using a variety of RNN models and attention mechanism have been proposed. These methods make successful generalization when the difference between training and test commands are small.

Our research further enhances the capability of neural networks for compositional generalization with two representations of a sentence and entropy regularization. The proposed approach has shown promising results in various tasks of instruction learning and machine translation.

Compositionality is also important and applicable in multimodal problems, including AI complete tasks like Image Captioning (Karpathy and Fei-Fei, 2015), Visual Question Answering (Antol et al., 2015), Embodied Question Answering (Das et al., 2018). These tasks requires compositional understanding of sequences (captions or questions), where our method could help. Compositional understanding could also help enable better transfer over time that could improve continual learning (Zenke et al., 2017; Aljundi et al., 2018; Chaudhry et al., 2019) ability of AI models.

## 6 Conclusions

This work is a fundamental research for encoding compositionality in neural networks. Our approach leverages prior knowledge of compositionality by using two representations, and by reducing entropy in each representation. The experiments demonstrate significant improvements over the conventional methods in five NLP tasks including instruction learning and machine translation. In the SCAN domain, it boosts accuracies from 14.0% to 98.8% in Jump task, and from 92.0% to 99.7% in TurnLeft task. It also beats human performance on a few-shot learning task. To our best knowledge, this is the first time machines beat humans in a few-shot learning task. This breaks the common sense that humans have advantage over machines in learning from few samples. We hope this work opens a path to encourage machines learn quickly and generalize widely like humans do, and to make machines more helpful in various tasks.

## Acknowledgments

We thank Kenneth Church, Mohamed Elhoseiny, Ka Yee Lun and others for helpful suggestions.

## References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283.

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. 2018. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 139–154.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433.

Joost Bastings, Marco Baroni, Jason Weston, Kyunghyun Cho, and Douwe Kiela. 2018. Jump to better conclusions: SCAN both left and right. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 47–55, Brussels, Belgium. Association for Computational Linguistics.

Philémon Brakel and Stefan Frank. 2009. Strong systematicity in sentence processing by simple recurrent networks. In *31th Annual Conference of the Cognitive Science Society (COGSCI-2009)*, pages 1599–1604. Cognitive Science Society.

Paco Calvo and John Symons. 2014. *The Architecture of Cognition: Rethinking Fodor and Pylyshyn's Systematicity Challenge*. MIT Press.

Michael B Chang, Abhishek Gupta, Sergey Levine, and Thomas L Griffiths. 2018. Automatically composing representation transformations as a means for generalization. *arXiv preprint arXiv:1807.04640*.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. 2019. Efficient lifelong learning with a-gem. In *ICLR*.

Noam Chomsky. 1957. *Syntactic structures*. Walter de Gruyter.

Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. 2018. Embodied question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 2054–2063.

Jerry A Fodor and Ernest Lepore. 2002. *The compositionality papers*. Oxford University Press.

Jerry A Fodor and Zenon W Pylyshyn. 1988. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71.

K. He, X. Zhang, S. Ren, and J. Sun. 2016. Deep residual learning for image recognition. In *CVPR*.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3128–3137.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Markus Kliegl and Wei Xu. 2018. More systematic than claimed: Insights on the scan tasks. *OpenReview*.

A. Krizhevsky, I. Sutskever, and G.E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*.

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning*, pages 2879–2888.

Brenden M Lake, Tal Linzen, and Marco Baroni. 2019. Human few-shot learning of compositional instructions. *arXiv preprint arXiv:1901.04587*.

Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. 2017. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature*, 521(7553):436.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of lstms to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Adam Liška, Germán Kruszewski, and Marco Baroni. 2018. Memorize or generalize? searching for a compositional rnn in a haystack. *arXiv preprint arXiv:1802.06467*.

Joao Loula, Marco Baroni, and Brenden M Lake. 2018a. Rearranging the familiar: Testing compositional generalization in recurrent networks. *arXiv preprint arXiv:1807.07545*.

Joao Loula, Marco Baroni, Brenden M, and Linzen. 2018b. Rearrange the familiar: testing compositional generalization in recurrent networks. *arXiv preprint arXiv:1807.07545*.

Gary F Marcus. 1998. Rethinking eliminative connectionism. *Cognitive psychology*, 37(3):243–282.

Gary F Marcus. 2003. *The algebraic mind: Integrating connectionism and cognitive science*. MIT press.

Marvin Minsky. 1986. *Society of mind*. Simon and Schuster.

Richard Montague. 1970. Universal grammar. *Theoria*, 36(3):373–398.

Paul Rodriguez and Janet Wiles. 1998. Recurrent neural networks can learn to implement symbol-sensitive counting. In *Advances in Neural Information Processing Systems*, pages 87–93.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Gail Weiss, Yoav Goldberg, and Eran Yahav. 2018. On the practical computational power of finite precision rnns for language recognition. *arXiv preprint arXiv:1805.04908*.

Francis CK Wong and William SY Wang. 2007. Generalisation towards combinatorial productivity in language acquisition by simple recurrent networks. In *2007 International Conference on Integration of Knowledge Intensive Multi-Agent Systems*, pages 139–144. IEEE.

Y. Wu and et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. In *arXiv:1609.08144*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057.

Guangyu Robert Yang, Madhura R Joglekar, H Francis Song, William T Newsome, and Xiao-Jing Wang. 2019. Task representations in neural networks trained to perform many cognitive tasks. *Nature neuroscience*, page 1.

D. Yu and L. Deng. 2012. *Automatic Speech Recognition*. Springer.

Friedemann Zenke, Ben Poole, and Surya Ganguli. 2017. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3987–3995. JMLR. org.

# WikiCREM: A Large Unsupervised Corpus for Coreference Resolution

**Vid Kocijan**[1] **Oana-Maria Camburu**[1,2] **Ana-Maria Creţu**[3] **Yordan Yordanov**[1]
**Phil Blunsom**[1,4] **Thomas Lukasiewicz**[1,2]
[1]University of Oxford, UK [2]Alan Turing Institute, London, UK
[3]Imperial College London, UK [4]DeepMind, London, UK
firstname.lastname@cs.ox.ac.uk, a.cretu@imperial.ac.uk

## Abstract

Pronoun resolution is a major area of natural language understanding. However, large-scale training sets are still scarce, since manually labelling data is costly. In this work, we introduce WIKICREM (Wikipedia CoREferences Masked) a large-scale, yet accurate dataset of pronoun disambiguation instances. We use a language-model-based approach for pronoun resolution in combination with our WIKICREM dataset. We compare a series of models on a collection of diverse and challenging coreference resolution problems, where we match or outperform previous state-of-the-art approaches on 6 out of 7 datasets, such as GAP, DPR, WNLI, PDP, WINOBIAS, and WINOGENDER. We release our model to be used off-the-shelf for solving pronoun disambiguation.

## 1 Introduction

Pronoun resolution, also called coreference or anaphora resolution, is a natural language processing (NLP) task, which aims to link the pronouns with their referents. This task is of crucial importance in various other NLP tasks, such as information extraction (Nakayama, 2019) and machine translation (Guillou, 2012). Due to its importance, pronoun resolution has seen a series of different approaches, such as rule-based systems (Lee et al., 2013) and end-to-end-trained neural models (Lee et al., 2017; Liu et al., 2019). However, the recently released dataset GAP (Webster et al., 2018) shows that most of these solutions perform worse than naïve baselines when the answer cannot be deduced from the syntax. Addressing this drawback is difficult, partially due to the lack of large-scale challenging datasets needed to train the data-hungry neural models.

As observed by Trinh and Le (2018), language models are a natural approach to pronoun resolution, by selecting the replacement for a pronoun that forms the sentence with highest probability. Additionally, language models have the advantage of being pre-trained on a large collection of unstructured text and then fine-tuned on a specific task using much less training data. This procedure has obtained state-of-the-art results on a series of natural language understanding tasks (Devlin et al., 2018).

In this work, we address the lack of large training sets for pronoun disambiguation by introducing a large dataset that can be easily extended. To generate this dataset, we find passages of text where a personal name appears at least twice and mask one of its non-first occurrences. To make the disambiguation task more challenging, we also ensure that at least one other distinct personal name is present in the text in a position before the masked occurrence. We instantiate our method on English Wikipedia and generate the Wikipedia CoREferences Masked (WIKICREM) dataset with 2.4M examples, which we make publicly available for further usage[1]. We show its value by using it to fine-tune the BERT language model (Devlin et al., 2018) for pronoun resolution.

To show the usefulness of our dataset, we train several models that cover three real-world scenarios: (1) when the target data distribution is completely unknown, (2) when training data from the target distribution is available, and (3) the transductive scenario, where the unlabeled test data is available at the training time. We show that fine-tuning BERT with WIKICREM consistently improves the model in each of the three scenarios, when evaluated on a collection of 7 datasets. For example, we outperform the state-of-

[1]The code can be found at https://github.com/vid-koci/bert-commonsense.
The dataset and the models can be obtained from https://ora.ox.ac.uk/objects/uuid:c83e94bb-7584-41a1-aef9-85b0e764d9e3



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4303–4312,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

the-art approaches on GAP (Webster et al., 2018), DPR (Rahman and Ng, 2012), and PDP (Davis et al., 2017) by 5.9%, 8.4%, and 12.7%, respectively. Additionally, models trained with WIKICREM show increased performance and reduced bias on the gender diagnostic datasets WINOGENDER (Rudinger et al., 2018) and WINOBIAS (Zhao et al., 2018).

## 2 Related Work

There are several large and commonly used benchmarks for coreference resolution, such as (Pradhan et al., 2012; Schäfer et al., 2012; Ghaddar and Langlais, 2016). However, Webster et al. (2018) argue that a high performance on these datasets does not correlate with a high accuracy in practice, because examples where the answer cannot be deduced from the syntax (we refer to them as *hard pronoun resolution*) are underrepresented. Therefore, several hard pronoun resolution datasets have been introduced (Webster et al., 2018; Rahman and Ng, 2012; Rudinger et al., 2018; Davis et al., 2017; Zhao et al., 2018; Emami et al., 2019). However, they are all relatively small, often created only as a test set.

Therefore, most of the pronoun resolution models that address hard pronoun resolution rely on little (Liu et al., 2019) or no training data, via unsupervised pre-training (Trinh and Le, 2018; Radford et al., 2019). Another approach involves using external knowledge bases (Emami et al., 2018; Fähndrich et al., 2018), however, the accuracy of these models still lags behind that of the aforementioned pre-trained models.

A similar approach to ours for unsupervised data generation and language-model-based evaluation has been recently presented in our previous work (Kocijan et al., 2019). We generated MASKEDWIKI, a large unsupervised dataset created by searching for repeated occurrences of nouns. However, training on MASKEDWIKI on its own is not always enough and sometimes makes a difference only in combination with additional training on the DPR dataset (called WSCR) (Rahman and Ng, 2012). In contrast, WIKICREM brings a much more consistent improvement over a wider range of datasets, strongly improving models' performance even when they are not fine-tuned on additional data. As opposed to our previous work (Kocijan et al., 2019), we evaluate models on a larger collection of test sets, showing the usefulness of WIKICREM beyond the Winograd Schema Challenge.

Moreover, a manual comparison of WIKICREM and MASKEDWIKI (Kocijan et al., 2019) shows a significant difference in the quality of the examples. We annotated 100 random examples from MASKEDWIKI and WIKICREM. In MASKEDWIKI, we looked for examples where masked nouns can be replaced with a pronoun, and only in 7 examples, we obtained a natural-sounding and grammatically correct sentence. In contrast, we estimated that 63% of the annotated examples in WIKICREM form a natural-sounding sentence when the appropriate pronoun is inserted, showing that WIKICREM consists of examples that are much closer to the target data. We highlight that pronouns are not actually inserted into the sentences and thus none of the examples sound unnatural. This analysis was performed to show that WIKICREM consists of examples with data distribution closer to the target tasks than MASKEDWIKI.

## 3 The WIKICREM Dataset

In this section, we describe how we obtained WIKICREM. Starting from English Wikipedia[2], we search for sentences and pairs of sentences with the following properties: at least two distinct personal names appear in the text, and one of them is repeated. We do not use pieces of text with more than two sentences to collect concise examples only. Personal names in the text are called "candidates". One non-first occurrence of the repeated candidate is masked, and the goal is to predict the masked name, given the correct and one incorrect candidate. In case of more than one incorrect candidate in the sentence, several datapoints are constructed, one for each incorrect candidate.

We ensure that the alternative candidate appears before the masked-out name in the text, in order to avoid trivial examples. Thus, the example is retained in the dataset if:

(a) the repeated name appears after both candidates, all in a single sentence; or

(b) both candidates appear in a single sentence, and the repeated name appears in a sentence directly following.

Examples where one of the candidates appears in the same sentence as the repeated name, while the

[2] https://dumps.wikimedia.org/enwiki/ dump id: enwiki-20181201

other candidate does not, are discarded, as they are often too trivial.

We illustrate the procedure with the following example:

*When asked about **Adams**' report, **Powell** found many of the statements to be inaccurate, including a claim that **Adams** first surveyed an area that was surveyed in 1857 by Joseph C.*

The second occurrence of "Adams" is masked. The goal is to determine which of the two candidates ("Adams","Powell") has been masked out. The masking process resembles replacing a name with a pronoun, but the pronoun is not inserted to keep the process fully unsupervised and error-free.

We used the Spacy Named Entity Recognition library[3] to find the occurrences of names in the text. The resulting dataset consists of $2,438,897$ samples. $10,000$ examples are held out to serve as the validation set. Two examples from our dataset can be found on Figure 1.

*Gina arrives and she is furious with Denise for not protecting Jody from Kingsley, as* `[MASK]` *was meant to be the parent.*
**Candidates:** Gina, **Denise**

*When Ashley falls pregnant with Victor's child, Nikki is diagnosed with cancer, causing Victor to leave* `[MASK]`*, who secretly has an abortion.*
**Candidates: Ashley**, Nikki

Figure 1: WIKICREM examples. Correct answers are given in bold.

We note that our dataset contains hard examples. To resolve the first example, one needs to understand that Denise was assigned a task and "meant to be the parent" thus refers to her. To resolve the second example, one needs to understand that having an abortion can only happen if one falls pregnant first. Since both candidates have feminine names, the answer cannot be deduced just on the common co-occurrence of female names and the word "abortion".

We highlight that our example generation method, while having the advantage of being unsupervised, also does not give incorrect signals, since we know the ground truth reference.

Even though WIKICREM and GAP both use text from English Wikipedia, they produce differing examples, because their generating processes differ. In GAP, passages with pronouns are collected and the pronouns are manually annotated, while WIKICREM is generated by masking names that appear in the text. Even if the same text is used, different masking process will result in different inputs and outputs, making the examples different under the transductive hypothesis.

**WIKICREM statistics.** We analyze our dataset for gender bias. We use the Gender guesser library[4] to determine the gender of the candidates. To mimic the analysis of pronoun genders performed in the related works (Webster et al., 2018; Rudinger et al., 2018; Zhao et al., 2018), we observe the gender of the correct candidates only. There were $0.8$M "male" or "mostly_male" names and $0.42$M "female" or "mostly_female" names, the rest were classified as "unknown". The ratio between female and male candidates is thus estimated around $0.53$ in favour of male candidates. We will see that this gender imbalance does not have any negative impact on bias, as shown in Section 6.2.

However, our unsupervised generating procedure sometimes yields examples where the correct answer cannot be deduced given the available information, we refer to these as *unsolvable examples*. To estimate the percentage of unsolvable examples, we manually annotated $100$ randomly selected examples from the WIKICREM dataset. In order to prevent guessing, the candidates were not visible to the annotators. For each example, we asked them to state whether it was solvable or not, and to answer the solvable examples. In $100$ examples, we found $18$ unsolvable examples and achieved $95.1\%$ accuracy on the rest, showing that the annotation error rate is tolerable. These annotations can be found in Appendix A.

However, as shown in Section 6.2, training on WIKICREM alone does not match the performance of training on the data from the target distribution. The data distribution of WIKICREM differs from the data distribution of the datasets for evaluation. If we replace the `[MASK]` token with a pronoun instead of the correct candidate, the resulting sentence sometimes sounds unnatural and would not occur in a human-written text. On the annotated $100$ examples, we estimated the percentage of natural-sounding sentences to be $63\%$. While the these sentences are not incorrect, the

[3] https://spacy.io/usage/linguistic-features#named-entities

[4] https://pypi.org/project/gender-guesser/

distribution of the training data differ from the target data.

## 4 Model

We use a simple language-model-based approach to anaphora resolution to show the value of the introduced dataset. In this section, we first introduce BERT (Devlin et al., 2018), a language model that we use throughout this work. In the second part, we describe the utilization of BERT and the fine-tuning procedures employed.

### 4.1 BERT

The Bidirectional Encoder Representations from Transformers (BERT) language model is based on the transformer architecture (Vaswani et al., 2017). We choose this model due to its strong language-modeling abilities and high performance on several NLU tasks (Devlin et al., 2018).

BERT is initially trained on two tasks: next sentence prediction and masked token prediction. In the next sentence prediction task, the model is given two sentences and is asked to predict whether the second sentence follows the first one. In the masked token prediction task, the model is given text with approximately 15% of the input tokens masked, and it is asked to predict these tokens. The details of the pre-training procedure can be found in Devlin et al. (2018).

In this work, we only focus on the masked token prediction. We use the PyTorch implementation of BERT[5] and the pre-trained weights for BERT-large released by Devlin et al. (2018).

### 4.2 Pronoun Resolution with BERT

This section introduces the procedure for pronoun resolution used throughout this work. Let $S$ be the sentence with a pronoun that has to be resolved. Let $\mathbf{a}$ be a candidate for pronoun resolution. The pronoun in $S$ is replaced with a `[MASK]` token and used as the input to the model to compute the log-probability $\log \mathbb{P}(\mathbf{a}|S)$. If $\mathbf{a}$ consists of more than one token, the same number of `[MASK]` tokens is inserted into $S$, and the log-probability $\log \mathbb{P}(\mathbf{a}|S)$ is computed as the average of log-probabilities of all tokens in $\mathbf{a}$.

The candidate-finding procedures are dataset-specific and are described in Section 6. Given a sentence $S$ and several candidates $\mathbf{a}_1, \ldots, \mathbf{a}_n$, we select the candidate $\mathbf{a}_i$ with the largest $\log \mathbb{P}(\mathbf{a}_i|S)$.

### 4.3 Training

When training the model, the setup is similar to testing. We are given a sentence with a name or a pronoun masked out, together with two candidates. The goal is to determine which of the candidates is a better fit. Let $\mathbf{a}$ be the correct candidate, and $\mathbf{b}$ be an incorrect candidate. Following our previous work (Kocijan et al., 2019) we minimize the negative log-likelihood of the correct candidate, while additionally imposing a max-margin between the log-likelihood of the correct and incorrect terms. We observe that this combined loss consistently yields better results on validation sets of all experiments than negative log-likelihood or max-margin loss on their own.

$$\mathcal{L} = -\log \mathbb{P}(\mathbf{a}|S) + \\ +\alpha \cdot \max(0, \log \mathbb{P}(\mathbf{b}|S) - \log \mathbb{P}(\mathbf{a}|S) + \beta), \quad (1)$$

where $\alpha$ and $\beta$ are hyperparameters controlling the influence of the max-margin loss term and the margin between the log-likelihood of the correct and incorrect candidates, respectively.

The hyperparameter settings for fine-tuning BERT on WIKICREM were the same as by Devlin et al. (2018), except for the learning rate and introduced constants $\alpha$ and $\beta$. For our hyperparameter search, we used learning rate $lr \in \{3 \cdot 10^{-5}, 1 \cdot 10^{-5}, 5 \cdot 10^{-6}, 3 \cdot 10^{-6}\}$ and hyperparameters $\alpha \in \{5, 10, 20\}$, $\beta \in \{0.1, 0.2, 0.4\}$ with grid search. The hyperparameter search is performed on a subset of WIKICREM with $10^5$ datapoints to reduce the searching time. We compare the influence of hyperparameters on the validation set of WIKICREM dataset. The best validation score was achieved with $lr = 1 \cdot 10^{-5}$, $\alpha = 10$, and $\beta = 0.2$. We used batches of size 64.

Since WIKICREM is large and one epoch takes around two days even when parallelized on 8 Tesla P100 GPUs, we only fine-tune BERT on WIKICREM for a single epoch. We note that better results may be achieved with further fine-tuning and improved hyperparameter search.

Fine-tuning on other datasets is performed in the same way as training except for two differences. Firstly, in fine-tuning, the model is trained for 30 epochs due to the smaller size of datasets. Secondly, we do not sub-sample the training set for hyperparameter search. We validate the model

[5] https://github.com/huggingface/pytorch-pretrained-BERT

after every epoch, retaining the model that performs best on the WIKICREM validation set.

## 5 Evaluation Datasets

We now introduce the 7 datasets that were used to evaluate the models. We decide not to use the CONLL2012 and WINOCOREF (Pradhan et al., 2012; Peng et al., 2015) datasets, because they contain more general coreference examples than just pronouns. We did not evaluate on the KNOWREF dataset (Emami et al., 2019), since it was not yet publicly available at the time of writing.

**GAP.** GAP (Webster et al., 2018) is a collection of $4,454$ passages from Wikipedia containing ambiguous pronouns. It focuses on the resolution of personal pronouns referring to human names and has a $1:1$ ratio between masculine and feminine pronouns. In addition to the overall performance on the dataset, each model is evaluated also on its performance on the masculine subset ($F_1^M$), feminine subset ($F_1^F$), and its gender bias ($\frac{F_1^F}{F_1^M}$). The best performance was exhibited by the Referential Reader (Liu et al., 2019), a GRU-based model with additional external memory cells.

For each example, two candidates are given with the goal of determining whether they are the referent. In approximately $10\%$ of the training examples, none of the candidates are correct. When training on the GAP dataset, we discard such examples from the training set. We do not discard any examples from the validation or test set.

When testing the model, we use the Spacy NER library to find all candidates in the sentence. Since the GAP dataset mainly contains examples with human names, we only retain named entities with the tag PERSON. We observe that in $18.5\%$ of the test samples, the Spacy NER library fails to extract the candidate in question, making the answer for that candidate "FALSE" by default, putting our models at disadvantage. Because of this, $7.25\%$ of answers are always false negatives, and $11.25\%$ are always true negatives, regardless of the model. Taking this into account, we compute that the maximal $F_1$-score achievable by our models is capped at $91.1\%$.

We highlight that, when evaluating our models, we are stricter than previous approaches (Liu et al., 2019; Webster et al., 2018). While they count the answer as "correct" if the model returns a substring of the correct answer, we only accept the full answer. The aforementioned models return the exact location of the correct candidate in the input sentence, while our approach does not. This strictness is necessary, because a substring of a correct answer could be a substring of several answers at once, making it ambiguous.

**WSC.** The Winograd Schema Challenge (Levesque et al., 2011) is a hard pronoun resolution challenge inspired by the example from Winograd (1972):

*The city councilmen refused the demonstrators a permit because* ***they*** *[feared/advocated] violence.*
**Question:** Who [feared/advocated] violence?
**Answer:** the city councilmen / the demonstrators

A change of a single word in the sentence changes the referent of the pronoun, making it very hard to resolve. An example of a Winograd Schema must meet the following criteria (Levesque et al., 2011):

1. Two entities appear in the text.
2. A pronoun or a possessive adjective appears in the sentence and refers to one of the entities. It would be grammatically correct if it referred to the other entity.
3. The goal is to find the referent of the pronoun or possessive adjective.
4. The text contains a "special word". When switched for the "alternative word", the sentence remains grammatically correct, but the referent of the pronoun changes.

The Winograd Schema Challenge is specifically made up from challenging examples that require commonsense reasoning for resolution and should not be solvable with statistical analysis of co-occurence and association.

We evaluate the models on the collection of 273 problems used for the 2016 Winograd Schema Challenge (Davis et al., 2017), also known as WSC273. The best known approach to this problem uses the BERT language model, fine-tuned on the DPR dataset (Kocijan et al., 2019).

**DPR.** The Definite Pronoun Resolution (DPR) corpus (Rahman and Ng, 2012) is a collection of problems that resemble the Winograd Schema Challenge. The criteria for this dataset have been relaxed, and it contains examples that might not require commonsense reasoning or examples where the "special word" is actually a whole phrase. We remove 6 examples in the DPR training set that overlap with the WSC dataset. The

dataset was constructed manually and consists of 1316 training and 564 test samples after we removed the overlapping examples. The best result on the dataset was reported by Peng et al. (2015) using external knowledge sources and integer linear programming.

**PDP.** The Pronoun Disambiguation Problem (PDP) is a small collection of 60 problems that was used as the first round of the Winograd Schema Challenge in 2016 (Davis et al., 2017). Unlike WSC, the examples do not contain a "special word", however, they do require commonsense reasoning to be answered. The examples were manually collected from books. Despite its small size, there have been several attempts at solving this challenge (Fähndrich et al., 2018; Trinh and Le, 2018), the best result being held by the Marker Passing algorithm (Fähndrich et al., 2018).

**WNLI.** The Winograd Natural Language Inference (WNLI) is an inference task inspired by the Winograd Schema Challenge and is one of the 9 tasks on the GLUE benchmark (Wang et al., 2019). WNLI examples are obtained by rephrasing Winograd Schemas. The Winograd Schema is given as the "premise". A "hypothesis" is constructed by repeating the part of the premise with the pronoun and replacing the pronoun with one of the candidates. The goal is to classify whether the hypothesis follows from the premise.

A WNLI example obtained by rephrasing one of the WSC examples looks like this:

**Premise:** *The city councilmen refused the demonstrators a permit because they feared violence.*
**Hypothesis:** *The demonstrators feared violence.*
**Answer:** true / **false**

The WNLI dataset is constructed manually. Since the WNLI training and validation sets overlap with WSC, we use the WNLI test set only. The test set of WNLI comes from a separate source and does not overlap with any other dataset.

The currently best approach transforms examples back into the Winograd Schemas and solves them as a coreference problem (Kocijan et al., 2019). Following our previous work (Kocijan et al., 2019), we reverse the process of example generation in the same way. We automatically detect which part of the premise has been copied to construct the hypothesis. This locates the pronoun that has to be resolved, and the candidate in question. All other nouns in the premise are treated as alternative candidates. We find nouns in the premise with the Stanford POS tagger (Manning et al., 2014).

**WINOGENDER.** WINOGENDER (Rudinger et al., 2018) is a dataset that follows the WSC format and is aimed to measure gender bias. One of the candidates is always an occupation, while the other is a participant, both selected to be gender neutral. Examples intentionally contain occupations with strong imbalance in the gender ratio. Participant can be replaced with the neutral "someone", and three different pronouns (he/she/they) can be used. The aim of this dataset is to measure how the change of the pronoun gender affects the accuracy of the model.

Our models mask the pronoun and are thus not affected by the pronoun gender. They exhibit no bias on this dataset by design. We mainly use this dataset to measure the accuracy of different models on the entire dataset. According to Rudinger et al. (2018), the best performance is exhibited by Durrett and Klein (2013) when used on the male subset of the dataset. We use this result as the baseline.

**WINOBIAS.** Similarly to the WINOGENDER dataset, WINOBIAS (Zhao et al., 2018) is a WSC-inspired dataset that measures gender bias in the coreference resolution algorithms. Similarly to WINOGENDER, it contains instances of occupations with high gender imbalance. It contains 3,160 examples of Winograd Schemas, equally split into validation and test set. The test set examples are split into 2 types, where examples of type 1 are "harder" and should not be solvable using the analysis of co-occurrence, and examples of type 2 are easier. Additionally, each of these subsets is split into anti-stereotypical and pro-stereotypical subsets, depending on whether the gender of the pronoun matches the most common gender in the occupation. The difference in performance between pro- and anti-stereotypical examples shows how biased the model is. The best performance is exhibited by Lee et al. (2017) and Durrett and Klein (2013), as reported by Zhao et al. (2018).

## 6 Evaluation

We quantify the impact of WIKICREM on the introduced datasets.

### 6.1 Experiments

We train several different models to evaluate the contribution of the WIKICREM dataset in different real-world scenarios. In **Scenario A**, no information of the target distribution is available. In **Scenario B**, the distribution of the target data is known and a sample of training data from the target distribution is available. Finally, **Scenario C** is the transductive scenario where the unlabeled test samples are known in advance. All evaluations on the GAP test-set are considered to be Scenario C, because BERT has been pre-trained on the English Wikipedia and has thus seen the text in the GAP dataset at the pre-training time.

We describe the evaluated models below.

**BERT.** This model, pretrained by Devlin et al. (2018), is the starting point for all models and serves as the soft baseline for Scenario A.

**BERT_WIKIRAND.** This model serves as an additional baseline for Scenario A and aims to eliminate external factors that might have worked against the performance of BERT. To eliminate the effect of sentence lengths, loss function, and the percentage of masked tokens during the training time, we generate the RANDOMWIKI dataset. It consists of random passages from Wikipedia and has the same sentence-length distribution and number of datapoints as WIKICREM. However, the masked-out word from the sentence is selected randomly, while the alternative candidate is selected randomly from the vocabulary. BERT is then trained on this dataset in the same way as BERT_WIKICREM, as described in Section 4.3.

**BERT_WIKICREM.** BERT, additionally trained on WIKICREM. Its evaluation on non-GAP datasets serves as the evaluation of WIKICREM under Scenario A.

**BERT_DPR.** BERT, fine-tuned on DPR. We hold out 10% of the DPR train set (131 examples) to use them as the validation set. All datasets, other than GAP, were inspired by the Winograd Schema Challenge and come from a similar distribution. We use this model as the baseline for Scenario B.

**BERT_WIKICREM_DPR.** This model is obtained by fine-tuning BERT_WIKICREM on DPR using the same split as for BERT_DPR. It serves as the evaluation of WIKICREM under Scenario B.

**BERT_GAP_DPR.** This model serves as an additional comparison to the BERT_WIKICREM_DPR model. It is obtained by fine-tuning BERT_GAP on the DPR dataset.

**BERT_GAP.** This model is obtained by fine-tuning BERT on the GAP dataset. It serves as the baseline for Scenario C, as explained at the beginning of Section 6.1.

**BERT_WIKICREM_GAP.** This model serves as the evaluation of WIKICREM for Scenario C and is obtained by fine-tuning BERT_WIKICREM on GAP.

**BERT_ALL.** This model is obtained by fine-tuning BERT on all the available data from the target datasets at once. Combined GAP-train and DPR-train data are used for training. The model is validated on the GAP-validation set and the WINOBIAS-validation set separately. Scores on both sets are then averaged to obtain the validation performance. Since both training sets and both validation sets have roughly the same size, both tasks are represented equally.

**BERT_WIKICREM_ALL.** This model is obtained in the same way as the BERT_ALL model, but starting from BERT_WIKICREM instead.

### 6.2 Results

The results of the evaluation of the models on the test sets are shown in Table 1. We notice that additional training on WIKICREM consistently improves the performance of the models in all scenarios and on most tests. Due to the small size of some test sets, some of the results are subject to deviation. This especially applies to PDP (60 test samples) and WNLI (145 test samples).

We observe that BERT_WIKIRAND generally performs worse than BERT, with GAP and PDP being notable exceptions. This shows that BERT is a strong baseline and that improved performance of BERT_WIKICREM is not a consequence of training on shorter sentences or with different loss function. BERT_WIKICREM consistently outperforms both baselines on all tests, showing that WIKICREM can be used as a standalone dataset.

We observe that training on the data from the target distribution improves the performance the most. Models trained on GAP-train usually show more than a 20% increase in their $F_1$-score on GAP-test. Still, BERT_WIKICREM_GAP shows

| | Transductive scenario | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | GAP $F_1$ | GAP $F_1^F$ | GAP $F_1^M$ | Bias $\frac{F_1^F}{F_1^M}$ | DPR | WSC | WNLI | |
| SOTA | 72.1% | 71.4% | 72.8% | 0.98 | 76.4% | **72.5%** | **74.7%** | |
| BERT | 50.0% | 47.2% | 52.7% | 0.90 | 59.8% | 61.9% | 65.8% | no |
| BERT_WIKIRAND | 55.1% | 51.8% | 58.2% | 0.89 | 59.2% | 59.3% | 65.8% | train |
| BERT_WIKICREM | **59.0%** | **57.5%** | **60.5%** | **0.95** | **67.4%** | **63.4%** | **67.1%** | data |
| BERT_GAP | 75.2% | 75.1% | 75.3% | **1.00** | 66.8% | 63.0% | 68.5% | |
| BERT_WIKICREM_GAP | **77.4%** | **78.4%** | **76.4%** | 1.03 | **71.1%** | **64.1%** | **70.5%** | |
| BERT_DPR | 60.9% | 61.3% | 60.6% | 1.01 | **83.3%** | 67.0% | 71.9% | existing |
| BERT_GAP_DPR | **70.0%** | **70.4%** | **69.5%** | 1.01 | 79.4% | 65.6% | 72.6% | train |
| BERT_WIKICREM_DPR | 64.2% | 64.2% | 64.1% | **1.00** | 80.0% | **71.8%** | **74.7%** | data |
| BERT_ALL | 76.0% | 77.4% | 74.7% | **1.04** | 80.1% | **70.0%** | 74.0% | |
| BERT_WIKICREM_ALL | **78.0%** | **79.4%** | **76.7%** | **1.04** | **84.8%** | **70.0%** | **74.7%** | |

| | WB T1-a | WB T1-p | WB T2-a | WB T2-p | WINOGENDER | PDP | |
|---|---|---|---|---|---|---|---|
| SOTA | 60.6% | 74.9% | 78.0% | 88.6% | 50.9% | 74.0% | |
| BERT | 61.3% | 60.3% | 76.2% | 75.8% | 59.2% | 71.7% | no |
| BERT_WIKIRAND | 53.5% | 52.5% | 64.6% | 65.2% | 57.9% | 73.3% | train |
| BERT_WIKICREM | **65.2%** | **64.9%** | **95.7%** | **94.9%** | **66.7%** | **76.7%** | data |
| BERT_GAP | 64.6% | 63.8% | 88.1% | 87.9% | 67.5% | **85.0%** | |
| BERT_WIKICREM_GAP | **71.2%** | **70.5%** | **97.2%** | **98.2%** | **75.4%** | 83.3% | |
| BERT_DPR | **78.0%** | **78.2%** | 85.6% | 86.4% | 79.2% | 81.7% | existing |
| BERT_GAP_DPR | 77.8% | 76.5% | **89.6%** | **89.1%** | 75.8% | **86.7%** | train |
| BERT_WIKICREM_DPR | 76.0% | 76.3% | 81.3% | 80.3% | **82.1%** | 76.7% | data |
| BERT_ALL | **77.8%** | **77.2%** | 94.7% | 94.9% | **78.8%** | 81.7% | |
| BERT_WIKICREM_ALL | 76.8% | 75.8% | **98.7%** | **99.0%** | 76.7% | **86.7%** | |

Table 1: Evaluation of trained models on all test sets. GAP and WINOBIAS (abbreviated WB) are additionally split into subsets, as introduced in Section 5. Double lines in the table separate results from three different scenarios: when no training data is available, when additional training data exists, and the transductive scenario. The table is further split into sections separated with single horizontal lines. Each section contains a model that has been trained on WIKICREM and models that have not been. The best result in each section is in bold. The best overall result is underlined. Scores on GAP are measured as $F_1$-scores, while the performance on other datasets is given in accuracy. The source of each SOTA is listed in Section 5.

a consistent improvement over BERT_GAP on all subsets of the GAP test set. This confirms that WIKICREM works not just as a standalone dataset, but also as an additional pre-training in the transductive scenario.

Similarly, BERT_WIKICREM_DPR outperforms BERT_DPR on the majority of tasks, showing the applicability of WIKICREM to the scenario where additional training data is available. However, good results of BERT_GAP_DPR show that additional training on a manually constructed dataset, such as GAP, can yield similar results as additional training on WIKICREM. The reason behind this difference is the impact of the data distribution. GAP, DPR, and WIKICREM contain data that follows different distributions which strongly impacts the trained models. This can be seen when we fine-tune BERT_GAP on DPR to obtain BERT_GAP_DPR, as the model's performance on GAP-test drops by 8.2%. WIKICREM's data distribution strongly differs from the test sets' as described in Section 3.

However, the best results are achieved when all available data is combined, as shown by the models BERT_ALL and BERT_WIKICREM_ALL. BERT_WIKICREM_ALL achieves the highest performance on GAP, DPR, WNLI, and WINOBIAS among the models, and sets the new state-of-the-art result on GAP, DPR, and WINOBIAS. The new state-of-the-art result on the WINOGENDER dataset is achieved by the BERT_WIKICREM_DPR model, while BERT_WIKICREM_ALL and BERT_GAP_DPR set the new state-of-the-art on the PDP dataset.

## 7 Conclusions and Future Work

In this work, we introduced WIKICREM, a large dataset of training instances for pronoun resolution. We use our dataset to fine-tune the BERT language model. Our results match or outperform state-of-the-art models on 6 out of 7 evaluated datasets.

The employed data-generating procedure can be further applied to other large sources of text to generate more training sets for pronoun resolution. In addition, both variety and size of the generated datasets can be increased if we do not restrict ourselves to personal names. We hope that the community will make use of our released WIKICREM dataset to further improve the pronoun resolution task.

## Acknowledgments

This work was supported by the Alan Turing Institute under the UK EPSRC grant EP/N510129/1, by the EPSRC grant EP/R013667/1, by the EPSRC studentship OUCS/EPSRC-NPIF/VK/1123106, by the JP Morgan PhD Fellowship 2019-2020, and by an EPSRC Vacation Bursary. We also acknowledge the use of the EPSRC-funded Tier 2 facility JADE (EP/P020275/1).

## References

Ernest Davis, Leora Morgenstern, and Charles L. Ortiz. 2017. The first Winograd Schema Challenge at IJCAI-16. *AI Magazine*, 38(3):97–98.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *Computing Research Repository*, arXiv:1810.04805.

Greg Durrett and Dan Klein. 2013. Easy victories and uphill battles in coreference resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1971–1982, Seattle, Washington, USA. Association for Computational Linguistics.

Ali Emami, Noelia De La Cruz, Adam Trischler, Kaheer Suleman, and Jackie Chi Kit Cheung. 2018. A knowledge hunting framework for common sense reasoning. *Computing Research Repository*, arXiv:1810.01375.

Ali Emami, Paul Trichelair, Adam Trischler, Kaheer Suleman, Hannes Schulz, and Jackie Chi Kit Cheung. 2019. The KNOWREF Coreference Corpus: Removing gender and number cues for difficult pronominal anaphora resolution. In *ACL 2019*.

Johannes Fähndrich, Sabine Weber, and Hannes Kanthak. 2018. A marker passing approach to Winograd schemas. In *Semantic Technology*, pages 165–181, Cham. Springer International Publishing.

Abbas Ghaddar and Philippe Langlais. 2016. WikiCoref: An English coreference-annotated corpus of Wikipedia articles. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Portorož, Slovenia. European Language Resources Association (ELRA), European Language Resources Association (ELRA).

Liane Guillou. 2012. Improving pronoun translation for statistical machine translation. In *Proceedings of the Student Research Workshop at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1–10. Association for Computational Linguistics.

Vid Kocijan, Ana-Maria Cretu, Oana-Maria Camburu, Yordan Yordanov, and Thomas Lukasiewicz. 2019. A surprisingly robust trick for Winograd Schema Challenge. *Computing Research Repository*, arXiv:1905.06290.

Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2013. Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics*, 39(4):885–916.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197, Copenhagen, Denmark. Association for Computational Linguistics.

Hector J. Levesque, Ernest Davis, and Leora Morgenstern. 2011. The Winograd Schema Challenge. *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*, 46.

Fei Liu, Luke S. Zettlemoyer, and Jacob Eisenstein. 2019. The referential reader: A recurrent entity network for anaphora resolution. *Computing Research Repository*, abs/1902.01541.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Kotaro Nakayama. 2019. Wikipedia mining for triple extraction enhanced by co-reference resolution.

Haoruo Peng, Daniel Khashabi, and Dan Roth. 2015. Solving hard coreference problems. In *HLT-NAACL*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 Shared Task: Modeling multilingual unrestricted coreference in OntoNotes. In *Joint Conference on EMNLP and CoNLL - Shared Task*, CoNLL '12, pages 1–40, Stroudsburg, PA, USA. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Altaf Rahman and Vincent Ng. 2012. Resolving complex cases of definite pronouns: The Winograd Schema Challenge. In *Proceedings of EMNLP*.

Rachel Rudinger, Jason Naradowsky, Brian Leonard, and Benjamin Van Durme. 2018. Gender bias in coreference resolution. *Computing Research Repository*, abs/1804.09301.

Ulrich Schäfer, Christian Spurk, and Jörg Steffen. 2012. A fully coreference-annotated corpus of scholarly papers from the ACL Anthology. *Proceedings of COLING 2012: Posters*, pages 1059–1070.

T. H. Trinh and Q. V. Le. 2018. A Simple Method for Commonsense Reasoning. *Computing Research Repository*, arXiv:1806.02847.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Computing Research Repository*, arXiv:1706.03762.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In the Proceedings of ICLR.

Kellie Webster, Marta Recasens, Vera Axelrod, and Jason Baldridge. 2018. Mind the GAP: A balanced corpus of gendered ambiguous pronouns. In *Transactions of the ACL*.

Terry Winograd. 1972. *Understanding Natural Language*. Academic Press, Inc., Orlando, FL, USA.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2018. Gender bias in coreference resolution: Evaluation and debiasing methods. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 15–20, New Orleans, Louisiana. Association for Computational Linguistics.

# Identifying and Explaining Discriminative Attributes

**Armins Stepanjans** and **André Freitas**
Department of Computer Science
University of Manchester
armins.stepanjans@student.manchester.ac.uk
andre.freitas@manchester.ac.uk

## Abstract

Identifying what is at the center of the meaning of a word and what discriminates it from other words is a fundamental natural language inference task. This paper describes an explicit word vector representation model (WVM) to support the identification of discriminative attributes. A core contribution of the paper is a quantitative and qualitative comparative analysis of different types of data sources and Knowledge Bases in the construction of explainable and explicit WVMs: (i) *knowledge graphs built from dictionary definitions*, (ii) *entity-attribute-relationships graphs derived from images* and (iii) *commonsense knowledge graphs*. Using a detailed quantitative and qualitative analysis, we demonstrate that these data sources have complementary semantic aspects, supporting the creation of explicit semantic vector spaces. The explicit vector spaces are evaluated using the task of discriminative attribute identification, showing comparable performance to the state-of-the-art systems in the task (F1-score = 0.69), while delivering full model transparency and explainability.

## 1 Introduction

Word-vector/embedding models (WVM) have emerged as first-class representations in contemporary Natural Language Processing (NLP) tasks due to their ability to capture semantic similarity and relatedness in an unsupervised and comprehensive manner (Turney and Pantel, 2010), (Freitas, 2015). Additionally, the simplicity entailed by the vector space abstraction makes it an engineering-friendly representation, also explaining its widespread adoption and use (Freitas, 2015). However, the latent features (dense vectors) at the center of most of the best-performing models have limited their application to two main uses: (i) computing semantic similarity and relatedness measures and (ii) performing vocabulary generalization as an input layer on Machine Learning (ML) models.

The identification of discriminative attributes (IDA), recently introduced by Krebs et al. (2018), can motivate the development of word vector models which can support types of operations with finer semantics, going beyond the computation of semantic similarity and relatedness scores, with potential applications in fine-grained semantic inference tasks.

Concretely, using the example provided by (Krebs et al., 2018), given a pair of target terms *apple* and *banana*, the IDA task seeks to answer if the term *red* is a discriminative attribute for apple in comparison to banana. According to Krebs et al. *semantic difference* is a ternary relation between two concepts (*apple*, *banana*) and a discriminative feature (*red*) that characterizes the former concept but not the latter.

This paper focuses on proposing an explicit (sparse) WVM for detecting and explaining discriminative attributes. The proposed explicit WVM provides a dimension of *explainability* while keeping the simplicity of the vector space representation model. This addition allows the model to provide a *justification* while addressing tasks such as the computation of discriminative attributes (Figure 1). The explicit nature of the model and its ability to compute the difference between the term pairs is the core proposed contribution (instead of the improvement of the F1-score for the task). Models with the ability to compute discriminative attributes can provide a representation paradigm which can support more fine-grained semantic inference tasks.

Another core contribution of this paper is a comprehensive quantitative and qualitative comparative analysis on how different types of data sources and Knowledge Bases (KBs) affect the



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4313–4322,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

construction of explainable and explicit WVMs. The IDA task requires a granular representation of the necessary and sufficient conditions associated with the definition of a target concept, many of these transcending lexicographic and encyclopaedic representations. In order to address this problem, we analyse three types of data sources: (i) *knowledge graphs built from dictionary definitions*, (ii) *entity-attribute-relationship graphs derived from images* and (iii) *commonsense knowledge graphs*. The goal behind using these models is twofold: (i) to provide an explainable, fine-grained word-vector model and (ii) to support the capture of lexical-semantic relations not captured in a representative fashion in regular corpora used in existing distributional models (which in most cases combine news, encyclopedic and literary discourse).

In summary, this paper focuses on the following contributions: **(i)** *The creation of a novel interpretable word vector model based on the combination of structured definitions and visual features for the IDA task* and **(ii)** *a detailed comparative, quantitative and qualitative evaluation of the semantic contribution of different types of data sources for explicit semantics*.

## 2 Related Work

In this section we group related work into two major categories: (i) approaches for the identification of discriminative attributes (IDA) and (ii) definition-based word vector space models.

Existing approaches have explored combinations of linguistic and data resources (WordNet, ConceptNet, Wikipedia), linguistic features (syntactic dependencies), sparse word vector models (JoBimText), dense word-vector (DWV) models (W2V, GloVe) and supervised/ unsupervised machine learning approaches (SVM, MLP, Ensemble Methods).

With regard to interpretability and explainability we can classify IDA approaches into three categories. *Frequency-based models over text-based features*, heavily relying on textual features and frequency-based methods (Gamallo, 2018; González et al., 2018) ; *ML over Textual features* (Dumitru et al., 2018; Sommerauer et al., 2018; King et al., 2018; Mao et al., 2018) and *ML over dense vectors and textual features* (Brychcín et al., 2018; Attia et al., 2018; Wu et al., 2018; Dumitru et al., 2018; Arroyo-Fernández et al., 2018; Speer and Lowry-Duda, 2018; Santus et al., 2018; Grishin, 2018; Zhou et al., 2018; Vinayan et al., 2018; Kulmizev et al., 2018; Zhang and Carpuat, 2018; Shiue et al., 2018). While the first category concentrates on models with higher interpretability, none of these models provide explanations.

Comparatively, this work focuses on the creation of an explicit word vector space model (EWVM), with an associated explanation, evaluating the performance of different types of lexico-semantic resources in the context of the task of identification of discriminative attributes (IDA).

## 3 Identifying and Explaining Discriminative Attributes

### 3.1 Problem Definition

This paper provides an explainable word vector space model (EWVM) for detecting whether a given term is a discriminative attribute or not with regard to a pair of reference terms as defined by Krebs et al. (2018). Given a triple $< t_p, t_c, t_f >$, $t_f$ (*feature*) is considered discriminative if it is related to the first term $t_p$ (*pivot*) to a significantly higher extent than it is related to the second term $t_c$ (*comparison*), i.e. $t_f \in p(t_p) \wedge t_f \notin p(t_c)$, where the function $p(t)$ returns a set of properties associated with term $t$.

Additionally, beyond the identification of the discriminative attribute (i.e. assigning a value of true or false to the target term triple), this paper explores different notions of an explanation $e$ assigned to the inference, which can be both human and machine interpretable.

### 3.2 Types of Discriminative Attributes

The nature of the semantic relationships expressed in the task of identifying discriminative attributes can guide the selection of the supporting target corpora. In this section, we propose a classification of *term-discriminative attribute relationships* into three dual categories: **Essential vs. Incidental**, **Sensory vs. Logical** and **Relative vs. Absolute**.

**Essential vs. Incidental:** $p$ is an *essential* property of an object $o$ just in case it is necessary that $o$ has $p$, whereas $p$ is an *incidental* property of an object $o$ just in case $o$ has $p$ but it is possible that $o$ lacks $p$. Example: `(cognac, whiskey, french)` is *essential* and `(nose, throat, perfume)` is *incidental*. The notion of Essential v. Incidental is similar to the logical notion of

Necessary and Sufficient Conditions, but rooted in the philosophical notion of essential and accidental properties, particularly, as defined by Robertson and Atkins (2018).

**Sensory vs. Logical:** $p$ is a *sensory* property of an object $o$, in case $p$ can be identified as a property of $o$ exclusively through sensory information, whereas $p$ is a *logical* property of an object $o$ otherwise. Example: `(cheek, brow, red)` is *sensory* and `(grapes, wine, fruit)` is *logical*. The Sensory vs. Logical dual category is inspired by Hayes' Second Naïve Physics Manifesto, particularly, the distinction of 'three ways in which tokens can be attached to their denotations', where our notion of Sensory attribute corresponds to the second way ('some of the tokens can be attached to sensory and motor systems'), whereas our notion of Logical attribute corresponds to the remaining two ('tokens could be attached to the world through language' and 'token is a metatheory of some internal part of the theory') (Hayes, 1995).

**Relative vs. Absolute:** $p$ is a *relative* property of an object $o_a$ in relation to $o_b$ just in case there exists an object $o_c$ with the property $p$ where $p$ is not an attribute of $o_a$ in relation to $o_c$. Whereas $p$ is an *absolute* property of an object $o_a$ in case there does not exist an object $o_b$ with the property $p$ where $p$ is not an attribute of $o_a$ when compared to $o_b$. Example: `(giraffe, ostrich, tall)` is *relative* and `(bat, butterfly, fur)` is *absolute*. The Relative vs. Absolute dual category follows Hayes' idea of 'intrinsic qualities (absolute) versus the distance between such qualities (relative)' (Hayes, 1995).

The classification scheme will guide the creation of the discriminative word vector model described in the next section.

## 4 Building Explicit Word Vector Space Models (EWVM)

Based on the dual categories identified in the previous section, a composition of three word vector space models is used to define *explicit word vector spaces* (EWVM), using three different types of corpora:

**Definition-Based Model (DBM):** Consists of a dictionary-style definition corpus. In the context of this work, WordNet and Wiktionary *natural language definitions* are used as data sources.

**Visual Feature Model (VFM):** Built using lexical graph descriptors for images, containing entities, attributes and relations, grounded on image bounding boxes.

**Commonsense Knowledge Graph (CKG):** Consists on the use of lexico-semantic knowledge graphs such as ConceptNet.

The following sections describe the construction of the EWVM.

### 4.1 Definition-Based Models (DBMs)

DBMs are built out of natural language term definitions (glosses) found either in dictionaries or filtered out of larger corpora (for example, Wikipedia contains many definitional sentences which can be isolated using lexico-syntactic patterns). The intuition behind the use of natural language definitions is twofold: first they are succinct descriptions of the necessary attributes associated to a concept and secondly they are abundant across domains and languages as dictionaries or definitions embedded within discourse. The latter point makes DBMs potentially transportable across languages and domains.

#### 4.1.1 Model Construction

The representation behind the definition-based model is built by segmenting and categorizing natural language definitions into a set of semantic roles, a model which was introduced by (2016), (2018b), (2017). These roles aim to transform natural definitions into definition knowledge graphs, in order to facilitate natural language inference tasks (Silva et al., 2018b), (Silva et al., 2018a). The set of semantic roles includes: *supertype, differentia quality, differentia event, event location, purpose, accessory determiner, origin location*. Figure 1 (DBM) depicts an example of a classified definition.

The semantic roles are assigned by building a recurrent neural network (RNN) Definition Role Labeling (DRL) classifier using POS-tags, and pre-trained word vectors as features using the configuration of Silva et al. (2016). After the semantic roles are assigned, they are used as an input to build the supporting word vector space. For each definition segment, all tokens are lemmatized and stop-words removed. Afterwards, an inverse document frequency (idf) weighting scheme is applied, in order to support the computation of semantic similarity and relatedness scores within the model (despite not being the target use of the model). Ad-

ditionally, for each target term, we take into account its upward taxonomic chain, i.e. it inherits the definition attributes from the parent terms linked by the detected supertypes at the definition.

An inverted index is used to materialize the vector space. A workflow of the proposed model is depicted in Figure 1.

**Algorithm 1** EDAM: Identifying discriminative attributes

$D$: Set of definitions $d$, containing predicates $p^D$ and terms $t$
$I$: Set of images $i$, containing features with predicates $p^{[O,R,A]}$ and terms $t$
**Query:** $< t_p, t_c, t_a >$
**Output:** discriminative, explanation
**if** $(t_a, t_p) \in D \wedge (t_a, t_c) \notin D$ **then**
  discriminative $\leftarrow$ **true**
  explanation $\leftarrow$ template$(t_p, t_c, t_a, d)$
  **return**
**end if**
**if** $(t_a, t_p) \in I \wedge (t_a, t_c) \notin I$ **then**
  discriminative $\leftarrow$ **true**
  explanation $\leftarrow$ template$(t_p, t_c, t_a, i)$
  **return**
**end if**
discriminative $\leftarrow$ **false**

### 4.2 Visual Feature Model

Natural language definition corpora by design focus on *essential*, *logical* and *absolute* discriminative attribute types. However, discriminative attributes can also occur as *incidental*, *sensorial* or *relative* instances. Most distributional semantic models available today are built over journalistic, encyclopedic or narrative types of discourse. While incidental attributes can be captured as a second-order distributional phenomena, these corpora do not reflect explicit commonsense knowledge, in particular with regard to extra-linguistic phenomena.

Recent datasets introduced for the purpose of supporting image classification tasks, such as VisualGenome (Krishna et al., 2017) have provided multi-modal resources connecting sub-symbolic visual data types to symbolic-level categories. These datasets explore both modalities of visual and textual data.

VisualGenome is a dataset consisting of scenes (108,077 images) segmented into bounding boxes (5.4M) and annotated with a set of objects (3.8M), attributes (2.8M) and relationships (2.3M). Each image has an associated lexical-semantic model represented as a labeled graph. VisualGenome concentrates on the description of a large spectrum of commonsense scenes (not focusing on specific named entities). Common terms are: *man, person, tree, window, grass, table* (objects), *white, black, blueish, metallic, round* (attributes) and *along, inside, almost, above, ride* (relationships). Essentially, VisualGenome expresses facts about objects in commonsense scenes.

The commonsense nature of VisualGenome scenes and the *object-attribute* relations provide a foundation for covering visual attribute sets, potentially covering part of the *sensorial* attributes. Additionally, VisualGenome can be used to cover *incidental* attributes. VisualGenome can provide some level of support for identifying relative discriminative attributes, by focusing on features which are mediated by visual interpretation such as size (e.g. large, small). However, this work does not focus on a representation which can support the identification of relative attributes mediated by visual features.

#### 4.2.1 Model Construction

The Visual Feature Model (VFM) also uses a sparse explicit semantic vector space representation as its basis. The representation targets the identification of sensorial and incidental discriminative attributes, with a supporting explanatory model.

The model is based on the construction of two vector spaces: one for indexing objects and attributes, *object-attribute* (OA) space associated with bounding boxes and the other for indexing relationships, i.e. the *scene-object-relationship* (SOR) space. The OA space supports the identification of the set of attributes directly associated with objects while the SOR space supports the capture of association relations between objects (*objectA* can inherit an attribute from *objectB*).

### 4.3 Commonsense Knowledge Graph

Dictionary definitions have a limited ability to express incidental relations among concepts due to their conciseness. For example, relations which express affordances (e.g. 'can be used for') are not typically expressed in dictionaries. Labeled visual datasets are limited in their domain coverage. We use commonsense knowledge graphs as a third data source aiming to fill this gap. For ex-

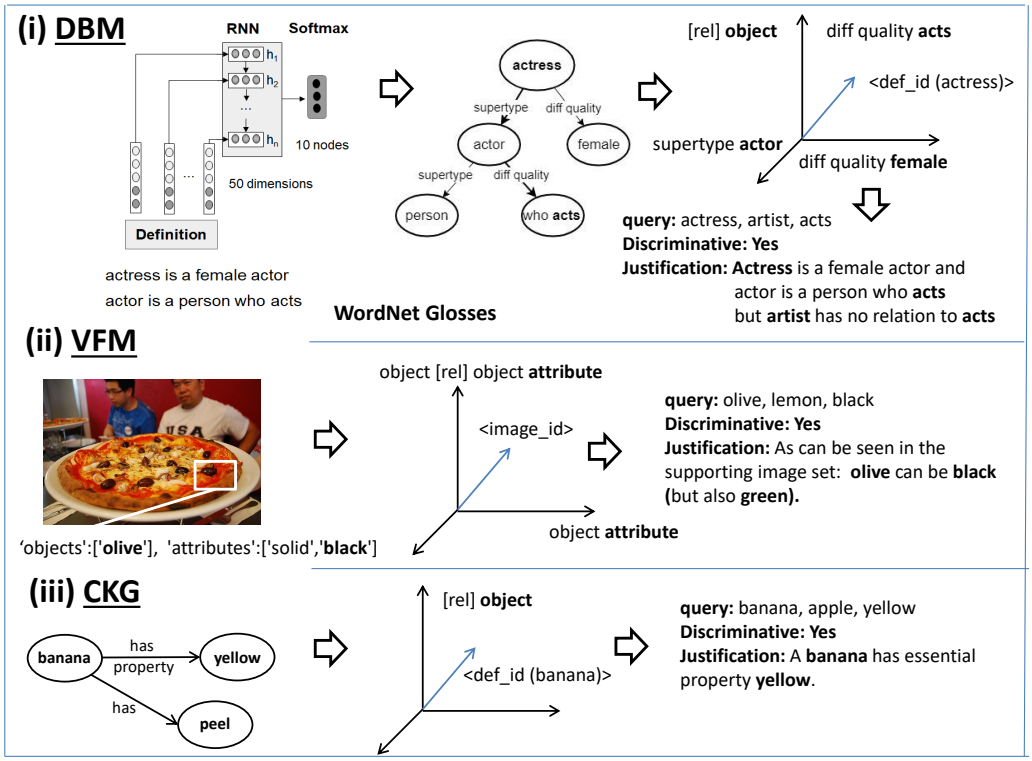


Figure 1: Sparse vector space representation and explanation of each model component.

ample Visual Genome contains 5.4 million region descriptions, whereas ConceptNet 5 contains 28 million relationships, captured using 39 relationship types.

#### 4.3.1 Model Construction

The Commonsense Knowledge Graph is mapped into a sparse explicit semantic vector space. The model targets essential and logical attributes. The model is constructed by indexing each relationship using an idf weighting scheme, while ignoring negation (prefixed by Not).

### 4.4 Combined Model: Explainable Discriminative Attribute Model (EDAM)

The goal of the EDAM model is twofold: to maximize the quality of the discriminative attribute identification (IDA) and to maximize the underlying interpretability of the model. The EDAM model combines all three models by composing them into an interpretability hierarchy.

Each component of the EDAM model provides a different type of explanation, described below:
**DBM/CKG**: Consists of an attribute path between the natural language definitions, using the structure defined by the graphs. It provides an explanatory model at the intensional level.

**VFM**: Consists of the description of the attribute-pairs incidence and the supporting ground image-set. It provides an explanatory model at an extensional level.

The explanations are generated using a template-based approach as described in Algorithm 1. Details on the construction of the models and on the formalization of the explanations can be found online[1].

## 5 Evaluation

The proposed EDAM model was evaluated using the Semeval2018 Task 10 gold-standard on identifying discriminative attributes (Krebs et al., 2018). The dataset consists of 2340 triples (pivot, comparison and attribute) terms, classified as either discriminative (if the attribute is a discriminative feature of pivot) or not (otherwise). The evaluation aimed at answering the following questions:
**Q1:** Can the combination of the Definition-Based Model (DBM), the Visual Feature Model (VFM), and the Commonsense Knowledge Graphs (CKG) support the task of discriminative attribute identification?
**Q2:** What is the contribution of each component

[1] https://github.com/ab-10/Hawk

for each type of attribute category?

In the experiment, DBM was instantiated using WordNet 3.0 (only glosses structured into its semantic roles), VFM used VisualGenome 1.4 and CKG used ConceptNet 5. Table 2 depicts the outcome of the evaluation of the combined model.

In order to answer Q1, we compared our model to a set of reference systems in the task of discriminative attribute identification (see table 1. While EDAM performs lower than the state-of-the-art (F1-Score 0.76 vs 0.69), from existing approaches EDAM provides the only explainable model. Compared to EDAM, all the top performing models used distributional methods derived from large-scale corpora, while EDAM uses the combination of definition, visual features and commonsense structured KBs.

In order to answer Q2, we evaluated the performance of each component of the model for the six dual-categories. The goal is to provide a quantitative basis to understand the contribution of each component in addressing the task.

In order to perform the evaluation, we selected a stratified random subset of 230 triples and manually classified each triple with the dual categories. A triple can contain one or more categories associated with it. The annotation was performed by two independent annotators, which reached an inter-annotator agreement of 81%. For the final annotated dataset, triples which were not consensual were eliminated (i.e. the final dataset has a 100% inter-annotator agreement). The F1-score of the combined EDAM model and of each component for each category are shown in Table 2.

The list below summarizes the best component performance for each category:

- **Sensory:** DBM significantly outperforms the CKG and VFM model (75% over the second best).
- **Logical:** CKG significantly outperforms the other components (outperforms the second best component by 52%).
- **Relative:** DBM significantly outperforms the other components (100% over the second best).
- **Absolute:** CKG outperforms the second best component by 7%.
- **Essential:** CKG significantly outperforms DBM and VFM (100% over the second best).
- **Incidental:** DBM outperforms VFM by 154% and CKG by 48%.

The analysis shows the complementary nature of the models, where *DBM* contributes more for the identification of the sensory and incidental categories, while *CKG* contributes to the logical and essential categories. Additionally, *VFM* provides contributions across all categories and has smaller overlaps with other models. It is important to emphasize that the models are complementary at each dimension: the combined model (EDAM) significantly outperforms the best individual models at each dimension (as it can be observed at the gain row on Table 2).

In order to observe the proportion on which each model contributes (and conversely, the amount of redundancy for each model), we analyze for each individual model, the pairwise overlap, in terms of true positives and false positives (Table 3). The average redundancy between the individual models is low (average 11%), showing that all models contribute significantly to the performance of the combined model.

Table 4 further breaks down the true positives for each dimension, where we can analyse the pairwise contribution of each model for each attribute category. *The analysis shows that the overall redundancy between the three models is low for all categories*. Excluding the relative category (in which all models perform poorly), there is little variance in the overlap between the pairwise models (where the intersection between DBM and VFM is the largest for each category).

The quantitative analysis shows that the main limitation of the model is in the identification of *relative attributes*. While DBMs are able to correctly identify a small set of triples with relative features such as `(skyscraper, apartment, tall)` as discriminative, interpretation of relative relations requires two types of features which are not targeted by the models, namely: *(i)* the extraction of precise numerical reference points at scale (dealing with variations of dimensional units) and *(ii)* the ability to extrapolate the relations for unobserved lexemes by an explicit mechanism of comparative/transitive reasoning. As a consequence, the current model is able to identify "skyscraper" as being taller than an "apartment", but fails to identify neither as taller than a "giraffe".

DBM forms a significant (49% each) contribu-

| Model | Approach | Explainability | F1 Score |
|---|---|---|---|
| (Lai et al., 2018) | SVM with GloVe | None | 0.76 |
| (Speer and Lowry-Duda, 2018) | SVM with ConceptNet, Wikipedia articles and WordNet synonyms | None | 0.74 |
| (Shiue et al., 2018) | MLP combining information from various DSMs, PMI, and ConceptNet | None | 0.73 |
| (Santus et al., 2018) | Gradient boosting with co-occurrence count features and *JoBimText* features | None | 0.73 |
| (Brychcín et al., 2018) | LexVec, word co-occurrence, and ConceptNet data combined using maximum entropy classifier | None | 0.72 |
| **Proposed Model (EDAM)** | Composes explicit vector spaces from WordNet Definitions, ConceptNet and Visual Genome | Fully Explainable | 0.69 |
| (Sommerauer et al., 2018) | Word2Vec cosine similarities of WordNet glosses | Transp. (No expl.) | 0.69 |
| (González et al., 2018) | Use of Wikipedia and ConceptNet | Transp. (No expl.) | 0.69 |
| (Attia et al., 2018) | Google 5 grams and Word2Vec embeddings as features for feedforward neural network | None | 0.67 |
| (Zhou et al., 2018) | Ensemble ML model with WordNet, PMI scores, Word2Vec, and GloVe embeddings | None | 0.67 |
| (Kulmizev et al., 2018) | A combination of GloVe and Paragram embeddings | None | 0.67 |
| (Zhang and Carpuat, 2018) | SVM with GloVe embeddings | None | 0.67 |
| (Vinayan et al., 2018) | CNN with GloVe embeddings | None | 0.66 |
| (Grishin, 2018) | Similarity calculations using a combination of DSMs | None | 0.65 |
| (Wu et al., 2018) | Word2Vec, GloVe, and FastText embeddings as features for MLP-CNN | None | 0.63 |
| (Gamallo, 2018) | Dependency parsing and co-occurrence analysis | Transp. (No expl.) | 0.63 |
| (Dumitru et al., 2018) | SVM with word co-occurrence | None | 0.63 |
| (Mao et al., 2018) | CBOW and skip-gram with WordNet definitions | Transp. (No expl.) | 0.62 |
| (King et al., 2018) | Word2Vec, WordNet, and co-occurrence scores | Transp. (No expl.) | 0.61 |
| (Arroyo-Fernández et al., 2018) | Convex Cone Method applied to GloVe embeddings | None | 0.60 |

Table 1: Performance of EDAM in contrast to baselines for discriminative attribute identification. *Transp.* means transparent covering different dimensions of interpretability (Silva et al., 2019) (but without an explanation).

| Model | Sensory | Logical | Relative | Absolute | Essential | Incidental |
|---|---|---|---|---|---|---|
| DBM | **0.49** | 0.33 | **0.40** | 0.41 | 0.29 | **0.46** |
| CKG | 0.28 | **0.50** | 0.20 | **0.44** | **0.58** | 0.31 |
| VFM | 0.13 | 0.11 | 0.05 | 0.14 | 0.10 | 0.13 |
| EDAM(DBM+CKG+VFM) | 0.62 | 0.63 | 0.50 | 0.65 | 0.68 | 0.60 |
| EDAM gain | 27% | 26% | 25% | 48% | 17% | 30% |

Table 2: Performance comparison (recall) against a random sample of categorized triples. The last row shows the relative gain of the combined model over the best performing individual model.

| Positives | DBM ∧ CKG | DBM ∧ VFM | CKG ∧ VFM | DBM ∧ CKG ∧ VFM | Avg. |
|---|---|---|---|---|---|
| True | 0.23 | 0.07 | 0.08 | 0.06 | 0.11 |
| False | 0.05 | 0.01 | 0.02 | 0.01 | 0.02 |

Table 3: Overlap between the components relative to the combined model.

| Category | DBM ∧ CKG | DBM ∧ VFM | CKG ∧ VFM | DBM ∧ CKG ∧ VFM | Average |
|---|---|---|---|---|---|
| Sensory | 0.31 | 0.10 | 0.14 | 0.10 | 0.16 |
| Logical | 0.35 | 0.09 | 0.15 | 0.09 | 0.17 |
| Relative | 0.20 | 0.10 | 0.10 | 0.10 | 0.13 |
| Absolute | 0.36 | 0.09 | 0.15 | 0.09 | 0.15 |
| Essential | 0.33 | 0.05 | 0.10 | 0.05 | 0.13 |
| Incidental | 0.33 | 0.12 | 0.17 | 0.12 | 0.19 |
| Avg. | 0.31 | 0.09 | 0.14 | 0.09 | |

Table 4: Categorical model overlap breakdown, relative to all true positives identified by the combined model.

tion to *sensory* attribute detection. CKG provides a significant (58%) contribution to *essential*. Additionally, CKG provides a significant (50%) contribution to *logical* attribute detection.

### 5.1 Error Analysis

**Definition Based Models:** False negatives comprise the majority (83%) of all model errors. The cause is that the pivot's definition does not include the discriminative attribute. False

negatives most commonly occurs with incidental features (e.g. `(potatoes, butter, mashed) - true:false` and `(nose, throat, perfume) - true:false`). False positives occur when the attribute applies to both pivot and comparison, however the comparison's definition does not include the feature. This is most prevalent with incidental attributes (e.g. `(banana,onions,peel) - false:true` and `(torah, bible, read) - false:true`).

**Common Sense Knowledge Graphs:** False negatives comprise the majority (80%) of the model's errors and are mainly caused by incidental attributes (e.g. `(trays,employee,wooden) - true:false`) and relative attributes (e.g. `(stool,tray,tall) - true:false`).

**Visual Feature Models:** Similarly as with DBMs and CKGs, false negatives comprise the majority (94%) of all false classifications. Most of the errors occur either with logical attributes (e.g. `(wife, lady, married) - true:false`, since these attributes are not likely to be expressed in a visual corpus or relative attributes (e.g. `(torah, bible, short) - true:false`. False positives occur due to domain incompleteness. This can be caused by a lack of domain coverage of the Visual Genome dataset (e.g. in `(cat, lion, whiskers) - false:true`; `(meal, supper, food) - false:true`).

**Combined EDAM:** *The decrease in the proportion of false negatives as compared to the individual models, illustrates the advantages of the model composition*. False negatives comprise 47% of model's total errors. False positives, on the contrary, illustrate the limitations of the model, since they occur when at least one of the model components incorrectly classifies a triple as discriminative: e.g. `(banana, onions, peel) - false:true` false positive of DBM and `(cat, lion, whiskers) - false:true` false positive of VFM.

### 5.2 Qualitative Analysis

In addition to correctly labeling common discriminative triples such as `(soup, meal, liquid)` and `(walnut, spinach, brown)` the technical specificity of definitions present at dictionaries supports the identification of discriminative triples such as `(brandy, whiskey, wine)`. Other noteworthy examples of discriminative attribute detection include labeling triples `(stomach, bladder, food)` and `(nightclub, bar, dancing)` as discriminative.

For the DBM, the structure induced by the extractor (e.g. the hypernym hierarchy) can support the transference of discriminative attributes across the taxonomic hierarchy. For the triple `(planet, moon, body)` using immediate definitions of pivot and comparison incorrectly suggests that the triple `(planet, moon, body)` is discriminative. However, after expanding using super-type definitions, the model correctly identifies `body` as a property of both `planet` and `moon`.

## 6 Conclusion

This paper described an explicit word vector model targeting the identification of discriminative attributes using the composition of definitions, visual features and commonsense knowledge graphs. The proposed model, which is built from structured representations from different types of data sources is able to achieve a state-of-the-art level F1-score (0.69) while producing, human interpretable explanations. The paper also provided an in-depth comparative quantitative and qualitative analysis on the contributions of different types of data sources for the generation of explicit semantic vector spaces (WordNet glosses, ConceptNet and Visual Genome), demonstrating the complementarity aspect of these resources. Future work will concentrate on extending the model to cope with relative attributes, the inclusion of additional data sources to increase model coverage, such as large-scale definition sets.

## Acknowledgments

The authors would like to thank Viktor Schlegel for the initial supporting machine learning baselines (not included here) and related work discussions in the beginning of the project.

## References

Ignacio Arroyo-Fernández, Ivan Meza, and Carlos-Francisco Meéndez-Cruz. 2018. Unam at semeval-2018 task 10: Unsupervised semantic discriminative attribute identification in neural word embedding cones. In *Proceedings of The 12th International*

*Workshop on Semantic Evaluation*, pages 977–984. Association for Computational Linguistics.

Mohammed Attia, Younes Samih, Manaal Faruqui, and Wolfgang Maier. 2018. Ghh at semeval-2018 task 10: Discovering discriminative attributes in distributional semantics. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 947–952. Association for Computational Linguistics.

Tomáš Brychcín, Tomáš Hercig, Josef Steinberger, and Michal Konkol. 2018. Uwb at semeval-2018 task 10: Capturing discriminative attributes from word distributions. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 935–939. Association for Computational Linguistics.

Bogdan Dumitru, Alina Maria Ciobanu, and Liviu P. Dinu. 2018. Alb at semeval-2018 task 10: A system for capturing discriminative attributes. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 963–967. Association for Computational Linguistics.

Andre Freitas. 2015. *Schema-agnositc queries over large-schema databases: a distributional semantics approach*. Ph.D. thesis.

Pablo Gamallo. 2018. Citiusnlp at semeval-2018 task 10: The use of transparent distributional models and salient contexts to discriminate word attributes. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 953–957. Association for Computational Linguistics.

José-Ángel González, Lluís-F. Hurtado, Encarna Segarra, and Ferran Pla. 2018. Elirf-upv at semeval-2018 task 10: Capturing discriminative attributes with knowledge graphs and wikipedia. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 968–971. Association for Computational Linguistics.

Maxim Grishin. 2018. Igevorse at semeval-2018 task 10: Exploring an impact of word embeddings concatenation for capturing discriminative attributes. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 995–998. Association for Computational Linguistics.

Patrick J. Hayes. 1995. The second naive physics manifesto. *Computation and Intelligence*.

Milton King, Ali Hakimi Parizi, and Paul Cook. 2018. Unbnlp at semeval-2018 task 10: Evaluating unsupervised approaches to capturing discriminative attributes. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 1013–1016. Association for Computational Linguistics.

Alicia Krebs, Allesandro Lenci, and Denis Paperno. 2018. Semeval 2018 task 10: Capturing discriminative attributes. In *Proceedings of the 12th international workshop on semantic evaluation (SemEval 2018)*.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123(1):32–73.

Artur Kulmizev, Mostafa Abdou, Vinit Ravishankar, and Malvina Nissim. 2018. Discriminator at semeval-2018 task 10: Minimally supervised discrimination. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 1008–1012. Association for Computational Linguistics.

Sunny Lai, Kwong Sak Leung, and Yee Leung. 2018. Sunnynlp at semeval-2018 task 10: A support-vector-machine-based method for detecting semantic difference using taxonomy and word embedding features. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 741–746. Association for Computational Linguistics.

Rui Mao, Guanyi Chen, Ruizhe Li, and Chenghua Lin. 2018. Abdn at semeval-2018 task 10: Recognising discriminative attributes using context embeddings and wordnet. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 1017–1021. Association for Computational Linguistics.

Teresa Robertson and Philip Atkins. 2018. Essential vs. accidental properties. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*, spring 2018 edition. Metaphysics Research Lab, Stanford University.

Enrico Santus, Chris Biemann, and Emmanuele Chersoni. 2018. Bomji at semeval-2018 task 10: Combining vector-, pattern- and graph-based information to identify discriminative attributes. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 990–994. Association for Computational Linguistics.

Yow-Ting Shiue, Hen-Hsen Huang, and Hsin-Hsi Chen. 2018. Ntu nlp lab system at semeval-2018 task 10: Verifying semantic differences by integrating distributional information and expert knowledge. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 1027–1033. Association for Computational Linguistics.

Vivian S. Silva, Andre Freitas, and Siegfried Handschuh. 2017. Building a knowledge graph from natural language definitions for interpretable text entailment recognition. In *11th Language Resources and Evaluation Conference*.

Vivian S Silva, Andre Freitas, and Siegfried Handschuh. 2018a. Exploring knowledge graphs in an interpretable composite approach for text entailment. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Vivian S. Silva, André Freitas, and Siegfried Handschuh. 2019. On the semantic interpretability of artificial intelligence models. *ArXiV (Preprint)*, abs/1907.04105.

Vivian S. Silva, Siegfried Handschuh, and André Freitas. 2016. Categorization of semantic roles for dictionary definitions. In *Proceedings of the 5th Workshop on Cognitive Aspects of the Lexicon (CogALex - V)*, pages 176–184.

Vivian S. Silva, Siegfried Handschuh, and Andre Freitas. 2018b. Recognizing and justifying text entailment through distributional navigation on definition graphs. In *Thirty-Second AAAI Conference on Artificial Intelligence*, United States. AAAI Press.

Pia Sommerauer, Antske Fokkens, and Piek Vossen. 2018. Meaning_space at semeval-2018 task 10: Combining explicitly encoded knowledge with information extracted from word embeddings. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 940–946. Association for Computational Linguistics.

Robert Speer and Joanna Lowry-Duda. 2018. Luminoso at semeval-2018 task 10: Distinguishing attributes using text corpora and relational knowledge. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 985–989. Association for Computational Linguistics.

Peter D Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37:141–188.

Vivek Vinayan, Anand Kumar M, and Soman K P. 2018. Amritanlp at semeval-2018 task 10: Capturing discriminative attributes using convolution neural network over global vector representation. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 1003–1007. Association for Computational Linguistics.

Chuhan Wu, Fangzhao Wu, Sixing Wu, Zhigang Yuan, and Yongfeng Huang. 2018. Thu_ngn at semeval-2018 task 10: Capturing discriminative attributes with mlp-cnn model. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 958–962. Association for Computational Linguistics.

Alexander Zhang and Marine Carpuat. 2018. Umd at semeval-2018 task 10: Can word embeddings capture discriminative attributes? In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 1022–1026. Association for Computational Linguistics.

Yunxiao Zhou, Man Lan, and Yuanbin Wu. 2018. Ecnu at semeval-2018 task 10: Evaluating simple but effective features on machine learning methods for semantic difference detection. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 999–1002. Association for Computational Linguistics.

# Patient Knowledge Distillation for BERT Model Compression

**Siqi Sun, Yu Cheng, Zhe Gan, Jingjing Liu**
Microsoft Dynamics 365 AI Research
{Siqi.Sun,Yu.Cheng,Zhe.Gan,jingjl}@microsoft.com

## Abstract

Pre-trained language models such as BERT have proven to be highly effective for natural language processing (NLP) tasks. However, the high demand for computing resources in training such models hinders their application in practice. In order to alleviate this resource hunger in large-scale model training, we propose a Patient Knowledge Distillation approach to compress an original large model (teacher) into an equally-effective lightweight shallow network (student). Different from previous knowledge distillation methods, which only use the output from the last layer of the teacher network for distillation, our student model *patiently* learns from multiple intermediate layers of the teacher model for incremental knowledge extraction, following two strategies: ($i$) PKD-Last: learning from the last $k$ layers; and ($ii$) PKD-Skip: learning from every $k$ layers. These two patient distillation schemes enable the exploitation of rich information in the teacher's hidden layers, and encourage the student model to *patiently* learn from and imitate the teacher through a multi-layer distillation process. Empirically, this translates into improved results on multiple NLP tasks with significant gain in training efficiency, without sacrificing model accuracy.[1]

## 1 Introduction

Language model pre-training has proven to be highly effective in learning universal language representations from large-scale unlabeled data. ELMo (Peters et al., 2018), GPT (Radford et al., 2018) and BERT (Devlin et al., 2018) have achieved great success in many NLP tasks, such as sentiment classification (Socher et al., 2013), natural language inference (Williams et al., 2017), and question answering (Lai et al., 2017).

Despite its empirical success, BERT's computational efficiency is a widely recognized issue because of its large number of parameters. For example, the original BERT-Base model has 12 layers and 110 million parameters. Training from scratch typically takes four days on 4 to 16 Cloud TPUs. Even fine-tuning the pre-trained model with task-specific dataset may take several hours to finish one epoch. Thus, reducing computational costs for such models is crucial for their application in practice, where computational resources are limited.

Motivated by this, we investigate the redundancy issue of learned parameters in large-scale pre-trained models, and propose a new model compression approach, *Patient Knowledge Distillation* (Patient-KD), to compress original teacher (e.g., BERT) into a lightweight student model without performance sacrifice. In our approach, the teacher model outputs probability logits and predicts labels for the training samples (extendable to additional unannotated samples), and the student model learns from the teacher network to mimic the teacher's prediction.

Different from previous knowledge distillation methods (Hinton et al., 2015; Sau and Balasubramanian, 2016; Lu et al., 2017), we adopt a *patient* learning mechanism: instead of learning parameters from only the last layer of the teacher, we encourage the student model to extract knowledge also from previous layers of the teacher network. We call this 'Patient Knowledge Distillation'. This patient learner has the advantage of distilling rich information through the deep structure of the teacher network for multi-layer knowledge distillation.

We also propose two different strategies for the distillation process: ($i$) PKD-Last: the student learns from the last $k$ layers of the teacher, under the assumption that the top layers of the original network contain the most informative knowledge

[1]Code will be avialable at https://github.com/intersun/PKD-for-BERT-Model-Compression.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4323–4332, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

to teach the student; and ($ii$) PKD-Skip: the student learns from every $k$ layers of the teacher, suggesting that the lower layers of the teacher network also contain important information and should be passed along for incremental distillation.

We evaluate the proposed approach on several NLP tasks, including Sentiment Classification, Paraphrase Similarity Matching, Natural Language Inference, and Machine Reading Comprehension. Experiments on seven datasets across these four tasks demonstrate that the proposed Patient-KD approach achieves superior performance and better generalization than standard knowledge distillation methods (Hinton et al., 2015), with significant gain in training efficiency and storage reduction while maintaining comparable model accuracy to original large models. To the authors' best knowledge, this is the first known effort for BERT model compression.

## 2 Related Work

**Language Model Pre-training** Pre-training has been widely applied to universal language representation learning. Previous work can be divided into two main categories: ($i$) feature-based approach; ($ii$) fine-tuning approach.

Feature-based methods mainly focus on learning: ($i$) context-independent word representation (e.g., word2vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014), FastText (Bojanowski et al., 2017)); ($ii$) sentence-level representation (e.g., Kiros et al. (2015); Conneau et al. (2017); Logeswaran and Lee (2018)); and ($iii$) contextualized word representation (e.g., Cove (McCann et al., 2017), ELMo (Peters et al., 2018)). Specifically, ELMo (Peters et al., 2018) learns high-quality, deep contextualized word representation using bidirectional language model, which can be directly plugged into standard NLU models for performance boosting.

On the other hand, fine-tuning approaches mainly pre-train a language model (e.g., GPT (Radford et al., 2018), BERT (Devlin et al., 2018)) on a large corpus with an unsupervised objective, and then fine-tune the model with in-domain labeled data for downstream applications (Dai and Le, 2015; Howard and Ruder, 2018). Specifically, BERT is a large-scale language model consisting of multiple layers of Transformer blocks (Vaswani et al., 2017). BERT-Base has 12 layers of Transformer and 110 million parameters, while BERT-Large has 24 layers of Transformer and 330 million parameters. By pre-training via masked language modeling and next sentence prediction, BERT has achieved state-of-the-art performance on a wide-range of NLU tasks, such as the GLUE benchmark (Wang et al., 2018) and SQuAD (Rajpurkar et al., 2016).

However, these modern pre-trained language models contain millions of parameters, which hinders their application in practice where computational resource is limited. In this paper, we aim at addressing this critical and challenging problem, taking BERT as an example, *i.e.,* how to compress a large BERT model into a shallower one without sacrificing performance. Besides, the proposed approach can also be applied to other large-scale pre-trained language models, such as recently proposed XLNet (Yang et al., 2019) and RoBERTa (Liu et al., 2019b).

**Model Compression & Knowledge Distillation** Our focus is model compression, *i.e.,* making deep neural networks more compact (Han et al., 2016; Cheng et al., 2015). A similar line of work has focused on accelerating deep network inference at test time (Vetrov et al., 2017) and reducing model training time (Huang et al., 2016).

A conventional understanding is that a large number of connections (weights) is necessary for training deep networks (Denil et al., 2013; Zhai et al., 2016). However, once the network has been trained, there will be a high degree of parameter redundancy. Network pruning (Han et al., 2015; He et al., 2017), in which network connections are reduced or sparsified, is one common strategy for model compression. Another direction is weight quantization (Gong et al., 2014; Polino et al., 2018), in which connection weights are constrained to a set of discrete values, allowing weights to be represented by fewer bits. However, most of these pruning and quantization approaches perform on convolutional networks. Only a few work are designed for rich structural information such as deep language models (Changpinyo et al., 2017).

Knowledge distillation (Hinton et al., 2015) aims to compress a network with a large set of parameters into a compact and fast-to-execute model. This can be achieved by training a compact model to imitate the soft output of a larger model. Romero et al. (2015) further demonstrated that intermediate representations learned by the

large model can serve as hints to improve the training process and the final performance of the compact model. Chen et al. (2015) introduced techniques for efficiently transferring knowledge from an existing network to a deeper or wider network. More recently, Liu et al. (2019a) used knowledge from ensemble models to improve single model performance on NLU tasks. Tan et al. (2019) tried knowledge distillation for multilingual translation. Different from the above efforts, we investigate the problem of compressing large-scale language models, and propose a novel *patient knowledge distillation* approach to effectively transferring knowledge from a teacher to a student model.

## 3 Patient Knowledge Distillation

In this section, we first introduce a vanilla knowledge distillation method for BERT compression (Section 3.1), then present the proposed Patient Knowledge Distillation (Section 3.2) in details.

**Problem Definition** The original large teacher network is represented by a function $f(\mathbf{x};\theta)$, where $\mathbf{x}$ is the input to the network, and $\theta$ denotes the model parameters. The goal of knowledge distillation is to learn a new set of parameters $\theta'$ for a shallower student network $g(\mathbf{x};\theta')$, such that the student network achieves similar performance to the teacher, with much lower computational cost. Our strategy is to force the student model to imitate outputs from the teacher model on the training dataset with a defined objective $L_{KD}$.

### 3.1 Distillation Objective

In our setting, the teacher $f(\mathbf{x};\theta)$ is defined as a deep bidirectional encoder, e.g., BERT, and the student $g(\mathbf{x};\theta')$ is a lightweight model with fewer layers. For simplicity, we use $\text{BERT}_k$ to denote a model with $k$ layers of Transformers. Following the original BERT paper (Devlin et al., 2018), we also use BERT-Base and BERT-Large to denote $\text{BERT}_{12}$ and $\text{BERT}_{24}$, respectively.

Assume $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{N}$ are $N$ training samples, where $\mathbf{x}_i$ is the $i$-th input instance for BERT, and $\mathbf{y}_i$ is the corresponding ground-truth label. BERT first computes a contextualized embedding $\mathbf{h}_i = \text{BERT}(\mathbf{x}_i) \in \mathbb{R}^d$. Then, a softmax layer $\hat{\mathbf{y}}_i = P(\mathbf{y}_i|\mathbf{x}_i) = softmax(\mathbf{W}\mathbf{h}_i)$ for classification is applied to the embedding of BERT output, where $\mathbf{W}$ is a weight matrix to be learned.

To apply knowledge distillation, first we need to train a teacher network. For example, to train a 12-layer BERT-Base as the teacher model, the learned parameters are denoted as:

$$\hat{\theta}^t = \arg\min_{\theta} \sum_{i\in[N]} L_{CE}^t(\mathbf{x}_i, \mathbf{y}_i; [\theta_{\text{BERT}_{12}}, \mathbf{W}]) \tag{1}$$

where the superscript $t$ denotes parameters in the teacher model, $[N]$ denotes set $\{1, 2, \ldots, N\}$, $L_{CE}^t$ denotes the cross-entropy loss for the teacher training, and $\theta_{\text{BERT}_{12}}$ denotes parameters of $\text{BERT}_{12}$.

The output probability for any given input $\mathbf{x}_i$ can be formulated as:

$$\begin{aligned}\hat{\mathbf{y}}_i = P^t(\mathbf{y}_i|\mathbf{x}_i) &= softmax\Big(\frac{\mathbf{W}\mathbf{h}_i}{T}\Big) \\ &= softmax\Big(\frac{\mathbf{W}\cdot \text{BERT}_{12}(\mathbf{x}_i;\hat{\theta}^t)}{T}\Big)\end{aligned} \tag{2}$$

where $P^t(\cdot|\cdot)$ denotes the probability output from the teacher. $\hat{\mathbf{y}}_i$ is fixed as soft labels, and $T$ is the temperature used in KD, which controls how much to rely on the teacher's soft predictions. A higher temperature produces a more diverse probability distribution over classes (Hinton et al., 2015). Similarly, let $\theta^s$ denote parameters to be learned for the student model, and $P^s(\cdot|\cdot)$ denote the corresponding probability output from the student model. Thus, the distance between the teacher's prediction and the student's prediction can be defined as:

$$\begin{aligned}L_{DS} = -\sum_{i\in[N]}\sum_{c\in C}\Big[&P^t(\mathbf{y}_i = c|\mathbf{x}_i;\hat{\theta}^t)\cdot \\ &\log P^s(\mathbf{y}_i = c|\mathbf{x}_i;\theta^s)\Big]\end{aligned} \tag{3}$$

where $c$ is a class label and $C$ denotes the set of class labels.

Besides encouraging the student model to imitate the teacher's behavior, we can also fine-tune the student model on target tasks, where task-specific cross-entropy loss is included for model training:

$$\begin{aligned}L_{CE}^s = -\sum_{i\in[N]}\sum_{c\in C}\Big[&\mathbb{1}[\mathbf{y}_i = c]\cdot \\ &\log P^s(\mathbf{y}_i = c|\mathbf{x}_i;\theta^s)\Big]\end{aligned} \tag{4}$$

Thus, the final objective function for knowledge distillation can be formulated as:

$$L_{KD} = (1-\alpha)L_{CE}^s + \alpha L_{DS} \tag{5}$$

where $\alpha$ is the hyper-parameter that balances the importance of the cross-entropy loss and the distillation loss.

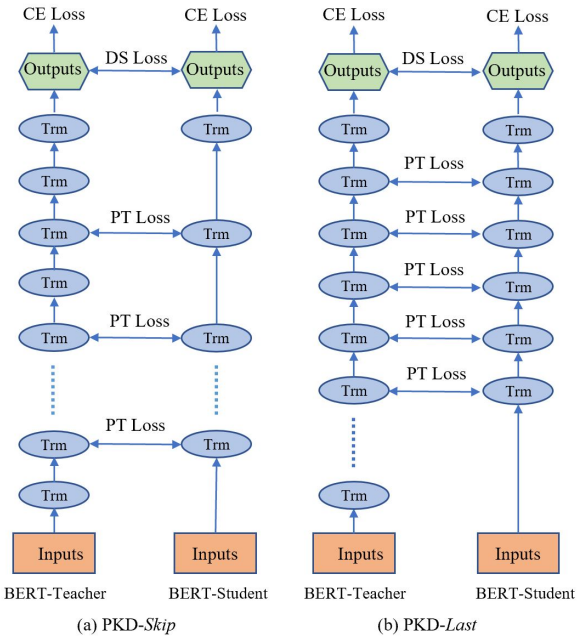


Figure 1: Model architecture of the proposed Patient Knowledge Distillation approach to BERT model compression. (Left) PKD-Skip: the student network learns the teacher's outputs in every 2 layers. (Right) PKD-Last: the student learns the teacher's outputs from the last 6 layers. Trm: Transformer.

### 3.2 Patient Teacher for Model Compression

Using a weighted combination of ground-truth labels and soft predictions from the last layer of the teacher network, the student network can achieve comparable performance to the teacher model on the training set. However, with the number of epochs increasing, the student model learned with this vanilla KD framework quickly reaches saturation on the test set (see Figure 2 in Section 4).

One hypothesis is that overfitting during knowledge distillation may lead to poor generalization. To mitigate this issue, instead of forcing the student to learn only from the logits of the last layer, we propose a "*patient*" teacher-student mechanism to distill knowledge from the teacher's intermediate layers as well. Specifically, we investigate two patient distillation strategies: $(i)$ PKD-Skip: the student learns from every $k$ layers of the teacher (Figure 1: Left); and $(ii)$ PKD-Last: the student learns from the *last* $k$ layers of the teacher (Figure 1: Right).

Learning from the hidden states of all the tokens is computationally expensive, and may introduce noise. In the original BERT implementation (Devlin et al., 2018), prediction is performed by only using the output from the last layer's `[CLS]` token. In some variants of BERT, like SDNet (Zhu et al., 2018), a weighted average of all layers' `[CLS]` embeddings is applied. In general, the final logit can be computed based on $\mathbf{h}_{\text{final}} = \sum_{j\in[k]} w_j \mathbf{h}_j$, where $w_j$ could be either learned parameters or a pre-defined hyper-parameter, $\mathbf{h}_j$ is the embedding of `[CLS]` from the hidden layer $j$, and $k$ is the number of hidden layers. Derived from this, if the compressed model can learn from the representation of `[CLS]` in the teacher's intermediate layers for any given input, it has the potential of gaining a generalization ability similar to the teacher model.

Motivated by this, in our Patient-KD framework, the student is cultivated to imitate the representations only for the `[CLS]` token in the intermediate layers, following the intuition aforementioned that the `[CLS]` token is important in predicting the final labels. For an input $\mathbf{x}_i$, the outputs of the `[CLS]` tokens for all the layers are denoted as:

$$\mathbf{h}_i = [\mathbf{h}_{i,1}, \mathbf{h}_{i,2}, \ldots, \mathbf{h}_{i,k}] = \text{BERT}_k(\mathbf{x}_i) \in \mathbb{R}^{k\times d} \tag{6}$$

We denote the set of intermediate layers to distill knowledge from as $I_{pt}$. Take distilling from $\text{BERT}_{12}$ to $\text{BERT}_6$ as an example. For the PKD-Skip strategy, $I_{pt} = \{2, 4, 6, 8, 10\}$; and for the PKD-Last strategy, $I_{pt} = \{7, 8, 9, 10, 11\}$. Note that $k = 5$ for both cases, because the output from the last layer (e.g., Layer 12 for BERT-Base) is omitted since its hidden states are connected to the softmax layer, which is already included in the KD loss defined in Eqn. (5). In general, for BERT student with $n$ layers, $k$ always equals to $n - 1$.

The additional training loss introduced by the patient teacher is defined as the mean-square loss between the normalized hidden states:

$$L_{PT} = \sum_{i=1}^{N}\sum_{j=1}^{M}\left|\left|\frac{\mathbf{h}_{i,j}^{s}}{||\mathbf{h}_{i,j}^{s}||_2} - \frac{\mathbf{h}_{i,I_{pt}(j)}^{t}}{||\mathbf{h}_{i,I_{pt}(j)}^{t}||_2}\right|\right|_2^2 \tag{7}$$

where $M$ denotes the number of layers in the student network, $N$ is the number of training samples, and the superscripts $s$ and $t$ in $\mathbf{h}$ indicate the student and the teacher model, respectively. Combined with the KD loss introduced in Section 3.1, the final objective function can be formulated as:

$$L_{PKD} = (1-\alpha)L_{CE}^{s} + \alpha L_{DS} + \beta L_{PT} \tag{8}$$

where $\beta$ is another hyper-parameter that weights the importance of the features for distillation in the intermediate layers.

## 4 Experiments

In this section, we describe our experiments on applying the proposed Patient-KD approach to four different NLP tasks. Details on the datasets and experimental results are provided in the following sub-sections.

### 4.1 Datasets

We evaluate our proposed approach on Sentiment Classification, Paraphrase Similarity Matching, Natural Language Inference, and Machine Reading Comprehension tasks. For Sentiment Classification, we test on Stanford Sentiment Treebank (SST-2) (Socher et al., 2013). For Paraphrase Similarity Matching, we use Microsoft Research Paraphrase Corpus (MRPC) (Dolan and Brockett, 2005) and Quora Question Pairs (QQP)[2] datasets. For Natural Language Inference, we evaluate on Multi-Genre Natural Language Inference (MNLI) (Williams et al., 2017), QNLI[3] (Rajpurkar et al., 2016), and Recognizing Textual Entailment (RTE).

More specifically, SST-2 is a movie review dataset with binary annotations, where the binary label indicates positive and negative reviews. MRPC contains pairs of sentences and corresponding labels, which indicate the semantic equivalence relationship between each pair. QQP is designed to predict whether a pair of questions is duplicate or not, provided by a popular online question-answering website Quora. MNLI is a multi-domain NLI task for predicting whether a given premise-hypothesis pair is entailment, contradiction or neural. Its test and development datasets are further divided into in-domain (MNLI-m) and cross-domain (MNLI-mm) splits to evaluate the generality of tested models. QNLI is a task for predicting whether a question-answer pair is entailment or not. Finally, RTE is based on a series of textual entailment challenges, created by General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018).

For the Machine Reading Comprehension task, we evaluate on RACE (Lai et al., 2017), a large-scale dataset collected from English exams, containing 25,137 passages and 87,866 questions. For each question, four candidate answers are provided, only one of which is correct. The dataset is further divided into RACE-M and RACE-H, containing exam questions for middle school and high school students.

### 4.2 Baselines and Training Details

For experiments on the GLUE benchmark, since all the tasks can be considered as sentence (or sentence-pair) classification, we use the same architecture in the original BERT (Devlin et al., 2018), and fine-tune each task independently.

For experiments on RACE, we denote the input passage as $P$, the question as $q$, and the four answers as $a_1, \ldots, a_4$. We first concatenate the tokens in $q$ and each $a_i$, and arrange the input of BERT as `[CLS]` $P$ `[SEP]` $q+a_i$ `[SEP]` for each input pair $(P, q + a_i)$, where `[CLS]` and `[SEP]` are the special tokens used in the original BERT. In this way, we can obtain a single logit value for each $a_i$. At last, a softmax layer is placed on top of these four logits to obtain the normalized probability of each answer $a_i$ being correct, which is then used to compute the cross-entropy loss for modeling training.

We fine-tune BERT-Base (denoted as $\text{BERT}_{12}$) as the teacher model to compute soft labels for each task independently, where the pretrained model weights are obtained from Google's official BERT's repo[4], and use 3 and 6 layers of Transformers as the student models ($\text{BERT}_3$ and $\text{BERT}_6$), respectively. We initialize $\text{BERT}_k$ with the first $k$ layers of parameters from pre-trained BERT-Base, where $k \in \{3, 6\}$. To validate the effectiveness of our proposed approach, we first conduct direct fine-tuning on each task without using any soft labels. In order to reduce the hyper-parameter search space, we fix the number of hidden units in the final softmax layer as 768, the batch size as 32, and the number of epochs as 4 for all the experiments, with a learning rate from {5e-5, 2e-5, 1e-5}. The model with the best validation accuracy is selected for each setting.

Besides direct fine-tuning, we further implement a vanilla KD method on all the tasks by optimizing the objective function in Eqn. (5). We set the temperature $T$ as {5, 10, 20}, $\alpha = \{0.2, 0.5, 0.7\}$, and perform grid search over $T$, $\alpha$ and learning rate, to select the model with the best validation accuracy. For our proposed Patient-KD approach, we conduct additional search over $\beta$

[2] https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs

[3] The dataset is derived from Stanford Question Answer Dataset (SQuAD).

[4] https://github.com/google-research/bert

| Model | SST-2 (67k) | MRPC (3.7k) | QQP (364k) | MNLI-m (393k) | MNLI-mm (393k) | QNLI (105k) | RTE (2.5k) |
|---|---|---|---|---|---|---|---|
| $BERT_{12}$ (Google) | 93.5 | 88.9/84.8 | 71.2/89.2 | 84.6 | 83.4 | 90.5 | 66.4 |
| $BERT_{12}$ (Teacher) | 94.3 | 89.2/85.2 | 70.9/89.0 | 83.7 | 82.8 | 90.4 | 69.1 |
| $BERT_6$-FT | 90.7 | 85.9/80.2 | 69.2/88.2 | 80.4 | 79.7 | 86.7 | 63.6 |
| $BERT_6$-KD | 91.5 | **86.2/80.6** | 70.1/88.8 | 80.2 | 79.8 | 88.3 | 64.7 |
| $BERT_6$-PKD | **92.0** | 85.0/79.9 | **70.7/88.9** | **81.5** | **81.0** | **89.0** | **65.5** |
| $BERT_3$-FT | 86.4 | 80.5/**72.6** | 65.8/86.9 | 74.8 | 74.3 | 84.3 | 55.2 |
| $BERT_3$-KD | 86.9 | 79.5/71.1 | 67.3/87.6 | 75.4 | 74.8 | 84.0 | 56.2 |
| $BERT_3$-PKD | **87.5** | **80.7**/72.5 | **68.1/87.8** | **76.7** | **76.3** | **84.7** | **58.2** |

Table 1: Results from the GLUE test server. The best results for 3-layer and 6-layer models are in-bold. Google's submission results are obtained from official GLUE leaderboard. $BERT_{12}$ (Teacher) is our own implementation of the BERT teacher model. FT represents direct fine-tuning on each dataset without using knowledge distillation. KD represents using a vanilla knowledge distillation method. And PKD represents our proposed Patient-KD-Skip approach. Results show that PKD-Skip outperforms the baselines on almost all the datasets except for MRPC. The numbers under each dataset indicate the corresponding number of training samples.

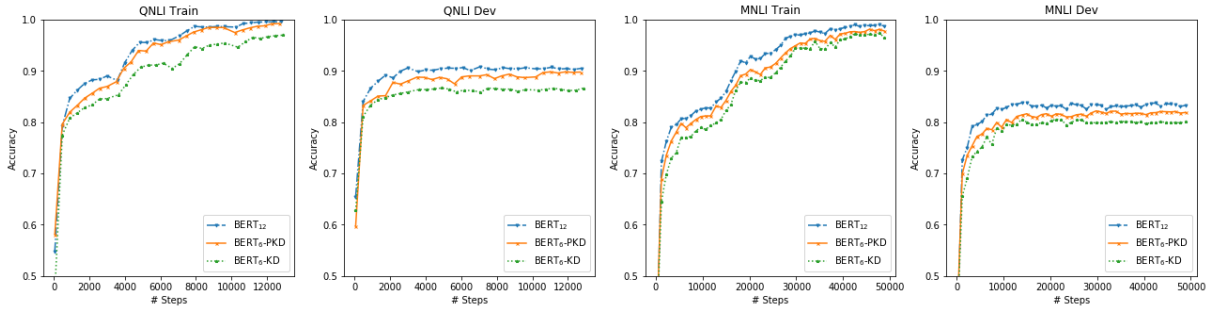


Figure 2: Accuracy on the training and dev sets of QNLI and MNLI datasets, by directly applying vanilla knowledge distillation (KD) and the proposed Patient-KD-Skip. The teacher and the student networks are $BERT_{12}$ and $BERT_6$, respectively. The student network learned with vanilla KD quickly saturates on the dev set, while the proposed Patient-KD starts to plateau only in a later stage.

from $\{10, 100, 500, 1000\}$ on all the tasks. Since there are so many hyper-parameters to learn for Patient KD, we fix $\alpha$ and $T$ to the values used in the model with the best performance from the vanilla KD experiments, and only search over $\beta$ and learning rate.

### 4.3 Experimental Results

We submitted our model predictions to the official GLUE evaluation server to obtain results on the test data. Results are summarized in Table 1. Compared to direct fine-tuning and vanilla KD, our Patient-KD models with $BERT_3$ and $BERT_6$ students perform the best on almost all the tasks except MRPC. For MNLI-m and MNLI-mm, our 6-layer model improves 1.1% and 1.3% over fine-tune (FT) baselines; for QNLI and QQP, even though the gap between $BERT_6$-KD and $BERT_{12}$ teacher is relatively small, our approach still succeeded in improving over both FT and KD baselines and further closing the gap between the student and the teacher models.

Furthermore, in 5 tasks out of 7 (SST-2 (-2.3% compared to BERT-Base teacher), QQP (-0.1%), MNLI-m (-2.2%), MNLI-mm (-1.8%), and QNLI (-1.4%)), the proposed 6-layer student coached by the patient teacher achieved similar performance to the original BERT-Base, demonstrating the effectiveness of our approach. Interestingly, all those 5 tasks have more than 60k training samples, which indicates that our method tends to perform better when there is a large amount of training data.

For the QQP task, we can further reduce the model size to 3 layers, where $BERT_3$-PKD can still have a similar performance to the teacher model. The learning curves on the QNLI and MNLI datasets are provided in Figure 2. The student model learned with vanilla KD quickly saturated on the dev set, while the proposed Patient-

| Model | SST-2 | MRPC | QQP | MNLI-m | MNLI-mm | QNLI | RTE |
|---|---|---|---|---|---|---|---|
| $BERT_6$ (PKD-Last) | 91.9 | **85.1**/79.5 | 70.5/**88.9** | 80.9 | **81.0** | 88.2 | 65.0 |
| $BERT_6$ (PKD-Skip) | **92.0** | 85.0/**79.9** | **70.7/88.9** | **81.5** | **81.0** | **89.0** | **65.5** |

Table 2: Performance comparison between PKD-Last and PKD-Skip on GLUE benchmark.

KD keeps learning from the teacher and improving accuracy, only starting to plateau in a later stage.

For the MRPC dataset, one hypothesis for the reason on vanilla KD outperforming our model is that the lack of enough training samples may lead to overfitting on the dev set. To further investigate, we repeat the experiments three times and compute the average accuracy on the dev set. We observe that fine-tuning and vanilla KD have a mean dev accuracy of 82.23% and 82.84%, respectively. Our proposed method has a higher mean dev accuracy of 83.46%, hence indicating that our Patient-KD method slightly overfitted to the dev set of MRPC due to the small amount of training data. This can also be observed on the performance gap between teacher and student on RTE in Table 5, which also has a small training set.

We further investigate the performance gain from two different patient teacher designs: PKD-Last vs. PKD-Skip. Results of both PKD variants on the GLUE benchmark (with $BERT_6$ as the student) are summarized in Table 2. Although both strategies achieved improvement over the vanilla KD baseline (see Table 1), PKD-Skip performs slightly better than PKD-Last. Presumably, this might be due to the fact that distilling information across every $k$ layers captures more diverse representations of richer semantics from low-level to high-level, while focusing on the last $k$ layers tends to capture relatively homogeneous semantic information.

Results on RACE are reported in Table 3, which shows that the Vanilla KD method outperforms direct fine-tuning by 4.42%, and our proposed patient teacher achieves further 1.6% performance lift, which again demonstrates the effectiveness of Patient-KD.

### 4.4 Analysis of Model Efficiency

We have demonstrated that the proposed Patient-KD method can effectively compress $BERT_{12}$ into $BERT_6$ models without performance sacrifice. In this section, we further investigate the efficiency of Patient-KD on storage saving and inference-time speedup. Parameter statistics and inference time

| Model | RACE | RACE-M | RACE-H |
|---|---|---|---|
| $BERT_{12}$ (Leaderboard) | 65.00 | 71.70 | 62.30 |
| $BERT_{12}$ (Teacher) | 65.30 | 71.17 | 62.89 |
| $BERT_6$-FT | 54.32 | 61.07 | 51.54 |
| $BERT_6$-KD | 58.74 | 64.69 | 56.29 |
| $BERT_6$-PKD-Skip | **60.34** | **66.57** | **57.78** |

Table 3: Results on RACE test set. $BERT_{12}$ (Leaderboard) denotes results extracted from the official leaderboard (http://www.qizhexie.com/data/RACE_leaderboard). $BERT_{12}$ (Teacher) is our own implementation. Results of $BERT_3$ are not included due to the large gap between the teacher and the $BERT_3$ student.

are summarized in Table 4. All the models share the same embedding layer with 24 millon parameters that map a 30k-word vocabulary to a 768-dimensional vector, which leads to 1.64 and 2.4 times of machine memory saving from $BERT_6$ and $BERT_3$, respectively.

To test the inference speed, we ran experiments on 105k samples from QNLI training set (Rajpurkar et al., 2016). Inference is performed on a single Titan RTX GPU with batch size set to 128, maximum sequence length set to 128, and FP16 activated. The inference time for the embedding layer is negligible compared to the Transformer layers. Results in Table 4 show that the proposed Patient-KD approach achieves an almost linear speedup, 1.94 and 3.73 times for $BERT_6$ and $BERT_3$, respectively.

### 4.5 Does a Better Teacher Help?

To evaluate the effectiveness of the teacher model in our Patient-KD framework, we conduct additional experiments to measure the difference between BERT-Base teacher and BERT-Large teacher for model compression.

Each Transformer layer in BERT-Large has 12.6 million parameters, which is much larger than the Transformer layer used in BERT-Base. For a compressed BERT model with 6 layers, $BERT_6$ with BERT-Base Transformer (denoted as $BERT_6$[Base]) has only 67.0 million parameters,

| # Layers | # Param (Emb) | # Params (Trm) | Total Params | Inference Time (s) |
|---|---|---|---|---|
| 3 | 23.8M | 21.3M | 45.7M (2.40×) | 27.35 (3.73×) |
| 6 | 23.8M | 42.5M | 67.0M (1.64×) | 52.51 (1.94×) |
| 12 | 23.8M | 85.1M | 109M (1×) | 101.89 (1×) |

Table 4: The number of parameters and inference time for $\text{BERT}_3$, $\text{BERT}_6$ and $\text{BERT}_{12}$. Parameters in Transformers (Trm) grow linearly with the increase of layers. Note that the summation of # Param (Emb) and # Param (Trm) does not exactly equal to Total Params, because there is another softmax layer with 0.6M parameters.

| Setting | Teacher | Student | SST-2 | MRPC | QQP | MNLI-m | MNLI-mm | QNLI | RTE |
|---|---|---|---|---|---|---|---|---|---|
| N/A | N/A | $\text{BERT}_{12}$ (Teacher) | 94.3 | 89.2/85.2 | 70.9/89.0 | 83.7 | 82.8 | 90.4 | 69.1 |
| N/A | N/A | $\text{BERT}_{24}$ (Teacher) | 94.3 | 88.2/84.3 | 71.9/89.4 | 85.7 | 84.8 | 92.2 | 72.8 |
| #1 | $\text{BERT}_{12}$ | $\text{BERT}_6$[Base]-KD | 91.5 | 86.2/80.6 | 70.1/88.8 | 79.7 | 79.1 | 88.3 | 64.7 |
| #2 | $\text{BERT}_{24}$ | $\text{BERT}_6$[Base]-KD | 91.2 | 86.1/80.7 | 69.4/88.6 | 80.2 | 79.7 | 87.5 | 65.7 |
| #3 | $\text{BERT}_{24}$ | $\text{BERT}_6$[Large]-KD | 89.6 | 79.0/70.0 | 65.0/86.7 | 75.3 | 74.6 | 83.4 | 53.7 |
| #4 | $\text{BERT}_{24}$ | $\text{BERT}_6$[Large]-PKD | 89.8 | 77.8/68.3 | 67.1/87.9 | 77.2 | 76.7 | 83.8 | 53.2 |

Table 5: Performance comparison with different teacher and student models. $\text{BERT}_6$[Base]/[Large] denotes a $\text{BERT}_6$ model with a BERT-Base/Large Transformer in each layer. For PKD, we use the PKD-Skip architecture.

while $\text{BERT}_6$ with BERT-Large Transformer (denoted as $\text{BERT}_6$[Large]) has 108.4 million parameters. Since the size of the `[CLS]` token embedding is different between BERT-Large and BERT-Base, we cannot directly compute the patient teacher loss (7) for $\text{BERT}_6$[Base] when BERT-Large is used as teacher. Hence, in the case where the teacher is BERT-Large and the student is $\text{BERT}_6$[Base], we only conduct experiments in the vanilla KD setting.

Results are summarized in Table 5. When the teacher changes from $\text{BERT}_{12}$ to $\text{BERT}_{24}$ (*i.e.*, Setting #1 vs. #2), there is not much difference between the students' performance. Specifically, $\text{BERT}_{12}$ teacher performs better on SST-2, QQP and QNLI, while $\text{BERT}_{24}$ performs better on MNLI-m, MNLI-mm and RTE. Presumably, distilling knowledge from a larger teacher requires a larger training dataset, thus better results are observed on MNLI-m and MNLI-mm.

We also report results on using BERT-Large as the teacher and $\text{BERT}_6$[Large] as the student. Interestingly, when comparing Setting #1 with #3, $\text{BERT}_6$[Large] performs much worse than $\text{BERT}_6$[Base] even though a better teacher is used in the former case. The $\text{BERT}_6$[Large] student also has 1.6 times more parameters than $\text{BERT}_6$[Base]. One intuition behind this is that the compression ratio for the $\text{BERT}_6$[Large] model is 4:1 (24:6), which is larger than the ratio used for the $\text{BERT}_6$[Base] model (2:1 (12:6)). The higher compression ratio renders it more challenging for the student model to absorb important weights.

When comparing Setting # 2 and #3, we observe that even when the same large teacher is used, $\text{BERT}_6$[Large] still performs worse than $\text{BERT}_6$[Base]. Presumably, this may be due to initialization mismatch. Ideally, we should pre-train $\text{BERT}_6$[Large] and $\text{BERT}_6$[Base] from scratch, and use the weights learned from the pre-training step for weight initialization in KD training. However, due to computational limits of training $\text{BERT}_6$ from scratch, we only initialize the student model with the first six layers of $\text{BERT}_{12}$ or $\text{BERT}_{24}$. Therefore, the first six layers of $\text{BERT}_{24}$ may not be able to capture high-level features, leading to worse KD performance.

Finally, when comparing Setting #3 vs. #4, where for setting #4 we use Patient-KD-Skip instead of vanilla KD, we observe a performance gain on almost all the tasks, which indicates Patient-KD is a generic approach independent of the selection of the teacher model ($\text{BERT}_{12}$ or $\text{BERT}_{24}$).

## 5 Conclusion

In this paper, we propose a novel approach to compressing a large BERT model into a shallow one via Patient Knowledge Distillation. To fully utilize the rich information in deep structure of the teacher network, our Patient-KD approach encourages the student model to patiently learn from the teacher through a multi-layer distillation process. Extensive experiments over four NLP tasks demonstrate the effectiveness of our proposed model.

For future work, we plan to pre-train BERT from scratch to address the initialization mismatch

issue, and potentially modify the proposed method such that it could also help during pre-training. Designing more sophisticated distance metrics for loss functions is another exploration direction. We will also investigate Patient-KD in more complex settings such as multi-task learning and meta learning.

## References


Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *TACL*.

Soravit Changpinyo, Mark Sandler, and Andrey Zhmoginov. 2017. The power of sparsity in convolutional neural networks. *arXiv preprint arXiv:1702.06257*.

Tianqi Chen, Ian J. Goodfellow, and Jonathon Shlens. 2015. Net2net: Accelerating learning via knowledge transfer. *arXiv preprint arXiv:1511.05641*.

Yu Cheng, Felix X. Yu, Rogerio S. Feris, Sanjiv Kumar, Alok Choudhary, and Shi-Fu Chang. 2015. An exploration of parameter redundancy in deep networks with circulant projections. In *ICCV*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *NIPS*.

Misha Denil, Babak Shakibi, Laurent Dinh, Marc'Aurelio Ranzato, and Nando de Freitas. 2013. Predicting parameters in deep learning. In *NIPS*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing*.

Yunchao Gong, Liu Liu, Ming Yang, and Lubomir D. Bourdev. 2014. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*.

Song Han, Huizi Mao, and William J Dally. 2016. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *ICLR*.

Song Han, Jeff Pool, John Tran, and William J. Dally. 2015. Learning both weights and connections for efficient neural networks. In *NIPS*.

Yihui He, Xiangyu Zhang, and Jian Sun. 2017. Channel pruning for accelerating very deep neural networks. In *ICCV*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *ACL*.

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. 2016. Deep networks with stochastic depth. In *ECCV*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *NIPS*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019a. Improving multi-task deep neural networks via knowledge distillation for natural language understanding. *arXiv preprint arXiv:1904.09482*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Lajanugen Logeswaran and Honglak Lee. 2018. An efficient framework for learning sentence representations. In *ICLR*.

Liang Lu, Michelle Guo, and Steve Renals. 2017. Knowledge distillation for small-footprint highway networks. In *ICASSP*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *NIPS*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*.

Antonio Polino, Razvan Pascanu, and Dan Alistarh. 2018. Model compression via distillation and quantization. *arXiv preprint arXiv:1802.05668*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *arXiv*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *EMNLP*.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2015. Fitnets: Hints for thin deep nets. In *ICLR*.

Bharat Bhusan Sau and Vineeth N Balasubramanian. 2016. Deep model compression: Distilling knowledge from noisy teachers. *arXiv preprint arXiv:1610.09650*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*.

Xu Tan, Yi Ren, Di He, Tao Qin, Zhou Zhao, and Tie-Yan Liu. 2019. Multilingual neural machine translation with knowledge distillation. In *ICLR*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Dmitry P. Vetrov, Jonathan Huang, Li Zhang, Maxwell Collins, Michael Figurnov, Ruslan Salakhutdinov, and Yukun Zhu. 2017. Spatially adaptive computation time for residual networks. In *CVPR*.

Alex Wang, Amapreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Shuangfei Zhai, Yu Cheng, Weining Lu, and Zhongfei Mark Zhang. 2016. Doubly convolutional neural networks. In *NIPS*.

Chenguang Zhu, Michael Zeng, and Xuedong Huang. 2018. Sdnet: Contextualized attention-based deep network for conversational question answering. *arXiv preprint arXiv:1812.03593*.

# Neural Gaussian Copula for Variational Autoencoder

**Prince Zizhuang Wang**
Department of Computer Science
University of California Santa Barbara
zizhuang_wang@ucsb.edu

**William Yang Wang**
Department of Computer Science
University of California Santa Barbara
william@cs.ucsb.edu

## Abstract

Variational language models seek to estimate the posterior of latent variables with an approximated variational posterior. The model often assumes the variational posterior to be factorized even when the true posterior is not. The learned variational posterior under this assumption does not capture the dependency relationships over latent variables. We argue that this would cause a typical training problem called posterior collapse observed in all other variational language models. We propose Gaussian Copula Variational Autoencoder (VAE) to avert this problem. Copula is widely used to model correlation and dependencies of high-dimensional random variables, and therefore it is helpful to maintain the dependency relationships that are lost in VAE. The empirical results show that by modeling the correlation of latent variables explicitly using a neural parametric copula, we can avert this training difficulty while getting competitive results among all other VAE approaches. [1]

## 1 Introduction

**Variational Inference (VI)** (Wainwright et al., 2008; Hoffman et al., 2013) methods are inspired by *calculus of variation* (Gelfand et al., 2000). It can be dated back to the 18th century when it was mainly used to study the problem of the change of *functional*, which is defined as the mapping from functions to real space and can be understood as the function of functions. **VI** takes a distribution as a *functional* and then studies the problem of matching this distribution to a target distribution using *calculus of variation*. After the rise of *deep learning* (Krizhevsky et al., 2012), a deep generative model called **Variational Autoencoder** (Kingma and Welling, 2014; Hoffman et al., 2013) is proposed based on the theory of **VI** and achieves great success over a huge number of tasks, such as transfer learning (Shen et al., 2017), unsupervised learning (Jang et al., 2017), image generation (Gregor et al., 2015), semi-supervised classification (Jang et al., 2017), and dialogue generation (Zhao et al., 2017). VAE is able to learn a continuous space of latent random variables which are useful for a lot of classification and generation tasks.

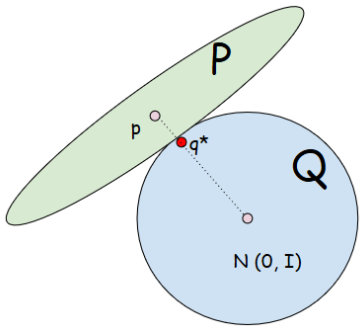


Figure 1: Intuitive illustration of **VI**: The elliptic $P$ is a distribution family containing the true posterior $p \in P$, and the circle $Q$ is a Mean-field variational family containing a standard normal prior $\mathcal{N}$. The optimal solution $q^*$ is the one in $Q$ that has the smallest $KL(q||p)$. In reality these two families may not overlap.

Recent studies (Bowman et al., 2015; Yang et al., 2017; Xiao et al., 2018; Xu and Durrett, 2018) show that when it comes to text generation and language modeling, VAE does not perform well and often generates random texts without making good use of the learned latent codes. This phenomenon is called **Posterior Collapse**, where the Kullback-Leibler (KL) divergence between the posterior and the prior (*often assumed to be a standard Gaussian*) vanishes. It makes the latent codes completely useless because any text input will be mapped to a standard Gaussian variable. Many recent studies (Yang et al., 2017; Xu and Durrett, 2018; Xiao et al., 2018; Miao et al.,

[1] Code will be released at https://github.com/kingofspace0wzz/copula-vae-lm



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4333–4343,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

2016; He et al., 2018) try to address this issue by providing new model architectures or by changing the objective functions. Our research lies in this second direction. **We review the theory of VAE, and we argue that one of the most widely used assumptions in VAE, the Mean-field assumption, is problematic**. It assumes that all approximated solutions in a family of variational distributions should be factorized or dimensional-wise independent for tractability. We argue that it leads to the posterior collapse problem since any variational posterior learned in this way does not maintain the correlation among latent codes and will never match the true posterior which is unlikely factorized.

We avert this problem by proposing a **Neural Gaussian Copula (Copula-VAE)** model to train VAE on text data. Copula (Nelsen, 2007) can model dependencies of high-dimensional random variables and is very successful in risk management (Kole et al., 2007; McNeil et al., 2005), financial management (Wang and Hua, 2014), and other tasks that require the modeling of dependencies. We provide a reparameterization trick (Kingma and Welling, 2014) to incorporate it with VAE for language modeling. We argue that by maintaining the dependency relationships over latent codes, we can dramatically improve the performance of variational language modeling and avoid posterior collapse. Our major contributions can be summarized as the following:

- We propose Neural parameterized Gaussian Copula to get a better estimation of the posterior for latent codes.
- We provide a reparameterization technique for Gaussian Copula VAE. The experiments show that our method achieves competitive results among all other variational language modeling approaches.
- We perform a thorough analysis of the original VAE and copula VAE. The results and analysis reveal the salient drawbacks of VAE and explain how introducing a copula model could help avert the posterior collapse problem.

## 2 Related Work

**Copula: Before the rise of Deep Learning** Copula (Nelsen, 2007) is a multivariate distribution whose marginals are all uniformly distributed. Over the years, it is widely used to extract correlation within high-dimensional random variables, and achieves great success in many subjects such as risk management (Kole et al., 2007; McNeil et al., 2005), finance (Wang and Hua, 2014), civil engineering(Chen et al., 2012; Zhang and Singh, 2006), and visual description generation (Wang and Wen, 2015). In the past, copula is often estimated by Maximum Likelihood method (Choroś et al., 2010; Jaworski et al., 2010) via parametric or semi-parametric approaches (Tsukahara, 2005; Choroś et al., 2010). One major difficulty when estimating the copula and extracting dependencies is the dimensionality of random variables. To overcome the curse of dimensionality, a graphical model called *vine copula* (Joe and Kurowicka, 2011; Czado, 2010; Bedford et al., 2002) is proposed to estimate a high-dimensional copula density by breaking it into a set of bivariate conditional copula densities. However, this approach is often hand-designed which requires human experts to define the form of each bivariate copula, and hence often results in overfitting. Therefore, Gaussian copula (Xue-Kun Song, 2000; Frey et al., 2001) is often used since its multivariate expression has a simple form and hence does not suffer from the curse of dimensionality.

**VAE for Text** Bowman et al. (2015) proposed to use VAE for text generation by using LSTM as encoder-decoder. The encoder maps the hidden states to a set of latent variables, which are further used to generate sentences. While achieving relatively low sample perplexity and being able to generate easy-to-read texts, the LSTM VAE often results in posterior collapse, where the learned latent codes become useless for text generation.

Recently, many studies are focusing on how to avert this training problem. They either propose a new model architecture or modify the VAE objective. Yang et al. (2017) seeks to replace the LSTM (Hochreiter and Schmidhuber, 1997) decoder with a CNN decoder to control model expressiveness, as they suspect that the over-expressive LSTM is one reason that makes KL vanish. Xiao et al. (2018) introduces a topic variable and pre-trains a *Latent Dirichlet Allocation* (Blei et al., 2003) model to get a prior distribution over the topic information. Xu and Durrett (2018) believes the *bubble soup effect* of high-dimensional Gaussian distribution is the main reason that causes KL vanishing, and therefore learns

a hyper-spherical posterior over the latent codes.

## 3 Variational Inference

The problem of inference in probabilistic modeling is to estimate the posterior density $p(\mathbf{z}|\mathbf{x})$ of latent variable $\mathbf{z}$ given input samples $\{x_i\}_{i=1}^{D}$. The direct computation of the posterior is intractable in most cases since the normalizing constant $\int p(\mathbf{z}, \mathbf{x})d\mathbf{z}$ lacks an analytic form. To get an approximation of the posterior, many approaches use sampling methods such as Markov chain Monte Carlo (MCMC) (Gilks et al., 1995) and Gibbs sampling (George and McCulloch, 1993). The downside of sampling methods is that they are inefficient, and it is hard to tell how close the approximation is from the true posterior. The other popular inference approach, **variational inference (VI)** (Wainwright et al., 2008; Hoffman et al., 2013), does not have this shortcoming as it provides a distance metric to measure the fitness of an approximated solution.

In **VI**, we assume a variational family of distributions $\mathcal{Q}$ to approximate the true posterior. The *Kullback-Leibler* (**KL**) divergence is used to measure how close $q \in Q$ is to the true $p(\mathbf{z}|\mathbf{x})$. The optimal variational posterior $q* \in Q$ is then the one that minimizes the KL divergence

$$KL(q||p) = \sum q(\mathbf{z}|\mathbf{x}) log \frac{q(\mathbf{z}|\mathbf{x})}{p(\mathbf{z}|\mathbf{x})}$$

Based on this, *variational autoencoder* (**VAE**) (Kingma and Welling, 2014) is proposed as a latent generative model that seeks to learn a posterior of the latent codes by minimizing the KL divergence between the true joint density $p_\theta(\mathbf{x}, \mathbf{z})$ the variational joint density $q_\phi(\mathbf{z}, \mathbf{x})$. This is equivalent to maximizing the following *evidence lower bound* **ELBO**,

$$\begin{aligned}\mathcal{L}(\theta; \phi; \mathbf{x}) &= -KL(q_\phi(\mathbf{z}, \mathbf{x})||p_\theta(\mathbf{x}, \mathbf{z})) \\ &= \mathbb{E}_{q(\mathbf{x})}[\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] \\ &\quad - KL(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))]\end{aligned}$$

In this case, Mean-field (Kingma and Welling, 2014) assumption is often used for simplicity. That is, we assume that the members of variational family $Q$ are dimensional-wise independent, meaning that the posterior $q$ can be written as $q(\mathbf{z}|\mathbf{x}) = \prod_{i=1}^{D} q(z_i|\mathbf{x})$. The simplicity of this form makes the estimation of ELBO very easy. However, it also leads to a particular training difficulty called posterior collapse, where the KL divergence term becomes zero and the factorized variational posterior collapses to the prior. The latent codes $\mathbf{z}$ would then become useless since the generative model $p(\mathbf{x}|\mathbf{z})$ no longer depends on it.

We believe the problem comes from the nature of variational family itself and hence we propose our **Copula-VAE** which makes use of the dependency modeling ability of copula model to guide the variational posterior to match the true posterior. We will provide more details in the following sections.

We hypothesize that the Mean-field assumption is problematic itself as the $q$ under this assumption can never recover the true structure of $p$. On the other hand, **Copula-VAE** makes use of the dependency relationships maintained by a copula model to guide the variational posterior to match the true posterior. Our approach differs from *Gaussian Copula-VAE* (Suh and Choi, 2016) in that we use copula to estimate the joint density $p(\mathbf{z})$ rather than the empirical data density $p(\mathbf{x})$.

## 4 Our Approach: Neural Gaussian Copula

### 4.1 Gaussian Copula

In this section, we review the basic concepts of Gaussian copula. Copula is defined as a probability distribution over a high-dimensional unit cube $[0, 1]^d$ whose univariate marginal distributions are uniform on $[0, 1]$. Formally, given a set of uniformaly distributed random variables $U_1, U_2, ..., U_n$, a copula is a joint distribution defined as

$$C(u_1, u_2, ..., u_n) = P(U_1 \leq u_1, ..., U_n \leq u_n)$$

What makes a copula model above so useful is the famous **Sklar's Theorem**. It states that for any joint cumulative distribution function **(CDF)** with a set of random variables $\{x_i\}_1^d$ and marginal CDF $F_i(x_i) = P(X_i \leq x)$, there exists one unique copula function such that the joint CDF is

$$F(x_1, ..., x_d) = C(F_1(x_1), ..., F_d(x_d))$$

By *probability integral transform*, each marginal CDF is a uniform random variable on $[0, 1]$. Hence, the above copula is a valid one. Since for each joint CDF, there is one unique copula function associated with it given a set of marginals, we can easily construct any joint distribution whose marginal univariate distributions are

the ones $F_i(x_i)$ that are given. And, for a given joint distribution, we can also find the corresponding copula which is the CDF function of the given marginals.

A useful representation we can get by **Sklar's Theorem** for a continuous copula is,

$$C(u_1, ...u_d) = F(F_1^-(u_1), ..., F_d^-(u_d))$$

If we further restrict the marginals to be Gaussian, then we can get an expression for Gaussian copula, that is,

$$C_\Sigma(u_1, ..., u_d) = \Phi(\Phi_1^-(u_1), ..., \Phi_d^-(u_d); 0, \Sigma)$$

where $\Phi(\cdot; \Sigma)$ is the CDF of a multivariate Gaussian Distribution $\mathcal{N}(0, \Sigma)$, and $\{\Phi_i^-\}$ is the inverse of a set of marginal Gaussian CDF.

To calculate the joint density of a copula function, we take the derivative with respect to random variables $u$ and get

$$\begin{aligned} c_\Sigma(u_1, ..., u_d) &= \frac{\partial C_\Sigma(u_1, ..., u_d)}{\partial u_1 \cdots u_d} \\ &= \frac{\partial C_\Sigma(u_1, ..., u_d)}{\partial q_1 \cdots q_d} \prod_{i=1}^{d} \frac{\partial q_i}{\partial u_i} \\ &= (\prod_{i=1}^{d} \sigma_i)|\Sigma^{-1/2}|exp(-\frac{1}{2} q^T M q) \end{aligned}$$

where $M = \Sigma^- - diag(\Sigma)^-$, and $q_i = \Phi_i^-(u_i)$.

Then, if the joint density $p(x_1, ..., x_d)$ has a Gaussian form, it can be expressed by a copula density and its marginal densities, that is,

$$\begin{aligned} p(x_1, ..., x_d) &= \frac{\partial F(\cdot)}{\partial x_1 \cdots x_d} \\ &= \frac{\partial C_\Sigma(\cdot)}{\partial u_1 \cdots u_d} \prod \frac{\partial u_i}{\partial x_i} \\ &= c_\Sigma(u_1, ..., u_d) \prod_i p(x_i) \end{aligned}$$

Therefore, we can decompose the problem of estimating the joint density into two smaller subproblems: one is the estimation for the marginals; the other is the estimation for the copula density function $c_\Sigma$. In many cases, we assume independence over random variables due to the intractability of the joint density. For example, in the case of variational inference, we apply Mean-Field assumption which requires the variational distribution family to have factorized form so that we can get a closed-form KL divergence with respect to the prior. This assumption, however, sacrifices the useful dependency relationships over the latent random variables and often leads to training difficulties such as the posterior collapse problem. If we assume the joint posterior of latent variables to be Gaussian, then the above Gaussian copula model can be used to recover the correlation among latent variables which helps obtain a better estimation of the joint posterior density. In the VAE setting, we can already model the marginal independent posterior of latent variables, so the only problem left is how to efficiently estimate the copula density function $c_\Sigma$. In the next section, we introduce a neural parameterized Gaussian copula model, and we provide a way to incorporate it with the reparameterization technique used in VAE.

### 4.2 Neural Gaussian Copula for VI

By Mean-field assumption, we construct a variational family $\mathcal{Q}$ assuming that each member $q \in \mathcal{Q}$ can be factorized,

$$q(\mathbf{z}) = \prod_i^d q(z_i)$$

This assumption, however, loses dependencies over latent codes and hence does not consider the non-factorized form of the true posterior. In this case, as pictured by Figure 2, when we search the optimal $q$*, it will never reach to the true posterior $p$. If we relieve the assumption, the variational family may overlap with the posterior family. However, this is intractable as the Monto Carlo estimator with respect to the objective often has very high variance (Kingma and Welling, 2014). Hence, we need to find a way to make it possible to match the variational posterior with the true posterior while having a simple and tractable objective function so that the gradient estimator of the expectation is simple and precise.

This is where Gaussian Copula comes into the story. Given a factorized posterior, we can construct a Gaussian copula for the joint posterior,

$$q_\phi(\mathbf{z}|x) = c_\Sigma(\Phi_1(z_1), ...\Phi(z_d)) \prod_i^d q_\phi(z_i|x)$$

where $c_\Sigma$ is the Gaussian copula density. If we take the log on both sides, then we have,

$$\log q_\phi(\mathbf{z}|x) = \log c_\Sigma(u_1, ..., u_d) + \sum_i^d \log q_\phi(z_i|x)$$

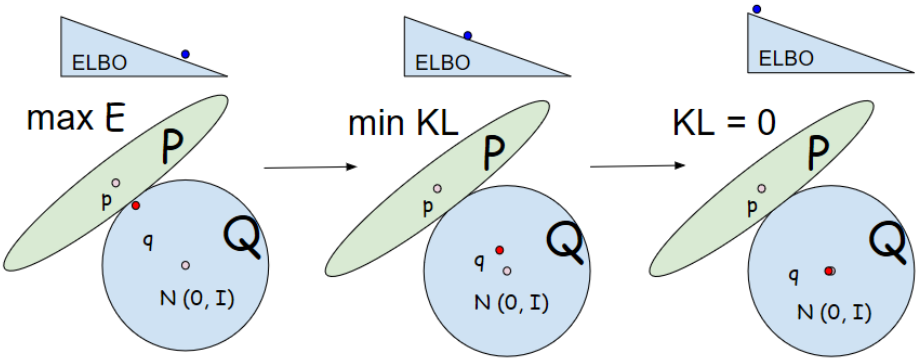


Figure 2: Training stage of VAE. Initially, the model tries to maximize ELBO by maximizing $\mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[p(\mathbf{x}|\mathbf{z})]$. Once $\mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[p(\mathbf{x}|\mathbf{z})]$ is maximized, the model maximizes ELBO by minimizing KL. During this stage, the posterior starts to move closer to the prior. In the final stage, the posterior collapses to the prior. But, the ELBO and $\mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[p(\mathbf{x}|\mathbf{z})]$ are already maximized, which means the model keeps constraining KL and there are not enough gradients to move the posterior away from the prior anymore.

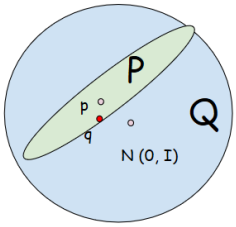


Figure 3: **Ideal** final stage of Copula-VAE. The family of distributions that contains the true posterior is now a subset of the variational family.

Note that the second term on the right hand side is just the factorized log posterior we have in the original VAE model. By reparameterization trick (Kingma and Welling, 2014), latent codes sampled from the posterior are parameterized as a deterministic function of $\mu$ and $\sigma^2$, that is, $\mathbf{z} = \mu + \sigma \cdot \epsilon, \epsilon \sim \mathcal{N}(0, I)$, where $\mu, \sigma^2$ are parameterized by two neural networks whose inputs are the final hidden states of the LSTM encoder. Since $\prod_i q_\phi(z_i|x) = \mathcal{N}(\mu, \sigma^2 I)$, we can compute the sum of log density of posterior by,

$$\sum_i^d \log q_\phi(z_i|x) = -\sum_i^d \log|\sigma_i| - \sum_i^d \frac{(z_i - \mu_i)^2}{2\sigma_i^2} - \frac{d}{2}\log 2\pi$$

Now, to estimate the log copula density $\log c_\Sigma(\cdot)$, we provide a reparameterization method for the copula samples $q \sim C_\Sigma(\Phi_1(q_1), ...\Phi(q_d))$. As suggested by Kingma and Welling (2014); Hoffman et al. (2013), reparameterization is needed as it gives a differentiable, low-variance estimator of the objective function. Here, we parameterize the copula samples with a deterministic function with respect to the Cholesky factor $L$ of its covariance matrix $\Sigma$. We use the fact that for any multivariate Gaussian random variables, a linear transformation of them is also a multivariate Gaussian random variable. Formally, if $X \sim \mathcal{N}(\mu, \Sigma)$, and $Y = AX$, then we must have $Y \sim \mathcal{N}(A\mu, A\Sigma A^T)$. Hence, for a Gaussian copula with the form $c_\Sigma = \mathcal{N}(0, \Sigma)$, we can reparameterize its samples $q$ by,

$$\epsilon \sim \mathcal{N}(0, I)$$
$$q = L \cdot \epsilon$$

It is easy to see that $q = L \cdot \epsilon \sim \mathcal{N}(0, LI^-L^T = \Sigma)$ is indeed a sample from the Gaussian copula model. This is the standard way of sampling from Gaussian distribution with covariance matrix $LL^T$. To ensure **numerical stability** of the above reparameterization and to ensure that the covariance $\Sigma = LL^T$ is positive definite, we provide the following algorithm to parameterize $L$.

**Algorithm 1:** Neural reparameterization of Copula: Cholesky approach

h = LSTM(x)
$w$ = ReLU($W_1 \cdot h + b_1$)
$a$ = Tanh($W_2 \cdot h + b_2$)
$\Sigma = w \cdot I + aa^T$
L = CholeskyFactorization($\Sigma$)

In Algorithm 1, we first parameterize the covariance matrix and then perform a Cholesky factor-

| |
|---|
| the company said it will be sold to the company 's promotional programs and _UNK |
| the company also said it will sell $ n million of soap eggs turning millions of dollars |
| the company said it will be _UNK by the company 's _UNK division n |
| the company said it would n't comment on the suit and its reorganization plan |
| mr . _UNK said the company 's _UNK group is considering a _UNK standstill agreement with the company |
| traders said that the stock market plunge is a _UNK of the market 's rebound in the dow jones industrial average |
| one trader of _UNK said the market is skeptical that the market is n't _UNK by the end of the session |
| the company said it expects to be fully operational by the company 's latest recapitalization |

| |
|---|
| i was excited to try this place out for the first time and i was disappointed . |
| the food was good and the food a few weeks ago , i was in the mood for a _UNK of the _UNK |
| i love this place . i ' ve been here a few times and i ' m not sure why i ' ve been |
| this place is really good . i ' ve been to the other location many times and it 's very good . |
| i had a great time here . i was n't sure what i was expecting . i had the _UNK and the |
| i have been here a few times and have been here several times . the food is good , but the food is good |
| this place is a great place to go for lunch . i had the chicken and waffles . i had the chicken and the _UNK |
| i really like this place . i love the atmosphere and the food is great . the food is always good . |

Table 1: Qualitative comparison between VAE and our proposed approach. First row: PTB samples generated from prior $p(\mathbf{z})$ by VAE (*upper half*) and copula-VAE (*lower half*). Second row: Yelp samples generated from prior $p(\mathbf{z})$ by VAE (*upper half*) and copula-VAE (*lower half*).

ization (Chen et al., 2008) to get the Cholesky factor $L$. The covariance matrix $\Sigma = w \cdot I + aa^T$ formed in this way is guaranteed to be positive definite. It is worth noting that we do not sample the latent codes from Gaussian copula. In fact, $\mathbf{z}$ still comes from the independent Gaussian distribution. Rather, we get sample $q$ from Gaussian copula $C_\Sigma$ so that we can compute the log copula density term in the following, which will then be used as a regularization term during training, in order to force the learned $\mathbf{z}$ to respect the dependencies among individual dimensions.

Now, to calculate the log copula density, we only need to do,

$$\log c_\Sigma = \sum_i^d \log \sigma_i - 1/2 \log |\Sigma| + 1/2 q^T M q$$

where $M = diag(\Sigma^-) - \Sigma^-$.

To make sure that our model maintains the dependency structure of the latent codes, we seek to maximize both the ELBO and the joint log posterior likelihood $\log q(\mathbf{z}|\mathbf{x})$ during the training. In other words, we maximize the following modified ELBO,

$$\mathcal{L}' = \mathcal{L} + \lambda(\log c_\Sigma(\cdot) + \sum_i^d \log q_\phi(z_i|\mathbf{x}))$$

where $\mathcal{L}$ is the original ELBO. $\lambda$ is the weight of log density of the joint posterior. It controls how good the model is at maintaining the dependency relationships of latent codes. The reparameterization tricks both for $\mathbf{z}$ and $q$ makes the above objective fully differentiable with respect to $\mu, \sigma^2, \Sigma$. Maximizing $\mathcal{L}'$ will then maximize the log input likelihood $\log p(\mathbf{x})$ and the joint posterior log-likelihood $\log q(\mathbf{z}|\mathbf{x})$. If the posterior collapses to the prior and has a factorized form, then the joint posterior likelihood will not be maximized since the joint posterior is unlikely factorized. Therefore, maximizing the joint posterior log-likelihood along with ELBO forces the model to generate readable texts while also considering the dependency structure of the true posterior distribution, which is never factorized.

### 4.3 Evidence Lower Bound

Another interpretation can be seen by taking a look at the prior. If we compose the copula density with the prior, then, like the non-factorized posterior, we can get the non-factorized prior,

$$\log p_\theta(\mathbf{z}) = \log c_\Sigma(u_1, ..., u_d) + \sum_i^d \log p_\theta(z_i)$$

And the corresponding ELBO is,

$$\begin{aligned}\mathcal{L}(\theta; \phi; \mathbf{x}) &= \mathbb{E}_{q(\mathbf{x})}[\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] \\ &\quad - KL(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}))] \\ &= \mathbb{E}_{q(\mathbf{x})}[\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] \\ &\quad - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log q_\phi(\mathbf{z}|\mathbf{x}) - \log p(\mathbf{z})] \\ &\quad + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log c_\Sigma(u_1, ..., u_d)]]\end{aligned}$$

Like Normalizing flow (Rezende and Mohamed, 2015), maximizing the log copula density will then learns a more flexible prior other than a standard Gaussian. The dependency among each $z_i$ is

then restored since the KL term will push the posterior to this more complex prior.

We argue that relieving the Mean-field assumption by maintaining the dependency structure can avert the posterior collapse problem. As shown in Figure 2, during the training stage of original VAE, if kl-annealing (Bowman et al., 2015) is used, the model first seeks to maximize the expectation $\mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[p(\mathbf{x}|\mathbf{z})]$. Then, since $q(\mathbf{z}|\mathbf{x})$ can never reach to the true $p(\mathbf{z}|\mathbf{x})$, $q$ will reach to a boundary and then the expectation can no longer increase. During this stage, the model starts to maximize the ELBO by minimizing the KL divergence. Since the expectation is maximized and can no longer leverage KL, the posterior will collapse to the prior and there is not sufficient gradient to move it away since ELBO is already maximized. On the other hand, if we introduce a copula model to help maintain the dependency structure of the true posterior by maximizing the joint posterior likelihood, then, in the ideal case, the variational family can approximate distributions of any forms since it is now not restricted to be factorized, and therefore it is more likely for $q$ in Figure 3 to be closer to the true posterior $p$. In this case, the $\mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[p(\mathbf{x}|\mathbf{z})]$ can be higher since now we have latent codes sampled from a more accurate posterior, and then this expectation will be able to leverage the decrease of KL even in the final training stage.

# 5 Experimental Results

## 5.1 Datasets

| Data | Train | Valid | Test | Vocab |
|---|---|---|---|---|
| Yelp13 | 62522 | 7773 | 8671 | 15K |
| PTB | 42068 | 3370 | 3761 | 10K |
| Yahoo | 100K | 10K | 10K | 20K |

Table 2: Size and vocabulary size for each dataset.

In the paper, we use Penn Tree (Marcus et al., 1993), Yahoo Answers (Xu and Durrett, 2018; Yang et al., 2017), and Yelp 13 reviews (Xu et al., 2016) to test our model performance over variational language modeling tasks. We use these three large datasets as they are widely used in all other variational language modeling approaches (Bowman et al., 2015; Yang et al., 2017; Xu and Durrett, 2018; Xiao et al., 2018; He et al., 2018; Kim et al., 2018). Table 2 shows the statistics, vocabulary size, and number of samples in Train/Validation/Test for each dataset.

| Model | NLL | KL | PPL |
|---|---|---|---|
| LSTM-LM (Yang et al., 2017) | 116.2 | - | 104.2 |
| VAE (Bowman et al., 2015) | 105.2 | 1.74 | 121 |
| vmf-VAE (Xu and Durrett, 2018) | 96.0 | 5.7 | 79.6 |
| VAE-NF | 96.8 | 0.87 | 82.9 |
| non-diag-VAE | 105.1 | 4.9 | 121.0 |
| **copula-VAE(cho)** $\lambda = 0.4$ | **92.2** | **7.3** | **67.2** |

Table 3: Variational language modeling on PTB

| Model | NLL | KL | PPL |
|---|---|---|---|
| VAE (Bowman et al., 2015) | 197.1 | 0.03 | 58.2 |
| vmf-VAE (Xu and Durrett, 2018) | 198.0 | 6.4 | 59.3 |
| VAE-NF | 200.4 | 0.1 | 62.5 |
| non-diag-VAE | 198.3 | 4.7 | 59.7 |
| **copula-VAE(cho)** $\lambda = 0.5$ | **187.7** | **10.0** | **48.0** |

Table 4: Variational language modeling on Yelp Reviews 13

| Model | NLL | KL | PPL |
|---|---|---|---|
| VAE (Bowman et al., 2015) | 351.6 | 0.3 | 81.6 |
| vmf-VAE (Xu and Durrett, 2018) | 359.3 | 17.9 | 89.9 |
| VAE-NF | 353.8 | 0.1 | 83.0 |
| lagging-VAE (He et al., 2018) | **326.6** | **6.7** | **64.9** |
| non-diag-VAE | 352.2 | 5.7 | 82.3 |
| **copula-VAE(cho)** $\lambda = 0.5$ | 344.2 | 15.4 | 74.4 |

Table 5: Variational language modeling on Yahoo

## 5.2 Experimental Setup

We set up a similar experimental condition as in (Bowman et al., 2015; Xiao et al., 2018; Yang et al., 2017; Xu and Durrett, 2018). We use LSTM as our encoder-decoder model, where the number of hidden units for each hidden state is set to 512. The word embedding size is 512. And, the number of dimension for latent codes is set to 32. For both encoder and decoder, we use a *dropout* layer for the initial input, whose dropout rate is $\alpha = 0.5$. Then, for inference, we pass the final hidden state to a linear layer following a *Batch Normalizing* (Ioffe and Szegedy, 2015) layer to get reparameterized samples from $\prod_i q(z_i|x)$ and from Gaussian copula $C_\Sigma$. For the training stage, the maximum vocabulary size for all inputs are set to 20000, and the maximum sequence length is set to 200. Batch size is set to 32, and we train for 30 epochs for each dataset, where we use the *Adam stochastic optimization* (Kingma and Ba, 2014) whose learning rate is $r = 10^{-3}$.

We use *kl annealing* (Bowman et al., 2015) during training. We also observe that the weight of *log copula* density is the most important factor which determines whether our model avoids the posterior collapse problem. We hyper tuned this parameter in order to find the optimal one.

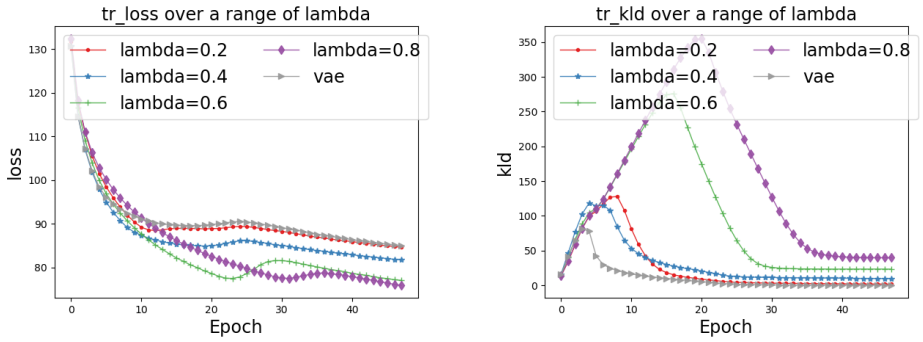


(a) train loss




(b) train kld

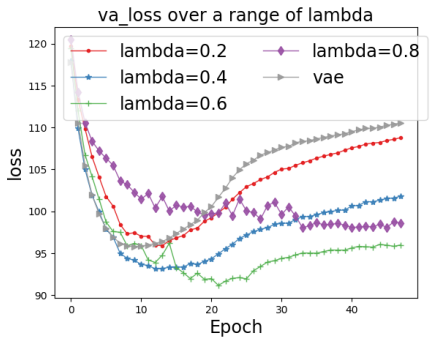


(c) validation loss

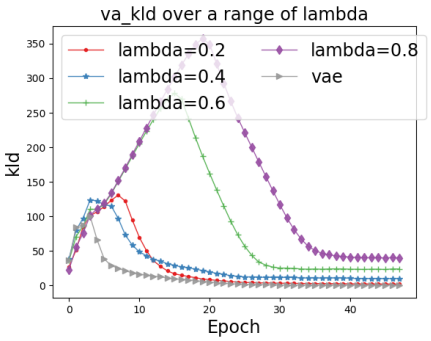


(d) validation kld

Figure 4: Training and Validation KL divergence and sequential loss for PTB using Cholesky Neural Copula with different copula density weights $\lambda$. It is obvious that the optimal $\lambda$ plays a huge role in alleviating the posterior collapse problem and does not result in overfitting as VAE does. Models are trained using 1-layer LSTM with 200 hidden units for Encoder/Decoder, in which the embedding size is 200 and the latent dimension is 32.

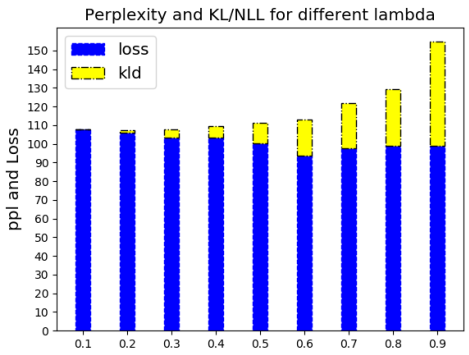


Figure 5: Reconstruction loss, KL divergence, and PPL (sum of loss and KL) on PTB. When we gradually increase $\lambda$, the KL divergence increases and the test reconstruction will decrease.

### 5.3 Language Modeling Comparison

#### 5.3.1 Comparison with other Variational models

We compare the variational language modeling results over three datasets. We show the results for Negative log-likelihood (NLL), KL divergence, and sample perplexity (PPL) for each model on these datasets. NLL is approximated by the *evidence lower bound*.

First, we observe that kl-annealing does not help alleviate the posterior collapse problem when it comes to larger datasets such as Yelp, but the problem is solved if we can maintain the latent code's dependencies by maximizing the copula likelihood when we maximize the ELBO. We also observe that the weight $\lambda$ of log copula density affects results dramatically. All $\lambda$ produce competitive results compared with other methods. Here, we provide the numbers for those weights $\lambda$ that produce the lowest PPL. For PTB, copula-VAE achieves the lowest sample perplexity, best NLL approximation, and do not result in posterior collapse when $\lambda = 0.4$. For Yelp, the lowest sample perplexity is achieved when $\lambda = 0.5$.

We also compare with VAE models trained with normalizing flows (Rezende and Mohamed, 2015). We observed that our model is superior to VAE based on flows. It is worth noting that Wasserstein Autoencoder trained with Normalizing flow (Wang and Wang, 2019) achieves the lowest PPL 66 on PTB, and 41 on Yelp. However, the

problem of designing flexible normalizing flow is orthogonal to our research.

#### 5.3.2 Generation

Table 1 presents the results of text generation task. We first randomly sample $\mathbf{z}$ from $p(\mathbf{z})$, and then feed it into the decoder $p(\mathbf{x}|\mathbf{z})$ to generate text using greedy decoding. We can tell whether a model suffers from posterior collapse by examining the diversity of the generated sentences. The original VAE tends to generate the same type of sequences for different $\mathbf{z}$. This is very obvious in PTB where the posterior of the original VAE collapse to the prior completely. Copula-VAE, however, does not have this kind of issue and can always generate a diverse set of texts.

### 5.4 Hyperparameter-Tuning: Copula weights play a huge role in the training of VAE

In this section, we investigate the influence of log copula weight $\lambda$ over training. From Figure 5, we observe that our model performance is very sensitive to the value of $\lambda$. We can see that when $\lambda$ is small, the log copula density contributes a small part to the objective, and therefore does not help to maintain dependencies over latent codes. In this case, the model performs like the original VAE, where the KL divergence becomes zero at the end. When we increase $\lambda$, test KL becomes larger and test reconstruction loss becomes smaller.

This phenomenon is also observed in validation datasets, as shown in Figure 4. The training PPL is monotonically decreasing in general. However, when $\lambda$ is small and the dependency relationships over latent codes are lost, the model quickly overfits, as the KL divergence quickly becomes zero and the validation loss starts to increase. This further confirms what we showed in Figure 2. For original VAE models, the model first maximizes $\mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[p(\mathbf{x}|\mathbf{z})]$ which results in the decrease of both train and validation loss. Then, as $q(\mathbf{z}|\mathbf{x})$ can never match to the true posterior, $\mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[p(\mathbf{x}|\mathbf{z})]$ reaches to its ceiling which then results in the decrease of KL as it is needed to maximize the ELBO. During this stage, the LSTM decoder starts to learn how to generate texts with standard Gaussian latent variables which then causes the increase of validation loss. On the other hand, if we gradually increase the contribution of copula density by increasing the $\lambda$, the model is able to maintain the dependencies of latent codes and hence the structure of the true posterior. In this case, $\mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[p(\mathbf{x}|\mathbf{z})]$ will be much higher and will leverage the decrease of KL. In this case, the decoder is forced to generate texts from non-standard Gaussian latent codes. Therefore, the validation loss also decreases monotonically in general.

One major drawback of our model is the amount of training time, which is 5 times longer than the original VAE method. In terms of performance, copula-VAE achieves the lowest reconstruction loss when $\lambda = 0.6$. It is clear that from Figure 5 that increasing $\lambda$ will result in larger KL divergence.

## 6 Conclusion

In this paper, we introduce **Copula-VAE** with Cholesky reparameterization method for Gaussian Copula. This approach averts **Posterior Collapse** by using Gaussian copula to maintain the dependency structure of the true posterior. Our results show that Copula-VAE significantly improves the language modeling results of other VAEs.

## References


Tim Bedford, Roger M Cooke, et al. 2002. Vines–a new graphical model for dependent random variables. *The Annals of Statistics*, 30(4):1031–1068.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *CoNLL*.

Lu Chen, Vijay P Singh, Shenglian Guo, Ashok K Mishra, and Jing Guo. 2012. Drought analysis using copulas. *Journal of Hydrologic Engineering*, 18(7):797–808.

Yanqing Chen, Timothy A Davis, William W Hager, and Sivasankaran Rajamanickam. 2008. Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate. *ACM Transactions on Mathematical Software (TOMS)*, 35(3):22.

Barbara Choroś, Rustam Ibragimov, and Elena Permiakova. 2010. Copula estimation. In *Copula theory and its applications*, pages 77–91. Springer.

Claudia Czado. 2010. Pair-copula constructions of multivariate copulas. In *Copula theory and its applications*, pages 93–109. Springer.

Rüdiger Frey, Alexander J McNeil, and Mark Nyfeler. 2001. Copulas and credit models. *Risk*, 10(111114.10).

Izrail Moiseevitch Gelfand, Richard A Silverman, et al. 2000. *Calculus of variations*. Courier Corporation.

Edward I George and Robert E McCulloch. 1993. Variable selection via gibbs sampling. *Journal of the American Statistical Association*, 88(423):881–889.

Walter R Gilks, Sylvia Richardson, and David Spiegelhalter. 1995. *Markov chain Monte Carlo in practice*. Chapman and Hall/CRC.

Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. 2015. Draw: A recurrent neural network for image generation. *ICML*.

Junxian He, Daniel Spokoyny, Graham Neubig, and Taylor Berg-Kirkpatrick. 2018. Lagging inference networks and posterior collapse in variational autoencoders.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. 2013. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. *ICLR*.

Piotr Jaworski, Fabrizio Durante, Wolfgang Karl Hardle, and Tomasz Rychlik. 2010. *Copula theory and its applications*, volume 198. Springer.

Harry Joe and Dorota Kurowicka. 2011. *Dependence modeling: vine copula handbook*. World Scientific.

Yoon Kim, Sam Wiseman, Andrew Miller, David Sontag, and Alexander Rush. 2018. Semi-amortized variational autoencoders. In *International Conference on Machine Learning*, pages 2683–2692.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. *stat*, 1050:10.

Erik Kole, Kees Koedijk, and Marno Verbeek. 2007. Selecting copulas for risk management. *Journal of Banking & Finance*, 31(8):2405–2423.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Alexander J McNeil, Rüdiger Frey, Paul Embrechts, et al. 2005. *Quantitative risk management: Concepts, techniques and tools*, volume 3. Princeton university press Princeton.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *ICML*.

Roger B Nelsen. 2007. *An introduction to copulas*. Springer Science & Business Media.

Danilo Jimenez Rezende and Shakir Mohamed. 2015. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *NIPS*.

Suwon Suh and Seungjin Choi. 2016. Gaussian copula variational autoencoders for mixed data. *arXiv preprint arXiv:1604.04960*.

Hideatsu Tsukahara. 2005. Semiparametric estimation in copula models. *Canadian Journal of Statistics*, 33(3):357–375.

Martin J Wainwright, Michael I Jordan, et al. 2008. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305.

Prince Zizhuang Wang and William Yang Wang. 2019. Riemannian normalizing flow on variational wasserstein autoencoder for text modeling. *arXiv preprint arXiv:1904.02399*.

William Yang Wang and Zhenhao Hua. 2014. A semiparametric gaussian copula regression model for predicting financial risks from earnings calls. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1155–1165.

William Yang Wang and Miaomiao Wen. 2015. I can has cheezburger? a nonparanormal approach to combining textual and visual information for predicting and generating popular meme descriptions. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Denver, CO, USA. ACL.

Yijun Xiao, Tiancheng Zhao, and William Yang Wang. 2018. Dirichlet variational autoencoder for text modeling. *arXiv preprint arXiv:1811.00135*.

Jiacheng Xu, Danlu Chen, Xipeng Qiu, and Xuangjing Huang. 2016. Cached long short-term memory neural networks for document-level sentiment classification. *EMNLP*.

Jiacheng Xu and Greg Durrett. 2018. Spherical latent spaces for stable variational autoencoders. *EMNLP*.

Peter Xue-Kun Song. 2000. Multivariate dispersion models generated from gaussian copula. *Scandinavian Journal of Statistics*, 27(2):305–320.

Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. 2017. Improved variational autoencoders for text modeling using dilated convolutions. *ICML*.

LSVP Zhang and VP Singh. 2006. Bivariate flood frequency analysis using the copula method. *Journal of hydrologic engineering*, 11(2):150–164.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. *ACL*.

# Transformer Dissection: An Unified Understanding for Transformer's Attention via the Lens of Kernel

**Yao-Hung Hubert Tsai**[1] **Shaojie Bai**[1] **Makoto Yamada**[34]
**Louis-Philippe Morency**[2] **Ruslan Salakhutdinov**[1]
{[1]Machine Learning Department,[2]Language Technology Institute}, Carnegie Mellon University
[3]Kyoto University [4]RIKEN AIP
{yaohungt, shaojieb, morency, rsalakhu}@cs.cmu.edu, myamada@i.kyoto-u.ac.jp

## Abstract

Transformer is a powerful architecture that achieves superior performance on various sequence learning tasks, including neural machine translation, language understanding, and sequence prediction. At the core of the Transformer is the attention mechanism, which concurrently processes all inputs in the streams. In this paper, we present a new formulation of attention via the lens of the kernel. To be more precise, we realize that the attention can be seen as applying kernel smoother over the inputs with the kernel scores being the similarities between inputs. This new formulation gives us a better way to understand individual components of the Transformer's attention, such as the better way to integrate the positional embedding. Another important advantage of our kernel-based formulation is that it paves the way to a larger space of composing Transformer's attention. As an example, we propose a new variant of Transformer's attention which models the input as a product of symmetric kernels. This approach achieves competitive performance to the current state of the art model with less computation. In our experiments, we empirically study different kernel construction strategies on two widely used tasks: neural machine translation and sequence prediction.

## 1 Introduction

Transformer (Vaswani et al., 2017) is a relative new architecture which outperforms traditional deep learning models such as Recurrent Neural Networks (RNNs) (Sutskever et al., 2014) and Temporal Convolutional Networks (TCNs) (Bai et al., 2018) for sequence modeling tasks across neural machine translations (Vaswani et al., 2017), language understanding (Devlin et al., 2018), sequence prediction (Dai et al., 2019), image generation (Child et al., 2019), video activity classification (Wang et al., 2018), music generation (Huang et al., 2018a), and multimodal sentiment analysis (Tsai et al., 2019a). Instead of performing recurrence (e.g., RNN) or convolution (e.g., TCN) over the sequences, Transformer is a feed-forward model that concurrently processes the entire sequence. At the core of the Transformer is its attention mechanism, which is proposed to integrate the dependencies between the inputs. There are up to three types of attention within the full Transformer model as exemplified with neural machine translation application (Vaswani et al., 2017): 1) Encoder self-attention considers the source sentence as input, generating a sequence of encoded representations, where each encoded token has a global dependency with other tokens in the input sequence. 2) Decoder self-attention considers the target sentence (e.g., predicted target sequence for translation) as input, generating a sequence of decoded representations[1], where each decoded token depends on previous decoded tokens. 3) Decoder-encoder attention considers both encoded and decoded sequences, generating a sequence with the same length as the decoded sequence. It should be noted that some applications has only the decoder self-attention such as sequence prediction (Dai et al., 2019). In all cases, the Transformer's attentions follow the same general mechanism.

At the high level, the attention can be seen as a weighted combination of the input sequence, where the weights are determined by the similarities between elements of the input sequence. We note that this operation is order-agnostic to the permutation in the input sequence (order is encoded with extra positional embedding (Vaswani et al., 2017; Shaw et al., 2018; Dai et al., 2019)). The

[1]The generated sequence can be regarded as a translated sequence (i.e., translating from the encoded sequence), where each generated token depends on all tokens in the encoded sequence.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4344–4353,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

above observation inspires us to connect Transformer's attention to kernel learning (Scholkopf and Smola, 2001): they both concurrently and order-agnostically process all inputs by calculating the similarity between the inputs. Therefore, in the paper, we present a new formulation for Transformer's attention via the lens of kernel. To be more precise, the new formulation can be interpreted as a kernel smoother (Wasserman, 2006) over the inputs in a sequence, where the kernel measures how similar two different inputs are. The main advantage of connecting attention to kernel is that it opens up a new family of attention mechanisms that can relate to the well-established literature in kernel learning (Scholkopf and Smola, 2001). As a result, we develop a new variant of attention which simply considers a product of symmetric kernels when modeling non-positional and positional embedding.

Furthermore, our proposed formulation highlights naturally the main components of Transformer's attention, enabling a better understanding of this mechanism: recent variants of Transformers (Shaw et al., 2018; Huang et al., 2018b; Dai et al., 2019; Child et al., 2019; Lee et al., 2018; Wang et al., 2018; Tsai et al., 2019a) can be expressed through these individual components. Among all the components, we argue that the most important one is the construction of the kernel function. We empirically study multiple kernel forms and the ways to integrate positional embedding in neural machine translation (NMT) using IWSLT'14 German-English (De-En) dataset (Edunov et al., 2017) and sequence prediction (SP) using WikiText-103 dataset (Merity et al., 2016).

## 2 Attention

This section aims at providing an understanding of attention in Transformer via the lens of kernel. The inspiration for connecting the kernel (Scholkopf and Smola, 2001) and attention instantiates from the observation: both operations concurrently processes all inputs and calculate the similarity between the inputs. We first introduce the background (i.e., the original formulation) of attention and then provide a new reformulation within the class of kernel smoothers (Wasserman, 2006). Next, we show that this new formulation allows us to explore new family of attention while at the same time offering a framework to categorize previous attention variants (Vaswani et al., 2017; Shaw et al., 2018; Huang et al., 2018b; Dai et al., 2019; Child et al., 2019; Lee et al., 2018; Wang et al., 2018; Tsai et al., 2019a). Last, we present a new form of attention, which requires fewer parameters and empirically reaches competitive performance as the state-of-the-art models.

For notation, we use lowercase representing a vector (e.g., $x$), bold lowercase representing a matrix (e.g., $\mathbf{x}$), calligraphy letter denoting a space (e.g., $\mathcal{X}$), and $S$ denoting a set. To relate the notations in sequence to sequence learning (Vaswani et al., 2017), $x$ represents a specific element of a sequence, $\mathbf{x} = [x_1, x_2, \cdots, x_T]$ denotes a sequence of features, $S_{\mathbf{x}} = \{x_{\text{exp}}, x_2, \cdots, x_T\}$ represents the set with its elements being the features in sequence $\mathbf{x}$, and we refer the space of set $S_{\mathbf{x}}$ as $\mathcal{S}$.

### 2.1 Technical Background

Unlike recurrent computation (Sutskever et al., 2014) (i.e., RNNs) and temporal convolutional computation (Bai et al., 2018) (i.e., TCNs), Transformer's attention is an *order-agnostic* operation given the order in the inputs (Vaswani et al., 2017). Hence, in the presentation of the paper, we consider the inputs as a set instead of a sequence. When viewing sequence as a set, we lose the temporal (positional) information in inputs which is often crucial for sequence modeling (Sutskever et al., 2014). As a result, Transformer (Vaswani et al., 2017) introduced positional embedding to indicate the positional relation for the inputs. Formally, a sequence $\mathbf{x} = [x_1, x_2, \cdots, x_T]$ defines each element as $x_i = (f_i, t_i)$ with $f_i \in \mathcal{F}$ being the non-temporal feature at time $i$ and $t_i \in \mathcal{T}$ as an temporal feature (or we called it positional embedding). Note that $f_i$ can be the word representation (in neural machine translation (Vaswani et al., 2017)), a pixel in a frame (in video activity recognition (Wang et al., 2018)), or a music unit (in music generation (Huang et al., 2018b)). $t_i$ can be a mixture of sine and cosine functions (Vaswani et al., 2017) or parameters that can be learned during back-propagation (Dai et al., 2019; Ott et al., 2019). The feature vector are defined over a joint space $\mathcal{X} := (\mathcal{F} \times \mathcal{T})$. The resulting permutation-invariant set is: $S_{\mathbf{x}} = \{x_1, x_2, \cdots, x_T\} = \{(f_1, t_1), (f_2, t_2), \cdots, (f_T, t_T)\}$.

Followed the definition by Vaswani et al. (2017), we use queries(q)/keys(k)/values(v) to represent the inputs for the attention. To be

more precise, $x_{\{q/k/v\}}$ is used for denoting a query/key/value data in the query/key/value sequence $\mathbf{x}_{\{q/k/v\}}$ ($x_{\{q/k/v\}} \in S_{\mathbf{x}_{\{q/k/v\}}}$) with $S_{\mathbf{x}_{\{q/k/v\}}}$ being its set representation. We note that the input sequences are the same ($\mathbf{x}_q = \mathbf{x}_k$) for self-attention and are different ($\mathbf{x}_q$ from decoder and $\mathbf{x}_k$ from encoder) for encoder-decoder attention.

Given the introduced notation, the attention mechanism in original Transformer (Vaswani et al., 2017) can be presented as:

$$\begin{aligned}&\text{Attention}(x_q\ ;\ S_{\mathbf{x}_k})\\ &=\text{softmax}\left(\frac{x_q W_q (\mathbf{x}_k W_k)^\top}{\sqrt{d_k}}\right)\mathbf{x}_k W_v\end{aligned} \tag{1}$$

with $x_q = f_q + t_q$, $\mathbf{x}_k = \mathbf{f}_k + \mathbf{t}_k$, $W_{q/k/v}$ being the weight, and $d_k$ being the feature dimension of $\mathbf{x}_k W_k$. Decoder self-attention further introduces a mask to block the visibility of elements in $S_{\mathbf{x}_k}$ to $x_q$. Particularly, decoder self-attention considers the decoded sequence as inputs ($\mathbf{x}_k = \mathbf{x}_q$), where the decoded token at time $t$ is not allowed to access the future decoded tokens (i.e., tokens decoded at time greater than $t$). On the contrary, encoder self-attention and decoder-encoder attention consider no additional mask to Eq. (1).

Recent work (Shaw et al., 2018; Dai et al., 2019; Huang et al., 2018b; Child et al., 2019; Lee et al., 2018; Parmar et al., 2018; Tsai et al., 2019a) proposed modifications to the Transformer for the purpose of better modeling inputs positional relation (Shaw et al., 2018; Huang et al., 2018b; Dai et al., 2019), appending additional keys in $S_{\mathbf{x}_k}$ (Dai et al., 2019), modifying the mask applied to Eq. (1) (Child et al., 2019), or applying to distinct feature types (Lee et al., 2018; Parmar et al., 2018; Tsai et al., 2019a). These works adopt different designs of attention as comparing to the original form (Eq. (1)). In our paper, we aim at providing an unified view via the lens of kernel.

### 2.2 Reformulation via the Lens of Kernel

We now provide the intuition to reformulate Eq. (1) via the lens of kernel. First, the softmax function can be realized as a probability function for $x_q$ observing the keys $\{x_k\}$s in $S_{\mathbf{x}_k}$ ($S_{\mathbf{x}_k}$ is the set representation of sequence $\mathbf{x}_k$). The probability is determined by the dot product between $x_q$ and $x_k$ with additional mappings $W_q/W_k$ and scaling by $d_k$, which we note the dot-product operation is an instance of kernel function. We also introduce a set filtering function $M(x_q, S_{\mathbf{x}_k}) : \mathcal{X} \times \mathcal{S} \to \mathcal{S}$ which returns a set with its elements that operate with (or are connected/visible to) $x_q$. The filtering function $M(\cdot,\cdot)$ plays as the role of the mask in decoder self-attention (Vaswani et al., 2017). Putting these altogether, we re-represent Eq. (1) into the following definition.

**Definition 1.** *Given a non-negative kernel function $k(\cdot,\cdot) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$, a set filtering function $M(\cdot,\cdot) : \mathcal{X} \times \mathcal{S} \to \mathcal{S}$, and a value function $v(\cdot) : \mathcal{X} \to \mathcal{Y}$, the Attention function taking the input of a query feature $x_q \in \mathcal{X}$ is defined as*

$$\begin{aligned}&\text{Attention}\Big(x_q\ ;\ M(x_q, S_{\mathbf{x}_k})\Big)\\ &=\sum_{x_k \in M(x_q, S_{\mathbf{x}_k})} \frac{k(x_q, x_k)}{\sum_{x_k{}' \in M(x_q, S_{\mathbf{x}_k})} k(x_q, x_k{}')} v(x_k).\end{aligned} \tag{2}$$

The Definition 1 is a class of linear smoothers (Wasserman, 2006) with kernel smoothing:

$$\begin{aligned}&\sum_{x_k \in M(x_q, S_{\mathbf{x}_k})} \frac{k(x_q, x_k)}{\sum_{x_k{}' \in M(x_q, S_{\mathbf{x}_k})} k(x_q, x_k{}')} v(x_k)\\ &= \mathbb{E}_{p(x_k|x_q)}\big[v(x_k)\big],\end{aligned}$$

where $v(x_k)$ outputs the "values" and $p(x_k|x_q) = \frac{k(x_q,x_k)}{\sum_{x_k{}' \in M(x_q, S_{\mathbf{x}_k})} k(x_q,x_k{}')}$ is a probability function depends on $k$ and $N$ when $k(\cdot,\cdot)$ is always positive. In the prior work (Vaswani et al., 2017), $k(x_q, x_k) = \exp\left(\langle x_q W_q, x_k W_k\rangle / \sqrt{d_k}\right)$ and $v(x_k) = x_k W_v$. Note that the kernel form $k(x_q, x_k)$ in the original Transformer (Vaswani et al., 2017) is a asymmetric exponential kernel with additional mapping $W_q$ and $W_k$ (Wilson et al., 2016; Li et al., 2017)[2].

The new formulation defines a larger space for composing attention by manipulating its individual components, and at the same time it is able to categorize different variants of attention in prior work (Shaw et al., 2018; Huang et al., 2018b; Dai et al., 2019; Child et al., 2019; Lee et al., 2018; Wang et al., 2018; Tsai et al., 2019a). In the following, we study these components by dissecting Eq. (2) into: 1) kernel feature space $\mathcal{X}$, 2)

[2]We note that rigorous definition of kernel function (Scholkopf and Smola, 2001) requires the kernel to be semi-positive definite and symmetric. While in the paper, the discussion on kernel allows it to be non-semi-positive definite and asymmetric. In Section 3, we will examine the kernels which are semi-positive and symmetric.

kernel construction $k(\cdot,\cdot)$, 3) value function $v(\cdot)$, and 4) set filtering function $M(\cdot,\cdot)$.

#### 2.2.1 Kernel Feature Space $\mathcal{X}$

In Eq. (2), to construct a kernel on $\mathcal{X}$, the first thing is to identify the kernel feature space $\mathcal{X}$. In addition to modeling sequences like word sentences (Vaswani et al., 2017) or music signals (Huang et al., 2018b), the Transformer can also be applied to images (Parmar et al., 2018), sets (Lee et al., 2018), and multimodal sequences (Tsai et al., 2019a). Due to distinct data types, these applications admit various kernel feature space:

*(i) Sequence Transformer (Vaswani et al., 2017; Dai et al., 2019):*

$$\mathcal{X} := (\mathcal{F} \times \mathcal{T})$$

with $\mathcal{F}$ being non-positional feature space and $\mathcal{T}$ being the positional embedding space of the position in the sequence.

*(ii) Image Transformer (Parmar et al., 2018):*

$$\mathcal{X} := (\mathcal{F} \times \mathcal{H} \times \mathcal{W})$$

with $\mathcal{F}$ being non-positional feature space, $\mathcal{H}$ being the positional space of the height in an image, and $\mathcal{W}$ being the positional space of the width in an image.

*(iii) Set Transformer (Lee et al., 2018) and Non-Local Neural Networks (Wang et al., 2018):*

$$\mathcal{X} := (\mathcal{F})$$

with no any positional information present.

*(iv) Multimodal Transformer (Tsai et al., 2019a):*

$$\mathcal{X} := (\mathcal{F}^{\ell} \times \mathcal{F}^{v} \times \mathcal{F}^{a} \times \mathcal{T})$$

with $\mathcal{F}^{\ell}$ representing the language feature space, $\mathcal{F}^{v}$ representing the vision feature space, $\mathcal{F}^{a}$ representing the audio feature space, and $\mathcal{T}$ representing the temporal indicator space.

For the rest of the paper, we will focus on the setting for sequence Transformer $\mathcal{X} = (\mathcal{F} \times \mathcal{T})$ and discuss the kernel construction on it.

#### 2.2.2 Kernel Construction and the Role of Positional Embedding $k(\cdot,\cdot)$

The kernel construction on $\mathcal{X} = (\mathcal{F} \times \mathcal{T})$ has distinct design in variants of Transformers (Vaswani et al., 2017; Dai et al., 2019; Huang et al., 2018b; Shaw et al., 2018; Child et al., 2019). Since now the kernel feature space considers a joint space, we will first discuss the kernel construction on $\mathcal{F}$ (the non-positional feature space) and then discuss how different variants integrate the positional embedding (with the positional feature space $\mathcal{T}$) into the kernel.

**Kernel construction on $\mathcal{F}$.** All the work considered the scaled asymmetric exponential kernel with the mapping $W_q$ and $W_k$ (Wilson et al., 2016; Li et al., 2017) for non-positional features $f_q$ and $f_k$:

$$k_{\exp}(f_q, f_k) = \exp\left(\frac{\langle f_q W_q, f_k W_k\rangle}{\sqrt{d_k}}\right). \quad (3)$$

Note that the usage of asymmetric kernel is also commonly used in various machine learning tasks (Yilmaz, 2007; Tsuda, 1999; Kulis et al., 2011), where they observed the kernel form can be flexible and even non-valid (i.e., a kernel that is not symmetric and positive semi-definite). In Section 3, we show that symmetric design of the kernel has similar performance for various sequence learning tasks, and we also examine different kernel choices (i.e., linear, polynomial, and rbf kernel).

**Kernel construction on $\mathcal{X} = (\mathcal{F} \times \mathcal{T})$.** The designs for integrating the positional embedding $t_q$ and $t_k$ are listed in the following.

*(i) Absolute Positional Embedding (Vaswani et al., 2017; Dai et al., 2019; Ott et al., 2019):* For the original Transformer (Vaswani et al., 2017), each $t_i$ is represented by a vector with each dimension being sine or cosine functions. For learned positional embedding (Dai et al., 2019; Ott et al., 2019), each $t_i$ is a learned parameter and is fixed for the same position for different sequences. These works defines the feature space as the direct sum of its temporal and non-temporal space: $\mathcal{X} = \mathcal{F} \oplus \mathcal{T}$. Via the lens of kernel, the kernel similarity is defined as

$$k\Big(x_q, x_k\Big) := k_{\exp}\Big(f_q + t_q, f_k + t_k\Big). \quad (4)$$

*(ii) Relative Positional Embedding in Transformer-XL (Dai et al., 2019):* $t$ represents the indicator of the position in the sequence, and the kernel is chosen to be asymmetric of mixing sine and cosine functions:

$$k\Big(x_q, x_k\Big) := k_{\exp}\Big(f_q, f_k\Big) \cdot k_{f_q}\Big(t_q, t_k\Big) \quad (5)$$

with $k_{f_q}\Big(t_q, t_k\Big)$ being an asymmetric kernel with coefficients inferred by $f_q$: $\log k_{f_q}\Big(t_q, t_k\Big) = \sum_{p=0}^{\lfloor d_k/2 \rfloor - 1} c_{2p} \sin(\frac{t_q - t_k}{10000^{\frac{2p}{512}}}) + c_{2p+1} \cos(\frac{t_q - t_k}{10000^{\frac{2p}{512}}})$ with $[c_0, \cdots, c_{d_k-1}] = f_q W_q W_R$ where $W_R$ is an learned weight matrix. We refer readers to Dai et al. (2019) for more details.

*(iii) Relative Positional Embedding of Shaw et al. (2018) and Music Transformer (Huang et al., 2018b):* $t.$ represents the indicator of the position in the sequence, and the kernel is modified to be indexed by a look-up table:

$$k\Big(x_q, x_k\Big) := L_{t_q - t_k, f_q} \cdot k_{\exp}\Big(f_q, f_k\Big), \quad (6)$$

where $L_{t_q - t_k, f_q} = \exp(f_q W_q a_{t_q - t_k})$ with $a.$ being a learnable matrix having matrix width to be the length of the sequence. We refer readers to Shaw et al. (2018) for more details.

Dai et al. (2019) showed that the way to integrate positional embedding is better through Eq. (5) than through Eq. (6) and is better through Eq. (6) than through Eq. (4). We argue the reason is that if viewing $f_i$ and $t_i$ as two distinct spaces $\Big(\mathcal{X} := (\mathcal{F} \times \mathcal{T})\Big)$, the direct sum $x_i = f_i + t_i$ may not be optimal when considering the kernel score between $x_q$ and $x_k$. In contrast, Eq. (5) represents the kernel as a product of two kernels (one for $f_i$ and another for $t_i$), which is able to capture the similarities for both temporal and non-temporal components.

#### 2.2.3 Value Function $v(\cdot)$

The current Transformers consider two different value function construction:

*(i) Original Transformer (Vaswani et al., 2017) and Sparse Transformer (Child et al., 2019):*

$$v(x_k) = v((f_k, t_k)) := (f_k + t_k) W_v. \quad (7)$$

*(ii) Transformer-XL (Dai et al., 2019), Music Transformer (Huang et al., 2018b), Self-Attention with Relative Positional Embedding (Shaw et al., 2018):*

$$v(x_k) = v((f_k, t_k)) := f_k W_v. \quad (8)$$

Compared Eq. (7) to Eq. (8), Eq. (7) takes the positional embedding into account for constructing the value function. In Section 3, we empirically observe that constructing value function with Eq. (8) constantly outperforms the construction with Eq. (7), which suggests that we do not need positional embedding for value function.

#### 2.2.4 Set Filtering Function $M(\cdot, \cdot)$

In Eq. (2), the returned set by the set filtering function $M(x_q, S_{\mathbf{x}_k})$ defines how many keys and which keys are operating with $x_q$. In the following, we itemize the corresponding designs for the variants in Transformers:

*(i) Encoder Self-Attention in original Transformer (Vaswani et al., 2017):* For each query $x_q$ in the encoded sequence, $M(x_q, S_{\mathbf{x}_k}) = S_{\mathbf{x}_k}$ contains the keys being all the tokens in the encoded sequence. Note that encoder self-attention considers $\mathbf{x}_q = \mathbf{x}_k$ with $\mathbf{x}_q$ being the encoded sequence.

*(ii) Encoder-Decoder Attention in original Transformer (Vaswani et al., 2017):* For each query $x_q$ in decoded sequence, $M(x_q, S_{\mathbf{x}_k}) = S_{\mathbf{x}_k}$ contains the keys being all the tokens in the encoded sequence. Note that encode-decoder attention considers $\mathbf{x}_q \neq \mathbf{x}_k$ with $\mathbf{x}_q$ being the decoded sequence and $\mathbf{x}_k$ being the encoded sequence.

*(iii) Decoder Self-Attention in original Transformer (Vaswani et al., 2017):* For each query $x_q$ in the decoded sequence, $M(x_q, S_{\mathbf{x}_k})$ returns a subset of $S_{\mathbf{x}_k}$ ($M(x_q, S_{\mathbf{x}_k}) \subset S_{\mathbf{x}_k}$). Note that decoder self-attention considers $\mathbf{x}_q = \mathbf{x}_k$ with $\mathbf{x}_q$ being the decoded sequence. Since the decoded sequence is the output for previous timestep, the query at position $i$ can only observe the keys being the tokens that are decoded with position $< i$. For convenience, let us define $S_1$ as the set returned by original Transformer (Vaswani et al., 2017) from $M(x_q, S_{\mathbf{x}_k})$, which we will use it later.

*(iv) Decoder Self-Attention in Transformer-XL (Dai et al., 2019):* For each query $x_q$ in the decoded sequence, $M(x_q, S_{\mathbf{x}_k})$ returns a set containing $S_1$ and additional memories ($M(x_q, S_{\mathbf{x}_k}) = S_1 + S_{mem}, M(x_q, S_{\mathbf{x}_k}) \supset S_1$). $S_{mem}$ refers to additional memories.

*(v) Decoder Self-Attention in Sparse Transformer (Child et al., 2019):* For each query $x_q$ in the decoded sentence, $M(x_q, S_{\mathbf{x}_k})$ returns a subset of $S_1$ ($M(x_q, S_{\mathbf{x}_k}) \subset S_1$).

To compare the differences for various designs, we see the computation time is inversely proportional to the number of elements in $M(x_q, S_{\mathbf{x}_k})$. For performance-wise comparisons, Transformer-XL (Dai et al., 2019) showed that, the additional memories in $M(x_q, S_{\mathbf{x}_k})$ are able to capture longer-term dependency than the original Transformer (Vaswani et al., 2017) and hence results in better performance. Sparse Transformer (Child

et al., 2019) showed that although having much fewer elements in $M(x_q, S_{\mathbf{x}_k})$, if the elements are carefully chosen, the attention can still reach the same performance as Transformer-XL (Dai et al., 2019).

### 2.3 Exploring the Design of Attention

So far, we see how Eq. (2) connects to the variants of Transformers. By changing the kernel construction in Section 2.2.2, we can define a larger space for composing attention. In this paper, we present a new form of attention with a kernel that is 1) valid (i.e., a kernel that is symmetric and positive semi-definite) and 2) delicate in the sense of constructing a kernel on a joint space (i.e., $\mathcal{X} = (\mathcal{F} \times \mathcal{T})$):

$$
\begin{aligned}
& k(x_q, x_k) := k_F\Big(f_q, f_k\Big) \cdot k_T\Big(t_q, t_k\Big) \\
& \text{with } k_F(f_q, f_k) = \exp\Big(\frac{\langle f_q W_F, f_k W_F \rangle}{\sqrt{d_k}}\Big) \\
& \text{and } \ k_T(t_q, t_k) = \exp\Big(\frac{\langle t_q W_T, t_k W_T \rangle}{\sqrt{d_k}}\Big),
\end{aligned}
\tag{9}
$$

where $W_F$ and $W_T$ are weight matrices. The new form considers product of kernels with the first kernel measuring similarity between non-temporal features and the second kernel measuring similarity between temporal features. Both kernels are symmetric exponential kernel. Note that $t_i$ here is chosen as the mixture of sine and cosine functions as in the prior work (Vaswani et al., 2017; Ott et al., 2019). In our experiment, we find it reaching competitive performance as comparing to the current state-of-the-art designs (Eq. (5) by Dai et al. (2019)). We fix the size of the weight matrices $W_\cdot$ in Eq. (9) and Eq. (5) which means we save 33% of the parameters in attention from Eq. (9) to Eq. (5) (Eq. (5) has weights $W_Q/W_K/W_R$ and Eq. (9) has weights $W_F/W_T$).

## 3 Experiments

By viewing the attention mechanism with Eq. (2), we aims at answering the following questions regarding the Transformer's designs:

**Q1.** What is the suggested way for incorporating positional embedding in the kernel function?

**Q2.** What forms of kernel are recommended to choose in the attention mechanism? Can we replace the asymmetric kernel with the symmetric version?

**Q3.** Is there any exception that the attention mechanism is not order-agnostic with respect to inputs? If so, can we downplay the role of positional embedding?

**Q4.** Is positional embedding required in value function?

We conduct experiments on neural machine translation (NMT) and sequence prediction (SP) tasks since these two tasks are commonly chosen for studying Transformers (Vaswani et al., 2017; Dai et al., 2019). Note that NMT has three different types of attentions (e.g., encoder self-attention, decoder-encoder attention, decoder self-attention) and SP has only one type of attention (e.g., decoder self-attention). For the choice of datasets, we pick IWSLT'14 German-English (De-En) dataset (Edunov et al., 2017) for NMT and WikiText-103 dataset (Merity et al., 2016) for SP as suggested by Edunov *et al.* (Edunov et al., 2017) and Dai *et al.* (Dai et al., 2019). For fairness of comparisons, we train five random initializations and report test accuracy with the highest validation score. We fix the position-wise operations in Transformer[3] and only change the attention mechanism. Similar to prior work (Vaswani et al., 2017; Dai et al., 2019), we report BLEU score for NMT and perplexity for SP.

### 3.1 Incorporating Positional Embedding

In order to find the best way to integrate positional embedding (PE), we study different PE incorporation in the kernel function $k(\cdot, \cdot)$ in Eq. (2). Referring to Sections 2.2.2 and 2.3, we consider four cases: 1) PE as direct sum in the feature space (see Eq. (4)), 2) PE as a look-up table (see Eq. (6)), 3) PE in product kernel with asymmetric kernel (see Eq. (5)), and 4) PE in product kernel with symmetric kernel (see Eq. (9)). We present the results in Table 1.

First, we see that by having PE as a look-up table, it outperforms the case with having PE as direct-sum in feature space, especially for SP task. Note that the look-up table is indexed by the relative position (i.e., $t_q - t_k$) instead of absolute position. Second, we see that PE in the product kernel proposed by Dai *et al.* (Dai et al., 2019) may not

[3]The computation of Transformer can be categorized into position-wise and inter-positions (i.e., the attention mechanism) operations. Position-wise operations include layer normalization, residual connection, and feed-forward mapping. We refer the readers to Vaswani *et al.* (Vaswani et al., 2017) for more details.

Table 1: Incorporating Positional Embedding (PE). NMT stands for neural machine translation on IWSLT'14 De-En dataset (Edunov et al., 2017) and SP stands for sequence prediction on WikiText-103 dataset (Merity et al., 2016). ↑ means the upper the better and ↓ means the lower the better.

| Approach | PE Incorporation | Kernel Form | NMT (BLEU↑) | SP (Perplexity↓) |
|---|---|---|---|---|
| Vaswani et al. (2017) (Eq. (4)) | Direct-Sum | $k_{\exp}\Big(f_q + t_q, f_k + t_k\Big)$ | 33.98 | 30.97 |
| Shaw et al. (2018) (Eq. (6)) | Look-up Table | $L_{t_q - t_k, f_q} \cdot k_{\exp}\Big(f_q, f_k\Big)$ | 34.12 | 27.56 |
| Dai et al. (2019) (Eq. (5)) | Product Kernel | $k_{\exp}\Big(f_q, f_k\Big) \cdot k_{f_q}\Big(t_q, t_k\Big)$ | 33.62 | **24.10** |
| Ours (Eq. (9)) | Product Kernel | $k_F\Big(f_q, f_k\Big) \cdot k_T\Big(t_q, t_k\Big)$ | **34.71** | 24.28 |

Table 2: Kernel Types. Other than manipulating the kernel choice of the non-positional features, we fix the configuration by Vaswani et al. (2017) for NMT and the configuration by Dai et al. (2019) for SP.

| Type | Kernel Form | NMT (BLEU↑) | | SP (Perplexity↓) | |
|---|---|---|---|---|---|
| | | Asym. ($W_q \neq W_k$) | Sym. ($W_q = W_k$) | Asym. ($W_q \neq W_k$) | Sym. ($W_q = W_k$) |
| Linear | $\langle f_a W_q, f_b W_k \rangle$ | not converge | not converge | not converge | not converge |
| Polynomial | $\Big(\langle f_a W_q, f_b W_k \rangle\Big)^2$ | 32.72 | 32.43 | 25.91 | 26.25 |
| Exponential | $\exp\Big(\frac{\langle f_a W_q, f_b W_k \rangle}{\sqrt{d_k}}\Big)$ | 33.98 | 33.78 | 24.10 | **24.01** |
| RBF | $\exp\Big(-\frac{\|f_a W_q - f_b W_k\|^2}{\sqrt{d_k}}\Big)$ | **34.26** | 34.14 | 24.13 | 24.21 |

constantly outperform the other integration types (it has lower BLEU score for NMT). Our proposed product kernel reaches the best result in NMT and is competitive to the best result in SP.

### 3.2 Kernel Types

To find the best kernel form in the attention mechanism, in addition to the exponential kernel (see Eq. (3)), we compare different kernel forms (i.e., linear, polynomial, and rbf kernel) for the non-positional features. We also provide the results for changing asymmetric to the symmetric kernel, when forcing $W_q = W_k$, so that the resulting kernel is a valid kernel (Scholkopf and Smola, 2001). The numbers are shown in Table 2. Note that, for fairness, other than manipulating the kernel choice of the non-positional features, we fix the configuration by Vaswani *et al.* (Vaswani et al., 2017) for NMT and the configuration by Dai *et al.* (Dai et al., 2019) for SP.

We first observe that the linear kernel does not converge for both NMT and SP. We argue the reason is that the linear kernel may have negative value and thus it violates the assumption in kernel smoother that the kernel score must be positive (Wasserman, 2006). Next, we observe the kernel with infinite feature space (i.e., exponential and rbf kernel) outperforms the kernel with finite feature space (i.e., polynomial kernel). And we see rbf kernel performs the best for NMT and exponential kernel performs the best for SP. We conclude that the choice of kernel matters for the design of attention in Transformer. Also, we see no much performance difference when comparing asymmetric to symmetric kernel. In the experiment, we fix the size of $W_\cdot$ in the kernel, and thus adopting the symmetric kernel benefits us from saving parameters.

### 3.3 Order-Invariance in Attention

The need of the positional embedding (PE) in the attention mechanism is based on the argument that the attention mechanism is an order-agnostic (or, permutation equivariant) operation (Vaswani et al., 2017; Shaw et al., 2018; Huang et al., 2018b; Dai et al., 2019; Child et al., 2019). However, we show that, for decoder self-attention, the operation is not order-agnostic. For clarification, we are not attacking the claim made by the prior work (Vaswani et al., 2017; Shaw et al., 2018; Huang et al., 2018b; Dai et al., 2019; Child et al., 2019), but we aim at providing a new look at the order-invariance problem when considering the attention mechanism with masks (masks refer to the set filtering function in our kernel formulation). In other words, previous work did not consider the

Table 3: Order-Invariance in Attention. To save the space, we denote Encoder Self-Attention / Encoder-Decoder Attention / Decoder Self-Attention as A/B/C. Note that SP only has decoder self-attention.

| Approach | Positional Embedding | NMT (BLEU↑) |
|---|---|---|
| Ours (Eq. (9)) | In A/B/C | **34.71** |
| Ours (Eq. (9)) | In A/B | 34.49 |
| No Positional Embedding | none | 14.47 |

| Approach | Positional Embedding | SP (Perplexity↓) |
|---|---|---|
| Vaswani et al. (2017) (Eq. (4)) | In C | 30.97 |
| Ours (Eq. (9) | In C | **24.28** |
| No Positional Embedding | none | 30.92 |

Table 4: Positional Embedding in Value Function.

| I: Value Function Considering Positional Embedding (Eq. (7)) / II: Value Function Considering no Positional Embedding (Eq. (8)) | | | | |
|---|---|---|---|---|
| Approach | NMT (BLEU↑) | | SP (Perplexity↓) | |
| | I $\left(v(x_k) := (f_k + t_k)W_V\right)$ | II $\left(v(x_k) := f_k W_V\right)$ | I $\left(v(x_k) := (f_k + t_k)W_V\right)$ | II $\left(v(x_k) := f_k W_V\right)$ |
| Vaswani et al. (2017) (Eq. (4)) | 33.98 | 34.02 | 30.97 | 30.50 |
| Shaw et al. (2018) (Eq. (6)) | 34.04 | 34.12 | 27.56 | 27.45 |
| Dai et al. (2019) (Eq. (5)) | 33.32 | 33.62 | 24.18 | **24.10** |
| Ours (Eq. (9)) | 34.60 | **34.71** | 24.42 | 24.28 |

mask between queries and keys when discussing the order-invariance problem (Pérez et al., 2019).

To put it formally, we first present the definition by Lee et al. (2018) for a permutation equivariance function:

**Definition 2.** *Denote* $\Pi$ *as the set of all permutations over* $[n] = \{1, \cdots, n\}$. *A function* $func : \mathcal{X}^n \rightarrow \mathcal{Y}^n$ *is permutation equivariant iff for any permutation* $\pi \in \Pi$, $func(\pi x) = \pi func(x)$.

Lee et al. (2018) showed that the standard attention (encoder self-attention (Vaswani et al., 2017; Dai et al., 2019) ) is permutation equivariant. Here, we present the non-permutation-equivariant problem on the decoder self-attention:

**Proposition 1.** *Decoder self-attention (Vaswani et al., 2017; Dai et al., 2019) is not permutation equivariant.*

To proceed the proof, we need the following definition and propositions.

**Definition 3.** *Denote* $\Pi$ *as the set of all permutations over* $[n] = \{1, \cdots, n\}$ *and* $S^{\pi}_{\mathbf{x}_k}$ *as performing permutation* $\pi$ *over* $S_{\mathbf{x}_k}$. $\text{Attention}(x_q; S_{\mathbf{x}_k})$ *is said to be permutation equivariant w.r.t.* $S_{\mathbf{x}_k}$ *if and only if for any* $\pi \in \Pi$, $\text{Attention}(x_q; S^{\pi}_{\mathbf{x}_k}) = \text{Attention}(x_q; S_{\mathbf{x}_k})$.

**Proposition 2.** *Attention with the set filtering function* $M(x_q, S_{\mathbf{x}_k}) = S_{\mathbf{x}_k}$ *is permutation equivariant w.r.t.* $S_{\mathbf{x}_k}$.

*Proof.* It is easy to show that if $M(x_q, S_{\mathbf{x}_k}) = S_{\mathbf{x}_k}$, Eq. (2) remains unchanged for any permutation $\pi$ performed on $S_{\mathbf{x}_k}$. ■

**Proposition 3.** *Attention with the set difference* $S_{\mathbf{x}_k} \setminus M(x_q, S_{\mathbf{x}_k}) \neq \phi$ *is not permutation equivariant w.r.t.* $S_{\mathbf{x}_k}$.

*Proof.* First, suppose that $\hat{x} \in S_{\mathbf{x}_k} \setminus M(x_q, S_{\mathbf{x}_k})$. Then, we construct a permutation $\pi$ such that $\hat{x} \in M(x_q, S^{\pi}_{\mathbf{x}_k})$. It is obvious that Eq. (2) changes after this permutation and thus $\text{Attention}\Big(x_q\ ;\ M(x_q, S_{\mathbf{x}_k})\Big)$ is not permutation equivariant w.r.t. $S_{\mathbf{x}_k}$. ■

*Proof.* [Proof for Proposition 1] First, we have $x_q \sim S_{\mathbf{x}_k}$. Hence, showing $\text{Attention}(x_q; S_{\mathbf{x}_k})$ not permutation equivariant w.r.t. $S_{\mathbf{x}_k}$ equals to showing Attention not permutation equivariant. Then, since the decoder self-attention considers masking (i.e., $M(x_q, S_{\mathbf{x}_k})$ returns a subset of $S_{\mathbf{x}_k}$), by Proposition 3, the decoder self-attention is not permutation equivariant. ■

In fact, not only being a permutation inequivariant process, the decoding process in the decoder self-attention already implies the order information from the data. To show this, take the decoded sequence $\mathbf{y} = [\text{init}, y_1, y_2, y_3, y_4]$ as an example. init stands for the initial token. When determining the output $y_1$ from init, the set filtering function is $M(\text{init}, S_{\mathbf{y}}) = \{\text{init}\}$. Similarly, we will have $M(y_1, S_{\mathbf{y}}), M(y_2, S_{\mathbf{y}}), M(y_3, S_{\mathbf{y}})$ to be $\{\text{init}, y_1\}, \{\text{init}, y_1, y_2\}, \{\text{init}, y_1, y_2, y_3\}$. Then, it raises a concern: do we require PE in decoder self-attention? By removing PE in decoder self-attention, we present the results in Table 3. From the table, we can see that, for NMT, removing PE

only in decoder self-attention results in slight performance drop (from 34.71 to 34.49). However, removing PE in the entire model greatly degrades the performance (from 34.71 to 14.47). On the other hand, for SP, removing PE from our proposed attention variant dramatically degrades the performance (from 24.28 to 30.92). Nonetheless, the performance is slightly better than considering PE from the original Transformer (Vaswani et al., 2017).

### 3.4 Positional Embedding in Value Function

To determine the need of positional embedding (PE) in value function, we conduct the experiments by adopting Eq. (7) or Eq. (8) in the attention mechanism. The results are presented in Table 4. From the table, we find that considering PE in value function (Eq. (7)) does not gain performance as compared to not considering PE in value function (Eq. (8)).

### 3.5 Take-Home Messages

Based on the results and discussions, we can now answer the questions given at the beginning of this section. The answers are summarized into the take-home messages in the following.

**A1.** We show that integrating the positional embedding in the form of product kernel (Eq. (5) or Eq. (9)) gives us best performance.

**A2.** The kernel form does matter. Adopting kernel form with infinite feature dimension (i.e., exponential kernel or rbf kernel) gives us best results. The symmetric design of the kernel may benefit us from saving parameters and barely sacrifice the performance as compared to the non-symmetric one.

**A3.** The decoder self-attention is not an order-agnostic operation with respect to the order of inputs. However, incorporating positional embedding into the attention mechanism may still improve performance.

**A4.** We find that there is no much performance difference by considering or not considering the positional embedding in value function.

## 4 Related Work

Other than relating Transformer's attention mechanism with kernel methods, the prior work (Wang et al., 2018; Shaw et al., 2018; Tsai et al., 2019b) related the attention mechanism with graph-structured learning. For example, Non-Local Neural Networks (Wang et al., 2018) made a connection between the attention and the non-local operation in image processing (Buades et al., 2005). Others (Shaw et al., 2018; Tsai et al., 2019b) linked the attention to the message passing in graphical models. In addition to the fundamental difference between graph-structured learning and kernel learning, the prior work (Wang et al., 2018; Shaw et al., 2018; Tsai et al., 2019b) focused on presenting Transformer for its particular application (e.g., video classification (Wang et al., 2018) and neural machine translation (Shaw et al., 2018)). Alternatively, our work focuses on presenting a new formulation of Transformer's attention mechanism that gains us the possibility for understanding the attention mechanism better.

## 5 Conclusions

In this paper, we presented a kernel formulation for the attention mechanism in Transformer, which allows us to define a larger space for designing attention. As an example, we proposed a new variant of attention which reaches competitive performance when compared to previous state-of-the-art models. Via the lens of the kernel, we were able to better understand the role of individual components in Transformer's attention and categorize previous attention variants in a unified formulation. Among these components, we found the construction of the kernel function acts the most important role, and we studied different kernel forms and the ways to integrate positional embedding on neural machine translation and sequence prediction. We hope our empirical study may potentially allow others to design better attention mechanisms given their particular applications.

## Acknowledgments

We thank Zhilin Yang for helpful discussion on the positional encoding in Transformer's Attention. This work was supported in part by the DARPA grant FA875018C0150, Office of Naval Research grant N000141812861, AFRL CogDeCON, NSF Awards #1734868 #1722822, National Institutes of Health, JST PRESTO program JPMJPR165A, and Apple. We would also like to acknowledge NVIDIA's GPU support.

## References


Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*.

Antoni Buades, Bartomeu Coll, and J-M Morel. 2005. A non-local algorithm for image denoising. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 60–65. IEEE.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Sergey Edunov, Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2017. Classical structured prediction losses for sequence to sequence learning. *arXiv preprint arXiv:1711.04956*.

Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Curtis Hawthorne, Andrew M Dai, Matthew D Hoffman, and Douglas Eck. 2018a. An improved relative self-attention mechanism for transformer with application to music generation. *arXiv preprint arXiv:1809.04281*.

Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Ian Simon, Curtis Hawthorne, Noam Shazeer, Andrew M Dai, Matthew D Hoffman, Monica Dinculescu, and Douglas Eck. 2018b. Music transformer: Generating music with long-term structure.

Brian Kulis, Kate Saenko, and Trevor Darrell. 2011. What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. In *CVPR 2011*, pages 1785–1792. IEEE.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R Kosiorek, Seungjin Choi, and Yee Whye Teh. 2018. Set transformer. *arXiv preprint arXiv:1810.00825*.

Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos. 2017. Mmd gan: Towards deeper understanding of moment matching network. In *Advances in Neural Information Processing Systems*, pages 2203–2213.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Łukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. 2018. Image transformer. *arXiv preprint arXiv:1802.05751*.

Jorge Pérez, Javier Marinković, and Pablo Barceló. 2019. On the turing completeness of modern neural network architectures. *arXiv preprint arXiv:1901.03429*.

Bernhard Scholkopf and Alexander J Smola. 2001. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Yao-Hung Hubert Tsai, Shaojie Bai, Paul Pu Liang, Louis-Philippe Morency, and Ruslan Salakhutdinov. 2019a. Multimodal transformer for unaligned multimodal language sequences. *ACL*.

Yao-Hung Hubert Tsai, Santosh Divvala, Louis-Philippe Morency, Ruslan Salakhutdinov, and Ali Farhadi. 2019b. Video relationship reasoning using gated spatio-temporal energy graph. *CVPR*.

Koji Tsuda. 1999. Support vector classifier with asymmetric kernel functions. In *in European Symposium on Artificial Neural Networks (ESANN*. Citeseer.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. 2018. Non-local neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7794–7803.

Larry Wasserman. 2006. *All of nonparametric statistics*. Springer Science & Business Media.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. 2016. Deep kernel learning. In *Artificial Intelligence and Statistics*, pages 370–378.

Alper Yilmaz. 2007. Object tracking by asymmetric kernel mean shift with automatic scale and orientation selection. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–6. IEEE.

# Learning to Learn and Predict: A Meta-Learning Approach for Multi-Label Classification

**Jiawei Wu** and **Wenhan Xiong** and **William Yang Wang**
Department of Computer Science
University of California, Santa Barbara
Santa Barbara, CA 93106 USA
{jiawei_wu,xwhan,william}@cs.ucsb.edu

## Abstract

Many tasks in natural language processing can be viewed as multi-label classification problems. However, most of the existing models are trained with the standard cross-entropy loss function and use a fixed prediction policy (e.g., a threshold of 0.5) for all the labels, which completely ignores the complexity and dependencies among different labels. In this paper, we propose a meta-learning method to capture these complex label dependencies. More specifically, our method utilizes a meta-learner to jointly learn the training policies and prediction policies for different labels. The training policies are then used to train the classifier with the cross-entropy loss function, and the prediction policies are further implemented for prediction. Experimental results on fine-grained entity typing and text classification demonstrate that our proposed method can obtain more accurate multi-label classification results.

## 1 Introduction

Multi-label classification aims at learning to make predictions on instances that are associated with multiple labels simultaneously, whereas in a classic multi-class classification setting, typically one instance has only one label. Multi-label classification is a common learning paradigm in a large amount of real-world natural language processing (NLP) applications, such as fine-grained entity typing (Ling and Weld, 2012; Shimaoka et al., 2017; Abhishek et al., 2017; Xin et al., 2018) and text classification (Nam et al., 2014; Liu et al., 2017; Chen et al., 2017; Wu et al., 2018).

Significant amounts of research studies have been dedicated to tackle the multi-label classification problem (Zhang and Zhou, 2014), from traditional statistical models (Zhang and Zhou, 2007; Zhou et al., 2012; Surdeanu et al., 2012) to neural network-based models (Nam et al., 2014; Shimaoka et al., 2017). These models have variable structures, but usually, share the standard cross-entropy loss function for training. After training, these models tend to use a single prediction policy for all the labels to generate the final predictions. Actually, the above process is based on the assumption that there is no dependency among the labels. However, as shown in Figure 1, this assumption is hard to be satisfied in real-world datasets, and the complex label dependencies receive little attention in multi-label classification (Dembszynski et al., 2010; Dembczyński et al., 2012). Owing to the impact of the label dependencies, although one object may have multiple labels simultaneously in the multi-label classification setting, the level of difficulty of prediction for different labels can vary a lot. Firstly, for those labels that are aligned with knowledge graphs, they usually indicate a hierarchical dependency struc-

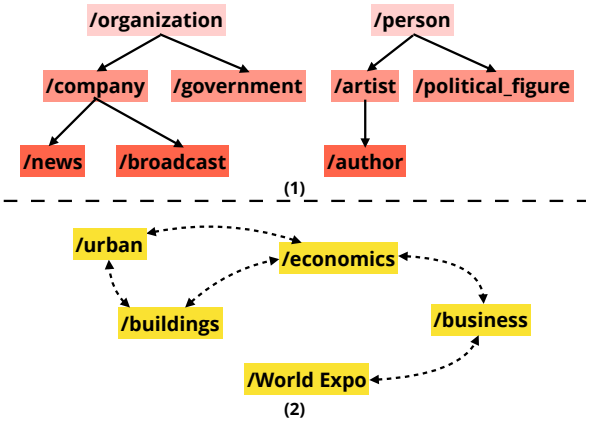


Figure 1: Illustration of complex label dependencies in multi-label classification. (1) Labels are aligned with some knowledge graphs. In this case, labels are organized in a hierarchical dependency structure. (2) Even if labels do not demonstrate explicit correlation, they can still have implicit semantic dependencies.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4354–4364,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

ture (Ling and Weld, 2012; Ren et al., 2016; Ruder et al., 2016). It is intuitive that one trained classifier is easier to distinguish high-level parent labels, such as `/organization` and `/person`, but harder to distinguish low-level child labels, such as `/news` and `/broadcast`. Secondly, for those cases where labels do not demonstrate explicit correlation, the labels still contain implicit semantic dependencies, which is extremely common in the NLP field. For instance, the label `/urban` and `/economics` have obvious semantic correlation even if they are not organized in a hierarchical structure. Meanwhile, the labels with more implicit dependencies are easier to predict because they expose more semantic information during the training. These intuitions inspire our work on learning different training policies and prediction policies for different labels.

The training policies and prediction policies for all the labels can be viewed as a series of hyper-parameters. However, to learn high-quality policies, one needs to specify both explicit and implicit label dependencies, which is not manually realistic. To resolve both issues mentioned above, we propose a meta-learning framework to model these label dependencies and learn training and prediction policies automatically. Concretely, we introduce a joint learning formulation of the meta-learning method and multi-label classification. A gated recurrent unit (GRU)-based (Chung et al., 2014) meta-learner is implemented to capture the label dependencies and learn these hyper-parameters during the training process of a classifier. Empirically, we show our method outperforms previous related methods on fine-grained entity typing and text classification problems. In summary, our contributions are three-fold:

- We are the first to propose a joint formulation of "learning to learn" and "learning to predict" in a multi-label classification setting.
- Our learning method can learn a weight and a decision policy for each label, which can then be incorporated into training and prediction.
- We show that our method is model-agnostic and can apply to different models in multi-label classification and outperform baselines.

In Section 2, we outline related work in multi-label classification and meta-learning. We then describe our proposed method in Section 3. We show experimental results in Section 4. Finally, we conclude in Section 7.

## 2 Related Work

### 2.1 Multi-Label Classification

Multi-label classification assigns instances with multiple labels simultaneously (Tsoumakas et al., 2006). (Shore and Johnson, 1980; De Boer et al., 2005) introduce prediction policies that weight the training loss function with external knowledge. However, in real-world multi-label classification, it is hard to obtain knowledge to determine prediction policies. As for the prediction policies, Yang (2001) select the thresholds that achieve the best evaluation measure on a validation set, while Lewis et al. (2004) utilize a cross-validation method to determine the thresholds. Lipton et al. (2014) propose an optimal decision rule to maximize F1 measure with thresholds. However, most of the previous methods can be viewed as post-processing because the prediction policies are computed after training. Meanwhile, the ability of prediction policies to help train the classifier is less explored.

Compared to previous related methods, we propose a principled approach that learns a training policy and a prediction policy for each label automatically with modeling label dependencies implicitly. Furthermore, the prediction policies are learned during the training process instead of the post-processing, which can also help to train a better classifier.

### 2.2 Meta-learning

Meta-learning is a "learning to learn" method, in which a learner learns new tasks and another meta-learner learns to train the learner (Bengio et al., 1990; Runarsson and Jonsson, 2000; Thrun and Pratt, 2012). There are two types of meta-learning:

- learning a meta-policy to update model parameters (Andrychowicz et al., 2016; Mishra et al., 2018).
- learning a good parameter initialization for fast adaptation (Duan et al., 2016; Vinyals et al., 2016; Finn et al., 2017; Snell et al., 2017; Gu et al., 2018).

In this paper, we propose to extend meta-learning algorithm for multi-label classification based on the first category. Instead of only training the

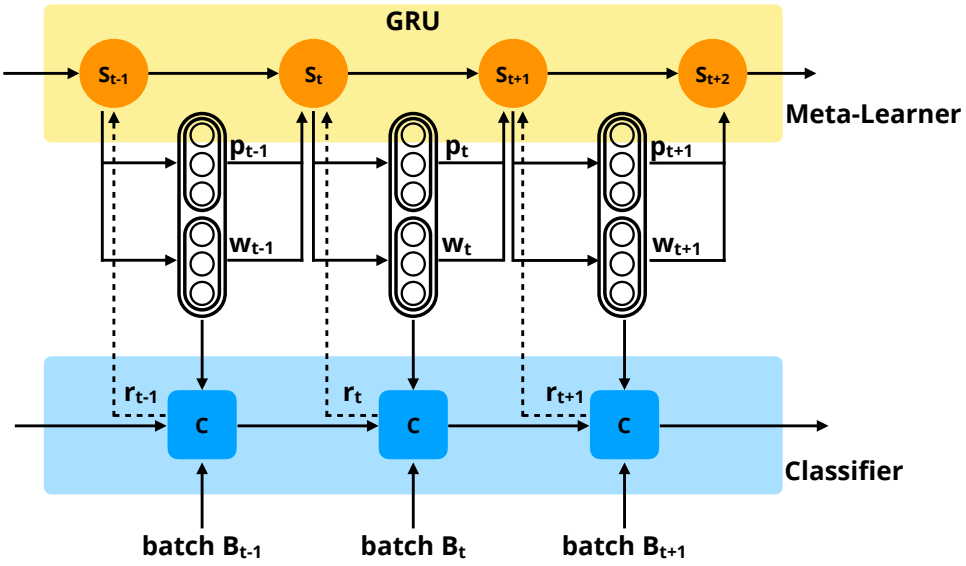


Figure 2: The meta-learning framework for multi-label classification.

model with meta-policy, we also consider prediction with meta-policy.

## 3 Method

In this section, we describe our meta-learning framework for learning and predicting multi-label classification in detail. In multi-label classification, the conventional methods usually learn classifiers with the standard cross-entropy loss function. After training, a single prediction policy (usually, a threshold of 0.5) is applied to all labels to generate the prediction. However, as mentioned in Figure 1, these methods ignore the explicit and implicit label dependencies among the labels. To improve the performance of multi-label classification, our main idea is to learn high-quality training policies and prediction policies by meta-learning, which can also model the label dependencies.

We view the training policies and prediction policies as a series of hyper-parameters, and then formulate the training of the classifier as a sequential process. At each iteration (time step $t$), the classifier updates its parameters with one sampled batch. The exploration of this classifier's hyper-parameters can be modeled by a meta-learner. We first briefly introduce the classifier model used in multi-label classification. Then we describe the meta-learner, and the interaction between the meta-learner and the classifier. Finally, we explain how to train the meta-learner in our unified framework.

### 3.1 Classifier Model

As mentioned in Section 1, many different model structures have already been explored in multi-label classification. Our framework is model-agnostic, which means different model structures are compatible if only they choose the cross-entropy loss function as the objective function. The classifier is represented as $C$. For a $N$- class multi-label classification, we represent the training policy as the vector $w = (w^{(1)}, w^{(2)}, \cdots, w^{(N)})$ and the prediction policy as the vector $p = (p^{(1)}, p^{(2)}, \cdots, p^{(N)})$, where $w^{(i)}$ and $p^{(i)}$ are the training weight and predicting threshold for the $i$-th class. $w_t$ and $p_t$ refer the weight vector and threshold vector at time step $t$. Then the goal of our framework is to learn a high-quality $w$ and $p$ for a certain classifier $C$.

To update the parameters of the classifier $C$, at each time step $t$, we sample a batch $B_t$ from the training set $U$. We then set a weighted cross-entropy objective function to update the $C$, which is defined as:

$$L(\theta_t^C) = -\sum_{i}^{B_t}\sum_{j}^{N} w_t^{(j)} N\{y_i^{*(j)} \log y_i^{(j)} + (1 - y_i^{*(j)}) \log(1 - y_i^{(j)})\}, \quad (1)$$

where $y_i^*$ indicates the ground truth prediction of the $i$-th sample from the $B_t$, and $y_i^{(j)}$ is the $j$-th entry of the corresponding output vector $y_i$. The

standard cross-entropy loss function is a special case when $w_t^{(j)} = \frac{1}{N}(j = 1, 2, \cdots, N)$.

### 3.2 Meta-Learner

Meta-learning is a widely used reinforcement learning method to learn the meta-information of a machine learning system (Bengio et al., 1990; Runarsson and Jonsson, 2000; Thrun and Pratt, 2012). The core of our proposed method is a meta-learner, which is trained to learn a training and a prediction policy for multi-label classification. At each time step $t$, the meta-learner observes the current state $s_t$, and then generate a training policy $w_t$ and a prediction policy $p_t$. Based on the policies $w_t$ and $p_t$, the parameters of classifier $C$ can be update with the sampled batch $B_t$ as described in Section 3.1. After training, the meta-learner receives a reward $r_t$. The goal of our meta-learner at each time step $t$ is to choose the two policies $w_t$ and $p_t$ that can maximize the future reward

$$R_t = \sum_{t'=t}^{T} r_{t'}, \tag{2}$$

where a training episode terminates at time $T$.

#### 3.2.1 State Representation

The state representation, in our framework, is designed to connect the policy generation and the environment. At each time step $t$, the training policy, and the prediction policy are generated based on the state $s_t$. A reward will be computed based on the change of the environment. In our case, the performance change of the classifier $C$. In order to successfully explore the policy space and generate high-quality policies, the meta-learner needs to remember what similar policies have already been tried and make further generation based on these memories.

Based on the above intuition, we formulate the meta-learner as a recurrent neural network (RNN)-based structure. To simplify our method, we use a GRU in our experiments. The state representation $s_t$ is directly defined as the hidden state $h_t$ of the GRU at time step $t$. The $s_t$ is computed according to:

$$s_t = \text{GRU}(s_{t-1}, \begin{bmatrix} p_{t-1} \\ w_{t-1} \end{bmatrix}), \tag{3}$$

where the input of GRU at time step $t$ is the concatenation of the prediction policy and training policy generated at time step $t-1$.

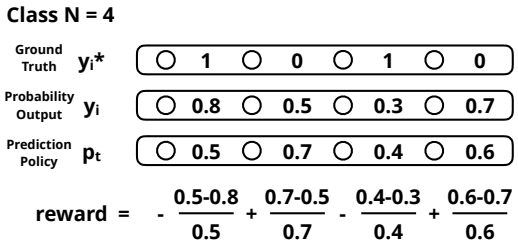


Figure 3: A example about the computation process of reward (one sample with class $N = 4$).

#### 3.2.2 Policy Generation

At each time step $t$, the meta-learner can generate two policies, the training policy $w_t$ and the prediction policy $p_t$. As mentioned in Section 3.1, both $w_t$ and $p_t$ are represented as a $N$-dimensional vector format.

To incorporate the training policy $w_t$ into the cross-entropy objective function in Equation 1 and keep the training gradients of the classifier in the same magnitude during the whole training episode, the condition $\sum_i^N w_t^{(i)} = 1$ for $w_t$ must be satisfied. Thus, at each time step $t$, the training policy is generated as:

$$w_t = \text{softmax}(W_w s_t + b_w). \tag{4}$$

As for the prediction policy, it is obvious that $p_t^{(i)} \in (0, 1)$ must be satisfied for $i = 1, 2, \cdots, N$. Then we define the prediction policy as:

$$p_t = \text{sigmoid}(W_p s_t + b_p). \tag{5}$$

The $s_t$ is the state representation of the meta-learner at time step $t$. $W_w$, $b_w$, $W_p$ and $b_p$ all are learnable parameters.

#### 3.2.3 Reward Function

The meta-leaner is trained to generate high-quality training and prediction policies jointly to improve the performance of the classifier $C$. To capture this performance change, we design a reward function based on the probability distributions of samples.

At each time step $t$, we first generate the training policy $w_t$ and the prediction policy $p_t$ according to Section 3.2.2. Then a batch $B_t$ is sampled from the training set and used to update the classifier $C$ based on Equation 1. We evaluate the output probability distribution of all the samples from $B_t$ on the classifier $C$ and compute the reward as:

$$r_t = \sum_{i}^{B_t} \sum_{j=1}^{N} (-1)^{y_i^{*(j)}} \frac{p_t^{(j)} - y_i^{(j)}}{p_t^{(j)}}, \tag{6}$$

where $y_i$ is the output probability vector of the $i$-th sample from the batch $B_t$ and $y_i^*$ is the corresponding ground truth vector. The superscript $(j)$ represents the $j$-th entry of a vector. A simple example about how the reward is computed is shown in Figure 3.

### 3.3 Training and Testing

The $\theta_{meta}$ is the set of all the parameters in the meta-learner, and the parameters can be trained by maximizing the total expect reward. The expected reward for an episode terminating at time step $T$ is defined as:

$$J(\theta_{meta}) = \mathbb{E}_{\pi}[\sum_{t=1}^{T} r_t]. \quad (7)$$

$J(\theta_{meta})$ can be maximized by directly applying gradient-based optimization methods. We optimize it using policy gradient descent and implement a reward baseline to lower the variance during the training (Sutton, 1984). The details of the training process are shown in Algorithm 1. During the training, the reason why we use sample batches instead of the full training set at each time step is that we want to explore more policy space with diversity instead of iteratively fitting the whole training set.

At test time, we rerun the meta-learner and the classifier simultaneously for one episode. However, we use the whole training set as a batch at each time step. The generated policies $w_T$ and $p_T$ is chosen as the final policies. We then train a classifier with the $w_T$-weighted cross-entropy objective function and test it based on the prediction policy $p_T$.

**Algorithm 1:** The algorithm of our meta-learning framework.

Given a set of labeled training data $U$
Given a untrained classifier $C$
**for** *episode* $\leftarrow$ *1* **to** *M* **do**
  Initialize $w_0 \leftarrow (\frac{1}{N}, \frac{1}{N}, \cdots, \frac{1}{N}) \in \mathbb{R}^N$
  Initialize $p_0 \leftarrow (0.5, 0.5, \cdots, 0.5) \in \mathbb{R}^N$
  **for** *time step* $t \leftarrow$ *1* **to** $T$ **do**
    $s_t \leftarrow \text{GRU}(s_{t-1}, \begin{bmatrix} p_{t-1} \\ w_{t-1} \end{bmatrix})$
    $w_t \leftarrow \text{softmax}(W_w s_t + b_w)$
    $p_t \leftarrow \text{sigmoid}(W_p s_t + b_p)$
    Sample a batch $B_t$ from $U$
    Update $C$ using $B_t$ with $w_t$-based objective function in Equation 1
    Compute reward $r_t$ with $p_t$ in Equation 6
  Update $\theta_{meta}$ using $g \propto \nabla_\theta J(\theta_{meta})$

## 4 Experimental Setups

We evaluate our proposed method in following two settings: (1) **Fine-grained Entity Typing**, where the labels have explicit hierarchical dependencies; (2) **Text Classification**, where one needs to model the implicit label dependencies.

### 4.1 Baselines

Since our method is model-agnostic, we directly employ the state-of-the-art (SOTA) models for both two tasks. The details of SOTA models will be discussed in Section 5 and 6. We compare our method with multiple baselines:

- **Hierarchy-Aware Training Policy**: To explicitly add hierarchical information during the training, the labels are first given integer weights. For instance, the first-level (parent) labels are given weight 1 while the third-level labels are given 3. All the integer weights then are normalized to add into the cross-entropy loss function in Equation 1.

- **SCutFBR Prediction Policy**: The rank-based policy method RCut and proportion-based assignments PCut are jointly considered to set prediction policies for different labels after obtaining the trained classifier (Yang, 2001).

- **ODR Prediction Policy**: After training, an optimal decision rule is to implement to choose prediction policies based on maximizing micro F1 scores (Lipton et al., 2014).

- **Predictions-as-Features**: The model trains a classifier for each label, organize the classifiers in a partially ordered structure, and take predictions produced by the former classifiers as the latter classifiers' features (Li et al., 2015).

- **Subset Maximization**: The model views the multi-label prediction problem as classifier chains, and then replace classifier chains with recurrent neural networks (Nam et al., 2017).

| Datasets | FIGER | OntoNotes | BBN |
|---|---|---|---|
| #Types | 128 | 89 | 47 |
| Max Hierarchy Depth | 2 | 3 | 2 |
| #Training | 2,690,286 | 220,398 | 86,078 |
| #Testing | 563 | 9,603 | 13,187 |

Table 1: The statistics of entity typing datasets.

## 5 Fine-grained Entity Typing

Entity type classification is the task for assigning semantic types to entity mentions based on their context. For instance, the system needs label the entity *"San Francisco Bay"* as `/location`, `/location/region` based on its context *"... the rest of San Francisco Bay, a spot ..."*.

In a fine-grained classification setting, entities are aligned with knowledge graphs (Ling and Weld, 2012), which typically makes labels be arranged in a hierarchical structure. We utilize our meta-learning framework to tackle this problem and evaluate our method.

**Datasets** We evaluate our method on three widely-used fine-grained entity typing datasets, FIGER, OntoNotes, and BBN, which are preprocessed by Ren et al. (2016). The statistics of these datasets are listed in Table 1.

**FIGER**: The training data is automatically generated by distant supervision, and then aligned with Freebase. The test data is collected from news reports and manually annotated by Ling and Weld (2012).

**OntoNotes**: The training sentences are collected from OntoNotes text corpus (Weischedel et al., 2013), and linked to Freebase. Gillick et al. (2014) releases a manually annotated test dataset.

**BBN**: The dataset consists of sentences from Wall Street Journal articles, which is entirely manually annotated (Weischedel and Brunstein, 2005).

**Implementation Details** We choose the SOTA model of fine-grained entity typing as the classifier $C$ (Shimaoka et al., 2017; Abhishek et al., 2017). The overall model structure is shown in Figure 4. Given en entity and its context sentence, the words are initialized with word embeddings (Mikolov et al., 2013). The mention representation is simply computed by averaging the word embeddings of entity mention words. As for the context representation, a bidirectional LSTM and attention mechanism are used to encode left and right context representation separately. The attention operation is computed using a Multi-Layer Perceptron (MLP) as follows:

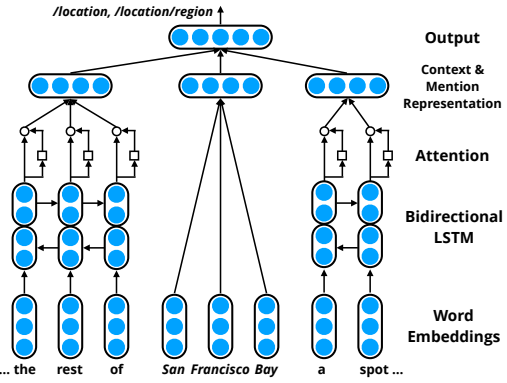


Figure 4: The SOTA model structure for fine-grained entity type classification (Shimaoka et al., 2017; Abhishek et al., 2017).

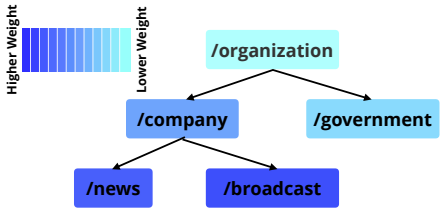


Figure 5: A case study about weights. See Section 5 for the detailed explanation.

$$a_i = \sigma\{W_1 \tanh(W_2 \begin{bmatrix} \overrightarrow{h}_i \\ \overleftarrow{h}_i \end{bmatrix})\}, \quad (8)$$

where $W_1$ and $W_2$ are parameter matrices of the MLP, and $\begin{bmatrix} \overrightarrow{h}_i \\ \overleftarrow{h}_i \end{bmatrix}$ is the hidden state of Bi-LSTM at the $i$-th position. Note that the parameters of MLP are shared by all the entities. The left, right context representations and mention representation are concatenated as the feature vector. A softmax layer is then implemented to perform final prediction. The standard cross-entropy loss function is used and the thresholds of all the labels are set as 0.5. To avoid overfitting, we employ dropout operation on entity mention representation.

Following the previous researches (Ling and Weld, 2012; Ren et al., 2016), we use strict accuracy, loose macro, and loose micro scores to evaluate the model performance.

**Results** The results of fine-grained entity typing are shown in Table 2. From the results, we observe that: (1) Our meta-learning method can outperform all the baselines in all three metrics, which

| Datasets | FIGER | | | OntoNotes | | | BBN | | |
|---|---|---|---|---|---|---|---|---|---|
| **Metrics** | **Acc.** | **Macro** | **Micro** | **Acc.** | **Macro** | **Micro** | **Acc.** | **Macro** | **Micro** |
| SOTA | 0.659 | 0.807 | 0.770 | 0.521 | 0.686 | 0.626 | 0.655 | 0.729 | 0.751 |
| SOTA+Hier-Training | 0.661 | 0.812 | 0.773 | 0.532 | 0.690 | 0.640 | 0.657 | 0.735 | 0.754 |
| SOTA+Meta-Training | 0.670 | 0.817 | 0.779 | 0.539 | 0.702 | 0.648 | 0.662 | 0.736 | 0.761 |
| SOTA+ScutFBR-Prediction | 0.662 | 0.814 | 0.782 | 0.542 | 0.695 | 0.650 | 0.661 | 0.736 | 0.758 |
| SOTA+ODR-Prediction | 0.669 | 0.818 | 0.782 | 0.537 | 0.703 | 0.648 | 0.664 | 0.738 | 0.764 |
| SOTA+Meta-Prediction | 0.674 | 0.823 | 0.786 | 0.544 | 0.709 | 0.657 | 0.671 | 0.744 | 0.769 |
| Predictions-as-Features | 0.663 | 0.816 | 0.785 | 0.544 | 0.699 | 0.655 | 0.663 | 0.738 | 0.761 |
| Subset Maximization | 0.678 | 0.827 | 0.790 | 0.546 | 0.713 | **0.661** | 0.673 | 0.748 | 0.772 |
| SOTA+Meta-Training-Prediction | **0.685** | **0.829** | **0.794** | **0.552** | **0.719** | **0.661** | **0.678** | **0.752** | **0.775** |

Table 2: The experimental results on fine-grained entity typing datasets. Acc.: Accuracy.

indicates the capability of our methods in improving the multi-label classification. (2) The SOTA model trained with learned training policy outperforms the one with hierarchy-aware training policy. It is because that our learned training policy can capture not only explicit hierarchical dependencies of labels, but also model other implicit label dependencies. (3) The three different prediction policies can improve the performance of the SOTA classifier. The results are consistent with previous researches that choosing a good prediction policy is an effective way to improve the performance (Fan and Lin, 2007; Lipton et al., 2014). (4) Compared with OntoNotes and BBN datasets, the FIGER show a relatively less improvement when applying these policies. The reason is that the test set of FIGER is not fine-grained enough (e.g., over 38% of entities are only annotated with `/person` and no more fine-grained labels) (Xin et al., 2018).

**Algorithm Robustness** Previous researches (Morimoto and Doya, 2005; Henderson et al., 2018) show that reinforcement learning-based methods usually lack robustness and are sensitive to the initialization, seeding datasets and pre-trained steps. Thus, we design an experiment to detect whether the trained meta-leaner is sensitive to the initialization. During the test time, instead of using the same initialization in Algorithm 1, we randomly initialize the $w_0$ and $p_0$ and learn 10 groups of policies $w_T$ and $p_T$. For each group of policies, we train a classifier with $w_T$ and evaluate it with $p_T$ using the same metric. The results are shown in Table 3. The results demonstrate that our trained meta-learner is robust to different initialization, which indicates that the meta-learner in our method can generate high-quality and robust training and prediction policies to improve the multi-label classification.

| Metrics | Best | Worst | Average | STDEV |
|---|---|---|---|---|
| Accuracy | 0.689 | 0.679 | 0.681 | 0.0043 |
| Macro-F1 | 0.835 | 0.821 | 0.827 | 0.0039 |
| Micro-F1 | 0.796 | 0.787 | 0.789 | 0.0036 |

Table 3: The robustness analysis on the FIGER dataset.

| Datasets | Reuters-21578 | RCV1-V2 |
|---|---|---|
| #Labels | 90 | 103 |
| #Average Labels/instance | 1.13 | 3.24 |
| #Training | 7,769 | 781,265 |
| #Testing | 3,019 | 23,149 |

Table 4: The statistics of text classification datasets.

**Weight Analysis** To analyze whether our meta-learner can really model the label dependencies, we perform a simple case study. In fine-grained entity classification, we choose 5 labels that are originally organized in a hierarchical structure from the OntoNotes dataset. The corresponding entries of the training policy vector $w_T$ are extracted and expressed with colors. The case study is shown in Figure 5. From the results, we can observe that high-level (parent) labels tend to have less training weights than low-level (child) labels. The results are consistent with our intuition that the labels with more dependencies expose more semantic information during the training.

## 6 Text Classification

Text classification is a classic problem for NLP, where one needs to categorized documents into pre-defined classes (Nam et al., 2014; Liu et al., 2017; Chen et al., 2017). We choose the datasets

| Datasets | Reuters-21578 | | | RCV1-V2 | | |
|---|---|---|---|---|---|---|
| **Metrics** | **Accuracy** | **Macro-F1** | **Micro-F1** | **Accuracy** | **Macro-F1** | **Micro-F1** |
| CNN | 0.537 | 0.472 | 0.841 | 0.616 | 0.642 | 0.838 |
| CNN+Meta-Training | 0.542 | 0.476 | 0.843 | 0.631 | 0.655 | 0.852 |
| CNN+ScutFBR-Prediction | 0.549 | 0.477 | 0.849 | 0.634 | 0.651 | 0.856 |
| CNN+ODR-Prediction | 0.541 | 0.475 | 0.848 | 0.630 | 0.653 | 0.849 |
| CNN+Meta-Prediction | 0.549 | 0.479 | 0.851 | 0.639 | 0.658 | 0.857 |
| Predictions-as-Features | 0.539 | 0.476 | 0.845 | 0.621 | 0.644 | 0.847 |
| Subset Maximization | 0.543 | 0.478 | 0.849 | 0.632 | 0.660 | 0.859 |
| CNN+Meta-Training-Prediction | **0.556** | **0.483** | **0.854** | **0.647** | **0.669** | **0.864** |

Table 5: The experimental results on text classification datasets.

| Metrics | Best | Worst | Average | STDEV |
|---|---|---|---|---|
| Accuracy | 0.652 | 0.641 | 0.646 | 0.0028 |
| Macro-F1 | 0.678 | 0.654 | 0.663 | 0.0041 |
| Micro-F1 | 0.874 | 0.855 | 0.863 | 0.0033 |

Table 6: The robustness analysis on the RCV1 dataset.

in which samples have multiple labels and evaluate our model on text classification problem.

**Datasets** Following the settings in (Nam et al., 2014), we choose two multi-label datasets, Reuters-21578 and RCV1-V2, to test our method. The statistics of two datasets are listed in Table 4.

**Reuters-21578**: The instances are collected Reuters news articles during the period 1987 to 1991. We use the same training/test split as previous work (Yang, 2001; Nam et al., 2014).

**RCV1-V2**: RCV1-V2 collects newswire stories from Reuters (Lewis et al., 2004). The training and test dataset originally consist of $23,149$ train and $781,265$ test instances, but we switch them to better training and evaluation (Nam et al., 2014).

**Setup** Many researches have proved convolutional neural networks (CNN) are effective in extracting information for text classification (LeCun et al., 1998; Kim, 2014; Zhang et al., 2015). Following the (Kim, 2014), we set a CNN model as the classifier $C$ to evaluate our method. Concretely, we use CNN-non-static mentioned in Kim (2014), which means we initialize the word embeddings with pre-trained Word2Vec and update the word embeddings during the training. The standard cross-entropy loss function is implemented, and the thresholds of all the classes are set as 0.5.

We still use strict accuracy, loose macro, and loose micro scores to evaluate the model performance following the settings in (Lewis et al., 2004; Yang and Gopal, 2012; Nam et al., 2014).

**Results** The results of text classification are shown in Table 5. From the results, we can observe that: (1) Our meta-learning method can outperform all the baselines on two text classification tasks, which indicates that our approach is consistent with different tasks. (2) Compared with RCV1-V2, the classification results on Reuters-21578 show less improvement. The reason is that the number of average labels per instance in Reuters-21578 is 1.13, while the number is 3.24 for RCV1-V2 according to Table 4. That means the multi-label classification on Reuters is close to the multi-class classification. There are little potential label dependencies in Reuters-21578.

**Algorithm Robustness** Similar to Section 5, we evaluate whether our trained meta-learner is sensitive to the initialization. We follow the same steps mentioned in Section 5, and show the results in Table 6. The results indicate that the robustness of our meta-learner is consistent within different tasks and model structures, which again shows that the trained meta-learner can generate high-quality and robust policies.

## 7 Conclusion

In this paper, we propose a novel meta-learner to improve the multi-label classification. By modeling the explicit and implicit label dependencies automatically, the meta-learner in our model can learn to generate high-quality training and prediction policies to help both the training and testing process of multi-label classifiers in a principled way. We evaluate our models on two tasks, fine-grained entity typing and text classification. Experimental results show that our method outper-

forms other baselines.

## Acknowledgments

The authors would like to thank the anonymous reviewers for their thoughtful comments. This research was supported in part by DARPA Grant D18AP00044 funded under the DARPA YFA program. The authors are solely responsible for the contents of the paper, and the opinions expressed in this publication do not reflect those of the funding agencies.

## References

Abhishek Abhishek, Ashish Anand, and Amit Awekar. 2017. Fine-grained entity type classification by jointly learning representations and label embeddings. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 797–807.

Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. 2016. Learning to learn by gradient descent by gradient descent. In *Proceedings of the 30th Conference on Neural Information Processing Systems (NeurIPS)*, pages 3981–3989.

Yoshua Bengio, Samy Bengio, and Jocelyn Cloutier. 1990. *Learning a synaptic learning rule*. Université de Montréal, Département d'informatique et de recherche opérationnelle.

Sheng Chen, Akshay Soni, Aasish Pappu, and Yashar Mehdad. 2017. Doctag2vec: An embedding based multi-label learning approach for document tagging. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 111–120.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. 2005. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134(1):19–67.

Krzysztof Dembczyński, Willem Waegeman, Weiwei Cheng, and Eyke Hüllermeier. 2012. On label dependence and loss minimization in multi-label classification. *Machine Learning*, 88(1-2):5–45.

Krzysztof Dembszynski, Willem Waegeman, Weiwei Cheng, and Eyke Hüllermeier. 2010. On label dependence in multilabel classification. In *ICML Workshop on Learning from Multi-Label Data*.

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. 2016. Rl2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*.

Rong-En Fan and Chih-Jen Lin. 2007. A study on threshold selection for multi-label classification. *Department of Computer Science, National Taiwan University*, pages 1–23.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 1126–1135.

Dan Gillick, Nevena Lazic, Kuzman Ganchev, Jesse Kirchner, and David Huynh. 2014. Context-dependent fine-grained entity type tagging. *arXiv preprint arXiv:1412.1820*.

Jiatao Gu, Yong Wang, Yun Chen, Victor OK Li, and Kyunghyun Cho. 2018. Meta-learning for low-resource neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3622–3631.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. 2018. Deep reinforcement learning that matters. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the Institute of Electrical and Electronics Engineers*, 86(11):2278–2324.

David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. 2004. Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397.

Li Li, Houfeng Wang, Xu Sun, Baobao Chang, Shi Zhao, and Lei Sha. 2015. Multi-label text categorization with joint learning predictions-as-features method. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 835–839.

Xiao Ling and Daniel S Weld. 2012. Fine-grained entity recognition. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI)*.

Zachary C Lipton, Charles Elkan, and Balakrishnan Naryanaswamy. 2014. Optimal thresholding of classifiers to maximize f1 measure. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*, pages 225–239.

Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. 2017. Deep learning for extreme multi-label text classification. In *Proceedings of the*

*40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 115–124.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 27th Conference on Neural Information Processing Systems (NeurIPS)*, pages 3111–3119.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. 2018. A simple neural attentive meta-learner. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*.

Jun Morimoto and Kenji Doya. 2005. Robust reinforcement learning. *Neural computation*, 17(2):335–359.

Jinseok Nam, Jungi Kim, Eneldo Loza Mencía, Iryna Gurevych, and Johannes Fürnkranz. 2014. Large-scale multi-label text classificationrevisiting neural networks. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*, pages 437–452.

Jinseok Nam, Eneldo Loza Mencía, Hyunwoo J Kim, and Johannes Fürnkranz. 2017. Maximizing subset accuracy with recurrent neural networks in multi-label classification. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPS)*, pages 5413–5423.

Xiang Ren, Wenqi He, Meng Qu, Lifu Huang, Heng Ji, and Jiawei Han. 2016. Afet: Automatic fine-grained entity typing by hierarchical partial-label embedding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1369–1378.

Sebastian Ruder, Parsa Ghaffari, and John G Breslin. 2016. A hierarchical model of reviews for aspect-based sentiment analysis. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 999–1005.

Thomas Philip Runarsson and Magnus Thor Jonsson. 2000. Evolution and design of distributed learning rules. In *Proceedings of the 2000 IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks*, pages 59–63.

Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. 2017. Neural architectures for fine-grained entity type classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 1271–1280.

John Shore and Rodney Johnson. 1980. Axiomatic derivation of the principle of maximum entropy and the principle of minimum cross-entropy. *IEEE Transactions on Information Theory*, 26(1):26–37.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPS)*, pages 4077–4087.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 455–465.

Richard Stuart Sutton. 1984. Temporal credit assignment in reinforcement learning.

Sebastian Thrun and Lorien Pratt. 2012. *Learning to learn*. Springer Science & Business Media.

Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. 2006. A review of multi-label classification methods. In *Proceedings of the 2nd ADBIS Workshop on Data Mining and Knowledge Discovery (ADMKD)*.

Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. In *Proceedings of the 30th Conference on Neural Information Processing Systems (NeurIPS)*, pages 3630–3638.

Ralph Weischedel and Ada Brunstein. 2005. Bbn pronoun coreference and entity type corpus. *Linguistic Data Consortium (LDC)*.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al. 2013. Ontonotes release 5.0 ldc2013t19. *Linguistic Data Consortium (LDC)*.

Jiawei Wu, Lei Li, and William Yang Wang. 2018. Reinforced co-training. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 1252–1262.

Ji Xin, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2018. Improving neural fine-grained entity typing with knowledge attention. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*.

Yiming Yang. 2001. A study of thresholding strategies for text categorization. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR)*, pages 137–145.

Yiming Yang and Siddharth Gopal. 2012. Multilabel classification with meta-level features in a learning-to-rank framework. *Machine Learning*, 88(1-2):47–68.

Min-Ling Zhang and Zhi-Hua Zhou. 2007. Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038–2048.

Min-Ling Zhang and Zhi-Hua Zhou. 2014. A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering*, 26(8):1819–1837.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the 29th Conference on Neural Information Processing Systems (NeurIPS)*, pages 649–657.

Zhi-Hua Zhou, Min-Ling Zhang, Sheng-Jun Huang, and Yu-Feng Li. 2012. Multi-instance multi-label learning. *Artificial Intelligence*, 176(1):2291–2320.

# Revealing the Dark Secrets of BERT

**Olga Kovaleva, Alexey Romanov, Anna Rogers, Anna Rumshisky**
Department of Computer Science
University of Massachusetts Lowell
Lowell, MA 01854
{okovalev,arum,aromanov}@cs.uml.edu

## Abstract

BERT-based architectures currently give state-of-the-art performance on many NLP tasks, but little is known about the exact mechanisms that contribute to its success. In the current work, we focus on the interpretation of self-attention, which is one of the fundamental underlying components of BERT. Using a subset of GLUE tasks and a set of handcrafted features-of-interest, we propose the methodology and carry out a qualitative and quantitative analysis of the information encoded by the individual BERT's heads. Our findings suggest that there is a limited set of attention patterns that are repeated across different heads, indicating the overall model overparametrization. While different heads consistently use the same attention patterns, they have varying impact on performance across different tasks. We show that manually disabling attention in certain heads leads to a performance improvement over the regular fine-tuned BERT models.

## 1 Introduction

Over the past year, models based on the Transformer architecture (Vaswani et al., 2017) have become the de-facto standard for state-of-the-art performance on many natural language processing (NLP) tasks (Radford et al., 2018; Devlin et al., 2018). Their key feature is the self-attention mechanism that provides an alternative to conventionally used recurrent neural networks (RNN).

One of the most popular Transformer-based models is BERT, which learns text representations using a bi-directional Transformer encoder pre-trained on the language modeling task (Devlin et al., 2018). BERT-based architectures have produced new state-of-the-art performance on a range of NLP tasks of different nature, domain, and complexity, including question answering, sequence tagging, sentiment analysis, and inference. State-of-the-art performance is usually obtained by fine-tuning the pre-trained model on the specific task. In particular, BERT-based models are currently dominating the leaderboards for SQuAD[1] (Rajpurkar et al., 2016) and GLUE benchmarks[2] (Wang et al., 2018).

However, the exact mechanisms that contribute to the BERT's outstanding performance still remain unclear. We address this problem through selecting a set of linguistic features of interest and conducting a series of experiments that aim to provide insights about how well these features are captured by BERT. This paper makes the following contributions:

- We propose a methodology and offer the first detailed analysis of BERT's capacity to capture different kinds of linguistic information by encoding it in its self-attention weights.
- We present evidence of BERT's overparametrization and suggest a counter-intuitive yet frustratingly simple way of improving its performance, showing absolute gains of up to 3.2%.

## 2 Related work

There have been several recent attempts to assess BERT's ability to capture structural properties of language. Goldberg (2019) demonstrated that BERT consistently assigns higher scores to the correct verb forms as opposed to the incorrect one in a masked language modeling task, suggesting some ability to model subject-verb agreement. Jawahar et al. (2019) extended this work to using multiple layers and tasks, supporting the claim that BERT's intermediate layers capture rich linguistic information. On the other hand, Tran et al. (2018) concluded that LSTMs generalize to longer

[1] https://rajpurkar.github.io/SQuAD-explorer/
[2] https://gluebenchmark.com/leaderboard



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4365–4374,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

sequences better, and are more robust with respect to agreement distractors, compared to Transformers.

Liu et al. (2019) investigated the transferability of contextualized word representations to a number of probing tasks requiring linguistic knowledge. Their findings suggest that (a) the middle layers of Transformer-based architectures are the most transferable to other tasks, and (b) higher layers of Transformers are not as task specific as the ones of RNNs. Tang et al. (2018) argued that models using self-attention outperform CNN- and RNN-based models on a word sense disambiguation task due to their ability to extract semantic features from text. Voita et al. (2019) analyzed the original Transformer model on a translation task and found out that only a small subset of heads is important for the given task, but these heads have interpretable linguistic functions.

Our work contributes to the above discussion, but rather than examining representations extracted from different layers, we focus on the understanding of the self-attention mechanism itself, since it is the key feature of Transformer-based models.

Another research direction that is relevant to our work is neural network pruning. Frankle and Carbin (2018) showed that widely used complex architectures suffer from overparameterization, and can be significantly reduced in size without a loss in performance. Goldberg (2019) observed that the smaller version of BERT achieves better scores on a number of syntax-testing experiments than the larger one. Adhikari et al. (2019) questioned the necessity of computation-heavy neural networks, proving that a simple yet carefully tuned BiLSTM without attention achieves the best or at least competitive results compared to more complex architectures on the document classification task. Wu et al. (2019) presented more evidence of unnecessary complexity of the self-attention mechanism, and proposed a more lightweight and scalable dynamic convolution-based architecture that outperforms the self-attention baseline. Michel et al. (2019) demonstrated that some layers in Transformer can be reduced down to a single head without significant degradation of model performance. These studies suggest a potential direction for future research, and are in good accordance with our observations.

## 3 Methodology

We pose the following research questions:

1. What are the common attention patterns, how do they change during fine-tuning, and how does that impact the performance on a given task? (Sec. 4.1, 4.3)
2. What linguistic knowledge is encoded in self-attention weights of the fine-tuned models and what portion of it comes from the pre-trained BERT? (Sec. 4.2, 4.4, 4.5)
3. How different are the self-attention patterns of different heads, and how important are they for a given task? (Sec. 4.6)

The answers to these questions come from a series of experiments with the basic pre-trained or the fine-tuned BERT models, as will be discussed below. All the experiments with the pre-trained BERT were conducted using the model provided with the PyTorch implementation of BERT (bert-base-uncased, 12-layer, 768-hidden, 12-heads, 110M parameters)[3]. We chose this smaller version of BERT because it shows competitive, if not better, performance while having fewer layers and heads, which makes it more interpretable.

We use the following subset of GLUE tasks (Wang et al., 2018) for fine-tuning:

- *MRPC*: the Microsoft Research Paraphrase Corpus (Dolan and Brockett, 2005)
- *STS-B*: the Semantic Textual Similarity Benchmark (Cer et al., 2017)
- *SST-2*: the Stanford Sentiment Treebank, two-way classification (Socher et al., 2013)
- *QQP*: the Quora Question Pairs dataset
- *RTE*: the Recognizing Textual Entailment datasets
- *QNLI*: Question-answering NLI based on the Stanford Question Answering Dataset (Rajpurkar et al., 2016)
- *MNLI*: the Multi-Genre Natural Language Inference Corpus, matched section (Williams et al., 2018)

Please refer to the original GLUE paper for details on the QQP and RTE datasets (Wang et al., 2018). We excluded two tasks: CoLa and the Winograd Schema Challenge. The latter is excluded due to the small size of the dataset. As for CoLa (the task of predicting linguistic acceptability judgments), GLUE authors report that the hu-

[3] https://github.com/huggingface/pytorch-pretrained-BERT

man performance is only 66.4, which is explained by the problems with the underlying methodology (Schutze, 1996). Note also that CoLa is not included in the upcoming version of GLUE (Wang et al., 2019). All fine-tuning experiments follow the parameters reported in the original study (a batch size of 32 and 3 epochs, see Devlin et al. (2018)).

In all these experiments, for a given input, we extract self-attention weights for each head in every layer. This results in a 2D float array of shape $L \times L$, where $L$ is the length of an input sequence. We will refer to such arrays as *self-attention maps*. Analysis of individual self-attention maps allows us to determine which target tokens are attended to the most as the input is processed token by token. We use these experiments to analyze how BERT processes different kinds of linguistic information, including the processing of different parts of speech (nouns, pronouns, and verbs), syntactic roles (objects, subjects), semantic relations, and negation tokens.

## 4 Experiments

In this section, we present the experiments conducted to address the above research questions.

### 4.1 BERT's self-attention patterns

Manual inspection of self-attention maps for both basic pre-trained and fine-tuned BERT models suggested that there is a limited set of self-attention map types that are repeatedly encoded across different heads. Consistently with previous observations[4], we identified five frequently occurring patterns, examples of which are shown in Figure 1:

- *Vertical*: mainly corresponds to attention to special BERT tokens *[CLS]* and *[SEP]* which serve as delimiters between individual chunks of BERT's inputs;
- *Diagonal*: formed by the attention to the previous/following tokens;
- *Vertical+Diagonal*: a mix of the previous two types,
- *Block*: intra-sentence attention for the tasks with two distinct sentences (such as, for example, RTE or MRPC),
- *Heterogeneous*: highly variable depending on the specific input and cannot be characterized by a distinct structure.

Note that, because the *Heterogeneous* category contains patterns not included in the other four categories, our constructed list of classes is exhaustive.

Whereas the attention to the special tokens is important for cross-sentence reasoning, and the attention to the previous/following token comes from language model pre-training, we hypothesize that the last of the listed types is more likely to capture interpretable linguistic features, necessary for language understanding.

To get a rough estimate of the percentage of attention heads that may capture linguistically interpretable information, we manually annotated around 400 sample self-attention maps as belonging to one of the five classes. The self-attention maps were obtained by feeding random input examples from selected tasks into the corresponding fine-tuned BERT model. This produced a somewhat unbalanced dataset, in which the "Vertical" class accounted for 30% of all samples. We then trained a convolutional neural network with 8 convolutional layers and ReLU activation functions to classify input maps into one of these classes. This model achieved the F1 score of 0.86 on the annotated dataset. We used this classifier to estimate the proportion of different self-attention patterns for the target GLUE tasks using up to 1000 examples (where available) from each validation set.

**Results** Figure 2 shows that the self-attention map types described above are consistently repeated across different heads and tasks. While a large portion of encoded information corresponds to attention to the previous/following token, to the special tokens, or a mixture of the two (the first three classes), the estimated upper bound on all heads in the "Heterogeneous" category (i.e. the ones that *could* be informative) varies from 32% (MRPC) to 61% (QQP) depending on the task.

We would like to emphasize that this only gives the upper bound on the percentage of attention heads that could potentially capture meaningful structural information beyond adjacency and separator tokens.

### 4.2 Relation-specific heads in BERT

In this experiment, our goal was to understand whether different syntactic and semantic relations are captured by self-attention patterns. While a

[4] https://towardsdatascience.com/deconstructing-bert-distilling-6-patterns-\from-100-million-parameters-b49113672f77

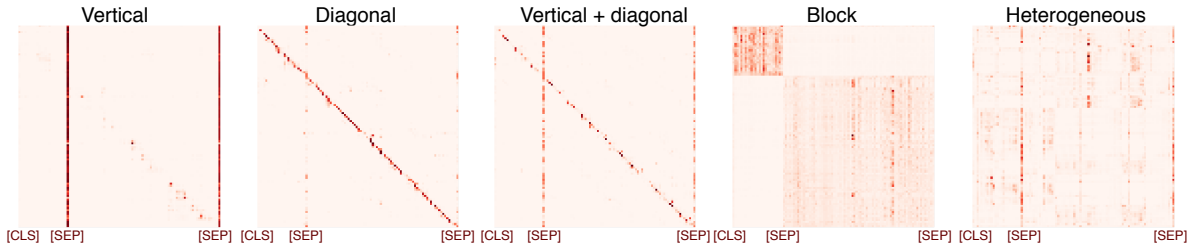


Figure 1: Typical self-attention classes used for training a neural network. Both axes on every image represent BERT tokens of an input example, and colors denote absolute attention weights (darker colors stand for greater weights). The first three types are most likely associated with language model pre-training, while the last two potentially encode semantic and syntactic information.

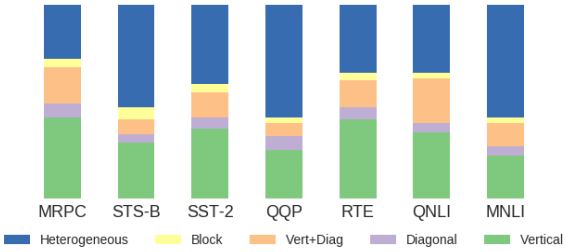


Figure 2: Estimated percentages of the identified self-attention classes for each of the selected GLUE tasks.

large number of such relations could be investigated, we chose to examine semantic role relations defined in frame semantics, since they can be viewed as being at the intersection of syntax and semantics. Specifically, we focused on whether BERT captures FrameNet's relations between frame-evoking lexical units (predicates) and core frame elements (Baker et al., 1998), and whether the links between them produce higher attention weights in certain specific heads. We used pre-trained BERT in these experiments.

The data for this experiment comes from FrameNet (Baker et al., 1998), a database that contains frame annotations for example sentences for different lexical units. Frame elements correspond to semantic roles for a given frame, for example, "buyer", "seller", and "goods" for the "Commercial_transaction" frame evoked by the words "sell" and "spend" or "topic" and "text" for the "Scrutiny" semantic frame evoked by the verb "address". Figure 4 shows an example of such annotation.

We extracted sample sentences for every lexical unit in the database and identified the corresponding core frame elements. Annotated elements in FrameNet may be rather long, so we considered only the sentences with frame elements of 3 tokens or less. Since each sentence is annotated only for one frame, semantic links from other frames can exist between unmarked elements. We therefore filter out all the sentences longer than 12 tokens, since shorter sentences are less likely to evoke multiple frames.

To establish whether BERT attention captures semantic relations that *do not* simply correspond to the previous/following token, we exclude sentences where the linked objects are less than two tokens apart. This leaves us with 473 annotated sentences.

For each of these sentences, we obtain pre-trained BERT's attention weights for each of the 144 heads. For every head, we return the maximum absolute attention weight among those token pairs that correspond to the annotated semantic link contained within a given sentence. We then average the derived scores over all the collected examples. This strategy allows us to identify the heads that prioritize the features correlated with frame-semantic relations within a sentence.

**Results** The heatmap of averaged attention scores over all collected examples (Figure 3) suggests that 2 out of 144 heads tend to attend to the parts of the sentence that FrameNet annotators identified as core elements of the same frame. The maximum attention weights averaged over all data examples for these identified heads account for 0.201 and 0.209, which are greater than a 99-th percentile of the distribution of values for all heads. Figure 3 shows an example of this attention pattern for these two heads. Both show high attention weight for "he" while processing "agitated" in the sentence "He was becoming agitated" (the frame "Emotion_directed").

We interpret these results as limited evidence that certain types of linguistic relations may be

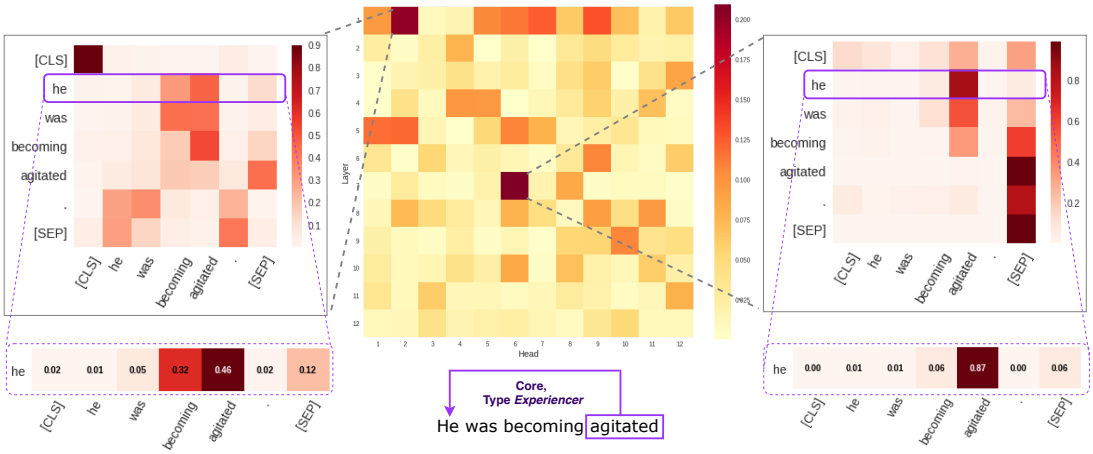


Figure 3: Detection of pre-trained BERT's heads that encode information correlated to semantic links in the input text. Two heads (middle) demonstrate their ability to capture semantic relations. Note that the heatmap in the middle is obtained through averaging of all the individual input example maps. For one random annotated FrameNet example (bottom) full attention maps with a zoom in the target token attention distribution are shown (leftmost and rightmost).

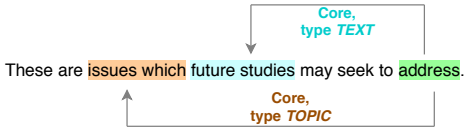


Figure 4: FrameNet annotation example for the "address" lexical unit with two core frame elements of different types annotated.

captured by self-attention patterns in specialized BERT heads. A wider range of relations remains to be investigated.

### 4.3 Change in self-attention patterns after fine-tuning

Fine-tuning has a huge effect on performance, and this section attempts to find out why. To study how attention per head changes on average for each of the target GLUE tasks, we calculate cosine similarity between pre-trained and fine-tuned BERT's flattened arrays of attention weights. We average the derived similarities over all the development set examples[5]. To evaluate contribution of pre-trained BERT to overall performance on the tasks, we consider two configurations of weights initialization, namely, pre-trained BERT weights and weights randomly sampled from normal distribution.

[5]If the number of development data examples for a given task exceeded 1000 (QQP, QNLI, MNLI, STS-B), we randomly sampled 1000 examples.

**Results** Figure 5 shows that for all the tasks except QQP, it is the last two layers that undergo the largest changes compared to the pre-trained BERT model. At the same time, Table 1 shows that fine-tuned BERT outperforms pre-trained BERT by a significant margin on all the tasks (with an average of 35.9 points of absolute difference). This leads us to conclude that the last two layers encode task-specific features that are attributed to the gain of scores, while earlier layers capture more fundamental and low-level information used in fine-tuned models. BERT with weights initialized from normal distribution and further fine-tuned for a given task consistently produces lower scores than the ones achieved with pre-trained BERT. In fact, for some tasks (STS-B and QNLI), initialization with random weights yields worse performance than pre-trained BERT without fine-tuning.

This suggests that pre-trained BERT does indeed contain linguistic knowledge that is help-

| Dataset | Pre-trained | Fine-tuned, initialized with normal distr. | Fine-tuned, initialized with pre-trained | Metric | Size |
|---|---|---|---|---|---|
| MRPC | 0/31.6 | 81.2/68.3 | 87.9/82.3 | F1/Acc | 5.8K |
| STS-B | 33.1 | 2.9 | 82.7 | Acc | 8.6K |
| SST-2 | 49.1 | 80.5 | 92 | Acc | 70K |
| QQP | 0/60.9 | 0/63.2 | 65.2/78.6 | F1/Acc | 400K |
| RTE | 52.7 | 52.7 | 64.6 | Acc | 2.7K |
| QNLI | 52.8 | 49.5 | 84.4 | Acc | 130K |
| MNLI-m | 31.7 | 61.0 | 78.6 | Acc | 440K |

Table 1: GLUE task performance of BERT models with different initialization. We report the scores on the validation, rather than test data, so these results differ from the original BERT paper.

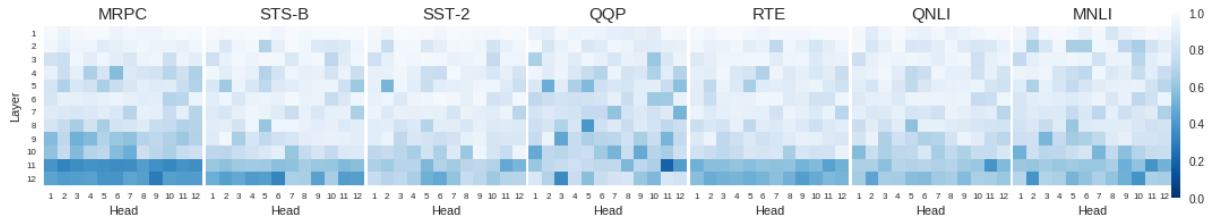


Figure 5: Per-head cosine similarity between pre-trained BERT's and fine-tuned BERT's self-attention maps for each of the selected GLUE tasks, averaged over validation dataset examples. Darker colors correspond to greater differences.




Figure 6: Per-task attention weights to the *[SEP]* (top row) and the *[CLS]* (bottom row) tokens averaged over input sequences' lengths and over dataset examples. Darker colors correspond to greater absolute weights.

ful for solving these GLUE tasks. These results are consistent with similar studies, e.g., Yosinski et al. (2014)'s results on fine-tuning a convolutional neural network pre-trained on ImageNet or Romanov and Shivade (2018)'s results on transfer learning for medical natural language inference.

### 4.4 Attention to linguistic features

In this experiment, we investigate whether fine-tuning BERT for a given task creates self-attention patterns which emphasize specific linguistic features. In this case, certain kinds of tokens may get high attention weights from all the other tokens in the sentence, producing vertical stripes on the corresponding attention maps (Figure 1).

To test this hypothesis we checked whether there are vertical stripe patterns corresponding to certain linguistically interpretable features, and to what extent such features are relevant for solving a given task. In particular, we investigated attention to nouns, verbs, pronouns, subjects, objects, and negation words[6], and special BERT tokens across the tasks.

For every head, we compute the sum of self-attention weights assigned to the token of interest from each input token. Since the weights depend on the number of tokens in the input sequence, this sum is normalized by sequence length. This allows us to aggregate the weights for this feature across different examples. If there are multiple tokens of the same type (e.g. several nouns or negations), we take the maximum value. We disregard input sentences that do not contain a given feature.

For each investigated feature, we calculate this aggregated attention score for each head in every layer and build a map in order to detect the heads potentially responsible for this feature. We then compare the obtained maps to the ones derived using the pre-trained BERT model. This comparison enables us to determine if a particular feature is important for a specific task and whether it contributes to some tasks more than to others.

**Results** Contrary to our initial hypothesis that the vertical attention pattern may be motivated by linguistically meaningful features, we found that it is associated predominantly, if not exclusively, with attention to *[CLS]* and *[SEP]* tokens (see

[6]Our manually constructed list of negation words consisted of the following words *neither, nor, not, never, none, don't, won't, didn't, hadn't, haven't, can't, isn't, wasn't, shouldn't, couldn't, nothing, nowhere.*

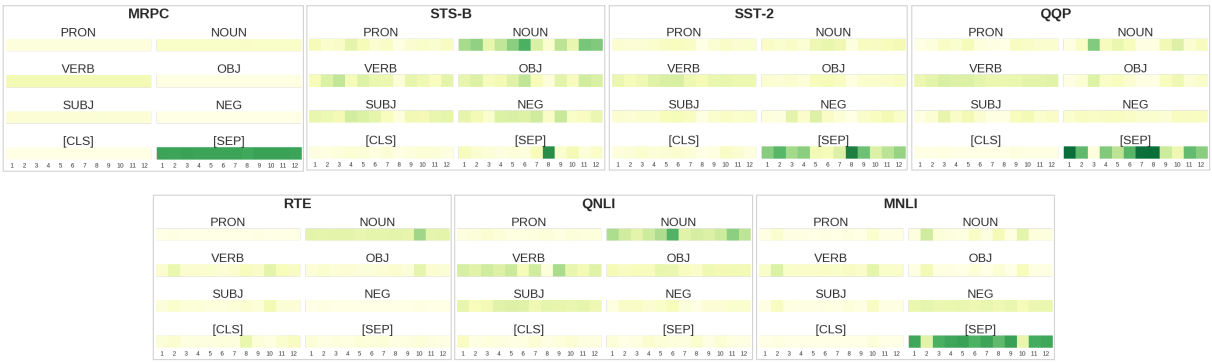


Figure 7: Per-task attention weights corresponding to the *[CLS]* token averaged over input sequences' lengths and over dataset examples, and extracted from the final layer. Darker colors correspond to greater absolute weights.

Figure 6. Note that the absolute *[SEP]* weights for the SST-2 sentiment analysis task are greater than for other tasks, which is explained by the fact that there is only one sentence in the model inputs, i.e. only one *[SEP]* token instead of two. There is also a clear tendency for earlier layers to pay attention to *[CLS]* and for later layers to *[SEP]*, and this trend is consistent across all the tasks. We did detect heads that paid increased (compared to the pre-trained BERT) attention to nouns and direct objects of the main predicates (on the MRPC, RTE and QQP tasks), and negation tokens (on the QNLI task), but the attention weights of such tokens were negligible compared to *[CLS]* and *[SEP]*. Therefore, we believe that the striped attention maps generally come from BERT pre-training tasks rather than from task-specific linguistic reasoning.

### 4.5 Token-to-token attention

To complement the experiments in Sec. 4.4 and 4.2, in this section, we investigate the attention patterns between tokens in the same sentence, i.e. whether any of the tokens are particularly important while a *given* token is being processed. We were interested specifically in the verb-subject relation and the noun-pronoun relation. Also, since BERT uses the representation of the *[CLS]* token in the last layer to make the prediction, we used the features from the experiment in Sec. 4.4 in order to check if they get higher attention weights while the model is processing the *[CLS]* token.

**Results** Our token-to-token attention experiments for detecting heads that prioritize noun-pronoun and verb-subject links resulted in a set of potential head candidates that coincided with diagonally structured attention maps. We believe that this happened due to the inherent property of English syntax where the dependent elements frequently appear close to each other, so it is difficult to distinguish such relations from the previous/following token attention coming from language model pre-training.

Our investigation of attention distribution for the *[CLS]* token in the output layer suggests that for most tasks, with the exception of STS-B, RTE and QNLI, the *[SEP]* gets attended the most, as shown in Figure 7. Based on manual inspection, for the mentioned remaining tasks, the greatest attention weights correspond to the punctuation tokens, which are in a sense similar to *[SEP]*.

### 4.6 Disabling self-attention heads

Since there does seem to be a certain degree of specialization for different heads, we investigated the effects of disabling different heads in BERT and the resulting effects on task performance. Since BERT relies heavily on the learned attention weights, we define disabling a head as modifying the attention values of a head to be constant $a = \frac{1}{L}$ for every token in the input sentence, where $L$ is the length of the sentence. Thus, every token receives the same attention, effectively disabling the learned attention patterns while maintaining the information flow of the original model. Note that by using this framework, we can disable an arbitrary number of heads, ranging from a single head per model to the whole layer or multiple layers.

**Results** Our experiments suggest that certain heads have a detrimental effect on the overall performance of BERT, and this trend holds for all the chosen tasks. Unexpectedly, disabling some heads leads *not* to a drop in accuracy, as one would expect, but to an increase in performance. This is ef-

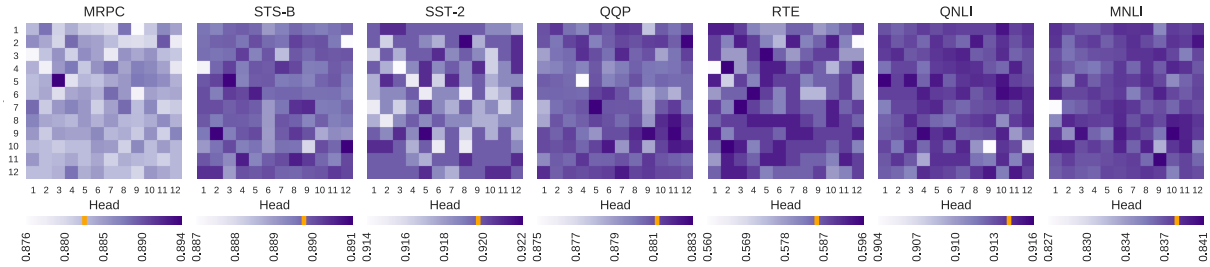


Figure 8: Performance of the model while disabling one head at a time. The orange line indicates the baseline performance with no disabled heads. Darker colors correspond to greater performance scores.

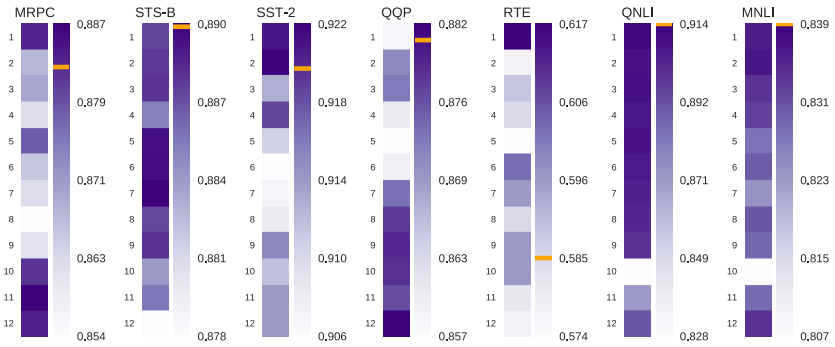


Figure 9: Performance of the model while disabling one layer (that is, all 12 heads in this layer) at a time. The orange line indicates the baseline performance with no disabled layers. Darker colors correspond to greater performance scores.

fect is different across tasks and datasets. While disabling some heads improves the results, disabling the others hurts the results. However, it is important to note that across all tasks and datasets, disabling some heads leads to an increase in performance. The gain from disabling a single head is different for different tasks, ranging from the minimum absolute gain of 0.1% for STS-B, to the maximum of 1.2% for MRPC (see Figure 8). In fact, for some tasks, such as MRPC and RTE, disabling a *random* head gives, on average, *an increase* in performance. Furthermore, disabling a whole layer, that is, all 12 heads in a given layer, also improves the results. Figure 9 shows the resulting model performance on the target GLUE tasks when different layers are disabled. Notably, disabling the first layer in the RTE task gives a significant boost, resulting in an absolute performance gain of 3.2%. However, effects of this operation vary across tasks, and for QNLI and MNLI, it produces a performance drop of up to -0.2%.

## 5 Discussion

In general, our results suggest that even the smaller base BERT model is significantly overparametrized. This is supported by the discovery of repeated self-attention patterns in different heads, as well as the fact that disabling both single and multiple heads is not detrimental to model performance and in some cases even improves it.

We found no evidence that attention patterns that are mappable onto core frame-semantic relations actually improve BERT's performance. 2 out of 144 heads that seem to be "responsible" for these relations (see Section 4.2) do not appear to be important in any of the GLUE tasks: disabling of either one does not lead to a drop of accuracy. This implies that fine-tuned BERT does not rely on this piece of semantic information and prioritizes other features instead.

For instance, we noticed that both STS-B and RTE fine-tuned models rely on attention in the same pair of heads (head 1 in the fourth layer, and head 12 in the second layer), as shown in Figure 8. We manually checked the attention maps in those heads for a set of random inputs, and established that both of them have high weights for words that appear in both sentences of the input examples. This most likely means that word-by-word comparison of the two sentences provides a solid strategy of making a classification prediction for STS-B and RTE. We were not able to find a conceptu-

ally similar interpretation of heads important for other tasks.

## 6 Conclusion

In this work, we proposed a set of methods for analyzing self-attention mechanisms of BERT, comparing attention patterns for the pre-trained and fine-tuned versions of BERT.

Our most surprising finding is that, although attention is the key BERT's underlying mechanism, the model can benefit from attention "disabling". Moreover, we demonstrated that there is redundancy in the information encoded by different heads and the same patterns get consistently repeated regardless of the target task. We believe that these two findings together suggest a further direction for research on BERT interpretation, namely, model pruning and finding an optimal sub-architecture reducing data repetition.

One of the directions for future research would be to study self-attention patterns in different languages, especially verb-final langauges and those with free word order. It is possible that English has relatively lower variety of self-attention patterns, as the subject-predicate relation happens to coincide with the following-previous token pattern.

## 7 Acknowledgments

We thank David Donahue for the help with image annotation. This project is funded in part by the NSF award number IIS-1844740.

## References

Ashutosh Adhikari, Achyudh Ram, Raphael Tang, and Jimmy Lin. 2019. Rethinking complex neural network architectures for document classification. In *Proceedings of NAACL 2019: Conference of the North American Chapter of the Association for Computational Linguistics*.

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*, pages 86–90. Association for Computational Linguistics.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Jonathan Frankle and Michael Carbin. 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*.

Yoav Goldberg. 2019. Assessing bert's syntactic abilities. *arXiv preprint arXiv:1901.05287*.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *57th Annual Meeting of the Association for Computational Linguistics (ACL)*, Florence, Italy.

Nelson F Liu, Matt Gardner, Yonatan Belinkov, Matthew Peters, and Noah A Smith. 2019. Linguistic knowledge and transferability of contextual representations. *arXiv preprint arXiv:1903.08855*.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? *arXiv preprint arXiv:1905.10650*.

Alec Radford, Karthik Narasimhan, Time Salimans, and Ilya Sutskever. 2018. Improving language understanding with unsupervised learning. Technical report, Technical report, OpenAI.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Alexey Romanov and Chaitanya Shivade. 2018. Lessons from natural language inference in the clinical domain. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1586–1596.

Carson T. Schutze. 1996. *The Empirical Base of Linguistics: Grammaticality Judgments and Linguistic Methodology*. University of Chicago Press.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Gongbo Tang, Mathias Müller, Annette Rios, and Rico Sennrich. 2018. Why self-attention? a targeted evaluation of neural machine translation architectures. *arXiv preprint arXiv:1808.08946*.

Ke Tran, Arianna Bisazza, and Christof Monz. 2018. The importance of being recurrent for modeling hierarchical structure. *arXiv preprint arXiv:1803.03585*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *arXiv preprint arXiv:1905.09418*.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems. *arXiv:1905.00537 [cs]*.

Alex Wang, Amapreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122.

Felix Wu, Angela Fan, Alexei Baevski, Yann N Dauphin, and Michael Auli. 2019. Pay less attention with lightweight and dynamic convolutions. *arXiv preprint arXiv:1901.10430*.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328.

# Machine Translation With Weakly Paired Documents

**Lijun Wu**[1], **Jinhua Zhu**[2], **Fei Gao**[3], **Di He**[4], **Tao Qin**[5], **Jianhuang Lai**[1], **Tie-Yan Liu**[5]
[1]School of Data and Computer Science, Sun Yat-sen University;
[2] University of Science and Technology of China;
[3] Institute of Computing Technology, Chinese Academy of Sciences;
[4] Peking University, [5] Microsoft Research Asia;
[1]{wulijun3@mail2, stsljh@mail}.sysu.edu.cn,
[2]teslazhu@mail.ustc.edu.cn, [3]gaofei17b@ict.ac.cn,
[4]di_he@pku.edu.cn, [5]{taoqin, tyliu}@microsoft.com

## Abstract

Neural machine translation, which achieves near human-level performance in some languages, strongly relies on the large amounts of parallel sentences, which hinders its applicability to low-resource language pairs. Recent works explore the possibility of unsupervised machine translation with monolingual data only, leading to much lower accuracy compared with the supervised one. Observing that weakly paired bilingual documents are much easier to collect than bilingual sentences, e.g., from Wikipedia, news websites or books, in this paper, we investigate training translation models with weakly paired bilingual documents. Our approach contains two components. 1) We provide a simple approach to mine implicitly bilingual sentence pairs from document pairs which can then be used as supervised training signals. 2) We leverage the topic consistency of two weakly paired documents and learn the sentence translation model by constraining the word distribution-level alignments. We evaluate our method on weakly paired documents from Wikipedia on six tasks, the widely used WMT16 German↔English, WMT13 Spanish↔English and WMT16 Romanian↔English translation tasks. We obtain 24.1/30.3, 28.1/27.6 and 30.1/27.6 BLEU points separately, outperforming previous results by more than 5 BLEU points in each direction and reducing the gap between unsupervised translation and supervised translation up to 50%.

## 1 Introduction

Neural machine translation (NMT) is a great success of deep learning for natural language processing (Koehn et al., 2003; Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2015; Jean et al., 2015; Vaswani et al., 2017; Gehring et al., 2017). It has significantly outperformed statistical machine translation and reached near human-level performance for several language pairs (Wu et al., 2016; Hassan et al., 2018). Such breakthroughs heavily depend on the availability of large scale bilingual sentence pairs. Taking WMT14 English→French task as an example, NMT uses 38 million parallel sentence pairs for training (Vaswani et al., 2017). As such bilingual sentence pairs are costly to collect, the success of NMT is not fully realized on the vast majority of language pairs, especially for low-resource languages. Recently, Artetxe et al. (2017); Lample et al. (2017) tackle this challenge by training NMT models using only monolingual data, which achieves considerably good accuracy but still far away from that of the state-of-the-art supervised models.

While it is costly to collect bilingual sentence pairs by human translation, we notice that there exists many weakly paired bilingual documents on the Web. For example, for the entity "machine learning", Wikipedia has multiple articles in different languages (e.g., the English and German articles). Though the two articles have very similar content, they are not sentence-by-sentence translations since they may be independently created by different people. Similarly, an English news in BBC and a Chinese news in China Daily talk about the same event but may have differences in details. Furthermore, a popular novel in different languages is usually liberal translation instead of literal translation. We call such weakly aligned documents *weakly paired bilingual documents*. In this paper, we explore learning NMT models from such weakly paired documents, which has several advantages. First, weakly paired documents are much easier to obtain than bilingual sentence pairs. We can obtain bilingual document pairs from Wikipedia pages, aligned news articles on internet, or even from books. Second, such weakly paired documents have great coverage of different



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4375–4384,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

languages. For example, Wikipedia covers more than 178 languages and most of them have paired pages to English. This means that learning translation models from paired bilingual documents are possible for many language pairs.

Weakly aligned document pairs can be utilized for NMT training from two aspects. First, although such two documents are not exactly sentence-by-sentence translations, it is possible that one specific sentence in one document is the translation of one sentence in the other document. Such a sentence pair can be used as bilingual signals for training. For example in Wikipedia, although the web structure and paragraphs are generally different for entity "Beijing" in English and "Pekin" in French, we find the first sentences of the pages are quite semantically similar, in which both of them define Beijing as the capital of China (*English: Beijing, formerly romanized as Peking, is the capital of the People's Republic of China. French: Pekin, galement appele Beijing, est la capitale de la Rpublique populaire de Chine.*) The challenge is how to mine such sentence pairs from those document pairs. Second, although the sentences in two weakly paired documents are not aligned, the topics of the documents are well aligned. Such topic alignment is a strong signal that can be used to train NMT models.

In this paper, we propose a method to train NMT models by leveraging weakly paired bilingual documents from Wikipedia. The key idea contains two aspects: a) Mining implicitly aligned sentence pairs. We provide a simple and efficient method to mine bilingual sentence pairs from weakly aligned document pairs. With the cross-lingual word embeddings trained from two monolingual corpora (one for each language) using MUSE (Conneau et al., 2017), we use weighted average of word embeddings in a sentence as the sentence embedding. We then select the sentence pairs using the sentence embeddings with large cosine similarity as bilingual sentence pairs to train NMT models. b) Leveraging topic alignment as regularization. Many previous works suggest that the word distribution can be usecd to well characterize the topic of the document (Petterson et al., 2010; Du et al., 2015; Funatsu et al., 2014; Pedrosa et al., 2016; Chemudugunta et al., 2007). To leverage the topic consistency between weakly paired documents, we minimize the KL-divergence of the word distributions between the ground-truth document and the model-generated document.

Taking Wikipedia corpus as the training data, we test our method on the widely used WMT16 German↔English, WMT13 Spanish↔English and WMT16 Romanian↔English translation tasks. Our method achieves 24.1/30.3 BLEU points for WMT16 German↔English translations, 28.1/27.6 BLEU points for WMT13 Spanish↔English translations and 30.1/27.6 BLEU scores for WMT16 Romanian↔English translations, outperforming the previous strong unsupervised method by more than 5 BLEU points and reduce the gap between unsupervised translation and supervised translation up to 50%.

## 2 Our method

We contribute two ways to leverage weakly paired documents: mining implicitly aligned sentence pairs from the document pairs and aligning the topic distributions of two documents in a weakly aligned pair. Before diving into details, we first introduce some notations.

Denote $D = \{(d_i^X, d_i^Y)\}, i \in \{1, 2, ..., M\}$ as the set of weakly paired documents, in which document $d_i^X$ is aligned to $d_i^Y$. For example, $d_i^X$ and $d_i^Y$ are two cross-lingual linked Wikipedia pages. Denote $n_i^X$ and $n_i^Y$ as the number of sentences in document $d_i^X$ and $d_i^Y$ respectively. Note that usually $n_i^X \neq n_i^Y$.

Without any confusions, we denote $x$ as a sentence in language $X$ and $y$ as a sentence in language $Y$. We denote *enc* as the encoder for language $X$ and $Y$, which maps a sentence $x$ or $y$ into a sequence of real vectors using parameter $\theta_{enc}$. We use *dec* with parameter $\theta_{dec}$ as the decoder, which takes the encoded vectors and target language tag ($X$ or $Y$) as inputs and outputs a probability distribution over sentences in the target language. Let $\theta$ denote all the parameters of the translation model. Similar to (Artetxe et al., 2017; Lample et al., 2017), such a model can handle both $X \to Y$ and $Y \to X$ translations.

### 2.1 Mining implicitly aligned sentence pairs

Different language versions of Wikipedia pages about the same entity/event are usually created by different people speaking different native languages, and therefore most sentences in two weakly aligned documents are not aligned. Even though, there is still a small chance that some bilingual sentences are aligned, and we try to

mine such implicitly aligned sentence pairs and use them as supervision for NMT model training.

Many works rely on aligned sentences or bilingual dictionaries to mine the sentence pairs (Adafre and De Rijke, 2006; Yasuda and Sumita, 2008; Fung and Cheung, 2004). For example, Imankulova et al. (2017) extract bilingual sentence pairs from Wikipedia using a well-trained translation model learned from supervised sentence pairs. This method does not work for us since we are under unsupervised scenario and we do not have any aligned bilingual sentence pairs. Instead, our idea is to compute the similarity of two sentences using their cross-lingual sentence embeddings and choose the pairs with large similarity as aligned bilingual sentence pairs.

Sentence embedding is widely used to measure textual similarity in text classification tasks (Arora et al., 2017; Le and Mikolov, 2014; Wieting et al., 2015). (Arora et al., 2017) compute the weighted average of the word embedding in one sentence where the weight depends on word frequency and then project away the weighted average sentence embeddings from their first principal component. This method achieves good performance on a range of monolingual textual similarity task. We extend their method from monolingual sentence embedding to cross-lingual sentence embedding, given the cross-lingual word embeddings are pretrained using MUSE (Conneau et al., 2017). The detailed method is described as follows.

For each word $w$, we denote $e_w$ as the word embedding trained from MUSE (Conneau et al., 2017), $p(w)$ as the estimated frequency from another document and $a$ as a predefined parameter to calculate the weight of word embedding. We denote $\hat{e}_s$ as the weighted average sentence embedding of sentence $s$ and $E$ as the embedding matrix of all the sentences over the monolingual corpus. Then we remove the first principal components $u_1$ of $E$ for every weighted average sentence embedding $\hat{e}_s$. Removing the first principal component from the sentence embeddings make them more diverse and expressive in the embedding space, and thus the resulted embeddings are more effective. We use the resulting embedding $e_s$ as the final sentence embedding, i.e.,

$$\begin{aligned} \hat{e}_s &= \sum_{w \in s} \frac{a}{a + p(w)} e_w, \\ u_1 &\leftarrow PCA(E), \\ e_s &= \hat{e}_s - u_1 u_1^T \hat{e}_s. \end{aligned}$$

Based on the sentence embedding, we estimate the similarity between two sentences in different languages by their cosine similarity $\text{sim}(s^X, s^Y) = \frac{<e_{s^X}, e_{s^Y}>}{\|e_{s^X}\| \|e_{s^Y}\|}$. For each weakly aligned document pair $(d_i^X, d_i^Y)$, we have $n_i^X \times n_i^Y$ pairs of sentences and form a bipartite graph between sentences in two documents where the weight of an edge between two cross-lingual sentences is their cosine similarity score. The goal is to find the most confident edges (sentence pairs) from this weighted bipartite graph. We adopt a greedy selection approach with two constraints: The first constraint is that the weight of a selected edge must be larger than threshold $c_1$, which is to ensure that the two sentences are similar enough. The second constraint is that the weight of a selected edge must be larger than the weights of all other edges connected to these two nodes (sentences) by threshold $c_2$. This ensures that the pair we selected is unique enough. Denote $S = \{(s_j^X, s_j^Y)\}$ as the set of selected sentence pairs. We use those pairs as supervision for model training, i.e., minimizing the negative log-likelihood as below.

$$\begin{aligned} L_p(S;\theta) = &-\frac{1}{|S|} \sum_{(s^X, s^Y) \in S} \log P_{X \to Y}(s^Y | s^X; \theta) \\ &-\frac{1}{|S|} \sum_{(s^X, s^Y) \in S} \log P_{Y \to X}(s^X | s^Y; \theta). \end{aligned}$$

### 2.2 Aligning Topic Distribution

Although cross-lingual linked Wikipedia pages are not aligned in sentences, they are usually aligned in topic distribution because they talk about the same event or entity. For example, the English topical words "politician", "United States" and "president" will appear in the English page for "Donald Trump", and similar topical words in French will appear in the corresponding French page. That is, if we translate an article from English to French sentence-by-sentence, the word distribution of the translated article should be generally similar to the word distribution of the corresponding article in French. Here we leverage the document-level word-distribution alignment to enhance and regularize the training of a translation model.

Given an NMT model, we first translate a document $d_i^X$ by sentence translation and obtain a document $\bar{d}_i^Y$. Then we evaluate the word distributions between the generated document $\bar{d}_i^Y$ and the ground-truth document $d_i^Y$, and use such signal to optimize the model. However, straight-forward

loss design, e.g., KL-divergence or Wasserstein distance between the word distributions of $\bar{d}_i^Y$ and $d_i^Y$ is not differentiable w.r.t. the NMT model, due to the non-differentiable operation (greedy search or beam search) while generating $\bar{d}_i^Y$.

To address this challenge, we need to design some loss function that is smooth with respect to the model parameters. Our proposal is assuming each generated sentence $\hat{s}_{i,k}^Y \in \bar{d}_i^Y$ is fixed and "refeeding" the pair $(s_{i,k}^X, \hat{s}_{i,k}^Y)$ to the model to get the probability distribution over all the words at each position. We calculate the word distribution by averaging word probability distributions over all sentences and positions of the generated document. Mathematically, we have

$$P(w_{i,k,t}^Y | s_{i,k}^X, \hat{s}_{i,k,<t}^Y) \sim P_{X\to Y}(w_t^Y | s_{i,k}^X, \hat{s}_{i,k,<t}^Y; \theta),$$
$$P(w^Y; d_i^X, \theta) \propto \sum_{i,k,t} P(w_t | \hat{s}_{i,k,<t}^Y),$$

where $P(w^Y; d_i^X, \theta)$ "acts" as the model-generated word distribution for document $d_i^X$. Note that this averaged distribution is differentiable to model parameters. The word distribution of the ground-truth document $d_i^Y$ is $P(w^Y; d_i^Y) = \frac{\#w \text{ in } d_i^Y}{\#token \text{ in } d_i^Y}$. We simply use KL-divergence loss as the objective. Then the document alignment loss for $X \to Y$ translation is defined as

$$L_d(D; \theta, X \to Y) = \frac{1}{|D|} \sum_{(d_i^X, d_i^Y) \in D} KL(P(w^Y; d_i^Y) || P(w^Y; d_i^X, \theta)).$$

The corresponding document loss for $Y \to X$ translation can be defined in the same way. The final loss is as follows, which can be optimized using backpropagation thanks to its smoothness.

$$L_d(D; \theta) = L_d(D; \theta, X \to Y) + L_d(D; \theta, Y \to X).$$

### 2.3 Overall algorithm

In addition to the above two ways of using Wikipedia data, the sentences in the weakly paired documents can be used as monolingual data to optimize the losses of the unsupervised machine translation. Therefore, our proposed losses can be combined with the loss functions of unsupervised machine translation. Here we first recap unsupervised machine translation (Lample et al., 2017), and then present our algorithm in Algorithm 1.

The unsupervised machine translation considers two loss functions. Given a monolingual sentence $s$ in language $X/Y$, the denoising auto-encoder loss is defined as $L_{dae} = -\mathbb{E}_{s\sim X}[\log P(s|c(s);\theta)] - \mathbb{E}_{s\sim Y}[\log P(s|c(s);\theta)]$, where $c(.)$ is to drop and swap words in sentence $s$. As for the reconstruction loss, given $s$ in language $X/Y$ and the translated sentence $s'$ in $Y/X$ by model $P_{X\to Y}/P_{Y\to X}$, the reconstruction loss is defined as $L_{rec} = -\mathbb{E}_{s\sim X}[\log P_{Y\to X}(s|s';\theta)] - \mathbb{E}_{s\sim Y}[\log P_{X\to Y}(s|s';\theta)]$.

Denote the combination of the monolingual data in language $X$ and $Y$ as $M$. We define the overall loss on monolingual data $M$ as $L_m(M;\theta) = L_{dae} + L_{rec}$. Finally, the overall training objective of our algorithm is to minimize the following loss function with hyperparameters $\alpha$ and $\beta$:

$$L = L_m(M;\theta) + \alpha L_p(S;\theta) + \beta L_d(D;\theta).$$

**Algorithm 1** Training Algorithm

**Require:** Initial translation model with parameter $\theta$; monolingual dataset $M$, implicitly aligned sentence pairs dataset $S$, weakly paired documents dataset $D$; optimizer *Opt*

1: **while** not converged **do**
2: Randomly sample a mini-batch monolingual sentences from $M$, implicitly aligned sentence pairs from $S$ and weakly paired documents from $D$
3: Calculate loss $L_m$, $L_p$ and $L_d$
4: Update $\theta$ by minimizing the overall objective $L$ using optimizer *Opt*
5: **end while**

## 3 Experiments

We test our method on several benchmark translation tasks. We first describe the data preparation and experimental design, and then present the main results, followed by some deep studies.

### 3.1 Data preparation

Wikimedia offers free copies of all available contents on Wikipedia for multiple languages. We download the language specific Wikipedia contents[1], and use WikiExtractor[2] to extract and clean

[1] For example, we download German Wikipedia contents from https://dumps.wikimedia.org/dewiki/.

[2] https://github.com/attardi/wikiextractor

| Language | #Wiki Documents |
|---|---|
| English | 5,684,240 |
| German | 2,201,782 |
| Spanish | 1,389,469 |
| Romanian | 387,627 |
| **Task** | **#Document Pairs** |
| English-German | 948,631 |
| English-Spanish | 836,564 |
| English-Romanian | 87,289 |

Table 1: Statistics of Wikipedia data, including numbers of documents and weakly paired documents.

the texts. The numbers of Wikipedia documents are listed in Table 1. We then use the sentence tokenizer from toolkit NLTK to generate segmented sentences from Wikipedia documents.

Many Wikipedia pages contain external links to the pages that describe the same entity but in different languages. We extract weakly paired documents using these external links. We filter out a document pair if any document in the pair contains less than 5 sentences. We also remove the sentences longer than 100 words. We conduct experiments on three language pairs: English-German (En-De), English-Spanish (En-Es) and English-Romanian (En-Ro) translations. The former two language pairs are indeed high-resource language pairs while the last is a low-resource language pair. Statistics of the processed Wikipedia documents are provided in Table 1.

We use the monolingual data as in Lample et al. (2017, 2018) together with Wikipedia document pairs to train NMT models. For the En-De task, we use all available sentences from the WMT monolingual News Crawl datasets from year 2007 to 2017 containing about 50 million sentences for each language. For Romanian, it has more than 2 million sentences. For En-Es, we use News Crawl datasets from year 2007 to 2012 containing about 10 million sentences. The translation models are evaluated on *newstest 2016* dataset for En-De and En-Ro, and *newstest 2013* dataset for En-Es which are widely used (Koehn and Knowles, 2017).

### 3.2 Experimental Design

To mine implicitly aligned sentences from weakly paired documents, we use the open-sourced word embeddings trained by Fasttext (Joulin et al., 2016) and use MUSE[3] to build cross-lingual word embeddings. We then generate sentence embeddings with the weighted average of cross-lingual word embeddings and further remove the top-1 principal components of sentence embedding matrix as introduced in Section 2.1. We set the two thresholds $c_1 = 0.7$ and $c_2 = 0.1$ respectively when selecting sentence pairs, and the parameter $a$ to calculate the weight of word embedding is $0.001$. To translate a document $d_i^X$ for topic alignment, we use greedy search.

For the training of translation models, the monolingual datasets, Wikipedia document pairs and mined sentence pairs are jointly processed by BPE (Sennrich et al., 2016b) with $60,000$ codes. For model initialization, we follow (Lample et al., 2018), which uses cross-lingual BPE embeddings to initialize the shared lookup tables, and the cross-lingual BPE embeddings are trained by fastText with embedding dimension $512$, a context windows of size $5$ and $10$ negative samples. We adopt the Transformer (Vaswani et al., 2017) architecture in our experiments. We stack $6$ layers in both the encoder and the decoder. Following (Lample et al., 2018), we share the lookup tables between the encoder and the encoder, and between the source and target languages. The dimension of the hidden state is $512$. The weights $\alpha$ and $\beta$ of the loss functions are set to be $1$ and $0.05$. For training, we use Adam optimizer (Kingma and Ba, 2014) and the same learning rate scheduler as used in (Vaswani et al., 2017). For decoding, we use beam search with beam width $4$ and length penalty $0.6$, and report the case-sensitive BLEU[4] score.

### 3.3 Main Results

Our method is compared with several previous works (Lample et al., 2018, 2017; Yang et al., 2018) in Table 2. We also consider some simple and heuristic ways of using the weakly paired documents as baselines. One baseline, referred as "NMT + First Wikipedia Sentence", is to directly use the first sentence of the aligned documents as an aligned sentence pair to train NMT model. The motivation behind it is that usually the first sentence of a Wikipedia document summarizes the main content of the document, which is more likely to be similar across languages. The second baseline, referred as "NMT + Document Translation", is to treat the weakly aligned doc-

[3] https://github.com/facebookresearch/MUSE

[4] https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl

| Unsupervised Method | En→De | De→En | En→Es | Es→En | En→Ro | Ro→En |
|---|---|---|---|---|---|---|
| Lample et al. (2017) | 9.6 | 13.3 | - | - | - | - |
| Yang et al. (2018) | 10.9 | 14.6 | - | - | - | - |
| NMT (Lample et al., 2018) | 17.2 | 21.0 | 19.7 | 20.0 | 21.2 | 19.5 |
| PBSMT (Lample et al., 2018) | 17.9 | 22.9 | - | - | 22.0 | 23.7 |
| PBSMT + NMT (Lample et al., 2018) | 20.2 | 25.2 | - | - | 25.1 | 23.9 |
| NMT + First Wiki Sentence | 16.3 | 19.3 | 17.3 | 18.3 | 19.4 | 18.1 |
| NMT + Document Translation | 12.0 | 14.9 | 14.5 | 15.3 | 16.8 | 15.7 |
| **Ours** | **24.2** | **30.3** | **28.1** | **27.6** | **30.1** | **27.6** |
| **Supervised NMT** | 33.6 | 38.2 | 33.2 | 32.9 | 32.8 | 35.4 |

Table 2: BLEU scores compared with previous approaches.

uments as two long sentences and use them as a bilingual sentence pair to train NMT models.

From Table 2, our approach achieves BLEU score of 24.1 and 30.3 on En→De and De→En translations respectively. The previous best performance is achieved by (Lample et al., 2018): combining phrase-based approach and neural machine translation together, they achieve 20.2 and 25.2 BLEU scores on En→De and De→En translation with monolingual data only. Our approach outperforms their method by more than 4 and 5 BLEU points on En→De and De→En respectively, with limited weakly paired sentences and documents. For the En-Es, we achieve 28.1 and 27.6 BLEU scores on En→Es and Es→En respectively, with more than 8 and 7 points improvement over unsupervised Transformer baseline models. Similarly, for the En-Ro low-resource task, we achieve 30.1 and 27.6 BLEU scores for En→Ro and Ro→En respectively, which has more than 5 and 4 points improvement compared with previously reported results. One interesting point is that there are only nearly $6k$ aligned sentence pairs mined by our approach for En-Ro[5], but we can still achieve a large BLEU improvement. This result can well demonstrate the effectiveness of our method in real low-resource settings and the sentences mined by our approach are in high quality.

For the two heuristic baselines, we find that they hurt the performance of NMT models with more than 2 points decrease in terms of the BLEU score. The results indicate that the careful utilization of Wikipedia data is important, which can also be verified by the superior performance of our approach.

Furthermore, we report the supervised result in the last row of Table 2. The supervised setting is conducted on the full training set on WMT16 En-De translation with 4.5 million parallel sentences, and WMT13 En-Es translation with 3.8 million parallel sentences. As shown in the table, our approach takes a big step towards the supervised result, and reduce the gap between unsupervised translation and supervised translation up to 50%.

### 3.4 Discussions

To make a clear comparison on sentence pair extraction, we work similarly to existing works in our unsupervised scenario.

As introduced before, previous works usually leverage bilingual pairs to train a model or a classifier, or use a bilingual dictionary to mine sentence pairs. In our unsupervised scenario, the most related initial model for selecting pairs is the unsupervised NMT model. Therefore, we first train an unsupervised NMT model followed Lample et al. (2018) and use this model to mine the sentence pairs. Specifically, we conduct following experiments: a). Similarly to Adafre and De Rijke (2006); Yasuda and Sumita (2008), for each $x$, we generate the translation using the unsupervised NMT model, and select the most similar sentence to the translation (w.r.t. BLEU) as paired data. b). We use the model outputted translation probability $p(y|x)$ as the scoring function, as used in Smith et al. (2010); Munteanu and Marcu (2005), and select sentence pairs with larger probabilities. As the unsupervised NMT model is not good enough, we found the selected sentence pairs were in poor quality and using such poor data does not work well. On De→En task, the BLEU score of the model trained in a) and b) can only reach 23.4 and 19.8 (dropped) respectively. Both show that the trained unsupervised NMT models are not good

[5]Compared with nearly $200k$ aligned sentences pairs for En-De, which can be found in Section 3.5.

| Our Method | En→De | De→En |
|---|---|---|
| with $L_p$ and $L_d$ | 24.2 | 30.3 |
| without $L_d$ | 22.9 | 28.7 |
| without $L_p$ | 18.5 | 23.3 |

Table 3: Ablation study of our approach.

| | English-German | | |
|---|---|---|---|
| $c_1$/$c_2$ | 0.0 | 0.1 | 0.2 |
| 0.70 | 257,947 | 199,965 | 132,403 |
| 0.75 | 100,497 | 84,271 | 58,814 |

Table 4: Number of the mined sentence pairs w.r.t. different thresholds for English-German.

as expected.

Besides, we build a bilingual dictionary based on the titles from cross-lingual Wikipedia pages which we want to use to mine sentence pairs like Fung and Cheung (2004) and evaluate the quality using tool from MUSE. However, the top 1 accuracy of the word translation is poor ($< 20\%$), which makes it unable to select trustable sentence pairs using such a weak lexicon. In contrast, by leveraging our cross-lingual word embedding and unsupervised sentence representation, the selected sentences are much better (see more in Section 3.5). We believe our findings are important to the field of unsupervised machine translation.

### 3.5 Further Studies

In this subsection, we provide an ablation study to our method, check the performance variance with respect to the thresholds for mining implicit sentence pairs, and finally present several cases of mined sentence pairs. The studies are conducted on the En-De translation pair.

**Ablation Study** Our method leverages weakly paired documents in two ways. To better understand the importance of the two ways, we report results from an ablation study in Table 3. From the table, we can see that removing the topic alignment, the accuracy drops with more than 1 BLEU points. Without the implicitly aligned sentence pairs, the accuracy decreases with about 6 BLEU points. These findings clearly demonstrate that both the two ways are important, and both contribute to the improvement of translation accuracy.

**Impact of Sentence Quality** As introduced before, to control the data quality of mined sentence pairs, we set two constraints with threshold $c_1$ and $c_2$. We present the paired sentence number and the BLEU scores with respect to these two thresholds in Table 4 and Figure 1. We vary the threshold $c_1$ in $\{0.70, 0.75\}$ and $c_2$ in $\{0.0, 0.1, 0.2\}$. For $c_1 \leq 0.70$, we observe that the sentence quality is poor, while for $c_1 \geq 0.75$, we can only extract few pairs. As shown in the table and the figure, with more strict constraint, we obtain fewer sentence pairs but with higher quality. We can see that there is a clear trade-off between the data quality and the data number. A good configuration is $c_1 = 0.7$ and $c_2 = 0.1$.

**Case Study of Mined Sentence Pairs** We present three cases of our selected sentence pairs in Table 5. We can see that our approach can mine high-quality pairs, such as the first case, in which one sentence is a good translation of the other one. Besides, our method can select interesting paired sentences with similar, if not the exactly same, semantics. As shown in the second case, the two sentences are almost semantically the same, while the German word *"gewählt"* (*"elected"* in English) is not included in the corresponding English sentence. Also in the last case, the detailed description *"15 Meter hoch"* (which means "15 meters high" in English) in the German sentence is missed in the English sentence. Although they are not exact translations, such sentence pairs are still very helpful for NMT model training, as demonstrated by the results of our method.

## 4 Related Work

Using monolingual data to boost the machine translation performance has attracted a lot of attention (Gulcehre et al., 2015; Sennrich et al., 2016a; Zhang and Zong, 2016; Wu et al., 2018; He et al., 2016; Wu et al., 2017), especially when the bilingual supervision is limited. Sennrich et al. (2016a) propose the back-translation approach which is an effective way to augment the training sentence pairs with ctarget-side monolingual data. He et al. (2016) leverage both source-side and target-side monolingual data in a dual learning framework. However, these methods still require a relatively large amount of labeled bilingual data. Recently, Lample et al. (2017) and Artetxe et al. (2017) make an initial study of unsupervised machine translation, in which the model is trained by the monolingual data only. Lample et al. (2017) leverage two key components to learn unsupervised translation models: 1) suitable initial-

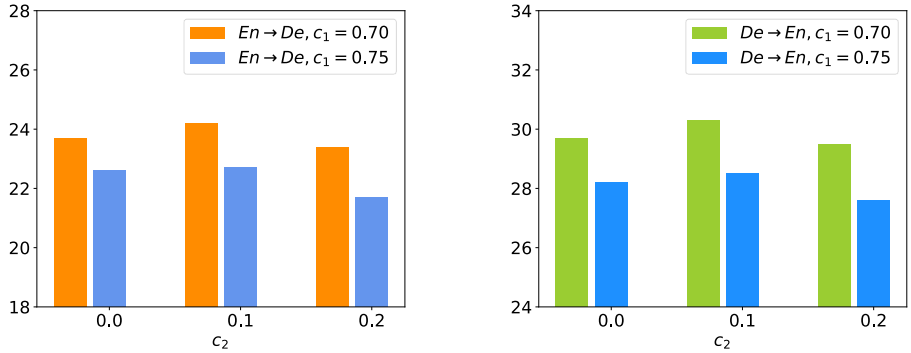


(a) En→De BLEU performance. (b) De→En BLEU performance.

Figure 1: BLEU performance w.r.t. different thresholds for English-German translation pair.

| | |
|---|---|
| **English** | There are several different types of roles to be used in different situations . |
| **German** | Es gibt verschiedene Arten von Rollen , die in unterschiedlichen Situationen eingesetzt werden . |
| **English** | He was a member of the Swedish Academy from 1912 . |
| **German** | 1912 wurde er zum Mitglied der Schwedischen Akademie **gewählt** . |
| **English** | The statue and its marble base stand tall . |
| **German** | Die Statue ist , mit ihrem Sockel aus Marmor , **15 Meter hoch** . |

Table 5: Cases-studies of the sentence pairs mined by our approach for English-German.

ization of the translation model by cross-lingual word embeddings, 2) denoising auto-encoder as language model and reconstruction loss based on translation-back-translation. Both works only leverage monolingual sentences without quite rich weakly paired documents from Web.

We study how to leverage document pairs for training without supervised bilingual sentence pairs from Wikipedia. Leveraging the free online Wikipedia database as an additional source to improve the natural language processing tasks has attracted interest in recent years. For example, Conneau et al. (2017) show that word translation can be effectively learned based on the embedding trained from Wikipedia corpora. This embedding further becomes one of the key components for unsupervised machine translation. Different from using Wikipedia to train warm-start word embeddings, we aim to leverage more and stronger signals from such weakly paired documents to train translation model. Hálek et al. (2011) use the category information in Wikipedia corpus to improve the translation of named entities. Drexler et al. (2014) incorporate language models from target language documents that are comparable to the source documents in Wikipedia pages to improve the document translation.

There are several works aim at extracting potential sentence pairs from comparable corpus, but most of them rely on a set of bilingual sentence pairs to train a model or obtain a bilingual lexicon and use this model/lexicon to select sentence pairs. For example, Adafre and De Rijke (2006) and Yasuda and Sumita (2008) use a strong machine translation system to obtain a rough translation of a given page in one language into another, and then calculate word overlap or BLEU score between sentences as measure. Smith et al. (2010) and Munteanu and Marcu (2005) develop a ranking model/binary classifier to learn how likely a sentence in target language is a translation of the source language using parallel corpora. (Fung and Cheung, 2004) use a small set of parallel corpora to initialize their EM lexical learning and further use this lexicon to iteratively mine sentence pairs. However, in our unsupervised scenario, we have no bilingual sentence pairs to train such a model or lexicon to further select new sentence pairs.

Our work is also related to document translation (Tu et al., 2018) but with different goals and settings. The goal of document translation is to enhance sentence translation with stronger signals beyond sentences by using richer inputs (e.g., the topic information from the documen-

t that contains this sentence). During training, it takes one sentence as well as the cross-sentence (document-level) information as input and predicts the ground-truth translation sentence in other languages. Therefore, training a document translation model requires bilingual sentence pairs and their surrounding contexts. In our scenario, our goal is to learn a sentence translation model with weaker signals than sentence pairs. We target to extract useful information from weakly paired documents to train a translation model without human-labeled bilingual data.

## 5 Conclusion and Future Work

In this work, we propose a general method to train machine translation models using weakly paired bilingual documents from Web, e.g., Wikipedia. Our approach contains two key components: mining implicitly aligned sentence pairs and aligning topic distributions. Experiments on public benchmarks verify the effectiveness of our method.

For future work, we will apply our method to more other language pairs. Besides, we will study using weakly paired documents from other data resources, such as news websites. Furthermore, we will investigate better ways to utilize such weakly paired documents, going beyond mining sentence pairs and aligning topic distributions.

## References


Sisay Fissaha Adafre and Maarten De Rijke. 2006. Finding similar sentences across multiple languages in wikipedia. In *Proceedings of the Workshop on NEW TEXT Wikis and blogs and other dynamic text sources*.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *ICLR*.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2017. Unsupervised neural machine translation. *arXiv preprint arXiv:1710.11041*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *ICLR*.

Chaitanya Chemudugunta, Padhraic Smyth, and Mark Steyvers. 2007. Modeling general and specific aspects of documents with a probabilistic topic model. In *Advances in neural information processing systems*, pages 241–248.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP*, pages 1724–1734. Association for Computational Linguistics.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Jennifer Drexler, Pushpendre Rastogi, Jacqueline Aguilar, Benjamin Van Durme, and Matt Post. 2014. A wikipedia-based corpus for contextualized machine translation. In *LREC*, pages 3593–3596.

Jianguang Du, Jing Jiang, Dandan Song, and Lejian Liao. 2015. Topic modeling with document relative similarities. IJCAI.

Toshiaki Funatsu, Yoichi Tomiura, Emi Ishita, and Kosuke Furusawa. 2014. Extracting representative words of a topic determined by latent dirichlet allocation. in eknow 2014. In *The Sixth International Conference on Information, Process, and Knowledge Management*, pages 112–117.

Pascale Fung and Percy Cheung. 2004. Mining very-non-parallel corpora: Parallel sentence and lexicon extraction via bootstrapping and e. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. *arXiv preprint arXiv:1503.03535*.

Ondrej Hálek, Rudolf Rosa, Ales Tamchyna, and Ondrej Bojar. 2011. Named entities from wikipedia for machine translation. In *ITAT*, pages 23–30.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, et al. 2018. Achieving human parity on automatic chinese to english news translation. *arXiv preprint arXiv:1803.05567*.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tieyan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In *Advances in Neural Information Processing Systems*, pages 820–828.

Aizhan Imankulova, Takayuki Sato, and Mamoru Komachi. 2017. Improving low-resource neural machine translation with filtered pseudo-parallel corpus. In *Proceedings of the 4th Workshop on Asian Translation (WAT2017)*, pages 70–78.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *ACL-IJCNLP*, pages 1–10. Association for Computational Linguistics.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. 2016. Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *NAACL-HLT*, pages 48–54. Association for Computational Linguistics.

Guillaume Lample, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2017. Unsupervised machine translation using monolingual corpora only. *arXiv preprint arXiv:1711.00043*.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Phrase-based & neural unsupervised machine translation. *arXiv preprint arXiv:1804.07755*.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196.

Dragos Stefan Munteanu and Daniel Marcu. 2005. Improving machine translation performance by exploiting non-parallel corpora. *Computational Linguistics*, 31(4):477–504.

Gabriel Pedrosa, Marcelo Pita, Paulo Bicalho, Anisio Lacerda, and Gisele L Pappa. 2016. Topic modeling for short texts with co-occurrence frequency-based expansion. In *2016 5th Brazilian Conference on Intelligent Systems (BRACIS)*, pages 277–282. IEEE.

James Petterson, Wray Buntine, Shravan M Narayanamurthy, Tibério S Caetano, and Alex J Smola. 2010. Word features for latent dirichlet allocation. In *Advances in Neural Information Processing Systems*, pages 1921–1929.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. In *ACL*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *ACL*, volume 1, pages 1715–1725.

Jason R Smith, Chris Quirk, and Kristina Toutanova. 2010. Extracting parallel sentences from comparable corpora using document level alignment. In *NAACL-HLT*, pages 403–411. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Zhaopeng Tu, Yang Liu, Shuming Shi, and Tong Zhang. 2018. Learning to remember translation history with a continuous cache. *Transactions of the Association of Computational Linguistics*, 6:407–420.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

John Wieting, Mohit Bansal, Kevin Gimpel, Karen Livescu, and Dan Roth. 2015. From paraphrase database to compositional paraphrase model and back. *arXiv preprint arXiv:1506.03487*.

Lijun Wu, Fei Tian, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2018. A study of reinforcement learning for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3612–3621.

Lijun Wu, Li Zhao, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2017. Sequence prediction with unlabeled data by reward function learning. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 3098–3104. AAAI Press.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. In *Transactions of the Association for Computational Linguistics*.

Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. 2018. Unsupervised neural machine translation with weight sharing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 46–55.

Keiji Yasuda and Eiichiro Sumita. 2008. Method for building sentence-aligned corpus from wikipedia. In *2008 AAAI Workshop on Wikipedia and Artificial Intelligence (WikiAI08)*, pages 263–268.

Jiajun Zhang and Chengqing Zong. 2016. Exploiting source-side monolingual data in neural machine translation. In *EMNLP*, pages 1535–1545.

# Countering Language Drift via Visual Grounding

**Jason Lee**[†], **Kyunghyun Cho**[†‡⋆], **Douwe Kiela**[‡]
[†] New York University; [⋆] CIFAR Azrieli Global Scholar; [‡] Facebook AI Research
jasonlee@cs.nyu.edu, kyunghyun.cho@nyu.edu, dkiela@fb.com

## Abstract

Emergent multi-agent communication protocols are very different from natural language and not easily interpretable by humans. We find that agents that were initially pretrained to produce natural language can also experience detrimental *language drift*: when a non-linguistic reward is used in a goal-based task, e.g. some scalar success metric, the communication protocol may easily and radically diverge from natural language. We recast translation as a multi-agent communication game and examine auxiliary training constraints for their effectiveness in mitigating language drift. We show that a combination of syntactic (language model likelihood) and semantic (visual grounding) constraints gives the best communication performance, allowing pre-trained agents to retain English syntax while learning to accurately convey the intended meaning.

## 1 Introduction

A long-standing goal of artificial intelligence research is to develop agents that can cooperate with other agents, including humans, to solve tasks. As Gauthier and Mordatch (2016) propose, one way to get closer to this goal is to develop agents that can flexibly use human language to coordinate with themselves and with humans.

Recently, there has been a renewed interest in multi-agent communication (Foerster et al., 2016; Lazaridou et al., 2016). While agents can be very effective in solving the tasks that they were trained on, their multi-agent communication protocols bear little resemblance to human languages. A major open question revolves around training multi-agent systems such that their communication protocols can be interpreted by humans.

One option is to pre-train in a supervised fashion with human language, but even then it is found that the protocols diverge quickly when the agents are fine-tuned on an external reward, as Lewis et al. (2017) showed on a negotiation task. Indeed, language drift is to be expected if we are optimizing for an external non-linguistic reward, such as a reward based on whether or not two agents successfully accomplish a negotiation.

| Intended message: | *2 elephants and 1 lion* |
|---|---|
| No constraints | *floopy globber* |
| Syntactic | *democracy is a political system* |
| Syntactic+Semantic | *a pair of elephants and a large feline* |

Table 1: Examples of valid messages under different constraints. Without constraints, agents may invent an arbitrary protocol to communicate the intended message. Syntactic constraints enforce "Englishness" but not semantic correspondence. Semantic constraints, e.g. with visual grounding, can enforce the communication channel to retain the meaning of the message.

Language drift might be avoided by imposing a "naturalness" constraint, e.g. by factoring language model likelihood into the reward function. However, such a constraint only acts on the *syntax* of the generated language, ignoring its *semantics*. See Table 1 for an illustration of different constraints. As has been advocated by multi-modal semantics (Baroni, 2016; Kiela, 2017), we investigate if appropriate semantic constraints can be imposed on the generated language through (in this case visually) *grounding* its meaning in a different modality.

In order to carefully study this problem, we require a task where drift can be accurately measured. Inspired by Lee et al. (2018), we use a multi-modal machine translation (MMT) dataset (Multi30k; Elliott et al., 2016) to construct a new communication game: Two machine translation agents—i.e., encoder-decoder models with attention—are tasked with successfully translating source language se-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4385–4395,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

quences to the target language using a third pivot language as an intermediary. The first agent's decoder output is fed into the second agent's encoder as input. We employ policy gradient methods to train the first agent with the target language log-likelihood as reward. Thus, we effectively fine-tune two pre-trained machine translation agents via a pivot language, facilitating the study of its drift.

Contrary to alternative two-agent communication tasks such as navigation, game-playing or dialogue—which either don't have clearly defined metrics or easily available natural language data—this pivot-based translation allows us to check exactly whether the communicated sequence corresponds to the intended meaning, as well as to the gold standard sequence. In addition, every single utterance has very clear and well-known metrics such as BLEU and log-likelihood, allowing us to measure performance at every single step.

In what follows, we show that language drift happens, and quite dramatically so, when fine-tuning using policy gradients. Next, we investigate imposing syntactic conformity (i.e., "Englishness") via language model constraints, and show that this does somewhat mitigate drift, but does not lead to semantic correspondence. We then show that additionally imposing semantic constraints via (visual) grounding leads to the best retention of original syntax and intended semantics, and minimizes drift while improving performance. We conduct a token frequency analysis, which corroborates our hypothesis, and show that grounding causes the model to better preserve the token frequency distribution of the pivot language (English), while fine-tuning with language model constraints alone leads to a frequency distribution different from the original natural language.

The ability to keep drift in check opens up exciting possibilities for natural language processing research: we could maximize reward while *retaining* the "Englishness" of the decoder, with obvious benefits for interpretability and interaction with humans. One general use case would be fine-tuning a language model pre-trained on large amounts of data for a given generation task with limited data, which is especially interesting given the recent interest in pre-trained language models (Radford et al., 2019). For instance when training chit-chat dialogue agents, we often want to optimize for some very high-level reward, such as engagingness or consistency, with hardly enough data to learn simple English grammar. The ability to fine-tune a pre-trained independent "language module", without drift, is an exciting prospect. With this work, we aim to take a step in that direction, and show that semantic constraints in the form of grounding play an important role.

## 2 Prior Work

Our work is inspired by recent work in protocols or languages that emerge from multi-agent interaction (Lazaridou et al., 2017; Lee et al., 2018; Andreas et al., 2017; Evtimova et al., 2018; Kottur et al., 2017; Havrylov and Titov, 2017; Mordatch and Abbeel, 2017). Work on the emergence of language in multi-agent settings goes back a long way (Steels, 1997; Nowak and Krakauer, 1999; Kirby, 2001; Briscoe, 2002; Skyrms, 2010). In our case, we are specifically interested in *tabula inscripta* agents that are already pre-trained to generate natural language, and we are primarily concerned with keeping their generated language as natural as possible during further training.

Reinforcement Learning has been applied to fine-tuning models for various natural language generation tasks, including summarization (Ranzato et al., 2015; Paulus et al., 2017), information retrieval (Nogueira and Cho, 2017), MT (Gu et al., 2017; Bahdanau et al., 2016) and dialogue (Li et al., 2017). Our work can be viewed as fine-tuning MT systems using an intermediary pivot language. In MT, there is a long line of work of pivot-based approaches, most notably Muraki (1986) and more recently with neural approaches (Wang et al., 2017; Cheng et al., 2017; Chen et al., 2018). There has also been work on using visual pivots directly (Hitschler et al., 2016; Nakayama and Nishida, 2017; Lee et al., 2018). Grounded language learning in general has been shown to give significant practical improvements in various natural language understanding tasks (Gella et al., 2017; Elliott and Kádár, 2017; Chrupała et al., 2015; Kiela et al., 2017; Kádár et al., 2018).

## 3 Task and Models

### 3.1 Communication Task

We recast pivot-based translation as a communication game involving two MT agents: Fr→En and En→De. See Figure 1. Our dataset consists of $N$ triples of aligned sentences $\{\text{Fr}_k, \text{En}_k, \text{De}_k\}_{k=1}^{N}$. Note that $\text{En}_k$ is only used for evaluation and is not required for training. We first feed the

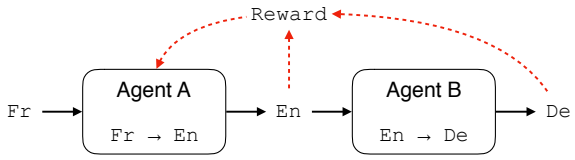


Figure 1: Diagram of our communication game.

French sentence $\text{Fr}_k$ to Agent A, which generates an English message $\overline{\text{En}_k}$ as output. Agent B is then trained to maximize the log-likelihood of the ground truth German sentence given the English message, i.e. $\log p(\text{De}_k|\overline{\text{En}_k})$. Agent A is trained using REINFORCE (Williams, 1992) with reward $R = \log p_B(\text{De}_k|\overline{\text{En}_k})$.[1] This encourages Agent A to develop helpful communication policies for Agent B, and allows Agent B to adapt to Agent A's new policies. In other words: communication via the pivot language (English) is a success if we are able to translate the intended source sequence (French) into the desired target sequence (German).

Both agents are pre-trained on their respective tasks before communication, which means that the intermediate language starts off as English in the early stages of the communication game, where the goal is to translate French to German. This work examines what happens to the intermediate language as we fine-tune the system jointly for the given goal: will the agents keep communicating in English, or diverge? And if so, what can we do to prevent that from happening?

This particular task and setup directly addresses the problem of language drift, as the availability of ground truth references and well-understood metrics (e.g. BLEU) allows us to exactly measure the degree of language drift over time. The Fr→En→De BLEU score informs communication success, while (the relative change in) the Fr→En BLEU score captures the degree of language drift.

### 3.2 Constraints via Auxiliary Tasks

The action space of Agent A is $|V|^L$, where $|V|$ is the size of the vocabulary (approximately 20k) and $L$ is the sequence length. We explore the two aforementioned constraints: a syntactic constraint via language modeling (LM) and a semantic constraint via grounding (G).

**Language Model (LM)** Given a language model pre-trained on a standard English corpus, the (sentence-level) log-likelihood of the English message informs its general "Englishness". We incorporate this into the reward for Agent A, so that it learns to send messages that are plausible English.[2] Reward for Agent A is:

$$R_k^{\text{LM}} = \log p_B(\text{De}_k|\overline{\text{En}_k}) + \beta_{LM} \log p_{LM}(\overline{\text{En}_k}).$$

**Grounding Model (G)** Let us assume we have access to a set of images $\{\text{Img}_k\}$ associated with each triple $\{\text{Fr}_k, \text{En}_k, \text{De}_k\}$. Given a pre-trained image-caption retrieval model, such as VSE++ (Faghri et al., 2018), the log-likelihood of the image given the English message (and vice versa) captures how much the English message is grounded in the original semantic content (Kiela et al., 2017). We incorporate the ranking loss into Agent A's reward.

$$R_k^{\text{G}} = \log p_B(\text{De}_k|\overline{\text{En}_k}) + \beta_G \log p_G(\text{Img}_k|\overline{\text{En}_k}).$$

$\beta_{LM}$ and $\beta_G$ are hyperparameters.

### 3.3 Training Objective

Let us denote the $t$-th token in the $k$-th English training example with $\text{En}_k^t$, the actual reward and the state-dependent baseline in the $k$-th training example as $R_k$ and $\overline{R_k^t}$.

**Policy Gradient Training** At decoding timestep $t$, Agent A takes an action (outputs token $\overline{\text{En}_k^t}$) given an environment (previous hidden states and previous token $\overline{\text{En}_k^{t-1}}$). It receives reward $R_k$ at the end of the sequence, from which we subtract a state-dependent baseline $\overline{R_k^t}$ to reduce variance. Therefore, we maximize $(R_k - \overline{R_k^t}) \log p(\overline{\text{En}_k^t}|\overline{\text{En}_k^{<t}}, \text{Fr}_k)$. In addition, we employ entropy regularization on Agent A's decoder to encourage exploration. Hence, Agent A's overall objective function is given as:

$$\begin{aligned}\mathbb{L}_A = \sum_{k=1}^{N} \Bigg\{ \sum_{t=1}^{T_k} \Big\{ & \alpha_{\text{entr}} H\left(p(\overline{\text{En}_k^t}|\overline{\text{En}_k^{<t}}, \text{Fr}_k)\right) \\ & - \alpha_{\text{b}} \text{MSE}(R_k, \overline{R_k^t}) \\ & + \alpha_{\text{pg}} \left(R_k - \overline{R_k^t}\right) \log p(\overline{\text{En}_k^t}|\overline{\text{En}_k^{<t}}, \text{Fr}_k) \Big\} \Bigg\},\end{aligned}$$

[1] We use subscript $B$ to denote that the probability is computed with Agent B.

[2] We also experimented with a dense LM reward on the word-level, but found this to lead to worse performance. We hypothesize that the model might be focusing too much on the dense LM reward, ignoring the sparse reward for the communication task and leading to poor performance. We did not use BLEU as it is a corpus-level metric.

where $H$ and MSE denote entropy and mean squared error losses. $T_k$ is the maximum decoding timestep in the $k$-th training example.

**Cross Entropy Training** Agent B is trained using standard cross entropy loss, i.e.

$$\mathbb{L}_B = \sum_{k=1}^{N} \Big\{ \sum_{t=1}^{T_k} \log p(\mathrm{De}_k^t | \mathrm{De}_k^{<t}, \overline{\mathrm{En}_k}) \Big\}.$$

We jointly train both agents by maximizing $\mathbb{L} = \mathbb{L}_A + \mathbb{L}_B$.

## 4 Experimental Settings

In this section we provide the details of our experimental setup: a Fr→X→De translation task where the intermediate language X is initialized as English, and subsequently fine-tuned with policy gradient methods. The model is trained either with no constraints (**PG**), syntactic constraints via language modeling (**PG+LM**) or both syntactic and semantic constraints via language modeling and grounding (**PG+LM+G**).

**Datasets** Agents A and B are initially pre-trained on IWSLT Fr→En and En→De, respectively (Cettolo et al., 2012). Fine-tuning is performed on Multi30k Task 1 (Elliott et al., 2016). That is, importantly, there is no overlap in the pre-training data and the fine-tuning data. Multi30k Task 1 consists of 30k images and one caption per image in English, French, German and Czech (of which we only use the first three). To ensure our findings are robust, we compare four different language models, trained on WikiText103, MS COCO, Flickr30k and all of the above.

The grounding model is trained on Flickr30k (Young et al., 2014). Following Faghri et al. (2018), we randomly crop training images for data augmentation. We use 2048-dimensional features from a pretrained and fixed ResNet-152 (He et al., 2016).

**Preprocessing** The same tokenization and vocabulary are used across different tasks and datasets. We lowercase and tokenize our corpora with Moses (Koehn et al., 2007) and use subword tokenization with Byte Pair Encoding (BPE) (Sennrich et al., 2016) with 10k merge operations. This allows us to use the same vocabulary across different models seamlessly (translation, language model, image-caption ranker model).

**Controling the English message length** When fine-tuning the agents, we observe that the length of English messages becomes excessively long. As Agent A has no explicit incentive to output the end-of-sentence (EOS) symbol, it tends to keep transmitting the same token repeatedly. While redundancy might be beneficial for communication, excessively long messages obscure evaluation of the communication protocol. For instance, BLEU score quickly deteriorates as the message length becomes longer, as it is a precision metric. When the message length is fixed, a drop in BLEU score will by necessity mean that the intermediate language has drifted away more. For this reason, we constrain the length of English messages to be no longer than the length of their French source sentence, or shorter if the model outputs the EOS symbol early. Recall that Agent B is supervised to predict the EOS symbol at the right position, so does not suffer from this issue.

**Model Architecture and Pretraining** Our MT agents are standard sequence-to-sequence models with attention (Bahdanau et al., 2015) with a unidirectional, 1-layer GRU (Cho et al., 2014) with 256 hidden units and 256-dimensional embeddings. During initial pre-training on IWSLT, we early-stop on the validation BLEU score (tst2013). The best checkpoints give 34.05 BLEU and 21.94 BLEU on IWSLT Fr→En and En→De development sets with greedy decoding. For our policy gradient value function, we use a 2-layer MLP with ReLU activations.

The language model is a 1-layer recurrent language model with 512 LSTM hidden units. The image-caption retrieval model is a recently proposed VSE++ model (Faghri et al., 2018), with a unidirectional 1-layer GRU with 512 hidden units and a single fully connected layer from 2048-dimensional ResNet features to 512-dimensional GRU hidden states.

**Training Details** When fine-tuning our agents, we perform learning rate annealing and early stopping based on Fr→En→De BLEU (communication performance) on the Multi30k development set. We use Adam (Kingma and Ba, 2014) with an initial learning rate of 0.001 and dropout (Srivastava et al., 2014) rate of 0.1. We grid search over the learning rate schedule and the reward coefficients ($\alpha_{\text{pg}}, \alpha_{\text{entr}}, \alpha_{\text{b}}$) for agent A and ($\beta_{LM}, \beta_G$) for agent B, respectively (see previous section). For

our joint systems with policy gradient fine-tuning, we run every model three times with different random seeds and report averaged results.

**Baseline and Upper Bound** Our main quantitative experiment has three baselines:

- Pretrained models : models pretrained on IWSLT are used without finetuning.
- Ensembling : Given Fr, we let Agent A generate $K$ En hypotheses with beam search. Then, we let Agent B generate the translation $\overline{\text{De}}$ using an ensemble of $K$ source sentences (Firat et al., 2016; Zoph and Knight, 2016).
- Agent A fixed : We fix Agent A (Fr→En) and only fine-tune Agent B using $\mathbb{L}_B$. This shows the communication performance achievable when Agent A cannot drift.

Meanwhile, we also train an NMT model of the same architecture and size directly on the Fr→De task in Multi30k Task 1 (without English intermediary). This serves as an upper bound on the Fr→De performance achievable with available data.

## 5 Quantitative Results

In Table 2, the top three rows are the baselines described above. The pretrained-only baseline performs relatively poorly on Fr→De, conceivably because it was pretrained on a different corpus in a different domain (IWLST dataset is compiled from TED talks, while Multi30k dataset is a collection of image captions). Ensembling multiple English hypotheses for Agent B gives a negligible increase in Fr→De performance. When only Agent B is fine-tuned and Agent A is kept fixed, we observe an increase from 16.30 to 22.37 in Fr→De. Unsurprisingly, the upper bound NMT model directly trained end-to-end on Multi30k Fr→De (without any pivot, at the bottom of the table) performs best.

When the joint system is fine-tuned on German log-likelihood with policy gradients (PG), we observe a large, 8 BLEU increase in Fr→De (16.30→24.51) at the cost of a substantial, 15 BLEU drop in Fr→En (27.18→12.38). This clearly shows that optimizing for external reward may improve performance on that metric, but at the expense of a drastic language drift in the communication channel on which the reward is imposed.

When the system is fine-tuned only on staying grounded but without any language model constraint (PG+G), we obtain small performance improvements. This makes sense, since BLEU first and foremost focuses on the surface form. When the agent is trained with the language model constraint (PG+LM), we notice a significant improvement in Fr→En BLEU. When the LM is trained on WikiText103, a widely used language modeling dataset, we observe an improvement of 9 BLEU score over PG (12.38→21.63). When the training corpus is closer to the target domain, such as MS COCO or Flickr30k, we observe even bigger increases. Fr→De translation also improves by 2–3 BLEU (24.51→26.88-27.67).

We see the biggest improvements in performance when agents are trained using both visual grounding feedback and the language model constraint (PG+LM+G). This is particularly pronounced with the LM trained on WikiText103: introducing visual grounding leads to more than 2 BLEU score improvement in Fr→En (21.63→23.65), and 1 BLEU score improvement in Fr→De (26.88→27.87). We hypothesize that the "Englishness" constraint forces agents to communicate with correct syntax and fluency, while the grounding model restricts the search space of languages to ones that are grounded in visual semantics. To investigate the contribution of grounding, we train a much stronger LM on all three datasets combined, and find that there is still more drift even with access to much more language modeling data (23.60→24.75).

It is important to check that the improvement from grounding is actually significant, so we perform a bootstrapped Wilcoxon signed-rank test (Wilcoxon, 1945) on paired English hypotheses for each reference sentence between PG+LM and PG+LM+G, using the model instance that gives the median communication performance (Fr→En→De BLEU) out of three runs. We assess significance on a bootstrapped test set (repeatedly sampled with replacement) and average the statistic over bootstrap samples. With the threshold of $p < 0.02$, PG+LM+G is found to differ significantly for all the LM models, including the All model that had access to much more data. See Table 3.

Figure 2 shows the learning curves, as measured by Fr→De BLEU (left), Fr→En BLEU (middle) and English LM negative log-likelihood (NLL; right). All models improve in fine-tuned task performance (left plot). We observe that vanilla PG fine-tuning quickly leads to highly "un-English" communication, as can be seen from a distinct in-

| | LM | Ranker | A: Fr→En | A&B: Fr→En→De |
|---|---|---|---|---|
| Pretrained | | | 27.18 | 16.30 |
| Ensembling | | | | 16.95 |
| Agent A fixed | | | 27.18 | 22.37 |
| PG | No LM | | 12.38 (0.67) | 24.51 (1.48) |
| PG+G | No LM | ✔ | 14.20 (1.58) | 26.23 (1.08) |
| PG+LM | WikiText103 | | 21.63 (1.25) | 26.88 (0.12) |
| | MS COCO | | 25.05 (1.40) | 27.66 (0.34) |
| | Flickr30k | | 24.85 (1.14) | 27.60 (0.27) |
| | All | | 23.60 (1.05) | 27.67 (0.39) |
| PG+LM+G | WikiText103 | ✔ | 23.65 (1.91) | 27.87 (0.15) |
| | MS COCO | ✔ | 26.24 (0.28) | 27.86 (0.24) |
| | Flickr30k | ✔ | 25.99 (1.62) | 27.82 (0.41) |
| | All | ✔ | 24.75 (0.40) | 28.08 (0.73) |
| Fr→De | | | | 30.73 |

Table 2: Results in BLEU score on Multi30k Task 1. For our models using policy gradient fine-tuning, we report results averaged over three runs and provide standard deviations in brackets. **PG** (no constraint): trained with vanilla policy gradient fine-tuning. **PG+G** (semantic): trained with grounding only. **PG+LM** (syntactic): trained with "Englishness" constraint. For MS COCO and Flickr30k, the LM was trained directly on image captions. **PG+LM+G** (syntactic+semantic): trained with grounding loss as well as the LM loss. Fr→En: degree of intermediate language drift in agent A; lower indicates more drift. Fr→En→De: overall A&B communication performance; higher is better. For LM=All the LM was trained on all three LM datasets combined.

| | | PG+LM | | | |
|---|---|---|---|---|---|
| | No LM | WikiText103 | MS COCO | Flickr30k | All |
| +G | ✔ | ✔ | ✔ | ✔ | ✔ |

Table 3: Using the bootstrapped Wilcoxon signed-rank test (Wilcoxon, 1945), Fr→En results of PG+LM+G are found to be significantly different from its baselines in all cases considered (on all LM datasets) within the threshold of $p = 0.02$.

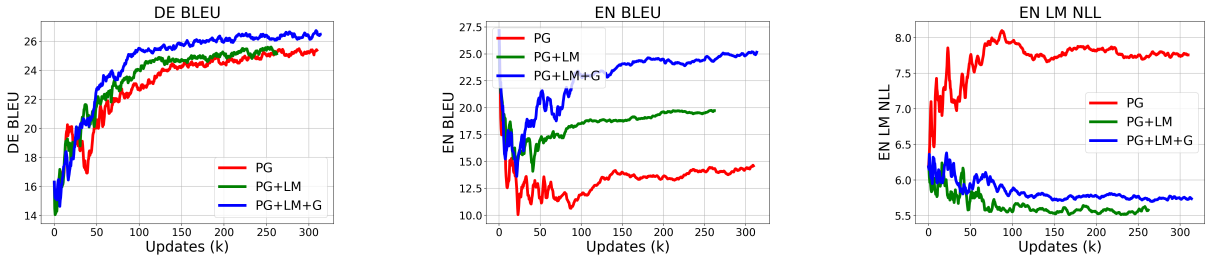


Figure 2: Test set performance over time. En LM NLL curves show the NLL of English messages, computed by a language model trained on WikiText103. Lower En BLEU and higher En LM NLL indicate more language drift.

crease in LM negative log-likelihood (right plot). While PG+LM achieves slightly lower LM NLL than PG+LM+G, its communication protocol drifts much more from English (middle plot). That is, for PG+LM, syntactic conformity is obtained at the expense of semantic preservation. Imposing both syntactic and semantic constraints makes models the least susceptible to drift, almost recovering to the original BLEU score (blue line, middle plot).

## 6 Analysis

A close investigation into the token statistics of each communication strategy reveals that PG fine-tuning causes the word frequency distribution to be flatter (see Figure 3). The PG model has negative frequency difference values for the most frequent tokens, indicating that PG downweighs frequent words severely, possibly because they are less discriminative. On the other hand, PG+LM gives highly positive frequency differences, meaning that language modeling alone disproportionately emphasizes frequent tokens. Using both the LM and grounding constraints keep the token frequencies closest to the pretrained regimes. Investigating the top 20 most frequent words shows that PG+LM disproportionately favors quotation marks, which are very common in many language modeling datasets but rare in Multi30k (see Table 5).

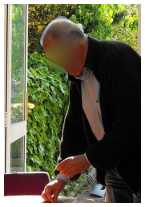

| | | |
|---|---|---|
| Ref | Fr | un vieil homme vêtu d'une veste noire regarde sur la table |
| | De | ein alter mann in einer schwarzen jacke blickt auf den tisch |
| | En | an old man wearing a black jacket is looking on the table |
| En | PG | a old teaching black watching on the table table table table table table |
| | +LM | a old man in a jacket looking on the table . " " |
| | +G | an old man in a black jacket looking on the table . |
| De | PG | ein älterer mann in einem schwarzen hemd schaut auf den tisch . |
| | +LM | ein alter mann in einer jacke beobachtet einen tisch . |
| | +G | ein älterer mann in einer schwarzen jacke schaut auf den tisch . |



| | | |
|---|---|---|
| Ref | Fr | un joueur de football américain en blanc et rouge parle à un entraîneur . |
| | De | ein rot-weiß gekleideter footballspieler spricht mit einem trainer . |
| | En | a football player in red and white is talking to a coach . |
| En | PG | a player football american football american and red talking talking a coach |
| | +LM | a player of white and red talking to a coach . " " " |
| | +G | a football player in white and red talking to a coach . |
| De | PG | ein footballspieler spricht mit einem spieler in einem roten trikot . |
| | +LM | ein weiß gekleideter fußballspieler spricht zu einem trainer . |
| | +G | ein fußballspieler in einem rot-weißen trikot spricht mit einem trainer . |

Table 4: Two communication examples on the data from the Multi30k development set with different models (PG, PG+LM, PG+LM+G). The top three rows list the ground truth sentences, the middle three rows are the English messages sent by the Fr→En agent, and the bottom three rows show the German output from the En→De agent. We also show the corresponding images, which are only used to train the grounding model.

| | |
|---|---|
| Reference | , . the and to of a that i in is it you we 's this " |
| Pretrained | , the . to of and a i that in it we you 's is this " was |
| PG | a the and , . in i " this of to is we you ? that not for |
| PG+LM | the " , of . and in a to this is i es you for we that with |
| PG+LM+G | the , . of a and to in is i this es we for that you at what |

Table 5: Top 20 most frequent tokens (sorted) in English reference (Reference) or the output from Fr→En models.

| | Function words | | | Content words | | | |
|---|---|---|---|---|---|---|---|
| | TO | . | DT | N | V | Adj | Adv |
| PG | .22 | .36 | .57 | .38 | .17 | .32 | .26 |
| PG+LM | .55 | .84 | .72 | .39 | .18 | .21 | .25 |
| PG+LM+G | .62 | .88 | .74 | .43 | .26 | .33 | .29 |

Table 6: Exact-match word recall by POS-tag on IWSLT development set: when the English reference contains a word of a certain POS tag, how often does the agent produce it.

Table 6 compares the degree of drift by part-of-speech, and shows that the PG model has very low recall on function words, such as periods and infinitives. Models trained with LM and grounding losses retain function words with much higher accuracy. PG fares relatively better with content words (nouns and verbs), but adding LM and grounding losses still outperform PG. Grounding leads to overall improvements in recall, particularly with content words. Conceivably, when optimizing Agent

| | IWSLT | | | Multi30k | | |
|---|---|---|---|---|---|---|
| | unique | /sent | /all | unique | /sent | /all |
| Reference | 5,303 | 19.7 | 0.86 | 3,046 | 11.9 | 0.91 |
| Pretrained | 4,657 | 17.9 | 0.85 | 2,867 | 12.0 | 0.87 |
| PG | 4,933 | 13.6 | 0.56 | 3,197 | 9.2 | 0.65 |
| PG+LM | 3,819 | 14.6 | 0.61 | 2,438 | 10.9 | 0.78 |
| PG+LM+G | 4,327 | 15.7 | 0.74 | 2,550 | 10.7 | 0.84 |

Table 7: Additional token frequency analysis. unique: the number of unique English tokens used in the whole development set. /sent: the number of unique English tokens used per sentence. /all: (the number of unique English tokens / the number of all English tokens.)

A's policy on the communication task alone, it is most crucial to relay content information to Agent B, and this might cause agents to ignore syntactic conformity in the original intermediate language. Imposing both syntactic and semantic constraints reduces the space of the intermediate communication protocol to a more stable language space, as reflected in overall task performance.

Table 7 corroborates the finding that vanilla PG fine-tuning leads to flatter token frequency distributions, as the number of unique tokens used by PG is greater than that of the pretrained model. Despite using a more diverse set of tokens, PG uses the smallest number of unique symbols per sentence (/sent) and overall (/all). This implies that PG communication is redundant. PG+LM uses fewer tokens overall, and learns a sharper distribution using a smaller set of high-frequency tokens. Using



| | |
|---|---|
| Fr src | un enfant assis sur un rocher. |
| En ref | a child sitting on a rock formation. |
| En hyp | a punk sitting sitting on on a broken |
| De ref | ein kind sitzt auf einem felsen . |
| De hyp | ein kind sitzt auf einem felsen . |
| Fr src | un petit enfant est assis à une table, en train de manger un goûter. |
| En ref | a toddler is sitting at a table eating a snack . |
| En hyp | a punk sits sitting sitting next next a airline |
| De ref | ein kleines kind sitzt an einem tisch und isst einen snack . |
| De hyp | ein kind sitzt an einem tisch und liest ein buch . |

Table 8: Evidence of token flipping. The agents use the word "punk" to denote "child" or "baby", which is clearly not desirable.

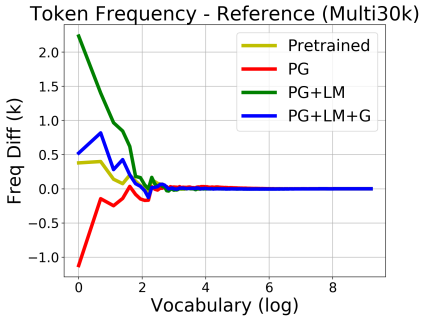


Figure 3: Token frequency analysis on three different models (PG, PG+LM, PG+LM+G) together with the pre-trained model before fine-tuning (Pretrained). We show word frequency curves for each model, after subtracting the reference English frequency statistics (both sorted in decreasing order). Positive y values indicate higher frequency values than the English reference, and negative y values indicate lower frequency values than English. The y-axis is the frequency difference in the thousands, and the x-axis shows the vocabulary index (sorted by frequency) in log scale.

both constraints yields a frequency distribution that most closely resembles the original one.

## 7 Qualitative Results

In the first example of Table 4 (previous page), it is clear that PG's communication messages have significantly diverged from English: the model is highly repetitive ("table table table table table") and misses some key content words such as "man". Agent B, however, correctly generates the German word "mann". This exemplifies a communication protocol that is successful in solving the task it was trained on, but not fully interpretable to humans. While the output from PG+LM is better, the grounded model's message (PG+LM+G) is distinctly the most fluent and semantically correct.

In the second example, observe that the PG Agent B misinterprets "talking talking a coach a coach" as "spricht mit einem spieler" (talking to a player). The PG+LM+G model again generates a flawless English sentence. Furthermore, its agents succeed in communicating both colors (red and white) to German while retaining the original English words, when the other models fail to do so.

Interestingly, we observe some instances of token flipping with the PG model and to a lesser extent with the PG+LM model. For example, one particular model uses "punk" to describe "child" (see Table 8). As no occurrence of "punk" in any training data is associated with "child", the agents must have acquired this new meaning assignment during fine-tuning. Among 35 examples in the Multi30k development set where the English reference contains "child", the model uses "punk" 15 times, indicating this is no random phenomenon. We did not observe such examples with the PG+LM+G model.

## 8 Conclusion

In this paper, we show that language drift happens when fine-tuning natural language agents with some external reward using policy gradients without constraints. We investigate what constraints to put on the communication channel in order to mitigate this. We find that imposing syntactic constraints (via adding language model log-likelihood to the reward) does somewhat mitigate drift, but does not preserve semantic correspondence. We then observe that additionally imposing semantic constraints, e.g. with a perceptual grounding loss, yields communication protocols that best retain the original syntax and intended semantics, while giving the overall best communication performance.

Further analysis into the learned communication protocols reveals that pure PG fine-tuning tends to learn flatter and repetitive token distributions, while encouraging naturalness under a language model disproportionately emphasizes frequent syntactic tokens, yielding a much sharper token distribution than a natural language. The grounded model best retains the original token frequencies.

We examined language drift within a translation game as this allows for direct measurements at each

step (input, intermediate, output), in a way where the semantics stays identical (i.e., the meaning is exactly the same for all languages and modalities) while the communication channel gets only an extrinsic reward (i.e., communication success). The findings in this work, however, are generally applicable to policy gradient fine-tuning of generative language models. We believe that our work shows an intuitive method for addressing language drift and hope that it opens up interesting directions for future work.

## Acknowledgments

We thank the anonymous reviewers for their helpful feedback. We are grateful for support by eBay and NVIDIA. This work was partly supported by Samsung Advanced Institute of Technology (Next Generation Deep Learning: from pattern recognition to AI), Samsung Electronics (Improving Deep Learning using Latent Structure).

## References

Jacob Andreas, Anca D. Dragan, and Dan Klein. 2017. Translating neuralese. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 232–242.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2016. An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Marco Baroni. 2016. Grounding distributional semantics in the visual world. *Language and Linguistics Compass*, 10(1):3–13.

Ted Briscoe. 2002. *Linguistic evolution through language acquisition*. Cambridge University Press.

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. Wit$^3$: Web inventory of transcribed and translated talks. In *Proceedings of the 16$^{th}$ Conference of the European Association for Machine Translation (EAMT)*, pages 261–268, Trento, Italy.

Yun Chen, Yang Liu, and Victor O. K. Li. 2018. Zero-resource neural machine translation with multi-agent communication game. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

Yong Cheng, Qian Yang, Yang Liu, Maosong Sun, and Wei Xu. 2017. Joint training for pivot-based neural machine translation. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 3974–3980.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Grzegorz Chrupała, Akos Kádár, and Afra Alishahi. 2015. Learning language through pictures. *arXiv preprint arXiv:1506.03694*.

Desmond Elliott, Stella Frank, Khalil Sima'an, and Lucia Specia. 2016. Multi30k: Multilingual english-german image descriptions. In *Proceedings of the 5th Workshop on Vision and Language*, pages 70–74.

Desmond Elliott and Akos Kádár. 2017. Imagination improves multimodal translation. *arXiv preprint arXiv:1705.04350*.

Katrina Evtimova, Andrew Drozdov, Douwe Kiela, and Kyunghyun Cho. 2018. Emergent language in a multi-modal, multi-step referential game. In *Proceedings of the International Conference on Learning Representations*.

Fartash Faghri, David J. Fleet, Jamie Kiros, and Sanja Fidler. 2018. VSE++: improving visual-semantic embeddings with hard negatives. In *British Machine Vision Conference*, page 12.

Orhan Firat, Baskaran Sankaran, Yaser Al-Onaizan, Fatos T Yarman Vural, and Kyunghyun Cho. 2016. Zero-resource translation with multi-lingual neural machine translation. *arXiv preprint arXiv:1606.04164*.

Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson. 2016. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2137–2145.

Jon Gauthier and Igor Mordatch. 2016. A paradigm for situated and goal-driven language learning. *arXiv preprint arXiv:1610.03585*.

Spandana Gella, Rico Sennrich, Frank Keller, and Mirella Lapata. 2017. Image pivoting for learning multilingual multimodal representations. *arXiv preprint arXiv:1707.07601*.

Jiatao Gu, Kyunghyun Cho, and Victor O. K. Li. 2017. Trainable greedy decoding for neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1968–1978.

Serhii Havrylov and Ivan Titov. 2017. Emergence of language with multi-agent games: Learning to communicate with sequences of symbols. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems*, pages 2146–2156.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 770–778.

Julian Hitschler, Shigehiko Schamoni, and Stefan Riezler. 2016. Multimodal pivots for image caption translation. *arXiv preprint arXiv:1601.03916*.

Ákos Kádár, Desmond Elliott, Marc-Alexandre Côté, Grzegorz Chrupała, and Afra Alishahi. 2018. Lessons learned in multilingual grounded language learning. *arXiv preprint arXiv:1809.07615*.

Douwe Kiela. 2017. *Deep Embodiment: Grounding Semantics in Perceptual Modalities*. Ph.D. thesis, University of Cambridge, Computer Laboratory.

Douwe Kiela, Alexis Conneau, Allan Jabri, and Maximilian Nickel. 2017. Learning visually grounded sentence representations. *arXiv preprint arXiv:1707.06320*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*.

Simon Kirby. 2001. Spontaneous evolution of linguistic structure-an iterated learning model of the emergence of regularity and irregularity. *IEEE Transactions on Evolutionary Computation*, 5(2):102–110.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*.

Satwik Kottur, José M. F. Moura, Stefan Lee, and Dhruv Batra. 2017. Natural language does not emerge 'naturally' in multi-agent dialog. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2962–2967.

Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. 2016. Multi-agent cooperation and the emergence of (natural) language. *arXiv preprint arXiv:1612.07182*.

Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. 2017. Multi-agent cooperation and the emergence of (natural) language. In *Proceedings of the International Conference on Learning Representations*.

Jason Lee, Kyunghyun Cho, Jason Weston, and Douwe Kiela. 2018. Emergent translation in multi-agent communication. In *Proceedings of the International Conference on Learning Representations*.

Mike Lewis, Denis Yarats, Yann N Dauphin, Devi Parikh, and Dhruv Batra. 2017. Deal or no deal? end-to-end learning for negotiation dialogues. *arXiv preprint arXiv:1706.05125*.

Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2157–2169.

Igor Mordatch and Pieter Abbeel. 2017. Emergence of grounded compositional language in multi-agent populations. *arXiv preprint arXiv:1705.10369*.

Kazunori Muraki. 1986. VENUS: two-phase machine translation system. *Future Generation Comp. Syst.*, 2(2):117–119.

Hideki Nakayama and Noriki Nishida. 2017. Zero-resource machine translation by multimodal encoder–decoder network with multimedia pivot. *Machine Translation*, 31(1-2):49–64.

Rodrigo Nogueira and Kyunghyun Cho. 2017. Task-oriented query reformulation with reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 574–583.

Martin A Nowak and David C Krakauer. 1999. The evolution of language. *Proceedings of the National Academy of Sciences*, 96(14):8028–8033.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Brian Skyrms. 2010. *Signals: Evolution, learning, and information*. Oxford University Press.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Luc Steels. 1997. The synthetic modeling of language origins. *Evolution of communication*, 1(1):1–34.

Yining Wang, Yang Zhao, Jiajun Zhang, Chengqing Zong, and Zhengshan Xue. 2017. Towards neural machine translation with partially aligned corpora. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing*, pages 384–393.

Frank Wilcoxon. 1945. Individual comparisons by ranking methods. *Biometrics bulletin*, 1(6):80–83.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78.

Barret Zoph and Kevin Knight. 2016. Multi-source neural translation. *arXiv preprint arXiv:1601.00710*.

# The Bottom-up Evolution of Representations in the Transformer: A Study with Machine Translation and Language Modeling Objectives

**Elena Voita**[1,2] **Rico Sennrich**[4,3] **Ivan Titov**[3,2]

[1]Yandex, Russia [2]University of Amsterdam, Netherlands
[3]University of Edinburgh, Scotland [4]University of Zurich, Switzerland
lena-voita@yandex-team.ru
sennrich@cl.uzh.ch ititov@inf.ed.ac.uk

## Abstract

We seek to understand how the representations of individual tokens and the structure of the learned feature space evolve between layers in deep neural networks under different learning objectives. We focus on the Transformers for our analysis as they have been shown effective on various tasks, including machine translation (MT), standard left-to-right language models (LM) and masked language modeling (MLM). Previous work used black-box probing tasks to show that the representations learned by the Transformer differ significantly depending on the objective. In this work, we use canonical correlation analysis and mutual information estimators to study how information flows across Transformer layers and how this process depends on the choice of learning objective. For example, as you go from bottom to top layers, information about the past in left-to-right language models gets vanished and predictions about the future get formed. In contrast, for MLM, representations initially acquire information about the context around the token, partially forgetting the token identity and producing a more generalized token representation. The token identity then gets recreated at the top MLM layers.

## 1 Introduction

Deep (i.e. multi-layered) neural networks have become the standard approach for many natural language processing (NLP) tasks, and their analysis has been an active topic of research. One popular approach for analyzing representations of neural models is to evaluate how informative they are for various linguistic tasks, so-called "probing tasks". Previous work has made some interesting observations regarding these representations; for example, Zhang and Bowman (2018) show that untrained LSTMs outperform trained ones on a word identity prediction task; and Blevins et al. (2018) show that up to a certain layer performance of representations obtained from a deep LM improves on a constituent labeling task, but then decreases, while with representations obtained from an MT encoder performance continues to improve up to the highest layer. These observations have, however, been somewhat anecdotal and an explanation of the process behind such behavior has been lacking.

In this paper, we attempt to explain more generally why such behavior is observed. Rather than measuring the quality of representations obtained from a particular model on some auxiliary task, we characterize how the learning objective determines the information flow in the model. In particular, we consider how the representations of individual tokens in the Transformer evolve between layers under different learning objectives.

We look at this task from the information bottleneck perspective on learning in neural networks. Tishby and Zaslavsky (2015) state that "the goal of any supervised learning is to capture and efficiently represent the relevant information in the input variable about the output-label variable" and hence the representations undergo transformations which aim to encode as much information about the output label as possible, while 'forgetting' irrelevant details about the input. As we study sequence encoders and look into representations of individual tokens rather than the entire input, our situation is more complex. In our model, the information preserved in a representation of a token is induced due to two roles it plays: (i) predicting the output label from a current token representation;[1] (ii) preserving information necessary to build representations of other tokens. For example, a language model constructs a representation which is not only useful for predicting an output label (in this case, the next token), but also informative for

[1]We will clarify how we define output labels for LM, MLM and MT objectives in Section 2.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4396–4406, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

producing representations of subsequent tokens in a sentence. This is different from the MT setting, where there is no single encoder state from which an output label is predicted. We hypothesize that the training procedure (or, in our notation, the **task**) defines

1. the nature of changes a token representation undergoes, from layer to layer;

2. the process of interactions and relationships between tokens;

3. the type of information which gets lost and acquired by a token representation in these interactions.

In this work, we study how the choice of objective affects the process by which information is encoded in token representations of the Transformer (Vaswani et al., 2017), as this architecture achieves state-of-the-art results on tasks with very different objectives such as machine translation (MT) (Bojar et al., 2018; Niehues et al., 2018), standard left-to-right language modeling (LM) (Radford et al., 2018) and masked language modeling (MLM) (Devlin et al., 2018). The Transformer encodes a sentence by iteratively updating representations associated with each token starting from a context-agnostic representation consisting of a positional and a token embedding. At each layer token representations exchange information among themselves via multi-head attention and then this information is propagated to the next layer via a feed-forward transformation. We investigate how this process depends on the choice of objective function (LM, MLM or MT) while keeping the data and model architecture fixed.

We start with illustrating the process of information loss and gain in representations of individual tokens by estimating the mutual information between token representations at each layer and the input token identity (i.e. the word type) or the output label (e.g., the next word for LM).

Then, we investigate behavior of token representations from two perspectives: how they influence and are influenced by other tokens. Using canonical correlation analysis, we evaluate the extent of change the representation undergoes and the degree of influence. We reveal differences in the patterns of this behavior for different tasks.

Finally, we study which type of information gets lost and gained in the interactions between tokens and to what extent a certain property is important for defining a token representation at each layer and for each task. As the properties, we consider token identities ('word type'), positions, identities of surrounding tokens and CCG supertags. In these analyses we rely on similarity computations.

We find, that (1) with the LM objective, as you go from bottom to top layers, information about the past gets lost and predictions about the future get formed; (2) for MLMs, representations initially acquire information about the context around the token, partially forgetting the token identity and producing a more generalized token representation; the token identity then gets recreated at the top layer; (3) for MT, though representations get refined with context, less processing is happening and most information about the word type does not get lost. This provides us with a hypothesis for why the MLM objective may be preferable in the pretraining context to LM. LMs may not be the best choice, because neither information about current token and its past nor future is represented well: the former since this information gets discarded, the latter since the model does not have access to the future.

Our key contributions are as follows:

- we propose to view the evolution of a token representation between layers from the compression/prediction trade-off perspective;

- we conduct a series of experiments supporting this view and showing that the two processes, losing information about input and accumulating information about output, take place in the evolution of representations (for MLM, these processes are clearly distinguished and can be viewed as two stages, 'context encoding' and 'token prediction');

- we relate to some findings from previous work, putting them in the proposed perspective, and provide insights into the internal workings of Transformer trained with different objectives;

- we propose an explanation for superior performance of the MLM objective over the LM one for pretraining.

All analysis is done in a model-agnostic manner by investigating properties of token represen-

tations at each layer, and can, in principle, be applied to other multi-layer deep models (e.g., multi-layer RNN encoders).

## 2 Tasks

In this section, we describe the tasks we consider. For each task, we define input $X$ and output $Y$.

### 2.1 Machine translation

Given source sentence $X = (x_1, x_2, ..., x_S)$ and a target sentence $Y = (y_1, y_2, ..., y_T)$, NMT models predict words in the target sentence, word by word, i.e. provide estimates of the conditional distribution $p(y_i|X, y_{1,i-1}, \theta)$.

We train a standard Transformer for the translation task and then analyze its encoder. In contrast to the other two tasks we describe below, representations from the top layers are not directly used to predict output labels but to encode the information which is then used by the decoder.

### 2.2 Language modeling

LMs estimate the probability of a word given the previous words in a sentence $P(x_t|x_1, \ldots, x_{t-1}, \theta)$. More formally, the model is trained with inputs $X = (x_1, \ldots, x_{t-1})$ and outputs $Y = (x_t)$, where $x_t$ is the output label predicted from the final (i.e. top-layer) representation of a token $x_{t-1}$. It is straightforward to apply the Transformer to this task (Radford et al., 2018; Lample and Conneau, 2019).

### 2.3 Masked language modeling

We also consider the MLM objective (Devlin et al., 2018), randomly sampling $15\%$ of the tokens to be predicted. We replace the corresponding input token by [MASK] or a random token in $80\%$ and $10\%$ of the time, respectively, keeping it unchanged otherwise.

For a sentence $(x_1, x_2, ..., x_S)$, where token $x_i$ is replaced with $\tilde{x_i}$, the model receives $X = (x_1, \ldots, x_{i-1}, \tilde{x_i}, x_{i+1}, \ldots, x_S)$ as input and needs to predict $Y = (x_i)$. The label $x_i$ is predicted from the final representation of the token $\tilde{x_i}$.

## 3 Data and Setting

As described below, for a fair comparison, we use the same training data, model architecture and parameter initialization across all the tasks. In order to make sure that our findings are reliable, we also use multiple datasets and repeat experiments with different random initializations for each task.

We train all models on the data from the WMT news translation shared task. We conduct separate series of experiments using two language pairs: WMT 2017 English-German (5.8m sentence pairs) and WMT 2014 English-French (40.8m sentence pairs). For language modeling, we use only the source side of the parallel data. We remove randomly chosen 2.8m sentence pairs from the English-French dataset and use the source side for analysis. English-French models are trained on the remaining 38m sentence pairs. We consider different dataset sizes (2.5m and 5.8m for English-German, 2.5m, 5.8m and 38m for English-French). We find that our findings are true for all languages, dataset sizes and initializations. In the following, all the illustrations are provided for the models trained on the full English-German dataset (5.8m sentence pairs).

We follow the setup and training procedure of the Transformer base model (Vaswani et al., 2017). For details, see the appendix.

## 4 The Information-Bottleneck Viewpoint

In this section, we give an intuitive explanation of the Information Bottleneck (IB) principle (Tishby et al., 1999) and consider a direct application of this principle to our analysis.

### 4.1 Background

The IB method (Tishby et al., 1999) considers a joint distribution of input-output pairs $p(X, Y)$ and aims to extract a compressed representation $\tilde{X}$ for an input $X$ such that $\tilde{X}$ retains as much as possible information about the output $Y$. More formally, the IB method maximizes the mutual information (MI) with the output $I(\tilde{X}; Y)$, while penalizing for MI with the input $I(\tilde{X}; X)$. The latter term in the objective ensures that the representation is indeed a compression. Intuitively, the choice of the output variable $Y$ determines the split of $X$ into irrelevant and relevant features. The relevant features need to be retained while irrelevant ones should be dropped.

Tishby and Zaslavsky (2015) argue that computation in a multi-layered neural model can be regarded as an evolution towards the theoretical optimum of the IB objective. A sequence of layers is viewed as a Markov chain, and the process of obtaining $Y$ corresponds to compressing the representation as it flows across layers and retaining only information relevant to predicting $Y$. This implies that $Y$ defines the information flow in the

model. Since $Y$ is different for each model, we expect to see different patterns of information flow in models, and this is the focus of our study.

### 4.2 IB for token representations

In this work, we view every sequence model (MT, LM and MLM) as learning a function from input $X$ to output $Y$. The input is a sequence of tokens $X = (x_1, x_2, \ldots, x_n)$ and the output $Y$ is defined in Section 2. Recall that we focus on representations of individual tokens in every layer rather than the representation of the entire sequence.

We start off our analysis of divergences in the information flow for different objectives by estimating the amount of information about input or output tokens retained in the token representation at each layer.

#### 4.2.1 Estimating mutual information

Inspired by Tishby and Zaslavsky (2015), we estimate MI between token representations at a certain layer and an input token. To estimate MI, we need to consider token representations at a layer as samples from a discrete distribution. To get such distribution, in the original works (Shwartz-Ziv and Tishby, 2017), the authors binned the neuron's $\arctan$ activations. Using these discretized values for each neuron in a layer, they were able to treat a layer representation as a discrete variable. They considered neural networks with maximum 12 neurons at a layer, but in practical scenarios (e.g. we have 512 neurons in each layer) this approach is not feasible. Instead, similarly to Sajjadi et al. (2018), we discretize the representations by clustering them into a large number of clusters. Then we use cluster labels instead of the continuous representations in the MI estimator.

Specifically, we take only representations of the 1000 most frequent (sub)words. We gather representations for 5 million occurrences of these at each layer for each of the three models. We then cluster the representations into $N = 10000$ clusters using mini-batch $k$-means with $k = 100$. In the experiments studying the mutual information between a layer and source (or target) labels we further filter occurrences. Namely, we take only occurrences where the source and target labels are among the top 1000 most frequent (sub)words.

#### 4.2.2 Results

First, we estimate the MI between an input token and a representation of this token at each layer. In this experiment, we form data for MLM as in the test regime; in other words, the input token is always the same as the output token. Results are shown in Figure 1. For LM, the amount of relevant information about the current input token decreases. This agrees with our expectations: some of the information about the history is intuitively not relevant for predicting the future. MT shows a similar behavior, but the decrease is much less sharp. This is again intuitive: the information about the exact identity is likely to be useful for the decoder. The most interesting and surprising graph is for MLM: first, similarly to other models, the information about the input token is getting lost but then, at two top layers, it gets recovered. We will refer to these phases in further discussion as context encoding and token reconstruction, respectively. Whereas such non-monotonic behavior is impossible when analyzing entire layers, as in Tishby and Zaslavsky (2015), in our case, it suggests that this extra information is obtained from other tokens in the context.

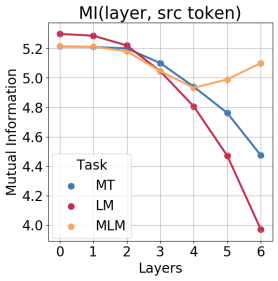


Figure 1: The mutual information between an input token and a representation of this token at each layer.

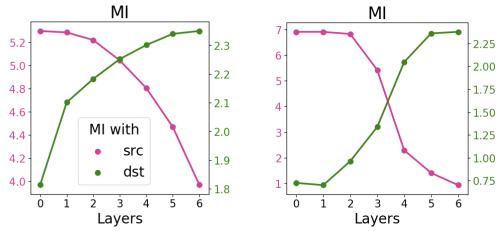


(a) LM (b) MLM

Figure 2: The mutual information of token representations at a layer and source (or target) tokens. For MLM, only tokens replaced at random are considered to get examples where input and output are different.

We perform the same analysis but now measuring MI with the output label for LM and MLM. In this experiment, we form data for MLM as in training, masking or replacing a fraction of tokens. We then take only tokens replaced with a random one to get examples where input and output tokens are different. Results are shown in Figure 2. We can see that, as expected, MI with input tokens

decreases while MI with output tokens increases. Both LM and MLM are trained to predict a token (next for LM and current for MLM) by encoding input and context information. While in Figure 1 we observed monotonic behavior of LM, when looking at the information with both input and output tokens, we can see the two processes, losing information about input and accumulating information about output, for both LM and MLM models. For MLM these processes are more distinct and can be thought of as the context encoding and token prediction (compression/prediction) stages. For MT, since nothing is predicted directly, we see only the encoding stage of this process. This observation relates also to the findings by Blevins et al. (2018). They show that up to a certain layer the performance of representations obtained from a deep multi-layer RNN LM improves on a constituent labeling task, but then decreases, while for representations obtained from an MT encoder performance continues to improve up to the highest layer. We further support this view with other experiments in Section 6.3.

Even though the information-theoretic view provides insights into processes shaping the representations, direct MI estimation from finite samples for densities on multi-dimensional spaces is challenging (Paninski, 2003). For this reason in the subsequent analysis we use more well-established frameworks such as canonical correlation analysis to provide new insights and also to corroborate findings we made in this section (e.g., the presence of two phases in MLM encoding). Even though we will be using different machinery, we will focus on the same two IB-inspired questions: (1) how does information flow across layers? and (2) what information does a layer represent?

## 5 Analyzing Changes and Influences

In this section, we analyze the flow of information. The questions we ask include: how much processing is happening in a given layer; which tokens influence other tokens most; which tokens gain most information from other tokens. As we will see, these questions can be reduced to a comparison between network representations. We start by describing the tool we use.

### 5.1 Canonical Correlation Analysis

We rely on recently introduced projection weighted Canonical Correlation Analysis (PWCCA) (Morcos et al., 2018), which is an improved version of SVCCA (Raghu et al., 2017). Both approaches are based on classic Canonical Correlation Analysis (CCA) (Hotelling, 1936).

CCA is a multivariate statistical method for relating two sets of observations arising from an underlying process. In our setting, the underlying process is a neural network trained on some task. The two sets of observations can be seen as 'two views' on the data. Intuitively, we look at the same data (tokens in a sentence) from two standpoints. For example, one view is one layer and another view is another layer. Alternatively, one view can be $l$-th layer in one model, whereas another view can be the same $l$-th layer in another model. CCA lets us measure similarity between pairs of views.

Formally, given a set of tokens $(x_1, x_2, \ldots, x_N)$ (with the sentences they occur in), we gather their representations produced by two models ($m_1$ and $m_2$) at layers $l_1$ and $l_2$, respectively. To achieve this, we encode the whole sentences and take representations of tokens we are interested in. We get two views of these tokens by the models: $v^{m_1,l_1} = (x_1^{m_1,l_1}, x_2^{m_1,l_1}, \ldots, x_N^{m_1,l_1})$ and $v^{m_2,l_2} = (x_1^{m_2,l_2}, x_2^{m_2,l_2}, \ldots, x_N^{m_2,l_2})$. The representations are gathered in two matrices $X_1 \in M_{a,N}$ and $X_2 \in M_{b,N}$, where $a$ and $b$ are the numbers of neurons in the models.[2] These matrices are then given to CCA (specifically, PWCCA). CCA identifies a linear relation maximizing the correlation between the matrices and computes the similarity. The values of PWCCA range from 0 to 1, with 1 indicating that the observations are linear transformations of each other, 0 indicating no correlation.

In the next sections, we vary two aspects of this process: tokens and the 'points of view'.

### 5.2 A coarse-grained view

We start with the analysis where we do not attempt to distinguish between different token types.

#### 5.2.1 Distances between tasks

As the first step in our analysis, we measure the difference between representations learned for different tasks. In other words, we compare representations for $v^{m_1,l}$ and $v^{m_2,l}$ at different layers $l$. Here the data is all tokens of 5000 sentences. We also quantify differences between representations of models trained with the same objective but different random initializations. The results are pro-

[2] In our experiments, $N > 100000$.

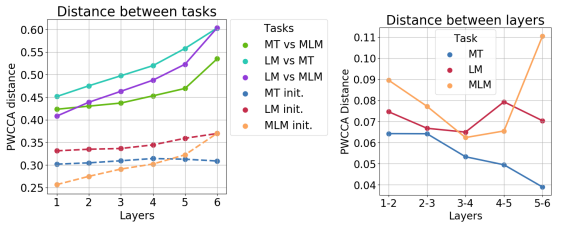


(a) (b)

Figure 3: PWCCA distance (a) between representations of different models at each layer ("init." indicates different initializations), (b) between consecutive layers of the same model.

vided in Figure 3a.

First, differences due to training objective are much larger than the ones due to random initialization of a model. This indicates that PWCCA captures underlying differences in the types of information learned by a model rather than those due to randomness in the training process.

MT and MLM objectives produce representations that are closer to each other than to LM's representations. The reason for this might be twofold. First, for LM only preceding tokens are in the context, whereas for MT and MLM it is the entire sentence. Second, both MT and MLM focus on a given token, as it either needs to be reconstructed or translated. In contrast, LM produces a representation needed for predicting the next token.

#### 5.2.2 Changes between layers

In a similar manner, we measure the difference between representations of consecutive layers in each model (Figure 3b). In this case we take views $v^{m,l}$ and $v^{m,l+1}$ and vary layers $l$ and tasks $m$.

For MT, the extent of change monotonically decreases when going from the bottom to top layers, whereas there is no such monotonicity for LM and MLM. This mirrors our view of LM and especially MLM as undergoing phases of encoding and reconstruction (see Section 4.2), thus requiring a stage of dismissing information irrelevant to the output, which, in turn, is accompanied by large changes in the representations between layers.

### 5.3 Fine-grained analysis

In this section, we select tokens with some predefined property (e.g., frequency) and investigate how much the tokens are influenced by other tokens or how much they influence other tokens.

**Amount of change.** We measure the extent of change for a group of tokens as the PWCCA distance between the representations of these tokens

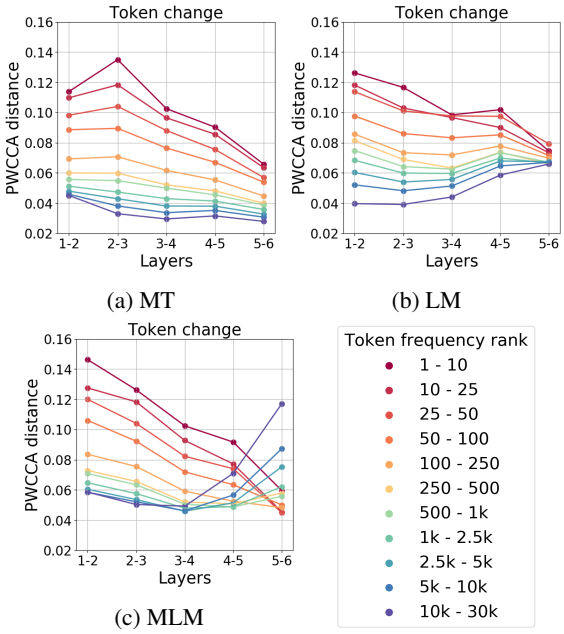


(a) MT (b) LM

(c) MLM

Figure 4: Token change vs its frequency.

for a pair of adjacent layers $(l, l+1)$. This quantifies the amount of information the tokens receive in this layer.

**Influence.** To measure the influence of a token at $l$th layer on other tokens, we measure PWCCA distance between two versions of representations of other tokens in a sentence: first after encoding as usual, second when encoding first $l-1$ layers as usual and masking out the influencing token at the $l$th layer.[3]

#### 5.3.1 Varying token frequency

Figure 4 shows a clear dependence of the amount of change on token frequency. Frequent tokens change more than rare ones in all layers in both LM and MT. Interestingly, unlike MT, for LM this dependence dissipates as we move towards top layers. We can speculate that top layers focus on predicting the future rather than incorporating the past, and, at that stage, token frequency of the last observed token becomes less important.

The behavior for MLM is quite different. The two stages for MLMs could already be seen in Figures 1 and 3b. They are even more pronounced here. The transition from a generalized token representation, formed at the encoding stage, to recreating token identity apparently requires more changes for rare tokens.

When measuring influence, we find that rare tokens generally influence more than frequent ones (Figure 5). We notice an extreme influence of rare

[3] By masking out we mean that other tokens are forbidden to attend to the chosen one.

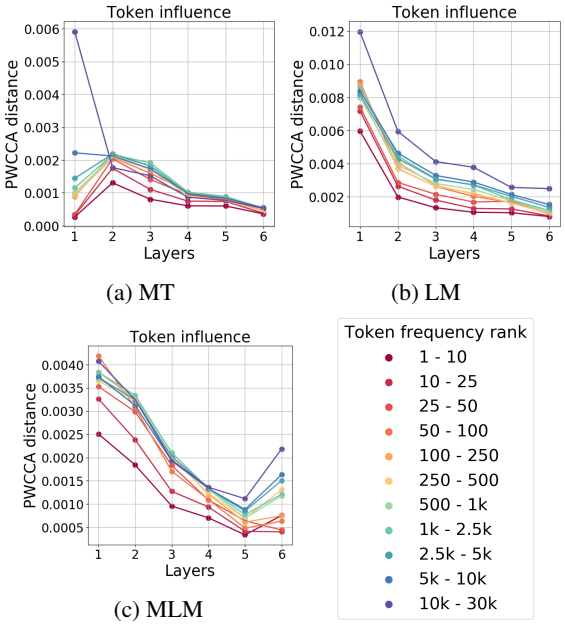


(a) MT (b) LM

(c) MLM

Figure 5: Token influence vs its frequency.

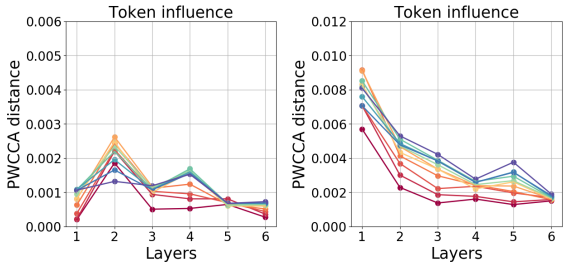


(a) MT (b) LM

Figure 6: Token influence vs its frequency for models trained with word dropout (in training, each input token is replaced with a random with the probability 10%).

tokens at the first MT layer and at all LM layers. In contrast, rare tokens are not the most influencing ones at the lower layers of MLM. We hypothesize that the training procedure of MLM, with masking out some tokens or replacing them with random ones, teaches the model not to over-rely on these tokens before their context is well understood. To test our hypothesis, we additionally trained MT and LM models with token dropout on the input side (Figure 6). As we expected, there is no extreme influence of rare tokens when using this regularization, supporting the above interpretation.

Interestingly, our earlier study of the MT Transformer (Voita et al., 2019) shows how this influence of rare tokens is implemented by the model. In that work, we observed that, for any considered language pair, there is one dedicated attention head in the first encoder layer which tends to point to the least frequent tokens in every sentence. The above analysis suggest that this phenomenon is likely due to overfitting.

We also analyzed the extent of change and influence splitting tokens according to their part of speech; see appendix for details.

## 6 What does a layer represent?

Whereas in the previous section we were interested in quantifying the amount of information exchanged between tokens, here we primarily want to understand what representation in each layer 'focuses' on. We evaluate to what extent a certain property is important for defining a token representation at each layer by (1) selecting a large number of token occurrences and taking their representations; (2) validating if a value of the property is the same for token occurrences corresponding to the closest representations. Though our approach is different from probing tasks, we choose the properties which will enable us to relate to other works reporting similar behaviour (Zhang and Bowman, 2018; Blevins et al., 2018; Tenney et al., 2019a). The properties we consider are token identity, position in a sentence, neighboring tokens and CCG supertags.

### 6.1 Methodology

For our analysis, we take 100 random word types from the top 5,000 in our vocabulary. For each word type, we gather 1,000 different occurrences along with the representations from all three models. For each representation, we take the closest neighbors among representations at each layer and evaluate the percentage of neighbors with the same value of the property.

### 6.2 Preserving token identity and position

In this section, we track the loss of information about token identity (i.e., word type) and position. Our motivation is three-fold. First, this will help us to confirm the results provided on Figure 1; second, to relate to the work reporting results for probing tasks predicting token identity. Finally, the Transformer starts encoding a sentence from a positional and a word embedding, thus it is natural to look at how this information is preserved.

#### 6.2.1 Preserving token identity

We want to check to what extent a model confuses representations of different words. For each of the considered words we add 9000 representations of words which potentially could be confused with

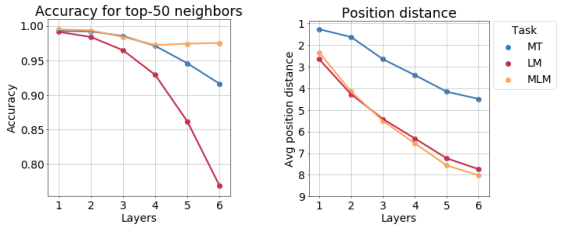


(a) (b)

Figure 7: Preserving (a) token identity, (b) position

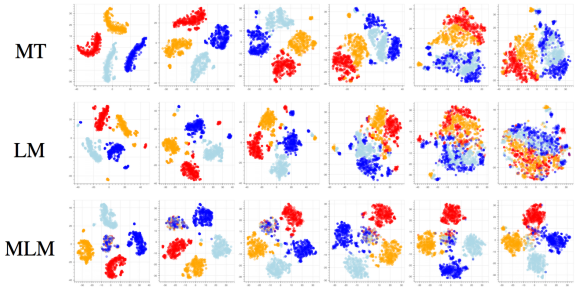


Figure 8: t-SNE of different occurrences of the tokens "is" (red), "are" (orange), "was" (blue), "were" (light-blue). On the x-axis are layers.

it.[4] For this extended set of representations, we follow the methodology described above.

Results are presented in Figure 7a. Reassuringly, the plot is very similar to the one computed with MI estimators (Figure 1), further supporting the interpretations we gave previously (Section 4). Now, let us recall the findings by Zhang and Bowman (2018) regarding the superior performance of untrained LSTMs over trained ones on the task of token identity prediction. They mirror our view of the evolution of a token representation as going through compression and prediction stages, where the learning objective defines the process of forgetting information. If a network is not trained, it is not forced to forget input information.

Figure 8 shows how representations of different occurrences of the words "is", "are", "was", "were" get mixed in MT and LM layers and disambiguated for MLM. For MLM, 15% of tokens were masked as in training. In the first layer, these masked states form a cluster separate from the others, and then they get disambiguated as we move bottom-up across the layers.

#### 6.2.2 Preserving token position

We evaluate the average distance of position of the current occurrence and the top 5 closest representations. The results are provided in Figure 7b. This

[4] See appendix for the details.

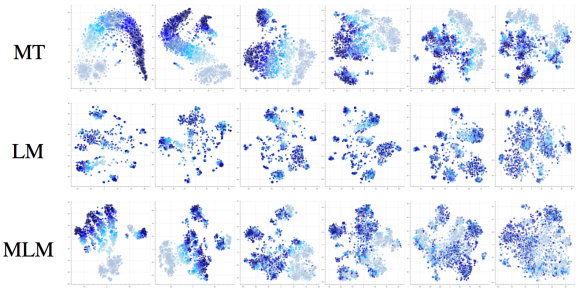


Figure 9: t-SNE of different occurrences of the token "it", position is in color (the larger the word index the darker its color). On the x-axis are layers.

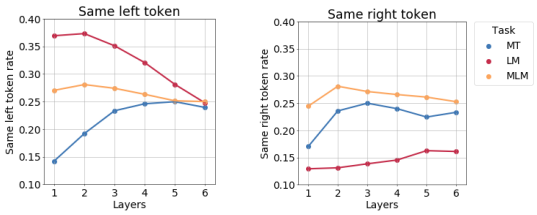


(a) left (b) right

Figure 10: Preserving immediate neighbors

illustrates how the information about input (in this case, position), potentially not so relevant to the output (e.g., next word for LM), gets gradually dismissed. As expected, encoding input positions is more important for MT, so this effect is more pronounced for LM and MLM. An illustration is in Figure 9. For MT, even on the last layer ordering by position is noticeable.

### 6.3 Lexical and syntactic context

In this section, we will look at the two properties: identities of immediate neighbors of a current token and CCG supertag of a current token. On the one hand, these properties represent a model's understanding of different types of context: lexical (neighboring tokens identity) and syntactic. On the other, they are especially useful for our analysis since they can be split into information about 'past' and 'future' by taking either left or right neighbor or part of a CCG tag.

#### 6.3.1 The importance of neighboring tokens

Figure 10 supports our previous expectation that for LM the importance of a previous token decreases, while information about future token is being formed. For MLM, the importance of neighbors gets higher until the second layer and decreases after. This may reflect stages of context encoding and token reconstruction.

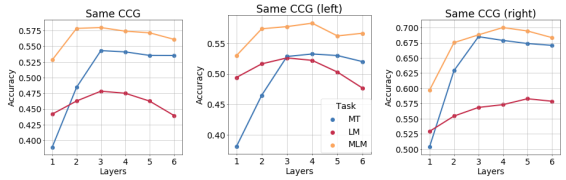


(a) (b) (c)

Figure 11: Preserving CCG supertag.

### 6.3.2 The importance of CCG tags

Results are provided in Figure 11a.[5] As in previous experiments, importance of CCG tag for MLM degrades at higher layers. This agrees with the work by Tenney et al. (2019a). The authors observe that for different tasks (e.g., part-of-speech, constituents, dependencies, semantic role labeling, coreference) the contribution[6] of a layer to a task increases up to a certain layer, but then decreases at the top layers. Our work gives insights into the underlying process defining this behavior.

For LM these results are not really informative since it does not have access to the future. We go further and measure importance of parts of a CCG tag corresponding to previous (Figure 11b) and next (Figure 11c) parts of a sentence. It can be clearly seen that LM first accumulates information about the left part of CCG, understanding the syntactic structure of the past. Then this information gets dismissed while forming information about future.

Figure 12 shows how representations of different occurrences of the token "is" get reordered in the space according to CCG tags (colors correspond to tags).

## 7 Additional related work

Previous work analyzed representations of MT and/or LM models by using probing tasks. Different levels of linguistic analysis have been considered including morphology (Belinkov et al., 2017a; Dalvi et al., 2017; Bisazza and Tump, 2018), syntax (Shi et al., 2016; Tenney et al., 2019b) and semantics (Hill et al., 2017; Belinkov et al., 2017b; Raganato and Tiedemann, 2018; Tenney et al., 2019b). Our work complements this line of research by analyzing how word representations evolve between layers and gives insights into how models trained on different tasks come to represent different information.

Canonical correlation analysis has been previously used to investigate learning dynamics of CNNs and RNNs, to measure the intrinsic dimensionality of layers in CNNs and compare representations of networks which memorize and generalize (Raghu et al., 2017; Morcos et al., 2018). Bau et al. (2019) used SVCCA as one of the methods used for identifying important individual neurons in NMT models. Saphra and Lopez (2019) used SVCCA to investigate how representations of linguistic structure are learned over time in LMs.

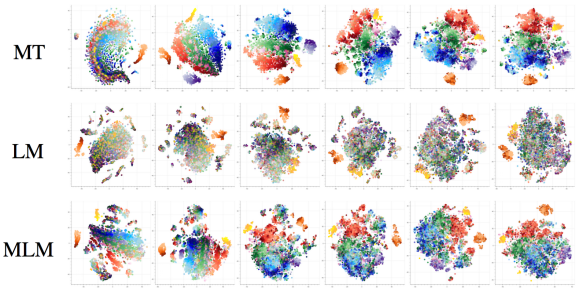


Figure 12: t-SNE of different occurrences of the token "is", CCG tag is in color (intensity of a color is a token position). On the x-axis are layers.

## 8 Conclusions

In this work, we analyze how the learning objective determines the information flow in the model. We propose to view the evolution of a token representation between layers from the compression/prediction trade-off perspective. We conduct a series of experiments supporting this view and propose a possible explanation for superior performance of MLM over LM for pretraining. We relate our findings to observations previously made in the context of probing tasks.

## Acknowledgments

We would like to thank the anonymous reviewers for their comments. The authors also thank Artem Babenko, David Talbot and Yandex Machine Translation team for helpful discussions and inspiration. Ivan Titov acknowledges support of the European Research Council (ERC StG BroadSem 678254) and the Dutch National Science Foundation (NWO VIDI 639.022.518). Rico Sennrich acknowledges support from the Swiss National Science Foundation (PP00P1_176727).

[5] To derive CCG supertags, we use Yoshikawa et al. (2017) tagger, the latest version with ELMO: https://github.com/masashi-y/depccg.

[6] In their experiments, representations are pooled across layers with the scalar mixing technique similar to the one used in the ELMO model (Peters et al., 2018). The probing classifier is trained jointly with the mixing weights, and the learned coefficients are used to estimate the contribution of different layers to a particular task.

## References


Anthony Bau, Yonatan Belinkov, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2019. Identifying and controlling important neurons in neural machine translation. In *International Conference on Learning Representations*, New Orleans.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017a. What do neural machine translation models learn about morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 861–872. Association for Computational Linguistics.

Yonatan Belinkov, Lluís Màrquez, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2017b. Evaluating layers of representation in neural machine translation on part-of-speech and semantic tagging tasks. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10. Asian Federation of Natural Language Processing.

Arianna Bisazza and Clara Tump. 2018. The lazy encoder: A fine-grained analysis of the role of morphology in neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2871–2876, Brussels, Belgium. Association for Computational Linguistics.

Terra Blevins, Omer Levy, and Luke Zettlemoyer. 2018. Deep RNNs encode soft hierarchical syntax. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 14–19, Melbourne, Australia. Association for Computational Linguistics.

Ondřej Bojar, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, and Christof Monz. 2018. Findings of the 2018 conference on machine translation (wmt18). In *Proceedings of the Third Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 272–307, Belgium, Brussels. Association for Computational Linguistics.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Yonatan Belinkov, and Stephan Vogel. 2017. Understanding and improving morphological learning in the neural machine translation decoder. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 142–151. Asian Federation of Natural Language Processing.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Felix Hill, Kyunghyun Cho, Sébastien Jean, and Y Bengio. 2017. The representational geometry of word meanings acquired by neural machine translation models. *Machine Translation*, 31.

Harold Hotelling. 1936. Relations between two sets of variates. *Biometrika*, 28:321–337.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland. Association for Computational Linguistics.

Ari S. Morcos, Maithra Raghu, and Samy Bengio. 2018. Insights on representational similarity in neural networks with canonical correlation. In *NeurIPS*, Montreal, Canada.

Jan Niehues, Ronaldo Cattoni, Sebastian Stüker, Mauro Cettolo, Marco Turchi, and Marcello Federico. 2018. The IWSLT 2018 Evaluation Campaign. In *Proceedings of the 15th International Workshop on Spoken Language Translation*, pages 118–123, Bruges, Belgium.

Liam Paninski. 2003. Estimation of entropy and mutual information. *Neural computation*, 15(6):1191–1253.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Alessandro Raganato and Jörg Tiedemann. 2018. An analysis of encoder representations in transformer-based machine translation. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 287–297, Brussels, Belgium. Association for Computational Linguistics.

Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. 2017. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In *NeurIPS*, Los Angeles.

Mehdi S. M. Sajjadi, Olivier Bachem, Mario Lucic, Olivier Bousquet, and Sylvain Gelly. 2018. Assessing generative models via precision and recall. In

*Advances in Neural Information Processing Systems 31*, pages 5228–5237.

Naomi Saphra and Adam Lopez. 2019. Understanding learning dynamics of language models with SVCCA. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3257–3267, Minneapolis, Minnesota. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural mt learn source syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1526–1534. Association for Computational Linguistics.

Ravid Shwartz-Ziv and Naftali Tishby. 2017. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019a. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R Bowman, Dipanjan Das, et al. 2019b. What do you learn from context? probing for sentence structure in contextualized word representations. In *International Conference on Learning Representations*.

Naftali Tishby, Fernando C Pereira, and William Bialek. 1999. The information bottleneck method. In *Proceedings of the 37-th Annual Allerton Conference on Communication, Control and Computing*, pages 368–377.

Naftali Tishby and Noga Zaslavsky. 2015. Deep learning and the information bottleneck principle. *2015 IEEE Information Theory Workshop (ITW)*, pages 1–5.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008, Los Angeles.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

Masashi Yoshikawa, Hiroshi Noji, and Yuji Matsumoto. 2017. A* CCG parsing with a supertag and dependency factored model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 277–287, Vancouver, Canada. Association for Computational Linguistics.

Kelly Zhang and Samuel Bowman. 2018. Language modeling teaches you more than translation does: Lessons learned through auxiliary syntactic task analysis. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 359–361, Brussels, Belgium. Association for Computational Linguistics.

# Do We Really Need Fully Unsupervised Cross-Lingual Embeddings?

**Ivan Vulić**[1], **Goran Glavaš**[2], **Roi Reichart**[3], **Anna Korhonen**[1]
[1] Language Technology Lab, University of Cambridge
[2] Data and Web Science Group, University of Mannheim
[3] Faculty of Industrial Engineering and Management, Technion, IIT
{iv250,alk23}@cam.ac.uk
goran@informatik.uni-mannheim.de roiri@ie.technion.ac.il

## Abstract

Recent efforts in cross-lingual word embedding (CLWE) learning have predominantly focused on *fully unsupervised* approaches that project monolingual embeddings into a shared cross-lingual space without any cross-lingual signal. The lack of any supervision makes such approaches conceptually attractive. Yet, their only core difference from (weakly) supervised projection-based CLWE methods is in the way they obtain a *seed dictionary* used to initialize an iterative *self-learning* procedure. The fully unsupervised methods have arguably become more robust, and their *primary use case* is CLWE induction for pairs of resource-poor and distant languages. In this paper, we question the ability of even the most robust unsupervised CLWE approaches to induce meaningful CLWEs in these more challenging settings. A series of bilingual lexicon induction (BLI) experiments with 15 diverse languages (210 language pairs) show that fully unsupervised CLWE methods still fail for a large number of language pairs (e.g., they yield zero BLI performance for 87/210 pairs). Even when they succeed, they never surpass the performance of weakly supervised methods (seeded with 500-1,000 translation pairs) using the same self-learning procedure in any BLI setup, and the gaps are often substantial. These findings call for revisiting the main motivations behind fully unsupervised CLWE methods.

## 1 Introduction and Motivation

The wide use and success of monolingual word embeddings in NLP tasks (Turian et al., 2010; Chen and Manning, 2014) has inspired further research focus on the induction of cross-lingual word embeddings (CLWEs). CLWE methods learn a *shared cross-lingual word vector space* where words with similar meanings obtain similar vectors regardless of their actual language. CLWEs benefit cross-lingual NLP, enabling multilingual modeling of meaning and supporting cross-lingual transfer for downstream tasks and resource-lean languages. CLWEs provide invaluable cross-lingual knowledge for, *inter alia*, bilingual lexicon induction (Gouws et al., 2015; Heyman et al., 2017), information retrieval (Vulić and Moens, 2015; Litschko et al., 2019), machine translation (Artetxe et al., 2018c; Lample et al., 2018b), document classification (Klementiev et al., 2012), cross-lingual plagiarism detection (Glavaš et al., 2018), domain adaptation (Ziser and Reichart, 2018), cross-lingual POS tagging (Gouws and Søgaard, 2015; Zhang et al., 2016), and cross-lingual dependency parsing (Guo et al., 2015; Søgaard et al., 2015).

The landscape of CLWE methods has recently been dominated by the so-called *projection-based* methods (Mikolov et al., 2013a; Ruder et al., 2019; Glavaš et al., 2019). They align two monolingual embedding spaces by learning a projection/mapping based on a training dictionary of translation pairs. Besides their simple conceptual design and competitive performance, their popularity originates from the fact that they rely on rather weak cross-lingual supervision. Originally, the seed dictionaries typically spanned several thousand word pairs (Mikolov et al., 2013a; Faruqui and Dyer, 2014; Xing et al., 2015), but more recent work has shown that CLWEs can be induced with even weaker supervision from small dictionaries spanning several hundred pairs (Vulić and Korhonen, 2016), identical strings (Smith et al., 2017), or even only shared numerals (Artetxe et al., 2017).

Taking the idea of reducing cross-lingual supervision to the extreme, the latest CLWE developments almost exclusively focus on *fully unsupervised approaches* (Conneau et al., 2018a; Artetxe et al., 2018b; Dou et al., 2018; Chen and Cardie, 2018; Alvarez-Melis and Jaakkola, 2018; Kim et al., 2018; Alaux et al., 2019; Mohiuddin and Joty, 2019, *inter alia*): they fully abandon any source of



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4407–4418, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

(even weak) supervision and extract the initial seed dictionary by exploiting topological similarities between pre-trained monolingual embedding spaces. Their *modus operandi* can roughly be described by three main components: **C1)** *unsupervised* extraction of a *seed dictionary*; **C2)** a *self-learning* procedure that iteratively refines the dictionary to learn projections of increasingly higher quality; and **C3)** a set of preprocessing and postprocessing steps (e.g., unit length normalization, mean centering, (de)whitening) (Artetxe et al., 2018a) that make the entire learning process more robust.

The induction of fully unsupervised CLWEs is an inherently interesting research topic *per se*. Nonetheless, the main practical motivation for developing such approaches in the first place is to facilitate the construction of multilingual NLP tools and widen the access to language technology for resource-poor languages and language pairs. However, the first attempts at fully unsupervised CLWE induction failed exactly for these use cases, as shown by Søgaard et al. (2018). Therefore, the follow-up work aimed to improve the robustness of unsupervised CLWE induction by introducing more robust self-learning procedures (Artetxe et al., 2018b; Kementchedjhieva et al., 2018). Besides increased robustness, recent work claims that fully unsupervised projection-based CLWEs can even match or surpass their supervised counterparts (Conneau et al., 2018a; Artetxe et al., 2018b; Alvarez-Melis and Jaakkola, 2018; Hoshen and Wolf, 2018; Heyman et al., 2019).

In this paper, we critically examine these claims on robustness and improved performance of unsupervised CLWEs by running a large-scale evaluation in the bilingual lexicon induction (BLI) task on 15 languages (i.e., 210 languages pairs, see Table 2 in §3). The languages were selected to represent different language families and morphological types, as we argue that fully unsupervised CLWEs have been designed to support exactly these setups. However, we show that even the most robust unsupervised CLWE method (Artetxe et al., 2018b) still fails for a large number of language pairs: 87/210 BLI setups are unsuccessful, yielding (near-)zero BLI performance. Further, even when the unsupervised method succeeds, it is because the components C2 (self-learning) and C3 (pre-/post-processing) can mitigate the undesired effects of noisy seed lexicon extraction. We then demonstrate that the combination of C2 and C3 with a small provided seed dictionary (e.g., 500 or 1K pairs) outscores the unsupervised method in *all* cases, often with a huge margin, and does not fail for *any* language pair. Furthermore, we show that the most robust unsupervised CLWE approach still fails completely when it relies on monolingual word vectors trained on domain-dissimilar corpora. We also empirically verify that unsupervised approaches cannot outperform weakly supervised approaches also for closely related languages (e.g., Swedish–Danish, Spanish–Catalan).

While the "no supervision at all" premise behind fully unsupervised CLWE methods is indeed seductive, our study strongly suggests that future research efforts should revisit the main motivation behind these methods and focus on designing even more robust solutions, given their current inability to support a wide spectrum of language pairs. In hope of boosting induction of CLWEs for more diverse and distant language pairs, we make all 210 training and test dictionaries used in this work publicly available at: `https://github.com/ivulic/panlex-bli`.

## 2 Methodology

We now dissect a general framework for unsupervised CLWE learning, and show that the "bag of tricks of the trade" used to increase their robustness (which often slips under the radar) can be equally applied to (weakly) supervised projection-based approaches, leading to their fair(er) comparison.

### 2.1 Projection-Based CLWE Approaches

In short, projection-based CLWE methods learn to (linearly) align independently trained monolingual spaces $\boldsymbol{X}$ and $\boldsymbol{Z}$, using a word translation dictionary $D_0$ to guide the alignment process. Let $\boldsymbol{X}_D \subset \boldsymbol{X}$ and $\boldsymbol{Z}_D \subset \boldsymbol{Z}$ be the row-aligned subsets of monolingual spaces containing vectors of aligned words from $D_0$. Alignment matrices $\boldsymbol{X}_D$ and $\boldsymbol{Z}_D$ are then used to learn orthogonal transformations $\boldsymbol{W}_x$ and $\boldsymbol{W}_z$ that define the joint bilingual space $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{W}_x \cup \boldsymbol{Z}\boldsymbol{W}_z$. While supervised projection-based CLWE models learn the mapping using a provided external (clean) dictionary $D_0$, their unsupervised counterparts automatically induce the seed dictionary in an unsupervised way (C1) and then refine it in an iterative fashion (C2).

**Unsupervised CLWEs.** These methods first induce a seed dictionary $D^{(1)}$ leveraging only two unaligned monolingual spaces (C1). While the

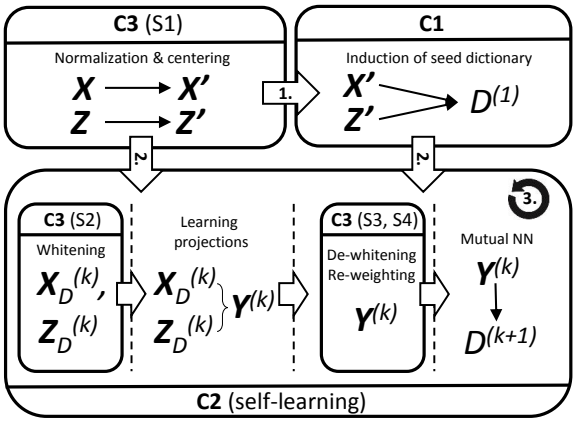


Figure 1: General unsupervised CLWE approach.

algorithms for unsupervised seed dictionary induction differ, they all strongly rely on the assumption of similar topological structure between the two pretrained monolingual spaces. Once the seed dictionary is obtained, the two-step iterative self-learning procedure (C2) takes place: 1) a dictionary $D^{(k)}$ is first used to learn the joint space $\boldsymbol{Y}^{(k)} = \boldsymbol{X}\boldsymbol{W}_x^{(k)} \cup \boldsymbol{Z}\boldsymbol{W}_z^{(k)}$; 2) the nearest neighbours in $\boldsymbol{Y}^{(k)}$ then form the new dictionary $D^{(k+1)}$. We illustrate the general structure in Figure 1.

A recent empirical survey paper (Glavaš et al., 2019) has compared a variety of latest unsupervised CLWE methods (Conneau et al., 2018a; Alvarez-Melis and Jaakkola, 2018; Hoshen and Wolf, 2018; Artetxe et al., 2018b) in several downstream tasks (e.g., BLI, cross-lingual information retrieval, document classification). The results of their study indicate that the VECMAP model of Artetxe et al. (2018b) is by far the most robust and best performing unsupervised CLWE model. For the actual results and analyses, we refer the interested reader to the original paper of Glavaš et al. (2019). Another recent evaluation paper (Doval et al., 2019) as well as our own preliminary BLI tests (not shown for brevity) have further verified their findings. We thus focus on VECMAP in our analyses, and base the following description of the components C1-C3 on that model.

### 2.2 Three Key Components

**C1. Seed Lexicon Extraction.** VECMAP induces the initial seed dictionary using the following heuristic: monolingual similarity distributions for words with similar meaning will be similar across languages.[1] The monolingual similarity distributions for the two languages are given as rows (or columns; the matrices are symmetric) of $\boldsymbol{M}_x = \boldsymbol{X}\boldsymbol{X}^T$ and $\boldsymbol{M}_z = \boldsymbol{Z}\boldsymbol{Z}^T$. For the distributions of similarity scores to be comparable, the values in each row of $\boldsymbol{M}_x$ and $\boldsymbol{M}_z$ are first sorted. The initial dictionary $D^{(1)}$ is finally obtained by searching for mutual nearest neighbours between the rows of $\sqrt{\boldsymbol{M}_x}$ and of $\sqrt{\boldsymbol{M}_z}$.

**C2. Self-Learning.** Not counting the preprocessing and postprocessing steps (component C3), self-learning then *iteratively* repeats two steps:

**1)** Let $\boldsymbol{D}^{(k)}$ be the binary matrix indicating the aligned words in the dictionary $D^{(k)}$.[2] The orthogonal transformation matrices are then obtained as $\boldsymbol{W}_x^{(k)} = \boldsymbol{U}$ and $\boldsymbol{W}_z^{(k)} = \boldsymbol{V}$, where $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ is the singular value decomposition of the matrix $\boldsymbol{X}^T\boldsymbol{D}^{(k)}\boldsymbol{Z}$. The cross-lingual space of the $k$-th iteration is then $\boldsymbol{Y}^{(k)} = \boldsymbol{X}\boldsymbol{W}_x^{(k)} \cup \boldsymbol{Z}\boldsymbol{W}_z^{(k)}$.

**2)** The new dictionary $D^{(k+1)}$ is then built by identifying nearest neighbours in $\boldsymbol{Y}^{(k)}$. These can be easily extracted from the matrix $\boldsymbol{P} = \boldsymbol{X}\boldsymbol{W}_x^{(k)}(\boldsymbol{Z}\boldsymbol{W}_z^{(k)})^T$. *All* nearest neighbours can be used, or additional *symmetry* constraints can be imposed to extract only mutual nearest neighbours: all pairs of indices $(i, j)$ for which $\boldsymbol{P}_{ij}$ is the largest value both in row $i$ and column $j$.

The above procedure, however, often converges to poor local optima. To remedy for this, the second step (i.e., dictionary induction) is extended with techniques that make self-learning more robust. First, the vocabularies of $\boldsymbol{X}$ and $\boldsymbol{Z}$ are cut to the top $k$ most frequent words.[3] Second, similarity scores in $\boldsymbol{P}$ are kept with probability $p$, and set to zero otherwise. This *dropout* allows for a wider exploration of possible word pairs in the dictionary and contributes to escaping poor local optima given the noisy seed lexicon in the first iterations.

**C3. Preprocessing and Postprocessing Steps.** While iteratively learning orthogonal transformations $\boldsymbol{W}_x$ and $\boldsymbol{W}_z$ for $\boldsymbol{X}$ and $\boldsymbol{Z}$ is the central step of unsupervised projection-based CLWE methods, preprocessing and postprocessing techniques are additionally applied before and after the transformation. While such techniques are often over-

[1] For instance, *zwei* and *two* will have similar distributions of similarities over their respective language vocabularies – *zwei* is expected to be roughly as (dis)similar to *drei* and *Katze* as *two* is to *three* and *cat*.

[2] I.e., $\boldsymbol{D}_{ij}^{(k)} = 1 \iff$ the $i$-th word of one language and the $j$-th word of the other are a translation pair in $D^{(k)}$.

[3] This is done to prevent spurious nearest neighbours consisting of infrequent words with unreliable vectors.

looked in model comparisons, they may have a great impact on the model's final performance, as we validate in §4. We briefly summarize two preprocessing (S1 and S2) and post-processing (S3 and S4) steps used in our evaluation, originating from the framework of Artetxe et al. (2018a).

S1) *Normalization and mean centering*. We first apply unit length normalization: all vectors in $\boldsymbol{X}$ and $\boldsymbol{Z}$ are normalized to have a unit Euclidean norm. Following that, $\boldsymbol{X}$ and $\boldsymbol{Z}$ are mean centered dimension-wise and then again length-normalized.

S2) *Whitening*. ZCA whitening (Bell and Sejnowski, 1997) is applied on (S1-processed) $\boldsymbol{X}$ and $\boldsymbol{Z}$: it transforms the matrices such that each dimension has unit variance and that the dimensions are uncorrelated. Intuitively, the vector spaces are easier to align along directions of high variance.

S3) *Dewhitening*. A transformation inverse to S2: for improved performance it is important to restore the variance information after the projection, if whitening was applied in S2 (Artetxe et al., 2018a).

S4) *Symmetric re-weighting*. This step attempts to further align the embeddings in the cross-lingual embedding space by measuring how well a dimension in the space correlates across languages for the current iteration dictionary $D^{(k)}$.[4] The best results are obtained when re-weighting is neutral to the projection direction, that is, when it is applied symmetrically in both languages.

In the actual implementation S1 is applied only once, before self-learning. S2, S3 and S4 are applied in each self-learning iteration.

**Model Configurations.** Note that C2 and C3 can be equally used on top of any (provided) seed lexicon (i.e., $D^{(1)}$:=$D_0$) to enable weakly supervised learning, as we propose here. In fact, the variations of the three key components, C1) seed lexicon, C2) self-learning, and C3) preprocessing and post-processing, construct various model configurations which can be analyzed to probe the importance of each component in the CLWE induction process. A selection of representative configurations evaluated later in §4 is summarized in Table 1.

## 3 Experimental Setup

**Evaluation Task.** Our task is *bilingual lexicon induction* (BLI). It has become the *de facto* standard evaluation for projection-based CLWEs (Ruder et al., 2019; Glavaš et al., 2019). In short, after a shared CLWE space has been induced, the task is to retrieve target language translations for a test set of source language words. Its lightweight nature allows us to conduct a comprehensive evaluation across a large number of language pairs.[5] Since BLI is cast as a ranking task, following Glavaš et al. (2019) we use mean average precision (MAP) as the main evaluation metric: in our BLI setup with only one correct translation for each "query" word, MAP is equal to mean reciprocal rank (MRR).[6]

**(Selection of) Language Pairs.** Our selection of test languages is guided by the following goals: **a)** following recent initiatives in other NLP research (e.g., for language modeling) (Cotterell et al., 2018; Gerz et al., 2018), we aim to ensure the coverage of different genealogical and typological language properties, and **b)** we aim to analyze a large set of language pairs and offer new evaluation data which extends and surpasses other work in the CLWE literature. These two properties will facilitate analyses between (dis)similar language pairs and offer a comprehensive set of evaluation setups that test the robustness and portability of fully unsupervised CLWEs. The final list of 15 diverse test languages is provided in Table 2, and includes samples from different languages types and families. We run BLI evaluations for all language pairs in both directions, for a total of 15×14=210 BLI setups.

**Monolingual Embeddings.** We use the 300-dim vectors of Grave et al. (2018) for all 15 languages, pretrained on Common Crawl and Wikipedia with fastText (Bojanowski et al., 2017).[7] We trim all

[4] More formally, assume that we are working with matrices $\boldsymbol{X}$ and $\boldsymbol{Z}$ that already underwent all transformations described in S1-S3. Another matrix $\boldsymbol{D}$ represents the current bilingual dictionary $D$: $D_{ij} = 1$ if the $i^{th}$ source word is translated by the $j^{th}$ target word and $D_{ij} = 0$ otherwise. Then, given the singular value decomposition $\boldsymbol{USV^T} = \boldsymbol{X^T DZ}$, the final re-weighted projection matrices are $\boldsymbol{W}_x = \boldsymbol{US}^{\frac{1}{2}}$ (and $\boldsymbol{W}_z = \boldsymbol{VS}^{\frac{1}{2}}$. We refer the reader to (Artetxe et al., 2018a) and (Artetxe et al., 2018b) for more details.

[5] While BLI is an intrinsic task, as discussed by Glavaš et al. (2019) it is a strong indicator of CLWE quality also for downstream tasks: relative performance in the BLI task correlates well with performance in cross-lingual information retrieval (Litschko et al., 2018) or natural language inference (Conneau et al., 2018b). More importantly, it also provides a means to analyze whether a CLWE method manages to learn anything meaningful at all, and can indicate "unsuccessful" CLWE induction (e.g., when BLI performance is similar to a random baseline): detecting such CLWEs is especially important when dealing with fully unsupervised models.

[6] MRR is more informative than the more common *Precision@1 (P@1)*; our main findings are also valid when P@1 is used; we do not report the results for brevity.

[7] Experiments with other monolingual vectors such as the

| Configuration | C1 | C2 | C3 |
|---|---|---|---|
| UNSUPERVISED | unsupervised | all tested, we always report the best one | S1-S4 (FULL) |
| ORTHG-SUPER | provided | – | length normalization only (partial S1) |
| ORTHG+SL+SYM | provided | symmetric: mutual nearest neighbours | length normalization only (partial S1) |
| FULL-SUPER | provided | – | S1-S4 (FULL) |
| FULL+SL | provided | (Artetxe et al., 2018b) with dropout | S1-S4 (FULL) |
| FULL+SL+NOD | provided | (Artetxe et al., 2018b) w/o dropout | S1-S4 (FULL) |
| FULL+SL+SYM | provided | symmetric: mutual nearest neighbours, w/o dropout | S1-S4 (FULL) |

Table 1: Configurations obtained by varying components C1, C2, and C3 used in our empirical comparison in §4.

| Language | Family | Type | ISO 639-1 |
|---|---|---|---|
| Bulgarian | IE: Slavic | fusional | BG |
| Catalan | IE: Romance | fusional | CA |
| Esperanto | – (constructed) | agglutinative | EO |
| Estonian | Uralic | agglutinative | ET |
| Basque | – (isolate) | agglutinative | EU |
| Finnish | Uralic | agglutinative | FI |
| Hebrew | Afro-Asiatic | introflexive | HE |
| Hungarian | Uralic | agglutinative | HU |
| Indonesian | Austronesian | isolating | ID |
| Georgian | Kartvelian | agglutinative | KA |
| Korean | Koreanic | agglutinative | KO |
| Lithuanian | IE: Baltic | fusional | LT |
| Bokmål | IE: Germanic | fusional | NO |
| Thai | Kra-Dai | isolating | TH |
| Turkish | Turkic | agglutinative | TR |

Table 2: The list of 15 languages from our main BLI experiments along with their corresponding language family (IE = Indo-European), broad morphological type, and their ISO 639-1 code.

vocabularies to the 200K most frequent words.

**Training and Test Dictionaries.** They are derived from PanLex (Baldwin et al., 2010; Kamholz et al., 2014), which was used in prior work on cross-lingual word embeddings (Duong et al., 2016; Vulić et al., 2017). PanLex currently spans around 1,300 language varieties with over 12M expressions: it offers some support and supervision also for low-resource language pairs (Adams et al., 2017). For each source language ($L_1$), we automatically translate their vocabulary words (if they are present in PanLex) to all 14 target ($L_2$) languages. To ensure the reliability of the translation pairs, we retain only unigrams found in the vocabularies of the respective $L_2$ monolingual spaces which scored above a PanLex-predefined threshold.

As in prior work (Conneau et al., 2018a; Glavaš et al., 2019), we then reserve the 5K pairs created from the more frequent $L_1$ words for training, while the next 2K pairs are used for test. Smaller training dictionaries (1K and 500 pairs) are created by again selecting pairs comprising the most frequent $L_1$ words.

original fastText and skip-gram (Mikolov et al., 2013b) trained on Wikipedia show the same trends in the final results.

**Training Setup.** In all experiments, we set the hyper-parameters to values that were tuned in prior research. When extracting the UNSUPERVISED seed lexicon, the 4K most frequent words of each language are used; self-learning operates on the 20K most frequent words of each language; with dropout the keep probability $p$ is 0.1; CSLS with $k = 10$ nearest neighbors (Artetxe et al., 2018b).

Again, Table 1 lists the main model configurations in our comparison. For the fully UNSUPERVISED model we always report the best performing configuration after probing different self-learning strategies (i.e., +SL, +SL+NOD, and +SL+SYM are tested). The results for UNSUPERVISED are always reported as averages over 5 restarts: this means that with UNSUPERVISED we count BLI setups as unsuccessful only if the results are close to zero in all 5/5 runs. ORTHG-SUPER is the standard supervised model with orthogonal projections from prior work (Smith et al., 2017; Glavaš et al., 2019).

## 4 Results and Discussion

Main BLI results averaged over each source language ($L_1$) are provided in Table 3 and Table 4. We now summarize and discuss the main findings across several dimensions of comparison.

**Unsupervised vs. (Weakly) Supervised.** First, when exactly the same components C2 and C3 are used, UNSUPERVISED is unable to outperform a (weakly) supervised FULL+SL+SYM variant, and the gap in final performance is often substantial. In fact, FULL+SL+SYM and FULL+SL+NOD outperform the best UNSUPERVISED for all 210/210 BLI setups: we observe the same phenomenon with varying dictionary sizes, that is, it equally holds when we seed self-learning with 5K, 1K, and 500 translation pairs, see also Figure 2. This also suggests that the main reason why UNSUPERVISED approaches were considered on-par with supervised approaches in prior work (Conneau et al., 2018a; Artetxe et al., 2018b) is because they were not compared under fair circumstances: while UNSUPER-

| | BG-* | CA-* | EO-* | ET-* | EU-* | FI-* | HE-* | HU-* |
|---|---|---|---|---|---|---|---|---|
| UNSUPERVISED | 0.208 | 0.224 | 0.128 | 0.155 | 0.036 | 0.181 | 0.186 | 0.206 |
| *Unsuccessful setups* | *3/14* | *2/14* | *3/14* | *6/14* | *10/14* | *4/14* | *2/14* | *3/14* |
| 5K:ORTHG-SUPER | 0.258 | 0.237 | 0.201 | 0.210 | 0.151 | 0.233 | 0.198 | 0.259 |
| 5K:ORTHG+SL+SYM | 0.281 | 0.264 | 0.219 | 0.225 | 0.164 | 0.256 | 0.217 | 0.283 |
| 5K:FULL-SUPER | 0.343 | 0.335 | 0.304 | 0.301 | 0.228 | 0.324 | 0.287 | 0.354 |
| 5K:FULL+SL | 0.271 | 0.262 | 0.240 | 0.236 | 0.161 | 0.260 | 0.217 | 0.282 |
| 5K:FULL+SL+NOD | 0.316 | 0.311 | 0.295 | 0.276 | 0.204 | 0.320 | 0.260 | 0.330 |
| 5K:FULL+SL+SYM | **0.361** | **0.356** | **0.336** | **0.316** | **0.244** | **0.348** | **0.294** | **0.374** |
| 1K:ORTHG-SUPER | 0.104 | 0.088 | 0.065 | 0.082 | 0.049 | 0.088 | 0.066 | 0.101 |
| 1K:ORTHG+SL+SYM | 0.203 | 0.167 | 0.106 | 0.157 | 0.079 | 0.168 | 0.133 | 0.191 |
| 1K:FULL-SUPER | 0.146 | 0.129 | 0.098 | 0.117 | 0.065 | 0.117 | 0.096 | 0.143 |
| 1K:FULL+SL | 0.268 | 0.260 | 0.238 | 0.232 | 0.158 | 0.257 | 0.217 | 0.279 |
| 1K:FULL+SL+NOD | 0.312 | 0.307 | 0.284 | 0.272 | 0.197 | 0.311 | 0.255 | 0.327 |
| 1K:FULL+SL+SYM | **0.341** | **0.327** | **0.302** | **0.293** | **0.212** | **0.329** | **0.268** | **0.354** |

| | ID-* | KA-* | KO-* | LT-* | NO-* | TH-* | TR-* | **Avg** |
|---|---|---|---|---|---|---|---|---|
| UNSUPERVISED | 0.110 | 0.106 | 0.001 | 0.179 | 0.239 | 0.000 | 0.133 | 0.140 |
| *Unsuccessful setups* | *7/14* | *6/14* | *14/14* | *4/14* | *3/14* | *14/14* | *6/14* | *87/210* |
| 5K:ORTHG-SUPER | 0.199 | 0.163 | 0.154 | 0.194 | 0.250 | 0.109 | 0.207 | 0.201 |
| 5K:ORTHG+SL+SYM | 0.216 | 0.177 | 0.166 | 0.212 | 0.273 | 0.117 | 0.226 | 0.220 |
| 5K:FULL-SUPER | 0.261 | 0.250 | 0.239 | 0.302 | 0.332 | **0.154** | 0.283 | 0.286 |
| 5K:FULL+SL | 0.180 | 0.191 | 0.152 | 0.217 | 0.274 | 0.056 | 0.204 | 0.214 |
| 5K:FULL+SL+NOD | 0.220 | 0.229 | 0.207 | 0.272 | 0.318 | 0.106 | 0.253 | 0.261 |
| 5K:FULL+SL+SYM | **0.272** | **0.263** | **0.251** | **0.310** | **0.356** | 0.148 | **0.299** | **0.302** |
| 1K:ORTHG-SUPER | 0.069 | 0.050 | 0.040 | 0.067 | 0.099 | 0.034 | 0.068 | 0.071 |
| 1K:ORTHG+SL+SYM | 0.119 | 0.092 | 0.063 | 0.135 | 0.186 | 0.052 | 0.129 | 0.132 |
| 1K:FULL-SUPER | 0.089 | 0.079 | 0.061 | 0.111 | 0.127 | 0.044 | 0.091 | 0.101 |
| 1K:FULL+SL | 0.180 | 0.185 | 0.148 | 0.220 | 0.274 | 0.054 | 0.204 | 0.212 |
| 1K:FULL+SL+NOD | 0.216 | 0.223 | 0.197 | 0.269 | 0.315 | 0.096 | 0.248 | 0.255 |
| 1K:FULL+SL+SYM | **0.243** | **0.237** | **0.203** | **0.284** | **0.337** | **0.103** | **0.274** | **0.274** |

Table 3: BLI scores (MRR) for all model configurations. The scores are averaged over all experimental setups where each of the 15 languages is used as $L_1$: e.g., CA-* means that the translation direction is from Catalan (CA) as source ($L_1$) to each of the remaining 14 languages listed in Table 2 as targets ($L_2$), and we average over the corresponding 14 CA-* BLI setups. $5k$ and $1k$ denote the seed dictionary size for (weakly) supervised methods ($D_0$). *Unsuccessful setups* refer to the number of BLI experimental setups with the fully UNSUPERVISED model that yield an MRR score $\leq 0.01$. The **Avg** column refers to the averaged MRR scores of each model configuration over all 15×14=210 BLI setups. The highest scores for two different seed dictionary sizes in each column are in bold, the second best are underlined. See Table 1 for the brief description of all model configurations in the comparison. Full results for each particular language pair are available in the supplemental material.

| | $\lvert D_0\rvert = 5k$ | | $\lvert D_0\rvert = 1k$ | |
|---|---|---|---|---|
| | **Unsuc.** | **Win** | **Unsuc.** | **Win** |
| UNSUPERVISED | 87 (94) | 0 | 87 (94) | 0 |
| ORTHG-SUPER | 0 (2) | 0 | 2 (82) | 0 |
| ORTHG+SL+SYM | 0 (1) | 0 | 1 (34) | 0 |
| FULL-SUPER | 0 (0) | 46 | 0 (41) | 0 |
| FULL+SL | 0 (7) | 0 | 0 (9) | 0 |
| FULL+SL+NOD | 0 (1) | 7 | 0 (3) | 33 |
| FULL+SL+SYM | 0 (0) | 157 | 0 (0) | 177 |

Table 4: Summary statistics computed over all 15×14=210 BLI setups. **a) Unsuc.** denotes the total number of unsuccessful setups, where a setup is considered unsuccessful if MRR $\leq 0.01$ or MRR $\leq 0.05$ (in the parentheses); **b) Win** refers to the total number of "winning" setups, that is, for all language pairs it counts how many times each particular model yields the best overall MRR score. We compute separate statistics for two settings ($\lvert D_0\rvert = 1k$ and $\lvert D_0\rvert = 5k$).

VISED relied heavily on the components C2 and C3, these were omitted when running supervised baselines. Our unbiased comparison reveals that there is a huge gap even when supervised projection-based approaches consume only several hundred translation pairs to initiate self-learning.

**Are Unsupervised CLWEs Robust?** The results also indicate that, contrary to the beliefs established by very recent work (Artetxe et al., 2018b; Mohiuddin and Joty, 2019), fully UNSUPERVISED approaches are still prone to getting stuck in local optima, and still suffer from robustness issues when dealing with distant language pairs: 87 out of 210 BLI setups ($= 41.4\%$) result in (near-)zero BLI performance, see also Table 4. At the same time, weakly supervised methods with a seed lexicon of 1k or 500 pairs do not suffer from the robustness problem and always converge to a good solution,

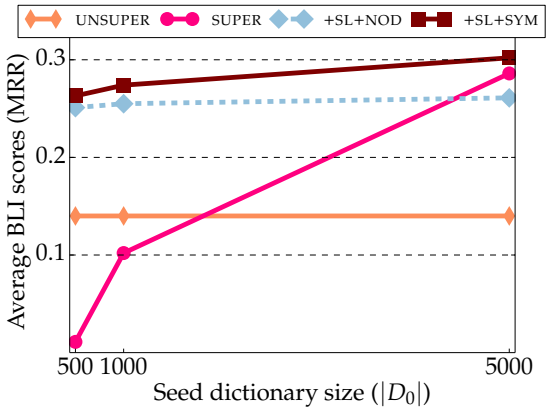


Figure 2: A comparison of average BLI scores with different seed dictionary sizes $D_0$ between a fully unsupervised method (UNSUPER), a supervised method without self-learning (SUPER), and two best performing weakly supervised methods with self learning (+SL+NOD and +SL+SYM). While SUPER without self-learning displays a steep drop in performance with smaller seed dictionaries, there is only a slight decrease when self-learning is used: e.g., 500 translation pairs are still sufficient to initialize robust self-learning.

as also illustrated by the results reported in Table 5.

**How Important are Preprocessing and Post-processing?** The comparisons between ORTHG-SUPER (and ORTHG+SL+SYM) on the one hand, and FULL-SUPER (and FULL+SL+SYM) on the other hand clearly indicate that the component C3 plays a substantial role in effective CLWE learning. FULL-SUPER, which employs all steps S1-S4 (see §2), outperforms ORTHG-SUPER in 208/210 setups with $|D_0|$=*5k* and in 210/210 setups with $|D_0|$=*1k*. Similarly, FULL+SL+SYM is better than ORTHG+SL+SYM in 210/210 setups (both for $|D_0|$=*1k,5k*). The scores also indicate that dropout with self-learning is useful only when we work with noisy unsupervised seed lexicons: FULL+SL+NOD and FULL+SL+SYM without dropout consistently outperform FULL+SL across the board.

**How Important is (Robust) Self-Learning?** We note that the best self-learning method is often useful even when $|D_0| = 5k$ (i.e., FULL+SL+SYM is better than FULL-SUPER in 164/210 setups). However, the importance of robust self-learning gets more pronounced as we decrease the size of $D_0$: FULL+SL+SYM is better than FULL-SUPER in 210/210 setups when $|D_0| = 500$ or $|D_0| = 1,000$. The gap between the two models, as shown in Figure 2, increases dramatically in favor of FULL+SL+SYM as we decrease $|D_0|$.

Again, just comparing FULL-SUPER and UNSUPERVISED in Figure 2 might give a false impression that fully unsupervised CLWE methods can match their supervised counterparts, but the comparison to FULL+SL+SYM reveals the true extent of performance drop when we abandon even weak supervision. The scores also reveal that the choice of self-learning (C2) does matter: all best performing BLI runs with $|D_0| = 1k$ are obtained by two configs with self-learning, and FULL+SL+SYM is the best configuration for 177/210 setups (see Table 4).

**Language Pairs.** As suggested before by Søgaard et al. (2018) and further verified by Glavaš et al. (2019) and Doval et al. (2019), the language pair at hand can have a huge impact on CLWE induction: the adversarial method of Conneau et al. (2018a) often gets stuck in poor local optima and yields degenerate solutions for distant language pairs such as English-Finnish. More recent CLWE methods (Artetxe et al., 2018b; Mohiuddin and Joty, 2019) focus on mitigating this robustness issue. However, they still rely on one critical assumption which leads them to degraded performance for distant language pairs: they assume *approximate isomorphism* (Barone, 2016; Søgaard et al., 2018) between monolingual embedding spaces to learn the initial seed dictionary. In other words, they assume very similar geometric constellations between two monolingual spaces: due to the Zipfian phenomena in language (Zipf, 1949) such near-isomorphism can be satisfied only for *similar languages* and for *similar domains* used for training monolingual vectors. This property is reflected in the results reported in Table 3, the number of unsuccessful setups in Table 4, as well as later in Figure 4.

For instance, the largest number of unsuccessful BLI setups with the UNSUPERVISED model is reported for Korean, Thai (a tonal language), and Basque (a language isolate): their morphological and genealogical properties are furthest away from other languages in our comparison. A substantial number of unsuccessful setups is also observed with other two language outliers from our set (see Table 2 again), Georgian and Indonesian, as well as with morphologically-rich languages such as Estonian or Turkish.

One setting in which fully UNSUPERVISED methods did show impressive results in prior work are similar language pairs. However, even in these settings when the comparison to the weakly supervised FULL-SUPER+SYM is completely fair (i.e.,

| | BG-EU | EU-TR | FI-KO | ID-ET | ID-TH | KA-FI | KA-ID | KO-TR | NO-EU | TR-TH |
|---|---|---|---|---|---|---|---|---|---|---|
| UNSUPERVISED | 0.005 | 0.000 | 0.000 | 0.000 | 0.000 | 0.004 | 0.000 | 0.000 | 0.000 | 0.000 |
| 1K:FULL+SL+SYM | **0.279** | **0.212** | **0.211** | **0.213** | **0.226** | **0.306** | **0.155** | **0.279** | **0.300** | **0.137** |
| 500:FULL+SL+SYM | 0.245 | 0.189 | 0.192 | 0.195 | 0.188 | 0.285 | 0.138 | 0.264 | 0.266 | 0.109 |

Table 5: Results for a selection of BLI setups which were unsuccessful with the UNSUPERVIED CLWE method.

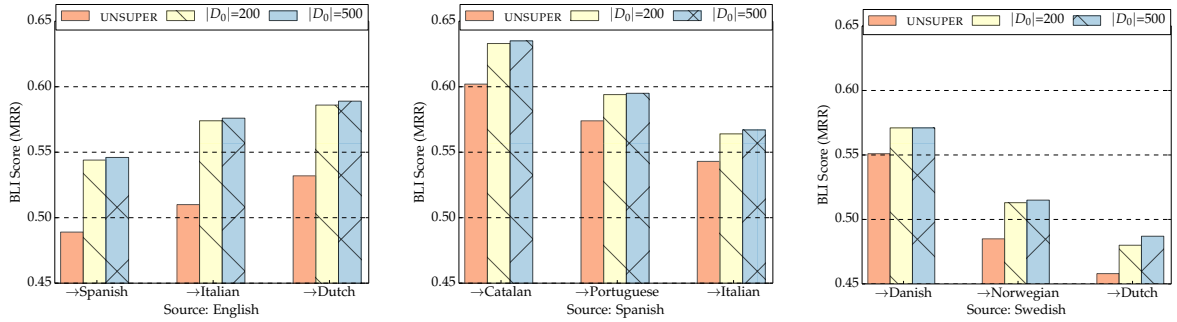


(a) English→ $L_2$ (b) Spanish→ $L_2$ (c) Swedish→ $L_2$

Figure 3: A comparison of BLI scores on "easy" (i.e., similar) language pairs between the fully UNSUPERVISED model and a weakly supervised model (seed dictionary size $|D_0| = 200$ or $|D_0| = 500$) which relies on the self-learning procedure with the symmetry constraint (FULL+SL+SYM).

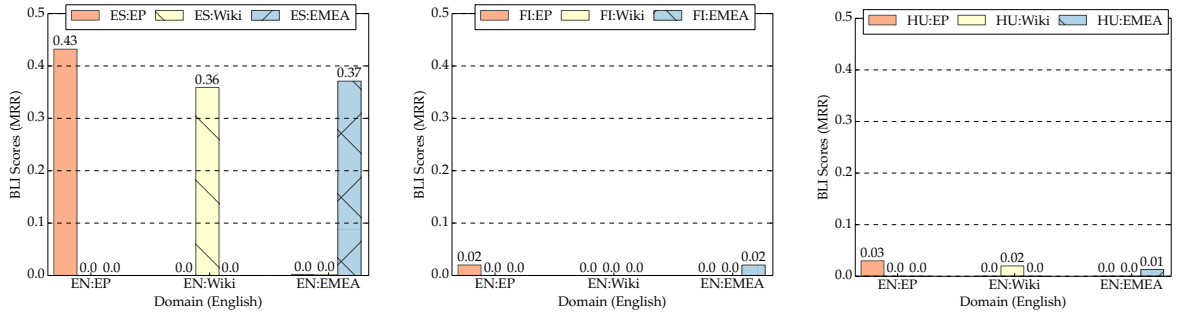


(a) English→ Spanish (b) English → Finnish (c) English → Hungarian

Figure 4: BLI scores with the (most robust) fully UNSUPERVISED model for different language pairs when monolingual word embeddings are trained on dissimilar domains: parliamentary proceedings (EuroParl), Wikipedia (Wiki), and medical corpora (EMEA). Training and test data are the same as in (Søgaard et al., 2018).

the same components C2 and C3 are used for both), UNSUPERVISED still falls short of FULL-SUPER+SYM. These results for three source languages are summarized in Figure 3. What is more, one could argue that we do not need unsupervised CLWEs for similar languages in the first place: we can harvest cheap supervision here, e.g., cognates. The main motivation behind unsupervised approaches is to support dissimilar and resource-poor language pairs for which supervision cannot be guaranteed.

**Domain Differences.** Finally, we also verify that UNSUPERVISED CLWEs still cannot account for domain differences when training monolingual vectors. We rely on the probing test of Søgaard et al. (2018): 300-dim fastText vectors are trained on 1.1M sentences on three corpora: 1) EuroParl.v7 (Koehn, 2005) (parliamentary proceedings); 2) Wikipedia (Al-Rfou et al., 2013), and 3) EMEA (Tiedemann, 2009) (medical), and BLI evaluation for three language pairs is conducted on standard MUSE BLI test sets (Conneau et al., 2018a). The results, summarized in Figure 4, reveal that UNSUPERVISED methods are able to yield a good solution only when there is no domain mismatch and for the pair with two most similar languages (English-Spanish), again questioning their robustness and portability to truly low-resource and more challenging setups. Weakly supervised methods ($|D_0| = 500$ or $D_0$ seeded with identical strings), in contrast, yield good solutions for all setups.

## 5 Further Discussion and Conclusion

The superiority of weakly supervised methods (e.g., FULL+SL+SYM) over unsupervised methods is especially pronounced for distant and typologi-

cally heterogeneous language pairs. However, our study also indicates that even carefully engineered projection-based methods with some seed supervision yield lower absolute performance for such pairs. While we have witnessed the proliferation of fully unsupervised CLWE models recently, some fundamental questions still remain. For instance, the underlying assumption of all projection-based methods (both supervised and unsupervised) is the topological similarity between monolingual spaces, which is why standard simple linear projections result in lower absolute BLI scores for distant pairs (see Table 4 and results in the supplemental material). Unsupervised approaches even exploit the assumption *twice* as their seed extraction is fully based on the topological similarity.

Future work should move beyond the restrictive assumption by exploring new methods that can, e.g., 1) increase the isomorphism between monolingual spaces (Zhang et al., 2019) by distinguishing between language-specific and language-pair-invariant subspaces; 2) learn effective non-linear or multiple local projections between monolingual spaces similar to the preliminary work of Nakashole (2018); 3) similar to Vulić and Korhonen (2016) and Lubin et al. (2019) "denoisify" seed lexicons during the self-learning procedure. For instance, keeping only mutual/symmetric nearest neighbour as in FULL+SL+SYM can be seen as a form of rudimentary denoisifying: it is indicative to see that the best overall performance in this work is reported with that model configuration.

Further, the most important contributions of unsupervised CLWE models are, in fact, the improved and more robust self-learning procedures (component C2) and technical enhancements (component C3). In this work we have demonstrated that these components can be equally applied to weakly supervised approaches: starting from a set of only several hundred pairs, they can guarantee consistently improved performance across the board. As there is still no clear-cut use case scenario for unsupervised CLWEs,[8] instead of "going fully unsupervised", one pragmatic approach to widen the scope of CLWE learning and its application might invest more effort into extracting at least some seed supervision for a variety of language pairs (Artetxe et al., 2017). This finding aligns well with the ongoing initiatives of the PanLex project (Kamholz et al., 2014) and the ASJP database (Wichmann et al., 2018), which aim to collate at least some translation pairs in most of the world's languages.[9]

Finally, this paper demonstrates that, in order to enable fair comparisons, future work on CLWEs should focus on evaluating the CLWE methods' constituent components (e.g, components C1-C3 from this work) instead of full-blown composite systems directly. One goal of the paper is to acknowledge that the work on fully unsupervised CLWE methods has indeed advanced state-of-the-art in cross-lingual word representation learning by offering new solutions also to weakly supervised CLWE methods. However, the robustness problems are still prominent with fully unsupervised CLWEs, and future work should invest more time and effort into developing more robust and more effective methods, e.g., by reaching beyond projection-based methods towards joint approaches (Ruder et al., 2019; Ormazabal et al., 2019).

[8] E.g., unsupervised CLWEs are fully substitutable with the superior weakly supervised CLWEs in unsupervised NMT architectures (Artetxe et al., 2018c; Lample et al., 2018a,b), or in domain adaptation systems (Ziser and Reichart, 2018) and fully unsupervised cross-lingual IR (Litschko et al., 2018).

## Acknowledgments

This work is supported by the ERC Consolidator Grant LEXICAL: Lexical Acquisition Across Languages (no 648909). The work of Goran Glavaš is supported by the Baden-Württemberg Stiftung (AGREE grant of the Eliteprogramm). Roi Reichart is partially funded by ISF personal grants No. 1625/18. We thank the three anonymous reviewers for their encouraging comments and suggestions.

## References

Oliver Adams, Adam Makarucha, Graham Neubig, Steven Bird, and Trevor Cohn. 2017. Cross-lingual word embeddings for low-resource language modeling. In *Proceedings of EACL*, pages 937–947.

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual NLP. In *Proceedings of CoNLL*, pages 183–192.

Jean Alaux, Edouard Grave, Marco Cuturi, and Armand Joulin. 2019. Unsupervised hyperalignment for multilingual word embeddings. In *Proceedings of ICLR*.

David Alvarez-Melis and Tommi Jaakkola. 2018. Gromov-Wasserstein alignment of word embedding spaces. In *Proceedings of EMNLP*, pages 1881–1890.

[9] https://asjp.clld.org/

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of ACL*, pages 451–462.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018a. Generalizing and improving bilingual word embedding mappings with a multi-step framework of linear transformations. In *Proceedings of AAAI*, pages 5012–5019.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018b. A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. In *Proceedings of ACL*, pages 789–798.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018c. Unsupervised neural machine translation. In *Proceedings of ICLR*.

Timothy Baldwin, Jonathan Pool, and Susan Colowick. 2010. PanLex and LEXTRACT: Translating all words of all languages of the world. In *Proceedings of COLING (Demo Papers)*, pages 37–40.

Antonio Valerio Miceli Barone. 2016. Towards cross-lingual distributed representations without parallel text trained with adversarial autoencoders. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 121–126.

Anthony Bell and Terrence Sejnowski. 1997. The 'Independent Components' of Natural Scenes are Edge Filters. *Vision Research*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the ACL*, 5:135–146.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of EMNLP*, pages 740–750.

Xilun Chen and Claire Cardie. 2018. Unsupervised multilingual word embeddings. In *Proceedings of EMNLP*, pages 261–270.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018a. Word translation without parallel data. In *Proceedings of ICLR*.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. 2018b. XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of EMNLP*, pages 2475–2485.

Ryan Cotterell, Sebastian J. Mielke, Jason Eisner, and Brian Roark. 2018. Are all languages equally hard to language-model? In *Proceedings of NAACL-HLT*.

Zi-Yi Dou, Zhi-Hao Zhou, and Shujian Huang. 2018. Unsupervised bilingual lexicon induction via latent variable models. In *Proceedings of EMNLP*, pages 621–626.

Yerai Doval, Jose Camacho-Collados, Luis Espinosa-Anke, and Steven Schockaert. 2019. On the robustness of unsupervised and semi-supervised cross-lingual word embedding learning. *CoRR*, abs/1908.07742.

Long Duong, Hiroshi Kanayama, Tengfei Ma, Steven Bird, and Trevor Cohn. 2016. Learning crosslingual word embeddings without bilingual corpora. In *Proceedings of EMNLP*, pages 1285–1295.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of EACL*, pages 462–471.

Daniela Gerz, Ivan Vulić, Edoardo Maria Ponti, Roi Reichart, and Anna Korhonen. 2018. On the relation between linguistic typology and (limitations of) multilingual language modeling. In *Proceedings of EMNLP*, pages 316–327.

Goran Glavaš, Marc Franco-Salvador, Simone P Ponzetto, and Paolo Rosso. 2018. A resource-light method for cross-lingual semantic textual similarity. *Knowledge-Based Systems*, 143:1–9.

Goran Glavaš, Robert Litschko, Sebastian Ruder, and Ivan Vulić. 2019. How to (properly) evaluate cross-lingual word embeddings: On strong baselines, comparative analyses, and some misconceptions. In *Proceedings of ACL*, pages 710–721.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. BilBOWA: Fast bilingual distributed representations without word alignments. In *Proceedings of ICML*, pages 748–756.

Stephan Gouws and Anders Søgaard. 2015. Simple task-specific bilingual word embeddings. In *Proceedings of NAACL-HLT*, pages 1386–1390.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of LREC*, pages 3483–3487.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *Proceedings of ACL*, pages 1234–1244.

Geert Heyman, Bregt Verreet, Ivan Vulić, and Marie-Francine Moens. 2019. Learning unsupervised multilingual word embeddings with incremental multilingual hubs. In *Proceedings of NAACL-HLT*, pages 1890–1902.

Geert Heyman, Ivan Vulić, and Marie-Francine Moens. 2017. Bilingual lexicon induction by learning to combine word-level and character-level representations. In *Proceedings of EACL*, pages 1085–1095.

Yedid Hoshen and Lior Wolf. 2018. Non-adversarial unsupervised word translation. In *Proceedings of EMNLP*, pages 469–478.

David Kamholz, Jonathan Pool, and Susan M. Colowick. 2014. Panlex: Building a resource for panlingual lexical translation. In *Proceedings of LREC*, pages 3145–3150.

Yova Kementchedjhieva, Sebastian Ruder, Ryan Cotterell, and Anders Søgaard. 2018. Generalizing Procrustes analysis for better bilingual dictionary induction. In *Proceedings of CoNLL*, pages 211–220.

Yunsu Kim, Jiahui Geng, and Hermann Ney. 2018. Improving unsupervised word-by-word translation with language model and denoising autoencoder. In *Proceedings of EMNLP*, pages 862–868.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing crosslingual distributed representations of words. *Proceedings of COLING*, pages 1459–1474.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of the 10th Machine Translation Summit (MT SUMMIT)*, pages 79–86.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018a. Unsupervised machine translation using monolingual corpora only. In *Proceedings of ICLR*.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018b. Phrase-based & neural unsupervised machine translation. In *Proceedings of EMNLP*, pages 5039–5049.

Robert Litschko, Goran Glavaš, Ivan Vulic, and Laura Dietz. 2019. Evaluating resource-lean cross-lingual embedding models in unsupervised retrieval. In *Proceedings of SIGIR*, pages 1109–1112.

Robert Litschko, Goran Glavaš, Simone Paolo Ponzetto, and Ivan Vulić. 2018. Unsupervised cross-lingual information retrieval using monolingual data only. In *Proceedings of SIGIR*, pages 1253–1256.

Noa Yehezkel Lubin, Jacob Goldberger, and Yoav Goldberg. 2019. Aligning vector-spaces with noisy supervised lexicon. In *Proceedings of NAACL-HLT*, pages 460–465.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013a. Exploiting similarities among languages for machine translation. *CoRR, abs/1309.4168*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S Corrado, and Jeffrey Dean. 2013b. Distributed Representations of Words and Phrases and their Compositionality. In *Proceedings of NIPS*, pages 3111–3119.

Tasnim Mohiuddin and Shafiq Joty. 2019. Revisiting adversarial autoencoder for unsupervised word translation with cycle consistency and improved training. In *Proceedings of NAACL-HLT*, pages 3857–3867.

Ndapa Nakashole. 2018. NORMA: Neighborhood sensitive maps for multilingual word embeddings. In *Proceedings of EMNLP*, pages 512–522.

Aitor Ormazabal, Mikel Artetxe, Gorka Labaka, Aitor Soroa, and Eneko Agirre. 2019. Analyzing the limitations of cross-lingual word embedding mappings. In *Proceedings of ACL*, pages 4990–4995.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2019. A survey of cross-lingual word embedding models. *Journal of Artificial Intelligence Research*, 65:569–631.

Samuel L. Smith, David H.P. Turban, Steven Hamblin, and Nils Y. Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In *Proceedings of ICLR*.

Anders Søgaard, Željko Agić, Héctor Martínez Alonso, Barbara Plank, Bernd Bohnet, and Anders Johannsen. 2015. Inverted indexing for cross-lingual NLP. In *Proceedings of ACL*, pages 1713–1722.

Anders Søgaard, Sebastian Ruder, and Ivan Vulić. 2018. On the limitations of unsupervised bilingual dictionary induction. In *Proceedings of ACL*, pages 778–788.

Jörg Tiedemann. 2009. News from OPUS - A collection of multilingual parallel corpora with tools and interfaces. In *Proceedings of RANLP*, pages 237–248.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of ACL*, pages 384–394.

Ivan Vulić and Anna Korhonen. 2016. On the role of seed lexicons in learning bilingual word embeddings. In *Proceedings of ACL*, pages 247–257.

Ivan Vulić and Marie-Francine Moens. 2015. Monolingual and cross-lingual information retrieval models based on (bilingual) word embeddings. In *Proceedings of SIGIR*, pages 363–372.

Ivan Vulić, Nikola Mrkšić, and Anna Korhonen. 2017. Cross-lingual induction and transfer of verb classes based on word vector space specialisation. In *Proceedings of EMNLP*, pages 2536–2548.

Søren Wichmann, André Müller, Viveka Velupillai, Cecil H Brown, Eric W Holman, Pamela Brown, Sebastian Sauppe, Oleg Belyaev, Matthias Urban, Zarina Molochieva, et al. 2018. The ASJP database (version 18).

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of NAACL-HLT*, pages 1006–1011.

Mozhi Zhang, Keyulu Xu, Ken-ichi Kawarabayashi, Stefanie Jegelka, and Jordan Boyd-Graber. 2019. Are girls neko or shōjo? Cross-lingual alignment of non-isomorphic embeddings with iterative normalization. In *Proceedings of ACL*, pages 3180–3189.

Yuan Zhang, David Gaddy, Regina Barzilay, and Tommi Jaakkola. 2016. Ten pairs to tag – Multilingual POS tagging via coarse mapping between embeddings. In *Proceedings of NAACL-HLT*, pages 1307–1317.

George Kingsley Zipf. 1949. *Human behavior and the principle of least effort: An introduction to human ecology*.

Yftah Ziser and Roi Reichart. 2018. Deep pivot-based modeling for cross-language cross-domain transfer with minimal guidance. In *Proceedings of EMNLP*, pages 238–249.

# Weakly-Supervised Concept-based Adversarial Learning for Cross-lingual Word Embeddings

**Haozhou Wang**[1], **James Henderson**[2], **Paola Merlo**[1]
[1] University of Geneva
[2] Idiap Research Institute
{haozhou.wang, paola.merlo}@unige.ch, james.henderson@idiap.ch

## Abstract

Distributed representations of words which map each word to a continuous vector have proven useful in capturing important linguistic information not only in a single language but also across different languages. Current unsupervised adversarial approaches show that it is possible to build a mapping matrix that aligns two sets of monolingual word embeddings without high quality parallel data, such as a dictionary or a sentence-aligned corpus. However, without an additional step of refinement, the preliminary mapping learnt by these methods is unsatisfactory, leading to poor performance for typologically distant languages.

In this paper, we propose a weakly-supervised adversarial training method to overcome this limitation, based on the intuition that mapping across languages is better done at the concept level than at the word level. We propose a concept-based adversarial training method which improves the performance of previous unsupervised adversarial methods for most languages, and especially for typologically distant language pairs.

## 1 Introduction

Distributed representations of words which map each word to a continuous vector have proven useful in capturing important linguistic information. Vectors of words that are semantically or syntactically similar have been shown to be close to each other in the same space (Mikolov et al., 2013a,c; Pennington et al., 2014), making them widely useful in many natural language processing tasks such as machine translation and parsing (Bansal et al., 2014; Mi et al., 2016), both in a single language and across different languages.

Mikolov et al. (2013b) first observed that the geometric positions of similar words in different languages were related by a linear relation. Zou et al. (2013) showed that a cross-lingually shared word embedding space is more useful than a monolingual space in an end-to-end machine translation task. However, traditional methods for mapping two monolingual word embeddings require high quality aligned sentences or dictionaries (Faruqui and Dyer, 2014; Ammar et al., 2016b).

Reducing the need for parallel data, then, has become the main issue for cross-lingual word embedding mapping. Some recent work aiming at reducing resources has shown competitive cross-lingual mappings across similar languages, using a pseudo-dictionary, such as identical character strings between two languages (Smith et al., 2017), or a simple list of numerals (Artetxe et al., 2017).

In a more general method, Zhang et al. (2017) have shown that learning mappings across languages via adversarial training (Goodfellow et al., 2014) can avoid using bilingual evidence. This generality comes at the expense of performance. To overcome this limitation, Lample et al. (2018) refine the preliminary mapping matrix trained by generative adversarial networks (GANs) and obtain a model that is again comparable to supervised models for several language pairs. Despite these big improvements, the performance of these refined GAN models depends largely on the quality of the preliminary mappings. It is probably for this reason that these models still do not show satisfactory performance for typologically distant languages.

In this paper, we explore new augmented models of the adversarial framework that are based on the intuition that the right level of lexical alignment between languages, especially distant languages, is not the word but the concept. We also observe that some cross-lingual resources, such as document-aligned data, are not as expensive as high-quality dictionaries. For example, Wikipedia provides thousands of aligned articles in many lan-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4419–4430,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

guages. Combining these two observations, we develop a concept-based, weakly supervised adversarial method for learning cross-lingual word embedding mappings that rely on these readily available cross-lingual resources.

The main novelty of our method lies in the use of Wikipedia concept-aligned article pairs in the discriminator, which encourages the generator to align words from different languages which are used to describe the same concept. We leverage the power of the adversarial framework and its ability to align distributions by defining concepts as the distribution of words in their respective Wikipedia articles.

The results of our experiments on bilingual lexicon induction show that the preliminary mappings (without post-refinement) trained by our proposed multi-discriminator model are much better than unsupervised adversarial training methods. When we include the post-refinement of Lample et al. (2018), in most cases, our models are comparable or even better than our dictionary-based supervised baselines, and considerably improve the performance of the method of Lample et al. (2018), especially for distant language pairs, such as Chinese-English and Finnish-English.[1] Compared to the previous state-of-the-art, our method still shows competitive performance.

## 2 Models

The basic idea of mapping two sets of pre-trained monolingual word embeddings together was first proposed by Mikolov et al. (2013b). They use a small dictionary of $n$ pairs of words $\{(w_{s_1}, w_{t_1}), ..., (w_{s_n}, w_{t_n})\}$ obtained from Google Translate to learn a transformation matrix $W$ that projects the embeddings $v_{s_i}$ of the source language words $w_{s_i}$ onto the embeddings $v_{t_i}$ of their translation words $w_{t_i}$ in the target language, by optimizing the objective:

$$\min_{W} \sum_{i=1}^{n} \|W v_{s_i} - v_{t_i}\|^2 \tag{1}$$

The trained matrix $W$ can then be used for detecting the translation for any source language word $w_s$ by simply finding the word $w_t$ whose embedding $v_t$ is nearest to $W v_s$. Recent work by Smith et al. (2017) has shown that forcing the matrix $W$ to be orthogonal can effectively improve the results of mapping.

### 2.1 GANs for cross-lingual word embeddings

Since the core of this linear mapping is derived from a dictionary, the quality and size of the dictionary can considerably affect the result. Recent attempts by Zhang et al. (2017) and Lample et al. (2018) have shown that, even without a dictionary or any other cross-lingual resource, training the transformation matrix $W$ is still possible using the GAN framework (Goodfellow et al., 2014). The standard GAN framework plays a min-max game between two models: a generative model $G$ and a discriminative model $D$. The generator $G$ learns from the source data distribution and tries to generate new samples that appear drawn from the distribution of target data. The discriminator $D$ discriminates the generated samples from the target data.

If we adapt the standard GAN framework to the goal of mapping cross-lingual word embeddings, the objective of the generator $G$ is to learn the transformation matrix $W$ that maps the source language embeddings to the target language embedding space, and the discriminator $D_l$, usually a neural network classifier, detects whether the input is from the target language, giving us the objective:

$$\begin{aligned} &\min_{G} \max_{D_l} \; \mathbb{E}_{v_t \sim p_{v_t}} \log D_l(v_t) \\ &+ \mathbb{E}_{v_s \sim p_{v_s}} \log(1 - D_l(G(v_s))) \end{aligned} \tag{2}$$

where $G(v_s) = W v_s$ and $D_l(v_t)$ denotes the probability that $v_t$ came from the distribution $p_{v_t}$ rather than the distribution $p_{v_s}$. The inputs of generator $G$ are the embeddings sampled from the distribution of source language word embeddings, $p_{v_s}$, and the inputs of discriminator $D_l$ are sampled from both the distribution $p_{v_t}$ of real target language word embeddings and the distribution $p_{G(v_s)}$ of generated target language word embeddings. Both $G$ and $D_l$ are trained simultaneously by stochastic gradient descent. A good transformation matrix $W$ can make $p_{G(v_s)}$ similar to $p_{v_t}$, so that $D_l$ can no longer distinguish between them. However, this kind of similarity is at the distribution level. Good monolingual word embeddings are widely spread in the vector space. Consequently, without any post-processing, the cross-lingual mappings are usually inaccurate when we look at specific words (Lample et al., 2018).

[1] We provide our definition of *similar* and *distant* in section 5.

### 2.2 Concept-based GANs for cross-lingual word embeddings

Under the assumption that words in different languages are similar when they are used to describe the same topic or concept (Søgaard et al., 2015), it seems possible to use concept-aligned documents to improve the mapping performance and make the generated embeddings of the source language words closer to the embeddings of corresponding words in the target language. Unlike dictionaries and sentence-aligned corpora, cross-lingual concept-aligned documents are much more readily available. For example, Wikipedia contains more than 40-million articles in more than 200 languages.[2] Therefore, many articles of different languages can be linked together because they are about the same concept.

Given two monolingual word embeddings $V_s^j = \{v_{s_1}, ..., v_{s_j}\}$ and $V_t^k = \{v_{t_1}, , ..., v_{t_k}\}$, our work consists in using a set of Wikipedia concept-aligned article pairs, such that $(C_s^h, C_t^h) = \{(c_{s_1}, c_{t_1})..., (c_{s_h}, c_{t_h})\}$, to reshape the transformation matrix $W$, where $c_{s_i} \in C_s^h$ represents the article of concept $c_i$ in the source language and $c_{t_i} \in C_t^h$ represents the article of the same concept in the target language.

As shown in Figure 1, the generator uses the transformation matrix $W$ to map the input embeddings $v_s$ from the source to the target language. Differently from the original GANs, the input of the generator and the input of the discriminator on concept $D_{cl}$ are not sampled from the whole distribution of $V_s^j$ or $V_t^k$, but from their sub-distribution conditioned on the concept $c$, denoted $p_{v_s|c}$ and $p_{v_t|c}$. $v_c$ represents the embedding of the shared concept $c$. In this paper, we use the embedding of the title of $c$ in the target language. For titles consisting of multiple words, we average the embeddings of its words.

The input of the discriminator on concept $D_{cl}$ is the concatenation of the embedding of the shared concept $v_c$ and the generated target embeddings $G(v_s)$ or the real target embeddings $v_t$. Instead of determining whether its input follows the whole target language distribution $p_{v_t}$, the objective of $D_{cl}$ now becomes to judge whether its input follows the distribution $p_{v_t|c}$, and its loss function is given in equation (3).

$$\log D_{cl}(v_t, v_c) + log(1 - D_{cl}(G(v_s), v_c)) \quad (3)$$

[2] https://en.wikipedia.org/wiki/List_of_Wikipedias

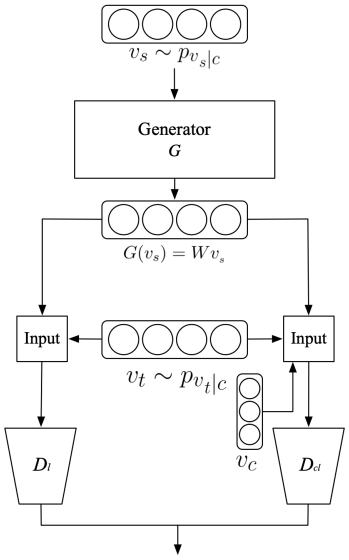


Figure 1: Architecture of our concept-based multi-discriminator model. $D_l$ word-level discriminator, $D_{cl}$ concept-level discriminator.

Because the sub-distribution conditioned on the concept $p_{v_t|c}$ is less spread than $p_{v_t}$ and the proportion of similar words in $p_{v_t|c}$ is higher than in $p_{v_t}$, the embeddings of the input source words have a greater chance of being trained to align with the embeddings of their similar words in the target language.

Our multi-discriminator concept-based model does not completely remove the usual language discriminator $D_l$ that determines whether its input follows the real target language distribution $p_{v_t}$ or the generated target language distribution, $p_{G(v_s)}$. The simpler discriminator $D_l$ is useful when the concept-based discriminator is not stable. The objective function of the multi-discriminator model, shown in equation (4), combines all these elements.

$$\begin{aligned} &\min_G \max_{D_l, D_{cl}} \mathbb{E}_{v_t \sim p_{v_t}} \log D_l(v_t) \\ &+ \mathbb{E}_{v_s \sim p_{v_s}} \log(1 - D_l(G(v_s))) \\ &+ \mathbb{E}_{v_t \sim p_{v_t|c}} \log D_{cl}(v_t, v_c) \\ &+ \mathbb{E}_{v_s \sim p_{v_s|c}} \log(1 - D_{cl}(G(v_s), v_c)) \end{aligned} \quad (4)$$

## 3 Shared Components

Both the GAN and the concept-based GAN models use a variety of further methods that have been shown to improve results.

### 3.1 Orthogonalization

Previous studies (Xing et al., 2015; Smith et al., 2017) show that enforcing the mapping matrix $W$ to be orthogonal can improve the performance and

make the adversarial training more stable. In this work, we perform the same update step proposed by Cisse et al. (2017) to approximate setting $W$ to an orthogonal matrix:

$$W \leftarrow (1+\beta)\,W - \beta\left(WW^{\top}\right)W \qquad (5)$$

According to Lample et al. (2018) and Chen and Cardie (2018), when setting $\beta$ to less than 0.01, the orthogonalization usually performs well.

### 3.2 Post-refinement

Previous work has shown that the core cross-lingual mapping can be improved by refining it by bootstrapping from a dictionary extracted from the learned mapping itself (Lample et al., 2018). Since this refinement process is not the focus of this work, we perform the same refinement procedure as Lample et al. (2018). After the core mapping matrix is learned, we use it to translate the ten thousand most frequent source words $(s_1, s_2, ..., s_{10000})$ into the target language. We then take these ten thousand translations $(t_1, t_2, ..., t_{10000})$ and translate them back into the source language. We call these translation $(s'_1, s'_2, ..., s'_{10000})$. We then consider all the word pairs $(s_i, t_i)$ where $s_i = s'_i$ as our seed dictionary and use this dictionary to update our preliminary mapping matrix using the objective in equation 1.

Moreover, following Xing et al. (2015) and Artetxe et al. (2016), we force the refined matrix $W^*$ to be orthogonal by using singular value decomposition (SVD):

$$W^* = ZU^{\top}, U\Sigma Z^{\top} = SVD({V_s}^{\top}V_t) \qquad (6)$$

### 3.3 Cross-Domain Similarity Local Scaling

Previous work (Radovanović et al., 2010; Dinu et al., 2015) has shown that standard nearest neighbour techniques are not effective to retrieve target similar words in high-dimensional spaces. The work of Lample et al. (2018) showed that using cross-domain similarity local scaling (CSLS) to retrieve target similar words is more accurate than standard nearest neighbour techniques. Instead of just considering the similarity between the source word and its neighbours in the target language, CSLS also takes into account the similarity between the target word and its neighbours in the source language:

$$CSLS(Wv_s, v_t) = \\ 2\cos(Wv_s, v_t) - r_t(Wv_s) - r_s(v_t) \qquad (7)$$

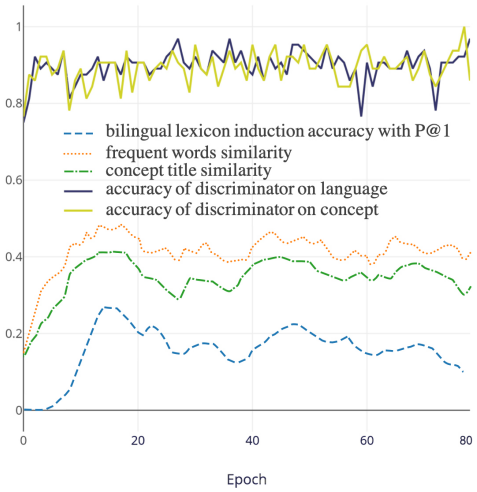


Figure 2: Model selection criteria

where $r_t(Wv_s)$ represents the mean similarity between a source embedding and its neighbours in the target language, and $r_s(v_t)$ represent the mean similarity between a target embedding and its neighbours in the source language. In this work, we use CSLS to build a dictionary for our post-refinement.

## 4 Training

In our weakly-supervised model, the sampling and model selection procedures are important.

**Sampling Procedure** As the statistics show in Table 1, even after filtering, the vocabulary of concept-aligned articles is still large. But it has been shown that the embeddings of frequent words are the most informative and useful (Luong et al., 2013; Lample et al., 2018), so we only keep the one-hundred thousand most frequent words for learning $W$.

For each training step, then, the input word $s$ of our generator is randomly sampled from the vocabulary that is common both to the source monolingual word embedding $S$ and the source Wikipedia concept-aligned articles. After the input source word $s$ is sampled, we sample a concept $c$ according to the frequency of $s$ in each source article of the ensemble of concepts. Then we uniformly sample a target word $t$ from the sub-vocabulary of the target article of concept $c$.[3]

**Model Selection** It is not as difficult to select the best model for weakly-supervised learning as it is

[3]We use uniform sampling for the target words because the distribution of the sampled sub-vocabulary is very small. Sampling $t$ according to its local frequency in the target article would significantly increase the sampling probability of function words, which have less semantic information.

| | Aligned Concepts | Non-English Vocab. | English Vocab. |
|---|---|---|---|
| de ⇄ en | 0.16M | 1.17M | 0.73M |
| es ⇄ en | 0.16M | 0.61M | 0.69M |
| fi ⇄ en | 0.02M | 0.35M | 0.17M |
| fr ⇄ en | 0.16M | 0.63M | 0.71M |
| it ⇄ en | 0.15M | 0.56M | 0.64M |
| ru ⇄ en | 0.10M | 0.89M | 0.56M |
| tr ⇄ en | 0.01M | 0.24M | 0.19M |
| zh ⇄ en | 0.06M | 0.39M | 0.35M |

Table 1: Statistics of Wikipedia concept-aligned articles for our experimentations (M=millions).

for unsupervised learning. In our task, the first selection criterion is the average of the cosine similarity between the embeddings of the concept title in the source and the target languages. Moreover, we find that another good indicator for selecting the best model is the average cosine similarity between the two sets of embeddings consisting of the average embeddings of the ten most frequent words in the source and in the target languages for each aligned article pair. As shown in Figure 2, both of these two criteria correlate well with the word translation evaluation task that we describe in section 5. In this paper, we use the average similarity of the cross-lingual title as the criterion for selecting the best model.

**Other Details** For our generator, the mapping matrix $W$ is initialized with a random orthogonal matrix. Our two discriminators $D_l$ and $D_{cl}$ are two multi-layer perceptron classifiers with different hidden layer sizes (1024 for $D_l$ and 2048 for $D_{cl}$), and a ReLU function as the activation function. In this paper, we set the number of hidden layers to 2 for both $D_l$ and $D_{cl}$. In practice, one hidden layer also performs very well.

## 5 Experiments

We experimentally evaluate our proposal on the task of Bilingual Lexicon Induction (BLI). This task evaluates directly the bilingual mapping ability of a cross-lingual word embedding model. For each language pair, we retrieve the best translations for source words in the test data, and we report the accuracy.

More precisely, for a given source language word, we map its embedding onto the target language and retrieve the closest word. If this closest word is included in the correct translations in the evaluation dictionary, we consider that it is a correct case. In this paper, we report the results that use CSLS to retrieve translations.

**Baselines** As the objective of this work is to improve the performance of unsupervised GANs by using our weakly-supervised concept-based model, we choose the model of Lample et al. (2018) (called MUSE below), as our unsupervised baseline, since it is a typical standard GAN model. We evaluate the models both in a setting without refinement and a setting with refinement. The procedure of refinement is described in section 3.2 and 3.3. Moreover, it is important to evaluate whether our model is comparable to previous supervised models. We use two different dictionary-based systems, VecMap proposed by Artetxe et al. (2017) and Procrustes, provided by Lample et al. (2018), as our supervised baselines. Previous experiments have already shown that these two systems are strong baselines.

### 5.1 Data

Different previous pieces of work on bilingual lexicon induction use different datasets. We choose two from the publicly available ones, selected to have a comprehensive evaluation of our method.

**BLI-1** The dataset provided by Lample et al. (2018) contains high quality dictionaries for more than 150 language pairs. For each language pair, it has a training dictionary of 5000 words and an evaluation dictionary of 1500 words. This dataset allows us to have a better understanding of the performance of our method on many different language pairs. We choose nine languages for testing and compare our method to our supervised and unsupervised baselines described in section 5: English (en), German (de), Finnish (fi), French (fr), Spanish (es), Italian (it), Russian (ru), Turkish (tr) and Chinese (zh). We classify *similar* and *distant* languages based on a combination of structural properties (directional dependency distance, as proposed and measured in Chen and Gerdes (2017)) and lexical properties, as measured by the clustering of current large-scale multilingual sentence embeddings.[4] We consider en ⇄ de, en ⇄ fr, en ⇄ es, en ⇄ it as similar language pairs and en ⇄ fi, en ⇄ ru, en ⇄ tr, en ⇄ zh as distant language pairs.

**BLI-2** Unlike BLI-1, the dataset of Dinu et al. (2015) and its extensions provided by Artetxe et al. (2017, 2018b) only consists of dictionaries of 4 language pairs trained on a Europarl parallel cor-

[4] See for example the clustering in https://code.fb.com/ai-research/laser-multilingual-sentence-embeddings/

| | | *Supervised* | | *Without refinement* | | *With refinement* | |
|---|---|---|---|---|---|---|---|
| | | VecMap | Procrustes | MUSE | Our Method | MUSE | Our Method |
| Similar language pairs | de-en | **72.5** | 72.0 | 55.3 | *66.3* | 72.2 | 72.4 |
| | en-de | 72.5 | 72.1 | 59.2 | *68.2* | 72.5 | **73.4** |
| | es-en | 83.4 | 82.9 | *78.1* | 78.0 | 83.1 | **83.6** |
| | en-es | **81.9** | 81.5 | *75.5* | 74.9 | 81.1 | 80.9 |
| | fr-en | 81.5 | 82.2 | 72.2 | *75.6* | 81.2 | **83.3** |
| | en-fr | 81.3 | 81.3 | 77.8 | *78.3* | **82.2** | **82.2** |
| | it-en | **77.7** | 79.1 | 61.0 | *63.3* | 76.4 | 76.2 |
| | en-it | 77.2 | 77.5 | 63.3 | *64.3* | 77.4 | **78.2** |
| | avg. | 78.5 | 78.6 | 67.8 | *71.1* | 78.3 | **78.8** |
| Distant language pairs | fi-en | **55.9** | 56.2 | 0.00 | *42.8* | 28.6 | 53.5 |
| | en-fi | 39.5 | 39.3 | 0.00 | *35.2* | 21.0 | **43.0** |
| | ru-en | 60.5 | **61.3** | 43.8 | *54.3* | 49.9 | 58.9 |
| | en-ru | 48.0 | 50.9 | 28.7 | *43.2* | 37.6 | **51.1** |
| | tr-en | 55.7 | **58.0** | 12.2 | *42.1* | 25.5 | 54.7 |
| | en-tr | 37.3 | 37.1 | 26.2 | *33.1* | 39.4 | **41.4** |
| | zh-en | **45.0** | 41.2 | 26.9 | *32.1* | 30.8 | 42.0 |
| | en-zh | 45.4 | **52.1** | 29.4 | *37.7* | 33.0 | 48.0 |
| | avg. | 48.4 | **49.5** | 20.9 | *40.1* | 33.2 | 49.1 |

Table 2: Results of bilingual lexicon induction (accuracy % P@1) for similar and distant language pairs on the dataset BLI-1. Procrustes and MUSE represent the supervised and unsupervised model of Lample et al. (2018), VecMap is the supervised model of Artetxe et al. (2017). Word translations are retrieved by using CSLS. Bold face indicates the best result overall and italics indicate the best result between the two columns without refinement.

pus. Each dictionary has a training set of 5000 entries and a test set of 1500 entries. Compared to BLI-1, this dataset is much noisier and the entries are selected from different frequency ranges. However, BLI-2 has been widely used for testing by previous methods (Faruqui and Dyer, 2014; Dinu et al., 2015; Xing et al., 2015; Artetxe et al., 2016; Zhang et al., 2016; Artetxe et al., 2017; Smith et al., 2017; Lample et al., 2018; Artetxe et al., 2018a,b).[5] Using BLI-2 allows us to have a direct comparison with the state-of-the-art.

**Monolingual Word Embeddings** The quality of monolingual word embeddings has a considerable impact on cross-lingual embeddings (Lample et al., 2018). Compared to CBOW and Skip-gram embeddings, FastText embeddings (Bojanowski et al., 2017) capture syntactic information better. The ideal situation would be to use FastText embeddings for both the BLI-1 and BLI-2 datasets. However, much previous work uses CBOW embeddings, so we use different monolingual word embeddings for BLI-1 and for BLI-2.

For BLI-1, we use FastText[6] to train our monolingual word embedding models with 300 dimensions for each language and default settings. The training corpus comes from a Wikipedia dump.[7] For European languages, words are lower-cased and tokenized by the scripts of Moses (Koehn et al., 2007).[8] For Chinese, we first use OpenCC[9] to convert traditional characters to simplified characters and then use Jieba[10] to perform tokenization. For each language, we only keep the words that appear more than five times.

For BLI-2, following the work of Artetxe et al. (2018a),[11] we use their pretrained CBOW embeddings of 300 dimensions. For English, Italian and German, the models are trained on the WacKy corpus. The Finnish model is trained from Common Crawl and the Spanish model is trained from WMT News Crawl.

**Concept Aligned Data** For concept-aligned articles, we use the Linguatools Wikipedia comparable corpus.[12] The statistics of our concept-aligned data are reported in Table 1.

### 5.2 Results and Discussion

Table 2 summarizes the results of bilingual lexicon induction on the BLI-1 dataset. We can see

[5] Most of these methods have been tested by Artetxe et al. (2018b) by using their own implementations.

[6] https://github.com/facebookresearch/fastText

[7] https://dumps.wikimedia.org/

[8] http://www.statmt.org/moses/

[9] https://github.com/BYVoid/OpenCC

[10] https://github.com/fxsjy/jieba

[11] In the work of Artetxe et al. (2018a) the authors reimplemented many previous methods described in table 3.

[12] http://linguatools.org/tools/corpora/wikipedia-comparable-corpora/

that with post-refinement, our method achieves the best performance on eight language pairs. Table 3 illustrates the results of bilingual lexicon induction on the BLI-2 dataset. Although our method (with refinement) does not achieve the best results, the gap between our method and the best state-of-the-art (Artetxe et al., 2018a) is very small, and in most cases, our method is better than previous supervised methods.

**Comparison with unsupervised GANs** As we have mentioned before, the preliminary mappings trained by the method of Lample et al. (2018) perform well for some similar language pairs, such as Spanish to English and French to English. After refinement, their unsupervised GANs models can reach the same level as supervised models for these similar language pairs. However, the biggest drawback of standard GANs is that they exhibit poor performance for distant language pairs. The results from Table 2 clearly confirm this. For example, without refinement, the mapping trained by the unsupervised GAN method can only correctly predict 12% of the words from Turkish to English. Given that the quality of preliminary mappings can seriously affect the effect of refinement, the low-quality preliminary mappings for distant language pairs severely limits the improvements brought by post-refinement. Notice that the method of Lample et al. (2018) scores a null result for English to Finnish on both BLI-1 and BLI-2, indicating that totally unsupervised adversarial training can yields rather unpredictable results.

Compared to the method of Lample et al. (2018), the improvement brought by our method is apparent. Our concept-based GAN models perform better than their unsupervised models on almost every language pair for both datasets, BLI-1 and BLI-2. For those languages where no better results are achieved, the gaps from the best are very small. Figure 4 reports the error reduction rate brought by our method for bilingual lexicon induction on the BLI-1 dataset. It can be clearly seen that the improvement brought by our method is more pronounced on distant language pairs than on similar language pairs.

**Comparison with supervised baselines** Our two dictionary-based supervised baselines are strong. In most cases, the preliminary mappings trained by our concept-based model are not as good, but the gap is small. After post-refinement, our method becomes comparable with these supervised methods. For some language pairs, such as French-English and for English to Finnish, our method performs better.

| Method | *en-de* | en-es | *en-fi* | en-it |
|---|---|---|---|---|
| Supervised | | | | |
| Mikolov et al. (2013b) | 35.0 | 27.3 | 25.9 | 34.9 |
| Faruqui and Dyer (2014) | 37.1 | 26.8 | 27.6 | 38.4 |
| Dinu et al. (2015) | 38.9 | 30.4 | 29.1 | 37.7 |
| Xing et al. (2015) | 41.3 | 31.2 | 28.2 | 36.9 |
| Artetxe et al. (2016) | 41.9 | 31.4 | 30.6 | 39.3 |
| Zhang et al. (2016) | 40.8 | 31.1 | 28.2 | 36.7 |
| Artetxe et al. (2017) | 40.9 | - | 28.7 | 39.7 |
| Smith et al. (2017) | 43.3 | 35.1 | 29.4 | 43.1 |
| Artetxe et al. (2018b) | 44.1 | 32.9 | **44.1** | 45.3 |
| Unsupervised | | | | |
| Zhang et al. (2017) | 0.00 | 0.00 | 0.00 | 0.00 |
| Lample et al. (2018) | 46.8 | 35.4 | 0.38 | 45.2 |
| Artetxe et al. (2018a) | **48.2** | **37.3** | 32.6 | **48.1** |
| Weakly-supervised | | | | |
| Our method | 47.7 | 37.2 | 30.8 | 46.4 |

Table 3: Results of bilingual lexicon induction (accuracy % P@1) on BLI-2 dataset, all the results of previous methods come from (Artetxe et al., 2018a).

**Comparison with the state-of-the-art** From the results shown in Table 3, we can see that in most cases, our method works better than previous supervised and unsupervised approaches. However, the performance of Artetxe et al. (2018a) is very strong and their method always works better than ours. Two potential reasons may cause this difference: First, their self-learning framework iteratively fine-tunes the transformation until convergence, while our refinement just runs for a certain number of iterations.[13] Second, their framework consists of many optimization steps, such as symmetric re-weighting of vectors, steps that we do not have.

**Mapping examples** Table 4 lists some examples of mapping, randomly selected from our experiment results on the dataset of BLI-1. From this table, we can see that for these selected English words, our model performs better than the unsupervised GANs model of Lample et al. (2018), but when we consider the top 3 translation candidates, both models are able to predict the correct translations well. Interestingly, the word "battery" is most commonly translated as Chinese "电池(dianchi)", correctly predicted by our model, and the top 3 translation candidates provided by

[13] As optimisation of the refinement is not the objective of this paper, we follow the work of Lample et al. (2018) and run only five iterations for post-refinement.

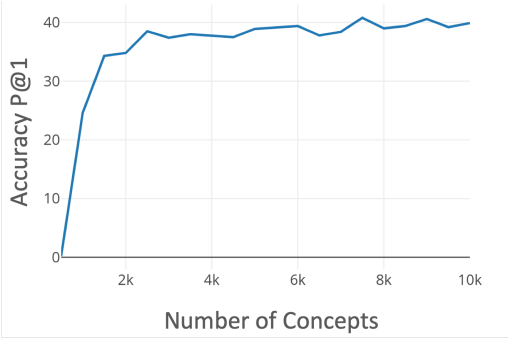


Figure 3: Accuracy of Chinese-English bilingual lexicon induction task for models trained from different concept numbers.

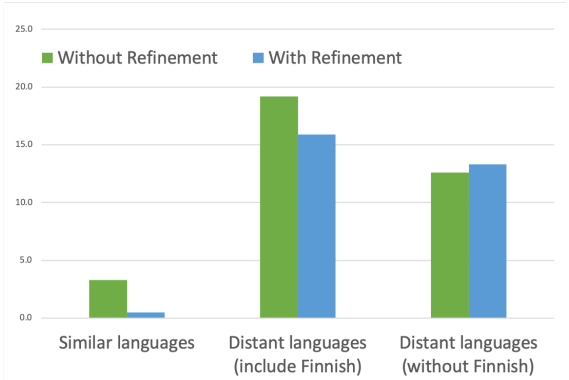


Figure 4: Average error reduction of our method compared to unsupervised adversarial method for bilingual lexicon induction on BLI-1 dataset (Lample et al., 2018). Since the Finnish-English pair is an outlier for the unsupervised method, we report both the average with and without this pair.

our model are all related to "electronic device". However, the translation candidates provided by MUSE are all related to "artillery", a different sense of the the word "battery" both in English and in Chinese, but not as common.

**Impact of number of concepts** To understand whether the number of aligned concepts affects our method, we trained our concept-based models on a range of Chinese-English concept-aligned article pairs, from 550 to 10'000. We test them on the BLI-1 dataset. Following the trend of performance change shown in Figure 3, we see that, when the number of shared concepts reaches 2500, the improvement in accuracy is already very close to the result of the model trained from the total number of concepts (reported in Table 1), thus indicating a direction for future optimization.

## 6 Related Work

Sharing a word embedding space across different languages has proven useful for many cross-lingual tasks, such as machine translation (Zou et al., 2013) and cross-lingual dependency parsing (Jiang et al., 2015, 2016; Ammar et al., 2016a). Generally, such spaces can be trained directly from bilingual sentence aligned or document aligned text (Hermann and Blunsom, 2014; Chandar A P et al., 2014; Søgaard et al., 2015; Vulić and Moens, 2013). However the performance of directly trained models is limited by their vocabulary size.

Initial work on the topic has shown that two monolingual spaces can be combined by applying a linear mapping matrix (Mikolov et al., 2013b). This simple approach has been improved upon in several ways: using canonical correlation analysis to map source and target embeddings (Faruqui and Dyer, 2014); or by forcing the mapping matrix to be orthogonal (Xing et al., 2015).

Recently, efforts have concentrated on how to limit or avoid reliance on dictionaries. Good results were achieved with some drastically minimal techniques. Zhang et al. (2016) achieved good results at bilingual POS tagging, but not bilingual lexicon induction, using only ten word pairs to build a coarse orthonormal mapping between source and target monolingual embeddings. The work of Smith et al. (2017) has shown that a singular value decomposition (SVD) method can produce a competitive cross-lingual mapping by using identical character strings across languages. Artetxe et al. (2017, 2018b) proposed a self-learning framework, which iteratively trains its cross-lingual mapping by using dictionaries trained in previous rounds. The initial dictionary of the self-learning can be reduced to 25 word pairs or even only a list of numerals and still have competitive performance. Furthermore, Artetxe et al. (2018a) extend their self-learning framework to unsupervised models, and build the state-of-the-art for bilingual lexicon induction. Instead of using a pre-build dictionary for initialization, they sort the value of the word vectors in both the source and the target distribution, treat two vectors that have similar permutations as possible translations and use them as the initialization dictionary. Additionally, their unsupervised framework also includes many optimization augmentations, such as stochastic dictionary induction and symmetric re-weighting, among others.

Theoretically, employing GANs for training cross-lingual word embedding is also a promising way to avoid the use of bilingual evidence. As far as we know, Miceli Barone (2016) was the first

| | MUSE | Our Method |
|---|---|---|
| | *English to Chinese* | |
| film | **电影(film)**, 该片(this film), 本片(this film) | **电影(film)**, **影片(film)**, 本片(this film) |
| debate | **辩论(debate)**, 争论(dispute), 讨论(discuss) | **辩论(debate)**, 争论(dispute), 议题(topic) |
| battery | 火炮(artillery), 炮(cannon), 装甲车辆(armored car) | **电池(battery)**, 锂电池(lithium battery), 控制器(controller) |
| | *English to French* | |
| january | novembre(november), février(february), **janvier(january)** | **janvier(january)**, février(february), septembre(september) |
| author | **auteur(author)**, écrivain(writer), romancier(novelist) | **auteur(author)**, écrivain(writer), romancier(novelist) |
| polytechnic | université(university), **polytechnique(polytechnic)**, collège(high school) | université(university), **polytechnique(polytechnic)**, faculté(faculty) |

Table 4: Top 3 English to Chinese and English to French translation candidates selected form our bilingual lexicon induction experiment on the dataset of BLI-1. Word translations are retrieved by using CSLS. Both MUSE and our model are after refinement. Bold face indicates the common translation of selected English words.

attempt at this approach, but the performance of their model is not competitive. Zhang et al. (2017) enforce the mapping matrix to be orthogonal during the adversarial training and achieve good performance on bilingual lexicon induction. The main drawback of their approach is that the vocabularies of their training data are small, and the performance of their models drops significantly when they use large training data. The recent model proposed by Lample et al. (2018) is so far the most successful, becoming competitive with previous supervised approaches through a strong CSLS-based refinement to the core mapping matrix trained by GANs. Even in this case, though, without refinement, the core mappings are not as good as hoped for some distant language pairs. More recently, Chen and Cardie (2018) extends the work of Lample et al. (2018) from the bilingual setting to the multi-lingual setting. Instead of training crosslingual word embeddings for only one language pair, their approach allows them to train crosslingual word embeddings for many language pairs at the same time. Another recent piece of work which is similar to Lample et al. (2018) comes from Xu et al. (2018). Their approach can be divided into 2 steps: first, using Wasserstein GAN (Arjovsky et al., 2017) to train a preliminary mapping between two monolingual distributions and then minimizing the Sinkhorn Distance across distributions. Although their method performs better than Lample et al. (2018) in several tasks, the improvement mainly comes from the second step, showing that the problem of how to train a better preliminary mapping has not been resolved.

## 7 Conclusions and Future Work

In this paper, we propose a weakly-supervised adversarial training method for cross-lingual word embedding mapping. Our approach is based on the intuition that mapping across distant languages is better done at the concept level than at the word level. We propose using the distributions of words in aligned Wikipedia articles as data for this concept mapping. The method improves the performance over previous unsupervised adversarial methods in almost all cases, especially for distant languages.

The monolingual word embeddings that we have used in this paper represent a word and not its different senses. These models do not handle polysemous words well. This also can have a negative impact on the performance of cross-lingual mapping. Recent work on contextualized word embeddings (Peters et al., 2018; Devlin et al., 2019) provides dynamic representations for words and has been shown to perform better at handling the problem of polysemy. Extending our concept-based cross-lingual mapping to contextualized word representations will be the focus of our future work.

## References


Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2016a. Many Languages, One Parser. *Transactions of the Association for Computational Linguistics*, 4:431–444.

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A. Smith. 2016b. Massively Multilingual Word Embeddings. *arXiv preprint arXiv:1309.4168*.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein GAN. *arXiv preprint arXiv:1309.4168*.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance . In *2016 Conference on Empirical Methods in Natural Language Processing Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2289–2294, Austin, Texas, USA. Association for Computational Linguistics.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *55th Annual Meeting of the Association for Computational Linguistics*, pages 451–462, Vancouver, Canada.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018a. A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. In *56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 789–798, Melbourne, Australia. Association for Computational Linguistics.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018b. Generalizing and improving bilingual word embedding mappings with a multi-step framework of linear transformations. In *Thirty-Second Conference on Artificial Intelligence*, pages 5012–5019.

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring Continuous Word Representations for Dependency Parsing. In *52nd Annual Meeting of the Association for Computational Linguistics*, pages 809–815, Baltimore, Maryland, USA.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Sarath Chandar A P, Stanislas Lauly, Hugo Larochelle, Mitesh Khapra, Balaraman Ravindran, Vikas C Raykar, and Amrita Saha. 2014. An Autoencoder Approach to Learning Bilingual Word Representations. In *2014 Advances in Neural Information Processing Systems*, pages 1853–1861, Montréal Canada.

Xilun Chen and Claire Cardie. 2018. Unsupervised Multilingual Word Embeddings. In *2018 Conference on Empirical Methods in Natural Language Processing*, pages 261–270. Association for Computational Linguistics.

Xinying Chen and Kim Gerdes. 2017. Classifying languages by dependency structure. typologies of delexicalized universal dependency treebanks. In *Proceedings of the Fourth International Conference on Dependency Linguistics (Depling 2017)*, pages 54–63, Pisa, Italy. Linköping University Electronic Press.

Moustapha Cisse, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. 2017. Parseval Networks: Improving Robustness to Adversarial Examples. In *34th International Conference on Machine Learning*, pages 854–863, Sydney, Australia. PMLR.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Georgiana Dinu, Angeliki Lazaridou, and Marco Baroni. 2015. Improving zero-shot learning by mitigating the hubness problem. In *3rd International Conference on Learning Representations*, pages 1–10, Toulon, France.

Manaal Faruqui and Chris Dyer. 2014. Improving Vector Space Word Representations Using Multilingual Correlation. In *14th Coference of the European Chapter of the Association for Computational Linguistics*, pages 462–471, Gothenburg, Sweden.

Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Networks. In *Advances in Neural Information Processing Systems 27*, pages 2672–2680, Montréal, Canada.

Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual Models for Compositional Distributed Semantics. In *52nd Annual Meeting of the Association for Computational Linguistics*, pages 58–68, Baltimore, Maryland, USA.

Guo Jiang, Che Wanxiang, Yarowsky David, Wang Haifeng, and Liu Ting. 2015. Cross-lingual dependency parsing based on distributed representations. In *53rd Annual Meeting of the Association for Computational Linguistics and 7th International Joint Conference on Natural Language Processing*, pages 1234–1244, Beijing, China.

Guo Jiang, Che Wanxiang, Yarowsky David, Wang Haifeng, and Liu Ting. 2016. A distributed representation-based framework for cross-lingual transfer parsing. *Journal of Artificial Intelligence Research*, 55:995–1023.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *45th Annual Meeting of the Association for Computational Linguistics*, Prague, Czech Republic.

Guillaume Lample, Alexis Conneau, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. In *6th International Conference on Learning Representations*, pages 1–14, Vancouver, Canada.

Minh-Thang Luong, Richard Socher, and Christopher D. Manning. 2013. Better word representations with recursive neural networks for morphology. In *17th Conference on Computational Natural Language Learning*, pages 104–113, Sofia, Bulgaria.

Haitao Mi, Baskaran Sankaran, Zhiguo Wang, and Abe Ittycheriah. 2016. Coverage Embedding Models for Neural Machine Translation. In *2016 Conference on Empirical Methods in Natural Language Processing*, pages 955–960, Austin, Texas, USA.

Antonio Valerio Miceli Barone. 2016. Towards cross-lingual distributed representations without parallel text trained with adversarial autoencoders. In *1st Workshop on Representation Learning for NLP*, pages 121–126, Berlin, Germany.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1309.4168*.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013b. Exploiting Similarities among Languages for Machine Translation . *arXiv preprint arXiv:1309.4168*, (Computation and Language):1–10.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013c. Distributed Representations of Words and Phrases and their Compositionality. In *26th International Conference on Neural Information Processing Systems*, pages 3111–3119, Lake Tahoe, Nevada, USA.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove - Global Vectors for Word Representation. In *2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, Doha, Qatar.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Miloš Radovanović, Alexandros Nanopoulos, and Mirjana Ivanović. 2010. Hubs in Space: Popular Nearest Neighbors in High-Dimensional Data. *Journal of machine learning research*, 11(Sep):2487–2531.

Samuel L. Smith, David H. P. Turban, Steven Hamblin, and Nils Y. Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In *15th International Conference on Learning Representations*, pages 1–10, Toulon, France.

Anders Søgaard, Zeljko Agić, Héctor Martínez Alonso, Barbara Plank, Bernd Bohnet, and Anders Johannsen. 2015. Inverted indexing for cross-lingual NLP. In *53rd Annual Meeting of the Association for Computational Linguistics and 7th International Joint Conference on Natural Language Processing*, pages 1713–1722, Beijing, China.

Ivan Vulić and Marie-Francine Moens. 2013. A Study on Bootstrapping Bilingual Vector Spaces from Non-Parallel Data (and Nothing Else). In *2013 Conference on Empirical Methods in Natural Language Processing*, pages 1613–1624, Seattle, Washington, USA.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized Word Embedding and Orthogonal Transform for Bilingual Word Translation. In *2015 Annual Conference of the North American Chapter of the ACL*, pages 1006–1011, Denver, Colorado, USA. Association for Computational Linguistics.

Ruochen Xu, Yiming Yang, Naoki Otani, and Yuexin Wu. 2018. Unsupervised Cross-lingual Transfer of Word Embedding Spaces. In *2018 Conference on Empirical Methods in Natural Language Processing*, pages 2465–2474. Association for Computational Linguistics.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Adversarial Training for Unsupervised Bilingual Lexicon Induction. In *55th Annual Meeting of the Association for Computational Linguistics*, pages 1959–1970, Vancouver, Canada. Association for Computational Linguistics.

Yuan Zhang, David Gaddy, Regina Barzilay, and Tommi Jaakkola. 2016. Ten Pairs to Tag - Multilingual POS Tagging via Coarse Mapping between Embeddings. In *15th Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 12–17, San Diego, California, USA. Association for Computational Linguistics.

Will Y Zou, Richard Socher, Daniel Cer, and Christopher D. Manning. 2013. Bilingual Word Embeddings for Phrase-Based Machine Translation. In *Conference on Empirical Methods in Natural Language Processing*, pages 1393–1398, Seattle, Washington, USA.

# Aligning Cross-Lingual Entities with Multi-Aspect Information

**Hsiu-Wei Yang**[1]*, **Yanyan Zou**[2]*, **Peng Shi**[1]*, **Wei Lu**[2], **Jimmy Lin**[1], and **Xu Sun**[3,4]

[1] David R. Cheriton School of Computer Science, University of Waterloo
[2] StatNLP Research Group, Singapore University of Technology and Design
[3] MOE Key Lab of Computational Linguistics, School of EECS, Peking University
[4] Center for Data Science, Beijing Institute of Big Data Research, Peking University

{h324yang,p8shi,jimmylin}@uwaterloo.ca
yanyan_zou@mymail.sutd.edu.sg
luwei@sutd.edu.sg, xusun@pku.edu.cn

## Abstract

Multilingual knowledge graphs (KGs), such as YAGO and DBpedia, represent entities in different languages. The task of cross-lingual entity alignment is to match entities in a source language with their counterparts in target languages. In this work, we investigate embedding-based approaches to encode entities from multilingual KGs into the same vector space, where equivalent entities are close to each other. Specifically, we apply graph convolutional networks (GCNs) to combine multi-aspect information of entities, including topological connections, relations, and attributes of entities, to learn entity embeddings. To exploit the literal descriptions of entities expressed in different languages, we propose two uses of a pretrained multilingual BERT model to bridge cross-lingual gaps. We further propose two strategies to integrate GCN-based and BERT-based modules to boost performance. Extensive experiments on two benchmark datasets demonstrate that our method significantly outperforms existing systems.

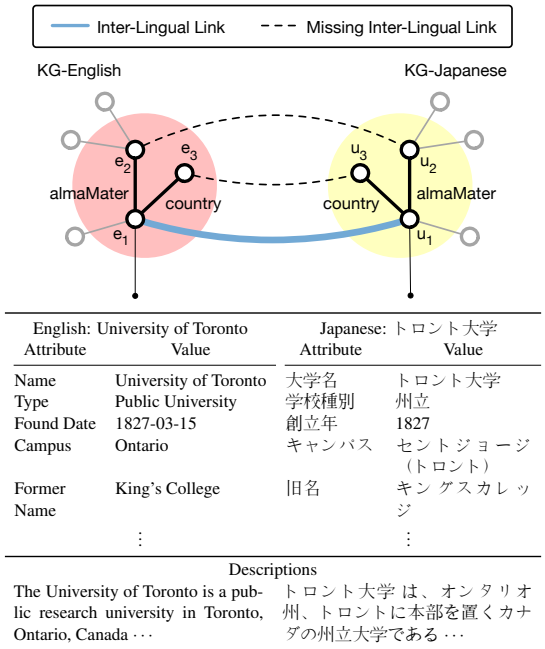


Figure 1: An example fragment of two KGs (in English and Japanese) connected by an inter-lingual link (ILL). In addition to the graph structures (top) consisting of entity nodes and typed relation edges, KGs also provide attributes and literal descriptions of entities (bottom).

## 1 Introduction

A growing number of multilingual knowledge graphs (KGs) have been built, such as DBpedia (Bizer et al., 2009), YAGO (Suchanek et al., 2008; Rebele et al., 2016), and BabelNet (Navigli and Ponzetto, 2012), which typically represent real-world knowledge as separately-structured monolingual KGs. Such KGs are connected via inter-lingual links (ILLs) that align entities with their counterparts in different languages, exemplified by Figure 1 (top). Highly-integrated multilingual KGs contain useful knowledge that can benefit many knowledge-driven cross-lingual NLP tasks, such as machine translation (Moussallem et al., 2018) and cross-lingual named entity recognition (Darwish, 2013). However, the coverage of ILLs among existing KGs is quite low (Chen et al., 2018): for example, less than 20% of the entities in DBpedia are covered by ILLs. The goal of cross-lingual entity alignment is to discover entities from different monolingual KGs that actually refer to the same real-world entities, i.e., discovering the missing ILLs.

Traditional methods for this task apply machine translation techniques to translate entity labels (Spohr et al., 2011). The quality of alignments in the cross-lingual scenario heavily depends on the quality of the adopted translation systems. In addition to entity labels, existing KGs also provide multi-aspect information of entities,

* Equal contribution.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4431–4441, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

including topological connections, relation types, attributes, and literal descriptions expressed in different languages (Bizer et al., 2009; Xie et al., 2016), as shown in Figure 1 (bottom). The key challenge of addressing such a task thus is how to better model and use provided multi-aspect information of entities to bridge cross-lingual gaps and find more equivalent entities (i.e., ILLs).

Recently, embedding-based solutions (Chen et al., 2017b; Sun et al., 2017; Zhu et al., 2017; Wang et al., 2018; Chen et al., 2018) have been proposed to unify multilingual KGs into the same low-dimensional vector space where equivalent entities are close to each other. Such methods only make use of one or two aspects of the aforementioned information. For example, Zhu et al. (2017) relied only on topological features while Sun et al. (2017) and Wang et al. (2018) exploited both topological and attribute features. Chen et al. (2018) proposed a co-training algorithm to combine topological features and literal descriptions of entities. However, combining these multi-aspect information of entities (i.e., topological connections, relations and attributes, as well as literal descriptions) remains under-explored.

In this work, we propose a novel approach to learn cross-lingual entity embeddings by using all aforementioned aspects of information in KGs. To be specific, we propose two variants of GCN-based models, namely MAN and HMAN, that incorporate multi-aspect features, including topological features, relation types, and attributes into cross-lingual entity embeddings. To capture semantic relatedness of literal descriptions, we fine-tune the pretrained multilingual BERT model (Devlin et al., 2019) to bridge cross-lingual gaps. We design two strategies to combine GCN-based and BERT-based modules to make alignment decisions. Experiments show that our method achieves new state-of-the-art results on two benchmark datasets. Source code for our models is publicly available at https://github.com/h324yang/HMAN.

## 2 Problem Definition

In a multilingual knowledge graph $\mathcal{G}$, we use $\mathcal{L}$ to denote the set of languages that $\mathcal{G}$ contains and $\mathcal{G}_i = \{E_i, R_i, A_i, V_i, D_i\}$ to represent the language-specific knowledge graph in language $L_i \in \mathcal{L}$. $E_i$, $R_i$, $A_i$, $V_i$ and $D_i$ are sets of entities, relations, attributes, values of attributes, and literal descriptions, each of which portrays one aspect of an entity. The graph $\mathcal{G}_i$ consists of relation triples $\langle h_i, r_i, t_i\rangle$ and attribute triples $\langle h_i, a_i, v_i\rangle$ such that $h_i, t_i \in E_i$, $r_i \in R_i$, $a_i \in A_i$ and $v_i \in V_i$. Each entity is accompanied by a literal description consisting of a sequence of words in language $L_i$, e.g., $\langle h_i, d_{h,i}\rangle$ and $\langle t_i, d_{t,i}\rangle$, $d_{h,i}, d_{t,i} \in D_i$.

Given two knowledge graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ expressed in source language $L_1$ and target language $L_2$, respectively, there exists a set of pre-aligned ILLs $I(\mathcal{G}_1, \mathcal{G}_2) = \{(e, u) | e \in E_1, u \in E_2\}$ which can be considered training data. The task of cross-lingual entity alignment is to align entities in $\mathcal{G}_1$ with their cross-lingual counterparts in $\mathcal{G}_2$, i.e., discover missing ILLs.

## 3 Proposed Approach

In this section, we first introduce two GCN-based models, namely MAN and HMAN, that learn entity embeddings from the graph structures. Second, we discuss two uses of a multilingual pretrained BERT model to learn cross-lingual embeddings of entity descriptions: POINTWISEBERT and PAIRWISEBERT. Finally, we investigate two strategies to integrate the GCN-based and the BERT-based modules.

### 3.1 Cross-Lingual Graph Embeddings

Graph convolutional networks (GCNs) (Kipf and Welling, 2017) are variants of convolutional networks that have proven effective in capturing information from graph structures, such as dependency graphs (Guo et al., 2019b), abstract meaning representation graphs (Guo et al., 2019a), and knowledge graphs (Wang et al., 2018). In practice, multi-layer GCNs are stacked to collect evidence from multi-hop neighbors. Formally, the $l$-th GCN layer takes as input feature representations $H^{(l-1)}$ and outputs $H^{(l)}$:

$$H^{(l)} = \phi\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l-1)}W^{(l)}\right) \quad (1)$$

where $\tilde{A} = A + I$ is the adjacency matrix, $I$ is the identity matrix, $\tilde{D}$ is the diagonal node degree matrix of $\tilde{A}$, $\phi(\cdot)$ is ReLU function, and $W^{(l)}$ represents learnable parameters in the $l$-th layer. $H^{(0)}$ is the initial input.

GCNs can iteratively update the representation of each entity node via a propagation mechanism through the graph. Inspired by previous studies (Zhang et al., 2018; Wang et al., 2018), we also

adopt GCNs in this work to collect evidence from multilingual KG structures and to learn cross-lingual embeddings of entities. The primary assumptions are: (1) equivalent entities tend to be neighbored by equivalent entities via the same types of relations; (2) equivalent entities tend to share similar or even the same attributes.

**Multi-Aspect Entity Features.** Existing KGs (Bizer et al., 2009; Suchanek et al., 2008; Rebele et al., 2016) provide multi-aspect information of entities. In this section, we mainly focus on the following three aspects: topological connections, relations, and attributes. The key challenge is how to utilize the provided features to learn better embeddings of entities. We discuss how we construct raw features for the three aspects, which are then fed as inputs to our model. We use $X_t$, $X_r$ and $X_a$ to denote the topological connection, relation, and attribute features, individually.

The topological features contain rich neighborhood proximity information of entities, which can be captured by multi-layer GCNs. As in Wang et al. (2018), we set the initial topological features to $X_t = I$, i.e., an identity matrix serving as index vectors (also known as the featureless setting), so that the GCN can learn the representations of corresponding entities.

In addition, we also consider the relation and attribute features. As shown in Figure 1, the connected relations and attributes of two equivalent entities, e.g., "*University of Toronto*" (English) and "トロント大学" (Japanese), have a lot of overlap, which can benefit cross-lingual entity alignment. Specifically, they share the same relation types, e.g., "country" and "almaMater", and some attributes, e.g., "foundDate" and "創立年". To capture relation information, Schlichtkrull et al. (2018) proposed RGCN with relation-wise parameters. However, with respect to this task, existing KGs typically contain thousands of relation types but few pre-aligned ILLs. Directly adopting RGCN may introduce too many parameters for the limited training data and thus cause overfitting. Wang et al. (2018) instead simply used the unlabeled GCNs (Kipf and Welling, 2017) with two proposed measures (i.e., functionality and inverse functionality) to encode the information of relations into the adjacency matrix. They also considered attributes as input features in their architecture. However, this approach may lose information about relation types. Therefore, we regard relations and attributes of entities as bag-of-words features to explicitly model these two aspects. Specifically, we construct count-based *N-hot* vectors $X_r$ and $X_a$ for these two aspects of features, respectively, where the $(i, j)$ entry is the count of the $j$-th relation (attribute) for the corresponding entity $e_i$. Note that we only consider the top-$F$ most frequent relations and attributes to avoid data sparsity issues. Thus, for each entity, both of its relation and attribute features are $F$-dimensional vectors.

**MAN.** Inspired by Wang et al. (2018), we propose the *Multi-Aspect Alignment Network* (MAN) to capture the three aspects of entity features. Specifically, three $l$-layer GCNs take as inputs the triple-aspect features (i.e., $X_t$, $X_r$, and $X_a$) and produce the representations $H_t^{(l)}$, $H_r^{(l)}$, and $H_a^{(l)}$ according to Equation 1, respectively. Finally, the multi-aspect entity embedding is:

$$H_m = [H_t^{(l)} \oplus H_a^{(l)} \oplus H_r^{(l)}] \quad (2)$$

where $\oplus$ denotes vector concatenation. $H_m$ can then feed into alignment decisions.

Such fusion through concatenation is also known as *Scoring Level Fusion*, which has been proven simple but effective for capturing multi-modal semantics (Bruni et al., 2014; Kiela and Bottou, 2014; Collell et al., 2017). It is worth noting that the main differences between MAN and the work of Wang et al. (2018) are two fold: First, we use the same approach as in Kipf and Welling (2017) to construct the adjacency matrix, while Wang et al. (2018) designed a new connectivity matrix as the adjacency matrix for the GCNs. Second, MAN explicitly regards the relation type features as model input, while Wang et al. (2018) incorporated such relation information into the connectivity matrix.

**HMAN.** Note that MAN propagates relation and attribute information through the graph structure. However, for aligning a pair of entities, we observe that considering the relations and attributes of neighboring entities, besides their own ones, may introduce noise. Merely focusing on relation and attribute features of the current entity could be a better choice. Thus, we propose the *Hybrid Multi-Aspect Alignment Network* (HMAN) to better model such diverse features, shown in Figure 2. Similar to MAN, we still leverage the $l$-th layer of a GCN to obtain topological embeddings $H_t^{(l)}$, but exploit feedforward neural networks to obtain

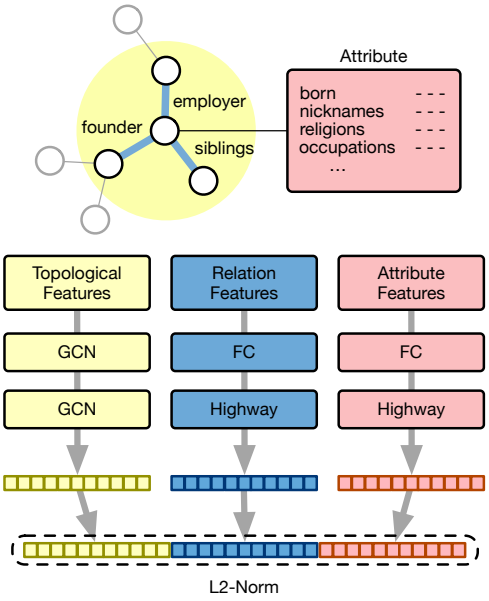


Figure 2: Architecture of HMAN.

the embeddings with respect to relations and attributes. The feedforward neural networks consist of one fully-connected (FC) layer and a highway network layer (Srivastava et al., 2015). The reason we use highway networks is consistent with the conclusions of Mudgal et al. (2018), who conducted a design space exploration of neural models for entity matching and found that highway networks are generally better than FC layers in convergence speed and effectiveness.

Formally, these feedforward neural networks are defined as:

$$\begin{aligned} S_f &= \phi(W_f^{(1)} X_f + b_f^{(1)}) \\ T_f &= \sigma(W_f^t S_f + b_f^t) \\ G_f &= \phi(W_f^{(2)} S_f + b_f^{(2)}) \cdot T_f + S_f \cdot (1 - T_f) \end{aligned} \tag{3}$$

where $f \in \{r, a\}$ and $X_f$ refer to one specific aspect (i.e., relation or attribute) and the corresponding raw features, respectively, $W_f^{(1,2,t)}$ and $b_f^{(1,2,t)}$ are model parameters, $\phi(\cdot)$ is ReLU function, and $\sigma(\cdot)$ is sigmoid function. Accordingly, we obtain the hybrid multi-aspect entity embedding $H_y = [H_t^{(l)} \oplus G_r \oplus G_a]$, to which $\ell_2$ normalization is further applied.

**Model Objective.** Given two knowledge graphs, $\mathcal{G}_1$ and $\mathcal{G}_2$, and a set of pre-aligned entity pairs $I\,(\mathcal{G}_1, \mathcal{G}_2)$ as training data, our model is trained in a supervised fashion. During the training phase, the goal is to embed cross-lingual entities into the same low-dimensional vector space where equivalent entities are close to each other. Following Wang et al. (2018), our margin-based ranking loss function is defined as:

$$\begin{aligned} J = \sum_{(e_1,e_2)\in I} \sum_{(e_1',e_2')\in I'} & [\rho(h_{e_1}, h_{e_2}) + \beta \\ & - \rho(h_{e_1'}, h_{e_2'})]_+ \end{aligned} \tag{4}$$

where $[x]_+ = \max\{0, x\}$, $I'$ denotes the set of negative entity alignment pairs constructed by corrupting the gold pair $(e_1, e_2) \in I$. Specifically, we replace $e_1$ or $e_2$ with a randomly-chosen entity in $E_1$ or $E_2$. $\rho(x, y)$ is the $\ell_1$ distance function, and $\beta > 0$ is the margin hyperparameter separating positive and negative pairs.

### 3.2 Cross-Lingual Textual Embeddings

Existing multilingual KGs (Bizer et al., 2009; Navigli and Ponzetto, 2012; Rebele et al., 2016) also provide literal descriptions of entities expressed in different languages and contain detailed semantic information about the entities. The key observation is that literal descriptions of equivalent entities are semantically close to each other. However, it is non-trivial to directly measure the semantic relatedness of two entities' descriptions, since they are expressed in different languages.

Recently, Bidirectional Encoder Representations from Transformer (BERT) (Devlin et al., 2019) has advanced the state-of-the-art in various NLP tasks by heavily exploiting pretraining based on language modeling. Of special interest is the multilingual variant, which was trained with Wikipedia dumps of 104 languages. The spirit of BERT in the multilingual scenario is to project words or sentences from different languages into the same semantic space. This aligns well with our objective—bridging gaps between descriptions written in different languages. Therefore, we propose two methods for applying multilingual BERT, POINTWISEBERT and PAIRWISEBERT, to help make alignment decisions.

**POINTWISEBERT.** A simple choice is to follow the basic design of BERT and formulate the entity alignment task as a text matching task. For two entities $e_1$ and $e_2$ from two KGs in $L_1$ and $L_2$, denoting source language and target language, respectively, their textual descriptions are $d_1$ and $d_2$, consisting of word sequences in two languages. The model takes as inputs [CLS] $d_1$ [SEP] $d_2$ [SEP], where [CLS] is the special classification token,

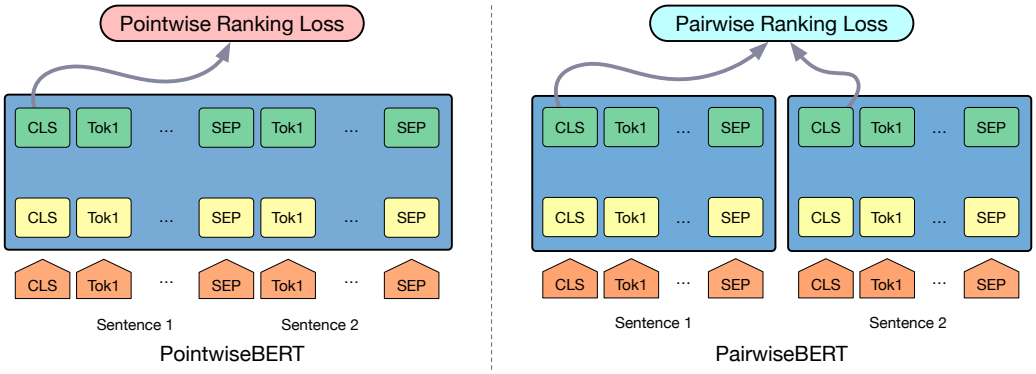


Figure 3: Architecture overview of POINTWISEBERT (left) and PAIRWISEBERT (right).

from which the final hidden state is used as the sequence representation, and [SEP] is the special token for separating token sequences, and produces the probability of classifying the pair as equivalent entities. The probability is then used to rank all candidate entity pairs, i.e., ranking score. We denote this model as POINTWISEBERT, shown in Figure 3 (left).

This approach is computationally expensive, since for each entity we need to consider all candidate entities in the target language. One solution, inspired by the work of Shi et al. (2019), is to reduce the search space for each entity with a *reranking strategy* (see Section 3.3).

**PAIRWISEBERT.** Due to the heavy computational cost of POINTWISEBERT, semantic matching between all entity pairs is very expensive. Instead of producing ranking scores for description pairs, we propose PAIRWISEBERT to encode the entity literal descriptions as cross-lingual textual embeddings, where distances between entity pairs can be directly measured using these embeddings.

The PAIRWISEBERT model consists of two components, each of which takes as input the description of one entity (from the source or target language), as depicted in Figure 3 (right). Specifically, the input is designed as [CLS] $d_1(d_2)$ [SEP], which is then fed into PAIRWISEBERT for contextual encoding. We select the hidden state of [CLS] as the textual embedding of the entity description for training and inference. To bring the textual embeddings of cross-lingual entity descriptions into the same vector space, a similar ranking loss function as in Equation 4 is used.

### 3.3 Integration Strategy

Sections 3.1 and 3.2 introduce two modules that separately collect evidence from knowledge graph structures and the literal descriptions of entities, namely graph and textual embeddings. In this section, we investigate two strategies to integrate these two modules to further boost performance.

**Reranking.** As mentioned in Section 3.2, the POINTWISEBERT model takes as input the concatenation of two descriptions for each candidate–entity pair, where conceptually we must process every possible pair in the training set. Such a setting would be cost prohibitive computationally.

One way to reduce the cost of POINTWISEBERT would be to ignore candidate pairs that are unlikely to be aligned. Rao et al. (2016) showed that uncertainty-based sampling can provide extra improvements in ranking. Following this idea, the GCN-based models (i.e., MAN and HMAN) are used to generate a candidate pool whose size is much smaller than the entire universe of entities. Specifically, GCN-based models provide top-$q$ candidates of target entities for each source entity (where $q$ is a hyperparameter). Then, the POINTWISEBERT model produces a ranking score for each candidate–entity pair in the pool to further rerank the candidates. However, the weakness of such a reranking strategy is that performance is bounded by the quality of (potentially limited) candidates produced by MAN or HMAN.

**Weighted Concatenation.** With the textual embeddings learned by PAIRWISEBERT denoted as $H^B$ and graph embeddings denoted as $H^G$, a simple way to combine the two modules is by weighted concatenation:

$$H^C = \tau \cdot H^G \oplus (1 - \tau) \cdot H^B \quad (5)$$

where $H^G$ is the graph embeddings learned by either MAN or HMAN, and $\tau$ is a factor to balance the contribution of each source (where $\tau$ is a hyperparameter).

### 3.4 Entity Alignment

After we obtain the embeddings of entities, we leverage $\ell_1$ distance to measure the distance between candidate–entity pairs. A small distance reflects a high probability for an entity pair to be aligned as equivalent entities. To be specific, with respect to the reranking strategy, we select the target entities that have the smallest distances to a source entity in the vector space learned by MAN or HMAN as its candidates. For weighted concatenation, we employ the $\ell_1$ distance of the representations of a pair derived from the concatenated embedding, i.e., $H^C$, as the ranking score.

## 4 Experiments

### 4.1 Datasets and Settings

We evaluate our methods over two benchmark datasets: DBP15K and DBP100K (Sun et al., 2017). Table 1 outlines the statistics of both datasets, which contain 15,000 and 100,000 ILLs, respectively. Both are divided into three subsets: Chinese-English (ZH-EN), Japanese-English (JA-EN), and French-English (FR-EN).

Following previous work (Sun et al., 2017; Wang et al., 2018), we adopt the same split settings in our experiments, where 30% of the ILLs are used as training and the remaining 70% for evaluation. *Hits@k* is used as the evaluation metric (Bordes et al., 2013; Sun et al., 2017; Wang et al., 2018), which measures the proportion of correctly aligned entities ranked in the top-$k$ candidates, and results in both directions, e.g., ZH-EN and EN-ZH, are reported.

In all our experiments, we employ two-layer GCNs and the top 1000 (i.e., $F$=1000) most frequent relation types and attributes are included to build the $N$-hot feature vectors. For the MAN model, we set the dimensionality of topological, relation, and attribute embeddings to 200, 100, and 100, respectively. When training HMAN, the hyperparameters are dependent on the dataset sizes due to GPU memory limitations. For DBP15K, we set the dimensionality of topological embeddings, relation embeddings, and attribute embeddings to 200, 100, and 100, respectively. For DBP100K, the dimensionalities are set to 100, 50, and 50, respectively. We adopt SGD to update parameters and the numbers of epochs are set to 2,000 and 50,000 for MAN and HMAN, respectively. The margin $\beta$ in the loss function is set to 3. The balance factor $\tau$ is determined by grid search, which shows that the best performance lies in the range from 0.8 to 0.7. For simplicity, $\tau$ is set to 0.8 in all associated experiments. Multilingual BERT-base models with 768 hidden units are used in POINTWISEBERT and PAIRWISEBERT. We additionally append one more FC layer to the representation of [CLS] and reduce the dimensionality to 300. Both BERT models are fine-tuned using the Adam optimizer.

| Datasets | | DBP15K | | | | |
|---|---|---|---|---|---|---|
| | | Entities | Rel. | Attr. | Rel.triples | Attr.triples |
| ZH-EN | Chinese | 66,469 | 2,830 | 8,113 | 153,929 | 379,684 |
| | English | 98,125 | 2,317 | 7,173 | 237,674 | 567,755 |
| JA-EN | Japanese | 65,744 | 2,043 | 5,882 | 164,373 | 354,619 |
| | English | 95,680 | 2,096 | 6,066 | 233,319 | 497,230 |
| FR-EN | French | 66,858 | 1,379 | 4,547 | 192,191 | 528,665 |
| | English | 105,889 | 2,209 | 6,422 | 278,590 | 576,543 |
| Datasets | | DBP100K | | | | |
| | | Entities | Rel. | Attr. | Rel.triples | Attr.triples |
| ZH-EN | Chinese | 106,517 | 4,431 | 16,152 | 329,890 | 1,404,615 |
| | English | 185,022 | 3,519 | 14,459 | 453,248 | 1,902,725 |
| JA-EN | Japanese | 117,836 | 2,888 | 12,305 | 413,558 | 1,474,721 |
| | English | 118,570 | 2,631 | 13,238 | 494,087 | 1,738,803 |
| FR-EN | French | 105,724 | 1,775 | 8,029 | 409,399 | 1,361,509 |
| | English | 107,231 | 2,504 | 13,170 | 513,382 | 1,957,813 |

Table 1: Statistics of DBP15K and DBP100K. Rel. and Attr. stand for relations and attributes, respectively.

### 4.2 Results on Graph Embeddings

We first compare MAN and HMAN against previous systems (Hao et al., 2016; Chen et al., 2017a; Sun et al., 2017; Wang et al., 2018). As shown in Table 2, MAN and HMAN consistently outperform all baselines in all scenarios, especially HMAN. It is worth noting that, in this case, MAN and HMAN use as much information as Wang et al. (2018), while Sun et al. (2017) require extra supervised information (relations and attributes of two KGs need to be aligned in advance). The performance improvements confirm that our model can better utilize topological, relational, and attribute information of entities provided by KGs.

To explain why HMAN achieves better results than MAN, recall that MAN collects relation and attribute information by the propagation mechanism in GCNs where such knowledge is exchanged through neighbors, while HMAN uses feedforward networks to capture expressive features directly from the input feature vectors without propagation. As we discussed before, it is not always the case that neighbors of equivalent entities share similar relations or attributes. Propagating such features through linked entities in GCNs may introduce noise and thus harm performance.

| Model | ZH → EN | | | EN→ ZH | | | JA → EN | | | EN→ JA | | | FR → EN | | | EN→ FR | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | @1 | @10 | @50 | @1 | @10 | @50 | @1 | @10 | @50 | @1 | @10 | @50 | @1 | @10 | @50 | @1 | @10 | @50 |
| | DBP15K | | | | | | | | | | | | | | | | | |
| Hao et al. (2016) | 21.2 | 42.7 | 56.7 | 19.5 | 39.3 | 53.2 | 18.9 | 39.9 | 54.2 | 17.8 | 38.4 | 52.4 | 15.3 | 38.8 | 56.5 | 14.6 | 37.2 | 54.0 |
| Chen et al. (2017a) | 30.8 | 61.4 | 79.1 | 24.7 | 52.4 | 70.4 | 27.8 | 57.4 | 75.9 | 23.7 | 49.9 | 67.9 | 24.4 | 55.5 | 74.4 | 21.2 | 50.6 | 69.9 |
| Sun et al. (2017) | 41.1 | 74.4 | 88.9 | 40.1 | 71.0 | 86.1 | 36.2 | 68.5 | 85.3 | 38.3 | 67.2 | 82.6 | 32.3 | 66.6 | 83.1 | 32.9 | 65.9 | 82.3 |
| Wang et al. (2018) | 41.2 | 74.3 | 86.2 | 36.4 | 69.9 | 82.4 | 39.9 | 74.4 | 86.1 | 38.4 | 71.8 | 83.7 | 37.2 | 74.4 | 86.7 | 36.7 | 73.0 | 86.3 |
| MAN | 46.0 | 79.4 | 90.0 | 41.5 | 75.6 | 88.3 | 44.6 | 78.8 | 90.0 | 43.0 | 77.1 | 88.7 | 43.1 | 79.7 | 91.7 | 42.1 | 79.1 | 90.9 |
| MAN w/o TE | 21.5 | 55.0 | 79.4 | 20.2 | 53.6 | 78.8 | 15.0 | 44.0 | 69.9 | 14.3 | 44.0 | 70.6 | 10.2 | 34.5 | 59.5 | 10.8 | 35.2 | 60.3 |
| MAN w/o RE | 45.6 | 79.1 | 89.5 | 41.1 | 75.0 | 87.3 | 44.2 | 78.7 | 89.8 | 43.0 | 76.9 | 88.1 | 42.8 | 79.7 | 91.4 | 42.1 | 78.9 | 90.6 |
| MAN w/o AE | 43.7 | 77.1 | 87.8 | 39.2 | 72.9 | 85.5 | 43.2 | 77.6 | 88.4 | 41.2 | 74.9 | 86.6 | 42.9 | 79.6 | 91.0 | 41.5 | 78.9 | 90.5 |
| HMAN | **56.2** | **85.1** | **93.4** | **53.7** | **83.4** | **92.5** | **56.7** | **86.9** | **94.5** | **56.5** | **86.6** | **94.6** | **54.0** | **87.1** | **95.0** | **54.3** | **86.7** | **95.1** |
| HMAN w/o TE | 3.2 | 16.7 | 38.3 | 3.5 | 17.2 | 38.5 | 5.4 | 22.3 | 45.5 | 5.2 | 22.0 | 45.5 | 2.4 | 13.9 | 35.3 | 2.2 | 13.7 | 35.3 |
| HMAN w/o RE | 50.2 | 78.4 | 86.5 | 49.3 | 78.6 | 87.0 | 52.6 | 81.6 | 89.1 | 52.4 | 81.1 | 89.8 | 52.7 | 84.2 | 91.4 | 52.0 | 83.9 | 91.1 |
| HMAN w/o AE | 49.2 | 81.0 | 89.8 | 48.8 | 80.9 | 90.0 | 52.2 | 83.3 | 91.6 | 51.5 | 83.1 | 91.6 | 52.3 | 85.6 | 93.7 | 52.3 | 85.1 | 93.2 |
| HMAN w/o HW | 46.8 | 76.1 | 84.1 | 46.0 | 76.2 | 84.6 | 50.5 | 79.5 | 87.5 | 49.9 | 79.1 | 87.5 | 51.9 | 82.7 | 90.9 | 51.6 | 82.5 | 90.6 |
| | DBP100K | | | | | | | | | | | | | | | | | |
| Hao et al. (2016) | - | 16.9 | - | - | 16.6 | - | - | 21.1 | - | - | 20.9 | - | - | 22.9 | - | - | 22.6 | - |
| Chen et al. (2017a) | - | 34.3 | - | - | 29.1 | - | - | 33.9 | - | - | 27.2 | - | - | 44.8 | - | - | 39.1 | - |
| Sun et al. (2017) | 20.2 | 41.2 | 58.3 | 19.6 | 39.4 | 56.0 | 19.4 | 42.1 | 60.5 | 19.1 | 39.4 | 55.9 | 26.2 | 54.6 | 70.5 | 25.9 | 51.3 | 66.9 |
| Wang et al. (2018) | 23.1 | 47.5 | 63.8 | 19.2 | 40.3 | 55.4 | 26.4 | 55.1 | 70.0 | 21.9 | 44.4 | 56.6 | 29.2 | 58.4 | 68.7 | 25.7 | 50.5 | 59.8 |
| MAN | 27.2 | 54.2 | **72.8** | 24.7 | 50.2 | **69.0** | 30.0 | 60.4 | **77.3** | 26.6 | 54.4 | 71.2 | 31.6 | 64.0 | 77.3 | 28.8 | 59.3 | 73.4 |
| MAN w/o TE | 11.8 | 28.6 | 47.7 | 11.2 | 28.3 | 47.9 | 7.4 | 21.7 | 39.4 | 7.2 | 21.6 | 39.8 | 5.4 | 19.4 | 38.2 | 5.1 | 18.8 | 37.1 |
| MAN w/o RE | 26.5 | 53.4 | 72.1 | 23.9 | 49.2 | 67.9 | 29.8 | 60.3 | 77.1 | 26.3 | 53.9 | 70.6 | 31.0 | 63.2 | 76.4 | 28.4 | 58.4 | 72.2 |
| MAN w/o AE | 25.5 | 51.7 | 70.4 | 22.8 | 47.6 | 66.3 | 29.4 | 59.4 | 76.1 | 25.9 | 52.9 | 69.7 | 30.8 | 62.7 | 75.8 | 28.1 | 57.8 | 71.5 |
| HMAN | **29.8** | **54.6** | 69.5 | **28.7** | **53.3** | **69.0** | **34.3** | **63.3** | 76.1 | **33.8** | **63.0** | **76.7** | **37.5** | **67.7** | **77.7** | **37.6** | **68.1** | **78.5** |
| HMAN w/o TE | 6.8 | 20.3 | 39.2 | 7.2 | 21.0 | 39.4 | 3.0 | 11.5 | 27.3 | 3.3 | 11.8 | 28.0 | 0.5 | 3.5 | 11.1 | 0.5 | 3.4 | 11.4 |
| HMAN w/o RE | 28.0 | 50.3 | 62.3 | 28.2 | 50.6 | 62.9 | 30.3 | 54.9 | 64.8 | 30.2 | 55.9 | 66.9 | 32.8 | 60.3 | 69.1 | 33.3 | 60.9 | 69.8 |
| HMAN w/o AE | 25.7 | 46.4 | 57.3 | 25.5 | 64.7 | 57.9 | 29.6 | 55.1 | 66.1 | 29.9 | 56.1 | 67.4 | 32.5 | 59.2 | 67.8 | 32.9 | 59.4 | 68.4 |
| HMAN w/o HW | 25.2 | 46.0 | 57.9 | 25.2 | 45.9 | 57.9 | 28.6 | 52.6 | 62.2 | 28.5 | 53.0 | 63.0 | 32.8 | 60.9 | 70.0 | 32.9 | 60.2 | 70.3 |

Table 2: Results of using graph information on DBP15K and DBP100K. @1, @10 and @50 refer to Hits@1, Hits@10 and Hits@50, respectively.

Moreover, we perform ablation studies on the two proposed models to investigate the effectiveness of each component. We alternatively remove each aspect of features (i.e., topological, relation, and attribute features) and the highway layer in HMAN, denoted as w/o TE (RE, AE, and HW). As reported in Table 2, we observe that after removing relation or attribute features, the performance of HMAN and MAN drops across all datasets. These figures prove that these two aspects of features are useful in making alignment decisions. On the other hand, compared to MAN, HMAN shows more significant performance drops, which also demonstrates that employing the feedforward networks can better categorize relation and attribute features than GCNs in this scenario. Interestingly, looking at the two variants MAN w/o TE and HMAN w/o TE, we can see the former achieves better results. Since MAN propagates relation and attribute features via graph structures, it can still implicitly capture topological knowledge of entities even after we remove the topological features. However, HMAN loses such structure knowledge when topological features are excluded, and thus its results are worse. From these experiments, we can conclude that the topological information is playing an indispensable role in making alignment decisions.

### 4.3 Results with Textual Embeddings

In this section, we discuss empirical results involving the addition of entity descriptions, shown in Table 3. Applying literal descriptions of entities to conduct cross-lingual entity alignment is relatively under-explored. The recent work of Chen et al. (2018) used entity descriptions in their model; however, we are unable to make comparisons with their work, as we do not have access to their code and data. Since we employ BERT to learn textual embeddings of descriptions, we consider systems that also use external resources, like Google Translate,[1] as our baselines. We directly

[1] https://cloud.google.com/translate/

| Model | ZH → EN | | | EN→ ZH | | | JA → EN | | | EN→ JA | | | FR → EN | | | EN→ FR | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | @1 | @10 | @50 | @1 | @10 | @50 | @1 | @10 | @50 | @1 | @10 | @50 | @1 | @10 | @50 | @1 | @10 | @50 |
| | DBP15K | | | | | | | | | | | | | | | | | |
| Translation∗ | 55.7 | 67.6 | 74.3 | 40.3 | 54.2 | 62.2 | 74.6 | 84.5 | 89.1 | 61.9 | 72.0 | 77.2 | - | - | - | - | - | - |
| JAPE + Translation∗ | 73.0 | 90.4 | 96.6 | 62.7 | 85.2 | 94.2 | 82.8 | 94.6 | 98.3 | 75.9 | 90.7 | 96.0 | - | - | - | - | - | - |
| PAIRWISEBERT | 74.3 | 94.6 | 98.8 | 74.8 | 94.7 | 99.0 | 78.6 | 95.8 | 98.5 | 78.3 | 95.4 | 98.4 | 95.2 | 99.2 | 99.6 | 94.9 | 99.2 | 99.7 |
| MAN (RERANK) | 84.2 | 93.6 | 94.8 | 82.1 | 91.8 | 93.1 | 89.4 | 94.0 | 94.8 | 88.2 | 93.3 | 94.0 | 93.1 | 95.2 | 95.4 | 93.1 | 95.3 | 95.4 |
| HMAN (RERANK) | 86.5 | 95.9 | 96.9 | 85.8 | 94.1 | 95.3 | 89.0 | 96.0 | 97.3 | 89.0 | 96.0 | 97.5 | 95.3 | 97.7 | 97.8 | 95.2 | 97.9 | 98.1 |
| MAN (WEIGHTED) | 85.4 | 98.2 | 99.7 | 83.8 | 97.7 | 99.5 | 90.8 | 98.8 | 99.7 | 89.9 | 98.5 | 99.5 | 96.8 | 99.6 | 99.8 | 96.7 | 99.7 | **99.9** |
| HMAN (WEIGHTED) | **87.1** | **98.7** | **99.8** | **86.4** | **98.5** | **99.8** | **93.5** | **99.4** | **99.9** | **93.3** | **99.3** | **99.9** | **97.3** | **99.8** | **99.9** | **97.3** | **99.8** | **99.9** |
| | DBP100K | | | | | | | | | | | | | | | | | |
| PAIRWISEBERT | 65.1 | 85.1 | 92.6 | 66.2 | 85.8 | 92.9 | 67.7 | 86.5 | 93.1 | 67.9 | 86.4 | 93.2 | 93.2 | 97.9 | 98.9 | 93.4 | 98.0 | 98.9 |
| MAN (RERANK) | 59.5 | 62.1 | 62.2 | 55.9 | 58.2 | 58.2 | 65.5 | 68.2 | 68.4 | 59.9 | 62.1 | 62.3 | 69.7 | 70.4 | 70.5 | 65.5 | 66.2 | 66.2 |
| HMAN (RERANK) | 58.9 | 61.2 | 61.3 | 57.9 | 60.2 | 60.3 | 66.9 | 69.4 | 69.6 | 67.0 | 69.6 | 69.8 | 72.1 | 72.9 | 73.0 | 72.7 | 73.5 | 73.5 |
| MAN (WEIGHTED) | **81.4** | **94.9** | **98.2** | **80.5** | 94.1 | 97.7 | 84.3 | 95.4 | 98.3 | 81.5 | 94.2 | 97.6 | 96.2 | 99.3 | **99.7** | 95.7 | 99.1 | 99.6 |
| HMAN (WEIGHTED) | 81.1 | 94.3 | 97.8 | 80.3 | **94.5** | **97.9** | **85.2** | **96.1** | **98.4** | **84.6** | **96.1** | **98.5** | **96.5** | **99.4** | **99.7** | **96.5** | **99.5** | **99.8** |

Table 3: Results of using both graph and textual information on DBP15K and DBP100K. @1, @10, and @50 refer to Hits@1, Hits@10, and Hits@50, respectively. ∗ indicates results are taken from (Sun et al., 2017).

take results reported by Sun et al. (2017), denoted as "Translation" and "JAPE+Translation".

The POINTWISEBERT model is used with GCN-based models, which largely reduces the search space, as indicated by MAN (RERANK) and HMAN (RERANK), where the difference is that the candidate pools are given by MAN and HMAN, respectively. For DBP15K, we select top-200 candidate target entities as the candidate pool while for DBP100K, top-20 candidates are selected due to its larger size. The reranking method does lead to performance gains across all datasets, where the improvements are dependent on the quality of the candidate pools. HMAN (RERANK) generally performs better than MAN (RERANK) since HMAN recommends more promising candidate pools.

The PAIRWISEBERT model learns the textual embeddings that map cross-lingual descriptions into the same space, which can be directly used to align entities. The results are listed under PAIRWISEBERT in Table 3. We can see that it achieves good results on its own, which also shows the efficacy of using multilingual descriptions. Moreover, such textual embeddings can be combined with graph embeddings (learned by MAN or HMAN) by weighted concatenation, as discussed in Section 3.3. The results are reported as MAN (WEIGHTED) and HMAN (WEIGHTED), respectively. As we can see, this simple operation leads to significant improvements and gives excellent results across all datasets. However, it is not always the case that KGs provide descriptions for every entity. For those entities whose descriptions are not available, the graph embeddings would be the only source for making alignment decisions.

| | English | Chinese |
|---|---|---|
| **ILL pair** | Casino_Royale_(2006_film) (3) | 007大戰皇家賭場 (3) |
| **Features** | starring, starring, distributor | starring, starring, language |
| **Neighbors** | Daniel_Craig (1), Eva_Green (4), Columbia_Pictures (9) | 丹尼爾·克雷格 (1), 伊娃·格蓮 (4), 英語 (832) |

Table 4: Case study of the noise introduced by the propagation mechanism.

## 4.4 Case Study

In this section, we describe a case study to understand the performance gap between HMAN and MAN. The example in Table 4 provides insights potentially explaining this performance gap. We argue that MAN introduces unexpected noise from heterogeneous nodes during the GCN propagation process. We use the number in parentheses (*) after entity names to denote the number of relation features they have.

In this particular example, the two entities "*Casino_Royale_(2006_film)*" in the source language (English) and "*007*大戰皇家賭場" in the target language (Chinese) both have three relation features. We notice that the propagation mechanism introduces some neighbors which are unable to find cross-lingual counterparts from the other end, marked in red. Considering the entity "英語" (English), a neighbor of "*007*大戰皇家賭場", no counterparts can be found in the neighbors of "*Casino_Royale_(2006_film)*". We also observe that "英語" (English) is a pivot node in

the Chinese KG and has 832 relations, such as "語言" (Language), "官方語言" (Official Language), and "頻道語言" (Channel Language). In this case, propagating features from neighbors can harm performance. In fact, the feature sets of the ILL pair already convey information that captures their similarity (e.g., the "starring" marked in blue are shared twice). Therefore, by directly using feedforward networks, HMAN is able to effectively capture such knowledge.

## 5 Related Work

**KG Alignment.** Research on KG alignment can be categorized into two groups: monolingual and multilingual entity alignment. As for monolingual entity alignment, main approaches align two entities by computing string similarity of entity labels (Scharffe et al., 2009; Volz et al., 2009; Ngomo and Auer, 2011) or graph similarity (Raimond et al., 2008; Pershina et al., 2015; Azmy et al., 2019). Recently, Trsedya et al. (2019) proposed an embedding-based model that incorporates attribute values to learn the entity embeddings.

To match entities in different languages, Wang et al. (2012) leveraged only language-independent information to find possible links cross multilingual Wiki knowledge graphs. Recent studies learned cross-lingual embeddings of entities based on TransE (Bordes et al., 2013), which are then used to align entities across languages. Chen et al. (2018) designed a co-training algorithm to alternately learn multilingual entity and description embeddings. Wang et al. (2018) applied GCNs with the connectivity matrix defined on relations to embed entities from multilingual KGs into a unified low-dimensional space.

In this work, we also employ GCNs. However, in contrast to Wang et al. (2018), we regard relation features as input to our models. In addition, we investigate two different ways to capture relation and attribute features.

**Multilingual Sentence Representations.** Another line of research related to this work is aligning sentences in multiple languages. Recent works (Hermann and Blunsom, 2014; Conneau et al., 2018; Eriguchi et al., 2018) studied cross-lingual sentence classification via zero-shot learning. Johnson et al. (2017) proposed a sequence-to-sequence multilingual machine translation system where the encoder can be used to produce cross-lingual sentence embeddings (Artetxe and Schwenk, 2018). Recently, BERT (Devlin et al., 2019) has advanced the state-of-the-art on multiple natural language understanding tasks. Specifically, multilingual BERT enables learning representations of sentences under multilingual settings. We adopt BERT to produce cross-lingual representations of entity literal descriptions to capture their semantic relatedness, which benefits cross-lingual entity alignment.

## 6 Conclusion and Future Work

In this work, we focus on the task of cross-lingual entity alignment, which aims to discover mappings of equivalent entities in multilingual knowledge graphs. We proposed two GCN-based models and two uses of multilingual BERT to investigate how to better utilize multi-aspect information of entities provided by KGs, including topological connections, relations, attributes, and entity descriptions. Empirical results demonstrate that our best model consistently achieves state-of-the-art performance across all datasets. In the future, we would like to apply our methods to other multilingual datasets such as YAGO and BabelNet. Also, since literal descriptions of entities are not always available, we will investigate alternative ways to design graph-based models that can better capture structured knowledge for this task.

## Acknowledgments

We would like to thank the three anonymous reviewers for their thoughtful and constructive comments. Waterloo researchers are supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada, with additional computational resources provided by Compute Ontario and Compute Canada. Yanyan Zou and Wei Lu are supported by Singapore Ministry of Education Academic Research Fund (AcRF) Tier 2 Project MOE2017-T2-1-156, and are partially supported by SUTD project PIE-SGP-AI-2018-01.

## References

Mikel Artetxe and Holger Schwenk. 2018. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *arXiv preprint arXiv:1812.10464*.

Michael Azmy, Peng Shi, Jimmy Lin, and Ihab F. Ilyas. 2019. Matching entities across different knowledge graphs with graph embeddings. *arXiv preprint arXiv:1903.06607*.

Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. 2009. DBpedia: A crystallization point for the web of data. *Web Semantics: science, services and agents on the world wide web*, 7(3):154–165.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of NIPS*.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *Journal of Artificial Intelligence Research*, 49:1–47.

Muhao Chen, Yingtao Tian, Kai-Wei Chang, Steven Skiena, and Carlo Zaniolo. 2018. Co-training embeddings of knowledge graphs and entity descriptions for cross-lingual entity alignment. In *Proceedings of IJCAI*.

Muhao Chen, Yingtao Tian, Mohan Yang, and Carlo Zaniolo. 2017a. Multilingual knowledge graph embeddings for cross-lingual knowledge alignment. In *Proceedings of IJCAI*.

Muhao Chen, Tao Zhou, Pei Zhou, and Carlo Zaniolo. 2017b. Multi-graph affinity embeddings for multilingual knowledge graphs. In *Proceedings of NIPS Workshop on Automated Knowledge Base Construction*.

Guillem Collell, Ted Zhang, and Marie-Francine Moens. 2017. Imagined visual representations as multimodal embeddings. In *Proceedings of AAAI*.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of EMNLP*.

Kareem Darwish. 2013. Named entity recognition using cross-lingual resources: Arabic as an example. In *Proceedings of ACL*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL*.

Akiko Eriguchi, Melvin Johnson, Orhan Firat, Hideto Kazawa, and Wolfgang Macherey. 2018. Zero-shot cross-lingual classification using multilingual neural machine translation. *arXiv preprint arXiv:1809.04686*.

Zhijiang Guo, Yan Zhang, Zhiyang Teng, and Wei Lu. 2019a. Attention guided graph convolutional networks for relation extraction. In *Proceedings of ACL*.

Zhijiang Guo, Yan Zhang, Zhiyang Teng, and Wei Lu. 2019b. Densely connected graph convolutional networks for graph-to-sequence learning. *Transactions of the Association of Computational Linguistics*.

Yanchao Hao, Yuanzhe Zhang, Shizhu He, Kang Liu, and Jun Zhao. 2016. A joint embedding method for entity alignment of knowledge bases. In *Proceedings of China Conference on Knowledge Graph and Semantic Computing*.

Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual models for compositional distributed semantics. In *Proceedings of ACL*.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *TACL*, 5:339–351.

Douwe Kiela and Léon Bottou. 2014. Learning image embeddings using convolutional neural networks for improved multi-modal semantics. In *Proceedings of EMNLP*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of ICLR*.

Diego Moussallem, Matthias Wauer, and Axel-Cyrille Ngonga Ngomo. 2018. Machine translation using semantic web technologies: A survey. *Journal of Web Semantics*, 51:1–19.

Sidharth Mudgal, Han Li, Theodoros Rekatsinas, An-Hai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. 2018. Deep learning for entity matching: A design space exploration. In *Proceedings of International Conference on Management of Data*.

Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.

Axel-Cyrille Ngonga Ngomo and Soren Auer. 2011. Limes: a time-efficient approach for large-scale link discovery on the web of data. In *Proceedings of IJCAI*.

Maria Pershina, Mohamed Yakout, and Kaushik Chakrabarti. 2015. Holistic entity matching across knowledge graphs. In *Proceedings of IEEE International Conference on Big Data*.

Yves Raimond, Christopher Sutton, and Mark B. Sandler. 2008. Automatic interlinking of music datasets on the semantic web. In *Proceedings of WWW workshop on Linked Data on the Web*.

Jinfeng Rao, Hua He, and Jimmy Lin. 2016. Noise-contrastive estimation for answer selection with deep neural networks. In *Proceedings of CIKM*.

Thomas Rebele, Fabian Suchanek, Johannes Hoffart, Joanna Biega, Erdal Kuzey, and Gerhard Weikum. 2016. YAGO: A multilingual knowledge base from

Wikipedia, WordNet, and GeoNames. In *Proceedings of International Semantic Web Conference*.

François Scharffe, Yanbin Liu, and Chuguang Zhou. 2009. RDF-AI: an architecture for RDF datasets matching, fusion and interlink. In *Proceedings of IJCAI workshop on Identity and Reference in Web-based Knowledge Representation*.

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *Proceedings of European Semantic Web Conference*.

Peng Shi, Jinfeng Rao, and Jimmy Lin. 2019. Simple attention-based representation learning for ranking short social media posts. In *Proceedings of NAACL*.

Dennis Spohr, Laura Hollink, and Philipp Cimiano. 2011. A machine learning approach to multilingual and cross-lingual ontology matching. In *Proceedings of International Semantic Web Conference*.

Rupesh K. Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In *Proceedings of NIPS*.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2008. YAGO: A large ontology from Wikipedia and WordNet. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(3):203–217.

Zequn Sun, Wei Hu, and Chengkai Li. 2017. Cross-lingual entity alignment via joint attribute-preserving embedding. In *Proceedings of International Semantic Web Conference*.

Bayu Distiawan Trsedya, Jianzhong Qi, and Rui Zhang. 2019. Entity alignment between knowledge graphs using attribute embeddings. In *Proceedings of AAAI*.

Julius Volz, Christian Bizer, Martin Gaedke, and Georgi Kobilarov. 2009. Discovering and maintaining links on the web of data. In *Proceedings of International Semantic Web Conference*.

Zhichun Wang, Juanzi Li, Zhigang Wang, and Jie Tang. 2012. Cross-lingual knowledge linking across wiki knowledge bases. In *Proceedings of WWW*.

Zhichun Wang, Qingsong Lv, Xiaohan Lan, and Yu Zhang. 2018. Cross-lingual knowledge graph alignment via graph convolutional networks. In *Proceedings of EMNLP*.

Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation learning of knowledge graphs with entity descriptions. In *Proceedings of AAAI*.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of EMNLP*.

Hao Zhu, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2017. Iterative entity alignment via joint knowledge embeddings. In *Proceedings of IJCAI*.

# Contrastive Language Adaptation for Cross-Lingual Stance Detection

**Mitra Mohtarami**[1], **James Glass**[1], **Preslav Nakov**[2]
[1]MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA, USA
[2]Qatar Computing Research Institute, HBKU, Doha, Qatar
{mitra,glass}@csail.mit.edu; pnakov@hbku.edu.qa

## Abstract

We study cross-lingual stance detection, which aims to leverage labeled data in one language to identify the relative perspective (or stance) of a given document with respect to a claim in a different target language. In particular, we introduce a novel contrastive language adaptation approach applied to memory networks, which ensures accurate alignment of stances in the source and target languages, and can effectively deal with the challenge of limited labeled data in the target language. The evaluation results on public benchmark datasets and comparison against current state-of-the-art approaches demonstrate the effectiveness of our approach.

## 1 Introduction

The rise of social media has enabled the phenomenon of "fake news," which could target specific individuals and can be used for deceptive purposes (Lazer et al., 2018; Vosoughi et al., 2018). As manual fact-checking is a time-consuming and tedious process, computational approaches have been proposed as a possible alternative (Popat et al., 2017; Wang, 2017; Mihaylova et al., 2018, 2019), based on information sources such as social media (Ma et al., 2017), Wikipedia (Thorne et al., 2018), and knowledge bases (Huynh and Papotti, 2018). Fact-checking is a multi-step process (Vlachos and Riedel, 2014): (*i*) checking the reliability of media sources, (*ii*) retrieving potentially relevant documents from reliable sources as evidence for each target claim, (*iii*) predicting the stance of each document with respect to the target claim, and finally (*iv*) making a decision based on the stances from (*iii*) for all documents from (*ii*).

Here, we focus on stance detection which aims to identify the relative perspective of a document with respect to a claim, typically modeled using labels such as *agree*, *disagree*, *discuss*, and *unrelated*. Current approaches to stance detection (Bar-Haim et al., 2017; Dungs et al., 2018; Kochkina et al., 2018; Sobhani et al., 2017; Mohtarami et al., 2018) are well-studied in mono-lingual settings, in particular for English, but less attention has been paid to other languages and *cross-lingual* settings. This is partially due to domain differences and to the lack of training data in other languages.

We aim to bridge this gap by proposing a cross-lingual model for stance detection. Our model leverages resources of a source language (e.g., English) to train a model for a target language (e.g., Arabic). Furthermore, we propose a novel contrastive language adaptation approach that effectively aligns samples with similar or dissimilar stances across source and target languages using task-specific loss functions. We apply our language adaptation approach to memory networks (Sukhbaatar et al., 2015), which have been found effective for mono-lingual stance detection (Mohtarami et al., 2018).

Our model can explain its predictions about stances of documents against claims in a different/target language by extracting relevant text snippets from the documents of the target language as evidence. We use evidence extraction as a measure to evaluate the trasferability of our model. This is because more accurate evidence extraction indicates that the model can better learn semantic relations between claims and pieces of evidence, and consequently can better transfer knowledge to the target language.

The contributions of this paper are summarized as follows:

- We propose a novel language adaptation approach based on contrastive stance alignment that aligns the class labels between source and target languages for effective cross-lingual stance detection.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4442–4452, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

- Our model is able to extract accurate text snippets as evidence to explain its predictions in the target language (results are in Section 4.2).
- To the best of our knowledge, this is the first work on cross-lingual stance detection.

We conducted our experiments on English (as source language) and Arabic (as target language). In particular, we used the Fake News Challenge dataset (Hanselowski et al., 2018) as source data and an Arabic benchmark dataset (Baly et al., 2018) as target data. The evaluation results have shown 2.7 and 4.0 absolute improvement in terms of macro-F1 and weighted accuracy for stance detection over the current state-of-the-art mono-lingual baseline, and 11.4, 14.9, 16.1, 12.9, and 13.1 points of absolute improvement in terms of precision at ranks 1–5 for extracting evidence snippets respectively. Furthermore, a key finding in our investigation is that, in contrast to other tasks (Devlin et al., 2019; Peters et al., 2018), pre-training with large amounts of source data is less effective for cross-lingual stance detection. We show that this is because pre-training can considerably bias the model toward the source language.

## 2 Method

Assume that we are given a training dataset for a source language, $\mathcal{D}_s$, which contains a set of triplets as follows: $\mathcal{D}_s = \left\{\left((c_i^s, d_i^s), y_i^s\right)\right\}_{i=1}^{N}$, where $N$ is the number of source samples, $(c_i^s, d_i^s)$ is a pair of claim $c_i^s$ and document $d_i^s$, and $y_i^s \in Y$, $Y = \{agree, disagree, discuss, unrelated\}$, is the corresponding label indicating the stance of the document with respect to the claim. In addition, we are given a very small training dataset for the target language $\mathcal{D}_t = \left\{\left((c_i^t, d_i^t), y_i^t\right)\right\}_{i=1}^{M}$, where $M$ ($M << N$) is the number of target samples, and $(c_i^t, d_i^t)$ is a pair of claim and document in the target language with stance label $y_i^t \in Y$. In reality, (*i*) the size of the target dataset is very small, (*ii*) claims and documents in the source and target languages are from different domains, and (*iii*) the only commonality between the source and target datasets is in their stance labels, i.e., $y_i^s, y_i^t \in Y$.

We develop a language adaptation approach to effectively use the commonality between the source and the target datasets in their label space and to deal with the limited size of the target training data. We apply our language adaptation approach to end-to-end memory networks (Sukhbaatar et al., 2015) for cross-lingual stance detection.

We use memory networks as they have achieved state-of-the-art performance for mono-lingual stance detection (Mohtarami et al., 2018). However, our language adaptation approach can be applied to any other type of neural network. The architecture of our cross-lingual stance detection model is shown in Figure 1. It has two main components: (*i*) *Memory Networks* indicated with two dashed boxes for the source and the target languages, and (*ii*) *Contrastive Language Adaptation* component. In what follows, we first explain our memory network model for cross-lingual stance detection (Section 2.1) and then present our contrastive language adaptation approach (Section 2.2).

### 2.1 Memory Networks

Memory networks are designed to remember past information (Sukhbaatar et al., 2015) and have been successfully applied to NLP tasks ranging from dialog (Bordes et al., 2017) to question answering (Xiong et al., 2016) and mono-lingual stance detection (Mohtarami et al., 2018). They include components that can potentially use different learning models and inference strategies. Our source and target memory networks follow the same architecture as depicted in Figure 1:

A memory network consists of six components. The network takes as input a document $d$ and a claim $c$ and encodes them into the *input space* $I$. These representations are stored in the *memory* component $M$ for future processing. The relevant parts of the input are identified in the *inference* component $F$, and used by the *generalization* component $G$ to update the memory $M$. Finally, the *output* component $O$ generates an output from the updated memory, and encodes it to a desired format in the *response* component $R$ using a prediction function, e.g., `softmax` for classification tasks. We elaborate on these components below.

**Input representation component $I$:** It encodes documents and claims into corresponding representations. Each document $d$ is divided into a sequence of paragraphs $X = (x_1, \ldots, x_l)$, where each $x_j$ is encoded as $\mathbf{m}_j$ using an LSTM network, and as $\mathbf{n}_j$ using a CNN; these representations are stored in the memory component $M$. Note that while LSTMs are designed to capture and memorize their inputs (Tan et al., 2016), CNNs emphasize the local interaction between individual words in sequences, which is important for obtaining good representation (Kim, 2014).

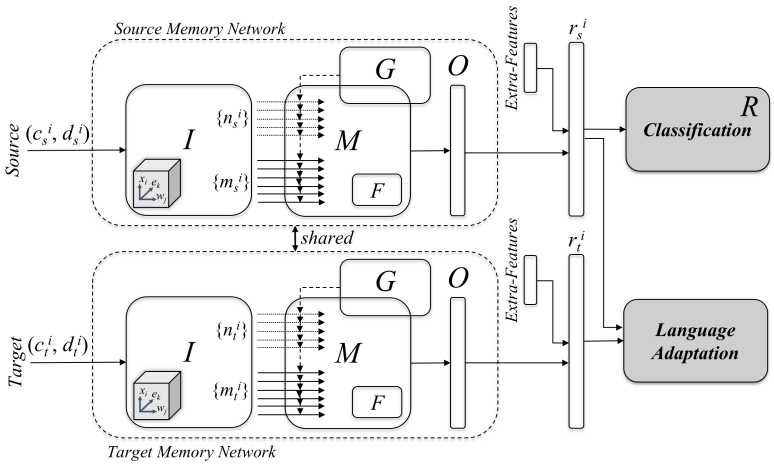


Figure 1: The architecture of our cross-lingual memory network for stance detection.

Thus, our $I$ component uses both LSTM and CNN representations. It also uses separate LSTM and CNN with their own parameters to represent each input claim $c$ as $\mathbf{c}_{lstm}$ and $\mathbf{c}_{cnn}$, respectively.

We consider each paragraph as a single piece of *evidence* because a paragraph usually represents a coherent argument, unified under one or more interrelated topics. We thus use the terms paragraph and evidence interchangeably.

**Inference component $F$:** Our inference component computes LSTM- and CNN-based similarity between each claim $c$ and evidence $x_j$ as follows:

$$P_j^{lstm} = \mathbf{c}_{lstm}{}^{\intercal} \times \mathbf{M} \times \mathbf{m}_j, \forall j$$
$$P_j^{cnn} = \mathbf{c}_{cnn}{}^{\intercal} \times \mathbf{M}' \times \mathbf{n}_j, \forall j$$

where $P_\cdot^{lstm}$ and $P_\cdot^{cnn}$ indicate claim-evidence similarity based on LSTM and CNN respectively, $\mathbf{c}_{lstm} \in \mathbb{R}^q$ and $\mathbf{m}_j \in \mathbb{R}^d$ are LSTM representations of $c$ and $x_j$ respectively, $\mathbf{c}_{cnn} \in \mathbb{R}^{q'}$ and $\mathbf{n}_j \in \mathbb{R}^{d'}$ are the corresponding CNN representations, and $\mathbf{M} \in \mathbb{R}^{q \times d}$ and $\mathbf{M}' \in \mathbb{R}^{q' \times d'}$ are similarity matrices trained to map claims and paragraphs into the same space with respect to their LSTM and CNN representations. The rationale behind using these similarity matrices is that, in the memory network, we seek a transformation of the input claim, i.e., $\mathbf{c}^{\intercal} \times \mathbf{M}$, in order to obtain the closest evidence to the claim.

Additionally, we compute another semantic similarity vector, $P_j^{tfidf}$, by applying a cosine similarity between the TF.IDF (Spärck Jones, 2004) representation of $x_j$ and $c$. This is particularly useful for stance detection as it can help filtering out unrelated pieces of evidence.

**Memory $M$ and Generalization $G$ components:** Our memory component stores representations and the generalization component improves their quality by filtering out unrelated evidence. For example, the LSTM representations of paragraphs, $\mathbf{m}_j, \forall j$, are updated using the claim-evidence similarity $P_j^{tfidf}$ as follows: $\mathbf{m}_j = \mathbf{m}_j \odot P_j^{tfidf}, \forall j$. This transformation will help filter out unrelated evidence with respect to claims. The updated $\mathbf{m}_j$ in conjunction with $\mathbf{c}_{lstm}$ are used by the inference component $F$ to compute $P_j^{lstm}, \forall j$ as we explained above. Then, $P_j^{lstm}$ are in turn used to update CNN representations in memory as follows: $\mathbf{n}_j = \mathbf{n}_j \odot P_j^{lstm}, \forall j$. Finally, the updated $\mathbf{n}_j$ and $\mathbf{c}_{cnn}$ are used to compute $P_j^{cnn}$.

**Output representation component $O$:** This component computes the output of the memory $M$ by concatenating the average vector of the updated $\mathbf{n}_j$ with the *maximum* and *average* of claim-evidence similarity vectors $P_j^{tfidf}$, $P_j^{lstm}$ and $P_j^{cnn}$. The maximum helps to identify parts of documents that are most similar to claims, while the average estimates the overall document-claim similarity.

**Response generation component $R$:** This component computes the final stance of a document with respect to a claim. For this, the output of component $O$ is concatenated with $\mathbf{c}_{lstm}$ and $\mathbf{c}_{cnn}$ and fed into a `softmax` to predict the stance of the document with respect to the claim.

All the memory network parameters, including those of CNN and LSTM in the $I$ component, the similarity matrices $\mathbf{M}$ and $\mathbf{M}'$ in $F$, and the classifier parameters in $R$, are jointly learned during the training process with our language adaptation.

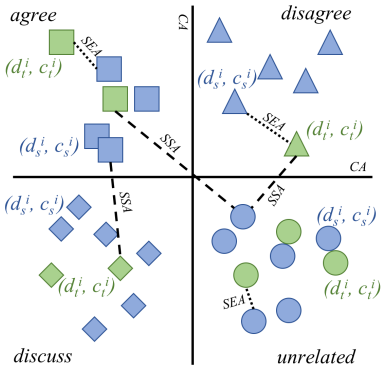


Figure 2: Illustration of stance equal alignment (SEA), stance separation alignment (SSA), and classification alignment (CA) constraints. Different shapes indicate different stance labels and colors specify source (*blue*) and target (*green*) languages.

## 2.2 Contrastive Language Adaptation

Memory networks are effective for stance detection in mono-lingual settings (Mohtarami et al., 2018) when there is sufficient training data. However, we show that these models have limited transferability to target languages with limited data. This could be due to discrepancy between the underlying data distributions in the source and target languages. We show that the performance of these networks can be trivially increased when the model, pre-trained on source data, is fine-tuned using small amounts of labeled target data. We further develop contrastive language adaptation that can exploit the labeled source data to perform well on target data.

Our contrastive adaptation approach:

- encourages pairs $(d_i^s, c_i^s)$ from the source language and $(d_i^t, c_i^t)$ from the target language with the same stance labels (i.e., $y_i^s = y_i^t$) to be nearby in the embedding space. We call this mapping *Stance Equal Alignment* (SEA), illustrated with dotted lines in Figure 2. Note that documents and claims in the two languages are often semantically different and are not corresponding translations of each other.
- encourages pairs $(d_i^s, c_i^s)$ from the source language and $(d_i^t, c_i^t)$ from the target language with different stance labels (i.e., $y_i^s \neq y_i^t$) to be far apart in the embedding space. We call this mapping *Stance Separation Alignment* (SSA), shown with dashed lines in Figure 2.
- encourages pairs $(d_i^s, c_i^s)$ from the source language and $(d_i^t, c_i^t)$ from the target language to be correctly classified as $(d_i^s, c_i^s) \rightarrow y_i^s$ and $(d_i^t, c_i^t) \rightarrow y_i^t$. We call this *Classification Alignment* (CA), solid lines in Figure 2.

**Algorithm 1. Cross-Lingual Stance Detection Model**

**Inputs:**

- $\{(d_i^s, c_i^s), y_i^s\}_1^{l^s}$: set of pairs of documents $d_i^s$ and claims $c_i^s$ in the source language, where $y_i^s \in Y$, $Y = \{agree, disagree, discuss, unrelated\}$
- $\{(d_i^t, c_i^t), y_i^t\}_1^{l^t}$: small set of pairs of documents $d_i^t$ and claims $c_i^t$ in the target language with $y_i^t \in Y$

**Output:**

- $\{(d_i^t, c_i^t) \rightarrow y_i^t\}_{l^t+1}^{M}$: assign stance labels $y_i^t \in Y$ to given unlabeled target pairs

**Cross-Lingual model:**

1 Create the sets $\{\big((d_i^s, c_i^s), (d_j^t, c_j^t)\big), (y_i^s, y_i')\}$ and $\{\big((d_j^t, c_j^t), (d_i^s, c_i^s)\big), (y_j^t, y_j')\}$, where $y' = 1$ if the source and the target have the same label, and $y' = 0$ otherwise.
2 **Loop for** $e$ **epochs:**
3 pass $(d_i^s, c_i^s)$ to the source memory network to create its representation $r_i^s$.
4 pass $(d_j^t, c_j^t)$ to the target memory network to create its representation $r_j^t$.
5 pass $(r_i^s, y_i^s)$ to the classification to compute its classification loss $L_{CA_i^s}$.
6 pass $(r_i^s, r_i^t, y'^i)$ to the language adaptation to compute the stance alignment loss $L_{CSA_i}$.
7 compute total loss $L_i^s = (1-\alpha)L_{CA_i^s} + \alpha L_{CSA_i}$.
8 repeat steps 2-7 with a change in step 5 by passing the target sample $(r_j^t, y_j^t)$ to the classification instead of the source sample, and compute its $L_j^t$ in step 7.
9 jointly optimize all parameters of the model using the average loss $L = \text{mean}(\{L_i^s\} + \{L_j^t\})$.

Table 1: Cross-lingual stance detection model.

We make complete use of the stance labels in the cross-lingual setting by parameterizing our model according to the distance between the source and the target samples in the embedding space.

For stance equal alignment (SEA) constraint, the objective is to minimize the distance between pairs of source and target data with the same stance labels. We achieve this using the following loss:

$$L_{SEA} = \begin{cases} \mathtt{D}\big(\mathtt{g}(d_i^s, c_i^s) - \mathtt{g}(d_i^t, c_i^t)\big)^2, & y_i^s = y_i^t \\ 0, & \text{otherwise,} \end{cases} \quad (1)$$

where $\mathtt{g}$ maps its input pair to an embedding space using our memory network or any mono-lingual model, and $\mathtt{D}$ computes Euclidean distance.

For stance separation alignment (SSA), the goal is to maximize the distance between pairs with different stance labels. We use the following loss:

$$L_{SSA} = \begin{cases} \max\Big(0, m - \\ \qquad \mathtt{D}\big(\mathtt{g}(d_i^s, c_i^s) - \mathtt{g}(d_i^t, c_i^t)\big)^2\Big), & y_i^s \neq y_i^t \\ 0, & \text{otherwise,} \end{cases} \quad (2)$$

where we maximize the distance between pairs with different stance labels up to the margin $m$.

The margin parameter $m$ specifies the extent of separability in the embedding space.

We can further use any classification loss to enforce classification alignment (CA). We use categorical cross-entropy and call it *Classification Alignment* loss $L_{CA}$.

We develop our overall language adaptation loss, named *Contrastive Stance Alignment* loss, $L_{CSA}$, by combining $L_{SEA}$ and $L_{SSA}$ as follows:

$$L_{CSA} = L_{SEA} + L_{SSA}. \quad (3)$$

Finally, the total loss of our cross-lingual stance detection model is defined as follows:

$$L = (1 - \alpha)L_{CA} + \alpha L_{CSA}, \quad (4)$$

where the $\alpha$ parameter controls the balance between classification and language adaptation losses, which we optimize on the validation dataset.

**Information Flow:** Our overall cross-lingual model for stance detection is shown in Figure 1, and a summary of the algorithm is presented in Table 1. As Figure 1 shows, each source and target pairs are passed to the source and to the target memory networks to obtain their corresponding representations (Lines 3-4 in Table 1). The source representation and its gold stance label are passed to the classifier to compute the classification loss (Line 5). In addition, the source and the target representations in conjunction with a binary parameter ($y'$, which is 1 if the source and the target have the same stance label, and 0 otherwise) are passed to the language adaptation component to compute the contrastive stance alignment loss $L_{CSA}$ (Line 6). Finally, the total loss is computed based on Equation (4) (Line 7).

The classifier also uses labeled target samples to create a shared embedding space and to fine-tune itself with respect to the target language. For this purpose, we repeat the above steps by switching the target and the source pipelines (Line 8). Finally, we compute the average of all losses and we use it to optimize the parameters of our model (Line 9).

**Pre-training for Language Adaptation:** Pre-training has been found effective in many language adaptation settings (Tzeng et al., 2017). To investigate the effect of pre-training, we first pre-train the source memory network and the classifier using $\mathcal{D}_s$ (only the top pipeline in Figure 1), and then we apply language adaptation with the full model.

## 3 Experiments

**Data and Settings.** As source data, we use the Fake News Challenge dataset[1] which contains 75.4K claim-document pairs in English with {*agree*, *disagree*, *discuss*, *unrelated*} as stance labels. As target dataset, we use 3K Arabic claim-document pairs developed in (Baly et al., 2018).[2]

We perform 5-fold cross-validation on the Arabic dataset, using each fold in turn for testing, and keeping 80% of the remaining data for training and 20% for development. We use 300-dimensional pretrained cross-lingual Wikipedia word embeddings from MUSE (Lample et al., 2018). We use 300-dimensional units for the LSTM and 100 feature maps with filter width of 5 for the CNN. We consider the first 9 paragraphs per document, which is the median number of paragraphs in source documents. We optimize all hyper-parameters on validation data using *Adam* (Kingma and Ba, 2014).

**Evaluation Measures.** We use the followings:

- *Accuracy*: The fraction of correctly classified examples. For multi-class classification, accuracy is equivalent to micro-average $F_1$ (Manning et al., 2008).
- *Macro-$F_1$*: The average of $F_1$ scores that were calculated for each class separately.
- *Weighted Accuracy*: This is a hierarchical metric, which first awards 0.25 points if the model correctly predicts a document-claim pair as *related*[3] or *unrelated*. If it is *related*, 0.75 additional points are assigned if the model correctly predicts the pair as *agree*, *disagree*, or *discuss*. The goal of this weighting schema is to balance out the large number of *unrelated* examples (Hanselowski et al., 2018).

**Baselines.** We consider the following baselines:

- **Heuristic**: Given the imbalanced nature of our data, we use two heuristic baselines where all test examples are labeled as *unrelated* or *agree*. The former is a majority class baseline favoring accuracy and macro-$F_1$, while the latter is better for weighted accuracy.
- **Gradient Boosting** (Baly et al., 2018): This is a Gradient Boosting classifier with $n$-gram features as well as indicators for refutation and polarity.

[1] Available at www.fakenewschallenge.org

[2] Available at http://groups.csail.mit.edu/sls/downloads/

[3] *related* = {*agree*, *disagree*, *discuss*}

| Methods | Weigh. Acc. | Acc. | Macro-F1 | F1 (*agree, disagree, discuss, unrelated*) |
|---|---|---|---|---|
| 1. All-*unrelated* | 34.8 | 68.1 | 20.3 | 0 / 0 / 0 / 81.0 |
| 2. All-*agree* | 40.2 | 15.6 | 6.7 | 27.0 / 0 / 0 / 0 |
| 3. Gradient Boosting (Baly et al., 2018) | 55.6 | **72.4** | 41.0 | 60.4 / 9.0 / 10.4 / 84.0 |
| 4. TFMLP (Riedel et al., 2017) | 49.3 | 66.0 | 37.1 | 47.0 / 7.8 / 13.4 / 80.0 |
| 5. EnrichedMLP (Hanselowski et al., 2018) | 55.1 | 70.5 | 41.3 | 59.1 / 9.2 / 14.1 / 82.3 |
| 6. Ensemble (Baird et al., 2017) | 53.6 | 71.6 | 37.2 | 57.5 / 2.1 / 6.2 / 83.2 |
| 7. MN $target \rightarrow target$ (Mohtarami et al., 2018) | 55.3 | 70.9 | 41.7 | 60.0 / 15.0 / 08.5 / 83.1 |
| 8. MN $source \rightarrow target$ | 53.2 | 64.2 | 36.0 | 40.4 / 02.0 / 19.1 / 82.4 |
| 9. MN $(source, target) \rightarrow target$ | 57.3 | 65.0 | 42.5 | 58.0 / 12.2 / 20.9 / 79.0 |
| 10. ADMN *(adversarial)* | 58.6 | 58.4 | 43.4 | 60.2 / 16.1 / 24.2 / 72.9 |
| 11. CLMN *(contrastive)* | **61.3** | 71.6 | **45.2** | 65.1 / 11.6 / 20.5 / 83.7 |

Table 2: Evaluation results on the target Arabic test dataset.

- **TFMLP** (Riedel et al., 2017): This is an MLP with normalized bag-of-words features enriched with a single TF.IDF-based similarity feature for each claim-document pair.
- **EnrichedMLP** (Hanselowski et al., 2018): This model combines five MLPs, each with six hidden layers and advanced features from topic models, latent semantic analysis, etc.
- **Ensemble** (Baird et al., 2017): It is an ensemble based on weighted average of a deep convolutional neural network and a gradient-boosted decision tree - the best model at the Fake News Challenge.
- **Mono-lingual Memory Network** (Mohtarami et al., 2018): This model is the current state-of-the-art for stance detection on our source dataset. It is an end-to-end memory network which incorporates both CNN and LSTM for prediction.
- **Adversarial Memory Network**: We use adversarial domain adaptation (Ganin et al., 2016) instead of contrastive language adaptation in our cross-lingual memory network.

**Results.** Table 2 shows the performance of all models on the target *Arabic* test set. The All-*unrelated* and All-*agree* baselines perform poorly across evaluation measures; All-*unrelated* performs better than All-*agree* because *unrelated* is the dominant class ($\sim 68\%$ of examples).

Rows 3–6 show that *Gradient Boosting* and *EnrichedMLP* yield similar results, while *TFMLP* performs the worst. We attribute this to the advanced features used in the two former models. *Gradient Boosting* has better accuracy due to its better performance on the dominant class. Note that *Ensemble* performs poorly because of the limited labeled data, which is insufficient to train a good CNN model.

Rows 7–9 show the results for the mono-lingual memory network (MN) from (Mohtarami et al., 2018). The performance of this model when trained on Arabic data only (row 7) is comparable to previous baselines (rows 3–6). But, it shows poor performance if trained on source English data and tested on Arabic test data (row 8). The model performs best (in terms of weighted accuracy and F1) if first pretrained on source data and then fine-tuned on target training data (row 9).

Row 10 in Table 2 shows the results for adversarial memory network (ADMN). It improves the performance of mono-lingual MN on weighted accuracy and F1, but its accuracy significantly drops. This is because adversarial approaches give higher weights to samples of the majority class (i.e., *unrelated*) which makes classification more challenging for the discriminator (Montahaei et al., 2018).

Row 11 shows the results for our cross-lingual memory network (CLMN) with ($\alpha = .7$); $\alpha$ controls the balance between classification and language adaptation losses (tuned using validation data). CLMN outperforms other baselines in terms of weighted accuracy and F1 while showing comparable accuracy. We show that the improvement is due to language adaptation being able to effectively transfer knowledge from the source to target language (see Section 4.2).

The last column in Table 2 shows that *unrelated* examples are the easiest ones. Also, although the *agree* and the *discuss* classes have roughly the same size, i.e., 474 and 409 examples, respectively, the results for *agree* are notably higher. This is mainly because the documents that discuss a claim often share the same topic with the claim, but they do not take a stance. In addition, the *disagree* examples are the most difficult ones; this class is by far the smallest one, with only 87 examples.

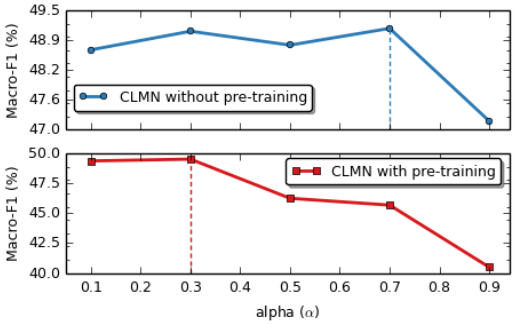


Figure 3: Impact of pretraining on the macro-F1 across $\alpha$ values. The Y axes show average results on the validation datasets with 5-fold cross-validation.

| Methods | Weigh. Acc. | Acc. | Macro-F1 |
|---|---|---|---|
| CLMN *(with pretraining)* | 60.2 | 69.8 | 43.2 |
| CLMN *(without pretraining)* | **61.3** | **71.6** | **45.2** |

Table 3: CLMN results on the target test dataset.

## 4 Discussion

### 4.1 Effect of Pretraining

Table 3 shows CLMN without pretraining ($\alpha = .7$) performs better on target test data than CLMN with pretraining ($\alpha = .3$), recall that $\alpha$ controls the balance between classification and language adaptation losses. Our further analysis shows that pretraining biases the model toward the source language. Figure 3 shows the impact of pretraining on macro-average F1 score for CLMN across different values of $\alpha$ on validation data. While the model without pretraining achieves its best performance with a large $\alpha$ ($\alpha = 0.7$), the model with pretraining performs well with a smaller $\alpha$ ($\alpha = 0.3$). This suggests that our model can capture the characteristics of the source dataset via pretraining when using small supervision from language adaptation (i.e., small $\alpha$). However, pretraining introduces bias to the source space and the performance drops when larger weights are given to language adaptation; see the results with pretraining in Figure 3.

### 4.2 Assessment of Model Transferability

The improvements of CLMN model over the monolingual MN models that use the target only, the source only, or both the target and the source (rows 7–9 in Table 2 respectively) indicate its transferability. We further estimate transferability by measuring the accuracy of the models in extracting evidence that support their predictions. A more accurate model should better transfer knowledge to the target language by accurately learning the relations between claims and pieces of evidence.

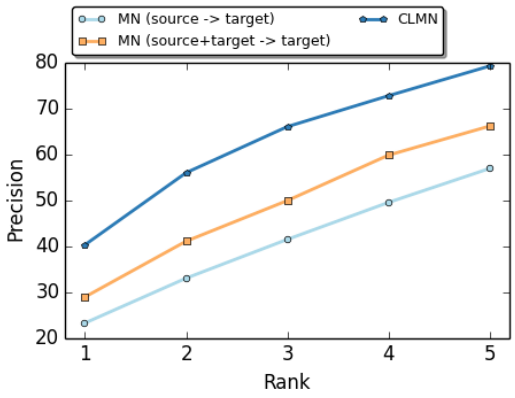


Figure 4: Transferability of our cross-lingual model.

Our target data has annotations (in terms of binary labels) for each piece of evidence (here paragraph) that indicate whether it is a rationale for the *agree* or for the *disagree* class. Moreover, our inference component ($I$) has a claim-evidence similarity vector, $P_j^{cnn}$, which can be used to rank pieces of evidence from the target document against the target claim.

We use the gold data and the rankings produced by our model in order to measure its precision in extracting evidence that supports its predictions. Figure 4 shows that our CLMN model achieves precision of 40.2, 55.9, 66.0, 72.7, and 79.2 at ranks 1–5 respectively, and outperforms mono-lingual MN models. This indicates that CLMN can better generalize and transfer knowledge to the target language through learning relations between pieces of evidence and claims.

### 4.3 Effect of Language Adaptation

Figures 5a and 5b show the classification ($L_{CA}$) versus contrastive stance ($L_{CSA}$) losses obtained from our best language adaptation model (i.e., without pretraining) across training epochs and $\alpha$ values. The results are averaged on validation data when performing 5-fold cross-validation. As Figure 5a shows, there is greater reduction in the classification loss for smaller values of $\alpha$, i.e., when classification loss contributes more to the overall loss; see Equation (4). On the other hand, Figure 5b shows that the CSA loss decreases with larger values of $\alpha$ as the model pays more attention to the CSA loss; see the red and green lines in Figure 5b. These results indicate that our language adaptation model can find a good balance between the classification loss and the CSA loss, with the value of $\alpha = .7$ yielding the best performance.

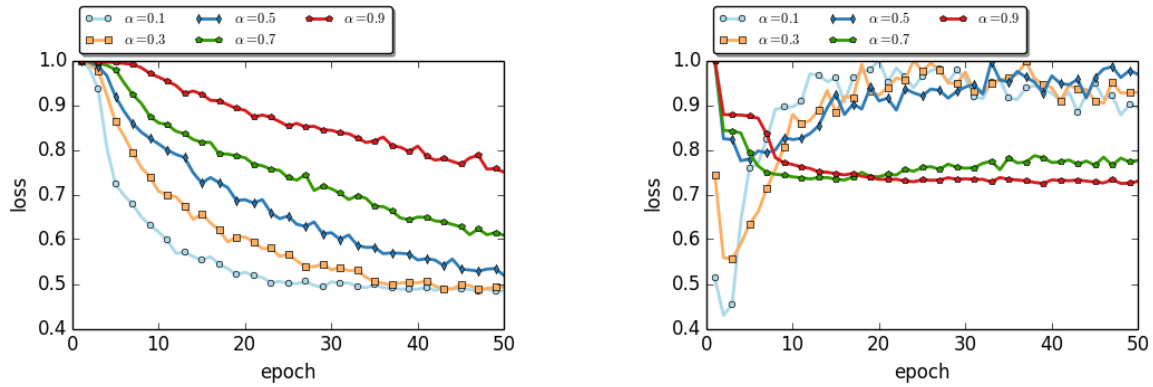


(a) Classification loss

(b) CSA loss

Figure 5: Classification vs. contrastive stance alignment (CSA) losses across training epochs and $\alpha$ values.

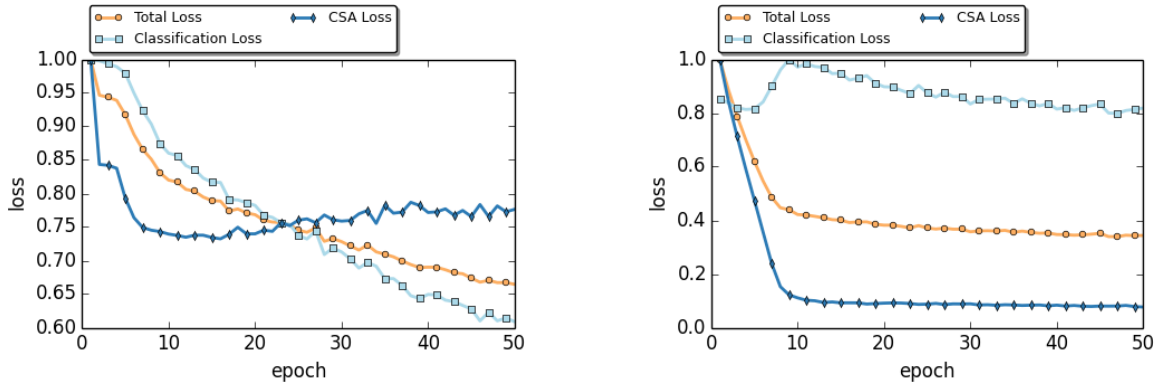


(a) without pretraining

(b) with pretraining

Figure 6: Classification loss vs. contrastive stance alignment loss (CSA) vs. total loss during training.

Figure 6 compares the classification $L_{CA}$, contrastive stance $L_{CSA}$, and total $L$ losses obtained by our CLMN model on the validation dataset across training epochs when the loss weight parameter ($\alpha$) is set to its best value. Figures 6a and 6b show the results without and with pretraining for $\alpha = .7$ and $\alpha = .3$ respectively. Without pretraining (Figure 6a), the classification (light-blue line) and CSA (dark-blue line) losses both decrease up to epoch 20, after which the classification loss keeps decreasing, but the CSA loss starts increasing. With pretraining (Figure 6b), the CSA loss rapidly decreases for the first 10 epochs (even though it has a small effect as $\alpha = .3$), and then continues with a smooth trend. This is because, during the initial training epochs, the model is biased to the *source* embedding space due to pretraining, and therefore the source and the target examples are far from each other. Then, our language adaptation model aligns the source and the target examples to form a much better shared embedding space and this alignment strategy yields a rapid decrease of the CSA loss in the first few epochs. Yet, in contrast to the CSA loss, the classification loss increases in the first few epochs. This is because the model enforces alignment between the source and the target samples due to the large distances. Finally, the total loss (orange line) indicates a good balance between the classification and the language adaptation losses, and it consistently decreases during training.

## 5 Related Work

**Domain Adaptation.** Previous work has presented several domain adaptation techniques. Unsupervised domain adaptation approaches (Ganin and Lempitsky, 2015; Long et al., 2016; Muandet et al., 2013; Gong et al., 2012) attempt to align the distribution of features in the embedding space mapped from the source and the target domains.

A limitation of such approaches is that, even with perfect alignment, there is no guarantee that the same-label examples from different domains would map nearby in the embedding space. Supervised domain adaptation (Daumé and Marcu, 2006; Becker et al., 2013; Bergamo and Torresani, 2010) attempts to encourage same-label examples from different domains to map nearby in the embedding space. While supervised approaches perform better than unsupervised ones, recent work (Motiian et al., 2017) has demonstrated superior performance by additionally encouraging class separation, meaning that examples from different domains and with different labels should be projected as far apart as possible in the embedding space. Here, we combined both types of alignments for cross-lingual stance detection.

**Stance Detection.** This is an important component for automatic fact-checking systems and veracity inference (Nadeem et al., 2019; Zhang et al., 2019; Atanasova et al., 2019). There have been some nuances in the way researchers have defined the stance detection task. Mohammad et al. (2016) and Zarrella and Marsh (2016) worked on stances regarding target propositions, e.g., entities, concepts or events, as *in-favor*, *against*, or *neither*. Most commonly, stance detection has been defined with respect to a *claim* as *agree*, *disagree*, *discuss* or *unrelated* (Hanselowski et al., 2018; Xu et al., 2018; Fang et al., 2019). Previous work mostly developed the models with rich hand-crafted features such as words, word embeddings, and sentiment lexicons (Riedel et al., 2017; Baird et al., 2017; Hanselowski et al., 2018). More recently, Mohtarami et al. (2018) presented a mono-lingual and feature-light memory network for stance detection. In this paper, we built on this work to extend previous efforts in stance detection to a cross-lingual setting, achieving the state-of-the-art result on the target language.

## 6 Conclusion and Future Work

We proposed an effective language adaptation approach to align class labels in source and target languages for accurate cross-lingual stance detection. Moreover, we investigated the behavior of our model in details and we have shown that it offers sizable performance gains over a number of competing approaches. In future, we will extend our language adaptation model to document retrieval and check-worthy claim detection tasks.


## Acknowledgments

We thank the anonymous reviewers for their insightful comments. This research was supported in part by the Qatar Computing Research Institute, HBKU[4] and DSTA of Singapore.


## References


Pepa Atanasova, Preslav Nakov, Lluís Màrquez, Alberto Barrón-Cedeño, Georgi Karadzhov, Tsvetomila Mihaylova, Mitra Mohtarami, and James Glass. 2019. Automatic fact-checking using context and discourse information. *ACM Journal of Data and Information Quality*, 11(3):12:1–12:27.

Sean Baird, Doug Sibley, and Yuxi Pan. 2017. Talos targets disinformation with fake news challenge victory. https://blog.talosintelligence.com/2017/06/talos-fake-news-challenge.html.

Ramy Baly, Mitra Mohtarami, James Glass, Lluís Màrquez, Alessandro Moschitti, and Preslav Nakov. 2018. Integrating stance detection and fact checking in a unified corpus. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), Volume 2*, pages 21–27, New Orleans, Louisiana, USA.

Roy Bar-Haim, Indrajit Bhattacharya, Francesco Dinuzzo, Amrita Saha, and Noam Slonim. 2017. Stance classification of context-dependent claims. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '17, pages 251–261, Valencia, Spain.

Carlos J Becker, Christos M Christoudias, and Pascal Fua. 2013. Non-linear domain adaptation with boosting. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, pages 485–493. Lake Tahoe, Nevada.

Alessandro Bergamo and Lorenzo Torresani. 2010. Exploiting weakly-labeled web images to improve object classification: a domain adaptation approach. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 1*, pages 181–189. Vancouver, British Columbia, Canada.

Antoine Bordes, Y-Lan Boureau, and Jason Weston. 2017. Learning end-to-end goal-oriented dialog. In *Proceedings of the 5th International Conference on Learning Representations*, ICLR '17, Toulon, France.


[4]This research is part of the Tanbih project (available at http://tanbih.qcri.org/) which aims to limit the effect of "fake news", propaganda and media bias by making users aware of what they are reading.

Hal Daumé, III and Daniel Marcu. 2006. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26(1):101–126.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, pages 4171–4186, Minneapolis, Minnesota, USA.

Sebastian Dungs, Ahmet Aker, Norbert Fuhr, and Kalina Bontcheva. 2018. Can rumour stance alone predict veracity? In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3360–3370, Santa Fe, NM, USA.

Wei Fang, Moin Nadeem, Mitra Mohtarami, and James Glass. 2019. Neural multi-task learning for stance prediction. In *Proceedings of the EMNLP Workshop on Fact Extraction and Verification*, Hong Kong, China.

Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *Proceedings of the 32nd International Conference on Machine Learning - Volume 37*, ICML'15, pages 1180–1189.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(1):2096–2030.

Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. 2012. Geodesic flow kernel for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2066–2073.

Andreas Hanselowski, Avinesh PVS, Benjamin Schiller, Felix Caspelherr, Debanjan Chaudhuri, Christian M. Meyer, and Iryna Gurevych. 2018. A retrospective analysis of the fake news challenge stance-detection task. In *Proceedings of the International Conference on Computational Linguistics*, COLING '18, pages 1859–1874, Santa Fe, NM, USA.

Viet-Phi Huynh and Paolo Papotti. 2018. Towards a benchmark for fact checking with knowledge bases. In *Companion Proceedings of the The Web Conference 2018*, WWW '18, pages 1595–1598, Lyon, France.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '14, pages 1746–1751, Doha, Qatar.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Elena Kochkina, Maria Liakata, and Arkaitz Zubiaga. 2018. All-in-one: Multi-task learning for rumour verification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3402–3413, Santa Fe, NM, USA.

Guillaume Lample, Alexis Conneau, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. In *Proceedings of the 6th International Conference on Learning Representations*, Vancouver, BC, Canada.

David M.J. Lazer, Matthew A. Baum, Yochai Benkler, Adam J. Berinsky, Kelly M. Greenhill, Filippo Menczer, Miriam J. Metzger, Brendan Nyhan, Gordon Pennycook, David Rothschild, Michael Schudson, Steven A. Sloman, Cass R. Sunstein, Emily A. Thorson, Duncan J. Watts, and Jonathan L. Zittrain. 2018. The science of fake news. *Science*, 359(6380):1094–1096.

Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I. Jordan. 2016. Unsupervised domain adaptation with residual transfer networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 136–144, Barcelona, Spain.

Jing Ma, Wei Gao, and Kam-Fai Wong. 2017. Detect rumors in microblog posts using propagation structure via kernel learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, ACL '17, pages 708–717, Vancouver, Canada.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

Tsvetomila Mihaylova, Georgi Karadzhov, Pepa Atanasova, Ramy Baly, Mitra Mohtarami, and Preslav Nakov. 2019. SemEval-2019 task 8: Fact checking in community question answering forums. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 860–869, Minneapolis, Minnesota, USA.

Tsvetomila Mihaylova, Preslav Nakov, Lluís Màrquez, Alberto Barrón-Cedeño, Mitra Mohtarami, Georgi Karadzhov, and James Glass. 2018. Fact checking in community forums. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, New Orleans, LA, USA.

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiao-Dan Zhu, and Colin Cherry. 2016. SemEval-2016 task 6: Detecting stance in tweets. In *Proceedings of Semantic Evaluation*, SemEval '16, pages 31–41, Berlin, Germany.

Mitra Mohtarami, Ramy Baly, James Glass, Preslav Nakov, Lluís Màrquez, and Alessandro Moschitti. 2018. Automatic stance detection using end-to-end memory networks. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (Long Papers)*, pages 767–776, New Orleans, LA, USA.

Ehsan Montahaei, Mahsa Ghorbani, Mahdieh Soleymani Baghshah, and Hamid R. Rabiee. 2018. Adversarial classifier for imbalanced problems. *CoRR*, abs/1811.08812.

Saeid Motiian, Marco Piccirilli, Donald A. Adjeroh, and Gianfranco Doretto. 2017. Unified deep supervised domain adaptation and generalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5715–5725, Venice, Italy.

Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. 2013. Domain generalization via invariant feature representation. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28, pages 10–18, Atlanta, GA, USA.

Moin Nadeem, Wei Fang, Brian Xu, Mitra Mohtarami, and James Glass. 2019. FAKTA: An automatic end-to-end fact checking system. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 78–83, Minneapolis, MN, USA.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2227–2237, New Orleans, LA, USA.

Kashyap Popat, Subhabrata Mukherjee, Jannik Strötgen, and Gerhard Weikum. 2017. Where the truth lies: Explaining the credibility of emerging claims on the Web and social media. In *Proceedings of the Conference on World Wide Web*, WWW '17, pages 1003–1012, Perth, Australia.

Benjamin Riedel, Isabelle Augenstein, Georgios P Spithourakis, and Sebastian Riedel. 2017. A simple but tough-to-beat baseline for the Fake News Challenge stance detection task. *ArXiv:1707.03264*.

Parinaz Sobhani, Diana Inkpen, and Xiaodan Zhu. 2017. A dataset for multi-target stance detection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2*, pages 551–557, Valencia, Spain.

Karen Spärck Jones. 2004. Idf term weighting and ir research lessons. *Journal of documentation*, 60(5):521–523.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, pages 2440–2448, Montreal, Canada.

Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2016. Improved representation learning for question answer matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 464–473, Berlin, Germany.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT '18, pages 809–819, New Orleans, LA, USA.

Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. 2017. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '17, Honolulu, HI, USA.

Andreas Vlachos and Sebastian Riedel. 2014. Fact checking: Task definition and dataset construction. In *Proceedings of the ACL Workshop on Language Technologies and Computational Social Science*, pages 18–22, Baltimore, MD, USA.

Soroush Vosoughi, Deb Roy, and Sinan Aral. 2018. The spread of true and false news online. *Science*, 359(6380):1146–1151.

William Yang Wang. 2017. "liar, liar pants on fire": A new benchmark dataset for fake news detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, ACL '17, pages 422–426, Vancouver, Canada.

Caiming Xiong, Stephen Merity, and Richard Socher. 2016. Dynamic memory networks for visual and textual question answering. In *Proceedings of the 33rd International Conference on Machine Learning*, ICML '16, pages 2397–2406, New York, NY, USA.

Brian Xu, Mitra Mohtarami, and James Glass. 2018. Adversarial domain adaptation for stance detection. In *Proceedings of the NeurIPS Workshop on Continual Learning*, Montreal, Canada.

Guido Zarrella and Amy Marsh. 2016. MITRE at SemEval-2016 Task 6: Transfer learning for stance detection. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval '16, pages 458–463, San Diego, CA, USA.

Yifan Zhang, Giovanni Da San Martino, Alberto Barrón-Cedeño, Salvatore Romeo, Jisun An, Haewoon Kwak, Todor Staykovski, Israa Jaradat, Georgi Karadzhov, Ramy Baly, Kareem Darwish, James Glass, and Preslav Nakov. 2019. Tanbih: Get to know what you are reading. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (Demonstrations)*, EMNLP '19, Hong Kong, China.

# Jointly Learning to Align and Translate with Transformer Models

**Sarthak Garg  Stephan Peitz  Udhyakumar Nallasamy  Matthias Paulik**
Apple Inc.
{sarthak_garg, speitz, udhay, mpaulik}@apple.com

## Abstract

The state of the art in machine translation (MT) is governed by neural approaches, which typically provide superior translation accuracy over statistical approaches. However, on the closely related task of word alignment, traditional statistical word alignment models often remain the go-to solution. In this paper, we present an approach to train a Transformer model to produce both accurate translations and alignments. We extract discrete alignments from the attention probabilities learnt during regular neural machine translation model training and leverage them in a multi-task framework to optimize towards translation and alignment objectives. We demonstrate that our approach produces competitive results compared to GIZA++ trained IBM alignment models without sacrificing translation accuracy and outperforms previous attempts on Transformer model based word alignment. Finally, by incorporating IBM model alignments into our multi-task training, we report significantly better alignment accuracies compared to GIZA++ on three publicly available data sets. Our implementation has been open-sourced[1].

## 1 Introduction

Neural machine translation (NMT) constitutes the state of the art in MT, with the Transformer model architecture (Vaswani et al., 2017) beating other neural architectures in competitive MT evaluations. The attention mechanism used in NMT models was motivated by the need to model word alignments, however it is now well known that the attention probabilities can differ significantly from word alignments in the traditional sense (Koehn and Knowles, 2017), since attending to the context words rather than the aligned source words might be helpful for translation. The presence of multi-layer, multi-head attention mechanisms in the Transformer model further complicate interpreting the attention probabilities and extracting high quality discrete alignments from them.

Finding source to target word alignments has many applications in the context of MT. A straightforward application of word alignments is to generate bilingual lexica from parallel corpora. Word alignments have also been used for external dictionary assisted translation (Chatterjee et al., 2017; Alkhouli et al., 2018; Arthur et al., 2016) to improve translation of low frequency words or to comply with certain terminology guidelines. Documents and webpages often contain word annotations such as formatting styles and hyperlinks, which need to be preserved in the translation. In such cases, word alignments can be used to transfer these annotations from the source sentence to its translation. In user facing translation services, providing word alignments as additional information to the users might improve their trust and confidence, and also help them to diagnose problems such as under-translation (Tu et al., 2016).

In this work, we introduce an approach that teaches Transformer models to produce translations and interpretable alignments simultaneously:

- We use a multi-task loss function combining negative log likelihood (NLL) loss used in regular NMT model training and an alignment loss supervising one attention head to learn alignments (Section 4.2).
- Conditioning on past target context is essential for maintaining the auto-regressive property for translation but can be limiting for alignment. We alleviate this problem by conditioning the different components of our multi-task objective on different amounts of context (Section 4.3).

[1]Code can be found at https://github.com/pytorch/fairseq/pull/1095



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4453–4462, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

- We demonstrate that the system can be supervised using seed alignments obtained by carefully averaging the attention probabilities of a regular NMT model (Section 4.1) or alignments obtained from statistical alignment tools (Section 4.4)

We show that our model outperforms previous neural approaches (Peter et al., 2017; Zenkel et al., 2019) and statistical alignment models (Och and Ney, 2003) in terms of alignment accuracy without suffering any degradation of translation accuracy.

## 2 Preliminaries

### 2.1 Word Alignment Task

Given a sentence $f_1^J = f_1, \ldots, f_j, \ldots f_J$ in the source language *and* its translation $e_1^I = e_1, \ldots, e_i, \ldots e_I$ in the target language, an alignment $\mathcal{A}$ is defined as a subset of the Cartesian product of the word positions (Och and Ney, 2003).

$$\mathcal{A} \subseteq \{(j, i) : j = 1, \ldots, J; i = 1, \ldots, I\} \quad (1)$$

The word alignment task aims to find a discrete alignment representing a many-to-many mapping from the source words to their corresponding translations in the target sentence.

### 2.2 Transformer Model

The Transformer model (Vaswani et al., 2017) is an encoder-decoder model that only relies on attention for computing the contextual representations for source and target sentences. Both the encoder and decoder are composed of multiple layers, each of which includes a multi-head self-attention and a feed-forward sub-layer. Layers in the decoder additionally apply a multi-head encoder-decoder attention between the self-attention and the feed-forward sub-layers. To maintain the auto-regressive property, the self-attention sub-layer in the decoder attends to the representations of only the *past* tokens computed by the lower layer.

In this work, we will be focusing on guiding the encoder-decoder attention sub-layer in the decoder. Let $d_{emb}, d_k, d_v, N$ denote the embedding dimension, dimensions of the key and value projections and number of heads, respectively. As described in Vaswani et al. (2017), for this sub-layer, the output of the previous decoder layer corresponding to the $i^{th}$ target token is used as a query vector $\mathbf{q}^i \in \mathbb{R}^{1 \times d_{emb}}$ and the encoder output for all the source tokens are packed together as the value $V \in \mathbb{R}^{J \times d_{emb}}$ and key $K \in \mathbb{R}^{J \times d_{emb}}$ matrices. To compute the output $\mathcal{M}(\mathbf{q}^i, K, V)$, $N$ heads first project the query vector and the key and value matrices into different subspaces, compute attention in their own subspaces, aggregate their outputs and project back to the original space:

$$\tilde{\mathbf{q}}_n^i = \mathbf{q}^i W_n^Q, \tilde{K}_n = K W_n^K, \tilde{V}_n = V W_n^V \quad (2)$$

$$H_n^i = \text{Attention}(\tilde{\mathbf{q}}_n^i, \tilde{K}_n, \tilde{V}_n) \quad (3)$$

$$\mathcal{M}(\mathbf{q}^i, K, V) = \text{Concat}(H_1^i, \ldots, H_N^i) W^O, \quad (4)$$

where the projection matrices $W_n^Q, W_n^K, W_n^V$ and $W^O$ are learnt parameters of the $n^{th}$ head. Each head employs a scaled dot-product attention:

$$\text{Attention}(\tilde{\mathbf{q}}_n^i, \tilde{K}_n, \tilde{V}_n) = \mathbf{a}_n^i \tilde{V}_n, \quad (5)$$

$$\text{where } \mathbf{a}_n^i = \text{softmax}(\frac{\tilde{\mathbf{q}}_n^i \tilde{K}_n^T}{\sqrt{d_k}}). \quad (6)$$

The vector $\mathbf{a}_n^i \in \mathbb{R}^{1 \times J}$ denotes the attention probabilities for the $i^{th}$ target token over all the source tokens, computed by the $n^{th}$ attention head. For any particular head, an attention matrix $A_{I \times J}$ can be constructed by grouping together the vectors $\mathbf{a}_n^i$ corresponding to all the target tokens. In the following sections, we analyze the quality of alignments that can be extracted from these attention matrices $A_{I \times J}$ and describe how they can be effectively supervised to learn word alignments.

## 3 Baseline Methods

A common baseline approach to extract word alignments from a regular NMT trained Transformer, is to average over all attention matrices $A_{I \times J}$ computed across all layers and heads. The resulting matrix gives a probability distribution over all source tokens for each target token. This distribution is then converted to a discrete alignment by aligning each target word to the corresponding source word with the highest attention probability.

Peter et al. (2017) guide the attention probabilities to be close to the alignments obtained from statistical MT toolkits by imposing an additional loss based on the distance between the alignment and attention distributions. They get improvements in alignment accuracy over previous works based on guided alignment training by feeding the *current* target word to the attention module, providing it more context about the target sentence.

Zenkel et al. (2019) proposed an method that does not rely on alignments from external toolkits for training. They instead add an extra attention layer on top of the Transformer architecture and directly optimize its activations towards predicting the given target word.

All the above methods involve training models for both the directions to get bidirectional alignments. These bidirectional alignments are then merged using the `grow diagonal` heuristic (Koehn et al., 2005).

## 4 Proposed Method

### 4.1 Averaging Layer-wise Attention Scores

The attention heads in a single layer are symmetrical, but the different layers themselves can learn drastically different alignments. To better understand the behavior of the encoder-decoder attention learnt at different layers, we average the attention matrices computed across all heads within each layer and evaluate the obtained alignments. We show that the attention probabilities from the penultimate layer naturally tend to learn alignments and provide significantly better results compared to naively averaging across all layers (cf. Section 5.3). For the rest of the paper, we refer to the former method as the layer average baseline.

### 4.2 Multi-task Learning

Translation and alignment tasks are very closely related. NMT models with attention (Bahdanau et al., 2015) have also shown to learn alignments in the intermediate attention layer. A neural model receiving supervision from given translations *and* given alignments can therefore benefit from multi-task learning by exploiting the correlations between these two tasks.

Annotating word alignments is a laborious and expensive task, but the layer average baseline described in Section 4.1 is able to generate reasonably good alignments in an unsupervised manner. We thus use the alignments generated by the layer average baseline as labels for supervising our model. We first convert the alignments into a probability distribution over source tokens for every target token. Let $G_{I\times J}$ denote a 0-1 matrix such that $G_{i,j} = 1$ if the $j^{th}$ source token is aligned to the $i^{th}$ target token. We simply normalize the rows of matrix $G$ corresponding to target tokens that are aligned to at least one source token to get a matrix $G^p$. As described in Section 2.2, the Transformer model computes multiple attention probability distributions over source tokens for every target token across different heads and layers of the network. Since we observed that the attention probabilities from the penultimate layer most naturally tend to learn alignments (Section 5.3), we arbitrarily select one head from the penultimate layer (subsequently referred to as the alignment head) and supervise its attention probability distribution to be close to the labeled alignment distribution ($G^p$). Let $A_{I\times J}$ denote the attention matrix computed by the alignment head. For every target word $i$, we minimize the Kullback-Leibler divergence between $G^p_i$ and $A_i$ which is equivalent to optimizing the following cross-entropy loss $\mathcal{L}_a$

$$\mathcal{L}_a(A) = -\frac{1}{I}\sum_{i=1}^{I}\sum_{j=1}^{J} G^p_{i,j}\log(A_{i,j}). \quad (7)$$

The motivation behind supervising one head is that it gives the model the flexibility to either use the representation computed by the alignment head, or depend more on the representations computed by other heads. We train our model to minimize $\mathcal{L}_a$ in conjunction with the standard NLL translation loss $\mathcal{L}_t$. The overall loss $\mathcal{L}$ is:

$$\mathcal{L} = \mathcal{L}_t + \lambda\mathcal{L}_a(A), \quad (8)$$

where $\lambda$ is a hyperparameter.

### 4.3 Providing Full Target Context

The Transformer decoder computes the probability of the next target token conditioned on the past target tokens and all source tokens. This is implemented by masking the self attention probabilities, i.e. while computing the representation for the $i^{th}$ target token, the decoder can only self-attend to the representations of $\{1, 2 \ldots i-1\}$ tokens from the previous layer. This auto-regressive behavior of the decoder is crucial for the model to represent a valid probability distribution over the target sentence. However, conditioning on just the past target tokens is limiting for the alignment task. As described in Section 4.2, the alignment head is trained to model the alignment distribution for the $i^{th}$ target token given only the past target tokens and all source tokens. Since the alignment head does not know the identity of the next target token, it becomes difficult for it to learn this token's alignment to the source tokens. Previous work has also identified this problem and alleviate

it by feeding the target token to be aligned as an input to the module computing the alignment (Peter et al., 2017), or forcing the module to predict the target token (Zenkel et al., 2019) or its properties, e.g. POS tags (Li et al., 2018). Feeding the next target token assumes that we know it in advance and thus calls for separate translation and alignment models. Forcing the alignment module to predict target token's properties helps but still passes the information of the target token in an indirect manner. We overcome these limitations by conditioning the two components of our loss function on different amounts of context. The NLL loss $\mathcal{L}_t$ is conditioned on the past target tokens to preserve the auto-regressive property:

$$\mathcal{L}_t = -\frac{1}{I}\sum_{i=1}^{I}\log(p(e_i|f_1^J, e_1^{i-1})). \quad (9)$$

However, the alignment loss $\mathcal{L}_a^{'}$ is now conditioned on the whole target sentence:

$$\mathcal{L}_a^{'} = \mathcal{L}_a(A|f_1^J, e_1^I). \quad (10)$$

This is implemented by executing two forward passes of the decoder model, one with the masking of the future target tokens for computing the NLL loss $\mathcal{L}_t$ and the other one with no masking for computing the alignment loss $\mathcal{L}_a^{'}$from the alignment head. Although this formulation forces the network to learn representations adapting to both full and partial target context, Section 5.5 shows that this approach does not degrade the translation quality while improving the alignment accuracy.

### 4.4 Alignment Training Data

Our method described so far does not rely on alignments from external statistical toolkits but performs self-training on alignments extracted from the layer average baseline. However, GIZA++ provides a robust method to compute accurate alignments. If achieving better alignment accuracy is paramount, then our multi-task framework can also leverage alignments from GIZA++ to produce even better alignment accuracy (Section 5.4). In this setting we use the GIZA++ alignments as labels instead of those obtained from the layer average baseline for supervising the alignment head.

## 5 Experiments

### 5.1 Setup

Our experiments show that our proposed approach is able to achieve state-of-the-art results in terms of alignment *and* maintain the same translation performance. In the following, we describe two setups to compare with previously established state-of-the-art results.

For all setups and models used in this work, we learn a joint source and target Byte-Pair-Encoding (BPE, Sennrich et al. (2016)) with 32k merge operations. We observe that even for statistical alignment models sub-word units are beneficial. To convert the alignments from sub-word-level back to word-level, we consider each target word as being aligned to a source word if an alignment between any of the target sub-words and source sub-words exists.

The alignment quality is evaluated by using the alignment error rate (AER) introduced in (Och and Ney, 2000). Significance of the differences in AER between two models is tested using a two-sided Wilcoxon signed-rank test ($\alpha = 0.1\%$).

#### 5.1.1 Alignment Task

The purpose of the this task is to fairly compare with state-of-the-art results in terms of alignment quality and perform a hyperparameter search. We use the same experimental setup as described in (Zenkel et al., 2019). The authors provide pre-processing and scoring scripts[2] for three different datasets: Romanian→English, English→French and German→English. Training data and test data for Romanian→English and English→French are provided by the NAACL'03 *Building and Using Parallel Texts* word alignment shared task[3] (Mihalcea and Pedersen, 2003). The Romanian→English training data are augmented by the Europarl v8 corpus increasing the amount of parallel sentences from 49k to 0.4M. For German→English we use the Europarl v7 corpus as training data and the gold alignments[4] provided by Vilar et al. (2006). The reference alignments were created by randomly selecting a subset of the Europarl v7 corpus and manually annotating them following the guidelines suggested in (Och

[2] https://github.com/lilt/alignment-scripts

[3] http://web.eecs.umich.edu/~mihalcea/wpt/index.html#resources

[4] https://www-i6.informatik.rwth-aachen.de/goldAlignment/

and Ney, 2003). Data statistics are shown in Table 1.

Table 1: Number of sentences for three datasets: German→English (DeEn), Romanian→English (RoEn) and English→French (EnFr). The datasets include training data and test data with gold alignments.

| | DeEn | RoEn | EnFr |
|---|---|---|---|
| training | 1.9M | 0.5k | 1.1M |
| test | 508 | 248 | 447 |

In all experiments for this task, we employ the `base` transformer configuration with an embedding size of 512, 6 encoder and decoder layers, 8 attention heads, shared input and output embeddings (Press and Wolf, 2017), the standard `relu` activation function and sinusoidal positional embedding. The total number of parameters is 60M. We train with a batch size of 2000 tokens on 8 Volta GPUs and use the validation translation loss for early stopping. Furthermore, we use Adam optimizer (Loshchilov and Hutter, 2019) with a learning rate of 3e-4, $\beta_1 = 0.9, \beta_2 = 0.98$, learning rate warmup over the first 4000 steps and inverse square root as learning rate scheduler. The dropout probability is set to 0.1. Additionally, we apply label smoothing with a factor of 0.1. To conveniently extract word alignments for both translation directions, we train bidirectional models, i.e. our models are able to translate and align from Romanian to English and vice versa.

#### 5.1.2 Align and Translate Task

The second setup is based on the WMT'18 English-German news translation task (Bojar et al., 2018). We apply the same corpus selection for bilingual data and model architecture as suggested by Edunov et al. (2018). However, we slightly modify the preprocessing pipeline to be able to evaluate the alignment quality against the gold alignments provided by Vilar et al. (2006). We use all available bilingual data (Europarl v7, Common Crawl corpus, News Commentary v13 and Rapid corpus of EU press releases) excluding the ParalCrawl corpus. We remove sentences longer than 100 words and sentence pairs with a source/target length ratio exceeding 1.5. This results in 5.2M parallel sentences. We apply the Moses tokenizer (Koehn et al., 2007) without aggressive hyphen splitting and without performing HTML escaping of apostrophes and quotes. Furthermore, we do not normalize punctuation marks. We use newstest2012 as validation and newstest2014 as test set.

To achieve state-of-the-art translation results, all models in this setup are trained unidirectional and we change to the `big` transformer configuration with an embedding size of 1024 and 16 attention heads. The total number of parameters is 213M. We train the layer average baseline with a batch size of 7168 tokens on 64 Volta GPUs for 30k updates and apply a learning rate of 1e-3, $\beta_1 = 0.9, \beta_2 = 0.98$. The dropout probability is set to 0.3. All other hyperparameters are as described in the previous section. Since training the multi-task models consumes more memory, we need to half the batch size, increase the number of updates accordingly and adapt the learning rate to 7e-4. We average over the last 10 checkpoints and run inference with a beam size of 5.

To fairly compare against state-of-the-art translation setups, we compute BLEU (Papineni et al., 2002) with `sacreBLEU` (Post, 2018).

### 5.2 Statistical Baseline

For both setups, the statistical alignment models are computed with the multi-threaded version of the GIZA++ toolkit[5] implemented by Gao and Vogel (2008). GIZA++ estimates IBM1-5 models and a first-order hidden Markov model (HMM) as introduced in (Brown et al., 1993) and (Vogel et al., 1996), respectively. In particular, we perform 5 iterations of IBM1, HMM, IBM3 and IBM4. Furthermore, the alignment models are trained in both translation directions and symmetrized by employing the `grow-diagonal` heuristic (Koehn et al., 2005). We use the resulting word alignments to supervise the alignment loss for the method described in Section 4.4.

### 5.3 Averaging Attention Results

For our experiments, we use the data and Transformer model setup described in Section 5.1.1. We perform the evaluation of alignments obtained by layer wise averaging of attention probabilities as described in Section 4.1. As shown in Table 2, all three language pairs exhibit a very similar pattern, wherein the attentions do not seem to learn meaningful alignments in the initial layers and show a remarkable improvement in the higher layers. This indicates that the initial layers are fo-

[5] https://github.com/moses-smt/mgiza/

Table 2: AER [%] per layer for all three language pairs: German→English (DeEn), Romanian→English (RoEn) and English→French (EnFr).

| Layer | DeEn | RoEn | EnFr |
|---|---|---|---|
| 1 (bottom) | 90.0 | 92.9 | 80.7 |
| 2 | 91.0 | 93.6 | 81.7 |
| 3 | 94.5 | 92.0 | 73.4 |
| 4 | 41.2 | 37.5 | 20.5 |
| 5 | **32.6** | **33.4** | **17.0** |
| 6 (top) | 56.3 | 48.4 | 37.9 |
| average | 55.8 | 38.6 | 23.2 |

cusing more on learning good representations of the sentence generated by the decoder so far by self attention. Once good contextual representations are learnt, the higher layers fetch the relevant representations from the encoder's output via the encoder-decoder attention. However, interestingly the penultimate layer outperforms the final layer suggesting the final layer uses the alignment-based features in the penultimate layer to derive its own representation.

### 5.4 Alignment Task Results

Table 3 compares the performance of our methods against statistical baselines and previous neural approaches. The layer average baseline provides relatively weak alignments, which are used for training our *multi-task* model. The improvement of the multi-task approach over the layer average baseline suggests that learning to translate helps produce better alignments as well. However still the multi-task approach falls short of the statistical and neural baselines, which have a strong advantage of having access to the full/partial target context. Exposing our model to the full target context gives the largest gains in terms of AER. Note that *full context* results are directly comparable to Zenkel et al. (2019) since both approaches do not leverage external knowledge from statistical models. We suspect that we are able to outperform Zenkel et al. (2019) because we provide the full target context instead of only the to-be aligned target word. Finally, by supervising our model on the alignments obtained from GIZA++ (GIZA++ *supervised*) rather than layer average baseline, we outperform GIZA++ and Peter et al. (2017).

We tuned the alignment loss weight $\lambda$ (Equation 8) using grid search on the German→English dataset. We achieve the best results with $\lambda = 0.05$.

Table 3: Results on the alignment task (in AER [%]). ‡Difference in AER w.r.t. GIZA++ (BPE-based) is statistically significant ($p<0.001$).

| Model | DeEn | RoEn | EnFr |
|---|---|---|---|
| GIZA++ (word-based) | 21.4 | 27.9 | 5.9 |
| GIZA++ (BPE-based) | 18.9 | 27.0 | 5.5 |
| Layer average baseline | 32.6 | 33.4 | 17.0 |
| Multi-task | 25.4 | 30.7 | 12.6 |
| + full-context | 20.2 | 26.0 | 7.7 |
| ++ GIZA++ supervised | **16.0**‡ | **23.1**‡ | **4.6**‡ |
| Peter et al. (2017) | 19.0 | - | - |
| Zenkel et al. (2019) | 21.2 | 27.6 | 10.0 |

### 5.5 Align and Translate Task Results

For fair comparison of our approach to the state-of-the-art translation models, we use the setup described in Section 5.1.2. Table 4 summarizes the results on alignment and translation tasks. The layer average baseline is based on regular NMT model training, therefore ideally it should achieve the same BLEU as Edunov et al. (2018), however we see a small drop of 0.3 BLEU points in practice which could be caused by the slightly different preprocessing procedure (cf. Section 5.1.2, no aggressive hyphen splitting/no punctuation normalization). The layer average baseline performs poorly in terms of the AER. The Precision and Recall results for the layer average baseline demonstrate the effectiveness of symmetrization. Symmetrization removes a majority of incorrect alignments and gives a high precision (94.2%) but low recall (29.6%). The high precision of the layer average baseline ensures that the multi-task model receives correct alignments for supervision, enabling it to get large improvements in AER over the layer average baseline.

Similar to the trend observed in 5.4, providing full target sentence context in the decoder helps the model to improve further and perform comparably to GIZA++. Lastly, supervision with GIZA++ gives the best AER and significantly outperforms GIZA++. The improvements in alignment quality and no degradation in BLEU compared to the layer average baseline shows the effectiveness of the proposed multi-task approach.

Table 4: Results on the align and translate task. Alignment quality is reported in AER, translation quality in BLEU. †baseline (without back-translation) `sacreBLEU` results were provided in https://github.com/pytorch/fairseq/issues/506#issuecomment-464411433. ‡Difference in AER w.r.t. GIZA++ (BPE-based) is statistically significant ($p<0.001$)

| Model | AER[%] (Precision[%], Recall[%]) DeEn | EnDe | Symmetrized | BLEU[%] DeEn | EnDe |
|---|---|---|---|---|---|
| GIZA++ (word-based) | 21.7 (85.4, 72.1) | 24.0 (85.8, 68.2) | 22.2 (93.5, 66.5) | - | - |
| GIZA++ (BPE-based) | 19.0 (89.1, 74.2) | 21.3 (86.8, 71.9) | 19.6 (93.2, 70.6) | - | - |
| Layer average baseline | 66.8 (32.0, 34.6) | 66.5 (32.5, 34.7) | 54.8 (94.2, 29.6) | 33.1 | 28.7 |
| Multi-task | 31.1 (67.2, 70.7) | 32.2 (66.6, 69.1) | 25.8 (88.1, 63.8) | 33.1 | 28.5 |
| + full-context | 21.2 (76.9, 80.9) | 23.5 (75.0, 78.0) | 19.5 (89.5, 72.9) | 33.2 | 28.5 |
| ++ GIZA++ supervised | **17.5**‡ (80.5, 84.7) | **19.8**‡ (78.8, 81.7) | **16.4**‡ (89.6, 78.2) | 33.1 | 28.8 |
| Edunov et al. (2018)† | - | - | - | - | 29.0 |

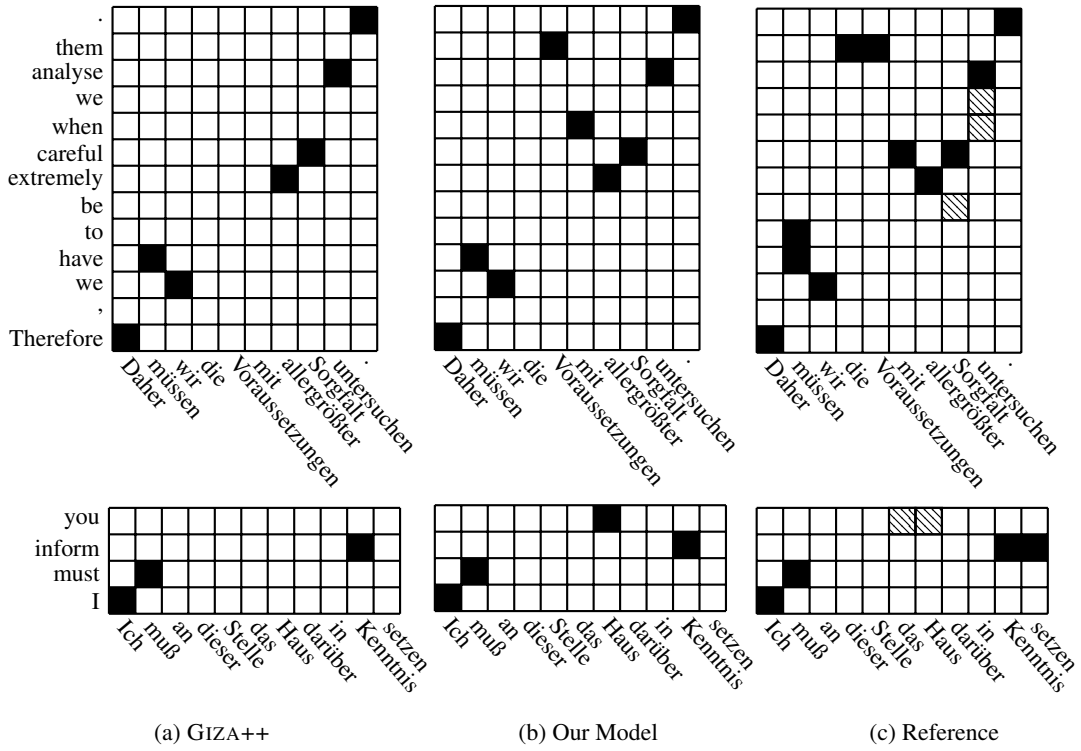


(a) GIZA++ (b) Our Model (c) Reference

Figure 1: Two examples from the German→English alignment test set. Alignments in (a) show the output from GIZA++ and (b) from our model (*Multi-task* with full-context and GIZA++ supervised). Gold Alignments in shown in (c). Black squares and hatched squares in the reference represent *sure* and *possible* alignments, respectively.

## 6 Analysis

To further investigate why our proposed approach is superior to GIZA++ in terms of AER, we analyze the generated word alignments of both models. We observe that our model tends to align pronouns (e.g. *you* or *them*) with regular nouns (e.g. objects or subjects). Given the gold alignments, it seems that these alignment links are correct or at least possible (Och and Ney (2003) provided annotators two options to specify alignments: *sure* and *possible* for unambiguous and ambiguous alignments respectively). Figure 1 shows two examples from the German→English alignment test set. In the first example, our model correctly aligns *them* with *Voraussetzungen (criteria)*. The German parliament speaker indeed mentioned *Verfahrensvoraussetzungen (procedural criteria)* in one of the preceding sentences and refers later to them by using the term *Voraussetzungen (criteria)*. In the second example, the pronoun *you* is correctly aligned to the noun *Haus*

*(house)* which is just another way to address the audience in the European parliament. Both alignment links are not generated by GIZA++. This could be related to fact that a statistical model is based on counting co-occurrences. We speculate that to generate such alignment links, a model needs to be able to encode contextual information. Experimental results in (Tang et al., 2018) suggest that NMT models learn to encode contextual information, which seems to be necessary for word sense disambiguation. Since pronouns can be ambiguous references, we assume that both problems are closely related and therefore believe that the ability to encode contextual information may be beneficial for generating word alignments.

From our experiments on the WMT'18 dataset, we observe that the alignment quality of the layer average baseline is quite low (cf. Table 4). To further investigate this, we plot the test AER and the validation NLL loss per epoch (Figure 2). The graph shows that the lowest AER of 42.7% is already reached in the fifth epoch. This suggests that

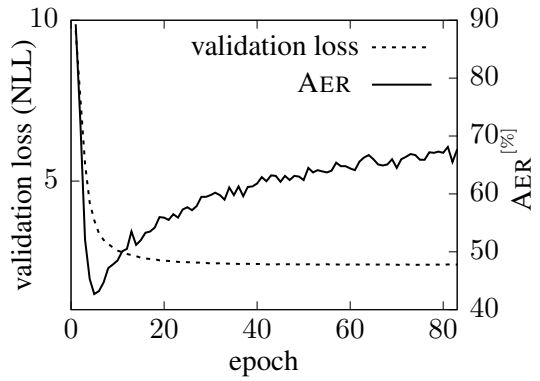


Figure 2: Test AER and validation loss (NLL) per epoch on the WMT'18 English→German task.

picking an earlier checkpoint for generating word alignments could be beneficial for better supervision. Unfortunately, an alignment validation set does not exist for this task.

## 7 Related Work

Leveraging alignments obtained from statistical MT toolkits to guide NMT attention mechanisms has been explored in the past. Mi et al. (2016), Chen et al. (2016), Liu et al. (2016) and Alkhouli and Ney (2017) supervise the attention mechanisms of recurrent models (Bahdanau et al., 2015) in this way. Our multi-task framework is inspired by these publications. However, we examine its effect on the Transformer model (Vaswani et al., 2017), which provides state-of-the-art results on several translation benchmarks. Previous works report significant gains in translation accuracy in low resource settings, however gains remain modest given larger amounts of parallel data (millions of sentences). These approaches also fail to achieve significantly better alignment accuracy than the statistical MT toolkits. Peter et al. (2017) and Li et al. (2018) improve upon the previous works in terms of alignment accuracy by providing an alignment module with additional information about the to-be-aligned target word. Expanding on this idea, we propose to leverage the full target sentence context leading to AER improvements. Zenkel et al. (2019) presents an approach that eliminates the reliance on statistical word aligners by instead by directly optimizing the attention activations for predicting the target word. We empirically compare our approach of obtaining high quality alignments without the need of statistical word aligners to Zenkel et al. (2019).

Augmenting the task objective with linguistic information, such as word alignments, also has had applications beyond MT. Strubell et al. (2018) showed that adding linguistic information from parse trees into one of the attention heads of the transformer model can help in the semantic role labeling. Inspired by Strubell et al. (2018), we inject the alignment information through one of the attention heads for the translation task instead.

As a by-product of developing our model, we present a simple way to quantitatively evaluate and analyze the quality of attention probabilities learnt by different parts of the Transformer model with respect to modeling alignments, which contributes to previous work on understanding attention mechanisms (Ghader and Monz, 2017; Raganato and Tiedemann, 2018; Tang et al., 2018).

## 8 Conclusions

This paper addresses the task of jointly learning to produce translations and alignments with a single Transformer model. By using a multi-task objective along with providing full target sentence context to our alignment module, we are able to produce better alignments than previous approaches not relying on external alignment toolkits. We demonstrate that our framework can be extended to use external alignments from GIZA++ to achieve significantly better alignment results compared to GIZA++, while maintaining the same

translation performance.

Currently, our self-training based approach needs two training runs. To train our model in a single run, we would like to investigate a training method which alternates between alignment extraction and model training.

## Acknowledgments

We would like to thank the LILT team for releasing scripts and datasets for alignment evaluation. We are also grateful to Andrew Finch, Matthias Sperber, Barry Theobald and the anonymous reviewers for their helpful comments. Many thanks to Dorothea Peitz for helpful discussions about significance testing, Yi-Hsiu Liao for suggesting interesting extensions and the rest of Siri Machine Translation Team for their support.

## References

Tamer Alkhouli, Gabriel Bretschner, and Hermann Ney. 2018. On the alignment problem in multi-head attention-based neural machine translation. In *Conference on Empirical Methods in Natural Language Processing*, pages 177–185, Brussels, Belgium.

Tamer Alkhouli and Hermann Ney. 2017. Biasing attention-based recurrent neural networks using external alignment information. In *Conference on Empirical Methods in Natural Language Processing*, pages 108–117, Copenhagen, Denmark.

Philip Arthur, Graham Neubig, and Satoshi Nakamura. 2016. Incorporating discrete translation lexicons into neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1557–1567.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*, San Diego, CA, USA.

Ondřej Bojar, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Philipp Koehn, and Christof Monz. 2018. Findings of the 2018 conference on machine translation (wmt18). In *Conference on Statistical Machine Translation*, pages 272–303, Belgium, Brussels.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Rajen Chatterjee, Matteo Negri, Marco Turchi, Marcello Federico, Lucia Specia, and Frédéric Blain. 2017. Guiding neural machine translation decoding with external knowledge. In *Proceedings of the Second Conference on Machine Translation*, pages 157–168.

Wenhu Chen, Evgeny Matusov, Shahram Khadivi, and Jan-Thorsten Peter. 2016. Guided alignment training for topic-aware neural machine translation. In *Association for Machine Translation in the Americas*, pages 121–134, Austin, TX, USA.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Conference on Empirical Methods in Natural Language Processing*, pages 489–500, Brussels, Belgium.

Qin Gao and Stephan Vogel. 2008. Parallel implementations of word alignment tool. In *Software Engineering, Testing, and Quality Assurance for Natural Language Processing*, pages 49–57, Columbus, OH, USA.

Hamidreza Ghader and Christof Monz. 2017. What does attention in neural machine translation pay attention to? In *International Joint Conference on Natural Language Processing*, pages 30–39, Taipei, Taiwan.

Philipp Koehn, Amittai Axelrod, Ra Birch Mayne, Chris Callison-burch, Miles Osborne, and David Talbot. 2005. Edinburgh system description for the 2005 iwslt speech translation evaluation. In *International Workshop on Spoken Language Translation*, pages 68–75, Pittsburgh, PA, USA.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Annual Meeting of the Association for Computational Linguistics*, pages 177–180, Prague, Czech Republic.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39.

Xintong Li, Lemao Liu, Zhaopeng Tu, Shuming Shi, and Max Meng. 2018. Target foresight based attention for neural machine translation. In *Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1380–1390, New Orleans, LA, USA.

Lemao Liu, Masao Utiyama, Andrew Finch, and Eiichiro Sumita. 2016. Neural machine translation with supervised attention. In *International Conference on Computational Linguistics*, pages 3093–3102, Osaka, Japan.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*, New Orleans, LA, USA.

Haitao Mi, Zhiguo Wang, and Abe Ittycheriah. 2016. Supervised attentions for neural machine translation. In *Conference on Empirical Methods in Natural Language Processing*, pages 2283–2288, Austin, TX, USA.

Rada Mihalcea and Ted Pedersen. 2003. An evaluation exercise for word alignment. In *Proceedings of the HLT-NAACL 2003 Workshop on Building and Using Parallel Texts: Data Driven Machine Translation and Beyond - Volume 3*, pages 1–10, Edmonton, Canada.

Franz Josef Och and Hermann Ney. 2000. Improved statistical alignment models. In *Annual Meeting of the Association for Computational Linguistics*, pages 440–447, Hong Kong.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, PA, USA.

Jan-Thorsten Peter, Arne Nix Nix, and Hermann Ney. 2017. Generating alignments using target foresight in attention-based neural machine translation. In *Conference of the European Association for Machine Translation*, pages 27–36, Prague, Czech Republic.

Matt Post. 2018. A call for clarity in reporting bleu scores. In *Conference on Statistical Machine Translation*, pages 186–191, Belgium, Brussels.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Conference of the European Chapter of the Association for Computational Linguistics*, pages 157–163, Valencia, Spain.

Alessandro Raganato and Jörg Tiedemann. 2018. An analysis of encoder representations in transformer-based machine translation. In *BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 287–297, Brussels, Belgium.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Annual Meeting of the Association for Computational Linguistics*, pages 1715–1725, Berlin, Germany.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Conference on Empirical Methods in Natural Language Processing*, pages 5027–5038, Brussels, Belgium.

Gongbo Tang, Rico Sennrich, and Joakim Nivre. 2018. An analysis of attention mechanisms: The case of word sense disambiguation in neural machine translation. In *Conference on Statistical Machine Translation*, pages 26–35, Brussels, Belgium.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 76–85.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 1–11, Long Beach, CA, USA.

David Vilar, Maja Popović, and Hermann Ney. 2006. AER: Do we need to "improve" our alignments? In *International Workshop on Spoken Language Translation*, pages 205–212, Kyoto, Japan.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *International Conference on Computational Linguistics*, pages 836–841, Copenhagen, Denmark.

Thomas Zenkel, Joern Wuebker, and John DeNero. 2019. Adding Interpretable Attention to Neural Translation Models Improves Word Alignment. *arXiv e-prints*.

# SOCIAL IQA: Commonsense Reasoning about Social Interactions

**Maarten Sap**[⋆] [◇♡] **Hannah Rashkin**[⋆] [◇♡] **Derek Chen**[♡] **Ronan Le Bras**[◇] **Yejin Choi**[◇♡]
[◇]Allen Institute for Artificial Intelligence, Seattle, WA, USA
[♡]Paul G. Allen School of Computer Science & Engineering, Seattle, WA, USA
{msap,hrashkin,dchen14,yejin}@cs.washington.edu
{ronanlb}@allenai.org

## Abstract

We introduce SOCIAL IQA, the first large-scale benchmark for commonsense reasoning about social situations. SOCIAL IQA contains 38,000 multiple choice questions for probing *emotional* and *social* intelligence in a variety of everyday situations (e.g., Q: "*Jordan wanted to tell Tracy a secret, so Jordan leaned towards Tracy. Why did Jordan do this?*" A: "*Make sure no one else could hear*"). Through crowdsourcing, we collect commonsense questions along with correct and incorrect answers about social interactions, using a new framework that mitigates stylistic artifacts in incorrect answers by asking workers to provide the right answer to a different but related question. Empirical results show that our benchmark is challenging for existing question-answering models based on pretrained language models, compared to human performance (>20% gap). Notably, we further establish SOCIAL IQA as a resource for transfer learning of commonsense knowledge, achieving state-of-the-art performance on multiple commonsense reasoning tasks (Winograd Schemas, COPA).

## 1 Introduction

Social and emotional intelligence enables humans to reason about the mental states of others and their likely actions (Ganaie and Mudasir, 2015). For example, when someone spills food all over the floor, we can infer that they will likely want to clean up the mess, rather than taste the food off the floor or run around in the mess (Figure 1, middle). This example illustrates how Theory of Mind, i.e., the ability to reason about the implied emotions and behavior of others, enables humans to navigate social situations ranging from simple conversations with friends to complex negotiations in courtrooms (Apperly, 2010).




Figure 1: Three context-question-answers triples from SOCIAL IQA, along with the type of reasoning required to answer them. In the top example, humans can trivially infer that Tracy pressed upon Austin because there was no room in the elevator. Similarly, in the bottom example, commonsense tells us that people typically root for the hero, not the villain.

While humans trivially acquire and develop such social reasoning skills (Moore, 2013), this is still a challenge for machine learning models, in part due to the lack of large-scale resources to train and evaluate modern AI systems' social and emotional intelligence. Although recent advances in pretraining large language models have yielded promising improvements on several commonsense inference tasks, these models still struggle to reason about social situations, as shown in this and previous work (Davis and Marcus,

[⋆] Both authors contributed equally.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4463–4473, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

2015; Nematzadeh et al., 2018; Talmor et al., 2019). This is partly due to language models being trained on written text corpora, where reporting bias of knowledge limits the scope of commonsense knowledge that can be learned (Gordon and Van Durme, 2013; Lucy and Gauthier, 2017).

In this work, we introduce Social Intelligence QA (SOCIAL IQA), the first large-scale resource to learn and measure *social* and *emotional* intelligence in computational models.[1] SOCIAL IQA contains 38k multiple choice questions regarding the pragmatic implications of everyday, social events (see Figure 1). To collect this data, we design a crowdsourcing framework to gather contexts and questions that explicitly address social commonsense reasoning. Additionally, by combining handwritten negative answers with adversarial question-switched answers (Section 3.3), we minimize annotation artifacts that can arise from crowdsourcing incorrect answers (Schwartz et al., 2017; Gururangan et al., 2018).

This dataset remains challenging for AI systems, with our best performing baseline reaching 64.5% (BERT-large), significantly lower than human performance. We further establish SOCIAL IQA as a resource that enables transfer learning for other commonsense challenges, through *sequential finetuning* of a pretrained language model on SOCIAL IQA before other tasks. Specifically, we use SOCIAL IQA to set a new state-of-the-art on three commonsense challenge datasets: COPA (Roemmele et al., 2011) (83.4%), the original Winograd (Levesque, 2011) (72.5%), and the extended Winograd dataset from Rahman and Ng (2012) (84.0%).

Our contributions are as follows: (1) We create SOCIAL IQA, the first large-scale QA dataset aimed at testing social and emotional intelligence, containing over 38k QA pairs. (2) We introduce question-switching, a technique to collect incorrect answers that minimizes stylistic artifacts due to annotator cognitive biases. (3) We establish baseline performance on our dataset, with BERT-large performing at 64.5%, well below human performance. (4) We achieve new state-of-the-art accuracies on COPA and Winograd through sequential finetuning on SOCIAL IQA, which implicitly endows models with social commonsense knowledge.

[1] Available at https://tinyurl.com/socialiqa

| SOCIAL IQA | | |
|---|---|---|
| # QA tuples | train | 33,410 |
| | dev | 1,954 |
| | test | 2,224 |
| | **total** | **37,588** |
| Train statistics | | |
| Average # tokens | context | 14.04 |
| | question | 6.12 |
| | answers (all) | 3.60 |
| | answers (correct) | 3.65 |
| | answers (incorrect) | 3.58 |
| Unique # tokens | context | 15,764 |
| | question | 1,165 |
| | answers (all) | 12,285 |
| | answers (correct) | 7,386 |
| | answers (incorrect) | 10,514 |
| Average freq. of answers | answers (correct) | 1.37 |
| | answers (incorrect) | 1.47 |

Table 1: Data statistics for SOCIAL IQA.

## 2 Task description

SOCIAL IQA aims to measure the social and emotional intelligence of computational models through multiple choice question answering (QA). In our setup, models are confronted with a question explicitly pertaining to an observed *context*, where the correct answer can be found among three competing options.

By design, the questions require *inferential* reasoning about the social causes and effects of situations, in line with the type of intelligence required for an AI assistant to interact with human users (e.g., know to call for help when an elderly person falls; Pollack, 2005). As seen in Figure 1, correctly answering questions requires reasoning about motivations, emotional reactions, or likely preceding and following actions. Performing these inferences is what makes us experts at navigating social situations, and is closely related to Theory of Mind, i.e., the ability to reason about the beliefs, motivations, and needs of others (Baron-Cohen et al., 1985).[2] Endowing machines with this type of intelligence has been a longstanding but elusive goal of AI (Gunning, 2018).

[2] Theory of Mind is well developed in most neurotypical adults (Ganaie and Mudasir, 2015), but can be influenced by age, culture, or developmental disorders (Korkmaz, 2011).

**ATOMIC**

As a starting point for our task creation, we draw upon social commonsense knowledge from ATOMIC (Sap et al., 2019) to seed our contexts and question types. ATOMIC is a large knowledge graph that contains inferential knowledge about the causes and effects of 24k short events. Each triple in ATOMIC consists of an event phrase with person-centric variables, one of nine inference dimensions, and an inference object (e.g., "PersonX pays for PersonY's ___", "xAttrib", "generous"). The nine inference dimensions in ATOMIC cover causes of an event (e.g., "X needs money"), its effects on the agent (e.g., "X will get thanked") and its effect on other participants (e.g., "Y will want to see X again"); see Sap et al. (2019) for details.

Given this base, we generate natural language contexts that represent specific instantiations of the event phrases found in the knowledge graph. Furthermore, the questions created probe the commonsense reasoning required to navigate such contexts. Critically, since these contexts are based off of ATOMIC, they explore a diverse range of motivations and reactions, as well as likely preceding or following actions.

## 3 Dataset creation

SOCIAL IQA contains 37,588 multiple choice questions with three answer choices per question. Questions and answers are gathered through three phases of crowdsourcing aimed to collect the *context*, the *question*, and a set of *positive* and *negative* answers. We run crowdsourcing tasks on Amazon Mechanical Turk (MTurk) to create each of the three components, as described below.

### 3.1 Event Rewriting

In order to cover a variety of social situations, we use the base events from ATOMIC as prompts for context creation. As a pre-processing step, we run an MTurk task that asks workers to turn an ATOMIC event (e.g., "PersonX spills ___ all over the floor") into a sentence by adding names, fixing potential grammar errors, and filling in placeholders (e.g., "Alex spilled food all over the floor.").[3]

### 3.2 Context, Question, & Answer Creation

Next, we run a task where annotators create full context-question-answers triples. We automatically generate question templates covering the nine commonsense inference dimensions in ATOMIC.[4] Crowdsourcers are prompted with an event sentence and an inference question to turn into a more detailed context[5] (e.g. "Alex spilled food all over the floor and it made a huge mess.") and an edited version of the question if needed for improved specificity (e.g. "What will Alex want to do next?"). Workers are also asked to contribute two potential correct answers.

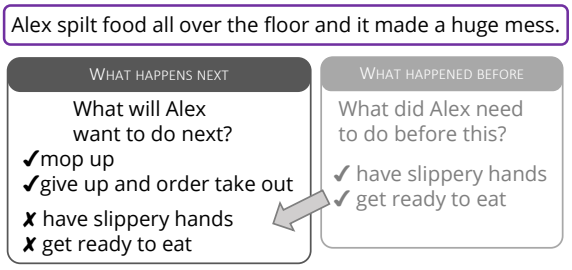


Figure 2: Question-Switching Answers (QSA) are collected as the correct answers to the wrong question that targets a different type of inference (here, reasoning about what happens before instead of after an event).

### 3.3 Negative Answers

In addition to correct answers, we collect four incorrect answer options, of which we filter out two. To create incorrect options that are adversarial for models but easy for humans, we use two different approaches to the collection process. These two methods are specifically designed to avoid different types of annotation artifacts, thus making it more difficult for models to rely on data biases. We integrate and filter answer options and validate final QA tuples with human rating tasks.

**Handwritten Incorrect Answers (HIA)** The first method involves eliciting handwritten incorrect answers that require reasoning about the context. These answers are handwritten to be similar to the correct answers in terms of topic, length, and style but are subtly incorrect. Two of these answers are collected during the same MTurk task as the original context, questions, and correct answers. We will refer to these negative responses as handwritten incorrect answers (HIA).

**Question-Switching Answers (QSA)** We collect a second set of negative (incorrect) answer

[3]This task paid $0.35 per event.

[4]We do not generate templates if the ATOMIC dimension is annotated as "none."

[5]Workers were asked to contribute a context 7-25 words longer than the event sentence.

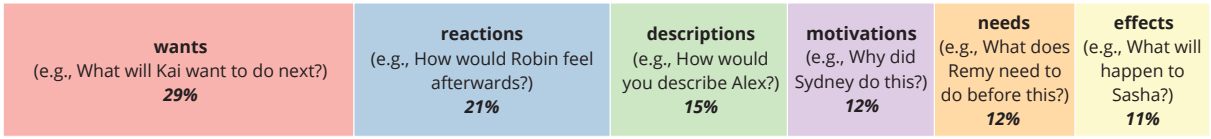


Figure 3: SOCIAL IQA contains several question types which cover different types of inferential reasoning. Question types are derived from ATOMIC inference dimensions.

candidates by switching the questions asked about the context, as shown in Figure 2. We do this to avoid cognitive biases and annotation artifacts in the answer candidates, such as those caused by writing incorrect answers or negations (Schwartz et al., 2017; Gururangan et al., 2018). In this crowdsourcing task, we provide the same context as the original question, as well as a question automatically generated from a different but similar ATOMIC dimension,[6] and ask workers to write two correct answers. We refer to these negative responses as question-switching answers (QSA).

By including answers to a different question about the same context, we ensure that these adversarial responses have the stylistic qualities of correct answers and strongly relate to the context topic, while still being incorrect, making it difficult for models to simply perform pattern-matching. To verify this, we compare valence, arousal, and dominance (VAD) levels across answer types, computed using the VAD lexicon by Mohammad (2018). Figure 4 shows effect sizes (Cohen's $d$) of the differences in VAD means, where the magnitude of effect size indicates how different the answer types are stylistically. Indeed, QSA and correct answers differ substantially less than HIA answers ($|d| \leq .1$).[7]

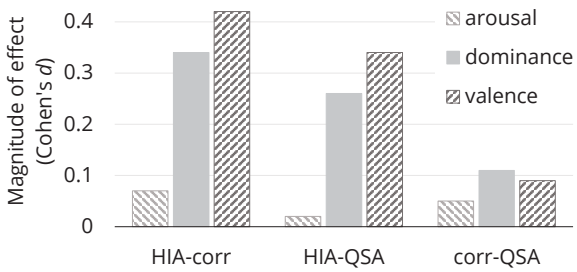


Figure 4: Magnitude of effect sizes (Cohen's $d$) when comparing average dominance, arousal and valence values of different answer types where larger $|d|$ indicates more stylistic difference. For valence (sentiment polarity) and dominance, the effect sizes comparing QSA and correct answers are much smaller, indicating that these are more similar tonally. Notably, all three answer types have comparable levels of arousal (intensity).

### 3.4 QA Tuple Creation

As the final step of the pipeline, we aggregate the data into three-way multiple choice questions. For each created context-question pair contributed by crowdsourced workers, we select a random correct answer and the incorrect answers that are least entailed by the correct one, following inspiration from Zellers et al. (2019a).

For the training data, we validate our QA tuples through a multiple-choice crowdsourcing task where three workers are asked to select the right answer to the question provided.[8] In order to ensure even higher quality, we validate the dev and test data a second time with five workers. Our final dataset contains questions for which the correct answer was determined by human majority voting, discarding cases without a majority vote. We also apply a lightweight form of adversarial filtering to make the task more challenging by using a deep stylistic classifier to remove easier examples on the dev and test sets (Sakaguchi et al., 2019).[9]

To obtain human performance, we run a separate task asking three new workers to select the correct answer on a random subset of 900 dev and 900 test examples. Human performance on these subsets is 87% and 84%, respectively.

### 3.5 Data Statistics

To keep contexts separate across train/dev/test sets, we assign SOCIAL IQA contexts to the same partition as the ATOMIC event the context was based on. Shown in Table 1 (top), this yields a

[6] Using the following three groupings of ATOMIC dimensions: {xWant, oWant, xNeed, xIntent}, {xReact oReact, xAttr}, and {xEffect, oEffect}.

[7] Cohen's $|d| < .20$ is considered small (Sawilowsky, 2009). We find similarly small effect sizes using other sentiment/emotion lexicons.

[8] Agreement on this task was high (Cohen's $\kappa$=.70)

[9] We also tried filtering to remove examples from the training set but found it did not significantly change performance. We will release tags for the easier training examples with the full data.

total set of around 33k training, 2k dev, and 2k test tuples. We additionally include statistics on word counts and vocabulary of the training data. We report the averages of correct and incorrect answers in terms of: token length, number of unique tokens, and number of times a unique answer appears in the dataset. Note that due to our three-way multiple choice setup, there are twice as many incorrect answers which influences these statistics.

We also include a breakdown (Figure 3) across question types, which we derive from ATOMIC inference dimensions.[10] In general, questions relating to what someone will feel afterwards or what they will likely do next are more common in SOCIAL IQA. Conversely, questions pertaining to (potentially involuntary) effects of situations on people are less frequent.

| Model | Accuracy (%) Dev | Test |
|---|---|---|
| Random baseline | 33.3 | 33.3 |
| GPT | 63.3 | 63.0 |
| BERT-base | 63.3 | 63.1 |
| BERT-large | **66.0** | **64.5** |
| w/o context | 52.7 | – |
| w/o question | 52.1 | – |
| w/o context, question | 45.5 | – |
| Human | 86.9* | 84.4* |

Table 2: Experimental results. We additionally perform an ablation by removing contexts and questions, verifying that both are necessary for BERT-large's performance. Human evaluation results are obtained using 900 randomly sampled examples.

## 4 Methods

We establish baseline performance on SOCIAL IQA, using large pretrained language models based on the Transformer architecture (Vaswani et al., 2017). Namely, we finetune OpenAI-GPT (Radford et al., 2018) and BERT (Devlin et al., 2019), which have both shown remarkable improvements on a variety of tasks. OpenAI-GPT is a uni-directional language model trained on the BookCorpus (Zhu et al., 2015), whereas BERT is a bidirectional language model trained on both the BookCorpus and English Wikipedia. As per previous work, we finetune the language model representations but fully learn the classifier specific parameters described below.

**Multiple choice classification** To classify sequences using these language models, we follow the multiple-choice setup implementation by the respective authors, as described below. First, we concatenate the context, question, and answer, using the model specific separator tokens. For OpenAI-GPT, the format becomes `_start_ <context> <question> _delimiter_ <answer> _classify_`, where `_start_`, `_delimiter_`, and `_classify_` are special function tokens. For BERT, the format is similar, but the classifier token comes before the context.[11]

For each triple, we then compute a score $l$ by passing the hidden representation from the classifier token $h_{CLS} \in \mathbb{R}^H$ through an MLP:

$$l = W_2 \tanh(W_1 h_{CLS} + b_1)$$

where $W_1 \in \mathbb{R}^{H \times H}$, $b_1 \in \mathbb{R}^H$ and $W_2 \in \mathbb{R}^{1 \times H}$. Finally, we normalize scores across all triples for a given context-question pair using a softmax layer. The model's predicted answer corresponds to the triple with the highest probability.

## 5 Experiments

### 5.1 Experimental Set-up

We train our models on the 33k SOCIAL IQA training instances, selecting hyperparameters based on the best performing model on our dev set, for which we then report test results. Specifically, we perform finetuning through a grid search over the hyper-parameter settings (with a learning rate in $\{1e{-}5, 2e{-}5, 3e{-}5\}$, a batch size in $\{3, 4, 8\}$, and a number of epochs in $\{3, 4, 10\}$) and report the maximum performance.

Models used in our experiments vary in sizes: OpenAI-GPT (117M parameters) has a hidden size $H$=768, BERT-base (110M params) and BERT-large (340M params) hidden sizes of $H$=768 and $H$=1024, respectively. We train using the HuggingFace PyTorch (Paszke et al., 2017) implementation.[12]

[10] We group agent and theme ATOMIC dimensions together (e.g., "xReact" and "oReact" become the "reactions" question type).

[11] BERT's format is `[CLS] <context> [UNUSED] <question> [SEP] <answer> [SEP]`

[12] https://github.com/huggingface/pytorch-pretrained-BERT

| | Context | Question | | Answer |
|---|---|---|---|---|
| (1) | Jesse was pet sitting for Addison, so Jesse came to Addison's house and walked their dog. | What does Jesse need to do before this? | ✓ 🤖 | (a) feed the dog<br>(b) get a key from Addison<br>(c) walk the dog |
| (2) | Kai handed back the computer to Will after using it to buy a product off Amazon. | What will Kai want to do next? | ✓ 🤖 | (a) wanted to save money on shipping<br>(b) Wait for the package<br>(c) Wait for the computer |
| (3) | Remy gave Skylar, the concierge, her account so that she could check into the hotel. | What will Remy want to do next? | 🤖<br>✓ | (a) lose her credit card<br>(b) arrive at a hotel<br>(c) get the key from Skylar |
| (4) | Sydney woke up and was ready to start the day. They put on their clothes. | What will Sydney want to do next? | 🤖<br>✓ | (a) go to bed<br>(b) go to the pool<br>(c) go to work |
| (5) | Kai grabbed Carson's tools for him because Carson could not get them. | How would Carson feel as a result? | 🤖<br>✓ | (a) inconvenienced<br>(b) grateful<br>(c) angry |
| (6) | Although Aubrey was older and stronger, they lost to Alex in arm wrestling. | How would Alex feel as a result? | 🤖<br>✓ | (a) they need to practice more<br>(b) ashamed<br>(c) boastful |

Table 3: Example CQA triples from the SOCIAL IQA dev set with BERT-large's predictions (🤖: BERT's prediction, ✓: true correct answer). The model predicts correctly in (1) and (2) and incorrectly in the other four examples shown here. Examples (3) and (4) illustrate the model choosing answers that might have happened before, or that might happen much later after the context, as opposed to right after the context situation. In Examples (5) and (6), the model chooses answers that may apply to people other than the ones being asked about.

### 5.2 Results

Our results (Table 2) show that SOCIAL IQA is still a challenging benchmark for existing computational models, compared to human performance. Our best performing model, BERT-large, outperforms other models by several points on the dev and test set. We additionally ablate our best model's representation by removing the context and question from the input, confirming that reasoning over both is necessary for this task.

**Learning Curve** To better understand the effect of dataset scale on model performance on our task, we simulate training situations with limited knowledge. We present the learning curve of BERT-large's performance on the dev set as it is trained on more training set examples (Figure 5). Although the model does significantly improve over a random baseline of 33% with only a few hundred examples, the performance only starts to converge after around 20k examples, providing evidence that large-scale benchmarks are required for this type of reasoning.

**Error Analysis** We include a breakdown of our best model's performance on various question types in Figure 6 and specific examples of errors in the last four rows of Table 3. Overall, questions related to pre-conditions of the context (people's motivations, actions needed before the context) are less challenging for the model. Conversely, the model seems to struggle more with questions relating to (potentially involuntary) effects, stative descriptions, and what people will want to do next.

Examples of errors in Table 3 further indicate that, instead of doing advanced reasoning about situations, models may only be learning lexical associations between the context, question, and answers, as hinted at by Marcus (2018) and Zellers et al. (2019b). This leads the model to select answers with incorrect timing (examples 3 and 4) or answers pertaining to the wrong participants (examples 5 and 6), despite being trained on large amounts of examples that specifically distinguish proper timing and participants. For instance, in (3) and (4), the model selects answers which

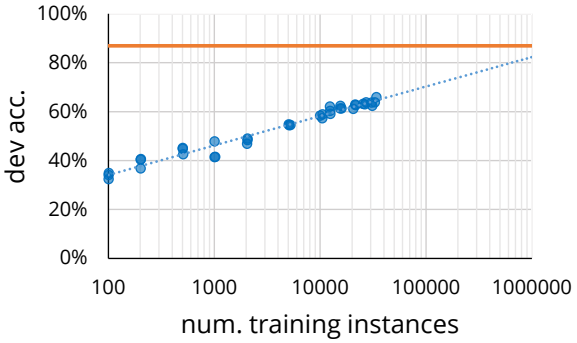


Figure 5: Dev accuracy when training BERT-large with various number of examples (multiple runs per training size), with human performance (86.9%) shown in orange. In order to reach >80%, the model would require nearly 1 million training examples.

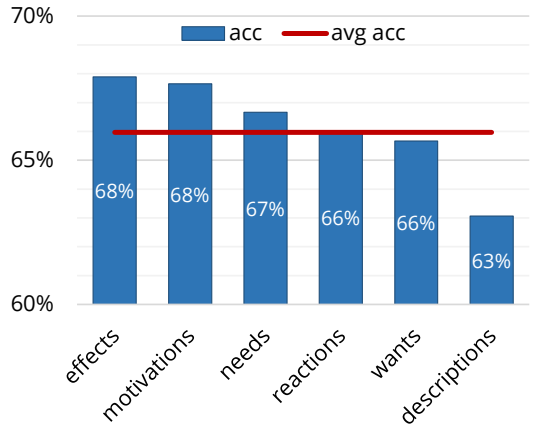


Figure 6: Average dev accuracy of BERT-large on different question types. While questions about effects and motivations are easier, the model still finds wants and descriptions more challenging.

are incorrectly timed with respect to the context and question (e.g., "arrive at a hotel" is something Remy likely did before checking in with the concierge, not afterwards). Additionally, the model often chooses answers related to a person other than the one asked about. In (6), after the arm wrestling, though it is likely that Aubrey will feel ashamed, the question relates to what Alex might feel–not Aubrey.

Overall, our results illustrate how reasoning about social situations still remains a challenge for these models, compared to humans who can trivially reason about the causes and effects for multiple participants. We expect that this task would benefit from models capable of more complex reasoning about entity state, or models that are more explicitly endowed with commonsense (e.g., from knowledge graphs like ATOMIC).

## 6 SOCIAL IQA for Transfer Learning

In addition to being the first large-scale benchmark for social commonsense, we also show that SOCIAL IQA can improve performance on downstream tasks that require commonsense, namely the Winograd Schema Challenge and the Choice of Plausible Alternatives task. We achieve state of the art performance on both tasks by sequentially finetuning on SOCIAL IQA before the task itself.

**COPA** The Choice of Plausible Alternatives task (COPA; Roemmele et al., 2011) is a two-way multiple choice task which aims to measure commonsense reasoning abilities of models. The dataset contains 1,000 questions (500 dev, 500 test) that ask about the causes and effects of a premise. This has been a challenging task for computational systems, partially due to the limited amount of training data available. As done previously (Goodwin et al., 2012; Luo et al., 2016), we finetune our models on the dev set, and report performance only on the test set.

**Winograd Schema** The Winograd Schema Challenge (WSC; Levesque, 2011) is a well-known commonsense knowledge challenge framed as a coreference resolution task. It contains a collection of 273 short sentences in which a pronoun must be resolved to one of two antecedents (e.g., in "The city councilmen refused the demonstrators a permit because *they* feared violence", *they* refers to the councilmen). Because of data scarcity in WSC, Rahman and Ng (2012) created 943 Winograd-style sentence pairs (1886 sentences in total), henceforth referred to as DPR, which has been shown to be slightly less challenging than WSC for computational models.

We evaluate on these two benchmarks. While the DPR dataset is split into train and test sets (Rahman and Ng, 2012), the WSC dataset contains a single (*test*) set of only 273 instances for evaluation purposes only. Therefore, we use the DPR dataset as training set when evaluating on the WSC dataset.

### 6.1 Sequential Finetuning

We first finetune BERT-large on SOCIAL IQA, which reaches 66% on our dev set (Table 2). We then finetune that model further on the task-specific datasets, considering the same set of hyperparameters as in §5.1. On each of the test sets,

| Task | Model | Acc. (%) best | mean | std |
|---|---|---|---|---|
| COPA | Sasaki et al. (2017) | 71.2 | – | – |
| | BERT-large | 80.8 | 75.0 | 3.0 |
| | BERT-SOCIAL IQA | **83.4** | 80.1 | 2.0 |
| WSC | Kocijan et al. (2019) | 72.5 | – | – |
| | BERT-large | 67.0 | 65.5 | 1.0 |
| | BERT-SOCIAL IQA | **72.5** | 69.6 | 1.7 |
| DPR | Peng et al. (2015) | 76.4 | – | – |
| | BERT-large | 79.4 | 71.2 | 3.8 |
| | BERT-SOCIAL IQA | **84.0** | 81.7 | 1.2 |

Table 4: Sequential finetuning of BERT-large on SOCIAL IQA before the task yields state of the art results (bolded) on COPA (Roemmele et al., 2011), Winograd Schema Challenge (Levesque, 2011) and DPR (Rahman and Ng, 2012). For comparison, we include previous published state of the art performance.

we report best, mean, and standard deviation of all models, and compare sequential finetuning results to a BERT-large baseline.

**Results** Shown in Table 4, sequential finetuning on SOCIAL IQA yields substantial improvements over the BERT-only baseline (between 2.6 and 5.5% max performance increases), as well as the general increase in performance stability (i.e., lower standard deviations). As hinted at by Phang et al. (2019), this suggests that BERT-large can benefit from both the large scale and the QA format of commonsense knowledge in SOCIAL IQA, which it struggles to learn from small benchmarks only. Notably, we find that sequentially finetuned BERT-SOCIAL IQA achieves state-of-the-art results on all three tasks, showing improvements of previous best performing models.[13]

**Effect of scale and knowledge type** To better understand these improvements in downstream task performance, we investigate the impact on COPA performance of sequential finetuning on less SOCIAL IQA training data (Figure 7), as well as the impact of the type of commonsense knowledge used in sequential finetuning. As expected, the downstream performance on COPA improves when using a model pretrained on more of SOCIAL IQA, indicating that the scale of the dataset

[13]Note that OpenAI-GPT was reported to achieve 78.6% on COPA, but that result was not published, nor discussed in the OpenAI-GPT white paper (Radford et al., 2018).

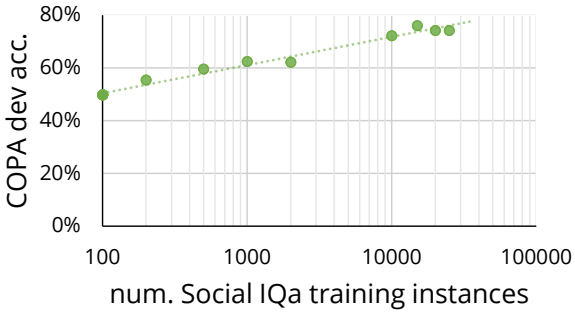


Figure 7: Effect of finetuning BERT-large on varying sizes of the SOCIAL IQA training set on the dev accuracy of COPA. As expected, the more SOCIAL IQA instances the model is finetuned on, the better the accuracy on COPA.

is one factor that helps in the fine-tuning. However, when using SWAG (a similarly sized dataset) instead of SOCIAL IQA for sequential finetuning, the downstream performance on COPA is lower (76.2%). This indicates that, in addition to its large scale, the social and emotional nature of the knowledge in SOCIAL IQA enables improvements on these downstream tasks.

## 7 Related Work

**Commonsense Benchmarks:** Commonsense benchmark creation has been well-studied by previous work. Notably, the WinoGrad Schema Challenge (WSC; Levesque, 2011) and the Choice Of Plausible Alternatives dataset (COPA; Roemmele et al., 2011) are expert-curated collections of commonsense QA pairs that are trivial for humans to solve. Whereas WSC requires physical and social commonsense knowledge to solve, COPA targets the knowledge of causes and effects surrounding social situations. While both benchmarks are of high-quality and created by experts, their small scale (150 and 1,000 examples, respectively) poses a challenge for modern modelling techniques, which require many training instances.

More recently, Talmor et al. (2019) introduce CommonsenseQA, containing 12k multiple-choice questions. Crowdsourced using ConceptNet (Speer and Havasi, 2012), these questions mostly probe knowledge related to factual and physical commonsense (e.g., "Where would I not want a fox?"). In contrast, SOCIAL IQA explicitly separates contexts from questions, and focuses on the types of commonsense inferences humans perform when navigating social situations.

**Commonsense Knowledge Bases:** In addition to large-scale benchmarks, there is a wealth of work aimed at creating commonsense knowledge repositories (Speer and Havasi, 2012; Sap et al., 2019; Zhang et al., 2017; Lenat, 1995; Espinosa and Lieberman, 2005; Gordon and Hobbs, 2017) that can be used as resources in downstream reasoning tasks. While SOCIAL IQA is formatted as a natural language QA benchmark, rather than a taxonomic knowledge base, it also can be used as a resource for external tasks, as we have demonstrated experimentally.

**Constrained or Adversarial Data Collection:** Various work has investigated ways to circumvent annotation artifacts that result from crowdsourcing. Sharma et al. (2018) extend the Story Cloze data by severely restricting the incorrect story ending generation task, reducing the sentiment and negation artifacts. Rajpurkar et al. (2018) create an adversarial version of the extractive question-answering challenge, SQuAD (Rajpurkar et al., 2016), by creating 50k unanswerable questions. Instead of using human-generated incorrect answers, Zellers et al. (2018, 2019b) use adversarial filtering of machine generated incorrect answers to minimize surface patterns. Our dataset also aims to reduce annotation artifacts by using a multi-stage annotation pipeline in which we collect negative responses from multiple methods including a unique adversarial question-switching technique.

## 8 Conclusion

We present SOCIAL IQA, the first large-scale benchmark for social commonsense. Consisting of 38k multiple-choice questions, SOCIAL IQA covers various types of inference about people's actions being described in situational contexts. We design a crowdsourcing framework for collecting QA pairs that reduces stylistic artifacts of negative answers through an adversarial question-switching method. Despite human performance of close to 90%, computational approaches based on large pretrained language models only achieve accuracies up to 65%, suggesting that these social inferences are still a challenge for AI systems. In addition to providing a new benchmark, we demonstrate how transfer learning from SOCIAL IQA to other commonsense challenges can yield significant improvements, achieving new state-of-the-art performance on both COPA and Winograd Schema Challenge datasets.

## Acknowledgments

We thank Chandra Bhagavatula, Hannaneh Hajishirzi, and other members of the UW NLP and AI2 community for helpful discussions and feedback throughout this project. We also thank the anonymous reviewers for their insightful comments and suggestions. This research was supported in part by NSF (IIS-1524371, IIS-1714566), DARPA under the CwC program through the ARO (W911NF-15-1-0543), DARPA under the MCS program through NIWC Pacific (N66001-19-2-4031), Samsung Research, and the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-1256082.

## References

Ian Apperly. 2010. *Mindreaders: the cognitive basis of" theory of mind"*. Psychology Press.

Simon Baron-Cohen, Alan M Leslie, and Uta Frith. 1985. Does the Autistic Child have a "Theory of Mind"? *Cognition*, 21(1):37–46.

Ernest Davis and Gary Marcus. 2015. Commonsense reasoning and commonsense knowledge in artificial intelligence. *Commun. ACM*, 58:92–103.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

José H. Espinosa and Henry Lieberman. 2005. Eventnet: Inferring temporal relations between commonsense events. In *MICAI*.

MY Ganaie and Hafiz Mudasir. 2015. A Study of Social Intelligence & Academic Achievement of College Students of District Srinagar, J&K, India. *Journal of American Science*, 11(3):23–27.

Travis Goodwin, Bryan Rink, Kirk Roberts, and Sanda M Harabagiu. 2012. UTDHLT: Copacetic system for choosing plausible alternatives. In *NAACL workshop on SemEval*, pages 461–466. Association for Computational Linguistics.

Andrew S Gordon and Jerry R Hobbs. 2017. *A Formal Theory of Commonsense Psychology: How People Think People Think*. Cambridge University Press.

Jonathan Gordon and Benjamin Van Durme. 2013. Reporting bias and knowledge acquisition. In *Proceedings of the 2013 Workshop on Automated Knowledge Base Construction*, AKBC '13, pages 25–30, New York, NY, USA. ACM.

David Gunning. 2018. Machine common sense concept paper.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R. Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *NAACL-HLT*.

Vid Kocijan, Ana-Maria Cretu, Oana-Maria Camburu, Yordan Yordanov, and Thomas Lukasiewicz. 2019. A surprisingly robust trick for the winograd schema challenge. In *ACL*.

Baris Korkmaz. 2011. Theory of mind and neurodevelopmental disorders of childhood. *Pediatr Res*, 69(5 Pt 2):101R–8R.

Douglas B Lenat. 1995. Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38.

Hector J. Levesque. 2011. The winograd schema challenge. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*.

Li Lucy and Jon Gauthier. 2017. Are distributional representations ready for the real world? evaluating word vectors for grounded perceptual meaning. In *RoboNLP@ACL*.

Zhiyi Luo, Yuchen Sha, Kenny Q Zhu, Seung-won Hwang, and Zhongyuan Wang. 2016. Commonsense causal reasoning between short texts. In *Fifteenth International Conference on the Principles of Knowledge Representation and Reasoning*.

Gary Marcus. 2018. Deep learning: A critical appraisal. *CoRR*, abs/1801.00631.

Saif Mohammad. 2018. Obtaining reliable human ratings of valence, arousal, and dominance for 20,000 english words. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 174–184.

Chris Moore. 2013. *The development of commonsense psychology*. Psychology Press.

Aida Nematzadeh, Kaylee Burns, Erin Grant, Alison Gopnik, and Thomas L. Griffiths. 2018. Evaluating theory of mind in question answering. In *EMNLP*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NIPS-W*.

Haoruo Peng, Daniel Khashabi, and Dan Roth. 2015. Solving hard coreference problems. In *HLT-NAACL*.

Jason Phang, Thibault Févry, and Samuel R. Bowman. 2019. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *CoRR*, abs/1811.01088.

Martha E. Pollack. 2005. Intelligent technology for an aging population: The use of ai to assist elders with cognitive impairment. *AI Magazine*, 26:9–24.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative Pre-Training.

Altaf Rahman and Vincent Ng. 2012. Resolving complex cases of definite pronouns: The winograd schema challenge. In *EMNLP*, EMNLP-CoNLL '12, pages 777–789, Stroudsburg, PA, USA. Association for Computational Linguistics.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy S. Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *EMNLP*.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S. Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Winogrande: An adversarial winograd schema challenge at scale. *ArXiv*, abs/1907.10641.

Maarten Sap, Ronan Le Bras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A Smith, and Yejin Choi. 2019. Atomic: An atlas of machine commonsense for if-then reasoning. In *AAAI*.

Shota Sasaki, Sho Takase, Naoya Inoue, Naoaki Okazaki, and Kentaro Inui. 2017. Handling multiword expressions in causality estimation. In *IWCS*.

Shlomo S. Sawilowsky. 2009. New effect size rules of thumb. *Journal of Modern Applied Statistical Methods*, 8(2):597–599.

Roy Schwartz, Maarten Sap, Ioannis Konstas, Li Zilles, Yejin Choi, and Noah A Smith. 2017. The effect of different writing tasks on linguistic style: A case study of the ROC story cloze task. In *CoNLL*.

Rishi Kant Sharma, James Allen, Omid Bakhshandeh, and Nasrin Mostafazadeh. 2018. Tackling the story ending biases in the story cloze test. In *ACL*.

Robyn Speer and Catherine Havasi. 2012. Representing general relational knowledge in conceptnet 5. In *LREC*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *NAACL*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Rowan Zellers, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019a. From recognition to cognition: Visual commonsense reasoning. In *CVPR*.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. SWAG: A large-scale adversarial dataset for grounded commonsense inference. In *EMNLP*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019b. Hellaswag: Can a machine really finish your sentence? In *ACL*.

Sheng Zhang, Rachel Rudinger, Kevin Duh, and Benjamin Van Durme. 2017. Ordinal common-sense inference. *Transactions of the Association of Computational Linguistics*, 5(1):379–395.

Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan R. Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 19–27.

# Self-Assembling Modular Networks for Interpretable Multi-Hop Reasoning

**Yichen Jiang** and **Mohit Bansal**
UNC Chapel Hill
{yichenj, mbansal}@cs.unc.edu

## Abstract

Multi-hop QA requires a model to connect multiple pieces of evidence scattered in a long context to answer the question. The recently proposed HotpotQA (Yang et al., 2018) dataset is comprised of questions embodying four different multi-hop reasoning paradigms (two bridge entity setups, checking multiple properties, and comparing two entities), making it challenging for a single neural network to handle all four. In this work, we present an interpretable, controller-based Self-Assembling Neural Modular Network (Hu et al., 2017, 2018) for multi-hop reasoning, where we design four novel modules (Find, Relocate, Compare, NoOp) to perform unique types of language reasoning. Based on a question, our layout controller RNN dynamically infers a series of reasoning modules to construct the entire network. Empirically, we show that our dynamic, multi-hop modular network achieves significant improvements over the static, single-hop baseline (on both regular and adversarial evaluation). We further demonstrate the interpretability of our model via three analyses. First, the controller can softly decompose the multi-hop question into multiple single-hop sub-questions to promote compositional reasoning behavior of the main network. Second, the controller can predict layouts that conform to the layouts designed by human experts. Finally, the intermediate module can infer the entity that connects two distantly-located supporting facts by addressing the sub-question from the controller.[1]

## 1 Introduction

The task of multi-hop question answering (QA) requires the model to answer a natural language question by finding multiple relevant information pieces scattered in a given natural language context. It has attracted more attention recently and multiple datasets have been proposed, including the recent HotpotQA (Yang et al., 2018) that is comprised of questions embodying four different multi-hop reasoning paradigms: inferring the bridge entity to complete the 2nd-hop question (first question in Fig. 1), inferring the answer through a bridge entity, checking multiple properties to select the answer, and comparing two entities (second question in Fig. 1). For the first question, it is necessary to first find the person "who portrayed Corliss Archer in the film Kiss and Tell", and then find out the "government position" she held. For the second question, one may arrive at the answer by finding the country where Scott and Ed are from, and then comparing the two nationalities to conclude whether they are the same.

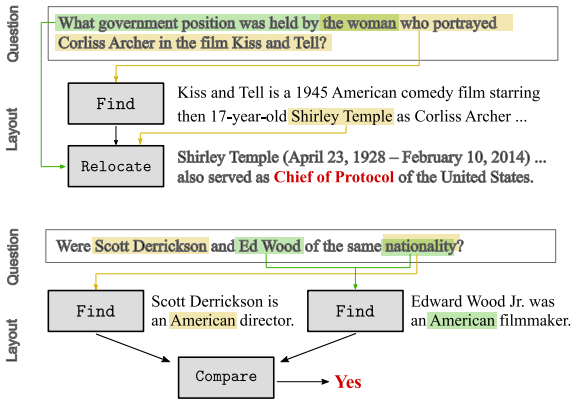


Figure 1: Two HotpotQA examples and the modular network layout predicted by the controller.

Multi-hop QA is more challenging than single-hop QA for two main reasons. First, the techniques used for single-hop QA are not sufficient to answer a multi-hop question. In single-hop QA tasks like SQuAD (Rajpurkar et al., 2016), the evidence necessary to answer the question is concentrated in a short context (Q: "What is the color of the grass", Context: "The grass is green.", An-

[1] Our code is publicly available at: https://github.com/jiangycTarheel/NMN-MultiHopQA



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4474–4484, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

swer: "Green"). Such datasets emphasize the role of matching the information between the question and the short context surrounding the answer ("sky→sky, color→blue"), and can be solved by building a question-aware context representation (Seo et al., 2017; Xiong et al., 2017). In contrast, for multi-hop QA, directly matching the semantics of the question and context leads to the entity that bridges the two supporting facts (e.g., "Shirley Temple"), or the entities that need to be compared against each other (e.g., nationalities of Scott and Ed). In both cases, further action is required on top of direct semantic matching in order to get to the final answer. Second, HotpotQA is comprised of questions of four different types of multi-hop reasoning paradigms. In Fig. 1, the first question embodies a sequential reasoning path where the model has to solve a sub-question to get an entity "Shirley Temple", which then leads to the answer to the main question about her "government position". The second question, on the other hand, requires a tree-structured reasoning path where the model first locates two entities on the leaves and then compares them to get the answer. The difference in the required reasoning skills makes it hard for a single static model to handle all types of questions.

To automatically discover the multiple elaborated reasoning steps as required in HotpotQA, a model needs to dynamically adopt a sequence of different reasoning behaviors based on specific questions, which is still unexplored in the field of large-scale text-based QA. In our work, we present a highly-interpretable self-assembling Neural Modular Network (Andreas et al., 2016a; Hu et al., 2017) with three novel modules designed for multi-hop NLP tasks: `Find`, `Relocate`, `Compare`, where each module embodies a unique type of reasoning behavior. The `Find` module is similar to the previously-introduced 1-hop bi-attention model (Seo et al., 2017; Xiong et al., 2017), which produces an attention map over the context words given the context and the question representation. For the first example in Fig. 1, a `Find` module is used to find the answer ("Shirley Temple") to the sub-question ("who portrayed Corliss Archer ..."). The `Relocate` module takes the intermediate answer to the previous sub-question ("Shirley Temple"), combines it with the current sub-question ("What government position was held by ..."), and then outputs a new attention map conditioned on the previous one. The `Compare` module intuitively "compares" the outputs from two previous modules based on the question. For the second example in Fig. 1, after previous modules find the nationality of "Scott" and "Ed" (both American), the `Compare` module outputs the answer "Yes" based on the word "same" in the question. We also use the `NoOp` module that keeps the current state of the network unchanged when the model decides no more action is needed. After all the reasoning steps are performed, the final attention map is used to predict the answer span, and the vector output from the `Compare` module is used to predict whether the answer should be a context span or "Yes/No".

Next, to dynamically assemble these modules into a network based on the specific question, we use a controller RNN that, at each step, infers the required reasoning behavior from the question, outputs the sub-question, and predicts a soft combination of all available modules. As shown in Fig. 1, a modular network `Relocate(Find())` is constructed for the bridge-type question and another network with the layout `Compare(Find(), Find())` is built for the comparison-type question. In order to make the entire model end-to-end differentiable in gradient descent, we follow Hu et al. (2018) to use continuous and soft layout prediction and maintain a differentiable stack-based data structure to store the predicted modules' output. This approach to optimize the modular network is shown to be superior to using a Reinforcement Learning approach which makes hard module decisions. Fig. 2 visualizes this controller that predicts the modular layout and the network assembled with the selected modules. We further apply intermediate supervision to the first produced attention map to encourage finding the bridge entity ("Shirley Temple" in Fig. 1). The entire model is trained end-to-end using cross entropy loss for all supervision.

Overall, our self-assembling controller-based Neural Modular Network achieves statistically significant improvements over both the single-hop bi-attention baseline and the original modular network (Hu et al., 2018) designed for visual-domain QA. We also present adversarial evaluation (Jiang and Bansal, 2019), where single-hop reasoning shortcuts are eliminated and compositional reasoning is enforced; and here our NMN again outperforms the BiDAF baseline significantly in the

EM score (as well as after adversarial training). We further demonstrate the interpretability of our modular network with three analyses. First, the controller understands the multi-hop semantics of the question and can provide accurate step-by-step sub-questions to the module to lead the main network to follow the reasoning path. Second, the controller can successfully predict layouts that conform to the layouts designed by human experts. Finally, before arriving at the final answer, the intermediate module can infer the bridge entity that connects two distantly-located supporting facts by leveraging the step-specific sub-question from the controller. All of these analyses show that our Modular Network is not operating as a black box, but demonstrates highly interpretable compositional reasoning behavior, which is beneficial in transparent model development and safe, trustworthy real-world applications. In summary, the contribution of this work is three-fold: 1) This is the first work to apply self-assembling modular networks to text-based QA; 2) We design three novel modules to handle multi-hop questions in HotpotQA; 3) The resulting network is interpretable in terms of the model's intermediate outputs and the assembled layout.

## 2 Related Works

**Multi-hop Reading Comprehension** The last few years have witnessed significant progress on large-scale QA datasets including cloze-style tasks (Hermann et al., 2015), open-domain QA (Yang et al., 2015), and more (Rajpurkar et al., 2016, 2018). However, all of the above datasets are confined to a single-document context per question domain. Joshi et al. (2017) introduced a multi-document QA dataset with some questions requiring cross-sentence inferences to answer. The bAbI dataset (Weston et al., 2016) requires the model to combine multiple pieces of evidence in the synthetic text. Welbl et al. (2017) uses Wikipedia articles as the context and a subject-relation pair as the query, and constructs the multi-hop QAngaroo dataset by traversing a directed bipartite graph so that the evidence required to answer a query could be spread across multiple documents. HotpotQA (Yang et al., 2018) is a more recent multi-hop QA dataset that has crowd-sourced questions with more diverse syntactic and semantic features compared to QAngaroo. It includes four types of questions, each requiring a different reasoning paradigm. Some examples require inferring the bridge entity from the question (Type I in Yang et al. (2018)), while others demand fact-checking or comparing subjects' properties from two different documents (Type II and comparison question). Concurrently to our work, Min et al. (2019b) also tackle HotpotQA by decomposing its multi-hop questions into single-hop sub-questions to achieve better performance and interpretability. However, their system approaches the question decomposition by having a decomposer model trained via human labels, while our controller accomplishes this task automatically with the soft attention over the question-words' representation and is only distantly supervised by the answer and bridge-entity supervision, with no extra human labels. Moreover, they propose a pipeline system with the decomposers, an answer-prediction model, and a decomposition scorer trained separately on the previous stage's output. Our modular network, on the other hand, is an end-to-end system that is optimized jointly.

**Neural Modular Network** Neural Modular Network (NMN) is a class of models that are composed of a number of sub-modules, where each sub-module is capable of performing a specific subtask. In NMN (Andreas et al., 2016b), N2NMN (Hu et al., 2017), PG+EE (Johnson et al., 2017) and TbD (Mascharka et al., 2018), the entire reasoning procedure starts by analyzing the question and decomposing the reasoning procedure into a sequence of sub-tasks, each with a corresponding module. This is done by either a parser (Andreas et al., 2016b) or a layout policy (Hu et al., 2017; Johnson et al., 2017; Mascharka et al., 2018) that turns the question into a module layout. Then the module layout is executed with a neural module network. Overall, given an input question, the layout policy learns what sub-tasks to perform, and the neural modules learn how to perform each individual sub-task.

However, since the the modular layout is sampled from the controller, the controller itself is not end-to-end differentiable and has to be optimized using Reinforcement Learning Algorithms like Reinforce (Williams, 1992). Hu et al. (2018) used soft program execution where the output of each step is the average of outputs from all modules weighted by the module distribution, and showed its superiority over hard-layout NMNs. All previous works in NMN, including Hu et al. (2018)

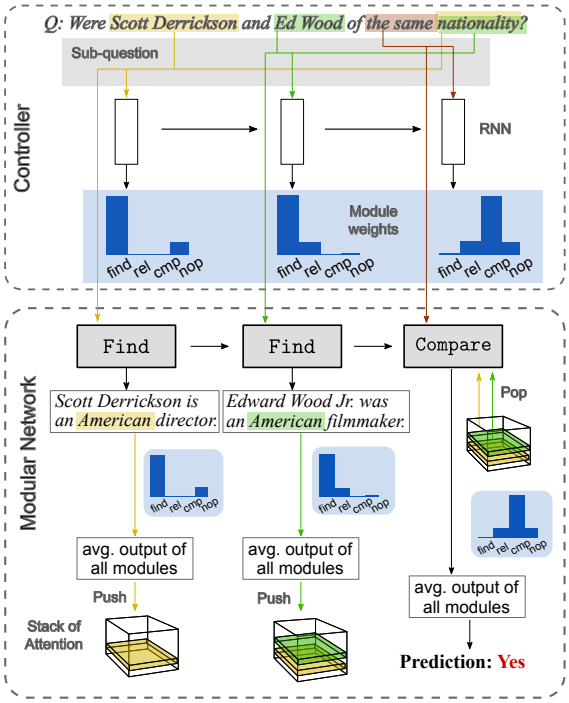


Figure 2: Modular network with a controller (top) and the dynamically-assembled modular network (bottom). At every step, the controller produces a sub-question vector and predicts a distribution to weigh the averages of the modules' outputs.

targeted visual question answering or structured knowledge-based GeoQA, and hence the modules are designed to process image or KB inputs. We are the first to apply modular networks to unstructured, text-based QA, where we redesigned the modules for language-based reasoning by using bi-attention (Seo et al., 2017; Xiong et al., 2017) to replace convolution and multiplication of the question vector with the image feature. Our model also has access to the full-sized bi-attention vector before it is projected down to the 1-d distribution.

**Architecture Learning** Our work also shares the spirit of recent research on Neural Architecture Search (NAS) (Zoph and Le, 2016; Pham et al., 2018; Liu et al., 2018), since the architecture of the model is learned dynamically based on controllers instead of being manually-designed. However, Neural Architecture Search aims to learn the structure of the individual CNN/RNN cell with fixed inter-connections between the cells, while Modular Networks have preset individual modules but learns the way to assemble them into a larger network. Moreover, Modular Networks' architectures are predicted dynamically on each data point, while previous NAS methods learn a single cell structure independent of the example.

# 3 Model

In this section, we describe how we apply the Neural Modular Network to the multi-hop HotpotQA task (Yang et al., 2018). In Sec. 3.2 and Sec. 3.3, we describe the controller, which sketches the model layout for a specific example, and introduce how it assembles the network with the predicted modules. In Sec. 3.4, we go into details of every novel module that we design for this task.

## 3.1 Encoding

We use a Highway Network (Srivastava et al., 2015) to merge the character embeddings and GloVe word embeddings (Pennington et al., 2014), building word representations for the context and the question as $\mathbf{x} \in \mathbb{R}^{S \times v}$ and $\mathbf{q} \in \mathbb{R}^{J \times v}$ respectively, where $S$ and $J$ are the lengths of the context and the question. We then apply a bi-directional LSTM-RNN (Hochreiter and Schmidhuber, 1997) of $d$ hidden units to get the contextualized word representations for the context and question: $\mathbf{h} = \text{BiLSTM}(\mathbf{x})$; $\mathbf{u} = \text{BiLSTM}(\mathbf{q})$ so that $\mathbf{h} \in \mathbb{R}^{S \times 2d}$ and $\mathbf{u} \in \mathbb{R}^{J \times 2d}$. We also use a self attention layer (Zhong et al., 2019) to get $qv$, a fixed-sized vector representation of the question.

## 3.2 Model Layout Controller

In a modular network, the controller reads the question and predicts a series of our novel modules that could be executed in order to answer the given question. For multi-hop QA, each module represents a specific 1-hop reasoning behavior and the controller needs to deduce a chain of reasoning behavior that uncovers the evidence necessary to infer the answer. Similar to the model in Hu et al. (2018), at the step $t$, the probability of selecting module $i$ is calculated as:

$$\begin{aligned} q_t &= W_{1,t} \cdot qv + b_{1,t} \\ cq_t &= W_2 \cdot [c_{t-1}; q_t] + b_2 \\ s_t &= W_3 \cdot cq_t \\ p_{t,i} &= \text{Softmax}(s_t) \end{aligned} \tag{1}$$

where $c_{t-1}$ is the controller's hidden state and $q_t$ is the vector representation of the question coming out of the encoding LSTM. $W_3$ projects the output to the dimension of $1 \times I$, where $I$ is the number of available modules. When predicting the next module, the controller also supplies the current module, which we design for the language domain, with a fixed-sized vector as the sub-question at the

current reasoning step (the gray-shaded area of the controller in Fig. 2). Consider the first example in Fig. 1. At the first reasoning step, the `Find` module's sub-question is "the woman who portrayed Corliss Archer in the film Kiss and Tell". To generate this sub-question vector, the controller first calculates a distribution over all question words and then computes the weighted average of all the word representations:

$$\begin{aligned} e_{t,j} &= W_4 \cdot (cq_t \cdot u_j) + b_4 \\ cv_t &= \text{Softmax}(e_t) \\ c_t &= \sum_j cv_{t,j} \cdot u_j \end{aligned} \tag{2}$$

The sub-question vector $c_t$ also serves as the controller's hidden state and is used to calculate the module probability in the next reasoning step. Similarly, in the second reasoning step, the sub-question of the `Relocate` module is "What government position was held by" and the answer from the `Find` module at the first step.

### 3.3 Stack NMN with Soft Program Execution

In our NMN, some modules may interact with the attention output from previous modules to collect information that could benefit the current reasoning step. For tree-structured layouts, such as `Compare(Find(), Find())` as shown in Fig. 1, the latter `Compare` module requires access to previous outputs. We follow Hu et al. (2018) to use a stack to store attention outputs generated by our modules. In the tree-structured layout, the two `Find` modules push two attention outputs onto the stack, and the `Compare` module pops these two attention outputs to compare the content they each focus on. To make the entire model differentiable, we also adopt soft program execution from Hu et al. (2018), where the output of each step is the sum of outputs from all modules weighted by the module distribution (blue-shaded area in Fig. 2) computed by the controller.

### 3.4 NMN Modules

We next describe all the modules we designed for our Neural Modular Network. All modules take the question representation $\mathbf{u}$, context representation $\mathbf{h}$, and sub-question vector $\mathbf{c_t}$ as input. The core modules that produce the attention map over the context are based on bi-attention (Seo et al., 2017) between the question and context, instead of relying on convolution as in previous NMNs (Hu et al., 2017, 2018). Every module outputs a fixed-size vector/zero vector as the memory state of the NMN, and is also able to push/pop attention maps onto/from the stack. These modules, each solving a single-hop sub-question, are chained together according to the layout predicted by the controller to infer the final answer to the multi-hop question.

**`Find`** This module performs the bidirectional attention (Seo et al., 2017) between question $\mathbf{u}$ and context $\mathbf{h'} = \mathbf{h} \cdot \mathbf{c_t}$. By multiplying the original context representation by $\mathbf{c_t}$, we inject the sub-question into the following computation. The model first computes a similarity matrix $M^{J \times S}$ between every question and context word and use it to derive context-to-query attention:

$$\begin{aligned} M_{j,s} &= W_1 \mathbf{u}_j + W_2 \mathbf{h'}_s + W_3 (\mathbf{u}_j \odot \mathbf{h'}_s) \\ p_{j,s} &= \frac{\exp(M_{j,s})}{\sum_{j=1}^{J} \exp(M_{j,s})} \\ \mathbf{c_{q}}_s &= \sum_{j=1}^{J} p_{j,s} \mathbf{u}_j \end{aligned} \tag{3}$$

where $W_1, W_2$ and $W_3$ are trainable parameters, and $\odot$ is element-wise multiplication. Then the query-to-context attention vector is derived as:

$$\begin{aligned} m_s &= \max_{1 \le s \le S} M_{j,s} \\ p_s &= \frac{\exp(m_s)}{\sum_{s=1}^{S} \exp(m_s)} \\ \mathbf{q_c} &= \sum_{s=1}^{S} p_s \mathbf{h}_S \end{aligned} \tag{4}$$

We then obtain the question-aware context representation as: $\widetilde{\mathbf{h}} = [\mathbf{h'}; \mathbf{c_q}; \mathbf{h'} \odot \mathbf{c_q}; \mathbf{c_q} \odot \mathbf{q_c}]$, and push this bi-attention result onto the stack. This process is visualized in the first two steps in Fig. 2.

**`Relocate`** For the first example in Fig. 1, the second reasoning step requires finding the "government position" held by the woman, who is identified in the first step. We propose the `Relocate` module to model this reasoning behavior of finding the answer based on the information from the question as well as the previous reasoning step. The `Relocate` module first pops an attention map $att_1$ from the stack and computes the bridge entity's representation $\mathbf{b}$ as the weighted average over $\mathbf{h}$ using the popped attention. It then computes a bridge-entity-awared representation of context $\mathbf{h_b}$, and then applies a `Find`

module between $\mathbf{h_b}$ and question $\mathbf{u}$:

$$\begin{aligned} \mathbf{b} &= \sum_s att_{1,s} \cdot \mathbf{h}_s \\ \mathbf{h_b} &= \mathbf{h} \odot \mathbf{b} \\ \widetilde{\mathbf{h}} &= \texttt{Find}(\mathbf{u}, \mathbf{h_b}, \mathbf{c_t}) \end{aligned} \tag{5}$$

The output $\widetilde{\mathbf{h}}$ is then pushed onto the stack.

**`Compare`** As shown in the final step in Fig. 2, the `Compare` module pops two attention maps $att_1$ and $att_2$ from the stack and computes two weighted averages over $\mathbf{h}$ using the attention maps. It then merges these two vectors with $c_t$ to update the NMN's memory state $m$.

$$\begin{aligned} hs_1 = \sum_s att_{1,s} \cdot \mathbf{h}_s; \quad hs_2 = \sum_s att_{2,s} \cdot \mathbf{h}_s \\ o_{in} = [c_t; hs_1; hs_2; c_t \cdot (hs_1 - hs_2)] \\ m = W_1 \cdot (\text{ReLU}(W_2 \cdot o_{in})) \end{aligned} \tag{6}$$

The intuition is that the `Compare` module aggregates the information from two attention maps and compares them according to the sub-question $c_t$.

**`NoOp`** This module updates neither the stack nor the memory state. It can be seen as a skip command when the controller decides no more computation is required for the current example.

### 3.5 Bridge Entity Supervision

Even with the multi-hop architecture to capture the hop-specific distribution over the question, there is no supervision on the controller's output distribution $c$ about which part of the question is related to the current reasoning step, thus preventing the controller from learning the composite reasoning skill. To address this problem, we supervise the first `Find` module to predict the bridge entity ("Shirley Temple" in Fig. 1), which indirectly encourages the controller to look for question information related to this entity ("the woman who portrayed Corliss Archer...") at the first step. Since the original dataset does not label the bridge entity, we apply a simple heuristic to detect the bridge entity that is the title of one supporting document while also appearing in the other supporting document.[2]

### 3.6 Prediction Layer

After all the reasoning steps have completed, to predict a span from the context as the answer, we pop the top attention map from the stack, apply self-attention (Yang et al., 2018), and project it down to get the span's start index and end index. To predict yes/no, we take the final memory ($m$ in Eqn. 6) and project it down to a 2-way classifier. We concatenate the question vector $qv$ and memory $m$ and then project down to a 2-way classifier to decide whether to output a span or yes/no.

### 3.7 Optimization

Previous works on modular network (Hu et al., 2017, 2018) optimize the controller parameters $\theta$ jointly with the modules parameters $\omega$ on the training set. However, we found that our controller converges[3] in less than 20 iterations under this training routine. It is also likely that this setup causes the model to overfit to the single-hop reasoning shortcuts in the training set. Hence, to prevent such premature convergence and reasoning-shortcut overfitting, we adopt the 2-phase training strategy that is widely used in Neural Architecture Search (Zoph and Le, 2016; Pham et al., 2018). The first phase updates $\omega$ for an entire epoch, followed by the second phase that updates $\theta$ over the entire dev set. We alternate between the two phases until the entire system converges.

## 4 Experimental Setup

**Dataset and Metric** We conduct our training and evaluation on the HotpotQA (Yang et al., 2018) dataset's distractor setting, which has 90,447 training examples (72991 bridge-type questions and 17456 comparison-type questions) and 7,405 dev examples (5918 bridge-type questions and 1487 comparison-type questions). Since there is no public test set available, we split the original dev set into a dev set of 3,702 examples on which we tune our models, and a test set of 3,703 examples that is only used for the final test results. Each question is paired with 10 documents, among which 2 are supporting documents that contain the evidence necessary to infer the answer, and the other 8 are distracting documents. We evaluate our models based on the exact-match and F1 score between the prediction and ground-truth answer.

**Data Augmentation** We augment the comparison-type questions in the training set to promote the model's robustness on this reasoning paradigm. We start with getting the

[2] There are two supporting documents per example.

[3] Generating modular layout with probability near 1.

| ORIGINAL question | answer | MUTATED question | answer |
|---|---|---|---|
| Were Pavel Urysohn and Levin known for the **same** type of work? | No | Were Henry Cavill and Li Na known for the **same** type of work? | Yes |
| Were Pavel Urysohn and Levin known for the **same** type of work? | No | Were Pavel Urysohn and Levin known for the **different** type of work? | Yes |
| Is Rohan Bopanna **older** than Sherwood Stewart? | Yes | Is Rohan Bopanna **younger** than Sherwood Stewart? | No |
| Was Howard Hawks a screenwriter of more productions than Arthur Berthelet? | Yes | Was Arthur Berthelet a screenwriter of more productions than Howard Hawks? | No |

Table 1: Examples of mutated comparison-type questions and answers from HotpotQA training set.

| | Dev | | Test | |
|---|---|---|---|---|
| | EM | F1 | EM | F1 |
| BiDAF Baseline | 44.68 | 57.19 | 42.7 | 55.81 |
| NMN | 31.04 | 40.28 | 30.87 | 39.90 |
| Our NMN | **50.67** | **63.35** | **49.58** | **62.71** |
| + Data aug. | 50.63 | 63.29 | 49.46 | 62.59 |
| - Bridge sup. | 46.56 | 58.60 | 45.91 | 57.22 |
| - Relocate | 47.81 | 60.22 | 46.75 | 59.23 |
| - Compare | 50.29 | 63.30 | 48.87 | 62.52 |
| - NoOp | 49.11 | 61.79 | 48.56 | 62.10 |

Table 2: EM and F1 scores on HotpotQA dev set and test set. All models are tuned on dev set.

part-of-speech tags and NER tag of the question using Corenlp (Manning et al., 2014). For each question whose answer is either "yes" or "no", we generate a new question by randomly sampling two titles of the same type (based on POS and NER) from the training set to substitute the original entities in the question and corresponding supporting documents (1st row in Table 1). We then employ three strategies to generate 5,342 extra questions by mutating original questions. First, if the question contains the word "same" and the answer is yes or no, we substitute "same" with "different" and vice versa (2nd row in Table 1). Second, we detect the comparative and superlative adjectives/adverbs in the original question, transform them into their antonyms using wordnet, and then transform the antonyms back to their comparative form (3rd row in Table 1). Finally, if the question has a comparative adj./adv., we flip the order of the two entities compared (4th row in Table 1). In all three cases, the answer to the mutated question is also flipped.

**Training Details** We use 300-d GloVe pre-trained embeddings (Pennington et al., 2014). The model is supervised to predict either the start and end index of a span or "Yes/No" for specific questions. The entire model (controller + modular network) is trained end-to-end using the Adam optimizer (Kingma and Ba, 2014) with an initial learning rate of 0.001.

## 5 Results

### 5.1 Baseline

Our baseline is the bi-attention (Seo et al., 2017) + self-attention model as in (Clark and Gardner, 2018; Yang et al., 2018), which was shown to be able to achieve strong performance on single-hop QA tasks like SQuAD (Rajpurkar et al., 2016) and TriviaQA (Joshi et al., 2017). Our baseline shares the preprocessing and encoding layer with the modular network.

### 5.2 Primary NMN Results

We first present our model's performance on the HotpotQA (Yang et al., 2018) dev and test set of our split. As shown in the first three rows of Table 2, our modular network achieves significant improvements over both the baseline and the convolution-based NMN (Hu et al., 2018) on our test set. In Table 3, we further break down the dev-set performance on different question types,[4] and our modular network obtains higher scores in both question types compared to the BiDAF baseline.

### 5.3 Ablation Studies

**Data Augmentation:** We also conduct ablation studies on our modeling choices. Augmenting the comparison-type questions in the training set boosts the performance on the comparison-type questions in the dev set (comparing row 3 and 4 in Table 3) without harming the scores on the bridge-type questions too much.

**Bridge-entity Supervision:** Supervising the bridge entity (described in 3.5) proves to be

[4]The 3 types of question other than "comparison" are all classified as "bridge-type" questions by Yang et al. (2018)

| | Bridge | | Compare | |
|---|---|---|---|---|
| | EM | F1 | EM | F1 |
| BiDAF Baseline | 43.17 | 57.74 | 45.26 | 51.73 |
| NMN | 30.15 | 41.49 | 34.27 | 40.07 |
| Our NMN | **49.85** | **64.49** | 51.24 | 57.20 |
| + Data aug. | 48.97 | 63.85 | **54.40** | **60.03** |
| - Bridge sup. | 45.59 | 59.45 | 48.82 | 51.73 |
| - Relocate | 46.11 | 60.13 | 51.92 | 58.10 |
| - Compare | 49.37 | 64.46 | 49.76 | 56.00 |
| - NoOp | 48.46 | 63.46 | 49.63 | 55.26 |

Table 3: EM and F1 scores on bridge-type and comparison-type questions from HotpotQA dev set.

| Train<br>Eval | Reg<br>Reg | Reg<br>Adv | Adv<br>Reg | Adv<br>Adv |
|---|---|---|---|---|
| BiDAF Baseline | 43.12 | 34.00 | 45.12 | 44.65 |
| Our NMN | **50.13** | **44.70** | **49.33** | **49.25** |

Table 4: EM scores after training on the regular data or on the adversarial data from Jiang and Bansal (2019), and evaluation on the regular dev set or the adv-dev set.

beneficial for the modular network to achieve good performance (row 5 in Table 2 and Table 3).

**Modules:** As shown in the 6th to 8th row, removing either the `Compare`, `Relocate` or `NoOp` module also causes drops in the metric scores. Specifically, removing the `Relocate` module causes significant degrade in bridge-type questions, which solidifies our claim that relocating the attention based on the inferred bridge entity is important for compositional reasoning. Similarly, removing the `Compare` module harms the model's performance on comparison-type questions, suggesting the effectiveness of the module in addressing questions that require comparing two entities' properties. These results demonstrate the contribution of each module toward achieving a self-assembling modular network with the strong overall performance.

### 5.4 Comparison with Original NMN Modules

One primary contribution of this work is that we adapt neural modular networks (NMN) (Andreas et al., 2016a; Hu et al., 2017, 2018), which were designed for visual-domain QA, to text-domain QA by rebuilding every reasoning module. We substitute convolution and multiplication between question vectors and context features with bi-attention as the basic reasoning component in the `Find` and `Relocate`. Moreover, our model maintains a stack of attention outputs before it is projected down to 1-d, thus enabling skip connections when predicting the answer span. As shown in Table 2 and Table 3, our adapted modular network outperforms the original NMN significantly.

### 5.5 Adversarial Evaluation

Multiple previous works (Chen and Durrett, 2019; Min et al., 2019a) have shown that models performing strongly on HotpotQA are not necessarily capable of compositional reasoning. Jiang and Bansal (2019) proposed to construct adversarial distractor documents to eliminate the reasoning shortcut and necessitate compositional reasoning on HotpotQA dataset. To test whether our modular network can perform robust multi-hop reasoning against such adversaries, we evaluate our models on the adversarial dev set. The second column of Table 4 shows that our NMN outperforms the baseline significantly (+10 points in EM score) on the adversarial evaluation, suggesting that our NMN is indeed learning stronger compositional reasoning skills compared to the BiDAF baseline. We further train both models on the adversarial training set, and the results are shown in the last two columns of Table 4. We observe that after adversarial training, both the baseline and our NMN obtain significant improvement on the adversarial evaluation, while our NMN maintains a significant advantage over the BiDAF baseline.

## 6 Analysis

In this section, we present three analysis methods to show that our multi-hop NMN is highly interpretable and makes decisions similar to human actions. Combining the analyses below, we can understand how our model reasons to break the multi-step task into multiple single-step subtasks that are solvable by specific modules, and harnesses intermediate outputs to infer the answer.

### 6.1 Controller's Attention on Questions

Different from single-hop questions, a multi-hop question usually encodes a sequence of sub-questions, among which the final sub-question requires an explicit answer. Therefore, the first step of solving a multi-hop question is to identify the hidden sub-questions and to sort them according to the correct reasoning order. Our controller handles this task by computing an attention distribution over all question words (see details in Sec. 3.2) at every reasoning step, which signifies which part of the question forms a sub-question that should be answered at the current step. Fig. 3 visualizes four

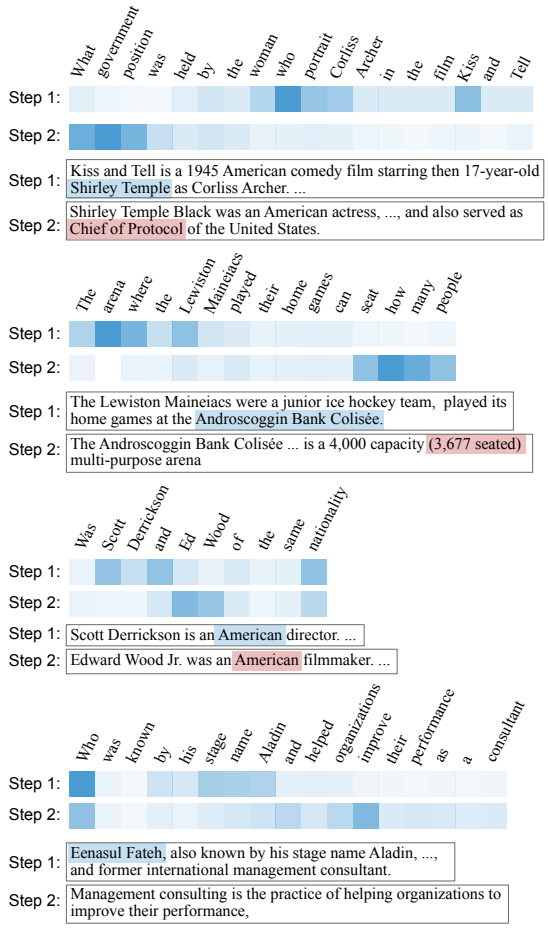


Figure 3: The controller's multi-hop attention map on the question (randomly chosen from the first 10 dev examples). This attention is used to compute the sub-question representation that is passed to the modules.

attention maps from the controller when dealing with questions of different reasoning paradigms (randomly chosen from the first 10 dev examples), suggesting that our controller is able to accurately identify the sub-questions in the correct order. In the first example, the controller is able to attend to the first sub-question "who portrayed Corliss Archer in ...", and then relocate the attention to the next sub-question "What government position was held by ..." to complete the 2-step reasoning.

### 6.2 Predicting Network Layouts

After the controller decides the sub-question at the current reasoning step, it then predicts a soft layout of the the current step by outputting a distribution $cv$ over all available modules (`Find`, `Relocate`, `Compare`, `NoOp`). As elaborated in Sec. 3.3, the output vector and attention map of all modules are weighted by $cv$. To quantitatively evaluate the layout predicted by the controller, we label the expert layout for each question in the dev set (`Find-Relocate` for bridge-type questions and `Find-Find/Relocate-Compare` for comparison-type questions). Next, we convert the soft module predictions into hard modular layouts by picking the module with the largest prediction probability at each step. Intuitively, we want the controller to assign higher weights to the module that can perform the desired reasoning. We then compute the percentage of dev set examples where the predicted layout matches the expert layout exactly. The empirical study shows that the layouts predicted by our NMN controller match the expert layout in 99.9% of bridge-type questions and 68.6% of comparison-type questions with Yes/No answer (examples shown in Fig. 1 and Fig. 2).

### 6.3 Finding Intermediate Bridge-Entities

The final action of an intermediate reasoning step is the predicted module reasoning through the question and context to answer the sub-question. For bridge-type questions (e.g., the first example in Fig. 3), the required action is to find the name ("Shirley Temple") of "the woman who portrait Corliss Archer ..." to complete the next sub-question "What government position was held by Shirley Temple". For comparison questions (e.g., the third example in Fig. 3), the model needs to infer the nationalities of "Scott" and "Ed" that will be compared in the following step.

## 7 Conclusion

In this work, we proposed a self-assembling neural modular network for multi-hop QA. We designed three modules that reason between the question and text-based context. The resulting model outperforms both the single-hop baseline and the original NMN on HotpotQA (Yang et al., 2018). Because of the interpretable nature of our model, we presented analyses to show that our model does in fact learn to perform compositional reasoning and can dynamically assemble the modular network based on the question.

## Acknowledgments

We thank the reviewers for their helpful comments. This work was supported by ARO-YIP Award #W911NF-18-1-0336, DARPA #YFA17-D17AP00022 and awards from Google, Facebook, Salesforce, and Adobe (plus Amazon and Google GPU cloud credits). The views are those of the authors and not of the funding agency.

## References


Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016a. Learning to compose neural networks for question answering. *arXiv preprint arXiv:1601.01705*.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016b. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 39–48.

Jifan Chen and Greg Durrett. 2019. Understanding dataset design choices for multi-hop reasoning. In *NAACL*.

Christopher Clark and Matt Gardner. 2018. Simple and effective multi-paragraph reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Ronghang Hu, Jacob Andreas, Trevor Darrell, and Kate Saenko. 2018. Explainable neural computation via stack neural module networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 53–69.

Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. 2017. Learning to reason: End-to-end module networks for visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 804–813.

Yichen Jiang and Mohit Bansal. 2019. Avoiding reasoning shortcuts: Adversarial evaluation, training, and model development for multi-hop qa. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Judy Hoffman, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017. Inferring and executing programs for visual reasoning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2989–2998.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2018. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

David Mascharka, Philip Tran, Ryan Soklaski, and Arjun Majumdar. 2018. Transparency by design: Closing the gap between performance and interpretability in visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4942–4950.

Sewon Min, Eric Wallace, Sameer Singh, Matt Gardner, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2019a. Compositional questions do not necessitate multi-hop reasoning. In *ACL*.

Sewon Min, Victor Zhong, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2019b. Multi-hop reading comprehension through question decomposition and rescoring. In *ACL*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. 2018. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*.

P. Rajpurkar, R. Jia, and P. Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Association for Computational Linguistics (ACL)*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *International Conference on Learning Representations (ICLR)*.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. In *International Conference on Machine Learning (ICML)*.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2017. Constructing datasets for multi-hop reading comprehension across documents. In *TACL*.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2016. Towards ai-complete question answering: A set of prerequisite toy tasks. In *ICLR*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Caiming Xiong, Victor Zhong, and Richard Socher. 2017. Dynamic coattention networks for question answering. In *ICLR*.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Victor Zhong, Caiming Xiong, Nitish Shirish Keskar, and Richard Socher. 2019. Coarse-grain fine-grain coattention network for multi-evidence question answering. In *ICLR*.

Barret Zoph and Quoc V Le. 2016. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*.

# Posing Fair Generalization Tasks for Natural Language Inference

**Atticus Geiger**
Stanford Symbolic Systems Program
atticusg@stanford.edu

**Ignacio Cases**
Stanford Linguistics
cases@stanford.edu

**Lauri Karttunen**
Stanford Linguistics
laurik@stanford.edu

**Christopher Potts**
Stanford Linguistics
cgpotts@stanford.edu

## Abstract

Deep learning models for semantics are generally evaluated using naturalistic corpora. Adversarial methods, in which models are evaluated on new examples with known semantic properties, have begun to reveal that good performance at these naturalistic tasks can hide serious shortcomings. However, we should insist that these evaluations be *fair* – that the models are given data sufficient to support the requisite kinds of generalization. In this paper, we define and motivate a formal notion of fairness in this sense. We then apply these ideas to natural language inference by constructing very challenging but provably fair artificial datasets and showing that standard neural models fail to generalize in the required ways; only task-specific models that jointly compose the premise and hypothesis are able to achieve high performance, and even these models do not solve the task perfectly.

## 1 Introduction

Evaluations of deep learning approaches to semantics generally rely on corpora of naturalistic examples, with quantitative metrics serving as a proxy for the underlying capacity of the models to learn rich meaning representations and find generalized solutions. From this perspective, when a model achieves human-level performance on a task according to a chosen metric, one might be tempted to say that the task is "solved". However, recent adversarial testing methods, in which models are evaluated on new examples with known semantic properties, have begun to reveal that even these state-of-the-art models often rely on brittle, local solutions that fail to generalize even to examples that are similar to those they saw in training. These findings indicate that we need a broad and deep range of evaluation methods to fully characterize the capacities of our models.

However, for any evaluation method, we should ask whether it is *fair*. Has the model been shown data sufficient to support the kind of generalization we are asking of it? Unless we can say "yes" with complete certainty, we can't be sure whether a failed evaluation traces to a model limitation or a data limitation that no model could overcome.

In this paper, we seek to address this issue by defining a formal notion of fairness for these evaluations. The definition is quite general and can be used to create fair evaluations for a wide range of tasks. We apply it to Natural Language Inference (NLI) by constructing very challenging but provably fair artificial datasets. We evaluate a number of different standard architectures (variants of LSTM sequence models with attention and tree-structured neural networks) as well as NLI-specific tree-structured neural networks that process aligned examples. Our central finding is that only task-specific models are able to achieve high performance, and even these models do not solve the task perfectly, calling into question the viability of the standard models for semantics.

## 2 Related Work

There is a growing literature that uses targeted generalization tasks to probe the capacity of learning models. We seek to build on this work by developing a formal framework in which one can ask whether one of these tasks is even possible.

In adversarial testing, training examples are systematically perturbed and then used for testing. In computer vision, it is common to adversarially train on artificially noisy examples to create a more robust model (Goodfellow et al., 2015; Szegedy et al., 2014). However, in the case of question answering, Jia and Liang (2017) show that training on one perturbation does not result in generalization to similar perturbations, revealing a



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4485–4495,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

need for models with stronger generalization capabilities. Similarly, adversarial testing has shown that strong models for the SNLI dataset (Bowman et al., 2015a) have significant holes in their knowledge of lexical and compositional semantics (Glockner et al., 2018; Naik et al., 2018; Nie et al., 2018; Yanaka et al., 2019; Dasgupta et al., 2018). In addition, a number of recent papers suggest that even top models exploit dataset artifacts to achieve good quantitative results (Poliak et al., 2018; Gururangan et al., 2018; Tsuchiya, 2018), which further emphasizes the need to go beyond naturalistic evaluations.

Artificially generated datasets have also been used extensively to gain analytic insights into what models are learning. These methods have the advantage that the complexity of individual examples can be precisely characterized without reference to the models being evaluated. Evans et al. (2018) assess the ability of neural models to learn propositional logic entailment. Bowman et al. (2015b) conduct similar experiments using natural logic, and Veldhoen and Zuidema (2018) analyze models trained on those same tasks, arguing that they fail to discover the kind of global solution we would expect if they had truly learned natural logic. Lake and Baroni (2017) apply similar methods to instruction following with an artificial language describing a simple domain.

These methods can provide powerful insights, but the issue of fairness looms large. For instance, Bowman (2013) poses generalization tasks in which entire reasoning patterns are held out for testing. Similarly, Veldhoen and Zuidema (2018) assess a model's ability to recognize De Morgan's laws without any exposure to this reasoning in training. These extremely difficult tasks break from standard evaluations in an attempt to expose model limitations. However, these tasks are not fair by our standards; brief formal arguments for these claims are given in Appendix A.

## 3 Compositionality and Generalization

Many problems can be solved by recursively composing intermediate representations with functions along a tree structure. In the case of arithmetic, the intermediate representations are numbers and the functions are operators such a plus or minus. In the case of evaluating the truth of propositional logic sentences, the intermediate representation are truth values and the functions are logical operators such as disjunction, negation, or the material conditional. We will soon see that, in the case of NLI, the intermediate representations are semantic relations between phrases and the functions are semantic operators such as quantifiers or negation. When tasked with learning some compositional problem, we intuitively would expect to be shown how every function operates on every intermediate value. Otherwise, some functions would be underdetermined. We now formalize the idea of recursive tree-structured composition and this intuitive notion of fairness.

**Data:** A composition tree $C = (T, \mathit{Dom}, \mathit{Func})$, a node $a \in N^T$, and an input $x \in \mathcal{I}_C$
**Result:** An output from $\mathit{Dom}(a)$

**function** $\texttt{compose}\,(C, a, x)$
  **if** $a \in N^T_{\mathit{leaf}}$ **then**
    $i \leftarrow \texttt{index}(a, T)$
    **return** $x_i$
  **else**
    $c_1, \ldots c_m \leftarrow \texttt{children}(a, T)$
    **return** $\mathit{Func}(a)($
      $\texttt{compose}(C, c_1, x), \ldots,$
      $\texttt{compose}(C, c_m, x))$
  **end**

**Algorithm 1:** Recursive composition up a tree. This algorithm uses helper functions $\texttt{children}(a, T)$, which returns the left-to-right ordered children of node $a$, and $\texttt{index}(a, T)$, which returns the index of a leaf according to left-to-right ordering.

We first define composition trees (Section 3.1) and show how these naturally determine baseline learning models (Section 3.2). These models implicitly define a property of fairness: a train/test split is fair if the baseline model learns the task perfectly (Section 3.3). This enables us to create provably fair NLI tasks in Section 4.

### 3.1 Composition Trees

A composition tree describes how to recursively compose elements from an input space up a tree structure to produce an element in an output space. Our baseline learning model will construct a composition tree using training data.

**Definition 1.** (Composition Tree) Let $T$ be an ordered tree with nodes $N^T = N^T_{\text{leaf}} \cup N^T_{\text{non-leaf}}$, where $N^T_{\text{leaf}}$ is the set of leaf nodes and $N^T_{\text{non-leaf}}$ is the set of non-leaf nodes for $T$. Let *Dom* be a map on $N^T$ that assigns a set to each node, called the *domain* of the node. Let *Func* be a map on

**Data:** An ordered tree $T$ and a set of training data $\mathcal{D}$ containing pairs $(x, Y)$ where $x$ is an input and $Y$ is a function defined on $N^T_{\text{non-leaf}}$ providing labels at every node of $T$.
**Result:** A composition tree $(T, \mathit{Dom}, \mathit{Func})$

**function** `learn` $(T, \mathcal{D})$
  $\mathit{Dom}, \mathit{Func} \leftarrow$ `initialize`$(T)$
  **for** $(x, Y) \in \mathcal{D}$ **do**
    $\mathit{Dom}, \mathit{Func} \leftarrow$ `memorize`$(x, Y, T, \mathit{Dom}, \mathit{Func}, r)$
  **end**
  **return** $(T, \mathit{Dom}, \mathit{Func})$

**function** `memorize` *(x,Y,T,Dom,Func,a)*
  **if** $a \in N^T_{\mathit{leaf}}$ **then**
    $i \leftarrow$ `index`$(a, T)$
    $\mathit{Dom}[a] \leftarrow \mathit{Dom}[a] \cup \{x_i\}$
    **return** *Dom*, *Func*
  **else**
    $\mathit{Dom}[a] \leftarrow \mathit{Dom}[a] \cup \{Y(a)\}$
    $c_1, \ldots, c_m \leftarrow$ `children`$(a, T)$
    $\mathit{Func}[a][(Y(c_1), \ldots, Y(c_m))] \leftarrow Y(a)$
    **for** $k \leftarrow 1 \ldots m$ **do**
      $\mathit{Dom}, \mathit{Func} \leftarrow$ `memorize`$(x, Y, T, \mathit{Dom}, \mathit{Func}, c_k)$
    **end**
    **return** *Dom*, *Func*
  **end**

**Algorithm 2:** Given a tree and training data with labels for every node of the tree, this learning model constructs a composition tree. This algorithm uses helper functions `children`$(a, T)$ and `index`$(a, T)$, as in Algorithm 1, as well as `initialize`$(T)$, which returns *Dom*, a dictionary mapping $N^T$ to empty sets, and *Func*, a dictionary mapping $N^T_{\text{non-leaf}}$ to empty dictionaries.

$N^T_{\text{non-leaf}}$ that assigns a function to each non-leaf node satisfying the following property: For any $a \in N^T_{\text{non-leaf}}$ with left-to-right ordered children $c_1, \ldots, c_m$, we have that $\mathit{Func}(a) : \mathit{Dom}(c_1) \times \cdots \times \mathit{Dom}(c_m) \to \mathit{Dom}(a)$. We refer to the tuple $C = (T, \mathit{Dom}, \mathit{Func})$ as a *composition tree*. The *input space* of this composition tree is the cartesian product $\mathcal{I}_C = \mathit{Dom}(l_1) \times \cdots \times \mathit{Dom}(l_k)$, where $l_1, \ldots, l_k$ are the leaf nodes in left-to-right order, and the *output space* is $\mathcal{O}_C = \mathit{Dom}(r)$ where $r$ is the root node.

A composition tree $C = (T, \mathit{Dom}, \mathit{Func})$ realizes a function $F : \mathcal{I}_C \to \mathcal{O}_C$ in the following way: For any input $x \in \mathcal{I}_C$, this function is given by $F(x) = \texttt{compose}(C, r, x)$, where $r$ is the root node of $T$ and `compose` is defined recursively in Algorithm 1. For a given $x \in \mathcal{I}_C$ and $a \in N^T_{\text{non-leaf}}$ with children $c_1, \ldots, c_m$, we say that the element of $\mathit{Dom}(c_1) \times \cdots \times \mathit{Dom}(c_m)$ that is input to $\mathit{Func}(a)$ during the computation of $F(x) = \texttt{compose}(C, r, x)$ is the input realized at $\mathit{Func}(a)$ on $x$ and the element of $\mathit{Dom}(a)$ that is output by $\mathit{Func}(a)$ is the output realized at $\mathit{Func}(a)$ on $x$. At a high level, $\texttt{compose}(C, a, x)$ finds the output realized at a node $a$ by computing node $a$'s function $\mathit{Func}(a)$ with the outputs realized at node $a$'s children as inputs. This recursion bottoms out when the components of $x$ are provided as the outputs realized at leaf nodes.

### 3.2 A Baseline Learning Model

Algorithm 2 is our baseline learning model. It learns a function by constructing a composition tree. This is equivalent to learning the function that tree realizes, as once the composition tree is created, Algorithm 1 computes the realized function. Because this model constructs a composition tree, it has an inductive bias to recursively compute intermediate representations up a tree structure. At a high level, it constructs a full composition tree when provided with the tree structure and training data that provides a label at every node in the tree by looping through training data inputs and memorizing the output realized at each intermediate function for a given input. As such, any learning model we compare to this baseline model should be provided with the outputs realized at every node during training.

### 3.3 Fairness

We define a training dataset to be fair with respect to some function $F$ if our baseline model perfectly learns the function $F$ from that training data. The guiding idea behind fairness is that the training data must expose every intermediate function of a composition tree to every possible intermediate input, allowing the baseline model to learn a global solution:

**Definition 2.** (A Property Sufficient for Fairness) A property of a training dataset $\mathcal{D}$ and tree $T$ that is sufficient for fairness with respect to a function $F$ is that there exists a composition tree $C = (T, \mathit{Dom}, \mathit{Func})$ realizing $F$ such that, for any $a \in N^T_{\text{non-leaf}}$ and for any input $i$ to $\mathit{Func}(a)$, there exists $(x, Y) \in \mathcal{D}$ where $i$ is the input realized at $\mathit{Func}(a)$ on $x$.

Not all fair datasets are challenging. For example, a scenario in which one trains and tests on the entire space of examples will be fair. The role of fairness is to ensure that, when we separately de-

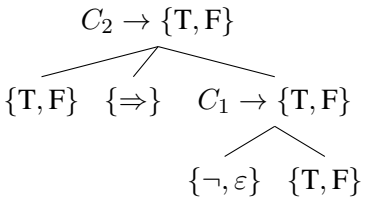


Figure 1: A composition tree that realizes a function evaluating propositional sentences. We define the functions $C_1(U, V_1) = U(V_1)$ and $C_2(V_1, \Rightarrow, V_2) = V_1 \Rightarrow V_2$ where $V_1, V_2 \in \{\mathrm{T}, \mathrm{F}\}$ and $U \in \{\neg, \varepsilon\}$.

| **Train** | **Test** |
|---|---|
| T $\Rightarrow$ $\varepsilon$ F | T $\Rightarrow$ $\neg$ T |
| T $\Rightarrow$ $\neg$ F | T $\Rightarrow$ $\varepsilon$ T |
| F $\Rightarrow$ $\neg$ T | F $\Rightarrow$ $\neg$ F |
| F $\Rightarrow$ $\varepsilon$ T | F $\Rightarrow$ $\varepsilon$ F |

Table 1: A fair train/test split for the evaluation problem defined by the composition tree in Figure 1. We give just the terminal nodes; the examples are full trees.

fine a challenging task, it is guaranteed to be possible. We noted in Section 2 that some challenging problems in the literature fail to meet this minimal requirement.

### 3.4 Fair Tasks for Propositional Evaluation

As a simple illustration of the above concepts, we consider the task of evaluating the truth of a sentence from propositional logic. We use the standard logical operators material conditional, $\Rightarrow$, and negation, $\neg$, as well as the unary operator $\varepsilon$, which we define to be the identity function on $\{\mathrm{T}, \mathrm{F}\}$. We consider a small set of eight propositional sentences, all which can be seen in Table 1. We illustrate a composition tree that realizes a function performing truth evaluation on these sentences in Figure 1, where a leaf node $l$ is labeled with its domain $\mathit{Dom}(l)$ and a non-leaf node $a$ is labeled with $\mathit{Func}(a) \rightarrow \mathit{Dom}(a)$.

A dataset for this problem is fair if and only if it has two specific properties. First, the binary operator $\Rightarrow$ must be exposed to all four inputs in $\{\mathrm{T}, \mathrm{F}\} \times \{\mathrm{T}, \mathrm{F}\}$ during training. Second, the unary operators $\neg$ and $\varepsilon$ each must be exposed to both inputs in $\{\mathrm{T}, \mathrm{F}\}$. Jointly, these constraints ensure that a model will see all the possibilities for how our logical operators interact with their truth-value arguments. If either constraint is not met, then there is ambiguity about which operators the model is tasked with learning. An example fair train/test split is given in Table 1.

Crucial to our ability to create a fair training dataset using only four of the eight sentences is that $\Rightarrow$ operates on the intermediate representation of a truth value, abstracting away from the specific identity of its sentence arguments. Because there are two ways to realize T and F at the intermediate node, we can efficiently use only half of our sentences to satisfy our fairness property.

| symbol | example | set theoretic definition |
|---|---|---|
| $x \equiv y$ | couch $\equiv$ sofa | $x = y$ |
| $x \sqsubset y$ | crow $\sqsubset$ bird | $x \subset y$ |
| $x \sqsupset y$ | bird $\sqsupset$ crow | $x \supset y$ |
| $x \,^\wedge\, y$ | human $^\wedge$ nonhuman | $x \cap y = \emptyset \wedge x \cup y = U$ |
| $x \mid y$ | cat $\mid$ dog | $x \cap y = \emptyset \wedge x \cup y \neq U$ |
| $x \smile y$ | animal $\smile$ nonhuman | $x \cap y \neq \emptyset \wedge x \cup y = U$ |
| $x \# y$ | hungry # hippo | (all other cases) |

Table 2: The seven basic semantic relations of MacCartney and Manning (2009): $\mathcal{B}$ = {# = independence, $\sqsubset$ = entailment, $\sqsupset$ = reverse entailment, $\mid$ = alternation, $\smile$ = cover, $^\wedge$ = negation, $\equiv$ = equivalence}.

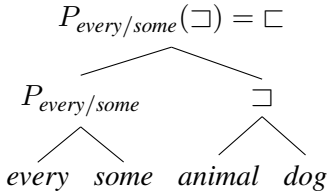


Figure 2: Natural logic inference cast as composition on aligned semantic parse trees. The joint projectivity signature $P_{every/some}$ operates on the semantic relation $\sqsupset$ determined by the aligned pair *animal/dog* to determine entailment ($\sqsubset$) for the whole. In contrast, if we reverse *every* and *some*, creating the example *some animal/every dog*, then the joint projectivity signature $P_{some/every}$ operates on $\sqsupset$, which determines reverse entailment ($\sqsupset$).

## 4 Fair Artificial NLI Datasets

Our central empirical question is whether current neural models can learn to do robust natural language inference if given fair datasets. We now present a method for addressing this question. To do this, we need to move beyond the simple propositional logic example explored above, to come closer to the true complexity of natural language. To do this, we adopt a variant of the *natural logic* developed by MacCartney and Manning (2007, 2009) (see also Sánchez-Valencia 1991; van Benthem 2008; Icard and Moss 2013). Natural logic

is a flexible approach to doing logical inference directly on natural language expressions. Thus, in this setting, we can work directly with natural language sentences while retaining complete control over all aspects of the generated dataset.

### 4.1 Natural Logic

We define natural logic reasoning over aligned semantic parse trees that represent both the premise and hypothesis as a single structure and allow us to calculate semantic relations for all phrases compositionally. The core components are *semantic relations*, which capture the direct inferential relationships between words and phrases, and *projectivity signatures*, which encode how semantic operators interact compositionally with their arguments. We employ the semantic relations of MacCartney and Manning (2009), as in Table 2. We use $\mathcal{B}$ to denote the set containing these seven semantic relations.

The essential concept for the material to come is that of *joint projectivity*: for a pair of semantic functions $f$ and $g$ and a pair of inputs $X$ and $Y$ that are in relation $R$, the joint projectivity signature $P_{f/g} : \mathcal{B} \to \mathcal{B}$ is a function such that the relation between $f(X)$ and $g(Y)$ is $P_{f/g}(R)$. Figure 2 illustrates this with the phrases *every animal* and *some dog*. We show the details of how the natural logic of MacCartney and Manning (2009), with a small extension, determines the joint projectivity signatures for our datasets in Appendix B.

### 4.2 A Fragment of Natural Language

Our fragment $G$ consists of sentences of the form:

$$\mathrm{Q_S\ Adj_S\ N_S\ Neg\ Adv\ V\ Q_O\ Adj_O\ N_O}$$

where $\mathrm{N_S}$ and $\mathrm{N_O}$ are nouns, V is a verb, $\mathrm{Adj_S}$ and $\mathrm{Adj_O}$ are adjectives, and Adv is an adverb. Neg is *does not*, and $\mathrm{Q_S}$ and $\mathrm{Q_O}$ can be *every*, *not every*, *some*, or *no*; in each of the remaining categories, there are 100 words. Additionally, $\mathrm{Adj_S}$, $\mathrm{Adj_O}$, Adv, and Neg can be the empty string $\varepsilon$, which is represented in the data by a unique token. Semantic scope is fixed by surface order, with earlier elements scoping over later ones.

For NLI, we define the set of premise–hypothesis pairs $\mathcal{S} \subset G \times G$ such that $(s_p, s_h) \in \mathcal{S}$ iff the non-identical non-empty nouns, adjectives, verbs, and adverbs with identical positions in $s_p$ and $s_h$ are in the $\#$ relation. This constraint on $\mathcal{S}$ trivializes the task of determining the lexical relations between adjectives, nouns, adverbs, and verbs, since the relation is $\equiv$ where the two aligned elements are identical and otherwise $\#$. Furthermore, it follows that distinguishing contradictions from entailments is trivial. The only sources of contradictions are negation and the negative quantifiers *no* and *not every*. Consider $(s_p, s_h) \in \mathcal{S}$ and let $C$ be the number of times negation or a negative quantifier occurs in $s_p$ and $s_h$. If $s_p$ contradicts $s_h$, then $C$ is odd; if $s_p$ entails $s_h$, then $C$ is even.

We constrain the open-domain vocabulary to stress models with learning interactions between logically complex function words; we trivialize the task of lexical semantics to isolate the task of compositional semantics. We also do not have multiple morphological forms, use artificial tokens that do not correspond to English words, and collapse *do not* and *not every* to single tokens to further simplify the task and isolate a model's ability to perform compositional logical reasoning.

Our corpora use the three-way labeling scheme of *entailment*, *contradiction*, and *neutral*. To assign these labels, we translate each premise–hypothesis pair into first-order logic and use Prover9 (McCune, 2005–2010). We assume no expression is empty or universal and encode these assumptions as additional premises. This label generation process implicitly assumes the relation between unequal nouns, verbs, adjectives, and adverbs is independence.

When we generate training data for NLI corpora from some subset $\mathcal{S}_{train} \subset \mathcal{S}$, we perform the following balancing. For a given example, every adjective–noun and adverb–verb pair across the premise and hypothesis is equally likely to have the relation $\equiv$, $\sqsubset$, $\sqsupset$, or $\#$. Without this balancing, any given adjective–noun and adverb–verb pair across the premise and hypothesis has more than a 99% chance of being in the independence relation for values of $\mathcal{S}_{train}$ we consider. Even with this step, 98% of the sentence pairs are neutral, so we again sample to create corpora that are balanced across the three NLI labels. This balancing across our three NLI labels justifies our use of an accuracy metric rather than an F1 score.

### 4.3 Composition Trees for NLI

We provide a composition tree for inference on $\mathcal{S}$ in Figure 3. This is an *aligned* composition tree, as in Figure 2: it jointly composes lexical items from the premise and hypothesis. The leaf nodes come in sibling pairs where one sibling is a lexical item from the premise and the other is a lexical item

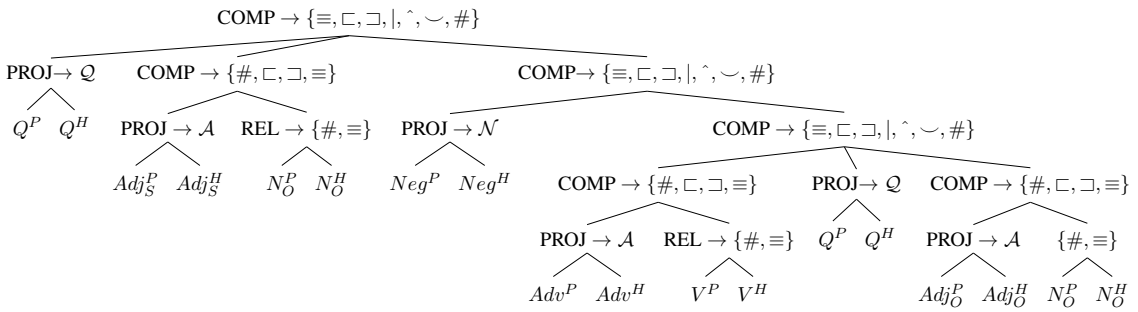


Figure 3: An aligned composition tree for inference on our set of examples $\mathcal{S}$. The superscripts $P$ and $H$ refer to premise and hypothesis. The semantic relations are defined in Table 2. The set $Q$ is {*some*, *every*, *no*, *not*, *every*}. The set *Neg* is $\{\varepsilon, \textit{not}\}$. $\mathcal{Q}$ is the set of 16 joint projectivity signatures between the elements of $Q$. $\mathcal{N}$ is the set of 4 joint projectivity signatures between $\varepsilon$ and *no*. $\mathcal{A}$ is the set of 4 joint projectivity signatures between $\varepsilon$ and an intersective adjective or adverb. REL computes the semantic relations between lexical items, PROJ computes the joint projectivity between two semantic functions (Section 4.1 and Appendix B), and COMP applies semantic relations to joint projectivity signatures. This composition tree defines over $10^{26}$ distinct examples.

from the hypothesis. If both leaf nodes in a sibling pair have domains containing lexical items that are semantic functions, then their parent node domain contains the joint projectivity signatures between those semantic functions. Otherwise the parent node domain contains the semantic relations between the lexical items in the two sibling node domains. The root captures the overall semantic relation between the premise and the hypothesis, while the remaining non-leaf nodes represent intermediate phrasal relations.

The sets $Adj_S$, $N_S$, $Adj_O$, $N_O$, *Adv*, and *V* each have 100 of their respective open class lexical items with *Adv*, $Adj_S$, and $Adj_O$ also containing the empty string $\varepsilon$. The set $Q$ is {*some*, *every*, *no*, *not*, *every*} and the set *Neg* is $\{\varepsilon, \textit{not}\}$. $\mathcal{Q}$ is the set of 16 joint projectivity signatures between the quantifiers *some*, *every*, *no*, and *not every*, $\mathcal{N}$ is the set of 4 joint projectivity signatures between the empty string $\varepsilon$ and *no*, and $\mathcal{A}$ is the set of 4 projectivity signatures between $\varepsilon$ and an intersective adjective or adverb. These joint projectivity signatures were exhaustively determined by us by hand, using the projectivity signatures of negation and quantifiers provided by MacCartney and Manning (2009) as well as a small extension (details in Appendix B).

The function PROJ computes the joint projectivity signature between two semantic functions, REL computes the semantic relation between two lexical items, and COMP inputs semantic relations into a joint projectivity signature and outputs the result. We trimmed the domain of every node so that the function of every node is surjective. Pairs of subexpressions containing quantifiers can be in any of the seven basic semantic relations; even with the contributions of open-class lexical items trivialized, the level of complexity remains high, and all of it emerges from semantic composition, rather than from lexical relations.

### 4.4 A Difficult But Fair NLI Task

A fair training dataset exposes each local function to all possible inputs. Thus, a fair training dataset for NLI will have the following properties. First, all lexical semantic relations must be included in the training data, else the lexical targets could be underdetermined. Second, for any aligned semantic functions $f$ and $g$ with unknown joint projectivity signature $P_{f/g}$, and for any semantic relation $R$, there is some training example where $P_{f/g}$ is exposed to the semantic relation $R$. This ensures that the model has enough information to learn full joint projectivity signatures. Even with these constraints in place, the composition tree of Section 3.1 determines an enormous number of very challenging train/test splits. Appendix C fully defines the procedure for data generation.

We also experimentally verify that our baseline learns a perfect solution from the data we generate. The training set contains 500,000 examples randomly sampled from $\mathcal{S}_{train}$ and the test and development sets each contain 10,000 distinct examples randomly sampled from $\mathcal{S}_{train}$. All random sampling is balanced across adjective–noun and adverb–verb relations as well as across the three NLI labels, as described in Section 4.2.

## 5 Models

We consider six different model architectures:

**CBoW** Premise and hypothesis are represented by the average of their respective word embeddings (continuous bag of words).

**LSTM Encoder** Premise and hypothesis are processed as sequences of words using a recurrent neural network (RNN) with LSTM cells, and the final hidden state of each serves as its representation (Hochreiter and Schmidhuber, 1997; Elman, 1990; Bowman et al., 2015a).

**TreeNN** Premise and hypothesis are processed as trees, and the semantic composition function is a single-layer feed-forward network (Socher et al., 2011b,a). The value of the root node is the semantic representation in each case.

**Attention LSTM** An LSTM RNN with word-by-word attention (Rocktäschel et al., 2015).

**CompTreeNN** Premise and hypothesis are processed as a single aligned tree, following the structure of the composition tree in Figure 3. The semantic composition function is a single-layer feed-forward network (Socher et al., 2011b,a). The value of the root node is the semantic representation of the premise and hypothesis together.

**CompTreeNTN** Identical to the CompTreeNN, but with a neural tensor network as the composition function (Socher et al., 2013).

For the first three models, the premise and hypothesis representations are concatenated. For the CompTreeNN, CompTreeNTN, and Attention LSTM, there is just a single representation of the pair. In all cases, the premise–hypothesis representation is fed through two hidden layers and a softmax layer.

All models are initialized with random 100-dimensional word vectors and optimized using Adam (Kingma and Ba, 2014). It would not be possible to use pretrained word vectors, due to the artificial nature of our dataset. A grid hyperparameter search was run over dropout values of $\{0, 0.1, 0.2, 0.3\}$ on the output and keep layers of LSTM cells, learning rates of $\{1e{-}2, 3e{-}3, 1e{-}3, 3e{-}4\}$, L2 regularization values of $\{0, 1e{-}4, 1e{-}3, 1e{-}2\}$ on all weights, and activation functions $\mathrm{ReLU}$ and $\tanh$. Each hyperparameter setting was run for three epochs and parameters with the highest development set score were used for the complete training runs.

| sentence | adverb-verb phrase |
|---|---|
| every tall kid $\epsilon$ happily kicks every $\epsilon$ rock<br>*entailment*<br>no tall kid does not $\epsilon$ kicks some large rock | happily kicks<br>$\sqsubset$<br>$\epsilon$ kicks |
| negated verb phrases | adjective-noun phrase |
| $\epsilon$ happily kicks every $\epsilon$ rock<br>\|<br>does not $\epsilon$ kicks some large rock | tall kid<br>$\equiv$<br>tall kid |
| verb phrases | single words |
| happily kicks every $\epsilon$ rock<br>$\sqsubset$<br>$\epsilon$ kicks some large rock | tall $\equiv$ tall<br>kid $\equiv$ kid<br>happily $\sqsubset$ $\epsilon$ |
| adjective-noun phrase | single words |
| $\epsilon$ rock<br>$\sqsupset$<br>large rock | kicks $\equiv$ kicks<br>$\epsilon$ $\sqsupset$ large<br>rock $\equiv$ rock |

Figure 4: For any example sentence pair (top left) the neural models are trained using the a weighted sum of the error on 12 prediction tasks shown above. The 12 errors are weighted to regularize the loss according to the length of the expressions being predicted on.

The training datasets for this generalization task are only fair if the outputs realized at every non-leaf node are provided during training just as they are in our baseline learning model. For our neural models, we accomplish this by predicting semantic relations for every subexpression pair in the scope of a node in the tree in Figure 3 and summing the loss of the predictions together. We do not do this for the nodes labeled PROJ $\rightarrow \mathcal{Q}$ or PROJ $\rightarrow \mathcal{N}$, as the function PROJ is a bijection at these nodes and no intermediate representations are created. For any example sentence pair the neural models are trained using the a weighted sum of the error on 12 prediction tasks shown in Figure 4. The 12 errors are weighted to regularize the loss according to the length of the expressions being predicted on.

The CompTreeNN and CompTreeNTN models are structured to create intermediate representations of these 11 aligned phrases and so intermediate predictions are implemented as in the sentiment models of Socher et al. (2013). The other models process each of the 11 pairs of aligned phrases separately. Different softmax layers are used depending on the number of classes, but otherwise the networks have identical parameters for all predictions.

## 6 Results and Analysis

Table 3 summarizes our findings on the hardest of our fair generalization tasks, where the training sets are minimal ones required for fairness.

| Model | Train | Dev | Test |
|---|---|---|---|
| CBoW | $88.04 \pm 0.68$ | $54.18 \pm 0.17$ | $53.99 \pm 0.27$ |
| TreeNN | $67.01 \pm 12.71$ | $54.01 \pm 8.40$ | $53.73 \pm 8.36$ |
| LSTM encoder | $98.43 \pm 0.41$ | $53.14 \pm 2.45$ | $52.51 \pm 2.78$ |
| Attention LSTM | $73.66 \pm 9.97$ | $47.52 \pm 0.43$ | $47.28 \pm 0.95$ |
| CompTreeNN | $99.65 \pm 0.42$ | $80.17 \pm 7.53$ | $80.21 \pm 7.71$ |
| CompTreeNTN | $99.92 \pm 0.08$ | $90.45 \pm 2.48$ | $90.32 \pm 2.71$ |

Table 3: Mean accuracy of 5 runs on our difficult but fair generalization task, with standard 95% confidence intervals. These models are trained on the intermediate predictions described in Section 5.

The four standard neural models fail the task completely. The CompTreeNN and CompTreeNTN, while better, are not able to solve the task perfectly either. However, it should be noted that the CompTreeNN outperforms our four standard neural models by $\approx 30\%$ and the CompTreeNTN improves on this by another $\approx 10\%$. This increase in performance leads us to believe there may be some other composition function that solves this task perfectly.

Both the CompTreeNN and CompTreeNTN have large 95% confidence intervals, indicating that the models are volatile and sensitive to random initialization. The TreeNN also has a large $95\%$ interval. On one of the five runs, the TreeNN achieved a test accuracy of $65.76\%$, much higher than usual, indicating that this model may have more potential than the other three.

Figure 5, left panel, provides further insights into these results by tracking dev-set performance throughout training. It is evident here that the standard models never get traction on the problem. The volatility of the CompTreeNN and CompTreeNTN is also again evident. Notably, the CompTreeNN is the only model that doesn't peak in the first four training epochs, showing steady improvement throughout training.

We can also increase the number of training examples so that the training data redundantly encodes the information needed for fairness. As we do this, the learning problem becomes one of trivial memorization. Figure 5, right panel, tracks performance on this sequence of progressively more trivial problems. The CompTreeNN and CompTreeNTN both rapidly ascend to perfect performance. In contrast, the four standard models continue to have largely undistinguished performance for all but the most trivial problems. Finally, CBoW, while competitive with other neural models initially, falls behind in a permanent way; its inability to account for word order prevents it from even memorizing the training data.

The results in Figure 5 are for models trained to predict the semantic relations for every subexpression pair in the scope of a node in the tree in Figure 3 (as discussed in Section 5), but we also trained the models without intermediate predictions to quantify their impact.

All models fail on our difficult generalization task when these intermediate values are withheld. Without intermediate values this task is unfair by our standards, so this to be expected. In the hardest generalization setting the CBoW model is the only one of the four standard models to show statistically significant improvement when intermediate predictions are made. We hypothesize that the model is learning relations between open-class lexical items, which are more easily accessible in its sentence representations. As the generalization task approaches a memorization task, the four standard models benefit more and more from intermediate predictions. In the easiest generalization setting, the four standard models are unable to achieve a perfect solution without intermediate predictions, while the CompTreeNN and CompTreeNTN models achieve perfection with or without the intermediate values. Geiger et al. (2018) show that this is due to standard models being unable to learn the lexical relations between open class lexical items when not directly trained on them. Even with intermediate predictions, the standard models are only able to learn the base case of this recursive composition.

## 7 The Problem is Architecture

One might worry that these results represent a failure of model capacity. However, the systematic errors remain even for much larger networks;

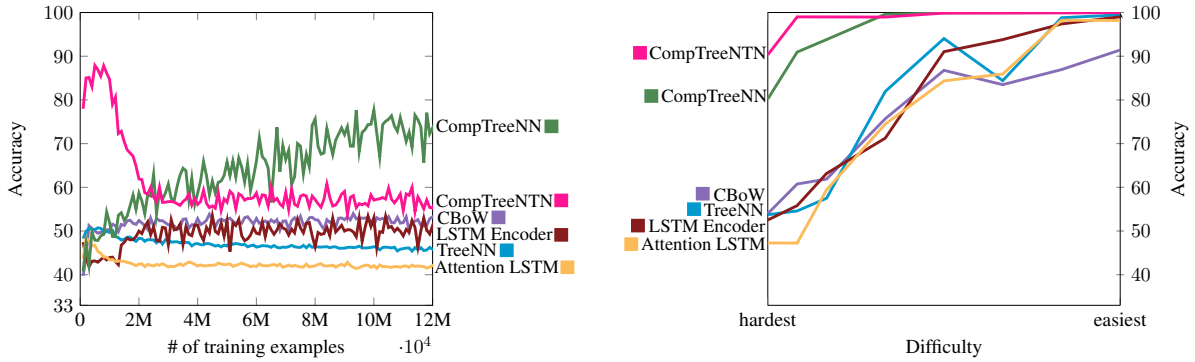


Figure 5: **Left**: Model performance on our difficult but fair generalization task throughout training. **Right**: Mean accuracy of 5 runs as we move from true generalization tasks ('hardest') to problems in which the training set contains so much redundant encoding of the test set that the task is essentially one of memorization ('easiest'). Only the task-specific CompTreeNN and CompTreeNTN are able to do well on true generalization tasks. The other neural models succeed only where memorization suffices, and the CBoW model never succeeds because it does not encode word order.

the trends by epoch and final results are virtually identical with 200-dimensional rather than 100-dimensions representations.

The reason these standard neural models fail to perform natural logic reasoning is their architecture. The CBoW, TreeNN, and LSTM Encoder models all separately bottleneck the premise and hypothesis sentences into two sentence vector embeddings, so the only place interactions between the two sentences can occur is in the two hidden layers before the softmax layer. However, the essence of natural logic reasoning is recursive composition up a tree structure where the premise and hypothesis are composed jointly, so this bottleneck proves extremely problematic. The Attention LSTM model has an architecture that can align and combine lexical items from the premise and hypothesis, but it cannot perform this process recursively and also fails. The CompTreeNN and CompTreeNTN have this recursive tree structure encoded as hard alignments in their architecture, resulting in higher performance. Perhaps in future work, a general purpose model will be developed that can learn to perform this recursive composition without a hard-coded aligned tree structure.

## 8 Conclusion and Future Work

It is vital that we stress-test our models of semantics using methods that go beyond standard naturalistic corpus evaluations. Recent experiments with artificial and adversarial example generation have yielded valuable insights here already, but it is vital that we ensure that these evaluations are fair in the sense that they provide our models with achievable, unambiguous learning targets. We must carefully and precisely navigate the border between meaningful difficulty and impossibility. To this end, we developed a formal notion of fairness for train/test splits.

This notion of fairness allowed us to rigorously pose the question of whether specific NLI models can learn to do robust natural logic reasoning. For our standard models, the answer is no. For our task-specific models, which align premise and hypothesis, the answer is more nuanced; they do not achieve perfect performance on our task, but they do much better than standard models. This helps us trace the problem to the information bottleneck formed by learning separate premise and hypothesis representations. This bottleneck prevents the meaningful interactions between the premise and hypothesis that are at the core of inferential reasoning with language. Our task-specific models are cumbersome for real-world tasks, but they do suggest that truly robust models of semantics will require much more compositional interaction than is typical in today's standard architectures.

## Acknowledgments

We thank Adam Jaffe for help developing mathematical notation and Thomas Icard for valuable discussions. This research is based in part upon work supported by the Stanford Data Science Initiative and by the NSF under Grant No. BCS-1456077. This research is based in part upon work supported by a Stanford Undergraduate Academic Research Major Grant.

## References


Johan van Benthem. 2008. A brief history of natural logic. In *Logic, Navya-Nyaya and Applications: Homage to Bimal Matilal*.

Samuel R. Bowman. 2013. Can recursive neural tensor networks learn logical reasoning? *CoRR*, abs/1312.6192.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015a. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Samuel R. Bowman, Christopher Potts, and Christopher D. Manning. 2015b. Recursive neural networks can learn logical semantics. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 12–21. Association for Computational Linguistics.

Ishita Dasgupta, Demi Guo, Andreas Stuhlmüller, Samuel J. Gershman, and Noah D. Goodman. 2018. Evaluating compositionality in sentence embeddings. *CoRR*, abs/1802.04302.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14(2):179–211.

Richard Evans, David Saxton, David Amos, Pushmeet Kohli, and Edward Grefenstette. 2018. Can neural networks understand logical entailment? *CoRR*, abs/1802.08535.

Atticus Geiger, Ignacio Cases, Lauri Karttunen, and Christopher Potts. 2018. Stress-testing neural models of natural language inference with multiply-quantified sentences. Ms., Stanford University. arXiv 1810.13033.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking nli systems with sentences that require simple lexical inferences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 650–655. Association for Computational Linguistics.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *ICLR*.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Thomas F. Icard and Lawrence S. Moss. 2013. Recent progress on monotonicity. *Linguistic Issues in Language Technology*, 9(7):1–31.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, Copenhagen, Denmark. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Brenden M. Lake and Marco Baroni. 2017. Still not systematic after all these years: On the compositional skills of sequence-to-sequence recurrent networks. *CoRR*, abs/1711.00350.

Bill MacCartney and Christopher D. Manning. 2007. Natural logic for textual inference. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, RTE '07, pages 193–200, Stroudsburg, PA, USA. Association for Computational Linguistics.

Bill MacCartney and Christopher D. Manning. 2009. An extended model of natural logic. In *Proceedings of the Eight International Conference on Computational Semantics*, pages 140–156. Association for Computational Linguistics.

W. McCune. 2005–2010. Prover9 and Mace4. `http://www.cs.unm.edu/~mccune/prover9/`.

Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Rose, and Graham Neubig. 2018. Stress test evaluation for natural language inference. *arXiv preprint arXiv:1806.00692*.

Yixin Nie, Yicheng Wang, and Mohit Bansal. 2018. Analyzing compositionality-sensitivity of NLI models. *CoRR*, abs/1811.07033.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis only baselines in natural language inference. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 180–191. Association for Computational Linguistics.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *CoRR*, abs/1509.06664.

V.M.S. Sánchez-Valencia. 1991. *Studies on Natural Logic and Categorial Grammar*. Universiteit van Amsterdam.

Richard Socher, Cliff Chiung-Yu Lin, Andrew Y. Ng, and Christopher D. Manning. 2011a. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 129–136, USA. Omnipress.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011b. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 151–161, Stroudsburg, PA, USA. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *International Conference on Learning Representations*.

Masatoshi Tsuchiya. 2018. Performance impact caused by hidden bias of training data for recognizing textual entailment. *CoRR*, abs/1804.08117.

Sara Veldhoen and Willem Zuidema. 2018. Can neural networks learn logical reasoning? In *Proceedings of the Conference on Logic and Machine Learning in Natural Language (LaML 2017)*, pages 35–41. Association for Computational Linguistics.

Hitomi Yanaka, Koji Mineshima, Daisuke Bekki, Kentaro Inui, Satoshi Sekine, Lasha Abzianidze, and Johan Bos. 2019. HELP: A dataset for identifying shortcomings of neural models in monotonicity reasoning. ArXiv:1904.12166.

# Everything Happens for a Reason: Discovering the Purpose of Actions in Procedural Text

**Bhavana Dalvi Mishra**[*], **Niket Tandon**[*], **Antoine Bosselut**,
**Wen-tau Yih**,[†] **Peter Clark**
Allen Institute for Artificial Intelligence, Seattle, WA
{bhavanad, nikett, antoineb, peterc}@allenai.org

## Abstract

Our goal is to better comprehend procedural text, e.g., a paragraph about photosynthesis, by not only predicting what happens, but *why* some actions need to happen before others. Our approach builds on a prior process comprehension framework for predicting actions' effects, to also identify subsequent steps that those effects enable. We present our new model (XPAD) that biases effect predictions towards those that (1) explain more of the actions in the paragraph and (2) are more plausible with respect to background knowledge. We also extend an existing benchmark dataset for procedural text comprehension, ProPara, by adding the new task of explaining actions by predicting their dependencies. We find that XPAD significantly outperforms prior systems on this task, while maintaining the performance on the original task in ProPara. The dataset is available at http://data.allenai.org/propara

## 1 Introduction

Procedural text is common in natural language, for example in recipes, how-to guides, and science processes, but understanding it remains a major challenge. While there has been substantial prior work on extracting the sequence of actions ("scripts") from such texts, the task of identifying *why* the sequence is the way it is has received less attention, and is the goal of this work. While "why" can mean many things, we treat it here as describing how one action produces effects that are required by another. For example in Figure 1, "CO2 enters the leaf" is necessary because it results in "CO2 is at the leaf", a precondition for "CO2 forms sugar". (Note that an action does not directly depend on previous actions, but rather on the *state of the world* resulting from previous actions). If one could determine such rationales, new capabilities would become possible, including explanation, identifying alternative event orderings, and answering "*what if...*" questions. However, this task is challenging as it requires knowledge of the preconditions and effects of actions, typically unstated in the text itself.

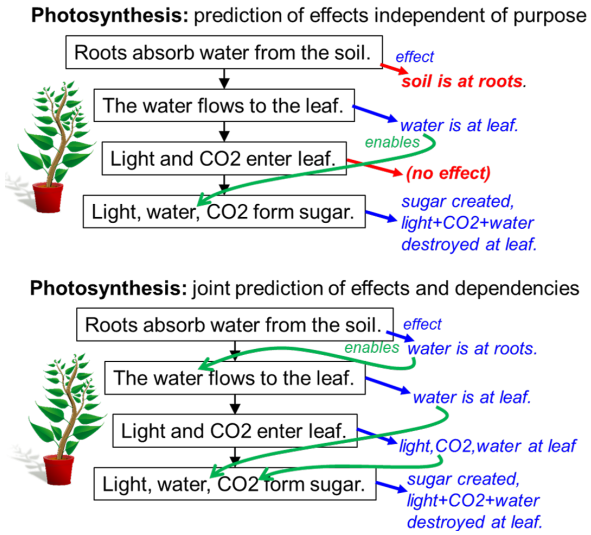


Figure 1: An action sequence (black) describes the order that actions happen, but not why. We add the new task of also predicting *why* the actions are needed, in the form of the actions' effects (blue) and subsequent actions that depend on those effects (green). While a system that predicts effects without considering dependencies can make errors (red, top part), we obtain better predictions by biasing the system to predict effects that *also* result in more dependencies (lower part).

Recent work in neural process modeling goes partway towards our goal by modeling the effects of actions, using annotated training data, allowing the states of entities to be tracked throughout a paragraph, e.g., EntNet (Henaff et al., 2017), NPN (Bosselut et al., 2018), and ProStruct (Tan-

[*]Bhavana Dalvi Mishra and Niket Tandon contributed equally to this work.
[1]Scott is currently at Facebook AI Research (scottyih@fb.com)



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4496–4505, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

don et al., 2018). However, these systems do not consider the purpose of those effects in the overall action sequence. As a result, they are unaware if they predict effects that have no apparent purpose in the process, possibly indicating a prediction error (e.g., the erroneous predictions in red in Figure 1).

To address these limitations, we extend procedural text comprehension with an additional task of predicting the dependencies between steps, in the form of their effects and which subsequent action(s) become possible. Building upon the state-of-the-art framework for predicting effects of actions, we present a new model, called XPAD ("eXPlaining Action Dependencies") that also considers the purpose of those effects. Specifically, XPAD biases those predictions towards those that (1) explain more of the actions in the paragraph and (2) are more plausible with respect to background knowledge. On a benchmark dataset for procedural text comprehension, ProPara (Dalvi et al., 2018), XPAD significantly improves on the prediction and explanation of action dependencies compared to prior systems, while also matching state-of-the-art results on the original tasks. We thus contribute:

1. A new task for procedural text comprehension, namely predicting and explaining the dependencies between actions ("what depends on what, and why"), including an additional dependency graph dataset for ProPara.
2. A model, XPAD, that significantly outperforms prior systems at predicting and explaining action dependencies, while maintaining its performance on the original tasks in ProPara.

## 2 Related Work

Understanding procedural text has a long history in AI. Early work attempted to construct semantic representations ("scripts") of event sequences (or partial orders) from text, including representations of the goals, effects, and purpose of actions, e.g., (Schank and Abelson, 1977; DeJong, 1979; Cullingford, 1986; Mooney, 1986). Some of these systems could explain why actions occurred in text, similar to our goals here, but only on a handful of toy examples using hand-coded background knowledge, and proved difficult to scale. More recent work on event extraction and script learning has proved more effective at extracting event/action sequences from text using statistical methods, e.g., (Chambers and Jurafsky, 2008), and neural techniques, e.g., (Modi and Titov, 2014; Modi, 2016; Pichotta and Mooney, 2016), but have largely focused on *what* happens and in what order, rather than *why*. For example, such representations cannot answer questions about which events would fail if an earlier action in a sequence did not occur. The 2014 ProRead system (Scaria et al., 2013; Berant et al., 2014) included dependency relationships between events that it extracted, but assessed dependencies based on surface language cues, hence could not explain why those dependencies held.

There has also been a line of research in reading procedural text in which the goal has been to track how entities' states change with time, e.g., EntNet (Henaff et al., 2017), ProStruct (Tandon et al., 2018), and Neural Process Networks (Bosselut et al., 2018), applied to procedural text such as cooking recipes (Kiddon et al., 2016), science processes (Dalvi et al., 2018), or toy stories (Weston et al., 2015). Using annotated data, these systems learn to predict the effects of actions from text, allowing simple simulations of how the world changes throughout the process. We build on this line of work to identify the purpose of actions, by connecting their effects to subsequent actions.

In addition to signal from training data, many systems use background knowledge to help bias predictions towards those consistent with that knowledge. For example, (McLauchlan, 2004) predicts prepositional phrase attachments that are more consistent with unambiguous attachments derived from thesaurii; (Clark and Harrison, 2009) tune a parser to prefer bracketings consistent with frequent bracketings stored a text-derived corpus of bracketings; and (Tandon et al., 2018) predicts state changes consistent with those seen in a large text corpus. For our purposes, while KBs such as (Speer and Havasi, 2013; Chu et al., 2017; Park and Nezhad, 2018) contain useful information about action dependencies (e.g., "smoking can cause cancer"), they lack explanations for those links. Instead, we create a KB of dependency links with explanations (Section 5.3), allowing us to similarly bias predictions with background knowledge.

## 3 Problem Definition

The input to our system is a paragraph of procedural text, along with a list of the *participant entities* in the procedure. The output is a *state change matrix* highlighting the changes undergone by the entities, as well as a *dependency explanation graph*

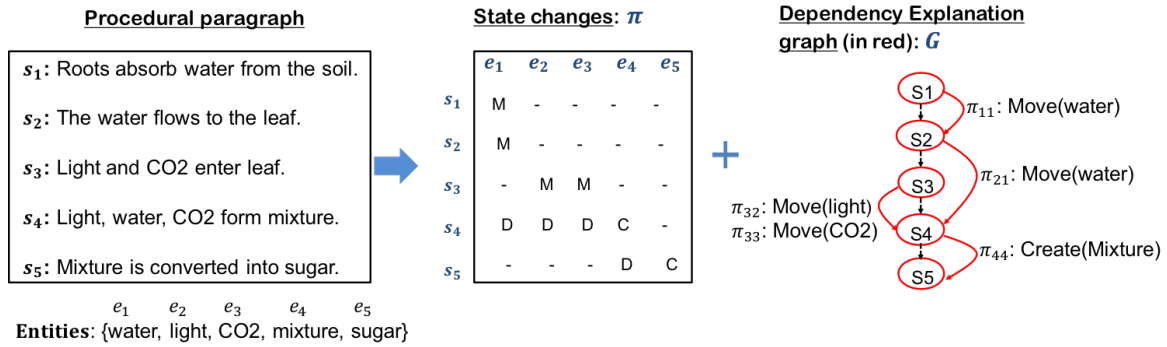


Figure 2: A procedural paragraph and target entities are provided as input. The task is to predict the state change matrix $\pi$, along with the dependency explanation graph $\mathcal{G}$ between steps (red, overlayed on the step sequence). $\pi_{tj}$ refers to the state change happening to entity $e_j$ in step $s_t$. In the ProPara task, there are 4 possible state changes denoted by 'M' (Move), 'C' (Create), 'D' (Destroy), and '-' (no change). Further, $\mathcal{G}$ describes which steps enable which other steps and how. For example, $s_4$ enables $s_5$ by creating `mixture`, required by $s_5$.

that encodes "what action[1] depends on what, and why", as illustrated in Figure 2. This problem definition is derived from the one used for the ProPara dataset (Dalvi et al., 2018), with the key addition of the dependency explanation graph. Although we use ProPara to illustrate and evaluate our work here, our approach is not specific to the details of that particular dataset.

**Paragraph:** We define a paragraph of procedural text $S = \langle s_1, \cdots, s_T \rangle$ as a sequence of sentences that describe a procedure (e.g., photosynthesis). We assume that sentences are chronologically ordered and treat a sentence $s_t$ as a step executed at time $t$.[2]

**Participant Entities:** We assume that the set of participant entities $E = \{e_1, \cdots, e_n\}$ in the paragraph $S$ is given. Notice that $E$ includes only entities that participate in the process where their state changes at some steps.

**State Change Matrix:** For each entity $e_j \in E$, the system tracks how its properties change (e.g., location, existence) after each step. We enumerate all the state changes in a $T \times n$ matrix $\pi$, where $\pi_{tj}$ denote how $e_j$ has changed after step $s_t$. For ProPara, we use just four possible state changes for $\pi_{tj}$, namely {`Move, Create, Destroy, None`}, sufficient to model the properties tracked in the dataset. (More generally, additional state changes could be used provided their effects on entity properties is clearly defined.) `Move` means that the location of $e_j$ changes after $s_t$. `Create` and `Destroy` indicate $e_j$ starts or ceases to exist. `None` means that the state of $e_j$ remains unchanged. A state change can optionally take arguments, e.g., `Move` is associated with before and after locations.

**Dependency Explanation Graph:** In addition to being able to predict when entities undergo state changes, the system also needs to understand the dependency relationships between steps in a procedure. That is, for a step $s_i$ to be executable, the system must be able to identify which previous steps are required to have been completed first. We represent these dependency relationships in procedural text using a *dependency explanation graph* $\mathcal{G} = \langle S, \mathcal{E} \rangle$, where each **node** is a step $s_i \in S$ and a **directed edge** $(s_i, s_j) \in \mathcal{E}$ indicates that $s_i$ is a precondition of $s_j$ ($i < j$). Moreover, each edge is associated with the **explanation** in the form of the entity state change that begets this dependency. For example, in the example shown in Fig. 2, both $s_2$ and $s_3$ are the parent nodes of $s_4$. For *light, water, $CO_2$* to form the *mixture* in $s_4$, the `water` has to move to the leaf in $s_2$ (i.e., $\pi_{21}$ = `Move`), and `light` and $CO_2$ also need to come to the leaf in $s_3$ (i.e., $\pi_{32} = \pi_{33}$ = `Move`). Note that the dependency graph is not the "script" (event sequence) of the process, but an overlay on the script that explains *why* some actions need to happen before others. Note also that the state changes are fully specified (including the from and to locations) - the location information is dropped in Figure 2 for simplicity.

## 4 The Dependency Graph Dataset

Dependency graphs were added to the ProPara dataset using a mixture of manual and automated methods. First, an algorithm was used to estimate

[1] We use a broad definition of action to mean any event that changes the state of the world (including non-volitional events such as roots absorbing water).

[2] This assumption holds in ProPara, the dataset used in this work. However, our techniques can generally be applied as long as a partial order of events is given.

dependencies using a heuristic method (below). Then, for the test set, these dependencies were manually reviewed and corrected. Approximately 15% of these dependencies needed to be changed, suggesting the heuristic algorithm is a reasonable approximation of the ground truth. The train and dev sets remain with the automatically generated (noisy but more numerous) dependency graphs.

As well as helping to augment the ProPara dataset, the algorithm can be used to automatically add approximate explanation graphs to existing models' state change predictions. On first sight, this might seem like a simple solution to the dependency prediction task. However, as we show later, this approach does not produce good results. This is because models' state change predictions were generated without regard for their explainability, and hence prediction errors can cascade into explainability errors. A better solution, which is the basis of XPAD, is to minimize a *joint* loss for both state change and explainability. In this way, an algorithm will be steered towards state changes that are also explainable (Figure 1).

The dependency graph algorithm is as follows, and is based on a *coherence assumption:* If step $s_j$ changes the state of entity $e_k$, we assume that the reason $s_j$ was included in the paragraph is because the next step mentioning $e_k$ requires that change as a precondition. By searching forward for the first subsequent step $s_j$ in which $e_k$ is again mentioned, or changes state again, we add an enable edge in the dependency explanation graph that points from $s_i$ to $s_j$, with the explanation label $\pi_{ik}$. As shown in Sec. 7.1, these dependency graphs that we added are mostly accurate (F1 $\approx$ 0.85).

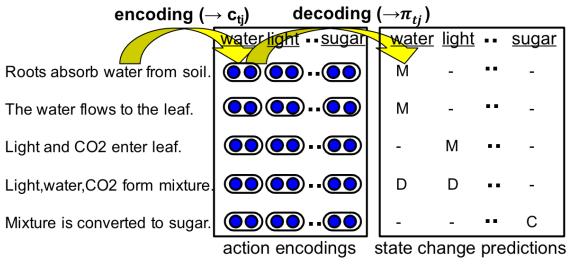


Figure 3: Encoder-decoder architecture used in XPAD: First encode the action of each sentence $s_t$ on entity $e_j$, then decode to a predicted state change (state change arguments not shown). For example, the action *Roots absorb water from soil* is predicted to `Move` (M) the `water` (from soil to roots, not shown). Global predictions are generated through beam search.

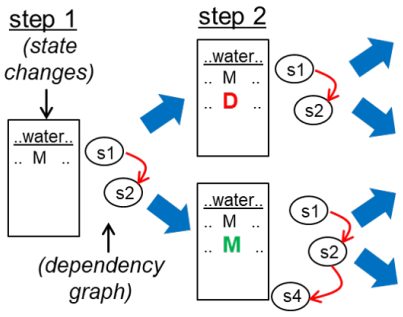


Figure 4: XPAD's beam search for the best decoding, one sentence (step) at a time, showing the predicted state changes and the resulting dependency graph after each step. In step 2, XPAD chooses between predicting whether `water` is `destroyed` (**D**, upper figure) or `moved` (**M**, lower). Predicting `M` results in a more connected dependency graph (as water is mentioned again in step 4), hence the lower choice is likely preferred (Eq. (4)) by our ***dependency graph score***. This score also assesses how a priori likely it is that $s_4$'s movement of water enables $s_5$, using a background KB (Eq. (5)).

## 5 XPAD Model

Our model builds on the approach used in the ProStruct system (Tandon et al., 2018), namely an encoder-decoder approach with beam search decoding. We follow its design for the encoder, but use a modified decoder that also generates dependency graphs, and biases search towards graphs that are both more connected and more a priori likely.

### 5.1 Encoder

For each sentence $s_t$ and each entity $e_j$, the encoder creates a vector $c_{tj}$, capturing how the actions in $s_t$ affect $e_j$ (Figure 3). During encoding, each word $w_i$ in $s_t$ is first represented by its GloVe vector, concatenated with two indicator variables: whether the word refers to $e_j$ and whether the word is a verb. The sentence is then fed into a BiLSTM to generate a contextualized vector $h_i$, which is then passed to a bilinear attention layer: $a_i = h_i B h_{ev} + b$, where $B$ and $b$ are learned parameters, and $h_{ev}$ is the concatenation of $h_e$ and $h_v$ (the averaged contextualized embedding of the entity and the verb words, respectively). The output vector $c_{tj}$ is the attention-weighted sum of the $h_i$: $c_{tj} = \sum_{i=1}^{I} a_i \cdot h_i$.

Each $c_{tj}$ will be decoded (Sec. 5.2) into one of $K$ possible state *changes*. (For the ProPara application, $K = 4$ for {`create`, `move`, `destroy`, `none`}). As preparation for this, we pass each $c_{tj}$ through a feedforward layer to generate $K$ logits that denote how likely each entity $e_j$ is to undergo each state change during step $s_t$. For changes that take addi-

tional arguments (e.g., `move`), the arguments are chosen using standard span prediction techniques over the encoded input $h_i$ (Seo et al., 2017a).

### 5.2 Decoder

The goal of decoding is to determine the state change matrix $\pi$ and the dependency graph $\mathcal{G}$. While each element $\pi_{tj}$ of the state change matrix $\pi$ could be determined solely by choosing the state change with the highest logit value, this greedy, *local* approach often results in nonsensical predictions (e.g., an entity is moved before being created). To avoid this, XPAD instead performs a beam search of possible decodings, one sentence $s_t$ at a time, using a scoring function that includes terms that ensure *global* consistency, as shown in Fig. 4. Let $\pi_t$ be the state change matrix up to step $t$, and $\mathcal{G}_t$ be the dependency graph up to step $t$. As $\pi_t$ is constructed, $\mathcal{G}_t$ is derived from it deterministically using the same heuristic procedure described in Section 4. Most importantly, the scoring function used in the beam search is a function of *both* $\pi_t$ and $\mathcal{G}_t$. This directs search towards state change predictions that are both globally consistent and produce a dependency graph that is well connected and a priori likely. Finally, the decoder outputs the complete state change grid $\pi$ (= $\pi_T$) and dependency graph $\mathcal{G}$ (= $\mathcal{G}_T$) from step 1 to $T$.

### 5.3 Dependency Aware Scoring Function

At each decoding step $t$, XPAD explores different options of $\pi_t$ (state changes for all entities) and $\mathcal{G}_t$ (corresponding dependency explanation graphs) as illustrated in Fig. 4. In particular, it scores the candidates at step $t$ based on two components:

$$\phi(\pi_t, \mathcal{G}_t) = \lambda \cdot f(\pi_t) + (1 - \lambda) \cdot g(\mathcal{G}_t|\pi_t), \quad (1)$$

where (1) $f(\pi_t)$ is the ***state change score*** based on the likelihood of selected state changes at step $t$ given the text and the state change history from steps $s_1$ to $s_{t-1}$, (2) $g(\mathcal{G}_t|\pi_t)$ is the ***dependency graph score*** based on the connectivity and likelihood of the resulting dependency explanation graph $\mathcal{G}_t$, and (3) $\lambda$ is a hyper-parameter that determines the importance of the accuracy of the state changes vs. the coherence of the action dependency graph.

**State Change Score:** To compute $f(\pi_t)$, we reuse the scoring function proposed by ProStruct:

$$f(\pi_t) = \sum_{j=1}^{|E|} \Big( \alpha \cdot \text{logit}(\pi_{tj}) + (1 - \alpha) \cdot \log P_{\text{ext}}(\pi_{tj}|e_j) \Big), \quad (2)$$

where $\alpha$ is a hyper-parameter controlling the degree of bias, $\text{logit}(\pi_{tj})$ is the logit value supplied by the encoder for the local prediction, and $P_{\text{ext}}(\pi_{tj}|e_j)$ indicates how likely entity $e_j$ will go through some change defined in $\pi_{tj}$, based on the topic of the procedural text. We use the knowledge base published by Tandon et al. (2018) to compute $P_{\text{ext}}(\pi_{tj}|e_j)$.

**Dependency Graph Score:** $g(\mathcal{G}_t|\pi_t)$ is the score of the resulting dependency graph given the selected state changes $\pi_t$, defined as:

$$g(\mathcal{G}_t|\pi_t) = \beta \cdot g_{edge}(\mathcal{G}_t|\pi_t) + (1 - \beta) \cdot g_{kb}(\mathcal{G}_t) \quad (3)$$

where (1) $g_{edge}(\mathcal{G}_t|\pi_t)$ scores $\pi_t$'s contributions to the dependency graph, (2) $g_{kb}(\mathcal{G}_t)$ encodes how *a priori* likely the added dependency edges, computed using background knowledge, and (3) $\beta$ is a hyperparameter tuned on the dev set.

With the help of the ***dependency graph score***, XPAD is able to bias the search toward predictions that have an identifiable purpose (i.e., dependency edges), and toward dependency graphs that are more likely according to the background knowledge. Next, we describe scores $g_{edge}$ and $g_{kb}$ in more details.

**(a) Score $g_{edge}$: Actions with Purpose**
We compute $g_{edge}$ in Eq. (3) to bias the search toward dependency graphs that are more connected:

$$g_{edge}(\mathcal{G}_t|\pi_t) = \sum_{j=1}^{|E|} \log\Big(1 + \text{score}_p(\mathcal{G}_t|\pi_{tj})\Big), \quad (4)$$

where $score_p$ is assigned based on whether $\pi_{tj}$ results in a purpose for step $s_t$ or not, as follows:

$$\text{score}_p(\mathcal{G}_t|\pi_{tj}) = \begin{cases} +c, & \text{if } \pi_{tj} \text{ adds an edge to } \mathcal{G}_t; \\ 0, & \text{if no edge can be added} \\ & \text{to } \mathcal{G}_t \text{ as } e_j \text{ isn't mentioned later;} \\ -c, & \text{if } \pi_{tj} \text{ doesn't add an edge} \\ & \text{to } \mathcal{G}_t \text{ and } e_j \text{ is mentioned later.} \end{cases}$$

where $0 < c < 1$ is a hyper-parameter tuned on the dev set. Conceptually, this scoring function checks whether a particular state change prediction results in a connection in the dependency explanation graph $\mathcal{G}_t$. For example, if $\pi_{tj}$ = `Create entity` $e$ and $e$ is mentioned in a later step $s_k$, then an edge will be added between $s_t$ and $s_k$ to $\mathcal{G}_t$. Consequently, $\text{score}_p = c > 0$, resulting in a positive boost in the score for $\pi_{tj}$.

**(b) Score $g_{kb}$: Action Dependency Priors**
To distinguish likely dependency links (e.g., $\mathcal{E}_{ij}$ in $\mathcal{G}_t$) from unlikely ones, we also bias the predictions in Equation 3 with $g_{kb}$. The scoring function $g_{kb}$ computes the prior likelihood of each dependency link $\mathcal{E}_{ij}$ that would be added to $\mathcal{G}_t$, as a result of the predicted actions $\pi_{ik} \in \pi_i$. This bias is particularly helpful when there is a limited amount of training data. If $\mathcal{E}_t$ is the set of edges added to $\mathcal{G}_t$ because of selected state changes $\pi_t$, then

$$g_{KB}(\mathcal{G}_t) = \sum_{\mathcal{E}_{ij} \in \mathcal{E}_t} log\Big(1 + score_e(\mathcal{E}_{ij})\Big) \quad (5)$$

where $score_e(\mathcal{E}_{ij})$ scores the action dependency link (if any, 0 otherwise) added by action $\pi_{ik}$ on an entity $e_k$ that is common to steps $s_i$ and $s_j$. For example, in Figure 2, action $\pi_{11}$ adds an edge `MOVE(water)`, between sentences "*Roots absorb water.*" and "*The water flows to the leaf.*"

We need to estimate the likelihood of $\mathcal{E}_{ij}$ (i.e., the likelihood that $s_i$ can enable $s_j$ via $\pi_{ik}$). This information (especially $\pi_{ik}$) is not present in any existing KB. Therefore, we train a model using a large collection of positive and negative examples of valid/invalid edges. To generate these training examples, we first extract a large collection of procedural texts from WikiHow.com, which contains 1.75M steps from 227K processes across real-world domains, including health, finance, education, home, food, hobbies, etc. On each text, we apply a rule-based system (Clark et al., 2018) that noisily generates $\pi_{train}$, and then apply the heuristics from Section 4 to obtain $\mathcal{G}_{train}$. The distribution in $\mathcal{G}_{train}$ can be quite different from that in ProPara (the current task), so we append $\mathcal{G}_{train}$ with dependency graphs obtained from the training set in ProPara. We then decompose $\mathcal{G}_{train}$ into its $\mathcal{E}_{ij}$ edges (negative examples are created by reversing these edges). This leads to 324,462 training examples. We then add 2,201 examples derived from the ProPara train set. This use of hand-written rules to generate a large number of potentially noisy examples follows others in the literature, e.g. (Sharp et al., 2016; Ahn et al., 2016; Bosselut et al., 2018).

To accommodate for lexical variations, we embed the database of training data in a neural model. Our model itself takes as input $\mathcal{E}_{ij}$ and outputs the likelihood of $\mathcal{E}_{ij}$. To do this, an embedding for $\mathcal{E}_{ij}$ is created using a deep network of biLSTMs, producing a contextual embedding based on the token level embeddings of $s_i$, $s_j$ and the state change vector $\pi_{ik}$. This contextual embedding is then decoded using a feedforward network to predict a score for whether $s_i$ enabled $s_j$ through $\pi_{ik}$. The loss function is designed such that errors on the training examples coming from ProPara are penalized $\theta$ times more than those from WikiHow, where $\theta$ is a hyperparameter tuned on the dev set.

While $g_{edge}$ only scores whether the same entity is used in the two connected events or not, ignoring location information, $g_{kb}$ models how likely there is a dependency between the two events, including using the from/to location of an entity. For example, $g_{kb}$ can model that moving 'light' and 'CO2' to 'location=leaf' is important before they can be turned into a 'mixture' (Figure 2).

### 5.4 Training and Testing XPAD

At training time, XPAD follows only the correct (gold) path through the search space, and learns to minimize the joint loss of predicting the correct state changes and dependency explanation graph for the paragraph. At test time, XPAD performs a beam search to predict the most likely state changes and dependency explanation graph.

### 5.5 Implementation Details for XPAD

We implement XPAD in PyTorch using AllenNLP (Gardner et al., 2018). We use the dataset reader published in ProStruct's publicly available code. We use 100D Glove embeddings (Pennington et al., 2014), trained on Wikipedia 2014 and Gigaword 5 corpora (6B tokens, 400K vocab, uncased). Starting from glove embeddings appended by entity and verb indicators, we use bidirectional LSTM layer to create contextual representation for every word. We use 100D hidden representations for the bidirectional LSTM (Hochreiter and Schmidhuber, 1997) shared between all inputs (each direction uses 50D hidden vectors). The attention layer on top of BiLSTM, uses a bilinear similarity function similar to (Chen et al., 2016) to compute attention weights over the contextual embedding for every word.

To compute the likelihood of all state changes individually, we use a single layer feedforward network with input dimension of 100 and output 4. We then use constrained decoder during training as explained in Section 5.2. We tune the hyperparameters in Equations 1, 2, and 3 as $\lambda = 0.5$, $\alpha = 0.8$, $\beta = 0.8$. We use Adadelta optimizer (Zeiler, 2012) with learning rate 0.2 to minimize total loss.

To make predictions, we use a constrained decoder with beam size of 20, i.e. top 20 choices are explored at each step.

## 6 Experiments

### 6.1 Evaluation metric and baselines

To evaluate XPAD, we measure its performance on dependency explanations, as well as its performance on the original state change prediction task. For state change prediction, we use the same dataset and evaluation metric described in (Tandon et al., 2018), consisting of 1095 test questions for the inputs, outputs, conversions, and movements in the process (straighforwardly derivable from the predicted state change matrix). For dependency explanations, we compare the curated, gold graphs $\mathcal{G}_{gold}$ that we added to ProPara (Section 4) with the predicted dependency graphs $\mathcal{G}_{pred}$. Each edge in a graph contains 3 elements of explanation for "Why $s_t$?", namely "$s_t$ enables $s_u$ due to a state change $\pi_{tk}$ in entity $e_k$", resulting in 2,814 elements to predict in the test set. We compute the precision and recall of predicting these elements for all steps $s_t$, yielding an overall $F_1$ score.

We consider the state-of-the-art process comprehension models reported in (Tandon et al., 2018) as our baselines, including Recurrent Entity Networks (EntNet) (Henaff et al., 2017), Query Reduction Networks (QRN) (Seo et al., 2017b), ProLocal and ProGlobal (Dalvi et al., 2018) and finally, ProStruct (Tandon et al., 2018). These models output state-change predictions, which are used to create the corresponding dependency explanation graph using the method in Section 4, but do not bias their state change predictions towards those that appear purposeful.

### 6.2 Results

**Results on the proposed dependency task:**

| Model | P | R | $F_1$ |
|---|---|---|---|
| ProLocal | 24.7 | 18.0 | 20.8 |
| QRN | 32.6 | 30.3 | 31.4 |
| EntNet | 32.8 | 38.6 | 35.5 |
| ProStruct | 76.3 | 21.3 | 33.4 |
| ProGlobal | 43.4 | 37.0 | 39.9 |
| XPAD | 62.0 | 32.9 | **43.0** |

Table 1: Results on the **dependency** task (test set).

Table 1 reports results of all models on the new **dependency** task. XPAD significantly outperforms the strongest baselines, ProGlobal and ProStruct, by more than 3 points $F_1$. XPAD has much higher precision than ProGlobal with similar recall, suggesting that XPAD's dependency-aware decoder helps it select more accurate dependencies. Compared with ProStruct, it yields more than 11.6 points improvement on recall. As XPAD adds a novel dependency layer on top of the ProStruct architecture, we note that all these gains come exclusively from the dependency layer.

**Impact of XPAD components:**

Table 2 shows the impact of removing dependency graph scores from XPAD. Removing $g_{kb}$, results in a substantial (9 points) drop in recall, since the action dependency priors uses background knowledge to help the model discover dependency links that the model could not infer from the current paragraph. Further, removing $g_{edge}$ score results in around 2 points drop in recall and a substantial increase in precision ($F_1$ is still lower). This is because the $g_{edge}$ score encourages the model to predict state changes that result in more action dependency links, often at the cost of a drop in precision. Thus, the $g_{edge}$ and $g_{kb}$ scores together help XPAD discover meaningful action dependency links.

| | Dependency Task | | |
|---|---|---|---|
| Ablation | P | R | $F_1$ |
| XPAD | 62.0 | 32.9 | **43.0** |
| - $g_{kb}$ | 67.9 | 23.1 | 34.5 |
| - $g_{kb}$ - $g_{edge}$ | 76.3 | 21.3 | 33.4 |

Table 2: Effect of ablating $g_{edge}$ and $g_{kb}$ from XPAD (test set).

**Results on the previous state-change task:**

On the original state-change prediction task, we find that XPAD performs slightly better (by 0.7 points F1) than the best published state-of-the-art system ProStruct[3], even though Equation 1 is not optimized solely for that task (Table 3). This illustrates that encouraging purposefulness in action prediction not only produces more explainable action sequences, but can improve the accuracy of those action predictions themselves.

[3]Since XPAD was developed, two higher unpublished results of 57.6 and 62.5 on the state-change task have appeared on arXiv (Das et al., 2019; Gupta and Durrett, 2019), their systems developed contemporaneously with XPAD. In principle XPAD's approach of jointly learning both state changes and dependencies could be also applied in these new systems. Our main contribution is to show that jointly learning state changes and dependencies can produce more rational (explainable) results, without damaging (here, slightly improving) the state change predictions themselves.

| | P | R | $F_1$ |
|---|---|---|---|
| ProStruct | 74.3 | 43.0 | 54.5 |
| XPAD | 70.5 | 45.3 | **55.2** |

Table 3: Results on the **state-change** task (test set), comparing with the best published prior result.

## 7 Analysis and Discussion

### 7.1 Dependency Graph Computation Errors

Our heuristic algorithm for estimating the dependency graph from state change predictions (Section 4) makes a *coherence assumption*, namely that the purpose of a state change on entity *e* is to enable the next event mentioning *e*. We now describe classes of errors that the algorithm makes, by summarizing errors that had to be corrected when curating the gold explanation graphs for the test set.

**1. Coreference (synonymy)** (35% of errors): Reference to a changed entity is missed if an alternative wording is used, e.g.,

*snow becomes* **ice**...**the mass** *grows smaller...*

**2. Unstated Linkage** (25%): Sometimes use of an entity will be implicit, e.g.,

*Animals eat plants...Animals make waste* **[from the plants]**...

preventing finding what "eat plants" enables.

**3. Ellipsis** (15%): (Sometimes ungrammatical) elided references to earlier entities will hide the purpose of those entity's state changes, e.g.,

**Water** *flows downwards... Enters the dam...*

Thus, this analysis suggests several ways that the dependency graph computation could be improved.

**4. Bridging Anaphora** (15%): In some cases, a sentence will refer indirectly to a changed entity, e.g., by referring to its part, whole, or some other association (i.e., an associative/bridging anaphoric reference (Wei, 2014)). As the algorithm does not resolve such indirect references, the purpose of the action changing that entity is unrecognized, e.g.,

*spark plug causes a* **spark**...**explosion** *occurs...*

*The* **leaf** *absorbs $CO_2$... The* **plant** *produces O2..*

**5. Bad State Change Annotations** (5%): In a few cases, state change annotations are missing or in error in the gold dataset, resulting in missing or incorrect prediction of their purpose.

**6. Long-Range Dependencies** (5%): We assume that an entity's change is to enable the next mentioned use of that entity. This assumption is based solely on typical protocols of discourse, and occasionally is violated. For example:

*Rainwater picks up $CO_2$... Rainwater goes into the soil... The water dissolves limestone...*

Here the first action is a prerequisite for the third, not the second, thus violating our assumption and creating an error in the dependency graph.

Note that around 45% of the sentences in ProPara have no associated state change because of unmodeled state changes (other than create/destroy/move) or the presence of stative sentences (no resultant state changes). As a result, the purpose of such sentences is inherently unrecoverable given these annotations, and out of reach of XPAD (and any other model).

### 7.2 Qualitative Analysis of XPAD's Results

**Corrected Predictions**

In many cases, the bias toward `enables` links results in XPAD predicting a state change when previously `none` had been predicted. For example, in sentence (6) of the paragraph snippet below:

*(6) Millions of years later the fossil forms.*
*(7) A person finds the fossil.*
**ProStruct (for step 6):** `None`
**XPAD:** `CREATE(fossil)` **[correct]**

ProStruct incorrectly predicted no state changes from (6), while XPAD (correctly) predicted that a fossil was created. The extra evidence causing XPAD to make this prediction is that it adds an edge in the dependency graph, giving step (6) a purpose, namely to enable step (7) in which the fossil is found. There are many such examples.

**Erroneous Predictions**

Both of XPAD's new biases encourage XPAD to make state-change predictions that result in more `enables` edges, either by counting them ($g_{edge}$) or summing their likelihoods ($g_{kb}$), resulting in some overlap in their overall effect (Table 6.2). The bias can help predict missing state changes (e.g., above), but can also cause XPAD to "hallucinate" state changes with weak evidence, simply to give each sentence a purpose. An example is:

*(5) Carbon-based mixture stays underground.*
*(6) Humans discover this carbon-based mixture.*
**ProStruct (for step 5):** None **[correct]**
**XPAD:** CREATE (carbon-based mixture)

Here, there is no action associated with step (5). However, in part because this is the first mention of carbon-based mixture, XPAD incorrectly labels it as being created, giving step (5) a purpose of enabling step (6). Although such errors occur, they

are less common than the corrected predictions, hence the improved overall performance.

Comparing the ablated versions of XPAD, we can also see how the different biases play out. Without $g_{kb}$, XPAD can sometimes predict `enables` edges that are nonsensical with respect to background knowledge, for example,

*(1) Nitrogen exists naturally in the atmosphere.*
*(9) Bacteria turn nitrogen into a gas.*
**XPAD (no $g_{kb}$) (for step 1):** CREATE (gas)
**XPAD:** None **[correct]**

In this example, XPAD without $g_{kb}$ produces an incorrect prediction for (1) (that `gas` has been created), as it results in an `enables` link from step (1) to (9). However, the background KB deems it unlikely that (1) enables (9); as a result, the full XPAD makes no such prediction.

## 8 Conclusion

Our goal is to better comprehend procedural text by not only predicting *what* happens, but *why*. To do this, we have expanded the traditional state-tracking task with the additional task of predicting and explaining dependencies between steps, and presented a new model, XPAD, for these tasks. XPAD is biased to prefer predictions that have an identifiable purpose (enables a subsequent step), and where that purpose is more plausible (judged using a large corpus). Experiments show that XPAD significantly improves the predictions of correct dependencies between actions, while matching state-of-the-art results on the earlier tasks.

Although our experiments have been with ProPara, there is nothing intrinsic to XPAD's architecture that limits it to that dataset's design: Given appropriate annotations, XPAD could be trained with a richer set of state change operators, longer paragraphs, and (with the addition of a temporal ordering module, e.g., (Ning et al., 2017)) non-chronological event orderings. In addition, given additional hand-labeled training data, a system might learn to directly predict dependency links (instead of deriving them heuristically from state changes). These would be valuable new directions to pursue. The Dependency Graph dataset is available within `http://data.allenai.org/propara`

### Acknowledgements

We are grateful to the AllenNLP and Beaker teams at AI2, and for the insightful discussions with other Aristo team members. Computations on beaker.org were supported in part by credits from Google Cloud.

## References

Sungjin Ahn, Heeyoul Choi, Tanel Pärnamaa, and Yoshua Bengio. 2016. A neural knowledge language model. *arXiv preprint arXiv:1608.00318*.

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D Manning. 2014. Modeling biological processes for reading comprehension. In *Proc. EMNLP'14*.

Antoine Bosselut, Omer Levy, Ari Holtzman, Corin Ennis, Dieter Fox, and Yejin Choi. 2018. Simulating action dynamics with neural process networks. *6th International Conference on Learning Representations (ICLR)*.

Nathanael Chambers and Daniel Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proc. ACL'08*.

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. *CoRR*, abs/1606.02858.

Cuong Xuan Chu, Niket Tandon, and Gerhard Weikum. 2017. Distilling task knowledge from how-to communities. In *WWW*.

Peter Clark and Philip Harrison. 2009. Large-scale extraction and use of knowledge from text. In *K-CAP*.

Peter Clark, Bhavana Dalvi Mishra, and Niket Tandon. 2018. What happened? leveraging verbnet to predict the effects of actions in procedural text. *arXiv preprint arXiv:1804.05435*.

Richard Cullingford. 1986. The script applier mechanism. In *Readings in natural language processing*, pages 627–649. Morgan Kaufmann Publishers Inc.

Bhavana Dalvi, Lifu Huang, Niket Tandon, Wen-tau Yih, and Peter Clark. 2018. Tracking state changes in procedural text: A challenge dataset and models for process paragraph comprehension. In *NAACL-HLT'18*.

Rajarshi Das, Tsendsuren Munkhdalai, Xingdi Yuan, Adam Trischler, and Andrew McCallum. 2019. Building dynamic knowledge graphs from text using machine reading comprehension. *arXiv preprint arXiv:1810.05682*. ICLR 2019.

Gerald DeJong. 1979. Prediction and substantiation: A new approach to natural language processing. *Cognitive Science*, 3(3):251–273.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018.

Allennlp: A deep semantic natural language processing platform. *arXiv preprint arXiv:1803.07640*.

Aditya Gupta and Greg Durrett. 2019. Tracking discrete and continuous entity state for process understanding. *arXiv preprint arXiv:1904.03518*. (To appear in NAACL'19 workshop on Structured Prediction for NLP).

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. 2017. Tracking the world state with recurrent entity networks. In *ICLR*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Chloé Kiddon, Luke S. Zettlemoyer, and Yejin Choi. 2016. Globally coherent text generation with neural checklist models. In *EMNLP*.

Mark McLauchlan. 2004. Thesauruses for prepositional phrase attachment. In *CoNLL*.

Ashutosh Modi. 2016. Event embeddings for semantic script modeling. In *CoNLL*.

Ashutosh Modi and Ivan Titov. 2014. Inducing neural models of script knowledge. In *CoNLL*.

Raymond J Mooney. 1986. Generalizing explanations of narratives into schemata. In *Machine Learning*, pages 207–212. Springer.

Qiang Ning, Zhili Feng, and Dan Roth. 2017. A structured learning approach to temporal relation extraction. In *EMNLP*, pages 1027–1037.

Hogun Park and Hamid R. Motahari Nezhad. 2018. Learning procedures from text: Codifying how-to procedures in deep neural networks. In *WWW*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Karl Pichotta and Raymond J. Mooney. 2016. Using sentence-level lstm language models for script inference. In *ACL*.

Aju Thalappillil Scaria, Jonathan Berant, Mengqiu Wang, Peter Clark, Justin Lewis, Brittany Harding, and Christopher D. Manning. 2013. Learning biological processes with global constraints. In *EMNLP*.

Roger C Schank and Robert P Abelson. 1977. *Scripts, plans, goals, and understanding: An inquiry into human knowledge structures*. Psychology Press.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017a. Bidirectional attention flow for machine comprehension. In *Proc. ICLR'17*.

Minjoon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. 2017b. Query-reduction networks for question answering. In *ICLR*.

Rebecca Sharp, Mihai Surdeanu, Peter Jansen, Peter Clark, and Michael Hammond. 2016. Creating causal embeddings for question answering with minimal supervision. In *EMNLP'16*.

Robert Speer and Catherine Havasi. 2013. Conceptnet 5: A large semantic network for relational knowledge. In *The People's Web Meets NLP*.

Niket Tandon, Bhavana Dalvi Mishra, Joel Grus, Wentau Yih, Antoine Bosselut, and Peter Clark. 2018. Reasoning about actions and state changes by injecting commonsense knowledge. *EMNLP'18*, arXiv preprint arXiv:1808.10012.

Zhao Wei. 2014. A survey of studies of bridging anaphora. *Canadian Social Science*, 10(3).

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards AI-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

Matthew Zeiler. 2012. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

# CLUTRR: A Diagnostic Benchmark for Inductive Reasoning from Text

**Koustuv Sinha** [1,3,4], **Shagun Sodhani** [2,3], **Jin Dong** [1,3],
**Joelle Pineau** [1,3,4] and **William L. Hamilton** [1,3,4]
[1] School of Computer Science, McGill University, Canada
[2] Université de Montréal, Canada
[3] Montreal Institute of Learning Algorithms (Mila), Canada
[4] Facebook AI Research (FAIR), Montreal, Canada
{koustuv.sinha, sshagunsodhani, jin.dong, jpineau, wlh}
@{mail.mcgill.ca, gmail.com, mail.mcgill.ca, cs.mcgill.ca, cs.mcgill.ca}

## Abstract

The recent success of natural language understanding (NLU) systems has been troubled by results highlighting the failure of these models to generalize in a systematic and robust way. In this work, we introduce a diagnostic benchmark suite, named CLUTRR, to clarify some key issues related to the robustness and systematicity of NLU systems. Motivated by classic work on inductive logic programming, CLUTRR requires that an NLU system infer kinship relations between characters in short stories. Successful performance on this task requires both extracting relationships between entities, as well as inferring the logical rules governing these relationships. CLUTRR allows us to precisely measure a model's ability for systematic generalization by evaluating on held-out combinations of logical rules, and it allows us to evaluate a model's robustness by adding curated noise facts. Our empirical results highlight a substantial performance gap between state-of-the-art NLU models (e.g., BERT and MAC) and a graph neural network model that works directly with symbolic inputs—with the graph-based model exhibiting both stronger generalization and greater robustness.

## 1 Introduction

Natural language understanding (NLU) systems have been extremely successful at reading comprehension tasks, such as question answering (QA) and natural language inference (NLI). An array of existing datasets are available for these tasks. This includes datasets that test a system's ability to extract factual answers from text (Rajpurkar et al., 2016; Nguyen et al., 2016; Trischler et al., 2016; Mostafazadeh et al., 2016; Su et al., 2016), as well as datasets that emphasize commonsense inference, such as entailment between sentences (Bowman et al., 2015; Williams et al., 2018).

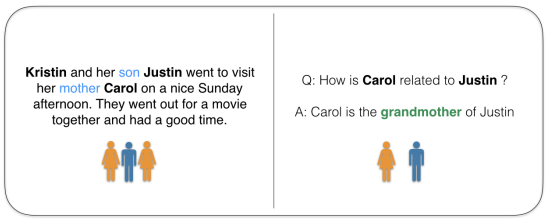


Figure 1: CLUTRR inductive reasoning task.

However, there are growing concerns regarding the ability of NLU systems—and neural networks more generally—to generalize in a systematic and robust way (Bahdanau et al., 2019; Lake and Baroni, 2018; Johnson et al., 2017). For instance, recent work has highlighted the brittleness of NLU systems to adversarial examples (Jia and Liang, 2017), as well as the fact that NLU models tend to exploit statistical artifacts in datasets, rather than exhibiting true reasoning and generalization capabilities (Gururangan et al., 2018; Kaushik and Lipton, 2018). These findings have also dovetailed with the recent dominance of large pre-trained language models, such as BERT, on NLU benchmarks (Devlin et al., 2018; Peters et al., 2018), which suggest that the primary difficulty in these datasets is incorporating the statistics of the natural language, rather than reasoning.

An important challenge is thus to develop NLU benchmarks that can precisely test a model's capability for robust and systematic generalization. Ideally, we want language understanding systems that can not only answer questions and draw inferences from text, but that can also do so in a systematic, logical, and robust way. While such reasoning capabilities are certainly required for many existing NLU tasks, most datasets combine several challenges of language understanding into one, such as co-reference/entity resolution, incorporating world knowledge, and semantic parsing—making it difficult to isolate and diagnose a model's capabilities for systematic generalization and robustness.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4506–4515,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

**Our work**. Inspired by the classic AI challenge of inductive logic programming (Quinlan, 1990)—as well as the recently developed CLEVR dataset for visual reasoning (Johnson et al., 2017)—we propose a semi-synthetic benchmark designed to explicitly test an NLU model's ability for systematic and robust logical generalization.

Our benchmark suite—termed CLUTRR (**C**ompositional **L**anguage **U**nderstanding and **T**ext-based **R**elational **R**easoning)—contains a large set of semi-synthetic stories involving hypothetical families. Given a story, the goal is to infer the relationship between two family members, whose relationship is not explicitly mentioned (Figure 1). To solve this task, a learning agent must extract the relationships mentioned in the text, induce the logical rules governing the kinship relationships (e.g., the transitivity of the sibling relation), and use these rules to infer the relationship between a given pair of entities. Crucially, the CLUTRR benchmark allows us to test a learning agent's ability for *systematic generalization* by testing on stories that contain unseen combinations of logical rules. CLUTRR also allows us to precisely test for the various forms of *model robustness* by adding different kinds of superfluous *noise facts* to the stories.

We compare the performance of several state-of-the-art NLU systems on this task—including Relation Networks (Santoro et al., 2017), Compositional Attention Networks (Hudson and Manning, 2018) and BERT (Devlin et al., 2018). We find that the generalization ability of these NLU systems is substantially below that of a Graph Attention Network (Veličković et al., 2018), which is given direct access to symbolic representations of the stories. Moreover, we find that the robustness of the NLU systems generally does not improve by training on noisy data, whereas the GAT model is able to effectively learn robust reasoning strategies by training on noisy examples. Both of these results highlight important open challenges for closing the gap between machine reasoning models that work with unstructured text and models that are given access to more structured input.

## 2 Related Work

We draw inspiration from the classic work on inductive logic programming (ILP), a long line of reading comprehension benchmarks in NLP, as well as work combining language and knowledge graphs.

**Reading comprehension benchmarks**. Many datasets have been proposed to test the reading comprehension ability of NLP systems. This includes the SQuAD (Rajpurkar et al., 2016), NewsQA (Trischler et al., 2016), and MCTest (Richardson et al., 2013) benchmarks that focus on factual questions; the SNLI (Bowman et al., 2015) and MultiNLI (Williams et al., 2018) benchmarks for sentence understanding; and the bABI tasks (Weston et al., 2015), to name a few. Our primary contribution to this line of work is the development of a carefully designed *diagnostic* benchmark to evaluate model robustness and systematic generalization in the context of NLU.

**Question-answering with knowledge graphs**. Our work is also related to the domain of question answering and reasoning in knowledge graphs (Das et al., 2018; Xiong et al., 2018; Hamilton et al., 2018; Wang et al., 2018; Xiong et al., 2017; Welbl et al., 2018; Kartsaklis et al., 2018), where either the model is provided with a knowledge graph to perform inference over or where the model must infer a knowledge graph from the text itself. However, unlike previous benchmarks in this domain—which are generally *transductive* and focus on leveraging and extracting knowledge graphs as a source of background knowledge about a fixed set of entities—CLUTRR requires *inductive logical reasoning*, where every example requires reasoning over a new set of previously unseen entities.

## 3 Benchmark Design

In order to design an NLU benchmark that explicitly tests inductive reasoning and systematic generalization, we build upon the classic ILP task of inferring family (i.e., kinship) relations (Hinton et al., 1986; Muggleton, 1991; Lavrac and Dzeroski, 1994; Kok and Domingos, 2007; Rocktäschel and Riedel, 2017). For example, given the facts that *"Alice is Bob's mother"* and *"Jim is Alice's father"*, one can infer with reasonable certainty that *"Jim is Bob's grandfather."* While this example may appear trivial, it is a challenging task to design models that can learn from data to *induce* the logical rules necessary to make such inferences, and it is even more challenging to design models that can systematically generalize by composing these induced rules.

Inspired by this classic task of logical induction and reasoning, the CLUTRR benchmark requires an NLU system to infer and reason about kinship

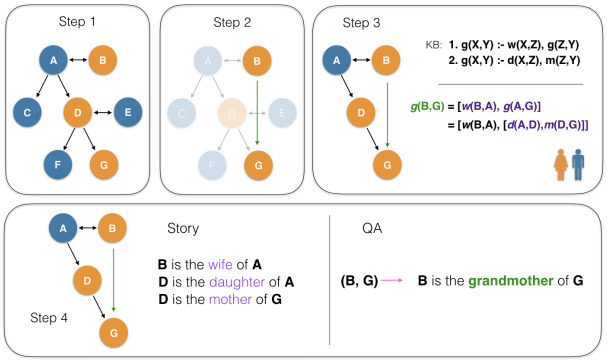


Figure 2: Data generation pipeline. Step 1: generate a kinship graph. Step 2: sample a target fact. Step 3: Use backward chaining to sample a set of facts. Step 4: Convert sampled facts to a natural language story.

relations by reading short stories. Requiring that the models learn directly from natural language makes this task much more challenging than the purely symbolic ILP setting. However, we leverage insights from traditional ILP to generate these stories in a semi-synthetic manner, providing precise control over the complexity of the reasoning required to solve the task.

In its entirety, the CLUTRR benchmark suite allows researchers to generate diverse semi-synthetic short stories to test different aspects of inductive reasoning capabilities. We publicly release the entire benchmark suite, including code to generate the semi-synthetic examples, the specific datasets that we introduce here, and the different baselines that we compare with.[1]

### 3.1 Overview of data generation process

The core idea behind the CLUTRR benchmark suite is the following: Given a natural language story describing a set of kinship relations, the goal is to infer the relationship between two entities, whose relationship is *not* explicitly stated in the story. To generate these stories, we first design a knowledge base (KB) with rules specifying how kinship relations resolve, and we use the following steps to create semi-synthetic stories based on this knowledge base:

**Step 1.** **Generate** a random kinship graph that satisfies the rules in our KB.

**Step 2.** **Sample a target fact** (i.e., relation) to predict from the kinship graph.

**Step 3.** **Apply backward chaining** to sample a set of facts that can prove the target relation (and optionally sample a set of "distracting" or "irrelevant" noise facts).

**Step 4.** **Convert the sampled facts into a natural language story** through pre-specified text templates and crowd-sourced paraphrasing.

Figure 2 provides a high-level overview of this idea, and the following subsections describe the data generation process in detail, as well as the diagnostic flexibility afforded by CLUTRR.

### 3.2 Story generation

The short stories in CLUTRR are essentially narrativized renderings of a set of logical facts. In this section, we describe how we sample the logical facts that make up a story by generating random kinship graphs and using backward chaining to produce logical reasoning chains. The conversion from logical facts to natural language narratives is then described in Section 3.3.

**Terminology and background.** Following standard practice in formal semantics, we use the term *atom* to refer to a *predicate* symbol and a list of terms, such as $[\texttt{grandfatherOf}, X, Y]$, where the predicate $\texttt{grandfatherOf}$ denotes the *relation* between the two *variables*, $X$ and $Y$. We restrict the predicates to have an arity of 2, i.e., binary predicates. A logical *rule* in this setting is of the form $\mathcal{H} \vdash \mathcal{B}$, where $\mathcal{B}$ is the *body* of the rule, i.e., a conjunction of two *atoms* ($[\alpha_1, \alpha_2]$) and $\mathcal{H}$ is the *head*, i.e., a single atom ($\alpha$) that can be viewed as the goal or query. For instance, given a knowledge base (KB) $R$ that contains the single rule $[\texttt{grandfatherOf}, X, Y] \vdash [[\texttt{fatherOf}, X, Z], [\texttt{fatherOf}, Z, Y]]$, the query $[\texttt{grandfatherOf}, X, Y]$ evaluates to true if and only if the body $\mathcal{B} = [[\texttt{fatherOf}, X, Z], [\texttt{fatherOf}, Z, Y]]$ is also true in a given world. A rule is called a *grounded* rule if all atoms in the rule are themselves *grounded*, i.e., all variables are replaced with *constants* or entities in a world. A *fact* is a grounded binary predicate. A *clause* is a conjunction of two or more atoms ($\mathcal{C} = (\mathcal{H}_\mathcal{C} \vdash \mathcal{B}_\mathcal{C} = ([\alpha_1, ..., \alpha_n]))$) which can be built using a set of rules.

In the context of our data generation process, we distinguish between the knowledge base, $R$, which contains a finite number of predicates and rules specifying how kinship relations in a family resolve, and a particular kinship graph $G$, which

[1] Benchmark suite code can be obtained from https://github.com/facebookresearch/clutrr. Generated datasets are available to view in this link.

contains a grounded set of atoms specifying the particular kinship relations that underlie a single story. In other words, $R$ contains the logical rules that govern all the generated stories in CLUTRR, while $G$ contains the grounded facts that underlie a specific story.

**Graph generation**. To generate the kinship graph $G$ underlying a particular story, we first sample a set of gendered[2] entities and kinship relations using a stochastic generation process. This generation process contains a number of tunable parameters—such as the maximum number of children at each node, the probability of an entity being married to another entity, etc.—and is designed to produce a valid, but possibly incomplete "backbone graph". For instance, this backbone graph generation process will specify "parent"/"child" relations between entities but does not add "grandparent" relations. After this initial generation process, we recursively apply the logical rules in $R$ to the backbone graph to produce a final graph $G$ that contains the full set of kinship relations between all the entities.

**Backward chaining**. The resulting graph $G$ provides the *background knowledge* for a specific story, as each edge in this graph can be treated as a grounded predicate (i.e., fact) between two entities. From this graph $G$, we sample the facts that make up the story, as well as the target fact that we seek to predict: First, we (uniformly) sample a target relation $\mathcal{H}_\mathcal{C}$, which is the fact that we want to predict from the story. Then, from this target relation $\mathcal{H}_\mathcal{C}$, we run a simple variation of the backward chaining (Gallaire and Minker, 1978) algorithm for $k$ iterations starting from $\mathcal{H}_\mathcal{C}$, where at each iteration we uniformly sample a subgoal to resolve and then uniformly sample a KB rule that resolves this subgoal. Crucially, unlike traditional backward chaining, we do not stop the algorithm when a proof is obtained; instead, we run for a fixed number of iterations $k$ in order to sample a set of $k$ facts $\mathcal{B}_\mathcal{C}$ that imply the target relation $\mathcal{H}_\mathcal{C}$.

### 3.3 Adding natural language

So far, we have described the process of generating a conjunctive logical clause $\mathcal{C} = (\mathcal{H}_\mathcal{C} \vdash \mathcal{B}_\mathcal{C})$, where $\mathcal{H}_\mathcal{C} = [\alpha^*]$ is the target fact (i.e., relation) we seek to predict and $\mathcal{B}_\mathcal{C} = [\alpha_1, ..., \alpha_k]$ is the set of supporting facts that imply the target relation. We now describe how we convert this logical representation to natural language through crowd-sourcing.

[2]Kinship and gender roles are oversimplified in our data (compared to the real world) to maintain tractability.

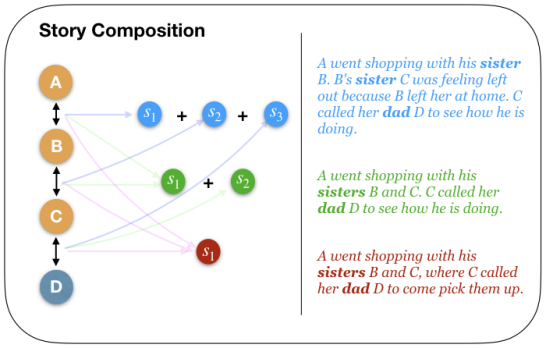


Figure 3: Illustration of how a set of facts can split and combined in various ways across sentences.

**Paraphrasing using Amazon Mechanical Turk**. The basic idea behind our approach is that we show Amazon Mechanical Turk (AMT) crowd-workers the set of facts $\mathcal{B}_\mathcal{C}$ corresponding to a story and ask the workers to paraphrase these facts into a narrative. Since workers are given a set of facts $\mathcal{B}_\mathcal{C}$ to work from, they are able to combine and split multiple facts across separate sentences and construct diverse narratives (Figure 3). Appendix 1.6 contains further details on our AMT interface (based on the ParlAI framework (Miller et al., 2017)), data collection, and the quality controls we employed.

**Reusability and composition**. One challenge for data collection via AMT is that the number of possible stories generated by CLUTRR grows combinatorially as the number of supporting facts increases, i.e., as $k = |\mathcal{B}_\mathcal{C}|$ grows. This combinatorial explosion for large $k$—combined with the difficulty of maintaining the quality of the crowd-sourced paraphrasing for long stories—makes it infeasible to obtain a large number of paraphrased examples for $k > 3$. To circumvent this issue and increase the flexibility of our benchmark, we reuse and compose AMT paraphrases to generate longer stories. In particular, we collected paraphrases for stories containing $k = 1, 2, 3$ supporting facts and then replaced the entities from these collected stories with placeholders in order to re-use them to generate longer semi-synthetic stories. An example of a story generated by stitching together two shorter paraphrases is provided below:

> [Frank] went to the park with his father, [Brett].
> [Frank] called his brother [Boyd] on the phone.
> He wanted to go out for some beers. [Boyd] went
> to the baseball game with his son [Jim].
> Q: What is [Brett] and [Jim]'s relationship?

Thus, instead of simply collecting paraphrases for a fixed number of stories, we instead obtain a diverse

Table 1: Statistics of the AMT paraphrases. Jaccard word overlap is calculated within the templates of each individual clause of length $k$.

| Number of Paraphrases | | | # clauses |
|---|---|---|---|
| | $k = 1$ | 1,868 | 20 |
| | $k = 2$ | 1,890 | 58 |
| | $k = 3$ | 2,258 | 236 |
| | Total | 6,016 | |
| Unique Word Count | | 3,797 | |
| Jaccard Word Overlap | Unigrams | 0.201 | |
| | Bigrams | 0.0385 | |

collection of natural language templates that can be programmatically recombined to generate stories with various properties.

**Dataset statistics**. At the time of submission, we have collected 6,016 unique paraphrases with an average of 19 paraphrases for every possible logical clause of length $k = 1, 2, 3$. Table 1 contains summary statistics of the collected paraphrases. Overall, we found high linguistic diversity in the collected paraphrases. For instance, the average Jaccard overlap in unigrams between pairs paraphrases corresponding to the same logical clause was only 0.201 and only 0.0385 for bigrams. The Appendix contains further examples of the paraphrases.

**Human performance**. To get a sense of the data quality and difficulty involved in CLUTRR, we asked human annotators to solve the task for random examples of length $k = 2, 3, ..., 6$. We found that time-constrained AMT annotators performed well (i.e., $> 70\%$) accuracy for $k \leq 3$ but struggled with examples involving longer stories, achieving 40-50% accuracy for $k > 3$. However, trained annotators with unlimited time were able to solve 100% of the examples (Appendix 1.7), highlighting the fact that this task requires attention and involved reasoning, even for humans.

### 3.4 Query representation and inference

**Representing the question**. The AMT paraphrasing approach described above allows us to convert the set of supporting facts $\mathcal{B}_\mathcal{C}$ to a natural language story, which can be used to predict the target relation/query $\mathcal{H}_\mathcal{C}$. However, instead of converting the target query, $\mathcal{H}_\mathcal{C} = [\alpha^*]$, to a natural language question, we instead opt to represent the target query as a $K$-way classification task, where the two entities in the target relation are provided as input and the goal is to classify the relation that holds between these two entities. This representation avoids the pitfall of revealing information about the answer in the question (Kaushik and Lipton, 2018).

**Representing entities**. When generating stories, entity names are randomly drawn from a set of 300 common gendered English names. Thus, depending on each run, the entities are never the same. This ensures that the entity names are simply placeholders and uncorrelated from the task.

### 3.5 Variants of CLUTRR

The modular nature of CLUTRR provides rich diagnostic capabilities for evaluating the robustness and generalization abilities of neural language understanding systems. We highlight some key diagnostic capabilities available via different variations of CLUTRR below. These diagnostic variations correspond to the concrete datasets that we generated in this work, and we describe the results on these Datasets in Section 4.

**Systematic generalization**. Most prominently, CLUTRR allows us to explicitly evaluate a model's ability for systematic generalization. In particular, we rely on the following hold-out procedures to test systematic generalization:

- During training, we hold out a subset of the collected paraphrases, and we only use this held-out subset of paraphrases when generating the test set. Thus, to succeed on CLUTRR, an NLU system must exhibit *linguistic generalization* and be robust to linguistic variation at test time.
- We also hold out a subset of the logical clauses during training (for clauses of length $k > 2$).[3] In other words, during training, the model sees all logical rules but does not see all *combinations* of these logical rules. Thus, in addition to linguistic generalization, success on this task also requires *logical generalization*.
- Lastly, as a more extreme form of both logical and linguistic generalization, we consider the setting where the models are trained on stories generated from clauses of length $\leq k$ and evaluated on stories generated from larger clauses of length $> k$. Thus, we explicitly test the ability for models to generalize on examples that require more steps of reasoning that any example they encountered during training.

[3] One should not holdout clauses from length $k = 2$ in order to allow models to learn the compositionality of all possible binary predicates.

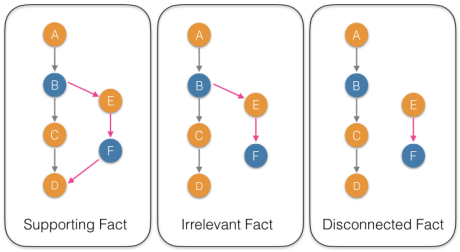


Figure 4: Noise generation procedures of CLUTRR.

**Robust Reasoning**. In addition to evaluating systematic generalization, the modular setup of CLUTRR also allows us to diagnose model robustness by adding *noise facts* to the generated narratives. Due to the controlled semi-synthetic nature of CLUTRR, we are able to provide a precise taxonomy of the kinds of noise facts that can be added (Figure 4). In order to structure this taxonomy, it is important to recall that any set of supporting facts $\mathcal{B}_\mathcal{C}$ generated by CLUTRR can be interpreted as a path, $p_\mathcal{C}$, in the corresponding kinship graph $G$ (Figure 2). Based on this interpretation, we view adding noise facts from the perspective of sampling three different types of noise paths, $p_n$, from the kinship graph $G$:

- *Irrelevant facts*: We add a path $p_n$, which has exactly one shared end-point with $p_c$. In this way, this is a *distractor* path, which contains facts that are connected to one of the entities in the target relation, $\mathcal{H}_\mathcal{C}$, but do not provide any information that could be used to help answer the query.
- *Supporting facts*: We add a path $p_n$, whose two end-points are on the path $p_\mathcal{C}$. The facts on this path $p_n$ are noise because they are not needed to answer the query, but they are supporting facts because they can, in principle, be used to construct alternative (longer) reasoning paths that connect the two target entities.
- *Disconnected facts*: We add paths which neither originate nor end in any entity on $p_c$. These disconnected facts involve entities and relations that are completely unrelated to the target query.

## 4 Experiments

We evaluate several neural language understanding systems on the proposed CLUTRR benchmark to surface the relative strengths and shortcomings of these models in the context of inductive reasoning and combinatorial generalization.[4] We aim to answer the following key questions:

**(Q1)** How do state-of-the-art NLU models compare in terms of systematic generalization? Can these models generalize to stories with unseen combinations of logical rules?

**(Q2)** How does the performance of neural language understanding models compare to a graph neural network that has full access to graph structure underlying the stories?

**(Q3)** How robust are these models to the addition of noise facts to a given story?

### 4.1 Baselines

Our primary baselines are neural language understanding models that take unstructured text as input. We consider bidirectional LSTMs (Hochreiter and Schmidhuber, 1997; Cho et al., 2014) (with and without attention), as well as recently proposed models that aim to incorporate inductive biases towards relational reasoning: Relation Networks (RN) (Santoro et al., 2017) and Compositional Memory Attention Network (MAC) (Hudson and Manning, 2018). We also use the large pretrained language model, BERT (Devlin et al., 2018), as well as a modified version of BERT having a trainable LSTM encoder on top of the pretrained BERT embeddings. All of these models (except BERT) were re-implemented in PyTorch 1.0 (Paszke et al., 2017) and adapted to work with the CLUTRR benchmark.

Since the underlying relations in the stories generated by CLUTRR inherently form a graph, we also experiment with a Graph Attention Network (GAT) (Veličković et al., 2018). Rather than taking the textual stories as input, the GAT baseline receives a structured graph representation of the facts that underlie the story.

**Entity and query representations**. We use the various baseline models to encode the natural language story (or graph) into a fixed-dimensional embedding. With the exception of the BERT models, we do not use pre-trained word embeddings and learn the word embeddings from scratch using end-to-end backpropagation. An important note, however, is that we perform Cloze-style anonymization (Hermann et al., 2015) of the entities (i.e., names) in the stories, where each entity name is replaced by a *@entity-k* placeholder, which is randomly sampled from a small, fixed pool of placeholder tokens. The embeddings for these placeholders are randomly initialized and fixed during training.[5]

[4] Code to reproduce all the results in this section will be released at https://github.com/facebookresearch/clutrr/.

[5] See Appendix 1.5 for a comparison of placeholder em-

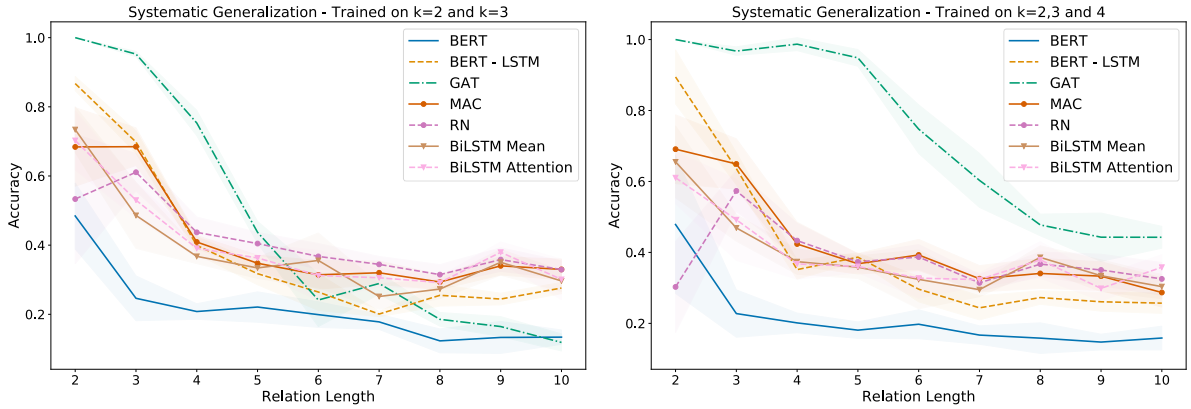


Figure 5: Systematic generalization performance of different models when trained on clauses of length $k = 2, 3$ (Left) and $k = 2, 3, 4$ (Right).

To make a prediction about a target query given a story, we concatenate the embedding of the story (generated by the baseline model) with the embeddings of the two target entities and we feed this concatenated embedding to a 2-layer feed-forward neural network with a softmax prediction layer.

### 4.2 Experimental Setup

**Hyperparameters**. We selected hyperparameters for all models using an initial grid search on the systematic generalization task (described below). All models were trained for 100 epochs with Adam optimizer and a learning rate of 0.001. The Appendix provides details on the selected hyperparameters.

**Generated datasets**. For all experiments, we generated datasets with 10-15k training examples. In many experiments, we report training and testing results on stories with different clause lengths $k$. (For brevity, we use the phrase "clause length" throughout this section to refer to the value $k = |\mathcal{B}_\mathcal{C}|$, i.e., the number of steps of reasoning that are required to predict the target query.) In all cases, the training set contains 5000 train stories per $k$ value, and, during testing, all experiments use 100 test stories per $k$ value. All experiments were run 10 times with different randomly generated stories, and means and standard errors over these 10 runs are reported. As discussed in Section 3.5, during training we holdout 20% of the paraphrases, as well as 10% of the possible logical clauses.

### 4.3 Results and Discussion

With our experimental setup in place, we now address the three key questions (**Q1-Q3**) outlined at the beginning of Section 4.

bedding approaches.

**Q1: Systematic Generalization**

We begin by using CLUTRR to evaluate the ability of the baseline models to perform systematic generalization (**Q1**). In this setting, we consider two training regimes: in the first regime, we train all models with clauses of length $k = 2, 3$, and in the second regime, we train with clauses of length $k = 2, 3, 4$. We then test the generalization of these models on test clauses of length $k = 2, ..., 10$.

Figure 5 illustrates the performance of different models on this generalization task. We observe that the GAT model is able to perform near-perfectly on the held-out logical clauses of length $k = 3$, with the BERT-LSTM being the top-performer among the text-based models but still significantly below the GAT. Not surprisingly, the performance of all models degrades monotonically as we increase the length of the test clauses, which highlights the challenge of "zero-shot" systematic generalization (Lake and Baroni, 2018; Sodhani et al., 2018). However, as expected, all models improve on their generalization performance when trained on $k = 2, 3, 4$ rather than just $k = 2, 3$ (Figure 5, right). The GAT, in particular, achieves the biggest gain by this expanded training.

**Q2: The Benefit of Structure**

The empirical results on systematic generalization also provide insight into how the text-based NLU systems compare against the graph-based GAT model that has full access to the logical graph structure underlying the stories (**Q2**). Indeed, the relatively strong performance of the GAT model (Figure 5) suggests that the language-based models fail to learn a robust mapping from the natural language narratives to the underlying logical facts.

To further confirm this trend, we ran experiments

Table 2: Testing the robustness of the various models when training and testing on stories containing various types of noise facts. The types of noise facts (supporting, irrelevant, and disconnected) are defined in Section 3.5.

| Models | | Unstructured models (no graph) | | | | | | Structured model (with graph) |
|---|---|---|---|---|---|---|---|---|
| Training | Testing | BiLSTM - Attention | BiLSTM - Mean | RN | MAC | BERT | BERT-LSTM | GAT |
| Clean | Clean | 0.58 ±0.05 | 0.53 ±0.05 | 0.49 ±0.06 | 0.63 ±0.08 | 0.37 ±0.06 | 0.67 ±0.03 | **1.0** ±0.0 |
| | Supporting | **0.76** ±0.02 | 0.64 ±0.22 | 0.58 ±0.06 | 0.71 ±0.07 | 0.28 ±0.1 | 0.66 ±0.06 | 0.24 ±0.2 |
| | Irrelevant | 0.7 ±0.15 | **0.76** ±0.02 | 0.59 ±0.06 | 0.69 ±0.05 | 0.24 ±0.08 | 0.55 ±0.03 | 0.51 ±0.15 |
| | Disconnected | 0.49 ±0.05 | 0.45 ±0.05 | 0.5 ±0.06 | 0.59 ±0.05 | 0.24 ±0.08 | 0.5 ±0.06 | **0.8** ±0.17 |
| Supporting | Supporting | 0.67 ±0.06 | 0.66 ±0.07 | 0.68 ±0.05 | 0.65 ±0.04 | 0.32 ±0.09 | 0.57 ±0.04 | **0.98** ±0.01 |
| Irrelevant | Irrelevant | 0.51 ±0.06 | 0.52 ±0.06 | 0.5 ±0.04 | 0.56 ±0.04 | 0.25 ±0.06 | 0.53 ±0.06 | **0.93** ±0.01 |
| Disconnected | Disconnected | 0.57 ±0.07 | 0.57 ±0.06 | 0.45 ±0.11 | 0.4 ±0.1 | 0.17 ±0.05 | 0.47 ±0.06 | **0.96** ±0.01 |
| Average | | **0.61** ±0.08 | 0.59 ±0.08 | 0.54 ±0.07 | **0.61** ±0.06 | 0.30 ±0.07 | 0.56 ±0.05 | **0.77** ±0.09 |

with modified train and test splits for the text-based models, where the same set of natural language paraphrases were used to construct the narratives in both the train and test splits (see Appendix 1.3 for details). In this simplified setting, the text-based models must still learn to reason about held-out logical patterns, but the difficulty of parsing the natural language is essentially removed, as the same natural language paraphrases are used during testing and training. We found that the text-based models were competitive with the GAT model in this simplified setting (Appendix Figure 1), confirming that the poor performance of the text-based models on the main task is driven by the difficulty of parsing the unseen natural language narratives.

**Q3: Robust Reasoning**

Finally, we use CLUTRR to systematically evaluate how various baseline neural language understanding systems cope with noise (**Q3**). In all the experiments we provide a combination of $k = 2$ and $k = 3$ length clauses in training and testing, with noise facts being added to the train and/or test set depending on the setting (Table 2). We use the different types of noise facts defined in Section 3.5.

Overall, we find that the GAT baseline outperforms the unstructured text-based models across most testing scenarios (Table 2), which showcases the benefit of a structured feature space for robust reasoning. When training on clean data and testing on noisy data, we observe two interesting trends that highlight the benefits and shortcomings of the various model classes:

1. All the text-based models excluding BERT actually perform better when testing on examples that have *supporting* or *irrelevant* facts added. This suggests that these models actually benefit from having more content related to the entities in the story. Even though this content is not strictly useful or needed for the reasoning task, it may provide some linguistic cues (e.g., about entity genders) that the models exploit. In contrast, the BERT-based models do not benefit from the inclusion of this extra content, which is perhaps due to the fact that they are already built upon a strong language model (e.g., that already adequately captures entity genders.)

2. The GAT model performs poorly when *supporting* facts are added but has no performance drop when *disconnected* facts are added. This suggests that the GAT model is sensitive to changes that introduce cycles in the underlying graph structure but is robust to the addition of noise that is disconnected from the target entities.

Moreover, when we trained on noisy examples, we found that only the GAT model was able to consistently improve its performance (Table 2). Again, this highlights the performance gap between the unstructured text-based models and the GAT.

## 5 Conclusion

In this paper we introduced the CLUTRR benchmark suite to test the systematic generalization and inductive reasoning capababilities of NLU systems. We demonstrated the diagnostic capabilities of CLUTRR and found that existing NLU systems exhibit relatively poor robustness and systematic generalization capabilities—especially when compared to a graph neural network that works directly with symbolic input. These results highlight the gap that remains between machine reasoning models that work with unstructured text and models that are given access to more structured input. We hope that by using this benchmark suite, progress can be made in building more compositional, modular, and robust NLU systems.

## 6 Acknowledgements

The authors would like to thank Jack Urbanek, Stephen Roller, Adina Williams, Dzmitry Bahdanau, Prasanna Parthasarathy, Harsh Satija for useful discussions and technical help. The authors would also like to thank Abhishek Das, Carlos Eduardo Lassance, Gunshi Gupta, Milan Aggarwal, Rim Assouel, Weiping Song, and Yue Dong for feedback on the draft. The authors also like to thank the many anonymous Mechanical Turk participants for providing paraphrases, and thank Sumana Basu, Etienne Denis, Jonathan Lebensold, and Komal Teru for providing human performance measures. The authors would also like to thank Sanghyun Yoo, Jehun Jeon and Dr Young Sang Choi of Samsung Advanced Institute of Technology (SAIT) for supporting the previous workshop version of this work. The authors are grateful to Facebook AI Research (FAIR) for providing extensive compute and GPU resources and support. This research was supported by the Canada CIFAR Chairs in AI program.

## References

Dzmitry Bahdanau, Shikhar Murty, Michael Noukhovitch, Thien Huu Nguyen, Harm de Vries, and Aaron Courville. 2019. Systematic generalization: What is required and can it be learned? In *International Conference on Learning Representations*.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2018. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *International Conference on Learning Representations*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Herve Gallaire and Jack Minker. 1978. *Logic and Data Bases*. Perseus Publishing.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112.

Will Hamilton, Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec. 2018. Embedding logical queries on knowledge graphs. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 2026–2037. Curran Associates, Inc.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Geoffrey E Hinton et al. 1986. Learning distributed representations of concepts. In *Proceedings of the eighth annual conference of the cognitive science society*, volume 1, page 12. Amherst, MA.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Drew Arad Hudson and Christopher D. Manning. 2018. Compositional attention networks for machine reasoning. In *International Conference on Learning Representations*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 1988–1997. IEEE.

Dimitri Kartsaklis, Mohammad Taher Pilehvar, and Nigel Collier. 2018. Mapping text to knowledge graph entities using multi-sense lstms. *arXiv preprint arXiv:1808.07724*.

Divyansh Kaushik and Zachary C Lipton. 2018. How much reading does reading comprehension require? a critical investigation of popular benchmarks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5010–5015.

Stanley Kok and Pedro Domingos. 2007. Statistical predicate invention. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 433–440, New York, NY, USA. ACM.

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning*, pages 2879–2888.

Nada Lavrac and Saso Dzeroski. 1994. Inductive logic programming. In *WLP*, pages 146–160. Springer.

Alexander Miller, Will Feng, Dhruv Batra, Antoine Bordes, Adam Fisch, Jiasen Lu, Devi Parikh, and Jason Weston. 2017. Parlai: A dialog research software platform. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 79–84.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of NAACL-HLT*, pages 839–849.

Stephen Muggleton. 1991. Inductive logic programming. *New Generation Computing*, 8(4):295–318.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*.

J R Quinlan. 1990. Learning logical definitions from relations. *Mach. Learn.*, 5(3):239–266.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 193–203.

Tim Rocktäschel and Sebastian Riedel. 2017. End-to-end differentiable proving. In *Advances in Neural Information Processing Systems*, pages 3788–3800.

Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. 2017. A simple neural network module for relational reasoning. In *Advances in neural information processing systems*, pages 4967–4976.

Shagun Sodhani, Sarath Chandar, and Yoshua. Bengio. 2018. On Training Recurrent Neural Networks for Lifelong Learning. *arXiv e-prints*.

Yu Su, Huan Sun, Brian Sadler, Mudhakar Srivatsa, Izzeddin Gur, Zenghui Yan, and Xifeng Yan. 2016. On generating characteristic-rich question sets for qa evaluation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 562–572.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2016. NewsQA: A machine comprehension dataset. *arXiv preprint*, pages 1–12.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations*.

Z. Wang, L. Li, D. D. Zeng, and Y. Chen. 2018. Attention-based multi-hop reasoning for knowledge graph. In *2018 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 211–213.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association of Computational Linguistics*, 6:287–302.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards AI-Complete question answering: A set of prerequisite toy tasks.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 1112–1122.

Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 564–573.

Wenhan Xiong, Mo Yu, Shiyu Chang, Xiaoxiao Guo, and William Yang Wang. 2018. One-shot relational learning for knowledge graphs. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1980–1990.

# Taskmaster-1: Toward a Realistic and Diverse Dialog Dataset

**Bill Byrne**[1*] **Karthik Krishnamoorthi**[1*] **Chinnadhurai Sankar**[2*] **Arvind Neelakantan**[1]
**Daniel Duckworth**[1] **Semih Yavuz**[3] **Ben Goodrich**[1]
**Amit Dubey**[1] **Andy Cedilnik**[1] **Kyu-Young Kim**[1]
[1]Google LLC, Mountain View, California
[2]Mila, Université de Montréal
[3]University of California, Santa Barbara
{billb,krishnamoorthi,aneelakantan}@google.com

## Abstract

A significant barrier to progress in data-driven approaches to building dialog systems is the lack of high quality, goal-oriented conversational data. To help satisfy this elementary requirement, we introduce the initial release of the Taskmaster-1 dataset which includes 13,215 task-based dialogs comprising six domains. Two procedures were used to create this collection, each with unique advantages. The first involves a two-person, spoken "Wizard of Oz" (WOz) approach in which trained agents and crowdsourced workers interact to complete the task while the second is "self-dialog" in which crowdsourced workers write the entire dialog themselves. We do not restrict the workers to detailed scripts or to a small knowledge base and hence we observe that our dataset contains more realistic and diverse conversations in comparison to existing datasets. We offer several baseline models including state of the art neural seq2seq architectures with benchmark performance as well as qualitative human evaluations. Dialogs are labeled with API calls and arguments, a simple and cost effective approach which avoids the requirement of complex annotation schema. The layer of abstraction between the dialog model and the service provider API allows for a given model to interact with multiple services that provide similar functionally. Finally, the dataset will evoke interest in written vs. spoken language, discourse patterns, error handling and other linguistic phenomena related to dialog system research, development and design.[1]

## 1 Introduction

Voice-based "personal assistants" such as Apple's SIRI, Microsoft's Cortana, Amazon Alexa, and the Google Assistant have finally entered the mainstream. This development is generally attributed to major breakthroughs in speech recognition and text-to-speech (TTS) technologies aided by recent progress in deep learning (Lecun et al., 2015), exponential gains in compute power (Steinkrau et al., 2005; Jouppi et al., 2017), and the ubiquity of powerful mobile devices. The accuracy of machine learned speech recognizers (Hinton et al., 2012) and speech synthesizers (van den Oord et al., 2016) are good enough to be deployed in real-world products and this progress has been driven by publicly available labeled datasets. However, conspicuously absent from this list is equal progress in machine learned conversational natural language understanding (NLU) and generation (NLG). The NLU and NLG components of dialog systems starting from the early research work (Weizenbaum, 1966) to the present commercially available personal assistants largely rely on rule-based systems. The NLU and NLG systems are often carefully programmed for very narrow and specific cases (Google, 2019; Amazon, 2019). General understanding of natural spoken behaviors across multiple dialog turns, even in single task-oriented situations, is by most accounts still a long way off. In this way, most of these products are very much hand crafted, with inherent constraints on what users can say, how the system responds and the order in which the various subtasks can be completed. They are high precision but relatively low coverage. Not only are such systems unscalable, but they lack the flexibility to engage in truly natural conversation.

Yet none of this is surprising. Natural language is heavily context dependent and often ambiguous, especially in multi-turn conversations across multiple topics. It is full of subtle discourse cues and pragmatic signals whose patterns have yet to be thoroughly understood. Enabling an automated

[*]Equal Contribution
[1]Dataset available at https://g.co/dataset/taskmaster-1



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4516–4525,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

system to hold a coherent task-based conversation with a human remains one of computer science's most complex and intriguing unsolved problems (Weizenbaum, 1966). In contrast to more traditional NLP efforts, interest in statistical approaches to dialog understanding and generation aided by machine learning has grown considerably in the last couple of years (Rojas-Barahona et al., 2017; Bordes et al., 2017; Henderson et al., 2013). However, the dearth of high quality, goal-oriented dialog data is considered a major hindrance to more significant progress in this area (Bordes et al., 2017; Lowe et al., 2015).

To help solve the data problem we present Taskmaster-1, a dataset consisting of 13,215 dialogs, including 5,507 spoken and 7,708 written dialogs created with two distinct procedures. Each conversation falls into one of six domains: ordering pizza, creating auto repair appointments, setting up ride service, ordering movie tickets, ordering coffee drinks and making restaurant reservations. For the spoken dialogs, we created a Wizard of Oz (WOz) system (Kelley, 1984) to collect two-person, spoken conversations. Crowdsourced workers playing the "user" interacted with human operators playing the digital assistant using a web-based interface. In this way, users were led to believe they were interacting with an automated system while it was in fact a human, allowing them to express their turns in natural ways but in the context of an automated interface. We refer to this spoken dialog type as "two-person dialogs". For the written dialogs, we engaged crowdsourced workers to write the full conversation themselves based on scenarios outlined for each task, thereby playing roles of both the user and assistant. We refer to this written dialog type as "self-dialogs". In a departure from traditional annotation techniques (Henderson et al., 2013; Rojas-Barahona et al., 2017; Budzianowski et al., 2018), dialogs are labeled with simple API calls and arguments. This technique is much easier for annotators to learn and simpler to apply. As such it is more cost effective and, in addition, the same model can be used for multiple service providers.

Taskmaster-1 has richer and more diverse language than the current popular benchmark in task-oriented dialog, MultiWOZ (Budzianowski et al., 2018). Table 1 shows that Taskmaster-1 has more unique words and is more difficult for language models to fit. We also find that Taskmaster-1 is more realistic than MultiWOZ. Specifically, the two-person dialogs in Taskmaster-1 involve more real-word entities than seen in MutliWOZ since we do not restrict conversations to a small knowledge base. Beyond the corpus and the methodologies used to create it, we present several baseline models including state-of-the-art neural seq2seq architectures together with perplexity and BLEU scores. We also provide qualitative human performance evaluations for these models and find that automatic evaluation metrics correlate well with human judgments. We will publicly release our corpus containing conversations, API call and argument annotations, and also the human judgments.

| Statistic | Self-dialogs | MultiWOZ |
|---|---|---|
| # unique words | **21,894** | 19,175 |
| # unique named entities | **8,218** | 1,338 |
| # utterances | **169,469** | 132,610 |
| # dialogs | 7,708 | **10,438** |
| Avg. utterances per dialog | **21.99** | 13.70 |
| Avg. tokens per utterance | 8.62 | **13.82** |
| Perplexity | **17.08** | 15.62 |
| BLEU | **6.53** | 11.02 |

Table 1: Statistics comparison: Self-dialogs vs MultiWOZ corpus both containing approximately 10k dialogues each.

## 2 Related work

### 2.1 Human-machine vs. human-human dialog

Serban et al. (2017) discuss the major features and differences among the existing offerings in an exhaustive and detailed survey of available corpora for data driven learning of dialog systems. One important distinction covered is that of human-human vs. human-machine dialog data, each having its advantages and disadvantages. Many of the existing task-based datasets have been generated from deployed dialog systems such as the Lets Go Bus Information System (Raux et al., 2003) and the various Dialog State Tracking Challenges (DSTCs) (Williams et al., 2016). However, it is doubtful that new data-driven systems built with this type of corpus would show much improvement since they would be biased by the existing system and likely

mimic its limitations (Williams and Young, 2007). Since the ultimate goal is to be able to handle complex human language behaviors, it would seem that human-human conversational data is the better choice for spoken dialog system development (Budzianowski et al., 2018). However, learning from purely human-human based corpora presents challenges of its own. In particular, human conversation has a different distribution of understanding errors and exhibits turn-taking idiosyncrasies which may not be well suited for interaction with a dialog system (Williams and Young, 2007; Serban et al., 2017).

### 2.2 The Wizard of Oz (WOz) Approach and MultiWOZ

The WOz framework, first introduced by Kelley (1984) as a methodology for iterative design of natural language interfaces, presents a more effective approach to human-human dialog collection. In this setup, users are led to believe they are interacting with an automated assistant but in fact it is a human behind the scenes that controls the system responses. Given the human-level natural language understanding, users quickly realize they can comfortably and naturally express their intent rather than having to modify behaviors as is normally the case with a fully automated assistant. At the same time, the machine-oriented context of the interaction, i.e. the use of TTS and slower turn taking cadence, prevents the conversation from becoming fully fledged, overly complex human discourse. This creates an idealized spoken environment, revealing how users would openly and candidly express themselves with an automated assistant that provided superior natural language understanding.

Perhaps the most relevant work to consider here is the recently released MultiWOZ dataset (Budzianowski et al., 2018), since it is similar in size, content and collection methodologies. MultiWOZ has roughly 10,000 dialogs which feature several domains and topics. The dialogs are annotated with both dialog states and dialog acts. MultiWOZ is an entirely written corpus and uses crowdsourced workers for both assistant and user roles. In contrast, Taskmaster-1 has roughly 13,000 dialogs spanning six domains and annotated with API arguments. The two-person spoken dialogs in Taskmaster-1 use crowdsourcing for the user role but trained agents for the assistant role. The assistant's speech is played to the user via TTS. The

| | |
|---|---|
| ASSISTANT: | How can I help you? |
| USER: | Hi, could you help me with booking movie tickets for tonight? |
| ASSISTANT: | What movie are you interested in? |
| USER: | The Upside. |
| ASSISTANT: | Did you have a theater in mind? |
| USER: | Could you check if the Regal Neshaminy... No, AMC Neshaminy in Neshaminy, PA is playing it? |
| ASSISTANT: | Could you spell that? |
| USER: | Sure, n e s h a m i n y. |
| ASSISTANT: | I have a showtime at 7:30 and at 10:30, is that okay? |
| USER: | Yes, could you get two tickets for the 7:30? |
| ASSISTANT: | One moment. Okay so that's 2 tickets for 7:30 at the AMC Neshaminy 24? |
| USER: | Yes. |
| ASSISTANT: | It'll be twenty-four ninety-nine for your tickets. |
| USER: | That sounds great. |
| ASSISTANT: | I've confirmed your tickets, they'll arrive via text shortly. Did you need any other information? |
| USER: | No, that was it. Thank you so much for your help. |
| ASSISTANT: | Great, no problem. I hope you have fun. |
| USER: | I hope so, too. Thank you so much. |

Figure 1: Sample Taskmaster-1 two-person dialog

remaining 7,708 conversations in Taskmaster-1 are self-dialogs, in which crowdsourced workers write the entire conversation themselves. As Krause et al. (2017); Moghe et al. (2018) show, self dialogs are surprisingly rich in content.

## 3 The Taskmaster Corpus

### 3.1 Overview

There are several key attributes that make Taskmaster-1 both unique and effective for data-driven approaches to building dialog systems and for other research.

**Spoken and written dialogs:** While the spoken sources more closely reflect conversational language (Chafe and Tannen, 1987), written dialogs are significantly cheaper and easier to gather. This allows for a significant increase in the size of the corpus and in speaker diversity.

**Goal-oriented dialogs:** All dialogs are based on one of six tasks: ordering pizza, creating auto repair appointments, setting up rides for hire, ordering movie tickets, ordering coffee drinks and making restaurant reservations.

**Two collection methods:** The two-person dialogs and self-dialogs each have pros and cons, revealing interesting contrasts.

MAIN TASK: Users will pretend they are using a voice-powered personal digital assistant to book movie tickets for a film they ALREADY have in mind.

1. In several turns (not just one!), cover the following:
   (a) Film name
   (b) Number of people
   (c) City
   (d) Theater
   (e) Time
   (f) If applicable: 3D vs. IMAX vs. standard.
2. They may also want to know things like:
   (a) Run time
   (b) End time
   (c) Director, actors, etc.
3. Make sure to CONFIRM all the relevant ticket details before the end of the dialogue INCLUDING:
   (a) Total cost for two tickets
   (b) Time, location, theater
4. You can assume you have the users account info with the ticket service–so no credit card information is necessary.
5. After confirming the details, end the conversation by confirming that the tickets are being sent to the users mobile device as a text message.

Figure 2: Sample instructions for agents playing "assistant" role

MAIN TASK: Pretend you are using your voice-powered digital assistant to book movie tickets.

1. Start by thinking of a particular movie PLAYING NOW in theaters that you'd like to see. (Use the internet to find one if necessary.)
2. Choose a DIFFERENT CITY from where you live, work, or happen to be at the moment.
3. Pretend you've decided to see this movie tonight and you're taking a friend.
4. The assistant will ask about all relevant details BUT you should make sure it covers all your needs.
5. You can assume you already have an account with the ticket service–so no credit card information is necessary.
6. The assistant will end the conversation by confirming that your tickets are being sent to your mobile device as a text message. (And you can respond thanks, goodbye, ok, etc. for a final closing turn, if you like).

Figure 3: Sample instructions for crowdsourced workers playing "user" role

**Multiple turns:** The average number of utterances per dialog is about 23 which ensures context-rich language behaviors.

**API-based annotation:** The dataset uses a simple annotation schema providing sufficient grounding for the data while making it easy for workers to apply labels consistently.

**Size:** The total of 13,215 dialogs in this corpus is on par with similar, recently released datasets such as MultiWOZ (Budzianowski et al., 2018).

### 3.2 Two-person, spoken dataset

In order to replicate a two-participant, automated digital assistant experience, we built a WOz platform that pairs agents playing the digital assistant with crowdsourced workers playing the user in task-based conversational scenarios. An example dialog from this dataset is given in Figure 1.

#### 3.2.1 WOz platform and data pipeline

While it is beyond the scope of this work to describe the entire system in detail, there are several platform features that help illustrate how the process works.

**Modality:** The agents playing the assistant type their input which is in turn played to the user via text-to-speech (TTS) while the crowdsourced workers playing the user speak aloud to the assistant using their laptop and microphone. We use WebRTC to establish the audio channel. This setup creates a digital assistant-like communication style.

**Conversation and user quality control:** Once the task is completed, the agents tag each conversation as either successful or problematic depending on whether the session had technical glitches or user behavioral issues. We are also then able to root out problematic users based on this logging.

**Agent quality control:** Agents are required to login to the system which allows us to monitor performance including the number and length of each session as well as their averages.

**User queuing:** When there are more users trying to connect to the system than available agents, a queuing mechanism indicates their place in line and connects them automatically once they move to the front of the queue.

**Transcription:** Once complete, the user's audio-only portion of the dialog is transcribed by a second set of workers and then merged with the assistant's

typed input to create a full text version of the dialog. Finally, these conversations are checked for transcription errors and typos and then annotated, as described in Section 3.4.

#### 3.2.2 Agents, workers and training

Both agents and crowdsourced workers are given written instructions prior to the session. Examples of each are given in Figure 2 and Figure 3. The instructions continue to be displayed on screen to the crowdsourced workers while they interact with the assistant. Instructions are modified at times (for either participant or both) to ensure broader coverage of dialog scenarios that are likely to occur in actual user-assistant interactions. For example, in one case users were asked to change their mind after ordering their first item and in another agents were instructed to tell users that a given item was not available. Finally, in their instructions, crowdsourced workers playing the user are told they will be engaging in conversation with a digital assistant. However, it is plausible that some suspect human intervention due to the advanced level of natural language understanding from the assistant side.

Agents playing the assistant role were hired from a pool of dialog analysts and given two hours of training on the system interface as well as on how to handle specific scenarios such as uncooperative users and technical glitches. Uncooperative users typically involve those who either ignored agent input or who rushed through the conversation with short phrases. Technical issues involved dropped sessions (e.g. WebRTC connections failed) or cases in which the user could not hear the agent or vice-versa. In addition, weekly meetings were held with the agents to answer questions and gather feedback on their experiences. Agents typically work four hours per day with dialog types changing every hour. Crowdsourced workers playing the user are accessed using Amazon Mechanical Turk. Payment for a completed dialog session lasting roughly five to seven minutes was typically in the range of $1.00 to $1.30. Problematic users are detected either by the agent involved in the specific dialog or by post-session assessment and removed from future requests.

### 3.3 Self-dialogs (one-person written dataset)

While the two-person approach to data collection creates a realistic scenario for robust, spoken dialog data collection, this technique is time consuming, complex and expensive, requiring considerable technical implementation as well as administrative procedures to train and manage agents and crowdsourced workers. In order to extend the Taskmaster dataset at minimal cost, we use an alternative self-dialog approach in which crowdsourced workers write the full dialogs themselves (i.e. interpreting the roles of both user and assistant).

1. Think of a particular movie PLAYING NOW in theaters that you'd like to see. (Use the internet to find one if necessary.)
2. Choose a DIFFERENT CITY from where you live, work, or happen to be at the moment.
3. Pretend you've decided to see this movie tonight and you're taking a friend.
4. Use the internet to look up the details of the city, the theater name, showtimes offered, ticket prices, and any additional options like 3D, etc.
5. MAIN TASK: Pretend you call your personal assistant on the phone who will book the ticket for you. Write the conversation that would happen between you and your assistant in order to buy two tickets.
6. MAKE SURE the assistant asks about all relevant details (see 4) INCLUDING the number of tickets needed. BUT you should choose the order that makes sense to you as far what details to ask (theater, times, etc)
7. You can assume you already have an account with the ticket service–so no credit card information is necessary.
8. The assistant should end the conversation by confirming that your tickets are being sent to your mobile device as a text message. (And you can respond thanks, goodbye, ok, etc. for a final closing turn, if you like).

- **YOUR TASK**: Write the conversation that results between you and your assistant. It must be at least 10 turns long (for both you and the assistant). Below we have provided 15 turns in case you need more. KEEP IT NEW AND FRESH! DON'T REPEAT DIALOGUES FROM THE PAST!

Figure 4: Sample instructions for written "self-dialogs"

#### 3.3.1 Task scenarios and instructions

Targeting the same six tasks used for the two-person dialogs, we again engaged the Amazon Mechanical Turk worker pool to create self-dialogs, this time as a written exercise. In this case, users are asked to pretend they have a personal assistant who can help them take care of various tasks in real time. They are told to imagine a scenario in which they are speaking to their assistant on the phone while the assistant accesses the services for one of the given tasks. They then write down the entire conversation. Figure 4 shows a sample set of

| | |
|---|---|
| USER: | Hi I would like to buy 2 tickets for Shazam! |
| ASSISTANT: | What city would you like to see this movie? |
| USER: | Ontario, California |
| ASSISTANT: | Ok, I'll check that location for you. |
| USER: | I would prefer the Edwards Ontario Mountain Village, since it's closest to me and my guest. |
| ASSISTANT: | What time is best for you? |
| USER: | Either 4 or 6 pm. |
| ASSISTANT: | I'm sorry, but it looks like the 4:10 and the 6:10 pm showings are sold out. |
| USER: | That's too bad. I really wanted to see that movie. |
| ASSISTANT: | I'm sorry. Is there another movie you would like to see? |
| USER: | How about Captain Marvel at the Edwards Ontario Mountain theater. |
| ASSISTANT: | Show times are 3:45, 7:10 and 10:10 pm. Which would you like? |
| USER: | I am interested in the 7:10 showing. |
| ASSISTANT: | I'm sorry, it looks like the 7:10 showing is also sold out. |
| USER: | Wow, that's too bad. |
| ASSISTANT: | I'm sorry. Is there another movie you would like me to look up? |
| USER: | No, I think I'll pass on the movies tonight since those were the two I really wanted to see. |
| ASSISTANT: | If you want, I can check another theater. |
| USER: | No, that's fine. Thank you for your help. |
| ASSISTANT: | You're welcome. |

Figure 5: Sample one-person, written dialog

instructions.

#### 3.3.2 Pros and cons of self-dialogs

The self-dialog technique renders quality data and avoids some of the challenges seen with the two-person approach. To begin, since the same person is writing both sides of the conversation, we never see misunderstandings that lead to frustration as is sometimes experienced between interlocutors in the two-person approach. In addition, all the self-dialogs follow a reasonable path even when the user is constructing conversations that include understanding errors or other types of dialog glitches such as when a particular choice is not available. As it turns out, crowdsourced workers are quite effective at recreating various types of interactions, both error-free and those containing various forms of linguistic repair. The sample dialog in Figure 5 shows the result of a self-dialog exercise in which workers were told to write a conversation with various ticket availability issues that is ultimately unsuccessful.

Two more benefits of the self-dialog approach are its efficiency and cost effectiveness. We were able to gather thousands of dialogs in just days without transcription or trained agents, and spent roughly six times less per dialog. Despite these advantages, the self-dialog written technique cannot recreate the disfluencies and other more complex error patterns that occur in the two-person spoken dialogs which are important for model accuracy and coverage.

| | |
|---|---|
| USER: | Finally, I need the table to be for three people and 8pm. |
| ASSISTANT: | One moment....OK, I have your table for three (**num.guests.accept**) at 8pm (**time.reservation.accept**) reserved. |

Figure 6: Indicating transaction status with "accept" or "reject"

### 3.4 Annotation

We chose a highly simplified annotation approach for Taskmaster-1 as compared to traditional, detailed strategies which require robust agreement among workers and usually include dialog state and slot information, among other possible labels. Instead we focus solely on API arguments for each type of conversation, meaning just the variables required to execute the transaction. For example, in dialogs about setting up UBER rides, we label the "to" and "from" locations along with the car type (UberX, XL, Pool, etc). For movie tickets, we label the movie name, theater, time, number of tickets, and sometimes screening type (e.g. 3D vs. standard). A complete list of labels is included with the corpus release.

As discussed in Section 3.2.2, to encourage diversity, at times we explicitly ask users to change their mind in the middle of the conversation, and the agents to tell the user that the requested item is not available. This results in conversations having multiple instances of the same argument type. To handle this ambiguity, in addition to the labels mentioned above, the convention of either "accept or "reject" was added to all labels used to execute the transaction, depending on whether or not that transaction was successful.

In Figure 6, both the number of people and the time variables in the assistant utterance would have the ".accept" label indicating the transaction was completed successfully. If the utterance describing a transaction does not include the variables by name, the whole sentence is marked with the dialog type. For example, a statement such as *The table has been booked for you* would be labeled as

| Statistic | Self-dialogs | Two Person |
|---|---|---|
| # unique words | 17,275 | 13,490 |
| # utterances | 110,074 | 132,407 |
| # dialogs | 5000 | 5000 |
| Avg. utterances per dialog | 22.01 | 24.04 |
| Avg. tokens per utterance | 8.62 | 7.54 |
| Perplexity | 16.28 | 6.44 |
| BLEU | 4.73 | 15.16 |
| Joint-Perplexity | 16.44 | 6.04 |
| Joint-BLEU | 5.80 | 13.09 |

Table 2: Statistics comparison: Self-dialogs vs two person corpus both containing 5k dialogs. Perplexity and BLEU are reported for Transformer baseline. Joint-Perplexity and Joint-BLEU are perplexity/BLEU scores from the joint training of self-dialogs and two-person but evaluated with their respective test sets.

**reservation.accept**.

## 4 Dataset Analysis

### 4.1 Self-dialogs vs MultiWOZ

We quantitatively compare our self-dialogs (Section 3.3) with the MultiWOZ dataset in Table 1. Compared to MultiWOZ, we do not ask the users and assistants to stick to detailed scripts and do not restrict them to have conversations surrounding a small knowledge base. Table 1 shows that our dataset has more unique words, and has almost twice the number of utterances per dialog than the MultiWOZ corpus. Finally, when trained with the Transformer (Vaswani et al., 2017) model, we observe significantly higher perplexities and lower BLEU scores for our dataset compared to MultiWOZ suggesting that our dataset conversations are difficult to model. Finally, Table 1 also shows that our dataset contains close to 10 times more real-world named entities than MultiWOZ and thus, could potentially serve as a realistic baseline when designing goal oriented dialog systems. MultiWOZ has only 1338 unique named entities and only 4510 unique values (including date, time etc.) in their datatset.

### 4.2 Self-dialogs vs Two-person

In this section, we quantitatively compare 5k conversations each of self-dialogs (Section 3.3) and two-person (Section 3.2). From Table 2, we find that self-dialogs exhibit higher perplexity ( almost 3 times) compared to the two-person conversations suggesting that self-dialogs are more diverse and contains more non-conventional conversational flows which is inline with the observations in Section-3.3.2. While the number of unique words are higher in the case of self-dialogs, conversations are longer in the two-person conversations. We also report metrics by training a single model on both the datasets together.

### 4.3 Baseline Experiments: Response Generation

We evaluate various seq2seq architectures (Sutskever et al., 2014) on our self-dialog corpus using both automatic evaluation metrics and human judgments. Following the recent line of work on generative dialog systems (Vinyals and Le, 2015), we treat the problem of response generation given the dialog history as a conditional language modeling problem. Specifically we want to learn a conditional probability distribution $P_\theta(U_t|U_{1:t-1})$ where $U_t$ is the next response given dialog history $U_{1:t-1}$. Each utterance $U_i$ itself is comprised of a sequence of words $w_{i_1}, w_{i_2} \dots w_{i_k}$. The overall conditional probability is factorized autoregressively as

$$P_\theta(U_t|U_{1:t-1}) = \prod_{i=1}^{n} P_\theta(w_{t_i}|w_{t_{1:i-1}}, U_{1:t-1})$$

$P_\theta$, in this work, is parameterized by a recurrent, convolution or Transformer-based seq2seq model.

**n-gram**: We consider 3-gram and 4-gram conditional language model baseline with interpolation. We use random grid search for the best coefficients for the interpolated model.

**Convolution**: We use the *fconv* architecture (Gehring et al., 2017) and default hyperparameters from the fairseq (Ott et al., 2019) framework.[2] We train the network with ADAM optimizer (Kingma and Ba, 2015) with learning rate of 0.25 and dropout probability set to 0.2.

**LSTM**: We consider LSTM models (Hochreiter and Schmidhuber, 1997) with and without attention (Bahdanau et al., 2015) and use the tensor2tensor (Vaswani et al., 2018) framework for the LSTM baselines. We use a two-layer LSTM network for both the encoder and the decoder with 128 dimensional hidden vectors.

**Transformer**: As with LSTMs, we use the tensor2tensor framework for the Transformer model.

[2]https://github.com/pytorch/fairseq

| Baseline Models | PPL | BLEU | Ratings (LIKERT) | Rank |
|---|---|---|---|---|
| GPT-2 (117M) | - | 0.26 | - | - |
| 3-gram | 38.12 | 0.20 | - | - |
| 4-gram | 34.49 | 0.21 | - | - |
| LSTM | 25.73 | 4.45 | - | - |
| Convolution | 21.25 | 5.09 | 2.89 | 3 |
| LSTM-attention | 20.05 | 5.12 | **3.51** | 2 |
| Transformer | **18.19** | **6.11** | 3.22 | **1** |

Table 3: Evaluation of various seq2seq architectures (Sutskever et al., 2014) on our self-dialog corpus using both automatic evaluation metrics and human judgments. Human evaluation ratings in the 1-5 LIKERT scale (higher the better), and human ranking are averaged over 500 x 3 ratings (3 crowdsourced workers per rating).

Our Transformer (Vaswani et al., 2017) model uses 256 dimensions for both input embedding and hidden state, 2 layers and 4 attention heads. For both LSTMs and Transformer, we train the model with ADAM optimizer ($\beta_1 = 0.85$, $\beta_2 = 0.997$) and dropout probability set to 0.2.

**GPT-2**: Apart from supervised seq2seq models, we also include results from pre-trained GPT-2 (Radford et al., 2019) containing 117M parameters.

We evaluate all the models with perplexity and BLEU scores (Table 3). Additionally, we perform two kinds of human evaluation - Ranking and Rating (LIKERT scale) for the top-3 performing models - Convolution, LSTM-attention and Transformer. For the ranking task, we randomly show 500 partial dialogs and generated responses of the top-3 models from the test set to three different crowdsourced workers and ask them to rank the responses based on their relevance to the dialog history. For the rating task, we show the model responses individually to three different crowdsourced workers and ask them to rate the responses on a 1-5 LIKERT scale based on their appropriateness to the dialog history. From Table-4, we see that inter-annotator reliability scores (Krippendorfs Alpha) are higher for the ranking task compared to the rating task. From Table 3, we see that Transformer is the best performing model on automatic evaluation metrics. It is interesting to note that there is a strong correlation between BLEU score and human ranking judgments.

| Evalation method | Inter-Annotator Reliability (Krippendorfs Alpha) |
|---|---|
| Rating (1-5 LIKERT) | 0.21 |
| Ranking | 0.29 |

Table 4: Inter-Annotator Reliability scores of seq2seq model responses computed for 500 self-dialogs from the test set, each annotated by 3 crowdsourced workers.

| Model | Micro F1 (%) |
|---|---|
| Transformer | 48.73 |
| Transformer + copy | 51.79 |

Table 5: API Argument prediction accuracy for Self-dialogs. API arguments are annotated as spans in the utterances.

### 4.4 Baseline Experiments: Argument Prediction

Next, we discuss a set of baseline experiments for the task of argument prediction. API arguments are annotated as spans in the dialog (Section 3.4). We formulate this problem as mapping text conversation to a sequence of output arguments. Apart from the seq2seq Transformer baseline, we consider an additional model - an enhanced Transformer seq2seq model where the decoder can choose to copy from the input or generate from the vocabulary (Merity et al., 2017; Gu et al., 2016). Since all the API arguments are input spans, the copy model having the correct inductive bias achieves the best performance.

## 5 Conclusion

To address the lack of quality corpora for data-driven dialog system research and development, this paper introduces Taskmaster-1, a dataset that provides richer and more diverse language as compared to current benchmarks since it is based on unrestricted, task-oriented conversations involving more real-word entities. In addition, we present two data collection methodologies, both spoken and written, that ensure both speaker diversity and conversational accuracy. Our straightforward, API-oriented annotation technique is much easier for annotators to learn and simpler to apply. We give several baseline models including state-of-the-art neural seq2seq architectures, provide qualitative human performance evaluations for these models, and find that automatic evaluation metrics correlate well with human judgments.

## References


Amazon. 2019. Alexa skills.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *ICLR*.

Antoine Bordes, Y-Lan Boureau, and Jason Weston. 2017. Learning end-to-end goal-oriented dialog. *ICLR*.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. 2018. Multiwoz - a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. *EMNLP*.

Wallace Chafe and Deborah Tannen. 1987. The relation between written and spoken language. *Annual Review of Anthropology*.

J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. 2017. Convolutional Sequence to Sequence Learning. *Arxiv*.

Google. 2019. Actions on google.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *ACL*.

Matthew Henderson, Blaise Thomson, and Steve Young. 2013. Deep neural network approach for the dialog state tracking challenge. *SIGDIAL*.

Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara Sainath, and Brian Kingsbury. 2012. Deep neural networks for acoustic modeling in speech recognition. *Signal Processing Magazine*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*.

Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, and Gaurav Agrawal et al. 2017. In-datacenter performance analysis of a tensor processing unit. *ISCA*.

John F Kelley. 1984. An iterative design methodology for user-friendly natural language office information applications. *ACM Transactions on Information Systems (TOIS)*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *ICLR*.

Ben Krause, Marco Damonte, Mihai Dobre, Daniel Duma, Joachim Fainberg, Federico Fancellu, Emmanuel Kahembwe, Jianpeng Cheng, and Bonnie Webber. 2017. Edina: Building an open domain socialbot with self-dialogues. *Arxiv*.

Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature*.

Ryan Lowe, Nissan Pow, Iulian V. Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. *SIGDIAL*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. *ICLR*.

Nikita Moghe, Siddhartha Arora, Suman Banerjee, and Mitesh M. Khapra. 2018. Towards exploiting background knowledge for building conversation systems. *EMNLP*.

Aron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alexander Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. Wavenet: A generative model for raw audio. *Arxiv*.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. *NAACL Demonstrations*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *Arxiv*.

Antoine Raux, Brian Langner, Alan W Black, and Maxine Eskenazi. 2003. Lets go: Improving spoken dialog systems for the elderly and non-natives. *Eurospeech*.

Lina Maria Rojas-Barahona, Milica Gasic, Nikola Mrksic, Pei-Hao Su, Stefan Ultes, Tsung-Hsien Wen, Steve J. Young, and David Vandyke. 2017. A network-based end-to-end trainable task-oriented dialogue system. *EACL*.

Iulian Vlad Serban, Ryan Lowe, Peter Henderson, Laurent Charlin, and Joelle Pineau. 2017. A survey of available corpora for building data-driven dialogue systems. *nnn*.

Dave Steinkrau, Patrice Y. Simard, and Ian Buck. 2005. Using gpus for machine learning algorithms. *ICDAR*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *NeurIPS*.

Ashish Vaswani, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan N. Gomez, Stephan Gouws, Llion Jones, Łukasz Kaiser, Nal Kalchbrenner, Niki Parmar, Ryan Sepassi, Noam Shazeer, and Jakob Uszkoreit. 2018. Tensor2tensor for neural machine translation. *CoRR*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *NeurIPS*.

Oriol Vinyals and Quoc V. Le. 2015. A neural conversational model. *Arxiv*.

Joseph Weizenbaum. 1966. Eliza a computer program for the study of natural language communication between man and machine. *Computational Linguistics*.

Jason Williams, Antoine Raux, and Matthew and Henderson. 2016. The dialog state tracking challenge series: A review. *Dialog and Discourse*.

Jason D. Williams and Steve J. Young. 2007. Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language*.

# Multi-Domain Goal-Oriented Dialogues (MultiDoGO): Strategies toward Curating and Annotating Large Scale Dialogue Data

**Denis Peskov**[1*†], **Nancy Clarke**[†2], **Jason Krone**[†2],
**Brigitta Fodor**[*3], **Yi Zhang**[2], **Adel Youssef**[2] **and Mona Diab**[2]
[1]University of Maryland, [2]Amazon AWS AI, [3]Wayfair
dpeskov@cs.umd.edu, bfodor@wayfair.com
{njc, kronej, yizhng, adel, diabmona}@amazon.com

## Abstract

The need for high-quality, large-scale, goal-oriented dialogue datasets continues to grow as virtual assistants become increasingly widespread. However, publicly available datasets useful for this area are limited either in their size, linguistic diversity, domain coverage, or annotation granularity. In this paper, we present strategies toward curating and annotating large scale goal oriented dialogue data. We introduce the `MultiDoGO` dataset to overcome these limitations. With a total of over 81K dialogues harvested across six domains, `MultiDoGO` is over 8 times the size of `MultiWOZ`, the other largest comparable dialogue dataset currently available to the public. Over 54K of these harvested conversations are annotated for intent classes and slot labels. We adopt a Wizard-of-Oz approach wherein a crowd-sourced worker (the "customer") is paired with a trained annotator (the "agent"). The data curation process was controlled via biases to ensure a diversity in dialogue flows following variable dialogue policies. We provide distinct class label tags for agents vs. customer utterances, along with applicable slot labels. We also compare and contrast our strategies on annotation granularity, i.e. turn vs. sentence level. Furthermore, we compare and contrast annotations curated by leveraging professional annotators vs the crowd. We believe our strategies for eliciting and annotating such a dialogue dataset scales across modalities and domains and potentially languages in the future. To demonstrate the efficacy of our devised strategies we establish neural baselines for classification on the agent and customer utterances as well as slot labeling for each domain.

[*]Work performed while at Amazon AWS AI
[†]Denotes equal contribution

## 1 Introduction

Modern Natural Language Understanding (NLU) frameworks for dialogues are by definition data hungry. They require large amounts of training data representative of goal oriented conversations reflecting both context and diversity. But human responses in goal-oriented dialogues are less predictable than automated systems (Bordes et al., 2016). For example, "Please do this" cannot be interpreted without a broader context. Only by seeing previous utterances, such as requests to book a flight on a specific day to a specific destination, can this task be performed. Additionally, a single intent can be phrased in multiple ways depending on context; "book my flight", "finalize my reservation", "Yes, the 6 pm one" may all be referring to a flight-booking intent. Hence, entire conversations, rather than independent utterances, must be collected. Such data is even more pertinent to modeling NLU and related tasks as they require large, varied, and ideally human-generated datasets. Moreover, recent work (Dong et al., 2015; Devlin et al., 2018) has shown the benefit of applying joint-training and transfer learning techniques to natural language processing tasks. However, these approaches have yet to become widely used in dialogue tasks, due to a lack of large-scale datasets. Furthermore, the latest state of the art end-to-end neural approaches benefit from such training data even more so than past work on goal-oriented dialogues structured around slot filling (Lemon et al., 2006; Wang and Lemon, 2013). One way to simulate data—and not risk releasing personally identifying information—for a domain is to use a Wizard-of-Oz data gathering technique, which requires that participants in a conver-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4526–4536, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| Role | Turn | Annotations |
|---|---|---|
| A | Hey there! Good morning. You're connected to LMT Airways. How may I help you? | DA = { elicitgoal } |
| C | Hi, I wonder if you can confirm my seat assignment on my flight tomorrow? | IC = { SeatAssignment } |
| A | Sure! I'd be glad to help you with that. May I know your last name please? | DA = { elicitslot } |
| C | My last name is Turker. | IC = { contentonly },<br>SL = {Name : Turker } |
| A | Alright Turker! Could you please share the booking confirmation number? | DA = { elicitslot } |
| C | I believe it's AMZ685. | IC = { contentonly },<br>SL = { Confirmation Number : AMZ685 } |
| ... | ... | ... |

Table 1: A segment of a dialogue from the airline domain annotated at the turn level. This data is annotated with agent dialogue acts (DA), customer intent classes (IC), and slot labels (SL). Roles C and A stand for "Customer" and "Agent", respectively.

sation fulfill a role (Kelley, 1984). This approach has been used in popular public goal-oriented datasets: DSTC and MultiWOZ (Williams et al., 2016; Budzianowski et al., 2018).

Conversations between people and automated systems occur with increasing frequency, especially in customer service. Customers reach out to agents, which could be automated bots or real individuals, to achieve a domain-specific goal. This creates a disparate conversation: agents are incentivized to operate within a set procedure and convey a patient and professional tone. In contrast, customers do not have this incentive. However, to date, the largest available multi-domain goal-oriented dialogue dataset assigns similar dialogue act annotations to both agents and customers (Budzianowski et al., 2018).

To solve the aforementioned challenges, we present our efforts to curate, annotate, and evaluate a large scale multi-domain set of goal oriented dialogues. The dataset is primarily gathered from workers in the crowd paired with professional annotators. The dataset elicited, MultiDoGO, comprises over 86K raw conversations of which 54,818 conversations are annotated at the turn level. We investigate multiple levels of annotation granularity. We annotate a subset of the data on both turn and sentence levels. A turn is defined as a sequence of one or more speech/text sentences by a participant in a conversation. A sentence is a period delimited sequence of words in a turn. A turn may comprise one or more sentences. We do use the term utterance to refer to a unit (turn or sentence, spoken or written by a participant).[1]

In our devised annotation strategy, we distinguish between dialogue speech acts for agents vs. customers. In MultiDoGO, the agents' speech acts [DA] are annotated with generic class labels common across all domains, while customer speech acts are labeled with intent classes [IC]. Moreover, we annotate customer utterances with the appropriate slot labels [SL], which consist of the SL span and corresponding tokens with that SL tag. We present the strategies we use to curate and annotate such data given its contextual setting. We furthermore illustrate the efficacy of our devised approaches and annotation decisions against intrinsic metrics and via extrinsic evaluation, namely by applying neural baselines for DA, IC and SL classification leveraging joint models.

## 2 Existing Dialogue Datasets

There are multiple existing goal-oriented dialogue collections generated by humans through Wizard-of-Oz techniques. The Dialog State Tracking Challenge, *aka* Dialog Systems Technology Challenge, (DSTC) spans 8 iterations and entails the domains of bus timetables, restaurant reservations, and hotel bookings, travel, alarms, movies, etc. (Williams et al., 2016). Frames (Asri et al., 2017) has 1369 dialogues about vacation packages. MultiWOZ contains 10,438 dialogues about Cambridge hotels and restaurants (Budzianowski et al., 2018). There are several dialogue datasets that specialize in a single domain. ATIS (Hemphill et al., 1990) comprises speech data about airlines structured around formal airline flight tables. Similarly, the Google Airlines dataset purportedly contains 400,000 templated dialogues about airline reservations (Wei

[1] We acknowledge that the term utterance is controversial in the literature (Pareti and Lando, 2018)

et al., 2018).[2] The Ubuntu Dialogue Corpus has over a million dialogues about Ubuntu technical support (Lowe et al., 2015).

On the other hand, Chit-chat style dialogues without goals have been popular since ELIZA and have been investigated with neural techniques (Weizenbaum, 1966; Li et al., 2016, 2017). However, these datasets cannot be used for modeling goal-oriented tasks. Related dialogue dataset collections used for Sequential Question Answering rely on dialogue to answer questions, but the task is notably different from our use case of modeling goal oriented conversational AI, hence leading to different evaluation considerations (Reddy et al., 2019; Choi et al., 2018).

## 3 MultiDoGO Dataset Curation

### 3.1 Data Collection Procedure

We employ both internal data associates, who we train, and crowd-sourced workers from Mechanical Turk (MTurkers) to generate conversational data using a Wizard-of-Oz approach. In each conversation, the data associates assumes the role of an agent while the MTurkers act as customers. In an effort to source competent MTurkers, we require that each MTurker have a Human Intelligence Task (HIT) accuracy minimum of 90%, a location in the United States, and have completed a significant number of HITs in the past. To facilitate goal-oriented conversations between the customer and agent, we give each agent a prompt listing the supported request types (dialog acts)and pieces of information (slots) needed to complete each request. We also specify criteria such as minimal conversation length, number of goals,number of complex requests, etc, to increase conversation diversity. See Figure 2 for an example prompt. In addition, we explicitly request that neither agents nor customers use any personally identifiable information. At an implementation level,we create a custom, web interface for the MTurkers and data associates that displays our instructions next to the current dialogue. This allows each participant to quickly refer to our guidelines without stopping the conversation. Despite following a familiar wizard-of-oz elicitation procedure, and curating data for multiple domains in a fashion similar to previous data collection efforts such as MultiWOZ, MultiDoGO comprises more varied domains, it is collected at an unprecedented scale, and it is curated with control over generating explicit biases in the conversations to allow for diverse conversation representation. To our knowledge this is a novel collection strategy as we explicitly guide/prod the participants in a dialogue to engage in conversations with specific biases such as intent change, slot change, multi-intent, multiple slot values, slot overfilling and slot deletion. For example, in the Fast Food domain, participants were instructed to pretend that they were ordering fast food from a drive-thru. After making their initial order, they were instructed to change their mind about what they were ordering ("I'd like a burger. No wait, can you make that a chicken sandwich?"). In the Financial domain, we asked participants to make sure that they requested multiple intents such as "I'd like to find my routing number and check my balance."[3] To that end, our collection procedure deliberately attempts to guide the dialogue flow to ensure diversity in dialogue policies.

### 3.2 Domain Selection

Our primary criteria for domain selection are twofold: covering a broad sweep of industries that use goal-oriented dialogues and selecting domains where conversational interfaces are already in use or likely to be implemented in the future. This set of criteria is especially well matched with domains that frequently involve customer support. Furthermore, there is a shortage of publicly available data in the domains we provide, such as Fast Food and Finance. To fulfill both of these needs, we include multiple domains in the MultiDoGO dataset. Ultimately, we curate conversations for six domains: Airline, Fast Food, Finance, Insurance, Media, and Software Support. When considered independently, the corpus of dialogues for each of these domains is the largest collection of human-elicited dialogues available for financial advice and help, media support, enterprise software support (non-technical level support, unlike the Ubuntu forum dataset (Lowe et al., 2015)), fast food, and insurance.

**Domains in our Data:**[4] **Airline** domain dialogues focus on booking airline flights, select-

[2]The Google Airlines dataset has not been released to date.

[3]For a full list of conversational biases with examples, please see the appendix.

[4]Detailed data collection and annotation schemata for each domain can be found in the appendix.

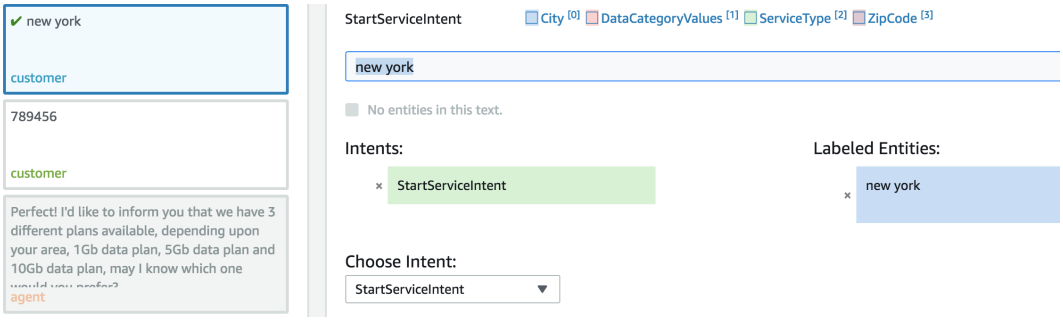


Figure 1: Crowd sourced annotators select an intent and choose a slot in our custom-built Mechanical Turk interface. Entire conversations are provided for reference. Detailed instructions are provided to users, but are not included in this figure. Options are unique per domain.

ing or changing seat assignments, and requesting boarding passes; **Fast Food** domain is the least similar to the others, as the intents primarily involve ordering food and the slots quantify their order. For example, the OrderBurgerIntent contains slots for size, quantity, and ingredients; **Finance** domain simulates dialogues a customer may have with a bank. These include opening a bank account, checking their balance, and reporting a lost credit card; **Insurance** domain simulates users calling about their insurance policy or requesting the fulfillment of a policy on their car or phone; **Media** domain simulates dialogues a customer may have ordering a service or paying bills related to telecommunications. This is our largest domain; **Software** domain involves customers inquiring about software services: products, outages, promotions, and bills. The majority of intents are domain specific.

### 3.3 Domain Schemata and Guidelines

Prior to dialogue collection, we develop schemata for each domain. These schemata are the set of slot labels, slot value types, and intents that pertain to the domain. To determine which slots, values, intents, and dialog acts to include, we rely on real word reference points. For instance, we populate slots for the Fast Food domain by identifying menu items, such as sodas, that are shared among popular fast food menus. Using this schema, we then write two sets of instructions. One set of instructions is for the "agents" and the other is for the "customers". The agents' instructions are meticulously detailed as we expect them to "structure" the conversation and appropriately respond to out of domain requests. Since our agents are trained for their role, we have high confidence in their ability to follow complex guidelines. In contrast, taking into consideration that the customer role is to be carried out by crowd-sourced workers, i.e. lay people, we create simplified instructions that are less detailed and shorter in length. For each task, we provide an annotated conversation, explain each answer option, and (most importantly) provide examples. Before scaling up our data collection, we run a pilot for each task and identify commonly missed questions. We use this pilot process to revise the instructions and add relevant examples iteratively.

## 4 Data Annotation

### 4.1 Annotated Dialogue Tasks

Our dataset has three types of annotation: Agent dialogue acts [DA], customer intent classes [IC], and slot labels [SL]. We intentionally decouple Agent and customer speech act tags into the categories DA and IC, respectively, to produce more fine-grained speech act tags than past iterations of dialog datasets. Intuitively, agent DAs are consistent across domains and more abstract in nature, since agents have a standard form of response. On the other hand, customer ICs are domain-specific and can entail reserving a hotel room or ordering a burger, depending on the domain. A conversation example with annotations is provided in Table 1.

**Agent Dialogue Acts (DA)** Agent dialogue acts are the most straightforward of our annotation tasks. There are eight possible DAs in all domains: *ElicitGoal, ElicitSlot, ConfirmGoal, ConfirmSlot, EndGoal, Pleasantries, Other*. The names are self-explanatory. *Elicit Goal/Slot* indicates that the agent is gathering information. *Confirm Goal/Slot* indicates that the agent is confirming previously provided information. The *EndGoal* and *Pleasantries* tags, identify non-task re-

lated actions. *Other* indicates that the selected utterance was not one of the other possible tags. Agent dialogue acts are consistent across domains and are often abstract (e.g. ElicitIntent, ConfirmSlot).

**Customer Intent Classes (IC):** Unlike Agent DA, customer IC vary for each domain and are more concrete. For example, the Airline domain has a "BookFlight" IC, Fast Food has an "OrderMeal" IC, and Insurance has an "OrderPolicy" IC in our annotation schema. Customer intents can overlap across domains (e.g. OpeningGreeting, ClosingGreeting) and other times be domain specific (e.g. RequestCreditLimitIncrease, OrderBurger, BookFlight).

**Slot Labels (SL):** Slot Labeling (SL) is a task contingent on Customer Intent Classes. Certain intents require that additional information, namely slot values, be captured. For instance, to open a bank account, one must solicit the customer's social security number. Slots can overlap across intents (e.g. Name, SSN Number) or they can be unique to a domain-specific intent (e.g. CarPolicy).

### 4.2 Data Annotation Procedure

Our annotators utilize a web interface, depicted in Figure 1, to select the appropriate intent class for an utterance out of a list of provided options. To annotate slot labels, our annotators use their cursors to highlight slot value character spans within an utterance and then select the corresponding slot label from a list of options. The output of this slot labeling process is a list of ⟨slot-label, slot-value, span⟩ triplets for each utterance.

### 4.3 Annotation Design Decisions

**Decoupled Agents and Customers Label sets:** Agents and customers have notably different goals and styles of communication. However, past dialogue datasets do not make this distinction at speech act schema level. Specificity is important for handling unique customer requests, but a relatively formulaic approach is required of agents across different industries. Our distinction between the customer and agent roles creates training data for a bot that explicitly simulates agents.

**Annotation Unit Granularity: Sentence vs. Turn Level** An important decision, which is often under discussed, is the proper semantic unit of text to annotate in a dialogue. Commonly, datasets provide annotations at the turn level (Budzianowski et al., 2018; Asri et al., 2017; Mihail et al., 2017). However, turn level annotations can introduce confusion for IC datasets, given multiple intents may be present in different sentences of a single turn. For instance, consider the turn "I would like to book a flight to San Francisco. Also, I want to cancel a flight to Austin." Here, the first sentence has the BookFlight intent and the second sentence has the CancelFlight intent. An turn level annotation of this utterance would yield the multi-class intent (BookFlight, CancelFlight). In contrast, a sentence level annotation of this utterance identifies that the first sentence corresponds to BookFlight while the second corresponds to CancelFlight. We annotate a subset our data, 2,500 conversation per domain for 15,000 conversations in total, at the sentence as well as turn level to access the impact of this design choice on downstream performance. The remainder of our dataset is annotated only at the turn level.

| ISAA | | |
|---|---|---|
| DA | IC | SL |
| 0.701 | 0.728 | 0.695 |

Table 2: Dialogue act (DA), Intent class (IC), and slot labeling (SL) Inter Source Annotation Agreement (ISAA) scores quantifying the agreement of crowd sourced and professional annotations.

**Professional vs. Crowd-Sourced Workers for annotation** For annotation, we compare and contrast professional annotators to crowd sourced annotators on a subset of data. Professional annotators assign DA, IC, and SL tags to the 15,000 conversations annotated at both the turn and sentence level; statistics for these conversations are given in table 6. In an effort to decrease annotation cost, we employ crowd source annotators via Mechanical Turk to label an additional 54,818 conversations rated as Good or Excellent quality during data collection. We provide statistics for this set of crowd annotated data in Table 3. To compare the quality of crowd sourced annotations against professional annotations, we use both strategies to annotate a shared subset of 8,450 conversations. We devise an Inter Source Annotation Agreement (ISAA) metric to quantify the agreement of these crowd sourced and professionally sourced annotations. ISAA is a relaxation of Cohen Kappa, in-

tended to count partial agreement of multi-tag labels. ISAA defines two sets of tags, $A$ and $B$, to be in agreement if there is at least one "shared" tag in both $A$ and $B$. $A$ and $B$ reflect the majority labels agreed upon per source (professionals or crowd workers). Using ISAA we find that crowd sourced and professional annotations have a substantial degree of shared annotations. We report ISAA for the DA, IC, and SL tasks in Table 2.

### 4.4 Quality Control

We institute three processes to enforce data quality. During data collection, our data associates report on the quality of each conversation. Specifically, the data associates grade the conversation on a scale from "Unusable", "Poor", "Good", to "Excellent". They were provided with guidelines to help decide on the chosen rating such as coherence, whether the dialogue achieved the purported goal, etc. To ensure high data quality we only utilize conversations with "Good" or "Excellent" ratings in subsequent annotation.

Secondly, for data annotation, each conversation is annotated at least twice. We remove inconsistent annotations by selecting the annotation given by the majority of annotators per item. We calculate inter-annotator agreement with Fleiss Kappa and find "substantial agreement", according to the metric. Our annotators must pass a qualification test as well as maintain an on-going level of accuracy in randomly distributed test questions throughout their annotation. Third, we preprocess our data to remove issues such as duplicate conversations and improperly entered slot value spans. We refer readers to our discussion of preprocessing in Section 5 for further detail.

### 4.5 Dataset Characterization and Statistics

MultiDoGO dataset is more diverse by virtue of covering more domains, but more importantly, it is more controlled since it was curated rather than being scraped from existing data sources that are not necessarily synchronous (Ubuntu). Table 3 shows the statistics for MultiDoGO raw conversations harvested, rated as Excellent or Good, and annotated for DA, IC and SL.

Table 4 shows the number of conversations per domain reflecting the specific biases used.

MultiDoGO is several orders of magnitude larger than comparable datasets as reflected in nearly every dimension: the number of conversations, the length of the conversation, the number of domains, and the diversity of the utterances used. Table 5 illustrates a comparative statistics to existing data sets.

**Agent Instructions**

Imagine you work at a bank. Customers may contact you about the following set of issues: checking account balances (checking or savings), transferring money between accounts, and closing accounts.

**GOAL**: Answer the customer's question(s) and complete their request(s).

For any request, you will need to collect at least the following information to be able to identify the customer: name, account PIN *or* last 4 digits of SSN.

For giving information on balances, or for closing accounts, you will also need the last 4 digits of the account number.

For transferring money, you will also need: last 4 digits of account to move from, last 4 digits of account to move to, and the sum of money to be transferred.

Your customer may ask you to do only one thing; that's okay, but make sure you confirm you achieved everything the Customer wanted before completing the conversation. Don't forget to signal the end of the conversation (see General guidelines)

Figure 2: Agents are provided with explicit fulfillment instructions. These are quick-reference instructions for the Finance domain. Agents serve as one level of quality control by evaluating a conversation between Excellent and Unusable.

We provide summary statistics for the subset of our data annotated at both turn and sentence granularity in Table 6. This describes the total size of the data per domain in number of conversations, turns, the unique number of intents and slots, and inter-annotator agreement (IAA) for both turn and sentence level annotations. It is worth observing that the DA annotations achieve a much higher IAA in Sentence level annotations compared to Turn level annotation, most notably in the Fast Food domain. IC and SL annotations reflect a slightly higher IAA in Turn level annotation granularity compared to Sentence level.

## 5 Dialogue Classification Baselines

To establish baseline performance for the MultiDoGO dataset we pre-process, create dataset splits, and evaluate the performance of three baseline models for each domain.

**Pre-processing:** We pre-process the corpus of dialogues for each domain to remove duplicate

| Domain | Elicited | Good/Excellent | IC/SL | DA/IC/SL |
|---|---|---|---|---|
| Airline | 15100 | 14205 | 7598 | 6287 |
| Fast Food | 9639 | 8674 | 7712 | 4507 |
| Finance | 8814 | 8160 | 8002 | 6704 |
| Insurance | 14262 | 13400 | 7799 | 7434 |
| Media | 33321 | 32231 | 19877 | 12891 |
| Software | 5562 | 4924 | 3830 | 2753 |
| **Total** | **86698** | **81594** | **54818** | **40576** |

Table 3: Total number of conversations per domain: raw conversations Elicited; Good/Excellent is the total number of conversations rated as such by the agent annotators; (IC/SL) is the number of conversations annotated for Intent Classes and Slot Labels only; (DA/IC/SL) is the total number of conversations annotated for Dialogue Acts, Intent Classes, and Slot Labels.

| Bias | Airlines | Fast Food | Finance | Insurance | Media | Software |
|---|---|---|---|---|---|---|
| IntentChange | | 1443 | | | | |
| MultiIntent | 2200 | 1913 | 1799 | 1061 | 607 | 2295 |
| MultiValue | | 354 | | | | |
| Overfill | | | 1486 | 2763 | | |
| SlotChange | 4207 | 2011 | 2506 | 3321 | 570 | 2085 |
| SlotDeletion | | 333 | | | | |
| **Total** | 6407 | 6054 | 5791 | 7145 | 1177 | 4380 |

Table 4: Number of conversations per domain collected with specific biases. Fast Food had the maximum number of biases. MultiIntent and SlotChange are the most used biases.

| Metric | DSTC 2 | WOZ2.0 | FRAMES | KVRET | M2M | MULTIWOZ | MULTIDOGO |
|---|---|---|---|---|---|---|---|
| Number of Dialogues | 1,612 | 600 | 1,396 | 2,425 | 1,500 | 8,438 | 40,576 |
| Total Number of Turns | 23,354 | 4,472 | 19,986 | 12,732 | 14,796 | 115,424 | 813,834 |
| Total Number of Tokens | 199,431 | 50,264 | 251,867 | 102,077 | 121,977 | 1,520,970 | 9,901,235 |
| Avg. Turns per Dialog | 14.49 | 7.45 | 14.60 | 5.25 | 9.86 | 15.91 | 20.06 |
| Avg. Tokens Per Turn | 8.54 | 11.24 | 12.60 | 8.02 | 8.24 | 13.18 | 12.16 |
| Total Unique Tokens | 986 | 2,142 | 12,043 | 2,842 | 1,008 | 24,071 | 70,003 |
| Number of Unique Slots | 8 | 4 | 61 | 13 | 14 | 25 | 73 |
| Number of Slot Values | 212 | 99 | 3,871 | 1,363 | 138 | 4,510 | 55,816 |
| Number of Domains | 1 | 1 | 1 | 3 | 1 | 7 | 6 |
| Number of Tasks | 1 | 1 | 1 | 2 | 2 | 2 | 3 |

Table 5: `MultiDoGO` is several times larger in nearly every dimension to the pertinent datasets as selected by Budzianowski et al. (2018). We provide counts for the training data, except for FRAMES, which does not have splits. Our number of unique tokens and slots can be attributed to us not relying on carrier phrases.

| Domain | #Conv | #Turn | #Turn/Conv | #Sentence | #Intent | #Slot | Turn-level IAA | Sentence-level IAA |
|---|---|---|---|---|---|---|---|---|
| Airline | 2,500 | 39,616 | 15.8 (15) | 66,368 | 11 | 15 | 0.514/0.808/0.802 | 0.670/0.788/0.771 |
| Fast Food | 2,500 | 46,246 | 18.5 (18) | 73,305 | 14 | 10 | 0.314/0.700/0.624 | 0.598/0.725/0.607 |
| Finance | 2,500 | 46,001 | 18.4 (18) | 70,828 | 18 | 15 | 0.521/0.827/0.772 | 0.700/0.735/0.714 |
| Insurance | 2,500 | 41,220 | 16.5 (16) | 67,657 | 10 | 9 | 0.521/0.862/0.848 | 0.703/0.821/0.826 |
| Media | 2,500 | 35,291 | 14.1 (14) | 65,029 | 16 | 16 | 0.499/0.812/0.725 | 0.678/0.802/0.758 |
| Software | 2,500 | 40,093 | 16.0 (15) | 70,268 | 16 | 15 | 0.508/0.748/0.745 | 0.709/0.764/0.698 |

Table 6: Data statistics by domain. Conversation length is shown in *average (median)* number of turns per conversation. Inter-annotator agreement (IAA) is measured with Fleiss' Kappa for the three annotation tasks: Agent DA (DA), Customer IC (IC), and Slot Labeling (SL).

conversations and utterances with inconsistent annotations. The most common source of inconsistent annotations in our dataset is imprecise selection of slot label spans by annotators, which results in sub-token slot labels. While much of this inconsistent data could likely be recovered by mapping each character span to the nearest token span, we drop these utterances to ensure these errors have no effect on our experimental results. Our post-processed data is pruned to approximately 90% of the original size. We form splits for each domain at the conversation level by randomly assigning 70% of conversations to train, 10% to development, and 20% to test. Conversation level splits enable the application of contextual models to our dataset, as each conversation is assigned to a single

| | | Airline | | | Fast Food | | | Finance | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Model | Annot | DA | IC | SL | DA | IC | SL | DA | IC | SL |
| MFC | S | 60.57 | 33.69 | 38.71 | 57.14 | 25.42 | 61.92 | 51.73 | 37.37 | 34.07 |
| LSTM | S | 97.20 | 90.84 | 74.16 | 90.40 | 86.09 | 72.93 | 93.90 | 90.06 | 69.09 |
| ELMO | S | **97.32** | **91.88** | **86.55** | **91.03** | **87.95** | **77.51** | **94.07** | **91.15** | **77.36** |
| MFC | T | 33.04 | 32.79 | 37.73 | 33.07 | 25.33 | 61.84 | 36.52 | 38.16 | 34.31 |
| LSTM | T | **84.25** | 89.15 | 75.78 | **66.41** | 87.35 | 73.57 | 76.19 | 92.30 | 70.92 |
| ELMO | T | 84.04 | **89.99** | **85.64** | 65.69 | **88.96** | **79.63** | **76.29** | **94.50** | **79.47** |
| | | Insurance | | | Media | | | Software | | |
| Model | Annot | DA | IC | SL | DA | IC | SL | DA | IC | SL |
| MFC | S | 56.87 | 38.37 | 53.75 | 57.02 | 30.42 | 82.06 | 58.14 | 33.32 | 53.96 |
| LSTM | S | **94.73** | 93.30 | 75.27 | **94.27** | 92.35 | 90.84 | 93.22 | 90.95 | 69.48 |
| ELMO | S | 94.63 | **94.27** | **88.45** | **94.27** | **93.32** | **93.99** | **93.66** | **92.25** | **76.04** |
| MFC | T | 36.39 | 39.42 | 54.66 | 29.90 | 31.82 | 78.83 | 36.79 | 33.78 | 54.84 |
| LSTM | T | **75.37** | 94.75 | 76.84 | **77.94** | 94.35 | 87.33 | **83.32** | 89.78 | 72.34 |
| ELMO | T | 75.34 | **95.39** | **89.51** | 77.81 | **94.76** | **91.48** | 82.97 | **90.85** | **76.48** |

Table 7: Dialogue act (DA), Intent class (IC), and slot labeling (SL) F1 scores by domain for the majority class, LSTM, and ELMO baselines on data annotated at the sentence (S) and turn (T) level. Bold text denotes the model architecture with the best performance for a given annotation granularity, i.e. sentence or turn level. Red highlight denotes the model with the best performance on a given task across annotation granularities.

| | Airline | | Fast Food | | Finance | | Insurance | | Media | | Software | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Annot | Single | Joint | Single | Joint | Single | Joint | Single | Joint | Single | Joint | Single | Joint |
| S | 97.32 | **97.44** | 91.03 | **91.26** | 94.07 | **94.27** | 94.63 | **94.99** | 94.27 | **94.47** | 93.66 | **94.00** |
| T | 84.04 | **84.64** | **65.69** | 65.35 | **76.29** | 75.68 | 75.34 | **75.89** | 77.81 | **78.56** | 82.97 | **83.76** |

Table 8: Joint training of ELMo on all agent DA data leads to a slight increase in test performance. However, we expect stronger joint models that leverage transfer learning should see a larger improvement. Bold text denotes the training strategy, i.e. single domain (Base) or multi-domain (Joint), with the best performance for a given annotation granularity. Red highlight denotes the strategy with the highest DA F1 score across annotation granularities.

split. However, our conversation level splits result in imbalanced intent and slot label distributions.

**Models:** We evaluate the performance of two neural models on each domain. The first is a bi-directional LSTM (Hochreiter and Schmidhuber, 1997) with GloVe word embeddings, a hidden state of size 512, and two fully connected output layers for slot labels and intent classes respectively. The second model, ELMO, is similar to the LSTM architecture but it addiitonally uses pre-trained ELMO (Peters et al., 2018) embeddings in addition to GloVe word embeddings, which are kept frozen during training. We combine these ELMO and GloVe embeddings via concatenation. As a sanity check, we also include a most frequent class (MFC) baseline. The MFC baseline assigns the most frequent class label in the training split to every utterance $u'$ in the test split for both DA and IC tasks. To adapt the MFC baseline to SL, we compute the most frequent slot label MFC$(w)$ for each word type $w$ in the training set. Then given a test utterance $u'$, we assign the pre-computed, most frequent slot MFC$(w')$ to each word $w' \in u'$ if $w'$ is present in the training set. If a given word $w' \in u'$ is not present the training set, we assign the *other* slot label, which denotes the absence of a slot, to $w'$. We utilize the AllenNLP (Gardner et al., 2017) library to implement these models and evaluate our performance. We use the Adam optimizer (Kingma and Ba, 2014) with a learning rate of 0.001 to train the LSTM and ELMO models for 50 epochs, using batch sizes 256 and 128, respectively. In addition, we employ early stopping on the validation loss with a tolerance of 10 epochs to prevent over fitting.

**Evaluation Metrics:** We report micro F1 score to evaluate DA and IC performance of our models. Similarly, we use a span based F1 score, implemented in the seqeval library [5], to evaluate SL performance.

### 5.1 Results

**DA/IC/SL Results.** Table 7 presents the MFC, LSTM, and ELMO results for each domain, on the subset of 15,000 conversations annotated at both the turn and sentence levels. In general for both granularities Turn and Sentence, both LSTM,

[5] https://github.com/chakki-works/seqeval

and ELMO outperform MFC significantly across all domains. Relative to the LSTM, we find that ELMO obtains a modest increase in IC accuracy of 0.41 to 2.20 F1 points and a significant increase in SL F1 score on all domains. Concretely, ELMO boosts SL F1 performance by 3.16 to 13.17 F1 points. We see the biggest SL gains on the Insurance domain, where sentence level ELMO achieves the 13.17 point F1 gain and turn level ELMO achieves a 12.67 point F1 gain. Performance gains on the Airline domain are also large; here, ELMO increases sentence and turn level SL F1 score by 12.38 and 9.86 F1 points, respectively. Both LSTM and ELMO yield similar performance in terms of F1 score on DA classification for which the difference in performance of these models is within one F1 point across all domains. In general, the FastFood domain yields the overall lowest absolute F1 scores. Recall that Fastfood had the most diverse dialogues (biases) as per Table 4 and the lowest IAA as per Table 6.

**Sentence vs. Turn Level Annotation Units.** Regarding the performance of the LSTM and ELMO models on sentence vs. turn level annotation units, our results suggest that turn level annotations increase the difficulty of the DA classification task. This finding is evidenced by DA performance of our models on the Fast Food domain, for which F1 score is up to 25 F1 points lower for turn level annotations than sentence level annotations. We believe the increased difficulty of turn level DA relative to sentence level DA is driven by a corresponding increase in the confusability of turn level dialogue acts. This assertion of greater turn level DA confusability is supported by the lower inter annotator agreement (IAA) scores on turn level DA, which range from 0.314 to 0.521, relative to IAA scores for sentence level DA, which range from 0.598 to 0.709. This experimental result highlights the importance of collecting sentence level annotations for conversational DA datasets. Somewhat surprisingly, our models achieve similar IC F1 and SL F1 scores on turn and sentence level annotations. We hypothesize that the choice of annotation unit has a lesser impact on the IC and SL tasks because customer utterances are more likely to focus on a single speech act, whereas Agent utterances may be more complex in comparison and include a greater number of speech acts.

**Joint Training on Agent DA.** Agent DA classification naturally lends itself to joint training, given agent DAs are shared among all domains. To explore the benefits of multi-domain training, we jointly train an agent DA classification model on all domains and report test results for each domain separately. These results are provided in Table 8. This straightforward technique leads to a consistent but less than one point improvement in F1 scores. We expect that more sophisticated transfer learning methods (Liu et al., 2017; Howard and Ruder, 2018) could generate larger improvements for these domains.

Overall, our results demonstrate that there is still headroom for performance improvement, especially for the SL task, across all domains. Consequently, `MultiDoGO` should be a relevant benchmark for developing new state-of-the-art NLU models for the foreseeable future.

## 6 Future Directions

The data collection and annotation methodology that we use to gather `MultiDoGO` can efficiently scale across languages. Several pilot experiments aimed at collecting Spanish dialogues in the same domains have shown preliminary success in quality assessment. The production of a NLU dataset with parallel data in multiple languages would be a boon to the cross-lingual research community. To date, cross-lingual NLU research (Upadhyay et al., 2018; Schuster et al., 2018) has relied on much smaller parallel corpora.

## 7 Conclusion

We present `MultiDoGO`, a new Wizard-of-Oz dialogue dataset that is the largest human-generated, multi-domain corpora of conversations to date. The scale and range of this data provides a testbed for future work in joint training and transfer learning. Moreover, our comparison of sentence and turn level annotations provides insight into the effect of annotation granularity on downstream model performance. By pairing crowd-source labor with professional data annotators, we balance the cost, diversity, and quality of these conversations in a scalable manner. We show that by adopting a modular annotation strategy, the crowds can reliably annotate dialogues at a level commensurate with trained professional annotators.

## References


Layla El Asri, Hannes Schulz, Shikhar Sharma, Jeremie Zumer, Justin Harris, Emery Fine, Rahul Mehrotra, and Kaheer Suleman. 2017. Frames: A corpus for adding memory to goal-oriented dialogue systems. *arXiv preprint arXiv:1704.00057*.

Antoine Bordes, Y-Lan Boureau, and Jason Weston. 2016. Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683*.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. Multiwoz-a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. *arXiv preprint arXiv:1810.00278*.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wentau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. Quac: Question answering in context. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1723–1732.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. Allennlp: A deep semantic natural language processing platform.

Charles T Hemphill, John J Godfrey, and George R Doddington. 1990. The atis spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339.

John F Kelley. 1984. An iterative design methodology for user-friendly natural language office information applications. *ACM Transactions on Information Systems (TOIS)*, 2(1):26–41.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Oliver Lemon, Kallirroi Georgila, James Henderson, and Matthew Stuttle. 2006. An isu dialogue system exhibiting reinforcement learning of dialogue policies: generic slot-filling in the talk in-car system. In *Proceedings of the Eleventh Conference of the European Chapter of the Association for Computational Linguistics: Posters & Demonstrations*, pages 119–122. Association for Computational Linguistics.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*.

Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1–10.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. *arXiv preprint arXiv:1506.08909*.

Eric Mihail, Krishnan Lakshmi, Charette Francois, and Manning Christopher. 2017. Key-value retrieval networks for task-oriented dialogue. In *Proceedings of the Special Interest Group on Discourse and Dialogue*.

Silvia Pareti and Tatiana Lando. 2018. Dialog intent structure: A hierarchical schema of linked dialog acts. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*, Miyazaki, Japan. European Languages Resources Association (ELRA).

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Siva Reddy, Danqi Chen, and Christopher D Manning. 2019. Coqa: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*.

Sebastian Schuster, Sonal Gupta, Rushin Shah, and Mike Lewis. 2018. Cross-lingual transfer learning for multilingual task oriented dialog. *arXiv preprint arXiv:1810.13327*.

Shyam Upadhyay, Manaal Faruqui, Gokhan Tür, Hakkani-Tür Dilek, and Larry Heck. 2018. (almost) zero-shot cross-lingual spoken language understanding. In *2018 IEEE International Conference on*

*Acoustics, Speech and Signal Processing (ICASSP)*, pages 6034–6038. IEEE.

Zhuoran Wang and Oliver Lemon. 2013. A simple and generic belief tracking mechanism for the dialog state tracking challenge: On the believability of observed information. In *Proceedings of the SIGDIAL 2013 Conference*, pages 423–432.

Wei Wei, Quoc Le, Andrew Dai, and Jia Li. 2018. Airdialogue: An environment for goal-oriented dialogue research. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3844–3854.

Joseph Weizenbaum. 1966. Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45.

Jason Williams, Antoine Raux, and Matthew Henderson. 2016. The dialog state tracking challenge series: A review. *Dialogue & Discourse*, 7(3):4–33.

# Build it Break it Fix it for Dialogue Safety: Robustness from Adversarial Human Attack

**Emily Dinan**
Facebook AI Research
edinan@fb.com

**Samuel Humeau**
Facebook AI Research
samuel.humeau@gmail.com

**Bharath Chintagunta**
Virginia Tech
jaic4@vt.edu

**Jason Weston**
Facebook AI Research
jase@fb.com

## Abstract

The detection of offensive language in the context of a dialogue has become an increasingly important application of natural language processing. The detection of trolls in public forums (Galán-García et al., 2016), and the deployment of chatbots in the public domain (Wolf et al., 2017) are two examples that show the necessity of guarding against adversarially offensive behavior on the part of humans. In this work, we develop a training scheme for a model to become robust to such human attacks by an iterative build it, break it, fix it strategy with humans and models in the loop. In detailed experiments we show this approach is considerably more robust than previous systems. Further, we show that offensive language used within a conversation critically depends on the dialogue context, and cannot be viewed as a single sentence offensive detection task as in most previous work. Our newly collected tasks and methods are all made open source and publicly available.

## 1 Introduction

The detection of offensive language has become an important topic as the online community has grown, as so too have the number of bad actors (Cheng et al., 2017). Such behavior includes, but is not limited to, trolling in public discussion forums (Herring et al., 2002) and via social media (Silva et al., 2016; Davidson et al., 2017), employing hate speech that expresses prejudice against a particular group, or offensive language specifically targeting an individual. Such actions can be motivated to cause harm from which the bad actor derives enjoyment, despite negative consequences to others (Bishop, 2014). As such, some bad actors go to great lengths to both avoid detection and to achieve their goals (Shachaf and Hara, 2010). In that context, any attempt to automatically detect this behavior can be expected to be adversarially attacked by looking for weaknesses in the detection system, which currently can easily be exploited as shown in (Hosseini et al., 2017; Gröndahl et al., 2018). A further example, relevant to the natural langauge processing community, is the exploitation of weaknesses in machine learning models that *generate* text, to force them to emit offensive language. Adversarial attacks on the Tay chatbot led to the developers shutting down the system (Wolf et al., 2017).

In this work, we study the detection of offensive language in dialogue with models that are robust to adversarial attack. We develop an automatic approach to the "Build it Break it Fix it" strategy originally adopted for writing secure programs (Ruef et al., 2016), and the "Build it Break it" approach consequently adapting it for NLP (Ettinger et al., 2017). In the latter work, two teams of researchers, "builders" and "breakers" were used to first create sentiment and semantic role-labeling systems and then construct examples that find their faults. In this work we instead fully automate such an approach using crowdworkers as the humans-in-the-loop, and also apply a fixing stage where models are retrained to improve them. Finally, we repeat the whole build, break, and fix sequence over a number of iterations.

We show that such an approach provides more and more robust systems over the fixing iterations. Analysis of the type of data collected in the iterations of the break it phase shows clear distribution changes, moving away from simple use of profanity and other obvious offensive words to utterances that require understanding of world knowledge, figurative language, and use of negation to detect if they are offensive or not. Further, data collected in the context of a dialogue rather than a sentence without context provides more sophisticated attacks. We show that model architectures that use the dialogue context efficiently perform much bet-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4537–4546,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

ter than systems that do not, where the latter has been the main focus of existing research (Wulczyn et al., 2017; Davidson et al., 2017; Zampieri et al., 2019).

Code for our entire build it, break it, fix it algorithm is made open source, complete with model training code and crowdsourcing interface for humans. Our data and trained models will also be made available for the community via ParlAI[1].

## 2 Related Work

The task of detecting offensive language has been studied across a variety of content classes. Perhaps the most commonly studied class is hate speech, but work has also covered bullying, aggression, and toxic comments (Zampieri et al., 2019).

To this end, various datasets have been created to benchmark progress in the field. In hate speech detection, recently Davidson et al. (2017) compiled and released a dataset of over 24,000 tweets labeled as containing hate speech, offensive language, or neither. In order to benchmark toxic comment detection, The Wikipedia Toxic Comments dataset (which we study in this work) was collected and extracted from Wikipedia Talk pages and featured in a Kaggle competition (Wulczyn et al., 2017; Google, 2018). Each of these benchmarks examine only single-turn utterances, outside of the context in which the language appeared. In this work we recommend that future systems should move beyond classification of singular utterances and use contextual information to help identify offensive language.

Many approaches have been taken to solve these tasks – from linear regression and SVMs to deep learning (Noever, 2018). The best performing systems in each of the competitions mentioned above (for aggression and toxic comment classification) used deep learning approaches such as LSTMs and CNNs (Kumar et al., 2018; Google, 2018). In this work we consider a large-pretrained transformer model which has been shown to perform well on many downstream NLP tasks (Devlin et al., 2018).

The broad class of adversarial training is currently a hot topic in machine learning (Goodfellow et al., 2014). Use cases include training image generators (Brock et al., 2018) as well as image classifiers to be robust to adversarial examples (Liu et al., 2019). These methods find the breaking examples algorithmically, rather than by using humans breakers as we do. Applying the same approaches to NLP tends to be more challenging because, unlike for images, even small changes to a sentence can cause a large change in the meaning of that sentence, which a human can detect but a lower quality model cannot. Nevertheless algorithmic approaches have been attempted, for example in text classification (Ebrahimi et al., 2018), machine translation (Belinkov and Bisk, 2018), dialogue generation tasks (Li et al., 2017) and reading comprehension (Jia and Liang, 2017). The latter was particularly effective at proposing a more difficult version of the popular SQuAD dataset.

As mentioned in the introduction, our approach takes inspiration from "Build it Break it" approaches which have been successfully tried in other domains (Ruef et al., 2016; Ettinger et al., 2017). Those approaches advocate finding faults in systems by having humans look for insecurities (in software) or prediction failures (in models), but do not advocate an automated approach as we do here. Our work is also closely connected to the "Mechanical Turker Descent" algorithm detailed in (Yang et al., 2018) where language to action pairs were collected from crowdworkers by incentivizing them with a game-with-a-purpose technique: a crowdworker receives a bonus if their contribution results in better models than another crowdworker. We did not gamify our approach in this way, but still our approach has commonalities in the round-based improvement of models through crowdworker interaction.

## 3 Baselines: Wikipedia Toxic Comments

In this section we describe the publicly available data that we have used to bootstrap our *build it break it fix it* approach. We also compare our model choices with existing work and clarify the metrics chosen to report our results.

**Wikipedia Toxic Comments** The Wikipedia Toxic Comments dataset (WTC) has been collected in a common effort from the Wikimedia Foundation and Jigsaw (Wulczyn et al., 2017) to identify personal attacks online. The data has been extracted from the Wikipedia Talk pages, discussion pages where editors can discuss improvements to articles or other Wikipedia pages. We considered the version of the dataset that corresponds to the Kaggle competition: "Toxic Comment Classification Challenge" (Google, 2018)

[1] https://parl.ai/projects/dialogue_safety/

which features 7 classes of toxicity: toxic, severe toxic, obscene, threat, insult, identity hate and non-toxic. In the same way as in (Khatri et al., 2018), every label except non-toxic is grouped into a class OFFENSIVE while the non-toxic class is kept as the SAFE class. In order to compare our results to (Khatri et al., 2018), we similarly split this dataset to dedicate 10% as a test set. 80% are dedicated to train set while the remaining 10% is used for validation. Statistics on the dataset are shown in Table 1.

**Models** We establish baselines using two models. The first one is a binary classifier built on top of a large pre-trained transformer model. We use the same architecture as in BERT (Devlin et al., 2018). We add a linear layer to the output of the first token ([CLS]) to produce a final binary classification. We initialize the model using the weights provided by (Devlin et al., 2018) corresponding to "BERT-base". The transformer is composed of 12 layers with hidden size of 768 and 12 attention heads. We fine-tune the whole network on the classification task. We also compare it the fastText classifier (Joulin et al., 2017) for which a given sentence is encoded as the average of individual word vectors that are pre-trained on a large corpus issued from Wikipedia. A linear layer is then applied on top to yield a binary classification.

**Experiments** We compare the two aforementioned models with (Khatri et al., 2018) who conducted their experiments with a BiLSTM with GloVe pre-trained word vectors (Pennington et al., 2014). Results are listed in Table 2 and we compare them using the weighted-F1, i.e. the sum of F1 score of each class weighted by their frequency in the dataset. We also report the F1 of the OFFENSIVE-class which is the metric we favor within this work, although we report both. (Note that throughout the paper, the notation F1 is always referring to OFFENSIVE-class F1.) Indeed, in the case of an imbalanced dataset such as Wikipedia Toxic Comments where most samples are SAFE, the weighted-F1 is closer to the F1 score of the SAFE class while we focus on detecting OFFENSIVE content. Our BERT-based model outperforms the method from Khatri et al. (2018); throughout the rest of the paper, we use the BERT-based architecture in our experiments. In particular, we used this baseline trained on WTC to bootstrap our approach, to be described subsequently.

| | Train | Valid | Test |
|---|---|---|---|
| SAFE | 89.8% | 89.7% | 90.1% |
| OFFENSIVE | 10.2% | 10.3% | 9.1% |
| Total | 114656 | 15958 | 15957 |

Table 1: Dataset statistics for our splits of Wikipedia Toxic Comments.

| | OFFENSIVE F1 | Weighted F1 |
|---|---|---|
| fastText | 71.4% | 94.8% |
| BERT-based | 83.4% | 96.7% |
| (Khatri et al., 2018) | - | 95.4% |

Table 2: Comparison between our models based on fastText and BERT with the BiLSTM used by (Khatri et al., 2018) on Wikipedia Toxic Comments.

## 4 Build it Break it Fix it Method

In order to train models that are robust to adversarial behavior, we posit that it is crucial collect and train on data that was collected in an adversarial manner. We propose the following automated build it, break it, fix it algorithm:

1. **Build it:** Build a model capable of detecting OFFENSIVE messages. This is our best-performing BERT-based model trained on the Wikipedia Toxic Comments dataset described in the previous section. We refer to this model throughout as $A_0$.

2. **Break it:** Ask crowdworkers to try to "beat the system" by submitting messages that our system ($A_0$) marks as SAFE but that the worker considers to be OFFENSIVE.

3. **Fix it:** Train a new model on these collected examples in order to be more robust to these adversarial attacks.

4. **Repeat:** Repeat, deploying the newly trained model in the **break it** phase, then **fix it** again.

See Figure 1 for a visualization of this process.

### 4.1 Break it Details

**Definition of** OFFENSIVE Throughout data collection, we characterize OFFENSIVE messages for users as messages that would not be "ok to send in a friendly conversation with someone you just met online." We use this specific language in an attempt to capture various classes of content that would be considered unacceptable in a friendly

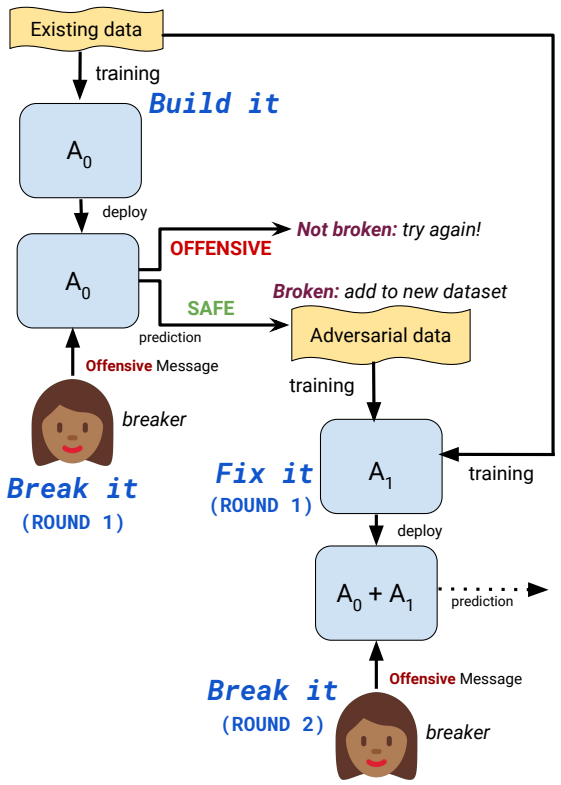


Figure 1: The build it, break it, fix it algorithm we use to iteratively train better models $A_0, \ldots, A_N$. In experiments we perform $N = 3$ iterations of the break it, fix it loop for the single-turn utterance detection task, and a further iteration for the multi-turn task in a dialogue context setting.

conversation, without imposing our own definitions of what that means. The phrase "with someone you just met online" was meant to mimic the setting of a public forum.

**Crowderworker Task** We ask crowdworkers to try to "beat the system" by submitting messages that our system marks as SAFE but that the worker considers to be OFFENSIVE. For a given round, workers earn a "game" point each time they are able to "beat the system," or in other words, trick the model by submitting OFFENSIVE messages that the model marks as SAFE. Workers earn up to 5 points each round, and have two tries for each point: we allow multiple attempts per point so that workers can get feedback from the models and better understand their weaknesses. The points serve to indicate success to the crowdworker and motivate to achieve high scores, but have no other meaning (e.g. no monetary value as in (Yang et al., 2018)). More details regarding the user interface and instructions can be found in Appendix B.

**Models to Break** During round 1, workers try to break the baseline model $A_0$, trained on Wikipedia Toxic Comments. For rounds $i$, $i > 1$, workers must break both the baseline model and the model from the previous "fix it" round, which we refer to as $A_{i-1}$. In that case, the worker must submit messages that both $A_0$ and $A_{i-1}$ mark as SAFE but which the worker considers to be OFFENSIVE.

### 4.2 Fix it Details

During the "fix it" round, we update the models with the newly collected adversarial data from the "break it" round. The training data consists of all previous rounds of data, so that model $A_i$ is trained on all rounds $n$ for $n \leq i$, as well as the Wikipedia Toxic Comments data. We split each round of data into train, validation, and test partitions. The validation set is used for hyperparameter selection. The test sets are used to measure how robust we are to new adversarial attacks. With increasing round $i$, $A_i$ should become more robust to increasingly complex human adversarial attacks.

## 5 Single-Turn Task

We first consider a single-turn set-up, i.e. detection of offensive language in one utterance, with no dialogue context or conversational history.

### 5.1 Data Collection

**Adversarial Collection** We collected three rounds of data with the build it, break it, fix it algorithm described in the previous section. Each round of data consisted of 1000 examples, leading to 3000 single-turn adversarial examples in total. For the remainder of the paper, we refer to this method of data collection as the *adversarial method*.

**Standard Collection** In addition to the *adversarial method*, we also collected data in a non-adversarial manner in order to directly compare the two set-ups. In this method – which we refer to as the *standard method*, we simply ask crowdworkers to submit messages that they consider to be OFFENSIVE. There is no model to break. Instructions are otherwise the same.

In this set-up, there is no real notion of "rounds", but for the sake of comparison we refer to each subsequent 1000 examples collected in this manner as a "round". We collect 3000 examples – or three rounds of data. We refer to a model trained on rounds $n \leq i$ of the *standard* data as $S_i$.

**Single-Turn Adversarial (Round 1) and Standard Task OFFENSIVE Examples**

| | contains profanity | non-profane offending words | contains negation | contains figurative language | requires world knowledge | contains sarcasm |
|---|---|---|---|---|---|---|
| Standard | 13% | 12% | 12% | 11% | 8% | 3% |
| Adversarial | 0% | 5% | 23% | 19% | 14% | 6% |

Table 3: Language analysis of the single-turn *standard* and *adversarial* (round 1) tasks by human annotation of various language properties. Standard collection examples contain more words found in an offensive words list, while *adversarial* examples require more sophisticated language understanding.

#### 5.1.1 Task Formulation Details

Since all of the collected examples are labeled as OFFENSIVE, to make this task a binary classification problem, we will also add SAFE examples to it. The "safe data" is comprised of utterances from the ConvAI2 chit-chat task (Dinan et al., 2019; Zhang et al., 2018) which consists of pairs of humans getting to know each other by discussing their interests. Each utterance we used was reviewed by two independent crowdworkers and labeled as SAFE, with the same characterization of SAFE as described before. For each partition (train, validation, test), the final task has a ratio of 9:1 SAFE to OFFENSIVE examples, mimicking the division of the Wikipedia Toxic Comments dataset used for training our baseline models. Dataset statistics for the final task can be found in Table 5. We refer to these tasks – with both SAFE and OFFENSIVE examples – as the *adversarial* and *standard* tasks.

#### 5.1.2 Model Training Details

Using the BERT-based model architecture described in Section 3, we trained models on each round of the *standard* and *adversarial* tasks, multi-tasking with the Wikipedia Toxic Comments task. We weight the multi-tasking with a mixing parameter which is also tuned on the validation set. Finally, after training weights with the cross entropy loss, we adjust the final bias also using the validation set. We optimize for the sensitive class (i.e. OFFENSIVE-class) F1 metric on the *standard* and *adversarial* validation sets respectively.

For each task (*standard* and *adversarial*), on round $i$, we train on data from all rounds $n$ for $n \leq i$ and optimize for performance on the validation sets $n \leq i$.

### 5.2 Experimental Results

We conduct experiments comparing the *adversarial* and *standard* methods. We break down the results into "break it" results comparing the data collected and "fix it" results comparing the models obtained.

| | % with profanity | % with "not" | avg. # chars | avg. # tokens |
|---|---|---|---|---|
| **Std. (Rnds 1-3)** | 18.2 | 2.8 | 48.6 | 9.4 |
| **Adv. Rnd 1** | 2.6 | 5.8 | 53.7 | 10.3 |
| **Adv. Rnd 2** | 1.5 | 5.5 | 44.5 | 9 |
| **Adv. Rnd 3** | 1.2 | 9.8 | 45.7 | 9.3 |
| **Multi-turn Adv.** | 1.6 | 4.9 | 36.6 | 7.8 |

Table 4: Percent of OFFENSIVE examples in each task containing profanity, the token "not", as well as the average number of characters and tokens in each example. Rows 1-4 are the single-turn task, and the last row is the multi-turn task. Later rounds have less profanity and more use of negation as human breakers have to find more sophisticated language to adversarially attack our models.

| Rounds {1, 2 and 3} | Train | Valid | Test |
|---|---|---|---|
| SAFE **Examples** | 21,600 | 2700 | 2700 |
| OFFENSIVE **Examples** | 2400 | 300 | 300 |
| **Total Examples** | 24,000 | 3,000 | 3,000 |

Table 5: Dataset statistics for the single-turn rounds of the *adversarial* task data collection. There are three rounds in total all of identical size, hence the numbers above can be divided for individual statistics. The *standard* task is an additional dataset of exactly the same size as above.

#### 5.2.1 Break it Phase

Examples obtained from both the *adversarial* and *standard* collection methods were found to be clearly offensive, but we note several differences in the distribution of examples from each task, shown in Table 4. First, examples from the *standard* task tend to contain more profanity. Using a list of common English obscenities and otherwise bad words[2], in Table 4 we calculate the percentage of examples in each task containing such obscenities, and see that the *standard* examples contain

[2]https://github.com/LDNOOBW/List-of-Dirty-Naughty-Obscene-and-Otherwise-Bad-Words

| | | WTC Baseline | *Standard* models | | | *Adversarial* models | | |
|---|---|---|---|---|---|---|---|---|
| Task Type | Task Round | $A_0$ | $S_1$ | $S_2$ | $S_3$ | $A_1$ | $A_2$ | $A_3$ |
| WTC | - | **83.3** | 80.6 | 81.1 | 82.1 | 81.3 | 78.9 | 78.0 |
| *Standard Task* | All (1-3) | 68.1 | 83.3 | 85.8 | **88.0** | 83.0 | 85.3 | 83.7 |
| *Adversarial Task* | 1 | 0.0 | 51.7 | 69.3 | 68.6 | 71.8 | **79.0** | 78.2 |
| | 2 | 0.0 | 10.8 | 26.4 | 31.8 | 0.0 | **64.4** | 62.1 |
| | 3 | 0.0 | 12.3 | 17.1 | 13.7 | 32.1 | 0.0 | **59.9** |
| | All (1-3) | 0.0 | 27.4 | 41.7 | 41.8 | 40.6 | 55.5 | **67.6** |

Table 6: Test performance of best standard models trained on *standard* task rounds (models $S_i$ for each round $i$) and best *adversarial* models trained on adversarial task rounds (models $A_i$). All models are evaluated using OFFENSIVE-class F1 on each round of both the *standard* task and *adversarial* task. $A_0$ is the baseline model trained on the existing Wiki Toxic Comments (WTC) dataset. Adversarial models prove to be more robust than standard ones against attack (Adversarial Task 1-3), while still performing reasonably on Standard and WTC tasks.

at least seven times as many as each round of the *adversarial* task. Additionally, in previous works, authors have observed that classifiers struggle with negations (Hosseini et al., 2017). This is borne out by our data: examples from the single-turn *adversarial* task more often contain the token "not" than examples from the *standard* task, indicating that users are easily able to fool the classifier with negations.

We also anecdotally see figurative language such as "snakes hiding in the grass" in the adversarial data, which contain no individually offensive words, the offensive nature is captured by reading the entire sentence. Other examples require sophisticated world knowledge such as that many cultures consider eating cats to be offensive. To quantify these differences, we performed a blind human annotation of a sample of the data, 100 examples of standard and 100 examples of adversarial round 1. Results are shown in Table 3. Adversarial data was indeed found to contain less profanity, fewer non-profane but offending words (such as "idiot"), more figurative language, and to require more world knowledge.

We note that, as anticipated, the task becomes more challenging for the crowdworkers with each round, indicated by the decreasing average scores in Table 7. In round 1, workers are able to get past $A_0$ most of the time – earning an average score of 4.56 out of 5 points per round – showcasing how susceptible this baseline is to adversarial attack despite its relatively strong performance on the Wikipedia Toxic Comments task. By round 3, however, workers struggle to trick the system, earning an average score of only 1.6 out of 5. A finer-grained assessment of the worker scores can be found in Table 11 in the appendix.

| | Single-Turn | | | Multi |
|---|---|---|---|---|
| Round | 1 | 2 | 3 | ("4") |
| Avg. score (0-5) | 4.56 | 2.56 | 1.6 | 2.89 |

Table 7: Adversarial data collection worker scores. Workers received a score out of 5 indicating how often (out of 5 rounds) they were able to get past our classifiers within two tries. In later single-turn rounds it is harder to defeat our models, but switching to multi-turn makes this easier again as new attacks can be found by using the dialogue context.

### 5.2.2 Fix it Phase

Results comparing the performance of models trained on the *adversarial* ($A_i$) and *standard* ($S_i$) tasks are summarized in Table 6, with further results in Table 13 in Appendix A.2. The adversarially trained models $A_i$ prove to be more robust to adversarial attack: on each round of adversarial testing they outperform standard models $S_i$.

Further, note that the *adversarial* task becomes harder with each subsequent round. In particular, the performance of the standard models $S_i$ rapidly deteriorates between round 1 and round 2 of the *adversarial* task. This is a clear indication that models need to train on adversarially-collected data to be robust to adversarial behavior.

*Standard* models ($S_i$), trained on the *standard* data, tend to perform similarly to the *adversarial* models ($A_i$) as measured on the *standard* test sets, with the exception of training round 3, in which $A_3$ fails to improve on this task, likely due to being too optimized for adversarial tasks. The *standard* models $S_i$, on the other hand, are improving

with subsequent rounds as they have more training data of the same distribution as the evaluation set. Similarly, our baseline model performs best on its own test set, but other models are not far behind.

Finally, we remark that all scores of 0 in Table 6 are by design, as for round $i$ of the *adversarial* task, both $A_0$ and $A_{i-1}$ classified each example as SAFE during the 'break it' data collection phase.

## 6 Multi-Turn Task

In most real-world applications, we find that adversarial behavior occurs in context – whether it is in the context of a one-on-one conversation, a comment thread, or even an image. In this work we focus on offensive utterances within the context of two-person dialogues. For dialogue safety we posit it is important to move beyond classifying single utterances, as it may be the case that an utterance is entirely innocuous on its own but extremely offensive in the context of the previous dialogue history. For instance, "Yes, you should definitely do it!" is a rather inoffensive message by itself, but most would agree that it is a hurtful response to the question "Should I hurt myself?"

### 6.1 Task Implementation

To this end, we collect data by asking crowdworkers to try to "beat" our best single-turn classifier (using the model that performed best on rounds 1-3 of the *adversarial* task, i.e., $A_3$), in addition to our baseline classifier $A_0$. The workers are shown truncated pieces of a conversation from the ConvAI2 chit-chat task, and asked to continue the conversation with OFFENSIVE responses that our classifier marks as SAFE. The resulting conversations – including the newly provided OFFENSIVE responses – are between 3 and 6 turns long. As before, workers have two attempts per conversation to try to get past the classifier and are shown five conversations per round. They are given a score (out of five) at the end of each round indicating the number of times they successfully fooled the classifier.

We collected 3000 offensive examples in this manner. As in the single-turn set up, we combine this data with SAFE examples with a ratio of 9:1 SAFE to OFFENSIVE for classifier training. The safe examples are dialogue examples from ConvAI2 for which the responses were reviewed by two independent crowdworkers and labeled as SAFE, as in the s single-turn task set-up. We refer to this overall task as the *multi-turn adversarial* task. Dataset statistics are given in Table 9.

### 6.2 Models

To measure the impact of the context, we train models on this dataset with and without the given context. We use the fastText and the BERT-based model described in Section 3. In addition, we build a BERT-based model variant that splits the last utterance (to be classified) and the rest of the history into two dialogue segments. Each segment is assigned an embedding and the input provided to the transformer is the sum of word embedding and segment embedding, replicating the setup of the Next Sentence Prediction that is used in the training of BERT (Devlin et al., 2018).

### 6.3 Experimental Results

#### 6.3.1 Break it Phase

During data collection, we observed that workers had an easier time bypassing the classifiers than in the single-turn set-up. See Table 7. In the single-turn set-up, the task at hand gets harder with each round – the average score of the crowdworkers decreases from 4.56 in round 1 to 1.6 in round 3. Despite the fact that we are using our best single-turn classifier in the multi-turn set-up ($A_3$), the task becomes easier: the average score per round is 2.89. This is because the workers are often able to use contextual information to suggest something offensive rather than say something offensive outright. See examples of submitted messages in Table 8. Having context also allows one to express something offensive more efficiently: the messages supplied by workers in the multi-turn setting were significantly shorter on average, see Table 4.

#### 6.3.2 Fix it Phase

During training, we multi-tasked the *multi-turn adversarial* task with the Wikipedia Toxic Comments task as well as the single-turn *adversarial* and *standard* tasks. We average the results of our best models from five different training runs. The results of these experiments are given in Table 10.

As we observed during the training of our baselines in Section 3, the fastText model architecture is ill-equipped for this task relative to our BERT-based architectures. The fastText model performs worse given the dialogue context (an average of 23.56 OFFENSIVE-class F1 relative to 37.1) than without, likely because its bag-of-

| Multi-Turn Adversarial Task Examples | | |
|---|---|---|
| **Context** | *[P1:]* | greetings and salutations my friend |
| | *[P2:]* | hi how are you doing? |
| | *[P1:]* | i'm well. enjoying some r r from the navy station |
| | *[P2:]* | i love to cook. do you cook? |
| | *[P1:]* | i do on occasion. i love to read more though. |
| OFFENSIVE **Response** | *[P2:]* | **reading the menu doesnt count** |
| **Context** | *[P1:]* | hey how are you |
| | *[P2:]* | good but i keep thinking about the time i almost played for nba |
| | *[P1:]* | aww why did not you play |
| | *[P2:]* | lost my nerve i guess. what do you do? |
| | *[P1:]* | oh that's to bad. i work at a gas station |
| OFFENSIVE **Response** | *[P2:]* | **i need to follow my dream to not work in a gas station** |

Table 8: Examples from the *multi-turn adversarial* task. Responses can be offensive only in context.

| **Multi-Turn Adversarial Task** | Train | Valid | Test |
|---|---|---|---|
| SAFE **Examples** | 21,600 | 2,700 | 2,700 |
| OFFENSIVE **Examples** | 2,400 | 300 | 300 |
| **Total Examples** | 24,000 | 3,000 | 3,000 |

Table 9: *Multi-turn adversarial* task data statistics.

| **Multi-Turn Adversarial Task Results** | | |
|---|---|---|
| | F1 | Weighted-F1 |
| **fastText** | | |
| with context | $23.6 \pm 1.9$ | $85.9 \pm 0.5$ |
| without context | $37.1 \pm 2.6$ | $88.8 \pm 0.6$ |
| **BERT-based** *(no segments)* | | |
| with context | $60.5 \pm 1.3$ | $92.2 \pm 0.3$ |
| without context | $56.8 \pm 1.6$ | $90.6 \pm 0.7$ |
| **BERT-based** *(dialogue segments)* | | |
| with context | $66.4 \pm 2.2$ | $93.2 \pm 0.4$ |
| without context | $59.0 \pm 2.5$ | $91.2 \pm 0.8$ |

Table 10: Results of experiments on the multi-turn *adversarial* task. We denote the average and one standard deviation from the results of five runs. Models that use the context as input ("with context") perform better. Encoding this in the architecture as well (via BERT *dialogue segment* features) gives us the best results.

embeddings representation is too simple to take the context into account.

We see the opposite with our BERT-based models, indicating that more complex models are able to effectively use the contextual information to detect whether the response is SAFE or OFFENSIVE. With the simple BERT-based architecture (that does not split the context and the utterance into separate segments), we observe an average of a 3.7 point increase in OFFENSIVE-class F1 with the addition of context. When we use segments to separate the context from the utterance we are trying to classify, we observe an average of a 7.4 point increase in OFFENSIVE-class F1. Thus, it appears that the use of contextual information to identify OFFENSIVE language is critical to making these systems robust, and improving the model architecture to take account of this has large impact.

## 7 Conclusion

We have presented an approach to build more robust offensive language detection systems in the context of a dialogue. We proposed a build it, break it, fix it, and then repeat strategy, whereby humans attempt to break the models we built, and we use the broken examples to fix the models. We show this results in far more nuanced language than in existing datasets. The adversarial data includes less profanity, which existing classifiers can pick up on, and is instead offensive due to figurative language, negation, and by requiring more world knowledge, which all make current classifiers fail. Similarly, offensive language in the context of a *dialogue* is also more nuanced than standalone offensive utterances. We show that classifiers that learn from these more complex examples are indeed more robust to attack, and that using the dialogue context gives improved performance if the model architecture takes it into account.

In this work we considered a binary problem (offensive or safe). Future work could consider classes of offensive language separately (Zampieri et al., 2019), or explore other dialogue tasks, e.g. from social media or forums. Another interesting direction is to explore how our build it, break it, fix it strategy would similarly apply to make neural generative models safe (Henderson et al., 2018).

## Acknowledgments

Thank you to Stephen Roller for providing insights and ideas for this project. The emoji image in Figure 1 is by Twemoji[3], and is licensed under CC BY-4.0.

## References

2018. *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and natural noise both break neural machine translation. In (DBL, 2018).

Jonathan Bishop. 2014. Representations of trolls in mass media communication: a review of media-texts and moral panics relating to internet trolling. *International Journal of Web Based Communities*, 10(1):7–24.

Andrew Brock, Jeff Donahue, and Karen Simonyan. 2018. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*.

Justin Cheng, Cristian Danescu-Niculescu-Mizil, Jure Leskovec, and Michael Bernstein. 2017. Anyone can become a troll: Causes of trolling behavior in online discussions. *American Scientist*, 105(3):152.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Eleventh International AAAI Conference on Web and Social Media*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Emily Dinan, Varvara Logacheva, Valentin Malykh, Alexander Miller, Kurt Shuster, Jack Urbanek, Douwe Kiela, Arthur Szlam, Iulian Serban, Ryan Lowe, et al. 2019. The second conversational intelligence challenge (convai2). *arXiv preprint arXiv:1902.00098*.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. Hotflip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 31–36. Association for Computational Linguistics.

[3] https://github.com/twitter/twemoji

Allyson Ettinger, Sudha Rao, Hal Daumé III, and Emily M Bender. 2017. Towards linguistically generalizable nlp systems: A workshop and shared task. *arXiv preprint arXiv:1711.01505*.

Patxi Galán-García, José Gaviria de la Puerta, Carlos Laorden Gómez, Igor Santos, and Pablo García Bringas. 2016. Supervised machine learning for the detection of troll profiles in twitter social network: Application to a real case of cyberbullying. *Logic Journal of the IGPL*, 24(1):42–53.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Google. 2018. Toxic comment classification challenge.

Tommi Gröndahl, Luca Pajola, Mika Juuti, Mauro Conti, and N Asokan. 2018. All you need is" love": Evading hate-speech detection. *arXiv preprint arXiv:1808.09115*.

Peter Henderson, Koustuv Sinha, Nicolas Angelard-Gontier, Nan Rosemary Ke, Genevieve Fried, Ryan Lowe, and Joelle Pineau. 2018. Ethical challenges in data-driven dialogue systems. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '18, pages 123–129, New York, NY, USA. ACM.

Susan Herring, Kirk Job-Sluder, Rebecca Scheckler, and Sasha Barab. 2002. Searching for safety online: Managing" trolling" in a feminist forum. *The information society*, 18(5):371–384.

Hossein Hosseini, Sreeram Kannan, Baosen Zhang, and Radha Poovendran. 2017. Deceiving google's perspective api built for detecting toxic comments. *arXiv preprint arXiv:1702.08138*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In (Palmer et al., 2017), pages 2021–2031.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics.

Chandra Khatri, Behnam Hedayatnia, Rahul Goel, Anushree Venkatesh, Raefer Gabriel, and Arindam Mandal. 2018. Detecting offensive content in open-domain conversations using two stage semi-supervision. *CoRR*, abs/1811.12900.

Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking aggression identification in social media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 1–11, Santa Fe, New

Mexico, USA. Association for Computational Linguistics.

Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. In (Palmer et al., 2017), pages 2157–2169.

Aishan Liu, Xianglong Liu, Jiaxin Fan, Yuqing Ma, Anlan Zhang, Huiyuan Xie, and Dacheng Tao. 2019. Perceptual-sensitive gan for generating adversarial patches.

David Noever. 2018. Machine learning suites for online toxicity detection. *arXiv preprint arXiv:1810.01869*.

Martha Palmer, Rebecca Hwa, and Sebastian Riedel, editors. 2017. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Andrew Ruef, Michael Hicks, James Parker, Dave Levin, Michelle L Mazurek, and Piotr Mardziel. 2016. Build it, break it, fix it: Contesting secure development. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 690–703. ACM.

Pnina Shachaf and Noriko Hara. 2010. Beyond vandalism: Wikipedia trolls. *Journal of Information Science*, 36(3):357–370.

Leandro Silva, Mainack Mondal, Denzil Correa, Fabrício Benevenuto, and Ingmar Weber. 2016. Analyzing the targets of hate in online social media. In *Tenth International AAAI Conference on Web and Social Media*.

Marty J Wolf, K Miller, and Frances S Grodzinsky. 2017. Why we should have seen that coming: comments on microsoft's tay experiment, and wider implications. *ACM SIGCAS Computers and Society*, 47(3):54–64.

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex machina: Personal attacks seen at scale. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*, pages 1391–1399. ACM.

Zhilin Yang, Saizheng Zhang, Jack Urbanek, Will Feng, Alexander H. Miller, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Mastering the dungeon: Grounded language learning by mechanical turker descent. In (DBL, 2018).

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval). *arXiv preprint arXiv:1903.08983*.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 2204–2213. Association for Computational Linguistics.

# GECOR: An End-to-End Generative Ellipsis and Co-reference Resolution Model for Task-Oriented Dialogue

**Jun Quan**[‡*], **Deyi Xiong**[‡†], **Bonnie Webber**[§] and **Changjian Hu**[¶]
[‡] School of Computer Science and Technology, Soochow University, Suzhou, China
[§] University of Edinburgh, Edinburgh, UK
[¶] Lenovo Research AI Lab, Beijing, China
terryqj0107@gmail.com, dyxiong@suda.edu.cn,
bonnie.webber@ed.ac.uk, hucj1@lenovo.com

## Abstract

Ellipsis and co-reference are common and ubiquitous especially in multi-turn dialogues. In this paper, we treat the resolution of ellipsis and co-reference in dialogue as a problem of generating omitted or referred expressions from the dialogue context. We therefore propose a unified end-to-end Generative Ellipsis and CO-reference Resolution model (GECOR) in the context of dialogue. The model can generate a new pragmatically complete user utterance by alternating the generation and copy mode for each user utterance. A multi-task learning framework is further proposed to integrate the GECOR into an end-to-end task-oriented dialogue. In order to train both the GECOR and the multi-task learning framework, we manually construct a new dataset on the basis of the public dataset CamRest676 with both ellipsis and co-reference annotation. On this dataset, intrinsic evaluations on the resolution of ellipsis and co-reference show that the GECOR model significantly outperforms the sequence-to-sequence (seq2seq) baseline model in terms of EM, BLEU and $F_1$ while extrinsic evaluations on the downstream dialogue task demonstrate that our multi-task learning framework with GECOR achieves a higher success rate of task completion than TSCP, a state-of-the-art end-to-end task-oriented dialogue model (Lei et al., 2018).

## 1 Introduction

Due to the rhetorical principle of saving words and avoiding repetitions, ellipsis and co-reference occur frequently in multi-turn dialogues leaving utterances paragmatically incomplete if they are separate from context. Humans can easily understand utterances with anaphorically referenced or absent information (e.g., $Q_2$ and $Q_3$ in Table 1) based on the dialogue context while dialogue systems often fail to understand such utterances correctly, which may result in false or incoherent responses.

If user utterances can be automatically supplemented with information that is left out or substituted by anaphora according to the dialogue context as humans do (e.g., *$Q_2$: I want cheap Italian restaurants. $Q_3$: Yes, I would like the phone number please.*), dialogue models may understand user requests correctly and would not generate wrong responses caused by ellipsis and co-reference phenomena. Especially in task-oriented dialogue systems, explicitly providing such information to the models can effectively improve the success rate of task completion.

In order to achieve this goal, we propose an end-to-end generative ellipsis and co-reference resolution model (GECOR) for task-oriented dialogue in this paper. The essential idea behind GECOR is that we treat the resolution of ellipsis and co-reference in user utterances as a generation task: transforming a user utterance with ellipsis or anaphora into a new utterance where the left-out or referred expressions are automatically generated from the dialogue context. We refer to the new utterance as the complete version of the original utterance. We use an end-to-end sequence-to-sequence model with two encoders for this transformation task, where one encoder reads the user utterance and the other the dialogue context and the decoder generates the complete utterance. Since most omitted expressions or antecedents can be found in the dialogue context, we resort to the attention and copy mechanism to detect such fragments in previous context and copy them into the generated complete utterance.

We then incorporate GECOR into an end-to-end task-oriented dialogue system in a multi-task

[*] Work performed during an internship at Lenovo Research AI Lab.
[†] Corresponding author



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4547–4557,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

learning framework. The entire model contains two encoders (one for user utterance and the other for the dialogue context) and three decoders: one decoder for predicting dialogue states, the second decoder for generating complete user utterances and the third decoder for generating system responses. The three decoders are jointly trained.

In order to train GECOR with the task-oriented dialogue model, we manually annotate the public task-oriented dialogue dataset CamRest676 with omitted expressions and substitute anaphora in the dataset with corresponding antecedents. The new dataset can be used either to train a standalone ellipsis or co-reference resolution model or to jointly train a task-oriented dialogue model equipped with the ellipsis / co-reference resolution model.

We conduct a series of experiments and analyses, demonstrating that the proposed method can significantly outperform a strong baseline model. Our contributions are threefold:

- We propose an end-to-end generative resolution model that attempts to solve the ellipsis and co-reference reolution in a single unified framework, significantly different from previous end-to-end co-reference resolution network with two phases of detection and candidate ranking.
- To the best of our knowledge, this is the first attempt to combine the task of ellipsis and co-reference resolution with the multi-turn task-oriented dialogue. The success rate of task completion is significantly improved with the assistance of the ellipsis and co-reference resolution.
- We construct a new dataset based on CamRest676 for ellipsis and co-reference resolution in the context of task-oriented dialogue.[1]

## 2 Related Work

**Ellipsis recovery:** The earliest work on ellipsis as far as we know is the PUNDIT system (Palmer et al., 1986) which discusses the communication between the syntactic, semantic and pragmatic modules that is necessary for making implicit linguistic information explicit. Dalrymple et al. (1991) and Shieber et al. (1996) establish a set of linguistic theories in the ellipsis recovery of English verb phrases. Nielsen (2003) first proposes an end-to-end computable system to perform English verb phrase ellipsis recovery on the original input text. Liu et al. (2016) propose to decompose the resolution of the verb phrase ellipsis into three sub-tasks: target detection, antecedent head resolution, and antecedent boundary detection.

**Co-reference resolution:** Co-reference resolution is mainly concerned with two sub-tasks, referring expressions (i.e., mentions) detection, and entity candidate ranking. Uryupina and Moschitti (2013) propose a rule-based approach for co-reference detection which employs parse tree features with an SVM model. Peng et al. (2015) improve the performance of mention detection by applying a binary classifier on their feature set. In recent years, applying deep neural networks to the co-reference resolution has gained great success. Clark and Manning (2016) apply reinforcement learning on mention-ranking co-reference resolution. Lee et al. (2017) introduce the first end-to-end co-reference resolution model. Lee et al. (2018) present a high-order co-reference resolution model with coarse-to-fine inference.

**Ellipsis and co-reference resolution in QA and Dialogue:** The methods mentioned above do not generalize well to dialogues because they normally require a large amount of well-annotated contextual data with syntactic norms and candidate antecedents. In recent years, a few studies try to solve ellipsis / co-reference resolution tailored for dialogue or QA tasks. Kumar and Joshi (2016) train a semantic sequence model to learn semantic patterns and a syntactic sequence model to learn linguistic patterns to tackle with the non-sentential (incomplete) questions in a question answering system. Zheng et al. (2018) builds a seq2seq neural network model for short texts to identify and recover ellipsis. However, these methods are still limited to short texts or one-shot dialogues. Our work is the first attempt to provide both solution and dataset for ellipsis and co-reference resolution in multi-turn dialogues.

**End-to-end task-oriented dialogue:** Task-oriented dialogue systems have evolved from traditional modularized pipeline architectures (Rudnicky et al., 1999; Zue et al., 2000; Zue and Glass, 2000) to recent end-to-end neural frameworks (Eric and Manning, 2017a,b; Lei et al.,

[1]The new dataset and the code of our proposed system are available at https://multinlp.github.io/GECOR/

| Turn | Dialogue |
|---|---|
| $Q_1$ | I would like an Italian restaurant. |
| $A_1$ | What price range do you have in mind? |
| $Q_2$ | I want cheap ones. |
| $A_2$ | Pizza Hut Cherry Hinton serves Italian food in the south part of town. Would you like their phone number? |
| $Q_3$ | Yes, please. |
| **User utterances after resolution** | |
| $Q_2$ | I want cheap Italian restaurants. |
| $Q_3$ | Yes, I would like the phone number please. |

Table 1: Examples of ellipsis and co-reference resolution

2018; Jin et al., 2018). Our work is an innovative combination of ellipsis and co-reference resolution and the end-to-end task-oriented dialogue.

## 3 The GECOR Model

In this section, we reformulate the ellipsis and co-reference resolution task in the context of multi-turn dialogue and detail the proposed GECOR model.

### 3.1 Ellipsis and Co-Reference Resolution Reformulation

Our task is to reconstruct a pragmatically complete utterance from a user utterance where the ellipsis and/or co-reference phenomena are present according to the dialogue context. Table 1 provides examples of reconstructed utterances in which the omitted information is recovered or the anaphor is substituted with referred expressions.

We attempt to solve the resolution of ellipsis and co-reference in a unified framework because in essence both ellipsis and co-reference can be understood from contextual clues. We consider these two problems in multi-turn dialogue and reformulate the resolution of them as a generation problem: generating the omitted or referred expressions. In this way, the modeling of ellipsis and co-reference is in line with response generation in dialogue modeling.

Unlike previous methods that combine detection and ranking models, our generation-based formulation is not constrained by the syntactic forms of ellipsis or co-reference in sentences. They can be either words (e.g., *noun, verb*) or phrases or even clauses. Furthermore, the formulation does not need to provide a set of candidate antecedents to be resolved. Previous studies usually need to traverse the text when there are multiple ellipsis or anaphora to be resolved, which leads to a high computational complexity.

In this reformulation, we assume that the dialogue context is composed of all utterances from the beginning of the dialogue to the current user utterance. Both the context and the user utterance in question are input to the GECOR model to generate the complete version of the user utterance.

### 3.2 Model Structure

The GECOR model is shown in Figure 1. The model essentially contains an embedding module, a user utterance encoder, a dialogue context encoder and a decoder with either copy (Gu et al., 2016) or gated copy mechanism (modified from See et al. (2017)). Both the generation probability over the entire vocabulary and the copy probability over all words from the dialogue context are taken into account for predicting the complete user utterance.

**Embedding Layer** In GECOR, we first tokenize the input user utterance and the dialogue context. We then use GloVe (Pennington et al., 2014) (the pre-trained 50-dimensional word vectors) in the embedding layer to obtain word embeddings. Let $\mathbf{U} = \{u_1, ..., u_m\}$, $\mathbf{C} = \{c_1, ..., c_n\}$ be representations of the tokenized utterance and context sequence.

**Utterance and Context Encoder** We use a single-layer bidirectional GRU to construct both encoders. The forward and backward hidden states over the input embeddings from the embedding layer are concatenated to form the hidden states of the two encoders.

**Decoder** The decoder is a single-layer unidirectional GRU. In the decoder, the attention distribution $a^t$ is calculated as in Bahdanau et al. (2015):

$$e_i^t = v^T tanh(W_h h_i + W_s s_{t-1} + b_{attn}) \quad (1)$$

$$a^t = softmax(e^t) \quad (2)$$

where $v$, $W_h$, $W_s$ and $b_{attn}$ are learnable parameters, $h_i$ is the hidden state for word $u_i$ from the sequence produced by the utterance encoder. The attention distribution $a^t$ is used to produce a weighted sum of the encoder hidden states, known as the context vector $h_t^*$:

$$h_t^* = \sum_i a_i^t h_i \quad (3)$$

It is fed into the single-layer unidirectional GRU together with the previous decoder state $s_t$ and the word embedding $y_{t-1}$ of the previously generated

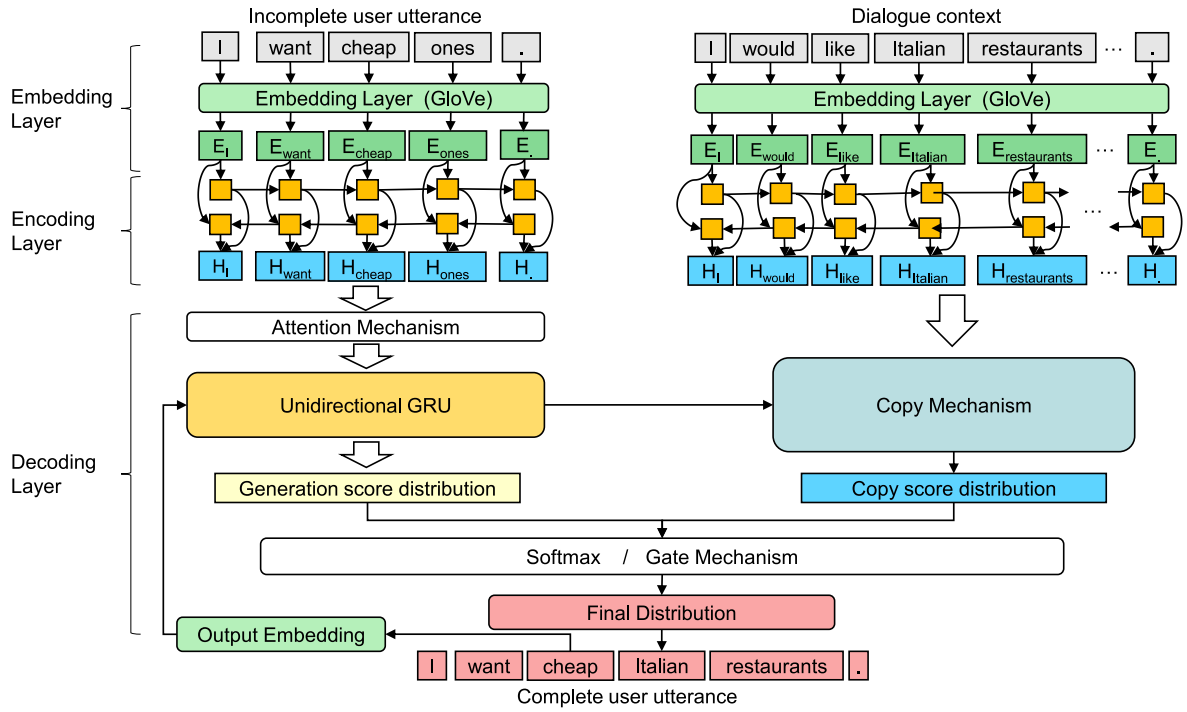


Figure 1: The end-to-end generative model for ellipsis and co-reference resolution (GECOR).

word to obtain the decoder state $s_t$. The updated $s_{t-1}$ is then concatenated with the context vector $h_t^*$ to produce the generation probability distribution over the vocabulary $\mathbf{V}$ as follows:

$$P^g(y_t) = \frac{1}{Z} e^{\psi_g(y_t)}, \quad y_t \in \mathbf{V} \tag{4}$$

$$\psi_g(y_t = v_i) = \mathbf{v}_i{}^T (W_g^h h_t^* + W_g^s s_t + b_g) \tag{5}$$

$$s_t = GRU([y_{t-1}; h_t^*], s_{t-1}) \tag{6}$$

where $W_g^h$, $W_g^s$ and $b_g$ are learnable parameters and $\mathbf{v}_i$ is the one-hot indicator vector for word $v_i \in \mathbf{V}$. $\psi_g$ is the score function for the generation-mode and $Z$ is the normalization term shared by the generation-mode and copy-mode.

**Copy Network** The copy network (Gu et al., 2016) is used to calculate the probabilities for words copied from the dialogue context. These words are parts of the omitted or referred expressions to be predicted. We build the copy network on the top of the context encoder. The probability for copying each word from the dialogue context is computed as follows:

$$P^c(y_t) = \frac{1}{Z} \sum_{i:c_i = y_t}^{|\mathbf{C}|} e^{\psi_c(c_i)}, \quad y_t \in \mathbf{C} \tag{7}$$

$$\psi_c(y_t = c_i) = \sigma(W_c h_i^c + b_c) s_t \tag{8}$$

where $W_c$ and $b_c$ are learnable parameters, $h_i^c$ is the output for word $c_i$ from the context encoder, and $\sigma$ is a non-linear activation function. $\psi_c$ is the score function for the copy-mode and $Z$ is the normalization term shared by equation (4) and (7).

Both probabilities from the two modes contribute to the final probability distribution over the extended vocabulary (the vocabulary plus the words from the dialogue context) which is calculated as follows:

$$P(y_t) = P^g(y_t) + P^c(y_t), \; y_t \in \mathbf{V} \cup \mathbf{C} \tag{9}$$

which is used to predict the final output word.

**Gated Copy** An alternative to the copy network is the gated copy mechanism that use a gate to regulate the contributions of the generation and copy mode to the final prediction. The gate $p_{gen}$ is calculated as follows:

$$p_{gen} = \sigma(W_h h_t^* + W_s s_t + W_y y_{t-1} + b_t) \tag{10}$$

$$P(y_t) = p_{gen} P^g(y_t) + (1 - p_{gen}) P^c(y_t) \tag{11}$$

where $W_h$, $W_s$, $W_y$ and $b_t$ are learnable parameters and $\sigma$ is the sigmoid function.

**Training** The standard cross-entropy loss is adopted as the loss function to train the GECOR model.

## 4 Task-Oriented Dialogue with GECOR

We integrate the proposed GECOR into an end-to-end task-oriented dialogue system TSCP proposed

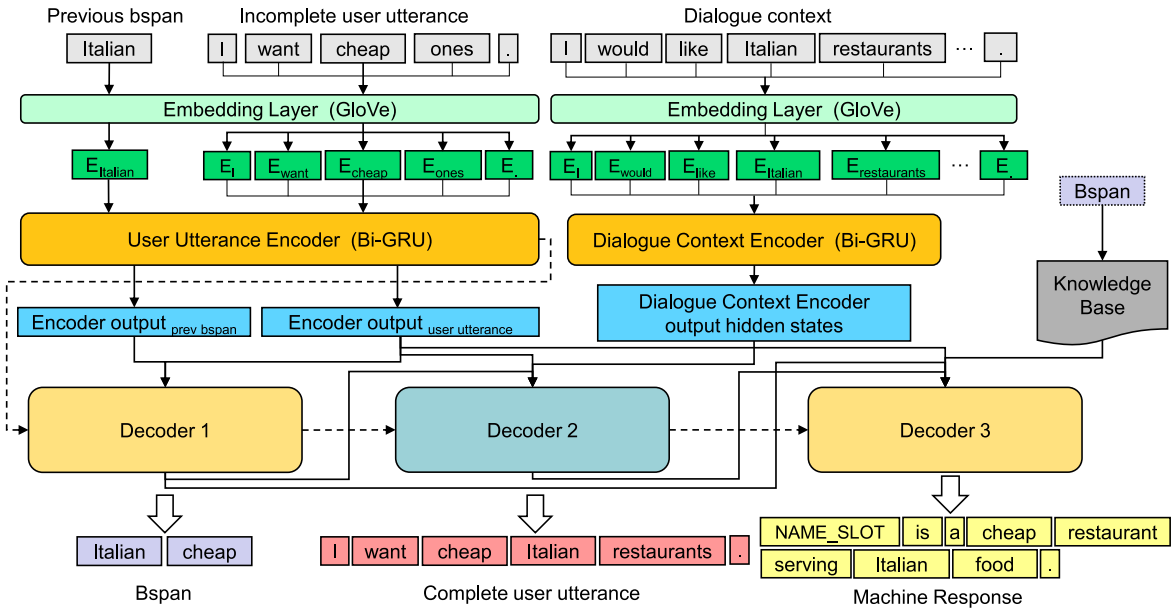


Figure 2: The architecture of the end-to-end task-oriented dialogue enhanced with the GECOR model. Decoder 1: BSpan decoder. Decoder 2: completed user utterance decoder. Decoder 3: machine response decoder.

by Lei et al. (2018) in a multi-task learning framework, which is shown in Figure 2. The GECOR-equipped TSCP model contains the embedding layer, the utterance and context encoders, and three decoders: decoder 1 for generating belief spans (BSpan) defined in (Lei et al., 2018) which are text spans for tracking dialogue states (e.g., $\langle inf \rangle Italian, cheap \langle /inf \rangle; \langle req \rangle phone \langle /req \rangle$), decoder 2 for complete user utterances and decoder 3 for machine responses. The embedding layer and encoders are the same as described in section 3.

**BSpan Decoder** Unlike Lei et al. (2018), we do not concatenate current user utterance with previously generated machine response. At each turn of dialogue, the user utterance and the previous BSpan (the dialogue states updated to the previous turn) are used as the inputs to the user utterance encoder. The outputs of this encoder are then fed into the BSpan decoder for predicting the new BSpan for the current turn and a cross-entropy loss $L_1$ is calculated. The user utterance encoder hidden states, the last hidden state and the output of the BSpan decoder are input into the other two decoders.

**Complete User Utterance Decoder** The basic structure of this decoder is the same as the decoder described in the last section.We pass the last hidden state of the BSpan decoder to the initial state of this decoder. In addition to the inputs from the user utterance encoder and the dialogue context encoder, we also input the output of the BSpan decoder into this decoder. The generation probability $P_t^g$, copy probability $P_t^{c1}$ for copying tokens in BSpan, and copy probability $P_t^{c2}$ for copying words in the dialogue context are calculated with a shared normalization term and combined for the final probability computation:

$$P_t = P_t^g + P_t^{c1} + P_t^{c2} \tag{12}$$

$P_t$ is then used to decode the words in the complete user utterance. For this decoder, the second cross-entropy loss $L_2$ is calculated.

**Machine Response Decoder** Similar to the previous two decoders, the machine response decoder is also a single-layer unidirectional GRU, the initial state of which is set to the last hidden state of the complete user utterance decoder. In this decoder, we compute three context vectors for each decoder state $s_t$. The first context vector $h_{t1}^*$ is calculated over the user utterance encoder hidden states while the other two context vectors $h_{t2}^*$, $h_{t3}^*$ are calculated over the BSpan decoder hidden states and the complete user utterance decoder hidden states, respectively. The concatenation of $s_t$, $h_{t1}^*$, $h_{t2}^*$, $h_{t3}^*$ and the Knowledge Base (KB) matching vector $K_t$ (a one-hot representation of the retrieval results in KB according to the constraints in the corresponding BSpan) is used to generate the output and update the decoder state. The generated output is then concatenated with the three context vectors to feed into a layer to produce the gener-

| Turn | Dialogue |
|---|---|
| $Q_1$ | I would like a traditional food restaurant. |
| $A_1$ | What price range do you have in mind? |
| $Q_2$ | I don't care. |
| $Q_2$ (Complete) | I don't care about the price range. |
| $Q_2$ (Ellipsis) | I don't care. |
| $Q_2$ (Co-reference) | I don't care about it. |

Table 2: An example of the ellipsis / co-reference annotation

ation probability distribution over the vocabulary. Similar to the complete user utterance decoder, we also use the copy mechanism in the machine response decoder. The third cross-entropy loss $L_3$ is then calculated.

**Training** The final loss for the multi-task learning framework is estimated as follows:

$$L = L_1 + L_2 + L_3 \quad (13)$$

We learn parameters to minimize the final loss.

## 5 Data Annotation for Ellipsis and Co-Reference Rosultion in Dialogue

Since there are no publicly available labeled data for the resolution of ellipsis and co-reference in dialogue, we manually annotate such a new dataset based on the public dataset CamRest676 (Wen et al., 2016a,b) from the restaurant domain.

**Annotation Specification** Annotation cases for user utterances can be summarized into the following three conventions:

- As shown in Table 2, if a user utterance contains an ellipsis or anaphor, we manually resolve the ambiguity of ellipsis or anaphor and supplement the user utterance with a correct expression by checking the dialogue context. In doing so, we create a pragmatically *complete* version for the utterance. If the utterance only contains an ellipsis and the ellipsis can be replaced with an anaphor, we create a *co-reference* version for it. Similarly, if the utterance only contains an anaphor and the anaphor can be omitted, we create an *ellipsis* version for the utterance.

- If the user utterance itself is pragmatically complete, without any ellipsis or anaphora, we create an anaphor and ellipsis version for it if such a creation is appropriate.

- If the utterance itself is complete and it is not suitable to create an ellipsis or anaphor version, we just do nothing.

With the annotation convention described above, for each user utterance in the dataset, we can label it as $l \in \{$*ellipsis*, *co-reference*, *complete*$\}$ or create two other versions for it if appropriate. Please notice that these labels are used only for dataset statistics or for designing experiments, not for training our models.

**Dataset statistics** The CamRest676 dataset contains 676 dialogues, with 2,744 user utterances. After annotation, 1,174 *ellipsis* versions and 1,209 *co-reference* versions are created from the 2,744 user utterances. 1,331 incomplete utterances are created that they are an either *ellipsis* or *co-reference* version. 1,413 of the 2,744 user utterances are complete and not amenable to change. No new versions are created from these 1,413 utterances.

**Dataset Split for Experiments** We split the new dataset into a training set (accounting for 80%) and validation set (accounting for 20%) which can be used for the stand-alone ellipsis/co-reference resolution task and the multi-task learning of both the ellipsis/co-reference resolution and end-to-end task-oriented dialogue.

## 6 Experiments

In this section we conducted experiments on the new dataset to examine the generative ellipsis/co-reference resolution model and its integration into the end-to-end task-oriented dialogue.

### 6.1 Evaluation Metrics

As far as we know, there is no end-to-end generative ellipsis and co-reference resolution model applied to multi-turn dialogues. Therefore there are no off-the-shelf metrics tailored to this evaluation. Since we deal with two tasks: the task of ellipsis/co-reference resolution (resolution task for short) and the task-oriented dialogue with integrated ellipsis/co-reference resolution (hereafter dialogue task), we use two sets of evaluation metrics. For the resolution task, we use the exact match rate (EM) that measures whether the generated utterances exatly match the gold utterances.

BLEU (Papineni et al., 2002) and $F_1$ score (a balance between word-level precision and recall) are also used for the resolution task to evaluate the quality of generated utterances at the n-gram and word level. We use the success $F_1$ which is defined as the $F_1$ score of requested slots correctly answered in dialogues to evaluate task comple-

| Data | Model | Resolution Task | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | EM(%) | EM 1(%) | EM 2(%) | BLEU(%) | $F_1$(%) | Prec.(%) | Rec.(%) | Reso.$F_1$(%) |
| Ellipsis | Baseline | 49.99 | 68.88 | 27.31 | 73.26 | 90.89 | 92.14 | 89.67 | 44.47 |
| | GECOR 1 | 67.56 | 92.07 | 37.18 | 83.69 | 96.25 | 98.28 | 94.30 | 70.48 |
| | GECOR 2 | 67.75 | 91.38 | 38.46 | 82.94 | 96.58 | 98.48 | 94.76 | 70.85 |
| Co-reference | Baseline | 55.64 | 76.03 | 33.60 | 78.12 | 92.58 | 93.28 | 91.89 | 44.24 |
| | GECOR 1 | 71.35 | 91.67 | 47.68 | 85.89 | 96.49 | 98.19 | 94.86 | 64.93 |
| | GECOR 2 | 71.18 | 93.80 | 44.92 | 85.93 | 97.09 | 98.46 | 95.76 | 71.26 |
| Mixed | Baseline | 50.38 | 70.89 | 28.57 | 74.11 | 90.93 | 91.72 | 90.15 | 44.10 |
| | GECOR 1 | 68.52 | 92.03 | 42.04 | 83.91 | 95.88 | 98.12 | 93.74 | 66.06 |
| | GECOR 2 | 66.22 | 91.64 | 37.45 | 82.98 | 96.47 | 98.41 | 94.60 | 66.16 |

Table 3: Results of the resolution task on the dataset. GECOR 1/2: the GECOR model with the copy/gated copy mechanism. EM 1 and EM 2 respectively indicate the situation that the input utterance is complete or incomplete while EM is the comprehensive evaluation of the two situations. Reso.$F_1$: Resolution_$F_1$

tion rate for the dialogue task, similar to Lei et al. (2018).

### 6.2 Parameter Settings

For all our models, both the size of hidden states and word embeddings were set to 50. The vocabulary size $|\mathbf{V}|$ was set to 800 and the batch size was set to 32. We trained our models via the Adam optimizer (Kingma and Ba, 2015), with a learning rate of 0.003 and a decay parameter of 0.5. Early stopping and dropout were used to prevent overfitting, and the dropout rate was set to 0.5.

### 6.3 Baselines and Comparisons

For the resolution task, we compared our GECOR model with the baseline model proposed by Zheng et al. (2018) which is a seq2seq neural network model that identifies and recovers ellipsis for short texts.

For the dialogue task, we compared our multi-task learning framework with the baseline model TSCP proposed by Lei et al. (2018) which is a seq2seq model enhanced with reinforcement learning. We ran the source code[2] on our dataset to get the baseline results for comparison.

For the resolution task, we also performed a comparison study to examine the impacts of the gate mechanism incorporated into the copy network on the GECOR model and on the multi-task learning dialogue model.

### 6.4 The GECOR Model

Our generative resolution model was trained on three types of data: the ellipsis data where only ellipsis version utterances from the annotated dataset were used, the co-reference data where only co-reference version utterances from the annotated dataset were used, and the mixed data where we randomly selected a version for each user utterance from {*ellipsis*, *co-reference*, *complete*}. In the mixed data, 633 turns are with ellipsis user utterances, 698 turns are with co-reference user utterances, and the rest are with complete user utterances. The experimental results of the GECOR and baseline model (Zheng et al., 2018) on the three different datasets are shown in Table 3.

**Overall results** From the third column of the table, we find that the GECOR model with the copy mechanism (GECOR 1) improves the exact match rate (EM) by more than 17 points on the ellipsis version data, more than 15 points on the co-reference data, and more than 18 points on the mixed data. We further define a metric we term as **Resolution_$F_1$** that is an $F_1$ score calculated by comparing machine-generated words with ground truth words for only the ellipsis / co-reference part of user utterances. The GECOR model achieves consistent and significant improvements over the baseline in terms of BLEU, $F_1$ and Resolution_$F_1$ in addition to the EM metric . The major difference between the GECOR and the baseline is that the former tries to copy words from the dialogue context. The improvements, especially the improvements on the ellipsis resolution (higher than those on the co-reference resolution) indicate that the copy mechanism is crucial for the recovery of ellipsis and co-reference.

**Effect of the two copy mechanisms** Comparing the GECOR 1 against the GECOR 2 (with the gated copy mechanism), we can find that the gating between copy and generation is helpful in terms of the word-level quality ($F_1$ and Resolution_$F_1$ score) but not in terms of the fragment

[2]https://github.com/WING-NUS/sequicity

| Data | Model | Resolution Task | | | | | Dialogue Task | | |
|---|---|---|---|---|---|---|---|---|---|
| | | EM(%) | BLEU(%) | F1(%) | Prec.(%) | Rec.(%) | Succ.F1(%) | Prec.(%) | Rec.(%) |
| Complete | TSCP | - | - | - | - | - | 86.30 | 89.60 | 83.23 |
| Ellipsis | TSCP | - | - | - | - | - | 84.56 | 87.25 | 82.02 |
| | Our Model | 60.83 | 78.89 | 95.64 | 97.79 | 93.58 | **85.33** | 88.69 | 82.21 |
| Co-reference | TSCP | - | - | - | - | - | 82.17 | 88.91 | 76.38 |
| | Our Model | 68.56 | 83.98 | 96.61 | 98.09 | 95.18 | **86.00** | 90.46 | 81.95 |
| Mixed | TSCP | - | - | - | - | - | 83.25 | 86.91 | 79.89 |
| | Our Model | 66.47 | 83.63 | 96.26 | 98.16 | 94.44 | **85.97** | 87.98 | 84.05 |

Table 4: Results of the multi-task learning model. This table is split into two parts: performance of resolution for the integrated GECOR on the left side and performance of dialogue task on the right side.

or sequence-based metrics (i.e., BLEU and EM). Therefore, we only integrate the GECOR model with the copy mechanism into the dialogue system.

**Incomplete vs. complete utterances** In multi-turn dialogues, user utterances may be incomplete or complete. A robust resolution model needs to be able to accurately identify whether the input utterance is complete or not. The model needs to keep it unchanged when it is complete and to predict the corresponding complete version when it is incomplete. For these cases, we tested our models and made statistical analysis on the three versions of data as shown in column 3, 4 and 5 of Table 3 (EM, EM 1, EM 2). We can find that the GECOR model beats the baseline model in all respects. However, the GECOR model needs further improvement when the input utterances are incomplete, compared with its good performance on complete utterances.

**Analysis on GECOR results for complete utterances** We then analyzed the experimental results of the GECOR 1 on the mixed data in detail. When the input user utterances are complete, the GECOR model can amazingly generate 92.03% utterances that exactly match the input utterances. Only 7.97% do not match perfectly. Most unmatched cases, as we found, are with: **(1) missed words** (e.g., User: *Can I get a Korean restaurant in the town centre?* GECOR: *Can I get a Korean restaurant in the town?*) **(2) Repetition** (e.g., User: *OK, thank you. That is all for today then.* GECOR: *OK, thank you. That is all for today for today then.*)

**Analysis on GECOR results for incomplete utterances** For incomplete input user utterances, GECOR can generate 42.04% exactly matched cases. Among the 57.96% cases that do not exactly match ground truth utterances, only 6.3% are not complete, which still contains unresolved ellipsis or co-reference, while 93.7% of these cases are complete with GECOR-generated words that do not match ground truth words. An in-depth analysis on these show that they can be clustered into 4 classes. **(1) Paraphrases.** We found that the majority of the unmatched complete utterances generated by GECOR are actually paraphrases to the ground truth complete utterances (e.g., User: *Any will be fine.* GECOR: *Any food type will be fine.* Reference: *Any type of restaurant will be fine.*). This is also confirmed by the high scores of the word-level evaluation metrics in Table 3. **(2) Partial resolution.** When a pronoun refers to more than one items, GECOR sometimes generate a partial resolution for the pronoun (e.g., User: *I do not care about them.* GECOR: *I do not care about the price range.* Reference: *I do not care about the price range or location.*). **(3) Minor errors.** In a few cases, the resolution part is correct while there are some errors elsewhere. (e.g., User: *How about Chinese food?* Prediction: *How about international food on the south side of town?* Reference: *How about Chinese food on the south side of town?*) **(4) Repetition.** Some cases contain repeatedly generated words.

We think that although not exactly matched, paraphrased complete utterances generated by GECOR are acceptable. These utterances are helpful for the downstream dialogue task. For other errors, such as partial resolution or repetition, it may be necessary to enhance the attention or copy mechanism further in GECOR.

### 6.5 The Multi-Task Learning Model

We further conducted experiments to extrinsically evaluate the GECOR model in task-oriented dialogue with the success $F_1$ metric. This is also to evaluate our multi-task learning framework in integrating the GECOR model into the end-to-end dialogue model. In addition to training the base-

line TSCP model on the ellipsis, co-reference and mixed dataset, we also trained it on the dataset with only complete user utterances. This is to examine the ability of the baseline model in using correct contextual information presented in user utterances. The experimental results are shown in Table 4.

**Overall results** In comparison to the baseline, we can see that our model improves the success $F_1$ score by nearly 4 points on the co-reference dataset, which is close to the score obtained by the baseline trained with the complete user utterances. On the mixed and ellipsis dataset, our model also achieves 2.7 points and 0.8 points of success $F_1$ score improvements, respectively.

**Resolution performance of the integrated GECOR** We also provide the performance of the integrated GECOR on the resolution task in Table 4. The performance is slightly lower than when the GECOR is trained independently as a standalone system. This suggests that the GECOR is able to perform well when integrated into a dialogue system. The overall results demonstrate that the proposed multi-task learning framework for the end-to-end dialogue is able to improve the task completion rate by incorporating an auxiliary ellipsis/co-reference resolution task.

Since the BSpan decoder is also used in the baseline system to capture contextual information and track dialogue states, we believe that our multi-task learning model with the integrated GECOR will play a more important role in end-to-end dialgoue models that do not use state tracking modules, e.g., neural open-domain conversation models (Vinyals and Le, 2015; Li et al., 2016).

## 7 Conclusion and Future Work

In this paper, we have extensively investigated the ellipsis and co-reference resolution in the context of multi-turn task-oriented dialogues. We have presented the GECOR, a unified end-to-end generative model for both ellipsis and co-reference resolution in multi-turn dialogues. A multi-task learning framework is further proposed to integrate the GECOR into the end-to-end task-oriented dialogue. In order to train and test the proposed model and framework, we manually created a new dataset with annotated ellipsis and co-reference information based on the publicly available CamRest676 dataset. Experiments on the resolution task show that the GECOR is able to significantly improve the performance in terms of the exact match rate, BLEU and word-level $F_1$ score. Experiments on the dialogue task demonstrate that the task completion rate of the task-oriented dialogue system is significantly improved with the aid of ellipsis and co-reference resolution.

Our work could be extended to end-to-end open-domain multi-turn dialogue. We will further improve our model by incorporating syntactic and location information. We would also like to adapt the proposed methods to document-level neural machine translation in the future.

## Acknowledgments

The present research was supported by the National Natural Science Foundation of China (Grant No.61861130364) and the Royal Society (London) (NAF\R1\180122). We would like to thank the anonymous reviewers for their insightful comments.

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*.

Kevin Clark and Christopher D Manning. 2016. Deep reinforcement learning for mention-ranking coreference models. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2256–2262.

Mary Dalrymple, Stuart M Shieber, and Fernando CN Pereira. 1991. Ellipsis and higher-order unification. *Linguistics and philosophy*, 14(4):399–452.

Mihail Eric and Christopher D Manning. 2017a. A copy-augmented sequence-to-sequence architecture gives good performance on task-oriented dialogue. *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers,*, pages 468–473.

Mihail Eric and Christopher D Manning. 2017b. Key-value retrieval networks for task-oriented dialogue. *Proceedings of the SIGDIAL 2017 Conference*, pages 37–49.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1631–1640.

Xisen Jin, Wenqiang Lei, Zhaochun Ren, Hongshen Chen, Shangsong Liang, Yihong Zhao, and Dawei

Yin. 2018. Explicit state tracking with semi-supervisionfor neural dialogue generation. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1403–1412. ACM.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*.

Vineet Kumar and Sachindra Joshi. 2016. Non-sentential question resolution using sequence to sequence learning. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2022–2031.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197.

Kenton Lee, Luheng He, and Luke Zettlemoyer. 2018. Higher-order coreference resolution with coarse-to-fine inference. *Proceedings of NAACL-HLT 2018*, pages 687–692.

Wenqiang Lei, Xisen Jin, Min-Yen Kan, Zhaochun Ren, Xiangnan He, and Dawei Yin. 2018. Sequicity: Simplifying task-oriented dialogue systems with single sequence-to-sequence architectures. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1437–1447.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016. Deep reinforcement learning for dialogue generation. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202.

Zhengzhong Liu, Edgar Gonzàlez Pellicer, and Daniel Gillick. 2016. Exploring the steps of verb phrase ellipsis. In *Proceedings of the Workshop on Coreference Resolution Beyond OntoNotes (CORBON 2016)*, pages 32–40.

Leif Arda Nielsen. 2003. A corpus-based study of verb phrase ellipsis. In *Proceedings of the 6th Annual cluk Research Colloquium*, pages 109–115.

Martha S. Palmer, Deborah A. Dahl, Rebecca J. Schiffman, Lynette Hirschman, Marcia Linebarger, and John Dowding. 1986. Recovering implicit information. In *Proceedings of the Workshop on Strategic Computing Natural Language*, HLT '86, pages 96–113, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Haoruo Peng, Kai-Wei Chang, and Dan Roth. 2015. A joint framework for coreference resolution and mention head detection. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 12–21.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Alexander I Rudnicky, Eric Thayer, Paul Constantinides, Chris Tchou, R Shern, Kevin Lenzo, Wei Xu, and Alice Oh. 1999. Creating natural dialogs in the carnegie mellon communicator system. In *Sixth European Conference on Speech Communication and Technology*.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Stuart M Shieber, Fernando CN Pereira, and Mary Dalrymple. 1996. Interactions of scope and ellipsis. *Linguistics and philosophy*, 19(5):527–552.

Olga Uryupina and Alessandro Moschitti. 2013. Multilingual mention detection for coreference resolution. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 100–108.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *Proceedings of the International Conference on Machine Learning, Deep Learning Workshop (2015)*.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Lina M Rojas Barahona, Pei-Hao Su, Stefan Ultes, David Vandyke, and Steve Young. 2016a. Conditional generation and snapshot learning in neural dialogue systems. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2153–2162.

Tsung-Hsien Wen, David Vandyke, Nikola Mrksic, Milica Gasic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2016b. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 438–449.

Jie Zheng, Fang Kong, and Guodong Zhou. 2018. Sequence to sequence model to ellipsis recovery for chinese short text. *Journal of Chinese Information Processing*, 32(12):92–99.

Victor Zue, Stephanie Seneff, James R Glass, Joseph Polifroni, Christine Pao, Timothy J Hazen, and Lee Hetherington. 2000. Juplter: a telephone-based

conversational interface for weather information. *IEEE Transactions on speech and audio processing*, 8(1):85–96.

Victor W Zue and James R Glass. 2000. Conversational interfaces: Advances and challenges. *Proceedings of the IEEE*, 88(8):1166–1180.

# Task-Oriented Conversation Generation Using Heterogeneous Memory Networks

**Zehao Lin** [1] **Xinjing Huang** [1] **Feng Ji** [2] **Haiqing Chen** [2] **Yin Zhang**[1]*
[1] College of Computer Science and Technology, Zhejiang University
[2] DAMO Academy, Alibaba Group
{georgelin,huangxinjing,zhangyin98}@zju.edu.cn
{zhongxiu.jf,haiqing.chenhq}@alibaba-inc.com

## Abstract

How to incorporate external knowledge into a neural dialogue model is critically important for dialogue systems to behave like real humans. To handle this problem, memory networks are usually a great choice and a promising way. However, existing memory networks do not perform well when leveraging heterogeneous information from different sources. In this paper, we propose a novel and versatile external memory networks called Heterogeneous Memory Networks (HMNs), to simultaneously utilize user utterances, dialogue history and background knowledge tuples. In our method, historical sequential dialogues are encoded and stored into the context-aware memory enhanced by gating mechanism while grounding knowledge tuples are encoded and stored into the context-free memory. During decoding, the decoder augmented with HMNs recurrently selects each word in one response utterance from these two memories and a general vocabulary. Experimental results on multiple real-world datasets show that HMNs significantly outperform the state-of-the-art data-driven task-oriented dialogue models in most domains.

## 1 Introduction

Compared with chitchat, task-oriented dialogue systems aim at solving tasks in specific domains with grounding knowledge. Though far from handling conversation like a real human, existing task-oriented dialogue systems have shown cheerful prospect in a specific domain, e.g. Siri and Cortana are personal assistants, helping people a lot in daily life and business work.

In general, knowledge-grounded task-oriented dialogue system can be divided into three important components: understanding user utterances, fetching right knowledge from external storage and replying right answer. As shown in Figure 1, agent is required to do a point-of-interest navigation. According to dialogue history, agent will fetch related knowledge base information, in our case represented as tuples (e.g. [$hotel\_keen$, $poi\_type$, $rest\_stop$], which indicates the point-of-i nterests type of $hotel\ keen$ is $rest\ stop$), as an external knowledge to answer correctly and complete task.

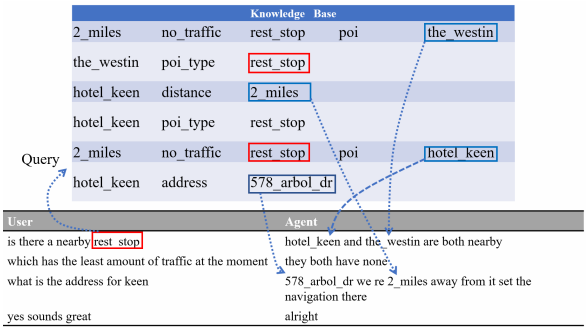


Figure 1: A multi-turn dialogue example. The upper table shows several n-tuples sampled from knowledge base. Lower table shows multi-turn dialogues. Agent needs to retrieve appropriate knowledge tuples to generate the proper response.

Traditional pipeline dialogue systems(Yan et al., 2017; Rojas-Barahona et al., 2017) and some end-to-end dialogue systems rely on the predefined the slot filling labels. Despite the consumption of human efforts, these kinds of systems are difficult to adapt to new domains.

Many works, e.g. (Vinyals and Le, 2015; Shang et al., 2015), show that training a fully data-driven end-to-end model is a promising way to build domain-agnostic dialogue system. Their models mostly try to use the attention mechanism, including memory networks techniques, to fetch the most similar knowledge (Sukhbaatar et al., 2015), then incorporate grounding knowledge into a seq2seq neural model to generate a suitable re-

*Corresponding Author



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4558–4567, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

sponse (Madotto et al., 2018).

However, existing memory networks equally treat information from multiple sources, e.g. sequential dialogue history and structure knowledge bases. Therefore two weaknesses arise in such methods: (1) It is difficult to model different types of structured information in only one memory network. (2) It is also difficult to model the effectiveness of knowledge from different sources in such a single memory network. To address these issues, we expand the architecture of memory networks used in a seq2seq neural model.

Our contributions are mainly three-fold:

- We propose a novel seq2seq neural conversation model augmented with Heterogeneous Memory Networks. We first model sequential dialogue history and grounding external knowledge with two different kinds of memory networks and then feed the output of context-aware memory to the context-free memory to search the representations of similar knowledge.

- Our context-aware memory networks is able to learn the context-aware latent representations and stores them into memory slots, by employing a gating mechanism when encoding dialogue history and user utterance.

- Experimental results demonstrate that our neural approach significantly outperforms the examined neural methods automatic metrics, and context-aware memory networks can learn and store more meaningful representations than the examined memory approaches.

## 2 Related Works

The end-to-end model uses deep neural net instead of several parts in pipeline models to generate responses. (Rojas-Barahona et al., 2017) propose a data-driven goal-oriented neural dialogue system by adding database operator and policy networks modules to introduce database information and track state which need extra labeling step that breaks differentiability. (Bordes and Weston, 2016) propose a testbed to break down the strengths and shortcomings of end-to-end dialog systems in goal-oriented applications. Those methods treated dialogue system as the problem of learning a mapping policy from dialogue histories to agents' responses.

The booming internet dialogue data lay the foundation of building data-driven models. (Ritter et al., 2011) first applied phrase-based Statistical Machine Translation(Setiawan et al., 2005). It treats the conversation system as a translation problem, a user utterance needs to be translated into an agent response.

(Sutskever et al., 2014) propose Sequence to sequence model (SEQ2SEQ) architecture and apply it to neural machine translation task. SEQ2SEQ has become a general basis of natural language generation tasks, e.g. question answering(Tan et al., 2018) and question generation (Zhou et al., 2017). By applying the RNN based encoder-decoder framework to generate responses, models(Shang et al., 2015; Cho et al., 2014b; Luong et al., 2015b) are able to utilize neural networks to learn the representation of dialogue histories and generate appropriate responses.

To deal with multi-turn information, (Sordoni et al., 2015) propose a model that represents the whole dialogue history (including the current message) with continuous representations or embeddings of words and phrases to address the challenge of the context-sensitive response generation.

By adding a knowledge base module, recent works (Ghazvininejad et al., 2018; Young et al., 2018) have shown the possibility of training an end-to-end task-oriented dialogue system on the sequence to sequence architecture. Ghazvininejad et al. (Ghazvininejad et al., 2018) generalize the SEQ2SEQ approach by conditioning responses on both conversation history and external knowledge, aiming at producing more contextual responses without slot filling.

CopyNet (Gu et al., 2016) and Pointer Networks (Vinyals et al., 2015) improve model's accuracy and ability of handling of out-of-vocabulary words using neural attention. Pointer-Generator networks (See et al., 2017) apply copy mechanism to the neural generation model. Their work shows copy mechanism can improve quality in text generation. (Dhingra et al., 2017) and (Li et al., 2017) apply reinforcement learning to make it differentiable.

Recent works on external memory (Graves et al., 2014; Henaff et al., 2016) provide an efficient method of introducing and reasoning different types of external information. (Sukhbaatar

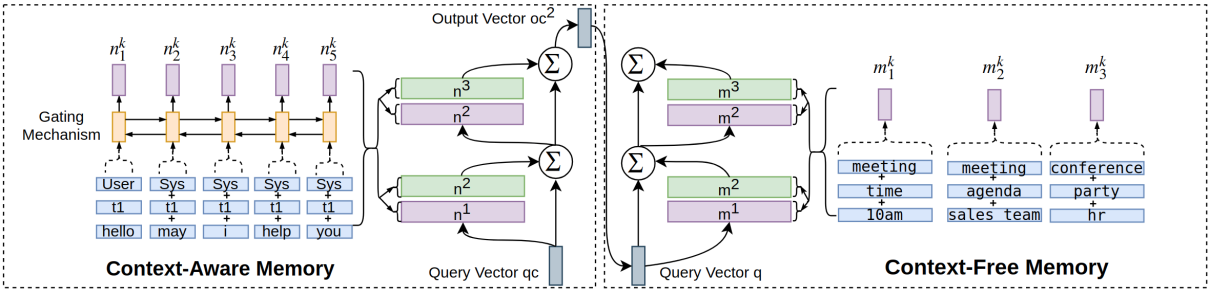


Figure 2: An example of Heterogeneous Memory Networks with two-hop attention. Context-aware memory which encodes dialog history to a context vector $oc^2$ while context-free memory loads knowledge base information. The output of context-aware memory will be employed as the query vector to the context-free memory.

et al., 2015) propose end-to-end memory networks with multiple attention hops model over a possibly large external memory. (Madotto et al., 2018) propose Mem2Seq that combines the end-to-end memory networks with the idea of pointer networks. (Chen et al., 2018) add the hierarchical structure and the variational memory network to capture both the high-level abstract variations and long-term memories during the dialogue tracking. To take care of information from different sources, (Fu and Feng, 2018) propose an attention mechanism to encourage the decoder to actively interact with the memory by taking its heterogeneity into account.

## 3 Proposed Framework

To generate responses using dialogue history and grounding knowledge, we introduce a novel encoder-decoder neural conversation model augmented with Heterogeneous Memory Networks (HMNs). The encoder module adopts a context-aware memory network to better understand the dialogue history and query. The decoder is enhanced with HMNs, which is able to incorporate external knowledge and dialog history when generating words.

### 3.1 Encoder

The encoder encodes the dialogue history into a fixed context vector. Here we adopt the context-aware memory as our encoder module. As shown in the left part of Figure 2, each word will be extended to the following parts: 1) token itself. 2) A turn tag. 3) An identity tag. For example, in the first turn a user says "hello" and the response from system is "may I help you", it will be concatenated as [(hello, t1, user),(may, t1, sys),(I, t1, sys),(help, t1, sys), (you, t1, sys)], *sys* means the word comes

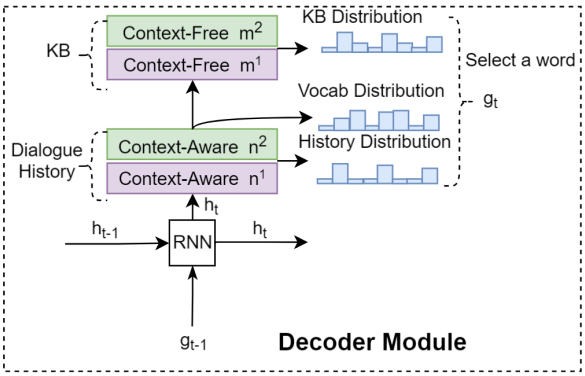


Figure 3: Decoder module with two attention hops

from the dialog system, so does *user*. *t1* indicates the word is from the first turn. Each word can be transformed into vector by embedding lookup, and we sum up vectors in each tuple to be the input sequence of the context-aware memory. Using a fixed vector to query the memory, a context vector $c$ can be obtained.

#### 3.1.1 Context-Aware Memory

To efficiently model the context information of sequential data, we present the context-aware memory. Memory slots $n^k$ are structured by concatenating all input vectors as $n^k = cat[n_1^k, n_2^k, ..., n_l^k]$, k stands for the k-th hop in memory, and $l$ means the length of the input. $n_l^k$ is the sum of each word tag embedding using each hop own randomly initialized embedding matrix $C^k$. And we adopt the adjacent weight sharing scheme, which means $C^k$ is not only the input embedding matrix in the k-th hop, but also the output embedding matrix in the (k-1)-th hop. We add a gating mechanism between memory cells. The gating mechanism applied is adopted from Bidirectional GRU (Cho et al., 2014a) in our case. Thus the context-dependent representation of in-

puts, denoted as $n^k = [\overrightarrow{n^k}, \overleftarrow{n^k}]$, where $\overrightarrow{n^k}$ and $\overleftarrow{n^k}$ are the forward and backward representation of inputs, respectively. The forward process can be illustrated as equations:

$$r_t = \sigma(W_1 v_t^k + W_2 \overrightarrow{n^k}_{(t-1)} + b_1) \quad (1)$$

$$z_t = \sigma(W_3 v_t^k + W_4 \overrightarrow{n^k}_{(t-1)} + b_2) \quad (2)$$

$$e_t = \tanh(W_5 v_t^k + r_t(W_6 \overrightarrow{n^k}_{(t-1)} + b_3)) \quad (3)$$

$$\overrightarrow{n_t^k} = (1 - z_t)e_t + z_t \overrightarrow{n^k}_{(t-1)} \quad (4)$$

where $W_1$, $W_2$, $W_3$, $W_4$, $W_5$, $W_6$ and $b_1$, $b_2$, $b_3$ are trainable weights and biases.

Given the query vector $qc^k$, attention weights over memory cells $n^k$ can be calculated by the equation:

$$p^k = Softmax(n^k \cdot qc^k) \quad (5)$$

the readout vector is the sum of output memory matrix $n^{k+1}$ with corresponding attention weights $p^k$.

$$u^k = \sum_i (p_i^k \cdot n_i^{k+1}) \quad (6)$$

By summing query and readout vector together, we can get the output from the k-th hop.

$$oc^k = qc^k + u^k \quad (7)$$

Note $oc^k$ is also the query vector of the (k+1)-th hop.

### 3.2 Decoder

The decoder contains HMNs and an RNN controller, as shown in Figure 3. The controller controls the process of querying HMNs.

In each time step, HMNs will generate: 1) a readout vector $oc^1$, which is the output of the first hop in history memory, and 2) attention weights of the last hop in two memories, called history word distribution $P_{his}$ and knowledge tuple distribution $P_{kb}$. The readout vector is concatenated with $h_t$ to predict the vocabulary distribution $P_{vocab}$. Formally,

$$P_{vocab} = Softmax(W_7[h_t, oc^1]) \quad (8)$$

where $W_7$ is trainable weight matrix.
We adopt a simple strategy ( Section 3.5 ) to select a word from three distributions $P_{vocab}$, $P_{his}$ and $P_{kb}$.

#### 3.2.1 Heterogeneous Memory Networks

HMNs stacks two types of memory: 1) context-aware memory and 2) context-free memory. Dialog history is loaded into context-aware memory, and knowledge base triples are loaded into context-free memory. Firstly, HMNs accepts query vector as inputs, then walk through context-aware memories. The final output $u^k$ in the last hop will be employed to query context-free memory. Context-aware memory has been detailed in Section 3.1.1.

Context-Free memory itself is end-to-end memory networks (Sukhbaatar et al., 2015). Compared with our context-aware memory, it has no gating mechanism. The input to the memory is the summed vectors in each knowledge triple, as depicted in the right part of Figure 2. Each hop owns randomly initialized embedding matrix $C^{k'}$ We denote memory slots as $m^k$. It accepts a query vector and then follows the same process 5 to 7, the output $u^{k'}$ can be obtained.

#### 3.2.2 Controller

We adopt GRU as our controller. It accepts the output $c$ from encoder as initial hidden state $h_0$. In each time step, it takes the previous generated word $g_{t-1}$'s embedding $E(g_{t-1})$ and last time hidden state $h_{t-1}$ as inputs. Formally:

$$h_t = GRU(E(g_{t-1}), h_{t-1}) \quad (9)$$

then $h_t$ is used to query the HMNs.

### 3.3 Copy Mechanism

We adopt copy mechanism to copy words from memories. Attention weights in the last hop of the two memories,$P_{kb}$ and $P_{his}$ will be the probability of the target word from those memories. If the target word does not appear in inputs, the position index will be the last position in memories, which is a sentinel added in preprocessing stage.

### 3.4 Joint Learning

To learn the distribution of three vocabularies $P_{vocab}$, $P_{kb}$ and $P_{his}$ in each time step, the loss in the t-th time step is the negative log-likelihood of the predict probability of the target word for that time step. Formally:

$$Loss = -\frac{1}{T} \sum_{t=0}^{t=T} \sum_i (\log p_{ti}) \quad (10)$$

Note that $p_{it}$ means the t-th word's probability in $i \in \{P_{vocab}, P_{kb}, P_{his}\}$ .

Table 1: The statistics of the bAbI-3, 4, 5, DSTC2 and Key-Value Retrieval datasets

| Datasets | Key-Value Retrieval dataset | DSTC 2 | bAbI-3 | bAbI-4 | bAbI-5 |
|---|---|---|---|---|---|
| Avg. History words | 25.5 | 63.4 | 49.9 | 20.5 | 62.6 |
| Avg. KB pairs | 64.7 | 42.7 | 23.4 | 7.0 | 23.6 |
| Avg. Response Length | 8.7 | 10.2 | 7.2 | 5.7 | 6.5 |
| Vocabulary Size | 1554 | 1066 | 739 | 1004 | 1135 |
| Dialogue Turns | 2.8 | 9.9 | 10.9 | 4.5 | 19.3 |

### 3.5 Word Selection Strategy

In our case, if words with the highest probability in $P_{his}$ and $P_{kb}$ vocabularies are not on sentinel positions, we directly compare the probability of each word and select the higher one. If one of the vocabularies points to the sentinel position, the model will select the word with the highest probability in the other vocabulary. At last, if both vocabularies get to sentinel positions, the word from $P_{vocab}$ will be selected.

## 4 Experimental Setup

### 4.1 Datasets

As the proposed approach is quite general, the model can be applied to any task-oriented dialogue datasets with conversation and knowledge base data. To evaluate and compare the results with the state-of-the-art methods in multiple dimensions, we choose three popular task conversation datasets including DSTC 2, Key-Value Retrieval dataset and the (6) dialog bAbI tasks. Table 1 shows the statistics of datasets.

- *Key-Value Retrieval dataset* (Eric and Manning, 2017). This dataset releases a corpus of 3,031 multi-turn dialogues. The dialogues consist of three different domains: calendar scheduling, weather information retrieval, and point-of-interest navigation.

- *The (6) dialog bAbI tasks* (Bordes and Weston, 2016). The (6) dialog bAbI tasks are a set of five subtasks within the goal-oriented context of restaurant reservations. Conversations are grounded with an underlying knowledge base of restaurants and their properties (location, type of cuisine, etc.). As task 1 and 2 have been achieved very well, we only test our model on task 3 to 5 and their OOV(out-of-vocabulary), where entities (e.g. restaurant names) in test sets may not have been able to see during training.

- *The Dialog State Tracking Challenge 2* (DSTC 2). DSTC 2 is a research challenge focused on improving the state-of-the-art in tracking the state of spoken dialogue systems. DSTC 2's training dialogues were gathered using Amazon Mechanical Turk related to restaurant search.

For all datasets, we employ the original conversation and knowledge base information only and drop the other labels e.g. slot filling labels. We take several metrics over all datasets to evaluate the performance on multiple dimensions. And to evaluate the context-aware memory networks, we also test the HMNs with only context-free memory on the dialog bAbI tasks.

### 4.2 Evaluation Method

To compare with the original datasets baselines, we apply evaluation methods on each datasets the same as datasets' original papers described in 4.1.

- Bilingual Evaluation Understudy (BLEU) (Papineni et al., 2002). BLEU has been widely employed in evaluating sequence generation including machine translation, text summarization, and dialogue systems. BLEU calculates the n-gram precision which is the fraction of n-grams in the candidate text which is present in any of the reference texts.

- F1 Score (F-measure): F1 evaluates the model's ability in terms of precision and recall, which is more comprehensive than just using precision or recall measure. We adopt F1 to evaluate if a model can extract information from a knowledge base precisely.

- Per-response accuracy and Per-dialog accuracy. Per-response and Per-dialog accuracy count the percentage of responses that are correct. Any incorrect words will make a response or a dialogue negative. Accuracy shows if the model is able to learn the distribution of reproducing factual details.

Table 2: Results on Key-Value Retrieval dataset. F1 score, including Entity F1, is micro-average over the entire set, and three subtasks. Human results are reported by Eric et al.(Eric and Manning, 2017)

| Model name | BLEU | Ent. F1 | Scheduling Ent. F1 | Weather Ent. F1 | Navigation Ent. F1 |
|---|---|---|---|---|---|
| Human* | 13.5 | 60.7 | 64.3 | 61.6 | 55.2 |
| SEQ2SEQ | 11.07 | 30.5 | 30.7 | **46.4** | 13.4 |
| SEQ2SEQ+Attn. | 11.19 | 35.6 | 40.5 | 44.0 | 23.0 |
| Mem2Seq | 12.06 | 31.1 | 51.8 | 34.3 | 12.3 |
| HMNs | **14.46** | **43.1** | **61.3** | 40.3 | **32.3** |

Table 3: Results of Per-response accuracy and Per-dialog accuracy (in brackets) on bAbI dialogues. Per-dialog accuracy presents the accuracy of complete dialogues.

| Task | SEQ2SEQ | SEQ2SEQ+Attn. | Mem2Seq | HMNs-CFO | HMNs |
|---|---|---|---|---|---|
| T3 | 74.8(0) | 74.8(0) | 83.9(15.6) | 93.7(55.9) | **93.6(56.1)** |
| T4 | 56.5(0) | 56.5(0) | 97.0(90.5) | 96.8(89.3) | **100(100)** |
| T5 | 98.9(82.9) | **98.6(83)** | 96.2(46.4) | 97.1(58.2) | 98.0(69.0) |
| T3-OOV | 74.9(0) | 74.0(0) | 83.6(18.1) | 92.3(45.2) | **92.5(48.2)** |
| T4-OOV | 56.5(0) | 57.0(0) | 97.0(89.4) | 96.1(90.3) | **100(100)** |
| T5-OOV | 67.2(0) | 67.6(0) | 71.4(0) | 78.3(0) | **84.1(2.6)** |

Table 4: The results on the DSTC 2

| Model name | F1 | BLEU |
|---|---|---|
| SEQ2SEQ | 69.7 | 55.0 |
| SEQ2SEQ+Attn. | 67.1 | **56.6** |
| SEQ2SEQ+Copy | 71.6 | 55.4 |
| Mem2Seq | 75.3 | 55.3 |
| Our model | **77.7** | 56.4 |

### 4.3 Baselines and Training Setup

The hyper-parameter settings are adopted as the best practice settings for each training set following the Madotto's (Madotto et al., 2018) and Manning's (Manning and Eric, 2017) best experimental results on baselines SEQ2SEQ and Mem2Seq. Detailed models and their settings are as follows:

- Sequence to sequence. For SEQ2SEQ, we adopt one layer LSTMs as encoder and decoder. For Key-Value Retrieval dataset, hidden size is placed at 512 and the dropout rate is 0.3. On dataset bAbI, the hidden size and dropout rate are 128 and 0.1 for task 3, 256 and 0.1 for task 4 and 5. Learning rates are set to 0.001 for bAbI and 0.0001 for DSTC 2 and Key-Value Retrieval dataset.

- SEQ2SEQ + Attention. We adopt the attention mechanism (Luong et al., 2015a) commonly used in neural machine translation. On dataset bAbI, hidden size and the dropout rate are 256 and 0.1 for task 3 and 4, 128 and 0.1 for task 5. For Key-Value Retrieval dataset, hidden size and dropout rate are 512 and 0.3. On the DSTC 2 task, hidden size is set to 353 and word embedding size is 300 (same with original work).

- Mem2Seq. Except 128 in task 3, hidden size in other tasks is 256. The dropout rate is set to 0.2 in task 3, 4 and Key-Value Retrieval dataset, 0.1 in task 5 and DSTC 2 dataset. We adopt three hops in DSTC 2 and Key-Value Retrieval dataset.

- HMNs with context-free only (HMNs-CFO). To test the performance of context-aware memory, we apply other context-free to encode dialogue history instead of context-aware memory in HMNs. All the other structure and parameter settings are the same as HMNs in this model.

All models are tested with various hyper-parameter settings to get their best performance, e.g. hidden size selected from [64, 128, 256, 512]. Note that settings from datasets are also tested like SEQ2SEQ + Attention's hidden size is 353 on Key-Value Retrieval dataset.

During the training, all experiments employ the teacher-forcing scheme, feeding the gold target of last time or highest probability word into decoder with probability 50%. We also randomly mask input with UNK according to the dropout rate.

Table 5: A generated example from Key-Value Retrieval dataset with correct knowledge entities in **bold**. Given the knowledge base and user's request, we list the generated responses of three models and the gold answers. This example is randomly selected from all generated sentences and we only show tuples been used by models.

| Dataset | |
|---|---|
| Knowledge Base | ...<br>valero poi_type gas_station<br>valero distance 2_miles<br>valero address 200_alester_ave<br>1_miles moderate_traffic parking_garage poi palo_alto_garage_r<br>... |
| User | address to the **gas_station** |
| Gold | **valero** is located at **200_alester_ave** |
| Generated Sentence | |
| SEQ2SEQ+Attn. | the closest **gas_station** is located at **200_alester_ave** 7_miles away would you like directions there |
| Mem2Seq | there is a **valero** 1_miles away |
| HMNs | there is a **gas_station** located **2_miles** away at **200_alester_ave** |

### 4.4 HMNs Training Settings

We test the hidden size in [64, 128, 256] and set dropout rate in [0.1, 0.2]. Learning rate is initiated with 0.001 and training batch is set to 64. The metrics results are coming from the best result settings for each dataset. We select hidden size and dropout rate at (256, 0.1) on bAbI task 3 and task 5, (256,0.2) on task 4. On the DSTC 2 task, we set hidden size and dropout rate at (128, 0.1). For Key-Value Retrieval dataset, the setting is hidden size 256 and dropout rate 0.1. Except for bAbI tasks' 1 layer, all HMNs and Mem2Seq tasks employ 3 layer memories.

## 5 Results and Discussion

### 5.1 Results and Analysis

The best results of the baselines and HMNs are gathered into tables and figures. Table 2 show the result of models on Key-Value Retrieval dataset. Except for F1 scores on Calendar Scheduling, HMNs get significantly better results on all benchmarks comparing the state-of-the-art models. HMMs' BLEU score is even higher than human results which are reported in (Eric and Manning, 2017). Results show our model's outstanding performance in generating a fluent and accurate response in most tasks.

Examples generated by our approach and baselines are given in Table 5. These two examples are randomly selected from all generated sentences. Comparing the generated sentences by humans, although entities and sentences are different with gold answer in example one, our approach is able to produce more fluent and accurate sentences. However, the result on task weather forecasting neither HMNs and Mem2Seq can outperform SEQ2SEQ. We will discuss it in the next section.

Table 4 shows our model gets the best F1 score on dataset DSTC 2, while SEQ2SEQ with attention gets the best BLEU result.

Table 3 shows results of models on bAbI tasks. HMNs and Mem2Seq adopt one hop attention only and note that all results are the best performance of each model in 100 epochs. HMNs achieved the best results on most tasks except T5. HMNs-CFO also outperforms the other models. This demonstrates that both training multiple distributions over heterogeneous information and employment of context-aware memory benefit the end-to-end dialogue system. The improvements in per-dialogue accuracy on out-of-vocabulary tests are even more significant. Figure 4 shows the changes of HMNs and HMNs-CFO's total loss across time. HMNs learns significantly faster.

Though automatic metrics cannot really examine human beings' diversified expression, existing dialogue systems aim at generating sentence by learning the patterns of training data, so we believe BLEU is still a metric of great concern in comparing the similar models' ability in learning the sentence patterns. Though human results show end-to-end machines have still a long way to go (60.7 to 43.1). Compared to other models, HMNs sig-

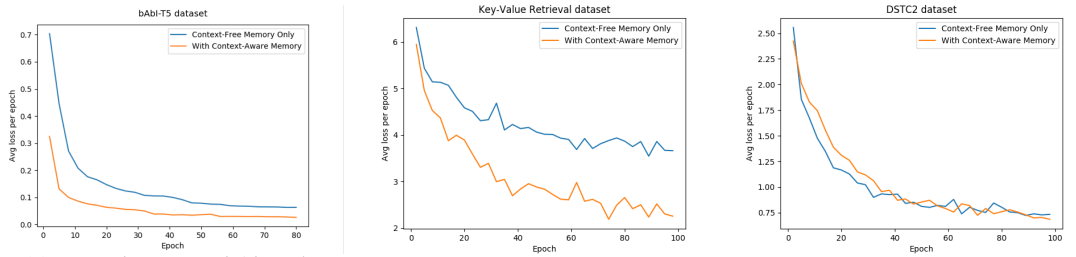


(a) Loss changes on bAbI T5

(b) Loss by epochs on Key-Value Retrieval dataset

(c) Loss by epochs on the DSTC2 dataset

Figure 4: The total loss by epochs on three different datasets in 100 epochs. One line with context-aware memory and the other with only context-free memory

nificantly improves performance in retrieving correct knowledge entities.

## 5.2 Discussion

### 5.2.1 Context-Aware Memory

To show whether context-aware memory benefits conversation learning, on bAbI tasks, we also tested HMNs-CFO memory only. From Table 3, we observe that HMNs-CFO is significantly better than original Mem2Seq as well as SEQ2SEQ + attention in several results and only loses slightly on task 4 (89.3 to 90.5). One reason is that one memory is difficult to learn best distribution over different sources. Respectively encoding sequential dialogue history and grounding knowledge can learn two better distributions than one general but not best distribution. This also indicates that using the query vector generated by history memory to retrieve information in knowledge base memory is reasonable.

As the HMNs model get the best results in all tasks except one, in addition the results of training speed of HMNs and HMNs-CFO (Figure 4), the context-aware memory is clearly to learn representation of the dialogue history much better and faster and also demonstrates that the importance of incorporating context information for dialogue systems. HMNs outperform the HMNs-CFO not only on BLEUs but also entity F1 on most tasks, showing building a good representation of dialogue history benefits knowledge reasoning, and help to improve the context-free memory by issuing a good query vector.

From above all, we can conclude that both stacked memory networks architecture and using context-aware memory to load sequential information can improve the performance of retrieving knowledge and generating sentences.

### 5.2.2 Shortcomings

From the results in Table 2, we note that HMNs and Mem2Seq failed on weather forecasting task. We analysed the average knowledge pairs of weather forecasting tasks and find it near three times the knowledge pairs of the other two tasks. Then we carried out another experiment that first narrows the KB candidates by performing a matching preprocessing operation, and the Weather Ent. F1 result of our method will climb to more than 48 which is the best. This may indicates that this kind of memory networks may have difficulties in handling large scale knowledge base. So perform a matching operation to narrow the candidate knowledge space is critical in a real-world large scale knowledge base.

And in this paper, we only show sequential data and knowledge triples data. For more types of information to integrate, model needs to add other memory networks, e.g. graph neural networks augmented memory networks (Zhou et al., 2018) for graph structured data.

## 6 Conclusion

In this paper, we propose a model that is able to incorporate heterogeneous information in an end-to-end dialogue system. The model applies Heterogeneous Memory Networks (HMNs) to model sequential history and structured database. Results on several datasets show model can significantly improve the performance of generating the response. Our proposed context-aware memory networks show outstanding performance in learning the distribution over dialogue history and retrieving knowledge. We present the possibility of efficiently using various structured data in end-to-end task-oriented dialogue without any extra labeling and module training.

## Acknowledgement

We thank the anonymous reviewers for their insightful comments on this paper. This work was supported by the NSFC (No.61402403), DAMO Academy (Alibaba Group), Alibaba-Zhejiang University Joint Institute of Frontier Technologies, Chinese Knowledge Center for Engineering Sciences and Technology, and the Fundamental Research Funds for the Central Universities.

## References

Antoine Bordes and Jason Weston. 2016. Learning end-to-end goal-oriented dialog. *CoRR*, abs/1605.07683.

Hongshen Chen, Zhaochun Ren, Jiliang Tang, Yihong Eric Zhao, and Dawei Yin. 2018. Hierarchical variational memory network for dialogue generation. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, pages 1653–1662.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of SSST@EMNLP 2014, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, Doha, Qatar, 25 October 2014*, pages 103–111.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734.

Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2017. Towards end-to-end reinforcement learning of dialogue agents for information access. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 484–495, Vancouver, Canada. Association for Computational Linguistics.

Mihail Eric and Christopher D Manning. 2017. Key-value retrieval networks for task-oriented dialogue. *arXiv preprint arXiv:1705.05414*.

Yao Fu and Yansong Feng. 2018. Natural answer generation with heterogeneous memory. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 185–195.

Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen-tau Yih, and Michel Galley. 2018. A knowledge-grounded neural conversation model. pages 5110–5117.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. 2016. Tracking the world state with recurrent entity networks. *CoRR*, abs/1612.03969.

Xuijun Li, Yun-Nung Chen, Lihong Li, Jianfeng Gao, and Asli Celikyilmaz. 2017. End-to-end task-completion neural dialogue systems. In *Proceedings of The 8th International Joint Conference on Natural Language Processing*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015a. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1412–1421.

Thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, and Wojciech Zaremba. 2015b. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 11–19.

Andrea Madotto, Chien-Sheng Wu, and Pascale Fung. 2018. Mem2seq: Effectively incorporating knowledge bases into end-to-end task-oriented dialog systems. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 1468–1478.

Christopher D. Manning and Mihail Eric. 2017. A copy-augmented sequence-to-sequence architecture gives good performance on task-oriented dialogue. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers*, pages 468–473.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA.*, pages 311–318.

Alan Ritter, Colin Cherry, and William B. Dolan. 2011. Data-driven response generation in social media. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 583–593.

Lina Maria Rojas-Barahona, Milica Gasic, Nikola Mrksic, Pei-Hao Su, Stefan Ultes, Tsung-Hsien Wen, Steve J. Young, and David Vandyke. 2017. A network-based end-to-end trainable task-oriented dialogue system. pages 438–449.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1073–1083.

Hendra Setiawan, Haizhou Li, Min Zhang, and Beng Chin Ooi. 2005. Phrase-based statistical machine translation: A level of detail approach. In *Natural Language Processing - IJCNLP 2005, Second International Joint Conference, Jeju Island, Korea, October 11-13, 2005, Proceedings*, pages 576–587.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1577–1586.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 196–205.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2440–2448.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Chuanqi Tan, Furu Wei, Nan Yang, Bowen Du, Weifeng Lv, and Ming Zhou. 2018. S-net: From answer extraction to answer synthesis for machine reading comprehension. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2692–2700.

Oriol Vinyals and Quoc V. Le. 2015. A neural conversational model. *CoRR*, abs/1506.05869.

Zhao Yan, Nan Duan, Peng Chen, Ming Zhou, Jianshe Zhou, and Zhoujun Li. 2017. Building task-oriented dialogue systems for online shopping. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 4618–4626.

Tom Young, Erik Cambria, Iti Chaturvedi, Hao Zhou, Subham Biswas, and Minlie Huang. 2018. Augmenting end-to-end dialogue systems with commonsense knowledge. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Hao Zhou, Tom Young, Minlie Huang, Haizhou Zhao, Jingfang Xu, and Xiaoyan Zhu. 2018. Commonsense knowledge aware conversation generation with graph attention. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden.*, pages 4623–4629.

Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. 2017. Neural question generation from text: A preliminary study. In *Natural Language Processing and Chinese Computing - 6th CCF International Conference, NLPCC 2017, Dalian, China, November 8-12, 2017, Proceedings*, pages 662–671.

# Aspect-based Sentiment Classification with Aspect-specific Graph Convolutional Networks

**Chen Zhang**
Beijing Institute of Technology
Beijing, China
gene@bit.edu.cn

**Qiuchi Li**
University of Padua
Padua, Italy
qiuchili@dei.unipd.it

**Dawei Song***
Beijing Institute of Technology
Beijing, China
dwsong@bit.edu.cn

## Abstract

Due to their inherent capability in semantic alignment of aspects and their context words, attention mechanism and Convolutional Neural Networks (CNNs) are widely applied for aspect-based sentiment classification. However, these models lack a mechanism to account for relevant syntactical constraints and long-range word dependencies, and hence may mistakenly recognize syntactically irrelevant contextual words as clues for judging aspect sentiment. To tackle this problem, we propose to build a Graph Convolutional Network (GCN) over the dependency tree of a sentence to exploit syntactical information and word dependencies. Based on it, a novel aspect-specific sentiment classification framework is raised. Experiments on three benchmarking collections illustrate that our proposed model has comparable effectiveness to a range of state-of-the-art models[1], and further demonstrate that both syntactical information and long-range word dependencies are properly captured by the graph convolution structure.

## 1 Introduction

Aspect-based (also known as aspect-level) sentiment classification aims at identifying the sentiment polarities of aspects explicitly given in sentences. For example, in a comment about a laptop saying "*From the speed to the multi-touch gestures this operating system beats Windows easily.*", the sentiment polarities for two aspects *operating system* and *Windows* are *positive* and *negative*, respectively. Generally, this task is formulated as predicting the polarity of a provided (sentence, aspect) pair.

Given the inefficiency of manual feature refinement (Jiang et al., 2011), early works of aspect-based sentiment classification are mainly based on neural network methods (Dong et al., 2014; Vo and Zhang, 2015). Ever since Tang et al. (2016a) pointed out the challenge of modelling semantic relatedness between context words and aspects, attention mechanism coupled with Recurrent Neural Networks (RNNs) (Bahdanau et al., 2014; Luong et al., 2015; Xu et al., 2015) starts to play a critical role in more recent models (Wang et al., 2016; Tang et al., 2016b; Yang et al., 2017; Liu and Zhang, 2017; Ma et al., 2017; Huang et al., 2018).

While attention-based models are promising, they are insufficient to capture syntactical dependencies between context words and the aspect within a sentence. Consequently, **the current attention mechanism may lead to a given aspect mistakenly attending to syntactically unrelated context words as descriptors** (Limitation 1). Look at a concrete example "*Its size is ideal and the weight is acceptable.*". Attention-based models often identify *acceptable* as a descriptor of the aspect *size*, which is in fact not the case. In order to address the issue, He et al. (2018) imposed some syntactical constraints on attention weights, but the effect of syntactical structure was not fully exploited.

In addition to the attention-based models, Convolutional Neural Networks (CNNs) (Xue and Li, 2018; Li et al., 2018) have been employed to discover descriptive multi-word phrases for an aspect, based on the finding (Fan et al., 2018) that the sentiment of an aspect is usually determined by key phrases instead of individual words. Nevertheless, the CNN-based models can only perceive multi-word features as consecutive words with the convolution operations over word sequences, but are **inadequate to determine sentiments depicted by multiple words that are not next to each other** (Limitation 2). In the sentence "*The*

*Corresponding author.

[1]Code and preprocessed datasets are available at https://github.com/GeneZC/ASGCN.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4568–4578, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

*staff should be a bit more friendly*" with *staff* as the aspect, a CNN-based model may make an incorrect prediction by detecting *more friendly* as the descriptive phrase, disregarding the impact of *should be* which is two words away but reverses the sentiment.

In this paper, we aim to tackle the two limitations identified above by using Graph Convolutional Networks (GCNs) (Kipf and Welling, 2017). GCN has a multi-layer architecture, with each layer encoding and updating the representation of nodes in the graph using features of immediate neighbors. Through referring to syntactical dependency trees, a GCN is potentially capable of drawing syntactically relevant words to the target aspect, and exploiting long-range multi-word relations and syntactical information with GCN layers. GCNs have been deployed on document-word relationships (Yao et al., 2018) and tree structures (Marcheggiani and Titov, 2017; Zhang et al., 2018), but how they can be effectively used in aspect-based sentiment classification is yet to be explored.

To fill the gap, this paper proposes an Aspect-specific Graph Convolutional Network (ASGCN), which, to the best of our knowledge, is the first GCN-based model for aspect-based sentiment classification. ASGCN starts with a bidirectional Long Short-Term Memory network (LSTM) layer to capture contextual information regarding word orders. In order to obtain aspect-specific features, a multi-layered graph convolution structure is implemented on top of the LSTM output, followed by a masking mechanism that filters out non-aspect words and keeps solely high-level aspect-specific features. The aspect-specific features are fed back to the LSTM output for retrieving informative features with respect to the aspect, which are then used to predict aspect-based sentiment.

Experiments on three benchmarking datasets show that ASGCN effectively addresses both limitations of the current aspect-based sentiment classification approaches, and outperforms a range of state-of-the-art models.

Our contributions are as follows:

- We propose to exploit syntactical dependency structures within a sentence and resolve the long-range multi-word dependency issue for aspect-based sentiment classification.
- We posit that Graph Convolutional Network (GCN) is suitable for our purpose, and propose a novel Aspect-specific GCN model. To our best knowledge, this is the first investigation in this direction.
- Extensive experiment results verify the importance of leveraging syntactical information and long-range word dependencies, and demonstrate the effectiveness of our model in capturing and exploiting them in aspect-based sentiment classification.

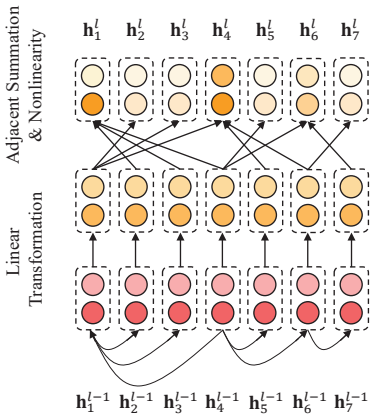


Figure 1: An example of GCN layer.

## 2 Graph Convolutional Networks

GCNs can be considered as an adaptation of the conventional CNNs for encoding local information of unstructured data. For a given graph with $k$ nodes, an adjacency matrix[2] $\mathbf{A} \in \mathbb{R}^{k \times k}$ is obtained through enumerating the graph. For convenience, we denote the output of the $l$-th layer for node $i$ as $\mathbf{h}_i^l$, where $\mathbf{h}_i^0$ represents the initial state of node $i$. For an $L$-layer GCN, $l \in [1, 2, \cdots, L]$ and $\mathbf{h}_i^L$ is the final state of node $i$. The graph convolution operated on the node representation can be written as:

$$\mathbf{h}_i^l = \sigma(\sum_{j=1}^{k} \mathbf{A}_{ij} \mathbf{W}^l \mathbf{h}_j^{l-1} + \mathbf{b}^l) \quad (1)$$

where $\mathbf{W}^l$ is a linear transformation weight, $\mathbf{b}^l$ is a bias term, and $\sigma$ is a nonlinear function, e.g. ReLU. For a better illustration, an example of GCN layer is shown in Figure 1.

As the graph convolution process only encodes information of immediate neighbors, a node in the graph can only be influenced by the neighbouring nodes within $L$ steps in an $L$-layer GCN. In

[2] $\mathbf{A}_{ij}$ indicates whether the $i$-th token is adjacent to the $j$-th token or not.

this way, the graph convolution over the dependency tree of a sentence provides syntactical constraints for an aspect within the sentence to identify descriptive words based on syntactical distances. Moreover, GCN is able to deal with the circumstances where the polarity of an aspect is described by non-consecutive words, as GCN over dependency tree will gather the non-consecutive words into a smaller scope and aggregate their features properly with graph convolution. Therefore, we are inspired to adopt GCN to leverage syntactical information and long-range word dependencies for aspect-based sentiment classification.

## 3 Aspect-specific Graph Convolutional Network

Figure 2 gives an overview of ASGCN. The components of ASGCN will be introduced separately in the rest of the section.

### 3.1 Embedding and Bidirectional LSTM

Given a $n$-word sentence $c = \{w_1^c, w_2^c, \cdots, w_{\tau+1}^c, \cdots, w_{\tau+m}^c, \cdots, w_{n-1}^c, w_n^c\}$ containing a corresponding $m$-word aspect starting from the $(\tau+1)$-th token, we embed each word token into a low-dimensional real-valued vector space (Bengio et al., 2003) with embedding matrix $\mathbf{E} \in \mathbb{R}^{|V| \times d_e}$, where $|V|$ is the size of vocabulary and $d_e$ denotes the dimensionality of word embeddings. With the word embeddings of the sentence, a bidirectional LSTM is constructed to produce hidden state vectors $\mathbf{H}^c = \{\mathbf{h}_1^c, \mathbf{h}_2^c, \cdots, \mathbf{h}_{\tau+1}^c, \cdots, \mathbf{h}_{\tau+m}^c, \cdots, \mathbf{h}_{n-1}^c, \mathbf{h}_n^c\}$, where $\mathbf{h}_t^c \in \mathbb{R}^{2d_h}$ represents the hidden state vector at time step $t$ from the bidirectional LSTM, and $d_h$ is the dimensionality of a hidden state vector output by an unidirectional LSTM.

### 3.2 Obtaining Aspect-oriented Features

Different from general sentiment classification, aspect-based sentiment classification targets at judging sentiments from the view of aspects, and thus calls for an aspect-oriented feature extraction strategy. In this study, we obtain aspect-oriented features by applying multi-layer graph convolution over the syntactical dependency tree of a sentence, and imposing an aspect-specific masking layer on its top.

#### 3.2.1 Graph Convolution over Dependency Trees

Aiming to address the limitations of existing approaches (as discussed in previous sections), we leverage a graph convolutional network over dependency trees of sentences. Specifically, after the dependency tree[3] of the given sentence is constructed, we first attain an adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ according to the words in the sentence. It is important to note that dependency trees are directed graphs. While GCNs generally do not consider directions, they could be adapted to the direction-aware scenario. Accordingly, we propose two variants of ASGCN, i.e. ASGCN-DG on dependency graphs which are un-directional, and ASGCN-DT concerning dependency trees which are directional. Practically, the only difference between ASGCN-DG and ASGCN-DT lies in their adjacency matrices: The adjacency matrix of ASGCN-DT is much more sparse than that of ASGCN-DG. Such setting is in accordance with the phenomenon that parents nodes are broadly influenced by their children nodes. Furthermore, following the idea of self-looping in Kipf and Welling (2017), each word is manually set adjacent to itself, i.e. the diagonal values of $\mathbf{A}$ are all ones.

The ASGCN variants are performed in a multi-layer fashion, on top of the bidirectional LSTM output in Section 3.1, i.e. $\mathbf{H}^0 = \mathbf{H}^c$ to make nodes aware of context (Zhang et al., 2018). Then the representation of each node is updated with graph convolution operation with normalization factor (Kipf and Welling, 2017) as below:

$$\tilde{\mathbf{h}}_i^l = \sum_{j=1}^{n} \mathbf{A}_{ij} \mathbf{W}^l \mathbf{g}_j^{l-1} \tag{2}$$

$$\mathbf{h}_i^l = \mathrm{ReLU}(\tilde{\mathbf{h}}_i^l / (d_i + 1) + \mathbf{b}^l) \tag{3}$$

where $\mathbf{g}_j^{l-1} \in \mathbb{R}^{2d_h}$ is the $j$-th token's representation evolved from the preceding GCN layer while $\mathbf{h}_i^l \in \mathbb{R}^{2d_h}$ is the product of current GCN layer, and $d_i = \sum_{j=1}^{n} \mathbf{A}_{ij}$ is degree of the $i$-th token in the tree. The weights $\mathbf{W}^l$ and bias $\mathbf{b}^l$ are trainable parameters.

It is worth noting that we do not have $\mathbf{h}_i^l$ immediately fed into successive GCN layer, but conduct a position-aware transformation in the first place:

$$\mathbf{g}_i^l = \mathcal{F}(\mathbf{h}_i^l) \tag{4}$$

[3]We use spaCy toolkit: https://spacy.io/.

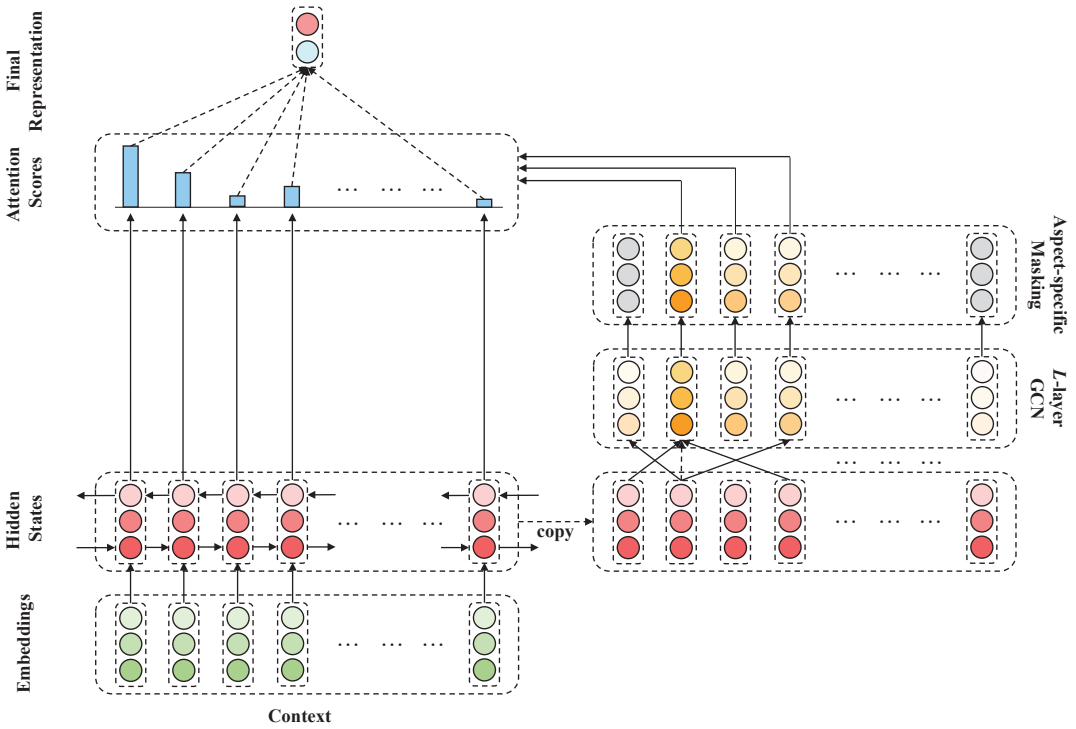


Figure 2: Overview of aspect-specific graph convolutional network.

where $\mathcal{F}(\cdot)$ is a function assigning position weights, widely adopted by previous works (Li et al., 2018; Tang et al., 2016b; Chen et al., 2017), for augmenting the importance of context words close to the aspect. By doing so we aim at reducing the noise and bias that may have naturally arisen from the dependency parsing process. Specifically, the function $\mathcal{F}(\cdot)$ is:

$$q_i = \begin{cases} 1 - \frac{\tau+1-i}{n} & 1 \leq i < \tau+1 \\ 0 & \tau+1 \leq i \leq \tau+m \\ 1 - \frac{i-\tau-m}{n} & \tau+m < i \leq n \end{cases} \quad (5)$$

$$\mathcal{F}(\mathbf{h}_i^l) = q_i \mathbf{h}_i^l \quad (6)$$

where $q_i \in \mathbb{R}$ is the position weight to $i$-th token. The final outcome of the $L$-layer GCN is $\mathbf{H}^L = \{\mathbf{h}_1^L, \mathbf{h}_2^L, \cdots, \mathbf{h}_{\tau+1}^L, \cdots, \mathbf{h}_{\tau+m}^L, \cdots, \mathbf{h}_{n-1}^L, \mathbf{h}_n^L\}$, $\mathbf{h}_t^L \in \mathbb{R}^{2d_h}$.

#### 3.2.2 Aspect-specific Masking

In this layer, we mask out hidden state vectors of non-aspect words and keep the aspect word states unchanged:

$$\mathbf{h}_t^L = \mathbf{0} \quad 1 \leq t < \tau+1, \tau+m < t \leq n \quad (7)$$

The outputs of this zero-masking layer are the aspect-oriented features $\mathbf{H}_{\text{mask}}^L = \{\mathbf{0}, \cdots, \mathbf{h}_{\tau+1}^L, \cdots, \mathbf{h}_{\tau+m}^L, \cdots, \mathbf{0}\}$. Through graph convolution, these features $\mathbf{H}_{\text{mask}}^L$ have perceived contexts around the aspect in such a way that considers both syntactical dependencies and long-range multi-word relations.

### 3.3 Aspect-aware Attention

Based on the aspect-oriented features, a refined representation of the hidden state vectors $\mathbf{H}^c$ is produced via a novel retrieval-based attention mechanism. The idea is to retrieve significant features that are semantically relevant to the aspect words from the hidden state vectors, and accordingly set a retrieval-based attention weight for each context word. In our implementation, the attention weights are computed as below:

$$\beta_t = \sum_{i=1}^{n} \mathbf{h}_t^{c\top} \mathbf{h}_i^L = \sum_{i=\tau+1}^{\tau+m} \mathbf{h}_t^{c\top} \mathbf{h}_i^L \quad (8)$$

$$\alpha_t = \frac{\exp(\beta_t)}{\sum_{i=1}^{n} \exp(\beta_i)} \quad (9)$$

Here, the dot product is used to measure the semantic relatedness between aspect component words and words in the sentence so that aspect-specific masking, i.e. zero masking, could take effect as shown in Equation 8. The final representation for prediction is therefore formulated as:

$$\mathbf{r} = \sum_{t=1}^{n} \alpha_t \mathbf{h}_t^c \tag{10}$$

### 3.4 Sentiment Classification

Having obtained the representation $\mathbf{r}$, it is then fed into a fully-connected layer, followed by a softmax normalization layer to yield a probability distribution $\mathbf{p} \in \mathbb{R}^{d_p}$ over polarity decision space:

$$\mathbf{p} = \text{softmax}(\mathbf{W}_p \mathbf{r} + \mathbf{b}_p) \tag{11}$$

where $d_p$ is the same as the dimensionality of sentiment labels while $\mathbf{W}_p \in \mathbb{R}^{d_p \times 2d_h}$ and $\mathbf{b}_p \in \mathbb{R}^{d_p}$ are the learned weight and bias, respectively.

### 3.5 Training

This model is trained by the standard gradient descent algorithm with the cross-entropy loss and $L_2$-regularization:

$$\text{Loss} = -\sum_{(c,\hat{p}) \in C} \log \mathbf{p}_{\hat{p}} + \lambda \|\Theta\|_2 \tag{12}$$

where $C$ denotes the collection of data sets, $\hat{p}$ is the label and $\mathbf{p}_{\hat{p}}$ means the $\hat{p}$-th element of $\mathbf{p}$, $\Theta$ represents all trainable parameters, and $\lambda$ is the coefficient of $L_2$-regularization.

## 4 Experiments

### 4.1 Datasets and Experimental Settings

Our experiments are conducted on five datasets: one (TWITTER) is originally built by Dong et al. (2014) containing twitter posts, while the other four (LAP14, REST14, REST15, REST16) are respectively from SemEval 2014 task 4 (Pontiki et al., 2014), SemEval 2015 task 12 (Pontiki et al., 2015) and SemEval 2016 task 5 (Pontiki et al., 2016), consisting of data from two categories, i.e. laptop and restaurant. Following previous work (Tang et al., 2016b), we remove samples with conflicting[4] polarities or without explicit aspects in the sentences in REST15 and REST16. The statistics of datasets are reported in Table 1.

For all our experiments, 300-dimensional pretrained GloVe vectors (Pennington et al., 2014) are used to initialize word embeddings. All model weights are initialized with uniform distribution. The dimensionality of hidden state vectors is set to 300. We use Adam as the optimizer with a learning rate of 0.001. The coefficient of $L_2$-regularization is $10^5$ and batch size is 32. Moreover, the number of GCN layers is set to 2, which is the best-performing depth in pilot studies.

[4] An opinion target is associated with different sentiment polarities.

| Dataset | | # Pos. | # Neu. | # Neg. |
|---|---|---|---|---|
| TWITTER | Train | 1561 | 3127 | 1560 |
| | Test | 173 | 346 | 173 |
| LAP14 | Train | 994 | 464 | 870 |
| | Test | 341 | 169 | 128 |
| REST14 | Train | 2164 | 637 | 807 |
| | Test | 728 | 196 | 196 |
| REST15 | Train | 912 | 36 | 256 |
| | Test | 326 | 34 | 182 |
| REST16 | Train | 1240 | 69 | 439 |
| | Test | 469 | 30 | 117 |

Table 1: Dataset statistics.

The experimental results are obtained by averaging 3 runs with random initialization, where Accuracy and Macro-Averaged F1 are adopted as the evaluation metrics. We also carry out paired t-test on both Accuracy and Macro-Averaged F1 to verify whether the improvements achieved by our models over the baselines are significant.

### 4.2 Models for Comparison

In order to comprehensively evaluate the two variants of our model, namely, ASGCN-DG and ASGCN-DT, we compare them with a range of baselines and state-of-the-art models, as listed below:

- SVM (Kiritchenko et al., 2014) is the model which has won SemEval 2014 task 4 with conventional feature extraction methods.
- LSTM (Tang et al., 2016a) uses the last hidden state vector of LSTM to predict sentiment polarity.
- MemNet (Tang et al., 2016b) considers contexts as external memories and benefits from a multi-hop architecture.
- AOA (Huang et al., 2018) borrows the idea of attention-over-attention from the field of machine translation.
- IAN (Ma et al., 2017) interactively models the relationships between aspects and their contexts.

| Model | TWITTER | | LAP14 | | REST14 | | REST15 | | REST16 | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Acc. | F1 | Acc. | F1 | Acc. | F1 | Acc. | F1 | Acc. | F1 |
| SVM | 63.40$^\sharp$ | 63.30$^\sharp$ | 70.49$^\natural$ | N/A | 80.16$^\natural$ | N/A | N/A | N/A | N/A | N/A |
| LSTM | 69.56 | 67.70 | 69.28 | 63.09 | 78.13 | 67.47 | 77.37 | 55.17 | 86.80 | 63.88 |
| MemNet | 71.48 | 69.90 | 70.64 | 65.17 | 79.61 | 69.64 | 77.31 | 58.28 | 85.44 | 65.99 |
| AOA | 72.30 | 70.20 | 72.62 | 67.52 | 79.97 | 70.42 | 78.17 | 57.02 | 87.50 | 66.21 |
| IAN | **72.50** | **70.81** | 72.05 | 67.38 | 79.26 | 70.09 | 78.54 | 52.65 | 84.74 | 55.21 |
| TNet-LF | **72.98** | **71.43** | **74.61** | **70.14** | 80.42 | 71.03 | 78.47 | 59.47 | **89.07** | **70.43** |
| ASCNN | 71.05 | 69.45 | 72.62 | 66.72 | **81.73** | **73.10** | 78.47 | 58.90 | 87.39 | 64.56 |
| ASGCN-DT | 71.53 | 69.68 | 74.14$^\dagger$ | 69.24$^\dagger$ | **80.86**$^\ddagger$ | **72.19**$^\ddagger$ | **79.34**$^{\dagger\ddagger}$ | **60.78**$^{\dagger\ddagger}$ | 88.69$^\dagger$ | 66.64$^\dagger$ |
| ASGCN-DG | 72.15$^\dagger$ | 70.40$^\dagger$ | **75.55**$^{\dagger\ddagger}$ | **71.05**$^{\dagger\ddagger}$ | 80.77$^\ddagger$ | 72.02$^\ddagger$ | **79.89**$^{\dagger\ddagger}$ | **61.89**$^{\dagger\ddagger}$ | **88.99**$^\dagger$ | **67.48**$^\dagger$ |

Table 2: Model comparison results (%). Average accuracy and macro-F1 score over 3 runs with random initialization. The best two results with each dataset are in bold. The results with $\natural$ are retrieved from the original papers and the results with $\sharp$ are retrieved from Dong et al. (2014). The marker $\dagger$ refers $p < 0.05$ by comparing with ASCNN in paired t-test and the marker $\ddagger$ refers $p < 0.05$ by comparing with TNet-LF in paired t-test.

- TNet-LF (Li et al., 2018) puts forward Context-Preserving Transformation (CPT) to preserve and strengthen the informative part of contexts.

In order to examine to what degrees GCN would outperform CNN, we also involve a model named ASCNN in the experiment, which replaces 2-layer GCN with 2-layer CNN in ASGCN[5].

### 4.3 Results

As is shown in Table 2, ASGCN-DG consistently outperforms all compared models on LAP14 and REST15 datasets, and achieves comparable results on TWITTER and REST16 datasets compared with baseline TNet-LF and on REST14 compared with ASCNN. The results demonstrate the effectiveness of ASGCN-DG and the insufficiency of directly integrating syntax information into attention mechanism as in He et al. (2018). Meanwhile, ASGCN-DG performs better than ASGCN-DT by a large margin on TWITTER, LAP14, Rest15 and REST16 datasets. And ASGCN-DT's result is lower than TNet-LF's on LAP14. A possible reason is that the information from parents nodes is as important as that from children nodes, so treating dependency trees as directed graphs leads to information loss. Additionally, ASGCN-DG outperforms ASCNN on all datasets except REST14, illustrating ASGCN is better at capturing long-range word dependencies, while to some extent ASCNN shows an impact brought by aspect-specific masking. We suspect REST14 dataset is not so sensitive to syntactic information. Moreover, the sentences from TWITTER dataset are less grammatical, restricting the efficacy. We conjecture this is likely the reason why ASGCN-DG and ASGCN-DT get sub-optimal results on TWITTER dataset.

### 4.4 Ablation Study

To further examine the level of benefit that each component of ASGCN brings to the performance, an ablation study is performed on ASGCN-DG. The results are shown in Table 3. We also present the results of BiLSTM+Attn as a baseline, which uses two LSTMs for the aspect and the context respectively.

First, removal of position weights (i.e. ASGCN-DG w/o pos.) leads to performance drops on LAP14, REST15 and REST16 datasets but performance boosts on TWITTER and REST14 datasets. Recall the main results on REST14 dataset, we conclude that the integration of position weights is not helpful to reduce noise of user generated contents if syntax is not crucial for the data. Moreover, after we get rid of aspect-specific masking (i.e. ASGCN-DG w/o masking), the model could not keep as competitive as TNet-LF. This verifies the significance of aspect-specific masking.

Compared with ASGCN-DG, ASGCN-DG w/o GCN (i.e. preserving position weights and aspect-specific masking, but without using GCN layers) is much less powerful on all five datasets except F1 metric on TWITTER dataset. However, ASGCN-DG w/o GCN is still slightly better than BiL-

[5]In order to ensure the length of input and output is consistent, kernel length is set to 3 and padding is 1.

| Model | TWITTER | | LAP14 | | REST14 | | REST15 | | REST16 | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Acc. | F1 | Acc. | F1 | Acc. | F1 | Acc. | F1 | Acc. | F1 |
| BiLSTM+Attn | 71.24 | 69.55 | 72.83 | 67.82 | 79.85 | 70.03 | 78.97 | 58.18 | 87.28 | 68.18 |
| ASGCN-DG | 72.15 | 70.40 | 75.55 | 71.05 | 80.77 | 72.02 | 79.89 | 61.89 | 88.99 | 67.48 |
| ASGCN-DG w/o pos. | 72.69 | 70.59 | 73.93 | 69.63 | 81.22 | 72.94 | 79.58 | 61.55 | 88.04 | 66.63 |
| ASGCN-DG w/o mask | 72.64 | 70.63 | 72.05 | 66.56 | 79.02 | 68.29 | 77.80 | 57.51 | 86.36 | 61.41 |
| ASGCN-DG w/o GCN | 71.92 | 70.63 | 73.51 | 68.83 | 79.40 | 69.43 | 79.40 | 61.18 | 87.55 | 66.19 |

Table 3: Ablation study results (%). Accuracy and macro-F1 scores are the average value over 3 runs with random initialization.

STM+Attn on all datasets except REST14 dataset, due to the strength of the aspect-specific masking mechanism.

Thus it could be concluded that GCN contributes to ASGCN to a considerable extent since GCN captures syntatic word dependencies and long-range word relations at the same time. Nevertheless, the GCN does not work well as expected on the datasets not sensitive to syntax information, as we have seen in TWITTER and REST14 datasets.

### 4.5 Case Study

To better understand how ASGCN works, we present a case study with several testing examples. Particularly, we visualize the attention scores offered by MemNet, IAN, ASCNN and ASGCN-DG in Table 4, along with their predictions on these examples and the corresponding ground truth labels.

The first sample "*great food but the service was dreadful!*" has two aspects within one sentence, which may hinder attention-based models from aligning the aspects with their relevant descriptive words precisely. The second sample sentence "*The staff should be a bit more friendly.*" uses a subjunctive word "*should*", bringing extra difficulty in detecting implicit semantics. The last example contains negation in the sentence, that can easily lead models to make wrong predictions.

MemNet fails in all three presented samples. While IAN is capable of differing modifiers for distinct aspects, it fails to infer sentiment polarities of sentences with special styles. Armed with position weights, ASCNN correctly predicts the label for the second sample as the phrase *should be* is close to the aspect *staff*, but failed for the third one with a longer-range word dependency. Our ASGCN-DG correctly handles all the three samples, implying that GCN effectively integrates syntactic dependency information into an enriched semantic representation. In particular, ASGCN-DG makes correct predictions on the second and the third sample, both having a seemingly biased focus. This shows ASGCN's capability of capturing long-range multi-word features.

## 5 Discussion

### 5.1 Investigation on the Impact of GCN Layers

As ASGCN involves an $L$-layer GCN, we investigate the effect of the layer number $L$ on the final performance of ASGCN-DG. Basically, we vary the value of $L$ in the set {1,2,3,4,6,8,12} and check the corresponding Accuracy and Macro-Averaged F1 of ASGCN-DG on the LAP14 dataset. The results are illustrated in Figure 3.

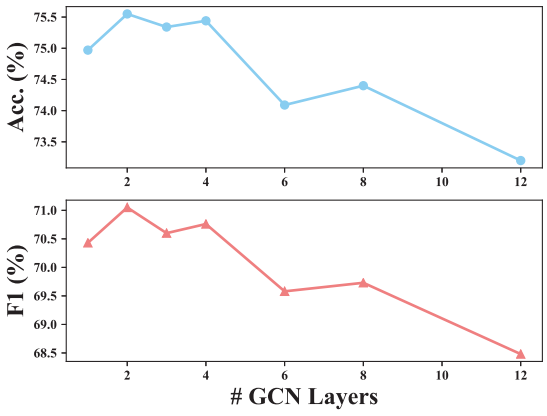


Figure 3: Effect of the number of GCN layers. Accuracy and macro-F1 scores are the average value over 3 runs with random initialization.

On both metrics, ASGCN-DG achieves the best performance when $L$ is 2, which justifies the selection on the number of layers in the experiment sec-

| Model | Aspect | Attention visualization | Prediction | Label |
|---|---|---|---|---|
| MemNet | food | great food but the service was dreadful ! | negative✗ | positive |
| | staff | The staff should be a bit more friendly . | positive✗ | negative |
| | Windows 8 | Did not enjoy the new Windows 8 and touchscreen functions . | positive✗ | negative |
| IAN | food | great food but the service was dreadful ! | positive✓ | positive |
| | staff | The staff should be a bit more friendly . | positive✗ | negative |
| | Windows 8 | Did not enjoy the new Windows 8 and touchscreen functions . | neutral✗ | negative |
| ASCNN | food | great food but the service was dreadful ! | positive✓ | positive |
| | staff | The staff should be a bit more friendly . | negative✓ | negative |
| | Windows 8 | Did not enjoy the new Windows 8 and touchscreen functions . | positive✗ | negative |
| ASGCN-DG | food | great food but the service was dreadful ! | positive✓ | positive |
| | staff | The staff should be a bit more friendly . | negative✓ | negative |
| | Windows 8 | Did not enjoy the new Windows 8 and touchscreen functions . | negative✓ | negative |

Table 4: Case study. Visualization of attention scores from MemNet, IAN, ASCNN and ASGCN-DG on testing examples, along with their predictions and correspondingly, golden labels. The marker ✓ indicates correct prediction while the marker ✗ indicates incorrect prediction.

tion. Moreover, a dropping trend on both metrics is present as $L$ increases. For large $L$, especially when $L$ equals to 12, ASGCN-DG basically becomes more difficult to train due to large amount of parameters.

### 5.2 Investigation on the Effect of Multiple Aspects

In the datasets, there might exist multiple aspect terms in one sentence. Thus, we intend to measure whether such phenomena would affect the effectiveness of ASGCN. We divide the training samples in LAP14 and REST14 datasets into different groups based on the number of aspect terms in the sentences and compute the training accuracy differences between these groups. It is worth noting that the samples with more than 7 aspect terms are removed as outliers because the sizes of these samples are too small for any meaningful comparison.

It can be seen in Figure 4 that when the number of aspects in the sentences is more than 3, the accuracy becomes fluctuated, indicating a low robustness in capturing multiple-aspect correlations and suggesting the need of modelling multi-aspect dependencies in future work.

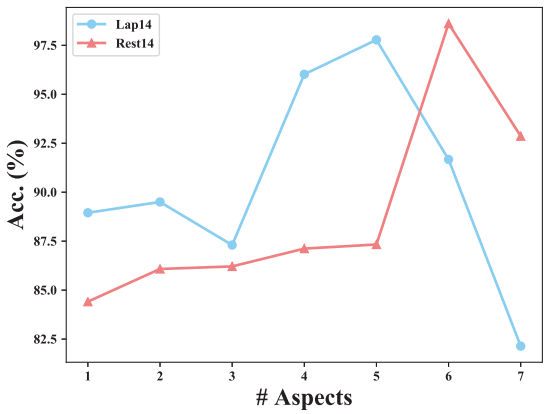


Figure 4: Accuracy versus the number of aspects (# Aspects) in the sentences.

## 6 Related Work

Constructing neural network models over word sequences, such as CNNs (Kim, 2014; Johnson and Zhang, 2015), RNNs (Tang et al., 2016a) and Recurrent Convolutional Neural Networks (RC-

NNs) (Lai et al., 2015), has achieved promising performances in sentiment analysis. However, the importance but lack of an effective mechanism of leveraging dependency trees for capturing distant relations of words has also been recognized. Tai et al. (2015) showed that LSTM with dependency trees or constituency trees outperformed CNNs. Dong et al. (2014) presented an adaptive recursive neural network using dependency trees, which achieved competitive results compared with strong baselines. More recent research showed that general dependency-based models are difficult to achieve comparable results to the attention-based models, as dependency trees are not capable of catching long-term contextualized semantic information properly. Our work overcomes this limitation by adopting Graph convolutional networks (GCNs) (Kipf and Welling, 2017) .

GCN has recently attracted a growing attention in the area of artificial intelligence and has been applied to Natural Language Processing (NLP). Marcheggiani and Titov (2017) claimed that GCN could be considered as a complement to LSTM, and proposed a GCN-based model for semantic role labeling. Vashishth et al. (2018) and Zhang et al. (2018) used graph convolution over dependency trees in document dating and relation classification, respectively. Yao et al. (2018) introduced GCN to text classification utilizing document-word and word-word relations, and gained improvements over various state-of-the-art methods. Our work investigates the effect of dependency trees in depth via graph convolution, and develops aspect-specific GCN model that integrates with the LSTM architecture and attention mechanism for more effective aspect-based sentiment classification.

## 7 Conclusions and Future Work

We have re-examined the challenges encountering existing models for aspect-specific sentiment classification, and pointed out the suitability of graph convolutional network (GCN) for tackling these challenges. Accordingly, we have proposed a novel network to adopt GCN for aspect-based sentiment classification. Experimental results have indicated that GCN brings benefit to the overall performance by leveraging both syntactical information and long-range word dependencies.

This study may be further improved in the following aspects. First, the edge information of the syntactical dependency trees, i.e. the label of each edge, is not exploited in this work. We plan to design a specific graph neural network that takes into consideration the edge labels. Second, domain knowledge can be incorporated. Last but not least, the ASGCN model may be extended to simultaneously judge sentiments of multiple aspects by capturing dependencies between the aspects words.

## Acknowledgments

This work is supported by The National Key Research and Development Program of China (grant No. 2018YFC0831704), Natural Science Foundation of China (grant No. U1636203, 61772363), Major Project of Zhejiang Lab (grant No. 2019DH0ZX01), and the European Unions Horizon 2020 Research and Innovation Programme under the Marie Skodowska-Curie grant agreement No. 721321.

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. 2017. Recurrent attention network on memory for aspect sentiment analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 452–461.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *Proceedings of the 52nd annual meeting of the association for computational linguistics (volume 2: Short papers)*, volume 2, pages 49–54.

Chuang Fan, Qinghong Gao, Jiachen Du, Lin Gui, Ruifeng Xu, and Kam-Fai Wong. 2018. Convolution-based memory network for aspect-based sentiment analysis. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1161–1164. ACM.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2018. Effective attention modeling for aspect-level sentiment classification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1121–1131.

Binxuan Huang, Yanglan Ou, and Kathleen M Carley. 2018. Aspect level sentiment classification with attention-over-attention neural networks. In *International Conference on Social Computing, Behavioral-Cultural Modeling and Prediction and Behavior Representation in Modeling and Simulation*, pages 197–206. Springer.

Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. 2011. Target-dependent twitter sentiment classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 151–160. Association for Computational Linguistics.

Rie Johnson and Tong Zhang. 2015. Semi-supervised convolutional neural networks for text categorization via region embedding. In *Advances in neural information processing systems*, pages 919–927.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

Svetlana Kiritchenko, Xiaodan Zhu, Colin Cherry, and Saif Mohammad. 2014. Nrc-canada-2014: Detecting aspects and sentiment in customer reviews. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 437–442.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Twenty-ninth AAAI conference on artificial intelligence*.

Xin Li, Lidong Bing, Wai Lam, and Bei Shi. 2018. Transformation networks for target-oriented sentiment classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 946–956.

Jiangming Liu and Yue Zhang. 2017. Attention modeling for targeted sentiment. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 572–577.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2017. Interactive attention networks for aspect-level sentiment classification. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 4068–4074. AAAI Press.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1506–1515.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, AL-Smadi Mohammad, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphée De Clercq, et al. 2016. Semeval-2016 task 5: Aspect based sentiment analysis. In *Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016)*, pages 19–30.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Suresh Manandhar, and Ion Androutsopoulos. 2015. Semeval-2015 task 12: Aspect based sentiment analysis. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 486–495.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 27–35.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1556–1566.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2016a. Effective lstms for target-dependent sentiment classification. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3298–3307.

Duyu Tang, Bing Qin, and Ting Liu. 2016b. Aspect level sentiment classification with deep memory network. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 214–224.

Shikhar Vashishth, Shib Sankar Dasgupta, Swayambhu Nath Ray, and Partha Talukdar. 2018. Dating documents using graph convolution networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1605–1615.

Duy-Tin Vo and Yue Zhang. 2015. Target-dependent twitter sentiment classification with rich automatic features. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Yequan Wang, Minlie Huang, Li Zhao, et al. 2016. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 606–615.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057.

Wei Xue and Tao Li. 2018. Aspect based sentiment analysis with gated convolutional networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2514–2523.

Min Yang, Wenting Tu, Jingxuan Wang, Fei Xu, and Xiaojun Chen. 2017. Attention based lstm for target dependent sentiment classification. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2018. Graph convolutional networks for text classification. *arXiv preprint arXiv:1809.05679*.

Yuhao Zhang, Peng Qi, and Christopher D Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2205–2215.

# Coupling Global and Local Context for Unsupervised Aspect Extraction

**Ming Liao**[1,2]*, **Jing Li**[3,†], **Haisong Zhang**[4], **Lingzhi Wang**[1,2], **Xixin Wu**[1], **Kam-Fai Wong**[1,2]
[1]The Chinese University of Hong Kong, Hong Kong, China
[2]MOE Laboratory of High Confidence Software Technologies, China
[3]Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China
[4]Tencent AI Lab, Shenzhen, China
[1,2]{mliao,lzwang,wuxx,kfwong}@se.cuhk.edu.hk
[3]jing-amilia.li@polyu.edu.hk, [4]hansonzhang@tencent.com

## Abstract

Aspect words, indicating opinion targets, are essential in expressing and understanding human opinions. To identify aspects, most previous efforts focus on using sequence tagging models trained on human-annotated data. This work studies *unsupervised* aspect extraction and explores how words appear in **global context** (on sentence level) and **local context** (conveyed by neighboring words). We propose a novel neural model, capable of coupling global and local representation to discover aspect words. Experimental results on two benchmarks, laptop and restaurant reviews, show that our model significantly outperforms the state-of-the-art models from previous studies evaluated with varying metrics. Analysis on model output show our ability to learn meaningful and coherent aspect representations. We further investigate how words distribute in global and local context, and find that aspect and non-aspect words do exhibit different context, interpreting our superiority in unsupervised aspect extraction.

## 1 Introduction

Opinion, one of the main factors shaping human behavior, is crucial to our daily activities (Liu, 2012). Every choice we make in our life, ranging from where to go for a Friday dinner to which job offer to pick up, is largely influenced by what other people think. To help individuals navigate decision-making processes, there exists growing attentions on opinion mining algorithms that distill massive opinion-rich texts — such as digital product reviews (Poddar et al., 2017) and social media discussions (Dusmanu et al., 2017) — into the opinionated information we need.

| |
|---|
| [R1]: It's truly a great **laptop** for the **price**. |
| [R2]: If you have the money, I suggest going for the **i7** |

Table 1: Two sample laptop review sentences. **Aspect words** are in boldface and blue. Wavy underlines indicate local context words indicating aspect word "*i7*".

Towards human opinion understanding, it is essential to figure out what target the opinion centers around. After all, previous studies have long pointed out that human language mostly conveys opinion with aspect and sentiment words (Liu, 2012). In this work, we focus on aspect extraction, targeting at the recognition of words indicating opinion aspects (henceforth **aspect words**). We believe developing effective aspect extraction models will benefit a broad range of compelling applications, such as aspect-based sentiment classification (Tang et al., 2016), opinion summarization (Wu et al., 2016), trending event tracking (Feng et al., 2016), and so forth.

To date, most progress made in aspect extraction has focused on training sequence tagging models on human-annotated data (Li and Lam, 2017; Xu et al., 2018; Wang and Pan, 2018). However, acquiring manual labels will inevitably undergo an expensive data annotation process and is hence difficult to scale for datasets from new domain or language. In this work, we explore how aspect words can be discovered in a fully *unsupervised* manner. We are inspired by the linguistic phenomenon that *aspect words generally distinguish themselves from other words in their occurrence patterns within global and local context*. Here **global context** refers to how pairs of words co-occur with each other at sentence level (without considering word order and can be extended to capture document-level context), while **local context** means what neighbors a word has.

To illustrate why global and local context can work together to indicate aspect words, Table 1

*This work was partially done when Ming Liao was an intern at Tencent AI Lab, Shenzhen, China.

†Jing Li is the corresponding author and conducted most of the work at Tencent AI Lab, Shenzhen, China.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4579–4589, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

shows two sentences from laptop review benchmark (Pontiki et al., 2016). As can be seen from R1, aspect words "*price*" and "*laptop*" tend to appear together in R1-like sentences concerning "*laptop price*". As for R2, its aspect words "*i7*", though not co-occurring with other aspects, have similar neighbors "*for the*" in local context with "*price*", which reveals its high likelihood of being aspect words, the same as "*price*".

Inspired by the phenomenon above, we propose a novel *unsupervised* model capable of coupling global and local context to discover aspect word clusters. Our model is built on the success of topic models in aspect extraction (Lin and He, 2009; Brody and Elhadad, 2010; Zhao et al., 2010). It is attributed to their ability to form latent topics with words likely to co-occur in a subset of sentences instead of widely appearing in the entire corpus (Blei et al., 2003). These words happen to exhibit similar patterns of how aspect words occur on sentence level (Lin and He, 2009). However, the above methods, only exploiting global context, are arguably suboptimal for largely ignoring the rich information delivered by local context. Some recent work (He et al., 2017), on the other way around, focus on using local context, yet ignore its coupled effects with global context.

Our work, to the best of our knowledge, is the first to explore *how global and local context jointly indicate aspect words*. Moreover, taken advantage of the recent advances in neural topic models (Miao et al., 2017; Srivastava and Sutton, 2017), we enable end-to-end learning of global and local representation, where the interaction between them contributed to aspect recognition can be automatically captured.

In experiments, we first compare our model with existing unsupervised models on aspect extraction. The results on restaurant and laptop reviews show that our model outperforms state-of-the-art approaches using global or local context only. For example, we achieve 36.1 F1 on laptop dataset, compared with 32.9 produced by He et al. (2017). Further discussions demonstrate our capability of capturing meaningful representations from global and local context, which interprets our superiority in aspect extraction. In addition, we empirically analyze global and local word context on our datasets. The results confirm that aspect words indeed vary in their global and local context compared with non-aspect ones, hence providing useful clues for aspect identification.

## 2 Related Work

Our work is mainly in the line with aspect extraction research. On this task, early studies mostly focus on the design of hand crafted rules (Hu and Liu, 2004; Zhuang et al., 2006; Qiu et al., 2011) or features (Jin et al., 2009; Li et al., 2010). Recently, the propose of neural models enables automatic representation learning without labor-intensive feature engineering (Wang et al., 2016, 2017; Li and Lam, 2017; Xu et al., 2018; Wang and Pan, 2018). These supervised models, rely on manually annotated data, thus restricted in their scaling ability for new domain or language. Instead, our work, focusing on *unsupervised* aspect extraction, can discover aspect words via exploiting how words occur in global and local context.

Our work is inspired by the unsupervised methods capturing latent aspect factors with LDA-style topic models (Lin and He, 2009; Brody and Elhadad, 2010; Zhao et al., 2010). We are also related with *non-neural* models incorporating word embeddings (encoding local context) to learn latent topics (discovered from global context) (Nguyen et al., 2015; Li et al., 2016; Shi et al., 2017). Compared with them — relying on expertise to customize inference algorithms, our model — in a neural architecture — does not require model-specific derivation, and enables interactions between global and local representations to be automatically learned. Though some neural models were recently proposed for our task (Wang et al., 2015; He et al., 2017), they focus on local context, unable to leverage global information. Distinguishing from them, we examine *how the coupled effects of global and local context can signal aspect words*, which have never been studied before in previous work.

## 3 Our Neural Model Coupling Global and Local Context

This section describes our neural model coupling the force of global and local context for aspect extraction. Figure 1 shows our overall architecture. There are two modules composed, one for local context modeling and the other for global.

In the following, we first describe the formulation of input and output in Section 3.1. Then the local and global context modeling process will be in turn given in Section 3.2 and 3.3.

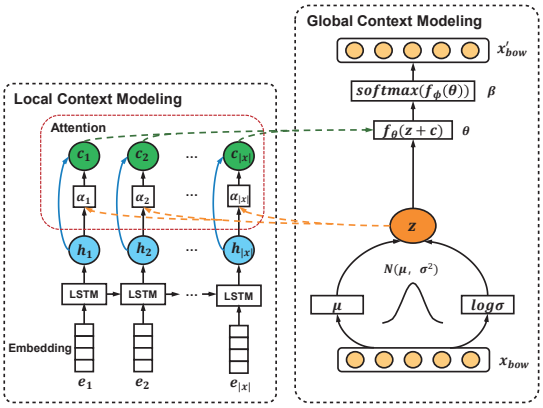


Figure 1: Our overall architecture with one module for local context modeling (on the left) and the other for global (on the right).

### 3.1 Input and Output

Before touching details to reveal how our model works, we first describe our input and output.

Formally, given a corpus $\mathcal{C}$ with $|C|$ sentences, $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{|C|}\}$, we process each sentence $\mathbf{x}$ into two forms: word sequence form $\mathbf{x}_{seq}$ and bag-of-words (BoW) form $\mathbf{x}_{bow}$. $\mathbf{x}_{seq} = \langle w_1, w_2, ..., w_{|\mathbf{x}|} \rangle$, where $w_n$ indicates word index of the $n$-th word and $|\mathbf{x}|$ denotes the number of words. $\mathbf{x}_{bow}$ is the BoW term vector over the vocabulary $V$. Here $\mathbf{x}_{seq}$ (considering word order) is fed for modeling local context and learning how words co-occur with their neighbors, while $\mathbf{x}_{bow}$ (following the bag-of-words assumption in most topic models (Blei et al., 2003; Miao et al., 2017)) serves as the input for global context modeling and capture sentence-level word co-occurrence.

Our goal is to output distributional clusters of aspect words. Then following Qiu et al. (2011)'s practice, the top $N$ nouns ($N$ as a hyperparameter) from each cluster (ranked by likelihood) are selected as the extracted aspect words, considering most aspects are nouns.

### 3.2 Local Context Modeling

As mentioned above, local context modeling module takes word sequence form, $\mathbf{x}_{seq}$, as its input. In this module, each word $w_n \in \mathbf{x}_{seq}$ is first processed with an embedding layer and converted into an embedding vector $\mathbf{e}_n$. Then we employ long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) network to explore local context. Word embeddings $\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_{|\mathbf{x}|}$ are processed into hidden states via recurrently exploring word co-occurrence with left neighbors. Specifically, for word $w_n$, its hidden states $\mathbf{h}_n$ is:

$$\mathbf{h}_n = f_{LSTM}(\mathbf{e}_n, \mathbf{h}_{n-1}) \quad (1)$$

where $f_{LSTM}(\cdot)$ refers to an LSTM unit. The hidden states $\langle \mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_{|\mathbf{x}|} \rangle$ are considered as the local representation, further leveraged in global context modeling and described later.

### 3.3 Global Context Modeling

Our global context modeling module is inspired by previous practice that discovers aspect words with LDA-fashion Bayesian graphical models (Lin and He, 2009). We assume there are $K$ latent aspect factors embedded in the the given corpus $\mathcal{C}$. Each factor $\phi_k$ ($k = 1, 2, ..., K$) is represented with a distributional word cluster over the vocabulary $V$.

Also Inspired by neural topic models (Miao et al., 2017), we adopt a variational auto-encoder (VAE) (Kingma and Welling, 2013), with an encoder and a decoder, to resemble the topic model-style data generation process. In doing so, we enable latent aspects, capturing word co-occurrence in both global and local context, to be learned in neural architecture. There are two main steps involved: First, the input sentence $\mathbf{x}$ (in BoW form $\mathbf{x}_{bow}$) is encoded to global representation $\mathbf{z}$. Conditioned on $\mathbf{z}$, together with local representation $\mathbf{h}_n$ (defined in Section 3.2 and $n = 1, 2, ..., |\mathbf{x}|$), decoder further generates $\mathbf{x}'_{bow}$, the BoW-form reconstruction of $\mathbf{x}_{bow}$. In the rest of this section, we first introduce how global representation $\mathbf{z}$ is learned by encoder from global context. Then we introduce how global and local representations are coupled to work together for data generation.

**Global Representation Encoding.** The encoder is employed to learn global representation, $\mathbf{z}$, from $\mathbf{x}_{bow}$. Following Miao et al. (2017), words in global context are assumed to satisfy Gaussian distribution, prior on mean $\mu$ and standard deviation $\sigma$. Their estimation formula are defined as:

$$\mu = f_\mu(f_e(\mathbf{x}_{bow})), \log \sigma = f_\sigma(f_e(\mathbf{x}_{bow})) \quad (2)$$

where $f_*(\cdot)$ is a neural perceptron performing a linear transformation operation followed by a non-linear ReLU activation (Nair and Hinton, 2010).

**Coupling Global and Local Context.** Recall we obtain the hidden states $\mathbf{h}_n$ ($n = 1, 2, ..., |\mathbf{x}|$) from local context modeling, and here we describe how we couple them with global representation $\mathbf{z}$.

Concretely, we employ attention mechanism (Bahdanau et al., 2015) over the hidden states in local representation, which, in aware

of global information, aims to identify words in $\mathbf{x}$ that can usefully indicate its aspect factors. We design attention weight $\alpha_n$ to measure the similarity between the semantic meaning of word $w_n$ and $\mathbf{x}$'s global representation $\mathbf{z}$:

$$\alpha_n = \mathbf{z}^T f_h(\mathbf{h}_n) \tag{3}$$

where $f_h(\cdot)$ is a ReLU activation function. The context vector of this attention, namely **globally-scoped local representation**, is defined as:

$$\mathbf{c} = \sum_n^{|\mathbf{x}|} \alpha_n \mathbf{h}_n \tag{4}$$

In the next step, the decoder will use $\mathbf{c}$ for learning corpus-level aspect factors and reproducing $\mathbf{x}_{bow}$. Below comes more details.

**Decoding Process.** Given the global representation $\mathbf{z}$ and the globally-scoped local representation $\mathbf{c}$, the decoder carries out the data generation process conditioned on both of them. For each input sentence $\mathbf{x}$, we assume each word $w_n \in \mathbf{x}_{bow}$ is sampled conditioned on its aspect mixture, $\theta$, a $K$-dim distribution reflecting $\mathbf{x}$'s composition of aspect factors. $\theta$ is then estimated with both $\mathbf{z}$ and $\mathbf{c}$, conveying global and local context of $\mathbf{x}$'s words. The story describing $\mathbf{x}$'s generation process is:

- Draw global representation $\mathbf{z} \sim \mathcal{N}(\mu, \sigma^2)$.
- Obtain globally-scoped local representation $\mathbf{c}$ with Eq. 4.
- Aspect mixture $\theta = softmax(f_\theta(\mathbf{z} + \mathbf{c}))$.
- For the $n$-th word in $\mathbf{x}$:
  - $\beta_n = softmax(f_\phi(\theta))$.
  - Draw the word $w_n \sim Multi(\beta_n)$.

Here $f_*(\cdot)$ is a ReLU-activated neural perceptron described above. Particularly, the weight matrix of $f_\phi(\cdot)$ (with the softmax normalization) are employed as the aspect-word distributions (distributional word clusters), $\phi$, used to represent the latent aspect factors and serves as our main output.

**Learning Objective.** We design the learning objective of our entire framework as:

$$\mathcal{L} = D_{KL}(p(\mathbf{z})||q(\mathbf{z}\,|\,\mathbf{x})) - \mathbb{E}_{p(\mathbf{z})}[p(\mathbf{x}\,|\,\mathbf{z})] \tag{5}$$

where $p(\mathbf{z})$ is a standard Gaussian prior. The first term reflects encoding loss while the second estimation likelihood (for decoding). We refer the readers to Miao et al. (2017) for more details.

## 4 Experimental Setup

**Datasets.** We conduct experiments on two benchmark datasets constructed for the SemEval aspect-based sentiment analysis (ABSA) challenge (Pontiki et al., 2014, 2015, 2016) with human annotated aspects — one gathers restaurant reviews (henceforth **restaurant**) and the other consists of laptop reviews (henceforth **laptop**). For model training and evaluation, we combine the training and test datasets for 2014-2016 ABSA (except for 2015 without laptop data released). Following common practice (Wagner et al., 2014), a review sentence is considered as a data sentence for input. The statistics of our datasets are displayed in Table 2. We can see that aspect words take around 25.0% of the vocabulary, yet less than 19.1% of the words per sentence. It is indicated the sparsity and diversity of aspect words, further suggesting the challenging of our task.

| | # of sen | \|Voc\| | Avg len per sen | \|Apt\| | # of apt per sen |
|---|---|---|---|---|---|
| **Laptop** | | | | | |
| Train | 6,355 | 3,374 | 14.51 | 837 | 2.35 |
| Test | 800 | 1,866 | 13.17 | 430 | 2.52 |
| **Rest** | | | | | |
| Train | 6,359 | 5,166 | 13.20 | 1,404 | 2.36 |
| Test | 2,161 | 3,800 | 13.51 | 948 | 2.41 |

Table 2: Statistics of laptop and restaurant (rest) datasets. |Voc|: the vocabulary size (including stop words). Avg len: average number of tokens. |Apt|: the number of distinct aspects. # of apt per sen: average number of aspects in a sentence.

**Preprocessing.** Here are our preprocessing steps. First, we adopted NLTK toolkit for text tokenization.[1] Then, we normalized all letters into their lower cases. Next, we removed words appearing less than five times. Finally, for BoW-form input, we removed all stop words and punctuation following common practice in topic models (Blei et al., 2003).

**Parameter Setting.** We applied pre-trained GloVe embedding (Pennington et al., 2014) for initialization in local context modeling.[2] The embedding dimension is set to 300, and batch size to 128. For the number of latent aspect factors, $K$, we tuned it on training data with five-fold validation and set it to 40. As for $N$, the number of nouns to be selected from each aspect word cluster, we set it to 30 following Qiu et al. (2011). In model training, we employ Adam optimizer (Kingma and Ba, 2015), with learning rate set to

[1] https://www.nltk.org

[2] https://nlp.stanford.edu/projects/glove/

$1e-3$, and run 20 epochs with early stop strategy adopted. Dropout strategy (Srivastava et al., 2014) is also adopted to avoid overfitting.

**Evaluation Metrics.** To ensure comparable performance, for clustering-based approaches, we select the top 30 nouns from 40 aspect clusters, same as our set up. For the rest, the top $1,200$ nouns are extracted. Here we adopt two sets of evaluation metrics. First, we follow Qiu et al. (2011) to test sentence-level aspect extraction, where the intersection of our selection and the words appear in a review sentence are considered as the extracted aspects. In this evaluation, we report precision, recall, and F1 scores. Second, we evaluate our ability to build aspect lexicon (a.k.a. corpus-level extraction) following Hamilton et al. (2016). We consider all the annotated aspects as gold standard lexicon and adopt accuracy for evaluation.

**Comparisons.** We first consider a simple baseline that randomly selects nouns as aspect words (henceforth RANDOM). We also compare with extracting- and clustering-based baselines — TF-IDF (Bahdanau et al., 2015), K-MEANS (Lloyd, 1982) (implemented with sklearn toolkit[3] and taking Glove embedding for similarity measure), and BTM[4] (Yan et al., 2013), state-of-the-art in short text topic modeling and well-performed in aspect extraction (He et al., 2017).

In addition, we consider the following recently proposed *unsupervised* models in comparison: LF-LDA (Nguyen et al., 2015), LDA topic model incorporating word embeddings (GloVe is applied here), and ABAE (He et al., 2017), the state-of-the-art attention-based model for unsupervised aspect extraction. Besides the existing models, we also compare with our variant that only models global context with neural topic model (henceforth GBC ONLY). The full model coupling global and local context is hence referred to as LCC+GBC.

## 5 Experimental Result

In this section, we first discuss comparison results with unsupervised aspect extraction models in Section 5.1. Section 5.2 shows what our model learns and interprets why it can discover aspect words. Next, in Section 5.3, we carry out an empirical study over how global and local context indicate aspect words. Last, we further discuss our parameter effects and main error in Section 5.4.

[3] https://scikit-learn.org/stable/

[4] https://github.com/xiaohuiyan/BTM

| Models | Restaurant | | | | Laptop | | | |
|---|---|---|---|---|---|---|---|---|
| | Pre | Rec | F1 | Acc | Pre | Rec | F1 | Acc |
| **Comparisons** | | | | | | | | |
| RANDOM | 24.9 | 20.4 | 22.4 | 17.3 | 24.1 | 39.0 | 29.8 | 20.8 |
| TF-IDF | 28.6 | 24.8 | 26.6 | 24.3 | 22.5 | 18.3 | 20.2 | 21.9 |
| K-MEANS | 28.1 | 40.0 | 33.0 | 19.0 | 23.0 | 35.6 | 27.9 | 23.5 |
| BTM | 30.6 | 56.4 | 39.7 | 21.2 | 25.8 | 48.8 | 33.7 | 31.3 |
| LF-LDA | 30.2 | 60.3 | 40.2 | 24.8 | 26.3 | 50.1 | 34.4 | 28.4 |
| ABAE | 30.9 | 57.8 | 40.2 | 23.6 | 25.4 | 46.5 | 32.9 | 32.0 |
| **Our models** | | | | | | | | |
| GBC ONLY | 30.5 | 57.9 | 39.9 | 24.2 | 25.6 | 49.8 | 33.8 | 29.8 |
| LCC+GBC | **31.2** | **60.5** | **41.2** | **26.0** | **28.0** | **50.2** | **36.1** | **33.7** |

Table 3: Precision (pre), Recall (rec), F1, and Accuracy (Acc) produced by various unsupervised models. Acc measures corpus-level extraction while others sentence-level. Our model significantly outperform others in F1 and Acc measure (paired t-test, $p < 0.05$).

### 5.1 Aspect Extraction Results

**Main Comparison Results.** In Table 3, we report the aspect extraction results on two datasets. Several interesting observations can be drawn:

• *All models tend to yield better F1 on restaurant yet better Acc on laptop.* We find that restaurant exhibits generally worse model performance on corpus-level extraction than that on sentence level. The opposite findings is whereas drawn on laptop. It might be because restaurant contains more distinct aspects (shown with the larger |Apt| in Table 2). It is possibly due to the prominence of rare aspects, which is challenging to be discovered.

• *Simple baselines do not work well.* Both RANDOM and TF-IDF perform poorly, indicating the challenge of unsupervised aspect extraction.

• *Global context can well indicate aspects.* We observe that approaches based on topic models (BTM, LF-LDA, and our models) perform better than others. The results indicate that aspect words do vary from other words in global context distributions. Topic model-based approaches, via exploiting sentence-level word co-occurrence, can thus effectively identify aspect words.

• *Coupling global and local context is effective.* By combining topic models (global context) with word embeddings (local context), LF-LDA produces the second best F1. Also, our full model LCC+GBC outperforms its variant GBC ONLY in F1. These observations indicate the benefit of joint modeling of global and local context to discover aspect words.

Besides, by comparing model performance over the two datasets, we observe that all models perform worse on laptop. An intuitive explanation is that laptop reviews generally concern wide

| GBC ONLY | LCC+GBC |
|---|---|
| *good*, **menu**, little, kind, **noise**, play, daniel, **fare**, **jelly**, much, details, father, **neighborhood**, **wine**, door, possible, murray, keep, vagan, heaviness, *cool*, wins, angel, upper, *romantic*, takes, **avenue**, **fruit**, pink, strips | **value**, **wine**, *date*, evening, **food**, **block**, **avenue**, **line**, *fun*, years, *love*, yes, hang, knows, must, **cheese**, *favorite*, **course**, *romantic*, **tip**, jeans, servers, cold, **pastrami**, **atmosphere**, *fine*, **counter**, word, **sauce**, phone |
| crash, **keyboard**, encounter, 39, **battery**, **wires**, **memory**, photographs, *sooner*, fits, **feel**, things, steve, overheat, would, seem, **pro**, **touch**, **cpu**, **mouse**, figure, user, *better*, users, ran, days, question, apple, worth, sit | needed, **skype**, advise, **keyboard**, remote, daily, matches, **high**, *love*, originally, wife, loud, *excellent*, **macbook**, freezes, *well*, anytime, got, friend, *weird*, even, **gui**, weeks, recently, **internet**, **8.1**, **laptops**, tapping, **speakers**, noon |

Table 4: Top 30 words of sample latent aspects learned by our variant GBC ONLY (on the left) and full model LCC+GBC (on the right). The top displays the outputs on restaurant dataset and the bottom laptop. **Aspect words** are in boldface and blue (annotated in least one sentence), while *sentiment words* are in italic and red.

range of aspects (e.g., screens, battery, etc.), while restaurant reviews tend to be centered around general aspects (e.g., food and service). Aspect words thus exhibit sparse occurrence patterns in laptop reviews, rendering generally worse model performance. Section 5.3, we will discuss more. For the same reason, local context helps LCC+GBC obtain larger margin on laptop, compared with models relying on global word co-occurrence.

### 5.2 Model Interpretation

Here we probe into our output and study why LCC+GBC model works.

**Topic Coherence.** We first analyze the coherence of our latent aspects, where $C_V$ metrics, a widely-applied automatic topic coherence measure (Röder et al., 2015) is adopted. LCC+GBC's latent aspects achieves $C_V$ coherence scores of 0.401 and 0.393 on restaurant and laptop dataset, respectively, compared to 0.382 and 0.377 produced by GBC ONLY. It hence suggests the joint effects of global and local context also helps produce coherent aspects.

**Sample Latent Aspects.** We further conduct a qualitative analysis on the produced latent aspects. Table 4 shows the top 30 words (ranked by likelihood) of the sample latent aspects. LCC+GBC's output aspects look more coherent, with words

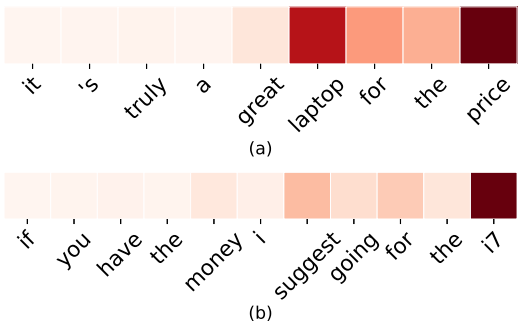


Figure 2: The globally-scoped local attention weights learned by our LCC+GBC model for the sample sentences in Table 1. Darker colors indicate higher values.

having similar semantics clustered together, such as "*course*", "*romantic*", and "*date*" learned from restaurant dataset, and "*skype*", "*remote*", and "*internet*" from laptop. The possible reason is that words conveying similar semantic meanings tend to appear in similar local context. LCC+GBC, via coupling local context with global one, is thus able to capture such semantic representations.

We also notice that LCC+GBC discovers more rare aspects, such as "*pastrami*" (from restaurant) and "*gui*" (from laptop). These words, though may exhibit sparse occurrence and unable to be discovered purely with global context, might be effectively indicated by their local context. This reveals the benefit of combining the effects of global and local context for aspect extraction.

In addition, we notice that our output aspect clusters include some sentiment words. It is possibly because aspect words tend to co-occur with sentiment ones in both global and local context. Thus without supervision, it is likely our models discover them together. These findings suggest our potential benefit on extracting sentiment words — which can be easily separated from aspects by their POS tags (Qiu et al., 2011)). Such extension is beyond the scope of this paper but worth exploring in future work.

**Case Study.** To understand what LCC+GBC learns resulting its superiority in aspect extraction. We take the two samples in Table 1 as input. Figure 2 visualizes their globally-scoped attention weights (defined in Eq. 3). We observe that LCC+GBC assigns high attention weights for aspect words "*laptop*", "*price*", and "*i7*". Also highlighted are the neighboring words "*for the*" in both attentions, usefully signaling their next word to be aspect. The results indicate LCC+GBC learns meaningful information via exploring inter-

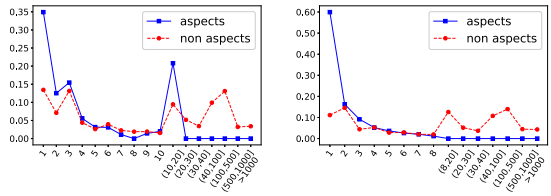


(a) Restaurant (b) Laptop

Figure 3: Sentence-level co-occurrence of aspect (in blue line) and non-aspect (in red line) word pairs distribution from two datasets. X-axis: word pair frequency. Y-axis: proportion of pairs. Aspect and non-aspect words exhibit diverse distributions.

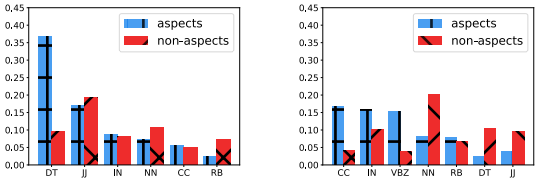


(a) Left Context (b) Right Context

Figure 4: The distribution of neighboring POS tags in left and right context ( laptop dataset). Aspect neighbors are in red and non-aspects' in blue. X-axis: neighboring POS tags. Y-axis: POS proportions. Aspects' and non-aspects' neighbors have different POS tags.

actions between global and local representation.

### 5.3 Analysis of Global and Local Context

To extensively understand the effects of global and local context on aspect identification, we carry out an empirical study on our datasets.

**Aspects vs. Non-aspects.** We first compare the global and local word occurrence statistics in context of aspect and non-aspect words. This is to empirically analyze why LCC+GBC can effectively distinguish aspect and non-aspect words.

To examine global context, Figure 3 presents distributions of word pair co-occurrence in sentences, with two lines corresponding to aspect and non-aspect pairs. It is seen different distribution are exhibited by aspect and non-aspect pairs, with non-aspect ones flatly distributed over varying frequency while the aspect pairs are more sparse. This indicates global context, capturing how words co-appear in sentences, can indeed help distinguish aspect and non-aspect words.

We also notice that aspect pair distributions are slightly different on two datasets. On restaurant dataset, we observe a pulse on pairs occurring $10-20$ times, while the distribution on laptop is a long tail. This demonstrates the sparse aspect occurrence patterns in laptop dataset (probably owing to the broad range of aspects discussed there), also explains the general worse performance on it (compared to restaurant and shown in Table 3).

We then analyze local context and show the distribution of POS tags (predicted with NLTK toolkit) in left and right neighbors. We take laptop dataset as an example to discuss, and similar observations are drawn from restaurant. For better displays, from 34 POS tags in total, we pick up the top 5 tags in aspects' and non-aspects' neighbors respectively. Distributions are shown over their union set in Figure 4. In both left and right context, we observe aspects and non-aspects exhibit different distributions for their neighboring POS tags. For example, aspect words are likely to have "*DT*" (e.g., *an*, *this*) appearing as left while "*RB*" (e.g., *highly*, *barely*) frequently acting as non-aspects' left neighbors while rarely in aspects'.

In addition, we display in Table 5 the top 10 neighboring bigrams in left context. Although it is merely a qualitative human judgement at this point, we can draw some interesting observations from the results. For example, some opinioned bigrams, such as "a great" and "love the", are likely to appear on the left local context of aspects (opinion targets). Such patterns may usefully indicate aspect words and help distinguish them from non-aspect ones. As for right bigram neighbors, they exhibit sparse occurrence patterns, hence might provide less useful clues. We will analyze the effects of left and right local context next.

| Aspects | Non-aspects |
|---|---|
| easy to, and the, of the, for the, with the, a great, that the, it 's, all the, love the | of the, it is, and the, is a, battery life, to use, it 's, with the, i have, for the |

Table 5: Top 10 neighboring bigrams in left context of aspects and non-aspects (laptop dataset).

**Local Context Modeling.** We then compare and discuss the effectiveness of varying modules to capture local representation. The performance of our variants combined with varying local encoders are shown in Table 6. It is observed that all variants with attention, in aware of global context and put over local representation, yield better performance. This shows that attention mechanism is able to capture interactions between global and local representations, which are useful in discovering aspect words. We also notice that LSTM encoder performs better than CNN and Bi-LSTM.

| | Restaurant | Laptop |
|---|---|---|
| W/O LCC | 39.9 | 33.6 |
| AVG EMB | 40.0 | 33.9 |
| LSTM *(w/o att)* | 40.1 | 34.4 |
| CNN *(w/ att)* | 40.4 | 34.0 |
| BI-LSTM *(w/ att)* | 40.6 | 34.8 |
| LSTM *(w/ att)* | **41.2** | **36.1** |

Table 6: F1 score of our variants with varying encoders for local context modeling. Here *att* refers to the attention to capture globally-scoped local representation (shown in Eq. 3 and 4). In the first column, W/O LCC refers to GBC ONLY variant. AVG EMB means average embedding. LSTM (*w/ att*) is our LCC+GBC model.

It is possibly because, in local context, left neighbors convey more useful clues for indicating aspect words, compared with right ones. As a result, CNN and BI-LSTM, equally considering left and right context, might be somehow affected by the noise in right context. They are thus outperformed by LSTM, which only models local context in left-to-right direction.

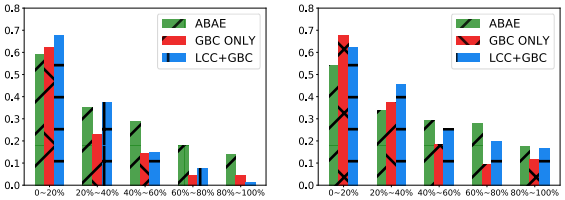


(a) Restaurant (b) Laptop

Figure 5: Recall scores for discovering varying quintiles (20%) of aspects (ranked by frequency). X-axis: quintiles of aspect frequency. Y-axis: recall scores.

**Varying Aspect Frequency.** Recall that the sample aspects in Table 4 suggest more rare aspects discovered by LCC+GBC compared with GBC ONLY. We conduct an analysis on model performance to discover aspects with varying frequency. Figure 5 compares the recall scores produced by ABAE, GBC ONLY, and LCC+GBC when retrieving varying aspect quintiles (5-quantile) (ranked by frequency). It shows ABAE performs better in discovering low-frequency aspects while GBC ONLY better at recognizing frequent ones. It suggests global context is more useful to indicate common aspects while local context better at signaling rare ones. LCC+GBC, capturing the coupled effects of global and local context, can identify both common and rare aspects, and thus yield superior performance.

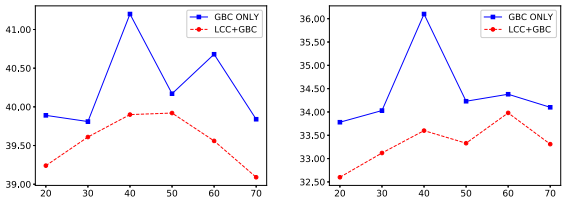


(a) Restaurant (b) Laptop

Figure 6: F1 scores of our models (in y-axis) given varying aspect number $K$ (in x-axis). LCC+GBC performs consistently better than GBC ONLY .

### 5.4 Further Discussion

**Parameter Analysis** In main results, we fix the number of latent aspects to $K = 40$. In Figure 6, we further examine how our models (GBC ONLY and LCC+GBC) perform given varying number of $K$. Here to ensure comparable performance, we set $N = 1200/K$. It is seen that LCC+GBC yields consistently better F1 than GBC ONLY. We also observe that both models do not exhibit monotomic curves, where LCC+GBC obtains the best performance given $K = 40$, consistent with the validation results.

**Error Analysis.** Here we analyze our main errors types. One major type is caused by wrongly identifying aspect phrases, such as "*windows 7*", where "*7*" is missed possibly ascribed to its sparse occurrence. We have such errors owning to modeling context in word level and sometimes fail to capture semantics in coarser grain. One possible solution is to extend our global context modeling module to learn phrase-level semantics (He, 2016). Another main errors occur when processing context-sensitive aspect words. For example, "*hard*" indicates aspect in "*The hard drive is fast.*", rather than "*It is hard to use that laptop.*". Our model, failing to distinguish "*hard*" in varying context, considers it as aspect for both sentences. To deal with such error, we can adopt context-aware decoders, such as Hu et al. (2017), to distinguish word semantics in different context.

## 6 Conclusion

We have presented a study of unsupervised aspect extraction via exploring the coupled effects of global and local context. A neural model has been proposed to learn the interactions between global and local representations indicative of aspect words. Experiment results on two benchmark datasets show our model outperform comparison approaches modeling local or global context only.

We find out three interesting points in empirical analysis over global and local context: First, aspects and non-aspects exhibit distinguishing distributions in either global and local context; Second, in local context, left neighbors can better indicate aspect words compared with the right; Third, local context can better indicate rare aspects while global signals common aspects better.

## Acknowledgments

This work is partially supported by the following HK grants: RGC-GRF (14232816, 14209416, 14204118, 3133237), NSFC (61877020) & ITF (ITS/335/18). We thank the three anonymous reviewers for the insightful suggestions on various aspects of this work.

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022.

Samuel Brody and Noemie Elhadad. 2010. An unsupervised aspect-sentiment model for online reviews. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 804–812, Los Angeles, California. Association for Computational Linguistics.

Mihai Dusmanu, Elena Cabrio, and Serena Villata. 2017. Argument mining on twitter: Arguments, facts and sources. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2317–2322, Copenhagen, Denmark. Association for Computational Linguistics.

Xiaocheng Feng, Lifu Huang, Duyu Tang, Heng Ji, Bing Qin, and Ting Liu. 2016. A language-independent neural network for event detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 66–71, Berlin, Germany. Association for Computational Linguistics.

William L. Hamilton, Kevin Clark, Jure Leskovec, and Dan Jurafsky. 2016. Inducing domain-specific sentiment lexicons from unlabeled corpora. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 595–605, Austin, Texas. Association for Computational Linguistics.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2017. An unsupervised neural attention model for aspect extraction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 388–397, Vancouver, Canada. Association for Computational Linguistics.

Yulan He. 2016. Extracting topical phrases from clinical documents. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 2957–2963. AAAI Press.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, Washington, USA, August 22-25, 2004*, pages 168–177.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1587–1596, International Convention Centre, Sydney, Australia. PMLR.

Wei Jin, Hung Hay Ho, and Rohini K Srihari. 2009. A novel lexicalized hmm-based learning framework for web opinion mining. In *Proceedings of the 26th annual international conference on machine learning*, pages 465–472. Citeseer.

Diederick P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Diederik P. Kingma and Max Welling. 2013. Auto-encoding variational bayes. *CoRR*, abs/1312.6114.

Chenliang Li, Haoran Wang, Zhiqian Zhang, Aixin Sun, and Zongyang Ma. 2016. Topic modeling for short texts with auxiliary word embeddings. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, SIGIR 2016, Pisa, Italy, July 17-21, 2016*, pages 165–174.

Fangtao Li, Chao Han, Minlie Huang, Xiaoyan Zhu, Yingju Xia, Shu Zhang, and Hao Yu. 2010. Structure-aware review mining and summarization. In *COLING 2010, 23rd International Conference on Computational Linguistics, Proceedings of the Conference, 23-27 August 2010, Beijing, China*, pages 653–661.

Xin Li and Wai Lam. 2017. Deep multi-task learning for aspect term extraction with memory interaction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2886–2892, Copenhagen, Denmark. Association for Computational Linguistics.

Chenghua Lin and Yulan He. 2009. Joint sentiment/topic model for sentiment analysis. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 375–384, New York, NY, USA. ACM.

Bing Liu. 2012. *Sentiment Analysis and Opinion Mining*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Stuart P. Lloyd. 1982. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28:129–137.

Yishu Miao, Edward Grefenstette, and Phil Blunsom. 2017. Discovering discrete latent topics with neural variational inference. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2410–2419, International Convention Centre, Sydney, Australia. PMLR.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 807–814, USA. Omnipress.

Dat Quoc Nguyen, Richard Billingsley, Lan Du, and Mark Johnson. 2015. Improving topic models with latent feature word representations. *Transactions of the Association for Computational Linguistics*, 3:299–313.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Lahari Poddar, Wynne Hsu, and Mong Li Lee. 2017. Author-aware aspect topic sentiment model to retrieve supporting opinions from reviews. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 472–481, Copenhagen, Denmark. Association for Computational Linguistics.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, Mohammad AL-Smadi, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphee De Clercq, Veronique Hoste, Marianna Apidianaki, Xavier Tannier, Natalia Loukachevitch, Evgeniy Kotelnikov, Núria Bel, Salud María Jiménez-Zafra, and Gülşen Eryiğit. 2016. Semeval-2016 task 5: Aspect based sentiment analysis. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 19–30, San Diego, California. Association for Computational Linguistics.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Suresh Manandhar, and Ion Androutsopoulos. 2015. Semeval-2015 task 12: Aspect based sentiment analysis. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 486–495, Denver, Colorado. Association for Computational Linguistics.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 27–35, Dublin, Ireland. Association for Computational Linguistics and Dublin City University.

Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. 2011. Opinion word expansion and target extraction through double propagation. *Computational Linguistics*, 37(1):9–27.

Michael Röder, Andreas Both, and Alexander Hinneburg. 2015. Exploring the space of topic coherence measures. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, WSDM '15, pages 399–408, New York, NY, USA. ACM.

Bei Shi, Wai Lam, Shoaib Jameel, Steven Schockaert, and Kwun Ping Lai. 2017. Jointly learning word embeddings and latent topics. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*, pages 375–384.

Akash Srivastava and Charles Sutton. 2017. Autoencoding variational inference for topic models. In *International Conference on Learning Representations (ICLR)*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2016. Effective lstms for target-dependent sentiment classification. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3298–3307. The COLING 2016 Organizing Committee.

Joachim Wagner, Piyush Arora, Santiago Cortes, Utsab Barman, Dasha Bogdanova, Jennifer Foster, and Lamia Tounsi. 2014. Dcu: Aspect-based polarity classification for semeval task 4. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 223–229, Dublin, Ireland. Association for Computational Linguistics and Dublin City University.

Linlin Wang, Kang Liu, Zhu Cao, Jun Zhao, and Gerard de Melo. 2015. Sentiment-aspect extraction based on restricted boltzmann machines. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language*

*Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 616–625.

Wenya Wang and Sinno Jialin Pan. 2018. Recursive neural structural correspondence network for cross-domain aspect and opinion co-extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2171–2181. Association for Computational Linguistics.

Wenya Wang, Sinno Jialin Pan, Daniel Dahlmeier, and Xiaokui Xiao. 2016. Recursive neural conditional random fields for aspect-based sentiment analysis. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 616–626, Austin, Texas. Association for Computational Linguistics.

Wenya Wang, Sinno Jialin Pan, Daniel Dahlmeier, and Xiaokui Xiao. 2017. Coupled multi-layer attentions for co-extraction of aspect and opinion terms. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3316–3322.

H. Wu, Y. Gu, S. Sun, and X. Gu. 2016. Aspect-based opinion summarization with convolutional neural networks. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 3157–3163.

Hu Xu, Bing Liu, Lei Shu, and Philip S. Yu. 2018. Double embeddings and cnn-based sequence labeling for aspect extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 592–598. Association for Computational Linguistics.

Xiaohui Yan, Jiafeng Guo, Yanyan Lan, and Xueqi Cheng. 2013. A biterm topic model for short texts. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 1445–1456, New York, NY, USA. ACM.

Wayne Xin Zhao, Jing Jiang, Hongfei Yan, and Xiaoming Li. 2010. Jointly modeling aspects and opinions with a maxent-lda hybrid. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, EMNLP 2010, 9-11 October 2010, MIT Stata Center, Massachusetts, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 56–65.

Li Zhuang, Feng Jing, and Xiaoyan Zhu. 2006. Movie review mining and summarization. In *Proceedings of the 2006 ACM CIKM International Conference on Information and Knowledge Management, Arlington, Virginia, USA, November 6-11, 2006*, pages 43–50.

# Transferable End-to-End Aspect-based Sentiment Analysis with Selective Adversarial Learning

**Zheng Li**[1], **Xin Li**[2], **Ying Wei**[3], **Lidong Bing**[4], **Yu Zhang**[5], **Qiang Yang**[1]
[1]The Hong Kong University of Science and Technology, Hong Kong
[2]The Chinese University of Hong Kong, Hong Kong
[3]Tencent AI Lab, Shenzhen, China
[4]R&D Center Singapore, Machine Intelligence Technology, Alibaba DAMO Academy
[5]Southern University of Science and Technology, Shenzhen, China
{zlict, qyang}@cse.ust.hk, lixin@se.cuhk.edu.hk, judywei@tencent.com
l.bing@alibaba-inc.com, yu.zhang.ust@gmail.com

## Abstract

Joint extraction of aspects and sentiments can be effectively formulated as a sequence labeling problem. However, such formulation hinders the effectiveness of supervised methods due to the lack of annotated sequence data in many domains. To address this issue, we firstly explore an unsupervised domain adaptation setting for this task. Prior work can only use common syntactic relations between aspect and opinion words to bridge the domain gaps, which highly relies on external linguistic resources. To resolve it, we propose a novel Selective Adversarial Learning (SAL) method to align the inferred correlation vectors that automatically capture their latent relations. The SAL method can dynamically learn an alignment weight for each word such that more important words can possess higher alignment weights to achieve fine-grained (word-level) adaptation. Empirically, extensive experiments[1] demonstrate the effectiveness of the proposed SAL method.

## 1 Introduction

End-to-End Aspect-Based Sentiment Analysis (E2E-ABSA) aims to jointly detect the aspect terms explicitly mentioned in sentences and predict the sentiment polarities over them (Liu, 2012; Pontiki et al., 2014). For example, in the sentence "*The **AMD Turin Processor** seems to always perform much better than **Intel***", the user mentions two aspect terms, i.e., "***AMD Turin Processor***" and "***Intel***", and expresses positive and negative sentiments over them, respectively.

Typically, prior work formulates E2E-ABSA as a sequence labeling problem over a unified tagging scheme (Mitchell et al., 2013; Zhang et al., 2015; Li et al., 2019a). The unified tagging scheme connects a set of *aspect boundary tags* (e.g., {`B`, `I`, `E`, `S`, `O`} denotes the beginning of, inside of, end of, single-word, and no aspect term), and *sentiment tags* (e.g. {`POS`, `NEG`, `NEU`} denotes positive, negative or neutral sentiment) together to constitute a joint label space for each word. As such, "***AMD Turin Processor***" and "***Intel***" should be tagged with {`B-POS`, `I-POS`, `E-POS`} and {`S-NEG`}, respectively, while the remaining words are tagged with `O`. This formulation makes two sub-tasks joint modeling easier, and meanwhile, tend to be low-resource. There usually exist few annotated data for each new domain, where labeling each word with a unified tag could be more time-consuming and expensive.

To alleviate the dependence on domain supervisions, we explore an unsupervised domain adaptation setting for E2E-ABSA, which aims to leverage knowledge from a labeled source domain to improve the sequence learning in an unlabeled target domain. The challenges in fulfillment of this setting are two-fold: (1) there exists a large feature distribution shift between domains since aspect terms in different domains are usually disjoint. For example, users usually mention "***pizza***" in the *Restaurant* domain while "***camera***" is often discussed in the *Laptop* domain; (2) Unlike domain adaptation in traditional sentiment classification (Blitzer et al., 2007) that learns shared sentence or document representations, we need to learn fine-grained (word-level) representations to be domain-invariant for sequence prediction.

Consider the first problem, i.e., what to transfer? Even though aspect terms from different domains behave distinctly, some association patterns between aspect and opinion words are common across domains; e.g., "*The **pizza** is great.*" from the *Restaurant* domain and "*The **camera** is excellent.*" from the *Laptop* domain. Both of them share the same syntactic pattern (*as-*

[1]The code is available at https://github.com/hsqmlzno1/Transferable-E2E-ABSA



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4590–4600, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

*pect words→nsubj→opinion words*). Inspired by this, existing studies use general syntactic relations as the pivot to bridge the domain gaps for cross-domain aspect extraction (Jakob and Gurevych, 2010; Ding et al., 2017), or aspect and opinion co-extraction (Li et al., 2012; Wang and Pan, 2018). Unfortunately, these methods highly rely on prior knowledge (e.g., manually-designed rules) or external linguistic resources (e.g., dependency parsers), which are inflexible and prone to bringing in knowledge errors. Instead, we introduce a multi-hop Dual Memory Interaction (DMI) mechanism to automatically capture the latent relations among aspect and opinion words. The DMI iteratively infers the correlation vectors of each word by interacting its local memory (LSTM hidden state) with both the global aspect and opinion memories, such that the inter-correlations between aspects and opinions, and the intra-correlations in aspects or opinions can be derived.

Second, how to transfer for this sequence prediction task? One straightforward way is to apply domain adaption methods to align all words within the sentence, however, it is observed that it will not yield significant improvements. Actually, not all the words contribute equally to the domain-invariant feature space though fine-grained adaptation is required. Thus, we propose a novel Selective Adversarial Learning (SAL) method to dynamically learn an alignment weight for each word, where more important words can possess higher alignment weights to achieve a local semantic alignment based on adversarial training. Empirically, the proposed model outperforms the state-of-the-art fine-grained adaptation methods by a large margin on four benchmark datasets. We also conduct extensive ablation studies to quantitatively and qualitatively demonstrate the effectiveness of the selectivity of adversarial learning.

Overall, our main contributions are summarized as: (1) to the best of our knowledge, an unsupervised domain adaptation setting is firstly explored for E2E-ABSA; (2) an effective SAL method is proposed to conduct a local semantic alignment for fine-grained domain adaptation; (3) extensive experiments verify the effectiveness of the proposed SAL method.

## 2 Task Definition

**Single-domain:** E2E-ABSA involves both aspect detection (**AD**) and aspect sentiment (**AS**) classification tasks, which are formulated as a unified sequence labeling problem. Given an input sequence of words $\mathbf{x}=\{w_1, w_2, ..., w_T\}$ and its word embeddings $\mathbf{e}=\{e_1, e_2, ..., e_T\}$, the goal is to predict a tag sequence $\mathbf{y}=\{y_1, y_2, ..., y_T\}$ over the *unified tags*, with $y_i \in \mathcal{Y}^{\mathcal{U}} = \{$`B-POS`, `I-POS`, `E-POS`, `S-POS`, `B-NEG`, `I-NEG`, `E-NEG`, `S-NEG`, `B-NEU`, `I-NEU`, `E-NEU`, `S-NEU`, `O`$\}$. **Cross-domain:** Here we are performing in a more challenging unsupervised domain adaptation setting. Given a set of labeled data $D_s = \{(\mathbf{x}_s^i, \mathbf{y}_s^i)\}_{i=1}^{N_s}$ from a source domain and a set of unlabeled data $D_t = \{(\mathbf{x}_t^j)\}_{j=1}^{N_t}$ from a target domain, we aim to transfer the knowledge of $D_s$ to improve the sequence learning in $D_t$.

## 3 Model Description

**Overview:** As shown in Figure 1, we adopt two stacked bi-directional LSTMs as the base model (Li et al., 2019a) for E2E-ABSA. The upper LSTM$^{\mathcal{U}}$ is for the high-level **ADS** (AD+AS) task and it predicts the *unified tags* as output, while the lower LSTM$^{\mathcal{B}}$ is for the low-level **AD** task and predicts the *aspect boundary tags* as the guidance. To adapt the base model, we design different components in terms of the two problems, i.e., what to transfer and how to transfer, respectively.

(1) To automatically capture the latent relations between aspect and opinion words as transferable knowledge across domains, we introduce a multi-hop Dual Memory Interaction (DMI) mechanism between the two LSTMs. At each hop, e.g., the 1st hop, each local memory $\mathbf{h}_i^{\mathcal{B}}$ will interact with both the global aspect and opinion memories, i.e., $\mathbf{m}_a^1$ and $\mathbf{m}_o^1$ based on the DMI, to produce two correlation vectors for aspect and opinion words co-detection, where the opinion detection is used as an auxiliary task for the AD task. The "local" memory denotes the hidden representation (LSTM$^{\mathcal{B}}$ hidden state) of each word within the sentence. Whereas the two "global" memories are globally shared by all input sentences, which are commonly used in memory networks (Sukhbaatar et al., 2015; Kumar et al., 2016) and can be seen as high-level representations for aspect and opinion words, respectively. The A-attention and O-attention then aggregate most relevant aspect or opinion words information to refine the two global memories for the next hop.

(2) To adapt these relations across domains, we propose a Selective Adversarial Learning (SAL)

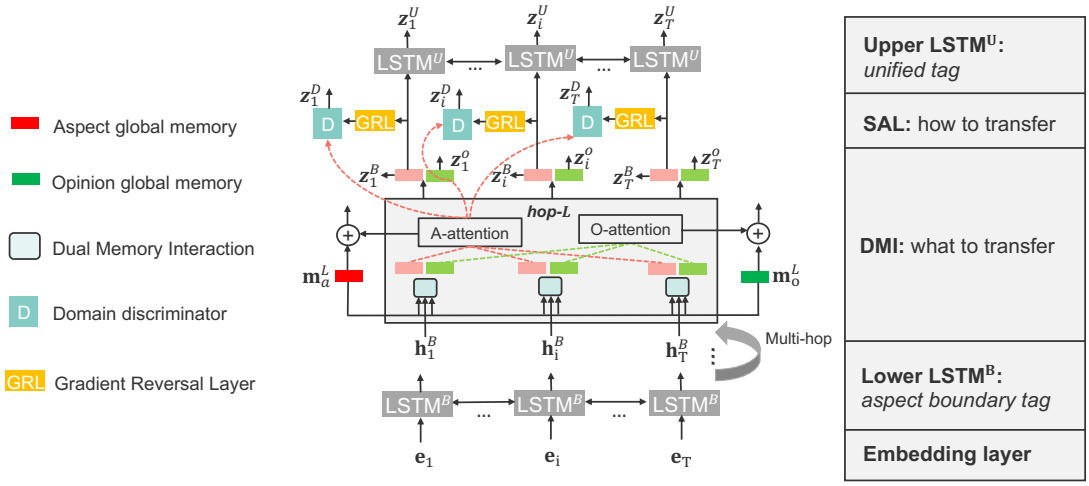


Figure 1: The framework of the proposed model.

method to dynamically focus on aligning the aspect words between domains. This is because the informative aspect words contribute more to the shared feature space than the unmeaning words tagged with O in the sentence (Zhou et al., 2019b). As such, an aspect tagger trained on a source domain can work well when applied to a target domain. Specifically, at the final hop, we adopt a domain discriminator for each word with a gradient reversal layer (Ganin et al., 2016) to perform domain adversarial learning over its correlation vector (alignment). While the A-attention module provides an aspect attention distribution as a selector to control a learnable alignment weight for each word (selectivity). Finally, each aligned correlation vector will be used to predict *aspect boundary tags* (AD task) and fed to the LSTM$^{\mathcal{U}}$ for the *unified tags* prediction (ADS task). In the following sections, we detail each component.

### 3.1 Base Model

We adopt two stacked bi-directional LSTMs as the base model. We link these two LSTM layers so that the hidden representations generated by the LSTM$^{\mathcal{B}}$ can be fed to LSTM$^{\mathcal{U}}$ as the guidance information. Specifically, their hidden representations $\mathbf{h}_i^{\mathcal{B}} \in \mathbb{R}^{\dim_h^{\mathcal{B}}}$ and $\mathbf{h}_i^{\mathcal{U}} \in \mathbb{R}^{\dim_h^{\mathcal{U}}}$ at the $i$-th time step ($i \in [1, T]$) are calculated as follows:

$$\mathbf{h}_i^{\mathcal{B}} = [\overrightarrow{\text{LSTM}}^{\mathcal{B}}(\mathbf{e}_i); \overleftarrow{\text{LSTM}}^{\mathcal{B}}(\mathbf{e}_i)],$$
$$\mathbf{h}_i^{\mathcal{U}} = [\overrightarrow{\text{LSTM}}^{\mathcal{U}}(\mathbf{h}_i^{\mathcal{B}}); \overleftarrow{\text{LSTM}}^{\mathcal{U}}(\mathbf{h}_i^{\mathcal{B}})].$$

The probability scores $\mathbf{z}_i^{\mathcal{B}} \in \mathbb{R}^{|\mathcal{Y}^{\mathcal{B}}|}$ over the *aspect boundary tags* $\mathcal{Y}^{\mathcal{B}} = \{\texttt{B}, \texttt{I}, \texttt{E}, \texttt{S}, \texttt{O}\}$ are calculated by a fully-connected softmax layer:

$$\mathbf{z}_i^{\mathcal{B}} = \mathbf{p}(\mathbf{y}_i^{\mathcal{B}}|\mathbf{h}_i^{\mathcal{B}}) = \text{Softmax}(\mathbf{W}_{\mathcal{B}}\mathbf{h}_i^{\mathcal{B}} + \mathbf{b}_{\mathcal{B}}).$$

Similarly, the scores $\mathbf{z}_i^{\mathcal{U}} \in \mathbb{R}^{|\mathcal{Y}^{\mathcal{U}}|}$ over the *unified tags* $\mathcal{Y}^{\mathcal{U}}$ defined in Section 2 are obtained as:

$$\mathbf{z}_i^{\mathcal{U}} = \mathbf{p}(\mathbf{y}_i^{\mathcal{U}}|\mathbf{h}_i^{\mathcal{U}}) = \text{Softmax}(\mathbf{W}_{\mathcal{U}}\mathbf{h}_i^{\mathcal{U}} + \mathbf{b}_{\mathcal{U}}).$$

### 3.2 Global-Local Memory Interaction

Before detailing the DMI module, we firstly introduce Global-Local Memory Interaction (GLMI) that describes the interaction between a local memory $\mathbf{h}_i \in \mathbb{R}^{\dim_h}$ and a global memory $\mathbf{m} \in \mathbb{R}^{\dim_h}$. Formally, we parameterize the GLMI $f(\mathbf{h}_i, \mathbf{m}; \mathbf{\Theta}, \mathbf{G})$, with $\mathbf{\Theta} = \{\mathbf{W}, \mathbf{b}\}$ and $\mathbf{G}$, which consists of a residual transformation and a tensor product operation. Specifically, we firstly incorporate the global memory information $\mathbf{m}$ into each local position with a residual transformation as $\tilde{\mathbf{h}}_i = \mathbf{h}_i + \text{ReLU}(\mathbf{W}[\mathbf{h_i}:\mathbf{m}] + \mathbf{b})$, where $[:]$ denotes the vector concatenation. As such, the global memory can distill more correlated local information and they are mapped into the same space. Then we compute a correlation vector $\mathbf{r}_i \in \mathbb{R}^K$ that encodes the strength of correlations between the global memory $\mathbf{m}$ and the transformed local memory $\tilde{\mathbf{h}}_i$ through a tensor product operation as:

$$\mathbf{r}_i = \mathbf{m}^T \mathbf{G} \tilde{\mathbf{h}}_i^{\mathcal{B}},$$

where the tensor $\mathbf{G} \in \mathbb{R}^{\dim_h \times \dim_h \times K}$ can be seen as multiple bilinear matrice that model $K$ kinds of latent relations between two objects. The $k$-th slice of the $\mathbf{G}$, i.e., $\mathbf{G}_k \in \mathbb{R}^{\dim_h \times \dim_h}$ denotes one type of latent relation that interacts with 2 vectors to constitute one type of composition.

### 3.3 Dual Memory Interaction

Following the notations in Section 3.2, we further define a global aspect memory $\mathbf{m}_a \in \mathbb{R}^{\dim_h^{\mathcal{B}}}$, a

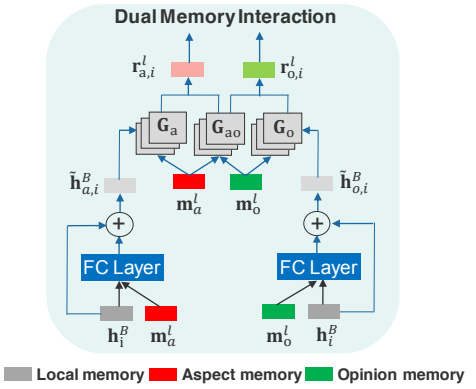


Figure 2: The Dual Memory Interaction (DMI).

global opinion memory $\mathbf{m}_o \in \mathbb{R}^{\dim_h^{\mathcal{B}}}$, and LSTM$^{\mathcal{B}}$ hidden states $\mathbf{H}^{\mathcal{B}} = \{\mathbf{h}_i^{\mathcal{B}}\}_{i=1}^{T}$ as the local memories. The global aspect and opinion memories are able to capture highly correlated aspect or opinion words from the local memories, respectively. Based on the observation that aspect words are often collocated with opinion words across domains, thus their associations can act as the pivot information to bridge the domain gaps. To automatically capture their latent relations within the sentences, at the $l$-th hop, each local memory $\mathbf{h}_i^{\mathcal{B}}$ will interact with the global memories $\mathbf{m}_a^l$ and $\mathbf{m}_o^l$ by the Dual Memory Interaction (DMI) shown in Figure 2, to produce two correlation vectors for aspect and opinion co-detection:

$$\mathbf{r}_{a,i}^l = [f(\mathbf{h}_i^{\mathcal{B}}, \mathbf{m}_a^l; \mathbf{\Theta}_a, \mathbf{G}_a) : f(\mathbf{h}_i^{\mathcal{B}}, \mathbf{m}_o^l; \mathbf{\Theta}_o, \mathbf{G}_{ao})],$$
$$\mathbf{r}_{o,i}^l = [f(\mathbf{h}_i^{\mathcal{B}}, \mathbf{m}_o^l; \mathbf{\Theta}_o, \mathbf{G}_o) : f(\mathbf{h}_i^{\mathcal{B}}, \mathbf{m}_a^l; \mathbf{\Theta}_a, \mathbf{G}_{ao}^{\top})],$$

where $\mathbf{G}_a, \mathbf{G}_o$ and $\mathbf{G}_{ao}$ denote the composition tensors of modeling the latent relations between aspect and aspect, opinion and opinion, and aspect and opinion, respectively.

The correlation vector measures the association strength between local and global memories; e.g., If $\mathbf{h}_i^{\mathcal{B}}$ for the word $w_i$ is both highly intra-correlated with the aspect memory $\mathbf{m}_a$ and inter-correlated with the opinion memory $\mathbf{m}_o$, $w_i$ is more likely to be an aspect term. Then the two correlation vectors can be transformed to a scalar **a**spect attention (**A**-attention) and **o**pinion attention (**O**-attention) weight $\alpha_{p,i}^l$, respectively, with $p \in \{a, o\}$ denoting the aspect or opinion, which indicates the possibility of each word in the sentence being an aspect word or an opinion word as:

$$\alpha_{p,i}^l = \frac{\exp(\mathbf{W}_p \mathbf{r}_{p,i}^l)}{\sum_{j=1}^{T} \exp(\mathbf{W}_p \mathbf{r}_{p,j}^l)},$$

where $\mathbf{W}_p$ is the weight of the attention module. The aspect or opinion attention weight $\alpha_{p,i}^l$ will summarize the local memories to update the global aspect and opinion memories, respectively, for the next hop, i.e., $\mathbf{m}_p^{l+1} = \mathbf{m}_p^l + \sum_{i=1}^{T} \alpha_{p,i}^l \mathbf{h}_i^{\mathcal{B}}$. The updates gradually refine the global memories to incorporate more relevant candidates based on the attention mechanism. In the DMI, all parameters are shared in different hops and domains.

At the final $L$-th hop, we use $\mathbf{r}_{a,i}^L$ for the AD task and feed it to the LSTM$^{\mathcal{U}}$ for the ADS task. For the auxiliary opinion detection task, we feed $\mathbf{r}_{o,i}^L$ into a softmax layer for predicting the probability scores $\mathbf{z}_i^{\mathcal{O}} \in \mathbb{R}^{|\mathcal{Y}^{\mathcal{O}}|}$ over the opinion labels[2] $\mathcal{Y}^{\mathcal{O}}$, i.e., a word is an opinion word or not, as:

$$\mathbf{z}_i^{\mathcal{O}} = \mathbf{p}(\mathbf{y}_i^{\mathcal{O}} | \mathbf{r}_{o,i}^L) = \mathrm{Softmax}(\mathbf{W}_{\mathcal{O}} \mathbf{r}_{o,i}^L + \mathbf{b}_{\mathcal{O}}).$$

### 3.4 Selective Adversarial Learning

To adapt the captured relations to be domain-invariant, we propose a Selective Adversarial Learning (SAL) method to dynamically align the words with high probability to fall into the aspect boundaries, i.e., being an aspect word. Specifically, we introduce a domain discriminator for each word, which aims to identify the domain label $\mathbf{y}_i^{\mathcal{D}} \in \mathbb{R}^{|\mathcal{Y}^{\mathcal{D}}|}$ of the input word, i.e., the word in the sentence is from the source or the target domain. While the feature extractor is to produce the domain-invariant correlation vector $\mathbf{r}_{a,i}^L$ that cannot be distinguished by the domain discriminator via a Gradient Reversal Layer (GRL) (Ganin et al., 2016). Mathematically, we formulate the GRL as a 'pseudo-function' $R_\lambda(\mathbf{x}) = \mathbf{x}$ with a reversal gradient $\frac{\partial R_\lambda(\mathbf{x})}{\partial \mathbf{x}} = -\lambda I$, where $\lambda$ is the adaptation rate. The correlation vector $\mathbf{r}_{a,i}^L$ will be fed to the GRL before the domain discriminator, which is used to predict the probability scores $\mathbf{z}_i^{\mathcal{D}} \in \mathbb{R}^{|\mathcal{Y}^{\mathcal{D}}|}$ over the domain labels $\mathcal{Y}^{\mathcal{D}}$ as:

$$\mathbf{z}_i^{\mathcal{D}} = \mathbf{p}(\mathbf{y}_i^{\mathcal{D}} | \mathbf{r}_{a,i}^L) = \mathrm{Softmax}(\mathbf{W}_{\mathcal{D}} R_\lambda(\mathbf{r}_{a,i}^L) + \mathbf{b}_{\mathcal{D}}).$$

And meanwhile, the aspect attention weight $\alpha_{a,i}^L$ at the final hop serves as a selector to be a learnable alignment weight for each word. Thus, the selective domain adversarial loss is a weighted cross-entropy loss $\ell$ for all the words from the labeled source data $D_s$ and unlabeled target data $D_t$:

$$\mathcal{L}_{\mathcal{D}} = \sum_{D_s \cup D_t} \sum_{i=1}^{T} \alpha_{a,i}^L \ell(\mathbf{z}_i^{\mathcal{D}}, \mathbf{y}_i^{\mathcal{D}}). \quad (1)$$

[2]The opinion lexicon (http://mpqa.cs.pitt.edu/) is used to provide opinion labels for both domains.

Existing studies (Yosinski et al., 2014; Mou et al., 2016) have already shown some evidence that low-level neural layer features (i.e., low-level task) are more easily transferred to different tasks or domains. Thus, we choose the $\mathbf{r}_{a,i}^{L}$ from the low-level AD task to be aligned instead of the feature $\mathbf{h}_i^{\mathcal{U}}$ from the high-level ADS task to transfer. Our ablation studies also confirm this assumption.

### 3.5 Alternating Training

The primary task loss consists of the cross-entropy losses $\ell$ for both the guided AD and main ADS tasks for the labeled source data $D_s$:

$$\mathcal{L}_{\mathcal{M}} = \sum_{D_s} \sum_{\mathcal{Q}\in\{\mathcal{B},\mathcal{U}\}} \sum_{i=1}^{T} \ell(\mathbf{z}_i^{\mathcal{Q}}, \mathbf{y}_i^{\mathcal{Q}}). \quad (2)$$

The auxiliary opinion detection loss is the cross-entropy loss for the labeled source data $D_s$ and unlabeled target data $D_t$ as follows:

$$\mathcal{L}_{\mathcal{O}} = \sum_{D_s \cup D_t} \sum_{i=1}^{T} \ell(\mathbf{z}_i^{\mathcal{O}}, \mathbf{y}_i^{\mathcal{O}}). \quad (3)$$

Traditionally, we can directly optimize the joint loss of Eqs. (1)-(3), i.e., $E = \mathcal{L}_{\mathcal{M}} + \rho\mathcal{L}_{\mathcal{O}} + \gamma\mathcal{L}_{\mathcal{D}}$ to obtain both discriminative and domain-invariant word representations, where $\rho$ and $\gamma$ are the trade-off factors. However, we found the optimization process tends to be unstable since it may be hard to jointly optimize many objectives. Thus, we propose an empirically alternating strategy to train the $\mathcal{L}_{\mathcal{M}} + \rho\mathcal{L}_{\mathcal{O}}$ and $\mathcal{L}_{\mathcal{D}}$ iteratively, which separates the whole word representation learning into a *discriminative* stage and a *domain-invariant* stage. Let $\boldsymbol{\theta}_f$, $\boldsymbol{\theta}_w$, $\boldsymbol{\theta}_d$ denote the parameters for feature learning of each word, word predictors for AD, ADS and opinion detection tasks, and domain discriminators, respectively. Based on our strategy, we are seeking the parameters $(\hat{\boldsymbol{\theta}}_f^{(1)}, \hat{\boldsymbol{\theta}}_f^{(2)}, \hat{\boldsymbol{\theta}}_w, \hat{\boldsymbol{\theta}}_d)$ that deliver a saddle point of $E$ among two stages:

$$(\hat{\boldsymbol{\theta}}_f^{(1)}, \hat{\boldsymbol{\theta}}_w) = \arg\min_{\boldsymbol{\theta}_f, \boldsymbol{\theta}_w} \mathcal{L}_{\mathcal{M}} + \rho\mathcal{L}_{\mathcal{O}}$$

$$(\hat{\boldsymbol{\theta}}_f^{(2)}, \hat{\boldsymbol{\theta}}_d) = \arg\min_{\boldsymbol{\theta}_d} \max_{\boldsymbol{\theta}_f} \mathcal{L}_{\mathcal{D}}.$$

At the saddle point, the feature learning parameters $\boldsymbol{\theta}_f$ minimize the word prediction losses (i.e., the features are discriminative) for the first stage. For the second stage, the domain classification loss is minimized by the domain discriminator parameters $\boldsymbol{\theta}_d$ while maximized by the feature learning parameters $\boldsymbol{\theta}_f$ via GRL (i.e., the features are domain-invariant). As such, we can achieve easier and more stable optimization for feature learning.

| Dataset | Domain | Sentences | Training | Testing |
|---|---|---|---|---|
| $\mathbb{L}$ | Laptop | 1,869 | 1,458 | 411 |
| $\mathbb{R}$ | Restaurant | 3,900 | 2,481 | 1,419 |
| $\mathbb{D}$ | Device | 1,437 | 954 | 483 |
| $\mathbb{S}$ | Service | 2,153 | 1,433 | 720 |

Table 1: Statistics of the datasets.

## 4 Experiments

### 4.1 Experimental Setup

**Datasets:** Our experiments are conducted on four benchmark datasets: Laptop ($\mathbb{L}$), Restaurant ($\mathbb{R}$), Device ($\mathbb{D}$), and Service ($\mathbb{S}$). $\mathbb{L}$ contains reviews from the laptop domain in SemEval ABSA challenge 2014 (Pontiki et al., 2014). Following the setup in (Li et al., 2019a), $\mathbb{R}$ is the union set of the restaurant datasets from SemEval ABSA challenge 2014, 2015, and 2016 (Pontiki et al., 2014, 2015, 2016). $\mathbb{D}$ is a combination of device reviews from 5 different digital products provided by (Hu and Liu, 2004). $\mathbb{S}$ is introduced by (Toprak et al., 2010) and contains reviews from web services. Detailed statistics are shown in Table 1.

**Settings:** We construct 10 transfer pairs like $D_s \rightarrow D_t$ with the four domains mentioned above, and we do not use the pairs $\mathbb{L}\rightarrow\mathbb{D}$ and $\mathbb{D}\rightarrow\mathbb{L}$ as these two domains are very similar. Note that for the unsupervised domain adaptation setting, no labels are available for the target domain. Therefore, for each transfer pair, its training dataset is the combination of the labeled training data of the source domain and the unlabeled training data of the target domain. Meanwhile, it employs the testing data of the source domain with labels as the validation set and the testing data of the target domain as the evaluation set. We report the results for both AD and ADS tasks. The evaluation metric is the Micro-F1 score under the **exact** match, which means that an output segment is considered to be correct only if it exactly matches with the gold standard span of the aspect term for the AD task or the aspect term and its sentiment for the ADS task. All experiments are repeated 5 times and we report the average results over 5 runs.

### 4.2 Implementation details

The word embeddings are 100-dimensional *word2vec* (Mikolov et al., 2013) vectors pre-trained on the combination of the Yelp Challenge

| Transfer Pair | TCRF | | RAP | | Hier-Joint | | Hier-Joint$^+$ | | RNSCN | | RNSCN$^+$ | | **Ours** | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | AD | ADS | AD | ADS | AD | ADS | AD | ADS | AD | ADS | AD | ADS | AD | ADS |
| $\mathbb{S}\rightarrow\mathbb{R}$ | - | 14.84 | - | 25.41 | - | 32.81 | 46.39 | 31.10 | - | 30.56 | 48.89 | 33.21 | **52.05** | **41.03** |
| $\mathbb{L}\rightarrow\mathbb{R}$ | - | 16.06 | - | 31.05 | - | 31.90 | 48.61 | 33.54 | - | 31.85 | 52.19 | 35.65 | **56.12** | **43.04** |
| $\mathbb{D}\rightarrow\mathbb{R}$ | - | 17.05 | - | 28.37 | - | 30.03 | 42.96 | 32.87 | - | 31.41 | 50.39 | 34.60 | **51.55** | **41.01** |
| $\mathbb{R}\rightarrow\mathbb{S}$ | - | 15.20 | - | 13.17 | - | 15.20 | 27.18 | 15.56 | - | 23.31 | 30.41 | 20.04 | **39.02** | **28.01** |
| $\mathbb{L}\rightarrow\mathbb{S}$ | - | 12.34 | - | 13.72 | - | 15.33 | 25.22 | 13.90 | - | 16.73 | 31.21 | 16.59 | **38.26** | **27.20** |
| $\mathbb{D}\rightarrow\mathbb{S}$ | - | 13.49 | - | 16.80 | - | 18.74 | 29.28 | 19.04 | - | 18.93 | 35.50 | 20.03 | **36.11** | **26.62** |
| $\mathbb{R}\rightarrow\mathbb{L}$ | - | 14.59 | - | 15.69 | - | 19.17 | 34.11 | 20.72 | - | 25.54 | **47.23** | 26.63 | 45.01 | **34.13** |
| $\mathbb{S}\rightarrow\mathbb{L}$ | - | 9.56 | - | 12.38 | - | 21.80 | 33.02 | 22.65 | - | 19.15 | 34.03 | 18.87 | **35.99** | **27.04** |
| $\mathbb{R}\rightarrow\mathbb{D}$ | - | 19.84 | - | 17.50 | - | 22.91 | 34.81 | 24.53 | - | 32.43 | **46.16** | 33.26 | 43.76 | **35.44** |
| $\mathbb{S}\rightarrow\mathbb{D}$ | - | 13.43 | - | 15.74 | - | 20.04 | 35.00 | 23.24 | - | 19.98 | 32.41 | 22.00 | **41.21** | **33.56** |
| Average | - | 14.64 | - | 18.98 | - | 22.79 | 35.66 | 23.72 | - | 24.99 | 40.84 | 26.09 | **43.91**$^\dagger$ | **33.71**$^\dagger$ |
| ($\Delta$) | - | (19.07) | - | (14.73) | - | (10.92) | (8.25) | (9.99) | - | (8.72) | (3.07) | (7.62) | - | - |

Table 2: Main results (%). $\Delta$ refers to the improvements of the full model over baseline methods. The marker $^\dagger$ means that our model significantly outperforms the best baseline **RNSCN**$^+$ with $p$-value $< 0.01$.

dataset[3] and the electronics dataset from Amazon reviews[4]. For out-of-vocabulary words, we randomly initialized them with a uniform distribution $U(-0.25, 0.25)$. The dimensions of two LSTM layers $\dim_h^{\mathcal{B}}$ and $\dim_h^{\mathcal{U}}$ are all set to 100. The number of hops $L$ is set to 2. The number of bilinear interactions $K$ is set to 50. The weight matrices are initialized with a uniform distribution $U(-0.2, 0.2)$. The adaptation rate $\lambda$ is 0.1 and the trade-off factor $\rho$ is 1.0. For training, the model is optimized by the Adam algorithm (Kingma and Ba, 2014) with the initial learning rate 0.001. The batch size is 64, with a half coming from the source and target domains, respectively. Gradients with the $\ell_2$ norm larger than 40 are normalized to be 40. To alleviate the overfitting, we apply the dropout on the word embeddings $\mathbf{e}_i$ and the learned word representations $\mathbf{r}_{a,i}^l$, $\mathbf{r}_{o,i}^l$, and $\mathbf{h}_i^{\mathcal{U}}$ with dropout rate 0.5. We use the same hyper-parameters, which are tuned on 10% randomly held-out training data of the source domain in $\mathbb{R}\rightarrow\mathbb{L}$, for all transfer pairs.

### 4.3 Baselines

We compare with several state-of-the-art fine-grained adaptation methods.

- **TCRF** (Jakob and Gurevych, 2010): Transferable CRF that uses a linear-chain CRF for sequence prediction based on shared non-lexical features across domains, e.g., POS tags and dependency relations.
- **RAP** (Li et al., 2012): A cross-domain Relational Adaptive Bootstrapping method that iteratively expands target aspect and opinion lexicons according to common opinion words and syntactic relations.
- **Hier-Joint** (Ding et al., 2017): A recurrent neural network (RNN) with manually designed rule-based auxiliary tasks based on common syntactic relations among aspect and opinion words.
- **RNSCN** (Wang and Pan, 2018): a recursive neural structural correspondence network that incorporates syntactic structures and exploits an auto-encoder to denoise relation labels generated from the parser.

As the first to address cross-domain E2E-ABSA, we have to adapt all the baselines which are originally proposed for cross-domain aspect detection, or aspect and opinion co-detection to return the ADS results by replacing their *aspect boundary tags* with the *unified tags*. Absent of the proposed stacking architecture, all the baselines fail to accomplish the auxiliary AD task meantime. Thus, we only report their ADS results. Besides, E2E-ABSA aims to simultaneously learn aspect terms along with their sentiments. Thus, the ADS is exactly our main task while the AD is only an auxiliary task used for the guidance.

To be more convincing, we extend neural models (i.e., Hier-Joint and RNSCN) to more powerful baselines named **Hier-Joint**$^+$ and **RNSCN**$^+$ with the proposed stacking architecture, respectively. Both of them stack an additional RNN layer on top of the original framework to produce the *unified tags* while the lower layer is to predict the *aspect boundary tags* as the guidance. The validity of such extensions is guaranteed by the fact that the extended versions achieve even better AD performances than the original versions. We use the source code of the baselines for experiments. For

[3] http://www.yelp.com/datasetchallenge

[4] http://jmcauley.ucsd.edu/data/amazon/links.html

| Transfer Pair | Lower bound | | Ablation Models | | | | | | Full Model | | Upper bound | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Base Model (SO) | | Base Model+DMI | | AD-AL | | ADS-SAL | | AD-SAL | | Base Model (TO) | |
| | AD | ADS | AD | ADS | AD | ADS | AD | ADS | AD | ADS | AD | ADS |
| $\mathbb{S}\rightarrow\mathbb{R}$ | 30.32 | 19.74 | 45.68 | 37.10 | 48.28 | 37.65 | 51.29 | 41.03 | **52.05** | **41.03** | 81.84 | 67.26 |
| $\mathbb{L}\rightarrow\mathbb{R}$ | 33.99 | 28.34 | 46.25 | 36.49 | 51.79 | 38.63 | 55.50 | 42.00 | **56.12** | **43.04** | | |
| $\mathbb{D}\rightarrow\mathbb{R}$ | 31.59 | 27.25 | 46.56 | 36.89 | 46.39 | 37.34 | 46.43 | 38.35 | **51.55** | **41.01** | | |
| $\mathbb{R}\rightarrow\mathbb{S}$ | 15.63 | 8.61 | 21.88 | 16.85 | 25.13 | 18.61 | 37.11 | 25.84 | **39.02** | **28.01** | 68.28 | 41.12 |
| $\mathbb{L}\rightarrow\mathbb{S}$ | 22.45 | 16.07 | 28.67 | 21.53 | 28.18 | 20.74 | 30.35 | 23.73 | **38.26** | **27.20** | | |
| $\mathbb{D}\rightarrow\mathbb{S}$ | 16.79 | 9.49 | 31.91 | 22.14 | 32.88 | 24.89 | 32.51 | 21.45 | **36.11** | **26.62** | | |
| $\mathbb{R}\rightarrow\mathbb{L}$ | 38.45 | 23.40 | 42.27 | 30.52 | 40.52 | 28.77 | 44.56 | 33.34 | **45.01** | **34.13** | 75.95 | 52.62 |
| $\mathbb{S}\rightarrow\mathbb{L}$ | 24.69 | 14.48 | 36.38 | 27.48 | 32.96 | 25.16 | 33.87 | 24.22 | **35.99** | **27.04** | | |
| $\mathbb{R}\rightarrow\mathbb{D}$ | 34.87 | 25.79 | 36.90 | 27.71 | 41.61 | 31.88 | **43.97** | 34.50 | 43.76 | **35.44** | 70.37 | 57.62 |
| $\mathbb{S}\rightarrow\mathbb{D}$ | 27.73 | 17.73 | 38.03 | 31.21 | 39.54 | 32.28 | 40.40 | 33.26 | **41.21** | **33.56** | | |
| Average | 27.65 | 19.09 | 37.45 | 28.79 | 38.73 | 29.60 | 41.60 | 31.77 | **43.91**$^{\dagger}$ | **33.71**$^{\dagger}$ | 74.11 | 54.66 |
| ($\Delta$) | (16.26) | (14.62) | (6.46) | (4.92) | (5.18) | (4.11) | (2.31) | (1.94) | - | - | - | - |

Table 3: Ablation results (%). $\Delta$ refers to the improvements of the full model over ablation methods. The marker $^{\dagger}$ means that the full model significantly outperforms the best ablation model **ADS-SAL** with $p$-value $< 0.01$.

fair comparison, all baselines use the same pre-trained word embeddings and the baselines that require opinion labels use the same opinion lexicon.

### 4.4 Main Results

Based on the results in Table 2, we have the following observations:

- Our model consistently and significantly achieves the best results on almost all transfer pairs, outperforming the strongest baseline RNSCN$^{+}$ by 3.07% and 7.62% Micro-F1 on average for the AD and ADS tasks, respectively. Our model can automatically model complicated relations among aspect and opinion words via the DMI as transferable knowledge. Besides, the proposed SAL method can dynamically learn an alignment weight for each word to achieve a local semantic alignment, which distills a better shared feature space and further improves the performances.

- Traditional non-neural methods like TCRF and RAP perform very poorly due to the reliance on hand-crafted features. Our model outperforms Hier-Joint and RNSCN, which are neural models, by 10.92% and 8.72% Micro-F1 on average for the ADS task, respectively. Both of them rely on the dependency parser to exploit syntactic relations, which are inflexible due to the no end-to-end manner and may bring in external errors.

- The extended version Hier-Joint$^{+}$ and RNSCN$^{+}$ can further improve the performances, which shows the benefits of the guidance from the low-level AD task. However, our model can still outperform them by a large margin, which demonstrates the effectiveness of the proposed methods.

### 4.5 Ablation Study

To investigate the effectiveness of each component, we conduct the ablation study to compare our full model with different ablation variants:

- **Base Model (SO / TO)**: it uses two stacked Bi-LSTMs as the Base Model. SO (Source Only) and TO (Target Only) denote that the base model is only trained on the labeled data from the source and target domain, respectively. We usually refer to them as a lower bound and a upper bound, respectively.

- **Base Model+DMI**: it uses two stacked Bi-LSTMs with a multi-hop dual memory interaction (DMI) between them.

- **AD-AL**: it performs pure adversarial learning (removing the selective weight $\alpha^{L}_{a,i}$ from the Eq. (1)) on each correlation vector $\mathbf{r}^{L}_{a,i}$ for the low-level AD task.

- **AD-SAL**: it advances the AD-AL by conducting selective adversarial learning.

- **ADS-SAL**: it conducts selective adversarial learning on each word representation $\mathbf{h}^{\mathcal{U}}_{i}$ for the high-level ADS task.

Note that, the AD-AL, AD-SAL (**Full model**) and ADS-SAL all use the same architecture as the Base Model+DMI. Based on the Table 3, we have the following observations to give us evidences:

- **No DMI** v.s. **DMI**: Base Model+DMI outperforms the Base Model (SO) by 9.80% and 9.70% Micro-F1 on average for the AD and ADS tasks, respectively. This demonstrates

| Input: (Target domain $\mathbb{L}$) | Base model+DMI | | AD-AL | | AD-SAL | |
|---|---|---|---|---|---|---|
| | AD | ADS | AD | ADS | AD | ADS |
| 1. This laptop has only 2 [*usb ports*]$_{\text{NEG}}$ , and they are both on the same side . | *ports*(✗), *side* (✗) | NONE(✗) | NONE(✗) | NONE(✗) | *usb ports* | [*usb ports*]$_{\text{NEG}}$ |
| 2. It is very easy to integrate [*bluetooth devices*]$_{\text{POS}}$ , and [*usb devices*]$_{\text{POS}}$ are recognized almost instantly . | *devices* (✗), *devices* (✗) | [*devices*]$_{\text{POS}}$ (✗), [*devices*]$_{\text{POS}}$ (✗) | NONE(✗) | NONE(✗) | *bluetooth devices*, *usb devices* | [*bluetooth devices*]$_{\text{POS}}$, [*usb devices*]$_{\text{POS}}$ |
| 3. I also wanted [*windows 7*]$_{\text{POS}}$ , which this one has . | NONE(✗) | NONE(✗) | NONE(✗) | NONE(✗) | *windows 7* | [*windows 7*]$_{\text{POS}}$ |
| 4. The [*speed*]$_{\text{POS}}$ , the [*simplicity*]$_{\text{POS}}$ , the [*design*]$_{\text{POS}}$ it is lightyears ahead of any pc i have ever owned . | *speed*, *design* | [*speed*]$_{\text{POS}}$, [*design*]$_{\text{POS}}$ | *speed*, *design*, *pc* (✗) | [*speed*]$_{\text{POS}}$, [*design*]$_{\text{POS}}$, [*pc*]$_{\text{POS}}$ (✗) | *speed*, *design*, *simplicity* | [*speed*]$_{\text{POS}}$, [*design*]$_{\text{POS}}$, [*simplicity*]$_{\text{POS}}$ |
| 6. The [*battery life*]$_{\text{POS}}$ is excellent , the [*display*]$_{\text{POS}}$ is excellent and [*downloading apps*]$_{\text{POS}}$ is a breeze . | *battery* (✗), *display*, *apps* (✗) | [*battery*]$_{\text{POS}}$ (✗), [*display*]$_{\text{POS}}$, [*apps*]$_{\text{POS}}$ (✗) | *battery* (✗), *display*, *apps* (✗) | [*battery*]$_{\text{POS}}$ (✗), [*display*]$_{\text{POS}}$, [*apps*]$_{\text{POS}}$ (✗) | *battery life*, *display*, *downloading apps* | [*battery life*]$_{\text{POS}}$, [*display*]$_{\text{POS}}$, [*downloading apps*]$_{\text{POS}}$ |

Table 4: Case analysis for the $\mathbb{R}\rightarrow\mathbb{L}$ pair. Note that we only show the sentiment part of the unified labels (i.e., POS, NEG, and NEU) and use brackets to indicate the boundary. The marker ✗ denotes an incorrect prediction.

that the original word hidden representations (LSTM$^{\mathcal{B}}$ hidden states) are not suitable for transfer. Thus, we need to resort to the correlation vectors inferred by the DMI that models the transferable latent relations between aspect and opinion words.

- **No SAL** v.s. **SAL**: AD-SAL significantly and consistently exceeds Base Model+DMI by 6.46% and 4.92% Micro-F1 on average for the AD and ADS tasks, respectively. Without any adaptation, the captured relations by the DMI may not work well across domains, while the proposed SAL method can effectively align these latent relations to be domain-invariant.

- **No Selectivity** v.s. **Selectivity**: AD-SAL outperforms AD-AL by 5.18% and 4.11% Micro-F1 on average for the AD and ADS tasks, respectively. This proves the necessity to conduct the selective alignment. The SAL method can dynamically learn to control an alignment weight for each word to achieve a local semantic alignment, which captures a better domain-invariant feature space than pure adversarial learning that treats all words equally for the fine-grained adaptation.

- **Low-level** v.s. **High-level**: AD-SAL exceeds ADS-SAL by 2.31% and 1.94% Micro-F1 on average for the AD and ADS tasks, respectively. The label space of the *unified tags* for the high-level ADS task is more complicated than that of the *aspect boundary tag* for the low-level AD task. This gives the evidence that low-level neural features are more easily to transfer than high-level features.

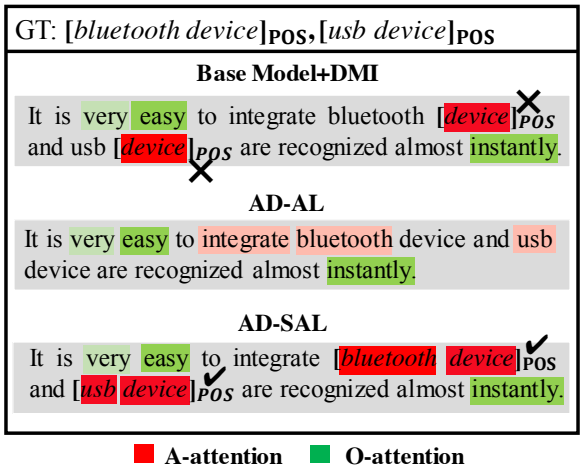


Figure 3: Visualization of attention for the $\mathbb{R}\rightarrow\mathbb{L}$ pair.

### 4.6 Case Analysis

As illustrated in Table 4, we perform case analysis of the $\mathbb{R}\rightarrow\mathbb{L}$ pair for the Base model+DMI, AD-AL, and the full model AD-SAL to demonstrate the necessity to conduct selective alignment for the fine-grained adaptation. The Base model+DMI can identify some domain-specific aspect words (e.g., *battery*, *ports*) without any supervision from the target domain. This is because the DMI can infer relational representations that capture some common latent relations between aspect and opinion words. However, it still cannot completely capture the multi-word aspect terms (e.g. *bluetooth device*, *battery life*) and sometimes it totally ignores them (e.g. *window 7*). The AD-AL performs pure adversarial learning for aligning all words in a sentence. Even though the domain adaptation method is adopted, it does not yield significant improvements and sometimes it becomes worse. For example, the AD-AL cannot even identify aspect words that can be captured

by the Base Model+DMI (e.g., *ports*, *devices*), or wrongly identifies some non-aspect words (e.g., *pc*). The reason is that pure adversarial learning treats all the words equally for the alignment, which may bring in noises of uninformative words into the shared feature space. To solve that, the full model AD-SAL performs a local semantic alignment to dynamically focus on aligning aspect words that contribute more to the domain-invariant feature space. Hence, AD-SAL model can precisely and completely identity all the aspect terms and make correct unified tag predictions.

Moreover, in Figure 3, we visualize the attentions from these models, where deeper colors denote larger weights. Compared with the Base model+DMI, the full model AD-SAL can precisely attend the complete aspect words from the target domain (A-attention), i.e., *bluetooth device* and *usb device*, and make correct predictions, while the AD-AL cannot achieve that. The AD-AL can only align all the words equally, which hinders the model to attend the aspect words, while the A-attention of the AD-SAL model can be used for both discriminative word tags predictions and acting as a learnable alignment weight for each word. This shows that the proposed SAL method can learn to align important aspect words to improve the transferability of the model for the fine-grained adaptation.

## 5 Related Works

E2E-ABSA can be broken into two sub-tasks, namely, aspect detection and aspect sentiment classification. The aspect detection aims to extract the aspect terms mentioned in the text and it has been actively studied (Qiu et al., 2011; Liu et al., 2015; Poria et al., 2016; Wang et al., 2016a; He et al., 2017; Wang et al., 2017; Majumder et al., 2018; Li et al., 2018b; Xu et al., 2018). The aspect sentiment classification is to predict the sentiment polarities of the given aspect terms and has also received a lot of attention recently (Dong et al., 2014; Tang et al., 2016; Wang et al., 2016b; Ma et al., 2017; Chen et al., 2017; Ma et al., 2018; He et al., 2018b; Li et al., 2018a, 2019b). For practical applications, a typical way is to pipeline these two sub-tasks together, which becomes ineffective due to the accumulated errors across tasks. These two sub-tasks have strong couplings and thus a unified formulation (Mitchell et al., 2013; Zhang et al., 2015; Li et al., 2019a) to handle them together in an end-to-end manner becomes a more promising direction. Despite its importance, existing studies are only exploring the performance in a single domain, while ignoring the transferability across domains. To address this problem, unsupervised domain adaptation methods can be applied. While existing methods focus on traditional cross-domain sentiment classification to learn shared representations for sentences or documents, including pivot-based methods (Blitzer et al., 2007; Pan et al., 2010; Bollegala et al., 2013; Yu and Jiang, 2016), auto-encoders (Glorot et al., 2011; Chen et al., 2012; Zhou et al., 2016), domain adversarial networks (Ganin et al., 2016; Li et al., 2017, 2018c), or semi-supervised methods (He et al., 2018a). Due to the difficulties in fine-grained adaptation, there exist very few methods for cross-domain aspect extraction (Jakob and Gurevych, 2010; Ding et al., 2017), which acts as a sub-task of E2E-ABSA, or aspect and opinion co-extraction (Li et al., 2012; Wang and Pan, 2018) that focuses on detecting aspect and opinion words, while E2E-ABSA needs to analyze more complicated correspondences between them. Besides, those methods can only rely on general syntactic relations between aspect and opinion words to bridge the domains. Different from them, our method leverages the attention mechanism (Bahdanau et al., 2014; Sukhbaatar et al., 2015; Shen et al., 2018, 2019) as a dynamic selector to automatically achieve the selective alignment.

## 6 Conclusion

The effectiveness of supervised methods for E2E-ABSA is limited due to the data scarcity. Our wok is the first attempt to resolve cross-domain E2E-ABSA by leveraging knowledge from other related domains to enhance the sequence learning of the target domain. Extensive experiments show the effectiveness of the proposed SAL method. Ablation studies also prove the necessity to perform selective alignment. In the future, the proposed SAL method can be potentially extended to other domain adaptation methods and applied to more general sequence labeling tasks including named entity recognition (Zhou et al., 2019c), part-of-speech tagging (Zhou et al., 2019a), etc.

## Acknowledgement

We thank the support of Hong Kong CERG grants (16209715 & 16244616) and NSFC 61673202.

## References


Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*, pages 440–447.

Danushka Bollegala, David Weir, and John Carroll. 2013. Cross-domain sentiment classification using a sentiment sensitive thesaurus. *TKDE*, pages 1719–1731.

Minmin Chen, Zhixiang Xu, Fei Sha, and Kilian Q Weinberger. 2012. Marginalized denoising autoencoders for domain adaptation. In *ICML*, pages 1627–1634.

Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. 2017. Recurrent attention network on memory for aspect sentiment analysis. In *EMNLP*, pages 452–461.

Ying Ding, Jianfei Yu, and Jing Jiang. 2017. Recurrent neural networks with auxiliary labels for cross-domain opinion target extraction. In *AAAI*, pages 3436–3442.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *ACL*, pages 49–54.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *JMLR*, pages 2096–2030.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *ICML*, pages 513–520.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2017. An unsupervised neural attention model for aspect extraction. In *ACL*, pages 388–397.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2018a. Adaptive semi-supervised learning for cross-domain sentiment classification. In *EMNLP*, pages 3467–3476.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2018b. Exploiting document knowledge for aspect-level sentiment classification. In *ACL*, pages 579–585.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *KDD*, pages 168–177.

Niklas Jakob and Iryna Gurevych. 2010. Extracting opinion targets in a single-and cross-domain setting with conditional random fields. In *EMNLP*, pages 1035–1045.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *ICML*, pages 1378–1387.

Fangtao Li, Sinno Jialin Pan, Ou Jin, Qiang Yang, and Xiaoyan Zhu. 2012. Cross-domain co-extraction of sentiment and topic lexicons. In *ACL*, pages 410–419.

Xin Li, Lidong Bing, Wai Lam, and Bei Shi. 2018a. Transformation networks for target-oriented sentiment classification. In *ACL*, pages 946–956.

Xin Li, Lidong Bing, Piji Li, and Wai Lam. 2019a. A unified model for opinion target extraction and target sentiment prediction. In *AAAI*, pages 6714–6721.

Xin Li, Lidong Bing, Piji Li, Wai Lam, and Zhimou Yang. 2018b. Aspect term extraction with history attention and selective transformation. In *IJCAI*, pages 4194–4200.

Zheng Li, Ying Wei, Yu Zhang, and Qiang Yang. 2018c. Hierarchical attention transfer network for cross-domain sentiment classification. In *AAAI*.

Zheng Li, Ying Wei, Yu Zhang, Xiang Zhang, Xin Li, and Qiang Yang. 2019b. Exploiting coarse-to-fine task transfer for aspect-level sentiment classification. In *AAAI*, pages 4253–4260.

Zheng Li, Yu Zhang, Ying Wei, Yuxiang Wu, and Qiang Yang. 2017. End-to-end adversarial memory network for cross-domain sentiment classification. In *IJCAI*, pages 2237–2243.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*.

Pengfei Liu, Shafiq Joty, and Helen Meng. 2015. Fine-grained opinion mining with recurrent neural networks and word embeddings. In *EMNLP*, pages 1433–1443.

Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2017. Interactive attention networks for aspect-level sentiment classification. *arXiv preprint arXiv:1709.00893*.

Yukun Ma, Haiyun Peng, and Erik Cambria. 2018. Targeted aspect-based sentiment analysis via embedding commonsense knowledge into an attentive lstm. In *AAAI*.

Navonil Majumder, Soujanya Poria, Alexander Gelbukh, Md. Shad Akhtar, Erik Cambria, and Asif Ekbal. 2018. IARM: Inter-aspect relation modeling with memory networks in aspect-based sentiment analysis. In *EMNLP*, pages 3402–3411.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Margaret Mitchell, Jacqui Aguilar, Theresa Wilson, and Benjamin Van Durme. 2013. Open domain targeted sentiment. In *EMNLP*, pages 1643–1654.

Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2016. How transferable are neural networks in NLP applications? In *EMNLP*, pages 479–489.

Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2010. Cross-domain sentiment classification via spectral feature alignment. In *WWW*, pages 751–760.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, AL-Smadi Mohammad, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphée De Clercq, et al. 2016. Semeval-2016 task 5: Aspect based sentiment analysis. In *SemEval*, pages 19–30.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Suresh Manandhar, and Ion Androutsopoulos. 2015. Semeval-2015 task 12: Aspect based sentiment analysis. In *SemEval*, pages 486–495.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *SemEval*, pages 27–35.

Soujanya Poria, Erik Cambria, and Alexander F. Gelbukh. 2016. Aspect extraction for opinion mining with a deep convolutional neural network. *Knowl.-Based Syst.*, 108:42–49.

Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. 2011. Opinion word expansion and target extraction through double propagation. *Computational Linguistics*, pages 9–27.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018. Disan: Directional self-attention network for rnn/cnn-free language understanding. In *AAAI*, pages 5446–5455.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Tensorized self-attention: Efficiently modeling pairwise and global dependencies together. In *NAACL*, pages 1256–1266.

Sainbayar Sukhbaatar, arthur szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *NIPS*, pages 2440–2448.

Duyu Tang, Bing Qin, and Ting Liu. 2016. Aspect level sentiment classification with deep memory network. *arXiv preprint arXiv:1605.08900*.

Cigdem Toprak, Niklas Jakob, and Iryna Gurevych. 2010. Sentence and expression level annotation of opinions in user-generated discourse. In *ACL*, pages 575–584.

Wenya Wang and Sinno Jialin Pan. 2018. Recursive neural structural correspondence network for cross-domain aspect and opinion co-extraction. In *ACL*, pages 2171–2181.

Wenya Wang, Sinno Jialin Pan, Daniel Dahlmeier, and Xiaokui Xiao. 2016a. Recursive neural conditional random fields for aspect-based sentiment analysis. In *EMNLP*, pages 616–626.

Wenya Wang, Sinno Jialin Pan, Daniel Dahlmeier, and Xiaokui Xiao. 2017. Coupled multi-layer attentions for co-extraction of aspect and opinion terms. In *AAAI*, pages 3316–3322.

Yequan Wang, Minlie Huang, Li Zhao, et al. 2016b. Attention-based lstm for aspect-level sentiment classification. In *EMNLP*, pages 606–615.

Hu Xu, Bing Liu, Lei Shu, and Philip S. Yu. 2018. Double embeddings and cnn-based sequence labeling for aspect extraction. In *ACL*, pages 592–598.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? In *NIPS*, pages 3320–3328.

Jianfei Yu and Jing Jiang. 2016. Learning sentence embeddings with auxiliary tasks for cross-domain sentiment classification. In *EMNLP*, pages 236–246.

Meishan Zhang, Yue Zhang, and Duy Tin Vo. 2015. Neural networks for open domain targeted sentiment. In *EMNLP*, pages 612–621.

Guangyou Zhou, Zhiwen Xie, Jimmy Xiangji Huang, and Tingting He. 2016. Bi-transferring deep neural networks for domain adaptation. In *ACL*, pages 322–332.

Joey Tianyi Zhou, Hao Zhang, Di Jin, and Xi Peng. 2019a. Dual adversarial transfer for sequence labeling. *TPAMI*, pages 1–1.

Joey Tianyi Zhou, Hao Zhang, Di Jin, Xi Peng, Yang Xiao, and Zhiguo Cao. 2019b. Roseq: Robust sequence labeling. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–11.

Joey Tianyi Zhou, Hao Zhang, Di Jin, Hongyuan Zhu, Meng Fang, Rick Siow Mong Goh, and Kenneth Kwok. 2019c. Dual adversarial neural transfer for low-resource named entity recognition. In *ACL*, pages 3461–3471.

# CAN: Constrained Attention Networks for Multi-Aspect Sentiment Analysis

**Mengting Hu**[1]* **Shiwan Zhao**[2]† **Li Zhang**[2] **Keke Cai**[2]
**Zhong Su**[2] **Renhong Cheng**[1] **Xiaowei Shen**[2]
[1] Nankai University [2] IBM Research - China
mthu@mail.nankai.edu.cn, {zhaosw, lizhang, caikeke, suzhong}@cn.ibm.com
chengrh@nankai.edu.cn, xwshen@cn.ibm.com

## Abstract

Aspect level sentiment classification is a fine-grained sentiment analysis task. To detect the sentiment towards a particular aspect in a sentence, previous studies have developed various attention-based methods for generating aspect-specific sentence representations. However, the attention may inherently introduce noise and downgrade the performance. In this paper, we propose constrained attention networks (CAN), a simple yet effective solution, to regularize the attention for multi-aspect sentiment analysis, which alleviates the drawback of the attention mechanism. Specifically, we introduce orthogonal regularization on multiple aspects and sparse regularization on each single aspect. Experimental results on two public datasets demonstrate the effectiveness of our approach. We further extend our approach to multi-task settings and outperform the state-of-the-art methods.

## 1 Introduction

Sentiment analysis (Nasukawa and Yi, 2003; Liu, 2012), an important task in natural language understanding, receives much attention in recent years. Aspect level sentiment classification (ALSC) is a fine-grained sentiment analysis task, which aims at detecting the sentiment towards a particular aspect in a sentence. ALSC is especially critical for multi-aspect sentences which contain multiple aspects. A multi-aspect sentence can be categorized as *overlapping* or *non-overlapping*. A sentence is annotated as non-overlapping if any two of its aspects have no overlap. Our study found that around 85% of the multi-aspect sentences are non-overlapping in the two public datasets. Figure 1 shows a simple example. The non-overlapping sentence contains two aspects. The aspect *food* is on the left side of the aspect *service*. Their distributions on words are **orthogonal** to each other. Another observation is that only a few words relate to the opinion expression in each aspect. As shown in Figure 1, only the word *"good"* is relevant to the aspect *food* and *"ok"* to *service*. The distribution of the opinion expression of each aspect is **sparse**.




Figure 1: Example of a non-overlapping sentence. The attention weights of the aspect *food* are from the model ATAE-LSTM (Wang et al., 2016).

To detect the sentiment towards a particular aspect, previous studies (Wang et al., 2016; Ma et al., 2017; Cheng et al., 2017; Ma et al., 2018; Huang et al., 2018) have developed various attention-based methods for generating aspect-specific sentence representations. In these works, the attention may inherently introduce noise and downgrade the performance (Li et al., 2018) since the attention scatters across the whole sentence and is prone to attend on noisy words, or the opinion words from other aspects. Take Figure 1 as an example, for the aspect *food*, we visualize the attention weights from the model (Wang et al., 2016). Much of the attention focuses on the noisy word *"sometimes"*, and the opinion word *"ok"* which is relevant to the aspect *service* rather than *food*.

To alleviate the above issue, we propose a model for multi-aspect sentiment analysis, which regularizes the attention by handling multiple aspects of a sentence simultaneously. Specifically, we introduce orthogonal regularization for attention weights among multiple non-overlapping as-

*This work was done when Mengting Hu was a research intern at IBM Research - China.
†Corresponding author.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4601–4610, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

pects. The orthogonal regularization tends to make the attention weights of multiple aspects concentrate on different parts of the sentence with less overlap. We also introduce the sparse regularization, which tends to make the attention weights of each aspect concentrate only on a few words. We call our networks with such regularizations *constrained attention networks* (CAN). There have been some works on introducing sparsity in attention weights in machine translation (Malaviya et al., 2018) and orthogonal constraints in domain adaptation (Bousmalis et al., 2016). In this paper, we add both sparse and orthogonal regularizations in a unified form inspired by the work (Lin et al., 2017). The details will be introduced in Section 3.

In addition to aspect level sentiment classification (ALSC), aspect category detection (ACD) is another task of aspect based sentiment analysis. ACD (Zhou et al., 2015; Schouten et al., 2018) aims to identify the aspect categories discussed in a given sentence from a predefined set of aspect categories (e.g., price, food, service). Take Figure 1 as an example, aspect categories *food* and *service* are mentioned. We introduce ACD as an auxiliary task to assist the ALSC task, benefiting from the shared context of the two tasks. We also apply our attention constraints to the ACD task. By applying attention weight constraints to both ALSC and ACD tasks in an end-to-end network, we can further evaluate the effectiveness of CAN in multi-task settings.

In summary, the main contributions of our work are as follows:

- We propose CAN for multi-aspect sentiment analysis. Specifically, we introduce orthogonal and sparse regularizations to constrain the attention weight allocation, helping learn better aspect-specific sentence representations.
- We extend CAN to multi-task settings by introducing ACD as an auxiliary task, and applying CAN on both ALSC and ACD tasks.
- Extensive experiments are conducted on public datasets. Results demonstrate the effectiveness of our approach for aspect level sentiment classification.

## 2 Related Work

**Aspect level sentiment analysis** is a fine-grained sentiment analysis task. Earlier methods are usually based on explicit features (Liu et al., 2010; Vo and Zhang, 2015). With the development of deep learning technologies, various neural attention mechanisms have been proposed to solve this fine-grained task (Wang et al., 2016; Ruder et al., 2016; Ma et al., 2017; Tay et al., 2017; Cheng et al., 2017; Chen et al., 2017; Tay et al., 2018; Ma et al., 2018; Wang and Lu, 2018; Wang et al., 2018). To name a few, Wang et al. (2016) propose an attention-based LSTM network for aspect level sentiment classification. Ma et al. (2017) use the interactive attention networks to generate the representations for targets and contexts separately. Cheng et al. (2017); Ruder et al. (2016) both propose hierarchical neural network models for aspect level sentiment classification. Wang and Lu (2018) employ a segmentation attention based LSTM model for aspect level sentiment classification. All these works can be categorized as single-aspect sentiment analysis, which deals with aspects in a sentence separately, without considering the relationship between aspects.

More recently, a few works have been proposed to take the relationship among multiple aspects into consideration. Hazarika et al. (2018) make simultaneous classification of all aspects in a sentence using recurrent networks. Majumder et al. (2018) employ memory network to model the dependency of the target aspect with the other aspects in the sentence. Fan et al. (2018) design an aspect alignment loss to enhance the difference of the attention weights towards the aspects which have the same context and different sentiment polarities. In this paper, we introduce orthogonal regularization to constrain the attention weights of multiple non-overlapping aspects, as well as sparse regularization on each single aspect.

**Multi-task learning** Caruana (1997) solves multiple learning tasks at the same time, achieving improved performance by exploiting commonalities and differences across tasks. Multi-task learning has been used successfully in many machine learning applications. Huang and Zhong (2018) learn both main task and auxiliary task jointly with shared representations, achieving improved performance in question answering. Toshniwal et al. (2017) use low-level auxiliary tasks for encoder-decoder based speech recognition, which suggests that the addition of auxiliary tasks can help in either optimization or generalization. Yu and Jiang (2016) use two auxiliary tasks to help induce a

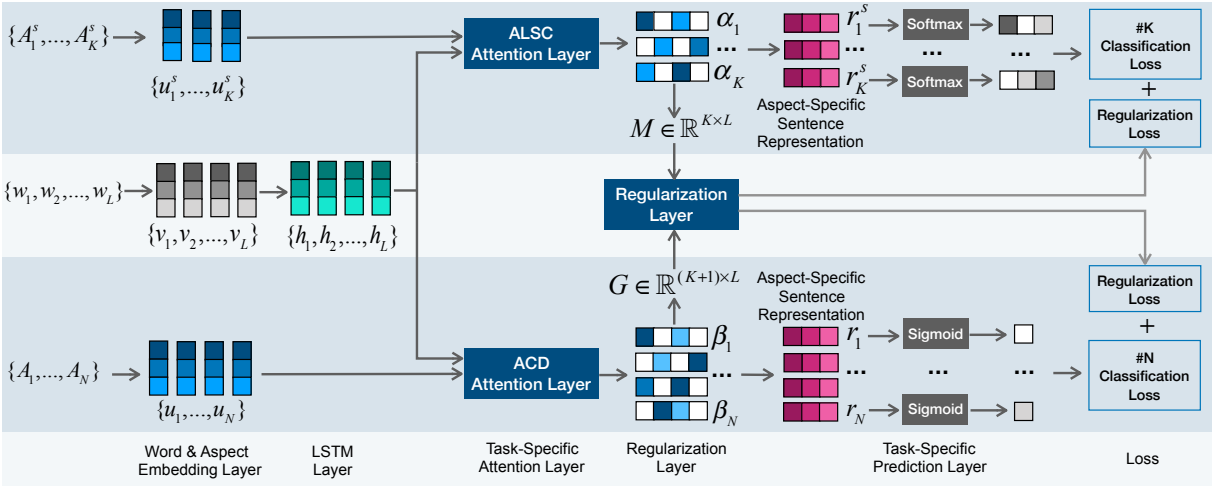


Figure 2: Network Architecture. The aspect categories are embedded as vectors. The model encodes the sentence using LSTM. Based on its hidden states, aspect-specific sentence representations for ALSC and ACD tasks are learned via constrained attention. Then aspect level sentiment prediction and aspect category detection are made.

sentence embedding that works well across domains for sentiment classification. In this paper, we adopt the multi-task learning approach by using ACD as the auxiliary task to help the ALSC task.

## 3 Model

We first formulate the problem. There are totally $N$ predefined aspect categories in the dataset, $A = \{A_1, ..., A_N\}$. Given a sentence $S = \{w_1, w_2, ..., w_L\}$, which contains $K$ aspects $A^s = \{A^s_1, ..., A^s_K\}$, $A^s \subseteq A$, the multi-task learning is to simultaneously solve the ALSC and ACD tasks, namely, the ALSC task predicts the sentiment polarity of each aspect $A^s_k \in A^s$, and the auxiliary ACD task checks each aspect $A_n \in A$ to see whether the sentence $S$ mentions it. Note that we only focus on aspect-category instead of aspect-term (Xue and Li, 2018) in this paper.

We propose CAN for multi-aspect sentiment analysis, supporting both ALSC and ACD tasks by a multi-task learning framework. The network architecture is shown in Figure 2. We will introduce all components sequentially from left to right.

### 3.1 Embedding and LSTM Layers

Traditional single-aspect sentiment analysis handles each aspect separately, one at a time. In such settings, a sentence $S$ with $K$ aspects will be copied to form $K$ instances. For example, a sentence $S$ contains two aspects: $A^s_1$ with polarity $p_1$ and $A^s_2$ with polarity $p_2$. Two instances, $\langle S, A^s_1, p_1\rangle$ and $\langle S, A^s_2, p_2\rangle$, will be constructed. Our multi-aspect sentiment analysis method handles multiple aspects together and takes the single instance $\langle S, [A^s_1, A^s_2], [p_1, p_2]\rangle$ as input.

The input sentence $\{w_1, w_2, ..., w_L\}$ is first converted to a sequence of vectors $\{v_1, v_2, ..., v_L\}$, and the $K$ aspects of the sentence are transformed to vectors $\{u^s_1, ..., u^s_K\}$, which is a subset of $\{u_1, ..., u_N\}$, the vectors of all aspect categories. The word embeddings of the sentence are then fed into an LSTM network (Hochreiter and Schmidhuber, 1997), which outputs hidden states $H = \{h_1, h_2, ..., h_L\}$. The sizes of the embedding and the hidden state are both set to be $d$.

### 3.2 Task-Specific Attention Layer

The ALSC and ACD tasks share the hidden states from the LSTM layer, while compute their own attention weights separately. The attention weights are then used to compute aspect-specific sentence representations.

**ALSC Attention Layer** The key idea of aspect level sentiment classification is to learn different attention weights for different aspects, so that different aspects can concentrate on different parts of the sentence. We follow the approach in the work (Bahdanau et al., 2015) to compute the attention. Particularly, given the sentence $S$ with $K$ aspects, $A^s = \{A^s_1, ..., A^s_K\}$, for each aspect $A^s_k$, its attention weights are calculated by:

$$\alpha_k = softmax(z^{a\mathrm{T}} tanh(W^a_1 H + W^a_2 (u^s_k \otimes e_L))) \quad (1)$$

where $u^s_k$ is the embedding of the aspect $A^s_k$, $e_L \in \mathbb{R}^L$ is a vector of 1s, $u^s_k \otimes e_L$ is the op-

eration repeatedly concatenating $u_k^s$ for $L$ times. $W_1^a \in \mathbb{R}^{d\times d}$, $W_2^a \in \mathbb{R}^{d\times d}$ and $z^a \in \mathbb{R}^d$ are the weight matrices.

**ACD Attention Layer** We treat the ACD task as multi-label classification problem for the set of $N$ aspect categories. For each aspect $A_n \in A$, its attention weights are calculated by:

$$\beta_n = softmax({z^b}^{\mathrm{T}} tanh(W_1^b H + W_2^b(u_n \otimes e_L))) \quad (2)$$

where $u_n$ is the embedding of the aspect $A_n$. $W_1^b \in \mathbb{R}^{d\times d}$, $W_2^b \in \mathbb{R}^{d\times d}$ and $z^b \in \mathbb{R}^d$ are the weight matrices.

The ALSC and ACD tasks use the same attention mechanism, but they do not share parameters. The reason to use separated parameters is that, for the same aspect, the attention of ALSC concentrates more on opinion words, while ACD focuses more on aspect target terms (see the attention visualizations in Section 4.6).

### 3.3 Regularization Layer

We simultaneously handles multiple aspects by adding constraints to their attention weights. **Note that this layer is only available in the training stage**, in which the ground-truth aspects are known for calculating the regularization loss, and then influence parameter updating in back propagation. While in the testing/inference stage, the true aspects are unknown and the regularization loss is not calculated so that this layer is omitted from the architecture.

In this paper, we introduce two types of regularizations: the sparse regularization on each single aspect; the orthogonal regularization on multiple non-overlapping aspects.

**Sparse Regularization** For each aspect, the sparse regularization constrains the distribution of the attention weights ($\alpha_k$ or $\beta_n$) to concentrate on less words. For simplicity, we use $\alpha_k$ as an example, $\alpha_k = \{\alpha_{k1}, \alpha_{k2}, ..., \alpha_{kL}\}$. To make $\alpha_k$ sparse, the sparse regularization term is defined as:

$$R_s = \mid \sum_{l=1}^{L} \alpha_{kl}^2 - 1 \mid \quad (3)$$

where $\sum_{l=1}^{L} \alpha_{kl} = 1$ and $\alpha_{kl} > 0$. Since $\alpha_k$ is normalized as a probability distribution, $L_1$ norm is always equal to 1 (the sum of the probabilities) and does not work as sparse regularization as usual. Minimizing Equation 3 will force the sparsity of $\alpha_k$. It has the similar effect as minimizing the entropy of $\alpha_k$, which leads to placing more probabilities on less words.

**Orthogonal Regularization** This regularization term forces orthogonality among attention weight vectors of multiple aspects, so that different aspects attend on different parts of the sentence with less overlap. Note that we only apply this regularization to non-overlapping multi-aspect sentences. Assume that the sentence $S$ contains $K$ non-overlapping aspects $\{A_1^s, ..., A_K^s\}$ and their attention weight vectors are $\{\alpha_1, ..., \alpha_K\}$. We pack them together as a two-dimensional attention matrix $M \in \mathbb{R}^{K\times L}$ to calculate the orthogonal regularization term.

$$R_o = \parallel M^{\mathrm{T}} M - I \parallel_2 \quad (4)$$

where $I$ is an identity matrix. In the resulted matrix of $M^{\mathrm{T}}M$, each non-diagonal element is the dot product between two attention weight vectors, minimizing the non-diagonal elements will force orthogonality between corresponding attention weight vectors. The diagonal elements of $M^{\mathrm{T}}M$ are subtracted by 1, which are the same as $R_s$ defined in Equation 3. As a whole, $R_o$ includes both sparse and orthogonal regularization terms.

Note that in the ACD task, we do not pack all the $N$ attention vectors $\{\beta_1, ..., \beta_N\}$ as a matrix. The sentence $S$ contains $K$ aspects. For simplicity, let $\{\beta_1, ..., \beta_K\}$ be the attention vectors of the $K$ aspects mentioned, while $\{\beta_{K+1}, ..., \beta_N\}$ be the attention vectors of the $N - K$ aspects not mentioned. We compute the average of the $N - K$ attention vectors, denoted by $\beta_{avg}$. We then construct the attention matrix $G = \{\beta_1, ..., \beta_K, \beta_{avg}\}$, $G \in \mathbb{R}^{(K+1)\times L}$. The reason why we calculate $\beta_{avg}$ is that if an aspect is not mentioned in the sentence, its attention weights often attend to meaningless stop words, such as *"to"*, *"the"*, *"was"*, etc. We do not need to distinguish among the $N - K$ aspects not mentioned, therefore they can share stop words in the sentence by being averaged as a whole, which keeps the $K$ aspects mentioned away from such stop words.

### 3.4 Task-Specific Prediction Layer

Given the attention weights of each aspect, we can generate aspect-specific sentence representation, and then make prediction for the ALSC and ACD tasks respectively.

**ALSC Prediction** The weighted hidden state is combined with the last hidden state to generate the

final aspect-specific sentence representation.

$$r_k^s = tanh(W_1^r \bar{h}_k + W_2^r h_L) \quad (5)$$

where $W_1^r \in \mathbb{R}^{d\times d}$ and $W_2^r \in \mathbb{R}^{d\times d}$. $\bar{h}_k = \sum_{l=1}^{L} \alpha_{kl} h_l$ is the weighted hidden state for aspect $k$. $r_k^s$ is then used to make sentiment polarity prediction.

$$\hat{y_k} = softmax(W_p^a r_k^s + b_p^a) \quad (6)$$

where $W_p^a \in \mathbb{R}^{d\times c}$ and $b_p^a \in \mathbb{R}^c$ are the parameters of the projection layer, and $c$ is the number of classes.

For the sentence $S$ with $K$ aspects mentioned, we make $K$ predictions simultaneously. That is why we call our approach multi-aspect sentiment analysis.

**ACD Prediction** We directly use the weighted hidden state as the sentence representation for ACD prediction.

$$r_n = \bar{h}_n = \sum_{l=1}^{L} \beta_{nl} h_l \quad (7)$$

We do not combine with the last hidden state $h_L$ since the aspect may not be mentioned by the sentence. We make $N$ predictions for all predefined aspect categories.

$$\hat{y_n} = sigmoid(W_p^b r_n + b_p^b) \quad (8)$$

where $W_p^b \in \mathbb{R}^{d\times 1}$ and $b_p^b$ is a scalar.

### 3.5 Loss

For the task ALSC, the loss function for the $K$ aspects of the sentence $S$ is defined by:

$$L_a = -\sum_{k=1}^{K}\sum_{c} y_{kc} \log \hat{y_{kc}}$$

where $c$ is the number of classes. For the task ACD, as each prediction is binary classification problem, the loss function for the $N$ aspects of the sentence $S$ is defined by:

$$L_b = -\sum_{n=1}^{N}[y_n \log \hat{y_n} + (1-y_n)\log(1-\hat{y_n})]$$

We jointly train our model for the two tasks. The parameters in our model are then trained by minimizing the combined loss function:

$$L = L_a + \frac{1}{N} L_b + \lambda R \quad (9)$$

| Dataset | #Single | #Multi | | | #Total |
|---|---|---|---|---|---|
| | | *OL* | *NOL* | *Total* | |
| Rest14_Train | 2053 | 67 | 415 | 482 | 2535 |
| Rest14_Val | 412 | 19 | 75 | 94 | 506 |
| Rest14_Test | 611 | 27 | 162 | 189 | 800 |
| Rest15_Train | 622 | 47 | 262 | 309 | 931 |
| Rest15_Val | 137 | 13 | 39 | 52 | 189 |
| Rest15_Test | 390 | 30 | 162 | 192 | 582 |

Table 1: The numbers of single- and multi-aspect sentences. *OL* and *NOL* denote the overlapping and non-overlapping multi-aspect sentences, respectively.

where $R$ is the regularization term mentioned previously, which can be $R_s$ or $R_o$. $\lambda$ is the hyper-parameter used for tuning the impact from regularization loss to the overall loss. To avoid $L_b$ overwhelming the overall loss, we divide it by the number of aspect categories.

## 4 Experiments

### 4.1 Datasets

We conduct experiments on two public datasets from SemEval 2014 task 4 (Pontiki et al., 2014) and SemEval 2015 task 12 (Pontiki et al., 2015) (denoted by Rest14 and Rest15 respectively). These two datasets consist of restaurant customer reviews with annotations identifying the mentioned aspects and the sentiment polarity of each aspect. To apply orthogonal regularization, we manually annotate the multi-aspect sentences with overlapping or non-overlapping[1]. We randomly split the original training set into training, validation sets in the ratio 5:1, where the validation set is used to select the best model. We count the sentences of single-aspect and multi-aspect separately. Detailed statistics are summarized in Table 1. Particularly, 85.23% and 83.73% of the multi-aspect sentences are non-overlapping in Rest14 and Rest15, respectively.

### 4.2 Comparison Methods

Since we focus on aspect-category sentiment analysis, many works (Ma et al., 2017; Li et al., 2018; Fan et al., 2018) which focus on aspect-term sentiment analysis are excluded.

- **LSTM**: We implement the vanilla LSTM to model the sentence and use the average of all

[1]We will release the annotated dataset later.

| Model | Rest14 | | | | Rest15 | | | |
|---|---|---|---|---|---|---|---|---|
| | 3-way | | Binary | | 3-way | | Binary | |
| | Acc | F1 | Acc | F1 | Acc | F1 | Acc | F1 |
| LSTM | 80.92 | 68.30 | 85.83 | 80.88 | 71.24 | 49.40 | 71.97 | 69.97 |
| AT-LSTM | 81.24 | 69.19 | 87.25 | 82.20 | 73.37 | 51.74 | 76.79 | 74.61 |
| ATAE-LSTM | 82.18 | 69.18 | 88.08 | 83.03 | 74.56 | 51.40 | 79.79 | 78.69 |
| GCAE | 82.08 | 70.20 | 87.72 | 83.84 | 76.69 | 53.00 | 79.66 | 77.96 |
| AT-CAN-$R_s$ | 82.28 | 70.94 | 88.43 | 84.07 | 75.62 | 53.56 | 78.36 | 76.69 |
| AT-CAN-$R_o$ | 82.81 | 71.32 | **89.37** | **85.66** | 76.92 | **55.67** | 79.92 | 78.77 |
| ATAE-CAN-$R_s$ | 81.97 | 72.19 | 88.90 | 84.29 | 77.28 | 52.45 | 81.49 | 80.61 |
| ATAE-CAN-$R_o$ | **83.33** | **73.23** | 89.02 | 84.76 | **78.58** | 54.72 | **81.75** | **80.91** |

Table 2: Results of the ALSC task in single-task settings in terms of accuracy (%) and Macro-F1 (%).

hidden states as the sentence representation. In this model, aspect information is not used.

- **AT-LSTM** (Wang et al., 2016): It adopts the attention mechanism in LSTM to generate a weighted representation of a sentence. The aspect embedding is used to compute the attention weights as in Equation 1. We do not concatenate the aspect embedding to the hidden state as in the work (Wang et al., 2016) and gain small performance improvement. We use this modified version in all experiments.

- **ATAE-LSTM** (Wang et al., 2016): This method is an extension of AT-LSTM. In this model, the aspect embedding is concatenated to each word embedding of the sentence as the input to the LSTM layer.

- **GCAE** (Xue and Li, 2018): This state-of-the-art method is based on the convolutional neural network with gating mechanisms, which is for both aspect-category and aspect-term sentiment analysis. We compare with its aspect-category sentiment analysis task.

### 4.3 Our Methods

To verify the performance gain of introducing constraints on attention weights, we first create several variants of our model for single-task settings.

- **AT-CAN-$R_s$**: Add sparse regularization $R_s$ to AT-LSTM to constrain the attention weights of each single aspect.

- **AT-CAN-$R_o$**: Add orthogonal regularization $R_o$ to AT-CAN-$R_s$ to constrain the attention weights of multiple non-overlapping aspects.

- **ATAE-CAN-$R_s$**: Add $R_s$ to ATAE-LSTM.

- **ATAE-CAN-$R_o$**: Add $R_o$ to ATAE-CAN-$R_s$.

We then extend attention constraints to multi-task settings, creating variants by different options: 1) no constraints, 2) adding regularizations only to the ALSC task, 3) adding regularizations to both tasks.

- **M-AT-LSTM**: This is the basic multi-task model without regularizations.

- **M-CAN-$R_s$**: Add $R_s$ to the ALSC task in M-AT-LSTM.

- **M-CAN-$R_o$**: Add $R_o$ to the ALSC task in M-CAN-$R_s$.

- **M-CAN-2$R_s$**: Add $R_s$ to both tasks in M-AT-LSTM.

- **M-CAN-2$R_o$**: Add $R_o$ to both tasks in M-CAN-2$R_s$.

### 4.4 Implementation Details

We set $\lambda = 0.1$ with the help of the validation set. All models are optimized by the Adagrad optimizer (Duchi et al., 2011) with learning rate 0.01. Batch size is 25. We apply a dropout of $p = 0.7$ after the embedding and LSTM layers. All words in the sentences are initialized with 300 dimension Glove Embeddings (Pennington et al., 2014). The aspect embedding matrix and parameters are initialized by sampling from a uniform distribution $U(-\varepsilon, \varepsilon)$, $\varepsilon = 0.01$. $d$ is set as 300. The models are trained for 100 epochs, during which the model with the best performance on the validation set is saved. We also apply early stopping in training, which means that the training will stop if the performance on validation set does not improve in 10 epochs.

| Model | Rest14 | | | | Rest15 | | | |
|---|---|---|---|---|---|---|---|---|
| | 3-way | | Binary | | 3-way | | Binary | |
| | Acc | F1 | Acc | F1 | Acc | F1 | Acc | F1 |
| M-AT-LSTM | 82.60 | 71.44 | 88.55 | 83.76 | 76.33 | 51.64 | 79.53 | 78.31 |
| M-CAN-$R_s$ | 83.65 | 73.97 | 89.26 | 85.43 | 75.74 | 52.43 | 79.66 | 78.46 |
| M-CAN-$R_o$ | 83.12 | 72.29 | 89.61 | 85.18 | 77.04 | 52.69 | 79.40 | 77.88 |
| M-CAN-2$R_s$ | 83.23 | 72.81 | 89.37 | 85.42 | **78.22** | **55.80** | 80.44 | 80.01 |
| M-CAN-2$R_o$ | **84.28** | **74.45** | **89.96** | **86.16** | 77.51 | 52.78 | **82.14** | **81.58** |

Table 3: Results of the ALSC task in multi-task settings in terms of accuracy (%) and Macro-F1 (%).

| Model | Rest14 | | | Rest15 | | |
|---|---|---|---|---|---|---|
| | Precision | Recall | F1 | Precision | Recall | F1 |
| M-AT-LSTM | 0.8626 | 0.8553 | 0.8589 | 0.6691 | 0.4748 | 0.5555 |
| M-CAN-2$R_s$ | 0.8698 | 0.8595 | 0.8645 | 0.6244 | **0.5019** | 0.5565 |
| M-CAN-2$R_o$ | **0.8907** | **0.8627** | **0.8765** | **0.7127** | 0.4865 | **0.5782** |

Table 4: Results of the ACD task. Rest14 has 5 aspect categories while Rest15 has 13 ones.

### 4.5 Results

Table 2 and 3 show our experimental results on the two public datasets for single-task and multi-task settings respectively. In both tables, "3-way" stands for 3-class classification (positive, neutral, and negative), and "Binary" for binary classification (positive and negative). The best scores are marked in bold.

**Single-task Settings** Table 2 shows our experimental results of ALSC in single-task settings. Firstly, we observe that by introducing attention regularizations (either $R_s$ or $R_o$), most of our proposed methods outperform their counterparts. Particularly, AT-CAN-$R_s$ and AT-CAN-$R_o$ outperform AT-LSTM in all results; ATAE-CAN-$R_s$ and ATAE-CAN-$R_o$ also outperform ATAE-LSTM in 15 of 16 results. For example, in the Rest15 dataset, ATAE-CAN-$R_o$ outperforms ATAE-LSTM by up to 5.39% of accuracy and 6.46% of the F1 score in the 3-way classification. Secondly, regularization $R_o$ achieves better performance improvement than $R_s$ in all results. This is because $R_o$ includes both orthogonal and sparse regularizations for non-overlapping multi-aspect sentences. Thirdly, our approaches, especially ATAE-CAN-$R_o$, outperform the state-of-the-art baseline model GCAE. Finally, the LSTM method outputs the worst results in all cases, because it can not distinguish different aspects.

**Multi-task Settings** Table 3 shows experimental results of ALSC in multi-task settings. We first observe that the overall results in multi-task settings outperform the ones in single-task settings, which demonstrates the effectiveness of multi-task learning by introducing the auxiliary ACD task to help the ALSC task. Second, in almost all cases, applying attention regularizations to both tasks gains more performance improvement than only to the ALSC task, which shows that our attention regularization approach can be extended to different tasks which involving aspect level attention weights, and works well in multi-task settings. For example, for the Binary classification in the Rest15 dataset, M-AT-LASTM outperforms AT-LSTM by 3.57% of accuracy and 4.96% of the F1 score, and M-CAN-2$R_o$ further outperforms M-AT-LSTM by 3.28% of accuracy and 4.0% of the F1 score.

Table 4 shows the results of the ACD task in multi-task settings. Our proposed regularization terms can also improve the performance of ACD. Regularization $R_o$ achieves the best performance in almost all metrics.

### 4.6 Attention Visualizations

Figure 3 depicts the attention weights from AT-LSTM, M-AT-LSTM and M-CAN-2$R_o$ methods, which are used to predict the sentiment polarity in the ALSC task. The subfigure (a), (b) and (c) show the attention weights of the same sentence, for the aspect *food* and *service* respectively. We observe that the attention weights of each word associated with each aspect are quite different for different methods. For AT-LSTM method in sub-

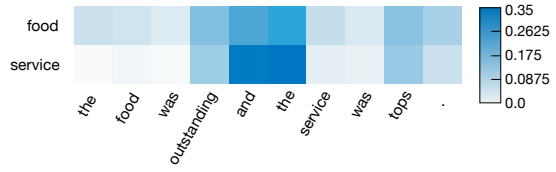


(a) AT-LSTM

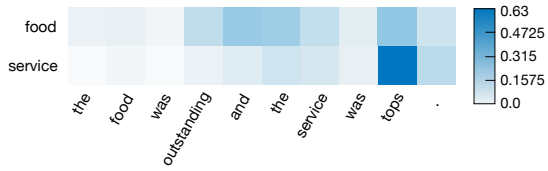


(b) M-AT-LSTM

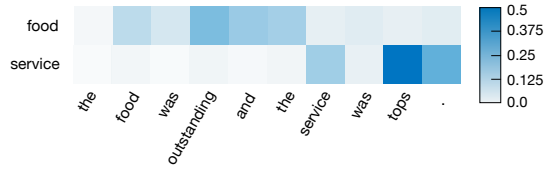


(c) M-CAN-2$R_o$

Figure 3: Visualization of attention weights of different aspects in the ALSC task. Three different models are compared.

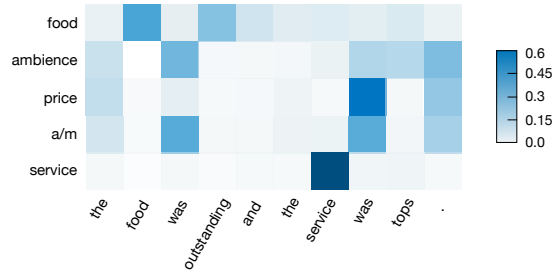


Figure 4: Visualization of attention weights of different aspects in the ACD task from M-CAN-2$R_o$. The a/m is short for anecdotes/miscellaneous.

figure (a), the attention weights of aspect *food* and *service* are both high in words *"outstanding"*, *"and"*, and *"the"*, but actually, the word *"outstanding"* is used to describe the aspect *food* rather than *service*. The same situation occurs with the word *"tops"*, which should associate with *service* rather than *food*. The attention mechanism alone is not good enough to locate aspect-specific opinion words and generate aspect-specific sentence representations in the ALSC task.

As shown in subfigure (b), the issue is mitigated in M-AT-LSTM. Multi-task learning can learn better hidden states of the sentence, and better aspect embeddings. However, it is still not good enough. For instance, the attention weights of the word

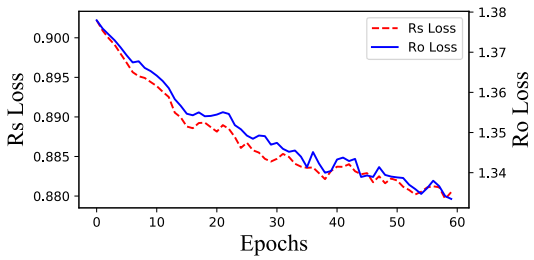


Figure 5: The regularization loss curves of $R_s$ and $R_o$ during the training of AT-CAN-$R_o$.

*"tops"* are both high for the two aspects, and the weights are overlapped in the middle part of the sentence.

As shown in subfigure (c), M-CAN+2$R_o$ generates the best attention weights. The attention weights of the aspect *food* are almost orthogonal to the weights of *service*. The aspect *food* concentrates on the first part of the sentence while *service* on the second part. Meanwhile, the key opinion words *"outstanding"* and *"tops"* get highest attention weights in the corresponding aspects.

We also visualize the attention for the auxiliary task ACD. Figure 4 depicts the attention weights from the method M-CAN-2$R_o$. There are five predefined aspect categories (*food, ambience, price, anecdotes/miscellaneous, service*) in the dataset, two of which are mentioned in the sentence. In the ACD task, we need to calculate the attention weights for all the five aspect categories, and then generate aspect-specific sentence representations to determine whether the sentence contains each aspect. As shown in Figure 4, attention weights for aspects *food* and *service* are pretty good. The aspect *food* concentrates on words *"food"* and *"outstanding"*, and the aspect *service* focuses on the word *"service"*. It is interesting that for aspects which are not mentioned in the sentence, their attention weights often attend to meaningless stop words, such as *"the"*, *"was"*, etc. We do not distinguish these aspects and just treat them as a whole.

We plot the regularization loss curves in Figure 5, which shows that both $R_s$ and $R_o$ decrease during the training of AT-CAN-$R_o$.

### 4.7 Case Studies

**Overlapping Case** We only add sparse regularization to overlapping sentences in which multiple aspects share the same opinion snippet. As shown in Figure 6, the sentence contains two aspects *food*

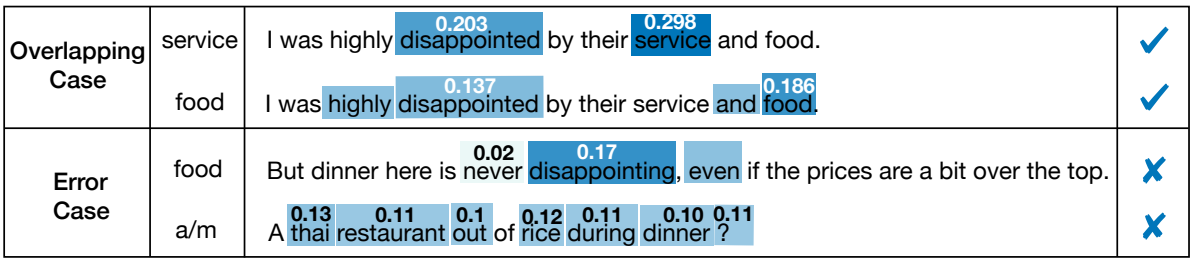


Figure 6: Examples of overlapping case and error case. The a/m is short for anecdotes/miscellaneous.

and *service*, both described by the opinion snippet *"highly disappointed"*. Our method can locate the aspect terms and shared opinion words for both aspects, and then classify the sentiment correctly.

**Error Case** With the help of attention visualization, we can conduct error analysis of our model conveniently. As shown in Figure 6, for the first sentence in error case, the aspect *food* attends on the right word *"disappointing"*, but fails to include the negation word *"never"*. This may be caused by the inaccurate sentence representation or aspect embedding. We can not rebuild the connection between the aspect and the word by our regularizations. The second sentence is negative but classified as neutral. The attention weights distribute evenly since the sentence does not contain any explicit opinion words. Since there is no tendency which words to concentrate, our model can not adjust the attention weights and help on such cases.

## 5 Conclusions

We propose constrained attention networks for multi-aspect sentiment analysis, which handles multiple aspects of a sentence simultaneously. Specifically, we introduce orthogonal and sparse regularizations on attention weights. Furthermore, we introduce an auxiliary task ACD for promoting the ALSC task, and apply CAN on both tasks. Experimental results demonstrate that our approach outperforms the state-of-the-art methods.

## 6 Acknowledgement

This work is supported by National Science and Technology Major Project, China (Grant No. 2018YFB0204304).

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. 2016. Domain separation networks. In *NIPS*.

Rich Caruana. 1997. Multitask learning. *Machine Learning*, 28(1):41–75.

Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. 2017. Recurrent attention network on memory for aspect sentiment analysis. In *EMNLP*.

Jiajun Cheng, Shenglin Zhao, Jiani Zhang, Irwin King, Xin Zhang, and Hui Wang. 2017. Aspect-level sentiment classification with heat (hierarchical attention) network. In *CIKM*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Feifan Fan, Yansong Feng, and Dongyan Zhao. 2018. Multi-grained attention network for aspect-level sentiment classification. In *EMNLP*.

Devamanyu Hazarika, Soujanya Poria, Prateek Vij, Gangeshwar Krishnamurthy, Erik Cambria, and Roger Zimmermann. 2018. Modeling inter-aspect dependencies for aspect-based sentiment analysis. In *NAACL*.

Sepp Hochreiter and Jrgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Binxuan Huang, Yanglan Ou, and Kathleen M Carley. 2018. Aspect level sentiment classification with attention-over-attention neural networks. *arXiv preprint arXiv:1804.06536*.

Yanzhou Huang and Tao Zhong. 2018. Multitask learning for neural generative question answering. *Machine Vision & Applications*, pages 1–9.

Xin Li, Lidong Bing, Wai Lam, and Bei Shi. 2018. Transformation networks for target-oriented sentiment classification. In *ACL*.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *ICLR*.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.

Feifan Liu, Dong Wang, Bin Li, and Yang Liu. 2010. Improving blog polarity classification via topic analysis and adaptive methods. In *NAACL*.

Dehong Ma, Sujian Li, Xiaodong Zhang, Houfeng Wang, Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2017. Interactive attention networks for aspect-level sentiment classification. In *IJCAI*.

Yukun Ma, Haiyun Peng, and Erik Cambria. 2018. Targeted aspect-based sentiment analysis via embedding commonsense knowledge into an attentive lstm. In *AAAI*.

Navonil Majumder, Soujanya Poria, Alexander Gelbukh, Md Shad Akhtar, Erik Cambria, and Asif Ekbal. 2018. IARM: Inter-aspect relation modeling with memory networks in aspect-based sentiment analysis. In *EMNLP*.

Chaitanya Malaviya, Pedro Ferreira, and André F. T. Martins. 2018. Sparse and constrained attention for neural machine translation. In *ACL*.

Tetsuya Nasukawa and Jeonghee Yi. 2003. Sentiment analysis:capturing favorability using natural language processing. In *K-CAP*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Suresh Manandhar, and Ion Androutsopoulos. 2015. SemEval-2015 task 12: Aspect based sentiment analysis. In *SemEval 2015*.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *SemEval 2014*.

Sebastian Ruder, Parsa Ghaffari, and John G. Breslin. 2016. A hierarchical model of reviews for aspect-based sentiment analysis. In *EMNLP*.

Kim Schouten, Onne Van Der Weijde, Flavius Frasincar, and Rommert Dekker. 2018. Supervised and unsupervised aspect category detection for sentiment analysis with co-occurrence data. *IEEE Transactions on Cybernetics*, 48(4):1263–1275.

Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2018. Learning to attend via word-aspect associative fusion for aspect-based sentiment analysis. In *AAAI*.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2017. Dyadic memory networks for aspect-based sentiment analysis. In *CIKM*.

Shubham Toshniwal, Hao Tang, Liang Lu, and Karen Livescu. 2017. Multitask learning with low-level auxiliary tasks for encoder-decoder based speech recognition. *INTERSPEECH*, pages 3532–3536.

Duy Tin Vo and Yue Zhang. 2015. Target-dependent twitter sentiment classification with rich automatic features. In *IJCAI*.

Bailin Wang and Wei Lu. 2018. Learning latent opinions for aspect-level sentiment classification. In *AAAI*.

Jingjing Wang, Jie Li, Shoushan Li, Yangyang Kang, Min Zhang, Luo Si, and Guodong Zhou. 2018. Aspect sentiment classification with both word-level and clause-level attention networks. In *IJCAI*.

Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016. Attention-based lstm for aspect-level sentiment classification. In *EMNLP*.

Wei Xue and Tao Li. 2018. Aspect based sentiment analysis with gated convolutional networks. In *ACL*.

Jianfei Yu and Jing Jiang. 2016. Learning sentence embeddings with auxiliary tasks for cross-domain sentiment classification. In *EMNLP*.

Xinjie Zhou, Xiaojun Wan, and Jianguo Xiao. 2015. Representation learning for aspect category detection in online reviews. In *AAAI*.

# Leveraging Just a Few Keywords for Fine-Grained Aspect Detection Through Weakly Supervised Co-Training

**Giannis Karamanolakis, Daniel Hsu, Luis Gravano**
Columbia University, New York, NY 10027, USA
{gkaraman, djhsu, gravano}@cs.columbia.edu

## Abstract

User-generated reviews can be decomposed into fine-grained segments (e.g., sentences, clauses), each evaluating a different aspect of the principal entity (e.g., price, quality, appearance). Automatically detecting these aspects can be useful for both users and downstream opinion mining applications. Current supervised approaches for learning aspect classifiers require many fine-grained aspect labels, which are labor-intensive to obtain. And, unfortunately, unsupervised topic models often fail to capture the aspects of interest. In this work, we consider weakly supervised approaches for training aspect classifiers that only require the user to provide a small set of seed words (i.e., weakly positive indicators) for the aspects of interest. First, we show that current weakly supervised approaches do not effectively leverage the predictive power of seed words for aspect detection. Next, we propose a student-teacher approach that effectively leverages seed words in a bag-of-words classifier (teacher); in turn, we use the teacher to train a second model (student) that is potentially more powerful (e.g., a neural network that uses pre-trained word embeddings). Finally, we show that iterative co-training can be used to cope with noisy seed words, leading to both improved teacher and student models. Our proposed approach consistently outperforms previous weakly supervised approaches (by 14.1 absolute F1 points on average) in six different domains of product reviews and six multilingual datasets of restaurant reviews.

## 1 Introduction

A typical review of an entity on platforms such as Yelp and Amazon discusses multiple aspects of the entity (e.g., price, quality) in individual review segments (e.g., sentences, clauses). Consider for example the Amazon product review in Figure 1. The text discusses various aspects of the TV such as price, ease of use, and sound quality. Given the vast number of online reviews, both sellers and customers would benefit from automatic methods for detecting fine-grained segments that discuss particular aspects of interest. Fine-grained aspect detection is also a key task in downstream applications such as aspect-based sentiment analysis and multi-document summarization (Hu and Liu, 2004; Liu, 2012; Pontiki et al., 2016; Angelidis and Lapata, 2018).

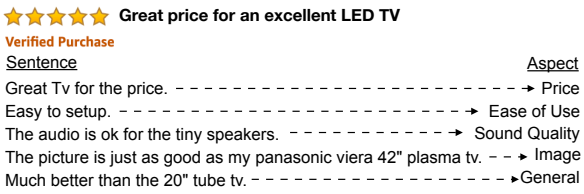


Figure 1: Example of product review with aspect annotations: each individual sentence of the review discusses a different aspect (e.g., price) of the TV.

In this work, we consider the problem of classifying individual segments of reviews to predefined aspect classes when ground truth aspect labels are not available. Indeed, reviews are often entered as unstructured, free-form text and do not come with aspect labels. Also, it is infeasible to manually obtain segment annotations for retail stores like Amazon with millions of different products. Unfortunately, fully supervised neural networks cannot be applied without aspect labels. Moreover, the topics learned by unsupervised neural topic models are not perfectly aligned with the users' aspects of interest, so substantial human effort is required for interpreting and mapping the learned topics to meaningful aspects.

Here, we investigate whether neural networks can be effectively trained under this challenging setting when only a small number of descriptive keywords, or *seed words*, are available for each



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4611–4621, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| Aspect | Seed Words |
|---|---|
| Price (EN) | price, value, money, worth, paid |
| Image (EN) | picture, color, quality, black, bright |
| Food (EN) | food, delicious, pizza, cheese, sushi |
| Drinks (FR) | vin, bière, verre, bouteille, cocktail |
| Ambience (SP) | ambiente, mesas, terraza, acogedor, ruido |

Table 1: Examples of aspects and five of their corresponding seed words in various domains (electronic products, restaurants) and languages ("EN" for English, "FR" for French, "SP" for Spanish).

aspect class. Table 1 shows examples of aspects and five of their corresponding seed words from our experimental datasets (described later in more detail). In contrast to a classification label, which is only relevant for a single segment, a seed word can implicitly provide aspect supervision to potentially many segments. We assume that the seed words have already been collected either manually or automatically. Indeed, collecting a small[1] set of seed words per aspect is typically easier than manually annotating thousands of segments for training neural networks. As we will see, even noisy seed words that are only weakly predictive of the aspect will be useful for aspect detection.

Training neural networks for segment-level aspect detection using just a few seed words is a challenging task. Indeed, as a contribution of this paper, we observe that current weakly supervised networks do not effectively leverage the predictive power of the available seed words. To address the shortcomings of previous seed word-based approaches, we propose a novel *weakly supervised* approach, which uses the available seed words in a more effective way. In particular, we consider a *student-teacher* framework, according to which a bag-of-seed-words classifier (teacher) is applied on unlabeled segments to supervise a second model (student), which can be any supervised model, including neural networks.

Our approach introduces several important contributions. First, our teacher model considers each individual seed word as a (noisy) aspect indicator, which as we will show, is more effective than previously proposed weakly supervised approaches. Second, by using only the teacher's aspect probabilities, our student generalizes better than the teacher and, as a result, the student outperforms both the teacher and previously proposed weakly supervised models. Finally, we show how iterative co-training can be used to cope with noisy seed words: the teacher effectively estimates the predictive quality of the noisy seed words in an unsupervised manner using the associated predictions by the student. Iterative co-training then leads to both improved teacher and student models. Overall, our approach consistently outperforms existing weakly supervised approaches, as we show with an experimental evaluation over six domains of product reviews and six multilingual datasets of restaurant reviews.

The rest of this paper is organized as follows. In Section 2 we review relevant work. In Section 3 we describe our proposed weakly supervised approach. In Section 4 we present our experimental setup and findings. Finally, in Section 5 we conclude and suggest future work. A preliminary version of this work was presented at the Second Learning from Limited Labeled Data Workshop (Karamanolakis et al., 2019).

## 2 Related Work and Problem Definition

We now review relevant work on aspect detection (Section 2.1), co-training (Section 2.2), and knowledge distillation (Section 2.3). We also define our problem of focus (Section 2.4).

### 2.1 Segment-Level Aspect Detection

The goal of segment-level aspect detection is to classify a segment $s$ to $K$ aspects of interest.

**Supervised Approaches.** Rule-based or traditional learning models for aspect detection have been outperformed by supervised neural networks (Liu et al., 2015; Poria et al., 2016; Zhang et al., 2018). Supervised neural networks first use an embedding function[2] (EMB) to compute a low dimensional segment representation $h = \text{EMB}(s) \in \mathbb{R}^d$ and then feed $h$ to a classification layer[3] (CLF) to predict probabilities for the $K$ aspect classes of interest: $p = \langle p^1, \ldots, p^K \rangle = \text{CLF}(h)$. For simplicity, we write $p = f(s)$. The parameters of the embedding function and the classification layer are learned using ground truth,

[1] In our experiments, we only consider around 30 seed words per aspect. For comparison, the vocabulary of the datasets has more than 10,000 terms.

[2] Examples of segment embedding functions are the average of word embeddings (Wieting et al., 2015; Arora et al., 2017), Recurrent Neural Networks (RNNs) (Yang et al., 2016; Wieting and Gimpel, 2017), Convolutional Neural Networks (CNNs) (Kim, 2014), self-attention blocks (Devlin et al., 2019; Radford et al., 2018), etc.

[3] The classification layer is usually a hidden layer followed by the softmax function.

segment-level aspect labels. However, aspect labels are not available in our setting, which hinders the application of supervised learning approaches.

**Unsupervised Approaches.** Topic models have been used to train aspect detection with unannotated documents. Recently, neural topic models (Iyyer et al., 2016; Srivastava and Sutton, 2071; He et al., 2017) have been shown to produce more coherent topics than earlier models such as Latent Dirichlet Allocation (LDA) (Blei et al., 2003). In their Aspect Based Autoencoder (ABAE), He et al. (2017) first use segment $s$ to predict aspect probabilities $p = f(s)$ and then use $p$ to reconstruct an embedding $h'$ for $s$ as a convex combination of $K$ aspect embeddings: $h' = \sum_{k=1}^{K} p^k A_k$, where $A_k \in \mathbb{R}^d$ is the embedding of the $k$-th aspect. The aspect embeddings $A_k$ are initialized by clustering the vocabulary embeddings using k-means with $K$ clusters. ABAE is trained by minimizing the segment reconstruction error.[4]

Unfortunately, unsupervised topic models are not effective when used directly for aspect detection. In particular, in ABAE, the $K$ topics learned to reconstruct the segments are not necessarily aligned with the $K$ aspects of interest. A possible fix is to first learn $K' >> K$ topics and do a $K'$-to-$K$ mapping as a post-hoc step. However, this mapping requires either aspect labels or substantial human effort for interpreting topics and associating them with aspects. This mapping is nevertheless not possible if the learned topics are not aligned with the aspects.

**Weakly Supervised Approaches.** Weakly supervised approaches use minimal domain knowledge (instead of ground truth labels) to model meaningful aspects. In our setting, domain knowledge is given as a set of seed words for each aspect of interest (Lu et al., 2011; Lund et al., 2017; Angelidis and Lapata, 2018). Lu et al. (2011) use seed words as asymmetric priors in probabilistic topic models (including LDA). Lund et al. (2017) use LDA with fixed topic-word distributions, which are learned using seed words as "anchors" for topic inference (Arora et al., 2013). Neither of these two approaches can be directly applied into more recent neural networks for aspect detection. Angelidis and Lapata (2018) recently proposed a weakly supervised extension of the unsupervised ABAE. Their model, named Multi-seed Aspect Extractor, or MATE, initializes the aspect embedding $A_k$ using the weighted average of the corresponding seed word embeddings (instead of the k-means centroids). To guarantee that the aspect embeddings will still be aligned with the $K$ aspects of interest after training, Angelidis and Lapata (2018) keep the aspect and word embeddings fixed throughout training. In this work, we will show that the predictive power of seed words can be leveraged more effectively by considering each individual seed word as a more direct source of supervision during training.

[4]The reconstruction error can be efficiently estimated using contrastive max-margin objectives (Weston et al., 2011; Pennington et al., 2014).

## 2.2 Co-training

Co-training (Blum and Mitchell, 1998) is a classic multi-view learning method for semi-supervised learning. In co-training, classifiers over different feature spaces are encouraged to agree in their predictions on a large pool of unlabeled examples. Blum and Mitchell (1998) justify co-training in a setting where the different views are conditionally independent given the label. Several subsequent works have relaxed this assumption and shown co-training to be effective in much more general settings (Balcan et al., 2005; Chen et al., 2011; Collins and Singer, 1999; Clark et al., 2018). Co-training is also related to self-training (or bootstrapping) (Yarowsky, 1995), which trains a classifier using its own predictions and has been successfully applied for various NLP tasks (Collins and Singer, 1999; McClosky et al., 2006).

Recent research has successfully revisited these general ideas to solve NLP problems with modern deep learning methods. Clark et al. (2018) propose "cross-view training" for sequence modeling tasks by modifying Bi-LSTMs for *semi-supervised* learning. Ruder and Plank (2018) show that classic bootstrapping approaches such as tri-training (Zhou and Li, 2005) can be effectively integrated in neural networks for semi-supervised learning under domain shift. Our work provides further evidence that co-training can be effectively integrated into neural networks and combined with recent transfer learning approaches for NLP (Dai and Le, 2015; Howard and Ruder, 2018; Devlin et al., 2019; Radford et al., 2018), in a substantially different, *weakly supervised* setting where no ground-truth labels but only a few seed words are available for training.

| Variable | Description |
|---|---|
| $s$ | Segment (e.g., sentence) of a text review |
| $K$ | Number of aspects of interest |
| $D$ | Total number of seed words |
| $G_i \quad (i = 1, \ldots, K)$ | Set of seed words for the $i$-th aspect |
| $h \in \mathbb{R}^d$ | Segment embedding (student) |
| $c \in \mathbb{N}^D$ | Bag-of-seed-words representation of $s$ |
| $p = \langle p^1, \ldots, p^K \rangle$ | Student's aspect predictions |
| $q = \langle q^1, \ldots, q^K \rangle$ | Teacher's aspect predictions |

Table 2: Notation.

### 2.3 Knowledge Distillation

Our approach is also related to the "knowledge distillation" framework (Buciluǎ et al., 2006; Ba and Caruana, 2014; Hinton et al., 2015), which has received considerable attention recently (Lopez-Paz et al., 2016; Kim and Rush, 2016; Furlanello et al., 2018; Wang, 2019). Traditional knowledge distillation aims at compressing a cumbersome model (teacher) to a simpler model (student) by training the student using both ground truth labels and the soft predictions of the teacher in a distillation objective. Our work also considers a student-teacher architecture and the distillation objective but under a considerably different, weakly supervised setting: (1) we do not use any labels for training and (2) we create conditions that allow the student to outperform the teacher; in turn, (3) we can use the student's predictions to learn a better teacher under co-training.

### 2.4 Problem Definition

Consider a corpus of text reviews from an entity domain (e.g., televisions, restaurants). Each review is split into segments (e.g., sentences, clauses). We also consider $K$ pre-defined aspects of interest $(1, \ldots, K)$, including the "General" aspect, which we assume is the $K$-th aspect for simplicity. Different segments of the same review may be associated with different aspects but ground-truth aspect labels are *not* available for training. Instead, a small number of seed words $G_k$ are provided for each aspect $k \in [K]$. Our goal is to use the corpus of training reviews and the available seed words $G = (G_1, \ldots, G_K)$ to train a classifier, which, given an unseen test segment $s$, predicts $K$ aspect probabilities $p = \langle p^1, \ldots, p^K \rangle$.

## 3 Our Student-Teacher Approach

We now describe our weakly supervised framework for aspect detection. We consider a student-teacher architecture (Figure 2), where the teacher

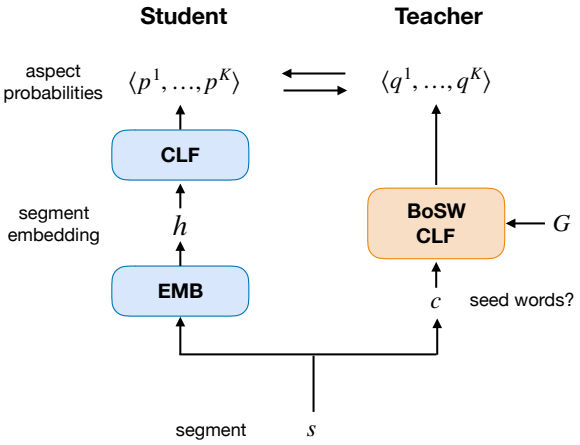


Figure 2: Our student-teacher approach for segment-level aspect detection using seed words.

is a bag-of-words classifier based solely on the provided seed words (i.e., a "bag-of-seed-words" classifier), and the student is an embedding-based neural network trained on data "softly" labeled by the teacher (as in the distillation objective). In the rest of this section, we describe the individual components of our student-teacher architecture and our proposed algorithm for performing updates.

### 3.1 Teacher: A Bag-of-Seed-Words Classifier

Our teacher model leverages the available seed words $G$ that are predictive of the $K$ aspects. Let $D$ denote the total number of seed words in $G$. We can represent a segment $s_i$ using a bag-of-seed-words representation $c_i \in \mathbb{N}^D$, where $c_i^j$ encodes the number of times the $j$-th seed word occurs in $s_i$. (Note that $c_i$ ignores the non-seed words.) The teacher's prediction for the $k$-th aspect is:

$$q_i^k = \frac{\exp(\sum_{j=1}^{D} \mathbb{1}\{j \in G_k\} \cdot c_i^j)}{\sum_{k'} \exp(\sum_{j=1}^{D} \mathbb{1}\{j \in G_{k'}\} \cdot c_i^j)}. \quad (1)$$

If no seed word appears in $s$, then the teacher predicts the "General" aspect by setting $q_i^K = 1$. Under this configuration the teacher uses seed words in a direct and intuitive way: it predicts aspect probabilities for the $k$-th aspect, which are proportional to the counts of the seed words under $G_k$, while if no seed word occurs in $s$, it predicts the "General" aspect. The classifier receives $c_i$ as input and predicts $q_i = \langle q_i^1, \ldots, q_i^K \rangle$.

Although the teacher only uses seed words to predict the aspect of a segment, we also expect

non-seed words to carry predictive power. Next, we describe the student network that learns to associate non-seed words with aspects.

### 3.2 Student: An Embedding-Based Network

Our student model is an embedding-based neural network: a segment is first embedded ($h_i = \text{EMB}(s_i) \in \mathbb{R}^d$) and then classified to the $K$ aspects ($p_i = \text{CLF}(h_i)$) (see Section 2.1). The student does not use ground-truth aspect labels for training. Instead, it is trained by optimizing the distillation objective, i.e., the cross entropy between the teacher's (soft) predictions and the student's predictions:

$$H(q_i, p_i) = -\sum_k q_i^k \log p_i^k \quad (2)$$

While the teacher only uses the seed words in $s_i$ to form its prediction $q_i$, the student uses all the words in $s_i$. Thus, using the distillation loss for training, the student learns to use both seed words and non-seed words to predict aspects. As a result, the student is able to generalize better than the teacher and *predict aspects even in segments that do not contain any seed words*. To regularize the student model, we apply L2 regularization to the classifier's weights and dropout regularization to the word embeddings (Srivastava et al., 2014). As we will show in Section 4, our student with this configuration outperforms the teacher in aspect prediction.

### 3.3 Iterative Co-Training

In this section, we describe our iterative co-training algorithm to cope with noisy seed words. The teacher in Section 3.1 considers each seed word equally, which can be problematic because not all seed words are equally good for predicting an aspect. In this work, we propose to estimate the predictive quality of each seed word in an unsupervised way. Our approach is inspired in the Model Bootstrapped Expectation Maximization (MBEM) algorithm of Khetan et al. (2018). MBEM is guaranteed to converge (under mild conditions) when the number of training data is sufficiently large and the worker quality is sufficiently high. Here, we treat seed words as "noisy annotators" and adopt an iterative estimation procedure similar to MBEM, as we describe next.

We model the predictive quality of the $j$-th seed word as a weight vector $z_j = \langle z_j^1, \ldots, z_j^K \rangle$, where

**Algorithm 1** Iterative Seed Word Distillation

**Input:** $\{s_i\}_{i\in[N]}$, $D$ seed words grouped into $K$ disjoint sets $G = (G_1, \ldots, G_K)$
**Output:** $\hat{f}$: predictor function for segment-level aspect detection

Predict $\{q_i\}_{i\in[N]}$ (Eq. (1)) ▷ *Apply teacher*
**Repeat until convergence criterion**
  Learn $\hat{f}$ (Eq. (2)) ▷ *Train student*
  Predict $\{p_i = \hat{f}(s_i)\}_{i\in[N]}$ ▷ *Apply student*
  Update $\{z_j\}_{j\in[D]}$ (Eq. (4)) ▷ *Update teacher*
  Predict $\{q_i\}_{i\in[N]}$ (Eq. (3)) ▷ *Apply teacher*

$z_j^k$ measures the strength of the association with the $k$-th aspect. We thus change the teacher to consider seed word quality. In particular, we replace Equation (1) by:

$$q_i^k = \frac{\exp \sum_{j=1}^{D} \mathbb{1}\{j \in G_k\} \cdot \hat{z}_j^k \cdot c_i^j}{\sum_{k'} \exp \sum_{j=1}^{D} \mathbb{1}\{j \in G_{k'}\} \cdot \hat{z}_j^{k'} \cdot c_i^j}, \quad (3)$$

where $\hat{z}_j$ is the current estimate of $z_j$. As no ground-truth labels are available, we follow Khetan et al. (2018) and estimate $z_j$ via Maximum Likelihood Estimation using the student's predictions as the current estimate of the ground truth labels. In particular, we assume that the prediction of the student for a training segment $s_i$ is $t_i = \text{argmax}_k\, p_i^k$. Then, for each seed word we compute the quality estimate for the $k$-th aspect using the student's predictions for $N$ segments:

$$\hat{z}_j^k = \frac{\sum_{i=1}^{N} \mathbb{1}\{c_i^j > 0\} \mathbb{1}\{t_i = k\}}{\sum_{k'} \sum_{i=1}^{N} \mathbb{1}\{c_i^j > 0\} \mathbb{1}\{t_i = k'\}}. \quad (4)$$

According to Equation (4), the quality of the $j$-th seed word is estimated according to the student-teacher agreement on segments where the seed word appears.

Building upon the previous ideas, we present our Iterative Seed Word Distillation (ISWD) algorithm for effectively leveraging the seed words for fine-grained aspect detection. Each round of ISWD consists of the following steps (Algorithm 1): (1) we apply the teacher on unlabeled training segments to get predictions $q_i$ (without considering seed word qualities); (2) we train the student using the teacher's predictions in the distillation objective of Equation (2);[5] (3) we apply the student in the training data to get predictions $p_i$; and

[5] Note that the quality-aware loss function proposed

(4) we update the seed word quality parameters using the student's predictions in Equation (4).

In contrast to MATE, which uses the validation set (with aspect labels) to estimate seed weights in an initialization step, our proposed method is an unsupervised approach to modeling and adapting the seed word quality during training. We stop this iterative procedure after the disagreement between the student's and teacher's hard predictions in the training data stops decreasing. We empirically observe that 2-3 rounds are sufficient to satisfy this criterion. This observation also agrees with Khetan et al. (2018), who only run their algorithm for two rounds.

## 4 Experiments

We evaluate our approach to aspect detection on several datasets of product and restaurant reviews.

### 4.1 Experimental Settings

**Datasets.** We train and evaluate our models on Amazon product reviews for six domains (Laptop Bags, Keyboards, Boots, Bluetooth Headsets, Televisions, and Vacuums) from the OPOSUM dataset (Angelidis and Lapata, 2018), and on restaurant reviews in six languages (English, Spanish, French, Russian, Dutch, Turkish) from the SemEval-2016 Aspect-based Sentiment Analysis task (Pontiki et al., 2016). Aspect labels (9-class for product reviews and 12-class for restaurant reviews) are available for each segment[6] of the validation and test sets. The restaurant reviews also come with training aspect labels, which we only use for training the fully supervised models. For a fair comparison, we use exactly the same 30 seed words (per aspect and domain) used in Angelidis and Lapata (2018) for the product reviews and use the same extraction method described in Angelidis and Lapata (2018) to extract 30 seed words for the restaurant reviews. See the supplementary material for more dataset details.

**Experimental Procedure.** For a fair comparison, we use exactly the same pre-processing (tokenization, stemming, and word embedding) and evaluation procedure as in Angelidis and Lapata (2018). For each domain, we train our model on the training set without using any aspect labels, and only use the seed words $G$ via the teacher. For each model, we report the average test performance over 5 different runs with the parameter configuration that achieves best validation performance. As evaluation metric, we use the micro-averaged F1.

**Model Configuration.** For the student network, we experiment with various modeling choices for segment representations: bag-of-words (BOW) classifiers, the unweighted average of word2vec embeddings (W2V), the weighted average of word2vec embeddings using bilinear attention (Luong et al., 2015) (same setting as He et al. (2017); Angelidis and Lapata (2018)), and the average of contextualized word representations obtained from the second-to-last layer of the pre-trained (self-attention based) BERT model (Devlin et al., 2019), which uses multiple self-attention layers (Vaswani et al., 2017) and has been shown to achieve state-of-the-art performance in many downstream NLP applications. For the English product reviews, we use the base uncased BERT model. For the multilingual restaurant reviews, we use the multilingual cased BERT model.[7]

In iterative co-training, we train the student network to convergence in each iteration (which may require more than one epoch over the training data). Moreover, we observed that the iterative process is more stable when we interpolate between weights of the previous iteration and the estimated updates instead of directly applying the estimated seed weight updates (according to Equation (3)).

**Model Comparison.** For a robust evaluation of our approach, we compare the following models and baselines:

- **LDA-Anchors:** The topic model of Lund et al. (2017) using seed words as "anchors."
- **ABAE:** The unsupervised autoencoder of He et al. (2017), where the learned topics were manually mapped to aspects.
- **MATE-*:** The MATE model of Angelidis and Lapata (2018) with various configurations: initialization of the aspect embeddings

in Khetan et al. (2018), which is an alternative form of noise-aware loss functions (Natarajan et al., 2013), is equivalent to our distillation loss: using the log loss as $l(.)$ in Equation (4) of Khetan et al. (2018) yields the cross entropy loss.

[6] In product reviews, elementary discourse units (EDUs) are used as segments. In restaurant reviews, sentences are used as segments.

[7] Both models can be found in https://github.com/google-research/bert/blob/master/multilingual.md. The multilingual cased BERT model is recommended by the authors instead of the multilingual uncased BERT model.

| Method | Product Review Domain | | | | | | |
|---|---|---|---|---|---|---|---|
| | **Bags** | **Keyboards** | **Boots** | **Headsets** | **TVs** | **Vacuums** | **AVG** |
| LDA-Anchors (Lund et al., 2017) | 33.5 | 34.7 | 31.7 | 38.4 | 29.8 | 30.1 | 33.0 |
| ABAE (He et al., 2017) | 38.1 | 38.6 | 35.2 | 37.6 | 39.5 | 38.1 | 37.9 |
| MATE (Angelidis and Lapata, 2018) | 46.2 | 43.5 | 45.6 | 52.2 | 48.8 | 42.3 | 46.4 |
| MATE-unweighted | 41.6 | 41.3 | 41.2 | 48.5 | 45.7 | 40.6 | 43.2 |
| MATE-MT (best performing) | 48.6 | 45.3 | 46.4 | 54.5 | 51.8 | 47.7 | 49.1 |
| Teacher | 55.1 | 52.0 | 44.5 | 50.1 | 56.8 | 54.5 | 52.2 |
| Student-BoW | 57.3 | 56.2 | 48.8 | 59.8 | 59.6 | 55.8 | 56.3 |
| Student-W2V | 59.3 | 57.0 | 48.3 | **66.8** | **64.0** | 57.0 | 58.7 |
| Student-W2V-RSW | 51.3 | 57.2 | 46.6 | 63.0 | 62.1 | 57.1 | 56.2 |
| Student-ATT | 60.1 | 55.6 | 49.9 | 66.6 | 63.4 | 58.2 | 58.9 |
| Student-BERT | **61.4** | **57.5** | **52.0** | 66.5 | 63.0 | **60.4** | **60.2** |

Table 3: Micro-averaged F1 reported for 9-class EDU-level aspect detection in product reviews.

| Method | Restaurant Review Language | | | | | | |
|---|---|---|---|---|---|---|---|
| | **En** | **Sp** | **Fr** | **Ru** | **Du** | **Tur** | **AVG** |
| W2V-Gold | 58.8 | 50.4 | 50.4 | 69.3 | 51.4 | 55.7 | 56.0 |
| BERT-Gold | 63.1 | 51.6 | 50.5 | 64.6 | 53.5 | 55.3 | 56.4 |
| MATE | 41.0 | 24.9 | 17.8 | 18.4 | 36.1 | 39.0 | 29.5 |
| MATE-unweighted | 40.3 | 18.3 | 19.2 | 21.8 | 31.5 | 25.2 | 26.1 |
| Teacher | 44.9 | 41.8 | 34.1 | 54.4 | 40.7 | 30.2 | 41.0 |
| Student-W2V | 47.2 | 40.9 | 32.4 | **59.0** | 42.1 | 42.3 | 44.0 |
| Student-ATT | 47.8 | 41.7 | 32.9 | 57.3 | 44.1 | 45.5 | 44.9 |
| Student-BERT | **51.8** | **42.0** | **39.2** | 58.0 | **43.0** | **45.0** | **46.5** |

Table 4: Micro-averaged F1 reported for 12-class sentence-level aspect detection in restaurant reviews. The fully supervised *-Gold models are not directly comparable with the weakly supervised models.

$A_k$ using the unweighted/weighted average of seed word embeddings and an extra multi-task training objective (MT).[8]

- **Teacher:** Our bag-of-seed-words teacher.
- **Student-*:** Our student network trained with various configurations for the EMB function.
- ***-Gold:** Supervised models trained using ground truth aspect labels, which are only available for restaurant reviews. These models are not directly comparable with the other models and baselines.

### 4.2 Experimental Results

Tables 3 and 4 show the results for aspect detection on product and restaurant reviews, respectively. The rightmost column of each table reports the average performance across the 6 domains/languages.

[8]The multi-task training objective in MATE requires datasets from different domains but same language, thus it cannot be applied in our datasets of restaurant reviews.

**MATE-* models outperform ABAE.** Using the seed words to initialize aspect embeddings leads to more accurate aspect predictions than mapping the learned (unsupervised) topics to aspects.

**LDA-Anchors performs worse than MATE-* models.** Although averages of seed words were used as "anchors" in the "Tandem Anchoring" algorithm, we observed that the learned topics did not correspond to our aspects of interest.

**The teacher effectively leverages seed words.** By leveraging the seed words in a more direct way, Teacher is able to outperform the MATE-* models. Thus, we can use Teacher's predictions as supervision for the student, as we describe next.

**The student outperforms the teacher.** Student-BoW outperforms Teacher: the two models have the same architecture but Teacher only considers seed words; regularizing Student's weights encourages Student to mimic the noisy aspect predictions of Teacher by also considering non-seed words for aspect detection. The benefits of our dis-

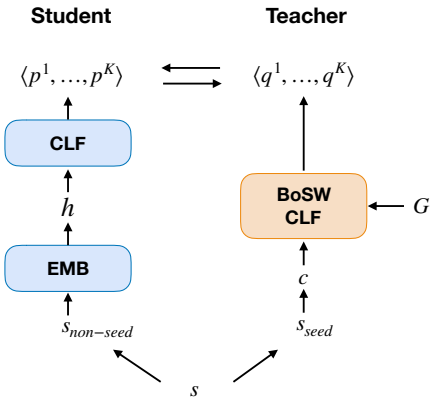


Figure 3: Our weakly supervised co-training approach when seed words are removed from the student's input (RSW baseline). Segment $s_{non-seed}$ is an edited version of $s$, where we replace each seed word in $s$ by an "UNK" special token (like out-of-vocabulary words).

tillation approach are highlighted using neural networks with word embeddings. Student-W2V outperforms both Teacher and Student-BoW, showing that obtaining segment representations as the average of word embeddings is more effective than using bag-of-words representations for this task.

**The student outperforms previous weakly supervised models even in one co-training round.** Student-ATT outperforms MATE-unweighted (by 36.3% in product reviews and by 52.2% in restaurant reviews) even in a single co-training round: although the two models use exactly the same seed words (without weights), pre-trained word embeddings, EMB function, and CLF function, our student-teacher approach leverages the available seed words more effectively as noisy supervision than just for initialization. Also, using our approach, we can explore more powerful methods for segment embedding without the constraint of a fixed word embedding space. Indeed, using contextualized word representations in Student-BERT leads to the best performance over all models.

As expected, our weakly supervised approach does not outperform the fully supervised (*-Gold) models. However, our approach substantially reduces the performance gap between weakly supervised approaches and fully supervised approaches by 62%. The benefits of our student-teacher approach are consistent across all datasets, highlighting the predictive power of seed words across different domains and languages.

**The student leverages non-seed words.** To better understand the extent to which non-seed words can predict the aspects of interest, we experiment with completely removing the seed words from Student-W2V's input during training (Student-W2V-RSW method; see Figure 3). Thus, in this setting, Student-W2V-RSW is forced to only use non-seed words to detect aspects. Note that the co-training assumption of conditionally independent views (Blum and Mitchell, 1998) is satisfied in this setting, where Teacher is only using seed words and Student-W2V is only using non-seed words. Student-W2V-RSW effectively learns to use non-seed words to predict aspects and performs better than Teacher (but worse than Student-W2V, which considers both seed and non-seed words). For additional ablation experiments, see the supplementary material.

| Method | Initial | Iterative |
|---|---|---|
| Product Reviews (AVG) | | |
| MATE | 46.4 | - |
| Teacher / Student-W2V | 52.2 / 58.7 | **58.5 / 59.7** |
| Teacher / Student-BERT | 52.2 / 60.2 | **58.6 / 60.8** |
| Restaurant Reviews (En) | | |
| MATE | 29.5 | - |
| Teacher / Student-W2V | 44.9 / 47.2 | **45.8 / 49.0** |
| Teacher / Student-BERT | 44.9 / 51.8 | **49.8 / 53.4** |

Table 5: Micro-averaged F1 scores during the first round (middle column) and after iterative co-training (right column) in product reviews (top) and restaurant reviews (bottom).

**Iterative co-training copes with noisy words.** Further performance improvement in Teacher and Student-* can be observed with the iterative co-training procedure of Section 3.3. Table 5 reports the performance of Teacher and Student-* after co-training for both product reviews (top) and English restaurant reviews (bottom). (For more detailed, per-domain results, see the supplementary material.) Compared to the initial version of Teacher that does not model the quality of the seed words, iterative co-training leads to estimates of seed word quality that improve Teacher's performance up to 12.3% (in product reviews using Student-BERT).

**A better teacher leads to a better student.** Co-training leads to improved student performance in both datasets (Table 5). Compared to MATE, which uses the validation set to estimate the seed weights as a pre-processing step, we estimate and iteratively adapt the seed weights using the

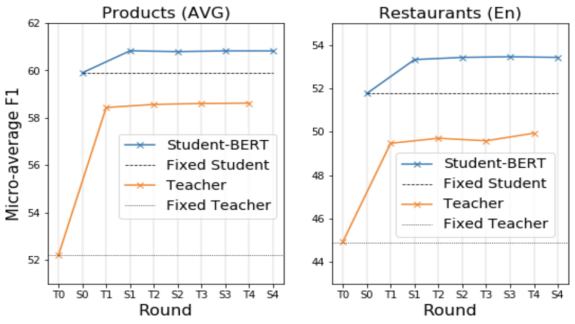


Figure 4: Co-training performance for each round reported for product reviews (left) and restaurant reviews (right). T<$i$>and S<$i$>correspond to the teacher's and student's performance, respectively, at the $i$-th round.

student-teacher disagreement, which substantially improves performance. Across the 12 datasets, Student-BERT leads to an average absolute increase of 14.1 F1 points.

Figure 4 plots Teacher's and Student-BERT's performance after each round of co-training. Most of the improvement for both Teacher and Student-BERT is gained in the first two rounds of co-training: "T0" (in Figure 4) is the initial teacher, while "T1" is the teacher with estimates of seed word qualities, which leads to more accurate predictions, e.g., in segments with multiple seed words from different aspects.

## 5 Conclusions and Future Work

We presented a weakly supervised approach for leveraging a small number of seed words (instead of ground truth aspect labels) for segment-level aspect detection. Our student-teacher approach leverages seed words more directly and effectively than previous weakly supervised approaches. The teacher model provides weak supervision to a student model, which generalizes better than the teacher by also considering non-seed words and by using pre-trained word embeddings. We further show that iterative co-training lets us estimate the quality of the (possibly noisy) seed words. This leads to a better teacher and, in turn, a better student. Our proposed method consistently outperforms previous weakly supervised methods in 12 datasets, allowing for seed words from various domains and languages to be leveraged for aspect detection. Our student-teacher approach could be applied for any classification task for which a small set of seed words describe each class. In future work, we plan to extend our framework to multi-task settings, and to incorporate interaction to learn better seed words.

## Acknowledgments

We thank the anonymous reviewers for their constructive feedback. This material is based upon work supported by the National Science Foundation under Grant No. IIS-15-63785.

## References

Stefanos Angelidis and Mirella Lapata. 2018. Summarizing opinions: Aspect extraction meets sentiment prediction and they are both weakly supervised. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Sanjeev Arora, Rong Ge, Yoni Halpern, David Mimno, Ankur Moitra, David Sontag, Yichen Wu, and Michael Zhu. 2013. A practical algorithm for topic modeling with provable guarantees. In *Proceedings of the International Conference on International Conference on Machine Learning*.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *Proceedings of the International Conference on Learning Representations*.

Jimmy Ba and Rich Caruana. 2014. Do deep nets really need to be deep? In *Advances in Neural Information Processing Systems*.

Maria-Florina Balcan, Avrim Blum, and Ke Yang. 2005. Co-training and expansion: Towards bridging theory and practice. In *Advances in Neural Information Processing Systems*.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the Annual Conference on Computational Learning Theory*.

Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Minmin Chen, Kilian Q Weinberger, and John Blitzer. 2011. Co-training for domain adaptation. In *Advances in Neural Information Processing Systems*.

Kevin Clark, Minh-Thang Luong, Christopher D Manning, and Quoc Le. 2018. Semi-supervised sequence modeling with cross-view training. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Michael Collins and Yoram Singer. 1999. Unsupervised models for named entity classification. In *1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Tommaso Furlanello, Zachary Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. 2018. Born-again neural networks. In *International Conference on Machine Learning*.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2017. An unsupervised neural attention model for aspect extraction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Mohit Iyyer, Anupam Guha, Snigdha Chaturvedi, Jordan Boyd-Graber, and Hal Daumé III. 2016. Feuding families and former friends: Unsupervised learning for dynamic fictional relationships. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Giannis Karamanolakis, Daniel Hsu, and Luis Gravano. 2019. Training neural networks for aspect extraction using descriptive keywords only. In *Proceedings of the Second Learning from Limited Labeled Data Workshop*.

Ashish Khetan, Zachary C Lipton, and Anima Anandkumar. 2018. Learning from noisy singly-labeled data. In *Proceedings of the International Conference on Learning Representations*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Yoon Kim and Alexander M Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.

Pengfei Liu, Shafiq Joty, and Helen Meng. 2015. Fine-grained opinion mining with recurrent neural networks and word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

David Lopez-Paz, Léon Bottou, Bernhard Schölkopf, and Vladimir Vapnik. 2016. Unifying distillation and privileged information. In *Proceedings of the International Conference on Learning Representations*.

Bin Lu, Myle Ott, Claire Cardie, and Benjamin K Tsou. 2011. Multi-aspect sentiment analysis with topic models. In *2011 IEEE International Conference on Data Mining Workshops*. IEEE.

Jeffrey Lund, Connor Cook, Kevin Seppi, and Jordan Boyd-Graber. 2017. Tandem anchoring: A multiword anchor approach for interactive topic modeling. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the 2006 Conference of the North American Chapter of the Association of Computational Linguistics*. Association for Computational Linguistics.

Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. 2013. Learning with noisy labels. In *Advances in Neural Information Processing Systems*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, AL-Smadi Mohammad, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphée De Clercq, et al. 2016. Semeval-2016 task 5: Aspect based sentiment analysis. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*.

Soujanya Poria, Erik Cambria, and Alexander Gelbukh. 2016. Aspect extraction for opinion mining with a deep convolutional neural network. *Knowledge-Based Systems*, 108:42–49.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *https://blog.openai.com/language-unsupervised*.

Sebastian Ruder and Barbara Plank. 2018. Strong baselines for neural semi-supervised learning under domain shift. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Akash Srivastava and Charles Sutton. 2071. Autoencoding variational inference for topic models. In *Proceedings of the International Conference on Learning Representations*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*.

Weiran Wang. 2019. Everything old is new again: A multi-view learning approach to learning using privileged information and distillation. *arXiv preprint arXiv:1903.03694*.

Jason Weston, Samy Bengio, and Nicolas Usunier. 2011. Wsabie: Scaling up to large vocabulary image annotation. In *Twenty-Second International Joint Conference on Artificial Intelligence*.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Towards universal paraphrastic sentence embeddings. *In Proceedings of the International Conference on Learning Representations*.

John Wieting and Kevin Gimpel. 2017. Revisiting recurrent networks for paraphrastic sentence embeddings. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Annual Meeting of the Association for Computational Linguistics*.

Lei Zhang, Shuai Wang, and Bing Liu. 2018. Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, page e1253.

Zhi-Hua Zhou and Ming Li. 2005. Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge & Data Engineering*, (11):1529–1541.

# Integrating Text and Image: Determining Multimodal Document Intent in Instagram Posts

**Julia Kruk**[1]*, **Jonah Lubin**[2]*, **Karan Sikka**[1], **Xiao Lin**[1],
**Dan Jurafsky**[3], **Ajay Divakaran**[1]†
∗equal contribution
[1]SRI International, Princeton, NJ [2]The University of Chicago, Chicago, Illinois
[3]Stanford University, Stanford, CA

## Abstract

Computing author intent from multimodal data like Instagram posts requires modeling a complex relationship between text and image. For example, a caption might evoke an ironic contrast with the image, so neither caption nor image is a mere transcript of the other. Instead they combine—via what has been called *meaning multiplication* Bateman (2014)—to create a new meaning that has a more complex relation to the literal meanings of text and image. Here we introduce a multimodal dataset of 1299 Instagram posts labeled for three orthogonal taxonomies: the authorial intent behind the image-caption pair, the contextual relationship between the literal meanings of the image and caption, and the semiotic relationship between the signified meanings of the image and caption. We build a baseline deep multimodal classifier to validate the taxonomy, showing that employing both text and image improves intent detection by 9.6% compared to using only the image modality, demonstrating the commonality of non-intersective meaning multiplication. The gain with multimodality is greatest when the image and caption diverge semiotically. Our dataset offers a new resource for the study of the rich meanings that result from pairing text and image.

## 1 Introduction

Multimodal social platforms such as Instagram let content creators combine visual and textual modalities. The resulting widespread use of text+image makes interpreting author intent in multimodal messages an important task for NLP for document understanding.

There are many recent language processing studies of images accompanied by basic text labels or captions (Chen et al., 2015; Faghri et al., 2018, inter alia). But prior work on image–text pairs has generally been asymmetric, regarding either image or text as the primary content, and the other as mere complement. Scholars from semiotics as well as computer science have pointed out that this is insufficient; often text and image are not combined by a simple addition or intersection of the component meanings (Bateman, 2014; Marsh and Domas White, 2003; Zhang et al., 2018).

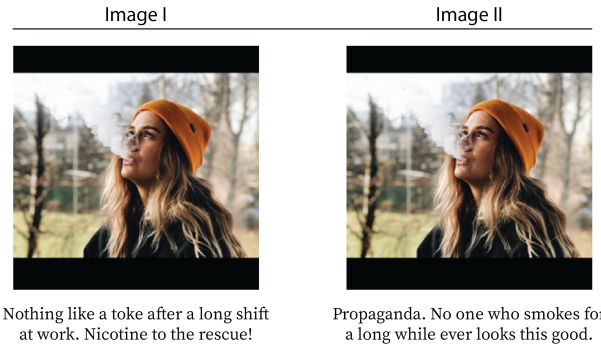


Figure 1: Image-Caption meaning multiplication: A change in the caption completely changes the overall meaning of the image-caption pair.

Rather, determining author intent with text+image content requires a richer kind of meaning composition that has been called *meaning multiplication* (Bateman, 2014): the creation of new meaning through integrating image and text. Meaning multiplication includes simple meaning intersection or concatenation (a picture of a dog with the label "dog", or the label "Rufus"). But it also includes more sophisticated kinds of composition, such as irony or indirection, where the text+image integration requires inference that creates a new meaning. For example in Figure 1, a picture of a young woman smoking is given two different hypothetical captions that result in different composed meanings. In Pairing I, the image and text are parallel, with the picture

* Work done while Julia (from Cornell University) and Jonah were interns at SRI International.

† Corresponding author, ajay.divakaran@sri.com.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4622–4632, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

used to highlight relaxation through smoking. Pairing II uses the tension between her image and the implications of her actions to highlight the dangers of smoking.

Computational models that detect complex relationships between text and image and how they cue author intent could be significant for many areas, including the computational study of advertising, the detection and study of propaganda, and our deeper understanding of many other kinds of persuasive text, as well as allowing NLP applications to news media to move beyond pure text.

To better understand author intent given such meaning multiplication, we create three novel taxonomies related to the relationship between text and image and their combination/multiplication in Instagram posts, designed by modifying existing taxonomies (Bateman, 2014; Marsh and Domas White, 2003) from semiotics, rhetoric, and media studies. Our taxonomies measure the **authorial intent** behind the image-caption pair and two kinds of text-image relations: the **contextual relationship** between the literal meanings of the image and caption, and the **semiotic relationship** between the signified meanings of the image and caption. We then introduce a new dataset, MDID (Multimodal Document Intent Dataset), with 1299 Instagram posts covering a variety of topics, annotated with labels from our three taxonomies.

Finally, we build a deep neural network model for annotating Instagram posts with the labels from each taxonomy, and show that combining text and image leads to better classification, especially when the caption and the image diverge. While our goal here is to establish a computational framework for investigating multimodal meaning multiplication, in other pilot work we have begun to consider some applications, such as using intent for social media event detection and for user engagement prediction. Both these directions highlight the importance of the intent and semiotic structure of a social media posting in determining its influence on the social network as a whole.

## 2 Prior Work

A wide variety of work in multiple fields has explored the relationship between text and image and extracting meaning, although often assigning a subordinate role to either text or images, rather than the symmetric relationship in media such as Instagram posts. The earliest work in the Barthesian tradition focuses on advertisements, in which the text serves as merely another connotative aspect to be incorporated into a larger connotative meaning (Heath et al., 1977). Marsh and Domas White (2003) offer a taxonomy of the relationship between image and text by considering image/illustration pairs found in textbooks or manuals. We draw on their taxonomy, although as we will see, the connotational aspects of Instagram posts require some additions.

For our model of speaker intent, we draw on the classic concept of illocutionary acts (Austin, 1962) to develop a new taxonomy of illocutionary acts focused on the kinds of intentions that tend to occur on social media. For example, we rarely see commissive posts on Instagram and Facebook because of the focus on information sharing and constructions of self-image.

Computational approaches to multi-modal document understanding have focused on key problems such as image captioning (Chen et al., 2015; Faghri et al., 2018), visual question answering (Goyal et al., 2017; Zellers et al., 2018; Hudson and Manning, 2019), or extracting the literal or connotative meaning of a post (Soleymani et al., 2017). More recent work has explored the role of image as context for interaction and pragmatics, either in dialog (Mostafazadeh et al., 2016, 2017), or as a prompt for users to generate descriptions (Bisk et al., 2019). Another important direction has looked at an image's *perlocutionary force* (how it is perceived by its audience), including aspects such as memorability (Khosla et al., 2015), saliency (Bylinskii et al., 2018), popularity (Khosla et al., 2014) and virality (Deza and Parikh, 2015; Alameda-Pineda et al., 2017).

Some prior work has focused on intention. Joo et al. (2014) and Huang and Kovashka (2016) study prediction of intent behind politician portraits in the news. Hussain et al. (2017) study the understanding of image and video advertisements, predicting topic, sentiment, and intent. Alikhani et al. (2019) introduce a corpus of the coherence relationships between recipe text and images. Our work builds on Siddiquie et al. (2015), who focused on a single type of intent (detecting politically persuasive video on the internet) and even more closely on Zhang et al. (2018), who study visual rhetoric as interaction between the image and the text slogan in advertisements. They categorize image-text relationships into par-

allel equivalent (image-text deliver same point at equal strength), parallel non-equivalent (image-text deliver the same point at different levels) and non-parallel (text or image alone is insufficient in point delivery). They also identify the novel issue of understanding the complex, non-literal ways in which text and image interacts. Weiland et al. (2018) study the non-literal meaning conveyed by image-caption pairs and draw on a knowledge-base to generate the gist of the image-caption pair.

## 3 Taxonomies

As Berger (1972) points out in discussing the relationship between one image and its caption:

> It is hard to define exactly how the words have changed the image but undoubtedly they have. (p. 28).

We propose three taxonomies in an attempt to answer Berger's implicit question, two (contextual and semiotic) to capture different aspects of the relationship between the image and the caption, and one to capture speaker intent.

### 3.1 Intent Taxonomy

The proposed intent taxonomy is a generalization and elaboration of existing rhetorical categories pertaining to illocution, that targets multimodal social networks like Instagram. We developed a set of eight illocutionary intents from our examination and clustering of a large body of representative Instagram content, informed by previous studies of intent in Instagram posts. There is some overlap between categories; to bound the burden on the annotators, however, we asked them to identify intent for the image-caption pairing as a whole and not for the individual components

For example drawing on Goffman's idea of the presentation of self (Goffman, 1978), Mahoney et al. (2016) in their study of Scottish political Instagram posts define acts like Presentation of Self, which, following Hogan (2010) we refer to as *exhibition*, or Personal Political Expression, which we generalize to *advocative*. Following are our final eight labels; Figure 2 shows some examples.

1. **advocative**: advocate for a figure, idea, movement, etc.
2. **promotive**: promote events, products, organizations etc.

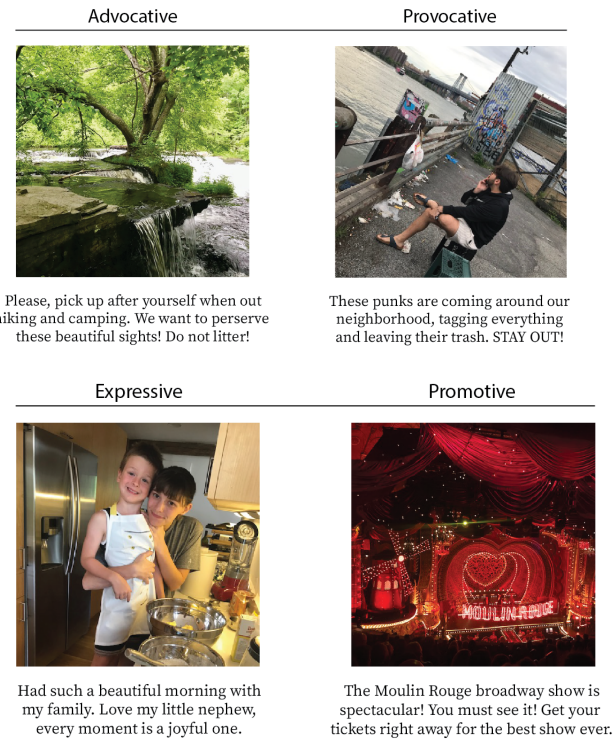


Figure 2: Examples of multimodal document intent: advocative, provocative, expressive and promotive content

3. **exhibitionist**: create a self-image reflecting the person, state etc. for the user using selfies, pictures of belongings (e.g. pets, clothes) etc.
4. **expressive**: express emotion, attachment, or admiration at an external entity or group.
5. **informative**: relay information regarding a subject or event using factual language.
6. **entertainment**: entertain using art, humor, memes, etc.
7. **provocative/discrimination**: directly attack an individual or group.
8. **provocative/controversial**: be shocking.

### 3.2 The Contextual Taxonomy

The contextual relationship taxonomy captures the relationship between the literal meanings of the image and text. We draw on the three top-level categories of the Marsh and Domas White (2003) taxonomy, which distinguished images that are minimally related to the text, highly related to the text, and related but going beyond it. These three classes— reflecting Marsh et al.'s primary interest in illustration—frame the image only as subordinate to the text. We slightly generalize the three

| Intent | |
|---|---|
| *Category* | *# Samples* |
| Provocative | 84 |
| Informative | 119 |
| Advocative | 97 |
| Entertainment | 310 |
| Expositive | 237 |
| Expressive | 95 |
| Promotive | 162 |

| Semiotic | |
|---|---|
| *Category* | *# Samples* |
| Divergent | 115 |
| Additive | 277 |
| Parallel | 712 |

| Contextual Relationship | |
|---|---|
| *Category* | *# Samples* |
| Minimal | 372 |
| Close | 585 |
| Transcendent | 147 |

Table 1: Counts of different labels in the Multimodal Document Intent Dataset (MDID).

top-level categories taxonomy of Marsh and Domas White (2003) to make them symmetric for the Instagram domain:

**Minimal Relationship:** The literal meanings of the caption and image overlap very little. For example, a selfie of a person at a waterfall with the caption "selfie". While such a terse caption does nevertheless convey a lot of information, it still leaves out details such as the location, description of the scene, etc. that are found in typical loquacious Instagram captions.

**Close Relationship:** The literal meanings of the caption and the image overlap considerably. For example, a selfie of a person at a crowded waterfall, with the caption "Selfie at Hemlock falls on a crowded sunny day".

**Transcendent Relationship:** The literal meaning of one modality picks up and expands on the literal meaning of the other. For example, a selfie of a person at a crowded waterfall with the caption "Selfie at Hemlock Falls on a sunny and crowded day. Hemlock falls is a popular picnic spot. There are hiking and biking trails, and a great restaurant 3 miles down the road ...".

Note that while the labels "minimal" and "close" could be thought of as lying on a continuous scale indicating semantic overlap, the label "transcendent" indicates an expansion of the meaning that cannot be captured by such a continuous scale.

### 3.3 The Semiotic Taxonomy

The contextual taxonomy described above does not deal with the more complex forms of "meaning multiplication" illustrated in Figure 1. For example, an image of three frolicking puppies with the caption "My happy family," sends a message of pride in one's pets that is not directly reflected in either modality taken by itself. First, it forces the reader to step back and consider what is being signified by the image and the caption, in effect offering a meta-comment on the text-image relation. Second, there is a tension between what is signified (a family and a litter of young animals respectively) that results in a richer idiomatic meaning.

Our third taxonomy therefore captures the relationship between what is *signified* by the respective modalities, their *semiotics*. We draw on the earlier 3-way distinction of Kloepfer (1977) as modeled by Bateman (2014) and the two-way (parallel vs. non-parallel) distinction of Zhang et al. (2018) to classify the semiotic relationship of image/text pairs as divergent, parallel and additive. A *divergent* relationship occurs when the image and text semiotics pull in opposite directions, creating a gap between the meanings suggested by the image and text. A *parallel* relationship occurs when the image and text independently contribute to the same meaning. An *additive* relationship occurs when the image and text semiotics amplify or modify each other.

The semiotic classification is not always homologous to the contextual. For example, an image of a mother feeding her baby with a caption "My new small business needs a lot of tender loving care", would have a minimal contextual relationship. Yet because both signify loving care and the image intensifies the caption's sentiment, the semiotic relationship is additive. Or a lavish formal farewell scene at an airport with the caption "Parting is such sweet sorrow", has a close contextual relationship because of the high overlap in literal meaning, but the semiotics would be additive, not parallel, since the image shows only the leave-taking, while the caption suggests love (or ironic lack thereof) for the person leaving.

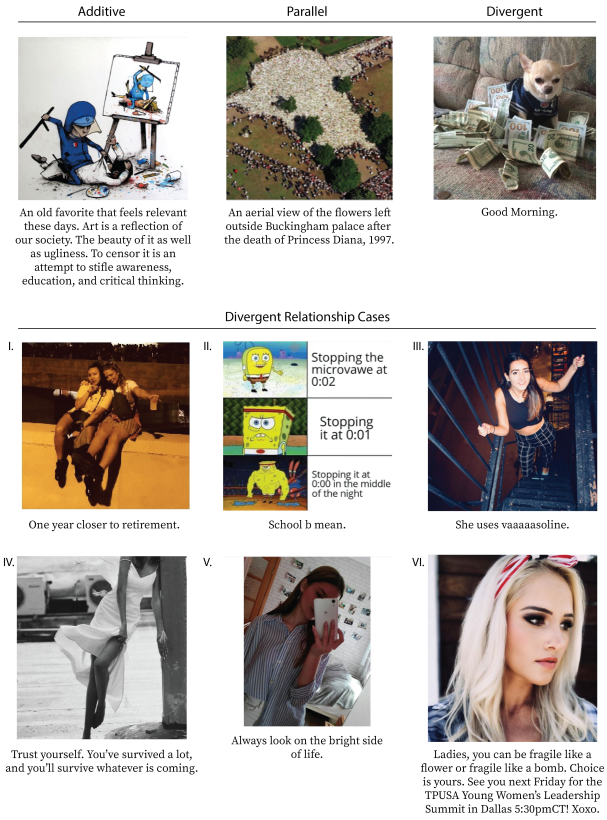


Figure 3: The top three images exemplify the semiotic categories. Images I-VI show instances of divergent semiotic relationships.

Figure 3 further illustrates the proposed semiotic classification. The first three image-caption pairs (ICP's) exemplify the three semiotic relationships. To give further insights into the rich complexity of the divergent category, the six ICP's below showcase the kinds of divergent relationships we observed most frequently on Instagram.

ICP I exploits the tension between the reference to retirement expressed in the caption and the youth projected by the two young women in the image to convey irony and thus humor in what is perhaps a birthday greeting or announcement. Many ironic and humorous posts exhibit divergent semiotic relationships. ICP II has the structure of a classic Instagram meme, where the focus is on the image, and the caption is completely unrelated to the image. This is also exhibited in the divergent "Good Morning" caption in the top row. ICP III is an example of a divergent semiotic relationship within an exhibitionist post. A popular communicative practice on Instagram is to combine selfies with a caption that is some sort of inside joke. The inside joke in ICP III is a lyric from a song a group of friends found funny and discussed the night this photo was taken. ICP IV is an aesthetic photo of a young woman, paired with a caption that has no semantic elements in common with the photo. The caption may be a prose excerpt, the author's reflection on what the image made them think or feel, or perhaps just a pairing of pleasant visual stimulus with pleasant literary material. This divergent relationship is often found in photography, artistic and other entertainment posts. ICP V uses one of the most common divergent relationships, in which exhibitionist visual material is paired with reflections or motivational captions. ICP V is thus similar to ICP III, but without the inside jokes/hidden meanings common to ICP III. ICP VI is an exhibitionist post that seems to be common recently among public figures on Instagram. The image appears to be a classic selfie or often a professionally taken image of the individual, but the caption refers to that person's opinions or agenda(s). This relationship is divergent—there are no common semantic elements in the image and caption—but the pair paints a picture of the individual's current state or future plans.

## 4 The MDID Dataset

Our dataset, MDID (the *Multimodal Document Intent Dataset*) consists of 1299 public Instagram posts that we collected with the goal of developing a rich and diverse set of posts for each of the eight illocutionary types in our intent taxonomy. For each intent we collected at least 16 hashtags or users likely to yield a high proportion of posts that could be labeled by that heading.

For the *advocative* intent, we selected mostly hashtags advocating and spanning political or social ideology such as #pride and #maga. For the *promotive* intent we relied on the #ad tag that Instagram has recently begun requiring for sponsored posts. For *exhibitionist* intent we used tags that focused on the self as the most important aspect of the post such as #selfie and #ootd (outfit of the day). The *expressive* posts were retrieved via tags that actively expressed a stance or an affective intent, such as #lovehim or #merrychristmas. Informative posts were taken from informative accounts such as news websites. Entertainment posts drew on an eclectic group of tags such as #meme, #earthporn, #fatalframes. Finally, provocative posts were extracted via tags that either expressed a controversial or provocative message or that would draw people into being influ-

enced or provoked by the post (#redpill, #antifa, #eattherich, #snowflake).

**Data Labeling:** Data was pre-processed (for example to convert all albums to single image-caption pairs). We developed a simple annotation toolkit that displayed an image–caption pair and asked the annotator to confirm whether the pair was relevant (contains both an image and text in English) and if so to identify the post's intent (advocative, promotive, exhibitionist, expressive, informative, entertainment, provocative), contextual relationship (minimal, close, transcendent), and semiotic relationship (divergent, parallel, additive). Two of the authors collaborated on the labelers manual and then labeled the data by consensus, and any label on which the authors disagreed after discussion was removed. Dataset statistics are shown in Table 1; see https://www.ksikka.com/document_intent.html for the data.

## 5 Computational Model

We train and test a deep convolutional neural network (DCNN) model on the dataset, both to offer a baseline model for users of the dataset, and to further explore our hypothesis about meaning multiplication.

Our model can take as input either image (Img), text (Txt) or both (Img + Txt), and consists of modality specific encoders, a fusion layer, and a class prediction layer. We use the ResNet-18 network pre-trained on ImageNet as the image encoder (He et al., 2016). For encoding captions, we use a standard pipeline that employs a RNN model on word embeddings. We experiment with both word2vec type (word token-based) embeddings trained from scratch (Mikolov et al., 2013) and pre-trained character-based contextual embeddings (ELMo) (Peters et al., 2018). For our purpose ELMo character embeddings are more useful since they increase robustness to noisy and often misspelled Instagram captions. For the combined model, we implement a simple fusion strategy that first linearly projects encoded vectors from both the modalities in the same embedding space and then adds the two vectors. Although naive, this strategy has been shown to be effective at a variety of tasks such as Visual Question Answering (Nguyen and Okatani, 2018) and image-caption matching (Ahuja et al., 2018). We then use the fused vector to predict class-wise scores using a fully connected layer.

## 6 Experiments

We evaluate our models on predicting intent, semiotic relationships, and image-text relationships from Instagram posts, using image only, text only, and both modalities.

### 6.1 Dataset, Evaluation and Implementation

We use the 1299-sample MDID dataset (section 4). We only use corresponding image and text information for each post and do not use other meta-data to preserve the focus on image-caption joint meaning. We perform basic pre-processing on the captions such as removing stopwords and non-alphanumeric characters. We do not perform any pre-processing for images.

Due to the small dataset, we perform 5-fold cross-validation for our experiments reporting average performance across all splits. We report classification accuracy (ACC) and also area under the ROC curve (AUC) (since AUC is more robust to class-skew), using macro-average across all classes (Jeni et al., 2013; Stager et al., 2006).

We use a pre-trained ResNet-18 model as the image encoder. For word token based embeddings we use 300 dimensional vectors trained from scratch. For ELMo we use a publicly available API [1] and use a pre-trained model with two layers resulting in a 2048 dimensional input. We use a bi-directional GRU as the RNN model with 256 dimensional hidden layers. We set the dimensionality of the common embedding space in the fusion layer to 128. In case there is a single modality, the fusion layer only projects features from that modality. We train with the Adam optimizer with a learning rate of 0.00005, which is decayed by 0.1 after every 15 epochs. We report results with the best model selected based on performance on a mini validation set.

### 6.2 Quantitative Results

We show results in Table 2. For the intent taxonomy images are more informative than (word2vec) text (76% for Img vs 72.7% for Txt-emb) but with ELMo text outperforms just using images (82.6% for Txt-ELMo, 76.0% for Img). ELMo similarly improves performance on the contextual taxonomy but not the semiotic taxonomy.

For the semiotic taxonomy, ELMo and word2vec embeddings perform similarly, (67.8%

[1] https://github.com/allenai/allennlp/blob/master/tutorials/how_to/elmo.md

| Method | Intent | | Semiotic | | Contextual | |
|---|---|---|---|---|---|---|
| | *ACC* | *AUC* | *ACC* | *AUC* | *ACC* | *AUC* |
| Chance | 28.1 | 50.0 | 64.5 | 50.0 | 53.0 | 50.0 |
| Img | 42.9 (±0.0) | 76.0 (±0.5) | 61.5 (±0.0) | 59.8 (±3.0) | 52.5 (±0.0) | 62.5 (±1.3) |
| Txt-emb | 42.9 (±0.0) | 72.7 (±1.5) | 58.9 (±0.0) | 67.8 (±1.7) | 60.7 (±0.5) | 74.9 (±3.0) |
| Txt-ELMo | 52.7 (±0.0) | 82.6 (±1.2) | 61.7 (±0.0) | 66.5 (±1.9) | 65.4 (±0.0) | 78.5 (±2.1) |
| Img + Txt-emb | 48.1 (±0.0) | 80.8 (±1.2) | 60.4 (±0.0) | 69.7 (±1.8) | 60.8 (±0.0) | 76.0 (±2.5) |
| Img + Txt-ELMo | 56.7 (±0.0) | 85.6 (±1.3) | 61.8 (±0.0) | 67.8 (±1.8) | 63.6 (±0.5) | 79.0 (±1.4) |

Table 2: Table showing results with various DCNN models– image-only (Img), text-only (Txt-emb and Txt-ELMo), and combined model (Img + Txt-emb and Img + Txt-ELMo). Here *emb* is the model using standard word (token) based embeddings, while *ELMo* is the pre-trained ELMo based word embeddings (Peters et al., 2018). The numbers in Table2 are standard deviations across 5 folds.

for Txt-emb vs. 66.5% for Txt-ELMo), suggesting that individual words are sufficient for the semiotic labeling task, and the presence of the sentence context (as in ELMo) is not needed.

Combining visual and textual modalities helps across the board. For example, for intent taxonomy the joint model Img + Txt-ELMo achieves 85.6% compared to 82.6% for Txt-ELMo. Images seem to help even more when using a word embedding based text model (80.8% for Img + Txt-emb vs. 72.7% for Txt-emb). Joint models also improve over single-modality on labeling the image-text relationship and the semiotic taxonomy. We show class-wise performances with the single- and multi-modality models in Table 3. It is particularly interesting that in the semiotic taxonomy, multi-modality helps the most with divergent semiotics (gain of 4.4% compared to the image-only model).

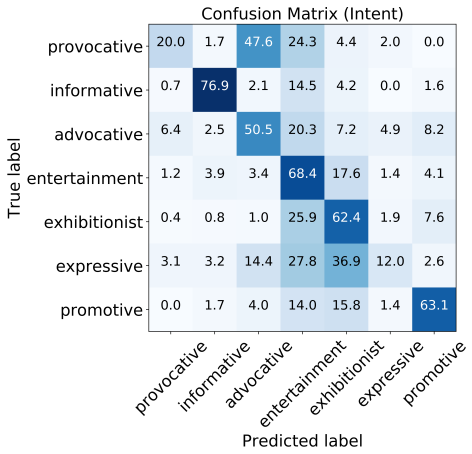


Figure 4: Confusion between intent classes for the intent classification task. The confusion matrix was obtained using the Img + Txt-ELMo model and the results are averaged over the 5 splits.

### 6.3 Discussion

In general, using both text and image is helpful, a fact that is unsurprising since combinations of text and image are known to increase performance on tasks such as predicting post popularity or user personality (Hessel et al., 2017; Wendlandt et al., 2017). Most telling, however, were the differences in this helpfulness across items. In the semiotic category the greatest gain came when the text-image semiotics were "divergent". By contrast, multimodal models help less when the image and text are additive, and help the least when the image and text are parallel and provide less novel information. Similarly, with contextual relationships, multimodal analysis helps the most with the "minimal" category (1.6%). This further supports the idea that on social media such as Instagram, the relation between image and text can be richly divergent and thereby form new meanings.

The category confusion matrix in Figure 4 provides further insights. The least confused category is informative. Informative posts are least similar to the rest of Instagram, since they consist of detached, objective posts, with little use of first person pronouns like "I" or "me." Promotive posts are also relatively easy to detect, since they are formally informative, telling the viewer the advantages and logistics of an item or event, with the addition of a persuasive intent reflecting the poster's personal opinions ("I love this watch".). We found that the *entertainment* label is often misapplied; perhaps to some extent all posts have a goal of entertaining, and any analysis must account for this filter of entertainment. The *Exhibitionist* intent tends to be predicted well, likely due to its visual and textual signifiers of individuality (e.g. selfies are almost always exhibitionist, as are captions

| Intent | | | |
|---|---|---|---|
| Class | Img | Txt-ELMo | Img+ Txt-ELMo |
| Provocative | 85.5 | 84.1 | 90.0 |
| Informative | 77.0 | 93.9 | 92.8 |
| Advocative | 84.8 | 82.4 | 87.4 |
| Entertainment | 69.0 | 78.6 | 80.5 |
| Exhibitionist | 81.7 | 78.7 | 84.9 |
| Expressive | 57.9 | 72.0 | 73.2 |
| Promotive | 76.3 | 88.5 | 90.1 |
| Mean | 76.0 | 82.6 | 85.6 |

| Semiotic | | | |
|---|---|---|---|
| Class | Img | Txt-emb | Img+ Txt-ELMo |
| Divergent | 69.8 | 72.7 | 77.1 |
| Additive | 55.0 | 66.7 | 68.2 |
| Parallel | 54.5 | 64.3 | 64.0 |
| Mean | 59.8 | 67.8 | 69.7 |

| Contextual | | | |
|---|---|---|---|
| Class | Img | Txt-ELMo | Img+ Txt-ELMo |
| Minimal | 60.9 | 79.7 | 81.3 |
| Close | 60.5 | 73.8 | 74.6 |
| Transcendent | 66.1 | 82.0 | 81.2 |
| Mean | 62.5 | 78.5 | 79.0 |

Table 3: Class-wise results (AUC) for the three taxonomies with different DCNN models on the MDID dataset. Except for the semiotic taxonomy we used ELMo text representations (based on the performance in Table 2).

like "I love my new hair"). There is a great deal of confusion, however, between the expressive and exhibitionist categories, since the only distinction lies in whether the post is about a general topic or about the poster, and between the provocative and advocative categories, perhaps because both often seek to prove points in a similar way.

With the contextual and semiotic taxonomies, some good results are obtained with text alone. In the "transcendent" contextual case, it is not necessarily surprising that using text alone enables 82% AUC because whenever a caption is really long or has many adverbs, adjectives or abstract concepts, it is highly likely to be transcendent. In the "divergent" semiotics case, we were surprised that text alone would predict divergence with 72.7% AUC. Examining these cases showed that many of them had lexical cues suggesting irony or sarcasm, allowing the system to infer that the image will diverge in keeping with the irony. There is, however, a consistent improvement when both modalities are used for both taxonomies.

### 6.4 Sample Outputs

We show some sample successful outputs of the (multimodal) model in Figure 5, in which the highest probability class in each of the three dimension corresponds to our gold labels. The top-left Image-caption pair (Image I) is classified as exhibitionist, closely followed by expressive; it is a picture of someone's home with a caption describing a domestic experience. The semiotic relationship is classified as additive; the image and caption together signify the concept of spending winter at home with pets before the fireplace. The contextual relationship is classified as transcendental; the caption indeed goes well beyond the image.

The top-right image-caption pair (Image II) is classified as entertainment; the image caption pair works as an ironic reference to dancing ("yeet") grandparents, who are actually reading, in language used usually by young people that a typical grandparent would never use. The semiotic relationship is classified as divergent and the contextual relationship is classified as minimal; there is semantic and semiotic divergence of the image-caption pair caused by the juxtaposition of youthful references with older people.

To further understand the role of meaning multiplication, we consider the change in intent and semiotic relationships when the same image of the British Royal Family is matched with two different captions in the bottom row of Figure 5 (Image IV). In both cases the semiotic relationship is parallel, perhaps due to the match between the multi-figure portrait setting and the word *family*. But the other two dimensions show differences. When the caption is "the royal family" our system classifies the intent as entertainment; presumably such pictures and caption pairs often appear on Instagram intending to entertain. But when the caption is "my happy family" the intent is classified as expressive, perhaps due to the family pride expressed in the caption.

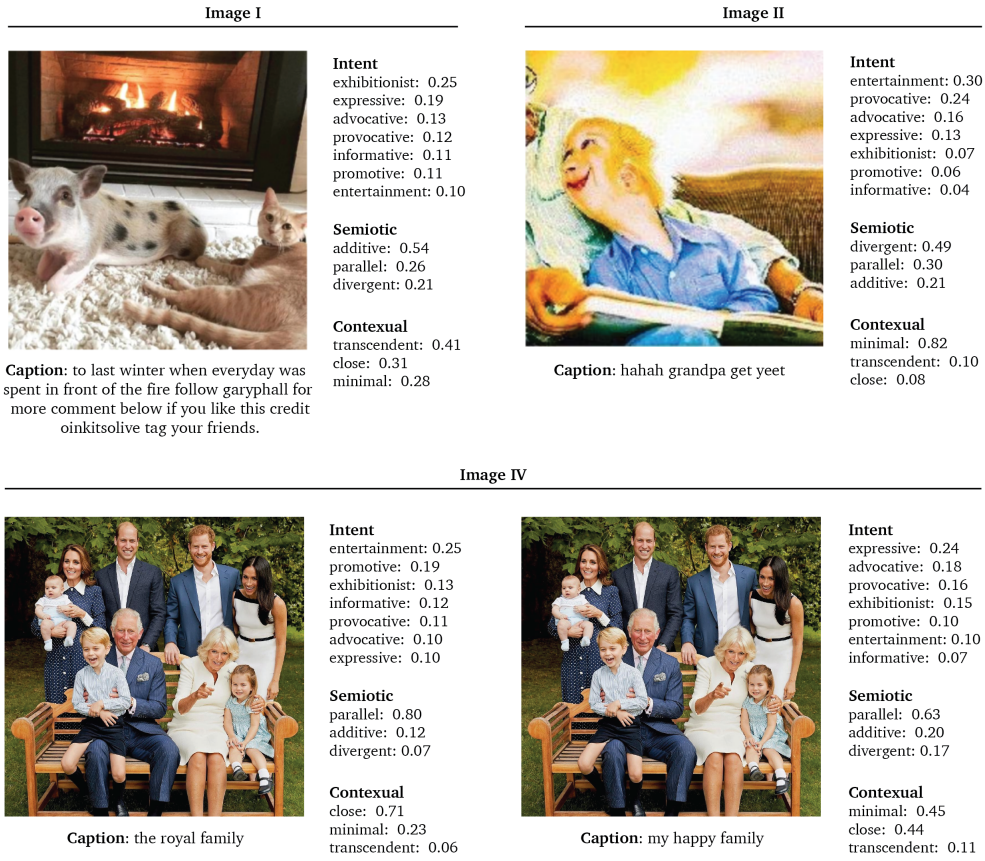


Figure 5: Sample successful output predictions for the three taxonomies, showing ranked classes and predicted probabilities. In images IV the same image when paired with a different caption gives rise to a different intent.

## 7 Conclusion

We have proposed a model to capture the complex *meaning multiplication* relationship between image and text in multimodal Instagram posts. Our three new taxonomies, adapted from the media and semiotic literature, allow the literal, semiotic, and illocutionary relationship between text and image to be coded. Of course our new dataset and the baseline classifier models are just a preliminary effort, and future work will need to examine larger datasets, consider additional data such as hashtags, richer classification schemes, and more sophisticated classifiers. Some of these may be domain-specific. For example, Alikhani et al. (2019) show how to develop rich coherence relations that model the contextual relationship between recipe text and accompanying images (specific versions of Elaboration or Exemplification such as "Shows a tool used in the step but not mentioned in the text"). Expanding our taxonomies with richer sets like these is an important goal. Nonetheless, the fact that we found multimodal classification to be most helpful in cases where the image and text diverged semiotically points out the importance of these complex relations, and our taxonomies, dataset, and tools should provide impetus for the community to further develop more complex models of this important relationship.

## 8 Acknowledgment

This project is sponsored by the Office of Naval Research (ONR) under the contract number N00014-17C-1008 and by DARPA Communicating with Computers (CwC) program under ARO prime contract number W911NF- 15-1-0462. Disclaimer: The views, opinions, and/or findings expressed are those of the author(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. We would like to thank the reviewers for their incisive critique. We would like to thank Prof. Joyce Chai. Ajay Divakaran would like to dedicate this paper to his mother Mrs. Bharathi Divakaran.

## References


Karuna Ahuja, Karan Sikka, Anirban Roy, and Ajay Divakaran. 2018. Understanding visual ads by aligning symbols and objects using co-attention. *arXiv preprint arXiv:1807.01448*.

Xavier Alameda-Pineda, Andrea Pilzer, Dan Xu, Nicu Sebe, and Elisa Ricci. 2017. Viraliency: Pooling local virality. In *CVPR*.

Malihe Alikhani, Sreyasi Nag Chowdhury, Gerard de Melo, and Matthew Stone. 2019. CITE: A corpus of image–text discourse relations. In *NAACL*.

John Langshaw Austin. 1962. *How to Do Things with Words*. Clarendon Press.

John Bateman. 2014. *Text and image: A critical introduction to the visual/verbal divide*. Routledge.

John Berger. 1972. *Ways of Seeing*. Penguin.

Yonatan Bisk, Jan Buys, Karl Pichotta, and Yejin Choi. 2019. Benchmarking hierarchical script knowledge. In *Proceedings of NAACL 2019*, pages 4077–4085.

Zoya Bylinskii, Tilke Judd, Aude Oliva, Antonio Torralba, and Frédo Durand. 2018. What do different evaluation metrics tell us about saliency models? *PAMI*.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. 2015. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*.

Arturo Deza and Devi Parikh. 2015. Understanding image virality. In *CVPR*, pages 1818–1826.

Fartash Faghri, David J Fleet, Jamie Ryan Kiros, and Sanja Fidler. 2018. Vse++: Improving visual-semantic embeddings with hard negatives. In *BMVC*.

Erving Goffman. 1978. *The presentation of self in everyday life*. Anchor Books, New York, NY.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *CVPR*, 2, page 3.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*, pages 770–778.

Stephen Heath et al. 1977. Image-music-text. *Fontana, London*.

Jack Hessel, Lillian Lee, and David Mimno. 2017. Cats and captions vs. creators and the clock: Comparing multimodal content to context in predicting relative popularity. In *WWW*, pages 927–936.

Bernie Hogan. 2010. The presentation of self in the age of social media: Distinguishing performances and exhibitions online. *Bulletin of Science, Technology & Society*, 30(6):377–386.

Xinyue Huang and Adriana Kovashka. 2016. Inferring visual persuasion via body language, setting, and deep features. In *CVPRW*, pages 73–79.

Drew A. Hudson and Christopher D. Manning. 2019. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6700–6709.

Zaeem Hussain, Mingda Zhang, Xiaozhong Zhang, Keren Ye, Christopher Thomas, Zuha Agha, Nathan Ong, and Adriana Kovashka. 2017. Automatic understanding of image and video advertisements. In *CVPR*, pages 1100–1110. IEEE.

László A Jeni, Jeffrey F Cohn, and Fernando De La Torre. 2013. Facing imbalanced data–recommendations for the use of performance metrics. In *ACII*, pages 245–251. IEEE.

Jungseock Joo, Weixin Li, Francis F Steen, and Song-Chun Zhu. 2014. Visual persuasion: Inferring communicative intents of images. In *CVPR*, pages 216–223.

Aditya Khosla, Atish Das Sarma, and Raffay Hamid. 2014. What makes an image popular? In *WWW*, pages 867–876. ACM.

Aditya Khosla, Akhil S Raju, Antonio Torralba, and Aude Oliva. 2015. Understanding and predicting image memorability at a large scale. In *ICCV*, pages 2390–2398.

R Kloepfer. 1977. Komplementaritat von Sprache und Bild, Am Beispiel von Comic, Karikatur, und Reklame. In R. Posner and H.-P. Reinecke, editors, *Zeichenprozesse, Semiotische Forschung in den Einzelwissenschaften*, pages 129–145. Athenaum.

Jamie Mahoney, Tom Feltwell, Obinna Ajuruchi, and Shaun Lawson. 2016. Constructing the visual online political self: an analysis of Instagram use by the Scottish Electorate. In *CHI*, pages 3339–3351. ACM.

Emily E Marsh and Marilyn Domas White. 2003. A taxonomy of relationships between images and text. *Journal of Documentation*, 59(6):647–672.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Nasrin Mostafazadeh, Chris Brockett, Bill Dolan, Michel Galley, Jianfeng Gao, Georgios Spithourakis, and Lucy Vanderwende. 2017. Image-grounded conversations: Multimodal context for

natural question and response generation. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing*, pages 462–472.

Nasrin Mostafazadeh, Ishan Misra, Jacob Devlin, Margaret Mitchell, Xiaodong He, and Lucy Vanderwende. 2016. Generating natural questions about an image. In *Proceedings of the ACL 2016*, pages 1802–1813.

Duy-Kien Nguyen and Takayuki Okatani. 2018. Improved fusion of visual and language representations by dense symmetric co-attention for visual question answering. *arXiv preprint arXiv:1804.00775*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Behjat Siddiquie, David Chisholm, and Ajay Divakaran. 2015. Exploiting multimodal affect and semantics to identify politically persuasive web videos. In *ICMI*, pages 203–210.

Mohammad Soleymani, David Garcia, Brendan Jou, Björn Schuller, Shih-Fu Chang, and Maja Pantic. 2017. A survey of multimodal sentiment analysis. *IVC*, 65:3–14.

Mathias Stager, Paul Lukowicz, and Gerhard Troster. 2006. Dealing with class skew in context recognition. In *ICDCSW*, pages 58–58. IEEE.

Lydia Weiland, Ioana Hulpus, Simone Paolo Ponzetto, Wolfang Effelsberg, and Laura Dietz. 2018. Knowledge-rich image gist understanding beyond literal meaning. *arXiv preprint 1904.08709*.

Laura Wendlandt, Rada Mihalcea, Ryan L. Boyd, and James W. Pennebaker. 2017. Multimodal analysis and prediction of latent user dimensions. In *International Conference on Social Informatics*, pages 323–340.

Rowan Zellers, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2018. From recognition to cognition: Visual commonsense reasoning. *arXiv preprint arXiv:1811.10830*.

Mingda Zhang, Rebecca Hwa, and Adriana Kovashka. 2018. Equal but not the same: Understanding the implicit relationship between persuasive images and text. *arXiv preprint arXiv:1807.08205*.

# Neural Conversation Recommendation with Online Interaction Modeling

**Xingshan Zeng**[1,2], **Jing Li**[3]*, **Lu Wang**[4], **Kam-Fai Wong**[1,2]
[1]The Chinese University of Hong Kong, Hong Kong, China
[2]MoE Key Laboratory of High Confidence Software Technologies, China
[3]Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China
[4]Khoury College of Computer Sciences, Northeastern University, Boston, United States
[1,2]{xszeng,kfwong}@se.cuhk.edu.hk
[3]jing-amelia.li@polyu.edu.hk, [4]luwang@ccs.neu.edu

## Abstract

The prevalent use of social media leads to a vast amount of online conversations being produced on a daily basis. It presents a concrete challenge for individuals to better discover and engage in social media discussions. In this paper, we present a novel framework to automatically recommend conversations to users based on their prior conversation behaviors. Built on neural collaborative filtering, our model explores deep semantic features that measure how a user's preferences match an ongoing conversation's context. Furthermore, to identify salient characteristics from interleaving user interactions, our model incorporates graph-structured networks, where both replying relations and temporal features are encoded as conversation context. Experimental results on two large-scale datasets collected from Twitter and Reddit show that our model yields better performance than previous state-of-the-art models, which only utilize lexical features and ignore past user interactions in the conversations.

## 1 Introduction

Social media has profoundly revolutionized people's social interactions, as many individuals now turn to online platforms to voice opinions and exchange ideas. Meanwhile, the abundance of information brings the problem of information explosion — the huge volume of online discussions produced every day has far outpaced any individual's capability of digesting them. It is hence difficult for one to discover online discussions that are potentially of interest. To address this issue, we study the problem of online conversation recommendation, with the goal of identifying conversations that fit a user's preferences, hence likely to result in the user's future engagement.

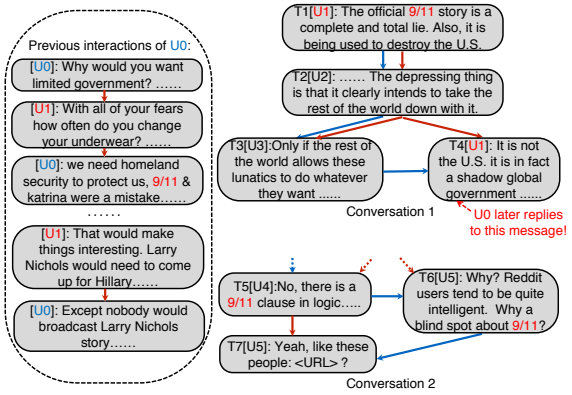


Figure 1: Two Reddit conversation snippets on the right. User $U_0$, whose historical interactions with another user $U_1$ shown on the left, only engages in Conversation 1 (which is initialized by $U_1$), but not Conversation 2 ($U_1$ does not participate in). Red arrows indicate in-reply-to relations, and blue arrows depict chronological orders.

In previous studies, it has been shown that effective online conversation recommendation has the potential to produce more positive online social interaction experience (Chen et al., 2011; Zeng et al., 2018). Prior work on this subject has focused on post-level recommendation (Yan et al., 2012; Chen et al., 2012), or conversation-level suggestion with handcrafted features (Chen et al., 2011) and word co-occurrence patterns (Zeng et al., 2018). Nevertheless, they ignore the useful information embedded in replying relations, where the conversation structure is formed via messages sent among users. In this work, we examine conversation context, and model the participants' interactions therein. This approach enables deep representation learning that reflects personal interests and conversation preferences, together signaling what conversations a user is likely to be involved in.

To illustrate how online interactions could indicate users' future conversation behavior, Figure 1

*This work was mainly conducted when Jing Li was affiliated with Tencent AI Lab, Shenzhen, China.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4633–4643, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

shows two conversation snippets on Reddit, both centering around the September 11 attack (9/11). As can be seen, user $U_0$, who had discussed the event according to the chat history, later engaged in Conversation 1 ($C_1$) instead of Conversation 2 ($C_2$). One explanation is that $C_1$ was initialized by user $U_1$, whose discussion topics overlap with $U_0$'s, and more importantly, used to interact with $U_0$ in many prior discussions.

To model user preferences from their prior interactions, we propose to employ graph-structured neural networks to explicitly encode *who replies to whom at when* in the conversation history. In this way, temporal features of conversations are also exploited to capture messages' chronological orders (shown in Figure 1 with blue arrows). We then incorporate the interaction representations into a novel neural collaborative filtering framework (He et al., 2017), which further aligns user's preferences with the conversation context. Compared with existing methods that are based on handcrafted features (Chen et al., 2011) or Bayesian models (Zeng et al., 2018), our end-to-end trained neural model learns to automatically recommend conversations as well as to encode user interests embedded in their conversation interactions. To the best of our knowledge, this is the first work to explore neural conversation recommendation with online interactions explicitly encoded for user preference modeling.

To evaluate our model, we conduct extensive experiments on two large-scale datasets with online conversations from Twitter and Reddit[1]. Experimental results show that our method significantly outperforms state-of-the-art models that do not capture user interactions. For example, our model obtains an MAP (Mean Average Precision) of 0.625 on Twitter, compared with 0.591 by Zeng et al. (2018). We further find that our model still exhibits superior performance when the sparsity levels of user history and conversation context are varied, demonstrating our model's potential ability to handle sparse conversation records. Additional experiments on an ablation study confirms the effectiveness of different components in our framework. A case study further reveals important interaction features captured by our model, which indicate their conversation entries and hence explain our model's advanced performance. Finally, we investigate the challenging task of first time replies prediction, where our model again produces significantly better results than existing popular recommendation models.

[1]The datasets and codes are available at: https://github.com/zxshamson/neural-conv-rec

## 2 Related Work

Our work is in line with conversation behavior analysis, where studies explore user interactions in ongoing conversations (Ritter et al., 2010) and how they signal the conversations' future trajectory, such as continued activity (Backstrom et al., 2013; Jiao et al., 2018; Zeng et al., 2019) and the risk of going awry (Zhang et al., 2018). Different from these proposals which do not model personal interests, we study conversation recommendation for a specific user, where we measure how a user's preferences match a conversation's context.

This work is also related to user response prediction (Artzi et al., 2012; Zhang et al., 2015) and post recommendation (Duan et al., 2010; Chen et al., 2012; Yan et al., 2012; Hong et al., 2013). While most of these studies focus on post modeling, we examine conversation context to predict user engagements, which goes beyond the post-level prediction task. Other prior work examining conversation-level recommendation relies on either manual features (Chen et al., 2011) or shallow word occurrence patterns (Zeng et al., 2018), largely ignoring the useful features from historical user interactions. On the contrary, we utilize online user interactions in the conversation history, to allow the inclusion of richer information of modeling personal interests. In addition, our neural network-based model enables automatic learning for a deeper representation of user interests, whereas existing methods require significant manual efforts for model customization (Chen et al., 2011; Zeng et al., 2018).

Furthermore, our user interaction module is inspired by prior work on conversation structure modeling. Compared with popular sequential conversation models that focus on messages' temporal features (Cheng et al., 2017; Jiao et al., 2018; Zeng et al., 2019), our module explicitly encodes the replying relationships to exploit the user conversation structure (Miura et al., 2018; Zayats and Ostendorf, 2018). It is shown that such structure indicates salient messages and can benefit various compelling applications, e.g., conversation summarization (Chang et al., 2013; Li et al., 2015) and discussion topic extraction (Li et al., 2016, 2018).

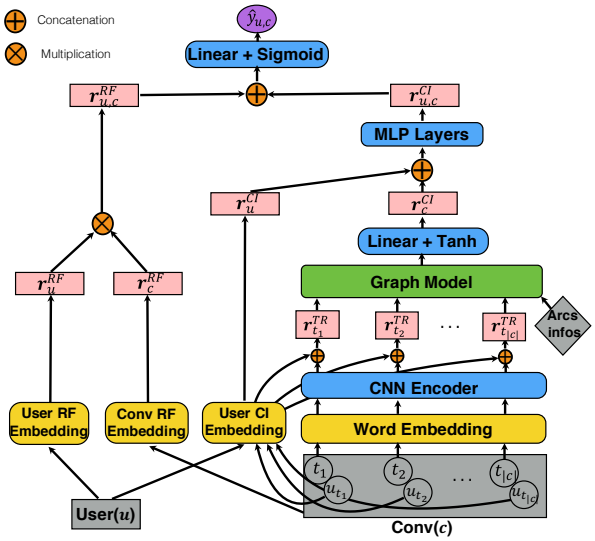


Figure 2: The architecture of our neural conversation recommendation framework, which models replying preferences and conversation interactions to predict a user's future engagement in given conversations. RF: replying factors modeling. CI: conversation interaction modeling.

However, its effect on conversation recommendation has not been explored yet, and our work aims to fill the gap.

## 3 Neural Conversation Recommendation

This section describes our neural recommendation model with interaction modeling. Figure 2 shows the overall architecture of our framework based on neural collaborative filtering (NCF) (He et al., 2017). Section 3.1 will present an overview showing how our model works, where both users' replying history and conversations' interaction structure will be encoded for recommendation. Their modeling details will be given in Section 3.2 and 3.3 in turn. At last, Section 3.4 shows the overall model training process.

### 3.1 Model Overview

Here we first describe the input and output. For training, our model is fed with a conversation dataset $\mathcal{C}$. Each conversation $c \in \mathcal{C}$ is formed with a sequence of turns $\langle t_1, t_2, ..., t_{|c|} \rangle$, where $|c|$ denotes the number of turns. Each turn $t$ is in form of a word sequence $\langle w_1, w_2, ..., w_{|t|} \rangle$ with $|t|$ being $t$'s word number. Its author is represented by user id $u_t$. We also record each turn's parent turn in replying relations (i.e. which turn it replies to) and chronological order (i.e. which turn posted before it), so that the interaction patterns within a conversation can be captured. We will talk more about it in Section 3.3.

For recommendation, our model is taken a user $u$ and a conversation $c$ as input, and then predict how likely $u$ will engage in $c$, conditioned on $u$'s previous behavior and $c$'s context history.

Our goal is to predict $\hat{y}_{u,c} \in [0, 1]$, which measures how likely user $u$ will engage in conversation $c$. Here to estimate $\hat{y}_{u,c}$, two types of information are encoded: replying history of users and interaction structure of conversations. The former is captured from what conversations a user previously replied to, where we learn $\boldsymbol{r}^{RF}_{u,c}$ to encode $u$'s replying preference on $c$. However, such learned representation captures user replying preferences without diving into turn-level features and interaction structure in conversations. So we utilize the latter to explore how users interact with each other in conversation context, which encodes words in turns and turn interactions to produce conversation interaction representation $\boldsymbol{r}^{CI}_{u,c}$, reflecting a denser preference of a user. In Section 3.2, we will present how to learn $\boldsymbol{r}^{RF}_{u,c}$, and in Section 3.3 we learn about $\boldsymbol{r}^{CI}_{u,c}$.

Coupling the $\boldsymbol{r}^{RF}_{u,c}$ and $\boldsymbol{r}^{CI}_{u,c}$, we predict $\hat{y}_{u,c}$ via the formula below:

$$\hat{y}_{u,c} = \sigma(\boldsymbol{h}_O^T[\boldsymbol{r}^{RF}_{u,c}; \boldsymbol{r}^{CI}_{u,c}]) \quad (1)$$

where $\sigma(\cdot)$ denotes sigmoid activation, $[;]$ indicates concatenation operation, and $\boldsymbol{h}_O$ is a learnable parameter. In recommendation for user $u$, we rank the conversations with $\hat{y}_{u,c}$ and the top $N$ results will serve as our final output.

### 3.2 Replying Factors Modeling

As mentioned in Section 3.1, we first model users' replying preferences with what conversations they entered before. We follow the practice in He et al. (2017) to use two embedding layers, $I^{RF}_U(\cdot)$ and $I^{RF}_C(\cdot)$, to capture the latent factors for users and conversations that result in user's previous replying history. For user $u$, we can obtain its user embedding $\boldsymbol{r}^{RF}_u$ by looking up $u$ in $I^{RF}_U(\cdot)$. A conversation embedding $\boldsymbol{r}^{RF}_c$ can be similarly obtained from $I^{RF}_C(\cdot)$. Then we measure user $u$'s replying preference over conversation $c$ with the similarity between $\boldsymbol{r}^{RF}_u$ and $\boldsymbol{r}^{RF}_c$:

$$\boldsymbol{r}^{RF}_{u,c} = \boldsymbol{r}^{RF}_u \odot \boldsymbol{r}^{RF}_c \quad (2)$$

where $\odot$ denotes element-wise product. As can be seen, $\boldsymbol{r}^{RF}_{u,c}$ is able to encode what conversations

a user engages in and analyze the factors of why it happen, simply with general replying history. More features will be explored via conversation interaction modeling presented in Section 3.3.

### 3.3 Conversation Interaction Modeling

We first explore users' prior interaction behavior in conversations. A user embedding layer $I_U^{CI}(\cdot)$ is hence employed, where the embedding $\boldsymbol{r}_u^{CI}$ for user $u$ reflects $u$'s interaction patterns, such as what they used to say and whom they usually interacted with. For the conversation modeling, we adopt graph-structured networks to model the interaction structure therein and yield a representation $\boldsymbol{r}_c^{CI}$ for conversation $c$. The effects of user- and conversation-specific interaction features are combined with Multilayer Perceptron (MLP):

$$\boldsymbol{r}_{u,c}^{CI} = \alpha(W_M^T(...\alpha(W_1^T[\boldsymbol{r}_u^{CI}; \boldsymbol{r}_c^{CI}] + b_1)...) + b_M) \quad (3)$$

where $\alpha(\cdot)$ is ReLU-activated function (Rectified Linear Unit) and $M$ is the number of layers in MLP. In the following, we will introduce how we obtain $\boldsymbol{r}_c^{CI}$ via modeling of intra-turn features and inter-turn interactions.

**Turn-level Modeling.** Here we describe how we model turn-level representations, which combine what content it conveys and who its author is.

Content representation is to reflect how words appear therein, where we employ a Convolutional Neural Network (CNN) (Kim, 2014) encoder to model a turn's word sequence. Specifically, given a turn $t$ in conversation $c$, we first map each word in $t$ into a word embedding layer (initialized with pre-trained word vectors) to explore deep word semantics. And then, to capture how a word appears in local context with its neighbors, a CNN encoder is exploited to generate the turn-level content representation $\boldsymbol{z}_t$.

Next, we concatenate $\boldsymbol{z}_t$, conveying content features, and $\boldsymbol{r}_{u_t}^{CI}$, embedded with the interaction patterns of $t$'s author $u_t$, to produce a turn representation $\boldsymbol{r}_t^{TR}$. It couples turn $t$'s word occurrence patterns and its author's history interactions with other conversation turns. Afterwards, $\boldsymbol{r}_t^{TR}$ is delivered to model $t$'s interaction with the other turns in $c$. We will describe how it is processed next.

**Turn Interaction Modeling.** To encode conversation interaction structure, we first organize the turns in a conversation $c$ as a reply tree to formulate who replies to whom. Each node therein represents a turn and the edges reflect replying relations (directed from turns to replies such as the red arrows in Figure 1). Moreover, to exploit temporal information, we add another kind of edges to indicate chronological order (such as the blue arrows in Figure 1). In doing so, a reply tree is extended to a directed graph (such as the one in Figure 1), with both replying and temporal interactions encoded and therefore named as an interaction graph. For each turn $t$ on the graph, we distinguish its neighbors into predecessors, denoted by $E^p(t)$, and successors, $E^s(t)$.

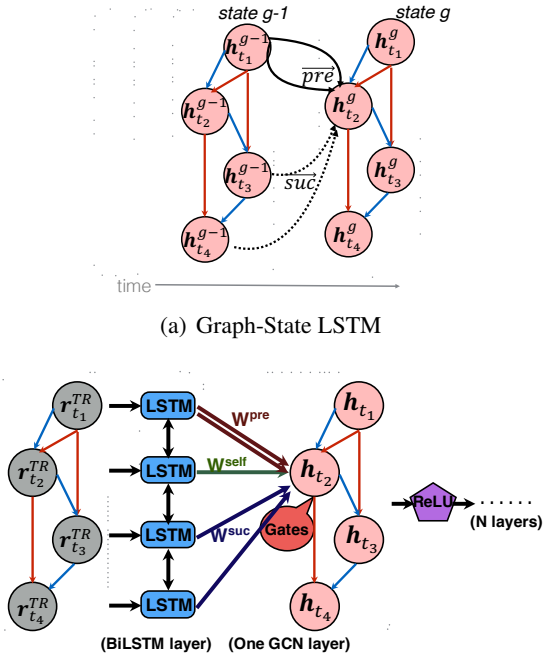


(a) Graph-State LSTM

(b) Graph Convolutional Networks

Figure 3: Two variants of our proposed graph-structured networks.

Then, we employ graph-structured networks to model the interaction structure. There are two modeling methods discussed here: **Graph-State LSTM** (Long Short-Term Memory) (henceforth GLSTM) (Beck et al., 2018; Song et al., 2018) and **Graph Convolutional Networks** (henceforth GCN) (Kipf and Welling, 2017; Marcheggiani and Titov, 2017), whose empirical effectiveness will be compared in Section 5.1. Here we present their architecture in Figure 3 and describe how they model conversation interactions below.

*Graph-State LSTM.* We start with GLSTM and show its architecture in Figure 3(a). It is an extension of LSTM from sequence to graph structure, where a turn's hidden states are updated conditioned on both the turn-level representation $\boldsymbol{r}_t^{TR}$

and the states of all its neighbors on the graph. The update strategy is the same as standard LSTM (Hochreiter and Schmidhuber, 1997), except for the following formula, which can be used in the update of input gate, output gate, forget gate, and content recorder:

$$\boldsymbol{g}_t^{GLSTM} = \sigma(W^p \boldsymbol{x}_t^p + W^s \boldsymbol{x}_t^s + U^p \boldsymbol{h}_t^p + U^s \boldsymbol{h}_t^s + b) \quad (4)$$

The first two terms explore the turn-level representations ($\boldsymbol{r}_t^{TR}$) from the neighbors. The third and forth terms capture turn interactions on the graph. $b$ denotes the bias. The superscripts $p$ and $s$ indicate the neighbor being a predecessor or successor. $\boldsymbol{x}_t^p$ takes the sum of predecessor $k$'s turn representations $\boldsymbol{r}_k^{TR}$ and so does $\boldsymbol{x}_t^s$ for successors. $\boldsymbol{h}_t^*$ means the neighbors' hidden states in their last updates. $W^*$ and $U^*$ are learnable parameter weights and $\sigma(\cdot)$ means sigmoid activation. Moreover, in GLSTM, we define the state number $g$ to reflect the maximum order of GLSTM state transitions, where the larger $g$ indicates longer turn dependency on graph paths encoded. Here due to the space limitation, we leave out the details of GLSTM and refer the readers to Song et al. (2018).

Afterwards, to produce conversation representation $\boldsymbol{r}_c^{CI}$ with turn interactions, we combine all turns' hidden states with average pooling and map them into the same dimension (with Tanh activation) as $\boldsymbol{r}_u^{CI}$ to measure user and conversation similarity.

*Graph Convolutional Networks.* Figure 3(b) shows the architecture of GCN, which can be considered as CNN on graph. Here following previous practice (Marcheggiani and Titov, 2017), before using GCN to model turn interactions, we first feed the turn representations $\boldsymbol{r}_t^{TR}$ into a sequential Bidirectional LSTM (BiLSTM) layer to capture the chronological turn interactions. Then, we take the $t$-th hidden states of BiLSTM $\boldsymbol{h}_t^{LSTM}$ to further capture turn $t$'s interaction with its neighbors on interaction graph. The formula describing this process is given as:

$$\begin{aligned}\boldsymbol{h}_t^{GCN} = &\sum_{i \in E^p(t)} \omega_{i,t}(W^{pre}\boldsymbol{h}_i^{LSTM} + b^{pre}) + \\ &\sum_{j \in E^s(t)} \omega_{j,t}(W^{suc}\boldsymbol{h}_j^{LSTM} + b^{suc}) + \\ &\omega_{t,t}(W^{self}\boldsymbol{h}_t^{LSTM} + b^{self})\end{aligned} \quad (5)$$

Here following Marcheggiani and Titov (2017), we use different sets of parameters to fit varying types of interactions: self interactions (self), interaction from predecessors to successors (pre), and that from the other way around (suc). $\omega_{i,j}$ is a scalar gate controlling weights defined below:

$$\omega_{i,j} = \sigma(\boldsymbol{h}_i^{LSTM} \cdot V^{dir(i,j)} + d^{dir(i,j)}) \quad (6)$$

It is to identify the neighbors affecting more to $t$ than others. $dir(i,j)$ indicates the type of $i$-$j$ direction (pre, suc, or self).

Furthermore, to allow deep interactions to be learned, we can stack multiple GCN layers to form a multi-layer GCNs, where we apply a ReLU activated function between two layers. After that, we take the similar operations as for GLSTM yield conversation representation $\boldsymbol{r}_c^{CI}$ as $c$'s conversation interaction representation.

### 3.4 Model Training

Here we describe how we formulate our learning objective and train our model. As mentioned above, our goal is to predict a score $\hat{y}_{u,c} \in [0,1]$ indicating how likely user $u$ will reply to conversation $c$. In training, we adopt binary cross-entropy as our learning loss with penalty given to negative feedback ($u$ does not engage in $c$ ). It is because negative feedback may happen for many unpredictable reasons, such as users being too busy to go online. Thus for conversation recommendation, we rely more on the positive feedback and design the *weighted binary cross-entropy loss* below:

$$\mathcal{L} = -\sum_{(u,c)\in\mathcal{T}} \left[\lambda \cdot y_{u,c}\log(\hat{y}_{u,c}) + (1-y_{u,c})\log(1-\hat{y}_{u,c})\right] \quad (7)$$

where $\mathcal{T}$ is a set of training instances. $y_{u,c}$ is a binary label indicating whether $u$ replied to $c$, and $\hat{y}_{u,c}$ is our predicted score. $\lambda$ ($\lambda > 1$) is a predefined parameter to trade off the weights of positive and negative instances.

In model training, the negative sampling strategy is adopted (He et al., 2017), whose sampling ratio (the number of negative samples for each positive instance) is set to 5. Also, we pre-train the embedding layers for both the replying factors and conversation interaction modeling with the parameters from He et al. (2017). We will discuss the effects of pre-training in Section 5.3.

## 4 Experimental Setup

**Data Collection and Preprocessing.** In our experiments, we use datasets from two different platforms: the first one is released by Zeng et al. (2018) containing **Twitter** conversations formed by tweets from the TREC 2011 microblog track

| | Twitter | Reddit |
|---|---|---|
| # of users | 10,122 | 13,134 |
| # of conversations | 7,500 | 29,477 |
| # of turns | 38,999 | 109,774 |
| Avg. # of conversations per user | 1.7 | 5.9 |
| Avg. # of turns per conversation | 5.2 | 3.7 |
| Avg. # of users per conversation | 2.3 | 2.6 |

Table 1: Dataset statistics.

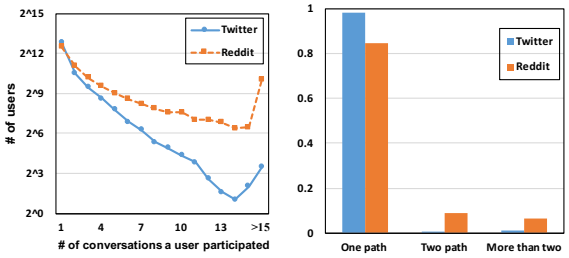


(a) User conversation dist. (b) Ratio of diff. structure

Figure 4: Distribution of number of conversations per user (left) and proportion of different conversation structures (right) for both datasets.

data[2] covering a diverse set of topics; the other is from Zeng et al. (2019), which is comprised of discussion threads about political issues on **Reddit**, a popular discussion website.

The tweets in Twitter dataset were mainly posted from Jan 23 to Feb 8, 2011, and discussion threads in Reddit dataset were posted from Jan to Dec, 2008. To discover the whole conversations, we retrieved all messages with replying relations (indicated by "*parent_id*" property in Reddit corpus, for example), and recorded their authors and parent messages. Finally, conversations with only one message were removed.

We applied the Glove tweet preprocessing toolkit (Pennington et al., 2014)[3] on the Twitter dataset. As for the Reddit dataset, we performed tokenization using open source natural language toolkit (NLTK) (Loper and Bird, 2002), with links replaced to a generic tag "URL" and all number tokens removed. We maintained a vocabulary with all the rest characters appearing in the corpus for both datasets, including punctuation and emoticons.

**Data Statistics and Analysis.** The statistics of two datasets are shown in Table 1, with more information in Figure 4. We can observe that Reddit dataset contains more conversations, with a higher average number of conversations per user. On the other hand, Twitter conversations are longer, with fewer participants. Figure 4(a) shows that most users participate in very few conversations in both datasets, indicating a potential sparsity problem. In terms of conversation structure (Figure 4(b)), most conversations only contain one path where the replying relations precisely follow the chronological order; whereas the Reddit dataset contains more tree-structured conversations with rich and complex interactions.

To further illustrate the effect of a conversation's structure on its future development, we calculate the likelihoods of (1) new users joining the discussion, and (2) current participants continuing the conversation, grouped by different conversation structures (Table 2). In general (especially on Reddit), conversations with only one path tend to continue within the current participants, and keep newcomers out, whereas conversations with complex structures are more likely to attract new users.

| | Twitter | | | Reddit | | |
|---|---|---|---|---|---|---|
| Paths amount | 1 | 2 | $> 2$ | 1 | 2 | $> 2$ |
| New comers rate | 0.18 | 0.20 | 0.16 | 0.30 | 0.45 | 0.50 |
| Continue rate | 0.70 | 0.64 | 0.74 | 0.51 | 0.31 | 0.38 |

Table 2: Different structures and the future trend of conversations.

**Model Setting.** We follow the experimental settings employed in previous work (Zeng et al., 2018). For each conversation, we take first 75% of the context as observation for training purpose. The rest is equally divided into a testing set and a development set. For negative instances, we also split the unobserved user-conversation pairs into three parts: 80%, 10%, and 10% for training, testing, and development, respectively. Furthermore, due to the large amount of conversations in Reddit dataset, we only sample 100 negative instances uniformly from them for testing and development.

For parameters setups, we initialize the word embedding layer with 200-dimensional Glove embedding (Pennington et al., 2014), where the Twitter version is used for our Twitter dataset, and the Common Crawl version is applied on the Reddit dataset[4]. Factor dimension for the RF part is set to 20, while for the CI part it is 100. For the CNN

[2] https://trec.nist.gov/data/tweets/

[3] https://nlp.stanford.edu/projects/glove/preprocess-twitter.rb

[4] https://nlp.stanford.edu/projects/glove/

encoders, we use filter windows of 2, 3, and 4, each with 100 feature maps. For the size of hidden states of our graph models, we set 200 (100 for each direction for BiLSTM). The number of MLP layers is 3. During training, the batch size is set to 512 and Adam optimizer (Kingma and Ba, 2014) is adopted with an initial learning rate of 0.01. We set the trade off weight in learning loss $\lambda = 100$.

**Evaluation and Comparisons.** Following Zeng et al. (2018)'s work, we adopt mean average precision (MAP), precision at 1 (P@1), and normalized Discounted Cumulative Gain at 5 (nDCG@5) for evaluation (we also try other metrics including P@5 and nDCG@10, and find similar trends). The metrics are first computed for users in the datasets, then averaged over all users.

For comparison, we first search for the best model among different interaction modeling (Section 5.1). We then consider two baselines: 1) ranking conversations randomly (RANDOM), 2) conversations with more participants ranked higher (POPULARITY). Previous work compared includes[5]:

• RSVM: Ranks conversations for each user with features described in Duan et al. (2010) by ranking SVM (Joachims, 2002).

• NCF: The neural CF model (He et al., 2017), not utilizing any context information.

• CONVMF: A CNN-based model for recommendation with reviews (Kim et al., 2016), where we adapt to use a hierarchical two-layer CNN to model words in turns and turn sequences.

• CR_JTD: The state-of-the-art method for our task (Zeng et al., 2018), with a Bayesian model jointly modeling topics and discourse.

## 5 Experimental Results

In this section, we first evaluate the effectiveness of varying modules for conversation interaction modeling in Section 5.1. Then our model with the best module is further compared with the baselines and previous recommendation systems in Section 5.2. There we also discuss the model performance given varying conversation context length and user interaction sparsity. We further discuss our model with an ablation study and a case study in Section 5.3. Finally in Section 5.4, we analyze the results of first time replies prediction.

[5]We did not compare with Chen et al. (2011) since that method mainly requires information about social relationship, which is unavailable in our datasets.

| Models | Train Time | MAP | |
|---|---|---|---|
| | | Twitter | Reddit |
| BiLSTM | 0.94 | 0.617 | 0.498 |
| GLSTM | 1.25 | 0.617 | 0.528 |
| GCN (W/O BiLSTM) | 1.03 | 0.619 | 0.530 |
| GCN (With BiLSTM) | 1.00 | **0.620** | **0.533** |

Table 3: Results of our model variants on development set. The best MAP results are in **bold**. "Train Time": training time per epoch divided by that of model GCN (With BiLSTM).

### 5.1 Interaction Modeling Comparison

We first compare the effects of varying interaction modeling methods (see Section 3.3) on conversation recommendation. Table 3 displays their results on development set. In comparison, we consider BiLSTM over turn sequence (only chronological order encoded and henceforth BiLSTM), GLSTM (state number $g = 6$), GCN (layer number set to 3) without BiLSTM-encoded temporal representations (henceforth GCN (W/O BiLSTM)), and the full GCN described in Section 3.3 (henceforth GCN (With BiLSTM) and layer number set to 1). The above hyper-parameters are tuned based on the training loss.

From the results, we find that BiLSTM exhibits the worst results for not encoding replying relations. Its difference from others are larger on Reddit attributed to the rich replying structure therein (as shown in Figure 4(b)). The best performance is achieved for GCN (With BiLSTM), with relatively less training time. This shows the effectiveness and efficiency to explore the order of turns with BiLSTM and the user interactions with GCN. In the later analysis, we will only discuss our model that exploits GCN (With BiLSTM) for interaction modeling.

### 5.2 Comparisons with Previous Work

**Main Results.** Table 4 shows the conversation recommendation results with baselines and state of the arts. Our model exhibits the best results on both datasets, significantly outperforming all the comparison models. It indicates the usefulness to encode user interactions for conversation recommendation. Particularly, CONVMF is able to encode turns' temporal orders yet ignores how they reply with each other in conversation history. It is outperformed by our model, showing the benefit to capture users' replying patterns for predicting what conversations will draw their engagement.

| Models | Twitter | | | Reddit | | |
|---|---|---|---|---|---|---|
| | MAP | P@1 | nDCG | MAP | P@1 | nDCG |
| **Baselines** | | | | | | |
| RANDOM | 0.006 | 0.001 | 0.002 | 0.040 | 0.010 | 0.022 |
| POPULARITY | 0.023 | 0.005 | 0.010 | 0.082 | 0.033 | 0.063 |
| **Comparisons** | | | | | | |
| RSVM | 0.554 | 0.575 | 0.559 | 0.453 | 0.457 | 0.466 |
| NCF | 0.573 | 0.593 | 0.576 | 0.412 | 0.544 | 0.461 |
| CONVMF | 0.579 | 0.596 | 0.583 | 0.485 | 0.532 | 0.520 |
| CR_JTD | 0.591 | 0.591 | 0.600 | 0.453 | 0.559 | 0.485 |
| OURS | **0.625** | **0.632** | **0.626** | **0.538** | **0.674** | **0.590** |

Table 4: Main results on conversation recommendation. "nDCG" stands for "nDCG@5". The best result for each column is in **bold**. Our model significantly outperforms all the comparisons ($p < 0.01$, paired t-test).

We also observe that both baseline models work poorly. It is because conversation recommendation is challenging, not possible to be well tackled with simple ranking strategies.

In addition, we notice that CR_JTD outperforms CONVMF on Twitter, with opposite observation made on Reddit. It is possibly because Twitter exhibits more informal language style. Thus CR_JTD, taken bag-of-words input, can better fit the data than CONVMF, taking word orders into account. Nevertheless, our model outperforms them both, showing that prior interactions among users can better signal their future reply behavior compared with the words they said.

The final observation is that, all comparing methods (except for the naive baselines) perform better on Twitter than Reddit. One reason is that the Twitter dataset is smaller and contains fewer users and conversations. Another possible reason might be that the topics in the Reddit dataset are mostly about politics, while the Twitter conversations are of diverse topics, which makes the model easier to distinguish user interests.

**Training with Varying Conversation History.** The main results are reported given the first 75% turns as conversation history. Here we investigate how the length of conversation history affects reply preference prediction. The models are hence trained with the first 25%, 50%, and 75% turns as conversation history and their MAP scores are shown in Figure 5.

As can be seen, all models exhibit better results when trained with longer history. This shows that users' future conversation preference can be better predicted with richer history data. We also observe

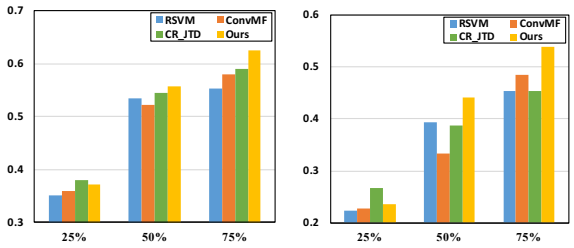


(a) Twitter (b) Reddit

Figure 5: MAP scores (in Y-axis) of models trained with the first 25%, 50%, and 75% turns as history.

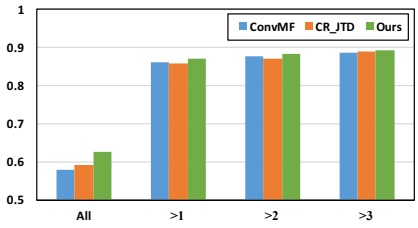


Figure 6: MAP scores (in Y-axis) for users with varying degrees of interaction history. X-axis indicates the number of conversations users previously engage in. *"All"* means recommendation for all users.

that our model obtains the best MAP on both the 50% and 75% setting, while for 25%, it is outperformed by CR_JTD. It might be ascribed to the sparse user interactions exhibited in 25% context, where there are only 1 or 2 turns on average according to Table 1.

**Results for Varying User Interaction Sparsity.** Figure 4(a) has shown that most users only engage in very few conversations. This results in severe data sparsity in user interaction history, especially on Twitter. We are hence interested in how models perform on varying degree of sparsity.

Figure 6 shows the MAP scores in recommendation to users engaged in varying number of conversations before on Twitter. As can be seen, user interaction sparsity can largely affect recommendation performance, where all models perform poorly for users exhibiting less than one conversation entry. We also observe that our model performs consistently better in varying degrees of sparsity. It may because our model is able to learn rich interactions from conversation context, which helps alleviate the sparsity in user history.

### 5.3 Further Discussion

Here we further discuss what our model learns leading to its superiority.

**Ablation Study.** We start with an ablation study to discuss the relative contributions of our different components. The MAP scores of their ablations are compared in Table 5. Our full model performs the best, showing that all components are useful. It is also seen that RF modeling contribute the most, meanwhile better contributions can be made with its parameters pre-trained. It indicates the crucial role RF modeling plays for neural conversation recommendation.

| Models | Twitter | Reddit |
|---|---|---|
| W/O RF modeling (Sec. 3.2) | 0.509 | 0.239 |
| W/O pre-training (Sec. 3.4) | 0.593 | 0.525 |
| W/O MLP layers (Eq. 3) | 0.600 | 0.537 |
| W/O gate control weights (Eq. 6) | 0.608 | 0.530 |
| Our full model | **0.625** | **0.538** |

Table 5: MAP scores obtained by our ablations. The best MAP results are highlighted in **bold**.

**Case Study.** To further understand how our model predicts users' conversation preferences, we take the example in Figure 1 and analyze what our model learns for it. Recall that user $U_0$ used to discuss a lot on 9/11 with $U_1$, who starts $C_1$. $U_0$ later engages in $C_1$ instead of $C_2$, though it also concerns 9/11.

| | $U_1$ | $U_2$ | $U_4$ | $U_5$ |
|---|---|---|---|---|
| $\boldsymbol{r}_u^{RF}$ | **0.761** | 0.731 | 0.681 | 0.659 |
| $\boldsymbol{r}_u^{CI}$ | **0.763** | 0.453 | 0.287 | 0.737 |

Table 6: Cosine similarity between $U_0$'s user embeddings in RF and CI modeling with others'.

Table 6 shows the similarity of user factors learned by our RF (replying factor modeling in Section 3.2) and CI (conversation interaction modeling in Section 3.3) modules. As can be seen, both modules learn that $U_0$ and $U_1$ are similar (probably referred from their frequent interactions). Their joint effects result in our successful prediction of $U_0$ to engage in $C_1$ rather than $C_2$. For the same reason, our model can further predict that $U_0$ is more likely to reply to $T_4$ (in $C_1$), which is posted by $U_1$, based on the similarity between user factors and turn representations.

### 5.4 First Time Replies Prediction

In some scenarios, users may be more interested in seeing new conversations, which they haven't seen before but potentially match their preferences. We hence examine model performance to predict only first time replies and show their MAP scores in Table 7. It is observed that all models perform poorly, which implies that recommending unseen conversations to users is extremely difficult. This finding is consistent with Zeng et al. (2018). However, our model still outperforms others by a large margin. It again demonstrates the effectiveness of modeling user interactions for recommendation.

| Models | Twitter | Reddit |
|---|---|---|
| RSVM | 0.002 | 0.049 |
| NCF | 0.033 | 0.038 |
| ConvMF | 0.049 | 0.210 |
| CR_JTD | 0.090 | 0.075 |
| Ours | **0.160** | **0.212** |

Table 7: MAP scores for first time replies prediction.

## 6 Conclusion

We study neural conversation recommendation with graph-structured networks to encode user interactions. Experimental results on Twitter and Reddit show our model significantly outperforms the state of the arts. We also observe that competitive results can still be obtained on varying conversation history length and user interaction sparsity. Further discussions analyze the contributions of different components of our model and the useful features we learn leading to our superiority. At last, we study a challenging task of first time replies prediction, where our model still exhibits its effectiveness.

## Acknowledgements

This work is partially supported by the following HK grants: RGC-GRF (14232816, 14209416, 14204118, 3133237), NSFC (61877020) & ITF (ITS/335/18). Lu Wang is supported in part by National Science Foundation through Grants IIS-1566382 and IIS-1813341. We thank the three anonymous reviewers for the insightful suggestions on various aspects of this work.

## References

Yoav Artzi, Patrick Pantel, and Michael Gamon. 2012. Predicting responses to microblog posts. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 602–606. Association for Computational Linguistics.

Lars Backstrom, Jon M. Kleinberg, Lillian Lee, and Cristian Danescu-Niculescu-Mizil. 2013. Characterizing and curating conversation threads: expansion, focus, volume, re-entry. In *Sixth ACM International Conference on Web Search and Data Mining, WSDM 2013, Rome, Italy, February 4-8, 2013*, pages 13–22.

Daniel Beck, Gholamreza Haffari, and Trevor Cohn. 2018. Graph-to-sequence learning using gated graph neural networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 273–283.

Yi Chang, Xuanhui Wang, Qiaozhu Mei, and Yan Liu. 2013. Towards twitter context summarization with user influence models. In *Sixth ACM International Conference on Web Search and Data Mining, WSDM 2013, Rome, Italy, February 4-8, 2013*, pages 527–536.

Jilin Chen, Rowan Nairn, and Ed Huai-hsin Chi. 2011. Speak little and well: recommending conversations in online social streams. In *Proceedings of the International Conference on Human Factors in Computing Systems, CHI 2011, Vancouver, BC, Canada, May 7-12, 2011*, pages 217–226.

Kailong Chen, Tianqi Chen, Guoqing Zheng, Ou Jin, Enpeng Yao, and Yong Yu. 2012. Collaborative personalized tweet recommendation. In *Proceedings of the 35th international ACM SIGIR Conference on Research and development in information retrieval*, pages 661–670. ACM.

Justin Cheng, Michael Bernstein, Cristian Danescu-Niculescu-Mizil, and Jure Leskovec. 2017. Anyone can become a troll: Causes of trolling behavior in online discussions. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*, CSCW '17, pages 1217–1230. ACM.

Yajuan Duan, Long Jiang, Tao Qin, Ming Zhou, and Heung-Yeung Shum. 2010. An empirical study on learning to rank of tweets. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 295–303. Association for Computational Linguistics.

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*, pages 173–182.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Liangjie Hong, Aziz S Doumith, and Brian D Davison. 2013. Co-factorization machines: modeling user interests and predicting individual decisions in Twitter. In *Proceedings of the sixth ACM International Conference on Web Search and Data Mining*, pages 557–566. ACM.

Yunhao Jiao, Cheng Li, Fei Wu, and Qiaozhu Mei. 2018. Find the conversation killers: A predictive study of thread-ending posts. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 1145–1154. International World Wide Web Conferences Steering Committee.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 133–142.

Dong Hyun Kim, Chanyoung Park, Jinoh Oh, Sungyoung Lee, and Hwanjo Yu. 2016. Convolutional matrix factorization for document context-aware recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, September 15-19, 2016*, pages 233–240.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Jing Li, Wei Gao, Zhongyu Wei, Baolin Peng, and Kam-Fai Wong. 2015. Using content-level structures for summarizing microblog repost trees. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 2168–2178.

Jing Li, Ming Liao, Wei Gao, Yulan He, and Kam-Fai Wong. 2016. Topic extraction from microblog posts using conversation structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Jing Li, Yan Song, Zhongyu Wei, and Kam-Fai Wong. 2018. A joint model of conversational discourse and latent topics on microblogs. *Computational Linguistics*, 44(4).

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 1506–1515.

Yasuhide Miura, Ryuji Kano, Motoki Taniguchi, Tomoki Taniguchi, Shotaro Misawa, and Tomoko Ohkuma. 2018. Integrating tree structures and graph structures with neural networks to classify discussion discourse acts. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 3806–3818.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Alan Ritter, Colin Cherry, and Bill Dolan. 2010. Unsupervised modeling of Twitter conversations. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics*, pages 172–180.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. A graph-to-sequence model for amr-to-text generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 1616–1626.

Rui Yan, Mirella Lapata, and Xiaoming Li. 2012. Tweet recommendation with graph co-ranking. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 516–525. Association for Computational Linguistics.

Victoria Zayats and Mari Ostendorf. 2018. Conversation modeling on reddit using a graph-structured LSTM. *TACL*, 6:121–132.

Xingshan Zeng, Jing Li, Lu Wang, Nicholas Beauchamp, Sarah Shugars, and Kam-Fai Wong. 2018. Microblog conversation recommendation via joint modeling of topics and discourse. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, Volume 1 (Long Papers)*, pages 375–385.

Xingshan Zeng, Jing Li, Lu Wang, and Kam-Fai Wong. 2019. Joint effects of context and user history for predicting online conversation re-entries. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 2809–2818.

Justine Zhang, Jonathan P Chang, Cristian Danescu-Niculescu-Mizil, Lucas Dixon, Yiqing Hua, Nithum Thain, and Dario Taraborelli. 2018. Conversations gone awry: Detecting early signs of conversational failure. *arXiv preprint arXiv:1805.05345*.

Qi Zhang, Yeyun Gong, Ya Guo, and Xuanjing Huang. 2015. Retweet behavior prediction using hierarchical Dirichlet process. In *AAAI*, pages 403–409.

# Different Absorption from the Same Sharing: Sifted Multi-task Learning for Fake News Detection

**Lianwei Wu, Yuan Rao, Haolin Jin, Ambreen Nazir, Ling Sun**
Lab of Social Intelligence and Complexity Data Processing
School of Software Engineering, Xi'an Jiaotong University, Xi'an, 710049, China
{stayhungry,jinhaolin,ambreen.nazir,sunling}@stu.xjtu.edu.cn
raoyuan@mail.xjtu.edu.cn

## Abstract

Recently, neural networks based on multi-task learning have achieved promising performance on fake news detection, which focus on learning shared features among tasks as complementary features to serve different tasks. However, in most of the existing approaches, the shared features are completely assigned to different tasks without selection, which may lead to some useless and even adverse features integrated into specific tasks. In this paper, we design a sifted multi-task learning method with a selected sharing layer for fake news detection. The selected sharing layer adopts gate mechanism and attention mechanism to filter and select shared feature flows between tasks. Experiments on two public and widely used competition datasets, i.e. RumourEval and PHEME, demonstrate that our proposed method achieves the state-of-the-art performance and boosts the F1-score by more than 0.87%, 1.31%, respectively.

## 1 Introduction

In recent years, the proliferation of fake news with various content, high-speed spreading, and extensive influence has become an increasingly alarming issue. A concrete instance[1] was cited by Time Magazine in 2013 when a false announcement of Barack Obama's injury in a White House explosion "wiped off 130 Billion US Dollars in stock value in a matter of seconds". Other examples, an analysis of the US Presidential Election in 2016 (Allcott and Gentzkow, 2017) revealed that fake news was widely shared during the three months prior to the election with 30 million total Facebook shares of 115 known pro-Trump fake stories and 7.6 million of 41 known pro-Clinton fake stories. Therefore, automatically detecting fake news has attracted significant research attention in both industries and academia.

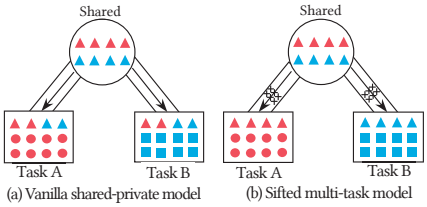


Figure 1: Two schemes for sharing features among tasks. Red circles and blue boxes represent the task-specific features, while the red and blue triangles mean shared features that benefit Task A and Task B, respectively.

Most existing methods devise deep neural networks to capture credibility features for fake news detection. Some methods provide in-depth analysis of text features, e.g., linguistic (Conroy et al., 2015), semantic (Yang et al., 2012), emotional (Wang et al., 2015), stylistic (Potthast et al., 2017), etc. On this basis, some work additionally extracts social context features (a.k.a. meta-data features) as credibility features, including source-based (Castillo et al., 2011), user-centered (Long et al., 2017), post-based (Wang, 2017) and network-based (Ruchansky et al., 2017), etc. These methods have attained a certain level of success. Additionally, recent researches (Thorne et al., 2017; Dungs et al., 2018) find that doubtful and opposing voices against fake news are always triggered along with its propagation. Fake news tends to provoke controversies compared to real news (Mendoza et al., 2010; Zubiaga et al., 2016b). Therefore, stance analysis of these controversies can serve as valuable credibility features for fake news detection.

There is an effective and novel way to improve the performance of fake news detection combined with stance analysis, which is to build multi-task learning models to jointly train both tasks (Ma

[1] http://business.time.com/2013/04/24/how-does-one-fake-tweet-cause-a-stock-market-crash/



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4644–4653,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

et al., 2018a; Kochkina et al., 2018; Li et al., 2018). These approaches model information sharing and representation reinforcement between the two tasks, which expands valuable features for their respective tasks. However, prominent drawback to these methods and even typical multi-task learning methods, like the shared-private model, is that the shared features in the shared layer are equally sent to their respective tasks without filtering, which causes that some useless and even adverse features are mixed in different tasks, as shown in Figure 1(a). By that the network would be confused by these features, interfering effective sharing, and even mislead the predictions.

To address the above problems, we design a sifted multi-task learning model with filtering mechanism (Figure 1(b)) to detect fake news by joining stance detection task. Specifically, we introduce a selected sharing layer into each task after the shared layer of the model for filtering shared features. The selected sharing layer composes of two cells: gated sharing cell for discarding useless features and attention sharing cell for focusing on features that are conducive to their respective tasks. Besides, to better capture long-range dependencies and improve the parallelism of the model, we apply transformer encoder module (Vaswani et al., 2017) to our model for encoding input representations of both tasks. Experimental results reveal that the proposed model outperforms the compared methods and gains new benchmarks.

In summary, the contributions of this paper are as follows:

- We explore a selected sharing layer relying on gate mechanism and attention mechanism, which can selectively capture valuable shared features between tasks of fake news detection and stance detection for respective tasks.
- The transformer encoder is introduced into our model for encoding inputs of both tasks, which enhances the performance of our method by taking advantages of its long-range dependencies and parallelism.
- Experiments on two public, widely used fake news datasets demonstrate that our method significantly outperforms previous state-of-the-art methods.

## 2 Related Work

**Fake News Detection** Exist studies for fake news detection can be roughly summarized into two categories. The first category is to extract or construct comprehensive and complex features with manual ways (Castillo et al., 2011; Ruchansky et al., 2017; Flintham et al., 2018). The second category is to automatically capture deep features based on neural networks. There are two ways in this category. One is to capture linguistic features from text content, such as semantic (Wang, 2017; Wu et al., 2018), writing styles (Potthast et al., 2017), and textual entailments (Oshikawa et al., 2018). The other is to focus on gaining effective features from the organic integration of text and user interactions (Qian et al., 2018; Wu et al., 2019). User interactions include users' behaviours, profiles, and networks between users. In this work, following the second way, we automatically learn representations of text and stance information from response and forwarding (users' behaviour) based on multi-task learning for fake news detection.

**Stance Detection** The researches (Lukasik et al., 2016; Zubiaga et al., 2016a) demonstrate that the stance detected from fake news can serve as an effective credibility indicator to improve the performance of fake news detection. The common way of stance detection in rumors is to catch deep semantics from text content based on neural networks(Mohtarami et al., 2018). For instance, Kochkina et al.(Kochkina et al., 2017) project branch-nested LSTM model to encode text of each tweet considering the features and labels of the predicted tweets for stance detection, which reflects the best performance in RumourEval dataset. In this work, we utilize transformer encoder to acquire semantics from responses and forwarding of fake news for stance detection.

**Multi-task Learning** A collection of improved models (Chen and Cardie, 2018; Chen et al., 2018; Liu et al., 2019) are developed based on multi-task learning. Especially, shared-private model, as a popular multi-task learning model, divides the features of different tasks into private and shared spaces, where shared features, i.e., task-irrelevant features in shared space, as supplementary features are used for different tasks. Nevertheless, the shared space usually mixes some task-relevant features, which makes the learning of different tasks introduce noise. To address this issue, Liu et al. (Liu et al., 2017) explore an adversarial shared-

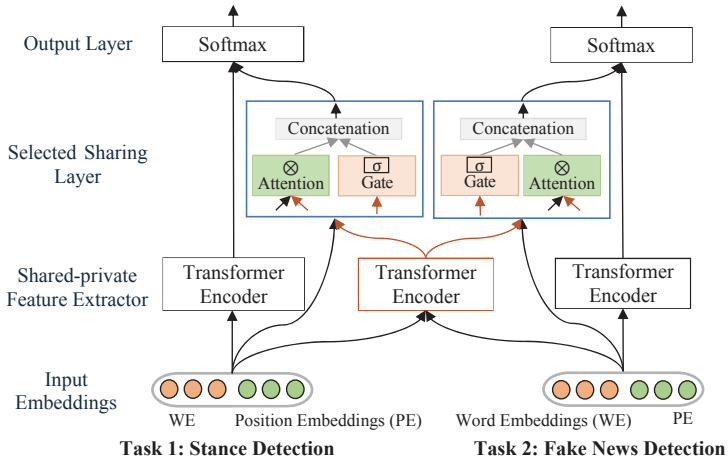


Figure 2: The architecture of the sifted multi-task learning method based on shared-private model. In particular, the two blue boxes represent selected sharing layers of stance detection and fake news detection and the red box denotes shared layer between tasks.

private model to alleviate the shared and private latent feature spaces from interfering with each other. However, these models transmit all shared features in the shared layer to related tasks without distillation, which disturb specific tasks due to some useless and even harmful shared features. How to solve this drawback is the main challenge of this work.

## 3 Method

We propose a novel sifted multi-task learning method on the ground of shared-private model to jointly train the tasks of stance detection and fake news detection, filter original outputs of shared layer by a selected sharing layer. Our model consists of a 4-level hierarchical structure, as shown in Figure 2. Next, we will describe each level of our proposed model in detail.

### 3.1 Input Embeddings

In our notation, a sentence of length $l$ tokens is indicated as $\mathbf{X} = \{x_1, x_2, ..., x_l\}$. Each token is concatenated by word embeddings and position embeddings. Word embeddings $w_i$ of token $x_i$ are a $d_w$-dimensional vector obtained by pre-trained Word2Vec model (Mikolov et al., 2013), i.e., $w_i \in \mathbb{R}^{d_w}$. Position embeddings refer to vectorization representations of position information of words in a sentence. We employ one-hot encoding to represent position embeddings $p_i$ of token $x_i$, where $p_i \in \mathbb{R}^{d_p}$, $d_p$ is the positional embedding dimension. Therefore, the embeddings of a sentence are represented as $\mathbf{E} = \{[w_1; p_1], [w_2; p_2], ..., [w_l; p_l]\}, \mathbf{E} \in \mathbb{R}^{l \times (d_p + d_w)}$. In particular, we adopt one-hot encoding to embed positions of tokens, rather than sinusoidal position encoding recommended in BERT model (Devlin et al., 2018). The reason is that our experiments show that compared with one-hot encoding, sinusoidal position encoding not only increases the complexity of models but also performs poorly on relatively small datasets.

### 3.2 Shared-private Feature Extractor

Shared-private feature extractor is mainly used for extracting shared features and private features among different tasks. In this paper, we apply the encoder module of transformer (Vaswani et al., 2017) (henceforth, transformer encoder) to the shared-private extractor of our model. Specially, we employ two transformer encoders to encode the input embeddings of the two tasks as their respective private features. A transformer encoder is used to encode simultaneously the input embeddings of the two tasks as shared features of both tasks. This process is illustrated by the shared-private layer of Figure 2. The red box in the middle denotes the extraction of shared features and the left and right boxes represent the extraction of private features of two tasks. Next, we take the extraction of the private feature of fake news detection as an example to elaborate on the process of transformer encoder.

The kernel of transformer encoder is the scaled dot-product attention, which is a special case of attention mechanism. It can be precisely described as follows:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{\mathbf{d}}})\mathbf{V} \quad (1)$$

where $\mathbf{Q} \in \mathbb{R}^{l\times(d_p+d_w)}$, $\mathbf{K} \in \mathbb{R}^{l\times(d_p+d_w)}$, and $\mathbf{V} \in \mathbb{R}^{l\times(d_p+d_w)}$ are query matrix, key matrix, and value matrix, respectively. In our setting, the query $\mathbf{Q}$ stems from the inputs itself, i.e., $\mathbf{Q} = \mathbf{K} = \mathbf{V} = \mathbf{E}$.

To explore the high parallelizability of attention, transformer encoder designs a multi-head attention mechanism based on the scaled dot-product attention. More concretely, multi-head attention first linearly projects the queries, keys and values $h$ times by using different linear projections. Then $h$ projections perform the scaled dot-product attention in parallel. Finally, these results of attention are concatenated and once again projected to get the new representation. Formally, the multi-head attention can be formulated as follows:

$$head_i = \text{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V) \quad (2)$$

$$\begin{aligned}\mathbf{H} &= \text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) \\ &= \text{Concat}(head_1, head_2, ..., head_h)\mathbf{W}^o\end{aligned} \quad (3)$$

where $\mathbf{W}_i^Q \in \mathbb{R}^{(d_p+d_w)\times d_k}$, $\mathbf{W}_i^K \in \mathbb{R}^{(d_p+d_w)\times d_k}$, $\mathbf{W}_i^V \in \mathbb{R}^{(d_p+d_w)\times d_k}$ are trainable projection parameters. $d_k$ is $(d_p + d_w)/h$, $h$ is the number of heads. In Eq.(3), $\mathbf{W}^o \in \mathbb{R}^{(d_p+d_w)\times(d_p+d_w)}$ is also trainable parameter.

### 3.3 Selected Sharing Layer

In order to select valuable and appropriate shared features for different tasks, we design a selected sharing layer following the shared layer. The selected sharing layer consists of two cells: gated sharing cell for filtering useless features and attention sharing cell for focusing on valuable shared features for specific tasks. The description of this layer is depicted in Figure 2 and Figure 3. In the following, we introduce two cells in details.

**Gated Sharing Cell** Inspired by forgotten gate mechanism of LSTM (Hochreiter and Schmidhuber, 1997) and GRU (Chung et al., 2014), we design a single gated cell to filter useless shared features from shared layer. There are two reasons why we adopt single-gate mechanism. One is that transformer encoder in shared layer can efficiently capture the features of long-range dependencies. The features do not need to capture repeatedly by multiple complex gate mechanisms of LSTM and GRU. The other is that single-gate mechanism is more convenient for training (Srivastava et al., 2015). Formally, the gated sharing cell can be expressed as follows:

$$\mathbf{g}_{fake} = \sigma(\mathbf{W}_{fake} \cdot \mathbf{H}_{shared} + \mathbf{b}_{fake}) \quad (4)$$

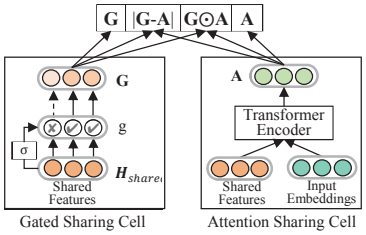


Figure 3: The details of selected sharing layer.

where $\mathbf{H}_{shared} \in \mathbb{R}^{1\times l(d_p+d_w)}$ denotes the outputs of shared layer upstream, $\mathbf{W}_{fake} \in \mathbb{R}^{l(d_p+d_w)\times l(d_p+d_w)}$ and $\mathbf{b}_{fake} \in \mathbb{R}^{1\times l(d_p+d_w)}$ are trainable parameters. $\sigma$ is a non-linear activation - sigmoid, which makes final choices for retaining and discarding features in shared layer.

Then the shared features after filtering via gated sharing cell $\mathbf{g}_{fake}$ for the task of fake news detection are represented as:

$$\mathbf{G}_{fake} = \mathbf{g}_{fake} \odot \mathbf{H}_{shared} \quad (5)$$

where $\odot$ denotes element-wise multiplication.

Similarly, for the auxiliary task - the task of stance detection, filtering process in the gated sharing cell is the same as the task of fake news detection, so we do not reiterate them here.

**Attention Sharing Cell** To focus on helpful shared features that are beneficial to specific tasks from upstream shared layer, we devise an attention sharing cell based on attention mechanism. Specifically, this cell utilizes input embeddings of the specific task to weight shared features for paying more attention to helpful features. The inputs of this cell include two matrixes: the input embeddings of the specific task and the shared features of both tasks. The basic attention architecture of this cell, the same as shared-private feature extractor, also adopts transformer encoder (the details in subsection 3.2). However, in this architecture, query matrix and key matrix are not projections of the same matrix, i.e., query matrix $\mathbf{E}_{fake}$ is the input embeddings of fake news detection task, and key matrix $\mathbf{K}_{shared}$ and value matrix $\mathbf{V}_{shared}$ are the projections of shared features $\mathbf{H}_{shared}$. Formally, the attention sharing cell can be formalized as follows:

$$\begin{aligned}head_i = \text{Attention}(&\\ &\mathbf{E}_{fake}\mathbf{W}_i^Q, \mathbf{K}_{shared}\mathbf{W}_i^K, \mathbf{V}_{shared}\mathbf{W}_i^V)\end{aligned} \quad (6)$$

$$\begin{aligned}\mathbf{A}_{fake} &= \text{MultiHead}(\mathbf{H}_{fake}, \mathbf{K}_{shared}, \mathbf{V}_{shared}) \\ &= \text{Concat}(head_1, head_2, ..., head_h)\mathbf{W}^o\end{aligned} \quad (7)$$

where the dimensions of $\mathbf{E}_{fake}$, $\mathbf{K}_{shared}$, and $\mathbf{V}_{shared}$ are all $\mathbb{R}^{l\times(d_p+d_w)}$. The dimensions of remaining parameters in Eqs.(6, 7) are the same as in Eqs.(2, 3). Moreover, in order to guarantee the diversity of focused shared features, the number of heads $h$ should not be set too large. Experiments show that our method performs the best performance when $h$ is equal to 2.

**Integration of the Two Cells** We first convert the output of the two cells to vectors **G** and **A**, respectively, and then integrate the vectors in full by the absolute difference and element-wise product (Mou et al., 2016).

$$\mathbf{SSL} = [\mathbf{G}; |\mathbf{G} - \mathbf{A}|; \mathbf{G} \odot \mathbf{A}; \mathbf{A}] \quad (8)$$

where $\odot$ denotes element-wise multiplication and ; denotes concatenation.

### 3.4 The Output Layer

As the last layer, softmax functions are applied to achieve the classification of different tasks, which emits the prediction of probability distribution for the specific task $i$.

$$\hat{\mathbf{y}}_i = \text{softmax}(\mathbf{W}_i\mathbf{F}_i + \mathbf{b}_i) \quad (9)$$

$$\mathbf{F}_i = [\mathbf{H}_i; \mathbf{SSL}_i] \quad (10)$$

where $\hat{\mathbf{y}}_i$ is the predictive result, $\mathbf{F}_i$ is the concatenation of private features $\mathbf{H}_i$ of task $i$ and the outputs $\mathbf{SSL}_i$ of selected sharing layer for task $i$. $\mathbf{W}_i$ and $\mathbf{b}_i$ are trainable parameters.

Given the prediction of all tasks, a global loss function forces the model to minimize the cross-entropy of prediction and true distribution for all the tasks:

$$\mathcal{L} = \sum_{i=1}^{N} \lambda_i L(\hat{\mathbf{y}}_i, \mathbf{y}_i) \quad (11)$$

$$L(\hat{\mathbf{y}}_i, \mathbf{y}_i) = \mathbf{y}_i log\hat{\mathbf{y}}_i + (1 - \mathbf{y}_i)log(1 - \hat{\mathbf{y}}_i) \quad (12)$$

where $\lambda_i$ is the weight for the task $i$, and $N$ is the number of tasks. In this paper, $N = 2$, and we give more weight $\lambda$ to the task of fake news detection.

## 4 Experiments

### 4.1 Datasets and Evaluation Metrics

We use two public datasets for fake news detection and stance detection, i.e., RumourEval (Derczynski et al., 2017) and PHEME (Zubiaga et al., 2016b). We introduce both the datasets in details from three aspects: content, labels, and distribution.

**Content.** Both datasets contain Twitter conversation threads associated with different newsworthy events including the Ferguson unrest, the shooting at Charlie Hebdo, etc. A conversation thread consists of a tweet making a true and false claim, and a series of replies. **Labels.** Both datasets have the same labels on fake news detection and stance detection. Fake news is labeled as true, false, and unverified. Because we focus on classifying true and false tweets, we filter the unverified tweets. Stance of tweets is annotated as support, deny, query, and comment. **Distribution.** RumourEval contains 325 Twitter threads discussing rumours and PHEME includes 6,425 Twitter threads. Threads, tweets, and class distribution of the two datasets are shown in Table 1.

In consideration of the imbalance label distributions, in addition to accuracy (A) metric, we add Precision (P), Recall (R) and F1-score (F1) as complementary evaluation metrics for tasks. We hold out 10% of the instances in each dataset for model tuning, and the rest of the instances are performed 5-fold cross-validation throughout all experiments.

### 4.2 Settings

**Pre-processing** - Processing useless and inappropriate information in text: (1) removing nonalphabetic characters; (2) removing website links of text content; (3) converting all words to lower case and tokenize texts.

**Parameters** - hyper-parameters configurations of our model: for each task, we strictly turn all the hyper-parameters on the validation dataset, and we achieve the best performance via a small grid search. The sizes of word embeddings and position embeddings are set to 200 and 100. In transformer encoder, attention heads and blocks are set to 6 and 2 respectively, and the dropout of multi-head attention is set to 0.7. Moreover, the minibatch size is 64; the initial learning rate is set to 0.001, the dropout rate to 0.3, and $\lambda$ to 0.6 for fake news detection.

### 4.3 Performance Evaluation

#### 4.3.1 Baselines

**SVM** A Support Vector Machines model in (Derczynski et al., 2017) detects misinformation relying on manually extracted features.

**CNN** A Convolutional Neural Network model (Chen et al., 2017) employs pre-trained word embeddings based on Word2Vec as input embeddings

| Datasets | Threads | Tweets | True | False | Unverified | Support | Deny | Query | Comment |
|---|---|---|---|---|---|---|---|---|---|
| RumourEval | 325 | 5,568 | 145 | 74 | 106 | 1,004 | 415 | 464 | 3,685 |
| PHEME | 6,425 | 105,354 | 1,067 | 638 | 697 | 891 | 335 | 353 | 2,855 |

Table 1: Statistics of the two datasets.

| Dataset | Measure | SVM | CNN | TE | DeClarE | MTL-LSTM | TRNN | Bayesian-DL | Ours |
|---|---|---|---|---|---|---|---|---|---|
| RumourEval | A(%) | 71.42 | 61.90 | 66.67 | 66.67 | 66.67 | 76.19 | 80.95 | **81.48** |
| | P(%) | 66.67 | 54.54 | 60.00 | 58.33 | 57.14 | 70.00 | **77.78** | 72.24 |
| | R(%) | 66.67 | 66.67 | 66.67 | 77.78 | **88.89** | 77.78 | 77.78 | 86.31 |
| | F1(%) | 66.67 | 59.88 | 63.15 | 66.67 | 69.57 | 73.68 | 77.78 | **78.65** |
| PHEME | A(%) | 72.18 | 59.23 | 65.22 | 67.87 | 74.94 | 78.65 | 80.33 | **81.27** |
| | P(%) | **78.80** | 56.14 | 63.05 | 64.68 | 68.77 | 77.11 | 78.29 | 73.41 |
| | R(%) | 75.75 | 64.64 | 64.64 | 71.21 | 87.87 | 78.28 | 79.29 | **88.10** |
| | F1(%) | 72.10 | 60.09 | 63.83 | 67.89 | 77.15 | 77.69 | 78.78 | **80.09** |

Table 2: Performance comparison of our sifted multi-task learning method against the baselines.

to capture features similar to n-grams.

**TE** Tensor Embeddings (Guacho et al., 2018) leverages tensor decomposition to derive concise claim embeddings, which are used to create a claim-by-claim graph for label propagation.

**DeClarE** Evidence-Aware Deep Learning (Popat et al., 2018) encodes claims and articles by Bi-LSTM and focuses on each other based on attention mechanism, and then concatenates claim source and article source information.

**MTL-LSTM** A multi-task learning model based on LSTM networks (Kochkina et al., 2018) trains jointly the tasks of veracity classification, rumor detection, and stance detection.

**TRNN** Tree-structured RNN (Ma et al., 2018b) is a bottom-up and a top-down tree-structured model based on recursive neural networks.

**Bayesian-DL** Bayesian Deep Learning model (Zhang et al., 2019) first adopts Bayesian to represent both the prediction and uncertainty of claim and then encodes replies based on LSTM to update and generate a posterior representations.

#### 4.3.2 Compared with State-of-the-art Methods

We perform experiments on RumourEval and PHEME datasets to evaluate the performance of our method and the baselines. The experimental results are shown in Table 2. We gain the following observations:

- On the whole, most well-designed deep learning methods, such as ours, Bayesian-DL, and TRNN, outperform feature engineering-based methods, like SVM. This illustrates that deep learning methods can represent better intrinsic semantics of claims and replies.
- In terms of recall (R), our method and MTL-LSTM, both based on multi-task learning, achieve more competitive performances than other baselines, which presents that sufficient features are shared for each other among multiple tasks. Furthermore, our method reflects a more noticeable performance boost than MTL-LSTM on both datasets, which extrapolates that our method earns more valuable shared features.
- Although our method shows relatively low performance in terms of precision (P) and recall (R) compared with some specific models, our method achieves the state-of-the-art performance in terms of accuracy (A) and F1-score (F1) on both datasets. Taking into account the tradeoff among different performance measures, this reveals the effectiveness of our method in the task of fake news detection.

### 4.4 Discussions

#### 4.4.1 Model Ablation

To evaluate the effectiveness of different components in our method, we ablate our method into several simplified models and compare their performance against related methods. The details of these methods are described as follows:

**Single-task** Single-task is a model with transformer encoder as the encoder layer of the model for fake news detection.

**MT-lstm** The tasks of fake news detection and stance detection are integrated into a shared-private model and the encoder of the model is achieved by LSTM.

| | RumourEval | | | | PHEME | | | |
|---|---|---|---|---|---|---|---|---|
| | A(%) | P(%) | R(%) | F1(%) | A(%) | P(%) | R(%) | F1(%) |
| Single-task | 62.86 | 54.21 | 65.43 | 59.29 | 60.94 | 55.56 | 64.53 | 58.57 |
| MT-lstm | 65.90 | 56.61 | 85.60 | 68.15 | 71.61 | 66.24 | 85.31 | 75.29 |
| MT-trans | 71.67 | 58.43 | 78.78 | 67.10 | 75.57 | 65.73 | 78.94 | 71.73 |
| MT-trans-G | 74.89 | 64.32 | 81.68 | 71.97 | 76.24 | 67.32 | 83.56 | 74.57 |
| MT-trans-A | 79.22 | 68.97 | 84.86 | 76.10 | 79.54 | 70.90 | 86.71 | 78.01 |
| MT-trans-G-A | 82.10 | 72.24 | 86.31 | 78.65 | 81.27 | 73.41 | 88.10 | 80.09 |

Table 3: Ablation analysis of the sifted multi-task learning method.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **SL:** | what | why | wrong | scary | like | fact | | | great | probably |
| **SSL-FND:** | what | | wrong | scary | | fact | few | | false | real |
| **SSL-SD:** | | why | wrong | | like | | | hold | confirm | misleading |
| **PL-FND:** | what | | | scary | | few | really | | no | notice |
| **PL-SD:** | | why | | | like | | really | hold | sure | doubt |

Figure 4: Typical tokens obtained by different layers of the sifted multi-task learning method. In our proposed method, typical tokens are captured by shared layer (SL), selected sharing layer for fake news detection (SSL-FND), selected sharing layer for stance detection (SSL-SD), private layer for fake news detection (PL-FND), and private layer for stance detection (PL-SD) respectively. A column of the same color represents the distribution of one token in different layers, while the last two columns denote unique tokens captured by different layers.

**MT-trans** The only difference between MT-trans and MT-lstm is that encoder of MT-trans is composed of transformer encoder.

**MT-trans-G** On the basis of MT-trans, MT-trans-G adds gated sharing cell behind the shared layer of MT-trans to filter shared features.

**MT-trans-A** Unlike MT-trans-G, MT-trans-A replaces gated sharing cell with attention sharing cell for selecting shared features.

**MT-trans-G-A** Gated sharing cell and attention sharing cell are organically combined as selected sharing layer behind the shared layer of MT-trans, called MT-trans-G-A.

Table 3 provides the experimental results of these methods on RumourEval and PHEME datasets. We have the following observations:

- Effectiveness of multi-task learning. MT-trans boosts about 9% and 15% performance improvements in accuracy on both datasets compared with Single-task, which indicates that the multi-task learning method is effective to detect fake news.

- Effectiveness of transformer encoder. Compared with MT-lstm, MT-trans obtains more excellent performance, which explains that transformer encoder has better encoding ability than LSTM for news text on social media.

- Effectiveness of the selected sharing layer. Analysis of the results of the comparison with MT-trans, MT-trans-G, MT-Trans-A, and MT-trans-G-A shows that MT-trans-G-A ensures optimal performance with the help of the selected sharing layer of the model, which confirms the reasonability of selectively sharing different features for different tasks.

#### 4.4.2 Error Analysis

Although the sifted multi-task learning method outperforms previous state-of-the-art methods on two datasets (From Table 2), we observe that the proposed method achieves more remarkable performance boosts on PHEME than on RumourEval. There are two reasons for our analysis according to Table 1 and Table 2. One is that the number of training examples in RumourEval (including 5,568 tweets) is relatively limited as compared with PHEME (including 105,354 tweets), which is not enough to train deep neural networks. Another is that PHEME includes more threads (6,425 threads) than RumourEval (325 threads) so that PHEME can offer more rich credibility features to our proposed method.

### 4.5 Case Study

In order to obtain deeper insights and detailed interpretability about the effectiveness of the selected shared layer of the sifted multi-task learning method, we devise experiments to explore some ideas in depth: 1) Aiming at different tasks, what effective features can the selected sharing layer in our method obtain? 2) In the selected sharing layer, what features are learned from different cells?

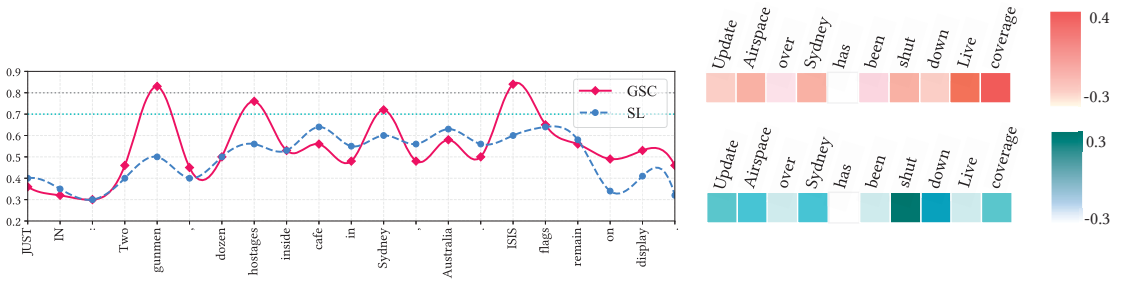


(a) Obtained weights of tokens by two models

(b) Neuron behaviours of $\mathbf{A}_{fake}$ and $\mathbf{A}_{stance}$

Figure 5: (a) In fake news detection task, the GSC line denotes the weight values $\mathbf{g}_{fake}$ of gated sharing cell, while the SL line represents feature weights of $\mathbf{H}_{shared}$ in the shared layer. Two horizontal lines give two different borders to determine the importance of tokens. (b) The red and green heatmaps describe the neuron behaviours of attention sharing cell $\mathbf{A}_{fake}$ in fake news detection task and $\mathbf{A}_{stance}$ in stance detection task, respectively.

#### 4.5.1 The Visualization of Shared Features Learned from Two Tasks

We visualize shared features learned from the tasks of fake news detection and stance detection. Specifically, we first look up these elements with the largest values from the outputs of the shared layer and the selected shared layer respectively. Then, these elements are mapped into the corresponding values in input embeddings so that we can find out specific tokens. The experimental results are shown in Figure 4. We draw the following observations:

- Comparing PL-FND and PL-SD, private features in private layer from different tasks are different. From PL-FND, PL-SD, and SLT, the combination of the private features and shared features from shared layer increase the diversity of features and help to promote the performance of both fake news detection and stance detection.

- By compared SL, SSL-FND, and SSL-SD, selected sharing layers from different tasks can not only filter tokens from shared layer (for instance, 'what', 'scary', and 'fact' present in SL but not in SSL-SD), but also capture helpful tokens for its own task (like 'false' and 'real' in SSL-FND, and 'confirm' and 'misleading' in SSL-SD).

#### 4.5.2 The Visualization of Different Features Learned from Different Cells

To answer the second question, we examine the neuron behaviours of gated sharing cell and attention sharing cell in the selected sharing layer, respectively. More concretely, taking the task of fake news detection as an example, we visualize feature weights of $\mathbf{H}_{shared}$ in the shared layer and show the weight values $\mathbf{g}_{fake}$ in gated sharing cell. By that we can find what kinds of features are discarded as interference, as shown in Figure 5(a). In addition, for attention sharing cell, we visualize which tokens are concerned in attention sharing cell, as shown in Figure 5(b). From Figure 5(a) and 5(b), we obtain the following observations:

- In Figure 5(a), only the tokens "gunmen, hostages, Sydney, ISIS" give more attention compared with vanilla shared-private model (SP-M). In more details, 'gunmen' and 'ISIS' obtain the highest weights. These illustrate that gated sharing cell can effectively capture key tokens.

- In Figure 5(b), "live coverage", as a prominent credibility indicator, wins more concerns in the task of fake news detection than other tokens. By contrast, when the sentence of Figure 5(b) is applied to the task of stance detection, the tokens "shut down" obtain the maximum weight, instead of "live coverage". These may reveal that attention sharing cell focuses on different helpful features from the shared layer for different tasks.

## 5 Conclusion

In this paper, we explored a sifted multi-task learning method with a novel selected sharing structure for fake news detection. The selected sharing structure fused single gate mechanism for filtering useless shared features and attention mechanism for paying close attention to features that were helpful to target tasks. We demonstrated the effectiveness of the proposed method on two public, challenging

datasets and further illustrated by visualization experiments. There are several important directions remain for future research: (1) the fusion mechanism of private and shared features; (2) How to represent meta-data of fake news better to integrate into inputs.

## Acknowledgments

The research work is supported by "the World-Class Universities(Disciplines) and the Characteristic Development Guidance Funds for the Central Universities"(PY3A022), Shenzhen Science and Technology Project(JCYJ20180306170836595), the National Natural Science Fund of China (No.F020807), Ministry of Education Fund Project "Cloud Number Integration Science and Education Innovation" (No.2017B00030), Basic Scientific Research Operating Expenses of Central Universities (No.ZDYF2017006).

## References

Hunt Allcott and Matthew Gentzkow. 2017. Social media and fake news in the 2016 election. *Journal of economic perspectives*, 31(2):211–36.

Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. 2011. Information credibility on twitter. In *Proceedings of the 20th international conference on World wide web*, pages 675–684. ACM.

Xilun Chen, Ahmed Hassan Awadallah, Hany Hassan, Wei Wang, and Claire Cardie. 2018. Zero-resource multilingual model transfer: Learning what to share. *arXiv preprint arXiv:1810.03552*.

Xilun Chen and Claire Cardie. 2018. Multinomial adversarial networks for multi-domain text classification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1226–1240.

Yi-Chin Chen, Zhao-Yang Liu, and Hung-Yu Kao. 2017. Ikm at semeval-2017 task 8: Convolutional neural networks for stance detection and rumor verification. In *SemEval-2017*, pages 465–469.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*.

Niall J Conroy, Victoria L Rubin, and Yimin Chen. 2015. Automatic deception detection: Methods for finding fake news. *Proceedings of the Association for Information Science and Technology*, 52(1):1–4.

Leon Derczynski, Kalina Bontcheva, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Arkaitz Zubiaga. 2017. Semeval-2017 task 8: Rumoureval: Determining rumour veracity and support for rumours. In *SemEval-2017*, pages 69–76.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Sebastian Dungs, Ahmet Aker, Norbert Fuhr, and Kalina Bontcheva. 2018. Can rumour stance alone predict veracity? In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3360–3370.

Martin Flintham, Christian Karner, Khaled Bachour, Helen Creswick, Neha Gupta, and Stuart Moran. 2018. Falling for fake news: investigating the consumption of news via social media. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 376. ACM.

Gisel Bastidas Guacho, Sara Abdali, Neil Shah, and Evangelos E Papalexakis. 2018. Semi-supervised content-based detection of misinformation via tensor embeddings. In *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 322–325. IEEE.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Elena Kochkina, Maria Liakata, and Isabelle Augenstein. 2017. Turing at semeval-2017 task 8: Sequential approach to rumour stance classification with branch-lstm. *arXiv preprint arXiv:1704.07221*.

Elena Kochkina, Maria Liakata, and Arkaitz Zubiaga. 2018. All-in-one: Multi-task learning for rumour verification. *arXiv preprint arXiv:1806.03713*.

Sizhen Li, Shuai Zhao, Bo Cheng, and Hao Yang. 2018. An end-to-end multi-task learning model for fact checking. *EMNLP 2018*, page 138.

Boyang Liu, Pang-Ning Tan, and Jiayu Zhou. 2019. Augmented multi-task learning by optimal transport. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, pages 19–27. SIAM.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1–10.

Yunfei Long, Qin Lu, Rong Xiang, Minglei Li, and Chu-Ren Huang. 2017. Fake news detection through multi-perspective speaker profiles. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing*, pages 252–256.

Michal Lukasik, PK Srijith, Duy Vu, Kalina Bontcheva, Arkaitz Zubiaga, and Trevor Cohn. 2016. Hawkes processes for continuous time sequence classification: an application to rumour stance classification in twitter. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 393–398.

Jing Ma, Wei Gao, and Kam-Fai Wong. 2018a. Detect rumor and stance jointly by neural multi-task learning. In *Companion of the The Web Conference 2018 on The Web Conference 2018*, pages 585–593. International World Wide Web Conferences Steering Committee.

Jing Ma, Wei Gao, and Kam-Fai Wong. 2018b. Rumor detection on twitter with tree-structured recursive neural networks. In *ACL*, pages 1980–1989.

Marcelo Mendoza, Barbara Poblete, and Carlos Castillo. 2010. Twitter under crisis: Can we trust what we rt? In *Proceedings of the first workshop on social media analytics*, pages 71–79. ACM.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Mitra Mohtarami, Ramy Baly, James Glass, Preslav Nakov, Lluís Màrquez, and Alessandro Moschitti. 2018. Automatic stance detection using end-to-end memory networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 767–776.

Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Natural language inference by tree-based convolution and heuristic matching. In *ACL*, page 130.

Ray Oshikawa, Jing Qian, and William Yang Wang. 2018. A survey on natural language processing for fake news detection. *arXiv preprint arXiv:1811.00770*.

Kashyap Popat, Subhabrata Mukherjee, Andrew Yates, and Gerhard Weikum. 2018. Declare: Debunking fake news and false claims using evidence-aware deep learning. In *EMNLP*, pages 22–32.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2017. A stylometric inquiry into hyperpartisan and fake news. *arXiv preprint arXiv:1702.05638*.

Feng Qian, Chengyue Gong, Karishma Sharma, and Yan Liu. 2018. Neural user response generator: Fake news detection with collective user intelligence. In *IJCAI*, pages 3834–3840.

Natali Ruchansky, Sungyong Seo, and Yan Liu. 2017. Csi: A hybrid deep model for fake news detection. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 797–806. ACM.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *arXiv preprint arXiv:1505.00387*.

James Thorne, Mingjie Chen, Giorgos Myrianthous, Jiashu Pu, Xiaoxuan Wang, and Andreas Vlachos. 2017. Fake news stance detection using stacked ensemble of classifiers. In *Proceedings of the 2017 EMNLP Workshop: Natural Language Processing meets Journalism*, pages 80–83.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

William Yang Wang. 2017. " liar, liar pants on fire": A new benchmark dataset for fake news detection. *arXiv preprint arXiv:1705.00648*.

Yilin Wang, Suhang Wang, Jiliang Tang, Huan Liu, and Baoxin Li. 2015. Unsupervised sentiment analysis for social media images. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Lianwei Wu, Yuan Rao, Hualei Yu, Yiming Wang, and Nazir Ambreen. 2019. A multi-semantics classification method based on deep learning for incredible messages on social media. *Chinese Journal of Electronics*, 28(4):754–763.

Lianwei Wu, Yuan Rao, Hualei Yu, Yiming Wang, and Ambreen Nazir. 2018. False information detection on social media via a hybrid deep model. In *International Conference on Social Informatics*, pages 323–333. Springer.

Fan Yang, Yang Liu, Xiaohui Yu, and Min Yang. 2012. Automatic detection of rumor on sina weibo. In *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*, page 13. ACM.

Qiang Zhang, Aldo Lipani, Shangsong Liang, and Emine Yilmaz. 2019. Reply-aided detection of misinformation via bayesian deep learning. In *Companion Proceedings of The Web Conference*.

Arkaitz Zubiaga, Elena Kochkina, Maria Liakata, Rob Procter, and Michal Lukasik. 2016a. Stance classification in rumours as a sequential task exploiting the tree structure of social media conversations. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2438–2448.

Arkaitz Zubiaga, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Peter Tolmie. 2016b. Analysing how people orient to and spread rumours in social media by looking at conversational threads. *PloS one*, 11(3):e0150989.

# Text-based inference of moral sentiment change

**Jing Yi Xie***, **Renato Ferreira Pinto, Jr.***, **Graeme Hirst , and Yang Xu**

Department of Computer Science, University of Toronto, Toronto, Canada
jingyi.xie@mail.utoronto.ca, {renato,gh,yangxu}@cs.toronto.edu

## Abstract

We present a text-based framework for investigating moral sentiment change of the public via longitudinal corpora. Our framework is based on the premise that language use can inform people's moral perception toward right or wrong, and we build our methodology by exploring moral biases learned from diachronic word embeddings. We demonstrate how a parameter-free model supports inference of historical shifts in moral sentiment toward concepts such as slavery and democracy over centuries at three incremental levels: moral relevance, moral polarity, and fine-grained moral dimensions. We apply this methodology to visualizing moral time courses of individual concepts and analyzing the relations between psycholinguistic variables and rates of moral sentiment change at scale. Our work offers opportunities for applying natural language processing toward characterizing moral sentiment change in society.

## 1 Moral sentiment change and language

People's moral sentiment—our feelings toward right or wrong—can change over time. For instance, the public's views toward *slavery* have shifted substantially over the past centuries (Oldfield, 2012). How society's moral views evolve has been a long-standing issue and a constant source of controversy subject to interpretations from social scientists, historians, philosophers, among others. Here we ask whether natural language processing has the potential to inform moral sentiment change in society at scale, involving minimal human labour or intervention.

The topic of moral sentiment has been thus far considered a traditional inquiry in philosophy (Hume, 1739; Smith, 1759; Kant, 1785), with contemporary development of this topic represented in social psychology (Piaget, 1932; Kohlberg, 1969; Stigler et al., 1990; Fiske and Taylor, 1991; Pizarro and Bloom, 2003), cognitive linguistics (Lakoff, 1996), and more recently, the advent of Moral Foundations Theory (Haidt and Joseph, 2004; Haidt et al., 2007; Graham et al., 2013). Despite the fundamental importance and interdisciplinarity of this topic, large-scale formal treatment of moral sentiment, particularly its evolution, is still in infancy from the natural language processing (NLP) community (see overview in Section 2).

We believe that there is a tremendous potential to bring NLP methodologies to bear on the problem of moral sentiment change. We build on extensive recent work showing that word embeddings reveal implicit human biases (Bolukbasi et al., 2016; Caliskan et al., 2017) and social stereotypes (Garg et al., 2018). Differing from this existing work, we demonstrate that moral sentiment change can be revealed by moral biases implicitly learned from diachronic text corpora. Accordingly, we present to our knowledge the first text-based framework for probing moral sentiment change at a large scale with support for different levels of analysis concerning *moral relevance*, *moral polarity*, and *fine-grained moral dimensions*. As such, for any query item such as *slavery*, our goal is to automatically infer its moral trajectories from sentiments at each of these levels over a long period of time.

Our approach is based on the premise that people's moral sentiments are reflected in natural language, and more specifically, in text (Bloom, 2010). In particular, we know that books are highly effective tools for conveying moral views to the public. For example, *Uncle Tom's Cabin* (Stowe, 1852) was central to the anti-slavery movement in the United States. The framework that we develop builds on this premise to explore

*Equal contribution.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4654–4663, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

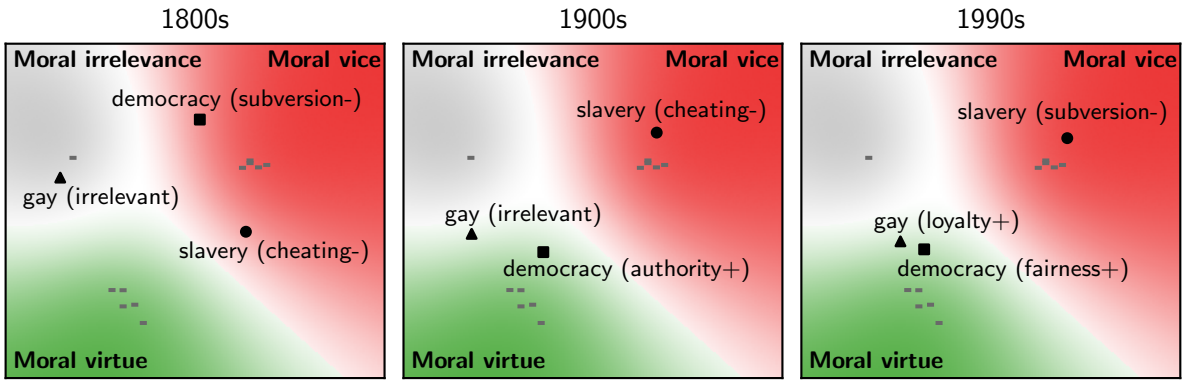


Figure 1: Illustration of moral sentiment change over the past two centuries. Moral sentiment trajectories of three probe concepts, *slavery*, *democracy*, and *gay*, are shown in moral sentiment embedding space through 2D projection from Fisher's discriminant analysis with respect to seed words from the classes of *moral virtue*, *moral vice*, and *moral irrelevance*. Parenthesized items represent moral categories predicted to be most strongly associated with the probe concepts. Gray markers represent the fine-grained centroids (or anchors) of these moral classes.

changes in moral sentiment reflected in longitudinal or historical text.

Figure 1 offers a preview of our framework by visualizing the evolution trajectories of the public's moral sentiment toward concepts signified by the probe words *slavery*, *democracy*, and *gay*. Each of these concepts illustrates a piece of "moral history" tracked through a period of 200 years (1800 to 2000), and our framework is able to capture nuanced moral changes. For instance, *slavery* initially lies at the border of moral virtue (positive sentiment) and vice (negative sentiment) in the 1800s yet gradually moves toward the center of moral vice over the 200-year period; in contrast, *democracy* considered morally negative (e.g., subversion and anti-authority under monarchy) in the 1800s is now perceived as morally positive, as a mechanism for fairness; *gay*, which came to denote homosexuality only in the 1930s (Kay et al., 2019), is inferred to be morally irrelevant until the modern day. We will describe systematic evaluations and applications of our framework that extend beyond these anecdotal cases of moral sentiment change.

The general text-based framework that we propose consists of a parameter-free approach that facilitates the prediction of public moral sentiment toward individual concepts, automated retrieval of morally changing concepts, and broad-scale psycholinguistic analyses of historical rates of moral sentiment change. We provide a description of the probabilistic models and data used, followed by comprehensive evaluations of our methodology.

## 2 Emerging NLP research on morality

An emerging body of work in natural language processing and computational social science has investigated how NLP systems can detect moral sentiment in online text. For example, moral rhetoric in social media and political discourse (Garten et al., 2016; Johnson and Goldwasser, 2018; Lin et al., 2018), the relation between moralization in social media and violent protests (Mooijman et al., 2018), and bias toward refugees in talk radio shows (Gillani and Levy, 2019) have been some of the topics explored in this line of inquiry. In contrast to this line of research, the development of a formal framework for moral sentiment change is still under-explored, with no existing systematic and formal treatment of this topic (Bloom, 2010).

While there is emerging awareness of ethical issues in NLP (Hovy et al., 2017; Alfano et al., 2018), work exploiting NLP techniques to study principles of moral sentiment change is scarce. Moreover, since morality is variable across cultures and time (Graham et al., 2013; Bloom, 2010), developing systems that capture the diachronic nature of moral sentiment will be a pivotal research direction. Our work leverages and complements existing research that finds implicit human biases from word embeddings (Bolukbasi et al., 2016; Caliskan et al., 2017; Garten et al., 2016) by developing a novel perspective on using NLP methodology to discover principles of moral sentiment change in human society.

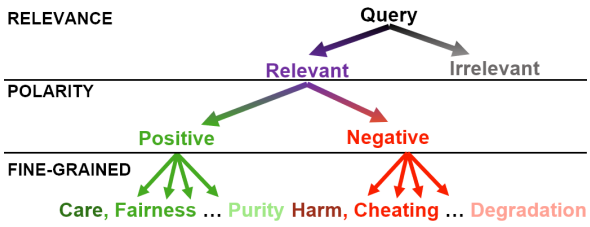


Figure 2: Illustration of the three-tier framework that supports moral sentiment inference at different levels.

## 3 A three-tier modelling framework

Our framework treats the moral sentiment toward a concept at three incremental levels, as illustrated in Figure 2. First, we consider moral relevance, distinguishing between morally irrelevant and morally relevant concepts. At the second tier, moral polarity, we further split morally relevant concepts into those that are positively or negatively perceived in the moral domain. Finally, a third tier classifies these concepts into fine-grained categories of human morality.

We draw from research in social psychology to inform our methodology, most prominently Moral Foundations Theory (MFT; Graham et al., 2013). MFT seeks to explain the structure and variation of human morality across cultures, and proposes five moral foundations: Care / Harm, Fairness / Cheating, Loyalty / Betrayal, Authority / Subversion, and Sanctity / Degradation. Each foundation is summarized by a positive and a negative pole, resulting in ten fine-grained moral categories.

### 3.1 Lexical data for moral sentiment

To ground moral sentiment in text, we leverage the Moral Foundations Dictionary (MFD; Graham et al., 2009). The MFD is a psycholinguistic resource that associates each MFT category with a set of *seed words*, which are words that provide evidence for the corresponding moral category in text. We use the MFD for moral polarity classification by dividing seed words into positive and negative sets, and for fine-grained categorization by splitting them into the 10 MFT categories.

To implement the first tier of our framework and detect moral relevance, we complement our morally relevant seed words with a corresponding set of seed words approximating moral irrelevance based on the notion of valence, i.e., the degree of pleasantness or unpleasantness of a stimulus. We refer to the emotional valence ratings collected by Warriner et al. (2013) for approximately 14,000 English words, and choose the words with most neutral valence rating that do not occur in the MFD as our set of morally irrelevant seed words, for an equal total number of morally relevant and morally irrelevant words.

### 3.2 Models

We propose and evaluate a set of probabilistic models to classify concepts in the three tiers of morality specified above. Our models exploit the semantic structure of word embeddings (Mikolov et al., 2013) to perform tiered moral classification of query concepts. In each tier, the model receives a query word embedding vector $\mathbf{q}$ and a set of seed words for each class in that tier, and infers the posterior probabilities over the set of classes $c$ to which the query concept is associated with.

The seed words function as "labelled examples" that guide the moral classification of novel concepts, and are organized per classification tier as follows. In moral relevance classification, sets $\mathbf{S}_0$ and $\mathbf{S}_1$ contain the morally irrelevant and morally relevant seed words, respectively; for moral polarity, $\mathbf{S}_+$ and $\mathbf{S}_-$ contain the positive and negative seed words; and for fine-grained moral categories, $\mathbf{S}_1, \ldots, \mathbf{S}_{10}$ contain the seed words for the 10 categories of MFT. Then our general problem is to estimate $p(c \mid \mathbf{q})$, where $\mathbf{q}$ is a query vector and $c$ is a moral category in the desired tier.

We evaluate the following four models:

- A Centroid model summarizes each set of seed words by its expected vector in embedding space, and classifies concepts into the class of closest expected embedding in Euclidean distance following a softmax rule;
- A Naïve Bayes model considers both mean and variance, under the assumption of independence among embedding dimensions, by fitting a normal distribution with mean vector and diagonal covariance matrix to the set of seed words of each class;
- A $k$-Nearest Neighbors ($k$NN) model exploits local density estimation and classifies concepts according to the majority vote of the $k$ seed words closest to the query vector;
- A Kernel Density Estimation (KDE) model performs density estimation at a broader scale by considering the contribution of each seed word toward the total likelihood of each

| Model | Parameter | Posterior Inference |
|---|---|---|
| Centroid | – | $p(c \mid \mathbf{q}) \propto \exp\left(-\left\|\mathbf{q} - \mathrm{E}\left[\mathbf{S}_c\right]\right\|\right)$ |
| Naïve Bayes | – | $p(c \mid \mathbf{q}) \propto \prod_{j=1}^{d} f_N\left(\mathbf{q}_j; \mu = \mathrm{E}\left[\mathbf{S}_{c,j}\right], \sigma^2 = \mathrm{Var}\left[\mathbf{S}_{c,j}\right]\right)$ |
| $k$-Nearest Neighbors ($k$NN) | $k$ | $p(c \mid \mathbf{q}) \propto \left\lvert \{k \text{ nearest seed words to } \mathbf{q}\} \cap \mathbf{S}_c \right\rvert$ |
| Kernel Density Estimation (KDE) | $h$ | $p(c \mid \mathbf{q}) \propto \frac{1}{\lvert\mathbf{S}_c\rvert} \sum_{\mathbf{w} \in \mathbf{S}_c} f_{MN}\left(\mathbf{q}; \mu = \mathbf{w}, \Sigma = \mathrm{diag}(h)\right)$ |

Table 1: Summary of models for moral sentiment classification. Each model infers moral sentiment of a query word vector $\mathbf{q}$ based on moral classes $c$ (at any of the three levels) represented by moral seed words $\mathbf{S}_c$. $\mathrm{E}\left[\mathbf{S}_c\right]$ is the mean vector of $\mathbf{S}_c$; $\mathrm{E}\left[\mathbf{S}_{c,j}\right], \mathrm{Var}\left[\mathbf{S}_{c,j}\right]$ refer to the mean and variance of $\mathbf{S}_c$ along the $j$-th dimension in embedding space. $d$ is the number of embedding dimensions; and $f_N, f_{MN}$ refer to the density functions of univariate and multivariate normal distributions, respectively.

class, regulated by a bandwidth parameter $h$ that controls the sensitivity of the model to distance in embedding space.

Table 1 specifies the formulation of each model. Note that we adopt a parsimonious design principle in our modelling: both Centroid and Naïve Bayes are parameter-free models, $k$NN only depends on the choice of $k$, and KDE uses a single bandwidth parameter $h$.

## 4 Historical corpus data

To apply our models diachronically, we require a word embedding space that captures the meanings of words at different points in time and reflects changes pertaining to a particular word as diachronic shifts in a common embedding space.

Following Hamilton et al. (2016), we combine skip-gram word embeddings (Mikolov et al., 2013) trained on longitudinal corpora of English with rotational alignments of embedding spaces to obtain diachronic word embeddings that are aligned through time.

We divide historical time into decade-long bins, and use two sets of embeddings provided by Hamilton et al. (2016), each trained on a different historical corpus of English:

- Google N-grams (Lin et al., 2012): a corpus of $8.5 \times 10^{11}$ tokens collected from the English literature (Google Books, all-genres) spanning the period 1800–1999.
- COHA (Davies, 2010): a smaller corpus of $4.1 \times 10^{8}$ tokens from works selected so as to be genre-balanced and representative of American English in the period 1810–2009.

## 5 Model evaluations

We evaluated our models in two ways: classification of moral seed words on all three tiers (moral relevance, polarity, and fine-grained categories), and correlation of model predictions with human judgments.

### 5.1 Moral sentiment inference of seed words

In this evaluation, we assessed the ability of our models to classify the seed words that compose our moral environment in a leave-one-out classification task. We performed the evaluation for all three classification tiers: 1) moral relevance, where seed words are split into morally relevant and morally irrelevant; 2) moral polarity, where moral seed words are split into positive and negative; 3) fine-grained categories, where moral seed words are split into the 10 MFT categories. In each test, we removed one seed word from the training set at a time to obtain cross-validated model predictions.

Table 2 shows classification accuracy for all models and corpora on each tier for the 1990–1999 period.[1] We observe that all models perform substantially better than chance, confirming the efficacy of our methodology in capturing moral dimensions of words. We also observe that models using word embeddings trained on Google N-grams perform better than those trained on COHA, which could be expected given the larger corpus size of the former.

In the remaining analyses, we employ the Centroid model, which offers competitive accuracy and a simple, parameter-free specification.

### 5.2 Alignment with human valence ratings

We evaluated the approximate agreement between our methodology and human judgments using valence ratings, i.e., the degree of pleasantness or un-

[1] We also computed average accuracy over all decades using Google N-grams, the only corpus covering all moral categories through time. See Supplementary Material.

| | Google N-grams | | | COHA | | |
|---|---|---|---|---|---|---|
| **Model** | **Relevance** | **Polarity** | **Category** | **Relevance** | **Polarity** | **Category** |
| Random | 0.50 | 0.50 | 0.10 | 0.50 | 0.50 | 0.10 |
| Centroid | **0.84** | 0.90 | **0.59** | 0.78 | **0.80** | **0.40** |
| Naïve Bayes | **0.84** | 0.89 | 0.53 | 0.76 | 0.78 | 0.39 |
| 1-NN | 0.80 | 0.88 | 0.53 | 0.74 | 0.76 | 0.32 |
| 5-NN | 0.83 | **0.93** | 0.57 | 0.74 | 0.75 | 0.33 |
| KDE | 0.82 | 0.90 | 0.57 | **0.80** | 0.76 | 0.33 |

Table 2: Classification accuracy of moral seed words for moral relevance, moral polarity, and fine-grained moral categories based on 1990–1999 word embeddings for two independent corpora, Google N-grams and COHA.

| **Corpus** | **Correlation** |
|---|---|
| Google N-grams | 0.43 ($n = 12293; p < 0.0001$) |
| COHA | 0.38 ($n = 7141; p < 0.0001$) |

Table 3: Pearson correlations between model predicted moral sentiment polarities and human valence ratings.

pleasantness of a stimulus. Our assumption is that the valence of a concept should correlate with its perceived moral polarity, e.g., morally repulsive ideas should evoke an unpleasant feeling. However, we do not expect this correspondence to be perfect; for example, the concept of *dessert* evokes a pleasant reaction without being morally relevant.

In this analysis, we took the valence ratings for the nearly 14,000 English nouns collected by Warriner et al. (2013) and, for each query word $q$, we generated a corresponding prediction of positive moral polarity from our model, $P(c_+ \mid \mathbf{q})$. Table 3 shows the correlations between human valence ratings and predictions of positive moral polarity generated by models trained on each of our corpora. We observe that the correlations are significant, suggesting the ability of our methodology to capture relevant features of moral sentiment from text.

In the remaining applications, we use the diachronic embeddings trained on the Google N-grams corpus, which enabled superior model performance throughout our evaluations.

## 6 Applications to diachronic morality

We applied our framework in three ways: 1) evaluation of selected concepts in historical time courses and prediction of human judgments; 2) automatic detection of moral sentiment change; and 3) broad-scale study of the relations between psycholinguistic variables and historical change of moral sentiment toward concepts.

### 6.1 Moral change in individual concepts

**Historical time courses.** We applied our models diachronically to predict time courses of moral relevance, moral polarity, and fine-grained moral categories toward two historically relevant topics: slavery and democracy. By grounding our model in word embeddings for each decade and querying concepts at the three tiers of classification, we obtained the time courses shown in Figure 3.

We note that these trajectories illustrate actual historical trends. Predictions for democracy show a trend toward morally positive sentiment, consistent with the adoption of democratic regimes in Western societies. On the other hand, predictions for slavery trend down and suggest a drop around the 1860s, coinciding with the American Civil War. We also observe changes in the dominant fine-grained moral categories, such as the perception of democracy as a fair concept, suggesting potential mechanisms behind the polarity changes and providing further insight into the public sentiment toward these concepts as evidenced by text.

**Prediction of human judgments.** We explored the predictive potential of our framework by comparing model predictions with human judgments of moral relevance and acceptability. We used data from the Pew Research Center's 2013 Global Attitudes survey (Pew Research Center, 2013), in which participants from 40 countries judged 8 topics such as *abortion* and *homosexuality* as one of "acceptable", "unacceptable", and "not a moral issue".

We compared human ratings with model predictions at two tiers: for moral relevance, we paired the proportion of "not a moral issue" human responses with irrelevance predictions $p(c_0 \mid \mathbf{q})$ for

| Concept | Rate (e−03/decade) | Relevant Moral Category | Switching Period |
|---|---|---|---|
| abortion | 4.24** | cheating− | 1890 |
| propriety | 4.06*** | fairness+ | 1870 |
| commandment | 3.68*** | sanctity+ | 1880 |
| righteousness | 3.56*** | sanctity+ | 1800 |
| authorities | 3.42*** | authority+ | 1890 |
| apostle | 3.41*** | authority+ | 1900 |
| intervention | 3.33** | authority+ | 1860 |
| jew | 3.32*** | degradation− | 1870 |
| foreigner | 3.30*** | authority+ | 1800 |
| individuality | 3.26*** | authority+ | 1860 |

Table 4: Top 10 changing words towards moral relevance during 1800–2000, with model-inferred moral category and switching period. *, **, and *** denote $p < 0.05$, $p < 0.001$, and $p < 0.0001$, all Bonferroni-corrected.

| Concept | Rate (e−03/decade) | Early Category | Modern Category | Switching Period |
|---|---|---|---|---|
| wage | 4.38** | subversion− | fairness+ | 1810 |
| commitment | 3.78 | harm− | authority+ | 1830 |
| innovation | 3.21 | authority+ | authority+ | 1800 |
| help | 3.20** | sanctity+ | care+ | 1880 |
| mandate | 3.17* | authority+ | authority+ | 1800 |
| guidance | 3.04*** | authority+ | authority+ | 1800 |
| licence | 3.01* | authority+ | authority+ | 1800 |
| abortion | 2.95 | degradation− | cheating− | 1890 |
| democracy | 2.93** | authority+ | fairness+ | 1800 |
| disclosure | 2.89*** | fairness+ | fairness+ | 1840 |
| propaganda | −7.75 * | authority+ | subversion− | 1910 |
| humiliation | −4.05 *** | authority+ | harm− | 1800 |
| seriousness | −4.02 ** | authority+ | harm− | 1800 |
| legacy | −3.76 *** | authority+ | subversion− | 1800 |
| behavior | −3.73 ** | harm− | authority+ | 1830 |
| cheerfulness | −3.66 *** | authority+ | sanctity+ | 1800 |
| candour | −3.37 * | authority+ | degradation− | 1800 |
| offense | −3.37 | fairness+ | subversion− | 1840 |
| indulgence | −3.37 *** | authority+ | degradation− | 1800 |
| exertion | −3.26 *** | authority+ | degradation− | 1800 |

Table 5: Top 10 changing words towards moral positive (upper panel) and negative (lower panel) polarities, with model-inferred most representative moral categories during historical and modern periods and the switching periods. *, **, and *** denote $p < 0.05$, $p < 0.001$, and $p < 0.0001$, all Bonferroni-corrected for multiple tests.

each topic, and for moral acceptability, we paired the proportion of "acceptable" responses with positive predictions $p(c_+ \mid \mathbf{q})$. We used 1990s word embeddings, and obtained predictions for two-word topics by querying the model with their averaged embeddings.

Figure 4 shows plots of relevance and polarity predictions against survey proportions, and we observe a visible correspondence between model predictions and human judgments despite the difficulty of this task and limited number of topics.

### 6.2 Retrieval of morally changing concepts

Beyond analyzing selected concepts, we applied our framework predictively on a large repertoire of words to automatically discover the concepts that have exhibited the greatest change in moral sentiment at two tiers, moral relevance and moral polarity.

We selected the 10,000 nouns with highest total frequency in the 1800–1999 period according to data from Hamilton et al. (2016), restricted to words labelled as nouns in WordNet (Miller,

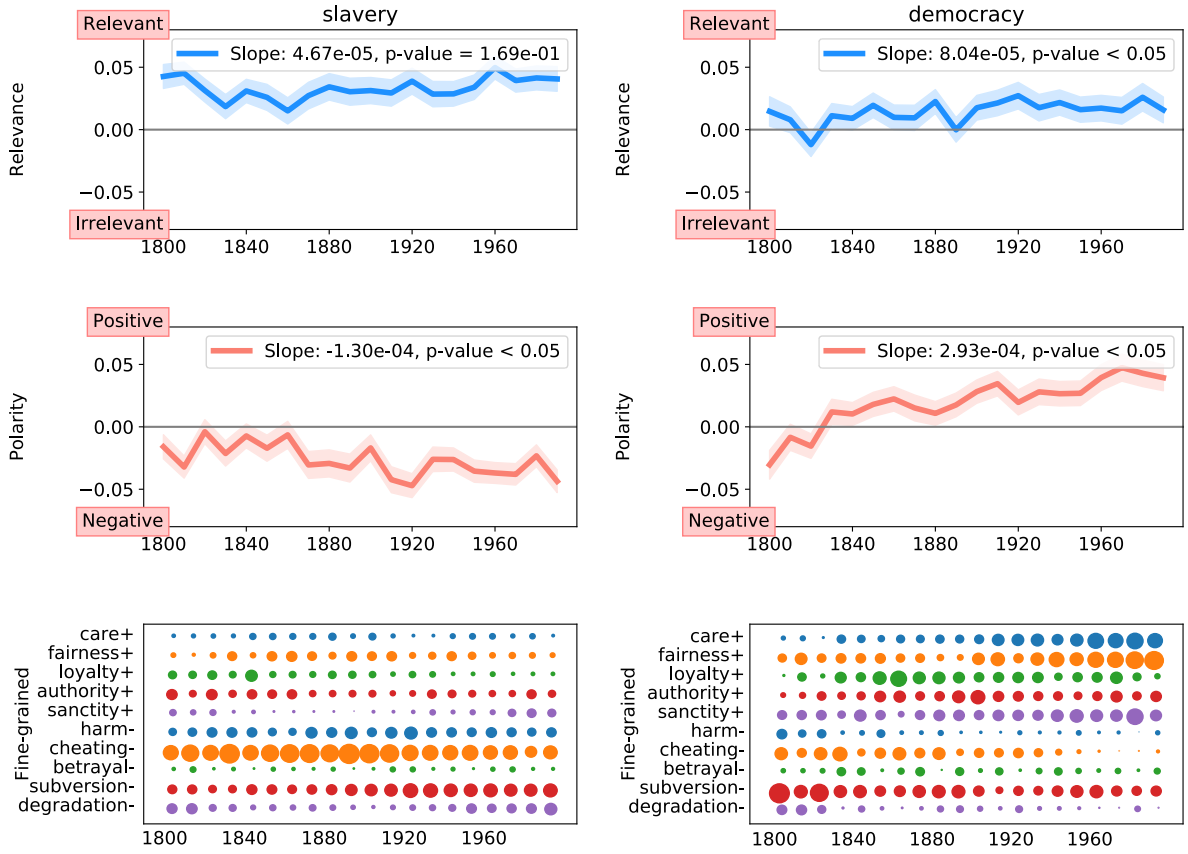


Figure 3: Moral sentiment time courses of *slavery* (left) and *democracy* (right) at each of the three levels, inferred by the Centroid model. Time courses at the moral relevance and polarity levels are in log odds ratios, and those for the fine-grained moral categories are represented by circles with sizes proportional to category probabilities.

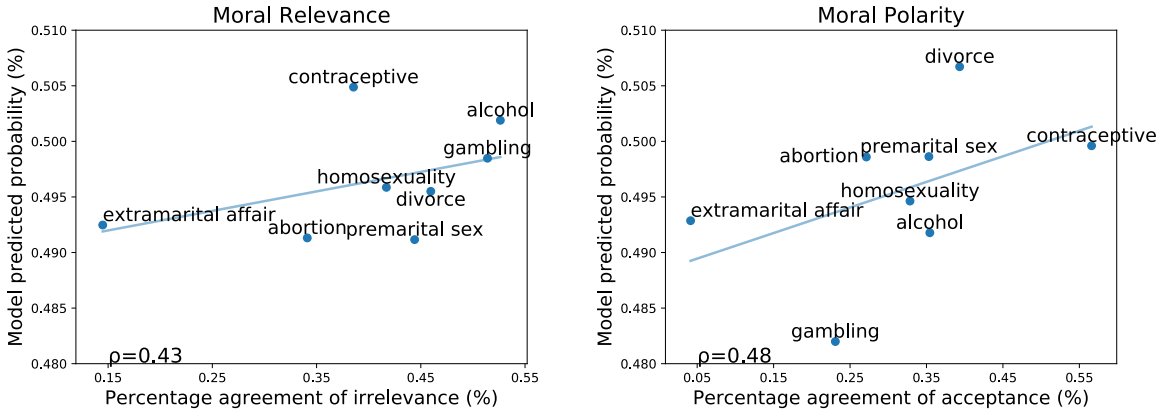


Figure 4: Model predictions against percentage of Pew respondents who selected "Not a moral concern" (left) or "Acceptable" (right), with lines of best fit and Pearson correlation coefficients $\rho$ shown in the background.

1995) for validation. For each such word $\mathbf{q}$, we computed diachronic moral relevance scores $R_i = p(c_1 \mid \mathbf{q}), i = 1,\ldots,20$ for the 20 decades in our time span. Then, we performed a linear regression of $R$ on $T = 1,\ldots,n$ and took the fitted slope as a measure of moral relevance change. We repeated the same procedure for moral polarity. Finally, we removed words with average relevance score below 0.5 to focus on morally relevant retrievals.

Table 4 shows the words with steepest predicted change toward moral relevance, along with their predicted fine-grained moral categories in mod-

ern times (i.e., 1900–1999). Table 5 shows the words with steepest predicted change toward the positive and negative moral poles. To further investigate the moral sentiment that may have led to such polarity shifts, we also show the predicted fine-grained moral categories of each word at its earliest time of predicted moral relevance and in modern times. Although we do not have access to ground truth for this application, these results offer initial insight into the historical moral landscape of the English language at scale.

### 6.3 Broad-scale investigation of moral change

In this application, we investigated the hypothesis that concept concreteness is inversely related to change in moral relevance, i.e., that concepts considered more abstract might become morally relevant at a higher rate than concepts considered more concrete. To test this hypothesis, we performed a multiple linear regression analysis on rate of change toward moral relevance of a large repertoire of words against concept concreteness ratings, word frequency (a correlate of semantic change; see Hamilton et al., 2016), and word length (as a proxy for concept complexity; see Lewis and Frank, 2016).

We obtained norms of concreteness ratings from Warriner et al. (2013). We collected the same set of high-frequency nouns as in the previous analysis, along with their fitted slopes of moral relevance change. Since we were interested in moral relevance change within this large set of words, we restricted our analysis to those words whose model predictions indicate change in moral relevance, in either direction, from the 1800s to the 1990s.

We performed a multiple linear regression under the following model:

$$\tilde{\rho}(w) = \beta_f \log(f(w)) + \beta_l l(w) + \beta_c c(w) + \beta_0 + \tilde{\varepsilon} \quad (1)$$

Here $\rho(w)$ is the slope of moral relevance change for word $w$; $f(w)$ is its average frequency; $l(w)$ is its character length; $c(w)$ is its concreteness rating; $\beta_f$, $\beta_l$, $\beta_c$, and $\beta_0$ are the corresponding factor weights and intercept, respectively; and $\varepsilon \sim \mathcal{N}(0, \sigma)$ is the regression error term.

Table 6 shows the results of multiple linear regression. We observe that concreteness is a significant negative predictor of change toward moral relevance, suggesting that abstract concepts are more strongly associated with increasing moral relevance over time than concrete concepts. This

| Factor | Coeff. (e-03) | Significance |
|---|---|---|
| Frequency | 0.1 | $p < 0.001$ |
| Length | $-0.03$ | n.s. ($\alpha = 0.05$) |
| Concreteness | $-0.2$ | $p < 0.002$ |

Table 6: Results from multiple regression that regresses rate of change in moral relevance against the factors of word frequency, length, and concreteness ($n = 606$).

significance persists under partial correlation test against the control factors ($p < 0.01$).

We further verified the diachronic component of this effect in a random permutation analysis. We generated 1,000 control time courses by randomly shuffling the 20 decades in our data, and repeated the regression analysis to obtain a control distribution for each regression coefficient. All effects became non-significant under the shuffled condition, suggesting the relevance of concept concreteness for diachronic change in moral sentiment (see Supplementary Material).

## 7 Discussion and conclusion

We presented a text-based framework for exploring the socio-scientific problem of moral sentiment change. Our methodology uses minimal parameters and exploits implicit moral biases learned from diachronic word embeddings to reveal the public's moral perception toward a large concept repertoire over a long historical period.

Differing from existing work in NLP that treats moral sentiment as a flat classification problem (Garten et al., 2016; Johnson and Goldwasser, 2018), our framework probes moral sentiment change at multiple levels and captures moral dynamics concerning relevance, polarity, and fine-grained categories informed by Moral Foundations Theory (Graham et al., 2013). We applied our methodology to the automated analyses of moral change both in individual concepts and at a broad scale, thus providing insights into psycholinguistic variables that associate with rates of moral change in the public.

Our current work focuses on exploring moral sentiment change in English-speaking cultures. Future research should evaluate the appropriateness of the framework to probing moral change from a diverse range of cultures and linguistic backgrounds, and the extent to which moral sentiment change interacts and crisscrosses with linguistic meaning change and lexical coinage. Our

work creates opportunities for applying natural language processing toward characterizing moral sentiment change in society.

## Acknowledgments

We would like to thank Nina Wang, Nicola Lacerata, Dan Jurafsky, Paul Bloom, Dzmitry Bahdanau, and the Computational Linguistics Group at the University of Toronto for helpful discussion. We would also like to thank Ben Prystawski for his feedback on the manuscript. JX is supported by an NSERC USRA Fellowship and YX is funded through a SSHRC Insight Grant, an NSERC Discovery Grant, and a Connaught New Researcher Award.

## References

Mark Alfano, Dirk Hovy, Margaret Mitchell, and Michael Strube. 2018. *Proceedings of the Second ACL Workshop on Ethics in Natural Language Processing*. Association for Computational Linguistics.

Paul Bloom. 2010. How do morals change? *Nature*, 464(7288):490.

Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. 2016. Man is to computer programmer as woman is to homemaker? Debiasing word embeddings. In *Advances in neural information processing systems*, pages 4349–4357.

Aylin Caliskan, Joanna J Bryson, and Arvind Narayanan. 2017. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186.

Mark Davies. 2010. The Corpus of Historical American English: 400 million words, 1810–2009.

Susan T Fiske and Shelley E Taylor. 1991. *Social Cognition*. McGraw-Hill Book Company.

Nikhil Garg, Londa Schiebinger, Dan Jurafsky, and James Zou. 2018. Word embeddings quantify 100 years of gender and ethnic stereotypes. *Proceedings of the National Academy of Sciences*, 115(16):E3635–E3644.

Justin Garten, Reihane Boghrati, Joe Hoover, Kate M Johnson, and Morteza Dehghani. 2016. Morality between the lines: Detecting moral sentiment in text. In *Proceedings of IJCAI 2016 Workshop on Computational Modeling of Attitudes*.

Nabeel Gillani and Roger Levy. 2019. Simple dynamic word embeddings for mapping perceptions in the public sphere. *Computing Research Repository*, arXiv:1904.03352.

Jesse Graham, Jonathan Haidt, Sena Koleva, Matt Motyl, Ravi Iyer, Sean P Wojcik, and Peter H Ditto. 2013. Moral foundations theory: The pragmatic validity of moral pluralism. In *Advances in Experimental Social Psychology*, volume 47, pages 55–130.

Jesse Graham, Jonathan Haidt, and Brian A Nosek. 2009. Liberals and conservatives rely on different sets of moral foundations. *Journal of Personality and Social Psychology*, 96(5):1029.

Jonathan Haidt and Craig Joseph. 2004. Intuitive ethics: How innately prepared intuitions generate culturally variable virtues. *Daedalus*, 133(4):55–66.

Jonathan Haidt, Craig Joseph, et al. 2007. The moral mind: How five sets of innate intuitions guide the development of many culture-specific virtues, and perhaps even modules. *The Innate Mind*, 3:367–391.

William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Diachronic word embeddings reveal statistical laws of semantic change. In *Proceedings of the ACL*, pages 1489–1501. Association for Computational Linguistics.

Dirk Hovy, Shannon Spruit, Margaret Mitchell, Emily M. Bender, Michael Strube, and Hanna Wallach. 2017. *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*. Association for Computational Linguistics.

David Hume. 1739. *A Treatise of Human Nature*. Clarendon Press.

Kristen Johnson and Dan Goldwasser. 2018. Classification of moral foundations in microblog political discourse. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 720–730, Melbourne, Australia. Association for Computational Linguistics.

Immanuel Kant. 1785. *Groundwork for the Metaphysics of Morals*. Oxford University Press.

Christian Kay, Jane Roberts, Michael Samuels, Iren Wotherspoon, and Marc Alexander (eds.). 2019. *The Historical Thesaurus of English*. Glasgow: University of Glasgow. Version 4.21.

Lawrence Kohlberg. 1969. *Stage and Sequence: The Cognitive-developmental Approach to Socialization*. Rand McNally.

George Lakoff. 1996. *Moral politics: What conservatives know that liberals don't*. University of Chicago Press, Chicago.

Molly L Lewis and Michael C Frank. 2016. The length of words reflects their conceptual complexity. *Cognition*, 153:182–195.

Ying Lin, Joe Hoover, Gwenyth Portillo-Wightman, Christina Park, Morteza Dehghani, and Heng Ji. 2018. Acquiring background knowledge to improve

moral value prediction. In *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 552–559. IEEE.

Yuri Lin, Jean-Baptiste Michel, Erez Aiden Lieberman, Jon Orwant, Will Brockman, and Slav Petrov. 2012. Syntactic annotations for the Google books NGram corpus. In *Proceedings of the ACL 2012 System Demonstrations*, pages 169–174. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

George A Miller. 1995. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41.

Marlon Mooijman, Joe Hoover, Ying Lin, Heng Ji, and Morteza Dehghani. 2018. Moralization in social networks and the emergence of violence during protests. *Nature Human Behaviour*, 2(6):389.

John R Oldfield. 2012. *Popular Politics and British Anti-Slavery: The mobilisation of public opinion against the slave trade 1787-1807*. Routledge.

Pew Research Center. 2013. Global Views on Morality. Pew Research Center's Global Attitudes Project.

Jean Piaget. 1932. *The Moral Judgment of the Child*. London: Routledge and Kegan Paul.

David A. Pizarro and Paul Bloom. 2003. The intelligence of the moral intuitions: A comment on Haidt (2001). *Psychological Review*, 110(1):193–196.

Adam Smith. 1759. *The Theory of Moral Sentiments*. History of Economic Thought Books. McMaster University Archive for the History of Economic Thought.

James W Stigler, Richard A Schweder, and Gilbert Herdt. 1990. *Cultural Psychology: Essays on Comparative Human Development*. Cambridge University Press.

Harriet Beecher Stowe. 1852. *Uncle Tom's Cabin*. Tauchnitz.

Amy Beth Warriner, Victor Kuperman, and Marc Brysbaert. 2013. Norms of valence, arousal, and dominance for 13,915 English lemmas. *Behavior Research Methods*, 45(4):1191–1207.

# Detecting Causal Language Use in Science Findings

Bei Yu[1], Yingya Li[1], and Jun Wang[2]

[1]School of Information Studies, Syracuse University
[2]Independent Researcher
{byu, yli48}@syr.edu, junwang4@gmail.com

## Abstract

Causal interpretation of correlational findings from observational studies has been a major type of misinformation in science communication. Prior studies on identifying inappropriate use of causal language relied on manual content analysis, which is not scalable for examining a large volume of science publications. In this study, we first annotated a corpus of over 3,000 PubMed research conclusion sentences, then developed a BERT-based prediction model that classifies conclusion sentences into "no relationship", "correlational", "conditional causal", and "direct causal" categories, achieving an accuracy of 0.90 and a macro-F1 of 0.88. We then applied the prediction model to measure the causal language use in the research conclusions of about 38,000 observational studies in PubMed. The prediction result shows that 21.7% studies used direct causal language exclusively in their conclusions, and 32.4% used some direct causal language. We also found that the ratio of causal language use differs among authors from different countries, challenging the notion of a shared consensus on causal language use in the global science community. Our prediction model could also be used to help identify the inappropriate use of causal language in science publications.

## 1 Introduction

Establishing causality is one of the most important goals and concerns of science. Therefore, the language that describes causal relationships plays a crucial role in communicating science findings among scientists and with the general public (Kleinberg and Hripcsak, 2011). However, scientists and journalists have been found to inappropriately use causal language in research publications and news articles. Specifically, causal interpretation of correlational findings from observational studies has been a major type of misinformation in science communication (Cofield et al., 2010; Sumner et al., 2014; Chiu et al., 2017; Boutron and Ravaud, 2018). In scientific research, observational studies (non-intervention) are designed for testing association/correlation between variables, while intervention studies, such as clinical trials, are for testing causal relations (Buhse et al., 2018). Misinterpreting correlations as causations can lead to serious consequences, such as wrong medical decisions (Buhse et al., 2018) or harmful misperception of certain groups of people in society (Richardson et al., 2014).

The misinterpretations are often attributed to humans' inherent tendency to conflate correlation and causation (Bleske-Rechek et al., 2015), and journalists' lack of scientific training or their pursuit of sensational effect (Fahnestock, 1998; Sumner et al., 2014). However, misinterpreting correlational findings as causal claims has been found not only in news stories and press releases written by journalists (Sumner et al., 2014), but also in research papers published by scientists. For example, Robinson et al. (2007) manually examined the methodologies of educational studies and found an increasing number of nonintervention studies using causal statements, from 34% in 1994 to 43% in 2004. Cofield et al. (2010), also through manual content analysis, found 31% of observational studies in obesity and nutrition inappropriately used causal language.

Scientists' inappropriate use of causal language may also be attributed to the pursuit of exaggeration in research contribution, or the lack of training in science writing, especially for non-English speaking researchers. For example, a number of linguistic studies have found that expressing and interpreting epistemic modality is not an easy task for learners of English as a second language (Holmes, 1982). There is also a lack of consistent



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4664–4674, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

teaching of strategies for thinking about probability and causality (Stanovich, 2010). With English as the most popular language for scientific writing, non-native English-speaking authors' native languages may also affect their choice of causal expressions (Hyland and Milton, 1997; Hyland, 1998; McEnery and Kifle, 2002; Kranich, 2009), adding to the challenge of establishing a shared standard of causal language use in the global science community.

In this study, we developed a prediction model to automatically identify correlational and causal statements in research conclusions. We then applied this model to the observational studies in PubMed to answer three research questions regarding causal language use. First, what is the overall ratio of causal lanugage use in observational studies? Second, what is the trend of causal language use? Third, do authors from different countries and language backgrounds use causal language at different levels? This study contributes to the field of computational social science by providing a new prediction model to identify causal language use in research findings, and by providing new evidence from large-scale analysis for answering research questions regarding causal language use in global science communication.

## 2 Related Work

### 2.1 Challenge in Describing Causal Relations

Making valid causal inferences from observational data is a challenging and risky process (Hernán and Robins, 2006). Linguistic expressions of causal relations vary greatly (Dunietz et al., 2017). Commonly used causal markers can be domain- and genre-dependent (Mulkar-Mehta et al., 2011). Therefore, choosing the appropriate linguistic cues to describe different causal levels has been identified as one of the main difficulties that student writers experience in managing epistemic meaning (Holmes, 1982). Though such difficulty is reflected in a number of best-practice guides for science writers (Zweig and DeVoto, 2018), suggestions in those guides are generally based on the judgment of only a few individuals (Adams et al., 2017), and there lacks consistent teaching of strategies for thinking about probability and causality (Stanovich, 1992). As a result, authors of observational studies sometimes wrongly extrapolate their results for clinical practice recommendations rather than suggesting follow-up randomized controlled trials (RCTs) (Prasad et al., 2013). Inappropriate use of causal language was then observed by researchers in different science reporting venues, such as leading research journals (e.g. Cofield et al., 2010; Brown et al., 2013; Lazarus et al., 2015), and press releases (e.g. Yavchitz et al., 2012).

### 2.2 Definition of Causal Relations

Detecting causal language use is closely related to the work of cause-effect relation extraction. NLP tasks such as SemEval-2007 (Girju et al., 2007) and SemEval-2010 (Hendrickx et al., 2009) included extracting cause-effect relations as a subtask of semantic relation extraction. Overall, the cause-effect relation could broadly refer to relations between events and entities (Bethard and Martin, 2008; Mirza and Tonelli, 2014; Chambers et al., 2014), clauses (Grivaz, 2010), and discourse arguments (Prasad et al., 2008).

Contrarily, some studies specifically focused on the causal relations expressed between independent variables and dependent variables. They normally formulate the task as a binary classification problem to distinguish causation from correlation (e.g. Cofield et al., 2010). To capture the different levels of certainty in research findings, Sumner et al. (2014) proposed a more fine-grained categorization with seven levels: *no mentioned relationship*, *statement of no relationship*, *statement of correlation*, *ambiguous statement of relationship*, *conditional statement of causation*, *statement of can*, and *statement of causation*.

In comparison, some defined patterns by the broader definition are not applicable to the narrower definition. For example, relations such as "purpose" and "motivation" (Dunietz et al., 2015) between news events are hard to be generalized to describe relations between variables in research findings. Meanwhile, discourse markers such as reporting verbs (Mihăilă et al., 2013) and conjunctions like "because" are commonly used indicators for causal relations at clause or discourse levels, but are rarely used to describe relations between variables. In this study, we adopted the narrow definition and focused on the causal relations expressed between dependent variable and independent variables.

### 2.3 Corpora for Causal Relation Analysis

Texts from different domains have been used for causal relation analysis, such as news articles (e.g.

Khoo et al., 1998; Girju et al., 2002; Girju, 2003; Prasad et al., 2008; Sumner et al., 2014), health and biomedical publications (e.g. Cofield et al., 2010; Khoo et al., 2000; Mihăilă et al., 2013; Sumner et al., 2014; Sharma et al., 2018), expository texts (Kaplan and Berry-Rogghe, 1991), and technical texts (Garcia et al., 1997). Mulkar-Mehta et al. (2011) compared the causal markers from different domains, some of which are domain-specific. Dunietz et al. (2017) also found that biomedical texts and other specialized fields may have context-specific language constructs about causality. A review of these prior studies showed that to date there is no existing corpus particularly focusing on research findings in the health domain.

### 2.4 Methods for Identifying Causal Relations

Both manual and machine learning approaches have been used to extract causal relations. Manual approaches often applied domain-specific, knowledge-based inferences (e.g. Kaplan and Berry-Rogghe, 1991) and hand-coded rules to encode the causal language cues (e.g. Garcia et al., 1997; Cofield et al., 2010; Lazarus et al., 2015; Chiu et al., 2017). However, such manual approaches often require an exhaustive list of linguistic cues, which costs a significant amount of human effort and its generalizability may be limited by the small sample size.

Meanwhile, machine learning algorithms, such as Logistic Regression (Bui et al., 2010), Naive Bayes (Chang and Choi, 2004), Decision Trees (Blanco et al., 2008), Conditional Random Fields (CRF) (Mihăilă and Ananiadou, 2013) and SVM, have also been used to extract causal relations. SVM was among the most frequently applied algorithms (e.g. Sarker and Gonzalez, 2015; Mirza and Tonelli, 2016).

Most recently, deep learning approaches have also been used to extract causal relations. For example, Miwa and Bansal (2016) proposed a LSTM-RNN model incorporated with rich linguistic structures to extract relations in nominal relation extraction in SemEval-2010 (Hendrickx et al., 2009), which outperformed the CNN-based models. Dasgupta et al. (2018) proposed a bidirectional LSTM model to extract causal relations from drug effect data and news articles, with better performance than the rule-based approaches and CRF.

Despite the progress on machine learning approaches for causal relation detection, due to the different definition of causal relations and the dataset, experimental results may not be necessarily comparable to each other. Their generalizability remains an open question for tasks focusing on research conclusions. To this end, Li et al. (2017) adopted a simplified taxonomy from (Sumner et al., 2014) for causal relations in research findings, and used SVM to develop a four-category classifier with a 0.718 F1-score, suggesting room for further performance improvement. In this study we will investigate traditional and deep learning methods for building causal language prediction model with a new, human-annotated corpus dedicated to research conclusions.

## 3 Corpus Construction

To perform automatic detection of causal language use in research findings, a data set consisting of both causal and non-causal statements is needed. We chose PubMed as the data source to construct the corpus. We focused on health because it is a research topic of high social impact, and inappropriate use of causal language in health literature has been reported in prior manual studies. PubMed provides convenient access to a large number of health research publications with rich metadata particularly useful for this study.

In this study we chose to analyze the abstracts instead of full-text articles because the PubMed abstracts are openly available to the public while many full-text articles are behind a paywall. The research papers in PubMed may include an unstructured abstract, or a structured abstract which usually has a conclusion subsection to present the major findings. For the convenience of locating research findings, we used the structured abstracts only.

We selected a sample of structured abstracts based on a three-dimensional stratified sampling strategy. First, to account for the vocabulary variation among different health issues, five common health topics —nutrition, diabetes, obesity, breast cancer, and cholesterol— were selected. Second, because different study designs can affect the strength of the research findings, and thus affect the language choice for correlational or causal relations, we sampled both observational studies and randomized controlled trials (RCTs) using PubMed's metadata property "Publication Type".

For observational studies, we also used an advanced keyword search query[1] to further identify four different types of observational studies (case-control, cross-sectional, retrospective cohort, and prospective cohort). Third, the abstracts were selected based on the length of the conclusion subsection. The XML files from PubMed contain occasional parsing errors, resulting in extremely long conclusions which actually included paragraphs in the following sections. For quality control purpose, we used the Stanford CoreNLP tool (Manning et al., 2014) to split the conclusion subsections into sentences and removed the articles with conclusions longer than four sentences. Eventually, articles were sampled with the conclusion length as 1, 2, 3, or 4 sentences.

We then constructed a coding schema to define the relation categories for research findings. Drawing on prior studies of factuality (Kilicoglu et al., 2015) and causal strength (Sumner et al., 2014), we defined a causal level taxonomy as a simplified combination of the taxonomies in the two studies: *correlational*, *conditional causal*, *direct causal*, and *no relationship*. Table 1 lists the category definitions and some common language cues used to identify the relation type for each category. Table 2 shows the examples of sentences with different relation types. Sometimes a sentence may contain language cues indicating a causal relationship, such as "a reliable way to determine"; however, if the sentence describes the function of certain tools or diagnoses rather than the explicit relations between independent variable and dependent variables, it should be labelled as "no relationship" (as shown in Example 6 of Table 2). Meanwhile, sentences that discuss study limitations, as shown in Example 7 of Table 2, should be labelled as "no relationship".

A sample of 30 abstracts were randomly selected for the inter-coder reliability test. Specifically, two annotators labelled the relation type for each sentence from the 30 conclusion subsections, and also highlighted the linguistic indicators. The overall Cohen's Kappa agreement (Cohen, 1960) was 0.98, indicating a near-perfect inter-coder agreement (McHugh, 2012). Disagreements in the annotation were later resolved by the two annotators through discussion.

One of the annotators, who has linguistics background, then annotated the rest of the data. The annotator also marked unsure cases and brought to team discussion before finalizing the annotations. The corpus contains 3,126 annotated sentences, of which 3,061 sentences contained only one type of relations, and the remaining 65 sentences had more than one type of relations. Table 3 shows the relation type distribution of single-class sentences in the corpus.

## 4 Experiments and Evaluation

We trained and evaluated four machine learning approaches for classifying causal language use in conclusion sentences. The learning approaches are bag-of-words based Linear SVM, Bidirectional Recurrent Neural Network, BERT (Devlin et al., 2018), and BioBERT (Lee et al., 2019).

### 4.1 Experimental Setup

**Linear SVM.** We used the Scikit Learn vectorizer with unigrams and bigrams as feature set to convert texts to sparse vectors. Adding tri-grams did not improve model performance. Performance on *count* and *tf-idf* vectorizations were similar. The parameter C in LinearSVM was set to 0.1 after tuning.

**BiRNN.** Different from the traditional bag-of-words methods that convert a text to a sparse vector, BiRNN and BERT are deep learning approaches designed to transform a text to a dense vector that can better represent the meaning of the text. In our implementation, we used the combination of two word embeddings: the PMC embedding based on full-text research publications from PubMed Central (PMC) [2], and the universal GloVe embedding. We found that the PMC embedding did better than GloVe, and their combination performed the best. In addition, we used GRU recurrent network units, which are generally simpler and faster than LSTM units.

**BERT.** BERT is the latest method for pre-training language representations, which has achieved the state-of-the-art results on a wide array of NLP tasks, including sentence classification. We used the Pytorch version of BERT with its pre-trained model (Cased BERT-Base, which performs better than the Uncased one) [3]. Because the distribution of sentence relationships in our corpus is unbalanced, we revised the loss function in the original

[1] Public Health: Search Strategies by Study Type (University of Adelaide)

[2] http://bio.nlplab.org/

[3] https://github.com/huggingface/pytorch-pretrained-BERT

| Relation Type | Description | Language Cue |
|---|---|---|
| Correlational | The statement describes the association between variables, but causation cannot be explicitly stated. | association, associated with, predictor, at high risk of |
| Conditional Causal | The statement shows that one variable directly changes the other. However, the relation carries an element of doubt in it, which is normally via hedges or modalities. | increase, decrease, lead to, effect on, contribute to, result in<br>*Cues indicating doubt:*<br>may, might, appear to, probably |
| Direct Causal | The statement says that the independent variable directly alters the dependent variable. | increase, decrease, lead to, effective in, contribute to, reduce |
| No Relationship | No correlation or causation relationship is mentioned in the statement. | |

Table 1: A taxonomy of relation types and examples of commonly used language cues.

| Annotation | Example Sentence |
|---|---|
| Correlational | 1 The findings from this large prospective study show that men who are taller and who have greater adiposity have an elevated risk of high-grade prostate cancer and prostate cancer death.<br>2 The association of high PCT and CRP was no more predictive of mortality than high CRP. |
| Conditional Causal | 3 MTHFR A1298C polymorphism might contribute to an increased risk of breast cancer and/or ovarian cancer susceptibility. |
| Direct Causal | 4 Participatory community-based nutrition education for caregivers improved child dietary diversity even in a food insecure area.<br>5 The adoption of a comprehensive preoperative multidisciplinary approach led to significant improvements in the postoperative outcomes and also in the compliance to the postoperative follow-up. |
| No Relationship | 6 A quick foot-of-the-bed clinical assessment is not a reliable way to determine frailty.<br>7 This approach may, however, be difficult to implement on a large scale. |

Table 2: Examples of sentences and annotated relation types.

| Label | Count | Ratio |
|---|---|---|
| Correlational | 998 | 32.6% |
| Conditional Causal | 213 | 7.0% |
| Direct Causal | 494 | 16.1% |
| No Relationship | 1356 | 44.3% |
| Total | 3061 | 100.0% |

Table 3: Distribution of sentence types in the corpus.

code to ensure that the penalty on misclassifying rare classes was high.

**BioBERT.** BioBERT is a domain specific language representation model that is based on BERT and further pre-trained on large-scale biomedical corpora. BioBERT has shown to outperform BERT on three representative biomedical text mining tasks (Lee et al., 2019). In our experiment, we used the same BERT code as described above, and replaced the Cased BERT-Base pre-trained model with the BioBERT model.

### 4.2 Evaluation

To compare the performance of the above four learning approaches, we evaluated them on our annotated corpus via 5-fold stratified cross-validation. The result in Table 4 shows that BioBERT performed the best by all measures, followed by BERT, BiRNN and then LinearSVM. BioBERT achieved 0.881 macro-averaged F1 score with fairly balanced performance on each category except the conditional causal one, which only covers 7% data in the training set. See the performance scores per category in Table 5 and the confusion matrix in Fig. 1.

| | Precision | Recall | Accuracy | F1 |
|---|---|---|---|---|
| LinearSVM | 0.739 | 0.711 | 0.772 | 0.722 |
| BiRNN | 0.801 | 0.825 | 0.836 | 0.811 |
| BERT | 0.867 | 0.883 | 0.889 | 0.874 |
| BioBERT | 0.878 | 0.886 | 0.901 | 0.881 |

Table 4: Model performance: BioBERT$>$ BERT $>$ BiRNN $>$ LinearSVM. (F1 represents macro-F1.)

### 4.3 Error Analysis

Error analysis of misclassified cases shows the most common disagreement between machine prediction and human annotation is between the categories of *direct causal* and *no relationship*, even though the two categories are not semanti-

| | Precision | Recall | F1 |
|---|---|---|---|
| Correlational | 0.917 | 0.924 | 0.921 |
| Conditional Causal | 0.791 | 0.854 | 0.822 |
| Direct Causal | 0.878 | 0.858 | 0.868 |
| No Relationship | 0.915 | 0.906 | 0.911 |

Table 5: Detailed performance of BioBERT on each category of sentence relationship.

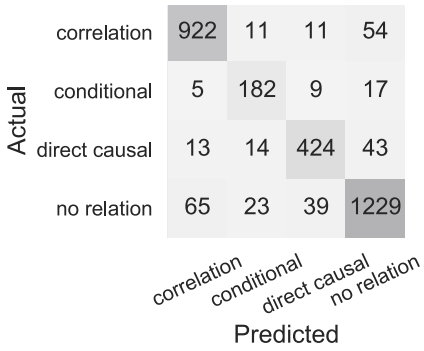


Figure 1: Confusion matrix. The prediction results on each test fold from 5-fold cross-validation were assembled together.

cally close in our dataset. This is largely caused by the confounding language cues in the *no relationship* examples. Specifically, a *no relationship* sentence that describes implications for future studies or functions of certain assessments may contain causal markers, such as "is critical to achieving", and "improved" in the following Examples 1 and 2, and thus mislead the prediction model. In addition, causal markers sometimes appeared in subordinate clauses instead of in the main clauses, which could also confuse the prediction model (see "the impact of" in Example 3). The prediction model also has some difficultly in recognizing the less commonly used causal markers, especially when nouns were used to describe causal relations, such as "cause" and "the key role of" in Examples 4 and 5.

*Sentence examples:*

1. However, training and experience of nurses in aromatherapy massage is critical to achieving positive results.
2. These findings pose the question: why has not the nutritional status of children improved, although the living conditions of their families have significantly improved?
3. Some studies that have assessed the impact of monitoring guidelines on clinical practice show only limited impact.
4. However, body mass index greater than 25 and smoking history are cause for caution.
5. Findings reinforce the key role of mvpa for childrens health.

### 4.4 Learning Curve

Our training dataset contains 3,061 sentences. To test whether the dataset size is sufficient to build a strong prediction model, we ran a learning curve test by setting aside 20% of the dataset for testing (i.e., about 600 examples), and then trained the BioBERT model with the 10%, 20%, ..., 100% of the remaining data (i.e., about 2500). We ran the experiment 5 times. As shown in Fig. 2, the average performance levels off around 60% of training data (i.e., about 1500), indicating that the training dataset is sufficient, although a larger dataset might help build a stronger model.




Figure 2: Learning curve. The average performance levels off around 60% of the training data, which is about 1500 examples.

## 5 Application to Observational Studies

In this section, we aim to investigate the causal language use in observational studies at a large scale. In 2014, the PubMed staff introduced *observational study* to their list of publication types[4]. To date 61,830 PubMed publications have been manually assigned to the observational study category. However, some complicated studies were assigned to multiple study design categories, such as both observational study and randomized controlled trial. After removing those with multiple study design types, we obtained 51,274 publications. Among them, 38,191 studies have structured abstracts with a conclusion subsection. For the reason mentioned in section 3, we do not consider those publications with more than 4 sentences in their conclusion subsections, since those with more than 4 sentences are often the result

[4]https://www.ncbi.nlm.nih.gov/mesh/68064888

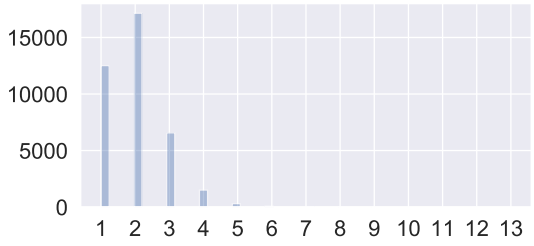


Figure 3: Distribution of number of sentences in the abstract conclusion subsections of the 38,191 observational studies. 98.8% of the publications have 4 or less sentences.

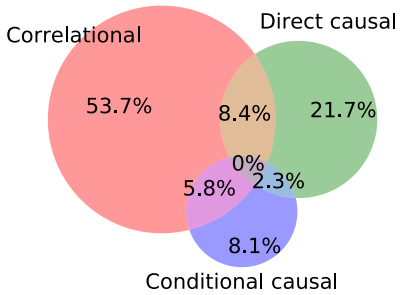


Figure 4: The constitution of the three relationships.

of PubMed metadata concatenating the conclusion subsection with the following paragraphs. Among the 38,191 publications, 98.8% have 4 or less sentences. (Fig. 3 shows the distribution of number of sentences included in the conclusion subsections.) The final data set of observational studies includes 37,746 publications, which in total contains 72,565 conclusion sentences.

We then applied the sentence classification model, as described in the previous section, to these conclusion sentences. The prediction results from the model are used to answer three research questions regarding causal language use in observational studies.

### 5.1 Results and Discussion

We aim to investigate the following questions:

**RQ1.** What is the overall ratio of causal lanugage use in observational studies?

The prediction result shows that among the 37,746 observational studies, 22% did not mention either correlation or causal relations. Because we would like to focus on distinguishing correlation vs. causal statements, the publications without mentioning relations were excluded, and the remaining 78% or 29,410 studies were used in further analyses.

Since PubMed also includes a small number of non-English publications with an English abstract, we used the PubMed metadata "language" to separate the 29,410 studies into the English subset (28,217 publications) and the non-English subset (1,193 publications).

The predictions were made on individual sentences. A conclusion subsection may contain multiple sentences; only 33% of the publications consist of one conclusion sentence.

Using the result from the 28,217 English publications, a Venn diagram (Fig. 4) was generated to show the proportion of studies mentioning one or more types of relations. It shows that 83.5% of studies contained only one type of relation in the conclusion: 53.7% correlational, 8.1% conditional causal, and 21.7% direct causal. Most of these studies contain only one sentence in the conclusion subsection. A manual examination of examples in the mixed categories found two main types of cases. The first type usually includes findings for multiple research questions. The second type uses multiple sentences to describe one finding; for example, the first sentence describes a correlation, and the second generalizes the correlation to a conditional or direct causal relation.

In this study we focus on the inappropriate use of direct causal language in observational studies. Therefore, we calculated the percentage of studies that used direct causal language in at least one finding sentence, and the percentage of studies that exclusively used direct causal language in their findings. Our result shows that 21.7% studies used direct causal exclusively in their conclusions, and 32.4% used some direct causal language. This result is close to the manual estimation reported in (Cofield et al., 2010), which is 31%.

**RQ2.** What is the trend of causal language use?

From now on, in the following figures and tables, for simplicity, we use word *causal* for the studies that used some direct causal language, and phrase *causal only* for those that used direct causal exclusively. Fig. 5 illustrates the ratios of causal language use and causal only from year 2013 to 2019. The two ratios follow the same pattern. Different from (Robinson et al., 2007), no obvious increase of causal language use was observed in our study.

One limitation of our study is that the PubMed labels of observational studies were introduced in 2014 (although some 2013 publications were an-

notated too); therefore, our dataset covers only seven years with partial data from 2013 and 2019. In the future we will design a classifier to automatically label observational studies, and then will be able to analyze the trend with more longitudinal data.

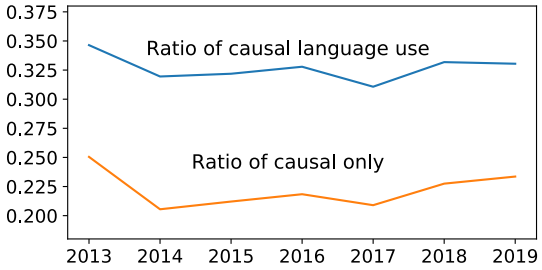


Figure 5: Causal language use over recent years.

**RQ3.** Do authors from different countries and language backgrounds use causal language at different levels? Prior studies in linguistics, as described in sections 1 and 2, have found that German, French, and Dutch translations tend to use stronger claims than the original English science publications. In this study, we will be able to compare causal language use in English and non-English publications, all with English abstracts. We hypothesize that if authors' native languages did affect their causal language use in English, the effect would be more pronounced in the English abstracts of the non-English publications, because these authors mainly used their native languages for science communication.

| Language | Papers | Causal | Causal only |
|---|---|---|---|
| German | 34 | 0.706 | 0.500 |
| French | 145 | 0.490 | 0.393 |
| Chinese | 80 | 0.487 | 0.388 |
| Italian | 32 | 0.469 | 0.375 |
| Spanish | 715 | 0.394 | 0.297 |
| English | 28217 | 0.323 | 0.217 |
| Portugese | 124 | 0.266 | 0.194 |

Table 6: Ratios of causal language use in studies published in English and other languages (sorted by *causal only*, reversely). Languages with fewer than 30 publications were excluded.

Table 6 shows the ratios of causal language use in the non-English subset. The result shows that publications in some languages, such as German, used much more causal language than those in English and Portugese.

| Country | Papers | Causal | Causal only |
|---|---|---|---|
| Germany | 909 | 0.421 | 0.295 |
| Italy | 1314 | 0.390 | 0.287 |
| France | 992 | 0.342 | 0.247 |
| Spain | 1621 | 0.365 | 0.239 |
| UK | 1071 | 0.331 | 0.222 |
| Brazil | 750 | 0.316 | 0.219 |
| Japan | 1299 | 0.296 | 0.216 |
| China | 919 | 0.301 | 0.215 |
| Netherlands | 797 | 0.359 | 0.213 |
| Korea | 772 | 0.284 | 0.201 |
| US | 4771 | 0.277 | 0.174 |

Table 7: Ratios of causal language use in different countries (sorted by *causal only*, reversely). Countries with fewer than 750 publications were excluded.

Table 7 shows the ratios in the English subset, broken down by the authors' countries. For each publication, each author's country was extracted from the PubMed affiliation metadata. Publications with authors from more than one country were excluded. The result shows that Germany has the highest ratio of causal language use (42.1%). In comparison, the US has the lowest ratio.

Because some languages are used in multiple countries and some countries did not publish many papers, not all results in Table 6 and Table 7 are comparable. However, we were able to identify four country-language pairs that correspond well to each other: Germany-German, Italy-Italian, France-French, and China-Chinese. In the case of other non-English languages, Portugese is majorly used by Brazil and Portugal, and Spanish is majorly used in Spain and Mexico. In the above four pairs, the ratios of causal language use in the non-English subsets were all higher than those in the English subset.

These results provide supportive evidence to the claim that causal language use varies across countries and languages, and thus challenge the notion of a shared consensus on causal language use in global science community.

## 6 Conclusion and Future Work

In this study we developed a manually-annotated corpus for causal statements in science publications, and used the corpus to train a prediction model to identify correlational, conditional causal, and direct causal statements in research conclu-

sions, achieving an accuracy of 0.90 and a macro-F1 score of 0.88. This prediction model provides an automated approach for identifying potential misuse of causal language in science writing. When applying the model to investigating association between causal language use and authors' countries or native languages, the result lends support to the claim that causal language use varies across countries and languages, and thus challenges the notion of a shared consensus of causal language use in the global science community. The result is also important for raising awareness towards recognizing disparities in causal language use in science communication.

In the future we will further our investigation by developing prediction models to automatically identify study designs in more fine-grained categories, such as cross-sectional, case-control, retrospective cohort, and prospective cohort. We will then be able to accurately identify inappropriate causal language use in weaker study design types, and also to compare among countries and languages in each specific study design types.

We also plan to extend our work to other domains in the future. In our current work we chose the biomedical domain as the starting point not only for the problem significance but also for the high-quality and rich metadata provided by PubMed that can help identify study types. By contrast, no such data are available in other domains yet. Toward this new goal, we also plan to develop a classification method to identify study types in other domains such as psychology and education.

The corpus and the code are available at https://github.com/junwang4/causal-language-use-in-science.

## Acknowledgments

This work was partially supported by Syracuse University CUSE Grant Program and NVIDIA GPU Grant Program. We thank the reviewers for helpful comments, and Albert Wang for help with annotation.

## References

Rachel C Adams, Petroc Sumner, Solveiga Vivian-Griffiths, Amy Barrington, Andrew Williams, Jacky Boivin, Christopher D Chambers, and Lewis Bott. 2017. How readers understand causal and correlational expressions used in news headlines. *Journal of experimental psychology: applied*, 23(1):1.

Steven Bethard and James H Martin. 2008. Learning semantic links from a corpus of parallel temporal and causal relations. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, pages 177–180. Association for Computational Linguistics.

Eduardo Blanco, Nuria Castell, and Dan I Moldovan. 2008. Causal relation extraction. In *Lrec*.

April Bleske-Rechek, Katelyn M Morrison, and Luke D Heidtke. 2015. Causal inference from descriptions of experimental and non-experimental research: Public understanding of correlation-versus-causation. *The Journal of general psychology*, 142(1):48–70.

Isabelle Boutron and Philippe Ravaud. 2018. Misrepresentation and distortion of research in biomedical literature. *Proceedings of the National Academy of Sciences of the United States of America*, 115(11):2613–2619.

Andrew W Brown, Michelle M Bohan Brown, and David B Allison. 2013. Belief beyond the evidence: using the proposed effect of breakfast on obesity to show 2 practices that distort scientific evidence. *The American journal of clinical nutrition*, 98(5):1298–1308.

Susanne Buhse, Anne Christin Rahn, Merle Bock, and Ingrid Mühlhauser. 2018. Causal interpretation of correlational studies–analysis of medical news on the website of the official journal for german physicians. *PloS one*, 13(5):e0196833.

Quoc-Chinh Bui, Breanndán Ó Nualláin, Charles A Boucher, and Peter MA Sloot. 2010. Extracting causal relations on hiv drug resistance from literature. *BMC bioinformatics*, 11(1):101.

Nathanael Chambers, Taylor Cassidy, Bill McDowell, and Steven Bethard. 2014. Dense event ordering with a multi-pass architecture. *Transactions of the Association for Computational Linguistics*, 2:273–284.

Du-Seong Chang and Key-Sun Choi. 2004. Causal relation extraction using cue phrase and lexical pair probabilities. In *International Conference on Natural Language Processing*, pages 61–70. Springer.

Kellia Chiu, Quinn Grundy, and Lisa Bero. 2017. spinin published biomedical literature: A methodological systematic review. *PLoS biology*, 15(9):e2002173.

Stacey S Cofield, Rachel V Corona, and David B Allison. 2010. Use of causal language in observational studies of obesity and nutrition. *Obesity facts*, 3(6):353–356.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46.

Tirthankar Dasgupta, Rupsa Saha, Lipika Dey, and Abir Naskar. 2018. Automatic extraction of causal relations from text using linguistically informed deep neural networks. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pages 306–316.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jesse Dunietz, Lori Levin, and Jaime Carbonell. 2015. Annotating causal language using corpus lexicography of constructions. In *Proceedings of The 9th Linguistic Annotation Workshop*, pages 188–196.

Jesse Dunietz, Lori Levin, and Jaime Carbonell. 2017. Automatically tagging constructions of causation and their slot-fillers. *Transactions of the Association for Computational Linguistics*, 5:117–133.

Jeanne Fahnestock. 1998. Accommodating science: The rhetorical life of scientific facts. *Written communication*, 15(3):330–350.

Daniela Garcia et al. 1997. Coatis, an nlp system to locate expressions of actions connected by causality links. In *International Conference on Knowledge Engineering and Knowledge Management*, pages 347–352. Springer.

Roxana Girju. 2003. Automatic detection of causal relations for question answering. In *Proceedings of the ACL 2003 workshop on Multilingual summarization and question answering-Volume 12*, pages 76–83. Association for Computational Linguistics.

Roxana Girju, Dan I Moldovan, et al. 2002. Text mining for causal relations. In *FLAIRS conference*, pages 360–364.

Roxana Girju, Preslav Nakov, Vivi Nastase, Stan Szpakowicz, Peter Turney, and Deniz Yuret. 2007. Semeval-2007 task 04: Classification of semantic relations between nominals. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 13–18. Association for Computational Linguistics.

Cécile Grivaz. 2010. Human judgements on causation in french texts. In *LREC*.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2009. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, pages 94–99. Association for Computational Linguistics.

Miguel A Hernán and James M Robins. 2006. Instruments for causal inference: an epidemiologist's dream? *Epidemiology*, pages 360–372.

Janet Holmes. 1982. Expressing doubt and certainty in english. *RELC journal*, 13(2):9–28.

Ken Hyland. 1998. *Hedging in scientific research articles*, volume 54. John Benjamins Publishing.

Ken Hyland and John Milton. 1997. Qualification and certainty in l1 and l2 students' writing. *Journal of second language writing*, 6(2):183–205.

Randy M Kaplan and Genevieve Berry-Rogghe. 1991. Knowledge-based acquisition of causal relationships in text. *Knowledge Acquisition*, 3(3):317–337.

Christopher SG Khoo, Syin Chan, and Yun Niu. 2000. Extracting causal knowledge from a medical database using graphical patterns. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 336–343. Association for Computational Linguistics.

Christopher SG Khoo, Jaklin Kornfilt, Robert N Oddy, and Sung Hyon Myaeng. 1998. Automatic extraction of cause-effect information from newspaper text without knowledge-based inferencing. *Literary and Linguistic Computing*, 13(4):177–186.

Halil Kilicoglu, Graciela Rosemblat, Michael Cairelli, and Thomas Rindflesch. 2015. A compositional interpretation of biomedical event factuality. In *Proceedings of the Second Workshop on Extra-Propositional Aspects of Meaning in Computational Semantics (ExProM 2015)*, pages 22–31.

Samantha Kleinberg and George Hripcsak. 2011. A review of causal inference for biomedical informatics. *Journal of biomedical informatics*, 44(6):1102–1112.

Svenja Kranich. 2009. Epistemic modality in english popular scientific texts and their german translations. *Trans-kom*, 2(1):26–41.

Clément Lazarus, Romana Haneef, Philippe Ravaud, and Isabelle Boutron. 2015. Classification and prevalence of spin in abstracts of non-randomized studies evaluating an intervention. *BMC medical research methodology*, 15(1):85.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. Biobert: pre-trained biomedical language representation model for biomedical text mining. *arXiv preprint arXiv:1901.08746*.

Yingya Li, Jieke Zhang, and Bei Yu. 2017. An NLP Analysis of Exaggerated Claims in Science News. In *Proceedings of the 2017 EMNLP Workshop: Natural Language Processing meets Journalism*, pages 106–111.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

Tony McEnery and Nazareth Amselom Kifle. 2002. Epistemic modality in argumentative essays of second-language writers. *Academic discourse*, pages 182–195.

Mary L McHugh. 2012. Interrater reliability: the kappa statistic. *Biochemia medica: Biochemia medica*, 22(3):276–282.

Claudiu Mihăilă and Sophia Ananiadou. 2013. Recognising discourse causality triggers in the biomedical domain. *Journal of bioinformatics and computational biology*, 11(06):1343008.

Claudiu Mihăilă, Tomoko Ohta, Sampo Pyysalo, and Sophia Ananiadou. 2013. Biocause: Annotating and analysing causality in the biomedical domain. *BMC bioinformatics*, 14(1):2.

Paramita Mirza and Sara Tonelli. 2014. An analysis of causality between events and its relation to temporal information. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2097–2106.

Paramita Mirza and Sara Tonelli. 2016. Catena: Causal and temporal relation extraction from natural language texts. In *The 26th international conference on computational linguistics*, pages 64–75. ACL.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. *arXiv preprint arXiv:1601.00770*.

Rutu Mulkar-Mehta, Andrew S Gordon, Jerry R Hobbs, and Eduard Hovy. 2011. Causal markers across domains and genres of discourse. In *Proceedings of the sixth international conference on Knowledge capture*, pages 183–184. ACM.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind K Joshi, and Bonnie L Webber. 2008. The penn discourse treebank 2.0. In *LREC*. Citeseer.

Vinay Prasad, Joel Jorgenson, John PA Ioannidis, and Adam Cifu. 2013. Observational studies often make clinical practice recommendations: an empirical evaluation of authors' attitudes. *Journal of clinical epidemiology*, 66(4):361–366.

Sarah S. Richardson, Cynthia R. Daniels, Matthew W Gillman, Janet Lynne Golden, Rebecca Kukla, Christopher W Kuzawa, and Janet E Rich-Edwards. 2014. Society: Don't blame the mothers. *Nature*, 512:131–132.

Daniel H Robinson, Joel R Levin, Greg D Thomas, Keenan A Pituch, and Sharon Vaughn. 2007. The incidence of causal statements in teaching-and-learning research journals. *American Educational Research Journal*, 44(2):400–413.

Abeed Sarker and Graciela Gonzalez. 2015. Portable automatic text classification for adverse drug reaction detection via multi-corpus training. *Journal of biomedical informatics*, 53:196–207.

Raksha Sharma, Girish Palshikar, and Sachin Pawar. 2018. An unsupervised approach for cause-effect relation extraction from biomedical text. In *International Conference on Applications of Natural Language to Information Systems*, pages 419–427. Springer.

Keith E Stanovich. 1992. *How to think straight about psychology*. HarperCollins Publishers.

Keith E Stanovich. 2010. How to think straight about psychology, 9 th.

Petroc Sumner, Solveiga Vivian-Griffiths, Jacky Boivin, Andy Williams, Christos A Venetis, Aimée Davies, Jack Ogden, Leanne Whelan, Bethan Hughes, Bethan Dalton, Fred Boy Boy, and Christopher D Chambers. 2014. The association between exaggeration in health related science news and academic press releases: retrospective observational study. *BMJ*, 349:g7015.

Amélie Yavchitz, Isabelle Boutron, Aida Bafeta, Ibrahim Marroun, Pierre Charles, Jean Mantz, and Philippe Ravaud. 2012. Misrepresentation of randomized controlled trials in press releases and news coverage: a cohort study. *PLoS medicine*, 9(9):e1001308.

Mark Zweig and Emily DeVoto. 2018. Observational studies: Does the language fit the evidence? association vs. causation. http://www.healthnewsreview.org/toolkit/tips-for-understanding-studies/. Accessed on 01/02/2018.

# Multilingual and Multi-Aspect Hate Speech Analysis

**Nedjma Ousidhoum, Zizheng Lin, Hongming Zhang, Yangqiu Song, Dit-Yan Yeung**
Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
nousidhoum@cse.ust.hk, zlinai@connect.ust.hk,
hzhangal@cse.ust.hk, yqsong@cse.ust.hk,
dyyeung@cse.ust.hk

## Abstract

Current research on hate speech analysis is typically oriented towards monolingual and single classification tasks. In this paper, we present a new multilingual multi-aspect hate speech analysis dataset and use it to test the current state-of-the-art multilingual multitask learning approaches. We evaluate our dataset in various classification settings, then we discuss how to leverage our annotations in order to improve hate speech detection and classification in general.

## 1 Introduction

With the expanding amount of text data generated on different social media platforms, current filters are insufficient to prevent the spread of hate speech. Most internet users involved in a study conducted by the Pew Research Center report having been subjected to offensive name calling online or witnessed someone being physically threatened or harassed online.[1] Additionally, Amnesty International within Element AI have lately reported that many women politicians and journalists are assaulted every 30 seconds on Twitter.[2] This is despite the Twitter policy condemning the promotion of violence against people on the basis of race, ethnicity, national origin, sexual orientation, gender identity, religious affiliation, age, disability, or serious disease.[3] Hate speech may not represent the general opinion, yet it promotes the dehumanization of people who are typically from minority groups (Soral et al., 2017; Martin et al., 2012) and can incite hate crime (Ross et al., 2017).

Moreover, although people of various linguistic backgrounds are exposed to hate speech (Waseem et al., 2017; Ross et al., 2017), English is still at the center of existing work on toxic language analysis. Recently, some research studies have been conducted on languages such as German (Kratzke, 2017), Arabic (Albadi et al., 2018), and Italian (Sanguinetti et al., 2018). However, such studies usually use monolingual corpora and do not contrast, or examine the correlations between online hate speech in different languages. On the other hand, tasks involving more than one language such as the hatEval task[4], which covers English and Spanish, include only separate classification tasks, namely (a) women and immigrants as target groups, (b) individual or generic hate and, (c) aggressive or non-aggressive hate speech.

Treating hate speech classification as a binary task may not be enough to inspect the motivation and the behavior of the users promoting it and, how people would react to it. For instance, the hateful tweets presented in Figure 1 show toxicity directed towards different targets, with or without using slurs, and generating several types of reactions. We believe that, in order to balance between truth and subjectivity, there are at least five important aspects in hate speech analysis. Hence, our annotations indicate (a) whether the text is direct or indirect; (b) if it is offensive, disrespectful, hateful, fearful out of ignorance, abusive, or normal; (c) the attribute based on which it discriminates against an individual or a group of people; (d) the name of this group; and (e) how the annotators feel about its content within a range of negative to neutral sentiments. To the best of our knowledge there are no other hate speech datasets that attempt to capture fear out of ignorance in hateful tweets or examine how people react to hate speech. We claim that our multi-aspect annotation schema would provide a valuable insight into sev-

[1] http://www.pewinternet.org/2017/07/11/online-harassment-2017/

[2] https://www.amnesty.org.uk/press-releases/women-abused-twitter-every-30-seconds-new-study

[3] https://help.twitter.com/en/rules-and-policies/hateful-conduct-policy

[4] https://competitions.codalab.org/competitions/19935



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4675–4684,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

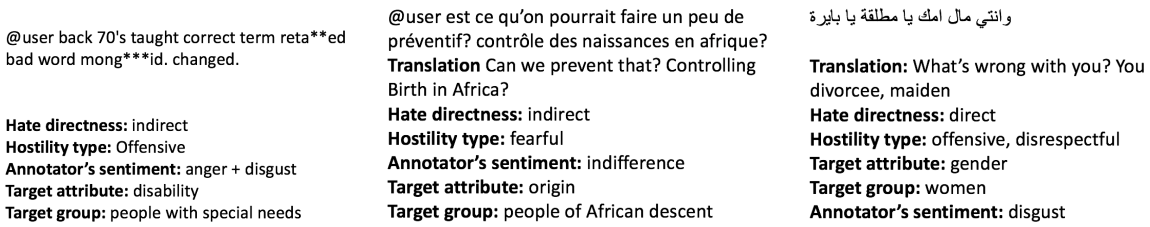


(a) English. (b) French. (c) Arabic.

Figure 1: Annotation examples in our dataset.

eral linguistic and cultural differences and bias in hate speech.

We use Amazon Mechanical Turk to label around 13,000 potentially derogatory tweets in English, French, and Arabic based on the above mentioned aspects and, regard each aspect as a prediction task. Since in natural language processing, there is a peculiar interest in multitask learning, where different tasks can be used to help each other (Collobert et al., 2011; Ruder et al., 2017; Hashimoto et al., 2017), we use a unified model to handle the annotated data in all three languages and five tasks. We adopt (Ruder et al., 2017) as a learning algorithm adapted to loosely related tasks such as our five annotated aspects and, use the Babylon cross-lingual embeddings (Smith et al., 2017) to align the three languages. We compare the multilingual multitask learning settings with monolingual multitask, multilingual single-task, and monolingual single-task learning settings respectively. Then, we report the performance results of the different settings and discuss how each task affects the remaining ones. We release our dataset and code to the community to extend research work on multilingual hate speech detection and classification.[5]

## 2 Related Work

There is little consensus on the difference between profanity and hate speech and, how to define the latter (Schmidt and Wiegand, 2017). As shown in Figure 2, slurs are not an unequivocal indicator of hate speech and can be part of a non-aggressive conversation, while some of the most offensive comments may come in the form of subtle metaphors or sarcasm (Malmasi and Zampieri, 2018). Consequently, there is no existing human annotated vocabulary that explicitly reveals the presence of hate speech, which makes the available hate speech corpora sparse and noisy (Nobata et al., 2016).

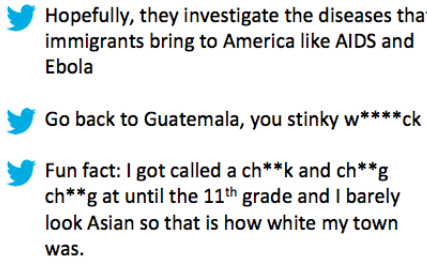


Figure 2: Three tweets in which (1) the first one accuses immigrants of harming society without using any direct insult; (2) the second insults a Hispanic person using a slur; and (3) the third one uses slurs to give a personal account. This shows that profanity is not a clear indicator of the presence of hate speech.

Given the subjectivity and the complexity of such data, annotation schemes have rarely been made fine-grained. Table 1 compares different labelsets that exist in the literature. For instance, Waseem and Hovy (2016) use racist, sexist, and normal as labels; Davidson et al. (2017) label their data as hateful, offensive (but not hateful), and neither, while ElSherief et al. (2018) present an English dataset that records the target category based on which hate speech discriminates against people, such as ethnicity, gender, or sexual orientation and ask human annotators to classify the tweets as hate and non hate. Founta et al. (2018) label their data as offensive, abusive, hateful, aggressive, cyberbullying, spam, and normal. On the other hand, Qian et al. (2018) have chosen to detect ideologies of hate speech counting 40 different hate ideologies among 13 extremist hate groups.

The detection of hate speech targets is yet another challenging aspect of the annotation. Park et al. (2018) report the bias that exists in the current datasets towards identity words, such as *women*, which may later cause false predictions. They propose to debias gender identity word em-

[5] our code is available on: https://github.com/HKUST-KnowComp/MLMA_hate_speech

| Dataset | # Tweets | Labels | Annotators/Tweet |
|---|---|---|---|
| Chatzakou et al. (2017) | 9,484 | aggressive, bullying, spam, normal | 5 |
| Waseem and Hovy (2016) | 16, 914 | racist, sexist, normal | 1 |
| Davidson et al. (2017) | 24, 802 | hateful, offensive (but not hateful), neither | 3 or more |
| Golbeck et al. (2017) | 35,000 | the worst, threats, hate speech, direct harassment, potentially offensive, non-harassment | 2 to 3 |
| Founta et al. (2018) | 80, 000 | offensive, abusive, hateful speech, aggressive, cyberbullying, spam, normal | 5 to 20 |
| ElSherief et al. (2018) | 28,608 | directed, generalized + target = archaic, class, disability, ethnicity, gender, nationality, religion, sexual orientation | 3 |
| Ours | 13,000 | Labels for five different aspects | 5 |

Table 1: Comparative table of some of the available hate speech and abusive language corpora in terms of labels and sizes.

beddings with additional data for training and tuning their binary classifier. We address this false positive bias problem and the common ambiguity of target detection by asking the annotators to label target attributes such as origin, gender, or religious affiliation within 16 named target groups such as refugees, or immigrants.

Furthermore, Klubička and Fernandez (2018) have reproduced the experiment of Waseem and Hovy (2016) in order to study how hate speech affects the popularity of a tweet, but discovered that some tweets have been deleted. For replication purposes, we provide the community with anonymized[6] tweet texts rather than IDs.

Non-English hate speech datasets include Italian, German, Dutch, and Arabic corpora. Sanguinetti et al. (2018) present a dataset of Italian tweets, in which the annotations capture the degree of intensity of offensive and aggressive tweets, in addition to whether the tweets are ironic and contain stereotypes or not. Ross et al. (2017) have collected more than 500 German tweets against refugees, and annotated them as hateful and not hateful. Hee et al. (2015) detect bullies and victims among youngsters in Dutch comments on AskFM, and classify cyberbullying comments as insults or threats. Moreover, Albadi et al. (2018) provide a corpus of Arabic sectarian speech.

Another predominant phenomenon in hate speech corpora is code switching. Bohra et al. (2018) present a dataset of code mixed Hindi-English tweets, while Galery et al. (2018) report the presence of Hindi tokens in English data and use multilingual word embeddings to deal with this issue when detecting toxicity. Similarly, we use such embeddings to take advantage of the multilinguality and comparability of our corpora during the classification.

Our dataset is the first trilingual dataset comprising English, French, and Arabic tweets that encompasses various targets and hostility types. Additionally, to the best of our knowledge, this is the first work that examines how annotators react to hate speech comments.

To fully exploit the collected annotations, we tested multitask learning on our dataset. Multitask learning (Collobert et al., 2011) allows neural networks to share parameters with one another and, thus, learn from related tasks. It has been used in different NLP tasks such as parsing (Hashimoto et al., 2017), dependency parsing (Peng et al., 2017), neural machine translation (Luong et al., 2016), sentiment analysis (Augenstein et al., 2018), and other tasks. Multitask learning architectures tackle challenges that include sharing the label space and the question of private and shared space for loosely related tasks (Ruder et al., 2017), for which techniques may involve a massive space of potential parameter sharing architectures.

## 3 Dataset

In this section, we present our data collection methodology and annotation process.

### 3.1 Data Collection

Considering the cultural differences and commonly debated topics in the main geographic regions where English, French, and Arabic are spoken, searching for equivalent terms in the three languages led to different results at first. Therefore, after looking for 1,000 tweets per 15 more or less equivalent phrases in the three languages, we revised our search words three times by ques-

[6] In conformity with Twitter terms and conditions.

tioning the results, adding phrases, and taking off unlikely ones in each of the languages. In fact, we started our data collection by searching for common slurs and demeaning expressions such as *"go back to where you come from"*. Then, we observed that discussions about controversial topics, such as *feminism* in general, *illegal immigrants* in English, *Islamo-gauchisme* ("Islamic leftism") in French, or *Iran* in Arabic were more likely to provoke disputes, comments filled with toxicity and thus, notable insult patterns that we looked for in subsequent search rounds.

### 3.2 Linguistic Challenges

All of the annotated tweets include original tweets only, whose content has been processed by (1) deleting unarguably detectable spam tweets, (2) removing unreadable characters and emojis, and (3) masking the names of mentioned users using *@user* and potentially enclosed URLs using *@url*. As a result, annotators had to face the lack of context generated by this normalization process.

Furthermore, we perceived code-switching in English where Hindi, Spanish, and French tokens appear in the tweets. Some French tweets also contain Romanized dialectal Arabic tokens generated by, most likely, bilingual North African Twitter users. Hence, although we eliminated most of these tweets in order to avoid misleading the annotators, the possibly remaining ones still added noise to the data.

One more challenge that the annotators and ourselves had to tackle, consisted of Arabic diglossia and switching between different Arabic dialects and Modern Standard Arabic (MSA). While MSA represents the standardized and literary variety of Arabic, there are several Arabic dialects spoken in North Africa and the Middle East in use on Twitter. Therefore, we searched for derogatory terms adapted to different circumstances, and acquired an Arabic corpus that combines tweets written in MSA and Arabic dialects. For instance, the tweet shown in Figure 1 contains a dialectal slur that means "maiden."

### 3.3 Annotation Process

We rely on the general public opinion and common linguistic knowledge to assess how people view and react to hate speech.[7] Given the subjectivity and difficulty of the task, we reminded the annotators not to let their personal opinions about the topics being discussed in the tweets influence their annotation decisions.

Our annotation guidelines explained the fact that offensive comments and hate do not necessarily come in the form of profanity. Since different degrees of discrimination work on the dehumanization of individuals or groups of people in distinct ways, we chose not to annotate the tweets within two or three classes. For instance, a sexist comment can be disrespectful, hateful, or offensive towards women. Our initial labelset was established in conformity with the prevalent antisocial behaviors people tend to deal with. We also chose to address the problem of false positives caused by the misleading use of identity words by asking the annotators to label both the target attributes and groups.

**Avoiding scams** To prevent scams, we also prepared three annotation guideline forms and three aligned labelsets written in English, French, and Modern Standard Arabic with respect to the language of the tweets to be annotated.

We requested native speakers to annotate the data and chose annotators with good reputation scores (more than 0.90). We informed the annotator in the guidelines, that in case of noticeable patterns of random labeling on a substantial number of tweets, their work will be rejected and we may have to block them. Since the rejection affects the reputation of the annotators and their chances to get new tasks on Amazon Mechanical Turk, well-reputed annotators are usually reliable. We have divided our corpora into smaller batches on Amazon Mechanical Turk in order to facilitate the analysis of the annotations of the workers and, fairly identify any incoherence patterns possibly caused by the use of an automatic translation system on the tweets, or the repetition of the same annotation schema. If we reject the work of a scam, we notify them, then reassign the tasks to other annotators.

### 3.4 Pilot Dataset

We initially put samples of 100 tweets in each of the three languages on Amazon Mechanical Turk. We showed the annotators the tweet along with lists of labels describing (a) whether it is direct or indirect hate speech; (b) if the tweet is dangerous, offensive, hateful, disrespectful, confident or supported by some URL, fearful out of ignorance,

[7] We have also provided the annotators with the Urban Dictionary definitions of some slang English words they may not be aware of.

or other; (c) the target attribute based on which it discriminates against people, specifically, race, ethnicity, nationality, gender, gender identity, sexual orientation, religious affiliation, disability, and other ("other" could refer to political ideologies or social classes.); (d) the name of its target group, and (e) whether the annotators feel anger, sadness, fear or nothing about the tweets.

Each tweet has been labeled by three annotators. We have provided them with additional text fields to fill in with labels or adjectives that would (1) better describe the tweet, (2) describe how they feel about it more accurately, and (3) name the group of people the tweet shows bias against. We kept the most commonly used labels from our initial labelset, took off some of the initial class names and added frequently introduced labels, especially the emotions of the annotators when reading the tweets and the names of the target groups. For instance, after this step, we have ended up merging *race, ethnicity, nationality* into one label *origin* given common confusions we noticed and; added *disgust* and *shock* to the emotion labelset; and introduced *socialists* as a target group label since many annotators have suggested these labels.

### 3.5 Final Dataset

The final dataset is composed of a pilot corpus of 100 tweets per language, and comparable corpora of 5,647 English tweets, 4,014 French tweets, and 3,353 Arabic tweets. Each of the annotated aspects represents a classification task of its own, that could either be evaluated independently, or, as intended in this paper, tested on how it impacts other tasks. The different labels are designed to facilitate the study of the correlations between the explicitness of the tweet, the type of hostility it conveys, its target attribute, the group it dehumanizes, how different people react to it, and the performance of multitask learning on the five tasks. We assigned each tweet to five annotators, then applied majority voting to each of the labeling tasks. Given the numbers of annotators and labels in each annotation sub-task, we allowed multilabel annotations in the most subjective classification tasks, namely the hostility type and the annotator's sentiment labels, in order to keep the right human-like approximations. If there are two annotators agreeing on two labels respectively, we add both labels to the annotation.

The average Krippendorff scores for inter-annotator agreement (IAA) are 0.153, 0.244, and 0.202 for English, French, and Arabic respectively, which are comparable to existing complex annotations (Sanguinetti et al., 2018) given the nature of the labeling tasks and the number of labels.

| Attribute | Label | En | Fr | Ar |
|---|---|---|---|---|
| Directness | Direct | 530 | 2,198 | 1,684 |
| | Indirect | 4,456 | 997 | 754 |
| Hostility | Abusive | 671 | 1,056 | 610 |
| | Hateful | 1,278 | 399 | 755 |
| | Offensive | 4,020 | 1,690 | 1,151 |
| | Disrespectful | 782 | 396 | 615 |
| | Fearful | 562 | 388 | 41 |
| | Normal | 1,359 | 1,124 | 1,197 |
| Target | Origin | 2,448 | 2,266 | 877 |
| | Gender | 638 | 27 | 548 |
| | SexOrient | 514 | 12 | 0 |
| | Religion | 68 | 146 | 145 |
| | Disability | 1,089 | 177 | 1 |
| | Other | 890 | 1,386 | 1,782 |
| Group | Individual | 497 | 918 | 915 |
| | Other | 1,590 | 1,085 | 1,470 |
| | Women | 878 | 62 | 722 |
| | SpecNeeds | 1,571 | 174 | 2 |
| | African | 86 | 311 | 51 |
| Annotator | Disgust | 3,469 | 602 | 778 |
| | Shock | 2,151 | 1,179 | 917 |
| | Anger | 2,955 | 531 | 356 |
| | Sadness | 2,775 | 1,457 | 388 |
| | Fear | 1,304 | 378 | 35 |
| | Confusion | 1,747 | 446 | 115 |
| | Indifference | 2,878 | 2,035 | 1,825 |
| Total number of tweets | | 5,647 | 4,014 | 3,353 |

Table 2: The label distributions of each task. The counts of direct and indirect hate speech include all tweets except those that are single labeled as "normal". Tweet and annotator's sentiment (Annotator) are multi-label classification tasks, while target attribute (Target) and target group (Group) are not.

We present the labelset the annotators refer to, and statistics of our annotated data in the following.

**Directness label** Annotators determine the explicitness of the tweet by labeling it as *direct* or *indirect* speech. This should be based on whether the target is explicitly named, or less easily discernible, especially if the tweet contains humor, metaphor, or figurative speech. Table 2 shows that even when partly using equivalent keywords to search for candidate tweets, there are still significant differences in the resulting data.

**Hostility type** To identify the hostility type of the tweet, we stick to the following conventions: (1) if the tweet sounds dangerous, it should be la-

beled as *abusive*; (2) according to the degree to which it spreads hate and the tone its author uses, it can be *hateful*, *offensive* or *disrespectful*; (3) if the tweet expresses or spreads fear out of ignorance against a group of individuals, it should be labeled as *fearful*; (4) otherwise it should be annotated as *normal*. We define this task to be multilabel. Table 2 shows that hostility types are relatively consistent across different languages and offensive is the most frequent label.

**Target attribute** After annotating the pilot dataset, we noticed common misconceptions regarding race, ethnicity, and nationality, therefore we merged these attributes into one label *origin*. Then, we asked the annotators to determine whether the tweet insults or discriminates against people based on their (1) *origin*, (2) *religious affiliation*, (3) *gender*, (4) *sexual orientation*, (5) *special needs* or (6) *other*. Table 2 shows there are fewer tweets targeting disability in Arabic compared to English and French and no tweets insulting people based on their sexual orientation which may be due to the fact that the labels of gender, gender identity, and sexual orientation use almost the same wording. On the other hand, French contains a small number of tweets targeting people based on their *gender* in comparison to English and Arabic. We have observed significant differences in terms of target attributes in the three languages. More data may help us examine the problems affecting targets of different linguistic backgrounds.

**Target group** We determined 16 common target groups tagged by the annotators after the first annotation step. The annotators had to decide on whether the tweet is aimed at *women, people of African descent, Hispanics, gay people, Asians, Arabs, immigrants in general, refugees*; people of different religious affiliations such as *Hindu, Christian, Jewish people, and Muslims*; or from political ideologies *socialists, and others*. We also provided the annotators with a category to cover hate directed towards one *individual*, which cannot be generalized. In case the tweet targets more than one group of people, the annotators should choose the group which would be the most affected by it according to them. Table 1 shows the counts of the five categories out of 16 that commonly occur in the three languages. In fact, most of the tweets target individuals or fall into the "other" category. In the latter case, they may target people with different political views such as liberals or conservatives in English and French, or specific ethnic groups such as Kurdish people in Arabic. English tweets tend to have more tweets targeting people with special needs, due to common language-specific demeaning terms used in conversations where people insult one another. Arabic tweets contain more hateful comments towards women for the same reason. On the other hand, the French corpus contains more tweets that are offensive towards African people, due to hateful comments generated by debates about immigrants.

**Sentiment of the annotator** We claim that the choice of a suitable emotion representation model is key to this sub-task, given the subjective nature and social ground of the annotator's sentiment analysis. After collecting the annotation results of the pilot dataset regarding how people feel about the tweets, and observing the added categories, we adopted a range of sentiments that are in the negative and neutral scales of the hourglass of emotions introduced by Cambria et al. (2011). This model includes sentiments that are connected to objectively assessed natural language opinions, and excludes what is known as self-conscious or moral emotions such as shame and guilt. Our labels include *shock*, *sadness*, *disgust*, *anger*, *fear*, *confusion* in case of ambivalence, and *indifference*. This is the second multilabel task of our model.

Table 2 shows more tweets making the annotators feel disgusted and angry in English, while annotators show more indifference in both French and Arabic. A relatively more frequent label in both French and Arabic is shock, therefore reflecting what some of the annotators were feeling during the labeling process.

## 4 Experiments

We report and discuss the results of five classification tasks: (1) the directness of the speech, (2) the hostility type of the tweet, (3) the discriminating target attribute, (4) the target group, and (5) the annotator's sentiment.

### 4.1 Models

We compare both traditional baselines using bag-of-words (BOW) as features on Logistic regression (LR), and deep learning based methods.

For deep learning based models, we run bidirectional LSTM (biLSTM) models with one hidden

layer on each of the classification tasks. Deeper BiLSTM models performed poorly due to the size of the tweets. We chose to use Sluice networks (Ruder et al., 2017) since they are suitable for loosely related tasks such as the annotated aspects of our corpora.

We test different models, namely single task single language (STSL), single task multilingual (STML), and multitask multilingual models (MTML) on our dataset. In multilingual settings, we tested Babylon multilingual word embeddings (Smith et al., 2017) and MUSE (Lample et al., 2017) on the different tasks. We use Babylon embeddings since they appear to outperform MUSE on our data.

Sluice networks (Ruder et al., 2017) learn the weights of the neural networks sharing parameters (sluices) jointly with the rest of the model and share an embedding layer, Babylon embeddings in our case, that associates the elements of an input sequence. We use a standard 1-layer BiLSTM partitioned into two subspaces, a shared subspace and a private one, forced to be orthogonal through a regularization penalty term in the loss function in order to enable the multitask network to learn both task-specific and shared representations. The hidden layer has a dimension of 200, the learning rate is initially set to 0.1 with a learning rate decay, and we use the DyNet (Neubig et al., 2017) automatic minibatch function to speed-up the computation. We initialize the cross-stitch unit to imbalanced, set the standard deviation of the Gaussian noise to 2, and use simple stochastic gradient descent (SGD) as the optimizer.

All compared methods use the same split as train:dev:test=8:1:1 and the reported results are based on the test set. We use the dev set to tune the threshold for each binary classification problem in the multilabel classification settings of each task.

### 4.2 Results and Analysis

We report both the micro and macro-F1 scores of the different classification tasks in Tables 3 and 4. *Majority* refers to labeling based on the majority label, *LR* to logistic regression, *STSL* to single task single language models, *STML* to single task multilingual models, and *MTML* to multitask multilingual models.

**STSL** STSL performs the best among all models on the directness classification, and it is also consistent in both micro and macro-F1 scores. This is due to the fact that the directness has only two labels and multilabeling is not allowed in this task. Tasks involving imbalanced data, multiclass and multilabel annotations harm the performance of the directness in multitask settings.

Since macro-F1 is the average of all F1 scores of individual labels, all deep learning models have high macro-F1 scores in English which indicates that they are particularly good at classifying the *direct* class. STSL is also comparable or better than traditional BOW feature-based classifiers when performed on other tasks in terms of micro-F1 and for most of the macro-F1 scores. This shows the power of the deep learning approach.

**MTSL** Except for the directness, MTSL usually outperforms STSL or is comparable to it. When we jointly train each task on the three languages, the performance decreases in most cases, other than the target group classification tasks. This may be due to the difference in label distributions across languages. Yet, multilingual training of the target group classification task improves in all languages. Since the target group classification task involves 16 labels, the amount of data annotated for each label is lower than in other tasks. Hence, when aggregating annotated data in different languages, the size of the training data also increases, due to the relative regularity of identification words of different groups in all three languages in comparison to other tasks.

**MTML** MTML settings do not lead to a big improvement which may be due to the class imbalance, multilabel tasks, and the difference in the nature of the tasks. In order to inspect which tasks hurt or help one another, we trained multilingual models for pairwise tasks such as (group, target), (hostility, annotator's sentiment), (hostility, target), (hostility, group), (annotator's sentiment, target) and (annotator's sentiment, group). We noticed that when trained jointly, the target attribute slightly improves the performance of the tweet's hostility type classification by 0.03,0.05 and 0.01 better than the best reported scores in English, French, and Arabic, respectively. When target groups and attributes are trained jointly, the macro F-score of the target group classification in Arabic improves by 0.25 and when we train the tweet's hostility type within the annotator's sentiment, we improve the macro F-score of Arabic by 0.02. We believe that we can take advantage of the

| Attribute | Model | Macro-F1 | | | | Micro-F1 | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | EN | FR | AR | Avg | EN | FR | AR | Avg |
| Directness | Majority | 0.50 | 0.11 | 0.50 | 0.47 | 0.79 | 0.41 | 0.54 | 0.58 |
| | LR | 0.52 | 0.50 | 0.53 | 0.52 | 0.79 | 0.50 | 0.56 | 0.62 |
| | STSL | **0.94** | **0.80** | **0.84** | **0.86** | **0.89** | **0.69** | **0.72** | **0.76** |
| | MTSL | **0.94** | 0.65 | 0.76 | 0.78 | **0.89** | 0.58 | 0.65 | 0.70 |
| | STML | **0.94** | 0.79 | 0.83 | 0.85 | 0.88 | 0.66 | **0.72** | 0.75 |
| | MTML | **0.94** | 0.78 | 0.74 | 0.82 | 0.88 | 0.66 | 0.65 | 0.73 |

Table 3: Full evaluation scores of the only binary classification task where the single task single language model consistently outperforms multilingual multitask models.

| Attribute | Model | Macro-F1 | | | | Micro-F1 | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | EN | FR | AR | Avg | EN | FR | AR | Avg |
| Tweet | Majority | 0.24 | 0.19 | 0.20 | 0.21 | 0.41 | 0.27 | 0.27 | 0.32 |
| | LR | 0.14 | 0.20 | 0.25 | 0.20 | 0.54 | 0.56 | **0.48** | 0.53 |
| | STSL | 0.24 | 0.12 | 0.31 | 0.23 | 0.49 | 0.51 | 0.47 | 0.49 |
| | MTSL | 0.09 | 0.20 | 0.33 | 0.21 | **0.55** | **0.59** | 0.46 | **0.54** |
| | STML | 0.04 | 0.07 | **0.35** | 0.16 | 0.54 | 0.47 | 0.37 | 0.46 |
| | MTML | **0.30** | **0.28** | **0.35** | **0.31** | 0.45 | 0.48 | 0.44 | 0.46 |
| Target Attribute | Majority | 0.15 | 0.13 | 0.28 | 0.19 | 0.25 | 0.32 | 0.40 | 0.32 |
| | LR | 0.41 | 0.35 | 0.47 | 0.41 | 0.52 | 0.55 | 0.53 | 0.53 |
| | STSL | 0.42 | 0.18 | **0.63** | 0.41 | **0.68** | 0.71 | 0.50 | 0.63 |
| | MTSL | 0.41 | **0.43** | 0.41 | **0.42** | **0.68** | 0.67 | **0.56** | **0.64** |
| | STML | 0.39 | 0.09 | 0.24 | 0.24 | 0.67 | 0.62 | 0.53 | 0.61 |
| | MTML | **0.43** | 0.24 | 0.16 | 0.28 | 0.66 | **0.72** | 0.51 | 0.63 |
| Target Group | Majority | 0.07 | 0.06 | 0.08 | 0.07 | 0.18 | 0.14 | 0.35 | 0.22 |
| | LR | **0.18** | 0.33 | **0.40** | **0.30** | 0.34 | 0.40 | 0.62 | 0.46 |
| | STSL | 0.04 | 0.21 | 0.04 | 0.10 | 0.48 | **0.59** | 0.58 | 0.55 |
| | MTSL | 0.04 | 0.27 | 0.15 | 0.15 | **0.50** | 0.54 | 0.55 | 0.53 |
| | STML | 0.11 | **0.37** | 0.13 | 0.20 | 0.49 | 0.57 | **0.64** | **0.56** |
| | MTML | 0.06 | 0.19 | 0.10 | 0.11 | **0.50** | 0.54 | 0.56 | 0.53 |
| Annotator's Sentiment | Majority | 0.42 | 0.21 | 0.17 | 0.27 | 0.46 | 0.31 | 0.32 | 0.39 |
| | LR | 0.29 | 0.15 | 0.14 | 0.19 | 0.45 | 0.30 | 0.46 | 0.40 |
| | STSL | **0.57** | **0.30** | 0.12 | **0.33** | 0.57 | 0.39 | **0.48** | 0.48 |
| | MTSL | **0.57** | 0.17 | 0.17 | 0.30 | 0.57 | **0.50** | 0.45 | 0.51 |
| | STML | 0.47 | 0.22 | 0.13 | 0.27 | **0.59** | 0.49 | **0.48** | **0.52** |
| | MTML | 0.55 | 0.20 | **0.21** | 0.32 | 0.58 | 0.45 | 0.45 | 0.49 |

Table 4: Full evaluation of tasks where multilingual and multitask models outperform on average single task single language model on four different tasks.

correlations between target attributes and groups along with other tasks, to set logic rules and develop better multilingual and multitask settings.

## 5 Conclusion

In this paper, we presented a multilingual hate speech dataset of English, French, and Arabic tweets. We analyzed in details the difficulties related to the collection and annotation of this dataset. We performed multilingual and multitask learning on our corpora and showed that deep learning models perform better than traditional BOW-based models in most of the multilabel classification tasks. Multilingual multitask learning also helped tasks where each label had less annotated data associated with it.

Better tuned deep learning settings in our multilingual and multitask models would be expected to outperform the existing state-of-the-art embeddings and algorithms applied to our data. The different annotation labels and comparable corpora would help us perform transfer learning and investigate how multimodal information on the tweets, additional unlabeled data, label transformation, and label information sharing may boost the classification performance in the future.

## Acknowledgement

This paper was supported by the Early Career Scheme (ECS, No. 26206717) from Research Grants Council in Hong Kong, and by postgraduate studentships from the Computer Science and Engineering department of the Hong Kong University of Science and Technology.

## References


Nuha Albadi, Maram Kurdi, and Shivakant Mishra. 2018. Are they our brothers? analysis and detection of religious hate speech in the arabic twittersphere. In *Proceedings of ASONAM*. IEEE Computer Society, pages 69–76.

Isabelle Augenstein, Sebastian Ruder, and Anders Søgaard. 2018. Multi-task learning of pairwise sequence classification tasks over disparate label spaces. In *Proceedings of NAACL*. Association for Computational Linguistics, pages 1896–1906.

Aditya Bohra, Deepanshu Vijay, Vinay Singh, Syed Sarfaraz Akhtar, and Manish Shrivastava. 2018. A dataset of hindi-english code-mixed social media text for hate speech detection. In *Proceedings of the Second Workshop on Computational Modeling of People's Opinions, Personality, and Emotions in Social Media*. Association for Computational Linguistics, pages 36–41.

Erik Cambria, Andrew Livingstone, and Amir Hussain. 2011. The hourglass of emotions. In *COST 2102 Training School*. Springer, volume 7403 of *Lecture Notes in Computer Science*, pages 144–157.

Despoina Chatzakou, Nicolas Kourtellis, Jeremy Blackburn, Emiliano De Cristofaro, Gianluca Stringhini, and Athena Vakali. 2017. Mean birds: Detecting aggression and bullying on twitter. In *Proceedings of WebSci '17*. ACM, pages 13–22.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12:2493–2537.

Thomas Davidson, Dana Warmsley, Michael W. Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Proceedings of ICWSM*. pages 512–515.

Mai ElSherief, Vivek Kulkarni, Dana Nguyen, William Y. Wang, and Elizabeth Belding. 2018. Hate lingo: A target-based linguistic analysis of hate speech in social media. In *Proceedings of ICWSM*.

Antigoni-Maria Founta, Constantinos Djouvas, Despoina Chatzakou, Ilias Leontiadis, Jeremy Blackburn, Gianluca Stringhini, Athena Vakali, Michael Sirivianos, and Nicolas Kourtellis. 2018. Large scale crowdsourcing and characterization of twitter abusive behavior. In *Proceedings ICWSM*. pages 491–500.

Thiago Galery, Efstathios Charitos, and Ye Tian. 2018. Aggression identification and multi lingual word embeddings. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying*. Association for Computational Linguistics, pages 74–79.

Jennifer Golbeck, Zahra Ashktorab, Rashad O. Banjo, Alexandra Berlinger, Siddharth Bhagwan, Cody Buntain, Paul Cheakalos, Alicia A. Geller, Quint Gergory, Rajesh Kumar Gnanasekaran, Raja Rajan Gunasekaran, Kelly M. Hoffman, Jenny Hottle, Vichita Jienjitlert, Shivika Khare, Ryan Lau, Marianna J. Martindale, Shalmali Naik, Heather L. Nixon, Piyush Ramachandran, Kristine M. Rogers, Lisa Rogers, Meghna Sardana Sarin, Gaurav Shahane, Jayanee Thanki, Priyanka Vengataraman, Zijian Wan, and Derek Michael Wu. 2017. A large labeled corpus for online harassment research. In *Proceedings of the 2017 ACM on Web Science Conference*. ACM, WebSci '17, pages 229–233.

Kazuma Hashimoto, caiming xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple nlp tasks. In *Proceedings of EMNLP*. Association for Computational Linguistics, pages 1923–1933.

Cynthia Van Hee, Els Lefever, Ben Verhoeven, Julie Mennes, Bart Desmet, Guy De Pauw, Walter Daelemans, and Véronique Hoste. 2015. Detection and fine-grained classification of cyberbullying events. In *Proceedings of RANLP*. pages 672–680.

Filip Klubička and Raquel Fernandez. 2018. Examining a hate speech corpus for hate speech detection and popularity prediction. In *Proceedings LREC*.

Nane Kratzke. 2017. The #BTW17 Twitter Dataset - Recorded Tweets of the Federal Election Campaigns of 2017 for the 19th German Bundestag. *Data* 2(4).

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2017. Unsupervised machine translation using monolingual corpora only. *arXiv preprint arXiv:1711.00043* .

Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In *International Conference on Learning Representations*.

Shervin Malmasi and Marcos Zampieri. 2018. Challenges in discriminating profanity from hate speech. *J. Exp. Theor. Artif. Intell.* 30(2):187–202.

Ryan Martin, Kelsey Ryan Coyier, Leah Vansistine, and Kelly Schroeder. 2012. Anger on the internet: The perceived value of rant-sites. *Cyberpsychology, behavior and social networking* 16.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980* .

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive language detection in online user content. In *Proceedings of WWW '16*. International World Wide Web Conferences Steering Committee, pages 145–153.

Ji Ho Park, Jamin Shin, and Pascale Fung. 2018. Reducing gender bias in abusive language detection. In *Proceedings of EMNLP*. Association for Computational Linguistics, pages 2799–2804.

Hao Peng, Sam Thomson, and Noah A. Smith. 2017. Deep multitask learning for semantic dependency parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 2037–2048.

Jing Qian, Mai ElSherief, Elizabeth Belding, and William Yang Wang. 2018. Hierarchical cvae for fine-grained hate speech classification. In *Proceedings of EMNLP*. Association for Computational Linguistics, pages 3550–3559.

Björn Ross, Michael Rist, Guillermo Carbonell, Benjamin Cabrera, Nils Kurowsky, and Michael Wojatzki. 2017. Measuring the reliability of hate speech annotations: The case of the european refugee crisis. *CoRR* abs/1701.08118.

Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. 2017. Sluice networks: Learning what to share between loosely related tasks. *arXiv preprint arXiv:1705.08142* .

Manuela Sanguinetti, Fabio Poletto, Cristina Bosco, Viviana Patti, and Marco Stranisci. 2018. An italian twitter corpus of hate speech against immigrants. In *LREC*. European Language Resources Association (ELRA).

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*. Association for Computational Linguistics, pages 1–10.

Samuel L. Smith, David H. P. Turban, Steven Hamblin, and Nils Y. Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. *CoRR* abs/1702.03859.

Wiktor Soral, Micha Bilewicz, and Mikoaj Winiewski. 2017. Exposure to hate speech increases prejudice through desensitization. *Aggressive Behavior* 44.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding abuse: A typology of abusive language detection subtasks. In *Proceedings of the First Workshop on Abusive Language Online*. Association for Computational Linguistics, pages 78–84.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL Student Research Workshop*. The Association for Computational Linguistics, pages 88–93.

# MultiFC: A Real-World Multi-Domain Dataset for Evidence-Based Fact Checking of Claims

**Isabelle Augenstein** **Christina Lioma** **Dongsheng Wang** **Lucas Chaves Lima**
**Casper Hansen** **Christian Hansen** **Jakob Grue Simonsen**
Department of Computer Science
University of Copenhagen
{augenstein,c.lioma,wang,lcl,c.hansen,chrh,simonsen}@di.ku.dk

## Abstract

We contribute the largest publicly available dataset of naturally occurring factual claims for the purpose of automatic claim verification. It is collected from 26 fact checking websites in English, paired with textual sources and rich metadata, and labelled for veracity by human expert journalists. We present an in-depth analysis of the dataset, highlighting characteristics and challenges. Further, we present results for automatic veracity prediction, both with established baselines and with a novel method for joint ranking of evidence pages and predicting veracity that outperforms all baselines. Significant performance increases are achieved by encoding evidence, and by modelling metadata. Our best-performing model achieves a Macro F1 of 49.2%, showing that this is a challenging testbed for claim veracity prediction.

## 1 Introduction

Misinformation and disinformation are two of the most pertinent and difficult challenges of the information age, exacerbated by the popularity of social media. In an effort to counter this, a significant amount of manual labour has been invested in fact checking claims, often collecting the results of these manual checks on fact checking portals or websites such as politifact.com or snopes.com. In a parallel development, researchers have recently started to view fact checking as a task that can be partially automated, using machine learning and NLP to automatically predict the *veracity* of claims. However, existing efforts either use small datasets consisting of naturally occurring claims (e.g. Mihalcea and Strapparava (2009); Zubiaga et al. (2016)), or datasets consisting of artificially constructed claims such as FEVER (Thorne et al., 2018). While the latter offer valuable contributions to further automatic claim verification work, they cannot replace real-world datasets.

| Feature | Value |
|---|---|
| ClaimID | farg-00004 |
| Claim | Mexico and Canada assemble cars with foreign parts and send them to the U.S. with no tax. |
| Label | distorts |
| Claim URL | https://www.factcheck.org/2018/10/factchecking-trump-on-trade/ |
| Reason | None |
| Category | the-factcheck-wire |
| Speaker | Donald Trump |
| Checker | Eugene Kiely |
| Tags | North American Free Trade Agreement |
| Claim Entities | United_States, Canada, Mexico |
| Article Title | FactChecking Trump on Trade |
| Publish Date | October 3, 2018 |
| Claim Date | Monday, October 1, 2018 |

Table 1: An example of a claim instance. Entities are obtained via entity linking. Article and outlink texts, evidence search snippets and pages are not shown.

**Contributions.** We introduce the currently largest claim verification dataset of naturally occurring claims.[1] It consists of 34,918 claims, collected from 26 fact checking websites in English; evidence pages to verify the claims; the context in which they occurred; and rich metadata (see Table 1 for an example). We perform a thorough analysis to identify characteristics of the dataset such as entities mentioned in claims. We demonstrate the utility of the dataset by training state of the art veracity prediction models, and find that evidence pages as well as metadata significantly contribute to model performance. Finally, we propose a novel model that jointly ranks evidence pages and performs veracity prediction. The best-performing model achieves a Macro F1 of 49.2%, showing that this is a non-trivial dataset with remaining challenges for future work.

[1]The dataset is found here: https://copenlu.github.io/publication/2019_emnlp_augenstein/



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4685–4697,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

## 2 Related Work

### 2.1 Datasets

Over the past few years, a variety of mostly small datasets related to fact checking have been released. An overview over core datasets is given in Table 2. The datasets can be grouped into four categories (I–IV). Category I contains datasets aimed at testing how well the veracity[3] of a claim can be predicted using the claim alone, without context or evidence documents. Category II contains datasets bundled with documents related to each claim – either topically related to provide context, or serving as evidence. Those documents are, however, not annotated. Category III is for predicting veracity; they encourage retrieving evidence documents as part of their task description, but do not distribute them. Finally, category IV comprises datasets annotated for both veracity and stance. Thus, every document is annotated with a label indicating whether the document supports or denies the claim, or is unrelated to it. Additional labels can then be added to the datasets to better predict veracity, for instance by jointly training stance and veracity prediction models.

Methods not shown in the table, but related to fact checking, are stance detection for claims (Ferreira and Vlachos, 2016; Pomerleau and Rao, 2017; Augenstein et al., 2016a; Kochkina et al., 2017; Augenstein et al., 2016b; Zubiaga et al., 2018; Riedel et al., 2017), satire detection (Rubin et al., 2016), clickbait detection (Karadzhov et al., 2017), conspiracy news detection (Tacchini et al., 2017), rumour cascade detection (Vosoughi et al., 2018) and claim perspectives detection (Chen et al., 2019).

Claims are obtained from a variety of sources, including Wikipedia, Twitter, criminal reports and fact checking websites such as politifact.com and snopes.com. The same goes for documents – these are often websites obtained through Web search queries, or Wikipedia documents, tweets or Facebook posts. Most datasets contain a fairly small number of claims, and those that do not, often lack evidence documents. An exception is Thorne et al. (2018), who create a Wikipedia-based fact checking dataset. While a good testbed for developing deep neural architectures, their dataset is artificially constructed and can thus not take metadata about claims into account.

**Contributions:** We provide a dataset that, uniquely among extant datasets, contains a large number of *naturally occurring* claims and rich additional meta-information.

### 2.2 Methods

Fact checking methods partly depend on the type of dataset used. Methods only taking into account claims typically encode those with CNNs or RNNs (Wang, 2017; Pérez-Rosas et al., 2018), and potentially encode metadata (Wang, 2017) in a similar way. Methods for small datasets often use handcrafted features that are a mix of bag of word and other lexical features, e.g. LIWC, and then use those as input to a SVM or MLP (Mihalcea and Strapparava, 2009; Pérez-Rosas et al., 2018; Baly et al., 2018). Some use additional Twitter-specific features (Enayet and El-Beltagy, 2017). More involved methods taking into account evidence documents, often trained on larger datasets, consist of evidence identification and ranking following a neural model that measures the compatibility between claim and evidence (Thorne et al., 2018; Mihaylova et al., 2018; Yin and Roth, 2018).

**Contributions:** The latter category above is the most related to our paper as we consider evidence documents. However, existing models are not trained jointly for evidence identification, or for stance and veracity prediction, but rather employ a pipeline approach. Here, we show that a joint approach that learns to weigh evidence pages by their importance for veracity prediction can improve downstream veracity prediction performance.

## 3 Dataset Construction

We crawled a total of 43,837 claims with their metadata (see details in Table 11). We present the data collection in terms of selecting sources, crawling claims and associated metadata (Section 3.1); retrieving evidence pages; and linking entities in the crawled claims (Section 3.3).

### 3.1 Selection of sources

We crawled all active fact checking websites in English listed by Duke Reporters' Lab[4] and on the Fact Checking Wikipedia page.[5] This resulted in

[3] We use *veracity*, *claim credibility*, and *fake news* prediction interchangeably here – these terms are often conflated in the literature and meant to have the same meaning.

[4] https://reporterslab.org/fact-checking/

[5] https://en.wikipedia.org/wiki/Fact_checking

| Dataset | # Claims | Labels | metadata | Claim Sources |
|---|---|---|---|---|
| **I: Veracity prediction w/o evidence** | | | | |
| Wang (2017) | 12,836 | 6 | Yes | Politifact |
| Pérez-Rosas et al. (2018) | 980 | 2 | No | News Websites |
| **II: Veracity** | | | | |
| Bachenko et al. (2008) | 275 | 2 | No | Criminal Reports |
| Mihalcea and Strapparava (2009) | 600 | 2 | No | Crowd Authors |
| Mitra and Gilbert (2015)† | 1,049 | 5 | No | Twitter |
| Ciampaglia et al. (2015)† | 10,000 | 2 | No | Google, Wikipedia |
| Popat et al. (2016) | 5,013 | 2 | Yes | Wikipedia, Snopes |
| Shu et al. (2018)† | 23,921 | 2 | Yes | Politifact, gossipcop.com |
| Datacommons Fact Check[2] | 10,564 | 2-6 | Yes | Fact Checking Websites |
| **III: Veracity (evidence encouraged, but not provided)** | | | | |
| Barrn-Cedeo et al. (2018) | 150 | 3 | No | factcheck.org, Snopes |
| **IV: Veracity + stance** | | | | |
| Vlachos and Riedel (2014) | 106 | 5 | Yes | Politifact, Channel 4 News |
| Zubiaga et al. (2016) | 330 | 3 | Yes | Twitter |
| Derczynski et al. (2017) | 325 | 3 | Yes | Twitter |
| Baly et al. (2018) | 422 | 2 | No | ara.reuters.com, verify-sy.com |
| Thorne et al. (2018)† | 185,445 | 3 | No | Wikipedia |
| **V: Veracity + evidence relevancy** | | | | |
| MultiFC | 36,534 | 2-40 | Yes | Fact Checking Websites |

Table 2: Comparison of fact checking datasets. † indicates claims are not 'naturally occuring': Mitra and Gilbert (2015) use events as claims; Ciampaglia et al. (2015) use DBPedia tiples as claims; Shu et al. (2018) use tweets as claims; and Thorne et al. (2018) rewrite sentences in Wikipedia as claims.

38 websites in total (shown in Table 11). Out of these, ten websites could not be crawled, as further detailed in Table 9. In the later experimental descriptions, we refer to the part of the dataset crawled from a specific fact checking website as a *domain*, and we refer to each website as *source*.

From each source, we crawled the ID, claim, label, URL, reason for label, categories, person making the claim (speaker), person fact checking the claim (checker), tags, article title, publication date, claim date, as well as the full text that appears when the claim is clicked. Lastly, the above full text contains hyperlinks, so we further crawled the full text that appears when each of those hyperlinks are clicked (outlinks).

There were a number of crawling issues, e.g. security protection of websites with SSL/TLS protocols, time out, URLs that pointed to pdf files instead of HTML content, or unresolvable encoding. In all of these cases, the content could not be retrieved. For some websites, no veracity labels were available, in which case, they were not selected as domains for training a veracity prediction model. Moreover, not all types of metadata (category, speaker, checker, tags, claim date, publish date) were available for all websites; and availability of articles and full texts differs as well.

We performed semi-automatic cleansing of the dataset as follows. First, we double-checked that the veracity labels would not appear in claims. For some domains, the first or last sentence of the claim would sometimes contain the veracity label, in which case we would discard either the full sentence or part of the sentence. Next, we checked the dataset for duplicate claims. We found 202 such instances, 69 of them with different labels. Upon manual inspection, this was mainly due to them appearing on different websites, with labels not differing much in practice (e.g. 'Not true', vs. 'Mostly False'). We made sure that all such duplicate claims would be in the training split of the dataset, so that the models would not have an unfair advantage. Finally, we performed some minor manual merging of label types for the same domain where it was clear that they were supposed to denote the same level of veracity (e.g. 'distorts', 'distorts the facts').

This resulted in a total of 36,534 claims with their metadata. For the purposes of fact verification, we discarded instances with labels that occur fewer than 5 times, resulting in 34,918 claims. The number of instances, as well as labels per domain, are shown in Table 6 and label names in Table 10 in the appendix. The dataset is split into a training part (80%) and a development and testing part (10% each) in a label-stratified manner. Note that

the domains vary in the number of labels, ranging from 2 to 27. Labels include both straight-forward ratings of veracity ('correct', 'incorrect'), but also labels that would be more difficult to map onto a veracity scale (e.g. 'grass roots movement!', 'misattributed', 'not the whole story'). We therefore do not postprocess label types across domains to map them onto the same scale, and rather treat them as is. In the methodology section (Section 4), we show how a model can be trained on this dataset regardless by framing this multi-domain veracity prediction task as a multi-task learning (MTL) one.

### 3.2 Retrieving Evidence Pages

The text of each claim is submitted verbatim as a query to the Google Search API (without quotes). The 10 most highly ranked search results are retrieved, for each of which we save the title; Google search rank; URL; time stamp of last update; search snippet; as well as the full Web page. We acknowledge that search results change over time, which might have an effect on veracity prediction. However, studying such temporal effects is outside the scope of this paper. Similar to Web crawling claims, as described in Section 3.1, the corresponding Web pages can in some cases not be retrieved, in which case fewer than 10 evidence pages are available. The resulting evidence pages are from a wide variety of URL domains, though with a predictable skew towards popular websites, such as Wikipedia or The Guardian (see Table 3 for detailed statistics).

### 3.3 Entity Detection and Linking

To better understand what claims are about, we conduct entity linking for all claims. Specifically, mentions of people, places, organisations, and other named entities within a claim are recognised and linked to their respective Wikipedia pages, if available. Where there are different entities with the same name, they are disambiguated. For this, we apply the state-of-the-art neural entity linking model by Kolitsas et al. (2018). This results in a total of 25,763 entities detected and linked to Wikipedia, with a total of 15,351 claims involved, meaning that 42% of all claims contain entities that can be linked to Wikipedia. Later on, we use entities as additional metadata (see Section 4.3). The distribution of claim numbers according to the number of entities they contain is shown in Figure 1. We observe that the majority of claims have one to four entities, and the maximum number of 35 entities occurs in one claim only. Out of the 25,763 entities, 2,767 are unique entities. The top 30 most frequent entities are listed in Table 4. This clearly shows that most of the claims involve entities related to the United States, which is to be expected, as most of the fact checking websites are US-based.

| Domain | % |
|---|---|
| https://en.wikipedia.org/ | 4.425 |
| https://www.snopes.com/ | 3.992 |
| https://www.washingtonpost.com/ | 3.025 |
| https://www.nytimes.com/ | 2.478 |
| https://www.theguardian.com/ | 1.807 |
| https://www.youtube.com/ | 1.712 |
| https://www.dailymail.co.uk/ | 1.558 |
| https://www.usatoday.com/ | 1.279 |
| https://www.politico.com/ | 1.241 |
| http://www.politifact.com/ | 1.231 |
| https://www.pinterest.com/ | 1.169 |
| https://www.factcheck.org/ | 1.09 |
| https://www.gossipcop.com/ | 1.073 |
| https://www.cnn.com/ | 1.065 |
| https://www.npr.org/ | 0.957 |
| https://www.forbes.com/ | 0.911 |
| https://www.vox.com/ | 0.89 |
| https://www.theatlantic.com/ | 0.88 |
| https://twitter.com/ | 0.767 |
| https://www.hoax-slayer.net/ | 0.655 |
| http://time.com/ | 0.554 |
| https://www.bbc.com/ | 0.551 |
| https://www.nbcnews.com/ | 0.515 |
| https://www.cnbc.com/ | 0.514 |
| https://www.cbsnews.com/ | 0.503 |
| https://www.facebook.com/ | 0.5 |
| https://www.newyorker.com/ | 0.495 |
| https://www.foxnews.com/ | 0.468 |
| https://people.com/ | 0.439 |
| http://www.cnn.com/ | 0.419 |

Table 3: The top 30 most frequently occurring URL domains.

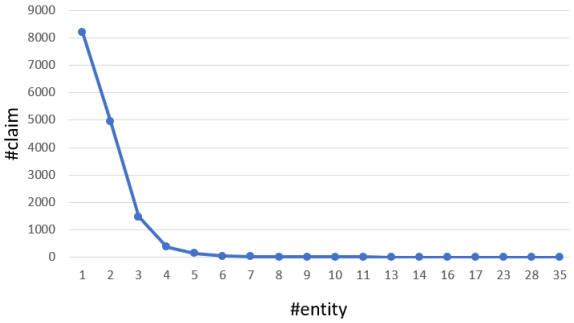


Figure 1: Distribution of entities in claims.

## 4 Claim Veracity Prediction

We train several models to predict the veracity of claims. Those fall into two categories: those that

| Entity | Frequency |
|---|---|
| United_States | 2810 |
| Barack_Obama | 1598 |
| Republican_Party_(United_States) | 783 |
| Texas | 665 |
| Democratic_Party_(United_States) | 560 |
| Donald_Trump | 556 |
| Wisconsin | 471 |
| United_States_Congress | 354 |
| Hillary_Rodham_Clinton | 306 |
| Bill_Clinton | 292 |
| California | 285 |
| Russia | 275 |
| Ohio | 239 |
| China | 229 |
| George_W._Bush | 208 |
| Medicare_(United_States) | 206 |
| Australia | 186 |
| Iran | 183 |
| Brad_Pitt | 180 |
| Islam | 178 |
| Iraq | 176 |
| Canada | 174 |
| White_House | 166 |
| New_York_City | 164 |
| Washington,_D.C. | 164 |
| Jennifer_Aniston | 163 |
| Mexico | 158 |
| Ted_Cruz | 152 |
| Federal_Bureau_of_Investigation | 146 |
| Syria | 130 |

Table 4: Top 30 most frequent entities listed by their Wikipedia URL with prefix omitted

only consider the claims themselves, and those that encode evidence pages as well. In addition, claim metadata (speaker, checker, linked entities) is optionally encoded for both categories of models, and ablation studies with and without that metadata are shown. We first describe the base model used in Section 4.1, followed by introducing our novel evidence ranking and veracity prediction model in Section 4.2, and lastly the metadata encoding model in Section 4.3.

### 4.1 Multi-Domain Claim Veracity Prediction with Disparate Label Spaces

Since not all fact checking websites use the same claim labels (see Table 6, and Table 10 in the appendix), training a claim veracity prediction model is not entirely straight-forward. One option would be to manually map those labels onto one another. However, since the sheer number of labels is rather large (165), and it is not always clear from the guidelines on fact checking websites how they can be mapped onto one another, we opt to learn how these labels relate to one another as part of the veracity prediction model. To do so, we employ the multi-task learning (MTL) approach inspired by collaborative filtering presented in Augenstein et al. (2018) (*MTL with LEL*–multitask learning with label embedding layer) that excels on pairwise sequence classification tasks with disparate label spaces. More concretely, each domain is modelled as its own task in a MTL architecture, and labels are projected into a fixed-length label embedding space. Predictions are then made by taking the dot product between the claim-evidence embeddings and the label embeddings. By doing so, the model implicitly learns how semantically close the labels are to one another, and can benefit from this knowledge when making predictions for individual tasks, which on their own might only have a small number of instances. When making predictions for individual domains/tasks, both at training and at test time, as well as when calculating the loss, a mask is applied such that the valid and invalid labels for that task are restricted to the set of known task labels.

Note that the setting here slightly differs from Augenstein et al. (2018). There, tasks are less strongly related to one another; for example, they consider stance detection, aspect-based sentiment analysis and natural language inference. Here, we have different domains, as opposed to conceptually different tasks, but use their framework, as we have the same underlying problem of disparate label spaces. A more formal problem definition follows next, as our evidence ranking and veracity prediction model in Section 4.2 then builds on it.

#### 4.1.1 Problem Definition

We frame our problem as a multi-task learning one, where access to labelled datasets for $T$ tasks $\mathcal{T}_1, \ldots, \mathcal{T}_T$ is given at training time with a target task $\mathcal{T}_T$ that is of particular interest. The training dataset for task $\mathcal{T}_i$ consists of $N$ examples $X_{\mathcal{T}_i} = \{x_1^{\mathcal{T}_i}, \ldots, x_N^{\mathcal{T}_i}\}$ and their labels $Y_{\mathcal{T}_i} = \{\mathbf{y}_1^{\mathcal{T}_i}, \ldots, \mathbf{y}_N^{\mathcal{T}_i}\}$. The base model is a classic deep neural network MTL model (Caruana, 1993) that shares its parameters across tasks and has task-specific softmax output layers that output a probability distribution $\mathbf{p}^{\mathcal{T}_i}$ for task $\mathcal{T}_i$:

$$\mathbf{p}^{\mathcal{T}_i} = \text{softmax}(\mathbf{W}^{\mathcal{T}_i}\mathbf{h} + \mathbf{b}^{\mathcal{T}_i}) \quad (1)$$

where $\text{softmax}(\mathbf{x}) = e^{\mathbf{x}} / \sum_{i=1}^{\|\mathbf{x}\|} e^{\mathbf{x}_i}$, $\mathbf{W}^{\mathcal{T}_i} \in \mathbb{R}^{L_i \times h}$, $\mathbf{b}^{\mathcal{T}_i} \in \mathbb{R}^{L_i}$ is the weight matrix and bias term of the output layer of task $\mathcal{T}_i$ respectively, $\mathbf{h} \in \mathbb{R}^h$ is the jointly learned hidden rep-

resentation, $L_i$ is the number of labels for task $\mathcal{T}_i$, and $h$ is the dimensionality of $\mathbf{h}$. The MTL model is trained to minimise the sum of individual task losses $\mathcal{L}_1 + \ldots + \mathcal{L}_T$ using a negative log-likelihood objective.

**Label Embedding Layer.** To learn the relationships between labels, a Label Embedding Layer (LEL) embeds labels of all tasks in a joint Euclidian space. Instead of training separate softmax output layers as above, a label compatibility function $c(\cdot, \cdot)$ measures how similar a label with embedding $\mathbf{l}$ is to the hidden representation $\mathbf{h}$:

$$c(\mathbf{l}, \mathbf{h}) = \mathbf{l} \cdot \mathbf{h} \tag{2}$$

where $\cdot$ is the dot product. Padding is applied such that $l$ and $h$ have the same dimensionality. Matrix multiplication and softmax are used for making predictions:

$$\mathbf{p} = \text{softmax}(\mathbf{Lh}) \tag{3}$$

where $\mathbf{L} \in \mathbb{R}^{(\sum_i L_i) \times l}$ is the label embedding matrix for all tasks and $l$ is the dimensionality of the label embeddings. We apply a task-specific mask to $\mathbf{L}$ in order to obtain a task-specific probability distribution $\mathbf{p}^{\mathcal{T}_i}$. The LEL is shared across all tasks, which allows the model to learn the relationships between labels in the joint embedding space.

## 4.2 Joint Evidence Ranking and Claim Veracity Prediction

So far, we have ignored the issue of how to obtain claim representation, as the base model described in the previous section is agnostic to how instances are encoded. A very simple approach, which we report as a baseline, is to encode claim texts only. Such a model ignores evidence for and against a claim, and ends up guessing the veracity based on surface patterns observed in the claim texts.

We next introduce two variants of evidence-based veracity prediction models that encode 10 pieces of evidence in addition to the claim. Here, we opt to encode search snippets as opposed to whole retrieved pages. While the latter would also be possible, it comes with a number of additional challenges, such as encoding large documents, parsing tables or PDF files, and encoding images or videos on these pages, which we leave to future work. Search snippets also have the benefit that they already contain summaries of the part of the page content that is most related to the claim.

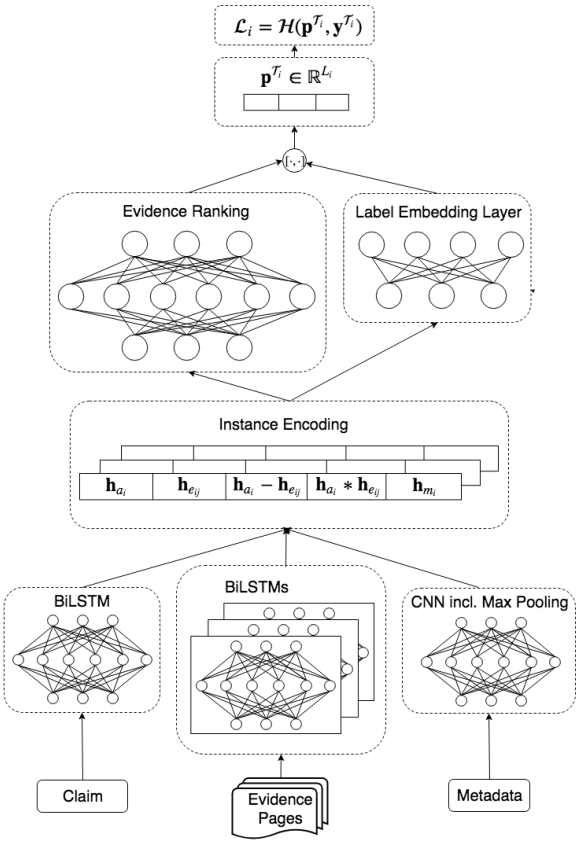


Figure 2: The Joint Veracity Prediction and Evidence Ranking model, shown for one task.

### 4.2.1 Problem Definition

Our problem is to obtain encodings for $N$ examples $X_{\mathcal{T}_i} = \{x_1^{\mathcal{T}_i}, \ldots, x_N^{\mathcal{T}_i}\}$. For simplicity, we will henceforth drop the task superscript and refer to instances as $X = \{x_1, \ldots, x_N\}$, as instance encodings are learned in a task-agnostic fashion. Each example further consists of a claim $a_i$ and $k = 10$ evidence pages $E_k = \{e_{1_0}, \ldots, e_{N_{10}}\}$.

Each claim and evidence page is encoded with a BiLSTM to obtain a sentence embedding, which is the concatenation of the last state of the forward and backward reading of the sentence, i.e. $\mathbf{h} = BiLSTM(\cdot)$, where $\mathbf{h}$ is the sentence embedding.

Next, we want to combine claims and evidence sentence embeddings into joint instance representations. In the simplest case, referred to as model variant *crawled_avg*, we mean average the BiLSTM sentence embeddings of all evidence pages (signified by the overline) and concatenate those with the claim embeddings, i.e.

$$\mathbf{s}_{g_i} = [\mathbf{h}_{a_i}; \overline{\mathbf{h}_{E_i}}] \tag{4}$$

where $s_{g_i}$ is the resulting encoding for training example $i$ and $[\cdot;\cdot]$ denotes vector concatenation.

However, this has the disadvantage that all evidence pages are considered equal.

**Evidence Ranking** The here proposed alternative instance encoding model, *crawled_ranked*, which achieves the highest overall performance as discussed in Section 5, learns the compatibility between an instance's claim and each evidence page. It ranks evidence pages by their utility for the veracity prediction task, and then uses the resulting ranking to obtain a weighted combination of all claim-evidence pairs. No direct labels are available to learn the ranking of individual documents, only for the veracity of the associated claim, so the model has to learn evidence ranks implicitly.

To combine claim and evidence representations, we use the matching model proposed for the task of natural language inference by Mou et al. (2016) and adapt it to combine an instance's claim representation with each evidence representation, i.e.

$$s_{r_{i_j}} = [\mathbf{h}_{a_i}; \mathbf{h}_{e_{i_j}}; \mathbf{h}_{a_i} - \mathbf{h}_{e_{i_j}}; \mathbf{h}_{a_i} \cdot \mathbf{h}_{e_{i_j}}] \quad (5)$$

where $s_{r_{i_j}}$ is the resulting encoding for training example $i$ and evidence page $j$ , $[\cdot;\cdot]$ denotes vector concatenation, and $\cdot$ denotes the dot product.

All joint claim-evidence representations $\mathbf{s}_{r_{i_0}}, \ldots, \mathbf{s}_{r_{i_{10}}}$ are then projected into the binary space via a fully connected layer FC, followed by a non-linear activation function $f$, to obtain a soft ranking of claim-evidence pairs, in practice a 10-dimensional vector,

$$\mathbf{o}_i = [f(\mathrm{FC}(s_{r_{i_0}})); \ldots; f(\mathrm{FC}(s_{r_{i_{10}}}))] \quad (6)$$

where $[\cdot;\cdot]$ denotes concatenation.

Scores for all labels are obtained as per (6) above, with the same input instance embeddings as for the evidence ranker, i.e. $s_{r_{i_j}}$. Final predictions for all claim-evidence pairs are then obtained by taking the dot product between the label scores and binary evidence ranking scores, i.e.

$$\mathbf{p}_i = \mathrm{softmax}(c(\mathbf{l}, \mathbf{s}_{\mathbf{r_i}}) \cdot \mathbf{o}_i) \quad (7)$$

Note that the novelty here is that, unlike for the model described in Mou et al. (2016), we have no direct labels for learning weights for this matching model. Rather, our model has to implicitly learn these weights for each claim-evidence pair in an end-to-end fashion given the veracity labels.

| Model | Micro F1 | Macro F1 |
|---|---|---|
| claim-only | 0.469 | 0.253 |
| claim-only_embavg | 0.384 | 0.302 |
| crawled-docavg | 0.438 | 0.248 |
| crawled_ranked | 0.613 | 0.441 |
| claim-only + meta | 0.494 | 0.324 |
| claim-only_embavg + meta | 0.418 | 0.333 |
| crawled-docavg + meta | 0.483 | 0.286 |
| crawled_ranked + meta | **0.625** | **0.492** |

Table 5: Results with different model variants on the test set, 'meta' means all metadata is used.

### 4.3 Metadata

We experiment with how useful claim metadata is, and encode the following as one-hot vectors: speaker, category, tags and linked entities. We do not encode 'Reason' as it gives away the label, and do not include 'Checker' as there are too many unique checkers for this information to be relevant. The claim publication date is potentially relevant, but it does not make sense to merely model this as a one-hot feature, so we leave incorporating temporal information to future work. Since all metadata consists of individual words and phrases, a sequence encoder is not necessary, and we opt for a CNN followed by a max pooling operation as used in Wang (2017) to encode metadata for fact checking. The max-pooled metadata representations, denoted $h_m$, are then concatenated with the instance representations, e.g. for the most elaborate model, *crawled_ranked*, these would be concatenated with $s_{cr_{i_j}}$.

## 5 Experiments

### 5.1 Experimental Setup

The base sentence embedding model is a BiLSTM over all words in the respective sequences with randomly initialised word embeddings, following Augenstein et al. (2018). We opt for this strong baseline sentence encoding model, as opposed to engineering sentence embeddings that work particularly well for this dataset, to showcase the dataset. We would expect pre-trained contextual encoding models, e.g. ELMO (Peters et al., 2018), ULMFit (Howard and Ruder, 2018), BERT (Devlin et al., 2018), to offer complementary performance gains, as has been shown for a few recent papers (Wang et al., 2018; Rajpurkar et al., 2018).

For claim veracity prediction without evidence documents with the MTL with LEL model, we use the following sentence encoding variants: *claim-*

*only*, which uses a BiLSTM-based sentence embedding as input, and *claim-only_embavg*, which uses a sentence embedding based on mean averaged word embeddings as input.

We train one multi-task model per task (i.e., one model per domain). We perform a grid search over the following hyperparameters, tuned on the respective dev set, and evaluate on the correspoding test set (final settings are underlined): word embedding size [64, 128, 256], BiLSTM hidden layer size [64, 128, 256], number of BiLSTM hidden layers [1, 2, 3], BiLSTM dropout on input and output layers [0.0, 0.1, 0.2, 0.5], word-by-word-attention for BiLSTM with window size 10 (Bahdanau et al., 2014) [True, False], skip-connections for the BiLSTM [True, False], batch size [32, 64, 128], label embedding size [16, 32, 64]. We use ReLU as an activation function for both the BiLSTM and the CNN. For the CNN, the following hyperparameters are used: number filters [32], kernel size [32]. We train using cross-entropy loss and the RMSProp optimiser with initial learning rate of 0.001 and perform early stopping on the dev set with a patience of 3.

### 5.2 Results

For each domain, we compute the Micro as well as Macro F1, then mean average results over all domains. Core results with all vs. no metadata are shown in Table 5. We first experiment with different base model variants and find that label embeddings improve results, and that the best proposed models utilising multiple domains outperform single-task models (see Table 8). This corroborates the findings of Augenstein et al. (2018). Per-domain results with the best model are shown in Table 6. Domain names are from hereon after abbreviated for brevity, see Table 11 in the appendix for correspondences to full website names. Unsurprisingly, it is hard to achieve a high Macro F1 for domains with many labels, e.g. tron and snes. Further, some domains, surprisingly mostly with small numbers of instances, seem to be very easy – a perfect Micro and Macro F1 score of 1.0 is achieved on ranz, bove, buca, fani and thal. We find that for those domains, the verdict is often already revealed as part of the claim using explicit wording.

**Claim-Only vs. Evidence-Based Veracity Prediction.** Our evidence-based claim veracity prediction models outperform claim-only veracity prediction models by a large margin. Unsurprisingly, *claim-only_embavg* is outperformed by *claim-only*. Further, *crawled_ranked* is our best-performing model in terms of Micro F1 and Macro F1, meaning that our model captures that not every piece of evidence is equally important, and can

| Domain | # Insts | # Labs | Micro F1 | Macro F1 |
|---|---|---|---|---|
| ranz | 21 | 2 | 1.000 | 1.000 |
| bove | 295 | 2 | 1.000 | 1.000 |
| abbc | 436 | 3 | 0.463 | 0.453 |
| huca | 34 | 3 | 1.000 | 1.000 |
| mpws | 47 | 3 | 0.667 | 0.583 |
| peck | 65 | 3 | 0.667 | 0.472 |
| faan | 111 | 3 | 0.682 | 0.679 |
| clck | 38 | 3 | 0.833 | 0.619 |
| fani | 20 | 3 | 1.000 | 1.000 |
| chct | 355 | 4 | 0.550 | 0.513 |
| obry | 59 | 4 | 0.417 | 0.268 |
| vees | 504 | 4 | 0.721 | 0.425 |
| faly | 111 | 5 | 0.278 | 0.5 |
| goop | 2943 | 6 | 0.822 | 0.387 |
| pose | 1361 | 6 | 0.438 | 0.328 |
| thet | 79 | 6 | 0.55 | 0.37 |
| thal | 163 | 7 | 1.000 | 1.000 |
| afck | 433 | 7 | 0.357 | 0.259 |
| hoer | 1310 | 7 | 0.694 | 0.549 |
| para | 222 | 7 | 0.375 | 0.311 |
| wast | 201 | 7 | 0.344 | 0.214 |
| vogo | 654 | 8 | 0.594 | 0.297 |
| pomt | 15390 | 9 | 0.321 | 0.276 |
| snes | 6455 | 12 | 0.551 | 0.097 |
| farg | 485 | 11 | 0.500 | 0.140 |
| tron | 3423 | 27 | 0.429 | 0.046 |
| avg | | 7.17 | 0.625 | 0.492 |

Table 6: Total number of instances and unique labels per domain, as well as per-domain results with model *crawled_ranked* + *meta*, sorted by label size

| Metadata | Micro F1 | Macro F1 |
|---|---|---|
| None | **0.627** | 0.441 |
| Speaker | 0.602 | 0.435 |
| + Tags | 0.608 | 0.460 |
| Tags | 0.585 | 0.461 |
| Entity | 0.569 | 0.427 |
| + Speaker | 0.607 | 0.477 |
| + Tags | 0.625 | **0.492** |

Table 7: Ablation results with base model *crawled_ranked* for different types of metadata

| Model | Micro F1 | Macro F1 |
|---|---|---|
| STL | 0.527 | 0.388 |
| MTL | 0.556 | 0.448 |
| MTL + LEL | **0.625** | **0.492** |

Table 8: Ablation results with *crawled_ranked* + *meta* encoding for STL vs. MTL vs. MTL + LEL training

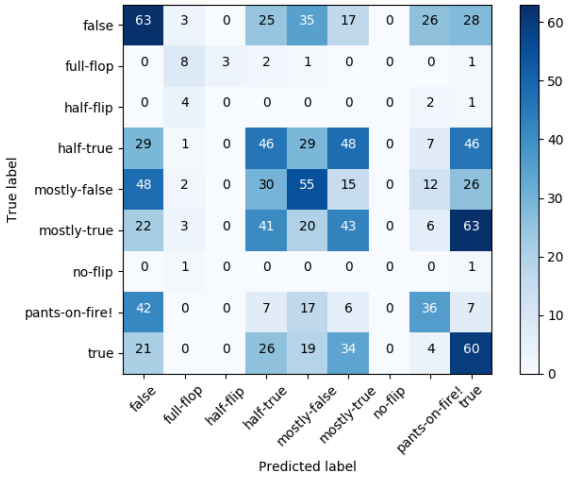


Figure 3: Confusion matrix of predicted labels with best-performing model, *crawled_ranked + meta*, on the 'pomt' domain

utilise this for veracity prediction.

**Metadata.** We perform an ablation analysis of how metadata impacts results, shown in Table 7. Out of the different types of metadata, topic tags on their own contribute the most. This is likely because they offer highly complementary information to the claim text of evidence pages. Only using all metadata together achieves a higher Macro F1 at similar Micro F1 than using no metadata at all. To further investigate this, we split the test set into those instances for which no metadata is available vs. those for which metadata is available. We find that encoding metadata within the model hurts performance for domains where no metadata is available, but improves performance where it is. In practice, an ensemble of both types of models would be sensible, as well as exploring more involved methods of encoding metadata.

## 6 Analysis and Discussion

An analysis of labels frequently confused with one another, for the largest domain 'pomt' and best-performing model *crawled_ranked + meta* is shown in Figure 3. The diagonal represents when gold and predicted labels match, and the numbers signify the number of test instances. One can observe that the model struggles more to detect claims with labels 'true' than those with label 'false'. Generally, many confusions occur over close labels, e.g. 'half-true' vs. 'mostly true'.

We further analyse what properties instances that are predicted correctly vs. incorrectly have, using the model *crawled_ranked meta*. We find that, unsurprisingly, longer claims are harder to classify correctly, and that claims with a high direct token overlap with evidence pages lead to a high evidence ranking. When it comes to frequently occurring tags and entities, very general tags such as 'government-and-politics' or 'tax' that do not give away much, frequently co-occur with incorrect predictions, whereas more specific tags such as 'brisbane-4000' or 'hong-kong' tend to co-occur with correct predictions. Similar trends are observed for bigrams. This means that the model has an easy time succeeding for instances where the claims are short, where specific topics tend to co-occur with certain veracities, and where evidence documents are highly informative. Instances with longer, more complex claims where evidence is ambiguous remain challenging.

## 7 Conclusions

We present a new, real-world fact checking dataset, currently the largest of its kind. It consists of 34,918 claims collected from 26 fact checking websites, rich metadata and 10 retrieved evidence pages per claim. We find that encoding the metadata as well evidence pages helps, and introduce a new joint model for ranking evidence pages and predicting veracity.

## Acknowledgments

This research is partially supported by QUARTZ (721321, EU H2020 MSCA-ITN) and DABAI (5153-00004A, Innovation Fund Denmark).

## References

Isabelle Augenstein, Tim Rocktäschel, Andreas Vlachos, and Kalina Bontcheva. 2016a. Stance Detection with Bidirectional Conditional Encoding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 876–885, Austin, Texas. Association for Computational Linguistics.

Isabelle Augenstein, Sebastian Ruder, and Anders Søgaard. 2018. Multi-Task Learning of Pairwise Sequence Classification Tasks over Disparate Label Spaces. In *NAACL-HLT*, pages 1896–1906. Association for Computational Linguistics.

Isabelle Augenstein, Andreas Vlachos, and Kalina Bontcheva. 2016b. USFD at SemEval-2016 Task 6: Any-Target Stance Detection on Twitter with Autoencoders. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 389–393, San Diego, California. Association for Computational Linguistics.

Joan Bachenko, Eileen Fitzpatrick, and Michael Schonwetter. 2008. Verification and implementation of language-based deception indicators in civil and criminal narratives. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 41–48. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of ICLR*.

Ramy Baly, Mitra Mohtarami, James R. Glass, Lluís Màrquez, Alessandro Moschitti, and Preslav Nakov. 2018. Integrating Stance Detection and Fact Checking in a Unified Corpus. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers)*, pages 21–27. Association for Computational Linguistics.

Alberto Barrn-Cedeo, Tamer Elsayed, Reem Suwaileh, Llus Mrquez, Pepa Atanasova, Wajdi Zaghouani, Spas Kyuchukov, Giovanni Da San Martino, and Preslav Nakov. 2018. Overview of the CLEF-2018 CheckThat! Lab on automatic identification and verification of political claims. Task 2: Factuality. In *CLEF (Working Notes)*, volume 2125 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Rich Caruana. 1993. Multitask Learning: A Knowledge-Based Source of Inductive Bias. In *Proceedings of ICML*.

Sihao Chen, Daniel Khashabi, Wenpeng Yin, Chris Callison-Burch, and Dan Roth. 2019. Seeing Things from a Different Angle: Discovering Diverse Perspectives about Claims. In *Proceedings of NAACL*.

G L Ciampaglia, P Shiralkar, L M Rocha, J Bollen, F Menczer, and A Flammini. 2015. Computational Fact Checking from Knowledge Networks. *PLoS One*, 10(6).

Leon Derczynski, Kalina Bontcheva, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Arkaitz Zubiaga. 2017. SemEval-2017 Task 8: RumourEval: Determining rumour veracity and support for rumours. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 69–76. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR*, abs/1810.04805.

Omar Enayet and Samhaa R. El-Beltagy. 2017. NileTMRG at SemEval-2017 Task 8: Determining Rumour and Veracity Support for Rumours on Twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 470–474. Association for Computational Linguistics.

William Ferreira and Andreas Vlachos. 2016. Emergent: a novel data-set for stance classification. In *HLT-NAACL*, pages 1163–1168. The Association for Computational Linguistics.

Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. In *ACL (1)*, pages 328–339. Association for Computational Linguistics.

Georgi Karadzhov, Pepa Gencheva, Preslav Nakov, and Ivan Koychev. 2017. We Built a Fake News / Click Bait Filter: What Happened Next Will Blow Your Mind! In *RANLP 2017*, pages 334–343.

Elena Kochkina, Maria Liakata, and Isabelle Augenstein. 2017. Turing at SemEval-2017 Task 8: Sequential Approach to Rumour Stance Classification with Branch-LSTM. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 475–480, Vancouver, Canada. Association for Computational Linguistics.

Nikolaos Kolitsas, Octavian-Eugen Ganea, and Thomas Hofmann. 2018. End-to-End Neural Entity Linking. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 519–529. Association for Computational Linguistics.

Rada Mihalcea and Carlo Strapparava. 2009. The lie detector: Explorations in the automatic recognition of deceptive language. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 309–312. Association for Computational Linguistics.

Tsvetomila Mihaylova, Preslav Nakov, Lluís Màrquez, Alberto Barrón-Cedeño, Mitra Mohtarami, Georgi Karadzhov, and James R. Glass. 2018. Fact Checking in Community Forums. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*. AAAI Press.

Tanushree Mitra and Eric Gilbert. 2015. Credbank: A large-scale social media corpus with associated credibility annotations. In *ICWSM*, pages 258–267.

Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Natural Language Inference by Tree-Based Convolution and Heuristic Matching. In *ACL (2)*. The Association for Computer Linguistics.

Verónica Pérez-Rosas, Bennett Kleinberg, Alexandra Lefevre, and Rada Mihalcea. 2018. Automatic Detection of Fake News. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3391–3401. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *NAACL-HLT*, pages 2227–2237. Association for Computational Linguistics.

Dean Pomerleau and Delip Rao. 2017. The Fake News Challenge: Exploring how artificial intelligence technologies could be leveraged to combat fake news. http://www.fakenewschallenge.org/. Accessed: 2019-02-14.

Kashyap Popat, Subhabrata Mukherjee, Jannik Strötgen, and Gerhard Weikum. 2016. Credibility Assessment of Textual Claims on the Web. In *CIKM*, pages 2173–2178.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know What You Don't Know: Unanswerable Questions for SQuAD. In *ACL (2)*, pages 784–789. Association for Computational Linguistics.

Benjamin Riedel, Isabelle Augenstein, Georgios P. Spithourakis, and Sebastian Riedel. 2017. A simple but tough-to-beat baseline for the Fake News Challenge stance detection task. *CoRR*, abs/1707.03264.

Victoria Rubin, Niall Conroy, Yimin Chen, and Sarah Cornwell. 2016. Fake News or Truth? Using Satirical Cues to Detect Potentially Misleading News. In *Proceedings of the Second Workshop on Computational Approaches to Deception Detection*, pages 7–17. Association for Computational Linguistics.

K. Shu, D. Mahudeswaran, S. Wang, D. Lee, and H. Liu. 2018. FakeNewsNet: A Data Repository with News Content, Social Context and Dynamic Information for Studying Fake News on Social Media. *ArXiv e-prints*.

Eugenio Tacchini, Gabriele Ballarin, Marco L. Della Vedova, Stefano Moret, and Luca de Alfaro. 2017. Some Like it Hoax: Automated Fake News Detection in Social Networks. In *Proceedings of the Second Workshop on Data Science for Social Good (SoGood)*, volume 1960 of *CEUR Workshop Proceedings*.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a Large-scale Dataset for Fact Extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana. Association for Computational Linguistics.

Andreas Vlachos and Sebastian Riedel. 2014. Fact Checking: Task definition and dataset construction. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*, pages 18–22. Association for Computational Linguistics.

Soroush Vosoughi, Deb Roy, and Sinan Aral. 2018. The spread of true and false news online. *Science*, 359(6380):1146–1151.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *BlackboxNLP@EMNLP*, pages 353–355. Association for Computational Linguistics.

William Yang Wang. 2017. "Liar, Liar Pants on Fire": A New Benchmark Dataset for Fake News Detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 422–426. Association for Computational Linguistics.

Wenpeng Yin and Dan Roth. 2018. TwoWingOS: A Two-Wing Optimization Strategy for Evidential Claim Verification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 105–114, Brussels, Belgium. Association for Computational Linguistics.

Arkaitz Zubiaga, Elena Kochkina, Maria Liakata, Rob Procter, Michal Lukasik, Kalina Bontcheva, Trevor Cohn, and Isabelle Augenstein. 2018. Discourse-aware rumour stance classification in social media using sequential classifiers. *Informatino Processing & Management*, 54(2):273–290.

Arkaitz Zubiaga, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Peter Tolmie. 2016. Analysing how people orient to and spread rumours in social media by looking at conversational threads. *PLOS ONE*, 11(3):1–29.

| Websites (Sources) | Reason |
|---|---|
| Mediabiasfactcheck | Website that checks other news websites |
| CBC | No pattern to crawl |
| apnews.com/APFactCheck | No categorical label and no structured claim |
| weeklystandard.com/tag/fact-check | Mostly no label, and they are placed anywhere |
| ballotpedia.org | No categorical label and no structured claim |
| channel3000.com/news/politics/reality-check | No categorical label, lack of structure, and no clear claim |
| npr.org/sections/politics-fact-check | No label and no clear claim (only some titles are claims) |
| dailycaller.com/buzz/check-your-fact | Is a subset of checkyourfact which has already been crawled |
| sacbee.com[6] | Contains very few labelled articles, and without clear claims |
| TheGuardian | Only a few websites have a pattern for labels. |

Table 9: The list of websites that we did not crawl and reasons for not crawling them.

| Domain | # Insts | # Labels | Labels |
|---|---|---|---|
| abbc | 436 | 3 | in-between, in-the-red, in-the-green |
| afck | 433 | 7 | correct, incorrect, mostly-correct, unproven, misleading, understated, exaggerated |
| bove | 295 | 2 | none, rating: false |
| chct | 355 | 4 | verdict: true, verdict: false, verdict: unsubstantiated, none |
| clck | 38 | 3 | incorrect, unsupported, misleading |
| faan | 111 | 3 | factscan score: false, factscan score: true, factscan score: misleading |
| faly | 71 | 5 | true, none, partly true, unverified, false |
| fani | 20 | 3 | conclusion: accurate, conclusion: false, conclusion: unclear |
| farg | 485 | 11 | false, none, distorts the facts, misleading, spins the facts, no evidence, not the whole story, unsupported, cherry picks, exaggerates, out of context |
| goop | 2943 | 6 | 0, 1, 2, 3, 4, 10 |
| hoer | 1310 | 7 | facebook scams, true messages, bogus warning, statirical reports, fake news, unsubstantiated messages, misleading recommendations |
| huca | 34 | 3 | a lot of baloney, a little baloney, some baloney |
| mpws | 47 | 3 | accurate, false, misleading |
| obry | 59 | 4 | mostly_true, verified, unobservable, mostly_false |
| para | 222 | 7 | mostly false, mostly true, half-true, false, true, pants on fire!, half flip |
| peck | 65 | 3 | false, true, partially true |
| pomt | 15390 | 9 | half-true, false, mostly true, mostly false, true, pants on fire!, full flop, half flip, no flip |
| pose | 1361 | 6 | promise kept, promise broken, compromise, in the works, not yet rated, stalled |
| ranz | 21 | 2 | fact, fiction |
| snes | 6455 | 12 | false, true, mixture, unproven, mostly false, mostly true, miscaptioned, legend, outdated, misattributed, scam, correct attribution |
| thet | 79 | 6 | none, mostly false, mostly true, half true, false, true |
| thal | 74 | 2 | none, we rate this claim false |
| tron | 3423 | 27 | fiction!, truth!, unproven!, truth! & fiction!, mostly fiction!, none, disputed!, truth! & misleading!, authorship confirmed!, mostly truth!, incorrect attribution!, scam!, investigation pending!, confirmed authorship!, commentary!, previously truth! now resolved!, outdated!, truth! & outdated!, virus!, fiction! & satire!, truth! & unproven!, misleading!, grass roots movement!, opinion!, correct attribution!, truth! & disputed!, inaccurate attribution! |
| vees | 504 | 4 | none, fake, misleading, false |
| vogo | 653 | 8 | none, determination: false, determination: true, determination: mostly true, determination: misleading, determination: barely true, determination: huckster propaganda, determination: false, determination: a stretch |
| wast | 201 | 7 | 4 pinnochios, 3 pinnochios, 2 pinnochios, false, not the whole story, needs context, none |

Table 10: Number of instances, and labels per domain sorted by number of occurrences

| Website | Domain | Claims | Labels | Category | Speaker | Checker | Tags | Article | Claim date | Publish date | Full text | Outlinks |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| abc | abbc | 436 | 436 | 436 | - | - | 436 | 436 | - | 436 | 436 | 7676 |
| africacheck | afck | 436 | 436 | - | - | - | - | 436 | - | 436 | 436 | 2325 |
| altnews | - | 496 | - | - | - | 496 | - | 496 | - | 496 | 496 | 6389 |
| boomlive | - | 302 | 302 | - | - | - | - | 302 | - | 302 | 302 | 6054 |
| checkyourfact | chht | 358 | 358 | - | - | 358 | - | - | - | 358 | 358 | 5271 |
| climatefeedback | clck | 45 | 45 | - | - | - | - | 45 | - | 45 | 45 | 489 |
| crikey | - | 18 | 18 | 18 | - | 18 | 18 | 18 | - | 18 | 18 | 212 |
| factcheckni | - | 36 | 36 | 36 | - | - | - | 36 | - | - | 36 | 151 |
| factcheckorg | farg | 512 | 512 | 512 | 512 | 512 | 512 | 512 | 512 | 512 | 512 | 8282 |
| factly | - | 77 | 77 | - | - | - | - | 77 | - | - | 77 | 658 |
| factscan | - | 115 | 115 | - | 115 | - | - | - | 115 | 115 | 115 | 1138 |
| fullfact | - | 336 | 336 | 336 | - | 336 | - | 336 | - | 336 | 336 | 3838 |
| gossipcop | goop | 2947 | 2947 | - | - | 2947 | - | 2947 | - | 2947 | 2947 | 12583 |
| hoaxslayer | hoer | 1310 | 1310 | - | - | 1310 | - | 1310 | - | 1310 | 1310 | 14499 |
| huffingtonpostca | huca | 38 | 38 | - | 38 | 38 | - | 38 | 38 | 38 | 38 | 78 |
| leadstories | - | 1547 | 1547 | - | - | 1547 | - | 1547 | - | 1547 | 1547 | 12015 |
| mprnews | mpws | 49 | 49 | - | - | 49 | - | 49 | - | 49 | 49 | 319 |
| nytimes | - | 17 | 17 | - | - | 17 | - | 17 | - | 17 | 17 | 271 |
| observatory | obry | 60 | 60 | - | - | 60 | - | 60 | - | 60 | 60 | 592 |
| pandora | para | 225 | 225 | 225 | 225 | 225 | - | 225 | - | 225 | 225 | 114 |
| pesacheck | peck | 67 | 67 | - | - | 67 | - | 67 | - | 67 | 67 | 521 |
| politico | - | 102 | 102 | - | - | 102 | - | 102 | - | 102 | 102 | 150 |
| politifact_promise | pose | 1361 | 1361 | 1361 | 1361 | - | - | 1361 | - | 1361 | 1361 | 6279 |
| politifact_stmt | pomt | 15390 | 15390 | - | 15390 | - | - | - | 15390 | 15390 | 15390 | 78543 |
| politifact_story | - | 5460 | - | - | - | 5460 | - | - | - | 5460 | 5460 | 24836 |
| radionz | ranz | 32 | 32 | 32 | 32 | - | - | 32 | 32 | 32 | 32 | 44 |
| snopes | snes | 6457 | 6457 | 6457 | - | 6457 | - | 6457 | - | 6457 | 6457 | 46735 |
| swissinfo | - | 20 | 20 | 20 | 20 | 20 | - | 20 | - | 20 | 20 | 40 |
| theconversation | - | 62 | 62 | 62 | 62 | 62 | 62 | 62 | - | 62 | 62 | 723 |
| theferret | thet | 81 | 81 | 81 | 81) | - | - | 81 | - | 81(81) | 81 | 885 |
| theguardian | - | 155 | 155 | 155 | - | 155 | - | 155 | - | 155 | 155 | 2600 |
| thejournal | thal | 179 | 179 | - | - | - | - | 179 | - | 179 | 179 | 2375 |
| truthorfiction | tron | 3674 | 3674 | 3674 | - | - | 3674 | 3674 | - | 3674 | 3674 | 8268 |
| verafiles | vees | 509 | 509 | - | - | - | 509 | 509 | - | 509 | 509 | 23 |
| voiceofsandiego | vogo | 660 | 660 | - | - | - | - | 660 | - | 660 | 660 | 2352 |
| washingtonpost | wast | 227 | 227 | - | 227 | 227 | - | 227 | - | 227 | 227 | 2470 |
| wral | - | 20 | 20 | - | - | 20 | 20 | 20 | - | 20 | 20 | 355 |
| zimfact | - | 21 | 21 | 21 | 21 | 21 | - | 21 | - | 21 | 21 | 179 |
| Total | | 43837 | 43837 | 43837 | 43837 | 43837 | 43837 | 43837 | 43837 | 43837 | 43837 | 260330 |

Table 11: Summary statistics for claim collection. 'Domain' indicates the domain name used for the veracity prediction experiments, '–' indicates that the website was not used due to missing or insufficient claim labels, see Section 3.2.

# A Deep Neural Information Fusion Architecture for Textual Network Embeddings

**Zenan Xu**[1,2], **Qinliang Su**[1,2]*, **Xiaojun Quan**[1], **Weijia Zhang**[3]
[1]School of Data and Computer Science, Sun Yat-sen University
[2]Guangdong Key Laboratory of Big Data Analysis and Processing, Guangzhou, China
[3]School of Electronics and Information Technology, Sun Yat-sen University
xuzn@mail2.sysu.edu.cn, suqliang@mail.sysu.edu.cn
quanxj3@mail.sysu.edu.cn, zhangwj39@mail2.sysu.edu.cn

## Abstract

Textual network embeddings aim to learn a low-dimensional representation for every node in the network so that both the structural and textual information from the networks can be well preserved in the representations. Traditionally, the structural and textual embeddings were learned by models that rarely take the mutual influences between them into account. In this paper, a deep neural architecture is proposed to effectively fuse the two kinds of informations into one representation. The novelties of the proposed architecture are manifested in the aspects of a newly defined objective function, the complementary information fusion method for structural and textual features, and the mutual gate mechanism for textual feature extraction. Experimental results show that the proposed model outperforms the comparing methods on all three datasets.

## 1 Introduction

Networks provide an effective way to organize heterogeneous relevant data, which can often be leveraged to facilitate downstream applications. For example, the huge amount of textual and relationship data in social networks contains abundant information on people's preferences, and thus can be used for personalized advertising and recommendation. To this end, traditionally a matrix representing the network structure is often built first, then subsequent tasks proceed. However, matrix methods are computationally expensive and cannot be applied to large-scale networks.

Network embedding (NE) maps every node of a network into a low-dimensional vector, while seeking to retain the original network information. Subsequent tasks (*e.g.* similar vertices search, linkage prediction) can proceed by leveraging these low-dimensional features. To obtain network embeddings, (Perozzi et al., 2014) proposed to first generate sequences of nodes by randomly walking along connected nodes. Word embedding methods are then employed to produce the embeddings for nodes by noting the analogies between node sequences and sentences in natural languages. Second-order proximity information is further taken into account in (Tang et al., 2015). To gather the network connection information more efficiently, the random walking strategy in (Perozzi et al., 2014) is modified to favor the important nodes in (Grover and Leskovec, 2016). However, all these methods only took the network structure into account, ignoring the huge amount of textual data. In Twitter social network, for example, tweets posted by a user contain valuable information on the user's preferences, political standpoints, and so on (Bandyopadhyay et al., 2018).

To include textual information into the embeddings, (Tu et al., 2017) proposed to first learn embeddings for the textual data and network structure respectively, and then concatenate them to obtain the embeddings of nodes. The textual and structural embeddings are learned with an objective that encourages embeddings of neighboring nodes to be as similar as possible. Attention mechanism is further employed to highlight the important textual information by taking the impacts of texts from neighboring nodes into account. Later, (Shen et al., 2018b) proposed to use the fine-grained word alignment mechanism to replace the attention mechanism in (Tu et al., 2017) in order to absorb the impacts from neighboring texts more effectively. However, both methods require the textual and structural embeddings from neighboring nodes to be as close as possible even if the nodes share little common contents. This could be problematic since a social network user may be connected to users who post totally different viewpoints because of different political standpoints. If

*Corresponding author.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4698–4706, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

two nodes are similar, it is the node embeddings, rather than the individual textual or structural embeddings, that should be close. Forcing representations of dissimilar data to be close is prone to yield bad representations. Moreover, since the structural and textual embeddings contain some common information, if they are concatenated directly, as done in (Tu et al., 2017; Shen et al., 2018b), the information contained in the two parts is entangled in some very complicated way, increasing the difficulties of learning representative network embeddings.

In this paper, we propose a novel deep neural **I**nformation **F**usion **A**rchitecture for textual **N**etwork **E**mbedding (NEIFA) to tackle the issues mentioned above. Instead of forcing the separate embeddings of structures and texts from neighboring nodes to be close, we define the learning objective based on the node embeddings directly. For the problem of information entanglement, inspired by the gating mechanism of long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997), we extract the complementary informations from texts and structures and then use them to constitute the node embeddings. A mutual gate is further designed to highlight the node's textual information that is consistent with neighbors' textual contents, while diminishing those that contracdict to each other. In this way, the model provides a mechanism to only allow the information that is consistent among neighboring nodes to flow into the node embeddings. The proposed network embedding method is evaluated on the tasks of link prediction and vertex classification, using three real-world datasets from different domains. It is shown that the proposed method outperforms state-of-the-art network embedding methods on the task of link prediction by a substantial margin, demonstrating that the obtained embeddings well retain the information in original networks. Similar phenomenons can also be observed in the vertex classification task. These results suggest the effectiveness of the proposed neural information fusion architecture for textual network embeddings.

## 2 Related Work

**Text Embedding** There has been various methods to embed textual information into vector representations for NLP tasks. The classical method for embedding textual information could be one-hot vector, term frequency inverse document frequency (TF-IDF), etc. Due to the high-dimension and sparsity problems in here, (Mikolov et al., 2013a) proposed a novel neural network based skip-gram model to learn distributed word embeddings via word co-occurrences in a local window of textual content. To exploit the internal structure of text, convolutional neural networks (CNNs) (Blunsom et al., 2014; Kim, 2014) is applied to obtain latent features of local textual content. Then, by following a pooling layer, fixed-length representations are generated. To have the embeddings better reflect the correlations among texts, soft attention mechanisms (Bahdanau et al., 2014; Vaswani et al., 2017) is proposed to calculate the relative importances of words in a sentence by evaluating their relevances to the content of comparing sentences. Alternatively, gating mechanism is applied to strengthen the relevant textual information, while weakening the irrelevant one by controlling the information-flow path of a network in (Dauphin et al., 2017; Zhou et al., 2017).

**Network Embedding** Network embedding methods can be categorized into two classes: (1) methods that solely utilize structure information; and (2) methods that consider both structure and textual content associated with vertices. For the first type of methods, DeepWalk (Perozzi et al., 2014) was the first to introduce neural network technique into the network embedding field. In DeepWalk, node sequences are generated via randomly walking on the network, dense latent representations are by feeding those node sequences into the skip-gram model. LINE (Tang et al., 2015) exploited the first-order and second-order proximity information of vertices in network by optimizing the joint and condition probability of edges. Further, Node2Vec (Grover and Leskovec, 2016) proposed a biased random walk to search a network and generate node sequences based on the depth-first search and width-first search. However, those methods only embed the structure information into vector representations, while ignoring the informative textual contents associated with vertices. To address this issue, some recent work seeks to the joint impact of structure and textual contents to obtain better representations. TADW (Yang et al., 2015) proved that DeepWalk is equivalent to the matrix factorization and the textual information can be incorporated by simply adding the textual feature into the matrix factorization. CENE (Sun et al., 2016) extends the original net-

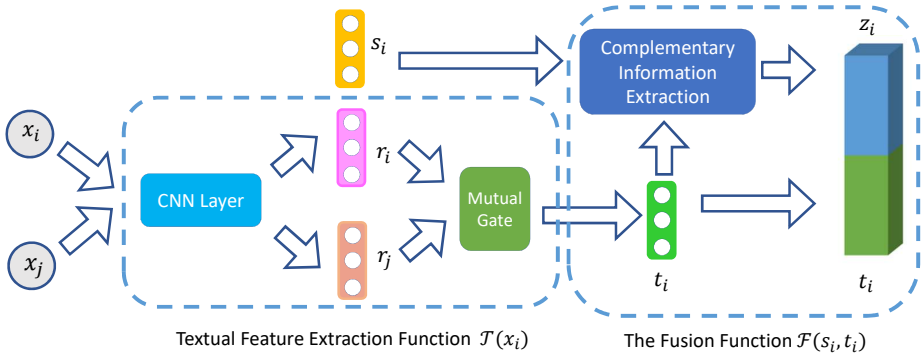


Figure 1: The overall framework of NEIFA.

works by transforming the textual content into another kinds of vertices, and the vertices are embedded into low-dimensional representations on the extended network. CANE (Tu et al., 2017) proposed to learn separate embeddings for the textual and structural information, and obtain the network embeddings by simply concatenating them, in which a mutual attention mechanism is used to model the semantic relationship between textual contents. WANE (Shen et al., 2018b) modified the semantic extraction strategy in CANE by introducing a fine-grained word alignment technique to learn word-level semantic information more effectively. However, most of recent methods force the textual and structural embeddings of two neighboring nodes close to each other irrespective of their underlying contents.

## 3 The Proposed Method

A textual network is defined as $G = \{V, E, T\}$, where $V$, $E$ and $T$ denote the vertices in the graph, edges between vertices and textual content associated with vertices, respectively. Each edge $e_{i,j} \in E$ suggests there is a relationship between vertex $v_i$ and $v_j$.

### 3.1 Training Objective

Suppose the structural and textual features of node $i$ are given and are denoted as $s_i$ and $t_i$, respectively. Existing methods are built on the objectives that encourage the structural and textual features of neighboring nodes to be as similar as possible. As discussed in the previous sections, this may make the node embeddings deviate from the true information in the nodes. In this paper, we define the objective based on the node embeddings directly, that is,

$$\mathcal{L} = \sum_{\{i,j\} \in E} \log p(h_i|h_j), \tag{1}$$

where $h_i$ is the network embedding of node $i$, and is constructed from $s_i$ and $t_i$ by

$$h_i = \mathcal{F}(s_i, t_i); \tag{2}$$

$\mathcal{F}(\cdot, \cdot)$ is the fusion function that maps the structural and textual features into the network embeddings; and $p(h_i|h_j)$ denotes the conditional probability of network embedding $h_i$ given the network embedding $h_j$. Following LINE (Tang et al., 2015), the conditional probability in (1) is defined as:

$$p(h_i|h_j) = \frac{\exp(h_i \cdot h_j)}{\sum_{z \in V} \exp(h_i \cdot h_z)}. \tag{3}$$

Note that the structural feature $s_i$ is randomly initialized and will be learned along with the other model parameters. For the textual feature $t_i$, it is obtained via a trainable feature extraction function from the given texts, i.e.,

$$t_i = \mathcal{T}(x_i), \tag{4}$$

where $x_i$ represents the texts associated with node $i$. From the definition of objective function (1), it can be seen that it is the network embeddings of nodes, rather than the individual structural or textual embeddings, that are encouraged to be close for neighboring nodes.

Details on how to realize the fusion function $\mathcal{F}(\cdot, \cdot)$ and feature extraction function $\mathcal{T}(\cdot)$ are deferred to Section 3.2 and Section 3.3, respectively. The overall framework of our proposed NEIFA is shown in Fig.1.

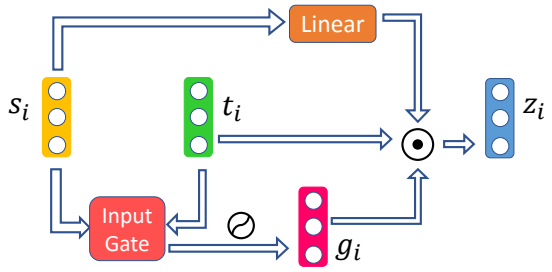


Figure 2: The Fusion Module.

### 3.2 Fusion of Structural and Textual Features $\mathcal{F}(s_i, t_i)$

In this section, we will present how to fuse the structural and textual features to yield the embeddings for nodes. The fusion module $\mathcal{F}(s_i, t_i)$ is illustrated in Fig.2. The simplest way to obtain network embeddings is to concatenate them directly, *i.e.* $h_i = [s_i; t_i]$. However, it is known that the structural and textual features are not fully exclusive, and often contain some common information. Thus, if the network embeddings are generated by simply concatenating the two features, different parts of the embeddings become entangled to each other in some unknown but complex way. This may make the process of optimizing the objective function more difficult and hinder the model to learn representative embeddings for the nodes. In this paper, we instead distill the information that is complementary to the textual feature $t_i$ from $s_i$ first, and then concatenate the two complementary information to constitute the embeddings of nodes.

To distill the complementary information from the structural feature $s_i$, inspired by LSTM, an input gate is designed to eliminate the information in $s_i$ that has already appeared in $t_i$. Specifically, the gate is designed as

$$g_i = 1 - \sigma((\mathbf{P}s_i + b_g) \odot t_i), \tag{5}$$

where $\sigma(\cdot)$ is the sigmoid function; $\odot$ denotes the element-wise multiplication; $\mathbf{P}$ and $b_g$ are used to align the structural feature $s_i$ to the space of textual features $t_i$. From the definition of $g_i$, it can be seen that if the values on some specific dimension of $\mathbf{P}s_i + b_g$ and $t_i$ are both large, which indicates the same information appears in both $s_i$ and $t_i$, the gate $g_i$ will be closed. So, if $(\mathbf{P}s_i + b_g) \odot t_i$ is multiplied to the gate $g_i$, only the information that is not contained in both $s_i$ and $t_i$ is allowed to pass through. Thus, $((\mathbf{P}s_i + b_g) \odot t_i) \odot g_i$ can be understood as the information in $s_i$ that is complementary to $t_i$. In practice, we untie the values of $\mathbf{P}$ and $b_g$, and use a new trainable matrix $\mathbf{Q}$ and bias $b_c$ instead. The complementary information is eventually computed as

$$z_i = ((\mathbf{Q}s_i + b_c) \odot t_i) \odot g_i. \tag{6}$$

Then, we concatenate complementary information $z_i$ to the textual features $t_i$ to produce the final network embedding

$$h_i = [z_i; t_i]. \tag{7}$$

In this way, given the structural and textual features $s_i$ and $t_i$, we successfully extract the complementary information from $s_i$ and generate the final network embedding $h_i$.

### 3.3 Textual Feature Extraction $\mathcal{T}(x_i)$

When extracting textual features for the embeddings of nodes, the impacts from neighboring nodes should also be taken into account, *i.e.* highlighting the consistent information, while dampening the inconsistent ones. To this end, we first repersent words with their corresponding embeddings, and then apply a one-layer CNN followed by an average pooling operator to extract the raw features for texts (Tu et al., 2017; Shen et al., 2018a). Given the raw textual features $r_i$ and $r_j$ of two neighboring nodes $i$ and $j$, we diminish the information that are not consistent in the two raw features. Specifically, the final textual features are computed for nodes $i$ and $j$ as

$$t_i = r_i \odot \sigma(r_j), \tag{8}$$

$$t_j = r_j \odot \sigma(r_i), \tag{9}$$

where $\sigma(\cdot)$ serves as the role of gating. Since the raw textual feature $r_i$ often exhibits specific meanings on different dimensions, the expressions (8) and (9) can be understood as a way to control which information is allowed to flow into the embeddings. Only the information that is consitent among neighboring nodes can appear in the textual feature $t_i$, which is then fused into the network embeddings. There are a variety of other nonlinear functions that can serve as the role of gating, but in this work, the simplest but effective sigmoid function is employed.

### 3.4 Training Details

Maximizing the objective function in (1) requires to compute the expensive softmax function repeatedly, in which the summation over all nodes of the networks is needed for iteration. To address this issue, for each edge $e_{i,j} \in E$, we introduce negative sampling (Mikolov et al., 2013b) to simplify the optimization process. Therefore, the conditional distribution $p(h_i|h_j)$ into the following form:

$$\log \sigma(h_i \cdot h_j) + \sum_{k=1}^{K} E_{h_k \sim P(h)}[\log \sigma(-h_k \cdot h_i)] \tag{10}$$

where $K$ is the number of negative samples, and $P(v) \propto d_v^{3/4}$ is the distribution of vertices with $d_v$ representing the out-degree of vertex $v$. Adam (Kingma and Ba, 2014) is employed to optimize the entire model based on randomly mini-batch of edges in each step.

## 4 Experiments

To evaluate the quality of the network embeddings generated by the proposed method, we apply them in two tasks: link prediction and vertex classification. Link prediction aims to predict whether there exists a link between two randomly chosen nodes based on the similarities of embeddings of the two nodes. Vertex classification, on the other hand, tries to classify the nodes into different categories based on the embeddings, provided that there exists some supervised information. Both tasks can achieve good performances only when the embeddings retain important information of the nodes, including both the structural and textual information. In the following, we will first introduce the datasets and baselines used in this paper, then describe the evaluation metric and experimental setups, and lastly report the performance of the proposed model on the tasks of link prediction and vertex classification, respectively.

### 4.1 Datasets and Baselines

Experiments are conducted on three real-world datasets: Zhihu (Sun et al., 2016), Cora (McCallum et al., 2000) and HepTh (Leskovec et al., 2005). Below shows the detailed descriptions of the three datasets, with their statistics summaries given in Table 1. The preprocessing procedure of the above datasets is the same as that in (Tu et al., 2017) [1].

| Datasets | Zhihu | Cora | HepTh |
|---|---|---|---|
| Vertices | 10000 | 2277 | 1038 |
| Edges | 43894 | 5214 | 1990 |
| #(Edges) | 2191 | 93 | 24 |
| *(Text) | 190 | 90 | 54 |
| Labels | - | 7 | - |

Table 1: Statistics of datasets, where #(Edges) denotes the max number of the connective relationship of a node, and *(Text) denotes the average lengths of the text.

- *Zhihu* (Sun et al., 2016) is a Q&A based community social network. In our experiment, 10000 active users and the descriptions of their interested topics are collected as the vertices and texts of the social network to be studied. There are total 43894 edges which indicate the relationship between active users.
- *Cora* (McCallum et al., 2000) is a citation network that consists of 2277 machine learning papers with text contents divided into 7 categories. The citation relations among the papers are reflected in the 5214 edges.
- *HepTh* (Leskovec et al., 2005) (High Energy Physics Theory) is a citation network from the e-print arXiv. In our experiment, 1038 papers with abstract information are collected, among which 1990 edges are observed.

To evaluate the effectiveness of our proposed model, several strong baseline methods are compared with, which are divided into two categories as follows:

- *Structure-only:* DeepWalk (Perozzi et al., 2014), LINE (Tang et al., 2015), Node2vec (Grover and Leskovec, 2016).
- *Structure and Text:* TADW (Yang et al., 2015), CENE (Sun et al., 2016), CANE (Tu et al., 2017), WANE (Shen et al., 2018b).

### 4.2 Evaluation Metrics and Experimental Setups

In link prediction, the performance criteria of area under the curve (AUC) (Hanley and McNeil,

[1] https://github.com/thunlp/CANE

| % Training Edges | 15% | 25% | 35% | 45 % | 55% | 65% | 75% | 85% | 95% |
|---|---|---|---|---|---|---|---|---|---|
| DeepWalk† | 56.6 | 58.1 | 60.1 | 60.0 | 61.8 | 61.9 | 63.3 | 63.7 | 67.8 |
| LINE† | 52.3 | 55.9 | 59.9 | 60.9 | 64.3 | 66.0 | 67.7 | 69.3 | 71.1 |
| node2vec† | 54.2 | 57.1 | 57.3 | 58.3 | 58.7 | 62.5 | 66.2 | 67.6 | 68.5 |
| TADW† | 52.3 | 54.2 | 55.6 | 57.3 | 60.8 | 62.4 | 65.2 | 63.8 | 69.0 |
| CENE† | 56.2 | 57.4 | 60.3 | 63.0 | 66.3 | 66.0 | 70.2 | 69.8 | 73.8 |
| CANE† | 56.8 | 59.3 | 62.9 | 64.5 | 68.9 | 70.4 | 71.4 | 73.6 | 75.4 |
| WANE‡ | 58.7 | 63.5 | 68.3 | 71.9 | 74.9 | 77.0 | 79.7 | 80.0 | 82.6 |
| NEIFA | **68.9** | **73.7** | **78.3** | **81.0** | **84.5** | **87.3** | **88.2** | **89.6** | **90.1** |

Table 2: AUC scores for link prediction on Zhihu. Note that † and ‡ indicate the results are taken from (Tu et al., 2017) and (Shen et al., 2018b), respectively.

| % Training Edges | 15% | 25% | 35% | 45 % | 55% | 65% | 75% | 85% | 95% |
|---|---|---|---|---|---|---|---|---|---|
| DeepWalk† | 56.0 | 63.0 | 70.2 | 75.5 | 80.1 | 85.2 | 85.3 | 87.8 | 90.3 |
| LINE† | 55.0 | 58.6 | 66.4 | 73.0 | 77.6 | 82.8 | 85.6 | 88.4 | 89.3 |
| node2vec† | 55.9 | 62.4 | 66.1 | 75.0 | 78.7 | 81.6 | 85.9 | 87.3 | 88.2 |
| TADW† | 86.6 | 88.2 | 90.2 | 90.8 | 90.0 | 93.0 | 91.0 | 93.4 | 92.7 |
| CENE† | 72.1 | 86.5 | 84.6 | 88.1 | 89.4 | 89.2 | 93.9 | 95.0 | 95.9 |
| CANE† | 86.8 | 91.5 | 92.2 | 93.9 | 94.6 | 94.9 | 95.6 | 96.6 | 97.7 |
| WANE‡ | **91.7** | **93.3** | 94.1 | 95.7 | 96.2 | 96.9 | 97.5 | 98.2 | 99.1 |
| NEIFA | 89.0 | 92.2 | **95.3** | **96.5** | **97.1** | **97.4** | **97.6** | **98.5** | **99.2** |

Table 3: AUC scores for link prediction on Cora. Note that † and ‡ indicate the results are taken from (Tu et al., 2017) and (Shen et al., 2018b), respectively.

1982) is used, which represents the probability that vertices in a random unobserved link are more similar than those in a random non-existent link.

For the vertex classification task, a logistic regression model is first trained to classify the embeddings into different categories based on the provided labels of nodes. Then, the trained model is used to classify the network embeddings in test set, and the classification accuracy is used as the performance criteria of this task.

To have a fair comparison with competitive methods, the dimension of network embeddings is set to 200 for all considered methods. The number of negative samples is set to 1 and the mini-batch size is set to 64 to speed up the training processes. Adam (Kingma and Ba, 2014) is employed to train our model with a learning rate of $1 \times 10^{-3}$.

### 4.3 Link Prediction

We randomly extract a portion of edges from the whole edges to constitute the training datasets, and use the rest as the test datasets. The AUC scores of different models under proportions ranging from 15% to 95% on Zhihu, Cora and HepTh datasets are shown in Table 2, Table 3 and Table 4, respectively, with the best performance highlighted in bold.

As can be seen from Table 2, our proposed method outperforms all other baselines in Zhihu dataset substantially, with approximately a 10 percent improvement over the current state-of-the-art WANE model. This may be partially attributed to the complicated Zhihu dataset, in which both the structures and texts contain important informations. If the two individual features are concatenated directly, there may be sever information overlapping problem, limiting the models to learning good embeddings. The proposed complementary information fusion method alleviate the issue by disentangling the structural and textual features. In adition, the proposed mutual gate mechanism that removes inconsistent textual information from a node's textual feature also contribute to the performance gains. On the other hand, the substantial gain may also be partially attributed to the objective function that is directly defined on the

| % Training Edges | 15% | 25% | 35% | 45 % | 55% | 65% | 75% | 85% | 95% |
|---|---|---|---|---|---|---|---|---|---|
| DeepWalk† | 55.2 | 66.0 | 70.0 | 75.7 | 81.3 | 83.3 | 87.6 | 88.9 | 88.0 |
| LINE† | 53.7 | 60.4 | 66.5 | 73.9 | 78.5 | 83.8 | 87.5 | 87.7 | 87.6 |
| node2vec† | 57.1 | 63.6 | 69.9 | 76.2 | 84.3 | 87.3 | 88.4 | 89.2 | 89.2 |
| TADW† | 87.0 | 89.5 | 91.8 | 90.8 | 91.1 | 92.6 | 93.5 | 91.9 | 91.7 |
| CENE† | 86.2 | 84.6 | 89.8 | 91.2 | 92.3 | 91.8 | 93.2 | 92.9 | 93.2 |
| CANE† | 90.0 | 91.2 | 92.0 | 93.0 | 94.2 | 94.6 | 95.4 | 95.7 | 96.3 |
| WANE‡ | **92.3** | 94.1 | 95.7 | 96.7 | **97.5** | 97.5 | 97.7 | 98.2 | 98.7 |
| NEIFA | 91.7 | **94.2** | **95.9** | **96.8** | 97.4 | **97.6** | **98.0** | **98.6** | **99.1** |

Table 4: AUC scores for link prediction on HepTh. Note that † and ‡ indicate the results are taken from (Tu et al., 2017) and (Shen et al., 2018b), respectively.

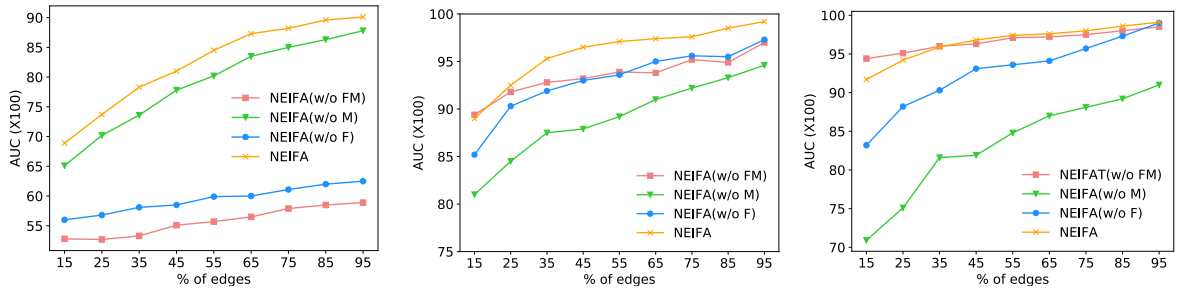


(a) Zhihu (b) Cora (c) HepTh

Figure 3: Ablation study of NEIFA model on different datasets on link prediction task.

network embeddings. That is because the inconsistencies of the structural or textual information among neighboring nodes are more likely to happen in complex networks.

For the other two datasets, as shown in Table 3 and Table 4, our proposed method outperforms baseline methods overall. The results strongly demonstrate that the network embeddings generated by the proposed model are easier to preserve the original information in the nodes. It can be also seen that the performance gains observed in the Cora and HepTh datasets are not as substantial as that in Zhihu dataset. The relatively small improvement may be attributed to the fact that the number of edges and neighbors in Cora and HepTh datasets are much smaller that Zhihu datasets. We speculate that the information in structures of the two datasets is far less than that in texts, implying that the overlapping issue is not as sever as that in Zhihu. Hence, direct concatenation will not induce significant performance loss.

**Ablation Study** To demonstrate the effectiveness of proposed fusion method and mutual gate mechanism, three variants of the proposed model are evaluated: (1)NEIFA(w/o FM): NEIFA without both fusion process and mutual gated mechanism where the raw textual features $r$ are directly regarded as network embeddings. (2) NEIFA(w/o F): NEIFA without fusion process where the textual features $t$ are directly regarded as network embeddings. (3) NEIFA(w/o M): NEIFA without mutual gated mechanism where the network embeddings are obtained by fusing the structural features and raw textual features. The three variants are compared with original NEIFA model on the three datasets above. The results are showed in Fig.3. It can be seen that for networks with very sparse structure, such as Hepth, the method that simply uses the raw textual features as their network embeddings can achieve pretty good performance. In the simple datasets, the proposed model even exhibits worse performance in the case of small proportion of training edges. As the datasets become larger and more complex network structure is included, the performance of only using the textual embeddings decreases rapidly. The reason may be that as the networks grow, the differences of structural or textual data among neighboring nodes become more apparent, and the advantages of the mutual gate mechanism and information fu-

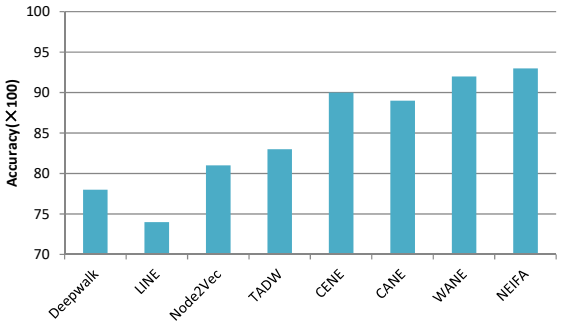


Figure 4: Vertex classification result on Cora dataset

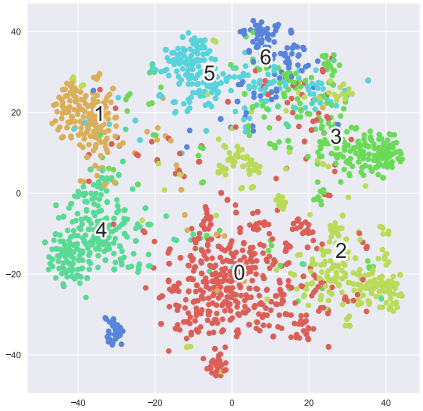


Figure 5: $t-$SNE visualization of our learned network embeddings on Cora dataset.

sion method will show up.

### 4.4 Vertex Classification

To demonstrate the superiority of proposed method, the vertex classification experiment is also considered on the Cora dataset. This experiment is established on the basis that if the original network contains different types of nodes, good embeddings mean that they can be classified into specific classes by a simple classifier easily. For the proposed method, the embedding of a node varies as it interacts with different nodes. To have the embedding fixed, we follow procedures in (Tu et al., 2017) to yield a node's embedding by averaing the embeddings that are obtained when the node interacts with different neighbors.

To this end, we randomly split the node embeddings of all nodes with a proportion of 50%-50% into a training and testing set, respectively. A logistic regression classifier regularized by $L_2$ distance (Fan et al., 2008) is then trained on the node embeddings from training set. The classification performance is tested on the hold-out testing set. The above procedures are repeated 10 times and their average value is reported as the final performance. It can be seen from Fig.4 that methods considering both structural and textual information show better classification accuracies than methods leveraging only structural information, demonstrating the importance of incorporating textual information into the embeddings. Moreover, NEIFA outperforms all methods considered, which further proves the superiority of our proposed model.

To intuitively understand the embeddings produced by the proposed model, we employ $t$-SNE to map our learned embeddings to a 2-D space. The result is shown in Fig.5, where different colors indicate that the nodes belong to different categories. Note that, although the mapping in t-SNE is trained without using any category labels, the latent label information is still partially extracted out. As shown in Fig.5, the points with the same color are closer to each other, while the ones with different colors are far apart.

## 5 Conclusions

In this paper, a novel deep neural architecture is proposed to effectively fuse the structural and textual informations in networks. Unlike existing embeddings methods which encourage both textual and structural embeddings of two neighboring nodes close to each other, we define the training objective based on the node embeddings directly. To address the information duplication problem in the structural and textual features, a complementary information fusing method is further developed to fuse the two features. Besides, a mutual gate is designed to highlight the textual information in a node that is consistent with the textual contents of neighboring nodes, while diminishing those that are conflicting to each other. Exhaustive experimental results on several tasks manifest the advantages of our proposed model.

## Acknowledgments

We thank Wei Liu and Wenxuan Li for their helps. This work is supported by the National Natural Science Foundation of China (NSFC) under Grant No. 61806223, U1711262, U1501252, U1611264 and U1711261, and National Key R&D Program of China (2018YFB1004404).

## References


Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Sambaran Bandyopadhyay, Harsh Kara, Anirban Biswas, and M N Murty. 2018. Sac2vec: Information network representation with structure and content.

Phil Blunsom, Edward Grefenstette, and Nal Kalchbrenner. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Proceedings of the 52nd Annual Meeting of the Association for Computational .

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 933–941. JMLR. org.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874.

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864. ACM.

James A Hanley and Barbara J McNeil. 1982. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2005. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 177–187. ACM.

Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. 2000. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM.

Dinghan Shen, Guoyin Wang, Wenlin Wang, Martin Renqiang Min, Qinliang Su, Yizhe Zhang, Chunyuan Li, Ricardo Henao, and Lawrence Carin. 2018a. Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 440–450.

Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. 2018b. Improved semantic-aware network embedding with fine-grained word alignment. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1829–1838.

Xiaofei Sun, Jiang Guo, Xiao Ding, and Ting Liu. 2016. A general framework for content-enhanced network representation learning. *arXiv preprint arXiv:1610.02906*.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077. International World Wide Web Conferences Steering Committee.

Cunchao Tu, Han Liu, Zhiyuan Liu, and Maosong Sun. 2017. Cane: Context-aware network embedding for relation modeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1722–1731.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang. 2015. Network representation learning with rich text information. In *IJCAI*, pages 2111–2117.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2017. Selective encoding for abstractive sentence summarization. *arXiv preprint arXiv:1704.07073*.

# You Shall Know a User by the Company It Keeps: Dynamic Representations for Social Media Users in NLP

**Marco Del Tredici**
University of Amsterdam
m.deltredici@uva.nl

**Diego Marcheggiani***
Amazon
marchegg@amazon.es

**Sabine Schulte im Walde**
University of Stuttgart
schulte@ims.uni-stuttgart.de

**Raquel Fernández**
University of Amsterdam
raquel.fernandez@uva.nl

## Abstract

Information about individuals can help to better understand what they say, particularly in social media where texts are short. Current approaches to modelling social media users pay attention to their social connections, but exploit this information in a static way, treating all connections uniformly. This ignores the fact, well known in sociolinguistics, that an individual may be part of several communities which are not equally relevant in all communicative situations. We present a model based on Graph Attention Networks that captures this observation. It dynamically explores the social graph of a user, computes a user representation given the most relevant connections for a target task, and combines it with linguistic information to make a prediction. We apply our model to three different tasks, evaluate it against alternative models, and analyse the results extensively, showing that it significantly outperforms other current methods.

## 1 Introduction

The idea that extra-linguistic information about speakers can help language understanding has recently gained traction in NLP. Several studies have successfully exploited social information to classify user-generated language in downstream tasks such as sentiment analysis (Yang and Eisenstein, 2017), abusive speech identification (Mishra et al., 2018) and sarcasm detection (Hazarika et al., 2018; Wallace et al., 2016). The underlying goal is to capture the sociological phenomenon of *homophily* (McPherson et al., 2001) – i.e., people's tendency to group together with others they share ideas, beliefs, and practices with – and to exploit it jointly with linguistic information to obtain richer text representations. In this paper, we advance this line of research. In particular, we address a common shortcoming in current models by using state-of-the-art graph neural networks to encode and leverage homophily relations.

Most current models represent speakers as multidimensional vectors derived by aggregating information about all known social relations of a given individual in a uniform way. A major limit of this approach is that it does not take into account the well known sociolinguistic observation that speakers typically belong to several communities at once, in the sense of 'communities of practice' (Eckert and McConnell-Ginet, 1992) denoting an aggregate of people defined by a common engagement, such as supporters of a political party or fans of a TV show. Membership to these communities has different relevance in different situations. For example, consider an individual who is both part of a supporters group of the USA Democratic Party and of a given sports team. While membership to these two communities can be equally important to characterise the person in general terms, the former is much more relevant when it comes to predicting whether linguistic content generated by this person expresses a certain stance towards president Trump. Current models fail to capture this context-dependent relevance.

In this work, we address this shortcoming, making the following contributions:

- We use Graph Attention Networks (Velickovic et al., 2018) to design a model that dynamically explores the social relations of an individual, learns which of these relations are more relevant for the task at hand, and computes the vector representation of the target individual accordingly. This is then combined with linguistic information to perform text classification tasks.
- To assess the generality and applicability of

*Research conducted when the author was at the University of Amsterdam.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4707–4717,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

the model, we test it on three different tasks (sentiment analysis, stance detection, and hate speech detection) using three annotated Twitter datasets and evaluate its performance against commonly used models for user representation.

- We show that exploiting social information leads to improvements in two tasks (stance and hate speech detection) and that our model significantly outperforms competing alternatives.
- We perform an extended error analysis, in which we show the robustness across tasks of user representations based on social graphs, and the superiority of dynamic representations over static ones.

## 2 Related Work

Several strands of research have explored different social features to create user representations for NLP in social media. Hovy (2015) and Hovy and Fornaciari (2018) focus on demographic information (age and gender), while Bamman et al. (2014) and Hovy and Purschke (2018) exploit geographic location to account for regional variation. Demographic and geographic information, however, need to be made explicit by users and thus are often not available or not reliable. To address this drawback, other studies have aimed at extracting user information by just observing users' behaviour on social platforms.

To tackle sarcasm detection on Reddit, Kolchinski and Potts (2018) assign to each user a random embedding that is updated during the training phase, with the goal of learning individualised patterns of sarcasm usage. Wallace et al. (2016) and Hazarika et al. (2018) address the same task, using ParagraphVector (Le and Mikolov, 2014) to condense all the past comments/posts of a user into a low dimensional vector, which is taken to capture their interests and opinions. All these studies use the concatenation of author and post embeddings for the final prediction, showing that adding author information leads to significant improvements.

While the approaches discussed above consider users individually, a parallel line of work has focused on leveraging the social connections of users in Twitter data. This methodology relies on creating a social graph where users are nodes connected to each other by their retweeting, mentioning, or following behaviour. Techniques such as Line (Tang et al., 2015), Node2Vec (Grover and Leskovec, 2016) or Graph Convolutional Networks (GCNs, Kipf and Welling, 2017) are then used to learn low-dimensional embeddings for each user, which have been shown to be beneficial in different downstream tasks when combined with textual information. For example, Mishra et al. (2018) and Mishra et al. (2019) use the concatenation strategy mentioned above for abusive language detection; Yang et al. (2016) optimise social and linguistic representations with two distinct scoring functions to perform entity linking; while Yang and Eisenstein (2017) use an ensemble learning setup for sentiment analysis, where the final prediction is given by the weighted combination of several classifiers, each exploring the social graph independently.

Methods like Line, Node2Vec and GCNs create user representations by aggregating the information coming from their connections in the social graph, without making any distinction among them. In contrast, we use Graph Attention Networks (GATs, Velickovic et al., 2018), a recent neural architecture that applies self attention to assign different relevance to different connections, and computes node representations accordingly. These representations have been used in several domains to obtain state of the art results in classification tasks where nodes were, for example, texts in a citation network or proteins in human tissues (Velickovic et al., 2018). To our knowledge, we are the first to use Graph Attention Networks to model relations among social media users.

## 3 Model

The model we present operates on annotated corpora made up of triples $(t, a, y)$, where $t$ is some user-generated text, $a$ is its author, and $y$ is a label classifying $t$. We address the task of predicting $y$ given $(t, a)$. Our focus is on user representations: We investigate how model predictions vary depending on how authors are represented.

### 3.1 General Model Architecture

Our model consists of two modules, one encoding the linguistic information in $t$ and the other one modelling social information related to $a$. The general architecture is shown in Figure 1. The output of the linguistic and social modules are vectors $l \in \mathbb{R}^d$ and $s \in \mathbb{R}^{d'}$, respectively. We adopt a standard *late fusion* approach in which these two vectors are concatenated and passed through a two-layer classifier, consisting of a layer $W_1$

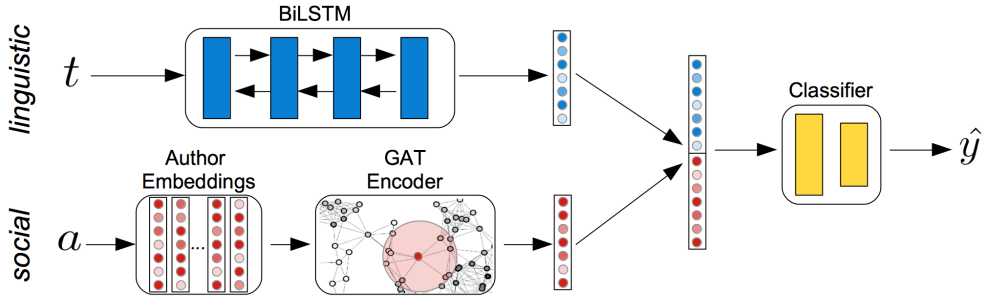


Figure 1: General model: The linguistic module returns a compact representation of the input tweet $t$. The social module takes as input the precomputed representation of the author $a$ and updates it using the GAT encoder. The output embeddings of the two modules are concatenated and fed into a classifier.

$\in \mathbb{R}^{d+d' \times c}$, where $c$ is a model parameter, and a layer $W_2 \in \mathbb{R}^{c \times o}$, where $o$ is the number of output classes. The final prediction is computed as follows, where $\sigma$ is a ReLU function (Nair and Hinton, 2010):

$$\hat{y} = softmax(W_2\,(\sigma\,(W_1\,(l \| s)))) \quad (1)$$

### 3.2 Linguistic Module

The linguistic module is implemented using a recurrent neural network, concretely an LSTM (Hochreiter and Schmidhuber, 1997). Since LSTMs have become ubiquitous in NLP, we omit a detailed description of the inner workings of the model here and refer readers to Tai et al. (2015); Tang et al. (2016); Barnes et al. (2017) for overviews. We use a bidirectional LSTM (BiLSTM) (Graves, 2012), whose final states are concatenated in order to obtain the representation of the input text.

### 3.3 Social Module

The goal of the social module is to return author representations which encode homophily relations among users, i.e., which assign similar vectors to users who are socially related.

We model social relations using graphs $G = (V, E)$, where $V$ is the set of nodes representing individuals and $E$ the set of edges representing relations among them. We use $v_i \in V$ to refer to a node in the social graph, and $e_{ij} \in E$ to denote the edge connecting nodes $v_i$ and $v_j$. Finally, we use $N(v_i)$ to refer to $v_i$'s neighbours, i.e., all nodes which are directly connected to $v_i$.[1]

In contrast to most existing models, where user representations are static, our model uses an encoder which takes as input a pre-computed user vector, performs a dynamic exploration of its neighbours in the social graph, and updates the user representation given the relevance of its connections for a target task. To pre-compute initial user representations we use Node2Vec (N2V, Grover and Leskovec, 2016). Similarly to word2vec's SkipGram model (Mikolov et al., 2013), for every node $v$, N2V implements a function $f : v \rightarrow \mathbb{R}^d$ which maps $v$ to a low-dimensional embedding of size $d$ that maximizes the probability of observing nodes belonging to $S(v)$, i.e., the set of $n$ nodes encountered in the graph by taking $k$ random walks starting from $v$ (where $n$ and $k$ are parameters of the model). No distinction is made among neighbours, thus ignoring the fact that different neighbours may have different importance depending on the task at hand. Our model addresses this fundamental problem by leveraging Graph Attention Networks (GATs, Velickovic et al., 2018).

GATs extend the Graph Convolutional Networks proposed by Kipf and Welling (2017)[2] by introducing a self-attention mechanism (Bahdanau et al., 2015; Parikh et al., 2016; Vaswani et al., 2017) which is able to assign different relevance to different neighbouring nodes. For a target node $v \in V$, an attention coefficient $e_{vu}$ is computed for every neighbouring node $u \in N(v)$ as:

$$e_{vu} = att\,(h_v \| h_u) \quad (2)$$

where $h_v$ and $h_u \in \mathbb{R}^d$ are the vectors representing $v$ and $u$, $\|$ is concatenation, and $att$ is a single-layer feed-forward neural network, parametrized by a weight matrix $W^a \in \mathbb{R}^{2d}$ with Leaky ReLU non-linearity (Maas et al., 2013). The attention co-

[1] Throughout the paper, we use the terms 'individual', 'author', 'user' and 'node' interchangeably.

[2] For a detailed description of Graph Convolutional Networks see https://tkipf.github.io/graph-convolutional-networks/.

efficients for all the neighbours are then normalized using a softmax function. Finally, the update of a node is computed as:

$$h_v^{k+1} = \sigma\left(\sum_{u \in N(v)} \alpha_{vu}^k \left(W^k h_u^k + b^k\right)\right) \quad (3)$$

where $W^k$ and $b^k$ are the layer-specific parameters of the model, $N(v)$ is the set of neighbours of the target node $v$, $h_v^{k+1}$ the updated node representation, $\sigma$ a ReLU function, $k$ the convolutional layer,[3] and $\alpha_{vu}$ the normalised attention coefficient, which defines how much neighbour $u$ should contribute to the update of $v$.

To stabilize the learning process, multiple attention mechanisms, or *heads*, can be used. The number of heads is a hyperparameter of the model. Given $n$ heads, $n$ real-valued vectors $h_v^{k+1} \in \mathbb{R}^{d'}$ are computed and, subsequently, concatenated, thus obtaining a single embedding $h_v^{k+1} \in \mathbb{R}^{n*d'}$. The resulting vector is then concatenated with the output of the linguistic module and fed into the classifier.

## 4 Experimental Setup

### 4.1 Alternative Models

We compare the performance of our model against several competing models. All the models except Frequency baseline and LING compute a user representation which is concatenated with the linguistic information present in a tweet and fed into the classifier.

**Frequency Baseline** Labels are sampled according to their frequency in the whole dataset.

**LING** We use the linguistic module (LING) alone to assess the performance of the model when no social information is provided.

**LING+random** We implement a setting similar to Kolchinski and Potts (2018). Each individual is assigned a random embedding, which is updated during training. Our implementation differs from the Kolchinski and Potts's model in two aspects: They use GRUs (Cho et al., 2014) rather than LSTMs for the linguistic module, and their author vectors have size 15, while ours 200 for a fair comparison with the other models (see below).

[3]The number of layers defines the depth of the neighbourhood, hence $k$ layers encode neighbours lying $k$ hops away in the graph. Also, $v \in N(v)$, i.e., there exists a *self-loop* $(v, v) \in E$ for each node $v$ which ensures that the initial representation of the node is considered during its update.

**LING+PV** In line with Wallace et al. (2016) and Hazarika et al. (2018), we compute a representation for each author by running Paragraph Vector (PV, Le and Mikolov, 2014) on the concatenation of all her previous tweets. Author representations are thus based on past linguistic usage.

**LING+N2V** While none of the previous models make use of the social graph, here we represent authors by means of the embeddings created with N2V (as e.g., Mishra et al., 2018). In contrast to our GAT-based model, the embeddings are computed without making any distinction among neighbours, and are not updated with respect to the task at hand.[4]

### 4.2 Hyperparameter Search

For all the models and for each dataset, we perform grid hyperparameter search on the validation set using early stopping. For batch size, we explore values 4, 8, 16, 32, 64; for dropout, values 0.0, ..., 0.9; and for L2 regularisation, values 0, $1^{e-05}$, $1^{e-04}$. For all the settings, we use Adam optimizer (Kingma and Ba, 2015) with learning rate of 0.001, $\beta_1$ = 0.9 and $\beta_2$ = 0.999. We run PV with the following hyperparameters: 30 epochs, minimum count of 5, vector size of 200. For N2V we use the default hyperparameters, except for vector size (200) and epochs (20). For the GAT encoder, we experiment with values 10, 15, 20, 25, 30, 50 for the size of the hidden layer; for the number of heads, we explore values 1, 2, 3, 4. We keep the number of hops equal to 1 and the alpha value for the Leaky ReLU of the attention heads equal to 0.2 across all the settings.[5]

Since our focus is on social information, we keep the hyperparameters of the linguistic module and the classifier fixed across all the settings. Namely, the BiLSTM has depth of 1, the hidden layer has 50 units, and uses 200-d GloVe embeddings pretrained on Twitter (Pennington et al., 2014). For the classifier, we set the dimensionality of the non-linear layer to 50.

[4]We also experimented with updating author embeddings during training, but did not observe any difference in the results.

[5]We implement PV using the Gensim library: https://radimrehurek.com/gensim/models/doc2vec.html. For N2V, we use the implementation at: https://github.com/aditya-grover/node2vec. For GAT, the implementation at: https://github.com/Diego999/pyGAT.

### 4.3 Tasks and Datasets

We test all the models on three Twitter datasets annotated for different tasks. For all datasets we tokenise and lowercase the text, and replace any URL, hashtag, and mention with placeholders.

**Sentiment Analysis** We use the dataset in Task-4 of SemEval-2017 (Rosenthal et al., 2017), which includes 62k tweets labelled as `POSITIVE` (35.6% of labels), `NEGATIVE` (18.8%) and `NEUTRAL` (45.6%). Tweets in the train set were collected between 2013 and 2015, while those in the test set in 2017. Information for old tweets is difficult to recover:[6] To have a more balanced distribution, we shuffle the dataset and then split it into train (80%), validation (10%) and test (10%).

**Stance Detection** We use the dataset released for Task-6 (Subtask A) of SemEval-2016 (Mohammad et al., 2016), which includes 4k tweets labelled as `FAVOR` (25.5% of labels), `AGAINST` (50.6%) and `NEUTRAL` (23.9%), with respect to five topics: 'Atheism', 'Climate change is a real concern', 'Feminist movement', 'Hillary Clinton', 'Legalization of abortion'. The dataset is split into train and test. We randomly extract 10% of tweets in the train split and use them for validation.

**Hate Speech Detection** We employ the dataset introduced by Founta et al. (2018), from which we keep only tweets labelled as `NORMAL` (93.4% of labels) and `HATEFUL` (6.6%) for a total of 44k tweets.[7] The latter are tweets which denigrate a person or group based on social features (e.g., ethnicity). We split into train (80%), validation (10%) and test (10%).

### 4.4 Optimization Metrics

We tune the models using different evaluation measures, according to the task at hand. The rationale behind this choice is to use, whenever possible, established metrics per task.

For Sentiment Analysis we use average recall, the same measure used for Task 4 of SemEval-2017 (Rosenthal et al., 2017), computed as:

$$AvgRec = \frac{1}{3}(R^P + R^N + R^U) \quad (4)$$

Where $R^P$, $R^N$ and $R^U$ refer to recall of the `POSITIVE`, the `NEGATIVE`, and the `NEUTRAL` class, respectively. The measure has been shown to have several desirable properties, among which robustness to class imbalance (Sebastiani, 2015).

For Stance Detection, we use the average of the F-score of `FAVOR` and `AGAINST` classes:

$$F_{avg} = \frac{F_{favor} + F_{against}}{2} \quad (5)$$

The measure, used for Task-6 (Subtask A) of SemEval-2016, is designed in such a way to optimize the performance of the model in the cases when an opinion toward the target entity is expressed, while it ignores the neutral class (Mohammad et al., 2016).

Finally, for Hate Speech Detection, a more recent task, we could not identify an established metric. We thus use F-score for the target class `HATEFUL`, the minority class accounting for 6.6% of the datapoints (see Section 4.3).

### 4.5 Social Graph Construction

In order to create the social graph, we initially retrieve, for each tweet, the `id` of its author using the Twitter API and scrape her timeline, i.e. her past tweets.[8] We then create an independent social graph $G = (V, E)$ for each dataset. We define $V$ as the set of users authoring the tweets in the dataset, while for $E$ we follow Yang and Eisenstein (2017) and instantiate an unweighted and undirected edge between two users if one retweets the other. Information about retweets is available in users' timeline. In order to make the graph more densely connected, we include external users not present in the dataset who have been retweeted by authors in the dataset at least 100 times. Information about the author of a tweet is not always available: In this case, we assign her an embedding computed as the centroid of the existing pre-computed author representations. We note that in our datasets, authors with more than one tweet are very rare (6.6% on average).

Table 1 summarises the main statistics of the datasets and their respective graphs. The three social graphs have different number of nodes: the network of the Sentiment dataset is the largest (∼62k nodes) while the Stance network is the smallest (∼4k nodes). The number of edges and the density of the network (i.e., the ratio of existing connections over the number of potential

[6] Tweets can be deleted by users or administrators. This happens more often for old tweets.

[7] The other labels in the dataset are `SPAM` and `ABUSIVE`.

[8] We access the API using the Python package Tweepy: `http://docs.tweepy.org/en/v3.5.0/`. The API returns a maximum of 3.2k tweets per user.

connections) vary according to graph size, while the number of connected components is 1 for all the graphs: this means that there are no disconnected sub-graphs in the social networks. The most relevant aspect for which we find differences across the three graphs is the amount of homophily observed, which we define as the percentage of edges which connect users whose tweets have the same label. This value is higher for the Stance and Hate Speech social graphs than for the Sentiment one. These figures indicate that, in our datasets, users expressing similar opinions about a topic (Stance) or using offensive language (Hate Speech) are more connected than those expressing the same sentiment in their tweets (Sentiment).

| | **Sentiment** | **Stance** | **Hate** |
|---|---|---|---|
| **# tweets** | 62,530 | 4,063 | 44,141 |
| **% with author** | 71.4% | 71.7% | 77.1% |
| **# nodes** | 50k | 6.9k | 25k |
| **# edges** | 4.1m | 258k | 1.3m |
| **density** | 0.003 | 0.010 | 0.004 |
| **# components** | 1 | 1 | 1 |
| **homophily** | 38% | 60% | 68% |

Table 1: Statistics for each dataset: Number of tweets; percentage of tweets for which we are able to retrieve information about the author; number of nodes; number of edges; density; number of connected components; and amount of homophily as percentage of connected authors whose tweets share the same label.

## 5 Results

We evaluate the performance of the models using the same metrics used for the optimization process (see Section 4.4). In Table 2 we report the results, that we compute as the average of ten runs with random parameter initialization.[9] We use the unpaired Welch's $t$ test to check for statistically significant difference between models.

**Tasks** The results show that social information helps improve the performance on Stance and Hate Speech detection, while it has no effect for Sentiment Analysis. While this result contrasts with the one reported by Yang and Eisenstein (2017), who use a previous version of the Sentiment dataset (Rosenthal et al., 2015), it is not surprising given the analysis made in the previous section regarding the amount of homophily in the three social graphs: In our version of the data, sentiment is not as related to the social standing of individuals as stance and hatefulness are. We reserve a deeper investigation of the impact of social information on the sentiment task to future work.

| **Model** | **Sentiment** | **Stance** | **Hate** |
|---|---|---|---|
| Frequency | 0.332 | 0.397 | 0.057 |
| LING | 0.676 | 0.569 | 0.624 |
| LING+random | 0.657 | 0.571 | 0.600 |
| LING+PV | 0.671 | 0.601$^{*}$ | 0.667$^{*}$ |
| LING+N2V | 0.672 | 0.629$^{*\diamond}$ | 0.656$^{*}$ |
| LING+GAT | 0.666 | 0.640$^{*\diamond\dagger}$ | 0.674$^{*\diamond\dagger}$ |

Table 2: Results for all the models on the three datasets in our experiment. Marked with $^{*}$ are the results which significantly improve over LING and LING+random ($p < 0.05$, also for the following results); $\diamond$ indicates a significant improvement over LING+PV; $\dagger$ a significant improvement over LING+N2V.

**Models** LING+random never improves over LING: We believe this is due to the fact that most of the authors have just one tweet, which hinders the possibility to learn at training time the representations of the users used at test time.

We find that both PV and N2V user representations lead to an improvement over LING. N2V vectors are especially effective for the Stance detection task, where LING+N2V outperforms LING+PV, while for Hate Speech the performance of the two models is comparable (the difference between LING+PV and LING+N2V is not statistically significative due to the high variance of the LING+PV results - see extended results table in the supplementary material).

Finally, our model outperforms any other model on both Stance and Hate Speech detection. This result confirms our initial hypothesis that a social attention mechanism which is able to assign different relevance to different neighbours allows for a more dynamic encoding of homophily relations in author embeddings and, in turn, leads to better results on the prediction tasks.[10]

[9] The standard deviation for all models is small, in the range [0.003-0.02]. The full table, including the results for each class, is in the supplementary material.

[10] In preliminary experiments, Graph Convolutional Networks with no attention showed no improvements over the N2V baseline. The result is to be expected since, similarly to N2V, the model computes the representation of the target node without making any distinction among its neighbours.

| | |
|---|---|
| (1) *@user: Yurtle the Turtle needs to be slapped with a f***ing chair..many times!* | HATEFUL |
| (2) *You stay the same through the ages... Your love never changes... Your love never fails* | AGAINST atheism |
| (3) *Why are Tumblr feminists so territorial? Pro-lifers can't voice their opinions without being attacked* | AGAINST abortion |
| (4) *@user No, just pointed out how idiotic your statement was* | HATEFUL |

Table 3: Examples from Stance (2, 3) and Hate Speech (1, 4) datasets, and their gold label.

## 6 Analysis

We analyse in more detail the strengths and weaknesses of the best models for the tasks where social information proved useful.

### 6.1 Paragraph Vector

Figure 2 (left) shows the user representations created with PV for the Hate Speech dataset.[11] The plot shows that users form sub-communities, with authors of hateful tweets (orange dots) mainly clustering at the top of the plot. The similarity between these individuals derives from their consistent use of strongly offensive words towards others over their posting history. This suggests that representing speakers in terms of their past linguistic usage can, to some extent, capture certain homophily relations. For example, tweet (1) in Table 3 is incorrectly labelled as NORMAL by the LING model.[12] By leveraging the PV author representation (which, given previous posting behaviour, is highly similar to authors of hateful tweets) the LING+PV model yields the right prediction in this case.

For Stance detection, which arguably is a less lexically determined task (Mohammad et al., 2016), PV user representations are less effective. This is illustrated in Figure 2 (center), where no clear clusters are visible. Still, PV vectors capture some meaningful relations, such as a small, close-knit cluster of users against atheism (see zoom in the figure), who tweet mostly about Islam.

### 6.2 Node2Vec

User representations created by exploiting the social network of individuals are more robust across datasets. For Hate Speech, the user representations computed with PV and N2V are very similar.[13] However, for the Stance dataset, N2V user representations are more informative. This is readily apparent when comparing the plots in the center and right of Figure 2: Users who were scattered when represented with PV now form community-related clusters, which leads to better predictions. For example, tweet (2) in Table 3 is authored by a user who is socially connected to other users who tweet against atheism (the orange cluster in the right-hand side plot of Figure 2). The LING+N2V model is able to exploit this information and make the right prediction, while the tweet is incorrectly classified by LING and LING+PV, which do not take into account the author's social standing.

N2V, however, is not effective for users connected to multiple sub-communities, because by definition the model will conflate this information into a fixed vector located between clusters in the social space.[14] For instance, the author of tweet (1) is connected to both users who post hateful tweets and users whose posts are not hateful. In the N2V user space, the ten closest neighbours of this author are equally divided between these two groups. In this case, the social network information captured by N2V is not informative enough and, as a result, the tweet ends up being wrongly labeled as NORMAL by the LING+N2V model, i.e., there is no improvement over LING.

### 6.3 Graph Attention Network

As hypothesised, the GAT model allows us to address the shortcoming of N2V we have described above. When creating a representation for the author of tweet (1), the GAT encoder identifies the connection to one of the authors of a hateful tweet as the most relevant for the task at hand, and assigns it the highest value. The user vector is updated accordingly, which results in the LING+GAT model correctly predicting the HATEFUL label.

This dynamic exploration of the social connections has the capacity to highlight homophily relations that are less prominent in the social network of a user, but more relevant in a given context. This is illustrated by how the models deal with tweet (3) in Table 3, which expresses a nega-

[11] Plots are created using the Tensorflow Projector available at: https://projector.tensorflow.org.

[12] Note that *'f***ing'* is often used with positive emphasis and is thus not a reliable clue for hate speech.

[13] The plot showing N2V user representations for the Hate Speech dataset is in the supplementary material.

[14] This effect has some parallels with how word2vec derives representations for polysemous words.

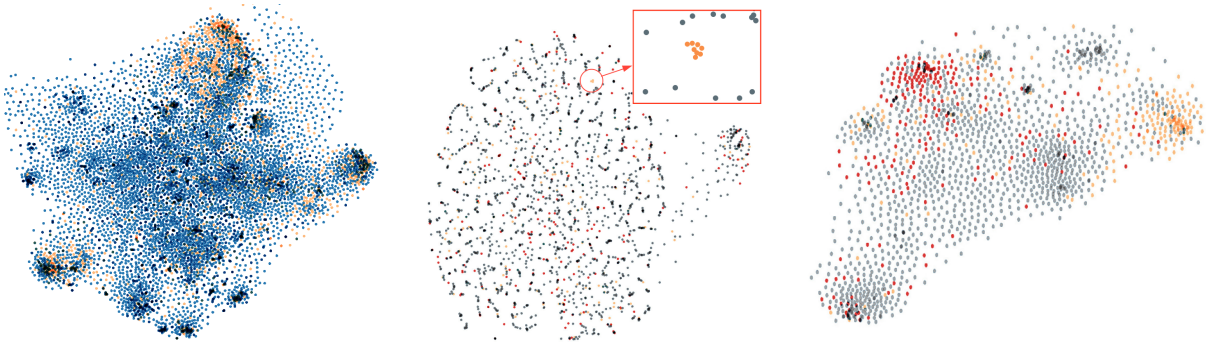

Figure 2: **Left**: PV user representations for the Hate Speech dataset. Orange dots are authors of HATEFUL tweets, blue dots of NORMAL tweets. **Center** and **Right**: PV and N2V user representations, respectively, for the Stance dataset. In both plots, orange dots are authors of tweets AGAINST atheism, red dots authors in FAVOR of 'climate change is a real concern'. All other users are represented as grey dots.

tive stance towards legalisation of abortion, and is incorrectly classified by LING and LING+PV. The social graph contributes rich information about its author, who is connected to many users (46 overall). Most of them are authors who tweet in favour of feminism: The N2V model effectively captures this information, as the representation of the target user is close to these authors in the vector space. Consequently, by simply focusing on the majority of the neighbourhood, the LING+N2V model misclassifies the tweet (i.e., it infers FAVOR for tweet (3) on the legalisation of abortion from a social environment that mostly expresses stances in favour of feminism). However, the information contributed by the majority of neighbours is not the most relevant in this case. In contrast, the GAT encoder identifies the connections with two users who tweet against legalisation of abortion as the most relevant ones, and updates the author representation in such a way to increase the similarity with them – see Figure 3 for an illustration of this dynamic process – which leads the LING+GAT model to make the right prediction.

Interestingly, GAT is able to recognise when the initial N2V representation already encodes the necessary information. For example, the LING+N2V model correctly classifies tweet (4) in Table 3, as the N2V vector of its author is close in social space to that of other users who post hateful tweets (7 out of 10 closest neighbours). In this case, the LING+GAT model assigns the highest value to the self-loop connection, thus avoiding to modify a representation which is already well tuned for the task.

Our error analysis reveals that there are two main factors which affect the performance of the GAT model. One is the size of the neighbourhood: As the size increases, the normalised attention values tend to be very small and equally distributed, which makes the model incapable of identifying relevant connections. The second is related to the fact that a substantial number of users ($\sim$800 for Stance and $\sim$2.4k for Hate Speech) are not connected to the relevant sub-community. This means that in the case of Stance, for example, a user is not connected to any other individual expressing the same stance towards a certain topic. While external nodes in the graph (see Section 4.5) help to alleviate the problem by allowing the information to propagate through the graph, this lack of connections is detrimental to GAT.

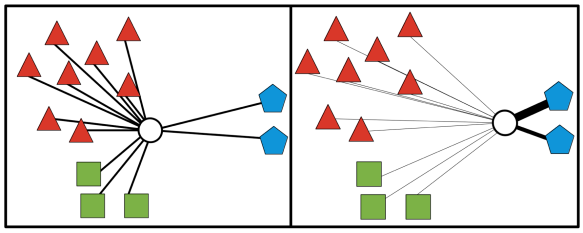

Figure 3: **Left:** Author of tweet (3) (white node) has many connections with users tweeting in favour of feminism (red triangles), fewer with authors tweeting in favour of Clinton (green squares) and against legalization of abortion (blue pentagons) (for simplicity, only some connections are shown). Recall that no label is available for nodes (users), and that their color is based on the label of the tweet they posted. Proximity in the space reflects vector similarity. **Right:** the GAT encoder assigns higher values to connections with the relevant neighbours (0.54 and 0.14; all other connections have values $\leq$ 0.02; thickness of the edges is proportional to their values) and updates the target author vector to make it proximal to them in social space.

## 7 Conclusion

In this work, we investigated representations for users in social media and their usefulness in downstream NLP tasks. We introduced a model that captures the fact that not all the social connections of an individual are equally relevant in different communicative situations. The model dynamically explores the connections of a user, identifies the ones that are more relevant for a specific task, and computes her representation accordingly. We showed that, when social information is proved useful, the dynamic representations computed by our model better encode homophily relations compared to the static representations obtained with other models. In contrast to most models proposed in the literature so far, which are tested on one single task, we applied our model to three tasks, comparing its performance against several competing models. Finally, we performed an extended analysis of the performance of all the models that effectively encode author information, highlighting strengths and weaknesses of each model.

In future work, we plan to perform a deeper investigation of cases in which social information does not prove beneficial, and to assess the ability of our model to dynamically update the representation of the *same* author in different contexts, a task that, due to the nature of the data, was not possible in present work.

## 8 Acknowledgements

This research was partially funded by the Netherlands Organisation for Scientific Research (NWO) under VIDI grants *Asymmetry in Conversation* (276-89-008) and *Semantic Parsing in the Wild* (639-022-518). We are grateful to the Integrated Research Training Group of the Collaborative Research Center SFB 732 for generously funding a research visit by Marco Del Tredici to the University of Stuttgart that fostered some of the ideas in this work. Finally, we kindly thank Mario Giulianelli and Jeremy Barnes for their valuable contribution to earlier versions of the models.

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*.

David Bamman, Chris Dyer, and Noah A Smith. 2014. Distributed representations of geographically situated language. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 828–834.

Jeremy Barnes, Roman Klinger, and Sabine Schulte im Walde. 2017. Assessing state-of-the-art sentiment models on state-of-the-art sentiment datasets. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 2–12, Copenhagen, Denmark. Association for Computational Linguistics.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics.

Penelope Eckert and Sally McConnell-Ginet. 1992. Communities of practice: Where language, gender, and power all live. In *Locating Power, Proceedings of the 1992 Berkeley Women and Language Conference*, pages 89–99.

Antigoni-Maria Founta, Constantinos Djouvas, Despoina Chatzakou, Ilias Leontiadis, Jeremy Blackburn, Gianluca Stringhini, Athena Vakali, Michael Sirivianos, and Nicolas Kourtellis. 2018. Large scale crowdsourcing and characterization of twitter abusive behavior. In *11th International Conference on Web and Social Media, ICWSM 2018*. AAAI Press.

Alex Graves. 2012. Supervised sequence labelling. In *Supervised sequence labelling with recurrent neural networks*, pages 5–13. Springer.

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864. ACM.

Devamanyu Hazarika, Soujanya Poria, Sruthi Gorantla, Erik Cambria, Roger Zimmermann, and Rada Mihalcea. 2018. Cascade: Contextual sarcasm detection in online discussion forums. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1837–1848. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Dirk Hovy. 2015. Demographic factors improve classification performance. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 752–762.

Dirk Hovy and Tommaso Fornaciari. 2018. Improving author attribute prediction by retrofitting linguistic representations with homophily. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 671–677.

Dirk Hovy and Christoph Purschke. 2018. Capturing regional variation with distributed place representations and geographic retrofitting. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4383–4394.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR), 2015*.

Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR), 2017*.

Y Alex Kolchinski and Christopher Potts. 2018. Representing social media users for sarcasm detection. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1115–1121.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196.

Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICLM*, volume 30.

Miller McPherson, Lynn Smith-Lovin, and James M Cook. 2001. Birds of a feather: Homophily in social networks. *Annual review of sociology*, 27(1):415–444.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *ICLR*.

Pushkar Mishra, Marco Del Tredici, Helen Yannakoudakis, and Ekaterina Shutova. 2018. Author profiling for abuse detection. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1088–1098.

Pushkar Mishra, Marco Del Tredici, Helen Yannakoudakis, and Ekaterina Shutova. 2019. Abusive language detection with graph convolutional networks. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*.

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. Semeval-2016 task 6: Detecting stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 31–41.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. Semeval-2017 task 4: Sentiment analysis in twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 502–518.

Sara Rosenthal, Preslav Nakov, Svetlana Kiritchenko, Saif Mohammad, Alan Ritter, and Veselin Stoyanov. 2015. Semeval-2015 task 10: Sentiment analysis in twitter. In *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*, pages 451–463.

Fabrizio Sebastiani. 2015. An axiomatically derived measure for the evaluation of classification algorithms. In *Proceedings of the 2015 International Conference on The Theory of Information Retrieval*, pages 11–20. ACM.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China. Association for Computational Linguistics.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2016. Effective lstms for target-dependent sentiment classification. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3298–3307, Osaka, Japan. The COLING 2016 Organizing Committee.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077. International World Wide Web Conferences Steering Committee.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *The 6th International Conference on Learning Representations (ICLR 2018)*.

Silvio Amir Byron C Wallace, Hao Lyu, and Paula Carvalho Mário J Silva. 2016. Modelling context with user embeddings for sarcasm detection in social media. *CoNLL 2016*, page 167.

Yi Yang, Ming-Wei Chang, and Jacob Eisenstein. 2016. Toward socially-infused information extraction: Embedding authors, mentions, and entities. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1452–1461.

Yi Yang and Jacob Eisenstein. 2017. Overcoming language variation in sentiment analysis with social attention. *Transactions of the Association of Computational Linguistics*, 5(1):295–307.

# Adaptive Ensembling: Unsupervised Domain Adaptation for Political Document Analysis

**Shrey Desai**[1], **Barea Sinno**[2], **Alex Rosenfeld**[3], and **Junyi Jessy Li**[3]

[1]Department of Computer Science
[2]Department of Government, Department of Statistics & Data Science
[3]Department of Linguistics
The University of Texas at Austin
shreydesai@utexas.edu, barea.sinno@utexas.edu
alexbrosenfeld@gmail.com, jessy@austin.utexas.edu

## Abstract

Insightful findings in political science often require researchers to analyze documents of a certain subject or type, yet these documents are usually contained in large corpora that do not distinguish between pertinent and non-pertinent documents. In contrast, we can find corpora that label relevant documents but have limitations (e.g., from a single source or era), preventing their use for political science research. To bridge this gap, we present *adaptive ensembling*, an unsupervised domain adaptation framework, equipped with a novel text classification model and time-aware training to ensure our methods work well with diachronic corpora. Experiments on an expert-annotated dataset show that our framework outperforms strong benchmarks. Further analysis indicates that our methods are more stable, learn better representations, and extract cleaner corpora for fine-grained analysis.

## 1 Introduction

Recent progress in natural language processing and computational social science have pushed political science research into new frontiers. For example, scholars have studied language use in presidential elections (Acree et al., 2018), legislative text in Congress (de Marchi et al., 2018), and similarities in national constitutions (Elkins and Shaffer, 2019). However, datasets used by political scientists are mostly homogeneous in terms of subject (e.g., immigration) or document type (e.g., constitutions). Labeled corpora with pertinent documents usually only stem from a single source; this makes it difficult to generalize conclusions derived from them to other sources. On the other hand, corpora spanning multiple decades and sources tend to be unlabeled. These corpora are largely untouched by political scientists; to illustrate some problems that arise with studying such data, Table 1 shows a sample of topics generated by Latent Dirichlet Allocation (LDA) (Blei et al., 2003), a popular topic model in social science, trained on 60,000 documents sampled from the Corpus of Historical American English (COHA) (Davies, 2008). The generated topics are extremely vague and not specific to politics.

| | |
|---|---|
| Topic 1 | like, day, would, a.m., center |
| Topic 2 | two, samour, family, veronica, son |
| Topic 3 | would, hospital, also, car, hyundai |
| Topic 4 | said, people, one, years, think |
| Topic 5 | city, 6-4, last, wine, york |

Table 1: Randomly sampled topics and top keywords derived from a 50-topic LDA model trained on a sample of COHA documents. Topic modeling results on a political subset of COHA are presented in Table 5. Additionally, topic model hyperparameters are detailed in Appendix A.

This paper bridges the gap between labeled and unlabeled corpora by framing the problem as one of domain adaptation. We develop *adaptive ensembling*, an unsupervised domain adaptation framework that learns from a single-source, labeled corpus (the source domain) and utilizes these representations effectively to obtain labels for a multi-source, unlabeled corpus (the target domain). Our method draws upon *consistency regularization*, a popular technique that stabilizes model predictions under input or weight perturbations (Athiwaratkun et al., 2019). At the framework-level, we introduce an adaptive, feature-specific approach to optimization; at the model-level, we develop a novel text classification model that works well with our framework. To better handle the diachronic nature of our corpora, we also incorporate time-aware training and representations.

Our experiments use the New York Times Annotated Corpus (NYT) (Sandhaus, 2008) as our source domain corpus and COHA as our target do-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4718–4730, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

main corpus. Concretely, we construct two classification tasks: a *binary task* to determine whether a document is political or non-political; and a *multi-label task* to categorize a document under three major areas of political science in the US: *American Government*, *Political Economy*, and *International Relations* (Goodin, 2009). We subsequently introduce an expert-labeled test set from COHA to evaluate our methods.

Our framework, equipped with our best model, significantly outperforms existing domain adaptation algorithms on our tasks. In particular, adaptive ensembling achieves gains of 11.4 and 10.1 macro-averaged F1 on the binary and multi-label tasks, respectively. Qualitatively, adaptive ensembling conditions the optimization process, learns smoother latent representations, and yields precise but diverse topics as demonstrated by LDA on an extracted political subcorpus of COHA. We release our code and datasets at `http://github.com/shreydesai/adaptive-ensembling`.

## 2 Motivation from Political Science

Quantitative studies of American public opinion over time have mostly been restricted to surveys such as the American National Election Survey (Baldassarri and Gelman, 2008; Campbell et al., 1980). However, surveys often do not pose well-formed questions, reflect true voter opinion, or capture mass public opinion (Zaller et al., 1992; Bishop, 2004). Therefore, researchers often seek to compare survey findings with those of mass media as the relationship between public opinion and the media has been widely established (Baum and Potter, 2008; McCombs, 2018). Press media, one form of mass media, manifests itself in large, diachronic collections of newspaper articles; such corpora provide a promising avenue for studying public opinion and testing theories, provided scholars can be confident that the measures they obtain over time are substantively invariant (Davidov et al., 2014). However, as alluded to earlier, such diachronic corpora are often unlabeled; political scientists cannot draw conclusions from these corpora in their raw form as they are unable to distinguish between political and non-political articles. We frame this problem as an exchange between two domains: a source, labeled corpus with modern articles (NYT) and a target, unlabeled corpus with decades of articles originating from a multitude of news sources (COHA). Using domain adaptation methods, we can extract a political subcorpus from COHA that would be amenable for the study of public opinion research over time.

## 3 Unsupervised Domain Adaptation

In this section, we detail the core concepts behind our unsupervised domain adaptation framework. We describe the problem setup (§3.1), an overview of self-ensembling and consistency regularization (§3.2-§3.4), and our novel contributions to this framework (§3.5-§3.6).

### 3.1 Problem Setup

Let $\mathcal{X}$ and $\mathcal{Y}$ denote the input and output spaces, respectively. We have access to labeled samples $\{x_L^{(i)}, y_L^{(i)}\}_{i=1}^N$ from a source domain $\mathcal{D}_S$ and unlabeled samples $\{x_U^{(i)}\}_{i=1}^M$ from a target domain $\mathcal{D}_T$. The goal of unsupervised domain adaptation is to learn a function $f : \mathcal{X} \rightarrow \mathcal{Y}$ that maximizes the likelihood of the target domain samples by only leveraging supervision from the source domain samples. We also assume the existence of a small amount of labeled target domain samples in order to create a development set, following existing work in unsupervised domain adaptation (Glorot et al., 2011; Chen et al., 2012; French et al., 2018; Zhang et al., 2017).

### 3.2 Self-Ensembling

Our unsupervised domain adaptation framework builds on top of *self-ensembling* (Laine and Aila, 2017), a semi-supervised learning algorithm based on *consistency regularization*, whereby models are trained to be robust against injected noise (Athiwaratkun et al., 2019).

Self-ensembling is an interplay between two neural networks: a student network $f(x; \theta)$ and a teacher network $f(x; \phi)$. The inputs to both networks are perturbed separately, and the objective is to measure the consistency of the student network's predictions against the teacher's. Both networks share the same base model architecture and initial parameter values, but follow different training paradigms (Laine and Aila, 2017). In particular, the student network is updated via backpropagation, then the teacher network is updated with an exponential average of the student network's parameters (Tarvainen and Valpola, 2017). The networks are trained in an alternating fashion until they converge. During test time, the teacher

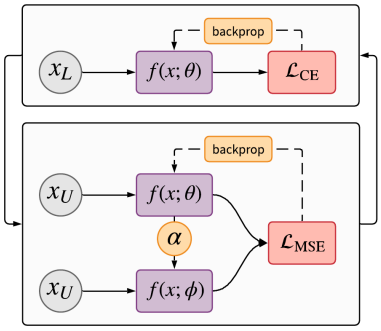


Figure 1: Visualization of the self-ensembling training procedure. Descriptions of individual components are detailed in §3.2-§3.4.

network is used to infer the labels for target domain samples. Figure 1 visualizes the overall training procedure. Further intuition behind self-ensembling is available in Appendix B.

Next, we discuss the training process for the student network (§3.3), the original fixed ensembling method in Tarvainen and Valpola (2017) (§3.4), and our proposed adaptive ensembling method (§3.5).

### 3.3 Student Training

The student network uses labeled samples from the source domain and unlabeled samples from the target domain to learn domain-invariant features. This is realized by using multiple loss functions, each with its own objective. The **supervised loss** is simply the cross-entropy loss of the student network outputs given source domain samples:

$$\mathcal{L}_{\text{CE}}(\theta) = \sum_{(x,y)\in\mathcal{D}_S} \log p(y|x')$$

However, the supervised loss alone prevents the student network from learning anything useful about the target domain. To address this, Laine and Aila (2017) introduce an **unsupervised loss** to ensure that the student and teacher networks have similar predictions for target domain samples. French et al. (2018) only enforce the consistency constraint for target domain samples, but we propose using both source *and* target domain samples with separately perturbed inputs $x'$ and $x''$; this provides a balanced source of supervision to train our adaptive constants, discussed in §3.5:

$$\mathcal{L}_{\text{MSE}}(\theta,\phi) = \sum_{x\in\mathcal{D}_S\cup\mathcal{D}_T} ||f(x';\theta) - f(x'';\phi)||^2$$

The overall objective is a combination of the two loss functions:

$$\mathcal{L}(\theta,\phi) = \mathcal{L}_{\text{CE}} + \mathcal{L}_{\text{MSE}}$$

### 3.4 Fixed Ensembling

The teacher network's parameters form an ensemble of the student network's parameters over the course of training:

$$\phi^{(t+1)} \leftarrow \alpha\phi^{(t)} + (1-\alpha)\theta^{(t)}$$

where $\alpha$ is a smoothing factor that controls the magnitude of the parameter updates. Since the labels for the target domain samples are inherently unknown, ensembling parameters in the presence of noise helps the teacher network's predictions converge to the *true* label (Tarvainen and Valpola, 2017).

**Limitations** Empirically, we find that the highly unstable loss surface presented by textual datasets causes large instabilities in the optimization process. One of the key insights of this paper is that these instabilities are due to the dynamics of the unsupervised loss. Because the unsupervised loss effectively regularizes the source domain representations to work well in the target domain (Laine and Aila, 2017), performance degrades rapidly if this loss fails to converge. This is a strong indicator that self-ensembling fails to learn useful, shared representations for knowledge transfer between textual domains. Qualitative evidence of the unsupervised loss' instability is shown in Figure 6a and further discussed in §7.

### 3.5 Adaptive Ensembling

We hypothesize that smoothing with a fixed hyperparameter $\alpha$ is responsible for said instabilities. For any given weight matrix (or bias vector), each hidden unit can be conceptualized as controlling one highly specific feature or attribute (Bau et al., 2019). These units may need to be updated with varying degrees throughout the course of training; therefore, smoothing each unit with a fixed constant severely overlooks dynamics at the parameter-level. We propose modifying fixed ensembling by introducing trainable smoothing constants for each unit—hereafter termed *adaptive constants*—as opposed to using a fixed smoothing constant:

$$\phi^{(t+1)} \leftarrow \mathbf{C}^{(t)} \odot \phi^{(t)} + (\mathbf{1} - \mathbf{C}^{(t)}) \odot \theta^{(t)}$$

where a matrix of adaptive constants $\mathbf{C}$ is applied element-wise to $\phi$ and $\theta$ at each step.

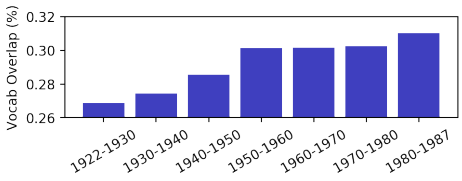


Figure 2: Vocabulary overlap between COHA and NYT, by decade. We collect COHA documents in each decade, create a decade vocabulary, and calculate the percentage overlap between each decade's vocabulary and the overall NYT vocabulary.

**Example** Assume we are training an arbitrary weight matrix $\mathbf{W} \in \mathbb{R}^{m \times n}$ in the $k$th layer of a fixed network architecture. Both the student and teacher network have their own copy of $\mathbf{W}$, denoted as $\mathbf{W}_{\text{STU}}$ and $\mathbf{W}_{\text{TEA}}$, respectively. To ensure each parameter $\mathbf{W}_{ij}$ has a corresponding adaptive constant $\alpha_{ij}$, $\mathbf{C}$ shares the same dimensionality as $\mathbf{W}_{\text{STU}}$ and $\mathbf{W}_{\text{TEA}}$. The previous equation can then be written as:

$$\mathbf{W}_{\text{TEA}}^{(t+1)} \leftarrow \mathbf{C}^{(t)} \odot \mathbf{W}_{\text{TEA}}^{(t)} + (\mathbf{1} - \mathbf{C}^{(t)}) \odot \mathbf{W}_{\text{STU}}^{(t)}$$

**Supervision** Because the adaptive constants are designed to stabilize training, it is a natural fit to train them using the unsupervised loss:

$$\mathbf{C}^{(t+1)} \leftarrow \mathbf{C}^{(t)} - \epsilon \nabla_{\mathbf{C}} \mathcal{L}_{\text{MSE}}$$

This forms a crucial difference between self-ensembling and adaptive ensembling: in the former method, the teacher network has no say in how its parameters are modified. Adaptive ensembling equips the teacher network with fine-grained control over gradient updates, making it far easier to align activations under a noisy setting.

### 3.6 Temporal Curriculum

Diachronic datasets important in political science can be difficult to adapt to given the minimal vocabulary overlap between the source and target domain documents. Source and target articles mention named entities and events that, for the most part, do not appear across both datasets. To ease the difficulty of domain adaptation, we exploit the temporal information in our datasets to introduce a curriculum (Bengio et al., 2009).

In particular, each article comes with metadata that includes the year in which the article was published. Figure 2 shows that COHA articles written closer to the time of NYT articles have a larger vocabulary overlap than those written in the distant past. Intuitively, it is easier to learn features from target domain samples that are more like the source domain samples. Hence, we sort the target domain mini-batches by year; the learning task becomes progressively harder as opposed to confusing the models during the early stages of training.

## 4 Model

In this section, we introduce a new convolutional neural network (CNN) as the plug-in model for our unsupervised domain adaptation framework. We motivate the use of CNNs (§4.1), formalize the model input (§4.2), and introduce several novel components for our task (§4.3).

### 4.1 Motivation

CNNs have emerged as strong baselines for text classification in NLP (Kim, 2014). CNNs are desirable candidates for our framework as they exhibit a high degree of parameter sharing, significantly reducing the number of parameters to train. In addition, they can be designed to solely optimize the log-likelihood of the training data. Experimentally, we find that models that optimize other distributions (e.g., attention distributions in Transformers (Vaswani et al., 2017) or Hierarchical Attention Networks (Yang et al., 2016)) do not work well with this framework.

### 4.2 Model Input

Given a discrete input $x = [w_1, \cdots, w_n]$ and vocabulary $V$, an embedding matrix $\mathbf{E} \in \mathbb{R}^{|V| \times d}$ replaces each word $w_i$ with its respective $d$-dimensional embedding. The resulting embeddings are stacked row-wise to obtain an input matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$. Following the notion of input perturbation used in consistency regularization algorithms (Athiwaratkun et al., 2019), we design several methods to inject noise into the input layer. Each input is perturbed with additive, isotropic Gaussian noise: $\tilde{\mathbf{X}} = \mathbf{X} + \mathcal{N}(0, \mathbf{I})$. Then, we apply dropout on the perturbed inputs to eliminate dependencies on any one word: $\mathbf{X}' = \tilde{\mathbf{X}} \odot \mathbf{M}$ where $\mathbf{M} \in \mathbb{R}^{n \times d}$ is a Bernoulli mask applied element-wise to the input matrix.

### 4.3 Model Architecture

**Background: 1D Convolutions** CNNs for text classification generally use 2D convolutions over the input matrix (Kim, 2014), but architectures using 1D convolutions have also been explored in

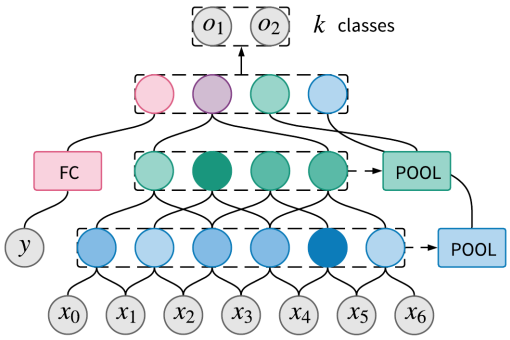


Figure 3: Example three-layer CNN architecture with sequence squeezing, state connections, and time embeddings. Detailed information about individual components is available in §4.3. FC represents a fully-connected layer; POOL represents a pooling layer. Best viewed in color.

other contexts, e.g., sequence modeling (Bai et al., 2018), machine translation (Kalchbrenner et al., 2016), and text generation (Yang et al., 2017). Our model draws upon the latter approach for political document classification. CNNs utilizing 1D convolutions are typically autoregressive in nature; that is, each output $y_t$ only depends on the inputs $x_{<t}$ to avoid information leakage into the future. Two approaches have been proposed to achieve this: history-padding (Bai et al., 2018, 2019) and masked convolutions (Kalchbrenner et al., 2016). Further, each successive convolution uses an exponentially increasing dilation factor, reducing the depth of the network significantly. Below, we elaborate on the components of our model:

**Sequence Squeezing** Given a model with $\ell$ layers, previous approaches (Bai et al., 2018, 2019) history-pad the input with $\sum_{i=1}^{\ell} d^{(i-1)}(f-1)$ zeros to obtain an output of length $n$, where $d$ is the dilation factor and $f$ is the filter size. However, we propose history-padding the input with $(\sum_{i=1}^{\ell} d^{(i-1)}(f-1)) - n + 1$ zeros to ensure the convolutions compress the sequence down to *one* output unit. Formally, this produces an output feature map of dimension $B \times C \times 1$ where $B$ is the batch size and $C$ is the number of channels; one can use a simple `squeeze()` operation to obtain the compact feature matrix $B \times C$. Though this is a subtle difference, our approach yields much richer representations for classification.

**State Connections** In each layer $\ell_i$, a kernel $\mathbf{W}_i$ convolves across an intermediate sequence, inducing a feature map $\mathbf{A}_i$. Because the input is presented as a sequence, the application of $\mathbf{W}_i$ along a one-dimensional axis encourages $\mathbf{A}_i$ to encode temporal features, similar to how the hidden state is formed by applying shared weights across a sequence in recurrent architectures. Further, because the receptive field grows exponentially, the convolutions build hierarchical representations of the input, implying $\mathbf{A}_{i+1}$ builds a more abstract representation of the input than $\mathbf{A}_i$. We exploit this stateful information by pooling each activation map $\mathbf{A}_i$ into a vector and concatenating them row-wise to create a *state matrix*:

$$\mathbf{S} = \text{Concat}(\text{Pool}(\mathbf{A}_1), \cdots, \text{Pool}(\mathbf{A}_{\ell-1}))$$

To the best of our knowledge, our paper is the first to explicitly use the temporal state embedded in causal 1D convolution activations as representations for an end task.

**Time Embedding** To make our model time aware, we learn representations for the years of the documents (available as metadata in COHA). Such time representations allow the model to reason about content as it appears in different decades. Given a year $y$ (e.g. 1954), we normalize it to the closed unit interval $[0, 1]$ and linearly transform it into a low-dimensional embedding $\mathbf{e}$:

$$\mathbf{e} = \mathbf{W}_e \left[ \frac{y - \max_y}{\max_y - \min_y} \right] + b_e$$

where $\max_y$ and $\min_y$ represent the maximum and minimum observed years in the training dataset, respectively.

**Overall Architecture** We concatenate the various components of our model $[\mathbf{X}'; \mathbf{S}; \mathbf{e}]$ to create a collective representation for classification. We use a 1D convolution ($f = 1$ and $d = 1$) to project this representation to $k$ classes:

$$\mathbf{y} = \text{Conv1D}([\mathbf{X}'; \mathbf{S}; \mathbf{e}]; \mathbf{W}_k)$$

We did not observe any performance advantages from using a fully-connected layer to perform the projection, so we opt to use a fully-convolutional architecture to minimize the number of parameters (Long et al., 2015). Finally, we apply softmax to the output vector $\mathbf{y} \in \mathbb{R}^k$ to obtain a valid probability distribution over the classes. An example of our model architecture is depicted in Figure 3.

## 5 Datasets

We present a dataset for identifying political documents with manual annotation from political science graduate students. The dataset is constructed for *binary* and *multi-label* tasks: (1) identifying whether a document is political (i.e. containing notable political content) and (2) if so, the area(s) among three major political science subfields in the US: *American Government*, *Political Economy*, and *International Relations* (Goodin, 2009).

**Source** We use NYT as the source dataset as it contains fine-grained descriptors of article content. We sample 4,800 articles with the descriptor US POLITICS & GOVERNMENT. To obtain non-political articles, we sample 4,800 documents whose descriptors do not overlap with an exhaustive list of political descriptors identified by a political science graduate student. For our multi-label task, the annotator grouped descriptors in NYT that belong to each area label we consider[1].

**Target** Our target data are historical documents from COHA, which contains a large collection of news articles since the 1800s. To ensure our dataset is useful for diachronic analysis (e.g., public opinion over time), we sample only from news sources that consistently appear across the decades. Further, we ensure there are at least 8,000 total documents in each decade group; this narrows down our time span to 1922–1986. From this subset, we sample ∼250 documents from each decade for annotation. Two political science graduate students each annotated a subset of the data.

To train our unsupervised domain adaptation framework, we use 9,600 unlabeled target examples (same number as NYT). The expert-annotated dataset is divided into three subsets: a training set of 984 documents (*only* for training the In-Domain classifier discussed in §6.2), development set of 246 documents, and test set of 350 documents (50 per decade)[2].

## 6 Experiments

### 6.1 Settings

Our CNN has 8 layers, each with 256 channels, $f = 3$, $d = 2^i$ (for the $i$th layer), and ReLU activation. We enforce a maximum sequence length of 200 and minimum word count from $[1, 2, 3]$ to build the vocabulary. The embedding matrix uses 300-D GloVe embeddings (Pennington et al., 2014) with a dropout rate of 0.5 (Srivastava et al., 2014). We history-pad our input with a zero vector, the state connections are obtained using average pooling, and the time embedding has a dimensionality of 10. The model is optimized with Adam (Kingma and Ba, 2015), learning rate from $[10^{-4}, 5 \cdot 10^{-5}, 10^{-5}]$, and mini-batch size from $[16, 32]$. Hyperparameters were discovered using a grid search on the held-out development set.

[1]These descriptors are available in Appendix C.

[2]The news sources used and label distributions for the expert-annotated dataset are available in Appendix D.

| | Binary Task | | Multi-Label Task | | |
|---|---|---|---|---|---|
| Method | Mi-F | Ma-F | Ma-P | Ma-R | Ma-F |
| Source Only | 55.7 | 46.2 | 28.8 | 70.0 | 39.6 |
| mSDA | 57.4 | 49.7 | 41.0 | 63.7 | 48.1 |
| DANN | 68.2 | 65.8 | **50.8** | 36.3 | 42.2 |
| SE | 64.0 | 59.5 | 42.7 | 64.1 | 51.0 |
| + curriculum | 66.4 | 62.3 | 44.4 | 71.7 | 54.5 |
| AE (ours) | 75.1 | 74.5 | 46.1 | 75.3 | 57.2 |
| + curriculum | **77.4** | **77.1** | 48.2 | **83.5** | **61.1** |
| In-Domain | 81.7 | 81.6 | 86.5 | 83.5 | 84.8 |

Table 2: Framework results for the binary label task (left) and multi-label task (right). For the binary task, we show micro- and macro-averaged F1 scores. For the multi-label task, we show macro-averaged precision, recall, and F1 scores.

### 6.2 Framework Results

Using our best model, we benchmark our unsupervised domain adaptation framework against established methods: (1) **Marginalized Stacked Denoising Autoencoders (mSDA):** Denoising autoencoders that marginalize out noise, enabling learning on *infinitely many* corrupted training samples (Chen et al., 2012). (2) **Self-Ensembling (SE):** A consistency regularization framework that stabilizes student and teacher network predictions under injected noise (discussed in detail in §3.2-§3.4) (Laine and Aila, 2017; Tarvainen and Valpola, 2017; French et al., 2018). (3) **Domain-Adversarial Neural Network (DANN):** Multi-component framework that learns domain-invariant representations through adversarial training (Ganin et al., 2016). We also benchmark against **Source Only** (classifier trained on the source domain only) and **In-Domain** (classifier trained on the target domain only) to establish lower and upper performance bounds, respectively (Zhang et al., 2017).

Framework results are presented in Table 2. Our

| | Binary Task | | Multi-Label Task | | |
|---|---|---|---|---|---|
| Model | Mi-F | Ma-F | Ma-P | Ma-R | Ma-F |
| LR | 63.7 | 60.9 | 53.3 | 67.3 | 31.9 |
| BiLSTM | 64.8 | 63.1 | 36.2 | 65.0 | 46.3 |
| CNN (2D) | 73.1 | 72.1 | **49.0** | 73.8 | 58.9 |
| CNN (ours) | 75.4 | 75.3 | 36.9 | **91.4** | 52.5 |
| + seq squeeze | 75.1 | 74.6 | 45.8 | 79.6 | 58.2 |
| + state conn | **80.2** | 76.3 | 45.3 | 85.5 | 59.2 |
| + time emb | 77.4 | **77.1** | 48.2 | 83.5 | **61.1** |

Table 3: Model results with adaptive ensembling for the binary label task (left) and multi-label task (right). For the binary task, we show micro- and macro-averaged F1 scores. For the multi-label task, we show macro-averaged precision, recall, and F1 scores.

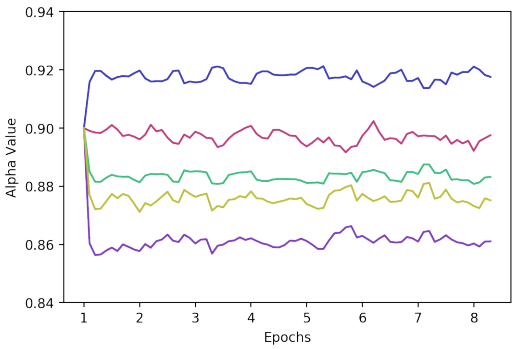


Figure 4: Trajectories of five randomly sampled adaptive constants from our CNN over the course of training. Best viewed in color.

method achieves the highest F1 scores for both tasks. The temporal curriculum further improves our results by a large margin, validating its effectiveness for domain adaptation on diachronic corpora. Although DANN achieves higher precision on the multi-label task, its recall largely suffers.

### 6.3 Model Results

Next, we ablate the various components of our model and evaluate several other strong text classification baselines under our framework: (1) **Logistic Regression (LR):** We average the word embeddings of each token in the sequence, then use these to train a logistic regression classifier. (2) **Bidirectional LSTM (BiLSTM):** A bidirectional LSTM obtains forwards and backwards input representations $[\overrightarrow{\mathbf{h}_t}; \overleftarrow{\mathbf{h}_t}]$ (Hochreiter and Schmidhuber, 1997); they are concatenated and passed through a fully-connected layer. (3) **CNN (2D):** A CNN using 2D kernels $3 \times 300$, $4 \times 300$, and $5 \times 300$ obtains representations (Kim, 2014). They are max-pooled, concatenated row-wise, and passed through a fully-connected layer.

Model ablations and results are presented in Table 3. Our full model achieves the highest F1 scores on both the binary and multi-label tasks, and each component consistently contributes to the overall F1 score. The 2D CNN also has decent F1 scores, showing that our framework works with standard CNN models. Further, the time embedding significantly improves both F1 scores, indicating the model effectively utilizes the unique temporal information present in our corpora.

## 7 Analysis

In this section, we pose and qualitatively answer numerous probing questions to further understand the strong performance of adaptive ensembling. We analyze several characteristics of the overall framework (§7.1), then qualitatively inspect its performance on our datasets (§7.2).

### 7.1 Framework

**Are the adaptive constants different across hidden units?** We randomly sample five adaptive constants and track their value trajectories over the course of training. Figure 4 shows all of them sharply converge to and bounce around the same general neighboorhood. This is strong evidence that we cannot use a fixed hyperparameter $\alpha$ to smooth each parameter, rather we need per-parameter smoothing constants to account for the functionality and behavior of each unit.

**How do the adaptive constants change by layer?** Figure 5 shows the distribution of weight and bias parameters of adaptive constants for a top, middle, and bottom layer of our CNN. For the weight parameters, the teacher relies heavily on the student ($\alpha$ is skewed towards smaller smoothing rates) in the top layer, but gradually reduces its dependence by learning target domain features in the lower layers ($\alpha$ is skewed towards larger smoothing rates). For the bias parameters, the teacher prominently shifts the student features to work for the target domain in the top layer, but reduces its dependence on the student in the lower layers. This shows why using a fixed hyperparameter $\alpha$ does not account for layer-wise dynamics, i.e. each layer requires a specific distribution of $\alpha$ values to achieve strong performance.

**Do adaptive constants benefit training and latent representations?** Figure 6a depicts the unsupervised loss trajectories for self-ensembling

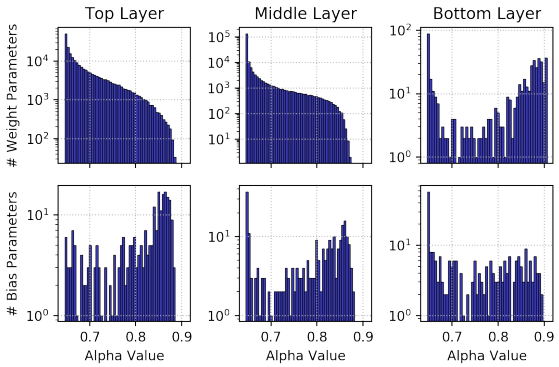


Figure 5: Distribution of teacher network adaptive constants for a top, middle, and bottom layer. We display adaptive constants for both weight (top) and bias (bottom) parameters. The $x$-axis is shared for both the weight and bias distributions.

(SE) and adaptive ensembling (AE). Compared to SE, the adaptive constants significantly stabilize the unsupervised loss. Next, Figure 6b shows the general training curves for AE and domain-adversarial neural networks (DANN). The DANN loss oscillates uncontrollably as the adversarial weight increases, but increasing the unsupervised loss weight for AE does not result in as much instability. We also compare the latent representations learned by SE and AE in Figure 7. While SE shows evidence of feature alignment, AE learns a much smoother manifold where source and target domain representations are intertwined.

### 7.2 Datasets

**Does adaptive ensembling yield better topics?** In Table 1, we showed that applying LDA directly on COHA yields noisy, unrecognizable topics. Here, we use the SOURCE ONLY model and the adaptive ensembling framework to obtain labels for the unlabeled pool of COHA documents. We extract the political documents, run a topic model on the political subcorpus, and randomly sample topics. The SOURCE ONLY results are shown in Table 4 and the adaptive ensembling results are shown in Table 5. The SOURCE ONLY model has poor recall, as most of the topics are extracted are vague and not inherently political in nature. In contrast, our framework is able to extract a wide range of clean, identifiable political topics. For example, the first topic reflects documents related to the Vietnam conflict while the third topic reflects documents related to important court proceedings.

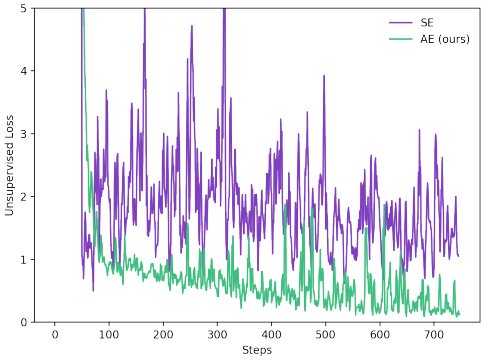


(a) Loss curves for self-ensembling (SE) and adaptive ensembling (AE) over the course of training.

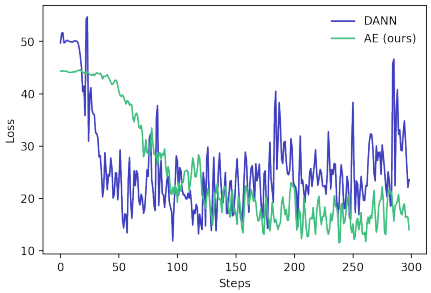


(b) Loss curves for domain-adversarial neural networks (DANN) and adaptive ensembling (AE). The adversarial loss weight and unsupervised loss weight are annealed for both methods. For fair comparison, we employ the adversarial weight annealing schedule outlined in Ganin et al. (2016).

| | |
|---|---|
| Topic 1 | dr, women, week, medical, doctors |
| Topic 2 | city, police, street, car, avenue |
| Topic 3 | trial, years, police, prison, court |
| Topic 4 | union, strike, workers, lewis, service |
| Topic 5 | like, man, years, little, week |

Table 4: Randomly sampled topics and top keywords derived from a 50-topic LDA model trained on 28K COHA articles identified as political using the SOURCE ONLY model.

**Does adaptive ensembling preserve the integrity of the original corpus?** In order for political scientists to effectively study latent variables—such as political polarization—over time, the extracted political subcorpus must contain a similar integrity as the original corpus. That is, the subcorpus' distribution of documents across years and sources must relatively match that of the original corpus. First, we analyze the document counts for each decade bin, shown in Figure 8. The political subcorpus shows a relatively consistent count across the deacdes, notably also capturing salient peaks from the 1920-1930s. Next, we analyze the document counts for each news source. Once again, the political subcorpus features documents from *all* sources that appear in

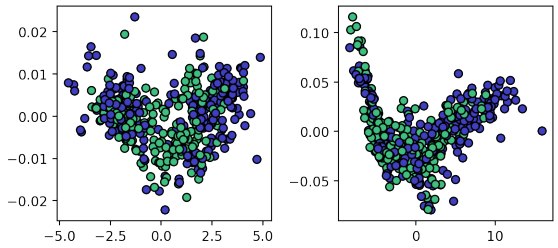

Figure 7: PCA performed on the latent representations of the teacher network in self-ensembling (left) and adaptive ensembling (right). We show representations for both source domain samples (green) and target domain samples (blue). Best viewed in color.

| | |
|---|---|
| Topic 1 | vietnam, hanoi, atomic, bombing, south |
| Topic 2 | germany, britain, france, europe, soviet |
| Topic 3 | court, justice, commission, law, attorney |
| Topic 4 | tax, oil, prices, petroleum, industry |
| Topic 5 | coal, union, strike, workers, miners |

Table 5: Randomly sampled topics and top keywords derived from a 50-topic LDA model trained on 28K COHA articles identified as political using ADAPTIVE ENSEMBLING.

the original corpus. In addition, the varied distribution across sources is also captured; Time Magazine (TM) has the most documents whereas Wall Street Journal (WSJ) has the least documents. Together, these results show that the resulting subcorpus is amenable for political science research as it exhibits important characteristics derived from the original COHA corpus.

## 8 Related Work

Early approaches for unsupervised domain adaptation use shared autoencoders to create cross-domain representations (Glorot et al., 2011; Chen et al., 2012). More recently, Ganin et al. (2016) introduce a new paradigm that create domain-invariant representations through adversarial training. This has gained popularity in NLP (Zhang et al., 2017; Fu et al., 2017; Chen et al., 2018), however the difficulties of adversarial training are well-established (Salimans et al., 2016; Arjovsky and Bottou, 2017). Consistency regularization methods (e.g., self-ensembling) outperform adversarial methods on visual semi-supervised and domain adaptation tasks (Athiwaratkun et al., 2019), but have rarely been applied to textual data (Ko et al., 2019). Finally, Huang and Paul (2018) establish the feasibility of using domain adaptation to label documents from discrete time periods.

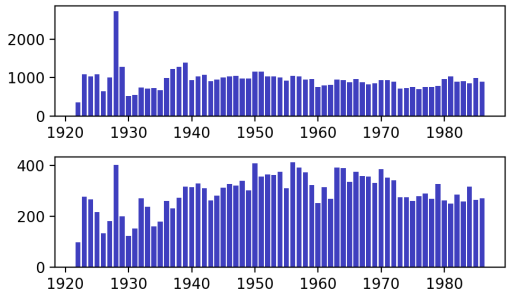

Figure 8: Document counts per decade for the original COHA corpus (top) and the extracted political subcorpus (bottom).

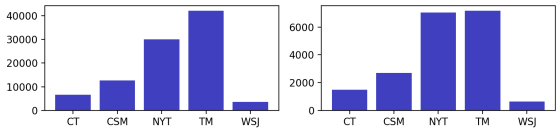

Figure 9: Document counts per news source for the original COHA corpus (left) and the extracted political subcorpus (right). The sources displayed include Chicago Tribute (CT), Christian Science Monitor (CSM), New York Times (NYT), Time Magazine (TM), and Wall Street Journal (WSJ).

Our work departs from previous work by proposing an adaptive, time-aware approach to consistency regularization provisioned with causal convolutional networks.

## 9 Conclusion

We present *adaptive ensembling*, an unsupervised domain adaptation framework capable of identifying political texts for a multi-source, diachronic corpus by only leveraging supervision from a single-source, modern corpus. Our methods outperform strong benchmarks on both binary and multi-label classification tasks. We release our system, as well as an expert-annotated set of political articles from COHA, to facilitate domain adaptation research in NLP and political science research on public opinion over time.


## Acknowledgments

The authors acknowledge the Texas Advanced Computing Center (TACC) at The University of Texas at Austin for providing HPC resources used to conduct this research. Thanks as well to Greg Durrett, Katrin Erk, and the anonymous reviewers for their helpful comments. This work was partially supported by the NSF Grant IIS-1850153.

## References


Brice DL Acree, Justin H Gross, Noah A Smith, Yanchuan Sim, and Amber E Boydstun. 2018. Etch-a-sketching: Evaluating the post-primary rhetorical moderation hypothesis. *American Politics Research*.

Martin Arjovsky and Léon Bottou. 2017. Towards principled methods for training generative adversarial networks. In *International Conference on Learning Representations*.

Ben Athiwaratkun, Marc Finzi, Pavel Izmailov, and Andrew Gordon Wilson. 2019. There are many consistent explanations of unlabeled data: Why you should average. In *International Conference on Learning Representations*.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*.

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. 2019. Trellis networks for sequence modeling. In *International Conference on Learning Representations*.

Delia Baldassarri and Andrew Gelman. 2008. Partisans without constraint: Political polarization and trends in american public opinion. *American Journal of Sociology*, 114(2):408–446.

Anthony Bau, Yonatan Belinkov, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2019. Identifying and controlling important neurons in neural machine translation. In *International Conference on Learning Representations*.

Matthew A Baum and Philip BK Potter. 2008. The relationships between mass media, public opinion, and foreign policy: Toward a theoretical synthesis. *Annual Review of Political Science*, 11:39–65.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 41–48.

George F Bishop. 2004. *The illusion of public opinion: Fact and artifact in American public opinion polls*. Rowman & Littlefield Publishers.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022.

Angus Campbell, Philip E Converse, Warren E Miller, and Donald E Stokes. 1980. *The American Voter*. University of Chicago Press.

Minmin Chen, Zhixiang Xu, Kilian Q. Weinberger, and Fei Sha. 2012. Marginalized denoising autoencoders for domain adaptation. In *Proceedings of the 29th International Conference on International Conference on Machine Learning*, pages 1627–1634.

Xilun Chen, Yu Sun, Ben Athiwaratkun, Claire Cardie, and Kilian Weinberger. 2018. Adversarial deep averaging networks for cross-lingual sentiment classification. *Transactions of the Association for Computational Linguistics*, 6:557–570.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46.

Eldad Davidov, Bart Meuleman, Jan Cieciuch, Peter Schmidt, and Jaak Billiet. 2014. Measurement equivalence in cross-national research. *Annual review of sociology*, 40:55–75.

Mark Davies. 2008. The corpus of contemporary american english: 450 million words, 1990-present.

Zachary Elkins and Robert Shaffer. 2019. On measuring textual similarity. Work in progress.

Geoff French, Michal Mackiewicz, and Mark Fisher. 2018. Self-ensembling for visual domain adaptation. In *International Conference on Learning Representations*.

Lisheng Fu, Thien Huu Nguyen, Bonan Min, and Ralph Grishman. 2017. Domain adaptation for relation extraction with domain adversarial neural network. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 425–429.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on Machine Learning*, pages 513–520.

Robert E Goodin. 2009. *The Oxford handbook of political science*, volume 11. Oxford University Press.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Xiaolei Huang and Michael J Paul. 2018. Examining temporality in document classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 694–699.

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference for Learning Representations*.

Wei-Jen Ko, Greg Durrett, and Junyi Jessy Li. 2019. Domain agnostic real-valued specificity prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Samuli Laine and Timo Aila. 2017. Temporal ensembling for semi-supervised learning. In *International Conference for Learning Representations*.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2015. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440.

Scott de Marchi, Spencer Dorsey, and Michael J Ensley. 2018. Policy and the structure of roll call voting in the us house. *Available at SSRN 3262316*.

Maxwell McCombs. 2018. *Setting the agenda: Mass media and public opinion*. John Wiley & Sons.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training GANs. In *Advances in Neural Information Processing Systems 29*, pages 2234–2242.

Evan Sandhaus. 2008. The New York Times annotated corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Antti Tarvainen and Harri Valpola. 2017. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in Neural Information Processing Systems 30*, pages 1195–1204.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008.

Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. 2017. Improved variational autoencoders for text modeling using dilated convolutions. In *Proceedings of the 34th International Conference on Machine Learning*, pages 3881–3890.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

John R Zaller et al. 1992. *The nature and origins of mass opinion*. Cambridge University Press.

Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2017. Aspect-augmented adversarial networks for domain adaptation. *Transactions of the Association for Computational Linguistics*, 5:515–528.

## A LDA Topic Model

We experimented with a range of hyperparameters to ensure the Latent Dirichlet Allocation (LDA) model was best optimized for our datasets, leveraging the Gensim[3] library. In particular, we removed all stopwords, extremely rare words (tail 10-20% from a unigram distribution), and set the number of topics to 50.

## B Self-Ensembling

The core intuition behind consistency regularization is that ensembled predictions are more likely to be correct than single predictions (Laine and Aila, 2017; Tarvainen and Valpola, 2017). To this end, Laine and Aila (2017) introduce a **student** and **teacher** network that yield single predictions and ensembled predictions, respectively.

After learning from labeled samples, the student may produce varying, dissimilar predictions for unlabeled samples due to the stochastic nature of optimization. One potential solution is to ensemble predictions across time to converge at the *most likely* prediction (Laine and Aila, 2017). Tarvainen and Valpola (2017) improve upon this method by showing that ensembling parameters (as opposed to predictions) results in better predictions. Because the teacher's parameters are smoothed with the student's learned parameters at each iteration, the teacher effectively becomes an ensemble of the student across time.

Further, to ensure that the features learned from the labeled samples are compatible with the unlabeled samples, Laine and Aila (2017); Tarvainen and Valpola (2017); French et al. (2018) motivate a consistency-enforcing approach to bring the student and teacher's predictions closer together. In essence, if a feature learned from samples in the labeled domain is incompatible with samples in the unlabeled domain, the consistency (unsupervised) loss penalizes its incompatibility. Therefore, the interplay between these two networks creates a robust, domain-invariant feature space that characterizes both labeled and unlabeled samples (French et al., 2018). A detailed visualization of the training procedure is presented in Figure 1 in the main body of this paper.

[3] https://radimrehurek.com/gensim/

| | Political | | | Non-Political |
|---|---|---|---|---|
| | AG | PE | IR | |
| Train | 333 | 8 | 156 | 497 |
| Dev | 82 | 1 | 33 | 116 |
| Test | 125 | 8 | 47 | 208 |

Table 6: Distribution of train (In-Domain benchmark *only*), dev, test documents in our expert-annotated COHA subcorpus. For political documents, we break down the distribution into American Government (AG), Political Economy (PE), and International Relations (IR).

## C NYT Descriptors

We build a list of "political" descriptors in NYT to determine (a) which labels we can or cannot sample non-political documents from; and (b) which descriptors fall under the three areas of political science we consider for our multi-label task (American Government, Political Economy, and International Relations).

Because documents can be tagged with multiple descriptors, we build a list of descriptors whose documents have significant overlap with US POLITICS & GOVERNMENT. The second author, a political science graduate student, filtered this list to 57 descriptors that are political in nature.

For (a), we sample 4,600 non-political documents whose descriptors do not overlap with the 57 political descriptors described above. For (b), the same political science graduate student assigns each descriptor with one or more area labels. We use this label information to build an NYT dataset for our tasks. The 57 political descriptors and their corresponding area labels are tabulated in Table 7.

## D Expert-Annotated Dataset

To create an initial COHA subcorpus of 56,000 documents (8,000 per decade), we sample from the following news sources that consistently appear in across decades: Chicago Tribune, Christian Science Monitor, New York Times, Time Magazine, and Wall Street Journal. Note that these NYT articles (up to year 1986) do not appear in the NYT annotated corpus (Sandhaus, 2008) (starting from year 1987), which we used as our source, training dataset.

From this subcorpus, we perform additional steps to create an expert-annotated dataset (§5). Label distributions for our dataset are presented in Table 6. Although political economy (PE) is

severely underrepresented, we experimentally find that these documents have salient features and are not as difficult to classify. In addition, we employ class imbalance penalties to prevent our model from ignoring these documents.

The source dataset (NYT) was already annotated; to ensure label agreement with our target dataset (COHA), we sampled documents from the source dataset and had our political science graduate students label them to compare against the original label. There were minimal problems here—because NYT has fine-grained labels for their documents, the politically-labeled articles were clearly political and vice-versa.

The target datatset (COHA) was divided into halves and each political science graduate student annotated a half. Prior to annotation, they agreed upon a set of rules to minimize bias in the annotation process. In addition, both of them worked side-by-side during all annotation periods, so they were able to ask each other's opinion in case there was confusion. We also took measures to ensure label correctness after annotation was completed. Each political science graduate student sampled a batch of their political and non-political annotations and sent it to the other to evaluate. Again, there was not much disagreement here as the rules decided upon in the beginning were sufficient to cover most edge cases. Quantitatively, Cohen's $\kappa = 0.95$ as calculated on a mutually annotated subset (Cohen, 1960).

| | Area Label | | |
|---|---|---|---|
| **Topic** | **AG** | **PE** | **IR** |
| International Relations | | | ✓ |
| Presidents and Presidency (US) | ✓ | | |
| Presidential Elections (US) | ✓ | | |
| War and Revolution | | | ✓ |
| Presidential Election of 2000 | ✓ | | |
| Presidential Election of 2004 | ✓ | | |
| Law and Legislation | ✓ | | |
| Civil War and Guerrilla Warfare | | | ✓ |
| International Trade and World Market | | ✓ | |
| Presidential Election of 1996 | ✓ | | |
| Public Opinion | | | |
| Economic Conditions and Trends | | ✓ | |
| Bombs and Explosives | | | ✓ |
| Arms Sales Abroad | | | ✓ |
| United States Economy | | ✓ | |
| Missiles and Missile Defense Systems | | | ✓ |
| Oil (Petroleum) and Gasoline | | ✓ | |
| Appointments and Executive Changes | ✓ | | |
| Foreign Service | | | ✓ |
| Prisoners of War | | | ✓ |
| War Crimes, Genocide and Crimes Against Humanity | | | ✓ |
| Vice Presidents and Vice Presidency (US) | ✓ | | |
| Arms Control and Limitation and Disarmament | | | ✓ |
| Military Bases and Installations | | | ✓ |
| Presidential Election of 2008 | ✓ | | |
| Whitewater Case | ✓ | | |
| Vietnam War | ✓ | | ✓ |
| Governors (US) | ✓ | | |
| Energy and Power | | ✓ | |
| Stocks and Bonds | | ✓ | |
| State of the Union Message (US) | ✓ | | |
| Wages and Salaries | | ✓ | |
| Church-State Relations | ✓ | | |
| Shiite Muslims | | | ✓ |
| Special Prosecutors (Independent Counsel) | ✓ | | |
| White House (Washington, DC) | ✓ | | |
| Federal Taxes (US) | | ✓ | |
| Illegal Aliens | ✓ | | |
| Social Security (US) | ✓ | | |
| Political Prisoners | ✓ | | ✓ |
| Watergate Affair | ✓ | | |
| Government Employees | ✓ | | |
| Sunni Muslims | | | ✓ |
| Third World and Developing Countries | | | ✓ |
| Customs (Tariff) | | ✓ | |
| Welfare (US) | | ✓ | |
| Gun Control | ✓ | | |
| Global Warming | ✓ | | |
| Interest Rates | | ✓ | |
| Vetoes (US) | ✓ | | |
| Futures and Options Trading | | ✓ | |
| Attorneys General | ✓ | | |
| Layoffs and Job Reductions | | ✓ | |
| Nazi Policies Toward Jews and Minorities | | | ✓ |
| Government Bonds | | ✓ | |
| Police Brutality and Misconduct | ✓ | | |
| Executive Privilege, Doctrine of | ✓ | | |

Table 7: Political descriptors in NYT. Each descriptor is categorized under one or more political science areas: American Government (AG), Political Economy (PE), and International Relations (IR).

This page is intentionally left blank due to a retracted paper.

# A Hierarchical Location Prediction Neural Network for Twitter User Geolocation

**Binxuan Huang**
School of Computer Science
Carnegie Mellon University
binxuanh@cs.cmu.edu

**Kathleen M. Carley**
School of Computer Science
Carnegie Mellon University
kathleen.carley@cs.cmu.edu

## Abstract

Accurate estimation of user location is important for many online services. Previous neural network based methods largely ignore the hierarchical structure among locations. In this paper, we propose a hierarchical location prediction neural network for Twitter user geolocation. Our model first predicts the home country for a user, then uses the country result to guide the city-level prediction. In addition, we employ a character-aware word embedding layer to overcome the noisy information in tweets. With the feature fusion layer, our model can accommodate various feature combinations and achieves state-of-the-art results over three commonly used benchmarks under different feature settings. It not only improves the prediction accuracy but also greatly reduces the mean error distance.

## 1 Introduction

Accurate estimation of user location is an important factor for many online services, such as recommendation systems (Quercia et al., 2010), event detection (Sakaki et al., 2010), and disaster management (Carley et al., 2016). Though internet service providers can directly obtain users' location information from some explicit metadata like IP address and GPS signal, such private information is not available for third-party contributors. With this motivation, researchers have developed location prediction systems for various platforms, such as Wikipedia (Overell, 2009), Facebook (Backstrom et al., 2010), and Twitter (Han et al., 2012).

In the case of Twitter, due to the sparsity of geotagged tweets (Graham et al., 2014) and the unreliability of user self-declared home location in profile (Hecht et al., 2011), there is a growing body of research trying to determine users' locations automatically. Various methods have been proposed for this purpose. They can be roughly divided into three categories. The first type consists of tweet text-based methods, where the word distribution is used to estimate geolocations of users (Roller et al., 2012; Wing and Baldridge, 2011). In the second type, methods combining metadata features such as time zone and profile description are developed to improve performance (Han et al., 2013). Network-based methods form the last type. Several studies have shown that incorporating friends' information is very useful for this task (Miura et al., 2017; Ebrahimi et al., 2018). Empirically, models enhanced with network information work better than the other two types, but they do not scale well to larger datasets (Rahimi et al., 2015a).

In recent years, neural network based prediction methods have shown great success on this Twitter user geolocation prediction task (Rahimi et al., 2017; Miura et al., 2017). However, these neural network based methods largely ignore the hierarchical structure among locations (eg. country versus city), which have been shown to be very useful in previous study (Mahmud et al., 2012; Wing and Baldridge, 2014). In recent work, Huang and Carley (2017) also demonstrate that country-level location prediction is much easier than city-level location prediction. It is natural to ask whether we can incorporate the hierarchical structure among locations into a neural network and use the coarse-grained location prediction to guide the fine-grained prediction. Besides, most of these previous work uses word-level embeddings to represent text, which may not be sufficient for noisy text from social media.

In this paper, we present a hierarchical location prediction neural network (HLPNN) for user geolocation on Twitter. Our model combines text features, metadata features (personal description, profile location, name, user language, time zone), and network features together. It uses a character-aware word embedding layer to deal with the noisy



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4732–4742,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

text and capture out-of-vocabulary words. With transformer encoders, our model learns the correlation between different feature fields and outputs two classification representations for country-level and city-level predictions respectively. It first computes the country-level prediction, which is further used to guide the city-level prediction. Our model is flexible in accommodating different feature combinations, and it achieves state-of-the-art results under various feature settings.

## 2 Related Work

Because of insufficient geotagged data (Graham et al., 2014; Huang and Carley, 2019), there is a growing interest in predicting Twitter users' locations. Though there are some potential privacy concerns, user geolocation is a key factor for many important applications such as earthquake detection (Earle et al., 2012), and disaster management (Carley et al., 2016), health management (Huang and Carley, 2018).

Early work tried to identify users' locations by mapping their IP addresses to physical locations (Buyukokkten et al., 1999). However, such private information is only accessible to internet service providers. There is no easy way for a third-party to find Twitter users' IP addresses. Later, various text-based location prediction systems were proposed. Bilhaut et al. (2003) utilize a geographical gazetteer as an external lexicon and present a rule-based geographical references recognizer. Amitay et al. (2004) extracted location-related information listed in a gazetteer from web content to identify geographical regions of webpages. However, as shown in (Berggren et al., 2016), performances of gazetteer-based methods are hindered by the noisy and informal nature of tweets.

Moving beyond methods replying on external knowledge sources (e.g. IP and gazetteers), many machine learning based methods have recently been applied to location prediction. Typically, researchers first represent locations as earth grids (Wing and Baldridge, 2011; Roller et al., 2012), regions (Miyazaki et al., 2018; Qian et al., 2017), or cities (Han et al., 2013). Then location classifiers are built to categorize users into different locations. Han et al. (2012) first utilized feature selection methods to find location indicative words, then they used multinomial naive Bayes and logistic regression classifiers to find correct locations. Han et al. (2013) further present a stacking based method that combines tweet text and metadata together. Along with these classification methods, some approaches also try to learn topic regions automatically by topic modeling, but these do not scale well to the magnitude of social media (Hong et al., 2012; Zhang et al., 2017).

Recently, deep neural network based methods are becoming popular for location prediction (Miura et al., 2016). Huang and Carley (2017) integrate text and user profile metadata into a single model using convolutional neural networks, and their experiments show superior performance over stacked naive Bayes classifiers. Miura et al. (2017); Ebrahimi et al. (2018) incorporate user network connection information into their neural models, where they use network embeddings to represent users in a social network. Rahimi et al. (2018) also uses text and network feature together, but their approach is based on graph convolutional neural networks.

Similar to our method, some research has tried to predict user location hierarchically (Mahmud et al., 2012; Wing and Baldridge, 2014). Mahmud et al. (2012) develop a two-level hierarchical location classifier which first predicts a coarse-grained location (country, time zone), and then predicts the city label within the corresponding coarse region. Wing and Baldridge (2014) build a hierarchical tree of earth grids. The probability of a final fine-grained location can be computed recursively from the root node to the leaf node. Both methods have to train one classifier separately for each parent node, which is quite time-consuming for training deep neural network based methods. Additionally, certain coarse-grained locations may not have enough data samples to train a local neural classifier alone. Our hierarchical location prediction neural network overcomes these issues and only needs to be trained once.

## 3 Method

There are seven features we want to utilize in our model — tweet text, personal description, profile location, name, user language, time zone, and mention network. The first four features are text fields where users can write anything they want. User language and time zone are two categorical features that are selected by users in their profiles. Following previous work (Rahimi et al., 2018), we construct mention network directly from mentions in tweets, which is also less expensive to collect

than following network[1].

The overall architecture of our hierarchical location prediction model is shown in Figure 1. It first maps four text features into a word embedding space. A bidirectional LSTM (Bi-LSTM) neural network (Hochreiter and Schmidhuber, 1997) is used to extract location-specific features from these text embedding vectors. Following Bi-LSTM, we use a word-level attention layer to generate representation vectors for these text fields. Combining all the text representations, a user language embedding, a timezone embedding, and a network embedding, we apply several layers of transformer encoders (Vaswani et al., 2017) to learn the correlation among all the feature fields. The probability for each country is computed after a field-level attention layer. Finally, we use the country probability as a constraint for the city-level location prediction. We elaborate details of our model in the following sections.

### 3.1 Word Embedding

Assume one user has $T$ tweets, there are $T+3$ text fields for this user including personal description, profile location, and name. We first map each word in these $T+3$ text fields into a low dimensional embedding space. The embedding vector for word $w$ is computed as $x_w = [E(w), CNN_c(w)]$, where $[,]$ denotes vector concatenation. $E(w)$ is the word-level embedding retrieved directly from an Embedding matrix $E \in R^{V \times D}$ by a lookup operation, where $V$ is the vocabulary size, and $D$ is the word-level embedding dimension. $CNN_c(w)$ is a character-level word embedding that is generated from a character-level convolutional layer. Using character-level word embeddings is helpful for dealing with out-of-vocabulary tokens and overcoming the noisy nature of tweet text.

The character-level word embedding generation process is as follows. For a character $c_i$ in the word $w = (c_1, ..., c_k)$, we map it into a character embedding space and get a vector $v_{c_i} \in R^d$. In the convolutional layer, each filter $u \in R^{l_c \times d}$ generates a feature vector $\boldsymbol{\theta} = [\theta_1, \theta_2, ..., \theta_{k-l_c+1}] \in R^{k-l_c+1}$, where $\theta_i = \textit{relu}(u \circ v_{c_i:c_{i+l_c-1}} + b)$. $b$ is a bias term, and "$\circ$" denotes element-wise inner product between $u$ and character window $v_{c_i:c_{i+l_c-1}} \in R^{l_c \times d}$. After this convolutional operation, we use a max-pooling operation to select the most representative feature $\hat{\theta} = max(\boldsymbol{\theta})$. With $D$ such filters, we get the character-level word embedding $CNN_c(w) \in R^D$.

### 3.2 Text Representation

After the word embedding layer, every word in these $T+3$ texts are transformed into a $2D$ dimension vector. Given a text with word sequence $(w_1, ..., w_N)$, we get a word embedding matrix $X \in R^{N \times 2D}$ from the embedding layer. We then apply a Bi-LSTM neural network to extract high-level semantic representations from text embedding matrices.

At every time step $i$, a forward LSTM takes the word embedding $x_i$ of word $w_i$ and previous state $\overrightarrow{h_{i-1}}$ as inputs, and generates the current hidden state $\overrightarrow{h}_i$. A backward LSTM reads the text from $w_N$ to $w_1$ and generates another state sequence. The hidden state $h_i \in R^{2D}$ for word $w_i$ is the concatenation of $\overrightarrow{h_i}$ and $\overleftarrow{h_i}$. Concatenating all the hidden states, we get a semantic matrix $H \in R^{N \times 2D}$

$$\overrightarrow{h_i} = \overrightarrow{LSTM}(x_i, \overrightarrow{h_{i-1}})$$
$$\overleftarrow{h_i} = \overleftarrow{LSTM}(x_i, \overleftarrow{h_{i+1}})$$

Because not all words in a text contribute equally towards location prediction, we further use a multi-head attention layer (Vaswani et al., 2017) to generate a representation vector $f \in R^{2D}$ for each text. There are $h$ attention heads that allow the model to attend to important information from different representation subspaces. Each head computes a text representation as a weighted average of these word hidden states. The computation steps in a multi-head attention layer are as follows.

$$f = \text{MultiHead}(q, H) = [\text{head}_1, ..., \text{head}_\text{h}]W^O$$
$$\text{head}_\text{i}(q, H) = \text{softmax}(\frac{qW_i^Q \cdot (HW_i^K)^T}{\sqrt{d_k}})HW_i^V$$

where $q \in R^{2d}$ is an attention context vector learned during training, $W_i^Q, W_i^K, W_i^V \in R^{2D \times d_k}$, and $W^O \in R^{2D \times 2D}$ are projection parameters, $d_k = 2D/h$. An attention head $head_i$ first projects the attention context $q$ and the semantic matrix $H$ into query and key subspaces by $W_i^Q$, $W_i^K$ respectively. The matrix product between query $qW_i^Q$ and key $HW_i^K$ after softmax normalization is an attention weight that indicates important words among the projected value vectors $HW_i^V$. Concatenating $h$ heads together, we

[1] https://developer.twitter.com

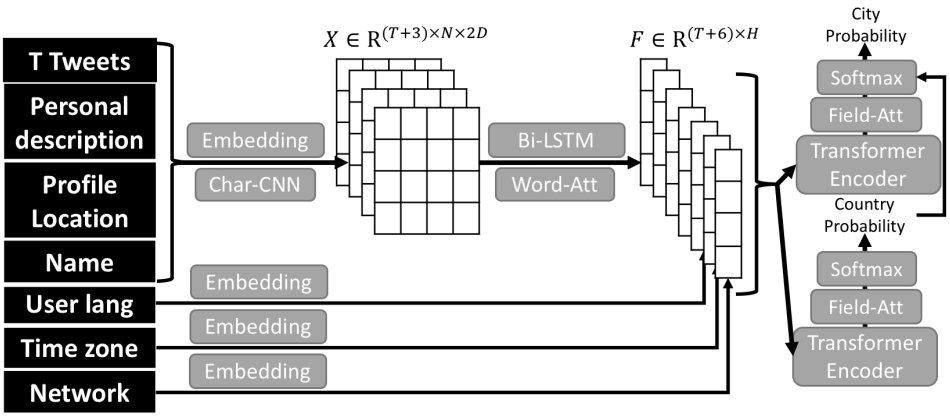


Figure 1: The architecture of our hierarchical location prediction neural network.

get one representation vector $f \in R^{2D}$ after projection by $W^O$ for each text field.

### 3.3 Feature Fusion

For two categorical features, we assign an embedding vector with dimension $2D$ for each time zone and language. These embedding vectors are learned during training. We pretrain network embeddings for users involved in the mention network using LINE (Tang et al., 2015). Network embeddings are fixed during training. We get a feature matrix $F \in R^{(T+6)\times 2D}$ by concatenating text representations of $T+3$ text fields, two embedding vectors of categorical features, and one network embedding vector.

We further use several layers of transformer encoders (Vaswani et al., 2017) to learn the correlation between different feature fields. Each layer consists of a multi-head self-attention network and a feed-forward network (FFN). One transformer encoder layer first uses input feature to attend important information in the feature itself by a multi-head attention sub-layer. Then a linear transformation sub-layer $FFN$ is applied to each position identically. Similar to Vaswani et al. (2017), we employ residual connection (He et al., 2016) and layer normalization (Ba et al., 2016) around each of the two sub-layers. The output $F_1$ of the first transformer encoder layer is generated as follows.

$$F' = \mathrm{LayerNorm}(\mathrm{MultiHead}(F, F) + F)$$
$$F_1 = \mathrm{LayerNorm}(\mathrm{FFN}(F') + F')$$

where $\mathrm{FFN}(F') = \max(0, F'W_1 + b_1)W_2 + b2$, $W_1 \in R^{2D\times D_{ff}}$, and $W_2 \in R^{D_{ff}\times 2D}$.

Since there is no position information in the transformer encoder layer, our model cannot distinguish between different types of features, eg. tweet text and personal description. To overcome this issue, we add feature type embeddings to the input representations $F$. There are seven features in total. Each of them has a learned feature type embedding with dimension $2D$ so that one feature type embedding and the representation of the corresponding feature can be summed.

Because the input and the output of transformer encoder have the same dimension, we stack $L$ layers of transformer encoders to learn representations for country-level prediction and city-level prediction respectively. These two sets of encoders share the same input $F$, but generate different representations $F_{co}^L$ and $F_{ci}^L$ for country and city predictions.

The final classification features for country-level and city-level location predictions are the row-wise weighted average of $F_{co}$ and $F_{ci}$. Similar to the word-level attention, we use a field-level multi-head attention layer to select important features from $T+6$ vectors and fuse them into a single vector.

$$F_{co} = \mathrm{MultiHead}(q_{co}, F_{co}^L)$$
$$F_{ci} = \mathrm{MultiHead}(q_{ci}, F_{ci}^L)$$

where $q_{co}, q_{ci} \in R^{2D}$ are two attention context vectors.

### 3.4 Hierarchical Location Prediction

The final probability for each country is computed by a softmax function

$$P_{co} = \mathrm{softmax}(W_{co}F_{co} + b_{co})$$

where $W_{co} \in R^{M_{co}\times 2D}$ is a linear projection parameter, $b_{co} \in R^{M_{co}}$ is a bias term, and $M_{co}$ is the number of countries.

| | Twitter-US | | | Twitter-World | | | WNUT | | |
|---|---|---|---|---|---|---|---|---|---|
| | Train | Dev. | Test | Train | Dev. | Test | Train | Dev. | Test |
| # users | 429K | 10K | 10K | 1.37M | 10K | 10K | 742K | 7.46K | 10K |
| # users with meta | 228K | 5.32K | 5.34K | 917K | 6.50K | 6.48K | 742K | 7.46K | 10K |
| # tweets | 36.4M | 861K | 831K | 11.2M | 488K | 315K | 8.97M | 90.3K | 99.7K |
| # tweets per user | 84.60 | 86.14 | 83.12 | 8.16 | 48.83 | 31.59 | 12.09 | 12.10 | 9.97 |

Table 1: A brief summary of our datasets. For each dataset, we report the number of users, number of users with metadata, number of tweets, and average number of tweets per user. We collected metadata for 53% and 67% of users in Twitter-US and Twitter-World. Time zone information was not available when we collected metadata for these two datasets. About 25% of training and development users' data was inaccessible when we collected WNUT in 2017.

After we get the probability for each country, we further use it to constrain the city-level prediction

$$P_{ci} = \text{softmax}(W_{ci}F_{ci} + b_{ci} + \lambda P_{co}Bias)$$

where $W_{ci} \in R^{M_{ci} \times 2D}$ is a linear projection parameter, $b_{ci} \in R^{M_{ci}}$ is a bias term, and $M_{ci}$ is the number of cities. $Bias \in R^{M_{co} \times M_{ci}}$ is the country-city correlation matrix. If city $j$ belongs to country $i$, then $Bias_{ij}$ is 0, otherwise $-1$. $\lambda$ is a penalty term learned during training. The larger of $\lambda$, the stronger of the country constraint. In practise, we also experimented with letting the model learn the country-city correlation matrix during training, which yields similar performance.

We minimize the sum of two cross-entropy losses for country-level prediction and city-level prediction.

$$loss = -(Y_{ci} \cdot logP_{ci} + \alpha Y_{co} \cdot logP_{co})$$

where $Y_{ci}$ and $Y_{co}$ are one-hot encodings of city and country labels. $\alpha$ is the weight to control the importance of country-level supervision signal. Since a large $\alpha$ would potentially interfere with the training process of city-level prediction, we just set it as 1 in our experiments. Tuning this parameter on each dataset may further improve the performance.

## 4 Experiment Settings

### 4.1 Datasets

To validate our method, we use three widely adopted Twitter location prediction datasets. Table 1 shows a brief summary of these three datasets. They are listed as follows.

**Twitter-US** is a dataset compiled by Roller et al. (2012). It contains 429K training users, 10K development users, and 10K test users in North America. The ground truth location of each user is set to the first geotag of this user in the dataset. We assign the closest city to each user's ground truth location using the city category built by Han et al. (2012). Since this dataset only covers North America, we change the first level location prediction from countries to administrative regions (eg. state or province). The administrative region for each city is obtained from the original city category.

**Twitter-World** is a Twitter dataset covering the whole world, with 1,367K training users, 10K development users, and 10K test users (Han et al., 2012). The ground truth location for each user is the center of the closest city to the first geotag of this user. Only English tweets are included in this dataset, which makes it more challenging for a global-level location prediction task.

We downloaded these two datasets from Github [2]. Each user in these two datasets is represented by the concatenation of their tweets, followed by the geo-coordinates. We queried Twitter's API to add user metadata information to these two datasets in February 2019. We only get metadata for about 53% and 67% users in Twitter-US and Twitter-World respectively. Because of Twitter's privacy policy change, we could not get the time zone information anymore at the time of collection.

**WNUT** was released in the 2nd Workshop on Noisy User-generated Text (Han et al., 2016). The original user-level dataset consists of 1 million training users, 10K users in development set and test set each. Each user is assigned with the closest city center as the ground truth label. Because of Twitter's data sharing policy, only tweet IDs

[2]https://github.com/afshinrahimi/geomdn

of training and development data are provided. We have to query Twitter's API to reconstruct the training and development dataset. We finished our data collection around August 2017. About 25% training and development users' data cannot be accessed at that time. The full anonymized test data is downloaded from the workshop website [3].

### 4.2 Text Preprocessing & Network Construction

For all the text fields, we first convert them into lower case, then use a tweet-specific tokenizer from NLTK[4] to tokenize them. To keep a reasonable vocabulary size, we only keep tokens with frequencies greater than 10 times in our word vocabulary. Our character vocabulary includes characters that appear more than 5 times in the training corpus.

We construct user networks from mentions in tweets. For WNUT, we keep users satisfying one of the following conditions in the mention network: (1) users in the original dataset (2) users who are mentioned by two different users in the dataset. For Twitter-US and Twitter-World, following previous work (Rahimi et al., 2018), a unidirectional edge is set if two users in our dataset directly mentioned each other, or they co-mentioned another user. We remove celebrities who are mentioned by more than 10 different users from the mentioning network. These celebrities are still kept in the dataset and their network embeddings are set as 0.

### 4.3 Evaluation Metrics

We evaluate our method using four commonly used metrics listed below.

**Accuracy**: The percentage of correctly predicted home cities.

**Acc@161**: The percentage of predicted cities which are within a 161 km (100 miles) radius of true locations to capture near-misses.

**Median**: The median distance measured in kilometer from the predicted city to the true location coordinates.

**Mean**: The mean value of error distances in predictions.

### 4.4 Hyperparameter Settings

In our experiments, we initialize word embeddings with released 300-dimensional Glove vectors (Pennington et al., 2014). For words not appearing in Glove vocabulary, we randomly initialize them from a uniform distribution U(-0.25, 0.25). We choose the character embedding dimension as 50. The character embeddings are randomly initialized from a uniform distribution U(-1.0,1.0), as well as the timezone embeddings and language embeddings. These embeddings are all learned during training. Because our three datasets are sufficiently large to train our model, the learning is quite stable and performance does not fluctuate a lot.

Network embeddings are trained using LINE (Tang et al., 2015) with parameters of dimension 600, initial learning rate 0.025, order 2, negative sample size 5, and training sample size 10000M. Network embeddings are fixed during training. For users not appearing in the mention network, we set their network embedding vectors as 0.

| | Twitter-US | Twitter-World | WNUT |
|---|---|---|---|
| Batch size | 32 | 64 | 64 |
| Initial learning rate | $10^{-4}$ | $10^{-4}$ | $10^{-4}$ |
| $D$: Word embedding dimension | 300 | 300 | 300 |
| $d$: Char. embedding dimension | 50 | 50 | 50 |
| $l_c$: filter sizes in Char. CNN | 3,4,5 | 3,4,5 | 3,4,5 |
| Filter number for each size | 100 | 100 | 100 |
| $h$: number of heads | 10 | 10 | 10 |
| $L$: layers of transformer encoder | 3 | 3 | 3 |
| $\lambda$: initial penalty term | 1 | 1 | 1 |
| $\alpha$: weight for country supervision | 1 | 1 | 1 |
| $D_{ff}$: inner dimension of FFN | 2400 | 2400 | 2400 |
| Max number of tweets per user | 100 | 50 | 20 |

Table 2: A summary of hyperparameter settings of our model.

A brief summary of hyperparameter settings of our model is shown in Table 2. The initial learning rate is $10^{-4}$. If the validation accuracy on the development set does not increase, we decrease the learning rate to $10^{-5}$ and train the model for additional 3 epochs. Empirically, training terminates within 10 epochs. Penalty $\lambda$ is initialized as 1.0 and is adapted during training. We apply dropout on the input of Bi-LSTM layer and the output of two sub-layers in transformer encoders with dropout rate 0.3 and 0.1 respectively. We use the Adam update rule (Kingma and Ba, 2014)

[3] https://noisy-text.github.io/2016/geo-shared-task.html
[4] https://www.nltk.org/api/nltk.tokenize.html

| | Twitter-US | | | Twitter-World | | | WNUT | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Acc@161↑ | Median↓ | Mean↓ | Acc@161↑ | Median↓ | Mean↓ | Accuracy↑ | Acc@161↑ | Median↓ | Mean↓ |
| Text | | | | | | | | | | |
| Wing and Baldridge (2014) | 49.2 | 170.5 | 703.6 | 32.7 | 490.0 | 1714.6 | - | - | - | - |
| Rahimi et al. (2015b)* | 50 | 159 | 686 | 32 | 530 | 1724 | - | - | - | - |
| Miura et al. (2017)-TEXT | 55.6 | 110.5 | 585.1 | - | - | - | 35.4 | 50.3 | 155.8 | 1592.6 |
| Rahimi et al. (2017) | 55 | 91 | 581 | 36 | 373 | 1417 | - | - | - | - |
| HLPNN-Text | **57.1** | **89.92** | **516.6** | **40.1** | **299.1** | **1048.1** | **37.3** | **52.9** | **109.3** | **1289.4** |
| Text+Meta | | | | | | | | | | |
| Miura et al. (2017)-META | **67.2** | **46.8** | **356.3** | - | - | - | 54.7 | 70.2 | 0 | 825.8 |
| HLPNN-Meta | 61.1 | 64.3 | 454.8 | **56.4** | **86.2** | **762.1** | **57.2** | **73.1** | **0** | **572.5** |
| Text+Net | | | | | | | | | | |
| Rahimi et al. (2015a)* | 60 | 78 | 529 | 53 | 111 | 1403 | - | - | - | - |
| Rahimi et al. (2017) | 61 | 77 | 515 | 53 | 104 | 1280 | - | - | - | - |
| Miura et al. (2017)-UNET | 61.5 | 65 | 481.5 | - | - | - | **38.1** | **53.3** | **99.9** | 1498.6 |
| Do et al. (2017) | 66.2 | 45 | 433 | 53.3 | 118 | 1044 | - | - | - | - |
| Rahimi et al. (2018)-MLP-TXT+NET | 66 | 56 | 420 | 58 | **53** | 1030 | - | - | - | - |
| Rahimi et al. (2018)-GCN | 62 | 71 | 485 | 54 | 108 | 1130 | - | - | - | - |
| HLPNN-Net | **70.8** | **31.6** | **361.5** | **58.9** | 59.9 | **827.6** | 37.8 | **53.3** | 105.26 | **1297.7** |
| Text+Meta+Net | | | | | | | | | | |
| Miura et al. (2016) | - | - | - | - | - | - | 47.6 | - | 16.1 | 1122.3 |
| Jayasinghe et al. (2016) | - | - | - | - | - | - | 52.6 | - | 21.7 | 1928.8 |
| Miura et al. (2017) | 70.1 | 41.9 | 335.7 | - | - | - | 56.4 | 71.9 | **0** | 780.5 |
| HLPNN | **72.7** | **28.2** | **323.1** | **68.4** | **6.20** | **610.0** | **57.6** | **73.4** | **0** | **538.8** |

Table 3: Comparisons between our method and baselines. We report results under four different feature settings: Text, Text+Metadata, Text+Network, Text+Metadata+Network. "-" signifies that no results were published for the given dataset, "*" denotes that results are cited from Rahimi et al. (2017). Note that Miura et al. (2017) only used 279K users added with metadata in their experiments of Twitter-US.

to optimize our model. Gradients are clipped between -1 and 1. The maximum numbers of tweets per user for training and evaluating on Twitter-US are 100 and 200 respectively. We only tuned our model, learning rate, and dropout rate on the development set of WNUT.

## 5 Results

### 5.1 Baseline Comparisons

In our experiments, we evaluate our model under four different feature settings: Text, Text+Meta, Text+Network, Text+Meta+Network. HLPNN-Text is our model only using tweet text as input. HLPNN-Meta is the model that combines text and metadata (description, location, name, user language, time zone). HLPNN-Net is the model that combines text and mention network. HLPNN is our full model that uses text, metadata, and mention network for Twitter user geolocation.

We present comparisons between our model and previous work in Table 3. As shown in the table, our model outperforms these baselines across three datasets under various feature settings.

Only using text feature from tweets, our model HLPNN-Text works the best among all these text-based location prediction systems and wins by a large margin. It not only improves prediction accuracy but also greatly reduces mean error distance. Compared with a strong neural model equipped with local dialects (Rahimi et al., 2017), it increases Acc@161 by an absolute value 4% and reduces mean error distance by about 400 kilometers on the challenging Twitter-World dataset, without using any external knowledge. Its mean error distance on Twitter-World is even comparable to some methods using network feature (Do et al., 2017).

With text and metadata, HLPNN-Meta correctly predicts locations of 57.2% users in WNUT dataset, which is even better than these location prediction systems that use text, metadata, and network. Because in the WNUT dataset the ground truth location is the closest city's center, Our model achieves 0 median error when its accuracy is greater than 50%. Note that Miura et al. (2017) used 279K users added with metadata in their experiments on Twitter-US, while we use all 449K users for training and evaluation, and only 53% of them have metadata, which makes it difficult to make a fair comparison.

Adding network feature further improves our model's performances. It achieves state-of-the-art results combining all features on these three datasets. Even though unifying network information is not the focus of this paper, our model still outperforms or has comparable results to some well-designed network-based location prediction systems like (Rahimi et al., 2018). On Twitter-US dataset, our model variant HLPNN-Net achieves a

4.6% increase in Acc@161 against previous state-of-the-art methods (Do et al., 2017) and (Rahimi et al., 2018). The prediction accuracy of HLPNN-Net on WNUT dataset is similar to (Miura et al., 2017), but with a noticeable lower mean error distance.

### 5.2 Ablation Study

In this section, we provide an ablation study to examine the contribution of each model component. Specifically, we remove the character-level word embedding, the word-level attention, the field-level attention, the transformer encoders, and the country supervision signal one by one at a time. We run experiments on the WNUT dataset with text features.

| | Accuracy | Acc@161 | Median | Mean |
|---|---|---|---|---|
| HLPNN | 37.3 | 52.9 | 109.3 | 1289.4 |
| w/o Char-CNN | 36.3 | 51.0 | 130.8 | 1429.9 |
| w/o Word-Att | 36.4 | 51.5 | 130.2 | 1377.5 |
| w/o Field-Att | 37.0 | 52.0 | 121.8 | 1337.5 |
| w/o encoders | 36.8 | 52.5 | 117.4 | 1402.9 |
| w/o country | 36.7 | 52.6 | 124.8 | 1399.2 |

Table 4: An ablation study on WNUT dataset.

The performance breakdown for each model component is shown in Table 4. Compared to the full model, we can find that the character-level word embedding layer is especially helpful for dealing with noisy social media text. The word-level attention also provides performance gain, while the field-level attention only provides a marginal improvement. The reason could be the multi-head attention layers in the transformer encoders already captures important information among different feature fields. These two transformer encoders learn the correlation between features and decouple these two level predictions. Finally, using the country supervision can help model to achieve a better performance with a lower mean error distance.

### 5.3 Country Effect

To directly measure the effect of adding country-level supervision, we define a relative country error which is the percentage of city-level predictions located in incorrect countries among all misclassified city-level predictions.

$$\text{relative country error} = \frac{\#\text{ of incorrect country}}{\#\text{ of incorrect city}}$$

The lower this metric means the better one model can predict the city-level location, at least in the correct country.

We vary the weight $\alpha$ of country-level supervision signal in our loss function from 0 to 20. The larger $\alpha$ means the more important the country-level supervision during the optimization. When $\alpha$ equals 0, there is no country-level supervision in our model. As shown in Figure 2, increasing $\alpha$ would improve the relative country error from 26.2% to 23.1%, which shows the country-level supervision signal indeed can help our model predict the city-level location towards the correct country. This possibly explains why our model has a lower mean error distance when compared to other methods.

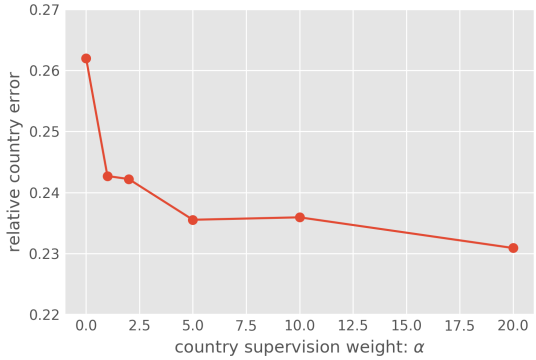


Figure 2: Relative country error with varying $\alpha$ on test dataset. Experiments were conducted on WNUT dataset with text feature.

## 6 Conclusion

In this paper, we propose a hierarchical location prediction neural network, which combines text, metadata, network information for user location prediction. Our model can accommodate various feature combinations. Extensive experiments have been conducted to validate the effectiveness of our model under four different feature settings across three commonly used benchmarks. Our experiments show our HLPNN model achieves state-of-the-art results on these three datasets. It not only improves the prediction accuracy but also significantly reduces the mean error distance. In our ablation analysis, we show that using character-aware word embeddings is helpful for overcoming noise in social media text. The transformer encoders effectively learn the correlation between different features and decouple the two different

level predictions. In our experiments, we also analyzed the effect of adding country-level regularization. The country-level supervision could effectively guide the city-level prediction towards the correct country, and reduce the errors where users are misplaced in the wrong countries.

Though our HLPNN model achieves great performances under Text+Net and Text+Meta+Net settings, potential improvements could be made using better graph-level classification frameworks. We currently only use network information to train network embeddings as user-level features. For future work, we would like to explore ways to combine graph-level classification methods and our user-level learning model. Propagating features from connected friends would provide much more information than just using network embedding vectors. Besides, our model assumes each post of one user all comes from one single home location but ignores the dynamic user movement pattern like traveling. We plan to incorporate temporal states to capture location changes in future work.

## Acknowledgments

This work was supported in part by the Office of Naval Research (ONR) N0001418SB001 and N000141812108. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the ONR.

## References

Einat Amitay, Nadav Har'El, Ron Sivan, and Aya Soffer. 2004. Web-a-where: geotagging web content. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 273–280. ACM.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Lars Backstrom, Eric Sun, and Cameron Marlow. 2010. Find me if you can: improving geographical prediction with social and spatial proximity. In *Proceedings of the 19th international conference on World wide web*, pages 61–70. ACM.

Max Berggren, Jussi Karlgren, Robert Östling, and Mikael Parkvall. 2016. Inferring the location of authors from words in their texts. *arXiv preprint arXiv:1612.06671*.

Frédérik Bilhaut, Thierry Charnois, Patrice Enjalbert, and Yann Mathet. 2003. Geographic reference analysis for geographic document querying. In *Proceedings of the HLT-NAACL 2003 workshop on Analysis of geographic references-Volume 1*, pages 55–62. Association for Computational Linguistics.

Orkut Buyukokkten, Junghoo Cho, Hector Garcia-Molina, Luis Gravano, and Narayanan Shivakumar. 1999. Exploiting geographical location information of web pages.

Kathleen M Carley, Momin Malik, Peter M Landwehr, Jürgen Pfeffer, and Michael Kowalchuck. 2016. Crowd sourcing disaster management: The complex nature of twitter usage in padang indonesia. *Safety science*, 90:48–61.

Tien Huu Do, Duc Minh Nguyen, Evaggelia Tsiligianni, Bruno Cornelis, and Nikos Deligiannis. 2017. Multiview deep learning for predicting twitter users' location. *arXiv preprint arXiv:1712.08091*.

Paul S Earle, Daniel C Bowden, and Michelle Guy. 2012. Twitter earthquake detection: earthquake monitoring in a social world. *Annals of Geophysics*, 54(6).

Mohammad Ebrahimi, Elaheh ShafieiBavani, Raymond Wong, and Fang Chen. 2018. A unified neural network model for geolocating twitter users. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 42–53.

Mark Graham, Scott A Hale, and Devin Gaffney. 2014. Where in the world are you? geolocation and language identification in twitter. *The Professional Geographer*, 66(4):568–578.

Bo Han, Paul Cook, and Timothy Baldwin. 2012. Geolocation prediction in social media data by finding location indicative words. *Proceedings of COLING 2012*, pages 1045–1062.

Bo Han, Paul Cook, and Timothy Baldwin. 2013. A stacking-based approach to twitter user geolocation prediction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 7–12.

Bo Han, Afshin Rahimi, Leon Derczynski, and Timothy Baldwin. 2016. Twitter geolocation prediction shared task of the 2016 workshop on noisy user-generated text. In *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*, pages 213–217.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Brent Hecht, Lichan Hong, Bongwon Suh, and Ed H Chi. 2011. Tweets from justin bieber's heart: the dynamics of the location field in user profiles. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 237–246. ACM.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Liangjie Hong, Amr Ahmed, Siva Gurumurthy, Alexander J Smola, and Kostas Tsioutsiouliklis. 2012. Discovering geographical topics in the twitter stream. In *Proceedings of the 21st international conference on World Wide Web*, pages 769–778. ACM.

Binxuan Huang and Kathleen M Carley. 2017. On predicting geolocation of tweets using convolutional neural networks. In *International Conference on Social Computing, Behavioral-Cultural Modeling and Prediction and Behavior Representation in Modeling and Simulation*, pages 281–291. Springer.

Binxuan Huang and Kathleen M Carley. 2018. Location order recovery in trails with low temporal resolution. *IEEE Transactions on Network Science and Engineering*.

Binxuan Huang and Kathleen M Carley. 2019. A large-scale empirical study of geotagging behavior on twitter. In *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2019*. ACM.

Gaya Jayasinghe, Brian Jin, James Mchugh, Bella Robinson, and Stephen Wan. 2016. Csiro data61 at the wnut geo shared task. In *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*, pages 218–226.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Jalal Mahmud, Jeffrey Nichols, and Clemens Drews. 2012. Where is this tweet from? inferring home locations of twitter users. In *Sixth International AAAI Conference on Weblogs and Social Media*.

Yasuhide Miura, Motoki Taniguchi, Tomoki Taniguchi, and Tomoko Ohkuma. 2016. A simple scalable neural networks based model for geolocation prediction in twitter. In *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*, pages 235–239.

Yasuhide Miura, Motoki Taniguchi, Tomoki Taniguchi, and Tomoko Ohkuma. 2017. Unifying text, metadata, and user network representations with a neural network for geolocation prediction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1260–1272.

Taro Miyazaki, Afshin Rahimi, Trevor Cohn, and Timothy Baldwin. 2018. Twitter geolocation using knowledge-based methods. In *Proceedings of the 2018 EMNLP Workshop W-NUT: The 4th Workshop on Noisy User-generated Text*, pages 7–16.

Simon E Overell. 2009. *Geographic information retrieval: Classification, disambiguation and modelling*. Ph.D. thesis, Citeseer.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Yujie Qian, Jie Tang, Zhilin Yang, Binxuan Huang, Wei Wei, and Kathleen M Carley. 2017. A probabilistic framework for location inference from social media. *arXiv preprint arXiv:1702.07281*.

Daniele Quercia, Neal Lathia, Francesco Calabrese, Giusy Di Lorenzo, and Jon Crowcroft. 2010. Recommending social events from mobile phone location data. In *2010 IEEE international conference on data mining*, pages 971–976. IEEE.

Afshin Rahimi, Trevor Cohn, and Tim Baldwin. 2018. Semi-supervised user geolocation via graph convolutional networks. *arXiv preprint arXiv:1804.08049*.

Afshin Rahimi, Trevor Cohn, and Timothy Baldwin. 2015a. Twitter user geolocation using a unified text and network prediction model. *arXiv preprint arXiv:1506.08259*.

Afshin Rahimi, Trevor Cohn, and Timothy Baldwin. 2017. A neural model for user geolocation and lexical dialectology. *arXiv preprint arXiv:1704.04008*.

Afshin Rahimi, Duy Vu, Trevor Cohn, and Timothy Baldwin. 2015b. Exploiting text and network context for geolocation of social media users. *arXiv preprint arXiv:1506.04803*.

Stephen Roller, Michael Speriosu, Sarat Rallapalli, Benjamin Wing, and Jason Baldridge. 2012. Supervised text-based geolocation using language models on an adaptive grid. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1500–1510. Association for Computational Linguistics.

Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. 2010. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*, pages 851–860. ACM.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077. International World Wide Web Conferences Steering Committee.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Benjamin Wing and Jason Baldridge. 2014. Hierarchical discriminative classification for text-based geolocation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 336–348.

Benjamin P Wing and Jason Baldridge. 2011. Simple supervised document geolocation with geodesic grids. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pages 955–964. Association for Computational Linguistics.

Yu Zhang, Wei Wei, Binxuan Huang, Kathleen M Carley, and Yan Zhang. 2017. Rate: Overcoming noise and sparsity of textual features in real-time location estimation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 2423–2426. ACM.

# Trouble on the Horizon: Forecasting the Derailment of Online Conversations as they Develop

**Jonathan P. Chang**
Cornell University
jpc362@cornell.edu

**Cristian Danescu-Niculescu-Mizil**
Cornell University
cristian@cs.cornell.edu

## Abstract

Online discussions often derail into toxic exchanges between participants. Recent efforts mostly focused on detecting antisocial behavior after the fact, by analyzing single comments in isolation. To provide more timely notice to human moderators, a system needs to preemptively detect that a conversation is heading towards derailment before it actually turns toxic. This means modeling derailment as an emerging property of a conversation rather than as an isolated utterance-level event.

Forecasting emerging conversational properties, however, poses several inherent modeling challenges. First, since conversations are dynamic, a forecasting model needs to capture the flow of the discussion, rather than properties of individual comments. Second, real conversations have an unknown horizon: they can end or derail at any time; thus a practical forecasting model needs to assess the risk in an online fashion, as the conversation develops. In this work we introduce a conversational forecasting model that learns an unsupervised representation of conversational dynamics and exploits it to predict future derailment as the conversation develops. By applying this model to two new diverse datasets of online conversations with labels for antisocial events, we show that it outperforms state-of-the-art systems at forecasting derailment.

## 1 Introduction

> "Ché saetta previsa vien più lenta."[1]
>
> – Dante Alighieri, Divina Commedia, Paradiso

Antisocial behavior is a persistent problem plaguing online conversation platforms; it is both widespread (Duggan, 2014) and potentially damaging to mental and emotional health (Raskauskas and Stoltz, 2007; Akbulut et al., 2010). The strain this phenomenon puts on community maintainers has sparked recent interest in computational approaches for assisting human moderators.

(1) **[User A]** What does [quote omitted] refer to? I assume it should be written from June 2010 to December 2011 and we should precise [sic] the months.
(2) **[User B]** No. It refers to 2007-2011 Belgian political crisis
(3) **[User A]** 2007-2011 Belgian political crisis is a little bit [of original research]. It merges 2 crisis in 1. Sources 2 and 4 in the article talk about a 18 months crisis in 2010-2011, ie what I refer to. What are the reliable sources that make this crisis go back to 2007?
(4) **[User B]** Yes it's not ridiculous at all to claim [it's original research] because it doesn't fit your argument. A crisis can be composed out of several smaller crisis. It's not original research if some of your sources only talk about parts [...]
(5) **[User A]** Where is the source that claim the crisis is 4 year long? Sources state claim it is 18 month long and refer to the period from June 2010 to December 2011.
(6) **[User B]** There were 4 governments and 2 years of no government in 4 years time. You can not sanely claim that this Must be viewed as two seperate crisis. What exactly splits them up? [...]

Figure 1: Example start of a conversation that will eventually derail into a personal attack.

Prior work in this direction has largely focused on *post-hoc* identification of various kinds of antisocial behavior, including hate speech (Warner and Hirschberg, 2012; Davidson et al., 2017), harassment (Yin et al., 2009), personal attacks (Wulczyn et al., 2017), and general toxicity (Pavlopoulos et al., 2017). The fact that these approaches only identify antisocial content *after the fact* limits their practicality as tools for assisting *pre-emptive* moderation in conversational domains.

Addressing this limitation requires *forecasting* the future derailment of a conversation based on early warning signs, giving the moderators time to potentially intervene before any harm is done (Liu et al. 2018, Zhang et al. 2018a, see Jurgens et al. 2019 for a discussion). Such a goal recognizes derailment as emerging from the devel-

[1] "The arrow one foresees arrives more gently."



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4743–4754, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

opment of the conversation, and belongs to the broader area of *conversational forecasting*, which includes future-prediction tasks such as predicting the eventual length of a conversation (Backstrom et al., 2013), whether a persuasion attempt will eventually succeed (Tan et al., 2016; Wachsmuth et al., 2018; Yang et al., 2019), whether team discussions will eventually lead to an increase in performance (Niculae and Danescu-Niculescu-Mizil, 2016), or whether ongoing counseling conversations will eventually be perceived as helpful (Althoff et al., 2016).[2]

Approaching such *conversational forecasting* problems, however, requires overcoming several inherent modeling challenges. First, conversations are *dynamic* and their outcome might depend on how subsequent comments interact with each other. Consider the example in Figure 1: while no individual comment is outright offensive, a human reader can sense a tension emerging from their succession (e.g., dismissive answers to repeated questioning). Thus a forecasting model needs to capture not only the content of each individual comment, but also the *relations* between comments. Previous work has largely relied on hand-crafted features to capture such relations—e.g., similarity between comments (Althoff et al., 2016; Tan et al., 2016) or conversation structure (Zhang et al., 2018b; Hessel and Lee, 2019)—, though neural attention architectures have also recently shown promise (Jo et al., 2018).

The second modeling challenge stems from the fact that conversations have an *unknown horizon*: they can be of varying lengths, and the to-be-forecasted event can occur at any time. So when is it a good time to make a forecast? Prior work has largely proposed two solutions, both resulting in important practical limitations. One solution is to assume (unrealistic) prior knowledge of when the to-be-forecasted event takes place and extract features up to that point (Niculae et al., 2015; Liu et al., 2018). Another compromising solution is to extract features from a fixed-length window, often at the start of the conversation (Curhan and Pentland, 2007; Niculae and Danescu-Niculescu-Mizil, 2016; Althoff et al., 2016; Zhang et al., 2018a, inter alia). Choosing a catch-all window-size is however impractical: short windows will miss information in comments they do not encompass (e.g., a window of only two comments would miss the chain of repeated questioning in comments 3 through 6 of Figure 1), while longer windows risk missing the to-be-forecasted event altogether if it occurs before the end of the window, which would prevent *early* detection.

In this work we introduce a model for forecasting conversational events that overcomes both these inherent challenges by processing comments, and their relations, as they happen (i.e., in an online fashion). Our main insight is that models with these properties already exist, albeit geared toward generation rather than prediction: recent work in context-aware dialog generation (or "chatbots") has proposed sequential neural models that make effective use of the intra-conversational dynamics (Sordoni et al., 2015b; Serban et al., 2016, 2017), while concomitantly being able to process the conversation as it develops (see Gao et al. (2018) for a survey).

In order for these systems to perform well in the generative domain they need to be trained on massive amounts of (unlabeled) conversational data. The main difficulty in directly adapting these models to the supervised domain of conversational forecasting is the relative scarcity of labeled data: for most forecasting tasks, at most a few thousands labeled examples are available, insufficient for the notoriously data-hungry sequential neural models.

To overcome this difficulty, we propose to decouple the objective of learning a neural representation of conversational dynamics from the objective of predicting future events. The former can be *pre-trained* on large amounts of unsupervised data, similarly to how chatbots are trained. The latter can piggy-back on the resulting representation after *fine-tuning* it for classification using relatively small labeled data. While similar pre-train-then-fine-tune approaches have recently achieved state-of-the-art performance in a number of NLP tasks—including natural language inference, question answering, and commonsense reasoning (discussed in Section 2)—to the best of our knowledge this is the first attempt at applying this paradigm to conversational forecasting.

To test the effectiveness of this new architecture in forecasting derailment of online conversations, we develop and distribute two new datasets. The

[2]We can distinguish two types of forecasting tasks, depending on whether the to-be-forecasted target is an event that might take place within the conversation (e.g., derailment) or an outcome measured after the conversation will eventually conclude (e.g., helpfulness). The following discussion of modeling challenges holds for both.

first triples in size the highly curated 'Conversations Gone Awry' dataset (Zhang et al., 2018a), where civil-starting Wikipedia Talk Page conversations are crowd-labeled according to whether they eventually lead to personal attacks; the second relies on in-the-wild moderation of the popular subreddit ChangeMyView, where the aim is to forecast whether a discussion will later be subject to moderator action due to "rude or hostile" behavior. In both datasets, our model outperforms existing fixed-window approaches, as well as simpler sequential baselines that cannot account for inter-comment relations. Furthermore, by virtue of its online processing of the conversation, our system can provide substantial prior notice of upcoming derailment, triggering on average 3 comments (or 3 hours) before an overtly toxic comment is posted.

To summarize, in this work we:

- introduce the first model for forecasting conversational events that can capture the dynamics of a conversation *as it develops*;
- build two diverse datasets (one entirely new, one extending prior work) for the task of forecasting derailment of online conversations;
- compare the performance of our model against the current state-of-the-art, and evaluate its ability to provide *early* warning signs.

Our work is motivated by the goal of assisting human moderators of online communities by preemptively signaling at-risk conversations that might deserve their attention. However, we caution that any automated systems might encode or even amplify the biases existing in the training data (Park et al., 2018; Sap et al., 2019; Wiegand et al., 2019), so a public-facing implementation would need to be exhaustively scrutinized for such biases (Feldman et al., 2015).

## 2 Further Related Work

**Antisocial behavior.** Antisocial behavior online comes in many forms, including harassment (Vitak et al., 2017), cyberbullying (Singh et al., 2017), and general aggression (Kayany, 1998). Prior work has sought to understand different aspects of such behavior, including its effect on the communities where it happens (Collier and Bear, 2012; Arazy et al., 2013), the actors involved (Cheng et al., 2017; Volkova and Bell, 2017; Kumar et al., 2018; Ribeiro et al., 2018) and connections to the outside world (Olteanu et al., 2018).

**Post-hoc classification of conversations.** There is a rich body of prior work on classifying the outcome of a conversation after it has concluded, or classifying conversational events after they happened. Many examples exist, but some more closely related to our present work include identifying the winner of a debate (Zhang et al., 2016; Potash and Rumshisky, 2017; Wang et al., 2017), identifying successful negotiations (Curhan and Pentland, 2007; Cadilhac et al., 2013), as well as detecting whether deception (Girlea et al., 2016; Pérez-Rosas et al., 2016; Levitan et al., 2018) or disagreement (Galley et al., 2004; Abbott et al., 2011; Allen et al., 2014; Wang and Cardie, 2014; Rosenthal and McKeown, 2015) has occurred.

Our goal is different because we wish to *forecast* conversational events before they happen and while the conversation is still ongoing (potentially allowing for interventions). Note that some post-hoc tasks can also be re-framed as forecasting tasks (assuming the existence of necessary labels); for instance, predicting whether an ongoing conversation *will* eventually spark disagreement (Hessel and Lee, 2019), rather than detecting already-existing disagreement.

**Conversational forecasting.** As described in Section 1, prior work on forecasting conversational outcomes and events has largely relied on hand-crafted features to capture aspects of conversational dynamics. Example feature sets include statistical measures based on similarity between utterances (Althoff et al., 2016), sentiment imbalance (Niculae et al., 2015), flow of ideas (Niculae et al., 2015), increase in hostility (Liu et al., 2018), reply rate (Backstrom et al., 2013) and graph representations of conversations (Garimella et al., 2017; Zhang et al., 2018b). By contrast, we aim to automatically learn neural representations of conversational dynamics through pre-training.

Such hand-crafted features are typically extracted from fixed-length windows of the conversation, leaving unaddressed the problem of unknown horizon. While some work has trained *multiple* models for different window-lengths (Liu et al., 2018; Hessel and Lee, 2019), they consider these models to be independent and, as such, do not address the issue of aggregating them into a single forecast (i.e., deciding at what point to make

a prediction). We implement a simple sliding windows solution as a baseline (Section 5).

**Pre-training for NLP.** The use of pre-training for natural language tasks has been growing in popularity after recent breakthroughs demonstrating improved performance on a wide array of benchmark tasks (Peters et al., 2018; Radford et al., 2018). Existing work has generally used a language modeling objective as the pre-training objective; examples include next-word prediction (Howard and Ruder, 2018), sentence autoencoding, (Dai and Le, 2015), and machine translation (McCann et al., 2017). BERT (Devlin et al., 2019) introduces a variation on this in which the goal is to predict the next sentence in a document given the current sentence. Our pre-training objective is similar in spirit, but operates at a *conversation* level, rather than a document level. We hence view our objective as *conversational modeling* rather than (only) language modeling. Furthermore, while BERT's sentence prediction objective is framed as a multiple-choice task, our objective is framed as a generative task.

## 3 Derailment Datasets

We consider two datasets, representing related but slightly different forecasting tasks. The first dataset is an expanded version of the annotated Wikipedia conversations dataset from Zhang et al. (2018a). This dataset uses carefully-controlled crowdsourced labels, strictly filtered to ensure the conversations are civil up to the moment of a personal attack. This is a useful property for the purposes of model analysis, and hence we focus on this as our primary dataset. However, we are conscious of the possibility that these strict labels may not fully capture the kind of behavior that moderators care about in practice. We therefore introduce a secondary dataset, constructed from the subreddit ChangeMyView (CMV) that does not use post-hoc annotations. Instead, the prediction task is to forecast whether the conversation will be subject to moderator action in the future.

**Wikipedia data.** Zhang et al.'s 'Conversations Gone Awry' dataset consists of 1,270 conversations that took place between Wikipedia editors on publicly accessible talk pages. The conversations are sourced from the WikiConv dataset (Hua et al., 2018) and labeled by crowdworkers as either containing a *personal attack* from within (i.e., hostile behavior by one user in the conversation directed towards another) or remaining civil throughout.

A series of controls are implemented to prevent models from picking up on trivial correlations. To prevent models from capturing topic-specific information (e.g., political conversations are more likely to derail), each attack-containing conversation is paired with a clean conversation from the same talk page, where the talk page serves as a proxy for topic.[3] To force models to actually capture conversational dynamics rather than detecting already-existing toxicity, human annotations are used to ensure that all comments preceding a personal attack are civil.

To the ends of more effective model training, we elected to expand the 'Conversations Gone Awry' dataset, using the original annotation procedure. Since we found that the original data skewed towards shorter conversations, we focused this crowdsourcing run on longer conversations: ones with 4 or more comments preceding the attack.[4] Through this additional crowdsourcing, we expand the dataset to 4,188 conversations, which we are publicly releasing as part of the Cornell Conversational Analysis Toolkit (ConvoKit).[5]

We perform an 80-20-20 train/dev/test split, ensuring that paired conversations end up in the same split in order to preserve the topic control. Finally, we randomly sample another 1 million conversations from WikiConv to use for the unsupervised pre-training of the generative component.

**Reddit CMV data.** The CMV dataset is constructed from conversations collected via the Reddit API. In contrast to the Wikipedia-based dataset, we explicitly avoid the use of post-hoc annotation. Instead, we use as our label whether a conversation eventually had a comment removed by a moderator for violation of Rule 2: "Don't be rude or hostile to other users".[6]

Though the lack of post-hoc annotation limits the degree to which we can impose controls on the data (e.g., some conversations may contain toxic comments not flagged by the moderators) we do reproduce as many of the Wikipedia data's controls as we can. Namely, we replicate the topic

[3] Paired conversations were also enforced to be similar in length, so that length distribution is the same between classes.

[4] We cap the length at 10 to avoid overwhelming the crowdworkers.

[5] convokit.cornell.edu

[6] The existence of this specific rule, the standardized moderation messages and the civil character of the ChangeMyView subreddit was our initial motivation for choosing it.

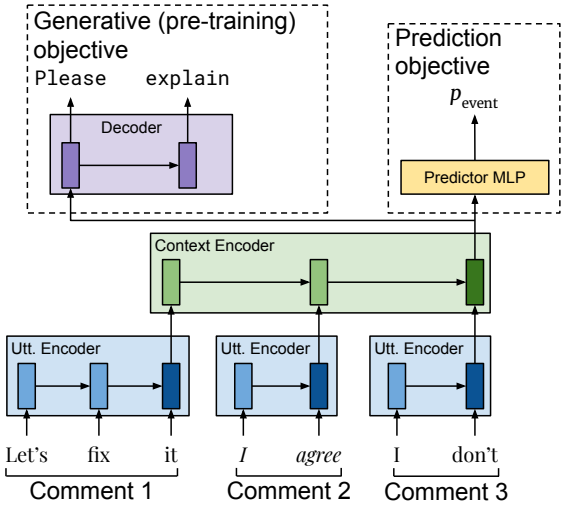


Figure 2: Sketch of the CRAFT architecture.

control pairing by choosing pairs of positive and negative examples that belong to the same top-level post, following Tan et al. (2016);[7] and enforce that the removed comment was made by a user who was previously involved in the conversation.[8] This process results in 6,842 conversations, to which we again apply a pair-preserving 80-20-20 split. Finally, we gather over 600,000 conversations that do not include any removed comment, for unsupervised pre-training.

## 4 Conversational Forecasting Model

We now describe our general model for forecasting future conversational events. Our model integrates two components: (a) a generative dialog model that learns to represent conversational dynamics in an unsupervised fashion; and (b) a supervised component that fine-tunes this representation to forecast future events. Figure 2 provides an overview of the proposed architecture, henceforth CRAFT (Conversational Recurrent Architecture for ForecasTing).

**Terminology.** For modeling purposes, we treat a conversation as a sequence of $N$ comments $C = \{c_1, \ldots, c_N\}$. Each comment, in turn, is a sequence of tokens, where the number of tokens may vary from comment to comment. For the $n$-th comment ($1 \leq n \leq N$), we let $M_n$ denote the number of tokens. Then, a comment $c_n$ can be represented as a sequence of $M_n$ tokens: $c_n = \{w_1, \ldots, w_{M_n}\}$.

[7] The top-level post is not part of the conversations.

[8] We also impose the same length restriction on the number of comments preceding the removed comment, for comparability and for computational considerations.

**Generative component.** For the generative component of our model, we use a hierarchical recurrent encoder-decoder (HRED) architecture (Sordoni et al., 2015a), a modified version of the popular sequence-to-sequence (seq2seq) architecture (Sutskever et al., 2014) designed to account for dependencies between consecutive inputs. Serban et al. (2016) showed that HRED can successfully model conversational context by encoding the temporal structure of previously seen comments, making it an ideal fit for our use case. Here, we provide a high-level summary of the HRED architecture, deferring deeper technical discussion to Sordoni et al. (2015a) and Serban et al. (2016).

An HRED dialog model consists of three components: an utterance encoder, a context encoder, and a decoder. The utterance encoder is responsible for generating semantic vector representations of comments. It consists of a recurrent neural network (RNN) that reads a comment token-by-token, and on each token $w_m$ updates a hidden state $h^{\text{enc}}$ based on the current token and the previous hidden state:

$$h_m^{\text{enc}} = f^{\text{RNN}}(h_{m-1}^{\text{enc}}, w_m) \tag{1}$$

where $f^{\text{RNN}}$ is a nonlinear gating function (our implementation uses GRU (Cho et al., 2014)). The final hidden state $h_M^{\text{enc}}$ can be viewed as a vector encoding of the entire comment.

Running the encoder on each comment $c_n$ results in a sequence of $N$ vector encodings. A second encoder, the context encoder, is then run over this sequence:

$$h_n^{\text{con}} = f^{\text{RNN}}(h_{n-1}^{\text{con}}, h_{M_n}^{\text{enc}}) \tag{2}$$

Each hidden state $h_n^{\text{con}}$ can then be viewed as an encoding of the full conversational context up to and including the $n$-th comment. To generate a response to comment $n$, the context encoding $h_n^{\text{con}}$ is used to initialize the hidden state $h_0^{\text{dec}}$ of a decoder RNN. The decoder produces a response token by token using the following recurrence:

$$\begin{aligned} h_t^{\text{dec}} &= f^{\text{RNN}}(h_{t-1}^{\text{dec}}, w_{t-1}) \\ w_t &= f^{\text{out}}(h_t^{\text{dec}}) \end{aligned} \tag{3}$$

where $f^{\text{out}}$ is some function that outputs a probability distribution over words; we implement this using a simple feedforward layer. In our implementation, we further augment the decoder with attention (Bahdanau et al., 2014; Luong et al.,

2015) over context encoder states to help capture long-term inter-comment dependencies. This generative component can be pre-trained using unlabeled conversational data.

**Prediction component.** Given a pre-trained HRED dialog model, we aim to extend the model to predict from the conversational context whether the to-be-forecasted event will occur. Our predictor consists of a multilayer perceptron (MLP) with 3 fully-connected layers, leaky ReLU activations between layers, and sigmoid activation for output. For each comment $c_n$, the predictor takes as input the context encoding $h_n^{\text{con}}$ and forwards it through the MLP layers, resulting in an output score that is interpreted as a probability $p_{\text{event}}(c_{n+1})$ that the to-be-forecasted event will happen (e.g., that the conversation will derail).

Training the predictive component starts by initializing the weights of the encoders to the values learned in pre-training. The main training loop then works as follows: for each positive sample—i.e., a conversation containing an instance of the to-be-forecasted event (e.g., derailment) at comment $c_e$—we feed the context $c_1, \ldots, c_{e-1}$ through the encoder and classifier, and compute cross-entropy loss between the classifier output and expected output of 1. Similarly, for each negative sample—i.e., a conversation where none of the comments exhibit the to-be-forecasted event and that ends with $c_N$—we feed the context $c_1, \ldots, c_{N-1}$ through the model and compute loss against an expected output of 0.

Note that the parameters of the generative component are not held fixed during this process; instead, backpropagation is allowed to go all the way through the encoder layers. This process, known as *fine-tuning*, reshapes the representation learned during pre-training to be more directly useful to prediction (Howard and Ruder, 2018).

We implement the model and training code using PyTorch, and we are publicly releasing our implementation and the trained models together with the data as part of ConvoKit.

## 5 Forecasting Derailment

We evaluate the performance of CRAFT in the task of forecasting conversational derailment in both the Wikipedia and CMV scenarios. To this end, for each of these datasets we pre-train the generative component on the unlabeled portion of the data and fine-tune it on the labeled training split (data size detailed in Section 3).

In order to evaluate our sequential system against conversational-level ground truth, we need to aggregate comment level predictions. If *any* comment in the conversation *triggers* a positive prediction—i.e., $p_{\text{event}}(c_{n+1})$ is greater than a threshold learned on the development split—then the respective conversation is predicted to derail. If this forecast is triggered in a conversation that actually derails, but *before* the derailment actually happens, then the conversation is counted as a true positive; otherwise it is a false positive. If no positive predictions are triggered for a conversation, but it actually derails then it counts as a false negative; if it does not derail then it is a true negative.

**Fixed-length window baselines.** We first seek to compare CRAFT to existing, fixed-length window approaches to forecasting. To this end, we implement two such baselines: *Awry*, which is the state-of-the-art method proposed in Zhang et al. (2018a) based on pragmatic features in the first comment-reply pair,[9] and *BoW*, a simple bag-of-words baseline that makes a prediction using TF-IDF weighted bag-of-words features extracted from the first comment-reply pair.

**Online forecasting baselines.** Next, we consider simpler approaches for making forecasts as the conversations happen (i.e., in an online fashion). First, we propose *Cumulative BoW*, a model that recomputes bag-of-words features on all comments seen thus far every time a new comment arrives. While this approach does exhibit the desired behavior of producing updated predictions for each new comment, it fails to account for relationships between comments.

This simple cumulative approach cannot be directly extended to models whose features are strictly based on a fixed number of comments, like Awry. An alternative is to use a *sliding window*: for a feature set based on a window of $W$ comments, upon each new comment we can extract features from a window containing that comment and the $W-1$ comments preceding it. We apply this to the Awry method and call this model *Sliding Awry*. For both these baselines, we aggregate comment-level predictions in the same way as in our main model.

**CRAFT ablations.** Finally, we consider two modified versions of the CRAFT model in order

[9] We use the ConvoKit implementation.

| | Capabilities | | | Wikipedia Talk Pages | | | | | Reddit CMV | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Model** | **D** | **O** | **L** | **A** | **P** | **R** | **FPR** | **F1** | **A** | **P** | **R** | **FPR** | **F1** |
| BoW | | | | 56.5 | 55.6 | 65.5 | 52.4 | 60.1 | 52.1 | 51.8 | 61.3 | 57.0 | 56.1 |
| Awry | ✓ | | | 58.9 | 59.2 | 57.6 | 39.8 | 58.4 | 54.4 | 55.0 | 48.3 | **39.5** | 51.4 |
| Cumul. BoW | | ✓ | | 60.6 | 57.7 | **79.3** | 58.1 | 66.8 | 59.9 | 58.8 | 65.9 | 46.2 | 62.1 |
| Sliding Awry | ✓ | ✓ | | 60.6 | 60.2 | 62.4 | 41.2 | 61.3 | 56.8 | 56.6 | 58.2 | 44.6 | 57.4 |
| CRAFT − CE | | ✓ | ✓ | 64.9 | **64.4** | 66.7 | **36.9** | 65.5 | 57.7 | 56.1 | 71.2 | 55.7 | 62.8 |
| CRAFT | ✓ | ✓ | ✓ | **66.5** | 63.7 | 77.1 | 44.1 | **69.8** | **63.4** | **60.4** | **77.5** | 50.7 | **67.9** |

Table 1: Comparison of the capabilities of each baseline and our CRAFT models (full and without the Context Encoder) with regards to capturing inter-comment (D)ynamics, processing conversations in an (O)nline fashion, and automatically (L)earning feature representations, as well as their performance in terms of (A)ccuracy, (P)recision, (R)ecall, False Positive Rate (FPR), and F1 score. Awry is the model previously proposed by Zhang et al. (2018a) for this task.

to evaluate the impact of two of its key components: (1) the pre-training step, and (2) its ability to capture inter-comment dependencies through its hierarchical memory.

To evaluate the impact of pre-training, we train the prediction component of CRAFT on only the labeled training data, without first pre-training the encoder layers with the unlabeled data. We find that given the relatively small size of labeled data, this baseline fails to successfully learn, and ends up performing at the level of random guessing.[10] This result underscores the need for the pre-training step that can make use of unlabeled data.

To evaluate the impact of the hierarchical memory, we implement a simplified version of CRAFT where the memory size of the context encoder is zero (*CRAFT − CE*), thus effectively acting as if the pre-training component is a vanilla seq2seq model. In other words, this model cannot capture inter-comment dependencies, and instead at each step makes a prediction based only on the utterance encoding of the latest comment.

**Results.** Table 1 compares CRAFT to the baselines on the test splits (random baseline is 50%) and illustrates several key findings. First, we find that unsurprisingly, accounting for full conversational context is indeed helpful, with even the simple online baselines outperforming the fixed-window baselines. On both datasets, CRAFT outperforms all baselines (including the other online models) in terms of accuracy and F1. Furthermore, although it loses on precision (to CRAFT − CE) and recall (to Cumulative BoW) individually on the Wikipedia data, CRAFT has the superior *balance* between the two, having both a visibly higher precision-recall curve and larger area under the curve (AUPR) than the baselines (Figure 3). This latter property is particularly useful in a practical setting, as it allows moderators to tune model performance to some desired precision without having to sacrifice as much in the way of recall (or vice versa) compared to the baselines and pre-existing solutions.

[10] We thus exclude this baseline from the results summary.

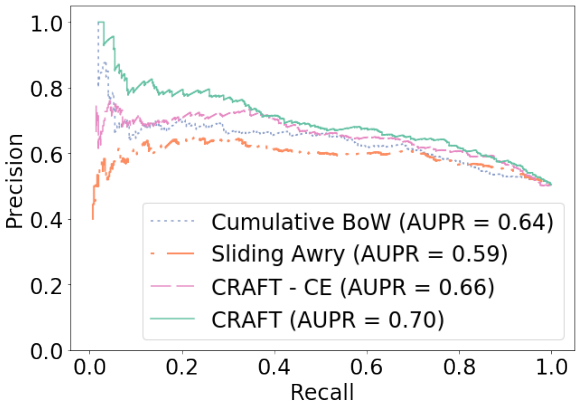


Figure 3: Precision-recall curves and the area under each curve. To reduce clutter, we show only the curves for Wikipedia data (CMV curves are similar) and exclude the fixed-length window baselines (which perform worse).

## 6 Analysis

We now examine the behavior of CRAFT in greater detail, to better understand its benefits and limitations. We specifically address the following questions: (1) How much early warning does the the model provide? (2) Does the model actually

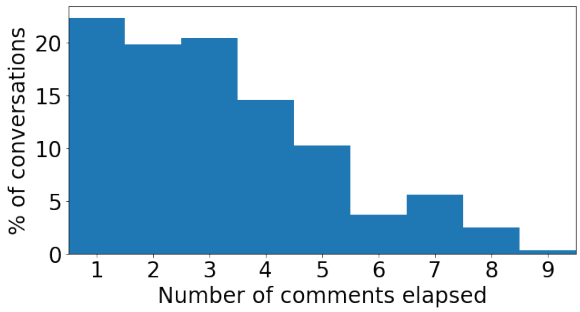


Figure 4: Distribution of number of comments elapsed between the model's first warning and the attack.

learn an order-sensitive representation of conversational context?[11]

**Early warning, but how early?** The recent interest in forecasting antisocial behavior has been driven by a desire to provide pre-emptive, actionable warning to moderators. But does our model trigger early enough for any such practical goals?

For each personal attack correctly forecasted by our model, we count the number of comments elapsed between the time the model is first triggered and the attack. Figure 4 shows the distribution of these counts: on average, the model warns of an attack 3 comments before it actually happens (4 comments for CMV). To further evaluate how much time this early warning would give to the moderator, we also consider the difference in timestamps between the comment where the model first triggers and the comment containing the actual attack. Over 50% of conversations get at least 3 hours of advance warning (2 hours for CMV). Moreover, 39% of conversations get at least *12 hours* of early warning before they derail.

**Does order matter?** One motivation behind the design of our model was the intuition that comments in a conversation are not independent events; rather, the order in which they appear matters (e.g., a blunt comment followed by a polite one feels intuitively different from a polite comment followed by a blunt one). By design, CRAFT has the capacity to learn an order-sensitive representation of conversational context, but how can we know that this capacity is actually used? It is conceivable that the model is simply computing an order-insensitive "bag-of-features". Neural network models are notorious for their lack of transparency, precluding an analysis of how *exactly* CRAFT models conversational context. Nevertheless, through two simple exploratory experiments, we seek to show that it does not completely ignore comment order.

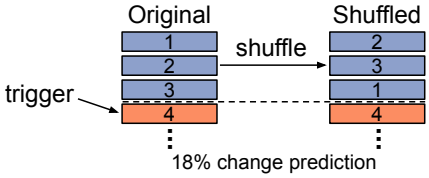


Figure 5: The prefix-shuffling procedure ($t = 4$).

The first experiment for testing whether the model accounts for comment order is a *prefix-shuffling* experiment, visualized in Figure 5. For each conversation that the model predicts will derail, let $t$ denote the index of the triggering comment, i.e., the index where the model first made a derailment forecast. We then construct *synthetic* conversations by taking the first $t-1$ comments (henceforth referred to as the *prefix*) and randomizing their order.[12] Finally, we count how often the model no longer predicts derailment at index $t$ in the synthetic conversations. If the model were ignoring comment order, its prediction should remain unchanged (as it remains for the Cumulative BoW baseline), since the actual *content* of the first $t$ comments has not changed (and CRAFT inference is deterministic). We instead find that in roughly one fifth of cases (12% for CMV) the model changes its prediction on the synthetic conversations. This suggests that CRAFT learns an order-sensitive representation of context, not a mere "bag-of-features".

To more concretely quantify how much this order-sensitive context modeling helps with prediction, we can actively prevent the model from learning and exploiting any order-related dynamics. We achieve this through another type of shuffling experiment, where we go back even further and shuffle the comment order in the conversations used for pre-training, fine-tuning and testing. This procedure preserves the model's ability to capture signals present within the individual comments processed so far, as the utterance encoder is unaffected, but inhibits it from capturing any meaningful order-sensitive dynamics. We find that this hurts the model's performance (65% ac-

[11] We choose to focus on the Wikipedia scenario since the conversational prefixes are hand-verified to be civil. For completeness we also report results for Reddit CMV throughout, but they should be taken with an additional grain of salt.

[12] We restrict the experiment to cases where $t \geq 3$, as prefixes consisting of only one comment cannot be reordered.

curacy for Wikipedia, 59.5% for CMV), lowering it to a level similar to that of the version where we completely disable the context encoder.

Taken together, these experiments provide evidence that CRAFT uses its capacity to model conversational context in an order-sensitive fashion, and that it makes effective use of the dynamics within. An important avenue for future work would be developing more transparent models that can shed light on exactly *what* kinds of order-related features are being extracted and *how* they are used in prediction.

## 7 Conclusions and Future Work

In this work, we introduced a model for forecasting conversational events that processes comments as they happen and takes the full conversational context into account to make an updated prediction at each step. This model fills a void in the existing literature on conversational forecasting, simultaneously addressing the dual challenges of capturing inter-comment dynamics and dealing with an unknown horizon. We find that our model achieves state-of-the-art performance on the task of forecasting derailment in two different datasets that we release publicly. We further show that the resulting system can provide substantial prior notice of derailment, opening up the potential for preemptive interventions by human moderators (Seering et al., 2017).

While we have focused specifically on the task of forecasting derailment, we view this work as a step towards a more general model for real-time forecasting of other types of emergent properties of conversations. Follow-up work could adapt the CRAFT architecture to address other forecasting tasks mentioned in Section 2—including those for which the outcome is extraneous to the conversation. We expect different tasks to be informed by different types of inter-comment dynamics, and further architecture extensions could add additional supervised fine-tuning in order to direct it to focus on specific dynamics that might be relevant to the task (e.g., exchange of ideas between interlocutors or stonewalling).

With respect to forecasting derailment, there remain open questions regarding what human moderators actually desire from an early-warning system, which would affect the design of a practical system based on this work. For instance, how early does a warning need to be in order for moderators to find it useful? What is the optimal balance between precision, recall, and false positive rate at which such a system is truly improving moderator productivity rather than wasting their time through false positives? What are the ethical implications of such a system? Follow-up work could run a user study of a prototype system with actual moderators to address these questions.

A practical limitation of the current analysis is that it relies on balanced datasets, while derailment is a relatively rare event for which a more restrictive trigger threshold would be appropriate. While our analysis of the precision-recall curve suggests the system is robust across multiple thresholds ($AUPR = 0.7$), additional work is needed to establish whether the recall tradeoff would be acceptable in practice.

Finally, one major limitation of the present work is that it assigns a single label to each conversation: does it derail or not? In reality, derailment need not spell the end of a conversation; it is possible that a conversation could get back on track, suffer a repeat occurrence of antisocial behavior, or any number of other trajectories. It would be exciting to consider finer-grained forecasting of conversational trajectories, accounting for the natural—and sometimes chaotic—ebb-and-flow of human interactions.

**Acknowledgements.** We thank Caleb Chiam, Liye Fu, Lillian Lee, Alexandru Niculescu-Mizil, Andrew Wang and Justine Zhang for insightful conversations (with unknown horizon), Aditya Jha for his great help with implementing and running the crowd-sourcing tasks, Thomas Davidson and Claire Liang for exploratory data annotation, as well as the anonymous reviewers for their helpful comments. This work is supported in part by the NSF CAREER award IIS-1750615 and by the NSF Grant SES-1741441. This material is based upon work supported by Google Cloud.

## References

Rob Abbott, Marilyn Walker, Pranav Anand, Jean E. Fox Tree, Robeson Bowmani, and Joseph King. 2011. How Can You Say Such Things?!?: Recognizing Disagreement in Informal Political Argument. In *Proceedings of the Workshop on Languages in Social Media*.

Yavuz Akbulut, Yusuf Levent Sahin, and Bahadir Eristi. 2010. Cyberbullying Victimization among

Turkish Online Social Utility Members. *Educational Technology & Society*, 13(4).

Kelsey Allen, Giuseppe Carenini, and Raymond T. Ng. 2014. Detecting Disagreement in Conversations using Pseudo-Monologic Rhetorical Structure. In *Proceedings of EMNLP*.

Tim Althoff, Kevin Clark, and Jure Leskovec. 2016. Large-scale Analysis of Counseling Conversations: An Application of Natural Language Processing to Mental Health. *Transactions of the Association for Computational Linguistics*, 4.

Ofer Arazy, Lisa Yeo, and Oded Nov. 2013. Stay on the Wikipedia Task: When Task-related Disagreements Slip Into Personal and Procedural Conflicts. *J. Am. Soc. Inf. Sci. Technol.*, 64(8).

Lars Backstrom, Jon Kleinberg, Lillian Lee, and Cristian Danescu-Niculescu-Mizil. 2013. Characterizing and Curating Conversation Threads: Expansion, Focus, Volume, Re-entry. In *Proceedings of WSDM*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of ICLR*.

Anais Cadilhac, Nicholas Asher, Farah Benamara, and Alex Lascarides. 2013. Grounding Strategic Conversation: Using Negotiation Dialogues to Predict Trades in a Win-Lose Game. In *Proceedings of EMNLP*.

Justin Cheng, Michael Bernstein, Cristian Danescu-Niculescu-Mizil, and Jure Leskovec. 2017. Anyone Can Become a Troll: Causes of Trolling Behavior in Online Discussions. In *Proceedings of CSCW*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of EMNLP*.

Benjamin Collier and Julia Bear. 2012. Conflict, Criticism, or Confidence: An Empirical Examination of the Gender Gap in Wikipedia Contributions. In *Proceedings of CSCW*.

Jared R. Curhan and Alex Pentland. 2007. Thin Slices of Negotiation: Predicting Outcomes From Conversational Dynamics Within the First 5 Minutes. *Journal of Applied Psychology*, 92.

Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised Sequence Learning. In *Proceedings of NeurIPS*.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. In *Proceedings of ICWSM*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL*.

Maeve Duggan. 2014. Online Harassment. http://www.pewinternet.org/2014/10/22/online-harassment/.

Michael Feldman, Sorelle A. Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. 2015. Certifying and Removing Disparate Impact. In *Proceedings of KDD*.

Michel Galley, Kathleen McKeown, Julia Hirschberg, and Elizabeth Shriberg. 2004. Identifying Agreement and Disagreement in Conversational Speech: Use of Bayesian Networks to Model Pragmatic Dependencies. In *Proceedings of ACL*.

Jianfeng Gao, Michel Galley, and Lihong Li. 2018. Neural Approaches to Conversational AI. In *Proceedings of SIGIR*.

Kiran Garimella, Gianmarco De Francisci Morales, Aristides Gionis, and Michael Mathioudakis. 2017. Quantifying Controversy in Social Media. *ACM Transactions on Social Computing*, 1(1).

Codruta Girlea, Roxana Girju, and Eyal Amir. 2016. Psycholinguistic Features for Deceptive Role Detection in Werewolf. In *Proceedings of NAACL*.

Jack Hessel and Lillian Lee. 2019. Something's Brewing! Early Prediction of Controversy-causing Posts from Discussion Features. In *Proceedings of NAACL*.

Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. In *Proceedings of ACL*.

Yiqing Hua, Cristian Danescu-Niculescu-Mizil, Dario Taraborelli, Nithum Thain, Jeffery Sorensen, and Lucas Dixon. 2018. WikiConv: A Corpus of the Complete Conversational History of a Large Online Collaborative Community. In *Proceedings of EMNLP*.

Yohan Jo, Shivani Poddar, Byungsoo Jeon, Qinlan Shen, Carolyn P. Rosé, and Graham Neubig. 2018. Attentive Interaction Model: Modeling Changes in View in Argumentation. In *Proceedings of NAACL*.

David Jurgens, Libby Hemphill, and Eshwar Chandrasekharan. 2019. A Just and Comprehensive Strategy for Using NLP to Address Online Abuse. In *Proceedings of ACL*.

Joseph M. Kayany. 1998. Contexts of uninhibited online behavior: Flaming in social newsgroups on usenet. *Journal of the American Society for Information Science*, 49(12).

Srijan Kumar, William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2018. Community Interaction and Conflict on the Web. In *Proceedings of WWW*.

Sarah Ita Levitan, Angel Maredia, and Julia Hirschberg. 2018. Linguistic Cues to Deception and Perceived Deception in Interview Dialogues. In *Proceedings of NAACL*.

Ping Liu, Joshua Guberman, Libby Hemphill, and Aron Culotta. 2018. Forecasting the Presence and Intensity of Hostility on Instagram Using Linguistic and Social Features. In *Proceedings of ICWSM*.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of EMNLP*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in Translation: Contextualized Word Vectors. In *Proceedings of NeurIPS*.

Vlad Niculae and Cristian Danescu-Niculescu-Mizil. 2016. Conversational Markers of Constructive Discussions. In *Proceedings of NAACL*.

Vlad Niculae, Srijan Kumar, Jordan Boyd-Graber, and Cristian Danescu-Niculescu-Mizil. 2015. Linguistic Harbingers of Betrayal: A Case Study on an Online Strategy Game. In *Proceedings of ACL*.

Alexandra Olteanu, Carlos Castillo, Jeremy Boy, and Kush Varshney. 2018. The Effect of Extremist Violence on Hateful Speech Online. In *Proceedings of ICWSM*.

Ji Ho Park, Jamin Shin, and Pascale Fung. 2018. Reducing Gender Bias in Abusive Language Detection. In *Proceedings of EMNLP*.

John Pavlopoulos, Prodromos Malakasiotis, and Ion Androutsopoulos. 2017. Deeper Attention to Abusive User Content Moderation. In *Proceedings of EMNLP*.

Verónica Pérez-Rosas, Mohamed Abouelenien, Rada Mihalcea, Yao Xiao, C. J. Linton, and Mihai Burzo. 2016. Verbal and Nonverbal Clues for Real-life Deception Detection. In *Proceedings of EMNLP*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of NAACL*.

Peter Potash and Anna Rumshisky. 2017. Towards Debate Automation: A Recurrent Model for Predicting Debate Winners. In *Proceedings of EMNLP*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving Language Understanding by Generative Pre-training. Technical report, OpenAI.

Juliana Raskauskas and Ann D. Stoltz. 2007. Involvement in Traditional and Electronic Bullying Among Adolescents. *Developmental Psychology*, 43(3).

Manoel Horta Ribeiro, Pedro H. Calais, Yuri A. Santos, Virgílio A. F. Almeida, and Wagner Meira Jr. 2018. Characterizing and Detecting Hateful Users on Twitter. In *Proceedings of ICWSM*.

Sara Rosenthal and Kathleen McKeown. 2015. I Couldn't Agree More: The Role of Conversational Structure in Agreement and Disagreement Detection in Online Discussions. In *Proceedings of SIGDIAL*.

Maarten Sap, Dallas Card, Saadia Gabriel, Yejin Choi, and Noah A. Smith. 2019. The Risk of Racial Bias in Hate Speech Detection. In *Proceedings of ACL*.

Joseph Seering, Robert Kraut, and Laura Dabbish. 2017. Shaping Pro and Anti-Social Behavior on Twitch Through Moderation and Example-Setting. In *Proceedings of CSCW*.

Iulian V. Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models. In *Proceedings of AAAI*.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. A Hierarchical Latent Variable Encoder-Decoder Model for Generating Dialogues. In *Proceedings of AAAI*.

Vivek K. Singh, Marie L. Radford, Qianjia Huang, and Susan Furrer. 2017. "They basically like destroyed the school one day": On Newer App Features and Cyberbullying in Schools. In *Proceedings of CSCW*.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015a. A Hierarchical Recurrent Encoder-Decoder for Generative Context-Aware Query Suggestion. In *Proceedings of CIKM*.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015b. A Neural Network Approach to Context-Sensitive Generation of Conversational Responses. In *Proceedings of NAACL*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Proceedings of NeurIPS*.

Chenhao Tan, Vlad Niculae, Cristian Danescu-Niculescu, and Lillian Lee. 2016. Winning Arguments: Interaction Dynamics and Persuasion Strategies in Good-faith Online Discussions. In *Proceedings of WWW*.

Jessica Vitak, Kalyani Chadha, Linda Steiner, and Zahra Ashktorab. 2017. Identifying Women's Experiences With and Strategies for Mitigating Negative Effects of Online Harassment. In *Proceedings of CSCW*.

Svitlana Volkova and Eric Bell. 2017. Identifying Effective Signals to Predict Deleted and Suspended Accounts on Twitter across Languages. In *Proceedings of ICWSM*.

Henning Wachsmuth, Shahbaz Syed, and Benno Stein. 2018. Retrieval of the Best Counterargument without Prior Topic Knowledge. In *Proceedings of ACL*.

Lu Wang, Nick Beauchamp, Sarah Shugars, and Kechen Qin. 2017. Winning on the Merits: The Joint Effects of Content and Style on Debate Outcomes. *Transactions of the Association for Computational Linguistics*, 5.

Lu Wang and Claire Cardie. 2014. A Piece of My Mind: A Sentiment Analysis Approach for Online Dispute Detection. In *Proceedings of ACL*.

William Warner and Julia Hirschberg. 2012. Detecting Hate Speech on the World Wide Web. In *Proceedings of the Second Workshop on Language in Social Media*.

Michael Wiegand, Josef Ruppenhofer, and Thomas Kleinbauer. 2019. Detection of Abusive Language: The Problem of Biased Datasets. In *Proceedings of NAACL*.

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex Machina: Personal Attacks Seen at Scale. In *Proceedings of WWW*.

Diyi Yang, Jiaao Chen, Zichao Yang, Dan Jurafsky, and Eduard Hovy. 2019. Let's Make Your Request More Persuasive: Modeling Persuasive Strategies via Semi-Supervised Neural Nets on Crowdfunding Platforms. In *Proceedings of NAACL*.

Dawei Yin, Zhenzhen Xue, and Liangjie Hong. 2009. Detection of Harassment on Web 2.0. In *Proceedings of CAW2.0*.

Justine Zhang, Jonathan P. Chang, Cristian Danescu-Niculescu-Mizil, Lucas Dixon, Nithum Thain, Yiqing Hua, and Dario Taraborelli. 2018a. Conversations Gone Awry: Detecting Early Signs of Conversational Failure. In *Proceedings of ACL*.

Justine Zhang, Cristian Danescu-Niculescu-Mizil, Christina Sauper, and Sean J. Taylor. 2018b. Characterizing Online Public Discussions Through Patterns of Participant Interactions. In *Proceedings of CSCW*.

Justine Zhang, Ravi Kumar, Sujith Ravi, and Cristian Danescu-Niculescu-Mizil. 2016. Conversational Flow in Oxford-style Debates. In *Proceedings of NAACL*.

# A Benchmark Dataset for Learning to Intervene in Online Hate Speech

**Jing Qian**[†], **Anna Bethke**[*], **Yinyin Liu**[*], **Elizabeth Belding**[†], **William Yang Wang**[†]
[†] University of California, Santa Barbara
[*] Intel AI
{jing_qian,ebelding,william}@cs.ucsb.edu
{anna.bethke,yinyin.liu}@intel.com

## Abstract

Countering online hate speech is a critical yet challenging task, but one which can be aided by the use of Natural Language Processing (NLP) techniques. Previous research has primarily focused on the development of NLP methods to automatically and effectively detect online hate speech while disregarding further action needed to calm and discourage individuals from using hate speech in the future. In addition, most existing hate speech datasets treat each post as an isolated instance, ignoring the conversational context. In this paper, we propose a novel task of generative hate speech intervention, where the goal is to automatically generate responses to intervene during online conversations that contain hate speech. As a part of this work, we introduce two fully-labeled large-scale hate speech intervention datasets[1] collected from Gab[2] and Reddit[3]. These datasets provide conversation segments, hate speech labels, as well as intervention responses written by Mechanical Turk[4] Workers. In this paper, we also analyze the datasets to understand the common intervention strategies and explore the performance of common automatic response generation methods on these new datasets to provide a benchmark for future research.

## 1 Introduction

The growing popularity of online interactions through social media has been shown to have both positive and negative impacts. While social media improves information sharing, it also facilitates the propagation of online harassment, including hate speech. These negative experiences can have a measurable negative impact on users. Recently, the Pew Research Center (Center, 2017) reported that "roughly four-in-ten Americans have personally experienced online harassment, and 63% consider it a major problem."

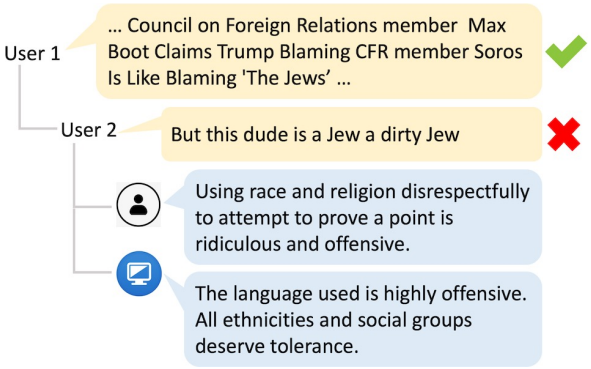


Figure 1: An illustration of hate speech conversation between User 1 and User 2 and the interventions collected for our datasets. The check and the cross icons on the right indicate a normal post and a hateful post. The utterance following the human icon is a human-written intervention, while the utterance following the computer icon is machine-generated.

To address the growing problem of online hate, an extensive body of work has focused on developing automatic hate speech detection models and datasets (Warner and Hirschberg, 2012; Waseem and Hovy, 2016; Davidson et al., 2017; Schmidt and Wiegand, 2017; ElSherief et al., 2018a,b; Qian et al., 2018a,b). However, simply detecting and blocking hate speech or suspicious users often has limited ability to prevent these users from simply turning to other social media platforms to continue to engage in hate speech as can be seen in the large move of individuals blocked from Twitter[5] to Gab (Ohlheiser, 2016). What's more, such a

[1] https://github.com/jing-qian/A-Benchmark-Dataset-for-Learning-to-Intervene-in-Online-Hate-Speech
[2] https://gab.ai
[3] https://www.reddit.com
[4] https://www.mturk.com
[5] https://www.twitter.com



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4755–4764, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

strategy is often at odds with the concept of free speech. As reported by the Pew Research Center (Center, 2017), "Despite this broad concern over online harassment, 45% of Americans say it is more important to let people speak their minds freely online; a slightly larger share (53%) feels that it is more important for people to feel welcome and safe online." The special rapporteurs representing the Office of the United Nations High Commissioner for Human Rights (OHCHR) have recommended that "The strategic response to hate speech is more speech." (Bielefeldt et al., 2011) They encourage to change what people think instead of merely changing what they do, so they advocate more speech that educates about cultural differences, diversity, and minorities as a better strategy to counter hate speech.

Therefore, in order to encourage strategies of countering online hate speech, we propose a novel task of generative hate speech intervention and introduce two new datasets for this task. Figure 1 illustrates the task. Our datasets consist of 5K conversations retrieved from Reddit and 12k conversations retrieved from Gab. Distinct from existing hate speech datasets, our datasets retain their conversational context and introduce human-written intervention responses. The conversational context and intervention responses are critical in order to build generative models to automatically mitigate the spread of these types of conversations.

To summarize, our contributions are three-fold:

- We introduce the generative hate speech intervention task and provide two fully-labeled hate speech datasets with human-written intervention responses.
- Our data is collected in the form of conversations, providing better context.
- The two data sources, Gab and Reddit, are not well studied for hate speech. Our datasets fill this gap.

Due to our data collecting strategy, all the posts in our datasets are manually labeled as hate or non-hate speech by Mechanical Turk workers, so they can also be used for the hate speech detection task. The performance of commonly-used classifiers on our datasets is shown in Section 6.

## 2 Related Work

In recent years, a few datasets for hate speech detection have been built and released by researchers. Most are collected from Twitter and are labeled using a combination of expert and non-expert hand labeling, or through machine learning assistance using a list of common negative words. It is widely accepted that labels can vary in their accuracy overall, though this can be mitigated by relying on a consensus rule to rectify disagreements in labels. A synopsis of these datasets can be found in Table 1.

Waseem and Hovy (2016) collect 17k tweets based on hate-related slurs and users. The tweets are manually annotated with three categories: sexist (20.0%), racist (11.7%), and normal (68.3%). Because the authors identified a number of prolific users during the initial manual search, the resulting dataset has a small number of users (1,236 users) involved, causing a potential selection bias. This problem is most prevalent on the 1,972 racist tweets, which are sent by only 9 Twitter users. To avoid this problem, we did not identify suspicious user accounts or utilize user information when collecting our data.

Davidson et al. (2017) use a similar strategy, which combines the utilization of hate keywords and suspicious user accounts to build a dataset from Twitter. But different from Waseem and Hovy (2016), this dataset consists of 25k tweets randomly sampled from the 85.4 million posts of a large number of users (33,458 users). This dataset is proposed mainly to distinguish hateful and offensive language, which tend to be conflated by many studies.

Golbeck et al. (2017) focus on online harassment on Twitter and propose a fine-grained labeled dataset with 6 categories. Founta et al. (2018) introduce a large Twitter dataset with 100k tweets. Despite the large size of this dataset, the ratio of the hateful tweets are relatively low (5%). Thus the size of the hateful tweets is around 5k in this dataset, which is not significantly larger than that of the previous datasets.

The dataset introduced by Chatzakou et al. (2017) is different from the other datasets as it investigates the behavior of hate-related users on Twitter, instead of evaluating hate-related tweets. The large majority of the 1.5k users are labeled as spammers (31.8%) or normal (60.3%). Only a small fraction of the users are labeled as bullies (4.5%) or aggressors (3.4%).

While most datasets are from single sources, Kennedy III et al. (2017) introduce a dataset with

| | Source | #Posts | Conv. | Categories | Interv. |
|---|---|---|---|---|---|
| Waseem and Hovy (2016) | Twitter | 17k | No | racist, sexist, normal | No |
| Davidson et al. (2017) | Twitter | 25k | No | hateful, offensive, neither | No |
| Golbeck et al. (2017) | Twitter | 35k | No | the worst, threats, hate speech, direct harassment, potentially offensive, non-harassment | No |
| Chatzakou et al. (2017) | Twitter | 9k | No | aggressive, bullying, spam, normal | No |
| Kennedy III et al. (2017) | Twitter, Reddit, The Gaurdian | 20k | No | harassment, non-harassment | No |
| Founta et al. (2018) | Twitter | 100k | No | abusive, hateful, normal, spam | No |
| Warner and Hirschberg (2012) | Yahoo! | 9k | No | anti-semitic, anti-black, anti-asian, anti-woman, anti-muslim, anti-immigrant, other-hate | No |
| Nobata et al. (2016) | Yahoo! | 2k | No | clean, hate, derogatory, profanity | No |
| Van Hee et al. (2015) | Ask.fm | 85k | No | threat/blackmail, insult, defamation, sexual talk, curse/exclusion, defense, encouragement to the harasser | No |
| Ours | Reddit | 22k | Yes | hate, non-hate | Yes |
| Ours | Gab | 34k | Yes | hate, non-hate | Yes |

Table 1: Comparison of our datasets with previous hate speech datasets. Conv.: Conversation. Interv.: Intervention.

a combination of Twitter (58.9%), Reddit, and The Guardian. In total 20,432 unique comments were obtained with 4,136 labeled as harassment (20.2%) and 16,296 as non-harassment (79.8%).

Since most of the publicly available hate speech datasets are collected from Twitter, previous research of hate speech mainly focus on Twitter posts or users (Waseem and Hovy, 2016; Gao et al., 2017; Burnap and Williams, 2016; Badjatiya et al., 2017; Davidson et al., 2017). While there are several studies on the other sources, such as Instagram (Zhong et al., 2016), Yahoo! (Warner and Hirschberg, 2012; Nobata et al., 2016), and Ask.fm (Van Hee et al., 2015), the hate speech on Reddit and Gab is not widely studied. What's more, all the previous hate speech datasets are built for the classification or detection of hate speech from a single post or user on social media, ignoring the context of the post and intervention methods needed to effectively calm down the users and diffuse negative online conversations.

## 3 Dataset Collection

### 3.1 Ethics

Our study got approval from our Internal Review Board. Workers were warned about the offensive content before they read the data and they were informed by our instructions to feel free to quit the task at any time if they are uncomfortable with the content. Additionally, all personally identifiable information such as user names is masked in the datasets.

### 3.2 Data Filtering

**Reddit:** To retrieve high-quality conversational data that would likely include hate speech, we referenced the list of the whiniest most low-key toxic subreddits[6]. Skipping the three subreddits that have been removed, we collect data from ten subreddits: *r/DankMemes, r/Imgoingtohellforthis, r/KotakuInAction, r/MensRights, r/MetaCanada, r/MGTOW, r/PussyPass, r/PussyPassDenied, r/The_Donald*, and *r/TumblrInAction*. For each of these subreddits, we retrieve the top 200 hottest submissions using Reddit's API. To further focus on conversations with hate speech in each submission, we use hate keywords (ElSherief et al., 2018b) to identify potentially hateful comments and then reconstructed the conversational context of each comment. This context consists of all comments preceding and following a potentially hateful comment. Thus for each potentially hateful comment, we rebuild the conversation where the comment appears. Figure 2 shows an example of the collected conversation, where the second comment contains a hate keyword and is considered as potentially hateful. Because a conversation may contain more than one comments with hate keywords, we removed any duplicated conversations.

**Gab:** We collect data from all the Gab posts in October 2018. Similar to Reddit, we use hate keywords (ElSherief et al., 2018b) to identify potentially hateful posts, rebuild the conversation context and clean duplicate conversations.

[6]https://www.vice.com/en_us/article/8xxymb/here-are-reddits-whiniest-most-low-key-toxic-subreddits

| Conversation | Hate Speech | Human-Written Intervention Responses |
|---|---|---|
| 1. User 1: United Kingdom: 'Schoolboy, 15, given detention for backing UKIP during classroom debate'<br>2. User 2: The education system is full of re***ds! Yes, most school teachers are ret***ed lefties! Teach your children to laugh at these ret***ed lefties!<br>3. User 3: Asking a teacher to not be a leftist is like asking a medieval munk to question the Pope.<br>4. User 4: The Jews are like Sjws, they infest everything. | 2, 4 | ➢ Use of this language is not tolerated and it is uncalled for.<br>➢ Use of the slurs and insults here is unacceptable in our discourse as it demeans and insults and alienates others.<br>➢ I recommend that you research the holocaust, you might change your opinion. |

Figure 2: An example of the aggregated data. The first column is the conversation text. Indexes are added to each post. Indentations before each post indicate the structure of replies. The second column is the indexes of the human-labeled hateful post. Each bullet point in the third column is a human-written response.

### 3.3 Crowd-Sourcing

After we collected the conversations from both Reddit and Gab, we presented this data to Mechanical Turk workers to label and create intervention suggestions. In order not to over-burden the workers, we filtered out conversations consisting of more than 20 comments. Each assignment consists of 5 conversations. For Reddit, we also present the title and content of the corresponding submission in order to give workers more information about the topic and context. For each conversation, a worker is asked to answer two questions:

- Q1: Which posts or comments in this conversation are hate speech?
- Q2: If there exists hate speech in the conversation, how would you respond to intervene? Write down a response that can probably hold it back (word limit: 140 characters).

If the worker thinks no hate speech exists in the conversation, then the answers to both questions are "n/a". To provide context, the definition of hate speech from Facebook[7]: "We define hate speech as a direct attack on people based on what we call protected characteristics race, ethnicity, national origin, religious affiliation, sexual orientation, caste, sex, gender, gender identity, and serious disease or disability." is presented to the workers. Also, to prevent workers from using hate speech in the response or writing responses that are too general, such as "Please do not say that", we provide additional instructions and rejected examples.

### 3.4 Data Quality

Each conversation is assigned to three different workers. To ensure data quality, we restrict the workers to be in an English speaking country including Australia, Canada, Ireland, New Zealand, the United Kingdom, and the United States, with a HIT approval rate higher than 95%. Excluding the rejected answers, the collected data involves 926 different workers. The final hate speech labels (answers to Q1) are aggregated according to the majority of the workers' answers. A comment is considered hate speech only when at least two out of the three workers label it as hate speech. The responses (answers to Q2) are aggregated according to the aggregated result of Q1. If the worker's label to Q1 agrees with the aggregated result, then their answer to Q2 is included as a candidate response to the corresponding conversation but is otherwise disregarded. See Figure 2 for an example of the aggregated data.

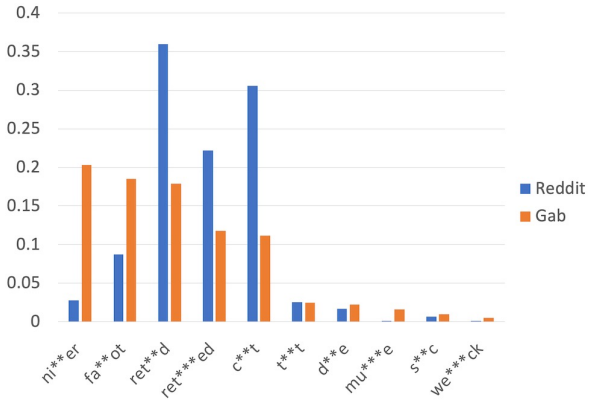


Figure 3: The distributions of the top 10 keywords in the hate speech collected from Reddit and Gab. Hate keywords are masked.

## 4 Dataset Analysis

### 4.1 Statistics

From Reddit, we collected 5,020 conversations, including 22,324 comments. On average, each conversation consists of 4.45 comments and the

[7] https://m.facebook.com/communitystandards/hate_speech/

length of each comment is 58.0 tokens. 5,257 of the comments are labeled as hate speech and 17,067 are labeled as non-hate speech. A majority of the conversations, 3,847 (76.6%), contain hate speech. Each conversation with hate speech has 2.66 responses on average, for a total of 10,243 intervention responses. The average length of the intervention responses is 17.96 tokens.

From Gab, we collected 11,825 conversations, consisting of 33,776 posts. On average, each conversation consists of 2.86 posts and the average length of each post is 35.6 tokens. 14,614 of the posts are labeled as hate speech and 19,162 are labeled as non-hate speech. Nearly all the conversations, 11,169 (94.5%), contain hate speech. 31,487 intervention responses were originally collected for conversations with hate speech, or 2.82 responses per conversation on average. The average length of the intervention responses is 17.27 tokens.

Compared with the Gab dataset, there are fewer conversations and comments in the Reddit dataset, comments and conversations are longer, and the distribution of hate and non-hate speech labels is more imbalanced. Figure 3 illustrates the distributions of the top 10 keywords in the hate speech collected from Reddit and Gab separately. The Gab dataset and the Reddit dataset have similar popular hate keywords, but the distributions are very different. All the statistics shown above indicate that the characteristics of the data collected from these two sources are very different, thus the challenges of doing detection or generative intervention tasks on the dataset from these sources will also be different.

### 4.2 Intervention Strategies

Removing duplicates, there are 21,747 unique intervention responses in the aggregated Gab dataset and 7,641 in the aggregated Reddit dataset. Despite the large diversity of the collected responses for intervention, we find workers tend to have certain strategies for intervention.

**Identify Hate Keywords**: One of the most common strategies is to identify the inappropriate terms in the post and then urge the user to stop using that work. For example, *"The C word and language attacking gender is unacceptable. Please refrain from future use."* This strategy is often used when the hatred in the post is mainly conveyed by specific hate keywords.

**Categorize Hate Speech:** This is another common strategy used by the workers. The workers classify hate speech into different categories, such as racist, sexist, homophobic, etc. This strategy is often combined with identifying hate keywords or targets of hatred. For example, *"The term ""fa**ot"" comprises homophobic hate, and as such is not permitted here."*

**Positive Tone Followed by Transitions:** This is a strategy where the response consists of two parts combined with a transitional word, such as "but" and "even though". The first part starts with affirmative terms, such as "I understand", "You have the right to", and "You are free to express", showing kindness and understanding, while the second part is to alert the users that their post is inappropriate. For example, *"I understand your frustration, but the term you have used is offensive towards the disabled community. Please be more aware of your words."*. Intuitively, compared with the response that directly warns, this strategy is likely more acceptable for the users and be more likely to clam down a quarrel full of hate speech.

**Suggest Proper Actions:** Besides warning and discouraging the users from continuing hate speech, workers also suggest the actions that the user should take. This strategy can either be combined with other strategies mentioned above or be used alone. In the latter case, a negative tone can be greatly alleviated. For example, *"I think that you should do more research on how resources are allocated in this country."*

## 5 Generative Intervention

Our datasets can be used for various hate speech tasks. In this paper, we focus on generative hate speech intervention.

The goal of this task is to generate a response to hate speech that can mitigate its use during a conversation. The objective can be formulated as the following equation:

$$Obj = \max \sum_{(c,r)\in D} \log p(r|c) \qquad (1)$$

where $c$ is the conversation, $r$ is the corresponding intervention response, and $D$ is the dataset. This task is closely related to the response generation and dialog generation, though several differences exist including dialog length, language cadence, and word imbalances. As a baseline, we chose the most common methods of these two tasks,

such as Seq2Seq and VAE, to determine the initial feasibility of automatically generate intervention responses. More recent Reinforcement Learning method for dialog generation (Li et al., 2016) can also be applied to this task with slight modification. Future work will explore more complex, and unique models.

Similar to (Li et al., 2016), a generative model is considered as an agent. However, different from dialog generation, generative intervention does not have multiple turns of utterance, so the action of the agent is to select a token in the response. The state of the agent is given by the input posts and the previously generated tokens. Another result due to this difference is that the rewards with regard to ease of answering or information flow do not apply to this case, but the reward for semantic coherence does. Therefore, the reward of the agent is:

$$rw(c, r) = \lambda_1 \log p(r|c) + \lambda_2 \log p_{back}(c|r) \quad (2)$$

where $rw(c, r)$ is the reward with regard to the conversation $c$ and its reference response $r$ in the dataset. $p(r|c)$ denotes the probability of generating response $r$ given the conversation $c$, and $p_{back}(c|r)$ denotes the backward probability of generating the conversation based on the response, which is parameterized by another generation network. The reward is a weighted combination of these two parts, which are observed after the agent finishing generating the response. We refer the readers to Li et al. (2016) for details.

## 6 Experiments

We evaluate the commonly-used detection and generation methods with our dataset. Due to the different characteristics of the data collected from the two sources (Section 4), we treat them as two independent datasets.

### 6.1 Experimental Settings

For **binary hate speech detection**, we experimented the following four different methods.

**Logistic Regression (LR):** We evaluate the Logistic Regression model with L2 regularization. The penalty parameter C is set to 1. The input features are the Term Frequency Inverse Document Frequency (TF-IDF) values of up to 2-grams.

**Support Vector Machine (SVM):** We evaluate the SVM model with linear kernels. We use L2 regularization and the coefficient is 1. The features are the same as in LR.

**Convolutional Neural Network (CNN):** We use the CNN model for sentence classification proposed by Kim (2014) with default hyperparameters. The word embeddings are randomly initialized (CNN in Table 2) or initialized with pretrained Word2Vec (Mikolov et al., 2013) embeddings on Google News (CNN$^*$ in Table 2).

**Recurrent Neural Network (RNN):** The model we evaluated consists of 2-layer bidirectional Gated Recurrent Unit (GRU) (Cho et al., 2014) followed by a linear layer. Same as for CNN, we report the performance of RNN with two different settings of the word embeddings.

The methods are evaluated on testing data randomly selected from the dataset with the ratio of 20%. The input data is not manipulated to manually balance the classes for any of the above methods. Therefore, the training and testing data retain the same distribution as the collected results (Section 4). The methods are evaluated using F-1 score, Precision-Recall (PR) AUC, and Receiver-Operating-Characteristic (ROC) AUC.

For **generative hate speech intervention**, we evaluated the following three methods.

**Seq2Seq** (Sutskever et al., 2014; Cho et al., 2014): The encoder consists of 2 bidirectional GRU layers. The decoder consists of 2 GRU layers followed by a 3-layer MLP (Multi-Layer Perceptron).

**Variational Auto-Encoder (VAE)** (Kingma and Welling, 2013): The structure of the VAE model is similar to that of the Seq2Seq model, except that it has two independent linear layers followed by the encoder to calculate the mean and variance of the distribution of the latent variable separately. We assume the latent variable follows a multivariate Gaussian Distribution. KL annealing (Bowman et al., 2016) is applied during training.

**Reinforcement Learning (RL):** We also implement the Reinforcement Learning method described in Section 5. The backbone of this model is the Seq2Seq model, which follows the same Seq2Seq network structure described above. This network is used to parameterize the probability of a response given the conversation. Besides this backbone Seq2Seq model, another Seq2Seq model is used to generate the backward probability. This network is trained in a similar way as the backbone Seq2Seq model, but with a response as input and the corresponding conversation as the target. In our implementation, the function of the

| Conversation | Seq2Seq | VAE | RL |
|---|---|---|---|
| User 1: Ok. Keep White Knighting for a sand ni**er who f**ks tons of White women. You're entitled to your opinion. | This is is not allowed and un time to treat people by their skin color | Use of the r-word is unacceptable in our discourse as it demeans and insults people with mental disabilities | I do not use derogatory racial slurs |
| User 1: Feminists are f**king ret***ed, why are they so pissed about manspreading shit makes no difference in space. I'm gonna start throwing things in womens cleavage as they walk by. | Using talking about someones character to describe a woman is insulting to hateful | I don't think using words that are sexist in nature contribute to a productive conversation | Your post uses speech that denigrates sex and gender, which is in violation of our Community Guidelines and TOS Agreement |
| User 1: Only a nig****ick could reach that c**t User 2: I bet a midget football team could hide under her overhang during a rainstorm and not get wet. User 3: Ni**er probably uses her as a waterbed | Use of the c-word is unacceptable in our discourse as it demeans and insults women | Please do not use derogatory language for intellectual disabilities | If you don't agree with you, there 's no need to resort to name calling |

Figure 4: Examples of the generated intervention responses. The hateful terms in the conversation are masked.

first part of the reward ($\log p(r|c)$) is conveyed by the MLE loss. A curriculum learning strategy is adopted for the reward of $\log p_{back}(c|r)$ as in Ranzato et al. (2016). Same as in Li et al. (2016) and Ranzato et al. (2016), a baseline strategy is employed to estimate the average reward. We parameterize it as a 3-layer MLP.

The Seq2Seq model and VAE model are evaluated under two different settings. In one setting, the input for the generative model is the complete conversation, while in the other setting, the input is the filtered conversation, which only includes the posts labeled as hate speech. The filtered conversation was necessary to test the Reinforcement Learning model, as it is too challenging for the backward model to reconstruct the complete conversation based only on the intervention response.

In our experiments on the generative hate speech intervention task, we do not consider conversations without hate speech. The testing dataset is then randomly selected from the resulting dataset with the ratio of 20%. Since each conversation can have multiple reference responses, we dis-aggregate the responses and construct a pair (conversation, reference response) for each of the corresponding references during training. Teacher forcing is used for each of the three methods. The automatic evaluation metrics include BLEU (Papineni et al., 2002), ROUGE-L (Lin, 2004), and METEOR (Banerjee and Lavie, 2005).

In order to validate and compare the quality of the generated results from each model, we also conducted human evaluations as previous research has shown that automatic evaluation metrics often do not correlate with human preference (Paulus et al., 2018). We randomly sampled 450 conversations from the testing dataset. We then generated responses using each of the above models trained with the filtered conversation setting. In each assignment, a Mechanical Turk worker is presented 10 conversations, along with corresponding responses generated by the three models. For each conversation, the worker is asked to evaluate the effectiveness of the generated intervention by selecting a response that can best mitigate hate speech. 9 of the 10 questions are filled with the sampled testing data and the generated results, while the other is artificially constructed to monitor response quality. After selecting the 10 best mitigation measures, the worker is asked to select which of the three methods has the best diversity of responses over all the 10 conversations. Ties are permitted for answers. Assignments failed on the quality check are rejected.

| Dataset | Gab | | | Reddit | | |
|---|---|---|---|---|---|---|
| Metric | F1 | PR | ROC | F1 | PR | ROC |
| LR | 88.2 | 94.5 | 95.4 | 64.7 | 80.4 | 91.4 |
| SVM | 88.6 | 94.7 | 95.6 | 75.7 | **81.1** | **92.0** |
| CNN | 87.5 | 92.8 | 92.6 | 74.8 | 76.8 | 87.5 |
| RNN | 87.6 | 93.9 | 94.2 | 71.7 | 76.1 | 88.6 |
| CNN* | **89.6** | **95.2** | **95.8** | 76.9 | 80.1 | 90.9 |
| RNN* | 89.3 | 94.8 | 95.5 | **77.5** | 79.4 | 90.6 |

Table 2: Experimental results for the detection task. PR is Precision-Recall AUC and ROC is ROC AUC. The models marked with * use pretrained Word2Vec embeddings. The best results are in bold.

### 6.2 Experimental Results and Discussion

The experimental results of the detection task and the generative intervention task are shown in Table 2 and Table 3 separately. The results of the human evaluation are shown in Table 4. Figure 4 shows examples of the generated responses.

As shown in Table 2 and 3, all the classification

| Dataset | Gab | | | | | | Reddit | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Inp. Set. | Complete | | | Filtered | | | Complete | | | Filtered | | |
| Metric | B | R | M | B | R | M | B | R | M | B | R | M |
| Seq2Seq | **13.2** | **33.8** | 23.0 | **15.0** | **34.2** | 23.6 | 5.5 | **29.5** | 19.5 | 5.9 | 28.2 | 20.0 |
| VAE | 12.2 | 32.5 | **23.4** | 12.4 | 32.8 | 21.8 | **6.8** | 29.0 | **20.2** | **7.0** | **29.1** | **20.1** |
| RL | - | - | - | 14.5 | 33.1 | **23.9** | - | - | - | 4.4 | **29.1** | 18.7 |

Table 3: Experimental results for generative intervention task. Inp. Set.: Input Setting (Section 6.1). B: BLEU. R: ROUGE-L. M: METEOR. Best results are in bold.

| Dataset | Gab | | Reddit | |
|---|---|---|---|---|
| Metric | Eff. | Div. | Eff. | Div. |
| Seq2Seq Wins | 22.4 | 28.0 | **31.1** | **34.0** |
| VAE Wins | 20.0 | 6.0 | 26.0 | 4.0 |
| RL Wins | **41.6** | **40.0** | 30.0 | 30.0 |
| Tie | 16.0 | 26.0 | 12.9 | 32.0 |

Table 4: Human evaluation results. Table values are the percentage of the answers. Eff.: Effectiveness, evaluates how well the generated responses can mitigate hate speech. Div: Diversity, evaluates how many different responses are generated. Best results are in bold.

and generative models perform better on the Gab dataset than on the Reddit dataset. We think this stems from the datasets' characteristics. First, the Gab dataset is larger and has a more balanced category distribution than the Reddit dataset. Therefore, it is inherently more challenging to train a classifier on the Reddit dataset. Further, the average lengths of the Reddit posts and conversations are much larger than those of Gab, potentially making the Reddit input nosier than the Gab input for both tasks. On both the Gab and Reddit datasets, the SVM classifier and the LR classifier achieved better performance than the CNN and RNN model with randomly initialized word embeddings. A possible reason is that without pretrained word embeddings, the neural network models tend to overfit on the dataset.

For the generative intervention task, three models perform similarly on all three automatic evaluation metrics. As expected, the Seq2Seq model achieves higher scores with filtered conversation as input. However, this is not the case for the VAE model. This indicates that the two models may have different capabilities to capture important information in conversations.

As shown in Table 3, applying Reinforcement Learning does not lead to higher scores on the three automatic metrics. However, human evaluation (Table 4) shows that the RL model creates responses that are potentially better at mitigating hate speech and are more diverse, which is consistent with Li et al. (2016). There is a larger performance difference with the Gab dataset, while the effectiveness and the diversity of the responses generated by the Seq2Seq model and the RL model are quite similar on the Reddit dataset. One possible reason is that the size of the training data from Reddit (around 8k) is only 30% the size of the training data from Gab. The inconsistency between the human evaluation results and the automatic ones indicates the automatic evaluation metrics listed in Table 3 can hardly reflect the quality of the generated responses. As mentioned in Section 4, annotators tend to have strategies for intervention. Therefore, generating the common parts of the most popular strategies for all the testing input can lead to high scores of these automatic evaluation metrics. For example, generating *"Please do not use derogatory language."* for all the testing Gab data can achieve 4.2 on BLEU, 20.4 on ROUGE, and 18.2 on METEOR. However, this response is not considered as high-quality because it is almost a universal response to all the hate speech, regardless of the context and topic.

Surprisingly, the responses generated by the VAE model have much worse diversity than the other two methods according to human evaluation. As indicated in Figure 4, the responses generated by VAE tend to repeat the responses related to some popular hate keyword. For example, *"Use of the r-word is unacceptable in our discourse as it demeans and insults people with mental disabilities."* and *"Please do not use derogatory language for intellectual disabilities."* are the generated responses for a large part of the Gab testing data. According to Figure 3, insults towards disabilities are the largest portion in the dataset, so we suspect that the performance of the VAE model is affected by the imbalanced keyword distribution.

The sampled results in Figure 4 show that the Seq2Seq and the RL model can generate reasonable responses for intervention. However, as is to be expected with machine-generated text, in the other human evaluation we conducted, where Mechanical Turk workers were also presented with

sampled human-written responses alongside the machine generated responses, the human-written responses were chosen as the most effective and diverse option a majority of the time (70% or more) for both datasets. This indicates that there is significant room for improvement while generating automated intervention responses.

In our experiments, we only utilized the text of the posts, but more information is available and can be utilized, such as the user information and the title of a Reddit submission.

## 7 Conclusion

Towards the end goal of mitigating the problem of online hate speech, we propose the task of generative hate speech intervention and introduce two fully-labeled datasets collected from Reddit and Gab, with crowd-sourced intervention responses. The performance of the three generative models: Seq2Seq, VAE, and RL, suggests ample opportunity for improvement. We intend to make our dataset freely available to facilitate further exploration of hate speech intervention and better models for generative intervention.

## Acknowledgments

This research was supported by the Intel AI Faculty Research Grant. The authors are solely responsible for the contents of the paper and the opinions expressed in this publication do not reflect those of the funding agencies.

## References

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 759–760. International World Wide Web Conferences Steering Committee.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Heiner Bielefeldt, Frank La Rue, and Githu Muigai. 2011. Ohchr expert workshops on the prohibition of incitement to national, racial or religious hatred. *Expert workshop on the Americas*.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, pages 10–21.

Pete Burnap and Matthew L Williams. 2016. Us and Them: Identifying Cyber Hate on Twitter across Multiple Protected Characteristics. *EPJ Data Science*, 5(1):11.

Pew Research Center. 2017. Online harassment 2017.

Despoina Chatzakou, Nicolas Kourtellis, Jeremy Blackburn, Emiliano De Cristofaro, Gianluca Stringhini, and Athena Vakali. 2017. Mean birds: Detecting aggression and bullying on twitter. In *Proceedings of the 2017 ACM on web science conference*, pages 13–22. ACM.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Eleventh International AAAI Conference on Web and Social Media*.

Mai ElSherief, Vivek Kulkarni, Dana Nguyen, William Yang Wang, and Elizabeth Belding. 2018a. Hate lingo: A target-based linguistic analysis of hate speech in social media. In *Twelfth International AAAI Conference on Web and Social Media*.

Mai ElSherief, Shirin Nilizadeh, Dana Nguyen, Giovanni Vigna, and Elizabeth Belding. 2018b. Peer to peer hate: Hate speech instigators and their targets. In *Twelfth International AAAI Conference on Web and Social Media*.

Antigoni Maria Founta, Constantinos Djouvas, Despoina Chatzakou, Ilias Leontiadis, Jeremy Blackburn, Gianluca Stringhini, Athena Vakali, Michael Sirivianos, and Nicolas Kourtellis. 2018. Large scale crowdsourcing and characterization of twitter abusive behavior. In *Twelfth International AAAI Conference on Web and Social Media*.

Lei Gao, Alexis Kuppersmith, and Ruihong Huang. 2017. Recognizing explicit and implicit hate speech using a weakly supervised two-path bootstrapping approach. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 774–782.

Jennifer Golbeck, Zahra Ashktorab, Rashad O Banjo, Alexandra Berlinger, Siddharth Bhagwan, Cody Buntain, Paul Cheakalos, Alicia A Geller, Quint Gergory, Rajesh Kumar Gnanasekaran, et al. 2017. A large labeled corpus for online harassment research. In *Proceedings of the 2017 ACM on Web Science Conference*, pages 229–233. ACM.

George W. Kennedy III, Andrew W. McCollough, Edward Dixon, Alexie Bastidas, John Ryan, Chris Loo, and Saurav Sahay. 2017. Technology solutions to combat online harassment. In *Proceedings of the first workshop on abusive language online*, pages 73–77.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1192–1202.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive language detection in online user content. In *Proceedings of the 25th international conference on world wide web*, pages 145–153. International World Wide Web Conferences Steering Committee.

Abby Ohlheiser. 2016. Banned from twitter? this site promises you can say whatever you want. *The Washington Post*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *6th International Conference on Learning Representations, ICLR 2018*.

Jing Qian, Mai ElSherief, Elizabeth Belding, and William Yang Wang. 2018a. Hierarchical cvae for fine-grained hate speech classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3550–3559.

Jing Qian, Mai ElSherief, Elizabeth Belding, and William Yang Wang. 2018b. Leveraging intra-user and inter-user representation learning for automated hate speech detection. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 118–123.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *4th International Conference on Learning Representations, ICLR 2016*.

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*.

Cynthia Van Hee, Els Lefever, Ben Verhoeven, Julie Mennes, Bart Desmet, Guy De Pauw, Walter Daelemans, and Véronique Hoste. 2015. Detection and Fine-grained Classification of Cyberbullying Events. In *RANLP'15: International Conference Recent Advances in Natural Language Processing*, pages 672–680.

William Warner and Julia Hirschberg. 2012. Detecting Hate Speech on the World Wide Web. In *ACL'12: Proceedings of the 2nd Workshop on Language in Social Media*, pages 19–26. Association for Computational Linguistics.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL student research workshop*, pages 88–93.

Haoti Zhong, Hao Li, Anna Cinzia Squicciarini, Sarah Michele Rajtmajer, Christopher Griffin, David J Miller, and Cornelia Caragea. 2016. Content-Driven Detection of Cyberbullying on the Instagram Social Network. In *IJCAI'16: Proceedings of the 25th International Joint Conference on Artificial Intelligence*, pages 3952–3958.

# Detecting and Reducing Bias in a High Stakes Domain

**Ruiqi Zhong**[1] *, **Yanda Chen**[1], **Desmond Patton**[2], **Charlotte Selous**[2], **Kathleen McKeown**[1]
[1]Department of Computer Science, Columbia University
[2]School of Social Work, Columbia University

{rz2383, yc3384, dp2787, cis2114}@columbia.edu
{kathy}@cs.columbia.edu

## Abstract

Gang-involved youth in cities such as Chicago sometimes post on social media to express their aggression towards rival gangs and previous research has demonstrated that a deep learning approach can predict aggression and loss in posts. To address the possibility of bias in this sensitive application, we developed an approach to systematically interpret the state of the art model. We found, surprisingly, that it frequently bases its predictions on stop words such as "a" or "on", an approach that could harm social media users who have no aggressive intentions. To tackle this bias, domain experts annotated the rationales, highlighting words that explain why a tweet is labeled as "aggression". These new annotations enable us to quantitatively measure how justified the model predictions are, and build models that drastically reduce bias. Our study shows that in high stake scenarios, accuracy alone cannot guarantee a good system and we need new evaluation methods.

## 1 Introduction

There has been increased interest in using natural language processing to help understand the root causes of violence in cities such as Chicago. Researchers in social work found that posts on social media by gang-involved youth mirror their life on the physical street (Patton et al., 2016, 2017) and suggested that if community outreach workers could use tools to help identify posts of aggression and grief, they could intervene before grief turns to retribution and help avert violence. Early research in this area (Blevins et al., 2016) used support vector machines for the three-way classification task of *aggression*, *loss* and *other* and recent work (Chang et al., 2018) demonstrated a significant increase in accuracy through the use of a neural net approach along with the representation of contextual information from previous conversations. In both of these approaches, computer scientists and social work researchers worked side by side to enable qualitative analysis of the data to use in machine learning and to provide meaningful error analyses.

Given the interest in using this work in a live setting to identify situations where community outreach workers could intervene, we are concerned with *how* the model makes the decision. This is especially important since this is a high stakes domain and machine learning systems are known to incorporate bias. For example, the COMPAS Recidivism Algorithm, which assesses the likelihood of a criminal defendant to re-offend and influences judges' sentencing decisions, is known to be biased against African Americans (Feller et al., 2016). To avoid similar errors in the gang-violence application, we examined whether the model is biased through systematic investigation of how the model makes the predictions. Since misclassifying a tweet as *aggression* might potentially criminalize a vulnerable population (young black and latino youth) that is already over criminalized, we focus on classification of the *aggression* class in our paper.

Our finding is surprising: in at least $10\%$ of the cases the Chang et al. (2018) model relies on stop words such as "a" and "on" to classify a tweet as aggression; these words carry no semantic or pragmatic indication of aggression or loss. We recognized that there might be more hidden biases, so we developed novel metrics to quantitatively compare the difference between how humans and models make their classification. To achieve this, we ask domain experts to annotate the rationale for the aggression tweets by highlighting words and phrases that indicate why experts classify them as

The first author is now a PhD student at University of California, Berkeley.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4765–4775, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

aggression. We compare the rationale against the models' interpretation.

We also develop a model to incorporate human rationales, and show that the system can make predictions based on similar reasons to those of domain experts. Our F-score is slightly improved over Chang et al. (2018) and we are able to reduce the bias drastically. We show how important this is by using an adversarial approach which demonstrates that we can change the prediction of posts predicted as *other* to *aggression* by the simple insertion of the non-offending function words. This implies that high F-score alone does not guarantee a good system, and that researchers in other high stakes domain should evaluate their algorithms from various angles.

Our contributions include:

- A validated finding that the previous state-of-the-art model is biased.
- A rich dataset created by domain experts containing annotated human rationales.
- New metrics to evaluate models' robustness under adversary, and the difference between model interpretations and human rationales.
- A model that incorporates human rationales in training and behaves more as humans do.

We also include an in-depth discussion on how misinterpretation might emerge when labels are being annotated by the experts. The battle against bias is far from solved, and we hope our insights and methodologies can guide future research in high stakes domains.

## 2 Revealing Model Biases

We used the code and data from Chang et al. (2018) and fully reproduced the state-of-the-art model that uses a Convolutional Neural Network, domain specific word embeddings and context features.

### 2.1 Locating Biases

We examine the model using the simple leave-one-out method similar to Ribeiro et al. (2016). Suppose each tweet $T$ is a sequence of $l$ tokens $[e_1, e_2...e_l]$ and the model gives the prediction confidence score $y$. For each token $e_i$, we calculate the model's prediction confidence $y_{-i}$ on the same tweet with $e_i$ masked as an unknown word; hence the influence of the $i^{th}$ token $I_i$ is approximated by $I_i = y - y_{-i}$. Then we manually examine the top two influential uni-grams for each tweet that the model classified as aggression.

As we expect, tokens strongly associated with aggression, such as "[redacted]", "dead" and many other curse words appear frequently as the most influential tokens. However, surprisingly, words such as "a" and "on" are frequently considered the most influential by the model as well. Out of 419 tweets that the model classifies as aggression, 15/32 times the model considers the token "a" to be the top first or second most influential token and "on" 24/51 times the top first or second, respectively. We also found that the model is biased against other stop words, such as "into" and "of". This indicates that more than **10%** of the time the model heavily relies on these unintentional differences of language use to classify tweets as aggression!

### 2.2 Confirming Biases Exist

Initially, we found this unexpected result surprising, so we checked the second order sensitivity (Feng et al., 2018) and constructed adversarial data sets.

**Second Order Consistency** Feng et al. (2018) defined this term when they found that the leave-one(token)-out (LOO) method might find a completely different set of most influential words if the most un-influential word is dropped from the sentence. However, we do not observe this problem in our case: out of 419 predictions, 379 times the most influential word remains the most influential and 33 times it becomes the second most influential one after we drop the most un-influential words.

**Adversarial Dataset** If we can identify a weakness of the model, we must be able to break it in a corresponding way. To manifest that there is indeed bias against "a" and "on," for each of these two words, we inserted it in each labeled tweet and counted the number of times the model changes its prediction from non-aggression to aggression.

To ensure that our insertions are natural, we trained a large language model (Peters et al., 2018) on the unlabeled corpus ($\sim$ 1 million tweets) used by Chang et al. (2018), and score the naturalness of the insertion by the change of perplexity given by the language model. For each tweet we calculate the most "natural" position to insert our biased word and only took the top 800 natural

tweets out of $\sim$ 7000 candidates as our adversarial dataset. We also subsample this dataset and check the subsample manually to make sure that these insertions make sense. In fact, since most tweets in the dataset are grammatically flexible in the first place, our insertion methods usually do not make them more unnatural - we have fewer constraints than adversarial attacks (Alzantot et al., 2018) conducted in domains such as IMDB reviews (Maas et al., 2011) or Textual Entailment (Bowman et al., 2015).

We find that even with these naive and seemingly harmless insertions, the model changes the prediction from non-aggression to aggression for about 3%/5% of the generated tweets when inserting "a"/"on". With this overwhelming evidence, we conclude that the model is biased.

The simplest way to fix this specific bias is a post-mortem one - just remove "a" and "on" in training and prediction. However, this only works if these are the only two points of bias and we suspect that there are many more biases lurking in the model. Hence, we need to comprehensively study the difference between how humans and models make their decision. We need a new dataset with annotated human rationales to achieve this goal.

## 3 Data

We use the same dataset and cross validation folds as in Chang et al. (2018), which contains 329 Aggression tweets, 734 Loss tweets, with the Other class comprising the remaining 3,873 tweets. The language in the dataset is hyper-local to the community we are studying and differs significantly from standard American English. Therefore, most of the toolkits (e.g. pre-trained language models, parsers, etc) developed for standard English do not apply. We extend their dataset by asking domain experts to annotate the rationales and additional information on context for each of the tweets labeled as aggression. Below is the set of annotation instructions.

- **Rationales in Tweets** We ask the domain experts to annotate the most important word/words (rationales) in the tweet that makes them think the tweet is aggression/loss; or say, the removal of which would make them not consider it as an aggressive tweet. We ask them to annotate the "minimum set of words" that is important for their final decision, thus excluding words that are not very meaningful, such as "on", "basically", "lol", etc. Examples are shown in Table 1.

- **Aggressive by Context** Sometimes the tweet itself does not contain aggressive content if we interpret them at face level, but placing it into context reveals that the user in fact conveyed aggressive intent. We ask the experts to label this attribute as True whenever they use an outside source to provide additional information needed to label a tweet. These sources can include news sources, past posts, interactions, online dictionaries, or reddit threads.

- **Aggressive by Mention/Retweet** Similarly, interactions between users that are suspected to be affiliated with the same or rival gangs might contain aggressive intent. For example, users may communicate with others in the same gang about the possibility of retaliation or they may directly taunt members of rival gangs. We ask the annotators to label this attribute as True if the context of interaction between users contributes to the aggression label. This corresponds to the pairwise interaction feature in Chang et al. (2018).

- **Aggressive by Picture/URL** This attribute indicates whether the picture or URL in the tweet contains aggressive content. Since our study is natural language processing based, we do not make use of this piece of information in this paper.

- **Controversial Labels** Occasionally our experts would interpret a tweet previously considered aggressive as non-aggressive. We do not use these labels in our experiments for fair comparison with Chang et al. (2018), but we include them in our new dataset.

Out of the 329 aggression tweets, 274 contain rationales in the tweet, while the other aggression labels are inferred by context rather than the content within the tweet itself. To keep notations uncluttered, we refer to all the "aggressive by *" labels as *context labels* in the rest of this paper. The meta-statistics on this can be seen in table 2. Additionally, 39 (12% of aggression) tweets are considered controversial by our experts. We also ask the annotator to write down thoughts and observa-

| Tweets |
|---|
| "@user_ do you think it's cool yo ass slow ? You sound dumb asf" |
| "Rob you fucking goofys that got some to say about rage just to show y'all bitches" |
| "GlizzyGang Bitch We Got Out Glocks Up😈" |

Table 1: Typical aggression tweets in our dataset, which includes annotations with human rationales (underlined).

tions while annotating, and we share these insights in our discussion section. [1]

| Context Label Types | Yes | No | N/A |
|---|---|---|---|
| Context | 107 | 74 | 148 |
| Mention/Retweet | 36 | 35 | 258 |
| URL | 9 | 3 | 317 |
| Picture | 14 | 15 | 300 |

Table 2: Whether aggressive intent is observed in Context, Mention/Retweet, URL and/or Picture. Many of the context label cannot be annotated due to Twitter access policy.

## 4 Methods

We compare the Convolutional Neural Network (CNN) model from Chang et al. (2018), an LSTM (Long Short Term Memory) Network model with attention mechanism (architecture defined below) and our new model that incorporates rationales by trained attention (Zhong et al., 2019).[2]

### 4.1 Overview of Chang et al. (2018)

Here we give a quick overview of their CNN model with context features. Let $l$ denote the length of the pre-processed tweet, $T = [e_1, e_2 ... e_l]$ be the input sequence, and $c$ the context feature. First each token $e_i$ is mapped to an embedded representation $w_i$ through their domain specific word embeddings. Then they used a convolution layer with kernel size 1 and 2, followed by a maxpooling layer and a linear fully connected layer to produce the tweet representation $z$. Finally, they concatenated $z$ with the context feature $c$ and fed it to the final classification layer. We refer the readers to their original paper for further details.

### 4.2 LSTM with Attention

We use a standard LSTM model with attention mechanism. The model encodes the embedded sequence $[w_1, w_2, ..., w_l]$ with an LSTM to obtain hidden state representation $h_i$ with hidden dimension 64 for each position $i$ and then calculate an attention weight for each position based on $h_i$. The final representation $z$ is a weighted average of the hidden states concatentated with the context feature, which we pass to the final classification layer. Formally,

$$\alpha_i = tanh(\mathbf{v}h_i), A(i) = \frac{e^{\alpha_i}}{\sum_j e^{\alpha_j}} \quad (1)$$

$$z = [\sum_{i=1}^{l} A(i)h_i, c] \quad (2)$$

where $\mathbf{v}$ are trainable weights we used to calculate the attention. We used binary cross entropy loss between the final prediction and the ground truth label, which we denote as $\mathcal{L}_{clf}$

### 4.3 Trained Attention with Rationale

We guide the model attention to attend to human rationales. We introduce a "ground truth attention" $A^*$, which is uniformly distributed over all the positions where the token is an annotated rationale. For example, if $R = \{r_1, r_2 \dots r_d\}$ are indexes of the tokens that are rationale words, then the ground truth attention distribution $A^*$ is defined as:

$$A^*(i) = \frac{1}{d} \text{ if } i \in R, 0 \text{ otherwise} \quad (3)$$

Then we impose a KL divergence loss component between the ground truth attention and the model's attention. We optimize the weighted sum between the label classification and the attention loss, i.e.

$$\mathcal{L}_{attn} = KL(A^*||A), \mathcal{L} = \mathcal{L}_{clf} + \lambda_{attn}\mathcal{L}_{attn} \quad (4)$$

where we fix $\lambda_{attn} = 4$ after tuning.

To make a fair comparison with Chang et al. (2018), we use the same set of hyper-parameters and training procedures. We find that taking the majority vote of an ensemble of 5 neural networks trained on the same data can improve the F-score by 1∼2 points, so we report both the averaged and the majority vote result across 5 independent

[1] Due to privacy concerns we cannot release the additional annotation publicly, but we will make it available to researchers who sign an MOU specifying their intended use of the data and their agreement with our ethical guidelines. Send email to the dp2787@columbia.edu. Please contact the authors of Chang et al. (2018) for the original dataset.

[2] Our code is publicly available at https://github.com/David3384/GI_2019

runs. We also experiment with representations from a large scale language model trained on our unlabeled corpus (Peters et al., 2018), but it only brought marginal improvements. For simplicity we do not include these experiments in our paper.

## 5 Evaluation Metrics

Besides the macro F-score in (Chang et al., 2018), we use rationale rank to characterize the difference between how humans and models make the decisions. We also quantitatively evaluate how the model uses the context feature. For fair comparison, we used the same labels as in (Chang et al., 2018) even though some labels are re-labeled as "controversial".

### 5.1 Rationale Rank

We first use the leave-one-out-method to approximate the influence $I_i$ of each token $e_i$ as documented in section 2.1. If the model makes predictions based on similar rationale words as human, $I_i$ should be large when $e_i$ is a rationale word, and should be small when $e_i$ is not a rationale. Therefore, a rationale word should have lower ranks if all the tokens are sorted by its influence on model's prediction in descending order. We define the influence rank of the $i^{th}$ token $rank(i)$ as the number of tokens in the tweet that have higher influence score than the $i^{th}$ token.

We further relax our requirement for the model: we do not have the unrealistic hope that the model is relying on every rationale word; instead, we only consider the lowest ranking rationale. Formally, suppose there are multiple rationale words with indexes $R = \{r_1, r_2..r_d\}$, then we define the "rationale rank" as:

$$rationalerank = min_{i \in R} rank(i) \quad (5)$$

We calculate the average rationale rank for all tweets that are classified as aggression and have rationale annotations on the cross validation test set for each fold.

Since the majority of tweets usually has rationale rank 0, the averaged rationale rank is small and the difference between each model is "diluted" by the prevailing zeros. Therefore, for each model, we also report the percentage of tweets that has rationale 0 or 1, respectively. This more intuitively reflects on what fraction of the tweets the models' reasoning agrees with human rationales.

### 5.2 Context Feature Evaluation

Chang et al. (2018) claimed that their pair-wise interaction feature captures information about users' past interactions, and they supported it by showing significant improvement when incorporating it into the CNN model. Nevertheless, it does not automatically entail that the model is using this feature in a way that humans do. Here we examine this claim quantitatively with our annotated interpretation of context.

Chang et al. (2018) concatenated the context features before the final classification softmax layer, so we can directly calculate the impact of a context feature by taking its dot product with the corresponding weight learned by the model. Formally, let $c_1$ be the "emotional state" feature, $c_2$ the "user history", $c_3$ the "pair-wise user interaction" and the pooled tweet representation is $z$, then the final prediction score is given by:

$$y = \sigma(w_z z + \sum_{i=1}^{3} w_i c_i + b) \quad (6)$$

where, $w_{(\cdot)}$ are trainable weights and $b$ the bias term.

We group the tweets by the annotated context label "by mention/retweet", a single label, annotated as described in section 3. If the pair-wise interaction feature $c_3$ truly models domain experts' interpretation of user interaction as Chang et al. (2018) has hypothesized, $w_3 c_3$ should be higher for those tweets that are annotated as "aggressive by mention/retweet" than those that are not. We do not examine use of other context labels and features, since other context features do not model URL, pictures or a general type of context.

## 6 Prediction Flips under Adversary

### 6.1 Generating Adversarial Tweets

We consider the simplest operation: inserting certain candidate neutral unigrams $u \in U = \{$ "a", "on", "da", "into", "of", "that"$\}$ that the model strongly associates with aggression. [3] Assuming that we have already trained a language model that gives a perplexity score $p(T)$ for a tweet $T = [e_1, e_2, \ldots e_l]$, then for each $u \in U$ we create a

[3] We describe how we choose these unigrams in the appendix A.2. We also experimented with "in", "be", "do", "any", "u", "out", "basically", "yea", "ever", "UP", "👉", "we" etc, but the results are not significant so we present them in the appendix.

dataset by inserting it into all of the tweets $T$ in our labeled dataset $\mathcal{T}$. We describe how we train the language model in the appendix section A.1.

For each $T \in \mathcal{T}$, there are $|T| + 1$ positions to insert the unigram $u$. We pick the resulting tweet $T'$ with $u$ inserted that has the highest likelihood score $p(T')$ among all the $|T| + 1$ possible insertions, and calculate the corresponding "insertion naturalness score" $p(T') - p(T)$. After inserting the unigram $u$ into all the tweets in the labeled dataset $\mathcal{T}$, we obtain a corresponding dataset $\mathcal{T}'$ and pick the top 800 tweets that have the highest "insertion naturalness score".

Then we calculate how many times a model would flip its prediction from non-aggression to aggression among these 800 pairs of tweets, before and after inserting the unigram $u$. Notice that even one percent of "flip rate" is significant: the unigrams $u$ we consider here are not rare words and thus have high coverage. If the model is deployed at a large scale, millions of tweet will be classified, and tens of thousands of tweets will be influenced, not mentioning that we are considering only one unigram and there are many potential other tokens biased by the model.

### 6.2 Evaluating Generation Naturalness

Since words are discrete, generating adversarial text inputs has been a hard problem in other domains for at least two reasons: 1) one perturbation might be large enough to change the label. 2) we cannot guarantee that the resulting text is still a natural input. However, these challenges do not play much of a role in our domain. First, the candidate words we are inserting are "function words", which we know should not cause the model to flip their predictions. Second, the large proportion of tweets we are working with are grammatically flexible in the first place. Therefore, in contrast to genres that follow the conventions of Standard American English grammar, we suspect that adding function words will not make the tweets look unnatural to the domain experts and we test this hypothesis.

To evaluate the plausibility of our generated tweets, we sample 18 tweets from all the generated tweets and 18 existing tweets from our labeled dataset. Then we ask our experts to try their best to classify whether each tweet comes from model generation or the existing dataset. We also ask them to annotate whether each tweet is "natural", "reasonable but awkward" or "nonsensical".

The experts classified 25 of them correctly ($\sim 70\%$), only slightly better than the $50\%$ random guess baseline even though they are aware that we are generating tweets by inserting function words. Out of 18 generated tweets, they considered 7 as "natural", 8 "reasonable but awkward" and 3 as "non-sensical". In comparison, out of the 18 existing tweets, they considered 14 as "natural", 1 as "reasonable but awkward" and 3 as "non-sensical." These results imply that although our generated tweets might not all be fluent, they are still reasonable enough as adversarial tweets. Interestingly, they also considered 3 existing tweets as "non-sensical", partly because the tweets are highly grammatically flexible and we told them that there are generated tweets.

## 7 Results

The model F-score performances are shown in table 3 and rationale rank in table 4. We present our results for macro F-score, Rationale Rank, Prediction Flips on Adversarial Dataset and Context Feature Evaluation. We also evaluate the models from (Blevins et al., 2016) for completeness. [4]

**F-score** We first notice that taking the ensemble of 5 independent runs trained on the same data consistently improves the performance over averaged individual run performance, so we report all of our results on ensemble models. Compared to Chang et al. (2018), our LSTM+Rationale model differs by at most one point on all of the macro-F score metrics, and sometimes achieves the best performance.

**Rationale Rank** A closer look at the rationale rank result reveals a huge difference (p-value $< 10^{-5}$): our model significantly beats all other models on this metrics.

**Prediction Flips under Adversary** The results are shown in table 5. For each candidate unigram we count the number of times the model flips its prediction from non-aggression to aggression among the top 800 insertions. By incorporating rationales into the model, we achieve the fewest flips for half of the unigrams and under none of the six cases does it flip the most predictions.

These two results demonstrate that although the CNN model (Chang et al., 2018) only slightly un-

[4] For consistency, we used their model but first classify aggression vs. rest, rather than other vs. rest as in (Blevins et al., 2016). The training procedure is deterministic, so the ensemble gives the same performance as individual ones.

| Model | Average Macro F1 | Ensemble Agg. F1 | Ensemble Loss F1 | Ensemble Other F1 | Ensemble Macro F1 |
|---|---|---|---|---|---|
| Blevins et al. (2016) | 63.5 | 34.5 | 67.3 | 88.7 | 63.5 |
| Chang et al. (2018) | **68.2** | 41.3 | **75.4** | 91.3 | 69.4 |
| CNN + Twitter | 65.8 | 36.2 | 73.9 | 91.3 | 67.1 |
| LSTM | 67.0 | 38.4 | 74.2 | 91.3 | 68.0 |
| LSTM + Rationale | **68.2** | **43.0** | 75.0 | **91.6** | **69.8** |

Table 3: Results comparing different model architectures. Chang et al. (2018) refers to the full CNN model with domain specific embedding and context feature; "CNN Twitter" replaces the domain specific embedding with GloVe twitter embeddings (Pennington et al., 2014); LSTM refers to LSTM Attention described in section 4.2; LSTM Rationale refers to the same LSTM Attention with trained attention described in section 4.3. For each column, the best performing entry is boldered.

| Model | Avg RR | RR = 0 | RR = 1 |
|---|---|---|---|
| Blevins et al. (2016) | 1.70 | 0.60 | 0.13 |
| Chang et al. (2018) | 1.42 | 0.54 | 0.17 |
| CNN + Twitter | 1.82 | 0.43 | 0.25 |
| LSTM | 1.73 | 0.50 | 0.15 |
| LSTM + Rationale | **0.86** | **0.69** | 0.14 |

Table 4: Results on (averaged) rationale rank (RR), fraction of tweets when RR = 0 and RR = 1, respectively. For (averaged) rationale rank and RR = 0, the best performing entry is boldered. We do not bolder the column RR = 1 since it is sub-optimal.

derperforms our model in terms of F-score performance, it behaves very differently behind the scene. Our model is more robust and makes predictions that match human predictions. This implies that traditional F-score measure alone neither provides us deeper understanding of the model, nor reveals the underlying biases.

**Context Feature Evaluation** We calculate the correlation between "how the human uses context" and "how the model uses context" as described in section 5.2. The mean difference of $w_3 c_3$ between the two groups split by the label value is 1.61 with statistical significance $10^{-3}$, indicating that the pairwise interaction feature correlates well with how domain experts interpret users' retweet/mentions.

## 8 Related Work

Patton et al. (2013) initiated a new field of study, coining the term *internet banging* to describe the phenomenon whereby online altercations lead to offline violence. This work is further motivated by challenging gun violence statistics in the city of Chicago which saw a 58% spike between 2015 and 2016. In addition, the number of homicides stemming from physical altercations decreased during that period with little empirical evidence to explain this shift (Kapustin et al., 2016). Investigating internet banging represents an important first step in understanding why gun deaths increase despite fewer physical altercations.

Blevins et al. (2016) is a collaboaration between social work and computer science experts that brings together social work's substantive knowledge of communities, youth development, and gun violence with data science's innovative machine learning tools. They developed a new analytical system that leverages community participatory methods and machine learning to automatically detect the expression of loss and aggression on Twitter. Blevins et al. (2016) collected and annotated tweets posted by Gakirah Barnes, a deceased gang members prolific on twitter, as well as her top communicators. Chang et al. (2018) extended this dataset to be six times larger and achieved the state-of-the-art result by using a CNN.

Our interpretability method is most similar to the leave-one-out (LOO) approach by Ribeiro et al. (2016). Nevertheless, it risks making the sentence ungrammatical, hence producing "rubbish examples" (Goodfellow et al., 2014) and pathological interpretations (Feng et al., 2018). We explicitly address this concern by checking the second-order consistency proposed in Feng et al. (2018). To further ensure that the removal of a word results in a "blank state", erasure-based method proposed in Li et al. (2016) can be used. It would also be interesting to explore different interpretability methods, such as using the influence function (Koh and Liang, 2017) or nearest neighbor in the feature space (Strobelt et al., 2019). Our method of automatically generating adversarial data accord-

| Models \ Unigrams | a | on | da | into | of | that |
|---|---|---|---|---|---|---|
| Blevins et al. (2016) | 44.16[!] | 121.00[!] | 37.44[!] | 117.72[!] | 36.84[!] | 74.48 [!] |
| Chang et al. (2018) | 26.56 | 36.00 | 8.52* | 21.68 | 19.20 | 7.52* |
| CNN + Twitter | 38.40 | 35.80 | 17.60 | 15.88 | 5.24* | 7.64 |
| LSTM | 16.16 | 40.28 | 21.52 | 26.92 | 13.20 | 13.12 |
| LSTM + Rationale | 13.52* | 23.16* | 12.40 | 10.28* | 11.40 | 8.96 |

Table 5: The number of model's prediction flip from non-aggressive to aggressive, out of the 800 attacking tweets generated by inserting a specified "neutral" unigram. To obtain a stable estimate, the count of number of flips of each model is averaged across 5 model runs and models trained on 5 folds - thus leading to non-integer results. For each column, the worst performing entry is marked with "!" and best with "*".

ing to a pre-identified weakness of a model without accessing the model output and parameters is most similar to Jia and Liang (2017). From Alzantot et al. (2018), we borrowed the idea of using a language model to ensure naturalness of generated sentences, and it is a promising next step to collect natural human-generated adversarial examples (Wallace et al., 2018). In the future, studying white-box (Ebrahimi et al., 2017; Bělohlávek, 2017) and character level attacks (Gröndahl et al., 2018) are important directions to provide additional robustness in this critical domain.

Trained attention was explored in event detection (Liu et al., 2017), machine translation (Mi et al., 2016) and sentiment analysis (Zhong et al., 2019). More sophisticated approaches include deriving attention from human rationales (Yujia Bao, 2018). Finding better ways to make use of this kind of information is a promising avenue, especially in this low-resource domain.

Ensuring machine learning fairness has increasingly caught researchers' attention (Corbett-Davies and Goel, 2018) since algorithms can create concerning negative societal impacts (Feller et al., 2016). Biases might be based on gender (Bolukbasi et al., 2016; Zhao et al., 2017), race (Kiritchenko and Mohammad, 2018; Rudinger et al., 2017), writing styles (Madnani et al., 2017), or other factors. More broadly, bias can also occur when annotators label the data. For example, Golbeck et al. (2017) found it hard to distinguish between mere offensive speech and hate speech.

## 9 Discussion

### 9.1 Limitations of the Annotation Instruction

The instructions for annotating rationales in tweets do not fully recognize the complex nature of language. Current annotation reduces the experts' underlying reason to annotate a tweet as aggression to a set of key words, which is not flexible enough to convey more complicated inferences. The experts' analyses are far more complex: they use context to disambiguate the meaning of the rationale words rather than identifying key words. Since the language is hyper-local and many of the non-aggressive tweets might appear aggressive in outsiders' eyes, interpreting the context is a crucial component while determining an aggression label and it is important to develop more flexible context rationale representations in the future.

Take, for example, the post "I talk to alotta Da Guyz In jail Everyday On l'a Dese N***as Ain't Sending Bro Nem A Dime 😒 But Steady Yellin Free Em Goof ass". At face value, the words "😒", "goof", and "ass" per se do not capture the reason why this post is aggressive; the aggression label for this post in fact derives from the poster calling out the dishonesty of those who support incarcerated individuals in principle, without supporting them monetarily.

### 9.2 Controversial Labels

In the annotation procedure, it can be challenging to choose a single label when users express both anger and grief in a single post. In these instances, we aim to capture and annotate the tweet's dominant theme.

For example, the post "Fuck Da System Fuck Da State Fuck 12z 🖕😔" contains elements of aggression–the word "fuck" and the middle finger emoji–and loss–the sad frown emoji and the post's larger context. This post fits the *aggression* label, by challenging authority and insulting, and the *loss* label, by evoking incarceration. However, a *loss* label is more suitable, because: 1) though the anger seems to target specific players, the text as a whole reveals that the poster is highlighting

institutional violence; and 2) the sad frown emoji, paired with the context that one of the poster's loved ones was recently arrested on a drug charge ("12z" refers to law enforcement's drug unit), cements that the user is grieving in this post.

## 10 Conclusions and Future Directions

In this paper, we interpret the previous state-of-the-art model, find it is biased against stop words, create a dataset of rationales, develop metrics to measure the difference between how models and humans make predictions, design an adversarial approach to test the robustness of the model, and propose a neural network architecture to incorporate rationales. Our work has an important message to the broader NLP research community that develops tools for high stakes domains: the dominant evaluation methods in current research approaches, F-score and qualitative error analysis, are far from enough to develop good systems that might potentially lead to bad social consequences. We need various different approaches to evaluate and build a model, and each of them requires further research for improvement:

*Interpretability*: A solid understanding of model behavior can inform researchers of the potential weaknesses and guide the process of developing corresponding solutions. When trying to interpret Chang et al. (2018)'s deep learning model, we find it biased against many function words. Still, interpretation of neural network is an ongoing research topic; even though we verified "second order consistency", future work is needed for better and more faithful explanations.

*Robustness under Adversary*: It is better to break the system before it is broken when it is deployed. We find the previous model to be susceptible to insertions of certain function words, while our model is more robust against them. Nevertheless, the model is not winning every column in table 5, so there is still room for improvement.

*Metrics Evaluating How Predictions are Made*: We need to quantitatively evaluate the difference between how models and humans classify tweets. We develop the metrics "rationale rank" and "prediction flips under adversary", and verify that our model behaves more similar to domain experts. F-score does not reflect this important improvement. Additionally, uni-gram (lexical level) influence is only a shallow approximation of human reasoning - a process that involves understanding the semantics of a tweet as a whole and analyzing its pragmatics in the real-world context. Ideally we hope to find a formal language to approximate such a process and compare it against the model's interpretation.

*Models that Incorporate Rationales*: Our new model, though simple and straightforward, successfully makes use of the rationales and outperforms the previous model under most robustness metrics proposed in this paper; we believe that more sophisticated modeling of rationales can push the performances further, both in terms of accuracy and robustness.

*Feature Evaluation*: Performance improvement caused by additional features does not entail that they are used by the model in an expected way. We verify the claim of Chang et al. (2018) that the pairwise interaction feature does capture user retweet/mention interactions.

*Domain Experts*: In our domain, stakes are high and user behavior is frequently subject to biases and misinterpretations. Therefore, we ask domain experts rather than crowd workers to extend the annotation, and ask them to provide further insights about a domain that computer scientists are usually not at all familiar with. Furthermore, we should keep a critical eye on the labels and be aware of the limitations of our evaluation metrics for model accuracy. It is convenient for computer scientists to regard the labels as ground truth, the dataset as static and try different models to improve the f-score. However, as domain experts gained a deeper understanding about the twitter users, they marked 12% of the aggression labels as controversial (section 3); the annotation may evolve over time. Indeed, we need a consistent method and dataset to compare different models fairly, but we should also keep in mind that f-score is a proxy and might not necessarily reflect an improvement in the real-world.

## References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. *arXiv preprint arXiv:1804.07998*.

Petr Bělohlávek. 2017. Using adversarial examples in natural language processing.

Terra Blevins, Robert Kwiatkowski, Jamie Macbeth, Kathleen McKeown, Desmond Patton, and Owen Rambow. 2016. Automatically processing tweets

from gang-involved youth: towards detecting loss and aggression. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2196–2206.

Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Advances in neural information processing systems*, pages 4349–4357.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Serina Chang, Ruiqi Zhong, Ethan Adams, Fei-Tzin Lee, Siddharth Varia, Desmond Patton, William R. Frey, Chris Kedzie, and Kathy McKeown. 2018. Detecting gang-involved escalation on social media using context. In *EMNLP*.

Sam Corbett-Davies and Sharad Goel. 2018. The measure and mismeasure of fairness: A critical review of fair machine learning. *arXiv preprint arXiv:1808.00023*.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2017. Hotflip: White-box adversarial examples for text classification. *arXiv preprint arXiv:1712.06751*.

Avi Feller, Emma Pierson, Sam Corbett-Davies, and Sharad Goel. 2016. A computer program used for bail and sentencing decisions was labeled biased against blacks. its actually not that clear. *The Washington Post*.

Shi Feng, Eric Wallace, II Grissom, Mohit Iyyer, Pedro Rodriguez, Jordan Boyd-Graber, et al. 2018. Pathologies of neural models make interpretations difficult. *arXiv preprint arXiv:1804.07781*.

Jennifer Golbeck, Zahra Ashktorab, Rashad O Banjo, Alexandra Berlinger, Siddharth Bhagwan, Cody Buntain, Paul Cheakalos, Alicia A Geller, Quint Gergory, Rajesh Kumar Gnanasekaran, et al. 2017. A large labeled corpus for online harassment research. In *Proceedings of the 2017 ACM on Web Science Conference*, pages 229–233. ACM.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Tommi Gröndahl, Luca Pajola, Mika Juuti, Mauro Conti, and N Asokan. 2018. All you need is" love": Evading hate-speech detection. *arXiv preprint arXiv:1808.09115*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. *arXiv preprint arXiv:1707.07328*.

Max Kapustin, Jens Ludwig, Marc Punkay, Kimberley Smith, Lauren Speigel, and David Welgus. 2016. Gun violence in chicago, 2016.

Svetlana Kiritchenko and Saif M Mohammad. 2018. Examining gender and race bias in two hundred sentiment analysis systems. *arXiv preprint arXiv:1805.04508*.

Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1885–1894. JMLR. org.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220*.

Shulin Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2017. Exploiting argument information to improve event detection via supervised attention mechanisms. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1789–1798.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Nitin Madnani, Anastassia Loukina, Alina von Davier, Jill Burstein, and Aoife Cahill. 2017. Building better open-source tools to support fairness in automated scoring. In *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*, pages 41–52.

Haitao Mi, Zhiguo Wang, and Abe Ittycheriah. 2016. Supervised attentions for neural machine translation. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Desmond U Patton, Sadiq Patel, Jun Sung Hong, Megan L Ranney, Marie Crandall, and Lyle Dungy. 2017. Tweets, gangs, and guns: a snapshot of gang communications in detroit. *Violence and victims*, 32(5):919–934.

Desmond Upton Patton, Robert D Eschmann, and Dirk A Butler. 2013. Internet banging: New trends in social media, gang violence, masculinity and hip hop. *Computers in Human Behavior*, 29(5):A54–A59.

Desmond Upton Patton, Robert D Eschmann, Caitlin Elsaesser, and Eddie Bocanegra. 2016. Sticks, stones and facebook accounts: What violence outreach workers know about social media and urban-based gang violence in chicago. *Computers in human behavior*, 65:591–600.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144. ACM.

Rachel Rudinger, Chandler May, and Benjamin Van Durme. 2017. Social bias in elicited natural language inferences. In *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*, pages 74–79.

Hendrik Strobelt, Sebastian Gehrmann, Michael Behrisch, Adam Perer, Hanspeter Pfister, and Alexander M Rush. 2019. S eq 2s eq-v is: A visual debugging tool for sequence-to-sequence models. *IEEE transactions on visualization and computer graphics*, 25(1):353–363.

Eric Wallace, Pedro Rodriguez, Shi Feng, and Jordan Boyd-Graber. 2018. Trick me if you can: Adversarial writing of trivia challenge questions. *arXiv preprint arXiv:1809.02701*.

Mo Yu Regina Barzilay Yujia Bao, Shiyu Chang. 2018. Deriving machine attention from human rationales. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2017. Men also like shopping: Reducing gender bias amplification using corpus-level constraints. *arXiv preprint arXiv:1707.09457*.

Ruiqi Zhong, Steven Shao, and Kathleen McKeown. 2019. Fine-grained sentiment analysis with faithful attention. *arXiv preprint arXiv:1908.06870*.

# CodeSwitch-Reddit: Exploration of Written Multilingual Discourse in Online Discussion Forums

**Ella Rabinovich** **Masih Sultani** **Suzanne Stevenson**
Dept. of Computer Science, University of Toronto, Canada
{ella,masih,suzanne}@cs.toronto.edu

## Abstract

In contrast to many decades of research on oral code-switching, the study of written multilingual productions has only recently enjoyed a surge of interest. Many open questions remain regarding the sociolinguistic underpinnings of written code-switching, and progress has been limited by a lack of suitable resources. We introduce a novel, large, and diverse dataset of written code-switched productions, curated from topical threads of multiple bilingual communities on the Reddit discussion platform, and explore questions that were mainly addressed in the context of spoken language thus far. We investigate whether findings in oral code-switching concerning content and style, as well as speaker proficiency, are carried over into written code-switching in discussion forums. The released dataset can further facilitate a range of research and practical activities.

## 1 Introduction

Multilingual communities adopt various communicative strategies that navigate among multiple languages. One of the most notable of such strategies is *code-switching* (CS) – when a bilingual mixes two or more languages within a discourse, or even within a single utterance, as in:

εγώ θα το ήθελα, θα ήταν το guilty pleasure μου
[ *I would like it, it would be my* guilty pleasure ]

The sociolinguistic underpinnings of code-switching as an *oral* conversational strategy have been investigated extensively for many decades. The purposes of oral CS have been shown to range from practical considerations of domain-specific lexical knowledge, to indications of register and nuanced meanings (see Gardner-Chloros, 2009, for a recent review). Oral code-switching is thus known to be a multifaceted component of successful communication in a multilingual community, interacting with factors of speaker proficiency, and both style and content of utterances.

By contrast, the analysis of *written* code-switching has only recently enjoyed a surge of interest, and remains seriously under-studied (Sebba et al., 2012). Written text often differs greatly from conversation in its levels of both spontaneity and formality, and findings thus far have differed in their conclusions regarding the extent to which various genres of written text reflect the same communicative functions of CS as observed in oral conversation (e.g., McClure, 2001; Chan, 2009; Gardner-Chloros and Weston, 2015).

The growing popularity of social media and online discussion platforms poses both opportunities and new research questions regarding written code-switching. Global online forums, in which English is a lingua franca, not only draw on but create wide-reaching multilingual communities. The resulting communications lead to a wealth of data that potentially includes a large amount of code-switching across multiple language pairs (e.g., Solorio et al., 2014; Diab et al., 2016; Aguilar et al., 2018b). Moreover, communication on discussion platforms often resembles a hybrid between speech and more formal writing (Sebba, 2013). These differing characteristics lead to new research questions regarding the extent to which findings from oral CS carry over to these online interactions.

Research is only just beginning to grapple with these issues. Computational work on code-switching in online venues has largely focused on the practical challenges that multiple interleaved languages pose to the application of standard NLP tools (as reviewed in Çetinoğlu et al., 2016), rather than on the communicative purposes of CS. More broadly, computational investigation of the sociolinguistic aspects of written CS is dominated by studies conducted with a limited number of



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4776–4786,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

language-pairs and/or authors (e.g., Sebba et al., 2012), thereby constraining the nature of questions that can be addressed with this data.

Our work here seeks to address these gaps in the study of code-switching in online interactions. We begin by introducing the *CodeSwitch-Reddit* corpus:[1] a novel, large, and diverse dataset of written code-switched productions, carefully curated from topical threads of multiple (including understudied) bilingual communities on the Reddit discussion platform.[2] The main corpus comprises over 135K code-switched messages by over 20K unique authors, spanning five language-pairs, with average post length of 75 tokens.

The uniform nature of our data (written communication from a single online discussion platform), as well as its ample size, pose novel opportunities for large-scale empirical investigation of research questions on code-switching – questions that have thus far been mainly addressed in the context of oral language. As a first study, here we explore fundamental questions about both the content and style of code-switched posts, as well as about the English proficiency level of authors who frequently code-switch. Due to the size and breadth – yet homogeneity – of our corpus, we can explore these questions with appropriate comparisons to monolingual posts in the same genre and register, and by the same authors.

The contribution of this work is, therefore, twofold: First, we construct a novel code-switching corpus, whose size, number of language pairs, and diversity of content (consisting of posts of unrestricted length in a wide variety of topic areas) make it a desirable testbed for a range of research questions on code-switching in online discussion forums. Second, we demonstrate the usefulness of this dataset through an empirical investigation that sheds new light on postulated universals of code-switching – involving linguistic proficiency, style, and content – when inspected through the lens of online communication.

## 2 Compilation of CodeSwitch-Reddit

One of the contributions of this work is the CodeSwitch-Reddit corpus – a large and diverse corpus comprising multilingual posts from online discussion forums. We selected Reddit for construction of our corpus due to its size (over 200M unique users), its structure of content-focused subcommunities (over 100K active subreddits), and the unrestricted length of its posts (in contrast to Twitter, e.g.). This enabled us to extract a large amount of code-switching in natural contexts, within multiple language pairs and by numerous authors, and with access to much monolingual text by the same authors for comparative purposes. Because English is the lingua franca of Reddit, we restricted our focus to CS between English and another language, ensuring commonality in one language across all CS posts.

### 2.1 Initial Data Extraction

Effective use of CS relies on interactions between bilinguals, hence we needed to identify subreddits likely to contain posts in multiple languages. We observed that country-specific subreddits (e.g., r/greece and r/philippines) often contained posts both in English and in the language of the country specified (e.g., Greek and Tagalog, respectively). We thus restricted our extraction to all country-specific subreddits, except for countries with English as a national language, e.g., r/Australia.

We collected all posts (both initial submissions and subsequent comments/replies) from these subreddits using an API[3] designed for searching Reddit content. Among the properties collected for each post are the unique userid of the author, subreddit name, date of posting, and text of the message. Additional metadata properties include information regarding the full conversational chain of a post (e.g., links to 'parent' messages), facilitating planned further extension of our corpus with contextual information. This final raw dataset consisted of over 6.88M posts from 71 country-specific subreddits.

### 2.2 Identifying Code-switching Posts

We first identified posts that contained multiple languages, of which one was English, by using polyglot,[4] a probabilistic tool that identifies the languages present in a (multilingual) text, along with their approximate percentage of the total text. This yielded a set of posts where two languages – English and another one – were detected. At this stage, we removed posts shorter than five tokens and those containing weblinks (which misled the language identification tool).

[1] All data and code are available at https://github.com/ellarabi/CodeSwitch-Reddit

[2] https://www.reddit.com/

[3] https://github.com/pushshift/api

[4] https://polyglot.readthedocs.io

Second, we needed to automatically determine which of these multilingual posts actually contained code-switching. We followed a relatively strict notion of CS (Bullock and Toribio, 2009) as a fluid alternation between two languages in an author's own words. Due to the inherently conversational nature of Reddit discussions, authors may include in their own post (parts of) another message they are replying to. We did not consider a post CS if the author's own text was in a single language, but the 'reply-to' text contained another language. In addition, we aimed to exclude from our code-switching corpus any posts whose only alternation between two languages was a use of: named entities (e.g., 'Amazon'); quotes (from websites, books, movies, etc.; e.g., "Don't cry because it's over, smile because it happened."); or translations (text in one language along with its equivalent in the other language).

To enforce this definition of CS, we applied the following filters to our set of multilingual posts. Because our dataset was plentiful, we aimed at achieving high precision (occasionally at the cost of recall) to ensure that resulting posts were likely to have true examples of code-switching.

**Replies:** A reply-to section is typically preceded by a unique sequence of characters. We used a regular expression to remove all reply-to section(s).

**Named entities:** We removed all named entities identified by a multilingual Named-Entity Recognizer (NER) available in spaCy.[5]

**Quotes:** Quotation marks are used both to indicate actual quotes, as well as to convey emphasis of a word or phrase, so they are a clear but noisy cue. We used a regular expression to locate text within quotation marks, and remove it if longer than 5 tokens (since quotation marks around short phrases were often for emphasis). Other methods for quoting, such as copy and pasted text, could not be reliably identified; these are the largest source of false positives of CS in our corpus.

**Translations:** Here we sought to identify words that indicated that text in one language was likely a translation from the other language. Exploring lexical characteristics typical to r/translator, a subreddit for translation requests, we constructed a list of such words, and removed from our dataset all posts including any word on this set. The list of terms can be found in supplemental material (A.1).

After obtaining a set of 'clean' posts by removing sections/posts identified by these filters, we applied the polyglot tool once again. If two languages (English and another) were still present, we identified the posts as code-switched. In order to retain as much contextual information as possible, we reinserted back into the code-switched posts any named entities and quotes that were removed by the filters above. Table 1 presents some examples of code-switched utterances in the dataset, both intra- and inter-sentential.

### 2.3 Precision of CS Identification

The method above for identifying code-switched posts may lead to false positives, due to either polyglot errors or inadequacies in our filters. To evaluate the precision of the resulting corpus, we obtained manual annotations, as actual CS or not, of a random sample of the posts.[6]

For language pairs with a large amount of data in the compiled dataset – English plus each of Tagalog, Greek, Romanian, Indonesian, Russian, Spanish, and French – we used FigureEight,[7] a crowd sourcing platform designed to support large annotation tasks. The tasks were split by individual language-pairs, with each including 500 randomly sampled posts from that pair in our corpus. Aiming at reaching bilingual speakers, the tasks were released to countries where the non-English language of the pair was a national language of that country (e.g., Greece for English-Greek).

Annotators were provided with detailed instructions and plentiful examples to indicate precisely what we considered to be code-switching; sample guidelines can be found in supplemental material (A.2). Only those who successfully passed a (manually compiled) quiz were allowed to perform the task. We asked the annotators to indicate yes/no whether an utterance was code-switched or not. In the case of a 'no' answer, they were prompted to provide a reason from these options: (1) no second language present; or, the presence of a second language is due to (2) a named entity, (3) a quote, or (4) other.

Each post was annotated by three annotators, and 100% agreement (three 'yes' or three 'no')

[5] https://spacy.io/

[6] We did not evaluate recall, given that our goal here was to collect a large set of CS, not to detect all CS.

[7] https://www.figure-eight.com/

| |
|---|
| Not valid enough for me. 7 PM pa daw kasi out niya at hihintayin ko pa rin daw siya eh<br>lagi ko namang ginagawa yun kapag magdidinner kami after work. So what's the difference this time? |
| Intreb pentru un prieten, care inca scrie SQL, si va scrie mult timp de acum inainte...<br>FYI, relational databases ain't gonna go away any time soon... |
| Πράγματι, ήταν too good to be true. |

Table 1: Examples of code-switched posts in English-Tagalog, English-Romanian, and English-Greek.

was achieved on 83.4% of the posts across all tasks. English-Indonesian and English-Tagalog had the highest percentage of unanimous annotation – 96% and 95%, respectively, followed by Greek (85%), Romanian (82%), and Russian (60%).[8] A majority of 'yes' answers indicated a true positive (actual CS post), and other patterns a false positive (not code-switching). The English-Tagalog and English-Indonesian tasks yielded 99% precision, the English-Romanian and English-Greek tasks 87%, and the English-Russian task 85%. The English-Spanish task had 70% of posts identified as code-switched, and the accuracy of English-French was extremely low; on manual inspection this appeared to be due to the high extent of shared lexical items, misleading the language identification tool.

Due to difficulty in securing bilingual workers on FigureEight, additional language pairs (with a smaller amount of data) were evaluated by an in-house team of annotators, with a relevant bilingual background where possible, or using an automatic translation tool. Five language pairs were identified as comprising between 70% and 85% code-switched posts.

### 2.4 Final Corpus and Additional Datasets

The final CodeSwitch-Reddit corpus includes the five language pairs that had 85% or higher precision. Table 2 (left) reports details on the corpus, which has over 135K posts, with an average of 75 tokens per post, thereby introducing a unique resource facilitating further research in this field. (A sample of English-Tagalog posts are in supplemental materials.) Table 2 (right) reports the details of the five additional language pairs that had at least 70% precision. Because of the lower quality, we omit these language-pairs from our experiments below. However, we recognize the potential usefulness of this additional data, and release it as an addendum to our main corpus, for possible further cleanup and preprocessing.

We also compiled an additional dataset of English *monolingual* posts from the same country-specific subreddits as our code-switched corpus. Here, as above, we excluded posts with weblinks and posts shorter than five words. This dataset serves as an appropriate comparison set for the code-switching corpus, and is used for some analyses in Section 3.

## 3 Sociolinguistic Aspects of Online CS

A large body of research has highlighted code-switching as a strategy used to shape the dynamics of the sociolinguistic situation, including aspects of identity, interpersonal relationships, and formality (e.g., Gardner-Chloros, 2009). In addition, studies have differed in their conclusions on whether those who code-switch in speech are highly adept bilinguals (e.g., Kootstra et al., 2012), or use code-switching to address lexical deficiencies (e.g., Poulisse, 1989). Using our new corpus, we aim to explore how such findings from the research on CS in oral communication apply in online written communication. Namely, we test whether code-switched text on the Reddit discussion platform is typified by unique topical and informality markers, and whether authors who frequently mix languages in their posts differ in their English proficiency level from those who do not. Our dataset, comprising communication in multiple language-pairs, as well as monolingual English writing by thousands of authors within the same register introduce an appropriate testbed for addressing these questions.

### 3.1 Topical Preferences in CS Text

We take a topic modeling approach to explore whether CS productions are characterized by a unique topical flavor, when compared to monolingual messages. Specifically, we compare two topic models, one created using English text from the CodeSwitch-Reddit corpus, and a second us-

[8] Manual inspection of English-Russian annotations revealed a wide range of (in part, subjective) usage cases employing CS; this sub-corpus accounts for 1.4% of our data.

| language-pair | authors | posts | avg post len | language-pair | authors | posts | avg post len |
|---|---|---|---|---|---|---|---|
| English-Tagalog | 12,159 | 88,038 | 74.6 | English-Spanish | 2,461 | 4,225 | 105.4 |
| English-Greek | 3,032 | 24,585 | 59.6 | English-Turkish | 635 | 1,459 | 85.3 |
| English-Romanian | 2,516 | 10,008 | 104.5 | English-Arabic | 582 | 1,018 | 65.8 |
| English-Indonesian | 1,851 | 10,354 | 84.5 | English-Croatian | 437 | 632 | 104.6 |
| English-Russian | 934 | 2,055 | 54.3 | English-Albanian | 187 | 482 | 86.2 |
| total | 20,492 | 135,313 | 75.5 | | 4,302 | 7,816 | 89.5 |

Table 2: Statistics of the main CodeSwitch-Reddit corpus (left) and the additional CS dataset (right).

ing the monolingual dataset from the same subreddits (see Section 2.4). We use only the text of each dataset from *a common set of authors* whose posts appear in both, thereby ensuring that any differences we find are likely due to the context of code-switching rather than individual differences in content between authors.

We identified a set of $4,843$ authors who posted both code-switched and monolingual messages in our datasets, resulting in $58,568$ CS posts and $58,603$ monolingual posts, having a minimum of $50$ tokens and averaging $108.7$ and $103.7$ tokens per post, respectively. We tokenized and lemmatized texts, excluded English stopwords, and removed name entities and tokens other than nouns, verbs, adjectives and adverbs using python spaCy tools. We applied a dictionary-based approach in order to filter out non-English words; we removed all words ranked higher than $10,000$ in the word-rank list built from the English Wikipedia corpus.[9]

We used a publicly-available topic modeling tool (MALLET, McCallum, 2002) to identify the prevalent discussion topics in the CS and monolingual messages. Each topic is represented by a probability distribution over the entire vocabulary, where terms more characteristic of a topic are assigned a higher probability. A common way to evaluate a topic learned from a set of documents is by computing its *coherence score* – a statistical measure reflecting the quality of a topic (Newman et al., 2010). The quality of a learned model is then estimated by averaging the scores of its individual topics – the *model* coherence score. We selected the optimal number of topics for each set of posts by maximizing its model coherence score, resulting in $17$ topics for CS posts (score of $0.58$), and $21$ topics for monolingual posts (score of $0.60$).

**Qualitative analysis:** We first examine similarities and differences across the two topical distributions by extracting the top $5$ topics – those with the highest individual coherence scores – in each of the code-switched and monolingual models. Table 3 presents the $15$ most probable words for these top-five topics in each of the code-switched and monolingual models (on the left and right sides, respectively); topics within each are ordered by decreasing coherence score (left to right).

While both CS and monolingual posts include a topic expressing *sentiment about events* (topics CS-3 and M-1), CS posts have two further topics that particularly discuss *relationships* and *family* (CS-2, CS-4). This suggests that the choice to switch languages on Reddit can delineate social, personal, and emotional territory. Other topics typify both CS and monolingual posts, discussing *studies* (CS-1, M-4) and *shopping/travel* (CS-5, M-5). Discussions on *politics/issues* are prevalent only in monolingual communication (M-2, M-3), again indicating that CS posts are more focused on personal experiences.

Table 4 presents a few examples of code-switched posts associated most with the *relationship* topic in the model. It is worth noting that this is one of the subjects most prominently associated with code-switching in speech. The fluid and seamless interleaving of languages highlights the way that emotion, sentiment, and personal experience can guide lexical choice in written communication on the Reddit platform.

**Quantitative analysis:** We further assess the extent of topical distinctions between code-switched and monolingual posts by performing statistical significance analysis. Namely, we randomly partition the monolingual set of posts into two equal-sized subsets and test whether the two parts exhibit higher mutual similarity than (on average) with their code-switched counterpart.

Formally, given two topic models $M_1$ with topics $T_1^1, ..., T_1^N$, and $M_2$ with topics $T_2^1, ..., T_2^K$, where $N$ and $K$ are the number of topics in $M_1$ and $M_2$, respectively, we define the similarity be-

[9] https://dumps.wikimedia.org/enwiki

| Topics with highest coherence scores in code-switched posts | | | | | Topics with highest coherence scores in monolingual posts | | | | |
|---|---|---|---|---|---|---|---|---|---|
| CS-1 | CS-2 | CS-3 | CS-4 | CS-5 | M-1 | M-2 | M-3 | M-4 | M-5 |
| student | friend | make | give | phone | thing | vote | problem | learn | plan |
| exam | feel | thing | parent | check | person | power | change | student | price |
| study | girl | happen | live | driver | love | political | issue | study | cost |
| teacher | love | talk | chance | internet | call | support | reason | experience | cheap |
| grade | date | wrong | reason | price | wrong | rule | fact | teach | sell |
| graduate | close | word | plan | mall | happen | majority | deal | education | stay |
| pass | break | hate | late | shop | hate | politician | current | skill | travel |
| subject | relationship | joke | decide | traffic | understand | citizen | control | teacher | visit |
| program | happy | sense | child | store | kind | election | matter | research | expensive |
| review | meet | mind | baby | brand | accept | leader | point | test | compare |

Table 3: Most coherent topics identified in code-switched and monolingual posts by the same set of authors.

tween two models as the average of their pairwise topic similarities:

$$sim(M_1, M_2) = \frac{1}{N \times K} \sum JSC(T_1^i, T_2^j) \quad (1)$$

where $i \in [1..N]$, $j \in [1..K]$, and the Jaccard similarity coefficient (JSC) is used to measure the extent of overlap between the top 100 terms associated with each of $T_1^i$ and $T_2^j$.

Given model $M_{cs}$ learned from the code-switched corpus, and models $M_{en}^1$ and $M_{en}^2$ from the two partitions of the monolingual posts, we can then compute two similarity metrics:

$$avg(sim(M_{cs}, M_{en}^1), sim(M_{cs}, M_{en}^2)) \quad (2)$$

and

$$sim(M_{en}^1, M_{en}^2) \quad (3)$$

We repeated the experiment with 30 random partitions of the monolingual corpus, resulting in mean *inter*-corpora similarity (Equation 2) of 0.030, and mean *intra*-corpus similarity (Equation 3) of 0.037.[10] A Wilcoxon rank-sum test for significance of the differences yielded a $p$-value of $2.87e$-11. Performing the same analysis with `max` instead of `average` topical similarity in Equation 1 (i.e., estimating sub-corpora topical similarity through two maximally similar topics), yields mean *inter*-corpora similarity of 0.384, and mean *intra*-corpus similarity of 0.470, with $p$-value of $4.99e$-7. We, therefore, conclude that significant differences can be observed in topical preferences in code-switched messages compared to monolingual texts in our datasets.

[10]The extremely low similarity scores reflect the (desirable) low average similarity between truly different topics.

### 3.2 (In)formality of code-switched content

Oral code-switching has been shown to be a marker of informality (Genesee, 1982; Myers-Scotton, 1995). While user-generated content on the Reddit discussion platform is by and large considered informal, in this section we test whether (presumably subtle) formality differences can be observed in CS posts compared to monolingual ones. In accord with findings in spoken language, we expect more informal context to stimulate more frequent language mixing.

We use the formal-informal GYAFC parallel dataset by Rao and Tetreault (2018) to extract a set of markers indicative of informal style. Originally collected from the Yahoo Answers platform, the corpus comprises over 50K sentence-pairs in the domain of Entertainment&Music and Family&Relationships, where each original *informal* sentence is paired with a manually generated *formal* counterpart. The parallel nature of the corpus facilitates extraction of a clean set of stylistic markers characteristic of informal text, while abstracting away from content. We use the *log-odds*

| |
|---|
| Entahlah, I'm self diagnosing, but maybe I'm on the spiral to depression...<br>[ *I've no idea*, I'm self diagnosing, but maybe I'm on the spiral to depression... ] |
| If you somehow read this... Лена, я дурак. Мне очень жаль. Я очень по тебе скучаю.<br>[ If you somehow read this... *Lena, I'm a fool. I'm sorry. I miss you so much.* ] |
| Lovely.... whataboutism at it's finest. Επισης κάνεις κάτι ατυχέστατες υποθέσεις.<br>[ Lovely.... whataboutism at it's finest. *You're also doing something wrong.* ] |

Table 4: Examples of CS posts (with *translations*) associated with emotion, sentiment, and relationships.

*ratio informative Dirichlet prior* (Monroe et al., 2008) to discover markers appearing more in informal style, by comparing the informal part of the GYAFC dataset to its formal part. We used the strict log-odds score of $-5$ as a threshold for collecting terms associated with informal writing. Among the markers identified by this procedure were {*!, ..., u, dont, n't, its, just, ur, im, thats, like, na, &, really, got, lol, cant, ya, coz, alot, but, yea, doesnt, so, kinda, hey, dude, 've, pretty*}.

Using the same sets of posts as in Section 3.1, we tested our hypothesis by inspecting these markers of informality in CS posts compared to monolingual posts by the same set of authors. Specifically, each author is assigned two measurements, reflecting the frequency of the informality markers in the entire collection of their code-switched posts and of their monolingual English posts. Frequency was computed by normalizing the total count of informality markers in an author's text (concatenation of individual posts) by its length.

The mean frequency of informality markers in CS posts was 0.160, while in monolingual posts by the same authors was 0.153. Figure 1 illustrates the kernel density estimation of these frequencies: the subtle deviation in formal style is mirrored by the slight right shift of the density function for code-switched texts. Moreover, a (paired difference) Wilcoxon signed-rank test for statistical significance of the difference in CS and monolingual posts resulted in a within-subjects effect size of 0.15 and a very low $p$-value ($1.01e$-23). Despite the generally high level of informality of Reddit postings, we see a small but reliably detected difference of even greater informality in CS utterances, mirroring the observations in oral CS being associated with informal speech.

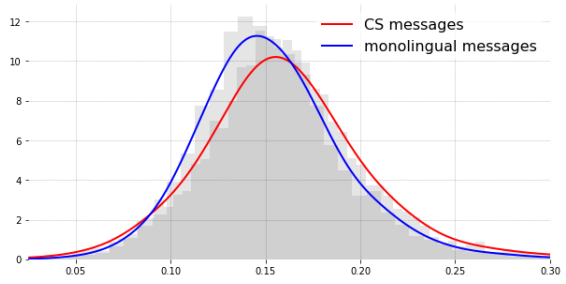


Figure 1: Kernel density estimate of informality markers' frequency in CS and monolingual messages.

### 3.3 Code-switching and language competence

Theoretical and experimental studies have differed in considering code-switching as a strategy of extremely proficient bilinguals, or as a coping mechanism of those with deficiencies in one of the languages (see Section 4). We address this issue in the context of written communication by computing a range of lexical and grammatical metrics from utterances of authors who tend to mix languages frequently, compared to those who do not.

We separated authors who were active contributors to bilingual subreddits into those who frequently code-switch vs. those who do not. We identified 288 authors as 'high code-switchers' – those who have over 100 messages in a country-specific subreddit, where at least 20% of these posts were code-switched. We further identified 262 authors as 'low code-switchers', having over 100 posts but mixing languages in less than 2% of their messages. Aiming at a reliable assessment of these authors' English linguistic skills, we gathered their entire monolingual English digital footprint from the Reddit platform, including their posts in all but country-specific subreddits. The collected set of messages contains an average of 1,060 and 856 posts per author, for high and low code-switchers, respectively.

We estimate linguistic competence by computing a set of lexical and grammatical measures commonly used for language proficiency assessment (Kyle and Crossley, 2015; Lu and Ai, 2015; Rabinovich et al., 2018). All posts were tokenized and lemmatized, and non-alphabetic tokens were excluded from the computation. We produce the following metrics:

**Normalized type-to-token ratio** (NTTR) – the ratio of the number of unique tokens and the total number of tokens, averaged over non-overlapping sliding windows of 1000 tokens.

**Lexical density** (lex. density) – the ratio of the number of content tokens (excluding function words) and the number of all tokens, using 426 function words from Koppel and Ordan (2011).

**Mean average age of acquisition** (mean AoA) – mean of the average AoA of tokens for over 30K English words (Kuperman et al., 2012).

**Mean word concreteness** (word conc.) – mean word concreteness of tokens, taken from ratings for 40K English words (Brysbaert et al., 2014).

**Mean word length** (word length) – the average

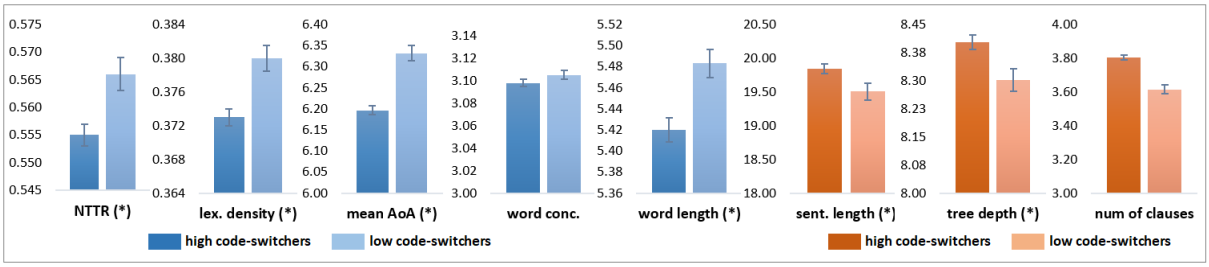


Figure 2: Linguistic proficiency metrics (mean and standard error; lexical on the left, grammatical on the right) of high and low code-switchers. '*' indicates significant difference in a Wilcoxon rank-sum test at the level of $p < .05$.

length in characters of a token.

**Mean sentence length** (sent. length) – the average length of a sentence (in tokens).

**Mean parse tree depth** (tree depth) – the average depth of a parse tree of a sentence, using the constituency parser of Kitaev and Klein (2018).

**Mean number of clauses** (num of clauses) – the average number of clauses in a sentence.

Figure 2 presents these metrics for high and low code-switchers. We find that high code-switchers exhibit statistically significant *lower* proficiency in the lexical metrics (with the exception of word concreteness level), but statistically significant *higher* proficiency for two of the grammatical metrics. Although the differences in both cases are small, they are reliably detected given the size of the corpora. Interestingly, while the lexical measures indicate less 'sophisticated' English words used by the high code-switchers, the distinctions in their grammatical complexity (reflected in longer sentences and greater parse tree depths) imply that mixing language in written communication may not necessarily be a manifestation of *overall* lower linguistic competence. These intriguing findings suggest that code-witching in online forums is a multi-faceted phenomenon: while bridging lexical deficiencies, it may require advanced grammatical capabilities in order to construct mixed sentences without distorting the 'grammaticality' of the target utterance. These results both demonstrate the value of studying code-switching at scale, and leave room for much further investigation. Our future plans include breaking down the data by individual native languages of the authors, in order to capture differences in proficiency stemming from various L1 backgrounds.

## 4 Related Work

Many studies have elucidated the communicative functions of oral code-switching. These include conveying nuanced attitudes through linguistic choices reflecting emotion and sentiment (Dewaele and Nakano, 2013), as well as establishing a level of (in)formality (Fishman, 1970; Genesee, 1982; Myers-Scotton, 1995). Other studies have focused more on the characteristics of code-switchers themselves: While CS has mostly been viewed as a deliberate choice of a proficient bilingual speaker, requiring adept control of two simultaneously-activated linguistic systems (Costa and Santesteban, 2004; Kootstra et al., 2012), mixing two languages may also serve as a strategy for bridging lexical inefficacy in a second language (Grosjean, 1982; Faerch and Kasper, 1983; Poulisse, 1989).

As noted earlier, written CS is much less studied and the extent to which these findings hold across various written genres is not clear (Sebba et al., 2012). Recent research from the sociolinguistic perspective has considered CS in an array of online genres: e.g., English-Jamaican Creole in email (Hinrichs, 2006), Malay-English language in online discussion forums (McLellan, 2005), English-Spanish bilingual blogging (Montes-Alcalá, 2007), and English-Chinese instant messaging (Lam, 2009). However, these studies have analyzed productions of a small number of authors (even a single person), and each is restricted to a single language-pair.

Most large-scale computational work on CS in social media has focused on essential prerequisites for various NLP tasks, such as POS tagging (Solorio and Liu, 2008; Soto et al., 2018), token-level language identification (Soto et al., 2018; Rijhwani et al., 2017), NER (Aguilar et al., 2018a), language modeling (Chandu et al., 2018), automatic speech recognition (Yılmaz et al., 2018),

etc. Almost no work has addressed theoretical and sociolinguistic aspects of written multilingual discourse computationally, at scale (for an exception, see Rudra et al., 2016).

Our work here addresses various gaps noted in the research above. We create (and make available) a novel dataset that enables investigation of research questions on written code-switching at scale, across multiple language pairs and a large number of speakers. Moreover, we use this dataset for computational, large-scale investigation of subtle issues of linguistic content and style, as well as speaker proficiency, that have been extensively studied in the context of spoken language, and enjoyed very limited attention in written communication.

## 5 Conclusion

Despite the presumed differences between code-switching in spoken and written communication, multilingual written discourse is understudied, partially due to the lack of adequate resources. In this work we introduced the CodeSwitch-Reddit corpus – a large, diverse and carefully curated dataset of code-switched posts collected from multiple multilingual discussions forums on Reddit. Computational investigation of this dataset reveals topical and stylistic distinctions between code-switched and monolingual communication, as well as differences in the proficiency level of authors who mix languages frequently, compared to those who do not. Our future plans include extending this dataset with conversational context and investigating additional aspects of online written multilingual discourse in richer contexts.

The released dataset suggests a wealth of both intra- and inter-sentential code-switched utterances, and investigation of sociolinguistic differences between the two introduces an interesting research question. As an example, while mixing languages within a sentence could be indicative of lexical deficiency, that may not necessarily be the case with inter-sentential code-switching.

## Acknowledgments

This research is supported by an NSERC Discovery Grant RGPIN-2017-06506 to S. Stevenson. We are thankful to Joel Tetreault for providing us the formal-informal GYAFC parallel corpus. We are also grateful to our anonymous reviewers for their insightful and constructive feedback.

## References

Gustavo Aguilar, Fahad AlGhamdi, Victor Soto, Mona Diab, Julia Hirschberg, and Thamar Solorio. 2018a. Named entity recognition on code-switched data: Overview of the CALCS 2018 shared task. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*. pages 138–147.

Gustavo Aguilar, Fahad AlGhamdi, Victor Soto, Thamar Solorio, Mona Diab, and Julia Hirschberg. 2018b. Proceedings of the third workshop on computational approaches to linguistic code-switching. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*.

Marc Brysbaert, Amy Beth Warriner, and Victor Kuperman. 2014. Concreteness ratings for 40 thousand generally known English word lemmas. *Behavior research methods* 46(3):904–911.

B. Bullock and A. Toribio. 2009. *The Cambridge Handbook of Linguistic Code-switching*. https://doi.org/10.1017/CBO9780511576331.009.

Özlem Çetinoğlu, Sarah Schulz, and Ngoc Thang Vu. 2016. Challenges of computational processing of code-switching. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching*. pages 1–11.

Brian Hok-shing Chan. 2009. English in Hong Kong Cantopop: Language choice, code-switching and genre. *World Englishes* 28(1):107–129.

Khyathi Chandu, Thomas Manzini, Sumeet Singh, and Alan W Black. 2018. Language informed modeling of code-switched text. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*. pages 92–97.

Albert Costa and Mikel Santesteban. 2004. Lexical access in bilingual speech production: Evidence from language switching in highly proficient bilinguals and L2 learners. *Journal of memory and Language* 50(4):491–511.

Jean-Marc Dewaele and Seiji Nakano. 2013. Multilinguals' perceptions of feeling different when switching languages. *Journal of Multilingual and Multicultural Development* 34(2):107–120.

Mona Diab, Pascale Fung, Mahmoud Ghoneim, Julia Hirschberg, and Thamar Solorio, editors. 2016. *Proceedings of the Second Workshop on Computational Approaches to Code Switching*.

Claus Faerch and Gabriele Kasper. 1983. *Strategies in interlanguage communication*. Longman Pub Group.

Joshua A Fishman. 1970. *Sociolinguistics: A brief introduction*. ERIC.

Penelope Gardner-Chloros. 2009. *Sociolinguistic factors in code-switching*. Cambridge University Press.

Penelope Gardner-Chloros and Daniel Weston. 2015. Code-switching and multilingualism in literature. *Language and Literature* 24(3):182–193.

Fred Genesee. 1982. The social psychological significance of code switching in cross-cultural communication. *Journal of language and social psychology* 1(1):1–27.

François Grosjean. 1982. *Life with two languages: An introduction to bilingualism*. Harvard University Press.

Lars Hinrichs. 2006. *Codeswitching on the web: English and Jamaican Creole in e-mail communication*, volume 147. John Benjamins Publishing.

Nikita Kitaev and Dan Klein. 2018. Multilingual constituency parsing with self-attention and pre-training. Technical Report arXiv:1812.11760 [cs.CL]. https://arxiv.org/pdf/1812.11760.

Gerrit Jan Kootstra, Janet G Van Hell, and Ton Dijkstra. 2012. Priming of code-switches in sentences: The role of lexical repetition, cognates, and language proficiency. *Bilingualism: Language and Cognition* 15(4):797–819.

Moshe Koppel and Noam Ordan. 2011. Translationese and its dialects. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, pages 1318–1326.

Victor Kuperman, Hans Stadthagen-Gonzalez, and Marc Brysbaert. 2012. Age-of-acquisition ratings for 30,000 english words. *Behavior research methods* 44(4):978–990.

Kristopher Kyle and Scott A Crossley. 2015. Automatically assessing lexical sophistication: Indices, tools, findings, and application. *Tesol Quarterly* 49(4):757–786.

Wan Shun Eva Lam. 2009. Multiliteracies on instant messaging in negotiating local, translocal, and transnational affiliations: A case of an adolescent immigrant. *Reading Research Quarterly* 44(4):377–397.

Xiaofei Lu and Haiyang Ai. 2015. Syntactic complexity in college-level English writing: Differences among writers with diverse L1 backgrounds. *Journal of Second Language Writing* 29:16–27.

Andrew Kachites McCallum. 2002. MALLET: A machine learning for language toolkit. http://mallet.cs.umass.edu.

Erica McClure. 2001. Oral and written Assyrian–English codeswitching. *Codeswitching worldwide II* pages 157–191.

James AH McLellan. 2005. *Malay-English language alternation in two Brunei Darussalam on-line discussion forums*. Ph.D. thesis, Curtin University.

Burt L Monroe, Michael P Colaresi, and Kevin M Quinn. 2008. Fightin'words: Lexical feature selection and evaluation for identifying the content of political conflict. *Political Analysis* 16(4):372–403.

Cecilia Montes-Alcalá. 2007. Blogging in two languages: Code-switching in bilingual blogs. In *Selected Proceedings of the Third Workshop on Spanish Sociolinguistics*. Cascadilla Proceedings Project Somerville, MA, pages 162–170.

Carol Myers-Scotton. 1995. *Social motivations for codeswitching: Evidence from Africa*. Oxford University Press.

David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. 2010. Automatic evaluation of topic coherence. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 100–108.

Wijnanda M Poulisse. 1989. *The use of compensatory strategies by Dutch learners of English*. [Sl: sn].

Ella Rabinovich, Yulia Tsvetkov, and Shuly Wintner. 2018. Native language cognate effects on second language lexical choice. *Transactions of the Association of Computational Linguistics* 6:329–342.

Sudha Rao and Joel Tetreault. 2018. Dear Sir or Madam, May I Introduce the GYAFC Dataset: Corpus, benchmarks and metrics for formality style transfer. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. pages 129–140.

Shruti Rijhwani, Royal Sequiera, Monojit Choudhury, Kalika Bali, and Chandra Shekhar Maddila. 2017. Estimating code-switching on twitter with a novel generalized word-level language detection technique. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. pages 1971–1982.

Koustav Rudra, Shruti Rijhwani, Rafiya Begum, Kalika Bali, Monojit Choudhury, and Niloy Ganguly. 2016. Understanding language preference for expression of opinion and sentiment: What do Hindi–English speakers do on Twitter? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 1131–1141.

Mark Sebba. 2013. Multilingualism in written discourse: An approach to the analysis of multilingual texts. *International Journal of Bilingualism* 17(1):97–118.

Mark Sebba, Shahrzad Mahootian, and Carla Jonsson. 2012. *Language mixing and code-switching in writing: Approaches to mixed-language written discourse*. Routledge.

Thamar Solorio, Elizabeth Blair, Suraj Maharjan, Steven Bethard, Mona Diab, Mahmoud Ghoneim, Abdelati Hawwari, Fahad AlGhamdi, Julia Hirschberg, Alison Chang, et al. 2014. Overview for the first shared task on language identification in code-switched data. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*. pages 62–72.

Thamar Solorio and Yang Liu. 2008. Learning to predict code-switching points. In *Conference on Empirical Methods in Natural Language Processing*.

Victor Soto, Nishmar Cestero, and Julia Hirschberg. 2018. The role of cognate words, POS tags, and entrainment in code-switching. *Proc. Interspeech 2018* pages 1938–1942.

Emre Yılmaz, Astik Biswas, Ewald van der Westhuizen, Febe de Wet, and Thomas Niesler. 2018. Building a unified code-switching ASR system for South African languages. *arXiv preprint arXiv:1807.10949* .

# Modeling Conversation Structure and Temporal Dynamics for Jointly Predicting Rumor Stance and Veracity

**Penghui Wei, Nan Xu, Wenji Mao**
†SKL-MCCS, Institute of Automation, Chinese Academy of Sciences (CASIA)
‡University of Chinese Academy of Sciences
{weipenghui2016,xunan2015,wenji.mao}@ia.ac.cn

## Abstract

Automatically verifying rumorous information has become an important and challenging task in natural language processing and social media analytics. Previous studies reveal that people's stances towards rumorous messages can provide indicative clues for identifying the veracity of rumors, and thus determining the stances of public reactions is a crucial preceding step for rumor veracity prediction. In this paper, we propose a hierarchical multi-task learning framework for jointly predicting rumor stance and veracity on Twitter, which consists of two components. The bottom component of our framework classifies the stances of tweets in a conversation discussing a rumor via modeling the structural property based on a novel graph convolutional network. The top component predicts the rumor veracity by exploiting the temporal dynamics of stance evolution. Experimental results on two benchmark datasets show that our method outperforms previous methods in both rumor stance classification and veracity prediction.

## 1 Introduction

Social media websites have become the main platform for users to browse information and share opinions, facilitating news dissemination greatly. However, the characteristics of social media also accelerate the rapid spread and dissemination of unverified information, i.e., rumors (Shu et al., 2017). The definition of *rumor* is "*items of information that are unverified at the time of posting*" (Zubiaga et al., 2018). Ubiquitous false rumors bring about harmful effects, which has seriously affected public and individual lives, and caused panic in society (Zhou and Zafarani, 2018; Sharma et al., 2019). Because online content is massive and debunking rumors manually is time-consuming, there is a great need for automatic methods to identify false rumors (Oshikawa et al., 2018).

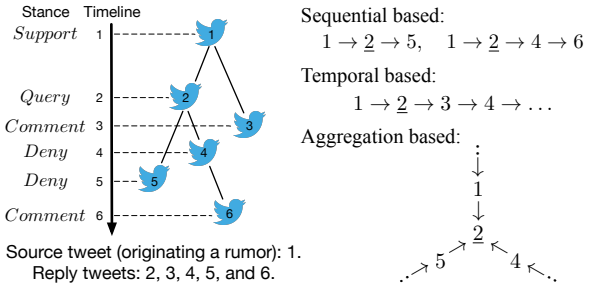


Figure 1: A conversation thread discussing the rumorous tweet "1". Three different perspectives for learning the stance feature of the reply tweet "2" are illustrated.

Previous studies have observed that public stances towards rumorous messages are crucial signals to detect trending rumors (Qazvinian et al., 2011; Zhao et al., 2015) and indicate the veracity of them (Mendoza et al., 2010; Procter et al., 2013; Liu et al., 2015; Jin et al., 2016; Glenski et al., 2018). Therefore, stance classification towards rumors is viewed as an important preceding step of rumor veracity prediction, especially in the context of Twitter conversations (Zubiaga et al., 2016a).

The state-of-the-art methods for rumor stance classification are proposed to model the sequential property (Kochkina et al., 2017) or the temporal property (Veyseh et al., 2017) of a Twitter conversation thread. In this paper, we propose a new perspective based on structural property: learning tweet representations through aggregating information from their neighboring tweets. Intuitively, a tweet's nearer neighbors in its conversation thread are more informative than farther neighbors because the replying relationships of them are closer, and their stance expressions can help classify the stance of the center tweet (e.g., in Figure 1, tweets "1", "4" and "5" are the one-hop neighbors of the tweet "2", and their influences on predicting the stance of "2" are larger than that of the two-hop neighbor "3"). To achieve this, we represent both



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4787–4798,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

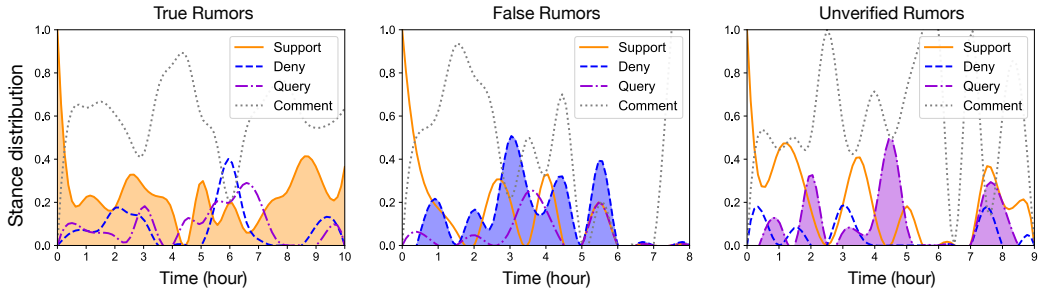


Figure 2: Stance distributions of tweets discussing $true$ rumors, $false$ rumors, and $unverified$ rumors, respectively (Better viewed in color). The horizontal axis is the spreading time of the first rumor. It is visualized based on SemEval-2017 task 8 dataset (Derczynski et al., 2017). All tweets are relevant to the event "Ottawa Shooting".

tweet contents and conversation structures into a latent space using a graph convolutional network (GCN) (Kipf and Welling, 2017), aiming to learn stance feature for each tweet by aggregating its neighbors' features. Compared with the sequential and temporal based methods, our aggregation based method leverages the intrinsic structural property in conversations to learn tweet representations.

After determining the stances of people's reactions, another challenge is how we can utilize public stances to predict rumor veracity accurately. We observe that the temporal dynamics of public stances can indicate rumor veracity. Figure 2 illustrates the stance distributions of tweets discussing $true$ rumors, $false$ rumors, and $unverified$ rumors, respectively. As we can see, $supporting$ stance dominates the inception phase of spreading. However, as time goes by, the proportion of $denying$ tweets towards $false$ rumors increases quite significantly. Meanwhile, the proportion of $querying$ tweets towards $unverified$ rumors also shows an upward trend. Based on this observation, we propose to model the temporal dynamics of stance evolution with a recurrent neural network (RNN), capturing the crucial signals containing in stance features for effective veracity prediction.

Further, most existing methods tackle stance classification and veracity prediction separately, which is suboptimal and limits the generalization of models. As shown previously, they are two closely related tasks in which stance classification can provide indicative clues to facilitate veracity prediction. Thus, these two tasks can be jointly learned to make better use of their interrelation.

Based on the above considerations, in this paper, we propose a hierarchical multi-task learning framework for jointly predicting rumor stance and veracity, which achieves deep integration between the preceding task (stance classification) and the subsequent task (veracity prediction). The bottom component of our framework classifies the stances of tweets in a conversation discussing a rumor via aggregation-based structure modeling, and we design a novel graph convolution operation customized for conversation structures. The top component predicts rumor veracity by exploiting the temporal dynamics of stance evolution, taking both content features and stance features learned by the bottom component into account. Two components are jointly trained to utilize the interrelation between the two tasks for learning more powerful feature representations.

The contributions of this work are as follows.

- We propose a hierarchical framework to tackle rumor stance classification and veracity prediction jointly, exploiting both structural characteristic and temporal dynamics in rumor spreading process.
- We design a novel graph convolution operation customized to encode conversation structures for learning stance features. To our knowledge, we are the first to employ graph convolution for modeling the structural property of Twitter conversations.
- Experimental results on two benchmark datasets verify that our hierarchical framework performs better than existing methods in both rumor stance classification and veracity prediction.

## 2 Related Work

**Rumor Stance Classification** Stance analysis has been widely studied in online debate forums (Somasundaran and Wiebe, 2009; Hasan and Ng, 2013), and recently has attracted increasing attention in different contexts (Mohammad et al., 2016; Augenstein et al., 2016; Ferreira and Vlachos, 2016; Mohtarami et al., 2018). After the pioneering studies on stance classification towards rumors in social media (Mendoza et al., 2010;

Qazvinian et al., 2011; Procter et al., 2013), linguistic feature (Hamidian and Diab, 2016; Zeng et al., 2016) and point process based methods (Lukasik et al., 2016, 2019) have been developed.

Recent work has focused on Twitter conversations discussing rumors. Zubiaga et al. (2016a) proposed to capture the sequential property of conversations with linear-chain CRF, and also used a tree-structured CRF to consider the conversation structure as a whole. Aker et al. (2017) developed a novel feature set that scores the level of users' confidence. Pamungkas et al. (2018) designed affective and dialogue-act features to cover various facets of affect. Giasemidis et al. (2018) proposed a semi-supervised method that propagates the stance labels on similarity graph. Beyond feature-based methods, Kochkina et al. (2017) utilized an LSTM to model the sequential branches in a conversation, and their system ranked the first in SemEval-2017 task 8. Veyseh et al. (2017) adopted attention to model the temporal property of a conversation and achieved the state-of-the-art performance.

**Rumor Veracity Prediction** Previous studies have proposed methods based on various features such as linguistics, time series and propagation structures (Castillo et al., 2011; Kwon et al., 2013; Wu et al., 2015; Ma et al., 2017). Neural networks show the effectiveness of modeling time series (Ma et al., 2016; Ruchansky et al., 2017) and propagation paths (Liu and Wu, 2018). Ma et al. (2018b)'s model adopted recursive neural networks to incorporate structure information into tweet representations and outperformed previous methods.

Some studies utilized stance labels as the input feature of veracity classifiers to improve the performance (Liu et al., 2015; Enayet and El-Beltagy, 2017). Dungs et al. (2018) proposed to recognize the temporal patterns of true and false rumors' stances by two hidden Markov models (HMMs). Unlike their solution, our method learns discriminative features of stance evolution with an RNN. Moreover, our method jointly predicts stance and veracity by exploiting both structural and temporal characteristics, whereas HMMs need stance labels as the input sequence of observations.

**Joint Predictions of Rumor Stance and Veracity** Several work has addressed the problem of jointly predicting rumor stance and veracity. These studies adopted multi-task learning to jointly train two tasks (Ma et al., 2018a; Kochkina et al., 2018; Poddar et al., 2018) and learned shared representations with parameter-sharing. Compared with such solutions based on "parallel" architectures, our method is deployed in a hierarchical fashion that encodes conversation structures to learn more powerful stance features by the bottom component, and models stance evolution by the top component, achieving deep integration between the two tasks' feature learning.

## 3 Problem Definition

Consider a Twitter conversation thread $\mathcal{C}$ which consists of a source tweet $t_1$ (originating a rumor) and a number of reply tweets $\{t_2, t_3, \ldots, t_{|\mathcal{C}|}\}$ that respond $t_1$ directly or indirectly, and each tweet $t_i$ ($i \in [1, |\mathcal{C}|]$) expresses its stance towards the rumor. The thread $\mathcal{C}$ is a tree structure, in which the source tweet $t_1$ is the root node, and the replying relationships among tweets form the edges.[1]

This paper focuses on two tasks. The first task is rumor stance classification, aiming to determine the stance of each tweet in $\mathcal{C}$, which belongs to $\{supporting, denying, querying, commenting\}$. The second task is rumor veracity prediction, with the aim of identifying the veracity of the rumor, belonging to $\{true, false, unverified\}$.

## 4 Proposed Method

We propose a Hierarchical multi-task learning framework for jointly Predicting rumor Stance and Veracity (named *Hierarchical-PSV*). Figure 3 illustrates its overall architecture that is composed of two components. The bottom component is to classify the stances of tweets in a conversation thread, which learns stance features via encoding conversation structure using a customized graph convolutional network (named *Conversational-GCN*). The top component is to predict the rumor's veracity, which takes the learned features from the bottom component into account and models the temporal dynamics of stance evolution with a recurrent neural network (named *Stance-Aware RNN*).

### 4.1 Conversational-GCN: Aggregation-based Structure Modeling for Stance Prediction

Now we detail *Conversational-GCN*, the bottom component of our framework. We first adopt a bidirectional GRU (BGRU) (Cho et al., 2014) layer to learn the content feature for each tweet in the thread $\mathcal{C}$. For a tweet $t_i$ ($i \in [1, |\mathcal{C}|]$), we run the

[1] We consider each replying relationship between two tweets in a conversation as an undirected edge in the tree.

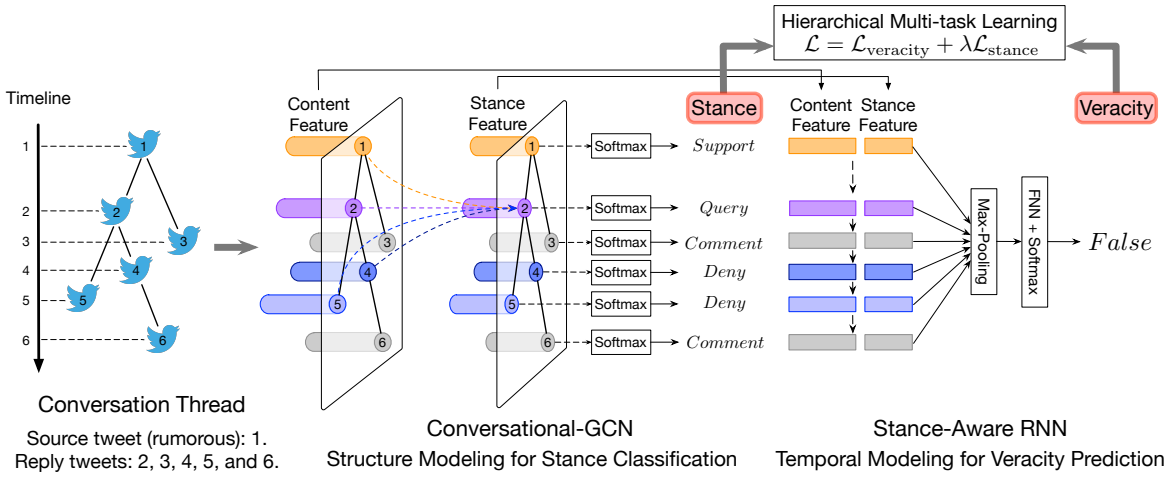


Figure 3: Overall architecture of our proposed framework for joint predictions of rumor stance and veracity. In this illustration, the number of GCN layers is one. The information aggregation process for the tweet $t_2$ based on original graph convolution operation (Eq. (1)) is detailed.

BGRU over its word embedding sequence, and use the final step's hidden vector to represent the tweet. The content feature representation of $t_i$ is denoted as $\boldsymbol{c}_i \in \mathbb{R}^d$, where $d$ is the output size of the BGRU.

As we mentioned in Section 1, the stance expressions of a tweet $t_i$'s nearer neighbors can provide more informative signals than farther neighbors for learning $t_i$'s stance feature. Based on the above intuition, we model the structural property of the conversation thread $\mathcal{C}$ to learn stance feature representation for each tweet in $\mathcal{C}$. To this end, we encode structural contexts to improve tweet representations by aggregating information from neighboring tweets with a graph convolutional network (GCN) (Kipf and Welling, 2017).

Formally, the conversation $\mathcal{C}$'s structure can be represented by a graph $\mathcal{C}_G = \langle \mathcal{T}, \mathcal{E} \rangle$, where $\mathcal{T} = \{t_i\}_{i=1}^{|\mathcal{C}|}$ denotes the node set (i.e., tweets in the conversation), and $\mathcal{E}$ denotes the edge set composed of all replying relationships among the tweets. We transform the edge set $\mathcal{E}$ to an adjacency matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{C}| \times |\mathcal{C}|}$, where $\mathbf{A}_{ij} = \mathbf{A}_{ji} = 1$ if the tweet $t_i$ directly replies the tweet $t_j$ or $i = j$. In one GCN layer, the graph convolution operation for one tweet $t_i$ on $\mathcal{C}_G$ is defined as:

$$\boldsymbol{h}_i^{\text{out}} = \tanh\left( \sum_{j \in \{j | \hat{\mathbf{A}}_{ij} \neq 0\}} \hat{\mathbf{A}}_{ij} \boldsymbol{W}^\top \boldsymbol{h}_j^{\text{in}} + \boldsymbol{b} \right), \tag{1}$$

where $\boldsymbol{h}_i^{\text{in}} \in \mathbb{R}^{d_{\text{in}}}$ and $\boldsymbol{h}_i^{\text{out}} \in \mathbb{R}^{d_{\text{out}}}$ denote the input and output feature representations of the tweet $t_i$ respectively. The convolution filter $\boldsymbol{W} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$ and the bias $\boldsymbol{b} \in \mathbb{R}^{d_{\text{out}}}$ are shared over all tweets in a conversation. We apply symmetric normalized transformation $\hat{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}$ to avoid the scale changing of feature representations, where $\mathbf{D}$ is the degree matrix of $\mathbf{A}$, and $\{j \mid \hat{\mathbf{A}}_{ij} \neq 0\}$ contains $t_i$'s one-hop neighbors and $t_i$ itself.

In this original graph convolution operation, given a tweet $t_i$, the receptive field for $t_i$ contains its one-hop neighbors and $t_i$ itself, and the aggregation level of two tweets $t_i$ and $t_j$ is dependent on $\hat{\mathbf{A}}_{ij}$. In the context of encoding conversation structures, we observe that such operation can be further improved for two issues. First, a tree-structured conversation may be very deep, which means that the receptive field of a GCN layer is restricted in our case. Although we can stack multiple GCN layers to expand the receptive field, it is still difficult to handle conversations with deep structures and increases the number of parameters. Second, the normalized matrix $\hat{\mathbf{A}}$ partly weakens the importance of the tweet $t_i$ itself. To address these issues, we design a novel graph convolution operation which is customized to encode conversation structures. Formally, it is implemented by modifying the matrix $\hat{\mathbf{A}}$ in Eq. (1):

$$\hat{\mathbf{A}} \leftarrow \hat{\mathbf{A}}\hat{\mathbf{A}} + \mathbf{I}, \tag{2}$$

where the multiplication operation expands the receptive field of a GCN layer, and adding an identity matrix elevates the importance of $t_i$ itself.

After defining the above graph convolution operation, we adopt an $L$-layer GCN to model conversation structures. The $l^{\text{th}}$ GCN layer ($l \in [1, L]$) computed over the entire conversation structure can

be written as an efficient matrix operation:

$$\boldsymbol{H}^{(l)} = \tanh(\hat{\mathbf{A}}\boldsymbol{H}^{(l-1)}\boldsymbol{W}^{(l)} + \boldsymbol{b}^{(l)}), \quad (3)$$

where $\boldsymbol{H}^{(l-1)} \in \mathbb{R}^{|\mathcal{C}|\times d_{l-1}}$ and $\boldsymbol{H}^{(l)} \in \mathbb{R}^{|\mathcal{C}|\times d_l}$ denote the input and output features of all tweets in the conversation $\mathcal{C}$ respectively.

Specifically, the first GCN layer takes the content features of all tweets as input, i.e., $\boldsymbol{H}^{(0)} = (\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_{|\mathcal{C}|})^\top \in \mathbb{R}^{|\mathcal{C}|\times d}$. The output of the last GCN layer represents the stance features of all tweets in the conversation, i.e., $\boldsymbol{H}^{(L)} = (\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_{|\mathcal{C}|})^\top \in \mathbb{R}^{|\mathcal{C}|\times 4}$, where $\boldsymbol{s}_i$ is the unnormalized stance distribution of the tweet $t_i$.

For each tweet $t_i$ in the conversation $\mathcal{C}$, we apply softmax to obtain its predicted stance distribution:

$$\hat{\boldsymbol{s}}_i = \mathrm{softmax}(\boldsymbol{s}_i) \in \mathbb{R}^4, \quad i \in [1, |\mathcal{C}|]. \quad (4)$$

The ground-truth labels of stance classification supervise the learning process of *Conversational-GCN*. The loss function of $\mathcal{C}$ for stance classification is computed by cross-entropy criterion:

$$\mathcal{L}_{\text{stance}} = \frac{1}{|\mathcal{C}|}\sum_{i=1}^{|\mathcal{C}|}\left(-s_i^\top \log \hat{\boldsymbol{s}}_i\right), \quad (5)$$

where $s_i$ is a one-hot vector that denotes the stance label of the tweet $t_i$. For batch-wise training, the objective function for a batch is the averaged cross-entropy loss of all tweets in these conversations.

In previous studies, GCNs are used to encode dependency trees (Marcheggiani and Titov, 2017; Zhang et al., 2018) and cross-document relations (Yasunaga et al., 2017; De Cao et al., 2019) for downstream tasks. Our work is the first to leverage GCNs for encoding conversation structures.

### 4.2 Stance-Aware RNN: Temporal Dynamics Modeling for Veracity Prediction

The top component, *Stance-Aware RNN*, aims to capture the temporal dynamics of stance evolution in a conversation discussing a rumor. It integrates both content features and stance features learned from the bottom *Conversational-GCN* to facilitate the veracity prediction of the rumor.

Specifically, given a conversation thread $\mathcal{C} = \{t_1, t_2, \ldots, t_{|\mathcal{C}|}\}$ (where the tweets $t_*$ are ordered chronologically), we combine the content feature and the stance feature for each tweet, and adopt a GRU layer to model the temporal evolution:

$$\boldsymbol{v}_i = \mathrm{GRU}([\boldsymbol{c}_i; \boldsymbol{s}_i], \boldsymbol{v}_{i-1}), \quad i \in [1, |\mathcal{C}|], \quad (6)$$

where $[\cdot;\cdot]$ denotes vector concatenation, and $(\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_{|\mathcal{C}|})$ is the output sequence that represents the temporal feature. We then transform the sequence to a vector $\boldsymbol{v}$ by a max-pooling function that captures the global information of stance evolution, and feed it into a one-layer feed-forward neural network (FNN) with softmax normalization to produce the predicted veracity distribution $\hat{\boldsymbol{v}}$:

$$\begin{aligned} \boldsymbol{v} &= \text{max-pooling}(\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_{|\mathcal{C}|}), \\ \hat{\boldsymbol{v}} &= \mathrm{softmax}(\mathrm{FNN}(\boldsymbol{v})). \end{aligned} \quad (7)$$

The loss function of $\mathcal{C}$ for veracity prediction is also computed by cross-entropy criterion:

$$\mathcal{L}_{\text{veracity}} = -v^\top \log \hat{\boldsymbol{v}}, \quad (8)$$

where $v$ denotes the veracity label of $\mathcal{C}$.

### 4.3 Jointly Learning Two Tasks

To leverage the interrelation between the preceding task (stance classification) and the subsequent task (veracity prediction), we jointly train two components in our framework. Specifically, we add two tasks' loss functions to obtain a joint loss function $\mathcal{L}$ (with a trade-off parameter $\lambda$), and optimize $\mathcal{L}$ to train our framework:

$$\mathcal{L} = \mathcal{L}_{\text{veracity}} + \lambda\mathcal{L}_{\text{stance}}. \quad (9)$$

In our *Hierarchical-PSV*, the bottom component *Conversational-GCN* learns content and stance features, and the top component *Stance-Aware RNN* takes the learned features as input to further exploit temporal evolution for predicting rumor veracity. Our multi-task framework achieves deep integration of the feature representation learning process for the two closely related tasks.

## 5 Experiments

In this section, we first evaluate the performance of *Conversational-GCN* on rumor stance classification and evaluate *Hierarchical-PSV* on veracity prediction (Section 5.3). We then give a detailed analysis of our proposed method (Section 5.4).

### 5.1 Data & Evaluation Metric

To evaluate our proposed method, we conduct experiments on two benchmark datasets.

The first is **SemEval-2017 task 8** (Derczynski et al., 2017) dataset. It includes 325 rumorous conversation threads, and has been split into training, development and test sets. These threads cover ten

| Dataset | # Thread | Depth | # Tweet | Stance Labels | | | | Veracity Labels | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | # *support* | # *deny* | # *query* | # *comment* | # *true* | # *false* | # *unverified* |
| SemEval | 325 | 3.5 | 5,568 | 1,004 | 415 | 464 | 3,685 | 145 | 74 | 106 |
| PHEME | 2,402 | 2.8 | 105,354 | – | | | | 1,067 | 638 | 697 |

Table 1: Statistics of two datasets. The column "Depth" denotes the average depth of all conversation threads.

| Method | Evaluation Metric | | | | | |
|---|---|---|---|---|---|---|
| | Macro-$F_1$ | $F_S$ | $F_D$ | $F_Q$ | $F_C$ | Acc. |
| Affective Feature + SVM (Pamungkas et al., 2018) | 0.470 | **0.410** | 0.000 | 0.580 | **0.880** | 0.795 |
| BranchLSTM (Kochkina et al., 2017) | 0.434 | 0.403 | 0.000 | 0.462 | 0.873 | 0.784 |
| TemporalAttention (Veyseh et al., 2017) | 0.482 | – | – | – | – | **0.820** |
| Conversational-GCN (Ours, $L = 2$) | **0.499** | 0.311 | **0.194** | **0.646** | 0.847 | 0.751 |

Table 2: Results of rumor stance classification. $F_S$, $F_D$, $F_Q$ and $F_C$ denote the $F_1$ scores of *supporting*, *denying*, *querying* and *commenting* classes respectively. "–" indicates that the original paper does not report the metric.

events, and two events of that only appear in the test set. This dataset is used to evaluate both stance classification and veracity prediction tasks.

The second is **PHEME** dataset (Zubiaga et al., 2016b). It provides 2,402 conversations covering nine events. Following previous work, we conduct leave-one-event-out cross-validation: in each fold, one event's conversations are used for testing, and all the rest events are used for training. The evaluation metric on this dataset is computed after integrating the outputs of all nine folds. Note that only a subset of this dataset has stance labels, and all conversations in this subset are already contained in SemEval-2017 task 8 dataset. Thus, PHEME dataset is used to evaluate veracity prediction task.

Table 1 shows the statistics of two datasets. Because of the class-imbalanced problem, we use macro-averaged $F_1$ as the evaluation metric for two tasks. We also report accuracy for reference.

### 5.2 Implementation Details

In all experiments, the number of GCN layers is set to $L = 2$. We list the implementation details in Appendix A.

### 5.3 Experimental Results

#### 5.3.1 Results: Rumor Stance Classification

**Baselines** We compare our *Conversational-GCN* with the following methods in the literature:

• *Affective Feature + SVM* (Pamungkas et al., 2018) extracts affective and dialogue-act features for individual tweets, and then trains an SVM for classifying stances.

• *BranchLSTM* (Kochkina et al., 2017) is the winner of SemEval-2017 shared task 8 subtask A. It adopts an LSTM to model the sequential branches in a conversation thread. Before feeding branches into the LSTM, some additional hand-crafted features are used to enrich the tweet representations.

• *TemporalAttention* (Veyseh et al., 2017) is the state-of-the-art method. It uses a tweet's "neighbors in the conversation timeline" as the context, and utilizes attention to model such temporal sequence for learning the weight of each neighbor. Extra hand-crafted features are also used.

**Performance Comparison** Table 2 shows the results of different methods for rumor stance classification. Clearly, the macro-averaged $F_1$ of Conversational-GCN is better than all baselines.

Especially, our method shows the effectiveness of determining *denying* stance, while other methods can not give any correct prediction for *denying* class (the $F_D$ scores of them are equal to zero). Further, Conversational-GCN also achieves higher $F_1$ score for *querying* stance ($F_Q$). Identifying *denying* and *querying* stances correctly is crucial for veracity prediction because they play the role of indicators for $false$ and $unverified$ rumors respectively (see Figure 2). Meanwhile, the class-imbalanced problem of data makes this a challenge. Conversational-GCN effectively encodes structural context for each tweet via aggregating information from its neighbors, learning powerful stance features without feature engineering. It is also more computationally efficient than sequential and temporal based methods. The information aggregations for all tweets in a conversation are worked in parallel and thus the running time is not sensitive to conversation's depth.

| Setting | Method | SemEval dataset | | PHEME dataset | |
|---|---|---|---|---|---|
| | | Macro-$F_1$ | Acc. | Macro-$F_1$ | Acc. |
| Single-task | TD-RvNN (Ma et al., 2018b) | 0.509 | 0.536 | 0.264 | 0.341 |
| | Hierarchical GCN-RNN (Ours) | 0.540 | 0.536 | 0.317 | 0.356 |
| Multi-task | BranchLSTM+NileTMRG (Kochkina et al., 2018) | 0.539 | 0.570 | 0.297 | 0.360 |
| | MTL2 (Veracity+Stance) (Kochkina et al., 2018) | 0.558 | 0.571 | 0.318 | 0.357 |
| | Hierarchical-PSV (Ours, $\lambda = 1$) | **0.588** | **0.643** | **0.333** | **0.361** |

Table 3: Results of veracity prediction. Single-task setting means that stance labels cannot be used to train models.

#### 5.3.2 Results: Rumor Veracity Prediction

To evaluate our framework *Hierarchical-PSV*, we consider two groups of baselines: single-task and multi-task baselines.

**Single-task Baselines** In single-task setting, stance labels are not available. Only veracity labels can be used to supervise the training process.

• *TD-RvNN* (Ma et al., 2018b) models the top-down tree structure using a recursive neural network for veracity classification.

• *Hierarchical GCN-RNN* is the single-task variant of our framework: we optimize $\mathcal{L}_{\text{veracity}}$ (i.e., $\lambda = 0$ in Eq. (9)) during training. Thus, the bottom Conversational-GCN only has indirect supervision (veracity labels) to learn stance features.

**Multi-task Baselines** In multi-task setting, both stance labels and veracity labels are available for training.

• *BranchLSTM+NileTMRG* (Kochkina et al., 2018) is a pipeline method, combining the winner systems of two subtasks in SemEval-2017 shared task 8. It first trains a BranchLSTM for stance classification, and then uses the predicted stance labels as extra features to train an SVM for veracity prediction (Enayet and El-Beltagy, 2017).

• *MTL2 (Veracity+Stance)* (Kochkina et al., 2018) is a multi-task learning method that adopts BranchLSTM as the shared block across tasks. Then, each task has a task-specific output layer, and two tasks are jointly learned.[2]

**Performance Comparison** Table 3 shows the comparisons of different methods. By comparing single-task methods, Hierarchical GCN-RNN performs better than TD-RvNN, which indicates that our hierarchical framework can effectively model conversation structures to learn high-quality tweet representations. The recursive operation in TD-RvNN is performed in a fixed direction and runs over all tweets, thus may not obtain enough useful information. Moreover, the training speed of Hierarchical GCN-RNN is significantly faster than TD-RvNN: in the condition of batch-wise optimization for training one step over a batch containing 32 conversations, our method takes only 0.18 seconds, while TD-RvNN takes 5.02 seconds.

Comparisons among multi-task methods show that two joint methods outperform the pipeline method (BranchLSTM+NileTMRG), indicating that jointly learning two tasks can improve the generalization through leveraging the interrelation between them. Further, compared with MTL2 which uses a "parallel" architecture to make predictions for two tasks, our Hierarchical-PSV performs better than MTL2. The hierarchical architecture is more effective to tackle the joint predictions of rumor stance and veracity, because it not only possesses the advantage of parameter-sharing but also offers deep integration of the feature representation learning process for the two tasks. Compared with Hierarchical GCN-RNN that does not use the supervision from stance classification task, Hierarchical-PSV provides a performance boost, which demonstrates that our framework benefits from the joint learning scheme.

### 5.4 Further Analysis and Discussions

We conduct additional experiments to further demonstrate the effectiveness of our model.

#### 5.4.1 Effect of Customized Graph Convolution

To show the effect of our customized graph convolution operation (Eq. (2)) for modeling conversation structures, we further compare it with the original graph convolution (Eq. (1), named Original-GCN) on stance classification task.

Specifically, we cluster tweets in the test set according to their depths in the conversation threads (e.g., the cluster "depth = 0" consists of all source

[2] Kochkina et al. (2018) also proposed *MTL3* that jointly trains three tasks (plus rumor detection (Zubiaga et al., 2017)). In our framework, we do not utilize data and labels from rumor detection task, and thus we choose *MTL2 (Veracity+Stance)* as the state-of-the-art method for comparison.

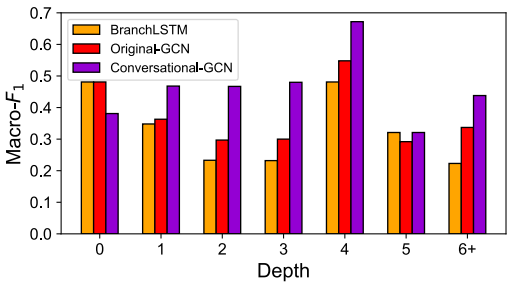


Figure 4: Stance classification results w.r.t. different depths (see Appendix B for exact numerical numbers).

| **Method** | Macro-$F_1$ | Acc. |
|---|---|---|
| Hierarchical-PSV (full model) | **0.333** | **0.361** |
| – stance features | 0.299 | 0.338 |
| – GRU, + CNN | 0.312 | 0.328 |
| – GRU | 0.288 | 0.326 |

Table 4: Ablation tests of stance features and temporal modeling for veracity prediction on PHEME dataset.

tweets in the test set). For BranchLSTM, Original-GCN and Conversational-GCN, we report their macro-averaged $F_1$ on each cluster in Figure 4.

We observe that our Conversational-GCN outperforms Original-GCN and BranchLSTM significantly in most levels of depth. BranchLSTM may prefer to "shallow" tweets in a conversation because they often occur in multiple branches (e.g., in Figure 1, the tweet "2" occurs in two branches and thus it will be modeled twice). The results indicate that Conversational-GCN has advantage to identify stances of "deep" tweets in conversations.

### 5.4.2 Ablation Tests

**Effect of Stance Features** To understand the importance of stance features for veracity prediction, we conduct an ablation study: we only input the content features of all tweets in a conversation to the top component RNN. It means that the RNN only models the temporal variation of tweet contents during spreading, but does not consider their stances and is not "stance-aware". Table 4 shows that "– stance features" performs poorly, and thus the temporal modeling process benefits from the indicative signals provided by stance features. Hence, combining the low-level content features and the high-level stance features is crucial to improve rumor veracity prediction.

**Effect of Temporal Evolution Modeling** We modify the Stance-Aware RNN by two ways: (i) we replace the GRU layer by a CNN that only captures local temporal information; (ii) we remove the GRU layer. Results in Table 4 verify that replacing or removing the GRU block hurts the performance, and thus modeling the stance evolution of public reactions towards a rumorous message is indeed necessary for effective veracity prediction.

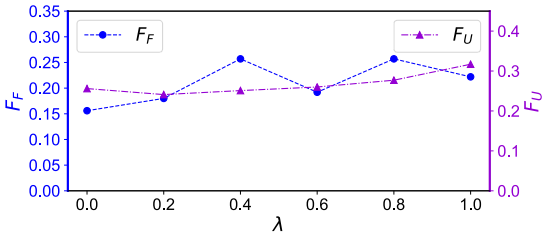


Figure 5: Veracity prediction results v.s. various values of $\lambda$ on PHEME dataset. $F_{\text{F}}$ and $F_{\text{U}}$ denote the $F_1$ scores of $false$ and $unverified$ classes respectively.

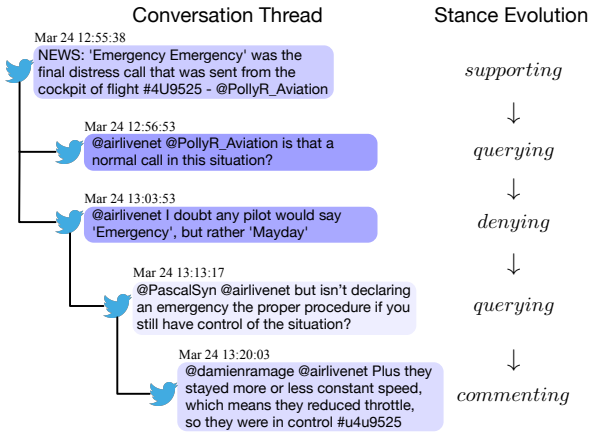


Figure 6: Case study: a $false$ rumor. Each tweet is colored by the number of dimensions it contributes to $v$ in the max-pooling operation (Eq. (7)). We show important tweets in the conversation and truncate others.

### 5.4.3 Interrelation of Stance and Veracity

We vary the value of $\lambda$ in the joint loss $\mathcal{L}$ and train models with various $\lambda$ to show the interrelation between stance and veracity in Figure 5. As $\lambda$ increases from 0.0 to 1.0, the performance of identifying $false$ and $unverified$ rumors generally gains. Therefore, when the supervision signal of stance classification becomes strong, the learned stance features can produce more accurate clues for predicting rumor veracity.

## 5.5 Case Study

Figure 6 illustrates a $false$ rumor identified by our model. We can observe that the stances of reply tweets present a typical temporal pattern "$supporting \rightarrow querying \rightarrow denying$". Our

model captures such stance evolution with RNN and predicts its veracity correctly. Further, the visualization of tweets shows that the max-pooling operation catches informative tweets in the conversation. Hence, our framework can notice salience indicators of rumor veracity in the spreading process and combine them to give correct prediction.

## 6 Conclusion

We propose a hierarchical multi-task learning framework for jointly predicting rumor stance and veracity on Twitter. We design a new graph convolution operation, Conversational-GCN, to encode conversation structures for classifying stance, and then the top Stance-Aware RNN combines the learned features to model the temporal dynamics of stance evolution for veracity prediction. Experimental results verify that Conversational-GCN can handle deep conversation structures effectively, and our hierarchical framework performs much better than existing methods. In future work, we shall explore to incorporate external context (Derczynski et al., 2017; Popat et al., 2018), and extend our model to multi-lingual scenarios (Wen et al., 2018). Moreover, we shall investigate the diffusion process of rumors from social science perspective (Vosoughi et al., 2018), draw deeper insights from there and try to incorporate them into the model design.

## Acknowledgments

This work was supported in part by the National Key R&D Program of China under Grant #2016QY02D0305, NSFC Grants #71621002, #71472175, #71974187 and #71602184, and Ministry of Health of China under Grant #2017ZX10303401-002. We thank all the anonymous reviewers for their valuable comments. We also thank Qianqian Dong for her kind assistance.

## References

Ahmet Aker, Leon Derczynski, and Kalina Bontcheva. 2017. Simple open stance classification for rumour analysis. In *Proceedings of RANLP*, pages 31–39.

Isabelle Augenstein, Tim Rocktäschel, Andreas Vlachos, and Kalina Bontcheva. 2016. Stance detection with bidirectional conditional encoding. In *Proceedings of EMNLP*, pages 876–885.

Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. 2011. Information credibility on Twitter. In *Proceedings of WWW*, pages 675–684.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of EMNLP*, pages 1724–1734.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2019. Question answering by reasoning across documents with graph convolutional networks. In *Proceedings of NAACL-HLT*, pages 2306–2317.

Leon Derczynski, Kalina Bontcheva, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Arkaitz Zubiaga. 2017. SemEval-2017 task 8: RumourEval: Determining rumour veracity and support for rumours. In *Proceedings of SemEval*, pages 69–76.

Sebastian Dungs, Ahmet Aker, Norbert Fuhr, and Kalina Bontcheva. 2018. Can rumour stance alone predict veracity? In *Proceedings of COLING*, pages 3360–3370.

Omar Enayet and Samhaa R El-Beltagy. 2017. NileTMRG at SemEval-2017 task 8: Determining rumour and veracity support for rumours on Twitter. In *Proceedings of SemEval*, pages 470–474.

William Ferreira and Andreas Vlachos. 2016. Emergent: A novel data-set for stance classification. In *Proceedings of NAACL-HLT*, pages 1163–1168.

Georgios Giasemidis, Nikolaos Kaplis, Ioannis Agrafiotis, and Jason Nurse. 2018. A semi-supervised approach to message stance classification. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*.

Maria Glenski, Tim Weninger, and Svitlana Volkova. 2018. Identifying and understanding user reactions to deceptive and trusted social news sources. In *Proceedings of ACL*, pages 176–181.

Sardar Hamidian and Mona Diab. 2016. Rumor identification and belief investigation on Twitter. In *Proceedings of NAACL-HLT*, pages 3–8.

Kazi Saidul Hasan and Vincent Ng. 2013. Extra-linguistic constraints on stance recognition in ideological debates. In *Proceedings of ACL*, pages 816–821.

Zhiwei Jin, Juan Cao, Yongdong Zhang, and Jiebo Luo. 2016. News verification by exploiting conflicting social viewpoints in microblogs. In *Proceedings of AAAI*, pages 2972–2978.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of EMNLP*, pages 1746–1751.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of ICLR*.

Elena Kochkina, Maria Liakata, and Isabelle Augenstein. 2017. Turing at SemEval-2017 task 8: Sequential approach to rumour stance classification with branch-LSTM. In *Proceedings of SemEval*, pages 475–480.

Elena Kochkina, Maria Liakata, and Arkaitz Zubiaga. 2018. All-in-one: Multi-task learning for rumour verification. In *Proceedings of COLING*, pages 3402–3413.

Sejeong Kwon, Meeyoung Cha, Kyomin Jung, Wei Chen, and Yajun Wang. 2013. Prominent features of rumor propagation in online social media. In *Proceedings of ICDM*, pages 1103–1108.

Xiaomo Liu, Armineh Nourbakhsh, Quanzhi Li, Rui Fang, and Sameena Shah. 2015. Real-time rumor debunking on Twitter. In *Proceedings of CIKM*, pages 1867–1870.

Yang Liu and Yi-fang Brook Wu. 2018. Early detection of fake news on social media through propagation path classification with recurrent and convolutional networks. In *Proceedings of AAAI*, pages 354–361.

Michal Lukasik, Kalina Bontcheva, Trevor Cohn, Arkaitz Zubiaga, Maria Liakata, and Rob Procter. 2019. Gaussian processes for rumour stance classification in social media. *ACM Transactions on Information Systems (TOIS)*, 37(2):20.

Michal Lukasik, PK Srijith, Duy Vu, Kalina Bontcheva, Arkaitz Zubiaga, and Trevor Cohn. 2016. Hawkes processes for continuous time sequence classification: An application to rumour stance classification in Twitter. In *Proceedings of ACL*, pages 393–398.

Jing Ma, Wei Gao, Prasenjit Mitra, Sejeong Kwon, Bernard J Jansen, Kam-Fai Wong, and Meeyoung Cha. 2016. Detecting rumors from microblogs with recurrent neural networks. In *Proceedings of IJCAI*, pages 3818–3824.

Jing Ma, Wei Gao, and Kam-Fai Wong. 2017. Detect rumors in microblog posts using propagation structure via kernel learning. In *Proceedings of ACL*, pages 708–717.

Jing Ma, Wei Gao, and Kam-Fai Wong. 2018a. Detect rumor and stance jointly by neural multi-task learning. In *Proceedings of WWW Companion*, pages 585–593.

Jing Ma, Wei Gao, and Kam-Fai Wong. 2018b. Rumor detection on Twitter with tree-structured recursive neural networks. In *Proceedings of ACL*, pages 1980–1989.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of EMNLP*, pages 1506–1515.

Marcelo Mendoza, Barbara Poblete, and Carlos Castillo. 2010. Twitter under crisis: Can we trust what we RT? In *Proceedings of the 1st Workshop on Social Media Analytics (SOMA)*, pages 71–79.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, pages 3111–3119.

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. SemEval-2016 task 6: Detecting stance in tweets. In *Proceedings of SemEval*, pages 31–41.

Mitra Mohtarami, Ramy Baly, James Glass, Preslav Nakov, Lluís Màrquez, and Alessandro Moschitti. 2018. Automatic stance detection using end-to-end memory networks. In *Proceedings of NAACL-HLT*, pages 767–776.

Ray Oshikawa, Jing Qian, and William Yang Wang. 2018. A survey on natural language processing for fake news detection. *arXiv preprint arXiv:1811.00770*.

Endang Wahyu Pamungkas, Valerio Basile, and Viviana Patti. 2018. Stance classification for rumour analysis in Twitter: Exploiting affective information and conversation structure. In *Proceedings of the 2nd International Workshop on Rumours and Deception in Social Media (RDSM)*.

Lahari Poddar, Wynne Hsu, Mong Li Lee, and Shruti Subramaniyam. 2018. Predicting stances in Twitter conversations for detecting veracity of rumors: A neural approach. In *Proceedings of ICTAI*, pages 65–72.

Kashyap Popat, Subhabrata Mukherjee, Andrew Yates, and Gerhard Weikum. 2018. DeClarE: Debunking fake news and false claims using evidence-aware deep learning. In *Proceedings of EMNLP*, pages 22–32.

Rob Procter, Farida Vis, and Alex Voss. 2013. Reading the riots on Twitter: Methodological innovation for the analysis of big data. *International Journal of Social Research Methodology*, 16(3):197–214.

Vahed Qazvinian, Emily Rosengren, Dragomir R Radev, and Qiaozhu Mei. 2011. Rumor has it: Identifying misinformation in microblogs. In *Proceedings of EMNLP*, pages 1589–1599.

Natali Ruchansky, Sungyong Seo, and Yan Liu. 2017. CSI: A hybrid deep model for fake news detection. In *Proceedings of CIKM*, pages 797–806.

Karishma Sharma, Feng Qian, He Jiang, Natali Ruchansky, Ming Zhang, and Yan Liu. 2019. Combating fake news: A survey on identification and mitigation techniques. *ACM Transactions on Intelligent Systems and Technology (TIST)*.

Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. 2017. Fake news detection on social media: A data mining perspective. *ACM SIGKDD Explorations Newsletter*, 19(1):22–36.

Swapna Somasundaran and Janyce Wiebe. 2009. Recognizing stances in online debates. In *Proceedings of ACL-IJCNLP*, pages 226–234.

Amir Pouran Ben Veyseh, Javid Ebrahimi, Dejing Dou, and Daniel Lowd. 2017. A temporal attentional model for rumor stance classification. In *Proceedings of CIKM*, pages 2335–2338.

Soroush Vosoughi, Deb Roy, and Sinan Aral. 2018. The spread of true and false news online. *Science*, 359(6380):1146–1151.

Weiming Wen, Songwen Su, and Zhou Yu. 2018. Cross-lingual cross-platform rumor verification pivoting on multimedia content. In *Proceedings of EMNLP*, pages 3487–3496.

Ke Wu, Song Yang, and Kenny Q Zhu. 2015. False rumors detection on Sina weibo by propagation structures. In *Proceedings of ICDE*, pages 651–662.

Michihiro Yasunaga, Rui Zhang, Kshitijh Meelu, Ayush Pareek, Krishnan Srinivasan, and Dragomir Radev. 2017. Graph-based neural multi-document summarization. In *Proceedings of CoNLL*, pages 452–462.

Li Zeng, Kate Starbird, and Emma S Spiro. 2016. #unconfirmed: Classifying rumor stance in crisis-related social media messages. In *Proceedings of ICWSM*, pages 747–750.

Yuhao Zhang, Peng Qi, and Christopher D Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of EMNLP*, pages 2205–2215.

Zhe Zhao, Paul Resnick, and Qiaozhu Mei. 2015. Enquiring minds: Early detection of rumors in social media from enquiry posts. In *Proceedings of WWW*, pages 1395–1405.

Xinyi Zhou and Reza Zafarani. 2018. Fake news: A survey of research, detection methods, and opportunities. *arXiv preprint arXiv:1812.00315*.

Arkaitz Zubiaga, Ahmet Aker, Kalina Bontcheva, Maria Liakata, and Rob Procter. 2018. Detection and resolution of rumours in social media: A survey. *ACM Computing Surveys (CSUR)*, 51(2):32.

Arkaitz Zubiaga, Elena Kochkina, Maria Liakata, Rob Procter, and Michal Lukasik. 2016a. Stance classification in rumours as a sequential task exploiting the tree structure of social media conversations. In *Proceedings of COLING*, pages 2438–2448.

Arkaitz Zubiaga, Maria Liakata, and Rob Procter. 2017. Exploiting context for rumour detection in social media. In *Proceedings of SocInfo*, pages 109–123.

Arkaitz Zubiaga, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Peter Tolmie. 2016b. Analysing how people orient to and spread rumours in social media by looking at conversational threads. *PloS one*, 11(3):e0150989.

# A Implementation Details

## A.1 Hyperparameters

**SemEval-2017 task 8 dataset** We pretrain 200-dimensional word embeddings by Skip-gram with negative sampling (Mikolov et al., 2013), and they are fixed during the training process. We set the dimension of content feature to 200. We use a two-layer GCN, and the output sizes of two layers are 200 and 4, respectively. The max-pooling function in Eq. (7) is to select the maximum value of each dimension, and thus the size of $\boldsymbol{v}$ is equal to that of $\boldsymbol{v}_i$. The trade-off parameter $\lambda$ is set to 1. We add dropout with 0.5 ratio to all but the last GCN layers and the FNN layer (used for veracity prediction). We train our model with 0.001 learning rate and 32 batch size using Adam (Kingma and Ba, 2015). For this dataset, after tuning hyperparameters on the development set, we merge the training and the development sets and re-train our model on the merged set.

**PHEME dataset** We set the learning rate to 0.005 for accelerating the training process. Other configurations are same to that of SemEval dataset. Because only a subset of PHEME dataset contains stance labels, if a training conversation does not have stance labels, we will not compute its loss function of rumor stance classification task during the training process.

## A.2 The CNN Layer in Ablation Study

In Section 5.4.2, to demonstrate the effectiveness of modeling the temporal dynamics of stance evolution, we replace the GRU layer by a CNN layer that only captures local temporal information. Specifically, this CNN layer consists of a 1D convolution layer and a max-pooling function (Kim, 2014). We use three different filter windows: 2, 3 and 4. Each filter window has 100 feature maps. The output vector of this CNN layer is then fed into an FNN layer with softmax function to obtain the predicted veracity distribution.

# B Numerical Results of Figure 4

Table 5 shows the exact numerical numbers of the results in Figure 4.

| Depth | Method | | |
|---|---|---|---|
| | BranchLSTM | Original-GCN | Ours |
| 0 | **0.481** | **0.481** | 0.381 |
| 1 | 0.348 | 0.363 | **0.468** |
| 2 | 0.233 | 0.297 | **0.467** |
| 3 | 0.232 | 0.300 | **0.480** |
| 4 | 0.481 | 0.548 | **0.672** |
| 5 | **0.321** | 0.292 | **0.321** |
| 6+ | 0.223 | 0.337 | **0.438** |

Table 5: Stance classification results w.r.t. different depths.

# Reconstructing Capsule Networks for Zero-shot Intent Classification

**Han Liu**[1*] **Xiaotong Zhang**[1*] **Lu Fan**[1*] **Xuandi Fu**[1]
**Qimai Li**[1] **Xiao-Ming Wu**[1†] **Albert Y.S. Lam**[2]
Department of Computing, The Hong Kong Polytechnic University, Hong Kong S.A.R.[1]
Fano Labs, Hong Kong S.A.R.[2]
{cshliu,csxtzhang,cslfan}@comp.polyu.edu.hk
{csxfu,csqmli,csxmwu}@comp.polyu.edu.hk, albert@fano.ai

## Abstract

Intent classification is an important building block of dialogue systems. With the burgeoning of conversational AI, existing systems are not capable of handling numerous fast-emerging intents, which motivates zero-shot intent classification. Nevertheless, research on this problem is still in the incipient stage and few methods are available. A recently proposed zero-shot intent classification method, IntentCapsNet, has been shown to achieve state-of-the-art performance. However, it has two unaddressed limitations: (1) it cannot deal with polysemy when extracting semantic capsules; (2) it hardly recognizes the utterances of unseen intents in the generalized zero-shot intent classification setting. To overcome these limitations, we propose to reconstruct capsule networks for zero-shot intent classification. First, we introduce a dimensional attention mechanism to fight against polysemy. Second, we reconstruct the transformation matrices for unseen intents by utilizing abundant latent information of the labeled utterances, which significantly improves the model generalization ability. Experimental results on two task-oriented dialogue datasets in different languages show that our proposed method outperforms IntentCapsNet and other strong baselines.

## 1 Introduction

With the advent of conversational AI, task-oriented spoken dialogue systems are becoming ubiquitous, e.g., chatbots deployed on different applications, or modules integrated in the popular virtual personal assistants like Apple Siri or Microsoft Cortana (Chen et al., 2017). To improve business effectiveness and user satisfaction, accurately identifying the intents behind user utterances is indispensable. However, it is extremely challenging not only because user queries are sometimes short and expressed diversely, but also because it may continuously encounter new or unacquainted intents popped up quickly from various domains. Conventional intent classification methods (Hu et al., 2009; Tur et al., 2012; Xu and Sarikaya, 2013; Ravuri and Stolcke, 2015; Liu and Lane, 2016; Nam et al., 2016) typically train a supervised learning model on large amounts of labeled data, and are not effective in recognizing emerging unseen intents.

Several zero-shot learning approaches attempted to address the challenges for classifying intents whose instances are not present during training. One common idea is to utilize some external resources (Ferreira et al., 2015a,b; Yazdani and Henderson, 2015; Kumar et al., 2017; Zhang et al., 2019) such as label ontologies or manually defined attributes. However, such external resources are usually unavailable, as they require substantial extra time and expensive human labour to produce. To implement zero-shot intent classification more easily and intelligently, recent works rely more on the word embeddings of intent labels, which can be easily pretrained on text corpus. Methods proposed by Chen et al. (2016) and Kumar et al. (2017) utilize neural networks to project intent labels and data samples to the same semantic space and then measure their similarity. However, learning a good projection function is usually difficult due to the diversity of user expressions, especially in some complex domains such as medical queries (Zhang et al., 2016).

Unlike previous models, IntentCapsNet (Xia et al., 2018) employs capsule networks to extract high-level semantic features and then transfers the prediction vectors for seen intents to unseen intents. Although IntentCapsNet has achieved impressive performance in some zero-shot intent

[*] Equal contribution.
[†] Corresponding author.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4799–4809,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

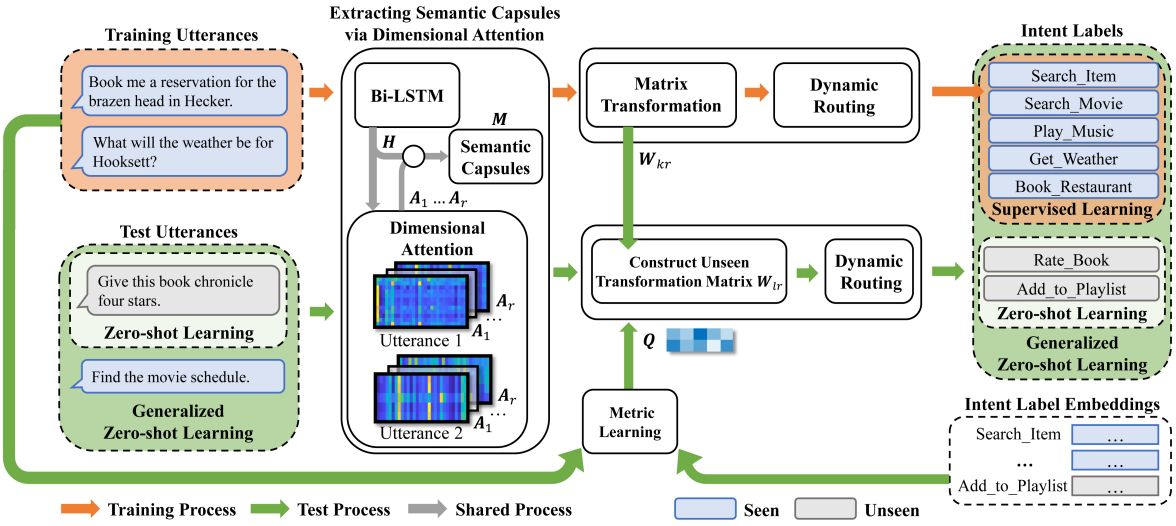


Figure 1: Illustration of our framework ReCapsNet-ZS. **In the training process**, labeled utterances are first encoded by Bi-LSTM. Then, a set of semantic capsules are extracted via the dimensional attention module. Finally, these semantic capsules are fed to a capsule network to train a model for predicting the seen intents. **In the testing process**, to predict the unseen intents, a metric learning method is trained on labeled utterances and intent label embeddings to learn the similarities between the unseen and seen intents. Then, the learned similarities and the transformation matrices for the seen intents trained by capsule networks are used to construct the transformation matrices for the unseen intents. When a test utterance arrives, it is first encoded into semantic capsules by the trained Bi-LSTM and dimensional attention module. There are two settings for intent classification. (1) Zero-shot intent classification: only utterances of the unseen intents participate in testing, so each utterance needs to be classified to one of the unseen intents. In this case, only the transformation matrices for the unseen intents are used for prediction. (2) Generalized zero-shot intent classification: test utterances may come from both the seen and unseen intents, so each utterance needs to be classified to either a seen or an unseen intent. In this case, the transformation matrices for the seen and unseen intents are all used for prediction.

classification tasks, it has two unaddressed limitations. (1) The self-attention module of IntentCapsNet cannot handle polysemy, which weakens the representation capacity of semantic capsules. (2) For the generalized zero-shot classification setting where newly arrived utterances come from both seen and unseen intents, the method of IntentCapsNet for constructing the prediction vectors can easily cause the model to completely fail in detecting unseen intents, which is clearly undesirable and inadequate for real dialogue systems.

In this paper, we propose to reconstruct capsule networks for zero-shot intent classification (ReCapsNet-ZS), which effectively addresses the limitations of IntentCapsNet and adapts well to the generalized zero-shot intent classification tasks. As illustrated in Figure 1, ReCapsNet-ZS consists of two components. First, it introduces a dimensional attention module to alleviate the polysemy problem, which helps to extract semantic features for capsule networks. Second, it computes the similarities between unseen and seen intents by utilizing the rich latent information of labeled utterances, and then constructs the transformation matrices for unseen intents with the computed similarities and the trained transformation matrices for seen intents, which greatly improves the generalization ability to unseen intents.

To verify the effectiveness of the proposed ReCapsNet-ZS for zero-shot intent classification, we conduct extensive experiments on two real task-oriented dialogue datasets in English and Chinese respectively. The empirical study validates our proposals and shows promising results of ReCapsNet-ZS, which are significantly better than state-of-the-art methods, especially on the generalized zero-shot intent classification tasks.

## 2 Related Works

**Zero-shot Intent Classification.** Zero-shot learning (Larochelle et al., 2008; Palatucci et al., 2009) aims to use the knowledge learned from seen classes, of which abundant labeled samples are typically available for training, to recognize unseen classes, of which no labeled samples are provided. It has been widely studied in computer

vision (Ba et al., 2015; Xian et al., 2016) and natural language processing (Sappadla et al., 2016; Zhang et al., 2019).

Zero-shot intent classification is an important and challenging task for many natural language understanding applications (Hu et al., 2009; Liu and Lane, 2016; Nam et al., 2016; Xu and Sarikaya, 2013), in which new intents emerge constantly and they cannot be easily recognized. Several methods have been proposed to tackle this problem. Ferreira et al. (2015a,b) and Yazdani and Henderson (2015) utilize external resources such as label ontologies or manually defined attributes to find the relationship between seen and unseen intent labels. However, the external resources are usually difficult to obtain, as collecting them is labor intensive and time consuming. Chen et al. (2016) and Kumar et al. (2017) project the utterances and intent labels to a same semantic space and then compute the similarities between utterances and intent labels (Chen et al., 2016; Kumar et al., 2017). However, diverse user expressions may make it difficult to learn a good projection function and thus affect the classification performance. Recently, Xia et al. (2018) extend capsule networks for zero-shot intent classification by transferring the prediction vectors from seen classes to unseen classes. However, there are some key issues left to be resolved, including how to deal with polysemy in word embeddings and how to improve the model generalization ability to unseen intents in the generalized zero-shot intent classification setting.

**Capsule Networks.** Capsule Networks (Sabour et al., 2017) were first proposed to address the shortcomings of convolutional neural networks (CNN) in the domain of computer vision. It allows the networks to learn part-whole invariant relationships consecutively. Recently, some studies have attempted to apply capsule networks in the domain of natural language processing (Yang et al., 2018; Geng et al., 2019; Xia et al., 2018) and obtained promising results. Yang et al. (2018) first extend capsule networks for text classification. Geng et al. (2019) successfully combine the dynamic routing algorithm with some meta-learning framework for few-shot text classification. However, their model still requires some labeled samples for each class. Xia et al. (2018) propose a model based on capsule networks for zero-shot intent classification and has achieved state-of-the-art performance, but as mentioned above, their model has some intrinsic limitations remained to be addressed.

## 3 Preliminaries

### 3.1 Problem Formulation

Given the set of all intent labels $Y = Y^s \bigcup Y^u$, where $Y^s = \{y_1^s, y_2^s, \ldots, y_K^s\}$ and $Y^u = \{y_1^u, y_2^u, \ldots, y_L^u\}$ are the sets of seen and unseen intent labels respectively. There is no overlap between $Y^s$ and $Y^u$, i.e., $Y^s \bigcap Y^u = \emptyset$, and $K$ and $L$ are the numbers of seen and unseen intent labels respectively. The embeddings of the seen and unseen intent labels are denoted by $E^s = \{\boldsymbol{e}_1^s, \boldsymbol{e}_2^s, \ldots, \boldsymbol{e}_K^s\}$ and $E^u = \{\boldsymbol{e}_1^u, \boldsymbol{e}_2^u, \ldots, \boldsymbol{e}_L^u\}$ respectively. Each embedding is a $d$-dimensional vector. For all the seen and unseen intent labels, their associated embeddings are available. The sample (utterance) sets for the seen and unseen intent labels are denoted by $X^s = \{x_1^s, x_2^s, \ldots, x_{n_s}^s\}$ and $X^u = \{x_1^u, x_2^u, \ldots, x_{n_u}^u\}$ respectively, where $n_s$ is the number of instances of the seen labels and $n_u$ is the number of instances of the unseen labels.

**Zero-shot Intent Classification.** For this setting, the training set is $X^{tr} = \{X^s, Y^s\}$, and $X^u$ is not available for training. In the test phase, the goal is to assign an unseen intent label $y \in Y^u$ to a given utterance.

**Generalized Zero-shot Intent Classification.** For this setting, the training procedure is the same as above, while the difference is in the test phase, where the goal is to assign an intent label $y \in Y^s \bigcup Y^u$ to a given utterance.

In this paper, we aim to reconstruct capsule networks for handling both of the two settings of zero-shot intent classification.

### 3.2 Limitations of IntentCapsNet

IntentCapsNet (Xia et al., 2018) is the first work to employ capsule networks for zero-shot intent classification. It exploits the self-attention mechanism to extract semantic features (capsules) of an utterance. For zero-shot intent classification, it utilizes the vote vectors of seen intents and the similarities between seen and unseen intents based on Euclidean distance to make predictions for unseen intents. Although IntentCapsNet has demonstrated strong performance, it has two fundamental limitations.

**Limitation 1.** The self-attention module of IntentCapsNet cannot handle the polysemy problem, which limits the representation capacity of semantic capsules.

Typically, a word is represented by a multi-dimensional embedding. Since a word can have different meanings in different contexts, some interesting recent studies (Shen et al., 2018; Şenel et al., 2018) suggest that different dimensions of a word embedding may tend to represent different semantic meanings. For example, the word "book" has different meanings in the two utterances: "Book a restaurant in Michigan for 4 people" and "Give 4 out of 6 points to this book". For the embedding of the word "book", it is hypothesized that some dimensions may be more indicative for the first meaning – "reserve", while some other dimensions may be more indicative for the second meaning. Apparently, the self-attention mechanism cannot pay more attentions to the dimensions that best describe the specific meaning of a word in a given context, as it assigns the same attention score for all the dimensions, which significantly limits the representation capacity of semantic capsules and undermines the performance of capsule networks.

**Limitation 2.** For the generalized zero-shot classification setting, the method of IntentCapsNet for constructing the prediction vectors is highly likely to cause the model to lose generalization ability to unseen intents.

Here, we provide an analysis of IntentCapsNet for predicting an unseen intent in the generalized zero-shot classification setting. In IntentCapsNet, the probability of a test utterance $x$ belonging to a seen intent label $k$ is computed as:

$$P_k = \|\sum_{r=1}^{R} c_{kr}\boldsymbol{p}_{k|r}\| = \|\sum_{r=1}^{R}\boldsymbol{g}_{k,r}\|, \quad (1)$$

where $\|\cdot\|$ is the L2-norm of a vector, $R$ is the number of semantic capsules, $\boldsymbol{p}_{k|r}$ is the prediction vector for the $r$-th semantic capsule with respect to the seen intent $k$, and $c_{kr}$ is the weight of the $r$-th semantic capsule with respect to the seen intent $k$, which is computed by the dynamic routing algorithm of capsule networks. $\boldsymbol{g}_{k,r} = c_{kr}\boldsymbol{p}_{k|r}$ is called the $r$-th vote vector for the seen intent $k$. By Eq. (1), we have a tight upper bound for $P_k$:

$$P_k \leqslant \sum_{r=1}^{R}\|\boldsymbol{g}_{k,r}\|. \quad (2)$$

IntentCapsNet computes the probability of $x$ belonging to an unseen intent label $l$ as:

$$P_l = \|\sum_{r=1}^{R} c_{lr}\boldsymbol{u}_{l|r}\| = \|\sum_{r=1}^{R} c_{lr}\sum_{k=1}^{K} q_{lk}\boldsymbol{g}_{k,r}\|, \quad (3)$$

where $\boldsymbol{u}_{l|r}$ is the prediction vector for the $r$-th semantic capsule with respect to the unseen intent $l$, and $c_{lr}$ is the weight of the $r$-th semantic capsule with respect to the unseen intent $l$, which is determined by the dynamic routing algorithm. $\boldsymbol{u}_{l|r} = \sum_{k=1}^{K} q_{lk}\boldsymbol{g}_{k,r}$, where $K$ is the number of seen intents, $\boldsymbol{g}_{k,r}$ is the $r$-th vote vector for the seen intent $k$, and $q_{lk}$ is the similarity between an unseen intent $y_l^u \in Y^u$ and a seen intent $y_k^s \in Y^s$. $q_{lk} = \frac{exp(-d(\boldsymbol{e}_k^s, \boldsymbol{e}_l^u))}{\sum_{k=1}^{K} exp(-d(\boldsymbol{e}_k^s, \boldsymbol{e}_l^u))}$, where $\boldsymbol{e}_k^s$ and $\boldsymbol{e}_l^u$ are the embeddings of the seen and unseen intents respectively, and $d(\boldsymbol{e}_k^s, \boldsymbol{e}_l^u)$ is the scaled squared Euclidean distance between $\boldsymbol{e}_k^s$ and $\boldsymbol{e}_l^u$. Since $q_{lk} \in (0,1)$, $c_{lr} \in (0,1)$, $\sum_{k=1}^{K} q_{lk} = 1$ and $\sum_{r=1}^{R} c_{lr} = 1$, we have a tight upper bound for $P_l$:

$$P_l \leqslant \sum_{r=1}^{R} c_{lr}\sum_{k=1}^{K} q_{lk}\|\boldsymbol{g}_{k,r}\| \leqslant \|\boldsymbol{g}_{k,r}\|_{\max}, \quad (4)$$

where $\|\boldsymbol{g}_{k,r}\|_{\max}$ is the maximum among $\|\boldsymbol{g}_{k,r}\|$, $\forall r \in \{1, 2, \ldots, R\}$ and $\forall k \in \{1, 2, \ldots, K\}$.

By Eq. (2) & (4), it can be seen that the upper bound of $P_k$ is much larger than $P_l$, indicating that for any utterance $x$, it is highly likely that $P(y \in Y^s|x)$ is larger than $P(y \in Y^u|x)$. Hence, for generalized zero-shot classification, with high probability IntentCapsNet will classify a test utterance to the seen intents, which is also verified by our experiments in section 5.

## 4 The Proposed Approach

To overcome the limitations of IntentCapsNet, we propose to reconstruct capsule networks for zero-shot intent classification. In particular, we introduce two modules to capsule networks: (1) a dimensional attention module that helps to extract more representative semantic capsules and (2) a new method for constructing the transformation matrices to improve the model generalization ability to unseen intents.

### 4.1 Dimensional Attention Capsule Networks

**Pre-processing.** An utterance with $T$ words can be represented as $x = \{\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_T\}$, where

$\boldsymbol{w}_t \in \mathbb{R}^{d_w}$ is the word embedding of the $t$-th word and can be pretrained by the skip-gram model (Mikolov et al., 2013). Each word can be further encoded sequentially using a recurrent neural network such as bidirectional LSTM (Hochreiter and Schmidhuber, 1997), i.e.,

$$\begin{aligned} \overrightarrow{\boldsymbol{h}}_t &= \text{LSTM}_{fw}(\boldsymbol{w}_t, \overrightarrow{\boldsymbol{h}}_{t-1}), \\ \overleftarrow{\boldsymbol{h}}_t &= \text{LSTM}_{bw}(\boldsymbol{w}_t, \overleftarrow{\boldsymbol{h}}_{t+1}), \end{aligned} \tag{5}$$

where $\text{LSTM}_{fw}$ and $\text{LSTM}_{bw}$ denote the forward and backward LSTM respectively, and $\overrightarrow{\boldsymbol{h}}_t \in \mathbb{R}^{d_h}$ and $\overleftarrow{\boldsymbol{h}}_t \in \mathbb{R}^{d_h}$ are the hidden states of the word $\boldsymbol{w}_t$ learned from $\text{LSTM}_{fw}$ and $\text{LSTM}_{bw}$ respectively. The entire hidden state of $\boldsymbol{w}_t$ is represented by concatenating $\overrightarrow{\boldsymbol{h}}_t$ and $\overleftarrow{\boldsymbol{h}}_t$, i.e., $\boldsymbol{h}_t = [\overrightarrow{\boldsymbol{h}}_t, \overleftarrow{\boldsymbol{h}}_t]$, and the hidden state matrix of the utterance is $\boldsymbol{H} = [\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_T] \in \mathbb{R}^{2d_h \times T}$.

#### 4.1.1 Extracting Semantic Capsules with Dimensional Attention

In general, an utterance is composed of multiple semantic features, and these semantic features collectively contribute to a more abstract intent label. For example, an utterance "I want to know the temperature of Hong Kong" is composed by multiple semantic features such as get_action (want to know), weather (temperature), and city_name (Hong Kong), and these semantic features collectively reflect the intent label "Get_Weather". Capsule networks provide a hierarchical reasoning structure for modeling semantic features for intent classification. First, the primary capsules in capsule networks can properly match multiple semantic features of an utterance. Second, the dynamic routing mechanism of capsule networks can be used to automatically learn the importance weight of each semantic feature and aggregate them into a high-level intent label.

It is assumed that a high-level semantic feature of an utterance is largely generated by some of its words that have similar semantic meaning (Xia et al., 2018). To extract the semantic features of an utterance, the key problem is to learn the importance weight of each word for a semantic feature. IntentCapsNet (Xia et al., 2018) utilizes the self-attention mechanism to extract the semantic features (capsules) of each utterance. However, self-attention cannot effectively deal with polysemy. Inspired by the work of Shen et al. (2018), we propose to use the dimensional attention mechanism to alleviate the polysemy problem in extracting semantic features. Dimensional attention can automatically assign different attention scores to different dimensions of a word embedding, which not only helps to solve the polysemy problem to some extent, but also expands the search space of the attention parameters, thus improving model flexibility and effectiveness.

Assume each utterance has $R$ semantic features. We propose to learn a dimensional attention matrix $\boldsymbol{A}_r \in \mathbb{R}^{2d_h \times T}$ that encodes the dimensional attentions of the $T$ words with respect to the $r$-th semantic feature by:

$$\boldsymbol{A}_r = \text{softmax}\left(\boldsymbol{F}_2 \, \text{ReLU}\left(\boldsymbol{F}_1 \boldsymbol{H}\right)\right), \tag{6}$$

where $\boldsymbol{F}_1 \in \mathbb{R}^{d_a \times 2d_h}$ and $\boldsymbol{F}_2 \in \mathbb{R}^{2d_h \times d_a}$ are the trainable parameters, and $\boldsymbol{A}_r(i, j)$ (the element of $\boldsymbol{A}_r$ in the $i$-th row and $j$-th column) means the importance weight of the $i$-th dimension of the $j$-th word embedding to the $r$-th semantic feature. Compared with self-attention, dimensional attention can help to choose the appropriate dimensions of a word embedding that can best express the specific meaning of the word in a given context.

After obtaining $\boldsymbol{A}_r$, the $r$-th semantic feature $\boldsymbol{m}_r \in \mathbb{R}^{2d_h}$ is computed by:

$$\boldsymbol{m}_r = \sum\nolimits_{\text{row}} \left(\boldsymbol{A}_r \odot \boldsymbol{H}\right), \tag{7}$$

where $\odot$ is element-wise multiplication, and $\sum_{\text{row}}$ is an operator that sums up elements of each row. The entire semantic features for each utterance is $\boldsymbol{M} = [\boldsymbol{m}_1, \boldsymbol{m}_2, \ldots, \boldsymbol{m}_R] \in \mathbb{R}^{2d_h \times R}$.

#### 4.1.2 Improved Max-margin Loss

The semantic features of the utterance can then be fed into a capsule network to learn the intent. First, we transform each semantic feature $\boldsymbol{m}_r$ of the utterance to a prediction vector with respect to each intent as:

$$\boldsymbol{p}_{k|r} = \boldsymbol{W}_{kr} \boldsymbol{m}_r, \tag{8}$$

where $\boldsymbol{p}_{k|r} \in \mathbb{R}^{d_p}$ is the prediction vector of the $r$-th semantic feature with respect to the $k$-th intent, and $\boldsymbol{W}_{kr} \in \mathbb{R}^{d_p \times 2d_h}$ is the associated transformation matrix.

In training, there are $K$ output capsules, corresponding to $K$ seen intents. The $k$-th output capsule $\boldsymbol{o}_k$ is the weighted sum of all the prediction vectors $\boldsymbol{p}_{k|r}$ ($r \in \{1, \ldots, R\}$),

$$\boldsymbol{o}_k = \sum_{r=1}^{R} c_{kr} \boldsymbol{p}_{k|r}, \tag{9}$$

**Algorithm 1** Dynamic Routing Algorithm

**Procedure** Dynamic Routing($\boldsymbol{p}_{k|r}, n_{\text{route}}$)
  for all semantic capsule $r$ and intent capsule $k$: $b_{kr} \leftarrow 0$.
  **for** $n_{\text{route}}$ iterations **do**
    for all semantic capsule $r$: $\boldsymbol{c}_r \leftarrow \text{softmax}(\boldsymbol{b}_r)$.
    for all intent capsule $k$: $\boldsymbol{o}_k \leftarrow \sum_{r=1}^{R} c_{kr}\boldsymbol{p}_{k|r}$.
    for all intent capsule $k$: $\boldsymbol{v}_k \leftarrow \text{squash}(\boldsymbol{o}_k)$.
    for all semantic capsule $r$ and intent capsule $k$:
      $b_{kr} \leftarrow b_{kr} + \boldsymbol{p}_{k|r} \cdot \boldsymbol{v}_k$.
  **end for**
  **return** $\boldsymbol{v}_k$.

where $c_{kr}$ is the coupling coefficient representing the contribution degree of the $r$-th semantic feature to the $k$-th intent, which can be computed by the dynamic routing algorithm (Algorithm 1).

Then, a squashing function $\text{squash}(\cdot)$ is applied on $\boldsymbol{o}_k$, and the final output capsule of the $k$-th intent is:

$$\boldsymbol{v}_k = \text{squash}(\boldsymbol{o}_k) = \frac{\|\boldsymbol{o}_k\|^2}{1+\|\boldsymbol{o}_k\|^2}\frac{\boldsymbol{o}_k}{\|\boldsymbol{o}_k\|}. \quad (10)$$

Now, the probability of the existence of the $k$-th intent can be represented as the length of the output capsule $\boldsymbol{v}_k$. The computation procedure of $\boldsymbol{v}_k$ is shown in Algorithm 1, where $\boldsymbol{p}_{k|r} \cdot \boldsymbol{v}_k$ denotes the inner product between $\boldsymbol{p}_{k|r}$ and $\boldsymbol{v}_k$.

To train the dimensional attention capsule network, we propose an improved max-margin loss function consisting of two parts.

The first part is the max-margin loss on each labeled utterance, which is the original loss function of capsule networks (Sabour et al., 2017):

$$\begin{aligned} L_1 = \sum_{k=1}^{K} \{ & y_k \max(0, m^+ - \|\boldsymbol{v}_k\|)^2 \\ & + \lambda(1-y_k)\max(0, \|\boldsymbol{v}_k\| - m^-)^2\}, \end{aligned} \quad (11)$$

where $y_k = 1$ if the utterance is of intent label $k$ and $y_k = 0$ otherwise, $\lambda$ is a down-weighting parameter, and $m^+$ and $m^-$ are the margins.

The second part is to ensure the diversity of the semantic capsules, i.e., different semantic capsules are likely to be generated by different words in an utterance. The importance weight of each word to the $r$-th semantic capsule can be represented by the average value of each column of the dimensional attention matrix $\boldsymbol{A}_r$, i.e.,

$$\boldsymbol{s}_r = \frac{1}{2d_h}\sum\nolimits_{\text{col}} \boldsymbol{A}_r, \quad (12)$$

where $\boldsymbol{s}_r \in \mathbb{R}^{1\times T}$ and $\sum_{\text{col}}$ is an operator that sums up elements of each column. Denote by $\boldsymbol{S} = [\boldsymbol{s}_1^\top, \boldsymbol{s}_2^\top, \ldots, \boldsymbol{s}_R^\top] \in \mathbb{R}^{T\times R}$ the importance weight matrix of each word to all the $R$ semantic capsules. To ensure the diversity of the semantic capsules, a natural idea is to constrain the columns of $\boldsymbol{S}$ to be orthogonal with the following loss function:

$$L_2 = \|\boldsymbol{S}^\top\boldsymbol{S} - \boldsymbol{I}\|_F^2, \quad (13)$$

where $\|\cdot\|_F$ is the Frobenius norm of a matrix.

Combining Eq. (11) and Eq. (13), the overall loss function of the proposed dimensional attention capsule network is:

$$L_{total} = L_1 + \beta L_2, \quad (14)$$

where $\beta$ is a trade-off parameter. By minimizing $L_{total}$ with gradient descent methods, all model parameters including $\boldsymbol{F}_1$, $\boldsymbol{F}_2$ and $\boldsymbol{W}_{kr}$ can be learned.

### 4.2 Zero-shot Intent Classification

To solve zero-shot intent classification with capsule networks, two key problems need to be addressed. (1) How to find the relationship between unseen and seen intents? (2) How to make predictions for unseen intents?

**Measuring Intent Relations.** To tackle the first problem, we propose to learn a Mahalanobis distance metric to measure the relationship between unseen and seen intents. Specifically, given the embeddings of an unseen intent $l$ and a seen intent $k$, their squared Mahalanobis distance is given by:

$$d_M(\boldsymbol{e}_l^u, \boldsymbol{e}_k^s) = (\boldsymbol{e}_l^u - \boldsymbol{e}_k^s)^\top \boldsymbol{\Omega}^{-1}(\boldsymbol{e}_l^u - \boldsymbol{e}_k^s), \quad (15)$$

where $\boldsymbol{\Omega}$ is a learnable covariance matrix which models the correlation between dimensions of the embedding. Note that IntentCapsNet (Xia et al., 2018) also tries to use Eq. (15) to model the relationship between unseen and seen intents, but it ignores the correlation between dimensions and simply sets $\boldsymbol{\Omega} = \sigma^2\boldsymbol{I}$ ($\sigma$ is a scaling hyper-parameter), which is actually a scaled squared Euclidean distance.

As the number of intents is limited, it is difficult to learn a desirable covariance matrix $\boldsymbol{\Omega}$ with the intent embeddings only. Fortunately, we can leverage the word embeddings of the utterances, which come from the same semantic space as the intent embeddings (pre-trained by the same skip-gram model). Hence, we propose to learn the covariance matrix $\boldsymbol{\Omega}$ with the labeled utterances in the

training set. Inspired by the work of Ying and Li (2012), we propose to learn the Mahalanobis distance metric by optimizing the objective:

$$\max_{\mathbf{\Omega}} \min_{(i,j)\in\mathcal{D}} d_M(\boldsymbol{z}_i^s, \boldsymbol{z}_j^s), \quad s.t. \sum_{(i,j)\in\mathcal{S}} d_M(\boldsymbol{z}_i^s, \boldsymbol{z}_j^s) \leqslant 1, \tag{16}$$

where $\mathcal{D}$ and $\mathcal{S}$ respectively denote the pair sets in which utterances belong to different classes and the same class. For an utterance $i$, $\boldsymbol{z}_i^s$ denotes the average sum of all the word embeddings. As shown by Ying and Li (2012), optimizing Eq. (16) with respect to $\mathbf{\Omega}$ is equivalent to solving an efficient eigenvalue optimization problem. With the learned metric $\mathbf{\Omega}$, we can have the relationship between any unseen and seen intents by substituting it into Eq. (15). Furthermore, we can compute the similarity between them by $q_{lk} = \exp\left(-\alpha \cdot d_M(\boldsymbol{e}_l^u, \boldsymbol{e}_k^s)\right)$, where $\alpha$ is a scaling parameter.

**Constructing Transformation Matrices.** Intuitively, if an unseen intent is similar to a seen intent, their corresponding transformation matrices should also be similar. Based on this, to solve the second problem, we propose to derive the transformation matrices of unseen intents using the transformation matrices of seen intents and the similarities between unseen and seen intents, and then make predictions for unseen intents with the transformation matrices. Specifically, given a matrix $\boldsymbol{Q} \in \mathbb{R}^{L\times K}$ that encodes the similarities between unseen and seen intents, for an unseen intent $l$, we propose to construct the transformation matrix $\boldsymbol{W}_{lr}$ for the $r$-th semantic capsule with respect to the $l$-th unseen intent by:

$$\boldsymbol{W}_{lr} = \sum_{k=1}^{K} q_{lk} \boldsymbol{W}_{kr}, \tag{17}$$

where $q_{lk}$ is the element in the $l$-th row and the $k$-th column of $\boldsymbol{Q}$, $\boldsymbol{W}_{kr}$ is the transformation matrix for the $r$-th semantic capsule with respect to the $k$-th seen intent. By Eq. (17), the transformation matrices for all unseen intents can be obtained. When a test utterance arrives, it can be directly fed into the trained dimensional attention capsule network for intent prediction.

| **Dataset** | **SNIPS-NLU** | **SMP-2018** |
|---|---|---|
| Vocab Size | 11641 | 2682 |
| Number of Samples | 13802 | 2460 |
| Average Sentence Length | 9.05 | 4.86 |
| Number of Seen Intents | 5 | 24 |
| Number of Unseen Intents | 2 | 6 |

Table 1: Dataset statistics.

| **Dataset** | $d_w$ | $d_h$ | $d_a$ | $d_p$ | $R$ | $n_{\text{route}}$ |
|---|---|---|---|---|---|---|
| **SNIPS-NLU** | 300 | 16 | 10 | 10 | 3 | 3 |
| **SMP-2018** | 300 | 32 | 30 | 10 | 8 | 3 |

Table 2: Network structure hyperparameters.

## 5 Experiments

### 5.1 Datasets

We evaluate our model on two real task-oriented dialogue datasets in different languages. Table 1 summarizes the dataset statistics.

**SNIPS-NLU.** Following (Xia et al., 2018), we use the SNIPS-NLU (SNIPS Natural Language Understanding) benchmark dataset (Coucke et al., 2018). SNIPS-NLU is an open-source single-turn English corpus, which contains crowdsourced queries evenly distributed in 7 intents.

**SMP-2018.** It is a real Chinese dialogue corpus released in SMP 2018 (The China National Conference on Social Media Processing) for user intent classification tasks in Chinese (Zhang et al., 2017). The dataset is provided by the iFLYTEK Corporation, and it can be divided into two parts: chit-chat dialogues and task-oriented dialogues. Here, we only use the task-oriented dialogues.

**Dataset Splitting.** For zero-shot intent classification, we take all the samples of seen intents as the training set, and all the samples of unseen intents as the test set. For generalized zero-shot intent classification, we randomly take 70% samples of each seen intent as the training set, and the remaining 30% samples of each seen intent and all the samples of unseen intents as the test set.

### 5.2 Baselines

We compare ReCapsNet-ZS with the following state-of-the-art zero-shot learning methods: DeViSE (Frome et al., 2013), CMT (Socher et al., 2013), CDSSM (Chen et al., 2016), Zero-shot DNN (Kumar et al., 2017) and IntentCapsNet (Xia et al., 2018). To make DeViSE and CMT suit-

| Method | SNIP-NLU | | | | | | SMP-2018 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Seen | | Unseen | | Overall | | Seen | | Unseen | | Overall | |
| | Acc | F1 | Acc | F1 | Acc | F1 | Acc | F1 | Acc | F1 | Acc | F1 |
| DeViSE | 0.9481 | 0.6536 | 0.0211 | 0.0398 | 0.4215 | 0.3049 | 0.8040 | 0.6740 | 0.0270 | 0.0310 | 0.5030 | 0.4250 |
| CMT | **0.9755** | 0.6648 | 0.0397 | 0.0704 | 0.4438 | 0.3271 | 0.8314 | 0.7221 | 0.0798 | 0.1069 | 0.5398 | 0.4834 |
| CDSSM | 0.9549 | **0.7033** | 0.0111 | 0.0218 | 0.4234 | 0.3194 | 0.6653 | 0.5540 | 0.1436 | 0.1200 | 0.4864 | 0.4052 |
| Zero-shot DNN | 0.9432 | 0.6679 | 0.0682 | 0.1041 | 0.4488 | 0.3493 | 0.7323 | 0.6116 | 0.0590 | 0.0869 | 0.5013 | 0.4316 |
| IntentCapsNet | 0.9741 | 0.6517 | 0.0000 | 0.0000 | 0.4200 | 0.2810 | **0.8850** | 0.7281 | 0.0000 | 0.0000 | 0.5375 | 0.4423 |
| ReCapsNet-ZS-Dim | **0.9743** | 0.6580 | 0.0000 | 0.0000 | 0.4201 | 0.2837 | **0.8952** | 0.7360 | 0.0000 | 0.0000 | 0.5437 | 0.4470 |
| ReCapsNet-ZS-TM | 0.9663 | **0.6845** | **0.0981** | **0.1534** | **0.4724** | **0.3824** | 0.7863 | **0.7390** | **0.1896** | **0.1537** | **0.5521** | **0.5092** |
| ReCapsNet-ZS | 0.9664 | 0.6743 | **0.1121** | **0.1764** | **0.4805** | **0.3911** | 0.8230 | **0.7450** | **0.1720** | **0.1526** | **0.5674** | **0.5124** |

Table 4: Results of generalized zero-shot intent classification. "Seen", "Unseen" and "Overall" respectively denote the performance on the utterances from seen intents, unseen intents, and both seen and unseen intents.

| Method | SNIPS-NLU | | SMP-2018 | |
|---|---|---|---|---|
| | Acc | F1 | Acc | F1 |
| DeViSE | 0.7447 | 0.7446 | **0.5456** | 0.3875 |
| CMT | 0.7396 | 0.7206 | 0.4452 | 0.4245 |
| CDSSM | 0.7588 | 0.7580 | 0.4308 | 0.3765 |
| Zero-shot DNN | 0.7165 | 0.7116 | 0.4615 | 0.3897 |
| IntentCapsNet | 0.7752 | 0.7750 | 0.4864 | 0.4227 |
| ReCapsNet-ZS-Dim | **0.7868** | **0.7859** | 0.5005 | 0.4501 |
| ReCapsNet-ZS-TM | 0.7860 | 0.7837 | 0.5315 | **0.4630** |
| ReCapsNet-ZS | **0.7996** | **0.7980** | **0.5418** | **0.4769** |

Table 3: Results of zero-shot intent classification.

able for intent classification, we use a multi-head self-attention Bi-LSTM model to encode the utterances and then feed the final hidden states to their zero-shot learning models. In addition, we conduct ablation study to evaluate the contribution of each module of our ReCapsNet-ZS. "ReCapsNet-ZS-Dim" refers to the model that only uses the dimensional attention mechanism, and "ReCapsNet-ZS-TM" refers to the one that only uses the proposed transformation matrix construction method.

### 5.3 Implementation Details

**Parameter Settings.** For SNIPS-NLU, we use 300-dim embeddings pre-trained on English Wikipedia (Bojanowski et al., 2017). For SMP-2018, we use 300-dim Chinese word embeddings pre-trained by Li et al. (2018). The main network structure hyperparameters are shown in Table 2. In addition, for the zero-shot classification setting, we set $\alpha$ to 1 for SNIPS-NLU and 10 for SMP-2018 respectively. For the generalized zero-shot classification setting, we set $\alpha$ to 1 for SNIPS-NLU and 5 for SMP-2018 respectively. To avoid overfitting, we use dropout with 0.5 dropout rate on the input of the attention layer. For the loss function, we set $\lambda = 0.5$, $m^+ = 0.9$, $m^- = 0.1$, $\beta = 0.001$, and use the Adam optimizer (Kingma and Ba, 2015) with initial learning rate 0.01.

**Evaluation Metrics.** We adopt two widely used metrics: accuracy (Acc) and micro-average F1-measure (F1) to evaluate the classification performance. Both metrics are computed with the average value weighted by the support of each class, where the support means the sample ratio of the corresponding class.

### 5.4 Result Analysis

**Zero-shot Intent Classification.** Table 3 summarizes the average results over 10 runs, where the top 2 results are highlighted in bold. The baseline results on SNIPS-NLU are taken from Xia et al. (2018). The results show that ReCapsNet-ZS outperforms all the baselines, demonstrating its superiority in tackling zero-shot intent classification. We can also see that ReCapsNet-ZS performs better than either ReCapsNet-ZS-Dim or ReCapsNet-ZS-TM, which shows the effectiveness of both of the dimensional attention mechanism and the transformation matrix construction method.

**Generalized Zero-shot Intent Classification.** Table 4 shows the average results over 10 runs, where the top 2 results are highlighted in bold. We can make the following observations. 1) The performances look much worse than the standard zero-shot setting, but it is not surprising since the seen intent labels are included in the test phase and it makes the problem harder. 2) ReCapsNet-ZS sometimes performs slightly worse than the baselines in detecting seen intents, which is because some baselines tend to classify the test utterances as seen intents, which on the other hand explain-

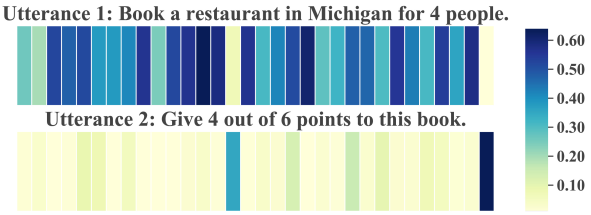


Figure 2: Comparison of the dimensional attentions of the word "book" in different contexts.

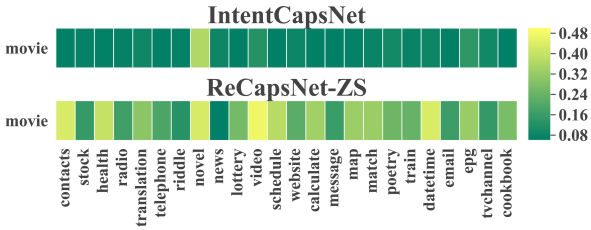


Figure 3: Comparison of the similarity scores between the unseen intent "movie" and the seen intents learned by IntentCapsNet and ReCapsNet-ZS (ours).

s why they perform much worse in detecting unseen intents. 3) ReCapsNet-ZS and ReCapsNet-ZS-TM perform much better than others in detecting unseen intents, whereas IntentCapsNet and ReCapsNet-ZS-Dim both have 0% Acc and F1. This verifies that the proposed transformation matrix construction method has much better generalization ability in detecting unseen intents. 4) Overall, ReCapsNet-ZS consistently performs the best, which further demonstrates the superiority of ReCapsNet-ZS on the generalized zero-shot intent classification tasks.

### 5.5 Visualization

**Dimensional Attentions.** Figure 2 visualizes the attention score for each dimension of the same word "book" in two different utterances (contexts) by heatmaps. The utterances are "Book a restaurant in Michigan for 4 people" and "Give 4 out of 6 points to this book", which are taken from SNIPS-NLU. It can be seen that for the two utterances the attention values of the word "book" exhibit completely different patterns, which makes sense as it contains different meanings in different contexts. Furthermore, for each utterance, the attention scores of "book" on different dimensions are also quite different. This shows that the dimensional attention mechanism can effectively capture the semantic differences of the same word in different contexts and encode more useful information than the traditional self-attention method, and thus helps to alleviate the polysemy problem.

**Similarity Scores.** Figure 3 visualizes the similarity scores between the unseen intent "movie" and the seen intents learned via metric learning by IntentCapsNet and ReCapsNet-ZS respectively on SMP-2018. It can be seen that IntentCapsNet can only discover few connections between the unseen and seen intents. In contrast, ReCapsNet-ZS can detect a lot more connections between them. Further, though the similarity scores of ReCapsNet-ZS may introduce some noise (e.g., "contacts"), they can capture many more positive relations between "movie" and the seen intents (e.g., "novel", "video", "schedule", and "datetime"), which is beneficial for detecting unseen intents.

## 6 Conclusion

In this paper, we have proposed a novel framework to reconstruct capsule networks for zero-shot intent classification and demonstrated empirically that it compares favourably with existing methods on some real dialogue datasets. The performance gains of our method come from two aspects: the introduction of a new dimensional attention module to capsule networks for feature extraction and the proposal of a new transformation scheme for detecting unseen intents.

There are several directions of the future works. One is to customize our model for few-shot intent classification. Another is to extend our framework to deal with multiple-intent classification. We also plan to apply our model in dialogue systems for low-resource languages such as Cantonese.

## Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments towards improving the manuscript. This research was supported by the grant HK ITF UIM/377.

## References

Lei Jimmy Ba, Kevin Swersky, Sanja Fidler, and Ruslan Salakhutdinov. 2015. Predicting deep zero-shot convolutional neural networks using textual descriptions. In *IEEE International Conference on Computer Vision (ICCV)*, pages 4247–4255.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Associa-*

*tion for Computational Linguistics (TACL)*, 5:135–146.

Yun-Nung Chen, Asli Çelikyilmaz, and Dilek Hakkani-Tür. 2017. Deep learning for dialogue systems. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 8–14.

Yun-Nung Chen, Dilek Z. Hakkani-Tür, and Xiaodong He. 2016. Zero-shot learning of intent embeddings for expansion by convolutional deep structured semantic models. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6045–6049.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, Maël Primet, and Joseph Dureau. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv*, abs/1805.10190.

Emmanuel Ferreira, Bassam Jabaian, and Fabrice Lefèvre. 2015a. Online adaptative zero-shot learning spoken language understanding using word-embedding. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5321–5325.

Emmanuel Ferreira, Bassam Jabaian, and Fabrice Lefèvre. 2015b. Zero-shot semantic parser for spoken language understanding. In *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pages 1403–1407.

Andrea Frome, Gregory S. Corrado, Jonathon Shlens, Samy Bengio, Jeffrey Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. 2013. Devise: A deep visual-semantic embedding model. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2121–2129.

Ruiying Geng, Binhua Li, Yongbin Li, Yuxiao Ye, Ping Jian, and Jian Sun. 2019. Few-shot text classification with induction network. *arXiv*, abs/1902.10482.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Jian Hu, Gang Wang, Frederick H. Lochovsky, Jian-Tao Sun, and Zheng Chen. 2009. Understanding user's query intent with wikipedia. In *International Conference on World Wide Web (WWW)*, pages 471–480.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Anjishnu Kumar, Pavankumar Reddy Muddireddy, Markus Dreyer, and Björn Hoffmeister. 2017. Zero-shot learning across heterogeneous overlapping domains. In *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pages 2914–2918.

Hugo Larochelle, Dumitru Erhan, and Yoshua Bengio. 2008. Zero-data learning of new tasks. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 646–651.

Shen Li, Zhe Zhao, Renfen Hu, Wensi Li, Tao Liu, and Xiaoyong Du. 2018. Analogical reasoning on chinese morphological and semantic relations. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 138–143.

Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. In *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pages 685–689.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations (ICLR)*.

Jinseok Nam, Eneldo Loza Menc'ia, and Johannes Fürnkranz. 2016. All-in text: Learning document, label, and word representations jointly. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 1948–1954.

Mark Palatucci, Dean Pomerleau, Geoffrey E. Hinton, and Tom M. Mitchell. 2009. Zero-shot learning with semantic output codes. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1410–1418.

Suman Ravuri and Andreas Stolcke. 2015. Recurrent neural network and lstm models for lexical utterance classification. In *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pages 135–139.

Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. 2017. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3859–3869.

Prateek Veeranna Sappadla, Jinseok Nam, Eneldo Loza Menc'ia, and Johannes Fürnkranz. 2016. Using semantic similarity for multi-label zero-shot classification of text documents. In *European Symposium on Artificial Neural Networks (ESANN)*.

Lütfi Kerem Şenel, Ihsan Utlu, Veysel Yücesoy, Aykut Koc, and Tolga Cukur. 2018. Semantic structure and interpretability of word embeddings. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(10):1769–1779.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018. Disan: Directional self-attention network for rnn/cnn-free language understanding. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 5446–5455.

Richard Socher, Milind Ganjoo, Christopher D. Manning, and Andrew Y. Ng. 2013. Zero-shot learning through cross-modal transfer. In *Advances in Neural Information Processing Systems (NIPS)*, pages 935–943.

Gokhan Tur, Li Deng, Dilek Hakkani-Tr, and Xiaodong He. 2012. Towards deeper understanding: Deep convex networks for semantic utterance classification. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5045–5048.

Congying Xia, Chenwei Zhang, Xiaohui Yan, Yi Chang, and Philip S. Yu. 2018. Zero-shot user intent detection via capsule neural networks. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3090–3099.

Yongqin Xian, Zeynep Akata, Gaurav Sharma, Quynh N. Nguyen, Matthias Hein, and Bernt Schiele. 2016. Latent embeddings for zero-shot classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 69–77.

Puyang Xu and Ruhi Sarikaya. 2013. Convolutional neural network based triangular CRF for joint intent detection and slot filling. In *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU Workshop)*, pages 78–83.

Min Yang, Wei Zhao, Jianbo Ye, Zeyang Lei, Zhou Zhao, and Soufei Zhang. 2018. Investigating capsule networks with dynamic routing for text classification. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3110–3119.

Majid Yazdani and James Henderson. 2015. A model of zero-shot learning of spoken language understanding. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 244–249.

Yiming Ying and Peng Li. 2012. Distance metric learning with eigenvalue optimization. *Journal of Machine Learning Research*, 13:1–26.

Chenwei Zhang, Wei Fan, Nan Du, and Philip S. Yu. 2016. Mining user intentions from medical queries: A neural network based heterogeneous jointly modeling approach. In *International Conference on World Wide Web (WWW)*, pages 1373–1384.

Jingqing Zhang, Piyawat Lertvittayakumjorn, and Yike Guo. 2019. Integrating semantic knowledge to tackle zero-shot text classification. *arXiv*, abs/1903.12626.

Weinan Zhang, Zhigang Chen, Wanxiang Che, Guoping Hu, and Ting Liu. 2017. The first evaluation of chinese human-computer dialogue technology. *arXiv*, abs/1709.10217.

# Domain Adaptation for Person-Job Fit with Transferable Deep Global Match Network

**Shuqing Bian**[1,2], **Wayne Xin Zhao**[*1,2], **Yang Song**[3], **Tao Zhang**[3] and **Ji-Rong Wen**[1,2]
[1]School of Information, Renmin University of China
[2]Beijing Key Laboratory of Big Data Management and Analysis Methods
[3]BOSS Zhipin
{bianshuqing,jrwen}@ruc.edu.cn batmanfly@gmail.com
{songyang,kylen.zhang}@kanzhun.com

## Abstract

Person-job fit has been an important task which aims to automatically match job positions with suitable candidates. Previous methods mainly focus on solving the match task in single-domain setting, which may not work well when labeled data is limited. We study the domain adaptation problem for person-job fit. We first propose a deep global match network for capturing the global semantic interactions between two sentences from a job posting and a candidate resume respectively. Furthermore, we extend the match network and implement domain adaptation in three levels, *i.e.,* sentence-level representation, sentence-level match, and global match. Extensive experiment results on a large real-world dataset consisting of six domains have demonstrated the effectiveness of the proposed model, especially when there is not sufficient labeled data.

## 1 Introduction

Recent years have witnessed the rapid growth of online recruitment platforms. According to (Iqbal, 2019), there were 590 million users and 11 million job listings in LinkedIn[1] from about over 200 countries and territories all over the world. With the increasing amount of online recruitment data, it has become an essential task that is able to automatically match jobs with suitable candidates, called *person-job fit* (Qin et al., 2018; Shen et al., 2018).

Due to the importance of person-job fit, many efforts have been devoted to improving the match algorithms (Xu et al., 2018; Malinowski et al., 2006; Paparrizos et al., 2011; Lee and Brusilovsky, 2007; Zhang et al., 2014). Among these studies, a typical approach is to casting the task as a supervised text match problem. Given a set of labeled data (*i.e.,* person-job match records), it aims to predict the match label based on the text content of job postings and candidate resumes. More recently, deep learning has shed light on person-job fit methods by learning more effective text representations or text match models (Qin et al., 2018; Zhu et al., 2018; Jiang et al., 2019).

Although these methods have achieved significant progress in person-job fit, they rely on a considerable amount of labeled data to learn the match models. When labeled data is insufficient, existing methods may not work well as expected. Therefore, the deficiency in labeled data has raised a significant challenge to the person-job fit algorithms. The availability of labeled data for different job categories is highly varying. A job position from a major category usually attracts hundreds of job applicants, while one from a minor category receives very few applications. Besides, due to the emergence of new positions, data accumulation under these new job categories is limited. For this purpose, we investigate domain adaptation for person-job fit, focusing on the text match between job postings and candidate resumes. We aim to utilize the acquired knowledge and information from a source domain with sufficient labeled data to improve the prediction performance in a target domain with limited or little labeled data.

Text-based domain adaptation has been extensively studied in the literature, such as text classification (Li et al., 2018; Glorot et al., 2011; Ziser and Reichart, 2017). However, these works mainly focus on how to model an individual document instead of the match for a document pair, which are not directly applicable to our task. As for existing person-job fit methods (Qin et al., 2018), they mainly learn an overall representation for both a job posting and a candidate resume. Then,

*Corresponding author
[1]https://www.linkedin.com



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4810–4820, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

the match score is measured via the similarity between the two overall representations, which is limited to extend for domain adaptation. A major reason is that job postings and resumes are written on top of basic semantic units (*e.g.,* sentences) related to skills, abilities or experiences, we have to model the global interaction between fine-grained semantic units for an accurate match. Such global match information is important to consider for effective domain adaptation.

To address these difficulties, we first design a deep global match network for modeling person-job fit in a single domain. This model can capture the comprehensive interaction information based on all pairwise sentence interactions given a job posting and a resume. Furthermore, we extend the match network for domain adaptation in three levels. First, to overcome the semantic gap or language variation across different domains, the sentence representations have been enhanced via structural correspondence learning. The derived sentence representations are more transferable across domains. Second, the sentence-level match function has been set to share parameters across domains for improving the match accuracy in target domains. We construct a match matrix to contain all the match results in sentence level. Third, we design a convolution based component to learn transferable match patterns and components across domains. Although different domains have varying semantic representations or match functions, the global match information reflected in the match matrix should share similar patterns.

To our knowledge, it is the first time that domain adaptation has been studied for the person-job fit task. To evaluate our proposed model, we construct a large real-world dataset containing two large and four small domains from an online recruitment platform. Extensive experiment results have demonstrated the effectiveness of the proposed model, especially when the labeled data is limited.

## 2 Related Work

As an important task in recruitment data mining (Li et al., 2017a; Xu et al., 2018; Oentaryo et al., 2018), person-job fit has been extensively studied in the literature. Early methods include treating person-job fit as a job/candidate recommendation problem (Lu et al., 2013; Diaby et al., 2013) and extending users' historical behavior by job application records for better match (Hong et al., 2013). Recently, deep learning methods have been utilized to design a hierarchical representation structure based on historical recruitment records (Qin et al., 2018; Zhu et al., 2018; Ramanath et al., 2018).

Domain adaptation is a hot topic in natural language processing, which has received much attention over the decades. Among the early works, Structural Correspondence Learning (SCL) (Blitzer et al., 2007, 2006) is a classic algorithm which learns transferable feature representations across domains. With the revival of deep learning, several studies focus on extending the neural network models for domain adaptation (Yu and Jiang, 2016; Ziser and Reichart, 2017). To further eliminate the distribution differences between source and target domains, the adversarial mechanism has been incorporated into domain adaptation (Li et al., 2017b, 2018; Ganin et al., 2016).

Besides, our work is also related to semantic text match (Mueller and Thyagarajan, 2016; Yin et al., 2016; Wang et al., 2017), specially the work which casts text match as image recognization (Pang et al., 2016) or captures both global and local interactions (Dai et al., 2018; Rao et al., 2018; Yang et al., 2018).

Our work is highly based on these related studies. To our knowledge, domain adaptation for person-job fit has seldom been studied before. We focus on how to transfer the match information between job postings and resumes across domains.

## 3 Problem Formulation

We assume text content of a job posting and a candidate resume are available as input. A job posting $p$ is described as $n_p$ sentences $\{\boldsymbol{w}_i^{(p)}\}_{i=1}^{n_p}$, and the $i$-th sentence $\boldsymbol{w}_i^{(p)}$ is a sequence of $n_{p,i}$ word tokens denoted by $\{w_{i,k}^{(p)}\}_{k=1}^{n_{p,i}}$, where $w_{i,k}^{(p)}$ denotes the $k$-th word of the $i$-th sentence in $p$. Each job posting sentence typically describes the skill or ability that is required for the position. Similarly, a resume $r$ consists of $n_r$ sentences $\{\boldsymbol{w}_i^{(r)}\}_{i=1}^{n_r}$, where $\boldsymbol{w}_i^{(r)}$ is a sequence of $n_{r,i}$ word tokens denoted by $\{w_{i,k}^{(r)}\}_{k=1}^{n_{r,i}}$. Each resume sentence typically describes the skills or experiences that a candidate has acquired. Let a binary label $y$ denote the match result for a person-job pair based on the text of job posting $p$ and

| Symbols | Descriptions |
|---|---|
| $p, r$ | the job posting and resume |
| $n_p, n_r$ | the number of sentences in the $p$ and $r$ |
| $\boldsymbol{w}_i^{(p)}, \boldsymbol{w}_i^{(r)}$ | the $i$-th sentence in the $p$ and $r$ |
| $\{w_{i,k}^{(p)}\}, \{w_{i,k}^{(r)}\}$ | the $k$-th word of the $i$-th sentence in $p$ and $r$ |
| $y$ | the match result for a person-job pair based on the $p$ and $r$ |
| $\mathcal{D}^t, \mathcal{D}^s$ | the training set for the target domain $t$ and source domain $s$ |
| $n_t, n_s$ | the number of target labeled instances and source labeled instances |
| $\boldsymbol{h}_{i,k}^{(p)}, \boldsymbol{h}_{i,k}^{(p)}$ | the hidden states learned from BiGRU network for the $k$-th word of the $i$-th sentence in the $p$ and $r$ |
| $L_P, L_R$ | the dimension size of hidden states for the $p$ and $r$ |
| $\mathbf{M}$ | the match matrix ($\in \mathbb{R}^{n_p \times n_r}$) for the sentence similarity between the $p$ and $r$ |
| $\mathbf{W}$ | the weight matrix ($\in \mathbb{R}^{K \times K}$) , $K$ is the dimension size |
| $\boldsymbol{m}^{p,r}$ | the match representation derived by stacked convolutional layers with max-pooling layers |
| $\Theta$ | all the involved parameters |
| $\boldsymbol{h}^p, \boldsymbol{h}^r$ | the representations of the $p$ and $r$ |
| $\hat{y}$ | the output score calculated by the Multi-Layer Perceptron |
| $\boldsymbol{x}_{i,\text{SCL}}^p, \boldsymbol{x}_{i,\text{SCL}}^r$ | the transferable representation transformed by the original feature in the $p$ and $r$ |
| $\widetilde{\boldsymbol{h}_i^p}, \widetilde{\boldsymbol{h}_i^r}$ | the enhanced job posting and resume sentence representation |
| $\mathbf{A}$ | the shared weight matrix by all the domains |
| $\mathbf{B}^d$ | the specific weight matrix to the domain $d$ |
| $\hat{\Theta}^s$ | the learned parameters for the auxiliary domain $s$ |
| $\hat{\Theta}^t$ | the domain-specific parameters for the target domain $t$ |
| $\widetilde{\boldsymbol{m}^{p,r}}$ | the new representation by the concatenation of the output from both components with the parameters $\hat{\Theta}^s$ and $\hat{\Theta}^t$ |

Table 1. The notations used in this paper.

candidate resume $r$.

Following (Qin et al., 2018; Zhu et al., 2018), we cast the person-job fit problem into a text match task based on the text information of job postings and candidate resumes. The *Person-Job Fit* task aims to learn a match function that is able to predict the actual label for a new job-person pair from the target domain.

Furthermore, we consider a domain adaptation setting for such an match task. Given a target domain $t$, a training set $\mathcal{D}^t = \{\langle p_j^t, r_j^t, y_j\rangle\}_{j=1}^{n_t}$ of $n_t$ labeled instances is given. Considering the target domain to be new or minor, there is very little labeled data. We assume an auxiliary training set $\mathcal{D}^s = \{\langle p_j^s, r_j^s, y_j\rangle\}_{j=1}^{n_s}$ from another source domain $s$ is given, consisting of $n_s$ labeled instances. The source domain (or auxiliary domain) is selected from an existing domain with sufficient match records. In this case, we have $n_t \ll n_s$. For simplicity, we will drop the domain index unless needed, which is the target domain by default. The notations used in this paper are summarized in Table 1.

# 4 The Proposed Model

In this section, we first present a deep global match network for person-job fit using only target domain data. Then, we extend the match network by leveraging the labeled data of an auxiliary domain.

## 4.1 A Deep Global Match Network for Single-Domain Person-Job Fit

Previous studies (Shen et al., 2018; Zhu et al., 2018) mainly focus on modeling the overall interaction between the representations of a job and a candidate. They cannot explicitly capture global match information in terms of fine-grained semantic units. In this section, we propose a deep global match network which is able to characterize global sentence-level interaction for semantic match between a job posting and a candidate resume.

**Hierarchical Attention-based RNN Encoder**. First, we employ the bi-directional recurrent neural networks with gated recurrent unit (BiGRU) to model both sentences and documents (*i.e.,* a job posting or a resume).

Formally, let $\overleftarrow{\boldsymbol{h}_{i,k}^{(p)}}$ and $\overrightarrow{\boldsymbol{h}_{i,k}^{(p)}}$ denote the forward and backward states learned from the BiGRU network for the $k$-th word of the $i$-th sentence in job posting $p$. We concatenate the two directional representations as the representation of a word, *i.e.,* $\boldsymbol{h}_{i,k}^{(p)} = [\overleftarrow{\boldsymbol{h}_{i,k}^{(p)}}, \overrightarrow{\boldsymbol{h}_{i,k}^{(p)}}]$. Since some words in a sentence are important than others, we apply the attention mechanism (Qin et al., 2018) to derive the sentence representation $\boldsymbol{h}_i^{(p)} \in \mathbb{R}^{L_P}$ in a job posting:

$$\boldsymbol{h}_i^{(p)} = \sum_{k=1}^{n_{p,i}} \alpha_k \cdot \boldsymbol{h}_{i,k}^{(p)}, \tag{1}$$

where the attention weight $\alpha_k$ is calculated as

$$\begin{aligned} \alpha_k &= \text{softmax}(a_k), \\ a_k &= \boldsymbol{v}_1^\top \cdot \tanh(\mathbf{W}_1 \boldsymbol{h}_{i,k}^{(p)} + \boldsymbol{b}_1). \end{aligned} \tag{2}$$

Similarly, we can derive the sentence representations for a resume, denoted by $\boldsymbol{h}_i^{(r)} \in \mathbb{R}^{L_R}$. Based on the learned sentence representations, we apply the similar attentional BiGRU network to derive the overall representations for a job posting and a

resume, denoted by $\boldsymbol{h}^p$ and $\boldsymbol{h}^r$:

$$\boldsymbol{h}^{(p)} = \sum_{i=1}^{n_p} \beta_i^{(p)} \cdot \boldsymbol{h}_i^{(p)}, \quad (3)$$

$$\boldsymbol{h}^{(r)} = \sum_{i=1}^{n_p} \beta_i^{(r)} \cdot \boldsymbol{h}_i^{(r)}, \quad (4)$$

where $\beta_i^{(p)}$ and $\beta_i^{(r)}$ are the attention weights defined in a similar way as Eq. 2.

**Global Match Representation**. After encoding the sentences, we further model global semantic interactions between sentence representations of a job and a resume. In specific, we compute the similarities between each sentence in a job posting and each sentence in a candidate resume. Assume that a job posting $p$ has $n_p$ sentences and a resume $r$ has $n_r$ sentences. We can derive a match matrix $\mathbf{M} \in \mathbb{R}^{n_p \times n_r}$ for modeling the global semantic interactions. Formally, to calculate the sentence similarity in $\mathbf{M}$, we apply a linear form as

$$M_{i,i'} = \boldsymbol{h}_i^{(p)\top} \cdot \mathbf{W} \cdot \boldsymbol{h}_{i'}^{(r)}, \quad (5)$$

where $\mathbf{W} \in \mathbb{R}^{K \times K}$ is the match matrix, $\boldsymbol{h}_i^{(p)}$ is the representation of the $i$-th sentence for a job posting $p$, and $\boldsymbol{h}_{i'}^{(r)}$ is the representation of the $i'$-th sentence for resume $r$. Inspired by the recent work on image-based text match models (Pang et al., 2016), we propose to use the convolution based method to model the match information. Formally, we stack two 2-D convolutional layers with two 2-D max-pooling layers in an interleaving way, and derive a match representation $\boldsymbol{m}^{p,r}$ as

$$\boldsymbol{m}^{p,r} = \text{ConvNet}(\mathbf{M}; \Theta), \quad (6)$$

where $\boldsymbol{m}^{p,r}$ summarizes the match information from global sentence interactions, and "$\Theta$" denote all the involved parameters.

**Predicting the Match Label**. With the learned match representation $\boldsymbol{m}^{p,r}$, we concatenate it with the representations of the job posting and the resume as the input for the predictor,

$$\hat{y} = \text{MLP}([\boldsymbol{m}^{p,r}; \boldsymbol{h}^p; \boldsymbol{h}^r]), \quad (7)$$

where we $\boldsymbol{h}^p$ and $\boldsymbol{h}^r$ are the representations of job posting $p$ and resume $r$ defined in Eq. 3 and Eq. 4 respectively, and $\text{MLP}(\cdot)$ is the Multi-Layer Perceptron with a nonlinear layer and a sigmoid layer. Compare with previous work (Qin et al., 2018; Zhu et al., 2018), a major novelty of this match model lies in the fact it has explicitly modeled the match information between a job posting and a resume. It is able to model sentence-level interactions between two text documents.

### 4.2 Domain Adaptation by Transferring Local and Global Match Information

In this part, we study how to extend the proposed deep global match network for domain adaptation. Instead of directly enriching the overall representation of a training instance, we seek a way to transfer local and global match information.

**Enhanced Sentence Representation with Structural Correspondence.** Different domains may have significant language variation or semantic gap, which makes it difficult to derive transferable sentence representations across domains. Inspired by the classic SCL algorithm (Blitzer et al., 2007), we propose to model the structural correspondence of sentences in different domains by using pivot keywords. Consider an example of two snippets from two different job domains:

$S_1$: *grasp* C programming skills
$S_2$: *grasp* picture editing skills

Although the semantics of computer programming and picture editing are very different, they are aligned skill requirements for the two domains via the pivot word *grasp*. By pre-selecting a number of high-quality pivot words, SCL algorithm is able to learn such semantic alignment via large-scale co-occurrence data. In specific, the SCL algorithm is able to learn a mapping function $f_{\text{SCL}}(\cdot)$ that transforms an original into a more transferable representation:

$$\boldsymbol{x}_{i,\text{SCL}}^p = f_{\text{SCL}}(\boldsymbol{h}_i^p). \quad (8)$$

For the $i$-th sentence of a job posting, we derive its representation by concatenating the original representation in Eq. 1 and the transformed representation using SCL algorithm:

$$\widetilde{\boldsymbol{h}_i^p} = \boldsymbol{h}_i^p \oplus \boldsymbol{x}_{i,\text{SCL}}^p, \quad (9)$$

where "$\oplus$" is the vector concatenation operation. Similarly, we can obtain the enhanced resume sentence representation, denoted by $\widetilde{\boldsymbol{h}_i^r}$. With the $\widetilde{\boldsymbol{h}_i^p}$ and $\widetilde{\boldsymbol{h}_i^r}$, we can update the overall representations of $\boldsymbol{h}^p$ and $\boldsymbol{h}^r$ in Eq. 3 and 4.

**Transferring Sentence-level Match Information**. After obtaining the new sentence representations, we study how to transfer the parameters

for modeling sentence-pair match. As shown in Eq. 5, we incorporate a transformation matrix $\mathbf{W}$ to compute sentence similarities. A simple transfer strategy is to share the whole matrix in both target and source domains. However, such a method will make it less flexible to capture domain-specific match information, since the local match information is likely to be varied across domains. Hence, we propose to factorize the matrix $\mathbf{W}$ into a product of two smaller matrices $\mathbf{A} \in \mathbb{R}^{l \times K}$ and $\mathbf{B}^d \in \mathbb{R}^{l \times K}$:

$$\mathbf{W}^d = \mathbf{A}^\top \cdot \mathbf{B}^d, \tag{10}$$

where $\mathbf{A}$ is shared by all the domains, while $\mathbf{B}^d$ is specific to domain $d$. In this factorization, we share the parameters across domains and meanwhile set domain-specific parameters to better capture domain-specific information.

**Transferring Global Match Information**. As shown in Eq. 5, we have constructed a match matrix consisting of pairwise sentence similarities between a job posting and a resume. Our idea is that although sentence-level representation or match is different across domains, global semantic interactions are likely to show the similar patterns. For example, the majority of required skills or abilities in the job posting should be well covered for a good candidate, corresponding to large entries in the match matrix $\mathbf{M}$. Based on this idea, we propose to transfer the parameters of the convolution module across domains. In specific, we first use the rich training data of the auxiliary domain $s$ to train the parameters in Eq. 6, and obtain the learned parameters $\hat{\Theta}^s$. We assume the mapping process from the match matrix to the final match representation (*i.e.,* $\mathbf{M} \rightarrow \boldsymbol{m}^{p,r}$) is transferable. Hence, we directly reuse $\hat{\Theta}^s$ trained from the source domain. For enhancing the modeling capacity, we also train a domain-specific convolution component with the parameters $\hat{\Theta}^t$. For the target domain, we derive the new match representation by using the concatenation of the output from both components with the parameters $\hat{\Theta}^s$ and $\hat{\Theta}^t$:

$$\widetilde{\boldsymbol{m}^{p,r}} = \text{ConvNet}(\mathbf{M}; \hat{\Theta}^s) \oplus \text{ConvNet}(\mathbf{M}; \hat{\Theta}^t). \tag{11}$$

### 4.3 The Final Model and Training

To predict the match label for domain adaptation, we replace $\boldsymbol{h}^p$, $\boldsymbol{h}^r$ and $\boldsymbol{m}^{p,r}$ in Eq. 7 with $\widetilde{\boldsymbol{h}^p}$, $\widetilde{\boldsymbol{h}^r}$, and $\widetilde{\boldsymbol{m}^{p,r}}$ respectively. We present an overview sketch of the proposed model in Fig. 1. We have highlighted the three points for transferring information across domain. Different from previous text-based domain adaption methods, we focus on the transfer of match information, including both local and global semantic interactions.

To optimize our model, we adopt the the binary cross-entropy loss over the entire training data as the total loss. To learn the model parameters, we adopt the Adam optimizer (Kingma and Ba, 2014). In order to avoid overfitting, we adopt the dropout strategy with a rate of 0.1. More parameter settings can be found in Table 2.

| Modules | Settings |
|---|---|
| HA-RNN | $L_R = L_P = 200$, #GRU-layer=2, batch-size=512,init.-learning-rate=0.001 |
| SCL | #pivots = 200, init.-learning-rate=0.01 |
| $\mathbf{W}, \mathbf{A}, \mathbf{B}^d$ | $K = 100, l = 50$ |
| Global Matching | kernel sizes = $\{4 \times 4, 6 \times 6\}$, init.-learning-rate=0.001, batch-size=512 |

Table 2. Parameter settings of our model.

| Domain | #instances | #ASPJ | #ASPR |
|---|---|---|---|
| Technology | 60,000 | 14.0 | 12.3 |
| Sales | 75,000 | 12.0 | 8.2 |
| Design | 3,000 | 14.2 | 10.5 |
| Product | 3,000 | 15.6 | 9.1 |
| Car | 3,000 | 12.7 | 7.9 |
| Journey | 3,000 | 9.5 | 7.0 |

Table 3. Statistics of the datasets. #ASPJ means the average number of sentences per job posting, and #ASPR means the average number of sentences per resume.

## 5 Experiments

In this section, we first set up the experiments, and then report the results and analysis.

### 5.1 Experimental Setup

We evaluate our model on a large real-world dataset provided by the largest online recruiting platform named "BOSS Zhipin" (the BOSS Recruiting)[2] in China. To protect the privacy of candidates, all the records have been anonymized by deleting identity information. Since the original amount of recruitment data is huge, we randomly sample a fraction of the entire data, containing job postings and resumes within a time period of six months. Specifically, the dataset has covered six job domains: two major domains have a large number person-job pairs and four

[2]https://www.zhipin.com

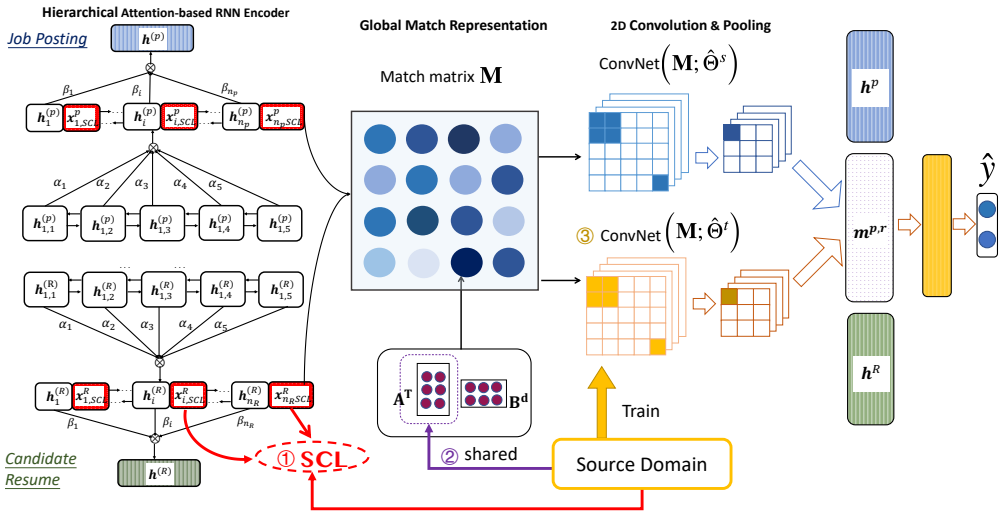


Figure 1. The overview of the transferable deep global match network.

| Methods | Technology | | Sales | | Design | | Cars | | Product | | Journey | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ACC | F1 | ACC | F1 | ACC | F1 | ACC | F1 | ACC | F1 | ACC | F1 |
| DSSM | 0.690 | 0.625 | 0.686 | 0.619 | 0.638 | 0.577 | 0.621 | 0.558 | 0.625 | 0.566 | 0.616 | 0.557 |
| PJFNN | 0.685 | 0.623 | 0.678 | 0.617 | 0.633 | 0.566 | 0.618 | 0.555 | 0.619 | 0.561 | 0.608 | 0.546 |
| BPJFNN | 0.681 | 0.619 | 0.669 | 0.611 | 0.627 | 0.565 | 0.611 | 0.554 | 0.610 | 0.552 | 0.602 | 0.545 |
| APJFNN | 0.694 | 0.630 | 0.688 | 0.626 | 0.641 | 0.580 | 0.627 | 0.563 | 0.630 | 0.571 | 0.619 | 0.559 |
| Our model | **0.713** | **0.642** | **0.709** | **0.638** | **0.659** | **0.600** | **0.644** | **0.582** | **0.648** | **0.591** | **0.637** | **0.575** |

Table 4. Performance comparisons on single domain evaluation for person-job fit.

minor domains have a limited number of person-job pairs. The match label information is obtained according to the acceptance status of the candidates, provided by the online recruiting platform. In our dataset, the ratio between matched and unmatched instances is 1:1[3]. We perform the basic preprocessing steps on the text data, including tokenization and stopword removal. The statistics of the dataset are summarized in Table 3. Our code and data are available at https://github.com/RUCAIBox/Person-Job-Fit.

We consider two evaluation settings, namely single domain evaluation and domain adaptation evaluation. For all domains, we first split the entire data set with a ratio of 1:1 into a training set and a test set. Single domain evaluation examines the performance of a model model trained with domain-specific training data respectively for each of all six domains, while domain adaptation evaluation examines the performance of a model trained with the data from both the source and target domains. For domain adaptation, we consider the four minor domains as target domains, and the major domains as source domains.

Since our task is casted as a classification task, we adopt four commonly used evaluation metrics, including Accuracy, Precision, Recall and F1.

## 5.2 Comparison Methods

**Single Domain**. For single domain evaluation, we consider three latest methods as baselines:

• DSSM (Huang et al., 2013): it utilizes a deep neural network (DNN) to map high-dimensional sparse features into low-dimensional dense features, and calculates the semantic similarity of the text pair.

• BPJFNN (Qin et al., 2018): It applies BiLSTMs to obtain the semantic representation of each word in job postings and resumes, and considers the text content as a single sequence.

• PJFNN (Zhu et al., 2018): It proposes a bipartite Convolutional Neural Networks that can effectively learn the joint representation of Person-Job fitness from historical job applications.

• APJFNN (Qin et al., 2018): It learns a word-level semantic representation for both job requirements and resumes based on RNN and four

[3]There were more negative instances in original data. we randomly sample an equal number of negative instances to reduce the data bias.

hierarchical ability-aware attention strategies.

**Domain Adaptation**. For domain adaptation evaluation, we consider three baseline methods:

• SCL-MI (Blitzer et al., 2007): It proposes the SCL algorithm where pivot features are seleced based on mutual information in the unlabeled data of both the source and target domains.

• AE-SCL-SR (Ziser and Reichart, 2017): It proposes to find a shared low dimensional representation in order to overcome the domain adaptation problem.

• ASP-MTL (Liu et al., 2017): It proposes an adversarial multi-task learning framework, alleviating the shared and private latent feature spaces from interfering with each other.

For our deep match network, we also prepare three simple single-domain variants, include (1) *Tgt-Only* trains the model with only target domain data, (2) *Src-Only* trains the model with only source domain data, and (3) *Mixed* trains the model with a simple mixture of source and target domain data.

### 5.3 Results and Analysis

In this part, we construct a series of experiments on the effectiveness of the proposed model for the person-job fit task.

**Single Domain Results**. We first report the results on single domain evaluation, where we only use the training data from the target domain. In Table 4, we compare our model with three recently proposed methods for person-job fit. Among the three baselines, the APJFNN method performs better than the other two methods. It adopts fine-grained attention mechanism based on skills and abilities. Our (single domain) model outperforms all the baselines. The proposed deep global match network extends the APJFNN by explicitly modeling the global match information for the sentence representations of job postings and resumes. These results indicate the global match information is useful to improve the performance of the person-job fit task.

**Domain Adaptation Results**. We select the domains of *technology* and *sales* as the source domains, since both contain much more training data, while the rest domains are selected as the target domains. Table 5 presents the performance of different comparison methods for domain adaptation. First, for the three variants for our single-domain model, the performance order is *Src-Only* $<$ *Mixed* $<$ *Tgt-Only*. The result indicates that directly incorporating the source domain data as training data may hurt the performance for the target domain. Second, ASP-MTL performs best among the three baselines. It sets up both shared and private feature representations, and apply the adversarial mechanism to alleviate the shared and private latent feature spaces from interfering with each other. Finally, our model outperforms all the baselines with a large margin. Previous methods (*e.g.,* ASP-MTL) learns an overall representation for a document, which cannot characterize global semantic interaction in fine-grained semantic units. As a comparison, our deep global match network is able to transfer information in three levels, namely sentence representation, sentence-level match and global match. Different from existing text-based domain adaption methods, our model is tailored to transfer the match information between a document pair.

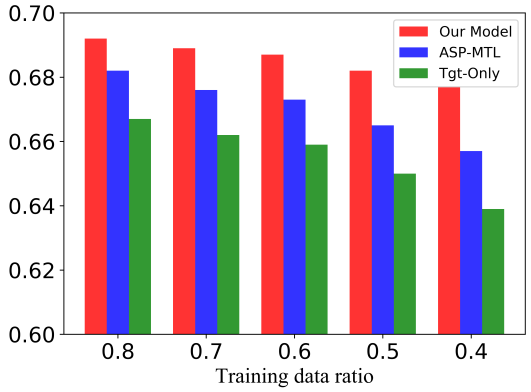


Figure 2. Domain adaptation with different train/test splitting ratios in the target domain for "Technology→Car".

**Ablation Analysis**. Recall we proposed three transfer techniques to improve the performance for domain adaptation in Section 4.2, including sentence representation (denoted by *SR*), sentence-level match (denoted by *SM*) and global match (denoted by *GM*). Now we examine the effect of each factor on the prediction performance. At each time, we remove an individual factor while keep the rest. Due to space limit, we only report the results with the cases "Technology→Cars" and "Sales→Product". As shown in Table 6, all the three factors are useful to improve the performance of our model. Especially, the transferred information in sentence representation

| source domain | methods | DESIGN | | CARS | | PRODUCT | | JOURNEY | |
|---|---|---|---|---|---|---|---|---|---|
| | | ACC | F1 | ACC | F1 | ACC | F1 | ACC | F1 |
| TECHNOLOGY | Src-Only | 0.566 | 0.503 | 0.542 | 0.484 | 0.550 | 0.491 | 0.528 | 0.469 |
| | Tgt-Only | 0.659 | 0.600 | 0.644 | 0.582 | 0.648 | 0.591 | 0.637 | 0.575 |
| | Mixed | 0.653 | 0.599 | 0.636 | 0.580 | 0.641 | 0.582 | 0.629 | 0.574 |
| | SCL-MI | 0.657 | 0.598 | 0.645 | 0.586 | 0.649 | 0.591 | 0.640 | 0.577 |
| | AE-SCL | 0.672 | 0.610 | 0.659 | 0.602 | 0.662 | 0.603 | 0.650 | 0.594 |
| | ASP-MTL | 0.675 | 0.616 | 0.661 | 0.600 | 0.666 | 0.608 | 0.654 | 0.595 |
| | Our Model | **0.696** | **0.638** | **0.680** | **0.625** | **0.687** | **0.627** | **0.669** | **0.610** |
| SALES | Src-Only | 0.547 | 0.485 | 0.535 | 0.474 | 0.562 | 0.603 | 0.539 | 0.484 |
| | Tgt-Only | 0.659 | 0.600 | 0.644 | 0.582 | 0.648 | 0.591 | 0.637 | 0.575 |
| | Mixed | 0.646 | 0.587 | 0.631 | 0.568 | 0.643 | 0.586 | 0.632 | 0.574 |
| | SCL-MI | 0.655 | 0.596 | 0.642 | 0.582 | 0.652 | 0.594 | 0.645 | 0.583 |
| | AE-SCL | 0.665 | 0.606 | 0.654 | 0.597 | 0.666 | 0.608 | 0.657 | 0.598 |
| | ASP-MTL | 0.671 | 0.612 | 0.658 | 0.599 | 0.672 | 0.613 | 0.660 | 0.601 |
| | Our Model | **0.689** | **0.630** | **0.675** | **0.618** | **0.692** | **0.634** | **0.676** | **0.618** |

Table 5. Performance comparisons on domain adaptation evaluation for person-job fit.

| Variants | T→C | | S→P | |
|---|---|---|---|---|
| | ACC | F1 | ACC | F1 |
| Our model | **0.680** | **0.625** | **0.692** | **0.634** |
| w/o SR | 0.659 | 0.601 | 0.671 | 0.614 |
| w/o SM | 0.676 | 0.623 | 0.685 | 0.631 |
| w/o GM | 0.654 | 0.598 | 0.667 | 0.612 |

Table 6. Ablation study on the three factors of our model.

and global match bring more improvement to our model.

**Varying the Amount of Training Data**. An important consideration for domain adaptation is the amount of training data in the target domain. Here, we vary the ratio of training data in the target domain with the case "Technology→Car". We re-split the dataset of the *Car* domain according to different training data ratios. We select the single domain variant of our model and the best baseline ASP-MTL as comparisons. Fig. 2 presents the varying results. With the decreasing of training data, the performance of all the methods drops accordingly. While, domain adaption techniques are useful to alleviate the deficiency of training data, since ASP-MTL and our model yield less performance drop. Especially, the performance of our model is relatively stable, which indicates its robustness for domain adaption.

**Parameters Sensitivity Analysis**. In our model, we have two important parameters, namely the dimension number $K$ of the matrix $\mathbf{W}$ in Eq. 10 and the number of pivot words in the SCL algorithm. We vary the two parameters and present the tuning results in Fig. 3. We select the best baseline ASP-MTL as a comparison. As we can see, the performance of our model is relatively stable *w.r.t.* the varying of the two parameters. A dimension number of 100 and a number of 200 pivot words lead to the best performance.

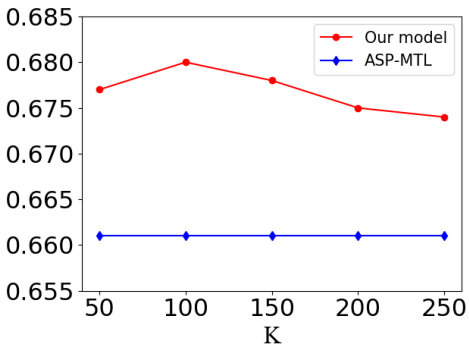


(a) Varying the dimension number $K$.

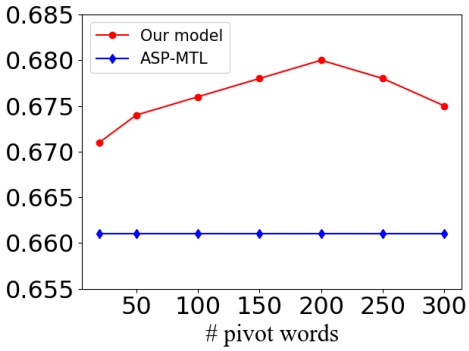


(b) Varying the number of pivot words.

Figure 3. Parameter tuning for our model with the case "Technology→Car".

## 5.4 Qualitative Analysis

A key point of our model is that it can transfer the global match information derived by computing the pairwise similarities between two sentence respectively from a job posting and a resume. Recall that we reuse the convolution-based component

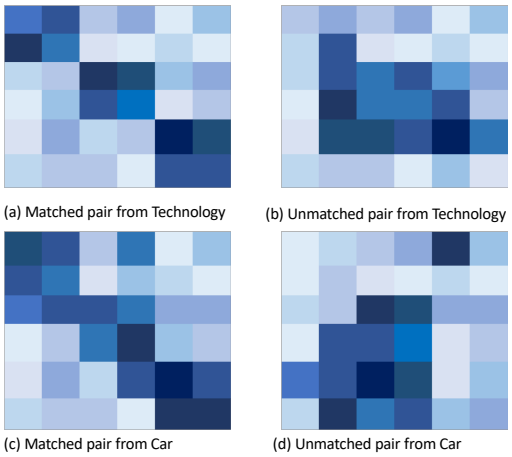


Figure 4. An illustrative example for the effect of the global match in our model. Here, “matched" and “unmatched" indicates the candidate person has been or not been accepted by the job position, respectively.

trained with the source domain data. Now, we study how such a mechanism work and why it is useful to improve the performance.

In Fig. 4, we present four person-job pairs from our dataset containing two matched pairs and two unmatched pairs. In a subfigure, each row corresponds to a sentence in a job posting, while a column corresponds to a sentence in a resume. Due to space limit, we only select six sentences for each document. We follow the method in Eq. 5 to construct the match matrix $\mathbf{M}$. We use the darkness degree to indicate the similarity between two sentences. For ease of understanding, we further perform the row and column permutation to make the large entries distributed in blocks.

From Fig. 4, we can make the following observations. First, global match matrices have a significant difference between a matched and an unmatched case. Although the match matrix for an unmatched pair also contains many large values, they are usually located in off-diagonal entries. Comparing Fig. 4(a) and Fig. 4(c), it is interesting to see that they share a similar distribution pattern, *i.e.,* large values are mainly distributed along the diagonal line and these values are also aggregated in small blocks. Such a phenomenon indicates that the majority of the required skills have been well covered by a candidate in a matched pair. Without considering global semantic interactions, it is difficult to correctly predict the label for the unmatched pairs in Fig. 4(b) and Fig. 4(d), since they are likely to have a large similarity measured by the overall representations.

## 6 Conclusion

This paper studied the domain adaptation for person-job fit. We first proposed a deep global match network for the single-domain match setting. Then, we extended the proposed model for domain adaptation in three aspects. We constructed extensive experiments on a large real-world recruitment dataset, containing six job domains. The results have demonstrated the effectiveness of our model in terms of the prediction accuracy for person-job fit, especially when the training data is limited. Our current characterization with global match information provides a good form to develop interpretable domain adaptation models, *i.e.,* what kind of information has been transferred across domains. As future work, we will work along this line and investigate the design interpretable solutions to domain adaptation for person-job fit. We will also consider how to model the domain relationship for effectively transferring information across domains.

## Acknowledgments

The authors would like to thank the anonymous reviewers for their helpful and constructive comments. This work was partially supported by the National Natural Science Foundation of China under Grant No. 61872369 and 61832017, the Fundamental Research Funds for the Central Universities, the Research Funds of Renmin University of China under Grant No. 18XNLG22 and 19XNQ047.

## References

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th annual meeting of the association of computational linguistics*, pages 440–447.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 120–128. Association for Computational Linguistics.

Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. 2018. Convolutional neural networks for soft-matching n-grams in ad-hoc search. In *Proceedings of the eleventh ACM international*

*conference on web search and data mining*, pages 126–134. ACM.

Mamadou Diaby, Emmanuel Viennet, and Tristan Launay. 2013. Toward the next generation of recruitment tools: an online social network-based job recommender system. In *2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013)*, pages 821–828. IEEE.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, Francois Laviolette, Mario Marchand, and Victor S Lempitsky. 2016. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(1):189–209.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 513–520.

Wenxing Hong, Siting Zheng, and Huan Wang. 2013. Dynamic user profile-based job recommender system. In *2013 8th International Conference on Computer Science & Education*, pages 1499–1503. IEEE.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information and knowledge management*, pages 2333–2338. ACM.

Mansoor Iqbal. 2019. Linkedin usage and revenue statistics 2018. http://www.businessofapps.com/data/linkedin-statistics/. Accessed February 27, 2019.

Jyun-Yu Jiang, Mingyang Zhang, Cheng Li, Mike Bendersky, Nadav Golbandi, and Marc Najork. 2019. Semantic text matching for long-form documents. In *International World Wide Web Conference, WWW*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Danielle H Lee and Peter Brusilovsky. 2007. Fighting information overflow with personalized comprehensive information access: A proactive job recommender. In *In Proceedings of the 3rd International Conference on Autonomic and Autonomous Systems*, pages 21–21. IEEE.

Huayu Li, Yong Ge, Hengshu Zhu, Hui Xiong, and Hongke Zhao. 2017a. Prospecting the career development of talents: A survival analysis perspective. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 917–925. ACM.

Zheng Li, Ying Wei, Yu Zhang, and Qiang Yang. 2018. Hierarchical attention transfer network for cross-domain sentiment classification. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Zheng Li, Yun Zhang, Ying Wei, Yuxiang Wu, and Qiang Yang. 2017b. End-to-end adversarial memory network for cross-domain sentiment classification. In *IJCAI*, pages 2237–2243.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1–10.

Yao Lu, Sandy El Helou, and Denis Gillet. 2013. A recommender system for job seeking and recruiting website. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 963–966. ACM.

Jochen Malinowski, Tobias Keim, Oliver Wendt, and Tim Weitzel. 2006. Matching people and jobs: A bilateral recommendation approach. In *In Proceedings of the 39th Annual Hawaii International Conference on System Sciences, Vol. 6. IEEE*.

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In *In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI 2016)*, pages 2786–2792. AAAI Press.

Richard J Oentaryo, Ee-Peng Lim, Xavier Jayaraj Siddarth Ashok, Philips Kokoh Prasetyo, Koon Han Ong, and Zi Quan Lau. 2018. Talent flow analytics in online professional network. *Data Science and Engineering*, 3(3):199–220.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016. Text matching as image recognition. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Ioannis Paparrizos, B Barla Cambazoglu, and Aristides Gionis. 2011. Machine learned job recommendation. In *In Proceedings of the 5th ACM International Conference on Recommender Systems*, pages 325–328. ACM.

Chuan Qin, Hengshu Zhu, Tong Xu, Chen Zhu, Liang Jiang, Enhong Chen, and Hui Xiong. 2018. Enhancing person-job fit for talent recruitment: An ability-aware neural network approach. In *In Proceedings of the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-2018) , Ann Arbor, Michigan, USA*.

Rohan Ramanath, Hakan Inan, Gungor Polatkan, Bo Hu, Qi Guo, Cagri Ozcaglar, Xianren Wu, Krishnaram Kenthapadi, and Sahin Cem Geyik. 2018. Towards deep and representation learning for talent search at linkedin. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 2253–2261. ACM.

Jinfeng Rao, Wei Yang, Yuhao Zhang, Ferhan Ture, and Jimmy Lin. 2018. Multi-perspective relevance matching with hierarchical convnets for social media search. In *The 32nd AAAI Conference on Artificial Intelligence (AAAI-2018) , New Orleans, LA, USA, 2018*, pages 7–11.

Dazhong Shen, Hengshu Zhu, Chen Zhu, Tong Xu, Chao Ma, and Hui Xiong. 2018. A joint learning approach to intelligent job interview assessment. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI-2018) , Stocholm, Sweden, 2018*.

Chenglong Wang, Feijun Jiang, and Hongxia Yang. 2017. A hybrid framework for text modeling with convolutional rnn. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 2061–2069. ACM.

Tong Xu, Hengshu Zhu, Chen Zhu, Pan Li, and Hui Xiong. 2018. Measuring the popularity of job skills in recruitment market: A multi-criteria approach. In *The 32nd AAAI Conference on Artificial Intelligence (AAAI-2018) , New Orleans, LA, USA, 2018*.

Zhilin Yang, Jake Zhao, Bhuwan Dhingra, Kaiming He, William W. Cohen, Ruslan R. Salakhutdinov, and Yann LeCun. 2018. Glomo: Unsupervisedly learned relational graphs as transferable representations. In *Proceedings of the 32nd Annual Conference on Neural Information Processing Systems (NIPS)*, pages 8950–8961.

Wenpeng Yin, Hinrich Schutze, Bing Xiang, and Bowen Zhou. 2016. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics*, 4(1):259–272.

Jianfei Yu and Jing Jiang. 2016. Learning sentence embeddings with auxiliary tasks for cross-domain sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 236–246.

Yingya Zhang, Cheng Yang, and Zhixiang Niu. 2014. A research of job recommendation system based on collaborative filtering. In *In Proceedings of the 7th International Symposium on Computational Intelligence and Design*, pages 533–538. IEEE.

Chen Zhu, Hengshu Zhu, Hui Xiong, Chao Ma, Fang Xie, Pengliang Ding, and Pan Li. 2018. Person-job fit: Adapting the right talent for the right job with joint representation learning. *ACM Transactions on Management Information Systems ACM TMIS*.

Yftah Ziser and Roi Reichart. 2017. Neural structural correspondence learning for domain adaptation. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 400–410.

# Heterogeneous Graph Attention Networks for Semi-supervised Short Text Classification

**Linmei Hu[1], Tianchi Yang[1], Chuan Shi[*1], Houye Ji[1], Xiaoli Li[2]**
[1]Beijing University of Posts and Telecommunications, China
[2]Institute for Infocomm Research, Singapore
{hulinmei,yangtianchi,shichuan,jhy1993}@bupt.edu.cn
xlli@i2r.a-star.edu.sg

## Abstract

Short text classification has found rich and critical applications in news and tweet tagging to help users find relevant information. Due to lack of labeled training data in many practical use cases, there is a pressing need for studying semi-supervised short text classification. Most existing studies focus on long texts and achieve unsatisfactory performance on short texts due to the sparsity and limited labeled data. In this paper, we propose a novel heterogeneous graph neural network based method for semi-supervised short text classification, leveraging full advantage of few labeled data and large unlabeled data through information propagation along the graph. In particular, we first present a flexible HIN (heterogeneous information network) framework for modeling the short texts, which can integrate any type of additional information as well as capture their relations to address the semantic sparsity. Then, we propose **H**eterogeneous **G**raph **AT**tention networks (HGAT) to embed the HIN for short text classification based on a dual-level attention mechanism, including node-level and type-level attentions. The attention mechanism can learn the importance of different neighboring nodes as well as the importance of different node (information) types to a current node. Extensive experimental results have demonstrated that our proposed model outperforms state-of-the-art methods across six benchmark datasets significantly.

## 1 Introduction

With the rapid development of online social media and e-commerce, short texts, such as online news, queries, reviews, tweets, are increasingly widespread on the Internet (Song et al., 2014). Short text classification can be widely applied in many domains, ranging from sentiment analysis to news tagging/categorization and query intent classification (Aggarwal and Zhai, 2012; Meng et al., 2018). In many practical scenarios, the labeled data is scarce, while human labeling is time-consuming and may require expert knowledge (Aggarwal and Zhai, 2012). As a consequence, there is a pressing need for studying semi-supervised short text classification with a relatively small number of labeled training data.

Nevertheless, semi-supervised short text classification is nontrivial due to the following challenges. Firstly, short texts are usually semantically sparse and ambiguous, lacking contexts (Phan et al., 2008). While some methods have been proposed to incorporate additional information such as entities (Wang et al., 2013, 2017), they are unable to consider the relational data such as the semantic relations among entities. Secondly, the labeled training data is limited, which leads to traditional and neural supervised methods (Wang and Manning, 2012; Kim, 2014; Zhang et al., 2015) ineffective. As such, how to make full use of the limited labeled data and large number of unlabeled data has become a key problem for short text classification (Aggarwal and Zhai, 2012). Finally, we need to capture the importance of different information that is incorporated to address sparsity at multiple granularity levels and reduce the weights of noisy information to achieve more accurate classification results.

In this work, we propose a novel *heterogeneous graph neural network based method* for semi-supervised short text classification, which makes full use of both limited labeled data and large unlabeled data by allowing information propagation through our automatically constructed graph. Particularly, we first present a flexible HIN framework for modeling the short texts, which is able to incorporate any additional information (e.g., entities and topics) as well as capture the rich relations among the texts and the additional information. Then, we propose Heterogeneous Graph Attention



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4821–4830,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

networks (HGAT) to embed the HIN for short text classification based on a new dual-level attention mechanism including node-level and type-level attentions. Our HGAT method considers the heterogeneity of different node types. Additionally, the dual-level attention mechanism captures both the importance of different neighboring nodes (reducing the weights of noisy information) and the importance of different node (information) types to a current node. The main contributions of this paper can be summarized as follows:

1) To the best of our knowledge, *this is the first attempt* to model short texts as well as additional information with an HIN and adapt graph neural networks on the HIN for semi-supervised classification.

2) We propose novel *heterogeneous graph attention networks* (HGAT) for the HIN embedding based on a new dual-level attention mechanism which can learn the importance of different neighboring nodes and the importance of different node (information) types to a current node.

3) Extensive experimental results have demonstrated that our proposed HGAT model significantly outperforms seven state-of-the-art methods across six benchmark datasets.

## 2 Related Work

### 2.1 Traditional Text Classification

Traditional text classification methods such as SVM (Drucker et al., 1999) need a feature engineering step for text representation. The most commonly used features are BoW and TF-IDF (Blei et al., 2003). Some recent studies (Rousseau et al., 2015; Wang et al., 2016) model texts as graphs and extract path based features for classification. Despite its initial success on formal and well-edited texts, all these methods fail to achieve satisfactory performance on short text classification, due to the insufficient features incurred by short texts. To address the problem, efforts have been made to enrich the semantics of *short texts*. For example, Phan et al. (2008) extracted the latent topics of the short texts with the help of an external corpus. Wang et al. (2013) introduced external entity information from Knowledge Bases, etc. However, these methods are not able to achieve good performance as the feature engineering step relies on domain knowledge.

### 2.2 Deep neural networks for Text Classification

Deep neural networks which automatically represent texts as embeddings, have been widely used for text classification. Two representative deep neural models, such as RNNs (Liu et al., 2016; Sinha et al., 2018) and CNNs, (Kim, 2014; Shimura et al., 2018) have shown their power in many NLP tasks, including text classification. To adapt it to *short text classification*, several methods have been proposed. For example, Zhang et al. (2015) designs a character-level CNN which alleviates the sparsity by mining different levels of information within the texts. Wang et al. (2017) incorporates the entities and concepts from KBs to enrich the semantics of short texts. However, these methods cannot capture the semantic relations (e.g., entity relations) and rely heavily on the number of training data. Clearly, lacking of training data is still a key bottleneck that prohibits them from successful practical applications.

### 2.3 Semi-supervised Text Classification

Considering the cost of human labeling and the fact that unlabeled texts also provide valuable information, semi-supervised methods have peen proposed. They can be categorized into two classes: (1) latent variable models (Lu and Zhai, 2008; Chen et al., 2015); and (2) embedding-based models (Meng et al., 2018). The former mainly extend topic model by user-provided seed information and then infer the documents' labels based on posterior category-topic assignment. The latter use seed information to derive embeddings for documents and label names for text classification. For example, PTE (Tang et al., 2015) models the documents, words and labels with graphs and learns text (node) embeddings for classification. Meng et al. (2018) leveraged seed information to generate pseudo-labeled documents for pre-training. Yin et al. (2015) used a semi-supervised learning method based on SVM to label the unlabeled documents in an iterative way. Recently, graph convolutional networks (GCN) have received wide attention for semi-supervised classification (Kipf and Welling, 2017). TextGCN (Yao et al., 2019) models the whole text corpus as a document-word graph and applies GCN for classification. However, all these methods focus on long texts. In addition, they fail to use attention mechanisms to capture important information.

## 3 Our Proposed Method

In this paper, we propose a novel heterogeneous graph neural network based method for semi-supervised short text classification, which takes full advantage of both limited labeled data and large unlabeled data by allowing information propagation along the graph. Our method includes two steps. Particularly, to alleviate the sparsity of short texts, we first present a flexible HIN framework for modeling the short texts, which can incorporate any additional information as well as capture the rich relations among the short texts and the added information. Then, we propose a novel model HGAT to embed the HIN for short text classification based on a new dual-level attention mechanism. HGAT considers the heterogeneity of different types of information. In addition, the attention mechanism can learn the importance of different nodes (reducing the weights of noisy information) as well as the importance of different node (information) types.

### 3.1 HIN for Short Texts

We first present the HIN framework for modeling the short texts, which enables integration of any additional information and captures the rich relations among the texts and the added information. In this way, the sparsity of the short texts is alleviated.

Previous studies have exploited latent topics (Zeng et al., 2018) and external knowledge (e.g., entities) from Knowledge Bases to enrich the semantics of the short texts (Wang et al., 2013, 2017). However, they fail to consider the semantic relation information, such as entity relations. Our HIN framework for short texts is flexible for integrating any additional information and modeling their rich relations.

Here, we consider two types of additional information i.e., topics and entities. As shown in Figure 1, we construct the HIN $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ containing the short texts $D = \{d_1, ..., d_m\}$, topics $T = \{t_1, ..., t_K\}$, and entities $E = \{e_1, ..., e_n\}$ as nodes, i.e., $\mathcal{V} = D \cup T \cup E$. The set of edges $\mathcal{E}$ represent their relations. The details of constructing the network are described as follows.

First, we mine the latent topics $T$ to enrich the semantics of short texts using LDA (Blei et al., 2003). Each topic $t_i = (\theta_1, ..., \theta_w)$ ($w$ denotes the vocabulary size) is represented by a probability distribution over the words. We assign each

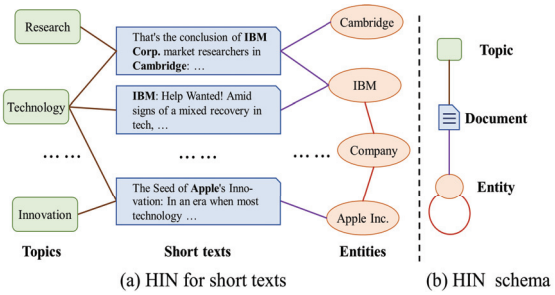


Figure 1: An example of HIN for short texts on AGNews.

document to the top $P$ topics with the largest probabilities. Thus, the edge between a document and a topic is built if the document is assigned to the topic.

Second, we recognize the entities $E$ in the documents $D$ and map them to Wikipedia with the entity linking tool TAGME[1]. The edge between a document and an entity is built if the document contains the entity. We take an entity as a whole word and learn the entity embeddings using word2vec[2] based on the Wikipedia corpus. To further enrich the semantics of short texts and advance the information propagation, we consider the relations between entities. Particularly, if the similarity score (cosine similarity) between two entities, computed based on their embeddings, is above a predefined threshold $\delta$, we build an edge between them.

By incorporating the topics, entities and the relations, we enrich the semantics of the short texts and thus greatly benefit the following classification task. For example, as shown in Figure 1, the short text "the seed of Apple's Innovation: In an era when most technology..." is semantically enriched by the relations with the entities "Apple Inc." and "company", as well as the topic "technology". Thus, it can be correctly classified into the category of "business" with high confidence.

### 3.2 HGAT

We then propose HGAT model (shown in Figure 2 ) to embed the HIN for short text classification based on a new dual-level attention mechanism including node level and type level. HGAT considers the heterogeneity of different types of information with heterogeneous graph convolution. In addition, the dual-level attention mechanism

[1]https://sobigdata.d4science.org/group/tagme/
[2]https://code.google.com/archive/p/word2vec/

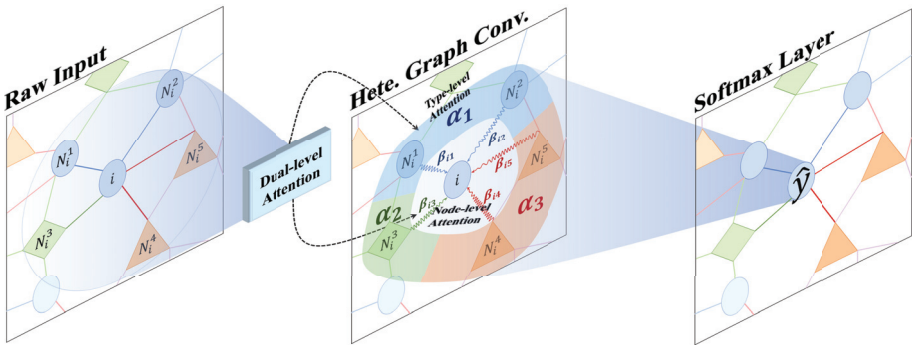


Figure 2: Illustration of our model HGAT.

captures the importance of different neighboring nodes (reducing the weights of noisy information) and the importance of different node (information) types to a specific node. Finally, it predicts the labels of documents through a softmax layer.

#### 3.2.1 Heterogeneous Graph Convolution

We first describe the heterogeneous graph convolution in HGAT, considering the heterogeneous types of nodes (information).

As known, GCN (Kipf and Welling, 2017) is a multi-layer neural network that operates directly on a homogeneous graph and induces the embedding vectors of nodes based on the properties of their neighborhoods. Formally, consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ and $\mathcal{E}$ represent the set of nodes and edges respectively. Let $X \in \mathbb{R}^{|\mathcal{V}| \times q}$ be a matrix containing the nodes with their features $x_v \in \mathbb{R}^q$ (each row $x_v$ is a feature vector for a node $v$). For the graph $\mathcal{G}$, we introduce its adjacency matrix $A' = A + I$ with added self-connections and degree matrix $M$, where $M_{ii} = \sum_j A'_{ij}$. Then the layer-wise propagation rule is as follows:

$$H^{(l+1)} = \sigma(\tilde{A} \cdot H^{(l)} \cdot W^{(l)}). \quad (1)$$

Here, $\tilde{A} = M^{-\frac{1}{2}} A' M^{-\frac{1}{2}}$ represents the symmetric normalized adjacency matrix. $W^{(l)}$ is a layer-specific trainable transformation matrix. $\sigma(\cdot)$ denotes an activation function such as ReLU. $H^{(l)} \in \mathbb{R}^{|\mathcal{V}| \times q}$ denotes the hidden representations of nodes in the $l^{th}$ layer. Initially, $H^{(0)} = X$.

Unfortunately, GCN cannot be directly applied to the HIN for short texts due to the node heterogeneity issue. Specifically, in the HIN, we have three types of nodes: documents, topics and entities with different feature spaces. For a document $d \in D$, we use the TF-IDF vector as its feature vector $x_d$. For the topic $t \in T$, the word distribution is used to represent the topic $x_t = \{\theta_i\}_{i=[1,w]}$. For each entity, to make full use of relevant information, we represent the entity $x_v$ by concatenating its embedding and TF-IDF vector of its Wikipedia description text.

A straightforward way to adapt GCN for the HIN containing different types of nodes $\mathcal{T} = \{\tau_1, \tau_2, \tau_3\}$ is to construct a new large feature space by concatenating together the feature spaces of different types of nodes. For example, each node is denoted as a feature vector with 0 values for the irrelevant dimensions for other types. We name this basic method for adapting GCN to HIN as *GCN-HIN*. However, it suffers from reduced performance since it ignores the heterogeneity of different information types.

To address the issue, we propose the heterogeneous graph convolution, which considers the difference of various types of information and projects them into an implicit common space with their respective transformation matrices.

$$H^{(l+1)} = \sigma(\sum_{\tau \in \mathcal{T}} \tilde{A}_\tau \cdot H_\tau^{(l)} \cdot W_\tau^{(l)}), \quad (2)$$

where $\tilde{A}_\tau \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}_\tau|}$ is the submatrix of $\tilde{A}$, whose rows represent all the nodes and columns represent their neighboring nodes with the type $\tau$. The representation of the nodes $H^{(l+1)}$ is obtained by aggregating information from the features of their neighboring nodes $H_\tau^{(l)}$ with different types $\tau$ using different transformation matrix $W_\tau^{(l)} \in \mathbb{R}^{q^{(l)} \times q^{(l+1)}}$. The transformation matrix $W_\tau^{(l)}$ considers the difference of different feature spaces and projects them into an implicit common space $\mathbb{R}^{q^{(l+1)}}$. Initially, $H_\tau^{(0)} = X_\tau$.

#### 3.2.2 Dual-level Attention Mechanism

Typically, given a specific node, different types of neighboring nodes may have different impacts on it. For example, the neighboring nodes of the same

type may carry more useful information. Additionally, different neighboring nodes of the same type could also have different importance. To capture both the different importance at both node level and type level, we design a new dual-level attention mechanism.

**Type-level Attention.** Given a specific node $v$, the type-level attention learns the weights of different types of neighboring nodes. Specifically, we first represent the embedding of the type $\tau$ as $h_\tau = \sum_{v'} \tilde{A}_{vv'} h_{v'}$, which is the sum of the neighboring node features $h_{v'}$ where the nodes $v' \in \mathcal{N}_v$ and are with the type $\tau$. Then, we calculate the type-level attention scores based on the current node embedding $h_v$ and the type embedding $h_\tau$:

$$a_\tau = \sigma(\mu_\tau^T \cdot [h_v || h_\tau]), \tag{3}$$

where $\mu_\tau$ is the attention vector for the type $\tau$, $||$ means "concatenate", and $\sigma(\cdot)$ denotes the activation function, such as Leaky ReLU.

Then we obtain the type-level attention weights by normalizing the attention scores across all the types with the softmax function:

$$\alpha_\tau = \frac{\exp(a_\tau)}{\sum_{\tau' \in \mathcal{T}} \exp(a_{\tau'})}. \tag{4}$$

**Node-level Attention.** We design the node-level attention to capture the importance of different neighboring nodes and reduce the weights of noisy nodes. Formally, given a specific node $v$ with the type $\tau$ and its neighboring node $v' \in \mathcal{N}_v$ with the type $\tau'$, we compute the node-level attention scores based on the node embeddings $h_v$ and $h_{v'}$ with the type-level attention weight $\alpha_{\tau'}$ for the node $v'$:

$$b_{vv'} = \sigma(\nu^T \cdot \alpha_{\tau'} [h_v || h_{v'}]), \tag{5}$$

where $\nu$ is the attention vector. Then we normalize the node-level attention scores with the softmax function:

$$\beta_{vv'} = \frac{\exp(b_{vv'})}{\sum_{i \in \mathcal{N}_v} \exp(b_{vi})}. \tag{6}$$

Finally, we incorporate the dual-level attention mechanism including type-level and node-level attentions into the heterogeneous graph convolution by replacing Eq. 2 with the following layer-wise propagation rule:

$$H^{(l+1)} = \sigma(\sum_{\tau \in \mathcal{T}} \mathcal{B}_\tau \cdot H_\tau^{(l)} \cdot W_\tau^{(l)}). \tag{7}$$

Here, $\mathcal{B}_\tau$ represents the attention matrix, whose element in the $v^{th}$ row $v'^{th}$ column is $\beta_{vv'}$ in Eq. 6.

### 3.3 Model Training

After going through an $L$-layer HGAT, we can get the embeddings of nodes (including short texts) in the HIN. The short text embeddings $H^{(L)}$ are then fed to a softmax layer for classification. Formlly,

$$Z = \text{softmax}(H^{(L)}), \tag{8}$$

During model training, we exploit the cross-entropy loss over training data with the L2-norm. Formally,

$$\mathcal{L} = -\sum_{i \in D_{\text{train}}} \sum_{j=1}^{C} Y_{ij} \cdot \log Z_{ij} + \eta \, \|\Theta\|_2, \tag{9}$$

where $C$ is the number of classes, $D_{\text{train}}$ is the set of short text indices for training, $Y$ is the corresponding label indicator matrix, $\Theta$ is model parameters, and $\eta$ is regularization factor. For model optimization, we adopt the gradient descent algorithm.

## 4 Experiments

In this section, we evaluate the empirical performance of different methods for semi-supervised short text classification.

### 4.1 Experimental Setup

#### 4.1.1 Datasets

We conduct extensive experiments on 6 benchmark short text datasets: AGNews, Snippets, Ohsumed, TagMyNews, MR and Twitter.

**AGNews**: This dataset is adopted from Zhang et al. (2015). We randomly select 6,000 pieces of news from AGNews, evenly distributed into 4 classes.

**Snippets**: This dataset is released by Phan et al. (2008). It is composed of the snippets returned by a web-search engine.

**Ohsumed**[3]: We use the benchmark bibliographic classification dataset released by Yao et al. (2019), where the documents with multiple labels are removed. We use the titles for short text classification.

**TagMyNews**: We use the news titles as instances from the benchmark classification dataset

[3]http://disi.unitn.it/moschitti/corpora.htm

| | #docs | #tokens | #entities | #classes |
|---|---|---|---|---|
| AGNews | 6,000 | 18.4 | 0.9 (72%) | 4 |
| Snippets | 12,340 | 14.5 | 4.4 (94%) | 8 |
| Ohsumed | 7,400 | 6.8 | 3.1 (96%) | 23 |
| TagMyNews | 32,549 | 5.1 | 1.9 (86%) | 7 |
| MR | 10,662 | 7.6 | 1.8 (76%) | 2 |
| Twitter | 10,000 | 3.5 | 1.1 (63%) | 2 |

Table 1: Statistics of the datasets.

released by Vitale et al. (2012), which contains English news from really simple syndication (RSS) feeds.

**MR**: It is a movie review dataset, in which each review only contains one sentence (Pang and Lee, 2005). Each sentence is annotated with positive or negative for binary sentiment classification.

**Twitter**: This dataset is provided by NLTK[4], a library of Python, which is also a binary sentiment classification dataset.

For each dataset, we randomly select 40 labeled documents per class, half of which for training and the other half for validation. Following Kipf and Welling (2017), all the left documents are for testing, which are also used as unlabeled documents during training.

We preprocess all the datasets as follows. We remove non-English characters, the stop words, and low-frequency words appearing less than 5 times. Table 1 shows the statistics of the datasets, including the number of documents, the number of average tokens and entities, the number of classes, and the proportion of texts containing entities in parentheses. In our datasets, most of the texts (around 80%) contain entities.

#### 4.1.2 Baselines

To comprehensively evaluate our proposed method for semi-supervised short text classification, we compare it with the following nine state-of-the-art methods:

**SVM**: SVM classifiers using TF-IDF features and LDA features (Blei et al., 2003), are denoted as SVM+TFIDF and SVM+LDA, respectively.

**CNN**: CNN (Kim, 2014) with 2 variants: 1) CNN-rand, whose word embeddings are randomly initialized, and 2) CNN-pretrain, whose word embeddings are pre-trained with Wikipedia Corpus.

**LSTM**: LSTM (Liu et al., 2016) with and without pre-trained word embeddings, named LSTM-rand and LSTM-pretrain, respectively.

[4]https://www.nltk.org/

**PTE**: A semi-supervised representation learning method for text data (Tang et al., 2015). It firstly learns word embedding based on the heterogeneous text networks containing three bipartite networks of words, documents and labels, then averages word embeddings as document embeddings for text classification.

**TextGCN**: Text GCN (Yao et al., 2019) models the text corpus as a graph containing documents and words as nodes, and applies GCN for text classification.

**HAN**: HAN (Wang et al., 2019) embeds HINs by first converting an HIN to several homogeneous sub-networks through pre-defined meta-paths and then applying graph attention networks.

For fair comparison, all of the above baselines, such as SVMs, CNN and LSTM, have used entity information.

#### 4.1.3 Parameter Settings

We choose the parameter values of $K$, $T$ and $\delta$ that achieve the best results on the validation set. To construct HIN for short texts, we set the number of topics $K = 15$ in LDA for the datasets AGNews, TagMyNews, MR and Twitter. We set $K = 20$ for Snippets and $K = 40$ for Ohsumed. For all the datasets, each document is assigned to top $P = 2$ topics with the largest probabilities. The similarity threshold $\delta$ between entities is set $\delta = 0.5$.

Following previous studies (Vaswani et al., 2017), we set the hidden dimension of our model HGAT and other neural models to $d = 512$ and the dimension of pre-trained word embeddings to 100. We set the layer number $L$ of HGAT, GCN-HIN and TextGCN as 2. For model training, we set the learning rate as 0.005, dropout rate as 0.8 and the regularization factor $\eta = 5e$-6. Early stopping is applied to avoid overfitting.

### 4.2 Experimental Results

Table 2 shows the classification accuracy of different methods on 6 benchmark datasets. We can see that our methods significantly outperform all the baselines by a large margin, which shows the effectiveness of our proposed method on semi-supervised short text classification.

The traditional method SVMs based on the human-designed features, achieve better performance than the deep models with random initialization, i.e., CNN-rand and LSTM-rand in most cases. While CNN-pretrain and LSTM-pretrain using the pre-trained vectors achieve significant

| Dataset | SVM +TFIDF | SVM +LDA | CNN -rand | CNN -pretrain | LSTM -rand | LSTM -pretrain | PTE | TextGCN | HAN | HGAT |
|---|---|---|---|---|---|---|---|---|---|---|
| AGNews | 57.73 | 65.16 | 32.65 | 67.24 | 31.24 | 66.28 | 36.00 | 67.61 | 62.64 | **72.10*** |
| Snippets | 63.85 | 63.91 | 48.34 | 77.09 | 26.38 | 75.89 | 63.10 | 77.82 | 58.38 | **82.36*** |
| Ohsumed | 41.47 | 31.26 | 35.25 | 32.92 | 19.87 | 28.70 | 36.63 | 41.56 | 36.97 | **42.68*** |
| TagMyNews | 42.90 | 21.88 | 28.76 | 57.12 | 25.52 | 57.32 | 40.32 | 54.28 | 42.18 | **61.72*** |
| MR | 56.67 | 54.69 | 54.85 | 58.32 | 52.62 | 60.89 | 54.74 | 59.12 | 57.11 | **62.75*** |
| Twitter | 54.39 | 50.42 | 52.58 | 56.34 | 54.80 | 60.28 | 54.24 | 60.15 | 53.75 | **63.21*** |

Table 2: Test accuracy (%) of different models on six standard datasets. The second best results are underlined. The note * means our model significantly outperforms the baselines based on $t$-test ($p < 0.01$).

| Dataset | GCN -HIN | HGAT w/o ATT | HGAT -Type | HGAT -Node | HGAT |
|---|---|---|---|---|---|
| AGNews | 70.87 | 70.97 | 71.54 | 71.76 | **72.10*** |
| Snippets | 76.69 | 80.42 | 81.68 | 81.93 | **82.36*** |
| Ohsumed | 40.25 | 41.31 | 41.95 | 42.17 | **42.68*** |
| TagMyNews | 56.33 | 59.41 | 60.78 | 61.29 | **61.72*** |
| MR | 60.81 | 62.13 | 62.27 | 62.31 | **62.75*** |
| Twitter | 61.59 | 62.35 | 62.95 | 62.45 | **63.21*** |

Table 3: Test accuracy (%) of our variants.

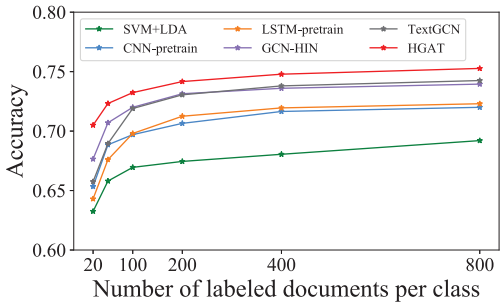


Figure 3: The test accuracy with different number of labeled documents.

improvements and outperform SVMs. The graph based model PTE achieves inferior performance compared to CNN-pretrain and LSTM-pretrain. The reason may be that PTE learns text embeddings based on word co-occurrences, which, however, are sparse in short text classification. Graph neural network based models TextGCN and HAN achieve comparable results with the deep models CNN-pretrain and LSTM-pretrain. Our model HGAT consistently outperforms all the state-of-the-art models by a large margin, which shows the effectiveness of our proposed method. The reasons include that 1) we construct a flexible HIN framework for modeling the short texts, enabling integration of additional information to enrich the semantics and 2) we propose a novel model HGAT to embed the HIN for short text classification based on a new dual-level attention mechanism. The attention mechanism not only captures the importance of different neighboring nodes (reducing the weights of noisy information) but also the importance of different types of nodes.

#### 4.2.1 Comparison of Variants of HGAT

We also compare our model HGAT with some variants to validate the effectiveness of our model. As shown in Table 3, we compare our HGAT with four variant models. The basic model GCN-HIN directly applies GCN on our constructed HIN for short texts by concatenating the feature spaces of different types of information. It does not consider the heterogeneity of various information types. HGAT w/o ATT considers the heterogeneity through our proposed heterogeneous graph convolution, which projects different types of information to an implicit common space with respective transformation matrices. HGAT-Type and HGAT-Node respectively consider only type-level attention and node-level attention.

We can see from Table 2, HGAT w/o ATT consistently outperforms GCN-HIN on all datasets, demonstrating the effectiveness of our proposed heterogeneous graph convolution which considers the heterogeneity of various information types. HGAT-Type and HGAT-Node further improve HGAT w/o ATT by capturing the importance of different information (reducing the weights of noisy information). HGAT-Node achieves better performance than HGAT-Type, indicating that node-level attention is more important. Finally, HGAT significantly outperforms all the variants by considering the heterogeneity and applying dual-level attention mechanism including node-level and type-level attentions.

#### 4.2.2 Impact of Number of Labeled Docs

We choose 6 representative methods with the best performance: SVM+LDA, CNN-pretrain, LSM-

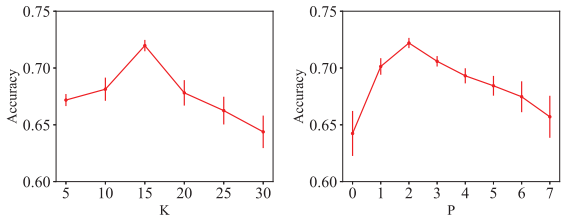


(a) $K$: Number of topics. (b) $P$: Top $P$ relevant topics.

Figure 4: The average accuracy with different number of topics and top relevant topics on AGNews.

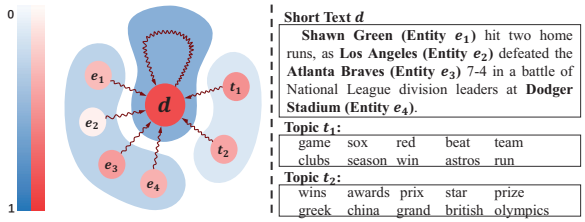


Figure 5: Visualization of the dual-level attention including node-level attention (shown in red) and type-level attention (shown in blue). Each topic $t$ is represented by top 10 words with highest probabilities.

pretrain, GCN-HIN, TextGCN and HGAT, to study the impact of the number of labeled documents. Particularly, we vary the number of labeled documents per class and compare their performance on the AGNews dataset. We run each method 10 times and report the average performance. As shown in Figure 3, with the increase of labeled documents, all the methods achieve better results in terms of accuracy. Generally, the graph based methods GCN-HIN, TextGCN and HGAT achieve better performance, indicating that graph-based methods can make better use of limited labeled data through information propagation. Our method outperforms all the other methods consistently. When fewer labeled documents are provided, the baselines exhibit obvious performance drop, while our model still achieves relatively high performance. It demonstrates that our method can more effectively take advantage of the limited labeled data for short text classification. We believe our method benefits from the flexible HIN and the proposed heterogeneous graph attention networks with dual-level attention.

#### 4.2.3 Parameter Analysis

Figure 4 (a) and (b) show the test accuracy of our HGAT model on the AGNews dataset with different number of topics $K$ and Top $P$ relevant topics assigned to a document. As we can see clearly, for the number of topics, the test accuracy first increases with the increase of the number of topics, reaching the highest value at 15; it falls when its number is larger than 15. We also tried the different numbers of topics for baselines, and have observed that the best K is the same as in our model. This is consistent with the intuition that the number of topics should fit the dataset, i.e., it should be model free. For the number of top relevant topics $P$ assigned to the documents, the test accuracy first increases with the increase of $P$ and then decreases when $P$ is larger than 2. In our experiments, the two parameters are set based on the validation set of each dataset.

#### 4.2.4 Case Study

As Figure 5 shows, we take a short text from AGNews as an example (which is classified to the class of sports correctly) to illustrate the dual-level attention of HGAT. The type-level attention assigns high weight (0.7) to the short text itself, while lower weights (0.2 and 0.1) to entities and topics. It means that the text itself contributes more for classification, than the entities and topics. The node-level attention assigns different weights to neighboring nodes. The node-level weights of nodes belonging to a same type sum to 1. As we see, the entities $e_3$ (Atlanta Braves, a baseball team), $e_4$ (Dodger Stadium, a baseball gym), $e_1$ (Shawn Green, a baseball player) have higher weights than $e_2$ (Los Angeles, referring to a city at most time). The topics $t_1$ (game) and $t_2$ (win) have almost the same importance for classifying the text to the class of sports. The case study shows that our proposed dual-level attention can capture key information at multiple granularities for classification and reduce the weights of noisy information.

## 5 Conclusion

In this paper, we propose a novel heterogeneous graph neural network based method for semi-supervised short text classification, which takes full advantage of both limited labeled and large unlabeled data by information propagation. Particularly, we first present a flexible HIN framework for modeling the short texts, which can integrate any additional information and capture their rich relations to address the semantic sparsity of short texts. Then, we propose a novel model HGAT to embed the HIN based on a dual-level attention mechanism including node-level and type-level at-

tentions. HGAT considers the heterogeneity of various information types by projecting them into an implicit common space. Additionally, the dual-level attention captures the key information at multiple granularity levels and reduces the weights of noisy information. Extensive experimental results demonstrated that our proposed model significantly outperforms the state-of-the-art methods across six benchmark datasets consistently.

As our model HGAT is a general HIN embedding approach, it would be interesting to apply it to other tasks, e.g., HIN based recommendation.

## Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 61806020, 61772082, 61702296), the National Key Research and Development Program of China (2017YFB0803304), the Beijing Municipal Natural Science Foundation (4182043), the CCF-Tencent Open Fund, and the Fundamental Research Funds for the Central Universities.

## References

Charu C Aggarwal and ChengXiang Zhai. 2012. A survey of text classification algorithms. In *Mining text data*, pages 163–222. Springer.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Xingyuan Chen, Yunqing Xia, Peng Jin, and John Carroll. 2015. Dataless text classification with descriptive lda. In *AAAI*.

Harris Drucker, Donghui Wu, and Vladimir N Vapnik. 1999. Support vector machines for spam categorization. *IEEE Transactions on Neural Networks*, 10(5):1048–1054.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751.

Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. In *IJCAI*, pages 2873–2879. AAAI Press.

Yue Lu and Chengxiang Zhai. 2008. Opinion integration through semi-supervised topic modeling. In *WWW*, pages 121–130. ACM.

Yu Meng, Jiaming Shen, Chao Zhang, and Jiawei Han. 2018. Weakly-supervised neural text classification. In *CIKM*, pages 983–992. ACM.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL*, pages 115–124. Association for Computational Linguistics.

Xuan-Hieu Phan, Le-Minh Nguyen, and Susumu Horiguchi. 2008. Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *WWW*, pages 91–100. ACM.

François Rousseau, Emmanouil Kiagias, and Michalis Vazirgiannis. 2015. Text categorization as a graph classification problem. In *ACL*, volume 1, pages 1702–1712.

Kazuya Shimura, Jiyi Li, and Fumiyo Fukumoto. 2018. HFT-CNN: Learning hierarchical category structure for multi-label short text categorization. In *EMNLP*, pages 811–816, Brussels, Belgium. Association for Computational Linguistics.

Koustuv Sinha, Yue Dong, Jackie Chi Kit Cheung, and Derek Ruths. 2018. A hierarchical neural attention-based text classifier. In *EMNLP*, pages 817–823, Brussels, Belgium. Association for Computational Linguistics.

Ge Song, Yunming Ye, Xiaolin Du, Xiaohui Huang, and Shifu Bie. 2014. Short text classification: A survey. *Journal of Multimedia*, 9(5):635.

Jian Tang, Meng Qu, and Qiaozhu Mei. 2015. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *SIGKDD*, pages 1165–1174. ACM.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*, pages 5998–6008. Curran Associates, Inc.

Daniele Vitale, Paolo Ferragina, and Ugo Scaiella. 2012. Classification of short texts by deploying topical annotations. In *ECIR*, pages 376–387. Springer.

Chenguang Wang, Yangqiu Song, Haoran Li, Ming Zhang, and Jiawei Han. 2016. Text classification with heterogeneous information network kernels. In *AAAI*, pages 2130–2136.

Jin Wang, Zhongyuan Wang, Dawei Zhang, and Jun Yan. 2017. Combining knowledge with deep convolutional neural networks for short text classification. In *IJCAI*, volume 350.

Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *ACL*, pages 90–94. Association for Computational Linguistics.

Xiang Wang, Ruhua Chen, Yan Jia, and Bin Zhou. 2013. Short text classification using wikipedia concept based document representation. In *ICITA*, pages 471–474. IEEE.

Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Peng Cui, P. Yu, and Yanfang Ye. 2019. Heterogeneous graph attention network. In *WWW*.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. *AAAI*.

Chunyong Yin, Jun Xiang, Hui Zhang, Jin Wang, Zhichao Yin, and Jeong-Uk Kim. 2015. A new svm method for short text classification based on semi-supervised learning. In *AITS*, pages 100–103. IEEE.

Jichuan Zeng, Jing Li, Yan Song, Cuiyun Gao, Michael R Lyu, and Irwin King. 2018. Topic memory networks for short text classification. In *EMNLP*, pages 3120–3131.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NIPS*, pages 649–657.

# Comparing and Developing Tools to Measure the Readability of Domain-Specific Texts

**Elissa M. Redmiles[1], Lisa Maszkiewicz[1], Emily Hwang[1], Dhruv Kuchhal[2], Everest Liu[1], Miraida Morales[3], Denis Peskov[1], Sudha Rao[1], Rock Stevens[1], Kristina Gligorić[4], Sean Kross[5], Michelle L. Mazurek[1],** and **Hal Daumé III[1,6]**
[1]University of Maryland {eredmiles,mmazurek,hal}@cs.umd.edu
[2]Georgia Tech [3]Rutgers University [4]EPFL [5]UCSD [6]Microsoft Research

## Abstract

The readability of a digital text can influence people's ability to learn new things about a range topics from digital resources (e.g., Wikipedia, WebMD). Readability also impacts search rankings, and is used to evaluate the performance of NLP systems. Despite this, we lack a thorough understanding of how to validly measure readability at scale, especially for domain-specific texts.

In this work, we present a comparison of the validity of well-known readability measures and introduce a novel approach, *Smart Cloze*, which is designed to address shortcomings of existing measures. We compare these approaches across four different corpora: crowdworker-generated stories, Wikipedia articles, security and privacy advice, and health information. On these corpora, we evaluate the convergent and content validity of each measure, and detail tradeoffs in score precision, domain-specificity, and participant burden. These results provide a foundation for more accurate readability measurements and better evaluation of new natural-language-processing systems and tools.

## 1 Introduction

Readability metrics are used in a variety of computational contexts, including to evaluate the quality of novel natural language processing (NLP) systems (Sugawara et al., 2017; Kandula et al., 2010) or machine generated translations (House, 2014), or to rank search results (Slegg, 2018). A variety of readability metrics are available for assessing comprehensibility of texts: human-expert-written comprehension questions, automatically generated readability tests, and computed metrics requiring no human/agent input (Bormuth, 1968; Flesch, 1948; Taylor, 1953; Graesser et al., 2004).

Despite being used frequently in computational contexts, the majority of readability assessments were developed for grade-school texts and validated with grade-school readers. Online texts are generally targeted toward adult readers, leading to differences in text structure (e.g., bullet points), word abstraction (Graesser et al., 2004), and domain-specificity (e.g., medical advice, digital-security advice). Such differences may affect the accuracy of computed metrics and automatically generated readability tests, which are increasingly used to scale readability measurements in the digital world (Friedman and Hoffman-Goetz, 2006; Eysenbach et al., 2002; Bernstam et al., 2005). Despite this use, the validity of readability assessment techniques has rarely been re-evaluated for online contexts.

In this work, we make three contributions: **First**, we evaluate the most commonly used methods for measuring readability in terms of *content validity* (the degree to which different measures relate to theoretically-grounded linguistic components like text cohesion or syntactic complexity), *convergent validity* (the degree to which these measures correspond to each other), *redundancy* (the degree to which one measure is subsumed by another), and *score precision* (the shape of the distribution of score from a given measure, and how well it distinguishes among documents). **Second**, we identify a need for domain-specific automatically generated readability tests. To address this, we develop and evaluate a novel technique for automatically generating readability tests specifically for domain-specific texts: *Smart Cloze*. We find that Smart Cloze offers some benefits for domain-specific applications compared to existing measures. **Third**, we contribute two open-science resources: our open-source Smart Cloze tool, as well as a *Digital Readability* evaluation corpus of 100 documents, including 300 comprehension questions written by human experts, that we use in our evaluation.

## 2 Related Work

Human-written comprehension questions are the gold standard for measuring readability (Duke and Pearson, 2009; Sarroub and Pearson, 1998), but developing such questions is costly and difficult to scale. As such, prior work has explored various automated, scalable approaches to generating comprehension questions. One such approach is automatic reading test generation, typically using the Cloze (Taylor, 1953) procedure, which involves removing every *n*th word in a given document and requiring the reader to "fill-in-the-blank" with the correct word. The Cloze procedure was validated as a scalable method of comprehension assessment through comparison with expert-written comprehension questions for grade-school texts (Bormuth, 1967; Heilman, 2011; Rankin and Culhane, 1969; Oller et al., 1972).

Recently, researchers have explored approaches to adjusting the construction of Cloze tests: selecting particular key sentences or parts of speech to use as blanks, often to assess retention of factual knowledge or awareness of vocabulary (Goto et al., 2010; Chen et al., 2006; Gates, 2011; Lin and Ji, 2010; Lee and Seneff, 2007), and multiple-choice Cloze tests in which test-takers select from a set of distractors rather than filling in an open blank, avoiding potential scoring issues with typos and equally-correct synonyms (Goto et al., 2010; Narendra et al., 2013; Brown et al., 2005; Mostow and Jang, 2012; Gates, 2011; Pino et al., 2008; Hoshino and Nakagawa, 2007). Our Smart Cloze tool builds on this prior work by choosing distractors from a domain-specific rather than a general dictionary, answering the call from Collins-Thompson's 2014 review of readability measures for more domain-specific tool options.

The second scalable alternative consists of readability metrics that take no reader input. The first such metrics were readability formulae, the most popular of which is the Flesch reading ease score (FRES) (Tekfi, 1987; Flesch, 1948, 1943). FRES assumes that longer sentences and words — which often co-occur with complex syntax — indicate greater reading difficulty (Dale and Tyler, 1934; Feng et al., 2009). More recently, linguistic feature-based (McNamara et al., 2014; François and Miltsakaki, 2012; Collins-Thompson and Callan, 2004) and machine learning approaches have also been used to predict the readability of text (Kate et al., 2010; De Clercq and Hoste, 2016; Collins-Thompson, 2014). Finally, there has also been recent work that goes beyond readability to assess the overall *quality* of text, including factors such as topical interest, persuasiveness, or grammatical correctness (Louis and Nenkova, 2013; Pitler et al., 2010; Tan et al., 2016). In this work, we focus strictly on readability and exclude other quality measures from our comparative evaluation.

All but one of the new approaches to measuring readability (modified Cloze and linguistic metric/ML approaches) were evaluated only through comparison with annotator judgements of perceived readability or correlation with FRES-style formulae, rather than the gold standard of human comprehension questions (Benjamin, 2012), the exception being Mostow et al. (2004). Further, these evaluations were conducted strictly with grade-school texts, leaving a significant gap around online, adult texts (Benjamin, 2012). Our work fills this gap in evaluation (Benjamin, 2012; Collins-Thompson, 2014) by systematically evaluating the validity of currently used readability metrics through comparison with each other and with results from expert written comprehension tests on a wide set of both general and domain-specific documents.

## 3 Methods

In our evaluation, we compare readability scores from five sources: human-written comprehension questions; automatically generated readability tests including both traditional Cloze and our Smart Cloze domain-specific variant; annotator perceived ease (Sauro and Dumas, 2009; Rello et al., 2016), which has been used to evaluate readability metrics in the past; and the Flesch Reading Ease Score (FRES) (Flesch, 1948). We compared these metrics across our *Digital Readability* evaluation corpus. Here we describe our corpus, how we generated each of the readability metrics, and how we conducted our validity analysis.

### 3.1 Digital Readability Corpus

We draw our final evaluation corpus from four source corpora, as follows:

**Story corpus.** We drew our crowd-worker-created stories from the MCTest (Richardson et al., 2013) dataset, which consists of 500 simple stories created by Amazon Mechanical Turk crowdworkers and validated manually for quality.

**Wikipedia corpus.** We drew our Wikipedia articles from a corpus of 20,000 Wikipedia articles scraped from Wikipedia and cleaned for quality by Shaoul (2010). We selected Wikipedia articles as a baseline of adult texts against which to compare the domain-specific texts. Wikipedia articles have a mean FRES similar to our domain-specific texts (mean FRES for the wikipedia sample = 47.9; for the health documents = 53.7; and for the security documents = 48.7), suggesting that, at least by one measure, the texts should be similar in readability.

**Health corpus.** We drew health articles from the 500-document Health Text Readability Corpus (Morales and Wacholder, 2018). This corpus includes consumer health-information documents made available for public use by the CDC, NIH, American Heart Association, American Diabetes Association, and the National Library of Medicine's Medline Plus resource. Worksheets, posters, infographics, and websites are not included. More than half (N=293) of the documents were found in "Easy to Read" collections; that is, the document has been designated by its source agency as appropriate for adults who read at or below a 7th-8th grade reading level.

**Security corpus.** We collected security advice documents through two methods: (a) asking MTurk workers to create Google search queries for computer security advice, then scraping the top 20 Google results of each query, using the DiffBot API[1] to parse and sanitize HTML body elements within each identified site, and (b) by asking 10 security experts and librarians to recommend digital security advice sources and scraping those websites. These two approaches, along with a manual cleaning process in which we performed spot checks and also manually reviewed 144 documents identified as outliers by FRES or length, generated 1,878 security advice documents.

The last two corpora – health and security advice – are domain-specific: focused on a singular domain and often containing jargon or topics not typically encountered in daily life.

**Final evaluation corpus.** To ensure comparability of results, we used a standardized subsampling procedure to select 25 documents from each corpus. To ensure that our evaluation captured some variance in documents, we subsampled by length. We first remove the shortest and longest 5% of documents, then we then divide the documents into five bins by length, based on how many standard deviations the length of a given document is from the mean length for that corpus. We manually reviewed all selected documents to ensure that they were on-topic and appropriately clean.[2]

### 3.2 Readability Metrics

We created three **comprehension questions** for each of the documents in our evaluation corpus: one True/False question and two multiple choice questions with four answer options each, per comprehension question best practices (Anderson, 1972; Day and Park, 2005). Domain-specific questions were written by three co-authors who were domain experts in digital security or in health; the general questions were written by two other co-authors. All 300 comprehension questions were reviewed and edited by a paid comprehension question specialist, who had experience writing and evaluating comprehension questions for the SAT, Discovery Science, and similar organizations; the specialist spent more than 10 hours editing and refining the questions.[2]

We selected the FRES as our **computed measure**, as it is the most-used by number of citations, and anecdotally, by wide-spread application. We computed the FRES for each document using the Python `textstat` package [3].

For our **annotator perception of ease** measurement, we use a single-item question "How easy is this document to read?" with 5-point Likert-item response choices ranging from "Very Easy" to "Very Hard."

Finally, for our **automatically generated readability tests** we used both the traditional Cloze Procedure and our Smart Cloze procedure. Prior work suggests that the frequency of blanks does not significantly affect results (Taylor, 1953). We select set $n = 5$, up to a maximum of 35 target words, for both our traditional Cloze implementation and our Smart Cloze tests, as was done in the original Cloze implementation (Taylor, 1956).

**Smart Cloze tool.** Prior work to improve Cloze tests offered a multiple-choice variant of the traditional Cloze procedure in which distractors (incorrect answer choices) are randomly drawn from a general dictionary containing other words with the

[1] https://www.diffbot.com

[2] You can find the documents in our corpus, the 300 comprehension questions, and the code for generating traditional Cloze and Smart Cloze tests at: https://github.com/SP2-MC2/Readability-Resources.

[3] https://pypi.org/project/textstat/

same part of speech. While such multiple-choice variants offer improvements in test-taker time, they are potentially inappropriate for domain-specific applications. For example, replacing the word "encryption" in a cybersecurity text with "dog" creates a very easy test. As such, we implemented a novel approach that we call Smart Cloze: we construct a domain-specific dictionary from the same corpus for which we are generating tests and draw distractors from it. The goal is to offer relevant alternatives such as "antivirus" and "key" as distractors for "encryption."

To construct a Smart Cloze test for some document $d$ selected from a domain-specific corpus $c$, our tool follows the following procedure. First, we bin all of the words in $c$ by part of speech (tagged using Spacy[4]) to create a domain-specific dictionary. We then construct a similar part-of-speech-tagged *document-specific dictionary* using only the words in $d$. Third, we identify *target words* in $d$ to be replaced by multiple-choice questions. Fourth, we generate distractors for each target. We randomly select up to 14 potential distractors with the same part of speech as the target word from each of the domain-specific and document-specific dictionaries. We then process these distractors in random order, optimizing to obtain two from each dictionary, until we have found four satisfactory distractors.

We measure whether a potential distractor is satisfactory by examining how probable it is that the distractor might substitute for the target word within $d$. To do this, we first look up the bigram probabilities of the target word ($w_c$) with its preceding ($w_{c-1}$) and following ($w_{c+1}$) words in Google's n-gram corpus. This gives us a baseline for how probable the correct answer is. We then look up bigram probabilities of the potential distractor (say $w_d$) in combination with the same preceding ($w_{c-1}$) and following ($w_{c+1}$) words. Satisfactory distractors have both preceding-distractor and distractor-following bigram probabilities within two orders of magnitude of those for the correct target word. [5] More precisely, a distractor $w_d$ will be accepted if:

$$[P(w_d|w_{c+1}) \geq P(w_c|w_{c+1})] \wedge [P(w_{c-1}|w_d) >= P(w_{c-1}|w_c)]$$

If we do not find four satisfactory distractors (by this definition) within the candidate 28, we instead select the potential distractors with the highest bigram probabilities until we obtain the desired four distractors. Finally, to avoid very small lists of distractor options for certain part of speech (e.g., `TO` only contains to'), we merge parts of speech with small wordlists with larger, related parts of speech until enough unique distractors can be found.

### 3.3 Validity Evaluation

To evaluate the validity of these readability metrics and compare them, we needed readers to answer the comprehension questions, Cloze tests, and ease question for our documents. We recruited U.S. Amazon Mechanical Turk workers (MTurkers) with a 95% approval rating or above to complete these tasks. Each worker completed one randomly selected readability measure for four documents, including one randomly selected from each of the four corpora. MTurkers were compensated with $1.50 for completing the task. We recruited at least five distinct MTurkers for each type of measure and each document (n=841).

We compare our five readability metrics by examining their construct validity (Cronbach and Meehl, 1955): the degree to which it appears that the measures are accurately measuring readability. To do so, we examine:

- *Content validity:* the degree to which the measures relate to concepts that have been theorized to be relevant to readability; and
- *Convergent validity:* the degree to which related measures (e.g., multiple measures of the same construct) are correlated.

We also explore three factors that are relevant to selecting an appropriate readability measure:

- *Redundancy:* the degree to which any measure is fully, and redundantly, covered by another measure;
- *Score precision:* the precision with which the measure distinguishes between different documents; and
- *Participant burden:* the cost of the measure to the participant (and the researcher) in time to complete.

To assess **content validity**, we examine the degree to which five core linguistic components (narrativity, syntactic simplicity, word concreteness, referential cohesion, and deep cohesion) theorized to be related to readability (Graesser et al., 2004) can *explain the variance* in the measure scores. We

[4] https://spacy.io

[5] We selected two orders of magnitude heuristically to narrow the search space for faster computation while obtaining an appropriate difficulty for the test. Future work could explore alternative heuristics in more detail.

measure these components using the Cohmetrix tool (Graesser et al., 2004). We construct linear regression models, in which the mean measure score for a document is the outcome variable and the input variables are the five linguistic components.

As we wish to understand *which* components are related to which measures, we seek to ensure that we construct a model of best fit. To do so, we perform feature selection via stepwise backward selection, minimizing AIC (Bursac et al., 2008). We further measure applicability to domain-specific texts by including the source corpora of the document as a sixth covariate in the regression model. We set Wikipedia as the baseline for corpora source, as it represents a broad set of non-domain-specific documents with similar FRES to the domain-specific documents.

To assess **convergent validity**, we compute the Pearson correlation between the scores for each readability method in our evaluation dataset. We report the $\rho$ value (strength of the correlation) for correlations significant at $\alpha < 0.05$; Holm-Bonferonni (Abdi, 2010) correction is applied to account for multiple testing.

We also assess **redundancy**, which is not strictly a property of convergent validity, but is relevant when comparing multiple measures that attempt to assess the same construct. Demonstrating that two related measures are correlated establishes convergent validity, but if they are perfectly correlated, then it is unlikely both are needed (Quinn et al., 2010). For this analysis, we construct linear regression models in which the mean score from a given measure for a given document is the outcome variable and the input variables are the three other types of measures (note that we do not include both Cloze measures in any model, but instead construct separate, three-variable models, each with FRES, comprehension questions, ease, and one of the Cloze measures). We consider the degree of redundancy to be the proportion of variance in measure scores explained by the other measures (that is, the $R^2$ value of this regression model).

To assess **score precision**, we examine the shape of the distribution of scores for a given measure. Per best practice for observing distributions, we do so both through visual inspection and by measuring kurtosis (a statistical measure of the 'tailness' of a distribution) (DeCarlo, 1997).

Finally, we assess **participant burden** in terms of time to complete the task (which also proxies for researcher cost). We compare time by bootstrapping confidence intervals for the mean time for completion of a readability assessment for a given document. Non-overlapping confidence intervals indicate a significant difference in completion time.

### 3.4 Limitations

Our work is subject to four primary limitations. First, automatic selection of distractors means that there may be differences in the difficulty of different distractors (or variances in difficulty of tests generated by the method when used repeatedly). Based on a manual review of the Cloze tests we conducted before deployment, we did not find trivial distractors to be highly prevalent, given the breadth of words available in each dictionary. However, future work may wish to explore methods for measuring and ensuring consistency in distractor difficulty. Second, MTurk respondents are known to be more educated than the general population, and thus the results of our work may not generalize to low-literacy populations, second-language learners, and others (Redmiles et al., 2019; Ipeirotis, 2010). Third, while we attempted to cover a relatively broad space of online documents, other types of documents (e.g., news articles, Facebook posts) may perform differently. Finally, it is possible that MTurkers were inattentive to our tasks, limiting the validity of our data. We mitigate this possibility by restricting our sample to workers with 95% approval rates on past tasks, as shown in prior work to ensure participant attention to surveys as well as gold-standard 'test' questions (Peer et al., 2014).

## 4 Results

In this section we summarize our results for content validity (including domain sensitivity of measurements), convergent validity, redundancy, score precision, and participant burden (Table 1).

### 4.1 Content Validity

We find that comprehension question scores are significantly related to the narrativity ($p = 0.003$) and syntactic complexity ($p = 0.035$) of the document, while performance on comprehension questions is not significantly related to the other three linguistic factors we examined (word concreteness, referential cohesion, deep cohesion) or to

| | Linguistic Components (Content Validity) | | | | | Additional Considerations | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Narrativity | Syntactic Simplicity | Word Concreteness | Referential Cohesion | Deep Cohesion | Burden (Mean Time) | Mean Score | Score Precision (Distribution Trend) | Domain Sensitivity |
| Comprehension | ✓ | ✓ | | | | 2.86 min | 75.7% | exponential | |
| Traditional Cloze | ✓ | | | ✓ | | 5.05 min | 34.1% | normal | |
| Smart Cloze | ✓ | ✓ | | ✓ | | 4.55 min | 52.4% | normal | ✓ |
| Ease | | | ✓ | | | 1.67 min | 67.1% | uniform | ✓ |
| FRES | ✓ | ✓ | ✓ | | | — | 61.0% | uniform | ✓ |

Table 1: Summary of our results on content validity (significant relationships between readability measure and linguistic components theorized to explain comprehension) and other considerations for selecting a readability measure (time for participants to complete a test for a given measure on an average document, average score achieved across documents, trend in the shape of the distribution of scores achieved with a measure, and whether the measure exhibits variation by document domain.

type of document (source corpus).

Traditional and Smart Cloze scores are significantly related to the narrativity (Traditional: $p < 0.001$; Smart: $p = 0.040$) and referential cohesion (Traditional: $p = 0.035$; Smart: $p = 0.008$) of the document. Smart Cloze scores were significantly related to the syntactic complexity ($p = 0.005$) of the document; traditional Cloze scores were not significantly related to syntactic complexity. Finally, neither type of Cloze score was significantly related to deep cohesion or to word concreteness. Smart Cloze scores vary significantly by document domain, while traditional Cloze scores do not. Specifically, Smart Cloze scores are significantly higher for domain-specific documents: those from the health ($p < 0.001$) and security (0.031) source corpora, than for Wikipedia documents. We hypothesize that this is the case because the topics of domain-specific documents are narrower — there are fewer reasonable options for any given blank space — than in the Wikipedia documents, resulting in easier multiple-choice questions. (Anecdotal observation of the generated questions seems to align with this theory.)

Ease perceptions are significantly related only to word concreteness ($p = 0.015$) and document domain: stories ($p = 0.027$) and security ($p = 0.015$) documents are perceived as significantly easier to read than Wikipedia articles. The relationship between ease perceptions and concreteness (and lack of relationship with the other linguistic features we examined) is worth remark. Concreteness of words appears to be easy for readers to assess with a quick glance at an article. This assessment, and their overall perception of ease, may in turn determine whether readers are willing to further read a document they encounter "in the wild," at which point other readability factors may become more relevant. We therefore hypothesize that ease and other measures may complement

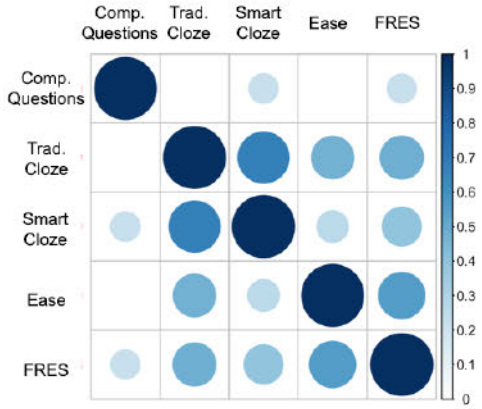


Figure 1: Correlation matrix showing the convergent validity of the measures. That is, the correlation between readability measurement methods. Non-significant correlations ($p > 0.05$) are not shown.

each other. Finally, FRES scores are significantly related to narrativity ($p < 0.001$), word concreteness ($p < 0.001$), and syntactic complexity ($p < 0.001$); but not to either referential or deep cohesion. Perhaps unsurprisingly, FRES scores were significantly higher for stories than for Wikipedia ($p < 0.001$). FRES scores were also higher for security than for Wikipedia ($p = 0.015$), but the health and Wikipedia documents in our sample did not differ in FRES.

While the regression models we constructed explained a significant portion of the variance in scores for ease [6] ($R^2 = 0.504$), FRES ($R^2 = 0.758$), Smart Cloze ($R^2 = 0.389$) and traditional Cloze ($R^2 = 0.334$), these factors explained much less of the variance for comprehension question scores ($R^2 = 0.132$).

[6] This result closely parallels prior work, which predicted perceived ease of Wall Street Journal articles using discourse, vocabulary and length, resulting in an $R^2$ of 0.503 (Pitler and Nenkova, 2008).

### 4.2 Convergent Validity

To examine **convergent validity**, we examine the correlation between scores from different measures (Figure 1). Comprehension question scores have the least correlation with scores from the other methods: no correlation with traditional Cloze or ease ratings, and small correlation with FRES ($\rho = 0.22$) and Smart Cloze ($\rho = 0.23$).

This low correlation between comprehension questions and the other methods of measuring readability, together with the low explanation of variance noted above, suggest that comprehension questions assess a combination of the readability of the text and the reader's cognitive abilities, different from the other metrics, which may be more specific to just the text itself (Sarroub and Pearson, 1998). Traditional Cloze, on the other hand, correlates relatively well with all other methods. Perhaps unsurprisingly, there is high correlation ($\rho = 0.71$) between traditional and Smart Cloze scores. Traditional Cloze also correlates well with ease ($\rho = 0.47$) and FRES ($\rho = 0.48$). Smart Cloze correlates less with ease than does traditional Cloze (ease: $\rho = 0.264$, FRES: $\rho = 0.44$). Finally, ease and FRES correlate relatively strongly with each other ($\rho = 0.56$).

### 4.3 Redundancy

By constructing regression models with the mean score from a given measure on a given document as the outcome variable, and the other measures as the input variables, we find that 4.02% of the variance in the comprehension question scores can be explained by ease perception, FRES, and traditional Cloze (7.92% with Smart Cloze). 20.1% of the variance in traditional Cloze is explained by the other measures, while 22.1% of the variance in Smart Cloze is explained by these measures. 36.0% of the variance in ease perception is explained by mean comprehension question scores, FRES, and traditional Cloze (31.8% with Smart Cloze), while 35.8% of the variance in FRES measurements is explained by scores on comprehension questions, ease perception, and traditional Cloze (37.8% Smart Cloze). Thus, none of the measures are redundant, as the variance in no measure is fully (or even more than 50%) explained by the others.

### 4.4 Score Precision

Researchers selecting a readability measurement method may also wish to consider the **score precision**: that is, are you trying to find a few bad outliers in a corpus of highly readable documents, or are you expecting a relatively normal distribution of document quality? Figure 2 shows the score distributions by method across all documents and for each document type.

Across domains, the Cloze tests provide the most normal distributions of scores (average traditional Cloze kurtosis = 2.34, average Smart Cloze kurtosis = 3.08)[7]. Cloze scores are thus useful in cases where the relative readability of documents is of interest and where you hypothesize that a normal distribution of readability may be appropriate. The distribution of traditional Cloze scores is transposed left, with a mean of 0.341 (95% confidence interval: [0.329, 0.353]), while the Smart Cloze distribution is centered, with a mean of 0.524 (95% confidence interval: [0.510, 0.537]). Traditional Cloze scores may thus need to be scaled (considered relative to each other rather than as absolute values) to account for this observed ceiling effect.

Ease ratings and FRES, on the other hand, have a more platykurtic distribution (ease: average kurtosis 1.91; FRES: average kurtosis 1.94; fully uniform or platykurtic distribution is 1). A platykurtic distribution has fewer outliers than a normal distribution. Thus, these methods may be more useful in corpora where you expect few readability outliers. Further, ease ratings and FRES both have means higher than 0.5: ease has a mean across domains of 0.671 (95% CI: [0.657, 0.685]) and FRES has a mean of 0.610 (95% CI: [0.594, 0.625]). Given these relatively high means, these methods may also need to be scaled, or may be most useful in cases where you anticipate that an average document in your corpus will be fairly readable. Comprehension questions provide a similarly platykurtic distribution (average kurtosis: 2.06), but with a very high mean (0.757, 95% CI:[0.739, 0.778]).

### 4.5 Participant Burden

Finally, research is often constrained by resources, including time and budget, and ethically we must be mindful of the burden we impose on our partici-

[7]The kurtosis of a normal distribution is 3; the kurtosis of a unifom distribution is 1.

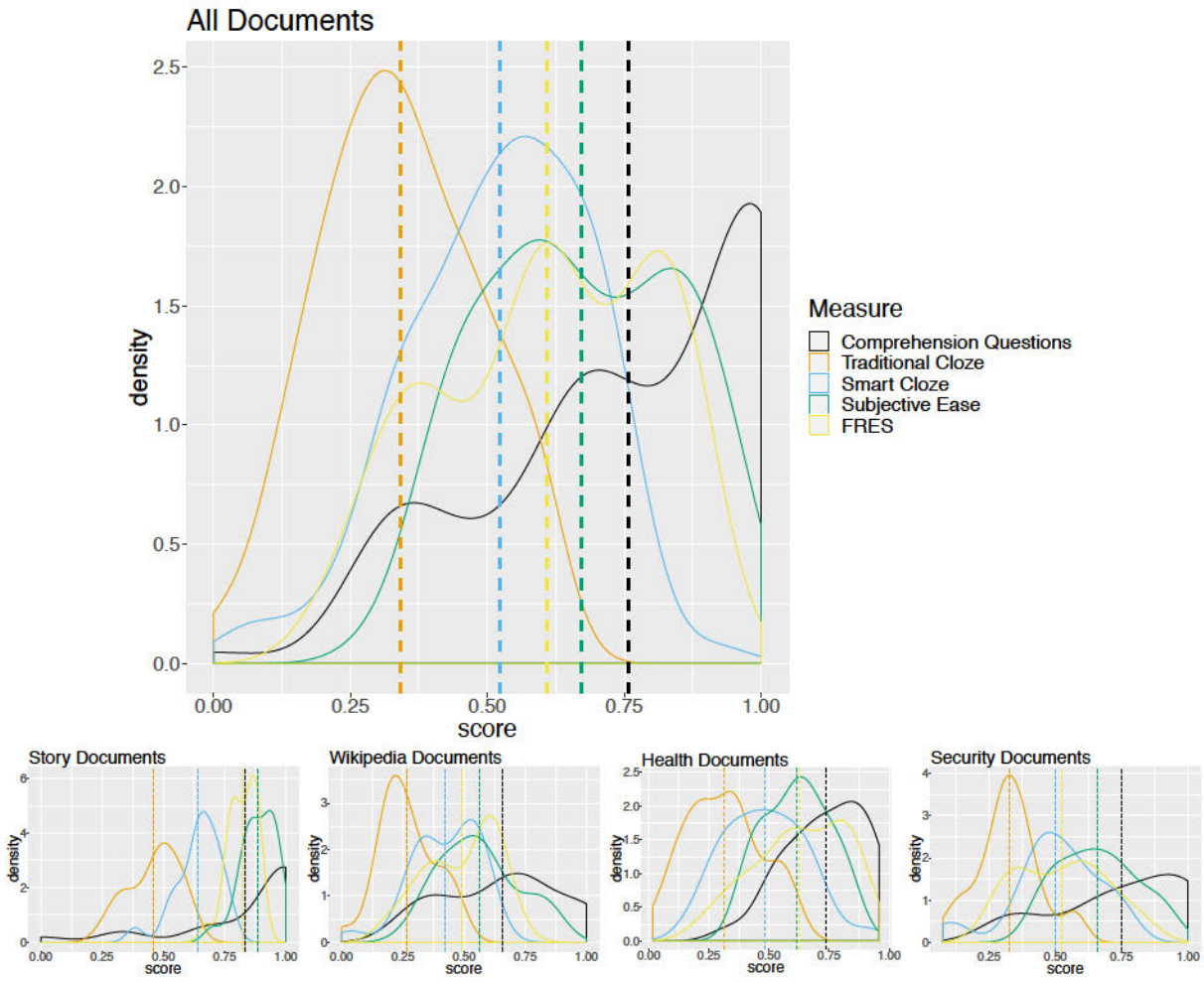


Figure 2: Score distributions by method, across all corpora (top) and by corpus (bottom).

pants. Ease perception (one question) is the fastest test for a worker to complete, with participants spending an average of 1.67 minutes (95% CI: [1.56, 1.78]) per document. Comprehension questions (three questions) took a significantly longer period of time, averaging 2.86 minutes ([2.64, 3.12]) per document, followed by Smart Cloze with an average of 4.55 minutes ([4.08, 4.60]) per document. Finally, traditional Cloze took significantly longer than Smart Cloze, averaging 5.05 minutes per document ([4.72, 5.42]).

## 5 Moving Forward

In sum, no single readability metric outperformed all the others. Each metric offers different benefits and tradeoffs, and human-written comprehension questions differ the most from the other metrics. We summarize the relevant considerations for selecting a readability metric in Figure 3 and encourage the use of multiple metrics in cases where creating comprehension questions is not scalable.

We find that comprehension questions and Smart Cloze both relate significantly to syntactic complexity, perhaps because they require selection among different possible answer choices. Traditional and Smart Cloze relate to referential cohesion, which makes logical sense, as filling-in-the-blank questions require context from prior sentences. Finally, ease and FRES relate to word concreteness, potentially providing relevant assessments of "first glance" readability reactions. The readability metrics examined also exhibit convergent validity, with the three traditional methods (traditional Cloze, subjective ease, and FRES exhibiting the strongest correlation in scores. Finally, the measures are not redundant: a significant portion of the variance in each remains unexplained by the others.

These different methods offer different levels of precision: the Cloze methods trend toward normal distributions with low (traditional) and centered (Smart) means. On the other hand, ease

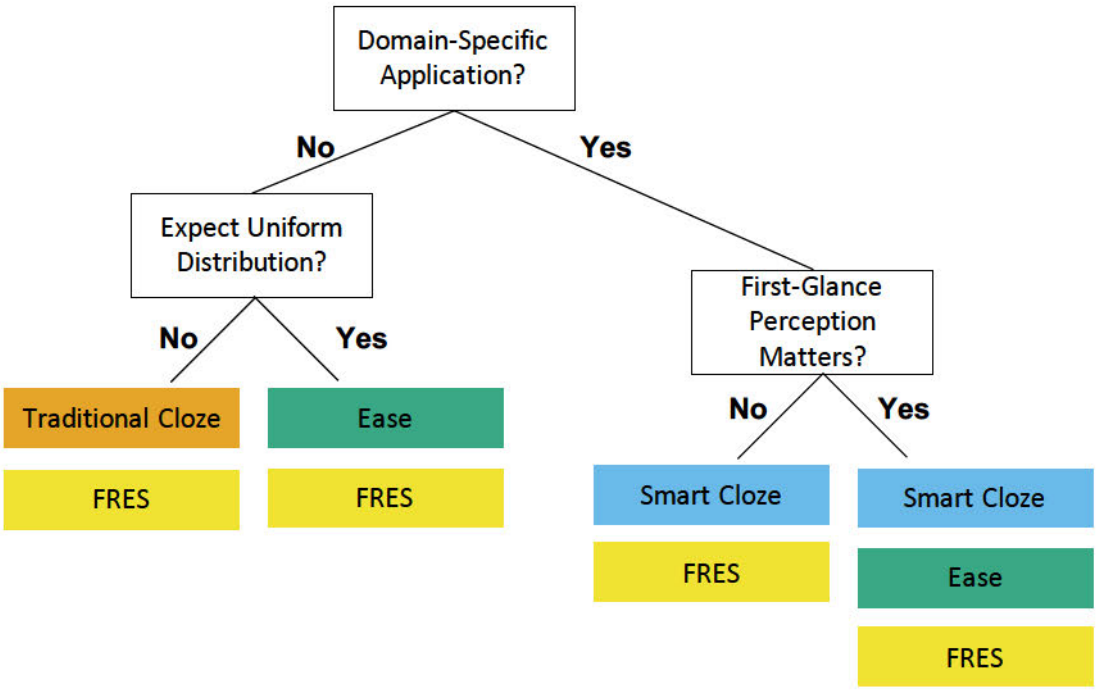


Figure 3: Flow chart for selecting readability measures.

and FRES assessments are more uniformly distributed, with higher means (near 60 and 70%, respectively). Further, Smart Cloze, FRES, and ease measurements all significantly co-varied with document type: Smart Cloze scores were significantly higher for the domain-specific documents (health, security) than for Wikipedia articles, while FRES and ease scores were significantly higher for the story and security documents than for Wikipedia.

While it may be tempting to exclusively use linguistic features because they are cheap and easy to obtain, we find that for the five linguistic factors we explored in this work, these factors explain only 30-50% of the variance in the reader-input readability metrics. Future work may wish to explore additional linguistic factors (Tham, 1987; Hancke et al., 2012; Vajjala and Meurers, 2013; Kate et al., 2010), beyond those covered in this work. In the mean time, our results suggest that when possible, researchers should still consider augmenting these factors with a human-input method. The Smart Cloze tool we propose offers improvements in participant burden, especially for domain-specific documents: scores are higher on average than for traditional Cloze, and tests are 30 seconds faster on average (54 seconds faster for domain-specific documents). However, Smart Cloze is less correlated with perceived ease than traditional Cloze, possibly because the multiple choice option makes the test easier to complete, lessening the chance that participants will "give up." Thus, Smart Cloze is best used in cases where cursory or first glance assessment of readability is less relevant, or in combination with an ease assessment.

## Acknowledgements

We are grateful to all reviewers for their thoughtful feedback. We also thank Hanna Wallach introducing us to the ideas of measurement modeling, as well as in depth discussions on the topic, as well as sharing her work with Abigail Jacobs on this topic in their in-preparation paper (Jacobs and Wallach).

This material is based upon work supported by a UMIACS contract under the partnership between the University of Maryland and DoD. Elissa M. Redmiles additionally wishes to acknowledge support from the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE 1322106 and a Facebook Fellowship.

## References

Hervé Abdi. 2010. Holm's sequential bonferroni procedure. *Encyclopedia of research design*, 1(8):1–8.

Richard C Anderson. 1972. How to construct achievement tests to assess comprehension. *Review of educational research*, 42(2):145–170.

Rebekah George Benjamin. 2012. Reconstructing readability: Recent developments and recommendations in the analysis of text difficulty. *Educational Psychology Review*, 24(1):63–88.

Elmer V Bernstam, Dawn M Shelton, Muhammad Walji, and Funda Meric-Bernstam. 2005. Instruments to assess the quality of health information on the world wide web: what can our patients actually use? *International journal of medical informatics*, 74(1):13–19.

John R Bormuth. 1967. Comparable cloze and multiple-choice comprehension test scores. *Journal of Reading*, 10(5):291–299.

John R Bormuth. 1968. Cloze test readability: Criterion reference scores. *Journal of educational measurement*, 5(3):189–196.

Jonathan C Brown, Gwen A Frishkoff, and Maxine Eskenazi. 2005. Automatic question generation for vocabulary assessment. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*. ACL.

Zoran Bursac, C Heath Gauss, David Keith Williams, and David W Hosmer. 2008. Purposeful selection of variables in logistic regression. *Source code for biology and medicine*, 3(1):17.

Chia-Yin Chen, Hsien-Chin Liou, and Jason S Chang. 2006. Fast: an automatic generation system for grammar tests. In *Proceedings of the COLING/ACL on Interactive presentation sessions*. ACL.

Kevyn Collins-Thompson. 2014. Computational assessment of text readability: A survey of current and future research. *ITL-International Journal of Applied Linguistics*, 165(2):97–135.

Kevyn Collins-Thompson and James P Callan. 2004. A language modeling approach to predicting reading difficulty. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*.

Lee J Cronbach and Paul E Meehl. 1955. Construct validity in psychological tests. *Psychological bulletin*, 52(4):281.

Edgar Dale and Ralph W Tyler. 1934. A study of the factors influencing the difficulty of reading materials for adults of limited reading ability. *The Library Quarterly*, 4(3):384–412.

Richard R Day and Jeong-suk Park. 2005. Developing reading comprehension questions. *Reading in a foreign language*, 17(1):60–73.

Orphée De Clercq and Véronique Hoste. 2016. All mixed up? finding the optimal feature set for general readability prediction and its application to english and dutch. *Computational Linguistics*, 42(3):457–490.

Lawrence T DeCarlo. 1997. On the meaning and use of kurtosis. *Psychological methods*, 2(3):292.

Nell K Duke and P David Pearson. 2009. Effective practices for developing reading comprehension. *Journal of education*, 189(1-2):107–122.

Gunther Eysenbach, John Powell, Oliver Kuss, and Eun-Ryoung Sa. 2002. Empirical studies assessing the quality of health information for consumers on the world wide web: a systematic review. *Jama*, 287(20):2691–2700.

Lijun Feng, Noémie Elhadad, and Matt Huenerfauth. 2009. Cognitively motivated features for readability assessment. In *Proceedings of the 12th Conference of the European Chapter of the ACL*, pages 229–237. ACL.

Rudolf Flesch. 1943. Marks of readable style; a study in adult education. *Teachers College Contributions to Education*.

Rudolph Flesch. 1948. A new readability yardstick. *Journal of applied psychology*, 32(3):221.

Thomas François and Eleni Miltsakaki. 2012. Do nlp and machine learning improve traditional readability formulas? In *Proceedings of the First Workshop on Predicting and Improving Text Readability for target reader populations*, pages 49–57. Association for Computational Linguistics.

Daniela B Friedman and Laurie Hoffman-Goetz. 2006. A systematic review of readability and comprehension instruments used for print and web-based cancer information. *Health Education & Behavior*, 33(3):352–373.

Donna Marie Gates. 2011. How to generate cloze questions from definitions: A syntactic approach. In *AAAI Fall Symposium Series*.

Takuya Goto, Tomoko Kojiri, Toyohide Watanabe, Tomoharu Iwata, and Takeshi Yamada. 2010. Automatic generation system of multiple-choice cloze questions and its evaluation. *Knowledge Management & E-Learning*, 2(3):210.

Arthur C Graesser, Danielle S McNamara, Max M Louwerse, and Zhiqiang Cai. 2004. Coh-metrix: Analysis of text on cohesion and language. *Behavior research methods, instruments, & computers*, 36(2):193–202.

Julia Hancke, Sowmya Vajjala, and Detmar Meurers. 2012. Readability classification for german using lexical, syntactic, and morphological features. In *Proceedings of COLING 2012*, pages 1063–1080.

Michael Heilman. 2011. *Automatic factual question generation from text*. Ph.D. thesis, Carnegie Mellon University.

Ayako Hoshino and Hiroshi Nakagawa. 2007. Assisting cloze test making with a web application. In *Society for Information Technology & Teacher Education International Conference*. AACE.

Juliane House. 2014. Translation quality assessment: Past and present. In *Translation: A multidisciplinary approach*, pages 241–264. Springer.

Panagiotis G Ipeirotis. 2010. Demographics of mechanical turk.

Abigail Z. Jacobs and Hanna Wallach. Measurement and fairness. In preparation.

Sasikiran Kandula, Dorothy Curtis, and Qing Zeng-Treitler. 2010. A semantic and syntactic text simplification tool for health content. In *Annual symposium proceedings*. AMIA.

Rohit J Kate, Xiaoqiang Luo, Siddharth Patwardhan, Martin Franz, Radu Florian, Raymond J Mooney, Salim Roukos, and Chris Welty. 2010. Learning to predict readability using diverse linguistic features. In *Proceedings of the 23rd international conference on computational linguistics*, pages 546–554. Association for Computational Linguistics.

John Lee and Stephanie Seneff. 2007. Automatic generation of cloze items for prepositions. In *Eighth Annual Conference of the International Speech Communication Association*.

Wen-Pin Lin and Heng Ji. 2010. Automatic cloze generation based on cross-document information extraction. In *Asian Conference on Education*.

Annie Louis and Ani Nenkova. 2013. What makes writing great? first experiments on article quality prediction in the science journalism domain. *Transactions of the ACL*, 1:341–352.

Danielle S McNamara, Arthur C Graesser, Philip M McCarthy, and Zhiqiang Cai. 2014. *Automated evaluation of text and discourse with Coh-Metrix*. Cambridge University Press.

Miraida Morales and Nina Wacholder. 2018. Conceptualizing the role of reading and literacy in health information practices. In *International Conference on Information*. Springer.

Jack Mostow, Joseph Beck, Juliet Bey, Andrew Cuneo, June Sison, Brian Tobin, Joseph Valeri, et al. 2004. Using automated questions to assess reading comprehension, vocabulary, and effects of tutorial interventions. *Technology Instruction Cognition and Learning*, 2:97–134.

Jack Mostow and Hyeju Jang. 2012. Generating diagnostic multiple choice comprehension cloze questions. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*. ACL.

Annamaneni Narendra, Manish Agarwal, et al. 2013. Automatic cloze-questions generation. In *Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013*.

John W Oller, J Donald Bowen, Ton That Dien, and Victor W Mason. 1972. Cloze tests in english, thai, and vietnamese: Native and non-native performance. *Language Learning*, 22(1):1–15.

Eyal Peer, Joachim Vosgerau, and Alessandro Acquisti. 2014. Reputation as a sufficient condition for data quality on amazon mechanical turk. *Behavior research methods*, 46(4):1023–1031.

Juan Pino, Michael Heilman, and Maxine Eskenazi. 2008. A selection strategy to improve cloze question quality. In *Proceedings of the Workshop on Intelligent Tutoring Systems for Ill-Defined Domains. 9th International Conference on Intelligent Tutoring Systems, Montreal, Canada*.

Emily Pitler, Annie Louis, and Ani Nenkova. 2010. Automatic evaluation of linguistic quality in multi-document summarization. In *Proceedings of the 48th annual meeting of the ACL*, pages 544–554. ACL.

Emily Pitler and Ani Nenkova. 2008. Revisiting readability: A unified framework for predicting text quality. In *Proceedings of the conference on empirical methods in natural language processing*, pages 186–195. ACL.

Kevin M Quinn, Burt L Monroe, Michael Colaresi, Michael H Crespin, and Dragomir R Radev. 2010. How to analyze political attention with minimal assumptions and costs. *American Journal of Political Science*, 54(1):209–228.

Earl F Rankin and Joseph W Culhane. 1969. Comparable cloze and multiple-choice comprehension test scores. *Journal of Reading*, 13(3):193–198.

Elissa M Redmiles, Sean Kross, and Michelle L Mazurek. 2019. How well do my results generalize? comparing security and privacy survey results from mturk and web panels to the us. In *IEEE Symposium on Security and Privacy*.

Luz Rello, Martin Pielot, and Mari-Carmen Marcos. 2016. Make it big!: The effect of font size and line spacing on online readability. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM.

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Loukia Sarroub and P David Pearson. 1998. Two steps forward, three steps back: The stormy history of reading comprehension assessment. *The Clearing House*, 72(2):97–105.

Jeff Sauro and Joseph S Dumas. 2009. Comparison of three one-question, post-task usability questionnaires. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1599–1608. ACM.

Cyrus Shaoul. 2010. The westbury lab wikipedia corpus. *Edmonton, AB: University of Alberta*.

Jennifer Slegg. 2018. Google's use of readability, reading level & vocabulary metrics in search algorithms.

Saku Sugawara, Yusuke Kido, Hikaru Yokono, and Akiko Aizawa. 2017. Evaluation metrics for machine reading comprehension: Prerequisite skills and readability. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 806–817.

Chenhao Tan, Vlad Niculae, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2016. Winning arguments: Interaction dynamics and persuasion strategies in good-faith online discussions. In *Proceedings of the 25th international conference on world wide web*, pages 613–624. International World Wide Web Conferences Steering Committee.

Wilson L Taylor. 1953. Cloze procedure: A new tool for measuring readability. *Journalism Bulletin*, 30(4):415–433.

Wilson L Taylor. 1956. Recent developments in the use of "cloze procedure". *Journalism Quarterly*, 33(1):42–99.

Chaffai Tekfi. 1987. Readability formulas: An overview. *Journal of documentation*, 43(3):261–273.

Tuck Meng Tham. 1987. *Linguistic variables as predictors of Chinese text readability*. Ph.D. thesis.

Sowmya Vajjala and Detmar Meurers. 2013. On the applicability of readability models to web texts. In *Proceedings of the Second Workshop on Predicting and Improving Text Readability for Target Reader Populations*, pages 59–68.

# News2vec: News Network Embedding with Subnode Information

**Ye Ma**[1], **Lu Zong**[1*], **Yikang Yang**[2] **and Jionglong Su**[1]
[1]Mathematical Sciences, Xi'an Jiaotong-Liverpool University, Suzhou, 215028, China
[2]Mathematics and Statistics, Georgetwon University, Washington DC, 20007, US
{ye.ma, lu.zong, jionglong.su}@xjtlu.edu.cn
yy494@georgetown.edu

## Abstract

With the development of NLP technologies, news can be automatically categorized and labeled according to a variety of characteristics, at the same time be represented as low dimensional embeddings. However, it lacks a systematic approach that effectively integrates the inherited features and inter-textual knowledge of news to represent the collective information with a dense vector. With the aim of filling this gap, the News2vec model is proposed to allow the distributed representation of news taking into account its associated features. To describe the cross-document linkages between news, a network consisting of news and its attributes is constructed. Moreover, the News2vec model treats the news node as a bag of features by developing the Subnode model. Based on the biased random walk and the skip-gram model, each news feature is mapped to a vector, and the news is thus represented as the sum of its features. This approach offers an easy solution to create embeddings for unseen news nodes based on its attributes. To evaluate our model, dimension reduction plots and correlation heat-maps are created to visualize the news vectors, together with the application of two downstream tasks, the stock movement prediction and news recommendation. By comparing with other established text/sentence embedding models, we show that News2vec achieves state-of-the-art performance on these news-related tasks.

## 1 Introduction

News, carrying a large amount of information, can often guide public opinions, affect people's behavior and drive the evolution of social events. In the era of information, news text is considered to be a part of big data that is continuously updated. In order to facilitate the performance of downstream NLP tasks, it is rather essential to find an efficient way to represent news as continuous vectors that contain the collective information of its inherited features and the inter-textual knowledge between different news.

The most straightforward approach to embed news is to directly treat it as textual data and apply text/sentence embedding models. To represent texts/sentences as vectors, one of the classic techniques is to perform numerical operations on the word vectors (Mikolov et al., 2013b), which is recognized as a simple but powerful baseline (Conneau et al., 2018). SDAE (Vincent et al., 2010) uses an auto-encoder to compress texts into a low-dimensional vector. Paragraph Vector (Le and Mikolov, 2014) learns the distributed representation of sentences and documents by predicting their contexts, *i.e.*, words in the sentence/document. Additionally, Skip-Thought (Li and Hovy, 2014), FastSent (Hill et al., 2016) and Quick-Though (Logeswaran and Lee, 2018) learn distributed sentence-level representations through the coherence between sentences. BERT (Devlin et al., 2018) proposes a powerful pre-trained sentence encoder which is trained on unsupervised datasets based on the masked language model and next sentence prediction. The major limitation of embedding approaches discussed above is that they produce representations and features that solely describe the contextual information at the document level. In the case of news representation, the connection between different news events is also considered as crucial information that should be incorporated into the news embedding, as well as other labeled features such as the topic category and the polarity of the news.

With the argument that news stories describe events, news event embedding models are developed to offer an alternative solution for the vector representation of news. It is suggested to extract event tuples $E = (Actor, Predicate, Object)$ from headlines and learn vector representations



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4843–4852,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

by scoring the correct tuples higher than the corrupted ones (Ding et al., 2015, 2016). Event2vec (Setty and Hose, 2018) is constructed on a classified news event database from Wikipedia[1], where an event network is created by considering events, entities, event types and years as nodes of different types. The Event2vec model thus produces distributed vector representations of news events based on the network embedding mechanisms, which lacks the flexibility of generating vectors for nodes outside the trained network. In addition, the weights of some edges in the event network are determined by objective assumptions since types and years are not comparable.

In this study, the News2vec model is proposed to learn distributed representations of news, with the embedded vectors containing not only the semantic and labeled information, but also the latent connections between different news events. We start by building the news network that connects the news nodes with their corresponding event element nodes in order to create the aggregated news context that describes both the semantic information and the background linkages between different news events (See Figure 1 for a simplified example of the news network). According to Figure 1, the two pieces of news $Alibaba\ reinvested\ Suning\ after\ three\ years$ and $Suning's\ second-half\ profit\ rose\ by\ 5\%$ are connected by sharing the same contextual element $Suning$. The verbal keywords $reinvest$ and $rise$ of the two different news are thus connected through their common node $Suning$ and become the latent contextual element of the other news. Based on the constructed news network, the biased random walk approach (Grover and Leskovec, 2016) is adopted to generate adequate number of connected node sequences which are later used in the network embedding. Inspired by the Subword model (Bojanowski et al., 2016), we further represent each news node as a bag of news features including semantic features, categorical features, time features, *etc.*. The vector representation of the news feature is thus obtained based on the skip-gram architecture (Mikolov et al., 2013a), and the news vectors are computed as the sum of its feature embeddings.

The advantages of the proposed News2vec model are twofold. First, the News2vec embeddings capture both the semantic features and the potential connections between news by constructing the news network. Second, the Subnode model offers an easy solution to create embeddings for unseen (news) nodes outside the trained network, at the same time enriches the node vector with its node attributes (news features). As the experiments suggest, the News2vec model achieves state-of-the-art performance on the news-related downstream tasks, namely the stock movement prediction and news recommendation.

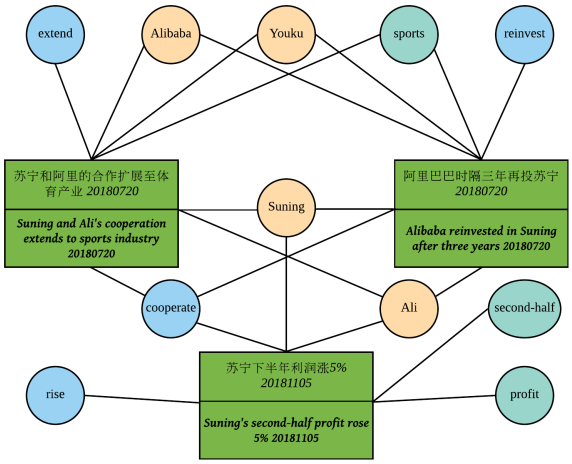


Figure 1: Simplified sample of the news network

## 2 The News2vec Model

In this section, the formation of the News2vec model is discussed in terms of the news network construction and news embedding based on the Subnode model. According to Figure 2, news elements including entities, actions and nouns, are first extracted from the news titles and texts. Based on these elements and their term frequencyinverse document frequency (tf-idf), the news network is built and sequences of nodes are sampled by the biased random walk. News nodes in these sequences are then represented as bags of extracted news features. The associated vector is thus assigned to each feature by the Subnode model.

### 2.1 News Network

The News2vec model creates news embeddings based on the news network that connects news with their elements. With the argument that a piece of news often describes one or several events which could be represented as a collection of elements, this study extracts news elements as enti-

[1] https://en.wikipedia.org/wiki/Portal:Currentevents

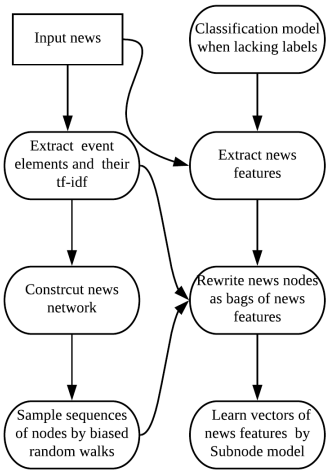


Figure 2: Flow chart of creating news embeddings with News2vec

ties, actions and nouns from news titles and texts, with respect to their tf-idf scores. Specifically, the tf-idf is a well recognized tool to measure the importance of a word/phrase inside the document and to extract the representative elements at the document level. The tf-idf score is formulated as:

$$tfidf_{k,e} = \frac{n_{k,e}}{\sum_w n_{w,e}} \times \log \frac{|D|}{|e : k \in d_e|}, \quad (1)$$

where $n_{k,e}$ denotes the number of the news element $k$ in the news story $e$, whereas its denominator is the total number of words/phrases in the news document. $|D|$ is the total number of documents, the denominator represents the number of stories containing the element $k$. In a nutshell, the tf-idf assigns a higher score to the element which appears more frequently in the news story than in others.

After determining the elements of news, the news network is hence constructed by connecting the news nodes $e \in V_e$ of the news title and published time with its element nodes $k \in V_k$. The weights of connective edges $(e, k)$ are computed based on the importance of the element to the news, which is determined by their tf-idf scores. The weight of an edge $(e, k)$ is computed as:

$$W_{e,k} = \frac{tfidf_{k,e}(title)}{Z_1} + \frac{tfidf_{k,e}(content)}{Z_2}, \quad (2)$$

where $Z$ is a normalization constant.

## 2.2 Network Embedding with Subnode Information

### Network embedding

The New2vec network embedding is based on the Node2vec (Grover and Leskovec, 2016) model, with the objective to learn a latent feature vector $F(v)$ for each node $v \in V$ that maximizes the probability of predicting the node $v's$ network neighborhood $N(v)$. The objective function is written as:

$$\max \sum_{v \in V} \log Pr(N(v)|F(v)). \quad (3)$$

Given the feature vector of node $v$, it is assumed that the prediction probabilities of its neighborhoods are independent from each other, which leads to:

$$Pr(N(v)|F(v)) = \prod_{u \in N(v)} Pr(u|F(v)). \quad (4)$$

To solve the objective function and obtain the optimized node representations, the skip-gram architecture (Mikolov et al., 2013a) is adopted on the basis of network neighboring sequences sampled by the biased random walk. To reduce the computational cost, negative sampling (Mikolov et al., 2013b) is used to replace the softmax classifier by multiple logistic binary classifiers. The optimizer is the Stochastic Gradient Descent (SGD).

Different from the uniform random walk of DeepWalk (Perozzi et al., 2014), News2vec uses the Breadth-first Search (BFS) and the Depth-first Search (DFS) as its search strategies. Suppose that the walk just transitioned from $t$ to $v$ and is now walking to its next step node $x$, BFS determines the next step with a greater chance of revisiting $t$, *i.e.*, $d_{tx} = 0$, whilst DFS drives the walk to go deeper and further away from the node $t$, *i.e.*, $d_{tx} = 2$. In line with the Node2vec model, News2vec defines a search bias $\alpha$ to control the search preference.

$$\alpha_{vx} = \begin{cases} \frac{1}{p}, & if\ d_{tx} = 0 \\ \frac{1}{q}, & if\ d_{tx} = 2 \end{cases} \quad (5)$$

Next, the transition probability of edge $(v, x)$ is defined as the product of the weight of the edge and the bias, *i.e.*, $\frac{\alpha_{vx} \times w_{vx}}{Z}$ where $Z$ is a normalization constant. Explicitly, if $p < q$, the walk is width-first which indicates that there is a greater chance to revisit the original node and sample the

nodes around it, that is, semantically similar news. If $p > q$, the walk is depth-first and tends to go to nodes that are not passed before, which leads to the exploration of news with latent relations. During the random walk, each node is used as a starting vertex for a fixed-length walk to produce a group of sequences. Based on the skip-gram model, we can train vector representations of nodes on these sequences.

**Subnode model**

Classic skip-gram model assigns a distinct vector to each word, neglecting the morphology information of words. With the purpose of addressing this problem, the Subword (Bojanowski et al., 2016) model represents each word using a bag of character $n-$grams, and the word vector is thus the summation of its character $n-$grams. The major advantage of the Subword model is that it allows the computation of word vectors that are not in the trained corpus by using their character $n-$gram vectors. Inspired by the Subword model, we introduce the Subnode model to solve network embedding problems by offering a solution to create inferential vectors of unseen news nodes outside the trained network, at the same time incorporate node attributes to the node vectors.

The biased random walk produces sequences of news and event element nodes from the news network. The proposed Subnode model expresses each of the news node as a bag of subnode information of its associated features including semantic features such as event elements with high tf-idf scores, text structure features such as words count and paragraphs count, and side information such as published date, news type and emotion. As a result, each news vertex $e \in V_e$ is represented as:

$$G =< entity_A, entity_B, action, \cdots,$$

$$words\ count, paragraph\ count,$$

$$month, day, week, type, sentiment, \cdots >$$

where word count, paragraph count and sentiment are ordinal data rather than real values. Further, the Subnode model represents a news node as the sum of its features, instead of a distinct vector obtained from the other network embedding models. Therefore, the objective function of news network embedding is expressed as:

$$max \sum_{e \in V_e} \log Pr(N(e) | \sum_{g \in G_e} f(g)) \qquad (6)$$

where $f(g)$ denotes the vector representation of a news feature. In addition, we remove nodes with only one edge to reduce the sparsity of the network. As it is mentioned in the previous section, the Subnode vector representations of news features are learned based on the skip-gram architecture by optimizing the updated objective function using SGD with negative sampling (Mikolov et al., 2013b). With an adequately large news network that contains a wide range of features, the Subnode mechanism of News2vec allows the vector representation of an unseen news node directly computed by extracting news features and summing up their corresponding vectors according to the $feature\ to\ vector$ dictionary. See Github[2] for the codes and examples of News2vec.

## 3 Experiments

Four experiments are implemented and explored in this section. In particular, we first visualize the News2vec embeddings in terms of the news vectors and feature vectors with the dimension reduction plot and the correlation heat maps. Two downstream experiments, *i.e.* the stock movement prediction and news recommendation, are then conducted to examine the News2vec model in comparison with other established text/sentence embedding models. Overall, News2vec achieves state-of-the-art performances in these news-related tasks which demonstrates its validity of addressing potential relevance of news via the network, as well as the integrated consideration of the news features.

### 3.1 Visualization of News Vectors

In this section, the THUCTC[3] (THU Chinese Text Classification) news data set is used to train the vectors of news features, including the semantic features and four additional features, namely $type$, $words\ count$, $paragraph\ count$ and $sentiment$. We determine the sentiment by counting positive words over negative words. There are 14 types of news in the news corpus and we randomly take 6,000 pieces of news from each type as the training set. In the test set, we additionally extract 1,000 pieces of news from each type. Based on these 84,000 pieces of news in the training set, we learn a vector representation of 128 dimensions for each news feature. The news vectors in the test set are

[2] https://github.com/yema2018/News2vec
[3] http://thuctc.thunlp.org

then computed, whereas their dimension reduction plots are produced using the t-Distributed Stochastic Neighbor Embedding (t-SNE).

Figure 3 shows the dimension reduction results of news vectors in the test set. According to the right panel, a high level of clustering is observed, indicating that News2vec allows the embedding of type information into news vectors after incorporating the type feature. Moreover, the distance between two clusters is in-line with the relevancy between the clustered topics in reality. In particular, neighboring relationships are exhibited between similar topics such as sports & lottery ticket, current affairs & society, stock & finance, and fashion & entertainment, whilst the distance between less relevant clusters, such as constellation & finance, game & education, is relatively larger. Nevertheless, the left panel of Figure 3 shows that news embeddings significantly lose classification information once vectors of the type feature are removed from the news vector, which shows that News2vec enriches the news vector representation by the news network and the Subnode information.

### 3.2 Correlations of News Features

In this section, we further investigate the vector correlation between three news features, namely the $type$, the $words\ count$ and the $sentiment$. The correlation coefficients (*i.e.*, normalized cosine similarity) heat maps for each feature are displayed in Figure 4.

The correlation heat map of news type features (upper panel) supports the conclusion in Section 3.1. It is observed that vector pairs of close topics such as $type : Stock$ and $type : Finance$ are highly correlated, indicating stock news and financial news share similar contextual elements. Other highly correlated pairs include sports and lottery ticket news, game and technology news, fashion and entertainment news, etc., whilst less relevant topics show low correlations, such as real estate and constellation, sports and real estate. According to the bottom left panel of Figure 4, a significant number of dark blocks are observed in the area from $wc200$ to $wc3000$ ($wc200$ means the news is less than 200 words and $wc2000$ means the news is between 1000 and 2000 words), and there is a clear reduction of correlation once the news exceeds 3,000 words. As it is indicated by the word count correlation heat map, news that has similar numbers of words is more likely to have similar content. The correlation reduction from $wc3000$ to $wc5000$ suggests that there is a group of news with specific content, that is often discussed in long articles. Meanwhile, the right bottom of Figure 4 shows that news with close sentiment levels tends to have higher correlations. As the sentiment moves from more positive to more negative, the correlation decreases. Overall, the correlation heat maps show that the News2vec feature vectors are reliable in terms of expressing Subnode information.

### 3.3 Stock Movement Prediction

**Experimental settings**

We use financial news (2009.10.19 to 2016.10.31) from Sohu [4] to predict the daily movement of Shanghai securities composite index. The news feature vectors are trained based on news from 2009.10.19 to 2015.12.31 and news vectors are computed by summing up these feature vectors. There are five additional news features, namely $month$, $week$, $sentiment$, $words\ count$ and $day$.

The length of the walk is fixed at 100, the context size is 10, return hyper-parameter ($p$ in Equation (5)) and input hyper-parameter ($q$ in Equation (5)) are both set to 1.

**Predictive model**

The predictive model is the long short term memory (LSTM) network (Hochreiter and Schmidhuber, 1997) with self-attention mechanism (Yang et al., 2016). As shown in 5, the LSTM model has $T$ time steps (*i.e.*, $T$ days) in total. At each time step, the input is the weighted average of news vectors in that day. Weights are initialized at the beginning and updated by the gradient descent. Based on the attention model, news that is closely related to the stock price movement is assigned with higher weights.

In this paper, we use the previous 20 days' ($T = 20$) news to predict the next day's stock index movement. The output layer is a Softmax classfier that indicates whether the index goes UP (rise percent $> 0.33\%$), DOWN (rise percent $< -0.29\%$) or PRESERVE ($-0.29\% <$ rise percent $< 0.33\%$). We employ a rolling window with $size = 20$ and $strider = 1$ to augment the sample. News before 2016 is used as the training set, whilst news after 2016 is used as the test-validation set.

[4] https://www.jianshu.com/p/370d3e67a18f

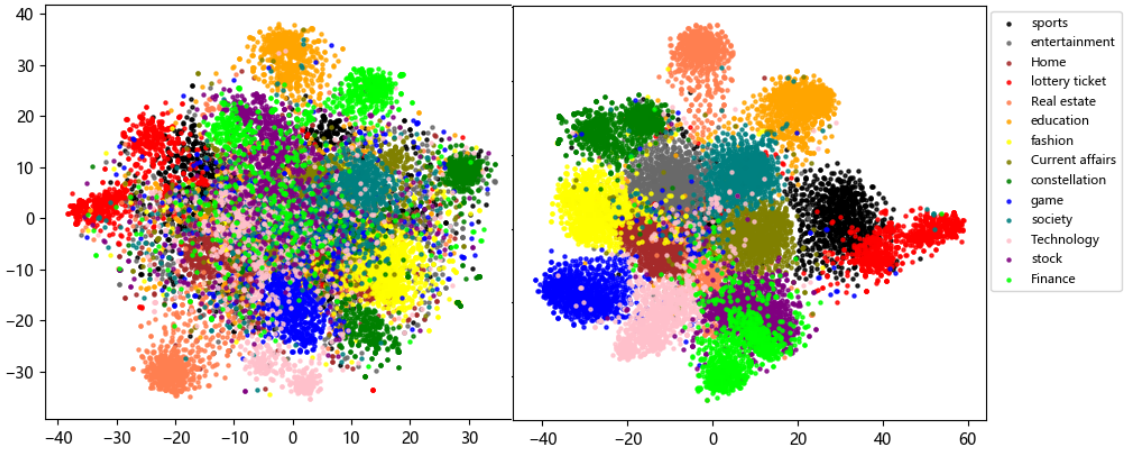


Figure 3: Dimension reduction plots of test set news without and with the type feature

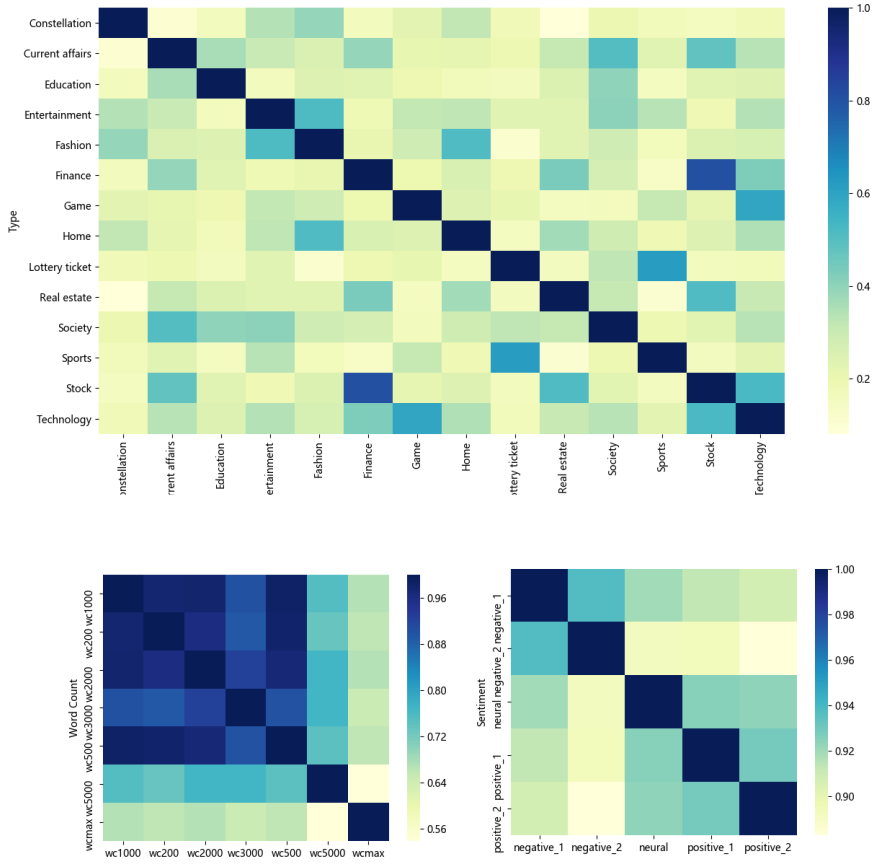


Figure 4: Correlations of news features (Upper panel: news type. Lower left: word count. Lower right: sentiment.

## Results

As for the baseline models, we include several established embedding approaches to compute news vectors in the experiment of the stock movement prediction:

**Average BOW**: News embedding is represented as the average of word vectors in the news titles and bodies (Hu et al., 2018). Word embedding is obtained by training the skip-gram Word2vec model (Mikolov et al., 2013a).

**Doc2vec: Paragraph Vector** (Le and Mikolov, 2014): News texts are represented as dense vectors by the Paragraph Vector model(Akita et al., 2016).

**Event embedding** (Ding et al., 2015, 2016): Event tuples are extracted from headlines and represented as vector representations by scoring the correct tuples higher than the corrupted tuples.

In addition, we apply a sentence-level encoder

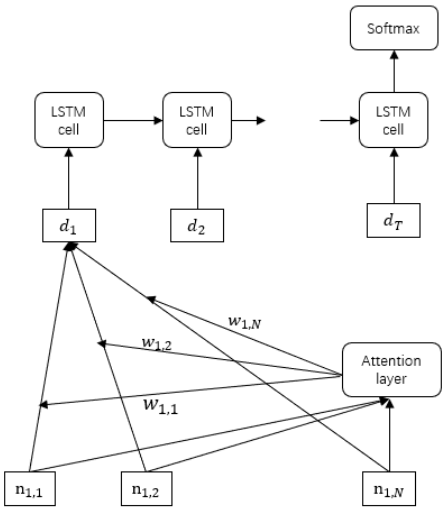


Figure 5: LSTM model with attention mechanism

BERT to encode news headlines as another baseline model:

**BERT** (Devlin et al., 2018): Average pooling is used to get a fixed representation of a headline based on the Chinese pre-trained model[5].

The proposed News2vec model is implemented separately with and without the five additional features (*i.e.* $month$, $week$, $sentiment$, $words\ count$ and $day$):

**News2vec-**: News vectors are the summation of solely the semantic features (*i.e.*, event elements).

**News2vec+**: News vectors are the summation of all news features including the semantic and the five additional features.

| | Acc | MCC |
|---|---|---|
| Average BOW($d = 128$) | 41.91% | 0.1137 |
| Doc2vec($d = 128$) | 43.90% | 0.1484 |
| Event embedding($d = 128$) | 43.76% | 0.1439 |
| BERT($d = 764$) | 45.83% | 0.1442 |
| News2vec-($d = 128$) | 44.68% | 0.1664 |
| News2vec+($d = 128$) | **46.24%** | **0.1936** |

Table 1: Acc and MCC Results of stock movements in the test set

In Table 1, test results are presented according to two measurements, namely the accuracy (Acc) and the Matthews Correlation Coefficient (MCC). We find the News2vec model outperforms BERT and all the other baseline models both in terms of the Acc and MCC, with a lower dimension of 128 (The dimension of BERT embeddings is 764). As the MCC suggests, Doc2vec, Event embedding and BERT produce stock movement predictions of similar qualities as the news vectors generated by the three approaches uniformly limit to the semantic information of news articles. As for news-driven stock price prediction, it is more reasonable to take into account the potential connections between news, such as having similar background story or leading to the same event, and reflect the connection by embed similar values in certain dimensions of the news vectors. In the case of the New2vec+ model, we find that extra features contribute to the improvement of the MCC, indicating the New2vec representations are able to capture the latent connections between news events. After considering all news features, the results are improved by 0.028, showing their positive effect on stock movement predictions. It is worth-mentioning that the New2vec model is performed in an unsupervised manner as all features are extracted from common news text without any artificial label or trained classifier. We believe that there is a large chance to further improve the result once artificial labels such as the polarity are incorporated.

### 3.4 News Recommendation

**Experiment settings**

In this section, a news recommendation task is implemented based on the data of news browsing records (2014.2 to 2014.3) using the news embedding models [6]. The obtained data set contains 16214 news browsing sequences in total, which is split into halfs for testing and fine-tuning, respectively. The overall objective is to recommend news based on the content by computing the cosine similarities of the news vectors.

News recommendations are made for each news by selecting the top ten news with the highest vector similarities. The recommendation is considered to be successful as long as the next browse in the sequence belongs to the ten news articles selected. We compute the success rate for each sequence. The average success rate of all sequences is the final evaluation result for the embedding model.

Since the news titles are absent in the data set, event elements are only extracted from news bodies. As a result, we only use the text-level em-

[5] https://github.com/google-research/bert#pre-trained-models

[6] https://github.com/YLonely/web-data-mining

bedding models (*i.e.*, Paragraph Vector and Average BOW) as the baselines to compare with the News2vec model. We use the trained news feature vectors in the previous section as the initial inputs of the new news network (Sohu+rec). Moreover, half of the news sequences are employed to fine-tune these vectors (Sohu+rec+seq). Due to the lack of information, only $sentiment$ and $word\ counts$ are included as the extra news features together with the existing semantic features.

**Evaluation of news recommendation**

| | Success rate |
|---|---|
| Paragraph Vector | 4.31% |
| Average BOW | 3.09% |
| Sohu (no extra features) | 3.36% |
| Sohu+rec (no extra features) | 3.69% |
| Sohu+rec+seq (no extra features) | 6.73% |
| Sohu | 2.95% |
| Sohu+rec | 3.61% |
| Sohu+rec+seq | **7.53%** |

Table 2: Evaluation results of news recommendation

From Table 2, it is observed that the success rate improves significantly after fine-tuning the news vectors with respect to the news browsing sequences. In comparison with Paragraph Vector, News2vec (Sohu+rec) demonstrates no significant advantage without sequence fine-tuning (Sohu+rec+seq), indicating potential connections between news have limited effect on this task. Nonetheless, News2vec achieves state-of-art performance after sequence fine-tuning. On the other hand, the result of Paragraph Vector cannot be further improved by fine-tuning. A possible explanation is that the Paragraph Vector model assigns a distinct vector to each of the news articles, thus inconsistency between news in different news browsing sequences is easily created by the sequence-level fine-tuning. On the other hand, fine-tuned News2vec embeddings carry the information of both the news network and the browsing sequences by treating the browsing sequence as a special sequence generated by the biased random walk. In addition, it is found that with fine-tuning, the additional features tend to have positive effects on the recommendation task (see Sohu to Sohu+rec+seq without extra features and Sohu to Sohu+rec+seq with extra features). As a matter of fact, additional news features, such as the release time and type, are considered to contain important information for news recommendation tasks. Unfortunately, due to the lack of data, only two extra features are included in this experiment.

## 4 Related Work

In most cases, news embeddings are directly created by sentence/text embedding approaches (Ding et al., 2015; Hu et al., 2018; Vargas et al., 2017; Akita et al., 2016), that are often divided into two groups: the unsupervised and the supervised models. For unsupervised learning, the auto-encoder model is an early solution to compress text data (Vincent et al., 2010). Inspired by word embedding, Paragraph Vector (Le and Mikolov, 2014; Li et al., 2016) represents texts or sentences as vectors based on the co-occurrence relation between words and documents, whilst Skip-Thought (Li and Hovy, 2014), FastSent (Hill et al., 2016) and Quick-Though (Logeswaran and Lee, 2018) are based on the coherence between sentences. For supervised learning-based models, most methods use sentence encoder such as LSTM or Transformer (Vaswani et al., 2017) to encode word vectors into sentence vectors and train the encoder on various NLP tasks (Conneau et al., 2017; Cer et al., 2018).

## 5 Conclusions

In this paper, we develop the News2vec model that learns the vector representation of news articles by constructing a news network. The Subnode model is further proposed to allow the embedding of unseen (news) nodes outside the existing network, at the same time to enrich the news vector with its associated feature vectors. Compared to other established text embedding models, the News2vec embedding contains not only the information of the contextual relationship between news and its event elements, but also potential connections as the network goes deeper. According to the dimension reduction plots and correlation heat-maps, it is suggested that the news/feature vectors contain adequate information as expected. Two downstream tasks, the news-driven stock movement prediction and news recommendation, show the News2vec embeddings demonstrate the latent connections between news articles, and the integration of news features enhances the model's performance in comparison to the baseline models.

## Acknowledgments

The authors would like to acknowledge the support by 2016 Jiangsu Science and Technology Programme: Young Scholar Programme (No. BK20160391).

## References

R. Akita, A. Yoshihara, T. Matsubara, and K. Uehara. 2016. Deep learning for stock prediction using numerical and textual information. In *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, pages 1–6.

P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. 2016. Enriching Word Vectors with Subword Information. *arXiv e-prints*.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder. *CoRR*, abs/1803.11175.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *CoRR*, abs/1705.02364.

Alexis Conneau, Germán Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single vector: Probing sentence embeddings for linguistic properties. *CoRR*, abs/1805.01070.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. 2015. Deep learning for event-driven stock prediction. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 2327–2333. AAAI Press.

Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. 2016. Knowledge-driven event embedding for stock prediction. In *COLING*.

Aditya Grover and Jure Leskovec. 2016. Node2vec: Scalable feature learning for networks. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 855–864, New York, NY, USA. ACM.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. *CoRR*, abs/1602.03483.

Sepp Hochreiter and Jrgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Ziniu Hu, Weiqing Liu, Jiang Bian, Xuanzhe Liu, and Tie-Yan Liu. 2018. Listening to chaotic whispers: A deep learning framework for news-oriented stock trend prediction. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, WSDM '18, pages 261–269, New York, NY, USA. ACM.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *CoRR*, abs/1405.4053.

Bofang Li, Zhe Zhao, Tao Liu, Puwei Wang, and Xiaoyong Du. 2016. Weighted neural bag-of-n-grams model: New baselines for text classification. In *COLING*.

Jiwei Li and Eduard Hovy. 2014. A model of coherence based on distributed sentence representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2039–2048, Doha, Qatar. Association for Computational Linguistics.

Lajanugen Logeswaran and Honglak Lee. 2018. An efficient framework for learning sentence representations. *CoRR*, abs/1803.02893.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 701–710, New York, NY, USA. ACM.

Vinay Setty and Katja Hose. 2018. Event2vec: Neural embeddings for news events. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, pages 1013–1016, New York, NY, USA. ACM.

M. R. Vargas, B. S. L. P. de Lima, and A. G. Evsukoff. 2017. Deep learning for stock market prediction from financial news articles. In *2017 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA)*, pages 60–65.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, California. Association for Computational Linguistics.

# Recursive Context-Aware Lexical Simplification

**Sian Gooding**
Dept of Computer Science and Technology
University of Cambridge
shg36@cam.ac.uk

**Ekaterina Kochmar**
ALTA Institute
Unversity of Cambridge
ek358@cam.ac.uk

## Abstract

This paper presents a novel architecture for recursive context-aware lexical simplification, REC-LS, that is capable of (1) making use of the wider context when detecting the words in need of simplification and suggesting alternatives, and (2) taking previous simplification steps into account. We show that our system outputs lexical simplifications that are grammatically correct and semantically appropriate, and outperforms the current state-of-the-art systems in lexical simplification.

## 1 Introduction

Text simplification (TS) is aimed at reducing the reading and grammatical complexity of text while retaining the meaning and grammaticality (Chandrasekar and Bangalore, 1997). This is usually achieved by a series of transformations at the lexical and syntactic level. A number of systems in the recent years have approached this task in an integral manner (Zhu et al., 2010; Kauchak, 2013; Zhang and Lapata, 2017). Such comprehensive systems can perform a number of simplification operations at once, but the results are sometimes ungrammatical and meaning can be changed, arguably making the original text less clear and more complex (Siddharthan, 2014).

In this paper, we assume the lexical and syntactic components of a TS system to be independent and complementary to each other, and focus on the *lexical simplification* (LS) component for a number of reasons. First, it has been shown that *lexical simplification* techniques positively impact the readability of text and improve reader understanding and information retention (Leroy et al., 2012). Secondly, it has been argued that a large number of people with reading difficulties, including those with disabilities, low-literacy, non-native backgrounds or non-expert knowledge benefit from LS (Xu et al., 2015). For instance, James (1998) shows that vocabulary of the non-native language plays central role in second language acquisition. As we aim to make information more accessible to such readers, the quality of the simplified text is of a paramount importance.

We stress the importance of three key points of quality assessment in LS: it is vital for the simplified text to be *of a lower complexity*, while being *semantically equivalent* to the original, and *grammatically correct*. Context plays a central role in fulfilling these requirements. For instance, consider the different uses of *situation* in the complex word dataset collected by Yimam et al. (2017):

(1) the gravity of the **economic *situation***

(2) the *situation* has remained unchanged

This example demonstrates two types of contextual effects: first of all, in (1) both *economic* and *situation* are marked as complex in mutual context, but when *situation* occurs in a different context it is not annotated as complex. Secondly, this example illustrates the impact of the context on the choice of the appropriate substitution: substituting *climate* for *situation* will work for (1), but will result in a semantically different expression in (2).

In addition, we argue that as word complexity depends on context, the order and choice of applied simplifications matters. For instance, consider the following simplifications for the sentence '*This is a problem in the contemporary world*':

(3) This is a problem in the *modern earth*

(4) This is a problem in the *modern* world

Example (3) shows an output of a system that tries to simplify all words, which results in a nonsensical sentence, while (4) exemplifies the result of recursive word replacement in context.

Context effects in LS have not been thoroughly investigated before. In this paper, we introduce a



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4853–4863,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

novel approach to LS that addresses these issues. In particular, we make the following contributions:

1. As each simplification step changes the complexity of the output sentence, our LS algorithm applies simplification *recursively taking word complexity in context into account*;
2. To ensure grammaticality and meaning equivalence to the input in the output, our algorithm *takes context both at the complex word identification and substitution selection steps into account*. We use a novel sequence labelling component for the former step, and assess semantic equivalence at the latter using deep contextualized word representations provided by ELMo (Peters et al., 2018).
3. To facilitate reproducibility, we release the code and the output of our system at `github.com/siangooding/lexical_simplification`.

## 2 Previous work

### 2.1 Approaches

Early approaches to TS have mostly relied on rule-based systems (Carroll et al., 1998; Canning et al., 2000; Siddharthan, 2006), with many of the earlier systems prioritising syntactic operations, such as sentence splitting, deletion or reordering. Some work combined lexical simplification with syntactic operations (Zhu et al., 2010; Coster and Kauchak, 2011a; Kauchak, 2013). The availability of parallel corpora of "normal" and simplified text has inspired a number of approaches that treated TS as a monolingual machine translation problem (Zhu et al., 2010; Coster and Kauchak, 2011a,b) or allowed the researchers to apply language modelling (Kauchak, 2013).

Building on this line of work, Zhang and Lapata (2017) combine a novel sequence-to-sequence encoder-decoder model with a deep reinforcement learning framework that rewards the system for providing simple, fluent output, similar in meaning to the input. However, one of the main challenges for such comprehensive end-to-end systems is the ability to address specific types of errors independently. For instance, the DRESS-LS model (Zhang and Lapata, 2017) sometimes changes the meaning of the input to the opposite as in "*Inspections, she said,* **rarely** *cost more than $ 1,400*" → "*Inspections, she said,* **often** *cost more than $ 1,400*", or produces nonsensical output as in "*Archaeologists digging on the* **grounds**" → *Archaeologists digging on the* **zebras**".

A number of approaches focused on generation and assessment of *lexical simplification* (Yatskar et al., 2010; Biran et al., 2011; Horn et al., 2014; Glavaš and Štajner, 2015). Paetzold and Specia (2016a) note the lack of consistent evaluation for text simplification, and in particular lexical simplification, and introduce an evaluation dataset BENCHLS, on which they perform benchmarking of several LS systems. They argue that lexical simplification consists of a number of steps, including substitution generation, substitution selection and ranking. They show that the unsupervised system of Paetzold and Specia (2016b) outperforms a range of other systems in all steps, with the only exception of the feature-based system by Horn et al. (2014), which performs better in terms of precision in a round-trip system evaluation.

Finally, Shardlow (2013) introduced complex word identification (CWI) as the first step in an LS pipeline to detect words within text that require simplification. He showed that systems' performance on this stage is crucial for the overall performance, as low recall of this component might result in an overly difficult text with many missed complex words, while low precision might result in meaning distortions. The recent shared task on CWI shows that most systems rely on classification approaches using features that pertain to individual words, not taking wider context into account (Yimam et al., 2018).

### 2.2 Datasets

Until recently, parallel Wikipedia and Simple Wikipedia datasets have been the most widely used data for training and evaluating TS systems (Zhu et al., 2010; Yatskar et al., 2010; Coster and Kauchak, 2011a,b; Biran et al., 2011; Kauchak, 2013; Horn et al., 2014). Wikipedia allows free access to large quantities of data, and Simple Wikipedia represents a simplified version of original articles that uses simpler vocabulary and syntactic structures (Coster and Kauchak, 2011a). Researchers in the past applied monolingual alignment techniques to construct parallel versions of the two Wikipedias and learn the transformations from these parallel versions using such tools as GIZA++ (Och and Ney, 2000).

Despite the popularity of the Wikipedia-based

datasets for TS research, Xu et al. (2015) argue that focusing on Wikipedia limits simplification research and propose using a dataset based on news articles. They use a dataset from Newsela as an example, where the texts are simplified by professional editors at 4 levels of simplificity in accordance with the grade levels defined by the Common Core Standards (Porter et al., 2011).

In their benchmarking study, Paetzold and Specia (2016a) introduce an evaluation dataset BENCHLS that combines two previously released datasets for TS – LexMTurk (Horn et al., 2014) and LSeval (De Belder and Moens, 2012). This dataset contains 929 instances with an original sentence, a target complex word, and several candidate substitutions ranked by English speakers from the U.S. according to their simplicity. Paetzold and Specia (2016a) additionally filter out misspelled candidates and inflect all candidates to the grammatical form of the target word. The dataset contains an average of 7.37 candidate substitutions per complex word.

Finally, the CEFR-LS dataset (Uchida et al., 2018) contains simplifications, which not only represent a semantically good fit and are grammatically correct, but are also at different levels of simplicity annotated with respect to non-native speakers. Simpler candidates at lower levels of language proficiency (A1-B1) according to the Common European Framework of Reference for Languages (CEFR) (Council of Europe, 2011) are provided for the original words that are at higher levels of language proficiency (B2-C2). The dataset contains 406 target words and 4912 possible substitutions. Unlike BENCHLS, substitute candidates within this dataset may not be correct for the given context, and if a substitute candidate is not appropriate in context it is labelled as such. The dataset contains an average of 2.35 candidate substitutions per complex word.

## 3 Data

In this study we implement an LS system that includes complex word identification, substitute generation, filtering and ranking as steps in a simplification pipeline. To train our CWI system, we use the CWI 2018 shared task dataset (Yimam et al., 2017), which contains texts on three genres – professionally written NEWS, amateurishly written WIKINEWS, and WIKIPEDIA articles. The words in the dataset are annotated as complex or not by 10 native and 10 non-native speakers of English.

We evaluate each step of the LS pipeline on two datasets – BENCHLS and CEFR-LS. We select the BENCHLS dataset because it contains multiple simplification alternatives ranked with respect to their simplicity by a number of human annotators. The CEFR-LS dataset is a useful resource for evaluation because, in addition to contextually suitable, grammatically correct, simpler alternatives, it also contains substitution candidates that do not fit the context. Furthermore, this dataset is aimed at non-native speakers of English which we view as the future target group for our LS system. For further details on datasets collection and annotation, we refer the readers to the original papers.

Finally, we evaluate our LS system in an end-to-end manner and compare its performance to that of the current state-of-the-art systems, including those reported in Paetzold and Specia (2016a) and the DRESS-LS system of Zhang and Lapata (2017). In contrast to Zhang and Lapata (2017), we perform lexical simplification only. For fair comparison of the two systems, we extract only lexical simplifications from the parallel "normal" to simplified versions of the data used in Zhang and Lapata (2017), as well as from the original "normal" text and the DRESS-LS system output.

To extract the lexical transformations from the data, we use GIZA++ (Och and Ney, 2000) similarly to previous research (Coster and Kauchak, 2011b; Xu et al., 2015). Following Horn et al. (2014), we extract the examples that constitute one-to-one word correspondences between the two sides identified by the automatically induced word alignment, where the part-of-speech tag of the two words is the same while the lemmas are different. In addition, we filter out the instances involving modification of stopwords on the original, "normal" side as well as rewrites involving proper nouns. All preprocessing steps for our algorithm are performed using the RASP parser (Briscoe et al., 2006).

## 4 Lexical Simplification Pipeline

Lexical simplification can be viewed as a multi-stage process involving complex word identification, substitute generation, filtering and substitute ranking. In this section we outline each step of the simplification process as presented in Figure 1.

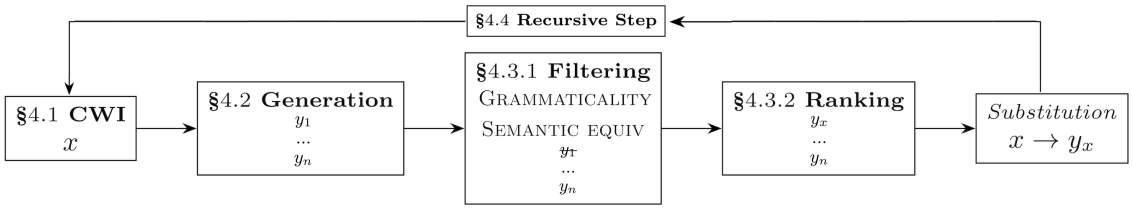


Figure 1: Recursive Simplification (REC-LS) pipeline: $x$ is a complex word and $y_i$ are the generated substitutes

### 4.1 Complex Word Identification (CWI)

The first stage of the algorithm is complex word identification (CWI), which aims to identify which words should be simplified within the text and, thus, allows for a personalisable and targeted approach to LS. Furthermore, it helps to reduce the number of unnecessary and potentially "harmful" simplifications performed by a 'simplify all words' approach. To train and test our CWI component we use the dataset by Yimam et al. (2017).

Word complexity depends on the surrounding context: for instance, between 3% and 10% of lexical items (depending on genre) in the Yimam et al. (2017) dataset receive different annotations in different contexts. However, most CWI systems to date have approached this task on an individual word basis (Yimam et al., 2018). For instance, the 2018 CWI shared task winning system CAMB uses a classification-based approach with 27 word-level features (Gooding and Kochmar, 2018). As context impacts the perceived complexity of text, we argue that CWI should be framed as a sequence labelling task and propose a novel architecture SEQ based on this idea (Gooding and Kochmar, 2019). We extend the implementation of a sequence labeller by Rei (2017),[1] which achieves state-of-the-art results on a number of NLP tasks. The design of this architecture is highly suitable for CWI as: (1) it uses bi-directional long short-term memory units (BiLSTM) (Hochreiter and Schmidhuber, 1997), which allow the system to learn about both the left and right context of a target word; (2) the context is combined with both word and character-level representations (Rei et al., 2016) which helps capture complexity due to rare character sequences as well as morphological structure; (3) this architecture uses a language modelling objective, which enables the model to take one of the highly informative complexity factors of word frequency into account.

[1] https://github.com/marekrei/sequence-labeler

| Test Set | Macro F-Score | |
|---|---|---|
| | CAMB | SEQ |
| NEWS | 0.8633 | **0.8763** |
| WIKINEWS | 0.8317 | **0.8540** |
| WIKIPEDIA | 0.7780 | **0.8140** |

Table 1: SEQ vs. CAMB system results on CWI

| DATASET | Precision | Recall | F-Score |
|---|---|---|---|
| CEFR-LS | 0.9327 | 0.7936 | 0.8575 |
| BENCHLS | 0.5000 | 0.4025 | 0.4460 |

Table 2: SEQ results for CWI on the two datasets

SEQ uses 300-dimensional GloVe embeddings as word representations (Pennington et al., 2014). The model is trained to predict the binary complexity of words as annotated in the dataset of Yimam et al. (2017). Training is performed over 20 iterations on randomly shuffled sentences from all genres included within the dataset. To test this novel architecture on CWI, we apply it to the CWI 2018 shared task test data (Yimam et al., 2018) and compare the results to the current state-of-the-art (SOTA) CAMB system. Table 1 shows that the SEQ model outperforms the current SOTA system on all three text genres for binary CWI (statistically significant using McNemar's test, $p$=0.0016, $\chi^2$=9.95).

The proposed SEQ model (Gooding and Kochmar, 2019) has a number of additional advantages: it takes context into account, helps avoid the necessity of extensive feature engineering relying on word embeddings as the only input information at run time, and generalises well across all three datasets. To further assess generalisability of the model, we test it on CEFR-LS, as well as BENCHLS for consistency (see Table 2). However, as the words in BENCHLS dataset were selected at random rather than according to their complexity (Paetzold and Specia, 2016a), the results as expected are lower.

#### 4.1.1 Lexical Complexity Threshold

The SEQ system labels each word with a lexical complexity score. This score represents the likelihood ($p$) of each word belonging to the complex class. Whether a word is considered as complex is set according to a predefined threshold for $p$, which allows the 'aggressiveness' of the algorithm to be tailored according to the application. For instance, consider the following example:

(5) I believe that ignoring public opinion ***discredits***$_{0.89}$ the *authorities*$_{0.79}$ and ***destabilises***$_{0.84}$ the situation.

The SEQ labeller assigns complexity probability $p$>0.80 to the words ***discredits*** and ***destabilises***, whereas for *authorities* $p$=0.79. If the complexity threshold in the simplification pipeline is set to 0.80, the word *authorities* would not be a candidate for lexical simplification. Our simplification pipeline will attempt to simplify words one at a time, starting with the highest $p$ values above this predefined threshold.

### 4.2 Substitute Generation

Substitute generation refers to the process of generating candidates that can be used as simpler alternatives to the target word. The goal of the system at this stage is to generate a diverse set of potential substitutions that can be filtered and ranked according to different criteria at later steps. We note that the use of multiple resources for substitute generation at this step is crucial as each individual resource has only a limited coverage. To this end, we first use traditional approaches, which do not take the context of the target word or the simplicity of substitutes into account. In summary, we rely on the following resources:

- we extract synonyms from WordNet (Fellbaum, 2012), following Devlin (1998);
- we use Big Huge Thesaurus[2] to find candidates by querying for the word form and lemma;
- following Glavaš and Štajner (2015), we use word embeddings to create substitutions. Synonyms are identified by finding 10 words with the highest cosine similarity to the target word in the vector space. These are obtained for both GloVe (Pennington et al., 2014) and Word2Vec embeddings (Mikolov et al., 2013), using the Gensim library.[3]

The candidate lemmas retrieved using these resources are added to the candidate substitutions set. To match the grammatical form of the original word, we convert all candidates to the appropriate word forms using NodeBox English Linguistics library.[4]

| | |
|---|---|
| $O$ | Water **engulfed**$_{0.89}$ Beringia. |
| $S_1$ | Water *overwhelmed*$_{0.78}$ Beringia. |
| $S_2$ | Water *flooded*$_{0.39}$ Beringia. |

Table 3: Contextual simplicity scores for 2 possible alternatives ($S_1$-$S_2$) for the original $O$

### 4.3 Substitution Filtering and Ranking

Following substitute generation, the next step in the LS pipeline is to choose one of the generated candidates to replace the original target word. This is done by filtering and ranking the candidates using a set of criteria, and choosing the top candidate. Previous work has framed this task as ranking words according to simplicity (Paetzold and Specia, 2016a). However, a system that is only aimed at selecting the simplest candidate may return a substitution that does not fit the surrounding context, is ungrammatical or not semantically equivalent to the original. Therefore, we consider all three aspects of high-quality substitution selection: *contextual simplicity*, *semantic equivalence* and *grammaticality*, which are outlined below.

**Contextual Simplicity**: The complexity of each candidate is calculated using our sequential CWI model from Section 4.1. We calculate the complexity for each substitution within context and refer to this as the simplicity score $S$. Table 3 shows 2 possible substitutions for *engulfed*, with their respective simplicity scores.

**Contextual Semantic Equivalence**: To assess whether a substitution is semantically equivalent to the original we use ELMo embeddings (Peters et al., 2018), which to the best of our knowledge have not been used for LS before. ELMo provides deep contextualized word representations, which are learned from the internal states of a deep bidirectional language model. As a result, these embeddings are able to model complex syntactic and

[2] https://words.bighugelabs.com

[3] https://github.com/RaRe-Technologies/gensim

[4] https://www.nodebox.net/code/index.php/Linguistics

| | |
|---|---|
| $O$ | This **illustrates** how sociologists [...] |
| $S_1$ | This *demonstrates*$_{0.10}$ how sociologists [...] |
| $S_2$ | This *shows*$_{0.16}$ how sociologists [...] |
| $S_3$ | This *draws*$_{0.44}$ how sociologists [...] |

Table 4: Contextual semantic equivalence scores for 3 candidate substitutions ($S_1$-$S_3$) for the original $O$

| | |
|---|---|
| $O$ | Oak is strong and also *gives* shade |
| g.c. | Oak *gives* shade |

Table 5: Original sentence $O$ and the grammatical context (g.c.) used to calculate $C_G$ for the word *gives*

semantic characteristics of word use, as well as how these word uses vary across linguistic contexts. In addition, contextualised embeddings help filter out antonyms that might be included in the set of potential substitutes by the previous step.

The similarity of words in a given sentential context is calculated using the ELMo embeddings associated with the target word and each of the substitutes. The contextual similarity score ($C_S$) is calculated by taking the cosine distance between the ELMo word vectors associated with the original word $v_o$ and the substitute $v_s$. This allows us to reason about the likelihood that the substitution is semantically equivalent to the original in the given context: the *smaller* the distance the *more likely* this word fits the context. Table 4 provides some examples for this score.

This technique works particularly well when the target word appears in the immediate context of the words grammatically related to it, for example when a verb is surrounded by its subject and object. To counteract the effect of long range dependencies in sentences, we constrain the context to the grammatical dependents and calculate a second contextual score $C_G$ using cosine distance. Table 5 provides an example of a full sentence $O$ and the grammatical context for the word *gives*.

**Grammaticality**: To assess whether a substitution results in a grammatically correct sentence, we calculate bigram frequencies of the candidate substitute and one word to the left and to the right of it using the 520 million word COCA corpus (Davies, 2014). If either of these frequencies equals 0, it is assumed that the substitute is not a valid grammatical fit or is extremely rare, making it a poor candidate for simplification. Table 6 shows one such example.

| | *left* | *right* |
|---|---|---|
| He was **capable** of [...] | | |
| *bigram* | was capable | capable of |
| *freq* | 778 | 12341 |
| He was **able** of [...] | | |
| *bigram* | was able | able of |
| *freq* | 10281 | 0 |

Table 6: Bigram frequencies for left and right contexts of a candidate substitute

Here, we use the bigram frequencies as a proxy for grammaticality: although "able" and "capable" are semantically similar, it is the grammatical constraints, captured by the bigram frequencies, that rule one of the alternatives out.

#### 4.3.1 Threshold-based filtering

Threshold-based filtering is performed by removing all substitutes that are unlikely to be grammatical or do not fit the target context. First, substitutes are removed from consideration if their right or left bigram frequency equals 0. Then, we remove substitutes if either of their contextual ELMo scores is below a given threshold $t$ ($C_S \vee C_G < t$), as this implies the substitute is not equivalent in this context. We test our filtering approach on the CEFR-LS dataset as it contains annotations for contextually suitable (value 1) as well as unsuitable (value 0) substitutions. We empirically find the optimal threshold value on the CEFR-LS dataset to be 0.175. With this optimal value, we can identify contextually suitable and grammatically correct substitutes on CEFR-LS with the *precision* of 0.7968, *recall* of 0.8081 and $F_1$=0.8014. As we show in Section 5, this filtering technique generalises well to other datasets.

#### 4.3.2 Ranking Algorithm

Once filtering has been performed we then rank substitutes. In order to rank substitutes, the simplicity and contextual semantic equivalence scores are combined to produce an overall suitability score. We evaluate our ranking techniques on the BENCHLS and CEFR-LS datasets.

We perform ranking using the sum of the contextual simplicity score ($S$) and the average contextual semantic equivalence score $\bar{C} = avg(C_S, C_G)$, and evaluate the results using the *TRank-at-n* measure introduced in Paetzold and Specia (2016a), which estimates the proportion of times a candidate with a gold-standard rank

| BENCHLS | | n=1 | n=2 | n=3 | MRR |
|---|---|---|---|---|---|
| $full_{(929)}$ | $S$ | 0.4974 | 0.7381 | 0.8899 | 0.6648 |
| | $\bar{C}$ | 0.3509 | 0.5885 | 0.7877 | 0.5998 |
| | $S$+$\bar{C}$ | **0.5602** | **0.8064** | **0.9428** | **0.7219** |
| $test_{(464)}$ | $S$ | 0.5839 | 0.7546 | 0.9302 | 0.7083 |
| | $\bar{C}$ | 0.4086 | 0.7142 | 0.8950 | 0.6563 |
| | $S$+$\bar{C}$ | **0.6774** | **0.7857** | **0.9308** | **0.8218** |
| | P&S | 0.4841 | 0.5596 | 0.7004 | 0.6615 |

Table 7: Ranking results on BENCHLS dataset

| CEFR-LS | n=1 | n=2 | n=3 | MRR |
|---|---|---|---|---|
| $S$ | 0.2166 | 0.3357 | 0.5271 | 0.4182 |
| $\bar{C}$ | 0.3754 | **0.6137** | **0.7689** | 0.5857 |
| $S$+$\bar{C}$ | **0.4624** | 0.5906 | 0.7314 | **0.6582** |
| P&S | 0.2129 | 0.4339 | 0.5760 | 0.4736 |

Table 8: Ranking results on CEFR-LS dataset

$r \leq n$ is ranked first by the system. For instance, our best performing ranking technique on the full BENCHLS dataset for $n$=1 is based on the combination of $S$+$\bar{C}$ scores and achieves *TRank-at-n*=0.5602. This means that for approximately 56% of the test instances the top ranked substitute in the gold standard is correctly ranked first. In addition, we report the *mean reciprocal rank (MRR)* (Voorhees, 1999), which takes into account the rank of the substitutes proposed by each ranking technique.

We report the results on the full BENCHLS dataset in the upper half of Table 7. In the lower half, we compare our results on the test set of 464 instances to those running the Paetzold and Specia (2016a) system (P&S) on the same test splits. Since the P&S system was trained on half of BENCHLS we cannot run it on the full dataset. Table 7 shows that ranking with $S$+$\bar{C}$ works best according to all measures and across both sets.

Table 8 reports the ranking results on the CEFR-LS data. We observe a decrease in performance, however this is expected: as all substitutes within the BENCHLS dataset are valid, ranking by simplicity is more informative; in contrast, the CEFR-LS dataset contains irrelevant substitutes, so contextual fit has more pronounced effects.

### 4.4 Recursive Step

Following CWI, substitute generation, filtering and substitute ranking steps, the system is able to perform a simplification. As outlined in Section 4.1.1, our system attempts to simplify one word at a time, starting with the word considered most complex. Since word complexity depends on the context, each individual lexical simplification made to a sentence has a subsequent impact on the perceived complexity of the surrounding words. Such sequential effects cannot be modelled by systems that apply several simplification steps at once. For instance, when considering examples (6) and (7) we see that the word *situation* is given a high likelihood of being complex in the context of another complex word *hazardous*. However, once *dangerous* is substituted for *hazardous*, the subsequent complexity of *situation* is reduced as well.

(6) It was a **hazardous**$_{0.90}$ **situation**$_{0.87}$

(7) It was a dangerous$_{0.30}$ situation$_{0.35}$

Our lexical simplification algorithm is applied to a sentence recursively: it starts by identifying and simplifying the most complex words in the sentence. Once the simplification is applied in step $n$, the algorithm reassesses word complexity in step $n$+1, which, in light of the simplifications applied in previous steps, might have changed. This prevents unnecessarily simplifying *situation* in example (7), which is no longer necessary after simplifying *hazardous* → *dangerous*. The simplification algorithm stops when there are no words with the complexity score above the predefined threshold left within the sentence, set to 0.5.

Table 9 exemplifies the benefit of using the recursive simplification approach REC-LS. We see that both the 'Simplify CW', aimed at individually identified complex words, and 'Simplify all' approaches result in unnecessary simplifications, whilst REC-LS stops when it recognises that the surrounding words are no longer complex.

| | |
|---|---|
| Original | *prepare*$_{0.32}$ *for a* *hazardous*$_{0.81}$ *journey*$_{0.52}$ |
| REC-LS | prepare for a **dangerous**$_{0.35}$ journey$_{0.45}$ |
| Simplify CW | prepare for a **dangerous trip** |
| Simplify all | **arrange** for a **dangerous trip** |

Table 9: Recursive simplification compared with simplifying only complex words and simplifying all words

We note that the only consequence of performing simplification recursively is that fewer words, or potentially different words, are simplified, and highlight that it does not lead to any error propagation. Table 10 presents the entire REC-LS algorithm including the recursive simplification step.

## 5 End-to-end System Performance

Finally, we test the full LS pipeline, visualised in Figure 1, on the BENCHLS and CEFR-LS

**Result:** Lexically simplified sentence
$S \leftarrow$ Input Sentence
$c \leftarrow$ Complexity threshold
REC-LS($S, c, ignore\_list$)
**while** *complexity(S)* $> c$ **do**
  complex_words $\leftarrow$ CWI($S$) - *ignore_list*
  SORT(complex_words)
  **for** *word in complex_words* **do**
    subs $\leftarrow$ SUB_GENERATION($word$);
    subs $\leftarrow$ MORPH($substitutes$);
    subs $\leftarrow$ SUB_RANK($substitutes$);
    top_substitute $\leftarrow$ $head(substitutes)$;
    **if** *top_substitute == word or []* **then**
      *ignore_list*.add(*word*)
      REC-LS($S, c, ignore\_list$)
    **else**
      REPLACE($S, word, top\_substitute$)
      REC-LS($S, c, ignore\_list$)
  **end**
**end**

Table 10: REC-LS Algorithm Steps

| | |
|---|---|
| (4) | Check if complexity of any word in $S$ is above the threshold $c$ |
| (5) | Check sentence for complex words minus those which are in *ignore_list* |
| (6) | Sort complex words according to confidence score |
| (8) | Generate substitutes for the top complex word |
| (9) | Convert substitutions to correct word form (e.g., tense for verbs) |
| (10) | Rank substitutes according to grammaticality, contextual semantic equivalence and simplicity |
| (11) | Take the top substitute |
| (13) | If no appropriate substitutes found, or best fit is original word, then retain original word and add word to *ignore_list* |
| (15) | Otherwise replace word and call function with new sentence |

datasets. The CWI step of our recursive LS algorithm identifies 77% of the target complex words in the BENCHLS and 86% of the target complex words in the CEFR-LS dataset. Next, the substitution generation and ranking produce lists of simplification candidates. We evaluate these steps using *precision* and measuring the percentage of times that our system ranks one of the gold standard candidates as its top choice for substitution.

The best precision $P$=0.7945 on BENCHLS is achieved with threshold-based filtering of the unsuitable candidates and ranking of the candidates according to the combination of contextual simplicity ($S$) and contextual fit ($\bar{C}$) scores. We note that this result outperforms the previous SOTA system by Horn et al. (2014), which is reported to have $P$=0.5460 (Paetzold and Specia, 2016a), by a large margin.

With a similar approach, we achieve $P$=0.4628 on the CEFR-LS, and to the best of our knowledge, this is the first time an LS system is benchmarked on this dataset. We note that the precision on this dataset is lower, which can be attributed to the smaller set of gold standard substitutes per word (an average of 2.35 in CEFR-LS vs 7.37 in BENCHLS). We also note that in a number of cases our system generates valid substitutes, which are not included in the gold standard. For example, *wealthy* → *rich* in "could participate in government just as **wealthy** men could". We present more examples of such cases in the Appendix.

We also run our entire recursive simplification system REC-LS on the three simplification datasets: WikiSmall, WikiLarge and Newsela, used in Zhang and Lapata (2017). We then compare the results to the SOTA simplification systems DRESS-LS by Zhang and Lapata (2017), which contains a specialised LS model, and the P&S simplification system (Paetzold and Specia, 2016a). As DRESS-LS is trained using the above datasets, we test using the 649 sentences that are reserved for testing only. Simplifications are assessed using the gold standard for lexical substitutions. For each dataset, the number of lexical simplifications performed by the systems is recorded. We then compare the simplifications performed by the system with the gold standard simplifications and calculate the *recall*, *precision* and *correct* proportion as follows:

- For *recall* we estimate the number of simplifications present in the gold standard $|G|$, and the total number produced by the system $S$, which are in the gold standard $|S \cap G|$. Recall is then calculated as $|S \cap G| / |G|$.
- For *precision*, we identify the proportion of simplifications out of the total $|S|$, which are also in the gold standard: $|S \cap G| / |S|$.

| | *Newsela* | | | *WikiSmall* | | | *WikiLarge* | | |
|---|---|---|---|---|---|---|---|---|---|
| | DRESS-LS | P&S | REC-LS | DRESS-LS | P&S | REC-LS | DRESS-LS | P&S | REC-LS |
| Precision | 0.1673 | 0.0836 | **0.3000** | 0.0000 | 0.0602 | **0.1275** | 0.2258 | 0.0697 | **0.2778** |
| Recall | 0.2389 | **0.9721** | 0.2393 | 0.0000 | **0.9743** | 0.4815 | 0.0004 | **0.9763** | 0.1856 |
| Correct | 0.3256 | 0.0402 | **0.5238** | 0.0000 | 0.0526 | **0.4615** | **0.4286** | 0.0787 | 0.4001 |

Table 11: Lexical substitution results for DRESS-LS, P&S and our system (REC-LS) on three genres

- Finally, *correct* stands for the proportion of instances where the top lexical substitution returned by the system is exactly the same as the gold standard one.

The results show that the recall for REC-LS is higher than that of DRESS-LS across all datasets, while the P&S system performs the best in terms of recall as it applies a 'simplify all' approach. The difference is especially pronounced in the WikiSmall dataset, where the DRESS-LS system does not perform any required lexical substitutions. REC-LS outperforms both DRESS-LS and P&S systems across all datasets in terms of precision, which indicates that the CWI stage of the algorithm is able to narrow down simplifications to relevant words. Finally, the three systems show different results in terms of correct proportion: REC-LS outperforms other systems on Newsela and WikiSmall, and DRESS-LS has a higher correct proportion on WikiLarge.

Finally, we note that there are many instances where the REC-LS system performs substitutions with valid alternatives that are not contained within the gold standard. For instance, consider the word "*separated*" in the following context:

(8) The island chain forms part of the Hebrides, *separated* from the Scottish mainland.

The gold standard substitution suggested by the human annotators in this context is "demarcated", whereas REC-LS substitutes this word with "split". We argue that, in this context, "split" would be both grammatically correct and semantically appropriate, while also being simpler than the original word and, arguably, the substitute suggested by the human annotators. However, as it is not an exact match with the gold standard substitution in the dataset, cases like this negatively impact precision of our system. We provide more examples of the system's output and compare the simplifications from the three systems in the Appendix.

## 6 Conclusions

In this paper, we have presented a novel recursive context-aware lexical simplification architecture REC-LS.[5] We summarise the main contributions of this work as follows:

1. our model takes the surrounding context into account at the complex word identification step, using a novel sequence labelling approach;
2. it filters and ranks the substitution alternatives taking contextual fit and grammaticality of the output into account;
3. the novel architecture performs simplification recursively and can be adapted to different levels of language complexity in the future;
4. as a result, REC-LS outperforms the current state-of-the-art systems in lexical simplification on various datasets.

We note that REC-LS can be used as a stand-alone lexical simplification system as well as an LS component as part of an integral TS system.

In this paper, we focus on intrinsic evaluation of our system, and we show that it outperforms SOTA in terms of the widely accepted set of measures, which we believe to be important for benchmarking purposes. In the future, we plan to extend this work with a syntactic simplification component and test the architecture extrinsically with non-native readers.

## Acknowledgments

We thank Cambridge English for supporting this research via the ALTA Institute. We are grateful to the anonymous reviewers for their valuable feedback. We also gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan V GPU used for this research.

[5]The code and output is available on github.com/siangooding/lexical_simplification

## References


Or Biran, Samuel Brody, and Noemie Elhadad. 2011. Putting it Simply: a Context-Aware Approach to Lexical Simplification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 496–501.

Ted Briscoe, John Carroll, and Rebecca Watson. 2006. The second release of the RASP system. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 77–80. Association for Computational Linguistics.

Yvonne Canning, John Tait, Jackie Archibald, and Ros Crawley. 2000. Cohesive generation of syntactically simplified newspaper text. In *Proceedings of the Third International Workshop on Text, Speech and Dialogue (TDS '00)*, pages 145–150.

John Carroll, Gido Minnen, Yvonne Canning, Siobhan Devlin, and John Tait. 1998. Practical simplification of English newspaper text to assist aphasic readers. In *Proceedings of AAAI Workshop on Integrating AI and Assistive Technology*, pages 7–10.

Raman Chandrasekar and Srinivas Bangalore. 1997. Automatic induction of rules for text simplification. *Knowledge Based Systems*, 10:183–190.

William Coster and David Kauchak. 2011a. Learning to Simplify Sentences Using Wikipedia. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 1–9.

William Coster and David Kauchak. 2011b. Simple English Wikipedia: A New Text Simplification Task. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 665–669.

Mark Davies. 2014. N-grams data from the Corpus of Contemporary American English (COCA).

Jan De Belder and Marie-Francine Moens. 2012. A dataset for the evaluation of lexical simplification. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 426–437. Springer.

Siobhan Devlin. 1998. The use of a psycholinguistic database in the simplification of text for aphasic readers. *Linguistic databases*.

Council of Europe Council of Europe. 2011. Common European Framework of Reference for Languages: Learning, Teaching, Assessment.

Christiane Fellbaum. 2012. WordNet. *The Encyclopedia of Applied Linguistics*.

Goran Glavaš and Sanja Štajner. 2015. Simplifying lexical simplification: do we need simplified corpora? In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 63–68.

Sian Gooding and Ekaterina Kochmar. 2018. CAMB at CWI Shared Task 2018: Complex Word Identification with Ensemble-Based Voting. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 184–194.

Sian Gooding and Ekaterina Kochmar. 2019. Complex Word Identification as a Sequence Labelling Task. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 1148–1153.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Colby Horn, Cathryn Manduca, and David Kauchak. 2014. Learning a Lexical Simplifier Using Wikipedia. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Short Papers)*, pages 458–463.

Carl James. 1998. *Errors in Language Learning and Use: Exploring Error Analysis*. London: Longman.

David Kauchak. 2013. Improving Text Simplification Language Modeling Using Unsimplified Text Data. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1537–1546.

Gondy Leroy, James E. Endicott, Obay Mouradi, David Kauchak, and Melissa L. Just. 2012. Improving perceived and actual text difficulty for health information consumers using semi-automated methods. In *Proceedings of the American Medical Informatics Association (AMIA) Fall Symposium*, pages 522–531.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Franz Josef Och and Hermann Ney. 2000. Improved Statistical Alignment Models. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics (ACL '00)*, pages 440–447.

Gustavo Paetzold and Lucia Specia. 2016a. Benchmarking lexical simplification systems. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 3074–3080, Paris, France. European Language Resources Association (ELRA).

Gustavo Paetzold and Lucia Specia. 2016b. Unsupervised lexical simplification for non-native speakers. In *Proceedings of The 30th AAAI*, pages 3761–3767.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT 2018*, pages 2227–2237.

Andrew Porter, Jennifer McMaken, Jun Hwang, and Rui Yang. 2011. Common Core Standards: The New U.S. Intended Curriculum. *Educational Researcher*, 40:103–116.

Marek Rei. 2017. Semi-supervised multitask learning for sequence labeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2121–2130.

Marek Rei, Gamal K.O. Crichton, and Sampo Pyysalo. 2016. Attending to characters in neural sequence labeling models. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 309–318.

Matthew Shardlow. 2013. A Comparison of Techniques to Automatically Identify Complex Words. In *Proceedings of the Student Research Workshop at the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 103–109.

Advaith Siddharthan. 2006. Syntactic Simplification and Text Cohesion. *Research on Language and Computation*, 4:77–109.

Advaith Siddharthan. 2014. A survey of research on text simplification. *Special issue of International Journal of Applied Linguistics*, 165:259–298.

Satoru Uchida, Shohei Takada, and Yuki Arase. 2018. CEFR-based Lexical Simplification Dataset. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*, pages 3254–3258.

Ellen M. Voorhees. 1999. TREC-8 Question Answering Track Report. In *Proceedings of the 8th Text Retrieval Conference*, pages 77–82.

Wei Xu, Chris Callison-Burch, and Courtney Napoles. 2015. Problems in Current Text Simplification Research: New Data Can Help. *Transactions of the Association for Computational Linguistics*, 3:283–297.

Mark Yatskar, Bo Pang, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2010. For the sake of simplicity: Unsupervised extraction of lexical simplifications from Wikipedia. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 365–368.

Seid Muhie Yimam, Chris Biemann, Shervin Malmasi, Gustavo Paetzold, Lucia Specia, Sanja Štajner, Anaïs Tack, and Marcos Zampieri. 2018. A Report on the Complex Word Identification Shared Task 2018. In *Proceedings of the 13th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 66–78.

Seid Muhie Yimam, Sanja Štajner, Martin Riedl, and Chris Biemann. 2017. CWIG3G2 - Complex Word Identification Task across Three Text Genres and Two User Groups. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 401–407.

Xingxing Zhang and Mirella Lapata. 2017. Sentence Simplification with Deep Reinforcement Learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 584–594.

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. 2010. A Monolingual Tree-based Translation Model for Sentence Simplification. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1353–1361.

# Leveraging Medical Literature for Section Prediction in Electronic Health Records

**Sara Rosenthal**
IBM Research
Yorktown Heights, NY, USA
sjrosenthal@us.ibm.com

**Ken Barker**
IBM Research
Yorktown Heights, NY, USA
kjbarker@us.ibm.com

**Jason Zhicheng Liang***
Rensselaer Polytechnic Institute
Troy, New York, USA
liangz4@rpi.edu

## Abstract

Electronic Health Records (EHRs) contain both structured content and unstructured (text) content about a patient's medical history. In the unstructured text parts, there are common sections such as *Assessment and Plan*, *Social History*, and *Medications*. These sections help physicians find information easily and can be used by an information retrieval system to return specific information sought by a user. However, it is common that the exact format of sections in a particular EHR does not adhere to known patterns. Therefore, being able to predict sections and headers in EHRs automatically is beneficial to physicians. Prior approaches in EHR section prediction have only used text data from EHRs and have required significant manual annotation. We propose using sections from medical literature (e.g., textbooks, journals, web content) that contain content similar to that found in EHR sections. Our approach uses data from a different kind of source where labels are provided without the need of a time-consuming annotation effort. We use this data to train two models: an RNN and a BERT-based model. We apply the learned models along with source data via transfer learning to predict sections in EHRs. Our results show that medical literature can provide helpful supervision signal for this classification task.

## 1 Introduction

Electronic Health Records (EHRs) are an important tool used by physicians that contain an abundance of information about each patient. In addition to aiding the physician in providing patient care, EHRs can be used for downstream tasks such as predicting patient outcome, representation learning, and information extraction (Shickel et al., 2017). All of these tasks can be applied to the unstructured text parts of an EHR. Furthermore, they would benefit from the section structure to pinpoint the likely location where the information should be found (e.g. drug codes are more likely to be in the *Medication* section). However, when physicians edit the unstructured text parts of an EHR, there are no set rules for indicating sections and format is not strictly enforced, nor is there any markup (e.g. XML). An added difficulty is that the formatting is not uniform across EHRs. This is even more common when the EHRs come from different hospitals. For example, one EHR may have the section heading "Assessment and Plan" where another may use "A&P". As an added challenge there may also be nested sections, such as information about a specific medication. Being able to predict sections and headers in EHRs automatically is beneficial to physicians. It allows them to find information easily as well as discover errors and omissions in an EHR.

Consider the publicly available EHR note[1] in Figure 1. The note is divided into the 10 sections found in that EHR (*Problems*, *Medications*, *History*, etc.). This example provides insight into why section prediction can be a difficult task. Although most of the headers appear to be bold, there is also plenty of bold text which is not the main header (e.g. see the *History* section). Additionally, in some cases (e.g. *Allergies* section) there is no text under the header at all. This makes it difficult to segment the data appropriately. Finally, although

*Work completed during internship at IBM Research

[1] http://www.ahrq.gov/professionals/prevention-chronic-care/improve/system/pfhandbook/mod8appbmonicalatte.html



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4864–4873,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

**Problems**
DIABETES MELLITUS (ICD-250.)
HYPERTENSION, BENIGN ESSENTIAL (ICD-401.1)

**Medications**
PRINIVIL TABS 20 MG (LISINOPRIL) 1 po qd
Last Refill: #30 x 2 : Carl Savem MD (08/27/2010)
HUMULIN INJ 70/30 (INSULIN REG & ISOPHANE (HUMAN)) 20 units ac breakfast
Last Refill: #600 u x 0 : Carl Savem MD (08/27/2010)

**Directives**

**Allergies and Adverse Reactions (! = critical)**

OFFICE VISIT

History of Present Illness
Reason for visit: Routine follow up
Chief Complaint: No complaints

**History**
**Diabetes Management**
**Hyperglycemic Symptoms**
**Polyuria:** no
. . .

**Review of Systems**
**General:** denies fatigue, malaise, fever, weight loss
**Eyes:** denies blurring, diplopia, irritation, discharge
**Ear/Nose/Throat:** denies ear pain or discharge, nasal obstruction or discharge, sore throat
**Cardiovascular:** denies chest pain, palpitations, paroxysmal nocturnal dyspnea, orthopnea, edema Respiratory: denies coughing, wheezing, dyspnea, hemoptysis
**Gastrointestinal:** denies abdominal pain, dysphagia, nausea, vomiting, diarrhea
. . .

**Vital Signs**
Ht: **64 in.** Wt: **140 lbs.**T: **98.0** degF. T site: **oral** P: **72** R: **16** BP: **158/90**

Physical Exam
General Appearance: well developed, well nourished, no acute distress
Eyes: conjunctiva and lids normal, PERRLA, EOMI, fundi WNL
Ears, Nose, Mouth, Throat: TM clear, nares clear, oral exam WNL
Respiratory: clear to auscultation and percussion, respiratory effort normal
Cardiovascular: regular rate and rhythm, S1-S2, no murmur, rub or gallop, no bruits, peripheral pulses normal and symmetric, no cyanosis, clubbing, edema or varicosities
Skin: clear, good turgor, color WNL, no rashes, lesions, or ulcerations
. . .

Plan
Medications:
HUMULIN INJ 70/30 20 u ac breakfast
PRINIVIL TABS 20 MG 1 qd
Treatment: Will have annual foot exam at next visit.
. . .

Figure 1: Shortened version of a public EHR[1]. Sections that we classify are highlighted in different colors.

**Review of systems**

a psychiatric review of systems may include screening questions directed at identifying or exploring co morbid psychiatric illnesses or issues e g sigecaps mnemonic or phq 9 for depression generalized anxiety disorder 7 for anxiety digfast mnemonic for mania or specific questioning around psychoses or other psychiatric complaints a full review of systems should attempt to identify and list all of the relevant stressors that may be impacting a patient s function and overall health

**Diagnosis**

ultrasound remains as one of the only effective ways of prenatally diagnosing larsen syndrome prenatal diagnosis is extremely important as it can help families prepare for the arrival of an infant with several defects ultrasound can capture prenatal images of multiple joint dislocations abnormal positioning of legs and knees depressed nasal bridge prominent forehead and club feet these symptoms are all associated with larsen syndrome so they can be used to confirm that a fetus has the disorder

**Medical History**

kniest dysplasia is an autosomal dominant condition this means that the person only needs to have one copy of the mutated gene in order to have the condition people with a family history are at a higher risk of having the disease than people with no family history a random mutation in the gene can cause a person with no family history to also have the condition

**Physical Examination**

the general appearance of patients may vary according to the experienced symptoms the patient may be comfortable or restless and in severe distress with an increased respiratory rate a cool and pale skin is common and points to vasoconstriction some patients have low grade fever 38 39°c blood pressure may be elevated or decreased and the pulse can become irregular if heart failure ensues elevated jugular venous pressure and hepatojugular reflux or swelling of the legs due to peripheral edema may be found on inspection rarely a cardiac bulge with a pace different from the pulse rhythm can be felt on precordial examination various abnormalities can be found on auscultation such as a 3rd and 4th heart sound systolic murmurs paradoxical splitting of the second heart sound a pericardial friction rub and rales over the lung

Figure 2: Sections from Wikipedia articles (*Psychiatric history*, *Larsen syndrome*, *Kniest dysplasia*, and *Myocardial infarction diagnosis*) that are similar to sections that occur in EHRs. Colors match sections types from Figure 1. The four headers map to the classes *ROS, Find, PMH, and Exam.*

medications have their own section, they also appear in the *Plan* section. Other issues that are not exposed in this example include: 1) Section order is not consistent across EHRs, 2) Headers may be missing, 3) Common features of headers (e.g. bold or colon) are not guaranteed to appear.

Prior approaches in EHR section prediction have been specific to one source and have required significant annotation effort. We propose to reduce the annotation burden by augmenting training data using sections in medical literature (journals, textbooks, web content). Section headers in medical literature are often much more consistent than in EHRs, allowing us to identify with high precision a large number of training example sections using a small number of simple patterns. And although the style and content of sections is different from EHRs, our hypothesis is that there is enough similarity in language to help train models to detect sections in EHRs. Figure 2 shows paragraphs from Wikipedia articles whose sections are similar to those found in EHRs. The text tends to be more descriptive of the kinds of things that could be found in an EHR, but there is semantic overlap.

Our approach is not specific to one hospital or software system and uses data where labels are easy to infer without the need of a time-consuming annotation effort. Furthermore, the large amount of data allows us to explore state of the art neural classification approaches. Finally, training from medical literature can aid in identification of common sections in EHRs from different sources, making it possible to share EHR information more easily across medical facilities and insurance companies which may have access to EHRs from multiple providers.

Our work has two main contributions, demonstrating that augmenting automatically labeled

data from medical literature with:

1. a *small amount* of labeled in-domain EHR training data significantly improves prediction in the EHR dataset.

2. labeled EHR data from a *different source* (out-of-domain) significantly improves the transferability of models trained when there is *no labeled* data in the in-domain EHR dataset.

## 2 Related Work

Prior work in EHR section prediction has focused on the following tasks:

1. *Section detection:* detecting the boundaries of sections; detecting section header text.

2. *Section classification:* assigning a class label to a section or to sentences (a section class label on a sentence indicates the section to which a sentence likely belongs).

Research may concentrate on section detection only (Ganesan and Subotin, 2014; Dai et al., 2015), section classification (with section boundaries assumed to be known) (Li et al., 2010; Haug et al., 2014) or both (Apostolova et al., 2009; Denny et al., 2009; Tepper et al., 2012). In this paper we focus on section-level classification and section classification at the sentence level.

Prior approaches to section prediction include Support Vector Machines leveraging features computed by bi-gram tf-idf vector representations (Apostolova et al., 2009), Hidden Markov Models (HMM) with sections regarded as part of a sequence (Li et al., 2010), Maximum Entropy Classifiers (Tepper et al., 2012), $\ell$1-Regularized Logistic Regression (Ganesan and Subotin, 2014), Bayesian models using N-gram features (Haug et al., 2014), and linear-chain Conditional Random Fields (CRF) to determine section headers (Dai et al., 2015). Most of these approaches rely heavily on hand-crafted features that are time consuming to develop and may not easily generalize across EHRs from different sources. To the best of our knowledge no prior work has explored deep learning or transfer learning. Given enough data, deep neural networks can extract useful features automatically. Some prior work is able to capture regularities in section ordering, either by using the Viterbi Algorithm (Ganesan and Subotin, 2014; Li et al., 2010) or beam search (Tepper et al., 2012) during the final section labeling phase, or by incorporating section ordering features (e.g. the class of the section preceding the section to be classified) into the model (Tepper et al., 2012). In this paper, since we mainly focus on transfer learning based on deep learning models, we predict labels for each section or sentence independently, leaving section sequence considerations to future work.

Two previous efforts have explored more than one dataset (Tepper et al., 2012; Ganesan and Subotin, 2014). Both evaluated domain adaptability and found significant reduction in performance across domains. The largest dataset, *i2b2*, has 13,962 expert-labeled sections (Dai et al., 2015). We use this dataset in our experiments as well. Because of the difficulty of annotating training data, some prior work adopts a semi-automated labeling approach (Apostolova et al., 2009; Denny et al., 2009; Li et al., 2010; Ganesan and Subotin, 2014) as we do for our medical literature dataset.

With the exception of the approaches using i2b2, all prior work uses proprietary datasets that are not available publicly making it difficult to compare to earlier approaches. We make our MedLit dataset as well as our extension of the labels for the i2b2 dataset available for research.

## 3 Data

We chose a set of 11 section class labels based on those used in the prior work discussed in Section 2 and on the most common sections from EHRs in our sources. The 11 classes are:

*Allergies* (Alrg)
*Assessment and Plan* (A&P)
*Chief Complaint* (CC)
*Examination* (Exam)
*Family History* (FHX)
*Diagnostic Findings* (Find)
*Medications* (Meds)
*Past Medical History* (PMH)
*Personal and Social History* (PSH)
*Procedures* (Proc)
*Review of Systems* (ROS)

For our experiments we use three datasets: Medical Literature (MedLit), Cleveland Clinic EHRs (ClvC), and i2b2 EHRs (i2b2).

| Source | # Sections | # Sentences |
|---|---|---|
| Textbooks | 807 | 7,407 |
| Guidelines | 4,781 | 53,013 |
| Wikipedia | 2,658 | 24,830 |

Table 1: Sections and sentences in each MedLit source.

### 3.1 Medical Literature (MedLit)

The medical literature dataset consists of passages from textbooks, guidelines, and a subset of medically relevant Wikipedia articles[2]. The number of sentences per source type is shown in Table 1. In total there are four sources in this dataset: Wikipedia and licensed content from DynaMed, Elsevier, and Wiley publishers.

To generate training data, we created a short list of common phrases for each of the 11 section classes by looking at labels used in prior work as well as section headers labeled in our annotated EHR datasets. We extracted sections in the MedLit corpus whose headers (indicated by XML markup) matched these phrases. For example, any section whose header matched "Chief Complaint", "Reason for Admission", "CC", "Signs and Symptoms", "History of Present Illness", or "SUBJ" was extracted as a positive example of the *Chief Complaint* class[3]. We experimented with partial phrase matches but ultimately decided to use exact match with stemming for higher precision on the common classes. For less frequent classes (e.g. *Allergies, Family History, Personal and Social History*), we allowed additional words to be present in the title. We make the 2,658 sections and labels for the Medical Wikipedia dataset available publicly for research purposes. Examples of sections and their headers are shown in Figure 2, with corresponding class labels indicated in the caption. In the discussion section (Section 5.3) we analyze the quality of the MedLit dataset as a cheaply labeled resource.

### 3.2 I2b2 EHRs (i2b2)

We use the i2b2 Risk Factors dataset (Stubbs and Uzuner, 2015; Stubbs et al., 2015) which was annotated for section header boundaries (Dai et al., 2015). The annotations do not indicate section

[2]Articles under the 'Clinical Medicine' category (https://en.wikipedia.org/wiki/Category:Clinical_medicine).

[3]It is common for *History of Present Illness* to be its own section but it is often part of or interchangeable with *Chief Complaint* so we opted to combine these two sections.

| | Source | # Sections | # Sentences | Ratio |
|---|---|---|---|---|
| Train | MedLit | 7042 | 72191 | 10 |
| | i2b2 | 1680 | 10093 | 6 |
| | ClvC | 294 | 3467 | 12 |
| Dev | MedLit | 1204 | 13059 | 11 |
| | i2b2 | 1591 | 9373 | 6 |
| | ClvC | 404 | 3282 | 8 |
| Test | i2b2 | 3098 | 19110 | 6 |
| | ClvC | 404 | 4046 | 10 |

Table 2: The number of sections and sentences as well as the ratio of sentences to sections in each of the three datasets for train, dev, and test sets.

class so we matched the headers to our 11 classes. We used partial matches for this data due to the variability of EHR header text. There were 743 unique headers in the Dai et al. (2015) dataset that map to our 11 headers.

### 3.3 Cleveland Clinic EHRs (ClvC)

The Cleveland Clinic dataset consists of 178 deidentified patient notes from 54 patients acquired through a research collaboration agreement with Cleveland Clinic. The notes were annotated by two medical students in prior work. Inter-annotator agreement was computed on the first 34 notes (containing 106 sections) annotated by both annotators. The $\kappa$ score was 0.86 for the sections and 0.80 at the note level. Due to the high agreement between the annotators, the remaining notes were annotated by one annotator each. We manually mapped the section class labels from those annotations to our 11.

### 3.4 Data Splits

The MedLit dataset was split 80/20 by section for training and tuning. We do not test on MedLit as our goal is finding sections in EHRs. The ClvC EHR dataset was split by patient into 60% for training and tuning, and 40% for testing. For i2b2 (Stubbs and Uzuner, 2015; Stubbs et al., 2015) we use Set 1 and 2 for training and development, and the i2b2 Test Set for testing. The distribution of the train, development/tuning (dev), and test is shown in Table 2. MedLit is the largest dataset. It has more than twice as many sentences as i2b2 and 8 times as many as ClvC. For some of the classes (e.g. *Allergies* and *Personal and Social History*) MedLit data is harder to find. *Procedures* is the smallest category in i2b2 (102 sentences) and *Allergies* is the smallest category in ClvC with

only 35 sentences in the training data.

## 4 Method and Results

Our work addresses the scenario of an EHR dataset (a *target* dataset) with little or no training data for section classification. We would like to measure how well models trained on a different EHR dataset and/or on medical literature (*source* datasets) can be transferred to classify the target dataset. We use each of the ClvC and i2b2 datasets in turn as target, with the other as source. We also experiment with using the MedLit dataset as source, both alone and together with data from a source EHR dataset. Specifically, for each target EHR dataset we compare the following models:

- ***ClvC:*** train on labeled data from ClvC EHRs
- ***i2b2:*** train on labeled data from i2b2 EHRs
- ***MedLit:*** train on medical literature data only
- ***MedLit + TR ClvC:*** take the MedLit model, then continue to train (transfer) on labeled ClvC data
- ***MedLit TR i2b2:*** take the MedLit model, then continue to train (transfer) on labeled i2b2 data

In all cases we evaluate our model on our two test sets: ClvC and i2b2. The data used in the test set is considered the **target**, or **in-domain** dataset. The other dataset(s) are considered **source**, or **out-of-domain**, for that experiment and will differ depending on the experiment. We consider the model that trains and tests on the same dataset to be an upper bound (UB). Given enough data[4], we would expect this configuration to perform the best.

We show our results with two different approaches. The first is a Recurrent Neural Network (RNN) using Gated Recurrent Unit (GRU) cell (Cho et al., 2014) and attention mechanism (Bahdanau et al., 2015). GRU aims to solve the vanishing gradient problem revealed in standard RNNs. In our model, attention is used to generate a weighted sum of GRU cell outputs for each word in the input text, for predicting the classification label (rather than only using the output from the last cell). The motivation is to let the model focus on those words that are the most useful for prediction, especially for long input text. The weights are computed by the soft alignment scores between each of the outputs and the last output of the RNN.

The second approach is based on BERT (Devlin et al., 2019), a state-of-the-art language representation model that pre-trains bidirectional representations by jointly conditioning on both left and right context. Once pre-trained, a BERT model can be fine-tuned to specific tasks. In our setting, we take the output of the transformer for the first token in the input, i.e., the special `[CLS]` word embedding, as the representation of the input, which is then used for label prediction by feeding into a classification layer.

We also experimented with a Convolutional Neural Network text classification model (Kim, 2014) as well as traditional machine learning models (Naive Bayes and SVM) using n-gram features, but all performed worse than our GRU RNN and BERT.

For the GRU RNN, we use the Adam optimizer, a batch size of 32, dropout of 0.2, and embedding size 300. We experimented with other parameter values on the development set, but these worked best. We ran each model for 50 epochs—enough for the training loss to converge. For our BERT experiments we use a PyTorch implementation[5] with the bert-base-uncased model. We use the default BERT parameters including the BERT Adam optimizer, a batch size of 32, dropout of 0.1, and embedding size 768. All text is cut off to the first 128 word-pieces. We experimented with different numbers of epochs, and chose the model that performed best on the dev set (usually one tuned at 10 epochs or fewer). Statistical significance was computed using McNemar's test. We experimented with both section classification and sentence classification as described in the following subsections.

Our transfer setting follows the pre-training and fine-tuning approach: after training on the large source domain (MedLit), we continue to tune the model on a small amount of labeled data from the target domain. We follow this approach using our RNN as well as the BERT model where we first tune BERT to the MedLit data and then continue to tune that model on the EHR data. We also experimented with using subsets of the MedLit dataset, downsampling, and class balancing. These exper-

[4] In the case of ClvC we do not have enough annotated data to train a model that would be considered a good upper bound

[5] https://github.com/huggingface/pytorch-pretrained-BERT

| Experiment | RNN F1 | BERT F1 |
|---|---|---|
| MedLit | 0.47 | 0.55 |
| ClvC (UB) | $0.59^{+}$ | 0.89 |
| MedLit + TR ClvC | **0.78**$^{*}$ | **0.90** |
| i2b2 | 0.71 | 0.83 |
| MedLit + TR i2b2 | 0.65 | 0.78 |

(a) Testing on ClvC

| Experiment | RNN F1 | BERT F1 |
|---|---|---|
| MedLit | 0.68 | 0.76 |
| i2b2 (UB) | **0.95** | **0.99** |
| MedLit + TR i2b2 | 0.93 | **0.99** |
| ClvC | $0.53^{+}$ | $0.84^{+}$ |
| MedLit + TR ClvC | 0.81$^{*}$ | 0.92$^{*}$ |

(b) Testing on i2b2

Table 3: Section-Level results for testing on the (a) ClvC and (b) i2b2 EHRs. F1-scores are micro averages across all classes. The best results in each column are highlighted in bold. The results are grouped by MedLit, in-domain ((a) ClvC, (b) i2b2), and out-of-domain ((a) i2b2, (b) ClvC) training. * is significantly higher than + at $p <= .01$.

iments resulted in no significant change or a negative change to the results[6], so we report results using the complete, raw MedLit dataset. We expect that the models learn which data sources are more useful.

### 4.1 Section Classification

Our first set of experiments follow the approach of prior work (Li et al., 2010; Haug et al., 2014) in classifying sections of the EHR. The section header is included in the data if it exists. As noted in prior work, this task is easier than sentence classification as it is common for the header to be discoverable with accuracy $>$ 90% for most categories. We find similar trends in the i2b2 data with the upper bound (train on i2b2, test on i2b2) achieving an average 95% F-score with the RNN and near perfect F-score of 99% with BERT.

Results of experiments using the Cleveland Clinic (ClvC) test set as target are shown in Table 3a. The ClvC upper bound using the RNN (59% average F-score) is much lower than is typical in prior work, underlining the difficulty of even section-level classification with small amounts of training data (294 sections). Using the RNN model trained on MedLit alone does quite poorly, but tuning on MedLit prior to training on ClvC improves results very significantly over ClvC training alone (average F-score of 78%). BERT improves the results significantly here with 89% F-score, but the 90% F-score when tuning on the MedLit data is not significantly better. The bottom rows of the table show the scenario where no in-domain (ClvC) data exists, so we train on out-of-domain (i2b2) data. In this case, training on i2b2 alone is better than pre-training first on MedLit. When there is enough EHR data (even from a different source), the MedLit data does not help.

Table 3b shows results using the i2b2 test set as target. With the large amount of in-domain i2b2 training data available, using MedLit does not help. The bottom rows of Table 3b shows performance if we did not have i2b2 training data. Using the out-of-domain (ClvC) training data performs poorly. Transferring MedLit with a very small amount of out-of-domain (ClvC) data significantly outperforms using just the ClvC data for both the RNN and BERT, improving average F-score from 53% to 81% for the RNN and from 84% to 92% for BERT.

### 4.2 Sentence Classification

Prior work on section-level classification assumes that the section segmentation and header are known. In practice, this is not always the case. It is common for headers to be missing or unclear. In our experiments, therefore, we don't assume to know where the header is and instead of trying to classify a section we classify all sentences in the EHR. For training data for this task we take the class label annotated on each section in our datasets and attach it to each sentence in the section. So each sentence in the section (including the header) is considered an instance of the class. The sentence predictions could be combined to provide section level boundaries. We discuss this further as future work in Section 6. In the MedLit training data, we exclude sentences that are too small (fewer than 15 characters) or too large (more than 400 characters). As with our section-level experiments (Section 4.1), we evaluate our performance with the ClvC and i2b2 test sets. We first tuned the MedLit model using the MedLit development set as described in Section 4. We found that with BERT, only one epoch was necessary, with more epochs resulting in over-fitting.

[6]There is some change in the performance when using Wikipedia only, but not in the overall trends

| Experiment | RNN F1 | BERT F1 |
|---|---|---|
| MedLit | 0.30 | 0.37 |
| ClvC (UB) | $0.61^{+}$ | $0.73^{+}$ |
| MedLit + TR ClvC | **0.62**$^{*}$ | **0.74**$^{*}$ |
| i2b2 | 0.56 | 0.62 |
| MedLit + TR i2b2 | 0.53 | 0.60 |

(a) Testing on ClvC

| Experiment | RNN F1 | BERT F1 |
|---|---|---|
| MedLit | 0.30 | 0.41 |
| i2b2 (UB) | $0.64^{+}$ | **0.71** |
| MedLit + TR i2b2 | **0.65**$^{*}$ | **0.71** |
| ClvC | $0.48^{+}$ | $0.59^{+}$ |
| MedLit + TR ClvC | 0.52$^{*}$ | 0.60$^{*}$ |

(b) Testing on i2b2

Table 4: Sentence-Level results for testing on the (a) ClvC and (b) i2b2 EHRs in Micro Avg F1-score. The best results in each column are highlighted in bold. The results are grouped by MedLit, in-domain ((a) ClvC, (b) i2b2) and out-of-domain ((a) i2b2, (b) ClvC) training. * is significantly higher than + at $p <= .01$.

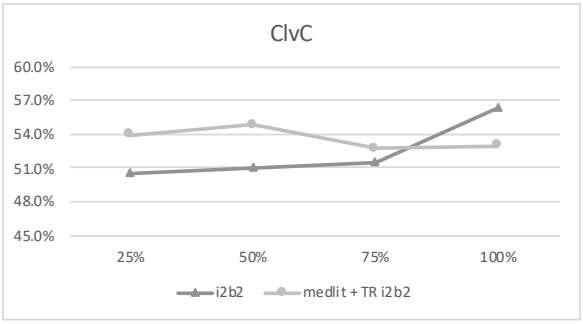


Figure 3: F-score trends of our RNN model training on smaller portions of the i2b2 data for the i2b2 and MedLit + TR i2b2 experiments and testing on ClvC

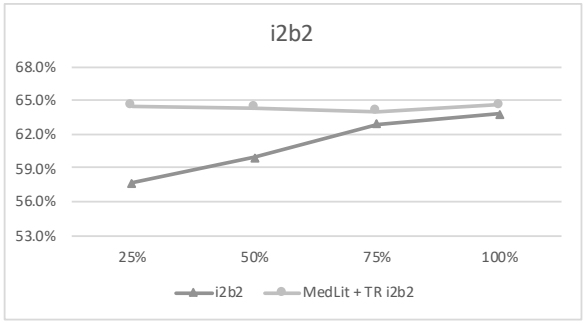


Figure 4: F-score trends of our RNN model training on smaller portions of the i2b2 data for the i2b2 and MedLit + TR i2b2 experiments and testing on i2b2

The results for testing on the ClvC data are shown in Table 4a. Although we call the configuration where we train and test on ClvC data the "upper bound", the MedLit + TR ClvC performs significantly ($p <= .01$) better for the RNN (62% vs 61%) and BERT models (74% vs 73%). We believe that this is because the ClvC dataset is small so the upper bound is not achieved. On the other hand, because the i2b2 training dataset is large, training on the MedLit data first does not provide a significant improvement with either model. We analyzed this result further by experimenting on training with subsets (25%, 50%, and 75%) of the i2b2 training data and testing on ClvC for the i2b2 and MedLit + TR i2b2 experiments. Figure 3 shows that with less out-of-domain EHR training data, transfer learning does help significantly (p-value $<= .01$) in all cases for the RNN model. In fact, there is no significant difference between using 25% or 100% of the i2b2 training data in the MedLit + TR i2b2 experiment.

The sentence-level results for testing with i2b2 as target are shown in Table 4b. Using the MedLit data in the RNN provides a significant improvement (65% vs 64%). However, there is no difference in the BERT models, likely because of the a large amount of i2b2 training data. Here again, we explored how having a smaller subset of the in-domain training data would impact the results. We found that running the MedLit + TR i2b2 experiment with just 25% of the i2b2 dataset with the RNN model performs as well (no significant difference) as training on the full i2b2 dataset (Figure 4), and 12% better than the model using 25% of the i2b2 data alone. In contrast, when using the BERT model, MedLit + TR i2b2 only performs better than i2b2 alone if less than 25% of the training data is available. This indicates that BERT can compensate for the lack of data while the RNN cannot, and MedLit is needed to improve the model. In the scenario where no in-domain training data is available we show that MedLit + TR ClvC does significantly better than just ClvC when testing on i2b2 for the RNN model (52% vs 48%) and BERT (60% vs 59%). We discuss these results further in the next section

| Experiment | F1@1 | F1@2 | F1@3 |
|---|---|---|---|
| RNN ClvC | 0.48 | 0.66 | 0.75 |
| RNN MedLit + TR ClvC | 0.52 | 0.69 | 0.79 |
| BERT ClvC | 0.59 | 0.75 | 0.82 |
| BERT MedLit + TR ClvC | 0.60 | 0.75 | 0.83 |

Table 5: Average F1 score for training on ClvC and testing on i2b2 when examining the top 3 predictions.

## 5 Discussion

### 5.1 How much do out-of-domain EHRs help?

One of our most interesting results using the full datasets is the improvement we are able to achieve training with a small amount of ClvC (out-of-domain) EHR data and testing on i2b2 as the target domain (the last two rows of Table 3b and Table 4b). All of these results on the sentence level are quite low due to similar text that may be found in multiple sections. For example, the text found for *Chief Complaint* and *Past Medical History* can be quite similar with the only difference being *when* the problems occurred. To determine if this kind of misclassification was common we looked at our performance based on whether the correct class was in the top 1, 2, or 3 predictions (F1@3). Both ClvC and MedLit+TR ClvC showed large improvements for the RNN and BERT models as shown in Table 5 with roughly a 27 point improvement for the RNN models and a 22 point improvement for the BERT models. We analyzed the confusion matrix and the most common misclassifications were to the majority MedLit and ClvC classes (*Assessment and Plan* and *Chief Complaint*).

### 5.2 Do models impact sections differently?

We also compared performance on the individual classes when transferring from MedLit. To do this, we explored the training on i2b2 and testing on BERT i2b2, and BERT MedLit + TR i2b2 experiments (See the middle rows in Table 4b) where both models performed the same. We found the BERT i2b2 performed better for *Allergies* (6.8%), but BERT MedLit + TR i2b2 performed better for *Family History* (5.8%), and *Procedures* (7.9%). This suggests that it may be worth exploring an ensemble approach.

### 5.3 What is the quality of the MedLit data?

How well our approach performs on EHR section classification relies on the MedLit data. The MedLit dataset is cheaply labeled data of a different genre than the target data, so we would not expect the MedLit model to perform particularly well on its own on EHR data. To judge the quality of the MedLit-trained model in *its own genre*, we analyzed its performance on the dev set made up solely of medical literature. The best average F-score was 65.2% using the RNN and 71.2% for BERT on the sentence classification task. In particular, we found that *Assessment and Plan* performs quite poorly. We examined the confusion matrix and the predictions are distributed among the other classes indicating that it is more vague. Looking to see if the correct label is at the second- or third-highest scoring predictions, we find the F1@3 is 88.9% for the RNN model and 92.6% for the BERT model. This is significantly better, but still shows that it is weakly labeled data. We also analyzed the performance from our different medical literature sources. This analysis caused us to drop one of our initial guideline sources because it had few useful labels. The two relevant labels it did have were easy to predict correctly causing a bias toward the source. Finally, we inspected the text for the section headers. There were 66 unique headers in the medical literature dev set ranging over the 11 classes. "Physical", which maps to the *Examination* class was the most common section header and was predicted correctly 74% and 79% of the time for the RNN and BERT models respectively. Some other good strings include "History of Present Illness" and "signs and symptoms" for the *Chief Complaint* class and "medication" for the *Medications* class. Our *Review of Systems* headers included common subsections found in this area of the EHR such as "eye", "neurology", and "genitourinary". Some of these did not perform as well, probably because they are likely to appear in many different sections. We also analyzed the confusion matrix to determine the common misclassifications. In most cases the common misclassifications were to *Chief Complaint*, the majority class. We also found that *Past Medical History* was often incorrectly labeled as *Medications*. We found this analysis to be consistent across both models indicating that the data is not model dependent.

## 6 Conclusion

In this paper we describe a novel approach to classifying sections in Electronic Health Records

when a limited amount of in-domain training data is available. We present a new dataset for EHR section prediction from Medical Literature, of which the Wikipedia part is available to the public for research purposes. We show that combining a very small amount of in-domain EHR data with a large amount of automatically labeled, out-of-domain, out-of-genre Medical Literature data can perform as well as using a large amount of in-domain EHR data at the section and sentence level. We also show that combining out-of-domain, out-of-genre Medical Literature data with out-of-domain EHRs can provide significant improvement over using just out-of-domain EHRs at the section and sentence level, depending on training data size. These results indicate that even though the data in Medical Literature is very different in style, the content can bridge between the domain-specific vocabularies of different EHR systems. We show that our approach can be used to achieve good results on *new unseen EHR datasets* with minimal or even no training data. In the future we would also like to explore using both i2b2 and ClvC together to see if a multi-task learning approach would provide additional improvements. In addition to BERT, we also briefly explored BioBERT (Lee et al., 2019), a BERT model pre-trained on a medical corpus. In our initial experiments BioBERT performed worse than BERT, but we would like to explore this further.

Finally, in this work we focus on individual sentence and section classification and show that we can achieve improvements in this regard. In addition, we could also exploit the structure of the document to provide additional improvements. The structure often follows a pattern in EHRs (for example, *Chief Complaint* tends to be the first section). Prior work has looked at CRFs and HMMs (Li et al., 2010; Dai et al., 2015) to exploit this property. We would like to explore whether we can improve our model by combining it with a model that takes into account trends at the document level. Using LSTM-CRFs (Lample et al., 2016; Huang et al., 2015) as a second level with BERT as pre-training on our model may provide such an improvement.

## 7 Acknowledgments

We thank the physicians and IT staff at Cleveland Clinic who provided de-identified EMRs under an IRB protocol and the students from New York Medical College and SUNY Downstate Medical Center for their enthusiastic and dedicated annotation work.

## References

Emilia Apostolova, David S Channin, Dina Demner-Fushman, Jacob Furst, Steven Lytinen, and Daniela Raicu. 2009. Automatic segmentation of clinical texts. In *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, pages 5905–5908. IEEE.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder – decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Hongjie Dai, Syed Abdul Shabbir, Chih-Wei Chen, and Chieh-Chen Wu. 2015. Recognition and evaluation of clinical section headings in clinical documents using token-based formulation with conditional random fields. *BioMed Research International*, 2015.

Joshua C. Denny, Anderson Spickard, III, Kevin B. Johnson, Neeraja B. Peterson, Josh F. Peterson, and Randolph A. Miller. 2009. Evaluation of a method to identify and categorize section headers in clinical documents. *Journal of the American Medical Informatics Association*, 16(6):806–815.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

K. Ganesan and M. Subotin. 2014. A general supervised approach to segmentation of clinical texts. In *2014 IEEE International Conference on Big Data (Big Data)*, pages 33–40.

Peter J. Haug, Xinzi Wu, Jeffrey P. Ferraro, Guergana Savova, Stanley M. Huff, and Christopher G. Chute. 2014. Developing a section labeler for clinical documents. In *AMIA 2014, American Medical Informatics Association Annual Symposium, Washington, DC, USA, November 15-19, 2014*.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270. Association for Computational Linguistics.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *CoRR*, abs/1901.08746.

Ying Li, Sharon Lipsky Gorman, and Noémie Elhadad. 2010. Section classification in clinical notes using supervised hidden markov model. In *Proceedings of the 1st ACM International Health Informatics Symposium*, IHI '10, pages 744–750, New York, NY, USA. ACM.

Benjamin Shickel, Patrick Tighe, Azra Bihorac, and Parisa Rashidi. 2017. Deep EHR: A survey of recent advances on deep learning techniques for electronic health record (EHR) analysis. *CoRR*, abs/1706.03446.

Amber Stubbs, Christopher Kotfila, Hua Xu, and Özlem Uzuner. 2015. Identifying risk factors for heart disease over time: Overview of 2014 i2b2/uthealth shared task track 2. *Journal of Biomedical Informatics*, 58:S67 – S77. Proceedings of the 2014 i2b2/UTHealth Shared-Tasks and Workshop on Challenges in Natural Language Processing for Clinical Data.

Amber Stubbs and Özlem Uzuner. 2015. Annotating risk factors for heart disease in clinical narratives for diabetic patients. *Journal of Biomedical Informatics*, 58:S78 – S91. Proceedings of the 2014 i2b2/UTHealth Shared-Tasks and Workshop on Challenges in Natural Language Processing for Clinical Data.

Michael Tepper, Daniel Capurro, Fei Xia, Lucy Vanderwende, and Meliha Yetisgen-Yildiz. 2012. Statistical section segmentation in free-text clinical records. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*. European Language Resources Association (ELRA).

# Neural News Recommendation with Heterogeneous User Behavior

**Chuhan Wu[1], Fangzhao Wu[2], Mingxiao An[3], Tao Qi[1], Jianqiang Huang[4], Yongfeng Huang[1], and Xing Xie[2]**
[1]Department of Electronic Engineering, Tsinghua University, Beijing 100084, China
[2]Microsoft Research Asia, Beijing 100080, China
[3]University of Science and Technology of China, Hefei 230026, China
[4]Peking University, Beijing 100871, China
{wu-ch19, qit16, yfhuang}@mails.tsinghua.edu.cn,
{fangzwu, xing.xie}@microsoft.com,
anmx@mail.ustc.edu.cn, 1701210864@pku.edu.cn

## Abstract

News recommendation is important for online news platforms to help users find interested news and alleviate information overload. Existing news recommendation methods usually rely on the news click history to model user interest. However, these methods may suffer from the data sparsity problem, since the news click behaviors of many users in online news platforms are usually very limited. Fortunately, some other kinds of user behaviors such as webpage browsing and search queries can also provide useful clues of users' news reading interest. In this paper, we propose a neural news recommendation approach which can exploit heterogeneous user behaviors. Our approach contains two major modules, i.e., news representation and user representation. In the news representation module, we learn representations of news from their titles via CNN networks, and apply attention networks to select important words. In the user representation module, we propose an attentive multi-view learning framework to learn unified representations of users from their heterogeneous behaviors such as search queries, clicked news and browsed webpages. In addition, we use word- and record-level attentions to select informative words and behavior records. Experiments on a real-world dataset validate the effectiveness of our approach.

## 1 Introduction

Online news platforms such as Google News[1] and MSN[2] News have gained huge popularity for online digital news reading (Das et al., 2007). Tens of thousands of news articles are streamed from various sources every day, making it very difficult for users to read all news to find their interested content (Phelan et al., 2011). Thus, personalized news recommendation is critical for online news platforms to target user interests and alleviate information overload (IJntema et al., 2010).




Figure 1: An illustrative example of news recommendation based on heterogeneous user behaviors.

News and user representation learning is critical for news recommendation. Many deep learning based methods have been proposed for this task (Okura et al., 2017; Wang et al., 2018). For example, Okura et al. (2017) proposed to learn news representations from news bodies via auto-encoders, and learn user representations from their clicked news via a gated recurrent unit (GRU) network. Wang et al. (2018) proposed to learn news representations from news titles via a knowledge-aware convolutional neural network (CNN), and learn user representations from their clicked news using the similarities between candidate news and clicked news. These methods rely on the news click histories to model users' news reading interest. However, these methods usually suffer from the data sparsity problem, since the news click behaviors of many users on news platforms are very limited, making it difficult for these methods to learn accurate representations of these users.

Luckily, there are several other kinds of user

[1]https://news.google.com/
[2]https://www.msn.com/en-us/news



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4874–4883,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

behaviors such as search queries and webpage browsing which are useful for news recommendation. Many online users frequently use search engines such as Google and Bing to search for desired information (Wu et al.). In addition, they may browse related webpages to get more detailed information. Thus, the search and browsing data accumulated by commercial search engines can cover a large number of online users. In addition, the search and browsing behaviors of users can provide rich information for inferring user interests. For example, in Fig. 1, since the user posts the search query "toyota crown", and browses the webpage of "Pros and Cons of Toyota" for detailed information, she may have high interest in Toyota cars and is likely to read news articles related to Toyota cars. However, this user may click only a few news articles, and her interest in toyota cars cannot be mined from the news click history. Thus, incorporating the heterogeneous behaviors of online users such as search queries and webpage browsing has the potential to improve the performance of news recommendation.

Our work is motivated by following observations. First, the characteristics of search queries, webpage titles and news have huge differences. For example, in Fig. 1, search queries are usually phrase pieces with a few words, while news and webpage titles are usually complete sentences. Thus, they should be handled differently. Second, different words usually have different importance in representing behavior records. For example, in Fig. 1 the word "NFL" is more informative than "Today". Third, different behavior records may also have different importance for representing users. For instance, the search query "safe cars" in Fig. 1 is more informative than the query "google map" in modeling user interest. Fourth, different kinds of user behaviors usually have different informativeness for user representation learning. For example, in Fig. 1 since the user clicks very few news articles, her news click behaviors are not very informative for modeling this user.

In this paper, we propose a neural news recommendation approach with heterogeneous user behavior (NRHUB). The core of our approach is a news representation module and a user representation module. In the news representation module, we learn news representations from news titles via a CNN network with word-level attentions to select important words. In the user representation module, we propose an attentive multi-view learning framework to incorporate the different kinds of user behavior data into our model to learn unified user representations. In each view, we propose to use hierarchical user encoders to first learn record representations from words, and then learn user representations from different kinds of behavior records. Since different views may have different informativeness for modeling users, we propose to use a view-level attention network to select important views for learning informative user representations. In addition, since different words and records may also have different informativeness, we apply word-level and record-level attention networks to select important words and records. Extensive experiments are conducted on a real-world news recommendation dataset. The results show that our approach can effectively improve the performance of news recommendation, especially in the cold-start scenario.

## 2 Related Work

News recommendation is an important task in both natural language processing and data mining fields, and has wide applications (Okura et al., 2017; Zheng et al., 2018). Learning accurate news and user representations is critical for news recommendation. Many of existing methods for news recommendation rely on manual feature engineering to build news and user representations (Liu et al., 2010; Son et al., 2013; Bansal et al., 2015). For example, Liu et al. (2010) proposed to use topic categories and interest features generated by a Bayesian model to build news and user representations. Son et al. (2013) proposed an Explicit Localized Semantic Analysis (ELSA) model for location-based news recommendation. They proposed to extract topic and location features from Wikipedia pages to build news representations. Lian et al. (2018) proposed a deep fusion model (DMF) to learn news representations from various handcrafted features such as title length, entities and news topics, and learn user representations from features extracted from the news reading, web browsing, and searching behaviors of users. However, these methods usually rely on manual feature engineering, which necessities massive domain knowledge and time to design. In addition, these methods cannot exploit contextual information in the behavior records of users, which may be insufficient to learn accurate user representations.

In recent years, several deep learning based methods are proposed for news recommendation (Okura et al., 2017; Wang et al., 2018; Khattar et al., 2018; Kumar et al., 2017; Zheng et al., 2018; Wu et al., 2019b; An et al., 2019; Wu et al., 2019c,a; Zhu et al., 2019). For example, Okura et al. (2017) proposed to learn news representations from news bodies via denoising autoencoders, and learn user representations from the news browsing sequence via a GRU network. Wang et al. (2018) proposed to learn news representations from news titles via a knowledge-aware CNN network, which can incorporate useful entity information from knowledge graphs. Wu et al. (2019b) proposed to learn news representations from news titles, and apply personalized attention mechanism at both word- and news-level to generate representations of news and users according to user preferences. However, these methods only learn user representations from a single kind of user behavior, i.e., news click, which may be insufficient. Different from these methods, our approach can learn unified user representations by incorporating heterogeneous user behaviors via an attentive multi-view learning framework. Extensive experiments on real-world dataset validate our approach can learn better news and user representations, and achieve better performance on news recommendation than existing methods.

## 3 Our Approach

In this section, we introduce our neural news recommendation approach with heterogeneous user behavior (NRHUB) in detail. The architecture of our *NRHUB* approach is shown in Fig. 2. Our approach has three major modules, i.e., *news representation*, *user representation* and *click predictor*. Next, we introduce the details of each module.

### 3.1 News Representation Learning

The *news representation* module is used to learn representations of news articles from their titles. It contains three layers.

The first one is a word embedding layer, which is used to convert a news title from a sequence of words into a sequence of low-dimensional semantic vectors. Denote a news title with $M$ words as $[w_1, w_2, ..., w_M]$. It is converted into a vector sequence $[\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_M]$ via a pre-trained embedding matrix.

The second one is a CNN layer (Kim, 2014). Usually, local contexts are very important in representing the news titles. For example, in the news title "Toyota Crown Classics for Sale", local contexts of "Crown" such as "Toyota" are useful for inferring the topic of this news, i.e., cars. Thus, we apply a CNN network to learn contextual representations of words within news titles by capturing their contexts. The CNN layer computes the contextual representations of each word, and outputs a vector sequence $[\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_M]$.

The third one is a word-level attention network. Different words in the same news title may have different importance in representing this news. For example, in the second news in Fig. 1, the word "NFL" is more informative than "Today" for representing this news. Thus, we propose to use attention mechanism to select important words in news titles for learning informative news representations. Denote the attention weight of the $i$-th word in a news title as $\alpha_i^w$:

$$
\begin{aligned}
a_i^w &= \mathbf{v}_w^n \times \tanh(\mathbf{V}_w \times \mathbf{c}_i + \mathbf{v}_w), \\
\alpha_i^w &= \frac{\exp(a_i^w)}{\sum_{j=1}^{M} \exp(a_j^w)},
\end{aligned} \tag{1}
$$

where $\mathbf{V}_w$ and $\mathbf{v}_w$ are projection parameters, $\mathbf{v}_w^n$ is the query vector in this attention network. The final contextual representation of a news title is the summation of its word representations weighted by their attention weights as follows:

$$
\mathbf{r}^n = \sum_{i=1}^{M} \alpha_i^w \mathbf{c}_i. \tag{2}
$$

### 3.2 User Representation Learning

The *user representation* module is used to learn the representations of users from different kinds of user behaviors, e.g., web searching, news click and webpage browsing. Since these user behaviors usually have different characteristics, simply aggregate them together into a long document may not be optimal for news representation. Thus, we propose an attentive multi-view learning framework to learn unified user representations by incorporating these three kinds of user behavior information as different views of news.

The first view is *news encoder*, which is used to learn user representations from his/her clicked news. There are two major submodules in this view. The first one is news representation, as described in section 3.1. Denote the sequence of news clicked by a user as $[D_1, D_2, ..., D_N]$, where

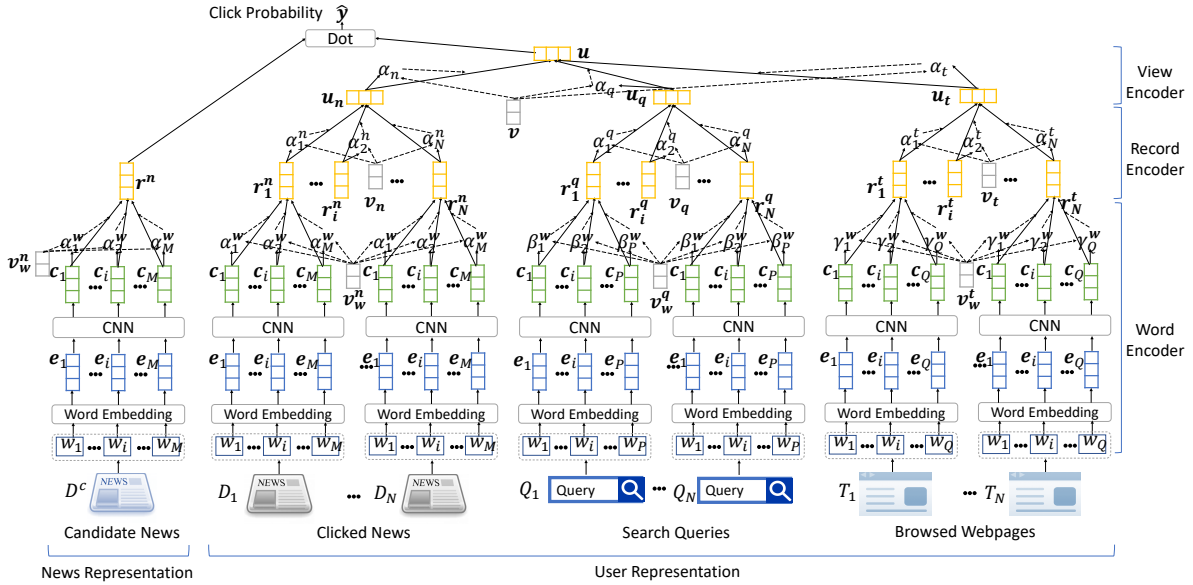


Figure 2: The framework of our *NRHUB* approach.

$N$ is the number of clicked news articles. It is transformed into a sequence of hidden news representations, denoted as $[\mathbf{r}_1^n, \mathbf{r}_2^n, ..., \mathbf{r}_N^n]$.

The second one is a news-level attention network. Different news clicked by the same user may have different informativeness for representing this user. For example, in Fig. 1, the second news is more informative than the first news in modeling user preferences, since the first one is usually not audience sensitive. Thus, we use a news-level attention network to select important news for learning informative user representations. Denote the attention weight of the $i$-th news clicked by a user as $\alpha_i^n$, which is computed as:

$$a_i^n = \mathbf{v}_n \times \tanh(\mathbf{V}_n \times \mathbf{r}_i + \mathbf{b}_n),$$
$$\alpha_i^n = \frac{\exp(a_i^n)}{\sum_{j=1}^{N} \exp(a_j^n)}, \tag{3}$$

where $\mathbf{V}_n$ and $\mathbf{b}_n$ are parameters, $\mathbf{v}_n$ is the attention query vector. The news based representation of a user $u$ is the summation of the representations of clicked news articles weighted by their attention weights as follows:

$$\mathbf{u}_n = \sum_{i=1}^{N} \alpha_i^n \mathbf{r}_i^n. \tag{4}$$

The second view is *query encoder*, which is used to learn representations of users from their search queries. There are two submodules in this view. The first module is query representation, which is used to learn query representations from words. It also has three layers.

The first one is a word embedding layer, which is shared with the news representation view. Denote a search query $q$ with $P$ words as $q = [w_1, w_2, ..., w_P]$. It is converted into a sequence of hidden word representations, denoted as $[\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_P]$.

The second one is a CNN layer. Local contexts are very important in learning query representations. For example, in the query "xbox one player", the local contexts of "one", i.e., "xbox" and "player" are important in representing this query. Thus, we apply CNN to capture the local contexts within queries, and compute the contextual representations of each word. The outputs of this CNN layer is denoted as $[\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_P]$.

The third one is a word-level attention network. Different words in the same search query usually have different informativeness for representation learning. For example, in the query "hot movies this week" "movies" is more informative than "week" in modeling this query. Thus, we propose to use a word-level attention network to recognize important words in search queries for learning informative query representations. The attention weight $\beta_i^w$ of the $i$-th word in a search query is computed as:

$$b_i^w = \mathbf{v}_w^q \times \tanh(\mathbf{V}_q \times \mathbf{c}_i + \mathbf{v}_q),$$
$$\beta_i^w = \frac{\exp(b_i^w)}{\sum_{j=1}^{P} \exp(b_j^w)}, \tag{5}$$

where $\mathbf{V}_q$, $\mathbf{v}_q$ and $\mathbf{v}_w^q$ are parameters in the attention network. The contextual representation of this query is the summation of its word representations weighted by their attention weights, as follows:

$$\mathbf{r}^q = \sum_{i=1}^{P} \beta_i^w \mathbf{c}_i. \tag{6}$$

The second module is a query-level attention network, which is used to learn user representations from query representations. Different queries posted by the same user usually have different importance in characterizing this user. For example, the query "toyota cars" is more informative than the query "google mail", since the later one does not contain information of user preference. To select important search queries for learning informative user representations, we propose to incorporate a query-level attention network. The attention weight $\alpha_i^q$ of the query $Q_i$ is computed as:

$$\begin{aligned} a_i^q &= \mathbf{v}_q \times \tanh(\mathbf{U}_q \times \mathbf{r}_i + \mathbf{u}_q), \\ \alpha_i^q &= \frac{\exp(a_i^q)}{\sum_{j=1}^{N} \exp(a_j^q)}, \end{aligned} \tag{7}$$

where $\mathbf{U}_q$, $\mathbf{u}_q$ and $\mathbf{v}_q$ are attention parameters. The user representation learned from queries is the summation of the query representations weighted by their attention weights as:

$$\mathbf{u}_q = \sum_{i=1}^{N} \alpha_i^q \mathbf{r}_i^q. \tag{8}$$

The third view is *webpage encoder*. It is used to learn representations of users from the titles of their browsed webpages. There are two submodules in this view. The first one is *webpage representation*, which is used to learn the representations of webpage titles from their words, and the second one is a webpage-level attention network which learns user representation from webpages. In the *webpage representation* module, we first use a CNN network to learn contextual representations of the words in webpage titles, and then we use a word-level attention network to highlight important words. The hidden representation of each webpage is the summation of the contextual representations of words weighted by their attention weights, which is similar to the search query and news views described previously. The webpage-level attention network is used to select informative webpages for user representation learning, and the final webpage based user representation $\mathbf{u}_t$ is the summation of the webpage representations weighted by their attention weights.

The last component in the *user representation* module is a view encoder. Usually, different views may have different informativeness for modeling users. For example, a user may never or rarely cl news, and his/her news click data may be uninformative for representing this user. Thus, modeling the informativeness of different views may benefit user representation learning. Motivated by these observations, we propose a view-level attention network to select important views for learning informative user representations. Denote the attention weights of the news, query and webpage views as $\alpha_n$, $\alpha_q$ and $\alpha_t$, respectively. For instance, the attention weight of the news title view is computed as follows:

$$\begin{aligned} a_n &= \mathbf{v} \times \tanh(\mathbf{U}_v \times \mathbf{u}_n + \mathbf{u}_v), \\ \alpha_n &= \frac{\exp(a_n)}{\exp(a_n) + \exp(a_q) + \exp(a_t)}, \end{aligned} \tag{9}$$

where $\mathbf{v}$, $\mathbf{U}_v$ and $\mathbf{u}_v$ are parameters. The attention weights of the search query and webpage title views are computed similarly. The unified user representation is the summation of the user representations from different views weighted by their attention weights, which is formulated as

$$\mathbf{u} = \alpha_n \mathbf{u}_n + \alpha_q \mathbf{u}_q + \alpha_t \mathbf{u}_t. \tag{10}$$

### 3.3 Click Predictor

The *click predictor* module is used to predict the probability of a user clicking a candidate news article from their representations. Denote the representation of the candidate news $D^c$ as $\mathbf{r}^n$. Following (Okura et al., 2017), the click probability score $\hat{y}$ is computed by the inner product of the user and candidate news representations, i.e., $\hat{y} = \mathbf{u}^T \mathbf{r}^n$.

### 3.4 Model Training

Motivated by (Huang et al., 2013), we propose to use the negative sampling technique for model training. For each news clicked by a user which is regarded as a positive sample, we randomly sample $K$ news in the same impression but not clicked by this user as the negative samples. Denote the click probability score of the positive news as $\hat{y}^+$, and scores of the $K$ negative news as $[\hat{y}_1^-, \hat{y}_2^-, ..., \hat{y}_K^-]$. We normalize these scores using the softmax function to compute the posterior

| | | | |
|---|---|---|---|
| **# users** | 10,000 | **avg. # words per new title** | 11.29 |
| **# news** | 42,255 | **avg. # displays per news** | 247.89 |
| **# queries** | 923,041 | **# user w/ queries** | 5,282 |
| **# webpage titles** | 813,852 | **# user w/ webpages** | 6,134 |
| **# impressions** | 360,428 | **avg. # words per query** | 3.25 |
| **# samples** | 10,474,493 | **avg. # words per webpage title** | 9.01 |
| **# positive samples** | 503,698 | **# negative samples** | 9,970,795 |

Table 1: Statistics of our dataset.

click probability of a positive sample as follows:

$$p_i = \frac{\exp(\hat{y}_i^+)}{\exp(\hat{y}_i^+) + \sum_{j=1}^{K} \exp(\hat{y}_{i,j}^-)}. \quad (11)$$

The loss function for model training is the negative log-likelihood of all positive samples:

$$\mathcal{L} = -\frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \log(p_i), \quad (12)$$

where $\mathcal{S}$ is the set of positive training samples.

## 4 Experiments

### 4.1 Datasets and Experimental Settings

We conducted experiments on a real-world news recommendation dataset[3] collected from MSN News[4] logs during Dec. 13, 2018 and Jan. 12, 2019. In addition, we crawled the search queries and titles of browsed webpages from the logs of the Bing search engine. The detailed statistics of this dataset are summarized in Table 1. The news data in the last week is used for test, and the rest is used for model training. In addition, we randomly sampled 10% of the training data for validation.

In our experiments, the word embeddings were 300-dimensional and we used the pre-trained Glove embedding for initialization (Pennington et al., 2014). All CNN networks had 400 filters, and their window size was set to 3. The dimension of attention query vectors was 200. Following (Wu et al., 2019b), the negative sampling ratio $K$ was set to 4. Adam (Kingma and Ba, 2014) was used as the optimization algorithm. We applied 30% dropout to the word embedding and CNN networks to mitigate overfitting. The training batch size was 64. These hyperparameters were all tuned on the validation set. We independently repeated each experiment 10 times and reported the average results in AUC, MRR, nDCG@5 and nDCG@10 of all impressions.

### 4.2 Performance Evaluation

We evaluate the performance of our *NRHUB* approach by comparing it with many baseline methods, including: (1) *LibFM* (Rendle, 2012), a matrix factorization method for recommendation; (2) *DSSM* (Huang et al., 2013), deep structured semantic model; (3) *Wide&Deep* (Cheng et al., 2016), a popular neural recommendation method with a wide linear model and a deep neural network; (4) *DeepFM* (Guo et al., 2017), a famous neural recommendation method with a factorization machine and a deep neural network; (5) *DFM* (Lian et al., 2018), a deep fusion model for news recommendation; (6) *GRU* (Okura et al., 2017), a neural news recommendation method based on autoencoder and GRU to learn news and user representations; (7) *DKN* (Wang et al., 2018), a deep knowledge-aware news recommendation method; (8) *Conv3D* (Kumar et al., 2017; Khattar et al., 2018), a nerual news recommendation method based on a 3-D convolutional neural network to learn user representations from the sequence of clicked news[5]; (9) *NRHUB*, our neural new recommendation approach with heterogeneous user behavior; (10) *NRHUB-basic*, a variant of our approach with only news click behavior; (11) *NRHUB-concat*, a variant of our approach which aggregates different kinds of user behaviors as a single view. Following (Wang et al., 2018), in methods (1, 3-5), we use one-hot encoded user ID, news ID and the TF-IDF features of news titles as the input features. The results are summarized in Table 2.

From Table 2, we have several observations. First, deep learning based news recommendation methods (e.g., *DSSM* and *NRHUB*) can outperform *LibFM*. This is because neural networks can

[3]Some publicly available resources can be found at https://github.com/wuch15/NRHUB.

[4]https://www.msn.com/en-us/news

[5]We only use news titles for fair comparison.

| Methods | AUC | MRR | nDCG@5 | nDCG@10 |
|---|---|---|---|---|
| LibFM | 0.5661 | 0.2414 | 0.2689 | 0.3552 |
| DSSM | 0.5949 | 0.2675 | 0.2881 | 0.3800 |
| Wide & Deep | 0.5812 | 0.2546 | 0.2765 | 0.3674 |
| DeepFM | 0.5830 | 0.2570 | 0.2802 | 0.3707 |
| DFM | 0.5861 | 0.2609 | 0.2844 | 0.3742 |
| GRU | 0.6114 | 0.2823 | 0.3006 | 0.3931 |
| DKN | 0.6070 | 0.2801 | 0.3012 | 0.3933 |
| Conv3D | 0.6051 | 0.2765 | 0.2987 | 0.3904 |
| NRHUB-basic | 0.6232 | 0.2927 | 0.3158 | 0.3986 |
| NRHUB-concat | 0.6265 | 0.2945 | 0.3177 | 0.4010 |
| NRHUB* | **0.6317** | **0.3020** | **0.3260** | **0.4076** |

Table 2: The results of different methods. *The improvement is significant at $p < 0.01$.

learn better news and user representations than traditional matrix factorization methods, which is beneficial for more accurate news recommendation. Second, among deep learning based methods, both *NRHUB-basic* and *NRHUB* can outperform all baseline methods. This is because our approaches incorporate both word- and news-level attention networks to simultaneously select important words and news for learning more informative news and user representations. Thus, our approaches can outperform baseline methods. Third, *NRHUB* and *NRHUB-concat* methods outperform *NRHUB-basic*. This is because search queries and browsed webpage can provide rich information of user interests, which is useful for learning user representation for news recommendation. Fourth, *NRHUB* consistently outperforms *NRHUB-concat*. This is because the characteristics of news titles, search queries and webpage titles are quite different, and they should be handled differently. Thus, simply merging them together is not the optimal way to exploit them. These results validate the effectiveness of our *NRHUB* approach in incorporating heterogeneous user behaviors for news recommendation.

### 4.3 Effectiveness of Attentive Multi-view Learning

In this section, we explore the effectiveness of the attentive multi-view learning framework in our approach. First, we want to verify the effectiveness of incorporating heterogeneous user behaviors as different views. We compare the performance of our *NRHUB* approach with its variants with different combinations of views, and the results are shown in Fig. 3. From Fig. 3, we have several observations. First, both search queries and browsed webpage titles are useful for improving the performance of news recommendation. This may be because users usually use post search queries to search engines to seek for interested content, and may browse several related webpages for detailed information. Thus, search queries and browsed webpage titles contain rich information of user preferences, which is useful for learning accurate user representations. Second, webpage titles are more important than search queries. This may be because webpage titles have larger coverage in our dataset. In addition, search queries are usually very short with a few words, while webpage titles are often like complete sentences with much more words. Thus, webpage titles are more informative for learning user representations. Third, incorporating both kinds of user behaviors can further improve the performance of our approach. These results validate the effectiveness of incorporating heterogeneous user behaviors via multi-view learning.

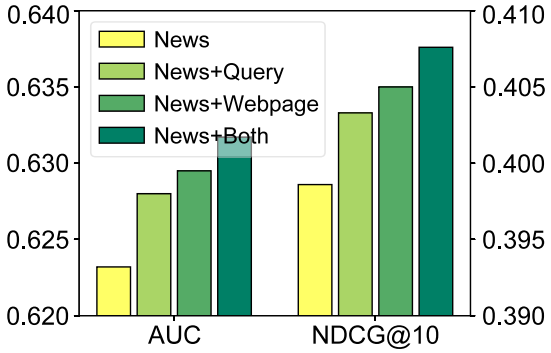


Figure 3: Effect of heterogeneous user behaviors.

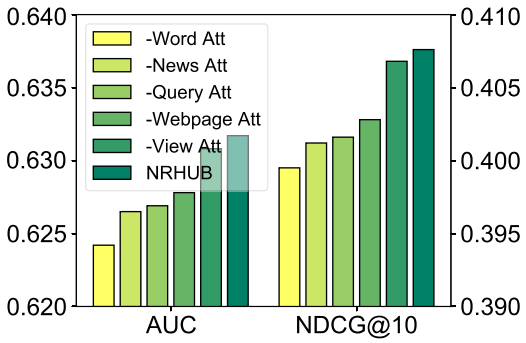


Figure 4: Influence of different attention networks.

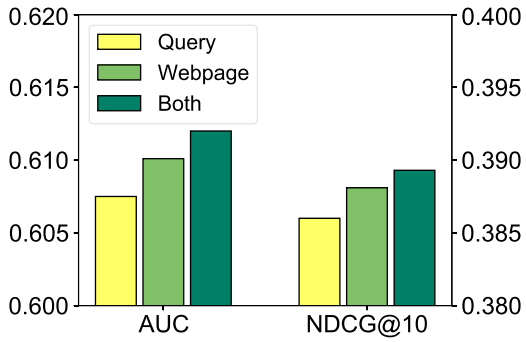


Figure 5: Cold-start performance of our approach.

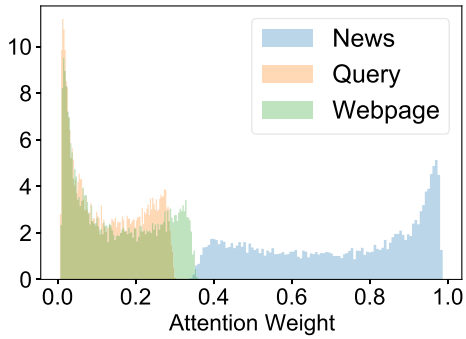


Figure 6: Distributions of view-level attention weights. The left, middle and right curves respectively denote search query, browsed webpage and clicked news.

Next, we explore the effectiveness of different attention networks in our approach. We conduct experiments using the leave-one-out scheme to evaluate the contribution of each attention network. The results are shown in Fig. 4. From Fig. 4, we find the word-level attention network can effectively improve the performance of our approach. This is because different words in the same news, query or webpage title may have different importance for learning user representations. Thus, selecting important words can help learn more informative news, query and webpage representations. In addition, the news-, query- and webpage-level attention networks are also important to our approach. This is because different news articles, search queries or webpages usually have different importance in representing users, and selecting important news, queries and webpages is beneficial for learning accurate user representations. Besides, the view-level attention network is also useful. This is because different views also have different informativeness in modeling user preferences. Thus, evaluating the informativeness of different views via a view-level attention network can learn better user representations. Moreover, combining all these kinds of attention networks can further improve the performance of our approach. These results validate the effectiveness of the attentive multi-view learning framework in our approach.

### 4.4 Cold-start Performance

In this section, we explore the cold-start performance of our approach. In practical use, massive users may rarely or have never clicked news articles on the target news platform, making it difficult to effectively recommend news to these users. Luckily, the search and browsing behaviors can cover many of these users. These user behaviors usually contain rich information of user preferences, and have the potential to address the cold-start problem. To validate the effectiveness of our approach, we compare the performance of several variants of our approach without the news encoder view to simulate the cold-start situation. The results are shown in Fig. 5. According to the results, we find our approach can still achieve satisfactory performance in the cold-start situation. This may be because search queries and webpage titles are important clues of user interests, which are very useful for news recommendation. Thus, our approach can recommend news effectively to the users with very limited or even no news browsing histories, which can effectively improve their experiences. In addition, combining the information of queries and webpage titles can further improve the performance of our approach. These results validate the effectiveness of our approach.

### 4.5 Case Study

In this section, we conduct several case studies to visually explore the effectiveness of different at-

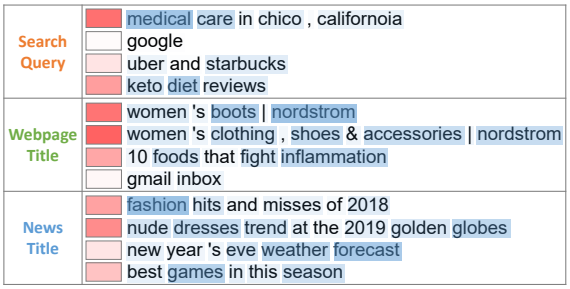


Figure 7: Visualization of the word-level and record-level attention weights from a randomly selected user.

tention networks in our approach. First, we visualize the distributions of the view-level attention weights in our *NRHUB* approach, and the results are shown in Fig. 6. According to the results, we find the attention weights of the news encoder view can be relatively low. This may be because the news browsing histories of some users are less informative for modeling these users. In addition, we find the attention weight of the webpage encoder view is higher than the query encoder view. This may be because webpage titles are usually complete sentences, while search queries are usually pieces of words. Thus, webpage titles contain richer information of user interests than search queries, which are more informative for representing users.

Next, we illustrate the attention weights in the word-level and record (news, query and webpage)-level attention networks. The results are shown in Fig. 7. From Fig. 7, we have several observations. First, we find search queries and webpage titles can provide much information of user interests which is not covered by the news browsing histories of users. For example, we can infer that this user is interested in the health topic from her queries and browsed webpages, which is not revealed by her clicked news. Thus, incorporating heterogeneous behaviors of users are useful for learning more accurate user representations. Second, we find that our approach can effectively select important words in queries, news and webpage titles. For example, the word "medical" gains high attention weight since it is informative for learning query representations, while the word "in" receives low attentions. In addition, our approach can effectively select important queries, news and webpages. For example, the query "deto diet reviews" and the webpage of "10 foods that fight inflammation" are assigned high attention weights since they are informative for learning user representations, but the query "google" and the webpage of "gmail inbox" are assigned low attention weights since they are uninformative. These results validate the effectiveness of the attention networks in our approach.

## 5 Conclusion and Furture Work

In this paper, we propose a neural news recommendation approach which can exploit heterogeneous user behaviors. In our approach, we propose an attentive multi-view learning framework to learn unified user representations from their search queries, clicked news, and browsed webpages by regarding them as different views. We propose to apply a view-level attention network to select important views for informative user representation learning. Besides, we propose to employ both word- and record-level attention networks to learn more informative news and user representations. Extensive experiments on a real-world dataset collected from MSN News show our approach can effectively improve the performance of neural news recommendation, and can achieve satisfactory results in cold-start scenarios.

In our future work, we will explore three potential directions. The first one is how to incorporate the interactions between different kinds of user behaviors, since the relatedness of them may be useful for modeling the interest evolution of users. The second one is how to incorporate the activities of users on social media platforms, since the opinions, experiences and events shared by users on social media are usually useful clues to indicate their interests. The third one is how to incorporate language models to generate context-aware word embeddings, which may enhance the representation learning of contexts in news and other kinds of user generated content.

## Acknowledgments

The authors would like to thank Microsoft News for providing technical support and data in the experiments, and Jiun-Hung Chen (Microsoft News) and Ying Qiao (Microsoft News) for their support and discussions. This work was supported by the National Key Research and Development Program of China under Grant No. 2018YFC1604002, the National Natural Science Foundation of China under Grant Nos. U1836204, U1705261, U1636113, U1536201, and U1536207.

## References


Mingxiao An, Fangzhao Wu, Chuhan Wu, Kun Zhang, Zheng Liu, and Xing Xie. 2019. Neural news recommendation with long-and short-term user representations. In *ACL*, pages 336–345.

Trapit Bansal, Mrinal Das, and Chiranjib Bhattacharyya. 2015. Content driven user profiling for comment-worthy recommendations of news and blog articles. In *RecSys.*, pages 195–202. ACM.

Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *DLRS*, pages 7–10. ACM.

Abhinandan S Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. 2007. Google news personalization: scalable online collaborative filtering. In *WWW*, pages 271–280. ACM.

Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. Deepfm: a factorization-machine based neural network for ctr prediction. In *AAAI*, pages 1725–1731. AAAI Press.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *CIKM*, pages 2333–2338. ACM.

Wouter IJntema, Frank Goossen, Flavius Frasincar, and Frederik Hogenboom. 2010. Ontology-based news recommendation. In *Proceedings of the 2010 EDBT/ICDT Workshops*, page 16. ACM.

Dhruv Khattar, Vaibhav Kumar, Vasudeva Varma, and Manish Gupta. 2018. Weave&rec: A word embedding based 3-d convolutional network for news recommendation. In *CIKM*, pages 1855–1858. ACM.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Vaibhav Kumar, Dhruv Khattar, Shashank Gupta, and Vasudeva Varma. 2017. Word semantics based 3-d convolutional neural networks for news recommendation. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 761–764. IEEE.

Jianxun Lian, Fuzheng Zhang, Xing Xie, and Guangzhong Sun. 2018. Towards better representation learning for personalized news recommendation: a multi-channel deep fusion approach. In *IJCAI*, pages 3805–3811.

Jiahui Liu, Peter Dolan, and Elin Rønby Pedersen. 2010. Personalized news recommendation based on click behavior. In *IUI*, pages 31–40. ACM.

Shumpei Okura, Yukihiro Tagami, Shingo Ono, and Akira Tajima. 2017. Embedding-based news recommendation for millions of users. In *KDD*, pages 1933–1942. ACM.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Owen Phelan, Kevin McCarthy, Mike Bennett, and Barry Smyth. 2011. Terms of a feather: Content-based news recommendation and discovery using twitter. In *ECIR*, pages 448–459. Springer.

Steffen Rendle. 2012. Factorization machines with libfm. *TIST*, 3(3):57.

Jeong-Woo Son, A Kim, Seong-Bae Park, et al. 2013. A location-based news article recommendation with explicit localized semantic analysis. In *SIGIR*, pages 293–302. ACM.

Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. Dkn: Deep knowledge-aware network for news recommendation. In *WWW*, pages 1835–1844.

Chuhan Wu, Fangzhao Wu, Mingxiao An, Jianqiang Huang, Yongfeng Huang, and Xing Xie. 2019a. Neural news recommendation with attentive multi-view learning. In *IJCAI*, pages 3863–3869.

Chuhan Wu, Fangzhao Wu, Mingxiao An, Jianqiang Huang, Yongfeng Huang, and Xing Xie. 2019b. Npa: Neural news recommendation with personalized attention. In *KDD*, pages 2576–2584. ACM.

Chuhan Wu, Fangzhao Wu, Mingxiao An, Yongfeng Huang, and Xing Xie. 2019c. Neural news recommendation with topic-aware news representation. In *ACL*, pages 1154–1159.

Chuhan Wu, Fangzhao Wu, Junxin Liu, Shaojian He, Yongfeng Huang, and Xing Xie. Neural demographic prediction using search query.

Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. 2018. Drn: A deep reinforcement learning framework for news recommendation. In *WWW*, pages 167–176.

Qiannan Zhu, Xiaofei Zhou, Zeliang Song, Jianlong Tan, and Li Guo. 2019. Dan: Deep attention neural network for news recommendation. In *AAAI*, volume 33, pages 5973–5980.

# Reviews Meet Graphs: Enhancing User and Item Representations for Recommendation with Hierarchical Attentive Graph Neural Network

**Chuhan Wu**[1], **Fangzhao Wu**[2], **Tao Qi**[1], **Suyu Ge**[1], **Yongfeng Huang**[1],**and Xing Xie**[2]
[1]Department of Electronic Engineering, Tsinghua University, Beijing 100084, China
[2]Microsoft Research Asia, Beijing 100080, China
{wu-ch19, qit16, gsy17, yfhuang}@mails.tsinghua.edu.cn,
{fangzwu, xing.xie}@microsoft.com

## Abstract

User and item representation learning is critical for recommendation. Many of existing recommendation methods learn representations of users and items based on their ratings and reviews. However, the user-user and item-item relatedness are usually not considered in these methods, which may be insufficient. In this paper, we propose a neural recommendation approach which can utilize useful information from both review content and user-item graphs. Since reviews and graphs have different characteristics, we propose to use a multi-view learning framework to incorporate them as different views. In the review content-view, we propose to use a hierarchical model to first learn sentence representations from words, then learn review representations from sentences, and finally learn user/item representations from reviews. In addition, we propose to incorporate a three-level attention network into this view to select important words, sentences and reviews for learning informative user and item representations. In the graph-view, we propose a hierarchical graph neural network to jointly model the user-item, user-user and item-item relatedness by capturing the first- and second-order interactions between users and items in the user-item graph. In addition, we apply attention mechanism to model the importance of these interactions to learn informative user and item representations. Extensive experiments on four benchmark datasets validate the effectiveness of our approach.

## 1 Introduction

Precisely learning user and item representations is critical for recommendation (Tay et al., 2018). Many existing recommendation methods learn user and item representations from their rating matrix (Koren et al., 2009; Mnih and Salakhutdinov, 2008; Berg et al., 2017). For example, Koren et al. (2009) proposed to use singular value decomposition (SVD) to learn latent user and item representations based on the review ratings that users gave to items. However, there are massive users and items in online platforms, making the rating matrix between users and items very sparse. Thus, it is difficult for these methods to learn accurate user and item representations (Zheng et al., 2017).

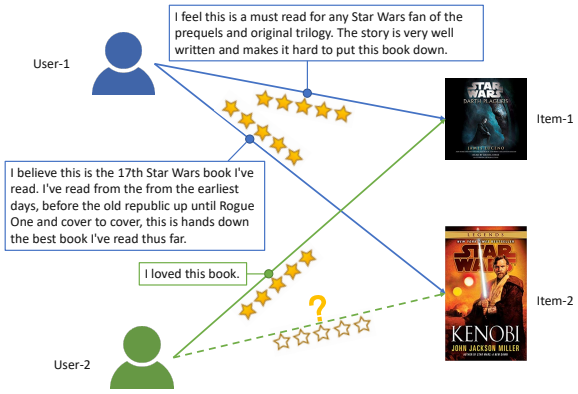


Figure 1: Two example users and items.

In recent years, several deep learning based recommendation methods are proposed to learn user and item representations from review texts (Zheng et al., 2017; Catherine and Cohen, 2017). For example, Zheng et al. (2017) proposed a DeepCoNN approach to learn the representations of users and items from their reviews via convolutional neural networks (CNN). However, these methods only consider the interactions of user-item pairs, while the relatedness between users or items are ignored, which may be insufficient for learning accurate user and item representations.

Our work is motivated by several observations. First, users who write reviews on the same products may have some relatedness. For example, in Fig. 1 both User-1 and User-2 give 5 stars to Item-1, and we can infer that both users are interested in Star Wars. Second, items that commented by the



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4884–4893,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

same user may also have relatedness. For example, in Fig. 1 both items are commented by User-1, and they share the same topics on Star Wars. Third, modeling the user-user and item-item interactions is useful for recommendation. For example, in Fig. 1 both users may have similar interests on Star Wars, and both books have related topics. Thus, it may be effective to recommend Item-2 to User-2. Fourth, the interactions between users and items, e.g., reviews, may have different importance for representing users and items. For example, in Fig. 1 the interaction between User-1 and Item-1 is more important than that between User-2 and Item-1 in representing this item, since the review of User-1 contain more details about item properties. In addition, different words and sentences in the same review may also have different importance. For instance, the sentence "this is the best book I've read thus far" is more important than "I've read from the earliest days", and the word "best" is more important than "read" in the former sentence.

In this paper, we propose a reviews meet graphs approach (*RMG*) to combine reviews and the information of user-item graphs via a multi-view learning framework. In the review content-view, we use a hierarchical model to learn user and item representations. It first learns sentence representations from words, then learn review representations from sentences, and finally learn user/item representations from reviews. In addition, we propose to apply a three-level attention network to select important words, sentences and reviews to learn informative user and item representations. In the graph-view, we propose to use a graph neural network to capture the user-item, user-user, and item-item relatedness in the user-item bipartite graphs by modeling the first-order and second-order interactions between different users and items. In addition, we propose to incorporate attention mechanism into the graph neural network to model the importance of these interactions for informative user and item representation learning. Extensive experiments on four benchmark datasets validate that our approach can effectively improve the performance of recommendation and outperform many baseline methods.

## 2 Related Work

Many previous works have studied the problem of learning user and item representations from reviews for recommendation (Zhang et al., 2014; Diao et al., 2014; He et al., 2015; Tan et al., 2016; Ren et al., 2017). Many of existing methods rely on topic models to extract topics from reviews for user and item representation (McAuley and Leskovec, 2013; Ling et al., 2014; Bao et al., 2014). For example, McAuley and Leskovec (2013) proposed a Hidden Factors as Topics (HFT) method to learn latent user and item factors from review content via a latent Dirichlet allocation (LDA) model. Ling et al. (2014) proposed a Ratings Meet Reviews (RMR) approach to represent users and items by first extracting topics from reviews and then aligning the dimensions of these topics with the latent user factors learned from review ratings via matrix factorization. Bao et al. (2014) proposed a TopicMF method to represent users and items by incorporate the topics extracted from review texts to enhance the learning of latent user and item factors from the rating matrix via non-negative matrix factorization (NMF). However, these methods only extract topics from reviews, and cannot effectively utilize the contexts and word orders in reviews, both of which are important for learning accurate user and item representations.

In recent years, several deep learning based methods proposed to learn user and item representations from original review texts (Zhang et al., 2016; Zheng et al., 2017; Catherine and Cohen, 2017; Seo et al., 2017b,a; Chen et al., 2018; Tay et al., 2018; Wu et al., 2019). For example, Zheng et al. (2017) proposed a DeepCoNN approach to learn user and item representations from reviews via CNN networks. Catherine and Cohen (2017) proposed a TransNets approach to use CNNs to learn user and item representations. In their approach, these representations are regularized to be close to the representations of reviews from the target user-item pairs. Seo et al. (2017b) proposed to use CNN networks to learn user and item representations, and applied a word-level attention network to select important words. Chen et al. (2018) proposed to learn review representations using CNNs and they modeled the usefulness of reviews via a review-level attention network to learn informative user and item representation learning. However, these methods can only model the user-item interactions, while the user-user and item-item relatedness is not considered. In addition, these methods usually aggregate reviews or

their texts together, and cannot fully model their informativeness. Different from these methods, in our approach we propose a review meet graph approach which can combine reviews with user-item graphs via a multi-view learning framework to enhance user and item representation learning. In the review text-view, we use a hierarchical framework to learn sentence representations from words first, then learn review representations form sentences, and finally learns user/item representations from their reviews. In addition, we apply attention network at each level to select important words, sentences and reviews. In the graph-view, our approach uses a hierarchical graph neural network to mine the user-item, user-user and item-item relatedness from user-item graphs, and attentively selects the interactions between users and items. Experiments on four benchmark datasets validate the effectiveness of our approach in recommendation.

## 3 Our Approach

In this section, we will introduce our reviews meet graphs approach for recommendation (denoted as *RMG*). The architecture of our basic *RMG* approach is shown in Fig. 2. Our approach consists of two different views, i.e., a *review content-view* and a *graph-view*.

### 3.1 Review Content View

The *review content-view* module is used to learn representations of users and items from their review texts. It contains three modules, i.e, *sentence encoder*, *review encoder* and *user/item encoder*.

There are three layers in the *sentence encoder*. The first one is word embedding. It is used to convert a sentence $s$ into a low-dimensional semantic vector sequence. Denote the word sequence of $s$ as $[w_1, w_2, ..., w_M]$, where $M$ is its length. It is converted into a vector sequence $[\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_M]$ via a pre-trained embedding matrix.

The second one is a convolutional neural network (CNN). Local contexts are important for review representation learning. For example, in the sentence "I am a star wars fan", the local contexts of "wars" such as "star" and "fan" are useful for inferring this sentence is about a famous movie series. Thus, we employ a CNN over word embeddings to learn contextual word representations by capturing local contexts. It takes the embedding sequence $[\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_M]$ as input, and outputs a sequence of word contextual representation vectors $[\mathbf{c}_1^w, \mathbf{c}_2^w, ..., \mathbf{c}_M^w]$.

The third layer is a word-level attention network. Different words in the same sentence may have different informativeness in representing users and items. For example, in the sentence "This story book is very interesting" , the word "interesting" is more important than the word "story" in representing this book. Thus, we use an attention network over word representations to learn informative sentence representations by selecting important words. The attention weight of the $i$-th word $w_i$ is computed as follows:

$$a_i^w = \tanh(\mathbf{w}_w \times \mathbf{c}_i^w + \mathbf{b}_w), \tag{1}$$

$$\alpha_i^w = \frac{\exp(a_i^w)}{\sum_{j=1}^{M} \exp(a_j^w)}, \tag{2}$$

where $\mathbf{w}_w$ and $\mathbf{b}_w$ are parameters. The final representation of the sentence $s$ is the summation of the contextual word representations weighted by their attention weights, i.e., $\mathbf{s} = \sum_{i=1}^{M} \alpha_i^w \mathbf{c}_i^w$.

The *review encoder* module is used to learn representations of reviews from their sentence representations. There are two layers in the *review encoder* module. The first layer is a sentence-level CNN network. Neighboring sentences usually have some relatedness in their content. For example, in the book review "The book is very well written. I will recommend it to others", the two neighboring sentences have close relatedness and they both express positive opinion towards the book. Thus, we use a sentence-level CNN network to learn contextual sentence representations by capturing sentence-level local contexts. We denote the representation sequence of the sentences in a review $r$ as $[\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_N]$. The CNN layer takes the representation sequence as input, and outputs a sequence of contextual sentence representations, denoted as $[\mathbf{c}_1^s, \mathbf{c}_2^s, ..., \mathbf{c}_N^s]$.

The second layer is a sentence-level attention network. Different sentences in the same review may have different informativeness for modeling users and items. For example, the sentence "The book tells a good story" is more informative than the sentence "I read this book today" in representing this book. Thus, we use a sentence-level attention network to select important sentences to learn informative representations of reviews for recommendation. The attention weight $\alpha_i^s$ of the $i$-th sentence is formulated as follows:

$$a_i^s = \tanh(\mathbf{w}_s \times \mathbf{c}_i^s + \mathbf{b}_s), \tag{3}$$

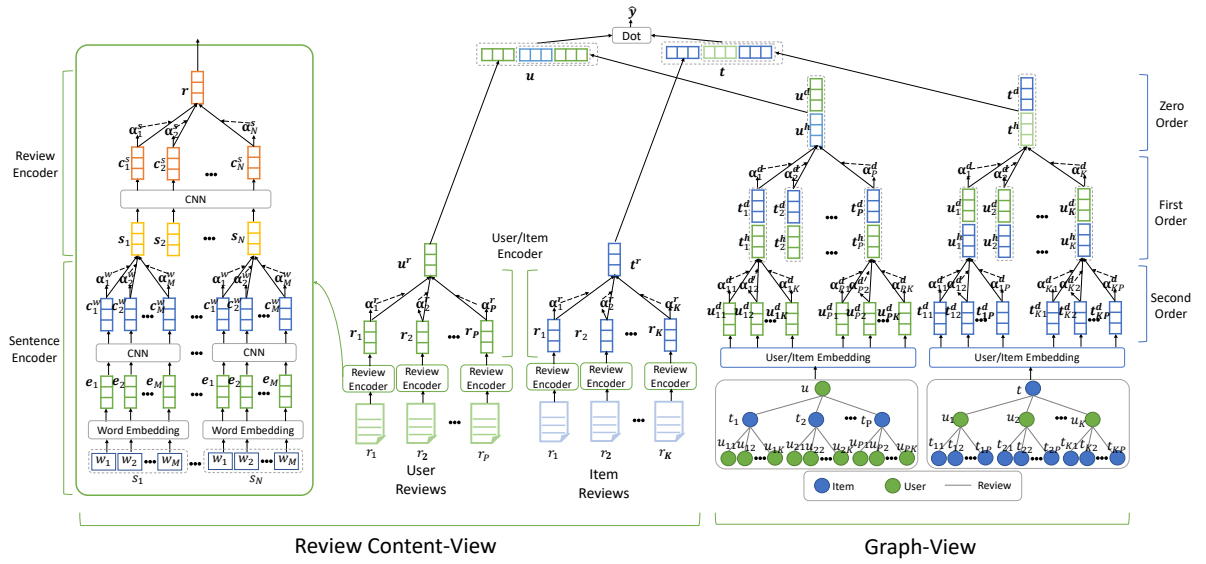


Figure 2: The framework of our *RMG* approach.

$$\alpha_i^s = \frac{\exp(a_i^s)}{\sum_{j=1}^{N} \exp(a_j^s)}, \tag{4}$$

where $\mathbf{w}_s$ and $\mathbf{b}_s$ are parameters in the attention network. The final contextual representation of the review $r$ the summation of the contextual sentence representations weighted by their attention weights, i.e., $\mathbf{r} = \sum_{i=1}^{N} \alpha_i^s \mathbf{c}_i^s$. We apply the *review encoder* to encode the reviews of users and items into their representation vectors.

The *user/item encoder* is used to learn representations of users and items from the their review representations. Different reviews usually have different informativeness in characterizing users and items. For example, in Fig. 1, the first review of Item-1 is more informative than its second review. Thus, we use a review-level attention network to recognize informative reviews and attend to them. Denote the reviews written by a user as $[r_1, r_2, ..., r_P]$, where $P$ is the number of reviews. The attention weight of the review $r_i$ is formulated as follows:

$$a_i^r = \tanh(\mathbf{w}_r \times \mathbf{r}_i + \mathbf{b}_r), \tag{5}$$

$$\alpha_i^r = \frac{\exp(a_i^r)}{\sum_{j=1}^{P} \exp(a_j^r)}, \tag{6}$$

where $\mathbf{w}_r$ and $\mathbf{b}_r$ are parameters. The user representation learned from reviews is the summation of the review representations weighted by their attention weights, i.e., $\mathbf{u}^r = \sum_{i=1}^{P} \alpha_i^r \mathbf{r}_i$. Denote the $K$ reviews commented on an item as $[r_1, r_2, ..., r_K]$. The item representation $\mathbf{t}^r$ learned from these reviews is computed similarly, i.e., $\mathbf{t}^r = \sum_{i=1}^{K} \alpha_i^r \mathbf{r}_i$, where $\alpha_i^r$ is the attention weight of the review $r_i$.

### 3.2 User-Item Graph View

The *graph-view* module is used to learn representations of users and items by modeling their interactions in the user-item bipartite graph, as shown in Fig. 2. In this bipartite graph, users and items are represented as nodes, and the interactions of user-item pairs are regarded as edges. However, different from typical graph neural networks (Veličković et al., 2017), the input of typical review-based recommender systems is only a user-item pair (two nodes with an edge) rather than the entire user-item graph. Therefore, it is difficult to directly capture high-order interactions between users and items via a stacked graph neural networks. To solve this problem, in our approach we propose a hierarchical attentive graph neural network for recommendation which can model high-order user-item interactions. It contains three core components, i.e., *second-order* encoder, *first-order* encoder, and *zero-order* encoder. We will introduce each one as follows.

The first one is a *second-order* encoder. Since a user-item graph is bipartite, there are no edges among users (or items). Thus, it is difficult to directly model their relatedness. Luckily, users bought the same items may have some relatedness, as well as the items bought by the same users. Thus, we propose to indirectly mine the user-

user and item-item relatedness by modeling the second-order interactions in the user-item graph, as shown in Fig. 2. For each user-item pair $(u, t)$ in the training set, we denote the items that this user gave ratings to as $[t_1, t_2, ..., t_P]$. Among these items, we denote the users who commented the $i$-th item as $[u_{i1}, u_{i2}, ..., u_{iK}]$. Motivated by typical graph neural networks (Veličković et al., 2017), we incorporate the embedding of user and item IDs as the node representation in the user-item graph, and then learn the representations of each item from the representations of the users who has commented this item. Usually different users who bought the same item may have different informativeness in representing this item. For example, in Fig. 1, the first user is more important than second user in representing the first item. Thus, we propose to use an attentive graph neural network to model the importance of the users that connected to the item node. Denote the representations of these users as $[\mathbf{u}_{i1}, \mathbf{u}_{i2}, ..., \mathbf{u}_{iK}]$. The attention weight $a_{ij}^d$ of the $j$-th user who comments the item $t_i$ is computed as:

$$a_{ij}^d = \tanh(\mathbf{w}_2 \times \mathbf{u}_{ij} + b_2), \tag{7}$$

$$\alpha_{ij}^d = \frac{\exp(a_{ij}^d)}{\sum_{k=1}^{K} \exp(a_{ik}^d)}, \tag{8}$$

where $\mathbf{w}_2$ and $b_2$ are parameters in this network. The user-based representation $\mathbf{t}_i^h$ of the item $t_i$ is the summation of the embeddings of its related user weighted by their attention weights, which is formulated as:

$$\mathbf{t}_i^h = \sum_{k=1}^{K} \alpha_{ik}^d \mathbf{u}_{ik}, \tag{9}$$

The second one is a *first-order* encoder. Since some latent properties of this item may not be revealed by the interactions with users, we incorporate the embedding of item IDs to enhance representation learning. The final representation of node $t_i$ is the concatenation of the item node representation $\mathbf{t}_i^h$ learned from user node representations and the item ID embedding $\mathbf{t}_i^d$, i.e., $\mathbf{t}_i = [\mathbf{t}_i^h, \mathbf{t}_i^d]$. The representations of the users (denoted as $[\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_K]$) who interact with the item $t$ are computed similarly. Similar with the *second-order* encoder, we first learn the item-based representation $\mathbf{u}^h$ of the user $u$ from the representations of its neighbor item nodes in an attentive manner. Denote the attention weight of the $i$-th item that the user $u$ commented as $a_i^d$, which is computed as:

$$a_i^d = \tanh(\mathbf{w}_1 \times \mathbf{t}_i + b_1), \tag{10}$$

$$\alpha_i^d = \frac{\exp(a_i^d)}{\sum_{p=1}^{P} \exp(a_p^d)}, \tag{11}$$

where $\mathbf{w}_1$ and $b_1$ are parameters. The item-based representation $\mathbf{u}^h$ of the user $u$ is the summation of the item representations associated with this user weighted by their attention weights, which is formulated as:

$$\mathbf{u}^h = \sum_{p=1}^{P} \alpha_p^d \mathbf{t}_p, \tag{12}$$

The third one is *zero-order* encoder. It learns the graph-based representation of user $u$ by using the concatenation of the user node representation $\mathbf{u}^h$ and the user ID embedding $\mathbf{u}^d$, i.e., $\mathbf{u}^g = [\mathbf{u}^h, \mathbf{u}^d]$. The representation of the item $t$ is computed similarly. Denote the representation of the item $t$ learned from user-item interactions as $\mathbf{t}^h$, and the embedding of its ID as $\mathbf{t}^d$. Then its graph-based representation $\mathbf{t}^g$ is calculated as $\mathbf{t}^g = [\mathbf{t}^h, \mathbf{t}^d]$.

### 3.3 Rating Scoring

In our approach, the final representations of users and items are the concatenation of the representations learned from the review content-view and graph-view, i.e., $\mathbf{u} = [\mathbf{u}^g, \mathbf{u}^r]$ and $\mathbf{t} = [\mathbf{t}^g, \mathbf{t}^r]$. The rating score of a user-item pair is predicted by the inner product of user and item representations, i.e., $\hat{y} = \mathbf{u}^T \mathbf{t}$.

### 3.4 Model Training

For model training, mean squared error is used as the loss function:

$$\mathcal{L} = \frac{1}{T} \sum_{i=1}^{T} (\hat{y}_i - y_i)^2, \tag{13}$$

where $T$ denotes the number of user-item pairs for training, $\hat{y}_i$ and $y_i$ respectively denote the predicted and gold rating score of the $i$-th user-item pair. By optimizing the loss function, all parameters can be tuned via backward propagation.

## 4 Experiments

### 4.1 Datasets and Experimental Settings

We conducted experiments on four widely used benchmark datasets in different domains and

| Dataset | #users | #items | #reviews |
|---|---|---|---|
| **Toys** | 19,412 | 11,924 | 167,597 |
| **Kindle** | 68,223 | 61,935 | 982,619 |
| **Movies** | 123,960 | 50,052 | 1,679,533 |
| **Yelp** | 199,445 | 119,441 | 3,072,129 |

Table 1: Statistics of the benchmark datasets.

scales to validate the effectiveness of our approach. Following (Chen et al., 2018), we used three datasets from the Amazon collection[1](He and McAuley, 2016), i.e., *Toys_and_Games*, *Kindle_Store*, and *Movies_and_TV* (respectively denoted as *Toys*, *Kindle* and *Movies*). Another dataset is from Yelp Challenge 2017[2] (denoted as **Yelp**), which is a large collection of restaurant reviews. Following (Chen et al., 2018), we only kept the users and items which have at least 5 reviews. The detailed statistics of the four datasets are summarized in Table 1. The ratings in these datasets are ranged in [1, 5].

In our experiments, the word embeddings were 300-dimensional. We used the pre-trained Google embedding (Mikolov et al., 2013) for initialization. The CNNs had 150 filters and their window sizes were set to 3. The user and item embedding vector was 100-dimensional. We applied 25% dropout (Srivastava et al., 2014) to each layer in our model to mitigate overfitting. Adam (Kingma and Ba, 2014) was used as the optimization algorithm. The batch size was set to 20. We randomly selected 80% of the user-item pairs in each dataset for training, 10% for validation and 10% for test. All hyperparameters were tuned according to the validation set. We independently repeated each experiment 5 times and reported the average Root Mean Square Error (RMSE).

### 4.2 Performance Evaluation

We evaluate the performance of our approach by comparing it with several baseline methods[3]. The methods to be compared include:

- *PMF*: Probabilistic Matrix Factorization, which models users and items based on ratings via matrix factorization (Mnih and Salakhutdinov, 2008).
- *NMF*: Non-negative Matrix Factorization for recommendation based on rating scores (Lee and Seung, 2001).
- *SVD++*: The recommendation method based on rating matrix via SVD and similarities between items (Koren, 2008).
- *HFT*: Hidden Factor as Topic (HFT), a method to combine reviews with ratings via LDA (McAuley and Leskovec, 2013).
- *DeepCoNN*: Deep Cooperative Neural Networks, a neural method to jointly model users and items from their reviews via CNN (Zheng et al., 2017).
- *Attn+CNN*: Attention-based CNN, which uses both CNN and attention over word embeddings to learn user and item representation from reviews (Seo et al., 2017b).
- *NARRE*: Neural Attentional Rating Regression with Review-level Explanations, which uses attention mechanism to model the informativeness of reviews for recommendation (Chen et al., 2018).
- *RMG-review*: A variant of our approach with the review content-view only.
- *RMG*: Our reviews meet graphs approach.

In Table 2, we present a simple comparison of different methods by summarizing the information they incorporated. Traditional matrix factorization methods such as *PMF*, *NMF* and *SVD++* learn user and item representations based on review ratings only, while other methods can exploit both rating scores and reviews texts to learn representations for recommendation. Among these methods, *HFT* is based on topic models and cannot effectively utilize the contexts and word orders of words. *DeepCoNN* and *Attn+CNN* simply concatenate all reviews into a long document, and cannot model the informativeness of different reviews. Although *NARRE* can model review informativeness using a review-level attention network, it simply aggregates all sentences and cannot model the informativeness of words and sentences. *RMG-review* can simultaneously recognize important word, sentences and reviews, but does not consider the information in user-item graphs. Different from these methods, our *RMG*

[1] http://jmcauley.ucsd.edu/data/amazon

[2] https://www.yelp.com/dataset_challenge

[3] Since in our framework the input is a single user-item pair, other popular graph neural architectures such as GCN may not be applicable. We will also explore incorporating them in our future work.

| Information | PMF | NMF | SVD++ | HFT | DeepCoNN | Attn+CNN | NARRE | RMG-review | RMG |
|---|---|---|---|---|---|---|---|---|---|
| Rating score | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Review text | | | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Context & word order | | | | | ✓ | ✓ | ✓ | ✓ | ✓ |
| Review attention | | | | | | | ✓ | ✓ | ✓ |
| Word attention | | | | | | ✓ | | ✓ | ✓ |
| Sentence attention | | | | | | | | ✓ | ✓ |
| User-item graph | | | | | | | | | ✓ |

Table 2: Comparisons of the information used in different methods.

| Methods | Toys | Kindle | Movies | Yelp |
|---|---|---|---|---|
| PMF | 1.3076 | 0.9914 | 1.2920 | 1.3340 |
| NMF | 1.0399 | 0.9023 | 1.1125 | 1.2916 |
| SVD++ | 0.8860 | 0.7928 | 1.0447 | 1.1735 |
| HFT | 0.8925 | 0.7917 | 1.0291 | 1.1699 |
| DeepCoNN | 0.8890 | 0.7876 | 1.0128 | 1.1642 |
| Attn+CNN | 0.8805 | 0.7796 | 0.9984 | 1.1588 |
| NARRE | 0.8769 | 0.7783 | 0.9965 | 1.1559 |
| RMG-review | 0.8685 | 0.7511 | 0.9679 | 1.1310 |
| RMG | **0.8438** | **0.7366** | **0.9514** | **1.1113** |

Table 3: The performance of different methods on the benchmark datasets. The metric is RMSE.

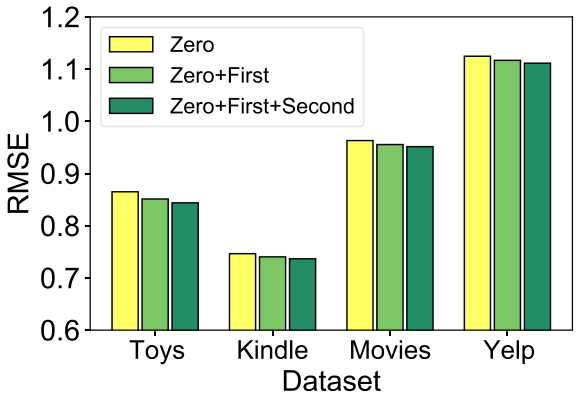


Figure 3: The influence of the depth of the hierarchical attentive graph neural network.

approach can learn user and item representations from both reviews and graphs, which has the potential to learn more informative user and item representations.

The performance of different methods is summarized in Table 3. The results lead to several observations. First, the methods which combine reviews and ratings (e.g., *HFT*, *DeepCoNN*, *NARRE* and *RMG*) usually outperform the methods based on ratings only (*PMF*, *NMF* and *SVD++*). This is probably because reviews usually contain rich clues on user preferences and item properties, which is useful for user and item representation learning for recommendation.

Second, among the methods considering the content information of reviews, neural network based methods (e.g., *DeepCoNN*, *Attn+CNN*, *NARRE* and *RMG*) usually outperform *HFT*, which is based on topic models. This is probably because *HFT* rely on bag-of-words features to extract topics from reviews, and cannot utilize the contexts and orders of words. Different from topic model based methods, neural methods can better model user preferences and item properties by utilizing the rich semantic information in reviews, which is beneficial for recommendation.

Third, among the methods based on deep learning, *RMG-review* and *RMG* can consistently outperform other baseline methods. This is because different words, sentences and reviews usually have different informativeness for modeling users and items. Different from *Attn+CNN*, *NARRE* and *DeepCoNN*, both *RMG-review* and *RMG* use a hierarchical model to learn user and item representations. In addition, our approach can select important words, sentences and reviews via a three-level attention network to help learn more accurate user and item representations for recommendation.

Fourth, our *RMG* approach which incorporates information of user-item graph outperforms other methods based on reviews only (e.g., *RMG-review*), and further hypothesis test results show that the advantage over *RMG-review* is significant. This is because the information of user-item graphs is useful for recommendation, and modeling the interactions between users amd items in graphs can help learn more accurate user and item representations. This result validates the effectiveness of our approach.

### 4.3 Influence of the Depth of Graph Neural Network

In this section, we conducted several experiments to explore the influence of the depth of the graph neural network on our approach. We compare the performance of our approach with its variants

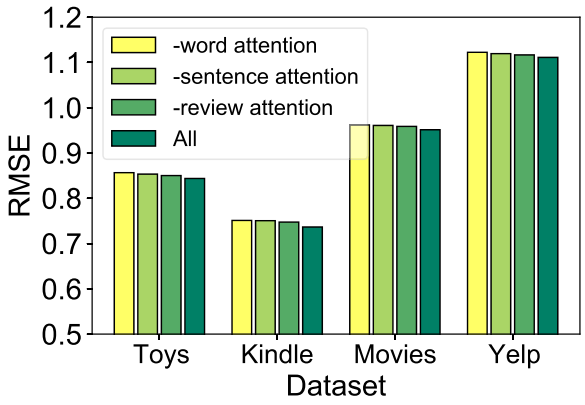


(a) Attentions in the review content-view.

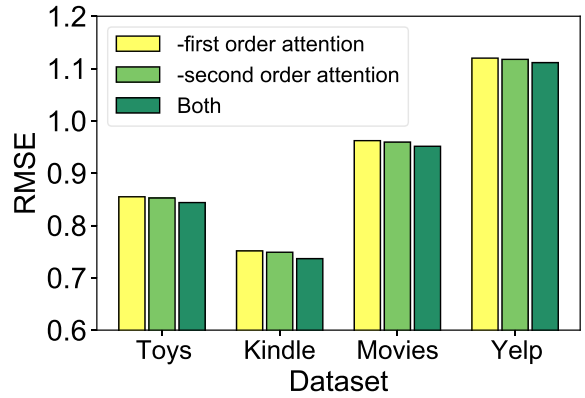


(b) Attentions in the graph-view.

Figure 4: The effectiveness of different attentions.

with the zero-order encoder only or with both the zero- and first-order encoder. The results are illustrated in Fig. 3. According to the results, we find the performance of our approach can be consistently improved as the depth increases. This is because the first-order information of graph contains the interactions between users and items, and the second-order information can reveal the user-user and item-item relatedness. Thus, more information can be incorporated as the depth increases, which can benefit user and item representation learning. Although the performance may be further improved when the depth gets deeper, in our approach we only set the depth of the graph neural network to 2. This is because the user-item, user-user and item-item interactions are usually sufficient for modeling users and items, and their high-order relatedness is usually weak. In addition, the computational cost is huge when higher-order information is considered. Thus, a moderate depth of the graph neural network (i.e., 2) is the most appropriate for our approach.

### 4.4 Effectiveness of Attention Mechanism

In this section, we conducted experiments to explore the effectiveness of the attention networks in our approach. First, we want to validate the effectiveness of the attention network in the review content-view. We compare the performance of several variants of our approach with one kind of attention network removed to evaluate the contribution of different attentions. The results are shown in Fig. 4(a). From Fig. 4(a), the word-level attention can effectively improve the performance of our approach. This is because different words in the same review have different importance in representing this review. Therefore, selecting important words via a word-level attention network can learn more informative sentence representations. In addition, the sentence-level attention is also useful. This is because different sentences also have different informativeness, and selecting important sentences using a sentence-level attention network can benefit review representation learning. Besides, the review-level attention is also useful in our approach. This is because different reviews have different informativeness for learning user and item representations, and distinguishing informative reviews from the uninformative ones can help learn more accurate user and item representations. Moreover, combining all three kinds of attentions can further improve our approach, which validates the effectiveness of our hierarchical attention framework.

Then, we want to validate the effectiveness of the attention network in the graph-view. We compare our approach with its variants by removing the attention networks in the first-order or second-order encoder to validate the effectiveness of them. The results are shown in Fig. 4(b). According to Fig. 4(b), we find the attention networks in both first- and second-order encoders are important in our approach. This may be because the interactions between users and items usually have different importance in representing them, and distinguish informative interactions from uninformative ones is useful for learning better user and item representations. These results validate the effectiveness of the attention networks in the graph-view.

### 4.5 Case Study

In this section, we conducted several case studies to visually explore our *RMG* approach can learn better user and item representations. For instance,

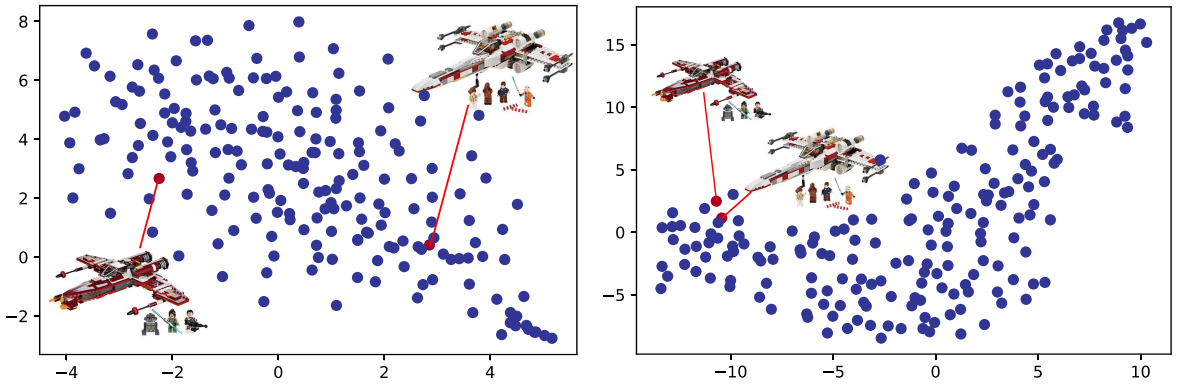

(a) *RMG-review* (b) *RMG*

Figure 5: Visualization of the item representations learned by *RMG-review* and *RMG* via t-SNE. Each point represents an item. The red points represent two similar items.

we use t-SNE to visualize the representations of several randomly selected items in the *Toys* dataset learned by *RMG-review* and *RMG*. Among these items there are two very similar items, both of which are Lego toys of the planes in Star Wars. The results are respectively shown in Fig. 5(a) and 5(b)[4]. From the results, we find the distance between the representations of these two items learned by *RMG-review* is large, which usually indicates their similarity is low. This may be because the reviews of both items are insufficient (only 3 reviews appeared in the training set), making it difficult for *RMG-review* to learn accurate representations of them. Different from *RMG-review*, our *RMG* approach can correctly recognize that both items are very similar. This is probably because our approach can mine the user-user and item-item relatedness by modeling the first-order and second-order interactions between users and items. Thus, our *RMG* approach can learn better user and item representations.

## 5 Conclusion and Future Work

In this paper, we propose a reviews meet graphs approach for recommendation which can exploit information of user-item graphs. In our approach, we use a multi-view framework to incorporate review texts and user-item graphs as different views. In the review content-view, we use a hierarchical model to first learn sentence representations from words, then learn review representations from sentences, and finally learn user/item representations from reviews. In addition, we apply a three-level attention network to select important words, sentences and reviews for informative representation learning. In the graph-view, we propose to use a graph neural network to jointly mine user-item, user-user and item-item relatedness by modeling the first- and second-order interactions between users and items in graphs. In addition, we propose to apply attention mechanism to model the importance of these interactions to learn more accurate user and item representations. Experiments on four benchmark datasets validate the effectiveness of our approach.

In our future work, we will explore several potential directions. First, since we only use the local neighbors of nodes in the user-item graph, we will explore the use of other kinds of graph neural networks to further enhance the learning of users and items. Second, since there are also other kinds of user and item information, we will explore how to incorporate heterogeneous user and item information to benefit recommendation. Third, we will work on how to reduce the computational cost of our approach, which can improve the applicability of our approach.

## Acknowledgments

This work was supported by the National Key Research and Development Program of China under Grant number 2018YFC1604002, the National Natural Science Foundation of China under Grant numbers U1836204, U1705261, U1636113, U1536201, and U1536207.

[4]We aims to show the relative similarities between the representations of the two items, and their absolute distances are not comparable in the two figures.

## References


Yang Bao, Hui Fang, and Jie Zhang. 2014. Topicmf: Simultaneously exploiting ratings and reviews for recommendation. In *AAAI*, volume 14, pages 2–8.

Rianne van den Berg, Thomas N Kipf, and Max Welling. 2017. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263*.

Rose Catherine and William Cohen. 2017. Transnets: Learning to transform for recommendation. In *RecSys*, pages 288–296. ACM.

Chong Chen, Min Zhang, Yiqun Liu, and Shaoping Ma. 2018. Neural attentional rating regression with review-level explanations. In *WWW*, pages 1583–1592.

Qiming Diao, Minghui Qiu, Chao-Yuan Wu, Alexander J Smola, Jing Jiang, and Chong Wang. 2014. Jointly modeling aspects, ratings and sentiments for movie recommendation (jmars). In *KDD*, pages 193–202.

Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *WWW*, pages 507–517.

Xiangnan He, Tao Chen, Min-Yen Kan, and Xiao Chen. 2015. Trirank: Review-aware explainable recommendation by modeling aspects. In *CIKM*, pages 1661–1670.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD*, pages 426–434.

Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37.

Daniel D Lee and H Sebastian Seung. 2001. Algorithms for non-negative matrix factorization. In *NIPS*, pages 556–562.

Guang Ling, Michael R Lyu, and Irwin King. 2014. Ratings meet reviews, a combined approach to recommend. In *RecSys*, pages 105–112.

Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *RecSys*, pages 165–172.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Andriy Mnih and Ruslan R Salakhutdinov. 2008. Probabilistic matrix factorization. In *NIPS*, pages 1257–1264.

Zhaochun Ren, Shangsong Liang, Piji Li, Shuaiqiang Wang, and Maarten de Rijke. 2017. Social collaborative viewpoint regression with explainable recommendations. In *WSDM*, pages 485–494.

Sungyong Seo, Jing Huang, Hao Yang, and Yan Liu. 2017a. Interpretable convolutional neural networks with dual local and global attention for review rating prediction. In *RecSys*, pages 297–305. ACM.

Sungyong Seo, Jing Huang, Hao Yang, and Yan Liu. 2017b. Representation learning of users and items for review rating prediction using attention-based convolutional neural network. In *MLRec*.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958.

Yunzhi Tan, Min Zhang, Yiqun Liu, and Shaoping Ma. 2016. Rating-boosted latent topics: Understanding users and items with ratings and reviews. In *IJCAI*, pages 2640–2646.

Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2018. Multi-pointer co-attention networks for recommendation. In *KDD*, pages 2309–2318.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Chuhan Wu, Fangzhao Wu, Junxin Liu, and Yongfeng Huang. 2019. Hierarchical user and item representation with three-tier attention for recommendation. In *NAACL*, pages 1818–1826.

Wei Zhang, Quan Yuan, Jiawei Han, and Jianyong Wang. 2016. Collaborative multi-level embedding learning from reviews for rating prediction. In *IJCAI*, pages 2986–2992.

Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. 2014. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *SIGIR*, pages 83–92.

Lei Zheng, Vahid Noroozi, and Philip S Yu. 2017. Joint deep modeling of users and items using reviews for recommendation. In *WSDM*, pages 425–434. ACM.

# Event Representation Learning Enhanced with External Commonsense Knowledge

**Xiao Ding, Kuo Liao, Ting Liu*, Zhongyang Li, Junwen Duan**
Research Center for Social Computing and Information Retrieval
Harbin Institute of Technology, China
{xding, kliao, tliu, zyli, jwduan}@ir.hit.edu.cn

## Abstract

Prior work has proposed effective methods to learn event representations that can capture syntactic and semantic information over text corpus, demonstrating their effectiveness for downstream tasks such as script event prediction. On the other hand, events extracted from raw texts lacks of commonsense knowledge, such as the intents and emotions of the event participants, which are useful for distinguishing event pairs when there are only subtle differences in their surface realizations. To address this issue, this paper proposes to leverage external commonsense knowledge about the intent and sentiment of the event. Experiments on three event-related tasks, i.e., event similarity, script event prediction and stock market prediction, show that our model obtains much better event embeddings for the tasks, achieving 78% improvements on hard similarity task, yielding more precise inferences on subsequent events under given contexts, and better accuracies in predicting the volatilities of the stock market[1].

## 1 Introduction

Events are a kind of important *objective* information of the world. Structuralizing and representing such information as machine-readable knowledge are crucial to artificial intelligence (Li et al., 2018b, 2019). The main idea is to learn distributed representations for structured events (i.e. event embeddings) from text, and use them as the basis to induce textual features for downstream applications, such as script event prediction and stock market prediction.

Parameterized additive models are among the most widely used for learning distributed event representations in prior work (Granroth-Wilding and Clark, 2016; Modi, 2016), which passes the concatenation or addition of event arguments' word embeddings to a parameterized function. The function maps the summed vectors into an event embedding space. Furthermore, Ding et al. (2015) and Weber et al. (2018) propose using neural tensor networks to perform semantic composition of event arguments, which can better capture the interactions between event arguments.

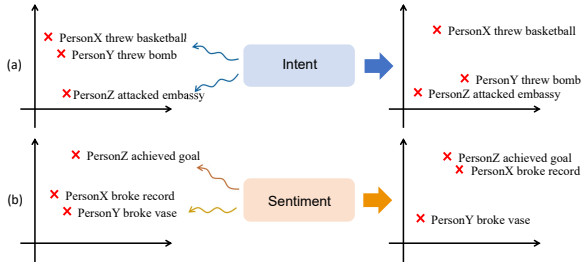


Figure 1: Intent and sentiment enhanced event embeddings can distinguish distinct events even with high lexical overlap, and find similar events even with low lexical overlap.

This line of work only captures shallow event semantics, which is not capable of distinguishing events with subtle differences. On the one hand, the obtained event embeddings cannot capture the relationship between events that are syntactically or semantically similar, if they do not share similar word vectors. For example, as shown in Figure 1 (a), "*PersonX threw bomb*" and "*PersonZ attacked embassy*". On the other hand, two events with similar word embeddings may have similar embeddings despite that they are quite unrelated, for example, as shown in Figure 1 (b), "*PersonX broke record*" and "*PersonY broke vase*". Note that in this paper, similar events generally refer to events with strong semantic relationships rather than just the same events.

One important reason for the problem is the lack

*Corresponding author

[1]The code and data are available on https://github.com/MagiaSN/CommonsenseERL_EMNLP_2019.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4894–4903, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

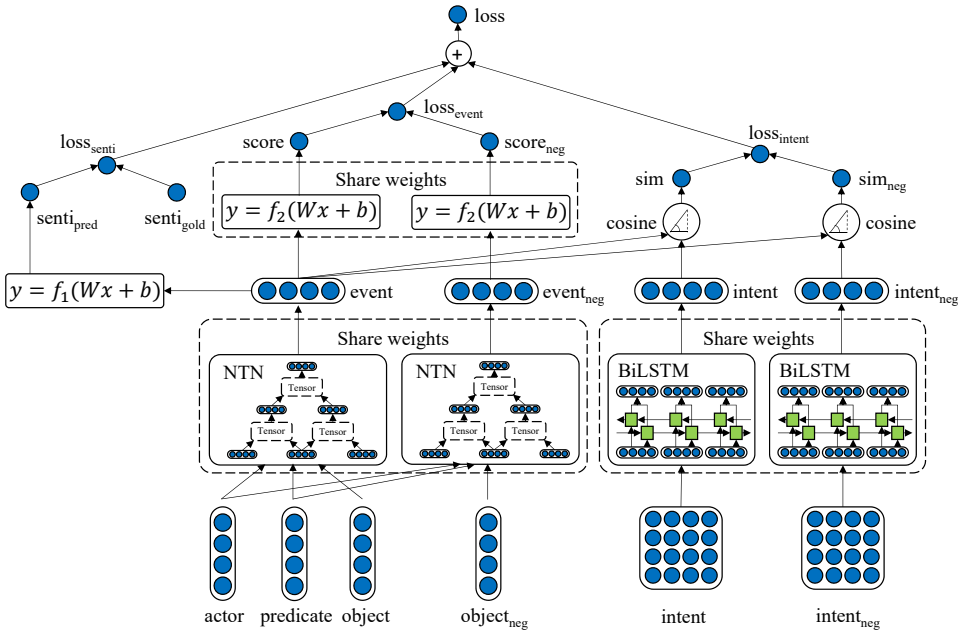


Figure 2: Architecture of the joint embedding model. $event_{neg}$ refers to the corrupted event tuple, which is derived by replacing each word of the event object with a random word in our dictionary. $intent_{neg}$ is the incorrect intent for the given event, which is randomly selected from the annotated dataset.

of the external commonsense knowledge about the mental state of event participants when learning the *objective* event representations. In Figure 1 (a), two event participants "*PersonY*" and "*PersonZ*" may carry out a terrorist attack, and hence, they have the same intent: "*to bloodshed*", which can help representation learning model maps two events into the neighbor vector space. In Figure 1 (b), a change to a single argument leads to a large semantic shift in the event representations, as the change of an argument can result in different emotions of event participants. Who "*broke the record*" is likely to be happy, while, who "*broke a vase*" may be sad. Hence, intent and sentiment can be used to learn more fine-grained semantic features for event embeddings.

Such commonsense knowledge is not explicitly expressed but can be found in a knowledge base such as Event2Mind (Rashkin et al., 2018) and ATOMIC (Sap et al., 2019). Thus, we aim to incorporate the external commonsense knowledge, i.e., **intent** and **sentiment**, into the learning process to generate better event representations. Specifically, we propose a simple and effective model to jointly embed events, intents and emotions into the same vector space. A neural tensor network is used to learn baseline event embeddings, and we define a corresponding loss function to incorporate intent and sentiment information.

Extensive experiments show that incorporating external commonsense knowledge brings promising improvements to event embeddings, achieving 78% and 200% improvements on hard similarity small and big dataset, respectively. With better embeddings, we can achieve superior performances on script event prediction and stock market prediction compared to state-of-the-art baseline methods.

## 2 Commonsense Knowledge Enhanced Event Representations

The joint embedding framework is shown in Figure 2. We begin by introducing the baseline event embedding learning model, which serves as the basis of the proposed framework. Then, we show how to model intent and sentiment information. Subsequently, we describe the proposed joint model by integrating intent and sentiment into the original objective function to help learn high-quality event representations, and introduce the training details.

### 2.1 Low-Rank Tensors for Event Embedding

The goal of event embedding is to learn low-dimension dense vector representations for event tuples $E = (A, P, O)$, where $P$ is the action or predicate, $A$ is the actor or subject and $O$ is the object on which the action is performed. Event

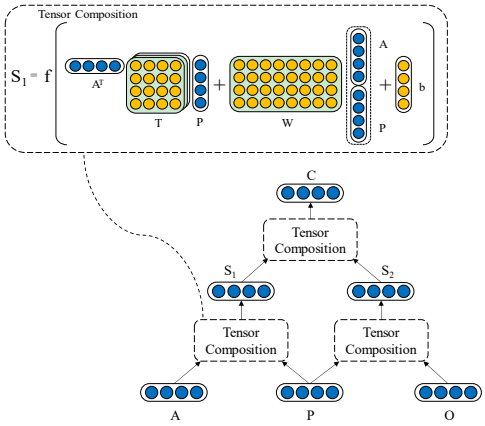


Figure 3: Baseline event-embedding model.

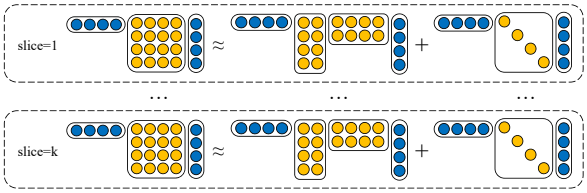


Figure 4: An illustration of low-rank neural tensor network for learning event embeddings.

embedding models compound vector representations over its predicate and arguments representations. The challenge is that the composition models should be effective for learning the interactions between the predicate and the argument. Simple additive transformations are incompetent.

We follow Ding et al. (2015) modelling such informative interactions through tensor composition. The architecture of neural tensor network (NTN) for learning event embeddings is shown in Figure 3, where the bilinear tensors are used to explicitly model the relationship between the actor and the action, and that between the object and the action.

The inputs of NTN are the word embeddings of $A$, $P$ and $O$, and the outputs are event embeddings. We initialized our word representations using publicly available $d$-dimensional ($d = 100$) GloVe vectors (Pennington et al., 2014). As most event arguments consist of several words, we represent the actor, action and object as the average of their word embeddings, respectively.

From Figure 3, $S_1 \in \mathbb{R}^d$ is computed by:

$$S_1 = f\left(A^T T_1^{[1:k]} P + W \begin{bmatrix} A \\ P \end{bmatrix} + b\right)$$
$$g(S_1) = g(A, P) = U^T S_1 \quad (1)$$

where $T_1^{[1:k]} \in \mathbb{R}^{d \times d \times k}$ is a tensor, which is a set of $k$ matrices, each with $d \times d$ dimensions. The bilinear tensor product $A^T T_1^{[1:k]} P$ is a vector $r \in \mathbb{R}^k$, where each entry is computed by one slice of the tensor ($r_i = A^T T_1^{[i]} P, i = 1, \cdots, k$). The other parameters are a standard feed-forward neural network, where $W \in \mathbb{R}^{k \times 2d}$ is the weight matrix, $b \in \mathbb{R}^k$ is the bias vector, $U \in \mathbb{R}^k$ is a hyper-parameter and $f = tanh$ is a standard nonlinearity applied element-wise. $S_2$ and $C$ in Figure 3 are computed in the same way as $S_1$.

One problem with tensors is *curse of dimensionality*, which limits the wide application of tensors in many areas. It is therefore essential to approximate tensors of higher order in a compressed scheme, for example, a low-rank tensor decomposition. To decrease the number of parameters in standard neural tensor network, we make low-rank approximation that represents each matrix by two low-rank matrices plus diagonal, as illustrated in Figure 4. Formally, the parameter of the $i$-th slice is $T_{appr}^{[i]} = T^{[i_1]} \times T^{[i_2]} + diag(t^{[i]})$, where $T^{[i_1]} \in \mathbb{R}^{d \times n}$, $T^{[i_2]} \in \mathbb{R}^{n \times d}$, $t^{[i]} \in \mathbb{R}^d$, $n$ is a hyper-parameter, which is used for adjusting the degree of tensor decomposition. The output of neural tensor layer is formalized as follows.

$$S_1 = f\left(A^T [T_{appr}]_1^{[1:k]} P + W \begin{bmatrix} A \\ P \end{bmatrix} + b\right), \quad (2)$$

where $[T_{appr}]_1^{[1:k]}$ is the low-rank tensor that defines multiple low-rank bilinear layers. $k$ is the slice number of neural tensor network which is also equal to the output length of $S_1$.

We assume that event tuples in the training data should be scored higher than corrupted tuples, in which one of the event arguments is replaced with a random argument. Formally, the corrupted event tuple is $E^r = (A^r, P, O)$, which is derived by replacing each word in $A$ with a random word $w^r$ in our dictionary $\mathcal{D}$ (which contains all the words in the training data) to obtain a corrupted counterpart $A^r$. We calculate the *margin loss* of the two event tuples as:

$$\mathcal{L}_\mathcal{E} = loss(E, E^r) = \max(0, 1 - g(E) + g(E^r)) + \lambda \|\Phi\|_2^2, \quad (3)$$

where $\Phi = (T_1, T_2, T_3, W, b)$ is the set of model parameters. The standard $L_2$ regularization is used, for which the weight $\lambda$ is set as 0.0001. The

algorithm goes over the training set for multiple iterations. For each training instance, if the loss $loss(E, E^r) = \max(0, 1-g(E)+g(E^r))$ is equal to zero, the online training algorithm continues to process the next event tuple. Otherwise, the parameters are updated to minimize the loss using back-propagation (Rumelhart et al., 1985).

### 2.2 Intent Embedding

Intent embedding refers to encoding the event participants' intents into event vectors, which is mainly used to explain why the actor performed the action. For example, given two events "*PersonX threw basketball*" and "*PersonX threw bomb*", there are only subtle differences in their surface realizations, however, the intents are totally different. "*PersonX threw basketball*" is just for fun, while "*PersonX threw bomb*" could be a terrorist attack. With the intents, we can easily distinguish these superficial similar events.

One challenge for incorporating intents into event embeddings is that we should have a large-scale labeled dataset, which annotated the event and its actor's intents. Recently, Rashkin et al. (2018) and Sap et al. (2019) released such valuable commonsense knowledge dataset (ATOMIC), which consists of 25,000 event phrases covering a diverse range of daily-life events and situations. For example, given an event "*PersonX drinks coffee in the morning*", the dataset labels *PersonX*'s likely intent is "*PersonX wants to stay awake*".

We notice that the intents labeled in ATOMIC is a sentence. Hence, intent embedding is actually a sentence representation learning task. Among various neural networks for encoding sentences, bi-directional LSTMs (BiLSTM) (Hochreiter and Schmidhuber, 1997) have been a dominant method, giving state-of-the-art results in language modelling (Peters et al., 2018) and syntactic parsing (Dozat and Manning, 2016).

We use BiLSTM model to learn intent representations. BiLSTM consists of two LSTM components, which process the input in the forward left-to-right and the backward right-to-left directions, respectively. In each direction, the reading of input words is modelled as a recurrent process with a single hidden state. Given an initial value, the state changes its value recurrently, each time consuming an incoming word.

Take the forward LSTM component for example. Denoting the initial state as $\overrightarrow{\boldsymbol{h}}^0$, which is a model parameter, it reads the input word representations $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, and the recurrent state transition step for calculating $\overrightarrow{\boldsymbol{h}}^1, \ldots, \overrightarrow{\boldsymbol{h}}^{n+1}$ is defined as Graves and Schmidhuber (2005).

The backward LSTM component follows the same recurrent state transition process as the forward LSTM component. Starting from an initial state $\overleftarrow{\boldsymbol{h}}^{n+1}$, which is a model parameter, it reads the input $\boldsymbol{x}_n, \boldsymbol{x}_{n-1}, \ldots, \boldsymbol{x}_0$, changing its value to $\overleftarrow{\boldsymbol{h}}^n, \overleftarrow{\boldsymbol{h}}^{n-1}, \ldots, \overleftarrow{\boldsymbol{h}}^0$, respectively.

The BiLSTM model uses the concatenated value of $\overrightarrow{\boldsymbol{h}}^t$ and $\overleftarrow{\boldsymbol{h}}^t$ as the hidden vector for $w_t$:

$$\boldsymbol{h}^t = [\overrightarrow{\boldsymbol{h}}^t; \overleftarrow{\boldsymbol{h}}^t] \quad (4)$$

A single hidden vector representation $\boldsymbol{v}_i$ of the input intent can be obtained by concatenating the last hidden states of the two LSTMs:

$$\boldsymbol{v}_i = [\overrightarrow{\boldsymbol{h}}^{n+1}; \overleftarrow{\boldsymbol{h}}^0] \quad (5)$$

In the training process, we calculate the similarity between a given event vector $\boldsymbol{v}_e$ and its related intent vector $\boldsymbol{v}_i$. For effectively training the model, we devise a ranking type loss function as follows:

$$\mathcal{L}_{\mathcal{I}} = max(0, 1 - cosine(\boldsymbol{v}_e, \boldsymbol{v}_i) + cosine(\boldsymbol{v}_e, \boldsymbol{v}_i')) \quad (6)$$

where $\boldsymbol{v}_i'$ is the incorrect intent for $\boldsymbol{v}_e$, which is randomly selected from the annotated dataset.

### 2.3 Sentiment Embedding

Sentiment embedding refers to encoding the event participants' emotions into event vectors, which is mainly used to explain how does the actor feel after the event. For example, given two events "*PersonX broke record*" and "*PersonX broke vase*", there are only subtle differences in their surface realizations, however, the emotions of *PersonX* are totally different. After "*PersonX broke record*", *PersonX* may be feel happy, while after "*PersonX broke vase*", *PersonX* could be feel sad. With the emotions, we can also effectively distinguish these superficial similar events.

We also use ATOMIC (Sap et al., 2019) as the event sentiment labeled dataset. In this dataset, the sentiment of the event is labeled as words. For example, the sentiment of "*PersonX broke vase*" is labeled as "(*sad, be regretful, feel sorry, afraid*)". We use SenticNet (Cambria et al., 2018) to normalize these emotion words ($W = \{w_1, w_2, \ldots, w_n\}$) as the positive (labeled as 1) or

the negative (labeled as -1) sentiment. The sentiment polarity of the event $P_e$ is dependent on the polarity of the labeled emotion words $P_W$: $P_e = 1$, if $\sum_i P_{w_i} > 0$, or $P_e = -1$, if $\sum_i P_{w_i} < 0$. We use the softmax binary classifier to learn sentiment enhanced event embeddings. The input of the classifier is event embeddings, and the output is its sentiment polarity (positive or negative). The model is trained in a supervised manner by minimizing the cross entropy error of the sentiment classification, whose loss function is given below.

$$\mathcal{L}_{\mathcal{S}} = -\sum_{x_e \in C} \sum_{l \in L} p_l^g(x_e) \cdot log(p_l(x_e)) \quad (7)$$

where $C$ means all training instances, $L$ is the collection of sentiment categories, $x_e$ means an event vector, $p_l(x_e)$ is the probability of predicting $x_e$ as class $l$, $p_l^g(x_e)$ indicates whether class $l$ is the correct sentiment category, whose value is 1 or -1.

### 2.4 Joint Event, Intent and Sentiment Embedding

Given a training event corpus with annotated intents and emotions, our model jointly minimizes a linear combination of the loss functions on events, intents and sentiment:

$$\mathcal{L} = \alpha\mathcal{L}_{\mathcal{E}} + \beta\mathcal{L}_{\mathcal{I}} + \gamma\mathcal{L}_{\mathcal{S}} \quad (8)$$

where $\alpha, \beta, \gamma \in [0, 1]$ are model parameters to weight the three loss functions.

We use the New York Times Gigaword Corpus (LDC2007T07) for pre-training event embeddings. Event triples are extracted based on the Open Information Extraction technology (Schmitz et al., 2012). We initialize the word embedding layer with 100 dimensional pre-trained GloVe vectors (Pennington et al., 2014), and fine-tune initialized word vectors during our model training. We use Adagrad (Duchi et al., 2011) for optimizing the parameters with initial learning rate 0.001 and batch size 128.

## 3 Experiments

We compare the performance of intent and sentiment powered event embedding model with state-of-the-art baselines on three tasks: event similarity, script event prediction and stock prediction.

### 3.1 Baselines

We compare the performance of our approach against a variety of event embedding models developed in recent years. These models can be categorized into three groups:

- **Averaging Baseline (Avg)** This represents each event as the average of the constituent word vectors using pre-trained GloVe embeddings (Pennington et al., 2014).
- **Compositional Neural Network (Comp. NN)** The event representation in this model is computed by feeding the concatenation of the subject, predicate, and object embedding into a two layer neural network (Modi and Titov, 2013; Modi, 2016; Granroth-Wilding and Clark, 2016).
- **Element-wise Multiplicative Composition (EM Comp.)** This method simply concatenates the element-wise multiplications between the verb and its subject/object.
- **Neural Tensor Network** This line of work use tensors to learn the interactions between the predicate and its subject/object (Ding et al., 2015; Weber et al., 2018). According to the different usage of tensors, we have three baseline methods: **Role Factor Tensor** (Weber et al., 2018) which represents the predicate as a tensor, **Predicate Tensor** (Weber et al., 2018) which uses two tensors learning the interactions between the predicate and its subject, and the predicate and its object, respectively, **NTN** (Ding et al., 2015), which we used as the baseline event embedding model in this paper, and **KGEB** (Ding et al., 2016), which incorporates knowledge graph information in NTN.

### 3.2 Event Similarity Evaluation

#### 3.2.1 Hard Similarity Task

We first follow Weber et al. (2018) evaluating our proposed approach on the hard similarity task. The goal of this task is that similar events should be close to each other in the same vector space, while dissimilar events should be far away with each other. To this end, Weber et al. (2018) created two types of event pairs, one with events that should be close to each other but have very little lexical overlap (e.g., police catch robber / authorities apprehend suspect), and another with events that should be farther apart but have high overlap (e.g., police catch robber / police catch disease).

| Method | Hard Similarity (*Accuracy* %) Small Dataset | Big Dataset | Transitive Sentence Similarity ($\rho$) |
|---|---|---|---|
| Avg | 5.2 | 13.7 | 0.67 |
| Comp. NN | 33.0 | 18.9 | 0.63 |
| EM Comp. | 33.9 | 18.7 | 0.57 |
| Role Factor Tensor | 43.5 | 20.7 | 0.64 |
| Predicate Tensor | 41.0 | 25.6 | 0.63 |
| KGEB | 52.6 | 49.8 | 0.61 |
| NTN | 40.0 | 37.0 | 0.60 |
| NTN+Int | 65.2 | 58.1 | 0.67 |
| NTN+Senti | 54.8 | 52.2 | 0.61 |
| NTN+Int+Senti | **77.4** | **62.8** | **0.74** |

Table 1: Experimental results on hard similarity dataset and transitive sentence similarity dataset. The small dataset (230 event pairs) of hard similarity task from Weber et al. (2018), and the big dataset (2,000 event pairs) is annotated by us. The best results are in bold.

The labeled dataset contains 230 event pairs (115 pairs each of similar and dissimilar types). Three different annotators were asked to give the similarity/dissimilarity rankings, of which only those the annotators agreed upon completely were kept. For each event representation learning method, we obtain the cosine similarity score of the pairs, and report the fraction of cases where the similar pair receives a higher cosine value than the dissimilar pair (we use *Accuracy* $\in [0, 1]$ denoting it). To evaluate the robustness of our approach, we extend this dataset to 1,000 event pairs (similar and dissimilar events each account for 50%), and we will release this dataset to the public.

#### 3.2.2 Transitive Sentence Similarity

Except for the hard similarity task, we also evaluate our approach on the transitive sentence similarity dataset (Kartsaklis and Sadrzadeh, 2014), which contains 108 pairs of transitive sentences: short phrases containing a single subject, object and verb (e.g., agent sell property). It also has another dataset which consists of 200 sentence pairs. In this dataset, the sentences to be compared are constructed using the same subject and object and semantically correlated verbs, such as 'spell' and 'write'; for example, 'pupils write letters' is compared with 'pupils spell letters'. As this dataset is not suitable for our task, we only evaluate our approach and baselines on 108 sentence pairs.

Every pair is annotated by a human with a similarity score from 1 to 7. For example, pairs such as (design, reduce, amount) and (company, cut, cost) are annotated with a high similarity score, while pairs such as (wife, pour, tea) and (worker, join, party) are given low similarity scores. Since each pair has several annotations, we use the average annotator score as the gold score[2]. To evaluate the cosine similarity given by each model and the annotated similarity score, we use the Spearman's correlation ($\rho \in [-1, 1]$).

#### 3.2.3 Results

Experimental results of hard similarity and transitive sentence similarity are shown in Table 1. We find that:

(1) Simple averaging achieved competitive performance in the task of transitive sentence similarity, while performed very badly in the task of hard similarity. This is mainly because hard similarity dataset is specially created for evaluating the event pairs that should be close to each other but have little lexical overlap and that should be farther apart but have high lexical overlap. Obviously, on such dataset, simply averaging word vectors which is incapable of capturing the semantic interactions between event arguments, cannot achieve a sound performance.

(2) Tensor-based compositional methods (NTN, KGEB, Role Factor Tensor and Predicate Tensor) outperformed parameterized additive models (Comp. NN and EM Comp.), which shows that tensor is capable of learning the semantic composition of event arguments.

(3) Our commonsense knowledge enhanced event representation learning approach outperformed all baseline methods across all datasets (achieving 78% and 200% improvements on hard similarity small and big dataset, respectively, compared to previous SOTA method), which indicates that commonsense knowledge is useful for distinguishing distinct events.

[2]To directly compare with baseline methods (Weber et al., 2018), this paper compares with averaged annotator scores, other than comparing with every annotator scores.

| Event1 | Event 2 | oScore | mScore | Event1 | Event 2 | oScore | mScore |
|---|---|---|---|---|---|---|---|
| man clears test | he passed exam | -0.08 | 0.40 | man passed car | man passed exam | 0.81 | 0.12 |
| he grind corn | cook chops beans | 0.31 | 0.81 | he grind corn | he grind teeth | 0.89 | 0.36 |
| he made meal | chef cooked pasta | 0.51 | 0.85 | chef cooked pasta | chef cooked books | 0.89 | 0.45 |
| farmer load truck | person packs car | 0.58 | 0.83 | farmer load truck | farmer load gun | 0.93 | 0.55 |
| player scored goal | she carried team | 0.19 | 0.44 | she carried bread | she carried team | 0.59 | 0.09 |

Table 2: Case study of the cosine similarity score changes with incorporating the intent and sentiment. oScore is the original cosine similarity score without intent and sentiment, and mScore is the modified cosine similarity score with intent and sentiment.

| **Methods** | **Acc (%)** |
|---|---|
| SGNN | 52.45 |
| SGNN+Int | **53.93** |
| SGNN+Senti | 53.57 |
| SGNN+Int+Senti | 53.88 |
| SGNN+PairLSTM | 52.71 |
| SGNN+EventComp | 54.15 |
| SGNN+EventComp+PairLSTM | 54.93 |
| SGNN+PairLSTM+Int+Senti | 54.14 |
| SGNN+EventComp+Int+Senti | 55.08 |
| SGNN+EventComp+PairLSTM+Int+Senti | **56.03** |

Table 3: Results of script event prediction on the test set. The improvement is significant at $p < 0.05$. Acc is short for Accuracy.

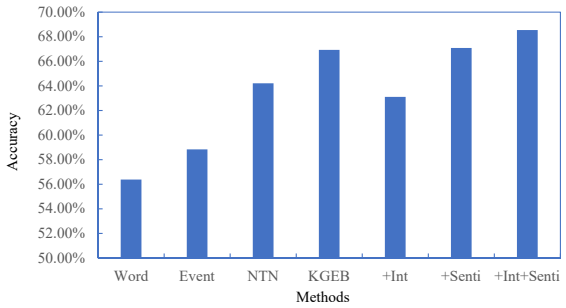


Figure 5: Experimental results on S&P 500 index prediction. "+Int" means that we encode the intent information into the original event embeddings.

#### 3.2.4 Case Study

To further analyse the effects of intents and emotions on the event representation learning, we present case studies in Table 2, which directly shows the changes of similarity scores before and after incorporating intent and sentiment. For example, the original similarity score of two events "*chef cooked pasta*" and "*chef cooked books*" is very high (0.89) as they have high lexical overlap. However, their intents differ greatly. The intent of "*chef cooked pasta*" is "*to hope his customer enjoying the delicious food*", while the intent of "*chef cooked books*" is "*to falsify their financial statements*". Enhanced with the intents, the similarity score of the above two events dramatically drops to 0.45. For another example, as the event pair "*man clears test*" and "*he passed exam*" share the same sentiment polarity, their similarity score is boosted from -0.08 to 0.40.

### 3.3 Script Event Prediction

Event is a kind of important real-world knowledge. Learning effective event representations can be benefit for numerous applications. Script event prediction (Chambers and Jurafsky, 2008) is a challenging event-based commonsense reasoning task, which is defined as giving an existing event context, one needs to choose the most reasonable subsequent event from a candidate list.

Following Li et al. (2018a), we evaluate on the standard multiple choice narrative cloze (MCNC) dataset (Granroth-Wilding and Clark, 2016). As SGNN proposed by Li et al. (2018a) achieved state-of-the-art performances for this task, we use the framework of SGNN, and only replace their input event embeddings with our intent and sentiment-enhanced event embeddings.

Wang et al. (2017) and Li et al. (2018a) showed that script event prediction is a challenging problem, and even 1% of accuracy improvement is very difficult. Experimental results shown in Table 3 demonstrate that we can achieve more than 1.5% improvements in single model comparison and more than 1.4% improvements in multi-model integration comparison, just by replacing the input embeddings, which confirms that better event understanding can lead to better inference results. An interesting result is that the event embeddings only incorporated with intents achieved the best result against other baselines. This confirms that capturing people's intents is helpful to infer their next plan. In addition, we notice that the event embeddings only incorporated with sentiment also achieve better performance than SGNN. This is mainly because the emotional consistency does also contribute to predicate the subsequent event.

### 3.4 Stock Market Prediction

It has been shown that news events influence the trends of stock price movements (Luss and d'Aspremont, 2012). As news events affect human decisions and the volatility of stock prices is influenced by human trading, it is reasonable to say that events can influence the stock market.

In this section, we compare with several event-driven stock market prediction baseline methods: (1) **Word**, Luss and d'Aspremont (2012) use bag-of-words represent news events for stock prediction; (2) **Event**, Ding et al. (2014) represent events by subject-predicate-object triples for stock prediction; (3) **NTN**, Ding et al. (2015) learn continues event vectors for stock prediction; (4) **KGEB**, Ding et al. (2016) incorporate knowledge graph into event vectors for stock prediction.

Experimental results are shown in Figure 5. We find that knowledge-driven event embedding is a competitive baseline method, which incorporates world knowledge to improve the performances of event embeddings on the stock prediction. Sentiment is often discussed in predicting stock market, as positive or negative news can affect people's trading decision, which in turn influences the movement of stock market. In this study, we empirically show that event emotions are effective for improving the performance of stock prediction (+2.4%).

## 4 Related Work

Recent advances in computing power and NLP technology enables more accurate models of events with structures. Using open information extraction to obtain structured events representations, we find that the actor and object of events can be better captured (Ding et al., 2014). For example, a structured representation of the event above can be (Actor = *Microsoft*, Action = *sues*, Object = *Barnes & Noble*). They report improvements on stock market prediction using their structured representation instead of words as features.

One disadvantage of structured representations of events is that they lead to increased sparsity, which potentially limits the predictive power. Ding et al. (2015) propose to address this issue by representing structured events using *event embeddings*, which are dense vectors. The goal of event representation learning is that similar events should be embedded close to each other in the same vector space, and distinct events should be farther from each other.

Previous work investigated compositional models for event embeddings. Granroth-Wilding and Clark (2016) concatenate predicate and argument embeddings and feed them to a neural network to generate an event embedding. Event embeddings are further concatenated and fed through another neural network to predict the coherence between the events. Modi (2016) encodes a set of events in a similar way and use that to incrementally predict the next event – first the argument, then the predicate and then next argument. Pichotta and Mooney (2016) treat event prediction as a sequence to sequence problem and use RNN based models conditioned on event sequences in order to predict the next event. These three works all model narrative chains, that is, event sequences in which a single entity (the protagonist) participates in every event. Hu et al. (2017) also apply an RNN approach, applying a new hierarchical LSTM model in order to predict events by generating descriptive word sequences. This line of work combines the words in these phrases by the passing the concatenation or addition of their word embeddings to a parameterized function that maps the summed vector into event embedding space. The additive nature of these models makes it difficult to model subtle differences in an event's surface form.

To address this issue, Ding et al. (2015), and Weber et al. (2018) propose tensor-based composition models, which combine the subject, predicate and object to produce the final event representation. The models capture multiplicative interactions between these elements and are thus able to make large shifts in event semantics with only small changes to the arguments.

However, previous work mainly focuses on the nature of the event and lose sight of external commonsense knowledge, such as the intent and sentiment of event participants. This paper proposes to encode intent and sentiment into event embeddings, such that we can obtain a kind of more powerful event representations.

## 5 Conclusion

Understanding events requires effective representations that contain commonsense knowledge. High-quality event representations are valuable for many NLP downstream applications. This paper proposed a simple and effective framework to incorporate commonsense knowledge into the learn-

ing process of event embeddings. Experimental results on event similarity, script event prediction and stock prediction showed that commonsense knowledge enhanced event embeddings can improve the quality of event representations and benefit the downstream applications.

## Acknowledgments

We thank the anonymous reviewers for their constructive comments, and gratefully acknowledge the support of the National Key Research and Development Program of China (SQ2018AAA010010), the National Key Research and Development Program of China (2018YFB1005103), the National Natural Science Foundation of China (NSFC) via Grant 61702137.

## References

Erik Cambria, Soujanya Poria, Devamanyu Hazarika, and Kenneth Kwok. 2018. Senticnet 5: discovering conceptual primitives for sentiment analysis by means of context embeddings. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*.

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of ACL-08: HLT*, pages 789–797. Association for Computational Linguistics.

Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. 2014. Using structured events to predict stock price movement: An empirical investigation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1415–1425, Doha, Qatar. Association for Computational Linguistics.

Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. 2015. Deep learning for event-driven stock prediction. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 2327–2333.

Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. 2016. Knowledge-driven event embedding for stock prediction. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2133–2142.

Timothy Dozat and Christopher D Manning. 2016. Deep biaffine attention for neural dependency parsing. *arXiv preprint arXiv:1611.01734*.

John C Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Mark Granroth-Wilding and Stephen Christopher Clark. 2016. What happens next? event prediction using a compositional neural network model. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 2727–2733.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5-6):602–610.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Linmei Hu, Juanzi Li, Liqiang Nie, Xiao-Li Li, and Chao Shao. 2017. What happens next? future subevent prediction using contextual hierarchical lstm. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3450–3456.

Dimitri Kartsaklis and Mehrnoosh Sadrzadeh. 2014. A study of entanglement in a categorical framework of natural language. *Electronic Proceedings in Theoretical Computer Science*, 172.

Zhongyang Li, Xiao Ding, and Ting Liu. 2018a. Constructing narrative event evolutionary graph for script event prediction. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden.*, pages 4201–4207.

Zhongyang Li, Xiao Ding, and Ting Liu. 2018b. Generating reasonable and diversified story ending using sequence to sequence model with adversarial training. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1033–1043, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Zhongyang Li, Xiao Ding, and Ting Liu. 2019. Story ending prediction by transferable bert. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 1800–1806.

Ronny Luss and Alexandre d'Aspremont. 2012. Predicting abnormal returns from news using text classification. *Quantitative Finance*, (doi:10.1080/14697688.2012.672762):1–14.

Ashutosh Modi. 2016. Event embeddings for semantic script modeling. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 75–83.

Ashutosh Modi and Ivan Titov. 2013. Learning semantic script knowledge with event embeddings. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Workshop Track Proceedings*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 2227–2237.

Karl Pichotta and Raymond J Mooney. 2016. Learning statistical scripts with lstm recurrent neural networks. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 2800–2806.

Hannah Rashkin, Maarten Sap, Emily Allaway, Noah A. Smith, and Yejin Choi. 2018. Event2mind: Commonsense inference on events, intents, and reactions. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 463–473. Association for Computational Linguistics.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1985. Learning internal representations by error propagation. Technical report, DTIC Document.

Maarten Sap, Ronan LeBras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A Smith, and Yejin Choi. 2019. Atomic: An atlas of machine commonsense for if-then reasoning. pages 3027–3035.

Michael Schmitz, Robert Bart, Stephen Soderland, Oren Etzioni, et al. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 523–534. Association for Computational Linguistics.

Zhongqing Wang, Yue Zhang, and Ching-Yun Chang. 2017. Integrating order information and event relation for script event prediction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 57–67. Association for Computational Linguistics.

Noah Weber, Niranjan Balasubramanian, and Nathanael Chambers. 2018. Event representations with tensor-based compositions. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*.

# Learning to Discriminate Perturbations for Blocking Adversarial Attacks in Text Classification

**Yichao Zhou***, **Jyun-Yu Jiang***, **Kai-Wei Chang and Wei Wang**
Computer Science Department
University of California, Los Angeles
{yz, jyunyu, kwchang, weiwang}@cs.ucla.edu

## Abstract

Adversarial attacks against machine learning models have threatened various real-world applications such as spam filtering and sentiment analysis. In this paper, we propose a novel framework, learning to discriminate perturbations (DISP), to identify and adjust malicious perturbations, thereby blocking adversarial attacks for text classification models. To identify adversarial attacks, a perturbation discriminator validates how likely a token in the text is perturbed and provides a set of potential perturbations. For each potential perturbation, an embedding estimator learns to restore the embedding of the original word based on the context and a replacement token is chosen based on approximate $k$NN search. DISP can block adversarial attacks for any NLP model without modifying the model structure or training procedure. Extensive experiments on two benchmark datasets demonstrate that DISP significantly outperforms baseline methods in blocking adversarial attacks for text classification. In addition, in-depth analysis shows the robustness of DISP across different situations.

## 1 Introduction

Deep learning techniques (Goodfellow et al., 2016) have achieved enormous success in many fields, such as computer vision and NLP. However, complex deep learning models are often sensitive and vulnerable to a tiny modification. In other words, malicious attackers can destroy the models by adding a few inconspicuous perturbations into input data, such as masking images with unrecognizable filters and making low-key modifications for texts. Therefore, developing techniques to equip models against adversarial attacks becomes a prominent research problem.

Existing studies on adversarial attacks can be classified into two groups, generation of adversarial examples and defense against adversarial attacks (Yuan et al., 2019). In the field of NLP, most of the existing studies focus on the former. For example, Ebrahimi et al. (2017); Alzantot et al. (2018) replace a word with synonyms or similar words while Gao et al. (2018); Liang et al. (2017); Ebrahimi et al. (2017) conduct character-level manipulations to fool the models. Moreover, it is not straightforward to adapt existing approaches for blocking adversarial attacks, such as data augmentation (Krizhevsky et al., 2012; Ribeiro et al., 2018; Ren et al., 2019) and adversarial training (Goodfellow et al., 2015; Iyyer et al., 2018; Marzinotto et al., 2019; Cheng et al., 2019; Zhu et al., 2019), to NLP applications. Hence, the defense against adversarial attacks in NLP remains a challenging and unsolved problem.

Recognizing and removing the inconspicuous perturbations are the core of defense against adversarial attacks. For instance, in computer vision, denoising auto-encoders (Warde-Farley and Bengio, 2017; Gu and Rigazio, 2015) are applied to remove the noises introduced by perturbations; Prakash et al. (2018) manipulate the images to make the trained models more robust to the perturbations; Samangouei et al. (2018) apply generative adversarial networks to generate perturbation-free images. However, all of these approaches cannot straightforwardly apply to the NLP tasks for the following two reasons. First, images consist of continuous pixels while texts are discrete tokens. As a result, a token can be replaced with another semantically similar token that drops the performance, so perturbations with natural looks cannot be easily recognized compared to previous approaches that capture unusual differences between the intensities of neighboring pixels. Second, sentences consist of words with an enormous

*Equal contribution. Listing order is random.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4904–4913, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

vocabulary size, so it is intractable to enumerate all of the possible sentences. Therefore, existing defense approaches in computer vision that rely on pixel intensities cannot be directly used for the NLP tasks.

After recognizing the perturbed tokens, the naïve way to eliminate the perturbations for blocking adversarial attacks is to remove these perturbed tokens. However, removing words from sentences results in fractured sentences, causing the performance of NLP models to degrade. Therefore, it is essential to recover the removed tokens. Nevertheless, training a satisfactory language model requires myriad and diverse training data, which is often unavailable. An inaccurate language model that incoherently patches missing tokens can further worsen the prediction performance. To tackle this difficult challenge, we propose to recover the tokens from discriminated perturbations by a masked language model objective with contextualized language modeling.

In this paper, we propose *Learning to Discriminate Perturbations* (DISP), as a framework for blocking adversarial attacks in NLP. More specifically, we aim to defend the model against adversarial attacks without modifying the model structure and the training procedure. DISP consists of three components, perturbation discriminator, embedding estimator, and hierarchical navigable small world graphs. Given a perturbed testing data, the perturbation discriminator first identifies a set of perturbed tokens. For each perturbed token, the embedding estimator optimized with a corpus of token embeddings infers an embedding vector to represent its semantics. Finally, we conduct an efficient $k$NN search over a hierarchical taxonomy to translate each of the embedding vectors into appropriate token to replace the associated perturbed word. We summarize our contributions in the following.

- To the best of our knowledge, this paper is the first work for blocking adversarial attacks in NLP without retraining the model.
- We propose a novel framework, DISP, which is effective and significantly outperforms other baseline methods in defense against adversarial attacks on two benchmark datasets.
- Comprehensive experiments have been conducted to demonstrate the improvements of DISP. In addition, we will release our implementations and the datasets to provide a testbed and facilitate future research in this direction.

## 2 Related Work

Adversarial examples crafted by malicious attackers expose the vulnerability of deep neural networks when they are applied to down-streaming tasks, such as image recognition, speech processing, and text classifications (Wang et al., 2019; Goodfellow et al., 2015; Nguyen et al., 2015; Moosavi-Dezfooli et al., 2017).

For adversarial attacks, white-box attacks have full access to the target model while black-box attacks can only explore the models by observing the outputs with limited trials. Ebrahimi et al. (2017) propose a gradient-based white-box model to attack character-level classifiers via an atomic flip operation. Small character-level transformations, such as swap, deletion, and insertion, are applied on critical tokens identified with a scoring strategy (Gao et al., 2018) or gradient-based computation (Liang et al., 2017). Samanta and Mehta (2017); Alzantot et al. (2018) replace words with semantically and syntactically similar adversarial examples.

However, limited efforts have been done on adversarial defense in the NLP fields. Texts as discrete data are sensitive to the perturbations and cannot transplant most of the defense techniques from the image processing domain such as Gaussian denoising with autoencoders (Meng and Chen, 2017; Gu and Rigazio, 2014). Adversarial training is the prevailing counter-measure to build a robust model (Goodfellow et al., 2015; Iyyer et al., 2018; Marzinotto et al., 2019; Cheng et al., 2019; Zhu et al., 2019) by mixing adversarial examples with the original ones during training the model. However, these adversarial examples can be detected and deactivated by a genetic algorithm (Alzantot et al., 2018). This method also requires retraining, which can be time and cost consuming for large-scale models.

Spelling correction (Mays et al., 1991; Islam and Inkpen, 2009) and grammar error correction (Sakaguchi et al., 2017) are useful tools which can block editorial adversarial attacks, such as swap and insertion. However, they cannot handle cases where word-level attacks that do not cause spelling and grammar errors. In our paper, we propose a general schema to block both word-level and character-level attacks.

## 3 DISP for Blocking Adversarial Attacks

In this section, we first formally define the goal of adversarial defense and then introduce the proposed framework DISP, learning to discriminate perturbations, for blocking adversarial attacks.

**Problem Statement.** Given an NLP model $F(X)$, where $X = \{t_1, \ldots, t_N\}$ is the input text of $N$ tokens while $t_i$ indicates the $i$-th token. A malicious attacker can add a few inconspicuous perturbations into the input text and generate an adversarial example $X_a$ so that $F(X) \neq F(X_a)$ with unsatisfactory prediction performance. For example, a perturbation can be an insertion, a deletion of a character in a token, a replacement of a token with its synonym. In this paper, we aim to block adversarial attacks for general text classification models. More specifically, we seek to preserve the model performances by recovering original input text and universally improve the robustness of any text classification model.

### 3.1 Framework Overview

Figure 1 illustrates the overall schema of the proposed framework. DISP consists of three components, (1) a perturbation discriminator, (2) an embedding estimator, and (3) a token embedding corpus with the corresponding small world graphs $G$. In the training phase, DISP constructs a corpus $D$ from the original corpus for training the perturbation discriminator so that it is capable of recognizing the perturbed tokens. The corpus of token embeddings $C$ is then applied to train the embedding estimator to recover the removed tokens after establishing the small world graphs $G$ of the embedding corpus. In the prediction phase, for each token in testing data, the perturbation discriminator predicts if the token is perturbed. For each potential perturbation that is potentially perturbed, the embedding estimator generates an approximate embedding vector and retrieve the token with the closest distance in the embedding space for token recovery. Finally, the recovered testing data can be applied for prediction. Note that the prediction model can be any NLP model.Moreover, DISP is a general framework for blocking adversarial attacks, so the model selection for the discriminator and estimator can also be flexible.

### 3.2 Perturbation Discrimination

**Perturbation Discriminator**. The perturbation discriminator plays an important role to classify whether a token $t_i$ in the input $X_a$ is perturbed based on its neighboring tokens. We adopt contextualized language modeling, such as BERT (Devlin et al., 2018), to derive $d$-dimension contextualized token representation $T_i^D$ for each token $t_i$ and then cascade it with a binary logistic regression classifier to predict if the token $t_i$ is perturbed or not. Figure 2 illustrates the perturbation discriminator based on a contextualized word encoder. The discriminator classifies a token $t_i$ into two classes $\{0, 1\}$ with logistic regression based on the contextual representation $T_i^D$ to indicate if the token is perturbed. More formally, for each token $t_i$, the discriminator predictions $r_i$ can then be derived as:

$$r_i = \underset{c}{\operatorname{argmax}}\, y_i^c = \underset{c}{\operatorname{argmax}} \left( \boldsymbol{w_c} \cdot T_i^D + b_c \right),$$

where $y_i^c$ is the logit for the class $c$; $\boldsymbol{w_c}$ and $b_c$ are the weights and the bias for the class $c$. Finally, the potential perturbations $R$ is the set of tokens with positive discriminator predictions $R = \{t_i \mid r_i = 1\}$.

### 3.3 Efficient Token-level Recovery with Embedding Estimator

After predicting the perturbations $R$, we need to correct these disorders to preserve the prediction performance. One of the most intuitive approaches to recover tokens with context is to exploit language models. However, language models require sufficient training data while the precision to exact tokens can be dispensable for rescuing prediction performance. Moreover, over-fitting limited training data can be harmful to the prediction quality. To resolve this problem, we assume that replacing the perturbed word with a word with similar meanings to the original word is sufficient for the downstream models to make the correct prediction. Based on the assumption, DISP first predicts the embeddings of the recovered tokens for the potential perturbations with an embedding estimator based on context tokens. The tokens can then be appropriately recovered by an efficient $k$-nearest neighbors ($k$NN) search in the embedding space of a token embedding corpus $C$.

**Embedding Estimator.** Similar to the perturbation discriminator, any regression model can be employed as an embedding estimator based on the proposed concept. Here we adopt the contextualized language modeling again as an example of the embedding estimator. For each token $t_i$, the

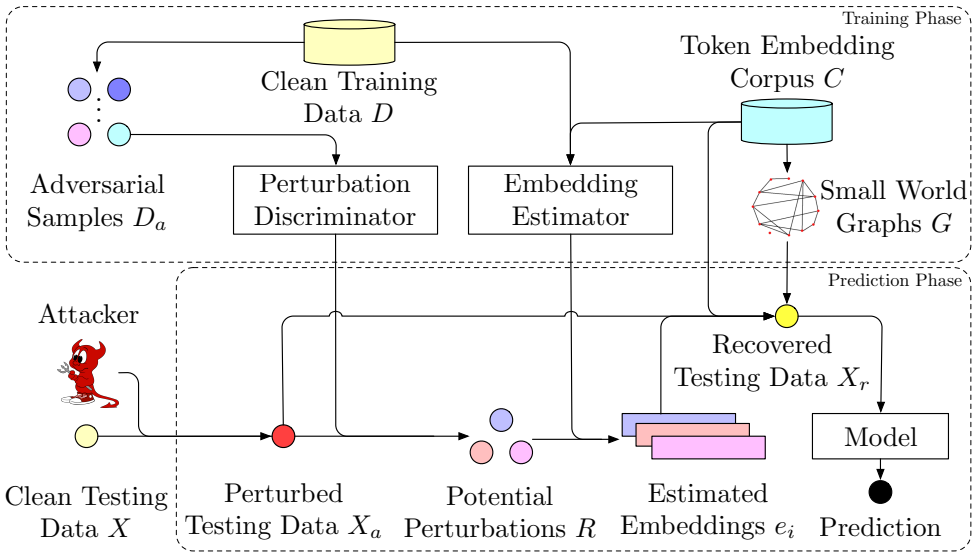


Figure 1: Schema of the proposed framework DISP.

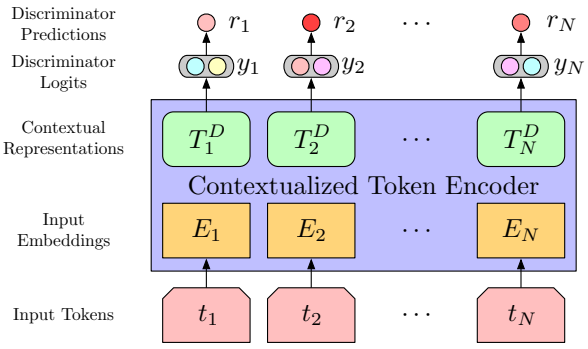


Figure 2: The illustration of the perturbation discriminator in DISP.

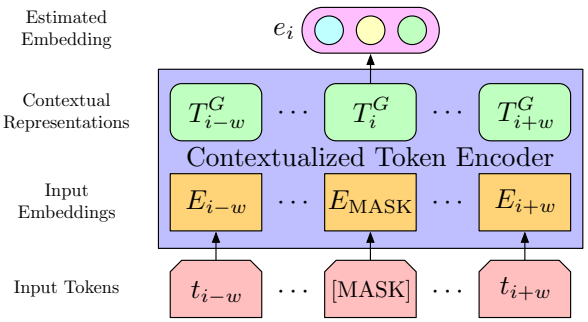


Figure 3: The illustration of the embedding estimator in DISP with a window size $2w+1$ for the token at the position $i$.

contextualized token embedding can be derived as a $d$-dimensional contextual representation vector $T_i^G$ to be features for estimating appropriate embeddings.

Figure 3 shows the embedding estimator based on BERT. For each potential perturbation $t_i \in R$, $2w$ neighboring tokens are selected as the context for estimating the appropriate embedding, where $w$ decides the window size. More precisely, a segment of tokens with a window size $2w+1$ from $t_{i-w}$ to $t_{i+w}$ is the input tokens for BERT, where $t_i$ is replaced with a `[MASK]` token as the perturbed position. Finally, for the target $t_i$, a weight matrix $W^G \in \mathbb{R}^{d \times k}$ projects the contextual representation $T_i^G$ to a $k$-dimensional estimated embedding $e_i$ as follows:

$$e_i = T_i^G W^G,$$

where the dimension size $k$ is required to be consistent with the embedding dimension in the token embedding corpus $C$.

**Efficient Token-level Recovery.** Finally, we recover the input sentence based on the predicted recover embeddings from the embedding estimator. Specifically, the input text $X$ needs to be recovered from the perturbed text $X_a$ by fixing token-level perturbations based on its approximate embeddings.

Given the token embedding corpus $C$, it is simple to transform an embedding to a token by finding the nearest neighbor token in the embedding space. However, a naïve $k$NN search query can take $O(kn)$ time complexity, where $n$ is the number of embeddings in $C$; $k$ is the embedding dimension. To accelerate the search process, we apply hierarchical navigable small world graphs (SWGs) (Malkov and Yashunin, 2018) for fast approximate $k$NN search. More precisely, em-

**Algorithm 1:** Efficient Token-level Recovery

**Input:** Perturbed text $X_a$, potential perturbations $R$, estimated embeddings $\{e_i\}$, small world graphs $G$, token embedding corpus $C$.
**Output:** Recovered text $X_r$.

```
X_r = X_a;
for t_i ∈ R do
    index = QuerySmallWorldGraph(G, e_i);
    z = C[index].token;
    Replace t_i in X_r with z;
return X_r;
```

beddings are transformed into a hierarchical set of SWGs based on the proximity between different embeddings. To conduct $k$NN searches, the property of degree distributions in SWGs significantly reduces the search space of each $k$NN query from $O(n)$ to $O(\log n)$ by navigating on the graphs, so a $k$NN query can be efficiently completed in $O(k \log n)$ time complexity. Finally, the recovered text $X_r$ can be obtained by replacing the perturbations $R$ in $X_a$ as shown in Algorithm 1.

### 3.4 Learning and Optimization

To learn a robust discriminator, we randomly sample adversarial examples from both character-level and word-level attacks in each training epoch. The loss function optimizes the cross-entropy between the labels and the probabilistic scores computed by the logits $y_i$ and the softmax function.

The learning process of embedding estimator is similar to masked language models. The major difference is that language models optimize the likelihood to generate the same original token while the embedding estimator minimizes the distance between the derived embedding and the original token embedding. To learn the embedding estimator, a size-$(2w+1)$ sliding window is applied to enumerate $(2w+1)$-gram training data for approximating embeddings with context tokens. For optimization, the embedding estimator is learned to minimize the mean square error (MSE) from the inferred embeddings to the original token embeddings.

To take advantage of hierarchical navigable SWGs for an efficient recovery, although a preprocess to construct SWGs $G$ is required, the preprocess can be fast. The established SWGs can also be serialized in advance. More precisely, the time complexity is $O(kn \log n)$ for one-time construction of reusable SWGs, where $n$ is the number of embeddings in the embedding corpus $C$.

| Dataset | Train | Test | Length | | |
|---|---|---|---|---|---|
| | | | Max. | Min. | Avg. |
| SST-2 | 67,349 | 1,821 | 56 | 1 | 19 |
| IMDb | 25,000 | 25,000 | 2,738 | 8 | 262 |

Table 1: The statistics of datasets.

| Attack Type | Example |
|---|---|
| No Attack | Old-form moviemaking at its best. |
| Insertion | Old-form moviemaking at its **beast**. |
| Deletion | Old-form moviemaking at its **be~~-~~t**. |
| Swap | Old-form moviemaking at its **bets**. |
| Random | Old-form moviemaking at its **aggrandize**. |
| Embed | Old-form moviemaking at its **way**. |

Table 2: Examples of each type of attack

## 4 Experiments

In this section, we conduct extensive experiments to evaluate the performance of DISP in improving model robustness.

### 4.1 Experimental Settings

**Experimental Datasets.** Experiments are conducted on two benchmark datasets: (1) Stanford Sentiment Treebank Binary (SST-2) (Socher et al., 2013) and (2) Internet Movie Database (IMDb) (Maas et al., 2011). SST-2 and IMDb are both sentiment classification datasets which involve binary labels annotating sentiment of sentences in movie reviews. Detailed statistics of two datasets are listed in Table 1.

**Attack Generation.** We consider three types of character-level attacks and two types of word-level attacks. The character-level attacks consist of *insertion*, *deletion*, and *swap*. *Insertion* and *deletion* attacks inject and remove a character, respectively, while a *swap* attack flips two adjacent characters. The word-level attacks include *random* and *embed*. A *random* attack randomly samples a word to replace the target word while a *embed* attack replaces the word with a word among the top-10 nearest words in the embedding space. The examples of each attack type are illustrated in Table 2. To obtain strong adversarial attack samples,w we consider to leverage oracle to identify the perturbations that cause prediction changes. Specifically, for each test sample we construct 50 adversarial examples by perturbing the test data. We sample one example in which model prediction changes after perturbing. If none of them can

| Dataset | Method | Metric | Character-level Attacks | | | Word-level Attacks | | Overall |
|---|---|---|---|---|---|---|---|---|
| | | | Insertion | Deletion | Swap | Random | Embed | Attacks |
| SST-2 | SC | Precision | 0.5087 | 0.4703 | 0.5044 | 0.1612 | 0.1484 | 0.3586 |
| | | Recall | 0.9369 | 0.8085 | 0.9151 | 0.1732 | 0.1617 | 0.5991 |
| | | F1 | 0.6594 | 0.5947 | 0.6504 | 0.1669 | 0.1548 | 0.4452 |
| | DISP | Precision | 0.9725 | 0.9065 | 0.9552 | 0.8407 | 0.4828 | 0.8315 |
| | | Recall | 0.8865 | 0.8760 | 0.8680 | 0.6504 | 0.5515 | 0.7665 |
| | | F1 | **0.9275** | **0.8910** | **0.9095** | **0.7334** | **0.5149** | **0.7952** |
| IMDb | SC | Precision | 0.0429 | 0.0369 | 0.0406 | 0.0084 | 0.0064 | 0.0270 |
| | | Recall | 0.9367 | 0.8052 | 0.8895 | 0.1790 | 0.1352 | 0.5891 |
| | | F1 | 0.0820 | 0.0706 | 0.0777 | 0.0161 | 0.0122 | 0.0517 |
| | DISP | Precision | 0.9150 | 0.8181 | 0.8860 | 0.5233 | 0.2024 | 0.6690 |
| | | Recall | 0.5068 | 0.4886 | 0.5000 | 0.3876 | 0.2063 | 0.4179 |
| | | F1 | **0.6523** | **0.6118** | **0.6392** | **0.4454** | **0.2044** | **0.5106** |

Table 3: Performance of SC and DISP on identifying perpetuated tokens.

change the prediction, the sample with the least confidence is selected.

**Base Model and Baselines.** We consider BERT (Devlin et al., 2018) as the base model as it achieves strong performance in these benchmarks. To evaluate the performance of DISP, we consider the following baseline methods: (1) Adversarial Data Augmentation (ADA) samples adversarial examples to increase the diversity of training data; (2) Adversarial Training (AT) samples different adversarial examples in each training epoch; (3) Spelling Correction (SC) is used as a baseline for discriminating perturbations and blocking character-level attacks. Note that ADA and AT require to re-train BERT with the augmented training data, while DISP and SC modify the input text and then exploit the original model for prediction. SC is also the only baseline for evaluating discriminator performance. In addition, we also try to ensemble DISP and SC (DISP+SC) by conducting DISP on the spelling corrected input.

**Evaluation Metrics.** We evaluate the performance of the perturbation discriminator by precision, recall and F1 scores, and evaluate the overall end-to-end performance by classification accuracy that the models recover.

**Implementation Details.** The model is implemented in PyTorch (Paszke et al., 2017). We set the initial learning and dropout parameter to be $2 \times 10^{-5}$ and 0.1. We use crawl-300d-2M word embeddings from *fastText* (Mikolov et al., 2018) to search similar words. The dimensions of word embedding $k$ and contextual representation $d$ are set as 300 and 768. $w$ is set as 2. We follow $\text{BERT}_{\text{BASE}}$ (Devlin et al., 2018) to set the numbers of layers (i.e., Transformer blocks) and self-attention heads as 12.

### 4.2 Experimental Results

**Performance on identifying perpetuated tokens.** Table 3 shows the performance of DISP and SC in discriminating perturbations. Compared to SC, DISP has an absolute improvement by 35% and 46% on SST-2 and IMDb in terms of F1-score, respectively. It also proves that the context information is essential when discriminating the perturbations. An interesting observation is that SC has high recall but low precision scores for character-level attacks because it is eager to correct misspellings while most of its corrections are not perturbations. Conversely, DISP has more balances of recall and precision scores since it is optimized to discriminate the perturbed tokens. For the word-level attacks, SC shows similar low performance on both *random* and *embed* attacks while DISP behaves much better. Moreover, DISP works better on the *random* attack because the embeddings of the original tokens tend to have noticeably greater Euclidean distances to randomly-picked tokens than the distances to other tokens.

**Defense Performance.** Table 4 reports the accuracy scores of all methods with different types of adversarial attacks on two datasets. Compared to the baseline BERT model, all of the methods alleviate the performance drops. All methods perform better on blocking character-level attacks than word-level attacks because word-level attacks eliminate more information. For the base-

| Dataset | Method | Attack -free | Character-level Attacks | | | Word-level Attacks | | Overall Attacks |
|---|---|---|---|---|---|---|---|---|
| | | | Insertion | Deletion | Swap | Random | Embed | |
| SST-2 | BERT | **0.9232** | 0.6498 | 0.6544 | 0.6774 | 0.5385 | 0.6556 | 0.6351 |
| | SC | 0.9174 | **0.9082** | 0.8186 | **0.8840** | 0.5993 | 0.7003 | 0.7821 |
| | ADA | 0.9174 | 0.8071 | 0.8071 | 0.8209 | 0.7394 | 0.7681 | 0.7885 |
| | AT | 0.9186 | 0.8186 | 0.8175 | 0.8025 | 0.6935 | 0.7646 | 0.7793 |
| | DISP | **0.9232** | 0.8278 | **0.8278** | 0.8301 | **0.7773** | **0.7784** | **0.8083** |
| | DISP+SC | 0.9197 | **0.9128** | **0.8681** | **0.9060** | **0.7784** | **0.7853** | **0.8501** |
| IMDb | BERT | **0.9431** | 0.8586 | 0.8599 | 0.8568 | 0.8468 | 0.8615 | 0.8567 |
| | SC | 0.9193 | 0.8834 | 0.8794 | 0.8825 | 0.8695 | 0.8753 | 0.8780 |
| | ADA | 0.9393 | 0.8766 | 0.8765 | 0.8754 | 0.8722 | 0.8755 | 0.8752 |
| | AT | 0.8998 | 0.8958 | 0.8822 | 0.8787 | 0.8886 | 0.8822 | 0.8855 |
| | DISP | 0.9378 | **0.9310** | **0.9297** | **0.9301** | **0.9281** | **0.9347** | **0.9307** |
| | DISP+SC | 0.9395 | **0.9316** | 0.8772 | **0.9313** | 0.8755 | 0.9292 | 0.9090 |

Table 4: The accuracy scores of methods with different adversarial attacks on two datasets.

lines, consistent with Table 3, SC performs the best for character-level attacks and the worst for word-level attacks. In contrast, ADA and AT are comparably more stable across different types of attacks. The differences between performance for character- and word-level attacks are less obvious in IMDb because documents in IMDb tend to be longer with more contexts to support the models. DISP works well to block all types of attacks. Compared with the best baseline models, DISP significantly improves the classification accuracy by 2.51% and 5.10% for SST-2 and IMDb, respectively. By ensembling SC and DISP, DISP+SC achieves better performance for blocking all types of attacks. However, the improvements are not consistent in IMDb. In particular, SC performs worse with lower discrimination accuracy and over-correcting the documents. In addition, DISP has a stable defense performance across different types of attacks on IMDb because richer context information in the documents benefits token recovery.

**Number of Attacks.** Figure 4 shows the classification accuracy of all methods over different numbers of attacks, i.e., perturbations, for different types of adversarial attacks. Without using a defense method, the performance of BERT dramatically decreases when the number of attacks increases. With defense approaches, the performance drops are alleveated. Moreover, the relations between the performance of methods are consistent across different perturbation numbers. DISP+SC consistently performs the best for all of the cases when DISP outperforms all of the single methods for most of the situations. These results demonstrate the robustness of the proposed approach.

| Method | Insertion | Delete | Swap |
|---|---|---|---|
| BERT | 0.6498 | 0.6544 | 0.6774 |
| $\text{DISP}_{\text{SST-2}}$ | 0.8278 | 0.8278 | 0.8301 |
| $\text{DISP}_{\text{IMDb}}$ | 0.8243 | 0.8197 | 0.8278 |
| **Method** | **Random** | **Embed** | **Overall** |
| BERT | 0.5385 | 0.6556 | 0.6351 |
| $\text{DISP}_{\text{SST-2}}$ | 0.7773 | 0.7784 | 0.8083 |
| $\text{DISP}_{\text{IMDb}}$ | 0.7623 | 0.7681 | 0.8005 |

Table 5: The accuracy of DISP over different types of attacks on the SST-2 dataset with the tokens recovered by the perturbation discriminator and the embedding estimator trained on the IMDb dataset for robust transfer defense. Note that $\text{DISP}_x$ indicates the framework is established on the dataset $x$.

**Robust Transfer Defense.** In practice, we may not have access to the original training corpus of a prediction model. In the following, we investigate if the perturbation discriminator can transfer across different corpora. We first train the discriminator and the estimator on IMDb denoted as $\text{DISP}_{\text{IMDb}}$ and then apply it to defend the prediction model on SST-2. Table 5 shows the experimental results of robust transfer defense. $\text{DISP}_{\text{IMDb}}$ achieves similar performance as the performance of $\text{DISP}_{\text{SST-2}}$ trained on the same training set. Hence, it shows that DISP can transfer the ability to recover perpetuated token across different sentiment copora.

**Case Study of Recovered Text.** Table 6 lists four

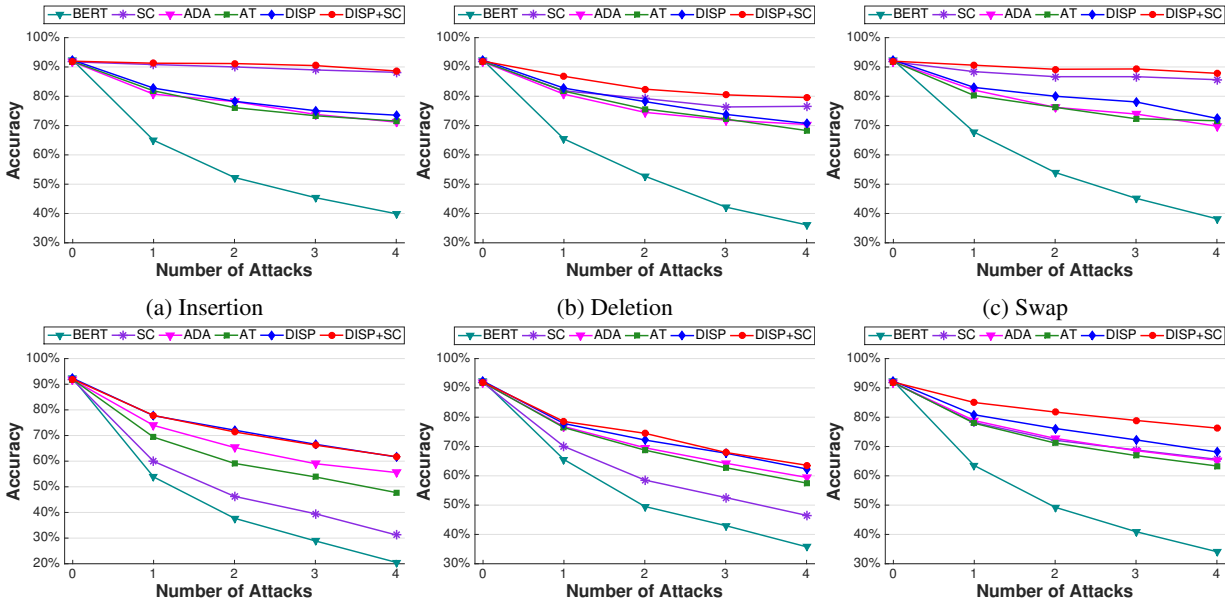


(a) Insertion (b) Deletion (c) Swap

(d) Random (e) Embed (f) Overall

Figure 4: The accuracy of methods over different numbers and types of attacks.

| # | Attacked Sentence | Recovered Token | Label | Pred |
|---|---|---|---|---|
| 1 | Mr. Tsai is a very **orig~~i~~nal** artist in his medium, and what time is it there? | imaginative | positive | positive |
| 2 | Old-form moviemaking at its **be~~s~~t**. | best | positive | positive |
| 3 | My reaction in a word: **disapponitment**. | that | negative | positive |
| 4 | a **painfulily funtny** ode to **g**bad behavior. | painfully; silly; one | positive | negative |

Table 6: A case study of recovered tokens in SST-2. Note that Label and Pred represent the ground-truth label and the predicted label.

documents from SST-2 for a case study. We successfully recovered the attacked words from "original" and "bet" in the cases 1 and 2 to "imaginative" and "best". It demonstrates that embeddings generated by the embedding estimator are robust to recover the appropriate tokens and block adversarial attacks. However, DISP performs worse when the remaining sentence is lack of informative contexts as case 3. When multiple attacks exist, the incorrect context may also lead to unsatisfactory recoveries, e.g., DISP converts "funny" to "silly" in case 4, thus flipping the prediction. This experiment depicts a disadvantage of DISP and demonstrates that DISP+SC can gain further improvements.

**Embedding Estimator.** Although DISP is not required to recover the ground-truth perturbed tokens, the embedding estimator plays an important role to derive appropriate embedding vectors that obtain the original semantics. We first evaluate the performance of embedding estimator as a regression task. The RMSE scores of estimated embeddings are 0.0442 and 0.1030 in SST-2 and IMDb datasets, which are small enough to derive satisfactory tokens. To further demonstrate the robustness of the embedding estimator and estimated embeddings, we identify the perturbations with our discriminator and replace them with the ground-truth tokens. Table 7 shows the accuracy scores over different types of attacks in the SST-2 dataset. DISP and DISP$_G$ denotes the recovery performance with our estimator and gound-truth tokens, respectively. More specifically, the accuracy of DISP$_G$ presents the upperbound performance gained by the embedding estimator. The experimental results demonstrate the robustness of

| Method | Insertion | Delete | Swap |
|---|---|---|---|
| DISP$_G$ | 0.8773 | 0.8681 | 0.8796 |
| DISP | 0.8278 | 0.8278 | 0.8301 |
| Method | Random | Embed | Overall |
| DISP$_G$ | 0.7970 | 0.7924 | 0.8429 |
| DISP | 0.7773 | 0.7784 | 0.8083 |

Table 7: The performance of DISP using ground-truth and recovered tokens over different types of attacks in SST-2. Result are in accuracy. Note that DISP$_G$ denotes DISP using ground-truth tokens.

| Method | Insertion | Delete | Swap |
|---|---|---|---|
| BERT | 0.1160 | 0.1407 | 0.1806 |
| DISP | 0.5856 | 0.5684 | 0.6008 |
| Method | Random | Embed | Overall |
| BERT | 0.0855 | 0.0817 | 0.1209 |
| DISP | 0.4848 | 0.5114 | 0.5502 |

Table 8: The accuracy scores of BERT and DISP over different types of attacks on the CoLA dataset for the task of linguistic acceptability classification. The accuracy score of BERT without any attack is 0.8519.

the embedding estimator while the estimated embeddings only slightly lower the accuracy of DISP.

**Linguistic Acceptability Classification.** In addition to the task of sentiment analysis, we also evaluate the performance of DISP in linguistic acceptability classification. The Corpus of Linguistic Acceptability (CoLA) is a binary classification task. The goal of this task is to predict whether an English sentence is linguistically acceptable or not (Warstadt et al., 2018). Table 8 presents the accuracy scores of BERT and DISP on the CoLA dataset with one adversarial attack of each type. It is interesting that the original BERT is extremely vulnerable to the adversarial attacks. This is because the linguistic acceptability can be easily affected by perturbations. The experimental results also depict that DISP can significantly alleviate the performance drops. DISP is capable of blocking adversarial attacks across different NLP tasks.

## 5 Conclusions

In this paper, we propose a novel approach to discriminate perturbations and recover the text semantics, thereby blocking adversarial attacks in NLP. DISP not only correctly identifies the perturbations but also significantly alleviates the performance drops caused by attacks.

## Acknowledgment

We would like to thank the anonymous reviewers for their helpful comments. The work was supported by NSF DGE-1829071 and NSF IIS-1760523.

## References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. *arXiv preprint arXiv:1804.07998*.

Minhao Cheng, Wei Wei, and Cho-Jui Hsieh. 2019. Evaluating and enhancing the robustness of dialogue systems: A case study on a negotiation agent. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3325–3335.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2017. Hotflip: White-box adversarial examples for text classification. *arXiv preprint arXiv:1712.06751*.

Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 50–56. IEEE.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (2015)*.

Shixiang Gu and Luca Rigazio. 2014. Towards deep neural network architectures robust to adversarial examples. *arXiv preprint arXiv:1412.5068*.

Shixiang Gu and Luca Rigazio. 2015. Towards deep neural network architectures robust to adversarial examples. In *ICLR*.

Aminul Islam and Diana Inkpen. 2009. Real-word spelling correction using google web it 3-grams. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3*, pages 1241–1249. Association for Computational Linguistics.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. 2017. Deep text classification can be fooled. *arXiv preprint arXiv:1704.08006*.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pages 142–150. Association for Computational Linguistics.

Yury A Malkov and Dmitry A Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*.

Gabriel Marzinotto, Géraldine Damnati, Frédéric Béchet, and Benoit Favre. 2019. Robust semantic parsing with adversarial learning for domain generalization. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Industry Papers)*, pages 166–173.

Eric Mays, Fred J Damerau, and Robert L Mercer. 1991. Context based spelling correction. *Information Processing & Management*, 27(5):517–522.

Dongyu Meng and Hao Chen. 2017. Magnet: a two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 135–147. ACM.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. 2017. Universal adversarial perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1765–1773.

Anh Nguyen, Jason Yosinski, and Jeff Clune. 2015. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 427–436.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

Aaditya Prakash, Nick Moran, Solomon Garber, Antonella DiLillo, and James Storer. 2018. Deflecting adversarial attacks with pixel deflection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8571–8580.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 1085–1097.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Semantically equivalent adversarial rules for debugging nlp models. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 856–865.

Keisuke Sakaguchi, Matt Post, and Benjamin Van Durme. 2017. Grammatical error correction with neural reinforcement learning. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 366–372.

Pouya Samangouei, Maya Kabkab, and Rama Chellappa. 2018. Defense-gan: Protecting classifiers against adversarial attacks using generative models. In *ICLR*.

Suranjana Samanta and Sameep Mehta. 2017. Towards crafting text adversarial samples. *arXiv preprint arXiv:1707.02812*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Wenqi Wang, Benxiao Tang, Run Wang, Lina Wang, and Aoshuang Ye. 2019. A survey on adversarial attacks and defenses in text. *arXiv preprint arXiv:1902.07285*.

David Warde-Farley and Yoshua Bengio. 2017. Improving generative adversarial networks with denoising feature matching. In *ICLR*.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2018. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*.

Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. 2019. Adversarial examples: Attacks and defenses for deep learning. *IEEE transactions on neural networks and learning systems*.

Qingfu Zhu, Lei Cui, Wei-Nan Zhang, Furu Wei, and Ting Liu. 2019. Retrieval-enhanced adversarial training for neural response generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3763–3773, Florence, Italy. Association for Computational Linguistics.

# A Neural Citation Count Prediction Model based on Peer Review Text

**Siqing Li**[1,2], **Wayne Xin Zhao**[*1,2], **Eddy Jing Yin** and **Ji-Rong Wen**[1,2]
[1]School of Information, Renmin University of China
[2]Beijing Key Laboratory of Big Data Management and Analysis Methods
{lisiqing,jrwen}@ruc.edu.cn
batmanfly@gmail.com eddy_yin@hotmail.com

## Abstract

Citation count prediction (CCP) has been an important research task for automatically estimating the future impact of a scholarly paper. Previous studies mainly focus on extracting or mining useful features from the paper itself or the associated authors. An important kind of data signals, *i.e.,* peer review text, has not been utilized for the CCP task. In this paper, we take the initiative to utilize peer review data for the CCP task with a neural prediction model. Our focus is to learn a comprehensive semantic representation for peer review text for improving the prediction performance. To achieve this goal, we incorporate the abstract-review match mechanism and the cross-review match mechanism to learn deep features from peer review text. We also consider integrating hand-crafted features via a wide component. The deep and wide components jointly make the prediction. Extensive experiments have demonstrated the usefulness of the peer review data and the effectiveness of the proposed model. Our dataset has been released online.

## 1 Introduction

In recent years, the number of scientific publications has been growing in a dramatic rate. For example, the numbers of submissions and accepted papers of EMNLP 2019 have increased to 2,877 and 684 respectively[1]. Given the huge volume of scholarly papers, a long-standing research challenge is how to effectively evaluate the impact of scientific literature (Garfield, 1999; Saha et al., 2003; Bornmann, 2013). A typical way to measure the impact of a scholarly paper is through the number of citations received after publication (Garfield, 1979; Aksnes, 2006), reflecting the influence in the research community.

Since citation count is an important evaluation measure for scientific impact, many researchers aim to develop automatic ways to predict the future citation of a paper (Castillo et al., 2007; Ibáñez et al., 2009; Davletov et al., 2014; Xiao et al., 2016). A typical approach is to casting the problem into a classification or regression task, focusing on extracting useful feature information (Yan et al., 2011; Chen and Zhang, 2015; Singh et al., 2015; Park et al., 2017) (*e.g.,* $h$-index and topic distribution). Although these studies have achieved important progress on this task, they mainly utilize information from the papers themselves or their associated authors. They have neglected an important kind of data signal for the prediction task, *i.e.,* peer review data.

Peer review is a widely adopted paper evaluation mechanism, in which three or more reviewers would be assigned to decide whether to accept or reject a paper. During the review process, the reviewers should assess the paper quality in terms of several important factors, including originality, correctness, substance and readability[2]. Intuitively, peer review data should be useful to predict future impact of a paper, since the review text contains assessment comments from domain experts. Fortunately, the mechanism of open review (Soergel et al., 2013) has made it possible to obtain peer review data for the citation count prediction (CCP) task.

Although it is appealing to leverage peer reviews for the CCP task, it is difficult to effectively extract supporting evidence and learn comprehensive semantic representations from peer review data. Reviews are usually written in natural language text, covering the assessment comments of a paper in multiple aspects. Some comments may not focus on the main contribution of a paper. For

*Corresponding author
[1]https://emnlp2019.org
[2]https://acl2018.org



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4914–4924,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

example, a review typically contains the reminders for minor spelling errors or format problems. Another interesting observation is that different reviewers may focus on different aspects in their comments, and even raise divergent attitudes on the same aspect. Hence, it is important to consider both coverage and divergence of review comments for making a comprehensive decision on the paper impact.

In this paper, we take the initiative to study how to utilize the peer review data in the CCP task. We focus on how to learn a comprehensive semantic representation from peer review text for improving the prediction performance. To identify relevant evidence from long text, we utilize the abstract-review match method to learn abstract-aware review representations by using abstract text as an attentive query. In this way, we can reduce the influence of irrelevant content or noise. To further characterize the interaction among multiple reviews, we propose a novel cross-review match mechanism. With such a mechanism, a review representation will be decomposed into a parallel representation and an orthogonal representation by referring to the rest of the reviews. Our model can derive an effective semantic representation for capturing the comprehensive semantics of all the reviewers.

To evaluate our model, we have constructed two peer review datasets with citation counts. Extensive experiments have demonstrated the superiority of the proposed model over several competitive baselines. To our knowledge, it is the first time that peer review data has been utilized in the CCP task. Our work has shown that peer review data is important to improve the prediction performance. Our code and dataset have been released at `https://github.com/RUCAIBox/Citation-Count-Prediction`.

## 2 Related Work

Citation count prediction has been a hot research topic in the literature (Castillo et al., 2007; Ibáñez et al., 2009; Chakraborty et al., 2014). Early studies casted this task as a classification or regression task (Fu and Aliferis, 2008). Their focus was to identify features in a certain aspect to explore the factors of the impact of papers. Following works formally defined this task and thoroughly examined various possible factors correlated with citation counts (Yan et al., 2011; Bhat et al., 2015; Chen and Zhang, 2015; Singh et al., 2015; Chen and Zhang, 2015; Park et al., 2017). These studies mainly model the long-term scientific impact (Wu et al., 2019; Abrishami and Aliakbary, 2018; Yuan et al., 2018). Furthermore, some researchers casted the problem as a time series task, and focused on analyzing temporal features or patterns in the process of citation growth (Davletov et al., 2014; Xiao et al., 2016; Yuan et al., 2018). However, to the best of our knowledge, no work has utilized peer review data of scholarly papers for citation count prediction.

As an important paper evaluation mechanism, peer review has been widely adopted in various journals and conferences (Ross et al., 2006; Fisher et al., 1994). Based on private review data, researchers have explored the usefulness of peer reviews in several aspects, such as issue localization (Xiong et al., 2010), review utility (Xiong and Litman, 2011) and quality/tone (Ramachandran and Gehringer, 2011). More recently, to lower the barrier to studying peer reviews for the scientific community, a public dataset of peer reviews has been released for research purpose (Kang et al., 2018). Based on this dataset, Wang and Wan (2018) have employed peer review text to predict the overall decision status for a paper submission. Compared with (Wang and Wan, 2018), we focus on a different task by studying how to leverage peer review for future impact estimation instead of paper acceptance, which has its own technical challenges. Besides, we have released our dataset with citation counts online.

Our work is also related to the studies that analyze scientific literature or citation data, including concept or keyphrase extraction (Shen et al., 2018; Gordon et al., 2016; Luan et al., 2017; Caragea et al., 2014), citation or influence analysis (Lauscher et al., 2018; Chakraborty and Narayanam, 2016; Bonab et al., 2018; Chen et al., 2018), context modeling (Cohan and Goharian, 2017; Jin and Szolovits, 2018) and automatic paper rating (Yang et al., 2018).

## 3 Problem Formulation

Let $d$ denote a scientific paper from a literature corpus $\mathcal{D}$. Following (Ibáñez et al., 2009), we only consider the abstract text for modeling paper content, which summarizes the main contributions of a paper. Let $\boldsymbol{a}_d$ denote the abstract text of $d$, which consists of multiple abstract sentences.

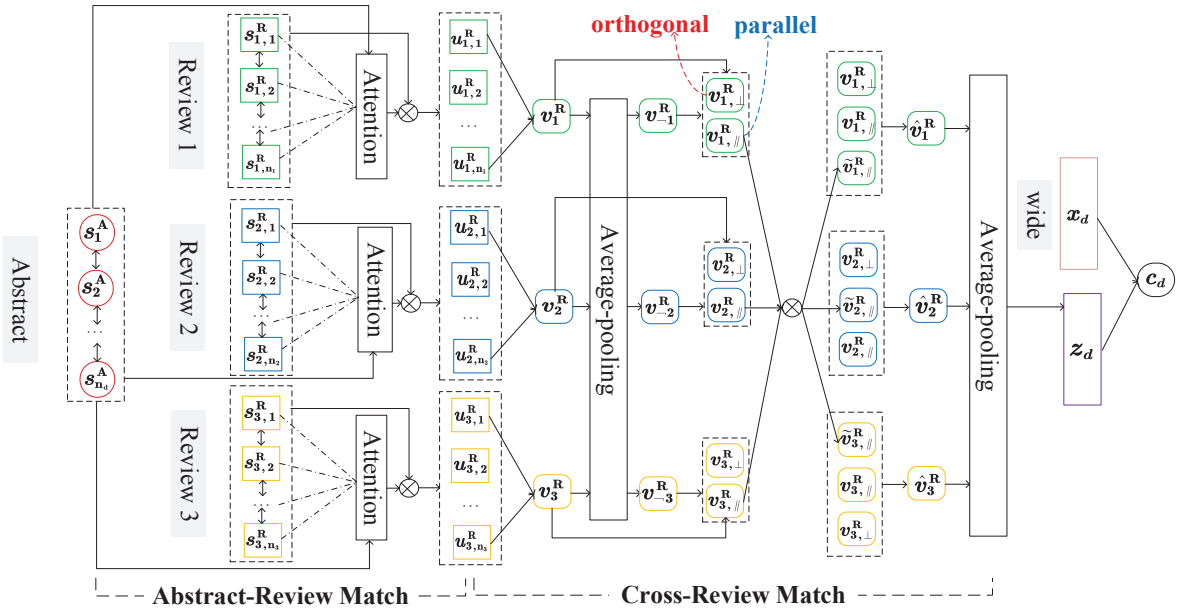


Figure 1: The overview of the proposed citation count prediction model.

We also assume that $K$ peer reviews are available for paper $d$, characterized by $\{\boldsymbol{r}_k\}_{k=1}^{K}$, where $\boldsymbol{r}_k$ denotes the $k$-th review consisting of multiple review sentences. We assume that both abstract and review text share the same vocabulary $\mathcal{V}$. Besides these features, we also assume other types of information (*e.g.,* authors' $h$-index) are also available for our task. We use a vectorized representation $\boldsymbol{x}_d$ to encode all non-review features.

Based on the above preliminaries, we now define the Citation Count Prediction (CCP) task. We aim to learn an effective predictive function that takes as input the abstract text, review text and other available information and estimates the future citation count after a given time period:

$$f(\boldsymbol{x}_d, \boldsymbol{a}_d, \{\boldsymbol{r}_k\}_{k=1}^{K}) \rightarrow \hat{c}_d, \tag{1}$$

where $\hat{c}_d$ is the estimated citation count for $d$. To make the citation number more predictable, we normalize the value range of $c_d$ within the interval $(0, 1)$. Here, we consider long-term citation count prediction in terms of years.

## 4 The Proposed Model

In this section, we present a neural citation count prediction model based on peer review text. Our model consists of two major components, namely the deep component and wide component, which model review-based text features and other hand-crafted features, respectively. Figure 1 presents an illustrative sketch for our model architecture. The notations and the descriptions are shown in Table 1.

| Symbols | Descriptions |
|---|---|
| $\boldsymbol{x}_d$ | the non-review features of paper $d$ |
| $\boldsymbol{z}_d$ | the final review representation of paper $d$ |
| $\boldsymbol{c}_d$ | the citation count of paper $d$ |
| $\boldsymbol{a}_d$ | the abstract text of paper $d$ |
| $\boldsymbol{r}_k$ | the $k$-th review consisting of multiple review sentences |
| $\boldsymbol{s}_j^A$ | the learned representations of the $j$-th sentence in the abstract text |
| $\boldsymbol{s}_{k,j}^R$ | the learned representations of the $j$-th sentence in the $k$-th review text |
| $h_S$ | the dimension of sentence vectors |
| $n_d$ | the number of sentences in abstract of paper $d$ |
| $n_k$ | the number of sentences in the $k$-th review of paper $d$ |
| $\boldsymbol{u}_t^R$ | the updated representation of the $t$-th sentence in a review after abstract-review match |
| $h_H$ | the GRU hidden size |
| $\boldsymbol{v}_k^R$ | the representation of the $k$-th review |
| $\boldsymbol{v}_{k,\parallel}^R$ | the parallel representation of the $k$-th review |
| $\boldsymbol{v}_{k,\perp}^R$ | the orthogonal representation of the $k$-th review |
| $\boldsymbol{v}_{\neg k}^R$ | the representation of other reviews excluding the $k$-th review |
| $\hat{\boldsymbol{v}}_k^R$ | the refined representation of the $k$-th review after cross-review match |

Table 1: Notations used in the paper.

### 4.1 The Deep Component

The deep component is the core part of our model, which aims to extract important semantic charac-

teristics from peer review text for the CCP task. We first encode abstract and review sentences into embedding vectors, and then distill the relevant evidence from review text by referring to the abstract. To characterize the interaction of multiple reviewers, we further design a cross-review match mechanism to capture both consistency and divergence among different reviews.

#### 4.1.1 Encoding Abstract and Review Sentences

We first pretrain the word embeddings using the word2vec model (Mikolov et al., 2013) using all the scientific corpus. To effectively encode the abstract and review sentences, we adopt the convolution-based method in (Kim, 2014) to model the sentences, sequentially consisting of a lookup layer, a convolution layer of 100 filters, and a max pooling layer. We denote the learned sentence representations of the abstract text as $\{\boldsymbol{s}_j^A\}_{j=1}^{n_d}$ and the $k$-th review text is denoted as $\{\boldsymbol{s}_{k,j}^R\}_{j=1}^{n_k}$, where each $\boldsymbol{s}_j^A$ or $\boldsymbol{s}_{k,j}^R$ is a $h_S$-dimensional vector for the $j$-th sentence in the abstract or the $k$-th review, and $n_d$ and $n_k$ is the number of sentences in abstract of paper $d$ and its $k$-th review.

#### 4.1.2 Improving the Review Representations with Abstract-Review Match

Review text reflects the subject assessment on a paper by the reviewers. A review is likely to cover the detailed comments from multiple aspects. It may contain irrelevant information for the prediction task, such as requesting source code release or pointing out minor spelling errors. It is key to identify relevant information focusing on the core contributions of a paper.

Intuitively, we can utilize the abstract information to purify the original review sentence representations, since it provides a good summary for the main contributions of a paper. Inspired by the recent progress on machine reading comprehension, we adopt the gated attention-based recurrent networks (Wang et al., 2017) for refining the review representations regarding to the abstract text. In our task, the abstract is considered as a question, and a review is considered as a passage. Similar to machine reading comprehension, we aim to learn a "question"-relevant review sentence representation that focuses on the core content from the abstract. For simplicity, we only consider the interaction between the abstract and an individual review, and omit the review index from the notations. Formally, we update the representation of the $t$-th sentence in a review as $\boldsymbol{u}_t^R \in \mathbb{R}^{h_H}$:

$$\boldsymbol{u}_t^R = \text{BiGRU}(\boldsymbol{u}_{t-1}^R, [\boldsymbol{s}_t^R, \boldsymbol{p}_t]^*), \tag{2}$$

where $\boldsymbol{p}_t \in \mathbb{R}^{h_S}$ is an attentional vector of a review computed based on the interaction between review and abstract sentence representations:

$$\begin{aligned} \boldsymbol{p}_t &= \sum_{i=1}^{n_d} \alpha_i^t \cdot \boldsymbol{s}_i^A, \\ \alpha_i^t &= \exp(\boldsymbol{h}_i^t) / \sum_{j=1}^{n_d} \exp(\boldsymbol{h}_j^t), \\ \boldsymbol{h}_j^t &= \boldsymbol{u}^\top \tanh(\boldsymbol{W}_s^A \boldsymbol{s}_j^A + \boldsymbol{W}_s^R \boldsymbol{s}_t^R + \boldsymbol{W}_u^R \boldsymbol{u}_{t-1}^R), \end{aligned} \tag{3}$$

where $\boldsymbol{W}$s $\in \mathbb{R}^{h_H \times h_S}$are parameter matrices to learn, $\boldsymbol{s}_j^A$ and $\boldsymbol{s}_t^R$ are the learned sentence representations for abstract and review text respectively in Sec. 4.1.1, $\alpha$s are the attention coefficients, and $[\boldsymbol{s}_t^R, \boldsymbol{p}_t]^*$ is a gated update of the concatenation vector $[\boldsymbol{s}_t^R, \boldsymbol{p}_t]$:

$$[\boldsymbol{s}_t^R, \boldsymbol{p}_t]^* = \boldsymbol{g}_t \odot [\boldsymbol{s}_t^R, \boldsymbol{p}_t], \tag{4}$$

$$\boldsymbol{g}_t = \text{sigmoid}(\boldsymbol{W}_g [\boldsymbol{s}_t^R, \boldsymbol{p}_t]), \tag{5}$$

where " $\odot$ " is an element-wise product operation for vectors.

In this way, for a review, we can obtain the abstract-aware review sentence representations $\{\boldsymbol{u}_j^R\}_{j=1}^{n_d}$, which encode more relevant information emphasized by the abstract. To learn the overall representation for the $k$-th review, we concatenate the sentence embeddings of the first and last sentences in it:

$$\boldsymbol{v}_k^R = [\boldsymbol{u}_1^R, \boldsymbol{u}_{n_k}^R], \tag{6}$$

where $\boldsymbol{u}_1^R$ and $\boldsymbol{u}_{n_k}^R$ are learned sentence embeddings using bidirectional Gated Recurrent Unit (BiGRU) in Eq. 2.

#### 4.1.3 Improving Review Representations with Cross-Review Match

Previously, we have considered the interaction between the abstract and an individual review. The evaluation process of a paper typically requires multiple reviewers to make the final decision. According to (Hirschauer, 2010), coverage and divergence should be considered for the acceptance decision of a paper. Therefore, we propose to utilize cross-review match to learn a comprehensive semantic representation from different reviewers.

Given a review, we take the rest of the reviews as a reference source. We aim to learn the common semantics that are also discussed by other reviews (maybe with different attitudes), and identify unmentioned semantics by other reviews. To implement this idea, we adopt the orthogonal decomposition strategy proposed in (Wang et al., 2016). We decompose the original review representation into a *parallel* representation and an *orthogonal* representation. Formally, the representation of the $k$-th review $\boldsymbol{v}_k^R \in \mathbb{R}^{h_H}$ (Eq. 6) is decomposed to:

$$\boldsymbol{v}_{k,\parallel}^R = \frac{\boldsymbol{v}_k^{R\top} \cdot \boldsymbol{v}_{\neg k}^R}{\boldsymbol{v}_{\neg k}^{R\top} \cdot \boldsymbol{v}_{\neg k}^R} \cdot \boldsymbol{v}_{\neg k}^R, \quad (7)$$

$$\boldsymbol{v}_{k,\perp}^R = \boldsymbol{v}_k^R - \boldsymbol{v}_{k,\parallel}^R, \quad (8)$$

$$\boldsymbol{v}_{\neg k}^R = \text{avg-pooling}(\{\boldsymbol{v}_j^R\}_{j \neq k}). \quad (9)$$

where the parallel representation $\boldsymbol{v}_{k,\parallel}^R \in \mathbb{R}^{h_H}$ encodes common semantics also discussed by other reviews, and the orthogonal representation $\boldsymbol{v}_{k,\perp}^R \in \mathbb{R}^{h_H}$ encodes unmentioned semantics in other reviews. Such a decomposition is useful to extract more comprehensive semantics from multiple reviews. We use average pooling to construct the reference vector of other reviews.

We perform the above decomposition for each review associated with a paper. The parallel representation reflects the common semantics shared by other reviews. Since different reviewers may have divergent comments (or attitudes) towards the same content, we further introduce a corresponding attentional representation for enriching the semantics of the original representation:

$$\widetilde{\boldsymbol{v}}_{k,\parallel}^R = \sum_{j=1}^{K} \alpha_{kj} \cdot \boldsymbol{v}_{j,\parallel}^R, \quad (10)$$

$$\alpha_{k,j} = \exp(s_{k,j}) / \sum_{j'=1}^{K} \exp(s_{i,j'}),$$

$$s_{k,j} = \begin{cases} 0, & \text{if } k = j \\ (\boldsymbol{v}_{k,\parallel}^R)^\top \cdot \boldsymbol{v}_{j,\parallel}^R, & \text{otherwise.} \end{cases}$$

Then we combine the three representations and adopt a fully connected layer to obtain the refined representation for the $k$-th review $\hat{v}_k^R \in \mathbb{R}^{h_H}$:

$$\hat{v}_k^R = \boldsymbol{W}[\boldsymbol{v}_{k,\perp}^R, \boldsymbol{v}_{k,\parallel}^R, \widetilde{\boldsymbol{v}}_{k,\parallel}^R]. \quad (11)$$

It is able to capture the coverage and divergence in semantics for peer review text to some extent. Finally we use an average pooling operation over all review representations of paper $d$ as its final representation $\boldsymbol{z}_d \in \mathbb{R}^{h_H}$:

$$\boldsymbol{z}_d = \text{avg-pooling}(\{\hat{v}_k^R\}_{k=1}^K). \quad (12)$$

### 4.2 The Wide Component

Besides review-based features, we consider directly integrating other important features for the prediction task, called *wide features*. Here, we use a vectorized representation $\boldsymbol{x}_d$ to represent all the wide features. We construct the vector by using the features proposed in previous studies (Yan et al., 2011; Bhat et al., 2015):

- Topic distribution: We utilize the Latent Dirichlet Allocation (LDA) (Blei et al., 2003) to learn the probability distribution over topics as the topic features.
- Diversity: We calculate the entropy of the paper's topic distribution to measure the topical breadth of a paper.
- Recency: We use the year of publication as the temporal feature to predict the citation count.
- Author influence: We use the number of authors and the average $h$-index as author features.

### 4.3 The Joint Deep and Wide Model

Finally, we integrate the two components into a unified model. We take as input the deep and wide features respectively discussed in Section 4.1 and Section 4.2, and combine them as the prediction function (Eq. 1) :

$$\hat{c}_d = \tanh(\boldsymbol{w}_{deep}^\top \cdot \boldsymbol{z}_d + \boldsymbol{w}_{wide}^\top \cdot \boldsymbol{x}_d + b), \quad (13)$$

where $\boldsymbol{z}_d$ and $\boldsymbol{x}_d$ are the derived feature representations from the deep and wide components respectively, and $\boldsymbol{w}_{deep}$ and $\boldsymbol{w}_{wide}$ are the corresponding parameter vectors. Furthermore, we define the citation count prediction error over the training set with the Mean Squared Error (MSE):

$$L(\theta) = \frac{1}{|\mathcal{D}|} \sum_{d \in \mathcal{D}} (\hat{c}_d - c_d)^2. \quad (14)$$

where $c_d$ is the normalized real citation count of each scholarly paper. We learn our model parameters via optimizing the $L(\theta)$ loss. The parameters in GRU and CNN are initialized by a normal distribution with zero mean and 0.01 variance, and the biases are initialized as zeros. To optimize our model, we adopt the Stochastic Gradient Descent (SGD) optimizer to learn the

| Dataset | NIPS (2013-2016) | | ICLR (2013-2017) | |
|---|---|---|---|---|
| | Abstract | Review | Abstract | Review |
| #Doc. | 1,739 | 7,171 | 384 | 1,119 |
| #Sent. | 10,964 | 109,674 | 2,368 | 15,553 |
| #Words | 6,448 | 16,695 | 2,993 | 6,680 |
| #Ave. Rev. | 4.12 | | 2.91 | |

Table 2: Statistics of our datasets after preprocessing.

model parameters. More implementation details can be found in Section 5.1.

## 5 Experiments

In this section, we first set up the experiments, and then present the results and analysis.

### 5.1 Experimental Setup

**Datasets**. Peer review data is not available for the majority of mainstream journals and conferences. Fortunately, ICLR and NIPS have provided the review text on their website. NIPS does not provide rating scores from the reviewers, and we only consider utilizing text data in this paper. For most of the published papers, it is difficult to accumulate a considerable number of citations in a short period. Hence, we only use the data ICLR 2013-2017 and NIPS 2013-2016 for evaluation, which has a two-year span to now for long-term impact prediction. The data of NIPS 2013-2016 and ICLR 2017 have been shared by Kang et al. (2018). The data of ICLR 2013-2016 was crawled from the OpenView website[3], including abstract text, review text and author data. Note that we only consider the accepted papers for citation prediction. We further crawl author data (*e.g.,* $h$-index) and paper citation from Google Scholar[4]. When encountering any ambiguity on author names or paper titles, a senior graduate student will manually collect the corresponding data. All the Google Scholar data has been accessed on March 31, 2019 to guarantee the recency of citation data. We remove the papers with only one reviewer. We perform basic text preprocessing using NLTK[5], including tokenization, lowercase, and stopword removal, and only retain the words that occur three times or more. In order to simulate the real situation, for both datasets, we take the data from the last year as test data, and the previous data as training set. The detailed statistics of the two datasets are summarized in Table 2. Since both datasets have a limited number of papers, we only hold out 5% training data as validation set. Our dataset and code are available at https://github.com/RUCAIBox/Citation-Count-Prediction.

**Implementation Details**. We pre-trained 300-dimensional word vectors as initial word embeddings using all the data in our dataset. The word vectors were fixed during the training process. The number of CNN filters $h_S$ and GRU hidden size $h_H$ are set to 128. For the wide component, we utilize the LDA to train a 100-topic model. All of the features in the wide component and the citation count are normalized within the interval $(0, 1)$. The dropout rate is set to 0.5 to prevent overfitting. For the hyper-parameters of SGD optimizer, we set the learning rate as $0.001$.

**Baseline Models**. We compare our model against a number of baseline models:

• Linear Regression (LR), K-NearestNeighbor (KNN), Support Vector Regression (SVR) and Gradient Boost Regression Tree (GBRT): The four methods are commonly used regression models for citation prediction (Yan et al., 2011; Bhat et al., 2015). We adopt the same wide features from (Yan et al., 2011) as our wide component.

• Wide & Deep (Cheng et al., 2016): We borrow the Wide & Deep leaning framework from recommender systems to predict the citation count. We modify the deep component by implementing it as a feed-forward neural network on top of a bi-directional RNN component over review text.

• MILAM (Wang and Wan, 2018): It is a multiple instance learning network with a novel abstract-based memory mechanism to predict the overall decision (accept, reject, or borderline) based on review text. We modify the loss of this model as the MSE loss for citation regression. For a fair comparison, we also integrate the wide features in a similar way as our wide component.

**Evaluation Metrics**. To evaluate the performance of different methods on citation count prediction, following previous studies (Bhat et al., 2015; Yuan et al., 2018), we adopt five evaluation metrics, including MAE, RMSE, OR@30, OR@50, and Spearman's rank correlation coefficient. MAE and RMSE measure the difference between the real value and the predicted value for a regression task.

[3]https://openreview.net/
[4]https://scholar.google.com
[5]https://www.nltk.org

| Datasets | Models | MAE ("↓") | RMSE ("↓") | OR(@30) ("↑") | OR(@50) ("↑") | Spearman's Rank ("↑") |
|---|---|---|---|---|---|---|
| NIPS | LR | 0.1776 | 0.1903 | 0.27 | 0.33 | 0.4776 |
| | KNN | 0.1701 | 0.1900 | 0.33 | 0.36 | 0.4848 |
| | SVR | 0.1677 | 0.1856 | 0.33 | 0.40 | 0.5279 |
| | GBRT | 0.1863 | 0.1974 | 0.23 | 0.34 | 0.5310 |
| | Wide&Deep | 0.1470 | 0.1848 | 0.30 | 0.38 | 0.5351 |
| | MILAM | 0.1426 | 0.1792 | 0.37 | 0.38 | 0.5458 |
| | Our model | **0.1349** | **0.1726** | **0.4** | **0.42** | **0.5561** |
| ICLR | LR | 0.2395 | 0.2723 | 0.40 | 0.70 | 0.1475 |
| | KNN | 0.2293 | 0.2674 | 0.40 | 0.72 | 0.1874 |
| | SVR | 0.2226 | 0.2578 | 0.40 | 0.70 | 0.1328 |
| | GBRT | 0.2223 | 0.2607 | 0.43 | 0.70 | 0.1469 |
| | Wide&Deep | 0.2182 | 0.2607 | 0.47 | 0.72 | 0.2440 |
| | MILAM | 0.2093 | 0.2510 | 0.47 | 0.72 | 0.2510 |
| | Our model | **0.1866** | **0.2279** | **0.50** | **0.76** | **0.3026** |

Table 3: Performance comparisons of different methods for citation count prediction using two datasets. "↑" ( "↓") indicates that a larger (smaller) value corresponds to a better performance.

Spearman's Rank measures the overall correlation between the predicted list and the ground-truth list sorted by the citation number descendingly. OR@$k$ measures the overlapping ratio between top $k$ predicted results and the real ordered list.

### 5.2 Results and Analysis

In this subsection, we construct a series of experiments on the effectiveness of the proposed model for the citation count prediction task.

**Main Results**. Table 3 presents the performance of different methods on citation count prediction. We can make the following observations. First, the four traditional baselines (LR, KNN, SVR and GBRT) perform worse than the two deep learning baselines (W&D, MILAM). These four baselines only utilize the wide features with traditional machine learning models. Second, MILAM performs consistently better than W&D, since it has designed a more elaborate architecture to model the review text. Finally, our model outperforms all the baselines with a substantial margin, especially for the ICLR dataset. Our model is able to integrate the wide features and learn the comprehensive representation for peer review text, which is the key of the performance improvement over baselines. Overall, the two datasets show the similar findings. In what follows, we will report the results on ICLR dataset due to space limit.

**Ablation Analysis**. The major novelty of our model is that it utilizes abstract-review match and cross-review match to learn a comprehensive abstract-aware representation for peer review text.

| Models | MAE | SR |
|---|---|---|
| Our full model | **0.1866** | **0.3026** |
| w/o abstract-review match | 0.1983 | 0.2861 |
| w/o cross-review match | 0.2025 | 0.2735 |
| w/o wide component | 0.2065 | 0.2697 |

Table 4: Ablation analysis on ICLR dataset (SR = Spearman's Rank).

| Models | MAE | | SR | |
|---|---|---|---|---|
| | original | +review | original | +review |
| LR | 0.2395 | 0.2289 | 0.1475 | 0.1700 |
| KNN | 0.2293 | 0.2176 | 0.1874 | 0.2155 |
| SVR | 0.2226 | 0.2174 | 0.1328 | 0.1768 |
| GBRT | 0.2223 | 0.2157 | 0.1469 | 0.2239 |
| Our Model | **0.1866** | | **0.3026** | |

Table 5: Analysis of the usefulness of peer review text on ICLR dataset (SR = Spearman's Rank).

Moreover, we integrate the wide features for the prediction task. To examine the contribution of the three parts, we examine the performance of the model variants by removing each module from the complete model. We present the MAE results of our model and its three variants in Table 4. As we can see, all components are useful to improve the final performance.

**Usefulness of Peer Review Text**. A major motivation of this paper is that peer review text is useful to the citation prediction task, which has been seldom studied in previous studies. Hence, we examine whether peer review text is also

| Test Datasets | NIPS | | ICLR | |
|---|---|---|---|---|
| Models | MAE | SR | MAE | SR |
| MILAM | 0.1530 | 0.5227 | 0.2056 | 0.26 |
| Our Model | 0.1425 | 0.5348 | 0.1889 | 0.2933 |

Table 6: Analysis on cross-venue prediction (SR = Spearman's Rank).

| Abstract | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| · · · we present a general end-to-end approach to sequence learning that makes minimal assumptions on the sequence structure. Our method uses a multilayered Long Short-Term Memory (LSTM) to map the input sequence to a vector of a fixed dimensionality, and then another deep LSTM to decode the target sequence from the vector. Our main result is that on an English to French translation task · · · Additionally, the LSTM did not have difficulty on long sentences. The LSTM also learned sensible phrase and sentence representations that are sensitive to word order and are relatively invariant to the active and the passive voice. Finally, we found that reversing the order of the words in all source sentences (but not target sentences) improved the LSTM's performance markedly. | | | | | | | | |
| Reviewer #1 | | | Reviewer #2 | | | Reviewer #3 | | |
| Sentences | $v_{\parallel}$ | $v_{\perp}$ | Sentences | $v_{\parallel}$ | $v_{\perp}$ | Sentences | $v_{\parallel}$ | $v_{\perp}$ |
| [0.31]-I think your last translated-sentence example kind of shows the kind of weirdness that can result. | 0.15 | 0.85 | [0.21]-This paper presents the elegant idea of translating from source to target languages with an LSTM. | 0.93 | 0.07 | [0.19]-The ideas represented in this work are extremely interesting, and I love the elegance and simplicity of the proposed RNN architecture. | 0.88 | 0.12 |
| [0.24]-A solution has to be scalable in principle to long sentences, and I think it's clear that your method cannot. | 0.38 | 0.62 | [0.15]-This is an elegant model, and I am inclined to accept it, despite the fact that it only "works" for sentences that do not have infrequent words. | 0.63 | 0.37 | [0.17]-The idea of the paper is good and very interesting, providing an elegant neural solution to machine translation. | 0.91 | 0.09 |
| [0.21]-I am skeptical that this idea could be a practical solution to MT. | 0.44 | 0.56 | [0.09]-The paper does have some major holes in the experiments. | 0.69 | 0.31 | [0.11]-I'm sceptical that common RNNs would find it hard to model "long-term dependencies"· · · | 0.14 | 0.86 |
| [0.14]-I believe the experimental investigation is competent and complete. | 0.73 | 0.27 | [0.05]-The experimental results are not convincing | 0.71 | 0.29 | [0.04]-It would be more convincing to have some more examples. | 0.54 | 0.46 |

Table 7: Samples of the abstract and reviews from NIPS dataset. For each review, we present four comment sentences. The similarity weights *w.r.t.* the abstract text have been shown in red font.

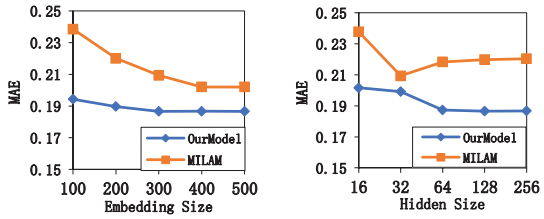


(a) Varing the word embedding size (b) Varing the RNN hidden size

Figure 2: The impact of embedding size and hidden size in terms of MAE scores on ICLR dataset.

useful to improve traditional prediction models (LR, KNN, SVR and KNN). Note that our focus is to verify the general usefulness of peer review text instead of the most suitable text features for the baselines. Here, we adopt the simple yet classic doc2vec model (Le and Mikolov, 2014) to encode peer text into a vectorized representation. Then, we integrate these text features into the four baseline methods. As shown in Table 5, the performance of all the four methods have been improved with the text features. The results have shown that the peer text is indeed generally useful for the citation prediction.

**Cross-venue Evaluation**. To examine the robustness of our model, we further perform a cross-venue evaluation. For ICLR (NIPS) test set, we apply the models trained on the full NIPS (ICLR) dataset. We only select the best baseline MILAM as a comparison. As shown in Table 6, we are able to see that the performance decreases compared with the results in Table 3, since we use a training set from a different venue. But the decline is not obvious. In the future we will consider how to improve the performance. Our model is still better than the baseline MILAM for both datasets.

**Parameter Sensitivity**. Next, we investigate the performance with respect to two major parameters in our model, *e.g.,* the word embedding size and the GRU hidden size. As shown in Figure 2(a) and Figure 2(b), our model is consistently better than the best baseline MILAM with all the parameter values. An embedding size of 300 and a hidden size of 128 yield the best results for our model.

### 5.3 Qualitative Analysis

Previously, we have shown that both the review-abstract match and cross-review match are useful in the prediction performance. In Table 7, we perform the qualitative analysis on a sample paper with three reviews for understanding how the two mechanisms work.

We first compute the similarity weight between the abstract representation and a review sentence, and sort the review sentences according to such weights. As we can see, the comments on model design have been overall ranked in a higher position than those on experiments. With the abstract-review match, our model indeed identifies

more relevant content regarding to the abstract text.

Then, we analyze the corresponding semantic explanation of the cross-review match, in which we decompose an overall review representation into a parallel representation $\boldsymbol{v}_{\parallel}$ and an orthogonal representation $\boldsymbol{v}_{\perp}$. It is difficult to directly understand the two vectors. Instead, we compute the similarity between a comment sentence and the review parallel (or orthogonal) representation. Then we normalize the two similarity values into a distribution over two representations (parallel or orthogonal). It can be seen that the comments focusing on the common aspect have a larger weight on $\boldsymbol{v}_{\parallel}$. For example, for the comments from the first row, reviewer #2 and reviewer #3 have similar general comments about the model architecture, and both comments have very large weights on the parallel representation. While, reviewer #1 raises a different comment on other model detail, corresponding to a large weight on the orthogonal representation. Interestingly, the fourth row corresponds to the comments on the experiments. The three comment sentences are more related to the parallel representation (*i.e.,* the common issue), while they have conveyed different attitudes. This phenomenon can be partially captured by the representation $\widetilde{\boldsymbol{v}}^{R}_{k,\parallel}$ (Eq. 10) by attending to the parallel representations of other reviews.

## 6 Conclusion

In this paper, we studied how to utilize the peer review text to improve the citation prediction task. We developed a joint deep and wide model that was able to integrate both deep and wide features into a unified predictive function. The deep features were learned from peer review text by applying the abstract-review and cross-review match mechanisms. We constructed two evaluation datasets with peer review text. Extensive results have demonstrated the effectiveness of our proposed model. Currently, we only consider the semantic-level representations from peer review. As future work, we will consider how to extend our work by modeling sentiments of review text.

## Acknowledgments

The authors would like to thank the anonymous reviewers for their helpful and constructive comments. This work was partially supported by the National Natural Science Foundation of China under Grant No. 61872369, 61832017 and 61572059, the Fundamental Research Funds for the Central Universities, the Research Funds of Renmin University of China under Grant No. 18XNLG22 and 19XNQ047.

## References

Ali Abrishami and Sadegh Aliakbary. 2018. Nncp: A citation count prediction methodology based on deep neural network learning techniques. *arXiv preprint arXiv:1809.04365*.

Dag W Aksnes. 2006. Citation rates and perceptions of scientific contribution. *Journal of the American Society for Information Science and Technology*, 57(2):169–185.

Harish S Bhat, Li-Hsuan Huang, Sebastian Rodriguez, Rick Dale, and Evan Heit. 2015. Citation prediction using diverse features. In *Data Mining Workshop (ICDMW), 2015 IEEE International Conference on*, pages 589–596. IEEE.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Hamed Bonab, Hamed Zamani, Erik G Learned-Miller, and James Allan. 2018. Citation worthiness of sentences in scientific reports. In *SIGIR*, pages 1061–1064.

Lutz Bornmann. 2013. What is societal impact of research and how can it be assessed? a literature survey. *Journal of the American Society for Information Science and Technology*, 64(2):217–233.

Cornelia Caragea, Florin Adrian Bulgarov, Andreea Godea, and Sujatha Das Gollapalli. 2014. Citation-enhanced keyphrase extraction from research papers: A supervised approach. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1435–1446.

Carlos Castillo, Debora Donato, and Aristides Gionis. 2007. Estimating number of citations using author reputation. In *International Symposium on String Processing and Information Retrieval*, pages 107–117. Springer.

Tanmoy Chakraborty, Suhansanu Kumar, Pawan Goyal, Niloy Ganguly, and Animesh Mukherjee. 2014. Towards a stratified learning approach to predict future citation counts. In *Proceedings of the 14th ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 351–360. IEEE Press.

Tanmoy Chakraborty and Ramasuri Narayanam. 2016. All fingers are not equal: Intensity of references in scientific articles. *arXiv preprint arXiv:1609.00081*.

Chengyao Chen, Zhitao Wang, Wenjie Li, and Xu Sun. 2018. Modeling scientific influence for research trending topic prediction. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Junpeng Chen and Chunxia Zhang. 2015. Predicting citation counts of papers. In *Cognitive Informatics & Cognitive Computing (ICCI* CC), 2015 IEEE 14th International Conference on*, pages 434–440. IEEE.

Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, pages 7–10. ACM.

Arman Cohan and Nazli Goharian. 2017. Contextualizing citations for scientific summarization using word embeddings and domain knowledge. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1133–1136. ACM.

Feruz Davletov, Ali Selman Aydin, and Ali Cakmak. 2014. High impact academic paper prediction using temporal and topological features. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 491–498. ACM.

Martin Fisher, Stanford B Friedman, and Barbara Strauss. 1994. The effects of blinding on acceptance of research papers by peer review. *Jama*, 272(2):143–146.

Lawrence D Fu and Constantin Aliferis. 2008. Models for predicting and explaining citation count of biomedical articles. In *AMIA Annual Symposium Proceedings*, volume 2008, page 222. American Medical Informatics Association.

Eugene Garfield. 1979. Is citation analysis a legitimate evaluation tool? *Scientometrics*, 1(4):359–375.

Eugene Garfield. 1999. Journal impact factor: a brief review.

Jonathan Gordon, Linhong Zhu, Aram Galstyan, Prem Natarajan, and Gully Burns. 2016. Modeling concept dependencies in a scientific corpus. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 866–875.

Stefan Hirschauer. 2010. Editorial judgments: A praxeology of ąővotingąŕin peer review. *Social Studies of Science*, 40(1):71–103.

Alfonso Ibáñez, Pedro Larrañaga, and Concha Bielza. 2009. Predicting citation count of bioinformatics papers within four years of publication. *Bioinformatics*, 25(24):3303–3309.

Di Jin and Peter Szolovits. 2018. Hierarchical neural networks for sequential sentence classification in medical scientific abstracts. *arXiv preprint arXiv:1808.06161*.

Dongyeop Kang, Waleed Ammar, Bhavana Dalvi, Madeleine van Zuylen, Sebastian Kohlmeier, Eduard Hovy, and Roy Schwartz. 2018. A dataset of peer reviews (peerread): Collection, insights and nlp applications. *arXiv preprint arXiv:1804.09635*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Anne Lauscher, Goran Glavaš, Simone Paolo Ponzetto, and Kai Eckert. 2018. Investigating the role of argumentation in the rhetorical analysis of scientific publications with neural multi-task learning models. Association for Computational Linguistics.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196.

Yi Luan, Mari Ostendorf, and Hannaneh Hajishirzi. 2017. Scientific information extraction with semi-supervised neural tagging. *arXiv preprint arXiv:1708.06075*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Ho-Min Park, Yenewondim Biadgie Sinshaw, and Kyung-Ah Sohn. 2017. Temporal citation network-based feature extraction for cited count prediction. In *International Conference on Mobile and Wireless Technology*, pages 380–388. Springer.

Lakshmi Ramachandran and Edward F Gehringer. 2011. Automated assessment of review quality using latent semantic analysis. In *2011 IEEE 11th International Conference on Advanced Learning Technologies*, pages 136–138. IEEE.

Joseph S Ross, Cary P Gross, Mayur M Desai, Yuling Hong, Augustus O Grant, Stephen R Daniels, Vladimir C Hachinski, Raymond J Gibbons, Timothy J Gardner, and Harlan M Krumholz. 2006. Effect of blinded peer review on abstract acceptance. *Jama*, 295(14):1675–1680.

Somnath Saha, Sanjay Saint, and Dimitri A Christakis. 2003. Impact factor: a valid measure of journal quality? *Journal of the Medical Library Association*, 91(1):42.

Zhihong Shen, Hao Ma, and Kuansan Wang. 2018. A web-scale system for scientific knowledge exploration. *arXiv preprint arXiv:1805.12216*.

Mayank Singh, Vikas Patidar, Suhansanu Kumar, Tanmoy Chakraborty, Animesh Mukherjee, and Pawan Goyal. 2015. The role of citation context in predicting long-term citation profiles: An experimental study based on a massive bibliographic text dataset. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1271–1280. ACM.

David Soergel, Adam Saunders, and Andrew McCallum. 2013. Open scholarship and peer review: a time for experimentation.

Ke Wang and Xiaojun Wan. 2018. Sentiment analysis of peer review texts for scholarly papers. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 175–184. ACM.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 189–198.

Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. 2016. Sentence similarity learning by lexical decomposition and composition. *arXiv preprint arXiv:1602.07019*.

Ziming Wu, Weiwei Lin, Pan Liu, Jingbang Chen, and Li Mao. 2019. Predicting long-term scientific impact based on multi-field feature extraction. *IEEE Access*, 7:51759–51770.

Shuai Xiao, Junchi Yan, Changsheng Li, Bo Jin, Xiangfeng Wang, Xiaokang Yang, Stephen M Chu, and Hongyuan Zha. 2016. On modeling and predicting individual paper citation count over time. In *IJCAI*, pages 2676–2682.

Wenting Xiong and Diane Litman. 2011. Automatically predicting peer-review helpfulness. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 502–507. Association for Computational Linguistics.

Wenting Xiong, Diane Litman, and Christian Schunn. 2010. Assessing reviewers' performance based on mining problem localization in peer-review data. In *Educational Data Mining 2010-3rd International Conference on Educational Data Mining*, pages 211–220.

Rui Yan, Jie Tang, Xiaobing Liu, Dongdong Shan, and Xiaoming Li. 2011. Citation count prediction: learning to estimate future citations for literature. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1247–1252. ACM.

Pengcheng Yang, Xu Sun, Wei Li, and Shuming Ma. 2018. Automatic academic paper rating based on modularized hierarchical convolutional neural network. *arXiv preprint arXiv:1805.03977*.

Sha Yuan, Jie Tang, Yu Zhang, Yifan Wang, and Tong Xiao. 2018. Modeling and predicting citation count via recurrent neural network with long short-term memory. *arXiv preprint arXiv:1811.02129*.

# Connecting the Dots: Document-level Neural Relation Extraction with Edge-oriented Graphs

**Fenia Christopoulou**[1], **Makoto Miwa**[2,3], **Sophia Ananiadou**[1]

[1]National Centre for Text Mining,
School of Computer Science, The University of Manchester, United Kingdom
[2]Toyota Technological Institute, Nagoya, 468-8511, Japan
[3]Artificial Intelligence Research Center (AIRC),
National Institute of Advanced Industrial Science and Technology (AIST), Japan
{efstathia.christopoulou, sophia.ananiadou}@manchester.ac.uk
makoto-miwa@toyota-ti.ac.jp

## Abstract

Document-level relation extraction is a complex human process that requires logical inference to extract relationships between named entities in text. Existing approaches use graph-based neural models with words as nodes and edges as relations between them, to encode relations across sentences. These models are node-based, i.e., they form pair representations based solely on the two target node representations. However, entity relations can be better expressed through unique edge representations formed as paths between nodes. We thus propose an edge-oriented graph neural model for document-level relation extraction. The model utilises different types of nodes and edges to create a document-level graph. An inference mechanism on the graph edges enables to learn intra- and inter-sentence relations using multi-instance learning internally. Experiments on two document-level biomedical datasets for chemical-disease and gene-disease associations show the usefulness of the proposed edge-oriented approach.[1]

## 1 Introduction

The extraction of relations between named entities in text, known as Relation Extraction (RE), is an important task of Natural Language Processing (NLP). Lately, RE has attracted a lot of attention from the field, in an effort to improve the inference capability of current methods (Zeng et al., 2017; Christopoulou et al., 2018; Luan et al., 2019).

In real-world scenarios, a large amount of relations are expressed across sentences. The task of identifying these relations is named inter-sentence RE. Typically, inter-sentence relations occur in textual snippets with several sentences, such as documents. In these snippets, each entity is usually repeated with the same phrases or aliases, the occurrences of which are often named entity mentions and regarded as instances of the entity. The multiple mentions of the target entities in different sentences can be useful for the identification of inter-sentential relations, as these relations may depend on the interactions of their mentions with other entities in the same document.

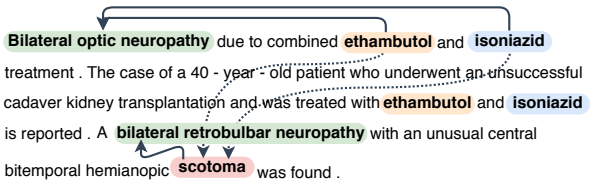


Figure 1: Example of document-level, inter-sentence relations adapted from the CDR dataset (Li et al., 2016a). The solid and dotted lines represent intra- and inter-sentence relations, respectively.

As shown in the example of Figure 1, the entities *bilateral optic neuropathy*, *ethambutol* and *isoniazid* have two mentions each, while the entity *scotoma* has one mention. The relation between the chemical *ethambutol* and the disease *scotoma* is clearly inter-sentential. Their association can only be determined if we consider the interactions between the mentions of these entities in different sentences. A mention of *bilateral optic neuropathy* interacts with a mention of *ethambutol* in the first sentence. Another mention of the former interacts with the mention of *scotoma* in the third sentence. This chain of interactions can help us infer that the entity *ethambutol* has a relation with the entity *scotoma*.

The most common technique that is currently

[1]Source code available at https://github.com/fenchri/edge-oriented-graph



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4925–4936,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

used to deal with multiple mentions of named entities is Multi-Instance Learning (MIL). Initially, MIL was introduced by Riedel et al. (2010) in order to reduce noise in distantly supervised corpora (Mintz et al., 2009). In DS, training instances are created from large, raw corpora using Knowledge Base (KB) entity linking and automatic annotation with heuristic rules. MIL in this setting considers multiple sentences (bags) that contain a pair of entities serving as the multiple instances of this pair. Verga et al. (2018) introduced another MIL setting for relation extraction between named entities in a document. In this setting, entities mapped to the same KB ID are considered as mentions of an entity concept and pairs of mentions correspond to the pair's multiple instances. However, document-level RE is not common in the general domain, as the entity types of interest can often be found in the same sentence (Banko et al., 2007). On the contrary, in the biomedical domain, document-level relations are particularly important given the numerous aliases that biomedical entities can have (Quirk and Poon, 2017).

To deal with document-level RE, recent approaches assume that only two mentions of the target entities reside in the document (Nguyen and Verspoor, 2018; Verga et al., 2018) or utilise different models for intra- and inter-sentence RE (Gu et al., 2016; Li et al., 2016b; Gu et al., 2017). In contrast with approaches that employ sequential models (Nguyen and Verspoor, 2018; Gu et al., 2017; Zhou et al., 2016), graph-based neural approaches have proven useful in encoding long-distance, inter-sentential information (Peng et al., 2017; Quirk and Poon, 2017; Gupta et al., 2019). These models interpret words as nodes and connections between them as edges. They typically perform on the nodes by updating the representations during training. However, a relation between two entities depends on different contexts. It could thus be better expressed with an edge connection that is unique for the pair. A straightforward way to address this is to create graph-based models that rely on edge representations rather focusing on node representations, which are shared between multiple entity pairs.

In this work, we tackle document-level, intra- and inter-sentence RE using MIL with a graph-based neural model. Our objective is to infer the relation between two entities by exploiting other interactions in the document. We construct a document graph with heterogeneous types of nodes and edges to better capture different dependencies between nodes. In the proposed graph, a node corresponds to either entities, mentions, or sentences, instead of words. We connect distinct nodes based on simple heuristic rules and generate different edge representations for the connected nodes. To achieve our objective, we design the model to be edge-oriented in a sense that it learns edge representations (between the graph nodes) rather than node representations. An iterative algorithm over the graph edges is used to model dependencies between the nodes in the form of edge representations. The intra- and inter-sentence entity relations are predicted by employing these edges. Our contributions can be summarised as follows:

- We propose a novel edge-oriented graph neural model for document-level relation extraction. The model deviates from existing graph models as it focuses on constructing unique nodes and edges, encoding information into edge representations rather than node representations.
- The proposed model is independent of syntactic dependency tools and can achieve state-of-the-art performance on a manually annotated, document-level chemical-disease interaction dataset.
- Analysis of the model components indicates that the document-level graph can effectively encode document-level dependencies. Additionally, we show that inter-sentence associations can be beneficial for the detection of intra-sentence relations.

## 2 Proposed Model

We build our model as a significant extension of our previously proposed sentence-level model (Christopoulou et al., 2018) for document-level RE. The most critical difference between the two models is the introduction and construction of a partially-connected document graph, instead of a fully-connected sentence-level graph. Additionally, the document graph consists of heterogeneous types of nodes and edges in comparison with the sentence-level graph that contains only entity-nodes and single edge types among them. Furthermore, the proposed approach utilises multi-instance learning when mention-level annotations are available.

As illustrated in Figure 2, the proposed model consists of four layers: sentence encoding, graph

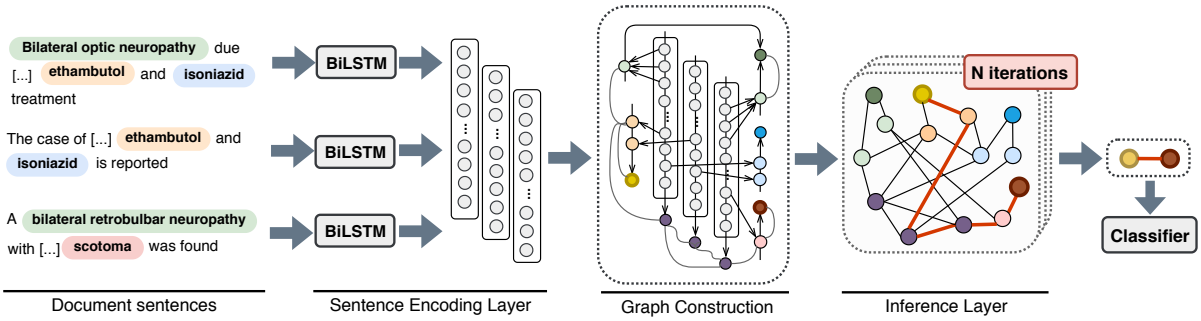


Figure 2: Abstract architecture of the proposed approach. The model receives a document and encodes each sentence separately. A document-level graph is constructed and fed into an iterative algorithm to generate edge representations between the target entity nodes. Some node connections are not shown for brevity.

construction, inference and classification layers. The model receives as input a document with identified concept-level entities and their textual mentions. Next, a document-level graph with multiple types of nodes and edges is constructed. An inference algorithm is applied on the graph edges to generate concept-level pair representations. In the final layer, the edge representations between the target concept-entity nodes are classified into relation categories.

For the remainder of this section, we first briefly introduce the document-level RE task setting and then explain the four layers of the proposed model.

### 2.1 Task Setting

In concept, document-level RE the input is considered an annotated document. The annotations include concept-level entities (with assigned KB IDs), as well as multiple occurrences of each entity under the same phrase of alias, i.e., entity mentions. We consider the associations of mentions to concept entities given (also known as entity linking (Shen et al., 2014)). The objective of the task is given an annotated document, to identify all the related concept-level pairs in that document. In this work, we refer to concept-level annotations as *entities* and mention-level annotations as *mentions*.

### 2.2 Sentence Encoding Layer

First, each word in the sentences of the input document is transformed into a dense vector representation, i.e., a word embedding. The vectorised words of each sentence are then fed into a Bi-directional LSTM network (BiLSTM) (Hochreiter and Schmidhuber, 1997; Schuster and Paliwal, 1997), named the encoder. The output of the encoder results in contextualised representations for each word of the input sentence.

### 2.3 Graph Layer

The contextualised word representations from the encoder are used to construct a document-level graph structure. The graph layer comprises of two sub-layers, a node construction layer and an edge construction layer. We compose the representations of the graph nodes in the first sub-layer and the representations of the edges in the second.

#### 2.3.1 Node construction

We form three distinct types of nodes in the graph: mention nodes (M) $n_m$, entity nodes (E) $n_e$, and sentence nodes (S) $n_s$. Each node representation is computed as the average of the embeddings of different elements. Firstly, mention nodes correspond to different mentions of entities in the input document. The representation of a mention node is formed as the average of the words ($w$) that the mention contains. Secondly, entity nodes represent unique entity concepts. The representation of an entity node is computed as the average of the mention ($m$) representations associated with the entity. Finally, sentence nodes correspond to sentences. A sentence node is represented as the average of the word representations in the sentence. In order to distinguish different node types in the graph, we concatenate a node type ($t$) embedding to each node representation. The final node representations are then estimated as $\mathbf{n}_m = [\text{avg}_{w_i \in m}(\mathbf{w}_i); \mathbf{t}_m]$, $\mathbf{n}_e = [\text{avg}_{m_i \in e}(\mathbf{m}_i); \mathbf{t}_e]$, $\mathbf{n}_s = [\text{avg}_{w_i \in s}(\mathbf{w}_i); \mathbf{t}_s]$.

#### 2.3.2 Edge construction

We initially construct non-directed edges between the graph nodes using heuristic rules that stem from the natural associations between the elements of a document, i.e., mentions, entities and sentences. As we cannot know in advance if two

entities are related, we do not directly connect entity nodes. Connections between nodes are based on pre-defined document-level interactions. The model objective is to generate entity-to-entity (EE) edge representations using other existing edges in the graph and consequently infer entity-to-entity relations. The different pre-defined edge types are described below.

**Mention-Mention (MM)**: Co-occurrence of mentions in a sentence might be a weak indication of an interaction. For this reason, we create mention-to-mention edges only if the corresponding mentions reside in the same sentence. The edge representation between each mention pair $m_i$ and $m_j$ is generated by concatenating the representations of the nodes, the contexts $c_{m_i,m_j}$ and a distance embedding associated with the distance between the two mentions $d_{m_i,m_j}$, in terms of intermediate words: $\mathbf{x}_{\text{MM}} = [\mathbf{n}_{m_i}; \mathbf{n}_{m_j}; \mathbf{c}_{m_i,m_j}; \mathbf{d}_{m_i,m_j}]$. Here, we generate the context representation for these pairs in order to encode local, pair-centric information. We use an argument-based attention mechanism (Wang et al., 2016), to measure the importance of other words in the sentence towards the mention, denoting $k \in \{1, 2\}$ as the mention arguments.

$$\begin{aligned} \alpha_{k,i} &= \mathbf{n}_{m_k}^{\intercal} \mathbf{w}_i, \\ \mathrm{a}_{k,i} &= \frac{\exp(\alpha_{k,i})}{\sum_{j \in [1,n], j \notin m_k} \exp(\alpha_{k,j})}, \\ \mathrm{a}_i &= (\mathrm{a}_{1,i} + \mathrm{a}_{2,i})/2, \\ \mathbf{c}_{m_1,m_2} &= \mathbf{H}^{\intercal}\, \mathbf{a}, \end{aligned} \tag{1}$$

where $\mathbf{n}_{m_k}$ is a mention node representation, $\mathbf{w}_i$ is a sentence word representation, $\mathrm{a}_i$ is the attention weight of word $i$ for mention pair $m_1, m_2$, $\mathbf{H} \in \mathbb{R}^{w \times d}$ is a sentence word representations matrix, $\mathbf{a} \in \mathbb{R}^w$ is the attention weights vector for the pair and $\mathbf{c}_{m_1,m_2}$ is the final context representation for the mention pair.

**Mention-Sentence (MS)**: Mention-to-sentence nodes are connected only if the mention resides in the sentence. Their initial edge representation is constructed as a concatenation of the mention and sentence nodes, $\mathbf{x}_{\text{MS}} = [\mathbf{n}_m; \mathbf{n}_s]$.

**Mention-Entity (ME)**: We connect a mention node to an entity node if the mention is associated with the entity, $\mathbf{x}_{\text{ME}} = [\mathbf{n}_m; \mathbf{n}_e]$.

**Sentence-Sentence (SS)**: Motivated by Quirk and Poon (2017), we connect sentence nodes to encode non-local information. The main differences with prior work is that our edges are unlabelled, non-directed and span multiple sentences. To encode the distance between sentences, we concatenate to the sentence node representations their distance in the form of an embedding: $\mathbf{x}_{\text{SS}} = [\mathbf{n}_{s_i}; \mathbf{n}_{s_j}; \mathbf{d}_{s_i,s_j}]$. We connect all sentence nodes in the graph. We consider $\textbf{SS}_{\textbf{direct}}$ as direct, ordered edges (distance equal to 1) and $\textbf{SS}_{\textbf{indirect}}$ as indirect, non-ordered edges (distance $> 1$) between S nodes, respectively. In our setting, **SS** denotes the combination of $\text{SS}_{\text{direct}}$ and $\text{SS}_{\text{indirect}}$.

**Entity-Sentence (ES)**: To directly model entity-to-sentence associations, we connect an entity node to a sentence node if at least one mention of the entity resides in this sentence, $\mathbf{x}_{\text{ES}} = [\mathbf{n}_e; \mathbf{n}_s]$.

In order to result in edge representations of equal dimensionality, we use different linear reduction layers for different edge representations,

$$\mathbf{e}_z^{(1)} = \mathbf{W}_z\, \mathbf{x}_z, \tag{2}$$

where $\mathbf{e}_z^{(1)}$ is an edge representation of length 1, $\mathbf{W}_z \in \mathbb{R}^{d_z \times d}$ corresponds to a learned matrix and $z \in [\text{MM}, \text{MS}, \text{ME}, \text{SS}, \text{ES}]$.

### 2.4 Inference Layer

We utilise an iterative algorithm to generate edges between different nodes in the graph, as well as to update existing edges. We initialise the graph only with the edges described in Section 2.3.2, meaning that direct entity-to-entity (EE) edges are absent. We can only generate EE edge representations by representing a path between their nodes. This implies that entities can be associated through an edge path of minimum length equal to 3[2].

For this purpose, we adapt our two-step inference mechanism, proposed in Christopoulou et al. (2018), to encode interactions between nodes and edges in the graph and hence model EE associations.

At the first step, we aim to generate a path between two nodes $i$ and $j$ using intermediate nodes $k$. We thus combine the representations of two consecutive edges $e_{ik}$ and $e_{kj}$, using a modified bilinear transformation. This action generates an edge representation of double length. We combine all existing paths between $i$ and $j$ through $k$. The $i$, $j$, and $k$ nodes can be any of the three node types E, M, or S. Intermediate nodes without adjacent

[2]Length 2 for an intra-sentence pair (E-S-E) or length 3 for an inter-sentence pair (E-S-S-E)

edges to the target nodes are ignored.

$$f\left(\mathbf{e}_{ik}^{(l)}, \mathbf{e}_{kj}^{(l)}\right) = \sigma\left(\mathbf{e}_{ik}^{(l)} \odot \left(\mathbf{W}\ \mathbf{e}_{kj}^{(l)}\right)\right), \quad (3)$$

where $\sigma$ is the sigmoid non-linear function, $\mathbf{W} \in \mathbb{R}^{d_z \times d_z}$ is a learned parameter matrix, $\odot$ refers to element-wise multiplication, $l$ is the length of the edge and $\mathbf{e}_{ik}$ corresponds to the representation of the edge between nodes $i$ and $k$.

During the second step, we aggregate the original (short) edge representation and the new (longer) edge representation resulted from Equation (3) with linear interpolation as follows:

$$\mathbf{e}_{ij}^{(2l)} = \beta\ \mathbf{e}_{ij}^{(l)} + (1-\beta) \sum_{k \neq i,j} f\left(\mathbf{e}_{ik}^{(l)}, \mathbf{e}_{kj}^{(l)}\right), \quad (4)$$

where $\beta \in [0, 1]$ is a scalar that controls the contribution of the shorter edge presentation. In general $\beta$ is larger for shorter edges as we expect that the relation between two nodes is better expressed through the shortest path between them (Xu et al., 2015; Borgwardt and Kriegel, 2005).

The two steps are repeated a finite number of times $N$. The number of iterations is correlated with the final length of the edge representations. With initial edge length $l$ equal to 1, the first iteration results in edges of length up-to 2. The second iteration results in edges of length up-to 4. Similarly, after $N$ iterations, the length of edges will be up-to $2^N$.

### 2.5 Classification Layer

To classify the concept-level entity pairs of interest, we incorporate a softmax classifier, using the entity-to-entity edges (EE) of the document graph that correspond to the concept-level entity pairs.

$$\mathbf{y} = \text{softmax}\left(\mathbf{W}_c\ \mathbf{e}_{\text{EE}} + \mathbf{b}_c\right), \quad (5)$$

where $\mathbf{W}_c \in \mathbb{R}^{r \times d_z}$ and $\mathbf{b}_c \in \mathbb{R}^r$ are learned parameters of the classification layer and $r$ is the number of relation categories.

## 3 Experimental Settings

The model was developed using PyTorch (Paszke et al., 2017). We incorporated early stopping to identify the best training epoch and used Adam (Kingma and Ba, 2015) as the model optimiser.

### 3.1 Data and Task Settings

We evaluated the proposed model on two datasets:
**CDR** (BioCreative V): The Chemical-Disease Reactions dataset was created by Li et al. (2016a) for document-level RE. It consists of 1,500 PubMed abstracts, which are split into three equally sized sets for training, development and testing. The dataset was manually annotated with binary interactions between Chemical and Disease concepts. For this dataset, we utilised PubMed pre-trained word embeddings (Chiu et al., 2016).
**GDA** (DisGeNet): The Gene-Disease Associations dataset was introduced by Wu et al. (2019), containing 30,192 MEDLINE abstracts, split into 29,192 articles for training and 1,000 for testing. The dataset was annotated with binary interactions between Gene and Disease concepts at the document-level, using distant supervision. Associations between concepts were generated by aligning the DisGeNet (Piñero et al., 2016) platform with PubMed[3] abstracts. We further split the training set into a 80/20 percentage split as training and development sets. For the GDA dataset, we used randomly initialized word embeddings.

### 3.2 Model Settings

We explore multiple settings of the proposed graph using different edges (MM, ME, MS, ES, SS) and enhancements (node type embeddings, mention-pairs context embeddings, distance embeddings). We name our model EoG, an abbreviation of Edge-oriented Graph. We briefly describe the model settings in this section.
EoG refers to our main model with edges {MM, ME, MS, ES, SS }. The EoG (`Full`) setting refers to a model with a *fully connected* graph, where the graph nodes are all connected to each other, including E nodes. For this purpose, we introduce an additional linear layer for the EE edges as in Equation (2). The EoG (`NoInf`) setting refers to a *no inference* model, where the iterative inference algorithm (Section 2.4) is ignored. The concatenation of the entity node embeddings is used to represent the target pair. In this case, we also make use of an additional EE linear layer for EE edges. Finally, the EoG (`Sent`) setting refers to a model that was trained on *sentences* instead of documents. For each entity-level pair we merge the predictions of the mention-level pairs in different sentences using a maximum assumption: if at

[3] https://www.ncbi.nlm.nih.gov/pubmed/

| Method | Overall (%) | | | Intra (%) | | | Inter (%) | | |
|---|---|---|---|---|---|---|---|---|---|
| | P | R | F1 | P | R | F1 | P | R | F1 |
| Gu et al. (2017) | 55.7 | 68.1 | 61.3 | 59.7 | 55.0 | 57.2 | 51.9 | 7.0 | 11.7 |
| Verga et al. (2018) | 55.6 | 70.8 | 62.1 | - | - | - | - | - | - |
| Nguyen and Verspoor (2018) | 57.0 | 68.6 | 62.3 | - | - | - | - | - | - |
| EoG | 62.1 | 65.2 | **63.6** | 64.0 | 73.0 | **68.2** | 56.0 | 46.7 | **50.9** |
| EoG (`Full`) | 59.1 | 56.2 | 57.6 | 71.2 | 62.3 | 66.5 | 37.1 | 42.0 | 39.4 |
| EoG (`NoInf`) | 48.2 | 50.2 | 49.2 | 65.8 | 55.2 | 60.2 | 25.4 | 38.5 | 30.6 |
| EoG (`Sent`) | 56.9 | 53.5 | 55.2 | 56.9 | 76.4 | 65.2 | - | - | - |
| Zhou et al. (2016) | 55.6 | 68.4 | 61.3 | - | - | - | - | - | - |
| Peng et al. (2016) | 62.1 | 64.2 | 63.1 | - | - | - | - | - | - |
| Li et al. (2016b) | 60.8 | 76.4 | 67.7 | 67.3 | 52.4 | 58.9 | - | - | - |
| Panyam et al. (2018) | 53.2 | 69.7 | 60.3 | 54.7 | 80.6 | 65.1 | 47.8 | 43.8 | 45.7 |
| Zheng et al. (2018) | 56.2 | 67.9 | 61.5 | - | - | - | - | - | - |

Table 1: Overall, intra- and inter-sentence pairs performance comparison with the state-of-the-art on the CDR test set. The methods below the double line take advantage of additional training data and/or incorporate external tools.

least one mention-level prediction indicates a relation then we predict the entity pair as related, similarly to Gu et al. (2017). All of the settings incorporate node type embeddings, contextual embeddings for MM edges and distance embeddings for MM and SS edges, unless otherwise stated.

## 4 Results

Table 1 depicts the performance of our proposed model on the CDR test set, in comparison with the state-of-the-art. We directly compare our model with models that do not incorporate external knowledge. Verga et al. (2018) and Nguyen and Verspoor (2018) consider a single pair per document, while Gu et al. (2017) develops separate models for intra- and inter-sentence pairs. As it can be observed, the proposed model outperforms the state-of-the-art in CDR dataset by 1.3 percentage points of overall performance. We also show the methods that take advantage of syntactic dependency tools. Li et al. (2016b) uses co-training with additional unlabeled training data. Our model performs significantly better on intra- and inter-sentential pairs, even compared to most of the models with external knowledge, except for Li et al. (2016b).

In addition, we report the performance of three baseline models. The EoG model outperforms all baselines for all pair types. In particular, for the inter-sentence pairs, performance significantly drops with a fully connected graph (`Full`) or without inference (`NoInf`). The former might indicate the existence of certain reasoning paths that should be followed in order to relate entities residing in different sentences. It is also important to note that the intra-sentence pairs substantially benefit from the document-level information, as EoG surpasses the performance of training on single sentences (`Sent`) by 3%. Finally, the performance drop in intra-sentence pairs, as a result of the inference algorithm removal (`NoInf`), suggests that multiple entity associations exist in sentences (Christopoulou et al., 2018). Their interactions can be beneficial in cases of lack of word context information.

| Model | Dev \| Test F1 (%) | | | | | |
|---|---|---|---|---|---|---|
| | Overall | | Intra | | Inter | |
| EoG | 78.7 | 81.5 | 82.5 | 85.2 | 48.8 | 50.0 |
| EoG (`Full`) | 78.6 | 80.8 | 82.4 | 84.1 | 52.3 | 54.7 |
| EoG (`NoInf`) | 71.8 | 74.6 | 76.8 | 79.1 | 45.5 | 49.3 |
| EoG (`Sent`) | 73.8 | 73.8 | 78.1 | 78.8 | - | - |

Table 2: Performance comparison on the GDA development and test sets.

We also apply our model on the distantly supervised GDA dataset. As shown in Table 2 results for intra-sentence pairs are consistent with the findings of the CDR dataset for both development and test sets. This indicates that document-level information is helpful. However, performance differs for inter-sentence pairs and in particular for the fully connected graph (`Full`) baseline. We partially attribute this behavior to the small number of inter-sentence pairs in the GDA dataset (only 13% compared to 30% in the CDR dataset) that results in inadequate learning patters for EoG. We leave further investigation as part of future work.

| Embeddings | F1 (%) | | |
|---|---|---|---|
| | Overall | Intra | Inter |
| EoG (PubMed) | 63.62 | 68.25 | 50.94 |
| EoG (GloVe) | 63.01 | 67.52 | 50.26 |
| EoG (random) | 61.41 | 66.80 | 46.51 |

Table 3: Performance of EoG on the CDR test set with different pre-trained word embeddings.

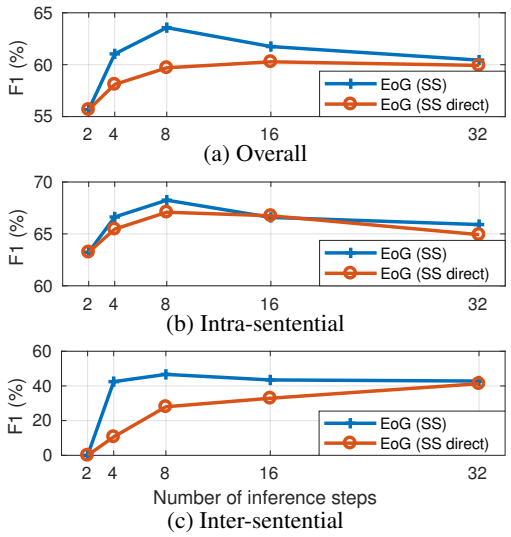


Figure 3: Performance as a function of the number of inference steps when using direct ($SS_{direct}$) or direct and indirect (SS) sentence-to-sentence edges, on the CDR development set.

## 5 Analysis & Discussion

We first analyse the performance of our main model (EoG) using different pre-trained word embeddings. Table 3 shows the performance difference between domain-specific (PubMed) (Chiu et al., 2016), general-domain (GloVe) (Pennington et al., 2014) and randomly initialized (random) word embeddings. As observed, our proposed model performs consistently with both in-domain and out-of-domain pre-trained word embeddings. The low performance of random embeddings is due to the small size of the dataset, which results in lower quality embeddings.

For further analysis, we choose the CDR dataset as it is manually annotated. To better analyse the behaviour of our model, we conduct analysis on the effect of direct and indirect sentence-to-sentence edges as a function of the inference steps. Figures 3a, 3b and 3c illustrate the performance of both graphs for overall, intra- and inter-sentence pairs respectively.

The first observation is that usage of direct edges only, reduces the overall performance almost by 4%, for inference step $l = 8$. This drop mostly affects inter-sentence pairs, where a 18% point drop is observed. In fact, ordered edges ($SS_{direct}$) need longer inference to perform better, in comparison with additional indirect edges (SS) for which less steps are required. The superiority of SS edges, for all inference steps, compared to $SS_{direct}$ edges on inter-sentence pairs detection, indicates that in a narrative, some intermediate information is not important. The observation that indirect edges perform slightly better than direct for intra-sentence pairs ($l \leq 16$) agrees with the results of Table 1 where we showed that inter-sentence information can act as complementary evidence for intra-sentence pairs.

| Edge Types | F1 (%) | | |
|---|---|---|---|
| | Overall | Intra | Inter |
| EE | 55.14 | 61.31 | 40.34 |
| EoG | 63.57 | 68.25 | 46.68 |
| −MM | 62.77 | 67.93 | 46.65 |
| −ME | 61.57 | 66.39 | 45.40 |
| −MS | 62.92 | 67.55 | 44.74 |
| −ES | 61.41 | 66.44 | 43.04 |
| $-SS_{indirect}$ | 59.70 | 67.09 | 28.00 |
| −SS | 57.41 | 65.45 | 1.59 |
| −MM, ME, MS | 60.46 | 66.07 | 39.56 |
| −ES, MS, SS | 56.86 | 64.63 | 0.00 |

Table 4: Ablation analysis for different edge and node types on the CDR development set.

We additionally conduct ablation analysis on the graph edges and nodes, as shown in Table 4. Usage of EE edges only results in poor performance across pairs. Removal of MM and ME edges does not significantly affect the performance as ES edges can replace their impact. Complete removal of connections to M nodes results in low inter-sentence performance. This behaviour pinpoints the importance of some local dependencies in identifying cross-sentence relations.

Removal of ES edges reduces the performance of all pairs, as encoding of EE edges becomes more difficult[4]. We further observe very poor identification of inter-sentence pairs without sentence-to-sentence connections. This is complementary with the inability of the model to identify any inter-sentence pairs without connections to S nodes. In this scenario, we enable identification of pairs across sentences only through MM and ME edges, as shown in Figure 4a. In the CDR dataset,

[4]Length 3 (E-M-M-E) for intra- and length 5 (E-M-S-S-M-E) for inter-sentence pairs.

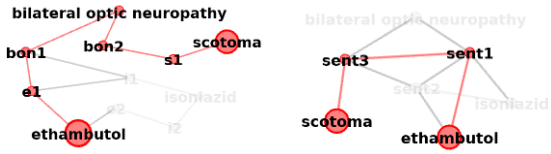


(a) MM, ME edges (b) ES, SS edges

Figure 4: Relation paths with different types of edges.

| Model | F1 (%) | | |
|---|---|---|---|
| | Overall | Intra | Inter |
| EoG | 63.57 | 68.25 | 46.68 |
| −node types (T) | 62.31 | 67.50 | 44.80 |
| −MM context (C) | 62.88 | 67.67 | 46.59 |
| −distances (D) | 62.53 | 68.00 | 41.53 |
| −T,C,D | 63.10 | 68.44 | 43.48 |

Table 5: Ablation analysis of edge enhancements on the CDR development set.

78% of inter-sentential pairs have at least one argument that is mentioned only once in the document. The identification of these pairs, without S nodes, requires very long inference paths[5]. As shown in Figure 4b, the introduction of S nodes results in a path with half the length, which we expect to better represent the relation. Longer inference representations are much weaker than shorter ones. This suggests that the inference mechanism has limited capability in identifying very complex associations.

We then investigate the additional enhancements of the graph edges in Table 5. In general, intra-sentence pairs are not affected by these settings. However, for inter-sentence pairs, removal of node type embeddings and distance embeddings results in a 2% and 5% drop in terms of F1-score. These results indicate that the interactions between different elements in a document, along with the distance between sentences and mentions, play an important role in inter-sentence pair inference. Removing all of these settings does not perform worse than removing one of them, which might indicate model overfitting. We plan to further investigate this as part of future work.

We examine the performance of different models on inter-sentence pairs, based on their sentence-level distances. Figure 5 illustrates that for long-distanced pairs, EoG has lower performance, indicating the difficulty in predicting them and a possible requirement for other, latent document-level information (EoG (Full)).

[5]Minimum inference length 6 (E-M-M-E-M-M-E).

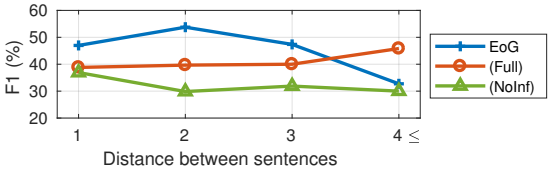


Figure 5: Performance of inter-sentence pairs on the CDR development set as a function of their sentence distance.

| |
|---|
| Following short exposure to oral **prednisone** [...]. Both presented in the emergency room with profound **coma**, **hypotension**, severe **hyperglycemia**, and acidosis. |
| The etiology of **pyeloureteritis cystica** has long been [...] The *disease* occurred subsequent to the initiation of **heparin** therapy [...] |
| Time trends in **warfarin**-associated ***hemorrhage***. [...] The proportion of patients with major and ***intracranial bleeding*** increased [...] |

Table 6: Inter-sentence pairs from the CDR development set that EoG fails to detect.

As final analysis, we investigate some of the cases where the graph models are unable to identify inter-sentence related pairs. For this purpose, we randomly check some of the common false negative errors among the EoG models. We identify three frequent cases of errors, as shown in Table 6. In the first case, when multiple entities reside in the same sentence and are connected with conjunctions (e.g., 'and') or commas, the model often failed to find associations with all of them. The second error derives from missing coreference connections. For instance, *pyeloureteritis cystica* is referred to as *disease*. Although our model cannot directly create these edges, S nodes potentially simulate such links, by encoding the co-referring entities into the sentence representation. Finally, incomplete entity linking results into additional model errors. For instance, in the third example, *hemorrhage* and *intracranial bleeding* are synonymous terms. However, they are assigned different KB IDs, hence treated as different entities. The model can find the intra-sentential relation but not the inter-sentential one.

## 6 Related Work

Traditional approaches focus on intra-sentence supervised RE, utilising CNN or RNN, ignoring multiple entities in a sentence (Zeng et al., 2014; Nguyen and Grishman, 2015) as well as incorporating external syntactic tools (Miwa and Bansal,

2016; Zhang et al., 2018). Christopoulou et al. (2018) considered intra-sentence entity interactions without domain dependencies by modelling long dependencies between the entities of a sentence.

Other approaches deal with distantly-supervised datasets but are also limited to intra-sentential relations. They utilise Piecewise Convolutional Neural Networks (PCNN) (Zeng et al., 2015), attention mechanisms (Lin et al., 2016; Zhou et al., 2018), entity descriptors (Jiang et al., 2016) and graph CNNs (Vashishth et al., 2018) to perform MIL on bags-of-sentences that contain multiple mentions of an entity pair. Recently, Zeng et al. (2017) proposed a method for extracting paths between entities using the target entities' mentions in several different sentences (in possibly different documents) as intermediate connectors. They allow mention-mention edges only if these mentions belong to the same entity and consider that a single mention pair exists in a sentence. On the contrary, we not only allow interactions between all mentions in the same sentence, but also consider multiple edges between mentions, entities and sentences in a document.

Current approaches that try to deal with document-level RE are mostly graph-based. Quirk and Poon (2017) introduced the notion of a document graph, where nodes are words and edges represent intra- and inter-sentential relations between the words. They connected words with different dependency edges and trained a binary logistic regression classifier. They evaluated their model on distantly supervised full-text articles from PubMed for Gene-Drug associations, restricting pairs within a window of consecutive sentences. Following this work, other approaches incorporated graphical models for document-level RE such as graph LSTM (Peng et al., 2017), graph CNN (Song et al., 2018) or RNNs on dependency tree structures (Gupta et al., 2019). Recently, Jia et al. (2019) improved $n$-ary RE using information from multiple sentences and paragraphs in a document. Similar to our approach, they choose to directly classify concept-level pairs rather than multiple mention-level pairs. Although they consider sub-relations to model related tuples, they ignore interactions with other entities outside of the target tuple in the discourse units.

Non-graph-based approaches utilise different intra- and inter-sentence models and merge the resulted predictions (Gu et al., 2016, 2017). Other approaches extract document-level representations for each candidate entity pair (Zheng et al., 2018; Li et al., 2018; Wu et al., 2019), or use syntactic dependency structures (Zhou et al., 2016; Peng et al., 2016). Verga et al. (2018) proposed a Transformer-based model for document-level relation extraction with multi-instance learning, merging multiple mention pairs. Nguyen and Verspoor (2018) used a CNN with additional character-level embeddings. Singh and Bhatia (2019) also utilised Transformer and connected two target entities by combining them directly and via a contextual token. However, they consider a single target entity pair per document.

## 7 Conclusion

We presented a novel edge-oriented graph neural model for document-level relation extraction using multi-instance learning. The proposed model constructs a document-level graph with heterogeneous types of nodes and edges, modelling intra- and inter-sentence pairs simultaneously with an iterative algorithm over the graph edges. To the best of our knowledge, this is the first approach to utilise an edge-oriented model for document-level RE.

Analysis on intra- and inter-sentence pairs indicated that the proposed, partially-connected, document graph structure can effectively encode dependencies between document elements. Additionally, we deduce that document-level information can contribute to the identification of intra-sentence pairs leading to higher precision and F1-score.

As future work, we plan to improve the inference mechanism and potentially incorporate additional information in the document-graph structure. We hope that this study will inspire the community to further investigate the usage of edge-oriented models on RE and other related tasks.

## Acknowledgments

The authors would like to thank the anonymous reviewers for their comments and AIST/AIRC for providing the computational resources for this study. Research was funded by the University of Manchester James Elson Studentship Award and the BBSRC Japan Partnering Award BB/P025684/1.

## References


Michele Banko, Michael J Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *Proceedings of the International Joint Conference on Artifical Intelligence*, pages 2670–2676. Morgan Kaufmann Publishers Inc.

Karsten M Borgwardt and Hans-Peter Kriegel. 2005. Shortest-path kernels on graphs. In *Proceedings of the IEEE International Conference on Data Mining*, pages 74–81. IEEE Computer Society.

Billy Chiu, Gamal Crichton, Anna Korhonen, and Sampo Pyysalo. 2016. How to train good word embeddings for biomedical nlp. In *Proceedings of the BioNLP workshop*, pages 166–174.

Fenia Christopoulou, Makoto Miwa, and Sophia Ananiadou. 2018. A walk-based model on entity graphs for relation extraction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (Volume 2)*, pages 81–88. Association for Computational Linguistics.

Jinghang Gu, Longhua Qian, and Guodong Zhou. 2016. Chemical-induced disease relation extraction with various linguistic features. *Database*.

Jinghang Gu, Fuqing Sun, Longhua Qian, and Guodong Zhou. 2017. Chemical-induced disease relation extraction via convolutional neural network. *Database*.

Pankaj Gupta, Subburam Rajaram, Hinrich Schtze, and Thomas Runkler. 2019. Neural relation extraction within and across sentence boundaries. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6513–6520.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Robin Jia, Cliff Wong, and Hoifung Poon. 2019. Document-level n-ary relation extraction with multiscale representation learning. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1)*, pages 3693–3704. Association for Computational Linguistics.

Xiaotian Jiang, Quan Wang, Peng Li, and Bin Wang. 2016. Relation extraction with multi-instance multi-label convolutional neural networks. In *Proceedings of the International Conference on Computational Linguistics: Technical Papers*, pages 1471–1480. The COLING Organizing Committee.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*.

Haodi Li, Ming Yang, Qingcai Chen, Buzhou Tang, Xiaolong Wang, and Jun Yan. 2018. Chemical-induced disease extraction via recurrent piecewise convolutional neural networks. *BMC medical informatics and decision making*, 18(2):60.

Jiao Li, Yueping Sun, Robin J Johnson, Daniela Sciaky, Chih-Hsuan Wei, Robert Leaman, Allan Peter Davis, Carolyn J Mattingly, Thomas C Wiegers, and Zhiyong Lu. 2016a. Biocreative v cdr task corpus: a resource for chemical disease relation extraction. *Database*.

Zhiheng Li, Zhihao Yang, Hongfei Lin, Jian Wang, Yingyi Gui, Yin Zhang, and Lei Wang. 2016b. CIDextractor: A chemical-induced disease relation extraction system for biomedical literature. In *IEEE International Conference on Bioinformatics and Biomedicine*, pages 994–1001. IEEE.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (Volume 1)*, pages 2124–2133.

Yi Luan, Dave Wadden, Luheng He, Amy Shah, Mari Ostendorf, and Hannaneh Hajishirzi. 2019. A general framework for information extraction using dynamic span graphs. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1)*, pages 3036–3046. Association for Computational Linguistics.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP (Volume 2)*, pages 1003–1011. Association for Computational Linguistics.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using LSTMs on sequences and tree structures. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (Volume 1)*, pages 1105–1116. Association for Computational Linguistics.

Dat Quoc Nguyen and Karin Verspoor. 2018. Convolutional neural networks for chemical-disease relation extraction are improved with character-based word embeddings. In *Proceedings of the BioNLP workshop*, pages 129–136. Association for Computational Linguistics.

Thien Huu Nguyen and Ralph Grishman. 2015. Relation extraction: Perspective from convolutional neural networks. In *Proceedings of the Workshop on Vector Space Modeling for Natural Language Processing*, pages 39–48.

Nagesh C Panyam, Karin Verspoor, Trevor Cohn, and Kotagiri Ramamohanarao. 2018. Exploiting graph kernels for high performance biomedical relation extraction. *Journal of biomedical semantics*, 9(1):7.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NIPS-W*.

Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen-tau Yih. 2017. Cross-sentence n-ary relation extraction with graph lstms. *Transactions of the Association for Computational Linguistics*, 5:101–115.

Yifan Peng, Chih-Hsuan Wei, and Zhiyong Lu. 2016. Improving chemical disease relation extraction with rich features and weakly labeled data. *Journal of cheminformatics*, 8(1):53.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the Empirical Methods in Natural Language Processing*, pages 1532–1543. Association for Computational Linguistics.

Janet Piñero, Àlex Bravo, Núria Queralt-Rosinach, Alba Gutiérrez-Sacristán, Jordi Deu-Pons, Emilio Centeno, Javier García-García, Ferran Sanz, and Laura I Furlong. 2016. Disgenet: a comprehensive platform integrating information on human disease-associated genes and variants. *Nucleic acids research*, pages D833–D839.

Chris Quirk and Hoifung Poon. 2017. Distant supervision for relation extraction beyond the sentence boundary. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics (Volume 1)*, pages 1171–1182. Association for Computational Linguistics.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Wei Shen, Jianyong Wang, and Jiawei Han. 2014. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Transactions on Knowledge and Data Engineering*, 27(2):443–460.

Gaurav Singh and Parminder Bhatia. 2019. Relation extraction using explicit context conditioning. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1442–1447. Association for Computational Linguistics.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. N-ary relation extraction using graph-state lstm. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2226–2235. Association for Computational Linguistics.

Shikhar Vashishth, Rishabh Joshi, Sai Suman Prayaga, Chiranjib Bhattacharyya, and Partha Talukdar. 2018. Reside: Improving distantly-supervised neural relation extraction using side information. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1257–1266. Association for Computational Linguistics.

Patrick Verga, Emma Strubell, and Andrew McCallum. 2018. Simultaneously self-attending to all mentions for full-abstract biological relation extraction. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1)*, pages 872–884. Association for Computational Linguistics.

Linlin Wang, Zhu Cao, Gerard de Melo, and Zhiyuan Liu. 2016. Relation classification via multi-level attention CNNs. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (Volume 1)*, pages 1298–1307. Association for Computational Linguistics.

Ye Wu, Ruibang Luo, Henry CM Leung, Hing-Fung Ting, and Tak-Wah Lam. 2019. Renet: A deep learning approach for extracting gene-disease associations from literature. In *International Conference on Research in Computational Molecular Biology*, pages 272–284. Springer.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, pages 1785–1794. Association for Computational Linguistics.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1753–1762.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of the International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344. Dublin City University and Association for Computational Linguistics.

Wenyuan Zeng, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2017. Incorporating relation paths in neural relation extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1768–1777. Association for Computational Linguistics.

Yuhao Zhang, Peng Qi, and Christopher D Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2205–2215. Association for Computational Linguistics.

Wei Zheng, Hongfei Lin, Zhiheng Li, Xiaoxia Liu, Zhengguang Li, Bo Xu, Yijia Zhang, Zhihao Yang, and Jian Wang. 2018. An effective neural model extracting document level chemical-induced disease relations from biomedical literature. *Journal of biomedical informatics*, 83:1–9.

Huiwei Zhou, Huijie Deng, Long Chen, Yunlong Yang, Chen Jia, and Degen Huang. 2016. Exploiting syntactic and semantics information for chemical–disease relation extraction. *Database*.

Peng Zhou, Jiaming Xu, Zhenyu Qi, Hongyun Bao, Zhineng Chen, and Bo Xu. 2018. Distant supervision for relation extraction with hierarchical selective attention. *Neural Networks*, 108:240–247.

## A Datasets

In Tables 7-8 we summarise the statistics for the CDR and GDA datasets, respectively. For all datasets, we used the GENIA Sentence Splitter[6] and GENIA Tagger[7] for sentence splitting and word tokenisation respectively. We additionally removed mentions in the given abstracts that were not grounded to a Knowledge Base ID (ID equal to $-1$).

Due to the small size of the CDR dataset, some approaches create a new split from the union of train and development sets (Verga et al., 2018; Zhou et al., 2018). We select to merge the train and development sets and re-train our model on their entire union for evaluation on the test set following Lin et al. (2016) and Zhou et al. (2016). To compare with related work, we followed Verga et al. (2018) and Gu et al. (2016) and ignored non-related pairs that correspond to general concepts (MeSH vocabulary hypernym filtering).

## B Hyper-parameter Setting

We used the development set to identify the stopping training epoch and tune the number of inference iterations. Except from these parameters, all experiments used the same hyperparameters, with a fixed initialisation seed. For the CDR dataset EoG, (`Full`) and (`Sent`) models performed best with $l = 8, 2, 4$ inference steps, respectively. The chosen batchsize was equal to 2. For the GDA dataset, EoG and EoG (`Full`) performed best with $l = 16$ and EoG (`Sent`) with $l = 8$ inference steps. The chosen batchsize was equal to 3. For all experiments performance was measured in terms of micro precision (P), recall (R) and F1-score (F1). We list the hyper-parameters used to train the proposed model in Table 9.

[6] http://www.nactem.ac.uk/y-matsu/geniass/

[7] http://www.nactem.ac.uk/GENIA/tagger/

| | Train | Dev | Test |
|---|---|---|---|
| Documents | 500 | 500 | 500 |
| Positive pairs | 1,038 | 1,012 | 1,066 |
| Intra | 754 | 766 | 747 |
| Inter | 284 | 246 | 319 |
| Negative pairs | 4,202 | 4,075 | 4,138 |
| Entities | | | |
| Chemical | 1,467 | 1,507 | 1,434 |
| Disease | 1,965 | 1,864 | 1,988 |
| Mentions | | | |
| Chemical | 5,162 | 5,307 | 5,370 |
| Disease | 4,252 | 4,328 | 4,430 |

Table 7: CDR (BioCreative V) dataset statistics.

| | Train | Dev | Test |
|---|---|---|---|
| Documents | 23,353 | 5,839 | 1,000 |
| Positive pairs | 36,079 | 8,762 | 1,502 |
| Intra | 30,905 | 7,558 | 1,305 |
| Inter | 5,174 | 1,204 | 197 |
| Negative pairs | 96,399 | 24,362 | 3,720 |
| Entities | | | |
| Gene | 46,151 | 11,406 | 1,903 |
| Disease | 67,257 | 16,703 | 2,778 |
| Mentions | | | |
| Gene | 205,457 | 51,410 | 8,404 |
| Disease | 226,015 | 56,318 | 9,524 |

Table 8: GDA (DisGeNet) dataset statistics.

| Parameter | Value |
|---|---|
| Batch size | $[2, 3]$ |
| Learning rate | 0.002 |
| Gradient clipping | 10 |
| Early stop patience | 10 |
| Regularization | $10^{-4}$ |
| Dropout word embedding layer | 0.5 |
| Dropout classification layer | 0.3 |
| Word dimension | 200 |
| Node type dimension | 10 |
| Distance dimension | 10 |
| Edge dimension | 100 |
| $\beta$ | 0.8 |
| Optimizer | Adam |
| Inference iterations | $[0, 5]$ |

Table 9: Hyper-parameter values used in the reported experiments.

# Semi-supervised Text Style Transfer: Cross Projection in Latent Space

**Mingyue Shang[1]*, Piji Li[2], Zhenxin Fu[1], Lidong Bing[4]**
**Dongyan Zhao[1,3], Shuming Shi[2], Rui Yan[1,3]†**
[1]Wangxuan Institute of Computer Technology, Peking University, China
[2]Tencent AI Lab, Shenzhen, China, [3]Center for Data Science, China
[4]R&D Center Singapore, Machine Intelligence Technology, Alibaba DAMO Academy
{shangmy, fuzhenxin, zhaodongyan, ruiyan}@pku.edu.cn
{pijili, shumingshi}@tencent.com, l.bing@alibaba-inc.com

## Abstract

Text style transfer task requires the model to transfer a sentence of one style to another style while retaining its original content meaning, which is a challenging problem that has long suffered from the shortage of parallel data. In this paper, we first propose a semi-supervised text style transfer model that combines the small-scale parallel data with the large-scale nonparallel data. With these two types of training data, we introduce a projection function between the latent space of different styles and design two constraints to train it. We also introduce two other simple but effective semi-supervised methods to compare with. To evaluate the performance of the proposed methods, we build and release a novel style transfer dataset that alters sentences between the style of ancient Chinese poem and the modern Chinese.

## 1 Introduction

Recently, the natural language generation (NLG) tasks have been attracting the growing attention of researchers, including response generation (Vinyals and Le, 2015), machine translation (Bahdanau et al., 2014), automatic summarization (Chopra et al., 2016), question generation (Gao et al., 2019), etc. Among these generation tasks, one interesting but challenging problem is text style transfer (Shen et al., 2017; Fu et al., 2018; Logeswaran et al., 2018). Given a sentence from one style domain, a style transfer system is required to convert it to another style domain as well as keeping its content meaning unchanged. As a fundamental attribute of text, style can have a broad and ambiguous scope, such as ancient poetry style v.s. modern language style and positive sentiment v.s. negative sentiment.

Building such a style transfer model has long suffered from the shortage of parallel training data, since constructing the parallel corpus that could align the content meaning of different styles is costly and laborious, which makes it difficult to train in a supervised way. Even some parallel corpora are built, they are still in a deficient scale for neural network based models. To tackle this issue, previous works utilize nonparallel data to train the model in an unsupervised way. One commonly used method is disentangling the style and content from the source sentence (John et al., 2018; Shen et al., 2017; Hu et al., 2017). For the input, they learn the representations of styles and style-independent content, expecting the later only keeps the content information. Then the content-only representation is coupled with the representation of a style that differs from the input to produce a style-transferred sentence. The crucial part of such methods is that the encoder should accurately disentangle the style and content information. However, Lample et al. (2019) illustrated that disentanglement is not a facile thing and the existing methods are not adequate to learn the style-independent representations.

Considering the above-discussed issues, in this paper, instead of disentangling the input, we propose a differentiable encoder-decoder based model that contains a projection layer to build a bridge between the latent spaces of different styles. Concretely, for texts in different styles, the encoder converts them into latent representations in different latent spaces. We introduce a projection function between two latent spaces that projects the representation in the latent space of one style to another style. Then the decoder generates an output using the projected representation.

In the majority of cases, nonparallel corpora of different styles are accessible. Based on these datasets, it is feasible to build small-scale paral-

*This work was done when Mingyue Shang was an intern at Tencent AI Lab.

†Corresponding author: Rui Yan (ruiyan@pku.edu.cn)



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4937–4946, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

lel datasets. Therefore, we design two kinds of objective functions for our model so that it could be trained under both supervised settings and unsupervised settings. With the parallel data, the model learns the projection relationship and the standard representation of the target latent space from ground-truth. Without the parallel data, we train the model by back-projection between the source and target latent space so that it could learn from itself. During training, we incorporate these two kinds of signals to train the model in a semi-supervised way.

We conduct experiments on an English formality transfer task that alters the sentence between formal and informal styles, and a Chinese literal style transfer task that alters between the ancient Chinese poem sentences and modern Chinese sentences. We evaluate the performance of models from the degree of content preservation, the accuracy of the transferred styles, and the fluency of sentences using both automatic metrics and human annotations. Experimental results show that our proposed semi-supervised method can generate more preferred output.

In summary, our contributions are manifolds:

- We designed a semi-supervised model that crossly projects the latent spaces of two styles to each other. In addition, this model is flexible in alternating the training mode between supervised and unsupervised.
- We introduce another two semi-supervised methods that are simple but effective to leverage both the nonparallel and parallel data.
- We build a small-scale parallel dataset that contains ancient Chinese poem style and modern Chinese style sentences. We also collect two large nonparallel datasets of these styles.[1]

## 2 Related Works

Recently, text style transfer has stimulated great interests of researchers from the area of neural language processing and some encouraging results are obtained (Shen et al., 2017; Rao and Tetreault, 2018; Prabhumoye et al., 2018; Hu et al., 2017; Jin et al., 2019). In the primary stage, due to the lacking of parallel corpus, most of the methods employ unsupervised learning paradigm to conduct the semantic modeling and transfer.

[1]Download link: https://tinyurl.com/yyc8zkqg

**Unsupervised Learning Methods.** Mueller et al. (2017) modify the latent variables of sentences in a certain direction guided by a classifier to generate the sentence with classifier favored style. Hu et al. (2017) employ variational autoencoders (Kingma and Welling, 2013) to conduct the style latent variable learning and design a strategy to disentangle the variables for content and style. Shen et al. (2017) first map the text corpora belonging to different styles to their own space respectively, and leverages the alignment of latent representations from different styles to perform style transfer. Chen et al. (2019) propose to extract and control the style of the image caption through domain layer normalization. Prabhumoye et al. (2018) and Zhang et al. (2018b) employ the back-translation mechanism to ensure that the input from the source style can be reconstructed from the transferred result in the target style. Liao et al. (2018) associate the style latent variable with a numeric discrete or continues numeric value and can generate sentences controlled by this value. Among them, many use the adversarial training mechanism (Goodfellow et al., 2014) to improve the performance of the basic models (Shen et al., 2017; Zhao et al., 2018).

To sum up, most of the existing unsupervised frameworks on the text style transfer focus on getting the disentangled representations of style and content. However, Lample et al. (2019) illustrated that the disentanglement not adequate to learn the style-independent representations, thus the quality of the transferred text is not guaranteed.

**Supervised Learning Methods.** Fortunately, Rao and Tetreault (2018) release a high-quality parallel dataset - GYAFC, which consists of two domains for Formality style transfer: Entertainment & Music and Family & Relationships. The quantity of the dataset is sufficient to train an attention-based sequence-to-sequence framework.

Nevertheless, building parallel corpus that could align the content meaning of different styles is costly and laborious, and it is impossible to build the parallel corpus for all the domains.

## 3 Task Formulation

We assume two nonparallel datasets $A$ and $B$ that are sentences in two different styles, denoted as style $s^a$ and style $s^b$, and a parallel dataset $P$ that contains the pairs of sentences with two styles. The size of the three

datasets are denoted as $|A|$, $|B|$ and $|P|$, respectively. As the nonparallel data is abundant but the parallel data is limited, size $|A|$, $|B| \gg |P|$. Formally, two nonparallel datasets are denoted as $A = \{a_1, a_2, \cdots, a_{|A|}\}$ and $B = \{b_1, b_2, \cdots, b_{|B|}\}$, where $a_i$ and $b_i$ refer to the $i$-th sentences of $s^a$ style and $s^b$ style, respectively. It should be clear that as $A$ and $B$ are nonparallel datasets, the sentences in the two datasets are not aligned with each other, which means that $a_i$ and $b_i$ are not required to have the same or similar content meaning though they have the same subscript. The parallel dataset is denoted as $P = \{(a_1^p, b_1^p), (a_2^p, b_2^p), \cdots, (a_{|P|}^p, b_{|P|}^p)\}$, where $(a_i^p, b_i^p)$ is the $i$-th pair of sentences that have the same content meaning but expressed in style $s^a$ and $s^b$ separately.

Due to the situation that the limited parallel data is deficient for neural network based model to train, our goal is to train a model in a semi-supervised way that could leverage the large volumes of nonparallel data to improve the performance. Given a sentence of the source style as input, the model learns to generate a target style output which preserves the meaning of the input to the greatest extent. In this paper, the style transfer process could be exerted in two directions, from $s^a$ to $s^b$ as well as from $s^b$ to $s^a$.

## 4 Basic Model Architecture

The text style transfer task could be interpreted as transforming a sequence of source style words to a sequence of target style words, which makes the sequence-to-sequence framework a suitable architecture. In this section, we describe the formulation of sequence-to-sequence (S2S) (Sutskever et al., 2014) model, which contains an encoder and a decoder, and our proposed models are built based on such an architecture.

Given an input, the encoder first converts it into an intermediate vector, then the decoder takes the intermediate representation as input to generate a target output. In this paper, we implement the encoder by a bi-directional Long Short-Term Memory (BiLSTM) (Hochreiter and Schmidhuber, 1997) and the decoder by a one layer LSTM.

Formally, with an input sentence $x = \{w_1^x, \cdots, w_T^x\}$ of length $T$, and a target sentence $y = \{w_1^y, \cdots, w_{T'}^y\}$ of length $T'$, where $w_i$ is the embedding of the $i$-th word, the probability of generating the target sentence $y$ given $x$ is defined as:

$$p(y|x) = \prod_{i=1}^{T'} p\left(w_i^y | x, w_1^y, \ldots, w_{i-1}^y\right) \quad (1)$$

In the training process, the encoder first encodes $x$ into an intermediate representation $c_x$. More specifically, at each time step $t$, the encoder produces a hidden state vector $h_t = [\overrightarrow{h}_t; \overleftarrow{h}_t]$, where the $\overrightarrow{h}_t$ and $\overleftarrow{h}_t$ represents the forward and backward hidden state, respectively, and $[;]$ means concatenation. The intermediate vector is formed by the concatenation of the forward and backward last hidden states, denoted as $c_x = Enc(x) = [\overrightarrow{h}_T; \overleftarrow{h}_1]$. Then this vector is fed to the decoder to generate a target sentence step by step as formulated in Equation 1.

Such neural network based models always have a large amount of parameters which empower the model with the ability to fit complex datasets. However, when trained with limited data, the model may get so closely fitted to the training data that loses the power of generalization, and thus performances poorly on new data.

## 5 Cross Projection in Latent Space

In this paper, we propose a semi-supervised style transfer method named cross projection in latent space (CPLS) to leverage the large volumes of nonparallel data as well as the limited parallel data. Based on the S2S architecture, we consider the representations in the latent space, and propose to establish projection relations between different latent spaces. To be specific, in the S2S architecture, the encoder converting the input into an intermediate vector is a process of extracting the semantic information of the input, including the style and the content information. Thus the intermediate vectors could be seen as the compressed representations in the latent space of the inputs. Since texts in different styles are in different latent spaces, we introduce a projection function to project the intermediate vector from one latent space to another. To combine the nonparallel data and parallel data in training, we design two kinds of constraints for the projection functions.

Concretely, we first train an auto-encoder for each style to learn the standard latent spaces of the style. After that, we train the projection functions which are exerted on latent vectors to establish projection relations between different latent spaces. We design a cross projection strategy and

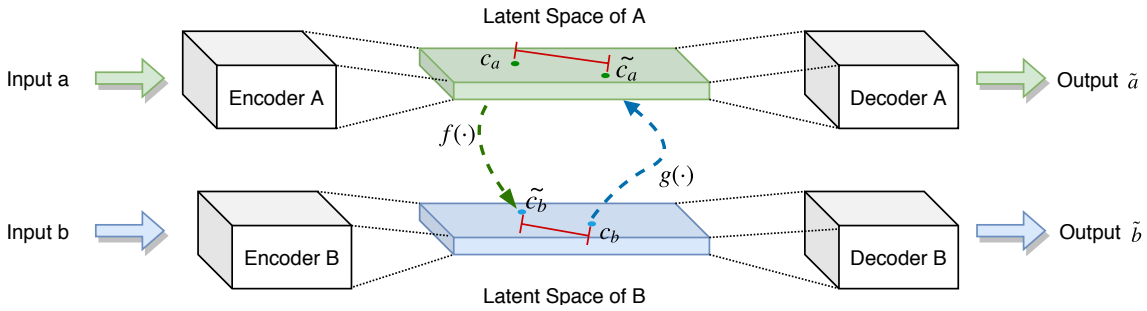


Figure 1: The process of training the sequence-to-sequence model. Take a sentence pair $(a, b)$ for example. After $a$ is encoded into $c_a$ by the encoder A, the projection function $f(\cdot)$ (shown by the dotted green line) projects it to the latent space of $B$, which is then fed to the decoder B to generate the output $\tilde{a}$. The encoder B gives the latent representation of $b$. One of the constraints is computed from the distance of $\tilde{c}_b$ and $c_b$ (shown by the red lines) and another is the NLL-loss of $\tilde{b}$ and $b$. The dotted blue line refers to the projection function from the space of $B$ to $A$.

a cycle projection strategy to utilize the parallel data and nonparallel data. These two strategies are exerted iteratively in the training process, thus our model is compatible with the two types of training data. The following subsections give the details of the modules and the strategies of model training.

### 5.1 Denoising Auto-Encoder

To learn the latent space representations of styles, we train an auto-encoder for each style of text through reconstructing the input, which is composed of an encoder and a decoder. The encoder takes a sentence as input and maps the sentence to a latent vector representation, and the decoder reconstructs the sentence based on the vector. But a common pitfall is that the decoder may simply copy every token from the input, making the encoder lose the ability to extract the features. To alleviate this issue, we adopt the denoising auto-encoder (DAE) following the previous work (Lample et al., 2019) that tries to reconstruct the corrupted input. Specifically, we randomly shuffle a small portion of words in the original sentence to the corrupted input which are then fed to the denoising auto-encoder. For example, the original input is "Listen to your heart and your mind.", then the corrupted version could be "Listen your to and heart your mind.". The decoder is required to reconstruct the original sentence.

The training of a DAE model relies on the corpus of one style only, thus we train each DAE model using the nonparallel corpus of each style. Formally, the encoder and decoder of style $s^a$ and $s^b$ are referred as $\text{Enc}^{(a)}$ - $\text{Dec}^{(a)}$ and $\text{Enc}^{(b)}$ - $\text{Dec}^{(b)}$. Given sentences $a$ and $b$ from two styles, the corresponding encoder takes the sentence as input and converts it to the latent vector $c_a$ and $c_b$, respectively, by which each encoder therefore constructs the latent space for the corresponding style.

### 5.2 Cross Projection

To perform the style transfer task, we establish a projection relationship between latent spaces and cross link the encoder and the decoder from different styles. Take the transfer from style $s^a$ to $s^b$ for example. Given a sentence $a$ that is required to transferred to $b$, we cross link the $\text{Enc}^{(a)}$ as the encoder and $\text{Dec}^{(b)}$ as the decoder to form the style transfer model. However, considering the DAE models for different styles are trained respectively, therefor the latent vector spaces are usually different. The latent vector produced by $\text{Enc}^{(a)}$ is the representation on the latent space of style $s^a$ while the $\text{Dec}^{(b)}$ relies on the information and features from the latent space of style $s^b$. Therefore, we introduce a projection function to project the latent vector from the latent space of $s^a$ to $s^b$.

Concretely, after we get the $c_a$ from $\text{Enc}^{(a)}$, a projection function $f(\cdot)$ is employed to project $c_a$ from the latent space of style $s^a$ to the latent space of $s^b$, denoted as $\tilde{c}_b = f(c_a)$. Then the decoder $\text{Dec}^{(b)}$ takes the the projected vector $\tilde{c}_b$ as input and generates a sentence $\tilde{b}$ of style $s^b$ base on the prediction probability, denoted as $p_b(\tilde{b}|\tilde{c}_b)$.

It is worth noting that we have only exploited the nonparallel corpus for the style transfer up to now. Recall that our framework can employ both the parallel corpus and nonparallel corpus for the model training. With parallel corpus, we design the cross projection strategy. When the input $a$ is accompanied with a ground-truth $b$, we can get the standard latent representation of $b$ by the $\text{Enc}^{(b)}$, denoted as $c_b$. In order to align the latent vectors from different spaces, we design the constraints to

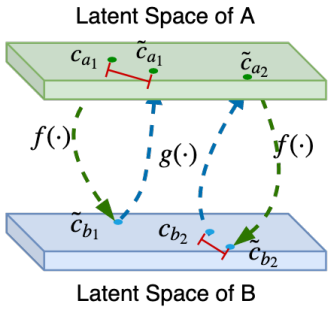


Figure 2: The cycle projection process between the latent space of $A$ and $B$ when training the denoising auto-encoders. The encoders and decoders are the same as illustrated in Figure 1 and are omitted in this figure.

train the $f(\cdot)$ from two aspects: the distance between $\tilde{c}_b$ and $c_b$ in the latent space should be close; the generated sentence $\tilde{b}$ should be similar with $b$. Then we define two losses as follows:

$$\tilde{c}_b = f(\text{Enc}^{(a)}(a)) \quad (2)$$
$$l_{s_1} = \|\tilde{c}_b - c_b\| \quad (3)$$
$$l_{s_2} = -\log p(b|\tilde{c}_b) \quad (4)$$
$$l_s = \alpha * l_{s_1} + \beta * l_{s_2} \quad (5)$$

where $l_{s_1}$ measures the Euclidean distance between $\tilde{c}_b$ and $c_b$ and $l_{s_2}$ is the negative log-likelihood (NLL) loss given $a$ as input and $b$ as the ground-truth. $\alpha$ and $\beta$ is the hyper-parameters that control the weight of $l_{s_1}$ and $l_{s_2}$.

Figure 1 shows the training process of the cross projection. Similarly, to transfer the style of text from $s^b$ to $s^a$, we cross link $\text{Enc}^{(b)}$ and $\text{Dec}^{(a)}$, and the projection function to project $c_b$ to the latent space of $s^a$ is denoted as $g(\cdot)$.

### 5.3 Cycle Projection Enhancement

Inspired by the concept of back-translation in machine translation (Sennrich et al., 2015; Lample et al., 2018; He et al., 2016), we design a cycle projection strategy to train the model on the nonparallel data. Given the input without the ground-truth, we train the projection function $f(\cdot)$ and $g(\cdot)$ by projecting back and forth to reconstruct the input. The cycle projection process is shown in Figure 2.

Formally, for a sentence $a$ in style $s^a$, after getting its latent representation $c_a$ in its own latent space by the $\text{Enc}^{(a)}$, we first project it to the latent space of $s^b$ by $f(\cdot)$ and get $\tilde{c}_b = f(c_a)$. Then we exert $g(\cdot)$ to project $\tilde{c}_b$ back to the latent space of style $s^a$, denoted as $\tilde{c}_a = g(\tilde{c}_b)$. Finally $\tilde{c}_a$ is fed to $\text{Dec}^{(a)}$ to produce an output $\tilde{a}$. Though the latent vector $\tilde{c}_b$ has no reference to train, the latent vector $\tilde{c}_a$ and the output $\tilde{a}$ could be trained by treating $c_a$ and $a$ as the references. The loss functions are formulated as:

$$\tilde{c}_a = g(f(\text{Enc}^{(a)}(a))) \quad (6)$$
$$l_{c_1} = \|\tilde{c}_a - c_a\| \quad (7)$$
$$l_{c_2} = -\log p(a|\tilde{c}_a) \quad (8)$$
$$l_c = \alpha * l_{c_1} + \beta * l_{c_2} \quad (9)$$

### 5.4 Training Procedure

In the training stage, we first pretrain DAE models for each style separately on the nonparallel data to get the latent spaces of styles. Then we alternately apply the cross projection strategy and cycle projection enhancement on parallel data and nonparallel data with loss $l_s$ and $l_c$.

## 6 Straightforward Semi-supervised Methods

We also introduce another two semi-supervised methods as the baselines to provide a more comprehensive comparison with the proposed CPLS model. The two semi-supervised methods are built from different perspectives.

### 6.1 Data Augmentation via Retrieval

One semi-supervised baseline is from the perspective of data augmentation. In order to alleviate the over-fitting issue caused by the small-scale parallel data, we propose a simple but efficient method that augments the parallel dataset by retrieving pseudo-references for the nonparallel datasets, denoted as DAR model.

**Pseudo-parallel Corpus Construction.** We employ Lucene[2] to build indexes for the nonparallel corpus of each style. Then the pseudo-references are retrieved based on the TF-IDF scores. To build the pseudo-parallel corpus of style $s^a$ and $s^b$, we first samples 80,000 sentences in style $s^a$ as queries. Specifically, each query searches a pseudo-reference from the nonparallel corpus of style $s^b$ according to the TF-IDF based cosine similarity. The sampled query is therefore coupled with the pseudo-reference to form a training pair. We also conduct the same operation that uses sampled queries from style $s^b$ to search pseudo-references from sentences of style $s^a$. After searching pseudo-references from two sides,

[2] http://lucene.apache.org/

| Dataset | Parallel | | | Nonparallel |
|---|---|---|---|---|
| | Train | Valid | Test | |
| Anc.p | 3,362 | 400 | 998 | 269k |
| M.zh | | | | 77k |
| F.en | 5,000 | 2,247 | 1,019 | 2,483k |
| Inf.en | | 2,788 | 1,332 | 4,891k |

Table 1: Statistics of two style transfer datasets.

we construct a pseudo-parallel corpus in the size of 150,000 pairs[3]. With the pseudo-parallel corpus, the model is exposed to more information and thus the problem of over-fitting can be mitigated to some extent. Though the relevance of the content between the input sentence and the pseudo-reference is not guaranteed, the encoder could better learn to extract the language features and the decoder could also benefit from the weak contextual correlation information.

**Training Procedure.** In the training stage, we first train the S2S model on the pseudo-parallel dataset and save a checkpoint every 2,000 steps. We then calculate the BLEU score of checkpoints on the validation set and select the one with the highest BLEU score as the final pretrained model. Then based on this pretrained model, we fine-tune the parameters using the true parallel data.

### 6.2 Shared Latent Space Model

The second semi-supervised baseline is similar to CPLS that first trains a DAE model on the corpus of each style. Instead of building a bridge between the latent spaces of two styles through projection functions, in this method, we simply share the latent representations by cross linking the encoder and decoder, denoted as SLS model. Given a pair of sentence $(a^p, b^p)$, the encoder $\text{Enc}^{(a)}$ encodes it into context $c_a$, then the decoder $\text{Dec}^{(b)}$ directly takes $c_a$ to produce $\tilde{b}$.

**Training Procedure.** For this method, we first pre-train two DAE models. Then we train the DAE models and the cross-linked S2S models from two transfer directions alternately. Considering the size imbalance between the nonparallel corpora and parallel corpus, to avoid the S2S falling into over-fitting too fast, We alternate the training in the form of training 20-step DAE models and then training one step S2S model.

[3]Some quires did not have corresponding retrieved result, therefore the total size is less than the number of queries.

## 7 Experiment

We conduct experiments on two bilateral style transfer tasks that each of them has a small-scale parallel corpus and two large-scale nonparallel corpora in two styles. DAR model and SLS model are treated as two baselines of CPLS model. In addition, we also train a S2S model with attention mechanism on the parallel data as the vanilla baseline. The following subsections elaborate the construction of the datasets and the detailed experimental settings.

### 7.1 Datasets

We construct a Chinese literal style dataset with ancient poetry style and modern Chinese style, and an English formality style dataset with formal style and informal style. The texts of ancient Chinese poem, modern Chinese, formal English and informal English are referred as Anc.P, M.zh, F.en and Inf.en for short. The overall statistics are shown in Table 1.

**Chinese Literal Style Dataset.** We consider the texts of ancient Chinese poem style and modern Chinese style which differ greatly in expression. As the ancient poems from different dynasties vary in style and form, in this paper, we focus on the poem from Tang dynasty.

To build the parallel dataset, we crawl from the website named Gushiwen[4] that provides ancient Chinese poems and some are coupled with modern Chinese interpretation in paragraphs. We split the collected paragraphs into independent sentences using punctuation based rules, and manually align the poem sentence with the interpretation sentence to form the parallel pair.

For the nonparallel corpus of ancient Chinese poem style, we collect all the poems from Quan Tangshi[5] and split them into sentences. To build the nonparallel dataset in modern Chinese style, we collect data from the Chinese ballad and took the lyrics. The reason we choose the Chinese ballad is that the content domain of the two styles should be close. Considering that most of the poems are about the natural scenery and the sentiments, the most suitable literary form is the lyrics of ballad. [6]

[4]https://www.gushiwen.org/

[5]The largest collection of Tang poems commissioned by the Qing dynasty Kangxi Emperor.

[6] We count the overlap of the topic words involved in the top 20 topics between the poem translation and ballad, and compare the overlap ratio with other modern Chinese corpora

| | | |
|---|---|---|
| to Anc.p | Source | 问客人为什么来，客人说为了上山砍伐树木来买斧头。<br>(Ask the guests why, the guest said he want to buy an axe in order to cut the trees at the mountain.) |
| | S2S | 客问谁客中，树。<br>(The guest asks who is in the guest, the tree) |
| | SLS | 问人何为人，<br>(Ask people what people are,) |
| | DAR | 客中何为客，山头为木头。<br>(What is the guest in the guest, the mountain head is the wooden head.) |
| | CPLS | 问客何为焉，客从满木盘。<br>(Ask the guest what he is going for, the guest need to get something out of full wooden forest.) |
| to M.zh | Source | 水寒风似刀。<br>(The water is as cold as a knife.) |
| | S2S | 水夜寒风吹动了寒冷。<br>(The cold wind with water at night blew the cold ) |
| | SLS | 寒月寒风吹得寒冷寒冷。<br>(The cold wind and the cold moon blew the cold ) |
| | DAR | 寒风吹得很寒冷。<br>(The cold wind is very cold) |
| | CPLS | 水边寒冷的寒风吹过好像寒冷的刀刀。<br>(The cold wind from the water side like a cold knife.) |
| to F.en | Source | give them a chance to discover you. |
| | S2S | Share them an opportunity to meet you. |
| | SLS | I give them a chance to discover you. |
| | DAR | Give them a chance to discover you. |
| | CPLS | You should give them a chance to discover you. |
| to Inf.en | Source | I think it is wrong that they cannot go out with her. |
| | S2S | I think it is wrong that is wrong with her. |
| | SLS | I think they can't go out with her. |
| | DAR | I think it is wrong that they cant go out with her. |
| | CPLS | It is wrong that they can't go out with her. |

Table 2: Case study: the sentences in the brackets are the translation of the corresponding sentence.

**Formality Dataset.** The formality dataset used in this paper is built based on the parallel dataset released by Rao and Tetreault (2018) which contains texts of formal and informal style.

With the released data, we randomly sample 5,000 sentence pairs from it as the parallel corpus with limited data volume. We then use the Yahoo Answers L6 corpus[7] as the source which is in the same content domain as the parallel data to construct the large-scale nonparallel data. To divide nonparallel dataset into two styles, we train a CNN-based classifier (Kim, 2014) on the parallel data with annotation of styles and use it to classify the nonparallel data.

### 7.2 Experimental Settings

We perform different data preprocessing on different datasets. The Chinese literary datasets are segmented by characters instead of word to alleviate the issue of unknown words. Our statistics show that the average length of ancient poems is 9 while 17 for modern Chinese sentences. Therefore, we set the minimum length of the ancient poem as 3, and the maximum length of the modern Chinese sentence as 30. For the formality datasets, we use NLTK (Loper and Bird, 2002) to tokenize the texts and set the minimum length as 5 and the maximum length as 30 for both formal and informal styles.

We adopt GloVE (Pennington et al., 2014) to pretrain the embeddings, and the dimensions of the embeddings are set to 300 for all the datasets. The hidden states are set to 500 for both encoders and decoders. We adopt SGD optimizer with the learning rate as 1 for DAE models and 0.1 for S2S models. The dropout rate is 0.4. In the inference stage, the beam size is set to 5.

## 8 Comparison Models and Evaluations

### 8.1 Baselines

We train a S2S model with attention mechanism on the parallel data as the supervised learning baseline. Since the existing works on text style transfer seldom explore the semi-supervised methods, we propose DAR and SLS model as two semi-supervised baselines.

### 8.2 Automatic Evaluation Metrics

Following previous works (Prabhumoye et al., 2018; Zhang et al., 2018a), we employ BLEU score (Papineni et al., 2002) and style accuracy

(news, open-domain conversations).

[7] https://webscope.sandbox.yahoo.com/catalog.php?datatype=1

| | to Anc.P | | | to M.zh | | | to F.en | | | to Inf.en | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Model | Acc | BLEU | GLEU | Acc | BLEU | GLEU | Acc | BLEU | GLEU | Acc | BLEU | GLEU |
| S2S | **87.2%** | 4.20 | 3.24 | 74.8% | 3.66 | 3.43 | 88.9% | 33.90 | 14.06 | **71.8%** | 18.34 | 2.99 |
| SLS | 82.0% | 5.89 | 4.49 | 81.9% | 3.05 | 1.88 | 89.5% | 41.41 | 16.77 | 63.5% | 19.21 | 2.55 |
| DAR | 82.5% | 6.33 | 5.21 | 80.4% | 4.72 | **4.26** | 89.2% | 44.72 | 18.52 | 63.5% | 23.32 | 3.26 |
| CPLS | 85.4% | **7.11** | **5.52** | **84.4%** | **4.95** | 4.12 | **91.3%** | **48.60** | **19.04** | 64.3% | **27.25** | **3.61** |

Table 3: Automatic evaluation results on four style transfer tasks. Acc refers to the style accuracy.

| Model | Dataset | Content | Style | Fluency |
|---|---|---|---|---|
| S2S | to M.zh | 0.1875 | 0.5675 | 0.3575 |
| | to Anc.P | 0.2275 | 0.5400 | 0.4425 |
| | to Inf.en | 0.3175 | 0.4600 | 0.5800 |
| | to F.en | 0.3625 | 0.5125 | 0.6325 |
| CPLS | to M.zh | 0.3100 | 0.5825 | 0.3050 |
| | to Anc.P | 0.4375 | 0.6875 | 0.5475 |
| | to Inf.en | 0.4675 | 0.4625 | 0.5725 |
| | to F.en | 0.4600 | 0.5675 | 0.6200 |

Table 4: The human annotation results of the S2S model and CPLS model from three aspects.

as the automatic evaluation metrics to measure the content preservation degree and the style changing degree. BLEU calculates the N-gram overlap between the generated sentence and the references, thus can be used to measure the preservation of text content. Considering that text style transfer is a monolingual text generation task, we also use GLEU, a generalized BLEU proposed by (Napoles et al., 2015). To evaluate the extent to which the sentences are transferred to the target style, we follow Shen et al. (2017); Hu et al. (2017) that build a CNN-based style classifier and use it to measure the style accuracy.

### 8.3 Human Evaluation

We also adopt human evaluations to judge the quality of the transferred sentences from three aspects, namely content, style and fluency. These aspects evaluate how well the transferred text preserve the content of the input, the style strength and the fluency of the transferred text. Take the content relevance for example, the criterion is as follows: **+2:** The transferred sentence has the same meaning with the input sentence. **+1:** The transferred sentence preserves part of the content meaning of the input sentence. **0:** The transferred sentence and the input sentence are irrelevant in content. The criteria for style strength and fluency are similar to the content relevance criterion.

To get the evaluation results, we first randomly sample 50 test cases from each dataset. As the style transfer is bilateral in this paper, there are 400 test cases in all. We invited four well-educated volunteers to score the results from the supervised baseline and the three semi-supervised models.

## 9 Results and Analysis

**Evaluation Results.** Table 3 presents the evaluation results of automatic metrics on the models. It can be seen that the BLEU scores and GLEU scores of the semi-supervised models on almost all the datasets are better than the baseline S2S model. This result indicates that the model benefits from the nonparallel data in terms of content preservation. One interesting thing is that the overall BLEU scores on the ancient poems and modern Chinese datasets are lower than other datasets. This result may be explained by the fact that the edit distance between formal and informal texts are smaller than between ancient poems and modern Chinese texts. Therefore, it is more challenging for model to preserve the content meaning when transferring between ancient poems and modern Chinese text. Among three semi-supervised models, CPLS model achieves the greatest improvement, verifying the effectiveness of the projection functions. However, the gain of CPLS model in the aspect of style accuracy is not that significant. A possible explanation may be the bias of the style classifier. Take the transfer task from ancient poems to modern Chinese text for example. We observe that the classifier tends to classify short sentences into ancient poems as length is an obvious feature. We analyse the sentences generated by S2S model and by the CPLS model, and the statistics show that the average length of the text generated by S2S model is shorter, which may lead to the bias of the style classifier. Therefore, we also adopt human evaluation to alleviate this issue.

Table 4 compares the human evaluation results of S2S model and CPLS model on all the datasets, which are calculated by the average score of the human annotations. As shown in the Table 4, the CPLS model outperforms the S2S model in

the aspects of the content preservation and style strength, and is on par in terms of fluency.

**Case Study.** We select the generated results of the S2S baseline and three semi-supervised models on two style transfer tasks due to the limited space as shown in Table 2. Compared with the Chinese literacy style datasets, the formality datasets are less challenging as discussed before. Thus it can be seen from the table that the generated sentences on the formality datasets are more fluent. For the task that transferring from the modern Chinese text to the ancient poem, the S2S generates a shorter sentence while the CPLS model generates the sentence that preserve the most content information.

## 10 Conclusion

In this paper, we design a differentiable semi-supervised model that introduces a projection function between the latent spaces of different styles. The model is flexible in alternating the training mode between supervised and unsupervised learning. We also introduce another two semi-supervised methods that are simple but effective to use the nonparallel and parallel data. We evaluate our models on two datasets that have small-scale parallel data and large-scale nonparallel data, and verify the effectiveness of the model on both automatic metrics and human evaluation.

## 11 Acknowledgement

We would like to thank the anonymous reviewers for their constructive comments. This work was supported by the National Key Research and Development Program of China (No. 2017YFC0804001), the National Science Foundation of China (NSFC No. 61876196 and NSFC No. 61672058). Rui Yan was sponsored by Tencent Collaborative Research Fund.

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Cheng-Kuan Chen, Zhufeng Pan, Ming-Yu Liu, and Min Sun. 2019. Unsupervised stylish image description generation via domain layer norm. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8151–8158.

Sumit Chopra, Michael Auli, and Alexander M Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2018. Style transfer in text: Exploration and evaluation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Yifan Gao, Lidong Bing, Wang Chen, Michael R. Lyu, and Irwin King. 2019. Difficulty controllable generation of reading comprehension questions. In *Proceedings of the Twenty-Eightth International Joint Conference on Artificial Intelligence, IJCAI-19*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In *Advances in Neural Information Processing Systems*, pages 820–828.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1587–1596. JMLR. org.

Zhijing Jin, Di Jin, Jonas Mueller, Nicholas Matthews, and Enrico Santus. 2019. Unsupervised text style transfer via iterative matching and translation. *arXiv preprint arXiv:1901.11333*.

Vineet John, Lili Mou, Hareesh Bahuleyan, and Olga Vechtomova. 2018. Disentangled representation learning for text style transfer. *arXiv preprint arXiv:1808.04339*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Diederik P. Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv*, abs/1312.6114.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Phrase-based & neural unsupervised machine translation. *arXiv preprint arXiv:1804.07755*.

Guillaume Lample, Sandeep Subramanian, Eric Smith, Ludovic Denoyer, Marc'Aurelio Ranzato, and Y-Lan Boureau. 2019. Multiple-attribute text rewriting. In *International Conference on Learning Representations*.

Yi Liao, Lidong Bing, Piji Li, Shuming Shi, Wai Lam, and Tong Zhang. 2018. Quase: Sequence editing under quantifiable guidance. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3855–3864.

Lajanugen Logeswaran, Honglak Lee, and Samy Bengio. 2018. Content preserving text generation with attribute controls. In *Advances in Neural Information Processing Systems*, pages 5103–5113.

Edward Loper and Steven Bird. 2002. Nltk: the natural language toolkit. *arXiv preprint cs/0205028*.

Jonas Mueller, David Gifford, and Tommi Jaakkola. 2017. Sequence to better sequence: continuous revision of combinatorial structures. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2536–2544. JMLR. org.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2015. Ground truth for grammatical error correction metrics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 588–593.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W Black. 2018. Style transfer through back-translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 866–876. Association for Computational Linguistics.

Sudha Rao and Joel Tetreault. 2018. Dear sir or madam, may i introduce the gyafc dataset: Corpus, benchmarks and metrics for formality style transfer. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 129–140. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709*.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in neural information processing systems*, pages 6830–6841.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Yi Zhang, Jingjing Xu, Pengcheng Yang, and Xu Sun. 2018a. Learning sentiment memories for sentiment modification without parallel data. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1103–1108. Association for Computational Linguistics.

Zhirui Zhang, Shuo Ren, Shujie Liu, Jianyong Wang, Peng Chen, Mu Li, Ming Zhou, and Enhong Chen. 2018b. Style transfer as unsupervised machine translation. *arXiv preprint arXiv:1808.07894*.

Yanpeng Zhao, Wei Bi, Deng Cai, Xiaojiang Liu, Kewei Tu, and Shuming Shi. 2018. Language style transfer from sentences with arbitrary unknown styles. *arXiv preprint arXiv:1808.04071*.

# Question Answering for Privacy Policies: Combining Computational and Legal Perspectives

**Abhilasha Ravichander** ♢ **Alan Black** ♢ **Shomir Wilson** ♡
**Thomas Norton**♠ **Norman Sadeh**♢
♢Carnegie Mellon University, Pittsburgh, PA
♡Penn State University, University Park, PA ♠Fordham Law School, New York, NY
{aravicha, awb, sadeh}@cs.cmu.edu
{shomir}@psu.edu, {tnorton1}@law.fordham.edu

## Abstract

Privacy policies are long and complex documents that are difficult for users to read and understand, and yet, they have legal effects on how user data is collected, managed and used. Ideally, we would like to empower users to inform themselves about issues that matter to them, and enable them to selective explore those issues. We present PRIVACYQA, a corpus consisting of 1750 questions about the privacy policies of mobile applications, and over 3500 expert annotations of relevant answers. We observe that a strong neural baseline underperforms human performance by almost 0.3 F1 on PRIVACYQA, suggesting considerable room for improvement for future systems. Further, we use this dataset to shed light on challenges to *question answerability*, with domain-general implications for any question answering system. The PRIVACYQA corpus offers a challenging corpus for question answering, with genuine real-world utility.

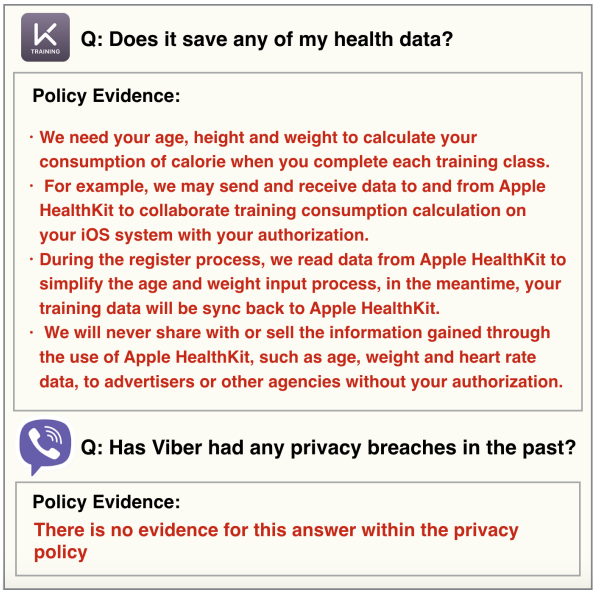


Figure 1: Examples of privacy-related questions users ask, drawn from two mobile applications: Keep[1] and Viber.[2] Policy evidence represents sentences in the privacy policy that are relevant for determining the answer to the user's question.[3]

## 1 Introduction

Privacy policies are the documents which disclose the ways in which a company gathers, uses, shares and manages a user's data. As legal documents, they function using the principle of *notice and choice* (Federal Trade Commission, 1998), where companies post their policies, and theoretically, users read the policies and decide to use a company's products or services only if they find the conditions outlined in its privacy policy acceptable. Many legal jurisdictions around the world accept this framework, including the United States and the European Union (Patrick, 1980; OECD, 2004). However, the legitimacy of this framework depends upon users actually reading and understanding privacy policies to determine whether company practices are acceptable to them (Reidenberg et al., 2015). In practice this is seldom the case (Cate, 2010; Cranor, 2012; Schaub et al., 2015; Gluck et al., 2016; Jain et al., 2016; US Federal Trade Commission et al., 2012; McDonald and Cranor, 2008). This is further complicated by the highly individual and nuanced compromises that users are willing to make with their data (Leon et al., 2015), discouraging a 'one-size-fits-all' approach to notice of data practices in privacy documents.

With devices constantly monitoring our environment, including our personal space and our bodies, lack of awareness of how our data is being used easily leads to problematic situations where users are outraged by information misuse, but companies insist that users have consented. The discovery of increasingly egregious uses of data by companies, such as the scandals involv-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4947–4958, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

ing Facebook and Cambridge Analytica (Cadwalladr and Graham-Harrison, 2018), have further brought public attention to the privacy concerns of the internet and ubiquitous computing. This makes privacy a well-motivated application domain for NLP researchers, where advances in enabling users to quickly identify the privacy issues most salient to them can potentially have large real-world impact.

Motivated by this need, we contribute PRIVACYQA, a corpus consisting of 1750 questions about the contents of privacy policies[4], paired with over 3500 expert annotations. The goal of this effort is to kickstart the development of question-answering methods for this domain, to address the (unrealistic) expectation that a large population should be reading many policies per day. In doing so, we identify several understudied challenges to our ability to answer these questions, with broad implications for systems seeking to serve users' information-seeking intent. By releasing this resource, we hope to provide an impetus to develop systems capable of language understanding in this increasingly important domain.[5]

## 2 Related Work

Prior work has aimed to make privacy policies easier to understand. Prescriptive approaches towards communicating privacy information (Kelley et al., 2009; Micheti et al., 2010; Cranor, 2003) have not been widely adopted by industry. Recently, there have been significant research effort devoted to understanding privacy policies by leveraging NLP techniques (**?**Liu et al., 2016; Oltramari et al., 2017; Mysore Sathyendra et al., 2017; Wilson et al., 2017), especially by identifying specific data practices within a privacy policy. We adopt a personalized approach to understanding privacy policies, that allows users to query a document and selectively explore content salient to them. Most similar is the PolisisQA corpus (Harkous et al., 2018), which examines questions users ask corporations on Twitter. Our approach differs in several ways: 1) The PRIVACYQA dataset is larger, containing 10x as many questions and answers. 2) Answers are formulated by domain experts with legal training. [6] 3) PRIVACYQA includes diverse question types, including unanswerable and subjective questions.

Our work is also related to reading comprehension in the open domain, which is frequently based upon Wikipedia passages (Rajpurkar et al., 2016, 2018; Joshi et al., 2017; Choi et al., 2018) and news articles (Trischler et al., 2017; Hermann et al., 2015; Onishi et al., 2016). Table.1 presents the desirable attributes our dataset shares with past approaches. This work is also tied into research in applying NLP approaches to legal documents (Monroy et al., 2009; Quaresma and Rodrigues, 2005; Do et al., 2017; Kim et al., 2015; Liu et al., 2015; Mollá and Vicedo, 2007; Frank et al., 2007). While privacy policies have legal implications, their intended audience consists of the general public rather than individuals with legal expertise. This arrangement is problematic because the entities that write privacy policies often have different goals than the audience. Feng et al. (2015); Tan et al. (2016) examine question answering in the insurance domain, another specialized domain similar to privacy, where the intended audience is the general public.

## 3 Data Collection

We describe the data collection methodology used to construct PRIVACYQA. With the goal of achieving broad coverage across application types, we collect privacy policies from 35 mobile applications representing a number of different categories in the Google Play Store.[7][8] One of our goals is to include both policies from well-known applications, which are likely to have carefully-constructed privacy policies, and lesser-known applications with smaller install bases, whose policies might be considerably less sophisticated. Thus, setting 5 million installs as a threshold, we ensure each category includes applications with installs on both sides

[1] https://play.google.com/store/apps/details?id=com.gotokeep.keep.intl

[2] https://play.google.com/store/apps/details?id=com.viber.voip

[3] A question might not have any supporting evidence for an answer within the privacy policy.

[4] All privacy policies in this corpus are in English.

[5] PRIVACYQA is freely available at https://github.com/AbhilashaRavichander/PrivacyQA_EMNLP.

[6] This choice was made as privacy policies are legal documents, and require careful expert understanding in order to be interpreted correctly.

[7] We choose categories that occupy atleast a 2% share of all application categories on the Play Store (Story et al., 2018)

[8] As of April 1, 2018

| Dataset | Document Source | Expert Annotator | Simple Evaluation | Unanswerable Questions | Asker Cannot See Evidence |
|---|---|---|---|---|---|
| PrivacyQA | Privacy Policies | ✓ | ✓ | ✓ | ✓ |
| NarrativeQA (Kočiskỳ et al., 2018) | Fiction | ✗ | ✗ | ✗ | ✓ |
| InsuranceQA (Feng et al., 2015) | Insurance | ✓ | ✓ | ✗ | ✓ |
| TriviaQA (Joshi et al., 2017) | Wikipedia | ✗ | ✓ | ✗ | ✓ |
| SQuAD 1.0 (Rajpurkar et al., 2016) | Wikipedia | ✗ | ✓ | ✗ | ✗ |
| SQuAD 2.0 (Rajpurkar et al., 2018) | Wikipedia | ✗ | ✓ | ✓ | ✗ |
| MS Marco (Nguyen et al., 2016) | Web Documents | ✗ | ✗ | ✓ | ✓ |
| MC Test (Richardson et al., 2013) | Fiction | ✗ | ✓ | ✗ | ✗ |
| NewsQA (Trischler et al., 2017) | News Articles | ✗ | ✓ | ✓ | ✓ |

Table 1: Comparison of the PRIVACYQA dataset to other question answering datasets. Expert annotator indicates domain expertise of the answer provider. Simple evaluation indicates the presence of an automatically calculable evaluation metric. Unanswerable questions indicates if the respective corpus includes unanswerable questions. 'Asker Cannot See Evidence' indicates that the asker of the question was not shown evidence from the document at the time of formulating questions.




Figure 2: User interface for question elicitation.

of this threshold.[9] All policies included in the corpus are in English, and were collected before April 1, 2018, predating many companies' GDPR-focused (Voigt and Von dem Bussche, 2017) updates. We leave it to future studies (Gallé et al., 2019) to look at the impact of the GDPR (e.g., to what extent GDPR requirements contribute to making it possible to provide users with more informative answers, and to what extent their disclosures continue to omit issues that matter to users).

### 3.1 Crowdsourced Question Elicitation

The intended audience for privacy policies consists of the general public. This informs the decision to elicit questions from crowdworkers on the contents of privacy policies. We choose not to show the contents of privacy policies to crowdworkers, a procedure motivated by a desire to avoid inadvertent biases (Weissenborn et al., 2017; Kaushik and Lipton, 2018; Poliak et al., 2018; Gururangan et al., 2018; Naik et al., 2018), and encourage crowdworkers to ask a variety of questions beyond only asking questions based on practices described in the document.

Instead, crowdworkers are presented with public information about a mobile application available on the Google Play Store including its name, description and navigable screenshots. Figure 2 shows an example of our user interface.[10] Crowdworkers are asked to imagine they have access to a trusted third-party privacy assistant, to whom they can ask any privacy question about a given mobile application. We use the Amazon Mechanical Turk platform[11] and recruit crowdworkers who have been conferred "master" status and are located within the United States of America. Turkers are asked to provide five questions per mobile application, and are paid $2 per assignment, taking ~eight minutes to complete the task.

[9] The final application categories represented in the corpus consist of books, business, education, entertainment, lifestyle, music, health, news, personalization, photography, productivity, tools, travel and game applications.

[10] Color blindness affects approximately 4.5% of the worlds population (Deeb, 2005) We design all our crowdworker user interfaces to accommodate red-green color blindness.

[11] https://www.mturk.com/

| Word | (%) | Example Question From PRIVACYQA |
|---|---|---|
| Will | 13.03 | will my data be sold or available to be used by third party entities? |
| Do | 2.70 | do you sell any of my information |
| What | 1.84 | what is the worst case scenario of giving my information to this game? |
| Are | 1.60 | are there any safety concerns i should know about using this app? |
| Can/ could | 1.47 | can i control who sees my data? |
| How | 1.40 | how secure is my stored data? |
| Where | 1.14 | where is my account info and online activity stored? |
| Has | 0.69 | has there ever been a security breach? |
| If | 0.61 | if i delete the app will you keep my data? |
| Does | 0.46 | does the app save the addresses that i enter? |

Table 2: Ten most frequent first words in questions in the PRIVACYQA dataset. We observe high lexical diversity in prefixes with 35 unique first word types and 131 unique combinations of first and second words.

### 3.2 Answer Selection

To identify legally sound answers, we recruit seven experts with legal training to construct answers to Turker questions. Experts identify relevant evidence within the privacy policy, as well as provide meta-annotation on the question's relevance, subjectivity, OPP-115 category (Wilson et al., 2016), and how likely any privacy policy is to contain the answer to the question asked.

### 3.3 Analysis

Table.4 presents aggregate statistics of the PRIVACYQA dataset. 1750 questions are posed to our imaginary privacy assistant over 35 mobile applications and their associated privacy documents. As an initial step, we formulate the problem of answering user questions as an extractive sentence selection task, ignoring for now background knowledge, statistical data and legal expertise that could otherwise be brought to bear. The dataset is partitioned into a training set featuring 27 mobile applications and 1350 questions, and a test set consisting of 400 questions over 8 policy documents. This ensures that documents in training and test splits are mutually exclusive. Every question is answered by at least one expert. In addition, in order to estimate annotation reliability and provide for better evaluation, every question in the test set is answered by at least two additional experts.

Table 2 describes the distribution over first words of questions posed by crowdworkers. We also observe low redundancy in the questions posed by crowdworkers over each policy, with each policy receiving ~49.94 unique questions despite crowdworkers independently posing questions. Questions are on average 8.4 words long. As declining to answer a question can be a legally sound response but is seldom practically useful, answers to questions where a minority of experts abstain to answer are filtered from the dataset. Privacy policies are ~3000 words long on average. The answers to the question asked by the users typically have ~100 words of evidence in the privacy policy document.

#### 3.3.1 Categories of Questions

Questions are organized under nine categories from the OPP-115 Corpus annotation scheme (Wilson et al., 2016):

1. First Party Collection/Use: What, why and how information is collected by the service provider
2. Third Party Sharing/Collection: What, why and how information shared with or collected by third parties
3. Data Security: Protection measures for user information
4. Data Retention: How long user information will be stored
5. User Choice/Control: Control options available to users
6. User Access, Edit and Deletion: If/how users can access, edit or delete information
7. Policy Change: Informing users if policy information has been changed
8. International and Specific Audiences: Practices pertaining to a specific group of users
9. Other: General text, contact information or practices not covered by other categories.

For each question, domain experts indicate one or more[12] relevant OPP-115 categories. We mark a category as relevant to a question if it is identified as such by at least two annotators. If no such category exists, the category is marked

[12]For example, some questions such as *'What information of mine is collected by this app and who is it shared with?'* can be identified as falling under both first party data/collection and third party collection/sharing categories.

| Privacy Practice | Proportion | Example Question From PRIVACYQA |
|---|---|---|
| First Party Collection/Use | 41.9 % | what data does this game collect? |
| Third Party Sharing/Collection | 24.5 % | will my data be sold to advertisers? |
| Data Security | 10.5 % | how is my info protected from hackers? |
| Data Retention | 4.1 % | how long do you save my information? |
| User Access, Edit and Deletion | 2.0 % | can i delete my information permanently? |
| User Choice/Control | 6.5 % | is there a way to opt out of data sharing |
| Other | 4.8 % | does the app connect to the internet at any point? |
| Policy Change | 0.2 % | where is the privacy statement |
| International and Specific Audiences | 0.2 % | what are your GDPR policies? |
| No Agreement | 5.4 % | how are features personalized? |

Table 3: OPP-115 categories most relevant to the questions collected from users.

| Dataset | Train | Test | All |
|---|---|---|---|
| # Questions | 1350 | 400 | 1750 |
| # Policies | 27 | 8 | 35 |
| # Sentences | 3704 | 1243 | 4947 |
| Avg Q. Length | 8.42 | 8.56 | 8.46 |
| Avg Doc. Length | 3121.3 | 3629.13 | 3237.37 |
| Avg Ans. Length | 123.73 | 153.44 | 139.62 |

Table 4: Statistics of the PRIVACYQA Dataset, where # denotes number of questions, policies and sentences, and average length of questions, policies and answers in words, for training and test partitions.

as 'Other' if atleast one annotator has identified the 'Other' category to be relevant. If neither of these conditions is satisfied, we label the question as having no agreement. The distribution of questions in the corpus across OPP-115 categories is as shown in Table.3. First party and third party related questions are the largest categories, forming nearly 66.4% of all questions asked to the privacy assistant.

#### 3.3.2 Answer Validation

When do experts disagree? We would like to analyze the reasons for potential disagreement on the annotation task, to ensure disagreements arise due to valid differences in opinion rather than lack of adequate specification in annotation guidelines. It is important to note that the annotators are experts rather than crowdworkers. Accordingly, their judgements can be considered valid, legally-informed opinions even when their perspectives differ. For the sake of this question we randomly sample 100 instances in the test data and analyze them for likely reasons for disagreements. We consider a disagreement to have occurred when more than one expert does not agree with the majority consensus. By disagreement we mean there is no overlap between the text identified as relevant by one expert and another.

We find that the annotators agree on the answer for 74% of the questions, even if the supporting evidence they identify is not identical i.e full overlap. They disagree on the remaining 26%. Sources of apparent disagreement correspond to situations when different experts: have differing interpretations of question intent (11%) (for example, when a user asks *'who can contact me through the app'*, the questions admits multiple interpretations, including seeking information about the features of the app, asking about first party collection/use of data or asking about third party collection/use of data), identify different sources of evidence for questions that ask if a practice is performed or not (4%), have differing interpretations of policy content (3%), identify a partial answer to a question in the privacy policy (2%) (for example, when the user asks *'who is allowed to use the app'* a majority of our annotators decline to answer, but the remaining annotators highlight partial evidence in the privacy policy which states that children under the age of 13 are *not* allowed to use the app), and other legitimate sources of disagreement (6%) which include personal subjective views of the annotators (for example, when the user asks 'is my DNA information used in any way *other than what is specified*', some experts consider the boilerplate text of the privacy policy which states that it abides to practices described in the policy document as sufficient evidence to answer this question, whereas others do not).

| | Acc. | P | R | F1 |
|---|---|---|---|---|
| Majority | 24.75 | 24.75 | 100 | 39.6 |
| SVM-BOW | 75.75 | 50.8 | 58.5 | 54.4 |
| + LEN | 76.75 | 52.7 | 57.5 | 55.0 |
| + LEN + POS | 77.0 | 53.2 | 58.5 | 55.7 |
| CNN | 80.0 | 61.1 | 52.5 | 56.5 |
| BERT | 81.15 | 62.6 | 62.6 | 62.6 |

Table 5: Classifier Performance (%) for answerability of questions. The Majority Class baseline always predicts that questions are unanswerable.

| Model | Precision | Recall | F1 |
|---|---|---|---|
| No Answer (NA) | 28.0% | 28.0% | 28.0% |
| Word Count -2 | 24.0% | 16.4% | 19.4% |
| Word Count -3 | 21.8% | 17.8% | 19.6% |
| Word Count -5 | 18.1% | 20.3% | 19.2% |
| BERT | 44.2% | 34.8% | 39.0% |
| BERT + Unans. | 44.3% | 36.1% | **39.8**% |
| Human | 68.8% | 69.0% | 68.9% |

Table 6: Performance of baselines on PRIVACYQA dataset.

## 4 Experimental Setup

We evaluate the ability of machine learning methods to identify relevant evidence for questions in the privacy domain.[13] We establish baselines for the subtask of deciding on the answerability (§4.1) of a question, as well as the overall task of identifying evidence for questions from policies (§4.2). We describe aspects of the question that can render it unanswerable within the privacy domain (§5.2).

### 4.1 Answerability Identification Baselines

We define answerability identification as a binary classification task, evaluating model ability to predict if a question can be answered, given a question in isolation. This can serve as a prior for downstream question-answering. We describe three baselines on the answerability task, and find they considerably improve performance over a majority-class baseline.

**SVM**: We define 3 sets of features to characterize each question. The first is a simple bag-of-words set of features over the question (SVM-BOW), the second is bag-of-words features of the question as well as length of the question in words (SVM-BOW + LEN), and lastly we extract bag-of-words features, length of the question in words as well as part-of-speech tags for the question (SVM-BOW + LEN + POS). This results in vectors of 200, 201 and 228 dimensions respectively, which are provided to an SVM with a linear kernel.

**CNN**: We utilize a CNN neural encoder for answerability prediction. We use GloVe word embeddings (Pennington et al., 2014), and a filter size of 5 with 64 filters to encode questions.

**BERT**: BERT (Devlin et al., 2019) is a bidirectional transformer-based language-model (Vaswani et al., 2017).[14] We fine-tune BERT-base on our binary answerability identification task with a learning rate of 2e-5 for 3 epochs, with a maximum sequence length of 128.

[13]The task of evidence identification can serve as a first step for future question answering systems, that can further learn to form abstractive summaries when required based on identifying relevant evidence.

| Privacy Practice | NA | BERT-U | Human |
|---|---|---|---|
| First Party | 0.22 | 0.36 | 0.67 |
| Third Party | 0.10 | 0.26 | 0.61 |
| Data Security | 0.24 | 0.42 | 0.74 |
| Data Retention | 0.02 | 0.33 | 0.67 |
| User Access | 0.07 | 0.32 | 0.66 |
| User Choice | 0.35 | 0.41 | 0.65 |
| Other | 0.44 | 0.45 | 0.72 |

Table 7: Stratification of classifier performance by OPP-115 category of questions.

| | BERT |
|---|---|
| # Answerability Mistakes | 137 |
| % Answerable ->Unanswerable | 124 |
| % Unanswerable ->Answerable | 13 |
| Out-of-scope | 2 |
| Subjective | 46 |
| Policy Silent | 19 |
| Unexpected | 6 |

Table 8: Analysis of BERT performance at identifying answerability. The majority of mistakes made by BERT are answerable Questions identified as unanswerable. These answerable questions are further analyzed along the factors of scope, subjectivity, presence of answer and whether the question could be anticipated.

[14]We utilize the HuggingFace implementtation available at https://github.com/huggingface/pytorch-transformers

### 4.2 Privacy Question Answering

Our goal is to identify evidence within a privacy policy for questions asked by a user. This is framed as an answer sentence selection task, where models identify a set of evidence sentences from all candidate sentences in each policy.

#### 4.2.1 Evaluation Metric

Our evaluation metric for answer-sentence selection is sentence-level F1, implemented similar to (Choi et al., 2018; Rajpurkar et al., 2016). Precision and recall are implemented by measuring the overlap between predicted sentences and sets of gold-reference sentences. We report the average of the maximum F1 from each n−1 subset, in relation to the heldout reference.

#### 4.2.2 Baselines

We describe baselines on this task, including a human performance baseline.

**No-Answer Baseline (NA)** : Most of the questions we receive are difficult to answer in a legally-sound way on the basis of information present in the privacy policy. We establish a simple baseline to quantify the effect of identifying every question as unanswerable.

**Word Count Baseline** : To quantify the effect of using simple lexical matching to answer the questions, we retrieve the top candidate policy sentences for each question using a word count baseline (Yang et al., 2015), which counts the number of question words that also appear in a sentence. We include the top 2, 3 and 5 candidates as baselines.

**BERT**: We implement two BERT-based baselines (Devlin et al., 2019) for evidence identification. First, we train BERT on each query-policy sentence pair as a binary classification task to identify if the sentence is evidence for the question or not (BERT). We also experiment with a two-stage classifier, where we separately train the model on questions only to predict answerability. At inference time, if the answerable classifier predicts the question is answerable, the evidence identification classifier produces a set of candidate sentences (BERT + UNANSWERABLE).

**Human Performance**: We pick each reference answer provided by an annotator, and compute the F1 with respect to the remaining references, as described in section 4.2.1. Each reference answer is treated as the prediction, and the remaining n-1 answers are treated as the gold reference. The average of the maximum F1 across all reference answers is computed as the human baseline.

## 5 Results and Discussion

The results of the answerability baselines are presented in Table 5, and on answer sentence selection in Table 6. We observe that BERT exhibits the best performance on a binary answerability identification task. However, most baselines considerably exceed the performance of a majority-class baseline. This suggests considerable information in the question, indicating it's possible answerability within this domain.

Table.6 describes the performance of our baselines on the answer sentence selection task. The No-answer (NA) baseline performs at 28 F1, providing a lower bound on performance at this task. We observe that our best-performing baseline, BERT + UNANSWERABLE achieves an F1 of 39.8. This suggest that BERT is capable of making some progress towards answering questions in this difficult domain, while still leaving considerable headroom for improvement to reach human performance. BERT + UNANSWERABLE performance suggests that incorporating information about answerability can help in this difficult domain. We examine this challenging phenomena of unanswerability further in Section **??**.

### 5.1 Error Analysis

Disagreements are analyzed based on the OPP-115 categories of each question (Table.7). We compare our best performing BERT variant against the NA model and human performance. We observe significant room for improvement across all categories of questions but especially for first party, third party and data retention categories.

We analyze the performance of our strongest BERT variant, to identify classes of errors and directions for future improvement (Table.8). We observe that a majority of answerability mistakes made by the BERT model are questions which are in fact answerable, but are identified as unanswerable by BERT. We observe that BERT makes 124 such mistakes on the test set. We collect expert judgments on relevance, subjectivity , silence and information about how likely the question is to be answered from the privacy pol-

icy from our experts. We find that most of these mistakes are relevant questions. However many of them were identified as subjective by the annotators, and at least one annotator marked 19 of these questions as having no answer within the privacy policy. However, only 6 of these questions were unexpected or do not usually have an answer in privacy policies. These findings suggest that a more nuanced understanding of answerability might help improve model performance in his challenging domain.

### 5.2 What makes Questions Unanswerable?

We further ask legal experts to identify potential causes of unanswerability of questions. This analysis has considerable implications. While past work (Rajpurkar et al., 2018) has treated unanswerable questions as homogeneous, a question answering system might wish to have different treatments for different categories of 'unanswerable' questions. The following factors were identified to play a role in unanswerability:

- **Incomprehensibility**: If a question is incomprehensible to the extent that its meaning is not intelligible.
- **Relevance**: Is this question in the scope of what could be answered by reading the privacy policy.
- **Ill-formedness**: Is this question ambiguous or vague. An ambiguous statement will typically contain expressions that can refer to multiple potential explanations, whereas a vague statement carries a concept with an unclear or soft definition.
- **Silence**: Other policies answer this type of question but this one does not.
- **Atypicality**: The question is of a nature such that it is unlikely for any policy policy to have an answer to the question.

Our experts attempt to identify the different 'unanswerable' factors for all 573 such questions in the corpus. 4.18% of the questions were identified as being incomprehensible (for example, 'any difficulties to occupy the privacy assistant'). Amongst the comprehendable questions, 50% were identified as likely to have an answer within the privacy policy, 33.1% were identified as being privacy-related questions but not within the scope of a privacy policy (e.g., *'has Viber had any privacy breaches in the past?'*) and 16.9% of questions were identified as completely out-of-scope (e.g., *"will the app consume much space?'*). In the questions identified as relevant, 32% were ill-formed questions that were phrased by the user in a manner considered vague or ambiguous. Of the questions that were both relevant as well as 'well-formed', 95.7% of the questions were not answered by the policy in question but it was reasonable to expect that a privacy policy would contain an answer. The remaining 4.3% were described as reasonable questions, but of a nature generally not discussed in privacy policies. This suggests that the answerability of questions over privacy policies is a complex issue, and future systems should consider each of these factors when serving user's information seeking intent.

We examine a large-scale dataset of "natural" unanswerable questions (Kwiatkowski et al., 2019) based on real user search engine queries to identify if similar unanswerability factors exist. It is important to note that these questions have previously been filtered, according to a criteria for bad questions defined as "(questions that are) ambiguous, incomprehensible, dependent on clear false presuppositions, opinion-seeking, or not clearly a request for factual information." Annotators made the decision based on the content of the question without viewing the equivalent Wikipedia page. We randomly sample 100 questions from the development set which were identified as unanswerable, and find that 20% of the questions are not questions (e.g., "all I want for christmas is you mariah carey tour"). 12% of questions are unlikely to ever contain an answer on Wikipedia, corresponding closely to our atypicality category. 3% of questions are unlikely to have an answer anywhere (e.g., 'what guides Santa home after he has delivered presents?'). 7% of questions are incomplete or open-ended (e.g., 'the south west wind blows across nigeria between'). 3% of questions have an unresolvable coreference (e.g., 'how do i get to Warsaw Missouri from here'). 4% of questions are vague, and a further 7% have unknown sources of error. 2% still contain false presuppositions (e.g., 'what is the only fruit that does not have seeds?') and the remaining 42% do not have an answer within the document. This reinforces our belief that though they have been understudied in past

work, any question answering system interacting with real users should expect to receive such unanticipated and unanswerable questions.

## 6 Conclusion

We present PRIVACYQA, the first significant corpus of privacy policy questions and more than 3500 expert annotations of relevant answers. The goal of this work is to promote question-answering research in the specialized privacy domain, where it can have large real-world impact. Strong neural baselines on PRIVACYQA achieve a performance of only 39.8 F1 on this corpus, indicating considerable room for future research. Further, we shed light on several important considerations that affect the *answerability* of questions. We hope this contribution leads to multidisciplinary efforts to precisely understand user intent and reconcile it with information in policy documents, from both the privacy and NLP communities.


## Acknowledgements

This research was supported in part by grants from the National Science Foundation Secure and Trustworthy Computing program (CNS-1330596, CNS-1330214, CNS-15-13957, CNS-1801316, CNS-1914486, CNS-1914444) and a DARPA Brandeis grant on Personalized Privacy Assistants (FA8750-15-2-0277). The US Government is authorized to reproduce and distribute reprints for Governmental purposes not withstanding any copyright notation. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the NSF, DARPA, or the US Government. The authors would like to extend their gratitude to Elias Wright, Gian Mascioli, Kiara Pillay, Harrison Kay, Eliel Talo, Alexander Fagella and N. Cameron Russell for providing their valuable expertise and insight to this effort. The authors are also grateful to Eduard Hovy, Lorrie Cranor, Florian Schaub, Joel Reidenberg, Aditya Potukuchi and Igor Shalyminov for helpful discussions related to this work, and to the three anonymous reviewers of this draft for their constructive feedback. Finally, the authors would like to thank all crowdworkers who consented to participate in this study.


## References


Carole Cadwalladr and Emma Graham-Harrison. 2018. Revealed: 50 million facebook profiles harvested for cambridge analytica in major data breach. *The Guardian*, 17.

Fred H Cate. 2010. The limits of notice and choice. *IEEE Security & Privacy*, 8(2).

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. QuAC: Question answering in context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2174–2184, Brussels, Belgium. Association for Computational Linguistics.

Lorrie Faith Cranor. 2003. P3p: Making privacy policies more useful. *IEEE Security & Privacy*, 99(6):50–55.

Lorrie Faith Cranor. 2012. Necessary but not sufficient: Standardized mechanisms for privacy notice and choice. *J. on Telecomm. & High Tech. L.*, 10:273.

SS Deeb. 2005. The molecular basis of variation in human color vision. *Clinical genetics*, 67(5):369–377.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Phong-Khac Do, Huy-Tien Nguyen, Chien-Xuan Tran, Minh-Tien Nguyen, and Minh-Le Nguyen. 2017. Legal question answering using ranking svm and deep convolutional neural network. *arXiv preprint arXiv:1703.05320*.

Federal Trade Commission. 1998. Privacy online: A report to congress. *Washington, DC, June*, pages 10–11.

Minwei Feng, Bing Xiang, Michael R Glass, Lidan Wang, and Bowen Zhou. 2015. Applying deep learning to answer selection: A study and an open task. *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*.

Anette Frank, Hans-Ulrich Krieger, Feiyu Xu, Hans Uszkoreit, Berthold Crysmann, Brigitte Jörg, and Ulrich Schäfer. 2007. Question answering from structured knowledge sources. *Journal of Applied Logic*, 5(1):20–48.

Matthias Gallé, Athena Christofi, and Hady Elsahar. 2019. The case for a gdpr-specific annotated dataset of privacy policies. *AAAI Symposium on Privacy-Enhancing AI and HLT Technologies*.

Joshua Gluck, Florian Schaub, Amy Friedman, Hana Habib, Norman Sadeh, Lorrie Faith Cranor, and Yuvraj Agarwal. 2016. How short is too short? implications of length and framing on the effectiveness of privacy notices. In *12th Symposium on Usable Privacy and Security (SOUPS)*, pages 321–340.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112, New Orleans, Louisiana. Association for Computational Linguistics.

Hamza Harkous, Kassem Fawaz, Rémi Lebret, Florian Schaub, Kang G Shin, and Karl Aberer. 2018. Polisis: Automated analysis and presentation of privacy policies using deep learning. *arXiv preprint arXiv:1802.02561*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Priyank Jain, Manasi Gyanchandani, and Nilay Khare. 2016. Big data privacy: a technological perspective and review. *Journal of Big Data*, 3(1):25.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Divyansh Kaushik and Zachary C. Lipton. 2018. How much reading does reading comprehension require? a critical investigation of popular benchmarks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5010–5015, Brussels, Belgium. Association for Computational Linguistics.

Patrick Gage Kelley, Joanna Bresee, Lorrie Faith Cranor, and Robert W Reeder. 2009. A nutrition label for privacy. In *Proceedings of the 5th Symposium on Usable Privacy and Security*, page 4. ACM.

Mi-Young Kim, Ying Xu, and Randy Goebel. 2015. Applying a convolutional neural network to legal question answering. In *JSAI International Symposium on Artificial Intelligence*, pages 282–294. Springer.

Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gáabor Melis, and Edward Grefenstette. 2018. The narrativeqa reading comprehension challenge. *Transactions of the Association of Computational Linguistics*, 6:317–328.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Pedro Giovanni Leon, Ashwini Rao, Florian Schaub, Abigail Marsh, Lorrie Faith Cranor, and Norman Sadeh. 2015. Privacy and behavioral advertising: Towards meeting users' preferences. In *Symposium on usable privacy and security (SOUPS)*.

Frederick Liu, Shomir Wilson, Florian Schaub, and Norman Sadeh. 2016. Analyzing vocabulary intersections of expert annotations and topic models for data practices in privacy policies. In *2016 AAAI Fall Symposium Series*.

Yi-Hung Liu, Yen-Liang Chen, and Wu-Liang Ho. 2015. Predicting associated statutes for legal problems. *Information Processing & Management*, 51(1):194–211.

Aleecia M McDonald and Lorrie Faith Cranor. 2008. The cost of reading privacy policies. *ISJLP*, 4:543.

Anca Micheti, Jacquelyn Burkell, and Valerie Steeves. 2010. Fixing broken doors: Strategies for drafting privacy policies young people can understand. *Bulletin of Science, Technology & Society*, 30(2):130–143.

Diego Mollá and José Luis Vicedo. 2007. Question answering in restricted domains: An overview. *Computational Linguistics*, 33(1):41–61.

Alfredo Monroy, Hiram Calvo, and Alexander Gelbukh. 2009. Nlp for shallow question answering of legal documents using graphs. *Computational Linguistics and Intelligent Text Processing*, pages 498–508.

Kanthashree Mysore Sathyendra, Shomir Wilson, Florian Schaub, Sebastian Zimmeck, and Norman Sadeh. 2017. Identifying the provision of choices in privacy policy text. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2774–2779, Copenhagen, Denmark. Association for Computational Linguistics.

Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Rose, and Graham Neubig. 2018. Stress test evaluation for natural language inference. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2340–2353, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

OCDE OECD. 2004. The oecd principles of corporate governance. *Contaduría y Administración*, (216).

Alessandro Oltramari, Dhivya Piraviperumal, Florian Schaub, Shomir Wilson, Sushain Cherivirala, Thomas B Norton, N Cameron Russell, Peter Story, Joel Reidenberg, and Norman Sadeh. 2017. Privonto: A semantic framework for the analysis of privacy policies. *Semantic Web*, (Preprint):1–19.

Takeshi Onishi, Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. 2016. Who did what: A large-scale person-centered cloze dataset. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2230–2235, Austin, Texas. Association for Computational Linguistics.

P Howard Patrick. 1980. Privacy restrictions on transnational data flows: A comparison of the council of europe draft convention and oecd guidelines. *Jurimetrics J.*, 21:405.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis only baselines in natural language inference. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 180–191, New Orleans, Louisiana. Association for Computational Linguistics.

Paulo Quaresma and Irene Pimenta Rodrigues. 2005. A question answer system for legal information retrieval. In *JURIX*, pages 91–100.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Joel R Reidenberg, Travis Breaux, Lorrie Faith Cranor, Brian French, Amanda Grannis, James T Graves, Fei Liu, Aleecia McDonald, Thomas B Norton, and Rohan Ramanath. 2015. Disagreeable privacy policies: Mismatches between meaning and users' understanding. *Berkeley Tech. LJ*, 30:39.

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 193–203.

Florian Schaub, Rebecca Balebako, Adam L Durity, and Lorrie Faith Cranor. 2015. A design space for effective privacy notices. In *Eleventh Symposium On Usable Privacy and Security (SOUPS 2015)*, pages 1–17.

Peter Story, Sebastian Zimmeck, and Norman Sadeh. 2018. Which apps have privacy policies? In *Annual Privacy Forum*, pages 3–23. Springer.

Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2016. Improved representation learning for question answer matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 464–473, Berlin, Germany. Association for Computational Linguistics.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. NewsQA: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 191–200, Vancouver, Canada. Association for Computational Linguistics.

FTC US Federal Trade Commission et al. 2012. Protecting consumer privacy in an era of rapid change: Recommendations for businesses and policymakers. *FTC Report*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Paul Voigt and Axel Von dem Bussche. 2017. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing*.

Dirk Weissenborn, Georg Wiese, and Laura Seiffe. 2017. Making neural QA as simple as possible but not simpler. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 271–280, Vancouver, Canada. Association for Computational Linguistics.

S. Wilson, F. Schaub, F. Liu, K.M. Sathyendra, S. Zimmeck, R. Ramanath, F. Liu, N. Sadeh, and N.A. Smith. 2017. Analyzing privacy policies at scale: From crowdsourcing to automated annotations. *ACM Transactions on the Web*.

Shomir Wilson, Florian Schaub, Aswarth Abhilash Dara, Frederick Liu, Sushain Cherivirala, Pedro Giovanni Leon, Mads Schaarup Andersen, Sebastian Zimmeck, Kanthashree Mysore Sathyendra, N Cameron Russell, et al. 2016. The creation and analysis of a website privacy policy corpus. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1330–1340.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018.

# Stick to Facts: Towards Fidelity-oriented Product Description Generation

**Zhangming Chan**[1,2,*,†], **Xiuying Chen**[1,2,*], **Yongliang Wang**[3], **Juntao Li**[1,2], **Zhiqiang Zhang**[3], **Kun Gai**[3], **Dongyan Zhao**[1,2] and **Rui Yan**[1,2,‡]
[1] Center for Data Science, AAIS, Peking University
[2] Wangxuan Institute of Computer Technology, Peking University
[3] Alibaba Group, Beijing
{zhangming.chan,xy-chen,ljt,zhaody,ruiyan}@pku.edu.cn
{yongliang.wyl,zhang.zhiqiang,jingshi.gk}@alibaba-inc.com

## Abstract

Different from other text generation tasks, in product description generation, it is of vital importance to generate faithful descriptions that stick to the product attribute information. However, little attention has been paid to this problem. To bridge this gap we propose a model named Fidelity-oriented Product Description Generator (FPDG). FPDG takes the entity label of each word into account, since the product attribute information is always conveyed by entity words. Specifically, we first propose a Recurrent Neural Network (RNN) decoder based on the Entity-label-guided Long Short-Term Memory (ELSTM) cell, taking both the embedding and the entity label of each word as input. Second, we establish a keyword memory that stores the entity labels as keys and keywords as values, and FPDG will attend to keywords through attending to their entity labels. Experiments conducted a large-scale real-world product description dataset show that our model achieves the state-of-the-art performance in terms of both traditional generation metrics as well as human evaluations. Specifically, FPDG increases the fidelity of the generated descriptions by 25%.

## 1 Introduction

The effectiveness of automatic text generation has been proved in various natural language processing applications, such as neural machine translation (Luong et al., 2015; Zhou et al., 2017), dialogue generation (Tao et al., 2018a; Hu et al., 2019), and abstractive text summarization (Chen et al.; Gao et al., 2019). One such task, product description generation has also attracted considerable attention (Lipton et al., 2015; Wang et al., 2017; Zhang et al., 2019a). An accurate and attractive description not only helps customers make an informed decision but also improves the likelihood of purchase. Concretely, the product description generation task takes several keywords describing the attributes of a product as input, and then outputs fluent and attractive sentences that highlight the feature of this product.

| Input | high-waist; straight; jeans; blue; ZARA |
|---|---|
| Bad output | This ~~UNIQLO low-waist~~ jeans are suitable for the curvature of the waistline and will not slip when worn for a long time. The straight version and ~~black~~ color together makes you even slimmer. |
| Good output | This blue jeans look personalized and chic, making you more distinctive. With a high-waist design, the legs look even more slender, and the whole person looks taller. The straight design hides the fat in the legs and is visually slimmer. |

Table 1: Example of product description generation. The text with underline demonstrates faithful description, and text with deleteline demonstrates wrong description.

There is one intrinsic difference between product description generation and other generation tasks, such as story or poem generation (Li et al., 2018; Yao et al., 2019); the generated description has to be faithful to the product attributes. An example case is shown in Table 1, where the good product description matches the input information, while the bad description mistakes the brand and the style of the jeans. Our preliminary study reveals that 48% of the outputs from a state-of-the-art sequence-to-sequence system suffer from this problem. In the real-world e-commerce product description generation system, generating text with unfaithful information is unacceptable. A wrong brand name will damage the interests of advertisers, while a wrong product category might break the law by misleading consumers. Any kind of unfaithful description will bring huge economic

[*] Equal contribution. Ordering is decided by a coin flip.
[†] Contribution during internship at Alibaba Group.
[‡] Corresponding author.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4959–4968, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

losses to online platforms.

Though of great importance, no attention has been paid to this problem. Existing product description generators include (Chen et al., 2019; Zhang et al., 2019b), where they focus on generating personalized descriptions and pattern-controlled descriptions, respectively.

In this paper, we address the fidelity problem in generation tasks by developing a model named *Fidelity-oriented Product Description Generator* (FPDG), which takes keywords about the product attributes as inputs. Since product attribute information is always conveyed by entity words (up to 89.72% in input keywords), FPDG generates faithful product descriptions by taking into account the entity label of each word. Specifically, first, we propose an Entity-label-guided Long Short-Term Memory (ELSTM) as the cell in the decoder RNN, which takes the entity label of an input word, as well as the word itself as input. Then, we establish a keyword memory storing the input keywords with their entity labels. In each decoding step, the current hidden state of the ELSTM focuses on proper words in this keyword memory in regard to their entity categories. We also collect a large-scale real-world product description dataset from one of the largest e-commerce platforms in China. Extensive experiments conducted on this dataset show that FPDG outperforms the state-of-the-art baselines in terms of traditional generation metrics and human evaluations. Specifically, FPDG greatly improves the fidelity of generated description by 24.61%.

To our best knowledge, we are the first to explore the fidelity problem of product description generation. Besides, to tackle this problem, we propose an ELSTM and a keyword memory to incorporate entity label information, so as to generate more accurate descriptions.

## 2 Related Work

We detail related work on text generation, entity-related generation, and product description.

**Text generation.** Recently, sequence-to-sequence (Seq2Seq) neural network models have been widely used in NLG approaches. Their effectiveness has been demonstrated in a variety of text generation tasks, such as neural machine translation (Luong et al., 2015; Bahdanau et al., 2014; Wu et al., 2016), abstractive text summarization (See et al., 2017a; Hsu et al., 2018; Chen and Bansal, 2018), dialogue generation (Tao et al., 2018a; Xing et al., 2017), etc. Along another line, there are also works based on an attention mechanism. Vaswani et al. (2017) proposed a Transformer architecture that utilizes the self-attention mechanism and has achieved state-of-the-art results in neural machine translation. Since then, the attention mechanism has been used in a variety of tasks (Devlin et al., 2018; Fan et al., 2018; Zhou et al., 2018).

**Entity-related generation.** Named entity recognition (NER) is a fundamental component in language understanding and reasoning (Greenberg et al., 2018; Katiyar and Cardie, 2018). In (Ji et al., 2017), they proved that adding entity related information can improve the performance of language modeling. Building upon this work, in (Clark et al., 2018), they combined entity context with previous-sentence context, and demonstrated the importance of the latter in coherence test. Another line of related work generates recipes using neural networks to track and update entity representations (Bosselut et al., 2018). Different from the above works, we utilize entity labels as supplementary information to assist decoding in the text generation task.

**Product descriptions.** Quality product descriptions are critical for providing a competitive customer experience in an e-commerce platform. Due to its importance, automatically generating the product description has attracted considerable interests. Initial works include (Wang et al., 2017), which incorporates statistical methods with the template to generate product descriptions. With the development of neural networks, (Chen et al., 2019) explored a new way to generate personalized product descriptions by combining the power of neural networks and a knowledge base. (Zhang et al., 2019b) proposed a pointer-generator neural network to generate product description whose patterns are controlled.

In real-world product description generation application, however, the most important prerequisite is the fidelity of generated text, and to the best of our knowledge, no research has been conducted on this so far.

## 3 Problem Formulation

FPDG takes a list of keywords $X = \{x_1, ..., x_{T_X}\}$ as inputs, where $T_X$ is the number of keywords. These keywords are all about the important at-

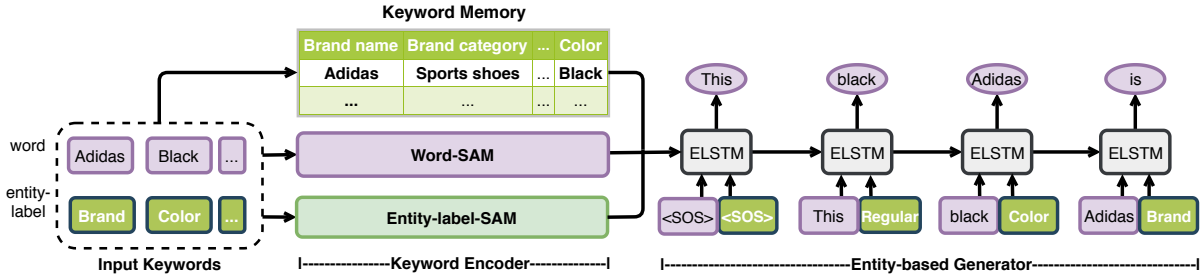


Figure 1: Overview of FPDG. Green denotes an entity label and purple denotes a word. We divide our model into two components: (1) The *Keyword Encoder* stores the word and its entity label in the token memory, and uses Self-Attention Modules (SAMs) to encode words and entity labels; (2) The *Entity-based Generator* generates product description based on the token memory and SAM encoders.

tributes of the product. The goal of FPDG is to generate a product description $\hat{Y} = \{\hat{y}_1, ..., \hat{y}_{T_{\hat{Y}}}\}$ that is not only grammatically correct but also consistent with the input information, such as the brand name and the fashion style. Essentially, FPDG tries to optimize the parameters to maximize the probability $P(Y|X) = \prod_{t=1}^{T_Y} P(y_t|X)$, where $Y = \{y_1, ..., y_{T_Y}\}$ is the ground truth answer.

## 4 Model

In this section, we introduce our *Fidelity-oriented Product Description Generator* (FPDG) model in detail. An overview of FPDG is shown in Figure 1 and can be split into two modules:

(1) *Keyword Encoder* (See § 4.1): We first use a key-value memory to store the entity-label as key and the corresponding word as value. To better learn the interaction between words, we employ two self-attention modules from Transformer (Vaswani et al., 2017) to model the input keywords and their entity-labels separately.

(2) *Entity-based Generator* (See § 4.2): We propose a recurrent decoder based on Entity-label-guided LSTM (ELSTM) to generate the product description.

### 4.1 Keyword Encoder

In the product description dataset, most of the input keywords are entity words (up to 89.72%), and the description text should be faithful to these entity words. Hence, we incorporate entity label information to improve the accuracy of the generated text. In this section, we introduce how to embed input keywords with entity label information.

To begin with, we use an embedding matrix $e$ to map a one-hot representation of each word in $x_i$ into a high-dimensional vector space. Since our input keywords have no order information, we use the Self-Attention Module (SAM) from Transformer (Vaswani et al., 2017) to model the temporal interactions between the words, instead of RNN-based encoder. We use three fully-connected layers to project $e(x_i)$ into three spaces, i.e., the query $q_i = F_q(e(x_i))$, the key $k_i = F_k(e(x_i))$ and the value $v_i = F_v(e(x_i))$. The attention module then takes $q_i$ to attend to each $k_\cdot$, and uses these attention distribution results $\alpha_{i,\cdot} \in \mathbb{R}^{T_X}$ as weights to obtain the weighted sum of $v_i$, as shown in Equation 2. Next, we add the original word representation $e(x_i)$ on $\beta_i$ as the residential connection layer, as shown in Equation 3:

$$\alpha_{i,j} = \frac{\exp(q_i k_j)}{\sum_{n=1}^{T_X} \exp(q_i k_n)}, \quad (1)$$

$$\beta_i = \sum_{j=1}^{T_X} \alpha_{i,j} v_j, \quad (2)$$

$$\hat{h}_i = e(x_i) + \beta_i, \quad (3)$$

where $\alpha_{i,j}$ denotes the attention weight of the $i$-th word on the $j$-th word. Finally, we apply a feed-forward layer on $\hat{h}_i$ to obtain the final word representation $h_i$:

$$h_i = \max(0, \hat{h}_i \cdot W_1 + b_1) \cdot W_2 + b_2, \quad (4)$$

where $W_1, W_2, b_1, b_2$ are all trainable parameters. We refer to the above process as:

$$h_i = \text{SAM}\left(F_q(x_i), F_k(x.), F_v(x.)\right). \quad (5)$$

In the meantime, we leverage the e-commerce-adapted AliNER[1] to label each word in the input such as "brand name" and "color". Non-entity words are labeled as "normal word". We denote $c_i$

[1] https://ai.aliyun.com/nlp/ner

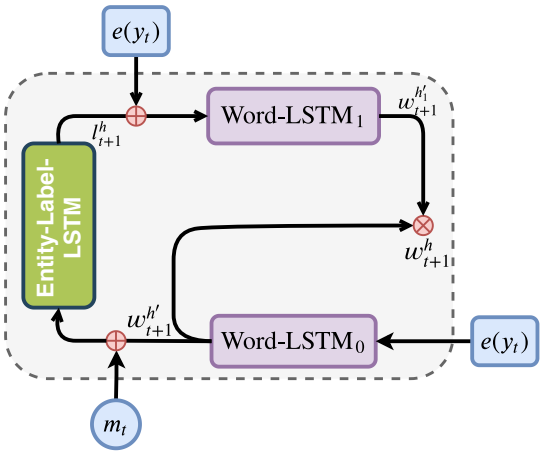


Figure 2: The structure of ELSTM, which is a hybrid of three LSTMs.

as the entity label for the $i$-th word. To better learn the interaction between these entity labels, we use a second SAM, taking $c.$ as input, and obtain the final representation for $c_i$ as $m_i$:

$$m_i = \text{SAM}\left(F_q(c_i), F_k(c.), F_v(c.)\right). \quad (6)$$

We also propose a key-value keyword memory that stores the label results as shown in Figure 1. The keys in the keyword memory are $T_C$ representations of different entity-labels, where $T_C$ is the number of entity categories. The values in the memory are self-attention representations of words that belong to each category. We denote the $k$-th entity-label in the keyword memory as $c^k$, and the $i$-th word belonging to this category as $e(x^k_i)$. This keyword memory will be incorporated into the generation process in §4.2.3.

### 4.2 Entity-based Generator

To incorporate entity label information into generation process, we propose a modified version of LSTM, named Entity-label-guided Long Short-Term Memory (ELSTM). We first introduce ELSTM and then introduce the RNN decoder based on ELSTM.

#### 4.2.1 ELSTM

As shown in Figure 3, ELSTM consists of three LSTMs, two Word-LSTMs and one Entity-label-LSTM. The hidden states of the two Word-LSTMs are integrated together, forming $w^h_t$, while the hidden state of Entity-Label-LSTM is $l^h_t$. The word hidden state $w^h_t$ attends to the SAM outputs to predict the next word, while the entity label hidden state $l^h_t$ is used to attend to the keyword memory and predict entity label of next word. In other words, ELSTM is a hybrid of three LSTMs with a deeper interaction between these cells. Overall, ELSTM takes two variables as input, the embedding of an input word $e(y_t)$ and its entity label $m_t$.

The structure of each LSTM in ELSTM is the same as original LSTM, thus, is omitted here due to space limitations. Next, we introduce the interaction in ELSTM in detail. As shown in Figure 2, Word-LSTM$_0$ takes the $e(y_t)$ as input, and outputs the initial word hidden state $w^{h'}_{t+1}$:

$$w^{h'}_{t+1} = \text{Word-LSTM}_0(w^h_t, e(y_t)). \quad (7)$$

Next we calculate the entity label hidden state by Entity-Label-LSTM, taking both $m_t$ and $w^{h'}_{t+1}$ as input:

$$l^{h'}_{t+1} = \text{Entity-Label-LSTM}(l^h_t, m_t + w^{h'}_{t+1}). \quad (8)$$

To further improve the accuracy of entity label prediction, we also apply another multi-layer projection, incorporating entity label context vector $c^m_t$ (introduced in §16) to obtain polished entity hidden state $l^h_{t+1}$:

$$l^h_{t+1} = FC([l^{h'}_{t+1}, c^m_t]), \quad (9)$$

where $[,]$ denotes the concatenation operation.

Finally, Word-LSTM$_1$ takes the entity-label hidden state $l^h_{t+1}$ and word embedding $e(y_t)$ as input, and outputs predicted word hidden state that contains entity label information:

$$w^{h'_1}_{t+1} = \text{Word-LSTM}_1(w^h_t, e(y_t) + l^h_{t+1}). \quad (10)$$

Note that it is possible that the predicted entity-label hidden state is not accurate enough. Hence, we apply a gate fusion combining $w^{h'_1}_{t+1}$ with initial word hidden state $w^{h'}_{t+1}$ to ensure the word hidden state quality:

$$\gamma = \sigma(FC([w^{h'_1}_{t+1}, w^{h'}_{t+1}])), \quad (11)$$

$$w^h_{t+1} = \gamma w^{h'_1}_{t+1} + (1-\gamma) w^{h'}_{t+1}. \quad (12)$$

The fusion result is $w^h_{t+1}$, *i.e.*, the updated word hidden state.

In this way, the entity label information and word information are fully interacted in ELSTM cell, while the hidden states are updated.

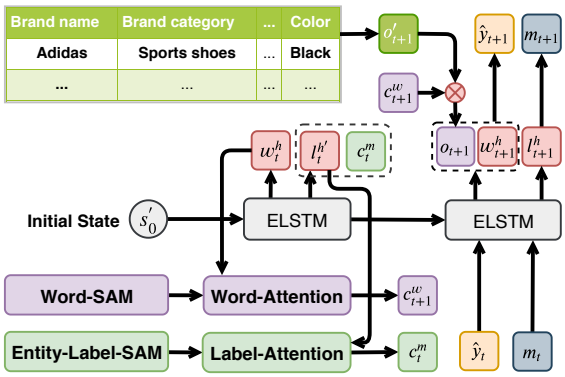


Figure 3: An overview of the description generator.

#### 4.2.2 ELSTM-based Attention Mechanism

Established on ELSTMs, we now have a new RNN decoder to generate descriptions, incorporating the two SAM encoders and the keyword memory. First, we apply an Bi-RNN to encode input keywords, and use its last hidden state as decoder initial hidden states, i.e., $w_0^h$ and $l_0^h$. The $t$-th decoding step is calculated as:

$$\begin{bmatrix} w_{t+1}^h \\ l_{t+1}^h \end{bmatrix} = \text{ELSTM}\left(\begin{bmatrix} w_t^h \\ l_t^h \end{bmatrix}, (e(y_t), m_t)\right). \quad (13)$$

Next, similar to the traditional attention mechanism from (Bahdanau et al., 2015), we summarize the input word representation $e(y_.)$ and entity label representation $m_.$ into the *word context vector* $c_t^w$ and *entity label vector* $c_t^m$, respectively. As for how to obtain $c_t^w$ and $c_t^m$, we use the two hidden states in ELSTM, i.e., $w_t^h$ and $l_t^h$ to attend to $h_.$ and $m_.$, respectively. Specifically, the entity label context vector $c_t^m$ is calculated as:

$$\hat{\gamma}_{t,i} = W_a \tanh\left(W_b l_t^h + W_h h_i\right), \quad (14)$$

$$\gamma_{t,i} = \exp\left(\hat{\gamma_{t,i}}\right) / \sum_{j=1}^{T_X} \exp\left(\hat{\gamma_{t,j}}\right), \quad (15)$$

$$c_t^m = \sum_{i=1}^{T_X} \gamma_{t,i} m_i. \quad (16)$$

The decoder state $l_t^h$ is used to attend to each entity label representation $m_i$, resulting in the attention distribution $\gamma_t \in \mathbb{R}^{T_X}$, as shown in Equation 15. Then we use the attention distribution $\gamma_t$ to obtain a weighted sum of the entity label representations $m_.$ as the word context vector $c_t^m$, shown in Equation 16. $c_t^w$ is obtained in a similarity by using hidden state $w_t^h$ attending to $e(y_.)$, and we omit the details for brevity. This two context vectors play different parts, and is introduced in §4.2.4.

#### 4.2.3 Incorporating Keyword Memory

So far, we have finished calculating the context vector. Next, we describe how to incorporate the guidance from the keyword memory. We first use the entity label hidden state to attend to the entity label keys in the keyword memory by gumbel softmax (Jang et al., 2016) (Equation 17), and then use the attention weights to obtain the weighted sum of self-attention representation of values in the memory (Equation 19):

$$\pi_t^{k'} = \frac{\exp((l_t^h W_e c^k + p^k)/\tau)}{\sum_{j=1}^{T_c} \exp((w_t^h W_e c^j + p^k)/\tau)}, \quad (17)$$

$$v_i^{k'} = \text{SAM}\left(F_q(e(x_i^k), F_k(e(x_.^k), F_v(e(x_.^k)\right), \quad (18)$$

$$o_{t+1}^{'} = \sum_{k=1}^{T_C}\left(\pi_t^k \sum_{i=1}^{T_{c^k}} (v_i^{k'})\right), \quad (19)$$

where $p^k = -\log(-\log(g^k)), g^k \sim U(0,1), \tau \in [0,1]$ is the softmax temperature. $T_{c_k}$ denotes the number of words in the input keywords that belong to the $k$-th entity category. We choose gumbel-softmax instead of regular softmax because a generated word can only belong to one entity category. In this way, FPDG first predicts the entity label of the predicted word and then uses $o_{t+1}^{'}$ to store the information of words that belong to this category.

#### 4.2.4 Projection Layers

Next, using a fusion gate $g_t$, $o_{t+1}^{'}$ is combined with word hidden state $c_{t+1}^w$ in a similar way in Equation 12 to obtain $o_{t+1}$. Finally, we obtain the final generation distribution $P_v$ over vocabulary:

$$P_{t+1}^v = \text{softmax}\left(W_v[o_{t+1}, w_{t+1}^h] + b_v\right). \quad (20)$$

We concatenate the memory vector $o_{t+1}$, the word context vector $c_{t+1}^w$, and the output of the decoder ELSTM $w_{t+1}^h$ as the input of the output projection layer.

Apart from predicting the next word, we also use the entity label hidden state $l_t^h$ to predict the entity label of the next word as an auxiliary task. The distribution $P_{t+1}^e$ over entity categories is calculated as:

$$P_{t+1}^e = \text{softmax}(W_e[l_{t+1}^{h'}, c_{t+1}^m] + b_e) \quad (21)$$

We use negative log-likelihood as loss function:

$$\mathcal{L} = -(\sum_{t=1}^{T_{\hat{Y}}} \log P_v(y_t) + \lambda \cdot \sum_{t=1}^{T_{\hat{Y}}} \log P_e(m_t)), \quad (22)$$

where $\lambda$ is the weight of entity label prediction loss. The gradient descent method is employed to update all parameters and minimize this loss function.

## 5 Experimental Setup

### 5.1 Research Questions

We list four research questions that guide the experiments: **RQ1** (See § 6.1): What is the overall performance of FPDG? Are generated descriptions faithful? **RQ2** (See § 6.2): What is the effect of each module in FPDG? **RQ3** (See § 6.3): How does the keyword memory work in each decoding step? **RQ4** (See § 6.4): Can ELSTM successfully capture entity label information?

### 5.2 Dataset

We collect a large-scale real-world product description dataset from one of the largest e-commerce platforms in China[2]. The inputs are keywords about the product attributes selected by sellers on the platform, and the product descriptions are written by professional experts that are good at marketing. Overall, there are 404,000 training samples, and 5,000 validation and test samples. On average, there are 10 keywords in the input, and 63 words in the product description. 89.72% inputs are entity words.

### 5.3 Evaluation Metrics

Following (Fu et al., 2017), we use the evaluation package of (Chen et al., 2015), which includes BLEU-1, BLEU-2, BLEU-3, BLEU-4, METEOR and ROUGE-L. BLEU is a popular machine translation metric that analyzes the co-occurrences of n-grams between the candidate and reference sentences. METEOR is calculated by generating an alignment between the words in the candidate and reference sentences, with an aim of 1:1 correspondence. ROUGE-L is a metric designed to evaluate text summarization algorithms.

(Tao et al., 2018b) notes that only using the BLEU metric to evaluate text quality can be misleading. Therefore, we also evaluate our model by human evaluation. Three highly educated participants are asked to score 100 randomly sampled summaries generated by Pointer-Gen and FPDG. We chose Pointer-Gen since its performance is relatively higher than other baselines. The statistical significance of observed differences between the performance of two runs are tested using a two-tailed paired t-test and is denoted using $^{\blacktriangle}$(or $^{\blacktriangledown}$) for strong significance for $\alpha = 0.01$.

### 5.4 Comparison Methods

We first conduct an ablation study to prove the effectiveness of each module in FPDG. Model w/o ELSTM is implemented with only keys in the memory because there are no entity representations as values. Then, to evaluate the performance of our proposed dataset and model, we compare it with the following baselines:

(1) **Seq2Seq**: The sequence-to-sequence framework (Sutskever et al., 2014) is one of the initial works proposed for the language generation task.

(2) **Pointer-Gen**: A sequence-to-sequence framework with pointer and coverage mechanism proposed in (See et al., 2017b).

(3) **Conv-Seq2seq**: A model combining the convolutional neural networks and the sequence-to-sequence network (Gehring et al., 2017).

(4) **FTSum**: A faithful summarization model proposed in (Cao et al., 2018), which leverages open information extraction and dependency parse technologies to extract actual fact descriptions from the source text.

(5) **Transformer**: A network architecture that solely based on attention mechanisms, dispensing with recurrence and convolutions entirely (Vaswani et al., 2017).

(6) **PCPG**: A pattern-controlled product description generation model proposed in (Zhang et al., 2019b). We adapt the model for our scenario.

To verify whether the performance improvement is obtained by adding additional entity label inputs, we directly concatenate the word embedding with the entity embedding as input for these baselines, denoted as "with Entity Embedding" in Table 2.

### 5.5 Implementation Details

We implement our experiments in Pytorch[3] on Tesla V100 GPUs[4]. The word and entity-label embedding dimensions are set to 256. The number of hidden units and the entity-label hidden size are also set to 256. All inputs were padded with zeros to a maximum keyword number of the batch. There are 36 categories of entity labels together.

[2] https://www.taobao.com/

[3] https://pytorch.org/

[4] Our model will be incorporated into Chuangyi-Taobao (https://chuangyi.taobao.com/pages/smartPhrase).

| Models | BLEU Metric | | | | METEOR | ROUGE-L |
|---|---|---|---|---|---|---|
| | BLEU-1 | BLEU-2 | BLEU-3 | BLEU-4 | | |
| Seq2seq | 30.10 | 13.61 | 7.564 | 4.824 | 14.08 | 25.04 |
| with Entity Embedding | 30.46 | 13.77 | 7.722 | 5.012 | 14.09 | 24.96 |
| Pointer-Gen | 30.66 | 13.84 | 7.933 | 5.278 | 14.10 | 24.93 |
| with Entity Embedding | 30.92 | 13.92 | 7.988 | 5.219 | 14.14 | 24.85 |
| Conv-Seq2seq | 28.22 | 12.15 | 6.393 | 4.012 | 13.72 | 23.92 |
| with Entity Embedding | 28.23 | 12.22 | 6.401 | 4.012 | 13.75 | 24.03 |
| FTSum | 30.33 | 13.57 | 7.563 | 4.689 | 13.86 | 24.63 |
| with Entity Embedding | 30.47 | 13.61 | 7.611 | 4.937 | 13.97 | 24.82 |
| Transformer | 29.11 | 12.77 | 7.116 | 4.620 | 13.95 | 23.36 |
| with Entity Embedding | 29.37 | 12.64 | 6.852 | 4.315 | 13.88 | 23.73 |
| PCPG(including Entity) | 29.46 | 13.02 | 7.259 | 4.620 | 13.72 | 24.50 |
| **Our FPDG** | **32.26** | **14.79** | **8.472** | **5.629** | **15.17** | **25.27** |
| w/o KW-MEM | 31.85 | 14.36 | 8.134 | 5.351 | 15.01 | 25.09 |
| w/o ELSTM | 31.41 | 14.57 | 8.430 | 5.630 | 14.98 | 25.25 |

Table 2: RQ1: Comparison between baselines.

$\lambda$ is set to 0 in the first 500 steps, and 0.6 in the rest of the training process. We performed mini-batch cross-entropy training with a batch size of 256 documents for 15 training epochs. we set the minimum encoding step to 15 and maximum decoding step size to 70. During decoding, we employ beam search with a beam size of 4 to generate a more fluent sentence. It took around 6 hrs on GPUs for training. After each epoch, we evaluated our model on the validation set and chose the best performing model for the test set. We use the Adam optimizer (Duchi et al., 2010) as our optimizing algorithm and the learning rate is 1e-3.

## 6 Experimental Results

### 6.1 Overall Performance

For research question **RQ1**, we examine the performance of our model and baselines in terms of BLEU, as shown in Table 2. Firstly, among all baselines, Pointer-Gen obtains the best performance, outperforming the worst baseline Conv-Seq2Seq by 2.44 in BLEU-1. Secondly, directly concatenating the entity label embedding with the word embedding does not bring much help, only leading to an improvement of 0.26 in BLEU-1 for the Pointer-Gen model. Finally, our model outperforms all baselines for all metrics, outperforming the strongest baseline Pointer-Gen, by 5.22%, 6.86%, 6.79%, and 6.65% in terms of BLEU-1, BLEU-2, BLEU-3, and BLEU-4, respectively.

| | Fluency | Informativity | Fidelity |
|---|---|---|---|
| Pointer-Gen | 2.23 | 1.84 | 1.91 |
| FPDG | **2.46**$^{\blacktriangle}$ | **2.19**$^{\blacktriangle}$ | **2.38**$^{\blacktriangle}$ |

Table 3: RQ1: Human evaluation comparison with Pointer-Gen baseline.

As for human evaluation, we ask three highly educated participants to rank generated summaries in terms of fluency, informativity, and fidelity. The rating score ranges from 1 to 3 with 3 being the best. The results are shown in Table 3, where FPDG outperforms Pointer-Gen by 10.31% and 19.02% in terms of fluency and informativity, and, specifically, FPDG greatly improves the fidelity value by 24.61%. We also conduct the paired student t-test between our model and Pointer-Gen, and the result demonstrates the significance of the above results. The kappa statistics are 0.35 and 0.49, respectively, which indicates fair and moderate agreement between annotators[5].

[5](Landis and Koch, 1977) characterize kappa values $< 0$ as no agreement, 0-0.20 as slight, 0.21-0.40 as fair, 0.41-0.60 as moderate, 0.61-0.80 as substantial, and 0.81-1 as almost

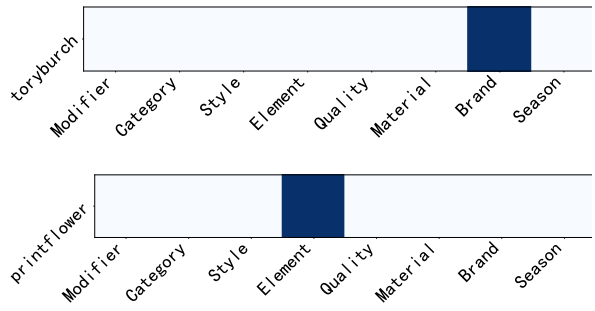


Figure 4: RQ3: Visualizations of entity-label-attention when generating the word on the left.

### 6.2 Ablation Study

Next, we turn to research question **RQ2**. We conduct ablation tests on the usage of the keyword memory and ELSTM, corresponding to FPDG w/o KW-MEM and ELSTM, respectively. The ROUGE score result is shown at the bottom of Table 2. Performances of all ablation models are worse than that of FPDG in terms of almost all metrics, which demonstrates the necessity of each module in FPDG. Specifically, ELSTM makes the greatest contribution to FPDG, improving the BLEU-1, BLEU-2 scores by 2.71% and 1.51%.

### 6.3 Analysis of Keyword Memory

We then address **RQ3** by analyzing the entity-label-level attention on the keyword memory. Two representative cases are shown in Figure 4. The figure in the above is the attention map when generating the word "toryburch", and the bottom figure is when generating the word "printflower". The darker the color, the higher the attention. Due to limited space, we omit irrelevant entity categories. When generating "toryburch", which is a brand name, the entity-label-level attention pays most attention to the "Brand" entity label, and when generating "flower", which is a style element, it mostly pays attention to "Element". This example demonstrates the effectiveness of the entity-label-level attention.

### 6.4 Analysis of ELSTM

We now turn to **RQ4**; whether or not the ELSTM can capture entity label information. We examine this question by verifying whether ELSTM can predict the entity label of the generated word. The accuracy of the predicted entity label is calculated in a teacher-forcing style, i.e., each ELSTM takes the ground truth entity label and word as input, and outputs the entity label of the next word. We employ recall at position $k$ in $n$ candidates ($R_n@k$) as evaluation metrics. Over the whole test dataset, $R_1@36$ is 64.12%, $R_2@36$ is 80.86%, and $R_3@36$ is 94.02%, which means ELSTM can capture the entity label information to a great extent and guide the word generation.

We also show a case study in Table 4. The description generated by Pointer-Gen introduces the Prada bag as a "clamshell bag has a good anti-theft effect", which is contrary to the fact. While our model generates the faithful description: "The opening and closing design of the zipper smoothes the stroke".

| | |
|---|---|
| Input: 正品; Prada;/;普拉达; 男包; 尼龙; 双肩背包; 男士; 休闲; 旅行包; 电脑包; 登山包(authentic; Prada; man bag; leisure; travel bag; laptop bag; backpack) | |
| Reference | 这款来自prada的拉链双肩包。采用纯色设计，包面上的字母印花图案十分显眼，令人眼前一亮，瞬间脱颖而出。开合双拉链处理，使用顺滑，方便拿取物品。大容量的包包，为物品的放置提供足够的空间。(This zippered shoulder bag from Prada. With a solid color design, the letter print on the surface of the bag is very conspicuous, making it instantly stand out. Open and close double zipper makes it easy to use take items smoothly. Large-capacity bag provides enough space for the placement of items.) |
| Pointer-Gen | 这款来自prada的翻盖双肩包。采用了翻盖的设计，具有很好的防盗效果，让你的出行更加安全可靠。大容量的包型设计，满足日常的生活所需，让你的出行更加方便快捷。(This flipover bag from Prada. The ~~clamshell design has a good anti-theft effect, making your trip safer and more reliable~~. The large-capacity package design meets the needs of everyday life, making your travel more convenient and quick. |
| FPDG | 这款来自prada的纯色双肩包。纯色的设计，给人一种大气又干练的感觉，打造时下流行的简约风格。拉链的开合设计，划拉顺畅，增加物品安全性。大容量的包型，既能容纳生活物品，还让携带更加的便捷。(This solid color backpack from Prada. The solid color design gives people an atmosphere and a sense of exquisiteness, creating a simple style that is popular nowadays. The opening and closing design of the zipper smoothes the stroke and increases the safety of the item. Large-capacity package can accommodate living items and make it easier to carry. |

Table 4: Examples of the generated answers by Pointer-Gen and FPDG. The text with underline demonstrates faithful description, and text with deleteline demonstrates wrong description.

## 7 Conclusion and Future Work

In this paper, we explore the fidelity problem in product description generation. To tackle this

perfect agreement.

challenge, based on the consideration that product attribute information is typically conveyed by entity words, we incorporate the entity label information of each word to enable the model to have a better understanding of the words and better focus on key information. Specifically, we propose an Entity-label-guided Long Short-Term Memory (ELSTM) and a token memory to store and capture the entity label information of each word. Our model outperforms state-of-the-art methods in terms of BLEU and human evaluations by a large margin. In the near future, we aim to fully prevent the generation of unfaithful descriptions and bring FPDG online.

## Acknowledgments

We would like to thank the reviewers for their constructive comments. This work was supported by the National Key Research and Development Program of China (No. 2017YFC0804001), the National Science Foundation of China (NSFC No. 61876196 and NSFC No. 61672058). Rui Yan was sponsored by Alibaba Innovative Research (AIR) Grant. Rui Yan is the corresponding author.

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In ICLR.

Antoine Bosselut, Omer Levy, Ari Holtzman, Corin Ennis, Dieter Fox, and Yejin Choi. 2018. Simulating action dynamics with neural process networks. ICLR.

Ziqiang Cao, Furu Wei, Wenjie Li, and Sujian Li. 2018. Faithful to the original: Fact aware neural abstractive summarization. In AAAI.

Qibin Chen, Junyang Lin, Yichang Zhang, Hongxia Yang, Jingren Zhou, and Jie Tang. 2019. Towards knowledge-based personalized product description generation in e-commerce. arXiv preprint arXiv:1903.12457.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. 2015. Microsoft coco captions: Data collection and evaluation server. arXiv preprint arXiv:1504.00325.

Xiuying Chen, Zhangming Chan, Shen Gao, Meng-Hsuan Yu, Dongyan Zhao, and Rui Yan. Learning towards abstractive timeline summarization. In IJCAI.

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. In ACL, pages 675–686.

Elizabeth Clark, Yangfeng Ji, and Noah A Smith. 2018. Neural text generation in stories using entity representations as context. In NAACL, pages 2250–2260.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

John C. Duchi, Elad Hazan, and Yoram Singer. 2010. Adaptive subgradient methods for online learning and stochastic optimization. JMLR, 12:2121–2159.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In ACL, pages 889–898.

Kun Fu, Junqi Jin, Runpeng Cui, Fei Sha, and Changshui Zhang. 2017. Aligning where to see and what to tell: image captioning with region-based attention and scene-specific contexts. PAMI, 39(12):2321–2334.

Shen Gao, Xiuying Chen, Piji Li, Zhaochun Ren, Lidong Bing, Dongyan Zhao, and Rui Yan. 2019. Abstractive text summarization by incorporating reader comments. In AAAI.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In ICML, pages 1243–1252. JMLR. org.

Nathan Greenberg, Trapit Bansal, Patrick Verga, and Andrew McCallum. 2018. Marginal likelihood training of bilstm-crf for biomedical named entity recognition from disjoint label sets. In EMNLP, pages 2824–2829.

Wan-Ting Hsu, Chieh-Kai Lin, Ming-Ying Lee, Kerui Min, Jing Tang, and Min Sun. 2018. A unified model for extractive and abstractive summarization using inconsistency loss. In ACL, pages 132–141.

Wenpeng Hu, Zhangming Chan, Bing Liu, Dongyan Zhao, Jinwen Ma, and Rui Yan. 2019. Gsn: A graph-structured network for multi-party dialogues. arXiv preprint arXiv:1905.13637.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. ICLR.

Yangfeng Ji, Chenhao Tan, Sebastian Martschat, Yejin Choi, and Noah A Smith. 2017. Dynamic entity representations in neural language models. In EMNLP, pages 1830–1839.

Arzoo Katiyar and Claire Cardie. 2018. Nested named entity recognition revisited. In ACL, pages 861–871.

J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. biometrics, pages 159–174.

Juntao Li, Yan Song, Haisong Zhang, Dongmin Chen, Shuming Shi, Dongyan Zhao, and Rui Yan. 2018. Generating classical chinese poems via conditional variational autoencoder and adversarial training. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3890–3900.

Zachary C Lipton, Sharad Vikram, and Julian McAuley. 2015. Capturing meaning in product reviews with character-level generative text models. arXiv preprint arXiv:1511.03683.

Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the rare word problem in neural machine translation. In ACL, volume 1, pages 11–19.

Abigail See, Peter J Liu, and Christopher D Manning. 2017a. Get to the point: Summarization with pointer-generator networks. In ACL, pages 1073–1083.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017b. Get to the point: Summarization with pointer-generator networks. pages 1073–1083. ACL.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In NIPS.

Chongyang Tao, Shen Gao, Mingyue Shang, Wei Wu, Dongyan Zhao, and Rui Yan. 2018a. Get the point of my utterance! learning towards effective responses with multi-head attention mechanism. In IJCAI, pages 4418–4424.

Chongyang Tao, Lili Mou, Dongyan Zhao, and Rui Yan. 2018b. Ruber: An unsupervised method for automatic evaluation of open-domain dialog systems. In AAAI.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In NIPS, pages 5998–6008.

Jinpeng Wang, Yutai Hou, Jing Liu, Yunbo Cao, and Chin-Yew Lin. 2017. A statistical framework for product description generation. In IJCAI, pages 187–192.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144.

Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2017. Topic aware neural response generation. In AAAI.

Lili Yao, Nanyun Peng, Ralph Weischedel, Kevin Knight, Dongyan Zhao, and Rui Yan. 2019. Plan-and-write: Towards better automatic storytelling. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pages 7378–7385.

Tao Zhang, Jin Zhang, Chengfu Huo, and Weijun Ren. 2019a. Automatic generation of pattern-controlled product description in e-commerce. In The World Wide Web Conference, pages 2355–2365. ACM.

Tao Zhang, Jin Zhang, Chengfu Huo, and Weijun Ren. 2019b. Automatic generation of pattern-controlled product description in e-commerce. In WWW, pages 2355–2365.

Long Zhou, Wenpeng Hu, Jiajun Zhang, and Chengqing Zong. 2017. Neural system combination for machine translation. arXiv preprint arXiv:1704.06393.

Xiangyang Zhou, Lu Li, Daxiang Dong, Yi Liu, Ying Chen, Wayne Xin Zhao, Dianhai Yu, and Hua Wu. 2018. Multi-turn response selection for chatbots with deep attention matching network. In ACL, pages 1118–1127.

# Fine-Grained Entity Typing via Hierarchical Multi Graph Convolutional Networks

**Hailong Jin**[1,2,3], **Lei Hou**[1,2,3]*, **Juanzi Li**[1,2,3], **Tiansi Dong**[4]
[1]Department of Computer Science and Technology, Tsinghua University
[2]KIRC, Institute for Artificial Intelligence, Tsinghua University
[3]Beijing National Research Center for Information Science and Technology
[4]B-IT, University of Bonn, Bonn, Germany
{jinhl15@mails., houlei@, lijuanzi@}tsinghua.edu.cn
dongt@bit.uni-bonn.de

## Abstract

This paper addresses the problem of inferring the fine-grained type of an entity from a knowledge base. We convert this problem into the task of graph-based semi-supervised classification, and propose Hierarchical Multi Graph Convolutional Network (**HMGCN**), a novel Deep Learning architecture to tackle this problem. We construct three kinds of connectivity matrices to capture different kinds of semantic correlations between entities. A recursive regularization is proposed to model the *subClassOf* relations between types in given type hierarchy. Extensive experiments with two large-scale public datasets show that our proposed method significantly outperforms four state-of-the-art methods.

## 1 Introduction

Nowadays, Knowledge Base ($\mathcal{KB}$ for short) attracts increasing research interests in various areas. One of the fundamental components in $\mathcal{KB}$ is the information of entity type, which clusters a group of entities with same properties, and is the glue that holds our mental world together (Murphy, 2004). Traditional entity typing focuses on a small set of types, such as *Person*, *Location* and *Organization* (Ratinov and Roth, 2009; Nadeau and Sekine, 2007), while Fine-Grained Entity Typing assigns more specific types to an entity, which normally forms a *type-path* in the type hierarchy in $\mathcal{KB}$ (Ren et al., 2016). For example, *Messi* is classified as having the path of following types: $FootballPlayer \subset Athlete \subset Person$. Types in $\mathcal{KB}$ are usually organized as a hierarchical structure, namely *type hierarchy*. Unfortunately, most $\mathcal{KB}$s are incomplete and lack of type information. For example, in DBpedia, the average number of types is only 2.9 (5,044,223 entities with 14,760,728 types), while 36.53% entities do not have type information. As Figure 1 shows, for each unlabeled entity, we will fully utilize its textual description, category and property to predict missing types. In this paper, we aim at assigning Fine-grained Entity Typing for entities in $\mathcal{KB}$ (such as Wikipedia and DBpedia).

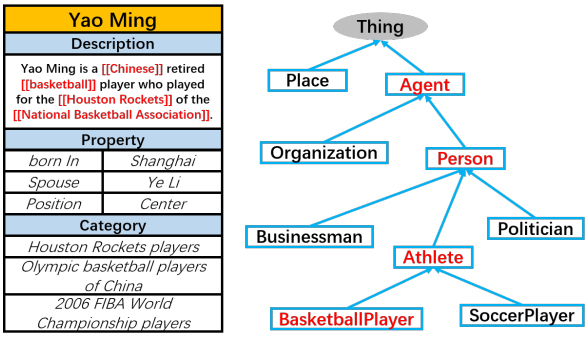


Figure 1: Entity (i.e., Yao Ming) with textual description (with anchor text in red), property (e.g., bornIn Shanghai) and category (e.g., Houston Rockets players). Yao Ming is associated with a type-path: $BasketballPlayer \subset Athlete \subset Person \subset Agent \subset Thing$

Many researches have been carried out in this field. State-of-the-art methods normally learn distributed representation for each entity, and apply a multi-label classification model to make type inference. For example, (Neelakantan and Chang, 2015) and (Xu et al., 2016) exploit various kinds of information to construct the feature representation of an entity, such as entity textual description, property and category. After that, a predict function is learned to infer whether entity $e$ is an instance of type $t$. (Yaghoobzadeh et al., 2017) focus on the names of entities and the context of entity mentions (in text), and design two scoring models for pairs of entities and types. All these works ignore the internal relations among entities, and assign types for each entity in isolation. (Jin et al., 2018) viewed the internal relations

* Corresponding author



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4969–4978, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

among entities as the structural information, and constructed an Entity Graph, further proposed a network embedding framework to learn the correlation among entities. It will be profitable to enrich entity features to entity graph structures, as recent studies have suggested to bring both node feature and graph structure information in a convolutional manner (Defferrard et al., 2016; Atwood and Towsley, 2016). Among these works, graph convolutional network (GCN) is the most widely-used model which can directly operate on graphs with arbitrary sizes and shapes (Kipf and Welling, 2016). The inputs of a GCN are feature vectors of nodes and graph structure. The most significant aspect of GCN is information diffusion, with which node's feature vector can be enriched by all the feature vectors of its neighbors.

Here, we convert entities in $\mathcal{KB}$ into three semantic graphs, each encoding a specific kind of correlation among entities, as follows: $\mathbf{A}_{co}$ a graph of the topical co-occurrence relations among entities, $\mathbf{A}_{cat}$ a category-based graph encoding the category-proximity between entities, and $\mathbf{A}_{prop}$ a property-based graph encoding the property-proximity between entities. We propose Hierarchical Multi Graph Convolutional Network (**HMGCN**), a novel deep learning architecture consisting of three Graph Convolutional Networks (GCNs): $\mathbf{GCN}_{co}$ and $\mathbf{GCN}_{cat}$ and $\mathbf{GCN}_{prop}$. Each connectivity matrix is fed to its corresponding GCN model, and the three GCN models are learned by shared parameters and consistency regularization. We adopt a simple and effective recursive regularization to deal with *subClassOf* relations between types. The main contributions of this paper are as follows:

- Graph convolutional network is applied for fine-grained entity typing task, which effectively integrates entity feature and structural information.
- Multi connectivity matrices are constructed to encode different kinds of semantic relatedness between entities. A recursive regularization is proposed to model *subClassOf* relations between types.
- Extensive experiments show that our proposed method significantly outperforms four state-of-the-art methods, with 1.7% and 1.3% improvement in Mi-F1 and Ma-F1 (on FIGER dataset), respectively.

The rest of this paper is organized as follows. Section 2 formally defines the problem of Fine-Grained Entity Typing in knowledge bases. Section 3 describes the proposed approach in detail. Section 4 reports a number of experiments with evaluations. Section 5 outlines some related works. Section 6 concludes our work.

## 2 Problem Formulation

In this section, we formally define the problem of *Fine-Grained Entity Typing* in $\mathcal{KB}$.

**Definition 1** ***Knowledge Base*** *$\mathcal{KB} = (\mathcal{E}, \mathcal{T}, \mathcal{R})$, where $\mathcal{E}$, $\mathcal{T}$ and $\mathcal{R}$ are sets of entities, types and isA relations, respectively. $|\mathcal{E}| = N$ and $|\mathcal{T}| = K$. $\mathcal{R} = \mathcal{R}_i \cup \mathcal{R}_s$ consists of instanceOf and subClassOf triples. $\mathcal{R}_i = \{(e_i, isA, t_j) | e_i \in \mathcal{E}, t_j \in \mathcal{T}\}$ collects all instanceOf triples and $\mathcal{R}_s = \{(t_i, isA, t_j) | t_i, t_j \in \mathcal{T}\}$ collects all subClassOf triples.*

Note that not each entity in a knowledge base $\mathcal{KB}$ is assigned with types. We split the entity set $\mathcal{E}$ into two subsets: $\mathcal{E}^l$ and $\mathcal{E}^u$, $\mathcal{E} = \mathcal{E}^l \cup \mathcal{E}^u$, $\mathcal{E}^l \cap \mathcal{E}^u = \emptyset$. $\mathcal{E}^l$ denotes the entity set whose member is assigned with type information. $\mathcal{E}^u$ denotes the entity set whose member has no type information. $|\mathcal{E}^l| = N^l$, $|\mathcal{E}^u| = N^u$. Types in $\mathcal{T}$ form a tree or a directed acyclic graph (DAG) via *subClassOf* relations, we refer it as type hierarchy $\mathcal{H} = (\mathcal{T}, \mathcal{R}_s)$.

For each entity, we distinguish three kinds of features, namely, *Text Description*, *Category*, and *Property*, which are beneficial to *Fine-Grained Entity Typing* task.

- *Text Description* is a succinct summary of an entity, providing valuable clues to predict entity types. As Figure 1 shows, there are a large number of anchor texts in textual description.
- *Category* refers, in Wikipedia, actually to tags/topics that group entities on similar subjects. For instance, *Yao Ming* in Wikipedia has category *Olympic basketball players of China*, which is the topic of the entity instead of its type. *Category* can be a useful information to infer missing type information.
- *Property* of entities is important cue in type inference. For instance, from *Yao Ming*'s property *playing position*, one may infer that *Yao Ming* is an *Athlete*.

Besides entity features above, there are many kinds of semantic correlations between entities, which organize entities into graph structures. We refer it as **entity graph**. Each kind of semantic correlation corresponds to a kind of connectivity matrix. There are many ways to construct entity graph via different connectivity manner, which will be discussed in Section 3.

**Definition 2** ***Entity Graph*** $\mathcal{G} = (\mathbf{A}, \mathbf{X}, \mathbf{Y})$. $\mathbf{A} \in \mathbb{R}^{N \times N}$ *is the connectivity matrix representing a kind of link relation between entities.* $\mathbf{X} \in \mathbb{R}^{N \times M}$ *is the feature matrix representing inherent features for all entities, where M is the number of features. Each row in* $\mathbf{X}$ *refers to an entity's feature vector* $\mathbf{X}_i$. $\mathbf{X}$ *can either be binary or take any real value.* $\mathbf{Y} \in \mathbb{R}^{N^l \times K}$ *is the type matrix collecting type information for entities in* $\mathcal{E}^l$. *Each row in* $\mathbf{Y}$ *refers to the instanceOf relations between an entity and all types.* $\mathbf{Y}_{ij} = 1$ *if* $e_i$ *has been assigned with type* $t_j$, *otherwise 0.*

The task of Fine-Grained Entity Typing can be formally described as follows: Given a partially-labeled Entity Graph $\mathcal{G} = (\mathbf{A}, \mathbf{X}, \mathbf{Y})$, we aim at learning a type predictor from $\mathbf{Y}$, which comprehensively takes graph structure and entity features into account. Then we utilize the learned type predictor to predict the missing *instanceOf* relations in $\mathcal{E}^u$, i.e., whether $(e_i, instanceOf, t_j)$ is true. In this way, we convert the task of Fine-Grained Entity Typing into a task of graph-based semi-supervised classification.

## 3 Hierarchical Multi Graph Convolutional Network (HMGCN)

State-of-the-art methods convert entities in $\mathcal{KB}$ into entity graph, then apply graph-based algorithm. The difficulties lie in two facts:

- Firstly, it is hard to effectively integrate entity features and structural information. Although network embedding methods, in terms of attributed network embedding, can take both graph structure and features into consideration (Huang et al., 2017), the node features are usually only served as auxiliary information in structure learning (Yang et al., 2016; Liao et al., 2017).

- Secondly, it is not easy to effectively integrate different connectivity matrix which capture different kinds of structural information (Wang et al., 2018; Zhuang and Ma, 2018).

We propose a Hierarchical Multi GCN model (**HMGCN**) to encode heterogeneous structural information in an ensemble manner, as illustrated in Figure 2. We construct three undirected entity graphs to capture different kinds of semantic correlations between entities, i.e., co-occurrence graph $\mathbf{A}_{co}$, category-based graph $\mathbf{A}_{cat}$, and property-based graph $\mathbf{A}_{prop}$. $\mathbf{A}_{co}$ encodes the topical relevance between entities, and is derived from textual anchor texts. $\mathbf{A}_{cat}$ is constructed through similarity computation based on category information, based on the assumption that entities with similar categories tend to have the same type. In the same way, we construct property-based graph $\mathbf{A}_{prop}$. Each entity graph is fed to corresponding GCN model, namely, $\mathbf{GCN}_{co}$, $\mathbf{GCN}_{cat}$ and $\mathbf{GCN}_{prop}$. These models use shared parameters and unsupervised consistency regularization, so that they can jointly consider opinions from three semantic perspectives. To achieve this, a simple and effective recursive regularization is adopted to deal with the *subClassOf* relations between types.

### 3.1 GCN-Based Entity Type Classification

GCN model consists of multiple stacked GCN layers. Given the input feature matrix $\mathbf{X} \in \mathbb{R}^{N \times M}$ and adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$, the output of the $i$-th hidden layer of the network $\mathbf{Z}^{(i)}$ is defined as:

$$\mathbf{Z}^{(i)} = \sigma(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{Z}^{(i-1)}\mathbf{W}^{(i)}) \quad (1)$$

$\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I_n}$ is the adjacency matrix with self-loops, where $\mathbf{I_n} \in \mathbb{R}^{N \times N}$ is the identity matrix. $\tilde{\mathbf{D}}_{ii} = \sum_j \tilde{\mathbf{A}}_{ij}$. Accordingly, $\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$ is the normalized adjacency matrix. $\mathbf{Z}_A^{(i-1)}$ is the output of the $(i-1)$-th layer, and $\mathbf{Z}^{(0)} = \mathbf{X}$. $\mathbf{W}^{(i)}$ are the trainable parameters of the network, and $\sigma(\cdot)$ denotes an activation function (e.g., *ReLU*, *Sigmoid*). A detailed explanation of GCN model is presented in the original work (Kipf and Welling, 2016).

Since Fine-grained Typing is a multi-label classification problem. The final layer further connects to a set of K Sigmoid functions, which correspond to the K types in the type hierarchy. Given the set of labeled entities $\mathcal{E}^l$, our model optimizes

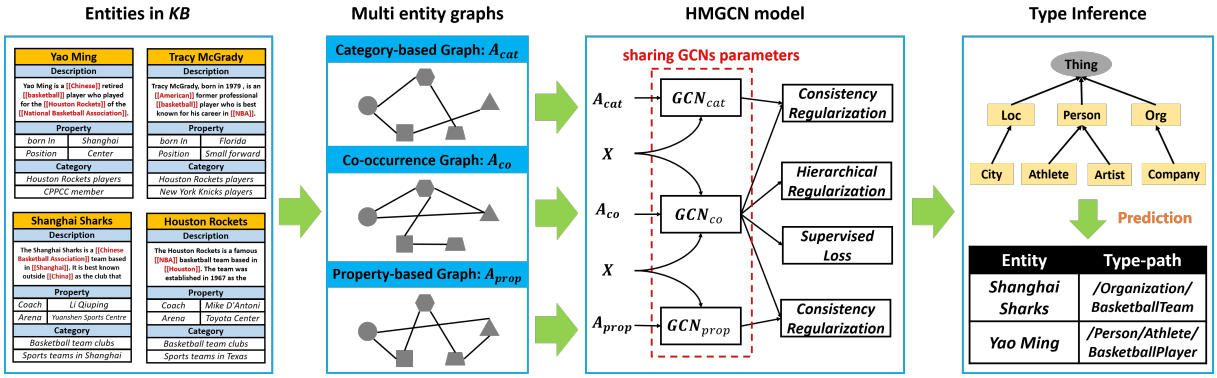


Figure 2: Framework of **HMGCN**. We convert entities in $\mathcal{KB}$ into multi kinds of semantic graphs. The co-occurrence graph $\mathbf{A}_{co}$ is constructed from anchor text in textual description. The category-based graph $\mathbf{A}_{cat}$ is derived from category proximity. The property-based graph $\mathbf{A}_{prop}$ is derived from property proximity. Each graph is fed to corresponding GCN model with same feature matrix $\mathbf{X}$. The three GCN models are learned by sharing parameters and consistency regularization. A hierarchical regularization is used to deal with the subClassOf relations between types.

the cross-entropy between the true type distribution and the predicted distribution.

$$L_s = -\sum_{i=1}^{N_l}\sum_{j=1}^{K} \mathbf{Y}_{i,j} \ln \hat{\mathbf{Z}}_{i,j} + (1-\mathbf{Y}_{i,j}) \ln (1-\hat{\mathbf{Z}}_{i,j}) \tag{2}$$

Where $\mathbf{Y}_{i,j}$ indicates whether entity $e_i$ belongs to type $t_j$. $\hat{\mathbf{Z}}_{i,j}$ refers to the probability of GCN prediction.

The role of $\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{Z}^{(i-1)}\mathbf{W}^{(i)}$ is to exactly conduct a 1-hop diffusion process in each layer. Namely, a node's feature vector is enriched in the way of linearly adding all the feature vectors of its neighbors. With more hidden layers, an entity's feature can diffuse to further entities (2-hop, 3-hop and so on). Research shows that there's no significant increase when the number of layers is higher than 3 (Kipf and Welling, 2016). Here, we assume adjacency matrix $\mathbf{A}$ and feature matrix $\mathbf{X}$ are given. In the next part of this section, we show how to compute $\mathbf{A}$ and $\mathbf{X}$.

### 3.2 Connectivity Matrix Designing

In Knowledge Graphs, there are large numbers of descriptive anchor texts that are helpful for type inference. For an entity $e$, $CXT(e)$ is an entity set which contains all entities occur in its textual description. For example in Figure 1, *Houston Rockets* is in $CXT(YaoMing)$. If entity $e_i \in CXT(e_j)$, we can say that $e_i$ occurs in $e_j$'context or $e_j$ cites $e_i$ in its textual description. These are co-occurrence information among entities, and can be explicitly extracted and represented by the connectivity matrix $\mathbf{A}_{co}$.

$$A_{co}[i,j] = \begin{cases} 1 & e_i \in CXT(e_j) \text{ or } e_j \in CXT(e_i) \\ 0 & otherwise \end{cases} \tag{3}$$

$\mathbf{A}_{co}$ can capture co-occurrence relations, though not precise enough. For example, *Houston Rockets* and *NBA* occur in *Yao Ming*'s context, but they do not belong to same type.

Besides $\mathbf{A}_{co}$, we calculate the correlation between entities via category information. If entity $e_i$ and $e_j$ share common categories, $e_i$ and $e_j$ will be likely belong to a same type. For example in Figure 2, both *Houston Rockets* and *Shanghai Sharks* have category *Basketball team clubs*, they have same type *BasketballTeam*. We construct $\mathbf{A}_{cat}$, and use Jaccard similarity coefficient to calculate each element $\mathbf{A}_{cat}[i,j]$:

$$\mathbf{A}_{cat}[i,j] = \frac{|Cat(e_i) \cap Cat(e_j)|}{|Cat(e_i) \cup Cat(e_j)|} \tag{4}$$

in which, $Cat(e)$ is the category set of entity $e$. In the same way, we can calculate the correlation between entities via property information, and construct $\mathbf{A}_{prop}$, whose element is calculated as follows.

$$\mathbf{A}_{prop}[i,j] = \frac{|Prop(e_i) \cap Prop(e_j)|}{|Prop(e_i) \cup Prop(e_j)|} \tag{5}$$

in which $Prop(e)$ is the property set of entity $e$.

Each connectivity matrix is fed to its corresponding GCN model, i.e., $\mathbf{GCN}_{co}$, $\mathbf{GCN}_{cat}$ and

$\mathbf{GCN}_{prop}$, respectively. As mentioned in Section 2, each entity is associated with text description, category and property. We used category and property information to construct connectivity matrix, therefore, words appeared in the text description of an entity can be approximately viewed as features of this entity. We apply fastText method, which treats the average of word/n-grams embeddings as entity feature (Joulin et al., 2016), and construct the feature matrix $\mathbf{X}$. As shown in Figure 2, both $\mathbf{GCN}_{co}$, $\mathbf{GCN}_{cat}$ and $\mathbf{GCN}_{prop}$ take $\mathbf{X}$ as input feature matrix.

### 3.3 Parameters Sharing and Consistency Regularization

The success of our model largely relies on the strategy that three GCN models share common parameters (i.e., neural network weights $\mathbf{W}^{(i)}$ in Eqs. 1), as shown in Figure 2. By doing so, our model (which is characterized by the parameters $\mathbf{W}^{(i)}$) can simultaneously consider the knowledge encoded in $\mathbf{A}_{co}$, $\mathbf{A}_{cat}$ and $\mathbf{A}_{prop}$. The prediction of $\mathbf{GCN}_{co}$ and $\mathbf{GCN}_{cat}$ are denoted as $\hat{\mathbf{Z}}^{co}$ and $\hat{\mathbf{Z}}^{cat} \in \mathbb{R}^{N\times K}$, respectively. To jointly consider the opinions from $\mathbf{GCN}_{co}$ and $\mathbf{GCN}_{cat}$, we apply an unsupervised regularizer for the ensemble. We minimize the mean squared difference between $\hat{\mathbf{Z}}^{co}$ and $\hat{\mathbf{Z}}^{cat}$ for all $N$ entities.

$$L_{c1} = \sum_{i=1}^{N} \left\| \hat{\mathbf{Z}}^{cat}_{i,:} - \hat{\mathbf{Z}}^{co}_{i,:} \right\|^2 \tag{6}$$

The prediction of $\mathbf{GCN}_{prop}$ is denoted as $\hat{\mathbf{Z}}^{prop}$. In the same way, we minimize the mean squared differences between $\hat{\mathbf{Z}}^{co}$ and $\hat{\mathbf{Z}}^{prop}$ over all $N$ entities.

$$L_{c2} = \sum_{i=1}^{N} \left\| \hat{\mathbf{Z}}^{prop}_{i,:} - \hat{\mathbf{Z}}^{co}_{i,:} \right\|^2 \tag{7}$$

Our model can jointly consider the opinions of $\mathbf{GCN}_{co}$, $\mathbf{GCN}_{cat}$ and $\mathbf{GCN}_{prop}$. Although sharing the same parameters $\mathbf{W}^{(i)}$, three GCN models utilize different connectivity matrices as input, which capture different semantic correlations between entities. This difference may cause the predictions of three GCN models to differ. Our model is expected to give a consistent prediction via the proposed unsupervised consistency regularizations, i.e., minimizing Eqs. 6 and 7. As a result, the learned parameter matrix $\mathbf{W}^{(i)}$ considers the judgments from both $\mathbf{GCN}_{co}$, $\mathbf{GCN}_{cat}$ and $\mathbf{GCN}_{prop}$.

### 3.4 Recursive Hierarchical Regularization

Leaf nodes (in type hierarchy) may have insufficient training examples. In that case, decisions can be regularized by its parent, if a type hierarchy is available. We introduce dependency relations among types to improve the classification performance. Similar to (Peng et al., 2018; Gopal and Yang, 2013), we use a recursive regularization over the final layer. Two types shall have similar embeddings, if they are close in a graph. In particular, types that are related by *subClassOf* relation shall have similar embeddings. For example, in Figure 1, there is an edge between *Athlete* and *BasketballPlayer*, so the parameters of the two types could be similar to each other. Assuming there are $n$ hidden layers, the last layer parameter matrix $\mathbf{W}^{(n)}$ can be regarded as the type embedding matrix, each row in $\mathbf{W}^{(n)}$ refers to a type representation. Let $C(t)$ refer to the sub-type set of $t$ ($t$'s children in the hierarchy). We use the following recursive regularization term to regularize the parameter of each type:

$$L_{hie} = \sum_{i=1}^{K} \sum_{j\in C(t_i)} \frac{1}{2} \left\| \mathbf{W}^{(n)}_{i,:} - \mathbf{W}^{(n)}_{j,:} \right\|^2 \tag{8}$$

### 3.5 Model Training

We calculate the supervised loss over all labeled entities for $\mathbf{GCN}_{co}$ (Eqs. 2). Although $\mathbf{Z}^{co}_i$ is regarded as the final type prediction result, our model can consider the opinions from three GCN models via parameters sharing and consistency regularization. The total loss is the sum of supervised loss, two consistency regularizations and recursive regularization.

$$Loss = L_s + \lambda(t)(L_{c1} + L_{c2}) + \lambda' L_{hie} \tag{9}$$

in which $\lambda(t)$ is a dynamic weight function, and $\lambda'$ is chosen among a fixed set based on performance on the dev set. At the beginning of the training process, $\lambda(t)$ is small, the loss function is mainly dominated by $L_s$. **HMGCN** is inclined to agree with $\mathbf{GCN}_{co}$ when making decisions, which encodes co-occurrence relation between entities. $\lambda(t)$ increases as time goes on, our model will obtain posterior distribution over the types using $\mathbf{GCN}_{co}$, and $\lambda(t)$ will force our model to consider simultaneously the knowledge encoded in $\mathbf{GCN}_{cat}$ and $\mathbf{GCN}_{prop}$. $\mathbf{GCN}_{cat}$ and $\mathbf{GCN}_{prop}$ play more important role when making decision. We adopt Adam (Kingma and Ba, 2014) to minimize the

above loss functions. For each $e_i \in \mathcal{E}^u$, we use $\mathbf{Z}_i^{co}$ as the type prediction result.

## 4 Experiments

### 4.1 Datasets and Metrics

**Datasets**: There are two public large-scale datasets available for Fine-Grained Entity Typing in $\mathcal{KB}$, *FIGER*[1] and *DBpedia* (Zhang et al., 2015). *FIGER* is a widely used dataset which is proposed by Yaghoobzadeh et al.. The *DBpedia* dataset is constructed by picking 14 non-overlapping types for text classification originally, we expand these 14 types and form a hierarchy. For each dataset, we extract entity feature from *DBpedia*[2]. For *FIGER* and *DBpedia*, we split entities into train (50%), dev (20%) and test (30%) sets. Since *DBpedia* dataset contains 630,000 entities, which is too big for computation, we divide it equally into three parts (i.e., DB-1, -2 and -3). Table 1 shows some statistics about the two datasets. Our source code is available[3] for reference. More detailed information can be found there.

Table 1: Statistics of the datasets.

| Dataset | **FIGER** | **DB-1** | **DB-2** | **DB-3** |
|---|---|---|---|---|
| #Entities | 200,000 | 210,000 | 210,000 | 210,000 |
| #Types | 102 | 48 | 48 | 48 |

**Metrics**: To evaluate the performance of our proposed method, we use Accuracy (**Strict-F1**), Micro-averaged F1 (**Mi-F1**) and Macro-averaged F1 (**Ma-F1**), which have been used in many fine-grained typing systems (Ling and Weld, 2012; Ren et al., 2016; Yaghoobzadeh et al., 2017).

### 4.2 Methods for Comparison

**Baseline**: we compare **HMGCN** with four state-of-the-art of methods and three variants of **HMGCN**:

- **CUTE:** (Xu et al., 2016) utilizes three kinds of entity features: category, property and property-value pair, and employs a hierarchical multi-label classification method.
- **MuLR:** (Yaghoobzadeh and Schütze, 2017) applies embeddings of words, entities and types in entity typing task, and uses multi-level representations of entities via character, word, and entity embedding technologies.
- **FIGMENT:** (Yaghoobzadeh et al., 2017) introduces global model and context model to provide complementary information for entity typing.
- **APE:** (Jin et al., 2018) applies an attributed and predictive entity embedding method to learn entity representations. In a way, it takes graph structure and entity features into account.
- **HMGCN's variants:** $\textbf{HMGCN}_{no\_cat}$ only use $\textbf{GCN}_{co}$ and $\textbf{GCN}_{prop}$ to make type inference. $\textbf{HMGCN}_{no\_prop}$ only use $\textbf{GCN}_{co}$ and $\textbf{GCN}_{cat}$ to make type inference. $\textbf{HMGCN}_{no\_co}$ only use $\textbf{GCN}_{cat}$ and $\textbf{GCN}_{prop}$ to make type inference. $\textbf{HMGCN}_{flat}$ ignores the correlation between types, i.e., removes hierarchical recursive regularization.

**Parameter Settings**: **HMGCN** is implemented based on the original GCN model (Kipf and Welling, 2016). In our implementation, both $\textbf{GCN}_{co}$, $\textbf{GCN}_{cat}$ and $\textbf{GCN}_{prop}$ have two hidden layers. Namely, there are two separate parameter vectors, $\mathbf{W}^{(1)}$ and $\mathbf{W}^{(2)}$, that need training. Table 2 presents detailed information about the implementation of our method for each dataset, including (1) number of hidden units; (2) dropout rate; and (3) learning rate $\eta$. $\lambda'$ is chosen among $\{0.00625, 0.0125, 0.025, 0.05, 0.1, 0.2, 0.4, 0.8\}$, we found $\lambda' = 0.2$ achieves best performance. We set $\lambda(t)$ in several ways, which is discussed in Section 4.4. We train all models for a maximum of 200 epochs (training iterations) using Adam (Kingma and Ba, 2014) and early stopping with a window size of 10 to minimize loss functions.

Table 2: Values of the Hyper-Parameters in **HMGCN** model.

| Dataset | $\mathbf{W}^{(1)}$ | $\mathbf{W}^{(2)}$ | Dropout rate | Learning rate |
|---|---|---|---|---|
| **DBpedia** | 32 | 48 | 30% | 0.001 |
| **FIGER** | 64 | 102 | 10% | 0.002 |

[1] https://github.com/yyaghoobzadeh/figment
[2] http://wiki.dbpedia.org/downloads-2016-10
[3] https://github.com/SIGKDD/HMGCN

### 4.3 Overall Comparison Results

Table 3 and 4 shows the overall performance on two datasets and we have the following conclusions:

**Comparison with Entity Typing methods.** Our model significantly outperforms the state-of-the-art entity typing methods over all datasets. **HMGCN** outperforms the best baseline with 1.7% and 1.3% improvement in Mi-F1 and Ma-F1 on *FIGER* dataset, 3.2% and 3.6% improvement in Mi-F1 and Ma-F1 on *DB-1* dataset, 3.9% and 3.5% improvement in Mi-F1 and Ma-F1 on *DB-2* dataset, 3.7% and 3.8% improvement in Mi-F1 and Ma-F1 on *DB-3* dataset, respectively. Compared with other methods, the performance of **CUTE** is relatively poor, as it ignores the rich structure information both in entity graph and type hierarchy. **FIGMENT** and **MuLR** achieve better performance, in part because powerful embedding technologies are applied and the decision is made in global-view and context-view. **APE** considers graph structure and entity features in attributed network embedding manner, while **HMGCN** applies this in a convolutional manner, which can embed the graph knowledge more sufficiently (Kipf and Welling, 2016; Zhuang and Ma, 2018).

**Comparison with Variants.** **HMGCN** consistently outperforms all four variants, because in **HMGCN** three GCN models provide complementary information. The key of **HMGCN** is parameter-sharing ($\mathbf{W}^{(i)}$ in $\mathbf{GCN}_{co}$, $\mathbf{GCN}_{cat}$ and $\mathbf{GCN}_{prop}$) and unsupervised regularizer for ensemble. The three GCNs encode different kinds of semantic correlations between entities, so that **HMGCN** could have a trade-off in decision-making. As a result, the trained parameters $\mathbf{W}^{(i)}$ have considered opinions from all three GCNs. $\mathbf{HMGCN}_{no\_co}$ and $\mathbf{HMGCN}_{no\_prop}$ demonstrate better performance than $\mathbf{HMGCN}_{no\_cat}$, which suggests that category information should be more import than co-occurrence and property information. $\mathbf{HMGCN}_{flat}$ ignores the correlations between types (i.e., *subClassOf* relations), while **HMGCN** take the structure of type hierarchy as prior knowledge. Infrequent types benefit from the recursive regularization in **HMGCN**. Because when an infrequent type has less training example, the decision can be regularized by its parent (super-type).

### 4.4 Result Analysis

**Effects of Labeled Data.** Intuitively, our model can learn better embeddings and achieve better performance, if we have more labeled entities. We investigate the effect of labeled data proportion. The results show that three metrics strikingly increase with more labeled data added, and gradually become stable when the proportion is over 0.5, as illustrated in Figure 3(a). This shows that our model can achieve a satisfactory result, even with not enough labeled data, and this advantage is benefited from the information diffusion of the GCN model, i.e., similar entities should share information between each other.

**Results of Frequent/Infrequent Types.** We evaluate the performance on *frequent types* ($frequency > 3,000$; 15 types) and *infrequent types* ($frequency < 200$; 36 types). The classification measure is the type macro average F1 (F1 of entities assigned to a type, then averaged over types) (Yosef et al., 2012). Note that it is different from Ma-F1 reported in Table 3. Generally, the performance on infrequent types is worse than frequent ones. Our model consistently outperforms the other methods on infrequent types, which demonstrates its ability on dealing with rare types. The results for infrequent and frequent types in *FIGER* dataset are illustrated in Figure 3(b).

**Effect of Regularization Weight $\lambda(\mathbf{t})$.** Our model uses a weight function $\lambda(t)$ to balance the trade-off between the supervised loss and unsupervised consistency regularizer. We devised several different weight functions (Zhuang and Ma, 2018), as shown in Figure 3(c). Here, $t$ is defined as the number of epochs. $f_i$ $(1 \leq i \leq 4)$ increases with different increment speed, as the number of epochs increases; while $f_5$ decreases as the number of epochs increases. $f_i$ $(1 \leq i \leq 4)$ strikingly outperforms $f_5$, as shown in Figure 3(d). We can see that the knowledge embedded in $\mathbf{GCN}_{cat}$ and $\mathbf{GCN}_{prop}$ is beneficial to entity typing task. At the beginning of training process, supervised loss function plays the leading role. After several epochs, $\lambda(t)$ forces our model to simultaneously consider the knowledge encoded in $\mathbf{GCN}_{cat}$ and $\mathbf{GCN}_{prop}$.

## 5 Related Work

Fine-grained entity typing in $\mathcal{KB}$ is an important sub-task of knowledge base completion. Most ex-

Table 3: Typing performance on DBpedia.

| Methods | DB-1 | | | DB-2 | | | DB-3 | | |
|---|---|---|---|---|---|---|---|---|---|
| | Strict | Ma-F1 | Mi-F1 | Strict | Ma-F1 | Mi-F1 | Strict | Ma-F1 | Mi-F1 |
| **CUTE** | 0.518 | 0.679 | 0.702 | 0.520 | 0.681 | 0.713 | 0.524 | 0.685 | 0.717 |
| **MuLR** | 0.608 | 0.748 | 0.771 | 0.612 | 0.757 | 0.784 | 0.610 | 0.752 | 0.775 |
| **FIGMENT** | 0.601 | 0.740 | 0.766 | 0.597 | 0.738 | 0.765 | 0.605 | 0.745 | 0.769 |
| **APE** | 0.612 | 0.758 | 0.784 | 0.614 | 0.761 | 0.785 | 0.611 | 0.760 | 0.782 |
| **HMGCN**$_{no\_cat}$ | 0.619 | 0.769 | 0.792 | 0.622 | 0.772 | 0.796 | 0.620 | 0.770 | 0.793 |
| **HMGCN**$_{no\_prop}$ | 0.621 | 0.771 | 0.793 | 0.625 | 0.782 | 0.801 | 0.623 | 0.775 | 0.799 |
| **HMGCN**$_{no\_co}$ | 0.627 | 0.789 | 0.808 | 0.629 | 0.790 | 0.814 | 0.626 | 0.786 | 0.812 |
| **HMGCN**$_{flat}$ | 0.626 | 0.785 | 0.812 | 0.633 | 0.794 | 0.820 | 0.628 | 0.791 | 0.817 |
| **HMGCN** | **0.635** | **0.794** | **0.816** | **0.637** | **0.796** | **0.824** | **0.631** | **0.798** | **0.819** |

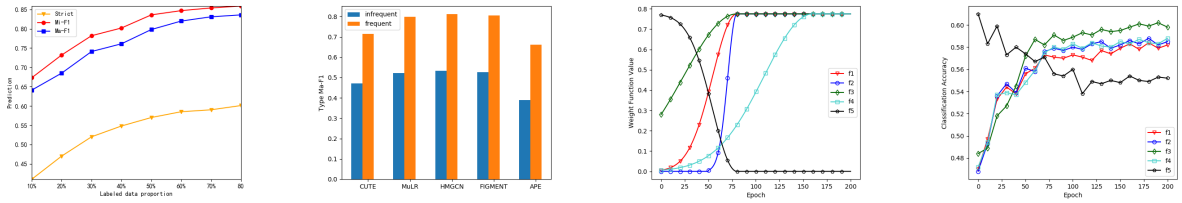

(a) Label proportion (b) Frequent/infrequent types (c) $\lambda(t)$ implementations (d) $\lambda(t)$ performances

Figure 3: (a). Performance on different labeled data proportion. (b). Performance on frequent and infrequent types in FIGER dataset (c). Five different implementations of the weight function. X-axis represents the function variable t and the Y-axis represents the function value. (d). Classification accuracy using different $\lambda(t)$ functions for training on FIGER dataset.

Table 4: Typing performance on FIGER.

| Methods | Strict | Ma-F1 | Mi-F1 |
|---|---|---|---|
| **CUTE** | 0.531 | 0.743 | 0.782 |
| **MuLR** | 0.548 | 0.776 | 0.812 |
| **FIGMENT** | 0.563 | 0.785 | 0.819 |
| **APE** | 0.515 | 0.722 | 0.756 |
| **HMGCN**$_{no\_cat}$ | 0.546 | 0.773 | 0.805 |
| **HMGCN**$_{no\_prop}$ | 0.552 | 0.781 | 0.814 |
| **HMGCN**$_{no\_co}$ | 0.557 | 0.784 | 0.821 |
| **HMGCN**$_{flat}$ | 0.564 | 0.789 | 0.827 |
| **HMGCN** | **0.570** | **0.798** | **0.836** |

isting methods do type inference by utilizing entity's inherent feature (textual description, property and category) or mention in text (anchor text). (Neelakantan and Chang, 2015) use text description as entity feature representation, then design a global objective function to predict missing entity types in a $\mathcal{KB}$. (Xu et al., 2016) use property and category information, along with a multi-label hierarchical classifier, to assign DBpedia types to Chinese entities. (Yaghoobzadeh and Schütze, 2015)first propose FIGMENT to address this problem. They only used contextual information to assign types for entities in $\mathcal{KB}$. After that, they present FIGMENT-Multi to learn multi-level representations of entities on three complementary levels (character, word and entity) (Yaghoobzadeh and Schütze, 2017). FIGMENT-Multi predicts whether an entity is a member of a type based on the learned embeddings. Finally, they propose an embedding based method which combines a global model with a context model. A global model that scores based on aggregated context information and a context model that aggregates the scores of individual contexts (Yaghoobzadeh et al., 2017). Recent research convert entities in $\mathcal{KB}$ to an entity graph and apply graph-based algorithm on it. (Jin et al., 2018) use links between entities to construct an entity graph, jointly utilize entity feature and graph structure to make type inference. They apply an attributed and predictive network embedding method to encode entity feature and graph structure. After that, a multi-label classifier is employed to carry out type classification.

## 6 Conclusion

We convert the task of Fine-Grained Entity Typing in $\mathcal{KB}$ to graph-based semi-supervised classification task, and propose a hierarchical multi graph

convolutional network (**HMGCN**) that fully utilizes entity features, entity graph structure, and type hierarchy structure to address this task. Three GCN models are devised to embed multi kinds of semantic correlations between entities. A recursive regularization is proposed to make the model understand, to a certain extent, the structure of type hierarchy. Experiments on two real datasets demonstrate the effectiveness of the proposed model.

## 7 Acknowledgements

This work is supported by National Key Research and Development Program of China (2017YFB1002101), NSFC key projects (U1736204, 6153301), a research fund by Alibaba, and THUNUS NExT Co-Lab.

## References

James Atwood and Don Towsley. 2016. Diffusion-convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1993–2001.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852.

Siddharth Gopal and Yiming Yang. 2013. Recursive regularization for large-scale classification with hierarchical and graphical dependencies. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 257–265. ACM.

Xiao Huang, Jundong Li, and Xia Hu. 2017. Label informed attributed network embedding. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 731–739. ACM.

Hailong Jin, Lei Hou, Juanzi Li, and Tiansi Dong. 2018. Attributed and predictive entity embedding for fine-grained entity typing in knowledge bases. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 282–292.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Lizi Liao, Xiangnan He, Hanwang Zhang, and Tat Seng Chua. 2017. Attributed social network embedding. *arXiv:1705.04969*.

Xiao Ling and Daniel S Weld. 2012. Fine-grained entity recognition. In *AAAI*, pages 94–100.

Gregory Murphy. 2004. *The big book of concepts*. MIT press.

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.

Arvind Neelakantan and Ming Wei Chang. 2015. Inferring missing entity type instances for knowledge base completion: New dataset and methods. *Computer Science*, pages 35–40.

Hao Peng, Jianxin Li, Yu He, Yaopeng Liu, Mengjiao Bao, Lihong Wang, Yangqiu Song, and Qiang Yang. 2018. Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 1063–1072. International World Wide Web Conferences Steering Committee.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155. Association for Computational Linguistics.

Xiang Ren, Wenqi He, Meng Qu, Lifu Huang, Heng Ji, and Jiawei Han. 2016. Afet: Automatic fine-grained entity typing by hierarchical partial-label embedding. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1369–1378.

Zhichun Wang, Qingsong Lv, Xiaohan Lan, and Yu Zhang. 2018. Cross-lingual knowledge graph alignment via graph convolutional networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 349–357.

Bo Xu, Yi Zhang, Jiaqing Liang, Yanghua Xiao, Seung-won Hwang, and Wei Wang. 2016. Cross-lingual type inference. In *International Conference on Database Systems for Advanced Applications*, pages 447–462.

Yadollah Yaghoobzadeh, Heike Adel, and Hinrich Schütze. 2017. Corpus-level fine-grained entity typing. *arXiv:1708.02275*.

Yadollah Yaghoobzadeh and Hinrich Schütze. 2015. Corpus-level fine-grained entity typing using contextual information. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 715–725.

Yadollah Yaghoobzadeh and Hinrich Schütze. 2017. Multi-level representations for fine-grained typing of knowledge base entities. *arXiv:1701.02025*.

Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *ICML*, pages 40–48.

M Amir Yosef, Sandro Bauer, Johannes Hoffart, Marc Spaniol, and Gerhard Weikum. 2012. Hyena: Hierarchical type classification for entity names. In *Proceedings of the 24th International Conference on Computational Linguistics*, pages 1361–1370.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Chenyi Zhuang and Qiang Ma. 2018. Dual graph convolutional networks for graph-based semi-supervised classification. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 499–508. International World Wide Web Conferences Steering Committee.

# Learning to Infer Entities, Properties and their Relations from Clinical Conversations

Nan Du, Mingqiu Wang, Linh Tran, Gang Li, and Izhak Shafran

{dunan, mingqiuwang, tranlm, leebird, izhak}@google.com

Google Inc.

## Abstract

Recently we proposed the Span Attribute Tagging (SAT) Model (Du et al., 2019) to infer clinical entities (e.g., symptoms) and their properties (e.g., duration). It tackles the challenge of large label space and limited training data using a hierarchical two-stage approach that identifies the span of interest in a tagging step and assigns labels to the span in a classification step.

We extend the SAT model to jointly infer not only entities and their properties but also relations between them. Most relation extraction models restrict inferring relations between tokens within a few neighboring sentences, mainly to avoid high computational complexity. In contrast, our proposed Relation-SAT (R-SAT) model is computationally efficient and can infer relations over the entire conversation, spanning an average duration of 10 minutes.

We evaluate our model on a corpus of clinical conversations. When the entities are given, the R-SAT outperforms baselines in identifying relations between symptoms and their properties by about 32% (0.82 vs 0.62 F-score) and by about 50% (0.60 vs 0.41 F-score) on medications and their properties. On the more difficult task of jointly inferring entities and relations, the R-SAT model achieves a performance of 0.34 and 0.45 for symptoms and medications respectively, which is significantly better than 0.18 and 0.35 for the baseline model. The contributions of different components of the model are quantified using ablation analysis.

## 1 Introduction

The widespread adoption of Electronic Health Records by clinics across United States has placed a disproportionately heavy burden on clinical providers, causing burnouts among them (Wachter and Goldsmith, 2018; Xu, 2018; Arndt et al., 2017). There has been considerable interest, both in academia and industry, to automate aspects of documentation so that providers can spend more time with their patients. One such approach aims to generate clinical notes directly from the doctor-patient conversations (Patel et al., 2018; Finley et al., 2018a,b). The success of such an approach hinges on extracting relevant information reliably and accurately from clinical conversations.

In this paper, we investigate the tasks of jointly inferring entities, specifically, symptoms (Sx), medications (Rx), their properties and relations between them from clinical conversations. These tasks are defined in Section 2. The key contributions of the work reported here include: (i) a novel model architecture for jointly inferring entities and their relations, whose parameters are learned using the multi-task learning paradigm (Section 4), (ii) comprehensive empirical evaluation of our model on a corpus of clinical conversations (Section 6), and (iii) understanding the model performance using ablation study and human error analysis (Section 6.7). Since clinical conversations include domain specific knowledge, we also investigate the benefit of augmenting the input feature representation with knowledge graph embedding. Finally, we summarize our conclusions and contributions in Section 7.

## 2 Task Definitions

For the purpose of defining the tasks, consider the snippet of a clinical conversation in Table 1.

### 2.1 The Symptom Task (Sx)

This task consists of extracting the tuples (*symType, propType, propContent*).

The "pain" in the example in Table 1 is annotated as *symType* (sym/msk/pain), where msk stands for musculo-skeletal system. We have



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4979–4990, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

**DR:** How often do you have pain in your arms?
**PT:** It hurts every morning.
**DR:** Are you taking anything for it?
**PT:** I've been taking Ibuprofen. Twice a day.

Table 1: An example to illustrate entities, properties and their relations. Entities – (sym/msk/pain: *pain*) & (meds/name: *Ibuprofen*); Properties – (symprop/freq: *every morning*) & (medsprop/freq: *twice a day*); Relations: (sym/msk/pain, symprop/freq, *every morning*), (Ibuprofen, medsprop/freq, *twice a day*).

pre-defined 186 categories for symptom types, curated by a team of practising physicians and scribes, based on how they appear in clinical notes. We deliberately abstained from the more exhaustive symptom labels such as UMLS and ICD codes (Bodenreider, 2004) in favor of this smaller set since our training data is limited.

The properties associated with the symptoms, *propType*, fall into four categories: symprop/severity, symprop/duration, symprop/location, and symprop/frequency. The *propContent* denotes the content associated with the property. In the running example, "every morning" is the content associated with the property type *symprop/frequency*.

Not all the symptoms mentioned in the course of clinical conversations are experienced by the patients. We explicitly infer the status of a symptom as experienced or not. This secondary task extracts the pair: (*symType, symStatus*).

### 2.2 The Medication Task (Rx)

This task consists of extracting tuples of the form: (*medContent, propType, propContent*).

While symptoms can be categorized into a closed set, the set of medications is very large and continually updated. Moreover, in conversations, we would like to extract indirect references such as "pain medications" as *medContent*. We define three types of properties: medsprop/dosage, medsprop/duration and medsprop/frequency. In the running example,"twice a day" is the *propContent* of the type *medsprop/frequency* associated with the *medContent* "ibuprofen".

## 3 Previous Work

Relation extraction is a long studied problem in the NLP domain and include tasks such as the ACE (Doddington et al., 2004), the SemEval (Hendrickx et al., 2010), the i2b2/VA Task (Uzuner et al., 2011a), and the BioNLP Shared Task (Kim et al., 2013). Many early algorithms such as DIPRE algorithm by Brin (1998) and SNOWBALL algorithm by Agichtein and Gravano (2000) relied on regular expressions and rules (Fundel et al., 2007; Peng et al., 2014). Subsequent work exploited syntactic dependencies of the input sentences. Features from the dependency parse tree were used in maximum entropy models (Kambhatla, 2004) and neural network models (Snow et al., 2005). Kernels were defined over tree structures (Zelenko et al., 2003; Culotta and Sorensen, 2004; Qian et al., 2008). More efficient methods were investigated including shortest dependency path (Bunescu and Mooney, 2005) and sub-sequence kernels (Mooney and Bunescu, 2006). Recent work on deep learning models investigated convolutional neural networks (Liu et al., 2013), graph convolutional neural networks over pruned trees (Zhang et al., 2018), recursive matrix-vector projections (Socher et al., 2012) and Long Short Term Memory (LSTM) networks (Miwa and Bansal, 2016). Other more recent approaches include two-level reinforcement learning models (Takanobu et al., 2019), two layers of attention-based capsule network models (Zhang et al., 2019), and self-attention with transformers (Verga et al., 2018). In particular, (Miwa and Sasaki, 2014; Katiyar and Cardie, 2016; Zhang et al., 2017; Zheng et al., 2017; Verga et al., 2018; Takanobu et al., 2019) also seek to jointly learn the entities and relations among them together. A large fraction of the past work focused on relations within a single sentences. The dependency tree based approaches have been extended across sentences by linking the root notes of adjacent sentences (Gupta et al., 2019). Coreference resolution is a similar task which requires finding all mentions of the same entity in the text (Martschat and Strube, 2015; Clark and Manning, 2016; Lee et al., 2017).

In the medical domain, the BioNLP shared task deals with gene interactions and is very different from our domain (Kim et al., 2013). The *i2b2/va challenge* is closer to our domain of clinical notes, however, that task is defined on a small corpus of written discharge summaries (Uzuner et al., 2011b). Written domain benefits from cues such as the section headings which are unavailable in clinical conversations. For a wider survey of extracting clinical information from written clinical documents, see (Liu et al., 2012).

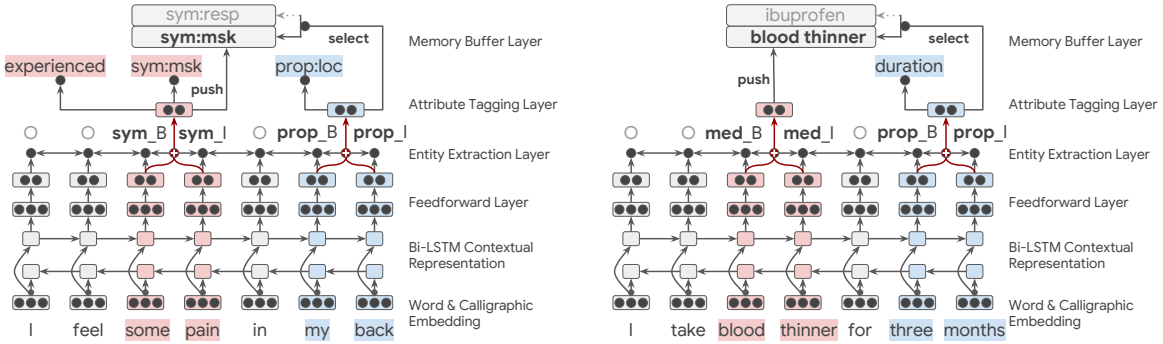


(a) Model architecture for the symptoms task.

(b) Model architecture for the medications task.

Figure 1: Variants of the R-SAT model architecture. The entity spans ("some pain", "blood thinner") are identified in a tagging step, which are pushed into a memory buffer along with their latent representation and in a subsequent step the property spans ("my back", "three months") selects the most related entity from the buffer.

## 4 Model

Our application requires performing multiple inferences simultaneously, that of identifying symptoms, medications, their properties and relations between them. For this purpose, we adopt the well-suited multitask learning framework and develop a model architecture, illustrated in Figure 1, that utilizes our limited annotated corpus efficiently.

### 4.1 Input Encoder Layer

Let $\mathbf{x}$ be an input sequence. We compute the contextual representation at the $k$-th step using a bidirectional LSTM, $\mathbf{h}'_\mathbf{k} = [\vec{\mathbf{h}}(\mathbf{x}_{\leq \mathbf{k}}|\vec{\Theta}_{LSTM}), \overleftarrow{\mathbf{h}}(\mathbf{x}_{\geq \mathbf{k}}|\overleftarrow{\Theta}_{LSTM})]$, which is fed into a two-layer fully connected feed-forward network. For simplicity, we drop the index $k$ from the rest. The final features are represented as $\mathbf{h}'' = MLP(\mathbf{h}'|\Theta_{FF})$. In our task, we found that the LSTM-based encoder performs better that the transformer-based encoder (Vaswani et al., 2017; Chen et al., 2018).

**Extending a Standard Tagging Model** In a typical tagging model, the contextual representation of the encoder $\mathbf{h}''$ is fed into a conditional random field (CRF) layer to predict the BIO-style tags (Collobert et al., 2011; Huang et al., 2015; Ma and Hovy, 2016; Chiu and Nichols, 2016; Lample et al., 2016; Peters et al., 2017; Yang et al., 2017; Changpinyo et al., 2018). Such a model can be extended to predict the relations. For example, in the utterance, "I feel some pain in my back", we could setup the tagger to predict the association between the symptom (*sym/msk*), and its property (*symprop/loc*) using a cross-product space where "my back" is tagged with *sym/msk+symprop/loc* so that the relation prediction problem is reformulated as a standard sequence labeling task. Although this would be a viable option for tasks where the tag set is small (e.g., place, organization, etc.), the cross-product space in our Sx task is unfortunately large (e.g., 186 Sx labels $\times$ 3 Sx property types, and 186 Sx labels $\times$ 3 Sx status types).

### 4.2 Span Extraction Layer

We propose an alternative formulation that tackles the problem in a hierarchical manner. We first identify the span of interest using ***generic*** tags with BIO notation, namely, (*sym_B*, *sym_I*) for symptoms and (*symprop_B*, *symprop_I*) for their properties, as in Figure 1(a). Likewise, (*med_B*, *med_I*) for medications and (*medsprop_B*, *medsprop_I*) for their properties as shown in Figure 1(b). This corresponds to highlighting, for example, "some pain" and "my back" as spans of interest.

Given the latent representations, $\mathbf{h} = (\boldsymbol{h}''_\mathbf{1}, \cdots, \boldsymbol{h}''_\mathbf{N})$, and the target tag sequence $\mathbf{y}^e = (y_1, \cdots, y_N)$ (e.g., *sym_B, sym_I, O, symprop_B, symprop_I*), we use the negative log-likelihood $-\log P(\mathbf{y}^e|\mathbf{h})$ under CRF as the loss of identifying spans of interest $-S(\mathbf{y}^e, \mathbf{h}) + \log \sum_{\mathbf{y}'} \exp(S(\mathbf{y}', \mathbf{h}))$, where $S(\mathbf{y}, \mathbf{h}) = \sum_{i=0}^{N} \mathbf{A}_{y_i, y_{i+1}} + \sum_{i=0}^{N} P(\mathbf{h}_i, \mathbf{y}_i)$ measures the compatibility between a sequence $\mathbf{y}$ and $\mathbf{h}$. The first component estimates the accumulated cost of transition between two neighboring tags using a learned transition matrix $\mathbf{A}$. $P(\mathbf{h}_i, \mathbf{y}_i)$ is computed via the inner product $\mathbf{h}_i^\top \mathbf{y}_i$ where

$y_i$ belongs to any sequence of tags $\mathbf{y}$ that can be decoded from $\mathbf{h}$. During training, the $\log P(\mathbf{y}^e|\mathbf{h})$ is estimated using forward-backward algorithm and during inference, the most probable sequence is computed using the Viterbi algorithm.

### 4.3 Attribute Tagging Layer

Using the latent representation of the highlighted span, we can predict one or more ***attributes*** of the span. In Figure 1(a), we can predict two attributes associated with "some pain": *sym/msk* as the symptom label and *symStatus/experienced* as its status. Similarly, in Figure 1(b), the span property span "three months" has the predicted property type *medsprop/duration*. Therefore, by forming semantic abstractions for each highlighted text span, we decompose a single complex tagging task in a large label space into correlated but simpler sub-tasks, which are likely to generalize better when the training data is limited.

Given the spans, either from the inferred or the ground truth sequence $\mathbf{y}^*$, a richer representation of the contexts can be used to predict attributes than otherwise possible. A contextual representation is computed from the starting $i$ and ending $j$ index of each span.

$$\boldsymbol{h}^s_{ij} = \text{Aggregate}(\boldsymbol{h}_k|\boldsymbol{h}_k \in \mathbf{h}, i \leq k < j) \quad (1)$$

where $\text{Aggregate}(\cdot)$ is the pooling function, implemented as mean, sum or attention-weighted sum of the latent states of the input encoder. The $k$th attributes associated with the span are modeled using $P(y^k_{attr}|\boldsymbol{h}^s_{ij})$. For example, while prediction symptom labels $s_x$ and their associated status $s_t$, the target attributes are $y^0_{attr} := y^{s_x}$ and $y^1_{attr} := y^{s_t}$. For predicting medication entities $r_x$ and their properties $p_r$, each span only has one attribute. Since each attribute comes from a pre-defined ontology, the multi-nomial distribution $P(y^k_{attr}|\boldsymbol{h}^s_{ij})$ can be modeled as $\text{Softmax}(\boldsymbol{h}^s_{ij}|\Theta^k)$ for each attribute.

### 4.4 Memory Buffer Layer

One of the critical components of our model is the memory buffer. Most previous models on joint inference of entities and relations consider all spans of entities and properties. This has the computational complexity of $\mathcal{O}(n^4)$ in the length of the input $n$, and makes it infeasible for application such as ours where the input could often be 1k words or more. We circumvent this problem using a memory buffer to cache all inferred candidate spans and test their relationship with inferred property spans. Note, unlike methods that cascade two such stages, our model is trained end-to-end jointly with multi-task learning.

The memory buffer saves different entries for symptom and medication tasks, as illustrated in Figure 1. At each occurrence of a symptom (medication) entity span, we push $\boldsymbol{m}^k = \text{Aggregate}(\{\boldsymbol{h}^{\boldsymbol{s}}_{\boldsymbol{ij}}, \boldsymbol{e}^{\boldsymbol{s}}\})$ into the $k$-th position of the memory buffer. For the symptom task, $\boldsymbol{e}^{\boldsymbol{s}}$ is the learned word embedding of one of the labels in the closed label set. In the medication case, $\boldsymbol{e}^{\boldsymbol{s}}$ is the $\text{Aggregate}$ of learned word embedding of the verbatim sub-sequence corresponding to the medication entity.

### 4.5 Relation Inference Layer

Each span of inferred property in the conversation is compared against each entry in the buffer. A property entity span is represented as $\boldsymbol{y}^p = \text{Aggregate}(\{\boldsymbol{h}^{\boldsymbol{p}}_{\boldsymbol{ij}}, \boldsymbol{e}^p\})$ where $\boldsymbol{e}^p$ is the $\text{Aggregate}$ of word embedding corresponding to the span. The multi-nomial likelihood is computed using a bilinear weight matrix $\boldsymbol{W}$. The most likely entry ($k$) is picked from the memory stack $\boldsymbol{M} = (\boldsymbol{m}^1, ..., \boldsymbol{m}^K)$ by maximizing the likelihood.

$$\begin{aligned} \hat{k} &= \arg\max_k P(k|\boldsymbol{y}^p) \\ &= \arg\max_k \text{Softmax}(\boldsymbol{y}^{p\top}\boldsymbol{W}\boldsymbol{M}) \end{aligned} \quad (2)$$

**Remarks** The computation cost of inferring relation between a property span and all the entities in the input is proportional to the memory buffer size. On our corpus, for Sx task, the mean and standard deviation per conversation was 22 and 15 respectively, and for Rx task, it was 32 and 23 respectively. Hence, the set of candidate entities considered is substantially smaller than all potential entities $O(n^2)$ in the input sequence.

The small size of the memory buffer also has an impact on rate of learning. In each training step, rather than updating all embedding, we only update a smaller number of embedding, those associated with the entries in the memory buffer. This makes the learning fast and efficient.

### 4.6 An End-to-end Learning Paradigm

We train the model end-to-end by minimizing the following loss function for each conversation:

$$\begin{aligned}\mathcal{L} = & - \alpha \log P(\mathbf{y}^e|\mathbf{h}) \\ & - \sum_{\{y^k_{attr} \in S\}} \log P(y^i_{attr}|\mathbf{h}) \\ & - \sum_{\{y^j_{pos} \in P\}} \log P(y^j_{pos}|\mathbf{h}) \end{aligned} \quad (3)$$

where $\mathbf{y}^e$ is the target sequence ($sym\_B$, $sym\_I$, $prop\_B$, $prop\_I$), $\{y^k_{attr}\}$ is the set of attribute labels for each highlighted span, $\{y^j_{pos}\}$ is the list of buffer slot indices and $\alpha$ is a relative weight.

During training, we are simultaneously attempting to detect the location of tags as well as classify the tags. Initially our model for locating the tags is unlikely to be reliable, and so we adopt a curriculum learning paradigm. Specifically, we provide the classification stage the reference location of the tag from the training data with probability $p$, and the inferred location of the tag with probability $1 - p$. We start the joint multi-task training by setting this probability to 1 and decrease it as training progresses (Bengio et al., 2015).

Since our model consists of span extraction and attribute tagging layers followed by relation extraction, we refer to our model as **Relational Span-Attribute Tagging Model** (R-SAT). One advantage of our model is that the computational complexity of joint inference is $\mathcal{O}(n)$ which is linear in the length of the conversation $n$. This is substantially cheaper than other previous work on joint relation prediction models where the computational complexity is $\mathcal{O}(N^4)$ (Lee et al., 2017).

## 5 Knowledge Graph Features

Medical domain knowledge could be helpful in increasing the likelihood of symptoms when related medications is mentioned in a conversation, and vice versa. One such source is a knowledge graph (KG) whose embedding represent a low-dimensional projection that captures structural and semantic information of its nodes. Previous work has demonstrated that KG embedding can improve relation extraction in written domain (Han et al., 2018). We utilize an internally developed KG that contains about 14k medical nodes of 87 different types (e.g., medications, symptoms, treatments, etc.). The nodes are represented by 256 dimension embedding vectors, which were trained

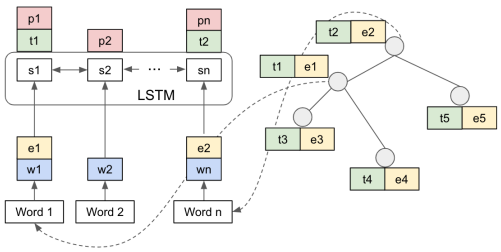


Figure 2: Illustration of how POS ($p$) features and knowledge graph (KG) are incorporated into our encoder. Dashed lines represent mappings from words ($w$) to KG nodes, which contain the embedding ($e$) and the type ($t$) information.

to minimize word2vec loss function on web documents (Mikolov et al., 2013). A given node may belong to multiple types and this is encoded as a sum of one-hot vectors. The input word sequences were mapped to KG nodes using an internal tool (Brown, 2013). For words that do not map to KG nodes, we use a learnable UNK vector of the same dimension as the KG embedding. In addition, we also represented linguistic information using part-of-speech (POS) tags as one-hot vector. The POS tags were inferred from the input sequence using an internal tool with 47 distinct tags (Andor et al., 2016). In our experiments, we find it most effective to concatenate word embedding with KG entities, and the encoder output with the embedding of POS tags and the KG entity types.

## 6 Experiments

We describe our corpus, evaluation metrics, the experimental setup, the evaluations of the proposed model and comparison with different baselines on both the symptom and medication tasks.

### 6.1 Corpus

Given the privacy-sensitive nature of clinical conversations, there aren't any publicly available corpora for this domain. Therefore, we utilize a private corpus consisting of 92K de-identified and manually transcribed audio recordings of clinical conversations, typically about 10 minutes long [IQR: 5-12 minutes] with mostly 2 participants (72.7%). Other participants when present included, for example, nurses and caregivers. The corresponding manual transcripts contained sequences that were on average 208 utterances or 1,459 words in length. We note that due to the casual conversational style of speech, an entity mentioned at the beginning can be related to a property mentioned at the end of the conversation. This

makes the problem of modeling relations much harder than previous work on extracting relations.

A subset of about 2,950 clinical conversations, related to primary care, were annotated by professional medical scribes. The ontology for labeling medication consisted of the type of medications (e.g., medications, supplements) and their properties (e.g., dosage, frequency, duration), and that for symptoms consisted of 186 symptom names and their properties. This resulted in 77K and 99K tags for the medication and symptom tasks, respectively. In all, there were 23k and 16k relationships between medications and symptoms and their properties, respectively. The conversations were divided into training (1,950), development (500) and test (500) sets.

In the case of medications, about 70% of the labels were about medications and the rest about their properties, of which 51% were dosage or quantity, and 40% were frequency. In the case of symptoms, 41% of the labels were about symptom names, another 41% about status, and the rest about properties, of which 39% were about frequency and 37% about body locations.

### 6.2 Pretraining

Since our labeled data is small, only about 3k, the input encoder of the model was pre-trained over the entire 92k conversations. For pre-training, given a short snippet of conversation, the model was tasked with predicting the next turn, similar to skip-thought (Kiros et al., 2015). Our models were trained using the Adam optimizer (Kingma and Ba, 2015) and the hyperparameters are described in the supplementary material.

### 6.3 Evaluation Metrics

As described in Section 2, our tasks consist of extracting tuples – (*symType, propType, propContent*) for symptoms task and (*medContent, propType, propContent*) for medications. The precision, recall and F1-scores are computed jointly over all the three elements and the content is treated as a list of tokens for evaluation purposes. To allow for partial content matches, we generalize the calculation of precision and recall such that

$$Precision = \frac{1}{|\mathcal{S}_{\hat{Y}}|} \sum_{i \in \mathcal{S}_{\hat{Y}}} \prod_{j=1}^{3} \mathcal{I}_{\hat{y}_i^j}(\hat{y}_i^j, y_i^j)$$

$$Recall = \frac{1}{|\mathcal{S}_{Y}|} \sum_{i \in \mathcal{S}_{Y}} \prod_{j=1}^{3} \mathcal{I}_{y_i^j}(\hat{y}_i^j, y_i^j)$$

where $\mathcal{S}_{\hat{Y}}$ denotes the set of predictions, $\mathcal{S}_Y$ denotes the set of ground truths, and $\mathcal{I}_{z_i^j}(\hat{x}_i^j, x_i^j) = |\hat{x}_i^j \cap x_i^j| / |z_i^j|$. We note that, as *symType* and *propType* are simply target classes, $\mathcal{I}$ reduces to a simple indicator function. Under the scenario that the content includes single elements, the entire calculation simplifies to the exact matching-based calculation of precision and recall over the set of predictions and ground truths. For the symptom task, we additionally evaluate the performance of predicting *symType* and *symStatus* by performing the exact matching-based calculation.

We illustrate this evaluation metric with an example below: There are two symptoms in the reference and the model extracted one of them. In the extracted symptom, the model correctly identified one out of the two content words. So, we score the precision as $1/1(1 * 1 * (1/1)) = 1$ and recall as $1/2((0 * 0 * 0) + (1 * 1 * (1/2))) = 0.25$.

Prediction: [(sym/sob, prop/severity, [bad])]
Reference: [(sym/unk, prop/location, [arm]),
(sym/sob, prop/severity, [really, bad])]

### 6.4 Baselines

**Symptom Task** As a baseline for this task, we train an extension of the standard tagging model, described in Section 4.1. The label space for extracting the relations between symptoms and their properties is *186 symptoms* $\times$ *3 properties*, and for extracting symptoms and their status is *186 symptoms* $\times$ *3 status*. Using the BIO-scheme, that adds up to 2,233 labels in total. The baseline consists of a bidirectional LSTM-encoder followed by two feed-forward layers $[512, 256]$ and then a 2,233 dimension softmax layer. The label space is too large to include a CRF layer. The encoder was pre-trained in the same way as described in Section 6.2, the hyperparameters were selected according to Table 2, and the model parameters were trained using cross-entropy loss.

**Medication Task** For this task, we adopt a different baseline since the generic medication entity type (e.g., drug name, supplement name) does not provide any useful information unlike the 186 symptom entity labels (e.g., sym/msk/pain). Instead, we adopt the neural co-reference resolution approach which is better suited to this task (Lee et al., 2017). The encoder is the same as the baseline for symptom task and pre-trained in the same manner. Since the BIO labels contain only 9 elements in this case, the encoder output is fed into a CRF layer. Each candidate relation is represented by concatenating the latent states of the head tokens of the medication entity and the property. This representation is augmented with an embedding of the token-distance, which is fed to a softmax layer whose binary output encodes whether they are related or not. Note our R-SAT model does not take the advantage of this distance embedding.

### 6.5 Parameter Tuning

Table 2 shows the parameters that were selected after evaluating over a range on a development set. In all experiments, the $\mathrm{Aggregate}(\cdot)$ function is implemented as the mean function for its simplicity.

| Parameter | Used | Range |
|---|---|---|
| Word emb | 256 | [128 – 512] |
| LSTM Cell | 1024 | [256 – 1024] |
| Enc/dec layers | 1 | [1 – 3] |
| Dropout | 0.4 | [0.0 – 0.5] |
| L2 | 1e-4 | [1e-5 – 1e-2] |
| Std of VN | 1e-3 | [1e-4 – 0.2] |
| $\alpha$ | 0.01 | [1e-4 – 0.1] |
| Learning rate | 1e-2 | [1e-4 – 1e-1] |

Table 2: Hyperparameters of our models for model reproducibility.

### 6.6 Results & Ablation Analysis

The performance of the proposed R-SAT model was compared with the baseline models, and the results are reported in Table 3.

**Symptom Task** The model was trained using multi-task learning for both tasks: (*symType, propType, propContent*) as well as (*symType, symStatus*). The performance was evaluated using all the elements of the tuple as described in Section 6.3. The baseline performs better on (*symType, symStatus*) compared to (*symType, propType, propContent*) possibly because there are more instances of the former in the training data than the latter. The R-SAT model performs significantly better than baselines on both tasks.

For understanding the contribution of different components of the model, we performed a series of ablation analysis by removing them one at a time. In extracting relations in *Sx + Property*, the KG embeddings along with POS tags contribute a relative gain of **13%** while the memory buffer brings a relative gain of **8%**. Neither of them impact *Sx + status*, and that is expected for memory buffer since the status is tagged on the same span as the contents of the memory buffer. Multi-task learning brings a relative improvement of **4%** on *Sx + Property*, and this may be because there are fewer instances of this relation in the training data, and jointly learning with *Sx + Status* helps to learn better representations. Note we have not checked other sequences for removing model components (e.g., removing Multi-tasking earlier or KG later).

**Medication Task** In the Rx case, we only have one task (*Rx + Property*), that is, predicting the relations between medications and their properties, e.g., ([ibuprofen], prop/dosage, [10 mg]). The baseline gives reasonable performance. Ablation analysis reveals that KG and POS features contribute about **4.6%** relative improvement, while the contextual span in memory buffer adds a substantial **43%** relative improvement. Since the medications are from an open set, we cannot run experiments without the buffer. Compared to symptoms task, the model performs better on medication task, and this may be due to lower variability in dosage.

**Relation Only Prediction** For teasing apart the strength and weakness of the model, we evaluated its performance when the entities and their properties were given, and the model was only required to decide whether a relation exists or not.

As a baseline, we compare our model with a most recently proposed model for document-level joint entity and relation extraction: BRAN, which achieved state-of-art performance for chemical-disease relation (Verga et al., 2018). When this model was originally used to test relations between all pairs of entities and properties in the entire conversation, it performed relatively poorly. Using the implementation released by the authors, the performance of BRAN was then optimized by restricting the distance between the pairs and by fine-tuning the threshold. The best results are re-

| Model | Sx + Property | Sx + Status | Rx + Property |
|---|---|---|---|
| Baseline | 0.18 | 0.44 | 0.35 |
| R-SAT | **0.34** | **0.57** | **0.45** |
| w/o [KG] | 0.30 | 0.56 | 0.43 |
| w/o [KG, Context] | 0.26 | 0.55 | 0.30 |
| w/o [KG, Context, Buffer] | 0.24 | 0.55 | n/a |
| w/o [KG, Context, Buffer, Multi-task] | 0.23 | n/a | n/a |
| Human | 0.51 | 0.78 | 0.52 |

Table 3: Comparison of the performance of the proposed R-SAT model with baselines and ablation analysis on different components (KG, Context, Buffer, Multi-task) where 'context' is the latent representation of the span $\boldsymbol{h}_{ij}^{s}$ in the memory buffer.

| Model | *Sx + Property* | *Rx + Property* |
|---|---|---|
| BRAN | 0.62 | 0.41 |
| R-SAT | 0.82 | 0.60 |

Table 4: Performance of the model when the entities and properties are given and it is only required to predict existence of relations.

ported in Table 4. Our proposed R-SAT model without any such constraints performs better than BRAN on both tasks by an absolute F1-score gain of about 0.20.

Interestingly, the performance of our model on *Sx + Property* jumps from 0.34 in the joint prediction task to 0.82 in the relation only prediction task. This reveals the primary weakness of the Sx model is in tagging the entities and the properties accurately. In contrast, the F1-score for *Rx + Property* is impacted less, and only moves up from 0.45 to 0.6.

The task of inferring whether a relation is present between a medication and its properties is more challenging than in the case of symptoms task. This is not entirely surprising since there is a higher correlation between symptom type and location (e.g., respiratory symptom being associated with nose) and relatively low correlation between dosage and medications (e.g., 400mg could be the dosage for several different medications).

### 6.7 Analysis

For understanding the inherent difficulty of extracting symptoms and medications and their properties from clinical conversations, we estimated human performance. A set of 500 conversations were annotated by 3 different scribes. We created a "voted" reference and compared the 3 annotations from each of the 3 scribes against them.

The F1-score of scribes were surprisingly low, with 0.51 for *Sx + Property* and 0.78 for *Sx + Status*. The model performance also finds extracting relation in *Sx + Property* to be more difficult than *Sx + Status* task. In summary, the model performance reaches 67% of human performance for *Sx + Property* and 73% for *Sx + Status*. The F1-score of scribes for *Rx + Property* is similar to that of *Sx + Property*. In this case, the model achieves about 85% of human performance. The human errors or inconsistencies in Sx and Rx annotations appear to be largely due to missed labels and not due to inconsistent spans for the same tags, or inconsistent tags for the same span.

While the majority of our relations in the reference annotations occurred within the same sentence, approximately 11.1% of relations occurred across 3 or more sentences. This typically occurred when the symptoms or medications are discussed over multiple dialog turns, as illustrated in Table 1. Among the relations correctly identified by the model, 10.6% were also across 3 or more sentences, which is very similar to the priors on the reference and seem to contain no bias. We notice that in certain cases, the model is able to link a property to an entity that is far away (100+ sentences) when a nearby mention of the same entity was missed by the model. Models that only examine relations in nearby sentences (2-3 sentences) would have missed the relation in such a scenario.

The majority of the errors result from our model missing the property span. Specifically, we see that 35% and 81% of the errors are due to model not detecting medications and symptoms property. For example, when i really have to, every three three months, which are rare mentions in informal language.

Our reference links each property to only one

entity. In certain cases, we notice that the model links the entity to an alternative mention or entity that is equally valid (Advil vs pain killer). So, our performance measure underestimates the actual model performance.

## 7 Conclusions

We propose a novel model to jointly infer entities and relations. The key components of the model are: a mechanism to highlight the spans of interest, classify them into entities, store the entities of interest in a memory buffer along with the latent representation of the context, and then infer relation between candidate property spans with the entities in the buffer. The components of the model are not tied to any domain. We have demonstrated applications in two different tasks. In the case of symptoms, the entities are categorized into 188 classes, while in the case of medications, the entities are an open set. The model is tailored for tasks where the training data is limited and the label space is large but can be partitioned into subsets. The two stage processing where the candidates are stored in a memory buffer allows us to perform the joint inference at a computational cost of $\mathcal{O}(n)$ in the length of the input $n$ compared to methods that explore all spans of entities and properties at a computational cost of $\mathcal{O}(n^4)$. The model is trained end-to-end. We evaluate the performance on three related tasks, namely, extracting symptoms and their status, relations between symptoms and their properties, and relations between medications and their properties. Our model outperforms the baselines substantially, by about 32-50%. Through ablation analysis, we observe that the memory buffer and the KG features contribute significantly to this performance gain. By comparing human scribes against "voted" reference, we see that the task is inherently difficult, and the models achieve about 67-85% of human performance.

## Acknowledgments

This work would not have been possible without the help of a number of colleagues, including Laurent El Shafey, Hagen Soltau, Yuhui Chen, Ashley Robson Domin, Lauren Keyes, Rayman Huang, Justin Stuart Paul, Mark Knichel, Jeff Carlson, Zoe Kendall, Mayank Mohta, Roberto Santana, Katherine Chou, Chris Co, Claire Cui, and Kyle Scholz.

## References

Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the ACM Conference on Digital Libraries*, pages 85–94. ACM.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Brian G. Arndt, John W. Beasley, Michelle D. Watkinson, Jonathan L. Temte, Wen-Jan Tuan, Christine A. Sinsky, and Valerie J. Gilchrist. 2017. Tethered to the EHR: primary care physician workload assessment using EHR event log data and time-motion observations. *Annals of Family Medicine*, 15(5):419–26.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 1171–1179, Cambridge, MA, USA. MIT Press.

Olivier Bodenreider. 2004. The unified medical language system (UMLS): integrating biomedical terminology. *Nucleic Acids Res.*, 32(Database issue):D267–70.

Sergey Brin. 1998. Extracting patterns and relations from the world wide web. In *In WebDB Workshop at 6th International Conference on Extending Database Technology, EDBT98*, pages 172–183.

Aaron Brown. 2013. Semantics and structured knowledge in practice at google. In *IEEE ICSC*. IEEE International conference on semantic computing.

Razvan C Bunescu and Raymond J Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 724–731, Stroudsburg, PA, USA. Association for Computational Linguistics.

Soravit Changpinyo, Hexiang Hu, and Fei Sha. 2018. Multi-task learning for sequence tagging: An empirical study. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2965–2977. Association for Computational Linguistics.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Niki Parmar, Mike Schuster, Zhifeng Chen, Yonghui Wu, and Macduff Hughes. 2018. The best of both worlds: Combining recent advances in neural machine translation. In *ACL*.

Jason Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Kevin Clark and Christopher D Manning. 2016. Deep reinforcement learning for Mention-Ranking coreference models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2256–2262, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, ACL '04.

George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. 2004. The automatic content extraction (ACE) program tasks, data, and evaluation. *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*.

Nan Du, Kai Chen, Anjuli Kannan, Linh Tran, Yuhui Chen, and Izhak Shafran. 2019. Extracting symptoms and their status from clinical conversations. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 915–925, Florence, Italy. Association for Computational Linguistics.

Greg P. Finley, Erik Edwards, Amanda Robinson, Najmeh Sadoughi, Mark Miller, David Suendermann-Oeft, and Nico Axtmann Michael Brenndoerfer. 2018a. An automated medical scribe for documenting clinical encounters. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Gregory Finley, Wael Salloum, Najmeh Sadoughi, Erik Edwards, Amanda Robinson, Nico Axtmann, Michael Brenndoerfer, Mark Miller, and David Suendermann-Oeft. 2018b. From dictations to clinical reports using machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*, pages 121–128. Association for Computational Linguistics.

Katrin Fundel, Robert Küffner, and Ralf Zimmer. 2007. RelEx–relation extraction using dependency parse trees. *Bioinformatics*, 23(3):365–371.

Pankaj Gupta, Subburam Rajaram, Hinrich Schütze, Bernt Andrassy, and Thomas A. Runkler. 2019. Neural relation extraction within and across sentence boundaries. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI) Conference on Artificial Intelligence*.

Xu Han, Zhiyuan Liu, and Maosong Sun. 2018. Neural knowledge acquisition via mutual attention between knowledge graph and text. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. SemEval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval '10, pages 33–38, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.

Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *Proceedings of the ACL 2004 on Interactive Poster and Demonstration Sessions*, ACLdemo '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Arzoo Katiyar and Claire Cardie. 2016. Investigating LSTMs for joint extraction of opinion entities and relations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 919–929, Berlin, Germany. Association for Computational Linguistics.

J D Kim, Y Wang, and Y Yasunori. 2013. The Genia event extraction shared task. *Proceedings of the BioNLP*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *ICLR*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in Neural Information Processing Systems 28*, pages 3294–3302.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270. Association for Computational Linguistics.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *ACL*.

ChunYang Liu, WenBo Sun, WenHan Chao, and Wanxiang Che. 2013. Convolution neural network for relation extraction. In *International Conference on Advanced Data Mining and Applications*, pages 231–242. Springer.

Feifan Liu, Chunhua Weng, and Hong Yu. 2012. *Natural Language Processing, Electronic Health Records, and Clinical Research*, pages 293–310. Springer Science & Business Media.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074. Association for Computational Linguistics.

Sebastian Martschat and Michael Strube. 2015. Latent structures for coreference resolution. *Transactions of the Association for Computational Linguistics*, 3(0):405–418.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA. Curran Associates Inc.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1116. Association for Computational Linguistics.

Makoto Miwa and Yutaka Sasaki. 2014. Modeling joint entity and relation extraction with table representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1858–1869, Doha, Qatar. Association for Computational Linguistics.

Raymond J. Mooney and Razvan C. Bunescu. 2006. Subsequence kernels for relation extraction. In Y. Weiss, B. Schölkopf, and J. C. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 171–178. MIT Press.

Pinal Patel, Disha Davey, Vishal Panchal, and Parth Pathak. 2018. Annotation of a large clinical entity corpus. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2033–2042. Association for Computational Linguistics.

Yifan Peng, Manabu Torii, Cathy H Wu, and K Vijay-Shanker. 2014. A generalizable NLP framework for fast development of pattern-based biomedical relation extraction systems. *BMC Bioinformatics*, 15(1):285.

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1756–1765. Association for Computational Linguistics.

Longhua Qian, Guodong Zhou, Fang Kong, Qiaoming Zhu, and Peide Qian. 2008. Exploiting constituent dependencies for tree kernel-based semantic relation extraction. In *Proceedings of the 22Nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 697–704, Stroudsburg, PA, USA. Association for Computational Linguistics.

Rion Snow, Daniel Jurafsky, and Andrew Y Ng. 2005. Learning syntactic patterns for automatic hypernym discovery. In *Advances in neural information processing systems*, pages 1297–1304.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, pages 1201–1211.

Ryuichi Takanobu, Tianyang Zhang, Jiexi Liu, and Minlie Huang. 2019. A hierarchical framework for relation extraction with reinforcement learning. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI) Conference on Artificial Intelligence*.

Özlem Uzuner, Brett R South, Shuying Shen, and Scott L DuVall. 2011a. 2010 i2b2/VA challenge on concepts, assertions, and relations in clinical text. *J. Am. Med. Inform. Assoc.*, 18(5):552–556.

Ozlem Uzuner, Brett R South, Shuying Shen, and Scott L. DuVall. 2011b. 2010 i2b2/va challenge on concepts, assertions, and relations in clinical text. *Journal of American Medical Informatics Association*, 18(5):552–6.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Patrick Verga, Emma Strubell, and Andrew McCallum. 2018. Simultaneously self-attending to all mentions for full-abstract biological relation extraction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 872–884. Association for Computational Linguistics.

Robert Wachter and Jeff Goldsmith. 2018. To combat physician burnout and improve care, fix the electronic health record. *Harvard Business Review*.

Rena Xu. 2018. The burnout crisis in american medicine. *The Atlantic*.

Zhilin Yang, Ruslan Salakhutdinov, and William Cohen. 2017. Transfer learning for sequence tagging with hierarchical recurrent networks. *ICLR)*.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *J. Mach. Learn. Res.*, 3:1083–1106.

Meishan Zhang, Yue Zhang, and Guohong Fu. 2017. End-to-end neural relation extraction with global optimization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1730–1740, Copenhagen, Denmark. Association for Computational Linguistics.

Xinsong Zhang, Pengshuai Li, Weijia Jia, and Hai Zhao. 2019. Multi-labeled relation extraction with attentive capsule network. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI) Conference on Artificial Intelligence*.

Yuhao Zhang, Peng Qi, and Christopher D Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2205–2215.

Suncong Zheng, Feng Wang, Hongyun Bao, Yuexing Hao, Peng Zhou, and Bo Xu. 2017. Joint extraction of entities and relations based on a novel tagging scheme. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1227–1236, Vancouver, Canada. Association for Computational Linguistics.

# Practical Correlated Topic Modeling and Analysis via the Rectified Anchor Word Algorithm

**Moontae Lee**[1] **Sungjun Cho**[2] **David Bindel**[2] **David Mimno**[2]
[1]University of Illinois at Chicago, Microsoft Research at Redmond
[2]Cornell University
moontae@uic.edu, {sc782,bindel,mimno}@cornell.edu

## Abstract

Despite great scalability on large data and their ability to understand correlations between topics, spectral topic models have not been widely used due to the absence of reliability in real data and lack of practical implementations. This paper aims to solidify the foundations of spectral topic inference and provide a practical implementation for anchor-based topic modeling. Beginning with vocabulary curation, we scrutinize every single inference step with other viable options. We also evaluate our matrix-based approach against popular alternatives including a tensor-based spectral method as well as probabilistic algorithms. Our quantitative and qualitative experiments demonstrate the power of Rectified Anchor Word algorithm in various real datasets, providing a complete guide to practical correlated topic modeling.

## 1 Introduction

Increasing access to massive data streams is useful only if it is equipped with proper tools to discover meaningful patterns. Topic models are capable of learning low-dimensional latent structures from groups of discrete observations, while being flexibly applicable to a wide range of modalities without human annotations. Users can assess consumer profiles by collecting purchase habits (Reisenbichler and Reutterer, 2019), shared sentiments or topics among comments in social networks (Nguyen et al., 2015), hidden genres/preferences on movie or music consumption (Lee et al., 2015), and latent communities from network snapshots (Gerlach et al., 2018). For clarity this paper sticks to the standard terms — words, documents, and topics — but the concepts generalize to various applications beyond these examples.

Traditional algorithms for topic modeling lack scalability. To learn quality topics, probabilistic algorithms such as Variational Inference (VI) or Markov Chain Monte Carlo (MCMC) require multiple passes through the input dataset until convergence, and thus struggle to process millions of documents. Online or stochastic algorithms (Hoffman et al., 2010, 2013) achieve some scalability but at the cost of sacrificing the quality of topics. As a result, the basic Latent Dirichlet Allocation (Blei et al., 2003) is still prevalent for practitioners despite various recent advances (Srivastava and Sutton, 2017; Xun et al., 2017; Xu et al., 2018). However, topics are *likely co-occurring terms* in essence. Spectral methods explicitly construct word co-occurrence moments as statistically unbiased estimators, providing alternatives to the probabilistic algorithms via moment-matching. Once the co-occurrence statistics is built with a single trivially parallelizable pass through the corpus, topic inference no longer needs to revisit individual training documents.

The Anchor Word algorithm (Arora et al., 2012, 2013; Lee et al., 2015) and tensor decomposition algorithms (Anandkumar et al., 2012a,b, 2013) factorize the second and third-order co-occurrence between pairs or triples of words, respectively, matching the corresponding posterior moments. In contrast to VI or MCMC, these spectral algorithms do not suffer from spurious local minima or slow mixing problems, learning consistently with provable guarantees under weak assumptions. However, inference is known to be sensitive to statistical noise, and its quality quickly degrades if the input data does not agree well with the underlying models similar to (Kulesza et al., 2014; Marinho, 2015). As a result, these algorithms have not been popular for real applications despite their great scalability.

This paper aims to provide a complete guide to practical correlated topic modeling. We first explain theoretical insights for spectral topic models based on the framework of Joint Stochastic Matrix Factorization (JSMF) (Lee et al., 2015). Then we



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4991–5001, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

introduce scalable implementations of the Rectified Anchor Word (RAW) algorithm and various evaluation metrics, investigating the impact of each inference step from vocabulary curation to topic inference. We also analyze quality of topics learned from annotated non-textual datasets as well as unsupervised textual corpora based on their top contributing words to the individual topics. To the best of our knowledge, this paper is the first comparative study that measures both quantitative and qualitative performance across different spectral topic models and their probabilistic counterparts.

The experimental results show that the rectification step in RAW is crucial for overcoming the *model-data mismatch* (Kulesza et al., 2014) but only needs a few iterations. The learned topics substantially outperform tensor-based methods and online VI, being comparable to expensive MCMC. Running RAW on a non-textual music dataset reveals quality genre topics, whereas the probabilistic correlated model (Blei and Lafferty, 2007) often learns overfitted topics that only maximize the co-occurrence of popular songs. To better support the community, we also provide scalable implementations in MATLAB[1] and Python[2] with full algorithmic details in the supplementary material.

## 2 Spectral Topic Inference

Topic modeling assumes a document representation that is sufficiently simple to allow tractable inference but also realistic to be useful. Each topic $k$ is defined as a distribution $p_{X|Z}(\cdot|k)$ over words where $p_{X|Z}(i|k)$ is a probability to choose a word $i$ given the topic $k$. Assuming there are $N$ words in the vocabulary and $K$ prepared topics, all topics can be compactly represented by the column-stochastic matrix $\boldsymbol{B} \in \mathbb{R}^{N\times K}$, where each column vector $\boldsymbol{b}_k \in \Delta^{N-1}$ stands for the topic $k$. Suppose there are $M$ documents in a corpus which are all written by admixing some of these $K$ topics with respect to a certain prior $\mathfrak{f}$. Then topic models explain that each document $m$ with length $n_m$ is written by: 1) Select a topic composition $\boldsymbol{w}_m \in \Delta^{K-1}$ with respect to $\mathfrak{f}$; 2) Write $n_m$ words by repeatedly selecting a topic $z$ from the composition $\boldsymbol{w}_m$ and a word $x$ from the topic $\boldsymbol{b}_z$.

Different models adopt different priors $\mathfrak{f}$ to better explain proper admixing of topics for the given data. For example, LDA assumes $\mathfrak{f} = \text{Dir}(\boldsymbol{\alpha})$ for $\boldsymbol{\alpha} \in \mathbb{R}^K_+$ (Blei et al., 2003). In correlated topic models, $\mathfrak{f} = \text{Logit-Normal}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ (CTM) or Probit-Normal$(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ for $\boldsymbol{\mu} \in \mathbb{R}^{K-1}$, $\boldsymbol{\Sigma} \in \mathbb{R}^{(K-1)\times(K-1)}$ (Blei and Lafferty, 2007; Yu and Fokoue, 2014). These models differ only in explaining the stochastic generation of topic composition: $\boldsymbol{w}_m \sim \mathfrak{f}$. Note that entries in every column vector $\boldsymbol{b}_k$ of $\boldsymbol{B}$ are parameters to recover in our setting, whereas probabilistic topic models often put another parametric prior $\mathfrak{g}(\boldsymbol{\beta})$ from which each $\boldsymbol{b}_k$ is sampled. The form of $\mathfrak{g}$ is not as crucial in learning quality topics as the form of $\mathfrak{f}$ (Asuncion et al., 2012), and can be similarly incorporated in spectral inference by putting additional regularizers when recovering each $\boldsymbol{b}_k$ (Nguyen et al., 2014).

Let $\boldsymbol{H} \in \mathbb{R}^{N\times M}$ be the word-document matrix where the $m$-th column vector $\boldsymbol{h}_m$ indicates the observed term-frequencies in document $m$. Say $\widetilde{\boldsymbol{H}}$ is the column-normalized $\boldsymbol{H}$ where each column is $\boldsymbol{h}_m/n_m$. Topic compositions of individual documents can also be described compactly by another column-stochastic matrix $\boldsymbol{W} \in \mathbb{R}^{K\times M}$ whose $m$-th column vector is $\boldsymbol{w}_m \in \Delta^{K-1}$. Then the main learning task of topic models is to find the word-topic matrix $\boldsymbol{B}$ and topic-document matrix $\boldsymbol{W}$ that approximates $\widetilde{\boldsymbol{H}} \approx \boldsymbol{B}\boldsymbol{W}$ with the column-stochastic constraints $\boldsymbol{B} \in \mathcal{CS}^{N\times K}$, $\boldsymbol{W} \in \mathcal{CS}^{K\times M}$. While this Non-negative Matrix factorization (NMF) is identifiable under additional sparsity constraints (Huang et al., 2014), directly applying NMF methods (Lee and Seung, 2001) produces incoherent topics despite small approximation errors (Stevens et al., 2012). $\boldsymbol{H}$ is too noisy and sparse to learn quality topics $\boldsymbol{B}$ and plausible compositions $\boldsymbol{W}$.

### 2.1 Joint Stochastic Matrix Factorization

Instead of directly decomposing $\widetilde{\boldsymbol{H}}$, JSMF decomposes smaller but aggregated statistics for revealing the latent topics and their correlations. Let $\boldsymbol{C} \in \mathbb{R}^{N\times N}$ be the empirical word co-occurrence matrix where $\boldsymbol{C}_{ij}$ is the joint probability $p_{X_1X_2}(i,j)$ to observe a pair of words $i$ and $j$ in the corpus. Define the topic co-occurrence matrix $\boldsymbol{A} \in \mathbb{R}^{K\times K}$ where $\boldsymbol{A}_{kl}$ is the joint probability $p_{Z_1Z_2}(k,l)$ between two latent topics $k$ and $l$. Then JSMF transforms the topic modeling objective into a second-order NMF: $\boldsymbol{C} \approx \boldsymbol{B}\boldsymbol{A}\boldsymbol{B}^T$, which is algebraically equivalent to $p(X_1, X_2|\boldsymbol{A}; \boldsymbol{B}) = \sum_{z_1}\sum_{z_2} p(X_1|Z_1; \boldsymbol{B})p(Z_1, Z_2|\boldsymbol{A})p(X_2|Z_2; \boldsymbol{B})$. The question is how this formulation provides better hints to learn the latent topics $\boldsymbol{B}$ given $\boldsymbol{C}$.

[1] https://github.com/moontae/jsmf-raw
[2] https://github.com/sc782/pyJSMF-RAW

Define $\boldsymbol{x}_1 \in \mathbb{R}^N$ as a random basis vector where only a single component corresponding to one randomly drawn word from the document $m$ is 1. Let $\boldsymbol{p}_m$ be the vector where its $i$-th component denotes the probability for word $i$ to occur in the document $m$. Then $\boldsymbol{p}_m = \boldsymbol{B}\boldsymbol{w}_m \in \mathbb{R}^N$, satisfying

$$\boldsymbol{x}_1 \sim \text{Categorical}(\boldsymbol{p}_m) \Rightarrow \mathbb{E}[\boldsymbol{x}_1|\boldsymbol{w}_m] = \boldsymbol{B}\boldsymbol{w}_m.$$

Denote $n_m$ consecutive random draws of words by $\{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_{n_m}\}$, and let $\boldsymbol{h}_m = \sum_{t=1}^{n_m} \boldsymbol{x}_t$. Then

$$\boldsymbol{h}_m \sim \text{Mult}(n_m, \boldsymbol{p}_m) \Rightarrow \mathbb{E}[\boldsymbol{h}_m|\boldsymbol{w}_m] = n_m\boldsymbol{B}\boldsymbol{w}_m.$$

As explained earlier, assuming that each observed $\boldsymbol{h}_m$ follows this model does not produce statistically meaningful information toward recovering $\boldsymbol{B}$. Since different words in each document $m$ share the same topic composition $\boldsymbol{w}_m$, however, the *cross moments* can provide useful information about co-occurring words even within a single document: $\mathbb{E}[\boldsymbol{h}_m\boldsymbol{h}_m^T|\boldsymbol{w}_m] = \mathbb{E}[\boldsymbol{h}_m|\boldsymbol{w}_m]\mathbb{E}[\boldsymbol{h}_m|\boldsymbol{w}_m]^T + \text{Cov}(\boldsymbol{h}_m|\boldsymbol{w}_m) = n_m(n_m - 1)\boldsymbol{B}\boldsymbol{w}_m\boldsymbol{w}_m^T\boldsymbol{B}^T + n_m \cdot \text{diag}(\boldsymbol{B}\boldsymbol{w}_m)$. Hence,

$$\frac{\mathbb{E}[\boldsymbol{h}_m\boldsymbol{h}_m^T|\boldsymbol{w}_m] - n_m \cdot \text{diag}(\boldsymbol{B}\boldsymbol{w}_m)}{n_m(n_m-1)} = \boldsymbol{B}\boldsymbol{w}_m\boldsymbol{w}_m^T\boldsymbol{B}^T.$$

Define the co-occurrence $\boldsymbol{C}_m$ for a single document $m$ in terms of the observed $\boldsymbol{h}_m$:

$$\boldsymbol{C}_m = \frac{\boldsymbol{h}_m\boldsymbol{h}_m^T - \text{diag}(\boldsymbol{h}_m)}{n_m(n_m-1)}. \tag{1}$$

If our observation $\boldsymbol{h}_m$ follows the model, then $\mathbb{E}[\boldsymbol{C}_m|\boldsymbol{w}_m] = \boldsymbol{B}\boldsymbol{w}_m\boldsymbol{w}_m\boldsymbol{B}^T$ by linearity of expectation. Then by the Law of Iterated Expectation,

$$\mathbb{E}[\boldsymbol{C}_m] = \mathbb{E}_{\boldsymbol{w}_m}[\mathbb{E}[\boldsymbol{C}_m|\boldsymbol{w}_m]] = \boldsymbol{B}\mathbb{E}_{\boldsymbol{w}_m}[\boldsymbol{w}_m\boldsymbol{w}_m^T]\boldsymbol{B}^T.$$

We can now construct the empirical word co-occurrence by averaging $\boldsymbol{C}_m$ across $M$ documents: $\boldsymbol{C} := \frac{1}{M}\sum_{m=1}^M \boldsymbol{C}_m$. Denoting the posterior topic-topic matrix by $\boldsymbol{A}^* := \frac{1}{M}\boldsymbol{W}\boldsymbol{W}^T \in \mathbb{R}^{K\times K}$, it is proven that $\boldsymbol{A}$ is entry-wisely close to both $\boldsymbol{A}^*$ and the population moments $\mathbb{E}_{\boldsymbol{w}\sim\mathfrak{f}}[\boldsymbol{w}\boldsymbol{w}^T]$ when $M$ is sufficiently large (Arora et al., 2012). Thus

$$\begin{aligned}\boldsymbol{C} \approx \mathbb{E}[\boldsymbol{C}] &= \boldsymbol{B}\big(\frac{1}{M}\sum_{m=1}^M \mathbb{E}_{\boldsymbol{w}_m}[\boldsymbol{w}_m\boldsymbol{w}_m^T]\big)\boldsymbol{B}^T \\ &= \boldsymbol{B}\mathbb{E}_{\boldsymbol{w}\sim\mathfrak{f}}[\boldsymbol{w}\boldsymbol{w}^T]\boldsymbol{B}^T \approx \boldsymbol{B}\boldsymbol{A}^*\boldsymbol{B}^T \approx \boldsymbol{B}\boldsymbol{A}\boldsymbol{B}^T.\end{aligned}$$

Once we construct the empirical moment $\boldsymbol{C}$ from the input data as an unbiased estimator of the underlying generative process, JSMF enables users to recover the correct $\boldsymbol{B}$ and $\boldsymbol{A}$ up to some precision by matching $\boldsymbol{C}$ to its posterior moments $\boldsymbol{B}\boldsymbol{A}^*\boldsymbol{B}^T$. The **separability assumption**: every topic has one specific *anchor word* that occurs only in the context of that topic, allows the model to satisfy non-negative-rank($\boldsymbol{B}$)=rank($\boldsymbol{B}$)=$K$, guaranteeing the existence of an identifiable factorization.

### 2.2 Tensor Decomposition

The separability assumption is necessary for JSMF because having only up to the second moments is not sufficient by itself to identify latent topics (Anandkumar et al., 2013). While one can release this assumption by adopting the *sufficiently scattered* condition, it maps the factorization into another NP-hard optimization problem (Huang et al., 2016). Alternatively, one can leverage third-order moments to provide sufficient statistics for identifiable topic inference (Anandkumar et al., 2012a,b). In contrast to JSMF, tensor-based algorithms first specify $\mathfrak{f}$ as a tractable parametric prior like the Dirichlet distribution. For example, if $\mathfrak{f} = \text{Dir}(\boldsymbol{\alpha})$ with $\alpha_0 = \sum_k \alpha_k$, then $\mathbb{E}^{(1st)}_{\boldsymbol{w}\sim\mathfrak{f}(\boldsymbol{\alpha})}[\boldsymbol{w}] = \boldsymbol{\alpha}/\alpha_0$, and

$$\mathbb{E}^{(2nd)}_{\boldsymbol{w}\sim\mathfrak{f}(\boldsymbol{\alpha})}[w_k w_l] = \begin{cases} \frac{\alpha_k(\alpha_k+1)}{\alpha_0(\alpha_0+1)} & (k=l) \\ \frac{\alpha_k\alpha_l}{\alpha_0(\alpha_0+1)} & (k\neq l)\end{cases}. \tag{2}$$

It makes the marginal expectations $\mathbb{E}[\boldsymbol{x}_1]$ and $\mathbb{E}[\boldsymbol{x}_1\boldsymbol{x}_2^T]$ further parametrized by $\boldsymbol{\alpha}$.

$$\begin{aligned}\mathbb{E}[\boldsymbol{x}_1] &= \mathbb{E}_{\boldsymbol{w}_m}[\boldsymbol{x}_1|\boldsymbol{w}_m] = B\mathbb{E}[\boldsymbol{w}_m] = B\boldsymbol{\alpha}/\alpha_0 \\ \mathbb{E}[\boldsymbol{x}_1\boldsymbol{x}_2^T] &= \mathbb{E}_{\boldsymbol{w}_m}[\mathbb{E}[\boldsymbol{x}_1|\boldsymbol{w}_m]\mathbb{E}[\boldsymbol{x}_2|\boldsymbol{w}_m]^T] \\ &= \boldsymbol{B}\mathbb{E}_{\boldsymbol{w}_m}[\boldsymbol{w}_m\boldsymbol{w}_m^T]\boldsymbol{B}^T = \boldsymbol{B}\mathbb{E}^{(2nd)}_{\boldsymbol{w}\sim\mathfrak{f}(\boldsymbol{\alpha})}\boldsymbol{B}^T\end{aligned}$$

Similarly we can represent up to the third moments:

$$\begin{aligned}\mathbb{E}[\boldsymbol{x}_1 \otimes \boldsymbol{x}_2] &= \mathbb{E}^{(2nd)}_{\boldsymbol{w}\sim\mathfrak{f}(\boldsymbol{\alpha})}[\boldsymbol{w}\otimes\boldsymbol{w}](\boldsymbol{B},\boldsymbol{B}), \\ \mathbb{E}[\boldsymbol{x}_1 \otimes \boldsymbol{x}_2 \otimes \boldsymbol{x}_3] &= \mathbb{E}^{(3rd)}_{\boldsymbol{w}\sim\mathfrak{f}(\boldsymbol{\alpha})}[\boldsymbol{w}^{\otimes 3}](\boldsymbol{B},\boldsymbol{B},\boldsymbol{B}).\end{aligned}$$

By assuming $\boldsymbol{w} \sim \text{Dir}(\boldsymbol{\alpha})$, we can fortunately attain closed form expressions of all three population moments only in terms of $\boldsymbol{B}$ and $\boldsymbol{\alpha}$, allowing the non-central second and third moments to be further represented by lower-order moments and $\alpha_0$ (Anandkumar et al., 2012a). Therefore, once users construct the empirical moments given the training data and choose $\alpha_0$, tensor decomposition allows us to recover $\boldsymbol{B}$ and $\boldsymbol{\alpha}$ up to some precision by

matching the empirical moments to these population moments.[3] But there are several caveats.

First, finding such closed-form moment combinations is not obvious. Normally all higher-order moments are necessary for learning with the general prior $\mathfrak{f}$ (Arabshahi and Anandkumar, 2017).[4] Second, $\mathbb{E}[\boldsymbol{w}^{\otimes 3}]$ should be a diagonal tensor in order to apply the popular CP-decomposition for learning topics $\boldsymbol{B}$. It means that we need to assume **uncorrelated topics** instead of the separability assumption. Whereas most large topics models are proven indeed separable (Ding et al., 2015), users of CP-decomposition can only capture weak negative correlations via the learned $\boldsymbol{\alpha}$ but depending on the user choices of $\alpha_0$ and $\mathfrak{f} = \mathrm{Dir}$. Tucker decomposition is another option for learning correlated topics, but it instead requires additional sparsity constraints on $\boldsymbol{B}$, demanding notably more parameters to be estimated (Anandkumar et al., 2013). Overall, correlated topic modeling via tensor decomposition is not as flexible as using JSMF even if we factor out the trivial difference in time and space complexities.

## 3 The Rectified Anchor Word Algorithm

Whereas probabilistic algorithms have an intrinsic capability to fit their models on the data that does not necessarily follow their generative processes, spectral algorithms are susceptible to model-data mismatch (Kulesza et al., 2014). The Rectified Anchor Word (RAW) algorithm (Lee et al., 2015) is the first working formalism that can learn quality topics and their correlations from real data. The overall algorithm consists of five clearly divided steps: 0) construct the word co-occurrence matrix $\boldsymbol{C}$; 1) rectify $\boldsymbol{C}$; 2) find the set of anchor words $\boldsymbol{S}$; 3) recover the topics $\boldsymbol{B}$; 4) recover the topic correlations $\boldsymbol{A}$. Each step has various algorithmic decisions that have been previously unclear. We carefully explore other viable options, providing details on efficient implementations in the supplementary material.

**Step 0: Create $\boldsymbol{C}$.** For spectral inference, we first construct the empirical word co-occurrence statistics as an unbiased estimator for the underlying generative process: $\boldsymbol{C} = (1/M)\sum_{m=1}^{M} \boldsymbol{C}_m$ with $\boldsymbol{C}_m$ specified in Equation (1). Due to the efficiency of anchor-based inference, the moment construction often becomes the most expensive step for large corpora, but it is trivially parallelizable as the last averaging step is the only computation that couples individual documents.

Instead of using the entire vocabulary, the standard procedure is to remove stop words and prune off rare words based on either corpus frequencies or tf-idf scores. Excluding words appearing on a majority of documents is also known to improve the quality of topics (Schofield et al., 2017a,b). Measuring the impact of vocabulary curation is not so straightforward in probabilistic topic models due to random draws in their algorithms. In contrast, spectral topic models learn topics consistently without any randomness. We later show how to pick the plausible size of vocabulary in the experiment section.

**Step 1: Rectify $\boldsymbol{C}$.** Rectifying the co-occurrence estimator is key to successful inference as low-rank spectral learning is highly susceptible to the mismatch between the model and the data (Lee et al., 2015). Though $\boldsymbol{C}$ is shown to be more statistically robust than $\widetilde{\boldsymbol{H}}$ (Arora et al., 2012), its empirical construction from real data hardly exhibits the proper structures of the posterior moments $\boldsymbol{B}\boldsymbol{A}^*\boldsymbol{B}^T$: low-rank ($\mathcal{LR}$), positive semidefinite ($\mathcal{PSD}$), nonnegative ($\mathcal{NN}$), and normalized ($\mathcal{NOR}$).[5] The rectification step transforms the noisy $\boldsymbol{C}$ into a desirable estimator by alternatingly projecting it to individual spaces (Lee et al., 2015). We also discover that cyclic Douglas-Rachford (DR) iterations can properly rectify $\boldsymbol{C}$, but its computational cost is almost twice as expensive as Alternating Projection (AP). Refer to the supplementary material for details.

By running a truncated eigenvalue decomposition, the first step of AP only finds $K$ largest eigenvalues $\boldsymbol{\Lambda}_K$ and the corresponding eigenvectors $\boldsymbol{U}$ at minimal cost. It then projects $\boldsymbol{C}$ to the intersection of $\mathcal{PSD}_N$ and $\mathcal{LR}_K$ by reconstructing $\boldsymbol{U}\boldsymbol{\Lambda}_K^{+}\boldsymbol{U}^T$. The next step orthogonally projects $\boldsymbol{C}$ to $\mathcal{NOR}_N$ by subtracting the mean average entry-wisely from the desired total, 1.0. The negative entries are then zeroed out in the subsequent projection to $\mathcal{NN}_N$. While the order of projec-

[3] JSMF does not ask users to specify $\alpha_0$, flexibly and transparently modeling arbitrary pairwise correlations between topics by the co-occurrence between pairs of the corresponding anchor words.

[4] They recently discover that having up to third-order moments suffices to perform CP-decomposition when $\mathfrak{f}$ is a class of Normalized Infinitely Divisible (NID) prior.

[5] Due to the diagonal penalty in (1) for the unbiased construction and the variance of the generative process, $\boldsymbol{C}$ is almost always full-rank and indefinite in finite real data.

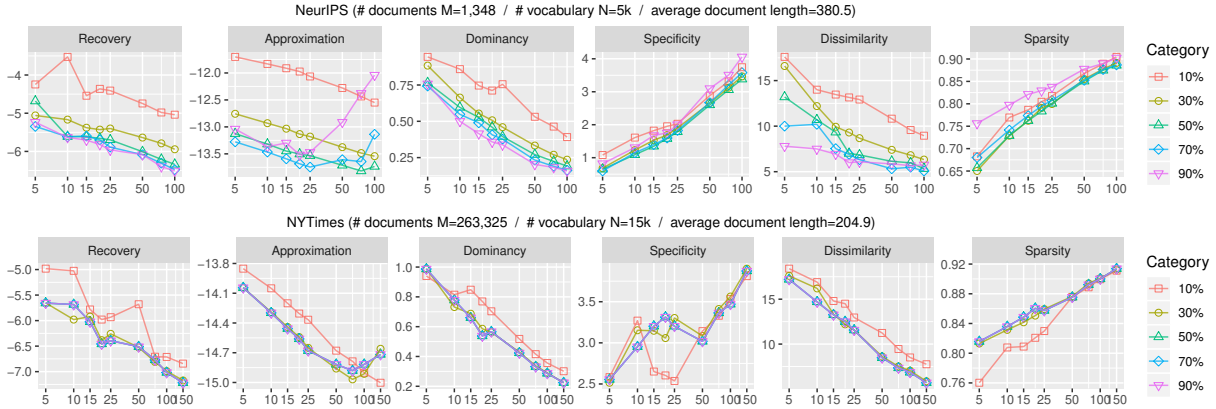


Figure 1: Vocabulary pruning assessed with AP+ADMM. A threshold of 50% implies that all words occurring in more than half of the documents are pruned. X-axis: the number of topics. Columns 1, 2, 3: lower is better / 4, 5, 6: higher is better.

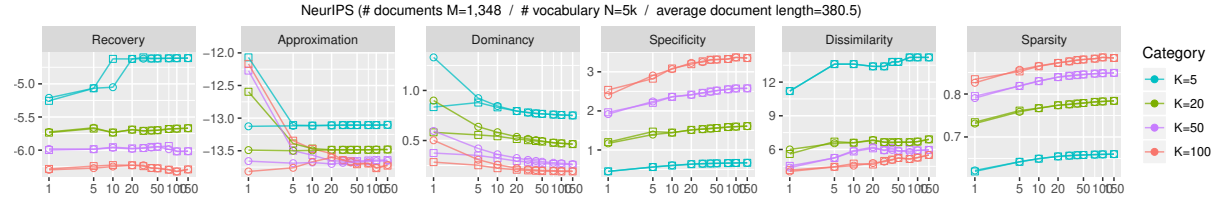


Figure 2: Alternating Projection (AP) vs Douglas-Rachford (DR). X-axis: the number of iterations in rectification. ⊟ AP+ADMM and ⊖ DR+ADMM mostly agree each other and converge within 15-20 iterations. 5 iterations are often enough.

tions in each iteration does not matter, performing a $\mathcal{NOR}_N$-projection after the loop ends helps with feasibility.[6] Note that tensor-based methods have a similar step called *whitening*, which runs a full SVD to transform the third-order moments into an orthogonal tensor for CP-decomposition.

**Step 2: Find $S$.** Say $\boldsymbol{C}$ now indicates the rectified co-occurrence. Then the next step is to find the anchor words. Denoting the set of $K$ anchor words by $\boldsymbol{S} = \{s_1, ..., s_K\}$, the separability assumption suggests: $p_{Z_1|X_1}(k'|s_k) = 1$ if $k' = k$ and $p_{Z_1|X_1}(k'|s_k) = 0$ if $k' \neq k$. Let $\overline{\boldsymbol{C}}$ be the row-normalized version of $\boldsymbol{C}$. Then by the conditional independence between a pair of words given one of their topics ($X_1 \perp X_2 | Z_1$ or $Z_2$) and separability, $\overline{\boldsymbol{C}}_{ij} = p_{X_2|X_1}(j|i) = \sum_{k'} p_{X_2|Z_1}(j|k') p_{Z_1|X_1}(k'|i)$. Thus $\overline{\boldsymbol{C}}_{ij} = \sum_k p_{Z_1|X_1}(k|i) \overline{\boldsymbol{C}}_{s_k,j}$, implying that every row vector of $\overline{\boldsymbol{C}}$ corresponding to a non-anchor word can be represented by a convex combination: $\sum_k p_{Z_1|X_1}(k|i) = 1$ of the rows $\{\overline{\boldsymbol{C}}_{s_k}\}$ corresponding to the anchor words $\{s_k\}$. Therefore, the inference quality depends primarily on the choice of the anchor words $S$, providing a clear metric for diagnosis. Note that rectification is also crucial for finding better anchors (Lee et al., 2015).

While using the pivoted QR (Arora et al., 2013) substantially expedites the running time against solving a number of LPs (Arora et al., 2012), its explicit projection of each non-anchor row to the current orthogonal complement quickly damages the sparsity of $\overline{\boldsymbol{C}}$. As a result, random projections are suggested for sizable vocabulary (Arora et al., 2013), but such projections do not maintain the joint-stochasticity of the rectified $\boldsymbol{C}$, degrading the topic quality (Lee and Mimno, 2014). Instead, we develop a sparse implicit pivoted QR that requires only $\mathcal{O}(NK)$ space to store the basis rows and performs implicit updates in $\mathcal{O}(nnz(\boldsymbol{C})K)$ time without modifying any entry in the input $\overline{\boldsymbol{C}}$.

**Step 3: Recover $B$.** Provided with the set of anchor words $S$ and the convex coefficients $\breve{\boldsymbol{B}}_{ki} = \{p_{Z_1|X_1}(k|i)\}$, one can easily recover $\boldsymbol{B}$ by applying Bayes' rule: $\boldsymbol{B}_{ik} = (\breve{\boldsymbol{B}}_{ki} c_i) / (\sum_{i'=1}^{N} \breve{\boldsymbol{B}}_{ki'} c_{i'})$, where $c_i := p_{X_1}(i)$ is the unigram probability of the word $i$, which is equal to $\sum_j \boldsymbol{C}_{ij}$. Hence the core of this step is to find the coefficient matrix $\breve{\boldsymbol{B}}$ by solving a Simplex-Constrained Least Squares (SCLS) that satisfy $\overline{\boldsymbol{C}}_{ij} = \sum_k \breve{\boldsymbol{B}}_{ki} \overline{\boldsymbol{C}}_{s_k,j}$ for each $i$. While the exponentiated gradient algorithm (ExpGrad) used in previous work (Arora et al., 2013; Lee et al., 2015) converges quickly, it is unclear how to tune the learning rate, weakening confidence in the learned topics. We propose another algorithm based on Alternating Direction Method

[6] Avoiding negative entries is useful because RAW is not a purely algebraic algorithm but uses probabilistic conditions.

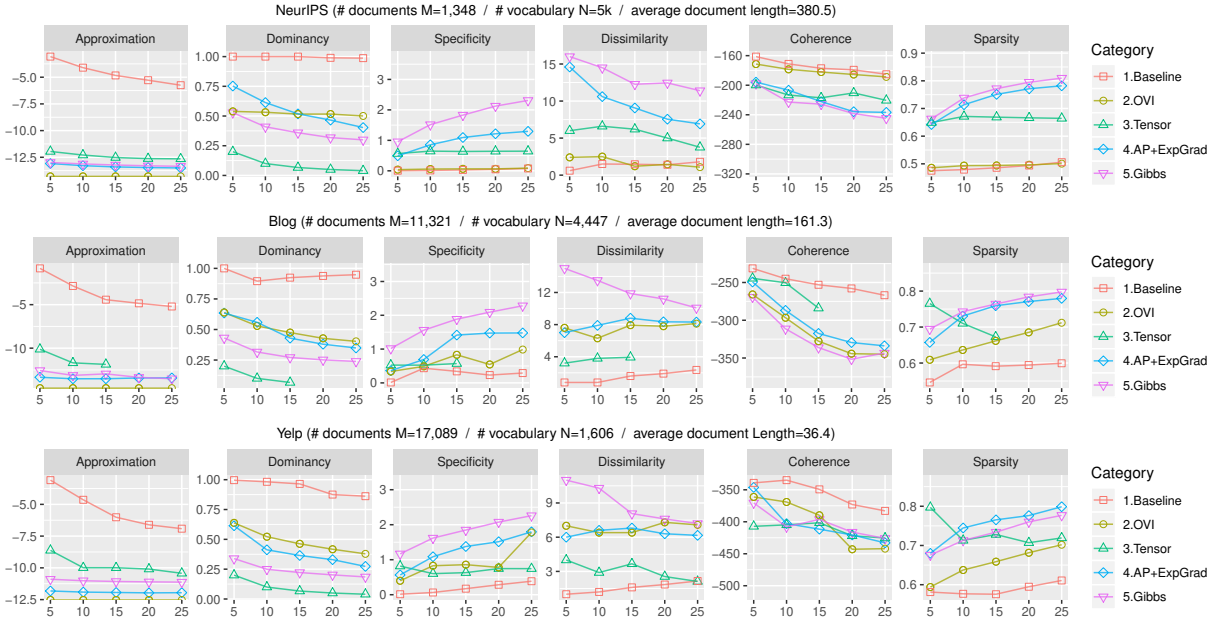


Figure 3: Quantitative results from various methods. Tensor (CP-decomposition (Anandkumar et al., 2012a)) performs better than the Baseline (Anchor Word algorithm with ExpGrad without any rectification (Arora et al., 2013)) and OVI (Online Variational Inference (Hoffman et al., 2010)), but much poorer than the AP+ExpGrad (AP-rectified Anchor Word algorithm (Lee et al., 2015)) and Gibbs (Collapsed Gibbs Sampling (Yao et al., 2009)). Surprisingly the tensor algorithm does not show consistent behavior for increasing number of topics in X-axis. Closer to Gibbs is generally better in Y-axis.

of Multipliers (ADMM), which is not sensitive to different parameter settings. Note that we also provide the Active-Set method that can solve SCLS within machine precision in our implementation, but our practical choice is ADMM due to the much higher cost of running the Active-Set method.

**Step 4: Recover $\boldsymbol{A}$.** The final step is to recover the topic correlation matrix $\boldsymbol{A}$, which summarizes the latent topic compositions $\boldsymbol{W}$ by $\boldsymbol{A} = (1/M)\boldsymbol{W}\boldsymbol{W}^T$. Instead of learning $\boldsymbol{W}$, Anchor Word algorithms learn the correlations $\boldsymbol{A}$ by again leveraging the separability assumption: $\sum_{l'}\big(\sum_{k'} p_{X_1|Z_1}(s_k|k')p_{Z_1Z_2}(k',l')\big)p_{X_2|Z_2}(s_l|l')$ $= p_{X_1|Z_1}(s_k|k)\big(\sum_{l'} p_{Z_1Z_2}(k,l')p_{X_2|Z_2}(s_l|l')\big)$ $= p_{X_1|Z_1}(s_k|k)p_{Z_1Z_2}(k,l)p_{X_2|Z_2}(s_l|l)$. Thus $p_{Z_1Z_2}(k,l) = p^{-1}_{X_1|Z_1}(s_k|k)p_{X_1X_2}(s_k,s_l)p^{-1}_{X_2|Z_2}(s_l|l)$, which can be simplified to $\boldsymbol{A} = \boldsymbol{B}^{-1}_{S*}\boldsymbol{C}_{SS}\boldsymbol{B}^{-1}_{S*}$. Therefore the co-occurrence of the anchor words $s_k$ and $s_l$ transparently captures the correlation between the pair of topics $k$ and $l$. Note that the anchor words are generally rare words — in order to be the vertices of an underlying convex hull of the word co-occurrence space — whose co-occurrences are even rarer and noisier. The rectification step in JSMF effectively balances these entries (Lee et al., 2015), thereby realizing robust and transparent correlated topic inference.

## 4 Experimental Results

We evaluate our models of interest on two standard textual corpora: NeurIPS and NYTimes. Full papers in NeurIPS are generally longer but share a smaller vocabulary (12k), whereas massive news articles in NYTimes have medium length with the largest vocabulary (103k). Due to high complexities of tensor decomposition, we prepare two small textual datasets: Blog and Yelp. They consist of political blogs (Eisenstein et al., 2011) and business reviews (Lee and Mimno, 2014) with the smallest vocabulary (4.4k, 1.6k), respectively. In addition, we adopt two non-textual preference datasets: Movies and Songs. They include 10m movie reviews[7] and music playlists from Yes.com.[8] In contrast to textual datasets, we can retrieve genre information for Movies and Songs.[9] You can find the exact statistics of each dataset in our figures.

Evaluating topic models with held-out likelihoods or perplexities only is often misleading (Passos et al., 2011; Lee and Mimno, 2014). Instead we follow six metrics used on (Lee et al.,

[7]Movies has a vocabulary of 10.7k movies. https://grouplens.org/datasets/movielens/10m/

[8]Songs has a vocabulary of 75.3k songs. http://csinpi.github.io/lme/data_page.html

[9]Genre information are already annotated as tags in these two non-textual datasets. If missing, we scrape the information from IMDB and Discogs.com, respectively.

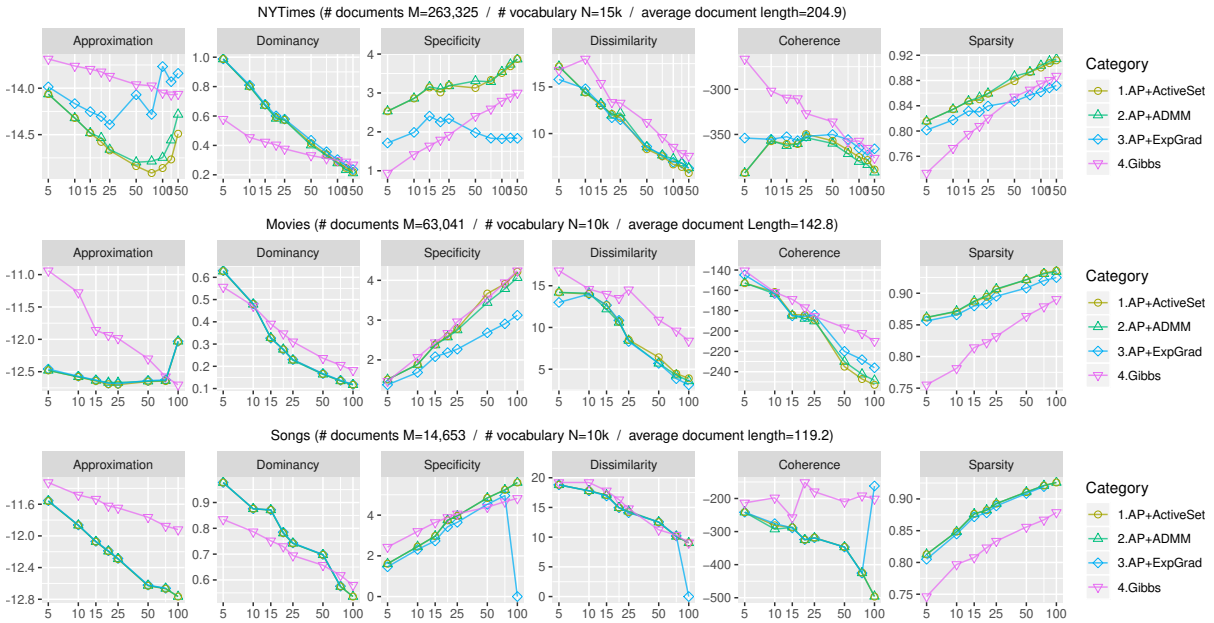


Figure 4: ExpGrad vs ADMM vs ActiveSet. Our △ AP+ADMM and ◇ AP+ActiveSet algorithms outperform the previous state-of-the-art rectified algorithm ◇ AP+ExpGrad, being more comparable to probabilistic ▽ Gibbs sampling. Columns 1, 2: lower is better / 3, 4, 6: higher is better / 5: closer to Gibbs is better.

2015) for fair and comprehensive evaluations. **Recovery** ($\frac{1}{N}\sum_i \|\overline{\boldsymbol{C}}_i - \sum_k \breve{\boldsymbol{B}}_{ki}\overline{\boldsymbol{C}}_{s_k}\|_2$) evaluates how successfully anchor words reconstruct the word co-occurrence space. **Approximation** ($\|\boldsymbol{C} - \boldsymbol{BAB}^T\|_F$) measures the closeness between the learned factorization and the unbiased co-occurrence statistics. Note that they are measured against *the original* $\boldsymbol{C}$ rather than the rectified one, and are visualized in logarithms of base 1.8 for readability. **Dominancy** ($\frac{1}{K}\sum_k \boldsymbol{A}_{kk}$) is the average self-correlations, indirectly gauging the loss of correlations between different topics. **Specificity** ($\frac{1}{K}\sum_k \mathrm{KL}(\boldsymbol{b}_k||\sum_i \boldsymbol{C}_{*i})$) measures the average KL-distance of each topic from the unigram distribution of the corpus. **Dissimilarity** counts the mean number of top words in each topic that do not belong to the top 20 words of other topics. **Coherence** ($\frac{1}{K}\sum_k \sum_{x_1,x_2\in Top_k}^{x_1\neq x_2} \log\frac{D_2(x_1,x_2)+\epsilon}{D_1(x_2)}$) penalizes any pair of top words in each topic that do not appear together in the training documents.[10] While we report Coherence, the metric could be deceptive if a model learns many duplicated topics whose top words are mostly frequent words (Huang et al., 2016). Thus following the trends of Gibbs generally implies better performance. We newly add **Sparsity** ($\frac{1}{K}\sum_k \frac{\sqrt{N}-(\|\boldsymbol{b}_k\|_1/\|\boldsymbol{b}_k\|_2)}{\sqrt{N}-1}$) (Hoyer, 2004) to gauge the average sparsity of the topics.

[10] $D_2(x_1, x_2)$ means the number of training documents where two words $x_1$ and $x_2$ jointly appear. $D_1(x_2)$ counts the number of training documents that include the word $x_2$.

**Vocab pruning:** We experiment different document frequency cut-offs. Figure 1 shows that removing all the words that occur more than 10% the documents is too aggressive, thus showing inconsistent behavior as the number of topics grows. In contrast, using 90% cut-off saves too many words. We process vocabulary identically to (Lee et al., 2015) for fair comparison, discarding the words that appear in more than 50% of the documents. The title on top of each figure indicates the size of the pruned vocabulary with the specific statistics.

**Quantitative analysis:** After constructing $\boldsymbol{C}$ (Step 0), the Baseline method (Arora et al., 2013) skips to finding the anchor words (Step 2) without any rectification. While we use the exponentiated gradient (ExpGrad) method adopted in the previous work (Arora et al., 2013), we do not perform any random projection or pseudo-inverse recovery of $\boldsymbol{A}$ in order to prevent further degradation of learning quality. For methods within the framework of JSMF, we execute 150 iterations of AP or DR rectifications (Step 1) , which is equivalent to (Lee et al., 2015). However, Figure 2 shows that running only 5 iterations of AP or DR sufficiently rectifies $\boldsymbol{C}$, and 15-20 iterations yields almost identical results to 150 iterations.

Whereas our new anchor-finding method (Step 2) only improves the time/space complexity, the method choice for SCLS (Step 3) actually affects

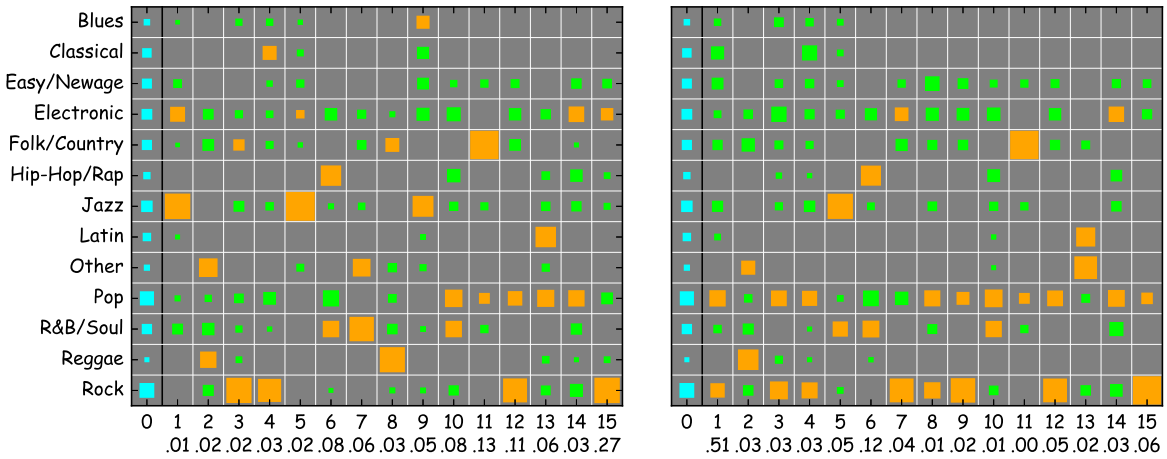


Figure 5: AP+ADMM (left) vs CTM (right). The column 0 shows the genre distribution of the entire corpus. Each column 1-15 stands for $k$-th topic where two most prominent genres are of orange colors. The size of each box is proportional to the relative intensity. Fractional value below each topic number on the X-axis indicates the marginal probability $p_{z_1}(k)$ of the latent topic $k$. AP+ADMM learns better topics that capture more coherent information about music genres.

the quality of topics. For ExpGrad, we set the learning rate as 50.0, which is the best-known from (Lee et al., 2015). For our ADMM, we set $\lambda = 1.9$ and $\gamma = 3.0$, but we also find that the algorithm is not at all sensitive to different settings. For probabilistic inference, we adopt a sufficiently mixed collapsed Gibbs Sampling (Gibbs) from the standard Mallet library[11], using 1,000 iterations after discarding the initial 200 burn-in samples. We also run Online Variational Inference (OVI) (Hoffman et al., 2010) in the standard Gensim package. Finally, we run CP-decomposition for the Tensor algorithm.[12]

Figure 3 shows that Tensor decomposition outperforms Baseline and OVI, but evaluation metrics fluctuate as the number of topics increases. We also observe that the topic distribution given a word becomes closer to uniform over growing number of topics. In contrast, AP+ExpGrad exhibits consistent behaviors as expected in spectral inference, being most comparable to Gibbs. Though SCLS (Step 3) is a convex problem, Figure 4 shows that AP+ADMM and AP+ActiveSet improve Specificity and Sparsity over AP+ExpGrad, making the learned topics even more comparable to Gibbs. We choose ADMM as our main optimization method especially because it is not only insensitive to its parameters, but also notably faster than ActiveSet. Users of ExpGrad must search through less intuitive learning rates for optimal performance, which can be different for each dataset.

[11]Gibbs is run on Java Mallet that implements time- and memory-efficient sampling with optimized multicore controls.

[12]https://github.com/FurongHuang/TensorDecomposition4TopicModeling

**Qualitative analysis:** We first verify the learned topics in the NeurIPS dataset. As given in Figure 6, AP+ADMM and Gibbs learn comparable topics, while the topics learned from OVI and Tensor are not sufficiently separated: OVI repeats 'cell' and 'neuron' in different topics. Similarly, 'neuron' and 'layer' contribute to nearly every topic in Tensor regardless of hidden differences in their themes.

When running Variational CTM (Blei and Lafferty, 2007) with default parameters, the resulting topics do not show distinguishable genre associations. Most topics involve Pop and Rock, emulating the overall genre distribution of the corpus as illustrated in Figure 5. In contrast, our AP+ADMM captures three Jazz topics (T1: Electronic, T5: Pure, T9: Blues style) and four specific Rock topics (T3: Folk Rock, T4: Rock n Roll, T12: Pop style, T15: Alternative Rock). While both models discover Reggae and Latin genres, CTM's associate more with generic Other genres, whereas AP+ADMM's associate more with Folk/Country or Pop.

In addition, CTM puts spuriously high marginal topic probability on T1, which is the topic closest to the corpus genre distribution. While it can highly contribute to maximizing the likelihood of the data, an unseen playlist would most likely be classified as a mixture of Pop and Rock even if it only contains a couple of Pop or Rock songs. This also happens in Movies, explaining why we prefer using various metrics than merely measuring the held-out likelihood or perplexity. Genre association in Movies turns out to be less clear than in Songs. While songs within a playlist are likely to

| OVI (Hoffman et al., 2010) | AP+ADMM (This paper) |
|---|---|
| cell layer object neuron node | neuron circuit synaptic cell layer |
| layer neuron node output rbf | control action dynamic optimal controller |
| image cell filter neuron vector | recognition layer hidden word speech |
| neuron cell object map activity | cell field visual direction image |
| cell object recognition layer vector | gaussian noise hidden approximation matrix |
| **Tensor (Anandkumar et al., 2012a)** | **Gibbs (Yao et al., 2009)** |
| cell neuron field visual direction | neuron cell visual signal response |
| layer hidden neuron field approximation | control action policy optimal reinforcement |
| object image layer recognition field | recognition image object feature word |
| neuron layer hidden threshold synaptic | hidden net layer dynamic neuron |
| hidden noise gaussian layer approximation | gaussian approximation matrix bound component |

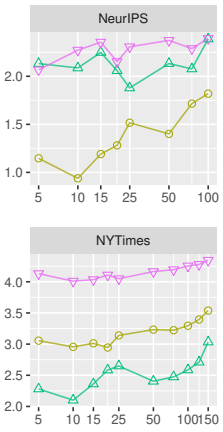


Figure 6: (Left) Each line consisted of top 5 words represents a topic from NeurIPS ($K = 5$). Both OVI and Tensor tend to repeat top words across different topics, whereas AP+ADMM discovers distinctive and meaningful topics similar to Gibbs. (Right) Scalability of the methods measured in $log_{10}$(seconds). Compared to ○ OVI and ▽ Gibbs, our △ AP+ADMM can infer quality topics while being more scalable to large corpora.

share a genre-specific theme, people often watch and review recently released movies rather than coherently consume movies within related genres. Thus topics inferred from Movies consist of year-specific topics as well as Fantasy or Sci-Fi.

## 5 Discussion

Runtime analyses on RAW alongside other methods demonstrate its strong scalability. In our experiment, the tensor algorithm takes 5 hours for learning 5 topics on NeurIPS and 48 days for 25 topics on Yelp. It runs indefinitely for 20-25 topics on Blog, which explains two missing data points in Figure 3. Topic learning with RAW takes less than a minute on these toy datasets and less than an hour on the largest NYTimes when using 15 iterations of AP with ADMM.

Two plots in Figure 6 further verify that RAW with AP+ADMM is approximately 4 times faster than OVI (in addition to learning better topics), and 40 times faster than Gibbs (but showing comparable results across various numbers of topics) on the NYTimes dataset. While OVI runs faster on NeurIPS, the superior quality of topics inferred with RAW far outweighs the additional cost. As running the entire pipeline of RAW takes less than 5 minutes in NeurIPS, OVI is not as competitive as RAW.[13] Lastly, the Variational CTM takes 15 minutes to learn 15 topics on Songs, but 6 hours for 50 topics. In contrast, our RAW method takes less than 10 minutes to find 50 topics on Songs.

[13]Note that slight fluctuations in Figure 6 are due to the load-balancing from the job queue on our high-performance computing cluster. For precision, we draw these two panels by averaging the running times from 10 different trials.

## 6 Conclusion

By removing the dependency on the training documents, spectral topic modeling provides scalable formalisms for finding compact high-level structures in sparse and discrete data such as text and user-preference. The Rectified Anchor Word (RAW) algorithm enjoys its transparent and consistent behaviors, working seamlessly on various types of textual and non-textual real datasets. In particular, our AP+ADMM algorithm outperforms the previous AP+ExpGrad (Lee et al., 2015), being more comparable to Gibbs sampling and less sensitive to parameters.

Through this paper, we closely investigate each step of inference with various algorithmic decisions. Proper pruning of vocabulary is shown necessary, and rectification is proven crucial for reliable topic inference under the model-data mismatch. Joint Stochastic Matrix Factorization (JSMF) with the rectification better models arbitrary pairwise topic correlations at lower cost than probabilistic correlated topic model and tensor decomposition. We hope that our work, built upon the theoretical insights on spectral inference, provide a complete guide to correlated topic modeling for both researchers and practitioners.

## Acknowledgements

This work was completed when the first author was visiting Microsoft Research at Redmond. We thank both the department of Information and Decision Sciences in UIC Business School and the Deep Learning group in Microsoft Research. They provided useful resources and thoughtful advice.

## References


Anima Anandkumar, Dean P. Foster, Daniel Hsu, Sham Kakade, and Yi-Kai Liu. 2012a. A spectral algorithm for latent Dirichlet allocation. In *NIPS*.

Animashree Anandkumar, Daniel J. Hsu, Majid Janzamin, and Sham Kakade. 2013. When are overcomplete topic models identifiable? uniqueness of tensor tucker decompositions with structured sparsity.

Animashree Anandkumar, Sham M Kakade, Dean P Foster, Yi-Kai Liu, and Daniel Hsu. 2012b. Two svds suffice: Spectral decompositions for probabilistic topic modeling and latent dirichlet allocation.

F. Arabshahi and A. Anandkumar. 2017. Spectral methods for correlated topic models. *AISTATS*.

S. Arora, R. Ge, and A. Moitra. 2012. Learning topic models – going beyond SVD. In *FOCS*.

Sanjeev Arora, Rong Ge, Yonatan Halpern, David Mimno, Ankur Moitra, David Sontag, Yichen Wu, and Michael Zhu. 2013. A practical algorithm for topic modeling with provable guarantees. In *ICML*.

Arthur U. Asuncion, Max Welling, Padhraic Smyth, and Yee Whye Teh. 2012. On smoothing and inference for topic models. *CoRR*, abs/1205.2662.

D. Blei and J. Lafferty. 2007. A correlated topic model of science. *Annals of Applied Statistics*.

D. Blei, A. Ng, and M. Jordan. 2003. Latent Dirichlet allocation. *JMLR*.

Weicong Ding, Prakash Ishwar, and Venkatesh Saligrama. 2015. Most large topic models are approximately separable. In *ITA, 2015*, pages 199–203. IEEE.

Jacob Eisenstein, Duen Horng Chau, Aniket Kittur, and Eric P. Xing. 2011. Topicviz: Semantic navigation of document collections. *CoRR*, abs/1110.6200.

Martin Gerlach, Tiago P. Peixoto, and Eduardo G. Altmann. 2018. A network approach to topic models. *Science Advances*, 4(7).

Matthew D. Hoffman, David M. Blei, and Francis Bach. 2010. Online learning for latent dirichlet allocation. In *NIPS*.

Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. 2013. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347.

Patrik O. Hoyer. 2004. Non-negative matrix factorization with sparseness constraints. *JMLR*.

K. Huang, N. D. Sidiropoulos, and A. Swami. 2014. Non-negative matrix factorization revisited: Uniqueness and algorithm for symmetric decomposition. *IEEE Transactions on Signal Processing*.

Kejun Huang, Xiao Fu, and Nikolaos D. Sidiropoulos. 2016. Anchor-free correlated topic modeling: Identifiability and algorithm. In *NIPS*.

Alex Kulesza, N Raj Rao, and Satinder Singh. 2014. Low-rank spectral learning. In *AISTATS*.

Daniel D. Lee and H. Sebastian Seung. 2001. Algorithms for non-negative matrix factorization. In *NIPS*.

Moontae Lee, David Bindel, and David Mimno. 2015. Robust spectral inference for joint stochastic matrix factorization. In *NIPS*.

Moontae Lee and David Mimno. 2014. Low-dimensional embeddings for interpretable anchor-based topic inference. In *EMNLP*. Association for Computational Linguistics.

Zita Marinho. 2015. Moment-based algorithms for structured prediction.

Thang Nguyen, Yuening Hu, and Jordan Boyd-Graber. 2014. Anchors regularized: Adding robustness and extensibility to scalable topic-modeling algorithms. In *ACL*.

Thien Hai Nguyen, Kiyoaki Shirai, and Julien Velcin. 2015. Sentiment analysis on social media for stock movement prediction. *Expert Systems with Applications*, 42(24):9603 – 9611.

Alexandre Passos, Hanna Wallach, and Andrew McCallum. 2011. Correlations and anticorrelations in lda inference. In *NIPS*.

Martin Reisenbichler and Thomas Reutterer. 2019. Topic modeling in marketing: recent advances and research opportunities. *Journal of Business Economics*, 89(3):327–356.

Alexandra Schofield, Måns Magnusson, and D Mimno. 2017a. Understanding text pre-processing for latent dirichlet allocation. In *Proceedings of the 15th conference of the European chapter of the Association for Computational Linguistics*, volume 2, pages 432–436.

Alexandra Schofield, Måns Magnusson, and David Mimno. 2017b. Pulling out the stops: Rethinking stopword removal for topic models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 432–436.

Akash Srivastava and Charles A. Sutton. 2017. Autoencoding variational inference for topic models. In *ICLR*.

Keith Stevens, Philip Kegelmeyer, David Andrzejewski, and David Buttler. 2012. Exploring topic coherence over many models and many topics. In *EMNLP-CoNLL*.

Hongteng Xu, Wenlin Wang, Wei Liu, and Lawrence Carin. 2018. Distilled wasserstein learning for word embedding and topic modeling. In *Advances in Neural Information Processing Systems 31*, pages 1716–1725.

Guangxu Xun, Yaliang Li, Wayne Xin Zhao, Jing Gao, and Aidong Zhang. 2017. A correlated topic model using word embeddings. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, pages 4207–4213.

Limin Yao, David Mimno, and Andrew McCallum. 2009. Efficient methods for topic model inference on streaming document collections. In *KDD*.

Xingchen Yu and Ernest Fokoue. 2014. Probit normal correlated topic model. In *Open Journal of Statistics*, pages 879–888.

# Modeling the Relationship between User Comments and Edits in Document Revision

**Xuchao Zhang**[†*], **Dheeraj Rajagopal**[‡*], **Michael Gamon**[§], **Sujay Kumar Jauhar**[§] and **Chang-Tien Lu**[†]
[†]Discovery Analytics Center, Virginia Tech, Falls Church, VA, USA
[‡]Carnegie Mellon University, Pittsburgh, PA, USA
[§]Microsoft Research, Redmond, WA, USA
[†]{xuczhang, ctlu}@vt.edu, [‡]dheeraj@cs.cmu.edu, [§]{mgamon, sjauhar}@microsoft.com

## Abstract

Management of collaborative documents can be difficult, given the profusion of edits and comments that multiple authors make during a document's evolution. Reliably modeling the relationship between edits and comments is a crucial step towards helping the user keep track of a document in flux. A number of authoring tasks, such as categorizing and summarizing edits, detecting completed to-dos, and visually rearranging comments could benefit from such a contribution. Thus, in this paper we explore the relationship between comments and edits by defining two novel, related tasks: Comment Ranking and Edit Anchoring. We begin by collecting a dataset with more than half a million comment-edit pairs based on Wikipedia revision histories. We then propose a hierarchical multi-layer deep neural-network to model the relationship between edits and comments. Our architecture tackles both Comment Ranking and Edit Anchoring tasks by encoding specific edit actions such as additions and deletions, while also accounting for document context. In a number of evaluation settings, our experimental results show that our approach outperforms several strong baselines significantly. We are able to achieve a precision@1 of 71.0% and a precision@3 of 94.4% for Comment Ranking, while we achieve 74.4% accuracy on Edit Anchoring.

## 1 Introduction

Comments are widely used in collaborative document writing as a natural way to suggest and track changes, annotate content and explain the intent of edits. Table 1 shows an example of a comment left by an editor in a Wikipedia revision. The comment explains the intent of adding a missing researcher's name (Sutskever) to a citation.

This example demonstrates that user comments closely relate to the intent of an edit, the actual edit operation, and the content and location in the document that underwent change. However, during collaborative document authoring, the tracking and maintenance of comments becomes increasingly more challenging due to the large number of edits and comments that authors make. For example, structurally refactoring a document can significantly change the order of paragraphs and sentences, stranding comments in confusing and contextually inappropriate locations. Or, comments may have already been addressed but continue to linger in the document without having been marked as completed. These issues, among others, are exacerbated when multiple authors collaborate on a document, often asynchronously. It becomes difficult to know which tasks have been completed and which haven't, especially if authors are not proactive about marking comments as addressed. This situation stands to benefit from an intelligent system capable of marking changes as completed, by understanding the relationship between edits and comments.

Many tasks in document management, such as summarizing and categorizing edits, detecting to-do item completion, prioritizing writing tasks and visually re-arranging document structure to reflect state in multi-author scenarios, require users to understand edit-comment interactions. Consider the scenario where multiple authors make edits to a document using the track-change feature available in popular document authoring apps. The result can be extremely confusing, and it is difficult to disentangle who edited what, over multiple versions. A feature that could summarize these edits, so that the visual burden of tracking changes is not on the UI, would certainly alleviate this problem. Such a system would necessarily first need to learn mappings between edits and natural lan-

[*] The work was done while these authors interned in Microsoft Research.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5002–5011, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

guage comments, before it could learn to generate them. Therefore, automatic solutions to these challenges stand to benefit from an ability to fundamentally model the relationship between edits and comments.

Yet, most existing studies on document revisions focus on modeling the document edits (Bronner and Monz, 2012) or comments (Shaver, 2016) separately; or using comments as a supplementary source of information to study the edits (Yatskar et al., 2010). To the best of our knowledge, no prior work focuses on jointly modeling the relationship between comments and edits.

Thus, in this paper we tackle two novel tasks. **Comment Ranking** considers a document edit operation and seeks to rank a set of candidate comments in order of their relevance to the edit. **Edit Anchoring** seeks to identify the locations in a document that are most likely to have undergone change as the result of a specific comment. Crucially, both tasks require jointly modeling comment-edit relationship.

We start by collecting a dataset of 780K Wikipedia revisions, each with their associated comment and edits. We then build a hierarchical multi-layer deep neural network, a model we call CmntEdit, which is capable of learning the relationship between comments and edits from this data. Our approach addresses both Comment Ranking and Edit Anchoring by sharing many of the model's components and parameters across tasks. Since edits can apply to discontiguous sequences of text, which pose a challenge for sequence modeling approaches, we explore novel ways to represent a document both before and after an edit, while also accounting for contextual information surrounding an edit. To differentiate the context from edit words, we also explore a novel mechanism to encode edit operations such as additions and deletions explicitly in the model. [1]

Finally, we evaluate our model on both tasks and in a number of experimental settings, demonstrating that our solution is significantly better than several strong baselines on jointly capturing the relationship between edits and comments. Our model outperforms the best baseline by 34.6% on NDCG for Comment Ranking and achieves a best score of 0.687 F1 for Edit Anchoring. Additionally, in an ablation study we demonstrate that our various modeling choices, which tackle the inherent challenges of comment-edit understanding, each contribute positively to empirical results.

[1]We are making the code for the CmntEdit model, and generation of our dataset publicly available to the research community at https://github.com/microsoft/WikiCommentEdit.

| Comment | # Sutskever missing |
|---|---|
| Pre-edit Version | In October 2012, a similar system by Krizhevsky and Hinton won the large-scale ImageNet competition by a significant margin over shallow... |
| Post-edit Version | In October 2012, a similar system by Krizhevsky and **Sutskever and** Hinton won the large-scale ImageNet competition by a significant margin over shallow... |

Table 1: Example of an edit and its associated comment. The added words "Sutskever and" in post-edit version is marked in red.

## 2 Related Work

Document revisions have been the subject of several studies in recent years (Nunes et al., 2011; Fischer, 2013). Most prior work focuses on modeling document edits only. For instance, Bronner and Monz (2012) build a classifier to distinguish fluency edits from factual edits. Zhu et al. (2017) study the semantic distance between the content in different versions of documents to detect document revisions. Grossman et al. (2013) propose a hierarchical navigation method to display document revision histories.

Some work utilizes comments associated with document edits as supplementary information to study the document revisions. For example, Yatskar et al. (2010) consider both comment and document revision for lexical simplification. However, they use comments as meta-data to identify trusted revisions, rather than directly modeling the relationship between comments and edits. Yang et al. (2017) featurize both comments and revisions to classify edit intent, but without explicitly modeling edit-comment relationship.

Wikipedia revision history data (Nunes et al., 2008) has been used in many NLP tasks (Zesch, 2012; Max and Wisniewski, 2010; Ganter and Strube, 2009). For instance, Yamangil and Nelken (2008) model Wikipedia revision histories for improving sentence compression, Aji et al. (2010) propose a new term weighting model leveraging Wikipedia revision histories, and Zanzotto and Pennacchiotti (2010) expand textual entailment corpora from Wikipedia revision histories using co-training. Again, however, none of these meth-

ods directly consider or model the relationship between comments and edits.

At a basic level, modeling the connection between comments and edits can be seen as a text matching problem, with superficial similarity to other common NLP tasks, such as Question Answering (Seo et al., 2016; Yu et al., 2018), document search (Burges et al., 2005; Nalisnick et al., 2016), and textual entailment (Androutsopoulos and Malakasiotis, 2010), among others. Note however, that edits are a (possibly discontinuous and non-trivial) delta between two versions of a text, making their representation and understanding more challenging than that of a simple string. We demonstrate this in our evaluation in Section 5.2, where we compare against several competing models that were designed for other text matching challenges.

## 3 Dataset

Our dataset – which we call *WikiCmnt* – is generated from Wikipedia revision histories. In the absence of publicly available document data, Wikipedia is a particularly rich resource: (i) It maintains full revision histories of every Wikipedia page, along with associated editor comments as meta-data. (ii) It is a large-scale instance of multi-author collaboration, with many editors contributing to and maintaining pages.

The specific historical dump we use is from May 1, 2018. It contains approximately 52.7 million pages, and 755.5 million unique revisions made by 300.8 million users. WikiCmnt is a subsample of 786,866 Wikipedia page revisions along with their associated metadata. Revisions are filtered out before sampling, if they violate any one of the following criteria: (i) The length of the comment is longer than 8 words. (ii) The edits made to the Wikipedia page span more than one section[2]. (iii) The Wikipedia page has an edit history containing fewer than 10 unique revisions.

We extract and store a number of data fields from the Wikipedia revision history as summarized in Table 2. For each specific revision of a page, we not only retrieve the text of the comment and edit but also sample 10 non-related comments (*Neg-Cmnts*) and 10 non-related edits (*Neg-Edits*) from the same page's history. Finally we also encode the individual edit operations in both pre-edit and post-edit versions of a text. For example, consider Figure 1, which shows the edit action encoding associated with an change from "chicken wing" to "salmon".

[2] https://en.wikipedia.org/wiki/Help:Section

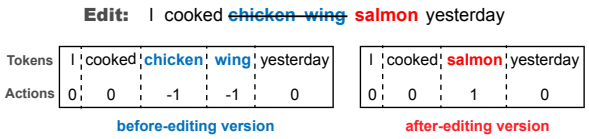


Figure 1: An example of the action encoding associated with changing the phrase "chicken wing" to "salmon". Specifically, the values -1, 1 and 0 are used to represent deletions, additions and unchanged tokens, respectively.

| Field | Description |
|---|---|
| Revision ID | Wiki revision ID |
| Parent ID | Parent revision ID |
| Timestamp | Timestamp of revision |
| Page Title | Title of Wikipedia page |
| Comment | Revision comment |
| Neg-Cmnts | Negative sampled comments |
| Src-Tokens | Tokens of pre-editing document |
| Src-Actions | Action encoding of pre-editing document |
| Tgt-Tokens | Tokens of post-editing document |
| Tgt-Actions | Action encoding of post-editing document |
| Pos-Edits | Edited sentences in post-editing document |
| Neg-Edits | Negative sampled sentences in post-editing document |

Table 2: Data Fields in *WikiCmnt* Dataset

## 4 Proposed Model

To model the relationship between edits and comments, we first formulate the problem with respect to the two tasks of Comment Ranking and Edit Anchoring; we then provide details of the neural architecture used to solve these problems; finally we provide some implementation details. We begin with preliminary notation.

Let us define $c$ as a comment consisting of word tokens $\{w_1, ..., w_q\}$. Minimally, let us also define a pre-edit $e_s$, as the contiguous sequence of words spanning the first to the last edited token (with possibly intervening tokens that are not edited) in a document's *pre-edit version*. The edit $e_s$ may optionally also contain some surrounding context. Formally this can be defined as:

$$\{\underbrace{\mathbf{w}_{i-k}, ...\mathbf{w}_{i-1}}_{\text{context before edit}}, \underbrace{\mathbf{w}_i, ...\mathbf{w}_{i+p}}_{\text{edit words}}, \underbrace{\mathbf{w}_{i+p+1}, ...\mathbf{w}_{i+p+k}}_{\text{context after edit}}\}$$

where $i$ and $i+p$ are the indices of the first and the last edited tokens in a revision, and $k$ is the

context window size. If there is more than one contiguous block of edited tokens, these blocks are concatenated to form the set of edit words with their context words.

We define a document edit as the pair $e = \{e_s, e_t\}$, where $e_t$ is similarly defined over edited tokens and their context words in a document's *post-edit version*.

### 4.1 Problem Formulation

**Comment Ranking** is the task of finding the most relevant comment among a list of potential candidates, given a document edit. The inputs of the comment ranking task are some set of user comments $C = \{c_1, c_2, \ldots c_m\}$ pertaining to a document, and an edit $e = \{e_s, e_t\}$ in the same document.The goal of the task is to produce a ranking on $C$, such that the true comment $c_i$ with respect to the edit $e = \{e_s, e_t\}$ is ranked above all the other comments (i.e. distractors). We use standard ranking metrics to evaluate model performance: Precision@K (P@K), Mean Reciprocal Rank (MRR) and Normalized Discounted Cumulative Gain (NDCG).

**Edit Anchoring** is the task of finding the sentences in a document that most likely underwent change, given a specific user comment. The inputs to the task are a user comment $c$ and a list of candidate sentences $S = \{s_1, s_2, \ldots s_n\}$ in the post-edit document $e_t$. Unlike with Comment Ranking, we operate under the assumption that an edit has already been completed, and therefore discard the information from the pre-edit version $e_s$. In the ground truth, at least one (but possibly more) of the sentences is an edit location for the comment $c$. The expected output is a list of binary classifications $R = \{r_i\}_{i=1}^{n}$, where $r_i = 1$ indicates that the sentence $s_i$ is a likely edit location, given comment $c$. We use Accuracy and F1 score to evaluate performance on this task.

### 4.2 Model Overview

For both tasks, our models are hierarchical deep neural networks with four layers: an input embedding layer, a contextual embedding layer, a comment-edit attention layer, and an output layer. The overall architecture is shown in Figure 2. We describe each of the four layers in what follows. Since both models share many components we will describe the more general case that covers all inputs for Comment Ranking; for Edit Anchoring

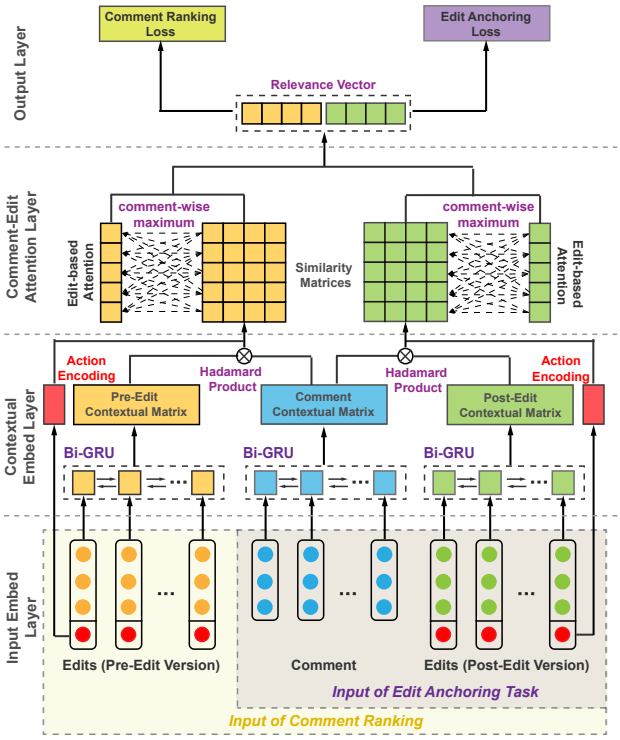


Figure 2: Overall Architecture of Proposed Model

those components that correspond to the pre-edit input are suitably omitted.

**Input Embedding Layer.** The input embedding layer maps each word in user comments $c$ and edits $e = \{e_s, e_t\}$ to a high-dimensional vector space. The output of the input embedding layer are matrices: $\boldsymbol{U} \in \mathbb{R}^{d \times M}$ representing the pre-edit document, $\boldsymbol{V} \in \mathbb{R}^{d \times M}$ representing the post-edit document, and $\boldsymbol{Q} \in \mathbb{R}^{d \times J}$ representing the comment. Here $M$ is the length of edits and $J$ is the length of the comment, while $d$ is the fixed-length dimension of word vectors.

**Contextual Embedding Layer.** We use a bi-directional Gated Recurrent Unit (GRU) (Chung et al., 2014) to model the sequential interaction between words. Operating over the output of the previous layer we obtain the contextual embedding matrices $\boldsymbol{U}^c \in \mathbb{R}^{2d \times M}$ and $\boldsymbol{V}^c \in \mathbb{R}^{2d \times M}$ for both pre- and post-edit versions. Also, we obtain $\boldsymbol{Q}^c \in \mathbb{R}^{2d \times J}$ from the comment word vectors. Note that the row dimension of contextual matrices $\boldsymbol{U}^c$, $\boldsymbol{V}^c$ and $\boldsymbol{Q}^c$ are $2d$ because of the concatenation of the GRU's output in both forward and backward directions.

**Comment-Edit Attention Layer.** Inspired by the attention mechanisms utilized in machine

comprehension (Seo et al., 2016; Yu et al., 2018), the comment-edit attention layer is designed to capture relationships between document edit and comment words. The attention layer maintains and processes both pre- and post-edit documents separately. This is to reduce the information loss that would have occurred if their representations were fused before the attention layer. Additionally, this layer incorporates an action encoding vector, which is designed to reflect the three kinds of edit operations: adding a word, deleting a word, or leaving it unchanged.

The inputs to the layer are the contextual matrices $\boldsymbol{U}^c$ and $\boldsymbol{V}^c$ of the pre- and post-edit documents respectively, the matrix $\boldsymbol{Q}^c$ representing the comment, and the supplemental action encoding vectors $\boldsymbol{a}^\dagger, \boldsymbol{a}^\ddagger \in \mathbb{Z}^M$ which encode the edit operation each token undergoes in the pre- and post-edit documents, respectively. The output is the comment-aware concatenated vector representations of the edit words in both pre- and post-edit documents, $\boldsymbol{h} \in \mathbb{R}^{2J}$.

Internally, we first calculate the shared similarity matrix $\boldsymbol{S}^\dagger \in \mathbb{R}^{M \times J}$ between the comment $\boldsymbol{Q}^c$ and contextual matrix $\boldsymbol{U}^c$ of pre-edit documents, while also accounting for the action encoding vector $\boldsymbol{a}^\dagger$. The elements of this shared similarity matrix are defined as follows:

$$\boldsymbol{S}^\dagger_{ij} = \mathcal{G}(\boldsymbol{U}^c_{:i}, \boldsymbol{Q}^c_{:j}, \boldsymbol{a}^\dagger_i) \tag{1}$$

where $\mathcal{G}$ is a trainable function that generates the similarity between the word-level representations of comments and edits with respect to an edit operation.

Here $\boldsymbol{U}^c_{:i} \in \mathbb{R}^{2d \times 1}$ is the vector representation of the $i$-th word in the pre-edit document and $\boldsymbol{Q}^c_{:j} \in \mathbb{R}^{2d \times 1}$ is the vector representation of $j$-th word in the comment. $\boldsymbol{a}^\dagger_i \in \{-1, 0, 1\}$ is the action encoding for the edit operation performed on the $i$-th word in the pre-edit document. We choose the trainable function $\mathcal{G}(\boldsymbol{u}, \boldsymbol{q}, a) = \boldsymbol{w}^T[\boldsymbol{u} \otimes \boldsymbol{q}; a]$, where $\boldsymbol{w} \in \mathbb{R}^{(2d+1) \times 1}$ is a trainable weight vector, $\otimes$ is the element-wise multiplication operator and $[;]$ represents the vector concatenation across the row dimension. Similarly, we can calculate the shared similarity matrix $\boldsymbol{S}^\ddagger \in \mathbb{R}^{M \times J}$ between the comment and contextual matrix of the post-edit document as $\boldsymbol{S}^\ddagger_{ij} = \mathcal{G}(\boldsymbol{V}^c_{:i}, \boldsymbol{Q}^c_{:j}, \boldsymbol{a}^\ddagger_i)$. Note that the weight vectors in function $\mathcal{G}$ are different for pre- and post-edit versions, and both are trained simultaneously in the model.

After the similarity matrices are computed, we use them to generate the Comment-to-Edit Attention (C2E) weights. C2E is used to represent the relevance of words in the edit to those that appear in the comment. This is critical for modeling the relationship between comments and edits. We obtain the C2E attention weights $\boldsymbol{c}^\dagger \in \mathbb{R}^M$ for edit words in the pre-edit document by taking $\boldsymbol{c}^\dagger = \mathrm{softmax}(\max_{\mathrm{col}}(\boldsymbol{S}^\dagger))$, where the $\max_{\mathrm{col}}(\cdot)$ operator finds the column-wise maximum value from a matrix. Similarly, for the post-edit document, we have $\boldsymbol{c}^\ddagger = \mathrm{softmax}(\max_{\mathrm{col}}(\boldsymbol{S}^\ddagger))$.

Finally we multiply the attention vectors to the previously computed similarity matrices for both pre- and post-edit documents, and concatenate the results to obtain the relevance vector $\boldsymbol{h} \in \mathbb{R}^{2J}$:

$$\boldsymbol{h} = \left[ \left(\boldsymbol{c}^\dagger\right)^T \boldsymbol{S}^\dagger \; ; \; \left(\boldsymbol{c}^\ddagger\right)^T \boldsymbol{S}^\ddagger \right]^T \tag{2}$$

The vector $\boldsymbol{h}$ intuitively captures the weighted sum of the relevance of the comment with respect to the edits in both pre- and post-edit documents.

**Output Layer and Loss Function.** The output layer and loss function of the network is task-specific. Comment Ranking requires ranking the relevance of candidate comments given a document edit. Broadly we obtain a ranked list by computing the relevance score between comments and edits by the output $r = \boldsymbol{\beta}^T \boldsymbol{h}$, where $\boldsymbol{\beta}$ is a trainable weight vector.

A data sample $i$ in Comment Ranking consists of one true comment-edit pair and $n_i$ negative comment-edit distractors. We denote $r^+_i$ as the relevance score of the true pair, and $r^-_{ij}$ as the relevance score of the $j$-th distractor pair (with $1 \leq j \leq n_i$). The goal of our task is to make $r^+_i > r^-_{ij}$ for all $j$. We therefore set the loss function to be a pairwise hinge loss between true and distractor relevance scores.

$$\mathcal{L}_c(\Theta) = \sum_{i=1}^{N} \sum_{j=1}^{n_i} \max(0, 1 - r^+_i + r^-_{ij}) \tag{3}$$

where $\Theta$ is the set of all trainable weights in the model and $N$ is the number of training samples in the dataset.

For Edit Anchoring, the goal is to predict whether a sentence in the document is likely to be the location of an edit, given a comment. This is a binary classification problem, and we can suitably set the output layer as $p = \mathrm{softmax}(\boldsymbol{\gamma}^T \boldsymbol{h})$ – the

probability of predicting the positive class. Here $\gamma$ is a trainable weight vector.

Given the binary nature of the classification problem we can use the cross entropy loss:

$$\mathcal{L}_e(\Phi) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{m_i}\Big[y_{ij}\log p_{ij} + \big(1-y_{ij}\big)\log\big(1-p_{ij}\big)\Big], \tag{4}$$

where $\Phi$ is the set of all trainable weights in the model, $N$ is the number of data samples, $m_i$ is the number of sentences in the $i$-th data sample, $p_{ij}$ is the predicted label of the $j$-th sentence in the $i$-th data sample, and $y_{ij}$ is the corresponding ground truth label.

### 4.3 Implementation Details

The CmntEdit model described in this section is implemented using the Pytorch[3] framework and trained on a single Tesla P100 GPU with 16GB memory. The word vectors are initialized with pre-trained Glove embeddings (Pennington et al., 2014) using the default dimensionality of 100. We set the number of training epochs to 5, the maximum length of a comment to 30 tokens and the maximum length of an edit to 300 tokens. For the Comment Ranking task, we set the batch size to 10 and consider 5 candidate comments in each data sample: one true comment and 4 distractors. We thus have 5 comment-edit pairs for each data sample and 50 pair-wise samples for each training batch. For the Edit Anchoring task, we also set the batch size to 10 and consider 5 candidate sentences, which yields an identical 50 training instances per batch.

It should be noted that while we train our model with only 5 candidate comments or sentences (for Comment Ranking or Edit Anchoring respectively), the models – once trained – can be applied to any number of candidates at test time for either task.

## 5 Experiment

In this section, we show evaluation results of our model on the previously collected Wikipedia dataset 3. We begin by introducing the experimental settings in Section 5.1. We then compare the performance achieved by the proposed method against several baseline models in Section 5.2. We also conduct an ablation study to evaluate the various components of our model, as well as provide some qualitative results to demonstrate it's effectiveness in practise.

[3] https://pytorch.org/

### 5.1 Experimental Setup

We begin by introducing the evaluation setting, metrics and baselines we use in our experiments.

#### 5.1.1 Datasets and Labels

We use the *WikiCmnt* dataset described in Section 3 for training and evaluation. The dataset contains 786,886 data samples in total. We reserve 70% of the data for training and split the remainder between 10% for validation and 20% for testing.

For the Comment Ranking task, we always have a single true comment, but in our test we experiment with both 4 and 9 distractors, yielding a total of 5 and 10 candidate comments. Similarly for the Edit Anchoring task, we also test with both 5 and 10 candidate sentences.

#### 5.1.2 Evaluation Metrics

We use common ranking evaluation metrics from the literature to evaluate models on the task of Comment Ranking. These include: (i) **Precision@K**. The proportion of predicted instances where the true comment appears in the ranked top-K result, with $K = 1, 3, 5$. (ii) **Mean Reciprocal Rank (MRR)**. The average multiplicative inverse of the rank of the correct answer, represented mathematically as $\mathrm{MRR} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\mathrm{rank}_i}$, where $N$ is the number of samples and $\mathrm{rank}_i$ is the rank assigned to the true comment by a model. (iii) **Normalized Discounted Cumulative Gain (NDCG)**. the normalized gain of each comment based on its ranking position in the results. We set the relevance score of the true comment to one and those of the distractors to zero.

For the Edit Anchoring task, we use **Accuracy** and **F1 Score** as evaluation metrics. These are computed based on a model's classification of candidate sentences in the post-edit version of the document.

#### 5.1.3 Baseline Methods

For the Comment Ranking task, we compare our approach to the following baselines:
(i) **Cosine-TfIdf** uses the cosine similarity between the TfIdf-weighted vectors of the comment and edit as a measure of relevance. (ii) **Cosine-InferSent** computes the cosine similarity between

| | Candidates=5 | | | | Candidates=10 | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | P@1 | P@3 | MRR | NDCG | P@1 | P@3 | P@5 | MRR | NDCG |
| Cosine-TfIdf | 0.266 | 0.519 | 0.470 | 0.597 | 0.137 | 0.291 | 0.444 | 0.296 | 0.453 |
| Cosine-InferSent | 0.228 | 0.597 | 0.471 | 0.600 | 0.115 | 0.305 | 0.497 | 0.302 | 0.461 |
| RankNet | 0.257 | 0.658 | 0.503 | 0.620 | 0.034 | 0.055 | 0.061 | 0.139 | 0.320 |
| LambdaMART | 0.310 | 0.726 | 0.549 | 0.661 | 0.152 | 0.384 | 0.593 | 0.352 | 0.502 |
| Gated RNN | 0.283 | 0.716 | 0.531 | 0.647 | 0.158 | 0.411 | 0.628 | 0.363 | 0.511 |
| CmntEdit-MT | 0.674 | 0.930 | 0.804 | 0.853 | 0.529 | 0.780 | 0.896 | 0.680 | 0.758 |
| CmntEdit-CR | **0.710** | **0.944** | **0.828** | **0.871** | **0.593** | **0.830** | **0.924** | **0.730** | **0.796** |

Table 3: Performance on Comment Ranking

comment and edit vectors generated by the state-of-the-art sentence embedding method InferSent (Conneau et al., 2017). (iii) **RankNet** (Burges et al., 2005) minimizes the number of inversions in ranking by optimizing a pair–wise loss function. We use a previous neural network implementation[4] with default settings. (iv) **LambdaMART** (Burges, 2010) leverages gradient boosted decision trees with a cost function derived from LambdaRank (Burges et al., 2007) for solving a ranking task. We use an existing python implementation[5], with a learning rate of 0.02 and 10 max leaf nodes. (v) **Gated Recurrent Neural Network** (Chung et al., 2014) models the sequence of words in comments and edits using pre-trained GloVe vectors as embedding units. Three fully-connected layers compute a final relevance score between comments and edits.

For the Edit Anchoring task, we chose the following popular classifiers as baselines: (i) **Random Forest** (Liaw et al., 2002) (ii) **Adaboost** (Rätsch et al., 2001) (iii) **Passive Aggressive** classifiers (Crammer et al., 2006) (iv) **Gated Recurrent Neural Network**. The features used in these models are based on both TfIdf scores, as well as sentence embedding features generated by InferSent. The Gated RNN model is trained with a task-specific fully connect layer for Edit Anchoring to compute the classification probability.

#### 5.1.4 Model Variants

We train and evaluate several different variants of our neural architecture. They include: (i) **CmntEdit-CR**, a variant of our model only trained with data samples for the Comment Ranking task. (ii) **CmntEdit-EA**, a variant of our model only trained with data samples for the Edit Anchoring task. (iii) **CmntEdit-MT**, a variant of our model jointy trained on data samples of both Comment Ranking and Edit Anchoring tasks. Unless otherwise specified, all models use a standard context window size of 5 tokens[6].

[4] https://github.com/shiba24/learning2rank

[5] https://github.com/jma127/pyltr

| | Candidates=5 | | Candidates=10 | |
|---|---|---|---|---|
| | Acc | F1 | Acc | F1 |
| Passive-Aggr | 0.581 | 0.533 | 0.716 | 0.262 |
| RandForest | 0.639 | 0.290 | 0.743 | 0.112 |
| Adaboost | 0.657 | 0.398 | **0.751** | 0.207 |
| Gated RNN | 0.696 | 0.651 | 0.665 | 0.539 |
| CmntEdit-MT | 0.635 | 0.587 | 0.619 | 0.468 |
| CmntEdit-EA | **0.744** | **0.687** | 0.726 | **0.583** |

Table 4: Performance on Edit Anchoring

### 5.2 Experimental Results

We now present and discuss the empirical results of our evaluation on both Comment Ranking and Edit Anchoring.

#### 5.2.1 Comment Ranking

Table 3 summarizes results of the Comment Ranking task. Our models significantly outperform all the baselines in every setting and on all metrics. The results are statistically significant at $p < 0.01$ using the Wilcoxon signed-rank test (Smucker et al., 2007). Since the Comment Ranking task only has one true comment, the other comments being distractors, the P@1 result becomes especially important for practical applications. Our approach achieves 71% precision, which is significantly better than the 31% precision of the best baseline method (LambdaMART) with 5 candidate comments. Our model similarly outperforms the best baseline with 10 candidate comments, ob-

[6] We performed an extensive evaluation of context size with window sizes ranging from 0 to 50. From the result, we found the our model tends to perform best when the context window size is close to 5 tokens.

| **Comment Ranking** | Candidates=5 | | Candidates=10 | |
|---|---|---|---|---|
| | P@1 | MRR | P@1 | MRR |
| w/o Action | 0.630 | 0.778 | 0.495 | 0.659 |
| w/o Attention | 0.625 | 0.775 | 0.488 | 0.655 |
| w/o Hadamard | 0.624 | 0.774 | 0.483 | 0.652 |
| CmntEdit-CR | **0.710** | **0.828** | **0.593** | **0.730** |
| **Edit Anchoring** | Candidates=5 | | Candidates=10 | |
| | Acc | F1 | Acc | F1 |
| w/o Action | 0.713 | 0.668 | 0.684 | 0.556 |
| w/o Attention | 0.723 | 0.670 | 0.701 | 0.562 |
| w/o Hadamard | 0.722 | 0.660 | 0.706 | 0.558 |
| CmntEdit-EA | **0.744** | **0.687** | **0.726** | **0.583** |

Table 5: Ablation study

taining a P@1 score of 59.3%. Additionally, the higher scores on MRR and NDCG indicate that our approach consistently ranks the true comment higher than the baseline methods.

The performance of CmntEdit-MT is 2% and 5.3% worse than CmntEdit-CR on NDCG and P@1, respectively. Surprisingly, this suggests that training our model in a multi-task fashion leads to slightly lower scores, and a model specifically trained for the individual task of Comment Ranking is to be preferred.

#### 5.2.2 Edit Anchoring

Table 4 shows the results for Edit Anchoring. Our method, CmntEdit-EA, outperforms the best baseline method, Gated-RNN, by 5.5% on F1 and 6.9% on accuracy. The improvements over all the baselines are statistically significant at a p-value of 0.01. The baseline classifiers including Passive-Aggressive, Random Forest and Adaboost have high accuracies, but low F1 scores. This is because of the imbalance between positive and negative samples in our data. Specifically, the number of negative samples is 4 times greater than the number of positive samples when the size of the candidate set is 5 – and even greater when it is 10. Therefore, the baseline classifiers tend to naively predict a negative label, which artificially boosts precision to the detriment of recall. In fact, Adaboost actually outperforms our models on accuracy when the candidate set size is 10, but yields a much lower F1 score.

As with Comment Ranking, the performance of CmntEdit-MT is slightly worse than CmntEdit-EA. Within the scope of our problem space, this reinforces the finding that targeted training seems to work better than joint training.

#### 5.2.3 Ablation Study

To verify the effectiveness of our modeling choices, we evaluate performance in the absence of each of the following model components: 1. **w/o Action**: we remove the action encoding from the trainable function $\mathcal{G}$ and instead simply use $\mathcal{G}(\boldsymbol{u}, \boldsymbol{q}) = \boldsymbol{w}^T[\boldsymbol{u}^T\boldsymbol{q}]$. 2. **w/o Attention**: we remove the edit-based attention from Equation (2). Instead, we generate the relevance vector $\boldsymbol{h}$ as follows: $\boldsymbol{h} = \left[\text{mean}_{\text{col}}(\boldsymbol{S}^{\dagger}); \text{mean}_{\text{col}}(\boldsymbol{S}^{\ddagger})\right]^T$. 3. **w/o Hadamard**: we use the inner product instead of the Hadamard product in the trainable function $\mathcal{G}$ as follows: $\mathcal{G}(\boldsymbol{u}, \boldsymbol{q}, a) = \boldsymbol{w}^T[\boldsymbol{u}^T\boldsymbol{q}; a]$.

Results in Table 5 show that each component improves the overall performance on both Comment Ranking and Edit Anchoring tasks, across our evaluation metrics. This indicates that our modeling choices are particularly suited to tackle the inherent challenges involved in modeling comment-edit relationship.

#### 5.2.4 Qualitative Evaluation

Table 6 provides a few output examples from our model on the Comment Ranking task, demonstrating its ability to learn abstract connections between comments and edits. Due to space constraints, only one illustrative distractor is shown.

The first example shows an edit summarized by the high-level comment "date and capitalization corrections". This comment is correctly assigned the highest relevance score by our model, despite the fact that no words are shared between the comment and edit. Meanwhile, one of the distractors has a lower score even though it shares the lexical item "Walgreens" with the context of the edit.

In the second example an entire sentence is removed by the editor. Again, although no words are shared between the comment and the edit, our model is correctly able to identify the delete operation, possibly by learning the common Wikipedia shorthand for deletions "rm". Meanwhile, one of the distractors contains the phrase "St Helens", which also appears in the edit, but is still assigned a lower score.

## 6 Conclusion and Future Work

In this paper, we have explored the relationship between comments and edits by defining two novel tasks: Comment Ranking and Editing Anchoring. In order to model the problem we collected a dataset with over 780K comment-edit pairs. Fur-

| | |
|---|---|
| **Case 1: Grammar fixing** | |
| Pre-Edits | In December **of** 2012, **A** judge ordered Walgreens to pay $16.57 million to settle a lawsuit claiming... |
| Post-Edits | In December 2012, **a** judge ordered Walgreens to pay $16.57 million to settle a lawsuit claiming... |
| Comment | Date and capitalization corrections [Relevance Score: 4.050] |
| Distractor | Dane Cook's comments about Walgreens do not belong in "Facts" [Relevance Score: 3.028] |
| **Case 2: Sentence removal** | |
| Pre-Edits | **[http://www.pilkington.co.uk/energikare Pilkington energiKare] - Pilkington product from St Helens helping the environment**... |
| Post-Edits | (whole sentence removed) |
| Comment | rm advertising, other link to major employer noted in text seems ok at first glance. [Relevance Score: 5.192] |
| Distractor | Fixing link to Kevin Brown - You know I never knew he was from St Helens [Relevance Score: 3.479] |

Table 6: Example of the edits and comments matched by the proposed model with one more deceptive distractor for each case. The scores for each comment/distractor are presented after the comment/distractor. The addition and removal in edit are highlighted in bold.

ther we proposed a hierarchical multi-layer neural network capable of tackling both our proposed tasks by encoding specific edit actions, such as additions and deletions, as well as document context. In our experiments we show that our approach outperforms several baselines by significant margins on both tasks, yielding a best score of 71% precision@1 for Comment Ranking and 74.4% accuracy for Edit Anchoring.

In future work we plan to explore sequences of revisions through the lifecycle of a document from creation to completion, with the ultimate goal of modeling document evolution. We also hope to apply our modeling approach to practical downstream applications, including: i) detecting completed to-dos based on related edits; ii) localizing the paragraphs that could be edited to address a given comments; iii) summarizing document revisions.

## Acknowledgement

We thank the anonymous reviewers and members of the Knowledge Technologies and Intelligent Experiences group in Microsoft Research for helpful feedback. We also thank Ryen W. White for the thoughtful feedback on the statistical hypothesis testing.

## References

Ablimit Aji, Yu Wang, Eugene Agichtein, and Evgeniy Gabrilovich. 2010. Using the past to score the present: Extending term weighting models through revision history analysis. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 629–638. ACM.

Ion Androutsopoulos and Prodromos Malakasiotis. 2010. A survey of paraphrasing and textual entailment methods. *Journal of Artificial Intelligence Research*, 38:135–187.

Amit Bronner and Christof Monz. 2012. User edits classification using document revision histories. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, pages 356–366, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96. ACM.

Christopher J Burges, Robert Ragno, and Quoc V Le. 2007. Learning to rank with nonsmooth cost functions. In *Advances in neural information processing systems*, pages 193–200.

Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11(23-581):81.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7(Mar):551–585.

Stephen E Fischer. 2013. Automated document revision markup and change control. US Patent 8,381,095.

Viola Ganter and Michael Strube. 2009. Finding hedges by chasing weasels: Hedge detection using wikipedia tags and shallow linguistic features. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 173–176. Association for Computational Linguistics.

Tovi Grossman, Justin Frank Matejka, and George Fitzmaurice. 2013. Hierarchical display and navigation of document revision histories. US Patent 8,533,593.

Andy Liaw, Matthew Wiener, et al. 2002. Classification and regression by randomforest. *R news*, 2(3):18–22.

Aurélien Max and Guillaume Wisniewski. 2010. Mining naturally-occurring corrections and paraphrases from wikipedia's revision history. In *LREC*.

Eric Nalisnick, Bhaskar Mitra, Nick Craswell, and Rich Caruana. 2016. Improving document ranking with dual word embeddings. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 83–84. International World Wide Web Conferences Steering Committee.

Sérgio Nunes, Cristina Ribeiro, and Gabriel David. 2008. Wikichanges: exposing wikipedia revision activity. In *Proceedings of the 4th International Symposium on Wikis*, page 25. ACM.

Sérgio Nunes, Cristina Ribeiro, and Gabriel David. 2011. Term weighting based on document revision history. *Journal of the American Society for Information Science and Technology*, 62(12):2471–2478.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Gunnar Rätsch, Takashi Onoda, and K-R Müller. 2001. Soft margins for adaboost. *Machine learning*, 42(3):287–320.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Robert Shaver. 2016. Document comment management. US Patent 9,418,054.

Mark D. Smucker, James Allan, and Ben Carterette. 2007. A comparison of statistical significance tests for information retrieval evaluation. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pages 623–632, New York, NY, USA. ACM.

Elif Yamangil and Rani Nelken. 2008. Mining wikipedia revision histories for improving sentence compression. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, pages 137–140. Association for Computational Linguistics.

Diyi Yang, Aaron Halfaker, Robert Kraut, and Eduard Hovy. 2017. Identifying semantic edit intentions from revisions in wikipedia. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2000–2010.

Mark Yatskar, Bo Pang, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2010. For the sake of simplicity: Unsupervised extraction of lexical simplifications from wikipedia. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 365–368. Association for Computational Linguistics.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*.

Fabio Massimo Zanzotto and Marco Pennacchiotti. 2010. Expanding textual entailment corpora fromwikipedia using co-training. In *Proceedings of the 2nd Workshop on The People's Web Meets NLP: Collaboratively Constructed Semantic Resources*, pages 28–36.

Torsten Zesch. 2012. Measuring contextual fitness using error contexts extracted from the wikipedia revision history. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 529–538. Association for Computational Linguistics.

Xiaofeng Zhu, Diego Klabjan, and Patrick N. Bless. 2017. Semantic document distance measures and unsupervised document revision detection. In *IJCNLP*.

# PRADO: Projection Attention Networks for Document Classification On-Device

**Prabhu Kaliamoorthi**
Google Research
Mountain View, CA, USA
prabhumk@google.com

**Sujith Ravi**
Google Research
Mountain View, CA, USA
sravi@google.com

**Zornitsa Kozareva**
Google
Mountain View, CA, USA
zornitsa@kozareva.com

## Abstract

Recently, there has been a great interest in the development of small and accurate neural networks that run entirely on devices such as mobile phones, smart watches and IoT. This enables user privacy, consistent user experience and low latency. Although a wide range of applications have been targeted from wake word detection to short text classification, yet there are no on-device networks for long text classification.

We propose a novel projection attention neural network PRADO that combines trainable projections with attention and convolutions. We evaluate our approach on multiple large document text classification tasks. Our results show the effectiveness of the trainable projection model in finding semantically similar phrases and reaching high performance while maintaining compact size. Using this approach, we train tiny neural networks just 200 Kilobytes in size that improve over prior CNN and LSTM models and achieve near state of the art performance on multiple long document classification tasks. We also apply our model for transfer learning, show its robustness and ability to further improve the performance in limited data scenarios.

## 1 Introduction

One of the fundamental tasks in Natural Language Processing is related to long text classification. Given a document, the goal is to assign one or more categories of interest to the text. This task is of high importance as it has wide applications in spam detection (Jindal and Liu, 2007), product categorization (Kozareva, 2015), sentiment classification (Pang and Lee, 2008) and it also plays an important role for improving document retrieval and ranking (Deerwester et al., 1990).

For a long time, the most successful text classification approaches relied on sparse lexical features such as n-grams, which are later used by linear or kernel models (Joachims, 1998; McCallum and Nigam, 1998; Joulin et al., 2016). However, with the recent advancements in deep learning, various neural network architectures like CNN (Kim, 2014), LSTM (Zhang et al., 2015), hierarchical attention mechanisms (Yang et al., 2016) showed improvement in performance.

Recently, (Ravi and Kozareva, 2018) and (Ravi and Kozareva, 2019) showed the importance of building on-device neural models for short text classification, which preserve user privacy, enable consistent user experience and most importantly perform inference on the device. One of the biggest challenges is how to fit these large and complex neural networks on devices with limited memory and computation capacity while still maintaining high performance. (Ravi and Kozareva, 2018, 2019) developed on-device self-governing neural networks (SGNN) and (SGNN++) based on locality-sensitive projections (Ravi, 2017, 2019). Those methods were evaluated on short text classification tasks such as dialog act and user intent understanding and outperformed prior RNN work (Khanpour et al., 2016; Ortega and Vu, 2017).

In this work, we take one step further by proposing a novel projection attention neural network called PRADO . Unlike SGNN which has static projections, PRADO combines trainable projections with attention and convolutions allowing it to capture long range dependencies and making it a powerful and flexible approach for long text classification. We study the impact of different hyperparameters on accuracy vs model size. We also address the problem of producing compact architectures by develop a quantized version of PRADO . In a series of experimental evaluations on multiple long text classification tasks, we show that our approach PRADO improves over prior baselines and



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5012–5021,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

neural networks such as character and word level CNNs and LSTMs. The main contributions of our work are as follows:

- Novel on-device projection attention neural network PRADO which combines transferable projections with attention and convolution for long text classification.
- Exhaustive experimental evaluation on multiple long text classification tasks, outperforming traditional feature engineered linear classifiers and deep learning approaches like CNN and LSTM.
- Quantized PRADO network, which results in tiny 200 Kilobytes in size model that improves over prior CNN and LSTM models.
- Applicability of PRADO for transfer learning, showed its robustness and ability to further improve performance in limited data scenarios.

## 2 Related Work

Early work on text classification relied on sparse lexical features such as n-grams and linear classifiers (Joachims, 1998; McCallum and Nigam, 1998; Joulin et al., 2016). But with the raise of deep learning, various CNN and LSTM approaches lead to significant improvement in performance and reaching state-of-the-art results. (Kim, 2014) used CNN architecture from computer vision for text classification. (Johnson and Zhang, 2015), used high-dimensional one hot vector and later introduced character-level CNN that achieved even more competitive results. (Tai et al., 2015) used tree structured LSTM for classification, while (Tang et al., 2015) use CNN or LSTM to capture sentence vector followed by bidirectional gated recurrent network which composes the vectors to get a document vector. Recently, (Yang et al., 2016) introduced hierarchical attention neural networks, which captures document representation by incorporating knowledge of the document structure into the model. This approach reaches the best performance on large set of text classification tasks.

The aforementioned prior work mostly focuses on building the best neural network model independent of any model size or memory constrains. However, recent work by (Ravi and Kozareva, 2018, 2019) show the importance of building on-device text classification models that can preserve user privacy, provide consistent user experience and most importantly are compact in size, while yet achieving state-of-art results. Previously, to build lightweight text classification approaches (Ravi, 2013) proposed fast sampling techniques, while (Bui et al., 2018) incorporated deep neural networks with graph learning. While successful, such approaches resulted in large models for response completion (Pang and Ravi, 2012) and Smart Reply (Kannan et al., 2016).

To address the challenge of fitting huge deep neural network on-device, (Ravi and Kozareva, 2018) developed a novel self-governing neural networks (SGNNs) that learns projections on the fly leading to small models. SGNN was applied on short text classification tasks such as dialog act and user intent understanding and showed significant improvement over state-of-the-art RNN (Khanpour et al., 2016) and RNN with attention (Ortega and Vu, 2017) approaches. In this work, we take one step further by developing trainable projection network with attention mechanism that captures long range dependencies making it a powerful and flexible approach for long text classification. In addition, we address the problem of producing compact architectures for text classification when we have limited amount of memory. FastText (Joulin et al., 2016) proposed product quantization to store word embeddings and have carried out evaluation which show models that require two orders of magnitude less memory. We use quantization techniques and show that we achieve 10x to 100x compression rate while still maintaining high performance and improving upon prior CNN and LSTM work. Unlike prior on-device text classification work, we also apply our model in a transfer learning scenario, which demonstrated the robustness of our approach and ability to further improve performance in limited data scenarios. Next, we describe the technical details of our approach, followed by experimental evaluation and results.

## 3 PRADO: Projection Attention Network

Figure 1 shows the overall architecture of our proposed network PRADO . It consists of a projected embedding layer, a convolutional and attention encoder mechanism and a final classification layer. We describe each component in detail below and contrast them with existing methods.

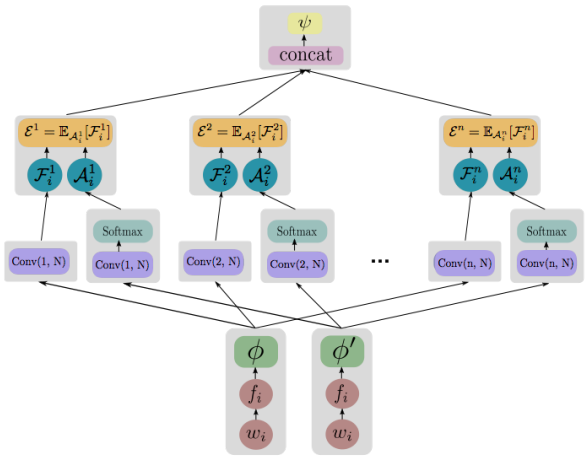


Figure 1: PRADO Model Architecture

### 3.1 Projected Embedding Layer

Let us assume that the input text has $T$ tokens or words. $w_i$ represents the $i$-th word, where $i \in \{0, ...T-1\}$. If $V$ is the number of words in the vocabulary, including an out of vocabulary token that represents all missing words, then each word $w_i$ is mapped to $\delta_i \in \mathbb{R}^V$. The first component in most neural networks designed for language tasks uses an embedding layer with trainable parameters $W \in \mathbb{R}^{d \cdot V}$ to map words to fixed length $d$-dimensional vectors $e_i = W\delta_i$, where $e_i \in \mathbb{R}^d$ are the word vectors that are processed by the rest of the network. A large fraction of the parameters in the network is concentrated in $W$, since $V$ often has to be large (upto hundreds of thousands or millions of words) to obtain good performance. Furthermore, by choosing $V$ upfront we are assuming that words or phrases relevant for the classification task are known apriori, which may not be true. It should be noted that though we express the operation to obtain the word vector $e_i$ as a matrix multiplication, in reality it is a look-up of the corresponding row in the embedding matrix as $\delta_i$ is modeled using the Dirac delta function.

**Embeddings via Trainable Projections:** Our approach PRADO replaces this embedding with a projection approach to build the word encoder. Instead of mapping $w_i$ to $\delta_i$, we map it to $f_i$ using a projection operator $\mathbb{P}$. Recent work (Ravi, 2017; Ravi and Kozareva, 2018; Ravi, 2019) has shown that projection-based neural approaches can help train compact neural networks that achieve good performance on certain language tasks. These networks learn robust representations (Sankar et al., 2019a) that can be also transferred to other tasks (Sankar et al., 2019b). We follow a similar strategy but unlike the static projections used in these works, we propose a new type of projection that decomposes the operation and makes the projection *trainable*, leading to more powerful encoders capable of capturing contextual information for long-text classification while maintaining a very low memory footprint. Our method does not rely on a fixed vocabulary.

The projection operator we use in this work first fingerprints the words and extracts $B$ bit features from the fingerprint. The word vectors $e_i$ are obtained using a neural network layer $e_i = \phi(f_i)$. This allows us to generate *compact* embeddings, *train the projection encoder* layer better and apply further optimizations like batch normalization (Ioffe and Szegedy, 2015) in this step. First, individual tokens $w_i$ in the input text are fingerprinted using a hashing function to generate $2B$ bits. The projection operator $\mathbb{P}$ then maps every consecutive two-bit sequence to the set $\{-1, 0, 1\}$ resulting in a vector $f_i \in \{-1, 0, 1\}^B$. We chose this specific form since it would enable further optimization such as those described in (Li and Liu, 2016) to reduce the computation in the first layer. We note that there could be alternative modeling choices for the specific form of the projection operator $\mathbb{P}$. Any projection operator that maps bits from the fingerprint to a bounded range is expected to perform equally well. $\phi(.)$ is a trainable function with $B \cdot d$ parameters that maps $B$-dimensional projection features into $d$-dimensional word embedding vectors that are computed dynamically during training and inference. In practice, $B \in [128, 512]$ and $d \in [32, 96]$ are tiny compared to $V$.

### 3.2 Convolution & Attention over Projections

Next, we introduce an encoder mechanism to map a sequence of projected word embeddings $e_0...e_{T-1}$ to a fixed length vector that represents the entire input text. There are many studies that use 1d convolutions on the word vectors and perform pooling to reduce the sequence to a fixed length vector. But we observe that most words or tokens in a sentence are not relevant for any classification problem, as a result methods like average pooling after convolutions do not effectively reduce the most relevant information needed for the task, especially when the text contains several sen-

tences. Other pooling methods like max or min do not let gradients flow effectively during backpropagation, making it difficult to train the network. To overcome this, we propose a method that uses convolutions and attention mechanism over the word projections and generate a fixed length encoding for the input text.

**Projected Attention:** In our approach, we use two independent convolutional networks for this step. First one, which we refer to as the *projected feature* network $\mathcal{F}$ captures the features that are useful for the classification task. This is comparable to the convolution networks used in existing studies except we perform convolutions over the sequence of projected word embeddings $e_i$.

$$\mathcal{F}_i^n = Conv(e_i, n, N) \tag{1}$$

where $n$ is the convolution kernel width, $N$ is the number of output channels in the convolution and $\mathcal{F}_i^n \in \mathbb{R}^N$. The second one, which we refer to as *attention network* $\mathcal{A}$, captures the importance of these features for the task.

$$\mathcal{W}_i^n = Conv(e_i, n, N) \tag{2}$$

We compute softmax over the sequence dimension of the results of $\mathcal{A}$. This provides a distribution over the word sequence that captures the relevance of features at different positions.

$$\mathcal{A}_i^n = \frac{e^{W_i^n}}{\sum_i e^{W_i^n}} \tag{3}$$

We compute an expectation using distribution $\mathcal{A}$ on $\mathcal{F}$ that turns the sequence into a fixed length encoding $\mathcal{E}^n$ for convolution kernel $n$.

$$\mathcal{E}^n = \sum_i \mathcal{A}_i^n \mathcal{F}_i^n \tag{4}$$

Our pooling scheme reduces to average pooling if the $\mathcal{A}_i^n$ is uniform over the sequence dimension and it becomes max or min pooling if $\mathcal{A}_i^n$ is a Dirac delta in the maximum or minimum value.

### 3.3 Sequence Convolution Kernels

For the convolution and attention encoder $\mathcal{E}^n$ above, we separately apply n-gram kernels of varying sizes $n = 1, 2, 3, ....$ In addition to n-grams, we used *masked* convolution kernels that simulate the effect of skip-grams. The masking effectively zeros out certain entries in the convolution kernel as shown in Figure 2. Each kernel $n$ generates a corresponding fixed length encoding $\mathcal{E}^n$ of the input sequence.

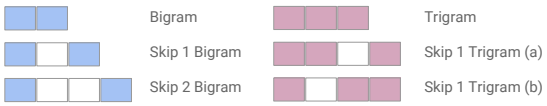


Figure 2: Skip-gram Simulation with Masked Convolution Kernel

### 3.4 Classification Layer

The convolution kernel width $n$ makes the network react to various word n-grams with a configurable parameter $N$ for each n-gram. We compute various n-gram and skip-gram convolution features and concatenate them to form a fixed length representation for the input document $x_k$:

$$\text{TextEncoder}(x_k) = \text{concat}(\mathcal{E}^1, \mathcal{E}^2, ...) \tag{5}$$

Finally, we use a feed-forward network with fully-connected layer over the fixed length text encoding for classification.

$$\text{output} = \psi(\text{TextEncoder}(x_k)) \tag{6}$$

We train the network with cross entropy loss and apply softmax over the output layer to obtain predicted probabilities $y_k^C$ for each class $C$ during inference.

## 4 Experiments & Results

### 4.1 Data Sets

We evaluate the performance of our approach on large scale document classification tasks, which are widely used in the research community.

- **Yelp** reviews from the Yelp Challenge (Tang et al., 2015) with ratings from 1 to 5.
- **Amazon** reviews from (Zhang et al., 2015) with ratings from 1 to 5.
- **Yahoo Answers** from (Zhang et al., 2015) with documents contain question title, question context and best answer and ten classes such as: Society & Culture; Science & Mathematics; Health; Education & Reference; Computers & Internet; Sports; Business & Finance; Entertainment & Musical; Family & Relationship; Politics & Government;

Table 1 shows the characteristics of each data set. We use the same test sets as (Tang et al., 2015).

| Data Sets | #Classes | Training | Test |
|---|---|---|---|
| Yelp | 5 | 650K | 50K |
| Amazon | 5 | 3M | 650K |
| Yahoo Answers | 10 | 1.4M | 60K |

Table 1: Data Set Characteristics

| Parameter | Description |
|---|---|
| $B$ | Dimension of the projection |
| $d$ | Dimension of the word embedding computed on the fly |
| $N_1$ | Number of unigram convolution channels |
| $N_2$ | Number of bigram convolution channels |
| $N_3$ | Number of trigram convolution channels |
| $N_4$ | Number of 4gram convolution channels |
| $N_5$ | Number of 5gram convolution channels |
| $S_2^1$ | Number of skip 1 bigram convolution channels |
| $S_2^2$ | Number of skip 2 bigram convolution channels |

Table 2: Hyper Parameters Searched

### 4.2 Experimental Setting

We setup our experimental evaluation, as follows: given a long text classification task and a data set, we construct a model with the hyper-parameters listed in Table 2 and we use a hyper-parameter search technique to find the optimal model for each data set. In addition to the parameters listed in Table 2, the search method also looks for optimal learning rate schedule and regularization scale. For the purpose of hyper-parameter search, we set aside 8% of the training data and use it as development set. The search method optimizes the *Accuracy* on the development set. Once found, we use the optimal model to evaluate *Accuracy* on the test set.

### 4.3 Implementation Details

Unlike prior document classification neural networks (Zhang et al., 2015; Tang et al., 2015; Yang et al., 2016) which rely on pre-trained word embeddings, our approach PRADO learns the projection weights on the fly during training (i.e word embeddings (or vocabularies) do not need to be stored). Prior to learning the projections, we did a simple pre-possessing that normalized the text to lowercase, introduced blank space before and after punctuation to make sure they are treated as separate tokens and tokenized the text by space.

We used different regularization scales for the initial fully connected layer and the rest of the network, as the majority of the parameters were in the first layer that computes the word embedding vectors on-the-fly. We used Adam optimizer (Kingma and Ba, 2014) with exponential learning rate schedule. For regularization, we used dropouts after the first layer and also distorted the input text by randomly inserting, deleting and transposing characters in the token with small probability.

### 4.4 Results and Model Comparisons

It is important to recall that the main goal of our work is to develop fast and efficient on-device neural text classification approach, which can achieve near state-of-the-art performance while satisfying the on-device small size and memory resource constrains. Therefore, it is not fair to directly compare PRADO on-device performance against existing approaches which do inference on cloud without constraints. Yet, we compare our approach against well established baselines and prior non-on-device work taking into consideration these differences. Table 3 shows the obtained results for each data set and method.

**Baseline Comparison**: We use the same baselines as described in (Zhang et al., 2015; Tang et al., 2015). They are traditional approaches, which rely on hand-crafted features such as bag-of-words with TFIDF and n-grams with TFIDF, and use linear or kernel classifiers. As it can be seen in Table 3, our PRADO approach consistently outperforms all baseline methods with +4.8 to +12.2 for Yelp, +5.9 to +17 for Amazon and +1.3 to +11.8 for Yahoo data sets. This definitely shows the power of the trainable projection on-device neural networks and attention mechanism.

**On-device Comparison**: We also show comparison against prior on-device neural network approach (Ravi and Kozareva, 2018). Their SGNN approach was targeted towards short text classification tasks and as shown in Table 3, our PRADO model achieves upto +40% improvement over SGNN demonstrating that PRADO is more powerful and suited for long text classification.

**Deep Learning Comparison**: Similarly, we also compare our PRADO approach against recent neural networks such as LSTM (Zhang et al., 2015), character and word-based CNNs (Zhang et al., 2015; Tang et al., 2015), Convolutional GRNN (Tang et al., 2015) and hierarchical attention (Yang et al., 2016). We compare performance for all data sets against (Tang et al., 2015; Zhang et al., 2015) since their test data sets are exactly the same as ours. However, (Yang et al., 2016) uses different test data sizes for the Yelp and Ya-

| Data Set | Yelp | Amazon | Yahoo |
|---|---|---|---|
| PRADO | **64.7** | **61.2** | **72.3** |
| PRADO 8-bit Quantized | **65.9** | **61.9** | **72.5** |
| SGNN (Ravi and Kozareva, 2018) | 35.4 | 39.1 | 36.6 |
| HN-ATT* (Yang et al., 2016) | - | **63.6** | - |
| HN-MAX* (Yang et al., 2016) | - | **62.9** | - |
| HN-AVE* (Yang et al., 2016) | - | **62.9** | - |
| LSTM-GRNN (Tang et al., 2015) | **67.6** | - | - |
| Conv-GRNN (Tang et al., 2015) | **66.0** | - | - |
| CNN-char (Zhang et al., 2015) | 62.0 | 59.6 | 71.2 |
| CNN-word (Tang et al., 2015) | 61.5 | - | - |
| CNN-word (Zhang et al., 2015) | 60.5 | 57.6 | 71.2 |
| Paragraph Vector (Tang et al., 2015) | 60.5 | - | - |
| LSTM (Zhang et al., 2015) | 58.2 | 59.4 | 70.8 |
| SVM + Bigrams (Tang et al., 2015) | 62.4 | - | - |
| SVM + Unigrams (Tang et al., 2015) | 61.1 | - | - |
| SVM + AverageSG (Tang et al., 2015) | 56.8 | - | - |
| SVM + SSWE (Tang et al., 2015) | 55.4 | - | - |
| BoW TFIDF (Zhang et al., 2015) | 59.9 | 55.3 | 71.0 |
| ngrams TFIDF (Zhang et al., 2015) | 54.8 | 52.4 | 68.5 |

Table 3: Evaluation Results

hoo Answers evaluation, therefore we do not report their results here. Another interesting aspect is that unlike prior work (Zhang et al., 2015; Tang et al., 2015; Yang et al., 2016) which rely on large word embeddings (with pre-training in many cases), our approach computes dynamically the projection embedding vectors. As shown in Table 3, PRADO significantly outperforms existing neural networks approaches like LSTM, CNN-char and CNN-word with +1.1 up to +6.5% depending on the task and data set, and it achieves comparable results to the hierarchical attention models of (Yang et al., 2016). This is very impressive given that PRADO produces magnitudes smaller and compact neural networks.

### 4.5 Impact of Hyper-parameters on Performance

Figure 3 shows *Accuracy* vs Model Size as a result of the different hyper-parameters explored during search. The figure shows results for each data set individually. Overall, PRADO's *Accuracy* improves as the number of parameters increases.

### 4.6 Model Size: PRADO vs Smaller RNNs

We further compare our PRADO model against smaller-sized variants of widely-used recurrent (LSTM) models. This study helps analyze the effectiveness of PRADO compared to other small neural models and answer the question: *Can popular RNN models be shrunk down to the same size as PRADO and still achieve high performance?* To construct baseline neural models at smaller sizes, we use an LSTM architecture with 64 hidden units as the state size and vary the input vocabulary size (i.e., picking top $K$ words ordered by frequency) and embedding dimensions $d$. Table 4 compares the performance of PRADO with different small and medium-sized LSTM models (achieved by varying $K, d$). Our results show that PRADO achieves the best performance with the lowest footprint (just 175K parameters) and high compression ratios (up to 100x smaller) compared to standard LSTM models.

| | Compression (PRADO ) | #Parameters | Accuracy |
|---|---|---|---|
| LSTM | 100x | 18.17M | 60.4 |
| | 43x | 7.6M | 60.3 |
| | 11x | 1.9M | 60.2 |
| | 3x | 504.7K | 59.9 |
| | 1x | 179.5K | 59.1 |
| PRADO | **1x** | 175K | **64.7** |

Table 4: Model Size vs. Performance for PRADO and small LSTMs on Yelp. Compression ratio for PRADO is shown wrt corresponding LSTM models.

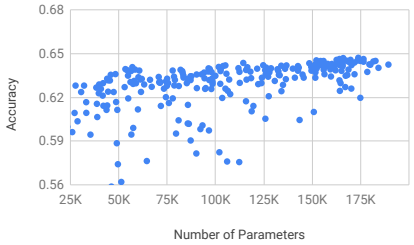


(a) Yelp 5

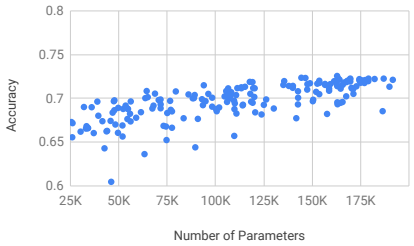


(b) Yahoo Answers 10

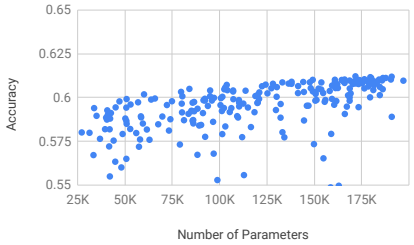


(c) Amazon 5

Figure 3: Model Size (*number of parameters* on x-axis) vs. Performance (*accuracy* on y-axis).

### 4.7 PRADO Analysis and Discussion

**N-gram Attention Focus**: To better understand our model, we analyze PRADO's attention distribution. We used a trained PRADO model from a particular data set and computed the attention distribution of the projected n-gram features for all samples in the test set. Then, we pull the word or word sequence index with the maximum attention and order them by frequency of the class. Table 5 shows examples of the bigrams for 5 and 1 star Yelp reviews. Note, that we did not select the vocabulary for the data, the model automatically learned to associate the words with the classes based on the trained projections.

**Skipgrams Attention Focus**: Similarly, we conduct an analysis for the attention distribution of the skip-1-bigrams channel. Table 5 shows sample of the most frequent entries for the 5 and 1 star Yelp reviews. The model captures and learns basic regular expressions. For instance, the skip gram "*waste * time*" captures " *waste your time*" or "*waste of time*". Similarly "*worst * ever*" captures "*worst food ever*", "*worst service ever*". Our analysis shows that overall our trainable projection with attention and convolution learns embedding representations that are powerful and capture the semantic similarity of words and phrases. This information helps PRADO during classification.

| *Bigrams* | |
|---|---|
| 5 Star | 1 Star |
| *highly recommend* | *zero stars* |
| *love this* | *horrible customer* |
| *hands down* | *1 star* |
| *was perfect* | *disgusting and* |
| *top notch* | *no stars* |
| *amazing service* | *better off* |
| *Skip-1-Bigrams* | |
| 5 Star | 1 Star |
| *waste * time* | *worst * ever* |
| *was * reasonable* | *give * stars* |
| *felt * comfortable* | *a * star* |
| *is * delicious* | *waste * money* |
| *you * comfortable* | *horrible * service* |
| *and * delicious* | *worst * experience* |

Table 5: Prado Attention Focus on Yelp Data

## 5 PRADO Runtime Performance

Our PRADO approach produces compact neural networks with tiny memory footprint. Next, we also show how to further optimize PRADO and help speed up on-device inference during runtime.

### 5.1 Training with quantization

We train a PRADO model variant with 8-bit quantization as described in (Jacob et al., 2018). This procedure simulates the quantization process during training by nudging the weights and activations towards a grid of discrete levels ($2^N$ levels, where $N$=8 is the number of bits). We estimate the activation ranges for each training batch and use exponential moving average to smooth the quantization ranges across training steps. (Jacob et al., 2018) noted that by training with quantization, they reached similar accuracy with 8-bit models as floating point ones on several image classification and object detection data sets. For text classification, we observed that training with quantization significantly improves accuracy as shown in Table 3. We believe that this is due to the improved regularization as quantization has the highest impact on Yelp. This dataset has relatively few training samples per class (see Table 1) which causes the model to overfit the training data and regularization provided by the operation that sim-

ulates quantization during training helps it generalize better. Furthermore, the model size of 8-bit quantized PRADO models is equal to the number of parameters. Figure 3 shows that PRADO can reach the performance reported in the Table 3 with model size of less than 200 Kilobytes. PRADO starts getting competitive results on the same data sets with tiny model size as low as 25 Kilobytes.

### 5.2 Computational Cost for Inference

We evaluate the computational cost of PRADO models for inference wrt *floating point (or integer) operations* and *latency* (in milliseconds). The number of floating point or integer operations in our model is dominated by the below factor

$$Td\left(2B + \sum_{k=1}^{5} kN_k + \sum_{s=1}^{2}(s+2)S_2^s\right) \quad (7)$$

which includes the operations from the first fully connected layer and the convolutional layers. It can be seen that our method scales linearly with the number of time steps $T$ and the dimension of the projected word embedding $d$. We measured the latency of processing a document with 1000 words using our quantized PRADO model on a Nexus 5x mobile phone to be between 20 to 40 ms.

## 6 Transfer Learning with PRADO

Recent popularity of several pre-trained word embedding approaches can be primarily attributed to their success and effectiveness at transfer learning for language tasks. Model representations trained on a data-rich domain can be leveraged for tasks and domains in limited-data scenarios. However, as we discussed earlier, these methods require storing and looking up huge pre-trained embedding tables unlike our PRADO approach which results in compact models.

Next, we evaluate the effectiveness of PRADO at learning robust representations and extend it for transfer learning scenarios. To establish a baseline, we took 10% of Yelp training data and trained and evaluated a baseline PRADO model with random initialization of weights. We compared this to initializing the parameters from a PRADO model trained on the larger Amazon data set. We ran two different experiments with this initialization.

- **Experiment A: Full Transfer** We freeze all weights in PRADO except the last fully connected classification layer. With this setup there were around 750 trainable parameters in the model.

- **Experiment B: Projection Transfer** We freeze the weights for only the first (projected embedding) layer and allow the convolution layers and the last classification layer to be trainable.

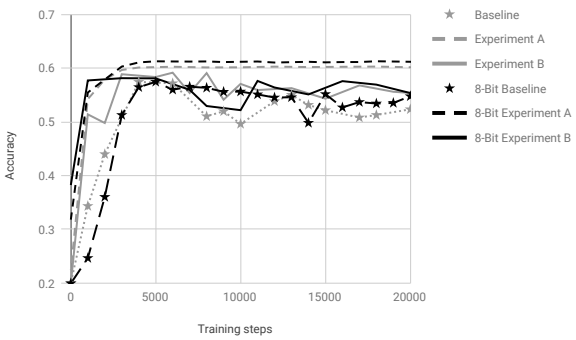


Figure 4: Transfer Learning Results on Yelp Data

We trained the baseline and transfer-learned variants with and without quantization. Figure 4 shows results of the transfer learning runs. We observe that with random initialization, the baseline PRADO model reaches a peak performance of around 57% and starts overfitting on the small Yelp training data set both with and without quantization. When transferring just the *projection layer* and fine-tuning the convolution and classification layers, training converges quickly and it reaches better peak accuracy. This demonstrates that the PRADO projection embeddings trained in one domain, even though tiny in size, captures useful information that can be leveraged to improve classification in another domain with limited training data. This is further improved when transferring the *full* PRADO model and fine-tuning just the last layer. In this case (Experiment A), the model converges to 61% and 60% accuracy with and without quantization respectively. The training also converges quicker than the baseline and it does not overfit anymore. We note that using our approach, a PRADO model transfer-learned on just 10% of training data achieves very competitive performance resulting in less than 10% drop in relative accuracy on Yelp data set (see Table 3).

## 7 Conclusion

In this paper, we proposed a novel trainable projection on-device neural network with attention,

which is capable of capturing long range dependencies making it flexible for solving long text classification tasks. We introduced trainable projection technique, which via the visualization of the attention mechanism shows that it effectively captures the semantic representation of the document, while still saving on storage and producing compact models. We demonstrated the effectiveness of our approach PRADO by conducting experiments on multiple large scale document classification tasks. The obtained results show that our approach improved upon traditional linear classifiers from 2% to 12%, character and word level CNNs and LSTMs neural approaches with 1% to 6% depending on the data set, task and approach. This is very impressive given the small compact model produced by PRADO . Similarly, the quantized version of our approach had consistent performance. Finally, we applied our model in a transfer learning scenario, which demonstrated the robustness of our approach and ability to further improve performance in limited data scenarios. In the future, we want to extend this approach to more natural language tasks and languages.

## References


Thang D. Bui, Sujith Ravi, and Vivek Ramavajjala. 2018. Neural graph learning: Training neural networks using graphs. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, WSDM '18, pages 64–71. ACM.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew G. Howard, Hartwig Adam, and Dmitry Kalenichenko. 2018. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 2704–2713.

Nitin Jindal and Bing Liu. 2007. Review spam detection. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 1189–1190.

Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, ECML'98, pages 137–142.

Rie Johnson and Tong Zhang. 2015. Effective use of word order for text categorization with convolutional neural networks. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 103–112. Association for Computational Linguistics.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *CoRR*, abs/1607.01759.

Anjuli Kannan, Karol Kurach, Sujith Ravi, Tobias Kaufmann, Andrew Tomkins, Balint Miklos, Greg Corrado, László Lukács, Marina Ganea, Peter Young, and Vivek Ramavajjala. 2016. Smart reply: Automated response suggestion for email. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 955–964.

Hamed Khanpour, Nishitha Guntakandla, and Rodney Nielsen. 2016. Dialogue act classification in domain-independent conversations using a deep recurrent neural network. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2012–2021. The COLING 2016 Organizing Committee.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Zornitsa Kozareva. 2015. Everyone likes shopping! multi-class product categorization for e-commerce. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1329–1333. Association for Computational Linguistics.

Fengfu Li and Bin Liu. 2016. Ternary weight networks. *CoRR*, abs/1605.04711.

Andrew McCallum and Kamal Nigam. 1998. A comparison of event models for naive Bayes text classification. In *Learning for Text Categorization: Papers from the 1998 AAAI Workshop*, pages 41–48.

Daniel Ortega and Ngoc Thang Vu. 2017. Neural-based context representation learning for dialog act classification. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages

247–252. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, 2(1-2):1–135.

Bo Pang and Sujith Ravi. 2012. Revisiting the predictability of language: Response completion in social media. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2012, July 12-14, 2012, Jeju Island, Korea*, pages 1489–1499.

Sujith Ravi. 2013. Scalable decipherment for machine translation via hash sampling. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 362–371, Sofia, Bulgaria. Association for Computational Linguistics.

Sujith Ravi. 2017. Projectionnet: Learning efficient on-device deep networks using neural projections. *CoRR*, abs/1708.00630.

Sujith Ravi. 2019. Efficient on-device models using neural projections. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5370–5379.

Sujith Ravi and Zornitsa Kozareva. 2018. Self-governing neural networks for on-device short text classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 804–810. Association for Computational Linguistics.

Sujith Ravi and Zornitsa Kozareva. 2019. On-device structured and context partitioned projection networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3784–3793.

Chinnadhurai Sankar, Sujith Ravi, and Zornitsa Kozareva. 2019a. On the robustness of projection neural networks for efficient text representation: An empirical study. *ArXiv*, abs/1908.05763.

Chinnadhurai Sankar, Sujith Ravi, and Zornitsa Kozareva. 2019b. Transferable neural projection representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3355–3360.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566. Association for Computational Linguistics.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *EMNLP*, pages 1422–1432. The Association for Computational Linguistics.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. 2016. Hierarchical attention networks for document classification. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 1480–1489.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 649–657. Curran Associates, Inc.

# Subword Language Model for Query Auto-Completion

**Gyuwan Kim**
Clova AI, NAVER Corp.
gyuwan.kim@navercorp.com

## Abstract

Current neural query auto-completion (QAC) systems rely on character-level language models, but they slow down when queries are long. We present how to utilize subword language models for the fast and accurate generation of query completion candidates. Representing queries with subwords shorten a decoding length significantly. To deal with issues coming from introducing subword language model, we develop a retrace algorithm and a reranking method by approximate marginalization. As a result, our model achieves up to 2.5 times faster while maintaining a similar quality of generated results compared to the character-level baseline. Also, we propose a new evaluation metric, mean recoverable length (MRL), measuring how many upcoming characters the model could complete correctly. It provides more explicit meaning and eliminates the need for prefix length sampling for existing rank-based metrics. Moreover, we performed a comprehensive analysis with ablation study to figure out the importance of each component[1].

## 1 Introduction

Query auto-completion (QAC) is one of the essential features for search engines. When a user types a query in the search box, QAC systems suggest most likely completion candidates (Cai et al., 2016). It not only saves time for users to enter search terms but also provides new information more than what was initially expected.

Recent neural QAC models in the literature employ character-level language models (Park and Chiba, 2017). It is a natural choice in that QAC systems need to respond whenever a user enters a query as input character-by-character. In addition to the accuracy, speed in terms of latency is also an indispensable prerequisite for practical QAC systems. The generation process is auto-regressive, and the size of the search space is exponential to the sequence length. Long character sequences make prediction slow and inaccurate in the constraints of limited computation. Also, character-level models are prone to errors due to long-range dependency (Sennrich, 2016). Therefore, these limitations arouse to consider alternatives to represent a query in a shorter sequence.

In this paper, we apply a subword language model for query auto-completion. Compared to character language models, subword language models reduce sequence length and the number of decoding steps significantly, thus resulting in much faster decoding. For subword-level modeling, a segmentation algorithm is necessary. Byte pair encoding (BPE) (Sennrich et al., 2015) is widely used, but noise in the data makes segmentation ambiguous and degrades BPE output. To address this issue, as well as BPE, we use subword regularization (SR) algorithm proposed by Kudo (2018) that stochastically samples multiple segmentations by utilizing a unigram language model. To our knowledge, we are the first to apply SR to language modeling.

Interestingly, language models for QAC should take care of the last token that may be incomplete. Like character language models, subword language models can represent incomplete tokens because it can generate any subsequence of sentences, whereas word language models cannot. If we segment prefix as given to encode it using neural networks, the segmentation of prefix may not match with that of ground truth query because the prefix is an incomplete substring of the original desired query. In that case, this enforced segmentation is less likely to appear in training, especially for deterministic segmentation such as BPE. As a result, the model starting from this segmentation

[1]Code is available at https://github.com/clovaai/subword-qac.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5022–5032,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

is unlikely to generate ground truth query. To consider every possible segmentation of target completion, we propose retrace algorithm that is going a few characters back from the end and generating candidates with the restriction that they should match with retraced characters. For the case of SR models, due to the stochasticity of segmentation, we should marginalize over all possible segmentations to calculate the likelihood of a query. For better approximation than just argmax, we perform reranking with approximated marginalization using the output of beam search. Experimental results show that these techniques improve the robustness of the decoding process of the subword language model to achieve close generation quality compared to the character baseline.

We propose a novel metric for query autocompletion evaluation, called mean recoverable length (MRL). This metric remedies shortcomings of common QAC evaluation metrics, mean reciprocal rank (MRR) and partial-matching MRR (PMRR), which require sampling of a prefix length and are favorable to short queries. We conduct comprehensive ablation study and analysis of our models on these three metrics.

## 2 Related Work

One of the successful traditional QAC approaches is most popular completion (MPC) (Bar-Yossef and Kraus, 2011), which returns the most frequent candidates among all previously observed queries that match the prefix. After extracting candidates, reranking algorithms (e.g., LambdaMART (Burges, 2010)) with additional features are used to align final candidates. These methods cannot generate previously unseen queries by nature. Contrary to traditional approaches based on information retrieval, neural approaches can generalize to unseen prefixes.

Choosing an appropriate granularity level for text segmentation has been long studied over the variety of natural language processing problems. It can be a character, subword, word, phrase, sentence, and even paragraph. A trade-off between them exists, and the best performing granularity often varies depending on tasks and datasets. Character models are widely used to address natural language processing tasks including text classification (Kim, 2014; Zhang et al., 2015; Conneau et al., 2016), language modeling (Hwang and Sung, 2017; Al-Rfou et al., 2018), machine translation (Chung et al., 2016; Lee et al., 2017), etc.

Currently, neural machine translation systems widely use subword segmentation as *de facto*. Mikolov et al. (2012) observed that a subword language model is advantageous in that it achieves better performance compared to character-level models with zero out-of-vocabulary rate and smaller model size. BERT (Devlin et al., 2018) uses a subword as the unit token for their (masked) language models.

Word-level segmentation can easily shorten sequence length compared to character-level. However, word-level models require larger vocabulary size and the number of parameters to learn. Also, it causes data sparsity issue. Because the vocabulary of words is usually fixed before training, it cannot generate out-of-vocabulary words by itself. Search systems are especially in an open vocabulary setting. For word-level models, it is hard to deal with the last incomplete token because it may not be in the vocabulary, unlike character-level naturally handle it. Even if the vocabulary contains this token, the decoding process may be somewhat different from expected.

Word-character hybrid models were proposed to overcome the out-of-vocabulary problem (Luong and Manning, 2016; Wu et al., 2016). A word-level decoder generates a word sequence, and when it generates a special <UNK> token, a character-level decoder generates a character sequence on top of it. These two decoders are connected hierarchically. Word models assume whitespace as the boundary of words. In some languages including Japanese and Chinese, segmentation of sentences into words is unknown in advance and sometimes vague. Moreover, input queries usually include much noise such as typos, grammatical errors and spacing errors. The problems mentioned above hinder word-level processing for QAC.

Park and Chiba (2017) and Fiorini and Lu (2018) incorporate word information by concatenating its embedding with character embedding only at the word boundary and use a special <INC> token embedding for non-boundary positions. This mechanism is inefficient in that the word signal is sparse. Most of the word-character hybrid models focus on input representation rather than generation. Usually, their representations are concatenated, or composition functions are learned (Kim et al., 2016; Miyamoto and Cho,

2016). Even though they use word information to the input, the decoding process of their models is still in the character-level.

We can interpret generating a subword which is a concatenation of characters as parallel decoding of characters (Stern et al., 2018). In this sense, non-autoregressive neural machine translation (Gu et al., 2017; Lee et al., 2018) is related to our work. They also aim to improve decoding speed with minimal performance degradation. Our model and decoding method can be used for non-autoregressive NMT in place of a character-level decoder, and in the opposite direction, we can apply their approaches to QAC vice versa.

## 3 Subword Language Model

Let $\Sigma$ be the set of all possible characters and $V$ be the vocabulary of tokens. Each token is a character or a concatenation of characters, and it is a subword in our case. A language model estimates the probability of a token sequence $\mathbf{t}$ where the probability distribution of token $t_i$ at each step $i$ is conditioned on the previous tokens $t_{<i}$:

$$p(\mathbf{t};\theta) = \prod_{i=1}^{|\mathbf{t}|} p(t_i|t_{<i};\theta)$$

where $t_i \in V$ and $\theta$ is a set of model parameters. For a token sequence $\mathbf{t}$, we can map it to a query $\mathbf{q} = concat(\mathbf{t})(= t_1 \oplus t_2 \oplus \cdots \oplus t_{|\mathbf{t}|})$ by concatenating itself sequentially. Then, the probability of a given query $\mathbf{q}$ is the sum of the probability over the set of all possible segmentation $\mathbf{t}$, $S(\mathbf{q}) = \{\mathbf{t} : concat(\mathbf{t}) = \mathbf{q}\}$:

$$p(\mathbf{q};\theta) = \sum_{\mathbf{t}\in S(\mathbf{q})} p(\mathbf{t};\theta).$$

Similar to (Chan et al., 2016), segmentation $\mathbf{t}$ can be interpretable as a latent decomposition of the query $\mathbf{q}$.

### 3.1 Segmentation

In character-level language modeling, token vocabulary $V$ is equal to $\Sigma$, and segmentation is performed by merely splitting every character. We exclude word-level language modeling which splits a sentence by whitespace from consideration due to its limitations mentioned in Section 2.

In the case of subword language modeling, we use two widely used segmentation algorithms: (1) byte pair encoding (BPE) and (2) subword regularization (SR). Formally, a segmentation algorithm defines a probability distribution over a token sequence $\mathbf{t}$ conditioned on given query $\mathbf{q}$: $p_{seg}(\mathbf{t}|\mathbf{q})$.

The BPE algorithm is deterministic because it segments greedily from left to right. On the other hand, SR can sample multiple segmentations stochastically. The number of possible segmentations is exponentially large. It is hard to calculate the likelihood of a given sentence using dynamic programming because even with the same prefix, hidden states vary upon different previous tokenization. Marginalization over all possible segmentations of very long sequences is intractable. In sum, we compare character-level and subword-level (BPE, SR) language modeling.

### 3.2 Training

We can derive an unbiased gradient estimator of the log-likelihood of a query by using Bayes' theorem and the identity of $\nabla_\theta f(x;\theta) = f(x;\theta)\nabla_\theta \log f(x;\theta)$ assuming $f(x;\theta) \neq 0$ for all $x$ (Williams, 1992):

$$\nabla_\theta \log p(\mathbf{q};\theta) = \mathop{\mathbb{E}}_{\mathbf{t}\sim p(\mathbf{t}|\mathbf{q};\theta)} \nabla_\theta \log p(\mathbf{t};\theta).$$

However, since sampling $\mathbf{t}$ from $p(\mathbf{t}|\mathbf{q};\theta)$ is computationally expensive, we heuristically use $p_{seg}(\mathbf{t}|\mathbf{q})$ instead. Regardless of the language model parameters $\theta$, segmentation model $p_{seg}$ is learned before the language model training and can be used to sample $\mathbf{t}$ easily. The better way to approximate the distribution $p(\mathbf{t}|\mathbf{q};\theta)$ will be explored in the future.

Our training objective becomes equivalent to maximizing the average log-likelihood of the segmentation of sentences:

$$\begin{aligned} L(\theta) &= \frac{1}{|Q|}\sum_{\mathbf{q}\in Q} \log p(\mathbf{t};\theta) \\ &= \frac{1}{|Q|}\sum_{\mathbf{q}\in Q}\sum_{i} \log p(t_i|t_{<i};\theta), \end{aligned}$$

where $Q$ is the training set of all queries, and $\mathbf{t}$ is the segmentation of a query $\mathbf{q}$ sampled from $p_{seg}(\mathbf{t}|\mathbf{q})$ which depends on the segmentation algorithm. This objective is equal to the average negative log-likelihood of sentences if and only if the segmentation is deterministic. The gradients of the loss function are computed using the back-propagation through time (BPTT) (Rumelhart et al., 1986).

## 4 Decoding

Given prefix $\mathbf{p}$, let the set of its completions be $Q_+(\mathbf{p})$ and the set of their tokenizations be $S_+(\mathbf{p}) = \{\mathbf{t} : concat(\mathbf{t}) \in Q_+(\mathbf{p})\}$. We want to find the most likely completion $\hat{\mathbf{q}}$:

$$\begin{aligned}\hat{\mathbf{q}} &= \underset{\mathbf{q}}{\operatorname{argmax}} \log p(\mathbf{q}|\mathbf{p}) \\ &= \underset{\mathbf{q}\in Q_+(\mathbf{p})}{\operatorname{argmax}} \log p(\mathbf{q}) \\ &= \underset{\mathbf{q}\in Q_+(\mathbf{p})}{\operatorname{argmax}} \log \sum_{\mathbf{t}\in S(\mathbf{q})} p(\mathbf{t}) \end{aligned} \quad (1)$$

however, this is obviously intractable to search in the infinitely large token sequence space. We approximate this by decoding for the best token sequence $\hat{\mathbf{t}}$ and then returning its corresponding query $\tilde{\mathbf{q}}$ by concatenating its token sequentially:

$$\begin{aligned}\hat{\mathbf{t}} &= \underset{\mathbf{t}\in S_+(\mathbf{p})}{\operatorname{argmax}} \log p(\mathbf{t}) \\ \tilde{\mathbf{q}} &= concat(\hat{\mathbf{t}})\end{aligned}$$

Basic practice is segmenting $\mathbf{p}$, feeding it into language model to encode $\mathbf{p}$, and using it for the decoding. Since finding $\hat{\mathbf{t}}$ is also intractable, beam search decoding is used but only results in suboptimal predictions. We will improve this incrementally with techniques following.

### 4.1 Retrace Algorithm

There is no guarantee that the end of given prefix matches the tokenization boundary of the completed query. To address this possibility of the incompleteness at the end of a prefix, we can retrace a few characters and generate from there. For the case (call it $R_r$) where the last token that overlaps with the prefix finishes $r$ characters before the end of the prefix, first starting from a token sequence of $\mathbf{p}_{1:|\mathbf{p}|-r}$, we can perform beam search decoding on the restriction that the next token should cover the remaining part of the prefix and the next new character. Figure 1 illustrates this algorithm.

This process is unnecessary for a character-level model since every token is a single character. On the other hand, the retrace algorithm is helpful for subword models, especially BPE models which have deterministic segmentation algorithm.

We can limit the maximum step of retrace by $L$ to only consider $R_r$ where $0 \leq r \leq L$ because of the computational issue. We will denote this limitation as $R^L$. $R^0$ is the usual case without the retrace algorithm, and $R^\infty$ counts every possible retrace steps.

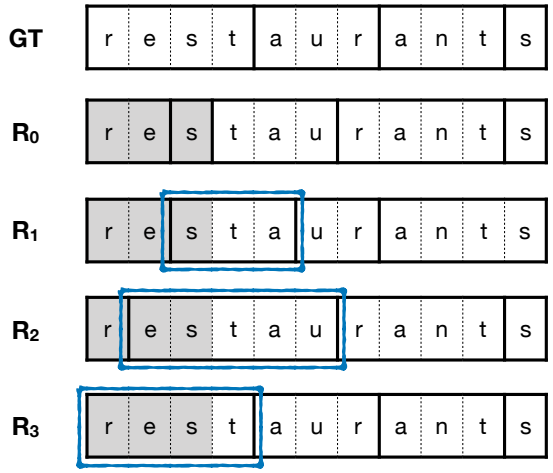


Figure 1: Illustration of retrace algorithm with the example of "restaurants." The gray area means given prefix ("res") of the query. The solid line indicates the boundary of the segmentation. GT is the segmentation of ground truth query. Possible examples of the generated sequence of tokens belonging to the case $R_r$ are visualized. Blue boxes indicate a fixed segmentation with retrace algorithm at the end of the prefix.

### 4.2 Reranking Method by Approximate Marginalization

QAC system has to suggest $N$ completion candidates sorted in order of likelihood rather than finding only the best completion candidate. We can extract a set of top $B(\geq N)$ candidates using beam search with beam size $B$, namely $T_B = \{\mathbf{t}^1, \cdots, \mathbf{t}^B\}$ in the descending order of likelihood. In the case of deterministic segmentation, $\mathbf{q}^i = concat(\mathbf{t}^i)$ are mutually different. i.e. $|Q_B| = B$ for $Q_B = \{\mathbf{q}^1, \cdots, \mathbf{q}^B\}$. Then, trivially our prediction would be $(\mathbf{q}^1, \cdots, \mathbf{q}^N)$.

On the other hand, in the case of stochastic segmentation, same query $\mathbf{q}^i, \mathbf{q}^j (i \neq j)$ with different token sequence $\mathbf{t}^i, \mathbf{t}^j$ may exist. The obvious way is merely removing duplicates.

On the assumption that $\log p(\mathbf{t}) \gg \log p(\mathbf{t}')$ for all $\mathbf{t} \in T_B$ and $\mathbf{t}' \notin T_B$, Equation (1) implies that marginalization over the final beam outputs can provide better approximation:

$$\begin{aligned}\hat{\mathbf{q}} &\approx \underset{\mathbf{q}\in Q_+(\mathbf{p})}{\operatorname{argmax}} \log \sum_{\mathbf{t}\in T_B} p(\mathbf{t}) \\ &= \underset{\mathbf{q}\in Q_B}{\operatorname{argmax}} \log \sum_{\mathbf{t}\in T_B} p(\mathbf{t})\end{aligned}$$

In other words, reranking after summing out the probability of duplicates can give better ordered list of candidates.

## 5 Evaluation Metric

### 5.1 MRR and PMRR

One of the most standard QAC evaluation metrics is the mean reciprocal rank (MRR). The MRR for the QAC system $m$ is calculated with test dataset $Q_{test}$ as follows:

$$MRR(m) = \frac{1}{|Q_{test}|} \sum_{\mathbf{q} \in Q_{test}} \mathrm{RR}(\mathbf{q}, m(\mathbf{p})),$$

where $\mathbf{p}$ is a prefix of a query $\mathbf{q}$ and $m(\mathbf{p})$ is the ranked list of candidate completions of $\mathbf{p}$ from $m$. RR is reciprocal rank of $\mathbf{q}$ if $\mathbf{q}$ is in $m(\mathbf{p})$, otherwise 0.

Since it is hard to get a pair of the desired query and its prefix in a real situation, we should synthetically select a prefix by cutting off a given query for the evaluation. Common practice is to uniformly sample from all possible prefixes within minimal length constraint in characters (or words) However, real distribution of the prefix length may differ to the uniform distribution. For example, users tend to engage with QAC at the position close to the boundary of words, and after typing half of query characters (Mitra et al., 2014). Due to the stochastic characteristic of prefix sampling processes or their difference among distinct QAC systems, evaluation results are inconsistent even with the same test dataset. To prevent this problem, a sampling function should be concretely specified.

Park and Chiba (2017) introduced a new metric, partial-matching MRR (PMRR):

$$PMRR(m) = \frac{1}{|Q_{test}|} \sum_{\mathbf{q} \in Q_{test}} \mathrm{PRR}(\mathbf{q}, m(\mathbf{p})),$$

where partial-matching reciprocal rank PRR is the reciprocal of the index of the first candidate such that the original query is the same as it or starts with it plus whitespace. If there is no such candidate, PRR equals to 0.

PMRR also requires sampling of the prefix length. PMRR values are often omitted in the literature because of the similar tendency to MRR. In other words, PMRR does not give much additional information about the quality of the generated results.

### 5.2 Recoverable Length

To avoid the synthetic sampling process and length dependency, we propose a new evaluation metric for query auto-completion, namely mean recoverable length (MRL). We define recoverable length RL as the number of characters right before the first position where candidates do not have the query. When all characters of a query are known, we can readily suggest itself. If we delete chars from right to left one-by-one, the ground truth query will disappear in the list of candidate completions. For example, if $\mathbf{q} \in m(\mathbf{q}_{1:|\mathbf{q}|-l})$ for $l = 1, 2, 3$ but not 4, recoverable length of this query with respect to the QAC system $m$ is 3. MRL is mean of recoverable length:

$$MRL(m) = \frac{1}{|Q_{test}|} \sum_{\mathbf{q} \in Q_{test}} \mathrm{RL}_m(\mathbf{q})$$

MRL is a useful metric for additive QAC which suggests one word at a time instead of a whole-query completion (Vargas et al., 2016) in that it measures how many characters the system can predict correctly at once. MRL does not care about the order of candidates and check whether they contain the target query or not. Lastly, it eliminates the need to choose a prefix length in the test data.

## 6 Experiments

### 6.1 Data

We use the public AOL query log dataset (Pass et al., 2006) for the experiments. We split data based on time. Among three months of the entire span, we use last one week as test data and one week right before the test data as validation data. It is close to a real scenario where future queries are unseen during the training.

We perform Unicode NFKC normalization and remove non-ASCII characters. For simplicity, we change uppercase alphabets to lowercase. After normalization and changing to lowercase, only 43 unique characters including special symbols `<BOS>`, `<EOS>` and `<UNK>` remain. We substituted multiple adjacent spaces to a single one and removed leading or trailing spaces. We merged duplicates which appear adjacently by the same user and the same query. Queries of a length shorter than three characters are filtered out.

In total, the training, validation, test data contain 17,521,031, 1,521,971, and 1,317,632 queries, respectively. Among the test data, 670,810 queries are seen, and 646,822 queries are unseen in the training data. Almost half of the test data are unseen.

| Model | MRR | | | PMRR | | | MRL | | | Execution Speed (QPS) | | Decode Length |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Seen | Unseen | All | Seen | Unseen | All | Seen | Unseen | All | CPU | GPU | |
| MPC | .570 | .000 | .290 | .616 | .095 | .360 | 8.06 | 0.00 | 4.10 | >100 | | N/A |
| Char | .458 | .160 | .311 | .552 | .372 | .464 | 5.77 | 4.24 | 5.02 | 11.0 (1.0x) | 16.5 (1.0x) | 14.5 |
| BPE | .242 | .085 | .164 | .305 | .232 | .269 | 0.49 | 0.54 | 0.51 | 24.2 (2.2x) | 37.4 (2.3x) | 7.1 |
| BPE+R$^1$ | .427 | .156 | .294 | .517 | .368 | .444 | 5.28 | 3.98 | 4.64 | 15.8 (1.4x) | 27.3 (1.7x) | 11.8 |
| BPE+R$^2$ | .430 | .157 | .296 | .520 | .369 | .446 | 5.44 | 4.01 | 4.74 | 15.5 (1.4x) | 27.2 (1.6x) | 12.2 |
| BPE+R$^\infty$ | **.431** | **.157** | **.296** | **.520** | **.369** | **.446** | **5.50** | **4.01** | **4.76** | 15.3 (1.4x) | 26.9 (1.6x) | 12.2 |
| SR | .422 | .148 | .288 | .541 | **.379** | .461 | 5.11 | 3.82 | 4.48 | 20.8 (1.9x) | 40.1 (2.4x) | 6.8 |
| SR+M | .424 | .149 | .289 | .535 | .373 | .455 | 5.14 | 3.85 | 4.50 | 19.6 (1.8x) | 40.0 (2.4x) | 6.8 |
| SR+R$^\infty$ | .423 | .148 | .289 | **.541** | .378 | **.461** | 5.14 | 3.83 | 4.50 | 16.3 (1.5x) | 29.6 (1.8x) | 7.4 |
| SR+R$^\infty$+M | **.427** | **.150** | **.291** | .538 | .375 | .458 | **5.19** | **3.88** | **4.54** | 16.2 (1.5x) | 28.7 (1.7x) | 7.4 |

Table 1: Results of completion generation. We group MPC, character language model baseline, and two subword language models separately. +R implies the retrace algorithm. +M implies reranking with approximate marginalization. QPS stands for query per seconds. The higher the QPS, the better. The best results for each column related to accuracy are shown in **bold** for each segmentation algorithm (BPE and SR). SR model shows higher unseen PMRR scores (underlined). Our models are faster than the character baseline.

### 6.2 Implementation Details

The language model used in the experiments consists of an input layer, a single LSTM layer, a projection layer, and an output layer. For the LSTM layer, following Melis et al. (2017) and Jaech and Ostendorf (2018), we apply layer normalization (Ba et al., 2016) to each gate and couple input and forget gates. We tie input and output embeddings for better generalization (Press and Wolf, 2016; Inan et al., 2016). We set the LSTM hidden size to 600 and the input embedding size to 100.

We train three individual language models: namely Char, BPE, SR. The only difference among models is how to segment a given sentence into tokens. We believe that increasing model size (number of LSTM layers, input size, and hidden size) would improve the performance. Also, the best set of a combination may differ depending on models. However, we use the same model size for the character baseline and our variants for the fairness since our goal is proposing a new method and comparing between baseline and ours rather than achieving the best performance with the restriction of having a similar number of parameters.

We use the off-the-shelf SentencePiece (Kudo and Richardson, 2018) library for vocabulary learning and segmentation of the vocabulary size 256 using BPE, SR algorithms. For the subword regularization, we use sampling parameters $l = \infty$, $\alpha = 0.2$ for the training. We choose this value by the generation accuracy on the validation data. Increase of model size and computation due to larger vocabulary size are not substantial. By setting a manageable amount of vocabulary size, we can balance performance and computational cost.

For the computational efficiency, we truncated queries in the training data to a length of 40. Only less than 3% of queries in the training data are longer than 40 characters. We train models for thirty epochs by the Adam (Kingma and Ba, 2014) optimizer with a learning rate 5e-3 and batch size 1024. Following Smith (2018), we use a large learning rate and batch size. We use recurrent dropout (Semeniuta et al., 2016) with probability of 0.25 for regularization. The best model is chosen using validation data.

Using QAC models, we generate $N = 10$ completion candidates using beam search decoding of a beam width $B = 30$.

For the SR models, the segmentation of $\mathbf{p}$ (or retraced $\mathbf{p}_{1:|\mathbf{p}|-r}$) is not deterministic and generated completions may differ depending on its segmented token sequences with their different encoded representation. By following (Kudo, 2018), we can find the most likely segmentation sequence $\mathbf{t}$ starting from all of the $n$-best segmentations $\tilde{\mathbf{t}}_1, \cdots, \tilde{\mathbf{t}}_n$ of $S(\mathbf{p})$ rather than from only $\tilde{\mathbf{t}}_1$. However, we observe that this $n$-best decoding performs worse than one-best decoding. One possi-

ble reason is that segmentations which are not the best have a smaller probability as itself and so less likely to appear in training and less competitive in the process of beam search. For this reason, we set $n$ to 1.

We used a trie (Fredkin, 1960) data structure to implement most popular completion baseline.

All experiments were performed on NAVER Smart Machine Learning (NSML) platform (Sung et al., 2017; Kim et al., 2018).

### 6.3 Decoding Results

We performed comprehensive experiments to analyze the performance of query auto-completion. Table 1 shows the generation result of MPC, the character baseline, and our model variants. For BPE models, we varied the maximum retrace step to 0 (without retrace algorithm), 1, 2, and $\infty$ (no limitation on retracing step size). For SR models, we compare decoding results without any techniques, with marginalization only, with retrace algorithm only, and with both.

MPC is a very fast and remarkably strong baseline. It is worse than language models in the overall score (MRR, PMRR, and MRL), but better for previously seen queries. However, it is unable to predict unseen queries. Even with efficient data structures, MPC requires huge memory to keep statistics of all previous queries. As a practical view, combining frequency-based traditional method and neural language model approach can boost the accuracy and meet trade-off between the performance and computational costs.

MRRs and PMRRs of our best methods are close to that of the character model with less than 0.02 point drop. Notably, the SR model has better generalization ability in that their PMRR for unseen queries is higher than that of the character model. In a real scenario, it is more critical because unseen queries come in increasingly as time goes by.

We measure execution time with Tesla P40 GPU and Xeon CPU. Subword-level models are up to 2.5 times faster than the character baseline with minimal loss in performance both in CPU and GPU. Decoding length which is maximum suffix length until beam search ends correlates with the number of floating-point operations. Subword models significantly reduce the decoding length from the character baseline more than two times shorter by generating multiple characters at once.

| **re** | | **nat** | |
|---|---|---|---|
| Char | SR | Char | SR |
| realtor.com | recipes | national city bank | national bank |
| recipes | rentals | nationalcity.com | national city |
| real estate | real estate | national city | nationwide |
| remax | restaurants | national geographic | national parks |
| realtor | resources | national car rental | national park |

Table 2: Examples of top 5 candidates of completions given "re" and "nat" as prefixes generaed by the character baseline and SR model.

Models with additional techniques perform better than without them. Especially, retrace algorithm gives huge improvement for BPE case. Without retrace algorithm, BPE models do not work well. On the other hand, SR models only obtain small improvement. Because retrace algorithm goes back, it increases the decoding length and slows down the speed. Although current retrace algorithm is implemented straightforwardly, it can be improved by merging beams efficiently. Most of the subword lengths are equal or shorter than 3, so retrace of step 2 is quite enough, and $R^2$ get a close result with $R^\infty$.

The reranking method by approximate marginalization gives a small amount of improvement and is orthogonal to retrace algorithm. Marginalization method increases MRR but decreases PMRR. It is plausible in that it changes the order of candidates by reranking. The effect of marginalization would be better if we use a bigger beam size. Because the reranking process is done after beam search decoding which takes most of the decoding time and only consists of summation and sorting the final beam outputs, it does not take a long time.

We also had experimented by increasing the vocabulary size. The accuracy of BPE models degrades fast as the vocabulary size increases. On the other hand, the performance of SR models is quite stable due to the regularization effect during training. As desired, the larger the dictionary size, the shorter the decoding length. Whereas computations run in parallel in GPU, the number of operations for the output layer in the language model is proportional to the vocabulary size in CPU. Therefore, a larger vocabulary size does not always guarantee speedup for execution in the CPU. More thorough investigation about the correlation between QAC performance and the vocabulary size of subword language models remains for future work.

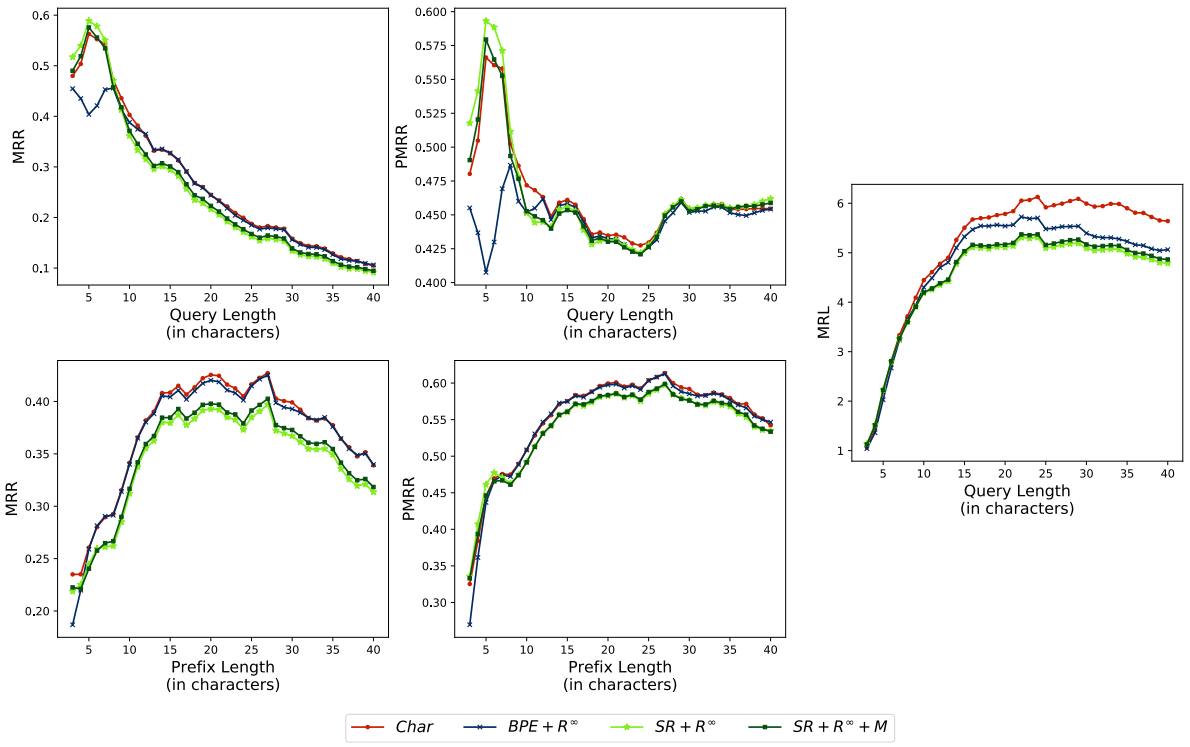


Figure 2: Comparison of the character-level baseline model and our best models by changing query length and prefix length in terms of three evaluation metrics: MRR, PMRR, and MRL. MRL is only varied by query length because it does not require prefix length sampling.

Table 2 shows examples of decoding results. Our model generates more concise and realistic examples.

### 6.4 Analysis on Evaluation Metrics

As shown in Figure 2, we compared our best models on three evaluation metrics (MRR, PMRR, and MRL) by changing the query length and prefix length. MRR and PMRR are more favorable to shorter queries. They drop very fast as the query becomes longer. For a longer query, the suffix length after sampling prefix has more chance to be longer. The search space increases exponentially with its suffix length. Even though QAC systems could generate realistic candidates, it is quite hard to match a long sequence with the ground truth. As the prefix length becomes longer which means that much information for determining the query has been given, the completion performance improves.

Interestingly, MRR and MRL of BPE are higher than those of SR, although BPE is worse in terms of PMRR than SR. For short queries, SR outperforms the character baseline. On the other hand, BPE is poor when the query length (or prefix length) is short. However, for a longer case, its MRR is almost close to that of the character baseline.

MRR and PMRR are highly dependent on the length distribution of test data. In contrast, MRL keeps the order between different methods as the query length changes. MRL is more reliable in the respect that it could provide consistent order between methods regardless of query length distribution. For long queries lengths, MRL stays in the flat area. Normalizing recoverable length based on the query length might be necessary.

## 7 Future Work

Approximation in training (Section 3.2) and decoding (Section 4) deteriorate the accuracy of subword language modeling. One possible solution to reduce the accuracy gap between the character language model baseline and the subword language model is knowledge distillation (Hinton et al., 2015; Kim and Rush, 2016; Liu et al., 2018) from character-level language models. A student model can learn to match an estimation of query probability with that of a teacher model.

Another interesting research direction is learning segmentation jointly with language model

(Kawakami et al., 2019; Grave et al., 2019) rather than using fixed pretrained segmentation algorithms. A conditional semi-Markov assumption allows exact marginalization using dynamic programming (Ling et al., 2016; Wang et al., 2017). Nevertheless, beam search decoding on those language models, especially faster decoding, is nontrivial.

Proposed method can be extended to wide range of tasks. Query suggestion (Sordoni et al., 2015; Dehghani et al., 2017) and query reformulation (Nogueira and Cho, 2017) are related to QAC and well-established problems. They both are also possible applications of the subword-level modeling. (Drexler and Glass, 2019) used subword regularization and beam search decoding for end-to-end automatic speech recognition.

Lastly, implementation with more advanced data structure (Hsu and Ottaviano, 2013) and parallel algorithms to speed up and meet memory limitation are necessary for the real deployment (Wang et al., 2018). It would be helpful if the computation is adaptively controllable on-the-fly (Graves, 2016) at the runtime depending on the situation.

## 8 Conclusion

In this paper, we propose subword language models for query auto-completion with additional techniques, retrace algorithm and reranking with approximate marginalization. We observed subword language models significant speedup compared to the character-level baseline while maintaining the generation quality. Our best models achieve up to 2.5 times faster decoding speed with less than 0.02 point drop of MRR and PMRR.

Using a subword language model, we build an accurate and much faster QAC system compared to the character-level language model baseline. Although there is still much room for improvement on hyperparameter optimization, decoding search, and neural architectures like Transformer (Vaswani et al., 2017; Dai et al., 2019), the goal of this work is to prove that the subword language model is an attractive choice for QAC as an alternative to the character-level language model, especially if latency is considered.

We believe that our newly proposed metric, mean recoverable length (MRL), provides fruitful information for the QAC research in addition to conventional evaluation metric based on ranks.

## Acknowledgments

The author would like to thank Clova AI members and the anonymous reviewers for their helpful comments.

## References

Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Llion Jones. 2018. Character-level language modeling with deeper self-attention. *arXiv preprint arXiv:1808.04444*.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Ziv Bar-Yossef and Naama Kraus. 2011. Context-sensitive query auto-completion. In *Proceedings of the 20th international conference on World wide web*, pages 107–116. ACM.

Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11(23-581):81.

Fei Cai, Maarten De Rijke, et al. 2016. A survey of query auto completion in information retrieval. *Foundations and Trends® in Information Retrieval*, 10(4):273–363.

William Chan, Yu Zhang, Quoc Le, and Navdeep Jaitly. 2016. Latent sequence decompositions. *arXiv preprint arXiv:1610.03035*.

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. *arXiv preprint arXiv:1603.06147*.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2016. Very deep convolutional networks for text classification. *arXiv preprint arXiv:1606.01781*.

Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

Mostafa Dehghani, Sascha Rothe, Enrique Alfonseca, and Pascal Fleury. 2017. Learning to attend, copy, and generate for session-based query suggestion. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1747–1756. ACM.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jennifer Drexler and James Glass. 2019. Subword regularization and beam search decoding for end-to-end automatic speech recognition. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6266–6270. IEEE.

Nicolas Fiorini and Zhiyong Lu. 2018. Personalized neural language models for real-world query auto completion. *arXiv preprint arXiv:1804.06439*.

Edward Fredkin. 1960. Trie memory. *Communications of the ACM*, 3(9):490–499.

Édouard Grave, Sainbayar Sukhbaatar, Piotr Bojanowski, and Armand Joulin. 2019. Training hybrid language models by marginalizing over segmentations. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 1477–1482.

Alex Graves. 2016. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. 2017. Non-autoregressive neural machine translation. *arXiv preprint arXiv:1711.02281*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Bo-June Paul Hsu and Giuseppe Ottaviano. 2013. Space-efficient data structures for top-k completion. In *Proceedings of the 22nd international conference on World Wide Web*, pages 583–594. ACM.

Kyuyeon Hwang and Wonyong Sung. 2017. Character-level language modeling with hierarchical recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 5720–5724. IEEE.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2016. Tying word vectors and word classifiers: A loss framework for language modeling. *arXiv preprint arXiv:1611.01462*.

Aaron Jaech and Mari Ostendorf. 2018. Personalized language model for query auto-completion. *arXiv preprint arXiv:1804.09661*.

Kazuya Kawakami, Chris Dyer, and Phil Blunsom. 2019. Learning to discover, ground and use words with segmental neural language models. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 6429–6441.

Hanjoo Kim, Minkyu Kim, Dongjoo Seo, Jinwoong Kim, Heungseok Park, Soeun Park, Hyunwoo Jo, KyungHyun Kim, Youngil Yang, Youngkwan Kim, et al. 2018. Nsml: Meet the mlaas platform with a real-world case study. *arXiv preprint arXiv:1810.09957*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *AAAI*, pages 2741–2749.

Yoon Kim and Alexander M Rush. 2016. Sequence-level knowledge distillation. *arXiv preprint arXiv:1606.07947*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. *ACL*.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully character-level neural machine translation without explicit segmentation. *Transactions of the Association for Computational Linguistics*, 5:365–378.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. *arXiv preprint arXiv:1802.06901*.

Wang Ling, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, Andrew Senior, Fumin Wang, and Phil Blunsom. 2016. Latent predictor networks for code generation. *arXiv preprint arXiv:1603.06744*.

Yijia Liu, Wanxiang Che, Huaipeng Zhao, Bing Qin, and Ting Liu. 2018. Distilling knowledge for search-based structured prediction. *arXiv preprint arXiv:1805.11224*.

Minh-Thang Luong and Christopher D Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. *arXiv preprint arXiv:1604.00788*.

Gábor Melis, Chris Dyer, and Phil Blunsom. 2017. On the state of the art of evaluation in neural language models. *arXiv preprint arXiv:1707.05589*.

Tomáš Mikolov, Ilya Sutskever, Anoop Deoras, Hai-Son Le, Stefan Kombrink, and Jan Cernocky. 2012. Subword language modeling with neural networks. *preprint (http://www. fit. vutbr. cz/imikolov/rnnlm/char. pdf)*, 8.

Bhaskar Mitra, Milad Shokouhi, Filip Radlinski, and Katja Hofmann. 2014. On user interactions with query auto-completion. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 1055–1058. ACM.

Yasumasa Miyamoto and Kyunghyun Cho. 2016. Gated word-character recurrent language model. *arXiv preprint arXiv:1606.01700*.

Rodrigo Nogueira and Kyunghyun Cho. 2017. Task-oriented query reformulation with reinforcement learning. *arXiv preprint arXiv:1704.04572*.

Dae Hoon Park and Rikio Chiba. 2017. A neural language model for query auto-completion. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1189–1192. ACM.

Greg Pass, Abdur Chowdhury, and Cayley Torgeson. 2006. A picture of search. In *InfoScale*, volume 152, page 1.

Ofir Press and Lior Wolf. 2016. Using the output embedding to improve language models. *arXiv preprint arXiv:1608.05859*.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *nature*, 323(6088):533.

Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. 2016. Recurrent dropout without memory loss. *arXiv preprint arXiv:1603.05118*.

Rico Sennrich. 2016. How grammatical is character-level neural machine translation? assessing mt quality with contrastive translation pairs. *arXiv preprint arXiv:1612.04629*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Leslie N Smith. 2018. A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay. *arXiv preprint arXiv:1803.09820*.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 553–562. ACM.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. 2018. Blockwise parallel decoding for deep autoregressive models. In *Advances in Neural Information Processing Systems*, pages 10106–10115.

Nako Sung, Minkyu Kim, Hyunwoo Jo, Youngil Yang, Jingwoong Kim, Leonard Lausen, Youngkwan Kim, Gayoung Lee, Donghyun Kwak, Jung-Woo Ha, et al. 2017. Nsml: A machine learning platform that enables you to focus on your models. *arXiv preprint arXiv:1712.05902*.

Saúl Vargas, Roi Blanco, and Peter Mika. 2016. Term-by-term query auto-completion for mobile search. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, pages 143–152. ACM.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Chong Wang, Yining Wang, Po-Sen Huang, Abdelrahman Mohamed, Dengyong Zhou, and Li Deng. 2017. Sequence modeling via segmentations. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3674–3683. JMLR. org.

Po-Wei Wang, J Zico Kolter, Vijai Mohan, and Inderjit S Dhillon. 2018. Realtime query completion via deep language models. *SIGIR eCom*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

# Enhancing Dialogue Symptom Diagnosis with Global Attention and Symptom Graph

**Qin Chen[1§], Xinzhu Lin[1§], Xiahui He[1], Huaixiao Tou[2], Ting Chen[3], Zhongyu Wei[1*]**

[1]School of Data Science, Fudan University
[2]Alibaba Group, China
[3]Ping An Technology, China
{qin_chen,18210980055,18210980050,17210980029,zywei}@fudan.edu.cn
huaixiao.thx@alibaba-inc.com

## Abstract

Symptom diagnosis is a challenging yet profound problem in natural language processing. Most previous research focus on investigating the standard electronic medical records for symptom diagnosis, while the dialogues between doctors and patients that contain more rich information are not well studied. In this paper, we first construct a dialogue symptom diagnosis dataset based on an online medical forum with a large amount of dialogues between patients and doctors. Then, we provide some benchmark models on this dataset to boost the research of dialogue symptom diagnosis. In order to further enhance the performance of symptom diagnosis over dialogues, we propose a global attention mechanism to capture more symptom related information, and build a symptom graph to model the associations between symptoms rather than treating each symptom independently. Experimental results show that both the global attention and symptom graph are effective to boost dialogue symptom diagnosis. In particular, our proposed model achieves the state-of-the-art performance on the constructed dataset.

## 1 Introduction

With the widespread use of electronic health records (EHRs) in medical treatment, symptom diagnosis based on EHRs have received a lot of attention in the natural language processing (NLP) research community (Linder et al., 2007; Shivade et al., 2013). Previous work on EHRs achieved great success in determining the diagnosis of clinical depression (Trinh et al., 2011), identifying community-acquired pneumonia (DeLisle et al., 2013), improving medication reconciliation (Persell et al., 2018) and infection detection (Tou et al., 2018).

§ Contributed equally
* Corresponding author

| **Conversation** |
|---|
| ..... |
| Patient: 孩子50天，嗓子有痰，而且咳嗽，是感冒吗?<br>My kid has been born for 50 days. He has a cough, with phlegm. Does he have a cold? |
| Doctor: 咳嗽，有痰，流黄鼻涕，说明有炎症了。<br>Coughing, sputum and a runny nose, indicate inflammation. |
| Doctor: 咳嗽频繁吗? 发烧吗?<br>Does he cough frequently? Had a fever? |
| Patient: 一阵一阵的，咳起来厉害，不发烧。<br>Occasionally, but his cough is very serious. No fever. |
| ..... |
| **Symptom Diagnosis** |
| True: .....有痰 (Phlegm) 咳嗽 (Cough)<br>流黄鼻涕 (Runny nose) 炎症 (Inflammation) ..... |
| False: .....发烧 (Fever)..... |
| Uncertain: .....感冒 (Cold)..... |

Table 1: An example of a doctor-patient dialogue and symptom diagnosis. Underlined phrases are symptom descriptions. *True*, *False*, and *Uncertain* are the inference results for whether the symptom exists in the patient.

However, EHRs usually contained historical information, such as the medical records or health records, which can not well reflect the current symptoms of the patients. In contrast, the dialogues between doctors and patients during the medical consultation process provide many valuable clues for the current symptom diagnosis. Only a few researchers focus on the dialogue between doctors and patients. (Wei et al., 2018) proposed a reinforcement learning based framework for medical dialogue system for automatic diagnosis. As shown in Table 1, a kid has a cough, the doctor asks the patient whether he has a fever. The patient describes his kid's real situation. Some symptoms like coughing appear, but some symptoms like fever don't appear. What's more, some symptoms are uncertain such as cold because doctor can not make a clear judgment at that time. Though the dialogues show great potential in med-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5033–5042,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

ical treatment, symptom diagnosis based on dialogues, namely dialogue symptom diagnosis, have rarely been studied. Moreover, there are no public datasets on dialogue symptom diagnosis as far as we know.

In this paper, we focus on the studies of dialogue symptom diagnosis and define it by two subtasks: symptom recognition and symptom inference. The symptom recognition aims to identify the symptom related entities from the dialogues, which is the basic step in finding symptoms or diseases. Symptom recognition is similar with the disease named entity recognition (NER) (Doğan et al., 2014) task that is generally considered as a sequence labeling problem (Chinchor and Robinson, 1997; Sang and De Meulder, 2003). Whereas, symptom recognition in dialogues is more challenging due to the short texts and nonstandard oral description. Regarding to the symptom inference, it focuses on making further decisions whether the symptom is *True*, *False*, or *Uncertain* with the patients, which helps the doctors diagnose the disease better in the next step.

To promote the research of dialogue symptom diagnosis, we collect a large amount of dialogues between patients and doctors from a Chinese online medical forum, and construct a dataset for the above two sub-tasks in dialogue symptom diagnosis. In addition, we provide several classical and advanced baselines on this dataset for further research. Furthermore, we propose an approach, which embeds a global attention and symptom graph to improve the performance of dialogue symptom diagnosis. Specifically, the global attention aims at incorporating more related information from the whole dialogue and corpus for better symptom entity representation, which will be used for symptom recognition and inference. Regarding to the symptom graph, it is built by treating each symptom as a node, and the edges are connected according to the true co-occurrence in the dialogues. We build the symptom graph to model the associations between symptoms rather than treating each symptom independently to improve the inference precision.

The contributions of this work can be summarized as follows:

- We provide a public dataset to promote the research of dialogue symptom diagnosis, which contains the annotation results in dialogues with respect to symptom recognition and symptom inference.
- We present a global attention mechanism, which captures more symptom related information from both dialogues and corpus to boost the performance of dialogue symptom diagnosis.
- We build a symptom graph to model the associations between symptoms, which further helps improve the precision of symptom inference.
- We perform extensive experiments, and the experimental results demonstrate the effectiveness of our proposed approach in the two sub-tasks of dialogue symptom diagnosis.

## 2 Related Work

Early attempts on biomedical NER task were based on rule-based dictionary matching method and machine learning method. (Lin et al., 2004) used maximum entropy as the underlying machine learning method incorporated with dictionary-based and rule-based methods for post-processing to identify biomedical entities. (Jimeno et al., 2008) used MetaMap which is provided from the National Library of Medicine and a dictionary matching method to identify diseases.

In recent years, researchers had proposed many neural network-based models on this problem. Most models use encoder-decoder architecture. (Collobert et al., 2011) used the convolutional neural network (CNN) as an encoder, and the conditional random field (CRF) (Lafferty et al., 2001) as a decoder. More recent works used LSTM as encoder which performed better in sequential problems, (Huang et al., 2015) used bidirectional LSTM as encoder, and the BiLSTM-CRF model achieved state-of-the-art on many datasets. Therefore, many researchers chose BiLSTM-CRF model as a baseline model when solving sequential problems. Some researchers made attempts to get better word representation. (Ma and Hovy, 2016) used an additional CNN to represent character-level features on the basis of BiLSTM-CRF. With character encoder, it can extract features inside words and get better representations.

In task of symptom NER, some symptom names entities are complex. There were many efforts to exploit features beyond individual sequences. (Yaghoobzadeh and Schütze, 2016) used knowledge base and aggregated corpus-level contextual

| Patient | |
|---|---|
| Content | My kid has been born for 50 days. He has a **cough**, with **phlegm**. Does he have a **cold**? |
| | 孩 子 5 0 天 ， 嗓 子 **有 痰** ， 而 且 **咳 嗽** ， 是 **感 冒** 吗 ？ |
| Recognition | O O O O O O O O B I O O O B I O O B I O O |
| Normalization | **痰** **咳嗽** **普通感冒** |
| Inference | True True Uncertain |
| Doctor | |
| Content | **Coughing**, **sputum** and **a runny nose**, indicate **inflammation**. |
| | **咳 嗽** ， **有 痰** ， **流 黄 鼻 涕** ， 说 明 有 **炎 症** 了 。 |
| Recognition | B I O B I O B I I I O O O O B I O O |
| Normalization | **咳嗽** **痰** **鼻流涕** **发炎** |
| Inference | True True True True |

Figure 1: An example utterance with annotations of symptoms in BIO format (Symptom entities are in bold).

information to learn an entity's classes. To address the challenges of identifying rare and complex disease names, (Xu et al., 2019) proposed a method that incorporates both disease dictionary matching and a document-level attention mechanism into BiLSTM-CRF for disease NER. (Xu et al., 2018) used the document-level attention mechanism to capture long-range contextual dependencies and address clinical NER tasks. Symptom recognition is a very important step, but these researchers focus on symptom recognition only and do not further infer the recognized symptoms.

## 3 Dataset

In this section, we make a description of our dataset. We have constructed a Chinese dataset from the pediatric department of a Chinese online health community[1]. Patients can submit their health problems and then doctors start a conversation to know more about the patient and provide professional suggestions.

**Annotation** Symptoms reflect the abnormal state of the patient or the presence of the disease. The annotation consists of three parts, namely symptom recognition, symptom normalization and symptom inference. Figure 1 gives an example. We apply BIO (begin-in-out) schema at character level and each symptom is tagged with an extra label (*True*, *False* and *Uncertain*) which indicates whether the patient really has the symptom. Each symptom also links to the most relevant one on SNOMED CT[2] for normalization. In order to ensure the quality of the dataset, we hired three annotators with medical background. Each character is marked by two annotators and the inconsistent part is further judged by the third annotator. The Cohen's kappa coefficient (Fleiss and Cohen, 1973) among the annotators are between 91.80% and 92.71%.

| Symptoms | Count | Ratio(%) |
|---|---|---|
| upper respiratory infection | 480 | 23.22 |
| functional dyspepsia | 485 | 23.46 |
| infantile diarrhea | 546 | 26.42 |
| bronchitis | 556 | 26.90 |
| Total | 2,067 | 100.00 |

Table 2: Symptom distributions.

| Description of the whole dataset | Avg | Std |
|---|---|---|
| # of sentence in each conversation | 42.09 | 13.51 |
| # of SNE in each conversation | 15.14 | 9.53 |
| # of character in each conversation | 544.45 | 276.86 |
| # of character in each sentence | 6.47 | 14.18 |

Table 3: Statistics of the dataset.

**Data Details** Our dataset has a total of 2,067 conversations and we focus on four diseases, namely, "upper respiratory infection", "functional dyspepsia", "infantile diarrhea" and "bronchitis". The distribution of the diseases is shown in Table 2. Table 3 presents some statistics of the dataset. SNE stands for symptom named entity. Besides, the proportion of symptom status as *True*, *False* and *Uncertain* is around 63%, 12% and 25%. In order to get a reasonable comparison, we split the dataset by a 3:1:1 ratio to obtain the training set, validation set and test set[3].

[1] http://muzhi.baidu.com

[2] https://www.snomed.org/snomed-ct

[3] The dataset is available at: www.sdspeople.fudan.edu.cn/zywei/data/emnlp2019-cmdd.zip

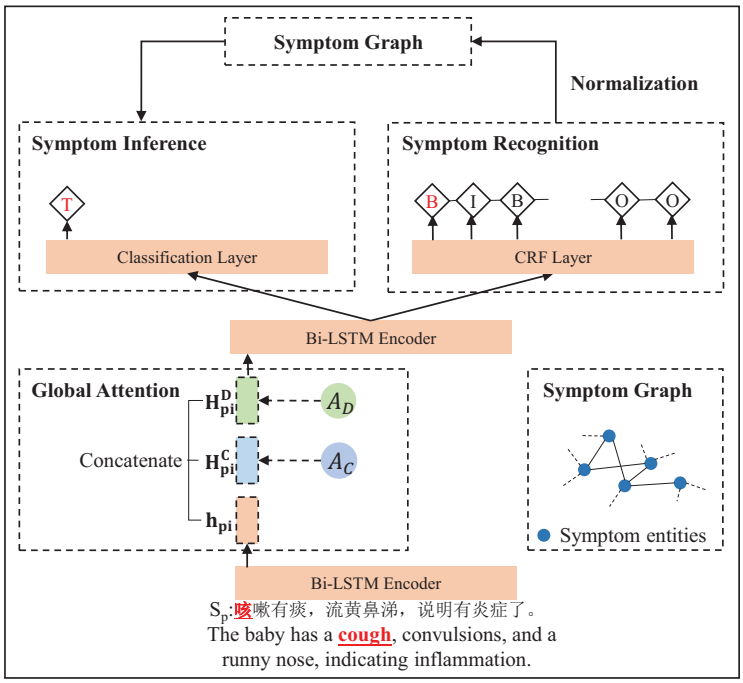


Figure 2: Architecture of our dialogue symptom diagnosis model with global attention and symptom graph. $A_D$ and $A_C$ denote document-level and corpus-level attention respectively.

## 4 Proposed Model

The framework of our proposed model is presented in Figure 2. Our model consists of three parts, the first part is symptom recognition, the second part is symptom graph, and the third part is symptom inference. We first encode the word sequence by Bi-LSTM. Then we present a global attention mechanism to get the contextual information from document level and corpus level. Next, we re-encode the hidden states obtained above and decode by CRF to recognize the symptoms. To model the associations between disease entities in the dialogue, we construct a symptom graph, which is then incorporated into the classification layer for symptom inference. The detailed description of each step is shown in the following sections.

### 4.1 Bi-LSTM Encoder

In this work, we use the bidirectional long short-term memory network (Bi-LSTM) (Hochreiter and Schmidhuber, 1997) to encode the input sequences. Bi-LSTM has been widely-used to extract contextual text features. Bi-LSTM encodes the input from left to right and the same sequence in reverse (Huang et al., 2015). Given input sequence $X = (x_1, x_2, ..., x_n)$, we can get the hidden states $H = (h_1, h_2, ..., h_n)$ where $h_t$ =Bi-LSTM$(x_t)$. Formally, the basic units including hidden state $h_t$ and the memory $c_t$ are updated with following equations:

$$\begin{aligned}
\mathbf{f}_t &= \sigma(\mathbf{W}_f[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f) \\
\mathbf{i}_t &= \sigma(\mathbf{W}_i[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i) \\
\widetilde{\mathbf{c}}_t &= tanh(\mathbf{W}_c[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_c) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \widetilde{\mathbf{c}}_t \\
\mathbf{o}_t &= \sigma(\mathbf{W}_o[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o) \\
\mathbf{h}_t &= \mathbf{o}_t \odot tanh(\mathbf{c}_t)
\end{aligned} \tag{1}$$

where $\sigma$ is the sigmoid function and $\odot$ is the element-wise product. $\mathbf{x}_t$ is the input vector at time $t$. $\mathbf{i}_t$, $\mathbf{f}_t$, $\mathbf{o}_t$ denote the input, forget and output gate respectively.

### 4.2 Global Attention

Our global attention mechanism is shown in Figure 3, which consists of two parts, namely document-level attention and corpus-level attention. We will describe the details in the following.

**Document-level Attention** In a dialogue, the information provided by a single sentence is very limited and the same word may indicate different meanings due to the ambiguity. Therefore, we apply the document-level attention mechanism to make full use of the information in the whole dialogue to alleviate the ambiguity problem.

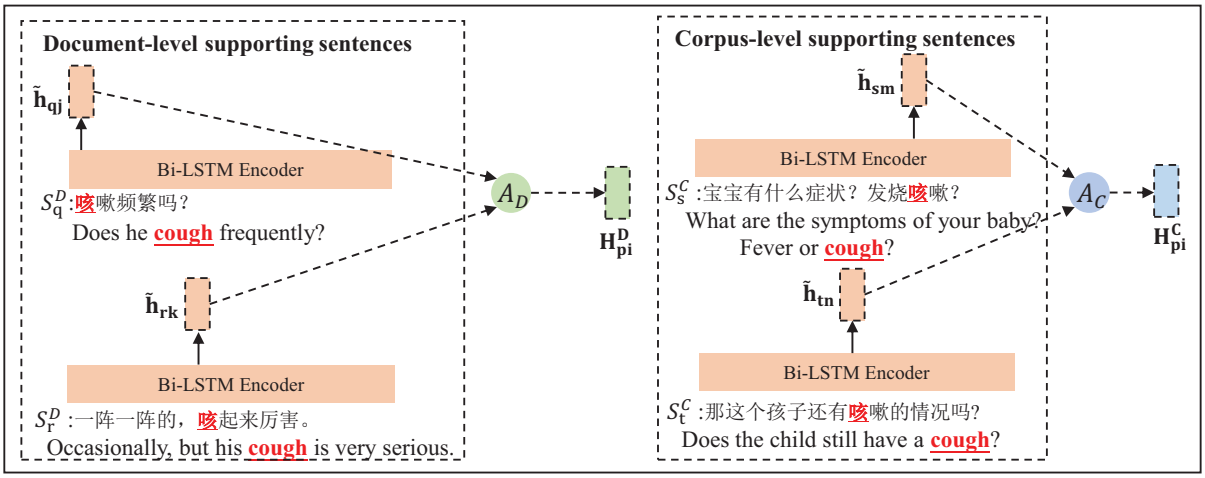


Figure 3: Global attention which consists of the document-level and corpus-level attentions. $A_D$ and $A_C$ denote document-level and corpus-level attention respectively.

We define a document (or dialogue) $D = (S_1, S_2, ...)$ and a sentence $S_p = (w_{p1}, w_{p2}, ...)$. $S_p$ represents the $p$th sentence of the document and $w_{pi}$ represents the $i$th word of $S_p$. $\mathbf{h}_{pi}$ is the hidden state of $w_{pi}$. We search for the sentence with the same word $w_{pi}$ from the current document, and feed the found sentence into the same Bi-LSTM model. For example, as shown in Figure 3, $w_{pi}$ represents the word "cough". The sentences as $S_q^D$ and $S_r^D$ in the current dialogue also contain "cough" . We add the hidden states of the word in the two sentences into a set $\tilde{\mathbf{h}}_{pi}$=$\{\tilde{\mathbf{h}}_{pi}^1, \tilde{\mathbf{h}}_{pi}^2, ...\}$. In Figure 3, $\tilde{\mathbf{h}}_{qj}$ and $\tilde{\mathbf{h}}_{rk}$ are $\tilde{\mathbf{h}}_{pi}^1$ and $\tilde{\mathbf{h}}_{pi}^2$ respectively. We weight the hidden states by document-level attention and the attentive representation is formulated as follows:

$$\begin{aligned} \mathbf{e}_{pi}^{D,j} &= \mathbf{v}^T \tanh(\mathbf{W}_h \mathbf{h}_{pi} + \mathbf{W}_{\tilde{h}} \tilde{\mathbf{h}}_{pi}^j + \mathbf{b_e}) \\ \boldsymbol{\alpha}_{pi}^D &= \mathbf{Softmax}(\mathbf{e}_{pi}^D) \\ \mathbf{H}_{pi}^D &= \sum_{j=1}^{k} \alpha_{pi}^{D,j} \tilde{\mathbf{h}}_{pi}^j \end{aligned} \tag{2}$$

$\mathbf{v}$, $\mathbf{W}_h$, $\mathbf{W}_{\tilde{h}}$ and $\mathbf{b}_e$ are the parameters to be learned. $\mathbf{H_{pi}^D}$ denotes the contextual information of the word $w_{pi}$ in the dialogue.

**Corpus-level Attention** Noting that the same word in different dialogues may indicate additional associations, we devise a corpus-level attention mechanism to capture the extra information.

We define the corpus $C = \{D_1, D_2, D_3, ...\}$. Similar to the document-level attention, we find the supporting sentences in the corpus that contain the current word. In Figure 3, the sentences as $S_s^C$ and $S_t^C$ contain the word "cough", and $\tilde{\mathbf{h}}_{sm}$ and $\tilde{\mathbf{h}}_{tn}$ are the corresponding hidden states. We apply corpus-level attention to obtain the attentive representation of the hidden states in the corpus:

$$\mathbf{H}_{pi}^C = \sum_{j=1}^{k} \alpha_{pi}^{C,j} \tilde{\mathbf{h}}_{pi}^j \tag{3}$$

$\mathbf{H}_{pi}^C$ denotes the related information of the word $w_{pi}$ in the corpus, $\alpha_{pi}^{C,j}$ is the attention weight for the corresponding hidden state in the corpus.

**Both Document and Corpus-level Attention** In order to integrate the information obtained based on document-level attention and corpus-level attention, we concatenate $\mathbf{h}_{pi}$, $\mathbf{H}_{pi}^C$ and $\mathbf{H}_{pi}^D$, and feed it into another Bi-LSTM model. Thus, the final hidden state of each word contains the complementary information from both the dialogue and corpus.

### 4.3 Symptom Recognition

In this work, we apply the Conditional Random Field (CRF) (Lafferty et al., 2001) as decoder for symptom recognition. CRF can compute the global optimal sequence and efficiently capture the dependencies among tags (e.g. label 'I' can not follow 'O') via jointly decoding the chain of labels. The Viterbi algorithm (Viterbi, 1967) is chosen for inference using dynamic programming. Given the representation of a sequence, we first map it to the tag space by a linear layer. Then, the score of the input along with a prediction y is given by:

$$s(X, y) = \sum_{i=1}^{n} (T_{y_{i-1}, y_i} + P_{i, y_i}) \tag{4}$$

where T is a transition matrix, $T_{i,j}$ represents the score of transition from tag i to tag j; P is a matrix of the output from the last layer, $P_{i,j}$ is the score of the $j^{th}$ tag for the $i^{th}$ word in the sentence. The goal is to predict the best tag path that given by:

$$y = argmax_{\widetilde{y}} s(X, \widetilde{y}) \tag{5}$$

### 4.4 Symptom Graph

Symptom entities have a certain probability of co-occurrence in a dialogue. For example, "fever" may appear in a conversation with "cold" at the same dialogue, and "cough" may appear in a conversation with "sputum" at the same dialogue. In order to capture the associations between the disease entities, we build a graph $G = (V, E)$, where $V = v_1, v_2, ..., v_m$ is the node set and $E \subset V \times V$ is the edge set. The edges $e_{i,j} = (v_i, v_j)$ in the graph is undirected. The nodes are the normalized symptom entities with status *True* from the training corpus. Edge $e_{i,j} = (v_i, v_j)$ indicates symptom entities $v_i$ and $v_j$ co-occur in a document. The co-occurrence number of two entities is normalized by min-max normalization to obtain the edge weight. Thus, the weight $w_{i,j} \in (0, 1)$.

### 4.5 Symptom Inference

Intuitively, the associations between symptom entities can help enhance symptom inference. Therefore, we first define a smoothness loss that quantitatively measures the entity associations in the constructed symptom graph:

$$S = \frac{1}{2} \sum_{i,j} w_{i,j} (y_i - y_j)^2 = \mathbf{y}^T L \mathbf{y} \tag{6}$$

where $y_i$ is 0 or 1 that depends whether the entity has been recognized by the symptom recognition module. $L = D - A$ denotes the Laplacian matrix of an undirected subgraph $G'$ with $k$ nodes and $m$ edges corresponding to the current document. $A \in R^{k \times k}$ denotes the weighted adjacency matrix of subgraph $G'$. $D$ is a degree matrix and $D_{ii} = \sum_j A_{ij}$. $\mathbf{y}^T$ is a k-dimensional vector. Theoretically, if the model fully recognizes the symptoms, the value of S is 0 indicating the symptom graph is smooth. When some symptoms are not recognized, the value of S depends on the weights between the nodes of the incorrect symptoms and the neighbor nodes.

With the smoothness loss defined above, we then incorporate it into the loss function for symptom inference. Symptoms are classified into three categories: *True*, *False*, or *Uncertain*, and we adopt a softmax function in the classification layer to predict the probability of the symptom belonging to each category. The classification layer and the CRF layer share the same hidden states of the upper Bi-LSTM encoder. The objective is defined as minimizing the joint cross-entropy loss in classification and the smoothness loss in the graph:

$$loss = -\sum_{\mathcal{D}} (\sum_i \sum_j \log p^c_{i,j} - \lambda S) \tag{7}$$

where $\mathcal{D}$ is the document set, $i$ is the index of the sentence, $j$ is the index of the symptom, $p^c_{i,j}$ is the predicted probability of the gold-standard polarity class $c$ for the $j$th symptom in the $i$th sentence of the document, and $\lambda$ is a weight parameter to control the importance of the smoothness loss.

## 5 Experiments and Results

### 5.1 Experimental Setup

We use 200-dimensional Chinese embeddings trained on Wikipedia and fine tune them during model training by back-propagating the gradients. The parameters of the weight matrix are initialized by the Xavier method (Glorot and Bengio, 2010). The Stochastic gradient descent (SGD) method with a momentum of 0.9 is used for optimization. The initial learning rate $\eta_0 = 0.015$ and the learning rate gradually decreases with the increasing epoch. The specific update formula is $\eta_t = \eta_0/(1 + \rho t)$ where $\rho = 0.05$ and t is the number of trained epoch. Gradient clipping is set to 5 in order to avoid "gradient exploding". Other experimental settings such as the dropout rate is 0.5 and the Bi-LSTM hidden dimension is 200. We build the symptom graph from our training set. There are 1,646 edges and 162 nodes. We initialize the labels as 1 for all nodes.

**Look-up Table and Stop Words** For the attention mechanism, we select at most three document-level supporting sentences and three corpus-level supporting sentences. We build a look-up table that can quickly get the index of a word in each sentence and the index of a sentence in each document. Therefore, the time complexity of finding supporting sentences and words is O(1). Meanwhile, we use a stop-word list that contains 178 words. In this way, we can further reduce the time cost.

**Symptom Normalization** We consider the symptom normalization as a text classification

problem. In our dataset, there are 162 normalized symptoms. We apply the convolution neural network (CNN) to classify all symptoms into 162 categories. The accuracy of symptom normalization on the test set is 97.04%. Thus, the symptom normalization doesn't bring too much noise to our model.

### 5.2 Performance of Symptom Recognition

Symptom recognition is the basis of symptom inference. We report the results of the recent advanced baselines as well as the variants of our proposed method. Specifically, we compare the performance of the following models:

- **Bi-RNNs (Dyer et al., 2015)**: The models use LSTM or GRU for the sentence encoder, and treat symptom entity recognition as a classification problem with the softmax function. From now on, we use RNNs to denote LSTM or GRU for ease of description.
- **Bi-RNNs-CRF (Huang et al., 2015)**: The models use RNNs for the sentence encoder and a CRF layer for decoder, which yields the tagging prediction for each token.
- **CNNs-Bi-RNNs-CRF (Ma and Hovy, 2016)**: Compared with Bi-RNNs-CRF, the CNNs-Bi-RNNs-CRF models additionally incorporate the character level information with CNN for encoder.
- **Corpus-level Attention**: It is a Bi-LSTM-CRF model that incorporates corpus-level features via our corpus-level attention.
- **Document-level Attention**: It is a Bi-LSTM-CRF model that incorporates document-level features via our document-level Attention.
- **Both Corpus and Document-level Attention**: It is a Bi-LSTM-CRF model that incorporates both document-level and corpus-level features via our global attention.

The overall results of symptom recognition are shown in Tabel 4. We observe that the Bi-RNNs models including Bi-GRU and Bi-LSTM have the similar performance, which get about 81% in the F1 score on our dataset. The Bi-RNNs-CRF models perform much better than Bi-RNNs, which indicates the effectiveness of the CRF model for sequence tagging. In addition, the performance can be slightly improved by incorporating the character level information with CNN. Furthermore, by integrating either our corpus-level attention or document-level attention into the existing models, the performance can be significantly boosted. In particular, our model with global attention achieves the best performance in terms of all metrics.

| Model | Prec. | Recall | F1 |
|---|---|---|---|
| Bi-GRU | 76.02% | 88.09% | 81.61% |
| Bi-LSTM | 76.64% | 87.60% | 81.62% |
| Bi-GRU-CRF | 86.44% | 89.13% | 87.77% |
| Bi-LSTM-CRF | 89.93% | 89.56% | 89.74% |
| CNNs-Bi-GRU-CRF | 87.08% | 90.82% | 88.91% |
| CNNs-Bi-LSTM-CRF | 90.45% | 90.48% | 90.47% |
| Corpus-level attention | 90.40% | 91.02% | 90.71% |
| Document-level attention | 90.53% | 91.67% | 91.10% |
| Both Corpus and Document-level attention | 91.09% | 92.17% | 91.62% |

Table 4: Performance of various models for symptom recognition.

| Method | inference of symptom | F1 |
|---|---|---|
| Bi-LSTM CRF-inference | True | 82.68% |
| | False | 59.22% |
| | Uncertain | 65.02% |
| Bi-LSTM CRF-inference with graph | True | 83.79% |
| | False | 60.04% |
| | Uncertain | 65.80% |
| Our joint model | True | 85.08% |
| | False | 66.09% |
| | Uncertain | 74.13% |
| Our joint model with graph | True | 86.46% |
| | False | 66.88% |
| | Uncertain | 74.25% |

Table 5: Performance of various models for symptom inference.

### 5.3 Performance of Symptom Inference

Table 5 presents the symptom inference results of the classical Bi-LSTM CRF-inference model and our proposed joint model (Figure 2). The results show that our proposed model with global attention significantly outperforms the Bi-LSTM CRF-inference model for symptom inference across all the categories. In particular, we achieve substantial improvements for inferring the *False* and *Uncertain* categories of symptoms, by utilizing the global information in the current dialogue and the whole corpus

To investigate the effect of the symptom graph for symptom inference, we compare the models with and without symptom graphs. The results in Table 5 show that when incorporating the symptom graphs for inference, the performance of each model can be further boosted. These observations have verified the effectiveness of modeling

the associations between symptoms via graphs for symptom inference.

### 5.4 Qualitative Analysis

Table 6 presents a case of symptom recognition based on the baseline and our model. It is observed that the baseline Bi-LSTM CRF model only identifies the word "allergic" as a symptom. In contrast, our model can recognize the phrase "allergic rhinitis" by utilizing the related information (i.e., "allergies rhinitis" and "allergic caused rhinitis") in the document-level and corpus-level supporting sentences, which is more accurate for the symptom description in this case.

Table 7 shows the results of symptom inference for a case by using the baseline and our joint model. From the patient's answer, we know that the kid has no "allergy". Whereas, the symptom "allergies" in the doctor's question sentence is inferred as *uncertain* by the baseline. By incorporating the global attention mechanism, our joint model can correctly infer the symptom as *false*.

| Model | Sentence |
|---|---|
| Bi-LSTM CRF | 医生：相对来说，这个年龄的孩子出现过敏性鼻炎比较少见。<br>Doctor: Relatively speaking, allergic rhinitis is rare in children of this age. |
| Our Model | 医生：相对来说，这个年龄的孩子出现过敏性鼻炎比较少见。<br>Doctor: Relatively speaking, allergic rhinitis is rare in children of this age. |
| Document-level supporting sentence | 患者：如果是过敏性的鼻炎，会持续多久?<br>Patient: If it is allergic caused rhinitis , how long will it last? |
| Corpus-level supporting sentence | 医生：那需要考虑过敏性鼻炎的可能。<br>Doctor: It may be allergic rhinitis. |

Table 6: Symptom recognition results of the baseline and our methods. Underlined phrases are symptoms.

| Model | Sentence |
|---|---|
| Bi-LSTM CRF-Inference | 医生：孩子小时候湿疹重不重? 平时易过敏吗?<br>Doctor: Is the child eczema serious? Is he susceptible to allergies in daily life?<br>Inference:Uncertain Uncertain<br>患者：不过敏啊。<br>Patient: No allergy.<br>Inference:False |
| Our joint model with graph | 医生：孩子小时候湿疹重不重? 平时易过敏吗?<br>Doctor: Is the child eczema serious? Is he susceptible to allergies in daily life?<br>Inference:Uncertain False<br>患者：不过敏啊。<br>Patient: No allergy.<br>Inference:False |

Table 7: Symptom inference results of the Bi-LSTM CRF-inference model and our joint model with symptom graph. Underlined phrases are symptoms.

To have an insight of why the symptom graph can help boost symptom inference, we select several frequent symptoms in dialogues, namely "Cough", "Sputum", "Fever", "Diarrhea", "Snot", "Cold" and "Indigestion", and visualize the associations between the symptoms in Figure 4. The darker color indicates a larger association weight between the symptoms. We observe that the "cough" and "sputum" are highly associated, which corresponds to our intuition that the patient will probably have a cough and sputum simultaneously. To make it more clear, we show the inference results for each symptom with and without graph in Figure 5. The results show that our model with graph achieves larger improvements than that without graph for inferring the highly associated symptoms such as "cough" and "sputum", which indicates the necessity to incorporate the symptom graph to enhance symptom diagnosis.

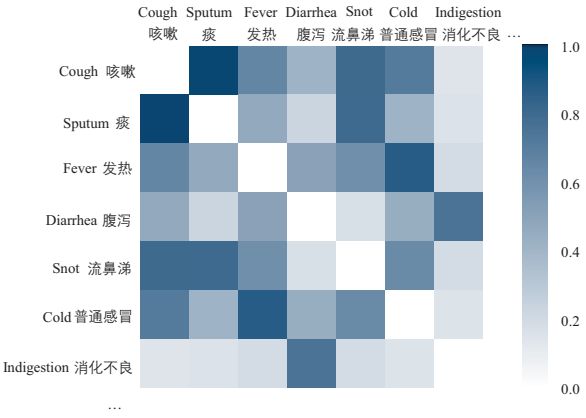


Figure 4: Symptom associations in the symptom graph.

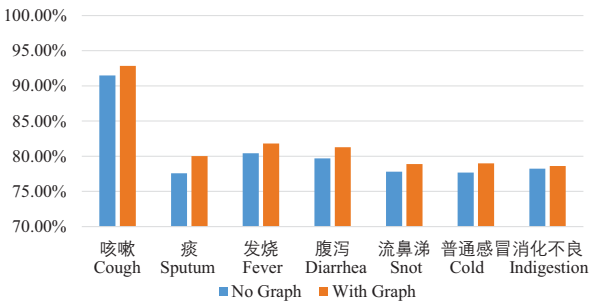


Figure 5: The impact of the symptom graph on F1 scores for symptom inference.

## 6 Conclusions and Future Works

In this paper, we construct a dataset for dialogue symptom diagnosis, and present a model with global attention and symptom graph for diagnosing symptoms in dialogues. Our global attention mechanism consists of the document-level

and corpus-level attentions, which select supporting sentences from the current dialogue and corpus to overcome the information limitations. Experiments on our dataset show that our global attention can effectively boost the performance of dialogue symptom diagnosis. Furthermore, we build a symptom graph to model the associations between symptoms, which helps improve the performance of symptom inference.

In the future, we will build a larger symptom graph and use external medical information to further improve the performance of symptom diagnosis on dialogues.

## Acknowledgments

This work is partially funded by National Natural Science Foundation of China (No. 61751201), National Natural Science Foundation of China (No. 61702106) and Shanghai Science and Technology Commission (No. 17JC1420200, No. 17YF1427600 and No. 16JC1420401).

## References

Nancy Chinchor and Patricia Robinson. 1997. Muc-7 named entity task definition. In *Proceedings of the 7th Conference on Message Understanding*, volume 29, pages 1–21.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(Aug):2493–2537.

Sylvain DeLisle, Bernard Kim, Janaki Deepak, Tariq Siddiqui, Adi Gundlapalli, Matthew Samore, and Leonard D'Avolio. 2013. Using the electronic medical record to identify community-acquired pneumonia: toward a replicable automated strategy. *PLoS One*, 8(8):e70944.

Rezarta Islamaj Doğan, Robert Leaman, and Zhiyong Lu. 2014. Ncbi disease corpus: a resource for disease name recognition and concept normalization. *Journal of biomedical informatics*, 47:1–10.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-based dependency parsing with stack long short-term memory. *arXiv preprint arXiv:1505.08075*.

Joseph L Fleiss and Jacob Cohen. 1973. The equivalence of weighted kappa and the intraclass correlation coefficient as measures of reliability. *Educational and psychological measurement*, 33(3):613–619.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Antonio Jimeno, Ernesto Jimenez-Ruiz, Vivian Lee, Sylvain Gaudan, Rafael Berlanga, and Dietrich Rebholz-Schuhmann. 2008. Assessment of disease named entity recognition on a corpus of annotated sentences. In *BMC bioinformatics*, volume 9, page S3. BioMed Central.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Yi-Feng Lin, Tzong-Han Tsai, Wen-Chi Chou, Kuen-Pin Wu, Ting-Yi Sung, and Wen-Lian Hsu. 2004. A maximum entropy approach to biomedical named entity recognition. In *Proceedings of the 4th International Conference on Data Mining in Bioinformatics*, pages 56–61. Springer-Verlag.

Jeffrey A Linder, Jun Ma, David W Bates, Blackford Middleton, and Randall S Stafford. 2007. Electronic health record use and the quality of ambulatory care in the united states. *Archives of internal medicine*, 167(13):1400–1405.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*.

Stephen D Persell, Kunal N Karmali, Danielle Lazar, Elisha M Friesema, Ji Young Lee, Alfred Rademaker, Darren Kaiser, Milton Eder, Dustin D French, Tiffany Brown, et al. 2018. Effect of electronic health record–based medication support and nurse-led medication therapy management on hypertension and medication self-management: a randomized clinical trial. *JAMA internal medicine*, 178(8):1069–1077.

Erik F Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*.

Chaitanya Shivade, Preethi Raghavan, Eric Fosler-Lussier, Peter J Embi, Noemie Elhadad, Stephen B Johnson, and Albert M Lai. 2013. A review of approaches to identifying patient phenotype cohorts using electronic health records. *Journal of the American Medical Informatics Association*, 21(2):221–230.

Huaixiao Tou, Lu Yao, Zhongyu Wei, Xiahai Zhuang, and Bo Zhang. 2018. Automatic infection detection based on electronic medical records. *BMC bioinformatics*, 19(5):117.

Nhi-Ha T Trinh, Soo Jeong Youn, Jessica Sousa, Susan Regan, C Andres Bedoya, Trina E Chang, Maurizio Fava, and Albert Yeung. 2011. Using electronic medical records to determine the diagnosis of clinical depression. *International journal of medical informatics*, 80(7):533–540.

Andrew Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*, 13(2):260–269.

Zhongyu Wei, Qianlong Liu, Baolin Peng, Huaixiao Tou, Ting Chen, Xuanjing Huang, Kam-Fai Wong, and Xiangying Dai. 2018. Task-oriented dialogue system for automatic diagnosis. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 201–207.

Guohai Xu, Chengyu Wang, and Xiaofeng He. 2018. Improving clinical named entity recognition with global neural attention. In *Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference on Web and Big Data*, pages 264–279. Springer.

Kai Xu, Zhenguo Yang, Peipei Kang, Qi Wang, and Wenyin Liu. 2019. Document-level attention-based bilstm-crf incorporating disease dictionary for disease named entity recognition. *Computers in biology and medicine*, 108:122–132.

Yadollah Yaghoobzadeh and Hinrich Schütze. 2016. Corpus-level fine-grained entity typing using contextual information. *arXiv preprint arXiv:1606.07901*.

# Counterfactual Story Reasoning and Generation

**Lianhui Qin** ♠♢ **Antoine Bosselut** ♠♢ **Ari Holtzman** ♠♢ **Chandra Bhagavatula** ♢
**Elizabeth Clark** ♠ **Yejin Choi** ♠♢
♠Paul G. Allen School of Computer Science & Engineering, University of Washington
♢Allen Institute for Artificial Intelligence
{lianhuiq,antoineb,ahai,eaclark7,yejin}@cs.washington.edu
chandrab@allenai.org

## Abstract

Counterfactual reasoning requires predicting how alternative events, contrary to what actually happened, might have resulted in different outcomes. Despite being considered a necessary component of AI-complete systems, few resources have been developed for evaluating counterfactual reasoning in narratives.

In this paper, we propose *Counterfactual Story Rewriting*: given an original story and an intervening counterfactual event, the task is to minimally revise the story to make it compatible with the given counterfactual event. Solving this task will require deep understanding of causal narrative chains and counterfactual invariance, and integration of such story reasoning capabilities into conditional language generation models.

We present TIMETRAVEL, a new dataset of 29,849 counterfactual rewritings, each with the original story, a counterfactual event, and human-generated revision of the original story compatible with the counterfactual event. Additionally, we include 81,407 counterfactual "branches" without a rewritten storyline to support future work on semi- or un-supervised approaches to counterfactual story rewriting.

Finally, we evaluate the counterfactual rewriting capacities of several competitive baselines based on pretrained language models, and assess whether common overlap and model-based automatic metrics for text generation correlate well with human scores for counterfactual rewriting.

## 1 Introduction

A desired property of AI systems is counterfactual reasoning: the ability to predict causal changes in future events given a counterfactual condition applied to the original chain of events (Goodman, 1947; Bottou et al., 2013). For example, given an original story shown in the left chain in Figure 1, where "Pierre loved Halloween. He decided to be a *vampire* this year. He got a *black cape and white face paint*..." and a counterfactual condition, "what if Pierre decided to be a *werewolf* instead of a *vampire*?", an intelligent system should be able to revise the subsequent events in the story appropriately, for example, that a *brown sweater* would be more appropriate than a *black cape*.

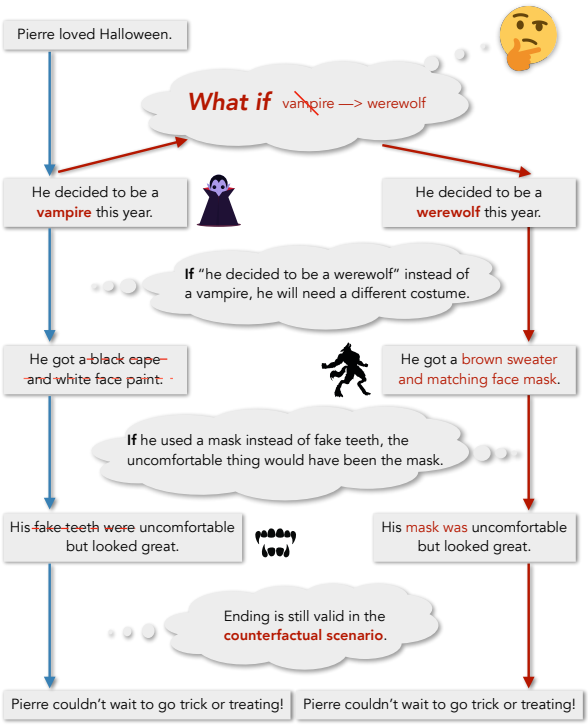


Figure 1: Given a short story (left column) and a *counterfactual context* ("He decided to be a werewolf this year"), the task is to revise the original story with minimal edits to be consistent with both the original premise ("Pierre loved Halloween") and the new counterfactual situation. The modified parts in the new story (right column) are highlighted in red.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5043–5053,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

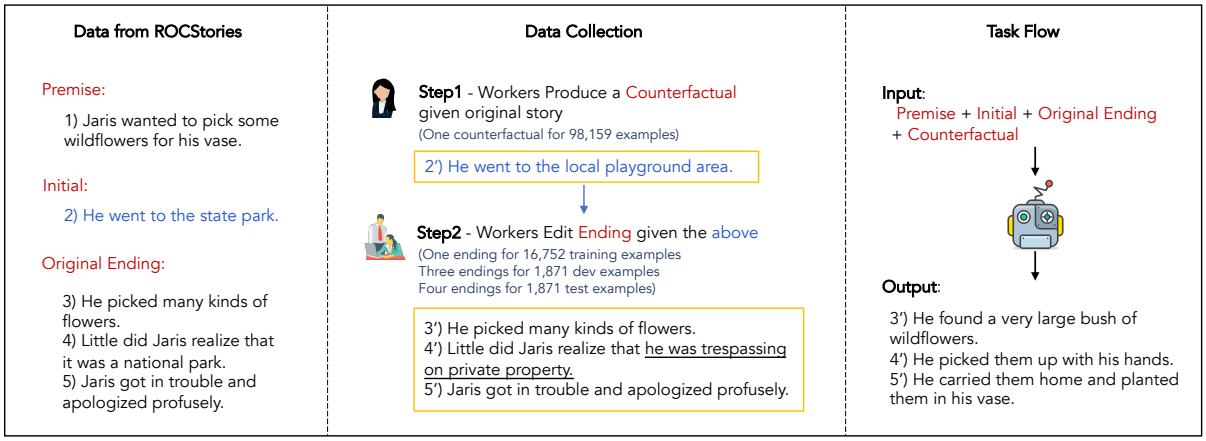


Figure 2: Data annotation process for the TIMETRAVEL dataset. Given a story from the ROCStories corpus, crowdworkers write a counterfactual sentence w.r.t the second sentence of the story. The counterfactual sentence and the original story are then presented to other workers to rewrite the story ending. Models for the task are expected to generate a rewritten ending given the original story and counterfactual sentence.

This notion of counterfactuals has become increasingly relevant in several recent benchmarks such as ROC story cloze (Mostafazadeh et al., 2016), COPA (Roemmele et al., 2011), and HellaSwag (Zellers et al., 2019), where the negative responses in multiple-choice problems implicitly construct counterfactual narratives. However, no existing benchmark to date has been designed to explicitly evaluate counterfactual narrative reasoning and revision as its principal focus, where a system is evaluated on its ability to make modifications to future events based on a counterfactual condition, as illustrated in Figure 1.

In this paper, we introduce *Counterfactual Story Rewriting* as a new challenge to story understanding and generation. Given an original story and a counterfactual condition, the task is to re-write the story to regain narrative consistency through counterfactual reasoning. An important challenge in counterfactual reasoning is *causal invariance*, namely, the aspects of future events that are invariant under the counterfactual conditions. This is necessary to accurately reason about the new consequences with minimal edits to the original sequence of events, instead of being confounded by spurious correlations (Woodward, 2002; Bottou, 2019). Therefore, a key measure of the task besides consistency is that the rewriting must perform *minimal edits* to the original story. This challenges the system to reason about causal invariance, which in turn, challenges the system to reason more carefully about the causal chains of how the story unfolds.

We introduce TIMETRAVEL, a new dataset with 29,849 counterfactual revisions to support research on counterfactual narrative reasoning and revision. In addition, our dataset provides 81,407 counterfactual *branches* without rewritten storylines to support potential future work on semi- or un-supervised approaches. Figure 2 illustrates (1) the structure of the original stories, (2) the counterfactual data construction process, and (3) the final task definition.

We establish baseline performances of state-of-the-art neural language models on this task, such as GPT (Radford et al., 2018) and GPT-2 (Radford et al., 2019), evaluated in zero-shot, unsupervised, and supervised learning settings. Empirical results indicate that while these models are able to capture certain instances of counterfactual reasoning, they generally struggle with rewriting endings with full consistency. Our results suggest that current neural language models operate based primarily on frequent patterns in language without true understanding of the causal chains in narratives, thus requiring more focused future research to integrate reasoning capabilities to neural language models.

## 2 Background

Counterfactual reasoning is the ability to consider *alternative possibilities* that diverge from current observed narratives. Due to their prevalence in common reasoning situations, counterfactuals have been studied in a wide range of disciplines, including psychology (Epstude and Roese, 2008),

cognitive science (Byrne, 2002), as well as natural language processing (Hobbs, 2005; Lawrence and Riezler, 2018; Son et al., 2017).

Meanwhile, despite the progress made in NLU tasks by adapting pretrained language representations such as BERT (Devlin et al., 2018) or GPT (Radford et al., 2018), models still have trouble discriminating between reasonable and unreasonable counterfactuals, as shown in (Zellers et al., 2019). Moreover, success in tasks linked to discrimination of reasonable alternatives often results in models learning to exploit latent artifacts of the dataset (Niven and Kao, 2019; Zellers et al., 2018), rather than learning to robustly reason about counterfactuals. In response to this, we hypothesize that learning to *generate the result* of counterfactual prompts will encourage models to learn to understand the underlying dynamics of a given situation, whereas discrimination between two alternatives is more likely to take advantage of dataset biases.

This goal shares many similarities with script learning (Pichotta and Mooney, 2014; Chambers, 2013), which attempts to canonicalize stereotypical event sequences for learning causal structure of narratives. However, because it is often difficult to capture the richness of causal dependencies with templatized structures (Sap et al., 2019), we instead study counterfactual reasoning in unstructured text directly and also require the model to *generate* the consequences of the counterfactual reasoning.

The "counterfactual event" in our task can be viewed as a causal intervention (Pearl, 2000) in the latent chain of events of the story. Such interventions demand changes to the written narrative in order to abide by the shared background knowledge that human readers have about how the world works. This neatly embeds the problem of causal reasoning in a space that laymen with no knowledge of formalized causality can understand. It also allows us to evaluate the capabilities and limitations of the recent advances in neural language models in the context of counterfactual reasoning.

Similar issues arise in the area of controllable language generation (e.g., Hu et al., 2017), which involves preserving the content of text while changing it along a single or multiple dimensions, such as theme (Koncel-Kedziorski et al., 2016), style (Lample et al., 2019), and sentiment (Shen et al., 2017). Reasoning in these tasks is limited to discrete axes (e.g., sentiment), which are often categorized with a closed label set ({positive, negative}). Because of controllability motivations, these axes and labels are generally known *a priori*. In contrast, counterfactual rewriting focuses on the causes and effects of a story, dimensions that can require more complex and diverse, yet potentially subtle, changes to accommodate the counterfactual event. Additionally, we put no restrictions on the nature of counterfactual events, yielding no clear set of discrete axes along which the story can change and no closed set of labels for them.

## 3 Counterfactual Story Rewriting

### 3.1 Task

We now formally introduce the task and establish the notation used in the paper. Each example consists of a five-sentence story $S = (\boldsymbol{s}_1, \ldots, \boldsymbol{s}_5)$ with a general structure where the first sentence $\boldsymbol{s}_1$ sets up the *premise*, the second sentence $\boldsymbol{s}_2$ provides more information of the *initial context*, and the last three sentences $\boldsymbol{s}_{3:5}$ are the *original ending* of story. We are further given an additional sentence $\boldsymbol{s}'_2$, which is counterfactual to the initial context $\boldsymbol{s}_2$. That is, $\boldsymbol{s}'_2$ states something contrary to that in $\boldsymbol{s}_2$, which in turn can make the original ending $\boldsymbol{s}_{3:5}$ no longer valid. Thus, the goal of the task is to rewrite the ending, such that the *edited ending* $\boldsymbol{s}'_{3:5}$ minimally modifies the original one and regains narrative coherency to the new counterfactual context.

The minimum edit goal differentiates our task from previous story ending studies, which have mostly focused on consistency in a given context. To achieve consistency with minimal edits, a model must understand the key mechanisms that drive the story's narrative so that it can filter out spurious correlations and capture counterfactual invariance. We thus consider the new task as a suitable testbed for studying counterfactual reasoning in combination with language generation.

### 3.2 Dataset: TIMETRAVEL

Our dataset is built on top of the ROCStories corpus (Mostafazadeh et al., 2016), which contains 98,159 five-sentences stories in the training set, along with 3,742 stories in the evaluation sets. Each story was written by

| | |
|---|---|
| **Premise** | Alec's daughter wanted more blocks to play with. |
| **Initial** | Alec figured that blocks would develop her scientific mind. |
| **Original Ending** | Alec bought blocks with letters on them. Alec's daughter made words with them rather than structures. Alec was happy to see his daughter developing her verbal ability. |
| **Counterfactual** | Alec couldn't afford to buy new blocks for his daughter. |
| **Edited Ending** | Alec decided to make blocks with letters on them instead. Alec's daughter made words with the blocks. Alec was happy to see his daughter developing her verbal ability. |
| **Premise** | Ana had just had a baby girl. |
| **Initial** | She wanted her girl to have pierced ears. |
| **Original Ending** | She took her baby to the studio and had her ears pierced. Then she fastened tiny diamond studs into the piercings. Ana loved the earrings. |
| **Counterfactual** | She didn't like the idea of having her ears pierced. |
| **Edited Ending** | She decided not to take her baby to the studio to get her ears pierced. So she took tiny diamond stickers and stuck them to her ear. Ana loved the fake earrings. |

Table 1: Examples from TIMETRAVEL

crowdworkers. To collect counterfactual events and new story continuations for TIMETRAVEL, we employ workers from Amazon Mechanical Turk (AMT) for a two-step task, which we describe in detail below.

### 3.3 Data Collection

**Counterfactual Event Collection** We present workers with an original five-sentence story $S = (\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_5)$ and ask them to produce a counterfactual event $\boldsymbol{s}'_2$ based on $\boldsymbol{s}_2$. Workers are instructed to produce counterfactual sentences $\boldsymbol{s}'_2$ that are:
(1) Topically related to the original context sentence $\boldsymbol{s}_2$, rather than a completely new sentence.
(2) Relevant to the original premise sentence, $\boldsymbol{s}_1$, allowing for a coherent story continuation.
(3) Influential to the subsequent storyline, such that at least one of the original ending's sentences, $\{\boldsymbol{s}_3, \boldsymbol{s}_4, \boldsymbol{s}_5\}$ is no longer appropriate given $\boldsymbol{s}_1$ and $\boldsymbol{s}'_2$, necessitating a rewritten story ending.

**Continuation Rewriting** Once a counterfactual sentence $\boldsymbol{s}'_2$ is provided, we present it to a new set of workers with the original story $S = (\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_5)$. Now that $\boldsymbol{s}'_2$ invalidates the original storyline, workers are instructed to make minimal edits to $\boldsymbol{s}_{3:5}$, such that the narrative is coherent again. Before beginning, workers are instructed to validate whether the counterfactual event satisfies the requirements from the previous stage of the pipeline. If not, we ask them to rewrite the counterfactual again, and the continuation rewriting step is reassigned to a new worker.

**Summary** We summarize the scale of TIMETRAVEL in Table 2. Overall, we collect 16,752 training examples of a counterfactual context and a rewritten ending. We also collect an additional 81,407 counterfactual contexts for the training set with no rewritten ending to support future work in unsupervised learning on this task. For the development and test sets, we gather multiple counterfactual contexts and rewritten endings for *each* example (3 new endings for development and 4 for test). Information regarding quality control and cost are provided in Appendix A.

| | Train | Valid | Test |
|---|---|---|---|
| *ROCStories data:* | | | |
| # Stories | 98,159 | 1,871 | 1,871 |
| TIMETRAVEL*:* | | | |
| # Counterfactual Context | 98,159 | 5,613 | 7,484 |
| # Edited Ending | 16,752 | 5,613 | 7,484 |

Table 2: Dataset statistics

## 4 Learning a Counterfactual Rewriter

Recent work in constructing large-scale generative language models based on transformers (Radford et al., 2018, 2019) has led to considerable improvements in natural language generation tasks. Due to their current prominence, we use them as baselines to study the extent to which the current neural text generation systems can perform and fail counterfactual narrative reasoning and revision. We focus on the family of GPT models, including GPT (Radford et al., 2018) and the latest small- (GPT2-S) and medium-sized (GPT2-M) transformer models from Radford et al. (2019). For each of the three pretrained language models, we fine-tune with multiple objectives, leading to 14 different model variants for the task, which we

describe in more detail below.

### 4.1 Unsupervised Training

Constructing large-scale counterfactual revision dataset is costly. Therefore, an ideal system must learn to reason without direct supervision. Toward this goal, we examine how unsupervised approaches to counterfactual story rewriting perform on our evaluation task. We devise the following unsupervised settings for models to learn to generate counterfactual story endings.

**Zero-shot (ZS)** In our simplest setting, we evaluate the counterfactual reasoning abilities already learned by these models due to pretraining on large corpora: the BooksCorpus dataset (Zhu et al., 2015) for GPT and the WebText corpus for GPT-2 (Radford et al., 2019). In this setting, models are not trained on any portion of the training data from TIMETRAVEL and must instead produce counterfactual rewritten stories for the evaluation set using only the representations learned from pretraining. At test time, the model receives the premise and the counterfactual context $(\boldsymbol{s}_1, \boldsymbol{s}'_2)$ as input and generates the tokens that constitute the rewritten counterfactual outcome.

**Fine-tuning (FT)** Because the domains on which both the GPT and GPT2 models were trained are broad and more complex than the domain of ROCStories, we investigate whether adapting the language model to the data distribution of ROCStories is helpful for learning to reason about counterfactuals. In this setting, the model is further fine-tuned to maximize the log-likelihood of the stories in the ROCStories corpus:

$$\mathcal{L}^{ft}(\boldsymbol{\theta}) = \log p_\theta(S), \tag{1}$$

where $p_\theta$ is the language model with parameters $\boldsymbol{\theta}$, and $S$ is the original story as defined in Section 3.1. This fine-tuning step encourages the model to generate text with the same consistent style of the stories. Similar to the zero-shot setting, the premise and the counterfactual sentence $(\boldsymbol{s}_1, \boldsymbol{s}'_2)$ are provided as input to the model.

**Fine-tuning + Counterfactual (FT + CF)** The above training loss, however, does not make use of the additional 81,407 counterfactual training sentences for fine-tuning. To inform the model with a larger set of possible counterfactual narratives in the training data, we propose an additional loss function that fits the model to the counterfactual sentences given the premise sentence:

$$\mathcal{L}^{cf}(\boldsymbol{\theta}) = \log p_\theta(\boldsymbol{s}'_2|\boldsymbol{s}_1), \tag{2}$$

where $p_\theta(\boldsymbol{s}'_2|\boldsymbol{s}_1)$ denotes that the language model first reads the premise $\boldsymbol{s}_1$ and maximizes the log-likelihood of counterfactual sentence $\boldsymbol{s}'_2$. The model is fine-tuned with both objectives in Eqs (1) and (2):

$$\mathcal{L}^{ft+cf}(\boldsymbol{\theta}) = \mathcal{L}^{ft} + \mathcal{L}^{cf}, \tag{3}$$

and receives inputs in the same format as the zero-shot and fine-tuned models at test time.

**Reconstruction + Counterfactual (RC + CF)** One issue with the above training procedures is that models are not explicitly trained to retain as much text of the original outcome $\boldsymbol{x}_{3:5}$ as possible (i.e., minimum edits). If these models are to learn to "rewrite" the original story ending given the counterfactual sentence, rather than learning to produce a completely new plot, they must be able to condition on the original ending during generation. Motivated by this requirement and following the goal of developing unsupervised methods for counterfactual rewriting, we design a reconstruction objective for learning a noisy reproduction of the original ending. Specifically, we provide the model with both the original story and a masked context as input $(S, [s], \boldsymbol{s}_1, [mask])$ and train the model to reconstruct the original ending $\boldsymbol{s}_{3:5}$:

$$\mathcal{L}^{rc}(\boldsymbol{\theta}) = \log p_\theta(\boldsymbol{s}_{3:5}|S, [s], \boldsymbol{s}_1, [mask]), \tag{4}$$

where `[s]` denotes a separator token and `[mask]` is a special mask token. In this setting, the model first reads the original story $S$ followed by the separation token `[s]`, and then reads the premise $\boldsymbol{x}_1$ again, followed by the mask token `[mask]`, which serves as a placeholder sentence for the counterfactual sentence. This objective encourages the model to reproduce the original ending $\boldsymbol{s}_{3:5}$ in the general case where the second sentence is not specified, thereby encouraging generations similar to the original ending regardless of the counterfactual provided. At test time, we replace `[mask]` in the input

| | |
|---|---|
| Premise | Rob loved to play soccer and was on his middle school team. |
| Initial | They had just started their season and were playing their first game. |
| Original Ending | Rob accidentally tripped over another player during the game. The other player was mad, but Rob apologized and kept his composure. Rob ended up playing the best game of his life! |
| Counterfactual | When he graduated from middle school, he started learning to play the guitar. |
| **Model** | **Generated Ending** |
| GPT-M + ZS | He played in bands for about a year, then got into touring with a band called The Muffs. He wrote the music for a few of The Muffs' songs. He was a very talented singer-songwriter and guitarist. |
| GPT-M + FT + CF | He took classes on the guitar for years to get better. He took a solo lesson and had a great time. Rob now is a famous guitarist. |
| GPT-M + Supervised | He accidentally tripped over the guitar during the song. The other person was mad, but Rob apologized and kept his composure. Rob ended up playing the best song of his life. |
| Premise | I called Carly to ask her on a date. |
| Initial | I got her voicemail. |
| Original Ending | I texted her that we should talk soon. She texted me back, asking why we should talk soon. I told her I liked her. |
| Counterfactual | She said that we should just stay friends which made me really sad. |
| **Model** | **Generated Ending** |
| GPT-M + ZS | She said that she went out and did something really stupid and that was all I could think about. I did my best to be nice to Carly by telling her how much I loved her and that I would be her best friend no matter what. There were nights when I felt like crying because I couldn't get out of bed. |
| GPT-M + FT + CF | I decided to watch a movie instead. After that I decided I should see a movie I had never seen before. I went to the theater and bought a ticket. |
| GPT-M + Sup | I texted her that we should just stay friends soon. She texted me back, asking why we should just stay friends soon. I told her I liked her. |

Table 3: Model Outputs

with the counterfactual sentence $s'_2$, and the model must generate the continuation of $(S, [s], s_1, s'_2)$. We also use the objective from Eq (2) above to inform the model with counterfactual information during training.

### 4.2 Supervised Training (Sup)

Our dataset also provides 16,752 training instances that include human annotated rewritten endings for supervised learning. To assess whether being able to train directly on alternative endings is helpful for learning counterfactual narrative understanding, we train models on this portion of data in a supervised manner. More concretely, the input to the model contains the full information $(S, [s], s_1, s'_2)$, and we train the model to maximize the log-likelihood of ground-truth rewritten endings:

$$\mathcal{L}^s(\boldsymbol{\theta}) = \log p_\theta(s'_{3:5}|S, [s], s_1, s'_2). \quad (5)$$

where `[s]` denotes a separator token.

### 4.3 Hyperparameters

We largely follow the same training and inference setups as in Radford et al. (2018) for the GPT model and Radford et al. (2019) for the GPT2 variants. Experiments are implemented with the text generation toolkit Texar (Hu et al., 2019). We provide more details in Appendix B.

| Model | **Pre (1)** | **Plot (2)** | **CF (3)** |
|---|---|---|---|
| GPT + ZS | 1.945 | 1.290 | 1.555 |
| GPT2-S + ZS | 1.945 | 1.335 | 1.475 |
| GPT2-M + ZS | 2.435 | 1.615 | 2.045 |
| GPT + FT | 2.485 | 1.750 | 2.005 |
| GPT2-S + FT | 2.365 | 1.645 | 1.895 |
| GPT2-M + FT | 2.580 | 1.790 | **2.070** |
| GPT + FT + CF | 2.310 | 1.595 | 1.925 |
| GPT2-S + FT + CF | 2.310 | 1.640 | 1.850 |
| GPT2-M + FT + CF | 2.395 | 1.650 | 1.945 |
| GPT2-S + RC + CF | 2.240 | 2.090 | 1.500 |
| GPT2-M + RC + CF | **2.780** | 2.595 | 1.660 |
| GPT + Sup | 2.630 | **2.690** | 1.460 |
| GPT2-S + Sup | 2.705 | 2.650 | 1.625 |
| GPT2-M + Sup | 2.750 | 2.620 | 1.820 |
| Human | 2.830 | 2.545 | 2.520 |

Table 4: Likert scale scores for different models. The top performing model for each question is **bolded**.

## 5 Human Study of Rewritten Sentences

To assess the quality of rewritten endings, we conduct two sets of human evaluation. To give a sense of the model generation, Table 3 presents example outputs by a subset of representative models on two test cases.

### 5.1 Rewritten Sentence Scoring

**Setup** In this setting, workers from Amazon Mechanical Turk were presented 100 outputs from 14 different models. For each example, two workers were presented the original premise sentence, the original ending, the counterfactual sentence, and the rewritten ending, and asked to answer the following three questions on a 3-point Likert scale:
**(1)** Does the rewritten ending keep in mind details of the original premise sentence?
**(2)** Is the plot of the rewritten ending relevant to the plot of the original ending?
**(3)** Does the rewritten ending respect the changes induced by the counterfactual sentence?
In addition to evaluating the 14 models, we also provided gold human annotated counterfactual endings for the same 100 test examples to compute an expected upper bound for how models should perform. We present the results from this study in Table 4 and share key observations below.

**Model Size and Pretraining Data** We observe that models with more parameters are better at the counterfactual rewriting task than smaller models. The GPT2-M variants consistently outperform the GPT and GPT2-S models, regardless of the objective on which the model was trained. Interestingly, however, the GPT model appears to generally outperform the GPT2-S model on the counterfactual question **(3)**, indicating that the domain on which models are pretrained does affect how adaptable their representations are to the story rewriting task.

**Domain Adaptation** Another pattern we notice is that fine-tuning on the ROCStories data (FT) is always helpful for increasing performance on counterfactual relevance (**CF (3)** in Table 4), indicating adapting to the ROCStories-style language distribution helps the model learn to produce relevant rewrites for counterfactuals, especially for models with fewer parameters. The **Plot (2)** question in Table 4 indicates why this might be the case, as the zero-shot models tend to produce more creative rewritings that are not at all tied to the original story. Interestingly, however, fine-tuning with the larger set of counterfactuals (CF loss) does not seem to help in rewriting endings that relate to the counterfactuals well.

**Supervised vs. Unsupervised Learning** A surprising observation is that using the dataset of labeled rewritten endings for training does not seem to help the language models learn to rewrite endings better. While the supervised models are generally able to adhere to the plot better than unsupervised methods, their new endings do not score well on question **(3)**, indicating that they may be copying the original ending or learning to paraphrase the original story ending without acknowledging the counterfactual sentence. This points to the fact that this task cannot be trivially solved by adding more paired data, since adding more data merely simplifies to having more stories in the dataset, without necessarily learning to handle counterfactuals more effectively.

| COUNTERFACTUAL - Human Judges Preferred | | | | |
|---|---|---|---|---|
| Best model | | Neutral | Comparator | |
| M+Sup | 20.0 | 7.0 | **29.5** | M+FT+CF |
| M+Sup | 19.0 | 3.0 | **38.5** | M+FT |
| M+Sup | **23.5** | 14.0 | 4.5 | M+Recon+ CF |
| M+Sup | 26.5 | 5.0 | **33.5** | M+ zero-shot |
| M+Sup | **14.0** | 18.5 | 6.0 | S+Sup |
| M+Sup | **18.5** | 20.0 | 8.0 | GPT + Sup |
| M+Sup | 10.0 | 15.0 | **52.0** | Human |

| PLOT - Human Judges Preferred | | | | |
|---|---|---|---|---|
| Best model | | Neutral | Comparator | |
| M+Sup | **57.5** | 14.5 | 13.5 | M+FT+CF |
| M+Sup | **58.5** | 16.5 | 12.5 | M+FT |
| M+Sup | 11.5 | 60.0 | **16.5** | M+Recon+CF |
| M+Sup | **63.0** | 14.5 | 11.0 | M+zero-shot |
| M+Sup | 11.5 | 62.5 | **12.5** | S+Sup |
| M+Sup | 14.5 | 61.0 | **15.0** | GPT+Sup |
| M+Sup | 22.0 | 47.5 | **25.0** | Human |

| PREMISE - Human Judges Preferred | | | | |
|---|---|---|---|---|
| Best model | | Neutral | Comparator | |
| M+Sup | **35.5** | 31.0 | 16.5 | M+FT+CF |
| M+Sup | **32.5** | 39.5 | 14.0 | M+FT |
| M+Sup | **10.5** | 65.0 | 9.0 | M+Recon+CF |
| M+Sup | **46.5** | 29.5 | 13.0 | M+zero-shot |
| M+Sup | **8.5** | 71.0 | 7.5 | S+Sup |
| M+Sup | **12.0** | 68.0 | 7.5 | GPT+Sup |
| M+Sup | 12.5 | 59.0 | **22.5** | Human |

Table 5: Pairwise human comparison between the best model (GPT2-M + Sup) and comparison models on all three questions. "Neutral" means both are "equally good". Percentage of "equally bad" are omitted.

### 5.2 Pairwise Model Preference

**Setup** We conduct a pairwise comparison between the best model (GPT2-M + Sup) with other models along the same three dimensions as in the first evaluation setting (section 5.1).

| Metric | (1) Prem | (2) Plot | (3) CF |
|---|---|---|---|
| BLEU-4 | **.2623** | **.6792** | **-.1804** |
| ROUGE-L | **.3187** | **.7484** | **-.1423** |
| WMS | **.2713** | **.5809** | -.0343 |
| S+WMS | **.2789** | **.6075** | -.0538 |
| BERT | **.2124** | **.1929** | **.1067** |
| BERT-FT | **.2408** | **.1847** | **.0995** |

Table 6: Pearson correlation between automatic metrics and human scores. **Bolded** numbers are statistically significant at $p < 0.05$.

Specifically, crowdworkers were presented outputs of a pair of systems, and asked to choose which one is better, or "equally good" or "equally bad", in terms of each of the three criteria. As in section 5.1, we evaluate 100 outputs of each model.

**Results** In Table 5, we present the human preference results, showing that the best model outperforms the comparison baselines in terms of consistency with premise, while being less consistently better with regards to the other two questions. Interestingly, a model that performs better on one of the evaluated dimensions often performs worse for another question, indicating plenty of room for future work in counterfactual reasoning for story rewriting.

## 6 Challenges for Automatic Metrics

While human scores provide the clearest insight into how models are able to complete the counterfactual rewriting task, their associated cost makes them difficult to scale to larger evaluation sets. To provide further insight into the performance of candidate models, we explore how different automatic metrics evaluate the produced generations.

### 6.1 Metrics

**Overlap Metrics** The most common metrics used in evaluating text generation are based on textual overlap between a candidate generated sequence and set of reference sequences provided by the dataset. **BLEU** (Papineni et al., 2002) is perhaps the most widely used metric in text generation, which computes the number of overlapping $n$-grams between the generated and reference sequences. Another commonly used metric in text generation (though originally designed for extractive summarization) is **ROUGE-L** (Lin, 2004), which measures the length of the longest common subsequence (LCS) between a candidate generation and reference. We report the performance of all models on both of these metrics.

**Model-based Metrics** Although BLEU and ROUGE are widely used in text generation, they use exact string matching, and thus fail to robustly match paraphrases and capture semantically-critical ordering changes. Recently, there has been a growing body of work in producing model-based metrics (Lowe et al., 2017) that use trained models and embeddings to score a sequence.

Kusner et al. (2015) proposed Word Mover's Distance, which defines the distance between two texts as the minimal cost of transforming one sequence's word embeddings to the other's. The measure finds a matching between the two texts that minimizes the total Euclidean distance between the matched word embeddings. Following Kilickaya et al. (2017), we take the negative exponential of this distance to get **Word Mover's Similarity (WMS)**. More recently, Clark et al. (2019) proposed **Sentence + Word Mover's Similarity (S+WMS)** to extend WMS for longer multi-sentence texts by using sentence representations in the minimum distance calculation in addition to word embeddings.[1]

Other recent methods use contextualized embeddings (Devlin et al., 2018) to compute similarity between sequences. We use **BERTScore** (Zhang et al., 2019), which computes cosine similarity between two sentences using BERT encodings. Zhang et al. show that BERTScore correlates better with human judgments than existing metrics such as BLEU, ROUGE, and other learning-based metrics. To adapt BERTScore to our task, we finetune BERT on ROCStories using the same training framework from Devlin et al. (2018) and compute **BERT-FT** the same way as before.

### 6.2 Human Correlation with Metrics

Recent work in text generation (Wiseman et al., 2017) and dialogue (Liu et al., 2016) have explored the limitations of automatic metrics for text production tasks. Due to the highly semantic nature of the counterfactual rewriting task and the need to recognize subtle changes in

[1]We follow previous work and use GloVe embeddings (Pennington et al., 2014) to represent words and the averaged word embeddings to represent sentences.

| | BLEU-4 | ROUGE-L | BERT | BERT-FT | WMS | W+SMS |
|---|---|---|---|---|---|---|
| *Training: Pretrained Only* | | | | *Input:* $\boldsymbol{s}_1\boldsymbol{s}_2'$ | | |
| GPT + zero-shot | 1.25 | 18.26 | 59.50 | 58.28 | 0.30 | 0.97 |
| GPT2-S + zero-shot | 1.28 | 20.27 | 59.62 | 58.11 | 0.33 | 1.09 |
| GPT2-M + zero-shot | 1.51 | 19.41 | 60.17 | 58.59 | 0.34 | 1.12 |
| *Training: Unsupervised + Generative* | | | | *Input:* $\boldsymbol{s}_1\boldsymbol{s}_2'$ | | |
| GPT + FT | 4.20 | 24.55 | 64.38 | 62.60 | 0.56 | 1.48 |
| GPT2-S + FT | 3.78 | 24.18 | 64.25 | 62.60 | 0.54 | 1.40 |
| GPT2-M + FT | 4.09 | 24.08 | 62.23 | 62.49 | 0.53 | 1.42 |
| GPT + FT + CF | 3.82 | 24.21 | 64.48 | 62.66 | 0.57 | 1.45 |
| GPT2-S + FT + CF | 3.96 | 24.06 | 64.50 | 62.71 | 0.53 | 1.44 |
| GPT2-M + FT + CF | 4.00 | 24.38 | 64.31 | 62.59 | 0.48 | 1.33 |
| *Training: Unsupervised + Discriminative* | | | | *Input:* $\boldsymbol{s}_1\boldsymbol{s}_2\boldsymbol{y}[S]\boldsymbol{s}_1[MASK]$ | | |
| GPT2-S + Recon + CF | 47.08 | 51.19 | **63.82** | 62.36 | 5.53 | 8.08 |
| GPT2-M + Recon + CF | **76.57** | **71.35** | 64.15 | 62.49 | **18.29** | **20.87** |
| *Training: Supervised + Discriminative* | | | | *Input:* $\boldsymbol{s}_1\boldsymbol{s}_2\boldsymbol{y}[S]\boldsymbol{s}_1\boldsymbol{s}_2'$ | | |
| GPT + Sup | 80.09 | 75.03 | 64.15 | 62.36 | 20.93 | 23.37 |
| GPT2-S + Sup | 79.03 | 73.31 | 64.14 | 62.40 | 20.57 | 22.97 |
| GPT2-M + Sup | 76.63 | 74.42 | 64.06 | **62.33** | 19.62 | 22.01 |
| Human | 65.12 | 68.58 | 63.58 | 61.82 | 16.95 | 19.16 |

Table 7: Results on automatic metrics for the cross-product of the models and loss functions proposed in Section 4. **Bolded** results are closest to the human score.

event descriptions, we anticipate that automatic metrics would have difficulty assessing rewritten endings. To test the correlation between available evaluation metrics for long-form generation and human opinions of quality of counterfactual generations, we compute the Pearson Correlation between automatic scores and human scores for 800 validation set data points, 300 taken from the gold annotations and 100 generated from each of the 5 GPT2-M variants.[2] For each example, we use the same questions and Likert scale evaluation as in §5 and report the results in Table 6.

As expected, the automatic metrics are decently correlated with human scores for adherence to the premise sentence and plot. However, these same metrics correlate negatively with question **(3)** – adherence to the counterfactual sentence – indicating poor measurement of counterfactual understanding if they were to be reported in their typical manner (i.e., higher score indicating superior performance). Only the BERTScore metrics appear to positively correlate with human scores for counterfactual understanding, making them usable for evaluating generations across properties related to all three questions. However, the correlation is weak, and the results in Table 7 indicate that the BERTScore metrics are difficult to distinguish between models.

[2]We include both human annotations and model-generated outputs in this computation to encourage diversity of source.

## 7 Conclusion

We introduced a new task of *Counterfactual Story Rewriting* that challenges current language understanding and generation systems with counterfactual reasoning. Our new dataset, TIMETRAVEL, provides nearly 30k counterfactual revisions to simple commonsense stories together with 100k counterfactual sentences. We establish baseline performances of state-of-the-art neural language models with over 14 model variants with zero-shot, unsupervised and supervised settings. The empirical results demonstrate that while neural language models show promises, they generally have difficulties in rewriting the consequences of the counterfactual condition with full consistency, suggesting more focused research on integrating true reasoning capabilities to neural language models.

## Acknowledgements

We thanks the anonymous reviewers as well as Michel Galley, Jianfeng Gao, and others for many helpful comments. This research was supported in part by NSF (IIS-1524371), DARPA CwC through ARO (W911NF15-1-0543), DARPA MCS program through NIWC Pacific (N66001-19-2-4031), and Allen Institute for AI.

## References


Leon Bottou. 2019. Learning representations using causal invariance. In *ICLR*.

Léon Bottou, Jonas Peters, Joaquin Quiñonero-Candela, Denis X Charles, D Max Chickering, Elon Portugaly, Dipankar Ray, Patrice Simard, and Ed Snelson. 2013. Counterfactual reasoning and learning systems: The example of computational advertising. *JMLR*.

Ruth MJ Byrne. 2002. Mental models and counterfactual thoughts about what might have been. *Trends in cognitive sciences*, 6(10):426–431.

Nathanael Chambers. 2013. Event schema induction with a probabilistic entity-driven model. In *EMNLP*, pages 1797–1807.

Elizabeth Clark, Asli Celikyilmaz, and Noah A. Smith. 2019. Sentence mover's similarity: Automatic evaluation for multi-sentence texts. In *ACL*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Kai Epstude and Neal J Roese. 2008. The functional theory of counterfactual thinking. *Personality and Social Psychology Review*, 12(2):168–192.

Nelson Goodman. 1947. The problem of counterfactual conditionals. *The Journal of Philosophy*, 44(5):113–128.

Jerry R Hobbs. 2005. Toward a useful concept of causality for lexical semantics. *Journal of Semantics*, 22(2):181–209.

Zhiting Hu, Haoran Shi, Bowen Tan, Wentao Wang, Zichao Yang, Tiancheng Zhao, Junxian He, Lianhui Qin, Di Wang, et al. 2019. Texar: A modularized, versatile, and extensible toolkit for text generation. In *ACL, System Demonstrations*.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *ICML*.

Mert Kilickaya, Aykut Erdem, Nazli Ikizler-Cinbis, and Erkut Erdem. 2017. Re-evaluating automatic metrics for image captioning. In *EACL*.

Rik Koncel-Kedziorski, Ioannis Konstas, Luke S. Zettlemoyer, and Hannaneh Hajishirzi. 2016. A theme-rewriting approach for generating algebra word problems. In *EMNLP*.

Matt J. Kusner, Yu Sun, Nicholas I. Kolkin, and Kilian Q. Weinberger. 2015. From word embeddings to document distances. In *ICML*.

Guillaume Lample, Sandeep Subramanian, Eric Michael Smith, Ludovic Denoyer, Marc'Aurelio Ranzato, and Y-Lan Boureau. 2019. Multiple-attribute text rewriting. In *ICLR*.

Carolin Lawrence and Stefan Riezler. 2018. Improving a neural semantic parser by counterfactual learning from human bandit feedback. In *ACL*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *EMNLP*.

Ryan Lowe, Michael Noseworthy, Iulian Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. 2017. Towards an automatic turing test: Learning to evaluate dialogue responses. In *ICLR*.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and evaluation framework for deeper understanding of commonsense stories. *arXiv preprint arXiv:1604.01696*.

Timothy Niven and Hung-Yu Kao. 2019. Probing neural network comprehension of natural language arguments. *arXiv preprint arXiv:1907.07355*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318.

Judea Pearl. 2000. *Causality: models, reasoning and inference*, volume 29. Springer.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Karl Pichotta and Raymond Mooney. 2014. Statistical script learning with multi-argument events. In *EACL*, pages 220–229.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1:8.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S. Gordon. 2011. Semeval-2012 task 7: Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *SemEval@NAACL-HLT*.

Maarten Sap, Ronan LeBras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A Smith, and Yejin Choi. 2019. ATOMIC: An atlas of machine commonsense for if-then reasoning. In *AAAI*.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *NeurIPS*.

Youngseo Son, Anneke Buffone, Joe Raso, Allegra Larche, Anthony Janocko, Kevin Zembroski, H. Andrew Schwartz, and Lyle H. Ungar. 2017. Recognizing counterfactual thinking in social media texts. In *ACL*.

Sam Wiseman, Stuart M. Shieber, and Alexander M. Rush. 2017. Challenges in data-to-document generation. In *EMNLP*.

Jim Woodward. 2002. What is a mechanism? a counterfactual account. *Philosophy of Science*, 69(S3):S366–S377.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. Swag: A large-scale adversarial dataset for grounded commonsense inference. In *EMNLP*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *ACL*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. BERTScore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan R. Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *ICCV*, pages 19–27.

# Encode, Tag, Realize: High-Precision Text Editing

**Eric Malmi**
Google Research
emalmi@google.com

**Sebastian Krause**
Google Research
bastik@google.com

**Sascha Rothe**
Google Research
rothe@google.com

**Daniil Mirylenka**
Google Research
dmirylenka@google.com

**Aliaksei Severyn**
Google Research
severyn@google.com

## Abstract

We propose LASERTAGGER—a sequence tagging approach that casts text generation as a text editing task. Target texts are reconstructed from the inputs using three main edit operations: *keeping* a token, *deleting* it, and *adding* a phrase before the token. To predict the edit operations, we propose a novel model, which combines a BERT encoder with an autoregressive Transformer decoder. This approach is evaluated on English text on four tasks: sentence fusion, sentence splitting, abstractive summarization, and grammar correction. LASERTAGGER achieves new state-of-the-art results on three of these tasks, performs comparably to a set of strong seq2seq baselines with a large number of training examples, and outperforms them when the number of examples is limited. Furthermore, we show that at inference time tagging can be more than two orders of magnitude faster than comparable seq2seq models, making it more attractive for running in a live environment.

## 1 Introduction

Neural sequence-to-sequence (seq2seq) models provide a powerful framework for learning to translate source texts into target texts. Since their first application to machine translation (MT) (Sutskever et al., 2014) they have become the de facto approach for virtually every text generation task, including summarization (Tan et al., 2017), image captioning (Xu et al., 2015), text style transfer (Rao and Tetreault, 2018; Nikolov and Hahnloser, 2018; Jin et al., 2019), and grammatical error correction (Chollampatt and Ng, 2018; Grundkiewicz et al., 2019).

We observe that in some text generation tasks, such as the recently introduced *sentence splitting* and *sentence fusion* tasks, output texts highly overlap with inputs. In this setting, learning a seq2seq model to generate the output text from scratch seems intuitively wasteful. Copy mechanisms (Gu et al., 2016; See et al., 2017) allow for choosing between copying source tokens and generating arbitrary tokens, but although such hybrid models help with out-of-vocabulary words, they still require large training sets as they depend on output vocabularies as large as those used by the standard seq2seq approaches.

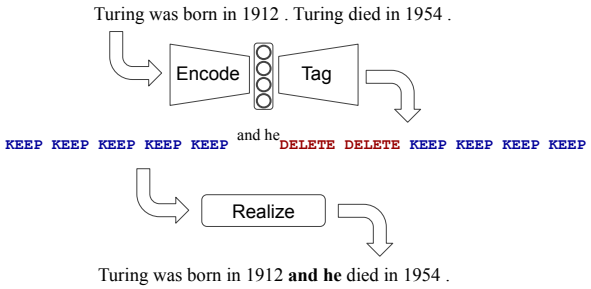


Figure 1: LASERTAGGER applied to sentence fusion.

In contrast, we propose learning a text editing model that applies a set of edit operations on the input sequence to reconstruct the output. We show that it is often enough to use a relatively small set of output tags representing text deletion, rephrasing and word reordering to be able to reproduce a large percentage of the targets in the training data. This results in a learning problem with a much smaller vocabulary size, and the output length fixed to the number of words in the source text. This, in turn, greatly reduces the number training examples required to train accurate models, which is particularly important in applications where only a small number of human-labeled data is available.

Our tagging approach, LASERTAGGER, consists of three steps (Fig. 1): (i) *Encode* builds a representation of the input sequence, (ii) *Tag* assigns edit tags from a pre-computed output vocabulary to the input tokens, and (iii) *Realize* applies a simple set of rules to convert tags into the output text tokens.

An experimental evaluation of LASERTAGGER



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5054–5065,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

on four different text generation tasks shows that it yields comparable results to seq2seq models when we have tens of thousands of training examples and clearly outperforms them when the number of examples is smaller.

Our contributions are the following:

**1)** We demonstrate that many text generation tasks with overlapping inputs and outputs can be effectively treated as text editing tasks.

**2)** We propose LASERTAGGER—a sequence tagging-based model for text editing, together with a method for generating the tag vocabulary from the training data.

**3)** We describe two versions of the tagging model: (i) LASERTAGGER$_{\text{FF}}$—a tagger based on BERT (Devlin et al., 2019) and (ii) LASERTAGGER$_{\text{AR}}$—a novel tagging model combining the BERT encoder with an autoregressive Transformer decoder, which further improves the results over the BERT tagger.

**4)** We evaluate LASERTAGGER against strong seq2seq baseline models based on the BERT architecture. Our baseline models outperform previously reported state-of-the-art results on two tasks.

**5)** We demonstrate that a) LASERTAGGER$_{\text{AR}}$ achieves state-of-the-art or comparable results on 3 out of 4 examined tasks, b) LASERTAGGER$_{\text{FF}}$ is up to *100x faster at inference time* with performance comparable to the state-of-the-art seq2seq models. Furthermore, both models: c) require much *less training data* compared to the seq2seq models, d) are more *controllable and interpretable* than seq2seq models due to the small vocabulary of edit operations, e) are *less prone* to typical seq2seq model errors, such as hallucination.

The code will be available at:
`lasertagger.page.link/code`

## 2 Related Work

Recent work discusses some of the difficulties of learning neural decoders for text generation (Wiseman et al., 2018; Prabhakaran et al., 2018). Conventional seq2seq approaches require large amounts of training data, are hard to control and to constrain to desirable outputs. At the same time, many NLP tasks that appear to be full-fledged text generation tasks are natural testbeds for simpler methods. In this section we briefly review some of these tasks.

**Text Simplification** is a paraphrasing task that is known to benefit from modeling edit operations. A simple instance of this type are sentence compression systems that apply a drop operation at the token/phrase level (Filippova and Strube, 2008; Filippova et al., 2015), while more intricate systems also apply splitting, reordering, and lexical substitution (Zhu et al., 2010). Simplification has also been attempted with systems developed for phrase-based MT (Xu et al., 2016a), as well as with neural encoder-decoder models (Zhang and Lapata, 2017).

Independent of this work, Dong et al. (2019) recently proposed a text-editing model, similar to ours, for text simplification. The main differences to our work are: $(i)$ They introduce an interpreter module which acts as a language model for the so-far-realized text, and $(ii)$ they generate added tokens one-by-one from a full vocabulary rather than from an optimized set of frequently added phrases. The latter allows their model to generate more diverse output, but it may negatively effect the inference time, precision, and the data efficiency of their model. Another recent model similar to ours is called Levenshtein Transformer Gu et al. (2019), which does text editing by performing a sequence of deletion and insertion actions.

**Single-document summarization** is a task that requires systems to shorten texts in a meaning-preserving way. It has been approached with deletion-based methods on the token level (Filippova et al., 2015) and the sentence level (Narayan et al., 2018; Liu, 2019). Other papers have used neural encoder-decoder methods (Tan et al., 2017; Rush et al., 2015; Paulus et al., 2017) to do *abstractive* summarization, which allows edits beyond mere deletion. This can be motivated by the work of Jing and McKeown (2000), who identified a small number of fundamental high-level editing operations that are useful for producing summaries (reduction, combination, syntactic transformation, lexical paraphrasing, generalization/specification, and reordering). See et al. (2017) extended a neural encoder-decoder model with a copy mechanism to allow the model to more easily reproduce input tokens during generation.

Out of available summarization datasets (Dernoncourt et al., 2018), we find the one by Toutanova et al. (2016) particularly interesting because (1) it specifically targets abstractive summarization systems, (2) the lengths of texts in this dataset (short paragraphs) seem well-suited for text editing, and (3) an analysis showed that the dataset covers many different summarization operations.

In **Grammatical Error Correction** (Ng et al.,

2013, 2014) a system is presented with input texts written usually by a language learner, and is tasked with detecting and fixing grammatical (and other) mistakes. Approaches to this task often incorporate task-specific knowledge, e.g., by designing classifiers for specific error types (Knight and Chander, 1994; Rozovskaya et al., 2014) that can be trained without manually labeled data, or by adapting statistical machine-translation methods (Junczys-Dowmunt and Grundkiewicz, 2014). Methods for the sub-problem of error *detection* are similar in spirit to sentence compression systems, in that they are implemented as word-based neural sequence labelers (Rei, 2017; Rei et al., 2017). Neural encoder-decoder methods are also commonly applied to the error correction task (Ge et al., 2018; Chollampatt and Ng, 2018; Zhao et al., 2019), but suffer from a lack of training data, which is why task-specific tricks need to be applied (Kasewa et al., 2018; Junczys-Dowmunt et al., 2018).

## 3 Text Editing as a Tagging Problem

Our approach to text editing is to cast it into a tagging problem. Here we describe its main components: (1) the tagging operations, (2) how to convert plain-text training targets into a tagging format, as well as (3) the realization step to convert tags into the final output text.

### 3.1 Tagging Operations

Our tagger assigns a tag to each input token. A tag is composed of two parts: a *base tag* and an *added phrase*. The base tag is either `KEEP` or `DELETE`, which indicates whether to retain the token in the output. The added phrase $P$, which can be empty, enforces that $P$ is added before the corresponding token. $P$ belongs to a vocabulary $V$ that defines a set of words and phrases that can be inserted into the input sequence to transform it into the output.

The combination of the base tag $B$ and the added phrase $P$ is treated as a single tag and denoted by ${}^{P}B$. The total number of unique tags is equal to the number of base tags times the size of the phrase vocabulary, hence there are $\approx 2|V|$ unique tags.

Additional task-specific tags can be employed too. For sentence fusion (Section 5.1), the input consists of two sentences, which sometimes need to be swapped. Therefore, we introduce a custom tag, `SWAP`, which can only be applied to the last period of the first sentence (see Fig. 2). This tag instructs the Realize step to swap the order of the input sentences before realizing the rest of the tags.

For other tasks, different supplementary tags may be useful. E.g., to allow for replacing entity mentions with the appropriate pronouns, we could introduce a `PRONOMINALIZE` tag. Given an access to a knowledge base that includes entity gender information, we could then look up the correct pronoun during the realization step, instead of having to rely on the model predicting the correct tag (${}^{\text{she}}$`DELETE`, ${}^{\text{he}}$`DELETE`, ${}^{\text{they}}$`DELETE`, etc.).

### 3.2 Optimizing Phrase Vocabulary

The phrase vocabulary consists of phrases that can be added between the source words. On the one hand, we wish to minimize the number of phrases to keep the output tag vocabulary small. On the other hand, we would like to maximize the percentage of target texts that can be reconstructed from the source using the available tagging operations. This leads to the following combinatorial optimization problem.

**Problem 1** *Given a collection of phrase sets $A_1, A_2, \ldots A_m$, where $A_i \subseteq P$ and $P$ is the set of all candidate phrases, select a phrase vocabulary $V \subset P$ of at most $\ell$ phrases (i.e. $|V| \leq \ell$) so that the number of covered phrase sets is maximized. A phrase set $A_i$ is covered if and only if $A_i \subseteq V$.*

This problem is closely related to the minimum $k$-union problem which is NP-hard (Vinterbo, 2002). The latter problem asks for a set of $k$ phrase sets such that the cardinality of their union is the minimum. If we were able to solve Problem 1 in polynomial time, we could solve also the minimum $k$-union problem in polynomial time simply by finding the smallest phrase vocabulary size $\ell$ such that the number of covered phrase sets is at least $k$. This reduction from the minimum $k$-union problem gives us the following result:

**Theorem 1** *Problem 1 is NP-hard.*

To identify candidate phrases to be included in the vocabulary, we first align each source text $s$ from the training data with its target text $t$. This is achieved by computing the *longest common subsequence* (LCS) between the two word sequences, which can be done using dynamic programming in time $\mathcal{O}(|s| \times |t|)$. The n-grams in the target text that are not part of the LCS are the phrases that would need to be included in the phrase vocabulary to be able to construct $t$ from $s$.

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Source:** | Dylan | won | Nobel | prize | . | Dylan | is | | an | American | musician | . | |
| **Tags:** | DELETE | KEEP | KEEP | KEEP | SWAP | KEEP | $^{comma}$DELETE | | KEEP | KEEP | KEEP | $^{comma}$DELETE | |
| **Realization:** | Dylan , an American musician , won Nobel prize . | | | | | | | | | | | | |

Figure 2: An example sentence fusion obtained by tagging using the SWAP tag, which swaps the order of the two source sentences.

In practice, the phrase vocabulary is expected to consist of phrases that are frequently added to the target. Thus we adopt the following simple approach to construct the phrase vocabulary: sort the phrases by the number of phrase sets in which they occur and pick $\ell$ most frequent phrases. This was found to produce meaningful phrase vocabularies based on manual inspection as shown in Section 5. E.g., the top phrases for sentence fusions include many discourse connectives.

We also considered a greedy approach that constructs the vocabulary one phrase at a time, always selecting the phrase that has the largest incremental coverage. This approach is not, however, ideal for our use case, since some frequent phrases, such as "(" and ")", are strongly coupled. Selecting "(" alone has close to zero incremental coverage, but together with ")", they can cover many examples.

### 3.3 Converting Training Targets into Tags

Once the phrase vocabulary is determined, we can convert the target texts in our training data into tag sequences. Given the phrase vocabulary, we do not need to compute the LCS, but can leverage a more efficient approach, which iterates over words in the input and greedily attempts to match them (1) against the words in the target, and in case there is no match, (2) against the phrases in the vocabulary $V$. This can be done in $\mathcal{O}(|s| \times n_p)$ time, where $n_p$ is the length of the longest phrase in $V$, as shown in Algorithm 1.

The training targets that would require adding a phrase that is not in our vocabulary $V$, will not get converted into a tag sequence but are filtered out. While making the training dataset smaller, this may effectively also filter out low-quality targets. The percentage of converted examples for different datasets is reported later in Section 5. Note that even when the target cannot be reconstructed from the inputs using our output tag vocabulary, our approach might still produce reasonable outputs with the available phrases. E.g., a target may require the use of the infrequent ";" token, which is not in our vocabulary, but a model could instead choose to predict a more common "," token.

**Algorithm 1** Converting a target string to tags.

**Input:** Source text $s = [s(1), \ldots, s(n_s)]$, target text $t = [t(1), \ldots, t(n_t)]$, phrase vocabulary $V$, and the maximum added phrase length $n_p$.
**Output:** Tag sequence $x$ of length $n_s$ or of length 0 if conversion is not possible.

$x(i) =$ DELETE, $\forall i = 1, \ldots, n_s$ ▷ Initialize tags.
$i_s = 1$ ▷ Current source word index.
$i_t = 1$ ▷ Current target word index.
**while** $i_t \leq n_t$ **do**
  **if** $i_s > n_s$ **then**
    **return** [] ▷ Conversion infeasible.
  **if** $s(i_s) == t(i_t)$ **then**
    $x(i_s) =$ KEEP
    $i_t = i_t + 1$
  **else**
    $p = []$ ▷ Added phrase (word sequence).
    match_found $= 0$
    **for** $j = 1, \ldots, n_p$ **do**
      $p$.append($t(i_t + j - 1)$)
      **if** $s(i_s) == t(i_t + j)$ **and** $p \in V$ **then**
        match_found $= 1$
        **break**
    **if** match_found **then**
      $x(i_s) = {}^{p}$KEEP
      $i_t = i_t + |p| + 1$
  $i_s = i_s + 1$
**return** $x$ ▷ Target has been consumed, so return tags.

### 3.4 Realization

After obtaining a predicted tag sequence, we convert it to text ("realization" step). While classic works on text generation make a distinction between *planning* and *realization*, end-to-end neural approaches typically ignore this distinction, with the exception of few works (Moryossef et al., 2019; Puduppully et al., 2018).

For the basic tagging operations of keeping, deleting, and adding, realization is a straightforward process. Additionally, we adjust capitalization at sentence boundaries. Realization becomes more involved if we introduce special tags, such as PRONOMINALIZE mentioned in Section 3.1. For this tag, we would need to look up the gender of the tagged entity from a knowledge base. Having a separate realization step is beneficial, since we can decide to pronominalize only when confident about the appropriate pronoun and can, otherwise, leave the entity mention untouched.

Another advantage of having a separate realiza-

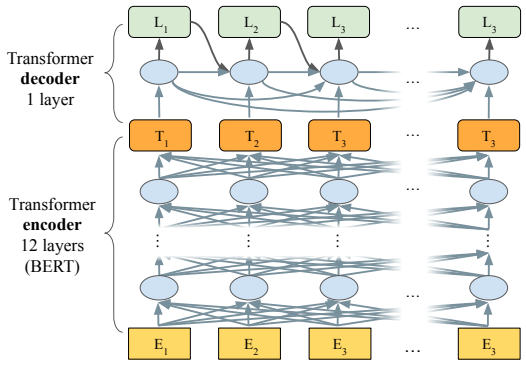


Figure 3: The architecture of LASERTAGGER$_{\text{AR}}$.

tion step is that specific loss patterns can be addressed by adding specialized realization rules. For instance, one could have a rule that when applying tag $^{\text{his}}$DELETE to an entity mention followed by *'s*, the realizer must always DELETE the possessive *'s* regardless of its predicted tag.

## 4 Tagging Model Architecture

Our tagger is composed of two components: an encoder, which generates activation vectors for each element in the input sequence, and a decoder, which converts encoder activations into tag labels.

**Encoder.** We choose the BERT Transformer model (Devlin et al., 2019) as our encoder, as it demonstrated state-of-the-art results on a number of sentence encoding tasks. We use the BERT-base architecture, which consists of 12 self-attention layers. We refer the reader to (Devlin et al., 2019) for a detailed description of the model architecture and its input representation. We initialize the encoder with a publicly available checkpoint of the pretrained case-sensitive BERT-base model.[1]

**Decoder.** In the original BERT paper a simple decoding mechanism is used for sequence tagging: the output tags are generated in a single feed-forward pass by applying an argmax over the encoder logits. In this way, each output tag is predicted independently, without modelling the dependencies between the tags in the sequence. Such a simple decoder demonstrated state-of-the-art results on the Named Entity Recognition task, when applied on top of the BERT encoder.

To better model the dependencies between the output tag labels, we propose a more powerful autoregressive decoder. Specifically, we run a single-layer Transformer decoder on top of the BERT encoder (see Fig. 3). At each step, the decoder is consuming the embedding of the previously predicted label and the activations from the encoder.

There are several ways in which the decoder can communicate with the encoder: (i) through a full attention over the sequence of encoder activations (similar to conventional seq2seq architectures); and (ii) by directly consuming the encoder activation at the current step. In our preliminary experiments, we found the latter option to perform better and converge faster, as it does not require learning additional encoder-decoder attention weights.

We experiment with both decoder variants (*feedforward* and *autoregressive*) and find that the autoregressive decoder outperforms the previously used *feedforward* decoder. In the rest of this paper, the tagging model with an *autoregressive* decoder is referred to as LASERTAGGER$_{\text{AR}}$ and the model with *feedforward* decoder as LASERTAGGER$_{\text{FF}}$.

## 5 Experiments

We evaluate our method by conducting experiments on four different text editing tasks: Sentence Fusion, Split and Rephrase, Abstractive Summarization, and Grammatical Error Correction.

**Baselines.** In addition to reporting previously published results for each task, we also train a set of strong baselines based on Transformer where both the encoder and decoder replicate the BERT-base architecture (Devlin et al., 2019). To have a fair comparison, similar to how we initialize a tagger encoder with a pretrained BERT checkpoint, we use the same initialization for the Transformer encoder. This produces a very strong seq2seq baseline (SEQ2SEQ$_{\text{BERT}}$), which already results in new state-of-the-art metrics on two out of four tasks.

### 5.1 Sentence Fusion

Sentence Fusion is the problem of fusing sentences into a single coherent sentence.

**Data.** We use the "balanced Wikipedia" portion of Geva et al. (2019)'s DiscoFuse dataset for our experiments (henceforth DfWiki). Out of the 4.5M fusion examples in the dataset, 10.5% require reordering of the input. To cope with this, we introduce the SWAP tag, which enables the model to flip the order of two input sentences. We construct the phrase vocabulary as described in Sec. 3.2 using the validation set of 46K examples. The top 15 phrases are shown in the first column of Table 1.

**Evaluation metrics.** Following Geva et al. (2019), we use two evaluation metrics: *Exact score*,

[1] github.com/google-research/bert

| DfWiki | WikiSplit | AS | GEC |
|---|---|---|---|
| , | .␣⟨::::⟩ | , | , |
| and | , | . | . |
| however␣, | .␣⟨::::⟩␣he | the | the |
| ,␣but | .␣⟨::::⟩␣it | a | a |
| he | the | & | to |
| because | and | and | in |
| ,␣although | was | is | of |
| but | is | in | on |
| ,␣and | " | " | at |
| although | .␣⟨::::⟩␣she | 's | for |
| his | .␣⟨::::⟩␣it␣is | with | have |
| ,␣while | a | for | is |
| it | .␣⟨::::⟩␣they | of | was |
| ,␣which | .␣⟨::::⟩␣however | n't | and |
| she | he | an | that |

Table 1: The 15 most frequently added phrases in the datasets studied in this work, in order of decreasing frequency. ⟨::::⟩ marks a sentence boundary. "AS"/"GEC" is short for Abstractive Summarization/Grammatical Error Correction.

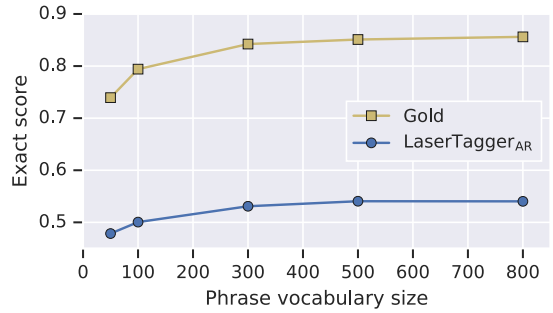


Figure 4: Performance of model LaserTagger$_{AR}$ on the DfWiki dataset, conditioned on the vocabulary size and the gold score, i.e. the percentage of examples that can be reconstructed via text-edit operations.

which is the percentage of exactly correctly predicted fusions, and *SARI* (Xu et al., 2016b), which computes the average F1 scores of the added, kept, and deleted n-grams.[2]

**Vocabulary Size.** To understand the impact of the number of phrases we include in the vocabulary, we trained models for different vocabulary sizes (only LaserTagger$_{AR}$). The results are shown in Figure 4. After increasing the vocabulary size to 500 phrases, Exact score reaches a plateau, so we set the vocabulary size to 500 in all the remaining experiments of this paper.[3] The Gold curve in Fig. 4 shows that this vocabulary size is sufficient to cover 85% of the training examples, which gives us an upper bound for the Exact score.

| Model | Exact | SARI |
|---|---|---|
| Transformer (Geva et al., 2019) | 51.1 | 84.5 |
| SEQ2SEQ$_{BERT}$ | 53.6 | 85.3 |
| LaserTagger$_{AR}$ (no SWAP) | 46.4 | 80.4 |
| LaserTagger$_{FF}$ | 52.2 | 84.1 |
| LaserTagger$_{AR}$ | **53.8** | **85.5** |

Table 2: Sentence fusion results on DfWiki.

**Comparison against Baselines.** Table 2 lists the results for the DfWiki dataset. We obtain new SOTA results with LaserTagger$_{AR}$, outperforming the previous SOTA 7-layer Transformer model from Geva et al. (2019) by 2.7% Exact score and 1.0% SARI score. We also find that the pretrained SEQ2SEQ$_{BERT}$ model yields nearly as good performance, demonstrating the effectiveness of unsupervised pretraining for generation tasks. The performance of the tagger is impaired significantly when leaving out the SWAP tag due to the model's inability to reconstruct 10.5% of the training set.

**Impact of Dataset Size.** We also study the effect of the training data size by creating four increasingly smaller subsets of DfWiki (see Fig. 5a).[4] When data size drops to 450 or 4 500 examples, LaserTagger still performs surprisingly well, clearly outperforming the SEQ2SEQ$_{BERT}$ baseline.

## 5.2 Split and Rephrase

The reverse task of sentence fusion is the *split-and-rephrase* task, which requires rewriting a long sentence into two or more coherent short sentences.

**Data.** We use the WikiSplit dataset (Botha et al., 2018), which consists of 1M human-editor created examples of sentence splits, and follow the dataset split suggested by the authors. Using the phrase vocabulary of size 500 yields a 31% coverage of the targets from the training set (top phrases shown in Table 1). The lower coverage compared to DfWiki suggests a higher amount of noise (due to Wikipedia-author edits unrelated to splitting).

**Results.** Botha et al. (2018) report results using a one-layer, bi-directional LSTM (cell size 512) with attention and a copying mechanism (See

[2]We use the implementation available at git.io/fj8Av, setting $\beta = 1$ for deletion (Geva et al., 2019).

[3]For smaller datasets, a smaller vocabulary size may yield better results, but for simplicity, we do not optimize the size separately for each dataset.

[4]For simplicity, we use the same phrase vocabulary of size 500 computed using the validation set of 46K examples for all experiments. Note that even though some subsampled training sets contain less than 46K examples, using the same vocabulary does not give the taggers an unfair advantage over the baselines, because the tagger will never predict a phrase it has not seen in the training data.




(a) Sentence Fusion on DfWiki.

(b) Split and Rephrase on WikiSplit.

Figure 5: SARI score as a function of the training-data size for three models. Unless we have tens of thousands of training examples, the tagging approach clearly outperforms the seq2seq baseline.

| Model | BLEU | Exact | SARI |
|---|---|---|---|
| seq2seq (Botha et al., 2018) | 76.0 | 14.6 | 60.6 |
| SEQ2SEQ$_{BERT}$ | **76.7** | 15.1 | **62.3** |
| LASERTAGGER$_{FF}$ | 76.0 | 14.4 | 61.3 |
| LASERTAGGER$_{AR}$ | 76.3 | **15.2** | 61.7 |

Table 3: Results on the WikiSplit dataset.

| Model | BLEU-4 | Exact | SARI | ROUGE-L |
|---|---|---|---|---|
| Filippova et al. (2015) | 26.7 | 0.0 | 36.2 | 70.3 |
| Clarke and Lapata (2008) | 28.5 | 0.3 | 41.5 | 77.5 |
| Cohn and Lapata (2008) | 5.1 | 0.1 | 27.4 | 40.7 |
| Rush et al. (2015) | 16.2 | 0.0 | 35.6 | 62.5 |
| SEQ2SEQ$_{BERT}$ | 8.3 | 0.1 | 32.1 | 52.7 |
| LASERTAGGER$_{FF}$ | 33.7 | 1.5 | 44.2 | 81.9 |
| LASERTAGGER$_{AR}$ | **35.6** | **3.8** | **44.8** | **82.8** |

Table 4: Results on summarization.

et al., 2017).[5] The results are shown in Table 3. SEQ2SEQ$_{BERT}$ and LASERTAGGER$_{AR}$ yield similar performance with each other, and they both outperform the seq2seq model with a copying mechanism from Botha et al. (2018).

We again studied the impact of training-data size by subsampling the training set, see Figure 5b. Similar to the previous experiment, the LASERTAGGER methods degrade more gracefully when reducing training-data size, and start to outperform the seq2seq baseline once going below circa 10k examples. The smallest training set for LASERTAGGER$_{AR}$ contains merely 29 examples. Remarkably, the model is still able to learn something useful that generalizes to unseen test examples, reaching a SARI score of 53.6% and predicting 5.2% of the targets exactly correctly. The following is an example prediction by the model:

**Source:** Delhi Public Library is a national depository library in Delhi , India , it has over 35 branches across the state .

**Prediction:** Delhi Public Library is a national depository library in Delhi , India . ⟨::::⟩ It has over 35 branches across the state .

Here the model has picked the right comma to replace with a period and a sentence separator.

[5]Botha et al. (2018) report only BLEU but they kindly shared with us their model's predictions, allowing us to compute the Exact and SARI score for their method. Similar to their work, we used NLTK v3.2.2 for the BLEU computation.

### 5.3 Abstractive Summarization

The task of summarization is to reduce the length of a text while preserving its meaning.

**Dataset.** We use the dataset from Toutanova et al. (2016), which contains 6,168 short input texts (one or two sentences) and one or more human-written summaries. The human experts were not restricted to just deleting words when generating a summary, but were allowed to also insert new words and reorder parts of the sentence, which makes this dataset particularly suited for *abstractive summarization* models.

We set the size of the phrase vocabulary to 500, as for the other tasks, and extract the phrases from the training partition. With a size of 500, we are able to cover 89% of the training data.

**Evaluation Metrics.** In addition to the metrics from the previous sections, we report ROUGE-L (Lin, 2004), as this is a metric that is commonly used in the summarization literature. ROUGE-L is a recall-oriented measure computed as the longest common sub-sequence between a reference summary and a candidate summary.

**Results.** Table 4 compares our taggers against seq2seq baselines and systems from the literature.[6] Filippova et al. (2015) and Clarke and Lapata (2008) proposed deletion-based approaches;

[6]Results are extracted from Toutanova et al. (2016).

the former uses a seq2seq network, the latter formulates summarization as an optimization problem that is solved via integer-linear programming. Cohn and Lapata (2008) proposed an early approach to abstractive summarization via a parse-tree transducer. Rush et al. (2015) developed a neural seq2seq model for abstractive summarization.

In line with the results on the subsampled fusion/splitting datasets (Figure 5), the tagger significantly outperforms all baselines. This shows that even though a text-editing approach is not well-suited for extreme summarization examples (a complete paraphrase with zero lexical overlap), in practice, already a limited paraphrasing capability is enough to reach good empirical performance.

Note that the low absolute values for the Exact metric are expected, since there is a very large number of acceptable summaries.

### 5.4 Grammatical Error Correction (GEC)

GEC requires systems to identify and fix grammatical errors in a given input text.

**Data.** We use a recent benchmark from a shared task of the 2019 Building Educational Applications workshop, specifically from the Low Resource track[7] (Bryant et al., 2019). The publicly available set has 4,384 ill-formed sentences together with gold error corrections, which we split 9:1 into a training and validation partition. We again create the phrase vocabulary from the 500 most frequently added phrases in the training partition, which gives us a coverage of 40% of the training data.

**Evaluation Metrics and Results.** We report precision and recall, and the task's main metric $F_{0.5}$, which gives more weight to the precision of the corrections than to their recall.

Table 5 compares our taggers against two baselines. Again, the tagging approach clearly outperforms the BERT-based seq2seq model, here by being more than seven times as accurate in the prediction of corrections. This can be accounted to the seq2seq model's much richer generation capacity, which the model can not properly tune to the task at hand given the small amount of training data. The tagging approach on the other hand is naturally suited to this kind of problem.

We also report the best-performing method by Grundkiewicz et al. (2019) from the shared task for informational purposes. They train a Transformer model using a dataset which is augmented by 100 million synthetic examples and 2 million Wikipedia edits, whereas we only use 4,384 sentences from the provided training dataset.

[7] www.cl.cam.ac.uk/research/nl/bea2019st/

| Model | $P$ | $R$ | $F_{0.5}$ |
|---|---|---|---|
| Grundkiewicz et al. (2019) | *70.19* | *47.99* | *64.24* |
| SEQ2SEQ$_{\text{BERT}}$ | 6.13 | 14.14 | 6.91 |
| LASERTAGGER$_{\text{FF}}$ | 44.17 | 24.00 | 37.82 |
| LASERTAGGER$_{\text{AR}}$ | **47.46** | **25.58** | **40.52** |

Table 5: Results on grammatical-error correction. Note that Grundkiewicz et al. (2019) augment the training dataset of 4,384 examples by 100 million synthetic examples and 2 million Wikipedia edits.

| batch size | LASERTAGGER$_{\text{FF}}$ | LASERTAGGER$_{\text{AR}}$ | SEQ2SEQ$_{\text{BERT}}$ |
|---|---|---|---|
| 1 | 13 | 535 | 1,773 |
| 8 | 47 | 668 | 8,279 |
| 32 | 149 | 1,273 | 27,305 |

Table 6: Inference time (in ms) across various batch sizes on GPU (Nvidia Tesla P100) averaged across 100 runs with random inputs.

### 5.5 Inference time

Getting state-of-the-art results often requires using larger and more complex models. When running a model in production, one cares not only about the accuracy but also the inference time. Table 6 reports latency numbers for LASERTAGGER models and our most accurate seq2seq baseline. As one can see, the SEQ2SEQ$_{\text{BERT}}$ baseline is impractical to run in production even for the smallest batch size. On the other hand, for a batch size 8, LASERTAGGER$_{\text{AR}}$ is already 10x faster than comparable-in-accuracy SEQ2SEQ$_{\text{BERT}}$ baseline. This difference is due to the former model using a 1-layer decoder (instead of 12 layers) and no encoder-decoder cross attention. We also tried training SEQ2SEQ$_{\text{BERT}}$ with a 1-layer decoder but it performed very poorly in terms of accuracy. Finally, LASERTAGGER$_{\text{FF}}$ is more than 100x faster while being only a few accuracy points below our best reported results.

### 5.6 Qualitative evaluation

To assess the qualitative difference between the outputs of LASERTAGGER and SEQ2SEQ$_{\text{BERT}}$, we analyzed the texts generated by the models on the test sets of the four tasks. We inspected the respective worst predictions from each model according to BLEU and identified seven main error patterns,

| Error type | LaserTagger | Seq2Seq$_{BERT}$ | Example |
|---|---|---|---|
| Imaginary words | not affected | affected | In: … Zenica (Cyrillic: "Зеница") is …<br>Out: … Zenica (Cyrillic: "**gratulation**еница") is … |
| Repeated phrases | not affected | affected | In: I'm your employee, to serve on your company.<br>Out: I'm your **company**, to serve on your **company**. |
| Premature end-of-sentence | less affected | affected | In: By the way, my favorite football team is Manchester United, they …<br>Out: By the way, my favorite football team is. |
| Hallucinations | less affected | affected | In: Tobacco smokers may also experience …<br>Out: **anthropology** smokers may also experience … |
| Coreference issues | affected | affected | In: She is the daughter of Alistair Crane … who secretly built …<br>Out: She is the daughter of Alistair Crane … ⟨::::⟩ **She** secretly built … |
| Misleading rephrasing | affected | affected | In: … postal service was in no way responsible …<br>Out: … postal service **was responsible** … |
| Lazy sentence splitting | affected | not affected | In: Home world of the Marglotta located in the Sagittarius Arm.<br>Out: Home world of the Marglotta . ⟨::::⟩ Located in the Sagittarius Arm. |

Table 7: Main error patterns observed in the output of the tagging and seq2seq models on their test sets (all tasks).

two of which are specific to the seq2seq model, and one being specific to LaserTagger.

This illustrates that LaserTagger is less prone to errors compared to the standard seq2seq approach, due to the restricted flexibility of its model. Certain types of errors, namely imaginary words and repeated phrases, are virtually impossible for the tagger to make. The likelihood of others, such hallucination and abrupt sentence ending, is at least greatly reduced.

In Table 7, we list the error classes and refer to Appendix A for more details on our observations.

## 6 Conclusions

We proposed a text-editing approach to text-generation tasks with high overlap between input and output texts. Compared to the seq2seq models typically applied in this setting, our approach results in a simpler sequence-tagging problem with a much smaller output tag vocabulary. We demonstrated that this approach has comparable performance when trained on medium-to-large datasets, and clearly outperforms a strong seq2seq baseline when the number of training examples is limited. Qualitative analysis of the model outputs suggests that our tagging approach is less affected by the common errors of the seq2seq models, such as hallucination and abrupt sentence ending. We further demonstrated that tagging can speed up inference by more than two orders of magnitude, making it more attractive for production applications.

**Limitations.** Arbitrary word reordering is not feasible with our approach, although limited reordering can be achieved with deletion and insertion operations, as well as custom tags, such as `SWAP` (see Section 3.1). To enable more flexible reordering, it might be possible to apply techniques developed for phrase-based machine translation. Another limitation is that our approach may not be straightforward to apply to languages that are morphologically richer than English, where a more sophisticated realizer might be needed to adjust, e.g., the cases of the words.

In future work, we would like to experiment with more light-weight tagging architectures (Andor et al., 2016) to better understand the trade-off between inference time and model accuracy.

## Acknowledgments

We would like to thank Enrique Alfonseca, Idan Szpektor, and Orgad Keller for useful discussions.

## References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2442–2452.

Jan A Botha, Manaal Faruqui, John Alex, Jason Baldridge, and Dipanjan Das. 2018. Learning to split and rephrase from wikipedia edit history. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Christopher Bryant, Mariano Felice, Øistein E Andersen, and Ted Briscoe. 2019. The bea-2019 shared task on grammatical error correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 52–75.

Shamil Chollampatt and Hwee Tou Ng. 2018. Neural quality estimation of grammatical error correction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2528–2539.

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression: An integer linear programming approach. *J. Artif. Intell. Res.*, 31:399–429.

Trevor Cohn and Mirella Lapata. 2008. Sentence compression beyond word deletion. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 137–144.

Franck Dernoncourt, Mohammad Ghassemi, and Walter Chang. 2018. A repository of corpora for summarization. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Yue Dong, Zichao Li, Mehdi Rezagholizadeh, and Jackie Chi Kit Cheung. 2019. EditNTS: An neural programmer-interpreter model for sentence simplification through explicit editing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Katja Filippova, Enrique Alfonseca, Carlos A Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with LSTMs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 360–368.

Katja Filippova and Michael Strube. 2008. Dependency tree based sentence compression. In *Proceedings of the Fifth International Natural Language Generation Conference*, pages 25–32.

Tao Ge, Furu Wei, and Ming Zhou. 2018. Fluency boost learning and inference for neural grammatical error correction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1055–1065.

Mor Geva, Eric Malmi, Idan Szpektor, and Jonathan Berant. 2019. DiscoFuse: A large-scale dataset for discourse-based sentence fusion. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3443–3455.

Roman Grundkiewicz, Marcin Junczys-Dowmunt, and Kenneth Heafield. 2019. Neural grammatical error correction systems with unsupervised pre-training on synthetic data. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 252–263.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640.

Jiatao Gu, Changhan Wang, and Jake Zhao. 2019. Levenshtein transformer. *arXiv preprint arXiv:1905.11006*.

Zhijing Jin, Di Jin, Jonas Mueller, Nicholas Matthews, and Enrico Santus. 2019. Unsupervised text style transfer via iterative matching and translation. *arXiv preprint arXiv:1901.11333*.

Hongyan Jing and Kathleen McKeown. 2000. Cut and paste based text summarization. In *1st Meeting of the North American Chapter of the Association for Computational Linguistics*.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2014. The AMU system in the CoNLL-2014 shared task: Grammatical error correction by data-intensive and feature-rich statistical machine translation. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 25–33.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Shubha Guha, and Kenneth Heafield. 2018. Approaching neural grammatical error correction as a low-resource machine translation task. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 595–606.

Sudhanshu Kasewa, Pontus Stenetorp, and Sebastian Riedel. 2018. Wronging a right: Generating better errors to improve grammatical error detection. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4977–4983.

Kevin Knight and Ishwar Chander. 1994. Automated postediting of documents. In *Proceedings of the 12th National Conference on Artificial Intelligence, Seattle, WA, USA, July 31 - August 4, 1994, Volume 1.*, pages 779–784.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81.

Yang Liu. 2019. Fine-tune bert for extractive summarization. *arXiv preprint arXiv:1903.10318*.

Amit Moryossef, Yoav Goldberg, and Ido Dagan. 2019. Step-by-step: Separating planning from realization in neural data-to-text generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2267–2277.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Ranking sentences for extractive summarization with reinforcement learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1747–1759.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14.

Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. 2013. The CoNLL-2013 shared task on grammatical error correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–12.

Nikola I Nikolov and Richard HR Hahnloser. 2018. Large-scale hierarchical alignment for author style transfer. *arXiv preprint arXiv:1810.08237*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization.

Vinodkumar Prabhakaran, Camilla Griffiths, Hang Su, Prateek Verma, Nelson Morgan, Jennifer L. Eberhardt, and Dan Jurafsky. 2018. Detecting institutional dialog acts in police traffic stops. *Transactions of the Association for Computational Linguistics*, 6:467–481.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2018. Data-to-text generation with content selection and planning. *arXiv preprint arXiv:1809.00582*.

Sudha Rao and Joel Tetreault. 2018. Dear sir or madam, may i introduce the GYAFC dataset: Corpus, benchmarks and metrics for formality style transfer. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 129–140.

Marek Rei. 2017. Semi-supervised multitask learning for sequence labeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2121–2130.

Marek Rei, Mariano Felice, Zheng Yuan, and Ted Briscoe. 2017. Artificial error generation with machine translation and syntactic patterns. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 287–292.

Alla Rozovskaya, Kai-Wei Chang, Mark Sammons, Dan Roth, and Nizar Habash. 2014. The Illinois-Columbia system in the CoNLL-2014 shared task. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 34–42.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with Pointer-Generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112.

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. Abstractive document summarization with a Graph-Based attentional neural model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1171–1181.

Kristina Toutanova, Chris Brockett, Ke M Tran, and Saleema Amershi. 2016. A dataset and evaluation metrics for abstractive compression of sentences and short paragraphs. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 340–350.

Staal A Vinterbo. 2002. A note on the hardness of the k-ambiguity problem. *Technical Report DSG-T R-2002-006*.

Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2018. Learning neural templates for text generation. *arXiv preprint arXiv:1808.10122*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2048–2057.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016a. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016b. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415.

Xingxing Zhang and Mirella Lapata. 2017. Sentence simplification with deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 584–594.

Wei Zhao, Liang Wang, Kewei Shen, Ruoyu Jia, and Jingming Liu. 2019. Improving grammatical error correction via pre-training a copy-augmented architecture with unlabeled data. pages 156–165.

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. 2010. A monolingual tree-based translation model for sentence simplification. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1353–1361.

# Answer-guided and Semantic Coherent Question Generation in Open-domain Conversation

**Weichao Wang , Shi Feng , Daling Wang , Yifei Zhang**
School of Computer Science and Engineering, Northeastern University, Shenyang, China
wangwecha@gmail.com
{fengshi,wangdaling,zhangyifei}@cse.neu.edu.cn

## Abstract

Generating intriguing question is a key step towards building human-like open-domain chatbots. Although some recent works have focused on this task, compared with questions raised by humans, significant gaps remain in maintaining semantic coherence with the posts, which may result in generating dull or deviated questions. We observe that the answer has strong semantic coherence to its question and post, which can be used to guide question generation. Thus, we devise two methods to further enhance semantic coherence between post and question under the guidance of answer. First, the coherence score between generated question and answer is used as the reward function in a reinforcement learning framework, to encourage the cases that are consistent with the answer in semantic. Second, we incorporate adversarial training to explicitly control question generation in the direction of question-answer coherence. The extensive experiments show that our two methods outperform state-of-the-art baseline algorithms with large margins in raising semantic coherent questions.

## 1 Introduction

Neural question generation (NQG) has been extensively studied because of its broad application in question answering (QA) systems (Tang et al., 2018; Duan et al., 2017), reading comprehension (Kim et al., 2019; Sun et al., 2018), and visual question answering (Fan et al., 2018). However, the research of open-domain conversational question generation (CQG) (Wang et al., 2018; Hu et al., 2018) is still in its infancy. Different from traditional NQG task, the goal of CQG is to enhance the interactiveness and persistence of chit-chatting. Raising appropriate questions in the conversational system is essential because a mature system should well interact with users by not only responding posts but also asking questions (Li et al., 2016b). Furthermore, asking intriguing questions can proactively probe the users to go deeper and further into the conversational topic (Yu et al., 2016).

The existing CQG methods usually suffer from **dullness** and **deviation** problem, because the semantic coherence is easily disrupted. Wang et al. (2018) first studied the CQG task with PMI model to predict topic words in the inference, which may incorporate noise and generate deviated questions. Hu et al. (2018) proposed an aspect-based question generation model with an encoder-decoder method that did not consider the diversity and might raise dull questions.

Taking Table 1 as a motivating example, the topic of this conversation is about cooking and dishes. For the initial post about enjoying cooking, the appropriate response is a question asking about the special dishes or making appetizers, and the answer tells the exact dish, i.e. "*beef*". If we ignore the answer information, a dull question (e.g. "*What do you mean?*") or a deviated question (e.g. "*How about going singing?*") may be raised.

Table 1 shows that there exists semantic coherence in the post-question-answer conversational thread. The performance limitations of previous CQG literature partially lie in that they only consider the post to generate the question, but the answer that contains rich conversational semantics is ignored. In reading comprehension task, answer information could help to identify appropriate interrogative words (Kim et al., 2019), or copy appropriate phrase by position-aware attention (Sun et al., 2018). Similarly, answer information is leveraged to established the connection between QA and QG (Tang et al., 2018). Although the answer semantics are very effective in traditional NQG tasks, they have never been utilized for generating questions in conversations. Unlike NQG that



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5066–5076,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

● Semantic coherence ● Dullness ● Deviation

| Post |
| --- |
| I like cooking more and more. |
| **Question candidates** |
| What are your special dishes ? ✓ |
| Are you good at making appetizers ? ✓ |
| What do you mean? ✗ |
| How about going singing? ✗ |
| **Answer** |
| Wow, I am only skillful in cooking beef. |

Table 1: A motivating example of CQG task, including the post, question, and answer. In question candidate set, we list two intriguing questions and the other two unappropriate questions. The green words indicate the semantic coherence phenomenon. The blue and red words state the dull and deviated responses respectively. Note that the questions asking about the special dishes or making appetizers are all reasonable, and there is no exact match between question and answer.

the question and answer are manually designed according to the context and its goal is to generate an **accurate** and **exact matching** question, the triples are all **casual** and **non-goal-oriented** conversational utterances in CQG as shown in Table 1.

In this paper, we intend to utilize semantics in answer to guide conversational question generation. To the best of our knowledge, there is no prior work incorporating coherence between questions and answers into the CQG task. It is quite challenging to include question-answer semantic coherence into the objective function of supervised CQG models because the coherence also depends on answer information that does not exist in the inference process, i.e. we do not know the answer to the question in advance in a live chatbot. The unavailable answers during inference is the main difference between traditional NQG and CQG task.

To tackle these challenges, we propose two separate learning frameworks based on reinforcement learning (RL) (Sutton et al., 2000) and generative adversarial network (GAN) (Goodfellow et al., 2014) respectively. For the RL framework, we utilize semantic coherence as an estimation of question quality, with inspiration of the fact that the semantic (topic) is coherent in the same question-answer pair. We propose a GRU-MatchPyramid model that incorporates a bi-directional GRU into MatchPyramid model (Pang et al., 2016) to capture higher level's word semantic. Then we employ the pretrained GRU-MatchPyramid to measure the coherence between a question and its corresponding answer. After that, the coherence model is regarded as the environment in an RL framework for optimization, which will guide the learning process to penalize the dull and deviated questions. In addition, the environment is not required during inference.

For the GAN framework, we incorporate adversarial training to explicitly improve coherence between questions and answers. We jointly train a question generator and a discriminator (i.e. GRU-MatchPyramid model) that measures the semantic coherence between generated questions (as well as ground-truth questions) and answers. Only a well-trained generator is needed in inference.

The conditional variational autoencoders (C-VAE) is effective in capturing the diversity of valid responses (Zhao et al., 2017). Based on the advantage of CVAE, we further incorporate prior knowledge of question type to generate questions with reasonable type. Finally, we integrate CVAE into RL framework and GAN frameworks respectively, and propose two novel CQG models, dubbed as RL-CVAE and A-CVAE. Our contributions are summarized in three folds:

- We are the first to incorporate the answer factor into CQG task. The semantic coherence between questions and answers could guide the model to generate more appropriate questions.
- We propose two novel answer-guided semantic coherent CQG models, i.e. RL-CVAE and A-CVAE. The answer information is exploited with reinforcement learning and adversarial learning respectively.
- We conduct comprehensive experiments on our crawled Reddit conversation dataset to confirm the effectiveness of our proposed model. The experimental results demonstrate that our models consistently outperform strong baseline methods.

## 2 Related Work

The neural network-based methods have been applied successfully in natural language generation problems, such as text summarization (See et al., 2017; Chen et al., 2019), machine translation (Bahdanau et al., 2015; Vaswani et al., 2017) and question generation. Traditional NQG tasks are explored in reading comprehension (Kim et al.,

2019; Sun et al., 2018), QA systems (Tang et al., 2018; Duan et al., 2017). In such tasks, the answer is known and is part of the input to the generated question. The answer information could help to predict reasonable interrogative words (Kim et al., 2019), or perceive appropriate phrase by modelling the relative distance between the context words and the answer (Sun et al., 2018), or bridge the relevance between QA and QG tasks (Tang et al., 2018).

Furthermore, NQG can also be deployed as chatbot components, such as visual question answering (Fan et al., 2018), task-oriented dialogues (Li et al., 2016b), and open-domain dialogue generation (Wang et al., 2018; Hu et al., 2018). Li et al. (2016b) devised several handcrafted templates, which is not applicable to open-domain conversational systems. Wang et al. (2018) first studied on open-domain CQG, and they devised soft and hard typed decoders by capturing different roles of different word types, and used PMI model to predict topic words. Hu et al. (2018) proposed an aspect-based question generation problem, and used an encoder-decoder model to generate aspect-specific questions.

The RL has been widely applied to conversational generation (Zhou and Wang, 2018; Li et al., 2016c; Heeman et al., 2012; Lemon et al., 2014), and the GAN has been successfully applied to text generation (Li et al., 2018; Yu et al., 2017). However, these methods did not explore the answer information. Inspired by the important role of answer information in traditional NQG tasks, we intend to use RL and GAN to solve the open-domain CQG problem. To the best of our knowledge, we are the first to incorporate answer and conversational content into the both RL and GAN frameworks for this task.

# 3 Proposed Models

## 3.1 Problem Formalization

Following the conversation question generation paradigm (Wang et al., 2018), our method has a similar task setting where a question is generated based on a post utterance. For the particularity of our task, we further leverage the question type to control and interpret the generated question, and take advantage of the semantic coherence guided by answer information to improve the quality of generated questions.

The conversational content ($U$, $q$, $l$, $a$), consists of post utterance $U$, response utterance (i.e. question) $q$ with its question type $l$, and answer utterance $a$ to question $q$. In training, we aim to estimate the probability $p(q, l|U)$ utilizing C-VAE, and use reinforcement learning and adversarial learning to explore the answer information $a$ with GRU-MatchPyramid measuring the coherence. In inference, given a new post utterance $U$, we can generate an appropriate question $q'$ and question type $l'$ according to the generation probability. Note that the answer utterance $a$ is only used in training process, and is unknown in inference process.

## 3.2 Conditional Variational Autoencoder

The conditional variational autoencoder (C-VAE) (Sohn et al., 2015) is an extention of sequence-to-sequence model, and proves to be very effective in promoting the diversity in conversation generation (Serban et al., 2017). Besides, prior linguistic features (e.g. dialogue acts (Zhao et al., 2017) or sentiment (Shen et al., 2017)) have been incorporated into CVAE for controlling and interpreting dialogue generation. Inspired by previous works, we apply CVAE model to generate diverse and meaningful questions for CQG task. Furthermore, question type can provide semantic hints that enhance the semantic relevance for generating controllable questions. So we incorporate question type into CVAE for controlling question generation with reasonable type.

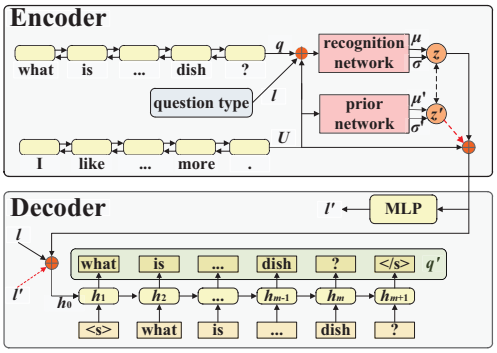


Figure 1: The CVAE model incorporated with question type embedding. $\bigoplus$ denotes the vector concatenation operation. In training process, approximated poster latent variable $z$ obtained from recognition network, together with true question type $l$ and post utterance $U$ are concatenated for decoder process. At the same time, red dashed arrows refer to inference process, where we replace $z$ with prior latent variable $z'$ obtained from prior network, and replace $l$ with predicted question type $l'$ in decoder process.

The CVAE model introduces a latent variable $z$ to capture the distribution over responses, so the generation probability is defined as $p(q, z, l|U) = p(q|z, U, l)p(l|z, U)p(z|U)$. As shown in Figure 1, the encoder neural network is formulated as the bi-directional GRU model, which is used to encode post $U$ and question $q$. The question type $l$ is represented by a real-valued, low dimensional vector which is learnt through training. Note that the reason why we use the actual question type $l$ during training is to generate appropriate question consistent with the ground-truth in content and question type simultaneously. By assuming $z$ follows multivariate Gaussian distribution, the prior network parameterized by $\theta_P$ is introduced to approximate $p_P(z'|U) \sim \mathcal{N}(\mu', \sigma'^2\mathbf{I})$. The recognition network parameterized by $\theta_R$ is introduced to approximate $p_R(z|U, q, l) \sim \mathcal{N}(\mu, \sigma^2\mathbf{I})$. The latent variables are sampled by reparameterization trick. The question type prediction network parameterized by $\theta_L$ is introduced to approximate $p_L(l|U, z)$. The decoder neural network parameterized by $\theta_D$ is formulated as a 1-layer GRU model, which is used to approximate $p_D(q|z, U, l)$. During training, $z$ obtained by recognition network, together with $l$ and $U$ are used to approximate $p_D(q|z, U, l)$. The response decoder then gets sematic representation $h_1$,...,$h_{m+1}$ with predicting the words of response. During inference, $z'$ obtained by prior network, together with predicted question type $l'$ and $U$ are regarded as input of decoder process, which corresponds to red dashed arrows in Figure 1.

Finally, the CVAE model incorporated with question type is trained by maximizing both the variational lower bound and question type prediction accuracy:

$$\begin{aligned}\mathcal{L}_{CVAE} = &-KL(p_R(z|U, q, l)||p_P(z|U)) \\ &+E_{p_R(z|U,q,l)}logp_D(q|z, U, l) \\ &+E_{p_R(z|U,q,l)}logp_L(l|z, U)\end{aligned} \quad (1)$$

We use KL annealing (Bowman et al., 2016) and bag-of-word loss (Zhao et al., 2017) to avoid the vanishing latent variable problem.

### 3.3 GRU-MatchPyramid Model

In this paper, we formulate measuring the semantic coherence between questions and answers as a text matching task.

The MatchPyramid (Pang et al., 2016) considers both semantic representation and semantic interactions in text matching. However, for large-scale conversational text, it is more essential to capture the word's higher level semantic rather than superficial meaning, as quite a part of sentence pairs in conversation do not have the same keywords but only contain semantic-related words (e.g. "*dish*" in ground truth question and "*cooking*", "*beaf*" in answer of Table 1). So we introduce bi-directional GRU into MatchPyramid model's word level to capture higher level's word semantic, and propose a new GRU-MatchPyramid model.

For question $q$'s $i$-th word $w_i$ with embedding $e_{w_i}$, and answer $a$'s $j$-th word $v_j$ with embedding $e_{v_j}$, the matching matrix $M$ is defined as:

$$M_{ij} = BiGRU(e_{w_i}) \otimes BiGRU(e_{v_j}) \quad (2)$$

where $M_{ij}$ is semantic coherence degree between word $w_i$ and word $v_j$, $BiGRU(e_{w_i})$ and $BiGRU(e_{v_j})$ are words' high-level semantic representation via bi-directional GRU, and $\otimes$ is dot product operation. Then a one-layer convolutional neural network is used to extract the matching pattern based on $M$, and the features matrix is obtained. Afterwards, a max-over-time pooling operation (Collobert et al., 2011) over the features matrix obtained by convolutional operation is used to capture the most important information $M'$. We flatten $M'$ and use a MLP model to generate the semantic coherence feature $f$. The semantic feature $M'$, coherence feature $f$ and coherence score $s$ are defined as:

$$M' = CNN(M) \quad (3)$$

$$f = \varphi(W_f \cdot flatten(M') + b_f) \quad (4)$$

$$s = \varphi(W_s f + b_s) \quad (5)$$

where $\varphi$ is the rectified linear units activation function, i.e. ReLU, $W_f$, $b_f$, $W_s$, $b_s$ are trainable parameters.

### 3.4 RL-CVAE

Inspired by reinforcement learning in controlling characteristics of generated responses (e.g. grammatical coherence (Li et al., 2016c), or sentiment expression (Zhou and Wang, 2018)), we conjecture that promoting the question-answer semantic coherence via reinforcement learning can help generate appropriate questions. The supervised learning cannot be applied to calculate question-answer semantic coherence, as it suffers from the non-differentiable problem caused by the sampling-based output decoding procedure.

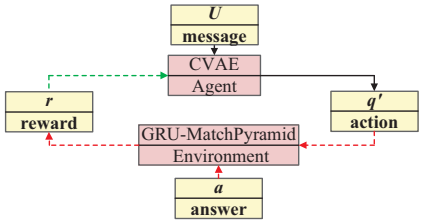


Figure 2: The CVAE with reinforcement learning (RL-CVAE). We formulate the CVAE model as an agent and pretrained GRU-MatchPyramid model as an environment in an RL framework. Solid arrows present the process of generating action $q'$ (i.e. generated question) on the condition of state $U$ (i.e. post). Red dashed arrows refer to agent's interaction process with the environment, where reward $r$ is obtained by using action $q'$ and answer $a$. Green dashed arrow refer to policy gradient optimization process with reward $r$.

To explicitly promote the semantic coherence between questions and answers, we propose a RL-CVAE model, as shown in Figure 2. Specifically, RL-CVAE is initialized by the pretrained CVAE model that acts as an agent in reinforcement learning. The action $q'$ is the generated question, and the state is denoted by the post $U$ and will affect the action. The policy of agent takes the form of CVAE, and it will output a probability distribution over actions conditioned on given states $U$. The pretrained GRU-MatchPyramid model acts as an environment, which considers answer $a$ and question $q'$ as input, and calculates the reward function to output reward value $r$. The reward value $r$ (i.e. semantic coherence score) can be interpreted as quality estimation of $q'$ so that it can guide the learning process to reward the cases that have strong coherence to the answer, and to penalize the cases that have little coherence to the answer (i.e. the dull and deviated cases).

We utilize pretrained GRU-MatchPyramid model as the measuring method to evaluate semantic coherence between $q'$ and $a$. When training GRU-MatchPyramid model, we use negative sampling method to construct the training data, which has been successfully applied in retrieval-based chatbot (Zhou et al., 2018; Zhang et al., 2018) and text matching task (Liu et al., 2016). Specifically, for each question (ground-truth) and answer pair $(q, a)$ in the conversation generation training data, we randomly sample five negative questions $q^-$, chosen from other posts' ground-truth responses. The training objective is that the score of $(q, a)$ should be larger than $(q^-, a)$ with at least $\triangle$ threshold. The loss function is defined as:

$$\mathcal{L}_{coh} = max(\triangle - s(q, a) + s(q^-, a), 0) \quad (6)$$

where $s$ is semantic coherence score between question and answer as defined in Equation 5.

After obtaining a generated question $q'$, we get coherence score ${r_0}^+ = s(q', a)$ between generated question and answer. However, ${r_0}^+$ has a wide range of value that it could not be used as reward value in reinforcement learning directly. Similar with GRU-MatchPyramid's training process, we create a coherence score set $R^-$ with negative examples, and use min-max scaling to normalize the coherence score ${r_0}^+$. Specifically, we randomly select $n$ negative questions ${q_1}^-, {q_2}^-, ... {q_n}^-$ from other posts's ground-truth responses, and then calculate their coherence scores with $a$ via Equation 5, i.e. $R^- = ({r_1}^-, {r_2}^-, ..., {r_n}^-)$. The final coherence score $r(q', a)$ is then defined as follows:

$$r = \frac{{r_0}^+ - min(R^-, {r_0}^+)}{max(R^-, {r_0}^+) - min(R^-, {r_0}^+)} \quad (7)$$

where $min(R^-, {r_0}^+)$ and $max(R^-, {r_0}^+)$ are the minimum value and maximum value of set $(R^-, {r_0}^+)$. We use the policy gradient methods (Sutton et al., 2000) for optimization with the expected reward defined as:

$$J(\theta) = \mathbb{E}_{p(q,z,l|U)}[r(q, a)] \quad (8)$$

The gradient is estimated as:

$$\nabla J(\theta) = r(q, a) \nabla log p(q, z, l|U) \quad (9)$$

### 3.5 A-CVAE

The GAN (Goodfellow et al., 2014) is effective in controlling attributes of generated text (e.g. sentiment (Logeswaran et al., 2018) or thematic (Li et al., 2018)). Based on our assumption that C-QG task can benefit from question-answer semantic coherence, we propose an A-CVAE model with a question generator and a question-answer coherence discriminator to further improve the quality of generated questions.

In Figure 3, the generator (i.e. CVAE model) is used to generate questions $q'$, and the discriminator (i.e. GRU-MatchPyramid model) is incorporated to learn the matching score between generated question $q'$ (as well as ground-truth question $q$)

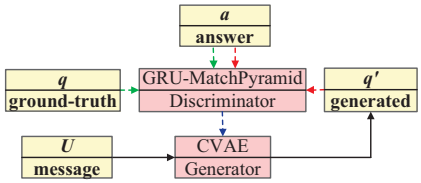


Figure 3: The CVAE with adversarial learning (A-CVAE). We formulate the CVAE as a generator and GRU-MatchPyramid as a discriminator using GAN. Solid arrows present the generation process of question $q'$ conditions on post utterance $U$. Red dashed arrows refer to the adversarial learning process of negative instance (i.e. $q'$ and answer $a$), and green dashed arrows refer to the positive instance (i.e. ground-truth question $q$ and $a$). Blue dashed arrow refer to the back-propagation of the discriminator to the generator.

and answer $a$. In the discriminator, we treat $q'$ as the negative sample incoherent to answer $a$, and treat $q$ as the positive sample, which on the contrary is coherent to $a$. Then the loss from this discriminator is back-propagated to the decoder process of the generator to improve the coherence.

Inspired by (Li et al., 2018), to better perform gradient calculation in adversarial process, the decoder's state sequence $(h_1, h_2, ..., h_m)$ shown in Figure 1 is used to represent generated question $q'$. To align dimension between $q'$ and $a$, we conduct nonlinear transformation on state $h_i$:

$$h'_i = \varphi(W_d h_i + b_d) \tag{10}$$

After transformation, the $q'$ is then denoted as $q' = (h'_1, h'_2, ..., h'_m)$. According to Equation 5, we get the semantic coherence score $s_g$ between $q'$ and $a$ (i.e. $s_g = s(q', a)$), and score $s_t$ between the ground-truth question $q$ and $a$ (i.e. $s_t = s(q, a)$).

Following the routine of improved wasserstein generative adversarial networks, i.e. WGAN-PG (Gulrajani et al., 2017), the discriminator is trained to measure the semantic coherence degree between generated (as well as ground truth) questions and answers, according to the semantic coherence scores $s_g$ and $s_t$, with minimizing the objective function:

$$\mathcal{L}_D = s_g - s_t \tag{11}$$

Note that the discriminator which utilizes the answer is only applied during the training process, because the answer information is unavailable during inference.

The overall objective of generator is to minimize

$$\mathcal{L}_G = -\mathcal{L}_{CVAE} - \lambda s_g \tag{12}$$

with respect to parameters of the CVAE, where $\mathcal{L}_{CVAE}$ is defined in Equation 1. $\lambda$ is a balancing parameter. Intuitively, during training the generator is trying to generate answer-coherent questions that would confuse the discriminator as to these questions' origin (i.e. ground-truth or generated) by maximizing $\mathcal{L}_D$, so that the discriminator is forced to become a strong semantic coherence evaluator to distinguish these questions' origin by minimizing $\mathcal{L}_D$, until both the generator and discriminator can not improve any more. In this way, the adversarial learning can push generator to generate questions indistinguishable from ground-truth.

## 4 Experiment

### 4.1 Experiment Setup

We extract our experiment corpus on Reddit[1], and the comment utterances are crawled for our experiment evaluation[2]. Inspired by (Fan et al., 2018), we categorize questions into 9 types, i.e. "what", "when", "where","who","why" "how","can (could)","do (did, does)","is (am, are, was, were)" according to interrogative words that are the most significant features to distinguish question types. We filter out the crawled data whose questions are not included in these 9 types, because these data occupy a small proportion in the crawled conversation dataset, and it is difficult to learn satisfying question type representation. Ultimately, we collected 1,164,345 pieces of data, and each of which has a post, a question, and an answer. We randomly selected 30,000 triples for validation and another 30,000 triples for testing. The average number of words in post/response/answer is 18.84/19.03/19.30 respectively. We choose top 40,000 frequent words as the vocabulary.

We use the 200d Glove embedding (Pennington et al., 2014) pretrained on Wikipedia 2014 dataset as word embedding. In CVAE, the size of encoder state is set to 300, and the size of decoder state is set to 400. We set the size of latent Gaussian variable to 200, the size of question type to 100, and the size of mini-batch to 30. The prior network and question type prediction network have a hidden layer whose dimension is 400. All weights are initialized by the xavier method (Glorot and

[1]http://www.reddit.com

[2]The dataset is available at https://drive.google.com /drive/folders/1wNG30YPHiMc_ZNyE3BH5wa1uVtR8l1pG

Bengio, 2010). The dataset is tokenized using the NLTK tokenizer. We optimize our model using Adam (Kingma and Ba, 2014) with learning state of 0.001 and gradient clipping at 5. We use KL-annealing strategy with the temperature varying from 0 to 1 and increased by 1/60k after each iteration of batch update.

For the GRU-MatchPyramid model, the bi-directional GRU model for encoding question and answer has a hidden state of 300 in size. In the convolutional operation, the kernel size is set to (3,3), with the stride size to 2.

In RL-CVAE, we set threshold $\triangle$ to 1 in GRU-MatchPyramid's training, and set number $n$ to 100 in min-max scaling. The pretrained CVAE model is obtained by bag-of-words loss along with KL annealing to 0.5 after training 30k batches. Then we use policy loss to obtain the final model with KL weight kept at 0.5. In A-CVAE, we set balancing parameter $\lambda$ in generator's loss to 0.1, and train the discriminator one time for every five times training of generator.

### 4.2 Baselines

**Seq2Seq**: A simple sequence-to-sequence model with attention mechanism.
**CVAE**: Conditional variational autoencoders with an auxiliary bag-of-words loss is used to generate diverse responses (Zhao et al., 2017).
**STD&HTD**: The STD uses soft typed decoder by estimating words' distribution over word types, and HTD uses hard typed decoder by specifying the type of each word explicitly (Wang et al., 2018). Because these two methods depend on PMI, we collected about 100,000 post-response pairs from Reddit to estimate the probabilities in PMI. The HTD is the state-of-the-art model for C-QG task.

### 4.3 Question Generation Evaluation

**Perplexity**: The perplexity (PPL) (Vinyals and Le, 2015) could evaluate whether generated responses satisfy the grammatical rules, and lower value reflects better fluency.
**RUBER**: RUBER (Referenced metric and Unreferenced metric Blended Evaluation Routine) (Tao et al., 2018) has shown a high correlation with human annotation in open-domain conversation response evaluation. RUBER measures the similarity between generated responses and ground-truth responses as referenced metric, and measures the relatedness as unreferenced metric. The similarity is calculated via cosine similarity, and the relatedness is obtained by a neural network pretrained via a utterance matching method. Then the geometric averaging (RubG) and arithmetic averaging (RubA) are obtained via these two metrics, and higher value reflects better semantic coherence.
**Distinct2** (Li et al., 2016a): We count numbers of distinct bigrams in the generated responses. The ratio (Dist2) is obtained by dividing the counted number by the total number of generated bigrams, and higher value reflects better diversity.

| Question Generation Evaluation | | | | |
|---|---|---|---|---|
| Models | RubA | RubG | Dist2 | PPL |
| Seq2Seq | 0.614 | 0.574 | 0.008 | 63.02 |
| CVAE | 0.682 | 0.649 | 0.112 | 20.39 |
| STD | 0.658 | 0.613 | 0.010 | 28.75 |
| HTD | 0.689 | 0.654 | 0.114 | 26.02 |
| CVAE (qt) | 0.688 | 0.652 | 0.114 | 20.03 |
| A-CVAE | 0.715 | 0.661 | 0.123 | 19.51 |
| RL-CVAE | **0.720** | **0.668** | **0.185** | **16.93** |

Table 2: Evaluation result of question generation.

As shown in Table 2, CVAE(qt) incorporates question type information, and slightly improves the performance compared with CVAE, because it could help generate questions with reasonable type. Furthermore, answer information could further improve the overall performance. First, RL-CVAE and A-CVAE perform fairly well with lower perplexities, because they could generate fluent questions. Second, they can obtain higher RubG and RubA scores, showing that taking advantage of coherence between question and answer could further enhance semantic coherence between post and question. Third, they obtain higher distinct values, because answer information provides extra clues to guide question generation.

Our RL-CVAE performs better than A-CVAE. This is because the decoder's hidden states are utilized to represent generated question word tokens in A-CVAE, which is asymmetric to the ground-truth question in adversarial process. Instead, the question sentence is observed in RL-CVAE after the model arrives at the end of a sequence, and it is then used to calculate coherence degree directly.

### 4.4 Effect of GRU-MatchPyramid

We use mean average precision (MAP) to evaluate the performance of GRU-MatchPyramid model. MAP is the metric to measure the accuracy of tex-

| Post | Generated Question | Answer |
|---|---|---|
| that's definitely a third party seller at amazon and not amazon . | Seq2Seq: is it a good thing?<br>CVAE: and how is this an official app?<br>STD: how do you like amazon?<br>HTD: are you heavily subscribed to amazon?<br>CVAE(qt): can you get the link?<br>i'm just using the amazon box.<br>**RL-CVAE: what's the difference between a third party seller and amazon ?**<br>**A-CVAE: does the third party seller ship amazon fulfillment?** | it's still third seller so amazon usually ships the item as it was shipped to them but in a box. i would buy directly from amazon. |
| i don't know why so many people don't realize the duke o death is only free for returning players. | Seq2Seq: what's your point?<br>CVAE: are you just about having fun?<br>STD: what's the difference?<br>HTD: are you returning player?<br>CVAE(qt): are there any members in your opinion?<br>**RL-CVAE: how much is it for normal players?**<br>**A-CVAE: how to realize that? i'm on the list of returning players.** | it's 660k for normal players. and in the newswire they said all the returning player stuff would be free for returning players. |

Table 4: Two cases comparison among different models.

| Models | MP | GRU-MatchPyramid |
|---|---|---|
| MAP | 0.793 | **0.826** |

Table 3: Evaluation result of GRU-MatchPyramid.

t matching, and it is the average of the maximum precisions at different recall values. As shown in Table 3, GRU-MatchPyramid model performs better than MatchPyramid (dubbed as MP) model in large scale conversational data, because of capturing higher level semantic of word. In this way, we get a better reward function in RL-CVAE, and a better discriminator in A-CVAE.

### 4.5 Question Answer Coherence Evaluation

To evaluate the degree of semantic coherence between generated questions and answers, we apply two evaluation methods: (1) cosine similarity: inspired by sentence-level similarity evaluation (Li et al., 2018), we use embedding-based metrics to calculate cosine similarity, and higher similarity implies better topic similarity. (2) matching score: we use pretrained GRU-MatchPyramid model to calculate the semantic coherence, and higher scores imply better semantic coherence.

As shown in Table 5, our proposed models could generate questions coherent to answers in inference process. The coherence of question and answer improved overall performance shown in Table 2 by a large margin.

### 4.6 Human Evaluation

We randomly sampled 500 cases and recruited three graduate students as human annotators. To

| Models | cosine | matching score |
|---|---|---|
| Seq2Seq | 0.494 | 5.304 |
| CVAE | 0.591 | 8.050 |
| STD | 0.540 | 6.882 |
| HTD | 0.593 | 8.063 |
| CVAE (qt) | 0.594 | 8.053 |
| A-CVAE | 0.612 | 8.420 |
| RL-CVAE | **0.617** | **8.512** |

Table 5: Evaluation result of question-answer semantic coherence.

| Models | A | S | W | Sum |
|---|---|---|---|---|
| seq2seq | 0.486 | 0.208 | 0.196 | 0.890 |
| CVAE | 0.458 | 0.484 | 0.408 | 1.350 |
| STD | 0.504 | 0.322 | 0.272 | 1.098 |
| HTD | 0.528 | 0.486 | 0.406 | 1.420 |
| CVAE(qt) | 0.462 | 0.508 | 0.468 | 1.438 |
| A-CVAE | 0.540 | 0.578 | 0.514 | 1.632 |
| RL-CVAE | **0.542** | **0.602** | **0.526** | **1.670** |

Table 6: Results of human evalution based on criteria of appropriateness (A), semantic coherence (S) and willingness to answer (W).

each annotator, we showed the same post utterance of a test example with all seven questions generated by different models. The seven questions were presented in random order, and the annotators were asked to evaluate whether each question satisfies criteria defined as: (1) **Appropriateness**: measures whether a question is reasonable in logic and grammar. (2) **Semantic coherence**: measures whether the question are coherent to the

given post. Incoherent questions include dull and deviated cases. (3) **Willingness to answer**: measures whether a user is willing to answer the question. This criterion is to justify how likely the generated questions can elicit further interactions.

We calculate the ratio of each criterion satisfied, and regard the sum as the evaluation result. The results are calculated by averaging the annotations from the three judges. We calculated the Fleiss' kappa (Fleiss, 1971) to measure inter-annotator agreement. Fleiss' kappa for **Appropriateness**, **Semantic coherence** and **Willingness to answer** is 0.486, 0.421 and 0.506 respectively, indicating "Moderate agreement" for all three criteria. As shown in Table 6, RL-CVAE and A-CVAE are consistently in line with the human perspective, especially in semantic coherence and willing to answer criteria. It indicates that with utilizing question-answer semantic coherence, the generated questions are more coherent to posts, and our models could help to drive the conversation to go further proactively. Compared with the state-of-the-art model HTD, our proposed models could effectively alleviate the dullness and deviation issues.

### 4.7 Case Study

As shown in Table 4, we list two cases from test corpus to compare different methods. The topics about post and answer in the two cases are about "*third-party seller of amazon*" and "*returning players*" respectively. The A-CVAE and RL-CVAE model could generate appropriate questions coherent to both posts and answers semantically, showing that answer information of training process helps to control and guide the question generation obviously. However, without exploring answer information, the baseline models may only generate dull questions (e.g. Seq2Seq), or deviated questions (e.g. CVAE, STD, and HTD). Furthermore, based on the given post and generated questions, we are more willing to answer the questions generated by our methods.

## 5 Conclusion

In this paper, we propose two novel models A-CVAE and RL-CVAE, which can generate semantic coherent questions under the guidance of answers in open-domain conversational system. For A-CVAE model, we use adversarial training to explicitly control question generation in the direction of question-answer coherence. For RL-CVAE, the coherence between questions and answers is leveraged as the reward function in an RL framework. Furthermore, our work of utilizing RL and GAN with answer information is general, and it can be easily extended to existing work.

## Acknowledgements

The work was supported by the National Key R&D Program of China under grant 2018YFB1004700, and National Natural Science Foundation of China (61872074, 61772122).

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations*.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*.

Yangbin Chen, Yun Ma, Xudong Mao, and Qing Li. 2019. Multi-task learning for abstractive and extractive summarization. *Data Science and Engineering*, 4(1):14–23.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Nan Duan, Duyu Tang, Peng Chen, and Ming Zhou. 2017. Question generation for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 866–874.

Zhihao Fan, Zhongyu Wei, Piji Li, Yanyan Lan, and Xuanjing Huang. 2018. A question type driven framework to diversify visual question generation. In *IJCAI*, pages 4048–4054.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron

Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. 2017. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777.

Peter A Heeman, Jordan Fryer, Rebecca Lunsford, Andrew Rueckert, and Ethan Selfridge. 2012. Using reinforcement learning for dialogue management policies: Towards understanding mdp violations and convergence. In *Thirteenth Annual Conference of the International Speech Communication Association*.

Wenpeng Hu, Bing Liu, Jinwen Ma, Dongyan Zhao, and Rui Yan. 2018. Aspect-based question generation. In *ICLR Workshop*.

Yanghoon Kim, Hwanhee Lee, Joongbo Shin, and Kyomin Jung. 2019. Improving neural question generation using answer separation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6602–6609.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Oliver Lemon, Srinivasan Chandrasekaran Janarthanam, and Verena Rieser. 2014. Reinforcement learning approaches to natural language generation in interactive systems. In *Natural Language Generation in Interactive Systems*, pages 151–175. Cambridge University Press.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. *Computer Science*.

Jiwei Li, Alexander H Miller, Sumit Chopra, Marc'Aurelio Ranzato, and Jason Weston. 2016b. Learning through dialogue interactions by asking questions. *arXiv preprint arXiv:1612.04936*.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016c. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*.

Juntao Li, Yan Song, Haisong Zhang, Dongmin Chen, Shuming Shi, Dongyan Zhao, and Rui Yan. 2018. Generating classical chinese poems via conditional variational autoencoder and adversarial training. In *EMNLP*, pages 3890–3900.

Pengfei Liu, Xipeng Qiu, Jifan Chen, and Xuanjing Huang. 2016. Deep fusion lstms for text semantic matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1034–1043.

Lajanugen Logeswaran, Honglak Lee, and Samy Bengio. 2018. Content preserving text generation with attribute controls. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 5103–5113. Curran Associates, Inc.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016. Text matching as image recognition. In *AAAI*, pages 2793–2799.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*.

Xiaoyu Shen, Hui Su, Yanran Li, Wenjie Li, Shuzi Niu, Yang Zhao, Akiko Aizawa, and Guoping Long. 2017. A conditional variational framework for dialog generation. *arXiv preprint arXiv:1705.00316*.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. 2015. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*, pages 3483–3491.

Xingwu Sun, Jing Liu, Yajuan Lyu, Wei He, Yanjun Ma, and Shi Wang. 2018. Answer-focused and position-aware neural question generation. In *EMNLP*, pages 3930–3939.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063.

Duyu Tang, Nan Duan, Zhao Yan, Zhirui Zhang, Yibo Sun, Shujie Liu, Yuanhua Lv, and Ming Zhou. 2018. Learning to collaborate for question answering and asking. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 1564–1574.

Chongyang Tao, Lili Mou, Dongyan Zhao, and Rui Yan. 2018. Ruber: An unsupervised method for automatic evaluation of open-domain dialog systems. In *AAAI*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Yansen Wang, Chenyi Liu, Minlie Huang, and Liqiang Nie. 2018. Learning to ask questions in open-domain conversational systems with typed decoders. In *ACL*.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Zhou Yu, Ziyu Xu, Alan W Black, and Alexander Rudnicky. 2016. Strategy and policy learning for non-task-oriented conversational systems. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 404–412.

Zhuosheng Zhang, Jiangtong Li, Pengfei Zhu, Hai Zhao, and Gongshen Liu. 2018. Modeling multi-turn conversation with deep utterance aggregation. *arXiv preprint arXiv:1806.09102*.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *ACL*, pages 654–664. Association for Computational Linguistics.

Xianda Zhou and William Yang Wang. 2018. Mojitalk: Generating emotional responses at scale. In *ACL*, pages 1128–1137.

Xiangyang Zhou, Lu Li, Daxiang Dong, Yi Liu, Ying Chen, Wayne Xin Zhao, Dianhai Yu, and Hua Wu. 2018. Multi-turn response selection for chatbots with deep attention matching network. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1118–1127.

# Read, Attend and Comment: A Deep Architecture for Automatic News Comment Generation

**Ze Yang**[†], **Can Xu**[◇], **Wei Wu**[◇], **Zhoujun Li**[†*]
[†]State Key Lab of Software Development Environment, Beihang University, Beijing, China
[◇]Microsoft Corporation, Beijing, China
{tobey, lizj}@buaa.edu.cn {wuwei, caxu}@microsoft.com

## Abstract

Automatic news comment generation is a new testbed for techniques of natural language generation. In this paper, we propose a "read-attend-comment" procedure for news comment generation and formalize the procedure with a reading network and a generation network. The reading network comprehends a news article and distills some important points from it, then the generation network creates a comment by attending to the extracted discrete points and the news title. We optimize the model in an end-to-end manner by maximizing a variational lower bound of the true objective using the back-propagation algorithm. Experimental results on two datasets indicate that our model can significantly outperform existing methods in terms of both automatic evaluation and human judgment.

## 1 Introduction

In this work, we study the problem of automatic news comment generation, which is a less explored task in the literature of natural language generation (Gatt and Krahmer, 2018). We are aware that numerous uses of these techniques can pose ethical issues and that best practices will be necessary for guiding applications. In particular, we note that people expect comments on news to be made by people. Thus, there is a risk that people and organizations could use these techniques at scale to feign comments coming from people for purposes of political manipulation or persuasion. In our intended target use, we explicitly disclose the generated comments on news as being formulated automatically by an entertaining and engaging chatbot (Shum et al., 2018). Also, we understand that the behaviors of deployed systems may need to be monitored and guided with methods, including post-processing techniques (van Aken et al., 2018). While there are risks with this kind of AI research, we believe that developing and demonstrating such techniques is important for understanding valuable and potentially troubling applications of the technology.

| |
|---|
| **Title:** FIFA rankings: France number one, Croatia and England soar, Germany and Argentina plummet |
| **Body (truncated):** World Cup glory has propelled France to the top of FIFA's latest world rankings, with the impact of Russia 2018 felt for better or worse among a number of football's heavyweight nations.<br>These are the first set of rankings released under FIFA's new formula that "relies on adding/subtracting points won or lost for a game to/from the previous point totals rather than averaging game points over a given time period".<br>FIFA world rankings: 1.France 2.Belgium 3. Brazil 4. Croatia 5. Uruguay 6. England 7. Portugal 8. Switzerland 9. Spain 10. Denmark |
| **Comment A:** If it's heavily based on the 2018 WC, hence England leaping up the rankings, how are Brazil at 3?<br>**Comment B:** England above Spain, Portugal and Germany. Interesting. |

Table 1: A news example from Yahoo!

Existing work on news comment generation includes preliminary studies, where a comment is generated either from the title of a news article only (Zheng et al., 2018; Qin et al., 2018) or by feeding the entire article (title plus body) to a basic sequence-to-sequence (s2s) model with an attention mechanism (Qin et al., 2018). News titles are short and succinct, and thus only using news titles may lose quite a lot of useful information in comment generation. On the other hand, a news article and a comment is not a pair of parallel text. The news article is much longer than the comment and contains much information that is irrelevant to the comment. Thus, directly applying the s2s model, which has proven effective in machine translation (Bahdanau et al., 2015), to the task of news comment generation is unsuitable, and may bring a lot of noise to generation. Both approaches

* Corresponding Author



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5077–5089,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

oversimplify the problem of news comment generation and are far from how people behave on news websites. In practice, people read a news article, draw attention to some points in the article, and then present their comments along with the points they are interested in. Table 1 illustrates news commenting with an example from Yahoo! News.[1] The article is about the new FIFA ranking, and we pick two comments among the many to explain how people behave in the comment section. First, both commenters have gone through the entire article, as their comments are built upon the details in the body. Second, the article gives many details about the new ranking, but both commenters only comment on a few points. Third, the two commenters pay their attention to different places of the article: the first one notices that the ranking is based on the result of the new world cup, and feels curious about the position of Brazil; while the second one just feels excited about the new position of England. The example indicates a "read-attend-comment" behavior of humans and sheds light on how to construct a model.

We propose a reading network and a generation network that generate a comment from the entire news article. The reading network simulates how people digest a news article, and acts as an encoder of the article. The generation network then simulates how people comment the article after reading it, and acts as a decoder of the comment. Specifically, from the bottom to the top, the reading network consists of a representation layer, a fusion layer, and a prediction layer. The first layer represents the title of the news with a recurrent neural network with gated recurrent units (RNN-GRUs) (Cho et al., 2014) and represents the body of the news through self-attention which can model long-term dependency among words. The second layer forms a representation of the entire news article by fusing the information of the title into the representation of the body with an attention mechanism and a gate mechanism. The attention mechanism selects useful information in the title, and the gate mechanism further controls how much such information flows into the representation of the article. Finally, the third layer is built on top of the previous two layers and employs a multi-label classifier and a pointer network (Vinyals et al., 2015) to predict a bunch of salient spans (e.g., words, phrases, and sentences, etc.) from the article. With the reading network, our model comprehends the news article and boils it down to some key points (i.e., the salient spans). The generation network is an RNN language model that generates a comment word by word through an attention mechanism (Bahdanau et al., 2015) on the selected spans and the news title. In training, since salient spans are not explicitly available, we treat them as a latent variable, and jointly learn the two networks from article-comment pairs by optimizing a lower bound of the true objective through a Monte Carlo sampling method. Thus, training errors in comment prediction can be back-propagated to span selection and used to supervise news reading comprehension.

We conduct experiments on two large scale datasets. One is a Chinese dataset published recently in (Qin et al., 2018), and the other one is an English dataset built by crawling news articles and comments from Yahoo! News. Evaluation results on the two datasets indicate that our model can significantly outperform existing methods on both automatic metrics and human judgment.

Our contributions are three-folds: (1) proposal of "read-attend-comment" procedure for news comment generation with a reading network and a generation network; (2) joint optimization of the two networks with an end-to-end learning approach; and (3) empirical verification of the effectiveness of the proposed model on two datasets.

## 2 Related Work

News comment generation is a sub-task of natural language generation (NLG). Among various NLG tasks, the task studied in this paper is most related to summarization (Rush et al., 2015; Nallapati et al., 2016; See et al., 2017) and product review generation (Tang et al., 2016; Dong et al., 2017). However, there is stark difference between news comment generation and the other two tasks: the input of our task is an unstructured document, while the input of product review generation is structured attributes of a product; and the output of our task is a comment which often extends the content of the input with additional information, while the output of summarization is a condensed version of the input that contains the main information from the original. Very recently, there emerge some studies on news comment generation. For example, Zheng et al. (2018) propose

[1] https://www.yahoo.com/news/fifa-rankings-france-number-one-112047790.html

a gated attention neural network model to generate news comments from news titles. The model is further improved by a generative adversarial net. Qin et al. (2018) publish a dataset with results of some basic models. Different from all the existing methods, we attempt to comprehend the entire news articles before generation and perform end-to-end learning that can jointly optimize the comprehension model and the generation model.

Our model is partially inspired by the recent success of machine reading comprehension (MRC), whose prosperity can be attributed to an increase of publicly available large scale annotated datasets, such as SQuAD (Rajpurkar et al., 2016, 2018) and MS Marco (Nguyen et al., 2016) etc. A great number of models have been proposed to tackle the MRC challenges, including BiDAF (Seo et al., 2016), r-net (Wang et al., 2017), DCN (Xiong et al., 2016), Document Reader (Chen and Bordes, 2017), QANet (Yu et al., 2018), and s-net (Tan et al., 2018) etc. Our work can be viewed as an application of MRC to a new NLG task. The task aims to generate a comment for a news article, which is different from existing MRC tasks whose goal is to answer a question. Our learning method is also different from those in the MRC works.

## 3 Approach

### 3.1 Problem Formalization

Suppose we have a dataset $\mathcal{D} = \{(T_i, B_i, C_i)\}_{i=1}^{N}$, where the $i$-th triple $(T_i, B_i, C_i)$ consists of a news title $T_i$, a news body $B_i$, and a comment $C_i$. Our goal is to estimate a probability distribution $P(C|T, B)$ from $\mathcal{D}$, and thus, given a new article $(T, B)$ with $T$ the news title and $B$ the news body, we can generate a comment $C$ following $P(C|T, B)$.

### 3.2 Model Overview

Figure 1 illustrates the architecture of our model. In a nutshell, the model consists of a reading network and a generation network. The reading network first represents a news title and a news body separately in a representation layer, then forms a representation of the entire article by fusing the title into the body through a fusion layer, and finally distills some salient spans from the article by a prediction layer. The salient spans and the news title are then fed to the generation network to synthesize a comment. With the two networks, we can factorize the generation probability $P(C|T, B)$ as $P(S|T, B) \cdot P(C|S, T)$, where $S = (s_1, \ldots, s_w)$ refers to a set of spans in $B$, $P(S|T, B)$ represents the reading network, and $P(C|S, T)$ refers to the generation network.

### 3.3 Reading Network

In the representation layer, let $T = (t_1, \ldots, t_n)$ be a news title with $t_j$ the $j$-th word, and $B = (b_1, \ldots, b_m)$ be the associated news body with $b_k$ the $k$-th word, we first look up an embedding table and represent $t_j$ and $b_k$ as $e_{T,j} \in \mathbb{R}^{d_1}$ and $e_{B,k} \in \mathbb{R}^{d_1}$ respectively, where $e_{T,j}$ and $e_{B,k}$ are randomly initialized, and jointly learned with other parameters. Different from the title, the body is long and consists of multiple sentences. Hence, to emphasize positional information of words in the body, we further expand $e_{B,k}$ with $o_{B,k}$ and $s_{B,k}$, where $o_{B,k}, s_{B,k} \in \mathbb{R}^{d_2}$ are positional embeddings with the former indexing the position of $b_k$ in its sentence and the latter indicating the position of the sentence in the entire body. The representation of $b_k$ is then given by $\hat{e}_{B,k} = \text{MLP}([e_{B,k}; o_{B,k}; s_{B,k}])$, where $\hat{e}_{B,k} \in \mathbb{R}^{d_1}$, $\text{MLP}(\cdot)$ refers to a multi-layer perceptron with two layers, and $[\cdot;\cdot;\cdot]$ means the concatenation of the three arguments.

Starting from $\mathcal{E}_T = (e_{T,1}, \ldots, e_{T,n})$ and $\mathcal{E}_B = (\hat{e}_{B,1}, \ldots, \hat{e}_{B,m})$ as initial representations of $T$ and $B$ respectively, the reading network then transforms $T$ into a sequence of hidden vectors $\mathcal{H}_T = (h_{T,1}, \ldots, h_{T,n})$ with a recurrent neural network with gated recurrent units (RNN-GRUs) (Cho et al., 2014). In the meanwhile, $B$ is transformed to $\mathcal{H}_B = (h_{B,1}, \ldots, h_{B,m})$ with the $k$-th entry $h_{B,k} \in \mathbb{R}^{d_1}$ defined as $\text{MLP}([\hat{e}_{B,k}; c_{B,k}])$ (two layers) and $c_{B,k}$ is an attention-pooling vector calculated by a scaled dot-product attention (Vaswani et al., 2017) denoted as *dot-att*$(\mathcal{E}_B, \hat{e}_{B,k})$:

$$
\begin{aligned}
c_{B,k} &= \textstyle\sum_{j=1}^{m} \alpha_{k,j} \hat{e}_{B,j}, \\
\alpha_{k,j} &= \exp(s_{k,j}) / \textstyle\sum_{l=1}^{m} \exp(s_{k,l}), \\
s_{k,j} &= (\hat{e}_{B,k}^{\top} \hat{e}_{B,j}) / \sqrt{d_1}.
\end{aligned} \tag{1}
$$

In Equation (1), a word is represented via all words in the body weighted by their similarity. By this means, we try to capture the dependency among words in long distance.

The fusion layer takes $\mathcal{H}_T$ and $\mathcal{H}_B$ as inputs, and produces $\mathcal{V} = (v_1, \ldots, v_m)$ as new representation of the entire news article by fusing $\mathcal{H}_T$ into $\mathcal{H}_B$. Specifically, $\forall h_{B,k} \in \mathcal{H}_B$, we first let $h_{B,k}$

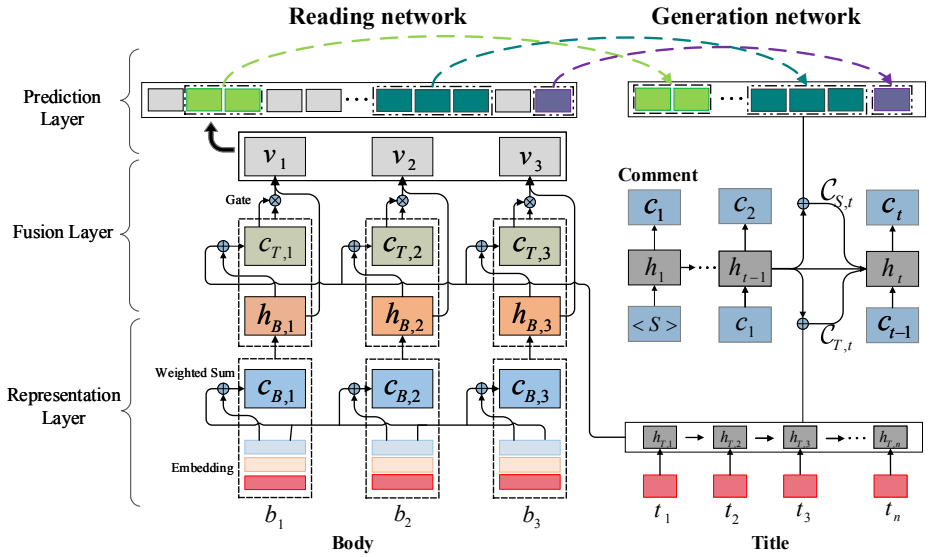


Figure 1: Architecture of our model. The black solid arrows represent differentiable operations and the dashed arrows are non-differentiable operations which represent distilling points from news body.

attend to $\mathcal{H}_T$ and form a representation $c_{T,k} = \textit{dot-att}(\mathcal{H}_T, h_{B,k})$, where $\textit{dot-att}(\mathcal{H}_T, h_{B,k})$ is parameterized as Equation (1). With this step, we aim to recognize useful information in the title. Then we combine $c_{T,k}$ and $h_{B,k}$ as $v_k \in \mathbb{R}^{d_1}$ with a gate $g_k$ which balances the impact of $c_{T,k}$ and $h_{B,k}$ and filters noise from $c_{T,k}$. $v_k$ is defined by:

$$\begin{aligned} v_k &= h_{B,k} + g_k \odot c_{T,k}, \\ g_k &= \sigma(W_g[h_{B,k}; c_{T,k}]), \end{aligned} \tag{2}$$

The top layer of the reading network extracts a bunch of salient spans based on $\mathcal{V}$. Let $S = ((a_1, e_1), \ldots, (a_w, e_w))$ denote the salient spans, where $a_i$ and $e_i$ refer to the start position and the end position of the $i$-th span respectively, we propose detecting the spans with a multi-label classifier and a pointer network. Specifically, we formalize the recognition of $(a_1, \ldots, a_w)$ as a multi-label classification problem with $\mathcal{V}$ as an input and $L = (l_1, \ldots, l_m)$ as an output, where $\forall k \in \{1, \ldots, m\}$, $l_k = 1$ means that the $k$-th word is a start position of a span, otherwise $l_k = 0$. Here, we assume that $(a_1, \ldots, a_w)$ are independent with each other, then the multi-label classifier can be defined as $m$ binary classifiers with the $k$-th classifier given by

$$\hat{y}_k = \text{softmax}(\text{MLP}_k(v_k)), \tag{3}$$

where $\text{MLP}_k(\cdot)$ refers to a two-layer MLP, and $\hat{y}_k \in \mathbb{R}^2$ is a probability distribution with the first entry as $P(l_k = 0)$ and the second entry as $P(l_k = 1)$. The advantage of the approach is that it allows us to efficiently and flexibly detect a variable number of spans from a variable-length news article, as there is no dependency among the $m$ classifiers, and they can be calculated in parallel.

Given $a_k$, the end position $e_k$ is recognized via a probability distribution $(\alpha_{a_k,1}, \ldots, \alpha_{a_k,m})$ which is defined by a pointer network:

$$\begin{aligned} \alpha_{a_k,j} &= \exp(s_{a_k,j}) / \textstyle\sum_{l=1}^{m} \exp(s_{a_k,l}), \\ s_{a_k,j} &= V^\top \tanh(W_v v_j + W_h h_{a_k,1}), \\ h_{a_k,1} &= \text{GRU}(h_0, [c_0; v_{a_k}]), \end{aligned} \tag{4}$$

where $h_0 = att(\mathcal{V}, r)$ is an attention-pooling vector based on parameter $r$:

$$\begin{aligned} h_0 &= \textstyle\sum_{j=1}^{m} \beta_j \cdot v_j, \\ \beta_j &= \exp(\beta'_j) / \textstyle\sum_{l=1}^{m} \exp(\beta'_l), \\ \beta'_j &= V_1^\top \tanh(W_{v,1} v_j + W_{h,1} r), \end{aligned} \tag{5}$$

$c_0 = att(\mathcal{V}, h_0)$ is defined in a similar way with $r$ replaced by $h_0$.

Let us denote $(a_1, \ldots, a_w)$ as $start$ and $P(l_i = 1)$ as $p_i$, then $P(S|T, B)$ can be formulated as

$$P(S|T,B) = \prod_{k=1}^{w} [p_{a_k} \cdot \alpha_{a_k,e_k}] \prod_{i \notin start, 1 \le i \le m} [1 - p_i]. \tag{6}$$

In practice, we recognize the $i$-th word as a start position if $p_i > 1 - p_i$, and determine the associated end position by $\arg\max_{1 \le k \le m} \alpha_{i,k}$. Note that we do not adopt a pointer network to detect the start positions, because in this case, either we have to set a threshold on the probability distribution, which is sensitive to the length of the news

article and thus hard to tune, or we can only pick a fixed number of spans for any articles by ranking the probabilities. Neither of them is favorable.

### 3.4 Generation Network

With $S = ((a_1, e_1), \ldots, (a_w, e_w))$ the salient spans, $\mathcal{V} = (v_1, \ldots, v_m)$ the representation of the news article, and $\mathcal{H}_T$ the representation of the news title given by the three layers of the reading network respectively, we define a representation of $S$ as $\mathcal{H}_S = (v_{a_1}, v_{a_1+1}, \ldots, v_{e_1}, \ldots, v_{a_w}, v_{a_w+1}, \ldots, v_{e_w})$. The generation network takes $\mathcal{H}_T$ and $\mathcal{H}_S$ as inputs and decodes a comment word by word via attending to both $\mathcal{H}_T$ and $\mathcal{H}_S$. At step $t$, the hidden state is $h_t = \text{GRU}(h_{t-1}, [e_{C,t-1}; \mathcal{C}_{T,t-1}; \mathcal{C}_{S,t-1}])$, $h_t \in \mathbb{R}^{d_1}$. $e_{C,t-1}$ is the embedding of the word generated at step $t$-1, $\mathcal{C}_{T,t-1} = att(\mathcal{H}_T, h_{t-1})$ and $\mathcal{C}_{S,t-1} = att(\mathcal{H}_S, h_{t-1})$ are context vectors that represent attention on the title and the spans respectively. $att(\cdot, \cdot)$ is defined as Equation (5).

With $h_t$, we calculate $\mathcal{C}_{T,t}$ and $\mathcal{C}_{S,t}$ via $att(\mathcal{H}_T, h_t)$ and $att(\mathcal{H}_S, h_t)$ respectively and obtain a probability distribution over vocabulary by $\mathcal{P}_t = \text{softmax}(V[h_t; \mathcal{C}_{T,t}; \mathcal{C}_{S,t}] + b)$. Let $C = (c_1, \ldots, c_o)$ be a comment where $\forall k \in \{1, \ldots, o\}$, $c_k$ is the index of the $k$-th word of $C$ in vocabulary, then $P(C|S, T)$ is defined as

$$\begin{aligned} P(C|S,T) &= P(c_1|S,T) \prod_{t=2}^{o} P(c_t|c_1, \ldots, c_{t-1}, S, T) \\ &= \mathcal{P}_1(c_1) \prod_{t=2}^{o} \mathcal{P}_t(c_t), \end{aligned}$$

where $\mathcal{P}_t(c_t)$ refers to the $c_t$-th entry of $\mathcal{P}_t$. In decoding, we define the initial state $h_0$ as an attention-pooling vector over the concatenation of $\mathcal{H}_T$ and $\mathcal{H}_S$ given by $att([\mathcal{H}_T; \mathcal{H}_S], q)$. $q$ is a parameter learned from training data.

### 3.5 Learning Method

We aim to learn $P(S|T, B)$ and $P(C|S, T)$ from $\mathcal{D} = \{(T_i, B_i, C_i)\}_{i=1}^{N}$, but $S$ is not explicitly available, which is a common case in practice. To address the problem, we treat $S$ as a latent variable, and consider the following objective:

$$\begin{aligned} \mathcal{J} &= \sum_{i=1}^{N} \log P(C_i|T_i, B_i) \\ &= \sum_{i=1}^{N} \log \Big( \sum_{S_i \in \mathcal{S}} P(S_i|T_i, B_i) P(C_i|S_i, T_i) \Big), \end{aligned} \tag{7}$$

where $\mathcal{S}$ refers to the space of sets of spans, and $S_i$ is a set of salient spans for $(T_i, B_i)$. Objective $\mathcal{J}$ is difficult to optimize, as logarithm is outside the summation. Hence, we turn to maximizing a lower bound of Objective $\mathcal{J}$ which is defined as:

$$\mathcal{L} = \sum_{i=1}^{N} \sum_{S_i \in \mathcal{S}} P(S_i|T_i, B_i) \log P(C_i|S_i, T_i) < \mathcal{J} \tag{8}$$

Let $\Theta$ denote all parameters of our model and $\frac{\partial \mathcal{L}_i}{\partial \Theta}$ denote the gradient of $\mathcal{L}$ on an example $(T_i, B_i, C_i)$, then $\frac{\partial \mathcal{L}_i}{\partial \Theta}$ is given by

$$\begin{aligned} \sum_{S_i \in \mathcal{S}} P(S_i|T_i, B_i) \Big[ &\frac{\partial \log P(C_i|S_i, T_i)}{\partial \Theta} + \\ &\log P(C_i|S_i, T_i) \frac{\partial \log P(S_i|T_i, B_i)}{\partial \Theta} \Big]. \end{aligned} \tag{9}$$

To calculate the gradient, we have to enumerate all possible $S_i$s for $(T_i, B_i)$, which is intractable. Thus, we employ a Monte Carlo sampling method to approximate $\frac{\partial \mathcal{L}_i}{\partial \Theta}$. Suppose that there are $J$ samples, then the approximation of $\frac{\partial \mathcal{L}_i}{\partial \Theta}$ is given by

$$\begin{aligned} \frac{1}{J} \sum_{n=1}^{J} \Big[ &\frac{\partial \log P(C_i|S_{i,n}, T_i)}{\partial \Theta} + \\ &\log P(C_i|S_{i,n}, T_i) \frac{\partial \log P(S_{i,n}|T_i, B_i)}{\partial \Theta} \Big], \end{aligned} \tag{10}$$

where $\forall n$, $S_{i,n}$ is sampled by first drawing a group of start positions according to Equation (3), and then picking the corresponding end positions by Equations (4)-(5). Although the Monte Carlo estimator is unbiased, it typically suffers from high variance. To reduce variance, we subtract baselines from $\log P(C_i|S_{i,n}, T_i)$. Inspired by (Mnih and Gregor, 2014), we introduce an observation-dependent baseline $\mathcal{B}_\psi(T_i, C_i)$ to capture the systematic difference in news-comment pairs during training. Besides, we also exploit a global baseline $\mathcal{B}$ to further control the variance of the estimator. The approximation of $\frac{\partial \mathcal{L}_i}{\partial \Theta}$ is then re-written as

$$\begin{aligned} \frac{1}{J} \sum_{n=1}^{J} \Big[ &\frac{\partial \log P(C_i|S_{i,n}, T_i)}{\partial \Theta} + \Big( \log P(C_i|S_{i,n}, T_i) \\ &- \mathcal{B}_\psi(T_i, C_i) - \mathcal{B} \Big) \frac{\partial \log P(S_{i,n}|T_i, B_i)}{\partial \Theta} \Big]. \end{aligned} \tag{11}$$

To calculate $\mathcal{B}_\psi(T_i, C_i)$, we first encode the word sequences of $T_i$ and $C_i$ with GRUs respectively, and then feed the last hidden states of the GRUs to a three-layer MLP. $\mathcal{B}$ is calculated as an average of $P(C_i|S_{i,n}, T_i) - \mathcal{B}_\psi(T_i, C_i)$ over the current mini-batch. The parameters

of the GRUs and the MLP are estimated via $\min_{(T_i,B_i)\in\text{mini-batch},1\le n\le J}[\log P(C_i|S_{i,n},T_i) - \mathcal{B}_\psi(T_i,C_i) - \mathcal{B}]^2$.

The learning algorithm is summarized in Algorithm 1. To speed up convergence, we initialize our model through pre-training the reading network and the generation network. Specifically, $\forall(T_i,B_i,C_i)\in\mathcal{D}$, we construct an artificial span set $\tilde{S}_i$, and learn the parameters of the two networks by maximizing the following objective:

$$\sum_{i=1}^{N}\log P(\tilde{S}_i|T_i,B_i)+\log P(C_i|T_i,\tilde{S}_i) \quad (12)$$

$\tilde{S}_i$ is established in two steps: first, we collect all associated comments for $(T_i,B_i)$, extract n-grams $(1\le \text{n}\le 6)$ from the comments, and recognize an n-gram in $B_i$ as a salient span if it exactly matches with one of the n-grams of the comments. Second, we break $B_i$ as sentences and calculate a matching score for a sentence and an associated comment. Each sentence corresponds to a group of matching scores, and if any one of them exceeds $0.4$, we recognize the sentence as a salient span. The matching model is pre-trained with $\{T_i,C_i\}_{i=1}^N$ with $C_i$ as a positive example and a randomly sampled comment from other news as a negative example. In the model, $T_i$ and $C_i$ are first processed by GRUs separately, and then the last hidden states of the GRUs are fed to a three-layer MLP to calculate a score.

**Algorithm 1:** Optimization Algorithm

**Input:** training data $\mathcal{D}$, initial learning rate $lr$, MaxStep, sample number $n$.
**Init:** $\Theta$
Construct $\{\tilde{S}_i\}_{i=1}^N$ and pre-train the model by maximizing Objective (12).
**while** *step* $<$ *MaxStep* **do**
  Randomly sample a mini-batch $k$ from $\mathcal{D}$.
  Compute distributions of start positions.
  **for** $n < J$ **do**
    Sample start positions.
    Compute distributions of end positions.
    Sample end positions.
    Compute the terms related to $S_{i,n}$ in Eq. (11).
  Compute $\mathcal{B}_\psi(C_i,T_i)$ and $\mathcal{B}$ and finish Eq. (11).
  Update the parameters of the model and $\mathcal{B}_\psi(C_i,T_i)$ with SGD.
**Output:** $\Theta$

## 4 Experiments

We test our model on two large-scale news commenting datasets.

| | | Train | Dev | Test |
|---|---|---|---|---|
| Tencent | # News | 191,502 | 5,000 | 1,610 |
| | Avg. # Cmts per News | 27 | 27 | 27 |
| Yahoo | # News | 152,355 | 5,000 | 3,160 |
| | Avg. # Cmts per News | 20.6 | 20.5 | 20.5 |

Table 2: Statistics of the two datasets.

### 4.1 Experimental Setup

The first dataset is a Chinese dataset built from Tencent News (news.qq.com) and published recently in (Qin et al., 2018). Each data point contains a news article which is made up of a title and a body, a group of comments, and some side information including upvotes and categories. Each test comment is labeled by two annotators according to a 5-scale labeling criteria presented in Table 3. All text in the data is tokenized by a Chinese word segmenter Jieba (https://github.com/fxsjy/jieba). The average lengths of news titles, news bodies, and comments are 15 words, 554 words and 17 words respectively. In addition to the Chinese data, we also build another dataset by crawling news articles and the associated comments from Yahoo! News. Besides upvotes and categories, side information in Yahoo data also includes paragraph marks, WIKI-entities, downvotes, abusevotes, and sentiment tagged by Yahoo!. Text in the data is tokenized by Stanford CoreNLP pipline (Manning et al., 2014). As pre-processing, we filter out news articles shorter than 30 words in the body and comments shorter than 10 words or longer than 100 words. Then, we remove news articles with less than 5 comments. If the number of comments of an article exceeds 30, we only keep top 30 comments with the most upvotes. On average, news titles, news bodies, and comments contain 12 words, 578 words and 32 words respectively. More information about Yahoo data can be found in Appendix A. After the pre-processing, we randomly sample a training set, a validation set, and a test set from the remaining data, and make sure that there is no overlap among the three sets. Table 2 summarizes the statistics of the two datasets. Note that we only utilize news titles, news bodies and comments to learn a generation model in this work, but both datasets allow modeling news comment generation with side information, which could be our future work.

Following (Qin et al., 2018), we evaluate the

| Score | Criteria |
|---|---|
| 5 | Rich in content; attractive; deep insights; new yet relevant viewpoints |
| 4 | Highly relevant with meaningful ideas |
| 3 | Less relevant; applied to other articles |
| 2 | Fluent/grammatical; irrelevant |
| 1 | Hard to read; Broken language; Only emoji |

Table 3: Human judgment criteria

performance of different models with both automatic metrics and human judgment. In terms of automatic evaluation, we employ BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), ROUGE (Lin, 2004), and CIDEr (Vedantam et al., 2015) as metrics on both data. Besides these metrics, Qin et al. (2018) propose human score weighted metrics including W-BLEU, W-METEOR, W-ROUGE and W-CIDEr. These metrics, however, requires human judgment on each comment in the test set. Thus, we only involve results w.r.t. these metrics in Tencent data. As Qin et al. (2018) do not publish their code for metric calculation, we employ a popular NLG evaluation project available at `https://github.com/Maluuba/nlg-eval`, and modify the scripts with the scores provided in the data according to the formulas in (Qin et al., 2018) to calculate all the metrics. In human evaluation, for each dataset, we randomly sample 500 articles from the test data and recruit three native speakers to judge the quality of the comments given by different models. For every article, comments from all models are pooled, randomly shuffled, and presented to the annotators. Each comment is judged by the three annotators under the criteria in Table 3.

### 4.2 Baselines

The following models are selected as baselines:

**Basic models**: the retrieval models and the generation models used in (Qin et al., 2018) including (1) IR-T and IR-TC: both models retrieve a set of candidate articles with associated comments by cosine of TF-IDF vectors. Then the comments are ranked by a convolutional neural network (CNN) and the top position is returned. The difference is that IR-T only utilizes titles, while IR-TC leverages both titles and news bodies; (2) Seq2seq: the basic sequence-to-sequence model (Sutskever et al., 2014) that generates a comment from a title; and (3) Att and Att-TC: sequence-to-sequence with attention (Bahdanau et al., 2015) in which the input is either a title (Att) or a concatenation of a title and a body (Att-TC). In Seq2seq, Att, and Att-TC, top 1 comment from beam search (beam size=5) is returned.

**GANN**: the gated attention neural network proposed in (Zheng et al., 2018). The model is further improved by a generative adversarial net.

We denote our model as "DeepCom" standing for "deep commenter", as it is featured by a deep reading-commenting architecture. All baselines are implemented according to the details in the related papers and tuned on the validation sets.

### 4.3 Implementation Details

For each dataset, we form a vocabulary with the top 30k frequent words in the entire data. We pad or truncate news titles, news bodies, and comments to make them in lengths of 30, 600, and 50 respectively. The dimension of word embedding and the size of hidden states of GRU in all models are set as 256. In our model, we set $d_1$ as 256 and $d_2$ (i.e., dimension of the position embedding in the reading network) as 128. The size of hidden layers in all MLPs is 512. The number of samples in Monte Carlo sampling is 1. In pre-training, we initialize our model with a Gaussian distribution $\mathcal{N}(0, 0.01)$ and optimize Objective (12) using AdaGrad (Duchi et al., 2011) with an initial learning rate 0.15 and an initial accumulator value 0.1. Then, we optimize $\mathcal{L}$ using stochastic gradient descent with a learning rate 0.01. In decoding, top 1 comment from beam search with a size of 5 is selected for evaluation. In IR-T and IR-TC, we use three types of filters with window sizes 1, 3, and 5 in the CNN based matching model. The number of each type of filters is 128.

### 4.4 Evaluation Results

Table 4 reports evaluation results in terms of both automatic metrics and human annotations. On most automatic metrics, DeepCom outperforms baseline methods, and the improvement is statistically significant (t-test with $p$-value $< 0.01$). The improvement on BLEU-1 and W-BLEU-1 is much bigger than that on other metrics. This is because BLEU-1 only measures the proportion of matched unigrams out of the total number of unigrams in the generated comments. In human evaluation, although the absolute numbers are different from those reported in (Qin et al., 2018) due to the difference between human judgements, the overall trend is consistent. In human evaluation, the values of Fleiss' kappa over all models are more

| Dataset | Models | METEOR | W-METEOR | Rouge_L | W-Rouge_L | CIDEr | W-CIDEr | BLEU-1 | W-BLEU-1 | Human | Kappa |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Tencent | IR-T | 0.107 | 0.086 | 0.254 | 0.217 | 0.018 | 0.014 | 0.495 | 0.470 | 2.43 | 0.64 |
| | IR-TC | 0.127 | 0.101 | 0.266 | 0.225 | 0.056 | 0.044 | 0.474 | 0.436 | 2.57 | 0.71 |
| | Seq2seq | 0.064 | 0.047 | 0.196 | 0.150 | 0.011 | 0.008 | 0.374 | 0.320 | 1.68 | 0.83 |
| | Att | 0.080 | 0.058 | 0.246 | 0.186 | 0.010 | 0.007 | 0.481 | 0.453 | 1.81 | 0.79 |
| | Att-TC | 0.114 | 0.082 | 0.299 | 0.223 | 0.023 | 0.017 | 0.602 | 0.551 | 2.26 | 0.69 |
| | GANN | 0.097 | 0.075 | 0.282 | 0.222 | 0.010 | 0.008 | 0.312 | 0.278 | 2.06 | 0.73 |
| | **DeepCom** | **0.181** | **0.138** | **0.317** | **0.250** | 0.029 | 0.023 | **0.721** | **0.656** | **3.58** | 0.65 |
| Yahoo | IR-T | 0.114 | - | 0.214 | - | 0.014 | - | 0.472 | - | 2.71 | 0.67 |
| | IR-TC | 0.117 | - | 0.219 | - | 0.017 | - | 0.483 | - | 2.86 | 0.61 |
| | Seq2seq | 0.061 | - | 0.203 | - | 0.011 | - | 0.365 | - | 2.26 | 0.68 |
| | Att | 0.075 | - | 0.217 | - | 0.017 | - | 0.462 | - | 2.29 | 0.78 |
| | Att-TC | 0.089 | - | 0.246 | - | 0.022 | - | 0.515 | - | 2.74 | 0.63 |
| | GANN | 0.079 | - | 0.228 | - | 0.019 | - | 0.496 | - | 2.52 | 0.64 |
| | **DeepCom** | 0.107 | - | **0.263** | - | **0.024** | - | **0.665** | - | **3.35** | 0.68 |

Table 4: Evaluation results on automatic metrics and human judgment. Human evaluation results are calculated by combining labels from the three judges. "Kappa" means Fleiss' kappa. Numbers in bold mean that improvement over the best baseline is statistically significant.

| Dataset | Metrics | No Reading | No Prediction | No Sampling | Full Model |
|---|---|---|---|---|---|
| Tencent | METEOR | 0.096 | 0.171 | 0.171 | 0.181 |
| | W-METEOR | 0.072 | 0.129 | 0.131 | 0.138 |
| | Rouge_L | 0.282 | 0.307 | 0.303 | 0.317 |
| | W-Rouge_L | 0.219 | 0.241 | 0.239 | 0.250 |
| | CIDEr | 0.012 | 0.024 | 0.026 | 0.029 |
| | W-CIDEr | 0.009 | 0.019 | 0.021 | 0.023 |
| | BLEU-1 | 0.426 | 0.674 | 0.667 | 0.721 |
| | W-BLEU-1 | 0.388 | 0.614 | 0.607 | 0.656 |
| Yahoo | METEOR | 0.081 | 0.092 | 0.102 | 0.107 |
| | Rouge_L | 0.232 | 0.245 | 0.244 | 0.263 |
| | CIDEr | 0.017 | 0.023 | 0.020 | 0.024 |
| | BLEU-1 | 0.490 | 0.531 | 0.609 | 0.665 |

Table 5: Model ablation results

than 0.6, indicating substantial agreement among the annotators. Although built in a complicated structure, GANN does not bring much improvement over other baseline methods, which demonstrates that only using news titles is not enough in comment generation. IR-TC and Att-TC represent the best retrieval model and the best generation model among the baselines on both datasets, implying that news bodies, even used in a simple way, can provide useful information to comment generation.

### 4.5 Discussions

**Ablation study:** We compare the full model of DeepCom with the following variants: (1) No Reading: the entire reading network is replaced by a TF-IDF based keyword extractor, and top 40 keywords (tuned on validation sets) are fed to the generation network; (2) No Prediction: the prediction layer of the reading network is removed, and thus the entire $\mathcal{V}$ is used in the generation network; and (3) No Sampling: we directly use the model pre-trained by maximizing Objective (12). Table 5 reports the results on automatic metrics. We can see that all variants suffer from performance drop and No Reading is the worst among the three variants. Thus, we can conclude that (1) span prediction cannot be simply replaced by TF-IDF based keyword extraction, as the former is based on a deep comprehension of news articles and calibrated in the end-to-end learning process; (2) even with sophisticated representations, one cannot directly feed the entire article to the generation network, as comment generation is vulnerable to the noise in the article; and (3) pre-training is useful, but optimizing the lower bound of the true objective is still beneficial.

To further understand why DeepCom is superior to its variants, we calculate BLEU-1 (denoted as $\text{BLEU}_{\text{span}}$) with the predicted spans and the ground truth comments in the test sets of the two data, and compare it with a baseline BLEU-1 (denoted as $\text{BLEU}_{\text{base}}$) which is calculated with the entire news articles and the ground truth comments. On Tencent data, $\text{BLEU}_{\text{span}}$ and $\text{BLEU}_{\text{base}}$ are 0.31 and 0.17 respectively, and the two numbers on Yahoo data are 0.29 and 0.16 respectively. This means that by extracting salient spans from news articles, we can filter out redundant information while keeping the points that people like to comment on, which explains why DeepCom is better than No Prediction. When comparing DeepCome with No Sampling, we find that spans in DeepCom are longer than those in No Sampling. In the test set of Tencent data, the average lengths of salient spans with and without sampling are 11.6 and 2.6 respectively, and the two numbers in Yahoo data are 14.7 and 2.3 respectively. Thus, DeepCom can leverage discourse-level informa-

| | | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| Tencent | IR-TC | 0.3% | 55.2% | 33.3% | 9.50% | 1.7% |
| | Att-TC | 19.3% | 49.5% | 18.5% | 11.7% | 1.0% |
| | DeepCom | 1.5% | 3.3% | 32.0% | 61.9% | 1.3% |
| Yahoo | IR-TC | 1.3% | 53.0% | 14.7% | 20.3% | 10.7% |
| | Att-TC | 20.3% | 22.3% | 22.6% | 32.1% | 2.7% |
| | DeepCom | 1.7% | 17.6% | 33.3% | 39.4% | 8.0% |

Table 6: Human score distributions

tion rather than a few single words or bi-grams in comment generation, which demonstrates the advantage of our learning method.

**Analysis of human annotations:** We check the distributions of human labels for DeepCom, Att-TC, and IR-TC to get insights on the problems these models suffer from. Table 6 shows the results. Most of the bad comments from IR-TC are labeled as "2", meaning that although IR-TC can give attractive comments with rich and deep content, its comments are easy to diverge from the news articles and thus be judged as "irrelevant". In terms of Att-TC, there are much more comments judged as "1" than the other two models, indicating that Att-TC often generates ill-formed sentences. This is because a news article and a comment are highly asymmetric in terms of both syntax and semantics, and thus the generation process cannot be simply modeled with an encoder-decoder structure. Bad cases from DeepCom concentrate on "3", reminding us that we need to further enrich the content of comments and improve their relevance in the future.

**Case Study:** Finally, to further understand our model, we visualize the predicted salient spans and the generated comment using a test example from Tencent dataset in Table 7. Due to space limitation, we truncate the body and only show three of the selected spans in the truncated body. The full article with the full set of spans and another test example from Yahoo! News dataset are shown in Appendix B. In spite of this, we can see that the model finds some interesting points after "reading" the article and synthesizes a comment along one of the spans (i.e. "Chinese Paladin 3"). More interestingly, the model extends the content of the article in the comment with "Luo Jin", who is Tiffany Tang's partner but is not talked about in the article. On the other hand, comments given by the baseline methods are either too general (Att-TC, the best generation baseline), or totally irrelevant with the article (IR-TC, the best retrieval baseline). The example demonstrates that our model can generate relevant and informative comments through analyzing and understanding news articles.

| |
|---|
| **Title:** 唐嫣为什么不演清宫剧? (Why Tiffany Tang never plays a role in a drama set in Qing dynasty?) |
| **Body (truncated):** 众多的戏剧影视作品让唐嫣获得不少人气，众多活动的多变造型树立其在大众心中百变小天后的形象。... 如果说唐嫣最美的造型是《仙剑奇侠传三》中的紫萱，那最丑的造型应该就是这个了吧！... (... The numerous television series and movies Tiffany Tang acted in have made her very popular, and her varied modelling in many activities has set her an image of "the queen of reinvention" in the hearts of the public. ... If her most beautiful role is Zi Xuan in "Chinese Paladin 3", then the most ugly one should be this! ... ) |
| **DeepCom:** 唐嫣罗晋的演技真的很好，特别喜欢她演《仙剑奇侠传》。我觉得这部剧挺好看。(Tiffany Tang and Luo Jin are really good actors. I especially like her role in "Chinese Paladin 3". I think the TV drama is worth watching.)<br>**Att-TC:** 我也是醉了(I have nothing to comment.)<br>**IR-TC:** 星爷和谁开撕过嘛，都是别人去撕星爷！(Stephen Chow never fights with others. It is others that fight with Stephen Chow!) |

Table 7: A Case from Tencent News dataset. The contents in the red box represent salient spans predicted by the reading network. The content in the blue box is a generated entity which is included in the salient spans.

## 5 Conclusions

We propose automatic news comment generation with a reading network and a generation network. Experimental results on two datasets indicate that the proposed model can significantly outperform baseline methods in terms of both automatic evaluation and human evaluation. On applications, we are motivated to extend the capabilities of a popular chatbot. We are aware of potential ethical issues with application of these methods to generate news commentary that is taken as human. We hope to stimulate discussion about best practices and controls on these methods around responsible uses of the technology.

## Acknowledgment

This work is supported in part by the National Natural Science Foundation of China (Grand Nos. U1636211, 61672081, 61370126), and the National Key R&D Program of China (No. 2016QY04W0802).

## References


Betty van Aken, Julian Risch, Ralf Krestel, and Alexander Löser. 2018. Challenges for toxic comment classification: An in-depth error analysis. *arXiv preprint arXiv:1809.07572*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72. Association for Computational Linguistics.

Danqi Chen and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111.

Li Dong, Shaohan Huang, Furu Wei, Mirella Lapata, Ming Zhou, and Ke Xu. 2017. Learning to generate product reviews from attributes. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 623–632.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61:65–170.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Andriy Mnih and Karol Gregor. 2014. Neural variational inference and learning in belief networks. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–1791–II–1799. JMLR.org.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Ça glar Gulçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. pages 280–290.

Tri Nguyen, Mir Rosenberg, Jianfeng Gao, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Lianhui Qin, Lemao Liu, Wei Bi, Yan Wang, Xiaojiang Liu, Zhiting Hu, Hai Zhao, and Shuming Shi. 2018. Automatic article commenting: the task and dataset. *arXiv preprint arXiv:1805.03668*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Alexander M Rush, Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Heung-Yeung Shum, Xiaodong He, and Di Li. 2018. From eliza to xiaoice: Challenges and opportunities with social chatbots. *Frontiers of Information Technology & Electronic Engineering*, 19(1):10–26.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3104–3112, Cambridge, MA, USA. MIT Press.

Chuanqi Tan, Furu Wei, Nan Yang, Bowen Du, Weifeng Lv, and Ming Zhou. 2018. S-net: From answer extraction to answer generation for machine reading comprehension. In *AAAI'18*.

Jian Tang, Yifan Yang, Sam Carton, Ming Zhang, and Qiaozhu Mei. 2016. Context-aware natural language generation with recurrent neural networks. *arXiv preprint arXiv:1611.09900*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4566–4575.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2692–2700. Curran Associates, Inc.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 189–198.

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604*.

Adams Wei Yu, Rui Dohan, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. In *International Conference on Learning Representations*.

Hai-Tao Zheng, Wei Wang, Wang Chen, and Arun Kumar Sangaiah. 2018. Automatic generation of news comments based on gated attention neural networks. *IEEE Access*, 6:702–710.

## A Yahoo! News Dataset

More information about the Yahoo dataset is shown in Table 8. The side information associated with news including:

- **Paragraph**. After pre-processing, we retain the paragraph structure of news articles.
- **Category**. There are 31 news categories and the distribution is shown in Figure 2.
- **Wiki-Entities**. The Wikipedia entities mentioned in the news articles are extracted.
- **Vote**. Each comment has upvote, downvote and abusevote information from news readers.
- **Sentiment**. Each comment is annotated with POSITIVE, NEGATIVE or NEUTRAL by Yahoo!.

| | | Train | Dev | Test |
|---|---|---|---|---|
| Tencent | # News | 191,502 | 5,000 | 1,610 |
| | Avg. # Comments per News | 27 | 27 | 27 |
| | Avg. #Upvotes per Comment | 5.9 | 4.9 | 3.4 |
| Yahoo | # News | 152,355 | 5,000 | 3,160 |
| | Avg. # Comments per News | 20.6 | 20.5 | 20.5 |
| | Avg. #Upvotes per Comment | 31.4 | 30.2 | 32.0 |
| | Avg. #DownVotes per Comment | 4.8 | 4.8 | 4.9 |
| | Avg. #AbuseVotes per Comment | 0.05 | 0.05 | 0.05 |

Table 8: Statistics of the two datasets.

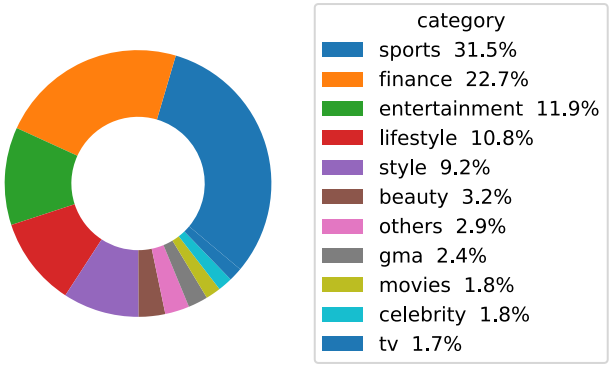


Figure 2: The category distribution of Yahoo! News dataset.

## B Case Study

We demonstrate the advantage of DeepCom over IR-TC and Att-TC with examples from the test sets of the two datasets. Table 9 and Table 10 show the comments given by the three models. The content in the red box represents salient spans predicted by the reading network and that in the blue box is generated entities included in the salient spans. We can see that compared to the two baseline methods, comments from DeepCom are more relevant to the content of the news articles. Comments from IR-TC are rich with content, but are irrelevant to the news articles, while Att-TC is prone to generate generic comments.

**Title:** 唐嫣为什么不演清宫剧？(Why Tiffany Tang never plays a role in a drama set in Qing dynasty?)

**Body:** 众多的戏剧影视作品让唐嫣博获得不少人气，众多活动的多变造型树立其在大众心中“百变小天后”的形象。商业活动高贵优雅、媒体活动清新脱俗，唐嫣每走到一处总能获得交口称赞和镁光灯的钟爱。唐嫣身材高挑、笑容甜美，精致的脸庞，时尚的装扮，有着“九头身美女”之称。如果说唐嫣最美的造型是《仙剑奇侠传三》中的紫萱，那最丑的造型应该就是这个了吧！无论是发型还是脸都无法直视啊！《仙剑奇侠传三》时候应该算是唐嫣的颜值巅峰期了吧！而穿上宫廷服装的唐嫣，真的不好看，宛如一个智障少女……这一段清宫戏，是唐嫣在电视剧《乱世佳人》的造型，但这部戏简直是唐嫣不堪回首的过去啊！这个民国时期的造型也不好看！终于明白为什么唐嫣从来不演清宫剧了！如果不用假刘海，就算脸在小，她的额头还是很大！对此，大家怎么看？
(The numerous television series and movies Tiffany Tang acted in have made her very popular, and her varied modelling in many activities has set her an image of ”the queen of reinvention” in the hearts of the public. Her elegant and unique nature showed in commercial events and media events brings her a lot of praise and spotlight. Tiffany Tang is tall, slim, has sweet smile and a beautiful face, and is always dressed in fashion. Thus, she is called “9-heads tall” girl by her fans. If her most beautiful role is Zi Xuan in “Chinese Paladin 3”, then the most ugly one should be this! Both hair-style and makeup look terrible. She is most alluring at the time of “Chinese Paladin 3”. When she dressed up with a costume in Qing Dynasty, however, she looks really bad, like an idiot. This is how Tiffany Tang looks like in a TV series “A Beauty in Troubled Times”, but this might be her forgettable moment. Her costume in Republic of China looks terrible too. I finally know why Tiffany Tang never plays a role in a drama set in Qing dynasty. If she does not use fake bangs, even her face is small, her forehead looks too big! What do you think?)

**DeepCom:** 唐嫣罗晋的演技真的很好，特别喜欢她演《仙剑奇侠传》。我觉得这部剧挺好看。(Tiffany Tang and Luo Jin are really good actors. I especially like her role in Chinese Paladin 3. I think the TV drama is worth watching.)
**Att-TC:** 我也是醉了(I have nothing to comment.)
**IR-TC:** 星爷和谁开撕过嘛, 都是别人去撕星爷！（Stephen Chow never fights with others. It is others that fight with Stephen Chow!）

Table 9: A case from Tencent News dataset.

**Title:** NBA notebook : Rockets targeting Anthony after losing Mbah a Moute

**Body:** The Houston Rockets are now determined to sign forward Carmelo Anthony after forward Luc Mbah a Moute joined the Los Angeles Clippers on a one-year, $4.3 million deal on Monday, according to an ESPN report. Anthony is currently a member of the Oklahoma City Thunder, but the two sides are reportedly working on parting ways, whether through a trade, a buyout or waiving via the stretch provision. Anthony is likely to become a free agent even if he is traded, as his new team would likely waive him. Multiple reports on Sunday said rockets guard Chris Paul wants Anthony, a good friend of his, to join the rockets, while Anthony is also believed to have interest in joining Lebron James with the Los Angeles Lakers. The Miami Heat are also reportedly interested in adding Anthony. Mbah a Moute spent the 2015-16 and 2016-17 seasons with the Clippers before joining the Rockets last season. The 31-year-old averaged 7.5 points , 3.0 rebounds and 1.2 steals in 25.6 minutes per game across 61 games (15 starts) in Houston. – The Cleveland cavaliers are looking to deal 37-year-old guard Kyle Korver and transition to a younger lineup, according to Terry Pluto of the Cleveland Plain Dealer. . . . Korver’s contract has $ 15.1 million remaining over the next two seasons, although only $ 3.4 million of his 2019-20 salary is guaranteed. He could prove to be a good option for a team looking for better perimeter shooting to compete for a league title. . . . – It ’s unclear whether Joakim Noah will remain with the New York Knicks moving forward, but the center said he hopes to say in the big apple, in a video published by TMZ. “I love New York,” Noah said, “I don’t know what’s going to happen , but Coach Fiz is cool, man.” . . .

**DeepCom:** the rockets are going to have a lot of fun in this series .
**Att-TC:** i think he is going to be a contender
**IR-TC:** the kings have a great future if cousins is n’t the team ’s leader . but if continues to be propped up as the guy , then the kings will only have a good future along with the bottom half of the west . cousins just is n’t enough of a leadership guy who will go through 3 more coaches before he finds a decent match not just for himself , but the team .

Table 10: A case from Yahoo! News dataset.

# A Topic Augmented Text Generation Model: Joint Learning of Semantics and Structural Features

**Hongyin Tang**[1,2], **Miao Li**[1,2], **Beihong Jin**[1,2,*]
[1]State Key Laboratory of Computer Science
Institute of Software, Chinese Academy of Sciences
[2]University of Chinese Academy of Sciences, Beijing China
{tanghongyin14,limiao17}@otcaix.iscas.ac.cn
Beihong@iscas.ac.cn

## Abstract

Text generation is among the most fundamental tasks in natural language processing. In this paper, we propose a text generation model that learns semantics and structural features simultaneously. This model captures structural features by a sequential variational autoencoder component and leverages a topic modeling component based on a Gaussian distribution to enhance the recognition of text semantics. To make the reconstructed text more coherent to the topics, the model further adapts the encoder of the topic modeling component for a discriminator. The results of experiments over several datasets demonstrate that our model outperforms several states of the art models in terms of text perplexity and topic coherence. Moreover, the latent representations learned by our model is superior to others in a text classification task. Finally, given the input texts, our model can generate meaningful texts which hold similar structures but under different topics.

## 1 Introduction

Text generation is a fundamental task in natural language processing (NLP). Existing methods for text generation are mostly limited in supervised setting and designed for specific applications (e.g., machine translation (Bahdanau et al., 2015), text summarization (Rush et al., 2015)). Several research work (Yu et al., 2017; Zhang et al., 2016) attempts generic text generation based on deep generative models (e.g., GAN, VAE). However, models based on GAN(Generative Adversarial Network) are not able to generate explicit latent codes with salient features of texts. Different from GAN-based models, the VAE (Variational Autoencoder) and its variants can get latent codes of texts with reconstructing texts by its decoder, where it is assumed that the generation process is controlled by codes in a continuous latent space. A natural way to implement the VAEs is to adopt autoregressive networks (e.g., RNNs) as the decoder and encoder. This kind of implementation of VAEs considers sequential information of texts, and is able to model the linguistic structure of texts. But it is not good at modeling semantics (Dieng et al., 2017) and may cause the latent variable collapse problem (Bowman et al., 2016) which leads that the decoder ignores information from the inferred latent codes.

In general, texts inherently contain semantic features and structural features. Only when the latent codes of texts contain structural and semantic information can high-quality texts be generated from the codes. As far as the standard VAE is concerned, it assumes that the latent code is Gaussian distributed so that we cannot distinguish which part of code controls the structure and which part controls the semantics. In other words, it is difficult to generate effective texts by controlling text features directly.

To generate high-quality texts (i.e., with semantic and structural information), we propose a model named TATGM which adopts a sequential VAE to learn structural features of texts and builds a topic modeling component which extracts semantic features of texts. The main contributions of this paper are summarized as follows:

- TATGM can capture the semantics of texts by introducing the topic modeling component. The topic modeling component generates words of texts based on a Gaussian distribution which enables us to take full advantage of information shared by the word embeddings. Moreover, the encoder of the topic modeling component is served as a discriminator to force the decoder of the sequence modeling component to generate texts having the semantics as close to the original texts as



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5090–5099,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

possible. Specifically, no extra training supervised by labeled data is needed for this discriminator.

- The latent code learned by TATGM is a concatenation of the latent variables of the topic modeling component and the sequence modeling component. By two separate parts of the code, TATGM can control structural and semantic information independently. Thus, we can get texts with the changed semantics or structure by changing one part of the code. One interesting practice is that we can get question-answering pairs with the same structure in different semantic spaces.

- The modeling ability of TATGM is evaluated by extensive experiments in terms of perplexity and the coherence of topics. Furthermore, to verify whether TATGM can learn salient features, a classification task using the latent codes is conducted. Experimental results show that TATGM achieves the best performance while compared with many existing models. Finally, several texts generated by TATGM are demonstrated, which indicates that TATGM can generate different expressions of texts of the same structure in different topics.

The rest of the paper is organized as follows. Section 2 introduces the related work. Section 3 describes the model TATGM in detail. Section 4 gives an experimental evaluation. Finally, the paper is concluded in Section 5.

## 2 Related Work

VAEs (Kingma and Welling, 2014) are a type of deep generative models, which learns explicit latent codes of input data in a continuous latent space and generates data with decoders. VAEs can extract the features of input texts by posterior inference with neural networks and reparameterization tricks. VAE-based models have been widely applied in image generation (Gregor et al., 2015), machine translation (Zhang et al., 2016), and knowledge graph reasoning (Zhang et al., 2018). VAE-based models can also be applied in different tasks of text processing, where the input document is treated as its bag of words (Miao et al., 2016; Srivastava and Sutton, 2017; Miao et al., 2017) or a sequential text (Bowman et al., 2016). The former type of models is always related to topics. For example, Miao et al. (2016) builds the connection by letting weights of the softmax decoder to indicate topic interpretability. Although these models overlook sequential information in texts, they can learn effective latent codes with global semantics. On the other hands, VAE-based models with the input of sequences usually adopt RNNs as encoders or decoders. However, latent variable collapse (i.e., the latent variable collapses to obey the distribution of the prior) makes the sequential VAE to generate texts without information of latent code so that the latent code cannot have enough information of text features.

To avoid the latent variable collapse problem, two types of methods are adopted. The first type tackles this problem by weakening the context modeling ability of decoders mostly in the model architecture level. For example, Bowman et al. (2016) adopts word dropout when feeding the decoder and Yang et al. (2017); Semeniuta et al. (2017) employ non-autoregressive networks (e.g., convolutional neural networks) in the decoder. The second type is to modify the original objective. For example, Bowman et al. (2016) adopts *KL* annealing which can be seen as gradually adding annealing weight into the objective function. Zhao et al. (2018) introduces an additional mutual information term to compensate for the objective function. Xiao et al. (2018) and Zhao et al. (2017) introduce the bag-of-words loss as an auxiliary loss, which measures how well predictions of words can be made from the latent codes. The ideas behind all these methods are same, i.e., to force the models to generate texts from the latent codes so as to make latent codes have more information on text features.

Improving the sequential model by incorporating topics have been explored (Lau et al., 2017; Wang et al., 2018; Dieng et al., 2017; Mikolov and Zweig, 2012). But these models do not have a code for the text structure. Taking TGVAE (Wang et al., 2018) as an example, it guides the generation of the VAE latent code by topic distribution. However, it does not separate the semantic and structure latent codes explicitly. In addition, the models mentioned above use a multinominal LDA which cuts off the possibility of leveraging the semantics in embeddings. As a remedy, Das et al. (2015) and Hu et al. (2012)

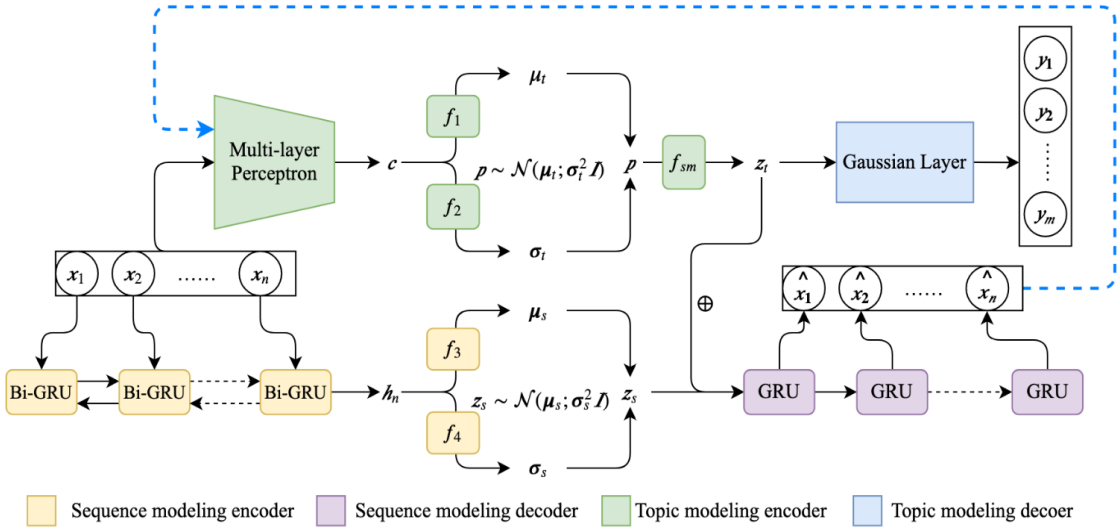


Figure 1: The holistic structure of TATGM. The dotted arrow denotes that the texts which are generated by the sequence modeling decoder are fed into the topic modeling encoder as the input of the discriminator.

propose to adopt a Gaussian-based topic model which assumes each word is generated by a Gaussian distribution. However, their learning algorithms are based on sampling and variational inference, which cannot be assembled in an end-to-end mode.

Hu et al. (2017) proposes a text generation model based on VAE, which aims at generating sentences with controllable styles by learning disentangled latent representations. It feeds generated texts into a discriminator to preserve that these texts have the given style and optimizes the generator with the signals backpropagated from the discriminator. Tian et al. (2018) uses a classifier which is trained with small datasets as the discriminator. Yang et al. (2018) adopts a language model as the discriminator to control the style transfer of generated texts.

Different from existing work, our model combines a sequence modeling component and a topic modeling component so that the final latent code of our model contains both topic semantics and sequential structure of texts. In particular, our model can learn these two features simultaneously, since AutoEncoding Variation Bayes (AEVB) are employed to train these two components. Moreover, our models generate bags-of-words text representations from the latent codes, which avoids the latent variable collapse problem.

## 3 Model

TATGM is essentially a hybrid autoencoder which comprises a topic modeling component capturing topic information and a sequence modeling capturing structural features. The topic modeling component captures topic information through a Gaussian-based topic model while the sequence modeling component integrates the topic information and generates text. As shown in Fig. 1, the input texts are fed into the two components simultaneously. Each component contains an encoder and decoder, which are labeled by its own color.

### 3.1 Neural Topic Modeling Component

Similar to existing topic modeling methods, we treat bag-of-words representations of texts as input. However, since the encoder in the topic modeling component is expected to be as a discriminator and guarantee that texts generated by the sequence modeling decoder have specific topic information, the encoder cannot be with the discrete representations (e.g., one-hot representation) of texts as input. It is because such discrete representations of texts cannot be compatible with backward gradients from the discriminator. Therefore, we employ the embeddings of words as input, considering that the word embedding space is continuous and the similarity between every two words can be calculated by the Euclidean distance. Further, we assume that there are $K$ topics in the corpus and each topic is represented as a multivariable Gaussian distribution with a mean and a variance (e.g., $\mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\sigma}_k^2 \boldsymbol{I})$ for a given topic $k$).

In our neural topic modeling component, a document with $n$ words is represented as $\boldsymbol{x} = \{\boldsymbol{x}_i\}_{i=1}^{n}$ where $\boldsymbol{x}_i \in \mathbb{R}^e$ denotes the embedding

of $i$-th word of the document. The generative process of the document is as follows.

1. For each document, draw the document-topic distribution $\boldsymbol{z}_t \sim Dir(\alpha)$.
2. For $i$-th word in the document.
   (a) Draw topic assignment $t_i \sim Cat(\boldsymbol{z}_t)$
   (b) Draw the word $\boldsymbol{x}_i \sim \mathcal{N}(\boldsymbol{\mu}_{t_i}, \boldsymbol{\sigma}_{t_i}^2 \boldsymbol{I})$

According to the generative process above and the marginalization of $t_i$, the likelihood $p(\boldsymbol{x})$ of the document $\boldsymbol{x}$ can be derived as

$$\begin{aligned} &\int_{\boldsymbol{z}_t} p(\boldsymbol{z}_t|\alpha) \left( \prod_{i=1}^{n} \sum_{t_i} p(\boldsymbol{x}_i|\boldsymbol{\mu}_{t_i}, \boldsymbol{\sigma}_{t_i}^2 \boldsymbol{I}) p(t_i|\boldsymbol{z}_t) \right) d\boldsymbol{z}_t \\ &= \int_{\boldsymbol{z}_t} p(\boldsymbol{z}_t|\alpha) \left( \prod_{i=1}^{n} p(\boldsymbol{x}_i|\boldsymbol{\mu}, \boldsymbol{\sigma}^2 \boldsymbol{I}, \boldsymbol{z}_t) \right) d\boldsymbol{z}_t \end{aligned} \tag{1}$$

where $\boldsymbol{\mu} = \{\boldsymbol{\mu}\}_{k=1}^{K}$, $\boldsymbol{\sigma} = \{\boldsymbol{\sigma}\}_{k=1}^{K}$. Further, we adopt AEVB and the reparameterization trick to achieve posterior inference and parameter learning. Specifically, similar to (Srivastava and Sutton, 2017), we first draw a Gaussian random vector by a reparameterization trick and then pass it through a softmax function to parameterize the multinomial document topic distributions. Therefore, we can replace $\alpha$ with the parameters of Gaussian prior $\boldsymbol{\mu}_0, \boldsymbol{\sigma}_0^2$.

Thus, we can get the ELBO of our topic modeling component.

$$\begin{aligned} \mathcal{L}_T =& \mathbb{E}_{q(\boldsymbol{z}_t|\boldsymbol{x})}[\log p(\boldsymbol{x}|\boldsymbol{z}_t)] \\ &- KL(q(\boldsymbol{z}_t|\boldsymbol{x})||p(\boldsymbol{z}_t|\boldsymbol{\mu}_0, \boldsymbol{\sigma}_0^2 \boldsymbol{I})) \end{aligned} \tag{2}$$

In Eq. 2, $\boldsymbol{z}_t$ is inferred by the neural network in the encoder. The inference process is detailed as follows. Document $\boldsymbol{x}$ is fed into a two-layer perceptron with *ReLU* as the activation function. The transformation results of the MLP are then processed by max-pooling, getting fixed-length representation $\boldsymbol{c}$ of the document. Parameters of the posterior $\boldsymbol{\mu}_t$ and $\boldsymbol{\sigma}_t^2$ are obtained from two feedforward neural networks. Next, a Gaussian variable is sampled by the reparameterization trick. Finally, $\boldsymbol{z}_t$ is obtained by passing $\boldsymbol{p}$ to the softmax function. Our choices are specified as follows.

$$\begin{aligned} &\boldsymbol{c} = Pooling(MLP(\boldsymbol{x})) \\ &\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t = f_1(\boldsymbol{c}), f_2(\boldsymbol{c}) \\ &\boldsymbol{p} \sim \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t^2 \boldsymbol{I}) \in \mathbb{R}^K \\ &\boldsymbol{z}_t = f_{sm}(\boldsymbol{W}_t \boldsymbol{p}) \end{aligned}$$

where $f_1, f_2$ denote two feedforward neural networks, $\boldsymbol{W}_t$ is a trainable parameter, $f_{sm}$ denotes softmax function.

Besides, $KL(q(\boldsymbol{z}_t|\boldsymbol{x})||p(\boldsymbol{z}_t|\boldsymbol{\mu}_0, \boldsymbol{\sigma}_0^2 \boldsymbol{I}))$ in Eq. 2 is obtained by calculating the *KL* divergence between $\mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t^2 \boldsymbol{I})$ and $\mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\sigma}_0^2 \boldsymbol{I})$.

In general, during the training of topic models, a smaller vocabulary is built after eliminating some specific words (e.g., stop words, frequent words, and rare words) in the corpus. This is a denoising preprocessing step that makes the model more reliable. However, to make the execution of the discriminator applicable, our model takes all of the words in the document as input while outputting the words in a smaller vocabulary as a topic model does. Besides the reason above, this setting can also be regarded as a regulation of the model which strengthen the ability of modeling documents.

Supposing we have a smaller vocabulary, the document represented by this vocabulary is $\boldsymbol{y} = \{\boldsymbol{y}_i\}_{i=1}^{m}$, where $m < n$ is the number of words and $\boldsymbol{y}_i \in \mathbb{R}^e$ is the word embedding of the word at position $i$.

The topic modeling decoder reconstructs the document from $\boldsymbol{z}_t$. We assume that the distribution of every topic is a Gaussian with identity variance matrix. Therefore, $p(\boldsymbol{x}|\boldsymbol{z}_t)$ of Eq. 2 is indeed $p(\boldsymbol{y}|\boldsymbol{z}_t)$ and can be expanded as:

$$\begin{aligned} p(\boldsymbol{y}|\boldsymbol{z}_t) &= \prod_{i=1}^{m} p(\boldsymbol{y}_i|\boldsymbol{\mu}_{t_i}, \boldsymbol{\sigma}_{t_i}^2 \boldsymbol{I}, \boldsymbol{z}_t) \\ &\propto \prod_{i=1}^{m} \boldsymbol{z}_t^T \exp{(\boldsymbol{y}_i - \boldsymbol{\mu})^T (\boldsymbol{y}_i - \boldsymbol{\mu})} \end{aligned} \tag{3}$$

To stabilize the training process, when calculate the likelihood, we first normalize each $\boldsymbol{y}_i$ and $\boldsymbol{\mu}_k$.

### 3.2 Sequence Modeling Component

Since the structural features are closely related to the word sequence information, we construct a sequential VAE. By the sequential VAE, the encoder infers the structural latent variable and the decoder reconstructs texts via integrating the topic latent variable and the structural latent variable. This process is depicted by the bottom half of Fig. 1. Since the semantic information is given by the topic modeling component, the reconstruction loss function makes the encoder of sequential VAE focusing on encoding the structural features. Let $\boldsymbol{z}_s$ denote the structural latent variable, the ELBO of this component is in the following form:

$$\begin{aligned}\mathcal{L}_S =& \mathbb{E}_{q(\boldsymbol{z}_t|\boldsymbol{x})q(\boldsymbol{z}_s|\boldsymbol{x})}\left[\log p(\boldsymbol{x}|\boldsymbol{z}_t, \boldsymbol{z}_s)\right] \\ & - KL(q(\boldsymbol{z}_s|\boldsymbol{x})||p(\boldsymbol{z}_s))\end{aligned} \tag{4}$$

We adopt a bi-directional GRU as the encoder. The last hidden state of the encoder is used to infer the parameters $\boldsymbol{\mu}_s, \boldsymbol{\sigma}_s$ of the Gaussian distribution. Then, we get $\boldsymbol{z}_s$ by the reparameterization trick.

$$\begin{aligned}\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_n &= BiGRU(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n) \\ \boldsymbol{\mu}_s, \boldsymbol{\sigma}_s &= f_3(\boldsymbol{h}_n), f_4(\boldsymbol{h}_n) \\ \boldsymbol{z}_s &\sim \mathcal{N}(\boldsymbol{\mu}_s, \boldsymbol{\sigma}_s^2 \boldsymbol{I})\end{aligned}$$

where $f_3$ and $f_4$ are the feedforward neural networks. Further, we obtain the holistic latent code $\boldsymbol{z}$ by concatenating $\boldsymbol{z}_t$ and $\boldsymbol{z}_s$, i.e., $\boldsymbol{z} = [\boldsymbol{z}_t; \boldsymbol{z}_s]$.

We adopt a GRU as the decoder to reconstruct document $\boldsymbol{x} = \{\boldsymbol{x}_i\}_{i=1}^n$ and latent code $\boldsymbol{z}$ as its initial state. So, the likelihood of the reconstructed document can be derived by Eq. 5.

$$\begin{aligned}p(\boldsymbol{x}|\boldsymbol{z}_t, \boldsymbol{z}_s) &= p(\boldsymbol{x}_1|\boldsymbol{z}) \prod_{i=1}^{n} p(\boldsymbol{x}_i|\boldsymbol{x}_{1:i-1}) \\ &= p(\boldsymbol{x}_1|\boldsymbol{z}) \prod_{i=1}^{n} p(\boldsymbol{x}_i|h_i)\end{aligned} \tag{5}$$

where $\boldsymbol{h}_i$ denotes the $i$-th hidden state of the GRU decoder.

### 3.3 Topic Encoder as a Discriminator

Although the holistic latent code contains semantics and structure features, the decoder may not fully leverage the semantic part of code. Besides the reconstruction loss which drives the generator to produce realistic sentences, we introduce a discriminator which enforces the generator to produce texts in a coherent topic with $\boldsymbol{z}_t$. Specifically, we let the encoder in our topic modeling component act as the discriminator. It is expected that topic distributions inferred from the generated texts are similar to the topic distributions of the original texts. However, if the outputs of the sequence decoder are discrete, it is impossible to propagate gradients from the discriminator through the discrete samples. We thus resort to a *Gumbel-Softmax* distribution as an approximation of the discrete samples.

In detail, in each step of the generative process of texts, we get the distribution of one word $p(\boldsymbol{x}_i|\boldsymbol{z}_s, \boldsymbol{z}_t) = [\pi_1, \pi_2, ..., \pi_{|V|}]$, and approximate the samples from $p(\boldsymbol{x}_i|\boldsymbol{z}_s, \boldsymbol{z}_t)$ by

$$u_i = \frac{\exp\left(\log(\pi_i) + g_i\right)/\tau}{\sum_{j=1}^{|V|} \exp\left(\log(\pi_j) + g_j\right)/\tau} \tag{6}$$

where $g_i$ and $g_j$ are samples from $Gumbel - Softmax(0, 1)$. As training proceeds, $\tau$ gets close to 0, yielding the increasingly peaked distribution that finally emulate the discrete cases. Thus, the $i$-th word generated by the decoder is

$$\widehat{\boldsymbol{x}}_i = \boldsymbol{u}^T \boldsymbol{W}_v \tag{7}$$

where $\boldsymbol{W}_v \in \mathbb{R}^{|V| \times e}$ denotes word embedding.

We feed the above generative texts into the topic modeling encoder and expect to get the maximum likelihood of the original topic distribution $\boldsymbol{z}_t$. So the loss function of the discriminator is specified as follows.

$$\mathcal{L}_D = \mathbb{E}_{p(\boldsymbol{z}_s)p(\boldsymbol{z}_t)} \log q(\boldsymbol{z}_t|\widehat{\boldsymbol{x}}) \tag{8}$$

Combining three parts of loss functions, we can get the loss function of the model as Eq. 9, where $\lambda_D$=0.1.

$$\begin{aligned}\mathcal{L} &= \mathcal{L}_T + \mathcal{L}_S + \lambda_D \mathcal{L}_D \\ &= \mathbb{E}_{q(\boldsymbol{z}_t|\boldsymbol{x})}\left[\log p(\boldsymbol{x}|\boldsymbol{z}_t) + \mathbb{E}_{q(\boldsymbol{z}_s|\boldsymbol{x})}(\log p(\boldsymbol{x}|\boldsymbol{z_s}, \boldsymbol{z_t}))\right] \\ &- \lambda_t KL(q(\boldsymbol{z}_t|\boldsymbol{x})||p(\boldsymbol{z}_t)) - \lambda_s KL(q(\boldsymbol{z}_s|\boldsymbol{x})||p(\boldsymbol{z}_s)) \\ &+ \lambda_D \mathbb{E}_{p(\boldsymbol{z}_s)p(\boldsymbol{z}_t)} \log q(\boldsymbol{z}_t|\widehat{\boldsymbol{x}})\end{aligned} \tag{9}$$

We learn parameters of topic modeling and sequence modeling components alternatively in the multitask learning setting. To avoid the latent variable collapse problem, we add weights $\lambda_t, \lambda_s$ to *KL* related terms in the loss function and make the weights increase slowly to 1 in training.

## 4 Experiments and Result Analyses

To evaluate the text generation ability, we do experiments from four perspectives. First, we evaluate the language modeling ability of the sequence modeling component and the topic coherence of the topic modeling component. Next, to evaluate the effectiveness of the learned latent codes of texts, we perform a semi-supervised classification task. Finally, we transfer the question-answering pairs on Yahoo dataset to different topics and demonstrate generated texts.

### 4.1 Dataset Description & Experiment Setup

We conduct experiments on five benchmark text datasets: APNEWS[1], IMDB (Maas et al., 2011),

[1]https://www.ap.org/en-gb/

| Data | #SM Voc | #TM Voc | #Training Docs | #Development Docs | #Test Docs | #Avg Len |
|---|---|---|---|---|---|---|
| APNEWS | 33k | 6k | 700k | 27.4k | 26.3k | 21.4 |
| IMDB | 34k | 8k | 900k | 200k | 200k | 22.5 |
| BNC | 40k | 9k | 800k | 44k | 52k | 22.6 |
| Yahoo | 20k | 8k | 100k | 10k | 10k | 75.3 |
| Yelp15 | 20k | 7k | 100k | 10k | 10k | 97.8 |

Table 1: Summary statistics for datasets. SM Voc and TM Voc stand for vocabularies of corpus in the sequence modeling component and the topic modeling component, respectively.

BNC (BNC Consortium, 2007), Yahoo Answer (Yahoo) and Yelp15. We randomly sample 100k as training data and 10k as validation and testing data, respectively. For all datasets, we first tokenize the texts using Stanford CoreNLP (Manning et al., 2014). Then we lowercase all word tokens, and filter words that occur less than 10 times. For Yahoo and Yelp15, we truncate the vocabulary to 20k words for fast training. For the bag-of-words input in the topic modeling component, we further remove stop words, and exclude the top 0.1% most frequent words and also words that appear in less than 100 documents. Table 1 shows summary statistics of all datasets.

We fix the max sequence length to 50 for the texts in APNEWS, IMDB, BNC and 150 for Yahoo and Yelp15. The 300-dimentional embeddings of words are shared by two components in our model. For the topic modeling component, we adopt a 2-layer MLP with 200 hidden units and *ReLU* as its activation function. We set the size of $\boldsymbol{z}_t$ to 50. For the sequence modeling component, we adopt a bidirectional single layer GRU with 600 hidden units (300 in each direction) as the encoder and a unidirectional GRU with 300 hidden units as the decoder. The size of $\boldsymbol{z}_s$ is set to 20. We use a batch size of 32 and train the model up to 40 epochs. Linear scheduling is used in the *KL* annealing and the weight grows from 0 at the beginning to 1 at 40k steps.

### 4.2 Sequence Modeling Evaluation

By the experiments on five datasets, we evaluate the language modeling ability in terms of perplexity (*PPL*). We compare several baselines including models based on language model (i.e., LSTM LM, LSTM+LDA, Topic-RNN, TDLM) and models based on VAE (i.e., LSTM VAE, VAE+HF, TGVAE, DCNNVAE, DVAE). LSTM LM is a plain language model implemented in LSTM. LSTM+LDA concatenates the hidden states with the topic distribution learned by a pre-trained LDA. Different from ours, LDA in LSTM+LDA is trained separately. Topic-RNN (Dieng et al., 2017) learns an LDA with a language model jointly and incorporates the topic distribution by a gate mechanism. TDLM (Lau et al., 2017) incorporates a convolutional topic model and also leverages the topic distribution in the same way that LSTM+LDA does. LSTM VAE is a standard VAE whose encoder and decoder are implemented by two LSTMs, respectively. VAE+HF (Wang et al., 2019) is a VAE with a mixture-of-Gaussians prior with Householder Flow. TGVAE (Wang et al., 2019) is a VAE guided by a Gaussian mixture distribution as prior with a jointly-trained LDA. DCNN-VAE (Yang et al., 2017) is a VAE using dilated CNN as its decoder. DVAE (Xiao et al., 2018) uses a Dirichlet latent variable to improve VAE. Besides the model we propose, we also evaluate our model without the discriminator (i.e., Ours w/o Dis).

The perplexity of VAE-based models is estimated in ELBO approximately which is comprised of a reconstruction term and a *KL* term. Besides the perplexity, we report the *KL* term in the VAE based models. For our model, we report the *KL* values of the sequence modeling component and the topic modeling component in the first and second rows, respectively.

For a fair comparison, the compared results are picked from the models with 50 topics. The results are shown in Table 2. We find that our model and our model without a discriminator occupy the top-2 positions. We attribute the improvements to the decoupling of semantic and structural features. From the results, we also verify that the discriminator in our model can help in decreasing the perplexity. In addition, the *KL* values in the topic modeling component is much larger than the sequential one. One possible reason is that the topic information reveals much of the diversity of texts.

### 4.3 Topic Coherence Evaluation

Topic models are traditionally evaluated using perplexity. However, (Chang et al., 2009) show that

| | APNEWS | | IMDB | | BNC | | Yahoo | | Yelp15 | |
|---|---|---|---|---|---|---|---|---|---|---|
| | *PPL* | *KL* | *PPL* | *KL* | *PPL* | *KL* | *PPL* | *KL* | *PPL* | *KL* |
| LSTM LM | 64.13 | - | 72.14 | - | 102.89 | - | 66.2 | - | 42.6 | - |
| LSTM+LDA | 57.05 | - | 69.58 | - | 96.42 | - | 53.5 | - | 37.2 | - |
| Topic-RNN | 56.77 | - | 68.74 | - | 94.66 | - | - | - | - | - |
| TDLM | 53.00 | - | 63.67 | - | 87.42 | - | - | - | - | - |
| LSTM VAE | 75.89 | 1.78 | 86.16 | 2.78 | 105.10 | 0.13 | 65.6 | 0.4 | 45.5 | 0.5 |
| VAE+HF | 71.60 | 0.83 | 83.67 | 1.51 | 104.82 | 0.17 | - | - | - | - |
| TGVAE | 48.73 | 3.55 | 57.11 | 5.02 | 87.86 | 4.57 | - | - | - | - |
| DCNN-VAE | - | - | - | - | - | - | 63.9 | 10.0 | 41.1 | 7.6 |
| DVAE | - | - | - | - | - | - | 47.6 | 31.9 | 34.7 | 30.5 |
| Ours w/o DIS | 47.80 | 2.13<br>7.32 | 54.50 | 3.24<br>9.82 | 83.92 | 1.32<br>7.83 | 42.92 | 3.12<br>11.45 | **31.87** | 2.43<br>8.72 |
| Ours | **47.23** | 2.88<br>8.18 | **52.01** | 3.87<br>9.34 | **80.78** | 2.54<br>7.76 | **40.80** | 4.25<br>11.51 | 32.90 | 2.92<br>8.34 |

Table 2: Language modeling results in the terms of *PPL* and *KL*.

| Model | APNEWS | IMDB | BNC | Yahoo | Yelp15 |
|---|---|---|---|---|---|
| LDA | 0.125 | 0.084 | 0.106 | 0.148 | 0.087 |
| TDLM | 0.149 | 0.104 | 0.102 | - | - |
| Topic-RNN | 0.134 | 0.103 | 0.102 | - | - |
| TGVAE | 0.157 | 0.105 | 0.113 | - | - |
| Ours w/o DIS | 0.170 | **0.121** | **0.115** | 0.182 | **0.114** |
| Ours | **0.171** | 0.120 | 0.114 | **0.182** | 0.113 |

Table 3: Topic coherence over the datasets in the term of *NPMI*.

| Model | 20NEWS | Yahoo | Yelp15 |
|---|---|---|---|
| VAE | 52.29 | 53.2* | 27.2* |
| LDA | 55.38 | 56.72 | 30.44 |
| DVAE | - | 57.6* | 42.4* |
| TDLM | 60.6* | - | - |
| Ours w/o DIS (topic) | 59.31 | 60.51 | 35.11 |
| Ours w/o DIS(seq) | 28.25 | 31.42 | 34.39 |
| Ours w/o DIS | 61.28 | 63.98 | 45.34 |
| Ours (topic) | 59.06 | 61.34 | 34.36 |
| Ours (seq) | 30.05 | 30.31 | 35.37 |
| Ours | **61.47** | **64.75** | **46.03** |

* are from the original papers.

Table 4: Test split accuracy of classifiers trained with learned representations.

perplexity does not correlate with the coherence of the generated topic. We adopt normalized PMI (NPMI) to evaluate the topic coherence following (Lau et al., 2017). Given the top-$n$ words of a topic, coherence is computed based on the sum of pairwise NPMI scores between topic words. We average topic coherence over the top 5/10/15/20 topic words. To aggregate topic coherence scores, we calculate the mean coherence over topics. In the experiments, the number of topics remains 50 among all baselines. From Table 3, we find that the discriminator gives little improvement. It is because that the role of the discriminator is to take the topic distributions as the supervised signals to improve the generation so that the sequence decoder can generate more topic relevant texts. That is, the discriminator does not improve the topic modeling component itself.

Besides topic coherence values, to understand the topics concretely, we also provide top five topic words from eight randomly chosen topics on each dataset in the supplementary material.

### 4.4 Semi-supervised Classification

To evaluate whether latent codes incorporate text features, we perform a semi-supervised classification task and compare our model with the other models. To make a comprehensive comparison, we use Yahoo, Yelp15 as well as 20NEWS for text classification. Here, 20NEWS is a collection of forum-like messages from 20 news-groups categories.

For any model to be evaluated, we first train it by the training documents of the dataset, and then executing the well-trained model to obtain latent codes of all documents in the dataset. Next, we sample 2,000 documents from the training data and train a 2-layer softmax classifier using these documents and their category labels. The accuracy on the testing set of documents is shown in Table 4. Since in our TATGM the latent code comprises two latent variables, we explore the accuracy on the two latent variables solely and collectively.

As shown in Table 4, our model which combines the two latent variables achieves the highest accuracy. For 20NEWS and Yahoo, the model only using the topic latent variable is better than that only using the structural latent variable. Since these two datasets are labeled by the topics, the combination improves little. For Yelp15 dataset, the performances of our two latent variables are similar whereas the combination improves up to

| Topic | Question | Answer |
|---|---|---|
| *Society&Culture* | what is the average age of the homeless population? | i think they are younger then they look. i 'd guess 30 years old. |
| *Science&Mathematics* | what is the meaning of life in the world? | i know that i can find out about it. |
| *Health* | what is the average weight to lose of a week? | i think it varies to lose weight. |
| *Education&Reference* | what is the formula for energy? | i have about 10 points for asking questions. |
| *Sports* | which team has the most winning world cup? | i think brazil will win the world cup finals. i think they will win. |
| *computers* | what is the problem with the yahoo messenger? | i think yahoo messenger is not working on yahoo messenger. |
| *Business&Finance* | what is the minimum wage for a tax income? | i do n't know if they are illegal to pay taxes. |
| *Entertainment&Music* | what is the way of listen songs? | i think listen a song on internet. |
| *Family&Relationships* | what is the average age of have sex? | i am 14 years old and i 'm not sure what age is it? |
| *Politics&Government* | what states is the president of the united states? | i do n't know if it is illegal to be illegal. |

Table 5: A topic transfer example of one question-answering pair on Yahoo dataset.

15%. The reason may be the fact that the Yelp15 dataset is labeled by the sentiment which is determined by not only single terms but also n-grams. For example, *good* and *not good* represents opposite sentiments. Our sequence modeling component can capture such kind of features more accurately.

### 4.5 Topic Transfer between Question-answering Pairs

In our model, each dimension of the latent variables in the topic modeling component corresponds to a topic, therefore we can manipulate the variables manually to verify whether the generation can express a given topic while remaining the same structure. Specifically, given a document and its latent code, we change the topic part of the code and keep the structural part of the code unchanged. Then we can check the text the decoder generated from the whole latent code.

We conduct the experiments on the Yahoo dataset where each item contains one question-answering pair and one label corresponds to its topic. At first, we treat each question as a single sentence to check whether the generated texts satisfy our assumption. From the second column of Table 5, we find that the generation can express a different topic while maintaining the original question structure.

We further try to transfer the question-answering pair from its original topic to target topics. We find that the model not only transfers topic of two sentences but also produces reasonable question-answering pairs, i.e., the new answer is meaningful to the new question.This is helpful for the automatic question-answering scenarios. Table 5 shows an example of topic transfer. The first row lists the original question-answering pair which are in the topic of *Society&Culture*. The rest rows show the generated question-answering pairs when we change the original topic to a target topic while keeping the structure latent variable unmodified. Specifically, we modify the original topic distribution to a new distribution whose dimension of the target topic is 1 and others are 0s. From the results, we can observe that while we change the topic to another, the generated questions remain almost the same structure as the original ones. Moreover, the generated answers are also transferred to the target topic and they together with the generated questions compose reasonable question-answering pairs. More examples are shown in the supplementary material.

## 5 Conclusion

In the paper, we present the text generation model TATGM. The model can learn semantics and structural features simultaneously. Moreover, the model employs a discriminator to ensure that the generated text is more coherent with the given topic information. Experimental results show that our model has a better text modeling ability than several state-of-the-art methods and learns disentangled latent representations for texts which shows the superiority in a classification task. Specifically, our model can generate meaningful question-answering pairs, which provides an alternative transfer learning way and helps to broaden

the knowledge in other fields.

## Acknowledgments

This work has been supported by National Key R&D Program of China (No. 2017YFC0803300).

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

BNC BNC Consortium. 2007. The british national corpus, version 3 (bnc xml edition). In *Distributed by Bodleian Libraries, University of Oxford, on behalf of the BNC Consortium*.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, pages 10–21.

Jonathan Chang, Jordan L. Boyd-Graber, Sean Gerrish, Chong Wang, and David M. Blei. 2009. Reading tea leaves: How humans interpret topic models. In *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009. Proceedings of a meeting held 7-10 December 2009, Vancouver, British Columbia, Canada.*, pages 288–296.

Rajarshi Das, Manzil Zaheer, and Chris Dyer. 2015. Gaussian LDA for topic models with word embeddings. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 795–804.

Adji B. Dieng, Chong Wang, Jianfeng Gao, and John W. Paisley. 2017. Topicrnn: A recurrent neural network with long-range semantic dependency. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. 2015. DRAW: A recurrent neural network for image generation. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 1462–1471.

Pengfei Hu, Wenju Liu, Wei Jiang, and Zhanlei Yang. 2012. Latent topic model based on gaussian-lda for audio retrieval. In *Chinese Conference on Pattern Recognition*, pages 556–563. Springer.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 1587–1596.

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Jey Han Lau, Timothy Baldwin, and Trevor Cohn. 2017. Topically driven neural language model. *arXiv preprint arXiv:1704.08012*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*, pages 142–150.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland. Association for Computational Linguistics.

Yishu Miao, Edward Grefenstette, and Phil Blunsom. 2017. Discovering discrete latent topics with neural variational inference. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 2410–2419.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *International conference on machine learning*, pages 1727–1736.

Tomas Mikolov and Geoffrey Zweig. 2012. Context dependent recurrent neural network language model. In *2012 IEEE Spoken Language Technology Workshop (SLT), Miami, FL, USA, December 2-5, 2012*, pages 234–239.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 379–389.

Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. 2017. A hybrid convolutional variational autoencoder for text generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 627–637.

Akash Srivastava and Charles A. Sutton. 2017. Autoencoding variational inference for topic models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Youzhi Tian, Zhiting Hu, and Zhou Yu. 2018. Structured content preservation for unsupervised text style transfer. *CoRR*, abs/1810.06526.

Wenlin Wang, Zhe Gan, Wenqi Wang, Dinghan Shen, Jiaji Huang, Wei Ping, Sanjeev Satheesh, and Lawrence Carin. 2018. Topic compositional neural language model. In *International Conference on Artificial Intelligence and Statistics, AISTATS 2018, 9-11 April 2018, Playa Blanca, Lanzarote, Canary Islands, Spain*, pages 356–365.

Wenlin Wang, Zhe Gan, Hongteng Xu, Ruiyi Zhang, Guoyin Wang, Dinghan Shen, Changyou Chen, and Lawrence Carin. 2019. Topic-guided variational autoencoders for text generation. *CoRR*, abs/1903.07137.

Yijun Xiao, Tiancheng Zhao, and William Yang Wang. 2018. Dirichlet variational autoencoder for text modeling. *CoRR*, abs/1811.00135.

Zichao Yang, Zhiting Hu, Chris Dyer, Eric P. Xing, and Taylor Berg-Kirkpatrick. 2018. Unsupervised text style transfer using language models as discriminators. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pages 7298–7309.

Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. 2017. Improved variational autoencoders for text modeling using dilated convolutions. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 3881–3890.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 2852–2858.

Biao Zhang, Deyi Xiong, Jinsong Su, Hong Duan, and Min Zhang. 2016. Variational neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 521–530.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J. Smola, and Le Song. 2018. Variational reasoning for question answering with knowledge graph. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 6069–6076.

Tiancheng Zhao, Kyusong Lee, and Maxine Eskénazi. 2018. Unsupervised discrete sentence representation learning for interpretable neural dialog generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 1098–1107.

Tiancheng Zhao, Ran Zhao, and Maxine Eskénazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 654–664.

# LXMERT: Learning Cross-Modality Encoder Representations from Transformers

**Hao Tan** **Mohit Bansal**
UNC Chapel Hill
{haotan, mbansal}@cs.unc.edu

## Abstract

Vision-and-language reasoning requires an understanding of visual concepts, language semantics, and, most importantly, the alignment and relationships between these two modalities. We thus propose the LXMERT (Learning Cross-Modality Encoder Representations from Transformers) framework to learn these vision-and-language connections. In LXMERT, we build a large-scale Transformer model that consists of three encoders: an object relationship encoder, a language encoder, and a cross-modality encoder. Next, to endow our model with the capability of connecting vision and language semantics, we pre-train the model with large amounts of image-and-sentence pairs, via five diverse representative pre-training tasks: masked language modeling, masked object prediction (feature regression and label classification), cross-modality matching, and image question answering. These tasks help in learning both intra-modality and cross-modality relationships. After fine-tuning from our pre-trained parameters, our model achieves the state-of-the-art results on two visual question answering datasets (i.e., VQA and GQA). We also show the generalizability of our pre-trained cross-modality model by adapting it to a challenging visual-reasoning task, NLVR$^2$, and improve the previous best result by 22% absolute (54% to 76%). Lastly, we demonstrate detailed ablation studies to prove that both our novel model components and pre-training strategies significantly contribute to our strong results.[1]

## 1 Introduction

Vision-and-language reasoning requires the understanding of visual contents, language semantics, and cross-modal alignments and relationships. There has been substantial past works in separately developing backbone models with better representations for the single modalities of vision and of language. For visual-content understanding, people have developed several backbone models (Simonyan and Zisserman, 2014; Szegedy et al., 2015; He et al., 2016) and shown their effectiveness on large vision datasets (Deng et al., 2009; Lin et al., 2014; Krishna et al., 2017). Pioneering works (Girshick et al., 2014; Xu et al., 2015) also show the generalizability of these pre-trained (especially on ImageNet) backbone models by fine-tuning them on different tasks. In terms of language understanding, last year, we witnessed strong progress towards building a universal backbone model with large-scale contextualized language model pre-training (Peters et al., 2018; Radford et al., 2018; Devlin et al., 2019), which has improved performances on various tasks (Rajpurkar et al., 2016; Wang et al., 2018) to significant levels. Despite these influential single-modality works, large-scale pretraining and fine-tuning studies for the modality-pair of vision and language are still under-developed.

Therefore, we present one of the first works in building a pre-trained vision-and-language cross-modality framework and show its strong performance on several datasets. We name this framework "LXMERT: Learning Cross-Modality Encoder Representations from Transformers" (pronounced: 'leksmert'). This framework is modeled after recent BERT-style innovations while further adapted to useful cross-modality scenarios. Our new cross-modality model focuses on learning vision-and-language interactions, especially for representations of a single image and its descriptive sentence. It consists of three Transformer (Vaswani et al., 2017) encoders: an object relationship encoder, a language encoder, and a cross-modality encoder. In order to better learn

[1] Published at EMNLP 2019. Code and pre-trained models publicly available at: https://github.com/airsplay/lxmert



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5100–5111,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

the cross-modal alignments between vision and language, we next pre-train our model with five diverse representative tasks: (1) masked cross-modality language modeling, (2) masked object prediction via RoI-feature regression, (3) masked object prediction via detected-label classification, (4) cross-modality matching, and (5) image question answering. Different from single-modality pre-training (e.g., masked LM in BERT), this multi-modality pre-training allows our model to infer masked features either from the visible elements in the same modality, or from aligned components in the other modality. In this way, it helps build both intra-modality and cross-modality relationships.

Empirically, we first evaluate LXMERT on two popular visual question-answering datasets, VQA (Antol et al., 2015) and GQA (Hudson and Manning, 2019). Our model outperforms previous works in all question categories (e.g., Binary, Number, Open) and achieves state-of-the-art results in terms of overall accuracy. Further, to show the generalizability of our pre-trained model, we fine-tune LXMERT on a challenging visual reasoning task, Natural Language for Visual Reasoning for Real (NLVR$^2$) (Suhr et al., 2019), where we do not use the natural images in their dataset for our pre-training, but fine-tune and evaluate on these challenging, real-world images. In this setup, we achieve a large improvement of 22% absolute in accuracy (54% to 76%, i.e., 48% relative error reduction) and 30% absolute in consistency (12% to 42%, i.e., 34% relative error reduction). Lastly, we conduct several analysis and ablation studies to prove the effectiveness of our model components and diverse pre-training tasks by removing them or comparing them with their alternative options. Especially, we use several ways to take the existing BERT model and its variants, and show their ineffectiveness in vision-and-language tasks, which overall proves the need of our new cross-modality pre-training framework.

## 2 Model Architecture

We build our cross-modality model with self-attention and cross-attention layers following the recent progress in designing natural language processing models (e.g., transformers (Vaswani et al., 2017)). As shown in Fig. 1, our model takes two inputs: an image and its related sentence (e.g., a caption or a question). Each image is represented as a sequence of objects, and each sentence is represented as a sequence of words. Via careful design and combination of these self-attention and cross-attention layers, our model is able to generate language representations, image representations, and cross-modality representations from the inputs. Next, we describe the components of this model in detail.

### 2.1 Input Embeddings

The input embedding layers in LXMERT convert the inputs (i.e., an image and a sentence) into two sequences of features: word-level sentence embeddings and object-level image embeddings. These embedding features will be further processed by the latter encoding layers.

**Word-Level Sentence Embeddings** A sentence is first split into words $\{w_1, \ldots, w_n\}$ with length of $n$ by the same WordPiece tokenizer (Wu et al., 2016) in Devlin et al. (2019). Next, as shown in Fig. 1, the word $w_i$ and its index $i$ ($w_i$'s absolute position in the sentence) are projected to vectors by embedding sub-layers, and then added to the index-aware word embeddings:

$$\begin{aligned}\hat{w}_i &= \text{WordEmbed}\left(w_i\right)\\ \hat{u}_i &= \text{IdxEmbed}\left(i\right)\\ h_i &= \text{LayerNorm}\left(\hat{w}_i + \hat{u}_i\right)\end{aligned}$$

**Object-Level Image Embeddings** Instead of using the feature map output by a convolutional neural network, we follow Anderson et al. (2018) in taking the features of detected objects as the embeddings of images. Specifically, the object detector detects $m$ objects $\{o_1, \ldots, o_m\}$ from the image (denoted by bounding boxes on the image in Fig. 1). Each object $o_j$ is represented by its position feature (i.e., bounding box coordinates) $p_j$ and its 2048-dimensional region-of-interest (RoI) feature $f_j$. Instead of directly using the RoI feature $f_j$ without considering its position $p_j$ in Anderson et al. (2018), we learn a position-aware embedding $v_j$ by adding outputs of 2 fully-connected layers:

$$\begin{aligned}\hat{f}_j &= \text{LayerNorm}\left(W_{\text{F}} f_j + b_{\text{F}}\right)\\ \hat{p}_j &= \text{LayerNorm}\left(W_{\text{P}} p_j + b_{\text{P}}\right)\\ v_j &= \left(\hat{f}_j + \hat{p}_j\right)/2\end{aligned} \quad (1)$$

In addition to providing spatial information in visual reasoning, the inclusion of positional information is necessary for our masked object prediction pre-training task (described in Sec. 3.1.2).

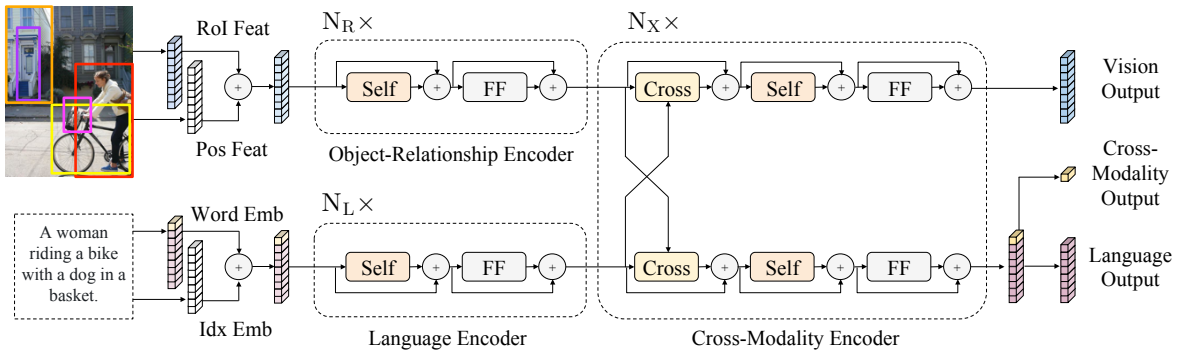


Figure 1: The LXMERT model for learning vision-and-language cross-modality representations. 'Self' and 'Cross' are abbreviations for self-attention sub-layers and cross-attention sub-layers, respectively. 'FF' denotes a feed-forward sub-layer.

Since the image embedding layer and the following attention layers are agnostic to the absolute indices of their inputs, the order of the object is not specified. Lastly, in Equation 1, the layer normalization is applied to the projected features before summation so as to balance the energy of the two different types of features.

## 2.2 Encoders

We build our encoders, i.e., the language encoder, the object-relationship encoder, and the cross-modality encoder, mostly on the basis of two kinds of attention layers: self-attention layers and cross-attention layers. We first review the definition and notations of attention layers and then discuss how they form our encoders.

**Background: Attention Layers** Attention layers (Bahdanau et al., 2014; Xu et al., 2015) aim to retrieve information from a set of *context* vectors $\{y_j\}$ related to a *query* vector $x$. An attention layer first calculates the matching score $a_j$ between the *query* vector $x$ and each *context* vector $y_j$. Scores are then normalized by softmax:

$$a_j = \text{score}(x, y_j)$$
$$\alpha_j = \exp(a_j) / \sum_k \exp(a_k)$$

The output of an attention layer is the weighted sum of the *context* vectors w.r.t. the softmax-normalized score: $\text{Att}_{\text{X}\to\text{Y}}(x, \{y_j\}) = \sum_j \alpha_j y_j$. An attention layer is called *self-attention* when the *query* vector $x$ is in the set of *context* vectors $\{y_j\}$. Specifically, we use the multi-head attention following Transformer (Vaswani et al., 2017).

**Single-Modality Encoders** After the embedding layers, we first apply two transformer encoders (Vaswani et al., 2017), i.e., a **language encoder** and an **object-relationship encoder**, and each of them only focuses on a single modality (i.e., language or vision). Different from BERT (Devlin et al., 2019), which applies the transformer encoder only to language inputs, we apply it to vision inputs as well (and to cross-modality inputs as described later below). Each layer (left dashed blocks in Fig. 1) in a single-modality encoder contains a self-attention ('Self') sub-layer and a feed-forward ('FF') sub-layer, where the feed-forward sub-layer is further composed of two fully-connected sub-layers. We take $\mathbf{N_L}$ and $\mathbf{N_R}$ layers in the language encoder and the object-relationship encoder, respectively. We add a residual connection and layer normalization (annotated by the '+' sign in Fig. 1) after each sub-layer as in Vaswani et al. (2017).

**Cross-Modality Encoder** Each cross-modality layer (the right dashed block in Fig. 1) in the cross-modality encoder consists of two self-attention sub-layers, one bi-directional cross-attention sub-layer, and two feed-forward sub-layers. We stack (i.e., using the output of $k$-th layer as the input of $(k{+}1)$-th layer) $\mathbf{N_X}$ these cross-modality layers in our encoder implementation. Inside the $k$-th layer, the bi-directional cross-attention sub-layer ('Cross') is first applied, which contains two uni-directional cross-attention sub-layers: one from language to vision and one from vision to language. The query and context vectors are the outputs of the $(k{-}1)$-th layer (i.e., language features $\{h_i^{k-1}\}$ and vision features $\{v_j^{k-1}\}$):

$$\hat{h}_i^k = \text{CrossAtt}_{\text{L}\to\text{R}}\left(h_i^{k-1}, \{v_1^{k-1}, \ldots, v_m^{k-1}\}\right)$$
$$\hat{v}_j^k = \text{CrossAtt}_{\text{R}\to\text{L}}\left(v_j^{k-1}, \{h_1^{k-1}, \ldots, h_n^{k-1}\}\right)$$

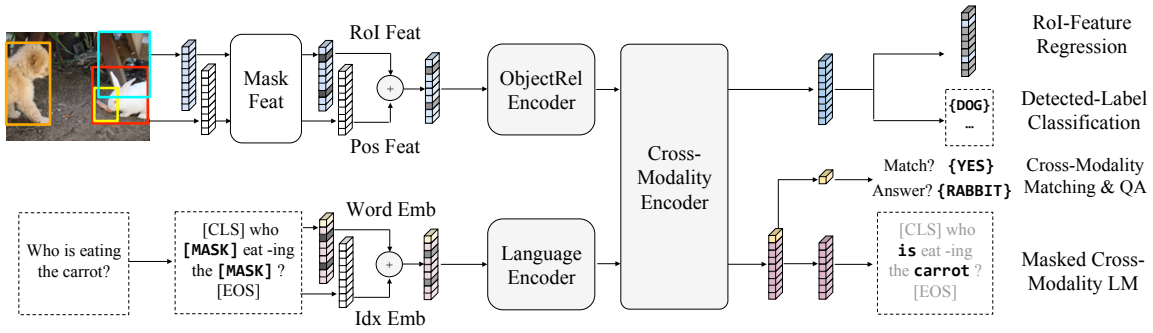


Figure 2: Pre-training in LXMERT. The object RoI features and word tokens are masked. Our five pre-training tasks learn the feature representations based on these masked inputs. Special tokens are in brackets and classification labels are in braces.

The cross-attention sub-layer is used to exchange the information and align the entities between the two modalities in order to learn joint cross-modality representations. For further building internal connections, the self-attention sub-layers ('Self') are then applied to the output of the cross-attention sub-layer:

$$\tilde{h}_i^k = \text{SelfAtt}_{\text{L}\to\text{L}}\left(\hat{h}_i^k, \{\hat{h}_1^k, \ldots, \hat{h}_n^k\}\right)$$
$$\tilde{v}_j^k = \text{SelfAtt}_{\text{R}\to\text{R}}\left(\hat{v}_j^k, \{\hat{v}_1^k, \ldots, \hat{v}_m^k\}\right)$$

Lastly, the $k$-th layer output $\{h_i^k\}$ and $\{v_j^k\}$ are produced by feed-forward sub-layers ('FF') on top of $\{\hat{h}_i^k\}$ and $\{\hat{v}_j^k\}$. We also add a residual connection and layer normalization after each sub-layer, similar to the single-modality encoders.

### 2.3 Output Representations

As shown in the right-most part of Fig. 1, our LXMERT cross-modality model has three outputs for language, vision, and cross-modality, respectively. The language and vision outputs are the feature sequences generated by the cross-modality encoder. For the cross-modality output, following the practice in Devlin et al. (2019), we append a special token [CLS] (denoted as the top yellow block in the bottom branch of Fig. 1) before the sentence words, and the corresponding feature vector of this special token in language feature sequences is used as the cross-modality output.

## 3 Pre-Training Strategies

In order to learn a better initialization which understands connections between vision and language, we pre-train our model with different modality pre-training tasks on a large aggregated dataset.

### 3.1 Pre-Training Tasks

#### 3.1.1 Language Task: Masked Cross-Modality LM

On the language side, we take the masked cross-modality language model (LM) task. As shown in the bottom branch of Fig. 2, the task setup is almost same to BERT (Devlin et al., 2019): words are randomly masked with a probability of 0.15 and the model is asked to predict these masked words. In addition to BERT where masked words are predicted from the non-masked words in the language modality, LXMERT, with its cross-modality model architecture, could predict masked words from the vision modality as well, so as to resolve ambiguity. For example, as shown in Fig. 2, it is hard to determine the masked word 'carrot' from its language context but the word choice is clear if the visual information is considered. Hence, it helps building connections from the vision modality to the language modality, and we refer to this task as masked *cross-modality* LM to emphasize this difference. We also show that loading BERT parameters into LXMERT will do harm to the pre-training procedure in Sec. 5.1 since BERT can perform relatively well in the language modality without learning these cross-modality connections.

#### 3.1.2 Vision Task: Masked Object Prediction

As shown in the top branch of Fig. 2, we pre-train the vision side by randomly masking objects (i.e., masking RoI features with zeros) with a probability of 0.15 and asking the model to predict proprieties of these masked objects. Similar to the language task (i.e., masked cross-modality LM), the model can infer the masked objects either from visible objects or from the language modality. Inferring the objects from the vision

| Image Split | Images | Sentences (or Questions) | | | | | |
|---|---|---|---|---|---|---|---|
| | | COCO-Cap | VG-Cap | VQA | GQA | VG-QA | All |
| MS COCO - VG | 72K | 361K | - | 387K | - | - | 0.75M |
| MS COCO ∩ VG | 51K | 256K | 2.54M | 271K | 515K | 724K | 4.30M |
| VG - MS COCO | 57K | - | 2.85M | - | 556K | 718K | 4.13M |
| All | 180K | 617K | 5.39M | 658K | 1.07M | 1.44M | 9.18M |

Table 1: Amount of data for pre-training. Each image has multiple sentences/questions. 'Cap' is caption. 'VG' is Visual Genome. Since MS COCO and VG share 51K images, we list it separately to ensure disjoint image splits.

side helps learn the object relationships, and inferring from the language side helps learn the cross-modality alignments. Therefore, we perform two sub-tasks: **RoI-Feature Regression** regresses the object RoI feature $f_j$ with L2 loss, and **Detected-Label Classification** learns the labels of masked objects with cross-entropy loss. In the 'Detected-Label Classification' sub-task, although most of our pre-training images have object-level annotations, the ground truth labels of the annotated objects are inconsistent in different datasets (e.g., different number of label classes). For these reasons, we take detected labels output by Faster R-CNN (Ren et al., 2015). Although detected labels are noisy, experimental results show that these labels contribute to pre-training in Sec. 5.3.

#### 3.1.3 Cross-Modality Tasks

As shown in the middle-rightmost part of Fig. 2, to learn a strong cross-modality representation, we pre-train the LXMERT model with 2 tasks that explicitly need both language and vision modalities.

**Cross-Modality Matching** For each sentence, with a probability of 0.5, we replace it with a mismatched[2] sentence. Then, we train a classifier to predict whether an image and a sentence match each other. This task is similar to 'Next Sentence Prediction' in BERT (Devlin et al., 2019).

**Image Question Answering (QA)** In order to enlarge the pre-training dataset (see details in Sec. 3.2), around $1/3$ sentences in the pre-training data are questions about the images. We ask the model to predict the answer to these image-related questions when the image and the question are matched (i.e., not randomly replaced in the cross-modality matching task). We show that pre-training with this image QA leads to a better cross-modality representation in Sec. 5.2.

[2] We take a sentence from another image as the mismatched sentence. Although the sentence and the image still have chance to match each other, this probability is very low.

### 3.2 Pre-Training Data

As shown in Table. 1, we aggregate pre-training data from five vision-and-language datasets whose images come from MS COCO (Lin et al., 2014) or Visual Genome (Krishna et al., 2017). Besides the two original captioning datasets, we also aggregate three large image question answering (image QA) datasets: VQA v2.0 (Antol et al., 2015), GQA balanced version (Hudson and Manning, 2019), and VG-QA (Zhu et al., 2016). We only collect **train and dev** splits in each dataset to avoid seeing any test data in pre-training. We conduct minimal pre-processing on the five datasets to create aligned image-and-sentence pairs. For each image question answering dataset, we take questions as sentences from the image-and-sentence data pairs and take answers as labels in the image QA pre-training task (described in Sec. 3.1.3). This provides us with a large aligned vision-and-language dataset of 9.18M image-and-sentence pairs on 180K distinct images. In terms of tokens, the pre-training data contain around 100M words and 6.5M image objects.

### 3.3 Pre-Training Procedure

We pre-train our LXMERT model on the large aggregated dataset (discussed in Sec. 3.2) via the pre-training tasks (Sec. 3.1). The details about the data splits are in the Appendix. The input sentences are split by the WordPiece tokenizer (Wu et al., 2016) provided in BERT (Devlin et al., 2019). The objects are detected by Faster R-CNN (Ren et al., 2015) which is pre-trained on Visual Genome (provided by Anderson et al. (2018)). We do not fine-tune the Faster R-CNN detector and freeze it as a feature extractor. Different from detecting variable numbers of objects in Anderson et al. (2018), we consistently keep 36 objects for each

| Method | VQA | | | | GQA | | | NLVR$^2$ | |
|---|---|---|---|---|---|---|---|---|---|
| | Binary | Number | Other | **Accu** | Binary | Open | **Accu** | Cons | **Accu** |
| Human | - | - | - | - | 91.2 | 87.4 | 89.3 | - | 96.3 |
| Image Only | - | - | - | - | 36.1 | 1.74 | 17.8 | 7.40 | 51.9 |
| Language Only | 66.8 | 31.8 | 27.6 | 44.3 | 61.9 | 22.7 | 41.1 | 4.20 | 51.1 |
| State-of-the-Art | 85.8 | 53.7 | 60.7 | 70.4 | 76.0 | 40.4 | 57.1 | 12.0 | 53.5 |
| LXMERT | **88.2** | **54.2** | **63.1** | **72.5** | **77.8** | **45.0** | **60.3** | **42.1** | **76.2** |

Table 2: Test-set results. VQA/GQA results are reported on the 'test-standard' splits and NLVR$^2$ results are reported on the unreleased test set ('Test-U'). The highest method results are in bold. Our LXMERT framework outperforms previous (comparable) state-of-the-art methods on all three datasets w.r.t. all metrics.

image to maximize the pre-training compute utilization by avoiding padding. For the model architecture, we set the numbers of layers $\mathbf{N_L}$, $\mathbf{N_X}$, and $\mathbf{N_R}$ to 9, 5, and 5 respectively.[3] More layers are used in the language encoder to balance the visual features extracted from 101-layer Faster R-CNN. The hidden size 768 is the same as $\text{BERT}_{\text{BASE}}$. We pre-train all parameters in encoders and embedding layers from scratch (i.e., model parameters are randomly initialized or set to zero). We also show results of loading pre-trained BERT parameters in Sec. 5.1. LXMERT is pre-trained with multiple pre-training tasks and hence multiple losses are involved. We add these losses with equal weights. For the image QA pre-training tasks, we create a joint answer table with 9500 answer candidates which roughly cover 90% questions in all three image QA datasets.

We take Adam (Kingma and Ba, 2014) as the optimizer with a linear-decayed learning-rate schedule (Devlin et al., 2019) and a peak learning rate at $1e-4$. We train the model for 20 epochs (i.e., roughly 670K[4] optimization steps) with a batch size of 256. We only pre-train with image QA task (see Sec. 3.1.3) for the last 10 epochs, because this task converges faster and empirically needs a smaller learning rate. The whole pre-training process takes 10 days on 4 Titan Xp.

**Fine-tuning** Fine-tuning is fast and robust. We only perform necessary modification to our model with respect to different tasks (details in Sec. 4.2). We use a learning rate of $1e-5$ or $5e-5$, a batch size of 32, and fine-tune the model from our pre-trained parameters for 4 epochs.

[3]If we count a single modality layer as one half cross-modality layer, the equivalent number of cross-modality layers is $(9+5)/2+5=12$, which is same as the number of layers in $\text{BERT}_{\text{BASE}}$.

[4]For comparison, ResNet on ImageNet classification takes 600K steps and BERT takes 1000K steps.

## 4 Experimental Setup and Results

In this section, we first introduce the datasets that are used to evaluate our LXMERT framework and empirically compare our single-model results with previous best results.

### 4.1 Evaluated Datasets

We use three datasets for evaluating our LXMERT framework: VQA v2.0 dataset (Goyal et al., 2017), GQA (Hudson and Manning, 2019), and NLVR$^2$. See details in Appendix.

### 4.2 Implementation Details

On VQA and GQA, we fine-tune our model from the pre-trained snapshot without data augmentation (analysis in Sec. 5.2). When training GQA, we only take raw questions and raw images as inputs and do not use other supervisions (e.g., functional programs and scene graphs). Since each datum in NLVR$^2$ has two natural images $img_0, img_1$ and one language statement $s$, we use LXMERT to encode the two image-statement pairs $(img_0, s)$ and $(img_1, s)$, then train a classifier based on the concatenation of the two cross-modality outputs. More details in Appendix.

### 4.3 Empirical Comparison Results

We compare our single-model results with previous best published results on VQA/GQA test-standard sets and NLVR$^2$ public test set. Besides previous state-of-the-art (SotA) methods, we also show the human performance and image-only/language-only results when available.

**VQA** The SotA result is BAN+Counter in Kim et al. (2018), which achieves the best accuracy among other recent works: MFH (Yu et al.,

2018), Pythia (Jiang et al., 2018), DFAF (Gao et al., 2019a), and Cycle-Consistency (Shah et al., 2019).[5] LXMERT improves the SotA overall *accuracy* ('Accu' in Table 2) by 2.1% and has 2.4% improvement on the 'Binary'/'Other' question sub-categories. Although LXMERT does not explicitly take a counting module as in BAN+Counter, our result on the counting-related questions ('Number') is still equal or better.[6]

**GQA** The GQA (Hudson and Manning, 2019) SotA result is taken from BAN (Kim et al., 2018) on the public leaderbaord. Our 3.2% *accuracy* gain over the SotA GQA method is higher than VQA, possibly because GQA requires more visual reasoning. Thus our framework, with novel encoders and cross-modality pre-training, is suitable and achieves a 4.6% improvement on open-domain questions ('Open' in Table 2).[7]

**NLVR$^2$** NLVR$^2$ (Suhr et al., 2019) is a challenging visual reasoning dataset where some existing approaches (Hu et al., 2017; Perez et al., 2018) fail, and the SotA method is 'MaxEnt' in Suhr et al. (2019). The failure of existing methods (and our model w/o pre-training in Sec. 5.1) indicates that the connection between vision and language may not be end-to-end learned in a complex vision-and-language task without large-scale pre-training. However, with our novel pre-training strategies in building the cross-modality connections, we significantly improve the *accuracy* ('Accu' of 76.2% on unreleased test set 'Test-U', in Table 2) by 22%. Another evaluation metric *consistency* measures the proportion of unique sentences for which all related image pairs[8] are correctly predicted. Our LXMERT model improves *consistency* ('Cons') to 42.1% (i.e., by 3.5 times).[9]

[5] These are state-of-the-art methods at the time of our EMNLP May 21, 2019 submission deadline. Since then, there have been some recently updated papers such as MCAN (Yu et al., 2019b), MUAN (Yu et al., 2019a), and MLI (Gao et al., 2019b). MCAN (VQA challenge version) uses stronger mixture of detection features and achieves 72.8% on VQA 2.0 test-standard. MUAN achieves 71.1% (compared to our 72.5%).

[6] Our result on VQA v2.0 'test-dev' is 72.4%.

[7] Our result on GQA 'test-dev' is 60.0%.

[8] Each statement in NLVR$^2$ is related to multiple image pairs in order to balance the dataset answer distribution.

[9] These are the unreleased test set ('Test-U') results. On the public test set ('Test-P'), LXMERT achieves 74.5% Accu and 39.7% Cons.

| Method | VQA | GQA | NLVR$^2$ |
|---|---|---|---|
| LSTM + BUTD | 63.1 | 50.0 | 52.6 |
| BERT + BUTD | 62.8 | 52.1 | 51.9 |
| BERT + 1 CrossAtt | 64.6 | 55.5 | 52.4 |
| BERT + 2 CrossAtt | 65.8 | 56.1 | 50.9 |
| BERT + 3 CrossAtt | 66.4 | 56.6 | 50.9 |
| BERT + 4 CrossAtt | 66.4 | 56.0 | 50.9 |
| BERT + 5 CrossAtt | 66.5 | 56.3 | 50.9 |
| Train + BERT | 65.5 | 56.2 | 50.9 |
| Train + scratch | 65.1 | 50.0 | 50.9 |
| Pre-train + BERT | 68.8 | 58.3 | 70.1 |
| **Pre-train + scratch** | **69.9** | **60.0** | **74.9** |

Table 3: Dev-set accuracy of using BERT.

## 5 Analysis

In this section, we analyze our LXMERT framework by comparing it with some alternative choices or by excluding certain model components/pre-training strategies.

### 5.1 BERT versus LXMERT

BERT (Devlin et al., 2019) is a pre-trained language encoder which improves several language tasks. As shown in Table 3, we discuss several ways to incorporate a $\text{BERT}_{\text{BASE}}$ pre-trained model for vision-language tasks and empirically compare it with our LXMERT approach. Although our full model achieves accuracy of 74.9% on NLVR$^2$, all results without LXMERT pre-training is around 22% absolute lower.

**BERT+BUTD** Bottom-Up and Top-Down (BUTD) attention (Anderson et al., 2018) method encodes questions with GRU (Chung et al., 2015), then attends to object RoI features $\{f_j\}$ to predict the answer. We apply BERT to BUTD by replacing its GRU language encoder with BERT. As shown in the first block of Table. 3, results of BERT encoder is comparable to LSTM encoder.

**BERT+CrossAtt** Since BUTD only takes the raw RoI features $\{f_j\}$ without considering the object positions $\{p_j\}$ and object relationships, we enhance BERT+BUTD with our novel position-aware object embedding (in Sec. 2.1) and cross-modality layers (in Sec. 2.2). As shown in the second block of Table 3, the result of 1 cross-modality layer is better than BUTD, while stacking more cross-modality layers further improves it. However, without our cross-modality pre-

| Method | VQA | GQA | NLVR$^2$ |
|---|---|---|---|
| 1. P20 + DA | 68.0 | 58.1 | - |
| 2. P20 + FT | 68.9 | 58.2 | 72.4 |
| 3. P10+QA10 + DA | 69.1 | 59.2 | - |
| **4. P10+QA10 + FT** | **69.9** | **60.0** | **74.9** |

Table 4: Dev-set accuracy showing the importance of the image-QA pre-training task. P10 means pre-training without the image-QA loss for 10 epochs while QA10 means pre-training with the image-QA loss. DA and FT mean fine-tuning with and without Data Augmentation, resp.

| Method | VQA | GQA | NLVR$^2$ |
|---|---|---|---|
| 1. No Vision Tasks | 66.3 | 57.1 | 50.9 |
| 2. Feat | 69.2 | 59.5 | 72.9 |
| 3. Label | 69.5 | 59.3 | 73.5 |
| **4. Feat + Label** | **69.9** | **60.0** | **74.9** |

Table 5: Dev-set accuracy of different vision pre-training tasks. 'Feat' is RoI-feature regression; 'Label' is detected-label classification.

training (BERT is language-only pre-trained), results become stationary after adding 3 cross-attention layers and have a 3.4% gap to our full LXMERT framework (the last bold row in Table 3).

**BERT+LXMERT** We also try loading BERT parameters[10] into LXMERT, and use it in model training (i.e., without LXMERT pre-training) or in pre-training. We show results in the last block of Table. 3. Compared to the 'from scratch' (i.e., model parameters are randomly initialized) approach, BERT improves the fine-tuning results but it shows weaker results than our full model. Empirically, pre-training LXMERT initialized with BERT parameters has lower (i.e., better) pre-training loss for the first 3 pre-training epochs but was then caught up by our 'from scratch' approach. A possible reason is that BERT is already pre-trained with single-modality masked language model, and thus could do well based only on the language modality without considering the connection to the vision modality (as discussed in Sec. 3.1.1).

### 5.2 Effect of the Image QA Pre-training Task

We show the importance of image QA pre-training task (introduced in Sec. 3.1.3) by excluding it or comparing it with its alternative: data augmentation.

**Pre-training w/ or w/o Image QA** To fairly compare with our original pre-training procedure (10 epochs w/o QA + 10 epochs w/ QA, details in Sec. 3.3) , we pre-train LXMERT model without image QA task for 20 epochs. As shown in Table 4 rows 2 and 4, pre-training with QA loss improves the result on all three datasets. The 2.1% improvement on NLVR$^2$ shows the stronger representations learned with image-QA pre-training, since all data (images and statements) in NLVR$^2$ are not used in pre-training.

**Pre-training versus Data Augmentation** Data augmentation (DA) is a technique which is used in several VQA implementations (Anderson et al., 2018; Kim et al., 2018; Jiang et al., 2018). It increases the amount of training data by adding questions from other image QA datasets. Our LXMERT framework instead uses multiple QA datasets in pre-training and is fine-tuned only on one specific dataset. Since the overall amounts of data used in pre-training and DA are similar, we thus can fairly compare these two strategies, and results show that our QA pre-training approach outperforms DA. We first exclude the QA task in our pre-training and show the results of DA fine-tuning. As shown in Table. 4 row 1, DA fine-tuning decreases the results compared to non-DA fine-tuning in row 2. Next, we use DA after QA-pre-training (row 3) and DA also drops the results.

### 5.3 Effect of Vision Pre-training tasks

We analyze the effect of different vision pre-training tasks in Table 5. Without any vision tasks in pre-training (i.e., only using the language and cross-modality pre-training tasks), the results (row 1 of Table 5) are similar to BERT+3 CrossAtt in Table 3. The two visual pre-training tasks (i.e., RoI-feature regression and detected-label classification) could get reasonable results (row 2 and row 3) on their own, and jointly pre-training with these two tasks achieves the highest results (row 4).

## 6 Related Work

**Model Architecture:** Our model is closely related to three ideas: bi-directional attention, Transformer, and BUTD. Lu et al. (2016) applies bi-

[10] Since our language encoder is same as BERT$_{\text{BASE}}$, except the number of layers (i.e., LXMERT has 9 layers and BERT has 12 layers), we load the top 9 BERT-layer parameters into the LXMERT language encoder.

directional attention to the vision-and-language tasks while its concurrent work BiDAF (Seo et al., 2017) adds modeling layers in solving reading comprehension. Transformer (Vaswani et al., 2017) is first used in machine translation, we utilize it as our single-modality encoders and design our cross-modality encoder based on it. BUTD (Anderson et al., 2018) embeds images with the object RoI features, we extend it with object positional embeddings and object relationship encoders.

**Pre-training:** After ELMo (Peters et al., 2018), GPT (Radford et al., 2018), and BERT (Devlin et al., 2019) show improvements in language understanding tasks with large-scale pre-trained language model, progress has been made towards the cross-modality pre-training. XLM (Lample and Conneau, 2019) learns the joint cross-lingual representations by leveraging the monolingual data and parallel data. VideoBert (Sun et al., 2019) takes masked LM on the concatenation of language words and visual tokens, where the visual tokens are converted from video frames by vector quantization. However, these methods are still based on a single transformer encoder and BERT-stype token-based pre-training, thus we develop a new model architecture and novel pre-training tasks to satisfy the need of cross-modality tasks.

**Recent works since our EMNLP submission:** This version of our paper (and all current results) was submitted to EMNLP[11] and was used to participate in the VQA and GQA challenges in May 2019. Since our EMNLP submission, a few other useful preprints have recently been released (in August) on similar cross-modality pre-training directions: ViLBERT (Lu et al., 2019) and VisualBERT (Li et al., 2019). Our LXMERT methods differs from them in multiple ways: we use a more detailed, multi-component design for the cross-modality model (i.e., with an object-relationship encoder and cross-modality layers) and we employ additional, useful pre-training tasks (i.e., RoI-feature regression and image question answering). These differences result in the current best performance (on overlapping reported tasks): a margin of 1.5% accuracy on VQA 2.0 and a margin of 9% accuracy on NLVR$^2$ (and 15% in consistency). LXMERT is also the only method which ranks in the top-3 on both the VQA and GQA challenges among more than 90 teams. We provide a detailed analysis to show how these additional pre-training tasks contribute to the fine-tuning performance in Sec. 5.2 and Sec. 5.3.

[11] EMNLP deadline was on May 21, 2019, and the standard ACL/EMNLP arxiv ban rule was in place till the notification date of August 12, 2019.

## 7 Conclusion

We presented a cross-modality framework, LXMERT, for learning the connections between vision and language. We build the model based on Transfermer encoders and our novel cross-modality encoder. This model is then pre-trained with diverse pre-training tasks on a large-scale dataset of image-and-sentence pairs. Empirically, we show state-of-the-art results on two image QA datasets (i.e., VQA and GQA) and show the model generalizability with a 22% improvement on the challenging visual reasoning dataset of NLVR$^2$. We also show the effectiveness of several model components and training methods via detailed analysis and ablation studies.

## Acknowledgments

We thank the reviewers for their helpful comments. This work was supported by ARO-YIP Award #W911NF-18-1-0336, and awards from Google, Facebook, Salesforce, and Adobe. The views, opinions, and/or findings contained in this article are those of the authors and should not be interpreted as representing the official views or policies, either expressed or implied, of the funding agency. We also thank Alane Suhr for evaluation on NLVR$^2$.

## References

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6077–6086.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2015. Gated feedback recur-

rent neural networks. In *International Conference on Machine Learning*, pages 2067–2075.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*.

Peng Gao, Zhengkai Jiang, Haoxuan You, Pan Lu, Steven C. H. Hoi, Xiaogang Wang, and Hongsheng Li. 2019a. Dynamic fusion with intra- and inter-modality attention flow for visual question answering. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Peng Gao, Haoxuan You, Zhanpeng Zhang, Xiaogang Wang, and Hongsheng Li. 2019b. Multi-modality latent interaction network for visual question answering. *arXiv preprint arXiv:1908.04289*.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6904–6913.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Dan Hendrycks and Kevin Gimpel. 2016. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *https://openreview.net/forum?id=Bk0MRI5lg*.

Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. 2017. Learning to reason: End-to-end module networks for visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 804–813.

Drew A Hudson and Christopher D Manning. 2019. Gqa: a new dataset for compositional question answering over real-world images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Yu Jiang, Vivek Natarajan, Xinlei Chen, Marcus Rohrbach, Dhruv Batra, and Devi Parikh. 2018. Pythia v0. 1: the winning entry to the vqa challenge 2018. *arXiv preprint arXiv:1807.09956*.

Jin-Hwa Kim, Jaehyun Jun, and Byoung-Tak Zhang. 2018. Bilinear attention networks. In *Advances in Neural Information Processing Systems*, pages 1564–1574.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123(1):32–73.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*.

Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. 2019. Visualbert: A simple and performant baseline for vision and language. *arXiv preprint arXiv:1908.03557*.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *arXiv preprint arXiv:1908.02265*.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2016. Hierarchical question-image co-attention for visual question answering. In *Advances In Neural Information Processing Systems*, pages 289–297.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. 2018. Film: Visual reasoning with a general conditioning layer. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *International Conference on Learning Representations*.

Meet Shah, Xinlei Chen, Marcus Rohrbach, and Devi Parikh. 2019. Cycle-consistency for robust visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Alane Suhr, Stephanie Zhou, Iris Zhang, Huajun Bai, and Yoav Artzi. 2019. A corpus for reasoning about natural language grounded in photographs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Chen Sun, Austin Myers, Carl Vondrick, Kevin Murphy, and Cordelia Schmid. 2019. Videobert: A joint model for video and language representation learning. *arXiv preprint arXiv:1904.01766*.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *EMNLP 2018*, page 353.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057.

Zhou Yu, Yuhao Cui, Jun Yu, Dacheng Tao, and Qi Tian. 2019a. Multimodal unified attention networks for vision-and-language interactions. *arXiv preprint arXiv:1908.04107*.

Zhou Yu, Jun Yu, Yuhao Cui, Dacheng Tao, and Qi Tian. 2019b. Deep modular co-attention networks for visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6281–6290.

Zhou Yu, Jun Yu, Chenchao Xiang, Jianping Fan, and Dacheng Tao. 2018. Beyond bilinear: Generalized multimodal factorized high-order pooling for visual question answering. *IEEE Transactions on Neural Networks and Learning Systems*, 29(12):5947–5959.

Yuke Zhu, Oliver Groth, Michael Bernstein, and Li Fei-Fei. 2016. Visual7w: Grounded question answering in images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4995–5004.

# Appendix

## A Evaluated Datasets Description

We use three datasets for evaluating our LXMERT framework.

**VQA** The goal of visual question answering (VQA) (Antol et al., 2015) is to answer a natural language question related to an image. We take VQA v2.0 dataset (Goyal et al., 2017) which reduces the answer bias compared to VQA v1.0. The dataset contains an average of 5.4 questions per image and the total amount of questions is 1.1M.

**GQA** The task of GQA (Hudson and Manning, 2019) is same as VQA (i.e., answer single-image related questions), but GQA requires more reasoning skills (e.g., spatial understanding and multi-step inference). 22M questions in the dataset are generated from ground truth image scene graph to explicitly control the question quality.

**NLVR$^2$** Since the previous two datasets are used in pre-training for increasing the amount of pre-training data to a certain scale, we evaluate our LXMERT framework on another challenging visual reasoning dataset NLVR$^2$ where all the sentences and images are not covered in pre-training. Each datum in NLVR$^2$ contains two related natural images and one natural language statement. The task is to predict whether the statement correctly describes these two images or not. NLVR$^2$ has 86K, 7K, 7K data in training, development, and test sets, respectively.

## B Details of NLVR$^2$ Fine-tuning

Each datum in NLVR$^2$ consists of a two-image pair ($img_0$, $img_1$), one statement $s$, and a ground truth label $y^*$ indicating whether the statement correctly describe the two images. The task is to predict the label $y$ given the images and the statement.

To use our LXMERT model on NLVR$^2$, we concatenate the cross-modality representations of the two images and then build the classifier with GeLU activation(Hendrycks and Gimpel, 2016). Suppose that $\text{LXMERT}(img, sent)$ is the single-vector cross-modality representation, the predicted probability is:

$$\begin{aligned} x_0 &= \text{LXMERT}(img_0, s) \\ x_1 &= \text{LXMERT}(img_1, s) \\ z^0 &= W_0[x_0; x_1] + b_0 \\ z^1 &= \text{LayerNorm}\left(\text{GeLU}(z^0)\right) \\ prob &= \sigma(W_1 z^1 + b_1) \end{aligned}$$

where $\sigma$ is sigmoid function. The model is optimized by maximizing the log-likelihood, which is equivalent to minimize the binary cross entropy loss:

$$\mathcal{L} = \text{-}y^* \log prob - (1 - y^*) \log(1 - prob)$$

## C Training, Validation, and Testing Splits

We carefully split each dataset to ensure that all testing images are not involved in any pre-training or fine-tuning steps. Our data splits for each dataset and reproducible code are available at https://github.com/airsplay/lxmert.

**LXMERT Pre-Traininig** Since MS COCO has a relative large validation set, we sample a set of 5k images from the MS COCO validation set as the mini-validation (minival) set. The rest of the images in training and validation sets (i.e., COCO training images, COCO validation images besides minival, and all the other images in Visual Genome) are used in pre-training. Although the captions and questions of the MS COCO test sets are available, we exclude all of them to make sure that testing images are not seen in pre-training.

**Fine-tuning** For training and validating VQA v2.0, we take the same split convention as in our LXMERT pre-training. The data related to images in LXMERT mini-validation set is used to validate model performance and the rest of the data in train+val are used in fine-tuning. We test our model on the VQA v2.0 'test-dev' and 'test-standard' splits. For GQA fine-tuning, we follow the suggestions in official GQA guidelines[12] to take *testdev* as our validation set and fine-tune our model on the joint train + validation sets. We test our GQA model on GQA 'test-standard' split. The images in NLVR$^2$ are not from either MS COCO or Visual Genome, we thus keep using the original split: fine-tune on train split, validate the model choice on val split, and test on the public ('Test-P') and unreleased ('Test-U') test splits.

## D Training Details of 'BERT versus LXMERT'

When training with BERT only, we train each experiments for 20 epochs with a batch size 64/128 since it was not pre-trained on these cross-modality datasets. The learning rate is set to $1e-4$ instead of $5e-5$.

[12] https://cs.stanford.edu/people/dorarad/gqa/evaluate.html

# Phrase Grounding by Soft-Label Chain Conditional Random Field

**Jiacheng Liu** **Julia Hockenmaier**
University of Illinois at Urbana-Champaign, Urbana, IL, USA 61801
{jl25, juliahmr}@illinois.edu

## Abstract

The phrase grounding task aims to ground each entity mention in a given caption of an image to a corresponding region in that image. Although there are clear dependencies between how different mentions of the same caption should be grounded, previous structured prediction methods that aim to capture such dependencies need to resort to approximate inference or non-differentiable losses. In this paper, we formulate phrase grounding as a sequence labeling task where we treat candidate regions as potential labels, and use neural chain Conditional Random Fields (CRFs) to model dependencies among regions for adjacent mentions. In contrast to standard sequence labeling tasks, the phrase grounding task is defined such that there may be multiple correct candidate regions. To address this multiplicity of gold labels, we define so-called Soft-Label Chain CRFs, and present an algorithm that enables convenient end-to-end training. Our method establishes a new state-of-the-art on phrase grounding on the Flickr30k Entities dataset. Analysis shows that our model benefits both from the entity dependencies captured by the CRF and from the soft-label training regime. Our code is available at github.com/liujch1998/SoftLabelCCRF

## 1 Introduction

Given an image and a corresponding caption, the phrase grounding task aims to ground each entity mentioned by a noun phrase in the caption to a region in the image. Phrase grounding has attracted much research interest due to its application in downstream tasks including image captioning (Karpathy et al., 2014; Fang et al., 2015; Donahue et al., 2017; Xu et al., 2015), image retrieval (Chen et al., 2017a; Radenovic et al., 2016), and visual question answering (Agrawal et al., 2017; Yu et al., 2017, 2018a).




(a) Dependency between entities. The visual relationship between grounding regions for "cheerleaders" and "a girl" should agree with context "toss ... high up into the air".

(b) Gold label multiplicity. The green box is the annotated gold grounding region for entity phrase "Old man", while the orange dash boxes are region proposals with IoU $\geq 0.5$ with gold.

Figure 1: Example image-caption pairs from Flickr30k Entities, illustrating entity dependencies and gold label multiplicity.

Phrase grounding systems typically work by ranking a set of candidate regions (Chen et al., 2017b; Yu et al., 2018b). Region proposals are generated from the image by a vision backbone model, without conditioning on the caption. Features of the phrase to be grounded are extracted, and subsequently interact with features of candidate regions, to determine phrase-region compatibility. Candidate regions are then ranked based on this compatibility metric, and the highest-scored candidate region is selected as the predicted grounding of the phrase.

In Flickr30k Entities (Plummer et al., 2017b), each caption contains an average of 2.76 entity phrases to ground (Figure 2a; phrases with no corresponding gold regions are not counted). It therefore stands to reason that phrases in the same caption should not be grounded independently (to op-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5112–5122, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

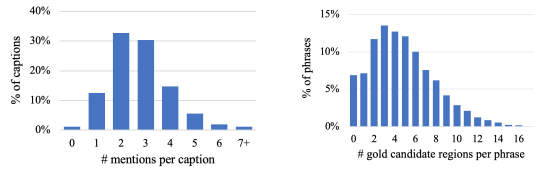


(a) Distribution of number of entity phrases per caption.

(b) Distribution of number of gold labels per entity phrase.

Figure 2: Validation set statistics for Flickr30k Entities.

timize each individual phrase-region assignment), but jointly (to optimize the global phrase-region assignment for the entire caption). Figure 1a illustrates this phenomenon. The caption contains a sequence of two entity phrases, "cheerleaders" and "a girl", and the task is to label each phrase with a candidate region that best grounds it. Since there are several women present in the image, "a girl" has ambiguous grounding by itself, but it can be disambiguated by encouraging the visual relationship between "a girl" and "cheerleaders" to conform with context provided in the caption.

Some works are aware that dependencies between entities in the same caption play an important role in building more accurate phrase grounding systems (Wang et al., 2016; Plummer et al., 2017a; Chen et al., 2017b). The success of these structured prediction methods shows the advantage of considering entity dependencies in learning and prediction. However, these approaches capture certain relations in an *ad hoc* manner, and resort to approximate inference (Wang et al., 2016; Plummer et al., 2017a) or non-differentiable losses (Chen et al., 2017b).

To obtain models and inference algorithms that facilitate more globally consistent phrase grounding predictions, we propose to formulate phrase grounding as a sequence labeling task where we treat candidate regions as potential labels for the phrases in the input sequence. This allows us to build phrase grounding models based on Conditional Random Fields (CRFs) (Lafferty et al., 2001) that capture entity dependencies in a universal and differentiable manner. Our results indicate that systems that capture dependencies between phrases in the same caption in a principled manner outperform systems that ignore these dependencies.

A second problem lies in the use of region proposals, which distinguishes phrase grounding from other sequence labeling tasks where CRFs are directly applicable. Following the metrics of object detection, in phrase grounding the correctness of a predicted region is judged by its overlap by Intersection-over-Union (IoU) with the gold region (Plummer et al., 2017b). To cover potential regions with high enough IoU, it is common to generate a myriad of region proposals and for these candidate regions to contain or substantially overlap with each other. As a result, there could be more than one candidate region with high IoU with the gold region, and they should all be considered as correct grounding for the phrase. This phenomenon of gold label multiplicity is illustrated in Figure 1b. We hypothesize that it is important to consider gold label multiplicity and identify all correct region proposals during training, since the model would receive contradictory training signals if some correct proposals were marked as incorrect. With region proposals generated by a Bottom-Up Attention (Anderson et al., 2018) visual backbone, in Flickr30k Entities each phrase has an average of 4.75 gold labels, and detailed statistics are presented in Figure 2b. To address this problem, we adopt the soft-label target distribution proposed by Yu et al. (2018b), and our experiments show that models trained with this regime significantly outperform those trained with one-hot target regime.

To combine the benefits brought by structured prediction from CRFs and by soft-label training regime, we define Soft-Label Chain CRFs, a variation of standard chain CRFs that allows us to work with gold label multiplicity. We adapt learning and inference algorithms from chain CRFs and develop an end-to-end training algorithm for our proposed model.

We evaluate the effectiveness of Soft-Label Chain CRF on phrase grounding by conducting experiments on the Flickr30k Entities dataset (Plummer et al., 2017b) and comparing grounding accuracy with strong baseline models, as well as with existing structured prediction methods and current state-of-the-art models. Experimental results show that our Soft-Label Chain CRF model outperforms its hard-label CRF counterpart by 2.43%, a vanilla non-CRF soft-label model by 0.40%, and the previous best results by about 1.4%, demonstrating that both of our contributions, modeling phrase grounding as a sequence labeling task, and training with soft label targets, matter for this task.

## 2 Related Work

**Phrase Grounding.** The phrase grounding task was first postulated by Karpathy and Fei-Fei (2017) and Plummer et al. (2017b), both of which moved from the holistic image captioning to the finer-grained task of matching regions with phrases in the caption. Datasets for this task include Flickr30k Entities (Plummer et al., 2017b), RefCOCO (Yu et al., 2016), and Visual Genome (Krishna et al., 2017). The general framework of proposal-generation-ranking has become adopted by most approaches to phrase grounding, and research in this area has focused on improving specific components of this framework. Our work can be viewed as an improvement to the training and prediction aspects.

**Structured Prediction in Phrase Grounding.** We summarize some works that consider entity dependencies by structured prediction. Structured Matching (Wang et al., 2016) formulates phrase grounding as a bipartite matching process between phrases and candidate regions, and encourages the spatial relationship between two grounding regions to conform to an extracted *partial coreference* relation between their corresponding phrases. The resulting discrete optimization problem is then relaxed into a linear program to enable end-to-end training. Phrase-Region CCA (Plummer et al., 2017a) mines *frequent patterns* of semantically related paired phrases and trains a separate model for each pattern. The addition of this pairwise score makes the optimization a quadratic programming problem that requires approximate inference. QRC Net (Chen et al., 2017b) assumes that phrases in a caption refer to distinct entities, and thus predicted grounding regions are penalized for *spatial overlapping*. However, overlapping regions can be penalized only after prediction, so this loss is not differentiable, and one has to resort to reinforcement learning. In these works, partial coreference extraction, frequent patterns mining and spatial overlap penalties are *ad hoc* entity dependency capturing, while we aim to universally encompass the spectrum of such dependencies.

**Soft-Label Training Regime.** Conventionally, region proposal ranking is done by predicting a probability distribution over all candidate regions for grounding a given entity phrase, which is learned to match a target distribution. Chen et al. (2017b) and Rohrbach et al. (2016) define the target distribution as a one-hot vector which only gives credit to the candidate region with highest IoU with the gold region, and cross-entropy loss is used as training objective. Under this hard-label training regime, the model is trained to pick only the best candidate region while rejecting all the inferior-than-best candidate regions, which is intuitively not a good behavior. Yu et al. (2018b) proposes a soft-label target distribution which gives weighted credit to all good candidate regions (i.e. those with above-threshold IoU with the gold region), and uses Kullback-Leibler (KL) divergence loss as training objective.

**Conditional Random Fields.** CRFs (Lafferty et al., 2001) are discriminative probabilistic models that have been found useful in sequence labeling tasks by capturing label dependencies (Ma and Hovy, 2016; Lample et al., 2016). We summarize some works relevant to CRFs learned in soft-label or multi-label settings. Multi-CRFs (Dredze et al., 2009) learn CRFs with noisy annotated data, where annotators may disagree on the label for input tokens. The assumption is that there is always only one gold label for each token, so the model favors single label while conforming to the prior distribution of labels set by annotators. To work with soft-label targets, it employs a mode-seeking, exclusive KL divergence definition, which does not imply moment-matching, a desired property of CRFs (and in general, exponential family models) that we show in Section 3.1 and 3.2 for the mean-seeking, inclusive KL divergence definition in our model. Rodrigues et al. (2014) models the latent reliability of individual annotators, and use this information to guide the selection of trustworthy annotation sources and estimation of real gold labels. Note that both works always assume one gold label per input token, where the ambiguity comes from unreliability of annotations, while our work focuses on cases where there may be multiple gold labels per input token by the nature of the task.

## 3 Soft-Label Chain CRF

CRFs model the probability of a label sequence $\boldsymbol{y} = y^{1:T}$ conditioned on an input sequence $\boldsymbol{x} = x^{1:T}$ in terms of a score function $s(\boldsymbol{x}, \boldsymbol{y})$:

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{\exp s(\boldsymbol{y}, \boldsymbol{x})}{\sum_{\boldsymbol{y}'} \exp s(\boldsymbol{y}', \boldsymbol{x})}$$

For a given training example $\{(\boldsymbol{x}, \boldsymbol{y})\}$, the negative log-likelihood loss (i.e. cross-entropy loss

w.r.t. a one-hot target distribution that gives credit to the gold label only) is

$$L = -\log p(\boldsymbol{y}|\boldsymbol{x}) = -s(\boldsymbol{y}, \boldsymbol{x}) + \log Z(\boldsymbol{x})$$

where $Z(\boldsymbol{x}) = \sum_{\boldsymbol{y}'} \exp s(\boldsymbol{y}', \boldsymbol{x})$. The gradient of this loss w.r.t. score function is

$$\frac{\partial L}{\partial s(\boldsymbol{y}', \boldsymbol{x})} = -\mathbb{I}(\boldsymbol{y}' = \boldsymbol{y}) + p(\boldsymbol{y}'|\boldsymbol{x})$$

which is known as moment-matching. This allows us to train CRFs with gradient methods and conveniently connect to backpropagation when the score function is modeled by a neural architecture.

### 3.1 Soft-Label CRF

In the standard CRF above, each input $x^t$ corresponds to a single gold label $y^t$. To account for gold label multiplicity in training stage, we replace the sequence of gold labels $\boldsymbol{y}$ with a sequence of distributions $\boldsymbol{q} = q^{1:T}$ where $q^t \in \mathbb{R}^K$ is the gold label distribution over all $K$ possible labels for input $x^t$. Note that this distribution should not be interpreted as the confidence of each label being correct; rather, it should be understood as a probabilistic gold label model: if we randomly choose a gold label, how likely is each label to be selected. With independence assumption, the gold probability of an arbitrary label sequence $\boldsymbol{y}$ is

$$q(\boldsymbol{y}|\boldsymbol{x}) = \prod_t q(y^t|\boldsymbol{x}) = \prod_t q(y^t|x^t) = \prod_t q^t_{y^t}$$

It is easy to see that $q(\boldsymbol{y}|\boldsymbol{x})$ is a distribution:

$$\sum_{\boldsymbol{y}} q(\boldsymbol{y}|\boldsymbol{x}) = \sum_{\boldsymbol{y}} \prod_t q^t_{y^t} = \prod_t \sum_{y^t} q^t_{y^t} = 1$$

And our goal is to learn this target distribution.

Since this target distribution is no longer degenerate, we use Kullback-Leibler (KL) divergence to measure the discrepancy between the model and the target distribution. Our training objective is the KL divergence loss (in mean-seeking, inclusive form):

$$L = \sum_{\boldsymbol{y}} \Big\{ q(\boldsymbol{y}|\boldsymbol{x}) \log \frac{q(\boldsymbol{y}|\boldsymbol{x})}{p(\boldsymbol{y}|\boldsymbol{x})} \Big\}$$

which also gives gradients that demonstrate moment-matching:

$$\frac{\partial L}{\partial s(\boldsymbol{y}', \boldsymbol{x})} = -q(\boldsymbol{y}'|\boldsymbol{x}) + p(\boldsymbol{y}'|\boldsymbol{x})$$

Note that if we had defined the KL divergence loss in its mode-seeking, exclusive form $\sum_{\boldsymbol{y}} p(\boldsymbol{y}|\boldsymbol{x}) \log \frac{p(\boldsymbol{y}|\boldsymbol{x})}{q(\boldsymbol{y}|\boldsymbol{x})}$, we would have lost this desired moment-matching property.

### 3.2 Factorization of Soft-Label Chain CRF

Learning CRFs of general graphs requires inference in unit of cliques, which is usually computationally intractable. By restricting to local, pairwise potentials, we reduce the model to a first-order linear chain CRF, whose scoring function factorizes as

$$\begin{aligned} s(\boldsymbol{y}, \boldsymbol{x}) &= \sum_t s(y^t, y^{t-1}, \boldsymbol{x}) \\ &= \sum_t \Big\{ \tau(y^t, y^{t-1}, \boldsymbol{x}) + \varepsilon(y^t, \boldsymbol{x}) \Big\} \end{aligned}$$

where $\tau(\cdot, \cdot, \cdot)$ is the transition score between labels at $t-1$ and $t$ that captures the dependency between labels for adjacent input tokens, and $\varepsilon(\cdot, \cdot)$ is the emission score between label and input at $t$.

Combining this factorization with soft-label targets gives the formal definition of Soft-Label Chain CRF. The loss can be written as

$$\begin{aligned} L &= \sum_{\boldsymbol{y}} \Big\{ q(\boldsymbol{y}|\boldsymbol{x}) \log \frac{q(\boldsymbol{y}|\boldsymbol{x})}{p(\boldsymbol{y}|\boldsymbol{x})} \Big\} \\ &= \sum_{\boldsymbol{y}} \Big\{ q(\boldsymbol{y}|\boldsymbol{x}) \big[ \log q(\boldsymbol{y}|\boldsymbol{x}) - s(\boldsymbol{y},\boldsymbol{x}) + \log Z(\boldsymbol{x}) \big] \Big\} \\ &\qquad \text{(Expand } p(\boldsymbol{y}|\boldsymbol{x}) \text{ by CRF modeling)} \\ &= \sum_{\boldsymbol{y}} \Big\{ q(\boldsymbol{y}|\boldsymbol{x}) \big[ \log q(\boldsymbol{y}|\boldsymbol{x}) - s(\boldsymbol{y},\boldsymbol{x}) \big] \Big\} + \log Z(\boldsymbol{x}) \\ &\qquad \text{(Marginalize } q(\boldsymbol{y}|\boldsymbol{x})\text{)} \\ &= \sum_t \sum_{y^t} \Big\{ q(y^t|\boldsymbol{x}) \log q(y^t|\boldsymbol{x}) \Big\} \\ &\quad - \sum_{\boldsymbol{y}} \Big\{ q(\boldsymbol{y}|\boldsymbol{x}) s(\boldsymbol{y},\boldsymbol{x}) \Big\} + \log Z(\boldsymbol{x}) \\ &\qquad \text{(Independence of } q(y^t|\boldsymbol{x}) \text{ across } t\text{)} \\ &= \sum_t \sum_{y^t} \Big\{ q(y^t|\boldsymbol{x}) \log q(y^t|\boldsymbol{x}) \Big\} \\ &\quad - \sum_{\boldsymbol{y}} \Big\{ q(\boldsymbol{y}|\boldsymbol{x}) \sum_t s(y^t,y^{t-1},\boldsymbol{x}) \Big\} + \log Z(\boldsymbol{x}) \\ &\qquad \text{(Factorization of } s(\boldsymbol{y},\boldsymbol{x})\text{)} \\ &= \sum_t \sum_{y^t} \Big\{ q(y^t|\boldsymbol{x}) \log q(y^t|\boldsymbol{x}) \Big\} \\ &\quad - \sum_t \sum_{y^t y^{t-1}} \Big\{ s(y^t,y^{t-1},\boldsymbol{x}) \sum_{\boldsymbol{y}|y^t y^{t-1}} q(\boldsymbol{y}|\boldsymbol{x}) \Big\} \\ &\quad + \log Z(\boldsymbol{x}) \\ &\qquad \text{(Reorganize sums by } s(y^t,y^{t-1},\boldsymbol{x})\text{)} \\ &= \sum_t \sum_{y^t} \Big\{ q(y^t|\boldsymbol{x}) \log q(y^t|\boldsymbol{x}) \Big\} \end{aligned}$$

**Algorithm 1** Modified forward algorithm to compute the KL divergence loss for Soft-Label Chain CRFs

**procedure** SOFTLABELCHAINCRFLOSS($\boldsymbol{q}, \varepsilon(y^t, \boldsymbol{x}), \tau(y^t, y^{t-1}, \boldsymbol{x})$)
  **for all** label $y^0$ **do**
    $\alpha^0_{y^0} \leftarrow 0$
    $g^0_{y^0} \leftarrow 0$
  **for** $t = 1 \dots T$ **do**
    **for all** label $y^t$ **do**
      $\alpha^t_{y^t} \leftarrow \sum_{y^{t-1}} \left\{ \alpha^{t-1}_{y^{t-1}} \exp\left[\tau(y^t, y^{t-1}, \boldsymbol{x}) + \varepsilon(y^t, \boldsymbol{x})\right] \right\}$
      $g^t_{y^t} \leftarrow \sum_{y^{t-1}} \left\{ \left[g^{t-1}_{y^{t-1}} + \tau(y^t, y^{t-1}, \boldsymbol{x})\right] q^{t-1}_{y^{t-1}} + \left[\varepsilon(y^t, \boldsymbol{x}) - \log q^t_{y^t}\right] \right\}$
  $Z \leftarrow \sum_{y^T} \alpha^T_{y^T}$
  $G \leftarrow \sum_{y^T} g^T_{y^T} q^T_{y^T}$
  $L \leftarrow -G + \log Z$
  **return** $L$

$$-\sum_t \sum_{y^t y^{t-1}} \left\{ q(y^t, y^{t-1}|\boldsymbol{x}) s(y^t, y^{t-1}, \boldsymbol{x}) \right\} + \log Z(\boldsymbol{x})$$

which gives moment-matching gradients

$$\frac{\partial L}{\partial s(y^t, y^{t-1}, \boldsymbol{x})} = -q(y^t, y^{t-1}|\boldsymbol{x}) + p(y^t, y^{t-1}|\boldsymbol{x})$$
$$\frac{\partial L}{\partial \tau(y^t, y^{t-1}, \boldsymbol{x})} = -q(y^t, y^{t-1}|\boldsymbol{x}) + p(y^t, y^{t-1}|\boldsymbol{x})$$
$$\frac{\partial L}{\partial \varepsilon(y^t, \boldsymbol{x})} = -q(y^t|\boldsymbol{x}) + p(y^t|\boldsymbol{x})$$

where

$$q(y^t|\boldsymbol{x}) = q^t_{y_t}$$
$$q(y^t, y^{t-1}|\boldsymbol{x}) = q^t_{y^t} q^{t-1}_{y^{t-1}}$$

are the probability of local label(s) marginalized over all possible non-local labels. Smoothing inference $p(y^t|\boldsymbol{x})$ and $p(y^t, y^{t-1}|\boldsymbol{x})$ can be computed with forward-backward algorithm.

### 3.3 As an Extension of Soft-Label Model

Note that if we omit all transition terms in Soft-Label Chain CRF, the loss reduces to

$$L' = \sum_t \sum_{y^t} \left\{ q(y^t|\boldsymbol{x}) \left[-\varepsilon(y^t, \boldsymbol{x}) + \log q(y^t|\boldsymbol{x})\right] \right\} + \log Z(\boldsymbol{x})$$
$$= \sum_t \sum_{y^t} \left\{ q(y^t|\boldsymbol{x}) \log \frac{q(y^t|\boldsymbol{x})}{p(y^t|\boldsymbol{x})} \right\}$$

which is a total factorization over time. This is as if each label is predicted independently using a soft-label training regime, which is exactly the KL divergence loss proposed by Yu et al. (2018b). Therefore, our Soft-Label Chain CRF can be viewed as an extension of this soft-label discriminative model.

### 3.4 Modified Forward Algorithm

For chain CRFs, computing the loss only requires forward algorithm, while computing the gradients requires a full forward-backward algorithm. It can be proved that backpropagation on the loss gives the same result as running forward-backward. This is a commonly used trick in modern deep learning frameworks to eliminate the need of implementing the backward pass. Algorithm 1 presents a modified forward algorithm that computes the loss for Soft-Label Chain CRF. In Section 1 and 2 of the Supplementary Materials, we prove the correctness of this algorithm, and that its backpropagation is also equivalent to forward-backward.

## 4 Phrase Grounding as Sequence Labeling

### 4.1 Task Formulation

We formulate phrase grounding as a sequence labeling task. Given an image $I$, a caption sentence $[c^1 \dots c^L]$ where $c^l$ is a word token, and a set of non-overlapping noun phrase spans $[p^1 \dots p^T]$ where $p^t = (s^t, e^t)$ denotes that the $t$'th phrase covers tokens $c^{s^t}$ to $c^{e^t}$ (inclusive), we generate a set of region proposals $\{r_1 \dots r_K\}$, label each phrase with a candidate region, and refine the region by performing a bounding box regression.

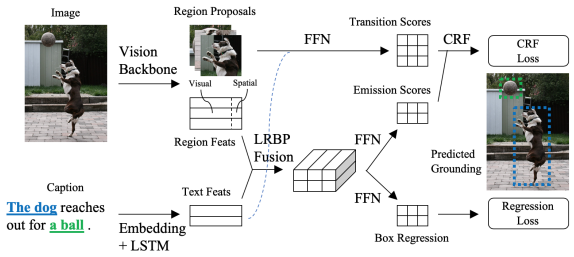


Figure 3: Our model for phrase grounding as a sequence labeling task. The $K \times K$ transition score matrix is derived from the features of $K$ region proposals. The $T \times K$ emission score matrix is derived from a joint representation of phrase-region pairs, which is fused from features of region proposals and $T$ entity phrases. Bounding box regression is applied to the sequence of regions predicted by the CRF. Cyan dashed line: contextualized transition score prediction (Section 4.2).

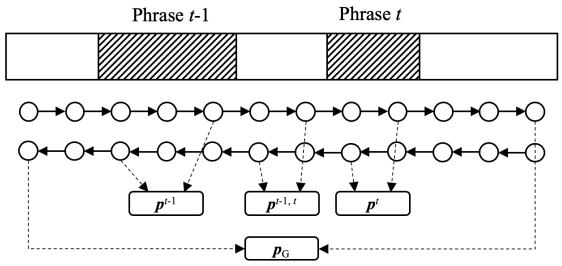


Figure 4: Text feature extraction for phrases in a caption. Shaded regions are entity phrase spans; circles represent LSTM cells. For phrase $t$ hidden states at its span boundaries are concatenated to form its text features $\boldsymbol{p}^t$, which is used in fusion with region features. For the contextualized transition score between phrases $t-1$ and $t$, hidden states at the boundaries of the context between them are concatenated into a context feature vector $\boldsymbol{p}^{t-1,t}$, which can be further extended by phrase features $\boldsymbol{p}^{t-1}$ and $\boldsymbol{p}^t$ as well as global text features $\boldsymbol{p}_G$.

### 4.2 Model Specification

Figure 3 outlines our phrase grounding model. $K$ region proposals and their visual and spatial features are extracted from an object detection vision backbone. We feed the token embeddings of the caption into a bi-directional LSTM (Hochreiter and Schmidhuber, 1997), and then concatenate the forward hidden state at the ending boundary of the phrase with the backward hidden state at the starting boundary of the phrase (see Figure 4). This phrase representation captures context both preceding and following the phrase in the caption.

$$(\overrightarrow{\boldsymbol{h}^{1:L}}, \overleftarrow{\boldsymbol{h}^{1:L}}) = \text{BiLSTM}(\text{Embed}([c^1 \dots c^L]))$$
$$\boldsymbol{p}^t = [\overrightarrow{\boldsymbol{h}^{e^t}} || \overleftarrow{\boldsymbol{h}^{s^t}}]$$

We use low-rank bilinear pooling (LRBP) (Kim et al., 2017) to fuse text and region features. Compared to simple concatenation, LRBP supports pairwise interaction between bimodal feature channels while keeping a reasonable computation overhead. Given a text feature vector $\boldsymbol{p}^t \in \mathbb{R}^{d_{\text{text}}}$ and a region feature vector $\boldsymbol{r}_k \in \mathbb{R}^{d_{\text{vis}}}$, LRBP fuses them into a joint representation $\boldsymbol{f}_k^t \in \mathbb{R}^{d_{\text{joint}}}$:

$$\boldsymbol{f}_k^t = \boldsymbol{P}^\top (\boldsymbol{U}^\top \boldsymbol{p}^t \circ \boldsymbol{V}^\top \boldsymbol{r}_k) + \boldsymbol{b}$$

where $\boldsymbol{U} \in \mathbb{R}^{d_{\text{text}} \times r}$, $\boldsymbol{V} \in \mathbb{R}^{d_{\text{vis}} \times r}$, pooling matrix $\boldsymbol{P} \in \mathbb{R}^{r \times d_{\text{joint}}}$, bias $\boldsymbol{b} \in \mathbb{R}^{d_{\text{joint}}}$, and $\circ$ is the Hadamard (i.e. element-wise) product.

As discussed in Section 3.2, the CRF score function consists of emission score and transition score. The emission score $\varepsilon(r_k, p^t)$ models the compatibility between each phrase and each candidate region. We feed the joint representation to a single-layer feed-forward neural network:

$$\varepsilon(r_k, p^t) = \text{FFN}(\boldsymbol{f}_k^t)$$

The transition score $\tau(r_k, r_{k'}, p^{1:T})$ is modeled by a two-layer feed-forward neural network with ReLU activation for the hidden layer:

$$\tau(r_k, r_{k'}, p^{1:T}) = \text{FFN}(\sigma(\text{FFN}([\boldsymbol{r}_k || \boldsymbol{r}_{k'}])))$$

To condition the transition scores on local and global context from the caption, we can extend the input $[\boldsymbol{r}_k || \boldsymbol{r}_{k'}]$ with the following text features: context in between the two phrases (feature vector $\boldsymbol{p}^{t-1,t}$), context from phrase features $\boldsymbol{p}^{t-1}$ and $\boldsymbol{p}^t$, and global context $\boldsymbol{p}_G$.

One important difference between the standard use of CRFs for sequence labeling and our task is that our "labels" do not correspond to a fixed set of classes that can be predicted for any input, but are as specific to the particular input example as the sequences to be labeled themselves. Hence, our transition and emission scores do not depend on the (arbitrary) indices of regions to be ground, but on their visual and spatial features (as well as on their corresponding linguistic contexts). Finally, although our approach could in principle be extended to higher-order CRFs, we restrict our attention here to first-order CRFs for computational efficiency. As a consequence, our models can only capture dependencies between string-adjacent phrases.

### 4.3 Training Objectives

For each image-caption instance, the loss is a linear combination of the labeling and bounding box regression loss:

$$L = L_{\text{label}} + \gamma L_{\text{reg}}$$

$L_{\text{label}}$ is the CRF loss defined in Section 3.2. $L_{\text{reg}}$ (Ren et al., 2017) is defined as

$$L_{\text{reg}} = (\beta, \hat{\beta}) \quad = \sum_{i \in \{x,y,w,h\}} \text{SmoothL1}(\hat{\beta}_i - \beta_i)$$

with the ground truth regression parameterization

$$\beta = [\frac{x - x_a}{w_a}, \frac{y - y_a}{h_a}, \log \frac{w}{w_a}, \log \frac{h}{h_a}]$$

and

$$\text{SmoothL1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

## 5 Experiments

### 5.1 Experiment Setup

**Dataset.** We train and evaluate our models on the Flickr30k Entities dataset (Plummer et al., 2017b), which contains $31,783$ images, each accompanied by $5$ captions. In keeping with previous work on this dataset, we assume that entity phrase boundaries are given, so inferring which phrases to ground is not part of our task. Following Plummer et al. (2017b), we merge all regions that are ground to the same phrase into one larger bounding box, and split the dataset into $29,783$ training images, $1k$ validation images and $1k$ test images.

We do not apply our method to RefCOCO (Yu et al., 2016) or Visual Genome (Krishna et al., 2017) because they consist of independently grounded entity phrases without any entity dependencies that CRFs could leverage.

**Implementation details.** For text feature extraction, we use the 1024-d contextualized word embeddings from the last layer of ELMo (Peters et al., 2018), followed by a bi-directional LSTM (Hochreiter and Schmidhuber, 1997) encoder with hidden dimension $d_{\text{hidden}} = 512$ for each direction, so that the text feature vector has dimension $d_{\text{text}} = 1024$. We use the Bottom-Up Attention model (Anderson et al., 2018) to generate region proposals and extract visual features, as in the state-of-the-art BAN (Kim et al., 2018) and DDPN (Yu et al., 2018b) models. $K = 100$ region proposals are generated for each image. Each candidate region with coordinates $(x_{\text{min}}, y_{\text{min}})$, $(x_{\text{max}}, y_{\text{max}})$ is represented by a $d_{\text{vis}} = 2053$ feature vector that consists of 2048-d visual features concatenated with 5-d spatial features $[x_{\text{min}}/W, y_{\text{min}}/H, x_{\text{max}}/W, y_{\text{max}}/H, wh/WH]$. The low-rank bilinear pooling (LRBP) layer used for text-region bimodal feature fusion has rank $r = 1024$ and output dimension $d_{\text{joint}} = 1024$. We train with a mini-batch size of 16 image-caption instances. Each instance contains all entity phrases to be grounded in the caption. Weights are initialized with Xavier (Glorot and Bengio, 2010). We apply a dropout rate of $p = 0.2$ after the word embedding layer, LSTM layer, and LRBP fusion layer. The loss weighting parameter $\gamma$ is $10.0$. All gradients are clipped by $\infty$-norm of $10.0$ to prevent gradient explosion. We do not fine-tune ELMo or the Bottom-Up Attention model. All models are trained for $50k$ iterations using Adam (Kingma and Ba, 2015) with learning rate $5e-5$ and $\beta_1 = 0.9, \beta_2 = 0.98$. Model snapshots are taken every $5k$ iterations and the model with the highest validation set accuracy is selected.

**Metrics.** We predict one grounded region for each entity phrase. Following Plummer et al. (2017b), a prediction is deemed accurate if it has at least 0.5 IoU overlap with the gold region. We report the percentage of accurately grounded phrases.

### 5.2 Quantitative Results

We compare our Soft-Label Chain CRF model against three baselines: a Hard-Label non-CRF model, a Hard-Label CRF, and a Soft-Label non-CRF model. The non-CRF models ground each phrase independently with a loglinear model. The Hard-Label models are trained with a standard one-hot training regime. The Soft-Label models use the soft-label training regime described above. The Soft-Label non-CRF model corresponds to the reduced form of the Soft-Label Chain CRF in Section 3.3.

Table 1 shows the performance of previous structured prediction models, current state-of-the-art models, our baseline models and the Soft-Label Chain CRF model. For a fair comparison with BAN (Kim et al., 2018), we also report result of the hard-label baseline with GloVe (Pennington et al., 2014) embeddings, while we obtain 0.33% higher result with ELMo. Training a non-CRF model on soft-label target distributions

| Method | Vision Backbone | Grounding Accuracy (%) |
|---|---|---|
| *Compared methods* | | |
| Structured Matching (Wang et al., 2016) | Fast R-CNN (Girshick, 2015) | 42.08 |
| Phrase-Region CCA (Plummer et al., 2017a) | Fast R-CNN (Girshick, 2015) | 55.85 |
| QRC Net (Chen et al., 2017b) | Fast R-CNN (Girshick, 2015) | 65.14 |
| BAN (Kim et al., 2018) | Bottom-Up Attention (Anderson et al., 2018) | 69.69 |
| DDPN (Yu et al., 2018b) | Bottom-Up Attention (Anderson et al., 2018) | 73.3 |
| *Our methods* | | |
| Hard-Label (GloVe (Pennington et al., 2014)) | Bottom-Up Attention (Anderson et al., 2018) | 71.88 |
| Hard-Label (HL) | Bottom-Up Attention (Anderson et al., 2018) | 72.21 |
| Soft-Label (SL) | Bottom-Up Attention (Anderson et al., 2018) | 74.29 |
| Hard-Label Chain CRF (HL-CCRF) | Bottom-Up Attention (Anderson et al., 2018) | 72.26 |
| Soft-Label Chain CRF (SL-CCRF) | Bottom-Up Attention (Anderson et al., 2018) | **74.69** |

Table 1: Performance of different phrase grounding methods on Flickr30k Entities (test set). Our CRF models has transition scores conditioned on features of context in between the two phrases ("M" in Table 2). Our methods, unless explicitly specified, uses ELMo (Peters et al., 2018) as word embeddings.

| Model | Transition Context | Accuracy (%) |
|---|---|---|
| SL-CCRF | – | 74.28 |
| SL-CCRF | M | **74.69** |
| SL-CCRF | M+LR | 74.45 |
| SL-CCRF | M+LR+G | 74.48 |

Table 2: Performance of Soft-Label Chain CRF models by conditioning transition scores on different sets of context features. –: input to transition score prediction is $[\boldsymbol{r}_k||\boldsymbol{r}_{k'}]$. M: input extended by features of context $\boldsymbol{p}^{t-1,t}$ in between the two phrases. M+LR: input further extended by features of LHS phrase $\boldsymbol{p}^{t-1}$ and RHS phrase $\boldsymbol{p}^{t}$. M+LR+G: input further extended by features of global context $\boldsymbol{p}_G$.

| Decoding Algorithm | HL-CCRF | SL-CCRF |
|---|---|---|
| Viterbi (MAP) | 72.26 | 74.69 |
| Smoothing | **72.30** | **74.73** |

Table 3: Decoding algorithms' impact on performance.

improves accuracy by a further 2.08%. On top of that, Soft-Label Chain CRF improves accuracy by another 0.40%, which shows the effectiveness of treating phrase grounding as a sequence labeling task and using CRFs to capture entity dependencies. We also observe that the Hard-Label Chain CRF outperforms the hard-label baseline by a mere margin of 0.05%, so our conjecture is that using chain CRFs works well only with a suitable choice of training regime. Soft-Label Chain CRF gives an overall improvement of 2.48% over the hard-label baseline; it significantly outperforms previous structured prediction models including Structured Matching (Wang et al., 2016), Phrase-Region CCA (Plummer et al., 2017a) and QRC Net (Chen et al., 2017b), and surpasses the state-of-the-art BAN (Kim et al., 2018) and DDPN (Yu et al., 2018b) models by a margin of 5.00% and about 1.4%, respectively.

We conduct an ablation study to find the most appropriate combination of context features for the transition scores in the SL-CCRF model. Table 2 shows that we obtain the best results by including the context in between the two phrases, which gives an improvement of 0.41%. We did not see any benefit from adding further text features from the left and right side of the phrases, or from the entire caption.

Besides the Viterbi decoding algorithm used in prediction in CRFs, we also experiment with a smoothing decoding algorithm. While Viterbi finds the MAP label sequence conditioned on the input sequence $\arg\max_{\boldsymbol{y}} p(\boldsymbol{y}|\boldsymbol{x})$, smoothing decoding finds the best label for each input $x^t$: $\arg\max_{y^t} p(y^t|\boldsymbol{x})$. This makes sense in some scenarios where we want to refine the predicted grounding of one entity by referring to the context instead of attempting to ground all entities mentioned in the description. Table 3 shows that in both Hard-Label Chain CRF and Soft-Label Chain CRF, smoothing decoding gives a prediction accuracy 0.04% higher than Viterbi decoding.

Without bounding box regression, the Soft-Label Chain CRF model has an accuracy of 69.85%, a 4.84% reduction compared to the setting with bounding box regression.

### 5.3 Qualitative Results

We visualize some phrase grounding results in the validation set of Flickr30k Entities in Figure 5. In (a), our CRF model avoids the error in grounding "a lounge chair" by constraining its relative posi-

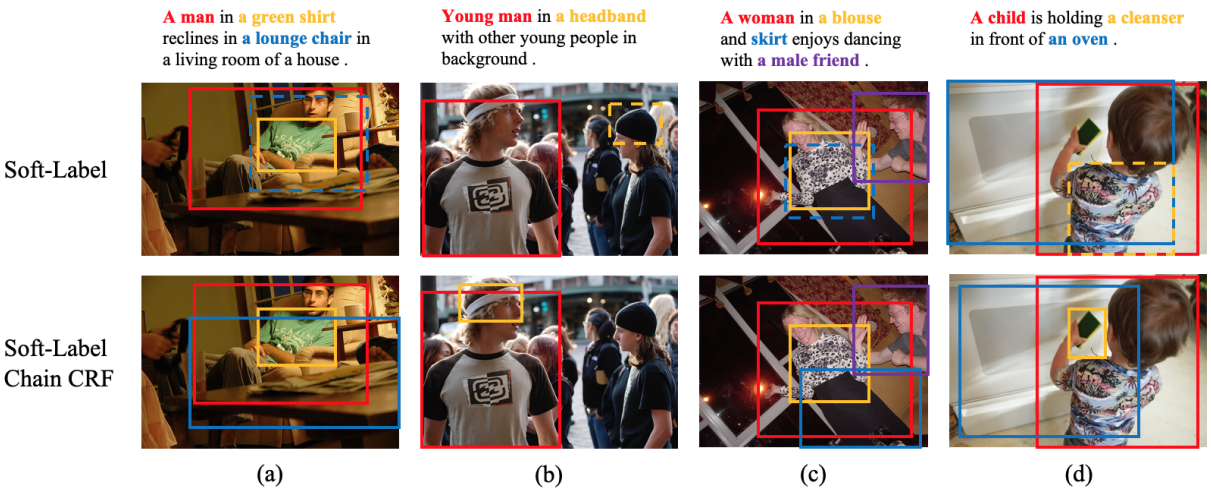


Figure 5: Selected visualization of phrase grounding results in the validation set of Flickr30k Entities. Solid boxes are correct predicted groundings, while dashed boxes are incorrect predicted groundings. Gold regions are not shown. Each entity phrase and its predicted grounding are marked with same color. Best viewed in color.

tion to "a man". In (b), although it may not have learned to distinguish "headband" and "hat", the CRF constrains the spatial position of "headband" to agree with the ownership dependency provided in context. In (c), it avoids the error in grounding "skirt" by spatially discriminating it from "a blouse". In (d), it avoids the error in grounding "a cleanser" by constraining its relative size w.r.t. "a child". These examples indicate that the CRF model may avoid grounding errors made by non-CRF models by leveraging entity dependencies, including relative position, spatial overlapping, and relative size.

## 6 Conclusion

In this paper, we formulate phrase grounding as a sequence labeling task and propose the Soft-Label Chain CRF model that successfully combines the benefits brought by global structured prediction and soft-label training regime that addresses the gold label multiplicity problem. Experimental results show that we achieve an overall improvement of 2.48% on grounding accuracy compared to a strong baseline, and that our model outperforms previous methods on phrase grounding.

## 7 Acknowledgements

This material is based upon work supported by the National Science Foundation under Grants No. 1405883 and 1563727. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

## References

Aishwarya Agrawal, Jiasen Lu, Stanislaw Antol, Margaret Mitchell, C. Lawrence Zitnick, Devi Parikh, and Dhruv Batra. 2017. VQA: visual question answering - www.visualqa.org. *International Journal of Computer Vision*, 123(1):4–31.

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and visual question answering. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 6077–6086.

Kan Chen, Trung Bui, Chen Fang, Zhaowen Wang, and Ram Nevatia. 2017a. AMC: attention guided multimodal correlation learning for image search. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 6203–6211.

Kan Chen, Rama Kovvuri, and Ram Nevatia. 2017b. Query-guided regression network with context policy for phrase grounding. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 824–832.

Jeff Donahue, Lisa Anne Hendricks, Marcus Rohrbach, Subhashini Venugopalan, Sergio Guadarrama, Kate Saenko, and Trevor Darrell. 2017. Long-term recurrent convolutional networks for visual recognition and description. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(4):677–691.

Mark Dredze, Partha Pratim Talukdar, and Koby Crammer. 2009. Sequence learning from data with mul-

tiple labels. In *ECMLPKDD 2009 workshop on learning from multi-label data*, page 39.

Hao Fang, Saurabh Gupta, Forrest N. Iandola, Rupesh Kumar Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C. Platt, C. Lawrence Zitnick, and Geoffrey Zweig. 2015. From captions to visual concepts and back. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 1473–1482.

Ross B. Girshick. 2015. Fast R-CNN. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 1440–1448.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, pages 249–256.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Andrej Karpathy and Li Fei-Fei. 2017. Deep visual-semantic alignments for generating image descriptions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(4):664–676.

Andrej Karpathy, Armand Joulin, and Fei-Fei Li. 2014. Deep fragment embeddings for bidirectional image sentence mapping. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 1889–1897.

Jin-Hwa Kim, Jaehyun Jun, and Byoung-Tak Zhang. 2018. Bilinear attention networks. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pages 1571–1581.

Jin-Hwa Kim, Kyoung Woon On, Woosang Lim, Jeonghee Kim, Jung-Woo Ha, and Byoung-Tak Zhang. 2017. Hadamard product for low-rank bilinear pooling. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123(1):32–73.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001*, pages 282–289.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 260–270.

Xuezhe Ma and Eduard H. Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 2227–2237.

Bryan A. Plummer, Arun Mallya, Christopher M. Cervantes, Julia Hockenmaier, and Svetlana Lazebnik. 2017a. Phrase localization and visual relationship detection with comprehensive image-language cues. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 1946–1955.

Bryan A. Plummer, Liwei Wang, Chris M. Cervantes, Juan C. Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. 2017b. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. *International Journal of Computer Vision*, 123(1):74–93.

Filip Radenovic, Giorgos Tolias, and Ondrej Chum. 2016. CNN image retrieval learns from bow: Unsupervised fine-tuning with hard examples. In *Com-*

*puter Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part I*, pages 3–20.

Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. 2017. Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(6):1137–1149.

Filipe Rodrigues, Francisco C. Pereira, and Bernardete Ribeiro. 2014. Sequence labeling with multiple annotators. *Machine Learning*, 95(2):165–181.

Anna Rohrbach, Marcus Rohrbach, Ronghang Hu, Trevor Darrell, and Bernt Schiele. 2016. Grounding of textual phrases in images by reconstruction. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part I*, pages 817–834.

Mingzhe Wang, Mahmoud Azab, Noriyuki Kojima, Rada Mihalcea, and Jia Deng. 2016. Structured matching for phrase localization. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VIII*, pages 696–711.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 2048–2057.

Licheng Yu, Patrick Poirson, Shan Yang, Alexander C. Berg, and Tamara L. Berg. 2016. Modeling context in referring expressions. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II*, pages 69–85.

Zhou Yu, Jun Yu, Jianping Fan, and Dacheng Tao. 2017. Multi-modal factorized bilinear pooling with co-attention learning for visual question answering. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 1839–1848.

Zhou Yu, Jun Yu, Chenchao Xiang, Jianping Fan, and Dacheng Tao. 2018a. Beyond bilinear: Generalized multimodal factorized high-order pooling for visual question answering. *IEEE Trans. Neural Netw. Learning Syst.*, 29(12):5947–5959.

Zhou Yu, Jun Yu, Chenchao Xiang, Zhou Zhao, Qi Tian, and Dacheng Tao. 2018b. Rethinking diversified and discriminative proposal generation for visual grounding. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden.*, pages 1114–1120.

# What You See is What You Get: Visual Pronoun Coreference Resolution in Dialogues

**Xintong Yu**[♡,*], **Hongming Zhang**[♣,*], **Yangqiu Song**[♣], **Yan Song**[♠], **and Changshui Zhang**[♡]
[♡]Institute for Artificial Intelligence, Tsinghua University (THUAI),
Department of Automation, Tsinghua University
[♣]Department of CSE, The Hong Kong University of Science and Technology
[♠]Sinovation Ventures
yuxt16@mails.tsinghua.edu.cn, hzhangal@cse.ust.hk, yqsong@cse.ust.hk,
clksong@gmail.com, zcs@mail.tsinghua.edu.cn

## Abstract

Grounding a pronoun to a visual object it refers to requires complex reasoning from various information sources, especially in conversational scenarios. For example, when people in a conversation talk about something all speakers can see, they often directly use pronouns (e.g., it) to refer to it without previous introduction. This fact brings a huge challenge for modern natural language understanding systems, particularly conventional context-based pronoun coreference models. To tackle this challenge, in this paper, we formally define the task of visual-aware pronoun coreference resolution (PCR) and introduce VisPro, a large-scale dialogue PCR dataset, to investigate whether and how the visual information can help resolve pronouns in dialogues. We then propose a novel visual-aware PCR model, VisCoref, for this task and conduct comprehensive experiments and case studies on our dataset. Results demonstrate the importance of the visual information in this PCR case and show the effectiveness of the proposed model.

## 1 Introduction

The question of how human beings resolve pronouns has long been an attractive research topic in both linguistics and natural language processing (NLP) communities, for the reason that pronoun itself has weak semantic meaning (Ehrlich, 1981) and the correct resolution of pronouns requires complex reasoning over various information. As a core task of natural language understanding, pronoun coreference resolution (PCR) (Hobbs, 1978) is the task of identifying the noun (phrase) that pronouns refer to. Compared with the general coreference resolution task, the string-matching methods are no longer effective for pronouns (Stoyanov et al., 2009), which makes PCR more challenging than the general coreference resolution task.

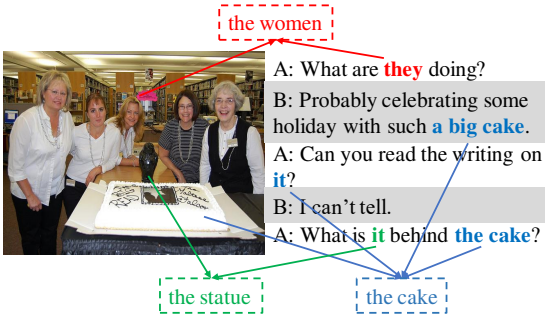


Figure 1: An example of a visual-related dialogue. Two people are discussing the view they can both see. Pronouns and noun phrases referring to the same entity are marked in same color. The first "it" in the dialogue labeled with blue color refers to the object "the big cake" and the second "it" labeled with green color refers to the statue in the image.

Recently, great efforts have been devoted into the coreference resolution task (Raghunathan et al., 2010; Clark and Manning, 2015, 2016; Lee et al., 2018) and good performance has been achieved on formal written text such as newspapers (Pradhan et al., 2012; Zhang et al., 2019a) and diagnose records (Zhang et al., 2019b). However, when it comes to dialogues, where more abundant information is needed, the performance of existing models becomes less satisfying. The reason behind is that, different from formal written language, correct understanding of spoken language often requires the support of other information sources. For example, when people chat with each other, if they intend to refer to some object that all speakers can see, they may directly use pronouns such as "it" instead of describing or mentioning it in the first place. Sometimes, the object (name or text description) that pronouns refer to may

[*] Equal contribution.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5123–5132,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

not even appear in a conversation, and thus one needs to ground the pronouns into something outside the text, which is extremely challenging for conventional approaches purely based on human-designed rules (Raghunathan et al., 2010) or contextual features (Lee et al., 2018).

A visual-related dialogue is shown in Figure 1. Both A and B are talking about a picture, in which several people are celebrating something. In the dialogue, the first "it" refers to the "the big cake," which is relatively easy for conventional models, because the correct mention just appears before the targeting pronoun. However, the second "it" refers to the statue in the image, which does not appear in the dialogue at all. Without the support of the visual information, it is almost impossible to identify the coreference relation between "it" and "the statue."

In this work, we focus on investigating how visual information can help better resolve pronouns in dialogues. To achieve this goal, we first create VisPro, a large-scale visual-supported PCR dataset. Different from existing datasets such as ACE (NIST, 2003) and CoNLL-shared task (Pradhan et al., 2012), VisPro is annotated based on dialogues discussing given images. In total, VisPro contains annotations for 29,722 pronouns extracted from 5,000 dialogues. Once the dataset is created, we formally define a new task, visual pronoun coreference resolution (Visual PCR), and design a novel visual-aware PCR model VisCoref, which can effectively extract information from images and leverage them to help better resolve pronouns in dialogues. Particularly, we align mentions in the dialogue with objects in the image and then jointly use the contextual and visual information for the final prediction.

The contribution of this paper is three-folded: (1) we formally define the task of visual PCR; (2) we present VisPro, the first dataset that focuses on PCR in visual-supported dialogues; (3) we propose VisCoref, a visual-aware PCR model. Comprehensive experiments and case studies are conducted to demonstrate the quality of VisPro and the effectiveness of VisCoref. The dataset, code, and models are available at: https://github.com/HKUST-KnowComp/Visual_PCR.

```
(S
  (NP (NP (DT A) (NN man)) (PP (IN with) (NP (DT a) (NN dog))))
  (VP
    (VBZ is)
    (VP (VBG walking) (PP (IN on) (NP (DT the) (NN grass)))))
 (. .))
```

Figure 2: Example syntax parsing result of the sentence "A man with a dog is walking on the grass."

## 2 The VisPro Dataset

To generate a high-quality and large-scale visual-aware PCR dataset, we select VisDial (Das et al., 2017) as the base dataset and invite annotators to annotate. In VisDial, each image is accompanied by a dialogue record discussing that image. One example is shown in Figure 1. In addition, VisDial also provides a caption for each image, which brings more information for us to create VisPro[1]. In this section, we introduce the details about the dataset creation in terms of pre-processing, survey design, annotation, and post-processing.

### 2.1 Pre-processing

To make the annotation task clear to annotators and help them provide accurate annotation, we first extract all the noun phrases and pronouns with Stanford Parser (Klein and Manning, 2003) and then provide the extracted noun phrases as candidate mentions to annotate on. To avoid the overlap of candidate noun phrases, we choose noun phrases with a height of two in parse trees. One example is shown in Figure 2. In the syntactic tree for the sentence "A man with a dog is walking on the grass," we choose "A man," "a dog" and "the grass" as candidates. If the height of noun phrases is not limited, then the noun phrase "A man with a dog" will cover "A man" and "a dog," leading to confusion in the options.

Following (Strube and Müller, 2003; Ng, 2005), we only select third-person personal (it, he, she, they, him, her, them) and possessive pronouns (its, his, her, their) as the targeting pronouns. In total, the VisPro dataset contains 29,722 pronouns of 5,000 dialogues selected from 133,351 dialogues in VisDial v1.0 (Das et al., 2017). We choose dialogues in which the number of pronouns ranges from four to ten for the following reasons. For one thing, dialogues with few pronouns are of lit-

[1]The information contained in the caption only helps provide noun phrase candidates for workers to annotate and will not be treated as part of the dialogue.

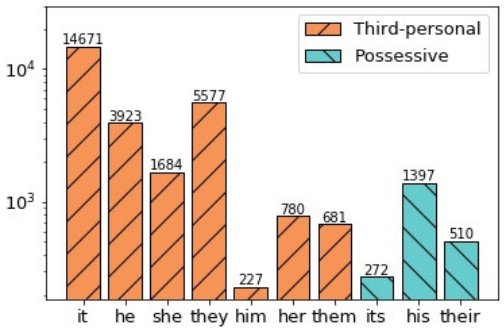


Figure 3: Distribution of pronouns in VisPro.

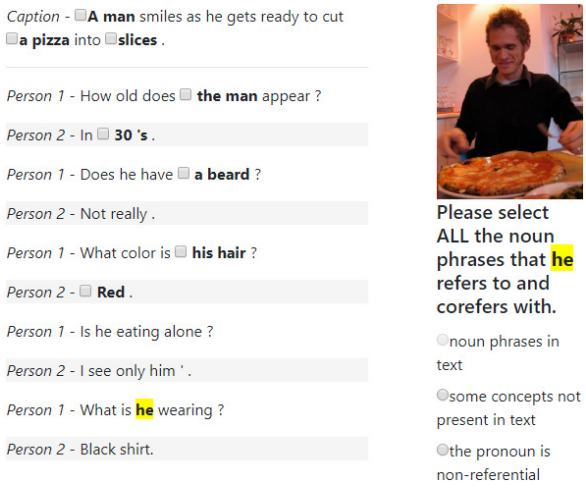


Figure 4: Annotation interface on MTurk. Workers are asked to select the anaphoricity type of the highlighted pronoun in the right panel and all the antecedents of an anaphoric pronoun on the left panel.

tle use to the task. For another, dialogues with too many pronouns often contain repeating pronouns referring to the same object, which makes the task too easy. The dialogues selected contain 5.94 pronouns on average. Figure 3 presents the distribution of different pronouns. From the figure we can see that "it" and "they" are used most frequently in the dialogues.

## 2.2 Survey Design

We divide 29,722 pronouns from 5,000 dialogues into 3,304 surveys. In each survey, besides the normal questions, we also include one checkpoint question to control the annotation quality[2]. In total, each survey contains ten questions (nine normal questions and one checkpoint question). Each question is about one pronoun.

The survey consists of three parts. We begin by explaining the task to the annotators, including how to deal with particular cases such as multi-word expressions. Then, we present examples to help the annotators better understand the task. Finally, we invite them to provide annotations for the pronouns in the dialogues.

One example of the annotation interface is shown in Figure 4. The text and the image of the dialogue are displayed on the left and right panel, respectively. For each of the targeting pronoun, the workers are asked to select all the mentions that it refers to. If any of the noun phrases is selected, the reference type of the pronoun on the right panel will be set to "noun phrases in text" automatically, and vice versa. If the concept that the pronoun refers to is not available in the options, or the pronoun is not referring to anything in particular, workers are asked to choose "some concepts not present in text" or "the pronoun is non-referential" on the right panel accordingly. The nine normal questions in each survey are consecutive so that pronouns in the same dialogue are displayed sequentially.

[2]We design the checkpoint dialogue straightforward and unambiguous such that any responsive worker can easily provide the correct annotation.

## 2.3 Annotation

We employ the Amazon Mechanical Turk platform (MTurk) for our annotations.

We require that our annotators have more than 1,000 approved tasks and a task approval rate more than 90%. They are also asked to pass a simple test of pronoun resolution to prove that they understand the instruction of the task. Based on these criteria, we identify 186 valid annotators. For each dialogue, we invite at least four different workers to annotate. In total, we collect 122,443 annotations at a total cost of USD 3,270.80. We support the multiple participation of annotators by ensuring that subsequent surveys are generated with their previously-unlabelled dialogues.

## 2.4 Post-processing

Before processing the annotation result, we first exclude the annotation of workers who fail to answer the checkpoint questions correctly. As a result, 116,300 annotations are kept, which is 95% of the overall annotation results. We then decide the gold mentions that pronouns refer to using the following procedure:

- Step one: We look into the annotations of each

worker to find out the coreference clusters he annotates for each dialogue. To achieve this goal, we merge the intersecting sets of noun phrase antecedents for pronouns in the same dialogue. We observe that annotators often make the right decision for noun phrases near the anaphor pronoun, but neglect antecedents far away. It also happens in the annotation of other coreference datasets (Chen et al., 2018). Therefore, we generate clusters from different pronouns in the same dialogue rather than merging antecedents for each pronoun separately. If an entity is mentioned and referred to by pronouns for multiple times in the dialogue, combining the antecedents for all pronouns could create a more accurate coreference cluster for the entity.

- Step two: We adjudicate the coreference clusters for the dialogue by majority voting within all workers.

- Step three: We then decide the anaphoric type of all pronouns by voting. If a pronoun is considered to refer to somef noun phrases in the text, we find out the coreference cluster it belongs to and choose the noun phrases in the cluster that precede it as its antecedents.

- Step four: We randomly split the collected data into train/val/test sets of 4,000/500/500 dialogues, respectively.

After collecting the data, we found out that 73.43% of pronouns act as an anaphor to some noun phrases, 5.67% of pronouns do not have a suitable antecedent, and the rest 20.90% are not referential. Among all the pronouns that have noun phrases as antecedents, 13.45% of them do not have an antecedent in the dialogue context.[3] For anaphoric pronouns, each has 2.06 antecedents on average. In the end, we calculate the inner-annotator agreement (IAA) to demonstrate the quality of the resulting dataset. Following conventional approaches (Pradhan et al., 2012), we use the average MUC score between individual workers and the adjudication result as the evaluation metric. The final IAA score is 72.4, which indicates that the workers can clearly understand our task and provide reliable annotation.

[3]The antecedent labeled by the worker is provided by the caption.

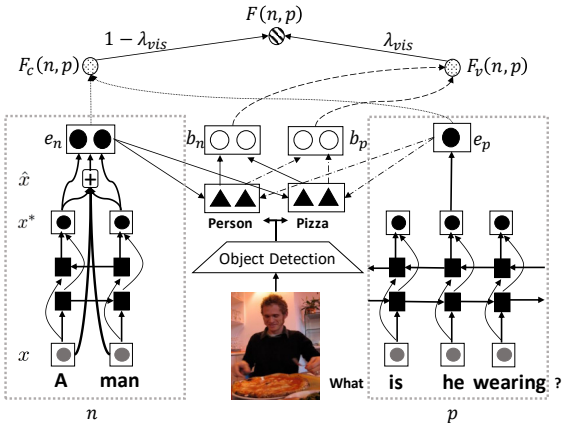


Figure 5: Overall structure of the VisCoref model. Text embedding $e$ is acquired via LSTM and inner-span attention mechanism. Object labels are detected from the image and the possibility $b$ of each text span referring to them is calculated. The contextual and visual score $F_c(n,p)$ and $F_v(n,p)$ are calculated upon the features, and the final score $F(n,p)$ is their weighted sum.

## 3 The Task

In this work, we focus on jointly leveraging the contextual and visual information to resolve pronouns. Thus, we formally define the visual-aware pronoun coreference resolution as follows:

Given an image $I$, a dialogue record $D$ which discusses the content in $I$, and an external mention set $\mathcal{M}$, for any pronoun $p$ that appears in $D$, the goal is to optimize the following objective:

$$\mathcal{J} = \frac{\sum_{c \in \mathcal{C}} e^{F(c,p,I,D)}}{\sum_{s \in \mathcal{S}} e^{F(s,p,I,D)}}, \quad (1)$$

where $F(\cdot)$ is the overall scoring function of $p$ refers to $c$ given $I$ and $D$. $c$ and $s$ denote the correct mention and the candidate mention, and $\mathcal{C}$ and $\mathcal{S}$ denote the correct mention set and the candidate mention set, respectively. Note that in the case where no golden mentions are annotated, the union of all possible spans in $D$ and $\mathcal{M}$ are used to form $\mathcal{S}$.

## 4 The Model

The overall framework of the proposed model VisCoref is presented in Figure 5. In VisCoref, we want to leverage both the contextual and visual information to resolve pronouns. Thus we split the scoring function into two parts as follows:

$$F(n,p) = (1-\lambda_{vis}) \cdot F_c(n,p) + \lambda_{vis} \cdot F_v(n,p), \quad (2)$$

where $F_c(\cdot)$ and $F_v(\cdot)$ are the scoring functions based on contextual and visual information respectively. $\lambda_{vis}$ is the hyper-parameter to control the importance of visual information in the model. The details of the two scoring functions are described in the following subsections.

### 4.1 Contextual Scoring

Before computing $F_c$, we first need to encode the contextual information into all the candidates and targeting pronouns through a mention representation module, which is shown as the dotted box in Figure 5.

Following (Lee et al., 2018), a standard bidirectional LSTM (BiLSTM) (Hochreiter and Schmidhuber, 1997) is used to encode each span with attentions (Bahdanau et al., 2015). Assume initial embeddings of words in a span $s_i$ are denoted as $\mathbf{x}_1, ..., \mathbf{x}_T$, and their encoded representation after BiLSTM as $\mathbf{x}_1^*, ..., \mathbf{x}_T^*$, the weighted embeddings of each span $\hat{\mathbf{x}}_i$ are obtained by

$$\hat{\mathbf{x}}_i = \sum_{t=1}^{T} a_t \cdot \mathbf{x}_t, \tag{3}$$

where $a_t$ is the inner-span attention computed by

$$a_t = \frac{e^{\alpha_t}}{\sum_{k=1}^{T} e^{\alpha_k}}, \tag{4}$$

in which $\alpha_t$ is obtained by a standard feed-forward neural network[4] $\alpha_t = NN_\alpha(\mathbf{x}_t^*)$.

After that, we concatenate the embeddings of the starting word ($\mathbf{x}_{start}^*$) and the ending word ($\mathbf{x}_{end}^*$) of each span, as well as its weighted embedding ($\hat{\mathbf{x}}_i$) and the length feature ($\phi(i)$) to form its final representation $\mathbf{e}$:

$$\mathbf{e}_i = [\mathbf{x}_{start}^*, \mathbf{x}_{end}^*, \hat{\mathbf{x}}_i, \phi(i)]. \tag{5}$$

On top of the extracted mention representation, we then compute the contextual score as follows:

$$F_c(n, p) = NN_c([\mathbf{e}_p, \mathbf{e}_n, \mathbf{e}_p \odot \mathbf{e}_n]), \tag{6}$$

where [ ] represents the concatenation, $\mathbf{e}_p$ and $\mathbf{e}_n$ are the mention representation of the targeting pronoun and current candidate mention, and $\odot$ indicates the element-wise multiplication.

[4]We use $NN$ to represent feed-forward neural networks.

### 4.2 Visual Scoring

In order to align mentions in the text with objects in the image, the first step of leveraging the visual information is to recognize the objects from the picture. We use a object detection module to identify object labels from each image $I$, such as "person," "car," or "dog." After that, we convert the identified labels into vector representations following the same encoding process in 4.1. For each image, we add a label "null," indicating that the pronoun is referring to none of the detected objects in $I$. We denote the resulting embeddings as $\mathbf{e}_{c_1}, \mathbf{e}_{c_2}, ..., \mathbf{e}_{c_K}$, in which $c_i$ denotes the detected labels, and $K$ is the total number of unique labels in the corresponding image.

After extracting objects from the image, we would like to know whether the mentions are referring to them. To achieve this goal, we calculate the possibility of a mention $n_i$ corresponding to each detected object $c_k$:

$$\beta_{n_i,c_k} = \mathrm{NN}_\beta\left(\mathrm{NN}_o(\boldsymbol{e}_{n_i}) \odot \mathrm{NN}_o(\boldsymbol{e}_{c_k})\right). \tag{7}$$

Then we take the softmax of $\beta_{n_i,c_k}$ as the final possibility of $n_i$ aligned with the object label $c_k$:

$$b_{n_i,c_k} = \frac{e^{\beta_{n_i,c_k}}}{\sum_{l=1}^{K} e^{\beta_{n_i,c_l}}}. \tag{8}$$

If $n_i$ corresponds to a certain object in $I$, the score of that label should be large. Otherwise, the possibility of "null" should be the largest. Therefore, we use the maximum of possibility scores among all $K$ classes except "null"

$$m_i = \max_{k=1,...,K} b_{n_i,c_k} \tag{9}$$

as the probability of $n_i$ related to some object in $I$.

Similarly, given two mentions $n_i$ and $n_j$, if they refer to the same detected object $c_k$, then both $b_{n_i,c_k}$ and $b_{n_j,c_k}$ should be large. Thus, we can use the maximum of their product among all K classes except "null"

$$m_{i,j} = \max_{k=1,...,K} b_{n_i,c_k} \cdot b_{n_j,c_k} \tag{10}$$

as the probability of $n_i$ and $n_j$ related to the same object in $I$.

In the end, we define the visual scoring function as follows:

$$F_v(n, p) = NN_v([m_p, m_n, m_p \cdot m_n, m_{p,n}]). \tag{11}$$

## 5 The Experiment

In this section, we introduce the implementation details, experiment setting, and baseline models.

### 5.1 Experiment Setting

As introduced in Section 2.4, we randomly split the data into training, validation, and test sets of 4,000, 500, and 500 dialogues, respectively. For each dialogue, a mention pool of size 30 is provided for models to detect plausible mentions outside the dialogue. The pool contains both mentions extracted from the corresponding caption and randomly selected negative mention samples from other captions. All models are evaluated based on the precision (P), recall (R), and F1 score. Last but not least, we split the test dataset by whether the correct antecedents of the pronoun appear in the dialogue or not. We denote these two groups as "Discussed" and "Not Discussed."

### 5.2 Implementation Details

Following previous work (Lee et al., 2018), we use the concatenation of the 300d GloVe embedding (Pennington et al., 2014) and the ELMo (Peters et al., 2018) embedding as the initial word representations. Out-of-vocabulary words are initialized with zero vectors. We adopt the "ssd_resnet_50_fpn_coco" model from Tensorflow detection model zoo[5] as the object detection module. The size of hidden states in the LSTM module is set to 200, and the size of the projected embedding for computing similarity between text spans and object labels is 512. The feed-forward networks for contextual scoring and visual scoring have two 150-dimension hidden layers and one 100-dimension hidden layer, respectively.

For model training, we use cross-entropy as the loss function and Adam (Kingma and Ba, 2015) as the optimizer. All the parameters are initialized randomly. Each mention selects the text span of the highest overall score among all previous text spans in the dialogue or the mention pool as its antecedent, so that all mentions in one dialogue are clustered into several coreference chains. The noun phrases in the same coreference cluster as a pronoun are deemed as the predicted antecedents of that pronoun. The models are trained with up to 50,000 steps, and the best one is selected based on its performance on the validation set.

[5] https://github.com/tensorflow/models/tree/master/research/object_detection

### 5.3 Baseline Methods

Since we are the first to proposed a visual-aware model for pronoun coreference resolution, we compare our results with existing models of general coreference resolution.

- Deterministic model (Raghunathan et al., 2010) is a rule-based system that aggregates multiple functions for determining whether two mentions are coreferent based on hand-craft features.
- Statistical model (Clark and Manning, 2015) learns upon human-designed entity-level features between clusters of mentions to produce accurate coreference chains.
- Deep-RL model (Clark and Manning, 2016) applies reinforcement learning to mention-ranking models to form coreference clusters.
- End-to-end model (Lee et al., 2018) is the state-of-the-art method of coreference resolution. It predicts coreference clusters via an end-to-end neural network that leverages pretrained word embeddings and contextual information.

Last but not least, to demonstrate the effectiveness of the proposed model, we also present a variation of the End-to-end model, which can also use the visual information, as an extra baseline:

- End-to-end+Visual first extracts features from images with ResNet-152 (He et al., 2016). Then it concatenates the image feature with the contextual feature in the original End-to-end model together to make the final prediction.

As the Deterministic, Statistical, and Deep-RL model are included in the Stanford CoreNLP toolkit[6], we use their released model as baselines. For the End-to-end model, we also use their released code[7].

## 6 The Result

The experimental results are shown in Table 1. Our proposed model VisCoref outperforms all the baseline models significantly, which indicates that the visual information is crucial for resolving pronouns in dialogues. Besides that, we also have the following interesting findings:

[6] https://stanfordnlp.github.io/CoreNLP/coref.html

[7] https://github.com/kentonl/e2e-coref

| Model | Discussed | | | Not Discussed | | | Overall | | |
|---|---|---|---|---|---|---|---|---|---|
| | P | R | F1 | P | R | F1 | P | R | F1 |
| Deterministic | 61.16 | 25.54 | 36.03 | 17.39 | 8.26 | 11.20 | 56.65 | 24.01 | 33.73 |
| Statistical | 79.37 | 26.65 | 39.90 | 6.10 | 1.03 | 1.77 | 76.18 | 24.23 | 36.76 |
| Deep-RL | 72.69 | 26.97 | 39.35 | 32.86 | 14.46 | 20.09 | 68.51 | 25.93 | 37.62 |
| End-to-end | 89.65 | 63.69 | 74.47 | 67.33 | 64.76 | 66.02 | 86.94 | 63.79 | 73.59 |
| End-to-end+Visual | 89.78 | 66.31 | 76.28 | 65.17 | 62.38 | 63.75 | 86.91 | 65.95 | 74.99 |
| Viscoref | 85.95 | 72.45 | **78.63** | 67.04 | 71.67 | **69.28** | 83.78 | 72.38 | **77.66** |
| Human | 95.03 | 81.82 | 87.93 | 86.18 | 93.57 | 89.73 | 94.02 | 82.91 | 88.12 |

Table 1: Experimental results on VisPro. Precision (P), recall (R) and F1 scores are presented. The best performed F1 scores are indicated with bold font.

1. For all the conventional models, the "Not Discussed" pronouns whose antecedents are absent in dialogues are more challenging than "Discussed" pronouns whose antecedents appear in the dialogue context. The reason behind is that if the correct mentions appear in the near context of pronouns, the information about the correct mention can be aggregated to the targeting pronoun through either human-designed rules or deep neural networks (Bi-LSTM). However, when the correct mention is not available in the near context, it is quite challenging for conventional models to understand the dialogue and correctly ground the pronoun to the object both speakers can see, as they do not have the support of visual information.

2. As is shown in the result of the "End-to-end+Visual" model, simply concatenating the visual feature to the contextual feature can help resolve "Discussed" pronouns but may hurt the performance of the model on "Not Discussed" pronouns. Different from them, the proposed Viscoref can improve the resolution of both the "Discussed" and "Not Discussed" pronouns. There are mainly two reasons behind: (1) The visual information in our model is first converted into textual labels and then transformed into vector representation in the same way as the dialogue context. Thus the vector space of contextual and visual information is perfectly aligned. (2) We introduce a hyper-parameter $\lambda_{vis}$ to balance the influence of different knowledge resources.

3. Even though our model outperforms all the baseline methods, we still can observe a huge gap between our model and human being. It indicates that current models still cannot fully understand the dialogue even with the support of visual information and further proves the value and necessity of proposing VisPro.

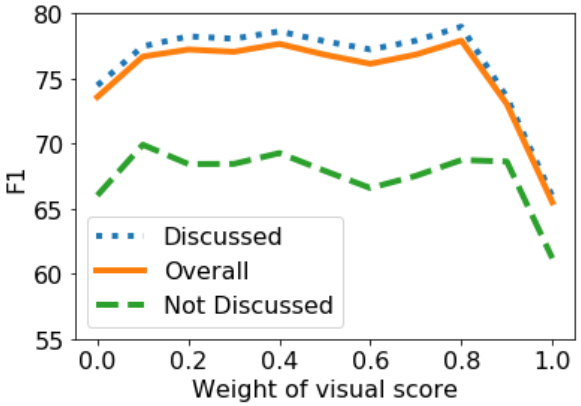


Figure 6: Effect of $\lambda_{vis}$. F1 scores of all categories are reported.

### 6.1 Hyper-parameter Analysis

We traverse different weights of visual and contextual information from 0 to 1, and the result is shown in Figure 6. Along with the increase of $\lambda_{vis}$, our model puts more weight on the visual information. As a result, our model can perform better. However, when our model focuses too much on the visual information (when $\lambda_{vis}$ equals to 0.9 or 1), the model overfits to the visual information and thus performs poorly on the task. To achieve the balance between the visual and contextual information, we set $\lambda_{vis}$ to be 0.4.

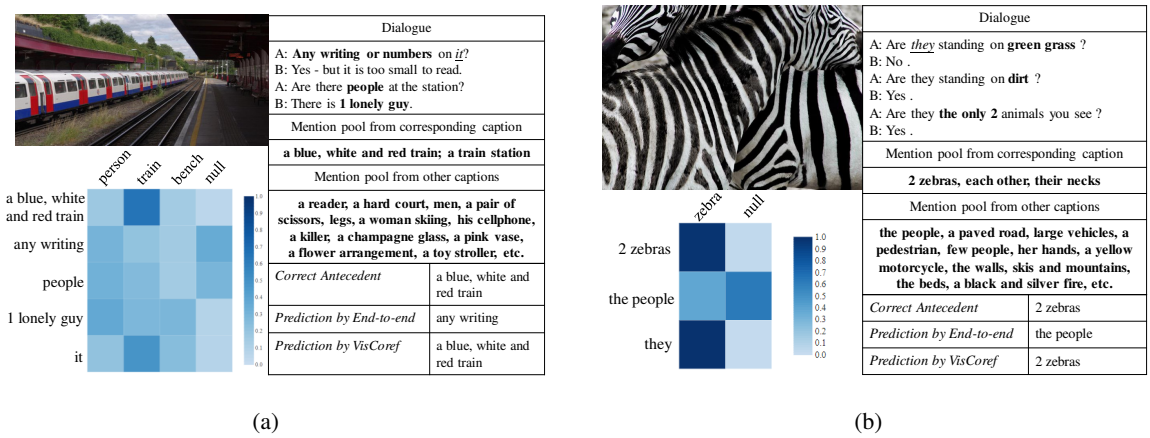


| Dialogue | |
|---|---|
| A: **Any writing or numbers** on *it*?<br>B: Yes - but it is too small to read.<br>A: Are there **people** at the station?<br>B: There is **1 lonely guy**. | |
| Mention pool from corresponding caption | |
| **a blue, white and red train; a train station** | |
| Mention pool from other captions | |
| **a reader, a hard court, men, a pair of scissors, legs, a woman skiing, his cellphone, a killer, a champagne glass, a pink vase, a flower arrangement, a toy stroller, etc.** | |
| *Correct Antecedent* | a blue, white and red train |
| *Prediction by End-to-end* | any writing |
| *Prediction by VisCoref* | a blue, white and red train |

(a)

| Dialogue | |
|---|---|
| A: Are *they* standing on **green grass** ?<br>B: No .<br>A: Are they standing on **dirt** ?<br>B: Yes .<br>A: Are they **the only 2** animals you see ?<br>B: Yes . | |
| Mention pool from corresponding caption | |
| **2 zebras, each other, their necks** | |
| Mention pool from other captions | |
| **the people, a paved road, large vehicles, a pedestrian, few people, her hands, a yellow motorcycle, the walls, skis and mountains, the beds, a black and silver fire, etc.** | |
| *Correct Antecedent* | 2 zebras |
| *Prediction by End-to-end* | the people |
| *Prediction by VisCoref* | 2 zebras |

(b)

Figure 7: Randomly selected examples from VisPro. The image, dialogue record, prediction result, and heatmap of the mention-object similarity are provided. We indicate the target pronoun with the *underlined italics* font and the candidate mentions with **bold** font. Only relevant parts of dialogues are presented. The row of the heatmap represents mentions in the context, and the column means detected object labels from the image.

### 6.2 Case Study

To further investigate how visual information can help solve PCR, we randomly select two examples and show the prediction results of VisCoref and End-to-end model in Figure 7.

In the first example in Figure 7(a), given the pronoun "it," the End-to-end model picks "any writing" from the dialogue, while the VisCoref model chooses "a blue, white and red train" from the candidate mention sets. Without looking at the picture, we cannot distinguish between these two candidates. However, when the picture is taken into consideration, we observe that there is a train in the image and thus "a blue, white and red train" is a more suitable choice, which proves the importance of visual information. A similar situation happens in Figure 7(b), where the End-to-end model connects "they" to "the people" but there is no human being in the image at all. On the contrary, as VisPro can effectively leverage the visual information and make the decision that "they" should refer to "2 zebras."

## 7 Related Work

In this section, we introduce the related work about pronoun coreference resolution and visual-aware natural language processing problems.

### 7.1 Pronoun Coreference Resolution

As one core task of natural language understanding, pronoun coreference resolution, the task of identifying mentions in text that the targeting pronoun refers to, plays a vital role in many downstream applications in natural language processing, such as machine translation (Guillou, 2012), summarization (Steinberger et al., 2007) and information extraction (Edens et al., 2003). Traditional studies focus on resolving pronouns in expert-annotated formal textual dataset such as ACE (NIST, 2003) or OntoNotes (Pradhan et al., 2012). However, models that perform well on these datasets might not perform as well in other scenarios such as dialogues due to the informal language and the lack of essential information (e.g., the shared view of two speakers). In this work, we thus focus on the PCR in dialogues and show that the information contained in the shared view can be crucial for understanding the dialogues and correctly resolving the pronouns.

### 7.2 Visual-aware NLP

As the intersection of computer vision (CV) and natural language processing (NLP), visual-aware NLP research topics have been popular in both communities. For instance, image captioning (Xu et al., 2015) focuses on generating captions for images, visual question answering (VQA) (Antol et al., 2015) on answering questions about a image, and visual dialogue (Das et al., 2017) on generating a suitable response based on images. As one vital step of all the aforementioned visual-aware natural language processing tasks (Kottur et al., 2018), the visual-aware PCR is still unexplored. To fill this gap, in this paper, we create VisPro, which is a large-scale visual-aware PCR

dataset, and introduce VisCoref to demonstrate how to leverage information hidden in the shared view to resolve pronouns in dialogues better and thus understand the dialogues better.

Another related work is the comprehension of referring expressions (Mao et al., 2016), which is inferring the object in an image that an expression describes. However, the task is formulated on isolated noun phrases specially designed for discriminative descriptions without putting them into a meaningful context. Instead, our task focuses on resolving pronouns in dialogues based on images as the shared view, which enhances the understanding of dialogues based on the comprehension of expressions and images.

## 8 Conclusion

In this work, we formally define the task of visual pronoun coreference resolution (PCR) and present VisPro, the first large-scale visual-supported pronoun coreference resolution dataset. Different from conventional pronoun datasets, VisPro focuses on resolving pronouns in dialogues which discusses a view that both speakers can see.

Moreover, we also propose VisCoref, the first visual-aware PCR model that aligns contextual information with visual information and jointly uses them to find the correct objects that the targeting pronouns refer to. Extensive experiments demonstrate the effectiveness of the proposed model. Further case studies also demonstrate that jointly using visual information and contextual information is an essential path for fully understanding human language, especially dialogues.

## Acknowledgements

This paper was supported by Beijing Academy of Artificial Intelligence (BAAI), the Early Career Scheme (ECS, No. 26206717) from Research Grants Council in Hong Kong, and the Tencent AI Lab Rhino-Bird Focused Research Program.

## References

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. VQA: visual question answering. In *Proceedings of ICCV*, pages 2425–2433.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*.

Hong Chen, Zhenhua Fan, Hao Lu, Alan L. Yuille, and Shu Rong. 2018. Preco: A large-scale dataset in preschool vocabulary for coreference resolution. In *Proceedings of EMNLP*, pages 172–181.

Kevin Clark and Christopher D. Manning. 2015. Entity-centric coreference resolution with model stacking. In *Proceedings of ACL*, pages 1405–1415.

Kevin Clark and Christopher D. Manning. 2016. Deep reinforcement learning for mention-ranking coreference models. In *Proceedings of EMNLP*, pages 2256–2262.

Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José M. F. Moura, Devi Parikh, and Dhruv Batra. 2017. Visual dialog. In *Proceedings of CVPR*, pages 1080–1089.

Richard J. Edens, Helen L. Gaylard, Gareth J. F. Jones, and Adenike M. Lam-Adesina. 2003. An investigation of broad coverage automatic pronoun resolution for information retrieval. In *Proceedings of SIGIR*, pages 381–382.

Kate Ehrlich. 1981. Search and inference strategies in pronoun resolution: an experimental study. In *Proceedings of ACL*, pages 89–93.

Liane Guillou. 2012. Improving pronoun translation for statistical machine translation. In *Proceedings of EACL*, pages 1–10.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of CVPR*, pages 770–778.

Jerry R Hobbs. 1978. Resolving pronoun references. *Lingua*, 44(4):311–338.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of ACL*, pages 423–430.

Satwik Kottur, José M. F. Moura, Devi Parikh, Dhruv Batra, and Marcus Rohrbach. 2018. Visual coreference resolution in visual dialog using neural module networks. In *Proceedings of ECCV*, pages 160–178.

Kenton Lee, Luheng He, and Luke Zettlemoyer. 2018. Higher-order coreference resolution with coarse-to-fine inference. In *Proceedings of NAACL-HLT*, pages 687–692.

Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L. Yuille, and Kevin Murphy. 2016. Generation and comprehension of unambiguous object descriptions. In *Proceedings of CVPR*, pages 11–20.

Vincent Ng. 2005. Supervised ranking for pronoun resolution: Some recent improvements. In *Proceedings of AAAI*, pages 1081–1086.

US NIST. 2003. The ace 2003 evaluation plan. *US National Institute for Standards and Technology (NIST)*, pages 2003–08.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Proceedings of EMNLP-CoNLL*, pages 1–40.

Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nate Chambers, Mihai Surdeanu, Dan Jurafsky, and Christopher D. Manning. 2010. A multi-pass sieve for coreference resolution. In *Proceedings of EMNLP*, pages 492–501.

Josef Steinberger, Massimo Poesio, Mijail A. Kabadjov, and Karel Jezek. 2007. Two uses of anaphora resolution in summarization. *Inf. Process. Manage.*, 43(6):1663–1680.

Veselin Stoyanov, Nathan Gilbert, Claire Cardie, and Ellen Riloff. 2009. Conundrums in noun phrase coreference resolution: Making sense of the state-of-the-art. In *Proceedings of ACL*, pages 656–664.

Michael Strube and Christoph Müller. 2003. A machine learning approach to pronoun resolution in spoken dialogue. In *Proceedings of ACL*, pages 168–175.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of ICML*, pages 2048–2057.

Hongming Zhang, Yan Song, and Yangqiu Song. 2019a. Incorporating context and external knowledge for pronoun coreference resolution. In *Proceedings of NAACL-HLT*, pages 872–881.

Hongming Zhang, Yan Song, Yangqiu Song, and Dong Yu. 2019b. Knowledge-aware pronoun coreference resolution. In *Proceedings of ACL*, pages 867–876.

# YouMakeup: A Large-Scale Domain-Specific Multimodal Dataset for Fine-Grained Semantic Comprehension

**Weiying Wang** **Yongcheng Wang** **Shizhe Chen** **Qin Jin** *
School of Information, Renmin Unversity of China
{wy.wang, 2015202012, cszhe1, qjin}@ruc.edu.cn

## Abstract

Multimodal semantic comprehension has attracted increasing research interests in recent years, such as visual question answering and caption generation. However, due to the data limitation, fine-grained semantic comprehension which requires to capture semantic details of multimodal contents has not been well investigated. In this work, we introduce "YouMakeup", a large-scale multimodal instructional video dataset to support fine-grained semantic comprehension research in specific domain. YouMakeup contains 2,800 videos from YouTube, spanning more than 420 hours in total. Each video is annotated with a sequence of natural language descriptions for instructional steps, grounded in temporal video range and spatial facial areas. The annotated steps in a video involve subtle difference in actions, products and regions, which require fine-grained understanding and reasoning both temporally and spatially. In order to evaluate models' ability for fined-grained comprehension, we further propose two groups of tasks including generation tasks and visual question answering tasks from different aspects. We also establish a baseline of step caption generation for future comparison. The dataset will be publicly available at https://github.com/AIM3-RUC/YouMakeup to support research investigation in fine-grained semantic comprehension.

## 1 Introduction

Videos which naturally contain rich multimodal semantic information have been one of the main sources for knowledge acquisition. In recent years, video semantic comprehension has attracted much research attention, with a number of datasets and tasks being proposed such as activity recognition (Xu et al., 2017), dense video captioning (Krishna et al., 2017a), visual question answering (Lei et al., 2018, 2019) etc. However, most works are limited to only capturing coarse semantic information such as action recognition in broad categories. Fine-grained comprehension instead has not been fully explored, especially for discriminating actions with subtle difference or understanding temporal relations of actions in a certain activity.

Instructional videos, which contain series of steps to accomplish certain tasks, are suitable sources to investigate fine-grained semantic comprehension and reasoning. As shown in Table 1, different instructional video datasets have been released. However, current datasets suffer from small data scales or coarse annotations to support fine-grained analysis. For example, datasets collected in (Alayrac et al., 2016; Rohrbach et al., 2012; Stein and McKenna, 2013; Kuehne et al., 2014; Das et al., 2013) only contain hundreds or fewer videos and actions. Although COIN dataset (Tang et al., 2019) is in large scale, it aims to cover wide range of action categories instead of distinguishing actions with subtle difference, and also lacks fine-grained step annotations. The YouCook2 dataset (Zhou et al., 2018) contains fairly large number of videos and temporal grounded sentences in the cooking domain. However, since different cooking steps contain apparent visual variation in actions, food and kitchen utilities, it might not require fine-grained reasoning over temporal and spatial dimensions to identify different steps.

In order to overcome previous limitations, we collect a new instructional video dataset named "YouMakeup" in specific makeup domain for fine-grained multimodal semantic comprehension. The advantages of opting for the makeup domain are threefolds. Firstly, makeup instructional videos are more fine-grained in nature because different steps share the same facial background but con-

* Corresponding author.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5133–5143,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| Dataset | # of videos | Total len(h) | Avg len(m) | Step Annotation # of steps | Type | T.G | S.G | Domain | Source |
|---|---|---|---|---|---|---|---|---|---|
| "5task" (Alayrac et al., 2016) | 150 | 5 | 2 | - | Sent. | ✓ | - | General | YouTube |
| COIN (Tang et al., 2019) | 11,827 | 476.5 | 2.6 | 46,354 | Ctg.(778) | ✓ | - | General | YouTube |
| MPII (Rohrbach et al., 2012) | 44 | 8 | 10.9 | 5,609 | Ctg.(65) | ✓ | ✓ | Cooking | recorded |
| YouCook (Das et al., 2013) | 88 | 2.5 | 1.6 | - | - | - | - | Cooking | YouTube |
| 50 Salads (Stein and McKenna, 2013) | 50 | 4.5 | 6.4 | 966 | Ctg.(17) | ✓ | - | Cooking | recorded |
| Breakfast (Kuehne et al., 2014) | 1,989 | 77 | 2.3 | 8,456 | Ctg.(48) | ✓ | - | Cooking | recorded |
| Ikea FA (Toyer et al., 2017) | 101 | 4 | 2.3 | 1,911 | Ctg.(-) | ✓ | ✓ | Furniture | recorded |
| YouCook2 (Zhou et al., 2018) | 2,000 | 176 | 5.3 | 3,829 | Sent. | ✓ | - | Cooking | YouTube |
| EPIC-KITCHENs (Damen et al., 2018) | 432 | 55 | 7.6 | 39,596 | Sent. | ✓ | ✓ | Cooking | recorded |
| YouMakeup (ours) | 2,800 | 421 | 9 | 30,626 | Sent. | ✓ | ✓ | Makeup | YouTube |

Table 1: Comparison of different instructional video datasets. Ctg. = pre-defined categories; Sent. = sentence; T.G = temporal grounding; S.G = spatial grounding; 'recorded' means videos are self-recorded by collectors. The YouMakeup dataset is unique in large-scale data size and fine-grained annotations.

tain at least one subtle but critical difference in action, tool or facial area. Therefore, it requires fine-grained discrimination within temporal and spatial context. Secondly, there are abundant makeup instructional videos on the Internet with manual commentary or scripts, which makes it easy to collect and annotate. Last but not least, makeup video analysis is of great value which can facilitate both editing and searching process for cosmetic companies and users. The collected YouMakeup dataset consists of 2,800 makeup videos crawled from YouTube, which spans more than 420 hours. As shown in Figure 1, we manually annotate a sequence of natural language sentences to describe different instructional steps for each video and each step is grounded both in temporal video segment and spatial face areas in fine-grained details. There are totally 30,626 steps with 10.9 steps on average for each video, indicating the complexity of makeup activities.

For the purpose of comprehensively evaluating fine-grained analysis, we propose two groups of potential semantic comprehension tasks on YouMakeup: Generation and Question Answering (QA) tasks. The Generation tasks include temporal step segmentation, step caption generation and spatial area grounding, which reflects an overall semantic comprehension performance. In order to further measure video semantic reasoning ability, we design four QA tasks for detailed evaluations from four aspects as illustrated in Figure 8. The Facial Image Ordering task aims to track subtle changes on facial appearance after each step, which requires to reason influences of actions on object states. The Step Ordering task is to sort step descriptions according to their temporal order in the video, requiring temporal action reasoning and visual semantic matching. The Time Range Selection task requires the precise temporal localization of specific step in the video, forcing models to distinguish fine-grained difference between makeup steps. The Theme Inference task aims to select a best theme for the video, which demands high-level summarization of video content.

The main contributions of this work are threefolds: 1) We introduce a large-scale fine-grained instructional video dataset "YouMakeup" in makeup domain to support research on fine-grained multimodal semantic comprehension. To the best of our knowledge, it is the largest instructional video dataset in specific domain with fine-grained temporal and spatial grounded annotation. 2) We propose two groups of tasks to evaluate fine-grained video comprehension abilities, including generation tasks and four QA tasks, which require fine-grained semantic understanding and reasoning in different aspects and levels. 3) We propose a baseline framework for the step caption generation task to demonstrate that the fine-grained analysis and long temporal dependencies are essential for multimodal semantic comprehension.

## 2 Related Work

### 2.1 Instructional Video Datasets

Existing instructional video datasets can be divided into two groups according to the domain diversity as summarized in Table 1. The first group aims to involve diverse activities from different domains. (Alayrac et al., 2016) contains 5 tasks such as "Making a coffee" and "Changing car tire". COIN (Tang et al., 2019) is a large scale dataset which contains videos of 180 different tasks in 12 domains related to our daily life. These datasets are constructed to improve model's generalization ability rather than support the fine-grained semantic comprehension. The




| Step | Temporal range | Spatial Face Area | Step Description |
|---|---|---|---|
| Step 1 | 00:02:10~00:02:39 | eyelid | Apply eyeshadow over eyelid with brush |
| Step 2 | 00:02:40~00:02:54 | brow | Draw eyebrow using the brow shadow with brush |
| Step 3 | 00:02:56~00:03:10 | lashline | Draw winged eyelinear with eyelinear pen |
| Step 4 | 00:03:11~00:03:24 | lip | Apply red lipgloss on lips |
| Step 5 | 00:03:25~00:03:28 | cheek, hailine | Apply bronzer on the cheeks and hairline with brush |
| Step 6 | 00:03:29~00:03:34 | cheekbone, forehead, nose | Apply highlighter on the cheekbones, forehead and nose with brush |
| Step 7 | 00:03:35~00:03:47 | lash | Curl the lashes and apply mascara on the lashes |
| Step 8 | 00:03:54~00:04:15 | lash | Apply false lashes on the lashes |

Figure 1: An example video in YouMakeup dataset. We annotate a sequence of step descriptions grounded in temporal video range and spatial face areas for each video. Best viewed in color.

other group focuses on specific domain, such as furniture assembling and cooking. (Rohrbach et al., 2012; Stein and McKenna, 2013; Kuehne et al., 2014) contain videos about simple cooking activities. YouCook (Das et al., 2013) consists of 88 videos with long text summarization. These datasets are limited in both number of videos and actions. YouCook2 (Zhou et al., 2018) is relatively large with 2000 videos, spanning 176 hours. Though cooking events are of rich semantics containing various foods, kitchen utilities and actions (Nishimura et al., 2019; Hahn et al., 2018), such variety makes it hard to measure fine-grained comprehension ability of models. For example, objects in "Sprinkle salt and pepper to the taste" and "Place the bacon at the top" are very different so that it might not be necessary to understand the whole details of actions to distinguish the two steps.

The strengths of our YouMakeup dataset compared with previous works are in two aspects: (1) it is large in scale with 420 hours in total. To the best of our knowledge, it is the largest instructional dataset in specific domain with rich fine-grained annotations. (2) Facial makeup is suitable for fine-grained comprehension in nature for all activities occur on the local facial area with subtle differences.

### 2.2 Video Comprehension Tasks

A wide range of tasks have been proposed for semantic comprehension on videos, such as action detection, dense video captioning and video question answering etc. The general video captioning task requires to generate a single sentence for the whole video, which cannot describe video content in details especially for long videos. So in the dense video captioning task, the model needs to detect meaningful events in the video and generate sentence to describe each one. Comparing to the dense captioning tasks which focus on activities with very different actions such as ActivityNet challenge (Krishna et al., 2017a), makeup instructional videos are more fine-grained which contain actions with subtle difference, providing more challenges for semantic comprehension.

Question answering is another way to effectively evaluate semantic understanding. Apart from image based QA datasets such as (Malinowski and Fritz, 2014; Antol et al., 2015; Ren et al., 2015a; Johnson et al., 2017), several video based datasets have been released to explore spatial and temporal inference of the video content. However, they mainly focus on comprehension within short video clips which contain simple activities and interactions, such as (Jang et al., 2017; Kim et al., 2016; Tapaswi et al., 2016; Maharaj et al., 2017), etc. The TVQA dataset (Lei et al., 2018, 2019) is constructed from complex TV shows, but each question is associated with a short clip up to 90 seconds and more focused on joint understanding of visual and speech content. In comparison, our proposed four QA tasks on YouMakeup dataset are used to evaluate video semantic reasoning abilities from different aspects, such as spatial and temporal understanding for long videos, causality reasoning of actions and

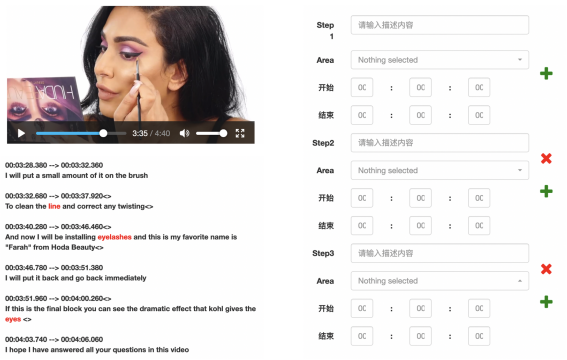


Figure 2: Annotation website

global semantic summarization.

## 3 YouMakeup Dataset

### 3.1 Data Collection

Our goal is to build a large-scale multimodal instructional video dataset in the makeup domain to support the fine-grained semantic comprehension research. We start from collecting a list of famous cosmetic brands, such as Chanel, Mac, etc., and beauty bloggers with more than tens of thousands followers. These companies and bloggers are authoritative and professional in the makeup domain with many people learning makeup skills from them. Videos in their official channels are of high quality. Based on the list, we search their official channels on the YouTube and crawl makeup instructional videos together with available meta data such as video id, duration, title, tags and English subtitles generated by YouTube automatically. We process the subtitles into complete sentences aligned with video time stamps.

Since specific names of cosmetic brands are not important to understand makeup procedures, we filter out them in raw titles and subtitles of videos. We first create an initial list of cosmetic brands and then refine it gradually by checking words frequency and finding similar words via word embedding model (Tomas Mikolov, 2013). Finally, we utilize the refined list to remove cosmetic brands in raw texts. We also create lists for facial areas and cosmetic products through the similar process.

#### 3.1.1 Step Annotation

We build an annotation system for step annotation. Figure 2 shows the interface of our annotation system. We provide video and English subtitles in the annotation page to assist the annotation. The products and facial areas in the subtitle are emphasized in red color to help annotators focus on related information. Annotators are asked to segment the video into a series of steps, which includes labeling the start and end time of each step, selecting the related facial areas in the given facial area list and creating the caption to describe the step according to the video and subtitles. We recruit female college students with more than two years of makeup experience as annotators. Before starting the annotation, each annotator is asked to annotate a test video to verify their capability for annotation. During annotation, each video is annotated by one person and reviewed by another to ensure the annotation quality.

#### 3.1.2 Facial Image Annotation

We extract two groups of facial images from the video. The first group is used to understand the effect of makeup activities on facial appearance, which supports our proposed QA task described in Section 4.2. We extract images at the beginning and the end of each step to capture the facial appearance before and after each step. In order to select images containing faces, we extract images in 40 frames around each time stamp and filter them with a pretrained Multi-Task CNN (MTCNN) (Zhang et al., 2016) for face detection. We then manually filter out unsuitable images such as side face images and pure product images.

The second group is to localize all facial areas referred in each step for spatial grounding. We extract key frames within the annotated segment of each step via comparing similarity of different frames followed by manual filtering of redundant frames. Then we automatically detect facial landmarks in the facial image and align the image region to corresponding facial areas of each step annotated in Section 3.1.1. Finally, we ask annotators to adjust the bounding box of these facial areas on the images and obtain the final grounded facial areas for each step.

#### 3.1.3 Theme Annotation

Inferring the theme of a video is a basic ability to understand the video content. Therefore, we annotate the theme for each makeup video. The original title of the video usually summarizes the content or highlights specific features in the video, which can be treated as the theme of the video. As mentioned above, the specific cosmetic brand names are removed in the title for generalization.




Figure 3: Different categories of the makeup instructional video

We further ask annotators to refine the title with the help of related meta information to make the theme more accurate.

### 3.2 Dataset Statistics

The final YouMakeup dataset contains 2800 videos, spanning 420 hours 50 minutes, and rich annotations. Videos can be mainly divided into three categories as shown in Figure 3: 1) Makeup for special occasions, such as school days or wedding days; 2) Makeup tips for specific facial area or cosmetic products, such as eye makeup or wearing red lipstick; 3) Makeup transformation, such as celebrities transformation. The first and third types usually create full looks, while the second focuses on specific step or facial area. We split the dataset into training, testing and validation set by 70%: 20%:10% and set up all the tasks on this division.

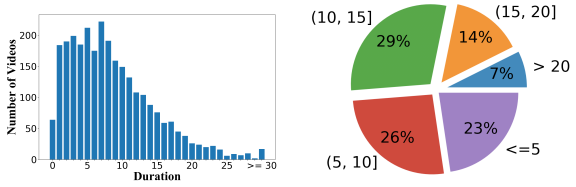


(a) Video duration. (b) Step number.

Figure 4: Statistics of video duration and step number.

#### 3.2.1 Video Length

Different from previous datasets containing videos with similar duration (Lei et al., 2018; Tang et al., 2019), videos in YouMakeup dataset are of various lengths, which reflects the complex situation in the realistic world. As shown in Figure 4(a), the length of videos varies from 15s to 1h with 9min on average. The large diversity of length results from the diverse video categories and styles. For example, tutorials from companies are usually short and come straight to the point, while those from beauty bloggers are more complex for they may show makeup skills or share their opinions about products in details.

#### 3.2.2 Makeup Steps

There are 30,626 annotated steps in total with average 10.9 steps per video in YouMakeup. Figure 4(b) shows the distribution of step number. Compared with instructional video datasets in general domains, YouMakeup is more complex with more steps on average and therefore bringing more challenge for semantic understanding.

Each step associates with at least one and up to seven facial areas. The frequency of grounded facial areas are presented in Figure 5. All these areas are close to each other on the face and might contain overlaps for some of them. For example, brow and brow bone are closely adjacent to each other and the under-eye area overlaps with cheeks. Therefore, fine-grained understanding is required to distinguish these areas.

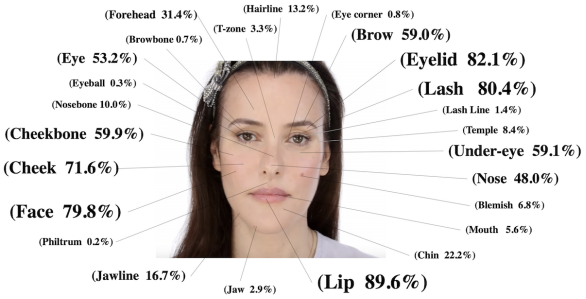


Figure 5: Frequency of grounded facial areas in YouMakeup dataset.

There are more than 1500 unique words occurred in the step captions. Figure 6(a) shows the Wordcloud for most frequent 100 words excluding stopwords. The most frequent words include actions, products, facial areas and tools as summarized in Table 2. These four categories of words can be combined in various ways, generating large number of different fine-grained makeup activities. For example, Figure 7 illustrates the fine-grained activities for the action "apply". The graph shows that the number of activities is large. Though the activities are similar, they are distinct in actions, products, facial areas or other aspects. Therefore, fine-grained comprehension is required for telling such subtle differences.

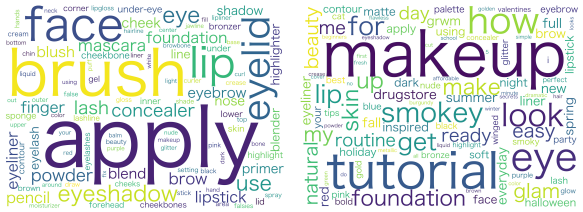


(a) Step caption. (b) Theme.

Figure 6: Wordcloud for different text annotations.

| Type | Terms |
|---|---|
| Actions | apply, use, blend, highlight, draw, curler, fix, clean, pat, shape, emphasize, press, sweep, correct, bake, tap, contour |
| Products | eyeshadow, concealer, powder, pencil, foundation, lipstick, eyeliner, blush, primer, shadow, highlighter, contour, bronzer, gel, cream, lipgloss, false lashes, blam, spray |
| Areas | face, lips, eyelid, eye, lashes, brow, cheek, nose, under-eye, cheekbone, lash line, forehead, chin, mouth, blemish, temple, jaw, hairline, jawline, t-zone, nosebone, browbone, eye corner, lash line, philtrum, eyeball |
| Tools | brush, blender, sponge, finger, puff, tweezer |

Table 2: Top words of four categories in step captions.

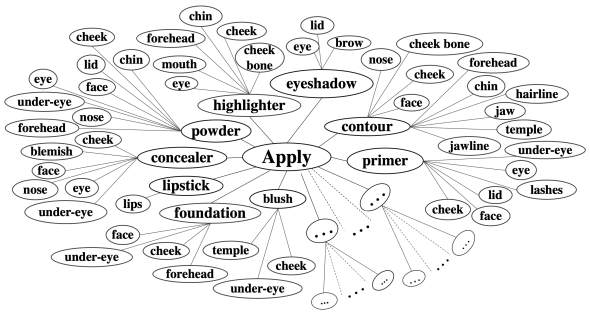


Figure 7: Fine-grained activity graph stemmed from action 'Apply'.

## 4 Semantic Comprehension Tasks

For the purpose of exploring fine-grained semantic comprehension in different aspects and levels, we propose two groups of tasks namely generation and question answering, which contain 5 detailed tasks according to the instructional and multimodal characteristics of YouMakeup.

### 4.1 Task I: Step Generation

The step generation task includes identifying the temporal boundaries for steps, grounding the facial areas and generating natural language step description for each step as shown in Figure 1. It is a classical task in instructional video analysis for evaluating model's ability to capture the temporal flow of steps and grasp the procedural knowledge. Due to the similarity between makeup activities such as the similar motion, appearance of products and the adjacent location of facial areas, this task calls for a fine-grained comprehension of video content. The task is closely related to dense video captioning (Krishna et al., 2017a), however, it needs to integrate makeup procedural knowledge to generate fine-grained step segmentation and step description instead of independent event descriptions.

### 4.2 Task II: Facial Image Ordering

The facial image ordering task is to sort a series of shuffled facial images into the right order according to the ordered step captions, as shown in Figure 8(a). Instructional videos present steps for accomplishing a certain task. Tracking the changes of object is crucial for procedure comprehension. The effect of makeup is the fine-grained changes of facial appearances. Some steps bring apparent changes, for example, "Apply red lipsticks on the lips" turns the lip color from nude to bold red. However, some changes can be subtle and difficult to identify, such as "Apply foundation on the face with brush", which may result in subtle changes of the skin tone. Under most circumstances, the result brought by each step is very complex, which not only depends on the current step but also relies on the prior state of the facial appearance. Tracking changes on the face is an effective way to evaluate models' fine-grained comprehension ability.

We choose five facial images of different steps from a video randomly to form a question. Then we set their original order as the ground truth and the other three random sorts as the candidate answers. We finally generate 12,000 questions, including 8,400 for training, 1,200 for validation, and 2,400 for testing.

### 4.3 Task III: Step Ordering

The step ordering task evaluates the capability of sorting a series of step captions into the right order according to the video. For human beings, comprehension is the prerequisite of ordering. The step ordering task aims at developing model's comprehension ability in a multimodal scenario, which calls for a joint understanding across different modalities. Models need to align natural language step captions with video content to solve the problem as shown in Figure 8(b), fine-grained understanding in both text and video content for temporal action reasoning and visual semantic matching is required.

We select videos with more than four steps to generate questions. For each question, we provide five captions of different steps from the video and four different order choices as the candidate

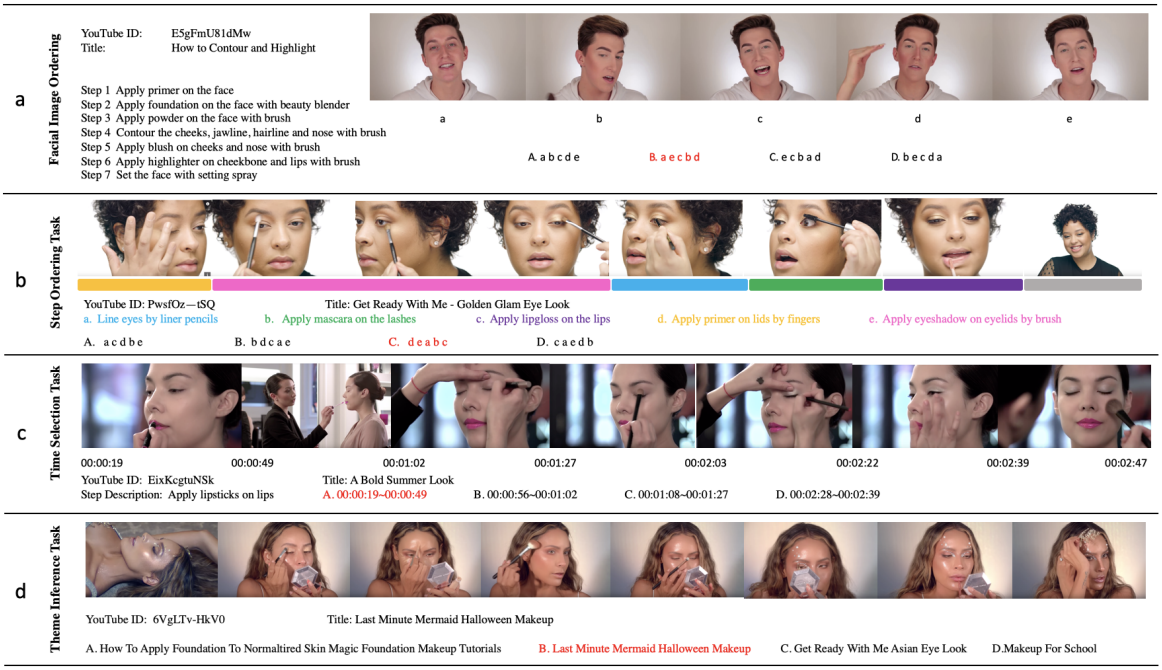


Figure 8: Four VQA tasks on YouMakeup Dataset. Best viewed in color.

answers, including the original one as the ground truth. We finally collect 12,000 questions for this task, including 8,400 for training, 1,200 for validation, and 2,400 for testing.

### 4.4 Task IV: Time Range Selection

In the time range selection task, models are required to localize a specific step in the video accurately. We set the task in QA form where both the video, the step caption and four candidate answers are provided. As the case shown in Figure 8(c), given the caption, the model needs to select the most accurate time range from four candidate answers after reviewing the whole video. To accomplish this task, models need to jointly understand the step caption and the video, distinguish the fine-grained difference of the makeup activities in each video clips to find the best answer.

To form the question, we first select one step from the video, provide its step caption as question and set its time range as ground truth. Then we choose the other three steps which have some overlap in facial area with the question step from the same video, and set their time range as the candidate answers. We finally collect 12,000 questions for this task, including 8,400 for training, 1,200 for validation and 2,400 for testing.

### 4.5 Task V: Theme Inference

The theme inference task requires to inference the theme of a video. People can compress complicated content and extract key information. Several tasks have been set up to help the model develop such ability, such as the video summarization task which summarizes the video in several sentences. Theme inference is another way to evaluate this kind of ability. Since a video can be summarized with focus on multiple aspects, it is difficult to treat theme inference as generation task. We set it in QA form based on the annotation in Section 3.1.3.

As shown in Figure 8(d), we provide four candidate answers including the ground-truth. To generate candidate answers, we set up the candidate answer set with titles and tags of all videos. Then we utilize the FastText (Bojanowski et al., 2016), Doc2Vec (Le and Mikolov, 2014) and Word2Vec (Tomas Mikolov, 2013) together for theme feature representation to search the top 10 nearest themes for each video. These themes are then selected by the annotator to form the candidate answers. Given the makeup videos, the model needs to select the most suitable theme from four choices.

Different from the video classification task which divides videos into fixed pre-defined categories, the candidate answers in the theme inference task are natural language sentences different from each other. As shown in Figure 6(b), the

themes involve diverse aspects, such as "smokey" and "natural" for makeup style, "night" and "halloween" for occasions, and "pink" and "red" for color tone. Thus, the theme inference task requires a fine-grained comprehensive understanding of the video content in various aspects.

## 5 Experiment

We build a step caption generation system to provide a baseline for our YouMakeup dataset and demonstrate the necessity of fine-grained spatial and temporal understanding to solve the task. The system utilizes the groundtruth step segmentation in order to evaluate the captioning ability alone, and is evaluated by the standard captioning metrics including BLEU (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2014), ROUGE (Flick, 2004), CIDEr (Vedantam et al., 2015) and SPICE (Anderson et al., 2016).

**Implementation Details** Our step caption generation model is based on the encoder-decoder captioning framework, which is widely used in video caption generation (Vinyals et al., 2015; Chen et al., 2019). The encoder converts the video clip into a fixed-dimensional vector and then the decoder generates word sequences conditioning on the encoded vector. Since both spatial and temporal reasoning are important for the task, we propose two types of encoder as follows: 1) spatial encoder: faster RCNN (Ren et al., 2015b) pretrained on the VisualGenome dataset (Krishna et al., 2017b) is used to extract object features in a single frame. We select one frame for each video clip and detect at most 36 objects in the frame. Mean pooling is applied on the extracted object features to generate the video-level representation. 2) temporal encoder: Resnet152 (He et al., 2016) pretrained on the ImageNet dataset (Deng et al., 2009) is used to extract features for each frame. We extract global frame-level features for every 16 frames and apply mean pooling on the temporal dimension to generate the global video-level representation. We employ the LSTM as our decoder, which contains 1 hidden layer with 512 hidden units. Adam optimizer is used to train our model with batch size of 128 and learning rate of 0.0001. We train at most 100 epochs and select the best model according to captioning performance on the validation set.

**Results.** Table 3 presents the step captioning performance with ground truth step segmentation on the testing set of YouMakeup. We can see that the combination of spatial and temporal features achieves the best performance, demonstrating spatial and temporal information is complementary to generate step captions. The captioning model based on the spatial feature alone is inferior to that based on the temporal feature because our framework does not utilize all clip information. The overall performance on Cider of the baseline step caption generation model is relatively higher than baselines on other video captioning datasets (Krishna et al., 2017a) due to the fine-grained characteristics of the proposed dataset. In the YouMakeup dataset, the styles of step captions are similar which can make it easy for the caption generation system to achieve high scores since evaluation metrics are not aware of the aspect importance in the caption such as detailed tools, actions etc.

Table 3: Step captioning performance with groundtruth step segmentation on the YouMakeup dataset.

| | B@4 | Meteor | Rouge | Cider | Spice |
|---|---|---|---|---|---|
| spatial | 13.1 | 19.7 | 46.3 | 94.6 | 27.1 |
| temporal | 16.9 | 21.9 | 48.8 | 130.8 | 34.3 |
| **spatial+temporal** | **17.9** | **22.4** | **50.3** | **134.7** | **34.5** |

Although the generation system achieves high evaluation scores, we find it fails to capture the fine-grained details in the makeup instructional videos. Figure 9 shows the caption generation results of the baseline system using both temporal and spatial features on a specific video for the first 6 steps. According to the captions of step 1 and 3, the system traces out the procedure roughly, lacking of details such as makeup tools and related facial area. The other three step captions indicate the system's weak ability on fine-grained video comprehension and confirm the fine-grained characteristics of YouMakeup. In step 2, the system mistakes color correction palette for eyeshadow due to their similar appearance. However, the eyeshadow is applied around eyes while color correction palette is used on the face for color correcting. From step 4 to 6, system shows confusion about the procedure of applying foundation and concealer because they are similar in both appearance and usage. System needs to associate products with their usage methods and facial areas they are applied in order to grasp the subtle difference between different makeup activities for generating accurate step description.



| Step | Generated Caption | Ground Truth |
|---|---|---|
| Step 1 | apply primer on face | apply the primer on face with fingers |
| Step 2 | apply eyeshadow on eyelid with brush | apply the color correction palette on face with fingers |
| Step 3 | apply concealer on face with sponge | apply the concealer with sponge on one side of the face |
| Step 4 | apply foundation on face with sponge | blend the concealer with brush on another side of the face |
| Step 5 | apply foundation on face | apply the foundation on skin with brush |
| Step 6 | apply concealer on face with sponge | apply the foundation on face with beauty sponge |

Figure 9: Comparison of generated captions and ground-truth captions for different steps. Best viewed in color.

## 6 Conclusion

In this paper, we introduce a new large-scale instructional video dataset named YouMakeup for fine-grained semantic comprehension. The YouMakeup dataset contains 2,800 makeup instructional videos spanning more than 420 hours in total. Based on the characteristics of makeup instructional videos and the rich annotations of temporal boundaries, grounded facial areas and natural language descriptions of steps, our collected dataset is more suitable to support the fine-grained video comprehension research than previous datasets. We further design a generation task and four question answering tasks to thoroughly evaluate the fine-grained semantic comprehension ability from different aspects and levels. A baseline system for step caption generation also demonstrates the necessity of fine-grained spatial and temporal information. In the future work, we plan to make thorough exploration on these proposed tasks. We will make the dataset publicly accessible in order to support the research investigation in fine-grained semantic comprehension.

## Acknowledgments

This work was supported by National Natural Science Foundation of China (No. 61772535), Beijing Natural Science Foundation (No. 4192028), and National Key Research and Development Plan (No. 2016YFB1001202). We would like to thank our group member Jingjun Liang for his help in building the annotation website and all the annotators for their careful annotations.

## References

Jean-Baptiste Alayrac, Piotr Bojanowski, Nishant Agrawal, Josef Sivic, Ivan Laptev, and Simon Lacoste-Julien. 2016. Unsupervised learning from narrated instruction videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4575–4583.

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2016. Spice: Semantic propositional image caption evaluation. *Adaptive Behavior*, 11(4):382–398.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Shizhe Chen, Yuqing Song, Yida Zhao, Qin Jin, Zhaoyang Zeng, Bei Liu, Jianlong Fu, and Alexander Hauptmann. 2019. Activitynet 2019 task 3: Exploring contexts for dense captioning events in videos. *arXiv preprint arXiv:1907.05092*.

Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. 2018. Scaling egocentric vision: The epic-kitchens dataset. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 720–736.

Pradipto Das, Chenliang Xu, Richard F Doell, and Jason J Corso. 2013. A thousand frames in just a few words: Lingual description of videos through latent topics and sparse object stitching. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2634–2641.

Jia Deng, Wei Dong, Richard Socher, Li Jia Li, Kai Li, and Fei Fei Li. 2009. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision Pattern Recognition*.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 376–380. Association for Computational Linguistics.

Carlos Flick. 2004. Rouge: A package for automatic evaluation of summaries. In *Workshop on Text Summarization Branches Out*.

Meera Hahn, Nataniel Ruiz, Jean-Baptiste Alayrac, Ivan Laptev, and James M Rehg. 2018. Learning to localize and align fine-grained actions to sparse instructions. *arXiv preprint arXiv:1809.08381*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Sun Jian. 2016. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision Pattern Recognition*.

Yunseok Jang, Yale Song, Youngjae Yu, Youngjin Kim, and Gunhee Kim. 2017. Tgif-qa: Toward spatio-temporal reasoning in visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2758–2766.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2901–2910.

K Kim, C Nan, MO Heo, SH Choi, and BT Zhang. 2016. Pororoqa: Cartoon video series dataset for story understanding. In *Proceedings of NIPS 2016 Workshop on Large Scale Computer Vision System*, volume 15.

Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. 2017a. Dense-captioning events in videos. In *The IEEE International Conference on Computer Vision (ICCV)*.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Li Jia Li, Li Jia Li, and David A. Shamma. 2017b. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123(1):32–73.

Hilde Kuehne, Ali Arslan, and Thomas Serre. 2014. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196.

Jie Lei, Licheng Yu, Mohit Bansal, and Tamara L Berg. 2018. Tvqa: Localized, compositional video question answering. In *EMNLP*.

Jie Lei, Licheng Yu, Tamara L Berg, and Mohit Bansal. 2019. Tvqa+: Spatio-temporal grounding for video question answering. *arXiv preprint arXiv:1904.11574*.

Tegan Maharaj, Nicolas Ballas, Anna Rohrbach, Aaron Courville, and Christopher Pal. 2017. A dataset and exploration of models for understanding video data through fill-in-the-blank question-answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6884–6893.

Mateusz Malinowski and Mario Fritz. 2014. A multi-world approach to question answering about real-world scenes based on uncertain input. In *Advances in neural information processing systems*, pages 1682–1690.

Taichi Nishimura, Atsushi Hashimoto, Yoko Yamakata, and Shinsuke Mori. 2019. Frame selection for producing recipe with pictures from an execution video of a recipe. In *Proceedings of the 11th Workshop on Multimedia for Cooking and Eating Activities*, pages 9–16. ACM.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Mengye Ren, Ryan Kiros, and Richard Zemel. 2015a. Exploring models and data for image question answering. In *Advances in neural information processing systems*, pages 2953–2961.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015b. Faster r-cnn: towards real-time object detection with region proposal networks.

Marcus Rohrbach, Sikandar Amin, Mykhaylo Andriluka, and Bernt Schiele. 2012. A database for fine grained activity detection of cooking activities. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1194–1201. IEEE.

Sebastian Stein and Stephen J McKenna. 2013. Combining embedded accelerometers with computer vision for recognizing food preparation activities. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pages 729–738. ACM.

Yansong Tang, Dajun Ding, Yongming Rao, Yu Zheng, Danyang Zhang, Lili Zhao, Jiwen Lu, and Jie Zhou. 2019. Coin: A large-scale dataset for comprehensive instructional video analysis. *arXiv preprint arXiv:1903.02874*.

Makarand Tapaswi, Yukun Zhu, Rainer Stiefelhagen, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2016. Movieqa: Understanding stories in movies through question-answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4631–4640.

Kai Chen Greg Corrado Jeffrey Dean Tomas Mikolov, Ilya Sutskever. 2013. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*.

Sam Toyer, Anoop Cherian, Tengda Han, and Stephen Gould. 2017. Human pose forecasting via deep markov models. In *2017 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–8. IEEE.

Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *CVPR*.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *IEEE Conference on Computer Vision  Pattern Recognition*.

Huijuan Xu, Abir Das, and Kate Saenko. 2017. R-c3d: Region convolutional 3d network for temporal activity detection. In *Proceedings of the IEEE international conference on computer vision*, pages 5783–5792.

K. Zhang, Z. Zhang, Z. Li, and Y. Qiao. 2016. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503.

Luowei Zhou, Chenliang Xu, and Jason J Corso. 2018. Towards automatic learning of procedures from web instructional videos. In *AAAI Conference on Artificial Intelligence*, pages 7590–7598.

# DEBUG: A Dense Bottom-Up Grounding Approach for Natural Language Video Localization

**Chujie Lu**[1*] **Long Chen**[1*] **Chilie Tan**[2] **Xiaolin Li**[2,3] **Jun Xiao**[1†]
[1]DCD Lab, College of Computer Science, Zhejiang University, China
[2]Tongdun Technology, China [3]University of Florida, USA
chujielu@outlook.com, {longc, junx}@zju.edu.cn
chilie.tan@tongdun.net, andyli@ece.ufl.edu

## Abstract

In this paper, we focus on natural language video localization: localizing (*i.e.*, grounding) a natural language description in a long and untrimmed video sequence. All currently published models for addressing this problem can be categorized into two types: (i) top-down approach: it does classification and regression for a set of pre-cut video segment candidates; (ii) bottom-up approach: it directly predicts probabilities for each video frame as the temporal boundaries (*i.e.*, start and end time point). However, both two approaches suffer several limitations: the former is computation-intensive for densely placed candidates, while the latter has trailed the performance of the top-down counterpart thus far. To this end, we propose a novel *dense bottom-up* framework: DEnse Bottom-Up Grounding (DEBUG). DEBUG regards all frames falling in the ground truth segment as foreground, and each foreground frame regresses the unique distances from its location to bi-directional ground truth boundaries. Extensive experiments on three challenging benchmarks (TACoS, Charades-STA, and ActivityNet Captions) show that DEBUG is able to match the speed of bottom-up models while surpassing the performance of the state-of-the-art top-down models.

## 1 Introduction

Vision-and-language understanding, *e.g.*, what the vision and text are, and how they relate with each other, is one of the core tasks in both computer vision and natural language processing. To test the machine comprehension of complex video scene and natural language simultaneously, a challenging task was proposed (Gao et al., 2017; Hendricks et al., 2017, 2018), called **Natural Language Video Localization** (NLVL). As shown in Figure 1, given a natural language description query and an untrimmed video sequence, NLVL needs to localize a segment in the video (*i.e.*, identify the start and end time point of the segment) which semantically corresponds to the reference sentence. Moreover, NLVL is an indispensable technique for many important applications, *e.g.*, text-oriented video highlight detection or retrieval.

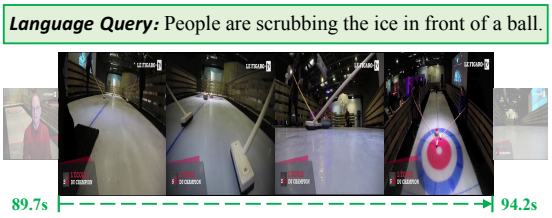


Figure 1: Natural language video localization is to localize a segment with the start point (89.7s) and end point (94.2s) in the video given a language description.

Currently, the overwhelming majority of NLVL models are ***top-down*** approaches: they first cut a video into a set of segment candidates, then they do classification and regression for each candidate. Specifically, they can be further grouped into two sub-types: 1) *sliding-window-based* (Gao et al., 2017; Hendricks et al., 2017; Liu et al., 2018b,a; Ge et al., 2019; Chen and Jiang, 2019; Xu et al., 2019; Zhang et al., 2019b): the video is explicitly segmented by multiple predefined temporal sliding-window scales. After extracting features for the query and all candidates, NLVL degrades into a multimodal matching problem. 2) *anchor-based* (Chen et al., 2018; Zhang et al., 2019a): it assigns each frame[1] with multi-scale temporal anchors, which follows the same spirit of anchor box in object detection (Ren et al., 2015).

Although top-down approaches have dominated NLVL for years, it is worth noting that they suffer

[*] Chujie Lu and Long Chen are co-first authors with equal contributions.

[†] Jun Xiao is the corresponding author.

[1]The frame is a general description for a frame in a video sequence or an element in a video frame feature sequence.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5144–5153, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

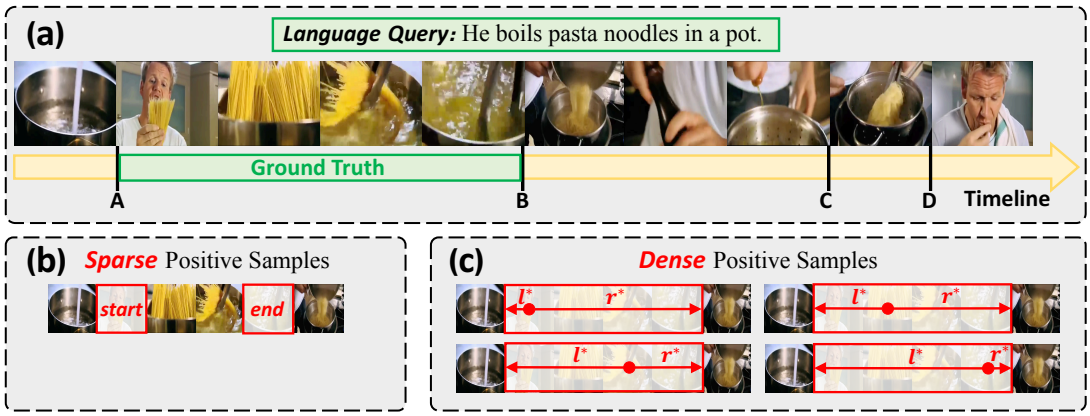


Figure 2: (a): A NLVL sample with ground truth segment (A→B). (b): Sparse positive samples in existing sparse bottom-up models (*i.e.*, two frames). (c): Dense positive samples in DEBUG (*i.e.*, all frames in range (A→B)).

several notorious limitations: 1) The performance is sensitive to the heuristic rules (*e.g.*, the temporal scales or the number of candidates). 2) In order to achieve a high recall, they are required to densely place candidates, which significantly increase the amount of computation and localization time.

To eliminate these inherent drawbacks in top-down framework, some recent NLVL works (Chen et al., 2019a; Yuan et al., 2019) start to borrow idea from reading comprehension (Xiong et al., 2017, 2018; Yu et al., 2018), which directly predicts the start and end boundaries. Although this ***sparse bottom-up*** approach is highly computation-efficient, the localization accuracy, especially for long videos (*e.g.*, TACoS) is behind its top-down counterpart. We argue that the main reasons come from three-fold: 1) Two boundary predictions are independent, *i.e.*, the model ignores the content consistency between two predictions. As an example shown in Figure 2 (a), two frames in B and D have a similar visual appearance. Thus, the model is prone to predict the result as (A→D), without considering the distinct content change in range (B→C). 2) The positive and negative training samples are extremely imbalanced. Since the number of video frames is large (*e.g.*, each video in TACoS has average 9,000 frames), but the positive training samples are *sparse*, *i.e.*, only two frames (Figure 2 (b)). 3) Detecting temporal action boundary from frames, *i.e.*, predicting a frame is query-related and at temporal boundary simultaneously by a single network, is still a challenging task, even without query constraint (Shou et al., 2018).

In this paper, we propose a ***dense bottom-up*** framework for NLVL: DEnse Bottom-Up Grounding (DEBUG), to mitigate the problems in existing NLVL frameworks. Specifically, we regard all frames falling in the ground truth segment as positive samples (*i.e.*, foreground). For each positive frame, DEBUG has a classification subnet to predict its relatedness with the query, and a boundary regression subnet to regress the unique distances from its location to bi-directional ground truth boundaries. This design helps to disentangle the temporal boundary detection from query-related prediction, relieving the burden of the single classification network in existing sparse bottom-up models. Meanwhile, we can utilize as many positive samples as possible to alleviate the imbalance problem between positive and negative samples (Figure 2 (c)). Since each pair of boundary predictions are based on the same frame feature, *i.e.*, two predictions act as a whole, which helps to avoid falling into the local optimum caused by independent predictions. In addition, we propose a temporal pooling to relieve unstable performance caused by single prediction. Moreover, DEBUG is agnostic to the upstream multimodal interaction network, *i.e.*, it can be seamlessly incorporated into any stronger backbone to boost performance.

We demonstrate the effectiveness of DEBUG on three challenging benchmarks: TACoS (Regneri et al., 2013), Charades-STA (Gao et al., 2017), and ActivityNet Captions (Krishna et al., 2017). Without bells and whistles, DEBUG surpasses the performance of the state-of-the-art models over various benchmarks and metrics at the highest speed.

## 2 Related Work

### 2.1 Natural Language Video Localization

NLVL is a very difficult task, which needs to understand both complex video scene and natu-

ral language simultaneously. Because most of NLVL models are under the top-down framework, they focus on designing more effective multi-modal interaction networks, *e.g.*, query-based attention on video frames (Liu et al., 2018a), visual-based attention on language words (Liu et al., 2018b), or co-attention between each frame-and-word pairs (Chen et al., 2018, 2019a; Yuan et al., 2019). It is worth noting that the improvement in multimodal interaction network is orthogonal to the DEBUG, *i.e.*, DEBUG can be seamlessly incorporated into any stronger interaction network.

To the best of our knowledge, there are only two exceptions among all NLVL models: RWM (He et al., 2019) and SM-RL (Wang et al., 2019), which are not under either top-down or bottom-up frameworks. They both formulate NLVL as a sequential decision making problem, solved by reinforcement learning, *e.g.*, actor critic (Chen et al., 2019b). The action space for each step is a set of handcraft-designed temporal box transformations.

### 2.2 Top-Down vs. Bottom-Up

Top-down and bottom-up approaches, which are widely co-exist in many CV and NLP tasks, are two different philosophical viewpoints for solving problems. The most related top-down and bottom-up concepts as the one in NLVL frameworks are:

**Object Detection.** Most of the object detectors after Faster-RCNN (Ren et al., 2015) are top-down models, *i.e.*, it predicts classification scores and regression offsets for multiple predefined anchors in each position. These models suffer the same drawbacks as above mentioned in the top-down approach for NLVL. However, with the advent of the first performance comparable bottom-up object detector: CornerNet (Law and Deng, 2018), the bottom-up approaches begin to gain unprecedented attention (Zhou et al., 2019b,a; Duan et al., 2019; Tian et al., 2019), which inspires us to explore a decent bottom-up framework for NLVL.

**Attention Mechanism.** Top-down attention has dominated many vision-and-language tasks, *e.g.*, visual captioning (Xu et al., 2015; Chen et al., 2017), visual QA (Xu and Saenko, 2016; Ye et al., 2017). Recently, a model combining both top-down and bottom-up attention reaches the winner of multiple challenges (Anderson et al., 2018). Thus, how to combine the top-down and bottom-up attentions effectively is still an unexplored problem in the vision-and-language tasks.

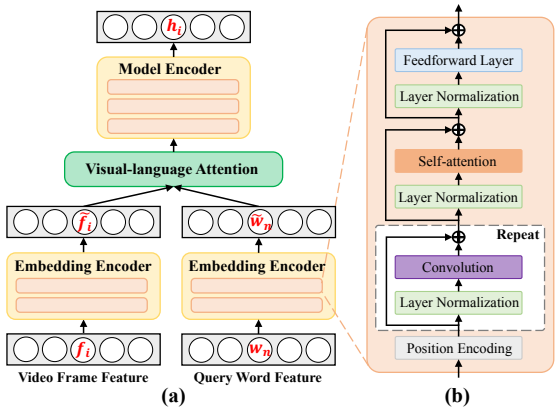


Figure 3: (a): The overview of QANet backbone. (b): The encoder block throughout the QANet.

## 3 Approach

The NLVL task considered in this paper, is defined as follows. Given a long and untrimmed video sequence $V$, and a natural language query $Q$ which describes a segment in $V$ from start time point $t_s$ to end time point $t_e$, NLVL needs to predict these two time points $(t_s, t_e)$ given $V$ and $Q$.

In this section, we first introduce the multi-modal interaction backbone of DEBUG, which is built upon the recently proposed QANet (Yu et al., 2018) for reading comprehension (Section 3.1). Then, we demonstrate the details about the proposed dense bottom-up grounding (Section 3.2). Finally, we describe the training and test stage of the whole DEBUG (Section 3.3).

### 3.1 Backbone: QANet

We adopt the QANet to model the interaction between two different modalities (*i.e.*, video and language), which serves as the backbone of DEBUG. The details of the QANet are shown in Figure 3 (a), which consists of three main components:

**Embedding Encoder Layer.** The input for this layer in two different branches are extracted video frame features $\boldsymbol{F} = \{\boldsymbol{f}_i\}_{i=1}^{T}$ and query word features $\boldsymbol{W} = \{\boldsymbol{w}_n\}_{n=1}^{N}$, respectively (see details in Section 4.2). $T$ and $N$ are the numbers of frames and words. The embedding encoder layer is a stack of encoder blocks as shown in Figure 3 (b), which contains multiple components, including convolutional layer, layer-normalization layer, self-attention layer, and feedforward layer. The output of this layer is new frame features $\tilde{\boldsymbol{F}} = \{\tilde{\boldsymbol{f}}_i\}_{i=1}^{T}$ or word features $\tilde{\boldsymbol{W}} = \{\tilde{\boldsymbol{w}}_n\}_{n=1}^{N}$, which encode the context in its respective modality.

**Visual-language Attention Layer.** It calculates

two different attention weights between the two modal features. Specifically, it first computes a similarity matrix $\boldsymbol{S} \in \mathbb{R}^{T\times N}$, where $\boldsymbol{S}_{ij}$ denotes the similarity between frame feature $\tilde{\boldsymbol{f}}_i$ and word feature $\tilde{\boldsymbol{w}}_j$. Then the two attention weights are:

$$\boldsymbol{A} = \bar{\boldsymbol{S}} \cdot \tilde{\boldsymbol{W}}, \quad \boldsymbol{B} = \bar{\boldsymbol{S}} \cdot \bar{\bar{\boldsymbol{S}}}^T \cdot \tilde{\boldsymbol{F}}, \tag{1}$$

where $\bar{\boldsymbol{S}}$ and $\bar{\bar{\boldsymbol{S}}}$ are the row-wise and column-wise normalized matrix of $S$, respectively.

**Model Encoder Layer.** Given the two attention weights $\boldsymbol{A}$ and $\boldsymbol{B}$, the model encoder layer begins to encode the interaction between the two modal features. This layer is also a stack of encoder blocks (Figure 3 (b)), and these encoder blocks share parameters. The input in $i$-th position is $[\boldsymbol{f}_i, \boldsymbol{a}_i, \boldsymbol{f}_i \odot \boldsymbol{a}_i, \boldsymbol{f}_i \odot \boldsymbol{b}_i]$, where $\boldsymbol{a}_i$ and $\boldsymbol{b}_i$ are $i$-th row of $\boldsymbol{A}$ and $\boldsymbol{B}$, $\odot$ is the element-wise multiplication, and $[\cdot]$ is a vector concatenate operation. The output is $\boldsymbol{H} = \{\boldsymbol{h}_i\}_{i=1}^T, \boldsymbol{H} \in \mathbb{R}^{T\times D}$, where $\boldsymbol{h}_i \in \mathbb{R}^D$ is a frame feature encoded with multimodal context. Then $\boldsymbol{H}$ is fed into the following head network (*i.e.*, dense bottom-up grounding) for boundaries prediction. We refer readers to QANet (Yu et al., 2018) paper for more details.

### 3.2 Dense Bottom-Up Grounding

Since the nature of the bottom-up approach, DEBUG regards each frame as a training sample. Different from the existing sparse bottom-up models which only use the exact start and end boundary frames as foreground, DEBUG utilizes all frames falling in the ground truth segment as positive samples. For each sample, there are three branch subnets, which aim to predict its classification score, boundary distances, and confidence score respectively. Specifically, the whole architecture of DEBUG is shown in Figure 4, and the details about the three branch subnets are:

**Classification Subnet.** The classification subnet predicts the relatedness between each video frame and the language query, *i.e.*, whether the frame is a foreground frame. Taking the multimodal feature $\boldsymbol{H} \in \mathbb{R}^{T\times D}$ from the backbone, this subnet applies four 1×3 conv layers, each with $D$ filters and each followed by ReLU activations, followed by a 1×3 conv layer with 1 filter. Finally, sigmoid activations are attached to output the foreground prediction score per location. For a positive sample (*i.e.*, foreground), its ground truth classification label is $c_i^* = 1$, otherwise $c_i^* = 0$.

**Boundary Regression Subnet.** The boundary regression subnet predicts the unique distances from

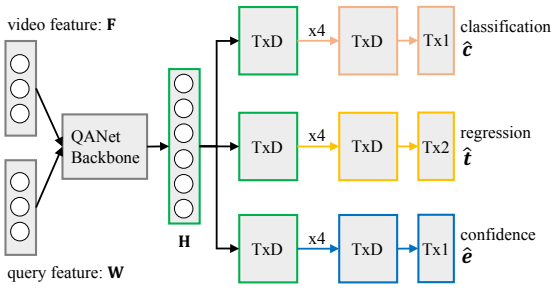


Figure 4: The whole architecture of the DEBUG. It consists of a QANet backbone and a head network with three branch subnets. $T \times *$ denotes the shape of corresponding matrix.

the location of each frame to the bi-directional ground truth boundaries. The design of the boundary regression subnet is identical to the classification subnet except it terminates in 2 outputs for both left and right distances. We only assign boundary regression targets for positive frames. Specifically, for a positive frame at $i$-th position, if the ground truth segment range is $(t_s, t_e)$ (*i.e.*, $t_s \le i \le t_e$), the regression target is $\boldsymbol{t}_i^* = (l_i^*, r_i^*)$:

$$l_i^* = i - t_s, \quad r_i^* = t_e - i, \tag{2}$$

where $l_i^*$ and $r_i^*$ represents the distance from $i$-th frame to the left and right boundaries, respectively.

**Confidence Subnet.** The design of the confidence subnet is identical to the classification subnet, but it predicts the confidence of the boundary regression results for each frame. The motivation of this subnet design comes from that the prediction confidences from different frames should be different, *e.g.*, it is more difficult for a frame near the start point to detect the end point than a frame near the end point. Therefore, we set the ground truth confidence of each frame based on its "centerness" in the segment. Given the regression target $l_i^*$ and $r_i^*$, the ground truth confidence score is defined as:

$$e_i^* = \frac{\min(l_i^*, r_i^*)}{\max(l_i^*, r_i^*)}. \tag{3}$$

The confidence score ranges from 1 to 0 with the frame position from segment center to boundary.

### 3.3 Training and Inference

**Loss.** Given all frames predictions $\{(\hat{c}_i, \hat{\boldsymbol{t}}_i, \hat{e}_i)\}$ and the corresponding ground truth $\{(c_i^*, \boldsymbol{t}_i^*, e_i^*)\}$, the total training loss function for DEBUG is:

$$L = \frac{1}{N}\sum_i L_{cls}(\hat{c}_i, c_i^*) + \frac{\alpha}{N_p}\mathbb{1}_{\{c_i^*=1\}} L_{reg}(\hat{\boldsymbol{t}}_i, \boldsymbol{t}_i^*) + \frac{\beta}{N_p}\mathbb{1}_{\{c_i^*=1\}} L_{conf}(\hat{e}_i, e_i^*) \tag{4}$$

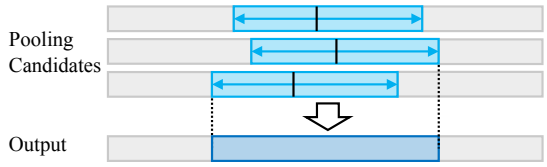


Figure 5: Illustration of temporal pooling.

where $L_{cls}$ and $L_{conf}$ both are binary cross entropy (BCE) loss for classification subnet and confidence subnet. $L_{reg}$ is the IOU loss for boundary regression subnet (*i.e.*, $-\ln \frac{\min(r_i^*,\hat{r}_i)-\max(l_i^*,\hat{l}_i)}{\max(r_i^*,\hat{r}_i)-\min(l_i^*,\hat{l}_i)}$). $N$ and $N_p$ denotes the number of total samples and positive samples, respectively. $\alpha$ and $\beta$ are loss weights to balance different losses, we set both $\alpha$ and $\beta$ to 1 in all experiments. $\mathbb{1}_{\{c_i^*=1\}}$ is an indicator function, being 1 if $c_i^* = 1$ and 0 otherwise.

**Inference.** Given a video and a language query, we forward them through the network and obtain $\hat{c}_i$, $\hat{\boldsymbol{t}}_i$, $\hat{e}_i$ for each frame from three subnets. Then, we rank all segment predictions by the score $\hat{s}$: $\hat{s}_i = \hat{c}_i \times \hat{e}_i$. A straightforward solution is selecting the segment with the highest score as the final prediction. However, segment prediction from a single frame is usually with high variance. To relieve this situation, we propose a simple yet effective strategy: **Temporal Pooling**, to fuse multiple frame predictions. As shown in Figure 5, temporal pooling directly uses the leftmost and rightmost boundaries among all pooling candidates as its output. As for the pooling candidates, they need to meet with two conditions simultaneously: 1) the predicted segment is overlapped with the one with the highest score; 2) the score is large than the highest score multiple a threshold $\delta$[2].

# 4 Experiments

## 4.1 Datasets and Metrics

**Datasets.** We evaluated the DEBUG on three challenging NLVL benchmarks: **TACoS** (Regneri et al., 2013). It consists of 127 videos and 17,344 text-to-clip pairs. In our experiments, we used the standard split same as (Gao et al., 2017), *i.e.*, 50% for training, 25% for validation and 25% for test. The average length of each video is 5 minutes. **Charades-STA** (Gao et al., 2017). It consists of 12,408 text-to-clip pairs for training, and 3,720 pairs for test. The average length of each video is 30 seconds. **ActivityNet Captions** (Krishna et al., 2017). It is not only the largest NLVL dataset (19,209 videos) but also with much more diverse context than the others. We followed (Yuan et al., 2019) and utilized the public train set (*i.e.*, 37,421 text-to-clip pairs) for training, and the validation set (*i.e.*, 17,505 text-to-clip pairs) for test. The average length of each video is 2 minutes.

**Evaluation Metrics.** Following the conventions in previous works, we evaluated NLVL on two prevailing evaluation metrics: 1) **R@N, IoU@$\theta$**: The percentage of testing samples which have at least one of the top-N results with IoU larger than $\theta$. Since the nature of the bottom-up framework, we only use $N$=1 in all our experiments; 2) **mIoU**: The average IoU over all testing samples.

## 4.2 Implementation Details

Given an untrimmed video $V$, we first downsampled frames and utilized the C3D (D.Tran et al., 2015) feature pretrained on Sports-1M (A.Karpathy et al., 2014) as the initial frame features. Then, we reduced the dimension of these features to 500 using PCA, which are the video frame features $\boldsymbol{F}$ (Section 3.1). For query $Q$, it was truncated or padded to a maximum length of 15 words. Each word was initialized with the 300-d Glove vector (J.Pennington et al., 2014), and all word embeddings were fixed. Then we learned a transformation matrix to map these embeddings to 500-d, which are the sentence word features $\boldsymbol{W}$ (Section 3.1). The dimension of all intermediate layers in the backbone and three subnets was set to 128. We trained the whole network for 100 epochs from scratch, and the loss was optimized by Adam algorithm (D.P.Kingma and J.Ba, 2015). The learning rate started from 0.0001 and was divided by 10 when the loss plateaus. The batch size was set to 16, and the dropout rate was 0.5.

## 4.3 Comparisons with State-of-the-Arts

**Settings.** We compared the DEBUG with all recently published state-of-the-art NLVL models. From the viewpoint of top-down and bottom-up framework, we group them into: 1) Sliding-window-based models: **VSA-RNN**, **VSA-STV**, **CTRL** (Gao et al., 2017), **ROLE** (Liu et al., 2018b), **ACRN** (Liu et al., 2018a), **MCF** (Wu and Han, 2018), **ACL** (Ge et al., 2019), **SAP** (Chen and Jiang, 2019), **QSPN** (Xu et al., 2019). 2) Anchor-based models: **TGN** (Chen et al., 2018). 3) Sparse bottom-up models: **L-Net** (Chen et al., 2019a), **ABLR-af**, **ABLR-aw** (Yuan et al., 2019)

[2]In experiments, we tested $\delta \in \{0.1, 0.2, ..., 0.9\}$ and selected the one with the highest performance for each dataset.

| | Method | IoU@0.1 | IoU@0.3 | mIoU |
|---|---|---|---|---|
| TACoS | VSA-RNN | 8.84 | 6.91 | - |
| | VSA-STV | 15.01 | 10.77 | - |
| | CTRL | 24.32 | 18.32 | - |
| | ACRN | 24.22 | 19.52 | - |
| | MCF | 25.84 | 18.64 | - |
| | SM-RL | 26.51 | 20.25 | - |
| | ACL | 28.31 | 22.07 | - |
| | SAP | 31.15 | - | - |
| | L-NET$^{\diamond}$ | - | - | 13.41 |
| | ABLR-aw$^{\diamond}$ | 31.60 | 18.90 | 12.50 |
| | ABLR-af$^{\diamond}$ | 34.70 | 19.50 | 13.40 |
| | **DEBUG**$^{*}$ | 35.22 | 22.07 | 14.56 |
| | **DEBUG** | **41.15** | **23.45** | **16.03** |

| | Method | IoU@0.3 | IoU@0.5 | IoU@0.7 |
|---|---|---|---|---|
| Charades-STA | VSA-RNN | - | 10.50 | 4.32 |
| | VSA-STV | - | 16.91 | 5.81 |
| | CTRL | - | 23.63 | 8.89 |
| | ROLE | 25.26 | 12.12 | - |
| | ACL | - | 26.47 | 11.23 |
| | SAP | - | 27.42 | 13.36 |
| | RWM | - | 36.70 | - |
| | SM-RL | - | 24.36 | 11.17 |
| | QSPN | 54.70 | 35.60 | 15.80 |
| | **DEBUG**$^{*}$ | 52.16 | 35.89 | **17.92** |
| | **DEBUG** | **54.95** | **37.39** | 17.69 |

| | Method | IoU@0.3 | IoU@0.5 | mIoU |
|---|---|---|---|---|
| ActivityNet | TGN | 43.81 | 27.93 | - |
| | QSPN | 45.30 | 27.70 | - |
| | RWM | - | 36.90 | - |
| | ABLR-af$^{\diamond}$ | 53.65 | 34.91 | 35.72 |
| | ABLR-aw$^{\diamond}$ | 55.67 | 36.79 | 36.99 |
| | **DEBUG**$^{*}$ | 55.82 | 39.20 | 39.01 |
| | **DEBUG** | **55.91** | **39.72** | **39.51** |

Table 1: Performance (%) over R@1,IoU@$\theta$ and mIoU compared with the state-of-the-art NLVL models on TACoS, Charades-STA and ActivityNet Captions. $^{\diamond}$ denotes the models that are under bottom-up framework. $^{*}$ denotes the DEBUG which is without temporal pooling. The **best** and second best methods under each setting are marked in according formats.

| Dataset | Metric | QANet-SE | DEBUG |
|---|---|---|---|
| TACoS | IoU@0.1 | 29.54 | **41.15** |
| | IoU@0.3 | 16.75 | **23.45** |
| | IoU@0.5 | 9.57 | **11.72** |
| | mIoU | 12.01 | **16.03** |
| Charades-STA | IoU@0.3 | 50.81 | **54.95** |
| | IoU@0.5 | 32.63 | **37.39** |
| | IoU@0.7 | 16.24 | **17.69** |
| | mIoU | 33.94 | **36.34** |
| ActivityNet | IoU@0.1 | 71.90 | **74.26** |
| | IoU@0.3 | 53.44 | **55.91** |
| | IoU@0.5 | 38.04 | **39.72** |
| | mIoU | 38.02 | **39.51** |

Table 2: Performance (%) over R@1,IoU@$\theta$ and mIoU compared with QANet-SE on TACoS, Charades-STA and ActivityNet Captions.

4) Others (*i.e.*, RL-based models): **RWM** (He et al., 2019), **SM-RL** (Wang et al., 2019).

**Results.** The results are reported in Table 1. From Table 1, we can observe that the DEBUG achieves a new state-of-the-art performance under all evaluation metrics and benchmarks. It is worth noting that DEBUG can especially improve the performance significantly in some more strict metrics (*e.g.*, 2.62% and 2.52% absolute improvement in mIoU on dataset TACoS and ActivityNet Captions, and 2.12% absolute improvement in IoU@0.7 on Charades-STA[3]), which demonstrates the effectiveness of the DEBUG.

### 4.4 Ablative Studies

In this section, we did extensive ablative experiments to thoroughly investigate the DEBUG.

#### 4.4.1 Sparse vs. Dense Bottom-Up

**Setting.** To eliminate the influence of backbones and equally investigate the performance gain between DEBUG and the existing sparse bottom-up framework, we designed a strong baseline dubbed as **QANet-SE**. Its backbone is identical to DEBUG (Figure 3), but its head network follows the sparse bottom-up framework, *i.e.*, it predicts the start and end time points directly.

**Results.** The results are reported in Table 2. We can observe that DEBUG surpasses QANet-SE over all metrics and benchmarks. Especially, the performance gains are much more obvious in TACoS (*e.g.*, over 20%∼40% relative improvements in all metrics). This is because the average length of each video in TACoS is largest among all benchmarks, and the QANet-SE style (*i.e.*, sparse bottom-up) method suffers severe positive and negative samples imbalance in long videos. Instead, DEBUG can relieve this problem by utilizing much more positive training samples.

#### 4.4.2 Importance of Each Component

We ran a number of experiments to analyze the importance of each component in DEBUG. Results are shown in Table 3 and discussed in detail next.

**Classification vs. Confidence Subnet.** From Table 3, we can observe that models with only a single classification or confidence subnet can get comparable performance. More precisely, the latter one is slightly better than the former. This is

[3]Since all published works had not reported their mIoU scores on Charades-STA, we only compare with them on the metric IoU@$\theta$.

| Datasets | | | TACoS | | | | Charades-STA | | | | ActivityNet | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CLS | CFD | TP | IoU@$\theta$ 0.1 | IoU@$\theta$ 0.3 | IoU@$\theta$ 0.5 | mIoU | IoU@$\theta$ 0.3 | IoU@$\theta$ 0.5 | IoU@$\theta$ 0.7 | mIoU | IoU@$\theta$ 0.1 | IoU@$\theta$ 0.3 | IoU@$\theta$ 0.5 | mIoU |
| ✓ | | | 31.67 | 20.30 | 11.61 | 13.47 | 47.39 | 33.92 | 17.12 | 32.03 | 70.54 | 54.09 | 38.64 | 36.97 |
| | ✓ | | 33.74 | 20.34 | 10.97 | 13.83 | 48.52 | 34.68 | 16.40 | 32.32 | 70.93 | 54.94 | 38.85 | 38.48 |
| ✓ | ✓ | | 35.22 | 22.07 | 11.44 | 14.56 | 52.16 | 35.89 | **17.92** | 34.04 | 73.34 | 55.82 | 39.20 | 39.01 |
| ✓ | | ✓ | 37.59 | 22.76 | 11.40 | 15.23 | 51.72 | 34.60 | 16.94 | 34.15 | 73.56 | 55.43 | 39.31 | 38.74 |
| | ✓ | ✓ | 40.14 | 22.27 | 11.58 | 15.43 | 51.67 | 35.38 | 15.51 | 33.86 | 73.62 | 55.52 | 39.00 | 39.33 |
| ✓ | ✓ | ✓ | **41.15** | **23.45** | **11.72** | **16.03** | **54.95** | **37.39** | 17.69 | **36.34** | **74.26** | **55.91** | **39.72** | **39.51** |

Table 3: Performance (%) over R@1,IoU@$\theta$ and mIoU in ablative experiments of each component of DEBUG model on TACoS, Charades-STA and ActivityNet Captions. CLS: with or without the classification subnet, CFD: with or without the confidence subnet, TP: with or without the temporal pooling.

| Methods | TACoS | ActivityNet |
|---|---|---|
| MCN | 9.41 | 0.30 |
| VSA-RNN | 6.45 | 2.29 |
| VSA-STV | 3.44 | 0.12 |
| CTRL | 4.02 | 0.13 |
| ACRN | 4.09 | 0.13 |
| ABLR$^\diamond$ | 0.14 | 0.02 |
| DEBUG | **0.02** | **0.02** |

Table 4: Average time (s) to localize one sentence for different methods on TACoS and ActivityNet Captions. $^\diamond$ denotes sparse bottom-up model.

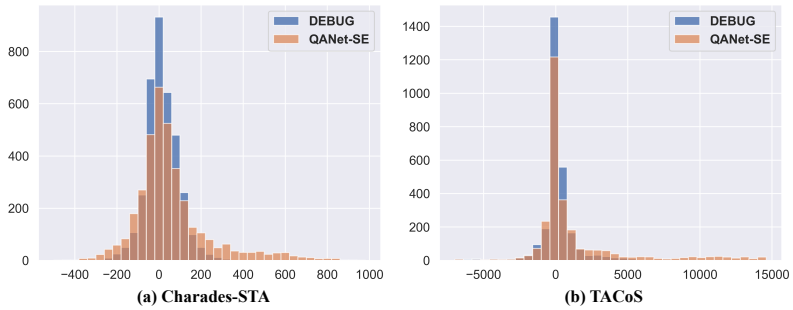


Figure 6: The number of samples across the length error (#frames difference) between predicted segment and ground truth segment on Charades-STA and TACoS.

because the confidence subnet considers the importance of each frame, instead, the classification subnet regards all foreground equally. Meanwhile, the performance of both models can be further boosted by utilizing two subnets simultaneously, which demonstrates that this multi-task design helps each subnet to focus on their own goal and both subnets benefit from sharing features (*i.e.*, one for foreground prediction, and another for "centerness" prediction).

**With vs. Without Temporal Pooling.** From Table 3, we can observe that the temporal pooling trick improves the performance in most of the situations, and the performance gains in TACoS are largest over all benchmarks. The main reason is that the visual appearance of each frame in TACoS is quite similar, *i.e.*, the performance based on single frame prediction is very unstable since multiple frames have similar predictions. Instead, the model with temporal pooling helps to avoid this by fusing multiple frame predictions.

#### 4.4.3 Efficiency Analysis.

We evaluated the efficiency of DEBUG, by comparing the average run time to localize one sentence in the video. As shown in Table 4, the DEBUG significantly reduce the localization time compared to all top-down models (MCN, VSA-RNN, VSA-STV, CTRL, ACRN), and the gap is much wider in long video datasets (*e.g.*, TACoS). This meets with the notorious drawback of the top-down framework, *i.e.*, it is computation-intensive for dense sliding windows or anchors. Meanwhile, DEBUG is even slightly faster than the sparse bottom-up model (ABLR), this is because the QANet backbone only uses conv and self-attention layer instead of the time-consuming RNN in ABLR backbone. All experiments are conducted on the same hardware (an NVIDIA GTX 1080Ti).

| | Methods | Charades-STA 100 | Charades-STA 200 | Charades-STA 300 | TACoS 100 | TACoS 200 | TACoS 300 |
|---|---|---|---|---|---|---|---|
| L | QANet-SE | 54.7 | 73.0 | 81.9 | **15.2** | 21.8 | 28.1 |
| | DEBUG | **56.5** | **75.1** | **83.2** | 13.2 | **22.7** | **29.8** |
| R | QANet-SE | 50.3 | 69.8 | 81.0 | **11.8** | 20.1 | 26.7 |
| | DEBUG | **54.6** | **75.1** | **83.2** | 11.3 | **20.4** | **27.5** |
| L&R | QANet-SE | 38.5 | 61.5 | 74.4 | **6.5** | 12.2 | 16.9 |
| | DEBUG | **46.5** | **71.1** | **81.1** | 4.9 | **12.8** | **19.5** |
| M | QANet-SE | 53.9 | 72.7 | 83.1 | 11.6 | 20.5 | 28.2 |
| | DEBUG | **57.3** | **75.8** | **83.7** | **13.3** | **23.2** | **30.2** |

Table 5: Accuracy (%) of multiple keypoints at different thresholds on Charades-STA and TACoS. L: left point; R: right point; L&R: both left and right point; M: middle point.

#### 4.4.4 Error Analysis.

To analyze the bottleneck of DEBUG and the existing sparse bottom-up framework, and help pave

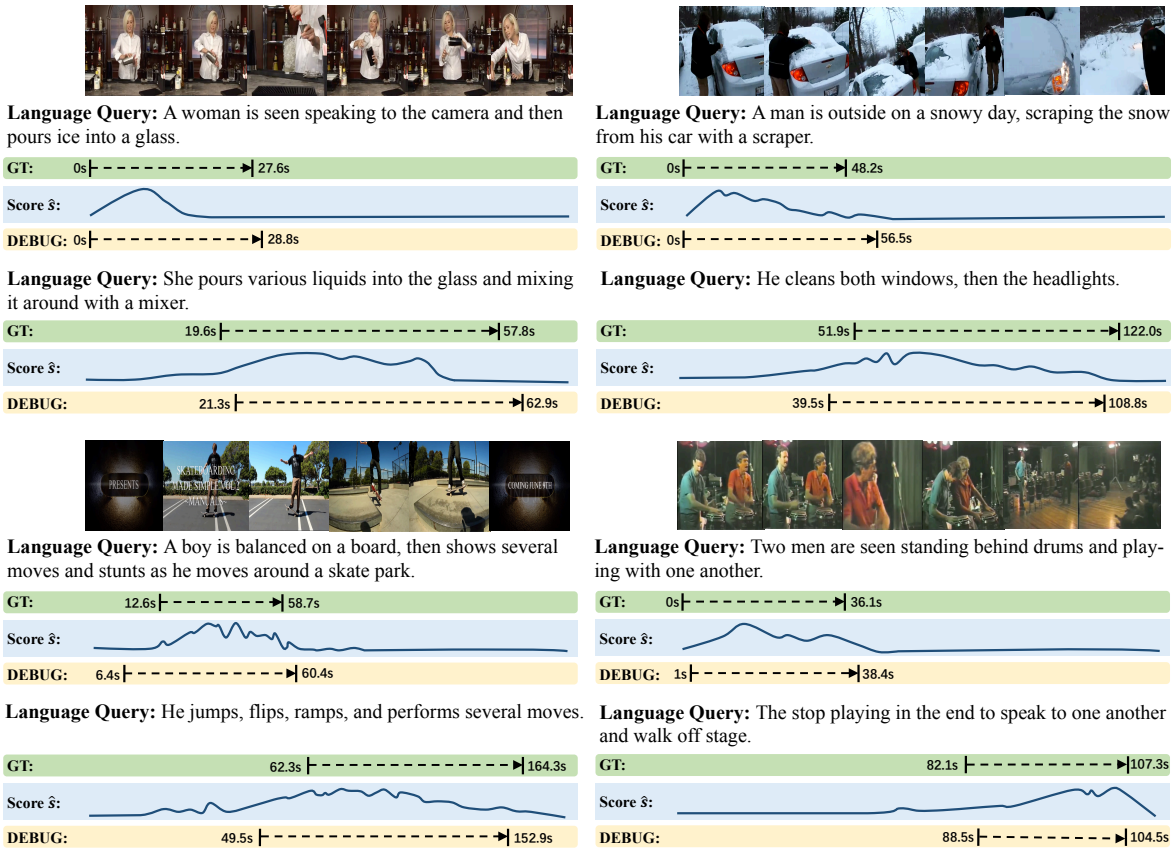


Figure 7: The qualitative results of DEBUG on the ActivityNet Captions dataset.

the way for future research on bottom-up approach for NLVL, We conducted several statistical analysis about the results of DEBUG and its sparse bottom-up counterpart QANet-SE (Section 4.4.1):

**Segment Length.** We compared the length error of the segments predicted by DEBUG and QANet-SE, the results are illustrated in Figure 6. We observe that QANet-SE is prone to predicting overlong segment range (*e.g.*, the samples with length error larger than 300 frames in Chardes-STA or 5,000 frames in TACoS take a large proportion.)

**Keypoint Accuracy.** We compared the accuracy of three keypoints of the ground truth segment: the left, right, and middle point. We regard a keypoint prediction as right if the absolute frame difference between the prediction and ground truth is smaller than a threshold. We used three thresholds (100, 200, 300) and the results are reported in Table 5. We have two observations: 1) For middle point, DEBUG always gets higher accuracy. 2) For boundary points, DEBUG only drops behind QANet-SE in TACoS at threshold 100.

**Analysis.** The sparse bottom-up approach (*e.g.*, QANet-SE) can get good boundary predictions even for long video datasets (*e.g.*, TACoS), which meets with the design of training a boundary classifier. But the bottleneck of this approach is that the predictions of start and end point are independent, which is prone to result in an overlong segment prediction. However, DEBUG predicts boundary from the same frame feature, which can avoid predicting overlong segments. Meanwhile, DEBUG has a confidence subnet to predict "centerness" of each frame, which helps to predict the middle point. But the bottleneck of DEBUG is the accuracy of boundary point in long video dataset.

### 4.4.5 Qualitative Results.

The qualitative results of DEBUG on ActivityNet Captions is illustrated in Figure 7. We can observe that DEBUG is sensitive to language query, *i.e.*, the predicted scores $\hat{s}$ are totally different when given different language queries even for the same video. Meanwhile, the score $\hat{s}$ is always a unimodal curve with its peak near the midpoint of ground truth segment, which meets with the design of DEBUG which uses "centerness" as the confidence target.

## 5 Conclusion

We proposed a novel dense bottom-up framework DEBUG for NLVL. It is the first bottom-up model, which surpasses all top-down models with the highest speed. Compared to the existing bottom-up models, DEBUG improve performance significantly by: 1) making full use of positive samples to alleviate the severe imbalance problem; 2) disentangling boundary detection from query-related prediction to relieve the burden of a single network; 3) predicting boundaries from same frame to avoid local optimum caused by independent predictions. Moving forward, we are going to narrow the gap between top-down and bottom-up models, and design a hybrid framework exploring both choices.

## Acknowledgement

This work was supported by National Key Research and Development Program of China (SQ2018AAA010010), Zhejiang Natural Science Foundation (LR19F020002, LZ17F020001), National Natural Science Foundation of China (61976185, 61572431), the Fundamental Research Funds for the Central Universities Chinese Knowledge Center for Engineering Sciences and Technology, and Joint Research Program of ZJU & Tongdun Technology. Long Chen was supported by 2018 Zhejiang University Academic Award for Outstanding Doctoral Candidates.

## References

A.Karpathy, G.Toderici, S.Shetty, T.Leung, and R.Sukthankar. 2014. Large-scale video classification with convolutional neural networks. In *CVPR*.

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and visual question answering. In *CVPR*.

Jingyuan Chen, Xinpeng Chen, Lin Ma, Zequn Jie, and Tat-Seng Chua. 2018. Temporally grounding natural sentence in video. In *EMNLP*.

Jingyuan Chen, Lin Ma, Xinpeng Chen, Zequn Jie, and Jiebo Luo. 2019a. Localizing natural language in videos. In *AAAI*.

Long Chen, Hanwang Zhang, Jun Xiao, Xiangnan He, Shiliang Pu, and Shih-Fu Chang. 2019b. Counterfactual critic multi-agent training for scene graph generation. In *ICCV*.

Long Chen, Hanwang Zhang, Jun Xiao, Liqiang Nie, Jian Shao, Wei Liu, and Tat-Seng Chua. 2017. Sca-cnn: Spatial and channel-wise attention in convolutional networks for image captioning. In *CVPR*.

Shaoxiang Chen and Yu-Gang Jiang. 2019. Semantic proposal for activity localization in videos via sentence query. In *AAAI*.

D.P.Kingma and J.Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

D.Tran, L.Bourdev, R.Fergus, L.Torresani, and M.Paluri. 2015. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*.

Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. 2019. Centernet: Keypoint triplets for object detection. In *arXiv*.

Jiyang Gao, Chen Sun, Zhenheng Yang, and Ram Nevatia. 2017. Tall: Temporal activity localization via language query. In *ICCV*.

Runzhou Ge, Jiyang Gao, Kan Chen, and Ram Nevatia. 2019. Mac: Mining activity concepts for language-based temporal localization. In *WACV*.

Dongliang He, Xiang Zhao, Jizhou Huang, Fu Li, Xiao Liu, and Shilei Wen. 2019. Read, watch, and move: Reinforcement learning for temporally grounding natural language descriptions in videos. In *AAAI*.

Lisa Anne Hendricks, Oliver Wang, Eli Shechtman, Josef Sivic, Trevor Darrell, and Bryan Russell. 2017. Localizing moments in video with natural language. In *ICCV*.

Lisa Anne Hendricks, Oliver Wang, Eli Shechtman, Josef Sivic, Trevor Darrell, and Bryan Russell. 2018. Localizing moments in video with temporal language. In *EMNLP*.

J.Pennington, R.Socher, and C.Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. 2017. Dense-captioning events in videos. In *ICCV*.

Hei Law and Jia Deng. 2018. Cornernet: Detecting objects as paired keypoints. In *ECCV*.

Meng Liu, Xiang Wang, Liqiang Nie, Xiangnan He, Baoquan Chen, and Tat-Seng Chua. 2018a. Attentive moment retrieval in videos. In *SIGIR*.

Meng Liu, Xiang Wang, Liqiang Nie, Qi Tian, Baoquan Chen, and Tat-Seng Chua. 2018b. Cross-modal moment localization in videos. In *ACM MM*.

Michaela Regneri, Marcus Rohrbach, Dominikus Wetzel, Stefan Thater, Bernt Schiele, and Manfred Pinkal. 2013. Grounding action descriptions in videos. *TACL*.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*.

Zheng Shou, Junting Pan, Jonathan Chan, Kazuyuki Miyazawa, Hassan Mansour, Anthony Vetro, Xavier Giro-i Nieto, and Shih-Fu Chang. 2018. Online detection of action start in untrimmed, streaming videos. In *ECCV*.

Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. 2019. Fcos: Fully convolutional one-stage object detection. In *arXiv*.

Weining Wang, Yan Huang, and Liang Wang. 2019. Language-driven temporal activity localization: A semantic matching reinforcement learning model. In *CVPR*.

Aming Wu and Yahong Han. 2018. Multi-modal circulant fusion for video-to-language and backward. In *IJCAI*.

Caiming Xiong, Victor Zhong, and Richard Socher. 2017. Dynamic coattention networks for question answering. In *ICLR*.

Caiming Xiong, Victor Zhong, and Richard Socher. 2018. Dcn+: Mixed objective and deep residual coattention for question answering. In *ICLR*.

Huijuan Xu, Kun He, L Sigal, S Sclaroff, and K Saenko. 2019. Multilevel language and vision integration for text-to-clip retrieval. In *AAAI*.

Huijuan Xu and Kate Saenko. 2016. Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. In *ECCV*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*.

Yunan Ye, Zhou Zhao, Yimeng Li, Long Chen, Jun Xiao, and Yueting Zhuang. 2017. Video question answering via attribute-augmented attention network learning. In *SIGIR*.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. In *ICLR*.

Yitian Yuan, Tao Mei, and Wenwu Zhu. 2019. To find where you talk: Temporal sentence localization in video with attention based location regression. In *AAAI*.

Da Zhang, Xiyang Dai, Xin Wang, Yuan-Fang Wang, and Larry S Davis. 2019a. Man: Moment alignment network for natural language moment retrieval via iterative graph adjustment. In *CVPR*.

Songyang Zhang, jinsong Su, and Jiebo Luo. 2019b. Exploiting temporal relationships in video moment localization with natural language. In *ACM MM*.

Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. 2019a. Objects as points. In *arXiv*.

Xingyi Zhou, Jiacheng Zhuo, and Philipp Krähenbühl. 2019b. Bottom-up object detection by grouping extreme and center points. In *CVPR*.

# CrossWeigh: Training Named Entity Tagger from Imperfect Annotations

**Zihan Wang*** **Jingbo Shang*** **Liyuan Liu*** **Lihao Lu** **Jiacheng Liu** **Jiawei Han**
University of Illinois at Urbana-Champaign, Urbana, IL, USA
{zihanw2, shang7, ll2, lihaolu2, jl25, hanj}@illinois.edu

## Abstract

Everyone makes mistakes. So do human annotators when curating labels for named entity recognition (NER). Such label mistakes might hurt model training and interfere model comparison. In this study, we dive deep into one of the widely-adopted NER benchmark datasets, CoNLL03 NER. We are able to identify label mistakes in about 5.38% test sentences, which is a significant ratio considering that the state-of-the-art test $F_1$ score is already around 93%. Therefore, we manually correct these label mistakes and form a cleaner test set. Our re-evaluation of popular models on this corrected test set leads to more accurate assessments, compared to those on the original test set. More importantly, we propose a simple yet effective framework, CrossWeigh, to handle label mistakes during NER model training. Specifically, it partitions the training data into several folds and train independent NER models to identify potential mistakes in each fold. Then it adjusts the weights of training data accordingly to train the final NER model. Extensive experiments demonstrate significant improvements of plugging various NER models into our proposed framework on three datasets. All implementations and corrected test set are available at our Github repo [1].

## 1 Introduction

Named entity recognition (NER), identifying both spans and types of named entities in text, is a fundamental task in the natural language processing pipeline. On one of the widely-adopted NER benchmarks, the CoNLL03 NER dataset (Sang and Meulder, 2003), the state-of-the-art NER performance has been pushed to a $F_1$ score around 93% (Akbik et al., 2019), through building end-to-end neural models (Lample et al., 2016; Ma and Hovy, 2016) and introducing language models for contextualized representations (Peters et al., 2017, 2018; Akbik et al., 2018; Liu et al., 2018a). Such high performance makes the label mistakes in manually curated "gold standard" data non-negligible. For example, given a sentence "*Chicago won game 1 with Derrick Rose scoring 25 points.*", this "*Chicago*", representing the NBA team Chicago Bulls, should be annotated as an organization. However, when annotators are not careful or lack background knowledge, this "*Chicago*" might be annotated as a location, thus being a label mistake.

These label mistakes bring up two challenges to NER: (1) mistakes in the test set can interfere the evaluation results and even lead to an inaccurate assessment of model performance; and (2) mistakes in the training set can hurt NER model training. Therefore, in this paper, we conduct empirical studies to understand these mistakes, correct the mistakes in the test set to form a cleaner benchmark, and develop a novel framework to handle the mistakes in the training set.

We dive deep into the CoNLL03 NER dataset, and find label mistakes in about 5.38% test sentences. Considering that the state-of-the-art $F_1$ score on this test set is already around 93%, these 5.38% mistakes should be considered as significant. So we hire human experts to correct these label mistakes in the test set. We then re-evaluate recent state-of-the-art NER models on this new, cleaner test set. Compared to the results on the original test set, the re-evaluation results are more accurate and stable. Therefore, we believe this new test set can better reflect the performance of NER models.

We further propose a novel, general framework, CrossWeigh, to handle the label mistakes during the NER model training stage. Figure 1 presents an overview of our proposed framework. It contains two modules: (1) mistake estimation: it iden-

*Equal Contributions.
[1] https://github.com/ZihanWangKi/CrossWeigh



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5154–5163, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

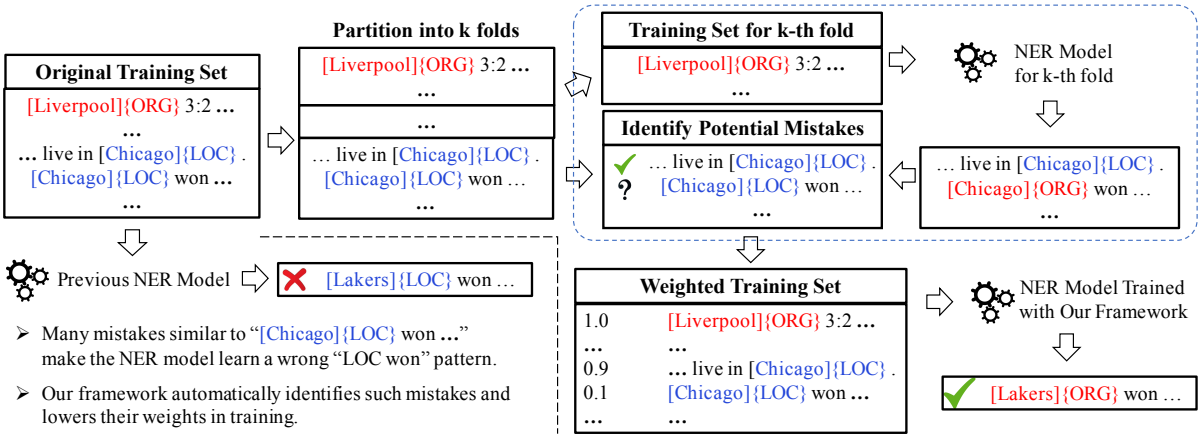


Figure 1: An overview of our proposed CrossWeigh framework. It can better handle label mistakes, identify low quality annotations and conduct learning from a weighted training set.

tifies the potential label mistakes in training data through a cross checking process and (2) mistake re-weighing: it lowers the weights of these instances during the training of the final NER model. The cross checking process is inspired by the k-fold cross validation; differently, in each fold's training data, it removes the data containing any of entities that appeared in this fold. In this way, each sentence will be scored by a NER model trained on a subset of training data not containing any entity in this sentence. Once we know where the potential mistakes are, we lower the weights of these sentences and train the final NER model based on this weighted training set. The final NER model is trained in a mistake-aware way, thus being more accurate. Note that, our proposed framework is general and fits most of, if not all, NER models that accept weighted training data.

To the best of our knowledge, we are the first to handle the label mistake issues systematically in the NER problem. We conduct extensive experiments on both the original CoNLL03 NER dataset and our corrected dataset. CrossWeigh is able to consistently improve performance when plugging with different NER models. In addition, we verify the effectiveness of CrossWeigh on emerging-entity and low-resource NER datasets. In summary, our major contributions are the following:

- We correct label mistakes in the test set of the CoNLL03 NER dataset and re-evaluate popular NER models. This establishes a more accurate NER benchmark.
- We propose a novel framework CrossWeigh to accommodate the mistakes during the model training stage. The proposed framework fits most of, if not all, NER models.
- Extensive experiments demonstrate the significant, robust test $F_1$ score improvements of plugging NER models into our proposed framework on three datasets, not only CoNLL03 but also emerging-entity and low-resource datasets.

**Reproducibility.** We release both the corrected test set and the implementation of CrossWeigh framework[2].

## 2 CoNLL03 NER Re-Examination

The CoNLL03 NER dataset is one of the widely-adopted NER benchmark datasets. Its annotation guideline is based on MUC Conventions[3] (Sang and Meulder, 2003). Following this guideline, the annotators are asked to mark entities of person (PER), location (LOC), and organization (ORG), while using an extra miscellaneous (MISC) type to deal with entities that do not fall in these categories. This dataset has been split into training, development, and test sets, with 14041, 3250, and 3453 sentences, respectively.

### 2.1 Test Set Correction

In order to understand and correct the label mistakes, we have hired 5 human experts as annotators. Before looking at the data, we first train the annotators by carefully going through the aforementioned guideline. During the correction process, we strongly encourage the annotators to use search engines for suspicious token spans. This helps them

[2] https://github.com/ZihanWangKi/CrossWeigh

[3] https://www-nlpir.nist.gov/related_projects/muc/proceedings/ne_task.html

| Sentence | Original labels | Corrected labels |
|---|---|---|
| Sporting Gijon 15 4 4 7 15 22 16 | [Sporting]{ORG} | [Sporting Gijon]{ORG} |
| SOCCER - JAPAN GET LUCKY WIN , CHINA IN SURPRISE DEFEAT . | [JAPAN ]{LOC}, [China]{PER} | [JAPAN]{LOC}, [China]{LOC} |
| NZ 's Bolger says Nats to meet NZ First on Sunday . | [NZ]{LOC}, [Bolger]{PER}, [Nats]{PER}, [NZ]{LOC} | [NZ]{LOC}, [Bolger]{PER}, [Nats]{ORG}, [NZ First]{ORG} |
| Seagramd ace 20/11/96 5,000 Japan | [Seagramd] {MISC}, [Japan]{LOC} | [Seagramd ace]{MISC}, [Japan] {LOC} |

Table 1: Typical Examples of Our Corrections on the CoNLL03 NER dataset.

have more background knowledge. We also allow annotators to look at the original paragraph containing the sentence. This helps them have a better understanding of the context.

For the whole test set, we randomly split the test sentences between each pair combination of 5 annotators. In this way, each sentence in the test set is checked by exactly two annotators. The inter-annotator agreement is 95.66%. This is a reasonable score, given that the inter-annotator agreement in POS tagging annotations is about 97% (Manning, 2011). After we collected all annotations, we run a final round of verification on each sentence, where the original annotation and the two annotators' are not all the same. In the end, we have corrected label mistakes in 186 sentences, which is about 5.38% of the test set.

Table 1 presents some typical examples of our corrections. In the first sentence, as a sport team, "Sporting Gijon" was not annotated completely. In the second sentence, while "JAPAN" is correctly marked as LOC, "China" is wrongly identified as PER instead of LOC. One may notice that they both represent sport teams. However, according to the aforementioned guideline, country names should be marked as LOC even when they are sports teams. More details about this type of labels are discussed in Section 5. In the third sentence, "NZ" is the abbreviation of New Zealand. However, "Nat" and "NZ First" in fact refer to political parties (i.e., New Zealand Young Nationals and New Zealand First). So they should be labelled as ORG. In the forth sentence, looking at its paragraph, our annotators figure out that this is a table about ships and vessels loading items at different locations. Through comparing with other sentences in the context, such as "Algoa Day 21/11/96 6,000 Africa", our annotators identified "Seagramd ace" as a vessel, thus marking it as MISC. We have verified that there is indeed a vessel called "Seagrand Ace" ("Seagramd ace" might be a typo).

## 2.2 CoNLL03 Re-Evaluation

**NER Algorithms.** We re-evaluate following popular NER algorithms:

| Method | Original | Corrected |
|---|---|---|
| LSTM-CRF | 90.64 (±0.23) | 91.47 (±0.15) |
| LSTM-CNNs-CRF | 90.65 (±0.57) | 91.87 (±0.50) |
| VanillaNER | 91.44 (±0.16) | 92.32 (±0.16) |
| Elmo | 92.28 (±0.19) | 93.42 (±0.15) |
| Flair | 92.87 (±0.08) | 93.89 (±0.06) |
| Pooled Flair | 93.14 (±0.14) | 94.13 (±0.11) |

Table 2: CoNLL03 Re-Evaluation: Test $F_1$ scores and standard deviations on both original and corrected datasets. The results are based on 5 different runs.

- **LSTM-CRF** (Lample et al., 2016) incorporates long short term memory (LSTM) neural network with conditional random field (CRF). It also uses a word-wise character LSTM.
- **LSTM-CNNs-CRF** (Ma and Hovy, 2016) has a similar structure as LSTM-CRF, but captures character-level information through a convolutional neural network (CNN) over the character embedding.
- **VanillaNER** (Liu et al., 2018a) also extends LSTM-CRF and LSTM-CNNs-CRF by using a sentence-wise character LSTM.
- **ELMo** (Peters et al., 2018) extends LSTM-CRF and leverages pre-trained word-level language models for better contextualized representations.
- **Flair** (Akbik et al., 2018) also aims for contextualized representations, utilizing pretrained character level language models.
- **Pooled-Flair** (Akbik et al., 2018) extends Flair and maintains an embedding pool for each word to bring in dataset-level word embedding.

We use the implementation released by the authors for each algorithm and report the performance on original test set and corrected test set averaging 5 runs.

**Results & Discussions.** We re-evaluate the performance of the NER algorithms on the corrected test set. Their performance on the original test set is also listed for the reference. From the results in Table 2, one can observe that all models have higher $F_1$ scores as well as smaller standard deviations on the corrected test set, compared to those on the orig-

inal test set. Moreover, LSTM-CRF has a similar performance as LSTM-CNNs-CRF on the original test set, but on average lower performance on the corrected test set. This indicates that the corrected test set may be more discriminative. Therefore, we believe this corrected test set can better reflect the accuracy of NER algorithms in a stable way.

## 3 Our Framework: CrossWeigh

In this section, we introduce our framework. It is worth mentioning that our framework is designed to be general and fits most of, if not all, NER models. The only requirement is the capability to consume weighted training set.

### 3.1 Overview

As we have seen in the Section 2, human curated NER datasets are by no means perfect. Label mistakes in the training set can directly hurt the model's performance. As shown in Figure 1, if there are many similar mistakes like wrongly annotating "Chicago" in "Chicago won ..." as `LOC` instead of `ORG`, the NER model will likely capture the wrong pattern "`LOC` won" and make wrong predictions in future.

Our proposed CrossWeigh framework automates this process. Figure 1 presents an overview. It contains two modules: (1) mistake estimation: it identifies the potential label mistakes in training data through a cross checking process and (2) mistake re-weighing: it lowers the weights of these instances for the NER model training. The workflow is summarized in Algorithm 1.

### 3.2 Preliminary

We denote the training sentences as $\{x_1, x_2, \ldots, x_n\}$ where $n$ is the number of sentences. Each sentence $x_i$ is formed up of a sequence of words. Correspondingly, the label sequence for each sentence is denoted as $\{y_1, y_2, \ldots, y_n\}$. We use $\mathbf{D}$ to denote the training set, including both sentences and their labels. We use $w_i$ to represent the weight of the $i$-th sentence. In most NER papers, the weights are uniform, i.e., $w_i = 1$.

We use $f(\mathbf{D}, w)$ to describe the training process of an NER model using the training set $\mathbf{D}$ weighted by $w$. This training process will return an NER model $M = f(\mathbf{D}, w)$. During this training, the weighted loss function is as below.

$$\mathcal{J} = \sum_{i=1}^{n} w_i \cdot l(M(x_i), y_i) \tag{1}$$

where $l(M(x_i), y_i)$ is the loss function of prediction $M(x_i)$ against its label sequence $y_i$. Typically, it is the negative log-likelihood of the model's prediction $M(x_i)$ compared to labeling sequence $y_i$.

**Algorithm 1:** Our CrossWeigh Framework

**Input**: A NER model $f$, the training set $\mathbf{D} = \langle\{x_1, \ldots, x_n\}, \{y_1, \ldots, y_n\}\rangle$, and hyper-parameters $k$, $t$, and $\epsilon$.
**Output**: A final NER model
**for** *i = 1 ... n* **do**
  $c_i \leftarrow 0, w_i \leftarrow 1$
**for** *iter* = $1 \ldots t$ **do**
  Randomly partition $\mathbf{D}$ into $k$ folds.
  **for** *Each fold* $D_i$ **do**
    Obtain test_entities$_i$. (Eq. 2)
    Build train_set$_i$. (Eq. 3).
    Train a NER model $M_i = f(\text{train\_set}_i, w)$.
    **for** *Each* $x_j \in D_i$ **do**
      $\hat{y}_j \leftarrow M_i$'s prediction on $x_j$.
      **if** $y_j \neq \hat{y}_j$ **then**
        $c_i \leftarrow c_i + 1$
**for** $i = 1 \ldots n$ **do**
  Compute $w_i$ (Eq. 4).
**Return** $f(\mathbf{D}, w)$.

### 3.3 Mistake Estimation

Our mistake estimation module is designed to let an NER model itself decide which sentences contain mistake and which do not. We would like to find sentences with label mistakes as many as possible (i.e. high recall), while keeping away from wrongly identified non-mistake sentences (i.e. high precision).

The basic idea of our mistake estimation module is similar to k-fold cross validation, however, in each fold's training data, it further removes the data containing any of entities appearing in this fold. The details are presented as follows.

We first randomly partition the training data into $k$ folds: $D_1, D_2, \ldots, D_k$.

We then train $k$ NER models separately based on these $k$ folds. The $i$-th ($1 \leq i \leq k$) NER model $M_i$ will be evaluated on the sentences in the hold-out fold $D_i$.

During its training, we avoid any sentence that may lead to "easy prediction" on this hold-out set.

| | Original CoNLL03 | | Corrected CoNLL03 | |
|---|---|---|---|---|
| | w/o CrossWeigh | w/ CrossWeigh | w/o CrossWeigh | w/ CrossWeigh |
| VanillaNER | 91.44 (±0.16) | **91.78 (±0.06)** | 92.32 (±0.16) | **92.64 (±0.08)** |
| Flair | 92.87 (±**0.08**) | **93.19** (±0.09) | 93.89 (±**0.06**) | **94.18 (±0.06)** |
| Pooled-Flair | 93.14 (±0.14) | **93.43 (±0.06)** | 94.13 (±0.11) | **94.28 (±0.05)** |

Table 3: Test $F_1$ scores and its standard deviations of models trained without or with CrossWeigh.

Therefore, we inspect every sentence in $D_i$ and get the set of entities as follows.

$$\text{test\_entities}_i = \bigcup_{x_j \in D_i} e_j \tag{2}$$

where $e_j$ is the set of named entities in sentence $x_j$. We only consider the surface name in this entity set. That is, no matter "Chicago" is `LOC` or `ORG`, it only counts as its surface name "Chicago".

All training sentences that have entities included in $\text{test\_entities}_i$ will be excluded in training process of the model $M_i$. Specifically,

$$\text{train\_set}_i = \{x_j | e_j \cap \text{test\_entities}_i = \emptyset, 1 \leq j \leq n\} \tag{3}$$

We call this step as *entity disjoint filtering*. The intuition behind this step is that we want the model to make prediction of an entity without prior information of the entity itself from training. This will be helpful to detect sentences that are inconsistent.

We train $k$ models $M_i$ by feeding each $\text{train\_set}_i$ into $f(\cdot, \cdot)$ with default uniform weight, and we use each $M_i$ to make predictions for $D_i$ and check for each sentence, whether the original label is the same as the model output. In this way, if the trained model $M_i$ makes correct predictions on some sentences in $D_i$, they are more likely mistake-free. For those sentences that have labels disagreeing with the model output, we mark them as *potentially mistake*.

We run this mistake estimation module multiple iterations (i.e. $t$ iterations) using different random partitions. Then, for each sentence in the training set, we get $t$ estimations for it. We denote $c_i$ ($0 \leq c_i \leq t$) as the confidence that sentence $x_i$ contains label mistakes. $c_i$ is defined as the the number of *potentially mistake* indications among all $t$ estimations.

The number of folds $k$ plays the role of a trade-off between the efficiency of the mistake estimation process and the number of training examples that can be used in each $M_i$. When $k$ becomes larger, each fold $D_i$ will be smaller, thus leading to a smaller size of $\text{test\_entities}_i$; correspondingly, a larger $\text{train\_set}_i$ will be picked. The model can therefore be trained with more examples. However, it also slows down the whole mistake estimation process. On the CoNLL03 NER dataset, we observe that $k = 10$ leads to effective results, while having a reasonable running time.

### 3.4 Mistake Reweighing

In the mistake reweighing module, we adjust weight $w_i$ for each sentence $x_i$ that is marked as *potentially mistake* in the mistake estimation step. Here, we assign a weight $w_i$ to all sentences marked, while the weights of other sentences remain 1. Specifically, we set $\forall i (1 \leq i \leq n)$,

$$w_i = \epsilon^{c_i} \tag{4}$$

where $\epsilon$ is a parameter. In practice, it can be chosen according to the quality of mistake estimation module. Particularly, we first estimate the precision of the detected mistakes of a single iteration. Let $p$ be the ratio of the number of true detected label mistakes over the number of detected label mistakes. $p$ can be roughly estimated through a manual check of a random sample from the detected label mistakes. Then, we choose $\epsilon = 1 - p$, because $1 - p$ represents the fraction of these detected label mistakes that might be still useful during the model training. Therefore, for the sentences that are marked as *potentially mistake* in that iteration, $\epsilon$ of them are actually correct. With more iterations, the confidence of being correct lowers like a binomial distribution, which is the reason that we chose an exponential decaying weight function in Equation 4.

## 4 Experiments

In this section, we conduct several experiments to show effectiveness of our CrossWeigh framework. We first evaluate the overall performance of CrossWeigh on benchmark NER datasets, by plugging it into three base NER models. Since we have two modules in CrossWeigh, we then dive into each module and explore different variants and ablations. In addition, we further verify the effectiveness of CrossWeigh on two more datasets:

an emerging-entity NER dataset from WNUT'17 and a low-resource language NER dataset of the Sinhalese language.

### 4.1 Experimental Settings

**Dataset.** We use both the original and corrected CoNLL03 datasets. We follow the standard train/dev/test splits and use both the train set and dev set for training (Peters et al., 2017; Akbik et al., 2018). Entity-wise $F_1$ score on the test set is the evaluation metric.

**Base NER Algorithm.** We mainly choose Flair as our base NER algorithm. Flair is a strong NER algorithm using external resources (large corpus to train a language model). While Pooled-Flair has even better performance, its computational cost refrains us from doing extensive experiments.

**Default Parameters in CrossWeigh.** For all NER algorithms we experiment with, their default parameters are used. For CrossWeigh parameters, by default, we set $k = 10$, $t = 3$, and $\epsilon = 0.7$. We decide $\epsilon = 0.7$ because among 100 randomly sampled sentences with *potentially mistake*, we find that 27 of them really contain label mistakes (i.e., the probability of one annotation to be correct is roughly 70%). We use both train and development set to train the models, and report average $F_1$ and its standard deviation on both original test set and our corrected test set across 5 different runs (Peters et al., 2017).

### 4.2 Overall Performance

We pair CrossWeigh with our base algorithm (i.e. Flair) and two best-performing NER algorithms with or without language models in Table 2 (i.e. Pooled-Flair and VanillaNER), and evaluate their performance. As shown in Table 3, compared with the three algorithms, applying CrossWeigh always leads to a higher $F_1$ score and a comparable, sometimes even smaller, standard deviation. Therefore, it is clear that CrossWeigh can improve the performance of NER models. The smaller standard deviations also imply that the models trained with CrossWeigh are more stable. All these results illustrate the superiority of training with CrossWeigh.

### 4.3 Ablations and Variants

We pick Flair as the base algorithm to conduct ablation study.

**Entity Disjoint Filtering.** There is an entity disjoint filtering step, when we are collecting training data train_set$_i$ for the NER model $M_i$ during the mistake estimation step. To study its importance, we have done a few ablation experiments.

| Method | Original | Corrected |
|---|---|---|
| w/o CrossWeigh | 92.87 (±0.08) | 93.89 (±0.06) |
| w/ CrossWeigh | 93.19 (±0.09) | 94.18 (±0.06) |
| − Entity Disjoint | 92.88 (±0.11) | 93.84 (±0.08) |
| + Random Discard | 93.01 (±0.10) | 93.94 (±0.10) |

Table 4: Importance of Entity Disjoint Filtering.

We have evaluated the following variants:

- Flair w/ CrossWeigh **– Entity Disjoint**: Skip the entity disjoint filtering step.
- Flair w/ CrossWeigh **+ Random Discard**: Instead of entity disjoint filtering, randomly discard the same number of sentences from each train_set$_i$ as it would do.

The results are listed in Table 4. One can easily observe that without the entity disjoint filtering, the $F_1$ scores are very close to the raw Flair model. This demonstrates that the entity disjoint filtering is critical to reduce the over-fitting risk in the mistake estimation step. Also, our proposed entity disjoint filtering strategy works more effective than random discard. This further confirms the effectiveness of entity disjoint filtering.

**Variants in Computing $c_i$.** There is definitely more than one way to determine $c_i$. Let $\delta$ be the number of "potentially mistake"s among the $t$ estimations, we can apply any of the following heuristics:

- **Ratio**: $c_i$ is the number of "potentially mistake" (i.e. $c_i = \delta$). This is the method mentioned in Section 3, and used by default.
- **At Least One**: $c_i$ is the indicator of at least one estimation being "potentially mistake" (i.e. $c_i = t \iff \delta >= 1$).
- **Majority**: $c_i$ is the indicator of at least $\lfloor t/2 \rfloor + 1$ estimations being "potentially mistake" (i.e. $c_i = t \iff \delta >= \lfloor t/2 \rfloor + 1$).
- **All**: $c_i$ is the indicator of all $t$ estimations being "potentially mistake" (i.e. $c_i = t \iff \delta = t$).

We evaluate the performance of these heuristics when used in CrossWeigh, as shown in Table 5. There is not much difference across these heuristics, while our default choice "Ratio" is the most stable.

### 4.4 Label Mistake Identification Results

Another usage of CrossWeigh is to identify potential label mistakes during label annotation process, thus improving the annotation quality. This could

| Heuristic | Original | Corrected |
|---|---|---|
| At Least One | 93.10 (±0.10) | 94.16 (±0.07) |
| Majority | 93.20 (±0.09) | 94.12 (±0.07) |
| All | 93.16 (±0.09) | 94.11 (±0.09) |
| Ratio | 93.19 (±0.09) | 94.18 (±0.06) |

Table 5: Different $c_i$ Estimation Heuristics.

| $t$ | Original | Corrected |
|---|---|---|
| 1 | 93.09 (±0.14) | 94.07 (±0.09) |
| 5 | 93.23 (±0.10) | 94.14 (±0.08) |
| 3 | 93.19 (±0.09) | 94.18 (±0.06) |

Table 6: Different Numbers of Iterations $t$.

| $k$ | Original | Corrected |
|---|---|---|
| 2 | 92.11 (±0.24) | 92.88 (±0.11) |
| 5 | 93.12 (±0.08) | 94.12 (±0.08) |
| 10 | 93.19 (±0.09) | 94.18 (±0.06) |

Table 7: Different Numbers of Folds $k$.

| $\epsilon$ | Original | Corrected |
|---|---|---|
| 0.3 | 92.79 (±0.14) | 93.68 (±0.15) |
| 0.5 | 93.21 (±0.09) | 94.18 (±0.07) |
| 0.9 | 93.01 (±0.10) | 93.96 (±0.09) |
| 0.7 | 93.19 (±0.09) | 94.18 (±0.06) |

Table 8: Different Weight Adjustments $\epsilon$.

be also helpful to active learning.

Specifically in this experiment, we apply our noise estimation module to the concatenation of training and testing data. As we have manually corrected the label mistakes in the testing set, we are able to report the number of true mistakes among the potential mistakes discovered in the test set.

The results are presented in Table 9. The potential mistakes are the total number of mistakes identified by CrossWeigh, and actual mistakes is the true positives among all identifications. From the results, we can see that when the base model is Flair, CrossWeigh is able to spot more than 75% of label mistakes, while maintaining a precision about 25%. It is worth noting that 25% is a reasonably high precision, given that the label mistake ratio is only 5.38%. The 75% recall indicates that CrossWeigh is able to identify most of the label mistakes, which are extremely valuable to improve the annotation quality.

### 4.5 Parameter Study

We study how CrossWeigh performs with different hyper-parameters, i.e., $t$ (the number of iterations that we run mistake estimation), $k$ (the number of folds in mistake estimation), and $\epsilon$ (the weight scaling factor of identified potential mistakes).

In principle, a larger $t$ usually gives us a more stable mistake estimation. However, a larger $t$ also requires more computation resources. In our experiments (see Table 6), we find that $t = 3$ provides a good enough result.

Specifically, during mistake estimation, we have to choose the number of folds to partition the data. The more partitions made, the smaller each $D_i$ is and the fewer sentences will be filtered, leading to more training data train_set$_i$ and better trained $M_i$. On the other hand, this is at the cost of higher computational expense. As shown in Table 7, we observe that $k = 5, k = 10$ are significantly better than $k = 2$. In fact, when $k = 2$, each train_set$_i$ has only around 5000 sentences and 1500 entities inside. These numbers become 7000 and 4000 when $k = 5$, and 9000 and 7000 when $k = 10$.

As we mentioned before, the value $\epsilon$ can be chosen by estimating the quality of mistake estimation. Table 8 presents some results when other values are used. $\epsilon = 0.3$ leads to the worst performance. Since our estimation does not have high precision, assigning $\epsilon$ to a low value like $0.3$ may not be a good choice. Interestingly $\epsilon = 0.5$ performs on par with $\epsilon = 0.7$, and even slightly better in the original test set. We hypothesize that this is because there are some ambiguous sentences that we did not count during estimating the quality of mistake estimation, see Section 5, and the actual precision could be higher.

### 4.6 Other Datasets

To show the generalizability of our method across domains and languages, we further evaluate CrossWeigh on an emerging-entity NER dataset from WNUT'17 and a Sinhalese NER dataset from LORELEI[4]. Sinhalese is a low-resource, morphology-rich language. For WNUT'17, we use the Flair as our base NER algorithm. For Sinhalese, we use BERT (Devlin et al., 2018) followed by a BiLSTM-CRF as our base NER algorithm. We use the same parameters as used in the previous CoNLL03 experiments, namely $k = 10, t = 3, \epsilon = 0.7$.

The results averaged across 5 runs are reported in Table 10. One can observe quite similar results as those in the previous CoNLL03 experiments.

[4]LDC2018E57

| Algorithm | Potential Mistakes | Actual Mistakes | Precision | Recall | F1 |
|---|---|---|---|---|---|
| **Flair** | 573.0 | 144.0 | 0.2513 | 0.7742 | 0.3794 |
| **VanillaNER** | 821.67 | 146.33 | 0.1781 | 0.7867 | 0.2904 |

Table 9: Quality of noise estimation. The number of true mistakes, based on our manual correction, is 186. The potential mistakes are counted based on average of 3 runs.

| **Dataset** | **w/o CrossWeigh** | **w/ CrossWeigh** |
|---|---|---|
| WNUT'17 | 48.96 (±0.97) | 50.03 (±0.40) |
| Sinhalese | 66.34 (±0.34) | 67.68 (±0.21) |

Table 10: Applying CrossWeigh on other datasets

Training with CrossWeigh leads to a significantly higher F1 and a smaller standard deviation. This suggests that CrossWeigh works well in other datasets and languages.

## 5 Case Studies

**Test Set Correction.** Despite the label mistakes that we have corrected, we also find some ambiguous but consistent cases. For instances, (1) All NBA/NHL divisions such as "CENTRAL DIVISION", "WESTERN DIVISION" were annotated as `MISC`, while all European leagues, such as "SPANISH FIRST DIVISION" and "ENGLISH PREMIER LEAGUE", are not marked as `MISC` correctly — only "SPANISH" and "ENGLISH" are labelled as `MISC`. And (2) "Team A at Team B" is a way to say "Team A" as an away team playing with Team B as a home team. However, in almost all cases (only 1 exception out of more than 100), "Team A" was labelled as `ORG` while "Team B" was labelled as `LOC`. For example, in "MINNESOTA AT MILWAUKEE", "NEW YORK AT CALIFORNIA", and "ORLANDO AT LA LAKERS", the second sports team "MILWAUKEE", "CALIFORNIA" and "LA LAKERS" were always labelled as `LOC`. Because these parts behave consistently and generally follow the annotation guideline, we didn't touch them during the test set correction.

**CrossWeigh Framework.** The mistakes in the training set can harm the generalizability of the trained model. For example, in Table 11, the original training sentence "Hapoel Haifa 3 Maccabi Tel Aviv 1" contains a label mistake, because "Maccabi Tel Aviv" is a sports team but was not annotated completely. Interestingly, there is a similar sentence in the test set – "Hapoel Jerusalem 0 Maccabi Tel Aviv 4". In all 5 different runs of the original Flair model, they failed to predict correctly that "Maccabi Tel Aviv" in the test sentence as `ORG` because of the label mistake in the training sentence, even though "`ORG` number `ORG` number" is an obvious pattern in the training set. In CrossWeigh, this label mistake in the training set was detected in all $t = 3$ iterations and therefore assigned a very low weight during training. After that, in all 5 different runs of Flair w/ CrossWeigh, they successfully predict that "Maccabi Tel Aviv" is `ORG` as a whole.

## 6 Related Work

In this section, we review related works from three aspects, mistake identification, cross validation & boosting, and NER algorithms.

### 6.1 Mistake Identification

Researchers have noticed the label mistakes in sophisticated natural language processing tasks for a while. For example, it is reported that the inter-annotator agreement is about 97% on the Penn Treebank POS tagging dataset (Manning, 2011; Subramanya et al., 2010).

There are a few attempts towards detecting label mistakes automatically. For example, Nakagawa and Matsumoto (2002) designed a support vector machine-based model to assign weights to examples that were hard to classify in the POS tagging task. Helgadóttir et al. (2014) further applied previous detection models and manually corrected Icelandic Frequency Dictionary (Pind et al., 1991) POS tagging dataset. However, these two methods are specifically developed for POS tagging and cannot be directly applied to NER.

Recently, Rehbein and Ruppenhofer (2017) extends variational inference with active learning to detect label mistakes in "silver standard" data generated by machines. In this paper, we focus on detecting label mistakes in "gold standard" data, which is a different scenario.

### 6.2 Cross Validation & Boosting

Our mistake estimation module shares some similarity with cross validation. Applying cross valida-

| | Training Set | Test Set |
|---|---|---|
| **Text** | Hapoel Haifa 3 Maccabi Tel Aviv 1 | Hapoel Jerusalem 0 Maccabi Tel Aviv 4 |
| **Original Annotations** | [Hapoel Haifa]{ORG}, **[Tel Aviv]{ORG}** | [Hapoel Jerusalem]{ORD}, [Maccabi Tel Aviv]{ORG} |
| **Correct Annotations** | [Hapoel Haifa]{ORG}, [Maccabi Tel Aviv]{ORG} | [Hapoel Jerusalem]{ORD}, [Maccabi Tel Aviv]{ORG} |
| | **Action** | **Result** |
| **Flair** | Assumes this sentence is equally reliable as others. | [Hapoel Jerusalem]{ORD}, **[Tel Aviv]{ORG}** |
| **Flair w/ CrossWeight** | Lowers the weight of this sentence as mistakes. | [Hapoel Jerusalem]{ORD}, [Maccabi Tel Aviv]{ORG} |

Table 11: Case Study on the CoNLL03 dataset. Errors are marked with red

tion to the training set is the same as our mistake estimation module, except that we have an *entity disjoint filtering* step. Experiments in Table 4 show that this step is crucial to our performance gain. The choice of ten folds also stems from cross validation (Kohavi, 1995).

Another similar thread of work is boosting, such as Adaboost (Freund et al., 1999; Schapire and Singer, 1999). For example, Abney et al. (1999) has applied Adaboost on the Penn Treebank POS tagging dataset and gained encouraging results on model performance. In boosting algorithms, the training data is assumed to be perfect. Therefore, it trains models using the full training set and then increases the weights of training instances that fails the current model in the next round of learning. In contrast, we decrease the weights of sentences that differ from the model built upon the entity disjoint training set. More importantly, our framework is a better fit for neural models, because they can likely overfit the training data and thus being bad choices as weak classifiers in boosting.

### 6.3 NER Algorithms

Neural models have been widely used for Named Entity Recognition, and the state-of-the-art models integrate LSTMs, conditional random field and language models (Lample et al., 2016; Ma and Hovy, 2016; Liu et al., 2018b; Peters et al., 2018; Akbik et al., 2018). In this paper, we focus on improving the annotation quality for NER, and our method has a big potential to help other methods, especially for noisy datasets (Shang et al., 2018).

## 7 Conclusion & Future work

In this paper, we explore and correct the label mistakes in the CoNLL03 NER dataset. Based on the corrected test set, we re-evaluate most of recent NER models. We further propose a novel framework, CrossWeigh, that is able to detect label mistakes in the training set and then train a more robust NER model accordingly. Extensive experiments demonstrate the effectiveness of CrossWeigh on three datasets and also indicate the potentials of using CrossWeigh to improve the annotation quality during the label curation process.

In future, we plan to extend our framework into an iterative setting, similar to those boosting algorithms. The bottleneck of doing this lies in the efficiency problems of training multiple deep neural models hundreds of times. One solution to overcome it is to apply meta learning. We can first train a meta model and only fine-tune on different training data on each fold. In this way, we can identify label mistakes more accurately and obtain a series of weighted models at the end.

## Acknowledge

We thank all reviewers for valuable comments and suggestions that brought improvements to our final version. Research was sponsored in part by U.S. Army Research Lab. under Cooperative Agreement No. W911NF-09-2-0053 (NSCTA), DARPA under Agreement No. W911NF-17-C-0099, National Science Foundation IIS 16-18481, IIS 17-04532, and IIS-17-41317, DTRA HDTRA11810026, Google Ph.D. Fellowship and grant 1U54GM114838 awarded by NIGMS through funds provided by the trans-NIH Big Data to Knowledge (BD2K) initiative (www.bd2k.nih.gov). Any opinions, findings, and conclusions or recommendations expressed in this document are those of the author(s) and should not be interpreted as the views of any U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon. This research was supported by grant 1U54GM114838 awarded by NIGMS through funds provided by the trans-NIH Big Data to Knowledge (BD2K) initiative (www.bd2k.nih.gov). This work was supported by Contracts HR0011-15-C-0113 and HR0011-18-2-0052 with the US Defense Advanced Research Projects Agency (DARPA).

## References


Steven Abney, Robert E Schapire, and Yoram Singer. 1999. Boosting applied to tagging and pp attachment. In *1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.

Alan Akbik, Tanja Bergmann, and Roland Vollgraf. 2019. Pooled contextualized embeddings for named entity recognition. In *NAACL*.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 1638–1649.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Yoav Freund, Robert Schapire, and Naoki Abe. 1999. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612.

Sigrún Helgadóttir, Hrafn Loftsson, and Eiríkur Rögnvaldsson. 2014. Correcting errors in a new gold standard for tagging icelandic text. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC 2014, Reykjavik, Iceland, May 26-31, 2014.*, pages 2944–2948.

Ron Kohavi. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *IJCAI*, pages 1137–1145. Morgan Kaufmann.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 260–270.

Liyuan Liu, Xiang Ren, Jingbo Shang, Xiaotao Gu, Jian Peng, and Jiawei Han. 2018a. Efficient contextualized representation: Language model pruning for sequence labeling. In *EMNLP*, pages 1215–1225. Association for Computational Linguistics.

Liyuan Liu, Jingbo Shang, Xiang Ren, Frank Fangzheng Xu, Huan Gui, Jian Peng, and Jiawei Han. 2018b. Empower sequence labeling with task-aware neural language model. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Xuezhe Ma and Eduard H. Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Christopher D. Manning. 2011. Part-of-speech tagging from 97% to 100%: Is it time for some linguistics? In *Computational Linguistics and Intelligent Text Processing - 12th International Conference, CICLing 2011, Tokyo, Japan, February 20-26, 2011. Proceedings, Part I*, pages 171–189.

Tetsuji Nakagawa and Yuji Matsumoto. 2002. Detecting errors in corpora using support vector machines. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *ACL (1)*, pages 1756–1765. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*, pages 2227–2237. Association for Computational Linguistics.

Jörgen Pind, Friðrik Magnússon, and Stefán Briem. 1991. Íslensk orðtíðnibók [the icelandic frequency dictionary]. *The Institute of Lexicography, University of Iceland, Reykjavik, Iceland*.

Ines Rehbein and Josef Ruppenhofer. 2017. Detecting annotation noise in automatically labelled data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1160–1170.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning, CoNLL 2003, Held in cooperation with HLT-NAACL 2003, Edmonton, Canada, May 31 - June 1, 2003*, pages 142–147.

Robert E Schapire and Yoram Singer. 1999. Improved boosting algorithms using confidence-rated predictions. *Machine learning*, 37(3):297–336.

Jingbo Shang, Liyuan Liu, Xiaotao Gu, Xiang Ren, Teng Ren, and Jiawei Han. 2018. Learning named entity tagger using domain-specific dictionary. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2054–2064.

Amarnag Subramanya, Slav Petrov, and Fernando C. N. Pereira. 2010. Efficient graph-based semi-supervised learning of structured tagging models. In *EMNLP*, pages 167–176. ACL.

# A Little Annotation does a Lot of Good: A Study in Bootstrapping Low-resource Named Entity Recognizers

**Aditi Chaudhary, Jiateng Xie, Zaid Sheikh,**
**Graham Neubig, Jaime G. Carbonell**
{aschaudh, jiatengx, zsheikh, gneubig, jgc}@cs.cmu.edu
Language Technologies Institute
Carnegie Mellon University

## Abstract

Most state-of-the-art models for named entity recognition (NER) rely on the availability of large amounts of labeled data, making them challenging to extend to new, lower-resourced languages. However, there are now several proposed approaches involving either cross-lingual transfer learning, which learns from other highly resourced languages, or active learning, which efficiently selects effective training data based on model predictions. This paper poses the question: given this recent progress, and limited human annotation, what is the most effective method for efficiently creating high-quality entity recognizers in under-resourced languages? Based on extensive experimentation using both simulated and real human annotation, we find a dual-strategy approach best, starting with a cross-lingual transferred model, then performing targeted annotation of only uncertain entity spans in the target language, minimizing annotator effort. Results demonstrate that cross-lingual transfer is a powerful tool when very little data can be annotated, but an entity-targeted annotation strategy can achieve competitive accuracy quickly, with just one-tenth of training data. The code is publicly available here.[1]

## 1 Introduction

Named entity recognition (NER) is the task of detecting and classifying named entities in text into a fixed set of pre-defined categories (person, location, etc.) with several downstream applications including machine reading (Chen et al., 2017), entity and event co-reference (Yang and Mitchell, 2016), and text mining (Han and Sun, 2012). Recent advances in deep learning have yielded state-of-the-art performance on many sequence labeling tasks, including NER (Collobert et al., 2011; Ma and Hovy, 2016; Lample et al., 2016; Peters et al., 2018). However, the performance of these models is highly dependent on the availability of large amounts of annotated data, and as a result their accuracy is significantly lower on languages that have fewer resources than English. In this work, we ask the question "how can we efficiently bootstrap a high-quality named entity recognizer for a low-resource language with only a small amount of human effort?" Specifically, we leverage recent advances in data-efficient learning for low-resource languages, proposing the following "recipe" for bootstrapping low-resource entity recognizers: First, we use *cross-lingual transfer learning* (Yarowsky et al., 2001; Ammar et al., 2016), which applies a model trained on another language to low-resource languages, to provide a good preliminary model to start the bootstrapping process. Specifically, we use the model of Xie et al. (2018), which reports strong results on a number of language pairs. Next, on top of this transferred model we further employ *active learning* (Settles and Craven, 2008; Marcheggiani and Artieres, 2014), which helps improve annotation efficiency by using model predictions to select informative, rather than random, data for human annotators. Finally, the model is *fine-tuned* on data obtained using active learning to improve accuracy in the target language.

Within this recipe, the choice of specific method for choosing and annotating data within active learning is highly important to minimize human effort. One relatively standard method used in previous work on NER is to select full sequences based on a criterion for the uncertainty of the entities recognized therein (Culotta and McCallum, 2005). However, as it is often the case that only a single entity within the sentence may be of interest, it can still be tedious and wasteful to annotate full sequences when only a small portion of the

[1] https://github.com/Aditi138/EntityTargetedActiveLearning



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5164–5174,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

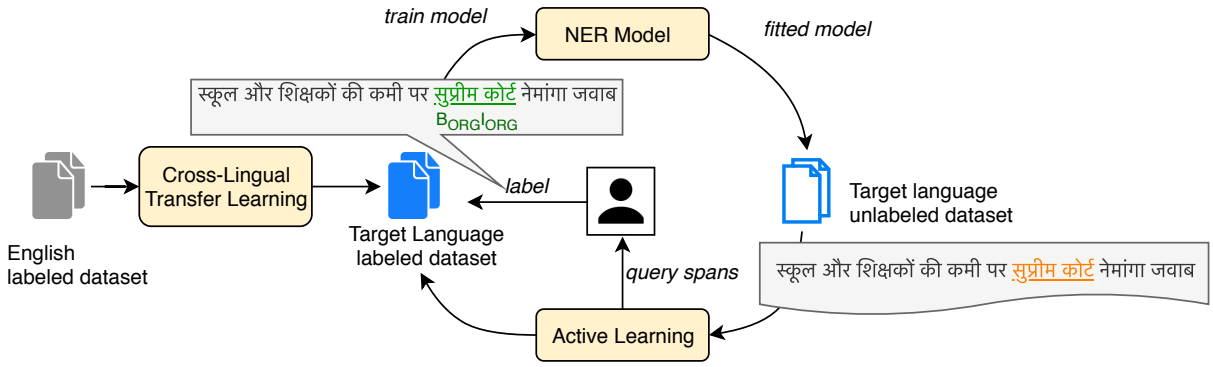


Figure 1: Our proposed recipe: cross-lingual transfer is used for projecting annotations from an English labeled dataset to the target language. Entity-targeted active learning is then used to select informative sub-spans which are likely entities for humans to annotate. Finally, the NER model is fine-tuned on this partially-labeled dataset.

sentence is of interest (Neubig et al., 2011; Sperber et al., 2014). Inspired by this finding and considering the fact that named entities are both important and sparse, we propose an entity-targeted strategy to save annotator effort. Specifically, we select uncertain subspans of tokens within a sequence that are most likely named entities. This way, the annotators only need to assign types to the chosen subspans without having to read and annotate the full sequence. To cope with the resulting partial annotation of sequences, we apply a constrained version of conditional random fields (CRFs), partial CRFs, during training that only learn from the annotated subspans (Tsuboi et al., 2008; Wanvarie et al., 2011).

To evaluate our proposed methods, we conducted simulated active learning experiments on 5 languages: Spanish, Dutch, German, Hindi and Indonesian. Additionally, to study our method in a more practical setting, we conduct human annotation experiments on two low-resource languages, Indonesian and Hindi, and one simulated low-resource language, Spanish. In sum, this paper makes the following contributions:

1. We present a bootstrapping recipe for improving low-resource NER. With just one-tenth of tokens annotated, our proposed entity-targeted active learning method provides the best results among all active learning baselines, with an average improvement of 9.9 F1.

2. Through simulated experiments, we show that cross-lingual transfer is a powerful tool, outperforming the un-transferred systems by an average of 8.6 F1 with only one-tenth of tokens annotated.

3. Human annotation experiments show that annotators are more accurate in annotating entities when using the entity-targeted strategy as opposed to full sequence annotation. Moreover, this strategy minimizes annotator effort by requiring them to label fewer tokens than the full-sequence annotation.

## 2 Approach

As noted in the introduction, our bootstrapping recipe consists of three components (1) cross-lingual transfer learning, (2) active learning to select relevant parts of the data to annotate, and (3) fine-tuning of the model on these annotated segments. Steps (2) and (3) are continued until the model has achieved an acceptable level of accuracy, or until we have exhausted our annotation budget. The system overview can be seen in Figure 1. In the following sections, we describe each of these three steps in detail.

### 2.1 Cross-lingual Transfer Learning

The goal of cross-lingual learning is to take a recognizer trained in a source language, and transfer it to a target language. Our approach to doing so for NER follows that of Xie et al. (2018), and we provide a brief review in this section.

To begin with, we assume access to two sets of pre-trained monolingual word embeddings in the source and target languages, $X$ and $Y$, one small bilingual lexicon, either provided or obtained in an unsupervised manner (Artetxe et al., 2017; Conneau et al., 2017a), and labeled training data in the source language. Using these resources, we train bilingual word embeddings (BWE) to create a word-to-word translation dictionary, and finally

use this dictionary to translate the source training data into the target language, which we use to train an NER model.

To learn BWE, we first obtain a linear mapping $W$ by solving the following objective:

$$W^* = \arg\min_W \|WX_D - Y_D\|_F \text{ s.t. } WW^\top = I,$$

where $X_D$ and $Y_D$ correspond to the aligned word embeddings from the bilingual lexicon. $F$ denotes the Frobenius norm. We can first compute the singular value decomposition $Y_D^T X_D = U\sum V^\top$, and solve the objective by taking $W^* = UV^\top$. We obtain BWE by linearly transforming the source and target monolingual word embeddings with $U$ and $V$, namely $XU$ and $YV$.

After obtaining the BWE, we find the nearest neighbor target word for every source word in the BWE space using the cross-domain similarity local scaling (CSLS) metric (Conneau et al., 2017b), which produces a word-to-word translation dictionary. We use this dictionary to translate the source training data into the target language, and simply copy the label for each word, which yields transferred training data in the target language. We train an NER model on this transferred data as our preliminary model. Going forward, we refer to the use of cross-lingual transferred data as CT.

### 2.2 Entity-Targeted Active Learning

After training a model using cross-lingual transfer learning, we start the active learning process based on this model's outputs. We begin by training a NER model $\Theta$ using the above model's outputs as training data. Using this trained model, our proposed entity-targeted active learning strategy, referred as ETAL, then selects the most informative spans from a corpus $D$ of unlabeled sequences. Given an unlabeled sequence $s$, ETAL first selects a span of tokens $s_i^j = s_i \cdots s_j$ such that $s_i^j$ is a likely named entity, where $i, j \in [0, |s|]$. Then, in order to obtain highly informative spans across $D$, ETAL computes the entropy $H$ for each occurrence of the span $s_i^j$ in $D$ and then aggregates them over the entire corpus $D$, given by:

$$H_{aggregate}(s_i^j) = \sum_{x_i^j \in D} H(x_i^j)\mathbb{1}(x_i^j = s_i^j)$$

where $x$ is an unlabeled sequence in $D$. Finally, the spans $s_i^j$ with the highest aggregate uncertainty $H_{aggregate}$ are selected for manual annotation.

We now describe the procedure for calculating $H(x_i^j)$, which is the entropy of a span $x_i^j$ being a likely entity. Given an unlabeled sequence $x$, the trained NER model $\Theta$ is used for computing the marginal probabilities $p_\theta(y_i|\mathbf{x})$ for each token $x_i$ across all possible labels $y_i \in Y$ using the forward-backward algorithm (Rabiner, 1989), where $Y$ is the set of all labels. Using these marginals we calculate the entropy of a given span $x_i^j$ being an entity as shown in Algorithm 1.

**Algorithm 1** Entity-Targeted Active Learning

*B* ← *label-set denoting beginning of an entity*
*I* ← *label-set denoting inside of an entity*
*O* ← *outside of an entity span*
$p_\theta(y_i|\mathbf{x})$ ← *marginal probability of label* $y_i$ *for token* $x_i$

**for** $i \leftarrow 1...len(x), j = 1$ **do**
  $p_{span}^{ij} = \sum_{y \in B} p_\theta(y_i|\mathbf{x})$
  **for** $j \leftarrow i+1...len(x)$ **do**
    $p_{entity} = p_{span}^{ij} * p_\theta(O_j|\mathbf{x})$
    $H =$ *entropy* $(p_{entity})$
    **if** $H > H_{threshold}$ **then**
      $H_{aggregate}(x_i^j) += H$
    $p_{span}^{ij} = p_{span}^{ij} * \sum_{y \in I} p_\theta(y_j|\mathbf{x})$

Let $B$ denote the set of labels indicating beginning of an entity, $I$ the set of labels indicating inside of an entity and $O$ denoting outside of an entity. First, we compute the probability of a span $x_i^j$ being an entity, starting with the token $i$, by marginalizing $p_\theta(y_i|\mathbf{x})$ over all labels in $B$, denoted as $p_{span}^{ij}$. Since an entity can span multiple tokens, for each subsequent token $j$ being part of that entity, we marginalize $p_\theta(y_j|\mathbf{x})$ over all labels in $I$ and combine it with $p_{span}^{ij}$. Finally, we compute $p_{entity} = p_{span}^{ij} * p_\theta(O_j|\mathbf{x})$, which denotes end of a likely entity. Since we use the marginal probability for computing $p_{entity}$, it already factors in the transition probability between tags. Thus, any invalid sequences such as $B_{PER}I_{ORG}$ have low scores. Since contiguous spans have overlapping tokens, using dynamic programming (DP) to compute $p_{span}^{ij}$ avoids an exponential computation when considering all possible spans in a sequence. Using $p_{entity}$, we compute the entropy $H$ and only consider the spans having $H$ higher than a pre-defined threshold $H_{threshold}$. The reason for this thresholding is purely for computational purposes as it allows us to discard all spans

that have a very low probability of being an entity, keeping the number of spans actually stored in memory low. As mentioned above, we aggregate the entropy of spans $H_{aggregate}$ over the entire unlabeled set, thus combining uncertainty sampling with a bias towards high frequency entities.

Using this strategy, we select subspans in each sequence for annotation. The annotator only needs to assign named entity types to the chosen subspans, adjust the span boundary if needed, and ignore the rest of the sequence, saving much effort.

### 2.3 Training the NER model

With the newly obtained training data from active learning, we attempt to improve the original transferred model. In this section, we first describe our model architecture, and try to address: 1) how to train the NER model effectively with partially annotated sequences? 2) what training scheme is best suited to improve the transferred model?

#### 2.3.1 Model Architecture

Our NER model is a BiLSTM-CNN-CRF model based on Ma and Hovy (2016) consisting of: a character-level CNN, that allows the model to capture subword information; a word-level BiLSTM, that consumes word embeddings and produces context sensitive hidden representations; and a linear-chain CRF layer that models the dependency between labels for inference. We use the above model for training the initial NER model on the transferred data as well as for re-training the model on the data acquired from active learning.

#### 2.3.2 PARTIAL-CRF

Active learning with span-based strategies such as ETAL, produces a training dataset of partially labeled sequences. To train the NER model on these partially labeled sequences, we take inspiration from Bellare and McCallum (2007); Tsuboi et al. (2008) and use a constrained CRF decoder. Normally, CRF computes the likelihood of a label sequence $\mathbf{y}$ given a sequence $\mathbf{x}$ as follows:

$$p_\theta(\mathbf{y}|\mathbf{x}) = \frac{\prod_{t=1}^{T} \psi_i(y_{t-1}, y_t, \mathbf{x}, t)}{Z(\mathbf{x})}$$

$$Z(\mathbf{x}) = \sum_{\mathbf{y} \in \mathbf{Y}(T)} \prod_{t=1}^{T} \psi_i(y_{t-1}, y_t, \mathbf{x}, t)$$

where $T$ is the length of the sequence, $\mathbf{Y}(T)$ denotes the set of all possible label sequences with length $T$, and $\psi_i(y_{t-1}, y_t, \mathbf{x}) = \exp(\mathbf{W}^T_{y_{t-1},y_t}\mathbf{x}_i + \mathbf{b}_{y_{t-1},y_t})$ is the energy function. To compute the likelihood of a sequence where some labels are unknown, we use a constrained CRF which marginalizes out the un-annotated tokens. Specifically, let $\mathbf{Y_L}$ denote the set of all possible sequences that include the partial annotations (for un-annotated tokens, all labels are possible), and we compute the likelihood as: $p_\theta(\mathbf{Y_L}|\mathbf{x}) = \sum_{\mathbf{y} \in \mathbf{Y_L}} p_\theta(\mathbf{y}|\mathbf{x})$, referred as PARTIAL-CRF.

#### 2.3.3 Training Scheme

To improve our model with the newly labeled data, we directly fine-tune the initial model, trained on the transferred data, on the data acquired through active learning, referred as FINETUNE. Each active learning run produces more labeled data, for which this training procedure is repeated again. We also compare the NER performance using two other training schemes: CORPUSAUG, where we train the model on the concatenated corpus of transferred data and the newly acquired data, and CORPUSAUG+FINETUNE, where we additionally fine-tune the model trained using CORPUSAUG on just the newly acquired data.

## 3 Experiments

In this section, we evaluate the effectiveness of our proposed strategy in both simulated (§3.2) and human-annotation experiments (§3.3).

### 3.1 Experimental Settings

**Datasets:** The first evaluation set includes the benchmark CoNLL 2002 and 2003 NER datasets (Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003) for Spanish (from the Romance family), Dutch and German (like English, from the Germanic family). We use the standard corpus splits for train/dev/test. The second evaluation set is for the low-resource setting where we use the Indonesian (from the Austronesian family), Hindi (from the Indo-Aryan family) and Spanish datasets released by the Linguistic Data Consortium (LDC).[2] We generate the train/dev/test split by random sampling. Details of the corpus statistics are in the Appendix.

**English-transferred Data:** We use the same experimental settings and resources as described in Xie et al. (2018) to get the translations of the English training data for each target language.

[2]LDC2017E62,LDC2016E97,LDC2017E66

**Active Learning Setup:** As described in Section §2.2, a DP-based algorithm is employed to select the uncertain entity spans which runs for all n-grams having length $<= 5$. This length was approximated by computing the 90th percentile on the length of entities in the English training data. $H_{threshold}$ is a hyper-parameter set to 1e-8. The details of the NER model hyper-parameters can be found in the Appendix.

### 3.2 Simulation Experiments

**Setup:** We use cross-lingual transfer (§2.1) to train our initial NER model and test on the target language. This is the same setting as Xie et al. (2018) and serves as our baseline. Then we use several active learning strategies to select data for manual annotation using this trained NER model. We compare our proposed ETAL strategy with the following baseline strategies:

**SAL:** Select whole sequences for which the model has least confidence in the most likely labeling (Culotta and McCallum, 2005).

**CFEAL:** Select least confident spans within a sequence using the confidence field estimation method (Culotta and McCallum, 2004).

**RAND:** Select spans randomly from the unlabeled set for annotation.

In this experimental setting, we simulate manual annotation by using gold labels for the data selected by active learning. At each subsequent run, we annotate 200 tokens and fine-tune the NER model on all the data acquired so far, which is then used to select data for the next run of annotation.

#### 3.2.1 Results

Figure 2 summarizes the results for all datasets across the different experimental settings. Each data-point on the x-axis corresponds to the NER performance after annotating 200 additional tokens. CT denotes using cross-lingual transferred data to train the initial NER model for both kick-starting the active learning process and also for fine-tuning the NER model on the newly-acquired data. PARTIAL-CRF/FULL-CRF denote the type of CRF decoder used in the NER model. Throughout this paper, we report results averaged across all active learning runs unless otherwise noted. Individual scores are reported in the Appendix.

As can be seen in the figure, our proposed recipe ETAL+PARTIAL-CRF+CT outperforms the previous active learning baselines for all the datasets. Holding the other two components of CT and PARTIAL-CRF constant, we conduct experiments to compare the different active learning strategies, which are denoted by the solid lines in Figure 2. We see that ETAL outperforms the other strategies by a significant margin for both the CoNLL datasets: German (+6.1 F1), Spanish (+5.3 F1), Dutch (+6.3 F1) and the LDC datasets: Hindi (+9.3 F1), Indonesian (+9.0 F1), Spanish (+7.5 F1), at the end of all runs. Furthermore, even with just one-tenth annotated tokens, the proposed recipe is only (avg.) -5.2 F1 behind the model trained using all labeled data, denoted by SUPERVISED ALL. Although CFEAL also selects informative spans, ETAL outperforms it because ETAL is optimized to select likely entities, causing more entities to be annotated for Hindi (+43), Indonesian (+207), Spanish-CoNLL (+1579), German (+906), Dutch (+836), except for Spanish-LDC (-184). Despite fully labeled data being adding in SAL, ETAL outperforms it because SAL selects longer sentences with fewer entities: Hindi (-934), Indonesian (-1290), Spanish-LDC (-527), Spanish-CoNLL (-2395), German (-2086), Dutch (-2213).

| Dataset | Tokens | ETAL | SAL | RAND | CFEAL |
|---|---|---|---|---|---|
| Hindi LDC | 200 | 54.8 ± 2.6 | 49.6 ± 2.8 | 50.2 ± 0.4 | 50.4 ± 2.8 |
| | 600 | 64.7 ± 2.6 | 51.5 ± 2.9 | 56.1 ± 2.6 | 54.3 ± 2.5 |
| | 1200 | 69.9 ± 2.5 | 56.6 ± 2.7 | 56.8 ± 2.6 | 64.4 ± 2.7 |
| Indonesian LDC | 200 | **47.4± 2.6** | **47.9 ± 2.4** | **46.8 ± 2.3** | **48.5 ± 2.3** |
| | 600 | 54.5 ± 2.4 | 44.5 ± 2.2 | 47.2 ± 2.2 | 46.0 ± 2.3 |
| | 1200 | 60.5 ± 2.3 | 44.7 ± 2.3 | 51.9 ± 2.3 | 49.5 ± 2.3 |
| Spanish LDC | 200 | 66.3 ± 3.8 | 62.0 ± 3.6 | 61.2 ± 1.2 | 62.5 ± 3.7 |
| | 600 | 65.8 ± 4.1 | 62.5 ± 3.7 | 61.9 ± 2.0 | 63.8 ± 3.7 |
| | 1200 | 78.9 ± 3.5 | 62.3 ± 3.6 | 64.6 ± 4.0 | 68.6 ± 3.9 |

Table 1: Variance analysis for significance testing of different active learning systems using paired bootstrap resampling. ± denotes the 95% confidence intervals. Systems which are not statistically significant than the best system ETAL are in bold. The CoNLL datasets reflect the same observation, as can be seen in Appendix.

From Figure 2 we see that ETAL performs better than the baselines across multiple runs. To verify that this is not an artifact of randomness in the test data, we use a paired bootstrap resampling method, as illustrated in Koehn (2004), to compare SAL, CFEAL, RAND with ETAL. For each system, we compute the F1 score on randomly sampled 50% of the data and perform 10k bootstrapping steps at three active learning runs. From Table 1 we see that the baselines are significantly worse than ETAL at 600 and 1200 annotated tokens.

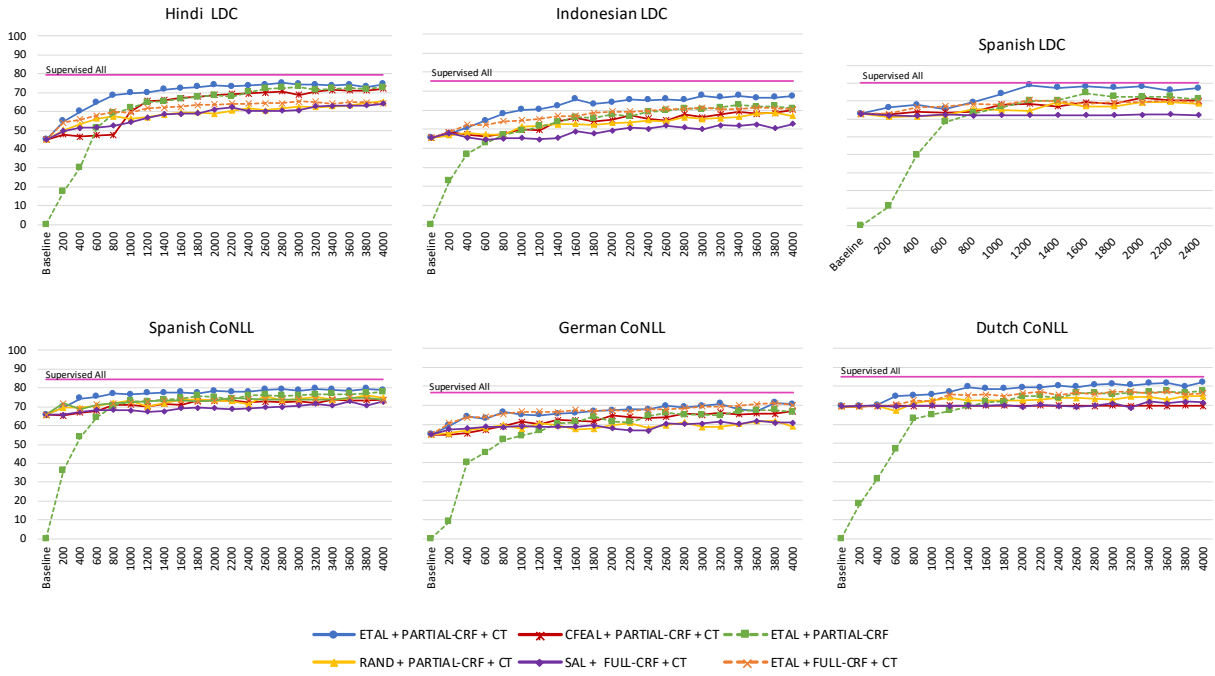


Figure 2: Comparison of the NER performance trained with the FineTune scheme, across six datasets. Solid lines compare the different active learning strategies. Dashed lines show the ablation experiments. The x-axis denotes the total number of tokens annotated and the y-axis denotes the F1 score.

### 3.2.2 Ablation Study

In order to study the contribution of CT and PARTIAL-CRF in improving the NER performance, we conduct the following ablation, denoted by dashed lines in Figure 2.

**CT:** We observe that the transferred data from English provides a good start to the NER model: 69.4 (Dutch), 63.0 (Spanish-LDC), 65.7 (Spanish-CoNLL), 54.7 (German), 45.4 (Indonesian), 45.0 (Hindi) F1. As expected, cross-lingual transfer helps more for the languages closely related to English which are Dutch, German, Spanish. For this ablation, we train a ETAL+PARTIAL-CRF where no transferred data is used. Therefore, to create the seed data, we randomly annotate 200 tokens in the target language and thereafter use ETAL. We observe that as more in-domain data is acquired, the un-transferred setting soon approaches the transferred setting ETAL+PARTIAL-CRF+CT suggesting that an efficient annotation strategy can help close the gap between these two systems with as few as ∼1000 tokens (avg.).

**PARTIAL-CRF:** We study the effect of using the original CRF (FULL-CRF) instead of the PARTIAL-CRF for training with partially labeled data. Since the former requires fully labeled sequences, the un-annotated tokens in a sequence are labeled with the model predictions. We see from Figure 2 that the ETAL+FULL-CRF+CT performs worse (avg. -4.1 F1) than ETAL+PARTIAL-CRF+CT. This is because the FULL-CRF significantly hurts the recall, as much as by an average of -11.0 points for Hindi, -1.4 for Indonesian, -7.4 for Spanish-LDC, -3.3 for German, -3.7 for Dutch, -4.8 for Spanish CoNLL.

### 3.2.3 Comparison of Training Schemes

We experiment with different NER training regimes (described in §2.3.3) for ETAL. We observe that generally fine-tuning not only speeds up the training but also gives better performance than CORPUSAUG. For brevity of space, we compare results for two languages in Figure 3:[3] Dutch, a relative of English, and Hindi, a distant language. We see that FINETUNE performs better for Hindi whereas CORPUSAUG+FINETUNE performs better for Dutch. This is because Dutch is closely related to English and benefits the most from the transferred data being explicitly augmented. Whereas for Hindi, which is typologically distant from English, the transferred data is noisy and thus the model doesn't gain much from the transferred data. Xie et al. (2018) make a similar observation in their

[3]The results for other datasets can be found in Appendix.

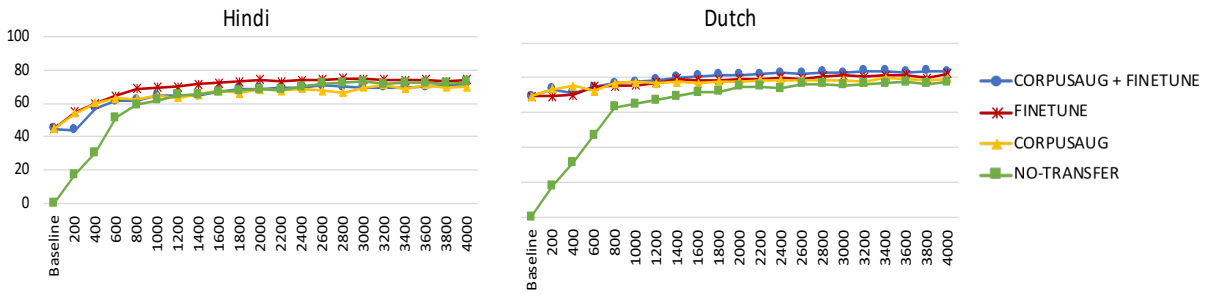


Figure 3: Comparison of the NER performance trained with different schemes for the ETAL strategy. The x-axis denotes the total number of tokens annotated and the y-axis denotes the F1 score.

experiments with German.

### 3.3 Human Annotation Experiments

**Setup:** We conduct human annotation experiments for Hindi, Indonesian and Spanish to understand whether ETAL helps reduce the annotation effort and improve annotation quality in practical settings. We compare ETAL with the full sequence strategy (SAL). We use six native speakers, two for each language, with different levels of familiarity with the NER task. Each annotator was provided with practice sessions to gain familiarity with the annotation guidelines and the user interface. The annotators annotated for 20 mins time for each strategy. For ETAL, the annotator was required to annotate single spans i.e each sequence contained one span of tokens. This involved assigning the correct label and adjusting the span boundary if required. For SAL, the annotator was required to annotate all possible entities in the sequence. We randomized the order in which the annotators had to annotate using ETAL and SAL strategy. Figure 5 illustrates the human annotation process for the ETAL strategy in the annotation interface. [4]

#### 3.3.1 Results

Table 2 records the results of the human annotation experiments. We first compare each annotator's annotation quality with respect to the oracle under both ETAL and SAL, denoted by *Annotator Performance*. We see that both Hindi and Spanish annotators have higher annotation quality using ETAL. This is because by selecting possible entity spans, ETAL not only saves effort on searching the entities in a sequence but also allows the annotators to read less overall and concentrate more on the things that they do read, as seen in Figure 4(a). However, for SAL we see that the annotator missed a likely entity because they focused on the other more salient entities in the sequence. For Indonesian, we see an opposite trend due to several inconsistencies in the gold labels. The most common inconsistency being when a common noun is part of an entity. For instance, the gold standard annotates the span *Kabupaten Bogor* as an entity where *Kabupaten* means "district". Whereas for ~~*Kabupaten*~~ *Aceh tengah*, the gold standard does not include *Kabupaten*. Similarly, the same span *gunung krakatau* is annotated inconsistently across different mentions where sometimes they exclude the *gunung* (mountain) token.Since the annotators referred to these examples during their practice session, their annotations had similar inconsistencies. This issue affects ETAL more than SAL because ETAL selects more entities for annotation.

| | **Annotator Performance** | | **Test Performance (# annotated tokens)** | | |
|---|---|---|---|---|---|
| | ETAL | SAL | ETAL | SAL | SAL-Full |
| HI-1 | **78.8** | 63.7 | **50.4** (326) | 44.2 (326) | 53.3 (1894) |
| HI-2 | **82.7** | 72.2 | **49.1** (234) | 45.9 (234) | 55.6 (2242) |
| ID-1 | 66.1 | **77.8** | **50.4** (425) | 45.8 (425) | 51.3 (3232) |
| ID-2 | 73.0 | **79.5** | **51.2** (251) | 46.5 (251) | 54.0 (2874) |
| ES-1 | **79.7** | 75.0 | **63.7** (204) | 62.2 (204) | 64.6 (2134) |
| ES-2 | **83.1** | 70.4 | **63.8** (199) | 62.2 (199) | 62.6 (2134) |

Table 2: Annotator performance measures F1 of each annotator with respect to the oracle. Test Performance measures the NER F1 scores using the annotations as training data. The number in the brackets denote the number of annotated tokens used for training the NER model. ES:Spanish, HI:Hindi, ID: Indonesian.

The *Test Performance* compares the performance of the NER models trained with these annotations. The number in the brackets denotes the total number of annotated tokens used for train-

[4]The code for the annotation interface can be found at https://gitlab.com/cmu_ariel/ariel-annotation-interface.

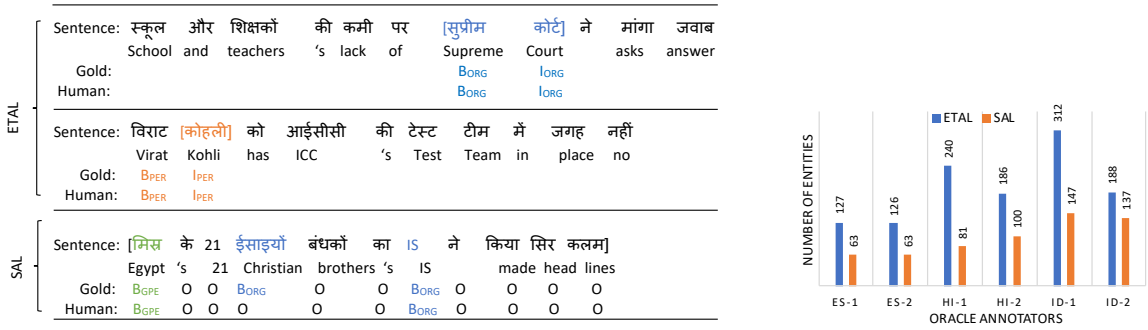


(a) (b)

Figure 4: (a) Examples from Hindi human annotation experiments for both ETAL and SAL. Square brackets denote the spans (for ETAL) or the entire sequence (for SAL) selected by the respective active learning strategy. (b) Comparing the number of entities in the data selected by ETAL and SAL, as annotated by oracle.

ing the NER model. We observe that SAL has a larger number of annotated tokens than ETAL. This is because most sequences selected by SAL did not have any entities. Since "not-an-entity" is the default label in the annotation interface, no operation is required for annotating these, allowing for more tokens to be annotated per unit times. When we count the number of entities present in the data selected by the two strategies, we see in Figure 4(b) that data selected by ETAL has a significantly larger number of entities than SAL, across all the 6 annotation experiments. Therefore, we first compare the NER performance on the same number of annotated tokens. From Table 2 we see that under this setting ETAL outperforms SAL, similar to the simulation results. We note that when we consider all the annotated tokens, SAL (denoted by SAL-FULL) has slightly better results. However, despite having 6 times fewer annotated tokens, the difference between ETAL and SAL-FULL is (avg.) 2.1 F1. This suggests that ETAL can achieve competitive performance with fewer annotations.

From both the simulation and human experiments, we can conclude that a targeted annotation strategy such as ETAL achieves competitive performance with less manual effort while maintaining high annotation quality. Given that ETAL can help find twice as many entities as SAL, a potential application of ETAL can also be for creating a high-quality entity gazetteer under a short time budget. Since a naive strategy of SAL allows for more labelled data to be acquired in the same amount of time, in the future we plan to explore mixed-mode annotation where we choose either full sequences or spans for annotation.

## 4 Related Work

**Cross-Lingual Transfer:** Transferring knowledge from high-resource languages has been extensively used for improving low-resource NER. More common approaches rely on annotation projection methods where annotations in source language are projected to the target language using parallel corpora (Zitouni and Florian, 2008; Ehrmann et al., 2011) or bilingual dictionaries (Xie et al., 2018; Mayhew et al., 2017). Cross-lingual word embeddings (Bharadwaj et al., 2016; Chaudhary et al., 2018) also provide a way to leverage annotations from related languages.

**Active Learning (AL):** AL has been widely explored for many NLP tasks- NER: Shen et al. (2017) explore token-level annotation strategies, Chen et al. (2015) present a study on AL for clinical NER; Baldridge and Palmer (2009) evaluate how well AL works with annotator expertise and label suggestions, Garrette and Baldridge (2013) study type and token based strategies for low-resource languages. Settles and Craven (2008) present a nice survey on the different AL strategies for sequence labeling tasks, whereas Marcheggiani and Artieres (2014) discuss the strategies for acquiring partially labeled data. Wanvarie et al. (2011); Neubig et al. (2011); Sperber et al. (2014) show the advantages of training a model on this partially labeled data. All above methods focus on either token or full sequence annotation.

The most similar work to ours perhaps is that of

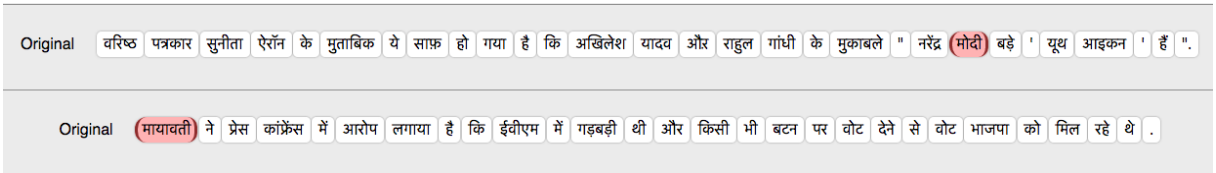


(a) Selected spans using ETAL strategy are highlighted for the human annotator to annotate.

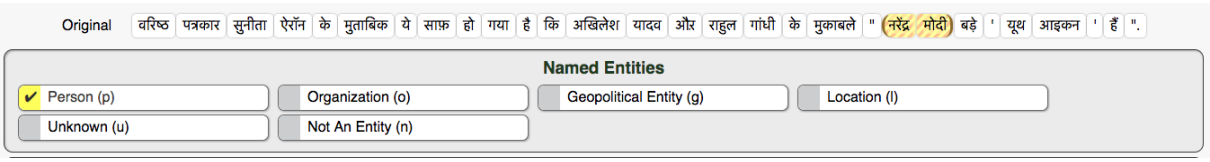


(b) Human annotator correcting the span boundary and assigning the correct entity type.

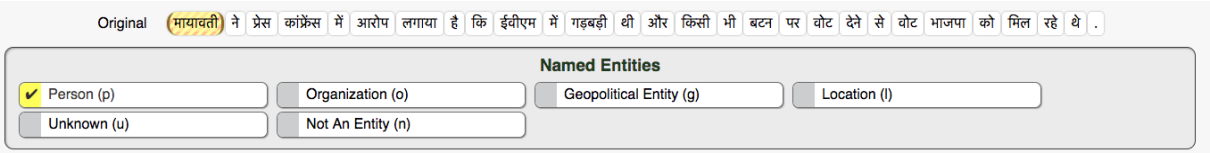


(c) Human annotator assigning the correct entity type only since selected span boundary is correct.

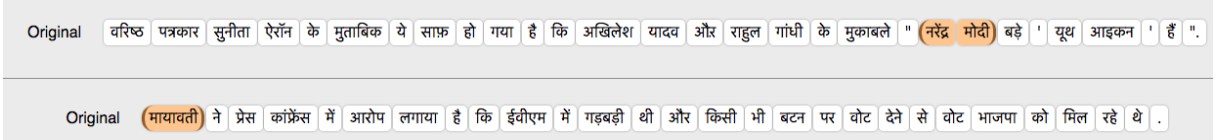


(d) Partially-annotated sequences after being annotated by the human annotator.

Figure 5: Example of the human annotation process for Hindi.

Fang and Cohn (2017), which selects informative word types for low-resource POS tagging. However, their method requires the annotator to annotate single tokens, which is not trivially applicable for multi-word entities in practical settings.

## 5 Conclusion

In this paper, we presented a study on how to efficiently bootstrap NER systems for low-resource languages using a combination of cross-lingual transfer learning and active learning. We conducted both simulated and human annotation experiments across different languages and found that: 1) cross-lingual transfer is a powerful tool, constantly beating systems without using transfer; 2) our proposed recipe works the best among known active learning baselines; 3) our proposed active learning strategy saves annotator much effort while ensuring high quality. In future, to account for different levels of annotator expertise, we plan to combine proactive learning (Li et al., 2017) with our proposed method.

## Acknowledgement

The authors would like to thank Sachin Kumar, Kundan Krishna, Aldrian Obaja Muis, Shirley Anugrah Hayati, Rodolfo Vega and Ramon Sanabria for participating in the human annotation experiments. This work is sponsored by Defense Advanced Research Projects Agency Information Innovation Office (I2O). Program: Low Resource Languages for Emergent Incidents (LORELEI). Issued by DARPA/I2O under Contract No. HR0011-15-C0114. This research was supported also in part by DARPA grant FA8750-18-2-0018 funded under the AIDA program. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

## References


Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A Smith. 2016. Massively multilingual word embeddings. *arXiv preprint arXiv:1602.01925*.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462, Vancouver, Canada. Association for Computational Linguistics.

Jason Baldridge and Alexis Palmer. 2009. How well does active learning *actually* work? Time-based evaluation of cost-reduction strategies for language documentation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 296–305, Singapore. Association for Computational Linguistics.

Kedar Bellare and Andrew McCallum. 2007. Learning extractors from unlabeled text using relevant databases. In *Sixth international workshop on information integration on the web*.

Akash Bharadwaj, David Mortensen, Chris Dyer, and Jaime Carbonell. 2016. Phonologically aware neural model for named entity recognition in low resource transfer settings. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1462–1472.

Aditi Chaudhary, Chunting Zhou, Lori Levin, Graham Neubig, David R. Mortensen, and Jaime Carbonell. 2018. Adapting word embeddings to new languages with morphological and phonological subword representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3285–3295. Association for Computational Linguistics.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada. Association for Computational Linguistics.

Yukun Chen, Thomas A Lasko, Qiaozhu Mei, Joshua C Denny, and Hua Xu. 2015. A study of active learning methods for named entity recognition in clinical text. *Journal of biomedical informatics*, 58:11–18.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017a. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017b. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Aron Culotta and Andrew McCallum. 2004. Confidence estimation for information extraction. In *Proceedings of HLT-NAACL 2004: Short Papers*, pages 109–112. Association for Computational Linguistics.

Aron Culotta and Andrew McCallum. 2005. Reducing labeling effort for structured prediction tasks. In *Association for the Advancement of Artificial Intelligence (AAAI)*, volume 5, pages 746–751.

Maud Ehrmann, Marco Turchi, and Ralf Steinberger. 2011. Building a multilingual named entity-annotated corpus using annotation projection. In *Proceedings of the International Conference Recent Advances in Natural Language Processing 2011*, pages 118–124.

Meng Fang and Trevor Cohn. 2017. Model transfer for tagging low-resource languages using a bilingual dictionary. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 587–593, Vancouver, Canada. Association for Computational Linguistics.

Dan Garrette and Jason Baldridge. 2013. Learning a part-of-speech tagger from two hours of annotation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 138–147, Atlanta, Georgia. Association for Computational Linguistics.

Xianpei Han and Le Sun. 2012. An entity-topic model for entity linking. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 105–115. Association for Computational Linguistics.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 conference on empirical methods in natural language processing*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.

Maolin Li, Nhung Nguyen, and Sophia Ananiadou. 2017. Proactive learning for named entity recognition. In *BioNLP 2017*, pages 117–125, Vancouver, Canada,. Association for Computational Linguistics.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074. Association for Computational Linguistics.

Diego Marcheggiani and Thierry Artieres. 2014. An experimental comparison of active learning strategies for partially labeled sequences. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 898–906.

Stephen Mayhew, Chen-Tse Tsai, and Dan Roth. 2017. Cheap translation for cross-lingual named entity recognition. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2536–2545.

Graham Neubig, Yosuke Nakata, and Shinsuke Mori. 2011. Pointwise prediction for robust, adaptable japanese morphological analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 529–533. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Lawrence R Rabiner. 1989. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Burr Settles and Mark Craven. 2008. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1070–1079. Association for Computational Linguistics.

Yanyao Shen, Hyokun Yun, Zachary Lipton, Yakov Kronrod, and Animashree Anandkumar. 2017. Deep active learning for named entity recognition. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 252–256, Vancouver, Canada. Association for Computational Linguistics.

Matthias Sperber, Mirjam Simantzik, Graham Neubig, Satoshi Nakamura, and Alex Waibel. 2014. Segmentation for efficient supervised language annotation with an explicit cost-utility tradeoff. *Transactions of the Association for Computational Linguistics*, 2:169–180.

Erik F. Tjong Kim Sang. 2002. Introduction to the conll-2002 shared task: Language-independent named entity recognition. In *Proceedings of the 6th Conference on Natural Language Learning - Volume 20*, COLING-02, pages 1–4, Stroudsburg, PA, USA. Association for Computational Linguistics.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Yuta Tsuboi, Hisashi Kashima, Hiroki Oda, Shinsuke Mori, and Yuji Matsumoto. 2008. Training conditional random fields using incomplete annotations. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 897–904. Association for Computational Linguistics.

Dittaya Wanvarie, Hiroya Takamura, and Manabu Okumura. 2011. Active learning with subsequence sampling strategy for sequence labeling tasks. *Information and Media Technologies*, 6(3):680–700.

Jiateng Xie, Zhilin Yang, Graham Neubig, Noah A. Smith, and Jaime Carbonell. 2018. Neural cross-lingual named entity recognition with minimal resources. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 369–379, Brussels, Belgium. Association for Computational Linguistics.

Bishan Yang and Tom M. Mitchell. 2016. Joint extraction of events and entities within a document context. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 289–299, San Diego, California. Association for Computational Linguistics.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of the first international conference on Human language technology research*, pages 1–8. Association for Computational Linguistics.

Imed Zitouni and Radu Florian. 2008. Mention detection crossing the language barrier. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 600–609. Association for Computational Linguistics.

# Open Domain Web Keyphrase Extraction Beyond Language Modeling

**Lee Xiong  Chuan Hu  Chenyan Xiong  Daniel Campos  Arnold Overwijk**
Microsoft AI and Research
Redmond, WA 98052, USA
{lexion, chuah, cxiong, dacamp, arnoldo}@microsoft.com

## Abstract

This paper studies keyphrase extraction in real-world scenarios where documents are from diverse domains and have variant content quality. We curate and release OpenKP, a large scale open domain keyphrase extraction dataset with near one hundred thousand web documents and expert keyphrase annotations. To handle the variations of domain and content quality, we develop BLING-KPE, a neural keyphrase extraction model that goes beyond language understanding using visual presentations of documents and weak supervision from search queries. Experimental results on OpenKP confirm the effectiveness of BLING-KPE and the contributions of its neural architecture, visual features, and search log weak supervision. Zero-shot evaluations on DUC-2001 demonstrate the improved generalization ability of learning from the open domain data compared to a specific domain.

## 1 Introduction

Automatically extracting keyphrases that are salient to the document meanings is an essential step to semantic document understanding. An effective keyphrase extraction (KPE) system can benefit a wide range of natural language processing and information retrieval tasks (Turney, 2001; Hasan and Ng, 2014). Recent neural methods formulate the task as a document-to-keyphrase sequence-to-sequence task. These neural KPE models have shown promising results compared to previous systems (Chen et al., 2018; Meng et al., 2017; Ye and Wang, 2018).

Noticeably, the recent progress in neural KPE is mostly observed in documents originating from the scientific domain (Meng et al., 2017; Augenstein et al., 2017). Perhaps because the scientific domain has sufficient training data for these neural methods: Authors are in the practice of assigning keyphrases to their publications. In real-world scenarios, most potential applications of KPE deal with diverse documents originating from sparse sources that are rather different from scientific papers. They often include a much diverse document structure and reside in various domains whose contents target much wider audiences than scientists. It is unclear how well the neural methods trained in the scientific domain generalize to other domains and in real-world scenarios.

This paper focuses on the task of open domain web keyphrase extraction, which targets KPE for web documents without any restriction of the domain, quality, nor content of the documents. We curate and release a large scale open domain KPE dataset, OpenKP, which includes about one hundred thousand web documents with expert keyphrase annotations.[1] The web documents are randomly sampled from the English fraction of a large web corpus and reflect the characteristics of typical web pages, with large variation in their domains and content qualities. To the best of our knowledge, this will be the first publicly available open domain manually annotated keyphrase extraction dataset at this scale.

This paper develops BLING-KPE, **B**eyond **L**anguage Understand**ING** **K**ey**P**hrase **E**xtraction, that tackles the challenges of KPE in documents from variant domains and content qualities. BLING-KPE uses a convolutional transformer architecture to model the language properties in the document, while also goes beyond by introducing the visual representation of the document and weak supervision from search user clicks.

The *visual presentations* of the document, including the location, size, font, and HTML structure of each text piece in the document, are integrated as visual features to the word embeddings

[1] The dataset, resources, and future updates are available at aka.ms/BLING.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5175–5184,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

in BLING-KPE. BLING-KPE learns to model the visual representations together with the document language in its network.

The *weak supervision from search clicks* is formulated as a pre-training task: Query Prediction. It trains the model to predict which phrase in the document has been used as a "click query", a query that a user issued to search and click on the document. The click queries on a document reflect the user's perceptions of the relatedness and importance when searching the document and can be considered as pseudo keyphrases. Pre-training on this weak supervision brings in training signals available at scale in commercial search systems.

Our experiments on OpenKP demonstrate the effectiveness of BLING-KPE. It outperforms standard KPE baselines, recent neural approaches and a highly optimized commercial KPE system by large margins. Ablation studies show the contributions of the neural architecture, visual features, and search weak supervision to BLING-KPE; removing any of them significantly reduces its accuracy.

Another advantage of learning from real-world open domain documents is improved generalization ability. We conduct zero-shot evaluations on the DUC-2001 news KPE datasets (Wan and Xiao, 2008b), where neural KPE systems are evaluated without seeing any labels from their news articles. BLING-KPE trained on OpenKP is the only neural method that outperforms traditional non-neural KPE methods, while neural KPE systems trained on the scientific documents do not generalize well to the news domain due to the domain differences.

## 2 Related Work

The classic keyphrase extraction systems typically include two components: *candidate keyphrase extraction* and *keyphrase importance estimation* (Hasan and Ng, 2014). The candidate keyphrases are often extracted by heuristic rules, for example, finding phrases following certain POS tag sequences (Wan and Xiao, 2008b; Liu et al., 2009a; Mihalcea and Tarau, 2004), predefined lexical patterns (Nguyen and Phan, 2009; Medelyan et al., 2009), or using entities as candidate phrases (Grineva et al., 2009).

The importance of the candidate keyphrases can be estimated by unsupervised or supervised methods. The unsupervised methods leverage the graph structures between phrases in the document (Mihalcea and Tarau, 2004; Wan and Xiao, 2008a,b), and topic information from topic modeling (Grineva et al., 2009; Liu et al., 2009b, 2010). The supervised keyphrase selection methods formulate a classification or ranking task and combine features from phrase frequencies (Witten et al., 2005), document structures (Chen et al., 2005; Yih et al., 2006), and external resources such as Wikipedia (Medelyan et al., 2009) and query log (Yih et al., 2006).

Recently, neural techniques have been applied to keyphrase tasks. Meng et al. formulate a seq2seq learning task that learns to extract and generate the keyphrase sequence from the document sequence; they incorporate a copy mechanism to the seq2seq RNN to extract phrases in the generation process (CopyRNN) (Meng et al., 2017). Improving this seq2seq setup has been the focus of recent research, for example, adding diverse constraints to reduce the duplication of produced keyphrases (Yuan et al., 2018; Chen et al., 2018), bringing auxiliary tasks to reduce the needs of training data (Ye and Wang, 2018), and adding title information to improve model accuracy (Chen et al., 2019).

The recent neural KPE methods have shown strong performances on the scientific domain, where large scale training data is available from the author assigned keyphrases on papers (Meng et al., 2017). Such specific domain training data limits the model generalization ability. Chen et al. show the seq2seq keyphrase generation models trained on scientific papers do not generalize well to another domain (Chen et al., 2018).

In general, previous research finds automatic keyphrase extraction a challenging task: its state-of-the-art accuracy is much lower than other language processing tasks, while supervised methods do not necessarily outperform simple unsupervised ones. Hasan and Ng (2014) pointed out potential ways to improve automatic keyphrase extraction, including better incorporation of *background knowledge*, better handling *long documents*, and better *evaluation* schemes. BLING-KPE aims to address these challenges by incorporating pre-training as a form of *background knowledge*, visual information to improve *long document* modeling, and OpenKP as a large scale open domain *evaluation* benchmark.

## 3 Open Domain Keyphrase Benchmark

This section describes the curation of OpenKP and its notable characteristics.

### 3.1 Data Curation

**Documents** in OpenKP include about seventy thousand web pages sampled from the index of Bing search engine.[2] The sampling is conducted on the pool of pages seen by United State users between Nov 2018 and Feb 2019.

There is no restriction on the domain or type of documents. They can be content-oriented pages like news articles, multi-media pages from video sites, or indexing pages with many hyperlinks. OpenKP is designed to reflect the diverse properties of web documents in the internet.

**Keyphrase Labels** are generated by our expert annotators. For each document, they examine the rendered web page and manually label 1-3 keyphrases following these definitions:

- *Salience:* A keyphrase captures the essential meaning of the page with no ambiguity.
- *Extraction:* The keyphrase has to appear in the document.
- *Fine-Grained:* The keyphrase cannot be general topics, such as "Sports" and "Politics".
- *Correct & Succinct:* The keyphrase has to form a correct English noun phrase, while also cannot be clauses or sentences.

We use the extraction setting to ensure labeling consistency and to increase annotation speed, which is around 42 pages per hour.

**Expert Agreements.** Our annotation experts are trained employees dedicated to providing high-quality annotations on web documents. We follow standard practice in generating annotations for production systems, which included regular touch-points to understand the confusion, as well as updates on the judgment guidelines to resolve ambiguities.

To study the task difficulty, we had five judges each annotate the same 50 random URLs. We measure the pairwise agreements between experts at different depths by Exact Match on the whole keyphrase, as well as the overlap between select keyphrases' unigrams. The agreement between judges is listed in Table 1.

[2] A new OpenKP version with 150K documents is available at msmarco.org.

Table 1: The agreements between pairs of expert judges at different annotation depth. Exact and Unigram show the percentage of judge agreement on exact keyphrases and overlapped unigrams.

| Judge Depth | Exact Match | Unigram Match |
|---|---|---|
| Keyphrase@1 | 64.74% | 64.74% |
| Keyphrase@2 | 48.30% | 63.12% |
| Keyphrase@3 | 43.51% | 57.66% |

The results confirm that open domain keyphrase extraction is not an easy task. When measuring agreement for the top 3 keyphrases, our expert judges completely agree on about 43% of keyphrase pairs. Compared to the previous small scale annotations, for example, on DUC-2001's news articles (Wan and Xiao, 2008b), annotating web pages with diverse contents and domains are harder.

We manually examined these annotations and found two sources of disagreement: Chunking Variances and KP Choices.

*Chunking Variances* we define as two judges pick different boundaries of the same concept. For example, one judge may select "Protein Synthesis" as the keyphrase, and others may select "Protein" and "Synthesis" as two separate keyphrases. We found Chunking Variances consist of about 20% of disagreements. As shown in Table 1, the judge agreements is substantially higher on Unigram overlaps than on Exact matches, indicating that they may select chunks that overlap with each other but not exactly the same.

*KP Choices* we define as two judges pick different keyphrases. The judges agree mostly (64%) on the first entered keyphrase, as shown in Table 1. The variations on the second and third keyphrases are larger. However, we found the variations are more about which keyphrases they choose to enter, *not about whether a phrase is a keyphrase or not*. The variations on judge labels mostly reflect the missing positives in OpenKP; most of the keyphrases annotated by judges are correct. We can reduce the missing positives by a deeper annotation, i.e. ten keyphrases per document, or by labeling all candidate phrases with classification labels (Liu et al., 2018). However, that will significantly reduce the number of total documents in OpenKP, as each document costs much more to annotate. We chose the current design choice of OpenKP to favor a larger amount of training labels, which, in our experience, is more effective in

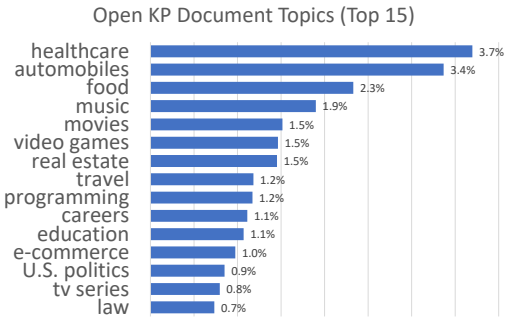


Figure 1: The most popular topics in OpenKP .

Table 2: Statistics of OpenKP used in our experiments. The new version on MSMARCO include 150K documents.

| Statistics | Mean | STD |
|---|---|---|
| Doc Length | 900.4 | 1494.4 |
| # of KPs per Doc | 1.8 | 0.8 |
| Keyphrase Length | 2.0 | 0.96 |
| Doc Vocabulary Size | 1.5M | n.a. |
| KP Vocabulary Size | 62K | n.a. |
| # of Documents | 68K | n.a. |
| # of Unique KPs | 99.6K | n.a. |

training deep neural models.

### 3.2 Data Characteristics

Table 2 lists the statistics of OpenKP. The document length is the length of the text parsed from the HTML of the web page, using a production HTML parser. The parsed texts will be released with the dataset. These statistics reflect the large variations in the document contents; their length varies a lot and share little common keyphrases, as shown by a large number of unique keyphrases.

We also leverage a production classifier to classify OpenKP documents into 5K predefined domains. The top 15 most popular classes and their distributions are shown in Figure 1. As expected, these documents have a large variation in their topic domains. The most popular domain, "healthcare", only covers 3.7% documents; the tenth most popular topic only covers 1% of documents. Moreover, the top 15 classes make up less than 25% of the entire dataset which showcases what a domain diverse dataset OpenKP is.

## 4 Keyphrase Extraction Model

This section describes the architecture, visual features, and weak supervision of BLING-KPE.

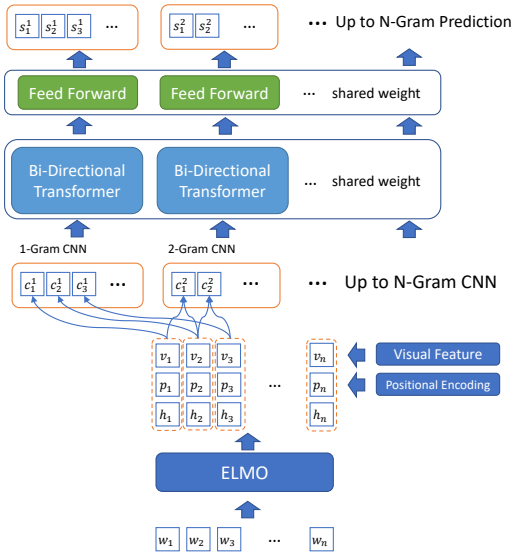


Figure 2: The BLING-KPE model architecture

### 4.1 Network Architecture

As shown in Figure 2, BLING-KPE is a keyphrase *extraction* model. It takes the word sequence of the document, $d = \{w_1, ...w_i, ...w_n\}$, and assigns keyphrase scores to its n-gram: $f(w_{i:i+k}, d)$. This process includes two main components: Hybrid Word Embedding and Convolutional Transformer.

**Hybrid Word Embedding.** BLING-KPE represents each word by its ELMo embedding, position embedding, and visual features.

The ELMo embedding brings the local contextual information:

$$\vec{h}_i = \text{ELMo}(w_i). \quad (1)$$

The standard pre-trained ELMo is used (Peters et al., 2018).

The position embedding models the location the word in the document content. It uses the standard sinusoidal position embedding (Vaswani et al., 2017):

$$\vec{\text{pos}}_i(2p) = sin(i/10000^{2p/P}), \quad (2)$$

$$\vec{\text{pos}}_i(2p+1) = cos(i/10000^{2p/P}). \quad (3)$$

The p-th dimension of the position embedding is a function of its position (i) and dimension (p).

The visual features represent the visual presentation of each word. We denote the visual feature as $\vec{v}_i$ and will describe its details in §4.2.

The hybrid word embedding is the concatenation of the three:

$$\vec{w}_i = \vec{h}_i \frown \vec{\text{pos}}_i \frown \vec{v}_i. \quad (4)$$

**Convolutional Transformer.** BLING-KPE uses a convolutional transformer architecture to model n-grams and their interactions.

It first composes the hybrid word embeddings to n-gram embeddings using CNNs. The embedding of i-th k-gram is calculated as

$$\vec{g}_i^k = \text{CNN}^k(\vec{w}_{i:i+k}), \tag{5}$$

where $k$ is the length of the n-gram, $1 \leq k \leq K$. $K$ is the maximum length of allowed candidate n-grams. Each k-gram has its own set of convolution filters $\text{CNN}^k$ with window size $k$ and stride 1.

It then models the interactions between k-grams using Transformer (Vaswani et al., 2017).

$$\vec{t}_i^k = \text{Transformer}_i(\vec{G}^k), \tag{6}$$

$$\vec{G}^k = \vec{g}_1^k \frown ... \frown \vec{g}_i^k ... \frown \vec{g}_{n-k+1}^k. \tag{7}$$

The sequence $\vec{G}^k$ is the concatenations of all k-gram embeddings. The Transformer models the self-attentions between k-grams and fuses them to global contextualized embeddings.

The Transformer is convolutional on all length $k$ of n-grams; the same parameters are used model the interactions between n-grams at each length, to reduce the parameter size. The intuition is that the interactions between bi-grams and that between tri-grams are not significantly different.

The final score of an n-gram is calculated by a feedforward layer upon the Transformer. Like the Transformer, the same feedforward layer is applied (convolutional) on all n-grams.

$$f(w_{i:i+k}, d) = \text{softmax}_{i,k}(s_i^k), \tag{8}$$

$$s_i^k = \text{Feedforward}(\vec{t}_i^k). \tag{9}$$

The softmax is taken over all possible n-grams at each position $i$ and each length $k$. The model decides the span location and length jointly.

**Learning.** The whole model is trained as a classification problem using cross-entropy loss:

$$l = \text{Cross-Entropy}(y_i^k, f(w_{i:i+k}, d)), \tag{10}$$

where $y_i^k$ is the label of whether the phrase $w_{i:i+k}$ is a keyphrase of the document.

### 4.2 Visual Features

We extract four groups of visual features for each word in the document.

- **Size** features include the height and width of the text block a word appears in.
- **Location** features include the 2-d location of the word in the rendered web page.
- **Font** feature includes the font size and whether the word is in Bold.
- **DOM** features include whether the word appears in "inline" or "block" HTML tags, also whether it is in a leaf node of the DOM tree.

Table 3: Visual Features. All features are extracted at per word level and the parent block level (the parent node of the word in the HTML DOM tree).

| Name | Dimension |
|---|---|
| Font Size | 1×2 |
| Text Block Size | 2×2 |
| Location in Rendered Page | 2×2 |
| Is Bold Font | 1×2 |
| Appear In Inline | 1×2 |
| Appear In Block | 1×2 |
| Appear In DOM Tree Leaf | 1×2 |

The full feature set is listed in Table 3. We double the features by including the same features from the word's parent block in the DOM tree. The visual features are included in the OpenKP releases.

### 4.3 Weak Supervisions from Search

An application of keyphrases is information retrieval. The extracted keyphrases are expected to capture the main topic of the document, thus can provide high quality document indexing terms (Gutwin et al., 1999) or new semantic ranking features (Xiong et al., 2018). Reversely, user clicks bring the user's perception of the document during the search and provide a large number of feedback signals for document understanding (Croft et al., 2010).

BLING-KPE leverages the user feedback signals as weak supervision, in the task of Query Prediction. Given the document $d$, BLING-KPE learns to predict its click queries $Q = \{q_1, ..., q_m\}$.

This pre-training step uses the same cross entropy loss:

$$l_{pre} = \text{Cross-Entropy}(y_i', f(q_i, d)), \tag{11}$$

where $y_i'$ indicates whether the query $q_i$ is a click query and also appears as an n-gram in the document $d$. The Query Prediction labels exist at scale in commercial search logs and provide a large number of pre-training signals.

Table 4: Statistics of Query Prediction Dataset. The data is from a sample of Bing search log in on week.

| Statistics | Mean | STD |
|---|---|---|
| Doc Length | 1211.8 | 1872.6 |
| # of Query per Doc | 1.32 | 0.76 |
| Query Length | 2.4 | 1.07 |
| Doc Vocabulary Size | 15M | n.a |
| Query Vocabulary Size | 383K | n.a |
| # of Documents | 1.6M | n.a. |
| # of Unique Queries | 1.5M | n.a. |

## 5 Experimental Methodology

**Datasets** used in our experiments include OpenKP, as described in §3, Query Prediction, and DUC-2001 (Wan and Xiao, 2008b).

The Query Prediction data is sampled from the Bing search log with navigational and offensive queries filtered out. We keep only the click queries that are included as an n-gram in the document to be consistent with OpenKP's extractive setting. The statistics of the sample is listed in Table 4.

DUC-2001 is the KPE extraction dataset on DUC news articles (Wan and Xiao, 2008b). It includes 309 news articles and on average 8 keyphrase per article.

We use random 80%-20% train-test splits on OpenKP and Query Prediction. On OpenKP, BLING-KPE is first pre-trained on Query Prediction and then further trained on its manual labels. There is no overlap between the documents in Query Prediction and OpenKP.

DUC-2001 uses the zero-shot evaluation setting from prior research (Meng et al., 2017; Chen et al., 2018); no labels in DUC-2001 are used to train nor validate the neural models. It tests neural models' generalization ability from the training domain to a different testing domain.

**Evaluation Metrics.** OpenKP and Query Prediction use **P**recision and **R**ecall@{1, 3, 5}. DUC-2001 uses F1@10, the same as prior research (Meng et al., 2017; Chen et al., 2018).

Statistically, significant improvements are evaluated by permutation test with $p<0.05$ on OpenKP and Query Prediction. The baselines on DUC-2001 reuse scores from previous results; the statistical significant test is not applicable as per document results are not shared.

**Baselines.** OpenKP and Query Prediction experiments compare `BLING-KPE` with: traditional KPE methods, production systems, and a neural baseline.

Table 5: Parameters to learn in BLING-KPE.

| Component | Dimensions |
|---|---|
| ELMo | pre-trained and frozen |
| Position Embedding | 256 |
| Visual Feature | 18 |
| N-gram CNN | 512 filter, 1-5 window size (ngram) |
| Transformer | 8 head, 512 hidden dimension |
| Feedforward | 512-relu-512-relu-1 |

Traditional KPE baselines include the follows.

- TFIDF is the unsupervised frequency based KPE system. The IDF scores are calculated on the corresponding corpus.
- TextRank is the popular graph-based unsupervised KPE model (Mihalcea and Tarau, 2004). Our in-house implementation is used.
- LeToR is the feature-based KPE model. It use LambdaMart (Burges, 2010) and standard KPE features, i.e. those in KEA (Witten et al., 2005).

The production baselines include two versions.

- PROD is our current feature-based production KPE system. It uses many carefully engineered features and LambdaMart.
- PROD (Body) is the same system but only uses the body text, i.e. the title is not used.

All these unsupervised and feature-based methods use the same keyphrase candidate selection system with PROD.

The neural baseline is CopyRNN (Meng et al., 2017). We use their open-source implementation and focus on the OpenKP dataset which is publicly available.

**Implementation Details.** Table 5 lists BLING-KPE parameters. The training uses Adam optimizer, learning rate 0.3 with logarithmic decreasing to 0.001, batch size 16, and 0.2 dropout probability in n-gram CNN, Transformer and feedforward layers. Learning takes about 2.5 hours (2 epochs) to converge on `OpenKPE` and about 13 hours (3 epochs) on Query Prediction, based on validation loss. In BLING-KPE, the maximum document length is 256 and documents are zero-padded or truncated to this length. Baselines use the original documents, except CopyRNN which works better with 256. The maximum n-gram length is set to five (K=5).

Table 6: Keyphrase Extraction Accuracy. **Bold** marks statistically significant improvements over all baselines.

| | OpenKP | | | | | | Query Prediction | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Method** | **P@1** | **R@1** | **P@3** | **R@3** | **P@5** | **R@5** | **P@1** | **R@1** | **P@3** | **R@3** | **P@5** | **R@5** |
| TFIDF | 0.283 | 0.150 | 0.184 | 0.284 | 0.137 | 0.347 | 0.403 | 0.332 | 0.204 | 0.491 | 0.133 | 0.526 |
| TextRank | 0.077 | 0.041 | 0.062 | 0.098 | 0.055 | 0.142 | 0.132 | 0.111 | 0.089 | 0.218 | 0.073 | 0.295 |
| LeToR | 0.301 | 0.158 | 0.173 | 0.268 | 0.127 | 0.324 | 0.328 | 0.271 | 0.169 | 0.406 | 0.119 | 0.471 |
| PROD | 0.353 | 0.188 | 0.195 | 0.299 | 0.131 | 0.331 | 0.376 | 0.308 | 0.197 | 0.468 | 0.129 | 0.505 |
| PROD (Body) | 0.214 | 0.094 | 0.130 | 0.196 | 0.094 | 0.234 | 0.353 | 0.287 | 0.191 | 0.454 | 0.125 | 0.492 |
| CopyRNN | 0.288 | 0.174 | 0.185 | 0.331 | 0.141 | 0.413 | – | – | – | – | – | – |
| BLING-KPE | **0.404** | **0.220** | **0.248** | **0.390** | **0.188** | **0.481** | **0.540** | **0.449** | **0.275** | **0.654** | **0.188** | **0.729** |

Table 7: Performance of BLING-KPE ablations. *Italic* marks statistically significant worse performances than Full Model.

| | OpenKP | | | | | | Query Prediction | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Method** | **P@1** | **R@1** | **P@3** | **R@3** | **P@5** | **R@5** | **P@1** | **R@1** | **P@3** | **R@3** | **P@5** | **R@5** |
| No ELMo | *0.270* | *0.145* | *0.172* | *0.271* | *0.132* | *0.347* | *0.323* | *0.274* | *0.189* | *0.450* | *0.136* | *0.527* |
| No Transformer | *0.389* | *0.211* | 0.247 | *0.385* | 0.189 | 0.481 | *0.489* | *0.407* | *0.258* | *0.618* | *0.178* | *0.698* |
| No Position | *0.394* | *0.213* | 0.247 | *0.386* | 0.187 | *0.475* | 0.543 | 0.452 | 0.281 | 0.666 | 0.191 | 0.742 |
| No Visual | *0.370* | *0.201* | *0.230* | *0.362* | *0.176* | *0.450* | *0.492* | *0.409* | *0.258* | *0.615* | *0.178* | *0.695* |
| No Pretraining | *0.369* | *0.198* | *0.236* | *0.367* | *0.181* | *0.460* | – | – | – | – | – | – |
| Full Model | 0.404 | 0.220 | 0.248 | 0.390 | 0.188 | 0.481 | 0.540 | 0.449 | 0.275 | 0.654 | 0.188 | 0.729 |

## 6 Evaluation Results

Three experiments are conducted to evaluate the accuracy of BLING-KPE, the source of its effectiveness, and its generalization ability.

### 6.1 Overall Accuracy

The overall extraction accuracy on OpenKP and Query Prediction is shown in Table 6.

TFIDF works well on both tasks. Frequency-based methods are often strong baselines in document representation tasks. LeToR performs better than its frequency feature TFIDF in OpenKP but worse on Query Prediction. Supervised methods are not necessarily stronger than unsupervised ones in KPE (Hasan and Ng, 2014). TextRank does not work well in our dataset; its word graph is likely misguided by the noisy contents.

PROD, our feature-based production system, outperforms all other baselines by large margins on OpenKP. It is expected as it is highly optimized with a lot of engineering efforts. Nonetheless, adapting a complex feature-based system to a new task/domain requires extra engineering work; directly applying it to the Query Prediction task does not work well. The feature-based Production system also needs the title information; PROD (Body) performs much worse than PROD.

CopyRNN performs relatively well on OpenKP, especially on later keyphases. The main challenge for CopyRNN is the low-quality and highly variant contents on the web. Real-world web pages are not cohesive nor well-written articles but include various structures such as lists, media captions, and text fragments. Modeling them as a word sequence is not ideal. The other differences are not as significant: The vocabulary size and training data size on Query Prediction are similar to CopyRNN's KP 40K dataset; CopyRNN performs better on keyphrase extraction than generation in KP20k (Meng et al., 2017).

BLING-KPE outperforms all other methods by large margins. The improvements are robust and significant on both tasks, both metrics, and on all depths. It achieves 0.404 P@1 on OpenKP and recovers 72% of clicked queries at depth 5 on Query Prediction. The sources of this effectiveness is studied in the next experiment.

### 6.2 Ablation Study

Table 7 shows ablation results on BLING-KPE's variations. Each variation removes a component and keeps all others unchanged.

**ELMo Embedding.** We first verify the effectiveness of using ELMo embedding by replacing ELMo with the WordPiece token embedding (Wu et al., 2016). The accuracy of this variation is much lower than the accuracy of the full model and others. The result is shown in the first row of Table 7. The context-aware word embedding is a necessary component of `BLING-KPE`.

**Network Architecture.** The second part of

Table 8: Performance on DUC-2001. Neural models are evaluated directly on DUC-2001 without fine-tuning on DUC labels. Results better than TFIDF are marked **Bold**.

| Method | F1@10 | Method | F1@10 |
|---|---|---|---|
| TFIDF | 0.270 | TopicRank | 0.154 |
| TextRank | 0.097 | KeyCluster | 0.140 |
| SingleRank | 0.256 | ExpandRank | 0.269 |
| **Trained by Scientific Papers** | | | |
| CopyRNN | 0.164 | CorrRNN | 0.173 |
| BLING-KPE (No Visual, No Pretraining) | | | 0.267 |
| **Trained by Open Domain Documents** | | | |
| BLING-KPE (No Visual, No Pretraining) | | | **0.282** |

Table 7 studies the contribution of Transformer and position embedding. Transformer contributes significantly to Query Prediction; with a lot of training data, the self-attention layers capture the global contexts between n-grams. But on OpenKP, its effectiveness is mostly observed on the first position. The position embedding barely helps, since real-world web pages are often not one text sequence.

**Beyond Language Understanding.** As shown in the second part of Table 7, both visual features and search pretraining contribute significantly to BLING-KPE's effectiveness. Without either of them, the accuracy drops significantly. Visual features even help on Query Prediction, though users issued the click queries and clicked on the documents *before* seeing its full page.

The crucial role of ELMo embeddings confirm the benefits of bringing background knowledge and general language understanding, in the format of pre-trained contextual embedding, in keyphrase extraction. The importance of visual features and search weak supervisions confirms the benefits of going beyond language understanding in modeling real-world web documents.

### 6.3 Generalization Ability

This experiment studies the generalization ability of BLING-KPE using the zero-shot evaluation on DUC-2001 (Meng et al., 2017; Chen et al., 2018). For fair comparisons, we use KP20K or OpenKP, the two public datasets, to train the No Visual & No Pretraining version of BLING-KPE, and evaluate on DUC-2001 directly. No labels in DUC are used to fine-tune the neural models.

To adjust to DUC's larger number of keyphrases, we apply the trained BLING-KPE on the 256-length chunks of DUC articles

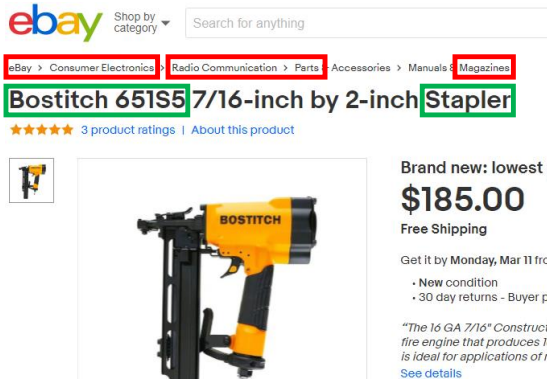


Figure 3: An example of visual feature contributions in BLING-KPE. The big green blocks are the keyphrases extracted with visual features. The small read blocks are those extracted without visual information.

and merge the extracted keyphrases using simple heuristics:

- Weighted Sum: Scores of the same keyphrase from different chunks are summed with weights $0.9^p$. P is the index of the chunk.
- Deduplication: A keyphrase is discarded if it is a sub-string of a top 1/4 ranked keyphrase.

The results are shown in Table 8. `BLING-KPE`, when trained with `OpenKP`, is the only neural method that outperforms TFIDF in this zero-shot evaluation. It outperforms previous neural methods by more than 60%, and itself when trained on KP20k, confirming the strong generalization ability of `BLING-KPE` and the training with `OpenKP`.

### 6.4 Discussion

Our manual case studies found many interesting examples that illustrate the advantage of modeling documents with visual information.

For example, in Figure 3, the page is annotated with "Bostitch 651S5", the product name, and "Stapler", the product type. Their salience is highlighted by larger and bold fonts, which are picked up by BLING-KPE. However, without the visual information, the product ontology names are extracted as keyphrases: they are meaningful concepts, correlated with the page content, and positioned at the beginning of the document—only that they appear less important in the web page by design.

## 7 Conclusion

This paper curates OpenKP, the first public large scale open domain keyphrase extraction benchmark to facilitate future research keyphrase extraction research in real-world scenarios. It also develops BLING-KPE, which leverages visual representation and search-based weak supervision to model real-world documents with variant contents, appearances, and diverse domains.

Our experiments demonstrate the robust improvements of BLING-KPE compared to previous approaches. Our studies showcase how BLING-KPE's language understanding, visual features and search weak supervision jointly deliver this effective performance, as well as its generalization ability to an unseen domain in zero-shot setting.

In the future, we plan to extend OpenKP with more annotated documents and connect it with downstream applications.

## References


Isabelle Augenstein, Mrinal Das, Sebastian Riedel, Lakshmi Vikraman, and Andrew McCallum. 2017. SemEval 2017 task 10: ScienceIE - extracting keyphrases and relations from scientific publications. In *SemEval@ACL*.

Chris J.C. Burges. 2010. From RankNet to LambdaRank to LambdaMART: An overview. MSR-TR-2010-82.

Jun Chen, Xiaoming Zhang, Yu Wu, Zhao Yan, and Zhoujun Li. 2018. Keyphrase generation with correlation constraints. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018*, pages 4057–4066.

Mo Chen, Jian-Tao Sun, Hua-Jun Zeng, and Kwok-Yan Lam. 2005. A practical system of keyphrase extraction for web pages. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 277–278. ACM.

Wang Chen, Yifan Gao, Jiani Zhang, Irwin King, and Michael R. Lyu. 2019. Title-guided encoding for keyphrase generation. In *The Thirty-Third AAAI Conference on Artificial Intelligence 2019*, pages 6268–6275.

W Bruce Croft, Donald Metzler, and Trevor Strohman. 2010. *Search Engines: Information Retrieval in Practice*. Addison-Wesley Reading.

Maria P. Grineva, Maxim N. Grinev, and Dmitry Lizorkin. 2009. Extracting key terms from noisy and multitheme documents. In *Proceedings of the 18th International Conference on World Wide Web, WWW 2009*, pages 661–670.

Carl Gutwin, Gordon W. Paynter, Ian H. Witten, Craig G. Nevill-Manning, and Eibe Frank. 1999. Improving browsing in digital libraries with keyphrase indexes. *Decision Support Systems*, 27:81–104.

Kazi Saidul Hasan and Vincent Ng. 2014. Automatic keyphrase extraction: A survey of the state of the art. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014*, pages 1262–1273.

Feifan Liu, Deana Pennell, Fei Liu, and Yang Liu. 2009a. Unsupervised approaches for automatic keyword extraction using meeting transcripts. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, HLT-NAACL 2009*, pages 620–628.

Zhengzhong Liu, Chenyan Xiong, Teruko Mitamura, and Eduard Hovy. 2018. Automatic event salience identification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018s*, pages 1226–1236.

Zhiyuan Liu, Wenyi Huang, Yabin Zheng, and Maosong Sun. 2010. Automatic keyphrase extraction via topic decomposition. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, EMNLP 2010*, pages 366–376.

Zhiyuan Liu, Peng Li, Yabin Zheng, and Maosong Sun. 2009b. Clustering to find exemplar terms for keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, EMNLP 2009*, pages 257–266.

Olena Medelyan, Eibe Frank, and Ian H. Witten. 2009. Human-competitive tagging using automatic keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, EMNLP 2009*, pages 1318–1327.

Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. 2017. Deep keyphrase generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017*, pages 582–592.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing , EMNLP 2004*, pages 404–411.

Chau Q. Nguyen and Tuoi T. Phan. 2009. An ontology-based approach for key phrase extraction. In *ACL 2009, Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP, ACL/IJCNLP 2009*, pages 181–184.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018*, pages 2227–2237.

Peter D. Turney. 2001. Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In *Machine Learning: EMCL 2001, 12th European Conference on Machine Learning, 2001, Proceedings*, pages 491–502.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems, NIPS 2017*, pages 5998–6008.

Xiaojun Wan and Jianguo Xiao. 2008a. CollabRank: Towards a collaborative approach to single-document keyphrase extraction. In *22nd International Conference on Computational Linguistics, Proceedings of the Conference, COLING 2008*, pages 969–976.

Xiaojun Wan and Jianguo Xiao. 2008b. Single document keyphrase extraction using neighborhood knowledge. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008*, pages 855–860.

Ian H Witten, Gordon W Paynter, Eibe Frank, Carl Gutwin, and Craig G Nevill-Manning. 2005. KEA: Practical automated keyphrase extraction. In *Design and Usability of Digital Libraries: Case Studies in the Asia Pacific*, pages 129–152. IGI Global.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Chenyan Xiong, Zhengzhong Liu, Jamie Callan, and Tie-Yan Liu. 2018. Towards better text understanding and retrieval through kernel entity salience modeling. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018*, pages 575–584.

Hai Ye and Lu Wang. 2018. Semi-supervised learning for neural keyphrase generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4142–4153.

Wen-tau Yih, Joshua Goodman, and Vitor R. Carvalho. 2006. Finding advertising keywords on web pages. In *Proceedings of the 15th international conference on World Wide Web, WWW 2006*, pages 213–222.

Xingdi Yuan, Tong Wang, Rui Meng, Khushboo Thaker, Daqing He, and Adam Trischler. 2018. Generating diverse numbers of diverse keyphrases. *CoRR*, abs/1810.05241.

# TuckER: Tensor Factorization for Knowledge Graph Completion

**Ivana Balažević**[1] **Carl Allen**[1] **Timothy M. Hospedales**[1,2]
[1] School of Informatics, University of Edinburgh, UK
[2] Samsung AI Centre, Cambridge, UK
{ivana.balazevic, carl.allen, t.hospedales}@ed.ac.uk

## Abstract

Knowledge graphs are structured representations of real world facts. However, they typically contain only a small subset of all possible facts. Link prediction is a task of inferring missing facts based on existing ones. We propose TuckER, a relatively straightforward but powerful linear model based on Tucker decomposition of the binary tensor representation of knowledge graph triples. TuckER outperforms previous state-of-the-art models across standard link prediction datasets, acting as a strong baseline for more elaborate models. We show that TuckER is a fully expressive model, derive sufficient bounds on its embedding dimensionalities and demonstrate that several previously introduced linear models can be viewed as special cases of TuckER.

## 1 Introduction

Vast amounts of information available in the world can be represented succinctly as *entities* and *relations* between them. *Knowledge graphs* are large, graph-structured databases which store facts in triple form $(e_s, r, e_o)$, with $e_s$ and $e_o$ representing subject and object entities and $r$ a relation. However, far from all available information is currently stored in existing knowledge graphs and manually adding new information is costly, which creates the need for algorithms that are able to automatically infer missing facts.

Knowledge graphs can be represented as a third-order binary tensor, where each element corresponds to a triple, 1 indicating a true fact and 0 indicating the unknown (either a false or a missing fact). The task of *link prediction* is to predict whether two entities are related, based on known facts already present in a knowledge graph, i.e. to infer which of the 0 entries in the tensor are indeed false, and which are missing but actually true.

A large number of approaches to link prediction so far have been linear, based on various methods of factorizing the third-order binary tensor (Nickel et al., 2011; Yang et al., 2015; Trouillon et al., 2016; Kazemi and Poole, 2018). Recently, state-of-the-art results have been achieved using non-linear convolutional models (Dettmers et al., 2018; Balažević et al., 2019). Despite achieving very good performance, the fundamental problem with deep, non-linear models is that they are non-transparent and poorly understood, as opposed to more mathematically principled and widely studied tensor decomposition models.

In this paper, we introduce TuckER (E stands for entities, R for relations), a straightforward linear model for link prediction on knowledge graphs, based on *Tucker decomposition* (Tucker, 1966) of the binary tensor of triples, acting as a strong baseline for more elaborate models. Tucker decomposition, used widely in machine learning (Schein et al., 2016; Ben-Younes et al., 2017; Yang and Hospedales, 2017), factorizes a tensor into a core tensor multiplied by a matrix along each mode. It can be thought of as a form of higher-order SVD in the special case where matrices are orthogonal and the core tensor is "all-orthogonal" (Kroonenberg and De Leeuw, 1980). In our case, rows of the matrices contain entity and relation embeddings, while entries of the core tensor determine the level of interaction between them. Subject and object entity embedding matrices are assumed equivalent, i.e. we make no distinction between the embeddings of an entity depending on whether it appears as a subject or as an object in a particular triple. Due to the low rank of the core tensor, TuckER benefits from *multi-task learning* by parameter sharing across relations.

A link prediction model should have enough expressive power to represent all relation types (e.g. symmetric, asymmetric, transitive). We thus show



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5185–5194, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

that TuckER is *fully expressive*, i.e. given any ground truth over the triples, there exists an assignment of values to the entity and relation embeddings that accurately separates the true triples from false ones. We also derive a dimensionality bound which guarantees full expressiveness.

Finally, we show that several previous state-of-the-art linear models, RESCAL (Nickel et al., 2011), DistMult (Yang et al., 2015), ComplEx (Trouillon et al., 2016) and SimplE (Kazemi and Poole, 2018), are special cases of TuckER.

In summary, key contributions of this paper are:

- proposing TuckER, a new *linear* model for link prediction on knowledge graphs, that is simple, expressive and achieves *state-of-the-art results* across all standard datasets;
- proving that TuckER is *fully expressive* and deriving a bound on the embedding dimensionality for full expressiveness; and
- showing how TuckER subsumes several previously proposed tensor factorization approaches to link prediction.

## 2 Related Work

Several *linear* models for link prediction have previously been proposed:

**RESCAL** (Nickel et al., 2011) optimizes a scoring function containing a bilinear product between subject and object entity vectors and a full rank relation matrix. Although a very expressive and powerful model, RESCAL is prone to overfitting due to its large number of parameters, which increases quadratically in the embedding dimension with the number of relations in a knowledge graph.

**DistMult** (Yang et al., 2015) is a special case of RESCAL with a diagonal matrix per relation, which reduces overfitting. However, the linear transformation performed on entity embedding vectors in DistMult is limited to a stretch. The binary tensor learned by DistMult is symmetric in the subject and object entity mode and thus DistMult cannot model asymmetric relations.

**ComplEx** (Trouillon et al., 2016) extends DistMult to the complex domain. Subject and object entity embeddings for the same entity are complex conjugates, which introduces asymmetry into the tensor decomposition and thus enables ComplEx to model asymmetric relations.

**SimplE** (Kazemi and Poole, 2018) is based on Canonical Polyadic (CP) decomposition (Hitchcock, 1927), in which subject and object entity embeddings for the same entity are independent (note that DistMult is a special case of CP). SimplE's scoring function alters CP to make subject and object entity embedding vectors dependent on each other by computing the average of two terms, first of which is a bilinear product of the subject entity head embedding, relation embedding and object entity tail embedding and the second is a bilinear product of the object entity head embedding, inverse relation embedding and subject entity tail embedding.

Recently, state-of-the-art results have been achieved with *non-linear* models:

**ConvE** (Dettmers et al., 2018) performs a global 2D convolution operation on the subject entity and relation embedding vectors, after they are reshaped to matrices and concatenated. The obtained feature maps are flattened, transformed through a linear layer, and the inner product is taken with all object entity vectors to generate a score for each triple. Whilst results achieved by ConvE are impressive, its reshaping and concatenating of vectors as well as using 2D convolution on word embeddings is unintuitive.

**HypER** (Balažević et al., 2019) is a simplified convolutional model, that uses a hypernetwork to generate 1D convolutional filters for each relation, extracting relation-specific features from subject entity embeddings. The authors show that convolution is a way of introducing sparsity and parameter tying and that HypER can be understood in terms of tensor factorization up to a non-linearity, thus placing HypER closer to the well established family of factorization models. The drawback of HypER is that it sets most elements of the core weight tensor to 0, which amounts to hard regularization, rather than letting the model learn which parameters to use via soft regularization.

Scoring functions of all models described above and TuckER are summarized in Table 1.

## 3 Background

Let $\mathcal{E}$ denote the set of all entities and $\mathcal{R}$ the set of all relations present in a knowledge graph. A triple is represented as $(e_s, r, e_o)$, with $e_s, e_o \in \mathcal{E}$ denoting subject and object entities respectively and $r \in \mathcal{R}$ the relation between them.

### 3.1 Link Prediction

In link prediction, we are given a subset of all true triples and the aim is to learn a *scoring function* $\phi$

| Model | Scoring Function | Relation Parameters | Space Complexity |
|---|---|---|---|
| RESCAL (Nickel et al., 2011) | $\mathbf{e}_s^\top \mathbf{W}_r \mathbf{e}_o$ | $\mathbf{W}_r \in \mathbb{R}^{{d_e}^2}$ | $\mathcal{O}(n_e d_e + n_r d_r^2)$ |
| DistMult (Yang et al., 2015) | $\langle \mathbf{e}_s, \mathbf{w}_r, \mathbf{e}_o \rangle$ | $\mathbf{w}_r \in \mathbb{R}^{d_e}$ | $\mathcal{O}(n_e d_e + n_r d_e)$ |
| ComplEx (Trouillon et al., 2016) | $\text{Re}(\langle \mathbf{e}_s, \mathbf{w}_r, \overline{\mathbf{e}}_o \rangle)$ | $\mathbf{w}_r \in \mathbb{C}^{d_e}$ | $\mathcal{O}(n_e d_e + n_r d_e)$ |
| ConvE (Dettmers et al., 2018) | $f(\text{vec}(f([\underline{\mathbf{e}}_s; \underline{\mathbf{w}}_r] * w))\mathbf{W})\mathbf{e}_o$ | $\mathbf{w}_r \in \mathbb{R}^{d_r}$ | $\mathcal{O}(n_e d_e + n_r d_r)$ |
| SimplE (Kazemi and Poole, 2018) | $\frac{1}{2}(\langle \mathbf{h}_{e_s}, \mathbf{w}_r, \mathbf{t}_{e_o} \rangle + \langle \mathbf{h}_{e_o}, \mathbf{w}_{r^{-1}}, \mathbf{t}_{e_s} \rangle)$ | $\mathbf{w}_r \in \mathbb{R}^{d_e}$ | $\mathcal{O}(n_e d_e + n_r d_e)$ |
| HypER (Balažević et al., 2019) | $f(\text{vec}(\mathbf{e}_s * \text{vec}^{-1}(\mathbf{w}_r \mathbf{H}))\mathbf{W})\mathbf{e}_o$ | $\mathbf{w}_r \in \mathbb{R}^{d_r}$ | $\mathcal{O}(n_e d_e + n_r d_r)$ |
| TuckER (ours) | $\mathcal{W} \times_1 \mathbf{e}_s \times_2 \mathbf{w}_r \times_3 \mathbf{e}_o$ | $\mathbf{w}_r \in \mathbb{R}^{d_r}$ | $\mathcal{O}(n_e d_e + n_r d_r)$ |

Table 1: Scoring functions of state-of-the-art link prediction models, the dimensionality of their relation parameters, and significant terms of their space complexity. $d_e$ and $d_r$ are the dimensionalities of entity and relation embeddings, while $n_e$ and $n_r$ denote the number of entities and relations respectively. $\overline{\mathbf{e}}_o \in \mathbb{C}^{d_e}$ is the complex conjugate of $\mathbf{e}_o$, $\underline{\mathbf{e}}_s, \underline{\mathbf{w}}_r \in \mathbb{R}^{d_w \times d_h}$ denote a 2D reshaping of $\mathbf{e}_s$ and $\mathbf{w}_r$ respectively, $\mathbf{h}_{e_s}, \mathbf{t}_{e_s} \in \mathbb{R}^{d_e}$ are the head and tail entity embedding of entity $e_s$, and $\mathbf{w}_{r^{-1}} \in \mathbb{R}^{d_r}$ is the embedding of relation $r^{-1}$ (which is the inverse of relation $r$). $*$ is the convolution operator, $\langle \cdot \rangle$ denotes the dot product and $\times_n$ denotes the tensor product along the $n$-th mode, $f$ is a non-linear function, and $\mathcal{W} \in \mathbb{R}^{d_e \times d_e \times d_r}$ is the core tensor of a Tucker decomposition.

that assigns a score $s = \phi(e_s, r, e_o) \in \mathbb{R}$ which indicates whether a triple is true, with the ultimate goal of being able to correctly score all missing triples. The scoring function is either a specific form of tensor factorization in the case of linear models or a more complex (deep) neural network architecture for non-linear models. Typically, a positive score for a particular triple indicates a true fact predicted by the model, while a negative score indicates a false one. With most recent models, a non-linearity such as the logistic sigmoid function is typically applied to the score to give a corresponding probability prediction $p = \sigma(s) \in [0, 1]$ as to whether a certain fact is true.

### 3.2 Tucker Decomposition

Tucker decomposition, named after Ledyard R. Tucker (Tucker, 1964), decomposes a tensor into a set of matrices and a smaller core tensor. In a three-mode case, given the original tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$, Tucker decomposition outputs a tensor $\mathcal{Z} \in \mathbb{R}^{P \times Q \times R}$ and three matrices $\mathbf{A} \in \mathbb{R}^{I \times P}$, $\mathbf{B} \in \mathbb{R}^{J \times Q}$, $\mathbf{C} \in \mathbb{R}^{K \times R}$:

$$\mathcal{X} \approx \mathcal{Z} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}, \tag{1}$$

with $\times_n$ indicating the tensor product along the n-th mode. *Factor matrices* $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$, when orthogonal, can be thought of as the principal components in each mode. Elements of the *core tensor* $\mathcal{Z}$ show the level of interaction between the different components. Typically, $P$, $Q$, $R$ are smaller than $I$, $J$, $K$ respectively, so $\mathcal{Z}$ can be thought of as a compressed version of $\mathcal{X}$. Tucker decomposition is not unique, i.e. we can transform $\mathcal{Z}$ without affecting the fit if we apply the inverse transformation to $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ (Kolda and Bader, 2009).

## 4 Tucker Decomposition for Link Prediction

We propose a model that uses Tucker decomposition for link prediction on the binary tensor representation of a knowledge graph, with entity embedding matrix $\mathbf{E}$ that is equivalent for subject and object entities, i.e. $\mathbf{E} = \mathbf{A} = \mathbf{C} \in \mathbb{R}^{n_e \times d_e}$ and relation embedding matrix $\mathbf{R} = \mathbf{B} \in \mathbb{R}^{n_r \times d_r}$, where $n_e$ and $n_r$ represent the number of entities and relations and $d_e$ and $d_r$ the dimensionality of entity and relation embedding vectors.

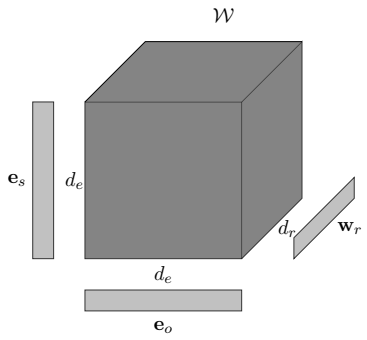


Figure 1: Visualization of the TuckER architecture.

We define the scoring function for TuckER as:

$$\phi(e_s, r, e_o) = \mathcal{W} \times_1 \mathbf{e}_s \times_2 \mathbf{w}_r \times_3 \mathbf{e}_o, \tag{2}$$

where $\mathbf{e}_s, \mathbf{e}_o \in \mathbb{R}^{d_e}$ are the rows of $\mathbf{E}$ representing the subject and object entity embedding vectors, $\mathbf{w}_r \in \mathbb{R}^{d_r}$ the rows of $\mathbf{R}$ representing the relation embedding vector and $\mathcal{W} \in \mathbb{R}^{d_e \times d_r \times d_e}$ is the core tensor. We apply logistic sigmoid to each score $\phi(e_s, r, e_o)$ to obtain the predicted probability $p$ of a triple being true. Visualization of the TuckER architecture can be seen in Figure 1. As proven in Section 5.1, TuckER is *fully expressive*. Further, its number of parameters increases *linearly* with

respect to entity and relation embedding dimensionality $d_e$ and $d_r$, as the number of entities and relations increases, since the number of parameters of $\mathcal{W}$ depends only on the entity and relation embedding dimensionality and not on the number of entities or relations. By having the core tensor $\mathcal{W}$, unlike simpler models such as DistMult, ComplEx and SimplE, TuckER does not encode all the learned knowledge into the embeddings; some is stored in the core tensor and shared between all entities and relations through *multi-task learning*. Rather than learning distinct relation-specific matrices, the core tensor of TuckER can be viewed as containing a shared pool of "prototype" relation matrices, which are linearly combined according to the parameters in each relation embedding.

### 4.1 Training

Since the logistic sigmoid is applied to the scoring function to approximate the true binary tensor, the implicit underlying tensor is comprised of $-\infty$ and $\infty$. Given this prevents an explicit analytical factorization, we use numerical methods to train TuckER. We use the standard data augmentation technique, first used by Dettmers et al. (2018) and formally described by Lacroix et al. (2018), of adding reciprocal relations for every triple in the dataset, i.e. we add $(e_o, r^{-1}, e_s)$ for every $(e_s, r, e_o)$. Following the training procedure introduced by Dettmers et al. (2018) to speed up training, we use *1-N scoring*, i.e. we simultaneously score entity-relation pairs $(e_s, r)$ and $(e_o, r^{-1})$ with all entities $e_o \in \mathcal{E}$ and $e_s \in \mathcal{E}$ respectively, in contrast to *1-1 scoring*, where individual triples $(e_s, r, e_o)$ and $(e_o, r^{-1}, e_s)$ are trained one at a time. The model is trained to minimize the Bernoulli negative log-likelihood loss function. A component of the loss for one entity-relation pair with all others entities is defined as:

$$L = -\frac{1}{n_e} \sum_{i=1}^{n_e} (\mathbf{y}^{(i)} \log(\mathbf{p}^{(i)}) + (1 - \mathbf{y}^{(i)}) \log(1 - \mathbf{p}^{(i)})), \tag{3}$$

where $\mathbf{p} \in \mathbb{R}^{n_e}$ is the vector of predicted probabilities and $\mathbf{y} \in \mathbb{R}^{n_e}$ is the binary label vector.

## 5 Theoretical Analysis

### 5.1 Full Expressiveness and Embedding Dimensionality

A tensor factorization model is fully expressive if for any ground truth over all entities and relations, there exist entity and relation embeddings that accurately separate true triples from the false. As shown in (Trouillon et al., 2017), ComplEx is fully expressive with the embedding dimensionality bound $d_e = d_r = n_e \cdot n_r$. Similarly to ComplEx, Kazemi and Poole (2018) show that SimplE is fully expressive with entity and relation embeddings of size $d_e = d_r = \textit{min}(n_e \cdot n_r, \gamma + 1)$, where $\gamma$ represents the number of true facts. They further prove other models are not fully expressive: DistMult, because it cannot model asymmetric relations; and transitive models such as TransE (Bordes et al., 2013) and its variants FTransE (Feng et al., 2016) and STransE (Nguyen et al., 2016), because of certain contradictions that they impose between different relation types. By Theorem 1, we establish the bound on entity and relation embedding dimensionality (i.e. decomposition rank) that guarantees full expressiveness of TuckER.

**Theorem 1.** *Given any ground truth over a set of entities $\mathcal{E}$ and relations $\mathcal{R}$, there exists a TuckER model with entity embeddings of dimensionality $d_e = n_e$ and relation embeddings of dimensionality $d_r = n_r$, where $n_e = |\mathcal{E}|$ is the number of entities and $n_r = |\mathcal{R}|$ the number of relations, that accurately represents that ground truth.*

*Proof.* Let $\mathbf{e}_s$ and $\mathbf{e}_o$ be the $n_e$-dimensional one-hot binary vector representations of subject and object entities $e_s$ and $e_o$ respectively and $\mathbf{w}_r$ the $n_r$-dimensional one-hot binary vector representation of relation $r$. For each subject entity $e_s^{(i)}$, relation $r^{(j)}$ and object entity $e_o^{(k)}$, we let the $i$-th, $j$-th and $k$-th element respectively of the corresponding vectors $\mathbf{e}_s$, $\mathbf{w}_r$ and $\mathbf{e}_o$ be 1 and all other elements 0. Further, we set the $ijk$ element of the tensor $\mathcal{W} \in \mathbb{R}^{n_e \times n_r \times n_e}$ to 1 if the fact $(e_s, r, e_o)$ holds and -1 otherwise. Thus the product of the entity embeddings and the relation embedding with the core tensor, after applying the logistic sigmoid, accurately represents the original tensor. □

The purpose of Theorem 1 is to prove that TuckER is capable of potentially capturing all information (and noise) in the data. In practice however, we expect the embedding dimensionalities needed for full reconstruction of the underlying binary tensor to be much smaller than the bound stated above, since the assignment of values to the tensor is not random but follows a certain structure, otherwise nothing unknown could be predicted. Even more so, low decomposition rank is actually a desired property of any bilin-

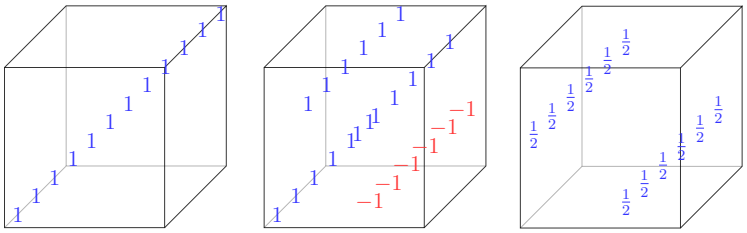

(a) DistMult (b) ComplEx (c) SimplE

Figure 2: Constraints imposed on the values of core tensor $\mathcal{Z} \in \mathbb{R}^{d_e \times d_e \times d_e}$ for DistMult and $\mathcal{Z} \in \mathbb{R}^{2d_e \times 2d_e \times 2d_e}$ for ComplEx and SimplE. Elements that are set to 0 are represented in white.

ear link prediction model, forcing it to learn that structure and generalize to new data, rather than simply memorizing the input. In general, we expect TuckER to perform better than ComplEx and SimplE with embeddings of lower dimensionality due to parameter sharing in the core tensor (shown empirically in Section 6.4), which could be of importance for efficiency in downstream tasks.

### 5.2 Relation to Previous Linear Models

Several previous tensor factorization models can be viewed as a special case of TuckER:

**RESCAL (Nickel et al., 2011)** Following the notation introduced in Section 3.2, the RESCAL scoring function (see Table 1) has the form:

$$\mathcal{X} \approx \mathcal{Z} \times_1 \mathbf{A} \times_3 \mathbf{C}. \tag{4}$$

This corresponds to Equation 1 with $I = K = n_e$, $P = R = d_e$, $Q = J = n_r$ and $\mathbf{B} = \mathbf{I}_J$ the $J \times J$ identity matrix. This is also known as Tucker2 decomposition (Kolda and Bader, 2009). As is the case with TuckER, the entity embedding matrix of RESCAL is shared between subject and object entities, i.e. $\mathbf{E} = \mathbf{A} = \mathbf{C} \in \mathbb{R}^{n_e \times d_e}$ and the relation matrices $\mathbf{W}_r \in \mathbb{R}^{d_e \times d_e}$ are the $d_e \times d_e$ slices of the core tensor $\mathcal{Z}$. As mentioned in Section 2, the drawback of RESCAL compared to TuckER is that its number of parameters grows *quadratically* in the entity embedding dimension $d_e$ as the number of relations increases.

**DistMult (Yang et al., 2015)** The scoring function of DistMult (see Table 1) can be viewed as equivalent to that of TuckER (see Equation 1) with a core tensor $\mathcal{Z} \in \mathbb{R}^{P \times Q \times R}$, $P = Q = R = d_e$, which is *superdiagonal* with 1s on the superdiagonal, i.e. all elements $z_{pqr}$ with $p = q = r$ are 1 and all the other elements are 0 (as shown in Figure 2a). Rows of $\mathbf{E} = \mathbf{A} = \mathbf{C} \in \mathbb{R}^{n_e \times d_e}$ contain subject and object entity embedding vectors $\mathbf{e}_s, \mathbf{e}_o \in \mathbb{R}^{d_e}$ and rows of $\mathbf{R} = \mathbf{B} \in \mathbb{R}^{n_r \times d_e}$ contain relation embedding vectors $\mathbf{w}_r \in \mathbb{R}^{d_e}$. It is interesting to note that the TuckER interpretation of the DistMult scoring function, given that matrices $\mathbf{A}$ and $\mathbf{C}$ are identical, can alternatively be interpreted as a special case of CP decomposition (Hitchcock, 1927), since Tucker decomposition with a superdiagonal core tensor is equivalent to CP decomposition. Due to enforced symmetry in subject and object entity mode, DistMult cannot learn to represent asymmetric relations.

**ComplEx (Trouillon et al., 2016)** *Bilinear models* represent subject and object entity embeddings as vectors $\mathbf{e}_s, \mathbf{e}_o \in \mathbb{R}^{d_e}$, relation as a matrix $\mathbf{W}_r \in \mathbb{R}^{d_e \times d_e}$ and the scoring function as a bilinear product $\phi(e_s, r, e_o) = \mathbf{e}_s \mathbf{W}_r \mathbf{e}_o$. It is trivial to show that both RESCAL and DistMult belong to the family of bilinear models. As explained by Kazemi and Poole (2018), ComplEx can be considered a bilinear model with the real and imaginary part of an embedding for each entity concatenated in a single vector, $[\text{Re}(\mathbf{e}_s); \text{Im}(\mathbf{e}_s)] \in \mathbb{R}^{2d_e}$ for subject, $[\text{Re}(\mathbf{e}_o); \text{Im}(\mathbf{e}_o)] \in \mathbb{R}^{2d_e}$ for object, and a relation matrix $\mathbf{W}_r \in \mathbb{R}^{2d_e \times 2d_e}$, constrained so that its leading diagonal contains duplicated elements of $\text{Re}(\mathbf{w}_r)$, its $d_e$-diagonal elements of $\text{Im}(\mathbf{w}_r)$ and its -$d_e$-diagonal elements of -$\text{Im}(\mathbf{w}_r)$, with all other elements set to 0, where $d_e$ and -$d_e$ represent offsets from the leading diagonal.

Similarly to DistMult, we can regard the scoring function of ComplEx (see Table 1) as equivalent to the scoring function of TuckER (see Equation 1), with core tensor $\mathcal{Z} \in \mathbb{R}^{P \times Q \times R}$, $P = Q = R = 2d_e$, where $3d_e$ elements on different tensor diagonals are set to 1, $d_e$ elements on one tensor diagonal are set to -1 and all other elements are set to 0 (see Figure 2b). This shows that the scoring function of ComplEx, which computes a bilinear product with *complex* entity and relation embeddings and disregards the imaginary part of the obtained result, is equivalent to a hard regularization of the core tensor of TuckER in the *real* domain.

**SimplE (Kazemi and Poole, 2018)** The authors show that SimplE belongs to the family of bilinear models by concatenating embeddings for head and tail entities for both subject and object into vectors $[\mathbf{h}_{e_s}; \mathbf{t}_{e_s}] \in \mathbb{R}^{2d_e}$ and $[\mathbf{h}_{e_o}; \mathbf{t}_{e_o}] \in \mathbb{R}^{2d_e}$ and constraining the relation matrix $\mathbf{W}_r \in \mathbb{R}^{2d_e \times 2d_e}$ so that it contains the relation embedding vector $\frac{1}{2}\mathbf{w}_r$ on its $d_e$-diagonal and the inverse relation embedding vector $\frac{1}{2}\mathbf{w}_{r^{-1}}$ on its -$d_e$-diagonal and 0s elsewhere. The SimplE scoring function (see Table 1) is therefore equivalent to that of TuckER (see Equation 1), with core tensor $\mathcal{Z} \in \mathbb{R}^{P \times Q \times R}$, $P = Q = R = 2d_e$, where $2d_e$ elements on two tensor diagonals are set to $\frac{1}{2}$ and all other elements are set to 0 (see Figure 2c).

### 5.3 Representing Asymmetric Relations

Each relation in a knowledge graph can be characterized by a certain set of properties, such as symmetry, reflexivity, transitivity. So far, there have been two possible ways in which linear link prediction models introduce *asymmetry* into factorization of the binary tensor of triples:

- distinct (although possibly related) embeddings for subject and object entities and a diagonal matrix (or equivalently a vector) for each relation, as is the case with models such as ComplEx and SimplE; or
- equivalent subject and object entity embeddings and each relation represented by a full rank matrix, which is the case with RESCAL.

The latter approach appears more intuitive, since asymmetry is a property of the relation, rather than the entities. However, the drawback of the latter approach is quadratic growth of parameter number with the number of relations, which often leads to overfitting, especially for relations with a small number of training triples. TuckER overcomes this by representing relations as vectors $\mathbf{w}_r$, which makes the parameter number grow linearly with the number of relations, while still keeping the desirable property of allowing relations to be asymmetric by having an asymmetric *relation-agnostic* core tensor $\mathcal{W}$, rather than encoding the relation-specific information in the entity embeddings. Multiplying $\mathcal{W} \in \mathbb{R}^{d_e \times d_r \times d_e}$ with $\mathbf{w}_r \in \mathbb{R}^{d_r}$ along the second mode, we obtain a full rank relation-specific matrix $\mathbf{W}_r \in \mathbb{R}^{d_e \times d_e}$, which can perform all possible linear transformations on the entity embeddings, i.e. rotation, reflection or stretch, and is thus also capable of modeling asymmetry. Regardless of what kind of transformation is needed for modeling a particular relation, TuckER can learn it from the data. To demonstrate this, we show sample heatmaps of learned relation matrices $\mathbf{W}_r$ for a WordNet symmetric relation "derivationally_related_form" and an asymmetric relation "hypernym" in Figure 3, where one can see that TuckER learns to model the symmetric relation with the relation matrix that is approximately symmetric about the main diagonal, whereas the matrix belonging to the asymmetric relation exhibits no obvious structure.

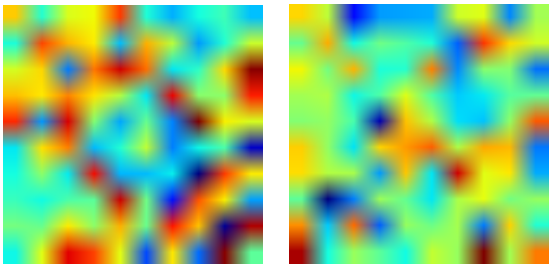

(a) $\mathbf{W}_{\text{derivationally\_related\_form}}$ (b) $\mathbf{W}_{\text{hypernym}}$

Figure 3: Learned relation matrices for a symmetric (derivationally_related_form) and an asymmetric (hypernym) WN18RR relation. $\mathbf{W}_{\text{derivationally\_related\_form}}$ is approximately symmetric about the leading diagonal.

## 6 Experiments and Results

### 6.1 Datasets

We evaluate TuckER using four standard link prediction datasets (see Table 2):

**FB15k** (Bordes et al., 2013) is a subset of Freebase, a large database of real world facts.

**FB15k-237** (Toutanova et al., 2015) was created from FB15k by removing the inverse of many relations that are present in the training set from validation and test sets, making it more difficult for simple models to do well.

**WN18** (Bordes et al., 2013) is a subset of WordNet, a hierarchical database containing lexical relations between words.

**WN18RR** (Dettmers et al., 2018) is a subset of WN18, created by removing the inverse relations from validation and test sets.

### 6.2 Implementation and Experiments

We implement TuckER in PyTorch (Paszke et al., 2017) and make our code available on GitHub.[1]

We choose all hyper-parameters by random search based on validation set performance. For

[1] https://github.com/ibalazevic/TuckER

FB15k and FB15k-237, we set entity and relation embedding dimensionality to $d_e = d_r = 200$. For WN18 and WN18RR, which both contain a significantly smaller number of relations relative to the number of entities as well as a small number of relations compared to FB15k and FB15k-237, we set $d_e = 200$ and $d_r = 30$. We use batch normalization (Ioffe and Szegedy, 2015) and dropout (Srivastava et al., 2014) to speed up training. We find that lower dropout values $(0.1, 0.2)$ are required for datasets with a higher number of training triples per relation and thus less risk of overfitting (WN18 and WN18RR), whereas higher dropout values $(0.3, 0.4, 0.5)$ are required for FB15k and FB15k-237. We choose the learning rate from $\{0.01, 0.005, 0.003, 0.001, 0.0005\}$ and learning rate decay from $\{1, 0.995, 0.99\}$. We find the following combinations of learning rate and learning rate decay to give the best results: $(0.003, 0.99)$ for FB15k, $(0.0005, 1.0)$ for FB15k-237, $(0.005, 0.995)$ for WN18 and $(0.01, 1.0)$ for WN18RR (see Table 5 in the Appendix A for a complete list of hyper-parameter values on each dataset). We train the model using Adam (Kingma and Ba, 2015) with the batch size 128.

At evaluation time, for each test triple we generate $n_e$ candidate triples by combining the test entity-relation pair with all possible entities $\mathcal{E}$, ranking the scores obtained. We use the *filtered* setting (Bordes et al., 2013), i.e. all known true triples are removed from the candidate set except for the current test triple. We use evaluation metrics standard across the link prediction literature: mean reciprocal rank (MRR) and hits@$k$, $k \in \{1, 3, 10\}$. Mean reciprocal rank is the average of the inverse of the mean rank assigned to the true triple over all candidate triples. Hits@$k$ measures the percentage of times a true triple is ranked within the top $k$ candidate triples.

| Dataset | # Entities ($n_e$) | # Relations ($n_r$) |
|---|---|---|
| FB15k | 14,951 | 1,345 |
| FB15k-237 | 14,541 | 237 |
| WN18 | 40,943 | 18 |
| WN18RR | 40,943 | 11 |

Table 2: Dataset statistics.

### 6.3 Link Prediction Results

Link prediction results on all datasets are shown in Tables 3 and 4. Overall, TuckER outperforms previous state-of-the-art models on all metrics across all datasets (apart from hits@10 on WN18 where a non-linear model, R-GCN, does better). Results achieved by TuckER are not only better than those of other linear models, such as DistMult, ComplEx and SimplE, but also better than the results of many more complex deep neural network and reinforcement learning architectures, e.g. R-GCN, MINERVA, ConvE and HypER, demonstrating the expressive power of linear models and supporting our claim that simple linear models should serve as a baseline before moving onto more elaborate models.

Even with fewer parameters than ComplEx and SimplE at $d_e = 200$ and $d_r = 30$ on WN18RR ($\sim$9.4 vs $\sim$16.4 million), TuckER consistently obtains better results than any of those models. We believe this is because TuckER exploits knowledge sharing between relations through the core tensor, i.e. multi-task learning. This is supported by the fact that the margin by which TuckER outperforms other linear models is notably increased on datasets with a large number of relations. For example, improvement on FB15k is $+14\%$ over ComplEx and $+8\%$ over SimplE on the toughest hits@1 metric. To our knowledge, ComplEx-N3 (Lacroix et al., 2018) is the only other linear link prediction model that benefits from multi-task learning. There, rank regularization of the embedding matrices is used to encourage a low-rank factorization, thus forcing parameter sharing between relations. We do not include their published results in Tables 3 and 4, since they use the highly non-standard $d_e = d_r = 2000$ and thus a far larger parameter number (18x more parameters than TuckER on WN18RR; 5.5x on FB15k-237), making their results incomparable to those typically reported, including our own. However, running their model with equivalent parameter number to TuckER shows comparable performance, supporting our belief that the two models both attain the benefits of multi-task learning, although by different means.

### 6.4 Influence of Parameter Sharing

The ability of knowledge sharing through the core tensor suggests that TuckER should need a lower number of parameters for obtaining good results than ComplEx or SimplE. To test this, we re-implement ComplEx and SimplE with reciprocal relations, 1-N scoring, batch normalization and dropout for fair comparison, perform random search to choose best hyper-parameters

| | | WN18RR | | | | FB15k-237 | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Linear | MRR | Hits@10 | Hits@3 | Hits@1 | MRR | Hits@10 | Hits@3 | Hits@1 |
| DistMult (Yang et al., 2015) | yes | .430 | .490 | .440 | .390 | .241 | .419 | .263 | .155 |
| ComplEx (Trouillon et al., 2016) | yes | .440 | .510 | .460 | .410 | .247 | .428 | .275 | .158 |
| Neural LP (Yang et al., 2017) | no | – | – | – | – | .250 | .408 | – | – |
| R-GCN (Schlichtkrull et al., 2018) | no | – | – | – | – | .248 | .417 | .264 | .151 |
| MINERVA (Das et al., 2018) | no | – | – | – | – | – | .456 | – | – |
| ConvE (Dettmers et al., 2018) | no | .430 | .520 | .440 | .400 | .325 | .501 | .356 | .237 |
| HypER (Balažević et al., 2019) | no | .465 | .522 | .477 | .436 | .341 | .520 | .376 | .252 |
| M-Walk (Shen et al., 2018) | no | .437 | – | .445 | .414 | – | – | – | – |
| RotatE (Sun et al., 2019) | no | – | – | – | – | .297 | .480 | .328 | .205 |
| TuckER (ours) | yes | **.470** | **.526** | **.482** | **.443** | **.358** | **.544** | **.394** | **.266** |

Table 3: Link prediction results on WN18RR and FB15k-237. The RotatE (Sun et al., 2019) results are reported without their self-adversarial negative sampling (see Appendix H in the original paper) for fair comparison.

| | | WN18 | | | | FB15k | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Linear | MRR | Hits@10 | Hits@3 | Hits@1 | MRR | Hits@10 | Hits@3 | Hits@1 |
| TransE (Bordes et al., 2013) | no | – | .892 | – | – | – | .471 | – | – |
| DistMult (Yang et al., 2015) | yes | .822 | .936 | .914 | .728 | .654 | .824 | .733 | .546 |
| ComplEx (Trouillon et al., 2016) | yes | .941 | .947 | .936 | .936 | .692 | .840 | .759 | .599 |
| ANALOGY (Liu et al., 2017) | yes | .942 | .947 | .944 | .939 | .725 | .854 | .785 | .646 |
| Neural LP (Yang et al., 2017) | no | .940 | .945 | – | – | .760 | .837 | – | – |
| R-GCN (Schlichtkrull et al., 2018) | no | .819 | **.964** | .929 | .697 | .696 | .842 | .760 | .601 |
| TorusE (Ebisu and Ichise, 2018) | no | .947 | .954 | .950 | .943 | .733 | .832 | .771 | .674 |
| ConvE (Dettmers et al., 2018) | no | .943 | .956 | .946 | .935 | .657 | .831 | .723 | .558 |
| HypER (Balažević et al., 2019) | no | .951 | 958 | **.955** | .947 | .790 | .885 | .829 | .734 |
| SimplE (Kazemi and Poole, 2018) | yes | .942 | .947 | .944 | .939 | .727 | .838 | .773 | .660 |
| TuckER (ours) | yes | **.953** | .958 | **.955** | **.949** | **.795** | **.892** | **.833** | **.741** |

Table 4: Link prediction results on WN18 and FB15k.

(see Table 6 in the Appendix A for exact hyperparameter values used) and train all three models on FB15k-237 with embedding sizes $d_e = d_r \in \{20, 50, 100, 200\}$. Figure 4 shows the obtained MRR on the test set for each model. It is important to note that at embedding dimensionalities 20, 50 and 100, TuckER has fewer parameters than ComplEx and SimplE (e.g. ComplEx and SimplE have $\sim$3 million and TuckER has $\sim$2.5 million parameters for embedding dimensionality 100).

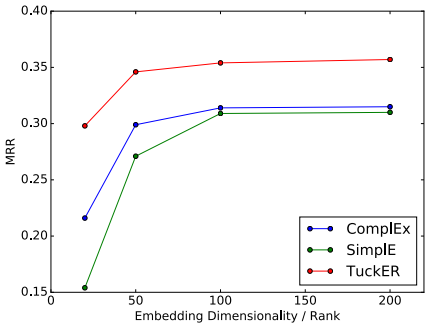


Figure 4: MRR for ComplEx, SimplE and TuckER for different embeddings sizes on FB15k-237.

We can see that the difference between the MRRs of ComplEx, SimplE and TuckER is approximately constant for embedding sizes 100 and 200. However, for lower embedding sizes, the difference between MRRs increases by $0.7\%$ for embedding size 50 and by $4.2\%$ for embedding size 20 for ComplEx and by $3\%$ for embedding size 50 and by $9.9\%$ for embedding size 20 for SimplE. At embedding size 20 ($\sim$300k parameters), the performance of TuckER is almost as good as the performance of ComplEx and SimplE at embedding size 200 ($\sim$6 million parameters), which supports our initial assumption.

## 7 Conclusion

In this work, we introduce TuckER, a relatively straightforward linear model for link prediction on knowledge graphs, based on the Tucker decomposition of a binary tensor of known facts. TuckER achieves state-of-the-art results on standard link prediction datasets, in part due to its ability to perform multi-task learning across relations. Whilst being fully expressive, TuckER's number of parameters grows linearly with respect to the number of entities or relations in the knowledge graph. We further show that previous linear state-of-the-art models, RESCAL, DistMult, ComplEx and SimplE, can be interpreted as special cases of our model. Future work might include exploring how to incorporate background knowledge on individual relation properties into the existing model.

## Acknowledgements

Ivana Balažević and Carl Allen were supported by the Centre for Doctoral Training in Data Science, funded by EPSRC (grant EP/L016427/1) and the University of Edinburgh.

## References

Ivana Balažević, Carl Allen, and Timothy M Hospedales. 2019. Hypernetwork Knowledge Graph Embeddings. In *International Conference on Artificial Neural Networks*.

Hedi Ben-Younes, Rémi Cadene, Matthieu Cord, and Nicolas Thome. 2017. MUTAN: Multimodal Tucker Fusion for Visual Question Answering. In *International Conference on Computer Vision*.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *Advances in Neural Information Processing Systems*.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2018. Go for a Walk and Arrive at the Answer: Reasoning over Paths in Knowledge Bases Using Reinforcement Learning. In *International Conference on Learning Representations*.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2D Knowledge Graph Embeddings. In *Association for the Advancement of Artificial Intelligence*.

Takuma Ebisu and Ryutaro Ichise. 2018. TorusE: Knowledge Graph Embedding on a Lie Group. In *Association for the Advancement of Artificial Intelligence*.

Jun Feng, Minlie Huang, Mingdong Wang, Mantong Zhou, Yu Hao, and Xiaoyan Zhu. 2016. Knowledge Graph Embedding by Flexible Translation. In *Principles of Knowledge Representation and Reasoning*.

Frank L Hitchcock. 1927. The Expression of a Tensor or a Polyadic as a Sum of Products. *Journal of Mathematics and Physics*, 6(1-4):164–189.

Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International Conference on Machine Learning*.

Seyed Mehran Kazemi and David Poole. 2018. SimplE Embedding for Link Prediction in Knowledge Graphs. In *Advances in Neural Information Processing Systems*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*.

Tamara G Kolda and Brett W Bader. 2009. Tensor Decompositions and Applications. *SIAM review*, 51(3):455–500.

Pieter M Kroonenberg and Jan De Leeuw. 1980. Principal Component Analysis of Three-Mode Data by Means of Alternating Least Squares Algorithms. *Psychometrika*, 45(1):69–97.

Timothée Lacroix, Nicolas Usunier, and Guillaume Obozinski. 2018. Canonical Tensor Decomposition for Knowledge Base Completion. In *International Conference on Machine Learning*.

Hanxiao Liu, Yuexin Wu, and Yiming Yang. 2017. Analogical Inference for Multi-relational Embeddings. In *International Conference on Machine Learning*.

Dat Quoc Nguyen, Kairit Sirts, Lizhen Qu, and Mark Johnson. 2016. STransE: a Novel Embedding Model of Entities and Relationships in Knowledge Bases. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A Three-Way Model for Collective Learning on Multi-Relational Data. In *International Conference on Machine Learning*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic Differentiation in PyTorch. In *NIPS-W*.

Aaron Schein, Mingyuan Zhou, David Blei, and Hanna Wallach. 2016. Bayesian Poisson Tucker Decomposition for Learning the Structure of International Relations. In *International Conference on Machine Learning*.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling Relational Data with Graph Convolutional Networks. In *European Semantic Web Conference*.

Yelong Shen, Jianshu Chen, Po-Sen Huang, Yuqing Guo, and Jianfeng Gao. 2018. M-Walk: Learning to Walk over Graphs using Monte Carlo Tree Search. In *Advances in Neural Information Processing Systems*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In *International Conference on Learning Representations*.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing Text for Joint Embedding of Text and Knowledge Bases. In *Empirical Methods in Natural Language Processing*.

Théo Trouillon, Christopher R Dance, Éric Gaussier, Johannes Welbl, Sebastian Riedel, and Guillaume Bouchard. 2017. Knowledge Graph Completion via Complex Tensor Factorization. *Journal of Machine Learning Research*, 18(1):4735–4772.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex Embeddings for Simple Link Prediction. In *International Conference on Machine Learning*.

Ledyard R Tucker. 1964. The Extension of Factor Analysis to Three-Dimensional Matrices. *Contributions to Mathematical Psychology*, 110119.

Ledyard R Tucker. 1966. Some Mathematical Notes on Three-Mode Factor Analysis. *Psychometrika*, 31(3):279–311.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *International Conference on Learning Representations*.

Fan Yang, Zhilin Yang, and William W Cohen. 2017. Differentiable Learning of Logical Rules for Knowledge Base Reasoning. In *Advances in Neural Information Processing Systems*.

Yongxin Yang and Timothy Hospedales. 2017. Deep Multi-task Representation Learning: A Tensor Factorisation Approach. In *International Conference on Learning Representations*.

# Human-grounded Evaluations of Explanation Methods for Text Classification

**Piyawat Lertvittayakumjorn** and **Francesca Toni**
Department of Computing
Imperial College London, UK
{pl1515, ft}@imperial.ac.uk

## Abstract

Due to the black-box nature of deep learning models, methods for explaining the models' results are crucial to gain trust from humans and support collaboration between AIs and humans. In this paper, we consider several model-agnostic and model-specific explanation methods for CNNs for text classification and conduct three human-grounded evaluations, focusing on different purposes of explanations: *(1)* revealing model behavior, *(2)* justifying model predictions, and *(3)* helping humans investigate uncertain predictions. The results highlight dissimilar qualities of the various explanation methods we consider and show the degree to which these methods could serve for each purpose.

## 1 Introduction

Explainable Artificial Intelligence (XAI) is aimed at providing explanations for decisions made by AI systems. The explanations are useful for supporting collaboration between AIs and humans in many cases (Samek et al., 2018). Firstly, if an AI outperforms humans in a certain task (e.g., AlphaGo (Silver et al., 2016)), humans can learn and distill knowledge from the given explanations. Secondly, if an AI's performance is close to human intelligence, the explanations can increase humans' confidence and trust in the AI (Symeonidis et al., 2009). Lastly, if an AI is duller than humans, the explanations help humans verify the decisions made by the AI and also improve the AI (Biran and McKeown, 2017).

One of the challenges in XAI is to explain prediction results given by deep learning models, which sacrifice transparency for high prediction performance. This type of explanations is called *local explanations* as they explain individual predictions (in contrast to *global explanations* which explain the trained model independently of any specific prediction). There have been several methods proposed to produce local explanations. Some of them are model-agnostic, applicable to any machine learning model (Ribeiro et al., 2016; Lundberg and Lee, 2017). Others are applicable to a class of models such as neural networks (Bach et al., 2015a; Dimopoulos et al., 1995) or to a specific model such as Convolutional Neural Networks (CNNs) (Zhou et al., 2016).

With so many explanation methods available, the next challenge is how to evaluate them so as to choose the right methods for different settings. In this paper, we focus on human-grounded evaluations of local explanation methods for text classification. Particularly, we propose three evaluation tasks which target different purposes of explanations for text classification – *(1)* revealing model behavior to human users, *(2)* justifying the predictions, and *(3)* helping humans investigate uncertain predictions. We then use the proposed tasks to evaluate nine explanation methods working on a standard CNN for text classification. These explanation methods are different in several aspects. For example, regarding granularity, four explanation methods select words from the input text as explanations, whereas the other five select n-grams as explanations. In terms of generality, one of the explanation methods is model-agnostic, two are random baselines, another two (newly proposed in this paper) are specific to 1D CNNs for text classification, and the rest are applicable to neural networks in general. Overall, the contributions of our work can be summarized as follows.

- We propose three human-grounded evaluation tasks to assess the quality of explanation methods with respect to different purposes of usage for text classification. (Section 3)
- To increase diversity in the experiments, we develop two new explanation methods for



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5195–5205,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

CNNs for text classification. One is based on gradient-based analysis (**Grad-CAM-Text**). The other is based on model extraction using **decision trees**. (Section 4.3.1-4.3.2)

- We evaluate both new methods as well as random baselines and well-known existing methods using the three evaluation tasks proposed. The results highlight dissimilar qualities of the explanation methods and show the degree to which these methods could serve for each purpose. (Section 5)

### 1.1 Terminology Used

We use the following terms throughout the paper. (1) **Model**: a deep learning classifier we want to explain, e.g., a CNN. (2) **Explanation**: an ordered list of text fragments (words or n-grams) in the input text which are most relevant to a prediction. Explanations for and against the predicted class are called **evidence** and **counter-evidence**, respectively. (3) **(Local) explanation method**: a method producing an explanation for a model and an input text. (4) **Evaluation method**: a process to quantitatively assign to explanations scores which reflect the quality of the explanation method.

## 2 Background and Related Work

This section discusses recent advances of explanation methods and evaluation for text classification as well as background knowledge about 1D CNNs – the model used in the experiments.

### 2.1 Local Explanation Methods

Generally, there are several ways to explain a result given by a deep learning model, such as explaining by examples (Kim et al., 2014) and generating textual explanations (Liu et al., 2019). For text classification in particular, most of the existing explanation methods identify parts of the input text which contribute most towards the predicted class (so called *attribution methods* or *relevance methods*) by exploiting various techniques such as input perturbation (Li et al., 2016), gradient analysis (Dimopoulos et al., 1995), and relevance propagation (Arras et al., 2017b). Besides, there are other explanation methods designed for specific deep learning architectures such as attention mechanism (Ghaeini et al., 2018) and extractive rationale generation (Lei et al., 2016).

We select some well-known explanation methods (which are applicable to CNNs for text classification) and evaluate them together with two new explanation methods proposed in this paper.

### 2.2 Evaluation Methods

Focusing on text classification, early works evaluated explanation methods by *word deletion* – gradually deleting words from the input text in the order of their relevance and checking how the prediction confidence drops (Arras et al., 2016; Nguyen, 2018). Arras et al. (2017a) and Xiong et al. (2018) used relevance scores generated by explanation methods to construct document vectors by weighted-averaging word vectors and checked how well traditional machine learning techniques manage these document vectors. Poerner et al. (2018) proposed two evaluation paradigms – hybrid documents and morphosyntactic agreements. Both check whether an explanation method correctly points to the (known) root cause of the prediction. Note that all of the aforementioned evaluation methods are conducted with no humans involved.

For human-grounded evaluation, Mohseni and Ragan (2018) proposed a benchmark which contains a list of relevant words for the actual class of each input text, identified by human experts. However, comparing human explanations with the explanations given by the tested method may be inappropriate since the mismatches could be due to not only the poor explanation method but also the inaccuracy of the model or the model reasoning differently from humans. Nguyen (2018) asked humans to guess the output of a text classifier, given an input text with the highest relevant words highlighted by the tested explanation method. Informative (and discriminative) explanations will lead to humans' correct guesses. Ribeiro et al. (2016) asked humans to choose a model which can generalize better by considering their local explanations. Also, they let humans remove irrelevant words, existing in the explanations, from the corpus to improve the prediction performance. Compared to previous work, our work is more comprehensive in terms of the various human-grounded evaluation tasks proposed and the number and dimensions of explanation methods being evaluated.

### 2.3 CNNs for Text Classification

CNNs have been found to achieve promising results in many text classification tasks (Johnson and Zhang, 2015; Gambäck and Sikdar, 2017; Zhang et al., 2019). Figure 1 shows a standard 1D CNN

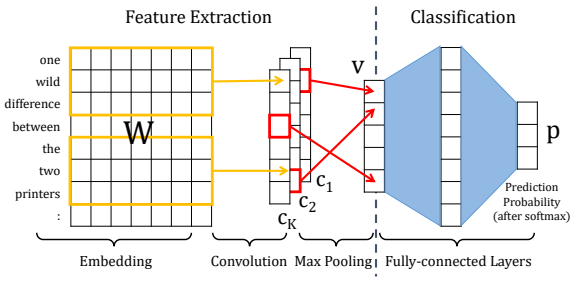


Figure 1: CNN for text classification.

for text classification which consists of four main steps: *(i)* embedding an input text into an embedded matrix $\mathbf{W}$; *(ii)* applying $K$ fixed-size convolution filters to $\mathbf{W}$ to find n-grams that possibly discriminate one class from the others; *(iii)* pooling only the maximum value found by each filter, corresponding to the most relevant n-gram in text, to construct a filter-based feature vector, $\mathbf{v}$, of the input; and *(iv)* using fully-connected layers ($FC$) to predict the results, and applying a softmax function to the outputs to obtain predicted probability of the classes ($\mathbf{p}$), i.e., $\mathbf{p} = softmax(FC(\mathbf{v}))$. While the original version of this model uses only one linear layer as $FC$ (Kim, 2014), more hidden layers can be added to increase the model capacity for prediction. Also, more than one filter size can be used to detect n-grams with short- and long-span relations (Conneau et al., 2017).

## 3 Human-grounded Evaluation Methods

We propose three human tasks to evaluate explanation methods for text classification as summarized in Table 1. Figure 2 gives an example question for each task, discussed next.

### 3.1 Revealing the Model Behavior

Task 1 evaluates whether explanations can expose irrational behavior of a poor model. This property of explanation methods is very useful when we do not have a labelled dataset to evaluate the model quantitatively. To set up the task, firstly, we train two models to make them have different performance on classifying testing examples (i.e., different capability to generalize to unseen data). Then we use these models to classify an input text and apply the explanation method of interest to explain the predictions – highlighting top-$m$ evidence text fragments on the text for each model. Next, we ask humans, based on the highlighted texts from the two models, which model is more reasonable? If the performance of the two models is clearly different, good explanation methods should enable humans to notice the poor model, which is more likely to decide based on non-discriminative words, even though both models predict the same class for an input text.

Additionally, there are some important points to note for this task. First, the chosen input texts must be classified into the same class by both models so the humans make decisions based only on the different explanations. However, it is worth to consider both the cases where both models correctly classify and where they misclassify. Second, we provide the choices for the two models along with confidence levels for the humans to select. If they select the right model with high confidence, the explanation method will get a higher positive score. In contrast, a confident but incorrect answer results in a large negative score. Also, the humans have the option to state no preference, for which the explanation method will get a zero score (See the last row of Table 1).

### 3.2 Justifying the Predictions

Explanations are sometimes used by humans as the reasons for the predicted class. This task tests whether the evidence texts are truly related to the predicted class and can distinguish it from the other classes, so called *class-discriminative* (Selvaraju et al., 2017). To set up the task, we use a well-trained model and select an input example classified by this model with high confidence ($\max_c \mathbf{p}_c > \tau_h$ where $\tau_h$ is a threshold parameter), so as to reduce the cases of unclear explanations due to low model accuracy or text ambiguity. (Note that we will look at low-confidence predictions later in Task 3.) Then we show only the top-$m$ evidence text fragments generated by the method of interest to humans and ask them to guess the class of the document containing the evidence. The explanation method which makes the humans surely guess the class predicted by the model will get a high positive score. As in the previous task, this task considers both the correct and incorrect predictions with high confidence to see how well the explanations justify each of the cases. For incorrect predictions, an explanation method gets a positive score when a human guesses the same incorrect class after seeing the explanation. In real applications, convincing explanations for incorrect classes can help humans understand the

| | **Task 1** (Section 3.1) | **Task 2** (Section 3.2) | **Task 3** (Section 3.3) |
|---|---|---|---|
| Assumption | Good explanations can reveal model behavior | Good explanations justify the predictions | Good explanations help humans investigate uncertain predictions |
| Model(s) | Two classifiers with different performance on a test dataset | One well-trained classifier | One well-trained classifier |
| Input text | A test example for which both classifiers predict the same class | A test example which the classifier predicts with high confidence ($\max_c \mathbf{p}_c > \tau_h$) | A test example which the classifier predicts with low confidence ($\max_c \mathbf{p}_c < \tau_l$) |
| Information displayed | 1. The input text<br>2. The predicted class<br>3. (Highlighted) top-$m$ evidence texts of each model | 1. Top-$m$ evidence texts | 1. The predicted class<br>2. The predicted probability $\mathbf{p}$<br>3. Top-$m$ evidence and top-$m$ counter-evidence texts |
| Human task | Select the more reasonable model and state if they are confident or not | Select the most likely class of the document which contains the evidence texts and state if they are confident or not | Select the most likely class of the input text and state if they are confident or not |
| Scores to the explanation method | (-)1.0: (In)correct, confident<br>(-)0.5: (In)correct, unconfident<br>0.0: No preference | (-)1.0: (In)correct, confident<br>(-)0.5: (In)correct, unconfident<br>0.0: No preference | (-)1.0: (In)correct, confident<br>(-)0.5: (In)correct, unconfident |

Table 1: A summary of the proposed human-grounded evaluation tasks.

model's weakness and create additional fixing examples to retrain and improve the model.

### 3.3 Investigating Uncertain Predictions

If an AI system makes a prediction with low confidence, it may need to raise the case with humans and let them decide, but with the analyzed results as additional information. This task aims to check if the explanations can help humans comprehend the situation and correctly classify the input text or not. To set up, we use a well-trained model and an input text classified by this model with low confidence ($\max_c \mathbf{p}_c < \tau_l$ where $\tau_l$ is a threshold parameter). Then we apply the explanation method of interest to find top-$m$ evidence and top-$m$ counter-evidence texts of the predicted class. We present both types of evidence to humans[1] together with the predicted class and probability $\mathbf{p}$ and ask the humans to use all the information to guess the actual class of the input text, without seeing the input text itself. The scoring criteria of this task are similar to the previous tasks except that we do not provide the "no preference" option as the humans can still rely on the predicted scores when all the explanations are unhelpful.

## 4 Experimental Setup

### 4.1 Datasets

We used two English textual datasets for the three tasks.

(1) **Amazon** Review Polarity is a sentiment analysis dataset with positive and negative classes (Zhang et al., 2015). We randomly selected 100K, 50K, and 100K examples for training, validating, and testing the CNN models, respectively.

(2) **ArXiv** Abstract is a text classification dataset we created by collecting abstracts of scientific articles publicly available on ArXiv[2]. Particularly, we collected abstracts from the "Computer Science (CS)", "Mathematics (MA)", and "Physics (PH)" categories, which are the three main categories on ArXiv. We then created a dataset with three disjoint classes removing the abstracts which belong to more than one of the three categories. In the experiments, we randomly selected 6K, 1.5K, and 10K examples for training, validating, and testing the CNN model, respectively.

### 4.2 Classification Models: 1D CNNs

As for the classifiers, we used 1D CNNs with the same structures for all the tasks and datasets. Specifically, we used 200-dim GloVe vectors as non-trainable weights in the embedding layer (Pennington et al., 2014). The convolution layer had three filter sizes [2, 3, 4] with 50 filters for each size, while the intermediate fully-connected layer had 150 units. The activation functions of the filters and the fully-connected layers are ReLU (except the softmax at the output layer). The models were implemented using Keras and trained with Adam optimizer. The macro-average F1 are 0.90 and 0.94 for the Amazon and the ArXiv datasets, respectively. Overall, the ArXiv appears to be an easier task as it is likely solvable by looking at individual keywords. In contrast, the Ama-

[1] We present counter-evidence as evidence for the other classes to simplify the task questions.

[2] https://arxiv.org

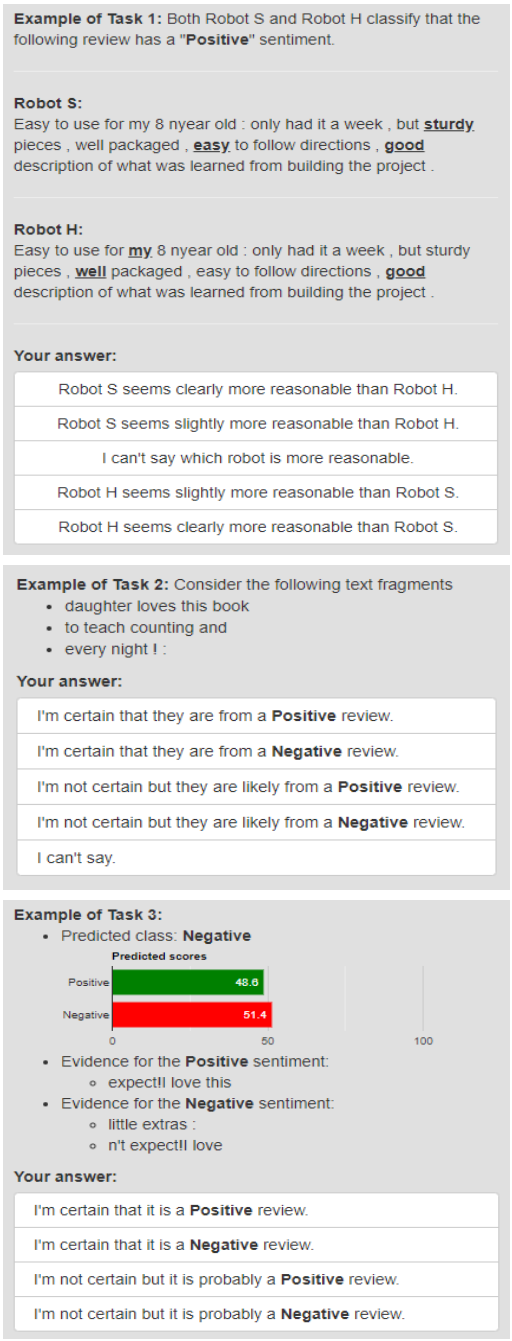


Figure 2: Example questions and user interfaces.

zon sentiment analysis is not quite easy. Many reviews mention both pros and cons of the products, so a classifier needs to analyze several parts of the input to reach a conclusion. However, this is still manageable by the CNN architecture we used.

Also, in task 1, we need another model which performs worse than the well-trained model. In this experiment, we trained the second CNNs (i.e., the worse models) for the two datasets in different ways to examine the capability of explanation methods in two different scenarios. For the Amazon dataset, while the first (well-trained) CNN needed eight epochs until the validation loss converges, we trained the second CNN for only one epoch to make it underfitting. For the ArXiv dataset, we trained the second CNN using the same number of examples as the first model but with more specific topics. To explain, we randomly selected only examples from the subclass 'Computation and Language', 'Dynamical Systems', and 'Quantum Physics' as training and validation examples for the class 'Computer Science', 'Mathematics', and 'Physics', respectively. In other words, the training and testing data of the worse CNN came from different distributions. As a result, the macro-average F1 of the worse CNNs are 0.81 and 0.85 for the Amazon and the ArXiv datasets, respectively.

| Method Name | Approach | Granularity |
|---|---|---|
| Random (W) | Random Baselines | Words |
| Random (N) | | N-grams |
| LIME | Perturbation | Words |
| LRP (W) | Relevance Propagation | Words |
| LRP (N) | | N-grams |
| DeepLIFT (W) | | Words |
| DeepLIFT (N) | | N-grams |
| Grad-CAM-Text | Gradient | N-grams |
| Decision Trees (DTs) | Model Extraction | N-grams |

Table 2: Nine explanation methods evaluated.

### 4.3 Explanation Methods

We evaluated nine explanation methods as summarized in Table 2. First, we used Random (W) and Random (N) as two baselines selecting words and non-overlapping n-grams randomly from the input text as evidence and counter-evidence. For the n-gram random baseline (and other n-gram based explanation methods in this paper), n is one of the CNN filter sizes [2, 3, 4].

Second, we selected LIME which is a well-known model-agnostic perturbation-based method (Ribeiro et al., 2016). It trains a linear model using samples (5,000 samples in this paper) around the input text to explain the importance of each word towards the prediction. The importance scores can be either positive (for the predicted class) or negative (against the predicted class).

Third, we selected layer-wise relevance propagation (LRP), specifically $\epsilon$-LRP (Bach et al., 2015b), and DeepLIFT (Shrikumar et al., 2017) which are applicable to neural networks in general and performed very well in several evalua-

tions using proxy tasks (Xiong et al., 2018; Poerner et al., 2018). LRP propagates the output of the target class (before softmax) back through layers to find attributing words, while DeepLIFT does the same but propagates the difference between the output and the predicted output of the reference input (i.e., all-zero embeddings in this paper). These two methods assign relevance scores to every word in the input text. Words with the highest and the lowest scores are selected as evidence for and counter-evidence against the predicted class, respectively. Also, we extended LRP and DeepLIFT to generate explanations at an n-gram level. We considered all possible n-grams in the input text where n is one of the CNN filter sizes. Then the explanations are generated based on the relevance score of each n-gram, i.e., the sum of relevance scores of all words in the n-gram.

Next, we searched for model-specific explanation methods which target 1D CNNs for text classification. We found that Jacovi et al. (2018) proposed one: listing only n-grams corresponding to feature values in $\mathbf{v}$ (see Figure 1) that pass thresholds for their filters. Each of the thresholds is set subject to sufficient purity of the classification results above it. However, their method is applicable to CNNs with only one linear layer as $FC$, while our CNNs have an additional hidden layer (with ReLU activation). So, we could not compare with their method in this work. To increase diversity in the experiments, we therefore propose two additional model-specific methods applicable to 1D CNNs with multiple layers in $FC$, presented next.

#### 4.3.1 Grad-CAM-Text

We adapt Grad-CAM (Selvaraju et al., 2017), originally devised for explaining 2D CNNs, to find the most relevant n-grams for text classification. Since each value in the feature vector $\mathbf{v}$ corresponds to an n-gram selected by a filter, we use $E_{j,k}$ to show the effect of an n-gram selected by the $k^{\text{th}}$ filter towards the prediction of class $j$:

$$E_{j,k} = |\max(\frac{\partial FC(\mathbf{v})_j}{\partial \mathbf{v}_k}, 0)| \times \mathbf{v}_k. \quad (1)$$

The partial derivative term shows how much the prediction of class $j$ changes if the value from the $k^{\text{th}}$ filter slightly changes. As we are finding the evidence for the target class $j$, we consider only the positive value of the derivative. Then $E_{j,k}$ combines this term with the strength of $\mathbf{v}_k$ to show the overall effect of the $k^{\text{th}}$ filter for the input text. Next, we calculate the effect of each word $w_i$ in the input text by aggregating the effects of all the n-grams containing $w_i$.

$$E_{j,w_i} = \sum_k (E_{j,k} \times \mathbb{I}[w_i \in N_k])$$

where $N_k$ is an n-gram detected by the $k^{\text{th}}$ filter. Lastly, we select, as the evidence, non-overlapping n-grams which are detected by at least one of the filters and have the highest sums of the effects of all the words they contain. For example, to decide whether we will select the n-gram $N_k$ as an evidence text or not, we consider $\sum_{w_i \in N_k} E_{j,w_i}$. Note that we can find counter-evidence by changing, in equation (1), from $\max$ to $\min$.

#### 4.3.2 Decision Trees

This explanation method is based on model extraction (Bastani et al., 2017). We create a decision tree ($DT$) which mimics the behavior of the classification part (fully-connected layers) of the trained CNN. Given a filter-based feature vector $\mathbf{v}$, the DT needs to predict the same class as predicted by the CNN. Formally, we want

$$DT(\mathbf{v}) = \underset{j}{\text{argmax}}\, \mathbf{p}_j = \underset{j}{\text{argmax}}\, FC(\mathbf{v})_j.$$

For multi-class classification, we construct one DT for each class (one vs. rest classification). We employ CART with Gini index for learning DTs (Leo et al., 1984). All the training examples are generated by the trained CNN using a training dataset, whereas a validation dataset is used to prune the DTs to prevent overfitting.

Also, for each feature $v_j$ in $\mathbf{v}$, we calculate the Pearson's correlation between $v_j$ and the output of each class (before softmax) in $FC(\mathbf{v})$ using the training dataset, so we know which class is usually predicted given a high score of $v_j$ (i.e., correlated most to this feature). We use $c_j$ denoting the most correlated class of the feature $v_j$.

We can consider the DTs as a global explanation of the model as it explains the CNN in general. To create a local explanation, we use the DT of the predicted class to classify the input. At each decision node, we collect associated n-grams passing the nodes' thresholds to be evidence for (or counter-evidence against) the predicted class (depending on the most correlated class of each splitting feature). For example, an input text $X$ is classified to class $a$, so we use the DT of class $a$ to predict the input. If a decision node checks whether feature $v_j$ of this input is greater than 0.25 and assume it is true for this input, the n-gram corresponding to $v_j$ will be evidence if the most correlated class of $v_j$ is class $a$ (i.e., $c_j = a$). Otherwise, it will be counter-evidence if $c_j \neq a$.

**An example from the Amazon dataset, Actual: Pos, Predicted: Pos, (Predicted scores: Pos 0.514, Neg 0.486):**
"*OK but not what I wanted: These would be ok but I didn't realize just how big they are. I wanted something I could actually cook with. They are a full 12" long. The handles didn't fit comfortably in my hand and the silicon tips are hard, not rubbery texture like I'd imagined. The tips open to about 6" between them. Hope this helps someone else know ...*"

| Method | Top-3 evidence texts | Top-3 counter-evidence texts |
|---|---|---|
| LIME (W) | comfortably / wanted / helps | not / else / someone |
| LRP (W) | are / not / 6 | : / tips / open |
| LRP (N) | are hard , not / about 6" between / not what I wanted | . The tips open / : These would / in my hand and |
| Grad-CAM-Text (N) | comfortably in my hand / I wanted : These / . The tips open | not what I wanted / not rubbery texture like / Hope this helps someone |
| DTs (N) | imagined . The tips | 'd imagined . / are . I wanted / would be ok |

Table 3: Examples of evidence and counter-evidence texts generated by some of the explanation methods.

### 4.4 Implementations[3]

We used public libraries of LIME[4], LRP (Alber et al., 2018), and DeepLIFT[5] in our experiments. Besides, the code for computing Grad-CAM-Text was adapted from keras-vis[6], whereas we used scikit-learn (Pedregosa et al., 2011) for decision tree construction. All the DTs achieved over 80% macro-F1 in mimicking the CNNs' predictions.

For the task parameters, we set $m = 3$, $\tau_h = 0.9$, and $\tau_l = 0.7$. For each task and dataset, we used 100 input texts, half of which were classified correctly by the model(s) and the rest were misclassified. So, with nine explanation methods being evaluated, each task had 900 questions per dataset for human participants to answer. Examples of questions for each task are given in Figure 2.

For the Amazon dataset, we posted our tasks on Amazon Mechanical Turk (MTurk). To ensure the quality of crowdsourcing, each question was answered by three workers and the scores were averaged. For the ArXiv dataset which requires background knowledge of the related subjects, we recruited graduates and post-graduate students in Computer Science, Mathematics, Physics, and Engineering to perform the tasks, and each question was answered by one participant. In total, we had 367 and 121 participants for the Amazon and the ArXiv datasets, respectively.

## 5 Results and Discussion

Examples of the generated explanations are shown in Table 3 and a separate appendix. Table 4 shows the average scores of each explanation method for each task and dataset, while Figure 3 displays the distributions of individual scores for all three tasks. We do not show the distributions of tasks 2 and 3 of the Amazon dataset as they look similar to the associated ones of the ArXiv dataset.

### 5.1 Task 1

For the Amazon dataset, though Grad-CAM-Text achieved the highest overall score, the performance was not significantly different from other methods including the random baselines. Also, the inter-rater agreement for this task was quite poor. It suggests that existing explanation methods cannot apparently reveal irrational behavior of the underfitting CNN to lay human users. So, the scores of most explanation methods distribute symmetrically around zero, as shown in Figure 3(a).

For the ArXiv dataset, LRP (N) and DeepLIFT (N) got the highest scores when both CNNs predicted correctly. Hence, they can help humans identify the poor model to some extent. However, there was no clear winner when both CNNs predicted wrongly. One plausible reason is that evidence for an incorrect prediction, even by a well-trained CNN, is usually not convincing unless we set a (high) lower bound of the confidence of the predictions (as we did in task 2).

Additionally, we found that psychological factors fairly affect this task. Based on the results, for two explanations with comparable semantic quality, humans prefer the explanation with more evidence texts and select it as more reasonable. This is consistent with the findings by Zemla et al. (2017). Conversely, the DTs method performed in the opposite way. The DT of the better model usually focuses on a few most relevant texts in the input and outputs fewer evidence texts. This possibly causes the low performance of the DTs method in this task. Also, we got feedback from the participants that they sometimes penalized an evidence

[3]The code and datasets of this paper are available at https://github.com/plkumjorn/CNNAnalysis

[4]https://github.com/marcotcr/lime

[5]https://github.com/kundajelab/deeplift

[6]https://github.com/raghakot/keras-vis

| Explanation Method | Task 1 | | | | | | Task 2 | | | | | | Task 3 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Amazon | | | ArXiv | | | Amazon | | | ArXiv | | | Amazon | | | ArXiv | | |
| | $\mathcal{A}$ | ✔ | ✘ | $\mathcal{A}$ | ✔ | ✘ | $\mathcal{A}$ | ✔ | ✘ | $\mathcal{A}$ | ✔ | ✘ | $\mathcal{A}$ | ✔ | ✘ | $\mathcal{A}$ | ✔ | ✘ |
| Random (W) | .02 | .00 | .04 | -.11 | -.05 | -.17 | .06 | .10 | .02 | .07 | .09 | .04 | .05 | .53 | -.43 | .01 | .32 | -.30 |
| Random (N) | .02 | .02 | .02 | -.12 | -.16 | -.07 | .12 | .13 | .12 | .29 | .32 | .25 | -.01 | .54 | -.55 | .02 | .29 | **-.25** |
| LIME (W) | -.02 | .02 | -.06 | .03 | .02 | **.03** | **.69** | **.74** | .64 | **.70** | **.75** | **.64** | .02 | .50 | -.45 | -.02 | .31 | -.34 |
| LRP (W) | .00 | -.01 | .02 | -.03 | -.01 | -.05 | .13 | .26 | -.01 | .26 | .36 | .16 | -.02 | .50 | -.54 | -.06 | .33 | -.44 |
| LRP (N) | -.07 | -.04 | -.09 | **.12** | **.24** | -.01 | .26 | .45 | .08 | .44 | .49 | .39 | .08 | .60 | -.43 | **.17** | **.60** | -.26 |
| DeepLIFT (W) | .04 | .03 | .04 | .07 | .13 | .00 | .21 | .37 | .04 | .26 | .35 | .16 | -.03 | .47 | -.53 | -.08 | .28 | -.44 |
| DeepLIFT (N) | .06 | .06 | **.05** | .06 | .22 | -.10 | .23 | .47 | -.01 | .38 | .47 | .28 | .05 | .59 | -.49 | .02 | .33 | -.30 |
| Grad-CAM-T (N) | **.07** | **.11** | .03 | -.03 | -.04 | -.01 | .65 | .64 | **.66** | .53 | .65 | .41 | .05 | .51 | -.42 | .06 | .56 | -.45 |
| DTs (N) | -.05 | -.02 | -.08 | -.13 | -.22 | -.03 | .64 | .68 | .59 | .51 | .69 | .32 | **.10** | **.60** | **-.40** | -.11 | .29 | -.50 |
| Fleiss $\kappa$ (Amazon) | 0.050 / 0.054 | | | N/A | | | 0.274 / 0.371 | | | N/A | | | 0.212 / 0.499 | | | N/A | | |

Table 4: The average scores of the three evaluation tasks. The score range is [-1,1] in which 1 is better. $\mathcal{A}$, ✔, and ✘are for all, correctly classified, and misclassified input texts, respectively. Boldface numbers are the highest average scores in the columns. A number is underlined when there is no statistically significant difference between the scores of the corresponding method and the best method in the same column (at a significance level of 0.05). The last row reports inter-rater agreement measures (Fleiss' kappa) in the format of $\alpha$ / $\beta$ where $\alpha$ considers answers with human confidence levels (5 categories for task 1-2 and 4 categories for task 3) and $\beta$ considers answers regardless of the human confidence levels (3 categories for task 1-2 and 2 categories for task 3).

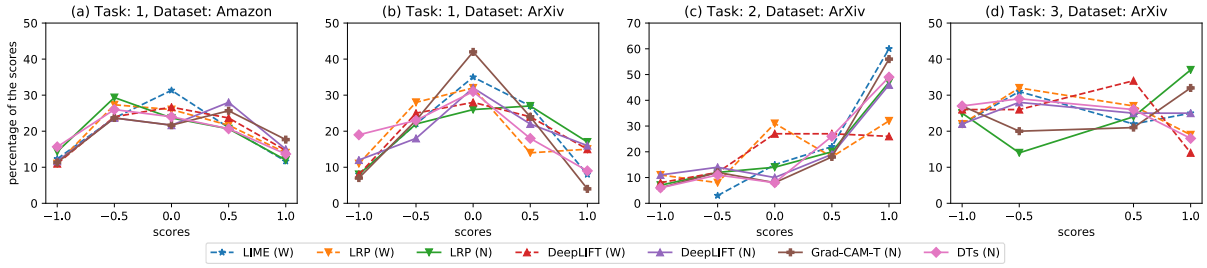


Figure 3: Score distributions from task 1 of the Amazon dataset and from all the tasks of the ArXiv dataset.

text which is highlighted in a strange way, such as "... greedy algorithm. In this paper, we ...". Hence, in real applications, syntax integrity should be taken into account to generate explanations.

### 5.2 Task 2

LIME clearly achieved the best results in task 2 followed by Grad-CAM-Text and DTs. These methods are class discriminative, being able to find good evidence for the predicted class regardless of whether the prediction is correct.

We believe that LIME performed well because it tests that the missing of evidence words from the input text greatly reduces the probability of the predicted class, so these words are semantically related to the predicted class (given that the model is accurate). Meanwhile, the DTs method selects evidence based on the most correlated class of the splitting features. So, the evidence n-grams are more likely related to the predicted class than the other classes. However, they may be less relevant than LIME's as the evidence is generated from a global explanation of the model (DTs). Besides, Grad-CAM-Text worked relatively well here probably because it preserves the class discriminative property of Grad-CAM (Selvaraju et al., 2017).

By contrast, LRP and DeepLIFT generated acceptable evidence only for the correct predictions. Also, LRP (N) and DeepLIFT (N) performed better than LRP (W) and DeepLIFT (W) in both datasets. This might be because one evidence n-gram contains more information than one evidence word. Nevertheless, even the Random (N) method surpasses the LRP (W) and the DeepLIFT (W) for the ArXiv dataset. Thereby, whenever we use LRP and DeepLIFT, we should present to humans the most relevant words together with their contexts.

### 5.3 Task 3

The negative scores under the ✘columns of task 3 show that using explanations to rectify the predictions is not easy. Hence, the overall average scores of many explanation methods stay close to zero.

DTs performed well only on the Amazon dataset. The average numbers of n-grams per explanation, generated by the DTs, are 2.00 and 1.77

for the Amazon and ArXiv datasets, respectively. Also, the reported n-grams could be repetitive and overlapping. This reduced the amount of useful information displayed, and it may be insufficient for humans to choose one of the CS, MA, and PH categories, which are more similar to one another than the positive and negative sentiments.

Meanwhile, LRP (N) performed consistently well on both datasets. This is reasonable considering our discussions in task 2. First, LRP (N) generates good evidence for correct predictions, so it can gain high scores in the ✔columns. On the other hand, the evidence for incorrect predictions (✘) is usually not convincing, so the counter-evidence (which is likely to be the evidence of the correct class) can attract humans' attention. Furthermore, the fact that LRP is not class discriminative does not harm it in this task as humans can recognize an evidence text even if it is selected by the LRP (N) as counter-evidence (and vice versa).

For example, in the ArXiv dataset, we found a case in which the predicted class is PH (score = 0.48) but the correct class is CS (score = 0.07). LRP (N) selected 'armed bandit settings with', 'the Wasserstein distance', and 'derive policy gradients' as evidence for the class PH. These n-grams, however, are not truly related to PH. Rather, they revealed the true class of this text and made a human choose the CS option with high confidence despite the low predicted score.

Regarding LIME, the situation is reversed as LIME can find both good evidence and counter-evidence. These make humans be indecisive and, possibly, select a wrong option as the explanation is presented at a word level (without any contexts).

### 5.4 Model Complexity

Apart from the results of the three tasks, it is worth to discuss the size of the DTs which mimic the four CNNs in our experiments. As shown in Table 5, the size of the DTs can reflect the complexity of the CNNs. Although the well-trained CNN of the Amazon dataset got 0.9 F1 score, the DTs of this CNN needed more than 5,500 nodes to achieve 85% fidelity (compared to only hundreds of nodes required for the ArXiv dataset). This illustrates the high complexity of the Amazon task compared to the ArXiv task even though both tasks were managed effectively by the same CNN architecture.

For the ArXiv dataset, the DTs of the poor CNN are smaller than the ones of the well-trained CNN.

| | #Nodes | Depth | #Leaves |
|---|---|---|---|
| **Amazon: 1st CNN (well-trained)** – $F = 0.85$ | | | |
| Negative | 5535 | 38 | 2768 |
| Positive | 5537 | 45 | 2769 |
| **Amazon: 2nd CNN (underfitting)** – $F = 0.82$ | | | |
| Negative | 6405 | 40 | 3203 |
| Positive | 6369 | 40 | 3185 |
| **ArXiv: 1st CNN (well-trained)** – $F = 0.89$ | | | |
| Computer Science | 363 | 25 | 182 |
| Mathematics | 565 | 24 | 283 |
| Physics | 325 | 24 | 163 |
| **ArXiv: 2nd CNN (specific data)** – $F = 0.84$ | | | |
| Computer Science | 107 | 17 | 54 |
| Mathematics | 263 | 28 | 132 |
| Physics | 237 | 29 | 119 |

Table 5: Metadata of the DTs in the experiments. $F$ refers to fidelity of the DTs (macro-average F1).

This is likely because the poor CNN was trained on a specific dataset (i.e., selected subtopics of the main categories), so it had to deal with fewer discriminative patterns in texts compared to the first CNN trained using texts from all subtopics.

Further studies of this quality of the DTs would be useful for some applications, e.g., measuring model complexity (Bianchini and Scarselli, 2014) and model compression (Cheng et al., 2018).

## 6 Conclusion

We proposed three human tasks to evaluate local explanation methods for text classification. Using the tasks in this paper, we experimented on 1D CNNs and found that *(i)* LIME is the most class discriminative method, justifying predictions with relevant evidence; *(ii)* LRP (N) works fairly well in helping humans investigate uncertain predictions; *(iii)* using explanations to reveal model behavior is challenging, and none of the methods achieved impressive results; *(iv)* whenever using LRP and DeepLIFT, we should present to humans the most relevant words together with their contexts and *(v)* the size of the DTs can also reflect the model complexity. Lastly, we consider evaluating on other datasets and other advanced architectures beneficial future work as it may reveal further interesting qualities of the explanation methods.

## Acknowledgments

We would like to thank all the participants in the experiments as well as Alon Jacovi and anonymous reviewers for helpful comments. Also, the first author would like to thank the support from Anandamahidol Foundation, Thailand.

## References


Maximilian Alber, Sebastian Lapuschkin, Philipp Seegerer, Miriam Hägele, Kristof T Schütt, Grégoire Montavon, Wojciech Samek, Klaus-Robert Müller, Sven Dähne, and Pieter-Jan Kindermans. 2018. innvestigate neural networks! *arXiv preprint arXiv:1808.04260*.

Leila Arras, Franziska Horn, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. 2016. Explaining predictions of non-linear classifiers in nlp. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 1–7. Association for Computational Linguistics.

Leila Arras, Franziska Horn, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. 2017a. " what is relevant in a text document?": An interpretable machine learning approach. *PloS one*, 12(8):e0181142.

Leila Arras, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. 2017b. Explaining recurrent neural network predictions in sentiment analysis. In *Proceedings of the EMNLP'17 Workshop on Computational Approaches to Subjectivity, Sentiment & Social Media Analysis (WASSA)*, pages 159–168. Association for Computational Linguistics.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. 2015a. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE*, 10(7):e0130140.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. 2015b. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140.

Osbert Bastani, Carolyn Kim, and Hamsa Bastani. 2017. Interpretability via model extraction. *CoRR*, abs/1706.09773.

Monica Bianchini and Franco Scarselli. 2014. On the complexity of neural network classifiers: A comparison between shallow and deep architectures. *IEEE Transactions on Neural Networks and Learning Systems*, 25(8):1553–1565.

Or Biran and Kathleen R McKeown. 2017. Human-centric justification of machine learning predictions. In *IJCAI*, pages 1461–1467.

Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. 2018. Model compression and acceleration for deep neural networks: The principles, progress, and challenges. *IEEE Signal Processing Magazine*, 35(1):126–136.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2017. Very deep convolutional networks for text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1107–1116. Association for Computational Linguistics.

Yannis Dimopoulos, Paul Bourret, and Sovan Lek. 1995. Use of some sensitivity criteria for choosing networks with good generalization ability. *Neural Processing Letters*, 2(6):1–4.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using convolutional neural networks to classify hate-speech. In *Proceedings of the First Workshop on Abusive Language Online*, pages 85–90. Association for Computational Linguistics.

Reza Ghaeini, Xiaoli Fern, and Prasad Tadepalli. 2018. Interpreting recurrent and attention-based neural models: a case study on natural language inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4952–4957, Brussels, Belgium. Association for Computational Linguistics.

Alon Jacovi, Oren Sar Shalom, and Yoav Goldberg. 2018. Understanding convolutional neural networks for text classification. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 56–65. Association for Computational Linguistics.

Rie Johnson and Tong Zhang. 2015. Effective use of word order for text categorization with convolutional neural networks. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 103–112. Association for Computational Linguistics.

Been Kim, Cynthia Rudin, and Julie Shah. 2014. The bayesian case model: A generative approach for case-based reasoning and prototype classification. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 1952–1960, Cambridge, MA, USA. MIT Press.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 107–117. Association for Computational Linguistics.

Breiman Leo, Jerome H Friedman, Richard A Olshen, and Charles J Stone. 1984. Classification and regression trees. *Wadsworth International Group*.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. Understanding neural networks through representation erasure. *CoRR*, abs/1612.08220.

Hui Liu, Qingyu Yin, and William Yang Wang. 2019. Towards explainable NLP: A generative explanation framework for text classification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5570–5581. Association for Computational Linguistics.

Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pages 4765–4774.

Sina Mohseni and Eric D Ragan. 2018. A human-grounded evaluation benchmark for local explanations of machine learning. *arXiv preprint arXiv:1801.05075*.

Dong Nguyen. 2018. Comparing automatic and human evaluation of local explanations for text classification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1069–1078.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Nina Poerner, Hinrich Schütze, and Benjamin Roth. 2018. Evaluating neural network explanation methods using hybrid documents and morphosyntactic agreement. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 340–350.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 1135–1144, New York, NY, USA. ACM.

Wojciech Samek, Thomas Wiegand, and Klaus-Robert Müller. 2018. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *ITU Journal: ICT Discoveries - Special Issue 1 - The Impact of Artificial Intelligence (AI) on Communication Networks and Services*, 1(1):39–48.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 618–626.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3145–3153, International Convention Centre, Sydney, Australia. PMLR.

David Silver, Aja Huang, Christopher J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503.

Panagiotis Symeonidis, Alexandros Nanopoulos, and Yannis Manolopoulos. 2009. Moviexplain: a recommender system with explanations. *RecSys*, 9:317–320.

Wenting Xiong, Iftitahu Ni'mah, Juan MG Huesca, Werner van Ipenburg, Jan Veldsink, and Mykola Pechenizkiy. 2018. Looking deeper into deep learning model: Attribution-based explanations of textcnn. *arXiv preprint arXiv:1811.03970*.

Jeffrey C Zemla, Steven Sloman, Christos Bechlivanidis, and David A Lagnado. 2017. Evaluating everyday explanations. *Psychonomic bulletin & review*, 24(5):1488–1500.

Jingqing Zhang, Piyawat Lertvittayakumjorn, and Yike Guo. 2019. Integrating semantic knowledge to tackle zero-shot text classification. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1031–1040. Association for Computational Linguistics.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 649–657, Cambridge, MA, USA. MIT Press.

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. 2016. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929.

# A Context-based Framework for Modeling the Role and Function of On-line Resource Citations in Scientific Literature

**He Zhao**[1], **Zhunchen Luo**[2], **Chong Feng**[1]†, **Anqing Zheng**[1], **Xiaopeng Liu**[2]
[1]Beijng Institute of Technology, School of Computer Science, Beijing, China
[2]Information Research Center of Military Science, PLA Academy of Military Science, Beijing, China
zhaohe1995@outlook.com zhunchenluo@gmail.com
fengchong@bit.edu.cn aqzheng1995@outlook.com liuxiaopeng1975@163.com

## Abstract

We introduce a new task of *modeling the role and function for on-line resource citations in scientific literature*. By categorizing the on-line resources and analyzing the purpose of resource citations in scientific texts, it can greatly help resource search and recommendation systems to better understand and manage the scientific resources. For this novel task, we are the first to create an annotation scheme, which models the different granularity of information from a hierarchical perspective. And we construct a dataset SciRes, which includes 3,088 manually annotated resource contexts. In this paper, we propose a possible solution by using a multi-task framework to build the scientific resource classifier (SciResCLF) for jointly recognizing the role and function types. Then we use the classification results to help a scientific resource recommendation (SciResREC) task. Experiments show that our model achieves the best results on both the classification task and the recommendation task. The SciRes dataset[1] will be released for future research.

## 1 Introduction

In this paper, we introduce a new task of *modeling the role and function for on-line resource citations in scientific literature*. With the number of scientific publications growing dramatically, numerous on-line resources are mentioned, released and used within the scientific literature. Tracing and modeling these resources such as software, tools and datasets can greatly help researchers by developing scientific resource searching and recommendation systems or constructing scientific resource knowledge graphs. Google has launched a new search engine[2] in 2018 to help scientists find the datasets they need, whereas the retrieved datasets can only be matched with their official names. Other current academic search engines such as Google Scholar can be only used for detecting the relevant papers where a certain resource is mentioned. However, the limitation actually exists is that the user can not learn more fine-grained information such as what role the resource plays in its paper and why the resource is cited in its contexts. To improve present works, we propose a context-based framework to model both the *resource role* and the *resource function* for on-line resource citations in scientific literature.

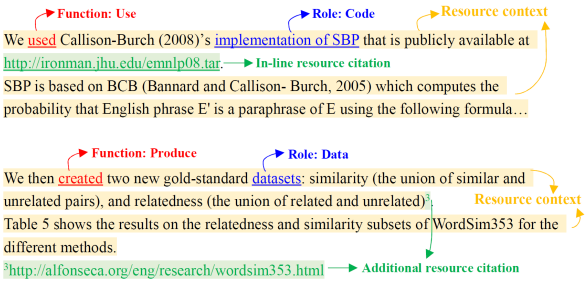


Figure 1: Examples for the two types of resource citations in scientific literature. We note the arguments (which are mostly the key verbs before the citation) for identifying the resource function and the arguments (which are mostly the target nominals before the citation) for identifying the resource role.

As is shown in Figure 1, we give examples for the two types of resource citations considering their appearances in the original publications. Through observing more than three hundred scientific publications from different domains of computer science, we find that most resource citations can be divided into two types according to the locations of their hyperlinks: the **in-line resource citations** in bodytexts and the **additional resource citations** in footnotes. We define a **resource citation** as a hyperlink mentioned in the

†Corresponding author.
[1]https://github.com/zhaohe1995/SciRes
[2]https://toolbox.google.com/datasetsearch



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5206–5215,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

scientific paper text, which links to a specific on-line resource. A **resource context** is a word sequence surrounding the resource citation. As Figure 1 shows, we set the context window size to 5 in our work. So that the context includes the sentence where the hyperlink is mentioned, the two sentences to the left and the two to the right. The **resource role** is the class of a resource indicating what role the resource plays in its context (e.g. *Code* and *Data* in Figure 1). The **resource function** is the specific purpose performed by the resource with respect to the current paper's work, indicating why the author has cited this resource here (e.g. *Use* and *Produce* in Figure 1). It's necessary to note that one resource may have different functions in a same paper due to the different contexts.

The information of resource role and function is very important for building applications to assist scientific research. To help develop more powerful resource search systems, identifying the role can enrich the resource repository and identifying the function is crucial for understanding more complicated queries and providing more accurate results. Moreover, to help researchers quickly be acquainted with the scientific resources and easily find applicable resources for their work, the scientific resource recommendation system will have great application prospects in the future. Previous works mostly make the efforts on the task of *citation function classification* (Teufel et al., 2006; Jurgens et al., 2018) and the task of *context-based citation recommendation* (Tang and Zhang, 2009; He et al., 2011; Huang et al., 2015). Different from previous works, our work specially focuses on the on-line resources in scientific text, which are not as well studied as the paper citations. To the best of our knowledge, we are the first to model the on-line resource citations on such a fine-grained level in scientific full text.

In this paper, we first propose a new annotation scheme, which models both the general role types and the fine-grained role types from a two-hierarchy perspective, and models the function types by analyzing the purpose of citing the resources. Based on the scheme, we construct a dataset SciRes by manually annotating more than three thousand resource contexts. To better classify each on-line resource citation and benefit from the associated information between roles and functions, we apply a multi-task framework SciResCLF, which jointly identifies the resource role types and the resource function types based on the word sequences of resource contexts. Experiments show that our model outperforms all the baselines on the classification task. We further address a context-based resource recommendation task and develop a framework SciResREC, which predicts the resource hyperlinks only given the masked resource contexts. Using the classification results, our model achieves good performance by the help of the role and function information in the recommendation task.

In summary, we make the following contributions. We introduce a novel task of modeling fine-grained information, especially the resource role and the resource function of scientific on-line resource citations. For this task, we propose a new annotation scheme and create a dataset for resource classification. We develop a multi-task learning model which can jointly identify the general role types, fine-grained role types and functions for the resource citations. Based on our united classifier we solve a resource recommendation task and analyze the evolution and maturity of the scientific resources.

## 2 Related Work

**Context-based Citation Analysis**

Context-based citation analysis addresses a citation's value by interpreting each citation based on its context at both the syntactic and semantic levels. One research focus is the *citation function*, which is defined as the authors reason for citing a given paper. Specific schemes for classifying the function of a citation have been developed (Teufel et al., 2006; Liakata et al., 2012; Jurgens et al., 2018). Jurgens et al. (2018) did their efforts to measure the evolution of a scientific field by framing the citation function. In the past years, many annotation schemes and approaches for identifying arguments in various domains have been developed. Following these previous schemes, we develop our annotation scheme especially for modeling the role and function for scientific resource, which will be detailed in Section 3. Other previous studies focus on the *context-based citation recommendation* (Tang and Zhang, 2009; He et al., 2010, 2011; Huang et al., 2012, 2015), which aims to recommend a short list of papers that need to be cited within the given context. A citation context is defined as a sequence

| | #Bodytext | #Footnote | Total | Annotation |
|---|---|---|---|---|
| ARC | 3,718 | 15,043 | 18,761 | 1,026 |
| NeurIPS | 686 | 896 | 1,582 | 1,056 |
| PUBMED | 31,852 | 510 | 32,362 | 1,006 |
| Total | 36,256 | 16,449 | 52,705 | 3,088 |

Table 1: A description for the collection of 52,705 data sample in the scientific resource corpus.

of words that appear around a particular citation. Based on this definition, we develop the concepts of *resource citation* and *resource context* in this paper.

**Resource Discovery for Scientific Text**

There are also some relevant research which focus on detecting resources in biomedical literature. Some approaches to this end are reported in (Duck et al., 2012, 2014, 2016; Yamamoto and Takagi, 2007; de la Calle et al., 2009). Yamamoto and Takagi (2007) crawled abstracts from MEDLINE and full text from BioMed to extract the URLs of resources using regular expressions and heuristic rules. The BIRI addressed by de la Calle et al. (2009) utilized handcrafted regular expressions expressed as transition networks for resource naming, functionality detection and resource classification to generate resource inventories. Duck et al. (2012) proposed the BioNERDS, which is a NER system for detecting database and software names in scientific literature using extraction rules generated by examining 30 bioinformatics articles. However, there has been no existing framework for modeling the resource role and function at such a fine-grained level in general domain scientific text.

## 3 Dataset

There are some current scientific literature datasets (e.g. Semantic Scholar Open Research Corpus[3] and Arxiv Dataset[4]) contain the structured metadata such as title, author and abstract. While as is illustrated in Figure 1, most on-line resource citations are placed in the full texts of the papers. We need to locate the resource citations and extract the resource contexts first to perform further analysis. To the best of our knowledge, due to the difficulties in collecting large scale of scientific full texts from the PDF publications, there is no ready-to-use dataset for our task. Hence we collect a large scientific resource context corpus of 52,705 data samples and construct a manually annotated dataset (called **SciRes**) which includes annotations for the resource role types and the resource function types. SciRes attempts to interpret the role of resource citations from a hierarchical perspective, with both the general role types and more fine-grained role types. And SciRes aims at understanding the author's intention of citing a resource by the function types. We hope the SciRes can facilitate future research for context-based resource analyzing in scientific literature.

### 3.1 Data Processing

We used the paper full texts from three different sources: the ACL Anthology Reference Corpus (ARC)[5], a corpus of scientific publications about computational linguistics; the NeurIPS Proceedings (NeurIPS)[6], a corpus of conference proceedings about neural information; and the PubMed[7], an archive of biomedical and life sciences journal literature. We collected 21,411 papers of ARC up to 2015, 7,147 papers of NeurIPS from 1988 to 2017, and randomly downloaded 11,043 publications from PubMed. For each paper in the corpus, we downloaded the PDF and used Omnipage[8] to perform OCR in translating the files to characters with coordinates and properties. So that we can detect the footnotes and find their anchors in the bodytext by developing some regular expression filters. Then we applied a conditional random field-based parsing tool, ParsCit[9] to get the structural raw text. To build the SciRes dataset we first extracted all the hyperlinks as resource citations in a scientific paper from both the bodytexts and the footnotes. Three PhD students from the above three fields were asked to read 200 randomly selected resource contexts to investigate the context window size. It was found that 95% samples can be determined the resource role and function types by getting across the 5 sentences around the citations. So we set the window size to 5 and extracted the sentences along with each hyperlink as the resource context. Finally, we construct a collection of 52,705 data samples, with a detailed description shown in Table 1.

[3] http://api.semanticscholar.org/corpus/

[4] https://www.kaggle.com/neelshah18/arxivdataset

[5] http://acl-arc.comp.nus.edu.sg/

[6] http://papers.nips.cc/

[7] https://www.ncbi.nlm.nih.gov/pubmed

[8] https://www.nuance.com

[9] https://github.com/knmnyn/ParsCit

| 1st-category | 2nd-category | #ARC | #NeurIPS | #PUBMED | 1st-Total | 1st-% | 2nd-Total | 2nd-% |
|---|---|---|---|---|---|---|---|---|
| Material | Data | 333 | 452 | 219 | 1,004 | 31.0 | 1,004 | 31.0 |
| Method | Tool | 340 | 106 | 171 | 617 | 19.0 | 1,057 | 38.7 |
| | Code | 74 | 256 | 91 | 421 | 13.0 | | |
| | Algorithm | 79 | 118 | 19 | 216 | 6.7 | | |
| Supplement | Document | 81 | 97 | 132 | 310 | 9.6 | 984 | 30.3 |
| | Website | 146 | 66 | 144 | 256 | 7.9 | | |
| | Paper | 3 | 3 | 242 | 248 | 7.6 | | |
| | License | 16 | 1 | 97 | 114 | 3.5 | | |
| | Media | 4 | 49 | 3 | 56 | 1.7 | | |

Table 2: The 3 *1-st category role* types and 9 *2-nd category role* types along with their statistics in **SciRes** from three scientific literature sources.

### 3.2 Annotation Scheme

Many annotation schemes for the citation functions have been created over the past years (Teufel et al., 2006; Liakata et al., 2012; Jurgens et al., 2018). Based on these previous works and some recent annotation schemes for ScienceIE (Luan et al., 2017; Augenstein et al., 2017; Zhao et al., 2019), we address our scheme:

**3 general (1st-category) Role types:** Material, Method, Supplement.

**9 fine-grained (2nd-category) Role types:** Data, Tool, Code, Algorithm, Document, Website, Paper, License, Media.

**6 Function types:** Use, Produce, Introduce, Extend, Compare, Other.

The hierarchical relationships between 1st-category role types and 2nd-category role types are shown in Table 2. More detailed definitions and examples for each type can be found in the Appendix. Annotations were performed by a group of 3 PhD students, of which one majors in NLP, one majors in deep learning and another majors in bioinformatics. We randomly selected 1,100 data samples from each scientific literature source. Since too short text might not cover sufficient information for identifying the target role and function types, we filtered out the samples of which the context sequence had less than 10 words. Each resource citation together with its context was assigned at least one label for the general role types, one label for the fine-grained role types and a unique label for the function types. Fleiss's Kappa ($\kappa$) is 0.79 for the 1st-category resource role, 0.54 for the 2nd-category resource role and 0.65 for the resource function, indicating a relatively high agreement between annotators considering the number of class and the difficulties of the fine-grained classification task. The decision is made by a majority rule when a conflict occurs. Finally we obtained the manually annotated SciRes dataset of 3,088 data samples. All the function and role types along with their statistics are shown in Table 2 and Table 3.

| | #ARC | #NeurIPS | #PUBMED | Total | % |
|---|---|---|---|---|---|
| Use | 560 | 578 | 347 | 1,485 | 48.1 |
| Produce | 125 | 308 | 201 | 634 | 20.5 |
| Introduce | 254 | 89 | 281 | 624 | 20.2 |
| Extend | 53 | 30 | 39 | 122 | 4.0 |
| Other | 17 | 1 | 106 | 124 | 4.0 |
| Compare | 18 | 50 | 30 | 98 | 3.2 |

Table 3: The 6 *resource function* types along with their statistics in **SciRes** from three scientific literature sources.

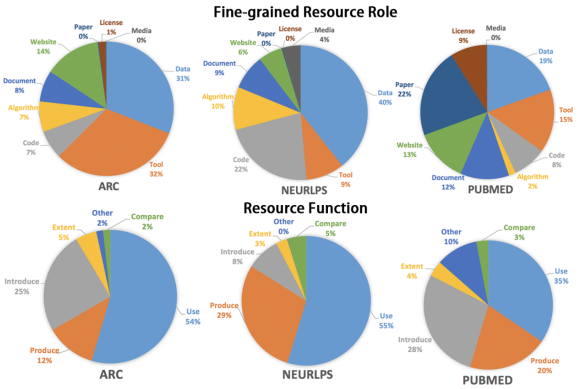


Figure 2: The distributions of resource role and function types in different data sources.

For the more fine-grained role types, the distribution is very skewed. The majority of 2-nd category roles are Data (31.0%) and then come the Tool (19.0%) and the Code (13.0%). The majority of functions are Use (48.1%) while the Compare (3.2%) and Extend (4.0%) are much fewer. The existence of a large imbalance between the types bring challenges for our classification task. There are also some findings when comparing the distributions of roles and functions between different data sources, as shown in Figure 2. For the fine-grained resource roles, the ARC dataset has relatively more Tools reflecting that there are more re-

search about NLP applications in the ARC. While the more theoretical NeurIPS has least Tools but most Algorithm and Data citations. Instead of the packaged software, the papers from NeurIPS prefer the implementations in codebases. Furthermore, due to the difference of article formats and writing styles in different domains, the papers in the field of bioinformatics from the PubMed tend to link papers by in-line hyperlinks in bodytexts while the papers from the ARC and the NeurIPS tend to cite papers in reference lists. For the resource functions, we can see that the papers from the NeurIPS tend to produce or release more new resources while the papers from the ARC and the PubMed tend to introduce more resources as background to support their works. We assume the difference in the resource function is because that the research from the NeurIPS are more theory-based and often put forward new methodologies, whereas the research from the ARC and the PubMed are more comprehensive and contain more application-oriented works which tend to review many related resources.

## 4 Classification Model

To identify the resource role types and the resource function types for the resource citations, we apply a multi-task learning framework, called **SciResCLF** to jointly classify the 1-st category roles, the 2-nd category roles and the functions by sharing the resource context representations as classification features. Figure 3 shows an overall architecture of our SciResCLF framework.

Based on our SciRes dataset, many challenges make the classification task not easy. First, it is important to parse, encode and model the semantic information in the resource citation contexts, which are relatively short texts having no more than 5 sentences. Second, as the examples shown in Figure 1, by observing the data samples we find in most cases some key nominals or verbs located nearby the citations can imply the role and function types of the resources (e.g. the nearest verb before a resource citation such as "use", "apply" or "adopt" often indicates the function of **Use**). For this reason, the relative positions of the words in the sequence is very significant information to be considered in our task. Furthermore, due to the limited labeled data most neural-based models with a large number of parameters can not be trained well.Therefore, effective methods for solving these particular challenges need to be developed for this new scientific resource classification task.

From the SciRes dataset we can find that there is a strong correlation between the role type and the function type for a resource citation in its context. To better incorporate the associated information, for our classification task we adopt a multi-task setup, which has been proven effective in the ScienceIE problems (Augenstein et al., 2017; Luan et al., 2018). Recently, the pre-trained language models, such as ELMo (Peters et al., 2018), OpenAI GPT (Radford, 2018), and BERT (Devlin et al., 2018), have shown their effectiveness to alleviate the effort of feature engineering. Especially, BERT has achieved excellent results in text classification problems such as sentiment analysis (Sun et al., 2019; Xu et al., 2019) and document classification (Adhikari et al., 2019). Based on previous works, our framework also takes advantage of the pre-trained BERT model to learn contextualized representations for the resource contexts, as shown in Figure 3. To adapt BERT to our classification tasks, following Devlin et al. (2018) we first get the hidden state corresponding to the $[CLS]$ input token. Then the hidden representation $h$ is passed into three separate softmax layers: $\mathbf{L}_{1st\_role} = softmax(W_1 h + b_1)$, $\mathbf{L}_{2nd\_role} = softmax(W_2 h + b_2)$, $\mathbf{L}_{func} = softmax(W_3 h + b_3)$, where $W_1, W_2, W_3 \in \mathrm{R}^{r_h}$, $h1, h2, h3 \in \mathrm{R}$ and $r_h$ is the size of the hidden dimension. So that given the word sequence input of a resource context, the total loss $\mathbf{L}_{CLF}$ is defined as a weighted sum of the cross entropy loss of the three multinomial classification tasks:

$$\mathbf{L}_{CLF} = \alpha \mathbf{L}_{1st\_role} + \beta \mathbf{L}_{2nd\_role} + \gamma \mathbf{L}_{func} \quad (1)$$

## 5 Recommendation Model

To show a practical application scenario of modeling the role and function for scientific resource citations, we further address a new resource recommendation task. The problem takes the scientific text sentences about an indexed resource as a query and aims to predict the hyperlink of a possible on-line resource to fill the citation blank of a resource context. We first give the formalized definition for the resource recommendation task. The input is the word sequence of a resource context $C = \{w_{n-l}, ..., w_{n-1}, w_n, w_{n+1}, ..., w_{n+r}\}$, in which $w_n$ is a placeholder "[URL]" of the resource hyperlink, $l$ is the length of the sequence

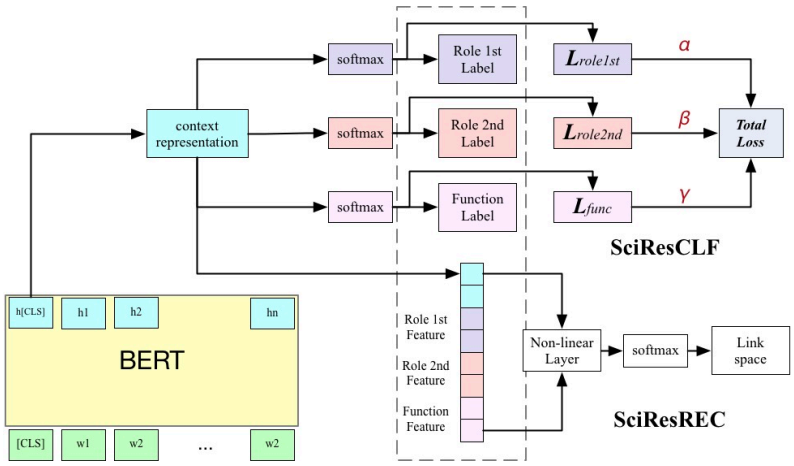


Figure 3: The overall architecture of the SciResCLF and the SciResREC.

left to the hyperlink and $r$ is the length of the sequence right to the hyperlink. To eliminate the information of the original resource citation, we mask the resource names mentioned in the resource context. So that the resource hyperlink can only be determined by the other words related to the target resource from the context query. The output is a ranking list, which contains the top N possible predictions from the resource hyperlink space.

For this task, we develop a framework, called **SciResREC** to predict the resource hyperlinks by learning information of the resource contexts and benefiting from the classification results of the SciResCLF. Figure 3 also shows a high-level overview of the SciResREC framework. For each input resource context sequence, we first apply the SciResCLF to respectively get the output results of the 1-st category role classifier, the 2-nd category role classifier and the function classifier. To incorporate the role and function information, the three classification labels are used as features for predicting the resource hyperlinks. In SciResREC, we use the same pre-trained BERT encoder to learn context representations. Then the hidden output $h_{[CLS]}$ is concatenated with the three label representations, which is passed to a non-linear layer with the ReLU activation function. Hence the recommendation task is transformed into a multi-class classification problem which maps the context feature vectors into the resource hyperlink space.

## 6 Experiments

### 6.1 Data and Metrics

To test the performance of SciResCLF, we do experiments on our SciRes dataset, which is split into 3 parts: 80% for training, 10% for testing and 10% for developing. To overcome the class imbalance, we utilize an up-sampling strategy by simply replicating the minority class samples. We respectively deploy the up-sampling strategy for three classification tasks to get the best model for each task. The strategy is only deployed on the training set while not on the testing and developing sets. Finally we get 2,988 data samples for training the 1st-category role classifier, 7,236 for the 2nd-category role classifier and 7,404 for the function classifier. The testing and developing sets are shared by the three tasks, both with 334 samples. The size of the dataset is relatively small to perform most neural-based methods, but we will show that our model can achieve good performance with even limited labeled data. For evaluation, we report the micro-F1 score and the macro-F1 score across the role and function types.

To build the training set for SciResREC, we first collect all the resource contexts and the citation hyperlinks from the ARC dataset (including articles up to December 2015). We select the top 100 most frequent resources and add their hyperlinks into the hyperlink space. For testing, we extract resource contexts from the publications of ACL2016, ACL2017, EMNLP2016 and EMNLP2017 and select the ones of which the hyperlink exists in the space. For each context, either the in-line resource citation or the additional resource citation is replaced by a "[URL]"

| Methods | 1st-category Roles | | 2nd-category Roles | | Functions | |
|---|---|---|---|---|---|---|
| | F1-micro | F1-macro | F1-micro | F1-macro | F1-micro | F1-macro |
| AvrgEmbed+LR | 0.733 | 0.732 | 0.453 | 0.462 | 0.534 | 0.475 |
| AvrgEmbed+SVM | 0.691 | 0.690 | 0.429 | 0.447 | 0.550 | 0.466 |
| FastText | 0.694 | 0.692 | 0.489 | 0.472 | 0.638 | 0.471 |
| CNN | 0.649 | 0.647 | 0.450 | 0.313 | 0.618 | 0.498 |
| RCNN | 0.697 | 0.697 | 0.430 | 0.407 | 0.621 | 0.517 |
| LSTM | 0.607 | 0.605 | 0.413 | 0.393 | 0.508 | 0.399 |
| LSTM+AT | 0.706 | 0.705 | 0.502 | 0.483 | 0.715 | 0.551 |
| LSTM+AT *multi-task* | 0.735 | 0.734 | 0.532 | 0.540 | 0.754 | 0.577 |
| SciResCLF *single-task* | 0.772 | 0.772 | 0.672 | 0.685 | 0.795 | 0.640 |
| + *joint 1st role* | - | - | 0.666 | 0.667 | **0.823** | 0.668 |
| + *joint 2nd role* | 0.759 | 0.760 | - | - | 0.805 | 0.675 |
| + *joint function* | 0.778 | 0.778 | 0.673 | 0.671 | - | - |
| **SciResCLF *multi-task*** | **0.784** | **0.783** | **0.706** | **0.709** | 0.817 | **0.715** |

Table 4: Comparison results of SciResCLF for the three classification tasks.

placeholder. And the words of resource name are masked with a "[MASK]" token. Finally we get a training set of 2,910 samples and a testing set of 235 samples. To evaluate the predicted ranking list, we report the Precision@Top3 and the MAP metrics.

### 6.2 Baselines and Setups

For the classification task, we compare the SciResCLF with widely used context classification approaches on the SciRes: **Average Embedding + LR/SVM**, two machine learning algorithms, logistic regression (LR) and SVM, using the average of word embeddings as input; **FastText**, an implementation for FastText of Joulin et al. (2017); **CNN**, an implementation for TextCNN of Kim (2014); **RCNN**, an implementation for a recurrent convolutional neural network of Lai et al. (2015); **LSTM**, a 3-layer structure with input word embeddings, a bidirectional LSTM layer and a softmax output layer; **LSTM+AT**, using the attention mechanism to LSTM; and **LSTM+AT *multi-task***, using LSTM+AT to jointly learn the three tasks.

For the recommendation task, we compare our SciResREC with the **Random Forest (RF) classifier**, which is robust to overfitting even with large numbers of features. For the RF classifier, we use two types of features: **BoW+TFIDF**, the 20,000 most frequent words from the training set are selected and the TFIDF of each word is used as features; **N-grams+TFIDF**, the TFIDF of the most frequent 20,000 N-grams (up to 5-grams).

Our BERT encoder is based on Googles reference implementation[10] (TensorFlow 1.12.0). For training, we begin with the BERT-Base model (uncased, 12-layer, 768-hidden, 12-heads, 110M parameters) and then fine-tune the model on our training sets for the three classification tasks. The maximum word sequence length is 128; the learning rate is set to 5e-5 and all other defaults settings are used. For the recommendation model, the feature embedding dimension of role labels and function labels is 32, which are initialized at random and the hidden size is set to 64. For all the baselines, we use the word embeddings pretrained on our large-scale scientific corpus, which is a full-text collection of 40 thousand research papers. And we stop the training when we found the best result in the developing set.

[10] https://github.com/google-research/bert

| 1st Roles | F1 | 2nd Roles | F1 | Functions | F1 |
|---|---|---|---|---|---|
| Material | 0.787 | Paper | 0.902 | Use | 0.870 |
| Method | 0.793 | License | 0.870 | Produce | 0.861 |
| Supplement | 0.769 | Media | 0.846 | Introduce | 0.824 |
| | | Data | 0.765 | Other | 0.716 |
| | | Code | 0.742 | Compare | 0.667 |
| | | Tool | 0.680 | Extend | 0.353 |
| | | Document | 0.592 | | |
| | | Algorithm | 0.539 | | |
| | | Website | 0.448 | | |

Table 5: Results (F1-score) on each category predicted by the best model for each of the three tasks.

### 6.3 Results and Discussions

**Classification Results**

Our model achieves the best results on all the three classification tasks, as shown in Table 4. And the F1-score for each role or function type is shown in Table 5. From the results we can see that there is still a large gap with other general context-based classification tasks, which provides considerable potential for advancement in the future research. For more in-depth performance

| Methods | Precision@Top3 | MAP |
|---|---|---|
| RF *(BoW+TFIDF)* | 0.438 | 0.275 |
| RF *(N-grams+TFIDF)* | 0.449 | 0.306 |
| **SciResREC** | **0.489** | **0.597** |
| *-Function feature* | 0.471 | 0.569 |
| *-Role 2nd feature* | 0.420 | 0.539 |
| *-Role 1st feature* | 0.399 | 0.497 |

Table 6: The Precision@Top3 and the MAP results for the ranking list predicted by SciResREC.

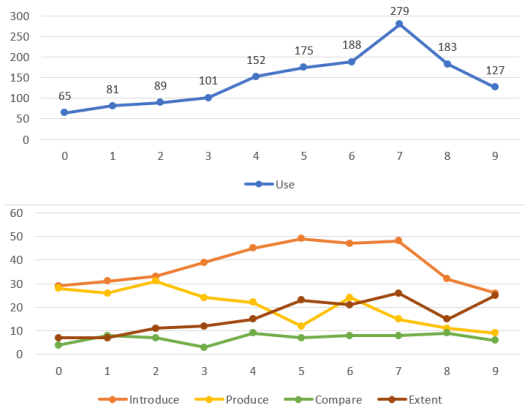


Figure 4: The revolution of resource functions in ten years starting from the first appearance.

analysis, we note that most neural-based models are inferior to the traditional machine learning methods, which is perhaps due to the limited labeled datasets. While our model benefiting from the BERT encoder, which is pre-trained on vast amounts of text and fine-tuned on our task-specific, show an effective solution to the data limitation. We also see that the LSTM performs better when the attention layer introduced. It indicates the attention mechanism, which is sufficient used in the transformer structure of BERT, is significant to learn which words of a sequence are more important for determining the labels in our classification tasks. And the position embeddings in BERT is also effective to learn the relative positions between the resource citation and other words. By comparing the results between different joint models, we can get some interesting findings about the relationships between the resource roles and the resource functions. First, the general role and the fine-grained role can not benefit each other, which is perhaps because the high-level constraints can reduce the inter-class errors but meanwhile introduce more intra-class errors. When introducing the function information, the results for the general role will improve while the fine-grained role will slightly drop off. Moreover, jointly learning the two-level role information can observably enhance the results for function classification. And considering the complex interaction among the three tasks, the multi-task model achieves the best performance.

The resulting function classifier is sufficient to be applied to the entire ARC dataset. Nonetheless, errors remain. Consider the following example:

*To give their network a better initialization, they learn word embeddings ... They **released** their 50-dimensional word embeddings (vocabulary size 130K) under the name SENNA. <CITE>*,

which is notably a review of others' work and the correct function is **Introduce**. While our model mistakes it to **Produce** caused by a higher attention score of the word "released". This case inspires us that the dependency parsing can also be considered in the future research.

### Recommendation Results

As Table 6 shows, our SciResREC framework outperforms the two baselines. An ablation test suggests that each feature component of our model contributes to the final performance, which indicates the information of role and function are helpful for understanding the scientific resources. And we can observe that the feature of 2-nd category role label has the largest impact on performance indicating that capturing fine-grained role types is important for recognizing specific resources.

### Resource Evolution Analysis

To study what the resource function can tell us about the scientific resource development, we apply our function classifier trained on SciRes to the large ARC dataset. We select the top 100 most frequent resources and filter out the ones which have been existing less than 10 years. Finally we obtain a set of 33 resources and use our trained classifier to identity the function of each resource citation in its context. A statistical result is shown in Figure 4. The horizontal axis represents the number of years after the resource first appeared in the scientific corpus. From the figure we can see that a resource will drive to its maturity stage and be widely used in 4-5 years after it first exists. And in 8-9 years it will gradually be out of date and replaced by other new technologies. Moreover, citing the resources as background and making extensions based on the resources progressively increase along with time, which is consistent with the general expect.

## 7 Conclusion and Future Work

We introduce a novel task of *modeling the role and function for on-line resource citations in scientific literature*. For this task we first create an annotation scheme, collect a large-scale corpus and construct a manually labeled dataset. And we propose a multi-task model to jointly classify the role and function types based on the resource contexts. By incorporating the associated information, our SciResCLF framework effectively improves the performance across all tasks. Moreover, we propose a resource recommendation task and develop the SciResREC framework using the predicted labels as features. Our frameworks can contribute to build more powerful search and recommendation systems for scientific on-line resources. For future work, we will try using pre-trained BERT for scientific domain such as the SciBERT (Beltagy et al., 2019) and explore using our frameworks to help more tasks such as the evaluation, prediction and knowledge graph construction for scientific resources.

## 8 Acknowledgements

This work was supported by the National Key R&D Program of China (No. 2017YFB1002101), the Joint Advanced Research Foundation of China Electronics Technology Group Corporation (CETC) (No.6141B08010102), and National Natural Science Foundation of China (No. U1636203, No. 61602490).

## References

Ashutosh Adhikari, Achyudh Ram, Raphael Tang, and Jimmy Lin. 2019. Docbert: Bert for document classification. *CoRR*, abs/1904.08398.

Isabelle Augenstein, Mrinal Das, Sebastian Riedel, Lakshmi Vikraman, and Andrew McCallum. 2017. Semeval 2017 task 10: Scienceie - extracting keyphrases and relations from scientific publications. In *SemEval@ACL*.

Iz Beltagy, Arman Cohan, and Kyle Lo. 2019. Scibert: Pretrained contextualized embeddings for scientific text. *ArXiv*, abs/1903.10676.

Guillermo de la Calle, Miguel García-Remesal, Stefano Chiesa, Diana de la Iglesia, and Victor Maojo. 2009. Biri: a new approach for automatically discovering and indexing available public bioinformatics resources from the literature. *BMC Bioinformatics*, 10:320 – 320.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Geraint Duck, Goran Nenadic, Andy Brass, David L. Robertson, and Robert Stevens. 2012. bionerds: exploring bioinformatics database and software use through literature mining. In *BMC Bioinformatics*.

Geraint Duck, Goran Nenadic, Andy Brass, David L. Robertson, and Robert Stevens. 2014. Extracting patterns of database and software usage from the bioinformatics literature. In *Bioinformatics*.

Geraint Duck, Goran Nenadic, Michele Filannino, Andy Brass, David L. Robertson, and Robert Stevens. 2016. A survey of bioinformatics database and software usage through mining the literature. In *PloS one*.

Qi He, Daniel Kifer, Jian Pei, Prasenjit Mitra, and C. Lee Giles. 2011. Citation recommendation without author supervision. In *WSDM*.

Qi He, Jian Pei, Daniel Kifer, Prasenjit Mitra, and C. Lee Giles. 2010. Context-aware citation recommendation. In *WWW*.

Wenyi Huang, Saurabh Kataria, Cornelia Caragea, Prasenjit Mitra, C. Lee Giles, and Lior Rokach. 2012. Recommending citations: translating papers into references. In *CIKM*.

Wenyi Huang, Zhaohui Wu, Liang Chen, Prasenjit Mitra, and C. Lee Giles. 2015. A neural probabilistic model for context based citation recommendation. In *AAAI*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *EACL*.

David Jurgens, Srijan Kumar, Raine Hoover, Daniel A. McFarland, and Daniel Jurafsky. 2018. Measuring the evolution of a scientific field through citation frames. *TACL*, 6:391–406.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *AAAI*.

Maria Liakata, Shyamasree Saha, Simon Dobnik, Colin R. Batchelor, and Dietrich Rebholz-Schuhmann. 2012. Automatic recognition of conceptualization zones in scientific articles and two life science applications. In *Bioinformatics*.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *EMNLP*.

Yi Luan, Mari Ostendorf, and Hannaneh Hajishirzi. 2017. Scientific information extraction with semi-supervised neural tagging. In *EMNLP*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations.

Alec Radford. 2018. Improving language understanding by generative pre-training.

Chi Sun, Luyao Huang, and Xipeng Qiu. 2019. Utilizing bert for aspect-based sentiment analysis via constructing auxiliary sentence. *CoRR*, abs/1903.09588.

Jie Tang and Jing Zhang. 2009. A discriminative approach to topic-based citation recommendation. In *PAKDD*.

Simone Teufel, Advaith Siddharthan, and Tidhar Dan. 2006. Automatic classification of citation function. In *Proc. 2006 Conference on Empirical Methods in Natural Language Processing, Sydney, Australia*, pages 103–110.

Hu Xu, Bing Liu, Lei Shu, and Philip S. Yu. 2019. Bert post-training for review reading comprehension and aspect-based sentiment analysis. *CoRR*, abs/1904.02232.

Yasunori Yamamoto and Toshihisa Takagi. 2007. Orefil: an online resource finder for life sciences. *BMC Bioinformatics*, 8:287 – 287.

He Zhao, Zhunchen Luo, Chong Feng, and Yuming Ye. 2019. A context-based framework for resource citation classification in scientific literatures. In *SIGIR*.

# Adversarial Reprogramming of Text Classification Neural Networks

**Paarth Neekhara**[1], **Shehzeen Hussain**[2], **Shlomo Dubnov**[1,3], **Farinaz Koushanfar**[2]
[1]Department of Computer Science
[2]Department of Electrical and Computer Engineering
[3]Department of Music
University of California San Diego
{pneekhar,ssh028}@ucsd.edu

## Abstract

In this work, we develop methods to repurpose text classification neural networks for alternate tasks without modifying the network architecture or parameters. We propose a context based vocabulary remapping method that performs a computationally inexpensive input transformation to reprogram a victim classification model for a new set of sequences. We propose algorithms for training such an input transformation in both white box and black box settings where the adversary may or may not have access to the victim model's architecture and parameters. We demonstrate the application of our model and the vulnerability of neural networks by adversarially repurposing various text-classification models including LSTM, bi-directional LSTM and CNN for alternate classification tasks.

## 1 Introduction

While neural network based machine learning models serve as the backbone of many text and image processing systems, recent studies have shown that they are vulnerable to adversarial examples. Traditionally, an adversarial example is a sample from the classifier's input domain which has been perturbed in such a way that is intended to cause a machine learning model to misclassify it. While the perturbation is usually imperceptible, such an adversarial input results in the neural network model outputting an incorrect class label with higher confidence. Several studies have shown such adversarial attacks to be successful in both the continuous input domain (Szegedy et al., 2014; Papernot et al., 2016a, 2017, 2016b; Brown et al., 2017; Goodfellow et al., 2015) and discrete input spaces (Papernot et al., 2016c; Hu and Tan, 2018; Yang et al., 2018).

Adversarial Reprogramming (Elsayed et al., 2019) is a new class of adversarial attacks where a machine learning model is repurposed to perform a new task chosen by the attacker. The proposed attack is interesting because it allows an adversary to move a step beyond mere mis-classification of a victim network's output onto having the control to repurpose the network fully. The authors demonstrated how an adversary may repurpose a pre-trained ImageNet (Deng et al., 2009) model for an alternate classification task like classification of MNIST digits or CIFAR-10 images without modifying the network parameters. Since machine learning agents can be reprogrammed to perform unwanted actions as desired by the adversary, such an attack can lead to theft of computational resources such as cloud-hosted machine learning models. Besides theft of computational resources, the adversary may perform a task that violates the code of ethics of the system provider.

The adversarial reprogramming approach proposed by (Elsayed et al., 2019) trains an additive contribution $\theta$ to the inputs of the neural network to repurpose it for the desired alternate task. This approach assumes a white-box attack scenario where the adversary has full access to the network's parameters. Also, the adversarial program proposed in this work is only applicable to tasks where the input space of the the original and adversarial task is continuous.

In our work, we propose a method to adversarially *repurpose* neural networks which operate on sequences from a *discrete* input space. The task is to learn a simple transformation (adversarial program) from the input space of the adversarial task to the input space of the neural network such that the neural network can be repurposed for the adversarial task. We propose a context-based vocabulary remapping function as an adversarial program for sequence classification networks. We propose training procedures for this adversarial program in both white-box and black-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5216–5225,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

box scenarios. In the white-box attack scenario, where the adversary has access to the classifier's parameters, a Gumbel-Softmax trick (Jang et al., 2016) is used to train the adversarial program. Assuming a black-box attack scenario, where the adversary may not have access to the classifier's parameters, we present a REINFORCE (Williams, 1992) based optimization algorithm to train the adversarial program.

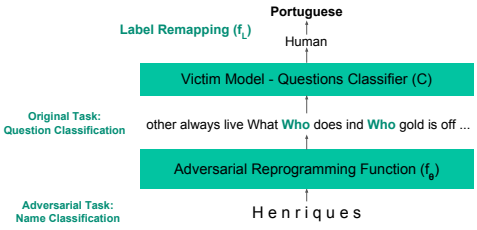


Figure 1: Example of Adversarial Reprogramming for Sequence Classification. We aim to design and train the adversarial reprogramming function $f_\theta$, such that it can be used to repurpose a pre-trained classifier C, for a desired adversarial task.

We apply our proposed methodology on various text classification models including Recurrent Neural Networks such as LSTMs and bidirectional LSTMs, and Convolutional Neural Networks (CNNs). We demonstrate experimentally, how these neural networks trained on a particular (original) text classification task can be repurposed for alternate (adversarial) classification tasks. We experiment with different text classification datasets given in table 1 as candidate original and adversarial tasks and adversarially reprogram the aforementioned text classification models to study the robustness of the attack.

## 2 Background and Related Work

### 2.1 Adversarial Examples

Traditionally, adversarial examples are intentionally designed inputs to a machine learning model that cause the model to make a mistake (Goodfellow et al., 2015). These attacks can be broadly classified into *untargeted* and *targeted* attacks. In the untargeted attack scenario, the adversary succeeds as long as the victim model classifies the adversarial input into *any* class other than the correct class, while in the *targeted* attack scenario, the adversary succeeds only if the model classifies the adversarial input into a specific incorrect class. In both these scenarios, the intent of the adversary is usually malicious and the outcome of the victim model is still limited to the original task being performed by the model.

Adversarial attacks of image-classification models often use gradient descent on an image to create a small perturbation that causes the machine learning model to mis-classify it (Szegedy et al., 2014; Biggio et al., 2013). There has been a similar line of adversarial attacks on neural networks with discrete input domains (Papernot et al., 2016c; Yang et al., 2018), where the adversary modifies a few tokens in the input sequence to cause mis-classification by a sequence model. In addition, efforts have been made in designing more general adversarial attacks in which the same modification can be applied to many different inputs to generate adversarial examples (Brown et al., 2017; Goodfellow et al., 2015; Moosavi-Dezfooli et al., 2017). For example, authors (Baluja and Fischer, 2018) trained an Adversarial Transformation Network that can be applied to all inputs to generate adversarial examples targeting a victim model or a set of victim models. In this work, we aim to learn such universal transformations of discrete sequences for a fundamentally different task: *Adversarial Reprogramming* described below.

### 2.2 Adversarial Reprogramming

Adversarial Reprogramming (Elsayed et al., 2019) introduced a new class of adversarial attacks where the adversary wishes to repurpose an existing neural network for a new task chosen by the attacker, without the attacker needing to compute the specific desired output. The adversary achieves this by first defining a hard-coded one-to-one label remapping function $h_g$ that maps the output labels of the adversarial task to the label space of the classifier $f$; and learning a corresponding adversarial reprogramming function $h_f(.;\theta)$ that transforms an input $(\tilde{X})$ [1] from the input space of the new task to the input space of the classifier. The authors proposed an adversarial reprogramming function $h_f(.;\theta)$, for repurposing ImageNet models for adversarial classification tasks. An adversarial example $X_{adv}$ for an input image $\tilde{X}$ can be generated using the following adversarial program: [2]

$$X_{adv} = h_f(\tilde{X};\theta) = \tilde{X} + \tanh(\theta)$$

[1] $\tilde{X}$ is an ImageNet size $(n \times n \times 3)$ padded input image

[2] Masking ignored because it is only a visualization convenience

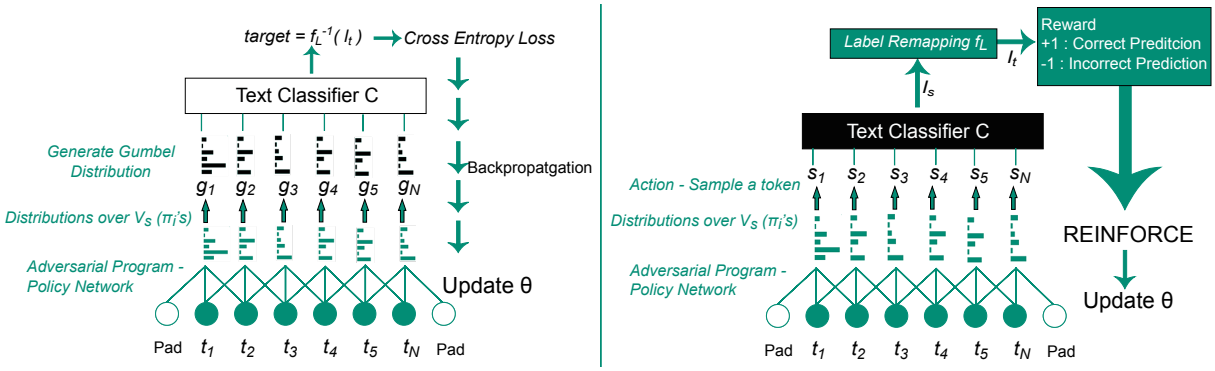


Figure 2: **Adversarial Reprogramming Function and Training Procedures. Left:** *White-box* Adversarial Reprogramming. The adversary generates gumbel distributions $g_i$ at each time-step which are passed as a soft version of one-hot vectors to the classifier C. The cross-entropy loss between the predictions and the mapped class is backpropagated to train the adversarial program $\theta$. **Right:** *Black-box* Adversarial Reprogramming. The adversarial reprogramming function is used as a policy network and the sampled action (sequence $s$) is passed to the classifier $C$ to get a reward based on prediction correctness. The adversarial program is then trained using REINFORCE.

where $\theta \in \mathbb{R}^{n \times n \times 3}$ is the learnable weight matrix of the adversarial program (where n is the ImageNet image width). Let $P(y|X)$ denote the probability of the victim model predicting label $y$ for an input $X$. The goal of the adversary is to maximize the probability $P(h_g(y_{adv})|X_{adv})$ where $y_{adv}$ is the label of the adversarial input $X_{adv}$. The following optimization problem that maximizes the log-likelihood of predictions for the adversarial classification task, can be solved using backpropagation to train the adversarial program parameterized by $\theta$:

$$\hat{\theta} = argmin_\theta \left( -\log P(h_g(y_{adv})|X_{adv}) + \lambda ||\theta||_2^2 \right) \quad (1)$$

where $\lambda$ is the regularization hyperparameter. Since the adversarial program proposed is a trainable additive contribution $\theta$ to the inputs, it's application is limited to neural networks with a continuous input space. Also, since the the above optimization problem is solved by backpropagating through the victim network, it assumes a white-box attack scenario where the adversary has gained access to the victim model's parameters. In this work, we describe how we can learn a simple transformation in the *discrete space* to extend the application of adversarial reprogramming on *sequence classification* problems. We also propose a training algorithm in the black-box setting where the adversary may not have access to the model parameters.

### 2.3 Transfer Learning

Transfer Learning (Raina et al., 2007) is a study closely related to Adversarial Reprogramming. During training, neural networks learn representations that are generic and can be useful for many related tasks. A pre-trained neural network can be effectively used as a feature extractor and the parameters of just the last few layers are retrained to realign the output layer of the neural network for the new task. Prior works have also applied transfer learning on text classification tasks (Do and Ng, 2006; Semwal et al., 2018). While transfer learning approaches exploit the learnt representations for the new task, they cannot be used to repurpose an exposed neural network for a new task without modifying some intermediate layers and parameters.

Adversarial Reprogramming (2019) studied whether it is possible to keep all the parameters of the neural network unchanged and simply learn an input transformation that can realign the outputs of the neural network for the new task. This makes it possible to repurpose *exposed* neural network models like cloud-based photo services to a new task where transfer learning is not applicable since we do not have access to intermediate layer outputs.

## 3 Methodology

### 3.1 Threat Model

Consider a sequence classifier $C$ trained on the original task of mapping a sequence $s \in S$ to a class label $l_S \in L_S$ i.e $C : s \mapsto l_S$. An adversary wishes to repurpose the original classifier $C$ for the adversarial task $C'$ of mapping a sequence $t \in T$ to a class label $l_T \in L_T$ i.e $C' : t \mapsto l_T$. The adversary can achieve this by hard-coding a

one-to-one label remapping function:

$$f_L : l_S \mapsto l_T$$

that maps an original task label to the new task label and learning a corresponding adversarial reprogramming function:

$$f_\theta : t \mapsto s$$

that transforms an input from the input space of the adversarial task to the input space of the original task. The adversary aims to update the parameters $\theta$ of the adversarial program $f_\theta$ such that the mapping $f_L(C(f_\theta(t)))$ can perform the adversarial classification task $C' : t \mapsto l_T$. Note that the victim model's parameters will be kept frozen while training the reprogramming function $f_\theta$.

### 3.2 Adversarial Reprogramming Function

The goal of the adversarial reprogramming function $f_\theta : t \mapsto s$ is to map a sequence $t$ to $s$ such that it is labeled correctly by the classifier $f_L(C)$.

The tokens in the sequence $s$ and $t$ belong to some vocabulary lists $V_S$ and $V_T$ respectively. We can represent the sequence $s$ as $s = s_1, s_2, .., s_N$ where $s_i$ is the vocabulary index of the $i_{th}$ token in sequence $s$ in the vocabulary list $V_S$. Similarly sequence $t$ can be represented as $t = t_1, t_2, .., t_N$ where $t_i$ is the vocabulary index of the $i_{th}$ token of sequence $t$ in the vocabulary list $V_T$.

In the simplest scenario, the adversary may try to learn a vocabulary mapping from $V_T$ to $V_S$ using which each $t_i$ can be independently mapped to some $s_i$ to generate the adversarial sequence. Such an adversarial program has limited potential since the representational capacity of such a reprogramming function is very limited. We experimentally support this hypothesis by showing how such a transformation has limited potential for the purpose of adversarial reprogramming.

A more sophisticated adversarial program can be a sequence to sequence machine translation model (Sutskever et al., 2014) that learns a translation $t \mapsto s$ for adversarial reprogramming. While theoretically this is a good choice, it defeats the purpose of adversarial reprogramming. This is because the computational complexity of training and using such a machine translation model would be similar if not greater than that of a new sequence classifier for the adversarial task $C'$.

The adversarial reprogramming function should be computationally inexpensive but powerful enough for adversarial repurposing. To this end, we propose a context-based vocabulary remapping model that produces a distribution over the target vocabulary at each time-step based on the surrounding input tokens. More specifically, we define our adversarial program as a trainable 3-d matrix $\theta_{k \times |V_T| \times |V_S|}$ where k is the context size. Using this, we generate a probability distribution $\pi_i$ over the vocabulary $V_S$ at each time-step $i$ as follows:

$$h_i = \sum_{j=0}^{k-1} \theta[j, t_{i+\lfloor k/2 \rfloor - j}] \tag{2}$$

$$\pi_i = softmax(h_i) \tag{3}$$

Both $h_i$ and $\pi_i$ are vectors of length $|V_S|$. To generate the adversarial sequence $s$ we sample each $s_i$ independently from the distribution $\pi_i$ i.e $s_i \sim \pi_i$

In practice, during training, we implement this adversarial program as a single layer of 1-d convolution over the sequence of one-hot encoded vectors of adversarial tokens $t_i$ with $|V_T|$ input channels and $|V_S|$ output channels with $k$-length kernels parameterized by $\theta_{k \times |V_T| \times |V_S|}$.

### 3.3 White-box Attack

In the white-box attack scenario, we assume that the adversary has gained access to the victim network's parameters and architecture. To train the adversarial reprogramming function $f_\theta$, we use an optimization objective similar to equation 1. Let $P(l|s)$ denote the probability of predicting label $l$ for a sequence $s$ by classifier $C$. We wish to maximize the probability $P(f_L^{-1}(l_t)|f_\theta(t))$ which is the probability of the output label of the classifier being mapped to the correct class $l_t$ for an input $t$ in the domain of the adversarial task. Therefore we need to solve the following log-likelihood maximization problem:

$$\hat{\theta} = argmin_\theta(-\sum_t log(P(f_L^{-1}(l_t)|f_\theta(t)))) \tag{4}$$

Note that that the output of the adversarial program $s = f_\theta(t)$ is a sequence of discrete tokens. This makes the above optimization problem non-differentiable. Prior works (Kusner and Hernández-Lobato, 2016; Gu et al., 2017; Yang et al., 2018) have demonstrated how we can smoothen such an optimization problem using the Gumbel-Softmax (Jang et al., 2016) distribution.

In order to backpropagate the gradient information from the classifier to the adversarial program, we smoothen the generated tokens $s_i$ using Gumbel-Softmax trick as per the following:

For an input sequence $t$, we generate a sequence of Gumbel distributions $g = g_1, g_2, .., g_N$. The $n_{th}$ component of distribution $g_i$ is generated as follows:

$$g_i^n = \frac{\exp((\log(\pi_i^n) + r_n)/temp)}{\sum_j \exp((\log(\pi_i^j) + r_j)/temp)}$$

where $\pi_i$ is the softmax distribution at the $i_{th}$ time-step obtained using equation 3, $r_n$ is a random number sampled from the Gumbel distribution (Gumbel, 1954) and $temp$ is the temperature of Gumbel-Softmax.

The sequence then passed to the classifier $C$ is the sequence $g$ which serves as a soft version of the one-hot encoded vectors of $s_i$'s. Since the model is now differentiable, we can solve the following optimization problem using backpropagation:

$$\hat{\theta} = argmin_\theta(-\sum_t log(P(f_L^{-1}(l_t)|g)))$$

During training the temperature parameter is annealed from some high value $t_{max}$ to a very low value $t_{min}$. The details of this annealing process for our experiments have been included in the appendix.

### 3.4 Black-box Attack

In the black-box attack scenario, the adversary can only query the victim classifier $C$ for labels (and not the class probabilities). We assume that the adversary has the knowledge of the vocabulary $V_S$ of the victim model. Since the adversarial program needs to produce a discrete output to feed as input to the classifier $C$, it is not possible to pass the gradient update from the classifier $f_L(C)$ to the adversarial program $\theta$ using standard back-propagation. Also, in the black-box attack setting it is not possible to back-propagate the cross entropy loss through the classifier $C$ in the first place.

We formulate the sequence generation problem as a Reinforcement Learning problem (Bachman and Precup, 2015; Bahdanau et al., 2016; Yu et al., 2016) where the adversarial reprogramming function is the policy network. The state of the adversarial program is a sequence $t \in T$ and an action of our policy network is to produce a sequence of tokens $s \in S$. The adversarial program parameterized by $\theta$, models the stochastic policy $\pi_{adv}(s|t;\theta)$ such that a sequence $s \in S$ may be sampled from this policy conditioned on $t \in T$. We use a simple reward function where we assign a reward +1 for a correct prediction and -1 for an incorrect prediction using the classifier $f_L(C)$ where $f_L$ is the label remapping function and $C$ is the classifier. Formally:

$$r(t,s) = \begin{cases} +1, & f_L(C(s)) = l_t \\ -1, & f_L(C(s)) \neq l_t \end{cases}$$

The optimization objective to train the policy network is the following:

$$\max_\theta J(\theta) \quad \text{where,} \quad J(\theta) = \mathbb{E}_{\pi_{adv}}[r(t,s)]$$

Following the REINFORCE algorithm (Williams, 1992) we can write the gradient of the expectation with respect to $\theta$ as per the following:

$$\begin{aligned}
\nabla_\theta J &= \nabla_\theta \left[ \mathop{\mathbb{E}}_{\pi_{adv}} [r(t,s)] \right] \\
&= \nabla_\theta \left[ \sum_s \pi_{adv}(s|t;\theta) r(t,s) \right] \\
&= \sum_s \pi_{adv}(s|t;\theta) \nabla_\theta \log(\pi_{adv}(s|t;\theta)) r(t,s) \\
&= \mathop{\mathbb{E}}_{\pi_{adv}} [r(t,s) \nabla_\theta \log(\pi_{adv}(s|t;\theta))] \\
&= \mathop{\mathbb{E}}_{\pi_{adv}} [r(t,s) \nabla_\theta \log(\pi_{adv}(s_1, .., s_N|t;\theta))] \\
&= \mathop{\mathbb{E}}_{\pi_{adv}} \left[ r(t,s) \nabla_\theta \log(\prod_i \pi_{adv}(s_i|t;\theta)) \right] \\
&= \mathop{\mathbb{E}}_{\pi_{adv}} \left[ r(t,s) \sum_i \nabla_\theta \log(\pi_{adv}(s_i|t;\theta)) \right]
\end{aligned}$$

Note that $\pi_{adv}(s_i|t;\theta)$ is the same as $\pi_i$ defined in equation 3 which can be differentiated with respect to $\theta$. The expectations are estimated as sample averages. Having obtained the gradient of expected reward, we can use mini-batch gradient ascent to update $\theta$ with a learning rate $\alpha$ as: $\theta \leftarrow \theta + \alpha \nabla_\theta J$.

## 4 Experiments

### 4.1 Datasets and Classifiers

We demonstrate the application of the proposed reprogramming techniques on various text-classification tasks. In our experiments, we design adversarial programs to attack both word-level and character-level text classifiers. Additionally, we

aim to adversarially repurpose a character-level text classifier for a word-level classification task and vice-versa. To this end, we choose the following text-classification datasets as candidates for the original and adversarial classification tasks:

***1. Surname Classification Dataset*** **(Names-18, Names-5) (Robertson, 2017)**: The dataset categorizes surnames from 18 languages of origin. We use a subset of this dataset *Names-5* containing Names from 5 classes: *Dutch, Scottish, Polish, Korean* and *Portuguese*, as a candidate for adversarial task in the experiments.

***2. Experimental Data for Question Classification*** **(Questions) (Li and Roth, 2002)**: categorizes around 5500 questions into 6 classes: *Abbreviation, Entity, Description, Human, Location, Numeric*.

***3. Sentiment Classification Dataset (Abdulla et al., 2013)*** **(Arabic)** : contains 2000 binary labeled tweets on diverse topics such as politics and arts.

***4. Large Movie Review Dataset*** **(IMDB) for sentiment classification (Maas et al., 2011)**: contains 50,000 movie reviews categorized into binary class of positive and negative sentiment.

The statistics of the above mentioned datasets have been given in table 1. We train adversarial reprogramming functions to repurpose various text-classifiers based on Long Short-Term Memory (LSTM) network (Hochreiter and Schmidhuber, 1997), bidirectional LSTM network (Graves et al., 2005) and Convolutional neural network (Kim, 2014) models. All the aforementioned models can be trained for both word-level and character-level classification. We use character level classifiers for *Names-18* and *Names-5* datasets and word-level classifiers for *IMDB*, *Questions* and *Arabic* datasets. We use randomly initialized word/character embeddings which are trained along with the classification model parameters. For LSTM, we use the output at last timestep for prediction. For the Bi-LSTM, we combine the outputs of the first and last time step for prediction. For the Convolutional Neural Network we follow the same architecture as (Kim, 2014). Additionally, to understand the effectiveness of adversarial reprogramming, we train a sigle layer CNN with convolutional filter width 5 and discuss the results of this experiment in Section 4.4. The details of these classifiers have been included in table 2 of the supplementary material.

| Data Set | # Classes | Train Samples | Test Samples | $\|V\|$ | Avg Length |
|---|---|---|---|---|---|
| Names-18 | 18 | 115,028 | 28,758 | 90 | 7.1 |
| Names-5 | 5 | 3632 | 909 | 66 | 6.5 |
| Questions | 6 | 4361 | 1091 | 1205 | 11.2 |
| Arabic | 2 | 1600 | 400 | 955 | 9.7 |
| IMDB | 2 | 25,000 | 25,000 | 10000 | 246.8 |

Table 1: Summary of datasets. $|V|$ denotes the vocabulary size of each dataset.

| Data Set | LSTM | Bi-LSTM | CNN | 1-CNN |
|---|---|---|---|---|
| Names-18 | 97.84 | 97.84 | **97.88** | 74.04 |
| Names-5 | **99.88** | **99.88** | 99.77 | 71.51 |
| Questions | 96.70 | **98.25** | 98.07 | 83.77 |
| Arabic | 87.25 | **88.75** | 88.00 | 74.75 |
| IMDB | 86.83 | 89.43 | **90.02** | 83.32 |

Table 2: Test accuracy of various classification models. We use character-level models for *Names-5* and *Names-18* and word-level models for all other tasks. 1-CNN is a single layer CNN model with filter width 5.

### 4.2 Experimental Setup

As described in the methodology section, the label remapping function $f_L$ we use, is a one-to-one mapping between the labels of the original task and the adversarial task. Therefore it is required to apply the constraint that the number of classes of the adversarial task are less than or equal to the number of classes of the original task. We choose *Names-5, Arabic* and *Question Classification* as candidates for the adversarial tasks and repurpose the models allowed under this constraint. We use context size $k = 5$ for all our experiments.

In white-box attacks, we use the Gumbel-Softmax based approach described in the methodology to train the adversarial program. The details of the temperature annealing process are included in table 1 of the supplementary material. For black-box attacks, we use the REINFORCE algorithm described in methodology, on mini-batches of sequences. Since the action space for certain reprogramming problems, (eg. reprogramming of IMDB classifier) is large ($|V_S| = 10000$), we restrict the output of the adversarial program to most frequent 1000 tokens in the vocabulary $V_S$. We use Adam optimizer (Kingma and Ba, 2015) for all our experiments. Hyperparameter details of all our experiments are included in table 1 of the supplementary material.

| Original Task | Adversarial Task | Victim Model | Test Accuracy (%) | | |
|---|---|---|---|---|---|
| | | | Black box | White Box | White Box on Random Network |
| Questions | Names-5 | LSTM | 80.96 | 97.03 | 44.33 |
| | | Bi-LSTM | **93.51** | **99.66** | 63.14 |
| | | CNN | 88.90 | 99.22 | 93.06 |
| Questions | Arabic | LSTM | 73.50 | 87.50 | 50.00 |
| | | Bi-LSTM | 81.75 | 83.50 | 70.00 |
| | | CNN | **82.25** | **87.25** | 76.25 |
| Names-18 | Questions | LSTM | 68.56 | 95.23 | 28.23 |
| | | Bi-LSTM | **94.96** | 97.15 | 80.01 |
| | | CNN | 71.03 | **97.61** | 33.45 |
| Names-18 | Arabic | LSTM | **83.00** | 84.75 | 51.50 |
| | | Bi-LSTM | 78.75 | 84.25 | 69.25 |
| | | CNN | 80.75 | **86.50** | 60.00 |
| IMDB | Arabic | LSTM | 80.75 | **88.25** | 50.50 |
| | | Bi-LSTM | 83.25 | 86.75 | 84.00 |
| | | CNN | **84.00** | 87.00 | 84.25 |

Table 3: **Adversarial Reprogramming Experiments:** The accuracies of *white-box* and *black-box* reprogramming experiments on different combinations of original task, adversarial task and model. Figures in bold correspond to our best results on a particular adversarial task in the given attack scenario (black-box and white-box). *White-box on Random Network* column presents results of the white-box attack on an untrained neural network. Context size $k = 5$ is used for all our experiments.

### 4.3 Results and Discussion

The accuracies of all adversarial reprogramming experiments have been reported in table 3. To interpret the results in context, the accuracies achieved by the LSTM, Bi-LSTM and CNN text classification models on the adversarial tasks can be found in table 1.

We demonstrate how character-level models trained on Names-18 dataset can be repurposed for word-level sequence classification tasks like Question Classification and Arabic Tweet Sentiment Classification. Similarly, word-level classifiers trained on Question Classification Dataset can be repurposed for the character-level Surname classification task. Interestingly, classifiers trained on IMDB Movie Review Dataset can be repurposed for Arabic Tweet Sentiment Classification even though there is a high difference between the vocabulary size (10000 vs 955) and average sequence length(246.8 vs 9.7) of the two tasks. It can be seen that all of the three classification models are susceptible to adversarial reprogramming in both white-box and black-box setting.

White-box based reprogramming outperforms the black-box based approach in all of our experiments. In practice, we find that training the adversarial program in the black-box scenario requires careful hyper-parameter tuning for REINFORCE to work. We believe that improved reinforcement learning techniques for sequence generation tasks (Bahdanau et al., 2016; Bachman and Precup, 2015) can make the training procedure for black-box attack more stable. We propose such improvement as a direction of future research.

To assess the importance of the original task on which the network was trained, we also present results of white-box adversarial reprogramming on untrained random network. Our results are coherent with similar experiments on adversarial reprogramming of untrained ImageNet models (Elsayed et al., 2019) demonstrating that adversarial reprogramming is less effective when it targets untrained networks. The figures in table 3 suggest that the representations learned by training a text classifier on an original task, are important for repurposing it for an alternate task. However another plausible reason as discussed by Elsayed et al. is that the reduced performance on random networks might be because of simpler reasons like poor scaling of network weight initialization making the optimization problem harder.

### 4.4 Complexity of Reprogramming Function

As motivated earlier in Section 3.2, computational efficiency of the adversarial program is critical for adversarial reprogramming to be of interest to an adversary. If the adversary can achieve the desired results using a computationally inexpensive classifier, it defeats the purpose of adversarial reprogramming. To understand if this is the case, we train a one-layer CNN with the same convolutional filter width as our adversarial program and average the activations across all time-steps to classify a given phrase. The results of such a classifier on various datasets have been reported in Table 2. We can observe that our white-box attack on pretrained networks, outperforms this classifier in all scenarios (refer to Table 3). Our best black-box attacks also outperform a one-layer CNN for all adversarial tasks. This experiment demonstrates that the reprogramming function exploits the learned feature representation of the victim model. Also, the observation that adversarial reprogramming is significantly less effective on randomly initialized

untrained networks further reinforces the importance of utilizing a trained victim model.

Since the reprogramming function is a context based vocabulary remapping function, we can implement it as a look-up table that maps a combination of $k$ tokens from the vocabulary $V_t$ to a token in the source vocabulary $V_s$. The time complexity for transforming a sentence $t$ to an adversarial sentence is just $O(length(t))$.

#### 4.4.1 Adversarial Sequences

Figure 3 shows some adversarial sequences generated by the adversarial program for Names-5 Classification while attacking a CNN trained on the Question Classification dataset. A sequence $t$ in the first column is transformed into the adversarial sequence $s$ in the second column by the trained adversarial reprogramming function. While these adversarial sequences may not make semantic or grammatical sense, it exploits the learned representation of the classifier to map the inputs to the desired class. For example, sequences that should be mapped to HUMAN class have words like *Who* in the generated adversarial sequence. Similarly, sequences that should be mapped to LOCATION class have words like *world, city* in the adversarial sequence. Other such interpretable transformations are depicted via colored text in the adversarial sequences of Figure 3.

| Adversarial Task Sequence (t) (Names-5) | Adversarial Program Output (s) (Question Classification) | Prediction by Classifier | Mapped Class | Actual Class |
|---|---|---|---|---|
| Ryoo | white sport substance animal All off .. | ENTITY | Korean | Korean |
| Houtum | player video exp abb What does off is off .. | ABBREVIATION | Dutch | Dutch |
| Winogrodzki | manner France manner video def oil def reason desc What do All off .. | DESCRIPTION | Polish | Polish |
| Murphy | world live exp city What university All is off .. | LOCATION | Scottish | Scottish |
| Paulissen | player stars along abb abb exp exp always abb What is off .. | ABBREVIATION | Dutch | Dutch |
| Henriques | other always live What Who does ind Who gold is off .. | HUMAN | Portuguese | Portuguese |
| Maly | world attend home abb home is off .. | LOCATION | Scottish | Polish |
| Kasprzak | does exp exp def manner does reason What does off .. | DESCRIPTION | Polish | Polish |
| Ferreiro | e-mail Who ind exp Who ind university university gold off .. | HUMAN | Portuguese | Portuguese |
| Hong | sport cremat substance university is off .. | ENTITY | Korean | Korean |

Figure 3: Adversarial sequences generated by our adversarial program for Names-5 Classification (adversarial task), when targeting a CNN trained on the Question Classification dataset (original task). Interpretable transformations are shown as colored words in the second column.

#### 4.4.2 Effect of Context Size

By varying the context size $k$ of the convolutional kernel $\theta_{k\times|V_T|\times|V_S|}$ in our adversarial program we are able to control the representational capacity of the adversarial reprogramming function. Figure 4 shows the percentage accuracy obtained when training the adversarial program with different context sizes $k$ on two different adversarial tasks: Arabic Tweets Classification and Name Classification. Using a context size $k = 1$ reduces the adversarial reprogramming function to simply a vocabulary remapping function from $V_S$ to $V_T$. It can be observed that the performance of the adversarial reprogramming model at $k = 1$ is significantly worse than that at higher values of $k$. While higher values of $k$ improve the performance of the adversarial program, they come at a cost of increased computational complexity and memory required for the adversarial reprogramming function. For the adversarial tasks studied in this paper, we observe that $k = 5$ is a reasonable choice for context size of the adversarial program.

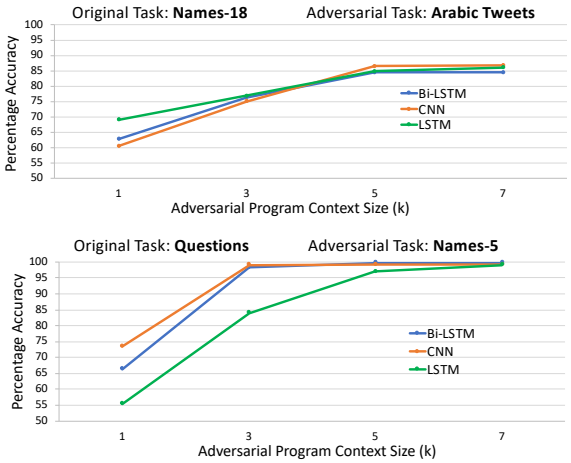


Figure 4: Accuracy vs Context size ($k$) plots for all 3 classification models on 2 different adversarial reprogramming experiments.

## 5 Conclusion

In this paper, we extend adversarial reprogramming, a new class of adversarial attacks, to target text classification neural networks. Our results demonstrate the effectiveness of such attacks in the more challenging black-box settings, posing them as a strong threat in real-world attack scenarios. We demonstrate that neural networks can be effectively reprogrammed for alternate tasks, which were not originally intended by a service provider. Our proposed end-to-end approach can be used to further understand the vulnerabilities and blind spots of deep neural network based text classification systems. We recommend future work to study the scope of adversarial reprogramming for other NLP applications such as machine translation, text to speech synthesis and text to im-

age synthesis where the input space is discrete. Furthermore, due to the threat presented by adversarial reprogramming, we recommend future work to study defenses against such attacks.

## References


N. A. Abdulla, N. A. Ahmed, M. A. Shehab, and M. Al-Ayyoub. 2013. Arabic sentiment analysis: Lexicon-based and corpus-based. In *2013 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT)*.

Philip Bachman and Doina Precup. 2015. Data generation as sequential decision making. In *Advances in Neural Information Processing Systems*, pages 3249–3257.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2016. An actor-critic algorithm for sequence prediction. *CoRR*, abs/1607.07086.

Shumeet Baluja and Ian Fischer. 2018. Learning to attack: Adversarial transformation networks. In *AAAI*.

Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. 2013. Evasion attacks against machine learning at test time. In *Machine Learning and Knowledge Discovery in Databases*.

Tom B. Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. 2017. Adversarial patch. *CoRR*, abs/1712.09665.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

Chuong B. Do and Andrew Y. Ng. 2006. Transfer learning for text classification. In Y. Weiss, B. Schölkopf, and J. C. Platt, editors, *Advances in Neural Information Processing Systems 18*. MIT Press.

Gamaleldin F. Elsayed, Ian J. Goodfellow, and Jascha Sohl-Dickstein. 2019. Adversarial reprogramming of neural networks. In *ICLR*.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*.

Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. 2005. Bidirectional lstm networks for improved phoneme classification and recognition. In *Proceedings of the 15th International Conference on Artificial Neural Networks: Formal Models and Their Applications - Volume Part II*, ICANN'05, pages 799–804, Berlin, Heidelberg. Springer-Verlag.

Jiatao Gu, Daniel Jiwoong Im, and Victor O. K. Li. 2017. Neural machine translation with gumbel-greedy decoding. *CoRR*, abs/1706.07518.

E.J. Gumbel. 1954. *Statistical theory of extreme values and some practical applications: a series of lectures*. Applied mathematics series. U. S. Govt. Print. Office.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8).

Weiwei Hu and Ying Tan. 2018. Black-box attacks against rnn based malware detection algorithms. In *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *CoRR*, abs/1611.01144.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In Proc. *ICLR*.

Matt J. Kusner and José Miguel Hernández-Lobato. 2016. GANS for sequences of discrete elements with the gumbel-softmax distribution. *CoRR*, abs/1611.04051.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, COLING '02, Stroudsburg, PA, USA. Association for Computational Linguistics.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Portland, Oregon, USA.

S. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard. 2017. Universal adversarial perturbations. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 86–94.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2017. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*. ACM.

Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. 2016a. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE.

Nicolas Papernot, Patrick D. McDaniel, and Ian J. Goodfellow. 2016b. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *CoRR*, abs/1605.07277.

Nicolas Papernot, Patrick D. McDaniel, Ananthram Swami, and Richard E. Harang. 2016c. Crafting adversarial input sequences for recurrent neural networks. *CoRR*, abs/1604.08275.

Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y. Ng. 2007. Self-taught learning: Transfer learning from unlabeled data. In *Proceedings of the Twenty-fourth International Conference on Machine Learning*.

Sean Robertson. 2017. Classifying names with a character-level rnn - pytroch tutorial. https://pytorch.org/tutorials/intermediate/char_rnn_classification_tutorial.html.

Tushar Semwal, Gaurav Mathur, Promod Yenigalla, and Shivashankar B. Nair. 2018. A practitioners' guide to transfer learning for text classification using convolutional neural networks. *CoRR*, abs/1801.06480.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *ICLR*.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4).

Puyudi Yang, Jianbo Chen, Cho-Jui Hsieh, Jane-Ling Wang, and Michael I. Jordan. 2018. Greedy attack and gumbel attack: Generating adversarial examples for discrete data. *CoRR*, abs/1805.12316.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2016. Seqgan: Sequence generative adversarial nets with policy gradient. *CoRR*, abs/1609.05473.

# Document Hashing with Mixture-Prior Generative Models

**Wei Dong**[1], **Qinliang Su**[1,2]*, **Dinghan Shen**[3], **Changyou Chen**[4]
[1] School of Data and Computer Science, Sun Yat-sen University
[2] Guangdong Key Laboratory of Big Data Analysis and Processing, Guangzhou, China
[3] ECE Department, Duke University [4] CSE Department, SUNY at Buffalo
dongw23@mail2.sysu.edu.cn, suqliang@mail.sysu.edu.cn
dinghan.shen@duke.edu, changyou@buffalo.edu

## Abstract

Hashing is promising for large-scale information retrieval tasks thanks to the efficiency of distance evaluation between binary codes. Generative hashing is often used to generate hashing codes in an unsupervised way. However, existing generative hashing methods only considered the use of simple priors, like Gaussian and Bernoulli priors, which limits these methods to further improve their performance. In this paper, two mixture-prior generative models are proposed, under the objective to produce high-quality hashing codes for documents. Specifically, a Gaussian mixture prior is first imposed onto the variational auto-encoder (VAE), followed by a separate step to cast the continuous latent representation of VAE into binary code. To avoid the performance loss caused by the separate casting, a model using a Bernoulli mixture prior is further developed, in which an end-to-end training is admitted by resorting to the straight-through (ST) discrete gradient estimator. Experimental results on several benchmark datasets demonstrate that the proposed methods, especially the one using Bernoulli mixture priors, consistently outperform existing ones by a substantial margin.

## 1 Introduction

Similarity search aims to find items that look most similar to the query one from a huge amount of data (Wang et al., 2018), and are found in extensive applications like plagiarism analysis (Stein et al., 2007), collaborative filtering (Koren, 2008; Wang et al., 2016), content-based multimedia retrieval (Lew et al., 2006), web services (Dong et al., 2004) etc. Semantic hashing is an effective way to accelerate the searching process by representing every document with a compact binary code. In this way, one only needs to evaluate the hamming distance between binary codes, which is much cheaper than the Euclidean distance calculation in the original feature space.

Existing hashing methods can be roughly divided into data-independent and data-dependent categories. Data-independent methods employ random projections to construct hash functions without any consideration on data characteristics, like the locality sensitive hashing (LSH) algorithm (Datar et al., 2004). On the contrary, data dependent hashing seeks to learn a hash function from the given training data in a supervised or an unsupervised way. In the supervised case, a deterministic function which maps the data to a binary representation is trained by using the provided supervised information (e.g. labels) (Liu et al., 2012; Shen et al., 2015; Liu et al., 2016). However, the supervised information is often very difficult to obtain or is not available at all. Unsupervised hashing seeks to obtain binary representations by leveraging the inherent structure information in data, such as the spectral hashing (Weiss et al., 2009), graph hashing (Liu et al., 2011), iterative quantization (Gong et al., 2013), self-taught hashing (Zhang et al., 2010) etc.

Generative models are often considered as the most natural way for unsupervised representation learning (Miao et al., 2016; Bowman et al., 2015; Yang et al., 2017). Many efforts have been devoted to hashing by using generative models. In (Chaidaroon and Fang, 2017), variational deep semantic hashing (VDSH) is proposed to solve the semantic hashing problem by using the variational autoencoder (VAE) (Kingma and Welling, 2013). However, this model requires a two-stage training since a separate step is needed to cast the continuous representations in VAE into binary codes. Under the two-stage training strategy, the model is more prone to get stuck at poor performance (Xu et al., 2015; Zhang et al., 2010; Wang et al.,

* Corresponding author.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5226–5235, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

2013). To address the issue, the neural architecture for generative semantic hashing (NASH) in (Shen et al., 2018) proposed to use a Bernoulli prior to replace the Gaussian prior in VDSH, and further use the straight-through (ST) method (Bengio et al., 2013) to estimate the gradients of functions involving binary variables. It is shown that the end-to-end training brings a remarkable performance improvement over the two-stage training method in VDSH. Despite of superior performances, only the simplest priors are used in these models, i.e. Gaussian in VDSH and Bernoulli in NASH. However, it is widely known that priors play an important role on the performance of generative models (Goyal et al., 2017; Chen et al., 2016; Jiang et al., 2016).

Motivated by this observation, in this paper, we propose to produce high-quality hashing codes by imposing appropriate mixture priors on generative models. Specifically, we first propose to model documents by a VAE with a Gaussian mixture prior. However, similar to the VDSH, the proposed method also requires a separate stage to cast the continuous representation into binary form, making it suffer from the same pains of two-stage training. Then we further propose to use a Bernoulli mixture as the prior, in hopes to yield binary representations directly. An end-to-end method is further developed to train the model, by resorting to the straight-through gradient estimator for neural networks involving binary random variables. Extensive experiments are conducted on benchmark datasets, which show substantial gains of the proposed mixture-prior methods over existing ones, especially the method with a Bernoulli mixture prior.

## 2 Semantic Hashing by Imposing Mixture Priors

In this section, we investigate how to obtain similarity-preserved hashing codes by imposing different mixture priors on variational encoder.

### 2.1 Preliminaries on Generative Semantic Hashing

Let $x \in \mathcal{Z}_+^{|V|}$ denote the bag-of-words representation of a document and $x_i \in \{0,1\}^{|V|}$ denote the one-hot vector representation of the $i$-th word of the document, where $|V|$ denotes the vocabulary size. VDSH in (Chaidaroon and Fang, 2017) proposed to model a document $\mathcal{D}$, which is defined by a sequence of one-hot word representations $\{x_i\}_{i=1}^{|\mathcal{D}|}$, with the joint PDF

$$p(\mathcal{D}, z) = p_\theta(\mathcal{D}|z)p(z), \tag{1}$$

where the prior $p(z)$ is the standard Gaussian distribution $\mathcal{N}(0, I)$; the likelihood has the factorized form $p_\theta(\mathcal{D}|z) = \prod_{i=1}^{|\mathcal{D}|} p_\theta(x_i|z)$, and

$$p_\theta(x_i|z) = \frac{\exp(z^T E x_i + b_i)}{\sum_{j=1}^{|V|} \exp(z^T E x_j + b_j)}; \tag{2}$$

$E \in \mathbb{R}^{m \times |V|}$ is a parameter matrix which connects latent representation $z$ to one-hot representation $x_i$ of the $i$-th word, with $m$ being the dimension of $z$; $b_i$ is the bias term and $\theta = \{E, b_1, ..., b_{|V|}\}$. It is known that generative models with better modeling capability often imply that the obtained latent representations are also more informative.

To increase the modeling ability of (1), we may resort to more complex likelihood $p_\theta(\mathcal{D}|z)$, such as using deep neural networks to relate the latent $z$ to the observation $x_i$, instead of the simple softmax function in (2). However, as indicated in (Shen et al., 2018), employing expressive nonlinear decoders likely destroy the distance-keeping property, which is essential to yield good hashing codes. In this paper, instead of employing a more complex decoder $p_\theta(\mathcal{D}|z)$, more expressive priors are leveraged to address this issue.

### 2.2 Semantic Hashing by Imposing Gaussian Mixture Priors

To begin with, we first replace the standard Gaussian prior $p(z) = \mathcal{N}(0, I)$ in (1) by the following Gaussian mixture prior

$$p(z) = \sum_{k=1}^{K} \pi_k \cdot \mathcal{N}\left(\mu_k, \text{diag}\left(\sigma_k^2\right)\right), \tag{3}$$

where $K$ is the number of mixture components; $\pi_k$ is the probability of choosing the $k$-th component and $\sum_k^K \pi_k = 1$; $\mu_k \in \mathbb{R}^m$ and $\sigma_k^2 \in \mathbb{R}_+^m$ are the mean and variance vectors of the Gaussian distribution of the $k$-th component; and diag$(\cdot)$ means diagonalizing the vector. For any sample $z \sim p(z)$, it can be equivalently generated by a two-stage procedure: 1) choosing a component $c \in \{1, 2, \cdots, K\}$ according to the categorical distribution Cat$(\pi)$ with $\pi = [\pi_1, \pi_2, \cdots, \pi_K]$; 2) drawing a sample from the

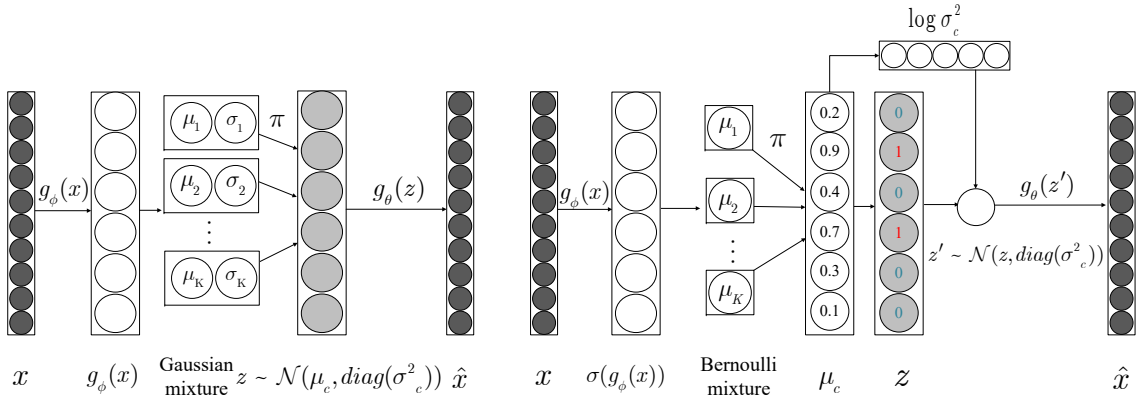


(a) GMSH (b) BMSH

Figure 1: The architectures of the GMSH and BMSH. The data generative process of GMSH is done as follows: (1) Pick a component $c \in \{1, 2, ..., K\}$ from $Cat(\pi)$ with $\pi = [\pi_1, \pi_2, ..., \pi_K]$; (2) Draw a sample $z$ from the picked Gaussian distribution $\mathcal{N}\left(\mu_c, diag(\sigma_c^2)\right)$; (3) Use $g_\theta(z)$ to decode the sample $z$ into an observable $\hat{x}$. The process of generating data in BMSH can be described as follows: (1) Choose a component $c$ from $Cat(\pi)$; (2) Sample a latent vector from the chosen distribution $Bernoulli(\gamma_c)$; (3) Inject data-dependent noise into $z$, and draw $z'$ from $\mathcal{N}(z, diag(\sigma_c^2))$; (4) Then use decoder $g_\theta(z')$ to reconstruct $\hat{x}$.

distribution $\mathcal{N}\left(\mu_c, \text{diag}\left(\sigma_c^2\right)\right)$. Thus, the document $\mathcal{D}$ is modelled as

$$p(\mathcal{D}, z, c) = p_\theta(\mathcal{D}|z)p(z|c)p(c), \tag{4}$$

where $p(z|c) = \mathcal{N}\left(\mu_c, \text{diag}\left(\sigma_c^2\right)\right)$, $p(c) = \text{Cat}(\pi)$ and $p_\theta(\mathcal{D}|z)$ is defined the same as (2).

To train the model, we seek to optimize the lower bound of the log-likelihood

$$\mathcal{L} = E_{q_\phi(z,c|x)}\left[\log \frac{p_\theta(\mathcal{D}|z)p(z|c)p(c)}{q_\phi(z, c|x)}\right], \tag{5}$$

where $q_\phi(z, c|x)$ is the approximate posterior distribution of $p(z, c|x)$ parameterized by $\phi$; here $x$ could be any representation of the documents, like the bag-of-words, TFIDF etc. For the sake of tractability, $q_\phi(z, c|x)$ is further assumed to maintain a factorized form, i.e., $q_\phi(z, c|x) = q_\phi(z|x)q_\phi(c|x)$. Substituting it into the lower bound gives

$$\begin{aligned}\mathcal{L} =& \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(\mathcal{D}|z)\right] - KL\left(q_\phi(c|x)||p(c)\right) \\ & - \mathbb{E}_{q_\phi(c|x)}\left[KL\left(q_\phi(z|x)||p(z|c)\right)\right].\end{aligned} \tag{6}$$

For simplicity, we assume that $q_\phi(z|x)$ and $q_\phi(c|x)$ take the forms of Gaussian and categorical distributions, respectively, and the distribution parameters are defined as the outputs of neural networks. The entire model, including the generative and inference arms, is illustrated in Figure 1(a). Using the properties of Gaussian and categorical distributions, the last two terms in (6) can be expressed in a closed form. Combining with the reparameterization trick in stochastic gradient variational bayes (SGVB) estimator (Kingma and Welling, 2013), the lower bound $\mathcal{L}$ can be optimized w.r.t. model parameters $\{\theta, \pi, \mu_k, \sigma_k, \phi\}$ by error backpropagation and SGD algorithms directly.

Given a document $x$, its hashing code can be obtained through two steps: 1) mapping $x$ to its latent representation by $z = \mu_\phi(x)$, where the $\mu_\phi(x)$ is the encoder mean $\mu_\phi(\cdot)$; 2) thresholding $z$ into binary form. As suggested in (Wang et al., 2013; Chaidaroon et al., 2018; Chaidaroon and Fang, 2017) that when hashing a batch of documents, we can use the median value of the elements in $z$ as the critical value, and threshold each element of $z$ into 0 and 1 by comparing it to this critical value. For presentation conveniences, the proposed semantic hashing model with a Gaussian mixture priors is referred as GMSH.

### 2.3 Semantic Hashing by Imposing Bernoulli Mixture Priors

To avoid the separate casting step used in GMSH, inspired by NASH (Shen et al., 2018), we further propose a *S*emantic *H*ashing model with a *B*ernoulli *M*ixture prior (BMSH). Specifically, we replace the Gaussian mixture prior in GMSH with the following Bernoulli mixture prior

$$p(z) = \sum_{k=1}^{K} \pi_k \cdot \text{Bernoulli}\left(\gamma_k\right), \tag{7}$$

where $\gamma_k \in [0,1]^m$ represents the probabilities of $z$ being 1. Effectively, the Bernoulli mixture prior, in addition to generating discrete samples, plays a similar role as Gaussian mixture prior, which make the samples drawn from different components have different patterns. The samples from the Bernoulli mixture can be generated by first choosing a component $c \in \{1, 2, \cdots, K\}$ from $\text{Cat}(\pi)$ and then drawing a sample from the chosen distribution $\text{Bernoulli}(\gamma_c)$. The entire model can be described as $p(\mathcal{D}, z, c) = p_\theta(\mathcal{D}|z)p(z|c)p(c)$, where $p_\theta(\mathcal{D}|z)$ is defined the same as (2), and $p(c) = \text{Cat}(\pi)$ and $p(z|c) = \text{Bernoulli}(\gamma_c)$.

Similar to GMSH, the model can be trained by maximizing the variational lower bound, which maintains the same form as (6). Different from GMSH, in which $q_\phi(z|x)$ and $p(z|c)$ are both in a Gaussian form, here $p(z|c)$ is a Bernoulli distribution by definition, and thus $q_\phi(z|x)$ is assumed to be the Bernoulli form as well, with the probability of the $i$-th element $z_i$ taking 1 defined as

$$q_\phi(z_i = 1|x) \triangleq \sigma\left(g_\phi^i(x)\right) \tag{8}$$

for $i = 1, 2, \cdots, m$. Here $g_\phi^i(\cdot)$ indicates the $i$-th output of a neural network parameterized by $\phi$. Similarly, we also define the posterior regarding which component to choose as

$$q_\phi(c = k|x) = \frac{\exp\left(h_\phi^k(x)\right)}{\sum_{i=1}^K \exp\left(h_\phi^i(x)\right)}, \tag{9}$$

where $h_\phi^k(x)$ is the $k$-th output of a neural network parameterized by $\phi$. With denotation $\alpha_i = q_\phi(z_i = 1|x)$ and $\beta_k = q_\phi(c = k|x)$, the last two terms in (6) can be expressed in close-form as

$$KL\left(q_\phi(c|x)||p(c)\right) = \sum_{c=1}^K \beta_c \log \frac{\beta_c}{\pi},$$

$$\mathbb{E}_{q_\phi(c|x)}\left[KL\left(q_\phi(z|x)||p(z|c)\right)\right]$$
$$= \sum_{c=1}^K \beta_c \sum_{i=1}^m \left(\alpha_i \log \frac{\alpha_i}{\gamma_c^i} + (1-\alpha_i) \log \frac{1-\alpha_i}{1-\gamma_c^i}\right),$$

where $\gamma_c^i$ denotes the $i$-th element of $\gamma_c$.

Due to the Bernoulli assumption for the posterior $q_\phi(z|x)$, the commonly used reparameterization trick for Gaussian distribution cannot be used to directly estimate the first term $\mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(\mathcal{D}|z)\right]$ in (6). Fortunately, inspired by the straight-through gradient estimator in (Bengio et al., 2013), we can parameterize the $i$-th element of binary sample $z$ from $q_\phi(z|x)$ as

$$z_i = 0.5 \times \left(sign\left(\sigma(g_\phi^i(x)) - \xi_i\right) + 1\right), \tag{10}$$

where $sign(\cdot)$ the is the sign function, which is equal to 1 for nonnegative inputs and -1 otherwise; and $\xi_i \sim \text{Uniform}(0, 1)$ is a uniformly random sample between 0 and 1.

The reparameterization method used above can guarantee generating binary samples. However, backpropagation cannot be used to optimize the lower bound $\mathcal{L}$ since the gradient of $sign(\cdot)$ w.r.t. its input is zero almost everywhere. To address this problem, the straight-through(ST) estimator (Bengio et al., 2013) is employed to estimate the gradient for the binary random variables, where the derivative of $z_i$ w.r.t $\phi$ is simply approximated by $0.5 \times \frac{\partial \sigma(g_\phi^i(x))}{\partial \phi}$. Thus, the gradients can then be backpropagated through discrete variables. Similar to NASH (Shen et al., 2018), data-dependent noises are also injected into the latent variables when reconstructing the document $x$ so as to obtain more robust binary representations. The entire model of BMSH, including generative and inference parts, is illustrated in Figure 1(b).

To understand how the mixture-prior model works differently from the simple prior model, we examine the main difference term $\mathbb{E}_{q_\phi(c|x)}\left[KL\left(q_\phi(z|x)||p(z|c)\right)\right]$ in (6), where $q_\phi(c|x)$ is the approximate posterior probability that indicates the document $x$ is generated by the $c$-th component distribution with $c \in \{1, 2, \cdots, K\}$. In the mixture-prior model, the approximate posterior $q_\phi(z|x)$ is compared to all mixture components $p(z|c) = \mathcal{N}\left(\mu_c, \text{diag}(\sigma_c^2)\right)$. The term $\mathbb{E}_{q_\phi(c|x)}\left[KL\left(q_\phi(z|x)||p(z|c)\right)\right]$ can be understood as the average of all these KL-divergences weighted by the probabilities $q_\phi(c|x)$. Thus, comparing to the simple-prior model, the mixture-prior model is endowed with more flexibilities, allowing the documents to be regularized by different mixture components according to their context.

### 2.4 Extensions to Supervised Hashing

When label information is available, it can be leveraged to yield more effective hashing codes since labels provide extra information about the similarities of documents. Specifically, a mapping from the latent representation $z$ to the cor-

responding label $y$ is learned for each document. The mapping encourages latent representations of documents with the same label to be close in the latent space, while those with different labels to be distant. A classifier built from a two-layer MLP is employed to parameterize this mapping, with its cross-entropy loss denoted by $\mathcal{L}_{dis}(z, y)$. Taking the supervised objective into account, the total loss is defined as

$$\mathcal{L}_{total} = -\mathcal{L} + \alpha \mathcal{L}_{dis}(z, y), \tag{11}$$

where $\mathcal{L}$ is the lower bound arising in GMSH or BMSH model; $\alpha$ controls the relative weight of the two losses. By examining the total loss $\mathcal{L}_{total}$, it can be seen that minimizing the loss encourages the model to learn a representation $z$ that accounts for not only the unsupervised content similarities of documents, but also the supervised similarities from the extra label information.

## 3 Related Work

Existing hashing methods can be categorized into data independent and data dependent methods. A typical example of data independent hashing is the local-sensitive hashing (LSH) (Datar et al., 2004). However, such method usually requires long hashing codes to achieve satisfactory performance. To yield more effective hashing codes, more and more researches focus on data dependent hashing methods, which include unsupervised and supervised methods. Unsupervised hashing methods only use unlabeled data to learn hash functions. For example, spectral hashing (SpH) (Weiss et al., 2009) learns the hash function by imposing balanced and uncorrelated constraints on the learned codes. Iterative quantization (ITQ) (Gong et al., 2013) generates the hashing codes by simultaneously maximizing the variance of each binary bit and minimizing the quantization error. In (Zhang et al., 2010), the authors proposed to decompose the learning procedure into two steps: first learning hashing codes for documents via unsupervised learning, then using $\ell$ binary classifiers to predict the $\ell$-bit hashing codes. Since the labels provide useful guidance in learning effective hash functions, supervised hashing methods are proposed to leverage the label information. For instance, binary reconstruction embedding (BRE) (Kulis and Darrell, 2009) learns the hash function by minimizing the reconstruction error between the original distances and the hamming distances of the corresponding hashing codes. Supervised hashing with kernels (KSH) (Liu et al., 2012) is a kernel-based method, which utilizes the pairwise information between samples to generate hashing codes by minimizing the hamming distances on similar pairs and maximizing those on dissimilar pairs.

Recently, VDSH (Chaidaroon and Fang, 2017) proposed to use a VAE to learn the latent representations of documents and then use a separate stage to cast the continuous representations into binary codes. While fairly successful, this generative hashing model requires a two-stage training. NASH (Shen et al., 2018) proposed to substitute the Gaussian prior in VDSH with a Bernoulli prior to tackle this problem, by using a straight-through estimator (Bengio et al., 2013) to estimate the gradient of neural network involving the binary variables. This model can be trained in an end-to-end manner. Our models differ from VDSH and NASH in that mixture priors are employed to yield better hashing codes, whereas only the simplest priors are used in both VDSH and NASH.

## 4 Experiments

### 4.1 Experimental Setups

**Datasets** Three public benchmark datasets are used in our experiments. $i)$ $Reuters21578$: A dataset consisting of 10788 news documents from 90 different categories; $ii)$ $20Newsgroups$: A collection of 18828 newsgroup posts that are divided into 20 different newsgroups; $iii)$ $TMC$: A dataset containing the air traffic reports provided by NASA, which includes 21519 training documents with 22 labels.

**Training Details** We experiment with the four models proposed in this paper, i.e., GMSH and BMSH for unsupervised hashing, and GMSH-S and BMSH-S for supervised hashing. The same network architectures as VDSH and NASH are used in our experiments to admit a fair comparison. Specifically, a two-layer feed-forward neural network with 500 hidden units and ReLU activation function is employed as the encoder and the extra classifier in the supervised case, while the decoder is the same as that stated in (2). Similar to VDSH and NASH (Chaidaroon and Fang, 2017; Shen et al., 2018), the TFIDF feature of a document is used as the input to the encoder. The Adam optimizer (Kingma and Ba, 2014) is used in the training of our models, and its learning rate is

| Datasets | TMC | | | | 20Newsgroups | | | | Reuters | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | 16bit | 32bit | 64bit | 128bit | 16bit | 32bit | 64bit | 128bit | 16bit | 32bit | 64bit | 128bit |
| LSH | 0.4393 | 0.4514 | 0.4553 | 0.4773 | 0.0597 | 0.0666 | 0.0770 | 0.0949 | 0.3215 | 0.3862 | 0.4667 | 0.5194 |
| S-RBM | 0.5108 | 0.5166 | 0.5190 | 0.5137 | 0.0604 | 0.0533 | 0.0623 | 0.0642 | 0.5740 | 0.6154 | 0.6177 | 0.6452 |
| SpH | 0.6055 | 0.6281 | 0.6143 | 0.5891 | 0.3200 | 0.3709 | 0.3196 | 0.2716 | 0.6340 | 0.6513 | 0.6290 | 0.6045 |
| STH | 0.3947 | 0.4105 | 0.4181 | 0.4123 | 0.5237 | 0.5860 | 0.5806 | 0.5433 | 0.7351 | 0.7554 | 0.7350 | 0.6986 |
| VDSH | 0.6853 | 0.7108 | 0.4410 | 0.5847 | 0.3904 | 0.4327 | 0.1731 | 0.0522 | 0.7165 | 0.7753 | 0.7456 | 0.7318 |
| NASH | 0.6573 | 0.6921 | 0.6548 | 0.5998 | 0.5108 | 0.5671 | 0.5071 | 0.4664 | 0.7624 | 0.7993 | 0.7812 | 0.7559 |
| GMSH | 0.6736 | 0.7024 | 0.7086 | 0.7237 | 0.4855 | 0.5381 | 0.5869 | 0.5583 | 0.7672 | 0.8183 | 0.8212 | 0.7846 |
| BMSH | **0.7062** | **0.7481** | **0.7519** | **0.7450** | **0.5812** | **0.6100** | **0.6008** | **0.5802** | **0.7954** | **0.8286** | **0.8226** | **0.7941** |

Table 1: The precisions of the top 100 retrieved documents on three datasets with different numbers of hashing bits in unsupervised hashing.

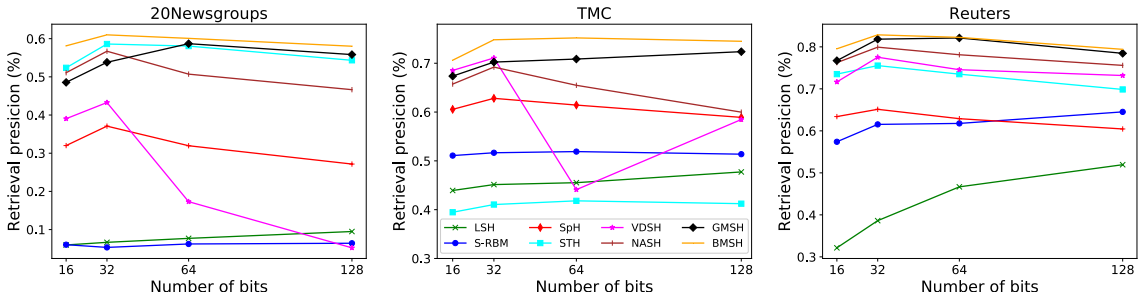


Figure 2: The performance of unsupervised hashing models on three datasets with various numbers of hashing bits.

set to be $1 \times 10^{-3}$, with a decay rate of 0.96 for every 10000 iterations. The component number $K$ and the parameter $\alpha$ in (11) are determined based on the validation set.

**Baselines** For unsupervised semantic hashing, we compare the proposed GMSH and BMSH with the following models: locality sensitive hashing (LSH), stack restricted boltzmann machines (S-RBM), spectral hashing (SpH), self-taught hashing (STH), variational deep semantic hashing (VDSH) and neural architecture for semantic hashing(NASH). For supervised semantic hashing, we also compare GMSH-S and BMSH-S with the following baselines: supervised hashing with kernels (KSH) (Liu et al., 2012), semantic hashing using tags and topic modeling (SHTTM) (Wang et al., 2013), supervised VDSH and supervised NASH.

**Evaluation Metrics** For every document from the testing set, we retrieve similar documents from the training set based on the hamming distance between their hashing codes. For each query, 100 closest documents are retrieved, among which the documents sharing the same label as the query are deemed as the relevant results. The ratio between the number of relevant ones and the total number, which is 100, is calculated as the similarity search precision. The averaged value over all testing documents is then reported. The retrieval precisions under the cases of 16 bits, 32 bits, 64 bits, 128 bits hashing codes are evaluated, respectively.

### 4.2 Performance Evaluation of Unsupervised Semantic Hashing

Table 1 shows the performance of the proposed and baseline models on three datasets under the unsupervised setting, with the number of hashing bits ranging from 16 to 128. From the experimental results, it can be seen that GMSH outperforms previous models under all considered scenarios on both TMC and Reuters. It also achieves better performance on 20Newsgroups when the length of hashing codes is large, e.g. 64 or 128. Comparing to VDSH using the simple Gaussian prior, the proposed GMSH using a Gaussian mixture prior exhibits better retrieval performance overall. This strongly demonstrates the benefits of using mixture priors on the task of semantic hashing. One possible explanation is that the mixture prior enables the documents from different categories to be regularized by different distributions, guiding the model to learn more distinguishable representations for documents from different categories. It can be further observed that among all methods, BMSH achieves the best performance under different datasets and hashing codes length consistently. This may be attributed to the imposed Bernoulli mixture prior, which offers both the ad-

| Datasets | TMC | | | | 20Newsgroups | | | | Reuters | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | 16bit | 32bit | 64bit | 128bit | 16bit | 32bit | 64bit | 128bit | 16bit | 32bit | 64bit | 128bit |
| KSH | 0.6842 | 0.7047 | 0.7175 | 0.7243 | 0.5559 | 0.6103 | 0.6488 | 0.6638 | 0.8376 | 0.8480 | 0.8537 | 0.8620 |
| SHTTM | 0.6571 | 0.6485 | 0.6893 | 0.6474 | 0.3235 | 0.2357 | 0.1411 | 0.1299 | 0.8520 | 0.8323 | 0.8271 | 0.8150 |
| VDSH-S | 0.7887 | 0.7883 | 0.7967 | 0.8018 | 0.6791 | 0.7564 | 0.6850 | 0.6916 | 0.9121 | 0.9337 | 0.9407 | 0.9299 |
| NASH-DN-S | 0.7946 | 0.7987 | 0.8014 | 0.8139 | 0.6973 | 0.8069 | 0.8213 | 0.7840 | 0.9327 | 0.9380 | 0.9427 | 0.9336 |
| GMSH-S | 0.7806 | 0.7929 | 0.8103 | 0.8144 | 0.6972 | 0.7426 | 0.7574 | 0.7690 | 0.9144 | 0.9175 | 0.9414 | 0.9522 |
| BMSH-S | **0.8051** | **0.8247** | **0.8340** | **0.8310** | **0.7316** | **0.8144** | **0.8216** | **0.8183** | **0.9350** | **0.9640** | **0.9633** | **0.9590** |

Table 2: The performances of different supervised hashing models on three datasets under different lengths of hashing codes.

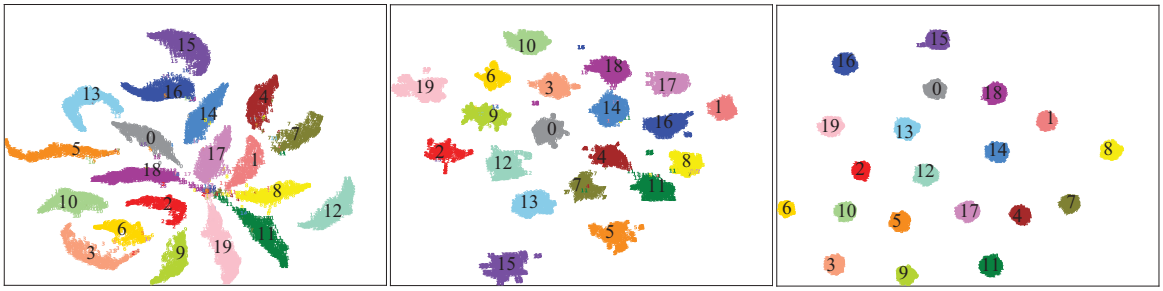

(a) VDSH-S (b) GMSH-S (c) BMSH-S

Figure 3: Visualization of the 32-dimensional document latent semantic embeddings learned by VDSH-S, GMSH-S and MBSH-S on 20Newsgroups dataset. Each data point in the figure denotes a document, with each color representing one category. The number shown with the color is the ground-true category ID.

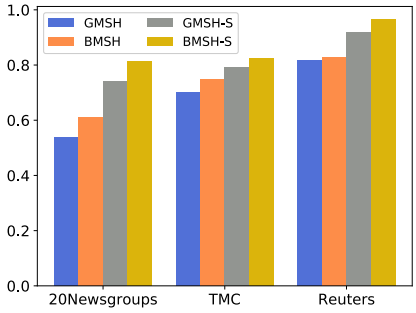


Figure 4: The retrieval precisions of GMSH and BMSH on three datasets in both unsupervised and supervised scenarios.

vantages of producing more distinguishable codes with a mixture prior and end-to-end training enabled by a Bernoulli prior. BMSH integrates the merits of NASH and GMSH, and thus is more suitable for the hashing task.

Figure 2 shows how retrieval precisions vary with the number of hashing bits on the three datasets. It can be observed that as the number increases from 32 to 128, the retrieval precisions of most previous models tend to decrease. This phenomenon is especially obvious for VDSH, in which the precisions on all three datasets drop by a significant margin. This interesting phenomenon has been reported in previous works (Shen et al., 2018; Chaidaroon and Fang, 2017; Wang et al., 2013; Liu et al., 2012), and the reason could be overfitting since the model with long hashing codes is more likely to overfitting (Chaidaroon and Fang, 2017; Shen et al., 2018). However, it can be seen that our model is more robust to the number of hashing bits. When the number is increased to 64 or 128, the performance of our models is kept almost unchanged. This may be also attributed to the mixture priors imposed in our models, which can regularize the models more effectively.

### 4.3 Performance Evaluation of Supervised Semantic Hashing

We evaluate the performance of supervised hashing in this section. Table 2 shows the performances of different supervised hashing models on three datasets under different lengths of hashing codes. We observe that all of the VAE-based generative hashing models (i.e VDSH, NASH, GMSH and BMSH) exhibit better performance, demonstrating the effectiveness of generative models on the task of semantic hashing. It can be also seen that BMSH-S achieves the best performance, suggesting that the advantages of Bernoulli mixture priors can also be extended to the supervised scenarios.

To gain a better understanding about the relative

| D / $K$ | 20Newsgroups | | TMC | | Reuters | |
|---|---|---|---|---|---|---|
| | GMSH | BMSH | GMSH | BMSH | GMSH | BMSH |
| 5 | 0.4708 | 0.5977 | 0.6886 | 0.7492 | 0.7888 | 0.8152 |
| 10 | 0.4778 | 0.6007 | 0.6862 | 0.7479 | 0.8039 | 0.8226 |
| 20 | 0.5381 | 0.6100 | 0.6883 | **0.7495** | 0.8182 | 0.**8286** |
| 40 | 0.5197 | 0.6015 | 0.7024 | 0.7481 | 0.8169 | 0.8258 |
| 80 | 0.5188 | 0.6012 | **0.7033** | 0.7467 | 0.8087 | 0.8253 |
| GT | **0.5381** | **0.6100** | 0.6960 | 0.7443 | **0.8183** | 0.8279 |

Table 3: Precisions of top 100 retrieved documentswith different numberer of clusters, $K$ denotes the number of components, D represents datasets, GT represents the ground truth number of classes for each dataset.

performance gain of the four proposed models, the retrieval precisions of GMSH, BMSH, GMSH-S and BMSH-S using 32-bit hashing codes on the three datasets are plotted together in Figure 4. It can be obviously seen that GMSH-S and BMSH-S outperform GMSH and BMSH by a substantial margin, respectively. This suggests that the proposed generative hashing models can also leverage the label information to improve the hashing codes' quality.

### 4.4 Impacts of the Component Number

To investigate the impacts of component number, experiments are conducted for GMSH and BMSH under different values of $K$. For demonstration convenience, the length of hashing codes is fixed to 32. Table 3 shows the precisions of top 100 retrieved documents when the number of components $K$ is set to different values. We can see that the retrieval precisions of the proposed models, especially the BMSH, are quite robust to this parameter. For BMSH, the difference between the best and worst precisions on the three datasets are 0.0123, 0.0052 and 0.0134, respectively, which are small comparing to the gains that BMSH has achieved. One exception is the performance of GMSH on 20Newsgroups dataset. However, as seen from Table 3, as long as the number $K$ is not too small, the performance loss is still acceptable. It is worth noting that the worst performance of GMSH on 20Newsgroups is 0.4708, which is still better than VDSH's 0.4327 as in Table 1. For the BMSH model, the performance is stable across all the considered datasets and $K$ values.

### 4.5 Visualization of Learned Embeddings

To understand the performance gains of the proposed models better, we visualize the learned representations of VDSH-S, GMSH-S and BMSH-S on 20Newsgroups dataset. UMAP (McInnes et al., 2018) is used to project the 32-dimensional latent representations into a 2-dimensional space, as shown in Figure 3. Each data point in the figure denotes a document, with each color representing one category. The number shown with the color is the ground truth category ID. It can be observed from Figure 3 (a) and (b) that more embeddings are clustered correctly when the Gaussian mixture prior is used. This confirms the advantages of using mixture priors in the task of hashing. Furthermore, it is observed that the latent embeddings learned by BMSH-S can be clustered almost perfectly. In contrast, many embeddings are found to be clustered incorrectly for the other two models. This observation is consistent with the conjecture that mixture prior and end-to-end training are both useful for semantic hashing.

## 5 Conclusions

In this paper, deep generative models with mixture priors were proposed for the tasks of semantic hashing. We first proposed to use a Gaussian mixture prior, instead of the standard Gaussian prior in VAE, to learn the representations of documents. A separate step was then used to cast the continuous latent representations into binary hashing codes. To avoid the requirement of a separate casting step, we further proposed to use the Bernoulli mixture prior, which offers the advantages of both mixture prior and the end-to-end training. Comparing to strong baselines on three public datasets, the experimental results indicate that the proposed methods using mixture priors outperform existing models by a substantial margin. Particularly, the semantic hashing model with Bernoulli mixture prior (BMSH) achieves state-of-the-art results on all the three datasets considered in this paper.

## 6 Acknowledgements

This work is supported by the National Natural Science Foundation of China (NSFC) under Grant No. 61806223, U1711262, U1501252, U1611264 and U1711261, and National Key R&D Program of China (2018YFB1004404).

## References


Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*.

Suthee Chaidaroon, Travis Ebesu, and Yi Fang. 2018. Deep semantic text hashing with weak supervision. SIGIR.

Suthee Chaidaroon and Yi Fang. 2017. Variational deep semantic hashing for text documents. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 75–84. ACM.

Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. Variational lossy autoencoder. *arXiv preprint arXiv:1611.02731*.

Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262. ACM.

Xin Dong, Alon Halevy, Jayant Madhavan, Ema Nemes, and Jun Zhang. 2004. Similarity search for web services. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 372–383. VLDB Endowment.

Hao Fu, Chunyuan Li, Xiaodong Liu, Jianfeng Gao, Asli Celikyilmaz, and Lawrence Carin. 2019. Cyclical annealing schedule: A simple approach to mitigating kl vanishing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 240–250.

Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. 2013. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12):2916–2929.

Prasoon Goyal, Zhiting Hu, Xiaodan Liang, Chenyu Wang, and Eric P Xing. 2017. Nonparametric variational auto-encoders for hierarchical representation learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5094–5102.

Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. 2016. Variational deep embedding: An unsupervised and generative approach to clustering. *arXiv preprint arXiv:1611.05148*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM.

Brian Kulis and Trevor Darrell. 2009. Learning to hash with binary reconstructive embeddings. In *Advances in neural information processing systems*, pages 1042–1050.

Michael S Lew, Nicu Sebe, Chabane Djeraba, and Ramesh Jain. 2006. Content-based multimedia information retrieval: State of the art and challenges. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 2(1):1–19.

Haomiao Liu, Ruiping Wang, Shiguang Shan, and Xilin Chen. 2016. Deep supervised hashing for fast image retrieval. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2064–2072.

Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang. 2012. Supervised hashing with kernels. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2074–2081. IEEE.

Wei Liu, Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. 2011. Hashing with graphs. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 1–8. Citeseer.

Leland McInnes, John Healy, Nathaniel Saul, and Lukas Grossberger. 2018. Umap: Uniform manifold approximation and projection. *The Journal of Open Source Software*, 3(29):861.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *International Conference on Machine Learning*, pages 1727–1736.

Dinghan Shen, Qinliang Su, Paidamoyo Chapfuwa, Wenlin Wang, Guoyin Wang, Lawrence Carin, and Ricardo Henao. 2018. Nash: Toward end-to-end neural architecture for generative semantic hashing. *arXiv preprint arXiv:1805.05361*.

Fumin Shen, Chunhua Shen, Wei Liu, and Heng Tao Shen. 2015. Supervised discrete hashing. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 37–45.

Benno Stein, Sven Meyer zu Eissen, and Martin Potthast. 2007. Strategies for retrieving plagiarized

documents. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 825–826. ACM.

Jingdong Wang, Ting Zhang, Nicu Sebe, Heng Tao Shen, et al. 2018. A survey on learning to hash. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):769–790.

Jun Wang, Wei Liu, Sanjiv Kumar, and Shih-Fu Chang. 2016. Learning to hash for indexing big dataa survey. *Proceedings of the IEEE*, 104(1):34–57.

Qifan Wang, Dan Zhang, and Luo Si. 2013. Semantic hashing using tags and topic modeling. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 213–222. ACM.

Yair Weiss, Antonio Torralba, and Rob Fergus. 2009. Spectral hashing. In *Advances in neural information processing systems*, pages 1753–1760.

Jiaming Xu, Peng Wang, Guanhua Tian, Bo Xu, Jun Zhao, Fangyuan Wang, and Hongwei Hao. 2015. Convolutional neural networks for text hashing. In *IJCAI*, pages 1369–1375.

Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. 2017. Improved variational autoencoders for text modeling using dilated convolutions. *arXiv preprint arXiv:1702.08139*.

Dell Zhang, Jun Wang, Deng Cai, and Jinsong Lu. 2010. Self-taught hashing for fast similarity search. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 18–25. ACM.

# On Efficient Retrieval of Top Similarity Vectors

**Shulong Tan, Zhixin Zhou, Zhaozhuo Xu, Ping Li**
Cognitive Computing Lab
Baidu Research USA
1195 Bordeaux Dr, Sunnyvale, CA 94089, USA
10900 NE 8th St, Bellevue, WA 98004, USA
{shulongtan,zhixinzhou,v_xuzhaozhuo,liping11}@baidu.com

## Abstract

Retrieval of relevant vectors produced by representation learning critically influences the efficiency in natural language processing (NLP) tasks. In this paper we demonstrate an efficient method for searching vectors via a typical non-metric matching function: inner product. Our method, which constructs an approximate Inner Product Delaunay Graph (**IPDG**) for top-1 Maximum Inner Product Search (MIPS), transforms retrieving the most suitable latent vectors into a graph search problem with great benefits of efficiency. Experiments on data representations learned for different machine learning tasks verify the outperforming effectiveness and efficiency of the proposed IPDG.

## 1 Introduction

With the popularity of representation learning methods, such as Word2vec (Mikolov et al., 2013a), words are represented as real-valued embedding vectors in the semantic space. Therefore, retrieval of similar word embeddings is one of the most basic operations in natural language processing with wide applicability in synonym extraction (Yoon et al., 2017), sentence alignment (Levy et al., 2017), polysemous word learning (Sun et al., 2017) and semantic search for documents related to a query.

In this work, we address on efficient retrieval of similar word embeddings via inner product (dot product) similarity. Inner product is a general semantic matching function with applications in neural probabilistic language models (Bengio et al., 2003), machine translation (Gao et al., 2014), question answering (Lee et al., 2015), and attention mechanisms (Vaswani et al., 2017). For normalized vectors, inner product is equivalent to cosine similarity, which is a common semantic textual similarity utilized in semantic classification and search (Sahami and Heilman, 2006; Ramage et al., 2009; Agirre et al., 2012; Huang et al., 2013; Liu et al., 2015; Faruqui et al., 2016; Sultan et al., 2016; Köper and im Walde, 2018; Gong et al., 2018), Relation Extraction (RE) (Plank and Moschitti, 2013) and text coherence evaluation (Putra and Tokunaga, 2017). For un-normalized vectors, although cosine similarity is still widely applied, the final matching scores of word embeddings are usually weighted (Acree et al., 2016; Srinivas et al., 2010) by ranking-based coefficients (e.g., the side information), which transforms the problem back to search via inner product (see Eq. (2)).

Formally, retrieving the most similar word with the inner product ranking function is a *Maximum Inner Product Search* (**MIPS**) problem. MIPS is a continuously addressed topic (Bachrach et al., 2014; Shrivastava and Li, 2014; Kalantidis and Avrithis, 2014; Shrivastava and Li, 2015; Guo et al., 2016; Wu et al., 2017), and it has non-trivial differences with traditional *Approximate Nearest Neighbor Search* (ANNS) (Friedman et al., 1975, 1977; Indyk and Motwani, 1998) problems. ANNS is an optimization problem of finding the close points to the query point in a given set. Usually, the "close" means smaller in metric distances such as cosine or Euclidean distance, which have obvious geometrical implications. However, inner product is a typical non-metric measure, which distinguishes MIPS from traditional ANNS problems. Thus, methods designed for ANNS may have performance limitations in MIPS. For NLP tasks, such as retrieving relevant word embeddings by cosine and Euclidean distances, different ANNS methods have been studied (Sugawara et al., 2016). To our best knowledge, there is little literature on MIPS for retrieving word or language representations.

Currently, search on graph methods, such as *Hierarchical Navigable Small World graphs* (HNSW), is regarded as the state-of-the-art ANNS method (Malkov and Yashunin, 2018). Performance evaluation has demonstrated that HNSW is



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5236–5246,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

able to strongly outperform other methods ANNS benchmarks for metric distances. Meanwhile, the graph structure also has the flexibility of defining measures on edges, making HNSW feasible for MIPS. Morozov et al (Morozov and Babenko, 2018) conduct HNSW for MIPS and achieve positive results and also they introduce concepts of Delaunay Graph to explain similarity graph based methods for MIPS. Nevertheless, the link between HNSW and Delaunay Graph is still tenuous. Although global optima of MIPS will be retrieved by Delaunay Graph, there are little evidence showing that HNSW approximates proper Delauny Graph for inner product. How to provide a solid graph-based MIPS method is still an open question.

In this paper, we propose a new search on graph method, namely Inner Product Delaunay Graph (**IPDG**), for MIPS. Our key contributions can be summarized as follows:

- Design an edge selection algorithm specifically for inner product that reduces useless edges on graph and thus improves the searching efficiency.
- Propose a two rounds graph construction algorithm for effectively approximating Delaunay Graph under inner product.
- Empirically evaluate the effectiveness and efficiency. Provide a state-of-the-art MIPS method for similarity search in word embedding datasets.

The organization of this paper is as below: in the next section, we will introduce the research background. In Section 3, the approximate Inner Product Delaunay Graph (IPDG) will be introduced. For Section 4, we explore the effectiveness and efficiency of IPDG in maximum inner product word retrieval and compare it with state-of-the-art MIPS methods. Section 5 concludes the whole paper.

## 2 Background

In this section, we will first introduce the definition of *Maximum Inner Product Search* (MIPS) problem and review state-of-the-art methods for MIPS. Later a theoretical solution for MIPS by searching on the Delaunay Graphs will be summarized.

### 2.1 Problem Statement

In machine learning tasks, embedding methods such as Word2vec (Mikolov et al., 2013a,b), Glove (Pennington et al., 2014) or deep collaborative filtering (Xu et al., 2018) learn representations of data as dense distributed real-value vectors. Formally, for latent space $X \subset \mathbb{R}^d$, given an arbitrary query vector $q \in X$ and a set of vectors $S = \{x_1, \ldots, x_n\} \subset X$, vector similarity is defined as a continuous symmetric matching function, $f : X \times X \to \mathbb{R}$. The goal of similar vector retrieval is to find:

$$\arg\max_{x \in S} f(x, q). \tag{1}$$

In our paper, we specially discuss the non-metric similarity measure, inner product:

$$f(x, q) = x^\top q, \quad x, q \in X = \mathbb{R}^d \backslash \{0\}.$$

Without loss of the generality, we can always assume $\|q\| = 1$. We are not interested in the zero vector since its inner product with any vector is always zero. The problem in Eq. (1) with respect to the inner product is often referred to as *Maximum Inner Product Search* (MIPS) in literature.

The weighted cosine ANNS problem can also be viewed as the MIPS problem. We consider a data set $S = \{(z_i, w_i) : i \in [n]\}$ where $w_i$ is an real scalar and $z_i$ is a vector.

$$w \cos(z, q) = w \frac{z^\top q}{\|z\|\|q\|} = \frac{wz}{\|z\|}^\top \frac{q}{\|q\|}, \tag{2}$$

where $\|q\| = 1$. As can be seen, weighted ANNS w.r.t. cosine similarity is equivalent to MIPS by letting $x_i = w_i z_i / \|z_i\|$.

### 2.2 Related Works

Previous approaches for *Maximum Inner Product Search* (**MIPS**) can be mainly categorized into: (1) reducing MIPS to ANNS; (2) non-reduction methods. Reduction methods add wrappers on indexed data and queries asymmetrically and reduce the MIPS problem to ANNS in metric spaces (Shrivastava and Li, 2015; Bachrach et al., 2014). For example, given the query $q$, the indexed data $S = \{x_1, ..., x_n\}$ and $\Phi = \max_i ||x_i||$, the wrapper can be defined as:

$$P(x) = [x/\Phi; \sqrt{1 - ||x||^2/\Phi^2}], \tag{3}$$

$$Q(q) = [q; 0]. \tag{4}$$

It is not difficult to prove that searching on the new data by cosine or $\ell_2$-distance is equal to search on the original data by inner product. Recently, researchers found that methods above can be improved further, based on the observation of the long tail distribution in data norms (Huang et al., 2018;

Yan et al., 2018). New approaches are proposed by adding wrappers for each norm range, such as **Range-LSH** (Yan et al., 2018). With reductions like the above one, any ANNS methods can be applied for MIPS. However, it was shown that there are performance limitations for the reduction MIPS methods (Morozov and Babenko, 2018).

Recently, more and more non-reduction methods are proposed, specifically for MIPS. Guo et al. proposed an MIPS method based on **Product Quantization (PQ)** (Guo et al., 2016). Yu et al. used an upper bound of inner product as the approximation of MIPS and designed a greedy search algorithm to find this approximation, called **Greedy-MIPS** (Yu et al., 2017). Graph-based non-reduction MIPS method, **ip-NSW**, was firstly introduced in Morozov and Babenko (2018) and the theoretical basis for conducting MIPS by similarity graph was also provided. Continuing of the advantages of similarity graph based methods for ANNS, ip-NSW showed superior performance for MIPS.

### 2.3 Delaunay Graph

Delaunay Graph plays an important role in the similarity search. The properties and construction of $\ell^2$-Delaunay Graph have been considered in literature (Aurenhammer, 1991; Cignoni et al., 1998). Indeed, one can generalize the definition to any real binary function, including inner product.

**Definition 2.1.** The *Voronoi cell* $R_i$ with respect to $f$ and $x_i$ is the set

$$R_i := \{q \in X : f(x_i, q) \geq f(x, q) \text{ for all } x \in S\}.$$

Moreover, $x \in S$ is an *extreme* point if it is associated with a nonempty Voronoi cell.

**Definition 2.2.** The *Delaunay Graph* $G$ with respect to $f$ and $S$ is an undirected graph with vertices $S$ satisfying $\{x_i, x_j\} \in G$ if and only if $R_i \cap R_j \neq \emptyset$.

An example of Voronoi cells and corresponding Delaunay Graph in inner product space is shown in Figure 1. Regions in different colors correspond to Voronoi cells for extreme points (red ones). Delaunay Graph connects extreme points. Different from metric similarities (e.g. $\ell^2$-norm), the Voronoi cells of some data points with respect to inner product are possibly empty. By Definition 2.2, a data point is isolated (i.e., have no incident edges) if its Voronoi cell is empty. As we can see in Figure 1, there are many isolated points (blue ones). The

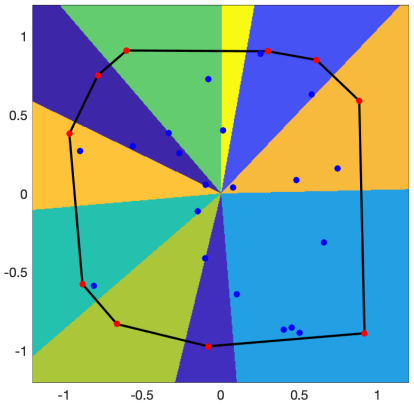


Figure 1: This shows the relation between Delaunay Graph and Voronoi cells in inner product space. The red dots are *extreme* points of each Voronoi cell. Delaunay Graph connects extreme points with black edges. If we search on this dataset, every query has a maximum inner product with one of these extreme points (i.e., red ones).

proportion of extreme points is relatively small in general. And Theorem 2.1 will show that only extreme points can achieve a maximum inner product score for any nonzero query.

The definition of an extreme point is equivalent to the one in (Barber et al., 1996), i.e., $x \in S$ is extreme if and only if $x$ is on the boundary of the convex hull of $S$. In the two dimensional cases, the edges form the boundary of the convex hull, which is also shown in Figure 1.

### 2.4 Search on Delaunay Graph

Searching on the Delaunay Graph is demonstrated effective for similarity search (Morozov and Babenko, 2018). In the inner product case, given any query vector $q \in X$, we start from an extreme point, then move to its neighbor that has a larger inner product with $q$. We repeat this step until getting an extreme point has a larger inner product with $q$ than all its neighbors and then we return it. It can be demonstrated this returned local optimum is actually the global optimum.

Generally, for any searching measure $f$, if the corresponding Voronoi cells are connected, then the local optimum returned by the greedy search is also the global optimum. Formally the statement can be summarized as below. The proof can be found in Morozov and Babenko (2018).

**Theorem 2.1.** Suppose $f$ satisfies that the Voronoi cells $R_i$ with respect to any subsets of $S$ (including $S$ itself) are connected on $X$, and $G$ is the Delaunay Graph with respect to $f$ and some $S$, then for $q \in X$, a local maximum in the greedy search starting

from an extreme point, that is, $x_i \in S$ satisfies

$$f(x_i, q) \geq \max_{x \in N(x_i)} f(x, q) \quad (5)$$
$$\text{where} \quad N(x_i) = \{x \in S : \{x_i, x\} \in G\}$$

is a global maximum.

Suppose the assumptions (i.e., connected Voronoi cells) in Theorem 2.1 hold, we say searching on Delaunay Graph can find the global maximum. It is easy to check that the assumptions hold for the inner product case since the Voronoi cells w.r.t. the inner product are either empty or a convex cone, so they are connected. Then we can claim that searching on Delaunay Graph in inner product, the vector in $S$ that has the maximum inner product with the query vector will be retrieved.

## 3 Inner Product Delaunay Graph

Although the Delaunay Graph has demonstrated its potentials in similarity search, the direct construction of the Delaunay Graph in large scale and high dimensional datasets is unfeasible due to the exponentially growing number of edges in high dimension. To remedy this issue, practical algorithms usually approximate Delaunay Graphs. In this section, we will present the new proposed algorithm for constructing approximate Delaunay Graph in inner product space, namely Inner Product Delaunay Graph (IPDG). Two key features of our algorithm will be introduced first: i) edge selection specifically for inner product; and ii) the two rounds graph construction. And then we will conduct a case study on the toy dataset to show the effectiveness of IPDG in constructing better approximate Delaunay Graphs for inner product.

### 3.1 Edge Selection for Inner Product

To balance the effectiveness (retrieval of the nearest neighbor) and the efficiency (complete the process within limit time) of the retrieval, some empirical tricks are usually applied in previous search on graph methods: a) use directed edges instead of undirected edges; b) restrict the degree of outgoing edges for each node; and c) select more diverse outgoing edges (Malkov and Yashunin, 2018; Morozov and Babenko, 2018).

Specifically, for the inner product case, ip-NSW proposed in (Morozov and Babenko, 2018) applies all tricks listed above (although the authors did not mention it in the paper, the implementation did inherit all features from HNSW). We found that the edge selection method is vital for the trade-off of effectiveness and efficiency in searching. However, the existing edge selection techniques used in HNSW and ip-NSW are actually designed for metric distances, which are inapplicable for the non-metric measure, e.g., inner product.

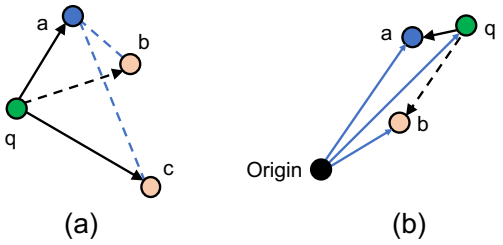


Figure 2: The example of edge selection used in constructing approximate Delaunay Graph. (a) the selection method for metric spaces used in HNSW and ip-NSW. $c$ is selected while $b$ is abandoned since it is not diverse from $a$. (b) the edge selection in IPDG. $b$ will be ignored because $a$ is a "super" point of it, which has been selected.

As shown in Figure 2 (a), the edge selection for metric spaces works as below: for each new inserting node (or edge updating node) $q$ and its nearest neighbor set (candidates) from Algorithm 2, a directed edge from $q$ to the nearest neighbor $a$ is constructed first. For other candidates, say $b$, the edge selection algorithm will check whether:

$$dis(q, b) < dis(a, b), \quad (6)$$

where $dis(\cdot, \cdot)$ is the distance of two vectors, such as $\ell_2$-distance or angular distance. If it is true, there will be an edge from $q$ to $b$, otherwise, $b$ will be abandoned in the selection. By this way, in a restricted degree, the new inserting node will have diverse outgoing neighbors. As shown in Figure 2 (a), $b$ is not selected while $c$ is selected.

It is obvious that the edge selection method for metric spaces is not suitable for inner product. As presented in Figure 2 (b), although $q^\top b > a^\top b$ (corresponding to $dis(q, b) < dis(a, b)$), $b$ should not be selected, since $a^\top b > b^\top b$ and for any query vector $q'$ with all positive elements, we have $q'^\top a > q'^\top b$. This means that $b$ is dispensable in the top-1 MIPS task and the edge from $q$ to $b$ should not be constructed. To solve this issue, we propose a new edge selection method by checking whether:

$$b^\top b > a^\top b. \quad (7)$$

If it is true, we will select $b$. Otherwise, we will skip $b$ since $a$ is a "super" point of $b$ and $b$ is dispensable. In this way, each inserting node will trend to connect with extreme points but not other short norm

vectors. The detailed algorithm is summarized in Algorithm 1 Lines $17-28$.

**Algorithm 1** IPDG Construction

1: **Input:** dataset $S$, the size of candidate size $N$, maximum outgoing degree of graph $M$.
2: Initialize graph $G = \emptyset$. $round = 0$
3: **while** $round < 2$ **do**
4: $round = round + 1$
5: **for** each $x$ in $S$ **do**
6: $A \leftarrow$ GREEDY_SEARCH$(x, G, N)$.
7: $B \leftarrow$ EDGE_SELECTION$(A, M)$.
8: Add edges $\overrightarrow{xy}$ to $G$ for every $y \in B$.
9: **for** each $y$ in B **do** ▷ Edge Updating
10: $C \leftarrow \{z \in S : \overrightarrow{yz} \in G\} \cup \{x\}$.
11: $D \leftarrow$ EDGE_SELECTION$(C, M)$.
12: Remove original outgoing edges of $y$, and add edges $\overrightarrow{yz}$ to $G$ for $z \in D$.
13: **end for**
14: **end for**
15: **end while**
16: **Output:** graph $G$.
17: **function** EDGE_SELECTION$(A, M)$
18: $B = \emptyset$.
19: **for** $y \in A$ **do**
20: **if** $y^\top y \geq \max_{z \in B} y^\top z$ **then**
21: $B = B \cup \{y\}$.
22: **end if**
23: **if** $|B| \geq M$ **then**
24: Break.
25: **end if**
26: **end for**
27: **Output:** $B$.
28: **end function**

### 3.2 Two-Round Construction

Based on the new edge selection method introduced above (and the reverse edge updating, see Algorithm 1 Lines $9-13$), nodes with larger norms will have higher probabilities to be selected as outgoing neighbors. So extreme points of the dataset will have more incoming edges and non-extremes points will more likely have no incoming edges in general. This is consistent with the true Delaunay Graphs in inner product space as previously shown in Figure 1.

However, at beginning of the graph construction, relatively "super" points are not true extreme points. Vectors coming in later may be better candidates (i.e., true extreme points). This issue will damage the overall graph quality and affect the final searching performance. A straightforward method may probably help: inserting data points with larger norms first. We tried this trick but it did not work well. The reason is that high norm points are not necessarily extreme points. Norms of extreme points for some Voronoi cells may be relatively small. The top large norm points may be just from one or a few Voronoi cells. In high dimensional data, it is difficult to find true extreme points. Alternatively, we design a two rounds construction algorithm to solve this issue and exploit the additional round construction to update edges, especially for nodes inserted in the beginning. In this way, the graph construction algorithm can detect extreme points automatically. We tried to conduct this two rounds construction method for ip-NSW too, but there are no significant improvements.

We share the graph construction algorithm for IPDG, including the edge selection function in Algorithm 1. After the graph being constructed, we perform MIPS via a greedy search algorithm presented in Algorithm 2. The greedy search algorithm is also used in the graph construction for candidates collecting.

**Algorithm 2** GREEDY_SEARCH$(q, G, N)$

1: **Input:** The query $q$, the index graph $G$, the size of candidate set $N$.
2: Randomly choose a node with outgoing edges, say $y$. $A \leftarrow \{y\}$. Mark $y$ as checked and the rest as unchecked. ▷
In the practical implementation, A is a priority queue for efficiency. We note A as a set here to simplify the expression.
3: **while** not all nodes in $G$ are checked **do**
4: $A \leftarrow A \cup \{z \in S : \overrightarrow{yz} \in G, y \in A, z \text{ unchecked}\}$
5: Mark nodes in $A$ as checked.
6: $A \leftarrow$ top $N$ candidates in $A \cup Z$ in descending order of inner product with $q$.
7: **if** $A$ does not update **then**
8: Break.
9: **end if**
10: **end while**
11: **Output:** $A$.

### 3.3 A Toy Example

To further explain the differences between the proposed method and previous state-of-the-art, ip-NSW, we conduct a case study on a toy example

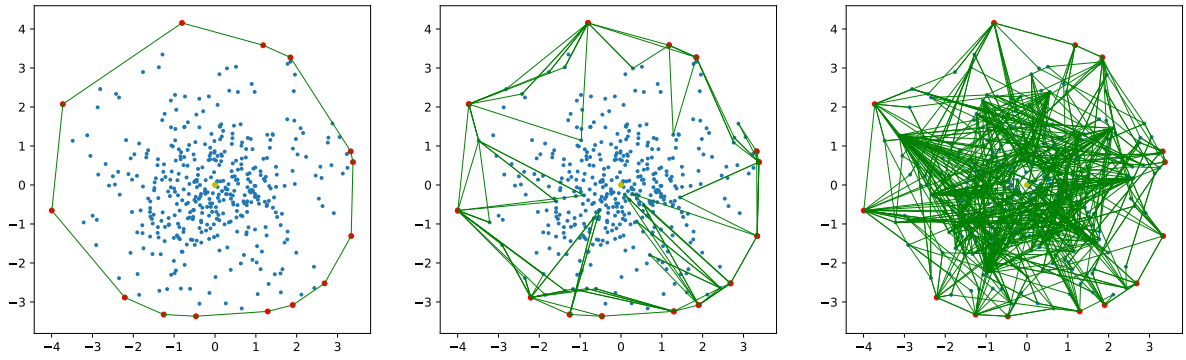

(a) (b) (c)

Figure 3: This is a toy example for approximate inner product Delaunay Graph construction (green lines are edges here) and red dots are $extreme$ points too. (a) is the true Delaunay Graph. (b) is an approximation by IPDG. (c) is built by ip-NSW. Note that IPDG and ip-NSW construct directed edges instead of undirected ones for the efficiency consideration. Only edges for nodes with incoming edges are shown in (b) and (c).

data which is shown in Figure 3. We randomly generate 400 two dimensional vectors following distribution Normal$(0, I_2)$. Figure 3 (a) shows the true Delaunay Graph for inner product. Red nodes correspond to extreme points of this dataset. Figure 3 (b) and (c) are graphs built by the proposed IPDG and ip-NSW, respectively. The parameter $N$ is set to 10 and $M$ is set to 2 for both algorithms in this study. Note that graphs built by IPDG and ip-NSW are directed graphs. To give better showing out, we only keep edges corresponding to nodes with incoming edges and other edges are ignored. Nodes without incoming edges will not be visited and do not affect the searching process, thus can be removed after the graph construction. As can be seen, the graph built by IPDG is more like the true Delaunay Graph and is more efficient for MIPS, while the graph built by ip-NSW have too many useless edges as shown in Figure 3 (c).

## 4 Experiments

In this section, we evaluate the proposed IPDG by comparing it with state-of-the-art MIPS methods.

### 4.1 Datasets

We used the following three pre-trained embeddings to investigate the performance of IPDG in MIPS for similar word searching. For each word embedding datasets, we random select 10000 vectors as queries and others as the base data.

**fastTextEn** and **fastTextFr** are 300 dimensional English and French word embeddings trained on Wikipedia using fastText (Joulin et al., 2016).

**GloVe50** are 50 dimensional word embeddings trained on Wikipedia2014 and Gigaword5 using GloVe (Pennington et al., 2014).

As most state-of-the-art MIPS algorithms evaluate their performance on recommendation datasets, we also benchmark IPDG on three recommendation datasets: Amazon Movie (**Amovie**), **Yelp** and **Netflix**. We use the Matrix Factorization (MF) method in (Hu et al., 2008) to obtain latent vectors of user and item. Then, in the retrieval process, user vectors are regarded as queries and the item vector that has the highest inner product score with each query should be returned by the MIPS algorithm.

| Datasets | Dimension | # Base Data |
|---|---|---|
| fastTextEn | 300 | 989873 |
| fastTextFr | 300 | 1142501 |
| GloVe | 50 | 1183514 |
| Amovie | 64 | 104708 |
| Yelp | 64 | 25815 |
| Netflix | 50 | 17770 |

Table 1: Statistics of the datasets.

Statistics of the six datasets are listed in Table 1. They vary in dimension (300, 64 and 50), sources (recommendation ratings, word documents) and extraction methods (fastText, GloVe and MF), which is sufficient for fair comparison. The ground truth is the top-1 nearest neighbor by inner product.

### 4.2 Baselines

In this paper, we compare IPDG with state-of-the-art MIPS methods. Firstly, reduction methods can be baselines. Some popular ANNS open source platforms utilize the reduction trick to solve MIPS, such as Annoy[1]. As introduced in Section 2.2, with reductions, any ANNS methods can be applied for

[1]https://github.com/spotify/annoy

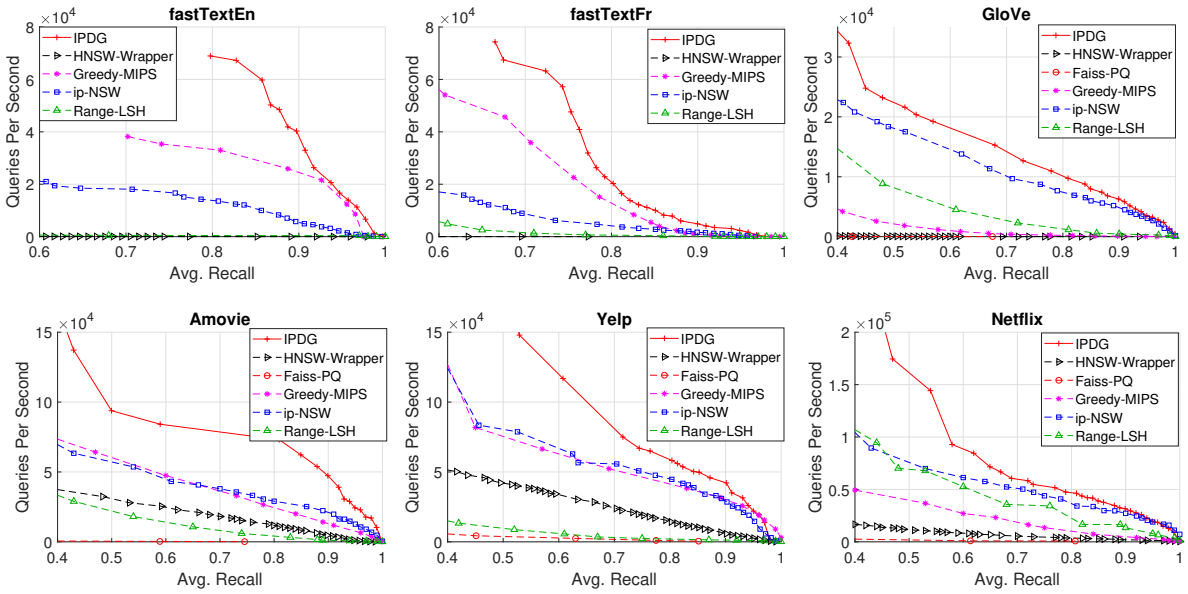


Figure 4: Recall vs. Time curves of for all methods in top-1 MIPS. Results for Faiss-PQ on fastTextEn and fastTextFr are not shown since they cannot produce recalls greater than 0.6. Best results are in upper right corners.

MIPS. In this line, we choose HNSW (Malkov and Yashunin, 2018) (referred to as **HNSW-Wrapper**) as the baseline and neglect other alternatives since HNSW is usually regarded as the most promising method for ANNS in metric spaces. We exploit the original implementation of HNSW[2] and add the wrapper introduced in Section 2.2.

**Range-LSH** (Yan et al., 2018) is also an reduction MIPS method and considers norm distribution of the data. The original implementation[3] is used.

**Faiss-PQ**[4] is a popular open source ANNS platform from Facebook, which is mainly implemented by Product Quantization (PQ) techniques. It contains MIPS as one component.

**Greedy-MIPS** is an MIPS algorithm from Yu et al. (2017). We use the original implementation[5].

**ip-NSW** is a state-of-the-art MIPS algorithm proposed in (Morozov and Babenko, 2018).[6].

### 4.3 Experimental Settings

There are two popular ways to evaluate ANNS/MIPS algorithms: i) **Recall vs. Time**; ii) **Recall vs. Computations**. Recall vs. Time reports the number of queries an algorithm can process per second at each recall level. Recall vs. Computations reports the amount/percentage of pairwise distance/similarity computations that the ANNS/MIPS algorithm costs at each recall level. Both evaluation indicators have their own pros and cons. Recall vs. Time is straightforward but it may introduce bias in implementation. Recall vs. Computations is beyond implementation but it does not consider the cost of different index structures. We will show both of these perspectives in the following experiments for the comprehensive evaluation.

All comparing methods have tunable parameters. In order to present a fair comparison, we vary all parameters over a fine grid for all methods. For each algorithm in each experiment, we will have multiple points scattered on the plane. To plot curves, we first find out the best result, $\max_x$, along with the x-axis (i.e., Recall). Then 100 buckets are produced by splitting the range from 0 to $max_x$ evenly. For each bucket, the best result along the y-axis (e.g., the biggest amount of queries per second or the lowest percentage of computations) is chosen. If there are no data points in the bucket, the bucket will be ignored. In this way, we shall have multiple pairs of data for drawing curves. All time-related experiments were performed on a 2X 3.00 GHz 8-core i7-5960X CPU server with 32GB memory.

### 4.4 Experimental Results

We first show experimental results for all comparison algorithms from the view of Recall vs. Time, which are shown in Figure 4. Overall, the proposed method IPDG performs consistently and significantly better than baselines on all six datasets. As

[2]https://github.com/nmslib
[3]https://github.com/xinyandai/similarity-search
[4]https://github.com/facebookresearch/faiss
[5]https://github.com/rofuyu/exp-gmips-nips17
[6]https://github.com/stanis-morozov/ip-nsw

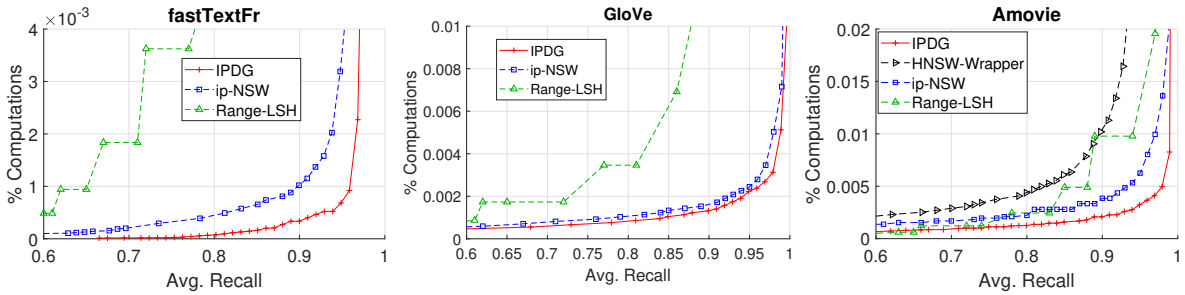


Figure 5: Recall vs. Computations curves in top-1 MIPS. Note that it is unable to show results for HNSW-Wrapper on fastTextFr and Glove in the showing scope. Best results are in lower right corners.

can be seen, some baselines show promising performance on partial datasets but they may work much worse on other datasets. For example, on lower dimensional datasets (i.e., the last four figures of Figure 4), ip-NSW work well but it fails on high dimensional datasets (i.e., fastTextEn and fastTextFr). Greedy-MIPS shows advantages on high dimensional datasets while becomes worse on some lower dimensional datasets, such as Netflix and GloVe. Among all methods, only IPDG works consistently well on all datasets which shows its effectiveness and robustness. Range-LSH performs badly in these experiments. The main reason is that Range-LSH does not have a good "budget" setting, similar to the budget in Greedy-MIPS and the $N_{search}$ parameter in graph-based methods. HNSW-Wrapper does not work comparably with IPDG either, especially on word embedding datasets. On some recall levels, say higher than 0.5, searching by HNSW-Wrapper is extremely slow (see the first three figures). It is clear that HNSW-Wrapper is far from state-of-the-art in challenging MIPS tasks, such as larger or higher dimensional vector datasets. The PQ based method, Faiss-PQ, works badly on all datasets since quantization codes can speedup the retrieval while may largely reduce the search performance, especially for the challenging top-1 MIPS problem. Note that results for Faiss-PQ on fastTextEn and fastTextFr are not shown in Figure 4 since they cannot produce recalls greater than 0.6.

We also show experimental results by Recall vs. Computations in Figure 5. Greedy-MIPS and Faiss-PQ cannot be evaluated from this view and the other four methods are explored here. Due to the limited space, only results on partial datasets are represented. As can be seen, only IPDG and ip-NSW work consistently well on all shown datasets. HNSW-Wrapper and Range-LSH work comparably with the other two methods on the recommendation dataset, Amovie, while performs much worse on the word embedding dataset, fastTextFr and GloVe. It is even unable to show the result for HNSW-Wrapper on fastTextFr and Glove in the showing scope. For IPDG and ip-NSW, they share similar index structures, it is fair to compare their computation amount for each query. To get a similar recall, IPDG requires much less inner product computation. For example, on fastTextFr, to reach the recall at 95%, ip-NSW requires about 0.3% computations while IPDG only needs 0.07% computations. This also demonstrates the efficiency of vector inner product retrieval by IPDG.

### 4.5 More Comparison with ip-NSW

| Datasets | ip-NSW | IPDG |
|---|---|---|
| fastTextEn | 144339 (14.6%) | 100138 (10.1%) |
| fastTextFr | 378875 (33.2%) | 250750 (21.9%) |
| GloVe | 622080 (52.6%) | 437378 (37.0%) |
| Amovie | 32434 (31.0%) | 12985 (12.4%) |
| Yelp | 5224 (20.2%) | 1871 (7.2%) |
| Netflix | 17154 (96.5%) | 14867 (83.7%) |

Table 2: Number and percentage of nodes with incoming edges for graphs built by ip-NSW and IPDG.

In this section, we will conduct a study by comparing the proposed IPDG and its closely related method ip-NSW on the index graph quality. The evaluation measure is the number of nodes with incoming edges. Intuitively, only extreme points of each dataset are useful for top-1 MIPS retrieval. Non-extreme points could be ignored in graph construction (i.e., without incoming edges so will not be visited in searching). Results for $N = 100$ and $M = 16$ are shown in Table 2. As can be seen, the graphs built by IPDG have much fewer nodes with incoming edges, which is consistent with the toy example introduced above. The reason can be

explained as below. The finely designed edge selection method in IPDG trends to select extreme points as outgoing neighbors for each newly inserted node or each edge updating node (see Algorithm 1 Lines $9-13$). Meanwhile, extreme points will have more opportunities to keep incoming edges in the edge updating and the second round graph construction. While non-extreme points will lose their incoming edges in these processes.

## 5 Conclusion and Future Work

Fast similarity search for data representations via inner product is a crucial and challenging task since it is one of the basic operations in machine learning algorithms and recommendation methods. To remedy this issue, we propose a search on graph method, namely Inner Product Delaunay Graph (IPDG), for Maximum Inner Product Search (MIPS) in embedded latent vectors. IPDG provides a better approximation to Delaunay Graphs for inner product than previous methods and is more efficient for the MIPS task. Experiments on extensive benchmarks demonstrate that IPDG outperforms previous state-of-the-art MIPS methods in retrieving latent vectors under inner product.

In this paper, we improve the top-1 MIPS performance by graph-based index. In the future, we will try to move the state-of-the-art frontier further, not only for top-1 MIPS but also for top-$n$, $n > 1$, MIPS results. Besides of metric measures (e.g., $\ell_2$-distance and cosine similarity) and inner product, more complicated measures has been studied, for example (Tan et al., 2019). It would be interesting to adopt these measures in NLP tasks. Another promising direction is to adopt a GPU-based system for fast ANNS or MIPS, which has been shown highly effective for generic ANNS tasks (Li et al., 2012; Johnson et al., 2017; Zhao et al., 2019). Developing GPU-based algorithms for MIPS is still a topic which has not been fully explored.

## Acknowledgement

The authors would like to sincerely thank the anonymous reviewers of NAACL 2019 and EMNLP 2019, for their helpful comments, which has improved the quality of this paper.

## References

Brice Acree, Eric Hansen, Joshua Jansa, and Kelsey Shoub. 2016. Comparing and evaluating cosine similarity scores, weighted cosine similarity scores and substring matching. Technical report.

Eneko Agirre, Mona Diab, Daniel Cer, and Aitor Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 385–393. Association for Computational Linguistics.

Franz Aurenhammer. 1991. Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, 23(3):345–405.

Yoram Bachrach, Yehuda Finkelstein, Ran Gilad-Bachrach, Liran Katzir, Noam Koenigstein, Nir Nice, and Ulrich Paquet. 2014. Speeding up the xbox recommender system using a euclidean transformation for inner-product spaces. In *Proceedings of the 8th ACM Conference on Recommender systems (RecSys)*, pages 257–264.

C Bradford Barber, David P Dobkin, and Hannu Huhdanpaa. 1996. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software (TOMS)*, 22(4):469–483.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3(Feb):1137–1155.

Paolo Cignoni, Claudio Montani, and Roberto Scopigno. 1998. Dewall: A fast divide and conquer delaunay triangulation algorithm in ed. *Computer-Aided Design*, 30(5):333–341.

Manaal Faruqui, Yulia Tsvetkov, Pushpendre Rastogi, and Chris Dyer. 2016. Problems with evaluation of word embeddings using word similarity tasks. *arXiv preprint arXiv:1605.02276*.

Jerome H. Friedman, F. Baskett, and L. Shustek. 1975. An algorithm for finding nearest neighbors. *IEEE Transactions on Computers*, 24:1000–1006.

Jerome H. Friedman, J. Bentley, and R. Finkel. 1977. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3:209–226.

Jianfeng Gao, Xiaodong He, Wen-tau Yih, and Li Deng. 2014. Learning continuous phrase representations for translation modeling. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 1, pages 699–709.

Hongyu Gong, Tarek Sakakini, Suma Bhat, and Jinjun Xiong. 2018. Document similarity for texts of varying lengths via hidden topics. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 1, pages 2341–2351.

Ruiqi Guo, Sanjiv Kumar, Krzysztof Choromanski, and David Simcha. 2016. Quantization based fast inner product search. In *Artificial Intelligence and Statistics (AISTATS)*, pages 482–490.

Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Proceedings of the Eighth IEEE International Conference on Data Mining (ICDM)*, pages 263–272.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM International Conference on Information Knowledge Management (CIKM)*, pages 2333–2338.

Qiang Huang, Guihong Ma, Jianlin Feng, Qiong Fang, and Anthony KH Tung. 2018. Accurate and fast asymmetric locality-sensitive hashing scheme for maximum inner product search. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, pages 1561–1570.

Piotr Indyk and Rajeev Motwani. 1998. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing (STOC)*, pages 604–613, Dallas, TX.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. 2016. Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.

Yannis Kalantidis and Yannis Avrithis. 2014. Locally optimized product quantization for approximate nearest neighbor search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2321–2328.

Maximilian Köper and Sabine Schulte im Walde. 2018. Analogies in complex verb meaning shifts: the effect of affect in semantic similarity models. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, volume 2, pages 150–156.

Moontae Lee, Xiaodong He, Wen-tau Yih, Jianfeng Gao, Li Deng, and Paul Smolensky. 2015. Reasoning in vector space: An exploratory study of question answering. *arXiv preprint arXiv:1511.06426*.

Omer Levy, Anders Søgaard, and Yoav Goldberg. 2017. A strong baseline for learning cross-lingual word embeddings from sentence alignments. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, volume 1, pages 765–774.

Ping Li, Anshumali Shrivastava, and Christian A. Konig. 2012. Gpu-based minwise hashing: Gpu-based minwise hashing. In *Proceedings of the 21st World Wide Web Conference (WWW)*, pages 565–566, Lyon, France.

Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. 2015. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 912–921.

Yury A Malkov and Dmitry A Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3111–3119.

Stanislav Morozov and Artem Babenko. 2018. Non-metric similarity graphs for maximum inner product search. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4722–4731.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Barbara Plank and Alessandro Moschitti. 2013. Embedding semantic similarity in tree kernels for domain adaptation of relation extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 1, pages 1498–1507.

Jan Wira Gotama Putra and Takenobu Tokunaga. 2017. Evaluating text coherence based on semantic similarity graph. In *Proceedings of TextGraphs-11: the Workshop on Graph-based Methods for Natural Language Processing*, pages 76–85.

Daniel Ramage, Anna N Rafferty, and Christopher D Manning. 2009. Random walks for text semantic similarity. In *Proceedings of the 2009 workshop on graph-based methods for natural language processing*, pages 23–31.

Mehran Sahami and Timothy D Heilman. 2006. A web-based kernel function for measuring the similarity of short text snippets. In *Proceedings of the 15th International Conference on World Wide Web (WWW)*, pages 377–386, Edinburgh, Scotland, UK.

Anshumali Shrivastava and Ping Li. 2014. Asymmetric LSH (ALSH) for sublinear time maximum inner product search (MIPS). In *Advances in Neural Information Processing Systems (NIPS)*, pages 2321–2329, Montreal, Canada.

Anshumali Shrivastava and Ping Li. 2015. Improved asymmetric locality sensitive hashing (ALSH) for maximum inner product search (MIPS). In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 812–821, Amsterdam, The Netherlands.

Gokavarapu Srinivas, Niket Tandon, and Vasudeva Varma. 2010. A weighted tag similarity measure based on a collaborative weight model. In *Proceedings of the 2nd International Workshop on Search and Mining User-Generated Contents*, pages 79–86.

Kohei Sugawara, Hayato Kobayashi, and Masajiro Iwasaki. 2016. On approximately searching for similar word embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 1, pages 2265–2275.

Md Arafat Sultan, Jordan Boyd-Graber, and Tamara Sumner. 2016. Bayesian supervised domain adaptation for short text similarity. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 927–936.

Yifan Sun, Nikhil Rao, and Weicong Ding. 2017. A simple approach to learn polysemous word embeddings. *arXiv preprint arXiv:1707.01793*.

Shulong Tan, Zhixin Zhou, Zhaozhuo Xu, and Ping Li. 2019. Fast Item Ranking under Neural Network based Measures. Technical report, Baidu Research.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5998–6008.

Xiang Wu, Ruiqi Guo, Ananda Theertha Suresh, Sanjiv Kumar, Daniel N Holtmann-Rice, David Simcha, and Felix Yu. 2017. Multiscale quantization for fast similarity search. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5745–5755.

Jun Xu, Xiangnan He, and Hang Li. 2018. Deep learning for matching in search and recommendation. In *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 1365–1368.

Xiao Yan, Jinfeng Li, Xinyan Dai, Hongzhi Chen, and James Cheng. 2018. Norm-ranging lsh for maximum inner product search. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2952–2961, Montreal, Canada.

Seunghyun Yoon, Pablo Estrada, and Kyomin Jung. 2017. Synonym discovery with etymology-based word embeddings. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–6.

Hsiang-Fu Yu, Cho-Jui Hsieh, Qi Lei, and Inderjit S Dhillon. 2017. A greedy approach for budgeted maximum inner product search. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5453–5462, Long Beach, CA.

Weijie Zhao, Shulong Tan, and Ping Li. 2019. SONG: Approximate Nearest Neighbor Search on GPU. Technical report, Baidu Research.

# Multiplex Word Embeddings for Selectional Preference Acquisition

**Hongming Zhang**[♣]*, **Jiaxin Bai**[♣]*, **Yan Song**[♠], **Kun Xu**[♡],
**Changlong Yu**[♣], **Yangqiu Song**[♣], **Wilfred Ng**[♣], **and Dong Yu**[♡]
[♣]Department of CSE, The Hong Kong University of Science and Technology
[♠]Sinovation Ventures
[♡]Tencent AI Lab
{hzhangal, jbai, cyuaq, yqsong, wilfred}@cse.ust.hk
clksong@gmail.com, {kxkunxu, dyu}@tencent.com

## Abstract

Conventional word embeddings represent words with fixed vectors, which are usually trained based on co-occurrence patterns among words. In doing so, however, the power of such representations is limited, where the same word might be functionalized separately under different syntactic relations. To address this limitation, one solution is to incorporate relational dependencies of different words into their embeddings. Therefore, in this paper, we propose a multiplex word embedding model, which can be easily extended according to various relations among words. As a result, each word has a center embedding to represent its overall semantics, and several relational embeddings to represent its relational dependencies. Compared to existing models, our model can effectively distinguish words with respect to different relations without introducing unnecessary sparseness. Moreover, to accommodate various relations, we use a small dimension for relational embeddings and our model is able to keep their effectiveness. Experiments on selectional preference acquisition and word similarity demonstrate the effectiveness of the proposed model, and a further study of scalability also proves that our embeddings only need 1/20 of the original embedding size to achieve better performance.

## 1 Introduction

Representing words as distributed representations is an important way for machines to process lexical semantics, which attracts much attention in natural language processing (NLP) in the past few years (Mikolov et al., 2013; Pennington et al., 2014; Song et al., 2017, 2018b; Song and Shi, 2018) with respect to its usefulness in many downstream tasks, e.g., parsing (Chen and Manning, 2014), machine translation (Zou et al., 2013), coreference resolution (Lee et al., 2018), etc. Conventional word embeddings, e.g., word2vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014), leverage the co-occurrence information among words to train a unified embedding for each word. Such models are popular and the resulting embeddings are widely used owing to their effectiveness and simplicity. However, these embeddings are not helpful for scenarios requiring words functionalizing separately under different situations, where selectional preference (SP) (Wilks, 1975) is a typical scenario.

In general, SP refers to that, given a word (predicate) and a dependency relation, human beings have certain preferences for the words (arguments) connecting to it. Such preferences are usually carried in dependency syntactic relations, for example, the verb 'sing' has plausible object words 'song' or 'rhythm' rather than other nouns such as 'house' or 'potato'. With such characteristic, SP is proven to be important in natural language understanding for many cases and widely applied over a variety of NLP tasks, e.g., sense disambiguation (Resnik, 1997), semantic role classification (Zapirain et al., 2013), and coreference resolution (Hobbs, 1978; Zhang et al., 2019b,c), etc.

Conventional SP acquisition methods are either based on counting (Resnik, 1997) or complex neural network (de Cruys, 2014), and the SP knowledge acquired in either way can not be directly leveraged into downstream tasks. On the other hand, the information captured by word embeddings can be seamlessly used in downstream tasks, which makes embedding a potential solution for the aforementioned problem. However, conventional word embeddings using one unified embedding for each word are not able to distinguish different relations types (such as various syntactic relations, which is crucial for SP) among words. For example, such embeddings

*Equal contribution.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5247–5256,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

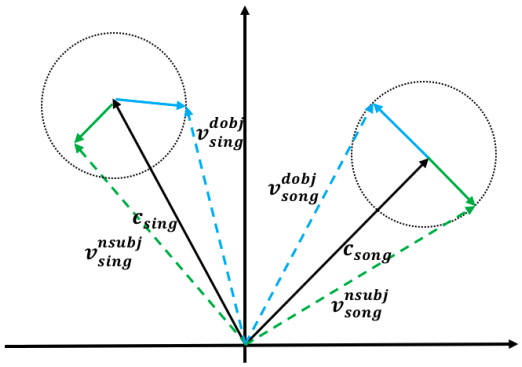


Figure 1: Illustration of the multiplex embeddings for 'sing' and 'song'. Black arrows present center embeddings for words' overall semantics; blue and green arrows refer to words' relational embeddings for relation-dependent semantics. All relational embeddings for each word are designed to near its center embedding. *nsubj* and *dobj* relations are used as examples.

treat 'food' and 'eat' as highly relevant words but never distinguish the function of 'food' to be a subject or an object to 'eat'. To address this problem, the dependency-based embedding model (Levy and Goldberg, 2014) is proposed to treat a word through separate ones, e.g., 'food@dobj' and 'food@nsubj', under different syntactic relations, with the skip-gram (Mikolov et al., 2013) model being used to train the final embeddings. However, this method is limited in two aspects. First, sparseness is introduced because each word is treated as two irrelevant ones (e.g., 'food@dobj' and 'food@nsubj'), so that the overall quality of learned embeddings is affected. Second, the resulting embedding size is too large[1], which is not appropriate either for storage or usage.

Therefore, in this paper, we propose a multiplex word embedding (MWE) model, which can be easily extended to various relations between two words. A multiplex network embedding model was originally proposed for modeling multiple relations among people in a social network (Zhang et al., 2018). Interestingly, we found it also useful in capturing various relations among different words. One example is shown in Figure 1. 'sing' and 'song' are highly related to each other with the 'predicate-objective' rather than the 'predicate-subject' relation. In our model, each word has a group of embeddings, including a center embedding representing its general semantics, and several embeddings representing their relation-dependent semantics. To ensure that the embeddings of the same word (under different relations) are similar to each other, we limit the Euclidean norm of relation-dependent embeddings within a small range, as shown in Figure 1. Moreover, considering that there could be many relations among words, if using a conventional dimension setting for embeddings to encode relations, the overall embedding size would be too big to be used in downstream tasks. To deal with it, we propose to use a small dimension for relation-dependent embeddings and use a transformation matrix for each relation to project them into the same space of the center embeddings. Thus the two types of embeddings can be jointly trained and the quality of the relation-dependent ones are guaranteed.

Experiments are conducted on SP acquisition over different dependency relations to evaluate whether the learned embeddings effectively capture words' semantics over these relations. In addition, the word similarity measurement is used to assess how well words' general semantics are learned in our model. Both evaluations confirm the superiority of our model, where the SP information is effectively preserved and the words' overall semantics are enhanced. Particularly, further analysis also indicates that our MWE embeddings are more powerful than all existing embedding methods in SP acquisition with around 1/20 in their size comparing to previous embeddings (Levy and Goldberg, 2014). All code and resulting embeddings are available at: https://github.com/HKUST-KnowComp/MWE.

## 2 Multiplex Word Embeddings

### 2.1 Model Overview

As introduced in Section 1, encoding selectional preference information into embeddings can be conducted by modeling word association patterns under different dependency relations. Similar to (Levy and Goldberg, 2014), the proposed MWE model also distinguishes different relations among words and learns separate embeddings for them on each dependency edge.

Formally, let $\mathcal{W}$ be the vocabulary set containing $n$ words $w_1, w_2, ..., w_n$ and $\mathcal{R}$ the relations set containing $m$ relations $r_1, r_2, ..., r_m$, the proposed model is expected to produce $m + 1$ embeddings for each word, where there are $m$ relational embeddings representing relations and a center em-

[1] Assuming one has 200,000 words in the vocabulary and 20 dependency relations, and follow conventional approaches (Mikolov et al., 2013; Pennington et al., 2014) to set the embedding dimension to 300, the resulting embedding size will be about 10 Gigabytes.

bedding $\mathbf{c}$ for the general semantics. Particularly, each relational embedding has an overall and a local version, denoted as $\mathbf{v}$ and $\mathbf{u}$, where $\mathbf{u}$ records the difference between $\mathbf{v}$ and $\mathbf{c}$. For both the relational embeddings ($\mathbf{v}$ and $\mathbf{u}$) and the center embedding, we use a similar way as that in word2vec (Mikolov et al., 2013) that each one has two variants to represent the semantics of a word $w$ being the head or tail in different relations. We denote the head and tail embeddings as $\mathbf{u}^r_{h,w} \in R^s$ and $\mathbf{u}^r_{t,w} \in R^s$ for the local embeddings of word $w$ under relation type $r$ and $\mathbf{c}_{w,h} \in R^d$ and $\mathbf{c}_{w,t} \in R^d$ for the center embedding, respectively, where $s$ is the dimension for $\mathbf{u}$ and $d$ the dimension for $\mathbf{c}$.

Although learning on similar information as that in (Levy and Goldberg, 2014), to reduce the sparseness of introducing $m$ relational embeddings for each word, we do not treat each word as multiple separate prototypes. Instead, we use its center embedding to transfer information among different relations and the sum of the center embedding and a local embedding to represent the final embedding for the corresponding relation[2]. Moreover, considering there are various relations among words, we use a lower dimension $s$ for storage compression, with $s < d$. Thus, a transformation matrix $\mathbf{X}$ is introduced to transform $\mathbf{u}$ into the same vector space of $\mathbf{c}$. As a result, the final embeddings $\mathbf{v}$ for head and tail of $w$ under relation $r$ are formulated as:

$$\begin{aligned} \mathbf{v}^r_{h,w} &= \mathbf{c}_{h,w} + \mathbf{X}^{rT}_h \mathbf{u}^r_{h,w}, \\ \mathbf{v}^r_{t,w} &= \mathbf{c}_{t,w} + \mathbf{X}^{rT}_t \mathbf{u}^r_{t,w}. \end{aligned} \tag{1}$$

## 2.2 Learning the MWE model

To train $\mathbf{v}^r_{h,w}$ and $\mathbf{v}^r_{t,w}$ for each $w$ under $r$, we adopt the negative sampling strategy to conduct the learning process. Specifically, for each $r$, we use a relation tuple set $\mathcal{T}_r$, with each tuple $t = (w_h, r, w_t) \in \mathcal{T}_r$, where $w_h$ is the head word and $w_t$ the tail word. For each $t$, we randomly generate two negative tuples by replacing $w_h$ and $w_t$ with the randomly selected fake head $w'_h$ and tail $w'_t$ respectively. Then the learning process is expected to distinguish the positive tuple against the negative ones. Therefore, formally, we maximize

$$\mathcal{J} = \frac{1}{|\mathcal{T}_r|} \sum_{t \in \mathcal{T}_r} log \frac{e^{f(w_h,r,w_t)}}{e^{f(w_h,r,w_t)} + e^{f(w_h,r,w'_t)} + e^{f(w'_h,r,w_t)}}, \tag{2}$$

over all tuples in $\mathcal{T}_r$ for each $r$, with $f(\cdot)$ evaluating $w_h$ and $w_t$ being positive samples under $r$ through

$$f(w_h, r, w_t) = {\mathbf{v}^r_{h,w_h}}^\top \cdot \mathbf{v}^r_{t,w_t}. \tag{3}$$

For each ($w_h$, $w_t$), we use cross entropy as the loss function

$$\begin{aligned} E = &- \log \sigma({\mathbf{v}^r_{h,w_h}}^\top \cdot \mathbf{v}^r_{t,w_t}) - \log \sigma(-{\mathbf{v}^r_{h,w_h}}^\top \cdot \mathbf{v}^r_{t,w'_t}) \\ &- \log \sigma(-{\mathbf{v}^r_{h,w'_h}}^\top \cdot \mathbf{v}^r_{t,w_t}), \end{aligned} \tag{4}$$

to measure the learning effect for Eq. (2), with $\sigma$ denoting the sigmoid function.

Combined with Eq. (1), the training process is thus to update $c$, $u^r$, and $X^r$ with the gradients passed from the loss function using stochastic gradient descent (SGD).

In detail, take $w_h$ as an example, its center embedding $\mathbf{c}_{h,w_h}$ is updated by

$$\begin{aligned} \mathbf{c}_{h,w_h} &= \mathbf{c}_{h,w_h} - \lambda \cdot \eta \cdot \frac{\partial E}{\partial \mathbf{v}^r_{h,w_h}} \\ &= \mathbf{c}_{h,w_h} - \lambda \cdot \eta \cdot e(w_h, w_t) \cdot \mathbf{v}^r_{t,w_t}, \end{aligned} \tag{5}$$

where $e(w_h, w_t) = \sigma({\mathbf{e}^r_{h,w_h}}^\top \cdot \mathbf{e}^r_{t,w_t}) - t_k$, with $t_k = 1$ if ($w_h$, $w_t$) is positive sample and $t_k = 0$ for negative ones. $\eta$ is the discounting learning rate. $\lambda$ is an alternating weight to control the contribution of the gradient, ranging from 0 to 1.

Moreover, $X^r$ and $\mathbf{u}^r$ are updated as follows:

$$\begin{aligned} \mathbf{u}^r_{h,w_h} &= \mathbf{u}^r_{h,w_h} - (1-\lambda) \cdot \eta \cdot \frac{\partial E}{\partial \mathbf{v}^r_{h,w_h}} \cdot \frac{\partial \mathbf{X}^r_h \mathbf{u}^r_{h,w_h}}{\partial \mathbf{u}^r_{h,w_h}} \\ &= \mathbf{u}^r_{h,w_h} - (1-\lambda) \cdot \eta \cdot e(w_h, w_t) \cdot \mathbf{X}^r_h \cdot \mathbf{v}^r_{t,w_t}, \end{aligned} \tag{6}$$

$$\begin{aligned} \mathbf{X}^r_h &= \mathbf{X}^r_h - (1-\lambda) \cdot \eta \cdot \frac{\partial E}{\partial \mathbf{v}^r_{h,w_h}} \cdot \frac{\partial \mathbf{X}^r_h \mathbf{u}^r_{h,w_h}}{\partial \mathbf{X}^r_h} \\ &= \mathbf{X}^r_h - (1-\lambda) \cdot \eta \cdot e(w_h, w_t) \cdot \mathbf{u}^r_{h,w_h} \cdot \mathbf{v}^r_{t,w_t}, \end{aligned} \tag{7}$$

Meanwhile, $\mathbf{c}_{t,w_t}$, $\mathbf{u}^r_{t,w_t}$, and $\mathbf{X}^r_t$ are updated in the same way of $\mathbf{c}_{h,w_h}$, $\mathbf{u}^r_{h,w_h}$, and $\mathbf{X}^r_h$ following Eqs. (5), (6), and (7).

The above learning process is conducted on all $m$ relations at the same time. During the training, the Euclidean norm of $\mathbf{X}^T\mathbf{u}$ for all embeddings is constrained to have a maximum semantic

[2] Conceptually shown in Figure 1, to ensure the resulting final embeddings not too far away from the center embeddings, we add restriction to the Euclidean norm of the local relational embeddings.

**Algorithm 1** Multiplex Word Embedding

**Input:** Relation specific tuple sets $\mathcal{T}_1, \ldots, \mathcal{T}_m$, $d$, $s$, $a$, and $\eta$.
**Output:** $\mathbf{c}_w$, $\mathbf{u}_w^r$, and $\mathbf{X}^r$.
Initialize $\mathbf{c}_w$, $\mathbf{u}_w^i$ and $\mathbf{X}^i$ randomly.
**for** *Each Iteration k* **do**
  Update $\lambda$.
  **for** *Each tuple* $t_p = (w_h, r, w_t) \in \mathcal{T}_r$ **do**
    Randomly generate the two negative examples $t_{n1} = (w'_h, r, w_t)$ and $t_{n2} = (w_h, r, w'_t)$.
    $\mathcal{T}_{tmp} = [t_p, t_{n1}, t_{n2}]$
    **for** $t \in \mathcal{T}_{tmp}$ **do**
      $e(w_h, w_t) = \sigma({\mathbf{v}'_{h,w_h}}^T \cdot \mathbf{v}^i_{t,n_t}) - t_k$
      Update $\mathbf{c}_{h,w_h}$, $\mathbf{u}^r_{h,w_h}$, $\mathbf{X}^r_h$, $\mathbf{c}_{t,w_t}$, $\mathbf{u}^r_{t,w_t}$, and $\mathbf{X}^r_t$ based on Eqs. (5), 6), and (7).
      **if** $\|{\mathbf{X}^r_h}^T \mathbf{u}^r_{h,w_h}\| > a$ **then**
        Update $\mathbf{X}^r_h$ and $\mathbf{u}^r_{h,w_h}$.
      **end**
      **if** $\|{\mathbf{X}^r_t}^T \mathbf{u}^r_{t,w_t}\| > a$ **then**
        Update $\mathbf{X}^r_t$ and $\mathbf{u}^r_{t,w_t}$.
      **end**
    **end**
  **end**
**end**

drifting range $a$, whose value controls the distance between the local relational embeddings and the center embedding. Specifically, if $\|\mathbf{X}^T\mathbf{u}\|$ equals to $a'$, which is greater than $a$, we scale down $\mathbf{X}^T$ and $\mathbf{u}$ with $\sqrt{a' \cdot \frac{1}{ka}}$, where $k$ is a scaling parameter set to 0.8 throughout this work.

The learning processing is summarized in Algorithm 1. For complexity analysis, it is obvious drawn that $m$ relation types and $n$ words result in $O(mn)$ and $O((d + s * m) * n)$ for time and space complexities, respectively.

### 2.3 The Alternating Optimization Strategy

To balance the learning process between the overall semantics and the relation-specific information, we adopt an alternating optimization strategy[3] (Bezdek and Hathaway, 2003) to adjust $\lambda$ based on different stage of the training instead of using a fixed weight for $\lambda$. Specifically, considering $\mathbf{c}$ is more reliable and possesses more information while $\mathbf{u}$ is learned with shared $\mathbf{c}$, we alternatively update $\mathbf{c}$ and $\mathbf{u}$ upon the convergence of $\mathbf{c}$. As a result, we set $\lambda$ to 1 in the first half of the training process and 0 afterwards.

[3] Alternating optimization was originally proposed to train different parameters in a sequential manner and applied in various areas.

## 3 Experiments

Experiments are conducted to evaluate how our embeddings are performed on SP acquisition and word similarity measurement.

### 3.1 Implementation Details

We use the English Wikipedia[4] as the training corpus. The Stanford parser[5] is used to obtain dependency relations among words. For the fair comparison, we follow existing work and set $d = 300$, $s = 10$, and $a = 1$. Following (Keller and Lapata, 2003) and (de Cruys, 2014), we select three dependency relations (*nsubj*, *dobj*, and *amod*) as follows:

- *nsubj*: The preference of subject for a given verb. For example, it is plausible to say 'dog barks' rather than 'stone barks'. The verb is viewed as the predicate (head) while the subject as the argument (tail).
- *dobj*: The preference of object for a given verb. For example, it is plausible for 'eat food' rather than 'eat house'. The verb is viewed as the predicate (head) while the object as the argument (tail).
- *amod*: The preference of modifier for a given noun. For example, it is plausible to say 'fresh air' rather than 'solid air'. The noun is viewed as the predicate (head) while the adjective as the argument (tail).

### 3.2 Baselines

We first compare the proposed multiplex embedding with the following embedding models. As it is trivial to apply these embedding models in downstream tasks, we label these models as downstream friendly.

- **word2vec**, the embedding model proposed by Mikolov et al. (2013). We use the skip-gram model for this baseline.
- **GloVe** (Pennington et al., 2014), learning word embeddings by matrix decomposition on word co-occurrences.

[4] https://dumps.wikipedia.org/enwiki/
[5] https://stanfordnlp.github.io/CoreNLP/

| Model | Downstream | Keller | | | SP-10K | | | |
|---|---|---|---|---|---|---|---|---|
| | | dobj | amod | average | nsubj | dobj | amod | average |
| word2vec | Friendly | 0.29 | 0.28 | 0.29 | 0.32 | 0.53 | 0.62 | 0.49 |
| GloVe | Friendly | 0.37 | 0.32 | 0.35 | 0.57 | 0.60 | 0.68 | 0.62 |
| D-embeddings | Friendly | 0.19 | 0.22 | 0.21 | 0.66 | 0.71 | 0.77 | 0.71 |
| ELMo | Friendly | 0.23 | 0.06 | 0.15 | 0.09 | 0.29 | 0.38 | 0.25 |
| BERT (static) | Friendly | 0.11 | 0.05 | 0.08 | 0.25 | 0.32 | 0.27 | 0.28 |
| BERT (dynamic) | Friendly | 0.19 | 0.23 | 0.21 | 0.35 | 0.45 | 0.51 | 0.41 |
| PP | Unfriendly | **0.66** | 0.26 | 0.46 | 0.75 | 0.74 | 0.75 | 0.75 |
| DS | Unfriendly | 0.53 | 0.32 | 0.43 | 0.59 | 0.65 | 0.67 | 0.64 |
| NN | Unfriendly | 0.16 | 0.13 | 0.15 | 0.70 | 0.68 | 0.68 | 0.69 |
| MWE | Friendly | 0.63 | **0.43**$^{\dagger}$ | **0.53**$^{\dagger}$ | **0.76** | **0.79**$^{\dagger}$ | **0.78** | **0.78**$^{\dagger}$ |

Table 1: Results on different SP acquisition evaluation sets. As Keller is created based on the PP distribution and has relatively small size while SP-10K is created based on random sampling and has a much larger size, we treat the performance on SP-10K as the main evaluation metric. Spearman's correlation between predicated plausibility and annotations are reported. The best performing models are denoted with bold font. † indicates statistical significant ($p < 0.005$) overall baseline methods.

- **D-embeddings**, the model proposed by Levy and Goldberg (2014) uses a skip-gram framework to encode dependencies into embeddings with multi-prototypes of words.

To investigate whether the SP knowledge can be captured by the pretrained contextualized word embedding models, we also treat following pretrained models as baselines.

- **ELMo** (Peters et al., 2018), a pretrained language model with contextual awareness. We use its static representations of words as the word embedding.
- **BERT** (Devlin et al., 2019), a pretrained bi-directional contextualized word embedding model with state-of-the-art performance on many NLP tasks.

Besides those embedding methods, we also compare with following conventional SP acquisition methods to demonstrate the effectiveness of the proposed multiplex embedding model, as it is still unclear how to leverage these methods in downstream tasks, we label these methods as downstream unfriendly.

- **Posterior Probability (PP)** (Resnik, 1997), a counting based method for the selectional preference acquisition task.
- **Distributional Similarity (DS)** (Erk et al., 2010), a method that uses the similarity of the embedding of the target argument and average embedding of observed golden arguments in the corpus to predict the preference strength.
- **Neural Network (NN)** (de Cruys, 2014), an NN-based method for the SP acquisition task. This model achieves the state-of-the-art performance on the pseudo-disambiguation task.

| SP Evaluation Set | #W | #P |
|---|---|---|
| Keller (Keller and Lapata, 2003) | 571 | 360 |
| SP-10K (Zhang et al., 2019a) | 2,500 | 6,000 |

Table 2: Statistics of Human-labeled SP Evaluation Sets. #W and #P indicate the numbers of words and pairs, respectively. As different datasets have different SP relations, we only report statistics about 'nsubj', 'dobj', and 'amod' (if available).

For word2Vec and GloVe, we use their released code. For D-embedding, we follow their original paper using the Gensim package[6]. The dimensions of the aforementioned embeddings are set to 300 according to their original settings. For ELMo and BERT, we use their pre-trained models. As BERT was not originally designed for word level semantics tasks, for the selectional preference acquisition task, we compare with two variations of the original BERT model: (1) BERT (static), which extracts static embedding from the BERT model in a similar way as that from the ELMo model; (2) BERT (dynamic), which takes a pair of

[6]https://radimrehurek.com/gensim/models/word2vec.html

| Model | | noun | verb | adjective | overall |
|---|---|---|---|---|---|
| word2vec | | 0.41 | 0.28 | 0.44 | 0.38 |
| Glove | | 0.40 | 0.22 | 0.53 | 0.37 |
| D-embedding | | 0.41 | 0.27 | 0.38 | 0.36 |
| nsubj | h | 0.46 | 0.29 | 0.54 | 0.43 |
| | t | 0.45 | 0.25 | 0.48 | 0.40 |
| | h+t | 0.44 | 0.23 | 0.50 | 0.40 |
| | [h,t] | 0.47 | 0.27 | 0.51 | 0.42 |
| dobj | h | 0.46 | 0.27 | 0.45 | 0.41 |
| | t | 0.45 | 0.23 | 0.46 | 0.40 |
| | h+t | 0.45 | 0.20 | 0.45 | 0.38 |
| | [h,t] | 0.46 | 0.25 | 0.48 | 0.42 |
| amod | h | 0.47 | 0.25 | 0.52 | 0.37 |
| | t | 0.46 | 0.24 | 0.50 | 0.38 |
| | h+t | 0.46 | 0.24 | 0.52 | 0.38 |
| | [h,t] | 0.47 | 0.26 | 0.52 | 0.38 |
| center | h | 0.51 | **0.33** | **0.57** | **0.48** |
| | t | 0.51 | 0.30 | 0.56 | 0.47 |
| | h+t | **0.52** | 0.31 | 0.54 | 0.46 |
| | [h,t] | 0.51 | 0.32 | 0.57 | **0.48** |

Table 3: Spearman's correlation of different embeddings for the WS measurement. 'nsubj', 'dobj', 'amod' represents the embeddings of the corresponding relation and 'center' indicates the center embeddings. h, t, h+t, and [h,t] refer to the head, tail, sum of two embeddings, and the concatenation of them, respectively. The best scores are marked in bold fonts.

words as input and produces a plausibility score for that pairs of words. The main difference between the static and dynamic version of BERT is that in the dynamic version, the contextual information can be fully utilized, which is more similar to the training objective of BERT.

### 3.3 Selectional Preference Acquisition

The first task in our experiment is SP acquisition. Currently, there are two methods that we can use to evaluate the quality of extracted SP knowledge: Pseudo-disambiguation (Ritter et al., 2010) and human labeled datasets (McRae et al., 1998; Keller and Lapata, 2003). However, as shown in (Zhang et al., 2019a), pseudo-disambiguation selects positive SP tuples from the corpus and generate negative SP tuples by randomly replacing the head/tail. As a result, pseudo-disambiguation only evaluates how good the model fits the corpus rather than evaluating the SP acquisition models based on ground truth. Thus, in this section, we evaluate different SP acquisition methods with ground truth (human labeled datasets). Two representative datasets, Keller (Keller and Lapata, 2003) and SP-10K (Zhang et al., 2019a), are selected as the benchmark datasets.

In each dataset, for each SP relation, various word pairs are provided. For each of the word pairs, the datasets also provide the annotated plausibility score of how likely a preference exists between that word pair under the corresponding SP relation. Thus the job for all the models is to predict the plausibility score for all word pairs under different SP relation settings. After that, we follow the conventional setting that uses the Spearman's correlation to assess the correspondence between the predicted plausibility scores and human annotations on all word pairs for all SP relations. The statistics of these datasets are shown in Table 2.

| Model | WS | Dimension | Training Time |
|---|---|---|---|
| ELMo | 0.434 | 512 | ≈40 |
| BERT | 0.486 | 768 | ≈300 |
| MWE | 0.476 | 300 | 4.17 |

Table 4: Comparison of MWE against language models on the WS task. Overall performance, embedding dimension, and training time (days) on a single GPU are reported.

For embedding based methods (word2vec, GloVe, D-embedding[7], and MWE[8]), we follow previous work (Mikolov et al., 2013; Levy and Goldberg, 2014) and use the cosine similarity between embeddings of head and tail words to predict their relations. For conventional SP acquisition methods (PP, DS, and NN), we follow their original paper to compute the plausibility scores.

The experimental results are shown in Table 1. As Keller is created based on the PP distribution and have relatively small size while SP-10K is created based on random sampling and has a much larger size, we treat the performance on SP-10K as the major evaluation. Our embeddings significantly outperform other baselines, especially embedding based baselines. The only exception is PP on the Keller dataset due to its biased distribution. In addition, there are other interesting observations. First, compared with 'dobj' and 'nsubj', 'amod' is simpler for word2vec and GloVe. The reason behind is that conventional embeddings only capture the co-occurrence information, which is enough to predict the selectional preference of

[7]For D-embedding, since it provides embeddings for different relations (e.g., 'food@nsubj', 'food@dobj), we follow their work and directly use embeddings over certain relations to predict the score for the tested pairs. For example, given the test tuple ('eat', *dobj*, 'food'), we compute the cosine similarity of 'eat' and 'food@dobj' rather than 'food@subj'.

[8]For the proposed model MWE, we use the cosine similarity between $v_{h,w_h}$ and $v_{t,w_t}$ as the predicted plausibility.

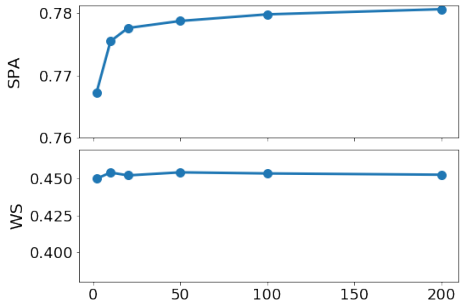


Figure 2: Effect of the different $s$ on SP acquisition (SP-10K) and WS tasks.

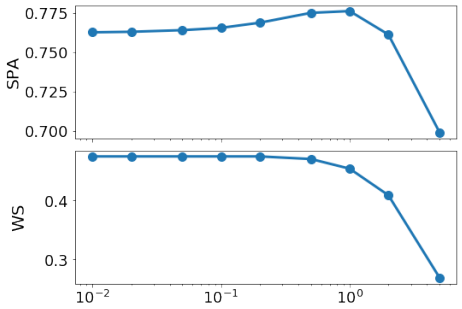


Figure 3: Effect of different $a$ on SP acquisition (SP-10K) and WS tasks.

'amod' rather than 'nsubj' or 'dobj'[9]. Second, even though large-scale contextualized word embedding models like ELMo and BERT have been proved useful in many other tasks, they are still limited in learning specific and detailed semantics and thus perform inferior to our model in the SP acquisition task. For Example, ELMo and BERT can know that there is a strong semantic connection between 'eat' and 'food', but they do not know whether 'food' is a plausible subject or object of 'eat'. Third, surprisingly, although NN achieves the state-of-the-art performance on the pseudo-disambiguation task, its performance is not satisfying against human annotation, especially on Keller, which is probably because the NN model overfits the training data, whose distribution is different from human SP knowledge.

### 3.4 Word Similarity Measurement

In addition to SP acquisition, we also evaluate our embeddings on word similarity (WS) measurement to test whether the learned embedding can effectively capture the overall semantics. We use SimLex-999 (Hill et al., 2015) as the evaluation dataset for this task because it contains different word types, i.e., 666 noun pairs, 222 verb pairs, and 111 adjective pairs. We follow the conventional setting that uses the Spearman's correlation to assess the correspondence between the similarity scores and human annotations on all word pairs. Evaluations are conducted on the final embeddings $\mathbf{v}$ for each relation and the center ones.

Results are reported in Table 3 with several observations. First, our model achieves the best overall performance and significantly better on nouns, which can be explained by that nouns appear in all three relations while most of the verbs and adjectives only appear in one or two relations. This result is promising since it is analyzed by Solovyev et al. (2017) that two-thirds of the frequent words are nouns; thus there are potential benefits if our embeddings are used in downstream NLP tasks. Second, the center embeddings achieve the best performance against all the other relation-dependent embeddings, which demonstrates the effectiveness of our model in learning relation-dependent information over words and also enhancing their overall semantics.

We also compare MWE with pre-trained contextualized word embedding models in Table 4 for this task, with overall performance, embedding dimensions, and training times reported. It is observed that that MWE outperforms ELMo and achieves comparable results with BERT with smaller embedding dimension and much less training complexities.

## 4 Analysis

With the experiments on SP acquisition and word similarity measurement, we conduct further analysis for different settings for hyper-parameters and learning strategies, as well as the scalability of our model.

### 4.1 Effect of the Local Embedding Dimension

We investigating the performance of MWE on SPA (SP acquisition) and WS tasks with different $s$. As shown in Figure 2, the performance of our model is in an increasing trend when we increase the value of $s$. When the dimension reaches 10, the performance almost reaches the top, which confirms that the local relational embeddings can be effective in small dimensions (about 1/30 of the center dimension) when using center embeddings as the constraint.

[9]The only possible SP relation between nouns and adjectives is 'amod', while multiple SP relations could exist between nouns and verbs, and co_occurrence information cannot effectively distinguish them.

| Training Strategy | Averaged SPA | Overall WS |
|---|---|---|
| $\lambda = 1$ | 0.762 | 0.476 |
| $\lambda = 0$ | 0.073 | 0.018 |
| $\lambda = 0.5$ | 0.493 | 0.323 |
| Alternating optimization | 0.775 | 0.476 |

Table 5: Comparisons of different training strategies.

### 4.2 Effect of the Semantic Drifting Range

Similar to $s$, we also investigate the model performance with the influence of different constraint $a$. It is observed in Figure 3 that, the constraint gets looser with the increase of $a$. For SP acquisition, in a small range, the model performance gradually improves along with the increasing value of $a$ because these embeddings are more flexible to capture intense relation-dependent information. However, once the range passes a threshold (1 in our experiment), the embeddings get farther away from their general semantics and start to overfit, which is also observed in the WS experiment, and thus the performance drops monotonically.

### 4.3 Effect of Alternating Optimization

An analysis is also conducted to demonstrate the effectiveness of the alternating optimization strategy. As shown in Table 5, we compare our model with several different strategies. The first one is to put all weights to the center embedding (fix $\lambda$ to 1), which never updates the local relational embeddings. As a result, it can achieve similar performance on word similarity measurement but is inferior in SP acquisition because no relation-dependent information is preserved. The second strategy is to put all weights to the local relational embeddings (fix $\lambda$ to 0). In this case, it performs very poor owing to the loss of general semantics. Last but not least, the alternating optimization also outperforms the setting with a fixed $\lambda = 0.5$, which can be explained by alternating optimization can first get a good overall semantic representation and then acquire the SP knowledge on top of that. To summarize, the alternating optimization provides an effective solution to training our model with a smart process in adjusting the contribution of different embeddings as well as stabilizing their optimization.

### 4.4 Scalability

Scalability of embeddings is an important factor in real applications. Practically, GPU or similar computation support is required to accommodate embeddings to applying neural models. Normally, the current most affordable GPUs typically have a memory size limitation around 10 GB. Thus, to ensure the learned embeddings can be used in downstream tasks, their size becomes an important concern when evaluating different embedding models, especially when there exist many different relations between words in our model.

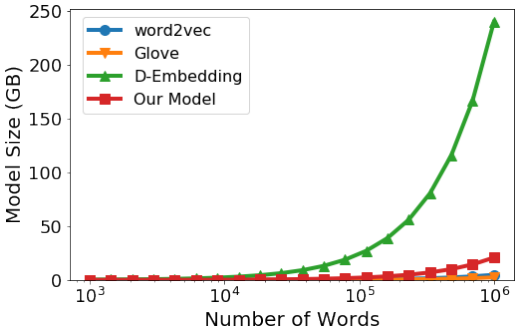


Figure 4: Scalability on embedding size against number of words.

Assuming that there are 20 relations among words, the size of all embedding models with the increasing word numbers in Figure 4. word2Vec and GloVe have the smallest size because they only train one embedding for each word while sacrificing the relation-dependent information among them. As a comparison, D-embeddings are trained on multiple prototypes for each word to record relation information, thus their size dramatically explodes with the increasing of the vocabulary. Consequently, it is easily computed for D-embeddings that 200,000 words will result in a 10GB model, which is unfeasible to be used in downstream tasks. Effectively with the small dimension for our local relational embeddings, relation information can be preserved in a small-sized model, which shows a compatible space requirement with the conventional embeddings.

## 5 Related Work

Learning word embeddings has become an important research topic in NLP (Bengio et al., 2003; Turney and Pantel, 2010; Collobert et al., 2011; Song and Shi, 2018), with the capability of embeddings demonstrated in different languages (Song et al., 2018a) and tasks such as parsing (Chen and Manning, 2014), machine translation (Zou et al., 2013), coreference resolution (Lee et al., 2018), etc. Conventional word embeddings (Mikolov et al., 2013; Pennington et al., 2014) often leverage

word co-occurrence patterns, resulting in a major limitation that they coalesce different relationships between words into a single vector space. To address this limitation, dependency-based embedding model (Levy and Goldberg, 2014) was proposed to represent each word with several separate embeddings, and then suffers from its sparseness and the huge size of the resulting embeddings. Alternatively, our MWE model uses a set of (constrained and small) embeddings for each word to encode its general and relation-dependent semantics to resolve the sparseness and the size problem.

Another line of work related to this paper is the research on SP, which is considered important in language understanding and has been proved helpful in various downstream tasks (Tang et al., 2016; Resnik, 1997; Hobbs, 1978; Inoue et al., 2016; Zapirain et al., 2013). Several studies attempted to acquire SP automatically from raw corpora (Resnik, 1997; Rooth et al., 1999; Erk et al., 2010; Santus et al., 2017). However, they are designed specifically for SP acquisition and are limited in leveraging SP knowledge for downstream tasks. Compared to them, the learned embeddings from our model are useful resources that incorporate SP knowledge and can be seamlessly leveraged in other NLP models.

## 6 Conclusion

In this paper, we present a multiplex word embedding model encoding selectional preference information from word dependency relations. Different from conventional embeddings, for each word, we proposed to use a set of embeddings to represent its general (center embedding) and relation-dependent (local relational embedding) semantics, where their combination present the final embedding for each word under particular relations. Experiments on SP acquisition and word similarity measurement illustrated that our model better encodes SP knowledge than different baselines without harming its capability in representing general semantics. Analysis is also conducted with respect to different settings and optimization strategies, as well as the effect of model size, which further illustrates the validity and effectiveness of the proposed model and its learning process.

## Acknowledgements

This paper was supported by the Early Career Scheme (ECS, No. 26206717) from Research Grants Council in Hong Kong and the Tencent AI Lab Rhino-Bird Focused Research Program.

## References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

James C Bezdek and Richard J Hathaway. 2003. Convergence of alternating optimization. *Neural, Parallel & Scientific Computations*, 11(4):351–368.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of EMNLP*, pages 740–750.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Tim Van de Cruys. 2014. A neural network approach to selectional preference acquisition. In *Proceedings of EMNLP*, pages 26–35.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Katrin Erk, Sebastian Padó, and Ulrike Padó. 2010. A flexible, corpus-driven model of regular and inverse selectional preferences. *Computational Linguistics*, 36(4):723–763.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Jerry R Hobbs. 1978. Resolving pronoun references. *Lingua*, 44(4):311–338.

Naoya Inoue, Yuichiroh Matsubayashi, Masayuki Ono, Naoaki Okazaki, and Kentaro Inui. 2016. Modeling context-sensitive selectional preference with distributed representations. In *Proceedings of COLING*, pages 2829–2838.

Frank Keller and Mirella Lapata. 2003. Using the web to obtain frequencies for unseen bigrams. *Computational Linguistics*, 29(3):459–484.

Kenton Lee, Luheng He, and Luke Zettlemoyer. 2018. Higher-order coreference resolution with coarse-to-fine inference. In *Proceedings of NAACL-HLT*, pages 687–692.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of ACL*, pages 302–308.

Ken McRae, Michael J Spivey-Knowlton, and Michael K Tanenhaus. 1998. Modeling the influence of thematic fit (and other constraints) in on-line sentence comprehension. *Journal of Memory and Language*, 38(3):283–312.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, pages 3111–3119.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237.

Philip Resnik. 1997. Selectional preference and sense disambiguation. *Tagging Text with Lexical Semantics: Why, What, and How?*

Alan Ritter, Mausam, and Oren Etzioni. 2010. A latent dirichlet allocation method for selectional preferences. In *Proceedings of ACL*, pages 424–434.

Mats Rooth, Stefan Riezler, Detlef Prescher, Glenn Carroll, and Franz Beil. 1999. Inducing a semantically annotated lexicon via em-based clustering. In *Proceedings of ACL*, pages 104–111.

Enrico Santus, Emmanuele Chersoni, Alessandro Lenci, and Philippe Blache. 2017. Measuring thematic fit with distributional feature overlap. In *Proceedings of EMNLP*, pages 648–658.

Valery D Solovyev, Vladimir V Bochkarev, and Anna V Shevlyakova. 2017. Dynamics of core of language vocabulary. *arXiv preprint:1705.10112*.

Yan Song, Chia-Jung Lee, and Fei Xia. 2017. Learning word representations with regularization from prior knowledge. In *Proceedings of CoNLL*, pages 143–152.

Yan Song and Shuming Shi. 2018. Complementary learning of word embeddings. In *Proceedings of IJCAI*, pages 4368–4374.

Yan Song, Shuming Shi, and Jing Li. 2018a. Joint learning embeddings for chinese words and their components via ladder structured networks. In *Proceedings of IJCAI*, pages 4375–4381.

Yan Song, Shuming Shi, Jing Li, and Haisong Zhang. 2018b. Directional skip-gram: Explicitly distinguishing left and right context for word embeddings. In *Proceedings of NAACL-HLT*, pages 175–180.

Haiqing Tang, Deyi Xiong, Min Zhang, and Zhengxian Gong. 2016. Improving statistical machine translation with selectional preferences. In *Proceedings of COLING*, pages 2154–2163.

Peter D. Turney and Patrick Pantel. 2010. From Frequency to Meaning: Vector Space Models of Semantics. *Journal of Artificial Intelligence Research*, 37(1):141–188.

Yorick Wilks. 1975. A preferential, pattern-seeking, semantics for natural language inference. *Artificial intelligence*, 6(1):53–74.

Beñat Zapirain, Eneko Agirre, Lluís Màrquez, and Mihai Surdeanu. 2013. Selectional preferences for semantic role classification. *Computational Linguistics*, 39(3):631–663.

Hongming Zhang, Hantian Ding, and Yangqiu Song. 2019a. SP-10K: A large-scale evaluation set for selectional preference acquisition. In *Proceedings of ACL*, pages 722–731.

Hongming Zhang, Liwei Qiu, Lingling Yi, and Yangqiu Song. 2018. Scalable multiplex network embedding. In *Proceedings of IJCAI*, pages 3082–3088.

Hongming Zhang, Yan Song, and Yangqiu Song. 2019b. Incorporating context and external knowledge for pronoun coreference resolution. In *Proceedings of NAACL-HLT*, pages 872–881.

Hongming Zhang, Yan Song, Yangqiu Song, and Dong Yu. 2019c. Knowledge-aware pronoun coreference resolution. In *Proceedings of ACL*, pages 867–876.

Will Y Zou, Richard Socher, Daniel Cer, and Christopher D Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of EMNLP*, pages 1393–1398.

# MulCode: A Multiplicative Multi-way Model for Compressing Neural Language Model

**Yukun Ma**[1]*, **Pei-Hung(Patrick) Chen**[2*], **Cho-Jui Hsieh**[2]
[1]AIR Labs, Continental Automotive Group, Singapore [2]Department of Computer Science, UCLA

## Abstract

It is challenging to deploy deep neural nets on memory-constrained devices due to the explosion of numbers of parameters. Especially, the input embedding layer and Softmax layer usually dominate the memory usage in an RNN-based language model. For example, input embedding and Softmax matrices in IWSLT-2014 German-to-English data set account for more than 80% of the total model parameters. To compress these embedding layers, we propose MulCode, a novel multi-way multiplicative neural compressor. MulCode learns an adaptively created matrix and its multiplicative compositions. Together with a prior weighted loss, MulCode is more effective than the state-of-the-art compression methods. On the IWSLT-2014 machine translation data set, MulCode achieved 17 times compression rate for the embedding and Softmax matrices, and when combined with quantization technique, our method can achieve 41.38 times compression rate with very little loss in performance.

## 1 Introduction

Deep neural language models with a large number of parameters have achieved great successes in facilitating a wide range of Natural Language Processing (NLP) applications. However, the overwhelming size of parameters used by these models stands out as the main obstacle hindering their deployment on memory-constrained devices. Given the over-parameterization of deep neural nets, the effective compression of them has been receiving increasing attention from the research community. The neural language models typically consist of three major components: the recurrent layers (e.g., a LSTM cell), an embedding layer for representing input tokens, and a Softmax layer for generating output tokens. The dimension of recurrent layers is typically small and independent of the size of input/output vocabulary. In contrast, the embedding layer and Softmax layer use a dense layer to map the vocabulary to a lower dimensional space, which grows linearly with the vocabulary size and contributes mostly to memory consumption. For example, the One Billion Word language modeling task (Chelba et al., 2013) has a vocabulary size around 800k, and more than 90% of the memory usage is used to store the input and Softmax matrices. Compressing the word embedding, therefore, is the key to reducing the memory usage of deep neural language models.

There exist several studies for compressing neural language models. In particular, quantization (Lin et al., 2016; Hubara et al., 2016) and group-wise low rank approximation (Chen et al., 2018a) achieve the state-of-the-art performance on tasks like language modeling and machine translation. However, these methods are rather computationally inspired than being motivated by intuitions of the semantic composition and thus suffer from poor interpretability. In contrast, deep compositional models (Shu and Nakayama, 2017; Chen et al., 2018b) have been explored to represent the original word vector with a set of discrete pseudo words and thus decompose the word vector to the addition of the corresponding pseudo word vectors learned in an end-to-end manner. These methods are seen as embracing the long-existing yet still, the most popular assumption (Foltz et al., 1998) on compositionality of semantics which states that the meaning of an element (e.g., a word, phrase, or sentence) can be obtained by taking the addition of its constituent parts.

Despite it's more semantically meaningful, current deep compositional models have several limitations. First, additive composition might turn to be counter-intuitive for that the addition does not bring new meaning beyond the individual element to the constructed element (Pinker, 2003; Mitchell and Lapata, 2009, 2008). Besides, exist-

* The two authors contribute equally



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5257–5266, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

ing deep code compressor is found hard to compress the output Softmax layer for that the conflicting codes (i.e. different words having same encoding) might make the language model less discriminative. Last, it is well-known that the distribution of word frequencies can be approximated by a power law. Defining importance of words in terms of frequencies in the corpus is also verified in (Chen et al., 2018a). However, current deep compositional frameworks encode every word equally without considering the frequency information.

In this paper, we propose MulCode, a novel and effective deep neural compressor to address the above issues. MulCode uses a multiplicative composition instead of addition. The multiplicative factors introduce a larger capacity empowering the encoding of more complicated semantic. From the perspective of semantic composition, this allows introducing new information to the base codebook vectors (sub-elements to compose a word vector), taking frequency information of each token into consideration. Technically, MulCode embraces the research line of dense matrix decomposition that has been investigated in many applications (Memisevic, 2011; Novikov et al., 2015). It trains the composition with weighted loss proportional to the frequency of each word. MulCode also adopts an adaptive way of constructing codebooks based on the frequency of each word.

MulCode outperforms state-of-the-arts methods on compressing language models and neural machine translation models. For example, on One-Billion-Word data set, MulCode achieves 18.2 times compression rate on embedding and Softmax matrices without losing much of performance. On IWSLT-14 DE-EN machine translation task, MulCode achieves 17 times compression with BLEU score difference smaller than one percent. Results can be further improved to 41.38 times compression rate when combined with quantization technique.

## 2 Related Work

### 2.1 Model Compression for Convolutional Neural Network

**Low-rank Approximation** The original word embedding vectors can be viewed as a large matrix with the rows being the words and columns being the vector values. One natural way to approximate it is to obtain the low-rank approximation of the matrix using SVD. Based on this idea, (Sainath et al., 2013) compressed the fully connected layers in neural nets. For convolution layers, (Jaderberg et al., 2014; Denton et al., 2014) applied higher-order tensor decomposition to compress CNN. Similarly, (Howard et al., 2017) developed another structural approximation. (Kim et al., 2016) proposed an algorithm to select rank for each layer. (Yu et al., 2017) reconstructed the weight matrices by using sparse plus low-rank approximation.

**Pruning** Many algorithms have been proposed to remove those unimportant weights in deep neural nets. To do this, one needs to define the importance of each weight. (LeCun et al., 1990) showed that the importance can be estimated by using the Hessian of loss function. (Han et al., 2015b) considered adding $\ell_1$ or $\ell_2$ regularization and applied iterative thresholding approaches to achieve very good compression rates. Later on, (Han et al., 2015a) demonstrated that CNNs can be compressed by combining pruning, weight sharing and quantization.

**Quantization** Using lower precision representations to store parameters has been used for model compression. (Hubara et al., 2016) showed that a simple uniform quantization scheme effectively reduces both the model size and the prediction time of a deep neural net. (Lin et al., 2016) showed that non-uniform quantization can further improve the performance. Recently, several advanced quantization techniques have been proposed for CNN compression (Xu et al., 2018; Choi et al., 2018).

### 2.2 Model Compression for Neural Language Models

Despite model compression has been studied extensively for CNN models, fewer work have focused on the compression for deep neural nets for NLP applications. In fact, directly applying popular approaches to NLP problems does not provide a satisfactory result. (Hubara et al., 2016) showed that the naive quantization can only achieve less than 3X compression rate on PTB data with no performance loss. (Lobacheva et al., 2017) showed that for word-level LSTM models, the pruning approach can only achieve 4 times compression with more than 5% performance loss. Most recently, (Chen et al., 2018a) proposed a decomposition scheme based on block-wise low

rank approximation of embedding matrix. Although this method achieves competitive empirical results, the learned model does not have a strong semantic interpretation.

**Additive Composition** One of the most popular assumptions about the composition of semantics is called the additive composition stating that the meaning of a unit (e.g., word, phrase, or sentences) can be obtained by summing up the meaning of its constituents. At the word level, a word might be decomposed into a set of subword units. For example, "disagree" = "dis"+"agree". Alternatively, a word can also be represented by a set of relevant words, e.g., "king" = "man" + "crown". (Gittens et al., 2017) has validated this particular assumption for a popular word embedding approach (i.e., Skip-Gram (Mikolov et al., 2013)).

Based on this assumption, (Chen et al., 2016) created a codebook by splitting the vocabulary into two disjoint sets based on the word frequency: the most frequent words and the rest. Less frequent words are represented using a sparse linear combination of the vectors of more frequent words. Instead of using an explicit set of words, (Shu and Nakayama, 2017) designed the codebooks in a more data-driven fashion where the selection of pseudo words and code vectors are learned automatically using the Gumbel-Softmax trick (Jang et al., 2016). Following the same approach, (Chen et al., 2018b) propose additional training objective to integrate the learning of discrete codes with the training of the language model. Our method, based on the multiplicative composition rather than addition, follows this research line. The effectiveness of multiplicative composition is verified in some language modeling tasks (Mitchell and Lapata, 2009, 2008).

## 3 Proposed Method

### 3.1 Multi-way Multiplicative Codes

Compositional models start with defining a set of $d_m$ dimensional vectors, which serve as basic codes to compose the targeted embedding matrix. These vectors could further be separated into $M$ groups and say each group contains $K$ vectors. We call each group a codebook, and each $d_m$ dimensional vector in the codebook a codeword.

An 1-way codebook then is defined as a $R^{1\times M\times K\times d_m}$ tensor. Correspondingly, we define $N$-way codebooks as $U \in R^{N\times M\times K\times d_m}$, where each of the $N$ ways consists of a set of $M$ codebooks, each codebook contains $K$ words, and each codeword is associated with a $d_m$ dimensional vector representing its semantics.

Since $U$ is a high-order tensor which is not memory-friendly, we model the composition of $U$ as applying a customized multiplicative operator on two tensors $C \in R^{M\times K\times d_m}$ and $S \in R^{N\times M\times d_m}$ as

$$U = C \odot S,$$

where $\odot$ is a multiplicative operator defining that

$$U_{i,j,k} = C_{j,k} \circ \sigma(S_{i,j}),$$
$$\forall i \in [1,N], j \in [1,M], \text{and } k \in [1,K],$$

$\sigma(\cdot)$ stands for the tangent hyperbolic function, and $\circ$ is the Hadamard product (entry-wise product). Tensor $C$ is referred to as the base codebook, consisting of $M$ codebooks where each codebook contains $K$ codeword vectors of $d_m$ dimensions. $S$ is called rescaling codebooks. Each codeword in the base codebook is injected with new meanings by a code vector in the rescaling codebook.

Despite rescaling codebook uses much less memory (O($NMd_m$)) than simply creating a $N$ times larger set of vectors (O($NMKd_m$)), using rescaling codebook still introduces additional costs of the memory. We propose to further reduce the cost by allowing rescaling code vectors to be shared. We replace $S$ with a $\tilde{S} \in R^{N\times d_m}$. That is, for each of the $N$ way codebook, we use only one $d_m$-dimensional vector to rescale the base codebook $C$. The number of parameters in $S$ can thus be reduced and the computation of $U$ becomes

$$U_{i,j,k} = C_{j,k} \circ \sigma(\tilde{S}_i),$$
$$\forall i \in [1,N], j \in [1,M], \text{and } k \in [1,K].$$

Now, with the $N$-way codebook defined by $C$ and $\tilde{S}$, given an embedding matrix $E$ with $V$ vocabularies (i.e. $E \in R^{V\times d_m}$), we could represent $E$ by finding an encoding $Q \in R^{V\times N\times M}$ to compose $E$ from $C$ and $\tilde{S}$. $Q_i$ contains encoding of corresponding vocabulary $V_i$. From each of $N$-way n, and each of $M$ codebooks m, $Q_{i,n,m}$ indicates which codeword to compose. Let the word vector for the $i$th word be $e_i$. We could construct $e_i$ by

$$e_i \approx \hat{e}_i = \sum_{n=1}^{N}\sum_{m=1}^{M} U_{n,m,Q_{i,n,m}}$$
$$= \sum_{n=1}^{N}\sum_{m=1}^{M} C_{m,Q_{i,n,m}} \odot \sigma(\tilde{S}_n).$$

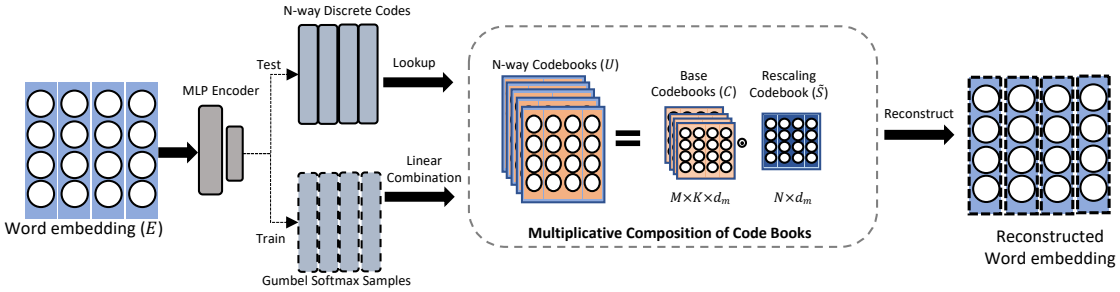


Figure 1: Overall architecture of the multi-way multiplicative compressor

The $N$-way discrete code could be learned in an end-to-end manner by using the Gumbel-Softmax trick (Jang et al., 2016). We first compute an encoding vector for the original word vector $e_i$ by feeding it to a neural network

$$a_i = Softmax(\sigma(W^\top \phi(W'^\top e_i + b) + b')),$$

where $W$, $W'$, $b$ and $b'$ are the parameters of the network, and $\phi$ is the softplus function. $a_i$ represents a real value tensor in a shape $N \times M \times K$, which is then fed to the Gumbel-Softmax to generate a continuous approximation of drawing discrete samples with respect to the last dimension

$$\hat{D}_i = Softmax((\log a_i + g)/\tau),$$

where $g$ is a random noise vector sampled from Gumbel distribution, and $\tau$ is the Softmax temperature controlling how close is the sample vector to a uniform distribution. $\hat{D}_i$ is an approximately one-hot out of $K$ drawing for each way $N$ and codebook $M$. We can then compute approximation of each dimension of $e_i$ as

$$\hat{e}_{i,d} = U_{:,:,:,d} \odot \hat{D}_i$$

Note that during training $\hat{D}_i$ is generated as a continuous approximation of the $N$-way discrete code. At the testing phase, the fixed encoding $Q_i$ of vector $e_i$ is directly computed as $Q_{i,n,m} = \arg\max_k a_{n,m,k}$. More details on Gumbel-trick could be found in (Jang et al., 2016).

### 3.2 Group Adaptive Coding

Given the rescaling codebooks $\tilde{S}$ is small, the memory consumption mainly consists of two parts: code vectors $C$ and discrete codes $Q$. On the one hand, the code vectors normally account for the major memory usage when dealing with relatively smaller vocabulary size. On the other hand, the size of discrete codes grows linearly with the size of vocabulary and the logarithm of $K$. To further reduce the memory usage, we propose to use codes of adaptive length and dimensions to deal with the linear dependency with vocabulary size. The intuition is to encode frequent words with a longer code length to achieve a relative lower reconstruction loss while representing rare words with fewer codes. To achieve this, we first sort the words according to their frequency and then split words into fixed number of groups $G$. The $i$th group could access only $\gamma_{G_i} = M \times (1 - \frac{i-1}{|G|})$ codebooks.

In the same time, we store the low-rank version of code vectors for some codebooks that were mainly used for representing rare words.

$$C_{i,j} = W_{G_i} c_i,$$

where $c_i \in R^{d_{G_i}}$ and $W_{G_i} \in R^{d_{G_i} \times d_m}$ is the linear transformation matrix to be shared by all the codebooks in the $i$th group. We resolve $d_{G_i}$ in an intuitive way as shown in Algorithm 1.

In practice, high frequency words tend to get accesses to codebooks with higher rank while fewer frequent words can only access codebooks of lower dimension. Thanks to the long tail of rare words, this could actually help save considerable memory space. Normally, using a low compression rate results in $d_{G_i} \ll \gamma_{G_i} \times K$ so that it is guaranteed to reduce the number of parameters in the base codebook.

### 3.3 Prior-weighted Reconstruction Loss

According to Zipf's law, the frequency of words conforms to a power law distribution. It means that word which falls to the long tail may rarely appear in the sentence. It motivates the modification on the learning objective so that the compressor will be more focused on words with high frequency. In this paper, we use the distribution of

Sort set G according to frequency;
$d$': the minimum dimension ;
$\delta$ : targeted compression rate;
$f_{G_i}$ : frequency of group $G_i$;
$o_{G_i}$ : bits consumed by discrete code matrix $Q$ for each word in $G_i$;
$n \leftarrow \delta \times V \times d_m \times 32(bits)$ ;
**for** $i \leftarrow 1$ *to* $|G|$ **do**
  compute a ratio based on frequency $\Delta_{G_i} \leftarrow \frac{f_{G_i}}{\sum_{i=1}^{|G|} f_{G_i}}$;
  calculate the dimension;
  $d_{G_i} \leftarrow (n - o_{G_i}) \times \Delta_{G_i} / (\gamma_{G_i} + d_m)$;
  ensure dimension is valid within $(d', d_m)$;
  $d_{G_i} \leftarrow \min(\max(d_{G_i}, d'), d_m)$;
  update $n$ by subtracting used bits;
  $n \leftarrow n - o_{G_i} - (\gamma_{G_i} - \gamma_{G_{i+1}} + d_m) \times d_{G_i}$
**end**

**Algorithm 1:** Algorithm to resolve the dimension of adaptive codebooks towards achieving a targeted compression rate.

the words in the training set as a prior knowledge to guide the learning of the compressor. In addition, similar to (Lin et al., 2017) we also want to let each of the $N$ way encoding focus on different aspects of the original word embedding. The proposed training objective function then becomes

$$\mathcal{L} = \sum_{i=1}^{V} (-\log \hat{p}_i \|\hat{e}_i - e_i\|_2^2 + \lambda \|\hat{v}_i \hat{v}_i^\top - I\|_2^2 + \lambda \|\hat{v}_i^\top \hat{v}_i - I\|_2^2),$$

where $\hat{p}_i$ represents the empirical distribution of word $V_i$ in the training set, $e_i$ is original word vector, $\hat{e}_i$ is the reconstructed vector and $\hat{v}_i \in R^{N \times d_m}$ is a compilation of reconstrcuted vectors from all $N$-way codebooks in a matrix form (i.e. $\hat{v}_{i,n} = \sum_{m=1}^{M} C_{m,Q_{i,n,m}} \odot \sigma(\tilde{S}_n)$ is the reconstructed vector from $nth$-way codebook). The new objective function thus focuses on high-frequency words and in the meanwhile allows the N-way coding to encode different information.

# 4 Experiment

## 4.1 Data Sets and Models

Following the experimental protocol of (Chen et al., 2018a), we evaluate our proposed method with two important NLP tasks: language modeling and machine translation. Table 1 summarizes the key characteristics of the four data sets

Table 1: Statistics of data sets. The number in parenthesis shows the ratio of embedding layers (input plus Softmax) respective to the entire model size

| Models | Vocab. Size | Dimension | Model Size |
|---|---|---|---|
| PTB-small | 10k | 200 | 17.7MB(85.8%) |
| PTB-large | 10k | 1500 | 251MB(45.4%) |
| OBW | 793k | 1024 | 6.8GB(91.2%) |
| NMT: DE-EN | 30k | 500 | 195MB(83.1%) |

and models. PTB-small is using 2-layer LSTM-based language model built on Penn Tree Bank (PTB) data set. The vocabulary size is 10k. Input embedding layer and output Softmax layer are both set to 200 dimensions. PTB-large is trained on the same PTB data set as PTB-small except the dimensions for input and output are increased to 1500. Neural machine translation (NMT: DE-EN) is a Seq2seq model initialized using Pytorch checkpoints provided by Open-NMT (Klein et al., 2017). The model is to perform German to English translation tasks on IWSLT-14 (Cettolo et al., 2014) data set. One billion words (OBW) uses a 2-layer LSTM (referred to as "2-LAYER LSTM-8192-1024" by (Jozefowicz et al., 2016)) trained on the OBW data set and the vocabulary size is 793,471. For LSTM-based language models, the input embedding matrix and Softmax embedding matrix account for the major memory usage (up to 91.2%). Therefore, we target compressing both the input and Softmax embedding matrices.

## 4.2 Implementation Details

We compress both input embedding and Softmax matrices. We trained MulCode by using Adam optimizer with learning rate 0.001. For PTB data set, we group the vocabulary using 3 groups and 8 groups for OBW. To resolve the dimension of codebooks for adaptive coding, we use targeted compression rate $\delta = 0.2$ for PTB-small and $\delta = 0.05$ for the rest three models[1].

After approximation, we retrain the rest of parameters by SGD optimizer with initial learning rate 0.01. Whenever the validation perplexity does not drop down, we decrease the learning rate to an order smaller. We did not include results of fine-tuning on OBW for that the re-training process takes too long (few days) which is not compliant with our motivation to compress the given pre-trained embeddings.

[1]The original language models can be downloaded from https://github.com/mayktian/MulCode.git

The compression rate and corresponding performance could certainly be plotted as a spectrum graph. The more we compress, the larger the performance drop. In this paper, as far as BLEU score is concerned, we report results of compressed models when the BLEU falls within 3 percent difference from the original score. For PTB data set, we target 3 percent drop of perplexity (PPL) after retrain. For OBW data set, since it has a larger vocabulary size, we report results within 10 percent difference from the PPL achieved by the uncompressed model. For each method we tested various parameters and report the smallest model size of the compression fulfilling above criteria.

Notice that some previous methods compress model directly during training phase (Khrulkov et al., 2019; Wen et al., 2017). In contrast, our problem setup follows (Chen et al., 2018a; Shu and Nakayama, 2017; Chen et al., 2018b) that given a pre-trained model, we want to compress the model with limited fine-tuning.

### 4.3 Comparison with Baseline Models

We refer to our proposed method as MulCode (Mul stands for both multi-way and multiplicative composition). We mainly compare with two state-of-the-art baseline compressors targeting compressing the embedding layer.

1. **GroupReduce** We refer to the results reported in (Chen et al., 2018a).

2. **DeepCode** The additive composition model by (Shu and Nakayama, 2017). We use the pytorch code[2] released by (Shu and Nakayama, 2017) to produce the results.

Table 2 summarizes the comparison between the proposed methods and state-of-the-art baselines for the four benchmark data sets and LSTM models. MulCode manages to compress the input embedding layer and Softmax embedding layer 6 to 18 times without suffering a significant loss in the performance.

In comparison, all the baseline models achieve much lower compression rate with PTB-small which has only 200 dimensions. It is reasonable since embedding layers of PTB-small contains less redundant information and thus can be hardly compressed. As compared with DeepCode, our method achieves much higher compression rate[3] for all the four models. Our method also consistently and significantly outperforms GroupReduce.

### 4.4 Comparison with Quantization

Quantization has been proven to be a strong baseline. In fact, the discrete coding of MulCode can be considered equivalent to a trainable quantization. On the other hand, we need to point out that quantization is not orthogonal to MulCode. MulCode could be combined with quantization to achieve better performance. Specifically, the $M \times K$ base codebooks as well as the rescaling codebook could be quantized to further reduce the memory usage. We summarize the results in Table 3. Quantized MulCode could achieve more than 30.8 times compression for both input embedding and Softmax matrix in OBW. In addition, on machine translation task, it achieves 41.38X with BLEU score drops around only 1% after retraining. In particular, we observe that the effect of retraining is more prominent for MulCode and simple quantization compared to GroupReduce. This implies that local precision lost for low-rank basis in GroupReduce is more difficult to be recovered. In contrast, the collective information of MulCode due to the compositional property is more robust when imprecise local vectors present.

### 4.5 Ablation Analysis

We summarize the ablation analysis in Table 4. The major technical features of MulCode are rescaling codebooks (RC), prior-weighted reconstruction loss (PRL), and the group adaptive coding (GAC). Besides, we also consider using full rescaling books (Full) in place of the one shared across $M$ codebooks. We remove or add these features one at a time. First, the rescaling codebook is shown contributing largely to the performance. On the other hand, since removing the rescaling codebook is equivalent to degrading to using pure additive composition, it also suggests the multiplicative composition is a better alternative to the additive composition. In the meantime, removing the rescaling codebook is equivalent to repeatedly selecting from the base codebook. It may force the codeword vector to encode the original meaning of

[2] https://github.com/zomux/neuralcompressor

[3] Aligned with what has been mentioned by (Shu and Nakayama, 2017) in their rebuttal, we found the DeepCode unable to work with the Softmax layer of OBW no matter how hard we have tuned it (99.5 as shown in Table 2).

Table 2: Compressed model evaluation with 3 language models and 1 machine translation model. Memory usage is reported as compared with the original input embedding and Softmax embedding. For example, 30x means the compressed embeddings use only 1/30 the memory space of the original embedding.

| Model | Metric | Original | GroupReduce | DeepCode | MulCode |
|---|---|---|---|---|---|
| PTB-small | Memory | 1X | 4X | 3.9X | 6.5X |
| | PPL(before retrain) | 112.28 | 115.38 | 120.05 | 115.38 |
| | PPL(after retrain) | - | 113.81 | 115.57 | 113.34 |
| PTB-large | Memory | 1X | 8X | 3.8X | 17.1X |
| | PPL(before retrain) | 78.32 | 84.79 | 84.73 | 83.85 |
| | PPL(after retrain) | - | 79.83 | 81.80 | 79.89 |
| NMT: DE-EN | Memory | 1X | 8X | 5.7X | 17X |
| | BLEU(before retrain) | 30.33 | 29.31 | 26.58 | 29.48 |
| | BLEU(after retrain) | - | 29.96 | 29.37 | 30.08 |
| OBW | Memory | 1X | 6.6X | 5.7X | 18.2X |
| | PPL | 31.04 | 32.47 | 99.59 | 32.66 |

Table 3: Results of compressing all input and Softmax embedding layers on three data sets. Experiments compare proposed Quantized MulCode with traditional Quantization and Quantized GroupReduce. 20x means approximated embedding uses 20 times smaller memory compared to original input embedding layer and Softmax layer..

| Model | Metric | Original | Quantization | Quantized GroupReduce | Quantized MulCode |
|---|---|---|---|---|---|
| PTB-small | Memory | 1X | 6.4X | 16X | 16.3X |
| | PPL(before retrain) | 112.28 | 115.81 | 116.54 | 116.33 |
| | PPL(after retrain) | - | 114.14 | 114.39 | 113.34 |
| PTB-large | Memory | 1X | 6.4X | 20X | 28.2X |
| | PPL(before retrain) | 78.32 | 81.69 | 81.53 | 81.55 |
| | PPL(after retrain) | - | 79.22 | 78.61 | 78.91 |
| NMT: ED-EN | Memory | 1X | 6.4X | 32X | 41.38X |
| | BLEU(before retrain) | 30.33 | 27.41 | 29.33 | 29.32 |
| | BLEU(after retrain) | - | 30.19 | 29.65 | 29.94 |
| OBW | Memory | 1X | 6.4X | 26X | 30.8X |
| | PPL(before retrain) | 31.04 | 32.63 | 34.43 | 34.36 |

words as a mix of different aspects of the semantics, which may turn out hard to be recovered. In fact, we find it difficult to train a model that can achieve low perplexity loss (33) on OBW with the rescaling codebooks removed.

Removing the prior-weighted loss from the model results in only marginal loss in the performance except for PTB-small. It indicates that, given the limited capacity of codebooks, the performance of compressed model could benefit more from lower reconstruction loss for those words with the highest frequencies. For example, MulCode achieves $3.5\%$ less perplexity loss compared to the model without using a prior-weighted reconstruction loss. Group adaptive coding is another key factor to reduce memory usage. It shows that it works well with all the four models. It handles the memory reduction in both ways: reducing the parameters in the code vectors and reducing the number of discrete codes to represent a word. For model with a very large vocabulary, it seems to achieve higher memory reduction than with a smaller model. We conjecture it is largely due to the prominent presence of rare words in data set like OBW. The significance of PRL and GAC verifies the importance of frequency in compressing natural words. Lastly, using the full tensor for rescaling codebook does not seem to be necessary. It fails to produce any improved performance of the compressed model at the expense of a non-trivial compression rate drop.

### 4.6 Selection of $M$,$N$ and $K$

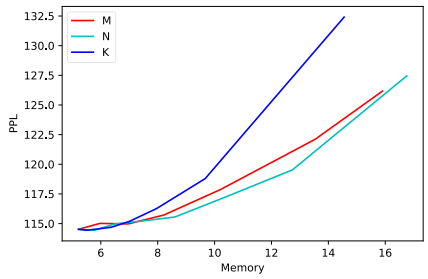


Figure 2: Influence of $M, N, K$ for PTB-small.

To understand how the choices of $M, N, K$ would affect the performance of the compressed model, we test the proposed model for PTB-small

Table 4: Ablation analysis of MulCode. We test the compression model by removing (or adding) one feature at a time. **-RC** stands for removing the rescaling codebook;**-PRL** stands for removing the prior-weighted reconstruction loss; **-GAC** means removing the group adaptive coding; **+ Full** means to use the untied tensor for rescaling codebook.

| Model | Metric | -RC | -PRL | -GAC | + Full | MulCode |
|---|---|---|---|---|---|---|
| PTB-small | Memory | 3.95X | 6.47X | 5.68X | 5.50X | 6.47X |
| | PPL | 124.78 | 115.58 | 115.65 | 115.96 | 115.38 |
| PTB-large | Memory | 5.64X | 15.42X | 13.08X | 10.20X | 17.1X |
| | PPL | 84.72 | 84.82 | 84.60 | 84.36 | 83.85 |
| NMT: DE-EN | Memory | 9.02X | 17.4X | 11.24X | 16X | 17.44X |
| | BLEU | 26.16 | 28.12 | 28.60 | 29.02 | 29.11 |
| OBW | Memory | 9.4X | 23.8X | 16.8X | 23.4X | 23.8X |
| | PPL | 37.42 | 34.05 | 33.45 | 32.81 | 32.95 |

with varying setting of $M$, $N$, and $K$. The departure point of this experiment is using $M = 32, K = 32, N = 8$. We then adjust the three parameters one at a time while fixing the other two. In order to plot the results in a single figure, we set the x-axis as the memory usage instead of different values of the three parameters. As shown in Figure 2, adjusting the values of $M$, $N$, and $K$ have similar effects on the PPL when the compression rate is low ($\leq$7X). With a larger compression rate, the performance becomes much more sensitive to the change of $K$. It suggests that it is safer to maintain a high value for $K$ while tuning the rest two parameters for the purpose of securing a reasonable performance of the compressed model.

### 4.7 Understanding Composed Codes

At the core of our approach is the multiplicative composition from two sets of codebooks. Hence, it is interesting to investigate what has been encoded in the $N$-way codebooks. One assumption is that each code in the base codebooks encodes a mix of information which can be disentangled by the rescaling codebook. We encode all the words of OBW corpus by a $32 \times 8 \times 4$ codes. We compute the hamming distance of example query words (shown in Table 5) for each of the $N$-way codes. We select the top ones with the smallest hamming distance from the 10,000 most frequent words that are likely to have low reconstruction errors.

Since the $N$-way codings are generated by selecting from the base codebooks and modified by rescaling codebooks, each channel can be seen as meaningful subspace. It shows that each of the $N$-way codings might have encoded a different subspace of the original meaning of words, including tenses (e.g., halt v.s. halted), plurals (e.g., bank v.s. banks), synonyms (e.g., soccer v.s. football),

| Word | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| tomorrow | today | Sunday | prompt | Tuesday |
| beautiful | elegant | iconic | great | fields |
| soccer | football | hockey | Ghana | basketball |
| where | when | experiencing | who | learn |
| bank | company | police | companies | banks |
| halt | stop | halted | afternoon | cover |
| like | just | Like | such | is |

Table 5: Most similar word computed using Hamming distance for each of the N-way codings.

co-occurrence (like v.s. just), topical relatedness (soccer v.s. hockey). It verifies that the multiplicative composition used in our approach is able to introduce new information to the base codebook.

## 5 Conclusion

In this paper, we propose a novel compression method for neural language models. Our method applies multiplicative factors on end-to-end learned codebooks. Our method also considers the frequency information in the corpus by adding weighted loss according to the importance. At the same time, the coding scheme is made adaptive based on the frequency information. Experimental results show that our method outperforms the state-of-the-art compression methods by a large margin. In particular, on the IWSLT-14 data set, our method combined with quantization achieves 41.38 times compression rate for both the embedding and Softmax matrices. It will facilitate deployment of large neural language models on memory-constrained devices.

## 6 Acknowledgements

Pei-Hung Chen and Cho-Jui Hsieh acknowledge the support from NSF via IIS1719097, Intel and Google Cloud.

## References


Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th iwslt evaluation campaign, iwslt 2014. In *Proceedings of the International Workshop on Spoken Language Translation, Hanoi, Vietnam*.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Patrick Chen, Si Si, Yang Li, Ciprian Chelba, and Cho-Jui Hsieh. 2018a. Groupreduce: Block-wise low-rank approximation for neural language model shrinking. In *Advances in Neural Information Processing Systems 31*, pages 10988–10998. Curran Associates, Inc.

Ting Chen, Martin Renqiang Min, and Yizhou Sun. 2018b. Learning k-way d-dimensional discrete codes for compact embedding representations. In *International Conference on Machine Learning*, pages 853–862.

Yunchuan Chen, Lili Mou, Yan Xu, Ge Li, and Zhi Jin. 2016. Compressing neural language models by sparse word representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 226–235.

Yoojin Choi, Mostafa El-Khamy, and Jungwon Lee. 2018. Universal deep neural network compression. *CoRR*, abs/1802.02271.

Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. 2014. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in neural information processing systems*, pages 1269–1277.

Peter W Foltz, Walter Kintsch, and Thomas K Landauer. 1998. The measurement of textual coherence with latent semantic analysis. *Discourse processes*, 25(2-3):285–307.

Alex Gittens, Dimitris Achlioptas, and Michael W Mahoney. 2017. Skip-gram- zipf+ uniform= vector additivity. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 69–76.

Song Han, Huizi Mao, and William J. Dally. 2015a. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *CoRR*, abs/1510.00149.

Song Han, Jeff Pool, John Tran, and William J. Dally. 2015b. Learning both weights and connections for efficient neural networks. *CoRR*, abs/1506.02626.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2016. Quantized neural networks: Training neural networks with low precision weights and activations. *arXiv preprint arXiv:1609.07061*.

Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. 2014. Speeding up convolutional neural networks with low rank expansions. *arXiv preprint arXiv:1405.3866*.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

Valentin Khrulkov, Oleksii Hrinchuk, Leyla Mirvakhabova, and Ivan Oseledets. 2019. Tensorized embedding layers for efficient model compression. *arXiv preprint arXiv:1901.10787*.

Yong-Deok Kim, Eunhyeok Park, Sungjoo Yoo, Taelim Choi, Lu Yang, and Dongjun Shin. 2016. Compression of deep convolutional neural networks for fast and low power mobile applications. In *ICLR*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada. Association for Computational Linguistics.

Yann LeCun, John S Denker, and Sara A Solla. 1990. Optimal brain damage. In *Advances in neural information processing systems*, pages 598–605.

Darryl Lin, Sachin Talathi, and Sreekanth Annapureddy. 2016. Fixed point quantization of deep convolutional networks. In *International Conference on Machine Learning*, pages 2849–2858.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Ekaterina Lobacheva, Nadezhda Chirkova, and Dmitry Vetrov. 2017. Bayesian sparsification of recurrent neural networks. *arXiv preprint arXiv:1708.00077*.

Roland Memisevic. 2011. Learning to relate images: Mapping units, complex cells and simultaneous eigenspaces. *arXiv preprint arXiv:1110.0107*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. *proceedings of ACL-08: HLT*, pages 236–244.

Jeff Mitchell and Mirella Lapata. 2009. Language models based on semantic composition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 430–439. Association for Computational Linguistics.

Alexander Novikov, Dmitrii Podoprikhin, Anton Osokin, and Dmitry P Vetrov. 2015. Tensorizing neural networks. In *Advances in neural information processing systems*, pages 442–450.

Steven Pinker. 2003. *The language instinct: How the mind creates language*. Penguin UK.

Tara N Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, and Bhuvana Ramabhadran. 2013. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6655–6659. IEEE.

Raphael Shu and Hideki Nakayama. 2017. Compressing word embeddings via deep compositional code learning. *arXiv preprint arXiv:1711.01068*.

Wei Wen, Yuxiong He, Samyam Rajbhandari, Wenhan Wang, Fang Liu, Bin Hu, Yiran Chen, and Hai Li. 2017. Learning intrinsic sparse structures within long short-term memory. *CoRR*, abs/1709.05027.

Yuhui Xu, Yongzhuang Wang, Aojun Zhou, Weiyao Lin, and Hongkai Xiong. 2018. Deep neural network compression with single and multiple level quantization. *CoRR*.

Xiyu Yu, Tongliang Liu, Xinchao Wang, and Dacheng Tao. 2017. On compressing deep models by low rank and sparse decomposition. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 67–76.

# It's All in the Name: Mitigating Gender Bias with Name-Based Counterfactual Data Substitution

**Rowan Hall Maudslay**[1] **Hila Gonen**[2] **Ryan Cotterell**[1] **Simone Teufel**[1]
[1] Department of Computer Science and Technology, University of Cambridge
[2] Department of Computer Science, Bar-Ilan University
{rh635,rdc42,sht25}@cam.ac.uk hilagnn@gmail.com

## Abstract

This paper treats gender bias latent in word embeddings. Previous mitigation attempts rely on the operationalisation of gender bias as a projection over a linear subspace. An alternative approach is *Counterfactual Data Augmentation* (CDA), in which a corpus is duplicated and augmented to remove bias, e.g. by swapping all inherently-gendered words in the copy. We perform an empirical comparison of these approaches on the English Gigaword and Wikipedia, and find that whilst both successfully reduce direct bias and perform well in tasks which quantify embedding quality, CDA variants outperform projection-based methods at the task of drawing non-biased gender analogies by an average of 19% across both corpora. We propose two improvements to CDA: *Counterfactual Data Substitution* (CDS), a variant of CDA in which potentially biased text is randomly substituted to avoid duplication, and the Names Intervention, a novel name-pairing technique that vastly increases the number of words being treated. CDA/S with the Names Intervention is the only approach which is able to mitigate indirect gender bias: following debiasing, previously biased words are significantly less clustered according to gender (cluster purity is reduced by 49%), thus improving on the state-of-the-art for bias mitigation.

## 1 Introduction

*Gender bias* describes an inherent prejudice against a gender, captured both by individuals and larger social systems. Word embeddings, a popular machine-learnt semantic space, have been shown to retain gender bias present in corpora used to train them (Caliskan et al., 2017). This results in gender-stereotypical vector analogies à la Mikolov et al. (2013), such as *man*:*computer programmer* :: *woman*:*homemaker* (Bolukbasi et al., 2016), and such bias has been shown to materialise in a variety of downstream tasks, e.g. coreference resolution (Rudinger et al., 2018; Zhao et al., 2018).

By operationalising gender bias in word embeddings as a linear subspace, Bolukbasi et al. (2016) are able to debias with simple techniques from linear algebra. Their method successfully mitigates **direct bias**: *man* is no longer more similar to *computer programmer* in vector space than *woman*. However, the structure of gender bias in vector space remains largely intact, and the new vectors still evince **indirect bias**: associations which result from gender bias between not explicitly gendered words, for example a possible association between *football* and *business* resulting from their mutual association with explicitly masculine words (Gonen and Goldberg, 2019). In this paper we continue the work of Gonen and Goldberg, and show that another paradigm for gender bias mitigation proposed by Lu et al. (2018), Counterfactual Data Augmentation (CDA), is also unable to mitigate indirect bias. We also show, using a new test we describe (non-biased gender analogies), that WED might be removing too much gender information, casting further doubt on its operationalisation of gender bias as a linear subspace.

To improve CDA we make two proposals. The first, Counterfactual Data Substitution (CDS), is designed to avoid text duplication in favour of substitution. The second, the Names Intervention, is a method which can be applied to either CDA or CDS, and treats bias inherent in first names. It does so using a novel name pairing strategy that accounts for both name frequency and gender-specificity. Using our improvements, the clusters of the most biased words exhibit a reduction of cluster purity by an average of 49% across both corpora following treatment, thereby offering a partial solution to the problem of indirect bias as formalised by Gonen and Goldberg (2019). Additionally, although one could expect that the debiased embeddings



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5267–5275, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

might suffer performance losses in computational linguistic tasks, our embeddings remain useful for at least two such tasks, word similarity and sentiment classification (Le and Mikolov, 2014).

## 2 Related Work

The measurement and mitigation of gender bias relies on the chosen operationalisation of gender bias. As a direct consequence, how researchers choose to operationalise bias determines both the techniques at one's disposal to mitigate the bias, as well as the yardstick by which success is determined.

### 2.1 Word Embedding Debiasing

One popular method for the mitigation of gender bias, introduced by Bolukbasi et al. (2016), measures the genderedness of words by the extent to which they point in a **gender direction**. Suppose we embed our words into $\mathbb{R}^d$. The fundamental assumption is that there exists a linear subspace $B \subset \mathbb{R}^d$ that contains (most of) the gender bias in the space of word embeddings. (Note that $B$ is a direction when it is a single vector.) We term this assumption the **gender subspace hypothesis**. Thus, by basic linear algebra, we may decompose any word vector $\mathbf{v} \in \mathbb{R}^d$ as the sum of the projections onto the bias subspace and its complement: $\mathbf{v} = \mathbf{v}_B + \mathbf{v}_{\perp B}$. The (implicit) operationalisation of gender bias under this hypothesis is, then, the magnitiude of the bias vector $||\mathbf{v}_B||_2$.

To capture $B$, Bolukbasi et al. (2016) first construct two sets, $\mathcal{D}_{male}$ and $\mathcal{D}_{female}$ containing the male- and female-oriented pairs, using a set of gender-definitional pairs, e.g., *man–woman* and *husband–wife*. They then define $\mathcal{D} = \mathcal{D}_{male} \cup \mathcal{D}_{female}$ as the union of the two sets. They compute the empirical covariance matrix

$$C = \sum_{w \in \mathcal{D}} (\vec{w} - \mu)(\vec{w} - \mu)^\top \quad (1)$$

where $\mu$ is the mean embeddings of the words in $\mathcal{D}$, then $B$ is taken to be the $k$ eigenvectors of $C$ associated with the largest eigenvalues. Bolukbasi et al. set $k = 1$, and thus define a gender direction.

Using this operalisation of gender bias, Bolukbasi et al. go on to provide a linear-algebraic method (Word Embedding Debiasing, WED, originally "hard debiasing") to remove gender bias in two phases: first, for non-gendered words, the gender direction is removed ("neutralised"). Second, pairs of gendered words such as *mother* and *father* are made equidistant to all non-gendered words ("equalised"). Crucially, under the gender subspace hypothesis, it is only necessary to identify the subspace $B$ as it is possible to perfectly remove the bias under this operationalisation using tools from numerical linear algebra.

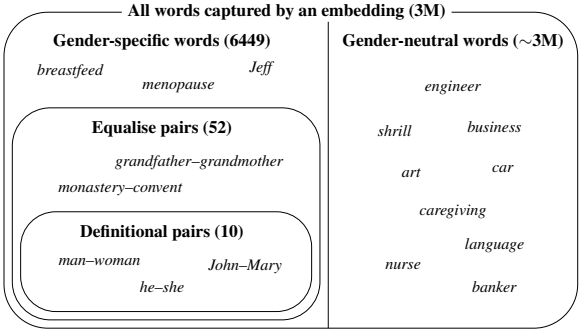


Figure 1: Word sets used by WED with examples

The method uses three sets of words or word pairs: 10 definitional pairs (used to define the gender direction), 218 gender-specific seed words (expanded to a larger set using a linear classifier, the compliment of which is neutralised in the first step), and 52 equalise pairs (equalised in the second step). The relationships among these sets are illustrated in Figure 1; for instance, gender-neutral words are defined as all words in an embedding that are not gender-specific.

Bolukbasi et al. find that this method results in a 68% reduction of stereotypical analogies as identified by human judges. However, bias is removed only insofar as the operationalisation allows. In a comprehensive analysis, Gonen and Goldberg (2019) show that the original structure of bias in the WED embedding space remains intact.

### 2.2 Counterfactual Data Augmentation

As an alternative to WED, Lu et al. (2018) propose Counterfactual Data Augmentation (CDA), in which a text transformation designed to invert bias is performed on a text corpus, the result of which is then appended to the original, to form a new bias-mitigated corpus used for training embeddings. Several interventions are proposed: in the simplest, occurrences of words in 124 gendered word pairs are swapped. For example, 'the woman cleaned the kitchen' would (counterfactually) become 'the man cleaned the kitchen' as *man–woman* is on the list. Both versions would then together be used in embedding training, in effect neutralising the *man–woman* bias.

The *grammar intervention*, Lu et al.'s improved intervention, uses coreference information to veto swapping gender words when they corefer to a proper noun.[1] This avoids *Elizabeth ... she ... queen* being changed to, for instance, *Elizabeth ... he ... king*. It also uses POS information to avoid ungrammaticality related to the ambiguity of *her* between personal pronoun and possessive determiner. In the context, 'her teacher was proud of her', this results in the correct sentence '*his* teacher was proud of *him*'.

## 3 Improvements to CDA

We prefer the philosophy of CDA over WED as it makes fewer assumptions about the operationalisation of the bias it is meant to mitigate.

### 3.1 Counterfactual Data Substitution

The duplication of text which lies at the heart of CDA will produce debiased corpora with peculiar statistical properties unlike those of naturally occurring text. Almost all observed word frequencies will be even, with a notable jump from 2 directly to 0, and a type–token ratio far lower than predicted by Heaps' Law for text of this length. The precise effect this will have on the resulting embedding space is hard to predict, but we assume that it is preferable not to violate the fundamental assumptions of the algorithms used to create embeddings.

As such, we propose to apply substitutions probabilistically (with 0.5 probability), which results in a non-duplicated counterfactual training corpus, a method we call **Counterfactual Data Substitution (CDS)**. Substitutions are performed on a per-document basis in order to maintain grammaticality and discourse coherence. This simple change should have advantages in terms of naturalness of text and processing efficiency, as well as theoretical foundation.

### 3.2 The Names Intervention

Our main technical contribution in this paper is to provide a method for better counterfactual augmentation, which is based on bipartite-graph matching of names. Instead of Lu et. al's (2018) solution of not treating words which corefer to proper nouns in order to maintain grammaticality, we propose an explicit treatment of first names. This is because we note that as a result of not swapping the gender of words which corefer with proper nouns, CDA could in fact reinforce certain biases instead of mitigate them. Consider the sentence 'Tom ... He is a successful and powerful executive.' Since *he* and *Tom* corefer, the counterfactual corpus copy will not replace *he* with *she* in this instance, and as the method involves a duplication of text, this would result in a stronger, not weaker, association between *he* and gender-stereotypic concepts present like *executive*. Even under CDS, this would still mean that biased associations are left untreated (albeit at least not reinforced). Treating names should in contrast effect a real neutralisation of bias, with the added bonus that grammaticality is maintained without the need for coreference resolution.

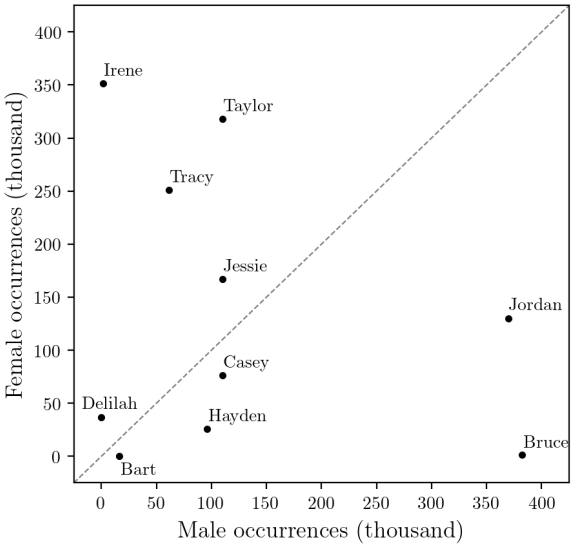


Figure 2: Frequency and gender-specificity of names in the SSA dataset

The United States Social Security Administration (SSA) dataset contains a list of all first names from Social Security card applications for births in the United States after 1879, along with their gender.[2] Figure 2 plots a few example names according to their male and female occurrences, and shows that names have varying degrees of gender-specificity.[3]

We fixedly associate pairs of names for swapping, thus expanding Lu et al.'s short list of gender pairs vastly. Clearly both name frequency and the degree of gender-specificity are relevant to this bipartite matching. If only frequency were considered, a more gender-neutral name (e.g. *Taylor*)

[1] We interpret Lu et al.'s (2018) phrase "cluster" to mean "coreference chain".

[2] https://www.ssa.gov/oact/babynames/background.html

[3] The dotted line represents gender-neutrality, and more frequent names are located further from the origin.

could be paired with a very gender-specific name (e.g. *John*), which would negate the gender intervention in many cases (namely whenever a male occurrence of *Taylor* is transformed into *John*, which would also result in incorrect pronouns, if present). If, on the other hand, only the degree of gender-specificity were considered, we would see frequent names (like *James*) being paired with far less frequent names (like *Sybil*), which would distort the overall frequency distribution of names. This might also result in the retention of a gender signal: for instance, swapping a highly frequent male name with a rare female name might simply make the rare female name behave as a new link between masculine contexts (instead of the original male name), as it rarely appears in female contexts.

Figure 3 shows a plot of various names' number of primary gender[4] occurances against their secondary gender occurrences, with red dots for primary-male and blue crosses for primary-female names.[5] The problem of finding name-pairs thus decomposes into a Euclidean-distance bipartite matching problem, which can be solved using the Hungarian method (Kuhn, 1955). We compute pairs for the most frequent 2500 names of each gender in the SSA dataset. There is also the problem that many names are also common nouns (e.g. *Amber*, *Rose*, or *Mark*), which we solve using Named Entity Recognition.

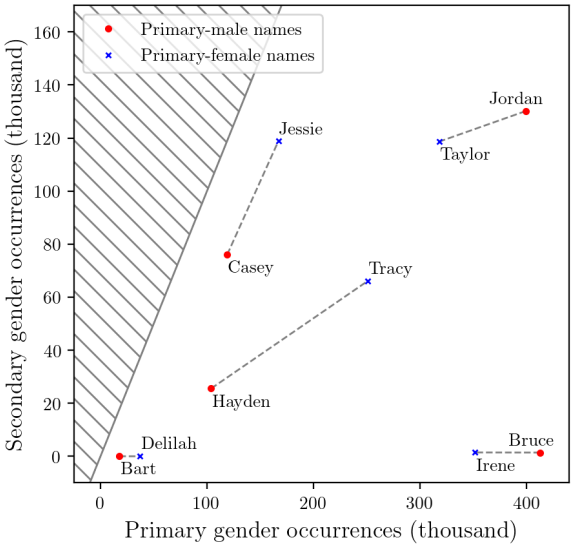


Figure 3: Bipartite matching of names by frequency and gender-specificity

## 4 Experimental Setup

We compare eight variations of the mitigation methods. **CDA** is our reimplementation of Lu et al.'s (2018) naïve intervention, **gCDA** uses their grammar intervention, and **nCDA** uses our new Names Intervention. **gCDS** and **nCDS** are variants of the grammar and Names Intervention using CDS. **WED40** is our reimplementation of Bolukbasi et al.'s (2016) method, which (like the original) uses a single component to define the gender subspace, accounting for $> 40\%$ of variance. As this is much lower than in the original paper (where it was 60%, reproduced in Figure 4), we define a second space, **WED70**, which uses a 2D subspace accounting for $> 70\%$ of variance. To test whether WED profits from additional names, we use the 5000 paired names in the names gazetteer as additional equalise pairs (**nWED70**).[6] As control, we also evaluate the unmitigated space (**none**).

We perform an empirical comparison of these bias mitigation techniques on two corpora, the Annotated English Gigaword (Napoles et al., 2012) and Wikipedia. Wikipedia is of particular interest, since though its Neutral Point of View (NPOV) policy[7] predicates that all content should be presented without bias, women are nonetheless less likely to be deemed "notable" than men of equal stature (Reagle and Rhue, 2011), and there are differences in the choice of language used to describe them (Bamman and Smith, 2014; Graells-Garrido et al., 2015). We use the annotation native to the Annotated English Gigaword, and process Wikipedia with CoreNLP (statistical coreference; bidirectional tagger). Embeddings are created using Word2Vec[8]. We use the original complex lexical input (gender-word pairs and the like) for each algorithm as we assume that this benefits each algorithm most. Expanding the set of gender-specific words for WED (following Bolukbasi et al., using a linear classifier) on Gigaword resulted in 2141 such words, 7146 for Wikipedia.[9]

[4]Defined as its most frequently occurring gender.

[5]The hatched area demarcates an area of the graph where no names can exist: if any name did then its primary and secondary gender would be reversed and it would belong to the alternate set.

[6]We use the 70% variant as preliminary experimentation showed that it was superior to WED40.

[7]https://en.wikipedia.org/wiki/Wikipedia:Neutral_point_of_view

[8]A CBOW model was trained over five epochs to produce 300 dimensional embeddings. Words were lowercased, punctuation other than underscores and hyphens removed, and tokens with fewer than ten occurrences were discarded.

[9]We modify or remove some phrases from the training data not included in the vocabulary of our embeddings.

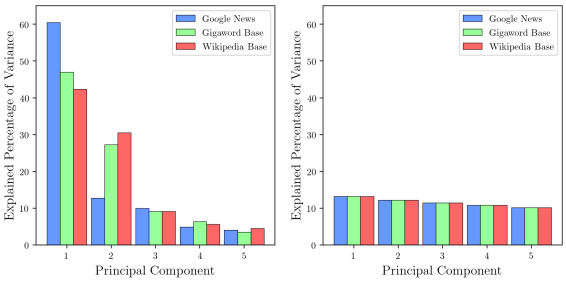


Figure 4: Variance explained by the top Principal Components of the definitional word pairs (left) and random unit vectors (right)

In our experiments, we test the degree to which the spaces are successful at mitigating direct and indirect bias, as well as the degree to which they can still be used in two NLP tasks standardly performed with embeddings, word similarity and sentiment classification. We also introduce one further, novel task, which is designed to quantify how well the embedding spaces capture an understanding of gender using non-biased analogies. Our evaluation matrix and methodology is expanded below.

**Direct bias** Caliskan et al. (2017) introduce the Word Embedding Association Test (WEAT), which provides results analogous to earlier psychological work by Greenwald et al. (1998) by measuring the difference in relative similarity between two sets of target words $X$ and $Y$ and two sets of attribute words $A$ and $B$. We compute Cohen's $d$ (a measure of the difference in relative similarity of the word sets within each embedding; higher is more biased), and a one-sided $p$-value which indicates whether the bias detected by WEAT within each embedding is significant (the best outcome being that no such bias is detectable). We do this for three tests proposed by Nosek et al. (2002) which measure the strength of various gender stereotypes: art–maths, arts–sciences, and careers–family.[10]

**Indirect bias** To demonstrate indirect gender bias we adapt a pair of methods proposed by Gonen and Goldberg (2019). First, we test whether the most-biased words prior to bias mitigation remain clustered following bias mitigation. To do this, we define a new subspace, $\vec{b}_{\text{test}}$, using the 23 word pairs used in the Google Analogy family test subset (Mikolov et al., 2013) following Bolukbasi et al.'s (2016) method, and determine the 1000 most biased words in each corpus (the 500 words most similar to $\vec{b}_{\text{test}}$ and $-\vec{b}_{\text{test}}$) in the unmitigated embedding. For each debiased embedding we then project these words into 2D space with tSNE (van der Maaten and Hinton, 2008), compute clusters with $k$-means, and calculate the clusters' V-measure (Rosenberg and Hirschberg, 2007). Low values of cluster purity indicate that biased words are less clustered following bias mitigation.

Second, we test whether a classifier can be trained to reclassify the gender of debiased words. If it succeeds, this would indicate that bias-information still remains in the embedding. We trained an RBF-kernel SVM classifier on a random sample of 1000 out of the 5000 most biased words from each corpus using $\vec{b}_{\text{test}}$ (500 from each gender), then report the classifier's accuracy when reclassifying the remaining 4000 words.

**Word similarity** The quality of a space is traditionally measured by how well it replicates human judgements of word similarity. The SimLex-999 dataset (Hill et al., 2015) provides a ground-truth measure of similarity produced by 500 native English speakers.[11] Similarity scores in an embedding are computed as the cosine angle between word-vector pairs, and Spearman correlation between embedding and human judgements are reported. We measure correlative significance at $\alpha = 0.01$.

**Sentiment classification** Following Le and Mikolov (2014), we use a standard sentiment classification task to quantify the downstream performance of the embedding spaces when they are used as a pretrained word embedding input (Lau and Baldwin, 2016) to Doc2Vec on the Stanford Large Movie Review dataset. The classification is performed by an SVM classifier using the document embeddings as features, trained on 40,000 labelled reviews and tested on the remaining 10,000 documents, reported as error percentage.

**Non-biased gender analogies** When proposing WED, Bolukbasi et al. (2016) use human raters to class gender-analogies as either *biased* (*woman*:*housewife* :: *man*:*shopkeeper*) or *appropriate* (*woman*:*grandmother* :: *man*::*grandfather*), and postulate that whilst biased analogies are undesirable, appropriate ones should remain. Our new analogy test uses the 506 analogies in the *fam-*

[10] In the careers–family test the gender dimension is expressed by female and male first names, unlike in the other sets, where pronouns and typical gendered words are used.

[11] It explicitly quantifies similarity rather than association or relatedness; pairs of entities like *coffee* and *cup* have a low rating.

| Method | Art–Maths $d$ | Art–Maths $p$ | Arts–Sciences $d$ | Arts–Sciences $p$ | Career–Family $d$ | Career–Family $p$ |
|---|---|---|---|---|---|---|
| **Gigaword** | | | | | | |
| none | 1.32 | $< 10^{-2}$ | 1.50 | $< 10^{-3}$ | 1.74 | $< 10^{-4}$ |
| CDA | 0.67 | .10 | 1.05 | .02 | 1.79 | $< 10^{-4}$ |
| gCDA | 1.16 | .01 | 1.46 | $< 10^{-2}$ | 1.77 | $< 10^{-4}$ |
| nCDA | −0.49 | .83 | 0.34 | .27 | 1.45 | $< 10^{-3}$ |
| gCDS | 0.96 | .03 | 1.31 | $< 10^{-2}$ | 1.78 | $< 10^{-4}$ |
| nCDS | −0.19 | .63 | 0.48 | .19 | 1.45 | $< 10^{-3}$ |
| WED40 | −0.73 | .92 | 0.31 | .28 | 1.24 | $< 10^{-2}$ |
| WED70 | −0.73 | .92 | 0.30 | .29 | 1.15 | $< 10^{-2}$ |
| nWED70 | 0.30 | .47 | 0.54 | .19 | 0.59 | .15 |
| **Wikipedia** | | | | | | |
| none | 1.64 | $< 10^{-3}$ | 1.51 | $< 10^{-3}$ | 1.88 | $< 10^{-4}$ |
| CDA | 1.58 | $< 10^{-3}$ | 1.66 | $< 10^{-4}$ | 1.87 | $< 10^{-4}$ |
| gCDA | 1.52 | $< 10^{-3}$ | 1.57 | $< 10^{-3}$ | 1.84 | $< 10^{-4}$ |
| nCDA | 1.06 | .02 | 1.54 | $< 10^{-4}$ | 1.65 | $< 10^{-4}$ |
| gCDS | 1.45 | $< 10^{-3}$ | 1.53 | $< 10^{-3}$ | 1.87 | $< 10^{-4}$ |
| nCDS | 1.05 | .02 | 1.37 | $< 10^{-3}$ | 1.65 | $< 10^{-4}$ |
| WED40 | 1.28 | $< 10^{-2}$ | 1.36 | $< 10^{-2}$ | 1.81 | $< 10^{-4}$ |
| WED70 | 1.05 | .02 | 1.24 | $< 10^{-2}$ | 1.67 | $< 10^{-3}$ |
| nWED70 | −0.46 | .52 | −0.42 | .51 | 0.85 | .05 |
| *Nosek et al.* | 0.82 | $< 10^{-2}$ | 1.47 | $< 10^{-24}$ | 0.72 | $< 10^{-2}$ |

Table 1: Direct bias results

*ily analogy* subset of the Google Analogy Test set (Mikolov et al., 2013) to define many such appropriate analogies that should hold even in a debiased environment, such as *boy*:*girl* :: *nephew*:*niece*.[12] We use a proportional pair-based analogy test, which measures each embedding's performance when drawing a fourth word to complete each analogy, and report error percentage.

## 5 Results

**Direct bias** Table 1 presents the $d$ scores and WEAT one-tailed $p$-values, which indicate whether the difference in samples means between targets $X$ and $Y$ and attributes $A$ and $B$ is significant. We also compute a two-tailed $p$-value to determine whether the difference between the various sets is significant.[13]

On Wikipedia, nWED70 outperforms every other method ($p < 0.01$), and even at $\alpha = 0.1$ bias was undetectable. In all CDA/S variants, the Names Intervention performs significantly better than other intervention strategies (average $d$ for nCDS across all tests 0.95 vs. 1.39 for the best non-names CDA/S variants). Excluding the Wikipedia careers–family test (in which the CDA and CDS

[12]The entire Google Analogy Test set contains 19,544 analogies, which are usually reported as a single result or as a pair of semantic and syntactic results.

[13]Throughout this paper, we test significance in the differences between the embeddings with a two-tailed Monte Carlo permutation test at significance interval $\alpha = 0.01$ with $r = 10,000$ permutations.

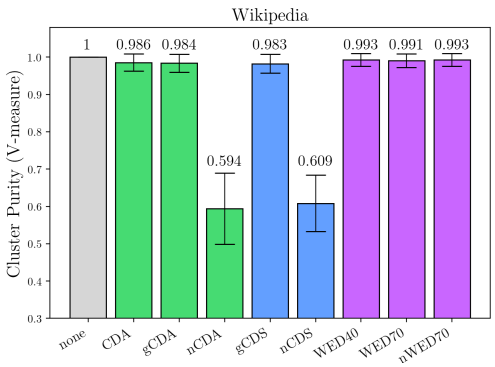


Figure 5: Most biased cluster purity results

variants are indistinguishable at $\alpha = 0.01$), the CDS variants are numerically better than their CDA counterparts in 80% of the test cases, although many of these differences are not significant.

Generally, we notice a trend of WED reducing direct gender bias slightly better than CDA/S. Impressively, WED even successfully reduces bias in the careers–family test, where gender information is captured by names, which were not in WED's gender-equalise word-pair list for treatment.

**Indirect bias** Figure 5 shows the V-measures of the clusters of the most biased words in Wikipedia for each embedding. Gigaword patterns similarly (see appendix). Figure 6 shows example tSNE projections for the Gigaword embeddings ("V" refers to their V-measures; these examples were chosen as they represent the best results achieved by Bolukbasi et al.'s (2016) method, Lu et al.'s (2018) method, and our new names variant). On both corpora, the new nCDA and nCDS techniques have significantly lower purity of biased-word cluster than all other evaluated mitigation techniques (0.420 for nCDS on Gigaword, which corresponds to a reduction of purity by 58% compared to the unmitigated embedding, and 0.609 (39%) on Wikipedia). nWED70's V-Measure is significantly higher than either of the other Names variants (reduction of 11% on Gigaword, only 1% on Wikipedia), suggesting that the success of nCDS and nCDA is not merely due to their larger list of gender-words.

Figure 7 shows the results of the second test of indirect bias, and reports the accuracy of a classifier trained to reclassify previously gender biased words on the Wikipedia embeddings (Gigaword patterns similarly).[14] These results reinforce the finding of the clustering experiment: once again,

[14]The 95% confidence interval is calculated by a Wilson score interval, i.e., assuming a normal distribution.

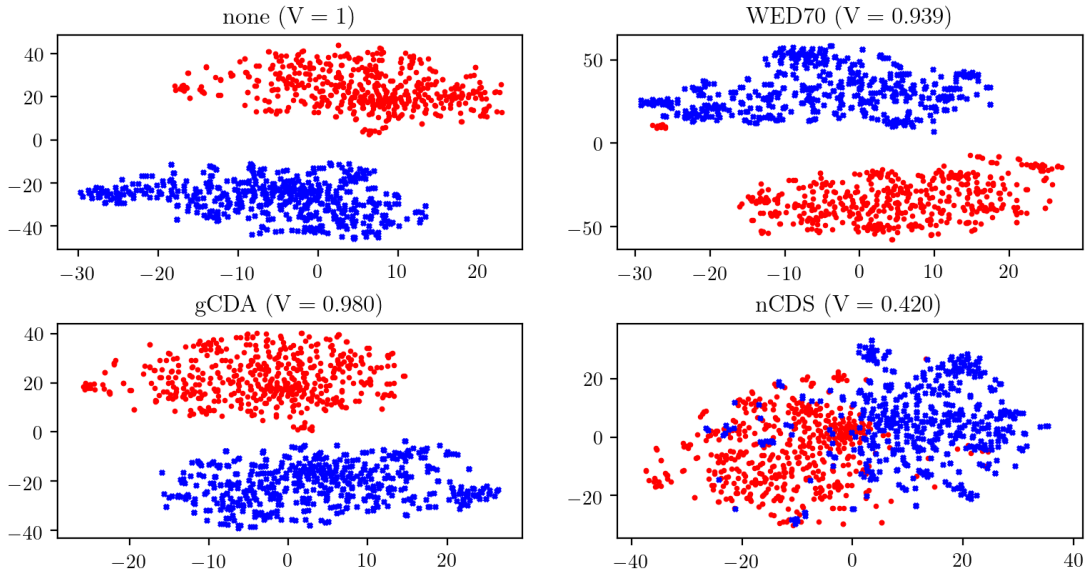


Figure 6: Clustering of biased words (Gigaword)

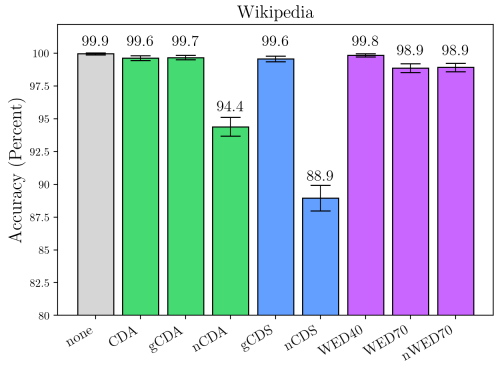


Figure 7: Reclassification of most biased words results

| Method | $r_s$ | |
|---|---|---|
| | **Gigaword** | **Wikipedia** |
| none | 0.385 | 0.368 |
| CDA | 0.381 | 0.363 |
| gCDA | 0.381 | 0.363 |
| nCDA | 0.380 | 0.365 |
| gCDS | 0.382 | 0.366 |
| nCDS | 0.380 | 0.362 |
| WED40 | 0.386 | 0.371 |
| WED70 | 0.395 | 0.375 |
| nWED70 | 0.384 | 0.367 |

Table 2: Word similarity Results

nCDS outperforms all other methods significantly on both corpora ($p < 0.01$), although it should be noted that the successful reclassification rate remains relatively high (e.g. 88.9% on Wikipedia).

We note that nullifying indirect bias associations entirely is not necessarily the goal of debiasing, since some of these may result from causal links in the domain. For example, whilst associations between *man* and *engineer* and between *man* and *car* are each stereotypic (and thus could be considered examples of direct bias), an association between *engineer* and *car* might well have little to do with gender bias, and so should not be mitigated.

**Word similarity** Table 2 reports the SimLex-999 Spearman rank-order correlation coefficients $r_s$ (all are significant, $p < 0.01$). Surprisingly, the WED40 and 70 methods outperform the unmitigated embedding, although the difference in result is small (0.386 and 0.395 vs. 0.385 on Gigaword, 0.371 and 0.367 vs. 0.368 on Wikipedia). nWED70, on the other hand, performs worse than the unmitigated embedding (0.384 vs. 0.385 on Gigaword, 0.367 vs. 0.368 on Wikipedia). CDA and CDS methods do not match the quality of the unmitigated space, but once again the difference is small. It should be noted that since SimLex-999 was produced by human raters, it will reflect the human biases these methods were designed to remove, so worse performance might result from successful bias mitigation.

**Sentiment classification** Figure 8 shows the sentiment classification error rates for Wikipedia (Gigaword patterns similarly). Results are somewhat inconclusive. While WED70 significantly improves the performance of the sentiment classifier from the unmitigated embedding on both corpora ($p < 0.05$), the improvement is small (never more than 1.1%). On both corpora, nothing outperforms WED70 or the Names Intervention variants.

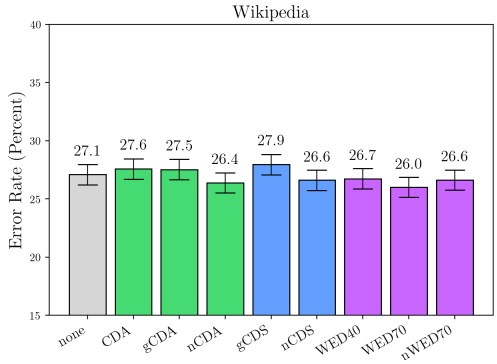


Figure 8: Sentiment classification results

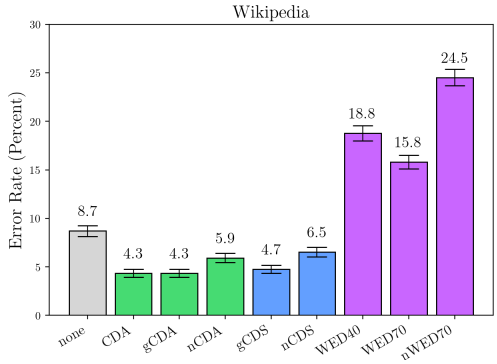


Figure 9: Non-biased gender analogy results

**Non-biased gender analogies** Figure 9 shows the error rates for non-biased gender analogies for Wikipedia. CDA and CDS are numerically better than the unmitigated embeddings (an effect which is always significant on Gigaword, shown in the appendices, but sometimes insignificant on Wikipedia). The WED variants, on the other hand, perform significantly worse than the unmitigated sets on both corpora (27.1 vs. 9.3% for the best WED variant on Gigaword; 18.8 vs. 8.7% on Wikipedia). WED thus seems to remove too much gender information, whilst CDA and CDS create an improved space, perhaps because they reduce the effect of stereotypical associations which were previously used incorrectly when drawing analogies.

## 6 Conclusion

We have replicated two state-of-the-art bias mitigation techniques, WED and CDA, on two large corpora, Wikipedia and the English Gigaword. In our empirical comparison, we found that although both methods mitigate direct gender bias and maintain the interpretability of the space, WED failed to maintain a robust representation of gender (the best variants had an error rate of 23% average when drawing non-biased analogies, suggesting that too much gender information was removed). A new variant of CDA we propose (the Names Intervention) is the only to successfully mitigate indirect gender bias: following its application, previously biased words are significantly less clustered according to gender, with an average of 49% reduction in cluster purity when clustering the most biased words. We also proposed Counterfactual Data Substitution, which generally performed better than the CDA equivalents, was notably quicker to compute (as Word2Vec is linear in corpus size), and in theory allows for multiple intervention layers without a corpus becoming exponentially large.

A fundamental limitation of all the methods compared is their reliance on predefined lists of gender words, in particular of pairs. Lu et al.'s pairs of *manager*::*manageress* and *murderer*::*murderess* may be counterproductive, as their augmentation method perpetuates a male reading of *manager*, which has become gender-neutral over time. Other issues arise from differences in spelling (e.g. *mum* vs. *mom*) and morphology (e.g. *his* vs. *her* and *hers*). Biologically-rooted terms like *breastfeed* or *uterus* do not lend themselves to pairing either. The strict use of pairings also imposes a gender binary, and as a result non-binary identities are all but ignored in the bias mitigation literature.

Future work could extend the Names Intervention to names from other languages beyond the US-based gazetteer used here. Our method only allows for there to be an equal number of male and female names, but if this were not the case one ought to explore the possibility of a many-to-one mapping, or perhaps a probablistic approach (though difficulties would be encountered sampling simultaneously from two distributions, frequency and gender-specificity). A mapping between nicknames (not covered by administrative sources) and formal names could be learned from a corpus for even wider coverage, possibly via the intermediary of coreference chains. Finally, given that names have been used in psychological literature as a proxy for race (e.g. Greenwald et al.), the Names Intervention could also be used to mitigate racial biases (something which, to the authors' best knowledge, has never been attempted), but finding pairings could prove problematic. It is important that other work looks into operationalising bias beyond the subspace definition proposed by Bolukbasi et al. (2016), as it is becoming increasingly evident that gender bias is not linear in embedding space.

## References


David Bamman and Noah A. Smith. 2014. Unsupervised discovery of biographical structure from text. *Transactions of the Association for Computational Linguistics*, 2:363–376.

Tolga Bolukbasi, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and Adam Tauman Kalai. 2016. Man is to computer programmer as woman is to homemaker? Debiasing word embeddings. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4349–4357.

Aylin Caliskan, Joanna J. Bryson, and Arvind Narayanan. 2017. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186.

Hila Gonen and Yoav Goldberg. 2019. Lipstick on a pig: Debiasing methods cover up systematic gender biases in word embeddings but do not remove them. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, Minneapolis, Minnesota. Association for Computational Linguistics.

Eduardo Graells-Garrido, Mounia Lalmas, and Filippo Menczer. 2015. First women, second sex: Gender bias in Wikipedia. In *Proceedings of the 26th ACM Conference on Hypertext & Social Media*, HT '15, pages 165–174, New York, NY, USA. ACM.

Anthony G. Greenwald, Debbie E. McGhee, and Jordan L. K. Schwartz. 1998. Measuring individual differences in implicit cognition: The Implicit Association Test. *Journal of Personality and Social Psychology*, 74(6):1464–1480.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. SimLex-999: Evaluating semantic models with (genuine) similarity estimation. *American Journal of Computational Linguistics*, 41(4):665–695.

Harold W Kuhn. 1955. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97.

Jey Han Lau and Timothy Baldwin. 2016. An empirical evaluation of doc2vec with practical insights into document embedding generation. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 78–86, Berlin, Germany. Association for Computational Linguistics.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1188–1196, Bejing, China. PMLR.

Kaiji Lu, Piotr Mardziel, Fangjing Wu, Preetam Amancharla, and Anupam Datta. 2018. Gender bias in neural natural language processing. *CoRR*, abs/1807.11714.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated gigaword. In *AKBC-WEKEX Workshop at NAACL 2012*.

Brian A. Nosek, Mahzarin R. Banaji, and Anthony G Greenwald. 2002. Harvesting implicit group attitudes and beliefs from a demonstration web site. *Group Dynamics: Theory, Research, and Practice*, 6 1:101–115.

Joseph Reagle and Lauren Rhue. 2011. Gender bias in Wikipedia and Britannica. *International Journal of Communication*, 5(0).

Andrew Rosenberg and Julia Hirschberg. 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 410–420, Prague, Czech Republic. Association for Computational Linguistics.

Rachel Rudinger, Jason Naradowsky, Brian Leonard, and Benjamin Van Durme. 2018. Gender bias in coreference resolution. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 8–14, New Orleans, Louisiana. Association for Computational Linguistics.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2018. Gender bias in coreference resolution: Evaluation and debiasing methods. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 15–20, New Orleans, Louisiana. Association for Computational Linguistics.

# Examining Gender Bias in Languages with Grammatical Gender

**Pei Zhou**[1,2], **Weijia Shi**[1], **Jieyu Zhao**[1], **Kuan-Hao Huang**[1],
**Muhao Chen**[1,3], **Ryan Cotterell**[4], **Kai-Wei Chang**[1]
[1]Department of Computer Science, University of California Los Angeles
[2]Department of Computer Science, University of Southern California
[3]Department of Computer and Information Science, University of Pennsylvania
[4]Department of Computer Science, Johns Hopkins University
peiz@usc.edu; {swj0419,jyzhao, khhuang, kwchang}@cs.ucla.edu;
muhao@seas.upenn.edu; ryan.cotterell@jhu.edu

## Abstract

Recent studies have shown that word embeddings exhibit gender bias inherited from the training corpora. However, most studies to date have focused on quantifying and mitigating such bias only in English. These analyses cannot be directly extended to languages that exhibit morphological agreement on gender, such as Spanish and French. In this paper, we propose new metrics for evaluating gender bias in word embeddings of these languages and further demonstrate evidence of gender bias in bilingual embeddings which align these languages with English. Finally, we extend an existing approach to mitigate gender bias in word embedding under both monolingual and bilingual settings. Experiments on modified Word Embedding Association Test, word similarity, word translation, and word pair translation tasks show that the proposed approaches effectively reduce the gender bias while preserving the utility of the embeddings.

## 1 Introduction

Word embeddings are widely used in modern natural language processing tools. By virtue of their being trained on large, human-written corpora, recent studies have shown that word embeddings, in addition to capturing a word's semantics, also encode gender bias in society (Bolukbasi et al., 2016; Caliskan et al., 2017). As a result of this bias, the embeddings may cause undesired consequences in the resulting models (Zhao et al., 2018a; Font and Costa-jussà, 2019). Therefore, extensive effort has been put toward analyzing and mitigating gender bias in word embeddings (Bolukbasi et al., 2016; Zhao et al., 2018b; Dev and Phillips, 2019; Ethayarajh et al., 2019).

Existing studies on gender bias almost exclusively focus on English (EN) word embeddings. Unfortunately, the techniques used to mitigate bias in English word embeddings cannot be directly applied to languages with grammatical gender[1], where all nouns are assigned a gender class and the corresponding dependent articles, adjectives, and verbs must agree in gender with the noun (e.g. in Spanish: *la buena enfermera* the good female nurse, *el buen enfermero* the good male nurse) (Corbett, 1991, 2006). This is because most existing approaches define bias in word embeddings based on the projection of a word on a gender direction (e.g. "nurse" in English is biased because its projection on the gender direction inclines towards female). When the grammatical gender exists, such bias definition is problematic as masculine and feminine words naturally carry gender information. For example, "enfermero" (male nurse) and "enfermera" (female nurse) lean toward male and female, respectively. This is due to morphological agreement and should not be considered as a stereotype.

However, gender bias in the embeddings of languages with grammatical gender indeed exists. For example, when we align Spanish (ES) embeddings to English embeddings, the word "abogado" (male lawyer) is closer to "lawyer" than "abogada" (female lawyer). This observation implies a discrepancy in semantics between the masculine and feminine forms of the same occupation in Spanish word embeddings. A similar observation is also found in other languages, such as French (FR).

In this paper, we refer languages that have gender distinctions for all nouns (e.g., Spanish and German) as *gendered languages* and others that do not mark grammatical gender as *languages without grammatical gender* (e.g., English). We use Spanish and French as running examples and pro-

[1]Grammatical gender is a complicated linguistic phenomenon; many languages contain more than two gender classes. In this paper, we focus on languages with masculine and feminine classes. For gender in semantics, we follow the literature and address only binary gender.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5276–5284,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

pose quantitative methods for evaluating bias in word embeddings of gendered languages and bilingual word embeddings that align a gendered language with English. We first define gender bias in the word embeddings of gendered languages by constructing two gender directions: the semantic gender direction and the grammatical gender direction. We then analyze gender bias in bilingual embeddings using a similar approach.

To mitigate gender bias in these embeddings, we propose two approaches. One is shifting words along the semantic gender direction with respect to an anchor point and the other is mitigating English first and then aligning the embedding spaces. Results show that a hybrid of these two approaches effectively mitigate bias in Spanish and French word embeddings as well as ES-EN, FR-EN bilingual word embeddings. We also show through word similarity and word translation experiments that the utility of the original monolingual and bilingual word embeddings is preserved.

Our contributions are summarized in the following. (1) We show that word embeddings of gendered languages such as Spanish and French contain gender bias and bilingual word embeddings aligning these languages to English also inherit the bias. (2) Based on the observation, we propose new definitions of gender bias by constructing two gender directions. (3) We propose methods to reduce gender bias for both monolingual and bilingual embeddings and show that they effectively mitigate bias while preserving the original utility of embeddings. Source code and data are available at https://github.com/shaoxia57/Bias_in_Gendered_Languages.

## 2 Gender Bias Analysis

This section discusses how to analyze bias in embeddings of gendered languages and bilingual word embeddings.

### 2.1 Bias in Gendered Languages

In gendered languages, all nouns are assigned a gender class. However, inanimate objects (e.g., water and spoon) do not carry the meaning of male or female. To address this issue, we define two gender directions: (1) *semantic gender*, which is defined by a set of gender definition words (e.g., man, male, waitress) and (2) *grammatical gender*, which is defined by a set of masculine and feminine nouns (e.g., water, table, woman). Most analysis of gender bias assumes that the language (e.g,. English) only has the former direction. However, when analyzing languages like Spanish and French, considering the second is necessary. We discuss these two directions in detail as follows.

**Semantic Gender** Following Bolukbasi et al. (2016), we collect a set of gender-definition pairs (e.g., "mujer" (woman) and "hombre" (man) in Spanish). Then, the gender direction is derived by conducting principal component analysis (PCA) (Jolliffe, 2011) over the differences between male- and female-definition word vectors. Similar to the analysis in English, we observe that there is one major principal component carrying the meaning of gender in French and Spanish and we define it as the semantic gender direction $\vec{d}_{PCA}$.

**Grammatical Gender** The number of grammatical gender classes ranges from two to several tens (Corbett, 1991). To simplify the discussion, we focus on noun class systems where two major gender classes are feminine and masculine. However, the proposed approach can be generalized to languages with multiple gender classes (e.g., German). To identify grammatical gender direction, we collect around 3,000 common nouns that are grammatically masculine and 3,000 nouns that are feminine. Since most nouns (e.g., water) do not have a paired word in the other gender class, we do not have pairs of words to represent different grammatical genders. Therefore, instead of applying PCA, we learn the grammatical gender direction $\vec{d}_g$ by Linear Discriminant Analysis (LDA) (Fisher, 1936), which is a standard approach for supervised dimension reduction. The model achieves an average accuracy of 0.92 for predicting the grammatical gender in Spanish and 0.83 in French with 5-fold cross-validation.

Comparing $\vec{d}_{PCA}$ with $\vec{d}_g$, the cosine similarity between them is 0.389, indicating these two directions are overlapped to some extent but not identical. This is reasonable because words such as "mujer (woman)", "doctor (female doctor)" are both semantically and grammatically marked as feminine. To better distinguish between these two directions, we project out the grammatical gender component in the computed gender direction to make the semantic gender direction $\vec{d}_s$ orthogonal to the grammatical gender direction:

$$\vec{d}_s = \vec{d}_{PCA} - \left\langle \vec{d}_{PCA}, \vec{d}_g \right\rangle \vec{d}_g,$$

where $\langle \vec{x}, \vec{y} \rangle$ represents the inner product of two vectors.

**Visualizing and Analyzing Bias in Spanish** We take Spanish as an example and analyze gender bias in Spanish fastText word embeddings (Bojanowski et al., 2017) pre-trained on Spanish Wikipedia. For simplicity, we assume that the embeddings contain all gender forms of the words. To show bias, we randomly select several pairs of gender-definition and occupation words as well as inanimate nouns in Spanish and visualize them on the two gender directions defined above.

Figure 1 shows that the inanimate nouns lie near the origin point on the semantic gender direction, while the masculine and feminine forms of the occupation words are on the opposite sides for both directions. However, while projections of occupation words on the grammatical gender direction are symmetric with respect to the origin point, they are asymmetric when projected on the semantic gender direction. Along the semantic gender direction, female occupation words incline more to the feminine side than the male occupation words to the masculine side. This discrepancy shows that the embeddings carry unequal information for two genders.

**Quantification of Gender Bias** To quantify gender bias in English embeddings, Caliskan et al. (2017) propose Word Embedding Association Test (WEAT), which measures the association between two sets of target concepts and two sets of attributes. Let $X$ and $Y$ be equal-sized sets of target concept embeddings (e.g., words related to mathematics and art) and let $A$ and $B$ be sets of attribute embeddings (e.g., male and female definition words such as "man", "girl"). Let $\cos(\vec{a}, \vec{b})$ denote the cosine similarity between vectors $\vec{a}$ and $\vec{b}$. The test statistic is a difference between sums over the respective target concepts,

$$s(X, Y, A, B) = \sum_{\vec{x} \in X} s(\vec{x}, A, B) - \sum_{\vec{y} \in Y} s(\vec{y}, A, B),$$

where $s(\vec{w}, A, B)$ is the difference between average cosine similarities of the respective attributes,

$$s(\vec{w}, A, B) = \frac{1}{|A|} \sum_{\vec{a} \in A} \cos(\vec{w}, \vec{a}) - \frac{1}{|B|} \sum_{\vec{b} \in B} \cos(\vec{w}, \vec{b}).$$

In the following, we extend the definition to quantify individual words in gendered languages.

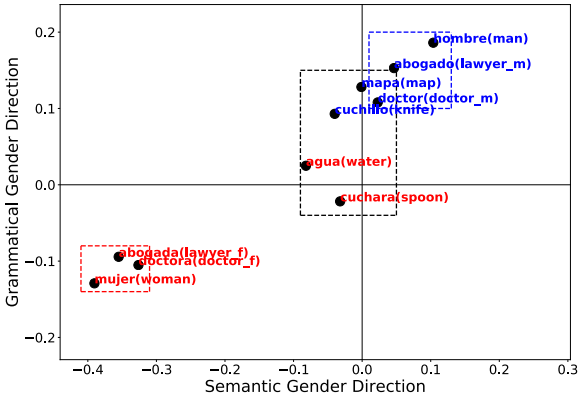


Figure 1: Projections of selected words in Spanish on grammatical and semantic directions with masculine nouns in blue and feminine nouns in red. We find that: (1) Most inanimate nouns (e.g., map, spoon; enclosed by black dotted lines) lie near the origin point of the semantic gender axis; (2) Masculine definition and occupation words (blue dotted lines) lie in the masculine side for both gender directions and so do feminine words (red dotted lines). But the feminine words are farther on the feminine side on the semantic gender direction compared to masculine words.

We consider two types of words. One is *inanimate nouns*, like "agua" (water, feminine). They are assigned as either a masculine or feminine noun. The other is *animate nouns* that usually have two gender forms, like "doctor" (male doctor) and "doctora" (female doctor).[2]

For inanimate nouns that should not be semantically leaning towards one gender, we can quantify the gender bias for word $w$ simply using WEAT,

$$\mathrm{b}_w = |s(\vec{w}, A, B)|,$$

which measures the association strength of the word $w$ with the gender concepts. The larger $\mathrm{b}_w$ is, the stronger its association with the gender concepts.

For nouns with two gender forms, we test if their masculine ($\vec{w}_m$) and feminine ($\vec{w}_f$) forms are symmetrical with respect to gender definition terms.

$$\mathrm{b}_w = ||s(\vec{w}_m, A, B)| - |s(\vec{w}_f, A, B)||. \quad (1)$$

The larger the value is, the larger gender bias presents. For example, if "doctor" (male doctor) leans more toward to masculine (i.e., close to masculine definition words like "el" (he) and far away

[2]Note that some animate nouns in certain languages, like "médecin" (male doctor) in French, only have one gender form. We omit such cases in this paper for the sake of simplicity.

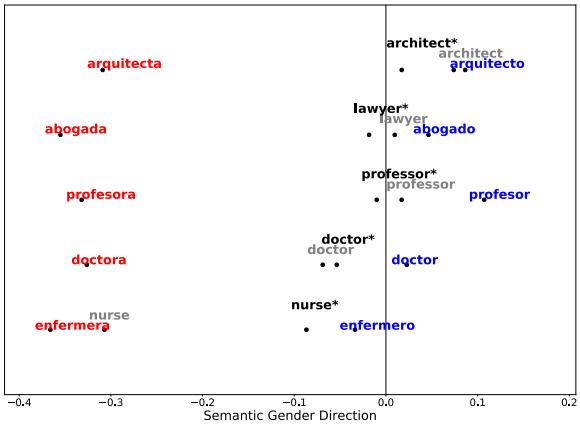


Figure 2: Projections of occupation words on the semantic gender direction in two gender forms (blue for male and red for female) in Spanish, the original English word (grey), and the bias-mitigated version of the English word (black with "*"). We find that feminine occupations are farther away from the English counterparts as well as the neutral position (indicated by the vertical line) compared to masculine words.

from feminine definition words like "ella" (she)) than "doctora" leans toward to feminine, then embeddings of "doctor" and "doctora" are biased. We will discuss detailed quantitative analysis using this metric in Section 4.

### 2.2 Bias in Bilingual Word Embeddings

Bilingual word embeddings are used widely in tasks such as multilingual transfer. It is essential to quantify and mitigate gender bias in these embeddings to avoid them from affecting downstream applications. We focus on bilingual word embeddings that align a grammatical gender language like Spanish with a language without grammatical gender like English. The definitions of the two gender directions are similar: we construct the grammatical gender direction using the same sets of words in the gendered language since the other language does not mark grammatical gender. We combine the gender-definition words for both languages and remove the grammatical gender component to get the semantic gender direction similarly.

We use bilingual word embeddings from MUSE (Conneau et al., 2018) that align English and Spanish fastText embeddings together in a single vector space. Figure 2 shows the projections of 5 occupation words on the semantic gender direction. First, we notice that the embeddings of some English occupations are much closer to one gender form than the other in Spanish. For example, "nurse" in English is closer to "enfermera" (female nurse) than "enfermero" (male nurse) in Spanish. We further separate out the bias in English embedding in the analysis by mitigating the English embeddings using "hard-debiasing" approach proposed in Bolukbasi et al. (2016) and aligning Spanish embedding with the bias-reduced English embedding (words annotated with "*"). Results in Figure 2 demonstrate another type of gender unequal in bilingual embedding: masculine occupation words are closer to the corresponding English occupation words than the feminine counterparts. This result aligns with the observation in Figure 1.

More quantitative analyses about gender bias in bilingual embedding will be discussed in Section 4.3.

## 3 Mitigation Methods

In the following, we first describe the optimization objectives for mitigating gender bias. Then, we present mitigation methods under two settings: (1) *monolingual setting*, where we only have access to word embeddings of the gendered language, and (2) *bilingual setting*, where the embeddings of a language without grammatical gender (e.g. English) aligned in the same space are also provided. Specifically, for the bilingual setting, we can use English word vectors to facilitate the bias mitigation.

### 3.1 Mitigation Objectives

For inanimate nouns, we project out their semantic gender component as they should not carry semantic gender. Equivalently, we minimize the inner product

$$\left\langle \vec{w}, \vec{d_s} \right\rangle \tag{2}$$

between the inanimate word vector $\vec{w}$ and the semantic gender direction $\vec{d_s}$.

For nouns with both gender forms, we aim at removing the gender inequality on the semantic gender direction as shown in Figure 1 and Figure 2. Specifically, we minimize the difference between the distance of feminine form of the word $\vec{w_f}$ and the masculine form of the word $\vec{w_m}$ to an anchor point (e.g., the origin point) on the semantic gender direciton $d_s$:

$$\left|\left|\left\langle \vec{w}_m, \vec{d_s} \right\rangle - \left\langle \vec{w}_a, \vec{d_s} \right\rangle\right| - \left|\left\langle \vec{w}_f, \vec{d_s} \right\rangle - \left\langle \vec{w}_a, \vec{d_s} \right\rangle\right|\right|. \tag{3}$$

Note that for most cases, $\vec{w_m}$ and $\vec{w_f}$ lie on the opposite sides of the anchor point on the semantic

gender axis. Therefore, we can simplify Eq. (3) as

$$\left|\left\langle \vec{w}_m, \vec{d}_s \right\rangle + \left\langle \vec{w}_f, \vec{d}_s \right\rangle - 2\left\langle \vec{w}_a, \vec{d}_s \right\rangle\right|. \quad (4)$$

Intuitively, this measures how much we need to move the word pair on the semantic gender direction so that they are symmetric with respect to the anchor point. If the anchor point is the origin point (i.e. $\left\langle \vec{w}_a, \vec{d}_s \right\rangle = 0$), this objective is the absolute value of the sum of two projections.

### 3.2 Monolingual Setting

**Shifting Along Semantic Gender Direction (Shift)** We mitigate bias by minimizing Eq. (4) and Eq. (2) in a post-processing manner. Effectively, for inanimate nouns, we remove the semantic gender component from the embeddings. For nouns with two gender forms (e.g., occupations), we shift the two forms along the semantic gender direction so that they have the same distance to the anchor point. For monolingual setting, we use the origin point on the gender direction as the anchor point and call the method Shift_Ori.

Note that Gonen and Goldberg (2019) show that mitigating gender bias in word embeddings by moving on the gender direction is not sufficient and words with gender bias still tend to group together. Similairly, our approach might not entirely remove the gender bias. However, the approach in Gonen and Goldberg (2019) to test the remainder bias cannot be applied directly for gendered languages, because grouping of the embeddings of masculine and feminine words does not always indicate bias due to grammatical gender .

### 3.3 Bilingual Setting

**Mitigating Before Alignment (De-Align)** As gender bias can be observed when aligning gendered languages with language without grammatical gender, we consider using English to facilitate mitigating bias. Specifically, we first apply the "hard-debasing" approach Bolukbasi et al. (2016) to English embedding and then align gendered language with the bias-reduced English. Because most words with both gender forms align to the same word in English (e.g., "enfermero" and "enfermera" in Spanish both align to "nurse" in English), the alignment places the two gender forms of the words in a more symmetric positions in the vector space after the alignment.

**Shifting Along Semantic Gender Direction (Shift)** We consider using English words as anchor positions when mitigating bias using Eq (4). We call this approach Shift_EN. For example, when applying Eq. (4) to re-position "enfermero" and "enfermera" in Spanish we take English word "nurse" aligned in the same space as the anchor position.

**Hybrid Method (Hybrid)** We also consider a hybrid method that integrates the aforementioned two approaches. In particular, we first mitigate English embeddings, align English embeddings with the embeddings of gendered languages, and then shift words in languages with grammatical gender along the semantic gender direction. We consider two variants based on how the anchor positions are chosen: (1) Hybrid_Ori uses the origin point as the anchor position; and (2) Hybrid_EN uses the corresponding bias-reduced English word as an anchor position when shifting a pair of words with different gender forms.

## 4 Experiments

We evaluate gender bias mitigation methods in both monolingual and bilingual settings and test the utility of the bias-reduced embeddings in word-level translation tasks Conneau et al. (2018). Following other analyses on gender bias (Bolukbasi et al., 2016; Caliskan et al., 2017), we quantify the gender bias based on a set of animate nouns including occupations and gender-definition words. These words are representative to the major challenge when quantifying and mitigating gender bias. Different from previous analyses in English, most occupation words in gendered languages have more than one forms and their embeddings are inherently different from each other due to morphological agreement.

### 4.1 Data and Configuration

We apply the proposed mitigation methods on Spanish-English and French-English bilingual embeddings from MUSE (Conneau et al., 2018) that align the fastText monolingual embeddings pretrained on Wikipedia corpora together in a single vector space. We collect 58 occupation words in Spanish from the occupation list provided by Font and Costa-jussà (2019) and 23 in French (some occupations in French do not distinguish between two forms) for the bias mitigation experiments. We use WEAT word lists from Caliskan et al. (2017) for

| Lang. | Metric | Original | Shift_Ori | Shift_EN | De-Align | Hybrid_Ori | Hybrid_EN |
|---|---|---|---|---|---|---|---|
| ES | MWEAT–Diff | 3.6918 | 0.3090 | 0.3324 | 3.5748 | **0.3090** | 2.2494 |
| ES | MWEAT–p-value | 0.0000 | 0.1130 | 0.0010 | 0.0010 | **0.7330** | 0.0020 |
| FR | MWEAT–Diff | 2.3437 | 0.2446 | 0.3882 | 2.3436 | **0.2446** | 1.1758 |
| FR | MWEAT–p-value | 0.0000 | 0.1470 | 0.0010 | 0.0020 | **0.5290** | 0.0910 |
| ES | Word Similarity | **0.7392** | 0.7363 | 0.7359 | **0.7392** | 0.7358 | 0.7356 |
| FR | Word Similarity | **0.7294** | 0.7218 | 0.7218 | 0.7156 | 0.7218 | 0.7218 |

Table 1: Analyses on Spanish and French monolingual embeddings before and after bias mitigation. Results show that the original Spanish and French embeddings exhibit strong bias and Hybrid_Ori significantly reduces the bias in the embedding to to an insignificant level (p-value $> 0.05$).

male and female attribute words for our WEAT experiments and translate them to Spanish and French. For the word pair translation task, we use 7 common adjectives and pair every adjective with each occupation, resulting in 406 pairs for Spanish and 161 for French.

### 4.2 Monolingual Experiments

**Modified Word Embedding Association Test (MWEAT)** As discussed in Section 2, we evaluate the gender bias using modified WEAT and we treat the absolute value of the sum of differences between mean cosine similarities of the two gender attributes for these occupation words $\left|\sum_{\vec{x}\in X} s(\vec{x}, A, B)\right|$ as the association of male occupation target concepts $X$ with the gender attributes. Similarly, we use $\left|\sum_{\vec{y}\in Y} s(\vec{y}, A, B)\right|$ as the association of female occupation words with the gender attributes. Finally our test statistic is

$$\left|\left|\sum_{\vec{x}\in X} s(\vec{x}, A, B)\right| - \left|\sum_{\vec{y}\in Y} s(\vec{y}, A, B)\right|\right|,$$

which measures the difference in the association strength for two target sets with the two attribute sets occupation words. We then calculate the one-side p-value of the permutation test.

**Word Similarity** We test the quality of the embeddings after mitigation on the SemEval 2017 word similarity task (Camacho-Collados et al., 2017) for monolingual embeddings. This task evaluates how well the cosine similarity between two words correlates with a human-labeled score for which we report the Pearson correlation score.

**Results** Table 1 shows that Hybrid_Ori significantly decreases the difference of association between two genders as indicated by MWEAT–Diff and p-value. Other methods only show marginal

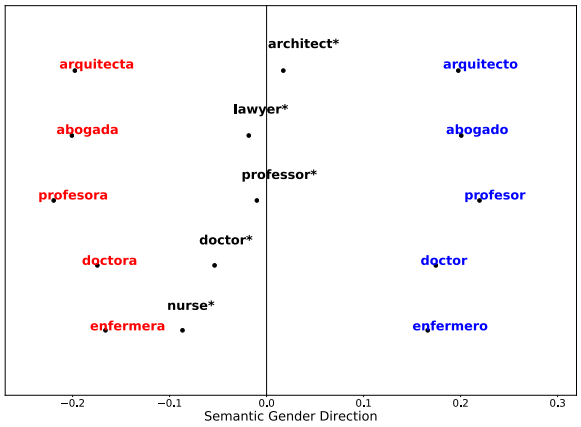


Figure 3: Projections of occupations words on the semantic gender direction after hybrid mitigation with origin as the anchor point. The two forms of occupations are a lot more symmetric.

mitigation effects in terms of p-value. We also find that the performance of our mitigation methods on word similarity is largely preserved as indicated by the Pearson correlation scores.

### 4.3 Cross-lingual Experiments

**Word Translation** We use word translation to examine whether our mitigation approaches preserve the utility of the bilingual word embedidings. This task aims to retrieve the translation of a source word in the target language. We use the test set provided by Conneau et al. (2018), which contains 200,000 randomly selected candidate words for 1,500 query words. We translate a query word and report the precision at $k$ (i.e., fraction of correct translations that are ranked not larger than $k$) by retrieving its $k$ nearest neighbours in the target language. We report both results with $k = 1$ and $k = 5$.

**Word Pair Translation by Analogy** We propose a word pair translation task to evaluate the bias

| Task | Metric | Original | Shift_Ori | Shift_EN | De-Align | Hyrid_Ori | Hyrid_EN |
|---|---|---|---|---|---|---|---|
| Word Translation | | | | | | | |
| EN→ES | P@1/P@5 | 79.2/89.0 | **80.7/90.3** | **80.7/90.3** | 76.5/88.9 | **80.7/90.3** | **80.7/90.3** |
| ES→EN | P@1/P@5 | 79.2/89.0 | 79.2/89.0 | 79.2/89.0 | **80.1/90.7** | 79.2/89.0 | 79.2/89.0 |
| EN→FR | P@1/P@5 | 78.2/89.4 | **79.9/91.1** | **79.9/91.1** | 74.3/87.8 | **79.9/91.1** | **79.9/91.1** |
| FR→EN | P@1/P@5 | **76.1/88.1** | **76.1/88.1** | **76.1/88.1** | 74.4/87.2 | **76.1/88.1** | **76.1/88.1** |
| Word Pair Translation | | | | | | | |
| EN→ES | ASD | 0.1082 | 0.0961 | 0.0961 | 0.0827 | **0.0755** | 0.0772 |
| | F_MRR | 0.2073 | 0.2507 | 0.2507 | 0.2919 | 0.3450 | 0.3150 |
| | M_MRR | 0.6940 | 0.6766 | 0.6766 | 0.6775 | 0.6398 | 0.6696 |
| | MRR Diff | 0.4867 | 0.4259 | 0.4259 | 0.3856 | **0.2949** | 0.3546 |
| EN→FR | ASD | 0.1208 | 0.1048 | 0.1082 | 0.0892 | **0.0735** | 0.0805 |
| | F_MRR | 0.1663 | 0.2101 | 0.1943 | 0.2679 | 0.3128 | 0.2975 |
| | M_MRR | 0.6549 | 0.6313 | 0.6419 | 0.6610 | 0.6393 | 0.6467 |
| | MRR Diff | 0.4886 | 0.4212 | 0.4476 | 0.3931 | **0.3265** | 0.3492 |

Table 2: Results on word translation and word pair translation based on Spanish-English and French-English bilingual embeddings. We find that the original bilingual embeddings have a large discrepancy between the two genders, indicated by the average cosine similarity difference (ASD) and mean reciprocal ranks (MRR) difference. After applying the mitigation methods, both ASD and MRR difference drop.

in gendered languages. In languages with grammatical gender, word forms of articles and adjectives need to agree with the gender of nouns they are associated with. For example, the word "good" has two forms in Spanish: "bueno" (masculine form) and "buena" (feminine form). Following the notion of analogy, we design an evaluation task to test how bilingual word embeddings translate an English occupation with the presence of another word. Specifically, given an English word $E_i$ (a noun, an adjective or a verb), an English occupation word $E_o$, and the corresponding Spanish/French translation $S_i$ of $E_i$, we adopt the analogy test "$E_i$:$E_o$ = $S_i$:?" (e.g., "good:doctor = buena: ?") to rank all the words $S_o$ in Spanish/French based on the cosine similarity: $\cos(S_o, E_o - E_i + S_i)$. Ideally, the correct translation of $E_o$ that agree with gender of $S_i$ should be ranked on the top ("doctora" in the above example). Quantitatively, we use the mean reciprocal ranks (MRR) to evaluate the performance of the ranking. If the bilingual embedding is not biased, the model should perform similarly on answering the translation of masculine and feminine occupations. Based on this, we use the gap between the performances on two form of genders to quantify the bias in the bilingual embedding space. Besides, we also report the average cosine similarity difference (ASD) of occupation words between two different gender forms.

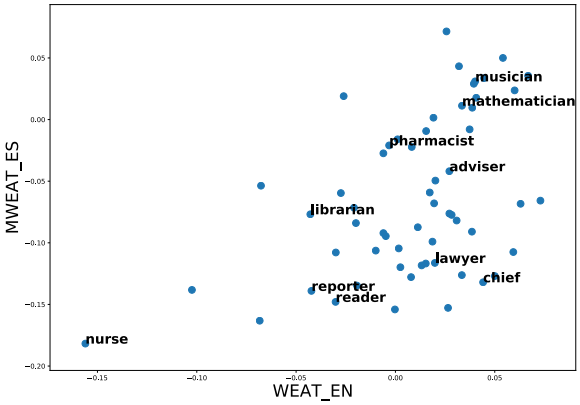


Figure 4: (M)WEAT scores for occupations in English and Spanish. For both axes, the smaller the number, the occupation is more leaning towards to female.

**Results** Table 2 shows the results. Results on word translation shows that the mitigation approaches do not harm the utility of bilingual word embedding and words in different languages still align well. From the results of word pair translation, the original bilingual embedding exhibits strong bias and the gap between two genders are around 0.49 in MRR in both ES-EN and FR-EN bilingual embeddings and the average cosine similarity difference is larger than 0.1. Hybrid_Ori results in smallest difference for cosine similarity and MRR gap between two gender forms. However, the discrepancy between two forms is not completely removed.

To visualize the effect of bias mitigation, we

project the bias-reduced embeddings of occupation words on the semantic gender direction as shown in Figure 2. Figure 3 shows that after debiasing using Hybrid_Ori, the two gender forms of occupation words are more symmteric with respect to the corresponding English embedding.

### 4.4 Bias Correlation between Languages

Finally, we compute the WEAT/MWEAT scores[3] for each occupation and show the commonality and difference of gender bias between English and Spanish embedding. Results in Figure 4 shows that occupations ("nurse" and "mathematician") lean towards the same gender in both languages. However, some occupations (e.g., "chef") lean to different genders in different languages. In general, there is a strong correlation between gender bias in English and Spanish embedding (Spearman correlation=0.45 with p-value=0.0004).

## 5 Related Work

Previous work has proposed to quantify bias in English Embedding definitions for gender bias in English word embeddings. Besides the aforementioned studies (Bolukbasi et al., 2016; Caliskan et al., 2017) in quantifying bias in English embedding, McCurdy and Serbeti (2017) examine grammatical gender bias in word embeddings of gendered languages by computing the WEAT association score (Caliskan et al., 2017) between gendered object nouns (e.g. moon-sun) and gender-definition words. They propose to mitigate bias by applying lemmatization to remove gender information from the training corpus. However, their focus is different from us as our approaches aim at keeping the grammatical gender information and only removing the bias in semantic genders. A few recent studies focus on measuring and reducing gender bias in contextualized word embeddings (Zhao et al., 2019; May et al., 2019; Basta et al., 2019). However, they only focus on English embeddings in which the gender is mostly only expressed by pronouns (Stahlberg et al., 2007). An interesting future direction is to extend their analyses to language with grammatical gender.

Regarding bias mitigation appraoches, Zhao et al. (2018b) mitigate bias by saving one dimension of the word vector for gender. Bordia and Bowman (2019) propose a regularization loss term for word-level language models. Zhang et al. (2018) use an adversarial network to mitigate bias in word embeddings. All these approaches consider only English embeddings. Moreover, Gonen and Goldberg (2019) show that mitigation methods based on gender directions are not sufficient, since the embeddings of socially-biased words still cluster together.

Bias in word embedding may affect the downstream applications (Zhao et al., 2018a; Rudinger et al., 2018; Font and Costa-jussà, 2019). Besides the gender bias in word embeddings, implicit stereotypes have been shown in other real world applications, such as online reviews (Wallace and Paul, 2016), advertisement (Sweeney, 2013) and web search (Kay et al., 2015).

## 6 Conclusion and Future Work

We analyze gender bias in the embeddings of languages with grammatical gender and bilingual word embeddings that align such a language with a language without grammatical gender. We propose new methods to evaluate and mitigate gender bias for languages with grammatical gender and bilingual word embeddings and results show that our methods can mitigate gender bias while preserving the quality of original embeddings.

Directions for future work include testing monolingual and bilingual word embeddings on downstream tasks like machine translation to measure bias and also test the performance for mitigation methods. Moreover, the number of noun classes for other languages with grammatical gender ranges from two to several tens (Corbett, 1991) and future work can extend methods proposed in this paper to address grammatical gender in languages with more gender forms.

## 7 Acknowledgements

This work was supported in part by DARPA (HR0011-18-9-0019). Ryan Cotterell was supported by a Facebook Fellowship. We acknowledge fruitful discussions with Fred Morstatter, Nanyun Peng, Ninareh Mehrab, Shunjie Wang, Yizhou Yao, and Aram Galstyan. We also thank anonymous reviewers for their comments.

[3] We use WEAT score for English and MWEAT for Spanish. To show the direction of bias, we take out the outer absolute function in Eq. (1).

## References


Christine Basta, Marta R Costa-jussà, and Noe Casas. 2019. Evaluating the underlying gender bias in contextualized word embeddings. *arXiv preprint arXiv:1904.08783*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Advances in Neural Information Processing Systems*, pages 4349–4357.

Shikha Bordia and Samuel R Bowman. 2019. Identifying and reducing gender bias in word-level language models. In *NAACL Student Research Workshop*.

Aylin Caliskan, Joanna J Bryson, and Arvind Narayanan. 2017. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186.

Jose Camacho-Collados, Mohammad Taher Pilehvar, Nigel Collier, and Roberto Navigli. 2017. Semeval-2017 task 2: Multilingual and cross-lingual semantic word similarity. In *International Workshop on Semantic Evaluation*, pages 15–26.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. In *International Conference on Learning Representations*.

Greville G. Corbett. 1991. *Gender*. Cambridge University Press.

Greville G Corbett. 2006. *Agreement*, volume 109. Cambridge University Press.

Sunipa Dev and Jeff Phillips. 2019. Attenuating bias in word vectors. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 879–887.

Kawin Ethayarajh, David Duvenaud, and Graeme Hirst. 2019. Understanding undesirable word embedding associations. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 1696–1705.

Ronald A Fisher. 1936. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188.

Joel Escudé Font and Marta R Costa-jussà. 2019. Equalizing gender biases in neural machine translation with word embeddings techniques. *arXiv preprint arXiv:1901.03116*.

Hila Gonen and Yoav Goldberg. 2019. Lipstick on a pig: Debiasing methods cover up systematic gender biases in word embeddings but do not remove them. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Ian Jolliffe. 2011. *Principal component analysis*. Springer.

Matthew Kay, Cynthia Matuszek, and Sean A Munson. 2015. Unequal representation and gender stereotypes in image search results for occupations. In *ACM Conference on Human Factors in Computing Systems*, pages 3819–3828.

Chandler May, Alex Wang, Shikha Bordia, Samuel R Bowman, and Rachel Rudinger. 2019. On measuring social biases in sentence encoders. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Katherine McCurdy and Oguz Serbeti. 2017. Grammatical gender associations outweigh topical gender bias in crosslinguistic word embeddings. *Widening NLP*.

Rachel Rudinger, Jason Naradowsky, Brian Leonard, and Benjamin Van Durme. 2018. Gender bias in coreference resolution. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, pages 8–14.

Dagmar Stahlberg, Friederike Braun, Lisa Irmen, and Sabine Sczesny. 2007. Representation of the sexes in language. *Social communication*, pages 163–187.

Latanya Sweeney. 2013. Discrimination in online ad delivery. *arXiv preprint arXiv:1301.6822*.

Byron C Wallace and Michael J Paul. 2016. jerk or judgemental? patient perceptions of male versus female physicians in online reviews.

Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. 2018. Mitigating unwanted biases with adversarial learning. In *AAAI/ACM Conference on AI, Ethics, and Society*, pages 335–340.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Ryan Cotterell, Vicente Ordonez, and Kai-Wei Chang. 2019. Gender bias in contextualized word embeddings. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2018a. Gender bias in coreference resolution: Evaluation and debiasing methods. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Jieyu Zhao, Yichao Zhou, Zeyu Li, Wei Wang, and Kai-Wei Chang. 2018b. Learning gender-neutral word embeddings. In *Empirical Methods in Natural Language Processing*, pages 4847–4853.

# Weakly Supervised Cross-lingual Semantic Relation Classification via Knowledge Distillation

**Yogarshi Vyas**
Department of Computer Science
University of Maryland
yogarshi@cs.umd.edu

**Marine Carpuat**
Department of Computer Science
University of Maryland
marine@cs.umd.edu

## Abstract

Words in different languages rarely cover the exact same semantic space. This work characterizes differences in meaning between words across languages using semantic relations that have been used to relate the meaning of English words. However, because of translation ambiguity, semantic relations are not always preserved by translation. We introduce a cross-lingual relation classifier trained only with English examples and a bilingual dictionary. Our classifier relies on a novel attention-based distillation approach to account for translation ambiguity when transferring knowledge from English to cross-lingual settings. On new English-Chinese and English-Hindi test sets, the resulting models largely outperform baselines that more naïvely rely on bilingual embeddings or dictionaries for cross-lingual transfer, and approach the performance of fully supervised systems on English tasks.

## 1 Introduction

Natural Language Processing (NLP) often uses translation lexicons for projecting models and data from one language to another under the assumption that words and their translations in these lexicons are synonyms (Mayhew et al., 2017; Tsvetkov et al., 2014). However, translation lexicons include semantic relations other than synonymy in practice, as can be seen (Table 1) in examples drawn from the MUSE dictionary (Lample et al., 2018). Peirsman and Padó (2011) show that distributional translation lexicons contain hyponyms and co-hyponyms, and that treating all translations as synonyms hurts cross-lingual projection performance. In the Paraphrase Database (Ganitkevitch et al., 2013), Pavlick et al. (2015) find that the diversity of semantic relations discovered in word-aligned parallel corpora yields paraphrases that span the lexical relations defined in the natural logic framework of MacCartney and Manning (2009). These non-synonymous translations are not just noise. They can be found even in high-quality parallel corpora, since the strategies used by professional translators to deal with words that do not have a direct equivalent in the target language include replacement by near-synonyms, hypernyms or negated antonyms (Baker, 2011; Venuti, 2012; Chesterman, 2016).

| Lexicon Entry | Semantic Relation |
|---|---|
| *writer*, रचनाकार | *writer* is more specific than रचनाकार (creator) |
| *council*, मंत्रिपरिषड | *council* is more general than मंत्रिपरिषड (council of ministers) |
| *father*, चचा | *father* is mutually exclusive to चचा (father's brother) |

Table 1: Semantic relations between word pairs in an English-Hindi lexicon (Lample et al., 2018)

In this work, we classify semantic relations between words in different languages. Given a word pair (water, पय), the classification task is to select one of the five entailment classes (Figure 1) defined under the natural logic framework of MacCartney and Manning (2009). This cross-lingual task cannot be solved by translation, as translation does not preserve semantic relations. We also cannot assume that labeled examples exist for all language pairs and learning from English labeled examples is complicated by translation ambiguity (Figure 1).

We introduce BILEXNET, a neural classifier for semantic relations based on cross-lingual distributional and path-based features inspired by the monolingual LEXNET model (Shwartz and Dagan, 2016a,b) (Section 3). We then design a novel training procedure for BILEXNET that leverages weak supervision in the form of examples translated from English via a knowledge distillation technique guided by translation dic-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5285–5296, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

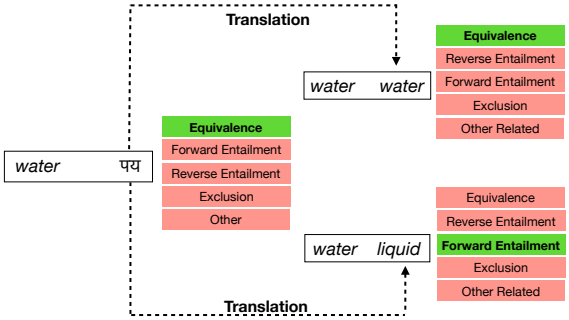


Figure 1: On the left, we illustrate cross-lingual semantic relation classification: given the pair (*water*, पय) as input, the task is to select the `Equivalence` class (in bold/green) from the five possible relations. On the right, we show that semantic relations change by translation. पय translates to *liquid* and *water*, and their respective semantic relations with *water* differ.

tionaries (Hinton et al., 2014) (Section 4). We collect and release MULTILEXREL, a crowd-sourced benchmark to evaluate models for this task on a high-resource (English-Chinese) and a low-resource (English-Hindi) language pair (Section 5). Experiments show that BILEXNET substantially outperforms translation baselines and approaches the performance of a fully supervised English semantic relation classifier (Section 7). Code for BILEXNET and the MULTILEXREL dataset are available at `https://github.com/yogarshi/BiLexNet/`.

## 2 Related Work

Cruse (1986) describes how lexical relations can be organized into congruence relations of identity (synonymy: dog-canine), inclusion (hyponymy: dog-animal), overlap (compatibility: dog-pet), disjunction (incompatibility: cat-dog), and antonyms (open-shut), and how these relations can be used to characterize differences in meaning. Variations on these fundamental relations have been used within semantic networks such as WordNet (Fellbaum, 1998), or as the basis of a framework for inference without formal logic representations (MacCartney and Manning, 2009). Recent work on semantic relation prediction largely focuses on a single relation between words in the same language (mostly English) (Nastase et al., 2013; Vulić and Mrkšić, 2018; Glavaš and Ponzetto, 2017; Ono et al., 2015). Methods that deal with multiple semantic relations are fewer (Pantel and Pennacchiotti, 2006; Pennacchiotti and Pantel, 2006; Turney, 2008), and recent shared tasks have shown that this is a challenging problem, especially when ontologies and other structured resources are not available, and models are trained only on raw corpora (Santus et al., 2016).

In cross-lingual settings, studies of semantic relations between words are mostly limited to the translation equivalence relation. Dictionary induction aims to automatically discover words that are translations of each other using monolingual or comparable corpora (Rapp, 1995; Fung and Yee, 1998; Irvine and Callison-Burch, 2017). The task is typically framed as unsupervised learning, and models rely on distributional properties to discover words that have the same meaning in the two languages. Recently, dictionary induction has also been used to evaluate multilingual word embeddings (Lample et al., 2018; Artetxe et al., 2018; Søgaard et al., 2018), leading to significant advances on the state-of-the-art.

However, Peirsman and Padó (2011) show that automatically induced bilingual lexicons exhibit multiple semantic relations including not only synonymy, but also hypernymy and co-hyponymy. This has prompted work on identifying hypernymy in cross-lingual settings (Vyas and Carpuat, 2016; Upadhyay et al., 2018). This paper broadens the scope of relations studied in cross-lingual settings, and addresses for the first time the task of distinguishing between multiple semantic relations for words in different languages.

Previous work studying semantic relations in multiple languages has focused on the different task of cross-lingual transfer. In such settings, the focus is on identifying semantic relations between two words in the *same* language, without training data in that language. There are broadly two strategies to solve this problem. One line of work (Glavaš and Vulić, 2018; Mrkšić et al., 2017) uses model transfer, where a single model is trained on data from a high-resource language, and is then ported to the target language using cross-lingual embeddings. In contrast, Roth and Upadhyay (2019) translate training data from English into a target language using a combination of unsupervised cross-lingual embeddings (Artetxe et al., 2018) and monolingual information from the target language.

Our approach for cross-lingual semantic relation classification builds on the monolingual classifier LEXNET (Shwartz and Dagan, 2016a,b), which achieved the highest performance (45 F1)

among participating teams on the CogALex-V shared task on identification of semantic relations (Santus et al., 2016) without ontologies or structured information. We adapt LEXNET to make cross-lingual predictions by proposing to model cross-lingual relations using lexico-syntactic paths from both languages.

Finally, our training procedure uses knowledge distillation (Hinton et al., 2014) to alleviate the lack of annotated cross-lingual pairs. Knowledge distillation has been proposed to compress a model with many parameters (the *teacher* model) to a model with fewer parameters (the *student* model). It has also been used successfully to learn mappings between languages (Nakashole and Flauger, 2017) or to transfer knowledge from models trained on one language to a different target language for text classification (Xu and Yang, 2017) and belief tracking (Chen et al., 2018a), in settings where the classification labels are translation invariant. This work adapts distillation to a setting where labels might change when samples are translated.

## 3 BILEXNET: a Classifier for Cross-Lingual Semantic Relations

The task of classifying semantic relations is a multi-class classification problem, where the classes are the set of five semantic relations from Pavlick et al. (2015): `Equivalence` (X is the same as Y), `Forward Entail` (X is more specific than/ is a type of Y), `Backward Entail` (X is more general than/encompasses Y), `Exclusion` (X is mutually exclusive with/is opposite to Y), and `Other` (X is not related or related in other ways to Y). We choose these relations as they have been useful in describing lexical relations between English paraphrases (Pavlick et al., 2015), and in downstream natural language inference systems (MacCartney and Manning, 2007, 2009).

Our classifier, BILEXNET, adapts the LEXNET English classifier (Shwartz and Dagan, 2016a,b) to cross-lingual settings. BILEXNET represents the input word pair ($x$, $y$) by a feature vector $\mathbf{v}_{xy}$, consisting of complementary distributional and path-based features i.e. $\mathbf{v}_{xy}$= [$\mathbf{v}_x$;$\mathbf{v}_y$;$\mathbf{v}_{paths(x,y)}$]. The *distributional semantic* properties of $x$ and $y$ are captured by bilingual word embeddings $\mathbf{v}_x$ and $\mathbf{v}_y$. $\mathbf{v}_{paths(x,y)}$ encodes *lexico-syntactic paths* that represent the relation between words $x$ and $y$ in context (Hearst, 1992; Snow et al., 2004; Shwartz et al., 2016). For classification, $\mathbf{v}_{xy}$ is input to a multi-class classifier, parameterized as a feed-forward neural network with a single hidden layer.

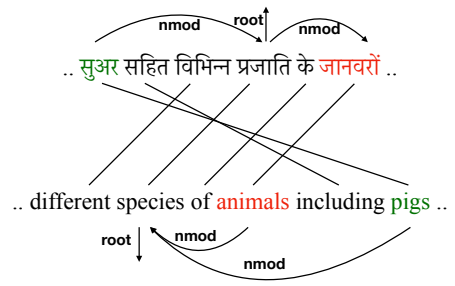


Figure 2: The English path between *animals* and *pigs* has three edges: [`X/NOUN/nmod/>`, `species/NOUN/root/^`, and `Y/NOUN/nmod/>`]. The path between animals and सुअर is defined as a combination of the English path and the Hindi path between जानवरों and सुअर.

$$\begin{aligned} \mathbf{l}_{out} &= \mathbf{W}_2 * \text{ReLU}(\mathbf{W}_1 * \mathbf{v}_{xy}) \\ \hat{l}_i &= \frac{\exp(l_{out,i})}{\sum_{j=1}^{k} \exp(l_{out,j})} \\ l_{pred} &= \arg\max_i \hat{l}_i \end{aligned} \quad (1)$$

$\mathbf{W}_1$ and $\mathbf{W}_2$ are the weights of the network, and the biases have been omitted for simplicity.

### 3.1 Cross-lingual Paths

In LEXNET, a lexico-syntactic path is the *sequence* of edges that lead from $x$ to $y$ in the dependency tree of a sentence. Each edge contains the word and part-of-speech tag of the source node, the dependency label of the edge, and the edge direction between two subsequent nodes (see Figure 2 for an example of the path connecting *pigs* and *animals*). The vector representation of each edge is formed by the concatenation of the vectors of these four components. $\mathbf{v}_{paths(x,y)}$ is obtained by encoding the sequence of edges using an LSTM (Hochreiter and Schmidhuber, 1997).

In English, these paths are extracted from sentences where $x$ and $y$ co-occur. However, when $x$ and $y$ are in different languages, a new path definition is required. For a cross-lingual pair ($x_e$,$y_f$), we extract cross-lingual paths $\mathbf{v}_{paths(x_e,y_f)}$ from a word-aligned parallel corpus (Figure 2). We first extract all parallel sentences which contain $x_e$ on the source side and $y_f$ on the target side. For each sentence, using word alignments, we can extract $x_f$, the target word aligned to $x_e$, and $y_e$ the source

word aligned to $y_f$. We then extract a path connecting the two word in the source sentence i.e. $x_e$ and $y_e$. Similarly, we also extract a corresponding path connecting the two word in the target sentence i.e. $x_f$ and $y_f$, since different languages can encode the same information differently due to structural divergences (Dorr, 1994). Thus, if the parallel corpus contains $m$ sentence pairs where $x_e$ occurs on the source side and $y_f$ on the target side, we extract a total of $2m$ paths. All of the $2m$ paths paths are encoded using a single LSTM, and averaged to form $\mathbf{v}_{paths(x_e,y_f)}$.

Two special cases arise from this definition. First, a path can be a single alignment link if $x_e$ and $y_f$ are aligned to each other i.e. $x_f = y_f$ and $y_e = x_e$. Second if no path is found in the corpus, $\mathbf{v}_{paths(x_e,y_f)}$ is set to the zero vector.

## 4 Weakly Supervised Training via Knowledge Distillation

Cross-lingual examples that would enable fully supervised training of BILEXNET are hard to obtain: examples of relations such as synonymy or hypernymy can be derived from multilingual WordNets (Bond and Foster, 2013), but such resources are not available for many languages, and only cover a subset of semantic relations. Instead, we introduce a dictionary-guided variant of *knowledge distillation* to train BILEXNET. This procedure only relies on a set of monolingual labeled examples that are readily available for various lexical relations in English, and a translation dictionary that maps words in the source language to the target language.

Our approach transfers knowledge from a monolingual *teacher model* to a cross-lingual *student model*. The **teacher model** is a monolingual LEXNET model (say $M_e$) trained on the source-language examples $S = \{(x_{e;i}, y_{e;i}, \mathbf{l}_i)\}$. Here, $x_{e;i}$ and $y_{e;i}$ are a pair of words in the same language and $\mathbf{l}_i \in \mathbb{R}^c$ is a one-hot encoding of the relation between $x_{e;i}$ and $y_{e;i}$ (the number of possible relations is $c$). Given $(x_{e;i}, y_{e;i}, \mathbf{l}_i) \in S$, $M_e$ is trained by minimizing the cross-entropy loss between the predicted output $\hat{\mathbf{l}}^{e\to e}$ and the gold label $\hat{\mathbf{l}}_i$:

$$L_1 = -\sum_{j=1}^{c} l_{ij} \log \hat{l}_j^{e\to e} \tag{2}$$

BILEXNET plays the role of the **student model** (denoted $M_{ef}$) and is trained to make predictions that agree with those of the teacher model. The student model is trained using **weak supervision** which is obtained by using a bilingual dictionary $D$ to translate the right side of each training pair into the target language $S$ to obtain $S' = \{(x_{e;i}, T_i, \mathbf{l}_i)\}$, where $T_i = \{t_{i1}, t_{i2}, .., t_{in}\}$ is the set of translations of $y_{e;i}$ in $D$. $S'$ serves as weak supervision because semantic relations are not translation invariant (Figure 1), and hence the label $\mathbf{l}_i$ is not correct for every $(x_{e;i}, t_{ik})$ pair.

To extract useful training signals from the weak supervision, we use an **attention mechanism** which guides the model to attend to translations that preserve the monolingual label. The attention component constructs the input representation for the cross-lingual model $M_{ef}$ in Equation 1 by averaging representations for all translation candidates, giving more weight to those that are likely to preserve the monolingual label. Given a training sample $(x_{e;i}, T_i, \mathbf{l}_i) \in S'$ with $T_i = \{t_{i1}, t_{i2}, .., t_{in}\}$, the score for a candidate translation $t_{ik}$ is calculated using the word embeddings of $x_{e;i}$ and $t_{ik}$, along with $\mathbf{l}$, an embedding of the gold label $\mathbf{l}_i$ as features to a feed-forward network $f$ (with one hidden layer). $\mathbf{l}$ is provided to help select translations that are consistent with the correct label for the monolingual pair. The scores for all translations are converted to probabilities using the softmax function, and the input features $\mathbf{v}'_{x_{e;i}y_{e;i}}$ for the student model $M_{ef}$ are a sum of the features obtained from each of these translations, weighted by the probabilities.

$$score((x_{e;i}, t_{ik}), \mathbf{l}_i) = f([\mathbf{x}_i; \mathbf{t}_{ik}; \mathbf{l}]) \tag{3}$$

$$p(t_{ik}) = \frac{\exp(score(t_{ik}))}{\sum_{j=1}^{n} \exp(score(t_{ij}))}$$

$$\mathbf{v}'_{x_{e;i}y_{e;i}} = \sum_{k=1}^{n} p(t_{ik}) \mathbf{v}_{x_{e;i}t_{ik}} \tag{4}$$

The student model is then trained to maximize the **distillation objective**:

$$L_2 = -\sum_{j=1}^{c} [(1-\alpha)\, l_{ij} \log \hat{l}_j^{e\to f} + \alpha\, \hat{l}_j^{e\to e} \log(\frac{\hat{l}_j^{e\to f}}{\hat{l}_j^{e\to e}})] \tag{5}$$

where $\hat{\mathbf{l}}^{e\to e}$ is calculated using $M_e$, $\hat{\mathbf{l}}^{e\to f}$ is calculated using the attended representation $\mathbf{v}'_{x_{e;i}y_{e;i}}$ as input to $M_{ef}$ and $\alpha$ is an interpolation parameter. The first term is again a cross-entropy loss

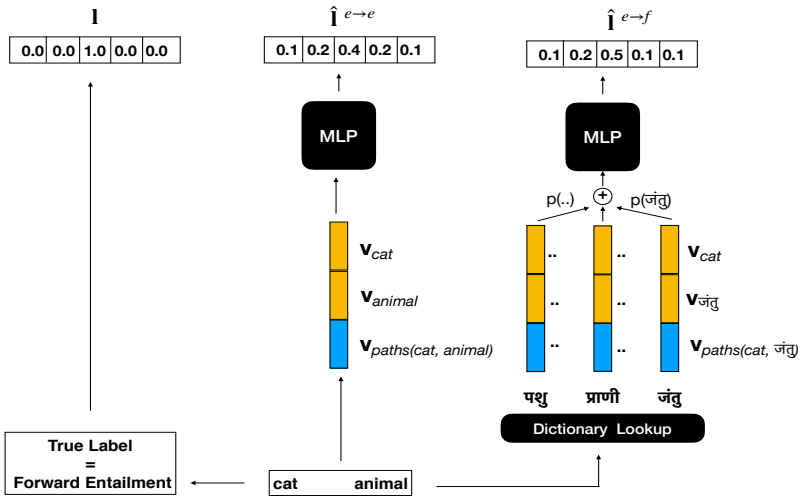


Figure 3: Illustration of weakly supervised training: For a given English example (*cat*, *animal*), we generate predictions $\hat{\mathbf{l}}^{e \to e}$ using the monolingual English teacher model. Simultaneously, we also generate predictions $\hat{\mathbf{l}}^{e \to f}$ using the cross-lingual *student* model after translating one of the two English words using a dictionary. The cross-lingual classifier attends to all translation candidates and predicts a class based on a weighted average of their features. The loss is defined as CROSS-ENTROPY($\hat{\mathbf{l}}^{e \to e}$, l) + CROSS-ENTROPY($\hat{\mathbf{l}}^{e \to f}$, l) + KL-DIVERGENCE($\hat{\mathbf{l}}^{e \to f}$, $\hat{\mathbf{l}}^{e \to e}$).

that aims to measure how well the cross-lingual model $M_{ef}$ predicts the relation given $\mathbf{v}'_{x_{e;i}y_{e;i}}$. The second term uses KL-Divergence (Kullback and Leibler, 1951) to penalize differences in predictions by $M_{ef}$ on the cross-lingual input $\mathbf{v}'_{x_{e;i}y_{e;i}}$ and the predictions by $M_e$ on the monolingual input $\mathbf{v}_{x_{e;i}y_{e;i}}$. As is typical in distillation, we flatten the softmax of both inputs to the KL-Divergence term with a temperature parameter $\tau$.

## 5 MULTILEXREL: A Dataset for Cross-lingual Semantic Relations

Existing resources containing annotated cross-lingual lexical relations are limited in scope, quality, and quantity. For instance, in previous work (Upadhyay et al., 2018), we provide datasets annotated cross-lingual hypernymy, but do not consider other relations. On the other hand, resources such as bilingual dictionaries or the Open Multilingual WordNet (Bond and Foster, 2013) can be mined for examples of synonyms, hypernyms and hyponyms, but these are likely to be noisy as these resources are created in a semi-automatic way.

In this work, we crowdsource[1] MULTILEXREL, a set of new high-quality annotations for English-Hindi (En-Hi) and English-Chinese (En-Zh) word pairs using the natural logic relations laid out in Section 3. We leverage monolingual annotations

| Relation | En-Hi | En-Zh |
|---|---|---|
| Equivalence | 158 | 174 |
| Forward Entail | 220 | 240 |
| Backward Entail | 215 | 236 |
| Exclusion | 124 | 154 |
| Other | 323 | 94 |
| Total | 1040 | 898 |

Table 2: Distribution of the five semantic relations for the two crowdsourced test sets.

to speed up the process and enable comparisons between monolingual and cross-lingual models. We use Google Translate to translate one side of a randomly sampled subset of the gold-standard dataset of semantic relations created by Pavlick et al. (2015), and ask crowdworkers whether the semantic relation holds after translation. Each example is annotated by five annotators and annotations are aggregated using MACE (Hovy et al., 2013), a Bayesian model that estimates the trustworthiness of annotators and accordingly assigns a label to each instance. The distributions of the five relations for each language pair are shown in Table 2. 40-45% of the examples shown to annotators were deemed to not preserve the monolingual relation after translation. The final test sets consist of the remaining 55-60% examples.[2]

[1] Via http://figure-eight.com/

[2] A detailed description is in the appendix.

## 6 Experimental Settings

MULTILEXREL is used as a test set to evaluate our models. Training only requires English labeled examples, and other resources derived from raw monolingual and parallel corpora.

### 6.1 Data

**English Supervision** The English training samples are derived from the English Lexical-XXXL PPDB. After filtering away pairs containing non-alphabetic characters, we choose a random sample as training pairs. The number of samples for all classes is balanced, except `Exclusion` (since there are fewer examples of this class in PPDB). All in all, the size of the training set is ∼20K pairs. Like previous work (Levy et al., 2015), we ensure a lexical split where the English words in the test data are not seen in the training data. This makes the task challenging as it prevents the model from memorizing patterns of words such as their "prototypicality" for certain relations i.e. whether certain words are likely to appear in specific relations.

**Validation data** Since we assume no access to labeled cross-lingual examples, we need to define a validation set using the resources available to us. We construct a validation set by randomly removing 1000 pairs from the training data, and automatically translating the right side of each example with the bilingual dictionary used for training. This process yields a noisy validation set, which is solely used for tuning hyper-parameters.

**Unlabeled Resources** The bilingual dictionary for knowledge distillation is obtained from the MUSE project (Lample et al., 2018) for En-Hi, while the MDBG dictionary is used for En-Zh.[3] We use FastText bilingual embeddings (Bojanowski et al., 2017).[4] We extract English paths for the monolingual model from a dump of the English Wikipedia.[5] Cross-lingual paths are extracted from a random sample of the WMT18 parallel corpora[6] for En-Zh (∼5M sentences) and the IIT Bombay English-Hindi corpus (Kunchukuttan et al., 2018) for En-Hi (∼1M sentences). All corpora are parsed using YaraParser (Rasooli and Tetreault, 2015) trained on the treebank of the corresponding language from the Universal Dependencies (v2.2) project (McDonald et al., 2013). Tokenization is performed using the Moses tokenizer for English (Koehn et al., 2007), the Indic NLP tokenizer for Hindi,[7] and the Jieba word segmenter for Chinese.[8]

### 6.2 Model Configurations and Baselines

**Model Configuration** The path-encoder LSTM has two layers with 60 hidden units each, with dropout (Srivastava et al., 2014) applied after the first layer. All feed-forward neural networks have a single hidden layer with 50 hidden units and dropout regularization. All models are trained in mini-batches of size 4 using the Adam optimizer (Kingma and Ba, 2015) with initial learning rate set to $10^{-3}$. The temperature parameter $\tau$ for knowledge distillation is tuned over $\{1.0, 1.5, 2.0, 5.0\}$, and the interpolation parameter $\alpha$ over $\{0.75, 0.90\}$.

**English-only Model: EnLexNet** We also use a vanilla LEXNET model applied to a monolingual test set in order to measure the gap between cross-lingual performance and monolingual performance. ENLEXNET is trained on the same English samples used for training the BILEXNET model, and evaluated on the En-En examples used to generate cross-lingual examples in Section 5.[9]

**Baselines** Our experiments aim to assess how the direct cross-lingual modeling of semantic relations in BILEXNET impacts predictions, and to isolate the impact of key training components: knowledge distillation and translation selection via attention. We compare against the following baselines:

RANDOM BASELINE: Randomly assign one of the five semantic relations to each word pair.

TRANSLATION BASELINE: This baseline combines dictionary translation and the English-only system ENLEXNET to gauge the relative difficulty of predicting semantic relations across languages rather than in English only. Each pair $(x_e, y_f)$ in the test set is translated into English using the bilingual dictionary $D$. Since $y_f$ can have multiple

[3] https://www.mdbg.net/chinese/dictionary?page=cc-cedict

[4] https://fasttext.cc/docs/en/aligned-vectors.html

[5] https://dumps.wikimedia.org/enwiki/

[6] http://statmt.org/wmt18/translation-task.html

[7] https://github.com/anoopkunchukuttan/indic_nlp_library

[8] https://github.com/fxsjy/jieba

[9] We re-implement the LEXNET model and verify its accuracy by replicating results on the CogALex dataset.

translations, we pair $x_e$ with each of these translations, and use ENLEXNET to predict the relation for each of these pairs. The relation for $(x_e, y_f)$ is then chosen as the most general relation among those predicted for the translated pairs according to the order in which they appear in Table 2.[10]

BILEXNET (NO DISTILLATION) A simple strategy for cross-lingual transfer consists of seeding a vanilla LEXNET model with bilingual embeddings in the source and target languages before training. This strategy has been successfully used for other NLP tasks (Klementiev et al., 2012; Guo et al., 2015, *inter alia*). By keeping the embeddings fixed, we can use source language data to train the monolingual LEXNET model using features based on source embeddings and source language paths as usual. At inference time, the model uses both the source and target embeddings as input, and the cross-lingual paths defined above.

SPECIALIZED TENSOR MODEL (STM) How does a model that has primarily been used for comparing words in the same language perform on cross-lingual comparisons? Our final baseline aims to answer this question. Proposed by Glavaš and Vulić (2018), STM is a neural architecture for identifying semantic relations that achieves state-of-the-art performance on two English dataset. STM is based on the hypothesis that specialized word embeddings are necessary to accurately disambiguate between semantic relations.

More precisely, STM assumes that different specializations of generic word embeddings are needed to recognize different relations and that interactions between the specialized vectors can be used to identify the semantic relations. These different specializations are implemented using $K$ feed-forward neural networks. Given a word pair, STM takes in as input a pair of generic word embeddings for the word pair which are then specialized by the $K$ transformations. Each pair of corresponding specialized embeddings is used to calculate a score based on a non-linear transformation of their bilinear product. Finally, the $K$ scores obtained from $K$ pairs of specialized embeddings are used as features to train a multi-class classifier.

Besides English, STM has also been used for cross-lingual *transfer*, where a model trained on one language (say English) is used to test on word-pairs in another language (say German). Here, we use STM in a new setting to predict the semantic relation between two words in different languages.

We use the official implementation of STM[11] with the same bilingual embeddings used by BILEXNET. We tune three hyperparameters on the validation set (a) The size of the specialized tensors {100, 200, 300, 500} (b) The number of specialization functions {3, 5, 7}, and (c) The learning rate {0.0001, 0.0003}. Default values are used for all other hyper-parameters.

## 7 Results

Tables 3 summarizes results obtained on the MULTILEXREL test sets. BILEXNET achieves F1 scores that are roughly double of those obtained by the random baseline for 5-way classification.

**Impact of cross-lingual modeling** We assess the impact of direct cross-lingual modeling in BILEXNET by comparing against the TRANSLATION BASELINE. Using a translation dictionary to naïvely convert cross-lingual relation prediction to an English-only task, the translation baseline F1 scores are 8 to 13 points higher than RANDOM for both language pairs. This difference can be attributed to easy examples where English semantic relations are preserved by simple dictionary translation. BILEXNET further improves F1 by 8 to 15 points over the TRANSLATION BASELINE, primarily by improving recall.

**Supervised English system** Without cross-lingual training samples, we cannot compare weakly supervised and fully supervised training for BILEXNET in a controlled fashion. However, the supervised monolingual ENLEXNET model (Section 6) evaluated on the En-En test set offers a reference point: remarkably the F1 scores of BILEXNET are only 1 to 3 points lower than those obtained by the supervised English model ($\sim$44 on the En-En test set).

**Impact of knowledge distillation** We compare the full BILEXNET model to the naïve baseline (BILEXNET (NO DISTILLATION)) that only relies on embeddings for cross-lingual transfer and does not perform cross-lingual distillation. This approach performs on par with or a little better than the translation baseline, but $\sim$9 points worse than the full BILEXNET model, losing on both

[10] We also experimented with a voting based approach for combination, but it generally performed worse.

[11] https://github.com/codogogo/stm

| Model | En-Hi | | | En-Zh | | |
|---|---|---|---|---|---|---|
| | P | R | F | P | R | F |
| RANDOM BASELINE | 22.9 ± 1.3 | 20.9 ± 2.5 | 21.4 ± 1.4 | 22.9 ± 1.3 | 20.9 ± 2.5 | 21.4 ± 1.4 |
| TRANSLATION BASELINE | 30.1 ± 1.7 | 26.3 ± 1.3 | 28.3 ± 1.5 | 50.1 ± 2.2 | 32.8 ± 0.8 | 33.0 ± 1.0 |
| STM (Glavaš and Vulić, 2018) | 32.0 | 33.0 | 29.0 | 20.0 | 15.0 | 16.0 |
| BILEXNET (No distillation) | 34.9 ± 1.2 | 34.3 ± 0.6 | 32.2 ± 0.8 | 41.7 ± 6.4 | 33.5 ± 6.2 | 32.6 ± 7.2 |
| BILEXNET (No attention) | 47.2 ± 1.0 | 42.9 ± 1.8 | 41.9 ± 2.3 | 45.0 ± 1.5 | 40.8 ± 1.8 | 39.8 ± 0.8 |
| BILEXNET (Full) | 47.7 ± 1.2 | 44.2 ± 0.8 | **43.3 ± 1.0** | 48.3 ± 1.6 | 41.6 ± 1.1 | **41.1 ± 0.9** |

Table 3: Precision (P), Recall (R) and F1-score (F) for BILEXNET and contrastive baselines on the two MULTILEXREL test sets. All configurations (except STM) are trained with five random seeds. We report the mean score and standard deviation. The full BILEXNET model performs best and is consistently better with attention.

precision and recall. This result confirms the benefit of aligning training and test conditions for our model with knowledge distillation and not relying solely on embeddings. These results are consistent with prior findings on distributional representations:

- Distributional representations have difficulties in discriminating between multiple semantic relations (Chersoni et al., 2016). As such, relying solely on word embeddings for cross-lingual transfer can cause loss of knowledge during transfer.
- Syntactic divergences cause differences in paths in the source and target languages. This can cause a distribution shift between the features seen by the classifier during training and test time, thereby affecting performance. Again, word embeddings are not sufficient to bridge the gap between the distributions of the two languages (Chen et al., 2018b).

**Impact of attention** We test the impact of the attention model in BILEXNET by removing it, and instead translating training samples for distillation using the single most frequent translation. Removing attention yields small but consistent degradations, suggesting that attending to multiple translations is beneficial, but leaves room for improvement. We analyze the behavior of the attention model in the next section.

**Specialization** Finally, we observe F1 scores of STM are significantly worse than those of BILEXNET. In fact, it is the weakest model for En-Zh, and is only 3 points better than the translation baseline for En-Hi. The relatively poor performance of STM highlights that our cross-lingual task, which directly compares words in two languages, is fundamentally different from the cross-lingual transfer task, where models trained in one language are ported to other languages. Thus, models such as STM, which have been designed for transfer, may not be directly suitable for our task.

## 8 Analysis

This section further breaks down the results, and highlight some successes and failures of BILEXNET to guide future work on cross-lingual semantic relation classification.

**Performance Per Class** We break down the performance of the BILEXNET model per target relation (Table 4). The `Equivalence` and `Exclusion` classes are the hardest to predict correctly, which is consistent with our monolingual results and those from prior work (Shwartz and Dagan, 2016a): distributional models have trouble distinguishing synonyms from antonyms (Yih et al., 2012) and synonyms rarely occur in the same sentence, and hence path-based methods are less useful for this class. However, in BILEXNET, words of the `Equivalence` class can occur in a parallel sentence pair where they are aligned to each other. Thus, there is a direct signal for examples of this class which helps discriminate between `Equivalence` and `Exclusion`.

The largest fraction of errors are caused by the model predicting `Other` instead of a specific relation. This suggests that special treatment of this class might improve performance, perhaps by using a multi-step process which filters out pairs not related under the relations that we are targeting, and then performs 4-way classification for the remaining examples. This is similar to the the Co-

| Class | En-Hi | En-Zh | En-En |
|---|---|---|---|
| Equivalence | 33 | 30 | 31 |
| Exclusion | 33 | 28 | 23 |
| Forward Entail | 47 | 48 | 48 |
| Backward Entail | 45 | 58 | 48 |
| Other | 51 | 29 | 53 |

Table 4: Per-class F1 scores for median En-Hi and En-Zh BILEXNET model and the ENLEXNET model.

gALex shared task (Santus et al., 2016), where the first part of the task is to eliminate *completely unrelated* pairs, before predicting relations on the remaining pairs. However, filtering out *unrelated* pairs is an easier task than filtering pairs in the Other category.

**Missing cross-lingual paths** Cross-lingual paths might not exist for all word pairs, particularly for language pairs with limited parallel data such as En-Hi. BILEXNET would then only rely on word embeddings as features to predict semantic relations. We assess the impact of missing paths by comparing the classification performance on pairs which have cross-lingual paths (70% of the test), against pairs which do not have paths in the En-Hi setting. The former subset has a higher F1 score (44.6) than the latter (40.2), mainly due to differences in recall. This difference in performance also confirms that the cross-lingual paths complement word embeddings, in the same way that monolingual paths do.

**Attention Analysis** We complement ablation experiments in Table 3 by examining a random sample of 25 monolingual training pairs $(x_e, y_e)$ where $y_e$ has multiple translations in the bilingual dictionary. We manually check for how many pairs the model places the highest attention weight on a translation that preserves the relation label of the monolingual pair. This happens in 64% of the cases (16 out of 25). The attention model is often able to modulate the choice of the right translation of $y_e$ based on the context provided by $x_e$ and the gold label. For example, given the monolingual example (*drop*, *fall*, Forward Entail) the model places the highest weight on the Hindi word गिरा, which captures the "moving downward" sense. On the other hand, for the example (*autumn*, *fall*, Equivalence), the model correctly identifies पतझड as the right translation.

There still remains a lot of overhead for improving the attention component. Some failure cases in the 25 examples occur for pairs where the set of translations of $y_e$ contains an incorrect translation which is totally unrelated to $x_e$ or $y_e$. For example, given (*country*, *uganda*), the model chooses the word कैंडल (transliteration for *candle*) and not युगांडा (transliteration for *uganda*). Of course, this is an extreme example, but such errors are also more likely to occur when the noisy translation is in the same domain as $x_e$ and $y_e$. Fixing such errors can help improve the training process.

## 9 Conclusion

This work contributes data and models to the task of classifying semantic relations between words in different languages with only monolingual English supervision. We introduced MULTILEXREL, a dataset of about 1000 English-Hindi and 900 English-Chinese word pairs annotated with the natural logic lexical entailment classes of MacCartney and Manning (2007), and BILEXNET, a cross-lingual relation classification model.

We also proposed a knowledge distillation algorithm for BILEXNET, which only needs annotated monolingual examples and a bilingual dictionary. Unlike previous uses of knowledge distillation for cross-lingual transfer, our approach does not assume that labels are translation invariant, and relies on an attention mechanism to select translations that best explain a given label. Experiments show that this method largely outperforms baselines that use bilingual embeddings or dictionaries more naïvely for cross-lingual transfer, and that it approaches the performance of fully supervised systems on an English-only version of the task.

## Acknowledgements

The authors would like to thank members of the CLIP Lab and the anonymous reviewers for feedback on previous versions of this paper. This material is based upon work supported by the National Science Foundation under Award No. 1750695.

## References

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018. A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 789–798.

Mona Baker. 2011. *In other words: A coursebook on translation*. Routledge.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Francis Bond and Ryan Foster. 2013. Linking and extending an open multilingual wordnet. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1352–1362. Association for Computational Linguistics.

Wenhu Chen, Jianshu Chen, Yu Su, Xin Wang, Dong Yu, Xifeng Yan, and William Yang Wang. 2018a. XL-NBT: A Cross-lingual Neural Belief Tracking Framework. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 414–424.

Xilun Chen, Yu Sun, Ben Athiwaratkun, Claire Cardie, and Kilian Weinberger. 2018b. Adversarial Deep Averaging Networks for Cross-Lingual Sentiment Classification. *Transactions of the Association for Computational Linguistics*, 6:557–570.

Emmanuele Chersoni, Giulia Rambelli, and Enrico Santus. 2016. CogALex-V Shared Task: ROOT18. In *Proceedings of the 5th Workshop on Cognitive Aspects of the Lexicon (CogALex - V)*, pages 98–103, Osaka, Japan. The COLING 2016 Organizing Committee.

Andrew Chesterman. 2016. *Memes of translation: The spread of ideas in translation theory*, volume 123. John Benjamins Publishing Company.

David Alan Cruse. 1986. *Lexical semantics*. Cambridge university press.

Bonnie Dorr. 1994. Machine Translation Divergences: A Formal Description and Proposed Solution. *Computational Linguistics*, 20(4):597–633.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

Pascale Fung and Lo Yuen Yee. 1998. An IR Approach for Translating New Words from Nonparallel, Comparable Texts. In *Proceedings of the 17th International Conference on Computational Linguistics - Volume 1*, COLING '98, pages 414–420, Stroudsburg, PA, USA. Association for Computational Linguistics.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The Paraphrase Database. In *HLT-NAACL*, pages 758–764.

Goran Glavaš and Simone Paolo Ponzetto. 2017. Dual tensor model for detecting asymmetric lexico-semantic relations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1757–1767. Association for Computational Linguistics.

Goran Glavaš and Ivan Vulić. 2018. Discriminating between Lexico-Semantic Relations with the Specialization Tensor Model. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 181–187, New Orleans, Louisiana. Association for Computational Linguistics.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual Dependency Parsing Based on Distributed Representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1234–1244. Association for Computational Linguistics.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *COLING 1992 Volume 2: The 15th International Conference on Computational Linguistics*.

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. 2014. Distilling the Knowledge in a Neural Network. In *NIPS Deep Learning and Representation Learning Workshop*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. Learning Whom to Trust with MACE. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1120–1130. Association for Computational Linguistics.

Ann Irvine and Chris Callison-Burch. 2017. A Comprehensive Analysis of Bilingual Lexicon Induction. *Computational Linguistics*, 43(2):273–310.

Diederick P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing Crosslingual Distributed Representations of Words. In *Proceedings of COLING 2012*, pages 1459–1474. The COLING 2012 Organizing Committee.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster*

*Sessions*, pages 177–180. Association for Computational Linguistics.

S. Kullback and R. A. Leibler. 1951. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86.

Anoop Kunchukuttan, Pratik Mehta, and Pushpak Bhattacharyya. 2018. The IIT Bombay English-Hindi Parallel Corpus. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Guillaume Lample, Alexis Conneau, Marc'Aurelio Ranzato, Ludovic Denoyer, and Herv Jgou. 2018. Word translation without parallel data. In *International Conference on Learning Representations*.

Omer Levy, Steffen Remus, Chris Biemann, and Ido Dagan. 2015. Do supervised distributional methods really learn lexical inference relations? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 970–976. Association for Computational Linguistics.

Bill MacCartney and Christopher D. Manning. 2007. Natural logic for textual inference. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing - RTE '07*, page 193, Prague, Czech Republic. Association for Computational Linguistics.

Bill MacCartney and Christopher D. Manning. 2009. An extended model of natural logic. In *Proceedings of the Eighth International Conference on Computational Semantics - IWCS-8 '09*, page 140, Tilburg, The Netherlands. Association for Computational Linguistics.

Stephen Mayhew, Chen-Tse Tsai, and Dan Roth. 2017. Cheap Translation for Cross-Lingual Named Entity Recognition. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2536–2545.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu, and Jungmee Lee. 2013. Universal Dependency Annotation for Multilingual Parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 92–97.

Nikola Mrkšić, Ivan Vulić, Diarmuid Ó Séaghdha, Ira Leviant, Roi Reichart, Milica Gašić, Anna Korhonen, and Steve Young. 2017. Semantic specialization of distributional word vector spaces using monolingual and cross-lingual constraints. *Transactions of the Association of Computational Linguistics*, 5:309–324.

Ndapandula Nakashole and Raphael Flauger. 2017. Knowledge Distillation for Bilingual Dictionary Induction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2497–2506.

Vivi Nastase, Preslav Nakov, Diarmuid Ó Séaghdha, and Stan Szpakowicz. 2013. Semantic Relations Between Nominals. *Synthesis Lectures on Human Language Technologies*, 6(1):1–119.

Masataka Ono, Makoto Miwa, and Yutaka Sasaki. 2015. Word Embedding-based Antonym Detection using Thesauri and Distributional Information. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 984–989. Association for Computational Linguistics.

Patrick Pantel and Marco Pennacchiotti. 2006. Espresso: Leveraging Generic Patterns for Automatically Harvesting Semantic Relations. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 113–120.

Ellie Pavlick, Johan Bos, Malvina Nissim, Charley Beller, Benjamin Van Durme, and Chris Callison-Burch. 2015. Adding Semantics to Data-Driven Paraphrasing. pages 1512–1522. Association for Computational Linguistics.

Yves Peirsman and Sebastian Padó. 2011. Semantic relations in bilingual lexicons. *ACM Transactions on Speech and Language Processing*, 8(2):1–21.

Marco Pennacchiotti and Patrick Pantel. 2006. A Bootstrapping Algorithm for Automatically Harvesting Semantic Relations. In *Proceedings of the Fifth International Workshop on Inference in Computational Semantics (ICoS-5)*.

Reinhard Rapp. 1995. Identifying word translations in non-parallel texts. In *Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics -*, page 320, Cambridge, Massachusetts. Association for Computational Linguistics.

Mohammad Sadegh Rasooli and Joel R. Tetreault. 2015. Yara Parser: A Fast and Accurate Dependency Parser. *CoRR*, abs/1503.06733.

Michael Roth and Shyam Upadhyay. 2019. Combining Discourse Markers and Cross-lingual Embeddings for Synonym–Antonym Classification. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3899–3905.

Enrico Santus, Anna Gladkova, Stefan Evert, and Alessandro Lenci. 2016. The CogALex-V Shared Task on the Corpus-Based Identification of Semantic Relations. In *Proceedings of the 5th Workshop*

*on Cognitive Aspects of the Lexicon (CogALex - V)*, pages 69–79. The COLING 2016 Organizing Committee.

Vered Shwartz and Ido Dagan. 2016a. CogALex-V Shared Task: LexNET - Integrated Path-based and Distributional Method for the Identification of Semantic Relations. In *Proceedings of the 5th Workshop on Cognitive Aspects of the Lexicon (CogALex - V)*, pages 80–85. The COLING 2016 Organizing Committee.

Vered Shwartz and Ido Dagan. 2016b. Path-based vs. Distributional Information in Recognizing Lexical Semantic Relations. In *Proceedings of the 5th Workshop on Cognitive Aspects of the Lexicon (CogALex - V)*, pages 24–29. The COLING 2016 Organizing Committee.

Vered Shwartz, Yoav Goldberg, and Ido Dagan. 2016. Improving hypernymy detection with an integrated path-based and distributional method. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2389–2398. Association for Computational Linguistics.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2004. Learning syntactic patterns for automatic hypernym discovery. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1297–1304. MIT Press.

Anders Søgaard, Sebastian Ruder, and Ivan Vulić. 2018. On the Limitations of Unsupervised Bilingual Dictionary Induction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 778–788. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Yulia Tsvetkov, Leonid Boytsov, Anatole Gershman, Eric Nyberg, and Chris Dyer. 2014. Metaphor Detection with Cross-Lingual Model Transfer. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 248–258, Baltimore, Maryland. Association for Computational Linguistics.

Peter Turney. 2008. A Uniform Approach to Analogies, Synonyms, Antonyms, and Associations. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING 2008)*, pages 905–912. COLING 2008 Organizing Committee.

Shyam Upadhyay, Yogarshi Vyas, Marine Carpuat, and Dan Roth. 2018. Robust Cross-Lingual Hypernymy Detection Using Dependency Context. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 607–618.

Lawrence Venuti. 2012. *The translation studies reader*. Routledge.

Ivan Vulić and Nikola Mrkšić. 2018. Specialising Word Vectors for Lexical Entailment. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1134–1145, New Orleans, Louisiana. Association for Computational Linguistics.

Yogarshi Vyas and Marine Carpuat. 2016. Sparse Bilingual Word Representations for Cross-lingual Lexical Entailment. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1187–1197.

Ruochen Xu and Yiming Yang. 2017. Cross-lingual Distillation for Text Classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1415–1425. Association for Computational Linguistics.

Wen-tau Yih, Geoffrey Zweig, and John Platt. 2012. Polarity Inducing Latent Semantic Analysis. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1212–1222, Jeju Island, Korea. Association for Computational Linguistics.

# Improved Word Sense Disambiguation Using Pre-Trained Contextualized Word Representations

**Christian Hadiwinoto, Hwee Tou Ng,** and **Wee Chung Gan**
Department of Computer Science, National University of Singapore
christian.hadiwinoto@u.nus.edu, nght@comp.nus.edu.sg,
gan_weechung@u.nus.edu

## Abstract

Contextualized word representations are able to give different representations for the same word in different contexts, and they have been shown to be effective in downstream natural language processing tasks, such as question answering, named entity recognition, and sentiment analysis. However, evaluation on word sense disambiguation (WSD) in prior work shows that using contextualized word representations does not outperform the state-of-the-art approach that makes use of non-contextualized word embeddings. In this paper, we explore different strategies of integrating pre-trained contextualized word representations and our best strategy achieves accuracies exceeding the best prior published accuracies by significant margins on multiple benchmark WSD datasets.

## 1 Introduction

Word sense disambiguation (WSD) automatically assigns a pre-defined sense to a word in a text. Different senses of a word reflect different meanings a word has in different contexts. Identifying the correct word sense given a context is crucial in natural language processing (NLP). Unfortunately, while it is easy for a human to infer the correct sense of a word given a context, it is a challenge for NLP systems. As such, WSD is an important task and it has been shown that WSD helps downstream NLP tasks, such as machine translation (Chan et al., 2007a) and information retrieval (Zhong and Ng, 2012).

A WSD system assigns a sense to a word by taking into account its context, comprising the other words in the sentence. This can be done through discrete word features, which typically involve surrounding words and collocations trained using a classifier (Lee et al., 2004; Ando, 2006; Chan et al., 2007b; Zhong and Ng, 2010). The classifier can also make use of continuous word representations of the surrounding words (Taghipour and Ng, 2015; Iacobacci et al., 2016). Neural WSD systems (Kågebäck and Salomonsson, 2016; Raganato et al., 2017b) feed the continuous word representations into a neural network that captures the whole sentence and the word representation in the sentence. However, in both approaches, the word representations are independent of the context.

Recently, pre-trained contextualized word representations (Melamud et al., 2016; McCann et al., 2017; Peters et al., 2018; Devlin et al., 2019) have been shown to improve downstream NLP tasks. Pre-trained contextualized word representations are obtained through neural sentence encoders trained on a huge amount of raw texts. When the resulting sentence encoder is fine-tuned on the downstream task, such as question answering, named entity recognition, and sentiment analysis, with much smaller annotated training data, it has been shown that the trained model, with the pre-trained sentence encoder component, achieves new state-of-the-art results on those tasks.

While demonstrating superior performance in downstream NLP tasks, pre-trained contextualized word representations are still reported to give lower accuracy compared to approaches that use non-contextualized word representations (Melamud et al., 2016; Peters et al., 2018) when evaluated on WSD. This seems counter-intuitive, as a neural sentence encoder better captures the surrounding context that serves as an important cue to disambiguate words. In this paper, we explore different strategies of integrating pre-trained contextualized word representations for WSD. Our best strategy outperforms prior methods of incorporating pre-trained contextualized word representations and achieves new state-of-the-art accuracy on multiple benchmark WSD datasets.

The following sections are organized as follows.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5297–5306,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

Section 2 presents related work. Section 3 describes our pre-trained contextualized word representation. Section 4 proposes different strategies to incorporate the contextualized word representation for WSD. Section 5 describes our experimental setup. Section 6 presents the experimental results. Section 7 discusses the findings from the experiments. Finally, Section 8 presents the conclusion.

## 2 Related Work

Continuous word representations in real-valued vectors, or commonly known as word embeddings, have been shown to help improve NLP performance. Initially, exploiting continuous representations was achieved by adding real-valued vectors as classification features (Turian et al., 2010). Taghipour and Ng (2015) fine-tuned non-contextualized word embeddings by a feed-forward neural network such that those word embeddings were more suited for WSD. The fine-tuned embeddings were incorporated into an SVM classifier. Iacobacci et al. (2016) explored different strategies of incorporating word embeddings and found that their best strategy involved exponential decay that decreased the contribution of surrounding word features as their distances to the target word increased.

The neural sequence tagging approach has also been explored for WSD. Kågebäck and Salomonsson (2016) proposed bidirectional long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) for WSD. They concatenated the hidden states of the forward and backward LSTMs and fed the concatenation into an affine transformation followed by softmax normalization, similar to the approach to incorporate a bidirectional LSTM adopted in sequence labeling tasks such as part-of-speech tagging and named entity recognition (Ma and Hovy, 2016). Raganato et al. (2017b) proposed a self-attention layer on top of the concatenated bidirectional LSTM hidden states for WSD and introduced multi-task learning with part-of-speech tagging and semantic labeling as auxiliary tasks. However, on average across the test sets, their approach did not outperform SVM with word embedding features. Subsequently, Luo et al. (2018) proposed the incorporation of glosses from WordNet in a bidirectional LSTM for WSD, and reported better results than both SVM and prior bidirectional LSTM models.

A neural language model (LM) is aimed at predicting a word given its surrounding context. As such, the resulting hidden representation vector captures the context of a word in a sentence. Melamud et al. (2016) designed *context2vec*, which is a one-layer bidirectional LSTM trained to maximize the similarity between the hidden state representation of the LSTM and the target word embedding. Peters et al. (2018) designed ELMo, which is a two-layer bidirectional LSTM language model trained to predict the next word in the forward LSTM and the previous word in the backward LSTM. In both models, WSD was evaluated by nearest neighbor matching between the test and training instance representations. However, despite training on a huge amount of raw texts, the resulting accuracies were still lower than those achieved by WSD approaches with pre-trained non-contextualized word representations.

End-to-end neural machine translation (NMT) (Sutskever et al., 2014; Bahdanau et al., 2015) learns to generate an output sequence given an input sequence, using an encoder-decoder model. The encoder captures the contextualized representation of the words in the input sentence for the decoder to generate the output sentence. Following this intuition, McCann et al. (2017) trained an encoder-decoder model on parallel texts and obtained pre-trained contextualized word representations from the encoder.

## 3 Pre-Trained Contextualized Word Representation

The contextualized word representation that we use is BERT (Devlin et al., 2019), which is a bidirectional transformer encoder model (Vaswani et al., 2017) pre-trained on billions of words of texts. There are two tasks on which the model is trained, i.e., masked word and next sentence prediction. In both tasks, prediction accuracy is determined by the ability of the model to understand the context.

A transformer encoder computes the representation of each word through an attention mechanism with respect to the surrounding words. Given a sentence $x_1^n$ of length $n$, the transformer computes the representation of each word $x_i$ through a multi-head attention mechanism, where the query vector is from $x_i$ and the key-value vector pairs are from the surrounding words $x_{i'}$ ($1 \leq i' \leq n$). The word representation produced by the transformer

captures the contextual information of a word.

The attention mechanism can be viewed as mapping a query vector $\mathbf{q}$ and a set of key-value vector pairs $(\mathbf{k}, \mathbf{v})$ to an output vector. The attention function $A(\cdot)$ computes the output vector which is the weighted sum of the value vectors and is defined as:

$$A(\mathbf{q}, \mathbf{K}, \mathbf{V}, \rho) = \sum_{(\mathbf{k},\mathbf{v})\in(\mathbf{K},\mathbf{V})} \alpha(\mathbf{q}, \mathbf{k}, \rho)\mathbf{v} \quad (1)$$

$$\alpha(\mathbf{q}, \mathbf{k}, \rho) = \frac{\exp(\rho \mathbf{k}^\top \mathbf{q})}{\sum_{\mathbf{k}'\in\mathbf{K}} \exp(\rho \mathbf{k}'^\top \mathbf{q})} \quad (2)$$

where $\mathbf{K}$ and $\mathbf{V}$ are matrices, containing the key vectors and the value vectors of the words in the sentence respectively, and $\alpha(\mathbf{q}, \mathbf{k}, \rho)$ is a scalar attention weight between $\mathbf{q}$ and $\mathbf{k}$, re-scaled by a scalar $\rho$.

Two building blocks for the transformer encoder are the multi-head attention mechanism and the position-wise feed-forward neural network (FFNN). The multi-head attention mechanism with $H$ heads leverages the attention function in Equation 1 as follows:

$$\text{MH}(\mathbf{q}, \mathbf{K}, \mathbf{V}, \rho) = \mathbf{W}_{\text{MH}} \bigoplus_{\eta=1}^{H} \text{head}_\eta(\mathbf{q}, \mathbf{K}, \mathbf{V}, \rho) \quad (3)$$

$$\text{head}_\eta(\mathbf{q}, \mathbf{K}, \mathbf{V}, \rho) = A(\mathbf{W}^{\mathbf{Q}}_\eta \mathbf{q}, \mathbf{W}^{\mathbf{K}}_\eta \mathbf{K}, \mathbf{W}^{\mathbf{V}}_\eta \mathbf{V}, \rho) \quad (4)$$

where $\oplus$ denotes concatenation of vectors, $\mathbf{W}_{\text{MH}} \in \mathbb{R}^{d_{\text{model}} \times H d_{\mathbf{v}}}$, $\mathbf{W}^{\mathbf{Q}}_\eta, \mathbf{W}^{\mathbf{K}}_\eta \in \mathbb{R}^{d_{\mathbf{k}} \times d_{\text{model}}}$, and $\mathbf{W}^{\mathbf{V}}_\eta \in \mathbb{R}^{d_{\mathbf{v}} \times d_{\text{model}}}$. The input vector $\mathbf{q} \in \mathbb{R}^{d_{\text{model}}}$ is the hidden vector for the ambiguous word, while input matrices $\mathbf{K}, \mathbf{V} \in \mathbb{R}^{d_{\text{model}} \times n}$ are the concatenation of the hidden vectors of all words in the sentence. For each attention head, the dimension of both the query and key vectors is $d_{\mathbf{k}}$ while the dimension of the value vector is $d_{\mathbf{v}}$. The encoder model dimension is $d_{\text{model}}$.

The position-wise FFNN performs a non-linear transformation on the attention output corresponding to each input word position as follows:

$$\text{FF}(\mathbf{u}) = \mathbf{W}_{\text{FF2}}(\max(0, \mathbf{W}_{\text{FF1}}\mathbf{u} + \mathbf{b}_{\text{FF1}})) + \mathbf{b}_{\text{FF2}} \quad (5)$$

in which the input vector $\mathbf{u} \in \mathbb{R}^{d_{\text{model}}}$ is transformed to the output vector with dimension $d_{\text{model}}$ via a series of linear projections with the ReLU activation function.

For the hidden layer $l$ ($1 \leq l \leq L$), the self-attention sub-layer output $\mathbf{f}^l_i$ is computed as follows:

$$\mathbf{f}^l_i = \text{LayerNorm}(\chi^l_{\mathbf{h},i} + \mathbf{h}^{l-1}_i)$$

$$\chi^l_{\mathbf{h},i} = \text{MH}^l_{\mathbf{h}}(\mathbf{h}^{l-1}_i, \mathbf{h}^{l-1}_{1:n}, \mathbf{h}^{l-1}_{1:n}, \frac{1}{\sqrt{d_{\mathbf{v}}}})$$

where LayerNorm refers to layer normalization (Ba et al., 2016) and the superscript $l$ and subscript $\mathbf{h}$ indicate that each encoder layer $l$ has an independent set of multi-head attention weight parameters (see Equations 3 and 4). The input for the first layer is $\mathbf{h}^0_i = \mathbf{E}(x_i)$, which is the non-contextualized word embedding of $x_i$.

The second sub-layer consists of the position-wise fully connected FFNN, computed as:

$$\mathbf{h}^l_i = \text{LayerNorm}(\text{FF}^l_{\mathbf{h}}(\mathbf{f}^l_i) + \mathbf{f}^l_i)$$

where, similar to self-attention, an independent set of weight parameters in Equation 5 is defined in each layer.

## 4 Incorporating Pre-Trained Contextualized Word Representation

As BERT is trained on the masked word prediction task, which is to predict a word given the surrounding (left and right) context, the pre-trained model already captures the context. In this section, we describe different techniques of leveraging BERT for WSD, broadly categorized into nearest neighbor matching and linear projection of hidden layers.

### 4.1 Nearest Neighbor Matching

A straightforward way to disambiguate word sense is through 1-nearest neighbor matching. We compute the contextualized representation of each word in the training data and the test data through BERT. Given a hidden representation $\mathbf{h}^L_i$ at the $L$-th layer for word $x_i$ in the test data, nearest neighbor matching finds a vector $\mathbf{h}^*$ in the $L$-th layer from the training data such that

$$\mathbf{h}^* = \arg\max_{\mathbf{h}'} \cos(\mathbf{h}^L_i, \mathbf{h}') \quad (6)$$

where the sense assigned to $x_i$ is the sense of the word whose contextualized representation is $\mathbf{h}^*$. This WSD technique is adopted in earlier work on contextualized word representations (Melamud et al., 2016; Peters et al., 2018).

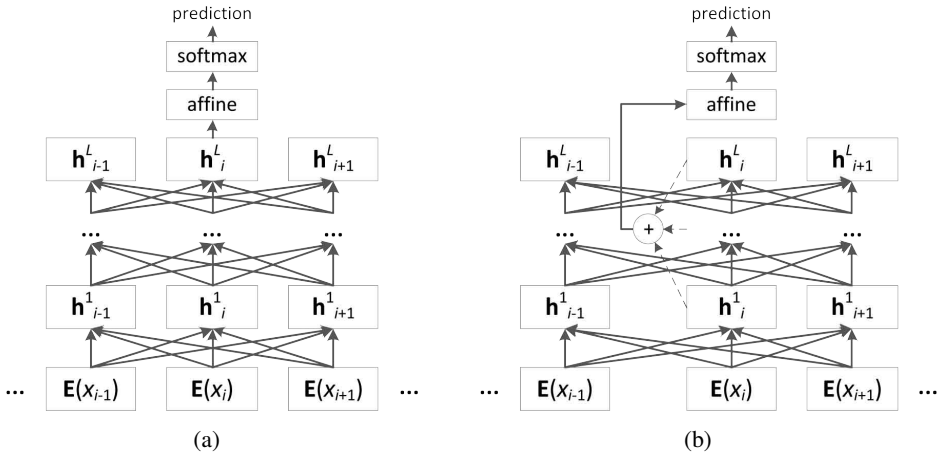


(a) (b)

Figure 1: Illustration of WSD models by linear projection of (a) the last layer and (b) the weighted sum of all layers.

### 4.2 Linear Projection of Hidden Layers

Apart from nearest neighbor matching, we can perform a linear projection of the hidden vector $\mathbf{h}_i$ by an affine transformation into an output sense vector, with its dimension equal to the number of senses for word $x_i$. The output of this affine transformation is normalized by softmax such that all its values sum to 1. Therefore, the predicted sense $\mathbf{s}_i$ of word $x_i$ is formulated as

$$\mathbf{s}_i = \text{softmax}(\mathbf{W}^{\text{lexelt}(x_i)}\mathbf{h}_i + \mathbf{b}^{\text{lexelt}(x_i)}) \quad (7)$$

where $\mathbf{s}_i$ is a vector of predicted sense distribution for word $x_i$, while $\mathbf{W}^{\text{lexelt}(x_i)}$ and $\mathbf{b}^{\text{lexelt}(x_i)}$ are respectively the projection matrix and bias vector specific to the lexical element (lexelt) of word $x_i$, which consists of its lemma and optionally its part-of-speech tag. We choose the sense corresponding to the element of $\mathbf{s}_i$ with the maximum value.

Training the linear projection model is done by the back-propagation algorithm, which updates the model parameters to minimize a cost function. Our cost function is the negative log-likelihood of the softmax output value that corresponds to the tagged sense in the training data. In addition, we propose two *novel* ways of incorporating BERT's hidden representation vectors to compute the sense output vector, which are described in the following sub-subsections.

#### 4.2.1 Last Layer Projection

Similar to the nearest neighbor matching model, we can feed the hidden vector of BERT in the last layer, $\mathbf{h}_i^L$, into an affine transformation followed by softmax. That is, $\mathbf{h}_i$ in Equation 7 is instantiated by $\mathbf{h}_i^L$. The last layer projection model is illustrated in Figure 1(a).

#### 4.2.2 Weighted Sum of Hidden Layers

BERT consists of multiple layers stacked one after another. Each layer carries a different representation of word context. Taking into account different hidden layers may help the WSD system to learn from different context information encoded in different layers of BERT.

To take into account all layers, we compute the weighted sum of all hidden layers, $\mathbf{h}_i^l$, where $1 \leq l \leq L$, corresponding to a word position $i$, through attention mechanism. That is, $\mathbf{h}_i$ in Equation 7 is replaced by the following equation:

$$\mathbf{h}_i = A(\mathbf{m}, \mathbf{W}^{\mathbf{s}}\mathbf{h}_i^{1:L}, \mathbf{h}_i^{1:L}, 1) \quad (8)$$

where $\mathbf{m} \in \mathbb{R}^{d_{\text{model}}}$ is a projection vector to obtain scalar values with the key vectors. The model with weighted sum of all hidden layers is illustrated in Figure 1(b).

#### 4.2.3 Gated Linear Unit

While the contextualized representations in the BERT hidden layer vectors are features that determine the word sense, some features are more useful than the others. As such, we propose filtering the vector values by a gating vector whose values range from 0 to 1. This mechanism is known as the gated linear unit (GLU) (Dauphin et al., 2017), which is formulated as

$$\text{GLU}(\mathbf{h}) = (\mathbf{h}+\mathbf{W}^{\mathbf{h}}\mathbf{h}+\mathbf{b}^{\mathbf{h}})\odot\sigma(\mathbf{W}^{\mathbf{g}}\mathbf{h}+\mathbf{b}^{\mathbf{g}}) \quad (9)$$

where $\mathbf{W}^{\mathbf{h}}$ and $\mathbf{W}^{\mathbf{g}}$ are separate projection matrices and $\mathbf{b}^{\mathbf{h}}$ and $\mathbf{b}^{\mathbf{g}}$ are separate bias vectors. The

symbols $\sigma(\cdot)$ and $\odot$ denote the sigmoid function and element-wise vector multiplication operation respectively.

GLU transforms the input vector $\mathbf{h}$ by feeding it to two separate affine transformations. The second transformation is used as the sigmoid gate to filter the input vector, which is summed with the vector after the first affine transformation.

## 5 Experimental Setup

We conduct experiments on various WSD tasks. The description and the statistics for each task are given in Table 1. For English, a lexical element (lexelt) is defined as a combination of lemma and part-of-speech tag, while for Chinese, it is simply the lemma, following the OntoNotes setup.

We exploit English $\text{BERT}_{\text{BASE}}$ for the English tasks and Chinese BERT for the Chinese task. We conduct experiments with different strategies of incorporating BERT as described in Section 4, namely 1-nearest neighbor matching (**1-nn**) and linear projection. In the latter technique, we explore strategies including **simple** last layer projection, layer weighting (**LW**), and gated linear unit (**GLU**).

In the linear projection model, we train the model by the Adam algorithm (Kingma and Ba, 2015) with a learning rate of $10^{-3}$. The model parameters are updated per mini-batch of 16 sentences. As update progresses, we pick the best model parameters from a series of neural network updates based on accuracy on a held-out development set, disjoint from the training set.

The state-of-the-art supervised WSD approach takes into account features from the neighboring sentences, typically one sentence to the left and one to the right apart from the current sentence that contains the ambiguous words. We also exploit this in our model, as BERT supports inputs with multiple sentences separated by a special `[SEP]` symbol.

For English all-words WSD, we train our WSD model on SemCor (Miller et al., 1994), and test it on Senseval-2 (SE2), Senseval-3 (SE3), SemEval 2013 task 12 (SE13), and SemEval 2015 task 13 (SE15). This common benchmark, which has been annotated with WordNet-3.0 senses (Raganato et al., 2017a), has recently been adopted in English all-words WSD. Following (Raganato et al., 2017b), we choose SemEval 2007 Task 17 (SE07) as our development data to pick the best model parameters after a number of neural network updates, for models that require back-propagation training.

| Task | # Instances | # Lexelts |
|---|---|---|
| English all-words | | |
| - SemCor (train) | 226,036 | 22,436 |
| - SemEval-2007 (dev) | 455 | 330 |
| - Senseval-2 (test) | 2,282 | 1,093 |
| - Senseval-3 (test) | 1,850 | 977 |
| - SemEval 2013 (test) | 1,664 | 751 |
| - SemEval 2015 (test) | 1,022 | 512 |
| English lexical sample | | |
| - Senseval-2 (train) | 8,611 | 177 |
| - Senseval-2 (test) | 4,328 | 146 |
| - Senseval-3 (train) | 8,022 | 57 |
| - Senseval-3 (test) | 3,944 | 57 |
| Chinese OntoNotes | | |
| - Train | 66,409 | 431 |
| - Dev | 9,523 | 341 |
| - BC (test) | 1,769 | 160 |
| - BN (test) | 3,227 | 253 |
| - MZ (test) | 1,876 | 223 |
| - NW (test) | 1,483 | 143 |
| - All (test) | 8,355 | 324 |

Table 1: Statistics of the datasets used for the English all-words task, English lexical sample task, and Chinese OntoNotes WSD task in terms of the number of instances and the number of distinct lexelts. For Chinese WSD task, "All" refers to the concatenation of all genres BC, BN, MZ, and NW.

We also evaluate on Senseval-2 and Senseval-3 English lexical sample tasks, which come with pre-defined training and test data. For each word type, we pick 20% of the training instances to be our development set and keep the remaining 80% as the actual training data. Through this development set, we determine the number of epochs to use in training. We then re-train the model with the whole training dataset using the number of epochs identified in the initial training step.

While WSD is predominantly evaluated on English, we are also interested in evaluating our approach on Chinese, to evaluate the effectiveness of our approach in a different language. We use OntoNotes Release 5.0[1], which contains a number of annotations including word senses for Chinese. We follow the data setup of Pradhan et al. (2013) and conduct an evaluation on four genres, i.e., broadcast conversation (BC), broadcast news (BN), magazine (MZ), and newswire (NW), as well as the concatenation of all genres. While the training and development datasets are divided into genres, we train on the concatenation of all genres and test on each individual genre.

[1] LDC2013T19

| System | SE07 | SE2 | SE3 | SE13 | SE15 | Avg |
|---|---|---|---|---|---|---|
| *Reported in previous papers* | | | | | | |
| MFS baseline | 54.5 | 65.6 | 66.0 | 63.8 | 67.1 | 65.6 |
| IMS (Zhong and Ng, 2010) | 61.3 | 70.9 | 69.3 | 65.3 | 69.5 | 68.8 |
| IMS+emb (Iacobacci et al., 2016) | 60.9 | 71.0 | 69.3 | 67.3 | 71.3 | 69.7 |
| SupWSD (Papandrea et al., 2017) | 60.2 | 71.3 | 68.8 | 65.8 | 70.0 | 69.0 |
| SupWSD+emb (Papandrea et al., 2017) | 63.1 | 72.7 | 70.6 | 66.8 | 71.8 | 70.5 |
| BiLSTMatt+LEX (Raganato et al., 2017b) | 63.7 | 72.0 | 69.4 | 66.4 | 72.4 | 70.1 |
| GASext Concat (Luo et al., 2018) | – | 72.2 | 70.5 | 67.2 | 72.6 | 70.6 |
| context2vec (Melamud et al., 2016) | 61.3 | 71.8 | 69.1 | 65.6 | 71.9 | 69.6 |
| ELMo (Peters et al., 2018) | 62.2 | 71.6 | 69.6 | 66.2 | 71.3 | 69.7 |
| *BERT nearest neighbor (ours)* | | | | | | |
| 1nn (1sent) | 64.0 | 73.0 | 69.7 | 67.8 | 73.3 | 71.0 |
| 1nn (1sent+1sur) | 63.3 | 73.8 | 71.6 | 69.2 | 74.4 | 72.3 |
| *BERT linear projection (ours)* | | | | | | |
| Simple (1sent) | 67.0 | 75.0 | 71.6 | 69.7 | 74.4 | 72.7 |
| Simple (1sent+1sur) | **69.3**$^*$ | 75.9$^*$ | 73.4 | 70.4$^*$ | 75.1 | 73.7$^*$ |
| LW (1sent) | 66.7 | 75.0 | 71.6 | 69.9 | 74.2 | 72.7 |
| LW (1sent+1sur) | 69.0$^*$ | **76.4**$^*$ | **74.0**$^*$ | 70.1$^*$ | 75.0 | 73.9$^*$ |
| GLU (1sent) | 64.9 | 74.1 | 71.6 | 69.8 | 74.3 | 72.5 |
| GLU (1sent+1sur) | 68.1$^*$ | 75.5$^*$ | 73.6$^*$ | **71.1**$^*$ | **76.2**$^*$ | **74.1**$^*$ |
| GLU+LW (1sent) | 65.7 | 74.0 | 70.9 | 68.8 | 73.6 | 71.8 |
| GLU+LW (1sent+1sur) | 68.5$^*$ | 75.5$^*$ | 73.4$^*$ | 71.0$^*$ | **76.2**$^*$ | 74.0$^*$ |

Table 2: English all-words task results in F1 measure (%), averaged over three runs. SemEval 2007 Task 17 (SE07) test set is used as the development set. We show the results of nearest neighbor matching (**1nn**) and linear projection, by **simple** last layer linear projection, layer weighting (**LW**), and gated linear units (**GLU**). Apart from BERT representation of one sentence (*1sent*), we also show BERT representation of one sentence plus one surrounding sentence to the left and one to the right (*1sent+1sur*). The best result in each dataset is shown in bold. Statistical significance tests by bootstrap resampling ($*$: $p < 0.05$) compare 1nn (1sent+1sur) with each of Simple (1sent+1sur), LW (1sent+1sur), GLU (1sent+1sur), and GLU+LW (1sent+1sur).

For Chinese WSD evaluation, we train IMS (Zhong and Ng, 2010) on the Chinese OntoNotes dataset as our baseline. We also incorporate pre-trained non-contextualized Chinese word embeddings as IMS features (Taghipour and Ng, 2015; Iacobacci et al., 2016). The pre-trained word embeddings are obtained by training the *word2vec* skip-gram model on Chinese Gigaword Fifth Edition[2], which after automatic word segmentation contains approximately 2 billion words. Following (Taghipour and Ng, 2015), we incorporate the embedding features of words within a window surrounding the target ambiguous word. In our experiments, we take into account 5 words to the left and right.

[2]LDC2011T13

## 6 Results

We present our experimental results and compare them with prior baselines.

### 6.1 English All-Words Tasks

For English all-words WSD, we compare our approach with three categories of prior approaches. Firstly, we compare our approach with the supervised SVM classifier approach, namely IMS (Zhong and Ng, 2010). We compare our approach with both the original IMS without word embedding features and IMS with non-contextualized word embedding features, that is, *word2vec* with exponential decay (Iacobacci et al., 2016). We also compare with SupWSD (Papandrea et al., 2017), which is an alternative implementation of IMS. Secondly, we compare our approach with the neural WSD approaches that leverage bidirectional LSTM (bi-LSTM). These include the

bi-LSTM model with attention trained jointly with lexical semantic labeling task (Raganato et al., 2017b) (BiLSTMatt+LEX) and the bi-LSTM model enhanced with gloss representation from WordNet (GAS). Thirdly, we show comparison with prior contextualized word representations for WSD, pre-trained on a large number of texts, namely *context2vec* (Melamud et al., 2016) and ELMo (Peters et al., 2018). In these two models, WSD is treated as nearest neighbor matching as described in Section 4.1.

Table 2 shows our WSD results in F1 measure. It is shown in the table that with the nearest neighbor matching model, BERT outperforms *context2vec* and ELMo. This shows the effectiveness of BERT's pre-trained contextualized word representation. When we include surrounding sentences, one to the left and one to the right, we get improved F1 scores consistently.

We also show that linear projection to the sense output vector further improves WSD performance, by which our best results are achieved. While BERT has been shown to outperform other pre-trained contextualized word representations through the nearest neighbor matching experiments, its potential can be maximized through linear projection to the sense output vector. It is worthwhile to note that our more advanced linear projection, by means of layer weighting (§4.2.2 and gated linear unit (§4.2.3) gives the best F1 scores on all test sets.

All our BERT WSD systems outperform gloss-enhanced neural WSD, which has the best overall score among all prior systems.

### 6.2 English Lexical Sample Tasks

For English lexical sample tasks, we compare our approach with the original IMS (Zhong and Ng, 2010) and IMS with non-contextualized word embedding features. The embedding features incorporated into IMS include CW embeddings (Collobert et al., 2011), obtained from a convolutional language model, fine-tuned (adapted) to WSD (Taghipour and Ng, 2015) (+*adapted CW*), and *word2vec* skip-gram (Mikolov et al., 2013) with exponential decay (Iacobacci et al., 2016) (+*w2v+expdecay*). We also compare our approach with the bi-LSTM, on top of which sense classification is defined (Kågebäck and Salomonsson, 2016), and *context2vec* (Melamud et al., 2016), which is a contextualized pre-trained bi-LSTM model trained on 2B words of text. Finally, we also compare with a prior multi-task and semi-supervised WSD approach learned through alternating structure optimization (ASO) (Ando, 2006), which also utilizes unlabeled data for training.

| System | SE2 | SE3 |
|---|---|---|
| *Reported* | | |
| IMS | 65.2 | 72.3 |
| IMS+adapted CW | 66.2 | 73.4 |
| IMS+w2v+expdecay | 69.9 | 75.2 |
| BiLSTM | 66.9 | 73.4 |
| context2vec | – | 72.8 |
| ASO+multitask+semisup | – | 74.1 |
| *BERT nearest neighbor (ours)* | | |
| 1nn (1sent) | 67.7 | 72.7 |
| 1nn (1sent+1sur) | 71.5 | 75.7 |
| *BERT linear projection (ours)* | | |
| Simple (1sent) | 73.3 | 78.6 |
| Simple (1sent+1sur) | **76.9**$^*$ | **80.0**$^*$ |
| LW (1sent+1sur) | 76.7$^*$ | **80.0**$^*$ |
| GLU (1sent+1sur) | 75.1$^*$ | 79.4$^*$ |
| GLU+LW (1sent+1sur) | 74.2$^*$ | 79.8$^*$ |

Table 3: English lexical sample task results in accuracy (%), averaged over three runs. Best accuracy in each dataset is shown in bold. Statistical significance tests by bootstrap resampling ($*$: $p < 0.05$) compare 1nn (1sent+1sur) with each of Simple (1sent+1sur), LW (1sent+1sur), GLU (1sent+1sur), and GLU+LW (1sent+1sur).

As shown in Table 3, our BERT-based WSD approach with linear projection model outperforms all prior approaches. *context2vec*, which is pre-trained on a large amount of texts, performs worse than the prior semi-supervised ASO approach on Senseval-3, while our best result outperforms ASO by a large margin.

Neural bi-LSTM performs worse than IMS with non-contextualized word embedding features. Our neural model with pre-trained contextualized word representations outperforms the best result achieved by IMS on both Senseval-2 and Senseval-3.

### 6.3 Chinese OntoNotes WSD

We compare our approach with IMS without and with word embedding features as the baselines. The results are shown in Table 4.

Across all genres, BERT outperforms the baseline IMS with word embedding (non-contextualized word representation) features (Taghipour and Ng, 2015). The latter also improves over the original IMS without word embedding features (Zhong and Ng, 2010). Linear

| System | BC | BN | MZ | NW | All |
|---|---|---|---|---|---|
| *Baseline* | | | | | |
| IMS | 79.8 | 85.7 | 83.0 | 91.0 | 84.8 |
| IMS+w2v | 80.2 | 86.4 | 82.3 | 92.0 | 85.2 |
| *BERT* | | | | | |
| 1nn | 81.7 | 88.5 | 85.1 | 93.3 | 87.1 |
| Simple | 84.6* | **90.4*** | **88.9*** | 93.9 | **89.5*** |
| LW | **84.7*** | 90.3* | 88.8* | **94.0** | **89.5*** |
| GLU | 84.6* | 90.0* | 88.3* | 93.3 | 89.0* |
| GLU+LW | 84.6* | 90.2* | 88.2* | 93.4 | 89.2* |

Table 4: Chinese OntoNotes WSD results in accuracy (%), averaged over three runs, for each genre. All BERT results in this table are obtained from the representation of one sentence plus one surrounding sentence to the left and to the right (*1sent+1sur*). We show results of various BERT incorporation strategy, namely nearest neighbor matching (**1nn**), **simple** projection, projection with layer weighting (**LW**) and gated linear unit (**GLU**). Best accuracy in each genre is shown in bold. Statistical significance tests by bootstrap resampling ($*$: $p < 0.05$) compare 1nn with each of Simple, LW, GLU, and GLU+LW.

projection approaches consistently outperform nearest neighbor matching by a significant margin, similar to the results on English WSD tasks.

The best overall result for the Chinese OntoNotes test set is achieved by the models with simple projection and hidden layer weighting.

## 7 Discussion

Across all tasks (English all-words, English lexical sample, and Chinese OntoNotes), our experiments demonstrate the effectiveness of BERT over various prior WSD approaches. The best results are consistently obtained by linear projection models, which project the last hidden layer or the weighted sum of all hidden layers to an output sense vector.

We can view the BERT hidden layer outputs as contextual features, which serve as useful cues in determining the word senses. In fact, the attention mechanism in transformer captures the surrounding words. In prior work like IMS (Zhong and Ng, 2010), these contextual cues are captured by the manually-defined surrounding word and collocation features. The features obtained by the hidden vector output are shown to be more effective than the manually-defined features.

We introduced two advanced linear projection techniques, namely layer weighting and gated linear unit (GLU). While Peters et al. (2018) showed that the second biLSTM layer results in better WSD accuracy compared to the first layer (nearer to the individual word representation), we showed that taking into account different layers by means of the attention mechanism is useful for WSD. GLU as an activation function has been shown to be effective for better convergence and to overcome the vanishing gradient problem in the convolutional language model (Dauphin et al., 2017). In addition, the GLU gate vector, with values ranging from 0 to 1, can be seen as a filter for the features from the hidden layer vector.

Compared with two prior contextualized word representations models, *context2vec* (Melamud et al., 2016) and ELMo (Peters et al., 2018), BERT achieves higher accuracy. This shows the effectiveness of the attention mechanism used in the transformer model to represent the context.

Our BERT WSD models outperform prior neural WSD models by a large margin. These prior neural WSD models perform comparably with IMS with embeddings as classifier features, in addition to the discrete features. While neural WSD approaches (Kågebäck and Salomonsson, 2016; Raganato et al., 2017b; Luo et al., 2018) exploit non-contextualized word embeddings which are trained on large texts, the hidden layers are trained only on a small amount of labeled data.

## 8 Conclusion

For the WSD task, we have proposed novel strategies of incorporating BERT, a pre-trained contextualized word representation which effectively captures the context in its hidden vectors. Our experiments show that linear projection of the hidden vectors, coupled with gating to filter the values, gives better results than the prior state of the art. Compared to prior neural and feature-based WSD approaches that make use of non-contextualized word representations, using pre-trained contextualized word representation with our proposed incorporation strategy achieves significantly higher scores.

## References

Rie Kubota Ando. 2006. Applying alternating structure optimization to word sense disambiguation. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 77–84.

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*, abs/1607.06450.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations*.

Yee Seng Chan, Hwee Tou Ng, and David Chiang. 2007a. Word sense disambiguation improves statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 33–40.

Yee Seng Chan, Hwee Tou Ng, and Zhi Zhong. 2007b. NUS-PT: Exploiting parallel texts for word sense disambiguation in the English all-words tasks. In *Proceedings of the Fourth International Workshop on Semantic Evaluations*, pages 253–256.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *Proceedings of the Thirty-Fourth International Conference on Machine Learning*, pages 933–941.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. Embeddings for word sense disambiguation: An evaluation study. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 897–907.

Mikael Kågebäck and Hans Salomonsson. 2016. Word sense disambiguation using a bidirectional LSTM. In *Proceedings of the 5th Workshop on Cognitive Aspects of the Lexicon*, pages 51–56.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*.

Yoong Keok Lee, Hwee Tou Ng, and Tee Kiah Chia. 2004. Supervised word sense disambiguation with support vector machines and multiple knowledge sources. In *Proceedings of SENSEVAL-3, the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 137–140.

Fuli Luo, Tianyu Liu, Qiaolin Xia, Baobao Chang, and Zhifang Sui. 2018. Incorporating glosses into neural word sense disambiguation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 2473–2482.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics,*, pages 1064–1074.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Proceedings of the Thirty-First Conference on Neural Information Processing Systems*, pages 6297–6308.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional LSTM. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 51–61.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of Workshop at the International Conference on Learning Representations*.

George A. Miller, Martin Chodorow, Shari Landes, Claudia Leacock, and Robert G. Thomas. 1994. Using a semantic concordance for sense identification. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey*, pages 240–243.

Simone Papandrea, Alessandro Raganato, and Claudio Delli Bovi. 2017. SupWSD: A flexible toolkit for supervised word sense disambiguation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 103–108.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2227–2237.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using OntoNotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152.

Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. 2017a. Word sense disambiguation: A unified evaluation framework and empirical comparison. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 99–110.

Alessandro Raganato, Claudio Delli Bovi, and Roberto Navigli. 2017b. Neural sequence learning models for word sense disambiguation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1156–1167.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the Twenty-Eighth Conference on Neural Information Processing Systems*, pages 3104–3112.

Kaveh Taghipour and Hwee Tou Ng. 2015. Semi-supervised word sense disambiguation using word embeddings in general and specific domains. In *Human Language Technologies: The 2015 Annual Conference of the North American Chapter of the ACL*, pages 314–323.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the Thirty-First Conference on Neural Information Processing Systems*, pages 5998–6008.

Zhi Zhong and Hwee Tou Ng. 2010. It Makes Sense: a wide-coverage word sense disambiguation system for free text. In *Proceedings of the ACL 2010 System Demonstrations*, pages 78–83.

Zhi Zhong and Hwee Tou Ng. 2012. Word sense disambiguation improves information retrieval. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 273–282.

# Do NLP Models Know Numbers? Probing Numeracy in Embeddings

**Eric Wallace**[*1], **Yizhong Wang**[*2], **Sujian Li**[2], **Sameer Singh**[3], **Matt Gardner**[1]
[1]Allen Institute for Artificial Intelligence
[2]Peking University
[3]University of California, Irvine
{ericw, mattg}@allenai.org, {yizhong, lisujian}@pku.edu.cn, sameer@uci.edu

## Abstract

The ability to understand and work with numbers (numeracy) is critical for many complex reasoning tasks. Currently, most NLP models treat numbers in text in the same way as other tokens—they embed them as distributed vectors. Is this enough to capture numeracy? We begin by investigating the numerical reasoning capabilities of a state-of-the-art question answering model on the DROP dataset. We find this model excels on questions that require numerical reasoning, i.e., it already captures numeracy. To understand how this capability emerges, we probe token embedding methods (e.g., BERT, GloVe) on synthetic list maximum, number decoding, and addition tasks. A surprising degree of numeracy is naturally present in standard embeddings. For example, GloVe and word2vec accurately encode magnitude for numbers up to 1,000. Furthermore, character-level embeddings are even more precise—ELMo captures numeracy the best for all pre-trained methods—but BERT, which uses sub-word units, is less exact.

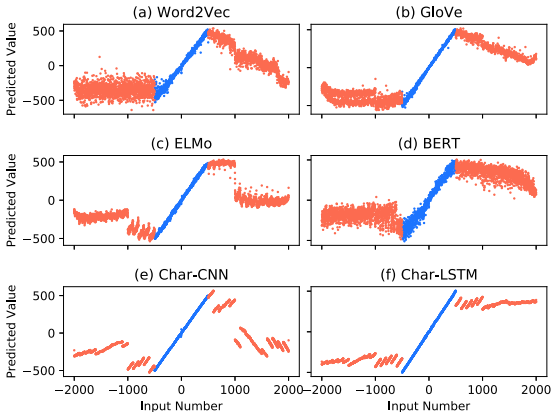


Figure 1: We train a probing model to decode a number from its word embedding over a random 80% of the integers from [-500, 500], e.g., "71" → 71.0. We plot the model's predictions for all numbers from [-2000, 2000]. The model accurately decodes numbers within the training range (in blue), i.e., pre-trained embeddings like GloVe and BERT capture numeracy. However, the probe fails to extrapolate to larger numbers (in red). The Char-CNN (e) and Char-LSTM (f) are trained jointly with the probing model.

## 1 Introduction

Neural NLP models have become the de-facto standard tool across language understanding tasks, even solving basic reading comprehension and textual entailment datasets (Yu et al., 2018; Devlin et al., 2019). Despite this, existing models are incapable of complex forms of reasoning, in particular, we focus on the ability to reason *numerically*. Recent datasets such as DROP (Dua et al., 2019), EQUATE (Ravichander et al., 2019), or Mathematics Questions (Saxton et al., 2019) test numerical reasoning; they contain examples which require comparing, sorting, and adding numbers in natural language (e.g., Figure 2).

The first step in performing numerical reasoning over natural language is *numeracy*: the ability to understand and work with numbers in either digit or word form (Spithourakis and Riedel, 2018). For example, one must understand that the string "23" represents a bigger value than "twenty-two". Once a number's value is (perhaps implicitly) represented, reasoning algorithms can then process the text, e.g., extracting the list of field goals and computing that list's maximum (first question in Figure 2). Learning to reason numerically over paragraphs with only question-answer supervision appears daunting for end-to-end models; our work seeks to understand *if* and *how* "out-of-the-box" neural NLP models already learn this.

We begin by analyzing the state-of-the-art NAQANet model (Dua et al., 2019) for DROP—testing it on a subset of questions that evaluate numerical reasoning (Section 2). To our surprise,

[*]Equal contribution; work done while interning at AI2.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5307–5315,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

the model exhibits excellent numerical reasoning abilities. Amidst reading and comprehending natural language, the model successfully computes list maximums/minimums, extracts superlative entities (argmax reasoning), and compares numerical quantities. For instance, despite NAQANet achieving only 49 F1 on the entire validation set, it scores 89 F1 on numerical comparison questions. We also stress test the model by perturbing the validation paragraphs and find one failure mode: the model struggles to extrapolate to numbers outside its training range.

We are especially intrigued by the model's ability to learn numeracy, i.e., how does the model know the value of a number given its embedding? The model uses standard embeddings (GloVe and a Char-CNN) and receives no direct supervision for number magnitude/ordering. To understand how numeracy emerges, we probe token embedding methods (e.g., BERT, GloVe) using synthetic list maximum, number decoding, and addition tasks (Section 3).

We find that all widely-used pre-trained embeddings, e.g., ELMo (Peters et al., 2018), BERT (Devlin et al., 2019), and GloVe (Pennington et al., 2014), capture numeracy: number magnitude is present in the embeddings, even for numbers in the thousands. Among all embeddings, character-level methods exhibit stronger numeracy than word- and sub-word-level methods (e.g., ELMo excels while BERT struggles), and character-level models learned directly on the synthetic tasks are the strongest overall. Finally, we investigate why NAQANet had trouble extrapolating—was it a failure in the model or the embeddings? We repeat our probing tasks and test for model extrapolation, finding that neural models struggle to predict numbers outside the training range.

## 2 Numeracy Case Study: DROP QA

This section examines the state-of-the-art model for DROP by investigating its accuracy on questions that require numerical reasoning.

### 2.1 DROP Dataset

DROP is a reading comprehension dataset that tests numerical reasoning operations such as counting, sorting, and addition (Dua et al., 2019). The dataset's input-output format is a superset of SQuAD (Rajpurkar et al., 2016): the answers are paragraph spans, as well as question spans, number answers (e.g., 35), and dates (e.g., 03/01/2014). The only supervision provided is the question-answer pairs, i.e., a model must learn to reason numerically while simultaneously learning to read and comprehend.

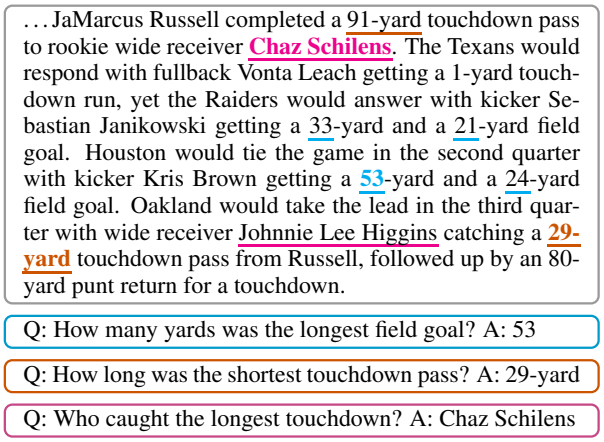


Figure 2: Three DROP questions that require numerical reasoning; the state-of-the-art NAQANet answers every question correct. Plausible answer candidates to the questions are underlined and the model's predictions are shown in bold.

### 2.2 NAQANet Model

Modeling approaches for DROP include both semantic parsing (Krishnamurthy et al., 2017) and reading comprehension (Yu et al., 2018) models. We focus on the latter, specifically on Numerically-augmented QANet (NAQANet), the current state-of-the-art model (Dua et al., 2019).[1] The model's core structure closely follows QANet (Yu et al., 2018) except that it contains four output branches, one for each of the four answer types (passage span, question span, count answer, or addition/subtraction of numbers.)

Words and numbers are represented as the concatenation of GloVe embeddings and the output of a character-level CNN. The model contains no auxiliary components for representing number magnitude or performing explicit comparisons. We refer readers to Yu et al. (2018) and Dua et al. (2019) for further details.

### 2.3 Comparative and Superlative Questions

We focus on questions that NAQANet requires numeracy to answer, namely *Comparative* and *Superlative* questions.[2] Comparative questions

[1] Result as of May 21st, 2019.

[2] DROP addition, subtraction, and count questions do not require numeracy for NAQANet, see Appendix A.

| Question Type | Example | Reasoning Required |
|---|---|---|
| Comparative (Binary) | Which country is a bigger exporter, Brazil or Uruguay? | Binary Comparison |
| Comparative (Non-binary) | Which player had a touchdown longer than 20 yards? | Greater Than |
| Superlative (Number) | How many yards was the shortest field goal? | List Minimum |
| Superlative (Span) | Who kicked the longest field goal? | Argmax |

Table 1: We focus on DROP *Comparative* and *Superlative* questions which test NAQANet's numeracy.

probe a model's understanding of quantities or events that are "larger", "smaller", or "longer" than others. Certain comparative questions ask about "either-or" relations (e.g., first row of Table 1), which test binary comparison. Other comparative questions require more diverse comparative reasoning, such as greater than relationships (e.g., second row of Table 1).

Superlative questions ask about the "shortest", "largest", or "biggest" quantity in a passage. When the answer type is a *number*, superlative questions require finding the maximum or minimum of a list (e.g., third row of Table 1). When the answer type is a *span*, superlative questions usually require an argmax operation, i.e., one must find the superlative action or quantity and then extract the associated entity (e.g., fourth row of Table 1). We filter the validation set to comparative and superlative questions by writing templates to match words in the question.

### 2.4 Emergent Numeracy in NAQANet

NAQANet's accuracy on comparative and superlative questions is significantly higher than its average accuracy on the validation set (Table 2).[3]

NAQANet achieves 89.0 F1 on binary (either-or) comparative questions, approximately 40 F1 points higher than the average validation question and within 7 F1 points of human test performance. The model achieves a lower, but respectable, accuracy on non-binary comparisons. These questions require multiple reasoning steps, e.g., the second question in Table 1 requires (1) extracting all the touchdown distances, (2) finding the distance that is greater than twenty, and (3) selecting the player associated with the touchdown of that distance.

We divide the superlative questions into questions that have number answers and questions with span answers according to the dataset's provided answer type. NAQANet achieves nearly 70 F1 on superlative questions with number answers, i.e., it can compute list maximum and minimums.

[3]We have a public NAQANet demo available https://demo.allennlp.org/reading-comprehension.

| Question Type | Count | EM | F1 |
|---|---|---|---|
| Human (Test Set) | 9622 | 92.4 | 96.0 |
| Full Validation | 9536 | 46.2 | 49.2 |
| Number Answers | 5842 | 44.3 | 44.4 |
| Comparative | 704 | 73.6 | 76.4 |
| Binary (either-or) | 477 | 86.0 | 89.0 |
| Non-binary | 227 | 47.6 | 49.8 |
| Superlative Questions | 861 | 64.6 | 67.7 |
| Number Answers | 475 | 68.8 | 69.2 |
| Span Answers | 380 | 59.7 | 66.3 |

Table 2: NAQANet achieves higher accuracy on questions that require numerical reasoning (*Superlative* and *Comparative*) than on standard validation questions. Human performance is reported from Dua et al. (2019).

The model answers about two-thirds of superlative questions with span answers correctly (66.3 F1), i.e., it can perform argmax reasoning.

Figure 2 shows examples of superlative questions answered correctly by NAQANet. The first two questions require computing the maximum/minimum of a list: the model must recognize which digits correspond to field goals and touchdowns passes, and then extract the maximum/minimum of the correct list. The third question requires argmax reasoning: the model must first compute the longest touchdown pass and then find the corresponding receiver "Chaz Schilens".

### 2.5 Stress Testing NAQANet's Numeracy

Just how far does the numeracy of NAQANet go? Here, we stress test the model by automatically modifying DROP validation paragraphs.

We test two phenomena: larger numbers and word-form numbers. For larger numbers, we generate a random positive integer and multiply or add that value to the numbers in each paragraph. For word forms, we replace every digit in the paragraph with its word form (e.g., "75" → "seventy-five"). Since word-form numbers are usually small in magnitude when they occur in DROP, we perform word replacements for integers in the range [0, 100]. We guarantee the ground-truth answer is

| Stress Test Dataset | All Questions F1 | All Questions Δ | Superlative F1 | Superlative Δ |
|---|---|---|---|---|
| Original Validation Set | 49.2 | - | 67.7 | - |
| Add [1, 20] | 47.7 | -1.5 | 64.1 | -3.6 |
| Add [21, 100] | 41.4 | -7.8 | 40.4 | -27.3 |
| Multiply [2, 10] | 41.1 | -8.1 | 39.3 | -28.4 |
| Multiply [11, 100] | 38.8 | -10.4 | 32.0 | -35.7 |
| Digits to Words [0, 20] | 45.5 | -3.7 | 63.8 | -3.9 |
| Digits to Words [21, 100] | 41.9 | -7.3 | 46.1 | -21.6 |

Table 3: We stress test NAQANet's numeracy by manipulating the numbers in the validation paragraphs. *Add* or *Multiply [x, y]* indicates adding or multiplying all of the numbers in the passage by a random integer in the range [x, y]. *Digits → Words [x, y]* converts all integers in the passage within the range [x, y] to their corresponding word form (e.g., "75" → "seventy-five").

still valid by only modifying NAQANet's internal representation (Appendix E).

Table 3 shows the results for different paragraph modifications. The model exhibits a tiny degradation in performance for small magnitude changes (e.g., NAQANet drops 1.5 F1 overall for Add [1,20]) but severely struggles on larger changes (e.g., NAQANet drops 35.7 F1 on superlative questions for Multiply [11,200]). Similar trends hold for word forms: the model exhibits small drops in accuracy when converting small numbers to words (3.9 degradation on Digits to Words [0,20]) but fails on larger magnitude word forms (21.6 F1 drop over [21,100]). These results show that NAQANet has a strong understanding of numeracy for numbers in the training range, but, the model can fail to extrapolate to other values.

### 2.6 Whence this behavior?

NAQANet exhibits numerical reasoning capabilities that exceed our expectations. What enables this behavior? Aside from reading and comprehending the passage/question, this kind of numerical reasoning requires two components: numeracy (i.e., representing numbers) and comparison algorithms (i.e., computing the maximum of a list).

Although the natural emergence of comparison algorithms is surprising, previous results show neural models are capable of learning to count and sort synthetic lists of scalar values when given explicit supervision (Weiss et al., 2018; Vinyals et al., 2016). NAQANet demonstrates that a model can learn comparison algorithms while simultaneously learning to read and comprehend, even with only question-answer supervision.

How, then, does NAQANet know numeracy? The source of numerical information eventually lies in the token embeddings themselves, i.e., the character-level convolutions and GloVe embeddings of the NAQANet model. Therefore, we can understand the source of numeracy by isolating and probing these embeddings.

## 3 Probing Numeracy of Embeddings

We use synthetic numerical tasks to probe the numeracy of token embeddings.

### 3.1 Probing Tasks

We consider three synthetic tasks to evaluate numeracy (Figure 3). Appendix C provides further details on training and evaluation.

**List Maximum** Given a list of the embeddings for five numbers, the task is to predict the index of the maximum number. Each list consists of values of similar magnitude in order to evaluate fine-grained comparisons (see Appendix C). As in typical span selection models (Seo et al., 2017), an LSTM reads the list of token embeddings, and a weight matrix and softmax function assign a probability to each index using the model's hidden state. We use the negative log-likelihood of the maximum number as the loss function.

**Decoding** The decoding task probes whether number magnitude is captured (rather than the relative ordering of numbers as in list maximum). Given a number's embedding, the task is to regress to its value, e.g., the embedding for the string "five" has a target of 5.0. We consider a linear regression model and a three-layer fully-connected network with ReLU activations. The models are trained using a mean squared error (MSE) loss.

**Addition** The addition task requires number manipulation—given the embeddings of two numbers, the task is to predict their sum. Our model concatenates the two token embeddings and feeds the result through a three-layer fully-connected network with ReLU activations, trained using MSE loss. Unlike the decoding task, the model needs to capture number magnitude internally without direct label supervision.

**Training and Evaluation** We focus on a numerical *interpolation* setting (we revisit extrapolation

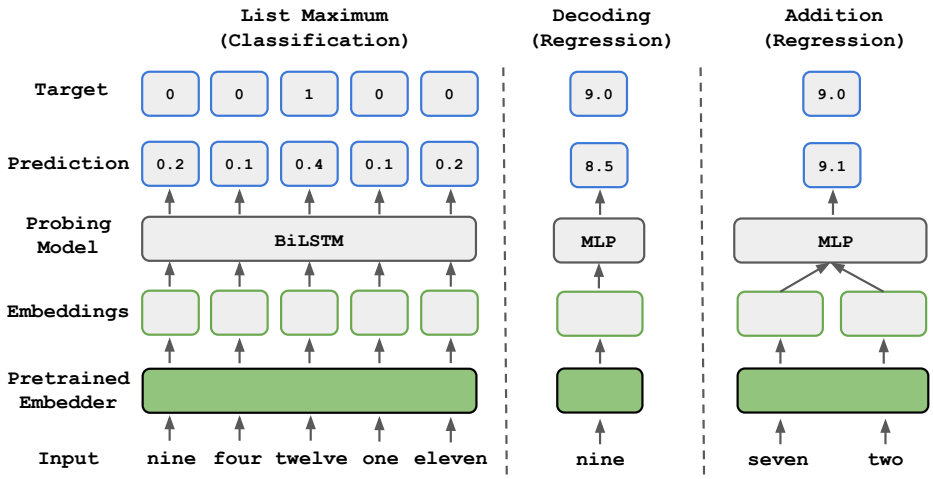


Figure 3: Our probing setup. We pass numbers through a pre-trained embedder (e.g., BERT, GloVe) and train a probing model to solve numerical tasks such as finding a list's maximum, decoding a number, or adding two numbers. If the probing model generalizes to held-out numbers, the pre-trained embeddings must contain numerical information. We provide numbers as either words (shown here), digits ("9"), floats ("9.1"), or negatives ("-9").

in Section 3.4): the model is tested on values that are within the training range. We first pick a range (we vary the range in our experiments) and randomly shuffle the integers over it. We then split 80% of the numbers into a training set and 20% into a test set. We report the mean and standard deviation across five different random shuffles for a particular range, using the exact same shuffles across all embedding methods.

Numbers are provided as integers ("75"), single-word form ("seventy-five"), floats ("75.1"), or negatives ("-75"). We consider positive numbers less than 100 for word-form numbers to avoid multiple tokens. We report the classification accuracy for the list maximum task (5 classes), and the Root Mean Squared Error (RMSE) for decoding and addition. Note that larger ranges will naturally amplify the RMSE error.

### 3.2 Embedding Methods

We evaluate various token embedding methods.

**Word Vectors** We use 300-dimensional GloVe (Pennington et al., 2014) and word2vec vectors (Mikolov et al., 2018). We ensure all values are in-vocabulary for word vectors.

**Contextualized Embeddings** We use ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019) embeddings.[4] ELMo uses character-level convolutions of size 1–7 with max pooling. BERT represents tokens via sub-word pieces; we use lowercased BERT-base with 30k pieces.

**NAQANet Embeddings** We extract the GloVe embeddings and Char-CNN from the NAQANet model trained on DROP. We also consider an ablation that removes the GloVe embeddings.

**Learned Embeddings** We use a character-level CNN (Char-CNN) and a character-Level LSTM (Char-LSTM). We use left character padding, which greatly improves numeracy for character-level CNNs (details in Appendix B).

**Untrained Embeddings** We consider two untrained baselines. The first baseline is random token vectors, which trivially fail to generalize (there is no pattern between train and test numbers). These embeddings are useful for measuring the improvement of pre-trained embeddings. We also consider a randomly initialized and untrained Char-CNN and Char-LSTM.

**Number's Value as Embedding** The final embedding method is simple: map a number's embedding directly to its value (e.g., "seventy-five" embeds to [75]). We found this strategy performs poorly for large ranges; using a base-10 logarithmic scale improves performance. We report this as *Value Embedding* in our results.[5]

[4]Since our inputs are numbers, not natural sentences, language models may exhibit strange behavior. We experimented with extracting the context-independent feature vector immediately following the character convolutions for ELMo but found little difference in results.

[5]We suspect the failures result from the raw values being too high in magnitude and/or variance for the model. We also experimented with normalizing the values to mean 0 and variance 1; a logarithmic scale performed better.

All pre-trained embeddings (all methods except the Char-CNN and Char-LSTM) are fixed during training. The probing models are trained on the synthetic tasks on top of these embeddings.

### 3.3 Results: Embeddings Capture Numeracy

We find that all pre-trained embeddings *contain* fine-grained information about number magnitude and order. We first focus on integers (Table 4).

**Word Vectors Succeed** Both word2vec and GloVe significantly outperform the random vector baseline and are among the strongest methods overall. This is particularly surprising given the training methodology for these embeddings, e.g., a continuous bag of words objective can teach fine-grained number magnitude.

**Character-level Methods Dominate** Models which use character-level information have a clear advantage over word-level models for encoding numbers. This is reflected in our probing results: character-level CNNs are the best architecture for capturing numeracy. For example, the NAQANet model without GloVe (only using its Char-CNN) and ELMo (uses a Char-CNN) are the strongest pre-trained methods, and a learned Char-CNN is the strongest method overall. The strength of the character-level convolutions seems to lie in the architectural prior—an *untrained* Char-CNN is surprisingly competitive. Similar results have been shown for images (Saxe et al., 2011): random CNNs are powerful feature extractors.

**Sub-word Models Struggle** BERT struggles for large ranges (e.g., 52% accuracy for list maximum for [0,9999]). We suspect this results from sub-word pieces being a poor method to encode digits: two numbers which are similar in value can have very different sub-word divisions.

**A Linear Subspace Exists** For small ranges on the decoding task (e.g., [0,99]), a linear model is competitive, i.e., a linear subspace captures number magnitude (Appendix D). For larger ranges (e.g., [0,999]), the linear model's performance degrades, especially for BERT.

**Value Embedding Fails** The Value Embedding method fails for large ranges. This is surprising as the embedding directly provides a number's value, thus, the synthetic tasks *should* be easy to solve. However, we had difficulty training models for large ranges, even when using numerous architecture variants (e.g., tiny networks with 10 hidden units and tanh activations) and hyperparameters. Trask et al. (2018) discuss similar problems and ameliorate them using new neural architectures.

**Words, Floats, and Negatives are Captured** Finally, we probe the embeddings on word-form numbers, floats, and negatives. We observe similar trends for these inputs as integers: pre-trained models exhibit natural numeracy and learned embeddings are strong (Tables 5, 6, and 10). The ordering of the different embedding methods according to performance is also relatively consistent across the different input types. One notable exception is that BERT struggles on floats, which is likely a result of its sub-word pieces. We do not test word2vec and GloVe on floats/negatives because they are out-of-vocabulary.

### 3.4 Probing Models Struggle to Extrapolate

Thus far, our synthetic experiments evaluate on held-out values *within* the same range as the training data (i.e., numerical interpolation). In Section 2.5, we found that NAQANet struggles to extrapolate to values *outside* the training range. Is this an idiosyncrasy of NAQANet or is it a more general problem? We investigate this using a numerical extrapolation setting: we train models on a specific integer range and test them on values greater than the largest training number and smaller than the smallest training number.

**Extrapolation for Decoding and Addition** For decoding and addition, models struggle to extrapolate. Figure 1 shows the predictions for models trained on 80% of the values from [-500,500] and tested on held-out numbers in the range [-2000, 2000] for six embedding types. The embedding methods fail to extrapolate in different ways, e.g., predictions using word2vec decrease almost monotonically as the input increases, while predictions using BERT are usually near the highest training value. Trask et al. (2018) also observe that models struggle outside the training range; they attribute this to failures in neural models themselves.

**Extrapolation for List Maximum** For the list maximum task, accuracies are closer to those in the interpolation setting, however, they still fall short. Table 7 shows the accuracy for models trained on the integer range [0,150] and tested on

| **Interpolation** | **List Maximum (5-classes)** | | | **Decoding (RMSE)** | | | **Addition (RMSE)** | | |
|---|---|---|---|---|---|---|---|---|---|
| *Integer Range* | [0,99] | [0,999] | [0,9999] | [0,99] | [0,999] | [0,9999] | [0,99] | [0,999] | [0,9999] |
| Random Vectors | 0.16 | 0.23 | 0.21 | 29.86 | 292.88 | 2882.62 | 42.03 | 410.33 | 4389.39 |
| Untrained CNN | 0.97 | 0.87 | 0.84 | 2.64 | 9.67 | 44.40 | 1.41 | 14.43 | 69.14 |
| Untrained LSTM | 0.70 | 0.66 | 0.55 | 7.61 | 46.5 | 210.34 | 5.11 | 45.69 | 510.19 |
| Value Embedding | **0.99** | 0.88 | 0.68 | **1.20** | 11.23 | 275.50 | **0.30** | 15.98 | 654.33 |
| *Pre-trained* | | | | | | | | | |
| Word2Vec | 0.90 | 0.78 | 0.71 | 2.34 | 18.77 | 333.47 | 0.75 | 21.23 | 210.07 |
| GloVe | 0.90 | 0.78 | 0.72 | 2.23 | 13.77 | 174.21 | 0.80 | 16.51 | 180.31 |
| ELMo | 0.98 | 0.88 | 0.76 | 2.35 | 13.48 | 62.20 | 0.94 | 15.50 | 45.71 |
| BERT | 0.95 | 0.62 | 0.52 | 3.21 | 29.00 | 431.78 | 4.56 | 67.81 | 454.78 |
| *Learned* | | | | | | | | | |
| Char-CNN | 0.97 | **0.93** | **0.88** | 2.50 | **4.92** | **11.57** | 1.19 | **7.75** | **15.09** |
| Char-LSTM | 0.98 | 0.92 | 0.76 | 2.55 | 8.65 | 18.33 | 1.21 | 15.11 | 25.37 |
| *DROP-trained* | | | | | | | | | |
| NAQANet | 0.91 | 0.81 | 0.72 | 2.99 | 14.19 | 62.17 | 1.11 | 11.33 | 90.01 |
| - GloVe | 0.88 | 0.90 | 0.82 | 2.87 | 5.34 | 35.39 | 1.45 | 9.91 | 60.70 |

Table 4: *Interpolation with integers (e.g., "18").* All pre-trained embedding methods (e.g., GloVe and ELMo) surprisingly capture numeracy. The probing model is trained on a randomly shuffled 80% of the *Integer Range* and tested on the remaining 20%. The probing model architecture and train/test splits are equivalent across all embeddings. We show the mean over 5 random shuffles (standard deviation in Appendix D).

| **Interpolation** | **List Maximum (5-classes)** | |
|---|---|---|
| *Float Range* | [0.0,99.9] | [0.0,999.9] |
| Rand. Vectors | $0.18 \pm 0.03$ | $0.21 \pm 0.04$ |
| ELMo | $0.91 \pm 0.03$ | $0.59 \pm 0.01$ |
| BERT | $0.82 \pm 0.05$ | $0.51 \pm 0.04$ |
| Char-CNN | $0.87 \pm 0.04$ | $0.75 \pm 0.03$ |
| Char-LSTM | $0.81 \pm 0.05$ | $0.69 \pm 0.02$ |

Table 5: *Interpolation with floats (e.g., "18.1") for list maximum.* Pre-trained embeddings capture numeracy for float values. The probing model is trained on a randomly shuffled 80% of the *Float Range* and tested on the remaining 20%. See the text for details on selecting decimal values. We show the mean alongside the standard deviation over 5 different random shuffles.

| **Interpolation** | **List Maximum (5-classes)** |
|---|---|
| *Integer Range* | [-50,50] |
| Rand. Vectors | $0.23 \pm 0.12$ |
| Word2Vec | $0.89 \pm 0.02$ |
| GloVe | $0.89 \pm 0.03$ |
| ELMo | $0.96 \pm 0.01$ |
| BERT | $0.94 \pm 0.02$ |
| Char-CNN | $0.95 \pm 0.07$ |
| Char-LSTM | $0.97 \pm 0.02$ |

Table 6: *Interpolation with negatives (e.g., "-18") on list maximum.* Pre-trained embeddings capture numeracy for negative values.

the ranges [151,160], [151,180], and [151,200]; all methods struggle, especially token vectors.

**Augmenting Data to Aid Extrapolation** Of course, in many real-word tasks it is possible to ameliorate these extrapolation failures by augmenting the training data (i.e., turn extrapolation into interpolation). Here, we apply this idea to aid in training NAQANet for DROP. For each superlative and comparative example, we duplicate the example and modify the numbers in its paragraph using the *Add* and *Multiply* techniques described in Section 2.5. Table 11 shows that this data augmentation can improve both interpolation and extrapolation, e.g., the accuracy on superlative questions with large numbers can *double*.

| Extrapolation | List Maximum (5-classes) | | |
|---|---|---|---|
| *Test Range* | [151,160] | [151,180] | [151,200] |
| Rand. Vectors | 0.17 | 0.22 | 0.15 |
| Untrained CNN | 0.80 | 0.47 | 0.41 |
| *Pre-trained* | | | |
| Word2Vec | 0.14 | 0.16 | 0.11 |
| GloVe | 0.19 | 0.17 | 0.21 |
| ELMo | 0.65 | 0.57 | 0.38 |
| BERT | 0.35 | 0.11 | 0.14 |
| *Learned* | | | |
| Char-CNN | 0.81 | 0.75 | 0.73 |
| Char-LSTM | **0.88** | **0.84** | **0.82** |
| *DROP* | | | |
| NAQANet | 0.31 | 0.29 | 0.25 |
| - GloVe | 0.58 | 0.53 | 0.48 |

Table 7: *Extrapolation on list maximum.* The probing model is trained on the integer range [0,150] and evaluated on integers from the *Test Range*. The probing model struggles to extrapolate when trained on the pre-trained embeddings.

## 4 Discussion and Related Work

An open question is how the training process elicits numeracy for word vectors and contextualized embeddings. Understanding this, perhaps by tracing numeracy back to the training data, is a fruitful direction to explore further (c.f., influence functions (Koh and Liang, 2017; Brunet et al., 2019)).

More generally, numeracy is one type of emergent knowledge. For instance, embeddings may capture the size of objects (Forbes and Choi, 2017), speed of vehicles, and many other "commonsense" phenomena (Yang et al., 2018). Vendrov et al. (2016) introduce methods to encode the order of such phenomena into embeddings for concepts such as hypernymy; our work and Yang et al. (2018) show that a relative ordering naturally emerges for certain concepts.

In concurrent work, Naik et al. (2019) also explore numeracy in word vectors. Their methodology is based on variants of nearest neighbors and cosine distance; we use neural network probing classifiers which can capture highly non-linear dependencies between embeddings. We also explore more powerful embedding methods such as ELMo, BERT, and learned embedding methods.

**Probing Models** Our probes of *numeracy* parallel work in understanding the linguistic capabilities (*literacy*) of neural models (Conneau et al., 2018; Liu et al., 2019). LSTMs can remember sentence length, word order, and which words were present in a sentence (Adi et al., 2017). Khandelwal et al. (2018) show how language models leverage context, while Linzen et al. (2016) demonstrate that language models understand subject-verb agreement.

**Numerical Value Prediction** Spithourakis and Riedel (2018) improve the ability of language models to *predict* numbers, i.e., they go beyond categorical predictions over a fixed-size vocabulary. They focus on improving models; our focus is probing embeddings. Kotnis and García-Durán (2019) predict numerical attributes in knowledge bases, e.g., they develop models that try to predict the population of Paris.

**Synthetic Numerical Tasks** Similar to our synthetic numerical reasoning tasks, other work considers sorting (Graves et al., 2014), counting (Weiss et al., 2018), or decoding tasks (Trask et al., 2018). They use synthetic tasks as a testbed to prove or design better models, whereas we use synthetic tasks as a probe to understand token embeddings. In developing the Neural Arithmetic Logic Unit, Trask et al. (2018) arrive at similar conclusions regarding extrapolation: neural models have difficulty outputting numerical values outside the training range.

## 5 Conclusion

How much do NLP models know about numbers? By digging into a surprisingly successful model on a numerical reasoning dataset (DROP), we discover that pre-trained token representations naturally encode numeracy.

We analyze the limits of this numeracy, finding that CNNs are a particularly good prior (and likely the cause of ELMo's superior numeracy compared to BERT) and that it is difficult for neural models to extrapolate beyond the values seen during training. There are still many fruitful areas for future research, including discovering *why* numeracy naturally emerges in embeddings, and what other properties are similarly emergent.

## Acknowledgements

We thank Mark Neumann, Suchin Gururangan, Pranav Goel, Shi Feng, Nikhil Kandpal, Dheeru Dua, the members of AllenNLP and UCI NLP, and the reviewers for their valuable feedback.

## References


Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2017. Finegrained analysis of sentence embeddings using auxiliary prediction tasks. In *ICLR*.

Marc-Etienne Brunet, Colleen Alkalay-Houlihan, Ashton Anderson, and Richard Zemel. 2019. Understanding the origins of bias in word embeddings. In *ICML*.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single vector: Probing sentence embeddings for linguistic properties. In *ACL*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *NAACL*.

Maxwell Forbes and Yejin Choi. 2017. Verb physics: Relative physical knowledge of actions and objects. In *ACL*.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. 2018. Sharp nearby, fuzzy far away: How neural language models use context. In *ACL*.

Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *ICML*.

Bhushan Kotnis and Alberto García-Durán. 2019. Learning numerical attributes in knowledge bases. In *AKBC*.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *EMNLP*.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. In *TACL*.

Nelson F Liu, Matt Gardner, Yonatan Belinkov, Matthew Peters, and Noah A Smith. 2019. Linguistic knowledge and transferability of contextual representations. In *NAACL*.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *LREC*.

Aakanksha Naik, Abhilasha Ravichander, Carolyn Rose, and Eduard Hovy. 2019. Exploring numeracy in word embeddings. In *ACL*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *EMNLP*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *North American Association for Computational Linguistics*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *EMNLP*.

Abhilasha Ravichander, Aakanksha Naik, Carolyn Rose, and Eduard Hovy. 2019. EQUATE: A benchmark evaluation framework for quantitative reasoning in natural language inference. *arXiv preprint arXiv:1901.03735*.

Andrew M Saxe, Pang Wei Koh, Zhenghao Chen, Maneesh Bhand, Bipin Suresh, and Andrew Y Ng. 2011. On random weights and unsupervised feature learning. In *ICML*.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. 2019. Analysing mathematical reasoning abilities of neural models.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *ICLR*.

Georgios Spithourakis and Sebastian Riedel. 2018. Numeracy for language models: Evaluating and improving their ability to predict numbers. In *ACL*.

Andrew Trask, Felix Hill, Scott Reed, Jack W. Rae, Chris Dyer, and Phil Blunsom. 2018. Neural arithmetic logic units. In *NeurIPS*.

Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. 2016. Order-embeddings of images and language. In *ICLR*.

Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. 2016. Order matters: Sequence to sequence for sets. In *ICLR*.

Gail Weiss, Yoav Goldberg, and Eran Yahav. 2018. On the practical computational power of finite precision rnns for language recognition. In *ACL*.

Yiben Yang, Larry Birnbaum, Ji-Ping Wang, and Doug Downey. 2018. Extracting commonsense properties from embeddings with limited human guidance. In *ACL*.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. 2018. QANet: Combining local convolution with global self-attention for reading comprehension. In *ICLR*.

# A Split-and-Recombine Approach for Follow-up Query Analysis

**Qian Liu**[†*], **Bei Chen**[§], **Haoyan Liu**[◇*], **Jian-Guang Lou**[§], **Lei Fang**[§],
**Bin Zhou**[†], **Dongmei Zhang**[§]
[†]State Key Laboratory of Virtual Reality Technology and Systems,
School of Computer Science and Engineering, Beihang University, China
[◇]State Key Lab of Software Development Environment, Beihang University, China
[§]Microsoft Research, Beijing, China
[†◇]{qian.liu, haoyan.liu, zhoubin}@buaa.edu.cn;
[§]{beichen, jlou, leifa, dongmeiz}@microsoft.com

## Abstract

Context-dependent semantic parsing has proven to be an important yet challenging task. To leverage the advances in context-independent semantic parsing, we propose to perform follow-up query analysis, aiming to restate context-dependent natural language queries with contextual information. To accomplish the task, we propose STAR, a novel approach with a well-designed two-phase process. It is parser-independent and able to handle multifarious follow-up scenarios in different domains. Experiments on the FollowUp dataset show that STAR outperforms the state-of-the-art baseline by a large margin of nearly 8%. The superiority on parsing results verifies the feasibility of follow-up query analysis. We also explore the extensibility of STAR on the SQA dataset, which is very promising.

## 1 Introduction

Recently, Natural Language Interfaces to Databases (NLIDB) has received considerable attention, as they allow users to query databases by directly using natural language. Current studies mainly focus on context-independent semantic parsing, which translates a single natural language sentence into its corresponding executable form (e.g. Structured Query Language) and retrieves the answer from databases regardless of its context. However, context does matter in real world applications. Users tend to issue queries in a coherent way when communicating with NLIDB. For example, after the query "How much money has Smith earned?" (**Precedent Query**), users may pose another query by simply asking "How about Bill Collins?" (**Follow-up Query**) instead of the complete "How much money has Bill Collins earned?" (**Restated Query**). Therefore, contextual information is essential for more accurate and robust semantic parsing, namely context-dependent semantic parsing.

Compared with context-independent semantic parsing, context-dependent semantic parsing has received less attention. Several attempts include a statistical model with parser trees (Miller et al., 1996), a linear model with context-dependent logical forms (Zettlemoyer and Collins, 2009) and a sequence-to-sequence model (Suhr et al., 2018). However, all these methods cannot apply to different domains, since the ATIS dataset (Dahl et al., 1994) they rely on is domain-specific. A search-based neural method DynSP* arises along with the SequentialQA (SQA) dataset (Iyyer et al., 2017), which takes the first step towards cross-domain context-dependent semantic parsing. Nevertheless, DynSP* focuses on dealing with relatively simple scenarios. All the aforementioned methods design context-dependent semantic parser from scratch. Instead, inspired by Liu et al. (2019), we propose to directly leverage the technical advances in context-independent semantic parsing. We define *follow-up query analysis* as restating the follow-up queries using contextual information in natural language, then the restated queries can be translated to the corresponding executable forms by existing context-independent parsers. In this way, we boost the performance of context-dependent semantic parsing.

In this paper, we focus on follow-up query analysis and present a novel approach. The main idea is to decompose the task into two phases by introducing a learnable intermediate structure *span*: two queries first get split into several spans, and then undergo the recombination process. As no intermediate annotation is involved, we design re-

*Work done during an internship at Microsoft Research.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5316–5326,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

wards to jointly train the two phases by applying reinforcement learning (RL) (Sutton and Barto, 1998). Our major contributions are as follows:

- We propose a novel approach, named **S**pli**T**-**A**nd-**R**ecombine (STAR)[1], to restate follow-up queries via two phases. It is parser-independent and can be seamlessly integrated with existing context-independent semantic parsers.
- We conduct experiments on the FollowUp dataset (Liu et al., 2019), which covers multifarious cross-domain follow-up scenarios. The results demonstrate that our approach significantly outperforms the state-of-the-art baseline.
- We redesign the recombination process and extend STAR to the SQA dataset, where the annotations are answers. Experiments show promising results, that demonstrates the extensibility of our approach.

## 2 Methodology

In this section, we first give an overview of our proposed method with the idea of two-phase process, then introduce the two phases in turn.

### 2.1 Overview of Split-And-Recombine

Let $\mathbf{x} = (x_1, \ldots, x_n)$, $\mathbf{y} = (y_1, \ldots, y_m)$ and $\mathbf{z} = (z_1, \ldots, z_l)$ denote the precedent query, follow-up query and restated query respectively, each of which is a natural language sentence. Our goal is to interpret the follow-up query $\mathbf{y}$ with its precedent query $\mathbf{x}$ as context, and generate the corresponding restated query $\mathbf{z}$. The restated query has the same meaning with the follow-up query, but it is complete and unambiguous to facilitate better downstream parsing. Formally, given the pair $(\mathbf{x}, \mathbf{y})$, we aim to learn a model $P_{\text{model}}(\mathbf{z}|\mathbf{x}, \mathbf{y})$ and maximize the objective:

$$\mathcal{L} = \mathbb{E}_{(\mathbf{x},\mathbf{y},\mathbf{z})\sim\mathcal{D}}[\log P_{\text{model}}(\mathbf{z}|\mathbf{x}, \mathbf{y})], \quad (1)$$

where $\mathcal{D}$ represents the set of training data. As to $P_{\text{model}}(\mathbf{z}|\mathbf{x}, \mathbf{y})$, since $\mathbf{z}$ always overlaps a great with $\mathbf{x}$ and $\mathbf{y}$, it is intuitively more straightforward to find a way to merge $\mathbf{x}$ and $\mathbf{y}$. To this end, we design a two-phase process and present a novel approach STAR to perform follow-up query analysis with reinforcement learning.

A concrete example of the two-phase process is shown in Figure 1. Phase I is to **Split** input queries

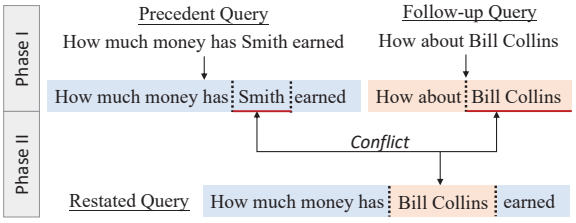


Figure 1: The two-phase process of an example from the FollowUp dataset (More real cases of diverse follow-up scenarios can be found in Table 3).

into several spans. For example, the precedent query is split into 3 spans: "How much money has", "Smith" and "earned". Let $q$ denote a kind of way to split $(\mathbf{x}, \mathbf{y})$, then Phase I can be formulated as $P_{\text{split}}(q|\mathbf{x}, \mathbf{y})$. Phase II is to **Recombine** the spans by finding out the most probable *conflicting way*, and generating the final output by restatement, denoted as $P_{\text{rec}}(\mathbf{z}|q)$. Two spans being conflicting means they are semantically similar. For example, "Smith" conflicts with "Bill Collins". A conflicting way contains all conflicts between the precedent and follow-up spans. Backed by the two-phase idea of splitting and recombination, the overall likelihood of generating $\mathbf{z}$ given $\mathbf{x}, \mathbf{y}$ is:

$$P_{\text{model}}(\mathbf{z}|\mathbf{x}, \mathbf{y}) = \sum_{q\in\mathcal{Q}} P_{\text{split}}(q|\mathbf{x}, \mathbf{y}) P_{\text{rec}}(\mathbf{z}|q), \quad (2)$$

where $\mathcal{Q}$ represents the set of all possible ways to split $(\mathbf{x}, \mathbf{y})$. Due to the lack of annotations for splitting and recombination, it is hard to directly perform supervised learning. Inspired by Liang et al. (2017), we employ RL to optimize $P_{\text{model}}$. Denoting the predicted restated query by $\tilde{\mathbf{z}}$, simplifying $\mathbb{E}_{(\mathbf{x},\mathbf{y},\mathbf{z})\sim\mathcal{D}}$ as $\mathbb{E}$, the goal of the RL training is to maximize following objective:

$$\mathcal{L}_{\text{rl}} = \mathbb{E}[\sum_{\tilde{\mathbf{z}}\in\mathcal{Z}}\sum_{q\in\mathcal{Q}} P_{\text{split}}(q|\mathbf{x}, \mathbf{y}) P_{\text{rec}}(\tilde{\mathbf{z}}|q) r(\mathbf{z}, \tilde{\mathbf{z}})], \quad (3)$$

where $\mathcal{Z}$ is the space of all restated query candidates and $r$ represents the reward defined by comparing $\tilde{\mathbf{z}}$ and the annotation $\mathbf{z}$. However, the overall candidate space $\mathcal{Q}\times\mathcal{Z}$ is vast, making it impossible to exactly maximize $\mathcal{L}_{\text{rl}}$. The most straightforward usage of the REINFORCE algorithm (Williams, 1992), sampling both $q$ and $\tilde{\mathbf{z}}$, also poses challenges for learning. To alleviate the problem, we propose to sample $q$ and enumerate all candidate $\tilde{\mathbf{z}}$ after $q$ is determined. It could shrink the sampling space with an acceptable computational cost, which will be discussed in Sec-

[1]Code is available at http://github.com/microsoft/EMNLP2019-Split-And-Recombine.

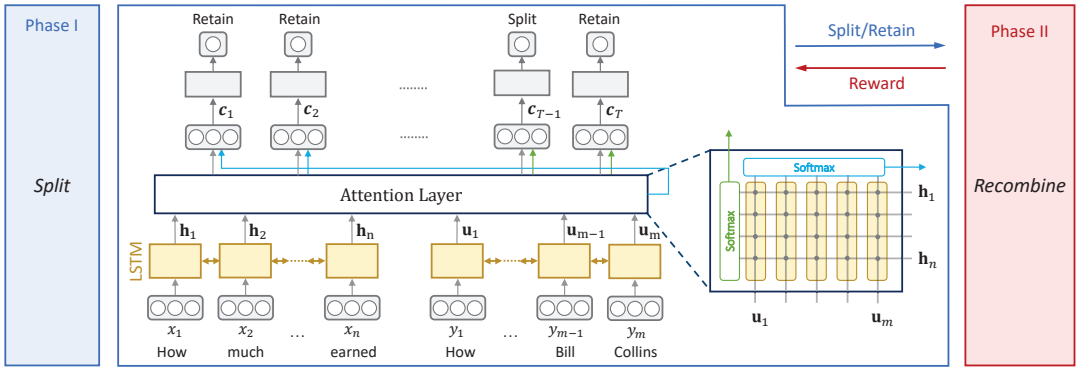


Figure 2: The overview of STAR with two phases.

tion 3.2.2. Thus the problem turns to design a reward function $R(q, \mathbf{z})$ to evaluate $q$ and guide the learning. To achieve it, we reformulate Equation 3 as:

$$\mathcal{L}_{\text{rl}} = \mathbb{E}[\sum_{q \in \mathcal{Q}} P_{\text{split}}(q|\mathbf{x}, \mathbf{y}) \sum_{\tilde{\mathbf{z}} \in \mathcal{Z}} P_{\text{rec}}(\tilde{\mathbf{z}}|q) r(\mathbf{z}, \tilde{\mathbf{z}})], \quad (4)$$

and set the $R(q, \mathbf{z})$ as:

$$\sum_{\tilde{\mathbf{z}} \in \mathcal{Z}} P_{\text{rec}}(\tilde{\mathbf{z}}|q) r(\mathbf{z}, \tilde{\mathbf{z}}). \quad (5)$$

The overview of STAR is summarized in Figure 2. Given $\mathbf{x}, \mathbf{y}$, during training of Phase I (in blue), we fix $P_{\text{rec}}$ to provide the reward $R(q, \mathbf{z})$, then $P_{\text{split}}$ can be learnt by the REINFORCE algorithm. During training of Phase II (in red), we fix $P_{\text{split}}$ and utilize it to generate $q$, $P_{\text{rec}}$ is trained to maximize Equation 5. In this way, $P_{\text{split}}$ and $P_{\text{rec}}$ can be jointly trained. The details are introduced below.

## 2.2 Phase I: Split

As mentioned above, fixed $P_{\text{rec}}$, Phase I updates $P_{\text{split}}$, the Split Neural Network (**SplitNet**). Taking the precedent query and follow-up query as input, as shown in Figure 2, splitting spans can be viewed as a sequence labeling problem over input. For each word, SplitNet outputs a label *Split* or *Retain*, indicating whether a split operation will be performed after the corresponding word. A label sequence uniquely identifies a way of splitting $(\mathbf{x}, \mathbf{y})$, mentioned as $q$ in Section 2.1. Figure 3 gives an example on the bottom. In the precedent query, two split operations are performed after "has" and "Smith" , since their labels are *Split*.

### 2.2.1 Split Neural Network

Intuitively, only after obtaining information from both the precedent query and follow-up query can SplitNet get to know the reasonable way to split spans. Inspired by BiDAF (Seo et al., 2017), we apply a bidirectional attention mechanism to capture the interrelations between the two queries.

**Embedding Layer** We consider embedding in three levels: character, word and sentence, respectively denoted as $\phi_c$, $\phi_w$ and $\phi_s$. Character-level embedding maps each word to a vector in a high-dimensional space using Convolutional Neural Networks (Kim, 2014). Word-level embedding is initialized using GloVe (Pennington et al., 2014), and then it is updated along with other parameters. Sentence-level embedding is a one-hot vector designed to distinguish between precedent and follow-up queries. Then, the overall embedding function is $\phi = [\phi_c; \phi_w; \phi_s]$.

**Context Layer** On top of the embedding layer, Bidirectional Long Short-Term Memory Network (BiLSTM) (Hochreiter and Schmidhuber, 1997; Schuster and Paliwal, 1997) is applied to capture contextual information within one query. For word $x_i (i = 1, \ldots, n)$ in the precedent query $\mathbf{x}$, the hidden state $\mathbf{h}_i = [\overrightarrow{\mathbf{h}}_i; \overleftarrow{\mathbf{h}}_i]$ is computed, where the forward hidden state is:

$$\overrightarrow{\mathbf{h}}_i = \overrightarrow{\text{LSTM}}(\phi(x_i); \overrightarrow{\mathbf{h}}_{i-1}). \quad (6)$$

Similarly, a hidden state $\mathbf{u}_j$ is computed for word $y_j (j = 1, \ldots, m)$. The BiLSTMs for $\mathbf{x}$ and $\mathbf{y}$ share the same parameters.

**Attention Layer** The interrelations between the precedent and follow-up queries are captured via attention layer. Let $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_n]$ and $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m]$ denote the hidden states of two queries respectively, the similarity matrix is:

$$\mathbf{A} = \cos(\mathbf{H}^\top \mathbf{U}), \quad (7)$$

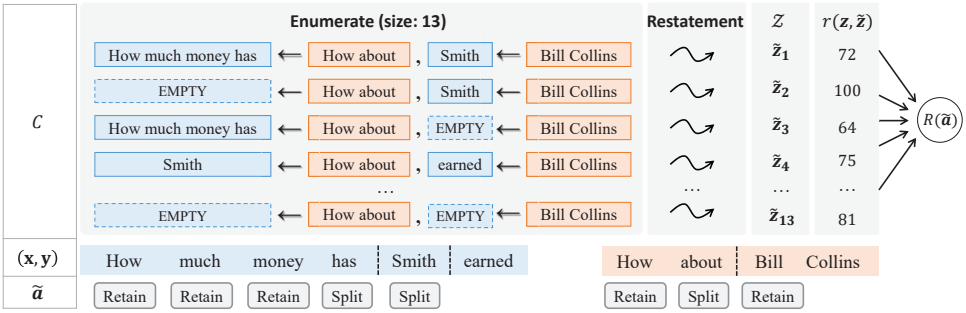


Figure 3: Illustration of reward computation in Phase II.

where $\mathbf{A} \in \mathbb{R}^{n \times m}$ and the entry $A_{i,j}$ represents the similarity between words $x_i$ and $y_j$. Then the softmax function is used to obtain the precedent-to-follow **(P2F)** attention and the follow-to-precedent **(F2P)** attention. P2F attention represents $y_j$ using the similarities between $y_j$ and every word in $\mathbf{x}$. Specifically, let $\mathbf{f}_j = \mathrm{softmax}(\mathbf{A}_{:,j})$, where $\mathbf{f}_j \in \mathbb{R}^n$ denotes the attention weights on $\mathbf{x}$ according to $y_j$. Then $y_j$ can be represented by a precedent-aware vector $\tilde{\mathbf{u}}_j = \sum_{k=1}^{n} \mathbf{f}_j[k] \cdot \mathbf{h}_k$. Similarly, F2P attention computes the attention weights on $\mathbf{y}$ according to $x_i$, and represents $x_i$ as $\tilde{\mathbf{h}}_i$.

**Output Layer** Combining the outputs of the context layer and the attention layer, we design the final hidden state as follows:

$$\mathbf{c}^{x_i} = [\mathbf{h}_i;\ \mathbf{h}_i \circ \tilde{\mathbf{h}}_i;\ \mathbf{h}_{i+1} \circ \tilde{\mathbf{h}}_{i+1}], \quad (8)$$

$$\mathbf{c}^{y_j} = [\mathbf{u}_j;\ \mathbf{u}_j \circ \tilde{\mathbf{u}}_j;\ \mathbf{u}_{j+1} \circ \tilde{\mathbf{u}}_{j+1}], \quad (9)$$

where $i \in \{1, \ldots, n-1\}, j \in \{1, \ldots, m-1\}$ and $\circ$ denotes element-wise multiplication (Lee et al., 2017). Let $\mathbf{c} = (\mathbf{c}_t)_{t=1}^{T} = (\mathbf{c}^{x_1}, ..., \mathbf{c}^{x_{n-1}}, \mathbf{c}^{y_1}, ..., \mathbf{c}^{y_{m-1}})$ denote the final hidden state sequence. At each position $t$, the probability of *Split* is $\sigma(\mathbf{W} * \mathbf{c}_t + b)$, where $\sigma$ denotes the *sigmoid* function and $\{\mathbf{W}, b\}$ denotes the parameters.

### 2.2.2 Training

It is difficult to train RL model from scratch. Therefore, we propose to initialize SplitNet via pre-training, and then use reward to optimize it.

**Pre-training** We obtain the pre-training annotation $\mathbf{a}$ by finding the common substrings between $(\mathbf{x}, \mathbf{y})$ and $\mathbf{z}$. One $\mathbf{a}$ is a label sequence, each of which is *Split* or *Retain*. Given the pre-training data set $\mathcal{D}_{\mathrm{pre}}$ whose training instance is as $(\mathbf{x}, \mathbf{y}, \mathbf{a})$, the objective function of pre-training is:

$$\mathcal{L}_{\mathrm{pre}}(\theta) = \mathbb{E}_{(\mathbf{x},\mathbf{y},\mathbf{a}) \sim \mathcal{D}_{\mathrm{pre}}} [\log p_\theta(\mathbf{a} | \mathbf{x}, \mathbf{y})], \quad (10)$$

where $\theta$ is the parameter of SplitNet.

**Policy Gradient** After pre-training, we treat the label sequence as a variable $\tilde{\mathbf{a}}$. The reward $R(\tilde{\mathbf{a}}, \mathbf{z})$ (details in Section 2.3) is used to optimize the parameter $\theta$ with policy gradient methods (Sutton et al., 1999). SplitNet is trained to maximize the following objective function:

$$\mathcal{L}_{\mathrm{rl}}(\theta) = \mathbb{E}_{(\mathbf{x},\mathbf{y},\mathbf{z}) \sim \mathcal{D}} [\mathbb{E}_{\tilde{\mathbf{a}} \sim p_\theta(\tilde{\mathbf{a}} | \mathbf{x}, \mathbf{y})} R(\tilde{\mathbf{a}}, \mathbf{z})]. \quad (11)$$

In practice, REINFORCE algorithm (Williams, 1992) is applied to approximate Equation 11 via sampling $\tilde{\mathbf{a}}$ from $p_\theta(\tilde{\mathbf{a}} | \mathbf{x}, \mathbf{y})$ for $M$ times, where $M$ is a hyper-parameter representing the sample size. Furthermore, subtracting a baseline (Weaver and Tao, 2001) on $R(\tilde{\mathbf{a}}, \mathbf{z})$ is also applied to reduce variance. The final objective function is as follows:

$$\mathcal{L}_{\mathrm{rl}}(\theta) = \mathbb{E}_{(\mathbf{x},\mathbf{y},\mathbf{z}) \sim \mathcal{D}} \Big[ \sum_{i=1}^{M} p_\theta(\tilde{\mathbf{a}}_i | \mathbf{x}, \mathbf{y}) \big( R(\tilde{\mathbf{a}}_i, \mathbf{z}) - \bar{R} \big) \Big],$$

$$\text{where } \bar{R} = \frac{1}{M} \sum_{i=1}^{M} R(\tilde{\mathbf{a}}_i, \mathbf{z}). \quad (12)$$

## 2.3 Phase II: Recombine

Here we present Phase II with two questions: (1) Receiving the sampled label sequence $\tilde{\mathbf{a}}$, how to compute its reward $R(\tilde{\mathbf{a}}, \mathbf{z})$; (2) How to do training and inference for $P_{\mathrm{rec}}$.

### 2.3.1 Reward Computation

Receiving the label sequence $\tilde{\mathbf{a}}$, we first enumerate all conflicting way candidates. Following the example in Figure 3, once we get a deterministic $\tilde{\mathbf{a}}$, the split of $(\mathbf{x}, \mathbf{y})$ is uniquely determined. Here $\mathbf{x}$ and $\mathbf{y}$ are split into 3 and 2 spans respectively. Treating spans as units, we enumerate all conflicting way candidates methodically. We act up to the one-to-one conflicting principle, which means a

span either has no conflict (denoted as *EMPTY*) or has only one conflict with a span in another query. Let $\mathcal{C}$ denote the set of all conflicting way candidates, the size of which is 13 in Figure 3.

For each conflicting way, we deterministically generate a restated query via the process named **Restatement**. In general, we simply replace spans in the precedent query with their conflicting spans to generate the restated query. For example, in Figure 3, the first one in $\mathcal{C}$ is restated as "How about Bill Collins earned". For spans in the follow-up query, if they contain column names or cell values and do not have any conflict, they are appended to the tail of the precedent query. It is designed to remedy the sub-query situation where there is no conflict (e.g. "Which opponent received over 537 attendance" and "And which got the result won 5-4"). Specially, if a span in the follow-up query contains a pronoun, we will in reverse replace it with its conflicting span to obtain the restated query.

Finally, the reward can be computed. Here we use BLEU and SymAcc[2] to build the reward function, expanding $r(\mathbf{z}, \tilde{\mathbf{z}})$ in Equation 5 as:

$$r(\mathbf{z}, \tilde{\mathbf{z}}) = \alpha \cdot \text{BLEU}(\mathbf{z}, \tilde{\mathbf{z}}) + \beta \cdot \text{SymAcc}(\mathbf{z}, \tilde{\mathbf{z}}), \quad (13)$$

where $\alpha, \beta > 0$ and $\alpha + \beta = 1$. The reward for $\tilde{\mathbf{a}}$ can be obtained using Equation 5.

#### 2.3.2 Training and Inference

Besides the reward computation, the recombination model $P_{\text{rec}}$ needs to be trained to maximize Equation 5. To achieve this, we define a conflicting probability matrix $\mathbf{F} \in \mathbb{R}^{N_x \times N_y}$, where $N_x$ and $N_y$ denote the number of spans in $\mathbf{x}$ and $\mathbf{y}$ respectively. The entry $\mathbf{F}_{u,v}$, the conflicting probability between the $u$-th span in $\mathbf{x}$ and the $v$-th span in $\mathbf{y}$, is obtained by normalizing the cosine similarity between their representations. Here the span representation is the subtraction representation (Wang and Chang, 2016; Cross and Huang, 2016), which means that span $(x_i, \ldots, x_k)$ is represented by $[\overrightarrow{\mathbf{h}_k} - \overrightarrow{\mathbf{h}_i}; \overleftarrow{\mathbf{h}_i} - \overleftarrow{\mathbf{h}_k}]$ from the same BiLSTM in the context layer in Section 2.2.1. Given a conflicting way denoted as $\tilde{c} \in \mathcal{C}$, the probability of generating its corresponding $\tilde{\mathbf{z}}$ can be written as the multiplication over $g(u, v)$:

$$P_{\text{rec}}(\tilde{\mathbf{z}}|\tilde{\mathbf{a}}) = P(\tilde{c}|\mathbf{F}) = \prod_{u=1}^{N_x} \prod_{v=1}^{N_y} g(u, v), \quad (14)$$

where $g(u, v) = \mathbf{F}_{u,v}$ if the $u$-th span in $\mathbf{x}$ conflicts with the $v$-th span in $\mathbf{y}$; otherwise, $g(u, v) = 1 - \mathbf{F}_{u,v}$. With the above formulation, we can maximize Equation 5 through automatic differentiation. To reduce the computation, we only maximize $P_{\text{rec}}(\tilde{\mathbf{z}}^*|\tilde{\mathbf{a}})$, the near-optimal solution to Equation 5, where $\tilde{\mathbf{z}}^* = \arg\max_{\tilde{\mathbf{z}} \in \mathcal{Z}}(r(\mathbf{z}, \tilde{\mathbf{z}}))$ denotes the best predicted restated query so far.

Guided by the golden restated query $\mathbf{z}$, in training, we find out $\tilde{\mathbf{z}}^*$ by computing the reward of each candidate. However in inference, where there is no golden restate query, we can only obtain $\tilde{\mathbf{z}}^*$ from $\mathbf{F}$. Specially, for the $v$-th span in the follow-up query, we find $u^* = \arg\max_u \mathbf{F}_{u,v}$. That means, compared to other spans in the precedent query, the $u^*$-th span has the highest probability to conflict with the $v$-th span in the follow-up query. Moreover, similar to Lee et al. (2017), if $\mathbf{F}_{u^*,v} < \lambda$, then the $v$-th span in the follow-up query has no conflict. The hyper-parameter $\lambda > 0$ denotes the threshold.

### 2.4 Extension

So far, we have introduced the whole process of STAR. Next we explore its extensibility. As observed, when the annotations are restated queries, STAR is parser-independent and can be incorporated into any context-independent semantic parser. But what if the annotations are answers to follow-up queries? Assuming we have an ideal semantic parser, a predicted restated query $\tilde{\mathbf{z}}$ can be converted into its corresponding answer $\tilde{w}$. For example, given $\tilde{\mathbf{z}}$ as "where are the players from", $\tilde{w}$ could be "Las Vegas". Therefore, revisiting Equation 3, in theory STAR is able to be extended by redesigning $r$ as $r(w, \tilde{w})$, where $w$ denotes the answer annotation. We conduct an extension experiment to verify it, as discussed in Section 3.3.

## 3 Experiments

In this section, we demonstrate the effectiveness of STAR on the FollowUp dataset[3] with restated query annotations, and its promising extensibility on the SQA dataset[4] with answer annotations.

### 3.1 Implementation details

We utilize PyTorch (Paszke et al., 2017) and AllenNLP (Gardner et al., 2018) for implementation, and adopt Adam (Kingma and Ba, 2015) as

[2]Their definitions along with the motivations of using them will be explained in Section 3.2.

[3]http://github.com/SivilTaram/FollowUp

[4]http://aka.ms/sqa

| Model | Dev | | Test | | |
|---|---|---|---|---|---|
| | SymAcc (%) | BLEU (%) | SymAcc (%) | BLEU (%) | AnsAcc (%) |
| SEQ2SEQ† (Bahdanau et al., 2015) | 0.63 ± 0.00 | 21.34 ± 1.14 | 0.50 ± 0.22 | 20.72 ± 1.31 | – |
| COPYNET† (Gu et al., 2016) | 17.50 ± 0.87 | 43.36 ± 0.54 | 19.30 ± 0.93 | 43.34 ± 0.45 | – |
| COPY+BERT (Devlin et al., 2019) | 18.63 ± 0.61 | 45.14 ± 0.68 | 22.00 ± 0.45 | 44.87 ± 0.52 | – |
| CONCAT† | – | – | 22.00 ± – | 52.02 ± – | 25.24 |
| E2ECR† (Lee et al., 2017) | – | – | 27.00 ± – | 52.47 ± – | 27.18 |
| FANDA† (Liu et al., 2019) | 49.00 ± 1.28 | 60.14 ± 0.98 | 47.80 ± 1.14 | 59.02 ± 0.54 | 60.19 |
| STAR | **55.38** ± 1.21 | **67.62** ± 0.65 | **54.00** ± 1.09 | **67.05** ± 1.05 | **65.05** |

Table 1: SymAcc, BLEU and AnsAcc on the FollowUp dataset. Results marked † are from Liu et al. (2019).

the optimizer. The dimensions of word embedding and hidden state are both 100. Variational dropout (Blum et al., 2015) is employed at embedding layer for better generalization ability (with probability 0.5). The learning rate is set to be 0.001 for pre-training, 0.0001 for RL training on FollowUp, and 0.0002 for SQA. In the implementation of the REINFORCE algorithm, we set $M$ to be 20. Finally, for hyper-parameters, we set $\alpha = 0.5$, $\beta = 0.5$ and $\lambda = 0.6$. All the results are averaged over 5 runs with random initialization.

### 3.2 Results on FollowUp dataset

The FollowUp dataset contains 1000 natural language query triples $(\mathbf{x}, \mathbf{y}, \mathbf{z})$. Each triple belongs to a single database table, and there are 120 tables in several different domains. Following the previous work, we split them into the sets of size 640/160/200 for train/dev/test. We evaluate the methods using both answer level and query level metrics. *AnsAcc* is to check the answer accuracy of predicted queries manually. Concretely, 103 golden restated queries can be successfully parsed by COARSE2FINE (Dong and Lapata, 2018). We parse their corresponding predicted queries into SQL using COARSE2FINE and manually check the answers. Although AnsAcc is most convincing, it cannot cover the entire test set. Therefore, we apply two query level metrics: *SymAcc* detects whether all the SQL-related words are correctly involved in the predicted queries, for example column names, cell values and so on. It reflects the approximate upper bound of AnsAcc, as the correctness of SQL-related words is a prerequisite of correct execution in most cases; *BLEU*, referring to the cumulative 4-gram BLEU score, evaluates how similar the predicted queries are to the golden ones (Papineni et al., 2002). SymAcc focuses on limited keywords, so we introduce BLEU to evaluate quality of the entire predicted query.

#### 3.2.1 Model Comparison

Our baselines fall into two categories. **Generation-based methods** conform to the architecture of sequence-to-sequence (Sutskever et al., 2014) and generate restated queries by decoding each word from scratch. SEQ2SEQ (Bahdanau et al., 2015) is the sequence-to-sequence model with attention, and COPYNET further incorporates a copy mechanism. COPY+BERT incorporates the latest pre-trained BERT model (Devlin et al., 2019) as the encoder of COPYNET. **Rewriting-based methods** obtain restated queries by rewriting precedent and follow-up queries. CONCAT directly concatenates the two queries. E2ECR (Lee et al., 2017) obtain restated queries by performing coreference resolution in follow-up queries. FANDA (Liu et al., 2019) utilizes a structure-aware model to merge the two queries. Our method STAR also belongs to this category.

**Answer Level** Table 1 shows AnsAcc results of competitive baselines on the test set. Compared with them, STAR achieves the highest, 65.05%, which demonstrates its superiority. Meanwhile, it verifies the feasibility of follow-up query analysis in cooperating with context-independent semantic parsing. Compared with CONCAT, our approach boosts over 39.81% on COARSE2FINE for the capability of context-dependent semantic parsing.

**Query Level** Table 1 also shows SymAcc and BLEU of different methods on the dev and test sets. As observed, STAR significantly outperforms all baselines, demonstrating its effectiveness. For example, STAR achieves an absolute improvement of 8.03% BLEU over the state-of-the-art baseline FANDA on testing. Moreover, the rewriting-based baselines, even the simplest CONCAT, perform better than the generation-based

| Variant | SymAcc (%) | BLEU (%) |
|---|---|---|
| STAR | 55.38 | 67.62 |
| – Phase I | 40.63 | 61.82 |
| – Phase II | 23.12 | 48.65 |
| – RL | 41.25 | 60.19 |
| + Basic Reward | 43.13 | 58.48 |
| + Oracle Reward | 45.20 | 63.04 |
| + Uniform Reward | 53.40 | 66.93 |

Table 2: Variant results on FollowUp dev set.

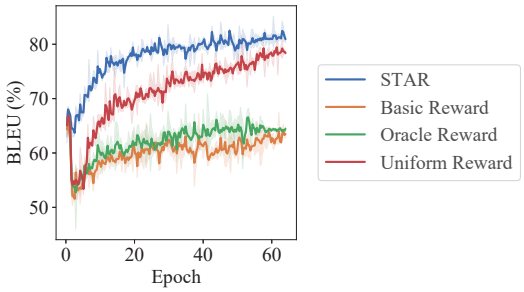


Figure 4: Learning curve on FollowUp train set.

ones. It suggests that the idea of rewriting is more reasonable for the task, where precedent and follow-up queries are of full utilization.

### 3.2.2 Variant Analysis

Besides baselines, we also conduct experiments with several variants of STAR to further validate the design of our model. As shown in Table 2, there are three variants with ablation: "– Phase I" takes out SplitNet and performs Phase II on word level; "– Phase II" performs random guess in the recombination process for testing; and "– RL" only contains pre-training. The SymAcc drops from about 55% to 40% by ablating Phase I, and to 23% by ablating Phase II. Their poor performances indicate both of the two phases are indispensable. "– RL" also performs worse, which again demonstrates the rationality of applying RL.

Three more variants are presented with different designs of $R(q, \mathbf{z})$ to prove the efficiency and effectiveness of Equation 5 as a reward. "+ Basic Reward" represents the most straightforward REINFORCE algorithm, which samples both $q \in \mathcal{Q}$ and $\tilde{\mathbf{z}} \in \mathcal{Z}$, then takes $r(\mathbf{z}, \tilde{\mathbf{z}})$ as $R(q, \mathbf{z})$. "+ Oracle Reward" assumes the conflicts are always correct and rewrites $R(q, \mathbf{z})$ as $\max_{\tilde{\mathbf{z}} \in \mathcal{Z}}(r(\mathbf{z}, \tilde{\mathbf{z}}))$. "+ Uniform Reward" assigns the same probability to all $\tilde{\mathbf{z}}$ and obtains $R(q, \mathbf{z})$ as $\text{mean}(\sum_{\tilde{\mathbf{z}} \in \mathcal{Z}} r(\mathbf{z}, \tilde{\mathbf{z}}))$. As shown in Table 2 and Figure 4, STAR learns better and faster than the variants due to the reasonable reward design. In fact, as mentioned in Section 2.1, the vast action space of the most straightforward REINFORCE algorithm leads to poor learning. STAR shrinks the space from $|\mathcal{Q}| \cdot |\mathcal{Z}|$ down to $|\mathcal{Q}|$ by enumerating $\tilde{\mathbf{z}}$. Meanwhile, statistics show that STAR obtains a $15\times$ speedup over "+ Basic Reward" on the convergence time.

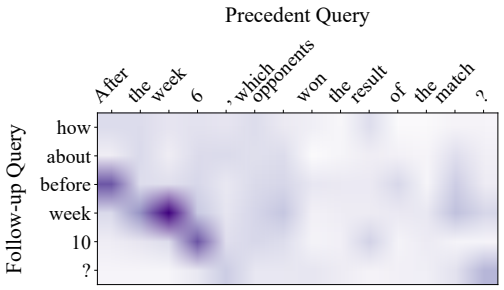


Figure 5: An example of similarity matrix in SplitNet.

### 3.2.3 Case Study

Figure 5 shows a concrete example of the similarity matrix $\mathbf{A}$ on attention layer of SplitNet. The span "before week 10" is evidently more similar to "After the week 6" than to others, which meets our expectations. Moreover, the results of three real cases are shown in Table 3. The spans in blue are those have conflicts, and the histograms represent the conflict probabilities to all the spans in precedent queries. In Case 1, "glebe park", "hampden park" and "balmoor" are all cell values in the database table with similar meanings. STAR correctly finds out the conflict between "compared to glebe park" and "compared to balmoor" with the highest probability. Case 2 shows STAR can discover the interrelation of words, where "the writer Nancy miller" is learnt as a whole span to replace "Nancy miller" in the precedent query. As for Case 3, STAR successfully performs coreference resolution and interprets "those two films" as "greatest love and promised land". Benefiting from two phases, STAR is able to deal with diverse follow-up scenarios in different domains.

### 3.2.4 Error Analysis

Our approach works well in most cases except for few ones, where SplitNet fails. For example, given the precedent query "what's the biggest zone?" and the follow-up query "the smallest one", STAR prefers to recognize "the biggest zone" and "the smallest one" as two spans, rather

| No | Case Analysis | |
|---|---|---|
| 1 | *Precedent* | : [ compared to glebe park ] [ , does ] [ hampden park ] [ holds more attendances at capacity ? ] |
| | *Follow-up* | : [ how about ] [ compared to balmoor ] |
| | STAR | : compared to balmoor , does hampden park holds more attendances at capacity ? |
| 2 | *Precedent* | : [ Is there any book which belongs to ] [ Nancy miller ] |
| | *Follow-up* | : [ I mean ] [ the writer Nancy miller ] |
| | STAR | : Is there any book which belongs to the writer Nancy miller |
| 3 | *Precedent* | : [ show directors of ] [ greatest love and promised land ] |
| | *Follow-up* | : [ show air date of ] [ those two films ] |
| | STAR | : show air date of greatest love and promised land |

Table 3: Case analysis of STAR on FollowUp dataset. Square brackets denote different spans.

| **Model** | Precedent | Follow-up |
|---|---|---|
| DynSP (Iyyer et al., 2017) | 70.9 | 35.8 |
| NP (Neelakantan et al., 2016) | 58.9 | 35.9 |
| NP + STAR | 58.9 | 38.1 |
| DynSP + STAR | 70.9 | 39.5 |
| DynSP* (Iyyer et al., 2017) | 70.4 | 41.1 |

Table 4: Answer accuracy on SQA test set.

than perform split operations inside them. The SplitNet fails probably because the conflicting spans, "the biggest" ↔ "the smallest" and "zone" ↔ "one", are adjacent, which makes it difficult to identify span boundaries well.

### 3.3 Extension on SQA dataset

Finally, we demonstrate STAR's extensibility in working with different annotations. As mentioned in Section 2.4, by designing $r(w, \tilde{w})$, STAR can cooperate with the answer annotations. We conduct experiments on the SQA dataset, which consists of 6066 query sequences (5042/1024 for train/test). Each sequence contains multiple natural language queries and their answers, where we are only interested in the first query and the immediate follow-up one. As discussed in (Iyyer et al., 2017), every answer can be represented as a set of cells in the tables, each of which is a multi-word value, and the intentions of the follow-up queries mainly fall into three categories. *Column selection* means the follow-up answer is an entire column; *Subset selection* means the follow-up answer is a subset of the precedent answer; and *Row selection* means the follow-up answer has the same rows with the precedent answer.

We employ two context-independent parsers, DynSP (Iyyer et al., 2017) and NP (Neelakantan et al., 2016), which are trained on the SQA dataset to provide relatively reliable answers for reward computing. Unfortunately, they both perform poorly for the restated queries, as the restated queries are quite different from the original queries in SQA. To address the problem, we redesign the recombination process. Instead of generating the restated query, we recombine the predicted precedent answer $\tilde{w}_x$ and the predicted follow-up answer $\tilde{w}_y$ to produce the restated answer $\tilde{w}$. Therefore, the objective of Phase II is to assign an appropriate intention to each follow-up span via an additional classifier. The goal of Phase I turns to split out spans having obvious intentions such as "of those". The way of recombining answer is determined by the voting from intentions on all spans. If the intention column selection wins, then $\tilde{w} = \tilde{w}_y$; for subset selection, we obtain the subset $\tilde{w}$ by taking the rows of $\tilde{w}_y$ as the constraint and applying it to $\tilde{w}_x$; and for row selection, we take the rows of $\tilde{w}_x$ and the columns of $\tilde{w}_y$ as the constraints, then apply them to the whole database table to obtain the answer $\tilde{w}$ retrieved by the predicted SQL. The reward $r(w, \tilde{w})$ is computed based on Jaccard similarity between the gold answer $w$ and $\tilde{w}$ as in (Iyyer et al., 2017), and the overall training process remains unchanged.

Table 4 shows the answer accuracy of precedent and follow-up queries on test set. DynSP* (Iyyer et al., 2017) is designed for SQA by introducing a special action *Subsequent* to handle follow-up queries based on DynSP. DynSP* is incapable of being extended to work with the annotation of the restated queries. We attempt to apply DynSP* (trained on SQA) directly on FollowUp test set, which results in an extremely low AnsAcc. On the contrary, STAR is extensible. "+STAR" means our method STAR is incorporated into the context-independent parser and empowers them with the ability to perform follow-up query analysis. As observed, integrating STAR consistently improves

performance for follow-up queries, which demonstrates the effectiveness of STAR in collaborating with different semantic parsers. The comparable results of DynSP+STAR to DynSP* further verifies the promising extensibility of STAR.

## 4 Related Work

Our work is closely related to two lines of work: context-dependent sentence analysis and reinforcement learning. From the perspective of context-dependent sentence analysis, our work is related to researches like reading comprehension in dialogue (Reddy et al., 2019; Choi et al., 2018), dialogue state tracking (Williams et al., 2013), conversational question answering in knowledge base (Saha et al., 2018; Guo et al., 2018), context-dependent logic forms (Long et al., 2016), and non-sentential utterance resolution in open-domain question answering (Raghu et al., 2015; Kumar and Joshi, 2017). The main difference is that we focus on the context-dependent queries in NLIDB which contain complex scenarios. As for the most related context-dependent semantic parsing, Zettlemoyer and Collins (2009) proposes a context-independent CCG parser and then conduct context-dependent substitution, Iyyer et al. (2017) presents a search-based method for sequential questions, and Suhr et al. (2018) presents a sequence-to-sequence model to solve the problem. Compared to their methods, our work achieves context-dependent semantic parsing via learnable restated queries and existing context-independent semantic parsers.

Moreover, the technique of reinforcement learning has also been successfully applied to natural language tasks in dialogue, such as hyper-parameters tuning for coreference resolution (Clark and Manning, 2016), sequential question answering (Iyyer et al., 2017) and coherent dialogue responses generation (Li et al., 2016). In this paper, we employ reinforcement learning to capture the structures of queries, which is similar to Zhang et al. (2018) for text classification.

## 5 Conclusion and Future Work

We present a novel method, named Split-And-Recombine (STAR), to perform follow-up query analysis. A two-phase process has been designed: one for splitting precedent and follow-up queries into spans, and the other for recombining them. Experiments on two different datasets demonstrate the effectiveness and extensibility of our method. For future work, we may extend our method to other natural language tasks.

## Acknowledgments

We thank all the anonymous reviewers for their valuable comments. This work was supported by the National Natural Science Foundation of China (Grant Nos. U1736217 and 61932003).

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015*.

Avrim Blum, Nika Haghtalab, and Ariel D. Procaccia. 2015. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems, NIPS 2015, Montreal, Quebec, Canada, December 7-12, 2015*, pages 2575–2583.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wentau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. QuAC: Question answering in context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018, Brussels, Belgium, October 31 - November 4, 2018*, pages 2174–2184.

Kevin Clark and Christopher D. Manning. 2016. Deep reinforcement learning for mention-ranking coreference models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2256–2262.

James Cross and Liang Huang. 2016. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1–11.

Deborah A. Dahl, Madeleine Bates, Michael Brown, William M. Fisher, Kate Hunicke-Smith, David S. Pallett, Christine Pao, Alexander I. Rudnicky, and Elizabeth Shriberg. 1994. Expanding the scope of the ATIS task: The ATIS-3 corpus. In *Proceedings of the 1994 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 1994, ,Plainsboro, New Jerey, USA, March 8-11, 1994*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1:Long and Short Papers*, pages 4171–4186.

Li Dong and Mirella Lapata. 2018. Coarse-to-Fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 731–742.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software, NLP-OSS, of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018*, pages 1–6.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, Berlin, Germany, August 7-12, 2016, Volume 1: Long Papers*.

Daya Guo, Duyu Tang, Nan Duan, Ming Zhou, and Jian Yin. 2018. Dialog-to-action: Conversational question answering over a large-scale knowledge base. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems, NeurIPS 2018, Montréal, Canada, December 3-8, 2018*, pages 2946–2955.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation, Volume 9*, pages 1735–1780.

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1821–1831.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, Doha, Qatar, October 25-29, 2014*, pages 1746–1751.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015*.

Vineet Kumar and Sachindra Joshi. 2017. Incomplete follow-up question resolution using retrieval based sequence to sequence learning. In *Proceedings of the 40th International ACM Conference on Research and Development in Information Retrieval, SIGIR 2017, Shinjuku, Tokyo, Japan, August 7-11, 2017*, pages 705–714.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 188–197.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1192–1202.

Chen Liang, Jonathan Berant, Quoc V. Le, Kenneth D. Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30-August 4, 2017, Volume 1: Long Papers*, pages 23–33.

Qian Liu, Bei Chen, Jian-Guang Lou, Ge Jin, and Dongmei Zhang. 2019. FANDA: A novel approach to perform follow-up query analysis. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence, AAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 6770–6777.

Reginald Long, Panupong Pasupat, and Percy Liang. 2016. Simpler context-dependent logical forms via model projections. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, Berlin, Germany, August 7-12, 2016, Volume 1: Long Papers*.

Scott Miller, David Stallard, Robert J. Bobrow, and Richard M. Schwartz. 1996. A fully statistical approach to natural language interfaces. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics, ACL 1996, University of California, Santa Cruz, California, June 24-27, 1996*, pages 55–61.

Arvind Neelakantan, Quoc V. Le, and Ilya Sutskever. 2016. Neural programmer: Inducing latent programs with gradient descent. In *Proceedings of the 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of*

*the 40th Annual Meeting of the Association for Computational Linguistics, ACL 2002, Philadelphia, PA, USA., July 6-12, 2002*, pages 311–318.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems, NIPS 2017, Long Beach, CA, USA, December 4-9, 2017*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, Doha, Qatar, October 25-29, 2014*, pages 1532–1543.

Dinesh Raghu, Sathish Indurthi, Jitendra Ajmera, and Sachindra Joshi. 2015. A statistical approach for non-sentential utterance resolution for interactive QA system. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue, SIGDIAL 2015, Prague, Czech Republic, September 2-4, 2015*, pages 335–343.

Siva Reddy, Danqi Chen, and Christopher D. Manning. 2019. CoQA: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics, Volume 7*, pages 249–266.

Amrita Saha, Vardaan Pahuja, Mitesh M. Khapra, Karthik Sankaranarayanan, and Sarath Chandar. 2018. Complex sequential question answering: Towards learning to converse over linked question answer pairs with a knowledge graph. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence, AAAI 2018, New Orleans, Louisiana, USA, February 2-7, 2018*, pages 705–713.

Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Trans. Signal Processing, Volume 45*, pages 2673–2681.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *Proceedings of the 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017*.

Alane Suhr, Srinivasan Iyer, and Yoav Artzi. 2018. Learning to map context-dependent sentences to executable formal queries. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1:Long Papers*, pages 2238–2249.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems, NIPS 2014, Montreal, Quebec, Canada, December 8-13, 2014*, pages 3104–3112.

Richard S. Sutton and Andrew G. Barto. 1998. Reinforcement learning: An introduction. *IEEE Trans. Neural Networks, Volume 9*, pages 1054–1054.

Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems 12: Annual Conference on Neural Information Processing Systems, NIPS 1999, Denver Colorado, USA, November 29 - December 4, 1999*, pages 1057–1063.

Wenhui Wang and Baobao Chang. 2016. Graph-based dependency parsing with bidirectional LSTM. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, Berlin, Germany, August 7-12, 2016, Volume 1: Long Papers*.

Lex Weaver and Nigel Tao. 2001. The optimal reward baseline for gradient-based reinforcement learning. In *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence, UAI 2001, Seattle, Washington, USA, August 2-5, 2001*, pages 538–545.

Jason D. Williams, Antoine Raux, Deepak Ramachandran, and Alan W. Black. 2013. The dialog state tracking challenge. In *Proceedings of the 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue, SIGDIAL 2013, Metz, France, August 22-24, 2013*, pages 404–413.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning, Volume 8*, pages 229–256.

Luke S. Zettlemoyer and Michael Collins. 2009. Learning context-dependent mappings from sentences to logical form. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics, ACL 2009, and the 4th International Joint Conference on Natural Language, IJCNLP 2009, Singapore, August 2-7, 2009*, pages 976–984.

Tianyang Zhang, Minlie Huang, and Li Zhao. 2018. Learning structured representation for text classification via reinforcement learning. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence, AAAI-18, New Orleans, Louisiana, USA, February 2-7, 2018*, pages 6053–6060.

# Text2Math: End-to-end Parsing Text into Math Expressions

**Yanyan Zou** and **Wei Lu**
StatNLP Research Group
Singapore University of Technology and Design
yanyan_zou@mymail.sutd.edu.sg, luwei@sutd.edu.sg

## Abstract

We propose Text2Math, a model for semantically parsing text into math expressions. The model can be used to solve different math related problems including arithmetic word problems (Roy and Roth, 2017; Liang et al., 2018) and equation parsing problems (Roy et al., 2016). Unlike previous approaches, we tackle the problem from an end-to-end structured prediction perspective where our algorithm aims to predict the complete math expression at once as a tree structure, where minimal manual efforts are involved in the process. Empirical results on benchmark datasets demonstrate the efficacy of our approach.

## 1 Introduction

Designing computer algorithms that can automatically solve math word problems is a challenge for the AI research community (Bobrow, 1964). Two representative tasks have been proposed and studied recently – solving arithmetic word problems (Wang et al., 2017; Roy and Roth, 2018; Zou and Lu, 2019b) and equation parsing (Roy et al., 2016), as illustrated in Figure 1. The former task focuses on mapping the input paragraph (which may involve multiple sentences) into a target math expression, from which an answer can be calculated. The latter task focuses on mapping a description (usually a single sentence) into a math equation that typically involves one or more unknowns. As we can observe from Figure 1, in both cases, the output can be represented as a tree structure.

Earlier approaches to solving arithmetic word problems focused on rule-based methods where hand-crafted rules have been used (Mukherjee and Garain, 2008; Hosseini et al., 2014). Recently, learning-based approaches based on statistical classifiers (Kushman et al., 2014; Roy and Roth, 2015; Roy et al., 2016; Liang et al., 2018) or

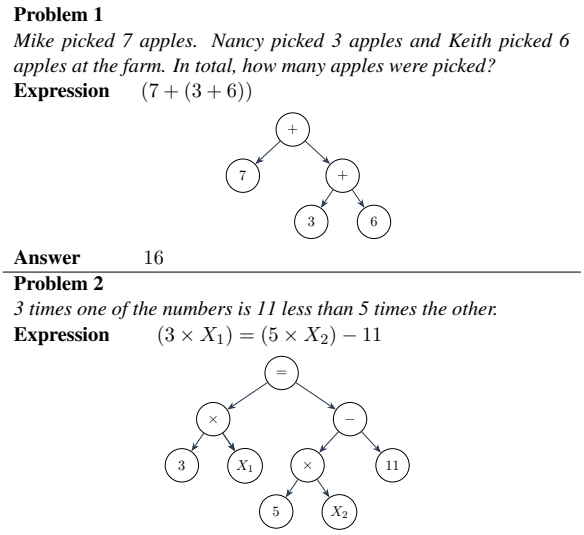


Figure 1: An example arithmetic word problem (top) and an example equation parsing problem (bottom) where the outputs can be represented as trees.

neural networks (Wang et al., 2017, 2018b) have been used for making decisions in the expression[1] construction process. However, these models do not focus on predicting the target tree as a complete structure at once, but locally trained classifiers are often used and local decisions are then combined. Such local classifiers often make predictions on the choice of the underlying operator between two operands (e.g., numbers) appearing in the text in a particular order. As a result, special treatments of the non-commutative operators such as *subtraction* ($-$) and *division* ($\div$) are often involved, where the introduction of inverse operators is typically required[2]. Shi et al. (2015) tackled the problem from a structured prediction perspective, where a semantic parsing algorithm us-

[1] In this work, we use the term *expression* to refer to a math expression (for arithmetic word problem) or an equation (for equation parsing).

[2] For example, the inverse operator $-_i$ applied to two operands $a$ followed by $b$ is used to denote $b - a$.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5327–5337, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

ing context-free grammars (CFG) was used. However, their approach relies on semi-automatically generated rules and involves a manual step for interpreting the semantic representation they used.

While all these approaches focused on solving arithmetic word problems only, separate models have been developed for the task of equation parsing (Roy et al., 2016). It is not clear how easy each of these models specifically designed for one task can be adapted for the other task. Motivated by the observation that both problems involve mapping a text sequence to a tree structured representation, we propose `Text2Math` which regards both tasks as a class of structured prediction problems, and tackle them from a semantic parsing perspective. We make use of an end-to-end latent-variable approach to automatically produce the target math expression at once as a complete structure, where no prior knowledge on the operators (such as whether an operator is non-commutative) is required. Our model outperforms all baselines on two benchmark datasets. To the best of our knowledge, this is the first approach based on semantic parsing that tackles both arithmetic word problems and equation parsing with a single model. Our code is available at `http://statnlp.org/research/ta`.

## 2 Approach

### 2.1 Expression Tree

We first define tree representations for math expressions, which will then be regarded as the semantic representations used in the standard semantic parsing setup.

The nodes involved in the math expression trees can be classified into two categories, namely, *operator* and *quantity* nodes. Specifically, operator nodes are the tree nodes that define the types of operations involved in expressions. In this work we consider ADD (*addition*, $+$), SUB (*subtraction*, $-$), MUL (*multiplication*, $\times$) and DIV (*division*, $\div$). We also regard the equation sign ($=$) as an operation involved in math expressions and use EQU to denote it. We consider two types of quantity nodes: CON denoting constants, and VAR for unknown variables. Table 1 lists the above nodes. Each tree node comes with an *arity* which specifies the number of direct child nodes that should appear below the given node. For example, the operator node SUB with arity 2 is expecting two child nodes below it in the expression tree, while CON with arity 0 is supposed to be a leaf node. The two math expressions in Figure 1 can be equivalently represented by expression trees consisting of such nodes, as illustrated in Figure 2.

| Category | Node | Interpretation | Arity |
|---|---|---|---|
| Operator | EQU | $n_i$ EQU $n_j \Leftrightarrow (n_i = n_j)$ | 2 |
| | ADD | $n_i$ ADD $n_j \Leftrightarrow (n_i + n_j)$ | 2 |
| | SUB | $n_i$ SUB $n_j \Leftrightarrow (n_i - n_j)$ | 2 |
| | MUL | $n_i$ MUL $n_j \Leftrightarrow (n_i \times n_j)$ | 2 |
| | DIV | $n_i$ DIV $n_j \Leftrightarrow (n_i \div n_j)$ | 2 |
| Quantity | CON | A constant | 0 |
| | VAR | A variable | 0 |

Table 1: Expression tree nodes with interpretations, where $n_i$ ($n_j$) refers to the first (second) operand.

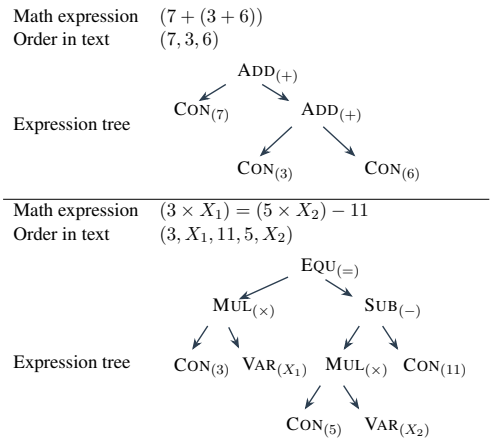


Figure 2: Expression trees for the two math expressions in Figure 1. "Order in text" refers to the order that the textual expressions of operands appear in the problem text. We use subscripts to indicate the actual semantic interpretations.

### 2.2 Latent Text-Math Tree

With the specifically designed expression trees for representing the math expressions, we will now be able to design a model for parsing the text into the expression tree. This is essentially a semantic parsing task. One of the key assumptions made by the various semantic parsing algorithms is the intermediate joint representation used for connecting the words and semantics (Wong and Mooney, 2006; Zettlemoyer and Collins, 2007; Lu et al., 2008; Artzi and Zettlemoyer, 2013b). In this work, we adopt an approach that is inspired by (Lu et al., 2008; Lu, 2014), which learns a latent joint representation for words and semantics in the form of *hybrid trees* where word-semantics correspondence information is captured. Specifically, we introduce a *text-math tree* representation that jointly encodes both text and the math expression tree.

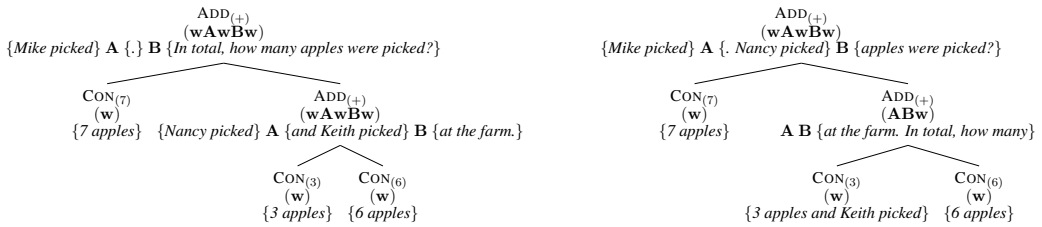


Figure 3: Example text-math trees for the arithmetic word problem example in Figure 1. The left tree captures the semantic correspondence well, while the right tree fails to capture the correct correspondence.

Such joint representations can be understood as a modified expression tree where each semantic node is now augmented with additional word information from the corresponding text.

From the joint representations we would be able to recover the semantic level correspondence information between words and math expressions. Possible joint representations of the two examples in Figure 1 are illustrated in Figure 3 and Figure 4 (left), respectively. Consider Problem 1 in Figure 1. We illustrate two possible text-math expression trees in Figure 3. Here each node in such joint representations is essentially a node in the original expression trees augmented with words from the problem text. For example, consider the left tree in Figure 3, the root node is an operator (the root node of the original expression tree) paired with discontiguous sequence of words "{*Mike picked*}...{.}...{*In total, how many apples were picked?*}" that appear in the problem text. This way, such a text-math tree is able to capture the semantic correspondence between words and basic units involved in the math expressions (i.e., operators, quantities). However, the text-math trees are not explicitly given during the training phase. For example, the right side of Figure 3 gives an alternative text-math tree that can also serve as a joint representation of both the text and the expression tree. Comparing both trees we may see the one on the left appears to be better at capturing the true semantic level correspondence between words and math expressions. Since there is no gold text-math tree explicitly given, we model it with a latent-variable approach.

Formally, given a text $\boldsymbol{x}$, paired with the expression $\boldsymbol{y}$ (or equivalently, the expression tree), we assume there exists a latent joint text-math representation in the form of text-math tree, that comprises exactly $\boldsymbol{x}$ and $\boldsymbol{y}$, denoted as $\boldsymbol{t}$. Each node is a word-semantics association $\langle x, y, p\rangle$ where $x$ is a (possibly discontiguous) word sequence of $\boldsymbol{x}$ and $y$ is an individual expression tree node from $\boldsymbol{y}$, and $p$ is the word association pattern that is used to specify how words interact with the expression tree (further details will be provided in Sec. 2.3). Intuitively, such a joint text-math representation should precisely contain the exact information associated with the text and its corresponding math expression and nothing else. We will defer the discussion on how to exactly construct such joint representations until Sec. 2.3.

The training corpus provides both the problem text $\boldsymbol{x}$ and its math expression, which we represent with an expression tree $\boldsymbol{y}$. The joint representation $\boldsymbol{t}$ is not available in the training data, which we model as a latent variable. The conditional random fields (CRF) (Lafferty et al., 2001) has been successfully applied to many tasks in the NLP community (Lample et al., 2016; Zou and Lu, 2018, 2019a). In this work, we also apply CRF to model the conditional probability of the latent variable $\boldsymbol{t}$ and output expression $\boldsymbol{y}$, conditioned on the input $\boldsymbol{x}$. The objective is defined as follows:

$$P_{\Lambda,\Theta}(\boldsymbol{y}|\boldsymbol{x}) = \sum_{\boldsymbol{t}\in\mathcal{T}(\boldsymbol{x},\boldsymbol{y})} P_{\Lambda,\Theta}(\boldsymbol{y},\boldsymbol{t}|\boldsymbol{x})$$
$$= \frac{\sum_{\boldsymbol{t}\in\mathcal{T}(\boldsymbol{x},\boldsymbol{y})} e^{[\Lambda\cdot\Phi(\boldsymbol{x},\boldsymbol{y},\boldsymbol{t})+G_\Theta(\boldsymbol{x},\boldsymbol{y},\boldsymbol{t})]}}{\sum_{\boldsymbol{y}',\boldsymbol{t}'\in\mathcal{T}(\boldsymbol{x},\boldsymbol{y}')} e^{[\Lambda\cdot\Phi(\boldsymbol{x},\boldsymbol{y}',\boldsymbol{t}')+G_\Theta(\boldsymbol{x},\boldsymbol{y}',\boldsymbol{t}')]}} \quad (1)$$

where $\Phi(\boldsymbol{x},\boldsymbol{y},\boldsymbol{t})$ returns a list of discrete features defined over the tuple $(\boldsymbol{x},\boldsymbol{y},\boldsymbol{t})$, $\Lambda$ is the feature weight vector, $G_\Theta$ is a neural scoring function parameterized by $\Theta$ and $\mathcal{T}(\boldsymbol{x},\boldsymbol{y})$ is a set of possible joint representations (i.e., text-math trees) for the pair $(\boldsymbol{x},\boldsymbol{y})$.

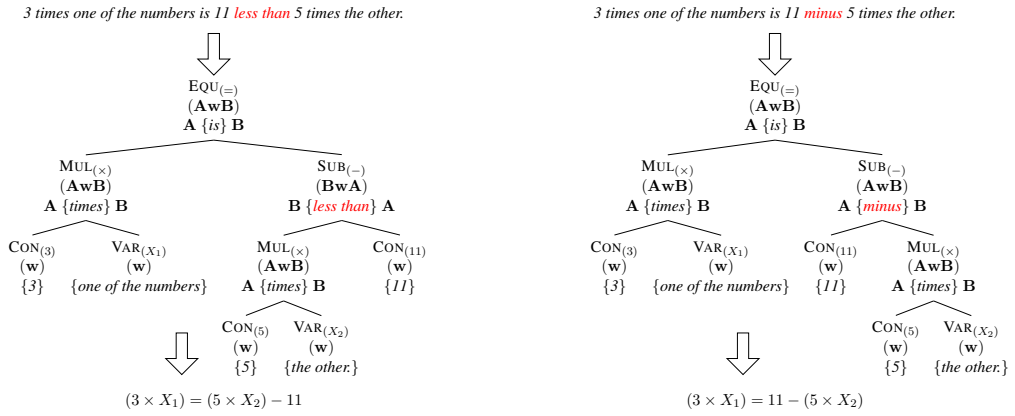


Figure 4: Left: example text-math tree for the equation parsing example in Figure 1, where the word association pattern **BwA** in the example is used for modeling reordering. Right: another example for a slightly different instance where the reordering is not required.

### 2.3 Inference

One challenge associated with the inference procedure in both training and decoding is how to handle the large space of latent structures defined by $\mathcal{T}(\boldsymbol{x}, \boldsymbol{y})$ and $\mathcal{T}(\boldsymbol{x})$. Without any constraints, searching or calculation that involves all possible structures within this space may be intractable. We therefore introduce some assumptions on the set of allowable structures, such that tractable inference can be applied to such structures.

We first introduce three symbols **A**, **B** and **w**. The symbol **A** refers to a placeholder for the left sub-tree (rooted by the left child node), and similarly **B** is a placeholder for the right sub-tree. The symbol **w** refers to a contiguous sequence of (1 or more) words. We will then use these three symbols to define the set of *word association patterns*, which are used to specify how the words interact with the sub-trees of the current node. Specifically, for expression tree nodes with arity 0 (i.e., quantity nodes), only one pattern **w** is allowed to be attached to them, indicating that a contiguous word sequence from a given problem text is associated with such expression tree nodes. As for expression tree nodes with arity 2 (i.e., operator nodes), we define 16 allowable patterns denoted as $\{[\mathbf{w}]\mathbf{A}[\mathbf{w}]\mathbf{B}[\mathbf{w}], [\mathbf{w}]\mathbf{B}[\mathbf{w}]\mathbf{A}[\mathbf{w}]\}$, where [] denotes optional. Based on such word association patterns, we will be able to define a set of possible text-math trees for a particular text-expression pair that we regard as valid.

Before we formally define what is a valid text-math tree, let us look at an example in Figure 3 which shows two valid trees. First of all, we can verify that, if we exclude the words from both trees, we arrive at the math expression that corresponds to the text. Second, we can also recover the text information from such a joint representation. Let us look at the right tree in Figure 3. Consider the right sub-tree of the complete tree rooted by the node $\langle x, y, p\rangle = \langle \{$*at the farm. In total, how many*$\}$, ADD, **ABw**$\rangle$. If we replace the placeholders **A** and **B** with the word sequences associated with its left and right sub-trees, respectively, we will arrive at the word sequence "*3 apples and Keith picked 6 apples at the farm. In total, how many*". Recursively performing such a rewriting procedure in a bottom-up manner, we will end up with a word sequence which is exactly the original input text as illustrated at the top of Figure 3.

Based on the above discussion, we can define $\mathcal{T}(\boldsymbol{x}, \boldsymbol{y})$ as a set that consists of the valid trees:

**Definition 2.1** *For a given text* $\boldsymbol{x}$ *and an expression tree* $\boldsymbol{y}$*, a* ***valid text-math tree*** *satisfies the following two properties: 1) the semantics portion of the tree gives exactly* $\boldsymbol{y}$*, and 2) the text obtained through the recursive rewriting procedure discussed above gives exactly* $\boldsymbol{x}$[3].

Given the definition of the valid text-math trees, we will be able to use a bottom-up procedure to construct the set $\mathcal{T}(\boldsymbol{x}, \boldsymbol{y})$. Similarly, we will be able to construct the set $\mathcal{T}(\boldsymbol{x})$ by considering a forest-structured semantic representation that encodes all possible expression trees following (Lu, 2015). One nice property associated with considering only such joint representations is that there are known algorithms that can be used for performing efficient inference. Indeed, the resulting text-math trees are similar to the *hybrid tree* representations used in (Lu et al., 2008; Lu,

[3]We regard words that appear at different positions in $\boldsymbol{x}$ as distinct words, regardless of their string forms.

2014)[4], where dynamic programming based inference algorithms have been developed. Such algorithms allow $\mathcal{O}(n^3m)$ time complexity for inference where $n$ is the text length and $m$ is the number of grammar rules[5] associated with the latent text-math trees.

We note that some prior systems (Roy and Roth, 2017, 2018) require extra inverse operators – inverse subtraction "$-_r$" and inverse division "$\div_r$" to handle the scenarios where the order of quantities appearing in the text is not consistent with the order that they appear in the expression. Exemplified by the example in the left of Figure 4, by introducing two operators $-_r$ and $\div_r$ to take their operands in a reverse order, the equation on the left is represented as "$(3 \times X_1) = 11 -_r (5 \times X_2)$". However, we do not need such two inverse operators. The two group patterns $[\mathbf{w}]\mathbf{A}[\mathbf{w}]\mathbf{B}[\mathbf{w}]$ and $[\mathbf{w}]\mathbf{B}[\mathbf{w}]\mathbf{A}[\mathbf{w}]$ are capable to capture both orders. A pattern from the first group handles the order that is consistent with the problem text, while a pattern from the second group is able to capture reordering of operands below an operator. Exemplified by Figure 4, reordering is required for the first example, but not for the second, though their texts only differ slightly. Unlike the second example, instead of using the pattern $\mathbf{AwB}$, the first joint representation adopts the pattern $\mathbf{BwA}$ for the SUB expression node. Thus, our model is able to work without the underlying knowledge on whether an operator is commutative or not.

### 2.4 Features

**Discrete Features.** The feature function $\Phi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{t})$ is defined over each node $\langle x, y, p\rangle$ in the joint tree as well as the complete expression tree $\boldsymbol{y}$. For each node $\langle x, y, p\rangle$, we extract word n-gram, the word association pattern, and POS tags for words (Manning et al., 2014). The knowledge that whether a number is relevant to the question (if available in the annotated data) is also taken as a binary feature. To assess the quality of the structure associated with the expression tree (i.e., features defined over $\boldsymbol{y}$), we extract parent-child relational information $(y_a, y_b)$ from $\boldsymbol{y}$, where $y_a$ is the parent of $y_b$, as features. Following previous works (Roy et al., 2016; Liang et al., 2018), we also consider incorporating a lexicon in our model so as to make a fair comparison with such works, although we would like to stress that our model does not strictly require such lexicons for learning. More details are in supplementary material.

**Neural Features.** We design neural features over the pair of the $L$-sized window surrounding the target word $x_i$ in $\boldsymbol{x}$ and an expression tree node $y_j$. The network takes as input the contiguous word sequence $(x_{i-L}, \ldots, x_i, \ldots, x_{i+L})$, whose distributed representation is a simple concatenation of embeddings of each word. The hidden layer applies an affine transformation with an element-wise nonlinear activation function, like tanh and ReLU. The final output layer contains as many nodes as there are expression tree nodes in the training set. The output is a score vector that gives a score for the input word sequence $(x_{i-L}, \ldots, x_i, \ldots, x_{i+L})$ and an expression tree node $y_j$. The neural scoring function is defined as follows:

$$G_\Theta(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{t}) = \sum_{(x,y)\in\mathcal{W}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{t})} c(x, y, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{t}) \times \psi(x, y)$$

where $\mathcal{W}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{t})$ is the set of $(x, y)$ pairs extracted from $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{t})$, $c$ returns the number of occurrences and $\psi(x, y)$ is a score of the target word $x$ with $L$-sized windows and the expression tree node $y$, returned by the neural network. We regard $L$ as a hyperparameter.

### 2.5 Algorithms

Given the complete training set, the log-likelihood is calculated as:

$$\begin{aligned}\mathcal{L}(\Lambda, \Theta) &= \sum_i \log P_{\Lambda,\Theta}(\boldsymbol{y}_i|\boldsymbol{x}_i) \\ &= \sum_i \log \sum_{\boldsymbol{t}\in\mathcal{T}(\boldsymbol{x}_i,\boldsymbol{y}_i)} P_{\Lambda,\Theta}(\boldsymbol{y}_i, \boldsymbol{t}|\boldsymbol{x}_i) \qquad (2)\end{aligned}$$

where $(\boldsymbol{x}_i,\boldsymbol{y}_i)$ refers to $i$-th instance in the training set. The additional $L_2$ regularization term can be introduced to avoid over-fitting. Here, we omit it for brevity.

The goal is to find optimal model parameters, i.e., $\Lambda$ and $\Theta$, which maximize the objective. We first consider the computation of gradients for $\Lambda$. Assuming $\Lambda = \langle\lambda_1, \lambda_2, \ldots, \lambda_N\rangle$, to learn the optimal feature weight values, we can calculate the

[4]They need to handle semantic nodes with arity 1 (which requires special constraints for properly defining $\mathcal{T}(\boldsymbol{x})$ (Lu, 2015)), and their semantic nodes are also assumed to convey semantic type information for guiding the expression tree construction process, while we do not need to consider them.

[5]The grammars are related to the word association patterns. The possible latent text-math trees are constructed based on such grammar rules.

gradient for each $\lambda_k$ in $\Lambda$ as:

$$\frac{\partial \mathcal{L}(\Lambda, \Theta)}{\partial \lambda_k} = \sum_i \sum_{\boldsymbol{t}} \boldsymbol{E}_{P_{\Lambda,\Theta}(\boldsymbol{t}|\boldsymbol{x}_i, \boldsymbol{y}_i)}[\phi_k(\boldsymbol{x}_i, \boldsymbol{y}_i, \boldsymbol{t})] - \sum_i \sum_{\boldsymbol{y},\boldsymbol{t}} \boldsymbol{E}_{P_{\Lambda,\Theta}(\boldsymbol{y},\boldsymbol{t}|\boldsymbol{x}_i)}[\phi_k(\boldsymbol{x}_i, \boldsymbol{y}, \boldsymbol{t})] \quad (3)$$

where $\phi_k(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{t})$ is the number of occurrences for the $k$-th feature extracted from $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{t})$.

We then compute the gradient for the neural network parameters $\Theta$. For an input word window $x$ and a semantic unit $y$, the gradient is defined as:

$$\frac{\partial \mathcal{L}(\Lambda, \Theta)}{\partial \psi(x, y)} = \sum_i \sum_{\boldsymbol{t}} \boldsymbol{E}_{P_{\Lambda,\Theta}(\boldsymbol{t}|\boldsymbol{x}_i, \boldsymbol{y}_i)}[c(x, y, \boldsymbol{x}_i, \boldsymbol{y}_i, \boldsymbol{t})] - \sum_i \sum_{\boldsymbol{y},\boldsymbol{t}} \boldsymbol{E}_{P_{\Lambda,\Theta}(\boldsymbol{y},\boldsymbol{t}|\boldsymbol{x}_i)}[c(x, y, \boldsymbol{x}_i, \boldsymbol{y}, \boldsymbol{t})] \quad (4)$$

The gradients (3,4) can be efficiently calculated by applying a generalized forward-backward algorithm, which allows us to conduct exact inference using the dynamic programming algorithm described in (Lu, 2014). Next, standard methods like gradient descent, L-BFGS (Liu and Nocedal, 1989) can be used to find optimal values for model parameters.

During decoding, the optimal equation tree $\boldsymbol{y}^*$ for a new input $\boldsymbol{x}$ can be calculated by:

$$\begin{aligned} \boldsymbol{y}^* &= \arg\max_{\boldsymbol{y}} P(\boldsymbol{y}|\boldsymbol{x}) \\ &= \arg\max_{\boldsymbol{y}} \sum_{\boldsymbol{t} \in \mathcal{T}(\boldsymbol{x},\boldsymbol{y})} e^{F_{\Lambda,\Theta}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{t})} \\ &\approx \arg\max_{\boldsymbol{y},\boldsymbol{t} \in \mathcal{T}(\boldsymbol{x},\boldsymbol{y})} e^{F_{\Lambda,\Theta}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{t})} \end{aligned} \quad (5)$$

where $\mathcal{T}(\boldsymbol{x}, \boldsymbol{y})$ refers to the set of all possible text-math trees that contain $\boldsymbol{x}$ and $\boldsymbol{y}$.

Instead of directly computing the summation over all possible latent text-math structures, we essentially replace the $\sum$ by the $\max$ operation inside the $\arg\max$. In other words, we first find the latent text-math tree $\boldsymbol{t}^*$ which yields the highest score and contains the input text $\boldsymbol{x}$. Then, the optimal expression tree $\boldsymbol{y}^*$ can be automatically extracted from $\boldsymbol{t}^*$.

An efficient dynamic programming based inference algorithm similar to the work of Lu (2014) was leveraged to find the optimal latent structure $\boldsymbol{t}^*$. We then obtain the optimal expression tree $\boldsymbol{y}^*$ from $\boldsymbol{t}^*$, which is the output of our system for the input problem text $\boldsymbol{x}$.

### 2.6 Comparisons with Roy et al. (2016)

It is worth noting that Roy et al. (2016) also proposed a system that maps text into an equation tree. Unlike this work that maps math problem texts into math expressions in an end-to-end fashion, Roy et al. (2016) designed three classifiers which sequentially make local decisions, namely identifying relevant numbers, recognizing possible variables and producing equations. They also require extra inverse operators to handle the non-commutative operation issues, which is not necessary for our model. Generating equations via a sequence of local classification decisions may propagate errors and even limit the ability to wholisticly understand the underlying semantics of problem texts which is important for predicting correct mathematical operations. In this work, we regard equation parsing problem as a structure prediction task that allows to parse the text to equations from a semantic parsing perspective. Moreover, `Text2Math` is capable to handle both tasks of equation parsing and arithmetic word problems, while the system of Roy et al. (2016) is specific to equation parsing.

## 3 Experiments

**Datasets.** Following prior works (Roy and Roth, 2015; Liang et al., 2018), we focus on two commonly-used benchmark datasets for arithmetic word problems, AI2 (Hosseini et al., 2014) and IL (Roy and Roth, 2015). We consider mathematical relations among numbers and calculate numerical values of the predicted expressions. For equation parsing, we also evaluate our model on the data released by (Roy et al., 2016). A predicted equation is regarded as a correct one if it is mathematically equivalent to the gold equation.

### 3.1 Empirical Results

**Arithmetic Word Problem.** Following previous work (Liang et al., 2018), we conduct 3-fold and 5-fold cross-validation on AI2 and IL, respectively, and report the accuracy scores, as shown in Table 2. Our method achieves competitive results on AI2 and IL. Overall, it performs better than previous systems in terms of average scores.

Ablation tests have been done to investigate the effectiveness of different components, such as POS tags, the lexicon and number relevance, as indicated by "-POS", "-LEX", "-ID" in Table 2. By eliminating POS tag features, we achieve

| System | | AI2 | IL | Average |
|---|---|---|---|---|
| ∗Liang et al. (2018) (Statistical) | | 81.5 | 81.0 | 81.25 |
| ∗Liang et al. (2018) (DNN) | | 69.8 | 70.6 | 70.20 |
| ∗Roy and Roth (2017) | | 76.2 | 71.0 | 73.60 |
| Roy and Roth (2015) | | 78.0 | 73.9 | 75.95 |
| Koncel-Kedziorski et al. (2015) | | 52.4 | 72.9 | 62.65 |
| Roy et al. (2015) | | - | 52.7 | - |
| Kushman et al. (2014) | | 64.0 | 73.7 | 68.85 |
| Hosseini et al. (2014) | | 77.7 | - | - |
| NON-NEURAL | Text2Math | 85.8 | 80.4 | 83.10 |
| | -POS | **86.0** | **81.0** | **83.50** |
| | -LEX | 76.8 | 69.4 | 73.10 |
| | -ID | 75.4 | 78.1 | 76.75 |
| NEURAL | $L = 0$ | 84.8 | 79.7 | 82.25 |
| | $L = 1$ | 84.5 | 80.3 | 82.40 |
| | $L = 2$ | 85.5 | 80.3 | 82.90 |
| | $L = 3$ | 86.2 | 80.9 | 83.55 |
| | $L = 4$ | 85.5 | 80.0 | 82.75 |
| | $L = 5$ | 85.2 | **81.4** | 83.30 |
| | $L = 6$ | **86.5** | 81.0 | **83.75** |

Table 2: **Arithmetic Word Problem**: Accuracy (%) on the two benchmark datasets. (-POS, -LEX, -ID mean that the model excludes POS tags feature, lexicon, or the number relevance feature.) ∗ indicates model uses prior knowledge, such as lexcion and inference rules.

new state-of-the-art results on two datasets, which shows POS tag features do not appear to be helpful in this case. Without using lexicon, the performance drops a lot as expected, but the results are still comparable with most previous systems. These figures demonstrate the effectiveness of the lexicon. It is worth noting that the work of Liang et al. (2018) that achieve previous state-of-the-art results leverage inference rules during the inference phase. Their approach can be regarded as a different way of using the lexicon similar to ours. We also consider the effects of neural features (see Sec. 2.4) with different window sizes $L \in \{0, 1, 2, 3, 4, 5, 6\}$. According to empirical results, a larger window size tends to give better results. One possible reason is that an arithmetic word problem often consists of several sentences, where a large word window is required to capture mathematical semantics.

**Equation Parsing.** We compare our model with previous work (Roy et al., 2016) on the equation parsing dataset, as shown in Table 3. Our method yields competitive results. Unlike the work of Roy et al. (2016), annotations of unknown variables are not required in our model. As reported in (Roy et al., 2016), they trained SPF (Artzi and Zettlemoyer, 2013a), a publicly available semantic parser, with sentence-equation pairs and a seed lexicon for mathematical terms. But it only obtained 3.1% accuracy. The result taken from (Roy et al., 2016) shows that it might be difficult for such a semantic parser in handling the equation

| System | | Equation |
|---|---|---|
| ∗Roy et al. (2016) (Pipeline) | | 71.3 |
| ∗Roy et al. (2016) (Joint) | | 60.9 |
| ∗ SPF (Artzi and Zettlemoyer, 2013a) | | 3.1 |
| NON-NEURAL | Text2Math | 71.4 |
| | -POS | 69.1 |
| | -LEX | 71.4 |
| | +G | **73.2** |
| | +G-LEX | **73.2** |
| NEURAL | $L = 0$ | 71.4 |
| | $L = 1$ | 71.9 |
| | $L = 2$ | 73.8 |
| | $L = 3$ | 73.5 |
| | $L = 4$ | **74.5** |
| | $L = 5$ | 74.0 |
| | $L = 6$ | 73.2 |

Table 3: **Equation Parsing**: Accuracy (%) on equation parsing dataset. -POS: without POS tag features; -LEX: without Lexicon; +G: with gold identification for numbers. ∗ indicates model uses lexicon. Result of SPF (Artzi and Zettlemoyer, 2013a) is taken from Roy et al. (2016).

parsing task even with a high precision lexicon. One possible reason is that mapping text into a math equation is essentially a structure prediction problem. Our model is capable to make guaranteed decisions from a structure prediction perspective. Different from arithmetic word problems, where numbers are explicitly given in the form of digits, some texts from equation parsing corpus describe numbers in string forms. Hence, a structured predictor is used to identify the numbers in the sentence, which achieves 95.3% accuracy. The identifications of numbers are taken as features. We also consider the gold label of numbers, indicated by (+G). The performance improves a lot, which shows that the accurate identification of numbers is necessary to in order to obtain a good performance. By removing POS tag features, there is a slight drop in accuracy. On the other hand, it is worth noting that even without the high precision lexicon, our model can still achieve new state-of-the-art accuracy in this task, while the previous work (Roy et al., 2016) always requires a high precision lexicon to boost performance. Incorporating neural features leads to new state-of-the-art accuracy of 74.5% when $L = 4$.

**Expression Construction.** In arithmetic word problems, the expression consists of several numbers only, exemplified by Problem 1 in Figure 1. In practice, an unknown variable X, representing the goal that the problem aims to calculate, can be appended to the expression to form an equation. We further investigate two constructions: appending the unknown variable X to the beginning or to the end of an expression. Results are listed in the first block of Table 4. For instance,

| Variants | AI2 | IL | Average |
|---|---|---|---|
| Text2Math | 85.8 | 80.4 | 83.10 |
| Prefix X | 84.3 | 81.0 | 82.65 |
| Suffix X | 84.3 | **81.2** | 82.75 |
| Text2Math + Inverse | **86.3** | 80.1 | **83.20** |
| Prefix X + Inverse | 84.0 | 79.9 | 80.45 |
| Suffix X + Inverse | 84.0 | 80.5 | 82.25 |

Table 4: Performance of different constructions for expression of arithmetic word problems and the effects of incorporating inverse operators.

two new constructions of the running example are $X = (29 + (16 + 20))$ as indicated by "Prefix X", and $(29 + (16 + 20)) = X$ reported as "Suffix X". It is interesting that including an $X$ and its position influences the performance. Overall, excluding $X$ works the best which is adopted in this work.

**Inverse Operators.** As we discussed in Sec. 2.3, one distinct advantage of our approach, as compared to others, is that we do not need inverse operators, such as "$-_r$" and "$\div_r$". Our designed word association patterns are capable to handle the reordering issue. Here, we consider model variants by introducing two inverse expression tree nodes, $\text{SUB}_r$ and $\text{DIV}_r$, to represent "$-_r$" and "$\div_r$", respectively. Empirical results, reported in the second block in Table 4, show that Text2Math (without including inverse operators) can obtain comparative results compared to the model variants with inverse operators. These results confirm that our model does not require additional knowledge of the semantics of operands, which is a unique property of our approach.

### 3.2 Qualitative Analysis

**Output Comparisons.** Equation parsing is more challenging than arithmetic word problems, since it requires generating unknown variables mapped to phrases residing in the text. We analyze output of this task to investigate the source that leads to better performance. Comparing predictions made by Pipeline (Roy et al., 2016) and our approach, we found that Text2Math can better capture the meaning of the problem text. We illustrate two examples in Table 5. The Pipeline approach fails to capture the meaning of "*rises to 36% from 3.4%*" which implies subtraction of two numbers, while our model is capable to capture such knowledge. In the second example, Pipeline misunderstands the meaning of "*five more than three*", although it seems correct in a local context. However, an equation should be mapped from the complete sentence that captures mathematical relations in a global perspective. Our model holds such a capability and makes more guaranteed predictions, which proves the efficacy of solving math problems from a structure prediction perspective.

| | |
|---|---|
| **Input**: | *Japan January jobless rate rises to* 3.6% *from* 3.4%. |
| **Gold**: | $X_1 = 0.036 - 0.034$ |
| **Pipeline**: | $X_1 \times 0.036 = 0.034$ |
| Text2Math: | $X_1 = 0.036 - 0.034$ |
| **Input**: | *The number of baseball cards he has is five more than three times the number of football cards.* |
| **Gold**: | $X_1 = 5 + (3 \times X_2)$ |
| **Pipeline**: | $X_1 \times X_2 = 5 - 3$ |
| Text2Math: | $X_1 = 5 + (3 \times X_2)$ |

Table 5: Comparison between predictions made by the previous state-of-the-art system (Roy et al., 2016) (denoted as Pipeline) and Text2Math.

**Robustness.** To further investigate the property of our model, we studied outputs. We found that our method is able to conduct self-correction. Exemplified by Example 3 in Table 6, considering the sentence *"Germany's DAX opens 0.7% lower at 18,842."* with annotated equation $X_1 + (0.007 \times X_1) = 18,842$, the prediction made by our method is $X_1 - (0.007 \times X_1) = 18,842$. It can be seen that the prediction made by our method is supposed to be the correct one, while the annotation is actually wrong. To make a fair comparison with previous works, we did not count such cases as correct during evaluation, which implies that accuracy reported in Table 3 is in fact higher.

**Error Analysis.** For arithmetic word problem, it is interesting that the operand of two operands should be addition/subtraction (multiplication/division), while the prediction is subtraction/addition (division/multiplication). Consider Example 4 and 5 in Table 6. Descriptions of such two problems share many words, such as *each*, *how many*, *there are*, etc. Slight difference in problem descriptions may lead to different results, which makes it a challenge.

As for equation parsing, the work of Roy et al. (2016) requires annotations on which phrases should be mapped to unknowns during the training phase. However, such supervised knowledge is not required for our method. In our setup, we did not make hard constraint that each prediction must contain one or two variables. Therefore, missing or redundant variables appearing in the predicted equations are one of the major error sources. Example 6 and 7 from Table 6 illustrate such cases. On the other hand, lack of professional background information also leads to miss-

| | |
|---|---|
| **Example 3**: | *Germany's DAX opens 0.7% lower at 10,842.* |
| **Gold**: | $X_1 + (0.007 \times X_1) = 10842$ |
| Text2Math: | $X_1 - (0.007 \times X_1) = 10842$ |
| **Example 4**: | *Each child has 5 bottle caps. If there are 9 children, how many bottle caps are there in total?* |
| **Gold**: | $5 \times 9$ |
| Text2Math: | $5 \div 9$ |
| **Example 5**: | *The school is planning a field trip. There are 14 students and 2 seats on each school bus. How many buses are needed to take the trip?* |
| **Gold**: | $14 \div 2$ |
| Text2Math: | $14 \times 2$ |
| **Example 6**: | *530 pesos can buy 4 kilograms of fish and 2 kilograms of pork.* |
| **Gold**: | $530 = (4 \times X_1) + (2 \times X_2)$ |
| Text2Math: | $530 \times X_3 = (4 \times X_1) + (2 \times X_2)$ |
| **Example 7**: | *Flying with the wind , a bird was able to make 150 kilometers per hour.* |
| **Gold**: | $X_1 + X_2 = 150$ |
| Text2Math: | $X_1 = 150$ |

Table 6: Examples with wrong predictions. Gold denotes the annotated correct equations and Text2Math refers to output equations generated by our method.

ing variables. Consider Example 6. Without world knowledge, it might be difficult for the algorithm to recognize that "*Flying with the wind*" implies the speed of the wind which should be considered as a variable of the equation.

## 4 Related Work

**Math Word Problems.** Mukherjee and Garain (2008) surveyed related approaches to this task in literature. Hosseini et al. (2014); Mitra and Baral (2016) solved the task by categorizing verbs or problems. The first method that can handle general arithmetic problems with multiple steps was proposed by Roy and Roth (2015), which was further extended by introducing (Roy and Roth, 2017, 2018). Zou and Lu (2019b,c) is the first work that proposed a sequence labelling approach to solving arithmetic word problems, which focuses on addition-subtraction word problems. Other systems include semantic parsing based approaches (Liang et al., 2018) and neural methods (Wang et al., 2017, 2018a,b). Unlike arithmetic word problems, the goal of algebra word problems is to map the text to an equation set (Kushman et al., 2014; Shi et al., 2015). Other types of problems have also been investigated, including probability problems (Dries et al., 2017), logic puzzle problems (Mitra and Baral, 2015; Chesani et al., 2017) and geometry problems (Seo et al., 2014, 2015). Besides the benchmark datasets used in this work, other popular datasets include Dolphin18K (Shi et al., 2015) and AQuA (Ling et al., 2017) for algebra word problems which are not the focus in this work. Roy et al. (2016) first proposed the Equation Parsing task and designed a pipeline method with three structured predictors.

**Semantic Parsing.** Another line of related works is semantic parsing (Wong and Mooney, 2006; Zettlemoyer and Collins, 2007; Kwiatkowksi et al., 2010; Liang et al., 2011; Dong and Lapata, 2018; Zou and Lu, 2018), which aims to map sentences into logic forms, including CCG-based lambda calculus expressions (Zettlemoyer and Collins, 2007; Kwiatkowksi et al., 2010; Artzi and Zettlemoyer, 2013b; Dong and Lapata, 2016), FunQL (Kate et al., 2005; Wong and Mooney, 2006; Jones et al., 2012), lambda-DCS (Liang et al., 2011; Berant et al., 2013; Jia and Liang, 2016), graph queries (Harris et al., 2013; Holzschuher and Peinl, 2013) and SQL (Yin et al., 2015; Sun et al., 2018). In this work, we adopt a text-math semantic representation encoding words and the expression tree.

## 5 Conclusion

In this work, we propose a unified structured prediction approach, Text2Math, to solving both arithmetic word problems and equation parsing tasks. We leverage a novel joint representation to automatically learn the correspondence between words and math expressions which reflects semantic closeness. Different from many existing models, Text2Math is agnostic of the semantics of operands and learns to map from text to math expressions in an end-to-end manner based on a data-driven approach. Experiments demonstrate the efficacy of our model. In the future, we would like to investigate how such an approach can be applied to more complicated math word problems, like algebra word problems where a problem usually maps to an equation set. Another interesting direction is to investigate how to incorporate world knowledge into the graph-based approach to boost the performance.

## Acknowledgments

We would like to thank the three anonymous reviewers for their thoughtful and constructive comments. This work is supported by Singapore Ministry of Education Academic Research Fund (AcRF) Tier 2 Project MOE2017-T2-1-156.

## References


Yoav Artzi and Luke Zettlemoyer. 2013a. Uw spf: The university of washington semantic parsing framework. *arXiv preprint arXiv:1311.3011*.

Yoav Artzi and Luke Zettlemoyer. 2013b. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of EMNLP*.

Daniel G Bobrow. 1964. Natural language input for a computer problem solving system. Technical report, MASSACHUSETTS INST OF TECH CAMBRIDGE PROJECT MAC.

Federico Chesani, Paola Mello, and Michela Milano. 2017. Solving mathematical puzzles: A challenging competition for ai. *AI Magazine*, 38.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of ACL*.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of ACL*.

Anton Dries, Angelika Kimmig, Jesse Davis, Vaishak Belle, and Luc De Raedt. 2017. Solving probability problems in natural language. In *Proceedings of IJCAI*.

Steve Harris, Andy Seaborne, and Eric Prud'hommeaux. 2013. Sparql 1.1 query language. *W3C Recommendation*, 21(10):778.

Florian Holzschuher and René Peinl. 2013. Performance of graph query languages: comparison of cypher, gremlin and native access in neo4j. In *Proceedings of the Joint EDBT/ICDT 2013 Workshops*.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of EMNLP*.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of ACL*.

Bevan Jones, Mark Johnson, and Sharon Goldwater. 2012. Semantic parsing with bayesian tree transducers. In *Proceedings of ACL*.

Rohit J Kate, Yuk Wah Wong, and Raymond J Mooney. 2005. Learning to transform natural to formal languages. In *Proceedings of AAAI*.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proceedings of ACL*.

Tom Kwiatkowksi, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic ccg grammars from logical form with higher-order unification. In *Proceedings of EMNLP*.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT*.

Chao-Chun Liang, Yu-Shiang Wong, Yi-Chung Lin, and Keh-Yih Su. 2018. A meaning-based statistical english math word problem solver. In *Proceedings of NAACL*.

Percy Liang, Michael Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of ACL*.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of ACL*.

Dong C Liu and Jorge Nocedal. 1989. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45.

Wei Lu. 2014. Semantic parsing with relaxed hybrid trees. In *Proceedings of EMNLP*.

Wei Lu. 2015. Constrained semantic forests for improved discriminative semantic parsing. In *Proceedings of ACL*.

Wei Lu, Hwee Tou Ng, Wee Sun Lee, and Luke S. Zettlemoyer. 2008. A generative model for parsing natural language to meaning representations. In *Proceedings of EMNLP*.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of ACL*.

Arindam Mitra and Chitta Baral. 2015. Learning to automatically solve logic grid puzzles. In *Proceedings of EMNLP*.

Arindam Mitra and Chitta Baral. 2016. Learning to use formulas to solve simple arithmetic problems. In *Proceedings of ACL*.

Anirban Mukherjee and Utpal Garain. 2008. A review of methods for automatic understanding of natural language mathematical problems. *Artificial Intelligence Review*, 29(2).

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of EMNLP*.

Subhro Roy and Dan Roth. 2017. Unit dependency graph and its application to arithmetic word problem solving. In *Proceedings of AAAI*.

Subhro Roy and Dan Roth. 2018. Mapping to declarative knowledge for word problem solving. *Transactions of the Association of Computational Linguistics*, 6:159–172.

Subhro Roy, Shyam Upadhyay, and Dan Roth. 2016. Equation parsing: Mapping sentences to grounded equations. In *Proceedings of EMNLP*.

Subhro Roy, Tim Vieira, and Dan Rote. 2015. Reasoning about quantities in natural language. *Transactions of the Association for Computational Linguistics*, 3(1):1–13.

Min Joon Seo, Hannaneh Hajishirzi, Ali Farhadi, and Oren Etzioni. 2014. Diagram understanding in geometry questions. In *Proceedings of AAAI*.

Minjoon Seo, Hannaneh Hajishirzi, Ali Farhadi, Oren Etzioni, and Clint Malcolm. 2015. Solving geometry problems: Combining text and diagram interpretation. In *Proceedings of EMNLP*.

Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu, and Yong Rui. 2015. Automatically solving number word problems by semantic parsing and reasoning. In *Proceedings of EMNLP*.

Yibo Sun, Duyu Tang, Nan Duan, Jianshu Ji, Guihong Cao, Xiaocheng Feng, Bing Qin, Ting Liu, and Ming Zhou. 2018. Semantic parsing with syntax- and table-aware sql generation. In *Proceedings of ACL*.

Lei Wang, Yan Wang, Deng Cai, Dongxiang Zhang, and Xiaojiang Liu. 2018a. Translating a math word problem to an expression tree. In *Proceedings of EMNLP*.

Lei Wang, Dongxiang Zhang, Lianli Gao, Jingkuan Song, Long Guo, and Heng Tao Shen. 2018b. Mathdqn: Solving arithmetic word problems via deep reinforcement learning. In *Proceedings of AAAI*.

Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017. Deep neural solver for math word problems. In *Proceedings of EMNLP*.

Yuk Wah Wong and Raymond Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of NAACL*.

Pengcheng Yin, Zhengdong Lu, Hang Li, and Ben Kao. 2015. Neural enquirer: Learning to query tables with natural language. *arXiv preprint arXiv:1512.00965*.

Luke Zettlemoyer and Michael Collins. 2007. Online learning of relaxed ccg grammars for parsing to logical form. In *Proceedings of EMNLP-CoNLL*.

Yanyan Zou and Wei Lu. 2018. Learning cross-lingual distributed logical representations for semantic parsing. In *Proceedings of ACL*.

Yanyan Zou and Wei Lu. 2019a. Joint detection and location of english puns. In *Proceedings of ACL*.

Yanyan Zou and Wei Lu. 2019b. Quantity tagger: A latent-variable sequence labeling approach tosolving addition-subtraction word problems. In *Proceedings of ACL*.

Yanyan Zou and Wei Lu. 2019c. Supplementary material for quantity tagger: A latent-variable sequence labeling approach tosolving addition-subtraction word problems. In *Proceedings of ACL*.

# Editing-Based SQL Query Generation for Cross-Domain Context-Dependent Questions

**Rui Zhang**[†] **Tao Yu**[†] **He Yang Er**[†]
**Sungrok Shim**[†] **Eric Xue**[†] **Xi Victoria Lin**[¶] **Tianze Shi**[§]
**Caiming Xiong**[¶] **Richard Socher**[¶] **Dragomir Radev**[†]
[†] Yale University [¶] Salesforce Research [§] Cornell University
{r.zhang, tao.yu, dragomir.radev}@yale.edu
{xilin, cxiong, rsocher}@salesforce.com

## Abstract

We focus on the cross-domain context-dependent text-to-SQL generation task. Based on the observation that adjacent natural language questions are often linguistically dependent and their corresponding SQL queries tend to overlap, we utilize the interaction history by editing the previous predicted query to improve the generation quality. Our editing mechanism views SQL as sequences and reuses generation results at the token level in a simple manner. It is flexible to change individual tokens and robust to error propagation. Furthermore, to deal with complex table structures in different domains, we employ an utterance-table encoder and a table-aware decoder to incorporate the context of the user utterance and the table schema. We evaluate our approach on the SParC dataset and demonstrate the benefit of editing compared with the state-of-the-art baselines which generate SQL from scratch. Our code is available at https://github.com/ryanzhumich/sparc_atis_pytorch.

## 1 Introduction

Generating SQL queries from user utterances is an important task to help end users acquire information from databases. In a real-world application, users often access information in a multi-turn interaction with the system by asking a sequence of related questions. As the interaction proceeds, the user often makes reference to the relevant mentions in the history or omits previously conveyed information assuming it is known to the system.

Therefore, in the context-dependent scenario, the contextual history is crucial to understand the follow-up questions from users, and the system often needs to reproduce partial sequences generated in previous turns. Recently, Suhr et al. (2018) proposes a context-dependent text-to-SQL model including an interaction-level encoder and an attention mechanism over previous utterances. To reuse what has been generated, they propose to copy complete segments from the previous query. While their model is successful to reason about explicit and implicit references, it does not need to explicitly address different database schemas because the ATIS contains only the flight-booking domain. Furthermore, the model is confined to copy whole segments which are extracted by a rule-based procedure, limiting its capacity to utilize the previous query when only one or a few tokens are changed in the segment.

To exploit the correlation between sequentially generated queries and generalize the system to different domains, in this paper, we study an editing-based approach for cross-domain context-dependent text-to-SQL generation task. We propose query generation by editing the query in the previous turn. To this end, we first encode the previous query as a sequence of tokens, and the decoder computes a switch to change it at the token level. This sequence editing mechanism models token-level changes and is thus robust to error propagation. Furthermore, to capture the user utterance and the complex database schemas in different domains, we use an utterance-table encoder based on BERT to jointly encode the user utterance and column headers with co-attention, and adopt a table-aware decoder to perform SQL generation with attentions over both the user utterance and column headers.

We evaluate our model on SParC (Yu et al., 2019b), a new large-scale dataset for cross-domain semantic parsing in context consisting of coherent question sequences annotated with SQL queries over 200 databases in 138 domains. Experiment results show that by generating from the previous query, our model delivers an improvement of 7% question match accuracy and 11% interaction



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5338–5349,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| | Context | Cross-Domain | Interaction (train / dev / test) | Question | Turn | Database | Table | Q. Length | Q. Vocab |
|---|---|---|---|---|---|---|---|---|---|
| Spider | ✗ | ✓ | 11,840 (8,659 / 1,034 / 2,147) | 11,840 | 1.0 | 200 | 1020 | 13.4 | 4,818 |
| ATIS | ✓ | ✗ | 1,658 (1,148 / 380 / 130) | 11,653 | 7.0 | 1 | 27 | 10.2 | 1,582 |
| SParC | ✓ | ✓ | 4,298 (3,034 / 422 / 842) | 12,726 | 3.0 | 200 | 1020 | 8.1 | 3,794 |

Table 1: Dataset Statistics.

| | WHERE | AGG | GROUP | ORDER | HAVING | SET | JOIN | Nested |
|---|---|---|---|---|---|---|---|---|
| Spider | 55.2 | 51.7 | 24.0 | 21.5 | 6.7 | 5.8 | 42.9 | 15.7 |
| ATIS | 100 | 16.6 | 0.3 | 0 | 0 | 0 | 96.6 | 96.6 |
| SParC | 42.8 | 39.8 | 20.1 | 17.0 | 4.7 | 3.5 | 35.5 | 5.7 |

Table 2: % of SQL queries that contain a particular SQL component.

Database: student dormitory containing 5 tables.

Goal: Find the first and last names of the students who are living in the dorms that have a TV Lounge as an amenity.

Q1: How many dorms have a TV Lounge?

S1:
```
SELECT COUNT(*) FROM dorm AS T1 JOIN has_amenity AS T2 ON T1.dormid = T2.dormid JOIN dorm_amenity AS T3
ON T2.amenid = T3.amenid WHERE T3.amenity_name = 'TV Lounge'
```

Q2: What is the total capacity of these dorms?

S2:
```
SELECT SUM(T1.student_capacity) FROM dorm AS T1 JOIN has_amenity AS T2 ON T1.dormid = T2.dormid JOIN
dorm_amenity AS T3 ON T2.amenid = T3.amenid WHERE T3.amenity_name = 'TV Lounge'
```

Q3: How many students are living there?

S3:
```
SELECT COUNT(*) FROM student AS T1 JOIN lives_in AS T2 ON T1.stuid = T2.stuid WHERE T2.dormid
IN (SELECT T3.dormid FROM has_amenity AS T3 JOIN dorm_amenity AS T4 ON T3.amenid = T4.amenid
WHERE T4.amenity_name = 'TV Lounge')
```

Q4: Please show their first and last names.

S4:
```
SELECT T1.fname, T1.lname FROM student AS T1 JOIN lives_in AS T2 ON T1.stuid = T2.stuid WHERE T2.dormid
IN (SELECT T3.dormid FROM has_amenity AS T3 JOIN dorm_amenity AS T4 ON T3.amenid = T4.amenid
WHERE T4.amenity_name = 'TV Lounge')
```

Table 3: SParC example.

match accuracy over the previous state-of-the-art. Further analysis shows that our editing approach is more robust to error propagation than copying segments, and the improvement becomes more significant if the basic text-to-SQL generation accuracy (without editing) improves.

## 2 Cross-Domain Context-Depencent Semantic Parsing

### 2.1 Datasets

We use SParC [1] (Yu et al., 2019b), a large-scale cross-domain context-dependent semantic parsing dataset with SQL labels, as our main evaluation benchmark. A SParC example is shown in Table 3. We also report performance on ATIS (Hemphill et al., 1990; Dahl et al., 1994a) for direct comparison to Suhr et al. (2018). In addition, we evaluate the cross-domain context-independent text-to-SQL ability of our model on Spider[2] (Yu et al., 2018c), which SParC is built on.

We summarize and compare the data statistics in Table 1 and Table 2. While the ATIS dataset has been extensively studied, it is limited to a particular domain. By contrast, SParC is both context-dependent and cross-domain. Each interaction in SParC is constructed using a question in Spider as the interaction goal, where the annotator asks inter-related questions to obtain information that completes the goal. SParC contains interactions over 200 databases and it follows the same database split as Spider where each database appears only in one of train, dev and test sets. In summary, SParC introduces new challenges to context-dependent text-to-SQL because it (1) contains more complex context dependencies, (2) has greater semantic coverage, and (3) adopts a cross-domain task setting.

### 2.2 Task Formulation

Let $X$ denote a natural language utterance and $Y$ denote the corresponding SQL query. Context-

[1] https://yale-lily.github.io/sparc

[2] https://yale-lily.github.io/spider

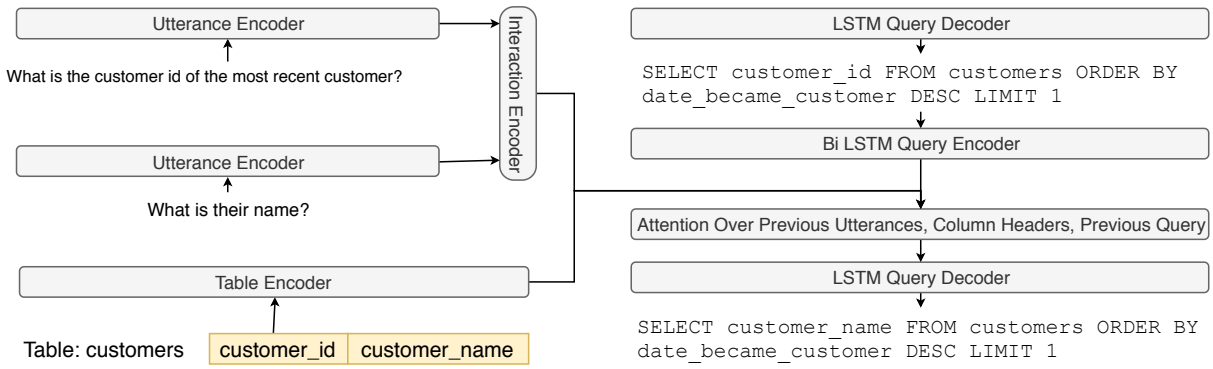


Figure 1: Model architecture of editing the previous query with attentions to the user utterances, the table schema, and the previously generated query.

independent semantic parsing considers individual $(X, Y)$ pairs and maps $X$ to $Y$. In context-dependent semantic parsing, we consider an interaction $I$ consisting of $n$ utterance-query pairs in a sequence:

$$I = [(X_i, Y_i)]_{i=1}^{n}$$

At each turn $t$, the goal is to generate $Y_t$ given the current utterance $X_t$ and the interaction history

$$[(X_1, Y_1), (X_2, Y_2), \ldots, (X_{t-1}, Y_{t-1})]$$

Furthermore, in the cross-domain setting, each interaction is grounded to a different database. Therefore, the model is also given the schema of the current database as an input. We consider relational databases with multiple tables, and each table contains multiple column headers:

$$T = [c_1, c_2, \ldots, c_l, \ldots, c_m]$$

where $m$ is the number of column headers, and each $c_l$ consists of multiple words including its table name and column name (§ 3.1).

## 3 Methodology

We employ an encoder-decoder architecture with attention mechanisms (Sutskever et al., 2014; Luong et al., 2015) as illustrated in Figure 1. The framework consists of (1) an utterance-table encoder to explicitly encode the user utterance and table schema at each turn, (2) A turn attention incorporating the recent history for decoding, (3) a table-aware decoder taking into account the context of the utterance, the table schema, and the previously generated query to make editing decisions.

### 3.1 Utterance-Table Encoder

An effective encoder captures the meaning of user utterances, the structure of table schema, and the relationship between the two. To this end, we build an utterance-table encoder with co-attention between the two as illustrated in Figure 2.

Figure 2b shows the utterance encoder. For the user utterance at each turn, we first use a bi-LSTM to encode utterance tokens. The bi-LSTM hidden state is fed into a dot-product attention layer (Luong et al., 2015) over the column header embeddings. For each utterance token embedding, we get an attention weighted average of the column header embeddings to obtain the most relevant columns (Dong and Lapata, 2018). We then concatenate the bi-LSTM hidden state and the column attention vector, and use a second layer bi-LSTM to generate the utterance token embedding $\mathbf{h}^E$.

Figure 2c shows the table encoder. For each column header, we concatenate its table name and its column name separated by a special dot token (i.e., `table name . column name`). Each column header is processed by a bi-LSTM layer. To better capture the internal structure of the table schemas (e.g., foreign key), we then employ a self-attention (Vaswani et al., 2017) among all column headers. We then use an attention layer to capture the relationship between the utterance and the table schema. We concatenate the self-attention vector and the utterance attention vector, and use a second layer bi-LSTM to generate the column header embedding $\mathbf{h}^C$.

Note that the two embeddings depend on each other due to the co-attention, and thus the column header representation changes across different utterances in a single interaction.

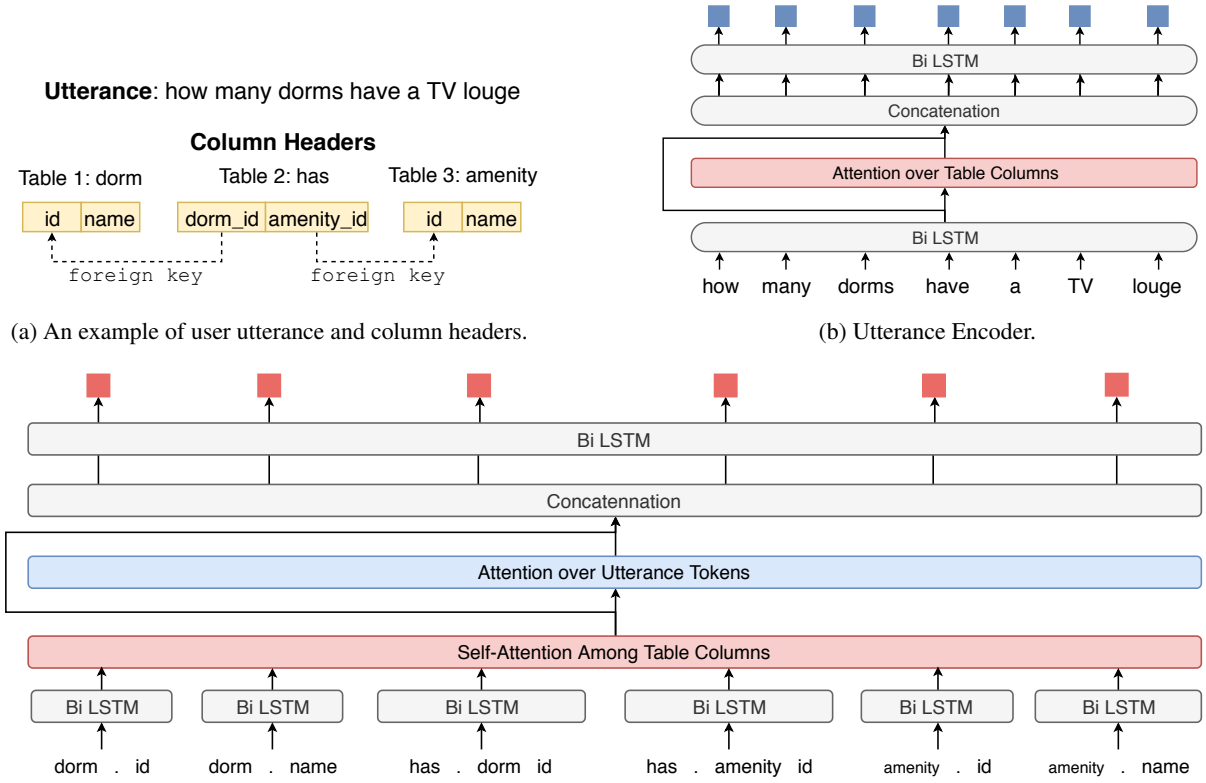


(a) An example of user utterance and column headers.

(b) Utterance Encoder.

(c) Table Encoder.

Figure 2: Utterance-Table Encoder for the example in (a).

**Utterance-Table BERT Embedding**. We consider two options as the input to the first layer bi-LSTM. The first choice is the pretrained word embedding. Second, we also consider the contextualized word embedding based on BERT (Devlin et al., 2019). To be specific, we follow Hwang et al. (2019) to concatenate the user utterance and all the column headers in a single sequence separated by the `[SEP]` token:

$$\texttt{[CLS]}, X_i, \texttt{[SEP]}, c_1, \texttt{[SEP]}, \ldots, c_m, \texttt{[SEP]}$$

This sequence is fed into the pretrained BERT model whose hidden states at the last layer is used as the input embedding.

### 3.2 Interaction Encoder with Turn Attention

To capture the information across different utterances, we use an interaction-level encoder (Suhr et al., 2018) on top of the utterance-level encoder. At each turn, we use the hidden state at the last time step from the utterance-level encoder as the utterance encoding. This is the input to a unidirectional LSTM interaction encoder:

$$\begin{aligned} \mathbf{h}_i^U &= \mathbf{h}_{i,|X_i|}^E \\ \mathbf{h}_{i+1}^I &= \text{LSTM}^I(\mathbf{h}_i^U, \mathbf{h}_i^I) \end{aligned}$$

The hidden state of this interaction encoder $\mathbf{h}^I$ encodes the history as the interaction proceeds.

**Turn Attention** When issuing the current utterance, the user may omit or explicitly refer to the previously mentioned information. To this end, we adopt the turn attention mechanism to capture correlation between the current utterance and the utterance(s) at specific turn(s). At the current turn $t$, we compute the turn attention by the dot-product attention between the current utterance and previous utterances in the history, and then add the weighted average of previous utterance embeddings to the current utterance embedding:

$$\begin{aligned} s_i &= \mathbf{h}_t^U \mathbf{W}_{\text{turn-att}} \mathbf{h}_i^U \\ \alpha^{\text{turn}} &= \text{softmax}(s) \\ \mathbf{c}_t^{\text{turn}} &= \mathbf{h}_t^U + \sum_{i=1}^{t-1} \alpha_i^{\text{turn}} \times \mathbf{h}_i^U \end{aligned} \tag{1}$$

The $\mathbf{c}_t^{\text{turn}}$ summarizes the context information and the current user query and will be used as the initial decoder state as described in the following.

### 3.3 Table-aware Decoder

We use an LSTM decoder with attention to generate SQL queries by incorporating the interaction

history, the current user utterance, and the table schema.

Denote the decoding step as $k$, we provide the decoder input as a concatenation of the embedding of SQL query token $\mathbf{q}_k$ and a context vector $\mathbf{c}_k$:

$$\mathbf{h}^D_{k+1} = \text{LSTM}^D([\mathbf{q}_k; \mathbf{c}_k], \mathbf{h}^D_k)$$

where $\mathbf{h}^D$ is the hidden state of the decoder $\text{LSTM}^D$, and the hidden state $\mathbf{h}^D_0$ is initialized by $\mathbf{c}^{\text{turn}}_t$. When the query token is a SQL keyword, $\mathbf{q}_k$ is a learned embedding; when it is a column header, we use the column header embedding given by the table-utterance encoder as $\mathbf{q}_k$. The context vector $\mathbf{c}_k$ is described below.

**Context Vector with the Table and User Utterance**. The context vector consists of attentions to both the table and the user utterance. First, at each step $k$, the decoder computes the attention between the decoder hidden state and the column header embedding.

$$\begin{aligned} s_l &= \mathbf{h}^D_k \mathbf{W}_{\text{column-att}} \mathbf{h}^C_l \\ \alpha^{\text{column}} &= \text{softmax}(s) \\ \mathbf{c}^{\text{column}}_k &= \sum_l \alpha^{\text{column}}_l \times \mathbf{h}^C_l \end{aligned} \tag{2}$$

where $l$ is the index of column headers and $\mathbf{h}^C_l$ is its embedding. Second, it also computes the attention between the decoder hidden state and the utterance token embeddings:

$$\begin{aligned} s_{i,j} &= \mathbf{h}^D_k \mathbf{W}_{\text{utterance-att}} \mathbf{h}^E_{i,j} \\ \alpha^{\text{utterance}} &= \text{softmax}(s) \\ \mathbf{c}^{\text{token}}_k &= \sum_{i,j} \alpha^{\text{utterance}}_{i,j} \times \mathbf{h}^E_{i,j} \end{aligned} \tag{3}$$

where $i$ is the turn index, $j$ is the token index, and $\mathbf{h}^E_{i,j}$ is the token embedding for the $j$-th token of $i$-th utterance. The context vector $\mathbf{c}_k$ is a concatenation of the two:

$$\mathbf{c}_k = [\mathbf{c}^{\text{column}}_k; \mathbf{c}^{\text{token}}_k]$$

**Output Distribution**. In the output layer, our decoder chooses to generate a SQL keyword (e.g., `SELECT`, `WHERE`, `GROUP BY`, `ORDER BY`) or a column header. This is critical for the cross-domain setting where the table schema changes across different examples. To achieve this, we use separate layers to score SQL keywords and column headers, and finally use the softmax operation

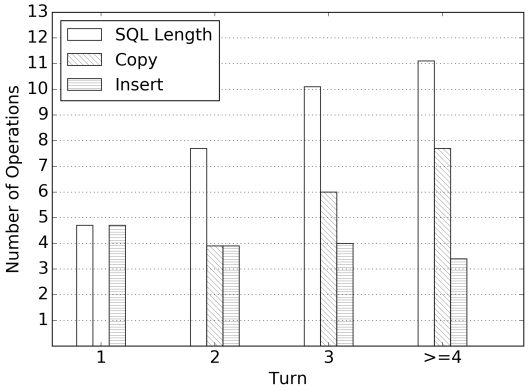


Figure 3: Number of operations at different turns.

to generate the output probability distribution:

$$\begin{aligned} \mathbf{o}_k &= \tanh([\mathbf{h}^D_k; \mathbf{c}_k]\mathbf{W}_o) \\ \mathbf{m}^{\text{SQL}} &= \mathbf{o}_k \mathbf{W}_{\text{SQL}} + \mathbf{b}_{\text{SQL}} \\ \mathbf{m}^{\text{column}} &= \mathbf{o}_k \mathbf{W}_{\text{column}} \mathbf{h}^C \\ P(y_k) &= \text{softmax}([\mathbf{m}^{\text{SQL}}; \mathbf{m}^{\text{column}}]) \end{aligned} \tag{4}$$

### 3.4 Query Editing Mechanism

In an interaction with the system, the user often asks a sequence of closely related questions to complete the final query goal. Therefore, the query generated for the current turn often overlaps significantly with the previous ones.

To empirically verify the usefulness of leveraging the previous query, we consider the process of generating the current query by applying copy and insert operations to the previous query[3]. Figure 3 shows the SQL query length and the number of copy and insert operations at different turns. As the interaction proceeds, the user question becomes more complicated as it requires longer SQL query to answer. However, more query tokens overlap with the previous query, and thus the number of new tokens remains small at the third turn and beyond.

Based on this observation, we extend our tableware decoder with a query editing mechanism. We first encode the previous query using another bi-LSTM, and its hidden states are the query token embeddings $\mathbf{h}^Q_{i,j'}$ (i.e., the $j'$-th token of the $i$-th query). We then extend the context vector with the attention to the previous query:

$$\mathbf{c}_k = [\mathbf{c}^{\text{column}}_k; \mathbf{c}^{\text{token}}_k; \mathbf{c}^{\text{query}}_k]$$

where $\mathbf{c}^{\text{query}}_k$ is produced by an attention to query tokens $\mathbf{h}^Q_{i,j'}$ in the same form as Equation 3.

[3]We use a diffing algorithm from https://github.com/paulgb/simplediff

At each decoding step, we predict a switch $p_{\text{copy}}$ to decide if we need copy from the previous query or insert a new token.

$$\begin{aligned} p_{\text{copy}} &= \sigma(\mathbf{c}_k \mathbf{W}_{\text{copy}} + \mathbf{b}_{\text{copy}}) \\ p_{\text{insert}} &= 1 - p_{\text{copy}} \end{aligned} \tag{5}$$

Then, we use a separate layer to score the query tokens at turn $t-1$, and the output distribution is modified as the following to take into account the editing probability:

$$\begin{aligned} P_{\text{prev\_SQL}} &= \text{softmax}(\mathbf{o}_k \mathbf{W}_{\text{prev\_SQL}} \mathbf{h}_{t-1}^{Q}) \\ \mathbf{m}^{\text{SQL}} &= \mathbf{o}_k \mathbf{W}_{\text{SQL}} + \mathbf{b}_{\text{SQL}} \\ \mathbf{m}^{\text{column}} &= \mathbf{o}_k \mathbf{W}_{\text{column}} \mathbf{h}^{C} \\ P_{\text{SQL} \bigcup \text{column}} &= \text{softmax}([\mathbf{m}^{\text{SQL}}; \mathbf{m}^{\text{column}}]) \\ P(y_k) &= p_{\text{copy}} \cdot P_{\text{prev\_SQL}}(y_k \in \text{prev\_SQL}) \\ &+ p_{\text{insert}} \cdot P_{\text{SQL} \bigcup \text{column}}(y_k \in \text{SQL} \bigcup \text{column}) \end{aligned} \tag{6}$$

While the copy mechanism has been introduced by Gu et al. (2016) and See et al. (2017), they focus on summarization or response generation applications by copying from the source sentences. By contrast, our focus is on editing the previously generated query while incorporating the context of user utterances and table schemas.

## 4 Related Work

Semantic parsing is the task of mapping natural language sentences into formal representations. It has been studied for decades including using linguistically-motivated compositional representations, such as logical forms (Zelle and Mooney, 1996; Clarke et al., 2010) and lambda calculus (Zettlemoyer and Collins, 2005; Artzi and Zettlemoyer, 2011), and using executable programs, such as SQL queries (Miller et al., 1996; Zhong et al., 2017) and other general-purpose programming languages (Yin and Neubig, 2017; Iyer et al., 2018). Most of the early studies worked on a few domains and small datasets such as GeoQuery (Zelle and Mooney, 1996) and Overnight (Wang et al., 2015).

Recently, large and cross-domain text-to-SQL datasets such as WikiSQL (Zhong et al., 2017) and Spider (Yu et al., 2018c) have received an increasing amount of attention as many data-driven neural approaches achieve promising results (Dong and Lapata, 2016; Su and Yan, 2017; Iyer et al., 2017; Xu et al., 2017; Finegan-Dollak et al., 2018; Yu et al., 2018a; Huang et al., 2018; Dong and Lapata, 2018; Sun et al., 2018; Gur et al., 2018; Guo et al., 2018; Yavuz et al., 2018; Shi et al., 2018). Most of them still focus on *context-independent* semantic parsing by converting single-turn questions into executable queries.

Relatively less effort has been devoted to *context-dependent* semantic parsing on datasets including ATIS (Hemphill et al., 1990; Dahl et al., 1994b), SpaceBook (Vlachos and Clark, 2014), SCONE (Long et al., 2016; Guu et al., 2017; Fried et al., 2018; Suhr and Artzi, 2018; Huang et al., 2019), SequentialQA (Iyyer et al., 2017), SParC (Yu et al., 2019b) and CoSQL (Yu et al., 2019a). On ATIS, Miller et al. (1996) maps utterances to semantic frames which are then mapped to SQL queries; Zettlemoyer and Collins (2009) starts with context-independent Combinatory Categorial Grammar (CCG) parsing and then resolves references to generate lambda-calculus logical forms for sequences of sentences. The most relevant to our work is Suhr et al. (2018), who generate ATIS SQL queries from interactions by incorporating history with an interaction-level encoder and copying segments of previously generated queries. Furthermore, SCONE contains three domains using stack- or list-like elements and most queries include a single binary predicate. SequentialQA is created by decomposing some complicated questions in WikiTableQuestions (Pasupat and Liang, 2015). Since both SCONE and SequentialQA are annotated with only denotations but not query labels, they don't include many questions with rich semantic and contextual types. For example, SequentialQA (Iyyer et al., 2017) requires that the answer to follow-up questions must be a subset of previous answers, and most of the questions can be answered by simple SQL queries with `SELECT` and `WHERE` clauses.

Concurrent with our work, Yu et al. (2019a) introduced CoSQL, a large-scale cross-domain conversational text-to-SQL corpus collected under the Wizard-of-Oz setting. Each dialogue in CoSQL simulates a DB querying scenario with a crowd worker as a user and a college computer science student who is familiar with SQL as an expert. Question-SQL pairs in CoSQL reflect greater diversity in user backgrounds compared to other corpora and involve frequent changes in user intent between pairs or ambiguous questions that require user clarification. These features pose new chal-

lenges for text-to-SQL systems.

Our work is also related to recently proposed approaches to code generation by editing (Hayati et al., 2018; Yin et al., 2019; Hashimoto et al., 2018). While they follow the framework of generating code by editing the relevant examples retrieved from training data, we focus on a context-dependent setting where we generate queries from the previous query predicted by the system itself.

## 5 Experimental Results

### 5.1 Metrics

On both Spider and SParC, we use the exact set match accuracy between the gold and the predicted queries [4]. To avoid ordering issues, instead of using simple string matching, Yu et al. (2018c) decompose predicted queries into different SQL clauses such as `SELECT`, `WHERE`, `GROUP BY`, and `ORDER BY` and compute scores for each clause using set matching separately. On SparC, we report two metrics: question match accuracy which is the score average over all questions and interaction match accuracy which is average over all interactions.

### 5.2 Baselines

**SParC.** We compare with the two baseline models released by Yu et al. (2019b).

(1) Context-dependent Seq2Seq (CD-Seq2Seq): This model is adapted from Suhr et al. (2018). The original model was developed for ATIS and does not take the database schema as input hence cannot generalize well across domains. Yu et al. (2019b) adapt it to perform context-dependent SQL generation in multiple domains by adding a bi-LSTM database schema encoder which takes bag-of-words representations of column headers as input. They also modify the decoder to select between a SQL keyword or a column header.

(2) SyntaxSQL-con: This is adapted from the original context-agnostic SyntaxSQLNet (Yu et al., 2018b) by using bi-LSTMs to encode the interaction history including the utterance and the associated SQL query response. It also employs a column attention mechanism to compute representations of the previous question and SQL query.

**Spider.** We compare with the results as reported in Yu et al. (2018b). Furthermore, we also include recent results from Lee (2019) who propose to use recursive decoding procedure, Bogin et al. (2019) introducing graph neural networks for encoding schemas, and Guo et al. (2019) who achieve state-of-the-art performance by using an intermediate representation to bridge natural language questions and SQL queries.

| | Dev Set | Test Set |
|---|---|---|
| SQLNet (Xu et al., 2017) | 10.9 | 12.4 |
| SyntaxSQLNet (Yu et al., 2018b) | 18.9 | 19.7 |
| +data augmentation (Yu et al., 2018b) | 24.8 | 27.2 |
| Lee (2019) | 28.5 | 24.3 |
| GNN (Bogin et al., 2019) | 40.7 | 39.4 |
| IRNet (Guo et al., 2019) | 53.2 | 46.7 |
| IRNet (BERT) (Guo et al., 2019) | 61.9 | 54.7 |
| Ours | 36.4 | 32.9 |
| + utterance-table BERT Embedding | 57.6 | 53.4 |

Table 4: Spider results on dev set and test set.

### 5.3 Implementation Details

Our model is implemented in PyTorch (Paszke et al., 2017). We use pretrained 300-dimensional GloVe (Pennington et al., 2014) word embedding. All LSTM layers have 300 hidden size, and we use 1 layer for encoder LSTMs, and 2 layers for decoder LSTMs. We use the ADAM optimizer (Kingma and Ba, 2015) to minimize the token-level cross-entropy loss with a batch size of 16. Model parameters are randomly initialized from a uniform distribution $U[-0.1, 0.1]$. The main model has an initial learning rate of 0.001 and it will be multiplied by 0.8 if the validation loss increases compared with the previous epoch. When using BERT instead of GloVe, we use the pretrained small uncased BERT model with 768 hidden size[5], and we fine tune it with a separate constant learning rate of 0.00001. The training typically converges in 10 epochs.

### 5.4 Overall Results

**Spider.** Table 4 shows the results on Spider dataset. Since each question is standalone, we don't use interaction-level decoder or query editing. Our method can achieve the performance of 36.4% on dev set and 32.9% on test set, serving as a strong model for the context-independent cross-domain text-to-SQL generation. This demonstrates the effectiveness of our utterance-table encoder and table-aware decoder to handle the semantics of user utterances and the complexity of table schemas to generate complex SQL queries in unseen domains.

Furthermore, adding the utterance-table BERT

[4] More details at https://github.com/taoyds/spider/tree/master/evaluation_examples

[5] https://github.com/google-research/bert

| | Question Match | | Interaction Match | |
|---|---|---|---|---|
| | Dev | Test | Dev | Test |
| SyntaxSQL-con (Yu et al., 2019b) | 18.5 | 20.2 | 4.3 | 5.2 |
| CD-Seq2Seq (Yu et al., 2019b)* | 21.9 | 23.2 | 8.1 | 7.5 |
| + segment copy (w/ predicted query) | 21.7 | 20.3 | 9.5 | 8.1 |
| + segment copy (w/ gold query) | 27.3 | 26.7 | 10.0 | 8.4 |
| Ours | 31.4 | – | 14.7 | – |
| + query attention and sequence editing (w/ predicted query) | 33.0 | – | 16.4 | – |
| + query attention and sequence editing (w/ gold query) | 40.6 | – | 17.3 | – |
| Ours + utterance-table BERT Embedding | 40.4 | – | 18.1 | – |
| + query attention (w/ predicted query) | 42.7 | – | 21.6 | – |
| + query attention and sequence editing (w/ predicted query) | **47.2** | **47.9** | **29.5** | **25.3** |
| + query attention and sequence editing (w/ gold query) | 53.4 | 54.5 | 29.2 | 25.0 |

Table 5: SParC results. For our models, we only report test set results of our best model on the dev set. *We improve the CD-Seq2Seq performance over Yu et al. (2019b) by separating and parsing the column names (e.g., stu_fname → student first name) and using the schema-specific output vocabulary during decoding.

| | Dev Set | | | Test Set | | |
|---|---|---|---|---|---|---|
| | Query | Relaxed | Strict | Query | Relaxed | Strict |
| FULL (Suhr et al., 2018) | $37.5_{\pm 0.9}$ | $63.0_{\pm 0.7}$ | $62.5_{\pm 0.9}$ | $43.6_{\pm 1.0}$ | $69.3_{\pm 0.8}$ | $69.2_{\pm 0.8}$ |
| Ours | 36.2 | 60.5 | 60.0 | 43.9 | 68.5 | 68.1 |

Table 6: ATIS results on dev set and test set.

embedding gives significant improvement, achieving 57.6% on dev set and 53.4% on test set, which is comparable to the state-of-the-art results from IRNet with BERT. We attribute our BERT model's high performance to (1) the empirically powerful text understanding ability of pretrained BERT model and (2) the early interaction between utterances and column headers when they are concatenated in a single sequence as the BERT input.

**SParC.** Table 5 shows the results on SParC dataset. Similar to Spider, our model without previous query as input already outperforms SyntaxSQL-con, achieving 31.4% question matching accuracy and 14.7% interaction matching accuracy. In addition, compared with CD-Seq2Seq, our model enjoys the benefits of the table-utterance encoder, turn attention, and the joint consideration of utterances and table schemas during the decoding stage. This boosts the performance by 10% question accuracy and 6% interaction accuracy.

Furthermore, we also investigate the effect of copying segment. We use the same segment copy procedure as Suhr et al. (2018): first deterministically extract segments from the previous query and encode each segment using an LSTM, then generate a segment by computing its output probability based on its segment encoding. However, since the segment extraction from Suhr et al. (2018) is exclusively designed for the ATIS dataset, we implement our own segment extraction procedure by extracting `SELECT`, `FROM`, `GROUP BY`, `ORDER BY` clauses as well as different conditions in `WHERE` clauses. In this way, 3.9 segments can be extracted per SQL on average. We found that adding segment copying to CD-Seq2Seq gives a slightly lower performance on question matching and a small gain on interaction matching, while using segments extracted from the gold query can have much higher results. This demonstrates that segment copy is vulnerable to error propagation. In addition, it can only copy whole segments hence has difficulty capturing the changes of only one or a few tokens in the query.

To better understand how models perform as the interaction proceeds, Figure 4 (Left) shows the performance split by turns on the dev set. The questions asked in later turns are more difficult to answer given longer context history. While the baselines have lower performance as the turn number increases, our model still maintains 38%-48% accuracy for turn 2 and 3, and 20% at turn 4 or beyond. Similarly, Figure 4 (Right) shows the performance split by hardness levels with the frequency of examples. This also demonstrates our model is more competitive in answering hard and extra hard questions.

**ATIS.** We also report our model performance

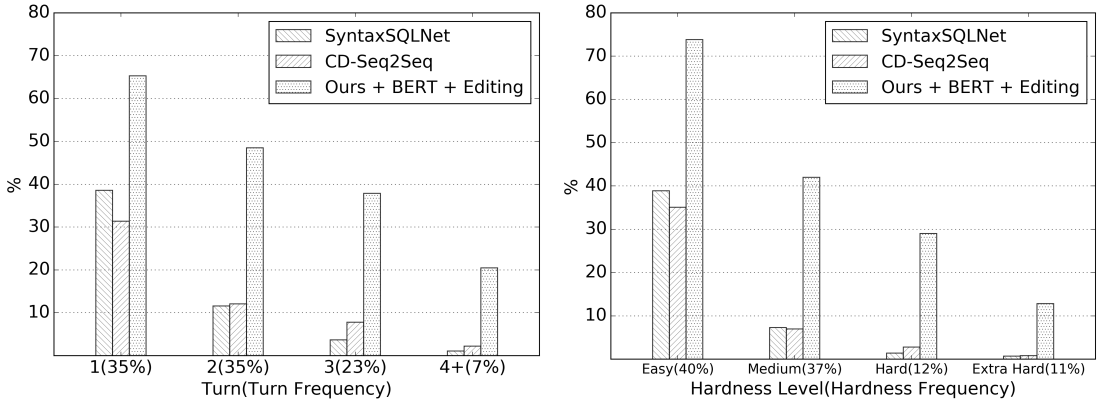


Figure 4: Performance split by different turns (Left) and hardness levels (Right) on SParC dev set.

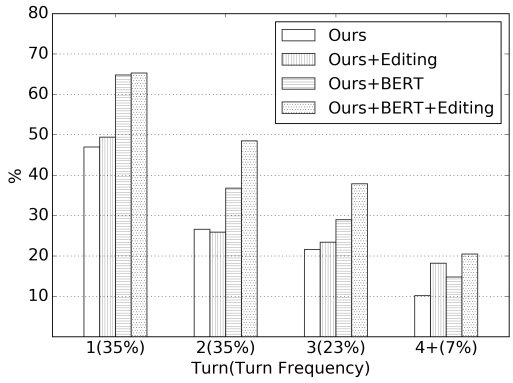


Figure 5: Effect of query editing at different turns on SParC dev set.

on ATIS in Table 6. Our model achieves 36.2% dev and 43.9% test string accuracy, comparable to Suhr et al. (2018). On ATIS, we only apply our editing mechanism and reuse their utterance encoder instead of the BERT utterance-table encoder, because ATIS is single domain.

### 5.5 Effect of Query Editing

We further investigate the effect of our query editing mechanism. To this end, we apply editing from both the gold query and the predicted query on our model with or without the utterance-table BERT embedding. We also perform an ablation study to validate the contribution of query attention and sequence editing separately.

As shown in Table 5, editing the gold query consistently improves both question match and interaction match accuracy. This shows the editing approach is indeed helpful to improve the generation quality when the previous query is the oracle.

Using the predicted query is a more realistic setting, and in this case, the model is affected by error propagation due to the incorrect queries produced by itself. For the model without the utterance-table BERT embedding, using the predicted query only gives around 1.5% improvement. As shown in Figure 5, this is because the editing mechanism is more helpful for turn 4 which is a small fraction of all question examples. For the model with the utterance-table BERT embedding, the query generation accuracy at each turn is significantly improved, thus reducing the error propagation effect. In this case, the editing approach delivers consistent improvements of 7% increase on question matching accuracy and 11% increase on interaction matching accuracy. Figure 5 also shows that query editing with BERT benefits all turns.

Finally, as an ablation study, Table 5 also reports the result with only query attention (use predicted query) on the dev set. This improves over our vanilla BERT model without query attention and achieves 42.7% question and 21.6% interaction matching accuracy. With query editing, our best model further improves to 47.2% question and 29.5% interaction matching accuracy. This demonstrates the effectiveness of our query attention and query editing separately, both of which are essential to make use of the previous query.

## 6 Conclusions

In this paper, we propose an editing-based encoder-decoder model to address the problem of context-dependent cross-domain text-to-SQL generation. While being simple, empirical results demonstrate the benefits of our editing mechanism. The approach is more robust to error propagation than copying segments, and its performance increases when the basic text-to-SQL generation quality (without editing) is better.

## Acknowledgements

We thank the anonymous reviewers for their thoughtful detailed comments.

## References


Yoav Artzi and Luke Zettlemoyer. 2011. Bootstrapping semantic parsers from conversations. In *Proceedings of the conference on empirical methods in natural language processing*, pages 421–432. Association for Computational Linguistics.

Ben Bogin, Matt Gardner, and Jonathan Berant. 2019. Representing schema structure with graph neural networks for text-to-sql parsing. In *ACL*.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *Proceedings of the fourteenth conference on computational natural language learning*, pages 18–27. Association for Computational Linguistics.

Deborah A. Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. 1994a. Expanding the scope of the atis task: The atis-3 corpus. In *Proceedings of the Workshop on Human Language Technology*, HLT '94, Stroudsburg, PA, USA. Association for Computational Linguistics.

Deborah A Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. 1994b. Expanding the scope of the atis task: The atis-3 corpus. In *Proceedings of the workshop on Human Language Technology*, pages 43–48. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving text-to-sql evaluation methodology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Daniel Fried, Jacob Andreas, and Dan Klein. 2018. Unified pragmatic models for generating and following instructions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Daya Guo, Yibo Sun, Duyu Tang, Nan Duan, Jian Yin, Hong Chi, James Cao, Peng Chen, and Ming Zhou. 2018. Question generation from sql queries improves neural semantic parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. 2019. Towards complex text-to-sql in cross-domain database with intermediate representation. In *ACL*.

Izzeddin Gur, Semih Yavuz, Yu Su, and Xifeng Yan. 2018. Dialsql: Dialogue based structured query generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Kelvin Guu, Panupong Pasupat, Evan Liu, and Percy Liang. 2017. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Tatsunori B Hashimoto, Kelvin Guu, Yonatan Oren, and Percy S Liang. 2018. A retrieve-and-edit framework for predicting structured outputs. In *Advances in Neural Information Processing Systems*, pages 10052–10062.

Shirley Anugrah Hayati, Raphael Olivier, Pravalika Avvaru, Pengcheng Yin, Anthony Tomasic, and Graham Neubig. 2018. Retrieval-based neural code generation. In *EMNLP*.

Charles T Hemphill, John J Godfrey, and George R Doddington. 1990. The atis spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*.

Hsin-Yuan Huang, Eunsol Choi, and Wen-tau Yih. 2019. Flowqa: Grasping flow in history for conversational machine comprehension. In *ICLR*.

Po-Sen Huang, Chenglong Wang, Rishabh Singh, Wen-tau Yih, and Xiaodong He. 2018. Natural language to structured query generation via meta-learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*.

Wonseok Hwang, Jinyeung Yim, Seunghyun Park, and Minjoon Seo. 2019. A comprehensive exploration on wikisql with table-aware word contextualization. *arXiv preprint arXiv:1902.01069*.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. 2018. Mapping language to code in programmatic context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Dongjun Lee. 2019. Recursive and clause-wise decoding for complex and cross-domain text-to-sql generation. In *CoRR*.

Reginald Long, Panupong Pasupat, and Percy Liang. 2016. Simpler context-dependent logical forms via model projections. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*.

Scott Miller, David Stallard, Robert Bobrow, and Richard Schwartz. 1996. A fully statistical approach to natural language interfaces. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NIPS 2017 Workshop Autodiff*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Tianze Shi, Kedar Tatwawadi, Kaushik Chakrabarti, Yi Mao, Oleksandr Polozov, and Weizhu Chen. 2018. Incsql: Training incremental text-to-sql parsers with non-deterministic oracles. *arXiv preprint arXiv:1809.05054*.

Yu Su and Xifeng Yan. 2017. Cross-domain semantic parsing via paraphrasing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Alane Suhr and Yoav Artzi. 2018. Situated mapping of sequential instructions to actions with single-step reward observation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Alane Suhr, Srinivasan Iyer, and Yoav Artzi. 2018. Learning to map context-dependent sentences to executable formal queries. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*.

Yibo Sun, Duyu Tang, Nan Duan, Jianshu Ji, Guihong Cao, Xiaocheng Feng, Bing Qin, Ting Liu, and Ming Zhou. 2018. Semantic parsing with syntax- and table-aware sql generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Andreas Vlachos and Stephen Clark. 2014. A new corpus and imitation learning framework for context-dependent semantic parsing. *Transactions of the Association for Computational Linguistics*.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Xiaojun Xu, Chang Liu, and Dawn Song. 2017. Sqlnet: Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv:1711.04436*.

Semih Yavuz, Izzeddin Gur, Yu Su, and Xifeng Yan. 2018. What it takes to achieve 100% condition accuracy on wikisql. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Pengcheng Yin, Graham Neubig, Miltiadis Allamanis, Marc Brockschmidt, and Alexander L Gaunt. 2019. Learning to represent edits. In *ICLR*.

Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. 2018a. Typesql: Knowledge-based type-aware neural text-to-sql generation. In *Proceedings of NAACL*. Association for Computational Linguistics.

Tao Yu, Michihiro Yasunaga, Kai Yang, Rui Zhang, Dongxu Wang, Zifan Li, and Dragomir Radev. 2018b. Syntaxsqlnet: Syntax tree networks for complex and cross-domain text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Tao Yu, Rui Zhang, He Yang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, Youxuan Jiang, Michihiro Yasunaga, Sungrok Shim, Tao Chen, Alexander Fabbri, Zifan Li, Luyao Chen, Yuwen Zhang, Shreya Dixit, Vincent Zhang, Caiming Xiong, Richard Socher, Walter Lasecki, and Dragomir Radev. 2019a. Cosql: A conversational text-to-sql challenge towards cross-domain natural language interfaces to databases. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing*. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018c. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Tao Yu, Rui Zhang, Michihiro Yasunaga, Yi Chern Tan, Xi Victoria Lin, Suyi Li, Heyang Er, Irene Li, Bo Pang, Tao Chen, Emily Ji, Shreya Dixit, David Proctor, Sungrok Shim, Jonathan Kraft, Vincent Zhang, Caiming Xiong, Richard Socher, and Dragomir Radev. 2019b. Sparc: Cross-domain semantic parsing in context. In *Proceedings of ACL*.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *AAAI/IAAI*, pages 1050–1055, Portland, OR. AAAI Press/MIT Press.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *UAI*.

Luke S Zettlemoyer and Michael Collins. 2009. Learning context-dependent mappings from sentences to logical form. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 976–984. Association for Computational Linguistics.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.

# Syntax-aware Multilingual Semantic Role Labeling

**Shexia He**[1,2,3,*], **Zuchao Li**[1,2,3,*], **Hai Zhao**[1,2,3,†]
[1]Department of Computer Science and Engineering, Shanghai Jiao Tong University
[2]Key Laboratory of Shanghai Education Commission for Intelligent Interaction
and Cognitive Engineering, Shanghai Jiao Tong University, Shanghai, China
[3]MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University
{heshexia, charlee}@sjtu.edu.cn, zhaohai@cs.sjtu.edu.cn

## Abstract

Recently, semantic role labeling (SRL) has earned a series of success with even higher performance improvements, which can be mainly attributed to syntactic integration and enhanced word representation. However, most of these efforts focus on English, while SRL on multiple languages more than English has received relatively little attention so that is kept underdevelopment. Thus this paper intends to fill the gap on multilingual SRL with special focus on the impact of syntax and contextualized word representation. Unlike existing work, we propose a novel method guided by syntactic rule to prune arguments, which enables us to integrate syntax into multilingual SRL model simply and effectively. We present a unified SRL model designed for multiple languages together with the proposed uniform syntax enhancement. Our model achieves new state-of-the-art results on the CoNLL-2009 benchmarks of all seven languages. Besides, we pose a discussion on the syntactic role among different languages and verify the effectiveness of deep enhanced representation for multilingual SRL.

## 1 Introduction

Semantic role labeling (SRL) aims to derive the meaning representation such as an instantiated predicate-argument structure for a sentence. The currently popular formalisms to represent the semantic predicate-argument structure are based on dependencies and spans. Their main difference is that dependency SRL annotates the syntactic head of argument rather than the entire constituent (span), and this paper will focus on the dependency-based SRL. Be it dependency or span, SRL plays a critical role in many natural language processing (NLP) tasks, including information extraction (Christensen et al., 2011), machine translation (Xiong et al., 2012) and question answering (Yih et al., 2016).

Almost all of traditional SRL methods relied heavily on syntactic features, which suffered the risk of erroneous syntactic input, leading to undesired error propagation. To alleviate this inconvenience, researchers as early as Zhou and Xu (2015) propose neural SRL models without syntactic input. Cai et al. (2018) employ the biaffine attentional mechanism (Dozat and Manning, 2017) for dependency-based SRL. In the meantime, a series of studies (Roth and Lapata, 2016; Marcheggiani and Titov, 2017; Strubell et al., 2018; Li et al., 2018) have introduced syntactic clue in creative ways for further performance improvement, which achieve favorable results. However, applying the $k$-order syntactic tree pruning of He et al. (2018) to the biaffine SRL model (Cai et al., 2018) does not boost the performance as expected, which indicates that exploiting syntactic clue in state-of-the-art SRL models still deserves deep exploration.

Besides, most of SRL literature is dedicated to impressive performance gains on English and Chinese, but other multiple languages have received relatively little attention. We even observe that to date the best reported results of some languages (Catalan and Japanese) are still from the initial CoNLL-2009 shared task (Hajič et al., 2009). Therefore, we launch this multilingual SRL study to fill the obvious gap ignored since a long time ago. Especially, we attempt to improve the overall performance of multilingual SRL by incorporating syntax and introducing contextualized word representation, and explore syntactic effect on other multiple languages.

Multilingual SRL needs to be carefully han-

* These authors made equal contribution. † Corresponding author. This paper was partially supported by National Key Research and Development Program of China (No. 2017YFB0304100) and Key Projects of National Natural Science Foundation of China (No. U1836222 and No. 61733011).



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5350–5359,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

dled for the diversity of syntactic and semantic representations among quite different languages. Despite such a diversity, in this paper, we manage to develop a simple and effective neural SRL model to integrate syntactic information by applying argument pruning method in a uniform way. Specifically, we introduce new pruning rule based on syntactic parse tree unlike the $k$-order pruning of He et al. (2018), which is only simply determined by the relative distance of predicate and argument. Furthermore, we propose a novel method guided by syntactic rule to prune arguments for dependency SRL, different from the existing work. With the help of the proposed pruning method, our model can effectively alleviate the imbalanced distribution of arguments and non-arguments, achieving faster convergence during training.

To verify the effectiveness and applicability of the proposed method, we evaluate the model on all seven languages of CoNLL-2009 datasets. Experimental results indicate that our argument pruning method is generally effective for multilingual SRL over our unified modeling. Moreover, our model using contextualized word representation achieves the new best results on all seven datasets, which is the first overall update since 2009. To the best of our knowledge, this is the first attempt to study seven languages comprehensively in deep learning models.

## 2 Model

Given a sentence, SRL can be decomposed into four classification subtasks, predicate identification and disambiguation, argument identification and classification. Since the CoNLL-2009 shared task has indicated all predicates beforehand, we focus on identifying arguments and labeling them with semantic roles. Our model builds on a recent syntax-agnostic SRL model (Cai et al., 2018) by introducing argument pruning and enhanced word representation. In this work, we handle argument identification and classification in one shot, treating the SRL task as word predicate-argument pair classification. Figure 1 illustrates the overall architecture of our model, which consists of three modules, (1) a bidirectional LSTM (BiLSTM) encoder, (2) an argument pruning layer which takes as input the BiLSTM representations, and (3) a biaffine scorer which takes as input the predicate and its argument candidates.

### 2.1 BiLSTM Encoder

Given a sentence and marked predicates, we adopt the bidirectional Long Short-term Memory neural network (BiLSTM) (Hochreiter and Schmidhuber, 1997) to encode sentence, which takes as input the word representation. Following Cai et al. (2018), the word representation is the concatenation of five vectors: randomly initialized word embedding, lemma embedding, part-of-speech (POS) tag embedding, pre-trained word embedding and predicate-specific indicator embedding.

Besides, the latest work (Li et al., 2018) has demonstrated that the contextualized representation ELMo (Embeddings from Language Models) (Peters et al., 2018) could boost performance of dependency SRL model on English and Chinese. To explore whether the deep enhanced representation can help other multiple languages, we further enhance the word representation by concatenating an external embedding from the recent successful language models, ELMo and BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2018), which are both contextualized representations. It is worth noting that we use ELMo or BERT to obtain pre-trained contextual embeddings rather than fine-tune the model, which are fixed contextual representations.

### 2.2 Argument Pruning Layer

For word pair classification modeling, one major performance bottleneck is caused by unbalanced data, especially for SRL, where more than 90% of argument candidates are non-arguments. A series of pruning methods are then proposed to alleviate the imbalanced distribution, such as the $k$-order pruning (He et al., 2018). However, it does not extend well to other languages, and even hinders the syntax-agnostic SRL model as Cai et al. (2018) has experimented with different $k$ values on English. The reason might be that this pruning method breaks up the whole sentence, leading the BiLSTM encoder to take the incomplete sentence as input and fail to learn sentence representation sufficiently.

To alleviate such a drawback from the previous syntax-based pruning methods, we propose a novel pruning rule extraction method based on syntactic parse tree, which generally suits multilingual cases at the same time. In detailed model implementation, we add an argument pruning layer guided by syntactic rule following BiL-

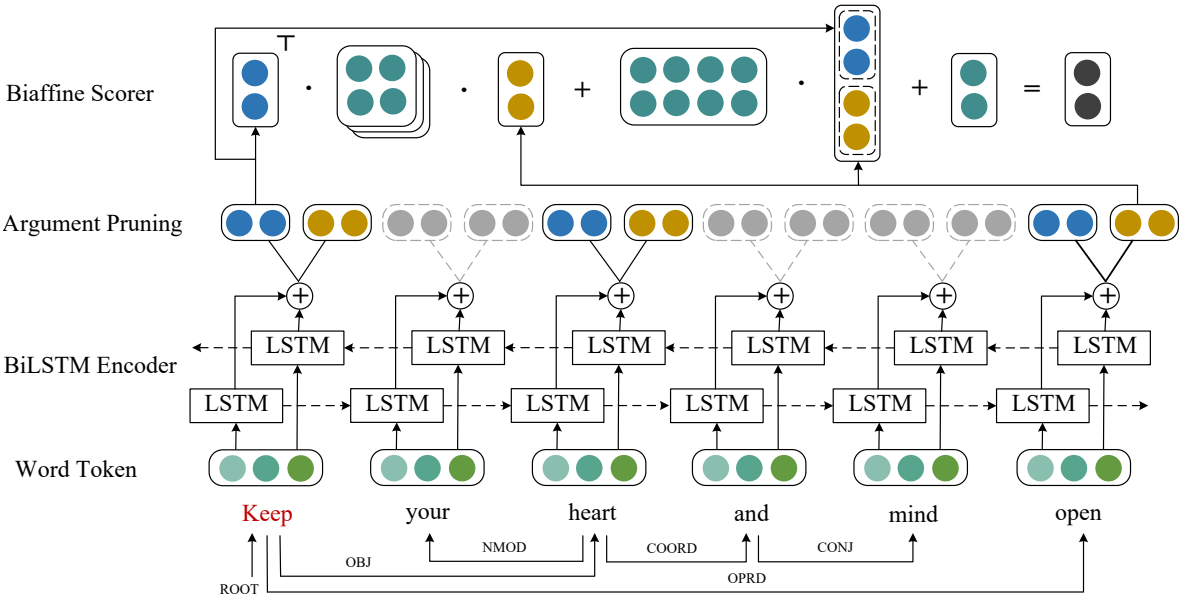


Figure 1: Overall architecture of our SRL model. Red denotes the given predicate, and gray indicates that these units are dropped according to syntactic rule. The bottom is syntactic dependency.

STM layers, which can absorb the syntactic clue simply and effectively.

**Syntactic Rule** Considering that all arguments are predicate-specific instances, it has been generally observed that the distances between predicate and its arguments on syntactic tree are within a certain range for most languages. Therefore, we introduce language-specific rule based on syntactic dependency parses to prune some unlikely arguments, henceforth syntactic rule. Specifically, given a predicate $p$ and its argument $a$, we define $d_p$ and $d_a$ to be the distance from $p$ and $a$ to their nearest common ancestor node (namely, the root of the minimal subtree which includes $p$ and $a$) respectively. For example, 0 denotes that predicate or argument itself is their nearest common ancestor, while 1 represents that their nearest common ancestor is the parent of predicate or argument. Then we use the distance tuple $(d_p, d_a)$ as their relative position representation inside the parse tree. Finally, we make a list of all tuples ordered according to how many times that each distance tuple occurs in the training data, which is counted for each language independently.

It is worth noting that our syntactic rule is determined by the top-$k$ frequent distance tuples. During training and inference, the syntactic rule takes effect by excluding all candidate arguments whose predicate-argument relative position in parse tree is not in the list of top-$k$ frequent tuples.

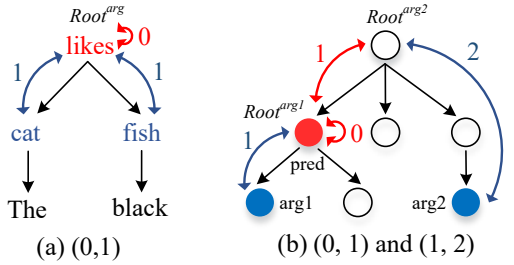


Figure 2: Syntactic parse tree examples (dependency relations are omitted). Red represents the current predicate, and blue indicates its arguments.

Figure 2 shows simplified examples of syntactic dependency tree. Given an English sentence in (a), the current predicate is *likes*, whose arguments are *cat* and *fish*. For *likes* and *cat*, the predicate (*likes*) is their common ancestor (denoted as $Root^{arg}$) according to the syntax tree. Therefore, the relative position representation of predicate and argument is $(0, 1)$, so it is for *likes* and *fish*. As for the right one in (b), suppose the marked predicate has two arguments$-arg1$ and $arg2$, the common ancestors of predicate and arguments are respectively $Root^{arg1}$ and $Root^{arg2}$. In this case, the relative position representations are $(0, 1)$ and $(1, 2)$.

**Argument Pruning Method** To maintain the integrity of sequential inputs from the whole sentence, we propose a novel syntax-based method to prune arguments, unlike most existing work (Xue and Palmer, 2004; Zhao et al., 2009a; He et al.,

2018) which prunes argument candidates in the pre-processing stage. As shown in Figure 1, the way to perform argument pruning strategy is very straightforward. In the argument pruning layer, our model drops these candidate arguments (more exactly, BiLSTM representations) which do not comply with the syntactic rule. In other words, only the predicates and arguments that satisfy the syntactic rule will be output to next layer.

For example (in Figure 1), given the sentence *Keep your heart and mind open*, the predicate *Keep* and the corresponding syntactic dependency (bottom), by definition, $(0, 1)$ is inside the syntactic rule on this occasion. Therefore, these candidate arguments (i.e., *your*, *and*, *mind*) will be pruned by the argument pruning layer.

### 2.3 Biaffine Scorer

As mentioned above, our model treats SRL task as a word pair classification problem, tackling argument identification and classification in one shot. To label arguments of given predicates, we employ a scorer with biaffine attention (Dozat and Manning, 2017) (biaffine scorer for short) as role classifier on top of argument pruning layer for the final prediction, similar to Cai et al. (2018). Biaffine scorer takes as input the BiLSTM hidden states of predicate and candidate arguments filtered by argument pruning layer, denoted by $h_p$ and $h_a$ respectively, and then computes the probability of corresponding semantic labels using biaffine transformation as follows:

$$\begin{aligned}\Phi_r(p, a) &= \textit{Biaffine}(h_p, h_a) \\ &= \{h_p\}^T \mathbf{W}_1 h_a + \mathbf{W}_2^T (h_p \oplus h_a) + \mathbf{b}\end{aligned}$$

where $\oplus$ represents concatenation operator, $\mathbf{W}_1$ and $\mathbf{W}_2$ denote the weight matrix of the bilinear and the linear terms respectively, and $\mathbf{b}$ is the bias item. Note that the predicate itself is also included in its own argument candidate list and will be applied to compute scores, because a nominal predicate sometimes takes itself as its own argument.

## 3 Experiments

Our model[1] is evaluated on the CoNLL-2009 benchmark datasets, including Catalan, Chinese, Czech, English, German, Japanese and Spanish. The statistics of the training datasets can be seen in Table 1. For the predicate disambiguation task, we follow previous work, using models (Zhao et al., 2009a) for Catalan and Spanish, and the ones (Björkelund et al., 2009) for other languages. Besides, we use the officially predicted POS tags and syntactic parses provided by CoNLL-2009 shared-task for all languages.[2] As for the contextualized representation, ELMo, we employ the multilingual version from Che et al. (2018). For BERT, this work uses the BERT-Base, Multilingual Cased model (Devlin et al., 2018). For syntactic rule in argument pruning layer, to ensure more than 99% coverage of true arguments in pruning output, we use the top-120 distance tuples on Japanese and top-20 on other multiple languages for a better trade-off between computation and coverage.

| Dataset | #sent | #token | #pred | #arg |
|---|---|---|---|---|
| Catalan | 13,200 | 390,302 | 37,431 | 84,367 |
| Chinese | 22,277 | 609,060 | 102,813 | 231,869 |
| Czech | 38,727 | 652,544 | 414,237 | 365,255 |
| English | 39,279 | 958,167 | 179,014 | 393,699 |
| German | 36,020 | 648,677 | 17,400 | 34,276 |
| Japanese | 4,393 | 112,555 | 25,712 | 43,957 |
| Spanish | 14,329 | 427,442 | 43,824 | 99,054 |

Table 1: Training data statistics of sentences, tokens, predicates and arguments. # denotes numbers.

### 3.1 Model Setup

In our experiments, all real vectors are randomly initialized, including 100-dimensional word, lemma, POS tag embeddings and 16-dimensional predicate-specific indicator embedding (He et al., 2018). The pre-trained word embedding is 100-dimensional GloVe vectors (Pennington et al., 2014) for English, 300-dimensional fastText vectors (Grave et al., 2018) trained on Common Crawl and Wikipedia for other languages, while the dimension of ELMo or BERT word embedding is 1024. Besides, we use 3 layers BiLSTM with 400-dimensional hidden states, applying dropout with an 80% keep probability between time-steps and layers. For biaffine scorer, we employ two 300-dimensional affine transfor-

[1]The code is available at https://github.com/bcmi220/multilingual_srl.

[2]There were two tracks in the CoNLL-2009 shared task, SRL-only and joint. For the former, all participants did not have to develop their own syntactic parsers and focused on the SRL model development, while for the latter, the participants had to build their own syntactic parser as well. For the sake of focusing the SRL work, in this work, we will take the official syntax provided by CoNLL-2009.

| Model | English | | | Chinese | | |
|---|---|---|---|---|---|---|
| | P | R | $F_1$ | P | R | $F_1$ |
| Zhao et al. (2009a) | – | – | 86.2 | 80.4 | 75.2 | 77.7 |
| Björkelund et al. (2009) | 88.6 | 85.2 | 86.9 | 82.4 | 75.1 | 78.6 |
| FitzGerald et al. (2015) | – | – | 87.3 | – | – | – |
| Roth and Lapata (2016) | 90.0 | 85.5 | 87.7 | 83.2 | 75.9 | 79.4 |
| Marcheggiani et al. (2017) | 88.7 | 86.8 | 87.7 | 83.4 | 79.1 | 81.2 |
| Marcheggiani and Titov (2017) | 89.1 | 86.8 | 88.0 | 84.6 | 80.4 | 82.5 |
| He et al. (2018) (with ELMo) | 89.7 | 89.3 | 89.5 | 84.2 | 81.5 | 82.8 |
| Cai et al. (2018) | 89.9 | 89.2 | 89.6 | 84.7 | 84.0 | 84.3 |
| Li et al. (2018) (with ELMo) | 90.3 | 89.3 | 89.8 | 84.8 | 81.2 | 83.0 |
| Li et al. (2019) (with ELMo) | 89.6 | 91.2 | 90.4 | – | – | – |
| Our baseline | 89.30 | 89.93 | 89.61 | 82.88 | 85.26 | 84.05 |
| + AP | 89.96 | 89.96 | 89.96 | 84.60 | 84.50 | 84.55 |
| + BERT | 89.80 | 91.20 | 90.50 | 85.76 | 86.50 | 86.13 |
| + AP + ELMo | 90.00 | 90.65 | 90.32 | 84.44 | 84.95 | 84.70 |
| + AP + BERT | **90.41** | **91.32** | **90.86** | **86.15** | **86.70** | **86.42** |

Table 2: Precision, recall and semantic $F_1$-score on CoNLL-2009 English in-domain data and Chinese test set.

mations with the ReLU non-linear activation, also setting the dropout probability to 0.2. During training, we use the categorical cross-entropy as objective, with Adam optimizer (Kingma and Ba, 2015) initial learning rate $2e^{-3}$. All models are trained for up to 500 epochs with batch size 64.

### 3.2 Results and Discussion

In Table 2, we compare our single model (**AP** is an acronym for argument pruning) against previous work on English in-domain data and Chinese test set. Our baseline is a modification to the model of Cai et al. (2018) which uniformly handled the predicate disambiguation. For English, our baseline gives slightly weaker performance than the work of Li et al. (2019), which used ELMo and employed a sophisticated span selection model for predicting predicates and arguments jointly. Our model with the proposed argument pruning layer (**+ AP**) brings absolute improvements of 0.35% and 0.5% $F_1$ on English and Chinese, respectively, which is on par with the best published scores. Moreover, we introduce deep enhanced representation based on the argument pruning. Our model utilizing BERT (**+ AP + BERT**) achieves the new best results on English and Chinese benchmarks.

Table 3 presents all test results on seven languages of CoNLL-2009 datasets. So far, the best previously reported results of Catalan, Japanese and Spanish are still from CoNLL-2009 shared task. Compared with previous methods, our baseline yields strong performance on all datasets except German. Especially for Catalan, Czech, Japanese and Spanish, our baseline performs better than existing methods with a large margin of 3.5% $F_1$ on average. Nevertheless, applying our argument pruning to the strong syntax-agnostic baseline can still boost the model performance, which demonstrates the effectiveness of proposed method. On the other hand, it indicates that syntax is generally beneficial to multiple languages, and can enhance the multilingual SRL performance with effective syntactic integration.

Besides, we report the scores of leveraging ELMo and BERT for multiple languages (the last three rows in Table 3). The use of contextualized word representation further improves model performance, which overwhelmingly outperforms previously published best results and achieves the new state of the art in multilingual SRL for the first time. Furthermore, we find that ELMo promotes the overall performance of SRL model, but BERT gives more significant performance increase than ELMo on all languages, which suggests that BERT is better at enriching contextual information. More interestingly, we observe that the performance gains from the proposed argument pruning method or these deep enhanced representations are relatively marginal on Japanese, one possible reason is the relatively small size of

| Model | Catalan | Chinese | Czech | English | German | Japanese | Spanish |
|---|---|---|---|---|---|---|---|
| CoNLL-2009 ST best system | 80.3 | 78.6 | 85.4 | 85.6 | 79.7 | 78.2 | 80.5 |
| Zhao et al. (2009a) | 80.3 | 77.7 | 85.2 | 86.2 | 76.0 | 78.2 | 80.5 |
| Roth and Lapata (2016) | – | 79.4 | – | 87.7 | 80.1 | – | 80.2 |
| Marcheggiani et al. (2017) | – | 81.2 | 86.0 | 87.7 | – | – | 80.3 |
| Li et al. (2019) | – | – | – | 90.4 | – | – | – |
| The best previously published | 80.3 | 84.3 | 86.0 | 90.4 | 80.1 | 78.2 | 80.5 |
| Our baseline | 84.07 | 84.05 | 88.35 | 89.61 | 78.36 | 83.08 | 83.47 |
| + AP | 84.35 | 84.55 | 88.76 | 89.96 | 78.54 | 83.12 | 83.70 |
| + BERT | 84.88 | 86.13 | 89.06 | 90.50 | 80.68 | 83.57 | 84.50 |
| + AP + ELMo | 84.35 | 84.70 | 89.52 | 90.32 | 78.65 | 83.43 | 83.82 |
| + AP + BERT | **85.14** | **86.42** | **89.66** | **90.86** | **80.87** | **83.76** | **84.60** |

Table 3: Semantic $F_1$-score on CoNLL-2009 in-domain test set. The first row is the best result of CoNLL-2009 shared task (Hajič et al., 2009). The previously best published results of Catalan and Japanese is from Zhao et al. (2009a), Chinese from Cai et al. (2018), Czech from Marcheggiani et al. (2017), English from Li et al. (2019), German and Spanish from Roth and Lapata (2016).

| Model | Catalan | | Chinese | | English | | German | | Spanish | |
|---|---|---|---|---|---|---|---|---|---|---|
| | PD | $F_1$ | PD | $F_1$ | PD | $F_1$ | PD | $F_1$ | PD | $F_1$ |
| Our baseline | 87.50 | 84.07 | 94.92 | 84.05 | 95.59 | 89.61 | 81.45 | 78.36 | 86.53 | 83.47 |
| *Biaffine SRL* | 89.10 | 84.70 | 95.60 | 84.56 | 95.04 | 89.60 | 81.64 | 78.45 | 87.44 | 83.85 |
| + AP | 89.52 | 84.90 | 95.60 | 84.76 | 95.38 | 89.88 | 81.65 | 78.50 | 87.56 | 83.92 |
| + AP + BERT | 90.08 | **86.04** | 96.17 | **86.90** | 96.37 | **91.00** | 82.36 | **81.14** | 88.27 | **85.15** |

Table 4: Results of full end-to-end model. PD denotes the accuracy of predicate disambiguation. + AP represents *biaffine SRL+Argument Pruning Layer*, while the last row indicates *biaffine SRL+Argument Pruning Layer+BERT*.

its training set. This observation also indicates that ELMo or BERT is more suitable for learning on large annotated corpus.

### 3.3 End-to-end SRL

As mentioned above, we combine the predicate sense output of previous work to make results directly comparable, since the official evaluation script includes such prediction in the $F_1$-score calculation. However, predicate disambiguation is considered a simpler task with higher semantic $F_1$-score and deserves more further research. To this end, we present a full end-to-end neural model for multilingual SRL, namely *Biaffine SRL*, following Cai et al. (2018).

Unlike most of SRL work treating the predicate sense disambiguation and semantic role assignment tasks as independent, we jointly handle predicate disambiguation and argument labeling in one shot by introducing a virtual node $<VR>$ as the nominal semantic head of predicate. It should be noted that the predicate sense annotation of Czech and Japanese is simply the lemmatized token of the predicate, a one-to-one predicate-sense mapping. Therefore, we ignore them and conduct experiments on other five languages.

Table 4 shows the results of end-to-end setting. Compared to the baseline, our full end-to-end model (*Biaffine SRL*) yields slightly higher precision of predicate disambiguation as a whole, which gives rise to a corresponding gain of semantic $F_1$. What is more, our model (using argument pruning and BERT) reaches the highest scores on the five benchmarks. Besides, experiments indicate that argument pruning promotes role labeling performance while BERT significantly improves the performance of predicate disambiguation.

## 4 Analysis

In this section, we perform further analysis to better understand our model, exploring the impact of language features, syntactic rule and syntac-

| Dataset | baseline | w/o POS tag | w/o lemma |
|---|---|---|---|
| Catalan | 84.07 | 83.83 (-0.24) | 83.60 (-0.47) |
| Czech | 88.35 | 88.10 (-0.25) | 88.20 (-0.15) |
| German | 78.36 | 77.80 (-0.56) | 78.12 (-0.24) |
| Japanese | 83.08 | 82.02 (-1.06) | 82.80 (-0.28) |
| Spanish | 83.47 | 83.15 (-0.32) | 83.00 (-0.47) |

Table 5: Ablation of POS tag and lemma on test set.

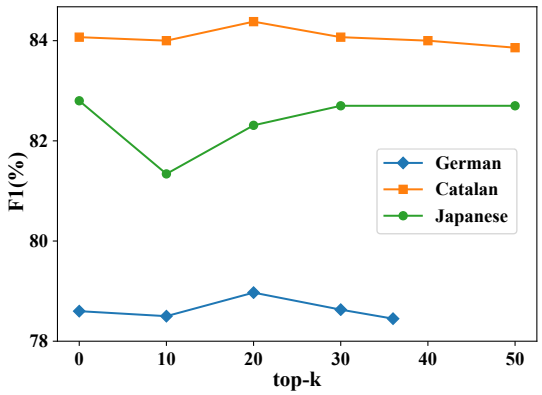


Figure 3: $F_1$ scores on test set by top-$k$ argument pruning for German, Catalan and Japanese.

| Dataset | syntactic rule | $k$-order | $\Delta$ $F_1$ |
|---|---|---|---|
| Catalan | 84.35 | 84.18 | −0.17 |
| Czech | 88.76 | 88.75 | −0.01 |
| German | 78.54 | 78.42 | −0.12 |
| Japanese | 83.12 | 82.61 | −0.51 |
| Spanish | 83.70 | 83.53 | −0.17 |

Table 6: Comparison of our model with syntactic rule and $k$-order argument pruning.

| Dataset | syntax-agnostic | syntax-aware | |
|---|---|---|---|
| | | predicted (UAS) | *gold* |
| Catalan | 84.07 | 84.35 (89.43) | *85.50* |
| Czech | 88.35 | 88.76 (85.69) | *88.92* |
| German | 78.36 | 78.54 (88.91) | *78.56* |
| Japanese | 83.08 | 83.12 (92.29) | *83.20* |
| Spanish | 83.47 | 83.70 (89.39) | *84.82* |

Table 7: Syntactic contribution to multilingual SRL. *predicted* and *gold* denote the use of syntactic parse. The UAS of predicted syntax is in parenthesis.

tic contribution for multilingual SRL. Since recent work well studied dependency SRL on English and Chinese, we focus on other five languages, and these analyses are performed on CoNLL-2009 test sets without using ELMo or BERT embeddings.

### 4.1 Effectiveness of Language Feature

As Marcheggiani et al. (2017) point out, POS tag information is highly beneficial for English. Consequently, we conduct an ablation study on in-domain test set to explore how the language features impact our model. Table 5 reports the $F_1$ scores of model which removes POS tag or lemma from the baseline. Results show that omitting POS tag or lemma leads to slight performance degradation ($-0.5\%$ and $-0.32\%$ $F_1$ on average, respectively), indicating that both can help improve performance of multilingual SRL. Interestingly, we see a drop of 1.0% $F_1$ for Japanese not using POS tag, which demonstrates its importance.

### 4.2 Effectiveness of Syntactic Rule

According to our statistics of syntactic rule on training data for seven languages, which is based on the automatically predicted parse provided by CoNLL-2009 shared task, the total number of distance tuples in syntactic rule is no more than 120 in these languages except that Japanese is about 260. It is in favor of previous hypothesis that the relative distances between predicate and its arguments are within a certain range. Besides, $(0, 1)$ is the most frequently occurring relationships in all languages except Japanese, which indicates that arguments most frequently appear to be the children of their predicate in dependency syntax tree.

Figure 3 shows $F_1$ scores on test set by top-$k$ argument pruning for German, Catalan and Japanese,[3] where $k$ = 0 represents the baseline without pruning. We observe that the case of $k$ = 20 yields the best performance on German and Catalan. As for Japanese, it falls short of the baseline in our current observation range, but our experiment has shown that the setting of top-120 can achieve the best results.

To reveal the strengths of proposed syntactic rule, we conduct further experiments, replacing our syntactic rule based pruning with the $k$-order argument pruning of He et al. (2018). Following their setting, we use the tenth-order pruning for pursuing the best performance. Table 6 shows the performance gaps between two pruning methods. Comparing with syntactic rule, the $k$-order prun-

[3]Note that the maximum values of $k$ for German, Catalan and Japanese are 36, 50 and 260, respectively. We thus show only the top 50 for Japanese due to the limited space.

ing declines model performance by 0.2% $F_1$ on average, showing that our syntactic rule based pruning method is more effective and can be extended well to multiple languages especially for Japanese.

### 4.3 Syntactic Impact

In this part, we attempt to explore the syntactic impact on other five languages. To investigate the most contribution of syntax to multilingual SRL, we perform experiments using the gold syntactic parse also officially provided by the CoNLL-2009 shared task instead of the predicted one.[4] To be more precise, the syntactic rule is counted based on gold syntactic tree and applied to argument pruning layer. The corresponding results of our syntax-agnostic and syntax-aware models are summarized in Table 7. We also report the unlabeled attachment scores (UAS) of predicted syntax as syntactic accuracy measurement, considering that we do not use the dependency labels.

Results indicate that high-quality syntax can further improve model performance, showing syntactic information is generally effective for multilingual SRL. In particular, based on gold syntax, the top-1 argument pruning for Catalan and Spanish has reached 100 percent coverage (namely, for Catalan and Spanish, all arguments are the children of predicates in gold dependency syntax tree), and hence our syntax-aware model obtains significant gains of 1.43% and 1.35%, respectively.

In addition, combining the results of Tables 6 and 7, we find that applying the $k$-order argument pruning of He et al. (2018) to syntax-agnostic model results in better performance on most languages. However, Cai et al. (2018) argue that $k$-order pruning does not boost the performance for English. One reason to account for this finding is the lack of effective approaches for incorporating syntactic information into sequential neural networks. Nevertheless, syntactic contribution is overall limited for multilingual SRL in this work, due to strong syntax-agnostic baseline. Therefore, more effective methods to incorporate syntax into neural SRL model are worth exploring. Besides, utilization of syntactic dependency labels information is also a promising direction and we leave it for future work.

[4] In this work, we use gold syntax rather than other better parse to explore the greatest syntactic contribution, considering the current state-of-the-art syntactic parsers are being upgraded so fast now.

## 5 Related Work

In early work of semantic role labeling, most of researchers were dedicated to feature engineering (Pradhan et al., 2005; Punyakanok et al., 2008; Zhao et al., 2009b, 2013). The first neural SRL model was proposed by Collobert et al. (2011), which used convolutional neural network but their efforts fell short. Later, Foland and Martin (2015) effectively extended their work by using syntactic features as input. Roth and Lapata (2016) introduced syntactic paths to guide neural architectures for dependency SRL.

However, putting syntax aside has sparked much research interest since Zhou and Xu (2015) employed deep BiLSTMs for span SRL. A series of neural SRL models without syntactic inputs were proposed. Marcheggiani et al. (2017) applied a simple LSTM model with effective word representation, achieving encouraging results on English, Chinese, Czech and Spanish. Cai et al. (2018) built a full end-to-end SRL model with biaffine attention and provided strong performance on English and Chinese. Li et al. (2019) also proposed an end-to-end model for both dependency and span SRL with a unified argument representation, obtaining favorable results on English.

Despite the success of syntax-agnostic SRL models, more recent work attempts to further improve performance by integrating syntactic information, with the impressive success of deep neural networks in dependency parsing (Zhang et al., 2016; Zhou and Zhao, 2019). Marcheggiani and Titov (2017) used graph convolutional network to encode syntax into dependency SRL. He et al. (2018) proposed an extended $k$-order argument pruning algorithm based on syntactic tree and boosted SRL performance. Li et al. (2018) presented a unified neural framework to provide multiple methods for syntactic integration. Our method is closely related to the one of He et al. (2018), designed to prune as many unlikely arguments as possible.

**Multilingual SRL** To promote NLP applications, the CoNLL-2009 shared task advocated performing SRL for multiple languages. Among the participating systems, Zhao et al. (2009a) proposed an integrated approach by exploiting large-scale feature set, while Björkelund et al. (2009) used a generic feature selection procedure. Until now, only a few of work (Lei et al., 2015;

Swayamdipta et al., 2016; Mulcaire et al., 2018) seriously considered multilingual SRL. Among them, Mulcaire et al. (2018) built a polyglot model (training one model on multiple languages) for multilingual SRL, but their results were far from satisfactory. Therefore, this work aims to complete the overall upgrade since CoNLL-2009 shared task and leaves polyglot training as our future work.

## 6 Conclusion

This paper is dedicated to filling the long-term performance gap of multilingual SRL since a long time ago with a newly proposed syntax-based argument pruning method. Experimental results demonstrate its effectiveness and shed light on a new perspective for many NLP tasks to incorporate syntax simply and effectively. Besides, our model substantially boosts multilingual SRL performance by introducing deep enhanced representation, achieving new state-of-the-art results on the in-domain CoNLL-2009 benchmark for Catalan, Chinese, Czech, English, German, Japanese and Spanish. These results further show that syntactic information and deep enhanced representation can also promote multiple languages rather than only the case of English.

## References


Anders Björkelund, Love Hafdell, and Pierre Nugues. 2009. Multilingual semantic role labeling. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 43–48.

Jiaxun Cai, Shexia He, Zuchao Li, and Hai Zhao. 2018. A full end-to-end semantic role labeler, syntactic-agnostic over syntactic-aware? In *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*, pages 2753–2765.

Wanxiang Che, Yijia Liu, Yuxuan Wang, Bo Zheng, and Ting Liu. 2018. Towards better UD parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 55–64.

Janara Christensen, Mausam, Stephen Soderland, and Oren Etzioni. 2011. An analysis of open information extraction based on semantic role labeling. In *Proceedings of the 6th International Conference on Knowledge Capture*, pages 113–120.

Ronan Collobert, Jason Weston, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(1):2493–2537.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Timothy Dozat and Christopher D Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proceedings of 5th International Conference on Learning Representations (ICLR)*.

Nicholas FitzGerald, Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic role labeling with neural network factors. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 960–970.

William Foland and James Martin. 2015. Dependency-based semantic role labeling using convolutional neural networks. In *Joint Conference on Lexical and Computational Semantics*, pages 279–288.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 1–18.

Shexia He, Zuchao Li, Hai Zhao, Hongxiao Bai, and Gongshen Liu. 2018. Syntax for semantic role labeling, to be, or not to be. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2061–2071.

Sepp Hochreiter and Jrgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

Tao Lei, Yuan Zhang, Lluís Màrquez, Alessandro Moschitti, and Regina Barzilay. 2015. High-order low-rank tensors for semantic role labeling. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL: HLT)*, pages 1150–1160.

Zuchao Li, Shexia He, Jiaxun Cai, Zhuosheng Zhang, Hai Zhao, Gongshen Liu, Linlin Li, and Luo Si. 2018. A unified syntax-aware framework for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2401–2411.

Zuchao Li, Shexia He, Hai Zhao, Yiqing Zhang, Zhuosheng Zhang, Xi Zhou, and Xiang Zhou. 2019. Dependency or span, end-to-end uniform semantic role labeling. *arXiv preprint arXiv:1901.05280*.

Diego Marcheggiani, Anton Frolov, and Ivan Titov. 2017. A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 411–420.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1506–1515.

Phoebe Mulcaire, Swabha Swayamdipta, and Noah A. Smith. 2018. Polyglot semantic role labeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 667–672.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL: HLT)*.

Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James Martin, and Daniel Jurafsky. 2005. Semantic role labeling using different syntactic views. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 581–588.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287.

Michael Roth and Mirella Lapata. 2016. Neural semantic role labeling with dependency path embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1192–1202.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5027–5038.

Swabha Swayamdipta, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2016. Greedy, joint syntactic-semantic parsing with stack lstms. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL)*, pages 187–197.

Deyi Xiong, Min Zhang, and Haizhou Li. 2012. Modeling the translation of predicate-argument structure for SMT. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 902–911.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 88–94.

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 201–206.

Zhisong Zhang, Hai Zhao, and Lianhui Qin. 2016. Probabilistic graph-based dependency parsing with convolutional neural network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1382–1392.

Hai Zhao, Wenliang Chen, Jun'ichi Kazama, Kiyotaka Uchimoto, and Kentaro Torisawa. 2009a. Multilingual dependency learning: Exploiting rich features for tagging syntactic and semantic dependencies. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning - Shared Task (CoNLL)*, pages 61–66.

Hai Zhao, Wenliang Chen, and Chunyu Kit. 2009b. Semantic dependency parsing of NomBank and PropBank: An efficient integrated approach via a large-scale feature selection. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 30–39.

Hai Zhao, Xiaotian Zhang, and Chunyu Kit. 2013. Integrative semantic dependency parsing via efficient large-scale feature selection. *Journal of Artificial Intelligence Research*, 46:203–233.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1127–1137, Beijing, China.

Junru Zhou and Hai Zhao. 2019. Head-driven phrase structure grammar parsing on Penn treebank. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*.

# Cloze-driven Pretraining of Self-attention Networks

**Alexei Baevski, Sergey Edunov*, Yinhan Liu*, Luke Zettlemoyer, Michael Auli**
Facebook AI Research
Menlo Park, CA and Seattle, WA

## Abstract

We present a new approach for pretraining a bi-directional transformer model that provides significant performance gains across a variety of language understanding problems. Our model solves a cloze-style word reconstruction task, where each word is ablated and must be predicted given the rest of the text. Experiments demonstrate large performance gains on GLUE and new state of the art results on NER as well as constituency parsing benchmarks, consistent with BERT. We also present a detailed analysis of a number of factors that contribute to effective pretraining, including data domain and size, model capacity, and variations on the cloze objective.

## 1 Introduction

Language model pretraining has recently been shown to provide significant performance gains for a range of challenging language understanding problems (Dai and Le, 2015; Peters et al., 2018; Radford et al., 2018). However, existing work has either used unidirectional (left-to-right) language models (LMs) (Radford et al., 2018) or bi-directional (both left-to-right and right-to-left) LMs (BiLMs) where each direction is trained with an independent loss function (Peters et al., 2018). In this paper, we show that even larger performance gains are possible by jointly pretraining both directions of a large language-model-inspired self-attention cloze model.

Our bi-directional transformer architecture predicts *every* token in the training data (Figure 1). We achieve this by introducing a cloze-style training objective where the model must predict the center word given left-to-right and right-to-left context representations. Our model separately computes both forward and backward states with a masked self-attention architecture, that closely resembles a language model. At the top of the network, the forward and backward states are combined to jointly predict the center word. This approach allows us to consider both contexts when predicting words and to incur loss for every word in the training set, if the model does not assign it high likelihood.

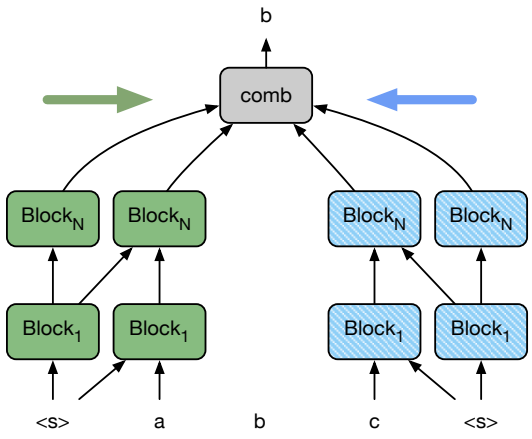


Figure 1: Illustration of the model. Block$_i$ is a standard transformer decoder block. Green blocks operate left to right by masking future time-steps and blue blocks operate right to left. At the top, states are combined with a standard multi-head self-attention module whose output is fed to a classifier that predicts the center token.

Experiments on the GLUE (Wang et al., 2018) benchmark show strong gains over the state of the art for each task, including a 9.1 point gain on RTE over Radford et al. (2018). These improvements are consistent with, if slightly behind, BERT (Devlin et al., 2018), which we will discuss in more detail in the next section. We also show that it is possible to stack task-specific architectures for NER and constituency parsing on top of our pretrained representations, and achieve new state-of-the-art performance levels for both tasks. We also present extensive experimental analysis to better

*Equal contribution.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5360–5369, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

understand these results, showing that (1) having multiple sentences in each training example is crucial for many tasks; (2) pre-training continues to improve performance with up to 18B tokens and would likely continue to improve with more data; and finally (3) our novel cloze-driven training regime is more effective than predicting left and right tokens separately.

## 2 Related work

There has been much recent work on learning sentence-specific representations for language understanding tasks. McCann et al. (2017) learn contextualized word representations from a sequence to sequence translation task and uses the representations from the encoder network to improve a variety of language understanding tasks. Subsequent work focused on language modeling pretraining which has been shown to be more effective and which does not require bilingual data (Zhang and Bowman, 2018).

Our work was inspired by ELMo (Peters et al., 2018) and the generative pretraining (GPT) approach of Radford et al. (2018). ELMo introduces language models to pretrain word representations for downstream tasks including a novel mechanism to learn a combination of different layers in the language model that is most beneficial to the current task. GPT relies on a left to right language model and an added projection layer for each downstream task without a task-specific model. Our approach mostly follows GPT, though we show that our model also works well with an ELMo module on NER and constituency parsing.

The BERT model (Devlin et al., 2018) is a transformer encoder model that captures left and right context. There is significant overlap between their work and ours but there are also significant differences: our model is a bi-directional transformer language model that predicts every single token in a sequence. Our model has two unidirectional components encoding either the left or right context and both are combined to predict center words. BERT is also a transformer encoder that has access to the entire input but this choice requires a special training regime. In particular, they multi-task between predicting a subset of masked input tokens, similar to a denoising autoencoder, and a next sentence prediction task. In comparison, we optimize a single loss function that requires the model to predict each token of an input sentence given all surrounding tokens. We use all tokens as training targets and therefore extract learning signal from every single token in the sentence and not just a subset. Melamud et al. (2016) follow a similar approach to ours by predicting the center word but their architecture is based on LSTMs and we include the center word when we actually fine-tune on downstream tasks.

BERT tailors pretraining to capture dependencies between sentences via a next sentence prediction task as well as by constructing training examples of sentence-pairs with input markers that distinguish between tokens of the two sentences. Our model is trained similarly to a classical language model since we do not adapt the training examples to resemble the end task data and we do not solve a denoising task during training.

Finally, BERT as well as Radford et al. (2018) consider only a single data source to pretrain their models, either BooksCorpus (Radford et al., 2018), or BooksCorpus and additional Wikipedia data (Devlin et al., 2018), whereas our study ablates the effect of various amounts of training data as well as different data sources.

## 3 Two tower model

Our cloze model represents a probability distribution $p(t_i|t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_n)$ for a sentence with $n$ tokens $t_1, \ldots, t_n$. There are two self-attentional towers each consisting of $N$ stacked blocks: the *forward* tower operates left-to-right and the *backward* tower operates in the opposite direction. To predict a token, we combine the representations of the two towers, as described in more detail below, taking care that neither representation contains information about the current target token.

The forward tower computes the representation $F_i^l$ for token $i$ at layer $l$ based on the forward representations of the previous layer $F_{\leq i}^{l-1}$ via self-attention; the backward tower computes representation $B_i^l$ based on information from the opposite direction $B_{\geq i}^{l-1}$. When examples of uneven length are batched, one of the towers may not have any context at the beginning. We deal with this issue by adding an extra zero state over which the self-attention mechanism can attend.

We pretrain on individual examples as they occur in the training corpora (§5.1). For News Crawl this is individual sentences while on Wikipedia, Bookcorpus, and Common Crawl examples are

paragraph length. Sentences are prepended and appended with sample boundary markers $< s >$.

### 3.1 Block structure

The structure of the blocks follows most of the architectural choices described in Vaswani et al. (2017). Each block consists of two sub-blocks: the first is a multi-head self-attention module with $H = 16$ heads for which we mask out any subsequent time-steps, depending on if we are dealing with the forward or backward tower. The second sub-block is a feed-forward module (FFN) of the form $ReLU(\boldsymbol{W}_1\boldsymbol{X} + b_1)\boldsymbol{W}_2 + b_2$ where $\boldsymbol{W}_1 \in \mathbb{R}^{e \times f}$, $\boldsymbol{W}_1 \in \mathbb{R}^{f \times e}$. Different to Vaswani et al. (2017) we apply layer normalization before the self-attention and FFN blocks instead of after, as we find it leads to more effective training. Sub-blocks are surrounded by a residual connection (He et al., 2015). Position is encoded via fixed sinusoidal position embeddings and we use a character CNN encoding of the input tokens for word-based models (Kim et al., 2016). Input embeddings are shared between the two towers.

### 3.2 Combination of representations

The forward and backward representations computed by the two towers are combined to predict the ablated word. To combine them we use a self-attention module which is followed by an FFN block (§3.1). The output of the FFN block $f$ is projected by $\mathbf{W}$ into $V$ classes representing the types in the vocabulary: $\mathbf{W}^T f$ to which a softmax is applied. When the model predicts token $i$, the input to the attention module are forward states $F_1^L \dots F_{i-1}^L$ and backward states $B_{i+1}^L \dots B_n^L$ where $n$ is the length of the sequence and $L$ is the number of layers. We implement this by masking $B_{\leq i}^L$ and $F_{\geq i}^L$. The attention query for token $i$ is a combination of $F_{i-1}^L$ and $B_{i+1}^L$. For the base model we sum the two representations and for the larger models they are concatenated. Keys and values are based on the forward and backward states fed to the attention module. In summary, this module has access to information about the entire input surrounding the current target token. During training, we predict every token in this way. The output of this module is fed to an output classifier which predicts the center token. We use an adaptive softmax for the output classifier (Grave et al., 2017) for the word based models and regular softmax for the BPE based models.

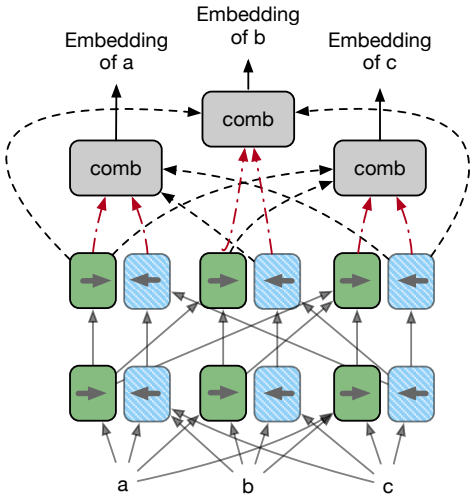


Figure 2: Illustration of fine-tuning for a downstream task. For classification problems, output of the first and last token is fed to a task-specific classifier. Masking for the final combination layer (comb) is removed which results in representations based on all forward and backward states (cf. Figure 1). The red dot-dashed arrows show connections that are masked during training, but unmasked for fine-tuning.

While all states that contain information about the current target word are masked in the final self-attention block during training, we found it beneficial to disable this masking when fine tuning the pretrained model for downstream tasks. This is especially true for tasks that label each token, such as NER, as this allows the model to access the full context including the token itself.

## 4 Fine-tuning

We use the following approach to fine-tune the pretrained two tower model to specific downstream tasks (Figure 2).

**Classification and regression tasks.** For single sentence classification tasks, we consider the language model outputs for the boundary tokens $< s >$ which we add before the start and end of each sentence. The language model outputs are the representations $f$ just before the final softmax layer (§3.2). The outputs are of dimension $d = 1024$ and we concatenate them to project to the number of classes $C$ in the downstream task with $W_1 \in \mathbb{R}^{C \times 2d}$ (Radford et al., 2018); we add a bias term $b \in \mathbb{R}^C$ and initialize all weights as well as the bias to zero. The output of the projection is softmax-normalized and the model is optimized with cross-entropy for classification tasks. Re-

gression tasks such as the Semantic Textual Similarity benchmark (STS-B; Cer et al., 2017) use $C = 1$ and are trained with mean squared error. For tasks involving sentence-pairs, we concatenate them and add a new separator token $< sep >$ between them. We add the output of this token to the final projection $W_2 \in \mathbb{R}^{C \times 3d}$.

**Structured prediction tasks.** For named entity recognition and parsing we use task-specific architectures which we fine-tune together with the language model but with different learning rate. The architectures are detailed in the respective results sections. The input to the architectures are the output representations of the pretrained language model.

**No Masking.** For fine-tuning, we found it beneficial to remove masking of the current token in the final layer that pools the output of the two towers. This is different than in the actual pretraining. It is important to have access to information about the token to be classified for token level classification tasks such as NER but we also found this to perform better for sentence classification tasks. In practice, we completely disable masking in the combination layer so that it operates over all forward and backward states. However, disabling masking below the combination layer does not perform well.

**Optimization.** During fine-tuning we use larger learning rates for the new parameters, that is $W_1$, $W_2$, $b$ or the task-specific architecture, compared to the pretrained model. For GLUE tasks, we do so by simply scaling the output of the language model before the $W_1$ and $W_2$ projections by a factor of 16. For structured prediction tasks, we explicitly use different learning rates for the pretrained model and the task-specific parameters.

We fine tune with the Adam optimizer (Kingma and Ba, 2015). For GLUE tasks, we disable dropout in the language model and add 0.1 dropout between language model output and the final output projection; for structured prediction tasks, we use 0.3 at all levels (within the pretrained model, within the task-specific architecture, and on the weights connecting them). In all settings, we use a batch size of 16 examples. We use a cosine schedule to linearly warm up the learning rate from 1e-07 to the target value over the first 10% of training steps, and then anneal the learning rate to 1e-06, following the cosine curve for the remaining steps. For GLUE tasks, we tuned the learning rate for each task and chose the best value over three settings: 1e-04, 5e-05 and 3e-05. For structured prediction tasks, we tuned on the pairs of learning rate, see the results section for details. For GLUE tasks, we train three seeds for each learning rate value for three epochs and choose the model after each epoch that performs best on the validation set. For structured prediction tasks, we train for up to 25 epochs and stop if the validation loss does not improve over the previous epoch.

## 5 Experimental setup

### 5.1 Datasets for pretraining

We train the two tower model on several datasets.

**Common Crawl.** We consider various subsets of Common Crawl which is web data. We follow the same pre-processing as Grave et al. (2018) which is based on the May 2017 Common Crawl dump. This setup add 20 copies of English Wikipedia resulting in about 14% of the final dataset to be Wikipedia. We subsample up to 18B tokens. All experiments use Common Crawl subsampled to 9B tokens, except §6.4.

**News Crawl.** We use up to 4.5B words of English news web data distributed as part of WMT 2018 (Bojar et al., 2018).

**BooksCorpus + Wikipedia.** This is similar to the training data used by BERT which comprises the BooksCorpus (Zhu et al., 2015) of about 800M words plus English Wikipedia data of 2.5B words.

### 5.2 Pretraining hyper-parameters

We adapt the transformer implementation available in the fairseq toolkit to our two tower architecture (Ott et al., 2019). For hyper-parameter and optimization choices we mostly follow Baevski and Auli (2018). Our experiments consider three model sizes shown in Table 1: There are two CNN input models in a base and large configuration as well as a Byte-Pair-Encoding based model (BPE; Sennrich et al., 2016). The CNN models have unconstrained input vocabulary, and an output vocabulary limited to 1M most common types for the large model, and 700K most common types for the base model. CNN models use an adaptive softmax in the output: the head band contains the 60K most frequent types with dimensionality

| Model | Parameters | Updates | Blocks | FFN Dim | Attn Heads (final layer) | Query formation (final layer) | Train time (days) |
|---|---|---|---|---|---|---|---|
| CNN Base | 177M | 600K | 6 | 4096 | 12 | Sum | 6 |
| CNN Large | 330M | 1M | 12 | 4096 | 32 | Concat | 10 |
| BPE Large | 370M | 1M | 12 | 4096 | 32 | Concat | 4.5 |

Table 1: Hyper-parameters for our models. Parameter count excludes the (adaptive) softmax layer. Train time as measured on 128 Volta GPUs for the CNN models and 64 Volta GPUs for the BPE model.

1024, followed by a 160K band with dimensionality 256. The remaining types have dimensionality 64; there are 480K types for the small model and 780K for the large model. The BPE model uses a vocabulary of 55K types and we share input and output embeddings in a flat softmax with dimension 1024 (Inan et al., 2016; Press and Wolf, 2017). The BPE vocabulary was constructed by applying 30K merge operations over the training data, then applying the BPE code to the training data and retaining all types occurring at least three times.

Every setup uses model dimensionaltiy $d = 1024$ with $H = 16$ attention heads for all but the final attention layer. Model based on character inputs use character embedding size 128 and we apply six filters of size 1x128, 2x256, 3x384, 4x512, 5x512, 6x512 followed by a single highway layer. The models are trained with model and attention dropout rate of 0.1 and ReLU dropout rate of 0.05.

Different to Vaswani et al. (2017) we use Nesterov's accelerated gradient method (Sutskever et al., 2013) with a momentum of 0.99 and we renormalize gradients if their norm exceeds 0.1 (Pascanu et al., 2013). The learning rate is linearly warmed up from $10^{-7}$ to 1 for 16K steps and then annealed using a cosine learning rate schedule with a single phase to 0.0001 (Loshchilov and Hutter, 2016).

We run experiments on DGX-1 machines with 8 NVIDIA V100 GPUs and machines are interconnected by Infiniband. We also use the NCCL2 library and the torch.distributed package for inter-GPU communication. We train models with 16-bit floating point precision, following Ott et al. (2018). The BPE model trains much faster than the character CNN models (Table 1).

## 6 Results

### 6.1 GLUE

First, we conduct experiments on the general language understanding evaluation benchmark (GLUE; Wang et al., 2018) and present a short overview of the tasks. More information can be found in Wang et al. (2018). There are two single-sentence classification tasks: First, the Corpus of Linguistic Acceptability (CoLA; Warstadt et al., 2018) is a binary task to judge sentence grammaticality; evaluation is in terms of the Matthews correlation coefficient (mcc). Second, the Stanford Sentiment Treebank (SST-2; Socher et al., 2013) requires to judge if movie reviews have positive or negative sentiment; evaluation is in terms of accuracy (acc).

There are three tasks assessing sentence similarity: The Microsoft Research Paragraph Corpus (MRPC; Dolan and Brockett, 2015) and the Quora Question Pairs benchmark (QQP); we evaluate in terms of F1. The Semantic Textual Similarity Benchmark (STS-B; Cer et al., 2017) requires predicting a similarity score between 1 and 5 for a sentence pair; we report the Spearman correlation coefficient (scc).

Finally, there are four natural language inference tasks: the Multi-Genre Natural Language Inference (MNLI; Williams et al., 2018), the Stanford Question Answering Dataset (QNLI; Rajpurkar et al., 2016), the Recognizing Textual Entailment (RTE; Dagan et al., 2006, Bar Haim et al., 2006, Ciampiccolo et al., 2007 Bentivogli et al., 2009). We exclude the Winograd NLI task from our results similar to Radford et al. (2018); Devlin et al. (2018) and report accuracy. For MNLI we report both matched (m) and mismatched (mm) accuracy on test.

We also report an average over the GLUE metrics. This figure is not comparable to the average on the official GLUE leaderboard since we exclude Winograd and do not report MRPC accuracy

| | CoLA (mcc) | SST-2 (acc) | MRPC (F1) | STS-B (scc) | QQP (F1) | MNLI-(m/mm) (acc) | QNLI (acc) | RTE (acc) | Avg |
|---|---|---|---|---|---|---|---|---|---|
| OpenAI GPT | 45.4 | 91.3 | 82.3 | 80.0 | 70.3 | 82.1/81.4 | 88.1 | 56.0 | 75.2 |
| CNN Base | 53.1 | 93.6 | 81.3 | 82.2 | 70.5 | 82.5/82.2 | 89.5 | 64.6 | 77.7 |
| CNN Large | 52.8 | 94.6 | 83.7 | 83.4 | 71.7 | 84.3/83.8 | 89.8 | 63.7 | 78.6 |
| BPE Large | 51.8 | 94.0 | 83.0 | 84.2 | 70.6 | 82.9/82.2 | 89.3 | 65.1 | 78.1 |
| GPT on STILTs | 47.2 | 93.1 | 87.7 | 84.8 | 70.1 | 80.7/80.6 | 87.2 | 69.1 | 77.8 |
| BERT$_{BASE}$ | 52.1 | 93.5 | 88.9 | 85.8 | 71.2 | 84.6/83.4 | 90.1 | 66.4 | 79.6 |
| BERT$_{LARGE}$ | 60.5 | 94.9 | 89.3 | 86.5 | 72.1 | 86.7/85.9 | 91.1 | 70.1 | 81.9 |

Table 2: Test results as per the GLUE evaluation server. The average column does not include the WNLI test set. mcc = Matthews correlation, acc = Accuracy, scc = Spearman correlation.

STS-B Pearson correlation as well as QQP accuracy.

Table 2 shows results for three configurations of our approach (cf. Table 1). The BPE model has more parameters than the CNN model but does not perform better in aggregate, however, it is faster to train. All our models outperform the unidirectional transformer (OpenAI GPT) of Radford et al. (2018), however, our model is about 50% larger than their model. We also show results for STILTs (Phang et al., 2018) and BERT (Devlin et al., 2018). Our CNN base model performs as well as STILTs in aggregate, however, on some tasks involving sentence-pairs, STILTs performs much better (MRPC, RTE); there is a similar trend for BERT.

STILTs adds another fine-tuning step on another downstream task which is similar to the final task. The technique is equally applicable to our approach. Training examples for our model are Common Crawl paragraphs of arbitrary length. We expect that tailoring training examples for language model pretraining to the end tasks to significantly improve performance. For example, BERT trains on exactly two sentences while as we train on entire paragraphs.

### 6.2 Structured Prediction

We also evaluated performance on two structured predictions tasks, NER and constituency parsing. For both problems, we stacked task-specific architectures from recent work on top of our pretrained two tower models. We evaluate two ways of stacking: (1) ELMo-style, where the pretrained models are not fine-tuned but are linearly combined at different depths, and (2) with fine-tuning, where we set different learning rates for the task-specific layers but otherwise update all of the parameters during the task-specific training.

| **Model** | **dev F1** | **test F1** |
|---|---|---|
| ELMo$_{BASE}$ | 95.7 | 92.2 |
| CNN Large + ELMo | 96.4 | 93.2 |
| CNN Large + fine-tune | 96.9 | 93.5 |
| BERT$_{BASE}$ | 96.4 | 92.4 |
| BERT$_{LARGE}$ | 96.6 | 92.8 |

Table 3: CoNLL-2003 Named Entity Recognition results. Test result was evaluated on parameter set with the best dev F1.

| **Model** | **dev F1** | **test F1** |
|---|---|---|
| ELMo$_{BASE}$ | 95.2 | 95.1 |
| CNN Large + ELMo | 95.1 | 95.2 |
| CNN Large + fine-tune | 95.5 | 95.6 |

Table 4: Penn Treebank Constituency Parsing results. Test result was evaluated on parameter set with the best dev F1.

#### 6.2.1 Named Entity Recognition

We evaluated span-level F1 performance on the CoNLL 2003 Named Entity Recognition (NER) task, where spans of text must be segmented and labeled as Person, Organization, Location, or Miscellaneous. We adopted the NER architecture in Peters et al. (2018), a biLSTM-CRF, with two minor modifications: (1) instead of two layers of biLSTM, we only used one, and (2) a linear projection layer was added between the token embedding and biLSTM layer. We did grid search on the pairs of learning rate, and found that projection-biLSTM-

| | CoLA (mcc) | SST-2 (acc) | MRPC (F1) | STS-B (scc) | QQP (F1) | MNLI-m (acc) | QNLI (acc) | RTE (acc) | Avg |
|---|---|---|---|---|---|---|---|---|---|
| cloze | 55.1 | 92.9 | 88.3 | 88.3 | 87.2 | 82.3 | 86.5 | 66.4 | 80.9 |
| bilm | 50.0 | 92.4 | 86.6 | 87.1 | 86.1 | 81.7 | 84.0 | 66.4 | 79.3 |
| cloze + bilm | 52.6 | 93.2 | 88.9 | 87.9 | 87.2 | 82.1 | 86.1 | 65.5 | 80.4 |

Table 5: Different loss functions on the development sets of GLUE (cf. Table 2). Results are based on the CNN base model (Table 1)

CRF with 1E-03 and pretrained language model with 1E-05 gave us the best result.

Table 3 shows the results, with comparison to previous published ELMo$_{BASE}$ results (Peters et al., 2018) and the BERT models. Both of our stacking methods outperform the previous state of the art, but fine tuning gives the biggest gain.

#### 6.2.2 Constituency Parsing

We also report parseval F1 for Penn Treebank constituency parsing. We adopted the current state-of-the-art architecture (Kitaev and Klein, 2018). We again used grid search for learning rates and number of layers in parsing encoder, and used 8E-04 for language model finetuning, 8E-03 for the parsing model parameters, and two layers for encoder.

Table 4 shows the results. Here, fine tuning is required to achieve gains over the previous state of the art, which used ELMo embeddings.

### 6.3 Objective functions for pretraining

The two-tower model is trained to predict the current token given representations of the entire left and right context (cloze). Next we compare this choice to two alternatives: First, Peters et al. (2018) train two language models operating left-to-right and right-to-left to predict the next word for each respective direction. We change the two-tower model to predict the next word using the individual towers only and remove the combination module on top of the two towers (bilm); however, we continue to jointly train the two towers.

Second, we combine the cloze loss with the bilm loss to obtain a triplet loss which trains the model to predict the current word given both left and right context, as well as just right or left context. The latter is much harder than the cloze loss since less context is available and therefore gradients for the bilm loss are much larger: the cloze model achieves perplexity of about 4 while as for the bilm it is 27-30, depending on the direction. This results in the bilm loss dominating the triplet loss and we found that scaling the bilm term by a factor of 0.15 results in better performance.

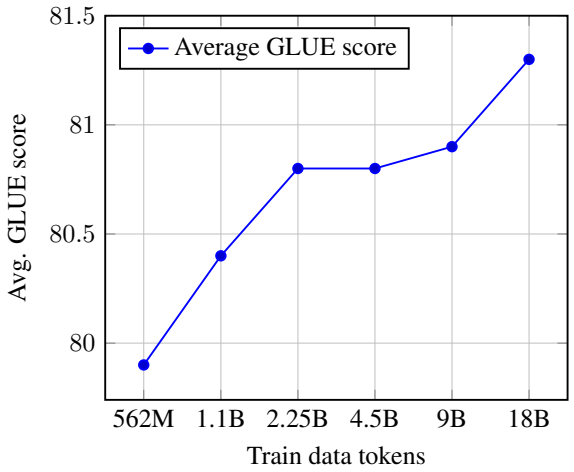


Figure 3: Average GLUE score with different amounts of Common Crawl data for pretraining.

Table 5 shows that the cloze loss performs significantly better than the bilm loss and that combining the two loss types does not improve over the cloze loss by itself. We conjecture that individual left and right context prediction tasks are too different from center word prediction and that their learning signals are not complementary enough.

### 6.4 Domain and amount of training data

Next we investigate how much pretraining benefits from larger training corpora and how the domain of the data influences end-task performance.

Figure 3 shows that more training data can significantly increase accuracy. We train all models with the exact same hyper-parameter settings on Common Crawl data using the CNN base architecture for 600K updates. We train on up to 18B Common Crawl tokens and the results suggest that more training data is likely to further increase performance.

Table 6 shows a breakdown into individual

| | train data (M tok) | CoLA (mcc) | SST-2 (acc) | MRPC (F1) | STS-B (scc) | QQP (F1) | MNLI-m (acc) | QNLI (acc) | RTE (acc) | Avg |
|---|---|---|---|---|---|---|---|---|---|---|
| ccrawl | 562 | 52.5 | 92.9 | 88.2 | 88.3 | 87.1 | 81.7 | 85.7 | 63.3 | 79.9 |
| | 1125 | 55.5 | 93.1 | 86.1 | 88.4 | 87.1 | 81.9 | 85.7 | 65.2 | 80.4 |
| | 2250 | 55.4 | 92.4 | 87.7 | 88.4 | 87.2 | 82.2 | 86.2 | 66.9 | 80.8 |
| | 4500 | 56.6 | 93.0 | 87.3 | 88.6 | 87.0 | 82.0 | 86.2 | 65.7 | 80.8 |
| | 9000 | 55.1 | 92.9 | 88.3 | 88.3 | 87.2 | 82.3 | 86.5 | 66.4 | 80.9 |
| | 18000 | 56.3 | 93.1 | 88.0 | 88.8 | 87.2 | 82.3 | 86.3 | 68.4 | 81.3 |
| news crawl | 562 | 50.9 | 92.8 | 81.4 | 78.2 | 84.9 | 79.1 | 82.0 | 55.7 | 75.6 |
| | 1125 | 51.4 | 93.0 | 83.0 | 82.3 | 85.2 | 79.7 | 82.8 | 53.9 | 76.4 |
| | 2250 | 54.8 | 92.9 | 83.5 | 82.8 | 85.4 | 80.4 | 82.4 | 54.8 | 77.1 |
| | 4500 | 53.9 | 93.6 | 83.8 | 83.1 | 85.5 | 80.4 | 83.6 | 54.2 | 77.3 |
| BWiki - sent | 3300 | 53.5 | 91.6 | 86.4 | 86.2 | 86.9 | 82.3 | 86.9 | 63.8 | 79.7 |
| BWiki - blck | 3300 | 50.6 | 91.9 | 86.4 | 87.1 | 86.8 | 81.9 | 86.2 | 60.4 | 78.9 |

Table 6: Effect of different domains and amount of data for pretraining on the on the development sets of GLUE (cf. Table 2). Results are based on the CNN base model (Table 1).

GLUE tasks. For pretraining on Common Crawl, CoLA and RTE benefit most from additional training data. The same table also shows results for News Crawl which contains newswire data. This data generally performs less well than Common Crawl, even on MRPC which is newswire. A likely reason is that News Crawl examples are *individual sentences* of 23 words on average which compares to several sentences or 50 words on average for Common Crawl. Mutli-sentence training examples are more effective for end-tasks based on sentence pairs, e.g., there is a 14 point accuracy gap on RTE between News Crawl and Common Crawl with 4.5B tokens. More News Crawl data is most beneficial for CoLA and STS-B.

We also experiment with BooksCorpus (Zhu et al., 2015) as well as English Wikipedia, similar to Devlin et al. (2018). Examples in BooksCorpus are a mix of individual sentences and paragraphs; examples are on average 36 tokens. Wikipedia examples are longer paragraphs of 66 words on average. To reduce the effect of training on examples of different lengths, we adopted the following strategy: we concatenate all training examples into a single string and then crop blocks of 512 consecutive tokens from this string. We train on a batch of these blocks (BWiki - blck). It turns out that this strategy did not work better compared to our existing strategy of simply using the data as is (BWiki - sent). BooksCorpus and Wikipedia performs very well on QNLI and MNLI but less well on other tasks.

In summary, more data for pretraining improves performance, keeping everything else equal. Also pretraining on corpora that retains paragraph structure performs better than individual sentences.

## 7 Conclusion

We presented a pretraining architecture based on a bi-directional transformer model that predicts every token in the training data. The model is trained with a cloze-style objective and predicts the center word given all left and right context.

Results on the GLUE benchmark show large gains over Radford et al. (2018) for each task, while experiments with model stacking set new state of the art performance levels for parsing and named entity recognition. We also did extensive experimental analysis to better understand these results, showing that (1) having multiple sentences in each training example is crucial for many tasks; (2) pre-training continues to improve performance up to 18B tokens and would likely continue to improve with more data; and finally (3) our novel cloze-driven training regime is more effective than predicting left and right tokens separately.

In future work, we will investigate variations of our architecture. In particular, we had initial success sharing the parameters of the two towers which allows training much deeper models without increasing the parameter count.

## References


Alexei Baevski and Michael Auli. 2018. Adaptive input representations for neural language modeling. *arXiv*, abs/1809.10853.

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2009. The fifth pascal recognizing textual entailment challenge. In *Proc. of TAC*.

Ondřej Bojar, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, and Christof Monz. 2018. Findings of the 2018 conference on machine translation (WMT18). In *Proc. of WMT*.

Daniel M. Cer, Mona T. Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2018. Semeval-2017 task 1: Semantic textual similarity - multilingual and cross-lingual focused evaluation. In *Proc. of SemEval*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognizing textual entailment challenge. *Machine learning challenges, evaluating predictive uncertainty, visual object classification, and recognizing textual entailment*, pages 177–190.

Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised sequence learning. *arXiv*, abs/1511.01432.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proc. of IWP*.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The pascal recognizing textual entailment challenge. *Proc. of the ACL-PASCAL workshop on textual entailment and paraphrasing*.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proc. of LREC*.

Edouard Grave, Armand Joulin, Moustapha Cissé, David Grangier, and Hervé Jégou. 2017. Efficient softmax approximation for gpus. In *Proc. of ICML*.

Roy Bar Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The pascal recognising textual entailment challenge.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. In *Proc. of CVPR*.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2016. Tying word vectors and word classifiers: A loss framework for language modeling. *arXiv*, abs/1611.01462.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *AAAI*, pages 2741–2749.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proc. of ICLR*.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proc. of ACL*.

Ilya Loshchilov and Frank Hutter. 2016. SGDR: stochastic gradient descent with restarts. *arXiv*, abs/1608.03983.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Proc. of NIPS*.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional lstm. In *Proc. of CoNLL*.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proc. of NAACL System Demonstrations*.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. 2018. Scaling neural machine translation. In *Proc. of WMT*.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proc. of ICML*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of ACL*.

Jason Phang, Thibault Fevry, and Samuel R. Bowman. 2018. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *arXiv*, abs/1811.01088.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proc. of EACL*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. *arXiv*, abs/1606.05250.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proc. of ACL*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. of EMNLP*.

Ilya Sutskever, James Martens, George E. Dahl, and Geoffrey E. Hinton. 2013. On the importance of initialization and momentum in deep learning. In *Proc. of ICML*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Proc. of NIPS*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *arXiv*, abs/1804.07461.

Alex Warstadt, Amanpreet Singh, and Sam Bowman. 2018. Corpus of linguistic acceptability. https://nyu-mll.github.io/CoLA.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proc. of NAACL*.

Kelly W. Zhang and Samuel R. Bowman. 2018. Language modeling teaches you more syntax than translation does: Lessons learned through auxiliary task analysis. *arXiv*, abs/1809.10040.

Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *arXiv*, abs/1506.06724.

# Bridging the Gap Between Relevance Matching and Semantic Matching for Short Text Similarity Modeling

**Jinfeng Rao,**[1] **Linqing Liu,**[2] **Yi Tay,**[3] **Wei Yang,**[2] **Peng Shi,**[2] and **Jimmy Lin**[2]
[1] Facebook Assistant
[2] David R. Cheriton School of Computer Science, University of Waterloo
[3] Nanyang Technological University
raojinfeng@fb.com

## Abstract

A core problem of information retrieval (IR) is *relevance matching*, which is to rank documents by relevance to a user's query. On the other hand, many NLP problems, such as question answering and paraphrase identification, can be considered variants of *semantic matching*, which is to measure the semantic distance between two pieces of short texts. While at a high level both relevance and semantic matching require modeling textual similarity, many existing techniques for one cannot be easily adapted to the other. To bridge this gap, we propose a novel model, HCAN (Hybrid Co-Attention Network), that comprises (1) a hybrid encoder module that includes ConvNet-based and LSTM-based encoders, (2) a relevance matching module that measures soft term matches with importance weighting at multiple granularities, and (3) a semantic matching module with co-attention mechanisms that capture context-aware semantic relatedness. Evaluations on multiple IR and NLP benchmarks demonstrate state-of-the-art effectiveness compared to approaches that do not exploit pretraining on external data. Extensive ablation studies suggest that relevance and semantic matching signals are complementary across many problem settings, regardless of the choice of underlying encoders.

## 1 Introduction

Neural networks have achieved great success in many NLP tasks, such as question answering (Rao et al., 2016; Chen et al., 2017a), paraphrase detection (Wang et al., 2017), and textual semantic similarity modeling (He and Lin, 2016). Many of these tasks can be treated as variants of a *semantic matching (SM)* problem, where two pieces of texts are jointly modeled through distributed representations for similarity learning. Various neural network architectures, e.g., Siamese networks (He et al., 2016) and attention (Seo et al., 2017; Tay et al., 2019b), have been proposed to model semantic similarity using diverse techniques.

A core problem of information retrieval (IR) is *relevance matching (RM)*, where the goal is to rank documents by relevance to a user's query. Though at a high level semantic and relevance matching both require modeling similarities in pairs of texts, there are fundamental differences. Semantic matching emphasizes "meaning" correspondences by exploiting lexical information (e.g., words, phrases, entities) and compositional structures (e.g., dependency trees), while relevance matching focuses on keyword matching. It has been observed that existing approaches for textual similarity modeling in NLP can produce poor results for IR tasks (Guo et al., 2016), and vice versa (Htut et al., 2018).

Specifically, Guo et al. (2016) point out three distinguishing characteristics of relevance matching: exact match signals, query term importance, and diverse matching requirements. In particular, exact match signals play a critical role in relevance matching, more so than the role of term matching in, for example, paraphrase detection. Furthermore, in document ranking there is an asymmetry between queries and documents in terms of length and the richness of signals that can be extracted; thus, symmetric models such as Siamese architectures may not be entirely appropriate.

To better demonstrate these differences, we present examples from relevance and semantic matching tasks in Table 1. Column 'Label' denotes whether sentence A and B are relevant or duplicate. The first example from tweet search shares many common keywords and is identified as relevant, while the second pair from Quora shares all words except for the subject and is *not* considered a duplicate pair. An approach based on keyword matching alone is unlikely to be able to distinguish



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5370–5381, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| Task | Label | Sentence A | Sentence B |
|---|---|---|---|
| Tweet Search | 1 | 2022 FIFA soccer | 2022 world cup FIFA could be held at the end of year in Qatar |
| Duplicate Detection | 0 | Does RBI send its employees for higher education, like MBA? | Does EY send its employees for higher education, like MBA? |
| Question Answering | 1 | What was the monetary value of the Nobel peace prize in 1989 ? | Each Nobel prize is worth $469,000 . |

Table 1: Sample sentence pairs from TREC Microblog 2013, Quora, and TrecQA.

between these cases. In contrast, the third example is judged as a relevant QA pair because different terms convey similar semantics.

These divergences motivate different architectural choices. Since relevance matching is fundamentally a matching task, most recent neural architectures, such as DRMM (Guo et al., 2016) and Co-PACRR (Hui et al., 2018), adopt an interaction-based design. They operate directly on the similarity matrix obtained from products of query and document embeddings and build sophisticated modules on top to capture additional $n$-gram matching and term importance signals. On the other hand, many NLP problems, such as question answering and textual similarity measurement, require more semantic understanding and contextual reasoning rather than specific term matches. Context-aware representation learning, such as co-attention methods (Seo et al., 2017), has been proved effective in many benchmarks. Though improvements have been shown from adding exact match signals into representation learning, for example, the Dr.QA model of Chen et al. (2017a) concatenates exact match scores to word embeddings, it remains unclear to what extent relevance matching signals can further improve models primarily designed for semantic matching.

To this end, we examine two research questions: (1) Can existing approaches to relevance matching and semantic matching be easily adapted to the other? (2) Are signals from relevance and semantic matching complementary? We present a novel neural ranking approach to jointly model both the relevance matching process and the semantic matching process. Our model, **HCAN** (**H**ybrid **C**o-**A**ttention **N**etwork), comprises three major components:

1. A hybrid encoder module that explores three types of encoders: *deep*, *wide*, and *contextual*, to obtain contextual sentence representations.
2. A relevance matching module that measures soft term matches with term weightings between pairs of texts, starting from word-level to phrase-level, and finally to sentence-level.
3. A semantic matching module with co-attention mechanisms applied at each encoder layer to enable context-aware representation learning at multiple semantic levels.

Finally, all relevance and semantic matching signals are integrated using a fully-connected layer to yield the final classification score.

**Contributions.** We see our work as making the following contributions:

- We highlight and systematically explore important differences between relevance matching and semantic matching on short texts, which lie at the core of many of IR and NLP problems.
- We propose a novel model, HCAN (Hybrid Co-Attention Network), to combine best practices in neural modeling for both relevance and semantic matching.
- Evaluations on multiple IR and NLP tasks, including answer selection, paraphrase identification, semantic similarity measurement, and tweet search, demonstrate state-of-the-art effectiveness compared to approaches that do not exploit pretraining on external data. Ablation studies show that relevance and semantic matching signals are complementary in many problems, and combining them can be more data efficient.

## 2 HCAN: Hybrid Co-Attention Network

The overview of our model is shown in Figure 1. It is comprised of three major components: (1) a hybrid encoder module that explores three types of encoders: *deep*, *wide*, and *contextual* (Sec. 2.1);

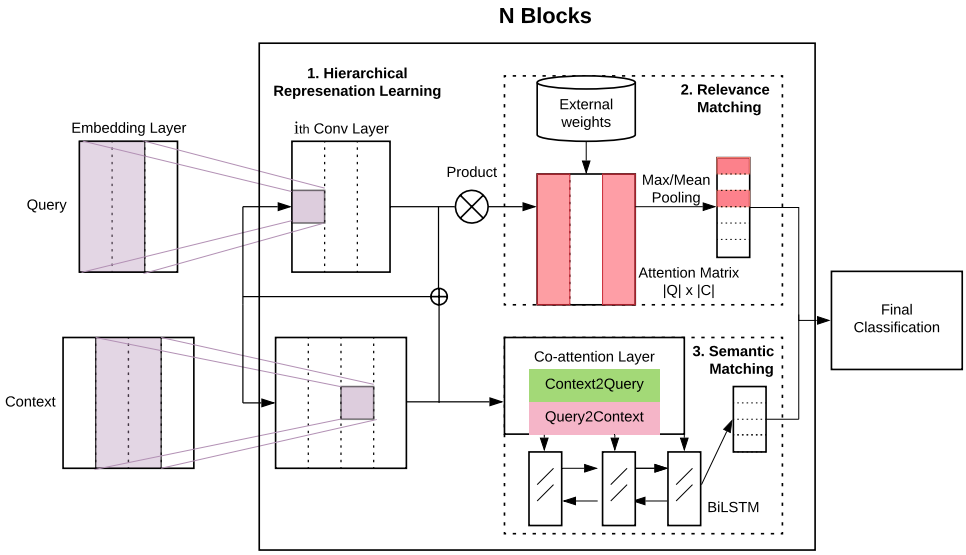


Figure 1: Overview of our Hybrid Co-Attention Network (HCAN). The model consists of three major components: (1) a hybrid encoder module that explores three types of encoders: *deep*, *wide*, and *contextual*; (2) a relevance matching module with external weights for learning soft term matching signals; (3) a semantic matching module with co-attention mechanisms for context-aware representation learning.

(2) a relevance matching module with external weights for learning soft term matching signals (Sec. 2.2); (3) a semantic matching module with co-attention mechanisms for context-aware representation learning (Sec. 2.3). Note that the relevance and semantic matching modules are applied at each encoder layer, and all signals are finally aggregated for classification.

## 2.1 Hybrid Encoders

Without loss of generality, we assume that inputs to our model are sentence pairs $(q, c)$, where $(q, c)$ can refer to a (query, document) pair in a search setting, a (question, answer) pair in a QA setting, etc. The query $q$ and context $c$ are denoted by their words, $\{w_1^q, w_2^q, ..., w_n^q\}$ and $\{w_1^c, w_2^c, ..., w_m^c\}$, respectively, where $n$ and $m$ are the number of words in the query and the context. A word embedding layer converts both into their embedding representations $\mathbf{Q} \in \mathbb{R}^{n \times L}$ and $\mathbf{C} \in \mathbb{R}^{m \times L}$, where $L$ is the dimension of the embeddings.

To learn effective phrase-level representations, we explore three different types of encoders: *deep*, *wide*, and *contextual*, detailed below.

**Deep Encoder**: This design consists of multiple convolutional layers stacked in a hierarchical manner to obtain higher-level $k$-gram representations. A convolutional layer applies convolutional filters to the text, which is represented by an embedding matrix $\mathbf{U}$ ($\mathbf{Q}$ or $\mathbf{C}$). Each filter is moved through the input embedding incrementally as a sliding window (with window size $k$) to capture the compositional representation of $k$ neighboring terms. Assuming a convolutional layer has $F$ filters, this CNN layer (with padding) produces an output matrix $\mathbf{U}^o \in \mathbb{R}^{\|U\| \times F}$.

For notational simplicity, we drop the superscript $o$ from all output matrices and add a superscript $h$ to denote the output of the $h$-th convolutional layer. Stacking $N$ CNN layers therefore corresponds to obtaining the output matrix of the $h$-th layer $\mathbf{U}^h \in \mathbb{R}^{\|U\| \times F^h}$ via:

$$\mathbf{U}^h = \text{CNN}^h(\mathbf{U}^{h-1}), h = 1, \ldots, N,$$

where $\mathbf{U}^{h-1}$ is the output matrix of the $(h-1)$-th convolutional layer. Note that $\mathbf{U}^0 = \mathbf{U}$ denotes the input matrix ($\mathbf{Q}$ or $\mathbf{C}$) obtained directly from the word embedding layer. The parameters of each CNN layer are shared by the query and the context.

**Wide Encoder**: Unlike the deep encoder that stacks multiple convolutional layers hierarchically, the wide encoder organizes convolutional layers in parallel, with each convolutional layer having a different window size $k$ to obtain the corresponding $k$-gram representations. Given $N$ convolutional layers, the window sizes of the CNN layers will be in $[k, k+1, ..., k+N-1]$.

**Contextual Encoder**: Different from both the deep and wide encoders that capture $k$-gram patterns with convolutions, the contextual encoder

leverages Bi-directional LSTMs to extract long-range contextual features. Given $N$ BiLSTM layers, the output at the $h$-th layer is computed as:

$$\mathbf{U}^h = \text{BiLSTM}^h(\mathbf{U}^{h-1}), h = 1, \ldots, N,$$

The three encoders represent different tradeoffs. The deep and wide encoders are easier for performing inference in parallel and are much faster to train than the contextual encoder. Additionally, the use of CNN layers allows us to explicitly control the window size for phrase modeling, which has been shown to be critical for relevance matching (Dai et al., 2018; Rao et al., 2019). On the other hand, the contextual encoder enables us to obtain long-distance contextual representations for each token. Comparing the deep and wide encoders, the deep encoder saves more parameters by reusing representations from the previous layer. The effectiveness of each encoder is an empirical question we will experimentally answer.

### 2.2 Relevance Matching

This section describes our efforts to capture keyword matching signals for relevance matching. We calculate the relevance score between the query and the context at each encoder layer by multiplying the query representation matrix $\mathbf{U}_q$ and the context representation matrix $\mathbf{U}_c$:

$$\mathbf{S} = \mathbf{U}_q {\mathbf{U}_c}^T, \mathbf{S} \in \mathbb{R}^{n \times m},$$

where $\mathbf{S}_{i,j}$ can be considered the similarity score by matching the query phrase vector $\mathbf{U}_q[i]$ with the context phrase vector $\mathbf{U}_c[j]$. Since the query and the context share the same encoder layers, similar phrases will be placed closer in a high-dimensional embedding space and their product will produce larger scores. Next, we obtain a normalized similarity matrix $\tilde{\mathbf{S}}$ by applying a *softmax* function over the context columns of $\mathbf{S}$ to normalize the similarity scores into the $[0, 1]$ range.

For each query phrase $i$, the above *softmax* function normalizes its matching scores over all phrases in the context and helps discriminate matches with higher scores. An exact match will dominate others and contribute a similarity score close to 1.0. We then apply *max* and *mean* pooling to the similarity matrix to obtain discriminative feature vectors:

$$\begin{aligned} \mathit{Max}(\mathbf{S}) &= [\max(\tilde{\mathbf{S}}_{1,:}), ..., \max(\tilde{\mathbf{S}}_{n,:})], \\ \mathit{Mean}(\mathbf{S}) &= [mean(\tilde{\mathbf{S}}_{1,:}), ..., mean(\tilde{\mathbf{S}}_{n,:})], \\ \mathit{Max}(\mathbf{S}), \mathit{Mean}(\mathbf{S}) &\in \mathbb{R}^n \end{aligned}$$

Each score generated from pooling can be viewed as matching evidence for a specific query phrase in the context, where the value denotes the significance of the relevance signal. Compared to *Max* pooling, *Mean* pooling is beneficial for cases where a query phrase is matched to multiple relevant terms in the context.

It's worth noting that term importance modeling can be important for some search tasks (Guo et al., 2016); therefore, we inject external weights as priors to measure the relative importance of different query terms and phrases. We multiply the score after pooling with the weights of that specific query term/phrase. These are provided as feature inputs to the final classification layer, denoted by $\mathbf{O}_{RM}$:

$$\begin{aligned} \mathbf{o}_{RM} = \{&wgt(q) \odot \mathit{Max}(\mathbf{S}), wgt(q) \odot \mathit{Mean}(\mathbf{S})\}, \\ &\mathbf{O}_{RM} \in 2 \cdot \mathbb{R}^n, \end{aligned} \tag{1}$$

where $\odot$ is an element-wise product between the weights of the query terms/phrases with the pooling scores, and $wgt(q)^i$ denotes the weight of the $i$-th term/phrase in the query; its value changes in the intermediate encoder layers since deeper/wider encoder layers capture longer phrases. We choose inverse document frequency (IDF) as our weighting function. A higher IDF weight implies a rarer occurrence in the collection and thus greater discriminative power. The weighting method also allows us to reduce the impact of large matching scores for common words like stopwords.

### 2.3 Semantic Matching

In addition to relevance matching, we aim to capture semantic matching signals via co-attention mechanisms on intermediate query and context representations. Our semantic matching method behaves similarly to the transformer (Vaswani et al., 2017), which also uses attention (specifically, self-attention) over hierarchical blocks to capture semantics at different granularities.

Given $\mathbf{U}_q \in \mathbb{R}^{n \times F}$ and $\mathbf{U}_c \in \mathbb{R}^{m \times F}$ generated by an intermediate encoder layer, we first calculate the bilinear attention as follows:

$$\begin{aligned} \mathbf{A} = \mathit{REP}(\mathbf{U}_q \mathbf{W}_q) + \mathit{REP}(\mathbf{U}_c \mathbf{W}_c) + \mathbf{U}_q \mathbf{W}_b \mathbf{U}_c^T \\ \mathbf{A} = \mathit{softmax}_{\text{col}}(\mathbf{A}) \\ \mathbf{A} \in \mathbb{R}^{n \times m} \end{aligned}$$

where $\mathbf{W}_q, \mathbf{W}_c \in \mathbb{R}^F$, $\mathbf{W}_b \in \mathbb{R}^{F \times F}$, and the *REP* operator converts the input vector to a $\mathbb{R}^{n \times m}$ matrix by repeating elements in the missing dimen-

sions. $\text{Softmax}_{\text{col}}$ is the column-wise softmax operator. Similar to Seo et al. (2017), we perform co-attention from two directions: query-to-context and context-to-query, as follows:

$$\tilde{\mathbf{U}}_q = \mathbf{A}^T \mathbf{U}_q$$
$$\tilde{\mathbf{U}}_c = \mathit{REP}(\max_{\text{col}}(\mathbf{A})\mathbf{U}_c)$$
$$\tilde{\mathbf{U}}_q \in \mathbb{R}^{m \times F}, \tilde{\mathbf{U}}_c \in \mathbb{R}^{m \times F}$$

where $\max_{\text{col}}$ is the column-wise max-pooling operator. $\tilde{\mathbf{U}}_q$ denotes query-aware context embeddings by attending the raw query representations to the attention weights, while $\tilde{\mathbf{U}}_c$ indicates the weighted sum of the most important words in the context with respect to the query.

We then take an enhanced concatenation to explore the interaction between $\tilde{\mathbf{U}}_q$ and $\tilde{\mathbf{U}}_c$, as in Equation 2. Finally, we apply an additional BiLSTM to the concatenated contextual embeddings $\mathbf{H}$ to capture contextual dependencies in the sequence, and use the last hidden state (with dimension $d$) as the output features of the semantic matching module $\mathbf{O}_{\text{SM}}$:

$$\begin{aligned} \mathbf{H} &= [\mathbf{U}_c; \tilde{\mathbf{U}}_q; \mathbf{U}_c \otimes \tilde{\mathbf{U}}_q; \tilde{\mathbf{U}}_c \otimes \tilde{\mathbf{U}}_q] \\ \mathbf{O}_{\text{SM}} &= \text{BiLSTM}(\mathbf{H}) \\ \mathbf{H} &\in \mathbb{R}^{m \times 4F}, \mathbf{O}_{\text{SM}} \in \mathbb{R}^{d} \end{aligned} \tag{2}$$

### 2.4 Final Classification

Given the relevance and semantic matching features $\{\mathbf{O}^l_{\text{RM}}, \mathbf{O}^l_{\text{SM}}\}$ (from Equations 1 and 2) learned at each encoder layer $l$, we concatenate them together and use a two-layer fully-connected layer with ReLU activation to generate the final prediction vector $o$. During training, we minimize the negative log likelihood loss $L$ summed over all samples $(o_i, y_i)$ below:

$$\begin{aligned} o =& \mathit{softmax}(\text{MLP}(\{\mathbf{O}^l_{\text{RM}}, \mathbf{O}^l_{\text{SM}}\})), \\ l =& 1, 2, ..., N \quad \text{and} \quad o \in \mathbb{R}^{\|class\|} \\ L =& -\sum_{(o_i, y_i)} \log o_i[y_i], \end{aligned}$$

where $N$ is the number of encoder layers.

## 3 Experimental Setup

### 3.1 Benchmarks and Metrics

We evaluated our proposed HCAN model on three NLP tasks and two IR datasets, as follows:

**Answer Selection.** This task is to rank candidate answer sentences based on their similarity to the question. We use the TrecQA (Wang et al., 2007) dataset (raw version)[1] with 56k question-answer pairs. We report mean average precision (MAP) and mean reciprocal rank (MRR).

**Paraphrase Identification.** This task is to identify whether two sentences are paraphrases of each other. We use the TwitterURL (Lan et al., 2017) dataset with 50k sentence pairs. We report the unweighted average of F1 scores on the positive and negative classes (macro-F1).

**Semantic Textual Similarity (STS).** This task is to measure the degree of semantic equivalence between pairs of texts. We use the Quora (Iyer et al., 2017) dataset with 400k question pairs collected from the Quora website. We report class prediction accuracy.

**Tweet Search.** This task is to rank candidate tweets by relevance with respect to a short query. We use the TREC Microblog 2013–2014 datasets (Lin and Efron, 2013; Lin et al., 2014), as prepared by Rao et al. (2019), where each dataset contains around 50 queries and 40k query-tweet pairs. We report MAP and precision at rank 30 (P@30).

### 3.2 Baselines and Implementations

For the answer selection, paraphrase identification, and STS tasks, we compared against the following baselines: InferSent (Conneau et al., 2017), ESIM (Chen et al., 2017b), DecAtt (Parikh et al., 2016), and PWIM (He and Lin, 2016). Additionally, we report state-of-the-arts results on each dataset from published literature. We also include the current state-of-the-art BERT (Devlin et al., 2019) results on each dataset.

For the tweet search task, we mostly follow the experimental setting in Rao et al. (2019). Baselines include the classic query likelihood (QL) method, RM3 query expansion (Abdul-Jaleel et al., 2004), learning to rank (L2R), as well as a number of neural ranking models: DRMM (Guo et al., 2016), DUET (Mitra et al., 2017), K-NRM (Xiong et al., 2017b), and PACRR (Hui et al., 2017). For the neural baselines, we used implementations in MatchZoo.[2] For L2R, we used LambdaMART (Burges, 2010) on the same feature sets as Rao et al. (2019): text-based, URL-

[1] The leaderboard can be found in https://aclweb.org/aclwiki/Question_Answering_(State_of_the_art)

[2] https://github.com/NTMC-Community/MatchZoo

| Model | TrecQA | | TwitterURL | Quora |
|---|---|---|---|---|
| | MAP | MRR | macro-F1 | Acc |
| InferSent | 0.521 | 0.559 | 0.797 | 0.866 |
| DecAtt | 0.660 | 0.712 | 0.785 | 0.845 |
| ESIM$_{seq}$ | 0.771 | 0.795 | **0.822** | 0.850 |
| ESIM$_{tree}$ | 0.698 | 0.734 | - | 0.755 |
| ESIM$_{seq+tree}$ | 0.749 | 0.768 | - | 0.854 |
| PWIM | 0.739 | 0.795 | 0.809 | 0.834 |
| State-of-the-Art Models | | | | |
| Rao et al. (2016) | **0.780** | 0.834 | - | - |
| Gong et al. (2018) | - | - | - | **0.891** |
| BERT | 0.838 | 0.887 | 0.852 | 0.892 |
| Our Approach | | | | |
| RM | 0.756 | 0.812 | 0.790 | 0.842 |
| SM | 0.663 | 0.725 | 0.708 | 0.817 |
| HCAN | 0.774 | **0.843** | 0.817 | 0.853 |

Table 2: Results on TrecQA, TwitterURL, and Quora. The best scores except for BERT are bolded. In these experiments, all our approaches use the *deep* encoder in Sec. 2.1. RM and SM denote that only relevance and semantic matching signals are used, respectively. HCAN denotes the complete HCAN model.

| Model | TREC-2013 | | TREC-2014 | |
|---|---|---|---|---|
| | MAP | P@30 | MAP | P@30 |
| QL | 0.2532 | 0.4450 | 0.3924 | 0.6182 |
| RM3 | 0.2766 | 0.4733 | **0.4480** | 0.6339 |
| L2R | 0.2477 | 0.4617 | 0.3943 | 0.6200 |
| Neural Baselines | | | | |
| DUET | 0.1380 | 0.2528 | 0.2680 | 0.4091 |
| DRMM | 0.2102 | 0.4061 | 0.3440 | 0.5424 |
| K-NRM | 0.1750 | 0.3178 | 0.3472 | 0.5388 |
| PACRR | 0.2627 | 0.4872 | 0.3667 | 0.5642 |
| BERT | 0.3357 | 0.5656 | 0.5176 | 0.7006 |
| Our Approach | | | | |
| RM | 0.2818 | 0.5222 | 0.4304 | 0.6297 |
| SM | 0.1365 | 0.2411 | 0.2414 | 0.3279 |
| HCAN | **0.2920** | **0.5328** | 0.4365 | **0.6485** |

Table 3: Results on TREC Microblog 2013–2014, organized in the same manner as Table 2.

based, and hashtag-based. Finally, we include the BERT results from Yang et al. (2019b).

In our experiments, we use trainable 300$d$ word2vec (Mikolov et al., 2013) embeddings with the SGD optimizer. For out-of-vocabulary words, we initialize their word embeddings with a uniform distribution from [0, 0.1]. Since our model also uses external weights, we calculate the IDF values from the training corpus. The number of convolutional layers $N$ is set to 4, and the convolutional filter size $k$ is set to 2. Hidden dimension $d$ is set to 150. We tune the learning rate in [0.05, 0.02, 0.01], the number of convolutional filters $F$ in [128, 256, 512], batch size in [64, 128, 256], and the dropout rate between 0.1 and 0.5. Our code and datasets are publicly available.[3]

## 4 Results

Our main results on the TrecQA, TwitterURL, and Quora datasets are shown in Table 2 and results on TREC Microblog 2013–2014 are shown in Table 3. The best numbers for each dataset (besides BERT) are bolded. We compare to three variants of our HCAN model: (1) only relevance matching signals (RM), (2) only semantic matching signals (SM), and (3) the complete model (HCAN). In these experiments, we use the *deep* encoder.

From Table 2, we can see that on all three datasets, relevance matching (RM) achieves significantly higher effectiveness than semantic matching (SM). It beats other competitive baselines (InferSent, DecAtt and ESIM) by a large margin on the TrecQA dataset, and is still comparable to those baselines on TwitterURL and Quora. This finding suggests that soft term matching signals alone are fairly effective for many textual similarity modeling tasks. However, SM performs much worse on TrecQA and TwitterURL, while the gap between SM and RM is reduced on Quora. By combining SM and RM signals, we observe consistent effectiveness gains in HCAN across all three datasets, establishing new state-of-the-art (non-BERT) results on TrecQA.

In Table 3, we observe that the query expansion method (RM3) outperforms most of the neural ranking models except for BERT, which is consistent with Yang et al. (2019a). We suggest two reasons: (1) tweets are much shorter and the informal text is "noisier" than longer documents in the web or newswire settings, which are what the previous neural models were designed for; (2) most neural baselines build directly on top of the embedding similarity matrix without any representation learning, which can be less effective.

Comparing our proposed approaches in Table 3, RM achieves fairly good scores while SM is not effective at all, affirming our hypothesis that term matching signals are essential to IR tasks. This finding further supports our motivation for bridging SM and RM. Indeed, semantic matching methods alone are ineffective when queries are comprised of only a few keywords, without much semantic information to exploit. However, the

[3] https://github.com/jinfengr/hcan.git

| Encoder | Model | TrecQA | | TwitURL | Quora | TREC-2013 | | TREC-2014 | |
|---|---|---|---|---|---|---|---|---|---|
| | | MAP | MRR | macro-F1 | Acc | MAP | P@30 | MAP | P@30 |
| Deep | RM | 0.756 | 0.812 | 0.790 | 0.842 | 0.282 | 0.522 | 0.430 | 0.630 |
| | SM | 0.663 | 0.725 | 0.708 | 0.817 | 0.137 | 0.241 | 0.241 | 0.328 |
| | HCAN | **0.774** | 0.843 | **0.817** | **0.853** | **0.292** | **0.533** | **0.437** | **0.649** |
| Wide | RM | 0.758 | 0.806 | 0.790 | 0.830 | 0.278 | 0.510 | 0.421 | 0.617 |
| | SM | 0.673 | 0.727 | 0.719 | 0.811 | 0.138 | 0.247 | 0.247 | 0.336 |
| | HCAN | 0.770 | **0.847** | 0.795 | 0.843 | 0.285 | 0.524 | 0.435 | 0.642 |
| Contextual | RM | 0.690 | 0.736 | 0.811 | 0.804 | 0.272 | 0.503 | 0.417 | 0.613 |
| | SM | 0.668 | 0.735 | 0.730 | 0.805 | 0.133 | 0.256 | 0.242 | 0.324 |
| | HCAN | 0.739 | 0.790 | 0.815 | 0.826 | 0.285 | 0.524 | 0.434 | 0.635 |

Table 4: Evaluation of different encoders in Sec. 2.1 (best numbers on each dataset are bolded).

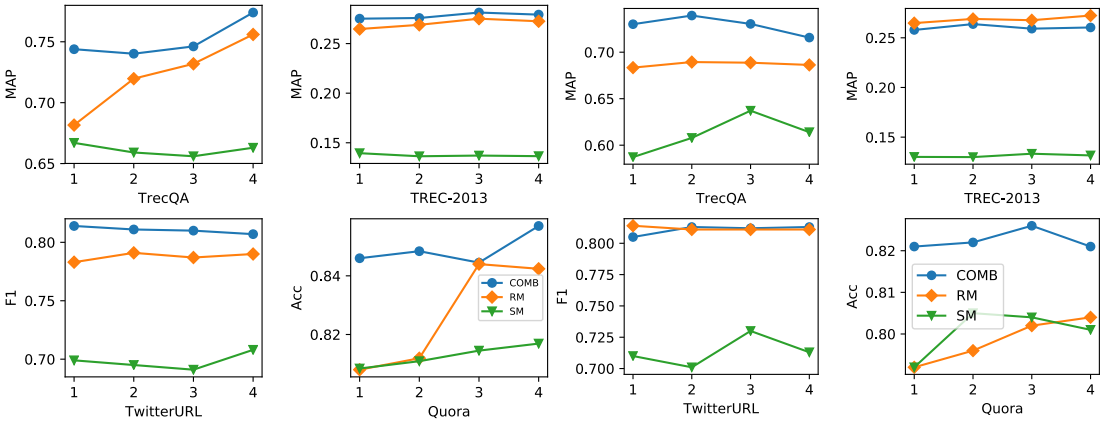


**(a)** Deep encoder **(b)** Contextual encoder

Figure 2: Model effectiveness with different numbers of encoder layers.

context-aware representations learned from SM *do* contribute to RM, leading to the superior results of our complete HCAN model.

### 4.1 Encoder Comparisons

We report results with the three different encoders from Sec 2.1 in Table 4. Overall, the effectiveness of the *deep* and *wide* encoders are quite close, given that the two encoders capture the same types of $n$-gram matching signals. The *contextual* encoder performs worse than the other two on TrecQA, but is comparable on all other datasets. This finding is consistent with Rao et al. (2017a), which shows that keyword matching signals are important for TrecQA. Also, we notice that the gaps between RM and SM are smaller for all encoders on Quora. We suspect that SM is more data-hungry than RM given its larger parameter space (actually, the RM module has no learnable parameters) and Quora is about 10× larger than the other datasets. For all encoders, combing RM and SM consistently improves effectiveness, affirming that relevance and semantic matching signals are complementary regardless of the underlying encoder choice.

To better understand the different encoder mechanisms, we vary the number of encoder layers for the *deep* and *contextual* encoders in Figure 2 (since the *wide* encoder behaves similarly to the *deep* encoder, we omit the analysis here). Our complete HCAN model has $N = 4$. In Figure 2a, we can see overall increases in effectiveness for the RM and HCAN (comb) as $N$ increases, showing that long-range phrase modeling is critical. However, increasing context window lengths don't help SM on the TrecQA and TREC-2013 datasets, likely because of dominant bigram matching signals ($N = 1$). Also, the complete HCAN model is consistently better than SM and RM alone in most settings, affirming its superior effectiveness, consistent with results in the above tables. In contrast, increasing the number of Bi-LSTM layers can sometimes even hurt, as shown in Figure 2b. This is not a surprise since a sin-

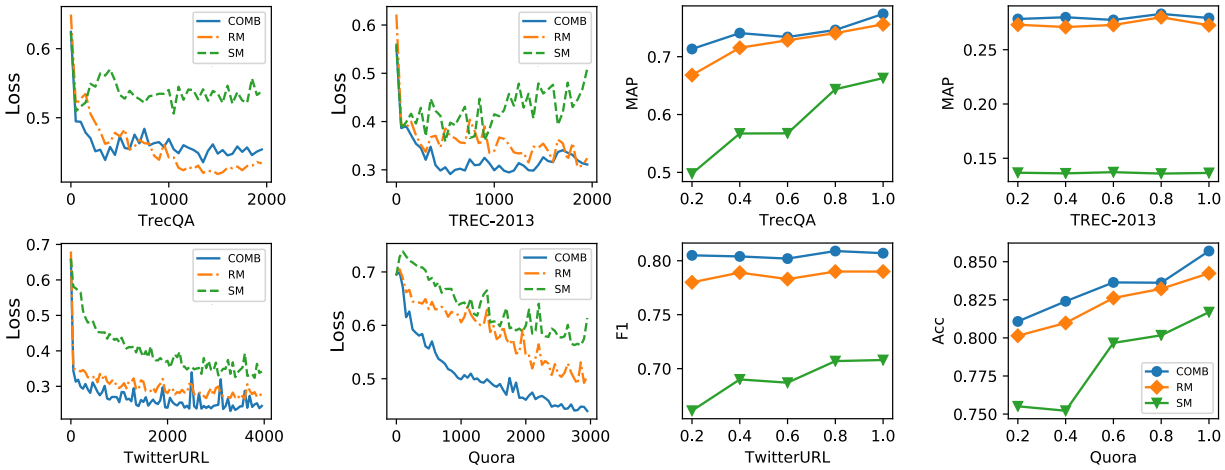


**(a)** Validation losses w.r.t. number of batches. **(b)** Effectiveness w.r.t. different percentage of training data.

Figure 3: Experiments exploring learning efficiency.

gle BiLSTM layer ($N = 1$) can already capture long-range contextual information and increasing the number of layers can introduce more parameters and lead to overfitting.

### 4.2 Learning Efficiency

We also designed two experiments to examine whether the complete HCAN model is learning more efficiently than SM or RM alone. In these experiments, we used the *deep* encoder. First, we show validation losses for different numbers of batches in Figure 3a. Second, we vary the training data size by randomly selecting different percentages (from 20% to 100%) of the original training set, shown in Figure 3b.

In Figure 3a, we can see that the validation loss for the complete model drops much faster than RM and SM alone, especially on TwitterURL and Quora. In Figure 3b, we can see that, as expected, all methods in general achieve higher scores when more data are used for training. An exception is SM on the TREC-2013 Twitter dataset, which we see is not effective in Table 3. Another important finding is that both RM and HCAN are more data efficient: for TREC-2013 and TwitterURL, both can achieve effectiveness comparable to the full training set with only 20% data.

### 4.3 Qualitative Sample Analysis

We present sample outputs in Table 5 to gain more insight into model behavior. For space considerations, we only show the Quora dataset, but our analysis reveals similar findings on the other datasets. The column "label" denotes the label of the sentence pair: 1 means semantically equivalent and 0 means not equivalent. For each model, we output its predicted label along with its confidence score; phrases with large attention weights are highlighted in orange and red.

In the first example, SM is able to correctly identify that the two sentences convey the same meaning with high confidence, while RM fails as the two sentences have no influential phrase matches (with high IDF weights). The sentence pair in the second example has a large text overlap. It is no surprise that RM would predict a high relevance score, while SM fails to capture their relatedness. In both examples, HCAN is able to integrate SM and RM to make correct predictions. Since the third example presents a similar pattern, we omit a detailed explanation. Overall, our quantitative and qualitative analyses show that relevance matching is better at capturing overlap-based signals, while combining semantic matching signals improve representation learning.

## 5 Related Work

### 5.1 Neural Relevance Matching

Recently, deep learning has achieved great success in many NLP and IR applications (He and Lin, 2016; Sutskever et al., 2014; Yin et al., 2016; Rao et al., 2017b). Current neural models for IR can be divided into representation-based and interaction-based approaches, discussed below:

Early neural IR models mainly focus on representation-based modeling between the query and documents, such as DSSM (Huang et al., 2013), C-DSSM (Shen et al., 2014), and SM-CNN (Severyn and Moschitti, 2015). These meth-

| Label | SM Score | RM Score | HCAN Score | Sample Pair |
|---|---|---|---|---|
| 1 | 1, 0.9119 | 0, 0.9353 | 1, 0.5496 | - How does it feel to kill a human ?<br>- How does it feel to be a murderer ? |
| 1 | 0, 0.9689 | 1, 0.8762 | 1, 0.8481 | - What are the time dilation effects on the ISS ?<br>- According to the theory of relativity , time runs slowly under the influence of gravity . Is there any time dilation experienced on the ISS ? |
| 0 | 0, 0.9927 | 1, 0.8473 | 1, 0.7280 | - Does RBI send its employees for higher education such as MBA , like sponsoring the education or allowing paid / unpaid leaves ?<br>- Does EY send its employees for higher education such as MBA , like sponsoring the education or allowing paid / unpaid leaves ? |

Table 5: Sample pairs from Quora. Phrases with large attention weights are highlighted in orange and red.

ods directly learn from query and document representations, and have been found to be ineffective when data is scarce.

Interaction-based approaches build on the similarity matrix computed from word pairs between the query and the document, often with count-based techniques to address data sparsity. For example, DRMM (Guo et al., 2016) introduced a pyramid pooling technique to convert the similarity matrix into histogram representations, on top of which a term gating network aggregates weighted matching signals from different query terms. Inspired by DRMM, Xiong et al. (2017b) proposed K-NRM, which introduced a differentiable kernel-based pooling technique to capture matching signals at different strength levels. Sharing similarities with our architecture is Rao et al. (2019), who developed a multi-perspective relevance matching method with a hierarchical convolutional encoder to capture character-level to sentence-level relevance signals from heterogeneous sources.

### 5.2 Neural Semantic Matching

Semantic matching is a fundamental problem for a variety of NLP tasks. For example, in paraphrase identification, SM is used to determine whether two sentences or phrases convey the same meaning. In question answering or reading comprehension (Xiong et al., 2017a; Tay et al., 2019a), SM can help identify the correct answer span given a question. Semantic understanding and reasoning for two pieces of texts lie at the core of SM. Existing state-of-the-art techniques for SM usually comprise three major components: (1) sequential sentence encoders that incorporate word context and sentence order for better sentence representations; (2) interaction and attention mechanisms (Tay et al., 2019b; Seo et al., 2017; Parikh et al., 2016; Conneau et al., 2017; Gong et al., 2018) to emphasize salient word pair interactions; (3) structure modeling (Chen et al., 2017b).

## 6 Conclusion

In this work, we examine the relationship between relevance matching and semantic matching, and highlight a few important differences between them. This is an important problem that lies at the core of many NLP and IR tasks. We propose the HCAN model with a relevance matching module to capture weighted $n$-gram matching signals and a semantic matching module for context-aware representation learning.

Thorough experiments show that relevance matching alone performs reasonably well for many NLP tasks, while semantic matching alone is not effective for IR tasks. We show that relevance matching and semantic matching are complementary, and HCAN combines the best of both worlds to achieve competitive effectiveness across a large number of tasks, in some cases, achieving the state of the art for models that do not exploit large-scale pretraining. We also find that our model can learn in a data efficient manner, further demonstrating the complementary nature of relevance matching and semantic matching signals.

## Acknowledgments

This research was supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada.

## References


Nasreen Abdul-Jaleel, James Allan, W. Bruce Croft, Fernando Diaz, Leah Larkey, Xiaoyan Li, Donald Metzler, Mark D. Smucker, Trevor Strohman, Howard Turtle, and Courtney Wade. 2004. UMass at TREC 2004: Novelty and HARD. In *Proceedings of the Thirteenth Text REtrieval Conference (TREC 2004)*, Gaithersburg, Maryland.

Christopher J. C. Burges. 2010. From RankNet to LambdaRank to LambdaMART: An overview. Technical Report MSR-TR-2010-82, Microsoft Research.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017a. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017b. Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1657–1668, Vancouver, Canada.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark.

Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. 2018. Convolutional neural networks for soft-matching n-grams in ad-hoc search. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, WSDM '18, pages 126–134, Marina Del Rey, CA, USA.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota.

Yichen Gong, Heng Luo, and Jian Zhang. 2018. Natural language inference over interaction space. In *Proceedings of the Sixth International Conference on Learning Representations (ICLR 2018)*.

Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. 2016. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 55–64, Indianapolis, Indiana, USA.

Hua He and Jimmy Lin. 2016. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 937–948, San Diego, California.

Hua He, John Wieting, Kevin Gimpel, Jinfeng Rao, and Jimmy Lin. 2016. UMD-TTIC-UW at SemEval-2016 task 1: Attention-based multi-perspective convolutional neural networks for textual similarity measurement. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1103–1108, San Diego, California.

Phu Mon Htut, Samuel Bowman, and Kyunghyun Cho. 2018. Training a ranking function for open-domain question answering. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 120–127, New Orleans, Louisiana.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*, CIKM '13, pages 2333–2338, San Francisco, California, USA.

Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. 2017. PACRR: A position-aware neural IR model for relevance matching. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1049–1058, Copenhagen, Denmark.

Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. 2018. Co-PACRR: A context-aware neural IR model for ad-hoc retrieval. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, WSDM '18, pages 279–287, Marina Del Rey, CA, USA.

Shankar Iyer, Nikhil Dandekar, and Kornél Csernai. 2017. *First Quora Dataset Release: Question Pairs*.

Wuwei Lan, Siyu Qiu, Hua He, and Wei Xu. 2017. A continuously growing dataset of sentential paraphrases. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1224–1234, Copenhagen, Denmark.

Wuwei Lan and Wei Xu. 2018. Neural network models for paraphrase identification, semantic textual similarity, natural language inference, and question answering. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3890–3902, Santa Fe, New Mexico, USA.

Jimmy Lin and Miles Efron. 2013. Overview of the TREC-2013 Microblog Track. In *Proceedings of the Twenty-Second Text REtrieval Conference (TREC 2013)*.

Jimmy Lin, Miles Efron, Yulu Wang, and Garrick Sherman. 2014. Overview of the TREC-2014 Microblog Track. In *Proceedings of the Twenty-Third Text REtrieval Conference (TREC 2014)*, Gaithersburg, Maryland.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119.

Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to match using local and distributed representations of text for web search. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pages 1291–1299, Perth, Australia.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255, Austin, Texas.

Jinfeng Rao, Hua He, and Jimmy Lin. 2016. Noise-contrastive estimation for answer selection with deep neural networks. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 1913–1916, Indianapolis, Indiana, USA.

Jinfeng Rao, Hua He, and Jimmy Lin. 2017a. Experiments with convolutional neural network models for answer selection. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 1217–1220, Shinjuku, Tokyo, Japan.

Jinfeng Rao, Ferhan Ture, Hua He, Oliver Jojic, and Jimmy Lin. 2017b. Talking to your TV: Context-aware voice search with hierarchical recurrent neural networks. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, pages 557–566, Singapore.

Jinfeng Rao, Wei Yang, Yuhao Zhang, Ferhan Ture, and Jimmy Lin. 2019. Multi-perspective relevance matching with hierarchical ConvNets for social media search. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)*, pages 232–240, Honolulu, Hawaii.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *Proceedings of the Fifth International Conference on Learning Representations (ICLR 2017)*.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 373–382, Santiago, Chile.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14 Companion, pages 373–374, Seoul, Korea.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112.

Yi Tay, Shuohang Wang, Anh Tuan Luu, Jie Fu, Minh C. Phan, Xingdi Yuan, Jinfeng Rao, Siu Cheung Hui, and Aston Zhang. 2019a. Simple and effective curriculum pointer-generator networks for reading comprehension over long narratives. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4922–4931, Florence, Italy.

Yi Tay, Aston Zhang, Anh Tuan Luu, Jinfeng Rao, Shuai Zhang, Shuohang Wang, Jie Fu, and Siu Cheung Hui. 2019b. Lightweight and efficient neural natural language processing with quaternion networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1494–1503, Florence, Italy.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. 2007. What is the Jeopardy model? A quasi-synchronous grammar for QA. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 22–32, Prague, Czech Republic.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI 2017)*, pages 4144–4150, Melbourne, Australia.

Caiming Xiong, Victor Zhong, and Richard Socher. 2017a. Dynamic coattention networks for question answering. In *Proceedings of the Fifth International Conference on Learning Representations (ICLR 2017)*.

Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017b. End-to-end neural ad-hoc ranking with kernel pooling. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 55–64, Shinjuku, Tokyo, Japan.

Wei Yang, Kuang Lu, Peilin Yang, and Jimmy Lin. 2019a. Critically examining the "neural hype": Weak baselines and the additivity of effectiveness gains from neural ranking models. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, pages 1129–1132, Paris, France.

Wei Yang, Haotian Zhang, and Jimmy Lin. 2019b. Simple applications of BERT for ad hoc document retrieval. *arXiv preprint arXiv:1903.10972*.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016. ABCNN: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics*, 4:259–272.

# A Syntax-aware Multi-task Learning Framework for Chinese Semantic Role Labeling

**Qingrong Xia, Zhenghua Li*, Min Zhang**
Institute of Artificial Intelligence, School of Computer Science and Technology,
Soochow University, China
kirosummer.nlp@gmail.com, {zhli13, minzhang}@suda.edu.cn

## Abstract

Semantic role labeling (SRL) aims to identify the predicate-argument structure of a sentence. Inspired by the strong correlation between syntax and semantics, previous works pay much attention to improve SRL performance on exploiting syntactic knowledge, achieving significant results. Pipeline methods based on automatic syntactic trees and multi-task learning (MTL) approaches using standard syntactic trees are two common research orientations. In this paper, we adopt a simple unified span-based model for both span-based and word-based Chinese SRL as a strong baseline. Besides, we present a MTL framework that includes the basic SRL module and a dependency parser module. Different from the commonly used hard parameter sharing strategy in MTL, the main idea is to extract implicit syntactic representations from the dependency parser as external inputs for the basic SRL model. Experiments on the benchmarks of Chinese Proposition Bank 1.0 and CoNLL-2009 Chinese datasets show that our proposed framework can effectively improve the performance over the strong baselines. With the external BERT representations, our framework achieves new state-of-the-art 87.54 and 88.5 F1 scores on the two test data of the two benchmarks, respectively. In-depth analysis are conducted to gain more insights on the proposed framework and the effectiveness of syntax.

## 1 Introduction

Semantic role labeling (SRL) is a fundamental and important task in natural language processing (NLP), which aims to identify the semantic structure (*Who* did *what* to *whom*, *when* and *where*, etc.) of each given predicate in a sentence. Semantic knowledge has been widely exploited in many down-stream NLP tasks, such as information extraction (Bastianelli et al., 2013), machine translation (Liu and Gildea, 2010; Gao and Vogel, 2011) and question answering (Shen and Lapata, 2007; Wang et al., 2015a).

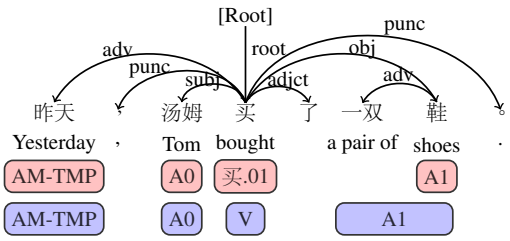


Figure 1: Example of span-based (blue blocks) and word-based (red blocks) SRL formulations in a sentence, where the top part is its dependency tree.

There are two formulations of SRL in the community according to the definition of semantic roles. The first is called *span-based* SRL, which employs a continuous word span as a semantic role and follows the manual annotations in the PropBank (Palmer et al., 2005) and NomBank (Meyers et al., 2004). The second is *word-based* SRL (Surdeanu et al., 2008), also called *dependency-based* SRL, whose semantic role is usually syntactic or semantic head word of the manually annotated word span. Figure 1 gives an example of the two forms in a sentence, where "bought" is the given predicate.

Intuitively, syntax and semantics are strongly correlative. For example, the semantic *A0* and *A1* roles are usually the syntactic subject and object, as shown in Figure 1. Inspired by the correlation, researchers try to improve SRL performance by exploring various ways to integrate syntactic knowledge (Roth and Lapata, 2016; He et al., 2018b; Swayamdipta et al., 2018). In contrast, some recent works (He et al., 2017; Tan et al., 2018; Cai et al., 2018) propose deep neural models for SRL without considering any syntactic in-

*Corresponding author.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5382–5392,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

formation, achieving promising results. Most recently, He et al. (2018a); Li et al. (2019) extend the span-based models to jointly tackle the predicate and argument identification sub-tasks of SRL.

Compared with the large amount of research for English SRL, Chinese SRL works are rare, mainly because of the limited amount of data and lack of attention of Chinese researchers. For Chinese, the commonly used datasets are Chinese Proposition Bank 1.0 (CPB1.0) (span-based) (Xue, 2008) and CoNLL-2009 Chinese (word-based) (Hajič et al., 2009). The CPB1.0 dataset follows the same annotation guideline with the English PropBank benchmark (Palmer et al., 2005). Wu and Palmer (2015) present a top model based selection preference approach to improve Chinese SRL. Since the amount of CPB1.0 dataset is small, Xia et al. (2017) exploit heterogeneous SRL data to improve the performance via a progressive learning approach. The CoNLL-2009 benchmark is released by the CoNLL-2009 shared task (Hajič et al., 2009). Previous works (Marcheggiani et al., 2017; He et al., 2018b; Cai et al., 2018) mainly focus on building more powerful models or exploring the usage of external knowledge on this dataset.

Inspired by the development of neural models and exploration of syntactic information, this paper proposes a MTL framework to extract syntactic representations as the external input features for the simple unified SRL model. The contributions of our paper are three-folds:

1. We introduce a simple unified model for span-based and word-based Chinese SRL.
2. We propose a MTL framework to extract implicit syntactic representations for SRL model, which significantly outperforms the baseline model.
3. Detailed analysis gains crucial insights on the effectiveness of our proposed framework.

We conduct experiments on the benchmarks of CPB1.0 and CoNLL-2009. The results show that our framework achieves new state-of-the-art 87.54 and 88.5 F1 scores on the two test data, respectively.

## 2 Basic SRL Model

Motivated by the recently presented span-based models (He et al., 2018a; Li et al., 2019) for jointly predicting predicates and arguments, we introduce a simple unified span-based model. Formally, given a sentence $s = w_1, w_2, ..., w_n$, the span-based model aims to predict a set of labeled predicate-argument relationships $\mathcal{Y} \subseteq \mathcal{P} \times \mathcal{A} \times \mathcal{R}$, where $\mathcal{P} = \{w_1, w_2, ..., w_n\}$ is the set of all candidate predicates, $\mathcal{A} = \{(w_i, ..., w_j) | 1 \leq i \leq j \leq n\}$ is the set of all candidate arguments, and $\mathcal{R}$ is the set of the semantic roles. Following He et al. (2018a), we also include a null label $\epsilon$ in the role set $\mathcal{R}$ indicating no relation between the focused predicate and argument. The model objective is to optimize the probability of the predicate-argument-role tuples $y \in \mathcal{Y}$ in a sentence $s$, which is formulated as:

$$
\begin{aligned}
P(y|s) &= \prod_{p \in \mathcal{P}, a \in \mathcal{A}, r \in \mathcal{R}} P(y_{(p,a,r)}|s) \\
&= \prod_{p \in \mathcal{P}, a \in \mathcal{A}, r \in \mathcal{R}} \frac{e^{\phi(p,a,r)}}{\sum_{r' \in \mathcal{R}} e^{\phi(p,a,r')}}
\end{aligned} \tag{1}
$$

where $\phi(p, a, r) = \phi_p(p) + \phi_a(a) + \phi_r(p, a)$ is the score of the predicate-argument-relation tuple. We directly adopt the model architecture of He et al. (2018a) as our basic SRL model with a modification on the argument representation. The architecture of the basic SRL module is shown in the right part of Figure 2, and we will describe it in the following subsections.

### 2.1 Input Layer

Following He et al. (2018a); Li et al. (2019), we employ CNNs to encode Chinese characters for each word $w_i$ into its character representation, denoted as $\boldsymbol{rep}_i^{char}$. Then, we concatenate $\boldsymbol{rep}_i^{char}$ with the word embedding $\boldsymbol{emb}_i^{word}$ to represent the word-level features as our basic model input. In addition, we also employ BERT representations (Devlin et al., 2019) to boost the performance of our baseline model, which we denote as $\boldsymbol{rep}_i^{BERT}$. Formally, the input representation of $w_i$ is:

$$
\boldsymbol{x}_i = \boldsymbol{rep}_i^{char} \oplus \boldsymbol{emb}_i^{word} \oplus \boldsymbol{rep}_i^{BERT} \tag{2}
$$

, where $\oplus$ is the concatenation operation. Our basic SRL model and BERT-enhanced baseline depend on whether including the BERT representation $\boldsymbol{rep}_i^{BERT}$ or not.

### 2.2 BiLSTM Encoder

Over the input layer, we employ the BiLSTMs with highway connections (Srivastava et al., 2015;

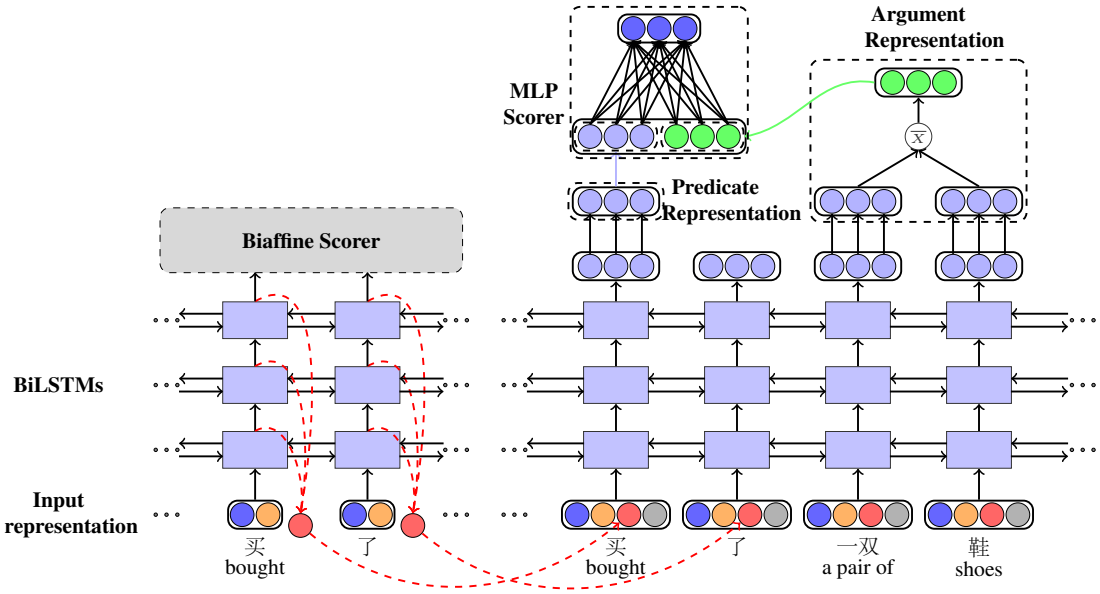


Figure 2: The detailed architecture of our proposed framework, where the left part is the dependency parser and the right part is the basic SRL module, respectively.

Zhang et al., 2016b) to encode long-range dependencies and obtain rich representations denoted as $\boldsymbol{h}_i$ for time stamp $i$. The highway connections are used to alleviate the gradient vanishing problem when training deep neural networks.

### 2.3 Predicate and Argument Representations

We directly employ the output of the top BiLSTM as the predicate representation at each time stamp. For all the candidate arguments, we simplify the representations by employing the mean operation over the BiLSTM outputs within the corresponding argument spans, which achieves similar results compared with the attention-based span representations (He et al., 2018a) on English SRL in our preliminary experiments. Formally,

$$\begin{aligned} \boldsymbol{rep}^p_i &= \boldsymbol{h}_i \\ \boldsymbol{rep}^a_{j,k} &= mean(\boldsymbol{h}_j, ..., \boldsymbol{h}_k) \\ & (1 \leq i \leq n; 1 \leq j \leq k \leq n) \end{aligned} \tag{3}$$

Specifically, for word-based SRL, we only need to set the length of candidate arguments to be 1.

### 2.4 MLP Scorer

We employ the MLP scorers as the scoring functions to determine whether the candidate predicates or arguments need to be pruned. Another MLP scorer is employed to compute the score of whether the focused candidate predicate and argument can compose a semantic relation.

$$\begin{aligned} \phi_p(p) &= \mathbf{w}_\mathbf{p}^\top \mathbf{MLP}_\mathbf{p}(\boldsymbol{rep}^p_i) \\ \phi_a(a) &= \mathbf{w}_\mathbf{a}^\top \mathbf{MLP}_\mathbf{a}(\boldsymbol{rep}^a_{j,k}) \\ \phi_r(p,a) &= \mathbf{w}_\mathbf{r}^\top \mathbf{MLP}_\mathbf{r}([\boldsymbol{rep}^p_i; \boldsymbol{rep}^a_{j,k}]) \end{aligned} \tag{4}$$

## 3 Proposed Framework

Our framework includes two modules, a basic SRL module and a dependency parser module, as shown in Figure 2. In this section, we will first describe the architecture of the employed dependency parser, and then illustrate the integration of the syntactic parser into the basic SRL model.

### 3.1 Dependency Parser Module

We employ the state-of-the-art biaffine parser proposed by Dozat and Manning (2017) as the dependency parser module in our framework, as shown by the left part of Figure 2. In order to better fit the dependency parser into our framework, we make some modifications on the original model architecture. First, we remove the Part-of-Speech (PoS) tagging embeddings and add the Chinese character CNN representations, so the resulting input representation is the same as the SRL module. Second, we substitute the BiLSTMs in the original biaffine parser with the same 3-layer highway BiLSTMs used in our SRL module. The biaffine scorer is proposed to compute the score of candidate syntactic head and modifier, which remains unchanged.

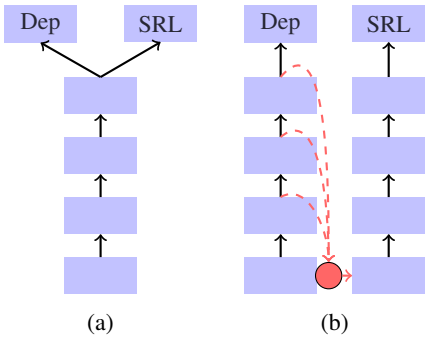


Figure 3: Comparison between the architectures of hard parameter sharing (a) and our proposed implicit representation integration (b).

### 3.2 Details of Integration

Multi-task learning (MTL) approaches can effectively exploit the standard dependency trees to improve the SRL performance, regarding dependency parsing as the auxiliary task. Hard parameter sharing (Ruder, 2017) is the most commonly used method in MTL, which shares several common layers between tasks and keeps the task-specific output layers, as illustrated by the sub-figure a in Figure 3. We propose a better way to integrate the focused two tasks in this work. In the following, we will describe the intuition and details of the integration on the two tasks.

As is well known, the hard parameter sharing approach can provide representations for all the shared tasks and reduce the probability of overfitting on the main task. However, this kind of sharing strategy somewhat weakens the representation framework maintains distinct model parameters for each task, due to the neutralization of knowledge introduced by the auxiliary task. Different from the hard parameter sharing strategy, we propose to integrate the syntactic information into the input layer of the basic SRL module, as illustrated by sub-figure b of Figure 3. And Figure 2 shows the detailed architecture. First, we extract all the 3 BiLSTM hidden outputs of the dependency parser as the syntactic representations. Second, we employ the normalized weights to sum the extracted representations as the final syntactic representation for word $w_i$, denoted as $\boldsymbol{rep}_i^{syn}$. Formally,

$$\boldsymbol{rep}_i^{syn} = \sum_{1 \le j \le N} \alpha_j * \boldsymbol{h}_i^j \tag{5}$$

where N is the layer number of the highway BiLSTMs, and $\alpha_j$ is the $j-$th softmax weight. Finally, the extracted syntactic representations are fed into the input layer of the SRL module, and concatenated with the original SRL module input. We design this framework for several considerations: 1) the proposed framework keeps the own model parameters for each task, thereby maximizing task-specific information for the main task, 2) the dependency parser module can be updated by the gradients returned from the extracted syntactic representations, which can encourage it to produce semantic preferred representations.

### 3.3 Training Objective

Given the sets of SRL data $\mathcal{S}$ and dependency data $\mathcal{D}$, the framework loss function is defined as the sum of the negative log-likelihood loss of the two tasks:

$$-\Big( \sum_{(Y_s^*, X_s) \in \mathcal{S}} \log P(Y_s^* | X_s) + \alpha \sum_{(Y_d^*, X_d) \in \mathcal{D}} \log P(Y_d^* | X_d) \Big) \tag{6}$$

where $Y_s^*$ and $Y_d^*$ are gold semantic and syntactic structures respectively, and $\alpha$ is a corpus weighting factor to control the loss contribution of the dependency data in each batch as discussed in the experiments.

## 4 Experiments

### 4.1 Settings

We evaluate the proposed MTL framework on two commonly used benchmark datasets of Chinese: Chinese Proposition Bank 1.0 (CPB1.0) (span-based) (Xue, 2008) and CoNLL-2009 (word-based) (Hajič et al., 2009). Following previous works, we report the results of span-based SRL in two setups: *pre-identified predicates* and *end-to-end*. For word-based SRL, we only report the results in the *pre-identified predicates* setting. Following Roth and Lapata (2016), we employ the mate-tools[1] (Björkelund et al., 2010) for the predicate disambiguation, which achieves 94.87% and 94.91% F1 scores on the CoNLL-2009 Chinese development and test data respectively.

**Dependency Parsing Data.** We employ the Chinese Open Dependency Treebank[2] constructed at Soochow University. The treebank construction

[1] https://code.google.com/archive/p/mate-tools/
[2] http://hlt.suda.edu.cn/index.php/CODT

project aims to continually build a large-scale Chinese dependency treebank that covers up-to-date texts from different domains and sources (Peng et al., 2019). So far, CODT contains 67,679 sentences from 9 different domains or sources.

**BERT Representations.** Recently, BERT (Bidirectional Encoder Representations from Transformers) is proposed by Devlin et al. (2019), which makes use of Transformers to learn contextual representations between words. In this paper, we use the pre-trained Chinese model[3] to extract the BERT representations for our span-based and word-based SRL datasets. We extract the fixed BERT representations from the last four hidden layers of the pre-trained model. Finally, we also employ the normalized weighted sum operation to obtain the final BERT representation for each word $w_i$, denoted as $\boldsymbol{rep}_i^{BERT}$.

**Hyperparameters.** We employ word2vec (Mikolov et al., 2013) to train the Chinese word embeddings on the Chinese Gigaword dataset[4]. The Chinese char embeddings are randomly initialized, and the dimension is 100. We employ the CNN to get the Chinese char representations, which has window size of 3, 4 and 5, and the output channel size is 100. For other parameter settings in the SRL module, we mostly follow the work of He et al. (2018a). As for the pruning of candidate predicates and arguments, we choose the pruning ratios according to the training data, using $\lambda_p = 0.4$ for predicates and $\lambda_a = 0.8$ for arguments with up to 30 words.

**Training Criterion.** We choose Adam (Kingma and Ba, 2015) optimizer with 0.001 as the initial learning rate and 0.1% as the decay rate for every 100 steps. Each data batch is composed of both SRL and dependency instances. We randomly shuffle the SRL and dependency training datasets if the smaller SRL data is used up. All baseline models are trained for at most 180,000 steps, and 100,000 steps for other models. In addition, we pick the best model on the development data for testing. We apply 0.5 dropout to the word embeddings and Chinese character representations and 0.2 dropout to all hidden layers. We employ variational dropout masks that are shared across all timesteps (Gal and Ghahramani, 2016) for the highway BiLSTMs, with 0.4 dropout rate.

**Evaluation.** We adopt the official scripts provided by CoNLL-2005[5] and CoNLL-2009[6] for span-based and word-based SRL evaluation, respectively. We conduct significant tests using the Dan Bikel's randomized parsing evaluation comparer.

### 4.2 Syntax-aware Methods

To illustrate the effectiveness and advantage of our proposed framework[7] (Integration of Implicit Representations, IIR), we conduct several experiments with the recently employed syntax-aware methods on CPB1.0 dataset for comparison:

- **Tree-GRU** Xia et al. (2019) investigate several syntax-aware methods for the English span-based SRL, showing the effectiveness of introducing syntactic knowledge into the SRL task. We only compare with the Tree-GRU method, since the other methods are all predicate-specific and hence not fit into our basic SRL model.

- **FIR** Following Yu et al. (2018) and Zhang et al. (2019), we extract the outputs of BiLSTMs as the fixed implicit representations (FIR) from a pre-trained biaffine parser. In detail, we train the biaffine parser with the same training data used in our framework, and employ the combination of development data of CDT (997 sentences) and PCTB7 (998 sentences) as the development data. The biaffine parser achieves 79.71% UAS and 74.74% LAS on the combined development data.

- **HPS** We employ the commonly used hard parameter sharing (HPS) strategy of MTL as a strong baseline, which shares the word and char embeddings and 3-layer BiLSTMs between the dependency parser and the basic SRL module.

### 4.3 Main Results

**Results of Syntax-aware Methods.** Table 1 shows the results of these syntax-aware methods on CPB1.0 dataset. First, the first line shows the results of our baseline model, which only employs the word embeddings and char representations as the inputs of the basic SRL model. Second, the

[3] https://github.com/google-research/bert#pre-trained-models

[4] https://catalog.ldc.upenn.edu/LDC2003T09

[5] http://www.cs.upc.edu/~srlconll/st05/st05.html

[6] https://ufal.mff.cuni.cz/conll2009-st/scorer.html

[7] https://github.com/xxx

| Methods | Dev | | | Test | | |
|---|---|---|---|---|---|---|
| | P | R | F1 | P | R | F1 |
| Baseline | 81.52 | 82.17 | 81.85 | 80.95 | 80.01 | 80.48 |
| Baseline + Dep (Tree-GRU) | 82.35 | 80.24 | 81.28 | 82.10 | 78.11 | 80.06 |
| Baseline + Dep (FIR) | **83.56** | 83.05 | 83.30 | 83.38 | 81.93 | 82.65 |
| Baseline + Dep (HPS) | 82.58 | **84.15** | 83.36 | 83.22 | **83.81** | 83.51 |
| Baseline + Dep (IIR) | 83.12 | 83.66 | **83.39** | **84.49** | 83.34 | **83.91** |

Table 1: Experimental results of syntax-aware methods we compare on CPB1.0 dataset.

Tree-GRU method only achieves 80.06 F1 score on the test data, which even didn't catch up with the baseline model. We think this is caused by the relatively low accuracy in Chinese dependency parsing. Third, the FIR approach outperforms the baseline by 2.17 F1 score on the test data, demonstrating the effectiveness of introducing fixed implicit syntactic representations. Forth, the HPS strategy achieves more significant performance by 83.51 F1 score. Finally, our proposed framework achieves the best performance of 83.91 F1 score among these methods, outperforming the baseline by 3.43 F1 score. All the improvements are statistically significant ($p < 0.0001$). From these experimental results, we can conclude that: 1) the quality of syntax has a crucial impact on the methods which depend on the systematic dependency trees, like Tree-GRU, 2) the implicit syntactic features have the potential to improve the down-stream NLP tasks, and 3) learning the syntactic features with the main task performs better than extract them from a fixed dependency parser.

**Results on CPB1.0.** Table 2 shows the results of our baseline model and proposed framework using external dependency trees on CPB1.0, as well as the corresponding results when adding BERT representations. It is clear that adding dependency trees into the baseline SRL model can effectively improve the performance ($p < 0.0001$), no matter whether employ the BERT representations or not. Especially, our proposed framework (IIR) consistently outperforms the hard parameter sharing strategy. So we only report the results of our proposed framework in later experiments. Our final results outperforms the best previous model (Xia et al., 2017) by 7.87 and 4.24 F1 scores with BERT representations or not, respectively.

Table 3 shows the results of our framework in the *end-to-end* setting. To our best knowledge, we are the first to present the results of *end-to-end* on the CPB1.0 dataset. We achieve the result of 85.57 in F1 score, which is a strong baseline for later works. It is clear that our framework can still achieve better results compared with the strong baseline, which employs BERT representations as the external input.

| **Methods** | **F1** |
|---|---|
| **Previous Works** | |
| Sun et al. (2009) | 74.12 |
| Wang et al. (2015b) | 77.59 |
| Sha et al. (2016) | 77.69 |
| Xia et al. (2017) | 79.67 |
| **Ours** | |
| Baseline | 80.48 |
| Baseline + Dep (HPS) | 83.51 |
| Baseline + Dep (IIR) | **83.91** |
| Baseline + BERT | 86.62 |
| Baseline + BERT + Dep (HPS) | 87.03 |
| Baseline + BERT + Dep (IIR) | **87.54** |

Table 2: Results and comparison with previous works on CPB1.0 test set.

**Results on CoNLL-2009.** Table 4 shows the results of our framework and comparison with previous works on the CoNLL-2009 Chinese test data. Our baseline achieves nearly the same per-

| **Methods** | **Dev** **F1** | **Test** **F1** |
|---|---|---|
| **Ours** | | |
| Baseline | 80.37 | 79.29 |
| Baseline + Dep (IIR) | **82.39** | **81.73** |
| Baseline + BERT | 85.30 | 85.26 |
| Baseline + BERT + Dep (IIR) | **85.92** | **85.57** |

Table 3: F1 scores of end-to-end settings on CPB1.0 test set.

| Methods | P | R | F1 |
|---|---|---|---|
| **Previous Works** | | | |
| Roth and Lapata (2016) | 83.2 | 75.9 | 79.4 |
| Marcheggiani et al. (2017) | 84.6 | 80.4 | 82.5 |
| He et al. (2018b) | 84.2 | 81.5 | 82.8 |
| Cai et al. (2018) | 84.7 | 84.0 | 84.3 |
| **Ours** | | | |
| Baseline | 83.7 | 84.8 | 84.2 |
| Baseline + Dep (IIR) | **84.6** | **85.7** | **85.1** |
| Baseline + BERT | 87.8 | **89.2** | **88.5** |
| Baseline + BERT + Dep (IIR) | **88.0** | 89.1 | **88.5** |

Table 4: Results and comparison with previous works on CoNLL-2009 Chinese test set.

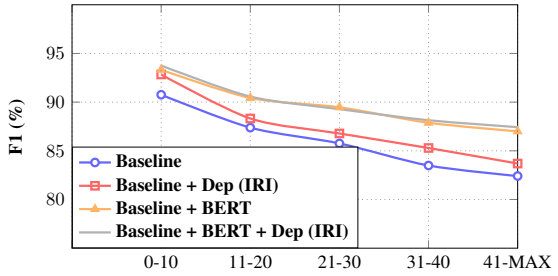


Figure 4: F1 scores regarding to the sentence length of the CoNLL-2009 Chinese dev data.

formance with Cai et al. (2018), which is an end-to-end neural model that consists of BiLSTM encoder and biaffine scorer. Our proposed framework outperforms the best reported result (Cai et al., 2018) by 0.8 F1 score and brings a significant improvement ($p < 0.0001$) of 0.9 F1 score over our baseline model. Our experimental result boosts to 88.5 F1 score when the framework is enhanced with BERT representations. However, compared with the results in the settings without BERT, the improvement is fairly small (88.53 - 88.47 = 0.06 F1 score, $p > 0.1$)[8] of the proposed framework, which we will discuss in Section 5.3.

## 5 Analysis

In this section, we conduct detailed analysis to understand the improvements introduced by our proposed framework.

### 5.1 Long-distance Dependencies

To analyze the effect of the proposed framework regarding to the distance of sentence lengths, we report the F1 scores of different sets of sentence lengths, as shown in Figure 4. We can see that improvements are obtained for nearly all sets of sentences, especially on the sentences with long-distance. It demonstrate that *syntactic knowledge is beneficial for SRL and effective to capture long-distance dependencies*.

[8] Following previous works, we only retrain the experimental results with one decimal point

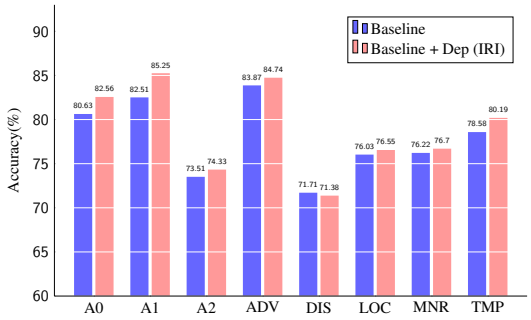


Figure 5: Accuracy comparison of different semantic roles between *Baseline* and *Baseline + Dep (IIR)* on CoNLL-2009 dev data.

### 5.2 Improvements on Semantic Roles

To find which semantic roles benefit from our syntax-aware framework, we report the F1 scores on several semantic role labels in Figure 5. We can see that syntax helps most on the *A0* and *A1* roles, which is consistent with the intuition that the semantic *A0* and *A1* roles are usually the syntactic subject and object of a verb predicate. Other adjunct semantic roles like *ADV*, *LOC*, *MNR* and *TMP* all benefit from the introduction of syntactic information. There is an interesting finding that the *DIS* role obtains worse performance when introduce syntactic information. We conduct error analysis on this phenomena, and we found that the framework mostly confuses *DIS* with *ADV*. The possible reason is that the two semantic roles are both labeled as *adv* in syntax.

### 5.3 Integration with BERT

BERT is employed to boost the performance of our basic SRL model and our proposed framework. Compared with results in the settings without BERT, the improvements of our framework over the BERT-enhanced baseline are fairly small on CoNLL-2009, as shown by the last two lines in Table 4. To analyze the difference between the two models (*Baseline + BERT* and *Baseline + BERT + Dep (IIR)*), we conduct an analysis on the sentence performance comparison between them, which is inspired by Zhang et al. (2016a). As shown in Fig-

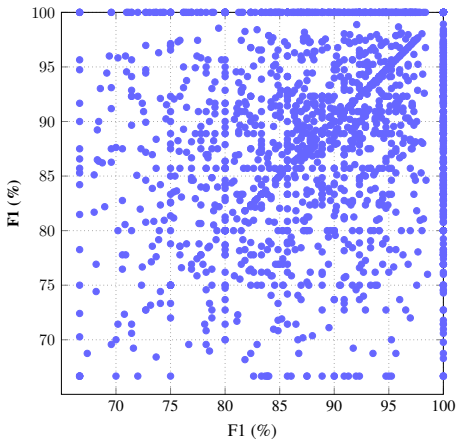


Figure 6: Sentence F1 scores comparison on CoNLL-2009 Chinese test data, where the x axis presents the F1 scores of *Baseline + BERT* and y axis shows the F1 scores of *Baseline + BERT + Dep (IIR)*, respectively.

ure 6, we can see that most of the scatter points are off the diagonal line, demonstrating strong differences between the two models. Based on this finding, how to better integrate syntactic knowledge and BERT representations becomes an interesting and meaningful question, and we leave it for future work.

## 6 Related Work

Traditional discrete-feature-based SRL works (Swanson and Gordon, 2006; Zhao et al., 2009) mainly make heavy use of syntactic information. Along with the impressive development of neural-network-based approaches in the NLP community, much attention has been paid to build more powerful neural model without considering any syntactic information. Zhou and Xu (2015) employ deep stacked BiLSTMs and achieve strong performance for span-based English SRL. He et al. (2017) extend their work (Zhou and Xu, 2015) by employing several advanced practices in recent deep learning literature, leading to significant improvements. Tan et al. (2018) present a strong self-attention based model, achieving significant improvements. Inspired by the span-based model proposed by Lee et al. (2017) for coreference resolution, He et al. (2018a); Ouchi et al. (2018) present similar span-based models for SRL which can exploit span-level features. For word-based SRL, Marcheggiani et al. (2017) propose a simple and fast syntax-agnostic model with rich input representations. Cai et al. (2018) present an end-to-end model with BiLSTMs and biaffine scorer to jointly handle the predicate disambiguation and the argument labeling sub-tasks.

Apart from the above syntax-free works, researchers also pay much attention on improving the neural-based SRL approaches by introducing syntactic knowledge. Roth and Lapata (2016) introduce the dependency path embeddings to the neural-based model and achieve substantial improvements. Marcheggiani and Titov (2017) employ the graph convolutional neural networks on top of the BiLSTM encoder to encode syntactic information. He et al. (2018b) propose a k-th order argument pruning algorithm based on systematic dependency trees. Strubell et al. (2018) propose a self-attention based neural MTL model which incorporate dependency parsing as a auxiliary task for SRL. Swayamdipta et al. (2018) propose a MTL framework using hard parameter strategy to incorporate constituent parsing loss into semantic tasks, i.e. SRL and coreference resolution, which outperforms their baseline by +0.8 F1 score. Xia et al. (2019) investigate and compare several syntax-aware methods on span-based SRL, showing the effectiveness of integrating syntactic information.

Compared with the large amount of works on English SRL, Chinese SRL works are rare, mainly because of the limitation of datasize and lack of attention of Chinese researchers. Sun et al. (2009) treat the Chinese SRL as a sequence labeling problem and build a SVM-based model by exploiting morphological and syntactic features. Wang et al. (2015b) build a basic BiLSTM model and introduce a way to exploit heterogeneous data by sharing word embeddings. Xia et al. (2017) propose a progressive model to learn and transfer knowledge from heterogeneous SRL data. The above works are all focus on the span-based Chinese SRL, and we compare with their results in Table 2. Different from them, we propose a MTL framework to integrate implicit syntactic representations into a simple unified model on both span-based and word-based SRL, achieving substantial improvements.

In addition to the hard parameter sharing strategy that we discuss in Section 3.2, partial parameter sharing strategy is also a commonly studied approach in MTL and domain adaptation. Kim et al. (2016) introduce simple neural extensions of feature argumentation by employing a global LSTM used across all domains and independent LSTMs used within individual domains. Peng

et al. (2017) explore a multitask learning approach which shares parameters across formalisms for semantic dependency parsing. In addition, Peng et al. (2018) present a multi-task approach for frame-semantic parsing and semantic dependency parsing with latent structured variables.

## 7 Conclusion

This paper proposes a syntax-aware MTL framework to integrate implicit syntactic representations into a simple unified SRL model. The experimental results show that our proposed framework can effectively improve the basic SRL model, even when the basic model is enhanced with BERT representations. Especially, our proposed framework is more effective at utilizing syntactic information, compared with the hard parameter sharing strategy of MTL. By utilizing BERT representations, our framework achieves new state-of-the-art performance on both span-based and word-based Chinese SRL benchmarks, i.e. CPB1.0 and CoNLL-2009 respectively. Detailed analysis shows that syntax helps most on the long sentences, because of the long-distance dependencies captured by syntax trees. Moreover, the comparison of sentence performance indicates that there is still a lot of work to do to better integrate syntactic information and BERT representation.

## Acknowledgments

We thank our anonymous reviewers for their helpful comments. This work was supported by National Natural Science Foundation of China (Grant No. 61525205, 61876116, 61432013) and a project funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions.

## References

Emanuele Bastianelli, Giuseppe Castellucci, Danilo Croce, and Roberto Basili. 2013. Textual inference and meaning representation in human robot interaction. In *Proceedings of JSSP*, pages 65–69.

Anders Björkelund, Bernd Bohnet, Love Hafdell, and Pierre Nugues. 2010. A high-performance syntactic and semantic dependency parser. In *Proceedings of COLING*, pages 33–36.

Jiaxun Cai, Shexia He, Zuchao Li, and Hai Zhao. 2018. A full end-to-end semantic role labeler, syntactic-agnostic over syntactic-aware? In *Proceedings of COLING*, pages 2753–2765.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Timothy Dozat and Christopher D Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proceedings of ICIR*.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Proceedings of NIPS*, pages 1019–1027.

Qin Gao and Stephan Vogel. 2011. Corpus expansion for statistical machine translation with semantic role label substitution rules. In *Proceedings of ACL*, pages 294–298.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, et al. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of CoNLL: Shared Task*, pages 1–18. ACL.

Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2018a. Jointly predicting predicates and arguments in neural semantic role labeling. In *Proceedings of ACL*, pages 364–369.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and whats next. In *Proceedings of ACL*, pages 473–483.

Shexia He, Zuchao Li, Hai Zhao, and Hongxiao Bai. 2018b. Syntax for semantic role labeling, to be, or not to be. In *Proceedings of ACL*, pages 2061–2071.

Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. 2016. Frustratingly easy neural domain adaptation. In *Proceedings of COLING*, pages 387–396.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of EMNLP*, pages 188–197.

Zuchao Li, Shexia He, Hai Zhao, Yiqing Zhang, Zhuosheng Zhang, Xi Zhou, and Xiang Zhou. 2019. Dependency or span, end-to-end uniform semantic role labeling. In *Proceedings of AAAI*.

Ding Liu and Daniel Gildea. 2010. Semantic role features for machine translation. In *Proceedings of COLING*, pages 716–724.

Diego Marcheggiani, Anton Frolov, and Ivan Titov. 2017. A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling. In *Proceedings of CoNLL*, pages 411–420.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of EMNLP*, pages 1506–1515.

Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. 2004. The nombank project: An interim report. In *Proceedings of HLT-NAACL*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, pages 3111–3119.

Hiroki Ouchi, Hiroyuki Shindo, and Yuji Matsumoto. 2018. A span selection model for semantic role labeling. In *Proceedings of EMNLP*, pages 1630–1642.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106.

Hao Peng, Sam Thomson, and Noah A Smith. 2017. Deep multitask learning for semantic dependency parsing. In *Proceedings of ACL*, pages 2037–2048.

Hao Peng, Sam Thomson, Swabha Swayamdipta, and Noah A Smith. 2018. Learning joint semantic parsers from disjoint data. In *Proceedings of NAACL-HLT*, pages 1492–1502.

Xue Peng, Zhenghua Li, Min Zhang, Rui Wang, Yue Zhang, and Luo Si. 2019. Overview of the nlpcc 2019 shared task: Cross-domain dependency parsing. In *Proceedings of The 8th CCF International Conference on Natural Language Processing and Chinese Computing (NLPCC)*.

Michael Roth and Mirella Lapata. 2016. Neural semantic role labeling with dependency path embeddings. In *Proceedings of ACL*, pages 1192–1202.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

Lei Sha, Sujian Li, Baobao Chang, Zhifang Sui, and Tingsong Jiang. 2016. Capturing argument relationship for chinese semantic role labeling. In *Proceedings of EMNLP*, pages 2011–2016.

Dan Shen and Mirella Lapata. 2007. Using semantic roles to improve question answering. In *Proceedings of EMNLP-CoNLL*, pages 12–21.

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In *Proceedings of NIPS*, pages 2377–2385.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of EMNLP*, pages 5027–5038.

Weiwei Sun, Zhifang Sui, Meng Wang, and Xin Wang. 2009. Chinese semantic role labeling with shallow parsing. In *Proceedings of EMNLP*, pages 1475–1483.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The conll-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of CoNLL*, pages 159–177.

Reid Swanson and Andrew S Gordon. 2006. A comparison of alternative parse tree paths for labeling semantic roles. In *Proceedings of COLING/ACL*, pages 811–818.

Swabha Swayamdipta, Sam Thomson, Kenton Lee, Luke Zettlemoyer, Chris Dyer, and Noah A Smith. 2018. Syntactic scaffolds for semantic structures. In *Proceedings of EMNLP*, pages 3772–3782.

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2018. Deep semantic role labeling with self-attention. In *Proceedings of AAAI*, pages 4929–4936.

Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. 2015a. Machine comprehension with syntax, frames, and semantics. In *Proceedings of ACL-IJCNLP*, pages 700–706.

Zhen Wang, Tingsong Jiang, Baobao Chang, and Zhifang Sui. 2015b. Chinese semantic role labeling with bidirectional recurrent neural networks. In *Proceedings of EMNLP*, pages 1626–1631.

Shumin Wu and Martha Palmer. 2015. Can selectional preferences help automatic semantic role labeling? In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, pages 222–227.

Qiaolin Xia, Lei Sha, Baobao Chang, and Zhifang Sui. 2017. A progressive learning approach to chinese srl using heterogeneous data. In *Proceedings of ACL*, pages 2069–2077.

Qingrong Xia, Zhenghua Li, Min Zhang, Zhang Meishan, Guohong Fu, Rui Wang, and Luo Si. 2019. Syntax-aware neural semantic role labeling. In *Proceedings of AAAI*.

Nianwen Xue. 2008. Labeling chinese predicates with semantic roles. *Computational linguistics*, 34(2):225–255.

Nan Yu, Meishan Zhang, and Guohong Fu. 2018. Transition-based neural rst parsing with implicit syntax features. In *Proceedings of COLING*, pages 559–570.

Meishan Zhang, Zhenghua Li, Guohong Fu, and Min Zhang. 2019. Syntax-enhanced neural machine translation with syntax-aware word representations. In *Proceedings of NAACL-HLT*, pages 1151–1161.

Meishan Zhang, Yue Zhang, and Guohong Fu. 2016a. Transition-based neural word segmentation. In *Proceedings of ACL*, pages 421–431.

Yu Zhang, Guoguo Chen, Dong Yu, Kaisheng Yaco, Sanjeev Khudanpur, and James Glass. 2016b. Highway long short-term memory rnns for distant speech recognition. In *Proceedings of ICASSP*, pages 5755–5759.

Hai Zhao, Wenliang Chen, and Chunyu Kit. 2009. Semantic dependency parsing of nombank and propbank: An efficient integrated approach via a large-scale feature selection. In *Proceedings of EMNLP*, pages 30–39.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of ACL-IJCNLP*, pages 1127–1137.

# Transfer Fine-Tuning: A BERT Case Study

**Yuki Arase**[1⋆] and **Junichi Tsujii**[⋆2]
[1]Osaka University, Japan
[⋆]Artificial Intelligence Research Center (AIRC), AIST, Japan
[2]NaCTeM, School of Computer Science, University of Manchester, UK
arase@ist.osaka-u.ac.jp, j-tsujii@aist.go.jp

## Abstract

A semantic equivalence assessment is defined as a task that assesses semantic equivalence in a sentence pair by binary judgment (*i.e.*, paraphrase identification) or grading (*i.e.*, semantic textual similarity measurement). It constitutes a set of tasks crucial for research on natural language understanding. Recently, BERT realized a breakthrough in sentence representation learning (Devlin et al., 2019), which is broadly transferable to various NLP tasks. While BERT's performance improves by increasing its model size, the required computational power is an obstacle preventing practical applications from adopting the technology. Herein, we propose to inject phrasal paraphrase relations into BERT in order to generate suitable representations for semantic equivalence assessment instead of increasing the model size. Experiments on standard natural language understanding tasks confirm that our method effectively improves a smaller BERT model while maintaining the model size. The generated model exhibits superior performance compared to a larger BERT model on semantic equivalence assessment tasks. Furthermore, it achieves larger performance gains on tasks with limited training datasets for fine-tuning, which is a property desirable for transfer learning.

## 1 Introduction

Paraphrase identification and semantic textual similarity (STS) measurements aim to assess semantic equivalence in sentence pairs. These tasks are central problems in natural language understanding research and its applications. In this paper, these tasks are defined as semantic equivalence assessments.

Sentence representation learning is the basis of assessing semantic equivalence. Unsupervised learning is becoming the preferred approach because it only requires plain corpora, which are now abundantly available. In this approach, a model is pre-trained to generate generic sentence representations that are broadly transferable to various natural language processing (NLP) tasks. Subsequently, it is fine-tuned to generate specific representations for solving a target task using an annotated corpus. Considering the high costs of annotation, a pre-trained model that efficiently fits the target task with a smaller amount of annotated corpus is desired.

Recently, Bidirectional Encoder Representations from Transformers (BERT) realized a breakthrough, which dramatically improved sentence representation learning (Devlin et al., 2019). BERT pre-trains its encoder using language modeling and by discriminating surrounding sentences in a document from random ones. Pre-training in this manner allows distributional relations between sentences to be learned. Intensive efforts are currently being made to pre-train larger models by feeding them enormous corpora for improvement (Radford et al., 2019; Yang et al., 2019). For example, a large model of BERT has 340M parameters, which is 3.1 times larger than its smaller alternative. Although such a large model achieves performance gains, the required computational power hinders its application to downstream tasks.

Given the importance of natural language understanding research, we focus on sentence representation learning for semantic equivalence assessment. Instead of increasing the model size, we propose the injection of semantic relations into a pre-trained model, namely BERT, to improve performance. Phang et al. (2019) showed that BERT's performance on downstream tasks improves by simply inserting extra training on data-rich supervised tasks. Unlike them, we inject semantic relations of finer granularity using phrasal paraphrase alignments automatically iden-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5393–5404,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

tified by Arase and Tsujii (2017) to improve semantic equivalent assessment tasks. Specifically, our method learns to discriminate phrasal and sentential paraphrases on top of the representations generated by BERT. This approach explicitly introduces the concept of the phrase to BERT and supervises semantic relations between phrases. Due to studies on sentential paraphrase collection (Lan et al., 2017) and generation (Wieting and Gimpel, 2018), a million-scale paraphrase corpus is ready for use. We empirically show that further training of a pre-trained model on relevant tasks transfers well to downstream tasks of the same kind, which we name as *transfer fine-tuning*.

The contributions of our paper are:

- We empirically demonstrate that transfer fine-tuning using paraphrasal relations allows a smaller BERT to generate representations suitable for semantic equivalence assessment. The generated model exhibits superior performance to the larger BERT while maintaining the small model size.
- Our experiments indicate that phrasal paraphrase discrimination contributes to representation learning, which complements simpler sentence-level paraphrase discrimination.
- Our model exhibits a larger performance gain over the BERT model for a limited amount of fine-tuning data, which is an important property of transfer learning.

We hope that this study will open up one of the crucial research directions that will make the approach of pre-trained models more practically useful. Our codes, datasets, and the trained models will be made publicly available at our web site.

## 2 Related Work

Sentence representation learning is an active research area due to its importance in various downstream tasks. Early studies employed supervised learning where a sentence representation is learned in an end-to-end manner using an annotated corpus. Among these, the importance of phrase structures in representation learning has been discussed (Tai et al., 2015; Wu et al., 2018). In this paper, we use structural relations in sentence pairs for sentence representations. Specifically, we employ phrasal paraphrase relations that introduce the notion of a phrase to the model.

The research focus of sentence representation learning has moved toward unsupervised learning in order to exploit the gigantic corpus. Skip-Thought, which was an early learning attempt, learns to generate surrounding sentences given a sentence in a document (Kiros et al., 2015). This can be interpreted as an extension of the distributional hypothesis on sentences. Quick-Thoughts, a successor of Skip-Thought, conducts classification to discriminate surrounding sentences instead of generation (Logeswaran and Lee, 2018). GenSen combines these approaches in massive multi-task learning (Subramanian et al., 2018) based on the premise that learning dependent tasks enriches sentence representations.

Embeddings from Language Models (ELMo) made a significant step forward (Peters et al., 2018). ELMo uses language modeling with bidirectional recurrent neural networks (RNN) to improve word embeddings. ELMo's embedding contributes to the performance of various downstream tasks. OpenAI GPT (Radford et al., 2018) replaced ELMo's bidirectional RNN for language modeling with the Transformer (Vaswani et al., 2017) decoder. More recently, BERT combined the approaches of Quick-Thoughts (*i.e.*, a next-sentence prediction approach) and language modeling on top of the deep bidirectional Transformer. BERT broke the records of the previous state-of-the-art methods in eleven different NLP tasks. While BERT's pre-training generates generic representations that are broadly transferable to various NLP tasks, we aim to fit them for semantic equivalence assessment by injecting paraphrasal relations. Liu et al. (2019) showed that BERT's performance improves when fine-tuning with a multi-task learning setting, which is applicable to our trained model for further improvement.

## 3 Background

### 3.1 Phrase Alignment for Paraphrases

In order to obtain phrasal paraphrases, we used the phrase alignment method proposed in (Arase and Tsujii, 2017) and apply it to our paraphrase corpora. The alignment method aligns phrasal paraphrases on the parse forests of a sentential paraphrase pair as illustrated in Fig. 1.

According to the evaluation results reported in (Arase and Tsujii, 2017), the precision and recall of alignments are 83.6% and 78.9%, which are 89% and 92% of those of humans, respec-

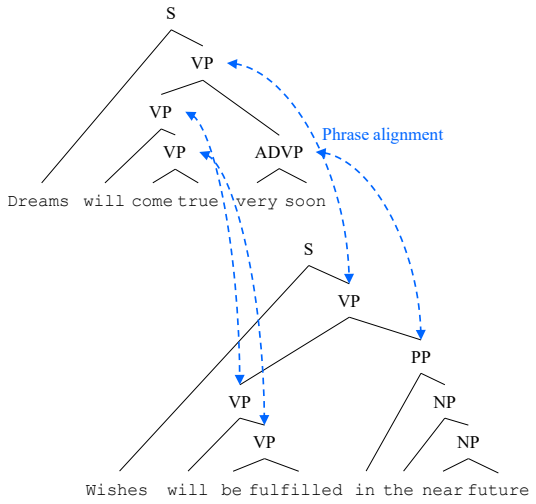


Figure 1: Phrasal paraphrases are obtained from (Arase and Tsujii, 2017); arrows indicate phrase alignments.

tively. Although alignment errors occur, previous studies show that neural networks are relatively robust against noise in a training corpus and still benefit from extra supervisions as demonstrated in (Edunov et al., 2018; Prabhumoye et al., 2018).

We collect all the spans of phrases in a sentential paraphrase pair and their alignments as pairs of phrase spans. Because the phrase alignment method allows unaligned phrases, not all of the phrases have aligned counterparts.

### 3.2 Pre-Training on BERT

BERT is a bidirectional Transformer that generates a sentence representation by conditioning both the left and right contexts of a sentence. A pre-trained BERT model can be easily fine-tuned for a wide range of tasks by just adding a fully-connected layer, without any task-specific architectural modifications. BERT achieved state-of-the-art performances for eleven NLP tasks, thereby outperforming the previous state-of-the-art methods by a large margin.

Pre-training in BERT accomplishes two tasks. The first task is masked language modeling, where some words in a sentence are randomly masked and the model then predicts them from the context. This task design allows the representation to fuse both the left and the right context. The second task predicts whether a pair of sentences are consecutive in a document to learn the relation between the sentences. Specifically, as illustrated in Fig. 2, BERT takes two sentences as input that are concatenated by a special token `[SEP]`.[1] The first token of every input is always the special token of `[CLS]`. The final hidden state corresponding to this `[CLS]` token is regarded as an aggregated representation of the input sentence pair. This is used to predict whether the sentence pair is composed of consecutive sentences in a document or not during pre-training.

BERT has a deep architecture. The BERT-base model has 12 layers of 768 hidden size and 12 self-attention heads. The BERT-large model has 24 layers of 1024 hidden size and 16 self-attention heads. Both BERT-bese and BERT-large models were pre-trained using BookCorpus (Zhu et al., 2015) and English Wikipedia (in total 3.3B words).

**Algorithm 4.1** Paraphrasal Relation Injection

**Input:** Paraphrase sentence pairs $P = \{\langle s, t\rangle\}$ , a pre-trained BERT model

Obtain a set of phrase alignments $A$ as pairs of spans for each $\langle s, t\rangle \in P$
WordPiece tokenization of $P$
Accommodate phrase spans in $A$ to BERT's token indexing: $A = \{\langle (j, k), (m, n)\rangle\}$
**repeat**
  **for all** mini-batch $b_t \in \{\langle P_i, A_i\rangle\}$ **do**
    Encode $b_t$ by the BERT model
    Compute loss: L($\Theta$)
      For phrasal paraphrase task: $L_p(\Theta)$
      For sentential paraphrase task: $L_s(\Theta)$
      $L(\Theta) = L_p(\Theta) + L_s(\Theta)$
    Compute gradient: $\nabla(\Theta)$
    Update the model parameters
**until** convergence

## 4 Transfer Fine-Tuning with Paraphrasal Relation Injection

We inject semantic relations between a sentence pair into a pre-trained BERT model through classification of phrasal and sentential paraphrases. After the training, the model can be fine-tuned in exactly the same manner as with BERT models.

### 4.1 Overview

Algorithm 4.1 provides an overview of our method. It takes a sentential paraphrase pair $\langle s, t\rangle$ as an input, which are referred to as the source and target, respectively, for the sake of clarity. First, a set of phrase alignments $A$ is obtained for $\langle s, t\rangle$

[1] Throughout the paper, `typewriter font` represents tokens and labels.

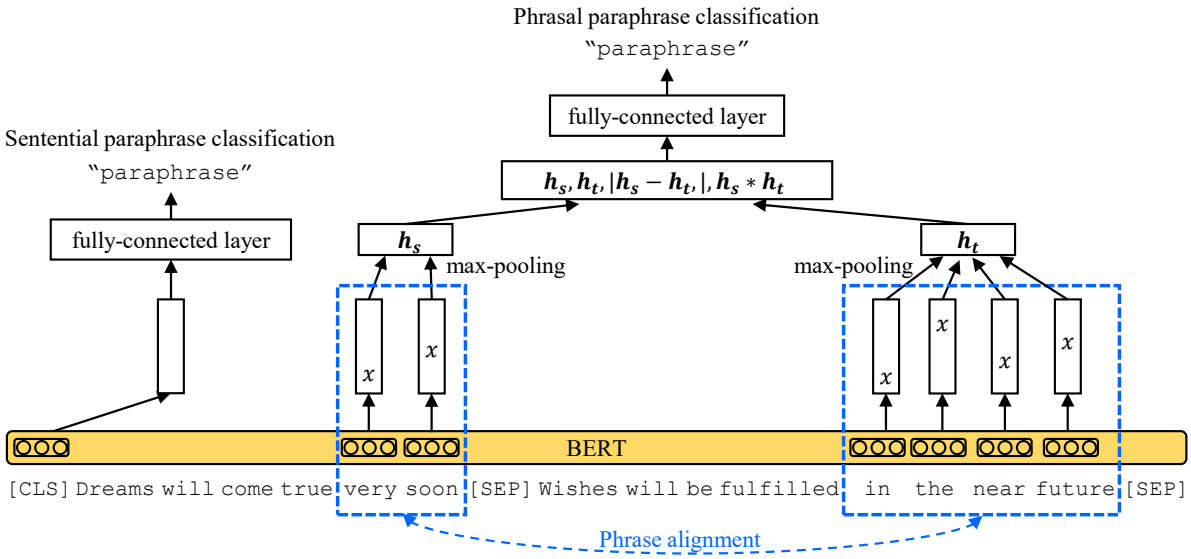


Figure 2: Our method injects semantic relations to sentence representations through paraphrase discrimination.

(line 1) as described in Sec. 3.1. Because BERT uses sub-words as a unit instead of words, all the input sentences are tokenized (line 2) by WordPiece (Wu et al., 2016). In addition, $t$ is concatenated to $s$ when being input to the BERT model, where the first token should always be `[CLS]` and the sentence pair is separated by `[SEP]` as described in Sec. 3.2. In order to accommodate to these factors, phrase spans in alignments $A$ are adjusted accordingly (line 3).

Our method learns to discriminate phrasal and sentential paraphrases simultaneously as illustrated in Fig. 2. Cross-entropy is used as the loss functions for both tasks (line 8, 9).

**Phrasal Paraphrase Classification** The middle part of Fig. 2 illustrates phrasal paraphrase classification. We first generate phrase embedding for each aligned phrase as follows. The tokenized sentence pair is encoded by the BERT model. For the input sequence of $N$ tokens $\{w_i\}_{i=1,\ldots,N}$, we obtain the final hidden states $\{\mathbf{h}_i\}_{i=1,\ldots,N}$ (*i.e.*, output of the bidirectional Transformer):

$$\mathbf{h}_i = \text{Transformer}(w_1, \ldots, w_N),$$

where $\mathbf{h}_i \in \mathbb{R}^\lambda$ and $\lambda$ is the hidden size. We then combine $\{\mathbf{h}_i\}_i$ for a phrase pair with an alignment $\langle (j,k),(m,n)\rangle$ where $2 \leq j < k < m < n \leq N-1$ represent indexes of the beginning and ending of phrases (recall that the first and last tokens are always special tokens in BERT). As a combination function, we apply max-pooling that showed strong performance in (Conneau et al., 2017) to generate a representation of source and target phrases:

$$\mathbf{h}_s = \text{max-pooling}(\mathbf{h}_j, \ldots, \mathbf{h}_k), \quad (1)$$
$$\mathbf{h}_t = \text{max-pooling}(\mathbf{h}_m, \ldots, \mathbf{h}_n). \quad (2)$$

The max-pooling$(\cdot)$ function selects the maximum value over each dimension of the hidden units.

Then $\mathbf{h}_s$ and $\mathbf{h}_t$ are converted to a single vector. To extract relations between $\mathbf{h}_s$ and $\mathbf{h}_t$, three matching methods are used (Conneau et al., 2017): (a) concatenating the representations $(\mathbf{h}_s, \mathbf{h}_t)$, (b) taking the element-wise product $\mathbf{h}_s * \mathbf{h}_t$, and (c) finding the absolute element-wise difference $|\mathbf{h}_s - \mathbf{h}_t|$. The final vector of $\mathbb{R}^{4\lambda}$ is fed into a classifier.[2]

Because our method aims to generate representations for semantic equivalence assessment, the classifier should be simple (Logeswaran and Lee, 2018). Otherwise, a sophisticated classifier would fit itself with the task instead of the representations. We use a single fully-connected layer culminating in a softmax layer as our classifier.

Previous studies have calculated interactions between words (He and Lin, 2016) and phrases (Chen et al., 2017) using the final hidden states of bidirectional RNN or recursive neural networks when composing a sentence representation. Our approach differs from these by giving explicit supervision of which phrase pairs have semantic interactions (*i.e.*, paraphrases).

[2]Our follow-up study confirms that a simpler feature generation improves the generality of our model to contribute not only to semantic equivalent assessment but also natural language inference. For details, please refer to the Appendix.

**Negative Example Selection** In paraphrase identification, non-paraphrases with large lexical differences are easy to discriminate. Discrimination becomes far more difficult when they contain a number of identical or related words. To effectively supervise the model by solving difficult discrimination problems, we designed a three-way classification task: discrimination of `paraphrase`, `random`, and `in-paraphrase` pairs.

The `random` examples are generated by pairing $s$ to a random sentence $t'$ from the training corpus, and then pairing all phrases in $s$ to randomly chosen phrases in $t'$. The `in-paraphrase` examples aim to make the discrimination problem difficult, which requires distinguishing true paraphrases and phrases in the paraphrasal sentence pair $t$. These may provide sub-phrases or ancestor phrases of true paraphrases as difficult negative examples, which tend to retain the same topic and similar wordings. To prepare such examples, for each phrase pair $\langle (j,k),(m,n)\rangle \in A$, the target span $(m,n)$ is replaced by a randomly chosen phrase span in $t$.

Phrasal paraphrase classification aims to give explicit supervision of semantic relations among phrases in representation learning. It also introduces structures in sentences, which is completely missed in BERT's pre-training. Swayamdipta et al. (2018) showed that supervision of phrase-based syntax improves the performance of a task relevant to semantics, *e.g.*, semantic role labeling.

**Sentential Paraphrase Classification** The left side of Fig. 2 illustrates the sentential paraphrase classification. The process is simple; the final hidden state of the `[CLS]` token, *i.e.*, $\mathbf{h}_1$, is fed into a classifier to discriminate whether a sentence pair is a paraphrase or a random sentence combination. Note that these random sentence pairs provide `random` phrases for the phrasal paraphrase classification described above.

### 4.2 Training Setting

We collected paraphrases from various sources as summarized in Table 1, which shows the numbers of sentential and phrasal paraphrase pairs after phrase alignment.[3] All the datasets were downloaded from the Linguistic Data Consortium (LDC) or authors' websites. The following bullets describe the sources.

| Source | Sentence | Phrase |
|---|---|---|
| NIST OpenMT | 47k | 711k |
| Simple Wikipedia | 97k | 1.4M |
| Twitter URL corpus | 50k | 396k |
| Para-NMT | 3.9M | 26.7M |
| Total | 4.1M | 29.2M |

Table 1: Numbers of sentential and phrasal paraphrases after the phrase alignment process.

- NIST OpenMT[4]: We randomly paired reference translations of the same source sentence as was done in (Arase and Tsujii, 2017).
- Twitter URL corpus (Lan et al., 2017): This corpus was collected from Twitter by linking tweets through shared URLs. We used a three-month collection of paraphrases.[5]
- Simple Wikipedia (Kauchak, 2013): This corpus aligned English Wikipedia and Simple English Wikipedia for text simplification. We used "sentence-aligned, version 2.0."[6]
- Para-NMT (Wieting and Gimpel, 2018): This corpus was created by translating the Czech side of a large Czech-English parallel corpus and pairing the translated English and originally target-side English as paraphrases. We used "Para-nmt-5m-processed."[7]

Note that these sentential and phrasal paraphrases are obtained by automatic methods. On the contrary, dataset creation for downstream tasks generally requires expensive human annotation.

We employed the pre-trained BERT-base model[8] and conducted paraphrase classification using the collected paraphrase corpora. Adam (Kingma and Ba, 2015) was applied as an optimizer with a learning rate of $5\mathrm{e}-5$. A dropout probability was $0.2$ for the fully-connected layers in the classifiers. A development set and a test

[3] The numbers of sentential paraphrase pairs were reduced due to parsing and alignment failures.

[4] LDC catalogue number: LDC2010T14, LDC2010T17, LDC2010T21, LDC2010T23, LDC2013T03

[5] https://github.com/lanwuwei/Twitter-URL-Corpus

[6] http://www.cs.pomona.edu/~dkauchak/simplification/data.v2/sentence-aligned.v2.tar.gz

[7] https://drive.google.com/file/d/19NQ87gEFYu3zOIp_VNYQZgmnwRuSIyJd/view?usp=sharing

[8] https://github.com/google-research/bert

set, each with 50k sentence pairs, were subtracted from the paraphrase corpus. The rest of the corpus was used for training. The training was conducted on four NVIDIA Tesla V100 GPUs with a batch-size of 100. Early stopping was applied to stop training at the second time decrease in the accuracy of the phrasal paraphrase classification, which was measured on the development set. The final test-set accuracies were 98.1% and 99.9% for phrasal and sentential paraphrase classification, respectively.

## 5 Evaluation Setting

### 5.1 Hypotheses to Verify

BERT's pre-training learns to generate sentence representations broadly transferable to different NLP tasks. In contrast, our method gives more direct supervision to generate representations suitable for semantic equivalence assessment tasks. We set up the following hypotheses on features of our method, which will be empirically verified through evaluation:

- H1 Our method contributes to semantic equivalence assessment tasks.
- H2 Our method achieves improvement on downstream tasks that only have small amounts of training datasets for fine-tuning.
- H3 Our method moderately improves tasks if they are relevant to semantic equivalence assessment.
- H4 Our training does not transfer to distant downstream tasks that are independent to semantic equivalence assessment.
- H5 Phrasal and sentential paraphrase classification complementarily benefits sentence representation learning.

### 5.2 GLUE Datasets

We empirically verified the hypotheses H1 to H5 using the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2019)[9], which is the standard benchmark and provides collections of datasets for natural language understanding tasks. Table 2 summarizes the tasks and evaluation metrics at GLUE. All the scores reported in this paper are computed at the GLUE evaluation server unless stated otherwise. Accuracies on MRPC and QQP and Spearman correlation on STS-B are omitted due to space limitations. Note that they showed the same trends as F1 and Pearson correlation, respectively, in our experiment. WNLI was excluded because the GLUE web site reports its issues.[10]

| Corpus | Task | Metrics |
|---|---|---|
| MRPC | paraphrase | F1 |
| STS-B | STS | Pearson corr. |
| QQP | paraphrase | F1 |
| MNLI-m | in-domain NLI | accuracy |
| MNLI-mm | cross-domain NLI | accuracy |
| RTE | NLI | accuracy |
| QNLI | QA/NLI | accuracy |
| SST | sentiment | accuracy |
| CoLA | acceptability | Matthews corr. |

Table 2: GLUE tasks and evaluation metrics.

GLUE tasks can be categorized according to their aims as follows.

**Semantic Equivalence Assessment Tasks (MRPC, STS-B, QQP)** These are the primary targets of our method, which are used to verify hypothesis H1. Paraphrase identification assesses semantic equivalence in a sentence pair by binary judgments. Microsoft Paraphrase Corpus (MRPC) (Dolan et al., 2004) consists of sentence pairs drawn from news articles, while Quora Question Pairs (QQP)[11] consists of question pairs from the community QA website.

STS assesses semantic equivalence by grading. STS benchmark (STS-B) (Cer et al., 2017) provides sentence pairs drawn from heterogeneous sources, which are human-annotated with a level of equivalence from 1 to 5.

**NLI Tasks (MNLI-m/mm, RTE, QNLI)** We use natural language inference (NLI) tasks to verify hypothesis H3 because they constitute a class of problems relevant to semantic equivalence assessment. NLI tasks are different from semantic equivalence assessment in that they often require logical inference and understanding of commonsense knowledge. The Multi-Genre Natural Language Inference Corpus (MNLI) (Williams et al., 2018) is a crowd-sourced corpus and covers heterogeneous domains. MNLI-m is an in-domain

[9] https://gluebenchmark.com/

[10] https://gluebenchmark.com/faq

[11] https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs

| Task / Model | Semantic Equivalence | | | NLI | | | Single-Sent. | |
|---|---|---|---|---|---|---|---|---|
| | MRPC | STS-B | QQP | MNLI (m/mm) | RTE | QNLI | SST | CoLA |
| BERT-base | 88.3 | 84.7 | 71.2 | 84.3/83.0 | 59.8 | 89.1 | 93.3 | 52.7 |
| BERT-large | 88.6 | 86.0 | **72.1** | **86.2/85.5** | **65.5** | **92.7** | **94.1** | **55.7** |
| Transfer Fine-Tuning | **89.2** | **87.4** | 71.2 | 83.9/83.1 | 64.8 | 89.3 | 93.1 | 47.2 |

Table 3: GLUE test results scored by the GLUE evaluation server. The best scores are represented in **bold** and scores higher than those of BERT-base are underlined.

NLI task while MNLI-mm is a cross-domain NLI task. The Recognizing Textual Entailment (RTE) corpus[12] was created from news and Wikipedia. Question-answering NLI (QNLI) was created from The Stanford Question Answering Dataset (Rajpurkar et al., 2016) on which all the sentences were drawn from Wikipedia.

**Single-Sentence Tasks (SST, CoLA)** We use these tasks to verify hypothesis H4. They aim to estimate features in a single sentence, which has little interaction with semantic equivalence assessment in a sentence pair. The Stanford Sentiment Treebank (SST) (Socher et al., 2013) task is a binary sentiment classification, while The Corpus of Linguistic Acceptability (CoLA) (Warstadt et al., 2018) task is a binary classification of grammatical acceptability.

### 5.3 Fine-Tuning on Downstream Tasks

Once trained, our model can be used in exactly the same manner as the pre-trained BERT models. For fine-tuning our models and replicating BERT's results under the same setting, we set the hyper-parameter values to those recommended in (Devlin et al., 2019): a batch size of 32, a learning rate of $3e-5$, the number of training epochs to 4, and a dropout probability of 0.1. We fine-tuned all the models on downstream tasks using the script provided in the Pytorch version of BERT.[13] For STS-B, we modified the script slightly to conduct regression instead of classification. All other hyper-parameters were set to the default values defined in the BERT's fine-tuning script.

For fair comparison, we kept the same hyper-parameter settings described above across all tasks and models. Phang et al. (2019) discussed that BERT performances become unstable when a training dataset with fine-tuning is small. In our evaluation, performances were stable when setting the same hyper-parameters, but further investigation is our future work.

## 6 Results and Discussion

### 6.1 Effect on Semantic Equivalence Assessment Tasks

Table 3 shows fine-tuning results on GLUE; our model, denoted as Transfer Fine-Tuning, is compared against BERT-base and BERT-large. The first set of columns shows the results of semantic equivalence assessment tasks. Our model outperformed BERT-base on MRPC (+0.9 points) and STS-B (+2.7 points). Furthermore, it outperformed even BERT-large by 0.6 points on MRPC and by 1.4 points on STS-B, despite BERT-large having 3.1 times more parameters than our model. Devlin et al. (2019) described that the next-sentence prediction task in BERT's pre-training aims to train a model that understands sentence relations. Herein, we argue that such relations are effective at generating representations broadly transferable to various NLP tasks, but are too generic to generate representations for semantic equivalence assessment tasks. Our method allows semantic relations between sentences and phrases that are directly useful for this class of tasks to be learned.

These results support hypothesis H1, indicating that our approach is more effective than blindly enlarging the model size. A smaller model size is desirable for practical applications. We have also applied our method on the BERT-large model, but its performance was not much improved to warrant the larger model size. Further investigation regarding pre-trained model sizes is our future work.

### 6.2 Effect of the Amount of Fine-Tuning Datasets

Our method did not improve upon BERT-base for QQP. We consider this is because a large QQP training set (364k sentence pairs) allows the BERT model to converge to a certain optimum. This also

[12] https://aclweb.org/aclwiki/Recognizing_Textual_Entailment

[13] run_classifier.py in https://github.com/huggingface/pytorch-pretrained-BERT

| Task | Train. size | BERT-base | Transfer Fine-Tuning |
|---|---|---|---|
| MRPC | 1k | 81.6 | 88.1 (+6.5) |
| | all (3.7k) | 89.4 | 90.2 (+0.8) |
| STS-B | 1k | 83.4 | 86.2 (+2.8) |
| | all (5.7k) | 88.1 | 90.1 (+2.0) |
| QQP | 1k | 69.9 | 71.4 (+1.5) |
| | 5k | 75.5 | 76.3 (+0.8) |
| | 10k | 77.0 | 77.6 (+0.6) |
| | 20k | 79.6 | 79.5 (−0.1) |
| | all (364k) | 87.7 | 87.7 (±0.0) |

Table 4: Development set scores of the BERT-base model and our model (and their differences) that were fine-tuned using subsamples and full-size training sets.

relates to hypothesis H2.

To investigate the effect of the sizes of training sets, we fine-tuned our model and BERT-base for semantic equivalence assessment tasks using randomly subsampled training sets. Table 4 shows scores on the development sets.[14] The result clearly indicates that our method is more beneficial when a training dataset is limited on a downstream task, which supports hypothesis H2. This property is preferable for a transfer learning scenario that unsupervised sentence representation learning assumes.

Another factor that may affect the performance is domain mismatch between our paraphrase corpora and QQP corpus. The former was mostly collected from news while the latter was extracted from a social QA forum. In the future, we will investigate the effects of domains by generating multi-domain paraphrase corpora using a method proposed by Wieting and Gimpel (2018).

### 6.3 Effect on NLI Tasks

The second set of columns in Table 3 shows the results on NLI tasks. Our model presents moderate improvements on most NLI tasks, which supports hypothesis H3. We consider this is because the majority of NLI tasks that require inferences in one-direction, contrary to bi-directional entailment relations of paraphrases, are uni-directional.

Another reason is that our elaborate feature generation for the phrasal paraphrase classifier tightly fits the model for paraphrase identification. This contributes to performance improvements on this task, but sacrifices the model's generality on relevant tasks. We tackle this issue in our follow-up study reported in the Appendix.

Among NLI tasks, our model largely outperformed BERT-base by 5.0 point on RTE. This may be again due to the property of our method that brings improvement on tasks with a limited training set as RTE has only 2.5k training sentence pairs.

### 6.4 Effect on Single-Sentence Tasks

The last two columns of Table 3 show results on single-sentence tasks; SST and CoLA, which are the most distant tasks from paraphrase classification. Our model presents a slightly lower score on SST compared to BERT-base and performed poorly on CoLA.

One potential reason for this degradation is that our training takes a sentence pair as input, which may weaken the ability to model a single sentence. Another cause is attributable to similarities between our training and fine-tuning tasks. For SST, sentiment analysis could be adversarial toward paraphrase discrimination tasks. Although paraphrasal sentences tend to have the same sentiments, sentences with the same sentiments do not generally hold paraphrastic relations. For CoLA, semantic relations unlikely contribute to determining grammatical acceptability, as required by CoLA task.

Together with the results in Sec. 6.3, hypothesis H4 is supported; the effectiveness of our method depends on relevance between paraphrase discrimination and downstream tasks. Our future work will be to examine what characteristics of NLP tasks make our method less effective.

### 6.5 Ablation Study

To verify hypothesis H5, we conducted an ablation study that investigates independent effects of sentential and phrasal paraphrase classification. Table 5 shows the results; the last three rows show performances when conducting only sentential paraphrase classification, phrasal paraphrase classification, and binary classification of `paraphrase` and `in-paraphrase` pairs, respectively. All the models were fine-tuned in the same manner as described in Sec. 5.3.

First, the results support the hypothesis; sentential and phrasal paraphrase classification complements each other on sentence representation learning. Our model achieved its best scores

[14] We used the development set because the GLUE server allows only two submissions per day. Note that the number of training epochs for fine-tuning is fixed in our experiments, hence, the development set was not used for other purposes.

| Task / Model | Semantic Equivalence | | | NLI | | | Single-Sent. | |
|---|---|---|---|---|---|---|---|---|
| | MRPC | STS-B | QQP | MNLI (m/mm) | RTE | QNLI | SST | CoLA |
| Transfer Fine-Tuning | **89.2** | 87.4 | **71.2** | 83.9/**83.1** | 64.8 | 89.3 | 93.1 | 47.2 |
| BERT-base | 88.3 | 84.7 | **71.2** | **84.3**/83.0 | 59.8 | 89.1 | **93.3** | **52.7** |
| +sentence | 88.2 | **87.6** | 71.1 | 83.2/82.8 | **66.2** | **90.2** | 92.4 | 39.8 |
| +3way-PP | 88.2 | 85.8 | 70.9 | 82.9/81.9 | 65.8 | 88.0 | 91.3 | 32.6 |
| +binary-PP | 87.7 | 82.8 | 70.7 | 83.7/82.2 | 61.2 | 87.6 | 92.5 | 42.1 |

Table 5: Results of the ablation study where the best scores are represented in **bold** and scores higher than those of BERT-base are underlined. The last three rows show performances when conducting only sentential paraphrase classification (+sentence), phrasal paraphrase classification (+3way-PP), and binary classification of phrasal paraphrase (+binary-PP), respectively.

on MRPC, MNLI-m/mm, SST, and CoLA tasks by conducting both sentential and phrasal paraphrase classification simultaneously. Interestingly, these scores are higher than those when sentential and phrasal paraphrase classification are conducted independently. This is reasonable considering the process of fine-tuning. Sentential paraphrase classification directly affects the representation of `[CLS]`, which is the primary tuning factor in fine-tuning for downstream tasks. Alternatively, phrasal paraphrase classification affects representations of phrases, which are the basis for generating the `[CLS]` representation. Simultaneously conducting both sentential and phrasal paraphrase classification thus creates synergy.

It is also obvious that the three-way classification of phrasal paraphrases, on which the model discriminates paraphrases, random combinations of phrases from a random pair of sentences, and random combinations of phrases in a paraphrasal sentence pair, is superior to binary classification. This shows that discriminating random combinations of phrases, which is a simpler and easier task, also contributes to representation learning.

## 7 Conclusion

We empirically demonstrate that sentential and phrasal paraphrase relations help sentence representation learning. While BERT's pre-training aims to generate generic representations transferable to a broad range of NLP tasks, our method generates representations suitable for the class of semantic equivalence assessment tasks. Our method achieves performance gains while maintaining the model size. Furthermore, it exhibits improvement on downstream tasks with limited amounts of training datasets for fine-tuning, which is a property crucial for transfer learning.

In the future, we plan to investigate the effects of our method on different sizes of BERT models. Additionally, we will apply our model to improve the alignment quality of the phrase alignment model.

## Acknowledgments

We appreciate the anonymous reviewers for their insightful comments and suggestions to improve the paper. This work was supported by JST, ACT-I, Grant Number JPMJPR16U2, Japan.

## References

Yuki Arase and Jun'ichi Tsujii. 2017. Monolingual phrase alignment on parse forests. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1–11, Copenhagen, Denmark.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the International Workshop on Semantic Evaluation (SemEval)*, pages 1–14, Vancouver, Canada.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced lstm for natural language inference. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1657–1668, Vancouver, Canada.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 670–680, Copenhagen, Denmark.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of

deep bidirectional transformers for language understanding. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 4171–4186.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 350–356, Geneva, Switzerland.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 489–500, Brussels, Belgium.

Hua He and Jimmy Lin. 2016. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. pages 937–948, San Diego, California.

David Kauchak. 2013. Improving text simplification language modeling using unsimplified text data. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1537–1546, Sofia, Bulgaria.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Proceedings of Conference on Neural Information Processing Systems (NeurIPS)*, pages 3294–3302.

Wuwei Lan, Siyu Qiu, Hua He, and Wei Xu. 2017. A continuously growing dataset of sentential paraphrases. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1224–1234, Copenhagen, Denmark.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. *arXiv*, 1901.11504.

Lajanugen Logeswaran and Honglak Lee. 2018. An efficient framework for learning sentence representations. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 2227–2237, New Orleans, Louisiana.

Jason Phang, Thibault Févry, and Samuel R. Bowman. 2019. Sentence encoders on STILTs: Supplementary training on intermediate labeled-data tasks. *arXiv*, 1811.01088.

Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W Black. 2018. Style transfer through back-translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 866–876, Melbourne, Australia.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. Technical report, OpenAI.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. Technical report, OpenAI.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2383–2392, Austin, Texas.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1631–1642, Seattle, Washington, USA.

Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J Pal. 2018. Learning general purpose distributed sentence representations via large scale multi-task learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Swabha Swayamdipta, Sam Thomson, Kenton Lee, Luke Zettlemoyer, Chris Dyer, and Noah A. Smith. 2018. Syntactic scaffolds for semantic structures. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3772–3782.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *ACL-IJCLNLP*, pages 1556–1566, Beijing, China.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of Conference on Neural Information Processing Systems (NeurIPS)*, pages 5998–6008.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2018. Neural network acceptability judgments. *arXiv*.

John Wieting and Kevin Gimpel. 2018. ParaNMT-50M: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 451–462, Melbourne, Australia.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. pages 1112–1122, New Orleans, Louisiana.

Wei Wu, Houfeng Wang, Tianyu Liu, and Shuming Ma. 2018. Phrase-level self-attention networks for universal sentence encoding. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3729–3738, Brussels, Belgium.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv*, 1810.04805.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. *arXiv*, 1906.08237.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of International Conference on Computer Vision (ICCV)*, pages 19–27.

## Appendix: Transfer Fine-Tuning with Simple Features

To further investigate effects of transfer fine-tuning using paraphrase relations on BERT, we designed a model that generates a simplest feature to input into the classifier in Fig. 2. We assume that this method transmits learning signals to the underlying BERT in a more effective manner. Specifically, we use mean-pooling to generate representations of source and target phrases in Eq. (1) and Eq. (2), respectively. These representations are simply concatenated as a feature representation and then fed into the classifier.

Table 6 compares this new model (denoted as Simple Transfer Fine-Tuning) to BERT models as well as our model with the elaborate feature generation described in Sec. 4 (denoted as Transfer Fine-Tuning) on semantic equivalent assessment and NLI tasks of GLUE benchmark. Table 7 reports an ablation study. The results and findings are summarized as follows.

- Our model with simple feature generation (Simple Transfer Fine-Tuning) on BERT-base outperformed BERT on both semantic equivalent assessment and NLI tasks. Furthermore, it performed on-par against BERT-large on MRPC and outperformed it on STS-B and RTE, despite BERT-large having 3.1 times more parameters than our model.

- The same trend was confirmed on the model trained on BERT-large, where our model outperformed BERT-large on all the tasks except QNLI.

- Simple Transfer Fine-Tuning also outperformed our model with elaborate feature generation (Transfer Fine-Tuning) on all semantic equivalent assessment and NLI tasks except MRPC. This result implies that elaborate feature generation tightly fits the model to paraphrase identification while sacrifices its generality to relevant tasks. Further investigation will be our future work.

- Sentential and phrasal paraphrase classification complements each other on sentence representation learning when using simple feature generation, as also confirmed when using the elaborate feature generation in Table 5. Simple Transfer Fine-Tuning achieved higher scores on STS-B, RTE, and QNLI tasks than models trained either with only sentential (+sentence) or phrasal paraphrase (+3way-PP [Simple Feature]) classification.

- Simple feature generation improves the performance of the model trained with only phrasal paraphrase classification; +3way-PP

| Task / Model | Semantic Equivalence | | | NLI | | |
|---|---|---|---|---|---|---|
| | MRPC | STS-B | QQP | MNLI (m/mm) | RTE | QNLI |
| BERT-base | 88.3 | 84.7 | 71.2 | 84.3/83.0 | 59.8 | 89.1 |
| Transfer Fine-tuning | 89.2 | 87.4 | 71.2 | 83.9/83.1 | 64.8 | 89.3 |
| Simple Transfer Fine-Tuning | 88.6 | **87.7** | 71.5 | 84.7/83.6 | 67.0 | 91.1 |
| BERT-large | 88.6 | 86.0 | 72.1 | 86.2/85.5 | 65.5 | **92.7** |
| Simple Transfer Fine-Tuning | **89.9** | 87.1 | **72.5** | **86.5/85.6** | **68.2** | 92.2 |

Table 6: Test results on semantic equivalence assessment and NLI tasks scored by the GLUE evaluation server. The best scores for each task are represented in **bold**. The scores higher than those of BERT counterparts (against BERT-base and BERT-large, respectively) are underlined. Our models with simple feature generation (Simple Transfer Fine-Tuning) consistently outperformed the BERT models and achieved the best scores for six out of seven tasks.

| Task / Model | Semantic Equivalence | | | NLI | | |
|---|---|---|---|---|---|---|
| | MRPC | STS-B | QQP | MNLI (m/mm) | RTE | QNLI |
| Simple Transfer Fine-Tuning | 88.6 | **87.7** | **71.5** | **84.7/83.6** | **67.0** | **91.1** |
| BERT-base | 88.3 | 84.7 | 71.2 | 84.3/83.0 | 59.8 | 89.1 |
| +sentence | 88.2 | 87.6 | 71.1 | 83.2/82.8 | 66.2 | 90.2 |
| +3way-PP [Elaborate Feature] | 88.2 | 85.8 | 70.9 | 82.9/81.9 | 65.8 | 88.0 |
| +3way-PP [Simple Feature] | **89.0** | 86.6 | **71.5** | **84.7/83.6** | 65.6 | 90.6 |

Table 7: Results of the ablation study where the best scores are represented in **bold** and scores higher than those of BERT-base are underlined. The third row shows performances when conducting only sentential paraphrase classification (+sentence) and the fourth row shows those when conducting only phrasal paraphrase classification with elaborate feature generation (+3way-PP [Elaborate Feature]), as reported in Table 5. The last row shows performances when conducing phrasal paraphrase classification with simple feature generation (+3way-PP [Simple Feature]). Results indicate that sentential and phrasal paraphrase classification complementarily contributes to Simple Transfer Fine-Tuning modeling.

[Simple Feature] outperformed +3way-PP [Elaborate Feature] on all tasks except RTE.

# Data-Anonymous Encoding for Text-to-SQL Generation

**Zhen Dong**[1]*, **Shizhao Sun**[2], **Hongzhi Liu**[1], **Jian-Guang Lou**[2] and **Dongmei Zhang**[2]
[1]Peking University [2]Microsoft Research
{zhendong,liuhz}@pku.edu.cn
{shizsu,jlou,dongmeiz}@microsoft.com

## Abstract

On text-to-SQL generation, the input utterance usually contains lots of tokens that are related to column names or cells in the table, called *table-related tokens*. These table-related tokens are troublesome for the downstream neural semantic parser because it brings complex semantics and hinders the sharing across the training examples. However, existing approaches either ignore handling these tokens before the semantic parser or simply use deterministic approaches based on string-match or word embedding similarity. In this work, we propose a more efficient approach to handle table-related tokens before the semantic parser. First, we formulate it as a sequential tagging problem and propose a two-stage anonymization model to learn the semantic relationship between tables and input utterances. Then, we leverage the implicit supervision from SQL queries by policy gradient to guide the training. Experiments demonstrate that our approach consistently improves performances of different neural semantic parsers and significantly outperforms deterministic approaches.

## 1 Introduction

Interacting with relational databases or tables through natural language is an important and challenging problem (Androutsopoulos et al., 1995; Li and Jagadish, 2014; Pasupat and Liang, 2015; Zhong et al., 2017; Yu et al., 2018c). To tackle this problem, the neural semantic parser has been widely studied (Dong and Lapata, 2016; Jia and Liang, 2016; Herzig and Berant, 2017; Dong and Lapata, 2018), which automatically maps the input natural language into the logical form (e.g., SQL query) following the typical encoder-decoder structure (Hochreiter and Schmidhuber, 1997; Bahdanau et al., 2015; Vinyals et al., 2015).

Semantic parsing usually tackles two kinds of problems (Goldman et al., 2018; Herzig and Berant, 2018), i.e., the *lexical problem* and the *structural problem*. On text-to-SQL generation, the lexical problem refers to mapping tokens in input utterances to constants in SQL queries, e.g., the column names or cells in SQL queries. The structural problem refers to mapping intentions conveyed by input utterances to operators in SQL queries, e.g., the aggregators or the existence of WHERE clause in SQL queries. Intuitively, the lexical problem can be formulated as a sequential tagging problem, called *anonymization*, where each token in the input utterance will be tagged as being related to a column name, a cell or nothing. For ease of reference, we call the tokens in input utterances related to column names or cells *table-related tokens*, and the tagged utterance *anonymous utterance*.

If the lexical problem can be reduced before the neural semantic parsing, the training difficulties will be greatly alleviated. The reason is two-fold. First, by anonymizing table-related tokens in the input utterance before the neural semantic parser, we can conceal the complex semantics of table-related tokens. Second, different input utterances, which have different table-related tokens but the same structure, can be reduced to the same anonymous utterance before they are fed into the parser. This will result in sharing across training data and thus alleviate training difficulties (Goldman et al., 2018; Herzig and Berant, 2018). For example, in Figure 1a and 1b, input utterances (denoted as $x$) seemingly ask different questions about country and college team, but they can be reduced to a similar anonymous utterance (denoted as $\tilde{x}$) by replacing the token related to the column name as $c$ and the token related to the cell as $v$.

However, reducing the lexical problem before the neural semantic parser is far from being well-studied on text-to-SQL generation although it

*This work was done when the author was visiting Microsoft Research Asia.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5405–5414,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics




(a) Example A (b) Example B (c) Example C

Figure 1: Examples for anonymizing table-related tokens on WikiSQL. For each subfigure, the upper box shows the utterance and the SQL query, while the lower box shows the table. In the upper box, $x$ denotes the input utterance, $\tilde{x}$ denotes the anonymous utterance and $y$ denotes the SQL query. In anonymous utterance $\tilde{x}$, we use $c$ and $v$ to replace the tokens that are related to column names and cells respectively. In addition, we use the color to indicate the correspondence of column names or cells among $x$, $\tilde{x}$ and the table.

has been demonstrated to be helpful on other tasks (Goldman et al., 2018; Herzig and Berant, 2018). First, most approaches ignores reducing the lexical problem (Zhong et al., 2017; Xu et al., 2017; Dong and Lapata, 2018; Shi et al., 2018; Wang et al., 2018a; Hwang et al., 2019). Without conducting anonymization before the neural semantic parser, these approaches cannot receive the aforementioned benefit, although they may have modules to predict the presence of column names and cells inside the parser. Second, a few studies have tried the anonymization by deterministic approaches, which compare the similarity based on string-match or word embedding between the input utterance and the table (Wang et al., 2018b; Yu et al., 2018a). These approaches cannot fully understand the semantic relationship between input utterances and tables, and ignore the relationship with components of SQL queries. For example, in Figure 1c, although the token 'player' is similar to the column name 'Player', it should not be tagged as a table-related token because the token "player" is just a non-sense demonstrative pronoun and the column 'Player' is not related to the SQL query.

To this end, we propose to use the learning-based approach to reduce the lexical problem, i.e., anonymize the table-related tokens. First, we propose a two-stage anonymization model to learn the semantic relationship between input utterances and tables. Then, considering that there is no labeled data for the anonymization, we propose to extract the set of column names and cells appearing in the SQL query and use this set as the supervision. Another benefit of leveraging such implicit supervision is that we can make the anonymization model consider the relationship with the components of the SQL query when it learns the semantic relationship, and thus avoid suffering the same disadvantage as deterministic approaches. Furthermore, to bridge the gap that our model is a sequential tagging model while the supervision extracted from the SQL query is an unordered set, we leverage the policy gradient (Williams, 1992) to train the model. Moreover, we train the anonymization model and the neural semantic parser as whole by a varational inference-like framework with the anonymous utterance as the hidden variable.

Experimental results demonstrate that our approach can consistently improve performances of different neural semantic parsers and significantly outperform typical deterministic approach on the anonymization problem.

## 2 Related Work

**Semantic Parsing.** Semantic parsing aims to map natural languages into executable programs. In the area of semantic parsing, the programs could be in various types, e.g., $\lambda$-calculus (Zettlemoyer and Collins, 2005), Python (Oda et al., 2015), SQL (Zhong et al., 2017), etc; the source of the knowledge can also be different, e.g., the knowledge base, the table, the image (Suhr et al., 2017), etc; and the supervision can take different forms as well, e.g., question-denotation pairs (Pasupat and Liang, 2015), question-program pairs,

etc. In this work, we focus on text-to-SQL generation with the table as the source of the knowledge and with question-SQL query pairs as supervision.

**Reducing Lexical Problem.** On tasks other than text-to-SQL generation, some studies have tried to reduce lexical problem by anonymizing tokens that are related to the program constants. Goldman et al. (2018) lifts tokens in utterances to an abstract form by referring to fixed mappings on visual reasoning task. Herzig and Berant (2018) employs a rule-based method to transform content words in utterances to an abstract representation. Although these studies inspire us to consider reducing lexical problem on text-to-SQL generation, the rules they employed cannot be directly applied to text-to-SQL generation.

On text-to-SQL generation, a few studies tried to anonymize table-related tokens although they do not aim at reducing lexical problem. To better understand rare entities, Yu et al. (2018a) used string match to recognize column names. To learn a domain-transferable semantic parser, Wang et al. (2018b) detected column names by measuring the closeness of edit distance and word embedding. As discussed in Section 1, these deterministic approaches can not fully understand the semantic relationship between input utterances and tables, and ignore the relationship with components of SQL queries.

**Entity Linking.** Our approach can be regarded as an implementation of entity linking on the concrete task, because general entity linking (Shen et al., 2015; Kolitsas et al., 2018) approaches fail to handle particular challenges in our scenario. On the one hand, there are lots of cases cannot be handled by the deterministic entity linking method which only relies on measuring the similarity; on the other hand, there is no labeled data for the learning-based entity linking method.

## 3 Problem Formulation

Denote the input utterance as $x = x_1 \dots x_{|x|}$ and the corresponding SQL query as $y = y_1 \dots y_{|y|}$. We formulate the lexical problem as a sequential tagging problem. Formally, for each token $x_t$ in the input utterance, we give it a tag $\tilde{x}_t \in \{\text{COL}_1, \dots, \text{COL}_K, \text{CELL}, \text{UNK}\}$, where $\text{COL}_k$ represents that $x_t$ is related to the $k$-th column name in the table, CELL represents that $x_t$ is related to a cell in the table and UNK represents that

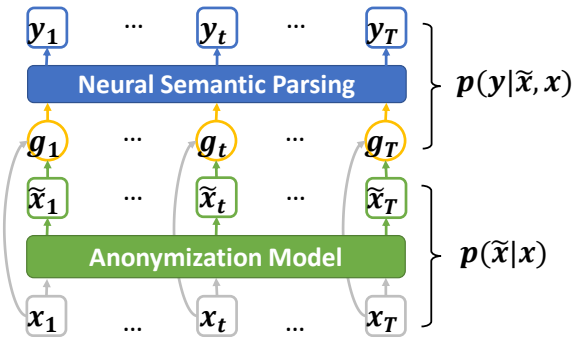


Figure 2: Framework.

$x_t$ is not related to the table. Here $K$ is the number of column names in the table. Note that indexes of column names cannot be ignored in the algorithm to better leverage the implicit supervision from SQL queries (see Section 4.2) although it is ignored when giving examples in Figure 1 and Table 4 for ease of read. For ease of reference, we call this sequential tagging problem *anonymization* and the tagged sequence $\tilde{x} = \tilde{x}_1 \dots \tilde{x}_{|x|}$ *anonymous utterance*.

The typical neural semantic parser aims to estimate $p(y|x)$. In our work, we decompose $p(y|x)$ into two processes (as shown in Figure 2), i.e.,

$$p(y|x) = \sum_{\tilde{x}} p(y|x, \tilde{x})\, p(\tilde{x}|x)\,. \tag{1}$$

Specifically, one process is to learn the anonymous utterance $\tilde{x}$ given the input utterance $x$ for the purpose of reducing lexical problem, i.e.,

$$p(\tilde{x}|x) = \prod_t p(\tilde{x}_t|x)\,, \tag{2}$$

while the other process is to learn the neural semantic parser with anonymous utterance $\tilde{x}$ as the additional input, i.e.,

$$p(y|x, \tilde{x}) = \prod_t p(y_t|y_{<t}, x, \tilde{x})\,. \tag{3}$$

In the following, we will discuss how to learn the anonymous utterance (Eqn (2)) and how to learn the neural semantic parser with anonymous utterance as additional input (Eqn (1)) in detail.

## 4 Approach

### 4.1 Anonymization Model

The anonymization is a structural problem by nature. First, we need to differentiate whether the token is related column names, cells or not related to

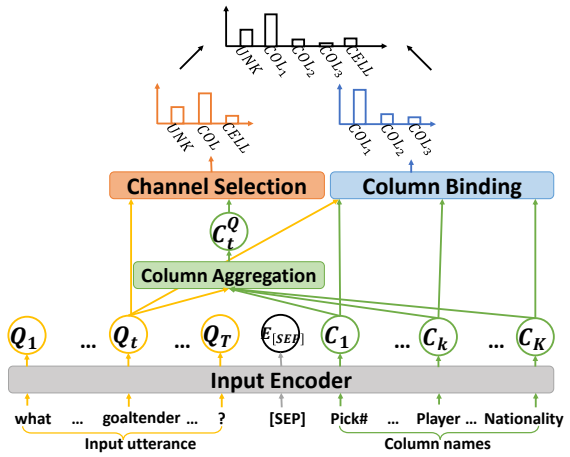


Figure 3: Illustration of the anonymization model when the table content is not available. We take the $t$-th token in the input utterance as the example.

the table. Second, we need to further recognize the concrete column/cell that the token is related to. Therefore, we design a model that uses two stages, i.e., *channel selection* stage and the *column/cell binding* stage, to tackle these two subproblems respectively. The probability distributions produced by these two stages together determine the result,

$$p\left(\tilde{x}_t|x\right) = \sum_{a_t} p_{\text{channel}}\left(a_t|x\right) p_{\text{binding}}\left(\tilde{x}_t|a_t, x\right), \tag{4}$$

where $a_t \in \{\text{COL}, \text{CELL}, \text{UNK}\}$ is the selected channel indicating that the token is related to column names, cells or nothing respectively, $p_{\text{channel}}\left(\cdot\right)$ is the probability produced by the channel selection stage, and $p_{\text{binding}}\left(\cdot\right)$ is the probability produced by the column/cell binding stage.

For ease of reference, we call the two-stage model *anonymization model*, and illustrate it in Figure 3. In the following, we introduce details of its different components.

**Input Encoder.** We leverage BiLSTM or BERT (Devlin et al., 2018) to encode both the input utterance and the table. Due to the privacy problem, the table content is not allowed to be used on most text-to-SQL tasks (Zhong et al., 2017; Yu et al., 2018c). Therefore, we use the concatenation of the embedding of the tokens in the input utterance, the embedding of the separator and the embedding of the column names in the table as the input for BiLSTM or BERT. We denote the output of input encoder as $\{Q_1, \ldots, Q_T, E_{[\text{SEP}]}, C_1, \ldots, C_K\}$, where $Q_t, t \in \{1, \ldots, T\}$ is the vector for the encoding of the $t$-th token in the input utterance, $E_{[\text{SEP}]}$ is the vector for the encoding of the separator, and $C_k, k \in \{1, \ldots, K\}$ is the vector for the encoding of the $k$-th column name in the table. Note that we run a BiLSTM or BERT between column names instead of over each column name. The reason is that Yu et al. (2018a) have shown that although the order of column names does not matter, running a BiLSTM or BERT between colomn names can capture relationships between them, which will benefit the accuracy and the training time.

**Channel Selection.** We implement a linear gate with encodings of the input utterance and aggregated encodings of column names as input:

$$p_{\text{channel}}\left(a_t|x\right) = \text{softmax}\left(W_{\text{gate}}\left[Q_t; C_t^Q\right]\right), \tag{5}$$

where $W_{\text{gate}}$ stands for learnable parameters, $C_t^Q$ is the aggregated encoding of column names, and $[Q_t; C_t^Q]$ is the concatenation of $Q_t$ and $C_t^Q$.

Specifically, to obtain the most relevant column names for the $t$-th token in the input utterance, we leverage attention mechanism (Bahdanau et al., 2015) towards column names $\{C_1, \ldots, C_K\}$ to compute the aggregated encoding of the column names $C_t^Q$, i.e., $C_t^Q = \sum_{k=1}^{K} \alpha_t^k C_k$, where $\alpha_t^k$ is obtained through $u_t^k = v_1 \tanh\left(W_1 Q_t + W_2 C_k\right)$ and $\alpha_t^k = \text{softmax}\left(u_t^k\right)$, $k \in \{1, \ldots, K\}$. Here, $v_1$, $W_1$ and $W_2$ are learnable parameters.

**Column/Cell Binding**[1]**.** The probability distribution generated by this stage, i.e., $p_{\text{binding}}(\cdot)$, can be categorized into three types, i.e., $p_{\text{binding}}^{\text{COL}}(\cdot)$, $p_{\text{binding}}^{\text{CELL}}(\cdot)$ and $p_{\text{binding}}^{\text{UNK}}(\cdot)$, corresponding to the three channels in the first stage respectively. Obviously, $p_{\text{binding}}^{\text{UNK}}\left(\tilde{x}_t|a_t, x\right) = 1$.

For column names, we produce a probability distribution over the $K$ columns in the table through measuring the relevance between the $t$-th token and the $k$-th column name:

$$p_{\text{binding}}^{\text{COL}_k}\left(\tilde{x}_t|a_t, x\right) = \text{softmax}\left(z_t^k\right), \tag{6}$$
$$z_t^k = v_2 \tanh\left(W_3 Q_t + W_4 C_k\right), \; k \in \{1, \ldots, K\},$$

where $v_2$, $W_3$ and $W_4$ are learnable parameters.

[1]For the situation that the input utterance contains table names or the table content is allowed to be used, we can handle table names or cells by the similar way of column names.

For cells, considering that table content is not available due to the privacy problem, we simply predict a substring from the input utterance. Specifically, we follow the longest match principle to merge consecutive tokens in the input utterance, which are labeled as CELL by the channel selection stage, to one cell. Therefore, the generated by column/cell binding stage is simply set as 1, i.e., $p^{\text{CELL}}_{\text{binding}}(\tilde{x}_t|a_t, x) = 1$.

### 4.2 Implicit Supervision from SQL Queries

To train the anonymization model, the ground truth for the anonymous utterance is indispensable. However, there is no such labeled data, and manually labeling the whole training data for each text-to-SQL task is unrealistic, especially when the amount of training data is tremendous.

To tackle this problem, we propose to extract the set of column names and cells appearing in the SQL query and use this set as the supervision to guide the training of the anonymization model. Another benefit of such approach is that we can make our model consider the relationship with components of the SQL query, and thus avoid suffering the same trouble as deterministic approaches. Concretely, for each SQL query $y$, we denote the set of column names and cells appearing in it as $S_{\text{SQL}}$, which consists of three parts, i.e., $S_{\text{SQL}} = S_{\text{sel\_col}} \cup S_{\text{other\_col}} \cup S_{\text{cell}}$:

1. $S_{\text{sel\_col}}$ is the set of column names appearing in the SELECT clause;
2. $S_{\text{other\_col}}$ is the set of column names appearing in other clauses;
3. $S_{\text{cell}}$ is the set of cells appearing in the whole SQL query.

### 4.3 Policy Gradient for Sequential Tagging

**Maximizing Expected Reward.** Ideally, if we have the ground truth, we can train the anonymization model by minimizing the gap (e.g., KL divergence) between the predicted probabilities and the ground truth. Unfortunately, it is infeasible because the implicit supervision from SQL queries (i.e., $S_{\text{SQL}}$) takes a form of the unordered set while the anonymization model faces a sequential tagging task. To address this problem, we propose to encourage the set of column names and cells appearing in the predicted anonymous utterance, denoted as $S_{\text{pred}}$, to be similar to that extracted from the SQL query, i.e., $S_{\text{SQL}}$. To this end, we define a reward of the predicted anonymous utterance $r(\tilde{x})$ as the similarity between $S_{\text{pred}}$ and $S_{\text{SQL}}$, and then train the anonymization model by maximizing expected reward,

$$J_{\text{ER}}(\theta) = \sum_{(x,\tilde{x})\in\mathcal{D}} \mathbb{E}_{\tilde{x}\sim p_\theta(\cdot|x)} r(\tilde{x}), \tag{7}$$

where $\mathcal{D}$ represents the set of training pairs.

However, directly computing expected reward requires traversing all the possible anonymous utterance $\tilde{x}$, which is unrealistic. Therefore, we leverage Monte Carlo estimate as the approximation of the expected reward. Concretely, we sample $N$ anonymous utterances following probability distribution generated by the anonymization model, and then average the reward of the samples to estimate the expected reward,

$$J_{\text{ER}}(\theta) \approx \sum_{(x,\tilde{x})\in\mathcal{D}} \sum_{j=1}^{N} \frac{1}{N} r(\tilde{x}^j), \tag{8}$$

where $\tilde{x}^j$ denotes the $j$-th sample and $\tilde{x}^j \sim p_\theta(\cdot|x)$.

To maximize the above objective function by gradient descent, we employ REINFORCE (Williams, 1992) method, i.e,

$$\nabla_\theta J_{\text{ER}}(\theta) \approx \sum_{(x,\tilde{x})\in\mathcal{D}} \sum_{j=1}^{N} \frac{1}{N} r(\tilde{x}^j) \nabla_\theta \log p_\theta\left(\tilde{x}^j|x\right). \tag{9}$$

**Measurement of Reward.** For ease of reference, we decompose $S_{\text{pred}}$ into two parts, i.e., $S_{\text{pred}} = S_{\text{pred\_col}} \cup S_{\text{pred\_cell}}$, where $S_{\text{pred\_col}}$ and $S_{\text{pred\_cell}}$ represent the set of column names and the set of cells appearing in the predicted anonymous utterance respectively.

The measurement of the reward $r(\tilde{x})$ is designed based on the similarity between $S_{\text{pred}}$ and $S_{\text{SQL}}$ by referring to the following principles:

1. The predicted anonymous utterance should contain the column names in the SELECT clause of the SQL query, i.e., $S_{\text{sel\_col}} \subset S_{\text{pred\_col}}$. The anonymization model will be punished when it misses the column names in $S_{\text{sel\_col}}$. This is motivated by our preliminary analysis that almost every column name in the SELECT clause has the corresponding token in the input utterance.

2. The column names in the predicted anonymous utterance should be at least a subset of the column names appearing in the SQL query, i.e., $S_{\text{pred\_col}} \subseteq S_{\text{sel\_col}} \cup S_{\text{other\_col}}$. Unlike $S_{\text{sel\_col}}$, the column names appearing in other clauses of the

SQL query may possible not have corresponding tokens in the input utterance. For example, in Figure 1a, the column name 'Position' does not have corresponding tokens in the input utterance. Therefore, it is unreasonable to strictly force column names in the predicted anonymous utterance to be the same as that appearing in the SQL query. Instead, it is better to punish the anonymization model when it predicts column names outside the set of column names appearing in the SQL query.

3. The cell names in the predicted anonymous utterance should be the same as that appearing in the SQL query, i.e., $S_{\text{pred\_cell}} = S_{\text{cell}}$. If there is any missing or extra cell in $S_{\text{pred\_cell}}$, the anonymization model will be punished.

According to above principles, we design the reward $r(\tilde{x})$ as,

1. if $S_{\text{sel\_col}} \subseteq S_{\text{pred\_col}}$, $S_{\text{pred\_cell}} = S_{\text{cell}}$ and $S_{\text{pred\_col}} \subseteq S_{\text{sel\_col}} \cup S_{\text{other\_col}}$ are all true, $r(\tilde{x}) = 1$;
2. if one of $S_{\text{sel\_col}} \nsubseteq S_{\text{pred\_col}}$, $S_{\text{pred\_cell}} \neq S_{\text{cell}}$ and $S_{\text{pred\_col}} \nsubseteq S_{\text{sel\_col}} \cup S_{\text{other\_col}}$ is true, $r(\tilde{x}) = -1$.

### 4.4 Training and Inference

To train the anonymization model and the parser as a whole, we regard the anonymous utterance $\tilde{x}$ as the hidden variable for the neural semantic parser $p(y|x)$. Then, maximizing the log likelihood of $p(y|x)$ is equivalent to maximizing its Evidence Lower BOund (ELBO) (Kim et al., 2018), i.e.,

$$\text{ELBO}\,(\varphi, \theta) = \mathbb{E}_{q_\theta(\tilde{x}|x)} \left[ \log \frac{p_\varphi(y, \tilde{x}|x)}{q_\theta(\tilde{x}|x)} \right] \quad (10)$$

One popular strategy to maximizing ELBO is the coordinate ascent. Specifically, it iteratively executes following two steps (Neal and Hinton, 1998): 1) *variational E-step*, which maximizes ELBO w.r.t. $\theta$ keeping $\varphi$ fixed, i.e., $\theta^\star = \arg\min_\theta \mathbb{KL}\left(q_\theta(\tilde{x}|x) \| p_\varphi(\tilde{x}|y, x)\right)$; and 2) *variational M-step*, which maximizes ELBO w.r.t. $\varphi$ keeping $\theta$ fixed, i.e., $\varphi^\star = \arg\max_\varphi \mathbb{E}_{q_\theta(\tilde{x}|x)} \left[\log p_\varphi(y, \tilde{x}|x)\right]$.

In our scenario, $\varphi$ and $\theta$ refer to the learnable parameters in the neural semantic parser and the anonymization model respectively. For variational E-step, it usually finds the best variational approximation to the true posterior (Neal and Hinton, 1998). As discussed in Section 4.2, the true posterior we can obtain is in the form of the unordered set. Thus, we actually minimize the the expected reward, i.e., $J_{\text{ER}}(\theta)$, instead of the KL divergence (see Section 4.3). For variational M-step, to save training time, we simply sample one example greedily, which approximates to maximizing log likelihood of $p_\varphi(y, \tilde{x}|x)$, i.e., $J_{\text{MLE}}(\varphi)$. Moreover, since performing coordinate ascent on the entire dataset is too expensive, the variational E-step and M-step are usually performed over mini-batches (Hoffman et al., 2013). To further improve time efficiency, we optimize objectives of the variational E-step and M-step simultaneously instead of alternatively. Thus, the actual objective is,

$$J(\varphi, \theta) = \underbrace{\sum_{(x,y)\in\mathcal{D}} \log p_\varphi(y|\tilde{x}, x)}_{J_{\text{MLE}}(\varphi)} + \underbrace{\sum_{(x,\tilde{x})\in\mathcal{D}} \mathbb{E}_{\tilde{x}\sim q_\theta(\cdot|x)} r(\tilde{x})}_{J_{\text{ER}}(\theta)}, \quad (11)$$

where $J_{\text{MLE}}(\varphi)$ is the log likelihood of the generated SQL query given the input utterance and the anonymous utterance, and $J_{\text{ER}}(\theta)$ is the expected reward of the anonymous utterance given the input utterance (see Eqn (8) for details.)

Specifically, when optimizing $J_{\text{MLE}}(\varphi)$, we use the concatenation of the encoding of the input utterance (denoted as $e(x_t)$) and the encoding of the anonymous utterance (denoted as $h(\tilde{x}_t)$) as the input for the neural semantic parser, i.e., $g_t = [e(x_t); h(\tilde{x}_t)]$, $t \in \{1, \ldots, T\}$ (see Figure 2). For $e(x_t)$, it is determined by the parser itself. For $h(\tilde{x}_t)$, it is concatenated by two parts: 1) embedding of the channel name, i.e., COL, CELL and UNK, 2) embedding of index $k$ when the channel is COL, indicating that the $t$-th token is related to the $k$-th column name in the table.

At test time, the prediction for input utterance $x$ is obtained by $\hat{x} = \arg\max_{\tilde{x}'} p\left(\tilde{x}'|x\right)$ and $\hat{y} = \arg\max_{y'} p\left(y'|\hat{x}, x\right)$.

## 5 Experiments

### 5.1 Experimental Setup

We conduct experiments on the WikiSQL (Zhong et al., 2017) and Spider (Yu et al., 2018c). Each example consists of a natural language question, a SQL query and a table. On WikiSQL, one question is only related to one table; while on Spider, one question is usually related to multiple tables.

Our model is implemented in PyTorch (Paszke et al., 2017). The type of the input encoder in the anonymization model (i.e., BiLSTM or BERT) is set the same as that in the concrete parser. Embedding vectors of the anonymous utterance are initiated by GloVe (Pennington et al., 2014). We

| Method | Dev (%) | | Test (%) | |
|---|---|---|---|---|
| | $ACC_{QM}$ | $ACC_{EX}$ | $ACC_{QM}$ | $ACC_{EX}$ |
| Seq2SQL (Zhong et al., 2017)[1] | 53.5 | 62.1 | 51.6 | 60.4 |
| Seq2SQL + DAE | **60.0**(+**6.5**) | **67.5**(+**5.4**) | **58.0**(+**6.4**) | **66.1**(+**5.7**) |
| SQLNet (Xu et al., 2017) | 63.2 | 69.8 | 61.3 | 68.0 |
| SQLNet + DAE | **64.6**(+**1.4**) | **71.1**(+**1.3**) | **63.4**(+**2.1**) | **70.0**(+**2.0**) |
| TypeSQL (Yu et al., 2018a) | 68.0 | 74.5 | 66.7 | 73.5 |
| TypeSQL + DAE | **68.6**(+**0.6**) | 74.5 | **67.8**(+**1.1**) | **74.1**(+**0.6**) |
| SQLova (Hwang et al., 2019)[2] | 80.3 | 85.8 | 79.4 | 85.2 |
| SQLova + DAE | **81.3**(+**1.0**) | **86.4**(+**0.6**) | **80.5**(+**1.1**) | **86.1**(+**0.9**) |
| – Supervision | 81.0 | 86.1 | 80.0 | 85.4 |
| – Co-training | 80.6 | 86.0 | 79.8 | 85.3 |

[1] For Seq2SQL, we use Xu et al. (2017)'s version because it achieves better performance.
[2] For SQLova, we use BERT-Base as the input encoder due to limited computation resources.

Table 1: Performance improvement of the neural semantic parser on WikiSQL.

| Method | Easy (%) | Medium (%) | Hard (%) | Extra Hard (%) | All (%) |
|---|---|---|---|---|---|
| SyntaxSQLNet (Yu et al., 2018b) | 38.4 | 15.0 | 16.1 | 3.5 | 18.9 |
| SyntaxSQLNet + DAE | **39.6**(+**1.2**) | **18.2**(+**3.2**) | **20.7**(+**4.6**) | **7.6**(+**4.1**) | **22.1**(+**3.2**) |
| SyntaxSQLNetAug (Yu et al., 2018b) | 44.4 | 23.0 | 23.0 | 2.9 | 24.9 |
| SyntaxSQLNetAug + DAE | **44.8**(+**0.4**) | **27.0**(+**4.0**) | **24.1**(+**1.1**) | **5.9**(+**3.0**) | **27.4**(+**2.5**) |

Table 2: Performance improvement of the neural semantic parser on Spider with different hardness levels.

| | #Total | #Distinct $x$ | #Distinct $\tilde{x}$ |
|---|---|---|---|
| Dev | 8421 | 8387 | 5488 |
| Test | 15878 | 15828 | 9680 |

Table 3: The number of distinct input utterances and distinct anonymous utterances on WikiSQL.

use the manually labeled training data (which is 10% of the whole training data) to initiate our anonymization model and then optimize the entire framework on the whole training data. All the other hyperparameters, e.g., the learning rate, hyperparameters in ADAM optimizer, the number of training epochs, etc., are tuned on the dev set[2].

### 5.2 Performance of Neural Semantic Parsing

First, we show that reducing the lexical problem through the anonymization model can improve the performance of neural semantic parser.

To this end, we add the anonymization model to typical neural semantic parsers as presented in Section 4. We use '[$A$]+DAE' to denote the neural semantic parser with the anonymization model, where $A$ stands for the original name of the concrete parser and **DAE** is the abbreviated name of our approach, i.e., data-anonymous encoding.

[2]For Spider, we tune hyperparameters on training set and test the model on dev set.

On WikiSQL, the performance is evaluated by query-match accuracy ($ACC_{QM}$) and execution accuracy ($ACC_{EX}$), which measure accuracies of canonical representation and execution result matches between the predicted SQL and the ground truth respectively (Yu et al., 2018a). Table 1 shows the results. First, we can observe that query match accuracy on test data can be improved by 6.4% at most and 1.1% at least. Furthermore, for TypeSQL, query match accuracy can be further improved by 1.1% although it has used string-match based approach to anonymize table-related tokens. Moreover, we perform ablation studies by 1) removing the supervision for the anonymization model (denoted as '-Supervision' in Table 1), and 2) simply using the output of the trained anonymization model as the input for the parser without training them as a whole (denoted as '-Co-training' in Table 1). We can observe that the performance improvement is limited without supervision and co-training, indicating that both of them are indispensable.

On Spider, the performance is evaluated by exact-match accuracy on different difficulty levels of SQL queries, i.e., easy, medium, hard and extra hard. (Yu et al., 2018c). Table 2 shows the results. First, the overall accuracy can be improved by 3.2% and 2.5% respectively. Furthermore, per-

| | |
|---|---|
| $\tilde{x}$ | which *column* have *column* of *cell*? |
| $x$ | which place [*column*] has a rank [*column*] of 71 [*cell*]?<br>which county [*column*] has a median household income [*column*] of $98,090 [*cell*] ? |
| $\tilde{x}$ | what be *column* when *column* be *cell*? |
| $x$ | what is the inclination [*column*] when the alt name [*column*] is ops-1584 [*cell*]?<br>what was the district [*column*] when the reason for change [*column*] was died january 1, 1974 [*cell*]? |
| $\tilde{x}$ | name *column* for *cell* |
| $x$ | name the candidates [*column*] for georgia 8 [*cell*]<br>name the party [*column*] for jack thomas brinkley [*cell*] |

Table 4: Top frequent anonymous utterances on dev set of WikiSQL. The tokens with underlines are table-related tokens. We use '[*column*]' and '[*cell*]' to differentiate whether the token is related to a column name or a cell.

| Method | Dev (%) | | | Test (%) | | |
|---|---|---|---|---|---|---|
| | $ACC_{SC}$ | $ACC_{OC}$ | $ACC_{CE}$ | $ACC_{SC}$ | $ACC_{OC}$ | $ACC_{CE}$ |
| TypeSQL (Yu et al., 2018a) | 75.9 | 92.9 | – | 76.0 | 92.9 | – |
| AnnotatedSeq2Seq (Wang et al., 2018b) | 88.8 | 64.6 | – | 88.8 | 63.6 | – |
| DAE | **92.6** | **93.6** | **86.7** | **92.0** | **93.7** | **86.2** |

Table 5: Performances of different anonymization models on WikiSQL.

formances on medium, hard and extra hard SQL queries achieve more improvement than that on easy SQL queries, indicating that our approach is more helpful for solving complicated cases.

### 5.3 Performance of Reducing Input Utterances

To further demonstrate the effectiveness of reducing the lexical problem, we show that different input utterance can be reduced to the same anonymous utterance. To this end, we process input utterances and anonymous utterances by 1) converting the characters to the lowercase, 2) lemmatizing the tokens and 3) removing the articles.

Table 3 shows that by anonymizing table-related tokens, the number of distinct utterances is reduced from 8387 to 5488 on dev set and from 15828 to 9680 on test set. Furthermore, Table 4 shows three anonymous utterances that are most frequent on dev set and examples of corresponding input utterances. All of these indicate that although input utterances can hardly be identical, they often share the same anonymous utterance.

### 5.4 Performance of Anonymization Methods

In addition, we compare performances of different anonymization methods, including 1) TypeSQL, which uses exact string match to detect column names, 2) AnnotatedSeq2Seq, which detects column names by setting a threshold for the edit distance and the cosine similarity of word embedding, and 3) DAE, our learning-based approach[3].

Anonymization methods are evaluated by following metrics: 1) whether the column names in SELECT clause is included in the prediction ($ACC_{SC} = \frac{\sum_{i=1}^{Z} \mathbb{I}(S^i_{\text{sel\_col}} \subseteq S^i_{\text{pred\_col}})}{Z}$); 2) whether there is wrongly predicted column names ($ACC_{OC} = \frac{\sum_{i=1}^{Z} \mathbb{I}(S^i_{\text{pred\_col}} \subseteq S^i_{\text{sel\_col}} \cup S^i_{\text{other\_col}})}{Z}$); and 3) whether all the cells in the SQL query are correctly predicted ($ACC_{CE} = \frac{\sum_{i=1}^{Z} \mathbb{I}(S^i_{\text{pred\_cell}} = S^i_{\text{cell}})}{Z}$). Here, $Z$ is the amount of test data, $\mathbb{I}(\cdot)$ is the indicator function, and superscript $i$ is the index of data.

Table 5 shows that DAE significantly outperforms TypeSQL and AnnotatedSeq2Seq on all the evaluation metrics. First, for $ACC_{SC}$, DAE outperforms TypeSQL and AnotatedSeq2Seq by 16% and 3.5% on test data; for $ACC_{OC}$, DAE outperforms TypeSQL and AnnotatedSeq2Seq by 0.8% and 28% on test data. Moreover, DAE can achieve around 86% for $ACC_{CE}$, while other methods fail to recognize cells when the table content is not available due to the privacy problem.

## 6 Conclusion

In this work, we propose a learning-based approach to reduce the lexical problem before the

[3]For experiments in this subsection, to make our anonymization model in the same settings as the others, we use BiLSTM as the input encoder and do not train the anonymization model with neural semantic parser as a whole.

neural semantic parser on text-to-SQL generation. Specifically, we propose a two-stage anonymization model and leverage implicit supervision from SQL queries by policy gradient to guide its training. In the future, we plan to improve the performance of the anonymization model by exploring more efficient expected reward. In addition, we also plan to extend our approach to the tasks with question-denotation pairs as supervision.

## References


Ion Androutsopoulos, Graeme D Ritchie, and Peter Thanisch. 1995. Natural language interfaces to databases–an introduction. *Natural language engineering*, 1(1):29–81.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 33–43.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 731–742.

Omer Goldman, Veronica Latcinnik, Ehud Nave, Amir Globerson, and Jonathan Berant. 2018. Weakly supervised semantic parsing with abstract examples. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 1809–1819.

Jonathan Herzig and Jonathan Berant. 2017. Neural semantic parsing over multiple knowledge-bases. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 623–628.

Jonathan Herzig and Jonathan Berant. 2018. Decoupling structure and lexicon for zero-shot semantic parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1619–1629.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. 2013. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347.

Wonseok Hwang, Jinyeung Yim, Seunghyun Park, and Minjoon Seo. 2019. A comprehensive exploration on wikisql with table-aware word contextualization. *arXiv preprint arXiv:1902.01069*.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 12–22.

Yoon Kim, Sam Wiseman, and Alexander M Rush. 2018. A tutorial on deep latent variable models of natural language. *arXiv preprint arXiv:1812.06834*.

Nikolaos Kolitsas, Octavian-Eugen Ganea, and Thomas Hofmann. 2018. End-to-end neural entity linking. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 519–529.

Fei Li and HV Jagadish. 2014. Constructing an interactive natural language interface for relational databases. *Proceedings of the VLDB Endowment*, 8(1):73–84.

Radford M Neal and Geoffrey E Hinton. 1998. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer.

Yusuke Oda, Hiroyuki Fudaba, Graham Neubig, Hideaki Hata, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2015. Learning to generate pseudo-code from source code using statistical machine translation. In *30th IEEE/ACM International Conference on Automated Software Engineering*, pages 574–584.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 1470–1480.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. *NIPS Workshop*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing*, pages 1532–1543.

Wei Shen, Jianyong Wang, and Jiawei Han. 2015. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Transactions on Knowledge and Data Engineering*, 27(2):443–460.

Tianze Shi, Kedar Tatwawadi, Kaushik Chakrabarti, Yi Mao, Oleksandr Polozov, and Weizhu Chen. 2018. Incsql: Training incremental text-to-sql parsers with non-deterministic oracles. *arXiv preprint arXiv:1809.05054*.

Alane Suhr, Mike Lewis, James Yeh, and Yoav Artzi. 2017. A corpus of natural language for visual reasoning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 217–223.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Chenglong Wang, Kedar Tatwawadi, Marc Brockschmidt, Po-Sen Huang, Yi Mao, Oleksandr Polozov, and Rishabh Singh. 2018a. Robust text-to-sql generation with execution-guided decoding. *arXiv preprint arXiv:1807.03100*.

Wenlu Wang, Yingtao Tian, Hongyu Xiong, Haixun Wang, and Wei-Shinn Ku. 2018b. A transfer-learnable natural language interface for databases. *arXiv preprint arXiv:1809.02649*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Xiaojun Xu, Chang Liu, and Dawn Song. 2017. Sqlnet: Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv:1711.04436*.

Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. 2018a. Typesql: Knowledge-based type-aware neural text-to-sql generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*, volume 2, pages 588–594.

Tao Yu, Michihiro Yasunaga, Kai Yang, Rui Zhang, Dongxu Wang, Zifan Li, and Dragomir Radev. 2018b. Syntaxsqlnet: Syntax tree networks for complex and cross-domain text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1653–1663.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018c. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, pages 658–666.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

# Capturing Argument Interaction in Semantic Role Labeling with Capsule Networks

**Xinchi Chen**[1] **Chunchuan Lyu**[1] **Ivan Titov**[1,2]
xchen13@exseed.ed.ac.uk chunchuan.lv@gmail.com ititov@inf.ed.ac.uk
[1] ILCC, School of Informatics, University of Edinburgh
[2]ILLC, University of Amsterdam

## Abstract

Semantic role labeling (SRL) involves extracting propositions (i.e. predicates and their typed arguments) from natural language sentences. State-of-the-art SRL models rely on powerful encoders (e.g., LSTMs) and do not model non-local interaction between arguments. We propose a new approach to modeling these interactions while maintaining efficient inference. Specifically, we use Capsule Networks (Sabour et al., 2017): each proposition is encoded as a tuple of *capsules*, one capsule per argument type (i.e. role). These tuples serve as embeddings of entire propositions. In every network layer, the capsules interact with each other and with representations of words in the sentence. Each iteration results in updated proposition embeddings and updated predictions about the SRL structure. Our model substantially outperforms the non-refinement baseline model on all 7 CoNLL-2019 languages and achieves state-of-the-art results on 5 languages (including English) for dependency SRL. We analyze the types of mistakes corrected by the refinement procedure. For example, each role is typically (but not always) filled with at most one argument. Whereas enforcing this approximate constraint is not useful with the modern SRL system, iterative procedure corrects the mistakes by capturing this intuition in a flexible and context-sensitive way.[1]

## 1 Introduction

The task of semantic role labeling (SRL) involves the prediction of predicate-argument structure, i.e., both the identification of arguments and labeling them with underlying semantic roles. The shallow semantic structures have been shown beneficial in many natural language processing (NLP) applications, including information extraction (Christensen et al., 2011), question answering (Eckert and Neves, 2018) and machine translation (Marcheggiani et al., 2018).

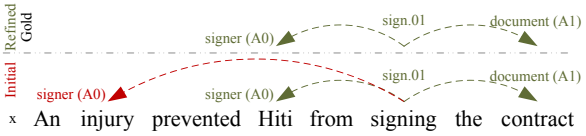


Figure 1: An example predicate-argument structure: green and red arcs denote correct and wrong predictions, respectively.

In this work, we focus on the dependency version of the SRL task (Surdeanu et al., 2008). An example of a dependency semantic-role structure is shown in Figure 1. Edges in the graph are marked with semantic roles, whereas predicates (typically verbs or nominalized verbs) are annotated with their senses.

Intuitively, there are many restrictions on potential predicate-argument structures. Consider the 'role uniqueness' constraint: each role is typically, but not always, realized at most once. For example, predicates have at most one *agent*. Similarly, depending on a verb class, only certain subcategorization patterns are licensed. Nevertheless, rather than modeling the interaction between argument labeling decisions, state-of-the-art semantic role labeling models (Marcheggiani and Titov, 2017; Cai et al., 2018; Li et al., 2019b) rely on powerful sentence encoders (e.g., multi-layer BiLSTMs (Zhou and Xu, 2015; Qian et al., 2017; Tan et al., 2018)). This contrasts with earlier work on SRL, which hinged on global declarative constraints on the labeling decisions (FitzGerald et al., 2015; Täckström et al., 2015; Das et al., 2012). Modern SRL systems are much more accurate and hence enforcing such hard and often approximate constraints is unlikely to be as beneficial (see our experiments in Section 7.2).

[1]Code: https://github.com/DalstonChen/CapNetSRL.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5415–5425,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

Instead of using hard constraints, we propose to use a simple iterative structure-refinement procedure. It starts with independent predictions and refines them in every subsequent iteration. When refining a role prediction for a given candidate argument, information about the assignment of roles to other arguments gets integrated. Our intuition is that modeling interactions through the output spaces rather than hoping that the encoder somehow learns to capture them, provides a useful inductive bias to the model. In other words, capturing this interaction explicitly should mitigate overfitting. This may be especially useful in a lower-resource setting but, as we will see in experiments, it appears beneficial even in a high-resource setting.

We think of semantic role labeling as extracting propositions, i.e. predicates and argument-role tuples. In our example, the proposition is `sign.01`(`Arg0:signer` = *Hiti*, `Arg1:document` = *contract*). Across the iterations, we maintain and refine not only predictions about propositions but also their embeddings. Each 'slot' (e.g., correspond to role `Arg0:signer`) is represented with a vector, encoding information about arguments assumed to be filling this slot. For example, if *Hiti* is predicted to fill slot `Arg0` in the first iteration then the slot representation will be computed based on the contextualized representation of that word and hence reflect this information. The combination of all slot embeddings (one per semantic role) constitutes the embedding of the proposition. The proposition embedding, along with the prediction about the semantic role structure, gets refined in every iteration of the algorithm.

Note that, in practice, the predictions in every iteration are soft, and hence proposition embeddings will encode current beliefs about the predicate-argument structure. The distributed representation of propositions provides an alternative ("dense-embedding") view on the currently considered semantic structure, i.e. information extracted from the sentence. Intuitively, this representation can be readily tested to see how well current predictions satisfy selection restrictions (e.g., *contract* is a very natural filler for `Arg1:document`) and check if the arguments are compatible.

To get an intuition how the refinement mechanism may work, imagine that both *Hiti* and *injury* are labeled as `Arg0` for `sign.01` in the first iteration. Hence, the representation of the slot `Arg0` will encode information about both predicted arguments. At the second iteration, the word *injury* will be aware that there is a much better candidate for filling the `signer` role, and the probability of assigning *injury* to this role will drop. As we will see in our experiments, whereas enforcing the hard uniqueness constraint is not useful, our iterative procedure corrects the mistakes of the base model by capturing interrelations between arguments in a flexible and context-sensitive way.

In order to operationalize the above idea, we take inspiration from the Capsule Networks (CNs) (Sabour et al., 2017). Note that we are not simply replacing Bi-LSTMs with generic CNs. Instead, we use CN to encode the structure of the refinement procedure sketched above. Each slot embedding is now a *capsule* and each proposition embedding is a tuple of capsules, one capsule per role. In every iteration (i.e. CN network layer), the capsules interact with each other and with representations of words in the sentence.

We experiment with our model on standard benchmarks for 7 languages from CoNLL-2019. Compared with the non-refinement baseline model, we observe substantial improvements from using the iterative procedure. The model achieves state-of-the-art performance in 5 languages, including English.

## 2 Base Dependency SRL Model

In dependency SRL, for each predicate $p$ of a given sentence $x = \{x_1, x_2, \cdots, x_n\}$ with $n$ words, the model needs to predict roles $y = \{y_1, y_2, \cdots, y_n\}$ for every word. The role can be *none*, signifying that the corresponding word is not an argument of the predicate.

We start with describing the factorized baseline model which is similar to that of Marcheggiani et al. (2017). It consists of three components: (1) an embedding layer; (2) an encoding layer and (3) an inference layer.

### 2.1 Embedding Layer

The first step is to map symbolic sentence $x$ and predicate $p$ into continuous embedded space:

$$\mathbf{e}_i = \text{Lookup}(x_i), \quad (1)$$

$$\mathbf{p} = \text{Lookup}(p), \quad (2)$$

where $\mathbf{e}_i \in \mathcal{R}^{d_e}$ and $\mathbf{p} \in \mathcal{R}^{d_e}$.

### 2.2 Encoding Layer

The encoding layer extracts features from input sentence $x$ and predicate $p$. We extract features from input sentence $x$ using stacked bidirectional LSTMs (Hochreiter and Schmidhuber, 1997):

$$\mathbf{x}_i = \text{Bi-LSTMs}(\mathbf{e}_i), \tag{3}$$

where $\mathbf{x}_i \in \mathcal{R}^{d_l}$. Then we represent each role logit of each word by a bi-linear operation with the target predicate:

$$b_{j|i} = \mathbf{x}_i \mathbf{W} \mathbf{p}, \tag{4}$$

where $\mathbf{W} \in \mathcal{R}^{d_l \times d_e}$ is a trainable parameter and $b_{j|i} \in \mathcal{R}$ is a scalar representing the logit of role $j$ for word $x_i$.

### 2.3 Inference Layer

The probability $P(y|x,p)$ is then computed as

$$P(y|x,p) = \prod_i P(y_i|x,i,p), \tag{5}$$

$$= \prod_i \text{softmax}(\mathbf{b}_{\cdot|i})_{y_i}, \tag{6}$$

where $\mathbf{b}_{\cdot|i} = \{b_{1|i}, b_{2|i}, \cdots, b_{|\mathcal{T}||i}\}$ and $|\mathcal{T}|$ denotes the number of role types.

## 3 Dependency based Semantic Role Labeling using Capsule Networks

Inspired by capsule networks, we use capsule structure for each role state to maintain information across iterations and employ the dynamic routing mechanism to derive the role logits $b_{j|i}$ iteratively. Figure 2 illustrates the architecture of the proposed model.

### 3.1 Capsule Structure

We start by introducing two capsule layers: (1) the word capsule layer and (2) the role capsule layer.

#### 3.1.1 Word Capsule Layer

The word capsule layer is comprised of capsules representing the roles of each word. Given sentence representation $\mathbf{x}_i$ and predicate embedding $\mathbf{p}$, the word capsule layer is derived as:

$$u^k_{j|i} = \mathbf{x}_i \mathbf{W}^k_j \mathbf{p}; \;\; k = 1, 2, \cdots, K, \tag{7}$$

where $\mathbf{W}^k_j \in \mathcal{R}^{d_l \times d_e}$ and $\mathbf{u}_{j|i} \in \mathcal{R}^K$ is the capsule vector for role $j$ of word $x_i$. $K$ denotes the capsule size. Intuitively, the capsule encodes the argument-specific information relevant to deciding if role $j$ is suitable for the word. These capsules do not get iteratively updated.

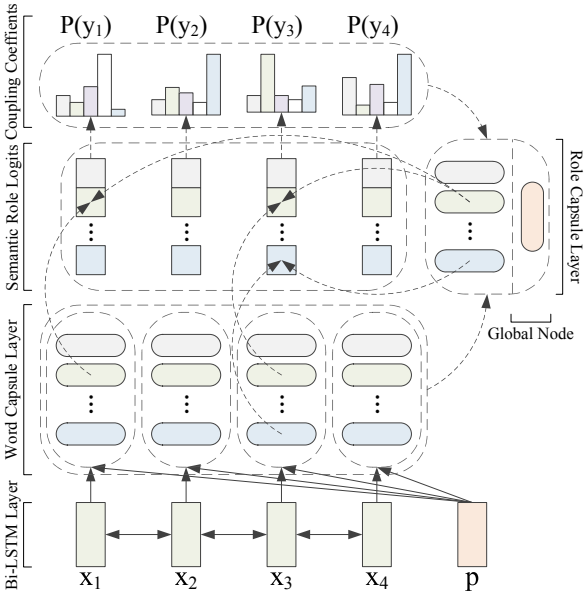


Figure 2: Architecture of the proposed CapsuleNet with the global capsule node. The dashed arcs are iterable.

#### 3.1.2 Role Capsule Layer

As discussed in the introduction, the role capsule layer could be viewed as an embedding of a proposition. The capsule network generates the capsules in the layer using "routing-by-agreement". This process can be regarded as a pooling operation. Capsules in the role capsule layer at $t$-th iteration are derived as:

$$\mathbf{v}^{(t)}_j = \text{Squash}(\mathbf{s}^{(t)}_j) = \frac{||\mathbf{s}^{(t)}_j||^2}{1 + ||\mathbf{s}^{(t)}_j||^2} \frac{\mathbf{s}^{(t)}_j}{||\mathbf{s}^{(t)}_j||}, \tag{8}$$

where intermediate capsule $s^{(t)}_j$ is generated with the linear combination of capsules in the word capsule layer with weights $c^{(t)}_{ij}$:

$$\mathbf{s}^{(t)}_j = \sum_i c^{(t)}_{ij} \mathbf{u}_{j|i}. \tag{9}$$

Here, $c^{(t)}_{ij}$ are coupling coefficients, calculated by "softmax" function over role logits $b^{(t)}_{j|i}$ . The coefficients $c^{(t)}_{ij}$ can be interpreted as the probability that word $x_i$ is assigned role $j$:

$$\mathbf{c}^{(t)}_i = \text{softmax}(\mathbf{b}^{(t)}_{\cdot|i}). \tag{10}$$

The role logits $b^{(t)}_{j|i}$ are decided by the iterative *dynamic routing* process. The Squash operation will deactive capsules receiving small input $\mathbf{s}^{(t)}_j$ (i.e. roles not predicted in the sentence) by pushing them further to the $\mathbf{0}$ vector.

**Input:** # of iterations $T$, all capsules in $l_w$ $\mathbf{u}_{j|i}$
**Result:** semantic role logits $b_{j|i}^{(T)}$
Initialization: $b_{j|i}^{(0)} = 0$ ;
**for** $t \in [0, 1, \cdots, T-1]$ **do**
  for all capsules in $l_w$: $\mathbf{c}_i^{(t)} = \text{softmax}(\mathbf{b}_{\cdot|i}^{(t)})$;
  for all capsules in $l_r$: $\mathbf{s}_j^{(t)} = \sum_i c_{ij}^{(t)} \mathbf{u}_{j|i}$;
  for all capsules in $l_r$: $\mathbf{v}_j^{(t)} = \text{Squash}(\mathbf{s}_j^{(t)})$;
  for all capsules in $l_w$ and $l_r$: $b_{j|i}^{(t+1)} = b_{j|i}^{(t)} + \mathbf{v}_j^{(t)} \mathbf{W} \mathbf{u}_{j|i}$;
**end**

**Algorithm 1:** Dynamic routing algorithm. $l_w$ and $l_r$ denote word capsule layer and role capsule layer, respectively.

### 3.2 Dynamic Routing

The dynamic routing process involves $T$ iterations. The role logits $b_{j|i}$ before first iteration are all set to zeros: $b_{j|i}^{(0)} = 0$. Then, the dynamic routing process updates the role logits $b_{j|i}$ by modeling agreement between capsules in two layers:

$$b_{j|i}^{(t+1)} = b_{j|i}^{(t)} + \mathbf{v}_j^{(t)} \mathbf{W} \mathbf{u}_{j|i}, \tag{11}$$

where $\mathbf{W} \in \mathcal{R}^{K \times K}$.

The whole dynamic routing process is shown in Algorithm 1. The dynamic routing process can be regarded as the role refinement procedure (see Section 5 for details).

## 4 Incorporating Global Information

When computing the $j$-th role capsule representation (Eq 9), the information originating from an $i$-th word (i.e. $u_{j|i}$) is weighted by the probability of assigning role $j$ to word $x_i$ (i.e. $c_{ij}$). In other words, the role capsule receives messages only from words assigned to its role. This implies that the only interaction the capsule network can model is competition between arguments for a role.[2] Note though that this is different from imposing the hard role-uniqueness constraint, as the network does this dynamically in a specific context. Still, this is a strong restriction.

In order to make the model more expressive, we further introduce a global node $\mathbf{g}^{(t)}$ to incorporate global information about all arguments at the current iterations. The global node is a compressed representation of the entire proposition, and used in the prediction of all arguments, thus permitting arbitrary interaction across arguments. The global node $\mathbf{g}^{(t)}$ at $t$-th iteration is derived as:

$$\mathbf{g}^{(t)} = \mathbf{W} \mathbf{s}^{(t)}, \tag{12}$$

where $\mathbf{W} \in \mathcal{R}^{K \times (K \cdot |\mathcal{T}|)}$. Here, $\mathbf{s}^{(t)} = \mathbf{s}_1^{(t)} \oplus \mathbf{s}_2^{(t)} \oplus \cdots \oplus \mathbf{s}_{|\mathcal{T}|}^{(t)} \in \mathcal{R}^{K \cdot |\mathcal{T}|}$ is the concatenation of all capsules in the role capsule layer.

We append an additional term for the role logits update in Eq (11):

$$b_{j|i}^{(t+1)} = b_{j|i}^{(t)} + \mathbf{v}_j^{(t)} \mathbf{W} \mathbf{u}_{j|i} + \mathbf{g}^{(t)} \mathbf{W}_g \mathbf{u}_{j|i}, \tag{13}$$

where $\mathbf{W} \in \mathcal{R}^{K \times K}$ and $\mathbf{W}_g \in \mathcal{R}^{K \times K}$.

## 5 Refinement

The dynamic routing process can be seen as iterative role refinement. Concretely, the coupling coefficients $\mathbf{c}_i^{(t)}$ in Eq (10) can be interpreted as the predicted distribution of the semantic roles for word $x_i$ in Eq (5) at $t$-th iteration:

$$\mathbf{c}_i^{(t)} = P^{(t)}(\cdot | x, i, p). \tag{14}$$

Since dynamic routing is an iterative process, semantic role distribution $\mathbf{c}_i^{(t)}$ in iteration $t$ will affect the semantic role distribution $\mathbf{c}_i^{(t+1)}$ in next iteration $t+1$:

$$\mathbf{c}_i^{(t+1)} = f(\mathbf{c}_i^{(t)}, x, i, p), \tag{15}$$

where $f(\cdot)$ denotes the refinement function defined by the operations in each dynamic routing iteration.

## 6 Training

We minimize the following loss function $\mathcal{L}(\theta)$:

$$\mathcal{L}(\theta) = -\frac{1}{n} \sum_i^n \log P^{(T)}(y_i | x, i, p; \theta) + \lambda ||\theta||_2^2, \tag{16}$$

where $\lambda$ is a hyper-parameter for the regularization term and $P^{(T)}(y_i | x, i, p; \theta) = \mathbf{c}_i^{(T)}$. Unlike standard refinement methods (Belanger et al., 2017; Lee et al., 2018) which sum losses across all refinement iterations, our loss is only based on the prediction made at the last iteration. This encourages the model to rely on the refinement process rather than do it in one shot.

Our baseline model is trained analogously, but using the cross-entropy for the independent classifiers (Eq (5)).

[2] In principle, it can model the opposite, i.e. collaboration / attraction but it is unlikely to be useful in SRL.

**Uniqueness Constraint Assumption** As we discussed earlier, for a given target predicate, each semantic role will typically appear at most once. To encode this intuition, we propose another loss term $\mathcal{L}_u(\theta)$:

$$\mathcal{L}_u(\theta) = \frac{1}{|\mathcal{T}|} \sum_j \log \text{softmax}(\mathbf{b}_{j|\cdot}^{(T)}), \quad (17)$$

where $\mathbf{b}_{j|\cdot}^{(T)}$ are the semantic role logits in $T$-th iteration. Thus, the final loss $\mathcal{L}^*(\theta)$ is the combination of the two losses:

$$\mathcal{L}^*(\theta) = \mathcal{L}(\theta) + \eta \mathcal{L}_u(\theta), \quad (18)$$

where $\eta$ is a discount coefficient.

## 7 Experiments

### 7.1 Datasets & Configuration

we conduct experiments on CoNLL-2009 (Hajič et al., 2009) for all languages, including Catalan (Ca), Chinese (Zh), Czech (Cz), English (En), German (De), Japanese (Jp) and Spanish (Es). As standard we do not consider predicate identification: the predicate position is provided as an input feature. We use the labeled 'semantic' F1 which jointly scores sense prediction and role labeling.

We use ELMo (Peters et al., 2018) (dimension $d_e$ as 1024) for English, and FastText embeddings (Grave et al., 2018) (dimension $d_e$ as 300) for all other languages on both the baseline model and the proposed CapsuleNet. LSTM state dimension $d_l$ is 500. Capsule size $K$ is 16. Batch size is 32. The coefficient for the regularization term $\lambda$ is 0.0004. We employ Adam (Kingma and Ba, 2015) as the optimizer and the initial learning rate $\alpha$ is set to 0.0001. Syntactic information is not utilized in our model.

### 7.2 Model Selection

Table 1 shows the performance of our model trained with loss $\mathcal{L}^*(\theta)$ for different values of discount coefficient $\eta$ on the English development set. The model achieves the best performance when $\eta$ equals to 0. It implies that adding uniqueness constraint on loss actually hurts the performance. Thus, we use the loss $\mathcal{L}^*(\theta)$ with $\eta$ equals to 0 in the rest of the experiments, which is equivalent to the loss $\mathcal{L}(\theta)$ in Eq (16). We also observe that the model with 2 refinement iterations performs the best (89.92% F1).

| $\eta$ | # of Iters | P | R | F | EM |
|---|---|---|---|---|---|
| 1.0 | 1 | **89.31** | 88.31 | 88.80 | 63.21 |
| | 2 | 88.34 | 88.92 | 88.63 | 62.80 |
| | 3 | 88.92 | **88.97** | **88.94** | **63.51** |
| 0.1 | 1 | 89.16 | 87.99 | 88.57 | 62.94 |
| | 2 | 88.70 | 88.89 | 88.80 | 63.00 |
| | 3 | **89.75** | **89.40** | **89.57** | **66.01** |
| 0.0 | 1 | 89.59 | **90.06** | 89.82 | 66.74 |
| | 2 | **89.88** | 89.97 | **89.92** | **66.93** |
| | 3 | 89.25 | 89.61 | 89.43 | 65.98 |

Table 1: The performance of our model (w/o global node) with different discount coefficients $\eta$ of loss $\mathcal{L}^*(\theta)$ on the English development set. EM denotes the ratio of exact match on propositions.

| Models | Test | Ood |
|---|---|---|
| Lei et al. (2015) | 86.60 | 75.60 |
| FitzGerald et al. (2015) | 87.70 | 75.50 |
| Foland and Martin (2015) | 86.00 | 75.90 |
| Roth and Lapata (2016) | 87.90 | 76.50 |
| Swayamdipta et al. (2016) | 85.00 | - |
| Marcheggiani et al. (2017) | 87.70 | 77.70 |
| Marcheggiani and Titov (2017) | 89.10 | 78.90 |
| He et al. (2018) | 89.50 | 79.30 |
| Li et al. (2018) | 89.80 | 79.80 |
| Cai et al. (2018) | 89.60 | 79.00 |
| Mulcaire et al. (2018) | 87.24 | - |
| Li et al. (2019b)† | 90.40 | 81.50 |
| Baseline† | 90.49 | 81.95 |
| **CapsuleNet† (This Work)** | **91.06** | **82.72** |

Table 2: The F1 scores of previous systems on the English test set and out-of-domain (Ood) set. † denotes that the model uses Elmo.

### 7.3 Overall Results

Table 2 compares our model with previous state-of-the-art SRL systems on English. Some of the systems (Lei et al., 2015; Li et al., 2018; Cai et al., 2018) only use local features, whereas others (Swayamdipta et al., 2016) incorporate global information at the expense of greedy decoding. Additionally, a number of systems exploit syntactic information (Roth and Lapata, 2016; Marcheggiani and Titov, 2017; He et al., 2018). Some of the results are obtained with ensemble systems (FitzGerald et al., 2015; Roth and Lapata, 2016).

As we observe, the baseline model (see Section 2) is quite strong, and outperforms the previous state-of-the-art systems on both in-domain and out-of-domain sets on English. The proposed CapsuleNet outperforms the baseline model on English (e.g. obtaining 91.06% F1 on the English test set), which shows the effectiveness of the cap-

| Models | Development | | | | Test | | | | Ood | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | P | R | F | EM | P | R | F | EM | P | R | F | EM |
| **CapsuleNet SRL** | 89.63 | **90.25** | **89.94** | **67.15** | 90.74 | **91.38** | **91.06** | **69.90** | **82.66** | **82.78** | **82.72** | **53.14** |
| w/o Global Node | 89.88 | 89.97 | 89.92 | 66.93 | **90.95** | 91.15 | 91.05 | 69.65 | 82.45 | 82.27 | 82.36 | 51.95 |
| w/o Role Capsule Layer | **89.95** | 89.64 | 89.80 | 66.70 | 90.82 | 90.66 | 90.74 | 68.46 | 82.26 | 81.64 | 81.95 | 50.75 |
| w/o Word Capsule Layer (Baseline) | 89.47 | 89.61 | 89.54 | 66.17 | 90.31 | 90.68 | 90.49 | 68.27 | 82.16 | 81.74 | 81.95 | 50.75 |

Table 3: Ablation results in English development and test sets. EM denotes the ratio of the exact match on propositions.

sule network framework. The improvement on the out-of-domain set implies that our model is robust to domain variations.

### 7.4 Ablation

Table 3 gives the performance of models with ablation on some key components, which shows the contribution of each component in our model.

- The model **without the global node** is described in Section 3.
- The model that further **removes the role capsule layer** takes the mean of capsules $\mathbf{u}_{j|i}$ in Eq (7) of word capsule layer as the semantic role logits $b_{j|i}$:

$$b_{j|i} = \frac{1}{K} \sum_{k}^{K} \mathbf{u}^{k}_{j|i}, \tag{19}$$

  where $K$ denotes the capsule size.
- The model that additionally **removes the word capsule layer** is exactly equivalent to the baseline model described in Section 2.

As we observe, CapsuleNet with all components performs the best on both development and test sets on English. The model without using the global node performs well too. It obtains 91.05% F1 on the English test set, almost the same performance as full CapsuleNet. But on the English out-of-domain set, without the global node, the performance drops from 82.72% F1 to 82.36% F1. It implies that the global node helps in model generalization. Further, once the role capsule layer is removed, the performance drops sharply. Note that the model without the role capsule layer does not use refinements and hence does not model argument interaction. It takes the mean of capsules $\mathbf{u}_{j|i}$ in the word capsule layer as the semantic role logits $b_{j|i}$ (see Eq (19)), and hence could be viewed as an enhanced ('ensembled') version of the baseline model. Note that we only introduced a very limited number of parameters for the dynamic routing mechanism (see Eq (11-13)). This suggests that the dynamic routing mechanism does genuinely captures interactions between argument labeling decisions and the performance benefits from the refinement process.

| Sent Len | # of Props |
|---|---|
| 0 - 9 | 181 |
| 10 - 19 | 1,806 |
| 20 - 29 | 3,514 |
| 30 - 39 | 3,102 |
| 40 - 49 | 1,383 |
| 50 - 59 | 391 |
| 60 - 69 | 121 |

| # of Args | # of Props |
|---|---|
| 1 | 2,591 |
| 2 | 4,497 |
| 3 | 2,189 |
| 4 | 889 |
| 5 | 259 |
| 6 | 39 |
| 7 | 7 |

Table 4: Numbers of propositions with different sentence lengths and different numbers of arguments on the English test set.

### 7.5 Error Analysis

The performance of our model while varying the sentence length and the number of arguments per proposition is shown in Figure 3. The statistics of the subsets are in Table 4. Our model consistently outperforms the baseline model, except on sentences of between 50 and 59 words. Note that the subset is small (only 391 sentences), so the effect may be random.

### 7.6 Effect of Refinement

#### 7.6.1 Refinement Performance

We take a CapsuleNet trained to perform refinements in two iterations and vary the number of iterations at test time (Figure 4). CapsuleNets with and without the global node perform very differently. CapsuleNet without the global node reduces the recall and increases the precision with every iteration. This corresponds to our expectations: the version of the network is only capable of modeling competition and hence continues with this filtering process. In contrast, CapsuleNet with the global

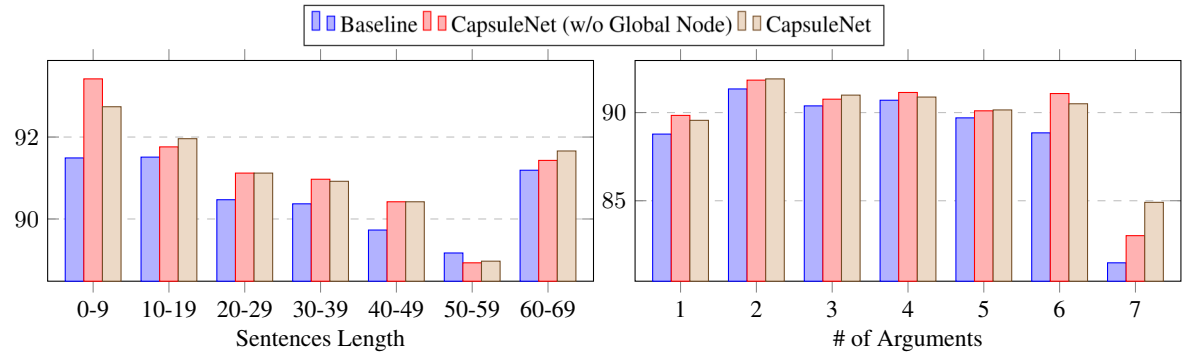


Figure 3: The F1 scores of different models while varying sentence length and argument number on the English test set.

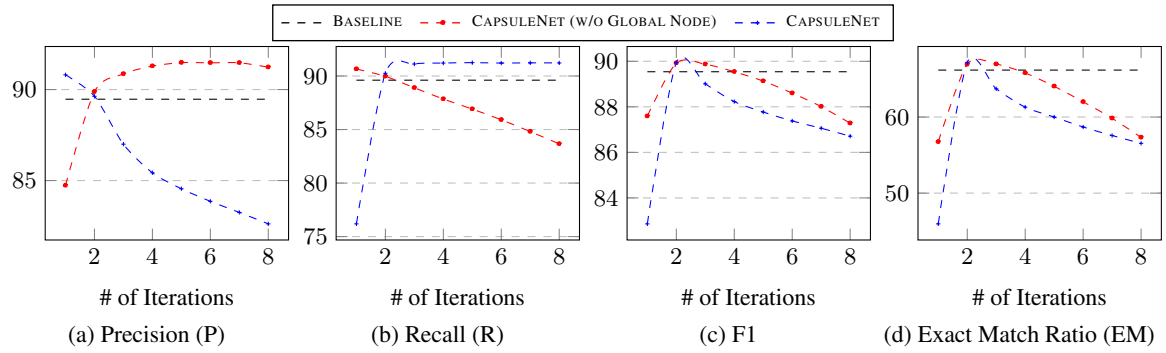


(a) Precision (P) (b) Recall (R) (c) F1 (d) Exact Match Ratio (EM)

Figure 4: The performance of different models while varying the number of refinement iterations on the English development set.

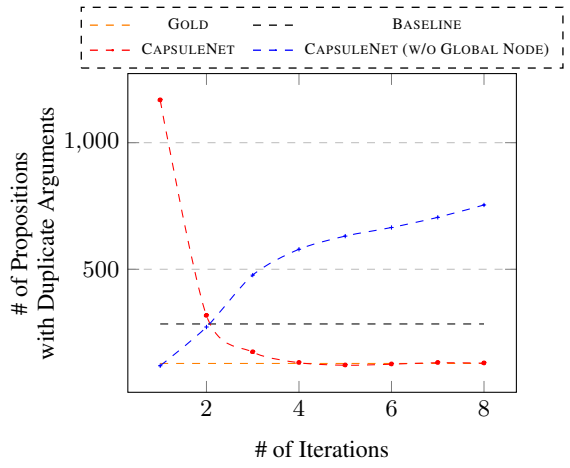


Figure 5: The number of propositions with duplicate arguments for various numbers of refinement iterations on the English development set.

node reduces the precision as the number of iteration grows. This model is less transparent, so it harder to see why it chooses this refinement strategy. As expected, the F1 scores for both models peak at the second iteration, the iteration number used in the training phase. The trend for the exact match score is consistent with the F1 score.

### 7.6.2 Duplicate Arguments

We measure the degree to which the role uniqueness constraint is violated by both models. We plot the number of violations as the function of the number of iterations (Figure 5). Recall that the violations do not imply errors in predictions as even the gold-standard data does not always satisfy the constraint (see the orange line in the figure). As expected, CapsuleNet without the global node captures competition and focuses on enforcing this constraint. Interestingly, it converges to keeping the same fraction of violations as the one observed in the gold standard data. In contrast, the full model increasingly ignores the constraint in later iterations. This is consistent with the over-generation trend evident from the precision and recall plots discussed above.

Figure 6 illustrates how many roles get changed between consecutive iterations of CapsuleNet, with and without the global node. Green indicates how many correct changes have been made, while red shows how many errors have been introduced. Since the number of changes is very large, we represent the non-zero number $q$ in the

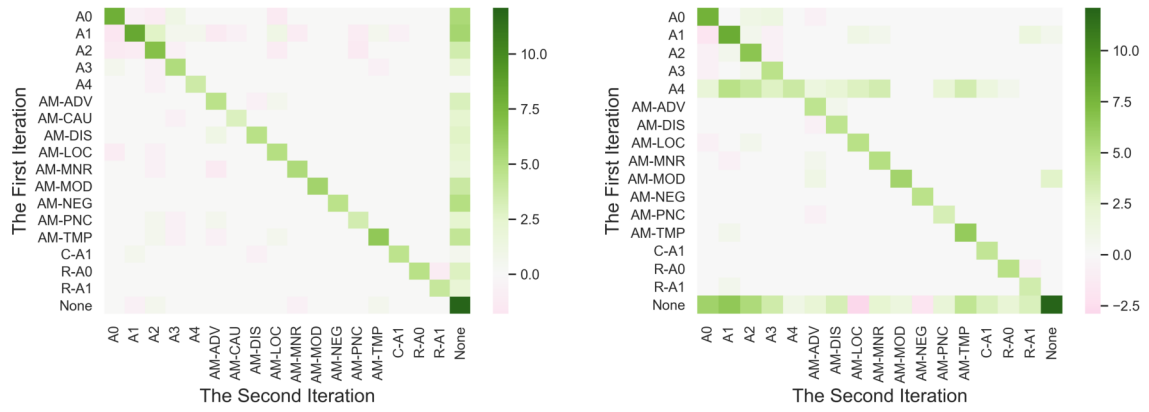


(a) Capsule Network w/o Global Node

(b) Capsule Network with Global Node

Figure 6: Changes in labeling between consecutive iterations on the English development set. Only argument types appearing more than 50 times are listed. "None" type denotes "NOT an argument". Green and red nodes denote the numbers of correct and wrong role transitions have been made respectively. The numbers are transformed into log space.

| Model | Ja | Es | Ca | De | Cz | Zh | En | Avg. |
|---|---|---|---|---|---|---|---|---|
| Previous Best Single Model | 78.69 | 80.50 | 80.32 | **80.10** | 86.02 | **84.30** | 90.40 | 82.90 |
| Baseline Model | 80.12 | 81.00 | 81.39 | 76.01 | 87.79 | 81.05 | 90.49 | 82.55 |
| **CapsuleNet SRL (This Work)** | **81.26** | **81.32** | **81.65** | 76.44 | **88.08** | 81.65 | **91.06** | **83.07** |

Table 5: Labeled F1 score (including senses) for all languages on the CoNLL-2009 in-domain test sets. For previous best result, Japanese (Ja) is from Watanabe et al. (2010), Catalan (Ca) is from Zhao et al. (2009), Spanish (Es) and German (De) are from Roth and Lapata (2016), Czech (Cz) is from Henderson et al. (2013), Chinese (Zh) is from Cai et al. (2018) and English (En) is from Li et al. (2019b).

log space $\tilde{q} = \text{sign}(q)\log(|q|)$.

As we expected, for both models, the majority of changes are correct, leading to better overall performance. We can again see the same trends. CapsuleNet without the global node tends to filter our arguments by changing them to "None". The reverse is true for the full model.

### 7.7 Multilingual Results

Table 5 gives the results of the proposed CapsuleNet SRL (with global node) on the in-domain test sets of all languages from CoNLL-2009. As shown in Table 5, the proposed model consistently outperforms the non-refinement baseline model and achieves state-of-the-art performance on Catalan (Ca), Czech (Cz), English (En), Japanese (Jp) and Spanish (Es). Interestingly, the effectiveness of the refinement method does not seem to be dependent on the dataset size: the improvements on the smallest (Japanese) and the largest datasets (English) are among the largest.

## 8 Additional Related Work

The capsule networks have been recently applied to a number of NLP tasks (Xiao et al., 2018; Gong et al., 2018). In particular, Yang et al. (2018) represent text classification labels by a layer of capsules, and take the capsule actions as the classification probability. Using a similar method, Xia et al. (2018) perform intent detection with the capsule networks. Wang et al. (2018) and Li et al. (2019a) use capsule networks to capture rich features for machine translation. More closely to our work, Zhang et al. (2018) and Zhang et al. (2019) adopt the capsule networks for relation extraction. The previous models apply the capsule networks to problems that have a fixed number of components in the output. Their approach cannot be directly applied to our setting.

## 9 Conclusions & Future Work

State-of-the-art dependency SRL methods do not account for any interaction between role labeling decisions. We propose an iterative approach to

SRL. In each iteration, we refine both predictions of the semantic structure (i.e. a discrete structure) and also a distributed representation of the proposition (i.e. the predicate and its predicted arguments). The iterative refinement process lets the model capture interactions between the decisions. We relied on the capsule networks to operationalize this intuition. We demonstrate that our model is effective, and results in improvements over a strong factorized baseline and state-of-the-art results on standard benchmarks for 5 languages (Catalan, Czech, English, Japanese and Spanish) from CoNLL-2019. In future work, we would like to extend the approach to modeling interaction between multiple predicate-argument structures in a sentence as well as to other semantic formalisms (e.g., abstract meaning representations (Banarescu et al., 2013)).

## Acknowledgments

We thank Diego Marcheggiani, Jonathan Mallinson and Philip Williams for constructive feedback and suggestions, as well as anonymous reviewers for their comments. The project was supported by the European Research Council (ERC StG BroadSem 678254) and the Dutch National Science Foundation (NWO VIDI 639.022.518).

## References

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186.

David Belanger, Bishan Yang, and Andrew McCallum. 2017. End-to-end learning for structured prediction energy networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 429–439.

Jiaxun Cai, Shexia He, Zuchao Li, and Hai Zhao. 2018. A full end-to-end semantic role labeler, syntax-agnostic over syntax-aware? *arXiv preprint arXiv:1808.03815*.

Janara Christensen, Mausam, Stephen Soderland, and Oren Etzioni. 2011. An analysis of open information extraction based on semantic role labeling. In *Proceedings of the Sixth International Conference on Knowledge Capture*, K-CAP '11, pages 113–120, New York, NY, USA. ACM.

Dipanjan Das, André FT Martins, and Noah A Smith. 2012. An exact dual decomposition algorithm for shallow semantic parsing with constraints. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 209–217. Association for Computational Linguistics.

Fabian Eckert and Mariana Neves. 2018. Semantic role labeling tools for biomedical question answering: a study of selected tools on the BioASQ datasets. In *Proceedings of the 6th BioASQ Workshop A challenge on large-scale biomedical semantic indexing and question answering*, pages 11–21, Brussels, Belgium. Association for Computational Linguistics.

Nicholas FitzGerald, Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic role labeling with neural network factors. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 960–970.

William Foland and James Martin. 2015. Dependency-based semantic role labeling using convolutional neural networks. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, pages 279–288.

Jingjing Gong, Xipeng Qiu, Shaojing Wang, and Xuanjing Huang. 2018. Information aggregation via dynamic routing for sequence encoding. *arXiv preprint arXiv:1806.01501*.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the 11th Language Resources and Evaluation Conference*, Miyazaki, Japan. European Language Resource Association.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, CoNLL '09, pages 1–18, Stroudsburg, PA, USA. Association for Computational Linguistics.

Shexia He, Zuchao Li, Hai Zhao, and Hongxiao Bai. 2018. Syntax for semantic role labeling, to be, or not to be. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2061–2071.

James Henderson, Paola Merlo, Ivan Titov, and Gabriele Musillo. 2013. Multilingual joint parsing of syntactic and semantic dependencies with a latent variable model. *Computational Linguistics*, 39(4):949–998.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1173–1182, Brussels, Belgium. Association for Computational Linguistics.

Tao Lei, Yuan Zhang, Lluis Marquez, Alessandro Moschitti, and Regina Barzilay. 2015. High-order low-rank tensors for semantic role labeling. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1150–1160.

Jian Li, Baosong Yang, Zi-Yi Dou, Xing Wang, Michael R Lyu, and Zhaopeng Tu. 2019a. Information aggregation for multi-head attention with routing-by-agreement. *arXiv preprint arXiv:1904.03100*.

Zuchao Li, Shexia He, Jiaxun Cai, Zhuosheng Zhang, Hai Zhao, Gongshen Liu, Linlin Li, and Luo Si. 2018. A unified syntax-aware framework for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2401–2411.

Zuchao Li, Shexia He, Hai Zhao, Yiqing Zhang, Zhuosheng Zhang, Xi Zhou, and Xiang Zhou. 2019b. Dependency or span, end-to-end uniform semantic role labeling. *arXiv preprint arXiv:1901.05280*.

Diego Marcheggiani, Joost Bastings, and Ivan Titov. 2018. Exploiting semantics in neural machine translation with graph convolutional networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 486–492, New Orleans, Louisiana. Association for Computational Linguistics.

Diego Marcheggiani, Anton Frolov, and Ivan Titov. 2017. A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling. *arXiv preprint arXiv:1701.02593*.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. *arXiv preprint arXiv:1703.04826*.

Phoebe Mulcaire, Swabha Swayamdipta, and Noah Smith. 2018. Polyglot semantic role labeling. *arXiv preprint arXiv:1805.11598*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Feng Qian, Lei Sha, Baobao Chang, LuChen Liu, and Ming Zhang. 2017. Syntax aware LSTM model for semantic role labeling. In *Proceedings of the 2nd Workshop on Structured Prediction for Natural Language Processing*, pages 27–32, Copenhagen, Denmark. Association for Computational Linguistics.

Michael Roth and Mirella Lapata. 2016. Neural semantic role labeling with dependency path embeddings. *arXiv preprint arXiv:1605.07515*.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. 2017. Dynamic routing between capsules. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3856–3866. Curran Associates, Inc.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The conll-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, CoNLL '08, pages 159–177, Stroudsburg, PA, USA. Association for Computational Linguistics.

Swabha Swayamdipta, Miguel Ballesteros, Chris Dyer, and Noah A Smith. 2016. Greedy, joint syntactic-semantic parsing with stack lstms. *arXiv preprint arXiv:1606.08954*.

Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Efficient inference and structured learning for semantic role labeling. *Transactions of the Association for Computational Linguistics*, 3:29–41.

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2018. Deep semantic role labeling with self-attention. In *AAAI*.

Mingxuan Wang, Jun Xie, Zhixing Tan, Jinsong Su, et al. 2018. Towards linear time neural machine translation with capsule networks. *arXiv preprint arXiv:1811.00287*.

Yotaro Watanabe, Masayuki Asahara, and Yuji Matsumoto. 2010. A structured model for joint learning of argument roles and predicate senses. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 98–102, Uppsala, Sweden. Association for Computational Linguistics.

Congying Xia, Chenwei Zhang, Xiaohui Yan, Yi Chang, and Philip Yu. 2018. Zero-shot user intent detection via capsule neural networks. In

*Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3090–3099, Brussels, Belgium. Association for Computational Linguistics.

Liqiang Xiao, Honglun Zhang, Wenqing Chen, Yongkun Wang, and Yaohui Jin. 2018. MCapsNet: Capsule network for text with multi-task learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4565–4574, Brussels, Belgium. Association for Computational Linguistics.

Min Yang, Wei Zhao, Jianbo Ye, Zeyang Lei, Zhou Zhao, and Soufei Zhang. 2018. Investigating capsule networks with dynamic routing for text classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3110–3119, Brussels, Belgium. Association for Computational Linguistics.

Ningyu Zhang, Shumin Deng, Zhanling Sun, Xi Chen, Wei Zhang, and Huajun Chen. 2018. Attention-based capsule networks with dynamic routing for relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 986–992, Brussels, Belgium. Association for Computational Linguistics.

Xinsong Zhang, Pengshuai Li, Weimin Jia, and Hai Zhao. 2019. Multi-labeled relation extraction with attentive capsule network. *CoRR*, abs/1811.04354.

Hai Zhao, Wenliang Chen, Jun'ichi Kazama, Kiyotaka Uchimoto, and Kentaro Torisawa. 2009. Multilingual dependency learning: Exploiting rich features for tagging syntactic and semantic dependencies. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, pages 61–66. Association for Computational Linguistics.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1127–1137, Beijing, China. Association for Computational Linguistics.

# Learning Programmatic Idioms for Scalable Semantic Parsing

**Srinivasan Iyer**[†], **Alvin Cheung**[§] **and Luke Zettlemoyer**[†‡]
[†]Paul G. Allen School of Computer Science and Engineering, Univ. of Washington, Seattle, WA
{sviyer, lsz}@cs.washington.edu
§Department of Electrical Engineering and Computer Sciences, UC Berkeley, Berkeley, CA
akcheung@cs.berkeley.edu
‡ Facebook AI Research, Seattle
lsz@fb.com

## Abstract

Programmers typically organize executable source code using high-level coding patterns or idiomatic structures such as nested loops, exception handlers and recursive blocks, rather than as individual code tokens. In contrast, state of the art (SOTA) semantic parsers still map natural language instructions to source code by building the code syntax tree one node at a time. In this paper, we introduce an iterative method to extract code idioms from large source code corpora by repeatedly collapsing most-frequent depth-2 subtrees of their syntax trees, and train semantic parsers to apply these idioms during decoding. Applying idiom-based decoding on a recent context-dependent semantic parsing task improves the SOTA by 2.2% BLEU score while reducing training time by more than 50%. This improved speed enables us to scale up the model by training on an extended training set that is 5× larger, to further move up the SOTA by an additional 2.3% BLEU and 0.9% exact match. Finally, idioms also significantly improve accuracy of semantic parsing to SQL on the ATIS-SQL dataset, when training data is limited.

## 1 Introduction

When programmers translate Natural Language (NL) specifications into executable source code, they typically start with a high-level plan of the major structures required, such as nested loops, conditionals, etc. and then proceed to fill in specific details into these components. We refer to these high-level structures (Figure 1 (b)) as code idioms (Allamanis and Sutton, 2014). In this paper, we demonstrate how learning to use code idioms leads to an improvement in model accuracy and training time for the task of semantic parsing, i.e., mapping intents in NL into general purpose source code (Iyer et al., 2017; Ling et al., 2016).

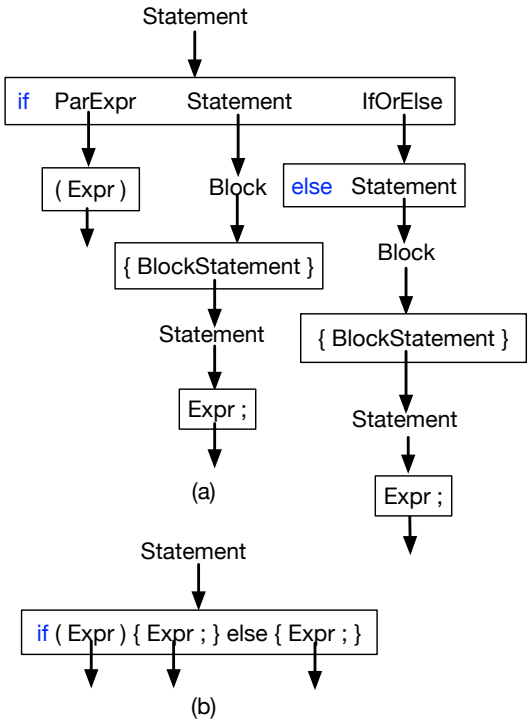


Figure 1: (a) Syntax tree based decoding for semantic parsing uses as many as 11 rules (steps) to produce the outer structure of a very frequently used `if-then-else` code snippet. (b) Direct application of an `if-then-else` idiom during decoding leads to improved accuracy and training time.

State-of-the-art semantic parsers are neural encoder-decoder models, where decoding is guided by the target programming language grammar (Yin and Neubig, 2017; Rabinovich et al., 2017; Iyer et al., 2018) to ensure syntactically valid programs. For general purpose programming languages with large formal grammars, this can easily lead to long decoding paths even for short snippets of code. For example, Figure 1 shows an intermediate parse tree for a generic `if-then-else` code snippet, for which the decoder requires as many as eleven decoding steps



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5426–5435, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

before ultimately filling in the slots for the `if` condition, the `then` expression and the `else` expression. However, the `if-then-else` block can be seen as a higher level structure such as shown in Figure 1 (b) that can be applied in one decoding step and reused in many different programs. In this paper, we refer to frequently recurring subtrees of programmatic parse trees as *code idioms*, and we equip semantic parsers with the ability to learn and directly generate idiomatic structures as in Figure 1 (b).

We introduce a simple iterative method to extract idioms from a dataset of programs by repeatedly collapsing the most frequent depth-2 subtrees of syntax parse trees. Analogous to the byte pair encoding (BPE) method (Gage, 1994; Sennrich et al., 2016) that creates new subtokens of words by repeatedly combining frequently occurring adjacent pairs of subtokens, our method takes a depth-2 syntax subtree and replaces it with a tree of depth-1 by removing all the internal nodes. This method is in contrast with the approach using probabilistic tree substitution grammars (pTSG) taken by Allamanis and Sutton (2014), who use the *explanation quality* of an idiom to prioritize idioms that are more interesting, with an end goal to suggest useful idioms to programmers using IDEs. Once idioms are extracted, we greedily apply them to semantic parsing training sets to provide supervision for learning to apply idioms.

We evaluate our approach on two semantic parsing tasks that map NL into 1) general-purpose source code, and 2) executable SQL queries, respectively. On the first task, i.e., context dependent semantic parsing (Iyer et al., 2018) using the CONCODE dataset, we improve the state of the art (SOTA) by 2.2% of BLEU score. Furthermore, generating source code using idioms results in a more than 50% reduction in the number of decoding steps, which cuts down training time to less than half, from 27 to 13 hours. Taking advantage of this reduced training time, we further push the SOTA on CONCODE to an EM of 13.4 and a BLEU score of 28.9 by training on an extended version of the training set (with 5× the number of training examples). On the second task, i.e., mapping NL utterances into SQL queries for a flight information database (ATIS-SQL; Iyer et al. (2017)), using idioms significantly improves denotational accuracy over SOTA models, when a limited amount of training data is used, and also marginally outperforms the SOTA when the full training set is used (more details in Section 7).

## 2 Related Work

Neural encoder-decoder models have proved effective in mapping NL to logical forms (Dong and Lapata, 2016) and also for directly producing general purpose programs (Iyer et al., 2017, 2018). Ling et al. (2016) use a sequence-to-sequence model with attention and a copy mechanism to generate source code. Instead of directly generating a sequence of code tokens, recent methods focus on constrained decoding mechanisms to generate syntactically correct output using a decoder that is either grammar-aware or has a dynamically determined modular structure paralleling the structure of the abstract syntax tree (AST) of the code (Rabinovich et al., 2017; Krishnamurthy et al., 2017; Yin and Neubig, 2017). Iyer et al. (2018) use a similar decoding approach but use a specialized context encoder for the task of context-dependent code generation. We augment these neural encoder-decoder models with the ability to decode in terms of frequently occurring higher level idiomatic structures to achieve gains in accuracy and training time.

Another different but related method to produce source code is using sketches, which are code snippets containing slots in the place of low-level information such as variable names, method arguments, and literals. Dong and Lapata (2018) generate such sketches using programming language-specific sketch creation rules and use them as intermediate representations to train token-based seq2seq models that convert NL to logical forms. Hayati et al. (2018) retrieve sketches from a large training corpus and modify them for the current input; Murali et al. (2018) use a combination of neural learning and type-guided combinatorial search to convert existing sketches into executable programs, whereas Nye et al. (2019) additionally also generate the sketches before synthesising programs. Our idiom-based decoder learns to produce commonly used subtrees of programming syntax-trees in one decoding step, where the non-terminal leaves function as slots that can be subsequently expanded in a grammer-aware fashion. Code idioms can be roughly viewed as a tree-structured generalization of sketches, that can be automatically extracted from large code corpora for any programming language, and unlike sketches, can

also be nested with other idioms or grammar rules.

More closely related to the idioms that we use for decoding is Allamanis and Sutton (2014), who develop a system (HAGGIS) to automatically mine idioms from large code bases. They focus on finding interesting and explainable idioms, e.g., those that can be included as preset code templates in programming IDEs. Instead, we learn frequently used idioms that can be easily associated with NL phrases in our dataset. The production of large subtrees in a single step directly translates to a large speedup in training and inference.

Concurrent with our research, Shin et al. (2019) also develop a system (PATOIS) for idiom-based semantic parsing and demonstrate its benefits on the Hearthstone (Ling et al., 2016) and Spider (Yu et al., 2018) datasets. While we extract idioms by collapsing frequently occurring depth-2 AST subtrees and apply them greedily during training, they use non-parametric Bayesian inference for idiom extraction and train neural models to either apply entire idioms or generate its full body.

## 3 Idiom Aware Encoder-Decoder Models

We aim to train semantic parsers having the ability to use idioms during code generation. To do this, we first extract frequently used idioms from the training set, and then provide them as supervision to the semantic parser's learning algorithm.

Formally, if a semantic parser decoder is guided by a grammar $\mathscr{G} = (N, \Sigma, R)$, where $N$ and $\Sigma$ are the sets of non-terminals and terminals respectively, and $R$ is the set of production rules of the form $A \rightarrow \beta, A \in N, \beta \in \{N \cup \Sigma\}^*$, we would like to construct an idiom set $I$ with rules of the form $B \rightarrow \gamma, B \in N, \gamma \in \{N \cup \Sigma\}^*$, such that $B \overset{\geq 2}{\Longrightarrow} \gamma$ under $\mathscr{G}$, i.e., $\gamma$ can be derived in two or more steps from $B$ under $\mathscr{G}$. For the example in Figure 1, $R$ would contain rules for expanding each non-terminal, such as *Statement* $\rightarrow$ *if ParExpr Statement IfOrElse* and *ParExpr* $\rightarrow$ { *Expr* }, whereas $I$ would contain the idiomatic rule *Statement* $\rightarrow$ *if ( Expr ) { Expr ; } else { Expr ; }* .

The decoder builds trees from $\hat{\mathscr{G}} = (N, \Sigma, R \cup I)$. Although the set of valid programs under both $\mathscr{G}$ and $\hat{\mathscr{G}}$ are exactly the same, this introduction of ambiguous rules into $\mathscr{G}$ in the form of idioms presents an opportunity to learn shorter derivations. In the next two sections, we describe the idiom extraction process, i.e., how $I$ is chosen, and the idiom application process, i.e., how the decoder is trained to learn to apply idioms.

**Procedure** *Extract-Idioms($\mathscr{D}$, $\mathscr{G}$, $n$)*
  **Input:** $\mathscr{D} \rightarrow$ Training Programs
  **Input:** $\mathscr{G} \rightarrow$ Parsing Grammar
  **Input:** $n \rightarrow$ Number of idioms
  $T \leftarrow \{\}$ ▷ `Stores all parse trees`
  **for** $d \in \mathscr{D}$ **do**
    $T \leftarrow T \cup$ Parse-Tree$(d, \mathscr{G})$
  **end**
  $I \leftarrow \{\}$
  **for** $i \leftarrow 1$ **to** $K$ **do**
    $s \leftarrow$ Most-frequent Depth-2-Subtree$(T)$
    **for** $t \in T$ **do**
      $t \leftarrow$ Collapse-Subtree(t, s)
    **end**
    $I \leftarrow I \cup \{s\}$
  **end**
**end**
**Procedure** *Collapse-Subtree($t$, $s$)*
  **Input:** $t \rightarrow$ parse-tree
  **Input:** $s \rightarrow$ Depth 2 subtree
  **while** $s$ *subtree-of* $t$ **do**
    Replace $s$ in $t$ with Collapse$(s)$
  **end**
**end**
**Procedure** *Collapse($s$)*
  **Input:** $s \rightarrow$ Depth 2 subtree
  frontier $\leftarrow$ leaves$(s)$
  return Tree($root(s)$,frontier)
**end**

**Algorithm 1:** Idiom Extraction.

## 4 Idiom Extraction

Algorithm 1 describes the procedure to add idiomatic rules, $I$, to the regular production rules, $R$. Our goal is to populate the set $I$ by identifying frequently occurring idioms (subtrees) from the programs in training set $\mathscr{D}$. Since enumerating all subtrees of every AST in the training set is infeasible, we observe that all subtrees $s'$ of a frequently occurring subtree $s$ are just as or more frequent than $s$, so we take a bottom-up approach by repeatedly collapsing the most frequent depth-2 subtrees. Intuitively, this can be viewed as a particular kind of generalization of the BPE (Gage, 1994; Sennrich et al., 2016) algorithm for sequences, where new subtokens are created by repeatedly combining frequently occurring adjacent pairs of subtokens. Note that subtrees of parse

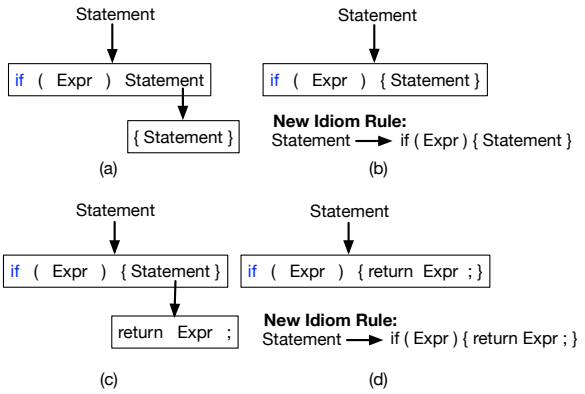


Figure 2: Two steps of the idiom extraction process described in Algorithm 1. (a) First, we find the most frequent depth-2 syntax subtree under Grammar $\mathscr{G}$ in dataset $\mathscr{D}$, collapse it to produce a new production rule (b), and replace all occurrences in the dataset with the collapsed version. Next, (c) is the most frequent depth-2 subtree which now happens to contain (b) within it. (c) is collapsed to form an idiom (d), which is effectively a depth-3 idiom.

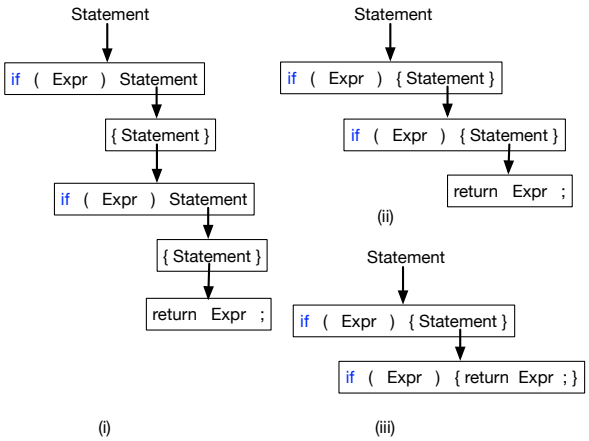


Figure 3: (i) Application of the two idioms extracted in Figure 2 on a new training example. (ii) We first perform two applications of the idiom in Figure 2b, (iii) followed by an application of the idiom in Figure 2d. The resulting tree can be represented with just 2 parsing rules instead of 5.

trees have an additional constraint, i.e., either all or none of the children of non-terminal nodes are included, since a grammar rule has to be used entirely or not at all.

We perform idiom extraction in an iterative fashion. We first populate $T$ with all parse trees of programs in $\mathscr{D}$ using grammar $\mathscr{G}$ (Step 4). Each iteration then comprises retrieving the most frequent depth-2 subtree $s$ from $T$ (Step 8), followed by post-processing $T$ to replace all occurrences of $s$ in $T$ with a collapsed (depth-1) version of $s$ (Step 10 and 17). The collapse function (Step 20) simply takes a subtree, removes all its internal nodes and attaches its leaves directly to its root (Step 22). The collapsed version of $s$ is a new idiomatic rule (a depth-1 subtree), which we add to our set of idioms, $I$ (Step 12). We illustrate two iterations of this algorithm in Figure 2 ((a)-(b) and (c)-(d)). Assuming (a) is the most frequent depth-2 subtree in the dataset, it is transformed into the idiomatic rule in (b). Larger idiomatic trees are learned by combining several depth-2 subtrees as the algorithm progresses. This is shown in Figure 2 (c) which contains the idiom extracted in (b) within it owing to the post-processing of the dataset after idiom (b) is extracted (Step 10 of Algorithm 1) which effectively makes the idiom in (d), a depth-3 idiom. We perform idiom extraction for $K$ iterations. In our experiments we vary the value of $K$ based on the number of idioms we would like to extract.

## 5 Model Training with Idioms

Once a set of idioms $I$ is obtained, we next train semantic parsing models to apply these idioms while decoding. We do this by supervising grammar rule generation in the decoder using a compressed set of rules for each example, using the idiom set $I$ (see Algorithm 2). More concretely, we first obtain the parse tree $t_i$ (or grammar rule set $p_i$) for each training program $y_i$ under grammar $\mathscr{G}$ (Step 3) and then greedily collapse each depth-2 subtree in $t_i$ corresponding to every idiom in $I$ (Step 5). Once $t_i$ cannot be further collapsed, we translate $t_i$ into production rules $r_i$ based on the collapsed tree, with $|r_i| \leq |p_i|$ (Step 7).

1 **Procedure** *Compress*($\mathscr{D}, \mathscr{G}, I$)
  **Input:** $\mathscr{D} \rightarrow$ Training Programs
  **Input:** $\mathscr{G} \rightarrow$ Parsing Grammar
  **Input:** $I \rightarrow$ Idiom set
2  **for** $d \in \mathscr{D}$ **do**
3  $t \leftarrow$ Parse-Tree$(d, \mathscr{G})$
4  **for** $i \in I$ **do**
5  $t \leftarrow$ Collapse-Subtree(t, i)
6  **end**
7  Train decoder using
  Production-Rules$(t)$
8  **end**
9 **end**

**Algorithm 2:** Training Example Compression.

This process is illustrated in Figure 3 where we

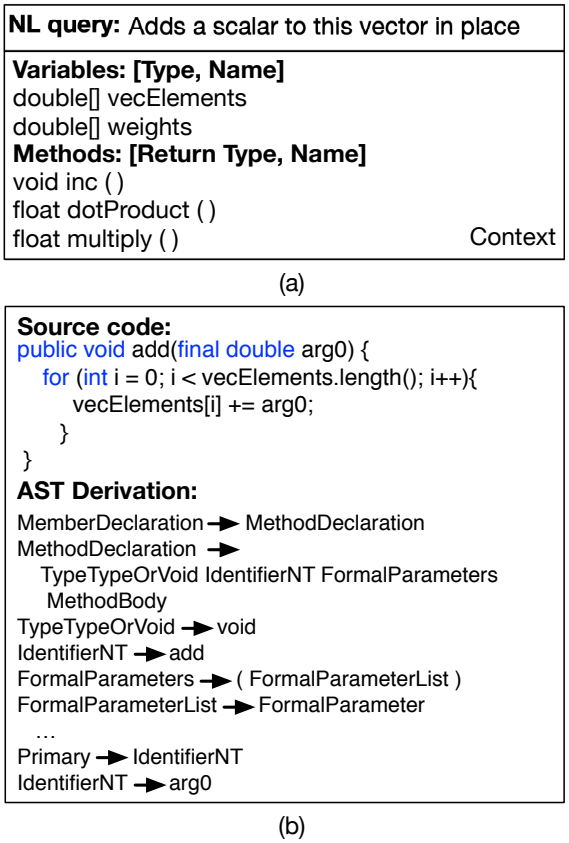


Figure 4: Context dependent code generation task of Iyer et al. (2018) that involves mapping an NL query together with a set of context variables and methods (each having an identifier name and data type) (a) into source code, represented as a sequence of grammar rules (b).

perform two applications of the first idiom from Figure 2 (b), followed by one application of the second idiom from Figure 2 (d), after which the tree cannot be further compressed using those two idioms. The final tree can be represented using $|r_i| = 2$ rules instead of the original $|p_i| = 5$ rules. The decoder is then trained similar to previous approaches (Yin and Neubig, 2017) using the compressed set of rules. We observe a rule set compression of more than 50% in our experiments (Section 7).

## 6 Experimental Setup

We apply our approach on 1) the context dependent encoder-decoder model of Iyer et al. (2018) on the CONCODE dataset, where we outperform an improved version of their best model, and 2) the task of mapping NL utterances to SQL queries on the ATIS-SQL dataset (Iyer et al., 2017) where an idiom-based model using the full training set outperforms the SOTA, also achieving significant

```
List all flights from Denver to Seattle

SELECT DISTINCT flight_1.flight_id FROM
 flight f1, airport_service as1, city c1,
 airport_service as2, city c2 WHERE
 f1.from_airport = as1.airport_code
 AND as1.city_code = c1.city_code
 AND c1.city_name = "Denver"
 AND f1.to_airport = as2.airport_code
 AND as2.city_code = c2.city_code
 AND c2.city_name = "Seattle";
```

Figure 5: Example NL utterance with its corresponding executable SQL query from the ATIS-SQL dataset.

gains when using a reduced training set.

### 6.1 Context Dependent Semantic Parsing

The CONCODE task involves mapping an NL query together with a class context comprising a list of variables (with types) and methods (with return types), into the source code of a class member function. Figure 4 (a) shows an example where the context comprises variables and methods (with types) that would normally exist in a class that implements a vector, such as `vecElements` and `dotProduct()`. Conditioned on this context, the task involves mapping the NL query *Adds a scalar to this vector in place* into a sequence of parsing rules to generate the source code in Figure 4 (b).

Formally, the task is: Given a NL utterance $q$, a set of context variables $\{v_i\}$ with types $\{t_i\}$, and a set of context methods $\{m_i\}$ with return types $\{r_i\}$, predict a set of parsing rules $\{a_i\}$ of the target program. Their best performing model is a neural encoder-decoder with a context-aware encoder and a decoder that produces a sequence of Java grammar rules.

**Baseline Model** We follow the approach of Iyer et al. (2018) with three major modifications in their encoder, which yields improvements in both speed and accuracy (Iyer-Simp). First, in addition to camel-case splitting of identifier tokens, we use byte-pair encoding (BPE) (Sennrich et al., 2016) on all NL tokens, identifier names and types and embed all these BPE tokens using a single embedding matrix. Next, we replace their RNN that contextualizes the subtokens of identifiers and types with an average of the subtoken embeddings instead. Finally, we consolidate their three separate RNNs for contextualizing NL, variable names with types, and method names with types, into a single shared RNN, which greatly reduces the number of model parameters. Formally, let

$\{q_i\}$ represent the set of BPE tokens of the NL, and $\{t_{ij}\}$, $\{v_{ij}\}, \{r_{ij}\}$ and $\{m_{ij}\}$ represent the $j$th BPE token of the $i$th variable type, variable name, method return type, and method name respectively. First, all these elements are embedded using a BPE token embedding matrix B to give us $\mathbf{q_i}$, $\mathbf{t_{ij}}$, $\mathbf{v_{ij}}$, $\mathbf{r_{ij}}$ and $\mathbf{m_{ij}}$. Using Bi-LSTM $f$, the encoder then computes:

$$h_1, \cdots, h_z = f(\mathbf{q_1}, \ldots, \mathbf{q_z}) \quad (1)$$

$$\mathbf{v_i} = Avg(\mathbf{v_{i1}}, \ldots, \mathbf{v_{ij}}) \quad (2)$$

$$\text{Similarly, compute } \mathbf{m_i}, \mathbf{t_i}, \mathbf{r_i} \quad (3)$$

$$\hat{t_i}, \hat{v_i} = f(\mathbf{t_i}, \mathbf{v_i}) \quad (4)$$

$$\hat{r_i}, \hat{m_i} = f(\mathbf{r_i}, \mathbf{m_i}) \quad (5)$$

Then, $h_1, \ldots, h_z$, and $\hat{t_i}, \hat{v_i}, \hat{r_i}, \hat{m_i}$ are passed to the attention mechanism in the decoder, exactly as in Iyer et al. (2018). The decoder remains the same as described in Iyer et al. (2018), and produces a probability distribution over grammar rules at each time step (full details in Supplementary Materials). This forms our baseline model (Iyer-Simp).

**Idiom Aware Training** To utilize idioms, we augment this decoder by retrieving the top-K most frequent idioms from the training set (Algorithm 1), followed by post-processing the training set by greedily applying these idioms (Algorithm 2; we denote this model as Iyer-Simp-K). We evaluate all our models on the CONCODE dataset which was created using Java source files from `github.com`. It contains 100K tuples of (NL, code, context) for training, 2K tuples for development, and 2K tuples for testing. We use a BPE vocabulary of 10K tokens (for matrix $B$) and get the best validation set results using the original hyperparameters used by Iyer et al. (2018). Since idiom aware training is significantly faster than without idioms, it enables us to train on an additional 400K training examples that Iyer et al. (2018) released as part of CONCODE. We report exact match accuracy, corpus level BLEU score (which serves as a measure of partial credit) (Papineni et al., 2002), and training time for all these configurations.

### 6.2 Semantic Parsing to SQL

This task involves mapping NL utterances into executable SQL queries. We use the ATIS-SQL dataset (Iyer et al., 2017) that contains NL questions posed to a flight database, together with their SQL queries and a database with 25 tables to execute them against (Figure 5 shows an example). The dataset is split into 4,379 training, 491 validation, and 448 testing examples following Kwiatkowski et al. (2011).

The SOTA by Iyer et al. (2017) is a Seq2Seq model with attention and achieves a denotational accuracy of 82.5% on the test set. Since using our idiom-based approach requires a model that uses grammar-rule based decoding, we use a modified version of the Seq2Prod model described in Iyer et al. (2018) (based on Yin and Neubig (2017)) as a baseline model (Seq2Prod), and augment the decoder with SQL idioms (Seq2Prod-K).

Seq2Prod is an encoder-decoder model, where the encoder executes an n-layer bi-LSTM over NL embeddings and passes the final layer LSTM

| Model | Exact | BLEU |
|---|---|---|
| Seq2Seq† | 3.2 (2.9) | 23.5 (21.0) |
| Seq2Prod† | 6.7 (5.6) | 21.3 (20.6) |
| Iyer et al. (2018)† | 8.6 (7.1) | 22.1 (21.3) |
| Iyer-Simp | 12.5 (9.8) | 24.4 (23.2) |
| Iyer-Simp + 200 idioms | 12.2 (9.8) | **26.6 (24.0)** |

Table 1: Exact Match and BLEU scores for our simplified model (Iyer-Simp) with and without idioms, compared with results from Iyer et al. (2018)† on the test (validation) set of CONCODE. Iyer-Simp achieves significantly better EM and BLEU score and reduces training time from 40 hours to 27 hours. Augmenting the decoding process with 200 code idioms further pushes up BLEU and reduces training time to 13 hours.

| Model | Exact | BLEU | Training Time (h) |
|---|---|---|---|
| Seq2Seq† | 2.9 | 21.0 | 12 |
| Seq2Prod† | 5.6 | 20.6 | 36 |
| Iyer et al. (2018)† | 7.1 | 21.3 | 40 |
| Iyer-Simp | 9.8 | 23.2 | 27 |
| + 100 idioms | 9.8 | 24.5 | 15 |
| + 200 idioms | 9.8 | 24.0 | 13 |
| + 300 idioms | 9.6 | 23.8 | 12 |
| + 400 idioms | 9.7 | 23.8 | 11 |
| + 600 idioms | 9.9 | 22.7 | 11 |

Table 2: Variation in Exact Match, BLEU score, and training time on the validation set of CONCODE with number of idioms used. After top-200 idioms are used, accuracies start to reduce, since using more specialized idioms can hurt model generalization. Training time plateaus after considering top-600 idioms.

| Model | Exact | BLEU |
|---|---|---|
| 1× Train | 12.0 (9.7) | 26.3 (23.8) |
| 2× Train | 13.0 (10.3) | 28.4 (25.2) |
| 3× Train | 13.3 (10.4) | 28.6 (26.5) |
| 5× Train | 13.4 (11.0) | 28.9 (26.6) |

Table 3: Exact Match and BLEU scores on the test (validation) set of CONCODE by training Iyer-Simp-400 on the extended training set released by Iyer et al. (2018). Significant improvements in training speed after incorporating idioms makes training on large amounts of data possible.

hidden states to an attention mechanism in the decoder. Note that the Seq2Prod encoder described in Iyer et al. (2018) encodes a concatenated sequence of NL and context, but ATIS-SQL instances do not include contextual information. Thus, if $q_i$ represents each lemmatized token of the NL, they are first embedded using a token embedding matrix $B$ to give us $\mathbf{q_i}$. Using Bi-LSTM $f$, the encoder then computes:

$$h_1, \cdots, h_z = f(\mathbf{q_1}, \ldots, \mathbf{q_z}) \quad (6)$$

Then, $h_1, \ldots, h_z$ are passed to the attention mechanism in the decoder.

The sequential LSTM-decoder uses attention and produces a sequence of grammar rules $\{a_t\}$. The decoder hidden state at time $t$, $s_t$, is computed based on an embedding of the current non-terminal $n_t$ to be expanded, an embedding of the previous production rule $a_{t-1}$, an embedding of the parent production rule, par$(n_t)$, that produced $n_t$, the previous decoder state $s_{t-1}$, and the decoder state of the LSTM cell that produced $n_t$, denoted as $s_{n_t}$.

$$s_t = \text{LSTM}_f(n_t, a_{t-1}, \text{par}(n_t), s_{t-1}, s_{n_t}) \quad (7)$$

$s_t$ is then used for attention and finally, produces a distribution over grammar rules.

We make two modifications in this decoder. First, we remove the dependence of LSTM$_f$ on the parent LSTM cell state $s_{n_t}$. Second, instead of using direct embeddings of rules $a_{t-1}$ and par$(n_t)$ in LSTM$_f$, we use another Bi-LSTM across the left and right sides of the rule (using separator symbol SEP) and use the final hidden state as inputs to LSTM$_f$ instead. More concretely, if a grammar rule is represented as $A \to B_1 \ldots B_n$, then:

$$\begin{aligned} &\text{Emb}(A \to B_1 \ldots B_n) = \\ &\qquad \text{LSTM}_g(\{A, \text{SEP}, B_1, \ldots, B_n\}) \\ &s_t = \text{LSTM}_f(n_t, \text{Emb}(a_{t-1}), \text{Emb}(\text{par}(n_t)), \\ &\qquad\qquad s_{t-1}) \end{aligned} \quad (8)$$

This modification can help the LSTM$_f$ cell locate the position of $n_t$ within rules $a_{t-1}$ and par$(n_t)$, especially for lengthy idiomatic rules. We present a full description of this model with all hyperparameters in the supplementary materials.

**Idiom Aware Training** As before, we augment the set of decoder grammar rules with top-K idioms extracted from ATIS-SQL. To represent SQL queries as grammar rules, we use the python `sqlparse` package.

| Model | Accuracy |
|---|---|
| Iyer et al. (2017)† | 82.5 |
| Seq2Prod | 79.1 |
| **Seq2Prod + 400 idioms** | **83.2** |

Table 4: Denotational Accuracy for Seq2Prod with and without idioms, compared with results from Iyer et al. (2017)† on the test set of ATIS using SQL queries. Results averaged over 3 runs.

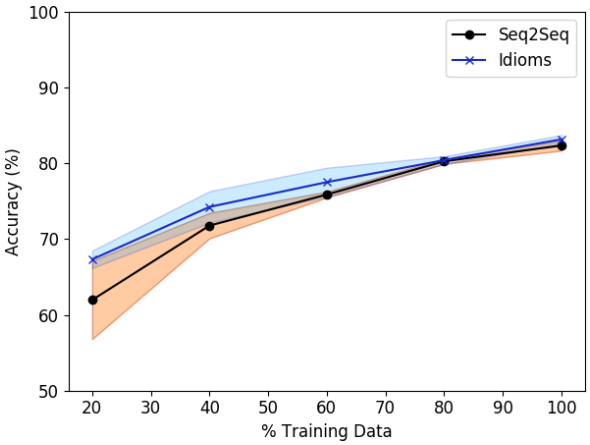


Figure 6: Accuracy of the Seq2Seq baseline (Iyer et al., 2017) compared with Seq2Prod-K, on the test set of ATIS-SQL for varying fractions of training data. Idioms significantly boost accuracy when training data is scarce. (Mean and Std. Dev. computed across 3 runs). 20-40% use 100 idioms and the rest use 400 idioms.

## 7 Results and Discussion

Table 1 presents exact match and BLEU scores on the original CONCODE train/validation/test split.

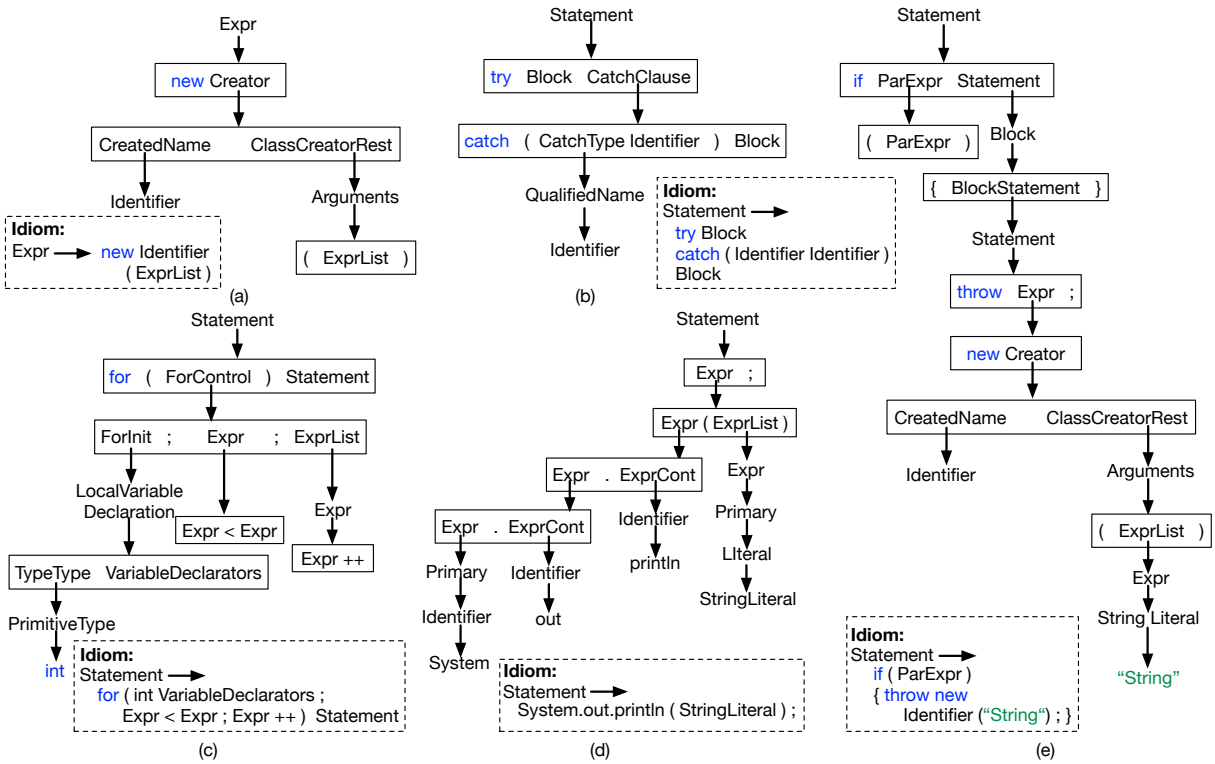


Figure 7: Examples of idioms learned from CONCODE. (a)-(c) represent idioms for instantiation of a new object, exception handling and integer-based looping respectively. (d) represents an idiom for applying a very commonly used library method (`System.out.println`). (e) is a combination of various idioms viz. an if-then idiom, an idiom for throwing exceptions and finally, reusing idiom (a) for creating new objects.

Iyer-Simp yields a large improvement of 3.9 EM and 2.2 BLEU over the best model of Iyer et al. (2018), while also being significantly faster (27 hours for 30 training epochs as compared to 40 hours). Using a reduced BPE vocabulary makes the model memory efficient, which allows us to use a larger batch size that in turn speeds up training. Furthermore, using 200 code idioms further improves BLEU by 2.2% while maintaining comparable EM accuracy. Using the top-200 idioms results in a target AST compression of more than 50%, which results in fewer decoder RNN steps being performed. This reduces training time further by more than 50%, from 27 hours to 13 hours.

In Table 2, we illustrate the variations in EM, BLEU and training time with the number of idioms. We find that 200 idioms performs best overall in terms of balancing accuracy and training time. Adding more idioms continues to reduce training time, but accuracy also suffers. Since we permit idioms to contain identifier names to capture frequently used library methods, having too many idioms hurts generalization, especially since the test set is built using repositories disjoint from the training set. Finally, the amount of compression, and therefore the training time, plateaus after the top-600 idioms are incorporated.

Compared to the model of Iyer et al. (2018), our significantly reduced training time enables us to train on their extended training set. We run Iyer-Simp using 400 idioms (taking advantage of even lower training time) on up to 5 times the amount of data, while making sure that we do not include in training any NL from the validation or the test sets. Since the original set of idioms learned from the original training set are quite general, we directly use them rather than relearn the idioms from scratch. We report EM and BLEU scores for different amounts of training data on the same validation and test sets as CONCODE in Table 3. In general, accuracies increase with the amount of data with the best model achieving a BLEU score of 28.9 and EM of 13.4.

Figure 7 shows example idioms extracted from CONCODE: (a) is an idiom to construct a new object with arguments, (b) represents a `try-catch` block, and, (c) is an integer-based `for` loop. In (e), we show how small idioms are combined to

form larger ones; it combines an `if-then` idiom with a `throw-exception` idiom, which throws an object instantiated using idiom (a). The decoder also learns idioms to directly generate common library methods such as `System.out.println( StringLiteral )` in one decoding step (d).

For the NL to SQL task, we report denotational accuracy in Table 4. We observe that Seq2Prod underperforms the Seq2Seq model of Iyer et al. (2017), most likely because a SQL query parse is much longer than the original query. This is remedied by using top-400 idioms, which compresses the decoded sequence size, marginally outperforming the SOTA (83.2%). Finegan-Dollak et al. (2018) observed that the SQL structures in ATIS-SQL are repeated numerous times in both train and test sets, thus facilitating Seq2seq models to memorize these structures without explicit idiom supervision. To test a scenario with limited repetition of structures, we compare Seq2Seq with Seq2Prod-K for limited training data (increments of 20%) and observe that (Figure 6) idioms are additionally helpful with lesser training data, consistent with our intuition.

## 8 Conclusions

We presented a general approach to make semantic parsers aware of target idiomatic structures, by first identifying frequently used idioms, followed by providing models with supervision to apply these idioms. We demonstrated this approach on the task of context dependent code generation where we achieved a new SOTA in EM accuracy and BLEU score. We also found that decoding using idioms significantly reduces training time and allows us to train on significantly larger datasets. Finally, our approach also outperformed the SOTA for a semantic parsing to SQL task on ATIS-SQL, with significant improvements under a limited training data regime.

## Acknowledgements

University of Washington's efforts in this research was supported in part by DARPA (FA8750-16-2-0032), the ARO (ARO-W911NF-16-1-0121), the NSF (IIS-1252835, IIS-1562364, IIS-1546083, IIS-1651489, OAC-1739419), the DOE (DE-SC0016260), the Intel-NSF CAPA center, and gifts from Adobe, Amazon, Google, Huawei, and NVIDIA. The authors thank the anonymous reviewers for their helpful comments.

## References

Miltiadis Allamanis and Charles Sutton. 2014. Mining idioms from source code. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 472–483. ACM.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43. Association for Computational Linguistics.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 731–742.

Catherine Finegan-Dollak, Jonathan K Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving text-to-sql evaluation methodology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 351–360.

Philip Gage. 1994. A new algorithm for data compression. *The C Users Journal*, 12(2):23–38.

Shirley Anugrah Hayati, Raphael Olivier, Pravalika Avvaru, Pengcheng Yin, Anthony Tomasic, and Graham Neubig. 2018. Retrieval-based neural code generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 925–930. Association for Computational Linguistics.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 963–973, Vancouver, Canada. Association for Computational Linguistics.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. 2018. Mapping language to code in programmatic context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1643–1652. Association for Computational Linguistics.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1516–1526, Copenhagen, Denmark. Association for Computational Linguistics.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical generalization in CCG grammar induction for semantic parsing. In *Proceedings of the 2011 Conference on*

*Empirical Methods in Natural Language Processing*, pages 1512–1523. Association for Computational Linguistics.

Wang Ling, Phil Blunsom, Edward Grefenstette, Moritz Karl Hermann, Tomáš Kočiský, Fumin Wang, and Andrew Senior. 2016. Latent predictor networks for code generation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 599–609. Association for Computational Linguistics.

Vijayaraghavan Murali, Letao Qi, Swarat Chaudhuri, and Chris Jermaine. 2018. Neural sketch learning for conditional program generation.

Maxwell Nye, Luke Hewitt, Joshua Tenenbaum, and Armando Solar-Lezama. 2019. Learning to infer program sketches. *arXiv preprint arXiv:1902.06349*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318.

Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract syntax networks for code generation and semantic parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1139–1149, Vancouver, Canada. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Richard Shin, Miltiadis Allamanis, Marc Brockschmidt, and Oleksandr Polozov. 2019. Program synthesis and semantic parsing with learned code idioms. *arXiv preprint arXiv:1906.10816*.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 440–450, Vancouver, Canada. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921.

# JuICe: A Large Scale Distantly Supervised Dataset for Open Domain Context-based Code Generation

**Rajas Agashe, Srinivasan Iyer and Luke Zettlemoyer**
Paul G. Allen School of Computer Science and Engineering, Univ. of Washington, Seattle, WA
{rajas, sviyer, lsz}@cs.washington.edu

## Abstract

Interactive programming with interleaved code snippet cells and natural language markdown is recently gaining popularity in the form of Jupyter notebooks, which accelerate prototyping and collaboration. To study code generation conditioned on a long context history, we present JuICe, a corpus of 1.5 million examples with a curated test set of 3.7K instances based on online programming assignments. Compared with existing contextual code generation datasets, JuICe provides refined human-curated data, open-domain code, and an order of magnitude more training data. Using JuICe, we train models for two tasks: (1) generation of the API call sequence in a code cell, and (2) full code cell generation, both conditioned on the NL-Code history up to a particular code cell. Experiments using current baseline code generation models show that both context and distant supervision aid in generation, and that the dataset is challenging for current systems.

## 1 Introduction

Interactive computing (IC) is a software engidneering paradigm where programmers write source code scripts in an incremental fashion, one block at a time, taking decisions based on the output of execution of previously written blocks. An increasingly adopted platform for IC is the Jupyter notebook (Kluyver et al., 2016), which additionally encourages the use of markdown in natural language (NL) between code snippets as a means of documentation (Figure 1). Under the framework of IC, models that automatically generate future code blocks based on NL instructions, have the potential to provide significant assistance and speed up development. In this paper, we collect and release a new large-scale dataset based off of




Figure 1: This Python Jupyter notebook, comprising of interleaved NL markdown and code cells, loads data from a file and trains a decision tree classifier. We aim to generate the target code cell (blue) based on the previous NL markdown *Create and train the model* and the prior NL and code history. To the left of each cell, *d* represents its distance from the target cell.

Jupyter notebooks named JuICe (for Jupyter Interactive Computing). We also present experiments to demonstrate baseline performance for the task of incremental context-based code generation.

Figure 1 presents part of a Jupyter notebook for training a decision tree classifier using Python, which includes a title, followed by interleaved NL markdown and code cells. The markdown typically describes the goal of the code cells that follow and often is of high quality, since notebooks are frequently used for sharing and collaboration amongst teams. The first two code cells import packages and perform data loading, and following this, the user would typically inspect the data and proceed to train the model by first writing the markdown as *Create and train the model*. Our goal is to train models that assist the user at this particular stage i.e. to generate the API calls `DecisionTreeClassifier()` and `fit()` and the source code contained in the blue cell in Figure 1 (which trains a Decision tree clas-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5436–5446, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

sifier), based on the previous markdown, and NL and code context. The closest NL utterance alone does not specify that a Decision Tree model must be trained, or which variables should be used as train and test data. Automatically making these decisions conditioned on prior history is one of the main challenges of this task.

Existing tasks for mapping NL to source code primarily use a single NL utterance (Zettlemoyer and Collins, 2005; Iyer et al., 2017) to generate database queries (semantic parsing), single line python code (Yin et al., 2018; Oda et al., 2015), multi-line domain-specific code (Ling et al., 2016; Rabinovich et al., 2017), or sequences of API calls (Gu et al., 2016b). A recent task by Iyer et al. (2018) on the CONCODE dataset maps a single utterance to an entire method, conditioned on environment variables and methods. In contrast, we tackle the task of general purpose code generation in an interactive setting, using an entire sequence of prior NL and code blocks as context. More closely related to our task is the context dependent semantic parsing task on the ATIS dataset (Zettlemoyer and Collins, 2009; Suhr et al., 2018) for mapping NL to database queries based on a prior history of NL and query pairs. One main difference is that while future queries can be built by modifying queries in previous utterances, code snippets in our task are almost always disjoint from previously generated code.

We collect an open-domain large-scale dataset (JuICe) of over 659K publicly available Jupyter notebooks from `github.com` to train models for our task. Each notebook contains an average of 39 NL/code cells and uses python packages from thousands of domains. Since obtaining source-code with NL annotations is expensive, most large NL-code datasets rely on noisy automatically scraped data from source code forums or repositories online (Yin et al., 2018; Iyer et al., 2018). However, this also results in excessively noisy test sets and thus, a persistent challenge has been the creation of high quality code generation test sets. For our dataset, we capitalize on the availability of nbgrader (Hamrick, 2016) notebooks, which are high-quality Jupyter notebooks that are manually created by instructors as programming assignments for classes. These notebooks contain accurate and detailed sequences of NL and code blocks, with blank code-cells intended to be filled in by students and later evaluated. We collect and release 13,905 nbgrader notebooks to evaluate models for code generation tasks, which, together with the training set, forms the first dataset for general purpose code generation in an interactive setting.

We define two code generation related tasks using JuICe: (1) generating API sequences, and (2) full code generation. The first task is more relaxed and aims to assist users by generating a sequence of all function calls (Gu et al., 2016b) needed to achieve the goal of the cell. The second task measures the ability of models to learn to generate fully functioning complete code snippets. For both tasks, we investigate strong neural baselines and study performance improvements obtained by varying the size of prior NL and code context considered, as well as varying training data size. In summary, we introduce the task of code generation under the paradigm of IC. We collect and release a new large-scale dataset together with a manually curated test set based on nbgrader. Finally, we evaluate strong neural baselines on two code generation tasks, which achieve reasonable performance, with significant room for improvement.

## 2 Related Work

There is significant existing research on mapping single NL utterances directly to executable programs in the form of logical forms (Zettlemoyer and Collins, 2005), $\lambda$-DCS (Liang et al., 2013), regular expressions (Kushman and Barzilay, 2013; Locascio et al., 2016), database queries (Iyer et al., 2017; Zhong et al., 2017) and general purpose programs (Balog et al., 2016; Allamanis et al., 2015). Ling et al. (2016) generate Java and Python source code from NL for card games, conditioned on categorical card attributes. Iyer et al. (2018) generate Java class methods using a single NL instruction and conditioned on a list of class environment variables and methods. Yin et al. (2018) mine a large NL-code dataset from Stackoverflow to train models to map a NL programming question into a short example solution snippet. Gu et al. (2016c) use neural models to map Java documentation strings to a sequence of API calls. In this work, we introduce a new task of mapping NL to source code under an interactive programming paradigm i.e., conditioned on a sequence of past NL and code cells representing past exploratory interactions.

Models for mapping NL to code have been evaluated on datasets containing templated code for

| Dataset | Context Based | Open Domain | Large Scale | Curated Dev/Test |
|---|---|---|---|---|
| **JuICe** | ✓ | ✓ | ✓ | ✓ |
| CONCODE (Iyer et al., 2018) | ✓ | ✓ | ✓ | |
| CoNaLa (Yin et al., 2018) | | ✓ | ✓ | ✓ |
| ATIS (Zettlemoyer and Collins, 2009) | ✓ | | | ✓ |
| SequentialQA (Iyyer et al., 2017) | ✓ | | | ✓ |
| SCONE (Long et al., 2016) | ✓ | | | ✓ |

Table 1: Comparison of JuICe with various recently released NL-code datasets along four dimensions. Most datasets that condition on historical natural language and code are small and domain specific, while large scale datasets only use human curated test sets for simple single line NL-code generation.

card games (Hearthstone & MTG; Ling et al., 2016), or manually labeled per-line comments (DJANGO; Oda et al., 2015). These datasets contain ∼20,000 programs with short textual descriptions possibly paired with categorical data and are highly domain specific with limited context. In this work, we collect the first large scale dataset containing over 1.5M source code cells paired with a sequence of NL and code cells that represents past exploratory interactions. Although various large scale datasets (Allamanis and Sutton, 2013, 2014; Allamanis et al., 2016; Iyer et al., 2018) to study code generation have been created from Github, their development and test set are randomly created from the same dataset since human curation is prohibitively expensive. Similarly, Yin et al. (2018) collect a large dataset from Stackoverflow.com (CoNaLa) for training, but only manage to curate a small portion (∼ 2,900 examples) of single line NL and code snippets for evaluation. We take advantage of nbgrader assignment notebooks to create an inexpensive high-quality human-curated test set of 3,725 NL statements with interactive history.

Neural encoder-decoder models have proved effective in mapping NL to logical forms and also for directly producing general purpose programs. Ling et al. (2016) use a sequence-to-sequence model with attention and a copy mechanism (Gu et al., 2016a) to generate source code. Recent methods focus on constrained decoding mechanisms to generate syntactically correct output using a decoder that is either grammar-aware or has a dynamically-determined modular structure paralleling the structure of the abstract syntax tree of the code (Dong and Lapata, 2016; Rabinovich et al., 2017; Krishnamurthy et al., 2017; Yin and Neubig, 2017; Iyer et al., 2018). Iyer et al. (2018) use a specialized context encoder that uses sub-word units (Sennrich et al., 2016) for representing code tokens and a grammar-aware decoder that attends to both NL and context to produce source code parses. Although syntax aware models are more accurate, they are significantly slower than Seq2Seq to train on large datasets. We present strong Seq2seq baselines with sub-word units for our task, that take interactive context into account and scale to large datasets.

| | Train | Dev/Test |
|---|---|---|
| # Examples | 1,518,049 | 1,744/1,981 |
| Avg Context Cells | 29.9 | 28.3 |
| Avg NL Tokens | 39.6 | 57.2 |
| Avg Code Tokens | 38.8 | 33.7 |
| # Unique NL Token | 851,127 | 11,142 |
| # Unique Code Token | 1,001,289 | 5,113 |
| % Use variables above | 45.3 | 58.6 |
| % Use methods above | 6.4 | 8.2 |
| % Contextual | 48.2 | 61.9 |
| % Multi-cell | 19.7 | 29.0 |

Table 2: Statistics for the JuICe dataset. The dev/test sets are obtained from high-quality programming assignments. Expanded descriptions for the bottom four rows are: % Examples using variables defined above, % Examples using methods declared above, % Examples using a variable or method from above (i.e. contextual), % Examples where the variables/methods come from at least 2 different code cells. To accurately measure contextual reasoning, we only consider those variables/methods that are not mentioned in the NL above.

## 3 Dataset Collection

We aim to study code generation in an interactive setting, conditioned on prior history of NL and code cells. Jupyter notebooks are a popular platform for exploratory programming that encourage users to write code one cell at a time and to make programming decisions by executing

previous cells and examining the output. Since these notebooks are heavily used for sharing and collaboration, they are typically well documented with high quality NL. The intent of the NL (markdown) is typically to describe the functionality of the code cells that follow, as opposed to code comments, which primarily describe why code is written in a certain way. We collect a large scale training set to serve as distant supervision for our human-curated dev/test sets based on programming assignments and exercises.

### 3.1 Training Set Creation

To create JuICe we first collect all publicly available Jupyter notebooks from `github.com` created before May 2019 and filter for notebooks having NL markdown in English and Python 2/3 as their kernel type. We observe that the presence of NL markdown is correlated with notebook quality and remove any notebooks that have more than three times the number of code cells as the number of NL cells, leaving us with $\sim$ 659K notebooks.

Within these notebooks, each code cell is a potential training example, and we define the sequence of NL and code cells above it as its context. We only consider code cells that have NL markdown in the immediate previous cell (henceforth, target NL), as target cells. For our dataset, we include target cells that have syntactically valid Python and at most one method definition (for cells containing methods). Following Iyer et al. (2018), we canonicalize strings in the target cell and truncate them to a maximum of 120 code tokens. This amounts to over 4 million (target cell, context) examples that can be used for training, which we downsample to 1.5 million by ensuring that each target cell and context is unique. Table 2 presents dataset statistics for JuICe.

### 3.2 Human Curated Dev/Test Set

While existing large scale NL-code datasets have leveraged noisy online code repositories/forums to create training sets, obtaining a human curated test set has always been prohibitively expensive, and is typically done only for simple single line NL to code tasks. For our dataset, we take advantage of programming assignments written in Jupyter notebooks to build a high quality dev/test set. Specifically, we collect 1,510 code context examples from assignments created with nbgrader (Hamrick, 2016), and mine an additional 2,215 examples from in-class exercise notebooks.

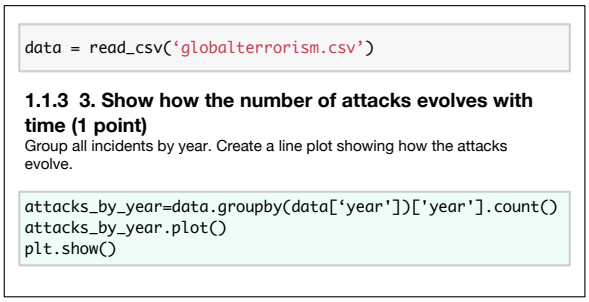


Figure 2: This example nbgrader problem prepared by an instructor prompts the student to fill in the blue colored cell with code to perform operations on previously loaded data. We leverage such notebooks with embedded solutions, posted on Github, to create a high quality dev and test set for JuICe.

**Nbgrader Assignments** Instructors use nbgrader to create and grade programming assignments within Jupyter notebooks. Figure 2 presents one such nbgrader assignment, where the student must wrangle some data that has been loaded earlier in the notebook to create a plot, and enter the solution in the cell in blue. Behind the scenes, nbgrader stores the following properties within each code cell's metadata: (1) the number of points its worth, (2) a unique code cell ID, (3) whether it's to be autograded or manually graded, (4) boilerplate code, (5) whether it is an instructor solution or a student submission. While students are expected to confidentially send their completed assignment to instructors, the solutions are often later posted on Github.

Taking advantage of this, we search Github for nbgrader notebooks with embedded metadata and collect 13,905 such notebooks, each containing multiple prompts and target code cells, with multiple student submissions for the same assignment. Based on the problem IDs and point values, we group together all student submissions that answer the same instructor prompt. Each prompt written by an instructor has an average of 8.9 different student submissions. If the metadata of any solution notebook indicates a successful execution on all the test cases without generating errors in the output, we select that as one of the correct submissions. If there are multiple correct submissions we select one at random, and wrong solutions are discarded. If the instructor provided boilerplate within the target cell, it is extracted and placed into the context above. Thus, each prompt is paired with one solution, which forms our target cell.

**In-Class Exercises** While instructors use

nbgrader for creating formal programming assignments, they often use vanilla Jupyter notebooks for creating informal in-class exercises. In this case, an instructor can release an exercise notebook intended for students to complete and a companion solution notebook to provide immediate feedback. The students are free to solve the exercise and upload the notebooks to their git repositories.

We use the following procedure to extract high quality solution target code cells from these notebooks. First, we collect notebooks containing the word *solution* in the title, and pair it with an exercise notebook from the same repository if over 50% of the cells are the same and *solution* is not in the exercise notebook's title. We find 170K such notebook pairs. If corresponding code cells in the two notebooks differ, then the exercise cell contains the student solution and the other, the instructor solution. We consider these cells as potential target cells, and to verify the instructor solution, we confirm that the same notebook and solution exist in at least 2 other Github repositories.

We find a total of 1,510 nbgrader assignment problems and 2,215 in-class exercise problems and split them randomly into the dev and test sets. We perform the same checks on the target cell as we do on the training set and additionally convert any solutions written in Python 2 to Python 3. Table 2 shows the dataset statistics. We release this, together with our training set, as the first large scale dataset to study code generation with prior NL and code history. Table 1 compares JuICe with various recently released code generation datasets on four different dimensions: interactive history, domain specificity, scale, and human annotation.

## 4 Data Statistics

Our quantitative and qualitative analysis of examples in JuICe show that the NL is diverse, the code broadly covers data science applications, and that context is important for generating the target cells.

**NL Analysis** Since assignment prompts tend to be more detailed in order to provide specific instructions for students, the average NL length in dev/test is ∼20 tokens longer than that in train (Table 2). Table 3 presents statistics of 50 target NL cells classified into overlapping categories. A high percentage (43%) of the NL is abstract language in the declarative or interrogative form, while 27% explicitly state which variables/functions to use, and 16% have a line-by-line correspondence between each line of language and code. For the latter two categories we believe that getting full exact match is possible since the procedure is clearly specified. The last three rows of Table 3 present interesting NL phenomenon. 11% refer to markdown far above in the same notebook, 9% show examples of the desired inputs and outputs of the code to clarify challenging language, and 7% use mathematical equations to formally specify the intended mathematical operations to be performed.

**Code Analysis** Table 4 presents statistics of 50 code cells sampled from the dev set. Majority of code cells are geared towards data science applications. The *Data Exploration* category covers scenarios for computing statistics or queries to gain new insights about the data, and the *Machine Learning* category covers training models, as well as tensor operations. 13% of code cells are used for plotting figures and the *Miscellaneous* category covers a broad range of utility functions.

**Quality Evaluation** We estimate the quality of our dataset using a sample of 50 examples from our train and dev splits. A good quality example is one in which there is enough signal to generate the full target code cell using the NL and the context. We find that 68% examples from train and 96% examples from dev are solvable. Most of the noise in the training set arises from the target NL referring to another code cell (for example, the cell above the NL). Other examples require sophisticated background knowledge e.g. the mathematical processes behind music composition.

**Contextual Reasoning** An analysis of 50 examples reveals the diverse types of contextual reasoning required to generate the target cell. Most examples require using variables or methods defined above (similar to statistics in Table 2). 39% require understanding the creation and usage of data structures in the context. For example, in row 2 of Table 5, knowing that `unemployment` stores the rates for every quarter in the array, suggests the use of the `diff` function which finds the difference between every consecutive value. In other cases, knowing the column names of tables is necessary and these are often specified as positional arguments while loading the data. Being able to condition on loaded table columns and values as in Krishnamurthy et al. (2017) using stored in-notebook variable outputs is an interesting area for future work. 25% examples require reusing idioms spe-

| NL Type | % | Example |
|---|---|---|
| High-level Declarative | 43 | Analyze the Average prices(open, high, low, close) for all stocks in the Month of May |
| Variables/Functions | 27 | Convert all categorical variables in `bank_features` into indicator variables |
| Arguments Mentioned | 20 | Use unigrams, bigrams, trigrams ... english stopwords |
| Question | 18 | Which is the cheapest video game ? |
| Line-by-line | 16 | Create a histogram. Use default settings. Add the label... |
| Long References | 11 | 3d ) Transform reviews...using the tfidf vectorizer we created in part 2 . Save the transformed... |
| Input/Output | 9 | ...Given strings a and b, return a single string with a and b separated by a space, except swap the first 2 chars of each string. e.g. (`mix, pod` → `pox mid` and (`dog, dinner` → `dig, donner`) |
| Equation | 7 | Solve the system $C_\epsilon \theta_\epsilon = b_\epsilon$ for $\theta_\epsilon$. Store the result in a variable, `theta_eps`. |

Table 3: Qualitative analysis of NL from the dev set of JuICe. A large percentage of the NL is abstract and describes the code at a high level. Furthermore, instructors use a variety of additional modalities to elucidate the prompt, including input-output examples and equations.

| Code Type | % |
|---|---|
| Data Exploration | 25 |
| Data Wrangling | 23 |
| Machine Learning | 20 |
| Miscellaneous | 16 |
| Visualization | 13 |
| Systems | 3 |

Table 4: JuICe includes code for various real world applications, primarily for data science/machine learning.

cific to the notebook to tailor the code to the programmer's style or to leverage efficient primitives that the user has defined. For example, the implementation for `countEvenDigits` (row 3) can reuse the efficient list comprehension defined in a similar method above. Lastly, 23% of examples require multi-cell reasoning where the model must incorporate elements from at least 3 cells. In row 4 of Table 5, the `GridSearch` object created in one cell needs to be used to select the best hyperparameter values, and retrain the model using the same procedure found in another code cell.

Finally, using rule-based metrics, we find that 61% examples are contextual i.e. use a token from the context (from Table 2), whereas 86% examples required contextual reasoning that includes reasoning from data and idioms as well.

## 5 Problem formulation

We investigate two tasks for target cell generation in notebooks, conditioned on the context cells up to the target cell: (1) The Full code generation task, which involves generating the entire code snippet in the target cell. (2) The API sequence task, which involves generating only the sequence of constructor and method calls within the target cell, without having to generate variables and arguments. This task is more relaxed and can itself be very helpful to developers under an interactive setting. We do not aim to generate keyword functions such as *range* as these are very commonly used. Figure 1 presents an example for our code generation task, where our goal is to learn models that will map the NL *Create and train the model*, together with the context cells above, into either the the full code snippet (in blue) that reuses previously defined variables `X_train` and `y_train`, or the API sequence, *DecisionTreeClassifier fit*.

Formally, let $a^{(i)}$ denote the API sequence, and $s^{(i)}$ denote the code token sequence of the target cell in the $i$th training example. Further, if $c_d^{(i)}$ represents the context cell at a distance $d, d > 0$, above the target cell of example $i$, our tasks are to generate $a^{(i)}$ and $s^{(i)}$, conditioned on the context cells. Note that $c_d^{(i)}$ can be a sequence of NL words or code tokens. In models that we present in later sections, we limit $d$ to a maximum of $K$ cells.

## 6 Baselines

We train and evaluate a selection of neural (using fairseq: Ott et al. (2019)) and non-neural baselines that are known to perform well for code generation tasks, particularly on large scale datasets.

### 6.1 Retrieval

This baseline presents the programmer with an existing code cell from elsewhere in the training corpus as a possible prediction for the target code cell, located using only the NL markdown immediately before the target cell i.e. $c_1$. We evaluate two variants: (1) Ret-Train uses the code cell underneath the NL markdown cell from the entire training set,

| Type | % | Example |
|---|---|---|
| Incorporates **Variables** or **Functions** declared or used above. | 48 | *d = 2* **NL:** Find all prices in the following test sentence.<br>*d = 1*<br>`test_string = """The iPhone X costs over $999, while the Android competitor comes in at around $550"""`<br>*Target*<br>`pattern = '\$\d+'`<br>`re.findall(pattern, test_string)` |
| Requires understanding the properties of the **data** and its underlying structure to perform logic. | 39 | *d = 6* **NL:** Each number in the array `unemployment` is the unemployment rate at the start of one quarter (a 3-month period) of a year.<br>*d = 1* **NL:** **Question 3.** What was the biggest increase in the unemployment rate from one quarter to the next?<br>*Target*<br>`biggest_increase = max(np.diff(unemployment))` |
| Incorporates coding patterns or **idioms** specific to the notebooks. | 25 | *d = 3*<br>`def sumDigits(number):`<br>`  listn = [int(i) for i in str(number)]`<br>`  ...`<br>*d = 1* **NL:** Complete the function to output the sum of even number digits of the input number.<br>*Target*<br>`def countEvenDigits(number):`<br>`  listn = [int(i) for i in str(number)]`<br>`  count = 0`<br>`  for i in listn:`<br>`  if i%2 == 0:`<br>`  ...` |
| **Multi-cell reasoning** involves reasoning over at least 3 context cells. | 23 | *d = 4*<br>`def cv_optimize_ridge(X, y, n_folds=4):`<br>`  est = Ridge()`<br>`  parameters = {"alpha": [1e-8, 1e-6, 1e-4, 1e-3, 1e-2, 1e-1, 1.0]}`<br>`  gs = GridSearchCV(est, param_grid=parameters, cv=n_folds)`<br>`  gs.fit(X, y)`<br>`  return gs`<br>*d = 3*<br>`fitmodel = cv_optimize_ridge(Xtrain, ytrain, n_folds=4)`<br>*d = 1* **NL:** Assign to variable est the classifier obtained by fitting the entire training set using the best alpha found above.<br>*Target*<br>`alphawechoose = fitmodel.best_params_['alpha']`<br>`est = Ridge(alpha=alphawechoose).fit(Xtrain,ytrain)` |

Table 5: Types of contextual reasoning required for target cell generation. Overall, 86% of examples require contextual reasoning.

that is most similar to $c_1$, (2) Ret-Context uses the code cell underneath the most similar NL cell amongst all $c_d$ (context above). The similarity between two NL cells is computed by representing both NL sequences as vectors using tf-idf representations for their dimensions and measuring the cosine distance between these vectors. We convert the retrieved code cell to an API sequence for the API sequence task.

### 6.2 LSTM with Attention

This baseline is a neural encoder-decoder model where the encoder computes contextualized representations of input sequence embeddings using an n-layer BiLSTM, and an LSTM-based decoder produces a sequence of code (API) tokens, while attending to the encoder representations at every time step $t$. We make use of the general global attention mechanism by Luong et al. (2015). To use this model for our task, we concatenate all $K$ context cells, $c_{1...K}$, using the type of each cell as separator symbols i.e. `CODE` or `MARKDOWN`. The output sequence is supervised to be either an API sequence or a sequence of code tokens.

### 6.3 Transformer

This baseline utilizes the Transformer (Vaswani et al., 2017) architecture where the encoder and decoder consist of multiple layers of multi-headed self-attention and position-wise feed forward layers. The inputs and outputs are the same as the LSTM model described above.

## 7 Experiments

We run code generation experiments to evaluate our baselines on JuICe and to study the effects of

| Models | K | N | Dev Full Code Bleu | Dev Full Code EM | Test Full Code Bleu | Test Full Code EM |
|---|---|---|---|---|---|---|
| Ret-Train | - | 100,000 | 5.52 | 0.82 | 4.76 | 0.48 |
| Ret-Context | $\|c\|$ | - | 3.42 | 0.12 | 3.24 | 0.00 |
| Transformer | 1 | 100,000 | 3.57 | 0.00 | 3.21 | 0.00 |
| Transformer | 3 | 100,000 | 10.90 | 0.27 | 11.08 | 0.38 |
| LSTM | 1 | 100,000 | 7.35 | 0.05 | 7.92 | 0.14 |
| LSTM | 3 | 100,000 | 15.88 | 1.42 | 17.03 | 1.33 |
| **LSTM** | **3** | **1,518,049** | **21.66** | **5.57** | **20.92** | **5.71** |

Table 6: Exact Match and BLEU score for the full code generation task on both the dev and test sets of JuICe for all baselines. All models benefit with additional code context ($K = 3$), which permits conditioning on variables and methods defined previously. Training on additional data further pushes up performance.

| Models | K | N | Dev API Precision | Dev API Recall | Test API Precision | Test API Recall |
|---|---|---|---|---|---|---|
| LSTM | 1 | 100,000 | 34.15 | 29.19 | 30.38 | 26.98 |
| LSTM | 3 | 100,000 | 37.09 | 31.48 | 37.44 | 33.23 |
| **LSTM** | **3** | **1,518,049** | **51.34** | **44.83** | **52.60** | **46.46** |

Table 7: Precision and Recall for the API sequence task on the dev/test sets of JuICe for all baselines

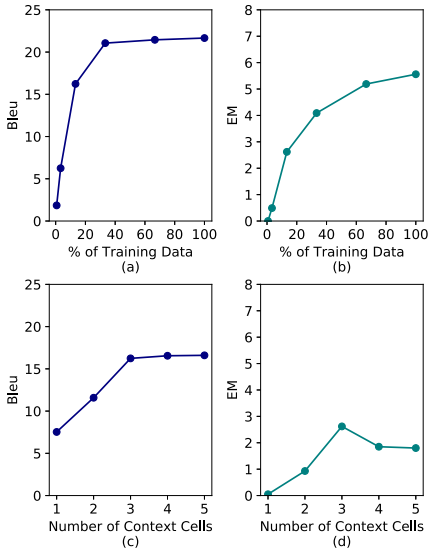


Figure 3: (a) and (b) show the effects of training data on LSTM(3-ctx) performance. Feeding the model more JuICe consistently improves EM accuracy. (c) and (d) show the effects of context cells on performance of LSTM trained with 200K examples. Performance improves for up to 3 cells, after which reasoning over long contexts becomes challenging for current methods.

context size $K$ and dataset size $N$.

To study the effect of context, we consider two configurations for our neural approaches, $K = 1$ which consists of only the NL markdown above the target cell, and $K = 3$. For both cases, we truncate the contents of each context cell to 75 tokens, except for the target NL cell, which remains as is. We consider $K = |c|$ for the retrieval baseline as it is allowed to access all context cells above. We train these configurations for the Seq2Seq models on a subset of 100K examples from the training set, and use the best settings to train on the full training set.

### 7.1 Hyperparameters

For Seq2Seq models, we use an embedding size of 1024 for the input and output tokens. We consolidate the NL and code vocabularies and apply BPE with 10,000 merges. Both encoder and decoder LSTMs use 2 layers with a hidden size of 1024. We use dropout $p = 0.5$ in between LSTM layers and over the decoder output. The model is trained for 40 epochs using gradient based methods using the Adam optimizer (Kingma and Ba, 2015) with a learning rate of 0.001, and we use beam search based decoding with a beam size of 5 during inference. The Transformer model uses 6 layers and 4 attention heads. The dimension of the model is 512 and the dimension of the feed forward network is 1024. We use a learning rate of 0.0005 with 2,000 warmup steps, train for 200 epochs, and pick the model at the epoch with highest validation performance.

### 7.2 Metrics

Similar to prior code generation tasks, we use Exact Match (EM) accuracy and corpus-level BLEU score (Papineni et al., 2002) as performance metrics for full code generation. While EM is a strict metric measuring the ability of models to generate fully functioning code, BLEU serves as a measure of partial credit for models that can provide users with a partially correct snippet.

For the API sequence task, we find BLEU and EM inadequately measure performance since the average API sequence length for our target code is 3.8. We therefore treat the predicted API calls as a set of values and compute precision/recall metrics against the gold set of API calls.

## 8 Results and Discussion

We present EM and BLEU scores for the code generation task in Table 6, and precision/recall for the API sequence task in Table 7, on both the dev

and test sets of JuICe for all our baselines.

All LSTM models outperform both retrieval baselines by more than 2% BLEU score. We find that increasing the amount of training data and the context length helps all models for both tasks. Our best model was trained on 1.5 million examples with a context length of 3 (Table 6). The improvement with context is most likely owing to the interactive nature of the notebook, where variables (methods) defined in previous code cells are operated upon (called) in future cells. However, note that the retrieval model which is given access to the full context performs significantly worse, suggesting that the code above is different than the target code and the model needs to reason over which pieces of it to incorporate.

Figure 3 illustrates the effects of varying training data size and context length on our LSTM model. We find that increasing the amount of training data helps significantly up to 200K examples, after which, performance begins to plateau. Furthermore, the model improves with more context up to 3 cells and then struggles to incorporate additional cells as the average input sequence length exceeds 156 tokens.

Using student notebook solutions for a subset of the test set, we estimate a human performance of 60% BLEU and 23% EM (can be much higher with identifier canonicalization) for the code generation task, and 84% precision and 85% recall for the API sequence task.

### 8.1 Error Analysis

We conduct a qualitative error analysis using 50 erroneous predictions of our top performing model - **LSTM K=3, N=1,518,049** on dev set examples (Table 8). We find that for 39% of cases, the model misunderstands the core intent of the NL and generates totally incorrect code. In 17% cases, it misses some positional arguments in the code. For example, *Use unigrams, bigrams, and trigrams. Use English stop words* should result in a Vectorizer object with arguments: `stop_words="english", ngram_range=(1,3)`. 26% of cases represent an inadequate understanding of the context and 10% require more cells than the 3 presented to the model. Retrieving relevant context cells far above the target is an interesting area for future work. 26% cases are partially correct i.e. they contain some correct code lines but miss some details. For example, for the NL *Create a histogram of all people's ages. Use the default settings. Add the label Age on the x-axis and Count on the y-axis*, the model generates the code for plotting the histogram but misses labeling the axes. For 15% of cases the model generates semantically equivalent code but is different at the surface level. In the future, executing the code to compare outputs or verifying that the nbgrader tests pass would alleviate this issue but requires the model to generate valid strings (currently strings are canonicalized).

| Error Category | % |
|---|---|
| Challenging NL Reasoning | 39 |
| Arguments Missed | 17 |
| Contextual Reasoning | 26 |
| Needs Longer Context | 10 |
| Partially Correct | 26 |
| Semantically Equivalent | 15 |

Table 8: Qualitative error analysis on 50 incorrectly generated code cells from our dev set for our best performing baseline. The first category represents cases where the model output was totally incorrect. Over 30% of errors require deeper contextual reasoning.

## 9 Conclusion

In this paper, we introduced the task of code generation under the paradigm of interactive computing, conditioned on a context of interleaved code snippet cells and NL markdown. To train models for this task, we collected a new large scale open-domain dataset (JuICe) from publicly available Jupyter notebooks, consisting of target code cells paired with sequences of NL and code context. Furthermore, we gathered a high quality evaluation set using nbgrader and in-class programming assignment notebooks with solutions to reliably test code generation models. We evaluated a variety of baseline models for two context dependent code generation tasks, API sequence generation and full code generation. Experiments showed that performance improves on using an increased amount of code context and training data, with significant room for improvement.

## Acknowledgements

The research was supported in part by the ARO (ARO-W911NF-16-1-0121) and the NSF (IIS-1252835, IIS-1562364). The authors thank the anonymous reviewers for their helpful comments.

## References


Miltiadis Allamanis, Hao Peng, and Charles Sutton. 2016. A convolutional attention network for extreme summarization of source code. In *International Conference on Machine Learning*, pages 2091–2100.

Miltiadis Allamanis and Charles Sutton. 2013. Mining source code repositories at massive scale using language modeling. In *Proceedings of the 10th Working Conference on Mining Software Repositories*, pages 207–216. IEEE Press.

Miltiadis Allamanis and Charles Sutton. 2014. Mining idioms from source code. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 472–483. ACM.

Miltiadis Allamanis, Daniel Tarlow, Andrew Gordon, and Yi Wei. 2015. Bimodal modelling of source code and natural language. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 2123–2132.

Matej Balog, Alexander L Gaunt, Marc Brockschmidt, Sebastian Nowozin, and Daniel Tarlow. 2016. Deepcoder: Learning to write programs. *arXiv preprint arXiv:1611.01989*.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43. Association for Computational Linguistics.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016a. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany. Association for Computational Linguistics.

Xiaodong Gu, Hongyu Zhang, Dongmei Zhang, and Sunghun Kim. 2016b. Deep api learning. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 631–642. ACM.

Xiaodong Gu, Hongyu Zhang, Dongmei Zhang, and Sunghun Kim. 2016c. Deep api learning. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, FSE 2016, pages 631–642, New York, NY, USA. ACM.

Jessica B Hamrick. 2016. Creating and grading ipython/jupyter notebook assignments with nbgrader. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, pages 242–242. ACM.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 963–973, Vancouver, Canada. Association for Computational Linguistics.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. 2018. Mapping language to code in programmatic context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1643–1652. Association for Computational Linguistics.

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1821–1831.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian E Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica B Hamrick, Jason Grout, Sylvain Corlay, et al. 2016. Jupyter notebooks-a publishing format for reproducible computational workflows.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1516–1526, Copenhagen, Denmark. Association for Computational Linguistics.

Nate Kushman and Regina Barzilay. 2013. Using semantic unification to generate regular expressions from natural language. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 826–836. Association for Computational Linguistics.

Percy Liang, I. Michael Jordan, and Dan Klein. 2013. Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2).

Wang Ling, Phil Blunsom, Edward Grefenstette, Moritz Karl Hermann, Tomáš Kočiský, Fumin Wang, and Andrew Senior. 2016. Latent predictor networks for code generation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 599–609. Association for Computational Linguistics.

Nicholas Locascio, Karthik Narasimhan, Eduardo De Leon, Nate Kushman, and Regina Barzilay. 2016. Neural generation of regular expressions from natural language with minimal domain knowledge. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages

1918–1923, Austin, Texas. Association for Computational Linguistics.

Reginald Long, Panupong Pasupat, and Percy Liang. 2016. Simpler context-dependent logical forms via model projections. *arXiv preprint arXiv:1606.05378*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Yusuke Oda, Hiroyuki Fudaba, Graham Neubig, Hideaki Hata, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2015. Learning to generate pseudo-code from source code using statistical machine translation (t). In *Automated Software Engineering (ASE), 2015 30th IEEE/ACM International Conference on*, pages 574–584. IEEE.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract syntax networks for code generation and semantic parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1139–1149, Vancouver, Canada. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Alane Suhr, Srinivasan Iyer, and Yoav Artzi. 2018. Learning to map context-dependent sentences to executable formal queries. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Pengcheng Yin, Bowen Deng, Edgar Chen, Bogdan Vasilescu, and Graham Neubig. 2018. Learning to mine aligned code and natural language pairs from stack overflow. In *International Conference on Mining Software Repositories*, MSR, pages 476–486. ACM.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 440–450, Vancouver, Canada. Association for Computational Linguistics.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: structured classification with probabilistic categorial grammars. In *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence*.

Luke S Zettlemoyer and Michael Collins. 2009. Learning context-dependent mappings from sentences to logical form. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 976–984. Association for Computational Linguistics.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

# Model-based Interactive Semantic Parsing: A Unified Framework and A Text-to-SQL Case Study

**Ziyu Yao[1], Yu Su[1], Huan Sun[1], Wen-tau Yih[2]***
{yao.470, su.809, sun.397}@osu.edu
scottyih@fb.com
[1]The Ohio State University
[2]Facebook AI Research, Seattle

## Abstract

As a promising paradigm, *interactive semantic parsing* has shown to improve both semantic parsing accuracy and user confidence in the results. In this paper, we propose a new, unified formulation of the interactive semantic parsing problem, where the goal is to design a *model-based* intelligent agent. The agent maintains its own state as the current predicted semantic parse, decides whether and where human intervention is needed, and generates a clarification question in natural language. A key part of the agent is a world model: it takes a percept (either an initial question or subsequent feedback from the user) and transitions to a new state. We then propose a simple yet remarkably effective instantiation of our framework, demonstrated on two text-to-SQL datasets (WikiSQL and Spider) with different state-of-the-art base semantic parsers. Compared to an existing interactive semantic parsing approach that treats the base parser as a black box, our approach solicits less user feedback but yields higher run-time accuracy.[1]

## 1 Introduction

Natural language interfaces that allow users to query data and invoke services without programming have been identified as a key application of semantic parsing (Berant et al., 2013; Thomason et al., 2015; Dong and Lapata, 2016; Zhong et al., 2017; Campagna et al., 2017; Su et al., 2017). However, existing semantic parsing technologies often fall short when deployed in practice, facing several challenges: (1) user utterances can be inherently ambiguous or vague, making it difficult to get the correct result in one shot, (2) the accuracy of state-of-the-art semantic parsers are still not high enough for real use, and (3) it is hard for users to validate the semantic parsing results, especially with mainstream neural network models that are known for the lack of interpretability.

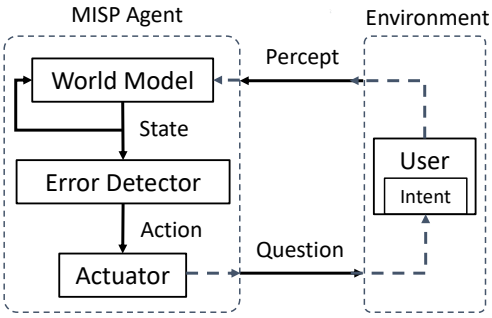


Figure 1: Model-based Interactive Semantic Parsing (MISP) framework.

In response to these challenges, *interactive semantic parsing* has been proposed recently as a practical solution, which includes human users in the loop to resolve utterance ambiguity, boost system accuracy, and improve user confidence via human-machine collaboration (Li and Jagadish, 2014; He et al., 2016; Chaurasia and Mooney, 2017; Su et al., 2018; Gur et al., 2018; Yao et al., 2019). For example, Gur et al. (2018) built the DialSQL system to detect errors in a generated SQL query and request user selection on alternative options via dialogues. Similarly, Chaurasia and Mooney (2017) and Yao et al. (2019) enabled semantic parsers to ask users clarification questions while generating an If-Then program. Su et al. (2018) showed that users overwhelmingly preferred an interactive system over the non-interactive counterpart for natural language interfaces to web APIs. While these recent studies successfully demonstrated the value of interactive semantic parsing in practice, they are often bound to a certain type of formal language or dataset, and the designs are thus ad-hoc and not easily generalizable. For example, DialSQL only applies

*Work started while at AI2

[1]Code available at https://github.com/sunlab-osu/MISP.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5447–5458, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

to SQL queries on the WikiSQL dataset (Zhong et al., 2017), and it is non-trivial to extend it to other formal languages (e.g., $\lambda$-calculus) or even just to more complex SQL queries beyond the templates used to construct the dataset.

Aiming to develop a general principle for building interactive semantic parsing systems, in this work we propose model-based interactive semantic parsing (MISP), where the goal is to design a *model-based intelligent agent* (Russell and Norvig, 2009) that can interact with users to complete a semantic parsing task. Taking an utterance (e.g., a natural language question) as input, the agent forms the semantic parse (e.g., a SQL query) in steps, potentially soliciting user feedback in some steps to correct parsing errors. As illustrated in Figure 1, a MISP agent maintains its *state* as the current semantic parse and, via an *error detector*, decides whether and where human intervention is needed (the *action*). This action is performed by a *question generator* (the *actuator*), which generates and presents to the user a human-understandable question. A core component of the agent is a *world model* (Ha and Schmidhuber, 2018) (hence *model-based*), which incorporates user feedback from the environment and transitions to a new agent state (e.g., an updated semantic parse). This process repeats until a terminal state is reached. Such a design endows a MISP agent with three crucial properties of interactive semantic parsing: (1) being *introspective* of the reasoning process and knowing when it may need human supervision, (2) being able to *solicit user feedback* in a human-friendly way, and (3) being able to *incorporate user feedback* (through state transitions controlled by the world model).

The MISP framework provides several advantages for designing an interactive semantic parser compared to the existing ad-hoc studies. For instance, the whole problem is conceptually reduced to building three key components (i.e., the world model, the error detector, and the actuator), and can be handled and improved separately. While each component may need to be tailored to the specific task, the general framework remains unchanged. In addition, the formulation of a model-based intelligent agent can facilitate the application of other machine learning techniques like reinforcement learning.

To better demonstrate the advantages of the MISP framework, we propose a simple yet remarkably effective instantiation for the text-to-SQL task. We show the effectiveness of the framework based on three base semantic parsers (SQLNet, SQLova and SyntaxSQLNet) and two datasets (WikiSQL and Spider). We empirically verified that with a small amount of targeted, test-time user feedback, interactive semantic parsers improve the accuracy by 10% to 15% absolute. Compared to an existing interactive semantic parsing system, DialSQL (Gur et al., 2018), our approach, despite its much simpler yet more general system design, achieves better parsing accuracy by asking only half as many questions.

## 2 Background & Related Work

**Semantic Parsing.** Mapping natural language utterances to their formal semantic representations, semantic parsing has a wide range of applications, including question answering (Berant et al., 2013; Dong and Lapata, 2016; Finegan-Dollak et al., 2018), robot navigation (Artzi and Zettlemoyer, 2013; Thomason et al., 2015) and Web API calling (Quirk et al., 2015; Su et al., 2018). The target application in this work is text-to-SQL, which has been popularized by the WikiSQL dataset (Zhong et al., 2017). One of the top-performing models on WikiSQL is SQLNet (Xu et al., 2017), which leverages the pre-defined SQL grammar sketches on WikiSQL and solves the SQL generation problem via "slot filling." By augmenting SQLNet with a table-aware BERT encoder (Devlin et al., 2019) and by revising the value prediction in `WHERE` clauses, SQLova (Hwang et al., 2019) advances further the state of the art. Contrast to WikiSQL, the recently released Spider dataset (Yu et al., 2018c) focuses on complex SQL queries containing multiple keywords (e.g., `GROUP BY`) and may join multiple tables. To handle such complexity, Yu et al. (2018b) proposed SyntaxSQLNet, a syntax tree network with modular decoders, which generates a SQL query by recursively calling a module following the SQL syntax. However, because of the more realistic and challenging setting in Spider, it only achieves 20% in accuracy.

We experiment our MISP framework with the aforementioned three semantic parsers on both WikiSQL and Spider. The design of MISP allows naturally integrating them as the base parser. For example, when SQLNet fills a sequence of slots to produce a SQL query, a "state" in MISP corresponds to a partially generated SQL query and it

transitions as SQLNet fills the next slot.

**Interactive Semantic Parsing.** To enhance parsing accuracy and user confidence in practical applications, interactive semantic parsing has emerged as a promising solution (Li and Jagadish, 2014; He et al., 2016; Chaurasia and Mooney, 2017; Su et al., 2018; Gur et al., 2018; Yao et al., 2019). Despite their effectiveness, existing solutions are somewhat ad-hoc and bound to a specific formal language and dataset. For example, DialSQL (Gur et al., 2018) is curated for WikiSQL, where SQL queries all follow the same and given grammar sketch. Similarly, (Yao et al., 2019) relies on a pre-defined two-level hierarchy among components in an If-Then program and cannot generalize to formal languages with a deeper structure. In contrast, MISP aims for a general design principle by explicitly identifying and decoupling important components, such as error detector, question generator and world model. It also attempts to integrate and leverage a strong base semantic parser, and transforms it to a natural interactive semantic parsing system, which substantially reduces the engineering cost.

## 3 Model-based Interactive Semantic Parsing

We now discuss the MISP framework (Figure 1) in more detail. Specifically, we highlight the function of each major building block and the relationships among them, and leave the description of a concrete embodiment to Section 4.

**Environment.** The *environment* consists of a user with a certain intent, which corresponds to a semantic parse that the user expects the agent to produce. Based on this intent, the user gives an initial natural language utterance $u_0$ to start a semantic parsing *session* and responds to any clarification question from the agent with feedback $u_t$ at interaction turn $t$.

**Agent State.** The *agent state* $s$ is an agent's internal interpretation of the environment based on all the available information. A straightforward design of the agent state is as the currently predicted semantic parse. It can also be endowed with meta information of the parsing process such as prediction probability or uncertainty to facilitate error detection.

**World Model.** A key component of a MISP agent is its *world model* (Ha and Schmidhuber, 2018), which compresses the historical percepts throughout the interaction and predicts the future based on the agent's knowledge of the world. More specifically, it models the transition of the agent state, $p(s_{t+1}|s_t, u_t)$, where $u_t$ is the user feedback at step $t$ and $s_{t+1}$ is the new state. The transition can be deterministic or stochastic.

**Error Detector.** A MISP agent introspects its state and decides whether and where human intervention is needed. The *error detector* serves this role. Given the current state $s_t$ (optionally the entire interaction history) and a set of *terminal states*, it decides on an *action* $a_t$: If the agent is at a terminal state, it terminates the session, executes the semantic parse, and returns the execution results to the user; otherwise, it determines a span in the current semantic parse that is likely erroneous and passes it, along with necessary context information needed to make sense of the error span, to the actuator.

**Actuator.** An *actuator* has a user-facing interface and realizes an agent's actions in a user-friendly way. In practice, it can be a natural language generator (NLG) (He et al., 2016; Gur et al., 2018; Yao et al., 2019) or an intuitive graphical user interface (Su et al., 2018; Berant et al., 2019), or the two combined.

## 4 MISP-SQL: An Instantiation of MISP for Text-to-SQL

Under the MISP framework, we design an interactive semantic parsing system (Figure 2), named MISP-SQL, for the task of text-to-SQL translation. MISP-SQL assumes a base text-to-SQL parser and leverages it to design the world model and the error detector. The world model is essentially a wrapper that takes the user input and changes the behavior of the base semantic parser (e.g., by changing the probability distribution or removing certain prediction paths). The error detector makes decisions based on the *uncertainty* of the predictions: if the parser is uncertain about a prediction, it is more likely to be an error. The actuator is a template-based natural language question generator developed for the general SQL language. Figure 2 shows an example of the MISP-SQL agent.

### 4.1 Agent State

For ease of discussion, we assume the base parser generates the SQL query by predicting a sequence

of SQL components,[2] as in many state-of-the-art systems (Xu et al., 2017; Wang et al., 2018; Yu et al., 2018a; Hwang et al., 2019). Agent state $s_t$ is thus defined as a *partial* SQL query, i.e., $s_t$=$\{o_1, o_2, ..., o_t\}$, where $o_t$ is the predicted SQL component at time step $t$, such as `SELECT place` in Figure 2. What constitutes a SQL component is often defined differently in different semantic parsers, but typically dictated by the SQL syntax. To support introspection and error detection, each prediction is associated with its uncertainty, which is discussed next.

### 4.2 Error Detector

The error detector in MISP-SQL is introspective and greedy. It is introspective because it examines the uncertainty of the predictions as opposed to the predictions themselves. On the other hand, it is greedy because its decisions are solely based on the last prediction $o_t$ instead of the entire state $s_t$.

We experiment with two uncertainty measures, based on the probability of $o_t$ estimated by the base semantic parser, as well as its standard deviation under Bayesian dropout (Gal and Ghahramani, 2016), respectively.

**Probability-based Uncertainty.** Intuitively if the base semantic parser gives a low probability to the top prediction at a step, it is likely uncertain about the prediction. Specifically, we say a prediction $o_t$ needs user clarification if its probability is lower than a threshold $p^*$, i.e.,

$$p(o_t) < p^*.$$

This strategy is shown to be strong in detecting misclassified and out-of-distribution examples (Hendrycks and Gimpel, 2017).

**Dropout-based Uncertainty.** Dropout (Srivastava et al., 2014) has been used as a Bayesian approximation for estimating model uncertainty (Gal and Ghahramani, 2016) in several tasks (Dong et al., 2018; Siddhant and Lipton, 2018; Xiao and Wang, 2019). Different from its standard application to prevent models from overfitting in training time, we use it at test time to measure model uncertainty, similar to (Dong et al., 2018). The intuition is that if the probability on a prediction varies dramatically (as measured by the standard deviation) across different perturbations under dropout, the model is likely uncertain about it. Specifically, the uncertainty on prediction $o_t$ is calculated as:

$$\text{STDDEV}\{p(o_t|W_i)\}_{i=1}^{N},$$

where $W_i$ is the parameters of the base semantic parser under the $i$-th dropout perturbation, and the uncertainty score is the standard deviation of the prediction probabilities over $N$ random passes. We say $o_t$ needs user clarification if its uncertainty score is greater than a threshold $s^*$.

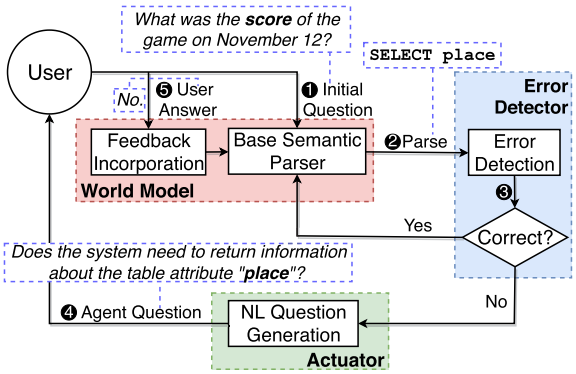


Figure 2: MISP-SQL Agent. The base semantic parser incrementally parses the user question (Step 1) into a SQL query by first selecting a column from the table (Step 2). This partial parse is examined by the error detector (Step 3), who determines that the prediction is incorrect (because the uncertainty is high) and triggers the actuator to ask the user a clarification question (Step 4). The user feedback is then incorporated into the world model (Step 5) to update the agent state. If the prediction was correct, Step 2 would be repeated to continue the parsing.

**Terminal State.** The only terminal state is when the base semantic parser indicates end of parsing.

### 4.3 Actuator: An NL Generator

The MISP-SQL agent performs its action (e.g., validating the column "*place*") via asking users *binary* questions, hence the actuator is a natural language generator (NLG). Although there has been work on describing a SQL query with an NL statement (Koutrika et al., 2010; Ngonga Ngomo et al., 2013; Iyer et al., 2016; Xu et al., 2018), few work studies generating *questions* about a certain SQL component in a systematic way.

Inspired by (Koutrika et al., 2010; Wang et al., 2015), we define a rule-based NLG, which consists of a seed lexicon and a grammar for deriving questions. Table 1 shows rules covering

[2] In practice this assumption may not be necessary as long as there is a reasonable way to chunk the semantic parse to calculate uncertainty and formulate clarification questions.

| **[Lexicon]** |
|---|
| is greater than\|equals to\|is less than → OP[>\|=\|<] |
| sum of values in\|average value in\|number of\|minimum value in\|maximum value in → AGG[sum\|avg\|count\|min\|max] |
| **[Grammar]** |
| "*col*" → COL[*col*] |
| Does the system need to return information about COL[*col*] ? → Q[*col*‖SELECT *agg*? *col*] |
| Does the system need to return AGG[*agg*] COL[*col*] ? → Q[*agg*‖SELECT *agg col*] |
| Does the system need to return a value after any mathematical calculations on COL[*col*] ? → Q[*agg*=None‖SELECT *col*] |
| Does the system need to consider any conditions about COL[*col*] ? → Q[*col*‖WHERE *col op val*] |
| The system considers the following condition: COL[*col*] OP[*op*] a value. Is this condition correct? → Q[*op*‖WHERE *col op val*] |
| The system considers the following condition: COL[*col*] OP[*op*] *val*. Is this condition correct? → Q[*val*‖WHERE *col op val*] |

Table 1: Domain-general lexicon and grammar for NL generation in MISP-SQL (illustrated for WikiSQL; a more comprehensive grammar for Spider can be found in Appendix A).

SQL queries on WikiSQL (Zhong et al., 2017). The seed lexicon defines NL descriptions for basic SQL elements in the form of "$n \rightarrow t[p]$", where $n$ is an NL phrase, $t$ is a pre-defined syntactic category and $p$ is either an aggregator (e.g., `avg`) or an operator (e.g., $>$). For example, "is greater than $\rightarrow$ OP[$>$]" specifies a phrase "is greater than" to describe the operator "$>$". In MISP-SQL, we consider four syntactic categories: AGG for aggregators, OP for operators, COL for columns and Q for generated questions. However, it can be extended with more lexicon entries and grammar rules to accommodate more complex SQL in Spider (Yu et al., 2018c), which we show in Appendix A.

The grammar defines rules to derive questions. Each column is described by itself (i.e., the column name). Rules associated with each Q-typed item "Q[$v$‖`Clause`]" constructs an NL question asking about $v$ in `Clause`. The `Clause` is the necessary context to formulate meaningful questions. Figure 3 shows a derivation example. Note that, both the lexicon and the grammar in our system are domain-agnostic in the sense that it is not specific to any database. Therefore, it can be reused for new domains in the future. Database-specific rules, such as naming each column with a more canonical phrase (rather than the column name), are also possible.

### 4.4 World Model

The agent incorporates user feedback and updates its state with a world model. Different from DialSQL which trains an additional neural network, the MISP-SQL agent directly employs the base semantic parser to transition states, which saves additional training efforts.

As introduced in Section 4.3, the agent raises a binary question to the user about a predicted SQL component $o_t$. Therefore, the received user feedback either confirms the prediction or negates it. In the former case, the state is updated by proceeding to the next decoding step, i.e., $s_{t+1}$={$o_1$, ..., $o_t$, $o_{t+1}$}, where $o_{t+1}$ is the predicted next component and $s_{t+1}$ shows the updated partial parse. In the latter case, the user feedback is incorporated to constrain the search space of the base parser (i.e., forbidding the parser from making the same wrong prediction), based on which the parser refreshes its prediction and forms a new state $s_{t+1}$={$o_1$, ..., $o_{t-1}$, $o_{t+1}$}, where $o_{t+1}$ is a predicted alternative to replace $o_t$. To avoid being trapped in a large search space, for each SQL component, we consider a maximum number of $K$ alternatives (in addition to the original prediction) to solicit user feedback on.

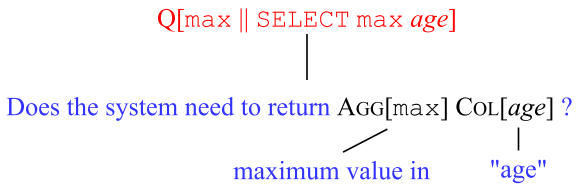


Figure 3: Deriving an NL question about the aggregator `max` in the clause "`SELECT max`(age)" from the rooted Q-typed item .

## 5 Experiments

We apply our approach to the task of mapping natural language questions to SQL queries. In this section, we first describe the basic setup, including the datasets and the base semantic parsers, followed by the system results on both simulated and real users.

### 5.1 Experimental Setup

We evaluate our proposed MISP-SQL agent on WikiSQL (Zhong et al., 2017), which contains 80,654 hand-annotated pairs of ⟨NL question,

| System | SQLNet $\text{Acc}_{\text{qm}}$ | SQLNet $\text{Acc}_{\text{ex}}$ | SQLNet Avg. #q | SQLova $\text{Acc}_{\text{qm}}$ | SQLova $\text{Acc}_{\text{ex}}$ | SQLova Avg. #q |
|---|---|---|---|---|---|---|
| no interaction | 0.615 | 0.681 | N/A | 0.797 | 0.853 | N/A |
| DialSQL | 0.690 | N/A | 2.4 | N/A | N/A | N/A |
| MISP-SQL$^{\text{Unlimit10}}$ | 0.932 | 0.948 | 7.445 | 0.985 | 0.991 | 6.591 |
| MISP-SQL$^{\text{Unlimit3}}$ | 0.870 | 0.900 | 7.052 | 0.955 | 0.974 | 6.515 |
| MISP-SQL$^{p^*=0.95}$ | 0.782 | 0.824 | 1.713 | 0.912 | 0.939 | 0.773 |
| MISP-SQL$^{p^*=0.8}$ | 0.729 | 0.779 | 1.104 | 0.880 | 0.914 | 0.488 |

Table 2: Simulation evaluation of MISP-SQL (based on SQLNet or SQLova) on WikiSQL Test set. "MISP-SQL$^{p^*=\text{X}}$" denotes our agent with probability-based error detection (threshold at X). "MISP-SQL$^{\text{UnlimitK}}$" denotes a variant that asks questions for every component, with up to $K+1$ questions per component.

SQL query⟩, distributed across 24,241 tables from Wikipedia. Our experiments follow the same data split as in (Zhong et al., 2017).

We experiment MISP-SQL with two base semantic parsers: SQLNet (Xu et al., 2017) and SQLova (Hwang et al., 2019). Unlike in DialSQL's evaluation (Gur et al., 2018), we do not choose Seq2SQL (Zhong et al., 2017) as a base parser but SQLova instead, because it achieves similar performance as SQLNet while SQLova is currently the best open-sourced model on WikiSQL, which can give us a more comprehensive evaluation. For each of the two base semantic parsers, we test our agent with two kinds of error detectors, based on prediction probability and Bayesian dropout, respectively (Section 4.2). We tune the threshold $p^*$ within $0.5 \sim 0.95$ and $s^*$ within $0.01 \sim 0.2$. Particularly for uncertainty-based detection measured by Bayesian dropout, the number of passes $N$ is set to 10, with a dropout rate 0.1. The dropout layers are applied at the same positions as when each semantic parser is trained. When the agent interacts with users, the maximum number of alternative options (in addition to the original prediction) per component, $K$, is set to 3. If the user negates all the $K+1$ predicted candidates, the agent will keep the original prediction, as in (Gur et al., 2018).

### 5.2 Simulation Evaluation

In simulation evaluation, each agent interacts with a *simulated* user, who gives a yes/no answer based on the ground-truth SQL query. If the agent fails to correct its predictions in three consecutive interaction turns, the user will leave the interaction early and the agent has to finish the remaining generation without further help from the user.

**Overall Comparison.** We first compare MISP-SQL with the two base semantic parsers without interactions in Table 2. For SQLNet, we also compare our system with the reported performance of DialSQL (Gur et al., 2018, Table 4). However, since DialSQL is not open-sourced and it is not easy to reproduce it, we are unable to adapt it to SQLova for more comparisons. Following (Xu et al., 2017; Hwang et al., 2019), we evaluate the SQL query match accuracy ("**$\text{Acc}_{\text{qm}}$**", after converting the query into its canonical form) and the execution accuracy ("**$\text{Acc}_{\text{ex}}$**") of each agent. "**Avg. #q**" denotes the average number of questions per query. For any base parser, MISP-SQL improves their performance by interacting with users. Particularly for SQLNet, MISP-SQL outperforms the DialSQL system with only *half* the number of questions (1.104 vs. 2.4), and has a much simpler design without the need of training an extra model (besides training the base parser, which DialSQL needs to do as well). Our agent can even boost the strong performance of SQLova from 85% to 94% in execution accuracy, with merely 0.773 questions per query.

We also present an "upper-bounded" accuracy of our agent, when it does not adopt any error detector and asks questions about *every* component with at most 10 ("MISP-SQL$^{\text{Unlimit10}}$") or 3 ("MISP-SQL$^{\text{Unlimit3}}$") alternatives. Interestingly, even for the weaker SQLNet parser, most true predictions have already been contained within the top 10 options (giving 0.932 query match accuracy). When equipped with the stronger SQLova parser, the agent has a potential to boost the execution accuracy to around 100% by considering only the top 3 options of every prediction. The complete results can be found in Appendix B.

**Error Detector Comparison.** We then compare the probability-based and dropout-based error detectors in Figure 4, where each marker indicates

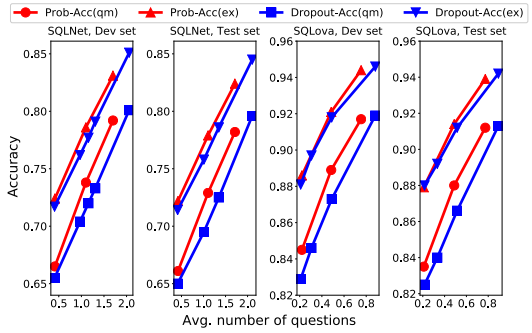


Figure 4: Comparison of probability- and dropout-based error detection.

the agent's accuracy and the average number of questions it needs under a certain error detection threshold. Consistently for both SQLNet and SQLova, the probability-based error detector can achieve the same accuracy with a lower number of questions than the dropout-based detector. It is also observed that this difference is greater in terms of query match accuracy, around 0.15 $\sim$ 0.25 for SQLNet and 0.1 $\sim$ 0.15 for SQLova. A more direct comparison of various settings under the same average number of questions can be found in Appendix C.

To better understand how each kind of error detectors works, we investigate the portion of questions that each detector spends on *right* predictions (denoted as "$\mathbf{Q_r}$"). An ideal system should ask fewer questions on right predictions while identify more truly incorrect predictions to fix the mistakes. We present the question distributions of the various systems in Table 3. One important conclusion drawn from this table is that probability-based error detection is much more effective on identifying incorrect predictions. Consider the system using probability threshold 0.5 for error detection (i.e., "$p^*$=0.5") and the one using dropout-based error detector with a threshold 0.2 (i.e., "$s^*$=0.2") on SQLNet. When both systems ask around the same number of questions during the interaction, the former spends only 16.9% of unnecessary questions on correct predictions ($Q_r$), while the latter asks twice amount of them (32.1%). Similar situations are also observed for SQLova. It is also notable that, when the probability threshold is lower (which results in a fewer total number of questions), the portion of questions on right actions drops significantly (e.g., from 23.0% to 16.9% when the threshold changes from 0.8 to 0.5 on SQLNet). However, this portion remains almost unchanged for dropout-based error detection.

| SQLNet | | | SQLova | | |
|---|---|---|---|---|---|
| **System** | **Avg. #q** | **$Q_r$%** | **System** | **Avg. #q** | **$Q_r$%** |
| $p^*$=0.8 | 1.099 | 23.0% | $p^*$=0.8 | 0.484 | 28.9% |
| $p^*$=0.5 | 0.412 | 16.9% | $p^*$=0.5 | 0.220 | 18.4% |
| $s^*$=0.07 | 1.156 | 34.5% | $s^*$=0.03 | 0.489 | 50.4% |
| $s^*$=0.2 | 0.406 | 32.1% | $s^*$=0.05 | 0.306 | 52.5% |

Table 3: Portion of interaction questions on right predictions ($Q_r$%) for each agent setting on WikiSQL Dev set (smaller is better). "$p^*$/$s^*$=X" denotes our agent with probability/dropout-based error detection (threshold at X).

### 5.3 Extend to Complex SQL Generation

A remarkable characteristic of MISP-SQL is its generalizability, as it makes the best use of the base semantic parser and requires no extra model training. To verify it, we further experiment MISP-SQL on the more complex text-to-SQL dataset "Spider" (Yu et al., 2018c). The dataset consists of 10,181 questions on multi-domain databases, where SQL queries can contain complex keywords such as `GROUP BY` or join several tables. We extend the NLG lexicon and grammar (Section 4.3) to accommodate this complexity, with details shown in Appendix A.

We adopt SyntaxSQLNet (Yu et al., 2018b) as the base parser.[3] In our experiments, we follow the same database split as in (Yu et al., 2018c) and report the Exact Matching accuracy ("$\mathbf{Acc_{em}}$") on Dev set.[4] Other experimental setups remain the same as when evaluating MISP-SQL on WikiSQL. Table 4 shows the results.

We first observe that, via interactions with simulated users, MISP-SQL improves SyntaxSQLNet by 10% accuracy with reasonably 3 questions per query. However, we also realize that, unlike on WikiSQL, in this setting, the probability-based error detector requires more questions than the Bayesian uncertainty-based detector. This can be explained by the inferior performance of the base SyntaxSQLNet parser (merely 20% accuracy without interaction). In fact, the portion of questions that the probability-based detector spends on right predictions ($Q_r$) is still half of that the dropout-based detector asks (12.8% vs. 24.8%). However, it wastes around 60% of questions on unsolvable wrong predictions. This typically hap-

[3]We chose SyntaxSQLNet because it was the best model by the paper submission time. In principle, our framework can also be applied to more sophisticated parsers such as (Bogin et al., 2019; Guo et al., 2019).

[4]We do not report results on Spider test set since it is not publicly available.

| System | $Acc_{em}$ | Avg. #q |
|---|---|---|
| no interaction | 0.190 | N/A |
| MISP-SQL$^{\text{Unlimit10}}$ | 0.522 | 14.878 |
| MISP-SQL$^{\text{Unlimit3}}$ | 0.382 | 11.055 |
| MISP-SQL$^{p^*=0.95}$ | 0.300 | 3.908 |
| MISP-SQL$^{p^*=0.8}$ | 0.268 | 3.056 |
| MISP-SQL$^{s^*=0.01}$ | 0.315 | 3.815 |
| MISP-SQL$^{s^*=0.03}$ | 0.290 | 2.905 |

Table 4: Simulation evaluation of MISP-SQL (built on SyntaxSQLNet) on Spider Dev set.

pens when the base parser is not strong enough, i.e., cannot rank the true option close to the top, or when there are unsolved wrong precedent predictions (e.g., in "`WHERE` *col op val*", when *col* is wrong, whatever *op*/*val* following it is wrong). This issue can be alleviated when more advanced base parsers are adopted in the future.

### 5.4 Human Evaluation

We further conduct human user study to evaluate the MISP-SQL agent. Our evaluation setting largely follows Gur et al. (2018). For each base semantic parser, we randomly sample 100 examples from the corresponding dataset (either WikiSQL Test set or Spider Dev set) and ask three human evaluators, who are graduate students with only rudimentary knowledge of SQL based on our survey, to work on each example and then report the averaged results. We present to the evaluators the initial natural language question and allow them to view the table headers to better understand the question intent. On Spider, we also show the name of the database tables. We select error detectors based on the simulation results: For SQLNet and SQLova, we equip the agent with a probability-based error detector (threshold at 0.95); for SyntaxSQLNet, we choose a Bayesian uncertainty-based error detector (threshold at 0.03). As in the simulation evaluation, we cannot directly compare with DialSQL in human evaluation because the code is not yet publicly available.

Table 5 shows the results. In all settings, MISP-SQL improves the base parser's performance, demonstrating the benefit of involving human interaction. However, we also notice that the gain is not as large as in simulation, especially on SQLova. Through interviews with the human evaluators, we found that the major reason is that they sometimes had difficulties understanding the true intent of some test questions that are ambiguous, vague, or contain entities they are not familiar with. We believe this reflects a general challenge of setting up human evaluation for semantic parsing that is close to the real application setting, and thus set forth the following discussion.

| System | $Acc_{qm/em}$ | $Acc_{ex}$ | Avg. #q |
|---|---|---|---|
| **SQLNet** | | | |
| no interaction | 0.580 | 0.660 | N/A |
| MISP-SQL (simulation) | 0.770 | 0.810 | 1.800 |
| MISP-SQL (real user) | 0.633 | 0.717 | 1.510 |
| **SQLova** | | | |
| no interaction | 0.830 | 0.890 | N/A |
| MISP-SQL (simulation) | 0.920 | 0.950 | 0.550 |
| MISP-SQL (real user) | 0.837 | 0.880 | 0.533 |
| + w/ full info. | 0.907 | 0.937 | 0.547 |
| **SyntaxSQLNet** | | | |
| no interaction | 0.180 | N/A | N/A |
| MISP-SQL (simulation) | 0.290 | N/A | 2.730 |
| MISP-SQL (real user) | 0.230 | N/A | 2.647 |

Table 5: Human evaluation on 100 random examples for MISP-SQL agents based on SQLNet, SQLova and SyntaxSQLNet, respectively.

### 5.5 Discussion on Future Human Evaluation

Most human evaluation studies for (interactive) semantic parsers so far (Chaurasia and Mooney, 2017; Gur et al., 2018; Su et al., 2018; Yao et al., 2019) use pre-existing test questions (e.g., from datasets like WikiSQL). However, this introduces an undesired discrepancy, that is, human evaluators may not necessarily be able to understand the true intent of the given questions in an faithful way, especially when the question is ambiguous, vague, or containing unfamiliar entities.

This discrepancy is clearly manifested in our human evaluation with SQLova (Table 5). When the base parser is strong, many of the remaining incorrectly parsed questions are challenging not only for the base parser but also for human evaluators. We manually examined the situations where evaluators made a different choice than the simulator and found that 80% of such choices happened when the initial question is ambiguous or the gold SQL annotation is wrong. For example, for the question "*name the city for kanjiža*" it is unlikely for human evaluators to know that "kanjiža" is an "Urban Settlement" without looking at the table content or knowing the specific background knowledge beforehand. This issue has also been reported as the main limitation to further improve SQLova (Hwang et al., 2019), which

could in principle be resolved by human interactions if the users have a clear and consistent intent in mind. To verify this, we conduct an additional experiment with SQLova where human evaluators can view the table content as well as the gold SQL query before starting the interaction to better understand the true intent (denoted as "w/ full info" in Table 5). As expected, the MISP-SQL agent performs much better (close to simulation) when users know what they are asking. It further confirms that a non-negligible part of the accuracy gap between simulation and human evaluation is due to human evaluators not fully understanding the question intent and giving false feedback.

To alleviate this discrepancy, a common practice is to show human evaluators the schema of the underlying database, as Gur et al. (2018) and we did (Section 5.4), but it is still insufficient, especially for entity-related issues (e.g., "kanjiža"). On the other hand, while exposing human evaluators to table content helps resolve the entity-related issues, it is likely to introduce undesired biases in favor of the system under test (i.e., "overexposure"), since human evaluators may then be able to give more informative feedback than real users.

To further reduce the discrepancy between human evaluation and real use cases, one possible solution is to ask human evaluators to come up with questions from scratch (instead of using pre-existing test questions), which guarantees intent understanding. While this solution may still require exposure of table content to evaluators (such that they can have some sense of each table attribute), overexposure can be mitigated by showing them only part (e.g., just a few rows) of the table content, similar to the annotation strategy by Zhong et al. (2017). Furthermore, the reduced controllability on the complexity of the evaluator-composed questions can be compensated by conducting human evaluation in a larger scale. We plan to explore this setting in future work.

## 6 Conclusion and Future Work

This work proposes a new and unified framework for the interactive semantic parsing task, named MISP, and instantiates it successfully on the text-to-SQL task. We outline several future directions to further improve MISP-SQL and develop MISP systems for other semantic parsing tasks:

**Improving Agent Components.** The flexibility of MISP allows improving on each agent component separately. Take the error detector for example. One can augment the probability-based error detector in MISP-SQL with probability calibration, which has been shown useful in aligning model confidence with its reliability (Guo et al., 2017). One can also use learning-based approaches, such as a reinforced decision policy (Yao et al., 2019), to increase the rate of identifying wrong and solvable predictions.

**Lifelong Learning for Semantic Parsing.** Learning from user feedback is a promising solution for lifelong semantic parser improvement (Iyer et al., 2017; Padmakumar et al., 2017; Labutov et al., 2018). However, this may lead to a non-stationary environment (e.g., changing state transition) from the perspective of the agent, making its training (e.g., error detector learning) unstable. In the context of dialog systems, Padmakumar et al. (2017) suggests that this effect can be mitigated by jointly updating the dialog policy and the semantic parser batchwisely. We leave exploring this aspect in our task to future work.

**Scaling Up.** It is important for MISP agents to scale to larger backend data sources (e.g., knowledge bases like Freebase or Wikidata). To this end, one can improve MISP from at least three aspects: (1) using more intelligent interaction designs (e.g., free-form text as user feedback) to speed up the hypothesis space searching globally, (2) strengthening the world model to nail down a smaller set of plausible hypotheses based on both the initial question and user feedback, and (3) training the agent to learn to improve the parsing accuracy while minimizing the number of required human interventions over time.

## Acknowledgments

This research was sponsored in part by the Army Research Office under cooperative agreements W911NF-17-1-0412, NSF Grant IIS1815674, Fujitsu gift grant, and Ohio Supercomputer Center (Center, 1987). The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Office or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notice herein.

## References


Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1:49–62.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544.

Jonathan Berant, Daniel Deutch, Amir Globerson, Tova Milo, and Tomer Wolfson. 2019. Explaining queries over web tables to non-experts. In *Proceedings of the 35th IEEE International Conference on Data Engineering (ICDE)*.

Ben Bogin, Matt Gardner, and Jonathan Berant. 2019. Representing schema structure with graph neural networks for text-to-SQL parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4560–4565.

Giovanni Campagna, Rakesh Ramesh, Silei Xu, Michael Fischer, and Monica S Lam. 2017. Almond: The architecture of an open, crowdsourced, privacy-preserving, programmable virtual assistant. In *Proceedings of the 26th International Conference on World Wide Web*, pages 341–350. International World Wide Web Conferences Steering Committee.

Ohio Supercomputer Center. 1987. Ohio supercomputer center. http://osc.edu/ark:/19495/f5s1ph73.

Shobhit Chaurasia and Raymond J Mooney. 2017. Dialog for language to code. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 175–180.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 33–43.

Li Dong, Chris Quirk, and Mirella Lapata. 2018. Confidence modeling for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 743–753.

Catherine Finegan-Dollak, Jonathan K Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving text-to-SQL evaluation methodology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 351–360.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1321–1330. JMLR. org.

Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. 2019. Towards complex text-to-SQL in cross-domain database with intermediate representation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4524–4535.

Izzeddin Gur, Semih Yavuz, Yu Su, and Xifeng Yan. 2018. DialSQL: Dialogue based structured query generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1339–1349.

David Ha and Jürgen Schmidhuber. 2018. World models. *ArXiv preprint arXiv:1803.10122*.

Luheng He, Julian Michael, Mike Lewis, and Luke Zettlemoyer. 2016. Human-in-the-loop parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2337–2342.

Dan Hendrycks and Kevin Gimpel. 2017. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *Proceedings of International Conference on Learning Representations*.

Wonseok Hwang, Jinyeung Yim, Seunghyun Park, and Minjoon Seo. 2019. A comprehensive exploration on WikiSQL with table-aware word contextualization. *ArXiv preprint arXiv:1902.01069*.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 963–973.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. 2016. Summarizing source code using a neural attention model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2073–2083.

Georgia Koutrika, Alkis Simitsis, and Yannis E Ioannidis. 2010. Explaining structured queries in natural language. In *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*, pages 333–344. IEEE.

Igor Labutov, Bishan Yang, and Tom Mitchell. 2018. Learning to learn semantic parsers from natural language supervision. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1676–1690.

Fei Li and HV Jagadish. 2014. Constructing an interactive natural language interface for relational databases. *Proceedings of the VLDB Endowment*, 8(1):73–84.

Axel-Cyrille Ngonga Ngomo, Lorenz Bühmann, Christina Unger, Jens Lehmann, and Daniel Gerber. 2013. Sorry, I don't speak SPARQL: translating SPARQL queries into natural language. In *Proceedings of the 22nd international conference on World Wide Web*, pages 977–988. ACM.

Aishwarya Padmakumar, Jesse Thomason, and Raymond J Mooney. 2017. Integrated learning of dialog strategies and semantic parsing. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 547–557.

Chris Quirk, Raymond Mooney, and Michel Galley. 2015. Language to code: Learning semantic parsers for If-This-Then-That recipes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 878–888.

Stuart Russell and Peter Norvig. 2009. Artificial intelligence: A modern approach.

Aditya Siddhant and Zachary C Lipton. 2018. Deep Bayesian active learning for natural language processing: Results of a large-scale empirical study. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2904–2909.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Yu Su, Ahmed Hassan Awadallah, Madian Khabsa, Patrick Pantel, Michael Gamon, and Mark Encarnacion. 2017. Building natural language interfaces to Web APIs. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 177–186. ACM.

Yu Su, Ahmed Hassan Awadallah, Miaosen Wang, and Ryen W White. 2018. Natural language interfaces with fine-grained user interaction: A case study on web APIs. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Jesse Thomason, Shiqi Zhang, Raymond J Mooney, and Peter Stone. 2015. Learning to interpret natural language commands through human-robot dialog. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Chenglong Wang, Marc Brockschmidt, and Rishabh Singh. 2018. Pointing out SQL queries from text.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1332–1342.

Yijun Xiao and William Yang Wang. 2019. Quantifying uncertainties in natural language processing tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7322–7329.

Kun Xu, Lingfei Wu, Zhiguo Wang, Yansong Feng, and Vadim Sheinin. 2018. SQL-to-text generation with graph-to-sequence model. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 931–936.

Xiaojun Xu, Chang Liu, and Dawn Song. 2017. SQLNet: Generating structured queries from natural language without reinforcement learning. *ArXiv preprint arXiv:1711.04436*.

Ziyu Yao, Xiujun Li, Jianfeng Gao, Brian Sadler, and Huan Sun. 2019. Interactive semantic parsing for if-then recipes via hierarchical reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 2547–2554.

Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. 2018a. TypeSQL: Knowledge-based type-aware neural text-to-SQL generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 588–594.

Tao Yu, Michihiro Yasunaga, Kai Yang, Rui Zhang, Dongxu Wang, Zifan Li, and Dragomir Radev. 2018b. SyntaxSQLNet: Syntax tree networks for complex and cross-domain text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1653–1663.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 2018c. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating structured queries from natural language using reinforcement learning. *ArXiv preprint arXiv:1709.00103*.

# Modeling Graph Structure in Transformer for Better AMR-to-Text Generation

**Jie Zhu**[1] **Junhui Li**[1*] **Muhua Zhu**[2]
**Longhua Qian**[1] **Min Zhang**[1] **Guodong Zhou**[1]
[1]School of Computer Science and Technology, Soochow University, Suzhou, China
[2]Alibaba Group, Hangzhou, China
zhujie951121@gmail.com, {lijunhui, qianlonghua, minzhang, gdzhou}@suda.edu.cn
muhua.zmh@alibaba-inc.com

## Abstract

Recent studies on AMR-to-text generation often formalize the task as a sequence-to-sequence (seq2seq) learning problem by converting an Abstract Meaning Representation (AMR) graph into a word sequence. Graph structures are further modeled into the seq2seq framework in order to utilize the structural information in the AMR graphs. However, previous approaches only consider the relations between directly connected concepts while ignoring the rich structure in AMR graphs. In this paper we eliminate such a strong limitation and propose a novel structure-aware self-attention approach to better modeling the relations between indirectly connected concepts in the state-of-the-art seq2seq model, i.e., the *Transformer*. In particular, a few different methods are explored to learn structural representations between two concepts. Experimental results on English AMR benchmark datasets show that our approach significantly outperforms the state of the art with 29.66 and 31.82 BLEU scores on LDC2015E86 and LDC2017T10, respectively. To the best of our knowledge, these are the best results achieved so far by supervised models on the benchmarks.

## 1 Introduction

AMR-to-text generation is a task of automatically generating a natural language sentence from an Abstract Meaning Representation (AMR) graph. Due to the importance of AMR as a widely adopted semantic formalism in representing the meaning of a sentence (Banarescu et al., 2013), AMR has become popular in semantic representation and AMR-to-text generation has been drawing more and more attention in the last decade. As the example in Figure 1(a) shows, nodes, such as *he* and *convict-01*, represent semantic concepts and edges, such as ":ARG1" and ":quant", refer to semantic relations between the concepts. Since two concepts close in an AMR graph may map into two segments that are distant in the corresponding sentence, AMR-to-text generation is challenging. For example in Figure 1, the neighboring concepts *he* and *convict-01* correspond to the words *he* and *convicted* which locate at the different ends of the sentence.

To address the above mentioned challenge, recent studies on AMR-to-text generation regard the task as a sequence-to-sequence (seq2seq) learning problem by properly linearizing an AMR graph into a sequence (Konstas et al., 2017). Such an input representation, however, is apt to lose useful structural information due to the removal of reentrant structures for linearization. To better model graph structures, previous studies propose various graph-based seq2seq models to incorporate graphs as an additional input representation (Song et al., 2018; Beck et al., 2018; Damonte and Cohen, 2019). Although such graph-to-sequence models can achieve the state-of-the-art results, they focus on modeling one-hop relations only. That is, they only model concept pairs connected directly by an edge (Song et al., 2018; Beck et al., 2018), and as a result, ignore explicit structural information of indirectly connected concepts in AMR graphs, e.g. the relation between concepts *he* and *possible* in Figure 1.

To make better use of structural information in an AMR graph, we attempt to model arbitrary concept pairs no matter whether directly connected or not. To this end, we extend the encoder in the state-of-the-art seq2seq model, i.e., the Transformer (Vaswani et al., 2017) and propose structure-aware self-attention encoding approach. In particular, several distinct methods are proposed to learn structure representations for the new self-attention mechanism.

*Corresponding Author: Junhui Li.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5459–5468,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

Empirical studies on two English benchmarks show that our approach significantly advances the state of the art for AMR-to-text generation, with the performance improvement of 4.16 BLEU score on LDC2015E86 and 4.39 BLEU score on LDC2017T10 respectively over the strong baseline. Overall, this paper makes the following contributions.

- To the best of our knowledge, this is the first work that applies the Transformer to the task of AMR-to-text generation. On the basis of the Transformer, we build a strong baseline that reaches the state of the art.
- We propose a new self-attention mechanism to incorporate richer structural information in AMR graphs. Experimental results on two benchmarks demonstrate the effectiveness of the proposed approach.
- Benefiting from the strong baseline and the structure-aware self-attention mechanism, we greatly advance the state of the art in the task.

## 2 AMR-to-Text Generation with Graph Structure Modeling

We start by describing the implementation of our baseline system, a state-of-the-art seq2seq model which is originally used for neural machine translation and syntactic parsing (Vaswani et al., 2017). Then we detail the proposed approach to incorporating structural information from AMR graphs.

### 2.1 Transformer-based Baseline

**Transformer**: Our baseline system builds on the Transformer which employs an encoder-decoder framework, consisting of stacked encoder and decoder layers. Each encoder layer has two sublayers: self-attention layer followed by a position-wise feed forward layer. Self-attention layer employs multiple attention heads and the results from each attention head are concatenated and transformed to form the output of the self-attention layer. Each attention head uses scaled dot-product attention which takes a sequence $x = (x_1, \cdots, x_n)$ of $n$ elements as input and computes a new sequence $z = (z_1, \cdots, z_n)$ of the same length:

$$z = Attention\,(x) \quad (1)$$

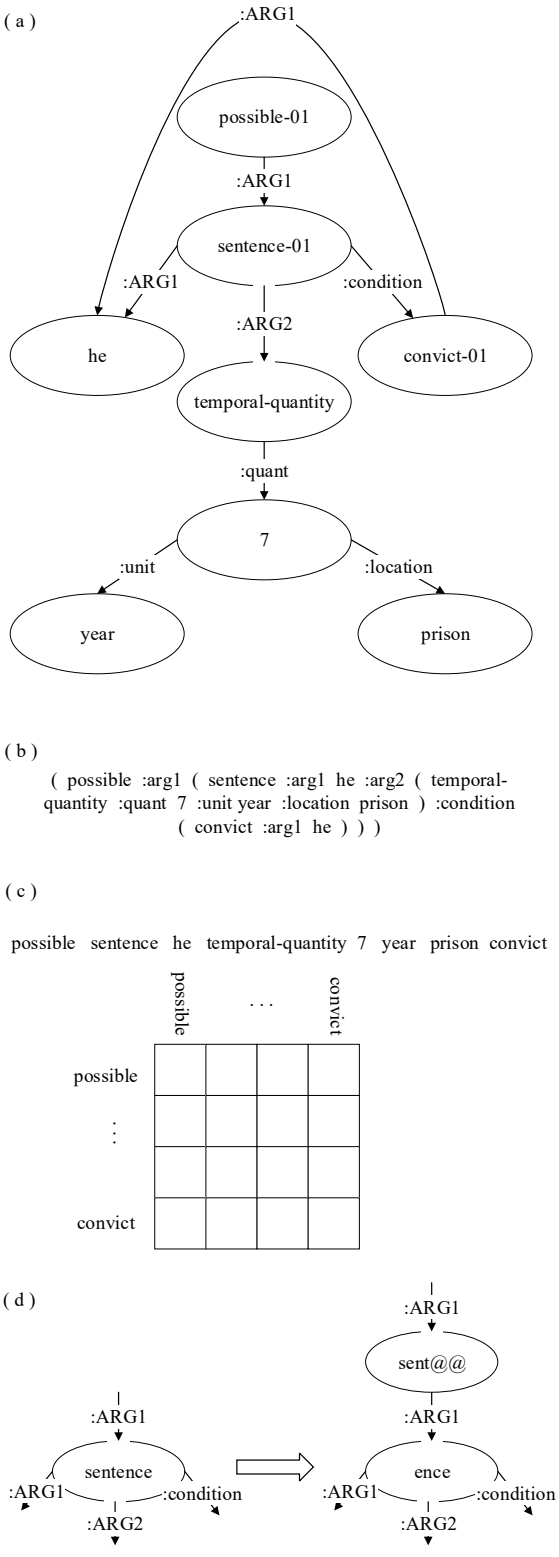


Figure 1: (a) An example of AMR graph for the sentence of *He could be sentenced to 7 years in prison if convicted.* (b) input to our baseline system, the seq2seq Transformer. (c) input to our proposed system based on structure-aware self-attention. (d) An example of graph structure extensions to sub-word units.

where $x_i \in \mathbb{R}^{d_x}$ and $z \in \mathbb{R}^{n \times d_z}$. Each output element $z_i$ is a weighted sum of a linear transformation of input elements:

$$z_i = \sum_{j=1}^{n} \alpha_{ij} \left( x_j W^V \right) \quad (2)$$

where $W^V \in \mathbb{R}^{d_x \times d_z}$ is matrix of parameters. The vectors $\alpha_i = (\alpha_{i1}, \cdots, \alpha_{in})$ in Equation 2 are obtained by the self-attention model, which captures the correspondences between $x_i$ and others. Specifically, the attention weight $\alpha_{ij}$ of each element $x_j$ is computed using a softmax function:

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{n} exp(e_{ik})} \quad (3)$$

where

$$e_{ij} = \frac{\left( x_i W^Q \right) \left( x_j W^K \right)^T}{\sqrt{d_z}} \quad (4)$$

is an alignment function which measures how well the input elements $x_i$ and $x_j$ match. $W^Q, W^K \in \mathbb{R}^{d_x \times d_z}$ are parameters to be learned.

**Input Representation**: We use the depth-first traversal strategy as in Konstas et al. (2017) to linearize AMR graphs and to obtain simplified AMRs. We remove variables, wiki links and sense tags before linearization. Figure 1(b) shows an example linearization result for the AMR graph in Figure 1(a). Note that the reentrant concept *he* in Figure 1 (a) maps to two different tokens in the linearized sequence.

**Vocabulary**: Training AMR-to-text generation systems solely on labeled data may suffer from data sparseness. To attack this problem, previous works adopt techniques like anonymization to remove named entities and rare words (Konstas et al., 2017), or apply a copy mechanism (Gulcehre et al., 2016) such that the models can learn to copy rare words from the input sequence. In this paper we instead use two simple yet effective techniques. One is to apply Byte Pair Encoding (BPE) (Sennrich et al., 2016) to split words into smaller, more frequent sub-word units. The other is to use a shared vocabulary for both source and target sides. Experiments in Section 3.2 demonstrate the necessity of the techniques in building a strong baseline.

### 2.2 Modeling Graph Structures in Transformer

**Input Representation:** We also use the depth-first traversal strategy to linearize AMR graphs and to obtain simplified AMRs which only consist of concepts. As shown in Figure 1 (c), the input sequence is much shorter than the input sequence in the baseline. Besides, we also obtain a matrix which records the graph structure between every concept pair, which implies their semantic relationship (Section2.3).

**Vocabulary**: To be compatible with sub-words, we extend the original AMR graph, if necessary, to include the structures of sub-words. As *sentence-01* in Figure 1(a) is segmented into *sent@@ ence-01*, we split the original node into two connected ones with an edge labeled as the incoming edge of the first unit. Figure 1(d) shows the graph structure for sub-words *sent@@ ence-01*.

**Structure-Aware Self-Attention**: Motivated by Shaw et al. (2018), we extend the conventional self-attention architecture to explicitly encode the relation between an element pair $(x_i, x_j)$ in the alignment model by replacing Equation 4 with Equation 5. Note that the relation $r_{ij} \in \mathbb{R}^{d_z}$ is the vector representation for element pair $(x_i, x_j)$, and will be learned in Section 2.3.

$$e_{ij} = \frac{\left( x_i W^Q \right) \left( x_j W^K + r_{ij} W^R \right)^T}{\sqrt{d_z}} \quad (5)$$

where $W^R \in \mathbb{R}^{d_z \times d_z}$ is a parameter matrix. Then, we update Equation 2 accordingly to propagate structure information to the sublayer output by:

$$z_i = \sum_{j=1}^{n} \alpha_{ij} \left( x_j W^V + r_{ij} W^F \right) \quad (6)$$

where $W^F \in \mathbb{R}^{d_z \times d_z}$ is a parameter matrix.

### 2.3 Learning Graph Structure Representation for Concept Pairs

The above structure-aware self-attention is capable of incorporating graph structure between concept pairs. In this section, we explore a few methods to learn the representation for concept pairs. We use a sequence of edge labels, along the path from $x_i$ to $x_j$ to indicate the AMR graph structure between concepts $x_i$ and $x_j$.[1] In order to distinguish the edge direction, we add a direction symbol to each label with $\uparrow$ for climbing up along the path, and $\downarrow$ for going down. Specifically, for the special case of $i == j$, we use *None* as the path. Table 1 demonstrates structural label sequences between a few concept pairs in Figure 1.

[1]While there may exist two or more paths connecting $x_i$ and $x_j$, we simply choose the shortest one.

| $x_i$ | $x_j$ | Structural label sequence |
|---|---|---|
| he | convict-01 | :ARG1↑ |
| he | 7 | :ARG1↑ :ARG2↓ :quant↓ |
| he | he | None |

Table 1: Examples of structural path between a few concept pairs in Figure 1.

Now, given a structural path with a label sequence $s = s_1, \cdots, s_k$ and its $d_x$-sized corresponding label embedding sequence $l = l_1, \cdots, l_k$, we use the following methods to obtain its representation vector $r$, which maps to $r_{ij}$ in Equation 5 and Equation 6.

**Feature-based**

A natural way to represent the structural path is to view it as a string feature. To this end, we combine the labels in the structural path into a string. Unsurprisingly, this will end up with a large number of features. We keep the most frequent ones (i.e., 20K in our experiments) in the feature vocabulary and map all others into a special feature *UNK*. Each feature in the vocabulary will be mapped into a randomly initialized vector.

**Avg-based**

To overcome the data sparsity in the above feature-based method, we view the structural path as a label sequence. Then we simply use the averaged label embedding as the representation vector of the sequence, i.e.,

$$r = \frac{\sum_{i=1}^{k} l_i}{k} \tag{7}$$

**Sum-based**

Sum-based method simply returns the sum of all label embeddings in the sequence, i.e.,

$$r = \sum_{i=1}^{k} l_i \tag{8}$$

**Self-Attention-based (SA-based for short)**

As shown in Figure 2, given the label sequence $s = s_1, \cdots, s_k$, we first obtain the sequence $e$, whose element is the addition of a word embedding and the corresponding position embedding. Then we use the self-attention, as presented in Eq. 1 to obtain its hidden states $h$, i.e, $h = Attention(e)$, where $h_i \in \mathbb{R}^{d_z}$. Our aim is to encode a variable length sentence into a $d_z$-sized vector. Motivated by (Lin et al., 2017), we achieve this by choosing a linear combination of the $k$ vectors in $h$. Computing the linear combination requires an attention mechanism which takes the whole hidden states $h$ as input, and outputs a vector of weights $\alpha$:

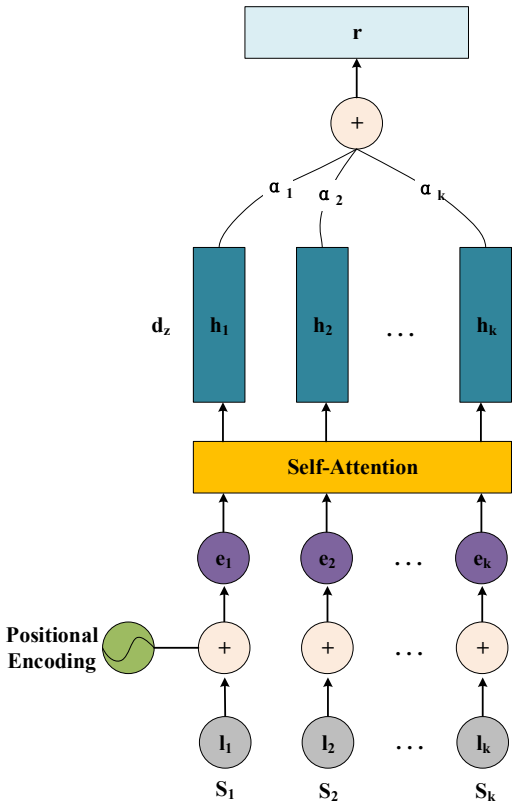


Figure 2: Self-Attention-based method.

$$\alpha = softmax(W^2 \tanh(W^1 h^T)) \tag{9}$$

where $W^1 \in \mathbb{R}^{d_w \times d_z}$ and $W^2 \in \mathbb{R}^{d_w}$. Then the label sequence representation vector is the weighted sum of its hidden states:

$$r = \sum_{i=1}^{k} \alpha_i h_i \tag{10}$$

**CNN-based**

Motivated by (Kalchbrenner et al., 2014), we use convolutional neural network (CNN) to convolute the label sequence $l$ into a vector $r$, as follow:

$$\begin{aligned} conv = Conv1D(&kernel\_size = (m), \\ &strides = 1, \\ &filters = d_z, \\ &input\_shape = d_z \\ &activation =' relu') \end{aligned} \tag{11}$$

$$r = conv\,(l) \tag{12}$$

where kernel size $m$ is set to 4 in our experiments.

## 3 Experimentation

### 3.1 Experimental Settings

For evaluation of our approach, we use the sentences annotated with AMRs from the LDC release LDC2015E86 and LDC2017T10. The two datasets contain 16,833 and 36,521 training AMRs, respectively, and share 1,368 development AMRs and 1,371 testing AMRs. We segment words into sub-word units by BPE (Sennrich et al., 2016) with 10K operations on LDC2015E86 and 20K operations on LDC2017T10.

For efficiently learning graph structure representation for concept pairs (except the feature-based method), we limit the maximum label sequence length to 4 and ignore the labels exceeding the maximum. In SA-based method, we set the filter size $d_w$ as 128.

We use *OpenNMT* (Klein et al., 2017) as the implementation of the Transformer seq2seq model.[2] In parameter setting, we set the number of layers in both the encoder and decoder to 6. For optimization we use Adam with $\beta 1 = 0.1$ (Kingma and Ba, 2015). The number of heads is set to 8. In addition, we set the embedding and the hidden sizes to 512 and the batch token-size to 4096. Accordingly, the $d_x$ and $d_z$ in Section 2 are 64. In all experiments, we train the models for 300K steps on a single K40 GPU.

For performance evaluation, we use BLEU (Papineni et al., 2002), Meteor (Banerjee and Lavie, 2005; Denkowski and Lavie, 2014), and CHRF++ (Popovi, 2017) as metrics. We report results of single models that are tuned on the development set.

We make our code available at `https://github.com/Amazing-J/structural-transformer`.

### 3.2 Experimental Results

We first show the performance of our baseline system. As mentioned before, BPE and sharing vocabulary are two techniques we applied to relieving data sparsity. Table 2 presents the results of the ablation test on the development set of LDC2015E86 by either removing BPE, or vocabulary sharing, or both of them from the baseline system. From the results we can see that BPE and vocabulary sharing are critical to building our baseline system (an improvement from 18.77 to 24.93 in BLEU), revealing the fact that they are two effective ways to address the issue of data sparseness for AMR-to-text generation.

| Model | BLEU | Meteor | CHRF++ |
|---|---|---|---|
| Baseline | 24.93 | 33.20 | 60.30 |
| -BPE | 23.02 | 31.60 | 58.09 |
| -Share Vocab. | 23.24 | 31.78 | 58.43 |
| -Both | 18.77 | 28.04 | 51.88 |

Table 2: Ablation results of our baseline system on the LDC2015E86 development set.

Table 3 presents the comparison of our approach and related works on the test sets of LDC2015E86 and LDC2017T10. From the results we can see that the Transformer-based baseline outperforms most of graph-to-sequence models and is comparable with the latest work by Guo et al. (2019). The strong performance of the baseline is attributed to the capability of the Transformer to encode global and implicit structural information in AMR graphs. By comparing the five methods of learning graph structure representations, we have the following observations.

- All of them achieve significant improvements over the baseline: the biggest improvements are 4.16 and 4.39 BLEU scores on LDC2015E86 and LDC2017T10, respectively.
- Methods using continuous representations (such as SA-based and CNN-based) outperform the methods using discrete representations (such as feature-based).
- Compared to the baseline, the methods have very limited affect on the sizes of model parameters (see the column of *#P (M)* in Table 3).

Finally, our best-performing models are the best among all the single and supervised models.

## 4 Analysis

In this section, we use LDC2017T10 as our benchmark dataset to demonstrate how our proposed approach achieves higher performance than the baseline. As representative, we use CNN-based method to obtain structural representation.

[2] `https://github.com/OpenNMT/OpenNMT-py`

| System | | LDC2015E86 | | | | LDC2017T10 | | |
|---|---|---|---|---|---|---|---|---|
| | | BLEU | Meteor | CHRF++ | #P (M) | BLEU | Meteor | CHRF++ |
| Baseline | | 25.50 | 33.16 | 59.88 | 49.1 | 27.43 | 34.62 | 61.85 |
| **Our Approach** | feature-based | 27.23 | 34.53 | 61.55 | 49.4 | 30.18 | 35.83 | 63.20 |
| | avg-based | 28.37 | 35.10 | 62.29 | 49.1 | 29.56 | 35.24 | 62.86 |
| | sum-based | 28.69 | 34.97 | 62.05 | 49.1 | 29.92 | 35.68 | 63.04 |
| | SA-based | **29.66** | **35.45** | **63.00** | 49.3 | 31.54 | 36.02 | 63.84 |
| | CNN-based | 29.10 | 35.00 | 62.10 | 49.2 | **31.82** | **36.38** | **64.05** |
| Previous works with single models | | | | | | | | |
| Konstas et al. (2017)* | | 22.00 | - | - | - | - | - | - |
| Cao and Clark (2019)* | | 23.5 | - | - | - | 26.8 | - | - |
| Song et al. (2018)† | | 23.30 | - | - | - | - | - | - |
| Beck et al. (2018)† | | - | - | - | - | 23.3 | - | 50.4 |
| Damonte and Cohen (2019)† | | 24.40 | 23.60 | - | - | 24.54 | 24.07 | - |
| Guo et al. (2019)† | | 25.7 | - | - | - | 27.6 | - | 57.3 |
| Song et al. (2016)‡ | | 22.44 | - | - | - | - | - | - |
| Previous works with either ensemble models or unlabelled data, or both | | | | | | | | |
| Konstas et al. (2017)* | | 33.8 | - | - | - | - | - | - |
| Song et al. (2018)† | | 33.0 | - | - | - | - | - | - |
| Beck et al. (2018)† | | - | - | - | - | 27.5 | - | 53.5 |
| Guo et al. (2019)† | | 35.3 | - | - | - | - | - | - |

Table 3: Comparison results of our approaches and related studies on the test sets of LDC2015E86 and LDC2017T10. #P indicates the size of parameters in millions. * indicates seq2seq-based systems while † for graph-based models, and ‡ for other models. All our proposed systems are significant over the baseline at 0.01, tested by bootstrap resampling (Koehn, 2004).

| System | BLEU |
|---|---|
| Baseline | 27.43 |
| Our approach | 31.82 |
| No indirectly connected concept pairs | 29.92 |

Table 4: Performance on the test set of our approach with or without modeling structural information of indirectly connected concept pairs.

### 4.1 Effect of Modeling Structural Information of Indirectly Connected Concept Pairs

Our approach is capable of modeling arbitrary concept pairs no matter whether directly connected or not. To investigate the effect of modeling structural information of indirectly connected concept pairs, we ignore their structural information by mapping all structural label sequences between two indirectly connected concept pairs into *None*. In this way, the structural label sequence for *he* and *7* in Table 1, for example, will be *None*.

Table 4 compares the performance of our approach with or without modeling structural information of indirectly connected concept pairs. It shows that by modeling structural information of indirectly connected concept pairs, our approach improves the performance on the test set from 29.92 to 31.82 in BLEU scores. It also shows that even without modeling structural information of indirectly connected concept pairs, our approach achieves better performance than the baseline.

### 4.2 Effect on AMR Graphs with Different Sizes of Reentrancies

Linearizing an AMR graph into a sequence unavoidably loses information about reentrancies (nodes with multiple parents). This poses a challenge for the baseline since there exists on obvious sign that the first *he* and the second *he*, as shown in Figure 1 (b), refer to the same person. By contrast, our approach models reentrancies explicitly. Therefore, it is expected that the benefit of our approach is more evident for those AMR graphs containing more reentrancies. To test this hypothesis, we partition source AMR graphs to different groups by their numbers of reentrancies and evaluate their performance respectively. As shown in Figure 3, the performance gap be-

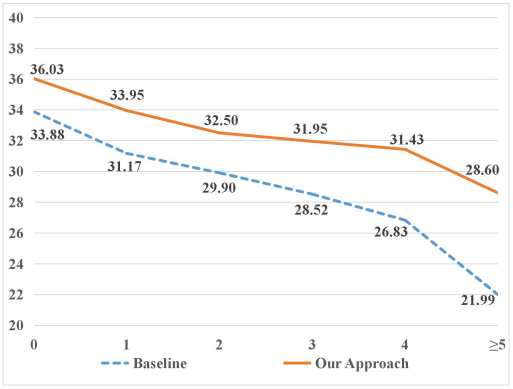


Figure 3: Performance (in BLEU) on the test set with respect to the reentrancy numbers of the input AMR graphs.

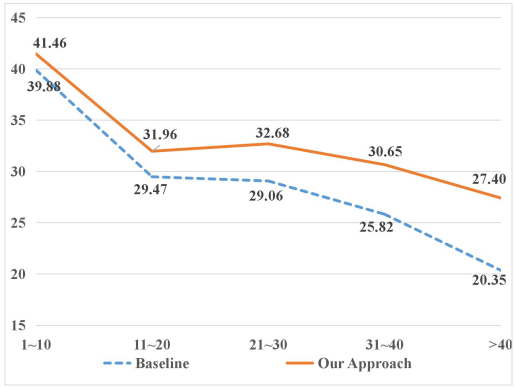


Figure 4: Performance (in BLEU) on the test set with respect to the size of the input AMR graphs.

tween our approach and the baseline goes widest for AMR graphs with more than 5 reentrancies, on which our approach outperforms the baseline by 6.61 BLEU scores.

### 4.3 Effect on AMR Graphs with Different Sizes

When we encode an AMR graph with plenty concepts, linearizing it into a sequence tends to lose great amount of structural information. In order to test the hypothesis that graphs with more concepts contribute more to the improvement, we partition source AMR graphs to different groups by their sizes (i.e., numbers of concepts) and evaluate their performance respectively. Figure 4 shows the results which indicate that modeling graph structures significantly outperforms the baseline over all AMR lengths. We also observe that the performance gap between the baseline and our approach increases when AMR graphs become big, revealing that the baseline seq2seq model is far from capturing deep structural details of big AMR graphs. Figure 4 also indicates that text generation becomes difficult for big AMR graphs. We think that the low performance on big AMR graphs is mainly attributed to two reasons:

- Big AMR graphs are usually mapped into long sentences while seq2seq model tends to stop early for long inputs. As a result, the length ratio[3] for AMRs with more than 40 concepts is 0.906, much lower than that for AMRs with less concepts.
- Big AMR graphs are more likely to have reentrancies, which makes the generation more challenging.

[3]Length ratio is the length of generation output, divided by the length of reference.

### 4.4 Case Study

In order to better understand the model performance, Figure 5 presents a few examples studied in Song et al. (2018) (Example (1)) and Damonte and Cohen (2019) (Examples (2) - (5)).

In Example (1), though our baseline recovers a propositional phrase for the noun *staff* and another one for the noun *funding*, it fails to recognize the anaphora and antecedent relation between the two propositional phrases. In contrast, our approach successfully recognizes *:prep-for c* as a reentrancy node and generates one propositional phrase shared by both nouns *staff* and *funding*. In Example (2), we note that although AMR graphs lack tense information, the baseline generates output with inconsistent tense (i.e., *do* and *found*) while our approach consistently prefers past tense for the two clauses. In Example (3), only our approach correctly uses *people* as the subject of the predicate *can*. In Example (4), the baseline fails to predict the direct object *you* for predicate *recommend*. Finally in Example (5), the baseline fails to recognize the subject-predicate relation between noun *communicate* and verb *need*. Overall, we note that compared to the baseline, our approach produces more accurate output and deal with reentrancies more properly.

Comparing the generation of our approach and graph-based models in Song et al. (2018) and Damonte and Cohen (2019), we observe that our generation is more close to the reference in sentence structure. Due to the absence of tense informa-

(1) (p/ provide-01
:ARG0 (a / agree-01)
:ARG1 (a2 / and
:op1 (s / staff
:prep-for (c / center
:mod (r / research-01)))
:op2 (f / fund-01
:prep-for c)))
**REF:** the agreement will provide staff and funding for the research center .
**SEQ1:** agreed to provide research and institutes in the center .
**G2S:** the agreement provides the staff of the research center and the funding .
**Baseline:** the agreement provides staff for research centres and funding for center .
**Our approach:** the agreement provided staff and funding for research centers .

(2) **REF:** i dont tell him but he finds out ,
**SEQ2:** i did n't tell him but he was out .
**GRAPH:** i do n't tell him but he found out .
**Baseline:** i do n't tell him but he found out .
**Our approach:** i did n't tell him but he found out .

(3) **REF:** if you tell people they can help you ,
**SEQ2:** if you tell him , you can help you !
**GRAPH:** if you tell them , you can help you .
**Baseline:** if you tell people , you might help you !
**Our approach:** if you tell people , people can help you !

(4) **REF:** i 'd recommend you go and see your doctor too .
**SEQ2:** i recommend you go to see your doctor who is going to see your doctor .
**GRAPH:** i recommend you going to see your doctor too .
**Baseline:** i would recommend going to see your doctor too .
**Our approach:** i would recommend you to go to see your doctor too .

(5) **REF:** tell your ex that all communication needs to go through the lawyer .
**SEQ2:** tell that all the communication go through lawyer .
**GRAPH:** tell your ex the need to go through a lawyer .
**Baseline:** tell your ex you need to go all the communication by lawyer .
**Our approach:** tell your ex that all communication needs to go through lawyers .

Figure 5: Examples of generation from AMR graphs. (1) is from Song et al. (2018), (2) - (5) are from Damonte and Cohen (2019). REF is the reference sentence. SEQ1 and G2S are the outputs of the seq2seq and the graph2seq models in Song et al. (2018), respectively. SEQ2 and GRAPH are the outputs of the seq2seq and the graph models in Damonte and Cohen (2019), respectively.

tion in AMR graphs, our model tends to use past tense, as *provided* and *did* in Example (1) and (2). Similarly, without information concerning singular form and plural form, our model is more likely to use plural nouns, as *centers* and *lawyers* in Example (1) and (5).

## 5 Related Work

Most studies in AMR-to-text generation regard it as a translation problem and are motivated by the recent advances in both statistical machine translation (SMT) and neural machine translation (NMT). Flanigan et al. (2016) first transform an AMR graph into a tree, then specify a number of tree-to-string transduction rules based on alignments that are used to drive a tree-based SMT model (Graehl and Knight, 2004). Pourdamghani et al. (2016) develop a method that learns to linearize AMR graphs into AMR strings, and then feed them into a phrase-based SMT model (Koehn et al., 2003). Song et al. (2017) use synchronous node replacement grammar (SNRG) to generate text. Different from synchronous context-free grammar in hierarchical phrase-based SMT (Chiang, 2007), SNRG is a grammar over graphs.

Moving to neural seq2seq approaches, Konstas et al. (2017) successfully apply seq2seq model together with large-scale unlabeled data for both text-to-AMR parsing and AMR-to-text generation. With special interest in the target side syn-

tax, Cao and Clark (2019) use seq2seq models to generate target syntactic structure, and then the surface form. To prevent the information loss in linearizing AMR graphs into sequences, (Song et al., 2018; Beck et al., 2018) propose graph-to-sequence models to encode graph structure directly. Focusing on reentrancies, Damonte and Cohen (2019) propose stacking encoders which consist of BiLSTM (Graves et al., 2013), TreeLSTMs (Tai et al., 2015), and Graph Convolutional Network (GCN) (Duvenaud et al., 2015; Kipf and Welling, 2016). Guo et al. (2019) propose densely connected GCN to better capture both local and non-local features. However, all the aforementioned graph-based models only consider the relations between nodes that are directly connected, thus lose the structural information between nodes that are indirectly connected via an edge path.

Recent studies also extend the Transformer to encode structural information for other NLP applications. Shaw et al. (2018) propose relation-aware self-attention to capture relative positions of word pairs for neural machine translation. Ge et al. (2019) extend the relation-aware self-attention to capture syntactic and semantic structures. Our model is inspired by theirs but aims to encode structural label sequences of concept pairs. Koncel-Kedziorski et al. (2019) propose graph Transformer to encode graph structure. Similar to the GCN, it focuses on the relations between directly connected nodes.

## 6 Conclusion and Future Work

In this paper we proposed a structure-aware self-attention for the task of AMR-to-text generation. The major idea of our approach is to encode long-distance relations between concepts in AMR graphs into the self-attention encoder in the Transformer. In the setting of supervised learning, our models achieved the best experimental results ever reported on two English benchmarks.

Previous studies have shown the effectiveness of using large-scale unlabelled data. In future work, we would like to do semi-supervised learning and use silver data to test how much improvements could be further achieved.

## Acknowledgments

We thank the anonymous reviewers for their insightful comments and suggestions. We are grateful to Linfeng Song for fruitful discussions. This work is supported by the National Natural Science Foundation of China (Grant No. 61876120, 61673290, 61525205), and the Priority Academic Program Development of Jiangsu Higher Education Institutions.

## References

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of 7th Linguistic Annotation Workshop & Interoperability with Discourse*, pages 178–186.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of ACL*, pages 65–72.

Daniel Beck, Gholamreza Haffari, and Trevor Cohn. 2018. Graph-to-sequence learning using gated graph neural networks. In *Proceedings of ACL*, pages 273–283.

Kris Cao and Stephen Clark. 2019. Factorising amr generation through syntax. In *Proceedings of NAACL*, pages 2157–2163.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33:201–228.

Marco Damonte and Shay B. Cohen. 2019. Structural neural encoders for AMR-to-text generation. In *Proceedings of NAACL*, pages 3649–3658.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of WMT*, pages 376–380.

David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timonthy Hirzel, Alan Aspuru-Guzik, and Ryan P Adams. 2015. Convolutional networks on graphs for learning molecular fingerprints. In *Proceedings of NIPS*, pages 2224–2232.

Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime Carbonell. 2016. Generation from abstract meaning representation using tree transducers. In *Proceedings of NAACL*, pages 731–739.

DongLai Ge, Junhui Li, Muhua Zhu, and Shoushan Li. 2019. Modeling source syntax and semantics for neural amr parsing. In *Proceedings of IJCAI*, pages 4975–4981.

Jonathan Graehl and Kevin Knight. 2004. Training tree transducers. In *Proceedings of NAACL*, pages 105–112.

Alex Graves, Abdel rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Proceedings of ICASSP*, pages 6645–6649.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of ACL*, pages 140–149.

Zhijiang Guo, Yan Zhang, Zhiyang Teng, and Wei Lu. 2019. Densely connected graph convolutional networks for graph-to-sequence learning. *Transactions of the Association of Computational Linguistics*, 7:297–312.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of ACL*, pages 655–665.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. In *Proceedings of ICLR*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. In *Proceedings of ACL, System Demonstrations*, pages 67–72.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP*, pages 388–395.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of NAACL*, pages 127–133.

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text generation from knowledge graphs with graph transformers. In *Proceedings of NAACL*, pages 2284–2293.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: Sequence-to-sequence models for parsing and generation. In *Proceedings of ACL*, pages 146–157.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *Proceedings of ICLR*.

Kishore Papineni, Salim Roukos, Ward Todd, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*, pages 311–318.

Maja Popovi. 2017. chrf++: words helping character n-grams. In *Proceedings of WMT*, pages 612–618.

Nima Pourdamghani, Kevin Knight, and Ulf Hermjakob. 2016. Generating english from abstract meaning representations. In *Proceedings of the 9th International Natural Language Generation conference*, pages 21–25.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of ACL*, pages 1715–1725.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of NAACL*, pages 464–468.

Linfeng Song, Xiaochang Peng, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2017. Amr-to-text generation with synchronous node replacement grammar. In *Proceedings of ACL*, pages 7–13.

Linfeng Song, Yue Zhang, Xiaochang Peng, Zhiguo Wang, and Daniel Gildea. 2016. Amr-to-text generation as a traveling salesman problem. In *Proceedings of EMNLP*, pages 2084–2089.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. A graph-to-sequence model for AMR-to-text generation. In *Proceedings of ACL*, pages 1616–1626.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of ACL*, pages 1556–1566.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N.Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of NIPS*, pages 5998–6008.

# Syntax-Aware Aspect Level Sentiment Classification with Graph Attention Networks

**Binxuan Huang**
School of Computer Science
Carnegie Mellon University
binxuanh@cs.cmu.edu

**Kathleen M. Carley**
School of Computer Science
Carnegie Mellon University
kathleen.carley@cs.cmu.edu

## Abstract

Aspect level sentiment classication aims to identify the sentiment expressed towards an aspect given a context sentence. Previous neural network based methods largely ignore the syntax structure in one sentence. In this paper, we propose a novel target-dependent graph attention network (TD-GAT) for aspect level sentiment classification, which explicitly utilizes the dependency relationship among words. Using the dependency graph, it propagates sentiment features directly from the syntactic context of an aspect target. In our experiments, we show our method outperforms multiple baselines with GloVe embeddings. We also demonstrate that using BERT representations further substantially boosts the performance.

## 1 Introduction

Aspect level sentiment classification aims to identify the sentiment polarity (eg. positive, negative, neutral) of an aspect target in its context sentence. Compared to sentence-level sentiment classification, which tries to detect the overall sentiment in a sentence, it is a more fine-grained task. Aspect level sentiment classification can distinguish sentiment polarity for multiple aspects in a sentence with various sentiment polarity, while sentence-level sentiment classification often fails in these conditions (Jiang et al., 2011). For example, in a sentence "great food but the service was dreadful", the sentiment polarity for aspects "food" and "service" are positive and negative respectively. In this case, however, it is hard to determine the overall sentiment since the sentence is mixed with positive and negative expressions.

Typically, researchers use machine learning algorithms to classify the sentiment of given aspects in sentences. Some early work manually designs features, eg. sentiment lexicons and linguistic features, to train classifiers for aspect level sentiment classification (Jiang et al., 2011; Wagner et al., 2014). Later, various neural network-based methods became popular for this task (Tang et al., 2016b; Wang et al., 2016), as they do not require manual feature engineering. Most of them are based on long short-term memory (LSTM) neural networks (Tang et al., 2016a; Huang et al., 2018) and few of them use convolutional neural networks (CNN) (Huang and Carley, 2018; Xue and Li, 2018).

Most of these neural network based methods treat a sentence as a word sequence and embed aspect information into the sentence representation via various methods, eg. attention (Wang et al., 2016) and gate (Huang and Carley, 2018). These methods largely ignore the syntactic structure of the sentence, which would be beneficial to identify sentiment features directly related to the aspect target. When an aspect term is separated away from its sentiment phrase, it is hard to find the associated sentiment words in a sequence. For example, in a sentence "The food, though served with bad service, is actually great", the word "great" is much closer to the aspect "food" in the dependency graph than in the word sequence. Using the dependency relationship is also helpful to resolve potential ambiguity in a word sequence. In a simple sentence "Good food bad service", "good" and "bad" can be used interchangeably. Using an attention-based method, it is hard to distinguish which word is associated with "food" or "service" among "good" and "bad". However, a human reader with good grammar knowledge can easily recognize that "good" is an adjectival modifier for "food" while "bad" is the modifier for "service".

In this paper, we propose a novel neural network framework named target-dependent graph attention network (TD-GAT), which leverages the syn-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5469–5477,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

tax structure of a sentence for aspect level sentiment classification. Unlike these previous methods, our approach represents a sentence as a dependency graph instead of a word sequence. In the dependency graph, the aspect target and related words will be connected directly. We employ a multi-layer graph attention network to propagate sentiment features from important syntax neighbourhood words to the aspect target. We further incorporate an LSTM unit in TD-GAT to explicitly capture aspect related information across layers during recursive neighbourhood expansion. Though some work tries to incorporate syntax knowledge using recursive neural networks (Dong et al., 2014), it has to convert the original dependency tree into a binary tree, which may move syntax related words away from the aspect term. Compared to (Dong et al., 2014), one advantage of our approach is that it keeps the original syntax order unchanged.

We apply the proposed method to laptop and restaurant datasets from SemEval 2014 (Pontiki et al., 2014). Our experiments show that our approach outperforms multiple baselines with GloVe embeddings (Pennington et al., 2014). We further demonstrate that using BERT representations (Devlin et al., 2018) boosts the performance a lot. In our analysis, we show that our model is lightweight in terms of model size. It achieves better performance and requires fewer computational resources and less running time than fine-tuning the original BERT model.

## 2 Related Work

Aspect level sentiment classification is a branch of sentiment analysis (Pang et al., 2008). The goal of this task is to identify the sentiment polarity of an aspect target within a given context sentence.

Some early work first converts an extensive set of features, eg. sentiment lexicons and parse context, into feature vectors, then train a classifier like support vector machine (SVM) based on these feature vectors. Wagner et al. (2014) combine sentiment lexicons, distance to the aspect target, and dependency path distance to train an SVM classifier. Kiritchenko et al. (2014) also propose a similar method and they show that adding parse context features could improve the prediction accuracy by more than one percent.

Later, many neural network based methods are introduced to approach this aspect level sentiment classification task. A majority of the work uses LSTM neural networks to model the word sequence in a sentence. Tang et al. (2016a) use two LSTMs to model the left and right contexts of an aspect target, then they take two final hidden states as classification features. Wang et al. (2016) introduce the attention mechanism into this task (Bahdanau et al., 2014). They use the aspect term embedding to generate an attention vector to select important aspect-related words in a sentence. Following this work, Huang et al. (2018) use two LSTM networks to model sentences and aspects in a joint way and explicitly capture the interaction between aspects and context sentences. From the sentence aspect correlation matrix, they find important words in aspects as well as in sentences. Li et al. (2018) further improve these attention-based methods by incorporating position information.

Except for these LSTM-based methods, there are some other neural methods existing in the literature. Tang et al. (2016b) propose a deep memory network which consists of multiple computation layers and each layer computes an attention vector over an external memory. There are also some attempts using convolutional neural networks (CNN) to approach this task (Huang and Carley, 2018; Xue and Li, 2018). Features generated from the aspect are used to control the information flow in the CNN applied to the sentence (Huang et al., 2018). Benefited from the rich linguistic knowledge learned from massive language modeling (Devlin et al., 2018), researchers show great progress on this task using BERT representations (Sun et al., 2019). Xu et al. (2019) utilizes additional in-domain datasets to post-train BERT's weights and then fine-tune it on this task. However, such a method requires a large corpus for post-training and the fine-tuning also takes a lot of computation resources and time.

Unlike previous discussed neural network-based methods, our approach explicitly utilizes the syntax structure among one sentence and these sentiment features are propagated towards the aspect target on the dependency graph instead of on the original word sequence. Some early work also tries to leverage the syntax structure (Dong et al., 2014; Nguyen and Shirai, 2015). They have to convert the original dependency tree into a binary tree first and place the aspect target at the root node. As a result, sentiment features can be propagated recursively from the leaf nodes to the root

node. However, such conversion may move modifying sentiment words farther away from the aspect target, while our approach keeps the original syntax order unchanged.

## 3 Method

### 3.1 Text Representation

Given a sentence $s = [w_1, w_2, ..., w_i, .., w_n]$ with length n and an aspect target $w_i$, we first map each word into a low-dimensional word embedding vector. For each word $w_i$, we get one vector $x_i \in R^d$ where $d$ is the dimension of the word embedding space.

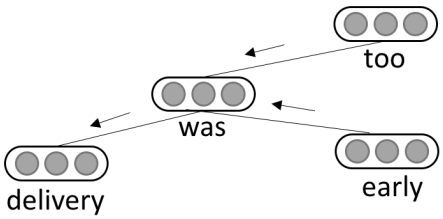


Figure 1: The dependency graph for "delivery was early too". Features can be propagated from neighbour nodes to the aspect node "delivery".

We transform the sentence into a dependency graph using an off-the-shelf dependency parser (Chen and Manning, 2014). Each node represents a word and is associated with an embedding vector as its local feature vector. An undirected edge between two words means these two words are syntactically related. In Figure 1, we show an example of the dependency graph for a sentence "delivery was early too". For the target "delivery", we can propagate features from its 2-hop neighbourhood to 1-hop neighbourhood and then to itself.

For an aspect target with more than one word, we first replace the whole target word sequence with a special symbol "__target__", then pass the modified sentence into the dependency parser. As a result, there is a meta-node representing the target sequence in the dependency graph and its local feature vector is the average of embedding vectors of words in the target.

### 3.2 Graph Attention Network

A graph attention network (GAT) (Veličković et al., 2017) is a variant of graph neural network (Scarselli et al., 2009) and is a key element in our method. It propagates features from an aspect's syntax context to the aspect node. Given a dependency graph with $N$ nodes, where each node is associated with a local word embedding vector $x$, one GAT layer compute node representations by aggregating neighbourhood's hidden states. With an $L$-layer GAT network, features from $L$ hops away can be propagated to the aspect target node.

Specifically, given a node $i$ with a hidden state $h_l^i$ at layer $l$ and the node's neighbours $n[i]$ as well as their hidden states, a GAT updates the node's hidden state at layer $l+1$ using multi-head attentions (Vaswani et al., 2017). The update process is as follows

$$h_{l+1}^i = \Big\Vert_{k=1}^{K} \sigma\Big(\sum_{j \in n[i]} \alpha_{lk}^{ij} W_{lk} h_l^j\Big) \quad (1)$$

$$\alpha_{lk}^{ij} = \frac{exp(f(a_{lk}^T [W_{lk} h_l^i || W_{lk} h_l^j]))}{\sum_{u \in n[i]} exp(f(a_{lk}^T [W_{lk} h_l^i || W_{lk} h_l^u]))} \quad (2)$$

where $||$ represents vector concatenation, $\alpha_{lk}^{ij}$ is the attention coefficient of node $i$ to its neighbour $j$ in attention head $k$ at layer $l$. $W_{lk} \in R^{\frac{D}{K} \times D}$ is a linear transformation matrix for input states. $D$ is the dimension of hidden states. $\sigma$ denotes a sigmoid function. $f(\cdot)$ is a LeakyReLU non-linear function (Maas et al., 2013). $a_{lk} \in R^{\frac{2D}{K}}$ is an attention context vector learned during training.

For simplicity, we can write such feature propagation process as

$$H_{l+1} = GAT(H_l, A; \Theta_l) \quad (3)$$

where $H_l \in R^{N \times D}$ is the stacked states for all nodes at layer $l$, $A \in R^{N \times N}$ is the graph adjacent matrix. $\Theta_l$ is the parameter set of the GAT at layer $l$.

### 3.3 Target-Dependent Graph Attention Network

To utilize the target information in such a GAT network explicitly, we further use an LSTM to model the dependency for the aspect target across layers, which is also helpful for overcoming noisy information in a graph (Huang and Carley, 2019). The basic idea is that at layer 0 the hidden state for an aspect target node $h_0^t$ is only dependent on the target's local features and at each layer $l$ information related with the target from $l$-hop neighbourhood is added into the hidden state by the LSTM unit.

Given the previous hidden state $h_{l-1}^t$ and cell state $c_{l-1}^t$ for any target node $t$, we first get a temporary hidden state $\hat{h}_l^t$ by aggregating its neighbour information using equation 1. Then we take this temporary hidden state as a new observation for an LSTM unit and update the hidden state at layer $l$ as follows:

$$i_l = \sigma(W_i \hat{h}_l^t + U_i h_{l-1} + b_i) \quad (4)$$

$$f_l = \sigma(W_f \hat{h}_l^t + U_f h_{l-1} + b_f) \quad (5)$$

$$o_l = \sigma(W_o \hat{h}_l^t + U_o h_{l-1} + b_o) \quad (6)$$

$$\hat{c}_l = tanh(W_c \hat{h}_l^t + U_c h_{l-1} + b_c) \quad (7)$$

$$c_l = f_l \circ c_{l-1} + i_l \circ \hat{c}_l \quad (8)$$

$$h_l = o_l \circ tanh(c_l) \quad (9)$$

where $\sigma(\cdot)$ and $tanh(\cdot)$ are the sigmoid function and hyperbolic tangent function respectively. $W_i, U_i, W_f, U_f, W_o, U_o, W_c, U_c$ are parameter matrices and $b_i, b_f, b_o, b_c$ are bias vectors to be learned during training. Symbol $\circ$ represents element-wise multiplication. $i_t$, $f_t$ and $o_t$ are input gate, forget gate and output gate, which control the information flow.

In summary, the feed-forward process of our target-dependent graph neural network can be written as

$$H_{l+1}, C_{l+1} = LSTM(GAT(H_l, A; \Theta_l), (H_l, C_l))$$
$$H_0, C_0 = LSTM(XW_p + [b_p]_N, (0, 0))$$

where $C_l$ is the stacked cell states of the LSTM at layer $l$. The initial hidden state and cell state of the LSTM are set as 0. $W_p \in R^{d \times D}$ is the projection matrix that transforms stacked embedding vectors $X$ into the dimension of hidden states and $[b_p]_N$ represents stacking the bias vector $b_p$ N times and forms a bias matrix with dimension $R^{N \times D}$. Similarly, we can also replace the LSTM unit with a GRU unit to model the layer-wise dependency for the target.

### 3.4 Final Classification

With $L$ layers of our TD-GAT networks, we get a final representation for our aspect target node. We just retrieve the corresponding hidden state $h_L^t$ for the aspect target node from all the node representations $H_L$.

We map the hidden state $h_L^t$ into the classification space by a linear transformation. Afterwards, the probability of a sentiment class $c$ is computed by a softmax function:

$$P(y = c) = \frac{exp(W h_L^t + b)_c}{\sum_{i \in C} exp(W h_L^t + b)_i} \quad (10)$$

where $W, b$ are the weight matrix and bias for the linear transformation, $C$ is the set of sentiment classes.

The final predicted sentiment polarity of an aspect target is the label with the highest probability. We minimize the cross-entropy loss with $L_2$ regularization to train our model

$$loss = -\sum_{c \in C} I(y = c) \cdot log(P(y = c)) + \lambda ||\Theta||^2$$

where $I(\cdot)$ is an indicator function. $\lambda$ is the $L_2$ regularization parameter and $\Theta$ is the set of all the parameters in our model.

## 4 Experiments

### 4.1 Datasets

We adopt two widely used datasets from SemEval 2014 Task 4 (Pontiki et al., 2014) to validate the effectiveness of our method. These are two domain-specific datasets collected from laptop and restaurant reviews. Each data point is a pair of a sentence and an aspect term. Experienced annotators tagged each pair with sentiment polarity. Following recent work (Tay et al., 2018; Huang and Carley, 2018), we first take 500 training instances as development set[1] to tune our model. We then combine the development dataset and training dataset to train our final model. Statistics of these two datasets are given in Table 1.

| Dataset | Positive | Neutral | Negative |
|---|---|---|---|
| Laptop-Train | 767 | 373 | 673 |
| Laptop-Dev | 220 | 87 | 193 |
| Laptop-Test | 341 | 169 | 128 |
| Restaurant-Train | 1886 | 531 | 685 |
| Restaurant-Dev | 278 | 102 | 120 |
| Restaurant-Test | 728 | 196 | 196 |

Table 1: Statistics of the datasets.

### 4.2 Implementation Details

We use the Stanford neural parser (Chen and Manning, 2014) to get dependency graphs. We

[1]The splits can be found at https://github.com/vanzytay/ABSA_DevSplits.

try two embedding methods in this paper. One is 300-dimensional GloVe embeddings (Pennington et al., 2014), where we just retrieve the corresponding embedding vector for each token in graphs. Another is BERT representations (Devlin et al., 2018), where we use the large uncased English model with dimension 1024 implemented in PyTorch [2]. The input of the BERT model is a text pair formatted as "[CLS]" + sentence + "[SEP]" + aspect + "[SEP]". The representations of the sentence are used for the downstream aspect-level sentiment classification task. Because the tokenizers used in the parser and BERT are different, we get the BERT representations for tokens in dependency graphs by averaging the corresponding representations of sub-word units ("wordpiece") from BERT. For example, the representation of the token "overload" is the average of representations of two sub-words "over" and "##load". Once the word representations are initialized, they are fixed during training.

We set the dimension of hidden states as 300 in our experiments. For the BERT representations, we first map word representations into 300 dimensional vectors by a linear projection layer. We use 6 attention heads in our model. We train our model with batch size of 32. We apply $l_2$ regularization with term $\lambda$ $10^{-4}$ and dropout (Srivastava et al., 2014) on the input word embedding with rate $0.7$. We first use Adam (Kingma and Ba, 2014) optimizer with learning rate $10^{-3}$ to train our model, then switch to stochastic gradient descent to fine-tune and stabilize our model.

We implemented our model using PyTorch Geometric (Fey and Lenssen, 2019) on a Linux machine with Titan XP GPUs.

### 4.3 Baseline Comparisons

To validate the effectiveness of our method, we compare it to following baseline methods:

**Feature-based SVM** utilizes n-gram features, parse features and lexicon features for aspect-level sentiment classification (Kiritchenko et al., 2014)

**TD-LSTM** is a direct competitor against our method. It uses two LSTM networks to model the preceding and following contexts surrounding the aspect term, while we use GAT to model the syntax context around an aspect. The last hidden states of these two LSTM networks are concatenated for predicting the sentiment polarity (Tang et al., 2016a).

**AT-LSTM** first models the sentence via a LSTM model. Then it combines the hidden states from the LSTM with the aspect term embedding to generate the attention vector. The final sentence representation is the weighted sum of the hidden states (Wang et al., 2016).

**MemNet** applies attention multiple times on the word embeddings, and the output of last attention is fed to softmax for prediction (Tang et al., 2016b).

**IAN** uses two LSTM networks to model the sentence and aspect term respectively. It uses the hidden states from the sentence to generate an attention vector for the target, and vice versa. Based on these two attention vectors, it outputs a sentence representation and a target representation for classification (Ma et al., 2017).

**PG-CNN** is a CNN based model where aspect features are used as gates to control the feature extraction on sentences (Huang and Carley, 2018).

**AOA-LSTM** introduces an attention-over-attention (AOA) based network to model aspects and sentences in a joint way and explicitly capture the interaction between aspects and context sentences (Huang et al., 2018).

**BERT-AVG** uses the average of the sentence representations to train a linear classifier.

**BERT-CLS** is a model where we directly use the representation of "[CLS]" as a classification feature to fine-tune the BERT model for paired sentence classification. We fine-tune it for 5 epochs using Adam optimizer with batch size 8 and learning rate $10^{-5}$.

The comparison results are shown in Table 2. With GloVe embeddings, our approach TD-GAT-GloVe (k), where k is the number of layers, outperforms all these previous methods. Among these baselines, Feature-based SVM achieves strong performance on this task, which indicates the importance of feature engineering and syntax knowledge.

As one direct competitor, TD-LSTM propagates sentiment features from the beginning and the end of the sentence to the aspect target, while our model propagates features from syntax dependent words to the target on a dependency graph. Compared to TD-LSTM, our model shows superior performance, which directly proves the necessity of incorporating syntax information.

[2] https://github.com/huggingface/pytorch-pretrained-BERT

| | Laptop | Restaurant |
|---|---|---|
| Feature+SVM | 70.5 | 80.2 |
| TD-LSTM | 68.1 | 75.6 |
| AT-LSTM | 68.9 | 76.2 |
| MemNet | 72.4 | 80.3 |
| IAN | 72.1 | 78.6 |
| PG-CNN | 69.1 | 78.9 |
| AOA-LSTM | 72.6 | 79.7 |
| TD-GAT-GloVe (3) | 73.7 | 81.1 |
| TD-GAT-GloVe (4) | **74.0** | 80.6 |
| TD-GAT-GloVe (5) | 73.4 | **81.2** |
| BERT-AVG | 76.5 | 78.7 |
| BERT-CLS | 77.1 | 81.2 |
| TD-GAT-BERT (3) | 79.3 | 82.9 |
| TD-GAT-BERT (4) | 79.8 | **83.0** |
| TD-GAT-BERT (5) | **80.1** | 82.8 |

Table 2: Comparison results of different methods on laptop and restaurant datasets. Numbers in parentheses indicate number of layers in our model.

Using BERT representations further boosts the performance of our model. BERT-AVG, which uses BERT representations without fine-tuning, achieves surprisingly excellent performance on this task. After fine-tuning, the performance of BERT-CLS becomes even better. However, we observe that such fine-tuning is quite unstable. The model cannot converge in some trials.

Even though the original BERT model already provides strong prediction power, our model consistently improves over BERT-AVG and BERT-CLS, which indicates that our model can better utilize these semantic representations. The accuracy of our model reaches about 80% and 83% on the laptop and restaurant datasets respectively.

### 4.4 Effects of Target Information

In this section, we provide an ablation study to show the effects of explicitly capturing target information. In the ablated model, we remove the LSTM unit in our TD-GAT model, so that it cannot utilize the aspect target information explicitly. We denote this ablated model as GAT.

As shown in Table 3, explicitly capturing aspect target information consistently improves the performance of the TD-GAT-GloVe over the GAT-GloVe model. On average, the accuracy of TD-GAT-GloVe increased by 1.2 percentage points. Capturing aspect-related information explicitly across layer is also useful for the BERT-based model as well. Even though the target information has been embedded in the BERT representation because of the contextual language modeling, TD-GAT-BERT still outperforms the GAT-BERT model. On average, the explicit target information contributes 0.95 percentage points to the final performance of the TD-GAT-BERT.

| Dataset | Laptop | | | Restaurant | | |
|---|---|---|---|---|---|---|
| layer | 3 | 4 | 5 | 3 | 4 | 5 |
| GAT-GloVe | 73.0 | 72.1 | 72.4 | 79.6 | 80.0 | 79.7 |
| TD-GAT-GloVe | 73.7 | 74.0 | 73.4 | 81.1 | 80.6 | 81.2 |
| GAT-BERT | 78.1 | 78.5 | 78.5 | 82.6 | 82.2 | 82.3 |
| TD-GAT-BERT | 79.3 | 79.8 | 80.1 | 82.9 | 83.0 | 82.8 |

Table 3: An ablation study shows the effect of explicit target information.

### 4.5 Effects of Model Depth

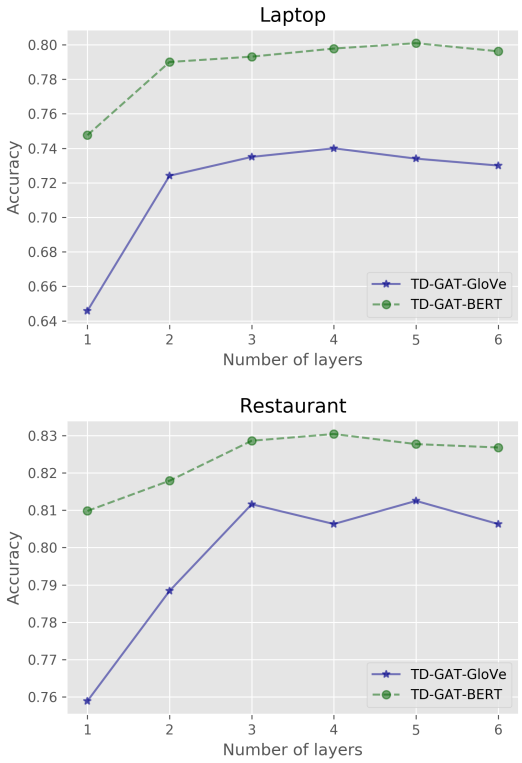


Figure 2: The impact of model depth (number of layers).

We explore the impact of model depth (number of layers) in this section. For our TD-GAT model, we vary its model depth ranging from 1 to 6. As shown in Figure 2, a one-layer TD-GAT model with GloVe embeddings does not work well, which implies target-related sentiment words are usually 2-hops away from the aspect target.

Increasing the model depth to 3 would greatly improve the performance of the TD-GAT-GloVe model.

Unlike the TD-GAT-GloVe model, our model with BERT representations are more robust to the model depth. Even a one-layer TD-GAT-BERT model still achieves satisfactory results on both datasets. One possible reason is that BERT representations already incorporate context words into these semantic representations. Hence nodes at one-hop away may consist of some global information. However, increasing the model depth still improve the performance in this case and our model reaches its optimal performance when model depth is larger than 3.

### 4.6 Model Size

We compare the model size of our TD-GAT model to various baseline methods as well as the BERT model. For these baseline methods, we use an open source PyTorch implementation [3] to measure their model sizes.

| Models | Model size ($\times 10^6$) |
|---|---|
| TD-LSTM | 1.45 |
| MemNet (3) | **0.36** |
| IAN | 2.17 |
| AOA-LSTM | 2.89 |
| TD-GAT-GloVe (3) | 1.00 |
| TD-GAT-GloVe (4) | 1.09 |
| TD-GAT-GloVe (5) | 1.18 |
| BERT-CLS | 335.14 |
| TD-GAT-BERT (3) | **1.30** |
| TD-GAT-BERT (4) | 1.39 |
| TD-GAT-BERT (5) | 1.49 |

Table 4: The model size (number of parameters) of our model as well as baselines.

The sizes of all these models are given in Table 4. Using the same dimension of hidden states, our TD-GAT-GloVe has a lower model size compared to these LSTM-based methods. MemNet is the model ranks the first in terms of the model size. The size of TD-GAT-BERT increases by $0.3 \times 10^6$ because of the linear projection layer applied on the input word representations. When we switch from GloVe embeddings to BERT representations, the training time for a three-layer TD-GAT model on the restaurant dataset only increases from 1.12 seconds/epoch to 1.15 seconds/epoch. On the contrary, fine-tuning the BERT model takes about 226.50 seconds for each training epoch. Training our TD-GAT-BERT model requires much fewer computation resources and less time compared to fine-tuning the original BERT model.

[3] https://github.com/songyouwei/ABSA-PyTorch

## 5 Conclusion

In this paper, we propose a novel target-dependent graph attention neural network for aspect level sentiment classification. It leverages the syntactic dependency structure of a sentence and uses the syntax context of an aspect target for classification. Compared to those methods applied on word sequences, our approach places modifying sentiment words closer to the aspect target and can resolve potential syntactic ambiguity. In our experiments, we demonstrate the effectiveness of our method on laptop and restaurant datasets from SemEval 2014. Using GloVe embeddings, our approach TD-GAT-GloVe outperforms various baseline models. After switching to BERT representations, we show that TD-GAT-BERT achieves much better performance. It is lightweight and requires fewer computational resources and less training time than fine-tuning the original BERT model.

To the best of our knowledge, this paper is the first attempt directly using the original dependency graph without converting its structure for aspect level sentiment classification. Many potential improvements could be made in this direction. In this work, the local feature vector of an aspect node is the average of embedding vectors of words in the aspect and each word in the aspect is equally important. Future work could consider using an attention mechanism to focus on important words in the aspect. Since this work only uses the dependency graph and ignores various types of relations in the graph, we plan to incorporate dependency relation types into our model and take part-of-speech tagging into consideration as well in the future. We would also like to combine such a graph-based model with a sequence-based model to avoid potential noise from dependency parsing errors.

## Acknowledgments

This work was supported in part by the Office of Naval Research (ONR) N000141812108 and N000141712675. The views and conclusions con-

tained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the ONR.

## References


Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *Proceedings of the 52nd annual meeting of the association for computational linguistics (volume 2: Short papers)*, volume 2, pages 49–54.

Matthias Fey and Jan Eric Lenssen. 2019. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*.

Binxuan Huang and Kathleen Carley. 2018. Parameterized convolutional neural networks for aspect level sentiment classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1091–1096.

Binxuan Huang and Kathleen M Carley. 2019. Residual or gate? towards deeper graph neural networks for inductive graph representation learning. *arXiv preprint arXiv:1904.08035*.

Binxuan Huang, Yanglan Ou, and Kathleen M Carley. 2018. Aspect level sentiment classification with attention-over-attention neural networks. In *International Conference on Social Computing, Behavioral-Cultural Modeling and Prediction and Behavior Representation in Modeling and Simulation*, pages 197–206. Springer.

Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. 2011. Target-dependent twitter sentiment classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 151–160. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Svetlana Kiritchenko, Xiaodan Zhu, Colin Cherry, and Saif Mohammad. 2014. Nrc-canada-2014: Detecting aspects and sentiment in customer reviews. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 437–442.

Lishuang Li, Yang Liu, and AnQiao Zhou. 2018. Hierarchical attention based position-aware network for aspect-level sentiment analysis. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 181–189.

Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2017. Interactive attention networks for aspect-level sentiment classification. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 4068–4074.

Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3.

Thien Hai Nguyen and Kiyoaki Shirai. 2015. Phrasernn: Phrase recursive neural network for aspect-based sentiment analysis. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2509–2514.

Bo Pang, Lillian Lee, et al. 2008. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, 2(1–2):1–135.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 27–35, Dublin, Ireland. Association for Computational Linguistics and Dublin City University.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Chi Sun, Luyao Huang, and Xipeng Qiu. 2019. Utilizing bert for aspect-based sentiment analysis via constructing auxiliary sentence. *arXiv preprint arXiv:1903.09588*.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2016a. Effective lstms for target-dependent sentiment classification. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3298–3307.

Duyu Tang, Bing Qin, and Ting Liu. 2016b. Aspect level sentiment classification with deep memory network. *arXiv preprint arXiv:1605.08900*.

Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2018. Learning to attend via word-aspect associative fusion for aspect-based sentiment analysis. In *AAAI*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Joachim Wagner, Piyush Arora, Santiago Cortes, Utsab Barman, Dasha Bogdanova, Jennifer Foster, and Lamia Tounsi. 2014. Dcu: Aspect-based polarity classification for semeval task 4. In *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, pages 223–229.

Yequan Wang, Minlie Huang, Li Zhao, et al. 2016. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 606–615.

Hu Xu, Bing Liu, Lei Shu, and Philip S Yu. 2019. Bert post-training for review reading comprehension and aspect-based sentiment analysis. *arXiv preprint arXiv:1904.02232*.

Wei Xue and Tao Li. 2018. Aspect based sentiment analysis with gated convolutional networks. *arXiv preprint arXiv:1805.07043*.

# Learning Explicit and Implicit Structures for Targeted Sentiment Analysis

**Hao Li** and **Wei Lu**
StatNLP Research Group
Singapore University of Technology and Design
hao_li@mymail.sutd.edu.sg
luwei@sutd.edu.sg

## Abstract

Targeted sentiment analysis is the task of jointly predicting target entities and their associated sentiment information. Existing research efforts mostly regard this joint task as a sequence labeling problem, building models that can capture explicit structures in the output space. However, the importance of capturing implicit global structural information that resides in the input space is largely unexplored. In this work, we argue that both types of information (implicit and explicit structural information) are crucial for building a successful targeted sentiment analysis model. Our experimental results show that properly capturing both information is able to lead to better performance than competitive existing approaches. We also conduct extensive experiments to investigate our model's effectiveness and robustness[1].

## 1 Introduction

Targeted sentiment analysis (TSA) is an important task useful for public opinion mining (Pang and Lee, 2008; Liu, 2010; Ortigosa et al., 2014; Smailović et al., 2013; Li and Wu, 2010). The task focuses on predicting the sentiment information towards a specific target phrase, which is usually a named entity, in a given input sentence. Currently, TSA in the literature may refer to either of the two possible tasks under two different setups: 1) predicting the sentiment polarity for a given specific target phrase (Dong et al., 2014; Wang et al., 2016; Zhang et al., 2016; Xue and Li, 2018); 2) jointly predicting the targets together with the sentiment polarity assigned to each target (Mitchell et al., 2013; Zhang et al., 2015; Li and Lu, 2017; Ma et al., 2018). In this paper, we focus on the latter setup which was originally proposed by Mitchell et al. (2013). Figure 1 presents an example sentence containing three targets. Each target is associated with a sentiment, where we use $+$ for denoting positive polarity, $0$ for neutral and $-$ for negative.




Figure 1: TSA with targets in bold and their associated sentiment on top. Boundaries for the sentiment scope are highlighted in dashed boxes.

Existing research efforts mostly regard this task as a sequence labeling problem by assigning a tag to each word token, where the tags are typically designed in a way that capture both the target boundary as well as the targeted sentiment polarity information together. Existing approaches (Mitchell et al., 2013; Zhang et al., 2015; Ma et al., 2018) build models based on conditional random fields (CRF) (Lafferty et al., 2001) or structural support vector machines (SSVM) (Taskar et al., 2005; Tsochantaridis et al., 2005) to explicitly model the sentiment information with structured outputs, where each targeted sentiment prediction corresponds to exactly one *fixed* output. While effective, such models suffer from their inability in capturing certain long-distance dependencies between sentiment keywords and their targets. To remedy this issue, Li and Lu (2017) proposed their "sentiment scope" model to learn flexible output representations. For example, three text spans with their corresponding targets in bold are presented in Figure 1, where each target's sentiment is characterized by the words appearing in the corresponding text span. They learn from data for each target a latent text span used for attributing its sentiment, resulting in *flexible* output structures.

However, we note there are two major limi-

[1]We release our code at http://www.statnlp.org/research/st.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5478–5488,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

tations with the approach of Li and Lu (2017). First, their model requires a large number of handcrafted discrete features. Second, the model relies on a strong assumption that the latent sentiment spans do not overlap with one another. For example, in Figure 1, their model will not be able to capture the interaction between the target word "*OZ*" in the first sentiment span and the keyword "*amazing*" due to the assumptions made on the explicit structures in the output space. One idea to resolve this issue is to design an alternative mechanism to capture such useful structural information that resides in the input space.

On the other hand, recent literature shows that feature learning mechanisms such as self-attention have been successful for the task of sentiment prediction when targets are given (Wang and Lu, 2018; He et al., 2018; Fan et al., 2018) (i.e., under the first setup mentioned above). Such approaches essentially attempt to learn rich *implicit* structural information in the input space that captures the interactions between a given target and all other word tokens within the sentence. Such implicit structures are then used to generate sentiment summary representation towards the given target, leading to the performance boost.

However, to date capturing rich implicit structures in the joint prediction task that we focus on (i.e., the second setup) remains largely unexplored. Unlike the first setup, in our setup the targets are not given, we need to handle exponentially many possible combinations of targets in the joint task. This makes the design of an algorithm for capturing both implicit structural information from the input space and the explicit structural information from the output space challenging.

Motivated by the limitations and challenges, we present a novel approach that is able to efficiently and effectively capture the explicit and implicit structural information for TSA. We make the following key contributions in this work:

- We propose a model that is able to properly integrate both *explicit* and *implicit* structural information, called **EI**. The model is able to learn flexible explicit structural information in the output space while being able to efficiently learn rich implicit structures by LSTM and self-attention for exponentially many possible combinations of targets in a given sentence.
- We conducted extensive experiments to validate our claim that both explicit and implicit structures are indispensable in such a task, and demonstrate the effectiveness and robustness of our model.

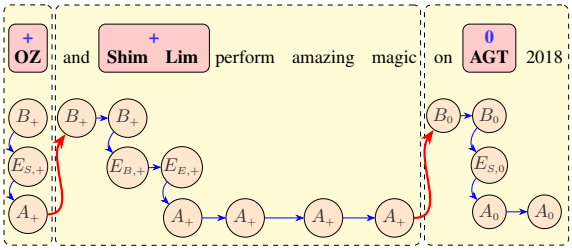


Figure 2: The structured output for representing entities and their sentiments with boundaries.

## 2 Approach

Our objective is to design a model to extract targets as well as their associated targeted sentiments for a given sentence in a joint manner. As we mentioned before, we believe that both explicit and implicit structures are crucial for building a successful model for TSA. Specifically, we first present an approach to learn flexible explicit structures based on latent CRF, and next present an approach to efficiently learn the rich implicit structures for exponentially many possible combinations of targets.

### 2.1 Explicit Structure

Motivated by Li and Lu (2017), we design an approach based on latent CRF to model flexible sentiment spans to capture better explicit structures in the output space. To do so, we firstly integrate target and targeted sentiment information into a label sequence by using 3 types of tags in our **EI** model: $\mathbf{B}_p$, $\mathbf{A}_p$, and $\mathbf{E}_{\epsilon,p}$, where $p \in \{+,-,0\}$ indicates the sentiment polarity and $\epsilon \in \{B,M,E,S\}$ denotes the *BMES* tagging scheme[2]. We explain the meaning of each type of tags as follows.

- $\mathbf{B}_p$ is used to denote that the current word is part of a sentiment span with polarity $p$, but appears before the target word or exactly as the first word of the target.

- $\mathbf{A}_p$ is used to denote that the current word is part of a sentiment span with polarity $p$, but appears after the target word or exactly as the last word of the target.

[2] *B* stands for the beginning of the target phrase, *M* for the middle, *E* for the end and *S* for a single-word target.

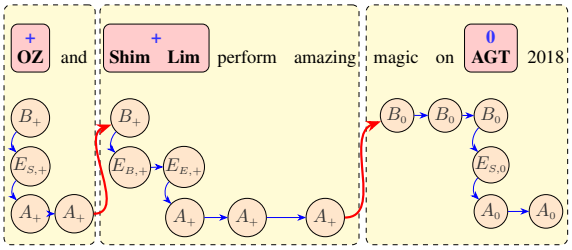


Figure 3: An alternative structured output for the same example with different sentiment boundaries.

- $\mathbf{E}_{\epsilon,p}$ is used to denote the current word is part of a sentiment span with polarity $p$, and is also a part of the target. The *BMES* sub-tag $\epsilon$ denotes the position information within the target phrase. For example, $\mathbf{E}_{B,+}$ represents that the current word appears as the first word of a target with the positive polarity.

We illustrate how to construct the label sequence for a specific combination of sentiment spans of the given example sentence in Figure 2, where three non-overlapping sentiment spans in yellow are presented. Each such sentiment span encodes the sentiment polarity in blue for a target in bold in pink square. At each position, we allow multiple tags in a sequence to appear such that the edge $\mathbf{A}_p\mathbf{B}_{p'}$ in red consistently indicates the boundary between two adjacent sentiment spans.

The first sentiment span with positive (+) polarity contains only one word which is also the target. Such a single word target is also the beginning and the end of the target. We use three tags $\mathbf{B}_+$, $\mathbf{E}_{S,+}$ and $\mathbf{A}_+$ to encode such information above.

The second sentiment span with positive (+) polarity contains a two-word target "*Shin Lim*". The word "and" appearing before such target takes a tag $\mathbf{B}_+$. The words "perform amazing magic" appearing after such target take a tag $\mathbf{A}_+$ at each position. As for the target, the word "Shin" at the beginning of the target takes tags $\mathbf{B}_+$ and $\mathbf{E}_{B,+}$, while the word "Lim" at the end of the target takes tags $\mathbf{E}_{E,+}$ and $\mathbf{A}_+$.

The third sentiment span with neutral (0) polarity contains a single-word target "AGT". Similarly, we use three tags $\mathbf{B}_0$, $\mathbf{E}_{S,0}$ and $\mathbf{A}_0$ to represent such single word target. The word "on" appearing before such target takes a tag $\mathbf{B}_0$. The word "2018" appearing afterwards takes a tag $\mathbf{A}_0$.

Note that if there exists a target with length larger than 2, the tag $\mathbf{E}_{M,p}$ will be used. For example in Figure 2, if the target phrase "Shin Lim" is replaced by "Shin Bob Lim", we will keep the tags at "Shin" and "Lim" unchanged. We assign a tag $\mathbf{E}_{M,+}$ at the word "Bob" to indicate that "Bob" appears in the middle of the target by following the *BMES* tagging scheme.

Finally, we represent the label sequence by connecting adjacent tags sequentially with edges. Notice that for a given input sentence and the output targets as well as the associated targeted sentiment, there exist exponentially many possible label sequences, each specifying a different possible combinations of sentiment spans. Figure 3 shows a label sequence for an alternative combination of the sentiment spans. Those label sequences representing the same input and output construct a latent variable in our model, capturing the flexible explicit structures in the output space.

We use a log-linear formulation to parameterize our model. Specifically, the probability of predicting a possible output $\mathbf{y}$, which is a list of targets and their associated sentiment information, given an input sentence $\mathbf{x}$, is defined as:

$$p(\mathbf{y}|\mathbf{x}) = \frac{\sum_{\mathbf{h}} \exp\left(s(\mathbf{x}, \mathbf{y}, \mathbf{h})\right)}{\sum_{\mathbf{y}', \mathbf{h}'} \exp(s(\mathbf{x}, \mathbf{y}', \mathbf{h}'))} \quad (1)$$

where $s(\mathbf{x}, \mathbf{y}, \mathbf{h})$ is a score function defined over the sentence $\mathbf{x}$ and the output structure $\mathbf{y}$, together with the latent variable $\mathbf{h}$ that provides all the possible combinations of sentiment spans for the $(\mathbf{x}, \mathbf{y})$ tuple. We define $E(\mathbf{x}, \mathbf{y}, \mathbf{h})$ as a set of all the edges appearing in all the label sequences for such combinations of sentiment spans. To compute $s(\mathbf{x}, \mathbf{y}, \mathbf{h})$, we sum up the scores of each edge in $E(\mathbf{x}, \mathbf{y}, \mathbf{h})$:

$$s(\mathbf{x}, \mathbf{y}, \mathbf{h}) = \sum_{e \in E(\mathbf{x}, \mathbf{y}, \mathbf{h})} \phi_{\mathbf{x}}(e)$$

where $\phi_{\mathbf{x}}(e)$ is a score function defined over an edge $e$ for the input $\mathbf{x}$.

The overall model is analogous to that of a neural CRF (Peng et al., 2009; Do et al., 2010); hence the inference and decoding follow standard marginal and MAP inference procedures. For example, the prediction of $\mathbf{y}$ follows the Viterbi-like MAP inference procedure.

## 2.2 Implicit Structure

We propose a design for **EI** to efficiently learn rich implicit structures for exponentially many combinations of targets to predict. To do so, we explain the process to assign scores to each edge

$e$ from our neural architecture. The three yellow boxes in Figure 4 compute scores for rich implicit structures from the neural architecture consisting of LSTM and self-attention.

Given an input token sequence $\mathbf{x} = \{x_1, x_2, \cdots, x_n\}$ of length $n$, we first compute the concatenated embedding $\mathbf{e}_k = [\mathbf{w}_k; \mathbf{c}_k]$ based on word embedding $\mathbf{w}_k$ and character embedding $\mathbf{c}_k$ at position $k$.

As illustrated on the left part in Figure 4, we then use a Bi-directional LSTM to encode context features and obtain hidden states $\mathbf{h}_k = \text{BiLSTM}(\mathbf{e_1}, \mathbf{e_2}, \cdots, \mathbf{e_n})$. We use two different linear layers $f_t$ and $f_s$ to compute scores for target and sentiment respectively. The linear layer $f_t$ returns a vector of length 4, with each value in the vector indicating the score of the corresponding tag under the *BMES* tagging scheme. The linear layer $f_s$ returns a vector of length 3, with each value representing the score of a certain polarity of $+, 0, -$. We assign such scores to each type of edge as follows:

$$\phi_{\mathbf{x}}(\mathbf{E}^k_{\epsilon,p}\mathbf{E}^{k+1}_{\epsilon',p}) = f_t(\mathbf{h}_k)_\epsilon$$

$$\phi_{\mathbf{x}}(\mathbf{E}^k_{\epsilon,p}\mathbf{A}^k_p) = f_t(\mathbf{h}_k)_\epsilon$$

$$\phi_{\mathbf{x}}(\mathbf{B}^k_p\mathbf{B}^{k+1}_p) = f_s(\mathbf{h}_k)_p$$

$$\phi_{\mathbf{x}}(\mathbf{A}^k_p\mathbf{A}^{k+1}_p) = f_s(\mathbf{h}_k)_p$$

$$\phi_{\mathbf{x}}(\mathbf{A}^k_p\mathbf{B}^{k+1}_{p'}) = f_s(\mathbf{h}_k)_p$$

Note that the subscript $p$ and $\epsilon$ at the right hand side of above equations denote the corresponding index of the vector that $f_t$ or $f_s$ returns. We apply $f_t$ on edges $\mathbf{E}^k_{\epsilon,p}\mathbf{E}^{k+1}_{\epsilon',p}$ and $\mathbf{E}^k_{\epsilon,p}\mathbf{A}^k_p$, since words at these edges are parts of the target phrase in a sentiment span. Similarly, we apply $f_s$ on edges $\mathbf{B}^k_p\mathbf{B}^{k+1}_p$,$\mathbf{A}^k_p\mathbf{A}^{k+1}_p$ and $\mathbf{A}^k_p\mathbf{B}^{k+1}_{p'}$, since words at these edges contribute the sentiment information for the target in the sentiment span.

As illustrated in Figure 4, we calculate $\mathbf{a}_k$, the output of self-attention at position $k$:

$$\mathbf{a}_k = \sum_{j=1}^{n} \alpha_{k,j}\mathbf{e}_j$$

$$\alpha_{k,j} = \underset{j}{\text{softmax}}(\beta_{k,j})$$

$$\beta_{k,j} = U^T \text{ReLu}(W[\mathbf{e}_k; \mathbf{e}_j] + b)$$

where $\alpha_{k,j}$ is the normalized weight score for $\beta_{k,j}$, and $\beta_{k,j}$ is the weight score calculated by target

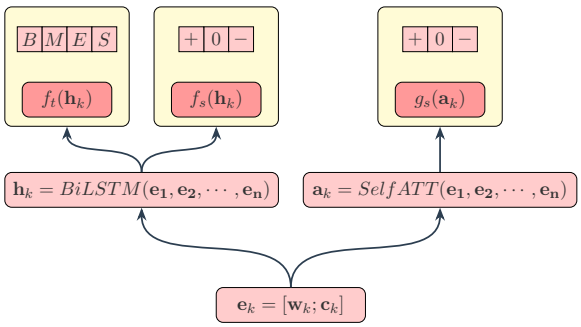


Figure 4: Neural Architecture

representation at position $k$ and contextual representation at position $j$. In addition, $W$ and $b$ as well as the attention matrix $U$ are the weights to be learned. Such a vector $\mathbf{a}_k$ encodes the implicit structures between the word $x_k$ and each word in the remaining sentence.

Motivated by the character embeddings (Lample et al., 2016) which are generated based on hidden states at two ends of a subsequence, we encode such implicit structures for a target similarly. For any target starting at the position $k_1$ and ending at the position $k_2$, we could use $\mathbf{a}_{k_1}$ and $\mathbf{a}_{k_2}$ at two ends to represent the implicit structures of such a target. We encode such information on the edges $\mathbf{B}^{k_1}_p\mathbf{E}^{k_1}_{\epsilon,p}$ and $\mathbf{E}^{k_2}_{\epsilon,p}\mathbf{A}^{k_2}_p$ which appear at the beginning and the end of a target phrase respectively with sentiment polarity $p$. To do so, we assign the scores calculated from the self-attention to such two edges:

$$\phi_{\mathbf{x}}(\mathbf{B}^{k_1}_p\mathbf{E}^{k_1}_{\epsilon,p}) = g_s(\mathbf{a}_{k_1})_p$$

$$\phi_{\mathbf{x}}(\mathbf{E}^{k_2}_{\epsilon,p}\mathbf{A}^{k_2}_p) \mathrel{+}= g_s(\mathbf{a}_{k_2})_p$$

where $g_s$ returns a vector of length 3 with scores of three polarities.

Note that $\mathbf{h}_k$ and $\mathbf{a}_k$ could be pre-computed at every position $k$ and assigned to the corresponding edges. Such an approach allows us to maintain the inference time complexity $O(Tn)$, where $T$ is the maximum number of tags at each position which is 9 in this work and $n$ is the number of words in the input sentence. This approach enables **EI** to efficiently learn rich implicit structures from LSTM and self-attention for exponentially many combinations of targets.

## 3 Experimental Setup

### Data

We mainly conduct our experiments on the datasets released by Mitchell et al. (2013). They

| | #Target | #+ | #− | #0 |
|---|---|---|---|---|
| English | 3,288 | 707 | 275 | 2,306 |
| Spanish | 6,658 | 1,555 | 1,007 | 4,096 |

(a) Statistics on polarity of named entities

| Target length | 1 | 2 | 3 | >= 4 |
|---|---|---|---|---|
| English | 1,910 | 1,032 | 232 | 114 |
| Spanish | 4,201 | 1,794 | 417 | 246 |

(b) Statistics on target length

| #Target | 1 | 2 | 3 | >= 4 |
|---|---|---|---|---|
| English | 1,692 | 465 | 135 | 58 |
| Spanish | 3,855 | 903 | 221 | 69 |

(c) Statistics on number of targets per sentence

Table 1: Corpus Statistics of Main Dataset

contain 2,350 English tweets and 7,105 Spanish tweets, with target and targeted sentiment annotated. See Table 1 for corpus statistics.

### Evaluation Metrics

Following the previous works, we report the *precision* ($P.$), *recall* ($R.$) and $F_1$ scores for target recognition and targeted sentiment. Note that a correct target prediction requires the boundary of the target to be correct, and a correct targeted sentiment prediction requires both target boundary and sentiment polarity to be correct.

### Hyperparameters

We adopt pretrained embeddings from Pennington et al. (2014) and Cieliebak et al. (2017) for English data and Spanish data respectively. We use a 2-layer LSTM (for both directions) with a hidden dimension of 500 and 600[3] for English data and Spanish data respectively. The dimension of the attention weight $U$ is 300. As for optimization, we use the Adam (Kingma and Ba, 2014) optimizer to optimize the model with batch size 1 and dropout rate 0.5. All the neural weights are initialized by Xavier (Glorot and Bengio, 2010).

### Training and Implementation

We train our model for a maximal of 6 epochs. We select the best model parameters based on the best $F_1$ score on the development data after each epoch. Note that we split 10% of data from the training data as the development data[4]. The selected model is then applied to the test data for evaluation. During testing, we map words not appearing in the training data to the *UNK* token. Following the previous works, we perform 10-fold cross validation and report the average results. Our models and variants are implemented using PyTorch (Paszke et al., 2017).

[3]We use a larger LSTM hidden size for Spanish since dimension of Spanish word embedding (200) is larger than dimension of English word embedding (100).

[4]Detailed split information is released with our code.

### Baselines

We consider the following baselines:

- Pipeline (Zhang et al., 2015) and Collapse (Zhang et al., 2015) both are linear-chain CRF models using discrete features and embeddings. The former predicts targets first and calculate targeted sentiment for each predicted target. The latter outputs a tag at each position by collapsing the target tag and sentiment tag together.
- Joint (Zhang et al., 2015) is a linear-chain SSVM model using both discrete features and embeddings. Such a model jointly produces target tags and sentiment tags.
- Bi-GRU (Ma et al., 2018) and MBi-GRU (Ma et al., 2018) are both linear-chain CRF models using word embeddings. The former uses bi-directional GRU and the latter uses multi-layer bi-directional GRU.
- HBi-GRU (Ma et al., 2018) and HMBi-GRU (Ma et al., 2018) are both linear-chain CRF models using word embeddings and character embedding. The former uses bi-directional GRU and the latter uses multi-layer bi-directional GRU.
- SS (Li and Lu, 2017) and SS + *emb* (Li and Lu, 2017) are both based on a latent CRF model to learn flexible explicit structures. The former uses discrete features and the latter uses both discrete features and word embeddings.
- SA-CRF is a linear-chain CRF model with self-attention. Such a model concatenates the hidden state from LSTM and a vector constructed by self-attention at each position, and feeds them into CRF as features. The model attempts to capture rich implicit structures in the input space, but it does not put effort on explicit structures in the output space.
- **E-I** is a weaker version of **EI**. Such a model removes the *BMES* sub-tags in the **E** tag,

| Model | Structure | | English | | | | | | Spanish | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Target Recognition | | | Targeted Sentiment | | | Target Recognition | | | Targeted Sentiment | | |
| | *Explicit* | *Implicit* | *P.* | *R.* | $F_1$ | *P.* | *R.* | $F_1$ | *P.* | *R.* | $F_1$ | *P.* | *R.* | $F_1$ |
| Pipeline (Zhang et al., 2015) | *fixed* | *MLP + discrete + emb* | 60.69 | 51.63 | 55.67 | 43.71 | 37.12 | 40.06 | 70.23 | 62.00 | 65.76 | 45.99 | 40.57 | 43.04 |
| Joint (Zhang et al., 2015) | *fixed* | *MLP + discrete + emb* | 61.47 | 49.28 | 54.59 | 44.62 | 35.84 | 39.67 | 71.32 | 61.11 | 65.74 | 46.67 | 39.99 | 43.02 |
| Collapse (Zhang et al., 2015) | *fixed* | *MLP + discrete + emb* | 63.55 | 44.98 | 52.58 | 46.32 | 32.84 | 38.36 | 73.51 | 53.30 | 61.71 | 47.69 | 34.53 | 40.00 |
| Bi-GRU (Ma et al., 2018) | *fixed* | *GRU + emb* | 58.13 | 43.46 | 49.62 | 45.76 | 32.29 | 37.73 | 65.24 | 53.02 | 58.45 | 46.33 | 37.50 | 41.45 |
| MBi-GRU (Ma et al., 2018) | *fixed* | *MGRU + emb* | 58.27 | 49.01 | 53.24 | 45.80 | 35.21 | 39.81 | 66.14 | 60.07 | 62.95 | 45.61 | 40.04 | 42.64 |
| HBi-GRU (Ma et al., 2018) | *fixed* | *GRU + emb + char* | 57.24 | 53.88 | 55.41 | 44.94 | 38.60 | 41.52 | 68.24 | 61.81 | 64.82 | 46.53 | 42.21 | 44.18 |
| HMBi-GRU (Ma et al., 2018) | *fixed* | *MGRU + emb + char* | 60.12 | 53.68 | 56.98 | 46.52 | 39.99 | 42.87 | 68.64 | 63.66 | 66.01 | 48.09 | 43.44 | 45.61 |
| SS (Li and Lu, 2017) | *flexible* | *discrete* | 63.18 | 51.67 | 56.83 | 44.57 | 36.48 | 40.11 | 71.49 | 61.92 | 66.36 | 46.06 | 39.89 | 42.75 |
| SS + *emb* (Li and Lu, 2017) | *flexible* | *discrete + emb* | 66.35 | 56.59 | 61.08 | 47.30 | 40.36 | 43.55 | 73.13 | 64.34 | 68.45 | 47.14 | 41.48 | 44.13 |
| SA-CRF | *fixed* | *LSTM + SA + emb + char* | 60.26 | 55.60 | 57.53 | 42.95 | 40.46 | 41.45 | 68.47 | 66.39 | 67.26 | 42.22 | 42.97 | 42.47 |
| **E-I** | *flexible* | *LSTM + SA + emb + char* | 67.11 | 58.37 | 62.34 | 47.47 | 41.31 | 44.11 | 73.47 | 65.91 | 69.44 | 47.80 | 42.90 | 45.19 |
| **EI-** | *flexible* | *LSTM + emb + char* | 68.67 | 57.52 | 62.54 | 48.73 | 40.89 | 44.42 | 72.62 | 66.97 | 69.61 | 47.06 | 43.45 | 45.14 |
| **EI** | *flexible* | *LSTM + SA + emb + char* | **69.70** | **58.33** | **63.48** | **49.78** | **41.71** | **45.37** | **74.25** | **68.37** | **71.17** | **48.10** | **44.29** | **46.11** |

Table 2: Main Results. *fixed* stands for chain structures and *flexible* for latent structures. *discrete*, *emb* and *char* denote discrete features, word embeddings and character embeddings respectively. *SA* represents self-attention.

causing the model to learn less explicit structural information in the output space.

- **EI-** is a weaker version of **EI**. Such a model removes the self-attention from **EI**, causing the model to learn less expressive implicit structures in the input space.

## 4 Results and Discussion

### 4.1 Main Results

The main results are presented in Table 2, where explicit structures as well as implicit structures are indicated for each model for clear comparisons.

In general, our model **EI** outperforms all the baselines. Specifically, it outperforms the strongest baseline **EI-** significantly with $p < 0.01$ on the English and Spanish datasets in terms of $F_1$ scores[5]. Note that **EI-** which models flexible explicit structures and less implicit structural information, achieves better performance than most of the baselines, indicating flexible explicit structures contribute a lot to the performance boost.

Now let us take a closer look at the differences based on detailed comparisons. First of all, we compare our model **EI** with the work proposed by Zhang et al. (2015). The Pipeline model (based on CRF) as well as Joint and Collapse models (based on SSVM) in their work capture *fixed* explicit structures. Such two models rely on multi-layer perceptron (MLP) to obtain the local context features for implicit structures. These two models do not put much effort to capture better explicit structures and implicit structures. Our model **EI** (and even **EI-**) outperforms these two models significantly. We also compare our work with models in Ma et al. (2018), which also capture *fixed* explicit structures. Such models leverage different GRUs (single-layer or multi-layer) and different input features (word embeddings and character representations) to learn better contextual features. Their best result by HMBi-GRU is obtained with multi-layer GRU with word embeddings and character embeddings. As we can see, our model **EI** outperforms HMBi-GRU under all evaluation metrics. On the English data, **EI** obtains 6.50 higher $F_1$ score and 2.50 higher $F_1$ score on target recognition and targeted sentiment respectively. On Spanish, **EI** obtains 5.16 higher $F_1$ score and 0.50 higher $F_1$ score on target recognition and targeted sentiment respectively. Notably, compared with HMBi-GRU, even **EI-** capturing the flexible explicit structures achieves better performance on most of metrics and obtains the comparable results in terms of precision and $F_1$ score on Spanish. Since both **EI** and **EI-** models attempt to capture the *flexible* explicit structures, the comparisons above imply the importance of modeling such *flexible* explicit structures in the output space.

We also compare **EI** with **E-I**. The difference between these two models is that **E-I** removes the *BMES* sub-tags. Such a model captures less explicit structural information in the output space. We can see that **EI** outperforms **E-I**. Such results show that adopting *BMES* sub-tags in the output space to capture explicit structural information is beneficial.

Now we compare **EI** with SA-CRF which is a linear-chain CRF model with self-attention. Such a model attempts to capture rich implicit structures, and *fixed* explicit structures. The difference between **EI** and SA-CRF is that our model **EI** captures *flexible* explicit structures in the output space

[5]We have conducted significance test using the bootstrap resampling method (Koehn, 2004).

| Model | Subj (+/-,o) | | | SA (+,-) | | |
|---|---|---|---|---|---|---|
| | *P.* | *R.* | $F_1$ | *P.* | *R.* | $F_1$ |
| Zhang et al. (2015) | 49.2 | 42.1 | 45.3 | 40.9 | 21.6 | 27.9 |
| SS + *emb* (Li and Lu, 2017) | 50.0 | 44.0 | 46.8 | 37.6 | 25.4 | 30.2 |
| SA-CRF | 44.8 | 45.2 | 44.9 | 35.2 | 25.6 | 29.3 |
| **EI-** | 49.7 | 45.8 | 47.6 | **43.0** | 24.9 | 30.2 |
| **EI** | **50.5** | **46.5** | **48.4** | 42.0 | **25.6** | **31.5** |

Table 3: Results on subjectivity as well as non-neutral sentiment analysis on the Spanish dataset. Subj(+/-,o): subjectivity for all polarities. SA(+,-): sentiment analysis for non-neutral polarities.

which model output representations as latent variables. We can see that **EI** outperforms SA-CRF on all the metrics. Such a comparison also implies the importance of capturing *flexible* explicit structures in the output space.

Next, we focus on the comparisons with SS (Li and Lu, 2017) and SS + *emb* (Li and Lu, 2017). Such two models as well as our models all capture the *flexible* explicit structures. As for the difference, both two SS models rely on hand-crafted discrete features to capture implicit structures, while our model **EI** and **EI-** learn better implicit structures by LSTM and self-attention. Furthermore, our models only require word embeddings and character embeddings as the input to our neural architecture to model rich implicit structures, leading to a comparatively simpler and more straightforward design. The comparison here suggests that LSTM and self-attention neural networks are able to capture better implicit structures than hand-crafted features.

Finally, we compare **EI** with **EI-**. We can see that the $F_1$ scores of targeted sentiment for both English and Spanish produced by **EI** are $0.95$ and $0.97$ points higher than **EI-**. The main difference here is that **EI** makes use of self-attention to capture richer implicit structures between each target phrase and all words in the complete sentence. The comparisons here indicate the importance of capturing rich implicit structures using self-attention on this task.

### Robustness

Overall, all these comparisons above based on empirical results show the importance of capturing both *flexible* explicit structures in the output space and rich implicit structures by LSTM and self-attention in the input space.

We analyze the model robustness by assessing the performance on the targeted sentiment for targets of different lengths. For both English and Spanish, we group targets into 4 categories respectively, namely length of 1, 2, 3 and $\geq 4$. Figure 5 reports the $F_1$ scores of targeted sentiment for such 4 groups on Spanish[6]. As we can see **EI** outperforms all the baselines on all groups.

Furthermore, following the comparisons in Zhang et al. (2015), we also measure the precision, recall and $F_1$ of subjectivity and non-neutral polarities on the Spanish dataset. Results are reported in Table 3[7]. The subjectivity measures whether a target phrase expresses an opinion or not according to Liu (2010). Comparing with the best-performing system's results reported in Zhang et al. (2015) and Li and Lu (2017), our model **EI** can achieve higher $F_1$ scores on subjectivity and non-neutral polarities.

### Error Analysis

We conducted error analysis for our main model **EI**. We calculate $F_1$ scores based on the partial match instead of exact match. The $F_1$ scores for target partial match is $76.04$ and $83.82$ for English and Spanish respectively. We compare these two numbers against $63.48$ and $71.17$ which are the $F_1$ scores based on exact match. This comparison indicates that boundaries of many predicted targets do not match exactly with those of the correct targets. Furthermore, we investigate the errors caused by incorrect sentiment polarities. We found that the major type of errors is to incorrectly predict positive targets as neutral targets. Such errors contribute $64\%$ and $36\%$ of total errors for English and Spanish respectively. We believe they are mainly caused by challenging expressions in the tweet input text. Such challenging expressions such as "*below expectations*" are very sparse in the data, which makes effective learning for such phrases difficult.

## 4.2 Effect of Implicit Structures

In order to understand whether the implicit structures are truly making contributions in terms of the overall performance, we compare the performance among four models: **EI** and **EI-** as well as two variants **EI** (*i:MLP*) and **EI** (*i:Identity*) (where *i* indicates the implicit structure). Such two variants replace the implicit structure by other components:

[6] See the English results in the supplementary material.

[7] Only Spanish results are available in Zhang et al. (2015).

| Model | English | | | | | | Spanish | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Target Recognition | | | Targeted Sentiment | | | Target Recognition | | | Targeted Sentiment | | |
| | Prec. | Rec. | $F_1$ | Prec. | Rec. | $F_1$ | Prec. | Rec. | $F_1$ | Prec. | Rec. | $F_1$ |
| **EI** | **69.70** | **58.33** | **63.48** | **49.78** | **41.71** | **45.37** | **74.25** | **68.37** | **71.17** | **48.10** | **44.29** | **46.11** |
| **EI** (*i:MLP*) | 64.47 | 56.58 | 60.20 | 46.23 | 40.48 | 43.12 | 70.95 | 65.80 | 68.27 | 43.64 | 40.46 | 41.98 |
| **EI** (*i:Identity*) | 63.24 | 55.73 | 59.20 | 45.10 | 39.79 | 42.24 | 69.38 | 66.27 | 67.77 | 43.66 | 41.68 | 42.63 |
| **EI-** | 68.67 | 57.52 | 62.54 | 48.73 | 40.89 | 44.42 | 72.62 | 66.97 | 69.61 | 47.06 | 43.45 | 45.14 |

Table 4: Effect of Implicit Structures

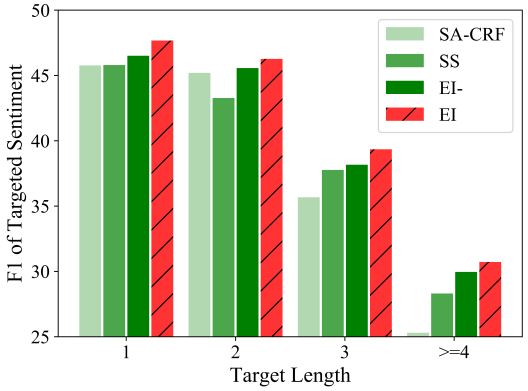


Figure 5: Results of different lengths on Spanish

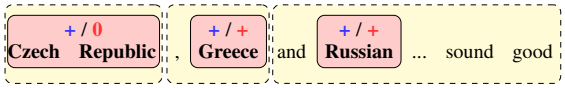


Figure 6: An example sentence in the test data.

- **EI** (*i:MLP*) replaces self-attention by multi-layer perceptron (MLP) for implicit structures. Such a variant attempts to capture implicit structures for a target phrase towards words restricted by a window of size 3 centered at the two ends of the target phrase.

- **EI** (*i:Identity*) replaces self-attention by an identity layer[8] as implicit structure. Such a variant attempts to capture implicit structures for a target phrase towards words at the two ends of the target phrase exactly.

Overall, those variants perform worse than **EI** on all the metrics. When the self-attention is replaced by MLP or the identity layer for implicit structures, the performance drops a lot on both target and targeted sentiment. Such two variants **EI** (*i:MLP*) and **EI** (*i:Identity*) consider the words within a small window centered at the two ends of the target phrase, which might not be capable of capturing the desired implicit structures. The **EI-** model capturing less implicit structural information achieves worse results than **EI**, but obtains better results than the two variants discussed above. This comparison implies that properly capturing implicit structures as the complement of explicit structural information is essential.

[8]The identity layer returns the identical input data.

### 4.3 Qualitative Analysis

We present an example sentence in the test data in Figure 6, where the gold targets are in bold, the predicted targets are in the pink boxes, the gold sentiment is in blue and predicted sentiment is in red. **EI** makes all correct predictions for three targets. **EI-** predicts correct boundaries for three targets and the targeted sentiment predictions are highlighted in Figure 6. As we can see, **EI-** incorrectly predicts the targeted sentiment on the first target as neural (0). The first target here is far from the sentiment expression "*sound good*" which is not in the first sentiment span, making **EI-** not capable of capturing such a sentiment expression. This qualitative analysis helps us to better understand the importance to capture implicit structures using both LSTM and self-attention.

### 4.4 Additional Experiments

We also conducted experiments on multi-lingual Restaurant datasets from SemEval 2016 Task 5 (Pontiki et al., 2016), where aspect target phrases and aspect sentiments are provided. [9] We regard each aspect target phrase as a target and assign such a target with the corresponding aspect sentiment polarity in the data. Note that we remove all the instances which contain no targets in the training data. Following the main experiment, we split 10% of training data as development set for the selection of the best model during training.

We report the $F_1$ scores of target and targeted sentiment for English, Dutch and Russian[10] respectively in Table 5. The results show that **EI**

[9]See the supplementary material for data statistics.

[10]We use the pretrained embedding for Dutch and Russian from https://github.com/Kyubyong/wordvectors.

achieves the best performance. The performance of SS (Li and Lu, 2017) is much worse on Russian due to the inability of discrete features in SS to capture the complex morphology in Russian.

## 5 Related Work

We briefly survey the research efforts on two types of TSA tasks mentioned in the introduction. Note that TSA is related to aspect sentiment analysis which is to determine the sentiment polarity given a target and an aspect describing a property of related topics.

***Predicting sentiment for a given target***

Such a task is typically solved by leveraging sentence structural information, such as syntactic trees (Dong et al., 2014), dependency trees (Wang et al., 2016) as well as surrounding context based on LSTM (Tang et al., 2016a), GRU (Zhang et al., 2016) or CNN (Xue and Li, 2018). Another line of works leverage self-attention (Liu and Zhang, 2017) or memory networks (Tang et al., 2016b) to encode rich global context information. Wang and Lu (2018) adopted the segmental attention (Kong et al., 2016) to model the important text segments to compute the targeted sentiment. Wang et al. (2018) studied the issue that the different combinations of target and aspect may result in different sentiment polarity. They proposed a model to distinguish such different combinations based on memory networks to produce the representation for aspect sentiment classification.

***Jointly predicting targets and their associated sentiment***

Such a joint task is usually regarded as sequence labeling problem. Mitchell et al. (2013) introduced the task of open domain targeted sentiment analysis. They proposed several models based on CRF such as the pipeline model, the collapsed model as well as the joint model to predict both targets and targeted sentiment information. Their experiments showed that the collapsed model and the joint model could achieve better results, implying the benefit of the joint learning on this task. Zhang et al. (2015) proposed an approach based on structured SVM (Taskar et al., 2005; Tsochantaridis et al., 2005) integrating both discrete features and neural features for this joint task. Li and Lu (2017) proposed the sentiment scope model motivated from a linguistic phenomenon to represent the structure information for both the targets and their associated sentiment polarities. They modelled the latent sentiment scope based on CRF with latent variables, and achieved the best performance among all the existing works. However, they did not explore much on the implicit structural information and their work mostly relied on hand-crafted discrete features. Ma et al. (2018) adopted a multi-layer GRU to learn targets and sentiments jointly by producing the target tag and the sentiment tag at each position. They introduced a constraint forcing the sentiment tag at each position to be consistent with the target tag. However, they did not explore the explicit structural information in the output space as we do in this work.

| Model | English | | Dutch | | Russian | |
|---|---|---|---|---|---|---|
| | *target* | *sent* | *target* | *sent* | *target* | *sent* |
| SS (Li and Lu, 2017) | 46.3 | 36.9 | 44.6 | 33.4 | 20.2 | 14.5 |
| SS + *emb* (Li and Lu, 2017) | 57.1 | 48.0 | 46.8 | 33.5 | 35.9 | 24.1 |
| SA-CRF | 60.8 | 51.4 | 49.7 | 34.0 | 54.2 | 43.4 |
| **EI-** | 57.7 | 48.2 | 47.2 | 33.7 | 52.8 | 38.9 |
| **EI** | **62.0** | **51.6** | **50.0** | **34.2** | **54.4** | **43.4** |

Table 5: $F_1$ scores of targets (*target*) and their associated sentiment (*sent*) on SemEval 2016 Restaurant Dataset.

## 6 Conclusion and Future Work

In this work, we argue that properly modeling both *explicit* structures in the output space and the *implicit* structures in the input space are crucial for building a successful targeted sentiment analysis system. Specifically, we propose a new model that captures explicit structures with latent CRF, and uses LSTM and self-attention to capture rich implicit structures in the input space efficiently. Through extensive experiments, we show that our model is able to outperform competitive baseline models significantly, thanks to its ability to properly capture both explicit and implicit structural information.

Future work includes exploring approaches to capture explicit and implicit structural information to other sentiment analysis tasks and other structured prediction problems.

## Acknowledgments

We would like to thank the anonymous reviewers for their thoughtful and constructive comments. This work is supported by Singapore Ministry of Education Academic Research Fund (AcRF) Tier 2 Project MOE2017-T2-1-156.

## References


Mark Cieliebak, Jan Milan Deriu, Dominic Egger, and Fatih Uzdilli. 2017. A twitter corpus and benchmark resources for german sentiment analysis. In *Proc. of the Fifth International Workshop on Natural Language Processing for Social Media*.

Trinh Do, Thierry Arti, et al. 2010. Neural conditional random fields. In *Proc. of AISTATS*.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *Proc. of ACL*.

Feifan Fan, Yansong Feng, and Dongyan Zhao. 2018. Multi-grained attention network for aspect-level sentiment classification. In *Proc. of EMNLP*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proc. of AISTATS*.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2018. Effective attention modeling for aspect-level sentiment classification. In *Proc. of COLING*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proc. of ICLR*.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proc. of EMNLP*.

Lingpeng Kong, Chris Dyer, and Noah A Smith. 2016. Segmental recurrent neural networks. In *Proc. of ICLR*.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proc. of NAACL*.

Hao Li and Wei Lu. 2017. Learning latent sentiment scopes for entity-level sentiment analysis. In *Proc. of AAAI*.

Nan Li and Desheng Dash Wu. 2010. Using text mining and sentiment analysis for online forums hotspot detection and forecast. *Decision support systems*, 48(2).

Bing Liu. 2010. Sentiment analysis and subjectivity. *Handbook of natural language processing*.

Jiangming Liu and Yue Zhang. 2017. Attention modeling for targeted sentiment. In *Proc. of EACL*.

Dehong Ma, Sujian Li, and Houfeng Wang. 2018. Joint learning for targeted sentiment analysis. In *Proc. of EMNLP*.

Margaret Mitchell, Jacqueline Aguilar, Theresa Wilson, and Benjamin Van Durme. 2013. Open domain targeted sentiment. In *Proc. of EMNLP*.

Alvaro Ortigosa, José M Martín, and Rosa M Carro. 2014. Sentiment analysis in facebook and its application to e-learning. *Computers in Human Behavior*.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2).

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *Proc. of NIPS*.

Jian Peng, Liefeng Bo, and Jinbo Xu. 2009. Conditional neural fields. In *Proc. of NIPS*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proc. of EMNLP*.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, Mohammed AL-Smadi, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphée De Clercq, et al. 2016. Semeval-2016 task 5: Aspect based sentiment analysis. In *Proc. of SemEval*.

Jasmina Smailović, Miha Grčar, Nada Lavrač, and Martin Žnidaršič. 2013. Predictive sentiment analysis of tweets: A stock market application. In *Human-Computer Interaction and Knowledge Discovery in Complex, Unstructured, Big Data*. Springer.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2016a. Effective LSTMs for target-dependent sentiment classification. In *Proc. of COLING*.

Duyu Tang, Bing Qin, and Ting Liu. 2016b. Aspect level sentiment classification with deep memory network. In *Proc. of EMNLP*.

Ben Taskar, Vassil Chatalbashev, Daphne Koller, and Carlos Guestrin. 2005. Learning structured prediction models: A large margin approach. In *Proc. of ICML*.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. 2005. Large margin methods for structured and interdependent output variables. *Journal of machine learning research*, 6(Sep):1453–1484.

Bailin Wang and Wei Lu. 2018. Learning latent opinions for aspect-level sentiment classification. In *Proc. of AAAI*.

Shuai Wang, Sahisnu Mazumder, Bing Liu, Mianwei Zhou, and Yi Chang. 2018. Target-sensitive memory networks for aspect sentiment classification. In *Proc. of ACL*.

Wenya Wang, Sinno Jialin Pan, Daniel Dahlmeier, and Xiaokui Xiao. 2016. Recursive neural conditional random fields for aspect-based sentiment analysis. In *Proc. of EMNLP*.

Wei Xue and Tao Li. 2018. Aspect based sentiment analysis with gated convolutional networks. In *Proc. of ACL*.

Meishan Zhang, Yue Zhang, and Duy-Tin Vo. 2015. Neural networks for open domain targeted sentiment. In *Proc. of EMNLP*.

Meishan Zhang, Yue Zhang, and Duy-Tin Vo. 2016. Gated neural networks for targeted sentiment analysis. In *Proc. of AAAI*.

# Capsule Network with Interactive Attention for Aspect-Level Sentiment Classification

[1,2]Chunning Du, [1,2]Haifeng Sun, [1,2]Jingyu Wang*, [1,2]Qi Qi, [1,2]Jianxin Liao
[1,2]Tong Xu, [1,2]Ming Liu
[1]State Key Laboratory of Networking and Switching Technology,
Beijing University of Posts and Telecommunications, Beijing 100876, China
[2]EBUPT Information Technology Co., Ltd., Beijing 100191, China
{duchunning,sunhaifeng_1,wangjingyu,qiqi}@ebupt.com

## Abstract

Aspect-level sentiment classification is a crucial task for sentiment analysis, which aims to identify the sentiment polarities of specific targets in their context. The main challenge comes from multi-aspect sentences, which express multiple sentiment polarities towards different targets, resulting in overlapped feature representation. However, most existing neural models tend to utilize static pooling operation or attention mechanism to identify sentimental words, which therefore insufficient for dealing with overlapped features. To solve this problem, we propose to utilize capsule network to construct vector-based feature representation and cluster features by an EM routing algorithm. Furthermore, interactive attention mechanism is introduced in the capsule routing procedure to model the semantic relationship between aspect terms and context. The iterative routing also enables encoding sentence from a global perspective. Experimental results on three datasets show that our proposed model achieves state-of-the-art performance.

## 1 Introduction

Aspect-level sentiment classification is a fine-grained task in the field of sentiment analysis (Pang and Lee, 2008; Liu, 2012), which aims to infer the sentiment polarity (e.g., positive, neutral, negative) of a sentence with respect to the aspect. It demands to differentiate sentiments towards different targets when there are multiple targets in one sentence. For example, given the mentioned aspect term {*food, price, drinks*}, and the sentence is *"The food was definitely good, but when all was said and done, I just could not justify it for the price including 2 drinks, $100/person."* For aspect term *food*, the sentimental polarity is positive, but for aspect term *price*, the polarity is negative while for aspect term *drinks*, the polarity is neutral. Recently, with the development of deep learning techniques, various neural networks are designed for this task and obtain promising results (Wang et al., 2016a; Ma et al., 2017; Fan et al., 2018).

The main challenge in aspect-level sentiment classification is that one sentence expresses multiple sentiment polarities, resulting in overlapped feature representation. Take the same example above, the sentence simultaneously reviews on 'food', 'price', and 'drinks', and expresses three different sentiment polarities. The highly overlapped features will confuse the classifier seriously. However, most existing methods only keep the most active feature by max-pooling operation or utilize attention mechanism to find the sentimental words, which fails to distinguish the overlapped features.

Therefore, we propose a novel capsule network and iterative EM routing method with interactive attention (IACapsNet) to solve this problem. Capsule network (Hinton et al., 2011; Sabour et al., 2017; Hinton et al., 2018) constructs vector-based feature representation. Capsules in adjacent layers are connected by dynamic routing, which shows strengths in distinguishing overlapped features by feature clustering (Sabour et al., 2017; Zhang et al., 2019). In the aspect-level sentimental classification task, the vector-based overlapped sentimental features towards different aspect terms will be clustered by an Expectation-Maximization (EM) routing algorithm, which makes the subsequent classification more clear. Furthermore, we further devise an interactive attention-based routing mechanism in order to highlight the word-level difference and model the semantic relationship between aspect terms and context.

Moreover, our iterative routing mechanism can

*Corresponding author.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5489–5498,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

be viewed as a top-down attention mechanism, which is more efficient because of the global perspective compared to the standard attention mechanism. Standard attention mechanism in this task only considers a part of the context information in a sentence without considering the overall meaning conveyed by the sentence, which may introduce noise and downgrade the prediction accuracy, especially for complex sentences. For example, in an ironic statement for the aspect term "mac os": "*Maybe the mac os improvement were not the product they want to offer.*", the standard attention mechanism will highlight the sentimental word '*improvement*' and confuse the classifier to make the wrong prediction to be positive. Our routing mechanism can tackle this by adjusting the contribution of each low-level capsule based on the high-level capsules (overall representation), and the iterative update makes the overall representation more accurate compared to other similar top-down attention (Liu et al., 2018; Zhao and Zhang, 2018).

Our proposed model (IACapsNet) is evaluated on three datasets: laptop, restaurant datasets from the SemEval 2014 Task 4 and Twitter collection. The experimental results show that our model outperforms other baseline methods and achieves state-of-the-art performance. Our contributions are summarized as follows:

- We apply capsule network to aspect-level sentiment classification to tackle the overlapped features by feature clustering. To the best of our knowledge, there is no work that investigates the performance of capsule network in this task.
- An interactive attention mechanism is introduced in the capsule routing to help model the semantic relationship between aspect term and context.

## 2 Related work

### 2.1 Aspect Level Sentiment Classification

Traditional approaches have designed rich features about content and syntactic structures to capture the sentiment polarity (Jiang et al., 2011; Pérez-Rosas et al., 2012). However, These feature-based methods are labor-intensive and the performance highly depends on the quality of the features. Recently, deep learning methods are becoming popular for aspect-level sentiment classification. Recurrent Neural Networks (RNNs) are the most commonly used technique for this task (Tang et al., 2016a). The attention mechanism is further introduced to model the target-context association (Wang et al., 2016b; Li et al., 2017; Ma et al., 2017). Furthermore, Fan et al. (2018) proposed MGAN to integrate fine-grained attention mechanisms, which is employed to characterize the word-level interactions between aspect and context words. Very recently, CNN-based models have shown the strengths in efficiency to tackle the aspect-level sentiment classification (Xue and Li, 2018; Huang and Carley, 2018; Li et al., 2018). However, all the previous methods utilize static pooling operation or attention mechanism to locate the sentimental words, which fails to handle the overlapped features. We introduces vector-based feature representation and feature clustering to address this.

### 2.2 Capsule Network

Capsule network was proposed to improve the representational limitations of CNN and RNN by extracting features in the form of vectors. The technique was firstly proposed in (Hinton et al., 2011) and improved in (Sabour et al., 2017; Hinton et al., 2018), which is mainly devised for image processing domain. Introducing capsules allows us to utilize a routing mechanism instead of pooling operation to generate high-level features which is a more efficient way for features encoding. Routing-by-agreement is able to cluster features in an iterative way, which achieved impressive performance recognizing highly overlapped digits.

Several types of capsule networks have been proposed for natural language processing. Yang et al. (2018) investigated capsule networks for text classification. They also found that capsule networks exhibit significant improvement when transferring single-label to multi-label text classification. Similar property has also been observed in the task of relation extraction (Zhang et al., 2019, 2018). However, interactive word-level attention is not considered in these typical capsule routing methods.

## 3 Model

In this section, we describe the proposed capsule network with interactive attention (IACapsNet) in details. The aim of aspect-level sentiment classification is to predict the sentiment class $y$ of a

sentence over a specific aspect term, where $y \in \{positive, negative, neutral\}$. The overall architecture is shown in Figure 1. It consists of the input embedding layer, bidirectional RNN layer, primary capsule layer, and output layer.

### 3.1 Input Embedding Layer

The context's input representations of IACapsNet include word embeddings $w_n$ and position embeddings $p_n$. The aspect term's input representation only consists of word embedding $w_n^a$.

Word embedding is a distributed representation of a word, where words from the vocabulary are mapped to vectors. Initializing words vectors via pre-trained word vectors can improve the performance due to their ability to capture syntactic and semantic information of words from large scale unlabeled text. In our model, we employ the pre-trained word vector GloVe (Pennington et al., 2014) to obtain the fixed word embedding $w_n, w_n^a \in \mathbb{R}^{d_w}$, where $d_w$ is the word vector dimension.

Considering that the context words with closer distance to an aspect may have higher influence on the sentiment analysis, we introduce position embedding to encode the relative distance $r_n$ from word $w_n$ to the aspect term. We define the position embedding matrix $P \in \mathbb{R}^{d_p \times N}$, which is randomly initialized and updated during the training process. Here, $d_p$ is the position embedding dimension and $N$ denotes the length of the sentence. The corresponding word's position embeddings $p_n$ can be obtained by looking up the position embedding matrix $P$ using $r_n$.

The input representation for each context word is the concatenation of word embeddings and position embeddings: $x_n = [w_n; p_n] \in \mathbb{R}^{d_w + d_p}$.

### 3.2 Bidirectional Recurrent Networks Layer

The recurrent neural networks can capture long-distance dependencies within a sentence. A bidirectional recurrent neural network is the first layer of IACapsNet. The forward direction captures the left context $h_l$ for a word and the backward direction captures the right context $h_r$. We concatenate the left context and the right context as the contextualized word representation $h_n^c$, $h_n^a \in \mathbb{R}^{2 \times d_l}$, where $d_l$ is the dimension of hidden state, $h^a$ and $h^c$ are the word representations for aspect term and context, respectively.

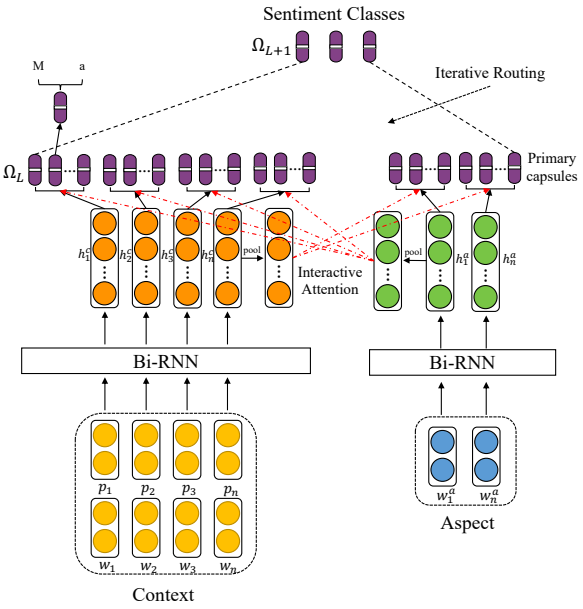


Figure 1: The architecture of IACapsNet

### 3.3 Primary Capsule Layer

The primary capsule is a group of neurons obtained from the output of the convolutional operation performed on $h_n^a$ and $h_n^c$. So, the output of capsule is a vector representing different properties of the same objective. In aspect-level sentiment classification task, the properties may contain the sentiment and aspect term features.

EM-based routing method (Hinton et al., 2018) is implemented in our model, and except for the high-dimensional output $M$, there is one more activation probability $\boldsymbol{a}$ in our capsule, which is like the activity in a standard neural net (shown in Figure 1).

### 3.4 Interactive Attention EM Routing

We have already decided on the outputs of all the capsules $\Omega_L$ in primary capsule layer and we now want to decide which capsules $\Omega_{L+1}$ to active in the layer above and how to assign each active low-level capsule to one active higher-level capsule.

The vector-based features get clustered in the high-level capsules by an EM based algorithm where the outputs of high-level capsules play the role of Gaussians and the output vectors of low-level capsules play the role of the datapoints. The means, variances, and activation probabilities of the output capsules, as well as the assignment probabilities $R$ of the input capsules are iteratively updated by alternating between an E-step and an M-step. It can also be viewed as a parallel atten-

tion mechanism in the opposite direction, which can adjust the low-level word's contribution based on the sentence global representation. Moreover, in order to model the semantic relationship between aspect term and context, we further devise an interactive attention-based routing mechanism.

#### 3.4.1 M-step

The M-step holds the assignment probabilities $R$ constant and adjusts each Gaussian (i.e. high-level capsules) to maximize the sum of the weighted log probabilities that the Gaussian would generate the datapoints (i.e. low-level capsules) assigned to it. This procedure aims to obtain the overall representation among the low-level capsules for a given iteration.

Firstly, every primary capsule $i$ is transformed by $W_{ij}$ to cast a vote $V_{ij} = M_i W_{ij}$ for the output of high-level capsule $j$. And we can get the mean $\mu_j$ of the votes from the input capsules and the variance $\sigma_j$ about that mean for each dimension $h$:

$$\mu_j^h = \frac{\sum_i R_{ij} V_{ij}^h}{\sum_i R_{ij}}, \tag{1}$$

$$(\sigma_j^h)^2 = \frac{\sum_i R_{ij} (V_{ij}^h - \mu_j^h)^2}{\sum_i R_{ij}}, \tag{2}$$

where $\mu_j^h$ is the $h^{th}$ component of the capsule $j$'s vectorized output $M_j$.

The activation probability of capsule $j$ is calculated by

$$cost_j^h = (\beta_u + log(\sigma_j^h)) \sum_i R_{ij}, \tag{3}$$

$$a_j = \text{sigmoid}(\lambda(\beta_\alpha - \sum_h cost_j^h)), \tag{4}$$

where $\beta_u$ and $\beta_\alpha$ are trainable parameters denoting fixed cost per input capsule when not activating it and fixed cost for coding the mean and variance of capsule $j$ when activating it. The variance $\sigma$ reflects the degree of agreement. An intuitive understanding of this activation probability is that if the votes from low-level capsules are not agreed on one high-level capsule, the activation of the high-level capsule should be low. $\lambda$ is an inverse temperature parameter set $1e{-}3$ with a fixed schedule.

#### 3.4.2 E-step

The E-step adjusts the assignment probabilities $R$ for each datapoint (i.e. low-level capsule) to the Gaussian (i.e. high-level capsules). This procedure aims to adjust the contribution of each capsule based on the high-level capsule (i.e. overall representation) for a given iteration.

We firstly compute the negative log probability density of the vectorized vote under the $j$'s Gaussian distribution:

$$p_j = \frac{1}{\sqrt{\prod_h^H 2\pi(\sigma_j^h)^2}} \exp(-\sum_h^H \frac{(V_{ij}^h - \mu_j^h)^2}{2(\sigma_j^h)^2}). \tag{5}$$

For capsule $i$ in primary capsule layer, the assignment probability is adjusted by:

$$R_{ij} = \frac{a_j p_j}{\sum_j a_j p_j}. \tag{6}$$

Alternating E-step and M-step will route the output of capsule to a capsule in the layer above that receives a cluster of high-dimensional features.

**Algorithm 1** Interactive attention EM routing. Capsule $i$ and $j$ denote a low-level and high-level capsule. $\Omega_L$ and $\Omega_{L+1}$ denote the low-level and high-level capsules set, respectively.

**procedure** INTERACTIVE ATTENTION EM ROUTING($a_i, V_{ij}$,$\alpha^{a\to c}$,$\alpha^{c\to a}$)
  $\forall$ capsules $i$ and capsules $j$: $R_{ij} = \frac{1}{|\Omega_{L+1}|}$
  $\forall$ capsules $i$ from context: $a_i = a_i * \alpha_n^{a\to c}$
  $\forall$ capsules $i$ from aspect: $a_i = a_i * \alpha_n^{c\to a}$
  $n$ is the token index from which the capsule $i$ comes.
  **for** r iterations **do**
    $\forall j \in \Omega_{L+1}$: M-STEP($a_i$, $R_{ij}$,$V_{ij}$,$j$)
    $\forall i \in \Omega_L$: E-STEP($\mu_j$, $\sigma_j$,$a_j$,$V_{ij}$,$i$)
  **end for**
  return $a_j$
**end procedure**
**procedure** M-STEP($a_i$,$R_{ij}$,$V_{ij}$)
  $\forall$ capsule $i$: $R_{ij} = R_{ij} * a_i$
  $\forall$ capsule $j$: compute $\mu_j$ and $\sigma_j$ by Eq. 1 and 2
  $\forall$ capsule $j$: compute $a_j$ by Eq. 4
**end procedure**
**procedure** E-STEP($\mu_j$,$\sigma_j$,$a_j$,$V_{ij}$)
  $\forall$ capsule $j$: compute $p_j$ and update $R_{ij}$ by Eq. 5 and 6
**end procedure**

#### 3.4.3 Interactive Attention

The overlapped features in primary capsule layer can be routed and clustered to high-level capsules in an iterative way. However, purely alternating the E-step and M-step will ignore the relationship between the context and aspect term. It has been demonstrated that target and context can determine the representation of each other. And the coordination of targets and their contexts can remarkably enhance the performance of sentiment classification (Ma et al., 2017; Fan et al., 2018). Word-level attention, which has already been demonstrated to be essential is also ignored. So, apart from the assignment probabilities, we further introduce an interactive attention weight $\alpha$ which is learned interactively between the context and the aspect term.

Specifically, we implement scaled dot-product attention which can be described as mapping a query and a set of key-value pairs to a weight on the word-level token. The queries are the averaged representation of the context $h^c$ and the aspect term $h^a$ which are transformed to dimension $d_k$ by trainable parameters:

$$q_c = \frac{\sum_n^N h_n^c}{N} * W_c^q \in \mathbb{R}^{d_k}, \tag{7}$$

$$q_a = \frac{\sum_n^N h_n^a}{N} * W_a^q \in \mathbb{R}^{d_k}, \tag{8}$$

The keys are the corresponding words' representation from aspect term and context which are also transformed to dimension $d_k$:

$$k_n^c = h_n^c * W_c^k \in \mathbb{R}^{d_k}, \tag{9}$$

$$k_n^a = h_n^a * W_a^k \in \mathbb{R}^{d_k}, \tag{10}$$

where $W_c^q, W_a^q, W_c^k, W_a^k \in \mathbb{R}^{2d_l * d_k}$

The attention weights can be computed as follows:

$$s_n^{a\to c}(q_a, k_n^c) = \frac{k_n^c * q_a^\mathsf{T}}{\sqrt{d_k}}, \tag{11}$$

$$s_n^{c\to a}(q_c, k_n^a) = \frac{k_n^a * q_c^\mathsf{T}}{\sqrt{d_k}}, \tag{12}$$

$$\alpha_n^{a\to c} = \frac{\exp(s_n^{a\to c}(q_a, k_n^c))}{\sum_{n=1}^N \exp(s_n^{a\to c}(q_a, k_n^c))}, \tag{13}$$

$$\alpha_n^{c\to a} = \frac{\exp(s_n^{c\to a}(q_c, k_n^a))}{\sum_{n=1}^N \exp(s_n^{c\to a}(q_c, k_n^a))}. \tag{14}$$

In order to ensure a proper magnitude of $s_n$ avoiding pushing the softmax function into regions where it has extremely small gradients, we introduce the scaling factor $\frac{1}{\sqrt{d_k}}$ following (Vaswani et al., 2017). We can thus get the word-level significance for each token in the context and aspect term in an interactive way, which will be adopted on each primary capsule's activation probability $a$. We detail the whole routing algorithm in Algorithm 1.

### 3.5 Training Objective

In IACapsNet, each top-level capsule corresponds to a sentiment category. The activation probability $a$ of each top-level capsule represents the probability that the input sentence belongs to the corresponding category. We use a spread margin loss, $L_k$ for each top-level capsule $k$ to directly maximize the gap between the activation of the target class $(a_t)$ and the activation of the other classes. The total loss $L$ is simply the sum of the losses of all top-level capsules:

$$L = \sum_{k\neq t}(max(0, m - (a_t - a_k))^2, \tag{15}$$

where $m$ is the margin which we set $0.9$ with a fixed schedule.

## 4 Experiments

In this section, we conduct extensive experiments on three datasets to evaluate the proposed IACapsNet.

### 4.1 Experimental Setup

The experiments are implemented on three datasets. The first two datasets are from the SemEval 2014 Task 4 (Pontiki et al., 2014), which contains reviews about laptops and restaurants, respectively. The third one is a Twitter dataset collected by (Dong et al., 2014). The statistics of these datasets are listed in Table 2. Following (Tang et al., 2016c), conflict category is removed from the SemEval 2014 datasets to avoid datasets getting unbalanced. Sentences are zero-padded to the length of the longest sentence in respective dataset. Results are measured by accuracy and Macro-Averaged F1 score.

In our experiments, the pre-trained GloVe (Pennington et al., 2014) is used to initialize the word embeddings from context and aspect term. The

| Model | Laptop | | Restaurant | | Twitter | |
|---|---|---|---|---|---|---|
| | ACC | Macro-F1 | ACC | Macro-F1 | ACC | Macro-F1 |
| ATAE-LSTM (Wang et al., 2016a) | 69.27 | - | 78.50 | - | 69.88 | - |
| TD-LSTM (Tang et al., 2016b) | 68.83 | 68.43 | 78.00 | 66.73 | 66.62 | 64.01 |
| MemNet (Tang et al., 2016c) | 72.37 | - | 80.32 | - | 68.50 | 66.91 |
| IAN (Ma et al., 2017) | 72.10 | - | 78.60 | - | - | - |
| RAM (Chen et al., 2017) | 75.01 | 70.51 | 79.79 | 68.86 | 71.88 | 70.33 |
| BILSTM-ATT-G | 73.12 | 69.80 | 79.73 | 69.25 | 70.38 | 68.37 |
| MGAN (Fan et al., 2018) | 75.39 | 72.47 | 81.25 | 71.94 | 72.54 | 70.81 |
| TNet (Li et al., 2018) | 76.54 | 71.75 | 80.79 | 71.27 | 74.97 | 73.60 |
| PBAN (Gu et al., 2018) | 74.12 | - | 81.16 | - | - | - |
| Cabasc (Liu et al., 2018) | 75.07 | - | 80.89 | - | 71.53 | - |
| IACapsNet | **76.80** | **73.29** | **81.79** | **73.40** | **75.01** | **73.81** |

Table 1: Experimental results. The baseline results are retrieved from the original papers and (Li et al., 2018)

out-of-vocabulary words are initialized by sampling from the uniform distribution $U(-0.1, 0.1)$. In the Bi-RNN layer, LSTM is utilized and the dimension of the hidden state is 300. The dimension of the capsule's output is 16. We use Adam optimizer (Kingma and Ba, 2014) as our optimization method with $5e-4$ learning rate. L2 regularization and dropout are adopted to avoid overfitting.

| Dataset | Positive | | Neural | | Negative | |
|---|---|---|---|---|---|---|
| | Train | Test | Train | Test | Train | Test |
| Restaurant | 2164 | 728 | 637 | 196 | 807 | 196 |
| Laptop | 994 | 341 | 464 | 169 | 870 | 128 |
| Twitter | 1561 | 173 | 3127 | 346 | 1560 | 173 |

Table 2: Summary statistics of the datasets.

### 4.2 Model Comparisons

IACapsNet is compared with the following methods:

**ATAE-LSTM**(Wang et al., 2016a): An LSTM-based model which learns attention embeddings and combine them with the LSTM hidden states to predict the polarity.

**TD-LSTM** (Tang et al., 2016b): It employs two LSTMs to estimate the left context and the right context, respectively. The concatenated context representations perform the predictions.

**IAN** (Ma et al., 2017): An interactive attention is implemented on the representation of context and aspect learned by two LSTMs.

**MemNet** (Tang et al., 2016c): It applies attention mechanism over the word embeddings multiple times and predicts sentiment based on the topmost sentence representation.

**RAM** (Chen et al., 2017): Similar to MemNet, RAM is a multi-layer architecture where each layer consists of attention-based aggregation of word features and a GRU cell to learn the sentence representation.

**BILSTM-ATT-G**: (Zhang and Liu, 2017): It models left and right contexts using two attention-based LSTMs and introduces gates to measure the importance of left context, right context, and the entire sentence for the prediction.

**MGAN** (Fan et al., 2018): MGAN leverages the fine-grained and coarse-grained attention, which is further employed to characterize the word-level interactions between aspect and context words.

**PBAN** (Gu et al., 2018): PBAN concentrates on the position information of aspect terms and mutually models the relation between aspect term and sentence by employing bidirectional attention.

**TNet** (Li et al., 2018): It employs a CNN layer to extract salient features from the transformed word representations originated from a bi-directional RNN layer.

**Cabasc** (Liu et al., 2018): Cabasc employs sentence-level content attention mechanism to capture the important information about given aspects from a global perspective.

### 4.3 Main Results

As shown in Table 1, IACapsNet achieves the best performance on all the datasets. From Table 1, we can have the following observations.

ATAE-LSTM performs better than TD-LSTM.

One main reason may be the attention mechanism in TD-LSTM that enables to notice the important parts based on the aspect term. BILSTM-ATT-G adopts a similar architecture with TD-LSTM by modeling left context and right context using attention-based LSTM, which achieves better results than ATAE-LSTM. IAN and MGAN introduce the interactive attention in coarse-grained and multi-grained ways respectively and bring remarkable improvements. PBAN similarly utilize a fine-grained bidirectional attention and performs comparably with MGAN.

MemNet utilizes a more complex structure that contains nine computational layers, which updates the query vector at each hop. RAM also learns multiple attended vectors on the memory, which achieves superior results among the baseline models, especially on Laptop dataset.

Our proposed IACapsNet consistently performs best on all the three datasets. The improvement is mainly attributed to the feature clustering ability to tackle the overlapped features and the iteratively updating on coupling coefficients, which considers the overall meaning of the contexts. Moreover, compared with Cabasc, which also incorporates the overall representation to typical attention mechanism in a static way, our iterative method shows remarkable strengths.

### 4.4 Ablation Study

To analyze the effect of different components including the routing mechanism and the introduced interactive attention, we report the results of variants of IACapsNet. The results in Table 3 indicate: (1) EM routing based IACapsNet outperforms IACapsNet-Cosine, which routes capsules by cosine similarity (Sabour et al., 2017). One main reason maybe cosine saturates at 1, which is insensitive to the difference between a quite good agreement and a very good agreement. (2) Integrating interactive attention in routing mechanism brings a remarkable improvement on both routing mechanisms, which demonstrates the necessity to consider the relationship between the aspect and contexts during the routing procedure.

Moreover, EM routing also brings a boost on efficiency with fewer trainable parameters and faster speed which is intuitively shown in Table 4 (IAN is listed as baseline). From the table, it is easy to conclude that IACapsNet is much more efficient than IACapsNet-Cosine with about 10% and 36% decrease in the number of trainable parameters and running speed, respectively. Moreover, compared to IAN, IACapsNet achieves a much better accuracy with fewer trainable parameters, meaning that capsule network is more efficient in feature encoding with fewer parameters. However, IACapsNet costs more time compared to IAN, which is the implicit deficiency of capsule network because of the iterative calculation during routing.

### 4.5 Effects of Routing Iteration Number

As our proposed IACapsNet involves iterative procedure during routing. In this section, we investigate the effects of different routing iteration numbers. Specifically, we conduct experiments on all the three datasets and vary routing iteration numbers $r$ from 1 to 4. The results are illustrated in Figure 3.

The results show that IACapsNet achieves the best performance at routing iteration number 2, 3 and 3 on the dataset RESTAURANT, LAPTOP, and Twitter, respectively. When the number of iteration is 1, our capsule network degrades to a standard network, which obtains comparable results with IAN. While increasing $r$ to 4, the performance gets worse dramatically. Moreover, as the number of iteration increases to 4, it brings many difficulties to train IACapsNet. The model becomes more sensitive, which fluctuates greatly in loss and accuracy during training. Therefore, it is appropriate to limit the routing iteration number $r$ and set it to be 2 or 3 depending on the performance.

| Model | Accuracy | Macro-F1 |
|---|---|---|
| IACapsNet-cosine routing | 81.12 | 72.05 |
| -w/o interactive attention | 80.64 | 71.76 |
| IACapsNet-EM routing | **81.79** | **73.40** |
| -w/o interactive attention | 80.89 | 71.84 |

Table 3: Ablation study on Restaurant dataset. Cosine routing denotes routing measured by cosine similarity (Sabour et al., 2017), and w/o means without.

### 4.6 Case Study

In order to assess the effect of our EM routing with interactive attention mechanism, we visualize the coupling coefficients. Our model is able to adjust the contribution of each part based on the global meaning of a sentence and shows superiority in modeling complicated sentence. In this section, we pick an example from RESTAURANT dataset

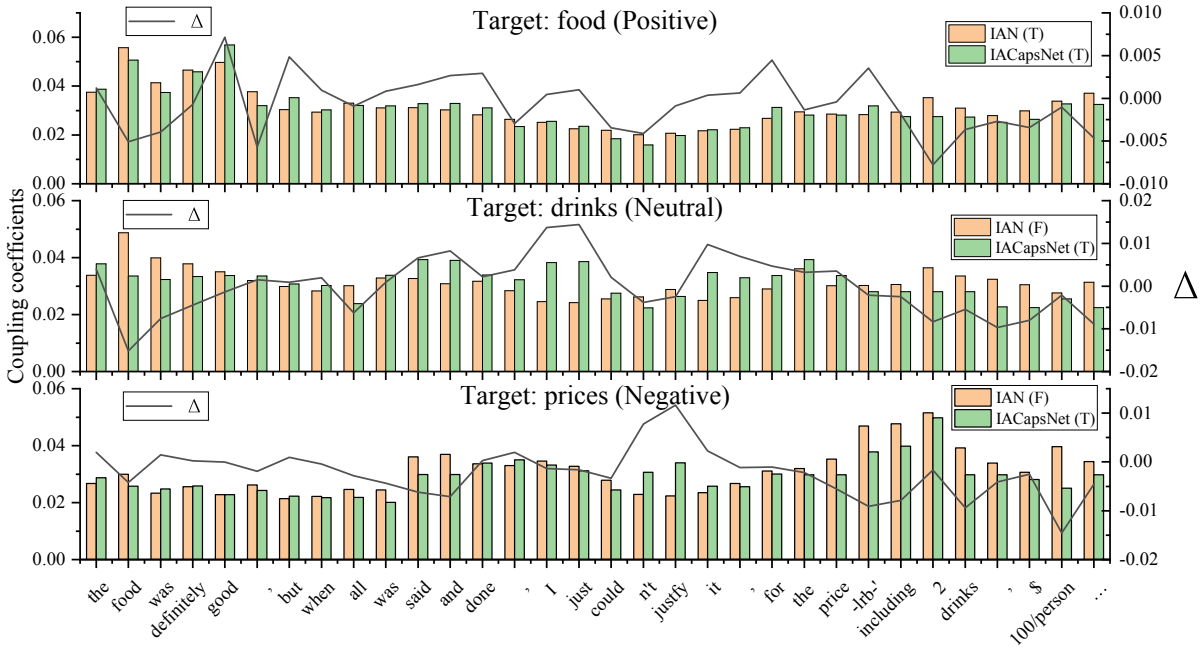


Figure 2: Routing Visualization

| **Model** | Para. | Train | Infer. |
|---|---|---|---|
| IAN (Ma et al., 2017) | 5.057 | **0.11** | **0.017** |
| IACapsNet-cosine routing | 4.614 | 0.33 | 0.044 |
| IACapsNet | **4.189** | 0.21 | 0.024 |

Table 4: Evaluation of efficiency. "Para." denotes the number of trainable parameters (M). "Train" and "Infer." denotes the training speed and inference speed(second/step). Speed is measured on 1 NVIDIA p100 GPU with batch size 64.

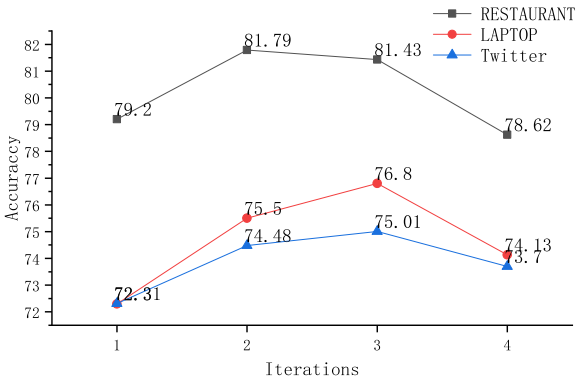


Figure 3: Effects of Routing Iteration Number

consisting of a long and complicated sentence and 3 aspect terms with 3 different sentimental polarities. Figure 2 shows this example and the visualization results of word-level coupling coefficients which are the sum of one word's coupling coefficients to the category. The attention weights from IAN are also shown as a baseline, which is normalized to the same scale with routing coupling coefficients. The line in the chart reflects the difference Δ. 'F' in Figure 2 means false sentiment classification, and 'T' means correct classification.

From the figure, we can observe that our routing methods can adjust the attended words according to different aspect terms, which helps make all the predictions correctly. Moreover, our routing method can locate on the more important words more efficiently. For example, in terms of the aspect 'price', the words "n't" and "justify" is attended, which are the corresponding essential sentimental words. However, they are ignored by the ordinary attention mechanism, which leads a wrong prediction. This shows that our routing mechanism can capture important parts of a sentence more accurately.

### 4.7 Conclusion and Future Work

We re-examine the deficiencies of existing models with attention mechanism for aspect-level sentiment classification. And we propose to utilize capsule network to handle the overlapped sentiment features by features clustering, and iteratively adjust the attention weights from a global perspective. To the best of our knowledge, capsule network is firstly applied in this task. Moreover, interactive attention is introduced to the dynamic rout-

ing to model the semantic relationship between aspect term and sentence. The experimental results verify that IACapsNet outperforms baseline models. The ablation and case studies show the efficacy of different proposed modules.

In the future, our theory can be generalized to other tasks that highly depends on the attention mechanism. For example, reading comprehension and machine translating.

## Acknowledgements

This work was jointly supported by: (1) National Natural Science Foundation of China (No. 61771068, 61671079, 61471063, 61372120, 61421061); (2) Beijing Municipal Natural Science Foundation (No.4182041, 4152039); (3) the National Basic Research Program of China (No. 2013CB329102); (4) Fundamental Research Funds for the Central Universities under Grant 2018RC20; (5) BUPT Excellent Ph.D. Students Foundation.

## References

Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. 2017. Recurrent attention network on memory for aspect sentiment analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 2: Short Papers*.

Feifan Fan, Yansong Feng, and Dongyan Zhao. 2018. Multi-grained attention network for aspect-level sentiment classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*.

Shuqin Gu, Lipeng Zhang, Yuexian Hou, and Yin Song. 2018. A position-aware bidirectional attention network for aspect-level sentiment analysis. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*.

Geoffrey Hinton, Sara Sabour, and Nicholas Frosst. 2018. Matrix capsules with em routing. In *ICLR 2018*.

Geoffrey E. Hinton, Alex Krizhevsky, and Sida D. Wang. 2011. Transforming auto-encoders. In *Artificial Neural Networks and Machine Learning - ICANN 2011 - 21st International Conference on Artificial Neural Networks, Espoo, Finland, June 14-17, 2011, Proceedings, Part I*.

Binxuan Huang and Kathleen M. Carley. 2018. Parameterized convolutional neural networks for aspect level sentiment classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*.

Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. 2011. Target-dependent twitter sentiment classification. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *ICLR*.

Cheng Li, Xiaoxiao Guo, and Qiaozhu Mei. 2017. Deep memory networks for attitude identification. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM 2017, Cambridge, United Kingdom, February 6-10, 2017*.

Xin Li, Lidong Bing, Wai Lam, and Bei Shi. 2018. Transformation networks for target-oriented sentiment classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*.

Bing Liu. 2012. *Sentiment Analysis and Opinion Mining*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Qiao Liu, Haibin Zhang, Yifu Zeng, Ziqi Huang, and Zufeng Wu. 2018. Content attention model for aspect based sentiment analysis. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*.

Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2017. Interactive attention networks for aspect-level sentiment classification. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*.

Verónica Pérez-Rosas, Carmen Banea, and Rada Mihalcea. 2012. Learning sentiment lexicons in spanish. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation, LREC 2012, Istanbul, Turkey, May 23-25, 2012*.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation, SemEval@COLING 2014, Dublin, Ireland, August 23-24, 2014*.

Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. 2017. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2016a. Effective lstms for target-dependent sentiment classification. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2016b. Effective lstms for target-dependent sentiment classification. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*.

Duyu Tang, Bing Qin, and Ting Liu. 2016c. Aspect level sentiment classification with deep memory network. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*.

Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016a. Attention-based LSTM for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*.

Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016b. Attention-based LSTM for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*.

Wei Xue and Tao Li. 2018. Aspect based sentiment analysis with gated convolutional networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*.

Min Yang, Wei Zhao, Jianbo Ye, Zeyang Lei, Zhou Zhao, and Soufei Zhang. 2018. Investigating capsule networks with dynamic routing for text classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*.

Ningyu Zhang, Shumin Deng, Zhanling Sun, Xi Chen, Wei Zhang, and Huajun Chen. 2018. Attention-based capsule network with dynamic routing for relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*.

Xinsong Zhang, Pengshuai Li, Weijia Jia, and Hai Zhao. 2019. Multi-labeled relation extraction with attentive capsule network. *AAAI*.

Yue Zhang and Jiangming Liu. 2017. Attention modeling for targeted sentiment. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers*.

Shenjian Zhao and Zhihua Zhang. 2018. Attention-via-attention neural machine translation. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*.

# Emotion Detection with Neural Personal Discrimination

**Xiabing Zhou**[1], **Zhongqing Wang**[1]*, **Shoushan Li**[1],
**Guodong Zhou**[1], **Min Zhang**[1]
[1]School of Computer Scienceand Technology, Soochow University, China
wangzq.antony@gmail.com,
{zhouxiabing, lishoushan, gdzhou, minzhang}@suda.edu.cn

## Abstract

There have been a recent line of works to automatically predict the emotions of posts in social media. Existing approaches consider the posts individually and predict their emotions independently. Different from previous researches, we explore the dependence among relevant posts via the authors' backgrounds, since the authors with similar backgrounds, e.g., *gender*, *location*, tend to express similar emotions. However, such personal attributes are not easy to obtain in most social media websites, and it is hard to capture attributes-aware words to connect similar people. Accordingly, we propose a Neural Personal Discrimination (NPD) approach to address above challenges by determining personal attributes from posts, and connecting relevant posts with similar attributes to jointly learn their emotions. In particular, we employ adversarial discriminators to determine the personal attributes, with attention mechanisms to aggregate attributes-aware words. In this way, social correlationship among different posts can be better addressed. Experimental results show the usefulness of personal attributes, and the effectiveness of our proposed NPD approach in capturing such personal attributes with significant gains over the state-of-the-art models.

## 1 Introduction

The advent of social media and its prosperity enable the creation of massive online user-generated content including opinions and product reviews. Analyzing such user-generated contents allows to detect the users' emotional states, which are useful for various downstream applications.

In the literature, there are a large number of works on emotion detection (Roberts et al., 2012; Abdul-Mageed and Ungar, 2017; Gupta et al., 2017), both discrete and neural models have been used to predict the emotions of posts in social media. For example, Roberts et al. (2012) used a series of binary SVM classifiers to detect the emotion of a post, while Gupta et al. (2017) used sentiment based semantic embedding and a LSTM model to learn the representation of a post for emotion detection.

Different from previous researches, which consider each post individually, we think that posts in social media are much correlated by the authors' backgrounds. Motivated by the *principle of homophily* (Lazarsfeld et al., 1954), the idea that similarity and connection tend to co-occur, or "birds of a feather flock together", suggests that users, connected by mutual personal backgrounds, may hold similar opinions toward a post (Thelwall, 2010). In the literature, the personal attributes, such as gender, location, age, have been proved useful in personal background construction (Li et al., 2016): people with different attributes tend to express their emotions through different ways. For example, the *happiness* emotion in [E1] is expressed through some femininity sense words, such as "little brother", "handsome", while the *happiness* emotion in [E2] is expressed using a dialectal word "bashi(*comfortable*)", which contains strong characteristic of the Sichuan dialect[1]. Therefore, it is necessary to jointly detect the emotion of posts with the personal attributes.

> **[E1]** 祝贺中国男篮夺冠，新疆小哥哥，真是帅。
> (*Congratulations to Chinese basketball team on becoming the champion, the little brother of Xinjiang is really handsome.*)

* corresponding auther

[1]Sichuan is a southwest province in China, Sichuan dialect(Sichuanese) is quite specific and different from standard Mandarin.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5499–5507,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

**[E2]** 这种天气吃这个真是巴适！
(*It is just bashi to eat this in such weather!*)

However, the personal attributes are not easy to obtain in most social media websites. On one hand, most websites may not contain useful personal information. On the other hand, people are normally not willing to attach their personal information in social media. Besides, integrating personal attributes into emotion detection is challenging, since it is hard to capture attributes-aware words, such as "little brother" and "bashi"(*comfortable*), to connect the posts with similar backgrounds. Although there are some related works on either personal attribute extraction (Wang et al., 2014) or emotion detection with personal attributes (Li et al., 2016), none of them address both challenges at the same time.

In this paper, we propose a Neural Personal Discrimination (NPD) model with both adversarial discriminators and attention mechanisms to tackle above challenges. Here, the *Adversarial discriminators* (Goodfellow et al., 2014) are used to determine the personal attributes, e.g., gender or location, of a post, providing the inherent correlationship between emotions and personal backgrounds, while the *Attention mechanisms* (Wang et al., 2016) are utilized to aggregate the representation of informative attributes-aware words into a vector for emotion prediction, providing insights into which words contribute to a personal background. Experimental results show the usefulness of personal attributes in emotion detection, and the effectiveness of our proposed NPD model with both adversarial discriminators and attention mechanisms over the state-of-the-art discrete and neural models.

## 2 Related Work

Earlier works on emotion detection are based on discrete models. For example, Yang et al. (2007) built a support vector machine (SVM) model and a conditional random field (CRF) model for the emotion detection. Bhowmick et al. (2009) used a multi-label $k$NN model to classify a new sentence into multiple emotion categories. Quan et al. (2015) proposed a logistic regression model for social emotion detection. Recently, with the development of artificial intelligence, neural network models have been successfully applied to various NLP tasks (Collobert et al., 2011; Goldberg, 2016). However, few works use neural network models for emotion detection. Abdul-Mageed and Ungar (2017) used a gated recurrent neural network model for emotion detection with a large-scale dataset. Zhang et al. (2018) used an auxiliary and attention based LSTM to detect emotion on a cross-lingual dataset.

Lexicon and social information are very important for emotion detection, and there are many researches focus on this topic. For example, Strapparava and Mihalcea (2008) used WordNet-Affect to compute the sentimental score of a post. More recently, In addition, Hovy (2015) used both the age and gender information of the authors to improve the performance of sentiment analysis. Vosoughi et al. (2016) explored the relationship among locations, date time, authors and sentiments.

Different from previous works which consider each post individually, we think that the posts in social media can be connected through the authors' backgrounds and should be better addressed. On the basis, we propose a neural personal discrimination model to determine the personal background attributes from each post through adversarial discriminators, and aggregate the representation of informative attributes-aware words through attention mechanisms.

## 3 Vanilla Model for Emotion Detection

In this section, we propose a vanilla model. In the next section, we show how to utilize the neural personal discrimination model to improve the vanilla model by capturing personal attributes.

### 3.1 Document Representation

In general, we denote a post as a document $d$ with $n$ words $\{w_1, w_2, ..., w_n\}$. Given the post, we use a standard Long Short-Term Memory (LSTM) model to learn the shared document representation. Specially, we transform each token $w_i$ into a real-valued vector $x_i$ using the word embedding vector of $w_i$, obtained by looking up a pre-trained word embedding table $D$ via the skip-gram algorithm to train embeddings (Mikolov et al., 2013). We then employ the LSTM model over $d$ to generate a hidden vector sequence $\{h_1, h_2, ..., h_n\}$. At each step $t$, the hidden vector $h_t$ of the LSTM model is computed based on the current vector $x_t$ and the previous vector $h_{t-1}$ with $h_t = \text{LSTM}(x_t, h_{t-1})$. The initial state and all stand LSTM parameters are randomly initialized and

tuned during training.

### 3.2 Multi-label Emotion Detection

Emotion detection aims to predict the emotion labels of posts. We follow (Wang et al., 2016) which adopts five kinds of emotions[2] in the study. Since there may be more than one emotion in a post, emotion detection can be considered as a multi-label classification task: we use $K$ emotion-specific binary perceptions ($K = 5$) to predict if the post has the corresponding emotion or not. The advantage of multi-label classification is that it learns and predicts all the emotion labels jointly.

Formally, giving an input vector $H$, a hidden layer is first used to induce a set of high-level features for each emotion $j$:

$$\tilde{H}^j = \sigma(W^j H + b^j), \quad (1)$$

and then, $H^j$ is used as inputs to a softmax output layer:

$$\hat{y}^j = \text{softmax}(\tilde{W}^j \tilde{H}^j + \tilde{b}^j) \quad (2)$$

Here, $W^j$, $b^j$, $\tilde{W}^j$, and $\tilde{b}^j$ are model parameters.

### 3.3 Training

Given the word sequence in a post, our training objective is to minimize the cross-entropy loss over a set of training examples $(x_i, y_i)|_{i=1}^N$, with a $\ell_2$-regularization term,

$$J_y(\theta_y) = -\sum_{i=1}^{N}\sum_{j=1}^{K} y_i^j \log \hat{y}_i^j + \frac{\lambda}{2}||\theta_y||^2 \quad (3)$$

where $y_i^j$ represents the label of the $j$-th emotion for $x_i$, $\theta_y$ is the set of model parameters and $\lambda$ is the parameter for $\ell_2$ regularization.

In this paper, the model parameters are optimized by AdaGrad (Duchi et al., 2011), and Skip-gram algorithm (Mikolov et al., 2013) is used for word embedding.

## 4 Neural Personal Discrimination Model

The drawback of above vanilla model is that it does not consider the deep personal correlationship among different posts.

[2]The five emotions include *happiness*, *sadness*, *anger*, *fear*, and *surprise*, please refer to "Experiments Section" of the paper for more details.

In this study, we think that the posts in social media can be connected through the authors' backgrounds. Therefore, we propose a Neural Personal Discrimination (NPD) model to connect people and learn their emotions collectively. We use *adversarial discriminators* to determine the personal attributes to construct the personal profiles, and employ *attention mechanisms* to aggregate attributes-aware words. In this way, the social correlationship between different posts can be well addressed.

Figure 1 illustrates our proposed neural personal discrimination model for emotion detection. In particular, we first learn the representation of each post using a LSTM model, same as the vanilla joint model. Then, we use adversarial discriminators to determine the personal attributes of each post. Finally, we employ attention mechanisms to aggregate the representation of informative attributes-aware words into a vector for the emotion prediction. In the following of this section, we illustrate the details of the infrastructure one by one.

### 4.1 Personal Adversarial Discriminators

A straightforward way to jointly detect personal attributes and the emotion of a post is to treat emotion detection as a multi-label classification. However, such model may not be able to separate the posts from different attributes directly, and thus fail to learn the correlationship between the emotion of a post and the personal backgrounds of the authors. To address this issue, we utilize adversarial discriminators to determine the personal attributes of the authors, and to learn the emotion and the attributes of the authors collectively. Adversarial networks have achieved much success in various studies, especially in image and text generation (Goodfellow et al., 2014; Wang and Wan, 2018; Fedus et al., 2018). In this part, we propose two adversarial discriminators, i.e, a gender discriminator and a location discriminator, to determine the personal attributes of each post.

**Gender Discriminator.** The gender discriminator is employed to determine the author's gender of each post. Let $g_i \in [0, 1]$ represents the probability of the gender label (female or male) for the gender discriminator, and $f$ is the function parameterized by $\theta_f$ which maps an embedding vector to a hidden representation $h_i^g$ from the post $x_i$. Here, the gender discriminator $G(h_i^g; \theta_g) \rightarrow \hat{g}_i$ param-

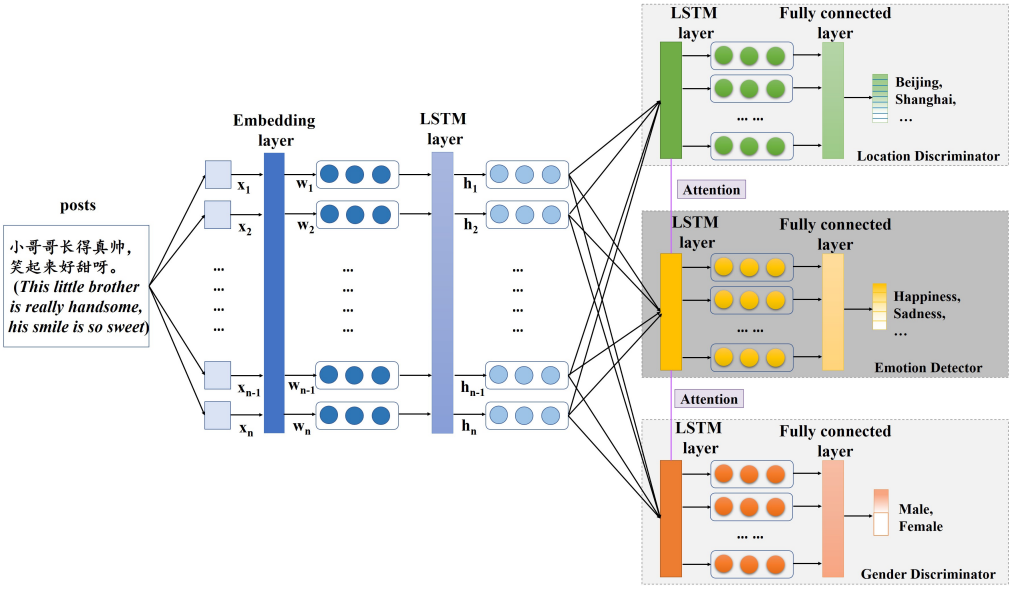


Figure 1: Overview of the neural personal discrimination model.

eterized by $\theta_g$ maps a hidden representation vector $h_i^g$ to a predicted gender label $\hat{g}_i$ with the loss function is:

$$J_{gend} = \sum_{i=1}^{N} \{g_i \log \hat{g}_i + (1-g_i) \log(1-\hat{g}_i)\}, \quad (4)$$

where $\hat{g}_i = G(f(x_i))$.

In this study, the gender discriminator is trained towards a saddle point of the loss function through maximizing the loss over $\theta_g$ while minimizing the loss over $\theta_f$ (Ganin et al., 2017).

**Location Discriminator.** The location discriminator is employed to determine the authors' location of a post[3]. Let $\ell_i^j \in [0,1]$ represents probability of the $j$-th location information of the $i$-th post and $j \in \{1, 2, \ldots, m\}$, where, $m$ is the number of provinces. The loss function is:

$$J_{loc} = \sum_{i=1}^{N} \sum_{j=1}^{m} \ell_i^j \log \hat{\ell}_i^j, \quad (5)$$

where, $\hat{\ell}_i^j = L(f(x_i))$ and $L(h_i^\ell; \theta_\ell)$ are the parameters of the location discriminator, $h_i^\ell = f(x_i)$ is the hidden represent from the post $x_i$

From the optimization of both discriminators, we can find that both $h^g$ and $h^\ell$ represent the latent feature representation of posts, which integrate the discrimination of various personal information. With the goal at $G(\theta_g)$ and $L(\theta_\ell)$ try best to determine the gender and location of the authors. The adversarial network makes use of min-max optimization.

[3]In this study, location discriminator is used to detect the author's province of a post.

### 4.2 Personal Attention Mechanisms

In emotion detection, not all words contribute equally to the representation of emotions and personal attributes. Hence, we employ attention mechanisms to extract the words that are important to the personal backgrounds of posts, and to aggregate the representations of those informative attributes-aware words. With regard to the two adversarial discriminators, we propose attention mechanisms to build two representation ($v^g$ and $v^\ell$) from the gender and location discriminators respectively, and then concatenate them together to construct the overall personal representation through the informative attributes-aware words .

**Gender Attention.** We use an attention function to aggregate the gender-aware representation of the salient words to formulate the gender attention vector $v^g$. Here, the gender attention model outputs a continuous vector $v^g \in \mathbb{R}^{d \times 1}$ recurrently by feeding the hidden representation vectors $\{h_1^g, h_2^g, \cdots, h_{n_t}^g\}$ as inputs. Specifically, $v^g$ is computed as a weighted sum of $h_i^g$ $(0 \leq i \leq n_t)$, namely

$$v^g = \sum_{i}^{n_t} \alpha_i h_i^g \quad (6)$$

where $n_t$ is the hidden variable size, $\alpha_i \in [0,1]$ is the weight of $h_i^g$, and $\sum_i \alpha_i = 1$. For each

piece of hidden state $h_i^g$, the scoring function is calculated as follows:

$$v_i^g = \tanh(W_g h_i^g + b^g) \tag{7}$$

$$\alpha_i = \frac{\exp(v^g)}{\sum_j \exp(v_j^g)} \tag{8}$$

**Location Attention.** Similar with the gender attention mechanism, the location attention model outputs a continuous vector $v^\ell \in \mathbb{R}^{d\times 1}$ recurrently by feeding the hidden representation vectors $\{h_1^\ell, h_2^\ell, \cdots, h_{n_k}^\ell\}$ as inputs. Specifically, $v^\ell$ is computed as a weighted sum of $h_i^\ell$ $(0 \le i \le n_t)$, namely

$$v^\ell = \sum_i^{n_k} \beta_i h_i^\ell \tag{9}$$

$$v_i^\ell = \tanh(W_\ell h_i^\ell + b^\ell) \tag{10}$$

$$\beta_i = \frac{\exp(v^\ell)}{\sum_j \exp(v_j^\ell)} \tag{11}$$

where, $n_k$ is the location hidden variable size, and $\beta_i$ is the same setting as $\alpha_i$.

Finally, we concatenate $v^g$ and $v^\ell$ to capture the overall personal representation through all the personal attributes discriminators.

$$V = v^g \oplus v^\ell \tag{12}$$

### 4.3 Adversarial Training with Neural Personal Discrimination

The proposed NPD model can be trained in a end-to-end manner once we obtain the loss function of the emotion detector and the attribute discriminators. Our ultimate training goal is to minimize the loss function with parameters $\theta = \{\theta_f, \theta_y, \theta_g, \theta_\ell\}$ as follow:

$$J(\theta) = \lambda_1 J_y + \lambda_2 J_{gend} + \lambda_3 J_{loc} \tag{13}$$

Where$\lambda_1$, $\lambda_2$ and $\lambda_3$ are the weight parameters to balance the importance of losses between the emotion detection and the two personal attribute discriminators.

Specifically, Eq. 13 is defined by finding a saddle point $\hat{\theta}_y, \hat{\theta}_f, \hat{\theta}_g, \hat{\theta}_\ell$ such that

$$(\hat{\theta}_f, \hat{\theta}_y) = \arg\min_{\theta_f, \theta_y} J(\theta_f, \theta_y, \hat{\theta}_g, \hat{\theta}_\ell) \tag{14}$$

$$\hat{\theta}_g = \arg\max_{\theta_g} J(\hat{\theta}_f, \hat{\theta}_y, \theta_g, \hat{\theta}_\ell) \tag{15}$$

$$\hat{\theta}_\ell = \arg\max_{\theta_\ell} J(\hat{\theta}_f, \hat{\theta}_y, \hat{\theta}_g, \theta_\ell) \tag{16}$$

As suggested previously, a saddle point is defined by Eq. 14−Eq. 16, and can be achieved as a stationary point the gradient updates:

$$\theta_f \leftarrow \theta_f - \mu(\lambda_1 \frac{\partial J_y^i}{\partial \theta_f} - \lambda_2 \frac{\partial J_{gend}^i}{\partial \theta_f} - \lambda_3 \frac{\partial J_{loc}^i}{\partial \theta_f}) \tag{17}$$

$$\theta_y \leftarrow \theta_y - \mu\lambda_1 \frac{\partial J_y^i}{\partial \theta_y} \tag{18}$$

$$\theta_g \leftarrow \theta_g - \mu\lambda_2 \frac{\partial J_{gend}^i}{\partial \theta_g} \tag{19}$$

$$\theta_\ell \leftarrow \theta_\ell - \mu\lambda_3 \frac{\partial J_{loc}^i}{\partial \theta_\ell} \tag{20}$$

where $\mu$ is the learning rate.

## 5 Experimentation

### 5.1 Experimental Settings

We collect the data from *Weibo.com*, one of the most popular SNS websites in China. We crawl all the posts and corresponding personal profiles from the website. The dataset contains 11,157 microblog posts from 839 users. We employed six graduated students to annotate the corpus with a well-defined annotation guideline. Every two annotators annotate a same part of corpus, if they have disagreement on some posts, we ask another annotator to vote with them. The annotation guideline is based on Lee et al. (2013). Five basic emotions are annotated, namely *happiness*, *sadness*, *fear*, *anger* and *surprise* (Lee et al., 2013; Wang et al., 2017). Table 1 illustrates the statistics of each emotion. From the table, we find that the frequency of *happiness* and *sadness* are similar. Moreover, the frequency of *fear* and *anger* is much less than other three emotions.

We randomly select 70% posts as the training data, and remaining 30% posts as the test data. For evaluation, F1-measure is used to evaluate the performance of proposed model in each emotion. Average F1-measure is used to evaluate the overall performance of all emotions.

The setting of hyper-parameters is: vocabulary size is 2000, batch size is set to 32, dropout ratio is 0.2, learning rate $\mu$ is set 0.0001, and $\lambda_1 : \lambda_2 : \lambda_3 = 1 : 1 : 1$

Table 2: Experimental results of different models.

| Method | F1. | | | | | |
|---|---|---|---|---|---|---|
| | Happiness | Sadness | Anger | Surprise | Fear | Average |
| SVM | 0.628 | 0.462 | 0.390 | 0.117 | 0.091 | 0.338 |
| Abdul17 | 0.656 | 0.492 | 0.429 | 0.103 | 0.111 | 0.358 |
| Vaswani17 | 0.644 | 0.494 | 0.392 | 0.113 | 0.121 | 0.353 |
| NPD | **0.657** | **0.510** | **0.459** | **0.135** | **0.127** | **0.378** |

Table 1: The statistics of emotion distribution in the dataset

| Emtion | Post Number |
|---|---|
| Happiness | 2,915 |
| Sadness | 2,454 |
| Fear | 359 |
| Anger | 153 |
| Surprise | 601 |
| None | 4,675 |

### 5.2 Experimental Results

We compare the proposed Neural Personal Discrimination (NPD) model with several representative baselines models in Table 2, where,1) **SVM** is a widely used baseline to predict the emotion of a post in social media (Yang et al., 2007).2) **Abdul17** is a standard LSTM model which consist of a LSTM layer and a fully connected layer, and it is modified from the model in Abdul-Mageed and Ungar (2017). The LSTM model yields the state-of-the-art performance on emotion detection in recent researches.3)**Vaswani17** is an improved LSTM model with a self-attention mechanism. The self-attention mechanism is used to capture the structural information and has been successfully applied in various natural language processing tasks recently (Cheng et al., 2016; Vaswani et al., 2017)

From Table 2, we find that all of the neural models outperform SVM significantly. This indicates that neural models are much more effective than discrete models in emotion detection. In addition, our proposed NPD model outperforms both the standard LSTM model (Abdul17) and the improved LSTM model with self-attention (Vaswani17) significantly. This shows the effectiveness of our proposed NPD model with both adversarial discriminators and attention mechanisms. This also shows the usefulness of personal attributes for emotion detection. Moreover, we find that the performance of Vaswani17 is even lower than the standard LSTM model. This shows that simply integrating a self-attention mechanism may not be able to well capture informative words for emotion detection in social media.

### 5.3 Analysis and Discussion

In this subsection, we analyze the influence of different factors in the proposed NPD model, and give some statistics and examples to illustrate the effectiveness of the proposed NPD model with different personal attributes.

#### 5.3.1 Influence of Personal Attributes

We illustrate the influence of personal attributes in the proposed NPD model in Table 3, where, 1)**LSTM-attributes** is a LSTM based multi-label classification model, which predicts both the emotion and the attribute labels of each post collectively.2)**NPD-gender** ablates the location attribute, i.e., only considering the gender attributes in the NPD model.3) **NPD-location** ablates the gender attribute, i.e., only considering the location attributes in the NPD model.

From the table, we can find that the performance of LSTM-attributes is much lower than LSTM model. This indicates that simple multi-label classification setting is not effective for integrating personal attributes. This may be due to the fact that basic multi-label setting fails to learn the correlationship between the emotions of posts and the personal attributes of the authors well. In addition, both the NPD-gender and the NPD-location perform better than the LSTM model respectively. This shows the effectiveness of the proposed NPD model with both adversarial and attention networks, and the usefulness of both gender and location attributes. Finally, the proposed NPD model with all the attributes significantly outperforms all the other models. This suggests that we should integrate all the personal attributes for emotion detection in social media.

Table 3: Comparison of various models with different personal attributes.

| Method | Average F1. |
|---|---|
| LSTM | 0.358 |
| LSTM-attributes | 0.341 |
| NPD-gender | 0.360 |
| NPD-location | 0.363 |
| NPD | **0.378** |

Table 4: Influence of network structures.

| Method | Average F1. |
|---|---|
| LSTM | 0.358 |
| LSTM-attention | 0.363 |
| LSTM-adversarial | 0.375 |
| NPD | **0.378** |

#### 5.3.2 Influence of Network Structures

After analyzing the influence of different attributes, we analyze the influence of network structures in Table 4, where **LSTM-attention** ablates the adversarial discriminators, and only utilizes attention mechanisms with a multi-label classification setting, and **LSTM-adversarial** ablates the attention mechanisms and only utilizes adversarial discriminators for emotion detection.

From Table 4, we can see that both the attention mechanisms (LSTM-attention) and the adversarial discriminators (LSTM-adversarial) are effective in emotion detection. Moreover, the adversarial discriminators are much more effective than the attention mechanisms. This shows that the personal attribute discriminator is more important than learning informative attributes-aware words. Moreover, the NPD model obtains the best results by integrating both adversarial discriminators and attention mechanisms.

#### 5.3.3 Statistics

We give the statistics of gender and location to explore the correlationship between emotion and personal attributes.

**Distribution of Gender.** Figure 2 illustrates the distribution of emotions between genders. Here, the Y-axis is conditional probability of each emotion given gender. From the figure, we can find that women tend to express the sadness emotion, while men tend to express anger emotion. This may be due to the fact that the different personality has different emotion expressions, i.e., sentimentality of the female and the impulsion of the male.

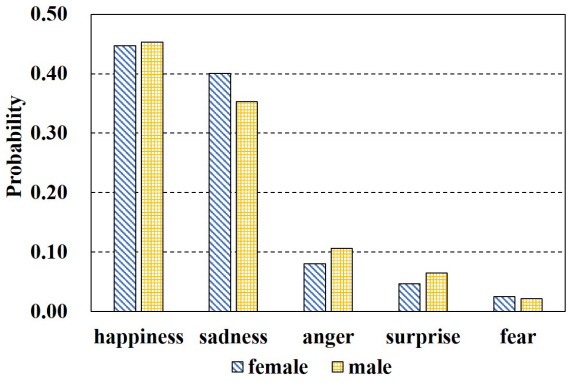


Figure 2: Distribution of gender.

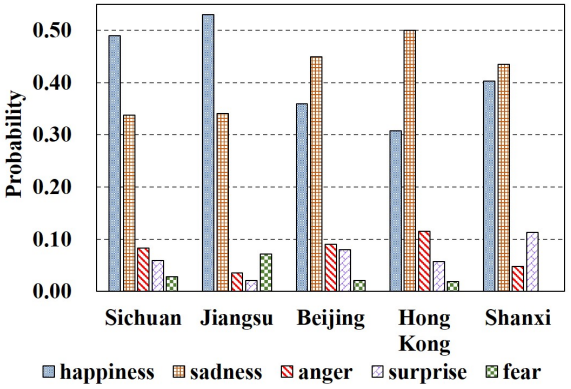


Figure 3: Distribution of location.

**Distribution of Location.** Figure 3 is an example of the distribution of emotions between locations. As discussed in the above section, we use the province of authors as location attributes. Here, the Y-axis represents the conditional probability of each emotion given location. From the figure, we can find that the authors' location can may influence their emotions in many aspects. For example, people tend to express the positive (i.e., happiness) emotion than the negative emotion in Jiangsu. One of the most comfortable and developed regions in China. Due to air pollution and populations, people tend to express the negative emotion than the positive emotion in Beijing. In addition, people in Hong Kong always feel crowding and tend to express the sadness emotion. Finally, people always feel happy and comfortable in Sichuan, well known as "country of paradise" in China.

From the results of statistics, we can see that the personal attributes, like gender, location, can affect the emotion detection. Also, the experimen-

Table 5: Examples predicted labels of LSTM and NPD

| Post | LSTM | NPD |
|---|---|---|
| [E3]@嬤嬤俊, 连九份芋圆都会做, 巴适惨了, 冰冰凉Q弹水果甜品. (*@Momo Jun, this guy can even make Taro Balls from Jiufen of Taiwan. It is so bash. This ice fruit dessert is so soft and chewy.*) | Sadness | Happiness |
| [E4]下期节目, 不容错过, 有我的男神出场. (*Next tv show is not to be missed, because my Mr. Mcdreamy will be coming.*) | None | Happiness |
| [E5]我这身体不敢恭维啊, 吃了好多的冰, 结果痛经了吧, 赶紧多喝点红糖姜茶. (*I can't compliment my body. Eating too much ice leads to my dysmenorrhea. I must to drink more brown sugar ginger tea*) | None | Sadness |

tal results of personal attribute influence shows the same conclusion.

#### 5.3.4 Case Study

We select three examples from the test set to evaluate the effectiveness of the proposed NPD model for better comparison with the LSTM model in Table 5. In [E3], through "bashi(*conformable*)" is a strong cue word of the *happiness* emotion, it is treated as a general word without indicating any location information from the LSTM model. At the same time, the location information (Sichuan Province) determined by the location discriminator and the attention mechanism makes "bashi(*conformable*)" critical in the NPD model, and these enable the NPD model determines the correct *happiness* emotion. In [E4] and [E5], both posts contain implicit gender information. For example, "Mr. Mcdreamy" implies the female's affection for the handsome male, and "dysmenorrhea" is a female physiological disease. Without the background of such gender information, it is impossible to infer any potential emotional information from these two words. This explains why the LSTM model fails to detect any emotion from the examples (*None* of emotion). However, our NPD model successfully determines the gender of these two posts by the gender discriminator, and improves the weight of these two words for emotion detection by the attention mechanisms. In this way, the emotions of the three examples are correctly detected through the proposed NPD model.

## 6 Conclusion

Most of previous studies consider each post individually in emotion detection, one of the most important tasks in sentiment analysis. However, since the posts in social media are generated by users, it is natural that these posts can be connected through authors' personal background attributes. In this paper, we propose a neural personal discrimination model with both adversarial discriminators and attention mechanisms to connect posts with similar personal attributes. Here, the adversarial discriminators are used to determine the personal attributes, and the attention mechanisms are employed to aggregate attributes-aware words. In this way the social correlations between different posts can be captured. The experimental results show the usefulness of the personal attributes and the effectiveness of our proposed neural personal discrimination model in modeling such personal attributes with significant performance improvement over the state-of-the-art baselines.

## Acknowledgment

This work was support by National Natural Science Foundation of China (No. 61702518, No.61806137, No.61525205) and a project funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions.. This work was also supported by the joint research project of Alibaba and Soochow University. Finally, we would like to thank the anonymous reviewers for their insightful comments and suggestions.

## References

Muhammad Abdul-Mageed and Lyle H. Ungar. 2017. Emonet: Fine-grained emotion detection with gated recurrent neural networks. In *ACL*, pages 718–728.

Plaban Kumar Bhowmick, Anupam Basu, Pabitra Mitra, and Abhishek Prasad. 2009. Multi-label text classification approach for sentence level news emotion analysis. In *ICPRAI*, pages 261–266.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. pages 551–561.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Niko Colnerič and Janez Demsar. 2018. Emotion recognition on twitter: Comparative study and training a unison model. *IEEE Transactions on Affective Computing*, PP(99):1–14.

John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

William Fedus, Ian J. Goodfellow, and Andrew M. Dai. 2018. Maskgan: Better text generation via filling in the _____. In *ICLR*, pages 1–16.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor S. Lempitsky. 2017. Domain-adversarial training of neural networks. pages 189–209.

Yoav Goldberg. 2016. A primer on neural network models for natural language processing. *J. Artif. Intell. Res.*, 57:345–420.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *NIPS*, pages 2672–2680.

Umang Gupta, Ankush Chatterjee, Radhakrishnan Srikanth, and Puneet Agrawal. 2017. A sentiment-and-semantics-based approach for emotion detection in textual conversations. *CoRR*, abs/1707.06996. Version 4.

Dirk Hovy. 2015. Demographic factors improve classification performance. In *ACL*, pages 752–762.

Paul F Lazarsfeld, Robert K Merton, et al. 1954. Friendship as a social process: A substantive and methodological analysis. *Freedom and control in modern society*, 18(1):18–66.

Sophia Yat Mei Lee, Ying Chen, Chu-Ren Huang, and Shoushan Li. 2013. Detecting emotion causes with a linguistic rule-based approach. *Computational Intelligence*, 29(3):390–416.

Junjie Li, Haitong Yang, and Chengqing Zong. 2016. Sentiment classification of social media text considering user attributes. In *NLPCC/ICCPOL*, pages 583–594.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Xiaojun Quan, Qifan Wang, Ying Zhang, Luo Si, and Wenyin Liu. 2015. Latent discriminative models for social emotion detection with emotional dependency. *ACM Trans. Inf. Syst.*, 34(1):2:1–2:19.

Kirk Roberts, Michael A. Roach, Joseph Johnson, Josh Guthrie, and Sanda M. Harabagiu. 2012. Empatweet: Annotating and detecting emotions on twitter. In *LREC*, pages 3806–3813.

Carlo Strapparava and Rada Mihalcea. 2008. Learning to identify emotions in text. In *SAC*, pages 1556–1560.

Mike Thelwall. 2010. Emotion homophily in social network site messages. *First Monday*, 15(4):1–8.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*, pages 6000–6010.

Soroush Vosoughi, Helen Zhou, and Deb Roy. 2016. Enhanced twitter sentiment classification using contextual information. *CoRR*, abs/1605.05195. Version 1.

Ke Wang and Xiaojun Wan. 2018. Sentigan: Generating sentimental texts via mixture adversarial networks. In *IJCAI*, pages 4446–4452.

Xuren Wang and Qiuhui Zheng. 2013. Text emotion classification research based on improved latent semantic analysis algorithm. In *ICCSEE*, pages 210–213.

Zhongqing Wang, Sophia Yat Mei Lee, Shoushan Li, and Guodong Zhou. 2017. Emotion analysis in code-switching text with joint factor graph model. *IEEE/ACM Trans. Audio, Speech & Language Processing*, 25(3):469–480.

Zhongqing Wang, Shoushan Li, Hanxiao Shi, and Guodong Zhou. 2014. Skill inference with personal and skill connections. In *COLING*, pages 520–529.

Zhongqing Wang, Yue Zhang, Sophia Yat Mei Lee, Shoushan Li, and Guodong Zhou. 2016. A bilingual attention network for code-switched emotion prediction. In *COLING*, pages 1624–1634.

Changhua Yang, Kevin Hsin-Yih Lin, and Hsin-Hsi Chen. 2007. Emotion classification using web blog corpora. In *WI-IAT*, pages 275–278.

Lu Zhang, Liangqing Wu, Shoushan Li, Zhongqing Wang, and Guodong Zhou. 2018. Cross-lingual emotion classification with auxiliary and attention neural networks. In *NLPCC*, pages 429–441.

# Specificity-Driven Cascading Approach for Unsupervised Sentiment Modification

**Pengcheng Yang[1], Junyang Lin[2], Jingjing Xu[2], Jun Xie[3], Xu Sun[2], Qi Su[2]**
[1] Deep Learning Lab, Beijing Institute of Big Data Research, Peking University
[2] MOE Key Lab of Computational Linguistics, School of EECS, Peking University
[3] Tencent, Beijing, China
{yang_pc,linjunyang,jingjingxu,xusun,sukia}@pku.edu.cn
stiffxie@tencent.com

## Abstract

The task of unsupervised sentiment modification aims to reverse the sentiment polarity of the input text while preserving its semantic content without any parallel data. Most previous work follows a two-step process. They first separate the content from the original sentiment, and then directly generate text with the target sentiment only based on the content produced by the first step. However, the second step bears both the target sentiment addition and content reconstruction, thus resulting in a lack of specific information like proper nouns in the generated text. To remedy this, we propose a specificity-driven cascading approach in this work, which can effectively increase the specificity of the generated text and further improve content preservation. The experiments show that our approach outperforms competitive baselines by a large margin, which achieves 11% and 38% relative improvements of the overall metric on the Yelp and Amazon datasets, respectively.

## 1 Introduction

The goal of unsupervised sentiment modification is to reverse the sentiment polarity of the input text while preserving its semantic content without any parallel data. It not only has a variety of practical applications, e.g., fighting offensive language on social media (dos Santos et al., 2018), but also serves as an ideal testbed for controllable text generation. However, the input for this task usually contains some specific information like proper nouns. For instance, "*Michael*" in Table 1 is a human name, which belongs to the specific information. This specific information is important because it is often the subject or object of the sentence and the bearer of sentiment. Therefore, it needs to be fully preserved in the process of sentiment modification. We refer this attribute as the specificity of the output.

| **Input** | ***Michael** 's work exceeds my expectations.* |
|---|---|
| Shen et al. (2017) | *Service is terrible.* |
| Fu et al. (2018) | *I was very disappointed.* |
| Xu et al. (2018) | *The girls do horrible work.* |

Table 1: Examples generated by four different previous methods. "***Michael***" is a proper noun, which belongs to specific information.

Most previous work on unsupervised sentiment modification follows a two-step process. They first separate the content from the original sentiment, either in an implicit (Hu et al., 2017; Shen et al., 2017; Fu et al., 2018; Tsvetkov et al., 2018) or explicit (Xu et al., 2018; Zhang et al., 2018a) way. Then, they directly generate text with the target sentiment only based on the content produced by the first step. In the second step, the decoder does not include the sentiment information in its source input, meaning that the source information is incomplete for generating a sentiment-transferred sentence. This *information gap* at both ends of the decoder causes the decoder needs to bear both the target sentiment addition and content reconstruction. Thus, it is difficult for the decoder to balance sentiment transformation and content preservation simultaneously, resulting in the loss of specific information contained in the source input during decoding. This not only makes the generated text lacks specificity, but also causes poor content preservation. Table 1 intuitively illustrates this problem. All three traditional methods miss proper noun "*Michael*" in the output.

In order to generate text containing more specific information, in this paper, we propose a specificity-driven cascading approach. Our approach consists of three modules: separation module, emotionalization module, and fusion module. The separation module is responsible for explicitly separating sentiment words from content words



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5508–5517,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

via the self-attention mechanism. Then, different from previous work, we divide the process of generating text with the target sentiment based on content words into two steps. First of all, in order to achieve sentiment addition, the emotionalization module generates corresponding target sentiment words based on the content words. Then, the fusion module performs content reconstruction by fusing semantic content and target sentiment words to generate sentiment-transferred sentences. Each step is dedicated to their respective goals, which reduces the burden of the decoder in the fusion module. Besides, with the help of the target sentiment words generated by the emotionalization module, the *information gap* is also eliminated. This allows specific information contained in the input to be well preserved, resulting in more specific output and better content preservation.

The main contributions of this paper are summarized as follows:

- We propose an effective specificity-driven cascading approach for unsupervised sentiment modification, which not only increases the specificity of the generated text, but also further improves content preservation.
- Experimental results show that our approach can outperform the competitive baselines by a large margin. Further analysis demonstrates that the proposed method can achieve better reconcile of sentiment transformation and content preservation.

## 2 Background

Here we introduce some necessary background knowledge. The core component of our proposed approach is the Seq2Seq model (Sutskever et al., 2014; Bahdanau et al., 2014), which usually consists of an encoder and a decoder with the attention mechanism.

**Encoder:** Given the input $\boldsymbol{x} = (x_1, \cdots, x_m)$, the encoder implemented as a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) structure reads $\boldsymbol{x}$ from both directions and computes the hidden states for each word,

$$\overrightarrow{h}_i = \overrightarrow{\text{LSTM}}(\overrightarrow{h}_{i-1}, x_i) \quad (1)$$

$$\overleftarrow{h}_i = \overleftarrow{\text{LSTM}}(\overleftarrow{h}_{i+1}, x_i) \quad (2)$$

where $e(x_i)$ denotes the embedding of the word $x_i$. The final hidden representation of the $i$-th word is $h_i = [\overrightarrow{h}_i; \overleftarrow{h}_i]$, which semicolon represents vector concatenation.

| Module | Source Input | Target Output | Role |
|---|---|---|---|
| $\textbf{SC}_{\phi}$ | $\boldsymbol{x}^+/\boldsymbol{x}^-$ | $+/-$ | Separate content and sentiment |
| $\textbf{S2S}_{\theta_{pe}}$ | $\boldsymbol{c}^+$ | $\boldsymbol{s}^+$ | Generate positive sentiment words |
| $\textbf{S2S}_{\theta_{ne}}$ | $\boldsymbol{c}^-$ | $\boldsymbol{s}^-$ | Generate negative sentiment words |
| $\textbf{S2S}_{\theta_{pf}}$ | $(\boldsymbol{c}^+, \boldsymbol{s}^+)$ | $\boldsymbol{x}^+$ | Generate positive sentence |
| $\textbf{S2S}_{\theta_{nf}}$ | $(\boldsymbol{c}^-, \boldsymbol{s}^-)$ | $\boldsymbol{x}^-$ | Generate negative sentence |

Table 2: The role of each component of our approach at the training stage. We use "+" and "-" to represent positive and negative sentiment, respectively.

**Decoder:** Given the hidden representations $(h_1, \cdots, h_m)$, the decoder generates words sequentially. Besides, the attention mechanism is used to automatically select the most informative words at different time steps. In detail, the hidden state $s_{t+1}$ of the decoder at time-step $t+1$ is computed as follows:

$$e_{t,i} = f(s_t, h_i) \quad (3)$$

$$\alpha_{t,i} = \frac{\exp(e_{t,i})}{\sum_{j=1}^{m} \exp(e_{t,j})} \quad (4)$$

$$c_t = \sum_{i=1}^{m} \alpha_{t,i} h_i \quad (5)$$

$$s_{t+1} = \text{LSTM}(s_t, [e(y_t); c_t]) \quad (6)$$

$$y_t \sim \text{softmax}\left(g(s_t)\right) \quad (7)$$

where $[e(y_t); c_t]$ means the concatenation of vectors $e(y_t)$ and $c_t$. $e(y_t)$ is the embedding of the word $y_t$ that has the highest probability. $f(s_t, h_i)$ is an aligned model, which measures the dependency between $s_t$ and $h_i$. $g$ is a function, which is used to transform the hidden state $s_t$ into the probability distribution over the vocabulary space.

The model is trained by maximizing the conditional likelihood of the ground-truth sequence $\boldsymbol{y}^* = (y_1^*, \cdots, y_n^*)$. Specially, the objective is to minimize the cross-entropy loss:

$$\mathcal{L} = -\sum_{t=1}^{n} \log\left(p(y_t^* | \boldsymbol{y}_{<t}^*, \boldsymbol{x})\right) \quad (8)$$

[1]

## 3 Methodology

### 3.1 Overview

Here we define some notations and describe the sentiment modification task. We use $\mathcal{D}^+ =$

[1] In this work, $\boldsymbol{y}_{<t}^*$ denotes the sequence $(y_1^*, \cdots, y_{t-1}^*)$.

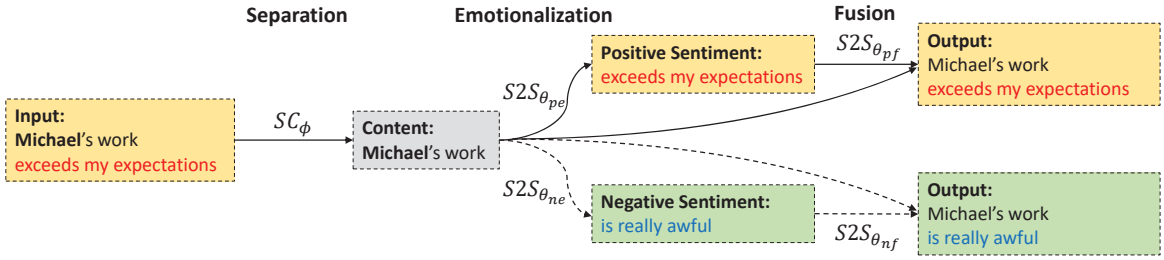


Figure 1: The overview of the proposed cascading approach with a positive input. The process with a negative input is in a similar way. Solid and dashed lines indicate the training process and the testing process, respectively.

$\{\boldsymbol{x}_i^+\}_{i=1}^{n_1}$ and $\mathcal{D}^- = \{\boldsymbol{x}_i^-\}_{i=1}^{n_2}$ to represent the positive corpus and negative corpus, respectively, where $\boldsymbol{x}^+$ ($\boldsymbol{x}^-$) is a positive (negative) sentence. $\mathcal{D} = \mathcal{D}^+ \cup \mathcal{D}^-$ denotes the complete corpus. The sentiment modification task aims to take the sentence $\boldsymbol{x}^+$ ($\boldsymbol{x}^-$) as input and generate a sentiment-transferred sentence $\hat{\boldsymbol{x}}^-$ ($\hat{\boldsymbol{x}}^+$) as output.

The overview of our approach is shown in Figure 1. Our approach consists of three modules: separation module, emotionalization module, and fusion module. The separation module aims to explicitly separate the sentiment words from content words and the emotionalization module receives content words as input to generate the corresponding sentiment words. Finally, the fusion module fuses the content and sentiment words to generate fluent sentences. Table 2 summarizes the source input, target output, and function of each component at the training stage. Besides, the training algorithm and testing algorithm are shown in Algorithm 1 and Algorithm 2, respectively.

### 3.2 Separation Module

The separation module aims to explicitly separate the sentiment words from content words. Given a sentence $\boldsymbol{x} = (x_1, \cdots, x_m)$, the separation can be accomplished by identifying whether each word $x_i$ is a sentiment word. Here we employ a sentiment classifier with the self-attention mechanism to achieve this goal. The classifier with the self-attention mechanism can select the most informative words automatically by assigning higher attention weights to sentiment words. Therefore, the attention weight $\alpha_i$ of the $i$-th word can be used to identify whether $x_i$ is a sentiment word.

In detail, we use the bidirectional LSTM model for implementation. The LSTM model reads the input $\boldsymbol{x}$ from both directions and computes the hidden representation $h_i = [\overrightarrow{h}_i; \overleftarrow{h}_i]$ for the $i$-th word, where semicolon represents vector concatenation. The self-attention mechanism produces final context vector $h$ by aggregating the hidden representations $(h_1, \cdots, h_m)$. Specifically,

$$\alpha_i = \text{softmax}(v^T h_i) \tag{9}$$

$$h = \sum_{i=1}^{m} \alpha_i h_i \tag{10}$$

where $v$ is a parameter vector. At the training stage, we perform classification based on the final context vector $h$ and the loss function is the cross-entropy loss. Then, for the well-trained sentiment classifier, the average attention value of sentence $\boldsymbol{x}$ is calculated as:

$$\bar{\alpha} = \frac{1}{m} \sum_{i=1}^{m} \alpha_i \tag{11}$$

Words with attention weight larger than $\bar{\alpha}$ can be regarded as sentiment words, and the remaining words can be regarded as content words.[2] Therefore, each sentence $\boldsymbol{x}$ can be separated into content words $\boldsymbol{c}$ and sentiment words $\boldsymbol{s}$.[3]

### 3.3 Emotionalization Module

In previous work, there is an *information gap* at both ends of the decoder, making the decoder to bear both the target sentiment addition and content reconstruction simultaneously. To eliminate the *information gap* and ease the burden of the decoder, we divide the process of generating sentiment sentences based on content into two separate steps: sentiment addition and content reconstruction. Each step is dedicated to their respective goals, which reduces the burden of the

[2]The reason is that there exists an obvious gap between the attention weights of content words and sentiment words, causing the weights of these two types of words to be distributed at both ends of the average. Exploratory experiments show that the separation model can achieve more than 80% $F_1$ score.

[3]Except for Section 2, we use $\boldsymbol{c}$ and $\boldsymbol{s}$ to represent content words and sentiment words, respectively. This is different from the context vector $c$ and hidden state $s$ in Section 2.

**Algorithm 1** The training algorithm.

**Require:** positive corpus $\mathcal{D}^+ = \{\boldsymbol{x}_i^+\}_{i=1}^{n_1}$; negative corpus $\mathcal{D}^- = \{\boldsymbol{x}_i^-\}_{i=1}^{n_2}$; complete corpus $\mathcal{D} = \mathcal{D}^+ \cup \mathcal{D}^-$

**Separation:**
  Train the sentiment classifier $\mathbf{SC}_\phi$ on corpus $\mathcal{D}$
  Separate positive sentence $\boldsymbol{x}^+$ in $\mathcal{D}^+$ into $(\boldsymbol{c}^+, \boldsymbol{s}^+)$ using $\mathbf{SC}_\phi$ to construct new positive corpus $\mathcal{N}^+ = \{(\boldsymbol{c}_i^+, \boldsymbol{s}_i^+, \boldsymbol{x}_i^+)\}_{i=1}^{n_1}$
  Separate negative sentence $\boldsymbol{x}^-$ in $\mathcal{D}^-$ into $(\boldsymbol{c}^-, \boldsymbol{s}^-)$ using $\mathbf{SC}_\phi$ to construct new negative corpus $\mathcal{N}^- = \{(\boldsymbol{c}_i^-, \boldsymbol{s}_i^-, \boldsymbol{x}_i^-)\}_{i=1}^{n_2}$
**Emotionalization:**
  Train $\mathbf{S2S}_{\theta_{pe}}$ using MLE on new positive corpus $\mathcal{N}^+$
  Train $\mathbf{S2S}_{\theta_{ne}}$ using MLE on new negative corpus $\mathcal{N}^-$
**Fusion:**
  Train $\mathbf{S2S}_{\theta_{pf}}$ using MLE on new positive corpus $\mathcal{N}^+$
  Train $\mathbf{S2S}_{\theta_{nf}}$ using MLE on new negative corpus $\mathcal{N}^-$

decoder. Here the emotionalization module explicitly performs sentiment addition by generating corresponding sentiment words based on content words.

In detail, through the separation module, the sentence $\boldsymbol{x}$ can be separated into $(\boldsymbol{c}, \boldsymbol{s})$, where $\boldsymbol{c}$ and $\boldsymbol{s}$ represent the content words and sentiment words, respectively. Therefore, the new corpus $\mathcal{N}^+ = \{(\boldsymbol{c}_i^+, \boldsymbol{s}_i^+, \boldsymbol{x}_i^+)\}_{i=1}^{n_1}$ can be constructed by explicitly separating the positive sentence $\boldsymbol{x}^+$. Similarly, $\mathcal{N}^- = \{(\boldsymbol{c}_i^-, \boldsymbol{s}_i^-, \boldsymbol{x}_i^-)\}_{i=1}^{n_2}$ can be constructed by separating the negative sentence $\boldsymbol{x}^-$.

The emotionalization module aims to automatically generate corresponding sentiment words according to the content. Since corpus $\mathcal{N}^+$ (or $\mathcal{N}^-$) provides pseudo-parallel data about content words $\boldsymbol{c}^+$ (or $\boldsymbol{c}^-$) and sentiment words $\boldsymbol{s}^+$ (or $\boldsymbol{s}^-$), here we use the Seq2Seq model to accomplish this goal. We train a Seq2Seq model $\mathbf{S2S}_{\theta_{pe}}$ on the new positive corpus $\mathcal{N}^+$, which is responsible for generating positive sentiment words $\boldsymbol{s}^+$ based on the content $\boldsymbol{c}^+$. Another Seq2Seq model $\mathbf{S2S}_{\theta_{ne}}$ is trained on the new negative corpus $\mathcal{N}^-$ to generate negative sentiment words $\boldsymbol{s}^-$ based on content words $\boldsymbol{c}^-$. Therefore, for the well-trained model $\mathbf{S2S}_{\theta_{pe}}$ (or $\mathbf{S2S}_{\theta_{ne}}$), given content words $\boldsymbol{c}$, $\mathbf{S2S}_{\theta_{pe}}$ (or $\mathbf{S2S}_{\theta_{ne}}$) is able to generate corresponding positive (or negative) sentiment words $\boldsymbol{s}^+$ (or $\boldsymbol{s}^-$).

### 3.4 Fusion Module

The fusion module aims to perform content reconstruction by fusing content words $\boldsymbol{c}$ and sentiment words $\boldsymbol{s}$ to generate fluent emotional sentence $\boldsymbol{x}$. Compared with the previous work, the decoder here has extra sentiment information $\boldsymbol{s}$, thus eliminating the *information gap* at both ends of the decoder. This allows the specific information to be better preserved during the generation process, leading to more specific generation results and better content preservation.

**Algorithm 2** The testing algorithm.

**Require:** positive source input $\boldsymbol{x}^+$

Separate $\boldsymbol{x}^+$ using $\mathbf{SC}_\phi$ to get content words $\boldsymbol{c}^+$
Generate the negative sentiment words $\boldsymbol{s}^-$ using $\mathbf{S2S}_{\theta_{ne}}$ with input $\boldsymbol{c}^+$
Generate the negative sentence $\hat{\boldsymbol{x}}^-$ using $\mathbf{S2S}_{\theta_{nf}}$ with input $(\boldsymbol{c}^+, \boldsymbol{s}^-)$

Similar to the emotionalization module, the fusion module is also composed of two Seq2Seq models, one for generating positive sentences and the other for generating negative sentences. Here each Seq2Seq model consists of two encoders and one decoder. The two encoders are responsible for compressing the semantic content words and emotional words into a dense vector, respectively. The decoder aims to generate fluent emotional sentence based on compressed content vector and sentiment vector. In detail, we train two bi-encoder based Seq2Seq models $\mathbf{S2S}_{\theta_{pf}}$ and $\mathbf{S2S}_{\theta_{nf}}$ on the new positive corpus $\mathcal{N}^+$ and negative corpus $\mathcal{N}^-$, respectively. Each model aims to output $\boldsymbol{x}$ based on the input $(\boldsymbol{c}, \boldsymbol{s})$. Note that the Seq2Seq model here has two encoders, so we need to make modifications to the relevant formulas. Two attention mechanisms similar to Eq. 3 - Eq. 5 aggregate the hidden representations of the content words $\boldsymbol{c}$ and sentiment words $\boldsymbol{s}$ respectively to form the content-side context $c_t^c$ and the sentiment-side context $c_t^s$. The hidden state $s_{t+1}$ of the decoder at time-step $t+1$ is computed as follows:

$$s_{t+1} = \mathrm{LSTM}(s_t, [e(y_t); c_t^c; c_t^s]) \qquad (12)$$

### 3.5 Training and Testing

At the training stage, each module is trained by optimizing the cross-entropy loss function. The testing algorithm is shown in Algorithm 2. Without loss of generality, we focus on reversing the positive input to the negative output. The process of reversing negative sentence to positive sentence is in a similar way. Given a positive input $\boldsymbol{x}^+$, the separation module can separate $\boldsymbol{x}^+$ into content words $\boldsymbol{c}^+$ and sentiment words $\boldsymbol{s}^+$. The model $\mathbf{S2S}_{\theta_{ne}}$ in the emotionalization module takes $\boldsymbol{c}^+$ as input to generate corresponding negative sentiment words $\boldsymbol{s}^-$. Finally, $(\boldsymbol{c}^+, \boldsymbol{s}^-)$ is inputted into the model

| Dataset | Training | Validation | Test |
|---|---|---|---|
| Yelp | 580K | 15K | 4K |
| Amazon | 230K | 10K | 2K |

Table 3: The statistics of two datasets.

$\mathbf{S2S}_{\theta_{nf}}$ in the fusion module to generate a fluent negative sentence $\hat{\boldsymbol{x}}^{-}$.

## 4 Experiments

In this section, we first introduce the datasets, evaluation metrics, experimental details, and all compared baselines.

### 4.1 Datasets

We conduct experiments on two datasets, Yelp[4] and Amazon[5] (He and McAuley, 2016). Both datasets are composed of a large number of reviews. Following previous work (Shen et al., 2017), reviews with scores below 3 are labeled as negative, and reviews with scores above 3 are labeled as positive. The reviews which exceed 20 words or less than 5 words are further filtered out. In addition, we train the sentiment classifier (Kim, 2014) to filter samples with the category confidence below 0.8. We divide each dataset into training, validation and test sets. The statistics of the two processed datasets is shown in Table 3.

### 4.2 Evaluation Metrics

In this paper, we adopt two evaluation methods: automatic evaluation and human evaluation.

#### 4.2.1 Automatic Evaluation

Following previous work (Xu et al., 2018; Shen et al., 2017), we perform automatic evaluation in terms of sentiment transformation and content preservation.

- **ACC:** We pre-train a sentiment classifier, which is implemented as CNN (Kim, 2014) structure, to evaluate whether the sentiment of the generated text matches the target sentiment. The pre-trained sentiment classifier achieves the accuracy of 87% and 89% on Yelp and Amazon datasets, respectively. The higher ACC value indicates better sentiment transformation.

[4] https://www.yelp.com/dataset/challenge

[5] https://www.amazon.com/

| | Yelp | | Amazon | |
|---|---|---|---|---|
| **Module** | **LSTM Layer** | **Hidden Size** | **LSTM Layer** | **Hidden Size** |
| $\mathbf{SC}_{\phi}$ | 2 | 512 | 2 | 256 |
| $\mathbf{S2S}_{\theta_{pe}}$ | (2, 3) | (256, 512) | (2, 2) | (256, 512) |
| $\mathbf{S2S}_{\theta_{ne}}$ | (2, 2) | (256, 512) | (1, 2) | (128, 256) |
| $\mathbf{S2S}_{\theta_{pf}}$ | (2, 3) | (256, 512) | (2, 2) | (256, 512) |
| $\mathbf{S2S}_{\theta_{nf}}$ | (2, 2) | (256, 512) | (1, 2) | (128, 256) |

Table 4: Main hyper-parameters. For the LSTM layer and hidden size, we use $(\cdot, \cdot)$ to represent the hyper-parameter of the encoder and decoder, respectively.

- **BLEU:** We use the BLEU score (Papineni et al., 2002) between the transferred sentence and the source sentence to evaluate content preservation. The higher BLEU score indicates better content preservation.

- **G-Score:** In order to evaluate the overall performance of different systems, following Xu et al. (2018), we compute the geometric mean (G-score) of ACC value and BLEU score as an overall evaluation metric.

#### 4.2.2 Human Evaluation

In order to evaluate the quality of generated sentences more accurately, we also perform human evaluation. Each item contains the source input, transfer direction, and the transferred sentences generated by different models. Then 200 items are distributed to 3 annotators with the linguistic background. The annotators are required to score the generated sentences from 1 to 5 based on the source input and transfer direction in terms of three criteria: sentiment, content, and overall performance. Sentiment represents whether the generated text matches the target sentiment. Content evaluates the content preservation degree. For each dataset and each metric, the average Pearson correlation coefficient of the scores given by three annotators is greater than 0.65, which indicates that the human scores are highly consistent.

### 4.3 Baselines

We compare our proposed approach with the following competitive systems:

- **Cross-Aligned Auto-Encoder (CAAE)** is proposed by Shen et al. (2017). They propose to use the refined alignment of latent representations in hidden layers to perform sentiment modification.

| Yelp | ACC | BLEU | G-Score |
|---|---|---|---|
| CAAE (Shen et al., 2017) | 50.20 | 1.35 | 8.23 |
| MDAE (Fu et al., 2018) | 81.30 | 3.27 | 16.30 |
| SEAE (Fu et al., 2018) | 42.25 | 19.18 | 28.47 |
| BT (Tsvetkov et al., 2018) | 61.45 | 2.56 | 12.54 |
| CRL (Xu et al., 2018) | 73.70 | 35.62 | 51.24 |
| Proposed Method | 68.65 | 47.42 | **57.06** |
| **Amazon** | **ACC** | **BLEU** | **G-Score** |
| CAAE (Shen et al., 2017) | 51.75 | 2.17 | 10.60 |
| MDAE (Fu et al., 2018) | 67.85 | 4.54 | 17.55 |
| SEAE (Fu et al., 2018) | 47.50 | 28.79 | 36.98 |
| BT (Tsvetkov et al., 2018) | 53.75 | 3.73 | 14.16 |
| CRL (Xu et al., 2018) | 65.20 | 22.66 | 38.44 |
| Proposed Method | 56.30 | 49.83 | **52.97** |

Table 5: Automatic evaluations of our method and baselines, from which we can see that our approach achieves the best overall performance on both datasets.

| Yelp | Sentiment | Content | Overall |
|---|---|---|---|
| CAAE (Shen et al., 2017) | 3.48 | 1.87 | 2.03 |
| MDAE (Fu et al., 2018) | 3.56 | 2.07 | 3.11 |
| SEAE (Fu et al., 2018) | 2.67 | 2.71 | 3.26 |
| BT (Tsvetkov et al., 2018) | 1.81 | 3.14 | 2.52 |
| CRL (Xu et al., 2018) | 3.64 | 3.03 | 3.89 |
| Proposed Method | 3.31 | 4.21 | **4.09** |
| **Amazon** | **Sentiment** | **Content** | **Overall** |
| CAAE (Shen et al., 2017) | 3.37 | 2.74 | 2.72 |
| MDAE (Fu et al., 2018) | 3.31 | 2.89 | 2.58 |
| SEAE (Fu et al., 2018) | 2.82 | 3.05 | 3.13 |
| BT (Tsvetkov et al., 2018) | 2.04 | 2.58 | 2.94 |
| CRL (Xu et al., 2018) | 3.68 | 3.21 | 3.06 |
| Proposed Method | 3.42 | 3.83 | **3.71** |

Table 6: Human evaluations of different systems, showing that our approach outperforms the baselines by a large margin, especially in content preservation.

- **Multi-Decoder Auto-Encoder (MDAE)** is proposed by Fu et al. (2018). They use a separate decoder for each sentiment to generate transferred sentences based on the extracted content vectors.
- **Style-Embedding Auto-Encoder (SEAE)** is also proposed by Fu et al. (2018). Each sentiment has a unique embedding, which is used to control the sentiment transfer direction of the decoder.
- **Back-Translation (BT)** (Tsvetkov et al., 2018) focuses on learning a latent representation of the input by means of language translation model and further perform generation.
- **Cycled Reinforcement Learning (CRL)** is proposed by Xu et al. (2018). They use a cycled reinforcement learning model to explicitly extract content to further generate transferred sentences.

### 4.4 Experiment Settings

For both datasets, the vocabulary size is 30,000 and out-of-vocabulary words are replaced with $unk$. We train *word2vec* (Mikolov et al., 2013) on an English Wikipedia corpus dump to get 300-dimensional word embeddings. The batch size is 64 and the hyper-parameters of each module on two datasets are shown in Table 4. For each module, we use the Adam (Kingma and Ba, 2014) optimizer and the initial learning rate is 0.0003. Besides, we make use of the dropout method (Srivastava et al., 2014) to avoid over-fitting and clip the gradients (Pascanu et al., 2013) to the maximum norm of 10. During training, we train the model for a fixed number of epochs and monitor its performance on the validation set after 100 updates. Once the training is finished, we select the model with the best G-score on the validation set as our final model and evaluate its performance on the test set.

## 5 Results and Discussion

In this section, we report the experimental results. Besides, further analysis is also provided.

### 5.1 Experimental Results

The automatic evaluation results on two datasets are shown in Table 5. Results show that our approach achieves the best overall performance. For instance, our model achieves 11% and 38% relative improvement of G-score over the best baseline on the two datasets, respectively. This shows that our approach can further reconcile sentiment transformation and content preservation. We also note that our method achieves particularly high BLEU scores, indicating that the proposed method can enhance the specificity of the generated text, leading to better content preservation.

The human evaluation results are shown in Table 6, from which we can draw similar conclusions. It is obvious that our approach can outperform the baselines by a large margin, especially in terms of content preservation. The reason is that there is no *information gap* at both ends of our decoder when generating sentiment-transferred sen-

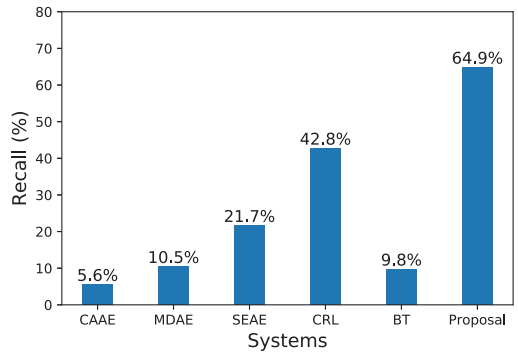


Figure 2: The recall scores of specific information (e.g. proper nouns) on different systems.

tences, which relieves the burden on the decoder. Therefore, more specific information is preserved.

### 5.2 Effectiveness of Enhancing Specificity

The analysis in Section 5.1 has shown that our approach can achieve excellent content preservation, which shows that the proposed model can generate text containing more specific information to some extent. To further verify this conclusion, we manually select test samples containing specific information (e.g. proper nouns) and then calculate the recall scores of this specific information on different systems. Figure 2 shows the related results, from which we can see that our approach achieves the highest recall score. This further illustrates that our approach can effectively improve the specificity of the generated text.

### 5.3 Ablation Study

We conduct an ablation study to verify the effectiveness of the emotionalization module. The results are shown in Table 7, illustrating that the removal of the emotionalization module leads to an obvious reduction in model performance. Compared to the ablated version, the full model achieves improvements of 11.14% and 9.87% G-score on two datasets, respectively. Besides, both sentiment transformation and content preservation can be improved with the emotionalization module. The emotionalization module can automatically generate corresponding sentiment words based on the extracted content, which eliminates *information gap* and relieves the burden of the decoder, leading to better model performance.

### 5.4 Case Study

Table 8 shows several examples of the testing process for our approach, which intuitively reflects

| **Yelp** | **ACC** | **BLEU** | **G-score** |
|---|---|---|---|
| Full model | 68.65 | 47.42 | 57.06 |
| w/o EM | 52.40 | 40.24 | 45.92 |
| **Amazon** | **ACC** | **BLEU** | **G-score** |
| Full model | 56.30 | 49.83 | 52.97 |
| w/o EM | 44.90 | 41.37 | 43.10 |

Table 7: Ablation study. "EM" denote the emotionalization module.

| |
|---|
| **Input:** *The worst and unprofessional company.*<br>**Separation:** *The worst and unprofessional company.*<br>**Emotionalization:** *Exceeded my expectations*<br>**Fusion:** *The company exceeded my expectations.* |
| **Input:** *Staff is awesome and always friendly.*<br>**Separation:** *Staff is awesome and always friendly.*<br>**Emotionalization:** *So rude unprofessional*<br>**Fusion:** *So rude and unprofessional staff.* |

Table 8: Outputs of different modules of our method. The underlined words are sentiment words recognized by separation module.

the function of each module. Given an input, the separation module can separate sentiment words from content words. For instance, the separation module successfully identifies the sentiment words "*worst and unprofessional*" in the first example. Then, the emotionalization module is able to generate corresponding target sentiment expression "*exceeded my expectations*" based on the given content words "*the company*". Beyond that, the fusion module can fuse content words and target sentiment expression to further generate a fluent sentiment-transferred sentence "*the company exceeded my expectations*".

Table 9 presents several outputs of different systems on the Yelp dataset. The baselines tend to generate outputs lacking specific information, resulting in poor content preservation. For instance, all baselines miss proper noun "*wendys*" in the first example. In contrast, our approach increases the specificity of the output, leading to better content preservation. Our method not only performs sentiment addition and content reconstruction independently, but also eliminates the *information gap*. This eases the burden of the decoder, allowing more specific information to be preserved.

### 5.5 Error Analysis

Although our approach outperforms the baselines by a large margin, we still observe some failed examples. We conduct a detailed analysis and categorize them into two types demonstrated below.

| |
|---|
| **Input:** ***Wendys*** *is my favorite chain, but this location is the worst.*<br>**CAAE:** *Great food and the best food!*<br>**MDAE:** *Best best i ever visit.*<br>**SEAE:** *This place had amazing stars!*<br>**CRL:** *Undoubtedly is my favorite chain.*<br>**Proposal:** ***Wendys*** *is my favorite chain, great location!* |
| **Input:** ***Annette*** *in optical was super helpful!*<br>**CAAE:** *I have been to say and i have n't be them.*<br>**MDAE:** *Terrible here is not disappointed , it!*<br>**SEAE:** *Walked in the service was super well.*<br>**CRL:** *Horrible stylist bank was extremely unprofessional.*<br>**Proposal:** ***Annette*** *in optical was rude!* |

Table 9: Examples generated by different systems on the Yelp dataset. "***Wendys***" and "***Annette***" belong to the specific information contained in the input.

One type of error is the neutral output without any sentiment polarity, such as "*I went to the bar on Friday*". This is because both datasets contain a large number of neutral reviews, which can be viewed as noises. Since these neutral reviews are also labeled as positive or negative, the separation module may incorrectly identify some content words as sentiment words. Training samples lacking sentiment information make the emotionalization module unable to generate sentiment words, leading to the neutral output.

The other type of error is the output that contains the original sentiment, which can be attributed to the failure of the separation of content and sentiment. Such failure usually occurs when the input contains sentiment words that are unseen in the training data, or the sentiment is implicitly expressed. The separation module may not recognize these unseen or obscure sentiment words, causing that the extracted content still contains original sentiment information. This content is further inputted to the emotionalization module, causing the output to contain original sentiment.

## 6 Related Work

This paper is mainly related to the following two series of work.

**Unsupervised sentiment modification.** One line of research for this task focuses on implicitly separating content from original sentiment. To control the sentiment, Hu et al. (2017) combines VAE and attribute discriminator while Shen et al. (2017) strives to align latent representations in hidden layers. Both Fu et al. (2018) and Yang et al. (2018) adopt adversarial training to ensure the success of separation. Tsvetkov et al. (2018) strives to obtain style-independent content representations through additional translation models. However, this implicit separation tends to result in poor content preservation since content and sentiment are entangled in an uninterpretable way. Different from this line of work, we adopt explicit separation of content and sentiment via self-attention, which achieves better content preservation. Another line strives to explicitly separate content and sentiment at the word-level. Li et al. (2018) proposes a retrieval system, which may lead to low-quality outputs. Xu et al. (2018) presents a reinforcement learning (RL) approach to remove sentiment words. However, the training of RL is unstable and sensitive to hyper-parameters. Further, Zhang et al. (2018a) tries to automatically extracts appropriate sentiment information from learned sentiment memories. Different from Zhang et al. (2018a) which extracts entangled sentiment memories from capacity-constrained memory matrices, our work can generate accurate explicit sentiment words flexibly based on specific content, leading to better performance. There also exist some other endeavors which construct pseudo-parallel data by means of back-translation (Sennrich et al., 2015). For instance, Zhang et al. (2018b) and Luo et al. (2019) jointly train two transfer systems via iterative back-translation. Further, Lample et al. (2019) extend this training framework to support multiple attribute control .

**Sentiment analysis.** Our work is also related to: sentiment analysis (Severyn and Moschitti, 2015; Cambria, 2016; Poria et al., 2017; Lam et al., 2018), sentiment embeddings (Tang et al., 2014; Barnes et al., 2018), and aspect extraction (Poria et al., 2016; He et al., 2017; Pablos et al., 2018).

## 7 Conclusion

In this paper, we propose a simple but effective specificity-driven cascading approach for unsupervised sentiment modification. The proposed approach performs target sentiment addition and content reconstruction independently, so that more specific information is preserved. Extensive experimental results show that our approach outperforms several competitive systems by a large margin. Further analysis demonstrates that the proposed method not only increases the specificity of the generated text, but also further improves content preservation.

## References


Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Jeremy Barnes, Roman Klinger, and Sabine Schulte im Walde. 2018. Bilingual sentiment embeddings: Joint projection of sentiment across languages. In *ACL*, pages 2483–2493.

Erik Cambria. 2016. Affective computing and sentiment analysis. *IEEE Intelligent Systems*, 31(2):102–107.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2018. Style transfer in text: Exploration and evaluation. In *AAAI*.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2017. An unsupervised neural attention model for aspect extraction. In *ACL*, pages 388–397.

Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *WWW*, pages 507–517.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. volume 9, pages 1735–1780. MIT Press.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Toward controlled generation of text. In *ICML*, pages 1587–1596.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Wai Lam, Xin Li, Bei Shi, and Lidong Bing. 2018. Transformation networks for target-oriented sentiment classification. In *ACL*, pages 946–956.

Guillaume Lample, Sandeep Subramanian, Eric Smith, Ludovic Denoyer, Marc'Aurelio Ranzato, and Y-Lan Boureau. 2019. Multiple-attribute text rewriting. In *ICLR*.

Juncen Li, Robin Jia, He He, and Percy Liang. 2018. Delete, retrieve, generate: a simple approach to sentiment and style transfer. In *NAACL-HLT*, pages 1865–1874.

Fuli Luo, Peng Li, Jie Zhou, Pengcheng Yang, Baobao Chang, Zhifang Sui, and Xu Sun. 2019. A dual reinforcement learning framework for unsupervised text style transfer. *arXiv preprint arXiv:1905.10060*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Aitor García Pablos, Montse Cuadros, and German Rigau. 2018. W2VLDA: almost unsupervised system for aspect based sentiment analysis. *Expert Syst. Appl.*, 91:127–137.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *ICML*, pages 1310–1318.

Soujanya Poria, Erik Cambria, and Alexander F. Gelbukh. 2016. Aspect extraction for opinion mining with a deep convolutional neural network. *Knowl.-Based Syst.*, 108:42–49.

Soujanya Poria, Haiyun Peng, Amir Hussain, Newton Howard, and Erik Cambria. 2017. Ensemble application of convolutional neural networks and multiple kernel learning for multimodal sentiment analysis. *Neurocomputing*, 261:217–230.

Cícero Nogueira dos Santos, Igor Melnyk, and Inkit Padhi. 2018. Fighting offensive language on social media with unsupervised text style transfer. In *ACL*, pages 189–194.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709*.

Aliaksei Severyn and Alessandro Moschitti. 2015. Twitter sentiment analysis with deep convolutional neural networks. In *SIGIR*, pages 959–962.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi S. Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *NIPS*, pages 6833–6844.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *ACL*, pages 1555–1565.

Yulia Tsvetkov, Alan W. Black, Ruslan Salakhutdinov, and Shrimai Prabhumoye. 2018. Style transfer through back-translation. In *ACL*, pages 866–876.

Jingjing Xu, Xu Sun, Qi Zeng, Xuancheng Ren, Xiaodong Zhang, Houfeng Wang, and Wenjie Li. 2018. Unpaired sentiment-to-sentiment translation: A cycled reinforcement learning approach. In *ACL*, pages 979–988.

Zichao Yang, Zhiting Hu, Chris Dyer, Eric P. Xing, and Taylor Berg-Kirkpatrick. 2018. Unsupervised text style transfer using language models as discriminators. In *NeurIPS*, pages 7298–7309.

Yi Zhang, Jingjing Xu, Pengcheng Yang, and Xu Sun. 2018a. Learning sentiment memories for sentiment modification without parallel data. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1103–1108.

Zhirui Zhang, Shuo Ren, Shujie Liu, Jianyong Wang, Peng Chen, Mu Li, Ming Zhou, and Enhong Chen. 2018b. Style transfer as unsupervised machine translation. *arXiv preprint arXiv:1808.07894*.

# LexicalAT: Lexical-Based Adversarial Reinforcement Training for Robust Sentiment Classification

**Jingjing Xu[1]*, Liang Zhao[2]*, Hanqi Yan[2], Qi Zeng[1], Yun Liang[3], Xu Sun[1,2]**
[1] MOE Key Lab of Computational Linguistics, School of EECS, Peking University
[2] Center for Data Science, Beijing Institute of Big Data Research, Peking University
[3] Center for Energy-efficient Computing and Applications, Peking University
{jingjingxu,zhao1iang,hanqi.yan,pkuzengqi}@pku.edu.cn
{ericlyun,xusun}@pku.edu.cn

## Abstract

Recent work has shown that current sentiment classification models are fragile and sensitive to simple perturbations. In this work, we propose a novel adversarial training approach, LexicalAT, to improve the robustness of current sentiment classification models. The proposed approach consists of a generator and a classifier. The generator learns to generate examples to attack the classifier while the classifier learns to defend these attacks. Considering the diversity of attacks, the generator uses a large-scale lexical knowledge base, WordNet, to generate attacking examples by replacing some words in training examples with their synonyms (e.g., sad and unhappy), neighbor words (e.g., fox and wolf), or super-superior words (e.g., chair and armchair). Due to the discrete generation step in the generator, we use policy gradient, a reinforcement learning approach, to train the two modules. Experiments show LexicalAT outperforms strong baselines and reduces test errors on various neural networks, including CNN, RNN, and BERT.[1]

## 1 Introduction

Sentiment classification is a fundamental research area in natural language processing (Pang et al., 2002; Glorot et al., 2011; Lai et al., 2015; Kiritchenko and Mohammad, 2018; Liu et al., 2018; Chen et al., 2018). With the development of deep learning, neural networks have obtained state-of-the-art results on many sentiment classification datasets (Kim, 2014; Dong et al., 2014; Tang et al., 2015). However, despite the promising results, recent work has shown that these models easily fail in adversarial examples[2] with little perturbations on real examples. This phenomenon shows that current sentiment classification models have poorly learned the true underlying patterns that determine the correct label. The over-fitting problem still needs to be further explored.

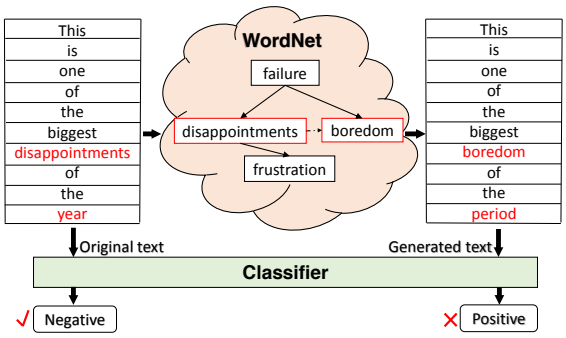


Figure 1: An attacking example for sentiment classification generated by the proposed approach. A pre-trained classifier correctly predicts the label of the original text but fails on the generated text.

Several approaches have been proposed in recent years for the attack problem. These studies can be roughly classified into two categories, data augmentation based approaches and adversarial training based approaches. The key idea of the former approaches is to assist the training of the classifier by augmenting the training data with pre-designed examples (Wang and Yang, 2015; Jia and Liang, 2017; Iyyer et al., 2018). Adversarial training based approaches (Miyato et al., 2017) aim to improve the generalization ability by adding random noises to word embeddings. Although these methods are good pioneering work, they either heavily rely on human knowledge or suffer from low diversity of attacks, which limits the robustness to diverse words and expressions.

In this work, we propose a lexical-based adversarial reinforcement training framework, LexicalAT, for robust sentiment classification. Compar-

*Equal Contribution.

[1]The code will be released at https://github.com/lancopku/LexicalAT

[2]Adversarial examples are intentionally designed by attackers to cause the model to make a mistake.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5518–5527,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

ing to previous studies, our approach can generate diversely attacking examples with self-learned policies. Our framework contains a generator and a classifier. The generator provides examples to attack the classifier while the classifier learns to defend these attacks. To generate diverse examples, we propose to use a large lexical knowledge base, WordNet, to add perturbations on training examples by replacing some words with their synonyms (e.g., sad and unhappy), neighbor words (e.g., fox and wolf), or super-superior words (e.g., chair and armchair)[3], as shown in Figure 1. Specifically, the output of the generator is a sequence of actions deciding which words should be replaced and their replacements. By involving the attacking examples into the training, the classifier will be more robust and powerful.

Since the generator has discrete generation steps, the gradient-based approach cannot be directly used to back-propagate the errors. We consider policy gradient, a reinforcement learning approach, to train the generator. With the feedback from the classifier as the reward, the generator is encouraged to generate tougher examples for the classifier. In return, with the increasing number of hard examples for training, the classifier becomes more robust and powerful. We evaluate the proposed approach on four popular sentiment classification datasets. Experiments show that LexicalAT outperforms strong baselines on various models and various datasets.

- We propose a lexical-based adversarial reinforcement training approach, LexicalAT, to improve the robustness of sentiment classification models.
- To the best of our knowledge, it is the first work of combining a knowledge base and adversarial learning. The knowledge base contributes to diverse example generation and the adversarial learning develops the attacking policy.
- Experiments show that LexicalAT outperforms strong baselines and improves results on various models, including CNN, RNN, and BERT.

[3]These relations are optional. If a relation has a high risk of changing labels in specific datasets, it is feasible to drop it.

## 2 Related Work

In this work, we focus on single-label sentiment classification where the input is a word sequence and the output is a single label. In many sentiment classification datasets, neural networks have achieved promising results, even comparable or super to humans. However, several studies have noted that these models are vulnerable and the performance is very sensitive to simple perturbations (Szegedy et al., 2014; Huang et al., 2017; Yuan et al., 2017).

Based on these findings, some studies have been proposed to improve the robustness of neural networks. These studies can be roughly classified into two categories, data augmentation based approaches and adversarial training based approaches. The main idea of the data augmentation based approaches is to augment the training data with pre-designed adversarial examples. Iyyer et al. (2018) propose a paraphrase method to generate syntactically adversarial examples for machine translation tasks. To explore semantically adversarial examples, Kobayashi (2018) replaces input words in real examples with the word predicted by a label-conditional language model. In these approaches, the attacking policy is task-specific and elaborately designed by humans. Unlike these approaches, Miyato et al. (2017) propose to use an adversarial training framework to attack the classifier by adding perturbations to the word embedding layer. However, the unchanged input text makes it hard to improve the robustness to diverse words and expressions.

In this work, we propose a new adversarial reinforcement training framework that aims to generate diverse attacks with self-learned policies. We build a generator that acts as a policy learner to automatically learn to attack the classifier. To generate diverse examples, we include WordNet, a large-scale lexical knowledge base, into the generator for word replacement.

## 3 Approach

### 3.1 Overview

Figure 2 shows the overall structure of LexicalAT. Given a sentence, the generator first generates a sequence of actions to replace some words with substitutes in WordNet and build a new example. Then, the new example is sent to the classifier to get the action reward. If the generated example

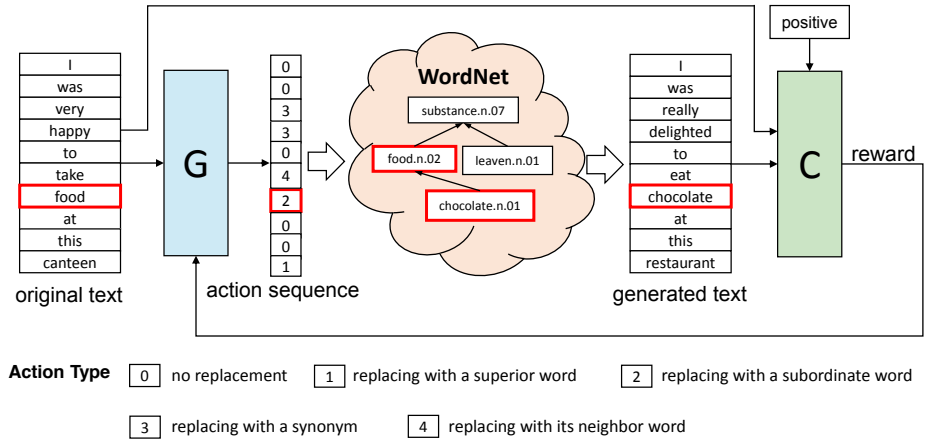


Figure 2: An illustration about the LexicalAT framework. The generator (G) generates an action sequence to replace some words in a training example with new words in WordNet. The classifier (C) calculates the reward of the generated example and returns it to the generator. **G** maximizes the expectation of the reward of generated examples by policy gradient and C minimizes the cross entropy loss between real and generated examples.

successfully confuses the classifier and decreases the probability of the original label, we regard it as a good example and give it a high reward. In this way, the generator is encouraged to generate tough examples for the classifier and the challenging generated examples are used in the training process of the classifier. By alternatively training the generator and the classifier, the classifier is trained to be more robust and powerful.

### 3.2 Generator

The generator generates attacking examples by adding noises on real examples with WordNet. WordNet is a large English lexical database. Nouns, verbs, adjectives and adverbs are grouped into cognitive synonyms sets (synsets). Each synset represents a distinct concept. As shown in Table 1, WordNet has two basic relations:

- Synonymy. It is the most basic relation, because WordNet uses sets of synonyms (synsets) to represent word senses.
- Hyponymy and hypernymy (super-subordinate). They are transitive relations between synsets. Because there is usually only one hypernym, the semantic relation hierarchically organizes the meanings of nouns.

The generation process is defined as a sequence labeling task for simplification. It takes a text sequence as input and an action sequence as output. Specifically, we define five actions representing the replacement decision, whether a word should be replaced or its replacement type, as shown in Table 2. For example, if a word is labeled with action "2", we choose a word from its subordinate words with the highest frequency as the replacement. Formally, assume that the input is a sequence of words $\boldsymbol{x} = \{x_1, x_2, \cdots, x_n\}$ where $n$ is the length of input text. This module generates a sequence of replacement actions $\boldsymbol{a} = \{a_1, a_2, \cdots, a_n\}$. Then, by querying WordNet based on $\boldsymbol{x}$ and $\boldsymbol{a}$, we can get a new sentence. Since the proposed framework is independent of the structure of the generator, for simplification we use a traditional sequence labeling model, Bi-Directional Long-Short Term Memory Network

| Relation | Syntactic Category | Examples |
|---|---|---|
| **Synonymy** (similar) | N, V, Aj, Av | (pipe, tube) (rise, ascend) (sad, unhappy) |
| **Hyponymy and Hypernymy** (super-subordinate) | N | (sugar maple, maple) (maple, tree) (tree, plant) |
| Note: N = Nouns, Aj = Adjectives, V = Verbs, Av = Adverbs | | |

Table 1: Basic relations in WordNet.

| Action | Description | Example |
|---|---|---|
| 0 | No replacement | N/A |
| 1 | Replacing with a super word | hamburger vs. sandwich |
| 2 | Replacing with a subordinate word | fish vs. salmon |
| 3 | Replacing with a synonymy | disappointed vs. frustrated |
| 4 | Replacing with a neighbor word | elephant vs.donkey |

Table 2: Actions defined in this work. The neighbor word is the word that shares the same super word with the word to be replaced.

(Bi-LSTM) (Hochreiter and Schmidhuber, 1997), as implementation.

### 3.3 Classifier

In this work, we focus on single-label sentiment classification. The input is a sequence of words and the output is a label from a pre-defined set $Y = \{y_1, y_2, \cdots, y_k\}$, where $k$ is the number of labels. To evaluate the effectiveness of the proposed approach on various settings, we implement two widely-used sentiment classification networks, Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN) (Kim, 2014), and one state-of-the-art pre-trained model, BERT (Devlin et al., 2018).

In the RNN-based classifier, the input word embeddings are fed into a single Long-Short Term Memory Network (LSTM). Then, a feed-forward layer transfers the last hidden vector of LSTM into the probability distribution of labels.

In the CNN-based classifier, we first feed the word embeddings into a convolutional layer with four convolutional filters. Then, the concatenated filter output applies a max-pooling and is fed into a two-layer feed-forward network with ReLU, followed by the softmax function.

In the BERT-based classifier, we use a model pre-trained on a large-scale language modeling dataset for initialization. Following the work (Devlin et al., 2018), we take the final hidden state for the first token in the input as sentence representation for classification. We use the released model and code, BERT-large, as implementation, which is based on 24 transformer layers.[4]

### 3.4 Adversarial Reinforcement Training

Due to the discrete choice in generation steps, we use policy gradient, a reinforcement learning approach, to adversarially train the two modules. The generator can be viewed as the agent which interacts with the classifier that acts as environment. The generator improves itself by maximizing the reward returned from the environment.

Given a real example $(\boldsymbol{x}, y)$, the generator first samples an action sequence $\hat{\boldsymbol{a}}$ based on the following probability distribution:

$$\hat{\boldsymbol{a}} \sim p_{\mathbf{G}}(\boldsymbol{a}|\boldsymbol{x}, \boldsymbol{\theta}), \tag{1}$$

where $\boldsymbol{\theta}$ represents the parameter of the generator.

[4]The pre-trained model is provided by goo.gl/language/bert

**Algorithm 1** Adversarial reinforcement training.

**Require:** Generator $\mathbf{G}$; classifier $\mathbf{C}$; training corpus $\mathcal{D}$
Initialize $\mathbf{G}$ and $\mathbf{C}$ with random weights
Pre-train $\mathbf{C}$ using Eq. (6) on $\mathcal{D}$
**repeat**
  **for** warm-steps **do**
    Sample an action sequence $\hat{\boldsymbol{a}}$ using Eq. (1)
    Build a new example $(\boldsymbol{x'}, y)$ using $\hat{\boldsymbol{a}}$
    Compute reward $r(\hat{\boldsymbol{a}})$ by Eq. (3)
    Update parameters of $\mathbf{G}$ by Eq. (4)
  **end for**
  **for** rl-steps **do**
    Sample an action sequence $\hat{\boldsymbol{a}}$ using Eq. (1)
    Build a new example $(\boldsymbol{x'}, y)$ using $\hat{\boldsymbol{a}}$
    Compute reward $r(\hat{\boldsymbol{a}})$ by Eq. (3)
    Update parameters of $\mathbf{G}$ by Eq. (4)
    Update parameters of $\mathbf{C}$ by Eq. (5)
    Update parameters of $\mathbf{C}$ by Eq. (6)
  **end for**
**until** convergence

Based on the sampled action sequence $\hat{\boldsymbol{a}}$, we get a new example $(\boldsymbol{x'}, y)$ after word replacement.

Then, we feed the real example $(\boldsymbol{x}, y)$ and the generated example $(\boldsymbol{x'}, y)$ into the classifier to get the replacement reward $r(\hat{\boldsymbol{a}})$. Specifically, we define the reward as the absolute difference of the probability of $y$ between the real example and the generated example:

$$r(\hat{\boldsymbol{a}}) = \log p_{\mathbf{C}}(y|\boldsymbol{x}; \boldsymbol{\phi}) - \log p_{\mathbf{C}}(y|\hat{\boldsymbol{x}}; \boldsymbol{\phi}), \tag{2}$$

where $\boldsymbol{\phi}$ is the parameter of the classifier. If the generated example successfully confuses the classifier and decreases the probability of $y$, we regard it as a good example and give it a high reward. In practice, we use the following equation to get a new reward to train the generator steadily:

$$r'(\hat{\boldsymbol{a}}) = r(\hat{\boldsymbol{a}}) - \mathbb{E}_{P_G(\boldsymbol{a}|\boldsymbol{x}, \boldsymbol{\theta})}(r(\boldsymbol{a})), \tag{3}$$

where $\mathbb{E}_{P_G(\boldsymbol{a}|\boldsymbol{x}, \boldsymbol{\theta})}(r(\boldsymbol{a}))$ is the expectation of $r(\boldsymbol{a})$.

Then, the reward is fed back to the generator. Formally, the expected gradient of the parameter $\boldsymbol{\theta}$ can be approximated as:

$$\begin{aligned} \nabla_{\boldsymbol{\theta}\mathcal{L}} \approx & - r'(\hat{\boldsymbol{a}}) \nabla_{\boldsymbol{\theta}} \log p_{\mathbf{G}}(\hat{\boldsymbol{a}}|\boldsymbol{x}; \boldsymbol{\theta}) \\ & - \mathbb{E}_{p_{\mathbf{G}}(\hat{\boldsymbol{a}}|\boldsymbol{x}; \boldsymbol{\theta})} r'(\hat{\boldsymbol{a}}), \end{aligned} \tag{4}$$

where $\hat{\boldsymbol{a}}$ is the generated action sequence. The second term $\mathbb{E}_{p_{\mathbf{G}}(\hat{\boldsymbol{a}}|\boldsymbol{x}; \boldsymbol{\theta})} r'(\hat{\boldsymbol{a}})$ means we approximate the expectation with sampling in practice.

The generated example $(\boldsymbol{x'}, y)$ is then used to train the classifier by minimizing the following cross-entropy loss:

$$\mathcal{L} = - \log p_{\mathbf{C}}(y|\boldsymbol{x'}, \boldsymbol{\phi}). \tag{5}$$

Furthermore, to prevent the classifier from forgetting the knowledge of training data, we also use teacher forcing (Goyal et al., 2016) to train the classifier. After each reinforcement training step, we run a teacher forcing step by directly using the real example $(\boldsymbol{x}, y)$ to train the classifier:

$$\mathcal{L} = -\log p_{\mathbf{C}}(y|\boldsymbol{x}, \boldsymbol{\phi}). \quad (6)$$

Considering that reinforcement learning process requires the two modules with the initial learning ability, we pre-train the generator and the classifier before the reinforcement learning stage. The classifier is directly pre-trained on real training data until convergence. Due to the lack of supervisory signals, we build a warm-up stage to pre-train the generator. In the warm-up stage, the feedback of the pre-trained classifier is used to train the generator.

## 4 Experiments

We evaluate LexicalAT on four sentiment classification datasets. We first introduce datasets, implementation details, and baselines. Then, we show experiment results and provide detailed analysis.

### 4.1 Datasets

**SST-2 & SST-5.** The Stanford sentiment treebank (Socher et al., 2013) is a single-sentence classification dataset built on movie reviews. Based on the difference of sentiment granularity, the human annotators design two label sets. Following existing work (Kobayashi, 2018), we run experiments on both label sets. For clarify, we call them SST-2 and SST-5.

**RT.** The rating inference dataset (Pang and Lee, 2005) is another sentiment classification dataset. The data is from internet movie reviews and has two types of labels.

**Yelp.** The dataset is built upon reviews from website Yelp.[5] Each review has a rating label varying from 1 to 5. We randomly select 100K for training, 10K for validation, and 10K for testing. It is used to verify whether the proposed approach is applied to tasks with large-scale data.

The dataset statistics are shown in Table 3. For SST-2 and SST-5, we use the standard split in their work. For RT, due to the lack of the standard split, we randomly divide all examples into 90% for the training set and 10% for the test set. To build the development set, we randomly take out 10% from the training set.

[5]https://www.yelp.com/dataset/challenge

| Dataset | #Class | Avg. #w | Train | Dev | Test |
|---|---|---|---|---|---|
| SST2 | 2 | 19 | 6,920 | 872 | 1,821 |
| SST5 | 5 | 18 | 8,544 | 1,101 | 2,210 |
| RT | 2 | 21 | 8,608 | 964 | 1,089 |
| Yelp | 5 | 89 | 100,000 | 10,000 | 10,000 |

Table 3: Dataset statistics. "Class" is the number of pre-defined labels. "Avg. #w" is the average word number in the input text. "Train", "Dev", and "Test" represent the sizes of the training set, the development set, and the test set.

### 4.2 Baselines

We compare our proposed approach with the following robustness-driven approaches.

**SynDA.** It is a synonymy based data augmentation approach (Zhang et al., 2015). It uses an English thesaurus, obtained from WordNet, to randomly replace some words in real examples with their synonymys to build new examples.

**ConDA.** It is a contextual data augmentation approach (Kobayashi, 2018). They build adversarial examples by randomly replacing words in real examples with the words that are predicted by a bidirectional language model at the word positions.

**VAT.** It is an adversarial training based approach (Miyato et al., 2017) for robust text classification. It adds perturbations to recurrent neural networks to improve the robustness. We use the released code for implementation.

### 4.3 Experiment settings

Based on the performance on the development sets, we set batch size to 64 except yelp whose batch size is 256. We use the Adam optimizer to train the modules. The details of model-specific hyper-parameter settings are shown in Table 4. In the pre-training stage, we train the classifier until convergence. In the warm-up stage, we train the generator by 3 epoch. In the reinforcement training stage, we set the maximum epcoh to 100 and adopt the early stopping mechanism.

### 4.4 Results and Discussion

The results of the proposed approach and baselines are shown in Table 5.

| | Hyper-Parameter | Setting |
|---|---|---|
| **Generator** | Embedding size | 300 |
| | Embedding dropout | 0.4 |
| | Hidden size | 300 |
| | BiLSTM layer | 1 |
| | Learning rate | 1e-3 |
| **CNN Classifier** | Embedding size | 300 |
| | CNN layer | 1 |
| | CNN output dropout | 0.1 |
| | CNN dense layer dropout | 0.4 |
| | Filter kernel size | [2,3,4,5] |
| | Filter kernel dimension | 300 |
| | Learning rate | 1e-4 |
| **RNN Classifier** | Embedding size | 300 |
| | Embedding dropout | 0.4 |
| | Hidden size | 300 |
| | LSTM layer | 1 |
| | LSTM output dropout | 0.1 |
| | Learning rate | 1e-4 |
| **BERT Classifier** | Transformer layer | 24 |
| | Embedding size | 1,024 |
| | Hidden size | 1,024 |
| | Head | 16 |
| | Learning rate | 2e-5 |
| | Fine-turning epoch | 3 |

Table 4: Settings of model-specific hyper-parameters.

| Approach | SST-2 | SST-5 | RT | Yelp |
|---|---|---|---|---|
| RNN(our implemented) | 80.61 | 40.54 | 75.85 | 60.94 |
| RNN (Kobayashi, 2018) | 80.30 | 40.20 | * | * |
| +SynDA (Zhang et al., 2015) | 80.20 | 40.50 | * | * |
| +ConDA (Kobayashi, 2018) | 80.10 | 41.10 | * | * |
| +VAT (Miyato et al., 2017) | 81.16 | 37.38 | 75.94 | 59.69 |
| **+LexicalAT (proposed)** | **81.60** | **41.99** | **76.12** | **61.18** |
| **Approach** | **SST-2** | **SST-5** | **RT** | **Yelp** |
| CNN(our implemented) | 80.62 | 40.81 | 75.85 | 60.77 |
| CNN (Kobayashi, 2018) | 79.50 | 41.30 | * | * |
| +SynDA (Zhang et al., 2015) | 80.00 | 40.70 | * | * |
| +ConDA (Kobayashi, 2018) | 80.80 | **42.10** | * | * |
| +VAT (Miyato et al., 2017) | * | * | * | * |
| **+LexicalAT (Proposed)** | **81.58** | 41.67 | **76.22** | **61.86** |
| **Approach** | **SST-2** | **SST-5** | **RT** | **Yelp** |
| BERT(our implemented) | 92.60 | 55.07 | 88.57 | 66.76 |
| +SynDA (Zhang et al., 2015) | * | * | * | * |
| +ConDA (Kobayashi, 2018) | * | * | * | * |
| +VAT (Miyato et al., 2017) | * | * | * | * |
| **+LexicalAT (proposed)** | **93.03** | **55.38** | **88.68** | **67.50** |

Table 5: Comparisons between LexicalAT and baselines on four datasets.[6] LexicalAT outperforms strong baselines on all datasets, including three small datasets and a large dataset, Yelp.

As expected, LexicalAT, the proposed approach, substantially outperforms the naive baselines (RNN, CNN and BERT). Since RNN and CNN are directly trained on training sets, they tend to suffer from over-fitting. Therefore, it is reasonable that LexicalAT brings improvements to RNN and CNN models. BERT has better generalization ability because it has been pre-trained on a large scale corpus. Even so, LexicalAT still gains accuracy improvements over BERT by 0.43 on SST-2, 0.31 on SST-5, 0.11 on RT, and 0.74 on Yelp, respectively. These results show that the proposed approach is universal and well applied to various models, even the state-of-the-art pre-trained model BERT.

By contrast, SynDA, the synonymy based data augmentation approach, does not bring significant performance improvements over the naive baselines. The main difference between LexicalAT and SynDA is that SynDA uses the random replacement policy to generate new examples, while LexicalAT uses the dynamic replacement policy learned by the proposed adversarial reinforcement training framework. The performance gap between LexicalAT and SynDA shows that the proposed adversarial reinforcement training framework is effective for learning the attacking policy toward the weaknesses of the classifier. Furthermore, the proposed approach beats VAT under various datasets. Since VAT only adds random noises on word embeddings without changing the input text, it does not augment the training data with new words and expressions, and thus limits the robustness improvement. In summary, with adversarial reinforcement training, lexicalAT is capable of learning the attacking policy toward the weaknesses of the classifier. With WordNet, lexicalAT can generate examples with diverse expressions, which improves the classifier robustness. These advantages make the proposed approach perform well and achieve the best performance on various datasets.

To illustrate the training process, we plot the performance on the SST-5 development set in Figure 3. LexicalAT converges to a higher accuracy than the naive baseline RNN. It shows that the adversarial reinforcement training mechanism is robust and can converge to good results.

Furthermore, we compare the performance of lexicalAT and naive baselines on defending attacks, taking RNN and CNN on SST-2, SST-5, and RT as examples. The results are shown in Table 6. "RNN-Attacking Set" and "CNN-Attacking Set" are two sets generated by two different gen-

[6]The results of SynDA and ConDA are from the work of Kobayashi (2018).

| SST-2 | Classifier (RNN) | Classifier (CNN) | LexicalAT (RNN) | LexicalAT (CNN) |
|---|---|---|---|---|
| **Test Set** | 80.61 | 80.62 | 81.60 | 81.58 |
| **RNN-Attacking Set** | 69.91 | 65.62 | 76.44 | 73.70 |
| **CNN-Attacking Set** | 68.81 | 68.04 | 74.62 | 76.28 |
| **SST-5** | **Classifier (RNN)** | **Classifier (CNN)** | **LexicalAT (RNN)** | **LexicalAT (CNN)** |
| **Test Set** | 40.54 | 40.81 | 41.99 | 41.67 |
| **RNN-Attacking Set** | 35.16 | 35.43 | 38.73 | 38.91 |
| **CNN-Attacking Set** | 34.98 | 36.60 | 37.65 | 38.96 |
| **RT** | **Classifier (RNN)** | **Classifier (CNN)** | **LexicalAT (RNN)** | **LexicalAT (CNN)** |
| **Test Set** | 75.85 | 75.85 | 76.12 | 76.22 |
| **RNN-Attacking Set** | 69.05 | 68.78 | 71.44 | 70.61 |
| **CNN-Attacking Set** | 62.90 | 61.89 | 69.88 | 71.17 |

Table 6: Comparisons of different classifiers on defending attacks. The classifiers trained in lexicalAT, lexicalAT (RNN) and lexicalAT (CNN), have better defending ability. "Classifier (RNN)" and "Classifier (CNN)" are two different naive baselines trained on real examples. "Lexical (RNN)" and "Lexical (CNN)" are two classifiers trained in the framework of lexicalAT. "Test Set" is the standard test set. "RNN-Attacking Set" and "CNN-Attacking Set" are two attacking sets generated by two generators in the framework of lexicalAT based on real test examples. The two generators are trained based on the interaction with RNN-based and CNN-based classifiers, respectively.

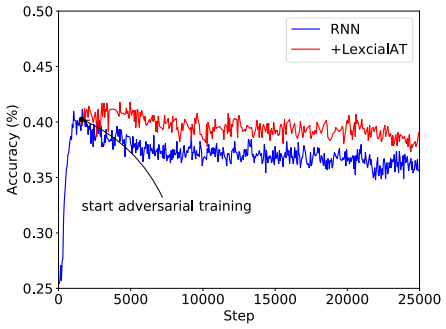


Figure 3: Learning curves of RNN and LexicalAT on the development set of SST-5.

| Setting | SST-2 | SST-5 | RT | Average |
|---|---|---|---|---|
| all words (RNN) | 81.60 | 41.14 | 75.76 | 66.17 |
| -super words | 80.72 | 41.17 | 75.02 | 65.64 |
| -subordinate words | 80.56 | 41.67 | 75.48 | 65.90 |
| -synonymy words | **80.45** | 41.99 | 76.12 | 66.19 |
| -neighbor words | 80.56 | **40.91** | **74.66** | **65.38** |
| **Setting** | **SST-2** | **SST-5** | **RT** | **Average** |
| all words (CNN) | 80.18 | 41.86 | 74.84 | 65.63 |
| -super words | 79.96 | 41.67 | 76.22 | 65.95 |
| -subordinate words | 81.58 | 41.49 | 75.39 | 66.15 |
| -synonymy words | **79.24** | 41.67 | 74.74 | **65.22** |
| -neighbor words | 81.33 | **40.68** | **74.47** | 65.49 |

Table 7: Effects of different replacement actions. As we can see, neighbor and synonymy words contribute most to the performance.

erators in lexicalAT based on real test examples. The two generators are trained based on the interaction with RNN-based and CNN-based classifiers, respectively. The naive baselines, RNN and CNN, have weak robustness performances and the results drop largely on attacking examples, even a 15% accuracy decrease for "Classifier (CNN)" on SST-2. In comparison, the classifiers trained in lexicalAT, lexicalAT (RNN) and lexicalAT (CNN), have better defending ability.

### 4.5 Analysis

**What is the effect of each action?** To show the effect of different actions, we take RNN and CNN on SST-2, SST-5, and RT as examples and conduct experiments by dropping one action at a time. As Table 7 shows, only some actions are useful for the robustness improvement. The average performance becomes better after dropping subordinate words under two different settings. Surprisingly, the neighbor words largely contribute to the performance. Without neighbor words, the average accuracies are dropped from 66.17 to 65.38 for RNN, and from 65.63 to 65.22 for CNN. Neighbor words share the same super word, like "fox" and "wolf". In WordNet, neighbor words usually have similar semantic meanings. By replacing a word with its similar word, the semantic diversity can be largely enhanced. Furthermore, the semantic similarity can reduce the risk of changing the label when modifying the input text. Therefore, it is a good choice to include this relation in models unless the replacement exactly impacts the original label in specific tasks.

**Is the learned attacking policy universal or model-specific?** To explore this question, we

| | Classifier (RNN) | Classifier (CNN) |
|---|---|---|
| **None** | 80.81 | 80.62 |
| **Policy-RNN** | **81.44** | 80.89 |
| **Policy-CNN** | 81.22 | **81.27** |

Table 8: Results of different classifiers trained with the examples generated by different attacking policies on SST-2 dataset. "Policy-RNN" and "Policy-CNN" are the learned policies from the interaction with RNN-based and CNN-based classifiers in the proposed framework.

conduct the following experiments.

We train two generators based on the feedback of the classifiers with different structures (e.g., RNN and CNN) on SST-2. For clarify, we call the learned attacking policies in the two generators as policy-RNN and policy-CNN, as shown in Table 8. Then, we put the examples generated by policy-RNN and policy-CNN into the training data to train the classifiers with different structures.

For the RNN-based classifier, two attacking policies both outperform the naive RNN baseline, which shows the universality of the learned attacking policy. Furthermore, policy-RNN brings more improvements than policy-CNN does on the RNN-based classifier. It demonstrates that the learned attacking policy has model-specific features. Similar result happens to the CNN-based classifier.

In summary, different classification models share some weaknesses and have their own unique vulnerability. Considering unique weaknesses are difficult to be summarized by human knowledge, our approach is an effective way to automatically learn the attacking policy toward the classifier weaknesses.

### 4.6 Case Study

Table 9 presents the adversarial examples generated by the generator on SST-5. Even the simple perturbations can confuse the classifier, which shows the weak robustness of the classifier.

### 4.7 Error Analysis

Although the proposed approach improves the robustness of current classifiers, it is important to note that there still have several problems to be further explored.

First, some generated examples contain low-quality phrases, such as the bottom one in Table 9. This problem is due to the inappropriate word replacement. In the future work, we would like to explore how to generate diverse and high-quality examples by taking sentence context, replaced word features, and replacing word features into consideration.

| | |
|---|---|
| **Original Text** | The **movie** is without **intent**. |
| **Generated Text** | The **film** is without **spirit**. |
| **Original Text** | The **script** is smart not **cloying**. |
| **Generated Text** | The **dialogue** is smart not **saccharine**. |
| **Original Text** | This is a gorgeous **film** vivid with **color music** and **life**. |
| **Generated Text** | This is a gorgeous **movie** vivid with **gloss sound** and **spirit**. |
| **Original Text** | Hollywood ending is the most **disappointing** woody allen **movie** ever. |
| **Generated Text** | Hollywood ending is the most **fail** woody allen **film** ever. |

Table 9: Adversarial examples generated by our approach.

Second, we regard the examples that confuse the classifier as good attacking examples and give them high reward to train the generator. However, not all confusing examples are useful for robustness improvements. To optimize the training time, in the future we will explore an advanced LexicalAT to automatically distinguish between "useful examples" and "useless examples".

## 5 Conclusion and Future Work

In this work, we propose a new adversarial training approach, LexicalAT, to improve the robustness of current sentiment classification models. The key idea is to use WordNet and adversarial reinforcement training to automatically learn the diversely attacking policy. We evaluate LexicalAT on four representative sentiment classification datasets. Experiments demonstrate that the proposed approach has better generality and reduces test errors on various neural networks, including CNN, RNN, and BERT.

In the future work, we would like to build an advanced version of LexicalAT from the following perspectives. First, to address the problem of low-quality phrases in generated text, we will explore how to keep the syntactic correctness of the generated text. Second, we would like to figure out the detailed effects of the generated examples on the robustness of sentiment classification models. By removing useless examples, we can obtain higher training speed.

## Acknowledgments

We thank all reviewers for providing the thoughtful and constructive suggestions. This work was supported in part by National Natural Science Foundation of China (No. 61673028). Xu Sun is the corresponding author of this paper.

## References

Xilun Chen, Yu Sun, Ben Athiwaratkun, Claire Cardie, and Kilian Q. Weinberger. 2018. Adversarial deep averaging networks for cross-lingual sentiment classification. *TACL*, 6:557–570.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 2: Short Papers*, pages 49–54.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 513–520.

Anirudh Goyal, Alex Lamb, Ying Zhang, Saizheng Zhang, Aaron C. Courville, and Yoshua Bengio. 2016. Professor forcing: A new algorithm for training recurrent networks. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4601–4609.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Sandy Huang, Nicolas Papernot, Ian J. Goodfellow, Yan Duan, and Pieter Abbeel. 2017. Adversarial attacks on neural network policies. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1875–1885.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2021–2031.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

Svetlana Kiritchenko and Saif Mohammad. 2018. Examining gender and race bias in two hundred sentiment analysis systems. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics, *SEM@NAACL-HLT, New Orleans, Louisiana, USA, June 5-6, 2018*, pages 43–53.

Sosuke Kobayashi. 2018. Contextual augmentation: Data augmentation by words with paradigmatic relations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pages 2267–2273.

Qi Liu, Yue Zhang, and Jiangming Liu. 2018. Learning domain representation for multi-domain sentiment classification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 541–550.

Takeru Miyato, Andrew M. Dai, and Ian J. Goodfellow. 2017. Adversarial training methods for semi-supervised text classification. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 115–124.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing, EMNLP 2002, Philadelphia, PA, USA, July 6-7, 2002*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1422–1432.

William Yang Wang and Diyi Yang. 2015. That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using #petpeeve tweets. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 2557–2563.

Xiaoyong Yuan, Pan He, Qile Zhu, Rajendra Rana Bhat, and Xiaolin Li. 2017. Adversarial examples: Attacks and defenses for deep learning. *CoRR*, abs/1712.07107.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 649–657.

# Leveraging Structural and Semantic Correspondence for Attribute-Oriented Aspect Sentiment Discovery

**Zhe Zhang**
Watson Group
IBM Corporation
Research Triangle Park, NC 27703-9141
zhangzhe@us.ibm.com

**Munindar P. Singh**
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206
singh@ncsu.edu

## Abstract

Opinionated text often involves attributes such as authorship and location that influence the sentiments expressed for different aspects. We posit that structural and semantic correspondence is both prevalent in opinionated text, especially when associated with attributes, and crucial in accurately revealing its latent aspect and sentiment structure. However, it is not recognized by existing approaches.

We propose Trait, an unsupervised probabilistic model that discovers aspects and sentiments from text and associates them with different attributes. To this end, Trait infers and leverages structural and semantic correspondence using a Markov Random Field. We show empirically that by incorporating attributes explicitly Trait significantly outperforms state-of-the-art baselines both by generating attribute profiles that accord with our intuitions, as shown via visualization, and yielding topics of greater semantic cohesion.

## 1 Introduction

Opinionated text is often associated with different *attributes*—latent variables that serve as reference frames relative to which the underlying aspects and sentiments are expressed. Common attributes in consumer reviews include author type (e.g., business traveler or tourist for hotel reviews; location for reviews of music (McDermott et al., 2016); culture on reviews of food (Bahauddin and Shaarani, 2015)). Whereas current approaches consider attributes in a one-off manner in each application, we posit that attributes can be systematically extracted if we can properly capture the structural and semantic correspondence that is prevalent in opinionated text. We claim that ignoring attributes may lead to biased inference on aspects and sentiments. As evidence, we demonstrate an approach that outperforms the state of the art and yields intuitive and cohesive topics.

We propose Trait, a general model for discovering attribute-oriented aspects and sentiments from text. By incorporating attributes, Trait automatically generates profiles that describe attributes in terms of sentiments and aspects. To leverage structural and semantic correspondence, Trait applies a Markov Random Field as regularization over sentences during inference. We evaluate Trait on four datasets from two domains and consider three attributes. Trait successfully discovers aspects associated with sentiments; the generated word clusters are more cohesive than the state-of-the-art baselines; the generated attribute profiles are well correlated with ground truth.

**Motivating Example.** Figure 1 presents two hotel reviews from TripAdvisor. We manually assign aspect labels for sentences and calculate pairwise cosine similarity between sentences using sentence embedding from a pretrained sentence encoding model (Cer et al., 2018).

Review A and Review B, which mention aspects *Room*, *Location*, and *Type*, exhibit structural and semantic correspondence. We posit that the correspondence of *Location* and *Type* is a result of the attribute value, Las Vegas, common to the two reviews. *Location* is a crucial aspect for hotels in Las Vegas. On a randomly selected set of 5,000 hotel reviews for Las Vegas, we observe that 4,281 sentences from 2,624 reviews mention the location "Strip." Using a similarity threshold of 0.6, we obtain 1,929 sentences similar to sentence A3 from 1,519 reviews, including sentence B3 in Review B. We obtain 133 sentences similar to sentence B4 from 128 reviews including A4 in Review A. Figure 1 shows some of these sentences. Likewise, using authorship as an attribute, we observe that users stick to their writing styles. For example, in hotel reviews, some users describe the condition of a room and others share travel tips.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5528–5538, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

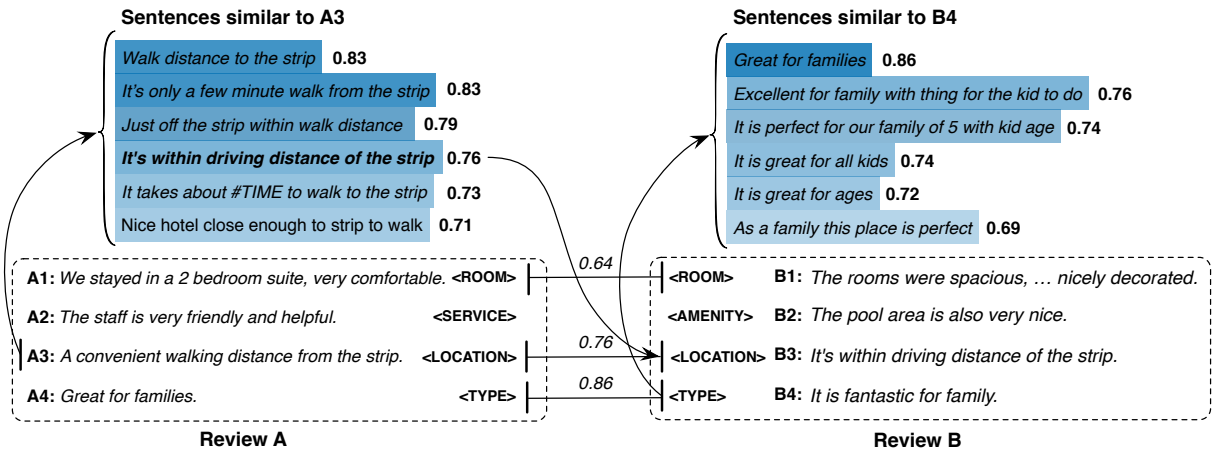


Figure 1: A motivating example.

**Contributions and Novelty.** Our contributions include: (1) a general model that generates attribute profiles associating aspects and sentiments with attributes in text; (2) empirical results demonstrating the benefit of incorporating attributes on a model's quality; and (3) empirical results demonstrating generalizability using diverse attributes and the quality of the generated attribute profiles.

Trait's novelty lies in its ability to accommodate attributes. First, it is general across attributes as opposed to being limited to predefined attributes. Second, the handling of attributes means that Trait avoids overfitting to the more prevalent attributes in a dataset. That is, Trait can learn a more refined conditional probability distribution that incorporates specific attributes than otherwise possible. Ignoring the observable attribute variables would relax the constraints on the distribution, meaning that the learned approximate distribution would be biased toward the majority attribute.

**Summary of Findings.** We demonstrate that incorporating attributes into generative models provides a superior, more refined representation of opinionated text. The resulting model generates topics with high semantic cohesion. We show that Markov Random Field can be used for effectively capturing structural and semantic correspondence.

## 2 Related Work

Generative probabilistic modeling has been widely applied for unsupervised text analysis. Given the observed variables, e.g., tokens in documents, a generative probabilistic model defines a set of dependencies between hidden and observed variables that encodes statistical assumptions underlying the data. Latent Dirichlet Allocation (LDA) (Blei et al., 2003), a well-known topic model, represents a document as a mixture of topics, each topic being a multinomial distribution over words. The learning process approximates the topic and word distributions based on their co-occurrence in documents.

Many efforts guide the topics learned by incorporating additional information. Rosen-Zvi et al.'s (2004) Author Topic model (AT) captures authorship by building a topic distribution for each author. When generating a word in a document, AT conditions the probability of topic assignment on the author of the document. Kim et al.'s (2012) model captures entities mentioned in documents and models the probability of generating a word as conditioned on both entity and topic. Diao and Jiang (2013) jointly model topics, events, and users on Twitter. Trait goes beyond these models by incorporating sentiments and attributes in a flexible way, which eliminates the model's dependency on specific attribute types.

Several probabilistic models tackle opinionated text. Titov and McDonald (2008b) handle global and local topics in documents. JST (Lin et al., 2012) and ASUM (Jo and Oh, 2011) model a review via multinomial distributions of topics and sentiments used to condition the probability of generating words. Kim et al. (2013) extend ASUM's probabilistic model to discover a hierarchical structure of aspect-based sentiments. Wang et al.'s (2016) topic model discovers aspect, sentiment, and both general and aspect-specific opinion words. Whereas these models identify aspects and sentiments, they disregard attribute information.

Titov and McDonald (2008a) discover top-

ics using aspect ratings provided by reviewers. Mukherjee et al.'s (2014) JAST considers authors during aspect and sentiment discovery. Poddar et al.'s (2017) AATS jointly considers author, aspect, sentiment, and the nonrepetitive generation of aspect sequences via a Bernoulli process. Zhang and Singh (2018)'s model jointly captures aspect, sentiment, author, and discourse relations. Trait is novel in that, unlike the above models, it is not tied to a specific attribute.

## 3 Model and Inference

We now introduce Trait's model and inference mechanism.

### 3.1 Sentence Embeddings

Measuring semantic similarity between sentences is integral to capturing the structural and semantic correspondence among reviews: high similarity indicates a high degree of correspondence. Cer et al. (2018) propose a pretrained sentence encoding model, Universal Sentence Encoder (USE). USE is based on Vaswani et al.'s (2017) attention-based neural network. Perone et al. (2018) show USE yields the best results among sentence embedding techniques on semantic relatedness and textual similarity tasks. Trait adopts USE to generate sentence embeddings and cosine similarity to measure semantic similarity between sentences.

### 3.2 Structural and Semantic Correspondence

A Markov Random Field (MRF) defines a joint probability distribution over a set of variables given the dependencies based on an undirected graph. The joint distribution is a factorized product of potential functions.

To capture structural and semantic correspondence, Trait defines an MRF over latent aspects of sentences. Given a set of reviews $D_a$ sharing a common attribute $a$, for sentence $l$ in $D_a$, Trait creates its corresponding sentence set $L$ by adding sentence $l_i$ in $D_a$ if the semantic similarity between sentence $l_i$ and $l$ is larger than a preset threshold $\rho$. For each pair of $l$ and $l_i$, Trait creates an undirected edge between the aspects associated with the two sentences ($t_l$, $t_{li}$). To promote $l$ and sentences in $L$ having a high probability of associating with the same aspect, Trait defines a binary edge potential, $\exp\{\mathcal{I}(t_l, t_{li})\}$, where $\mathcal{I}(\cdot)$ is an indicator function. This binary potential produces a large value if the two sentences have the same aspect and a small value otherwise. Given attribute $a$, sentiment $s$, and a document consisting of $N$ sentences, Trait computes the joint probability of aspect assignments of sentences as:

$$p(\boldsymbol{t}|\psi_{s,a}, \lambda) = \prod_{l}^{N} p(t_l|\psi_{s,a}) \exp\left\{\lambda \frac{\sum_{(t_l,t_{l_i})\in\mathcal{E}_l} \mathcal{I}(t_l = t_{l_i})}{|\mathcal{E}_l|}\right\}, \quad (1)$$

where $\psi_{s,a}$ is the aspect distribution given sentiment $s$ and attribute $a$; parameter $\lambda$ controls the reinforcing effects of correspondence regularization; and $\mathcal{E}_l$ is the set of undirected edges for $l$.

### 3.3 Generative Process

To capture the desired associations, given an attribute type, Trait generates a mixture over sentiments and aspects for each attribute value. Trait assumes that reviews are mixtures of sentiments and considers sentences the basic unit for a sentiment-aspect pair.

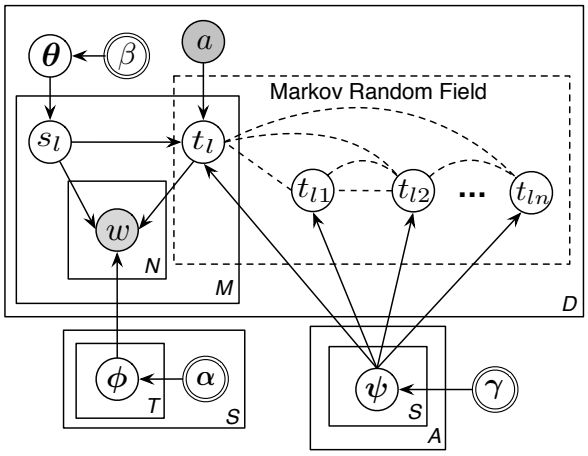


Figure 2: Generative process of Trait.

Figure 2 shows Trait's model. Hyperparameter $\alpha$ is the Dirichlet ($Dir(\cdot)$) prior of the word distribution $\phi$; $\beta$ is the Dirichlet prior of the sentiment distribution $\theta$; and $\gamma$ is the Dirichlet prior of the aspect distribution $\psi$. Given a set of reviews $D$ associated with a set of attribute values $A$ over a set of aspects $T$ and a set of sentiments $S$, each review contains $M$ sentences and each sentence contains $N$ words. Trait's generative process is as follows.

First, for each pair of aspect $t$ and sentiment $s$, draw a word distribution $\phi_{t,s} \sim Dir(\alpha)$. Second, for each attribute value $a$ and each sentiment

$s$, draw an aspect distribution $\psi_{s,a} \sim Dir(\gamma)$. Third, given a review $d$ with attribute $a$, draw a sentiment distribution $\theta_d \sim Dir(\beta)$, and for each sentence in $d$, (1) choose a sentiment $s \sim Multinomial(\theta_d)$; (2) given $s$, choose an aspect $t \sim Multinomial(\psi_{s,a})$; (3) given $t$ and $s$, sample word $w \sim Multinomial(\phi_{t,s})$.

Trait estimates $p(\boldsymbol{s}, \boldsymbol{t}|\boldsymbol{w}, a)$, the posterior distribution of latent variables, sentiments $\boldsymbol{s}$, and aspects $\boldsymbol{t}$, given all words used in reviews involving attribute $a$. We factor the joint probability of assignments of sentiments, aspects, and words for $a$:

$$p(\boldsymbol{s}, \boldsymbol{t}, \boldsymbol{w}|a) = p(\boldsymbol{w}|\boldsymbol{s}, \boldsymbol{t})p(\boldsymbol{t}|\boldsymbol{s}, a)p(\boldsymbol{s}). \quad (2)$$

### 3.4 Inference

We use collapsed Gibbs sampling (Liu, 1994) for posterior inference. By integrating over $\Phi = \{\boldsymbol{\phi_i}\}_{i=1}^{S\times T}$, we obtain Equation 2's first term (Section 4.1 explains $\alpha_v$).

$$p(\boldsymbol{w}|\boldsymbol{s},\boldsymbol{t},\boldsymbol{\alpha}) = \int p(\boldsymbol{w}|\boldsymbol{s},\boldsymbol{t},\Phi)p(\Phi|\boldsymbol{\alpha})\mathrm{d}\Phi$$
$$= \left(\frac{\Gamma(\sum_{v=1}^{W}\alpha_v)}{\prod_{v=1}^{W}\Gamma(\alpha_v)}\right)^{S\times T} \times \prod_{s=1}^{S}\prod_{t=1}^{T}\frac{\prod_{v=1}^{W}\Gamma(n_{s,t}^{v}+\alpha_v)}{\Gamma\left[\sum_{v=1}^{W}(n_{s,t}^{v}+\alpha_v)\right]}, \quad (3)$$

where $W$ is the vocabulary size; $n_{s,t}^{v}$ is the number of occurrences of word $v$ assigned to sentiment $s$ and aspect $t$; and $\Gamma(\cdot)$ is the Gamma function.

Next, by integrating over $\Psi_a = \{\boldsymbol{\psi_i}\}_{i=1}^{S}$, we calculate the second term in Equation 2 as (Section 4.1 explains $\gamma_t$):

$$p(\boldsymbol{t}|\boldsymbol{s},\boldsymbol{\gamma},a) = \int p(\boldsymbol{t}|\boldsymbol{s},\Psi_a,a)p(\Psi_a|\boldsymbol{\gamma})\mathrm{d}\Psi_a$$
$$= \left(\frac{\Gamma(\sum_{t=1}^{T}\gamma_t)}{\prod_{t=1}^{T}\Gamma(\gamma_t)}\right)^{S} \times \prod_{s=1}^{S}\frac{\prod_{t=1}^{T}\Gamma(n_{s,a}^{t}+\gamma_t)}{\Gamma\left[\sum_{t=1}^{T}(n_{s,a}^{t}+\gamma_t)\right]}$$
$$\times \prod_{m=1}^{M} exp\left\{\lambda\frac{\sum_{t_j\in L_m}\mathcal{I}(t_j=t)}{|L_m|}\right\}, \quad (4)$$

where $n_{s,a}^{t}$ equals the number of sentences in reviews associated with attribute $a$, sentiment $s$, and aspect $t$; $M$ is the number of sentences in reviews; $L_m$ is the set of sentences corresponding to sentence $m$.

Similarly, for the third term in Equation 2, by integrating over $\Theta = \{\boldsymbol{\theta_i}\}_{i=1}^{D}$, we obtain (Section 4.1 explains $\beta_s$):

$$p(\boldsymbol{s}|\boldsymbol{\beta}) = \int p(\boldsymbol{s}|\Theta)p(\Theta|\boldsymbol{\beta})\mathrm{d}\Theta$$
$$= \left(\frac{\Gamma(\sum_{s=1}^{S}\beta_s)}{\prod_{s=1}^{S}\Gamma(\beta_s)}\right)^{D} \times \prod_{d=1}^{D}\frac{\prod_{s=1}^{S}\Gamma(n_d^s+\beta_s)}{\Gamma\left[\sum_{s=1}^{S}(n_d^s+\beta_s)\right]}, \quad (5)$$

where $D$ is the number of reviews; $n_d^s$ is the number of times that a sentence from review $d$ is associated with sentiment $s$; and $n_d$ is the number of sentences in review $d$.

For each sweep of a Gibbs iteration, we sample latent aspect $t$ and sentiment $s$ as follows:

$$p(s_i = s, t_i = t|\boldsymbol{s_{-i}}, \boldsymbol{t_{-i}}, \boldsymbol{w}, a)$$
$$\propto \frac{n_{s,a,-i}^{t}+\gamma_t}{\sum_{t=1}^{T}(n_{s,a,-i}^{t}+\gamma_t)} \times \frac{n_{d,-i}^{s}+\beta_s}{\sum_{s=1}^{S}(n_{d,-i}^{s}+\beta_s)}$$
$$\times \frac{\prod_{v\in W_i}\prod_{c=0}^{C_v^i-1}(n_{s,t,-i}^{v}+\alpha_v+c)}{\prod_{c=0}^{C_i-1}(n_{s,t,-i}+\sum_{v=1}^{W}\alpha_v+c)} \quad (6)$$
$$\times exp\left\{\lambda\frac{\sum_{t_j\in L_i}\mathcal{I}(t_j=t)}{|L_i|}\right\},$$

where $n_{s,a}^{t}$ is the number of sentences from reviews associated with attribute $a$, sentiment $s$, and aspect $t$; $n_d^s$ is the number of sentences from review $d$ associated with sentiment $s$; $W_i$ is the set of words in sentence $i$. $C_v^i$ is the count of word $v$ in sentence $i$; $C_i$ is the number of words in sentence $i$; $n_{s,t}^{v}$ is the number of words $v$ assigned sentiment $s$ and aspect $t$; $n_{s,t}$ is the number of words assigned sentiment $s$ and aspect $t$ in all reviews; $L_i$ is the set of corresponding sentences of sentence $i$; and an index of $-i$ indicates excluding sentence $i$ from the count.

Equations 7, 8, and 9, respectively, approximate the probabilities of word $w$ occurring given sentiment $s$ and aspect $t$; of aspect $t$ of a sentence occurring given sentiment $s$ and attribute $a$; of sentiment $s$ occurring given document $d$.

$$\phi_{s,t,w} = \frac{n_{s,t}^{w}+\alpha_w}{\sum_{v=1}^{W}(n_{s,t}^{w}+\alpha_v)}, \quad (7)$$

$$\psi_{s,t,a} = \frac{n_{s,a}^{t}+\gamma_t}{\sum_{t=1}^{T}(n_{s,a}^{t}+\gamma_t)}, \quad (8)$$

$$\theta_{d,s} = \frac{n_d^s+\beta_s}{\sum_{s=1}^{S}(n_d^s+\beta_s)}. \quad (9)$$

The generalized Pólya Urn model (Mahmoud, 2008) has been used for encoding word co-

occurrence information into topic models. Consider an urn containing a mixture of balls, each of which is tagged with a term, for each sampling sweep we draw a ball from the urn. In a standard Pólya Urn model, as used in LDA, we return the ball to the urn with another ball tagged with the same term. This process provides burstiness of the probability of seeing a term but ignores the covariance. The probability increase of one term decreases the probability of the other words. In the generalized Pólya Urn model, when a ball is drawn from the urn, we replace it with two new balls with a set of balls tagged with related terms. Similar to previous models (Mimno et al., 2011; Chen et al., 2013; Fei et al., 2014; Zhang and Singh, 2018), to increase the probability of having semantically related words appear in the same topic, Trait applies a generalized Pólya Urn model in each Gibbs sweep and uses weight $\varepsilon$ to promote related words based on the cosine similarity between their Word2Vec (Mikolov et al., 2013) embeddings.

## 4 Evaluation

To assess Trait's effectiveness, we select the hotel and restaurant domains and prepare four review datasets associated with three attributes: author, trip type, and location. HotelUser, HotelType, and HotelLoc are sets of hotel reviews collected from TripAdvisor. HotelUser contains 28,165 reviews posted by 202 randomly selected reviewers, each of whom contributes at least 100 hotel reviews. HotelType contains reviews associated with five trip types including business, couple, family, friend, and solo. HotelLoc contains a total of 136,446 reviews about seven US cities, split approximately equally. ResUser is a set of restaurant reviews from Yelp Dataset Challenge (2019). It contains 23,874 restaurant reviews posted by 144 users, each of whom contributes at least 100 reviews. Table 1 summarizes our datasets. Datasets and source code are available for research purposes (Trait, 2019).

Table 1: Summary of the evaluation datasets.

| Statistic | HotelUser | ResUser | HotelType | HotelLoc |
|---|---|---|---|---|
| # of reviews | 28,165 | 23,873 | 22,984 | 136,446 |
| # of sentences | 362,153 | 276,008 | 302,920 | 1,428,722 |
| Sentence/Review | 13 | 12 | 13 | 10 |
| Words/Sentence | 8 | 7 | 7 | 7 |

We remove stop words and HTML tags, expand typical abbreviations, and mark special named entities using a rule-based algorithm (e.g., replace a URL by #LINK# and replace a monetary amount by #MONEY#) and the Stanford named entity recognizer (Finkel et al., 2005). We use Porter's (1980) stemming algorithm. To handle negation, for any word pair whose first word is *no*, *not*, or *nothing*, we replace the word pair by a negated term, e.g., producing *not_work* and *not_quiet*. Finally, we split each review into constituent sentences.

### 4.1 Parameter Settings

Trait includes three manually tuned hyperparameters that have a smoothing effect on the corresponding multinomial distributions. Hyperparameter $\alpha$ is the Dirichlet prior of the word distribution. We use asymmetric priors based on a sentiment lexicon. Table 2 shows Trait's sentiment word list as prior knowledge to set asymmetric priors. This list extends Turney and Littman's (2003) list with additional general sentiment words. For any word in the positive list, we set $\alpha$ to 0 if this word appears in a sentence assigned a negative sentiment, and to 5 if this word appears in a sentence assigned a positive sentiment, and conversely for words in the negative list. For all remaining words, we set $\alpha$ to 0.05. We set hyperparameter $\beta$, the Dirichlet prior of the sentiment distribution, to 5 for both sentiments. We set hyperparameter $\gamma$, the Dirichlet prior of the aspect distribution, to $50/T$ for all models, where $T$ is the number of aspects. We set the reinforcement weight of structural and semantic correspondence $\lambda$ to 1.0; sentence semantic similarity $\rho$ to 0.7; and, related word promoting weights to 0.3 for hotel reviews and 0.1 for restaurant reviews.

### 4.2 Quantitative Evaluation

Whether topics (word clusters) are semantically cohesive is crucial in assessing topic modeling approaches. As in previous studies (Nguyen et al., 2015a,b; Yang et al., 2017), we adopt Normalized Pointwise Mutual Information (NPMI) (Lau et al., 2014) and W2V (O'Callaghan et al., 2015) as our evaluation metrics. Higher NPMI and W2V scores indicate greater semantic cohesion. We compare Trait with four baselines: AT, JST, ASUM, and AATS. We perform our evaluation on HotelUser and ResUser based on the top 20 words in each sentiment-aspect pair. We split data into five folds

Table 2: Lists of sentiment words.

| **Positive** |
|---|
| amazing, attractive, awesome, best, comfortable, correct, enjoy, excellent, fantastic, favorite, fortunate, free, fun, glad, good, great, happy, impressive, love, nice, not_bad, perfect, positive, recommend, satisfied, superior, thank, worth |
| **Negative** |
| annoying, bad, complain, disappointed, hate, inferior, junk, mess, nasty, negative, not_good, not_like, not_recommend, not_worth, poor, problem, regret, slow, small, sorry, terrible, trouble, unacceptable, unfortunate, upset, waste, worst, worthless, wrong |

and use training split to train all models. For each number of aspects, we conduct a two-tailed paired t-test for each of the pairwise comparisons. Throughout, $*$, †, and ‡ indicate significance at 0.05, 0.01, and 0.001, respectively.

Table 3 shows average NMPI and W2V scores for different numbers of aspects. AT performs worst, possibly due to missing conditions on sentiments. ASUM and JST are comparable. Trait outperforms all others, with the highest NMPI and W2V scores for each number of aspects. Table 4 shows similar conclusions for restaurant reviews.

For both datasets, Trait's improvements of topic coherence over baseline models are statistically significant for HotelUser ($p < 0.001$) and ResUser ($p = 0.002$). Trait allows reviews written by the same or similar authors to have idiosyncratic preferences over aspects and sentiments. Trait assigns aspects to sentences by sampling attribute-specific aspect distributions. These distributions are regularized by the Markov Random Fields. Sentences with a high degree of correspondence have a high probability to be assigned the same aspects.

### 4.3 Sentiment Classification

Automatically detecting the sentiment of a document is an important task in sentiment analysis. We compare Trait with JST, ASUM, and AATS for document-level sentiment classification using HotelUser and ResUser. We use integer ratings of reviews to collect ground-truth labels. Reviews with ratings at three and above are labeled as positive and the rest are labeled as negative. Note that our datasets are imbalanced. We conduct five-fold cross-validation with the two-tailed paired t-test. For each user, we use 80% of reviews for training and 20% for testing. For evaluation metrics, we adopt accuracy (Acc) and area under the curve (AUC) of the Receiver Operating Characteristic (ROC) curve. ROC plots the true positive rate against the false positive rate. AUC-ROC is a standard metric for evaluating classifiers' performance on imbalanced data.

Table 3: Topic coherence: Hotel reviews.

| NPMI | T=10 | T=20 | T=30 | T=40 | T=50 | T=60 |
|---|---|---|---|---|---|---|
| AT | 3.64 | 4.04 | 4.37 | 4.49 | 4.86 | 5.14 |
| AATS | 5.63 | 9.08 | 10.41 | 10.78 | 11.05 | 11.00 |
| JST | 8.99 | 10.78 | 11.45 | 11.54 | 11.56 | 11.46 |
| ASUM | 9.48 | 10.64 | 11.02 | 11.33 | 11.39 | 11.56 |
| Trait | **15.50**$^{\ddagger}$ | **16.91**$^{\ddagger}$ | **17.31**$^{\ddagger}$ | **17.32**$^{\ddagger}$ | **16.46**$^{\ddagger}$ | **15.34**$^{\ddagger}$ |
| W2V | T=10 | T=20 | T=30 | T=40 | T=50 | T=60 |
| AT | 0.10 | 0.10 | 0.10 | 0.09 | 0.09 | 0.09 |
| AATS | 0.13 | 0.16 | 0.18 | 0.18 | 0.18 | 0.18 |
| JST | 0.15 | 0.18 | 0.18 | 0.19 | 0.19 | 0.19 |
| ASUM | 0.17 | 0.18 | 0.18 | 0.18 | 0.18 | 0.18 |
| Trait | **0.33**$^{\ddagger}$ | **0.35**$^{\ddagger}$ | **0.35**$^{\ddagger}$ | **0.34**$^{\ddagger}$ | **0.32**$^{\ddagger}$ | **0.31**$^{\ddagger}$ |

Table 4: Topic coherence: Restaurant reviews.

| NPMI | T=10 | T=20 | T=30 | T=40 | T=50 | T=60 |
|---|---|---|---|---|---|---|
| AT | 5.64 | 5.21 | 5.30 | 5.65 | 6.54 | 7.94 |
| AATS | 6.05 | 8.02 | 9.03 | 9.35 | 9.90 | 9.95 |
| JST | 9.46 | 11.13 | 11.73 | 11.92 | 12.14 | **12.31** |
| ASUM | 8.81 | 9.7 | 9.92 | 10.09 | 10.07 | 10.04 |
| Trait | **11.27**$^{\ddagger}$ | **13.02**$^{\ddagger}$ | **13.62**$^{\ddagger}$ | **13.18**$^{\ddagger}$ | **12.36** | 11.95 |
| W2V | T=10 | T=20 | T=30 | T=40 | T=50 | T=60 |
| AT | 0.16 | 0.15 | 0.14 | 0.13 | 0.13 | 0.14 |
| AATS | 0.11 | 0.14 | 0.16 | 0.17 | 0.18 | 0.18 |
| JST | 0.21 | 0.21 | 0.20 | 0.20 | 0.20 | 0.19 |
| ASUM | 0.20 | 0.19 | 0.18 | 0.18 | 0.17 | 0.17 |
| Trait | **0.25**$^{\ddagger}$ | **0.26**$^{\ddagger}$ | **0.25**$^{\ddagger}$ | **0.24**$^{\ddagger}$ | **0.24**$^{\ddagger}$ | **0.24**$^{\ddagger}$ |

Table 5: Accuracy and AUC of sentiment classification on hotel reviews.

| | T=20 | | T=40 | | T=60 | |
|---|---|---|---|---|---|---|
| | Acc | AUC | Acc | AUC | Acc | AUC |
| AATS | 0.79 | 0.45 | 0.82 | 0.48 | 0.84 | 0.48 |
| JST | 0.61 | 0.82 | 0.64 | 0.83 | 0.67 | 0.84 |
| ASUM | 0.80 | 0.83 | 0.84 | 0.84 | 0.87 | 0.83 |
| Trait | **0.85**$^{\dagger}$ | **0.86**$^{*}$ | **0.87**$^{*}$ | **0.85**$^{*}$ | **0.88** | **0.86**$^{\dagger}$ |

Table 5 reports the results of sentiment classification on hotel reviews. AATS achieves better accuracy but worse AUC scores than JST. ASUM yields better accuracy and comparable AUC scores compared with JST. Trait consistently outperforms all baseline models given different aspect numbers with an average gain in accuracy of 3%. Incorporating attributes and structural and semantic cor-

respondence into conditional probability distributions greatly benefit the model in capturing dependencies among attributes, aspects, and sentiments.

Table 6: Accuracy and AUC of sentiment classification on restaurant reviews.

| | T=20 | | T=40 | | T=60 | |
|---|---|---|---|---|---|---|
| | Acc | AUC | Acc | AUC | Acc | AUC |
| AATS | 0.79 | 0.47 | 0.80 | 0.48 | 0.82 | 0.50 |
| JST | 0.59 | 0.71 | 0.61 | 0.73 | 0.64 | 0.73 |
| ASUM | 0.80 | 0.78 | 0.84 | 0.78 | 0.87 | 0.74 |
| Trait | **0.86**$^{\dagger}$ | **0.79** | **0.87**$^{\dagger}$ | 0.78 | **0.88** | **0.79**$^{*}$ |

### 4.4 Attribute Profile

Given reviews with selected attributes, we expect Trait to generate profiles representing the characteristics associated with those attributes. To evaluate profiles, we run Trait on HotelUser, HotelLoc, and HotelType, associated with three attributes: authors, locations, and trip types, respectively.

#### 4.4.1 Summarization

Trait outputs profiles that summarize attributes in terms of aspects and sentiments in reviews.

Figure 3 shows the profiles of four US cities generated by Trait. We visualize the profiles as aspect-clouds using the top 30 aspects in each sentiment. The size of the aspect label corresponds to its aspect probabilities. Due to space constraints, we place the profiles of Boston, Chicago, and Orlando in Table 7.

These profiles yield salient summaries for each city. For example, *Strip* and *Casino* are the top two positive aspects for Las Vegas, a resort city for gambling. We see from the reviews that most hotels with high ratings are located on the Strip. For Boston, Chicago, and New York, *Location* and *PublicTrans* are the top positive aspects. These cities rank top on the lists of U.S. cities with high transit ridership (Ridership, 2019) and walkability (Walkability, 2019). Hotels' proximity to public transportation, shopping, restaurants, and attractions is appealing to several reviewers. We see *RoomSize* appears in the top five negative aspects. These three cities have among the most expensive hotel room rates (Statista, 2019). Assuming consumers expect more when they pay more, we conjecture that a failed expectation could be caused by room size, especially in New York, where room sizes are smaller than elsewhere in the US (NYC, 2019). For Miami, *Transportation* is attractive, presumably because many cruises depart from Miami.

Table 7: Top five aspects discovered by Trait.

| Boston (P) | Boston (N) | Chicago (P) | Chicago (N) |
|---|---|---|---|
| PublicTrans | RoomSize | Location | RoomSize |
| Location | Service | PublicTrans | Service |
| Decor | Parking | Decor | Parking |
| Bar | RoomNoise | Bar | Elevator |
| Service | StreetNoise | Service | ACHotwater |
| **Orlando (P)** | **Orlando (N)** | **Friend (P)** | **Friend (N)** |
| ThemePark | Pool | Tour | Upgrade |
| Pool | Transportation | Value | Staff |
| Food | Supplies | View | BreakfastArea |
| Amenity | Frontdesk | Breakfast | Bathroom |
| Overall | Bed | Location | Breakfast |
| **Author A (P)** | **Author A(N)** | **Author B (P)** | **Author B (N)** |
| Helpfulness | NotReturn | Helpfulness | NotReturn |
| View | Value | View | Value |
| Value | Charge | Breakfast | Checkin |
| Breakfast | Checkout | Value | HotelSize |
| Comfort | Checkin | Transportation | Checkout |
| **Author C (P)** | **Author C (N)** | **Author D (P)** | **Author D (N)** |
| Internet | Elevator | Staff | Value |
| Comfort | Room | Room | NotReturn |
| TripType | Food | Recommend | Checkout |
| View | StreetNoise | Value | Room |
| Breakfast | Checkout | TripType | Charge |

Figure 4 shows the results for HotelType: *Cleanliness*, *Internet*, *TV*, and *Upgrade* are most likely to lead to a negative sentiment for business travelers. For couples, *Atmosphere* and *RestArea* are most preferred positive aspects. *Family*, *Tour*, and *Attraction* are most positive aspects for families. Solo travelers, on business or tourism, express most opinions toward both *Transportation*.

Table 7 (bottom rows) lists the top five aspects for four authors from HotelUser. We can observe strong commonality between Author A and B, They both like to express positive sentiment on *Helpfulness*, *View*, *Value*, and *Breakfast*. They are like to express not returning a hotel and the negative sentiments are mostly toward *Value*, *Checkin*, and *Checkout*. There is little commonality between Authors C and D. For Author C, *Internet* and *Comfort* are most attractive whereas *Staff* and *Room* are most appealing aspects for Author D. In terms of negatives, *Checkout* is the only aspect shared between Authors C and D.

#### 4.4.2 Similarity

Attribute profiles can be used not only for summarization, but also for measuring similarity between attribute values with respect to aspects and sentiments, which can support attribute-based applica-

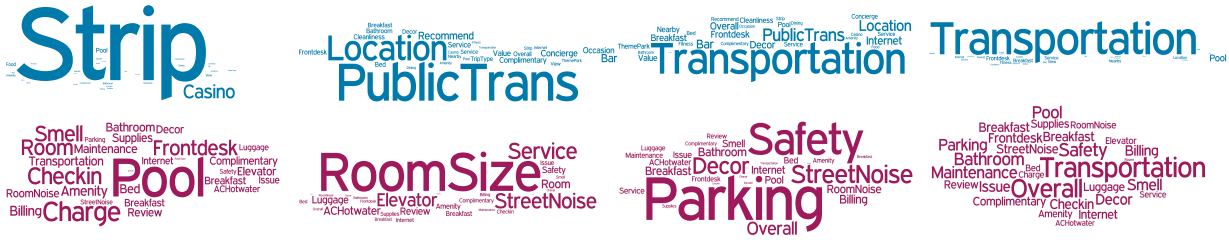


Figure 3: An aspect-cloud visualization of US cities (positive aspects above; negative aspects below).

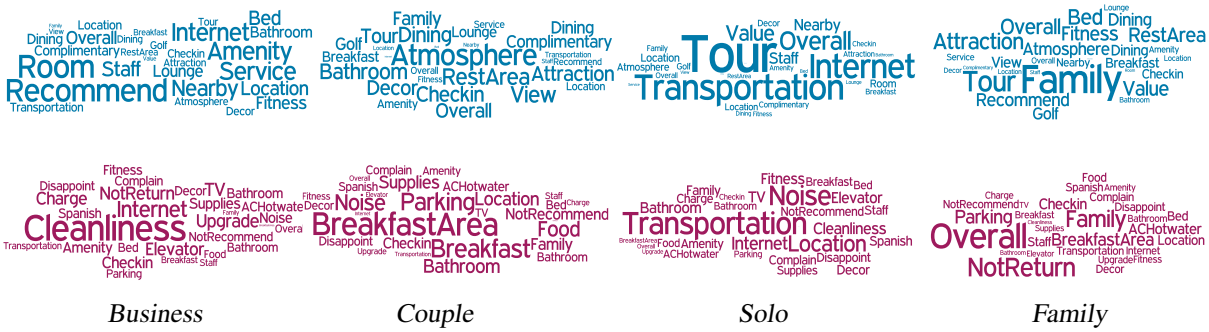


Figure 4: An aspect-cloud visualization of trip types. (positive aspects above; negative aspects below).

tions such as recommender systems. Our metric of similarity between distinct values of the same attribute is the Jensen-Shannon distance (JSD) (Endres and Schindelin, 2003), the square root of Jensen-Shannon divergence. We compute, $D_{\text{JS}}$, the JSD of attribute profiles $P$ and $Q$ as

$$D_{\text{JS}} = \sqrt{\frac{1}{2}D_{KL}(P||M) + \frac{1}{2}D_{KL}(Q||M)} \qquad (10)$$
$$M = \frac{1}{2}(P+Q),$$

where $D_{KL}(P||Q)$ is the Kullback-Leibler (KL) divergence of probability distributions $P = \{p_1, \ldots, p_n\}$ and $Q = \{q_1, \ldots, q_n\}$:

$$D_{KL}(P||Q) = \sum_i p_i \log \frac{p_i}{q_i}. \qquad (11)$$

As a baseline, we use a vector space model based on USE sentence embeddings. We calculate mean sentence embeddings for each review. Then, given two sets of reviews, $D = \{d_1, \ldots, d_m\}$ and $R = \{r_1, \ldots, r_n\}$, we compute their similarity as follows (here $sim(d_i, r_j)$ is the cosine similarity between review $d_i$ and $r_j$):

$$S_{D,R} = \frac{1}{m+n}\sum_{i=1}^{m}\sum_{j=1}^{n} sim(d_i, r_j). \qquad (12)$$

Figure 5a shows the similarities among the profiles of the seven cities generated by the baseline model. We see that Boston is close to New York and Chicago; Las Vegas is far away from Los Angeles and Miami but close to New York, Boston, and Chicago. Figure 5b shows the results generated by Trait. Here, dissimilarity corresponds to distance normalized to $[0, 1]$. Boston, Chicago, and New York are close to each other, as are Los Angeles and Miami; Las Vegas is far from each of the others; and, Orlando is far from all except Miami. Trait's results are arguably more plausible than what the baseline approach produces.

As discussed earlier, hotels in Boston, Chicago, and New York have common characteristics; Las Vegas differs strongly from the others because it is a resort city and its major hotels are combined with casinos; Orlando is a tourism destination but differs from Las Vegas in that it is famous for local attractions, such as theme parks. Orlando's profile exhibits that travelers there tend to be more aware of aspect *Attraction* than elsewhere. An interesting pair is Los Angeles and Miami. We see that *Location* appears as the most important aspect on the positive side for both of them. Such a similarity could be partially explained by the fact that both Los Angeles and Miami serve as locations for taking cruises. Also, the common aspect *Safety*

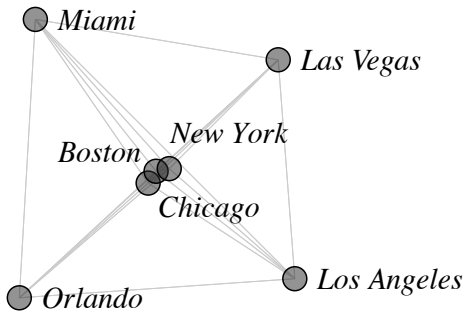


(a) Baseline.

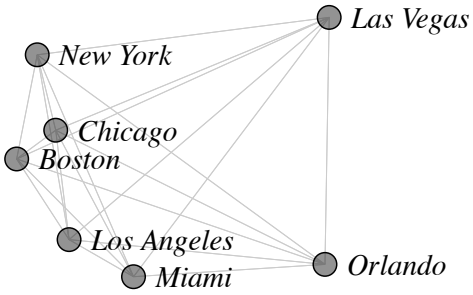


(b) Trait.

Figure 5: Force fields of similarities: cities.

(negative for both) could increase their similarity.

Figure 6a shows similarities among the five trip types generated by the baseline model. Business is closer to Family than Solo and Friend; Solo is closer to Family than Friend. Trait generates more reasonable results, as shown in Figure 6b. Business is far from others but is closer to Friend and Solo than to Couple and Family. Couple, Family, and Friend are relatively close to each other. Business reviewers attend to different aspects from other reviewers. Further, Solo and Friend contain reviews of business trips, although the authors did not select Business as the trip type. However, this situation does not happen for Couple and Family.

Figure 7 shows similarities among 20 authors, including the four authors mentioned in Section 4.4.1. We see that Authors A and B are close to each other whereas Authors C and D are far away from each other. The results are aligned with their aspect and sentiment profiles.

## 5 Discussion and Conclusion

Trait not only shows that capturing structural and semantic correspondence leads to improved performance in terms of coherence and naturalness of the aspects discovered but can also be realized in an unsupervised framework. Trait outperforms competing approaches across multiple datasets.

These results open up interesting directions for

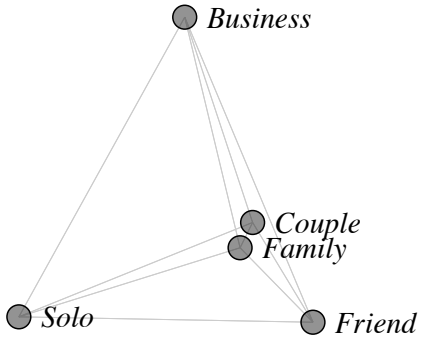


(a) Baseline.

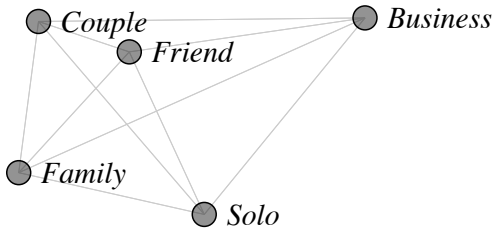


(b) Trait.

Figure 6: Force fields of similarities: Purpose.

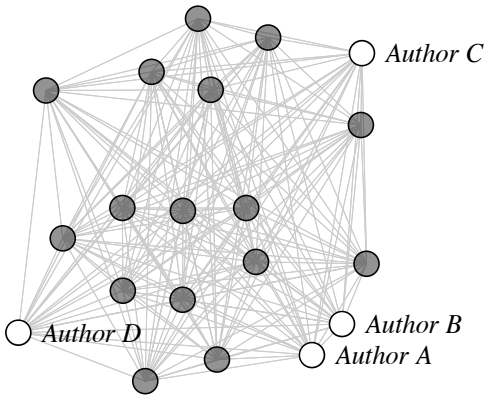


Figure 7: Force field of similarities: 20 authors.

future work. One direction is to learn disentangled latent representations for attributes in neural network's space, such as for disentangling aspects (Jain et al., 2018), text style (John et al., 2019), and syntax and semantics (Chen et al., 2019; Bao et al., 2019). Another direction is to develop a content-based recommender based on Trait, since it provides an effective unsupervised solution for generating profiles based on different attributes.

## 6 Acknowledgments

We would like to thank the anonymous reviewers for helpful comments and corrections. MPS would like to acknowledge the US Department of Defense for partial support through the NCSU Laboratory for Analytic Sciences.

## References


Ahmad Riduan Bahauddin and Sharifudin Md. Shaarani. 2015. The impact of geographical location on taste sensitivity and preference. *International Food Research Journal*, 22(2):731–738.

Yu Bao, Hao Zhou, Shujian Huang, Lei Li, Lili Mou, Olga Vechtomova, Xin-yu Dai, and Jiajun Chen. 2019. Generating sentences from disentangled syntactic and semantic spaces. In *Proceedings of the* $57^{th}$ *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 6008–6019, Florence.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder for English. In *Proceedings of the* $23^{rd}$ *Conference on Empirical Methods in Natural Language Processing (EMNLP): System Demonstrations*, pages 169–174, Brussels.

Mingda Chen, Qingming Tang, Sam Wiseman, and Kevin Gimpel. 2019. A multi-task approach for disentangling syntax and semantics in sentence representations. In *Proceedings of the* $17^{th}$ *Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 2453–2464, Minneapolis.

Zhiyuan Chen, Arjun Mukherjee, Bing Liu, Meichun Hsu, Malú Castellanos, and Riddhiman Ghosh. 2013. Exploiting domain knowledge in aspect extraction. In *Proceedings of the* $18^{th}$ *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1655–1667, Seattle.

Qiming Diao and Jing Jiang. 2013. A unified model for topics, events and users on Twitter. In *Proceedings of the* $18^{th}$ *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1869–1879, Seattle.

Dominik Maria Endres and Johannes E. Schindelin. 2003. A new metric for probability distributions. *IEEE Transactions on Information Theory*, 49(7):1858–1860.

Geli Fei, Zhiyuan Chen, and Bing Liu. 2014. Review topic discovery with phrases using the Pólya Urn model. In *Proceedings of the* $25^{th}$ *International Conference on Computational Linguistics (COLING)*, pages 667–676, Dublin.

Jenny Rose Finkel, Trond Grenager, and Christopher D. Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the* $43^{rd}$ *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 363–370, Ann Arbor.

Sarthak Jain, Edward Banner, Jan-Willem van de Meent, Iain J. Marshall, and Byron C. Wallace. 2018. Learning disentangled representations of texts with application to biomedical abstracts. In *Proceedings of the* $23^{rd}$ *Conference on Empirical Methods in Natural Language Processing (EMNLP): System Demonstrations*, pages 4683–4693, Brussels.

Yohan Jo and Alice Haeyun Oh. 2011. Aspect and sentiment unification model for online review analysis. In *Proceedings of the* $4^{th}$ *ACM International Conference on Web Search and Data Mining (WSDM)*, pages 815–824, Hong Kong.

Vineet John, Lili Mou, Hareesh Bahuleyan, and Olga Vechtomova. 2019. Disentangled representation learning for non-parallel text style transfer. In *Proceedings of the* $57^{th}$ *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 424–434, Florence.

Hyungsul Kim, Yizhou Sun, Julia Hockenmaier, and Jiawei Han. 2012. ETM: Entity topic models for mining documents associated with entities. In *Proceedings of the* $12^{th}$ *IEEE International Conference on Data Mining (ICDM)*, pages 349–358, Brussels.

Suin Kim, Jianwen Zhang, Zheng Chen, Alice H. Oh, and Shixia Liu. 2013. A hierarchical aspect-sentiment model for online reviews. In *Proceedings of the* $27^{th}$ *AAAI Conference on Artificial Intelligence (AAAI)*, pages 804–812, Bellevue.

Jey Han Lau, David Newman, and Timothy Baldwin. 2014. Machine reading tea leaves: Automatically evaluating topic coherence and topic model quality. In *Proceedings of the* $14^{th}$ *Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 530–539, Gothenburg.

Chenghua Lin, Yulan He, Richard Everson, and Stefan M. Rüger. 2012. Weakly supervised joint sentiment-topic detection from text. *IEEE Transactions on Knowledge and Data Engineering*, 24(6):1134–1145.

Jun S. Liu. 1994. The collapsed Gibbs sampler in Bayesian computations with applications to a gene regulation problem. *Journal of the American Statistical Association*, 89(427):958–966.

Hosam Mahmoud. 2008. *Pólya Urn Models*. Texts in Statistical Science. Chapman & Hall/CRC, London.

Josh H. McDermott, Alan F. Schultz, Eduardo A Undurraga, and Ricardo A. Godoy. 2016. Indifference to dissonance in native Amazonians reveals cultural variation in music perception. *Nature*, 535:547–550.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the* $27^{th}$ *Annual Conference on Neural Information Processing Systems (NIPS)*, pages 3111–3119, Lake Tahoe.

David M. Mimno, Hanna M. Wallach, Edmund M. Talley, Miriam Leenders, and Andrew McCallum. 2011. Optimizing semantic coherence in topic models. In *Proceedings of the* $16^{th}$ *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 262–272, Edinburgh.

Subhabrata Mukherjee, Gaurab Basu, and Sachindra Joshi. 2014. Joint author sentiment topic model. In *Proceedings of the* $14^{th}$ *International Conference on Data Mining (SDM)*, pages 370–378, Philadelphia.

Dat Quoc Nguyen, Richard Billingsley, Lan Du, and Mark Johnson. 2015a. Improving topic models with latent feature word representations. *Transactions of the Association for Computational Linguistics (TACL)*, 3:299–313.

Thang Nguyen, Jordan L. Boyd-Graber, Jeffrey Lund, Kevin D. Seppi, and Eric K. Ringger. 2015b. Is your anchor going up or down? Fast and accurate supervised topic models. In *Proceedings of the* $14^{th}$ *Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 746–755, Denver.

NYC. 2019. About New York hotels. http://www.nyc.com/visitor_guide/about_new_york_hotels.703528/editorial_review.aspx. Accessed: 05/20/2019.

Derek O'Callaghan, Derek Greene, Joe Carthy, and Pádraig Cunningham. 2015. An analysis of the coherence of descriptors in topic modeling. *Expert Systems with Applications*, 42(13):5645–5657.

Christian S. Perone, Roberto Silveira, and Thomas S. Paula. 2018. Evaluation of sentence embeddings in downstream and linguistic probing tasks. *CoRR*, abs/1806.06259.

Lahari Poddar, Wynne Hsu, and Mong-Li Lee. 2017. Author-aware aspect topic sentiment model to retrieve supporting opinions from reviews. In *Proceedings of the* $22^{nd}$ *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 472–481, Copenhagen.

Martin F. Porter. 1980. An algorithm for suffix stripping. *Program: Electronic Library and Information Systems*, 14(3):130–137.

Ridership. 2019. List of U.S. cities with high transit ridership. https://en.wikipedia.org/wiki/List_of_U.S._cities_with_high_transit_ridership. Accessed: 08/24/2019.

Michal Rosen-Zvi, Thomas L. Griffiths, Mark Steyvers, and Padhraic Smyth. 2004. The author-topic model for authors and documents. In *Proceedings of the* $20^{th}$ *Conference in Uncertainty in Artificial Intelligence (UAI)*, pages 487–494, Banff, Canada.

Statista. 2019. A ranking of the most expensive cities in the US. http://www.statista.com/statistics/214585/most-expensive-cities-in-the-us-ordered-by-hotel-prices-2010/. Accessed: 05/20/2019.

Ivan Titov and Ryan T. McDonald. 2008a. A joint model of text and aspect ratings for sentiment summarization. In *Proceedings of the* $46^{th}$ *Annual Meeting on Association for Computational Linguistics (ACL)*, pages 308–316, Columbus, Ohio.

Ivan Titov and Ryan T. McDonald. 2008b. Modeling online reviews with multi-grain topic models. In *Proceedings of the* $17^{th}$ *International Conference on World Wide Web (WWW)*, pages 308–316, Beijing.

Trait. 2019. https://research.csc.ncsu.edu/mas/code/trait/. Accessed: 08/17/2019.

Peter D. Turney and Michael L. Littman. 2003. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Transactions on Information Systems*, 21(4):315–346.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the* $31^{st}$ *Annual Conference on Neural Information Processing Systems (NIPS)*, pages 6000–6010, Long Beach.

Walkability. 2019. Most walkable cities in the United States. https://www.walkscore.com/cities-and-neighborhoods/. Accessed: 08/24/2019.

Shuai Wang, Zhiyuan Chen, and Bing Liu. 2016. Mining aspect-specific opinion using a holistic lifelong topic model. In *Proceedings of the* $25^{th}$ *International Conference on World Wide Web (WWW)*, pages 167–176, Montreal.

Weiwei Yang, Jordan L. Boyd-Graber, and Philip Resnik. 2017. Adapting topic models using lexical associations with tree priors. In *Proceedings of the* $22^{nd}$ *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1901–1906, Copenhagen.

Yelp. 2019. Yelp dataset challenge. https://www.yelp.com/dataset_challenge/. Accessed: 05/20/2019.

Zhe Zhang and Munindar P. Singh. 2018. Limbic: Author-based sentiment aspect modeling regularized with word embeddings and discourse relations. In *Proceedings of the* $23^{rd}$ *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3412–3422, Brussels.

# From the *Token* to the *Review*: A Hierarchical Multimodal approach to Opinion Mining

**Alexandre Garcia, Pierre Colombo,**
**Slim Essid, Florence d'Alché-Buc, Chloé Clavel**
Télécom ParisTech
Université Paris Saclay
{garcia,pcolombo,sessid,fdalche,cclavel}@telecom-paristech.fr

## Abstract

The task of predicting fine grained user opinion based on spontaneous spoken language is a key problem arising in the development of Computational Agents as well as in the development of social network based opinion miners. Unfortunately, gathering reliable data on which a model can be trained is notoriously difficult and existing works rely only on coarsely labeled opinions. In this work we aim at bridging the gap separating fine grained opinion models already developed for written language and coarse grained models developed for spontaneous multimodal opinion mining. We take advantage of the implicit hierarchical structure of opinions to build a joint fine and coarse grained opinion model that exploits different views of the opinion expression. The resulting model shares some properties with attention-based models and is shown to provide competitive results on a recently released multimodal fine grained annotated corpus.

## 1 Introduction

Recent years have witnessed the increasing popularity of social networks and video streaming platforms. People heavily rely on these channels to express their opinions through video-based discussions or reviews. Whereas such opinionated data has been widely studied in the context of written customer reviews (Liu, 2012) crawled on websites such as Amazon (Hu and Liu, 2004) and IMDB (Maas et al., 2011), only a few studies have been proposed in the case of video-based reviews. Such multimodal data has been shown to provide a mean to disambiguate some hard to understand opinion expressions such as irony and sarcasm (Attardo et al., 2003) and contains crucial information indicating the level of engagement and the persuasiveness of the speaker (Clavel and Callejas, 2016; Ben Youssef et al., 2019; Nojavanasghari et al., 2016). A key problem in this context is the lack of availability of fine grained opinion annotation *i.e.* annotations performed at the token or short span level and highlighting on the components of the structure of opinions. Indeed whereas such resources have been gathered in the case of textual data and can be used to deeply understand the expression of opinions (Wiebe et al., 2005; Pontiki et al., 2016), the different attempts at annotating multimodal reviews have shown that reaching good annotator agreement is nearly impossible at a fine grained level. This results from the disfluent aspect of spontaneous spoken language making it difficult to choose opinions' annotation boundaries (Garcia et al., 2019; Langlet and Clavel, 2015b). Thus the price to pay to gather reliable data is the definition of an annotation scheme focusing on coarse grained information such as long segment categorization as done by Zadeh et al. (2016a) or review level annotation (Park et al., 2014). Building models able to predict fine grained opinion information in a multimodal setting is in fact of high importance in the context of designing human–robot interfaces (Langlet and Clavel, 2016). Indeed the knowledge of opinions decomposed over a set of polarities associated to some targets is a building block of automatic human understanding pipelines (Langlet and Clavel, 2015a). The present work is motivated by the following observations:

- Despite the lack of reliability of fine grained labels collected for multimodal data, the redundancy of the opinion information contained at different granularities can be leveraged to reduce the inherent noise of the labelling process and to build improved opinion predictors. We build a model that takes advantage of this property and joinlty models the different components of an opinion.

- Hierarchical multi-task language models have been recently shown to improve upon the single tasks' models (Sanh et al., 2018). A careful choice of the tasks and the order in which



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5539–5548,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

they are sequentially presented to the model has been proved to be the key to build competitive predictors. It is not clear whether such type of hierarchical model could be adapted to handle multimodal data with the state of the art neural architectures (Zadeh et al., 2018a,b). We discuss in the experimental section the strategies and models that are adapted to the multimodal opinion mining context.

- In the case where no fine grained supervision is available, the attention mechanism (Vaswani et al., 2017) provides a compelling alternative to build models generating interpretable decisions with token-level explanations (Hemamou et al., 2018). In practice such models are notoriously hard to train and require the availability of very large datasets. On the other hand, the injection of fine-grained polarity information has been shown to be a key ingredient to build competitive sentiment predictors by Socher et al. (2013). Our hierarchical approach can be interpreted under the lens of attention-based learning where some supervision is provided at training to counterbalance the difficulty of learning meaningful patterns with spoken language data. We specifically experimentally show that providing this supervision is here necessary to build competitive predictors due to the limited number of data and the difficulty to extract meaningful patterns from it.

## 2 Background on fine grained opinion mining

The computational models of opinion are grounded in a linguistic framework defining how these objects can be structured over a set of interdependent functional parts. In this work we focus on the model of Martin and White (2013) that defines the expression of opinions as an *evaluation* towards an object. The expression of such evaluations can be summarized by the combination of three components: a *source* (mainly the speaker) expressing a statement on a *target* identifying the entity evaluated and a *polarized expression* making the attitude of the source explicit. In the literature, the task of finding the words indicating these components and categorizing them using a set of predefined possible targets and polarities has been studied under the name of Aspect Based Sentiment Analysis (ABSA) and popularized by the SEMEVAL campaigns (Pontiki et al., 2016). They defined a set of tasks including sentence-level prediction. *Aspect Category Detection* consists in finding the target of an opinion from a set of possible entities; *Opinion Target Expression* is a sequence tagging problem where the goal is to find the word indicating this entity; and *Sentiment Polarity* recognition is a classification task where the predictor has to determine whether the underlying opinion is positive, negative or neutral. Such problems have also been extended at the text level (*text-level ABSA*) where the participants were asked to predict a set of tuples (Entity category, Polarity level) summarizing the opinions contained in a review. In this work we adapt these tasks to a recently released fine-grained multimodal opinion mining corpus and study a category of hierarchical neural architecture able to jointly perform *token*-level, *sentence*-level and review-level predictions. In the next sections, we present the data available and the definition of the different tasks.

## 3 Data description and model

This work relies on a set of fine and coarse grained opinion annotations gathered for the Persuasive Opinion Multimedia (POM) corpus presented in Garcia et al. (2019). The dataset is composed of 1000 videos carrying a strong opinion content: in each video, a single speaker in frontal view makes a critique of a movie that he/she has watched. The corpus contains 372 unique speakers and 600 unique movie titles. The opinion of each speaker has been annotated at 3 levels of granularity as shown in Figure 1.

At the finest (*Token*) level, the annotators indicated for each token whether it is responsible for the understanding of the polarity of the sentence and whether it describes the target of an opinion. On top of this, a span-level annotation contains a categorization of both the target and the polarity of the underlying opinion in a set of predefined possible target *entities* and polarity *valences*. At the review level (or *text*-level since the annotations are aligned with the tokens of the transcript), an overall score describes the attitude of the reviewer about the movie.

As Garcia et al. (2019) have shown that the boundaries of span-level annotations are unreliable, we relax the corresponding boundaries at the sentence level. This *sentence* granularity is in our data the intermediate level of annotation between the *token* and the *text*. In practice, these

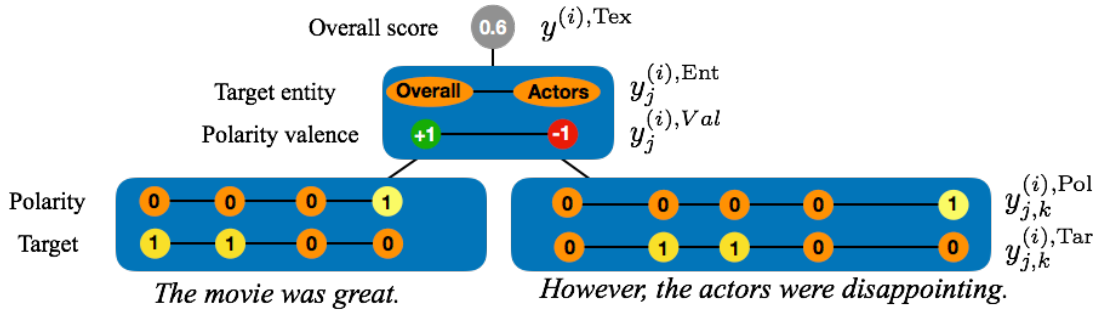


Figure 1: Structure of an annotated opinion

intermediate level labels can be modeled by tuples such as the one provided in the *text-level ABSA* SEMEVAL task which are given for each sentence in the dataset. In what follows, we will refer to the problem of predicting such information as the *sentence level*-prediction problem. Details concerning the determination of the sentence boundaries and the associated pre-processing of the data are given in the supplemental material.

The representation described above can be naturally converted into a mathematical representation: A review $\mathbf{x}^{(i)}$, $i \in \{1, \ldots, N\}$ is made of $S_i$ sentences each containing $W_{S_i}$ words. Thus the canonical feature representation of a review is the following $\mathbf{x}^{(i)} = \{\{x_{1,1}^{(i)}, \ldots, x_{1,W_{S_1}}^{(i)}\}, \ldots, \{x_{S_i,1}^{(i)}, \ldots, x_{S_i,W_{S_i}}^{(i)}\}\}$, where each $x$ is the feature representation of a spoken word corresponding to the concatenation of a textual, audio and video feature representation. It has been shown in (Zadeh et al., 2018a, 2016a,b) that whereas the textual modality carries the most information, taking into account video and audio modalities is mandatory to obtain state of the art results on sentiment analysis problems. Based on this input description, the learning task consists in finding a parameterized function $g_\theta : \mathcal{X} \to \mathcal{Y}$ that predicts various components of an opinion $\mathbf{y} \in \mathcal{Y}$ based on an input review $\mathbf{x} \in \mathcal{X}$. The parameters of such a function are obtained by minimizing an empirical risk:

$$\hat{\theta} = \min_\theta \sum_{i=1}^{N} l(g_\theta(\mathbf{x}^{(i)}), \mathbf{y}^{(i)}), \qquad (1)$$

where $l$ is a non-negative loss function penalizing wrong predictions. In general the loss $l$ is chosen as a surrogate of the evaluation metric whose purpose is to measure the similarity between the predictions and the true labels. In the case of complex objects such as opinions, there is no natural metric for measuring such proximity and we rely instead on distances defined on substructures of the opinion model. To introduce these distances, we first decompose the label-structures following the model previously described:

- *Token-level labels* are represented by a sequence of 2-dimensional binary label vectors $y_{j,k}^{(i),\text{Tok}} = \begin{pmatrix} y_{j,k}^{(i),\text{Pol}} \\ y_{j,k}^{(i),\text{Tar}} \end{pmatrix}$ where $y_{j,k}^{(i),\text{Pol}}$ and $y_{j,k}^{(i),\text{Tar}}$ are some binary variables indicating respectively whether the $k^{\text{th}}$ word of the sentence $j$ in review $i$ is a word indicating the polarity of an opinion , and the target of an opinion.

- *Sentence*-level labels carry 2 pieces of information: (1) the categorization of the target *entities* mentioned in an opinion expressed is represented by an $E$ dimensional binary vector $y_j^{(i),\text{Ent}}$ where each component encodes the presence of an entity among $E$ possible values; and (2) the polarity of the opinions contained in the sentence are represented by a 4-dimensional one-hot vector $y_j^{(i),\text{Val}}$ encoding the possible *valences*: *Positive*, *Negative*, *Neutral/Mixed* and *None*. Thus the sentence level label $y_j^{(i),\text{Sent}}$ is the concatenation of the two representations presented above: $y_j^{(i),\text{Sent}} = \begin{pmatrix} y_j^{(i),\text{Ent}} \\ y_j^{(i),\text{Val}} \end{pmatrix}$

- *Text*-level labels are composed of a single continuous score obtained for each review $y^{(i),Tex}$ summarizing the overall rating given by the reviewer to the movie described.

Based on these representations, we define a set of losses, $l^{(\text{Tok})}, l^{(\text{Sent})}, l^{(\text{Tex})}$ dedicated to measuring the similarity of each substructure prediction,

$\hat{\mathbf{y}}^{(\text{Tok})}, \hat{\mathbf{y}}^{(\text{Sent})}, \hat{\mathbf{y}}^{(\text{Tex})}$ with the ground-truth. In the case of binary variables and in the absence of prior preference between targets and polarities, we use the negative log-likelihood for each variable. Each task loss is then defined as the average of the negative log-likelihood computed on the variables that compose it. For continuous variables, we use the mean squared error as the task loss. Consequently the losses to minimize can be expressed as:

$$l^{(Tok)}(\mathbf{y}^{\text{Tok}}, \hat{\mathbf{y}}^{\text{Tok}}) = -\frac{1}{2}\sum_i \big(\big(\mathbf{y}_i^{Pol}\log(\hat{\mathbf{y}}_i^{Pol}) + \mathbf{y}_i^{Tar}\log(\hat{\mathbf{y}}_i{}^{Tar})\big)\big),$$

$$l^{(Sent)}(\mathbf{y}^{Sent}, \hat{\mathbf{y}}^{Sent}) = -\frac{1}{2}\sum_i \big(\mathbf{y}_i^{\text{Ent}}\log(\hat{\mathbf{y}}_i^{\text{Ent}}) + \mathbf{y}_i^{\text{Val}}\log(\hat{\mathbf{y}}_i^{\text{Val}})\big),$$

$$l^{(Tex)}(y^{\text{Tex}}, \hat{y}^{\text{Tex}}) = (y^{\text{Tex}} - \hat{y}^{\text{Tex}})^2,$$

Following previous works on multi-task learning (Argyriou et al., 2007; Ruder, 2017), we argue that optimizing simultaneously the risks derived from these losses should improve the results, compared to the case where they are treated separately, due to the knowledge transferred across tasks. In the multi-task setting, the loss $l$ derived from a set of task losses $l^{(t)}$, is a convex combination of these different task losses. Here the tasks corresponds to each granularity level: $t \in \text{Tasks} = \{\textit{Tok}, \textit{Sent}, \textit{Tex}\}$ weighted according to a set of task weights $\lambda_t$ :

$$l(\mathbf{y}, \hat{\mathbf{y}}) = \frac{\sum_{t\in\text{Tasks}} \lambda_t l^{(t)}(\mathbf{y}^t, \hat{\mathbf{y}}^t)}{\sum_{t\in\text{Tasks}} \lambda_t}, \ \forall \lambda_t \geq 0. \quad (2)$$

Optimizing this type of objectives in the case of hierarchical deep net predictors requires building some strategy in order to train the different parts of the model: the low level parts as well as the abstract ones. We discuss such an issue in the next section.

## 4 Learning strategies for multitask objectives

The main concern when optimizing objectives of the form of Equation 2 comes from the variable difficulty in optimizing the different objectives $l^{(t)}$. Previous works (Sanh et al., 2018) have shown that a careful choice of the order in which they are introduced is a key ingredient to correctly train deep hierarchical models. In the case of hierarchical labels, a natural hierarchy in the prediction complexity is given by the problem. In the task at hand, coarse grained labels are predicted by taking advantage of the information coming from predicting fine grained ones. The model processes the text by recursively merging and selecting the information in order to build an abstract representation of the review. In Experiment 1 we show that incorporating these fine grained labels into the learning process is necessary to obtain competitive results from the resulting predictors. In order to gradually guide the model from easy tasks to harder ones, we parameterize each $\lambda_t$ as a function of the number of epochs of the form $\lambda_t^{(n_{\text{epoch}})} = \lambda_{\max}\frac{\exp\left((n_{\text{epoch}} - Ns_t)/\sigma\right)}{1+\exp\left((n_{\text{epoch}} - Ns_t)/\sigma\right)}$ where $Ns_t$ is a parameter devoted to task $t$ controlling the number of epochs after which the weight switches to $\lambda_{\max}$ and $\sigma$ is a parameter controlling the slope of the transition. We construct 4 strategies relying on smooth transitions from a low state $\lambda_t^{(0)} = 0$ to a high state $\lambda_t^{(Ns_t)} = \lambda_t^{\max}$ of each task weight varying with the number of epochs. The different strategies described below are illustrated in the supplemental material.

- Strategy 1 (S1) consists in optimizing the different objectives one at a time from the easiest to the hardest. It consists in first moving vector $(\lambda_{\textit{Token}}, \lambda_{\textit{Sentence}}, \lambda_{\textit{Text}})^T$ values from $(1,0,0)^T$ to $(0,1,0)^T$ and then finally to $(0,0,1)^T$. The underlying idea is that the low level labels are only useful as an initialization point for higher level ones.

- Strategy 2 (S2) consists in adding sequentially the different objectives to each other from the easiest to the hardest. It goes from a word only loss $(\lambda_{\textit{Token}}, \lambda_{\textit{Sentence}}, \lambda_{\textit{Text}})^T = (\lambda_{\textit{Token}}^{(N)}, 0, 0)^T$ and then adds the intermediate objectives by setting $\lambda_{\textit{Sentence}}$ to $\lambda_{\textit{Sentence}}^{(N)}$ and then $\lambda_{\textit{Text}}$ to $\lambda_{\textit{Text}}^{(N)}$. This strategy relies on the idea that keeping a supervision on low level labels has a regularizing effect on high level ones. Note that this strategy and the two following require a choice of the stationary weight values $\lambda_{\textit{Token}}^{(N)}, \lambda_{\textit{Sentence}}^{(N)}, \lambda_{\textit{Text}}^{(N)}$.

- Strategy 3 (S3) is similar to (S2) except that the *sentence* and *text* weights are simultaneously increased. This strategy and the following one are introduced to test whether the order in which the tasks are introduced has some importance on the final scores.

- Strategy 4 (S4) is also similar to (S2) except that *text*-level supervision is introduced before the *sentence*-level one. This strategy uses the intermediate level labels as a way to regularize the video level model that would have been learned directly after the *token*-level supervision

These strategies can be implemented in any stochastic gradient training procedure of objectives (Equation 2) since it only requires modifying the values of the weight at the end of each epoch. In the next section, we design a neural architecture that jointly predicts opinions at the three different levels, *i.e.* the *token*, *sentence* and *text* levels, and discuss how to optimize multitask objectives built on top of opinion-based output representations.

## 5 Architecture

Before digging into the model description, we introduce the set of hidden variables $h^{(i),\text{Tex}}, h_j^{(i),Sent}, h_{j,k}^{(i),\text{Tok}}$ corresponding to the unconstrained scores used to predict the outputs: $\hat{y}^{(i),\text{Tex}} = \sigma^{\text{Tex}}(W^{\text{Tex}} h^{(i),\text{Tex}} + b^{\text{Tex}})$, $\hat{y}_j^{(i),\text{Sent}} = \sigma^{\text{Sent}}(W^{\text{Sent}} h_j^{(i)^{\text{Sent}}} + b^{\text{Sent}})$, $\hat{y}_{j,k}^{(i),\text{Tok}} = \sigma^{\text{Tok}}(W^{\text{Tok}} h_{j,k}^{(i),\text{Tok}} + b^{\text{Tok}})$, where the $W$ and $b$ are some parameters learned from data and the $\sigma$ are some fixed almost everywhere differentiable functions ensuring that the outputs "match" the inputs of the loss function. In the case of binary variables for example, it is chosen as the sigmoid function $\sigma(x) = \exp(x)/(1+\exp(x))$. From a general perspective, a hierarchical opinion predictor is composed of 3 functions $g^{\text{Tex}}, g^{\text{Sent}}, g^{\text{Tok}}$ encoding the dependency across the levels:

$$
\begin{aligned}
h_{j,k}^{(i),\text{Tok}} &= g_{\theta^{\text{Tok}}}^{\text{Tok}}(x_{j,:}^{(i),\text{Tok}}), \\
h_j^{(i)^{\text{Sent}}} &= g_{\theta^{\text{Sent}}}^{\text{Sent}}(h_{j,:}^{(i)^{\text{Tok}}}), \\
h^{(i)^{\text{Tex}}} &= g_{\theta^{\text{Tex}}}^{\text{Tex}}(h_{:}^{(i)^{\text{Sent}}}).
\end{aligned}
$$

In this setting, low level hidden representations are shared with higher level ones. A large body of work has focused on the design of the $g$ functions in the case of multimodal inputs. In this work we exploit state of the art sequence encoders to build our hidden representations that we detail below. The mathematical expression of the models and a more in depth description are provided in the supplemental material.

- Bidirectional Gated Recurrent Units (GRU) (Cho et al., 2014) especially when coupled with a self attention mechanism have been shown to provide state of the art results on tasks implying the encoding or decoding of a sentence in or from a fixed size representation. Such a problem is encountered in automatic machine translation (Luong et al., 2015), automatic summarization (Nallapati et al., 2017) or image captioning and visual question answering (Anderson et al., 2018). We experiment with both models mixing the 3 concatenated input feature modalities (BiGRU model in Experiment 1) and a model carrying 3 independent BiGRU with a hidden state per modality (Ind BiGRU models).

- The Multi-attention Recurrent Network (MARN) proposed in (Zadeh et al., 2018a) extends the traditional Long Short Term Memory (LSTM) sequential model by both storing a view specific dynamic (similar to the LSTM one) and by taking into account cross-view dynamics computed from the signal of the other modalities. In the original paper, this cross-view dynamic is computed using a multi-attention bloc containing a set of weights for each modality used to mix them in a joint hidden representation. Such a network can model complex dynamics but does not embed a mechanism dedicated to encoding very long-range dependencies.

- Memory Fusion Networks (MFN) are a second family of multi-view sequential models built upon a set of LSTM per modality feeding a joint delta memory. This architecture has been designed to carry some information in the memory even with very long sequences due to the choice of a complex retain / forget mechanism.

The 3 models described previously build a hidden representation of the data contained in each sequence. The transfer from one level of the hierarchy to the next coarser one requires building a fixed length representation summarizing the sequence. Note that in the case of the MARN and the MFN, the model directly creates such a representation. We present the strategies that we deployed to pool these representations in the case of the BiGRU sequential layer.

- Last state representation: Sequential models build their inner state based on observations

from the past. One can thus naturally use the hidden state computed at the last observation of a sequence to represent the entire sequence. In our experiments, this is the representation chosen for the BiGRU and Ind BiGRU models.

- Attention based sequence summarization: Another technique consists in computing a weighted sum of the hidden states of the sequence. The attention weights can be learned from data to focus on the important parts of the sequence only and avoid building too complex inner representations. An example of such a technique successfully applied to the task of text classification based on 3 levels of representation can be found in (Yang et al., 2016). In our experiments, we implemented the attention model for predicting only the *Sentence*-level labels (model Ind BiGRU + att Sent) and the *Sentence* and *Text*-level labels by sharing a common representation (Ind BiGRU + att model).

All the resulting architectures extend the existing hierarchical models by enabling the fusion of multimodal information at different granularity levels while maintaining the ability to introduce some supervision at any level.

## 6 Experiments

In this section we propose 3 sets of experiments that show the superiority of our model over existing approaches with respect to the difficulties highlighted in the introduction, and explore the question of the best way to train hierarchical models on multimodal opinion data.

All the results presented below have been obtained on the recently released fine grained annotated POM dataset (Garcia et al., 2019). The input features are computed using the CMU-Multimodal SDK: We represented each word by the concatenation of the 3 feature modalities. The textual features are chosen as the 300-dimensional pre-trained Glove embeddings (Pennington et al., 2014) (not updated during training). The acoustic and visual features have been obtained by averaging the descriptors computed following (Park et al., 2014) during the time of pronunciation of each spoken word. These features include MFCC and pitch descriptors for the audio signals. For the video descriptors, posture, head and gaze movement are taken into account. As far as the output representations are concerned, we merely re-scaled the *Text*-level polarity labels in the [0,1] range.

The results are reported in terms of mean average error (MAE) for the continuous labels and micro F1 score $\mu F1$ for binary labels. We used the provided train, val and test set and describe for each experiment the training procedure and displayed values below. More detail concerning the preprocessings and architectures can be found in the supplemental material.

### 6.1 Experiment 1: Which architecture provides the best results on the task of fine grained opinion polarity prediction?

In this first section, we describe our protocol to select an architecture devoted to performing fine grained multimodal opinion prediction. In order to focus on a restricted set of possible models, we only treat the polarity prediction problem in this section and selected the architectures that provided the best review-level scores (*i.e.* with lowest mean average prediction error). Taking into account the entity categories would only bring an additional level of complexity that is not necessary in this first model selection phase. Building upon previous works (Zadeh et al., 2018b), we use the MFN model as our *sentence*-level sequential model since it has been shown to provide state of the art results on *text*-level prediction problems on the POM dataset. For the *token*-level model, we test different state of the art models able to take advantage of the multimodal information. Our architecture is built upon the *token*-level encoders presented in section 5: the MFN, MARN and independent BiGRUs. Our baseline is computed similarly to Zadeh et al. (2018a): we represent each sentence by taking the average of the feature representation of the Tokens composing it. The best results reported were obtained after a random search on the parameters and presented in Table 1. In the top row, we report results obtained when only using the *text*-level labels to train the entire network. The baseline consisting in representing each sentence by the average of its tokens representation strongly outperforms all the other results. This is due to the moderate size of the training set (600 videos) which is not enough to learn meaningful fine grained representations. In the second part, we introduce some supervision at all levels and found that a choice of $\lambda_{Tok} = 0.05,\ \lambda_{Sent} = 0.5,\ \lambda_{Tex} = 1$ being respec-

| | $\lambda_{Tok} = \lambda_{Sent} = 0$: no fine grained supervision | | | | | | |
|---|---|---|---|---|---|---|---|
| MAE *Text* | 0.35 | 0.40 | 0.40 | 0.38 | 0.29 | 0.32 | **0.17** |
| | Supervision at the token, sentence and review levels | | | | | | |
| Metric \ Model | BiGRU | Ind BiGRU | Ind BiGRU + att Sent | Ind BiGRU + att | MARN | MFN | Av Emb |
| $\mu F1$ *Tokens* | 0.90 | **0.93** | **0.93** | **0.93** | 0.90 | 0.89 | X |
| $\mu F1$ *Sentence* | 0.68 | 0.72 | **0.75** | **0.75** | 0.52 | 0.47 | X |
| MAE *Text* | 0.16 | 0.15 | 0.15 | **0.14** | 0.35 | 0.37 | X |

Table 1: Scores on sentiment label

tively the *token*, *sentence* and *text* weights provides the best *text*-level results. This combination reflects the fact that the main objective (*text*-level) should receive the highest weight but low level ones also add some useful side supervision. Despite the ability of MARN and MFN to learn complex representations, the simpler BiGRU-based Token encoder retrieves the best results at all the levels and provides more than 12% of relative improvement over the Average Embedding based model at the video level. This behavior reveals that the high complexity of MARN and MFN makes them hard to train in the context of hierarchical models with limited data leading to suboptimal performance against simpler ones such as BiGRU. We fix the best architecture obtained in this experiment displayed in Figure 2 and reuse it in the subsequent experiments.

### 6.2 Experiment 2: What is the best strategy to take into account multiple levels of opinion information?

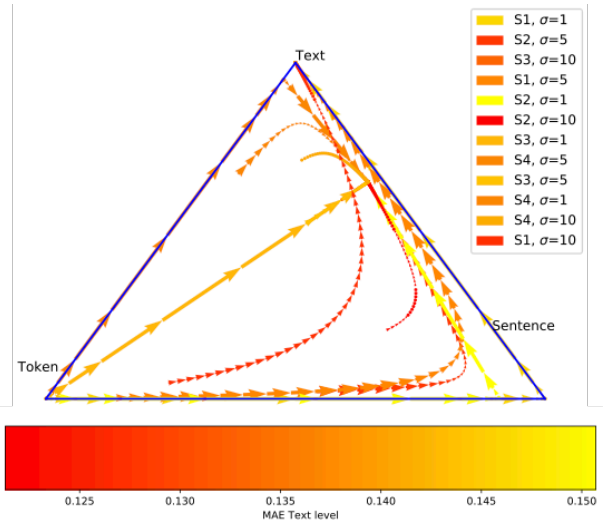


Figure 3: Path of the weight vector in the simplex triangle for the different tested strategies

Motivated by the issues concerning the training of multitask losses raised in Section 4, we implemented the 4 strategies described and chose the final stationary values as the best one obtained in Experiment 1: $(\lambda^{(N)}_{Token}, \lambda^{(N)}_{Sentence}, \lambda^{(N)}_{Text}) = (0.05, 0.5, 1)$ Note that each strategy corresponds to a path of the vector $(\lambda_{Tok}, \lambda_{Sent}, \lambda_{Tex})^T / \sum_t \lambda_t$ in the 3 dimensional simplex. We represent the 3 strategies tested in the Figure 3 corresponding to the projection of the weight vector onto the hyperplane containing the simplex.

The best paths for optimizing the *text*-level objectives are the one that smoothly move from a combination of *sentence* and *token*-level objectives to a *text* oriented one. The path in the simplex seems to be more important than the nature of the strategy since S1 and S2 reach the same *text*-level MAE score while working differently. It also appears than an objective with low $\sigma$[1] values corresponding to harder transitions tends to obtain lower scores than smooth transition based strategies. All the strategies are displayed as a function of the number of epochs in the supplemental material. In this last section we deal with the issue of the joint prediction of entities and polarities.

### 6.3 Experiment 3: Is it better to jointly predict opinions and entities ?

In this section, we introduce the problem of predicting the entities of the movie on which the predictions are expressed, as well as the tokens that mention them. This task is harder than the previously studied polarity prediction task due to (1) the problem of label imbalance appearing in the label distribution reported in the Table 3 and (2) the diversity of the vocabulary incurred when dealing with many entities. However since the presence of a polarity implies the presence of at least one entity, we expect that a joint prediction will perform better than an entitiy-based predictor only. Table 2 contains the results obtained with the architecture described in Figure 2 on the task of joint polarity

[1] described in Section 4

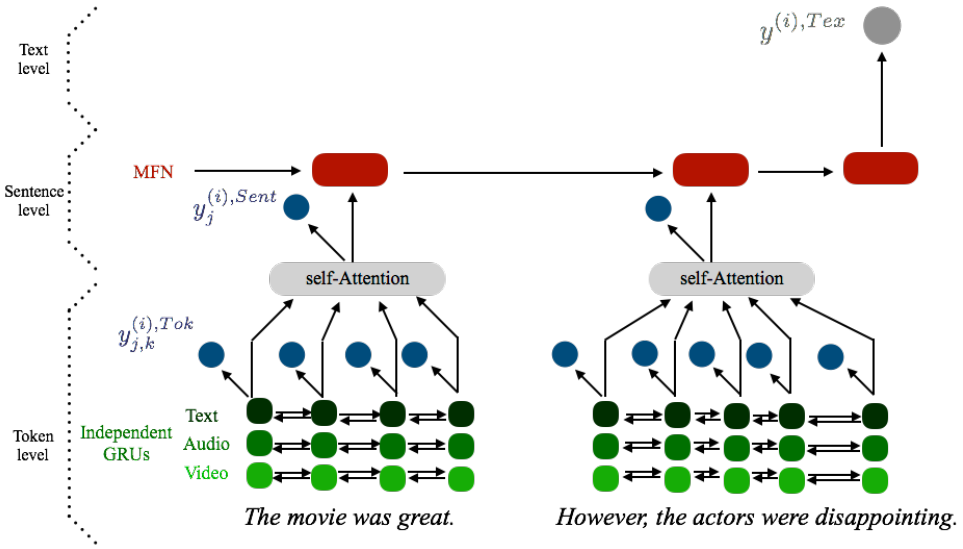


Figure 2: Best architecture selected during the Experiment 1

and entity prediction as well as the results obtained when dealing with these tasks independently.

Using either the joint or the independent models provides the same results on the polarity prediction problems at the *token* and *sentence*-level. The reason is that the polarity prediction problem is easier and relying on the entities prediction would only introduce some noise in the prediction. We

| | Polarity labels | Entity labels | Polarity + entities |
|---|---|---|---|
| F1 polarity tokens | 0.93 | X | 0.93 |
| F1 polarity valence | 0.75 | X | 0.75 |
| F1 entities tokens | X | 0.97 | 0.97 |
| F1 entities Entities | X | Table 3 | Table 3 |
| MAE score review level | 0.14 | 0.38 | 0.14 |

Table 2: Joint and independent prediction of entities and polarities

detail the case of *Entities* in the Table 3 and present the results obtained for the most common entity categories (among 11). As expected, the entity prediction tasks benefits from the polarity information on most of the categories except for the *Vision and special effects*. A 5% of relative improvement can be noted on the two most present *Entities*: *Overall* and *Screenplay*.

| | Entity | Entity + Polarity | Value Count |
|---|---|---|---|
| Overall | 0.71 | **0.73** | 1985 |
| Actors | **0.65** | **0.65** | 493 |
| Screenplay | 0.60 | **0.63** | 246 |
| Atmosphere and mood | 0.62 | **0.64** | 151 |
| Vision and special effects | **0.62** | 0.58 | 154 |

Table 3: F1 score per label for the top entity categories annotated at the sentence level (mean score averaged over 7 runs), value counts are provided on the test set.

## 7 Conclusion

The proposed framework enables the joint prediction of the different components of an opinion based on a hierarchical neural network. The resulting models can be fully or partially supervised and take advantage of the information provided by different views of the opinions. We have experimentally shown that a good learning strategy should first rely on the easy tasks (*i.e.* for which the labels do not require a complex transformation of the inputs) and then move to more abstract tasks by benefiting from the low level knowledge. Future work will explore the use of *structured output learning* methods dedicated to the opinion structure.

## 8 Acknowledgements

We would like to thanks Thibaud Besson and the whole french Cognitive Systems team of IBM for supporting our research with the server IBM Power AC922.

## References

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6077–6086.

Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. 2007. Multi-task feature learning. In *Advances in neural information processing systems*, pages 41–48.

Salvatore Attardo, Jodi Eisterhold, Jennifer Hay, and Isabella Poggi. 2003. Multimodal markers of irony and sarcasm. *Humor*, 16(2):243–260.

A. Ben Youssef, C. Clavel, and S. Essid. 2019. Early detection of user engagement breakdown in spontaneous human-humanoid interaction. *IEEE Transactions on Affective Computing*, pages 1–1.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*.

Chloe Clavel and Zoraida Callejas. 2016. Sentiment analysis: from opinion mining to human-agent interaction. *IEEE Transactions on affective computing*, 7(1):74–93.

Alexandre Garcia, Slim Essid, Florence D'alché-Buc, and Chloé Clavel. 2019. A multimodal movie review corpus for fine-grained opinion mining. *arxiv Preprint: arXiv:1902.10102*.

Leo Hemamou, Ghazi Felhi, Vincent Vandenbussche, Jean-Claude Martin, and Chloé Clavel. 2018. Hirenet: a hierarchical attention model for the automatic analysis of asynchronous video job interviews. In *AAAI 2019*. ACM.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Caroline Langlet and Chloé Clavel. 2015a. Adapting sentiment analysis to face-to-face human-agent interactions: from the detection to the evaluation issues. In *Affective Computing and Intelligent Interaction (ACII), 2015 International Conference on*, pages 14–20. IEEE.

Caroline Langlet and Chloé Clavel. 2015b. Improving social relationships in face-to-face human-agent interactions: when the agent wants to know user's likes and dislikes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1064–1073.

Caroline Langlet and Chloé Clavel. 2016. Grounding the detection of the user's likes and dislikes on the topic structure of human-agent interactions. *Knowledge-Based Systems*, 106:116–124.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pages 142–150. Association for Computational Linguistics.

James R Martin and Peter R White. 2013. *The language of evaluation*, volume 2. Springer.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Behnaz Nojavanasghari, Deepak Gopinath, Jayanth Koushik, Tadas Baltrušaitis, and Louis-Philippe Morency. 2016. Deep multimodal fusion for persuasiveness prediction. In *Proceedings of the 18th ACM International Conference on Multimodal Interaction*, pages 284–288. ACM.

Sunghyun Park, Han Suk Shim, Moitreya Chatterjee, Kenji Sagae, and Louis-Philippe Morency. 2014. Computational analysis of persuasiveness in social multimedia: A novel dataset and multimodal prediction approach. In *Proceedings of the 16th International Conference on Multimodal Interaction*, pages 50–57. ACM.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, AL-Smadi Mohammad, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphée De Clercq, et al. 2016. Semeval-2016 task 5: Aspect based sentiment analysis. In *Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016)*, pages 19–30.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

Victor Sanh, Thomas Wolf, and Sebastian Ruder. 2018. A hierarchical multi-task approach for learning embeddings from semantic tasks. *The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI 2019)*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39(2-3):165–210.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

A Zadeh, PP Liang, S Poria, P Vij, E Cambria, and LP Morency. 2018a. Multi-attention recurrent network for human communication comprehension. In *AAAI*.

Amir Zadeh, Paul Pu Liang, Navonil Mazumder, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. 2018b. Memory fusion network for multi-view sequential learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Amir Zadeh, Rowan Zellers, Eli Pincus, and Louis-Philippe Morency. 2016a. Mosi: multimodal corpus of sentiment intensity and subjectivity analysis in online opinion videos. *arXiv preprint arXiv:1606.06259*.

Amir Zadeh, Rowan Zellers, Eli Pincus, and Louis-Philippe Morency. 2016b. Multimodal sentiment intensity analysis in videos: Facial gestures and verbal messages. *IEEE Intelligent Systems*, 31(6):82–88.

# Shallow Domain Adaptive Embeddings for Sentiment Analysis

Prathusha K Sarma, Yingyu Liang and William A Sethares

University of Wisconsin-Madison
{kameswarasar,sethares}@wisc.edu,
yliang@cs.wisc.edu

## Abstract

This paper proposes a way to improve the performance of existing algorithms for text classification in domains with strong language semantics. We propose a domain adaptation layer learns weights to combine a generic and a domain specific (DS) word embedding into a domain adapted (DA) embedding. The DA word embeddings are then used as inputs to a generic encoder + classifier framework to perform a downstream task such as classification. This adaptation layer is particularly suited to datasets that are modest in size, and which are, therefore, not ideal candidates for (re)training a deep neural network architecture. Results on binary and multi-class classification tasks using popular encoder architectures, including current state-of-the-art methods (with and without the shallow adaptation layer) show the effectiveness of the proposed approach.

## 1 Introduction

Domain Adaptation (DA) algorithms are becoming increasingly relevant in addressing issues related to i) lack of availability of training data in domains of interest and in ii) exploiting domain idiosyncrasies to improve performance of out-of-domain algorithms. While some state of the art DA algorithms focus on improving on a downstream task, such as classification in bilingual or crosslingual framework (Hangya et al., 2018) (Liu et al., 2018), others (Zeng et al., 2018), (Shang et al., 2018) focus on tackling DA at the word level for various downstream applications such as classification, tagging etc.

Particularly, work by (An et al., 2018) and (K Sarma et al., 2018) address the issue of context-based domain adaptation within a single language. The authors of both works argue that word context causes significant changes, especially in document level sentiment associations. The idea is illustrated by providing examples of words such as 'kill,' that has a more positive sentiment in a document describing video games than when used in a news article. Similarly, words used in social media and medical domains tend to be atypical and idiosyncratic as compared to generic language use.

DA algorithms and related transfer learning algorithms are often successfully used to perform sentiment analysis, where given a document, the sentiment label for the document is to be determined. Since adaptation can be viewed as a shift in the position of words within the embeddings spaces across the domains of interest, shifts in the spatial position of the words can be measured to verify that the algorithm is indeed capturing domain knowledge. Once it is established that a given DA method is capturing domain semantics at the word level, the DA word embeddings can then be used as input to an encoder+classification layer to perform sentiment analysis.

This paper applies a generic adaptation framework to tasks that make use of domain semantics. Contributions of this paper are fourfold. i) First, we propose a generic domain adaptation layer that can be interfaced with any neural network block with the aim of improving performance on a downstream sentiment classification task. ii) Second, we measure the significance of the shift in word position when represented in a generic embedding space and when represented in the adapted embedding space. The strategy is to construct DA word embeddings as in (K Sarma et al., 2018) and make use of a seed lexicon as in (Hangya et al., 2018) and (Mikolov et al., 2013a). However, rather than use the seed lexicon to transform words from two different domains into a common space, we use the seed lexicon to verify that the DA word embeddings have indeed captured significant domain semantics. iii) We test



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5549–5558, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

our hypothesis on a recently introduced (Friedland et al., 2017)[1] data set of tweets from Liberal and Conservative Twitter users from the state of Wisconsin. As illustrated in (An et al., 2018), political discourse makes for an interesting setting to study domain adaptation. (iv) Via thorough experimentation we show that using a DA layer helps improve the performance of standard architectures on sentiment analysis tasks by $2-8\%$ on various binary and multi-class balanced and imbalanced data sets. We also, show that our DA architectures outperform sophisticated architectures such as BERT, LR-Bi-LSTM, Self-attention by $1-2\%$.

The rest of this paper is organized as follows, Section 2 discusses related work. Section 3 describes in detail, the proposed algorithm, Section 4 presents the experimental results and Section 5 concludes this work.

## 2 Related Work

This paper discusses domain adaptation techniques and the applicability of existing algorithms to diverse downstream tasks. Central themes of this paper tie into word level domain semantics, while being related to the overall objective of domain adaptation.

Recent work such as SEMAXIS (An et al., 2018) investigates the use of word level domain semantics for applications beyond sentiment analysis. The authors introduce the concept of a semantic axis based on word antonym pairs to capture semantic differences across corpora. Similarly, work by (Hamilton et al., 2016) captures domain semantics in the form of sentiment lexicons via graph propagation. While both these lexical based approaches are similar to the ideas of this paper, a major difference is that like (K Sarma et al., 2018), we do not make use of any predefined domain specific lexicons to capture domain semantics. Our idea is to use word occurrences and contexts to provide a raw estimate of the domain semantics. Using generic embedding spaces as baselines, adaptation is performed by projecting generic embeddings into a learned 'adaptation' space.

Typical downstream applications such as cross lingual and/or multi-domain sentiment classification, using algorithms proposed by (Hangya et al., 2018), (Liu et al., 2018), make use of DNNs with RNN blocks such as BiLSTMs to learn both generic and domain specific representations. Particularly, work focused on multi-domain sentiment classification as in (Liu et al., 2016), (Nam and Han, 2016), proposes building neural architectures for each domain of interest, in addition to shared representation layers across all domains. While these techniques are effective, they are not ideal in domains with limited data.

On the other hand work such as ELMo (Peters et al., 2018) and BERT (Devlin et al., 2018) propose deeply connected layers to learn sentence embeddings by exploiting bi-directional contexts. While both methods have achieved tremendous success in producing word (ELMo) and sentence (BERT) level encodings that perform well in several disparate NLP tasks such as question-answer solving, paraphrasing, POS tagging, sentiment analysis etc, these models are computationally expensive and require large amounts of training data. Particularly when used in a transfer learning setting, both algorithms assume that a large amount of data is present in the source as well as the target domains. In contrast, our proposed adaptation layer is particularly well suited in applications with limited data in the target domain.

Our proposed algorithms depart from these approaches by capturing domain semantics through shallow layers for use with generic encoder architectures. Since some of the most successful algorithms in text classification (Kim, 2014) and sentence embeddings (Conneau et al., 2017) make use of CNN and BiLSTM building blocks, we suggest a generic adaptation framework that can be interfaced with these standard neural network blocks to improve performance on downstream tasks, particularly on small sized data sets.

## 3 Shallow Domain Adaptation

This section introduces a 'shallow' adaptation layer[2], to be interfaced with a neural network based sentence encoding layer, followed by a classification layer. Figure 1 describes a generic framework of the proposed model. Brief descriptions of the three layers follow.

**Adaptation Layer:** Generic word embeddings from GloVe (Pennington et al., 2014) or word2vec (Mikolov et al., 2013b) are combined with an LSA-based (Deerwester et al., 1990) Do-

[1] https://github.com/naacl18sublong/Friedland

[2] Codes for shallow adaptation will be made available upon acceptance of paper for publication.

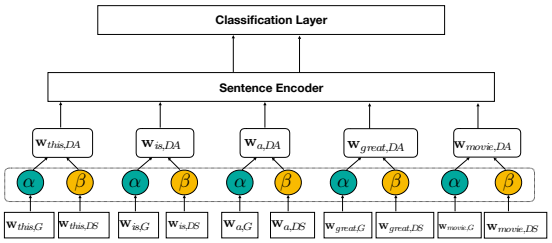


Figure 1: This figure illustrates the three part neural network model. The first part comprises of an adaptation layer, second, a generic sentence encoder and the lastly a classification layer. Inputs to the shallow adaptation layer are the generic and the domain specific (DS) word embeddings. Output of the adaptation layer is the domain adapted (DA) word embedding.

main Specific (DS) word embedding using KCCA as proposed in (K Sarma et al., 2018). Once the KCCA projections $(\bar{\mathbf{w}}_{i,G}, \bar{\mathbf{w}}_{i,DS})$ for the generic and DS word embeddings are obtained for a given word $i$, weights $\alpha$ and $\beta$ are learned such that the domain adapted word embedding is $\bar{\mathbf{w}}_{i,DA} = \alpha\bar{\mathbf{w}}_{i,G} + \beta\bar{\mathbf{w}}_{i,DS}$. Weights $\alpha$ and $\beta$ are learned by a single CNN layer. Since this layer learns only the weights used to obtain the DA embeddings, we call this a 'shallow' adaptation layer.

The output of this layer (i.e., the DA embeddings), is used to initialize a generic sentence encoder. Output from the sentence encoder is sent to a classification layer and the classification error is back propagated to update weights of the adaptation layer. By keeping the sentence encoder fixed, (i.e., the parameters of the encoder are not updated during back propagation), this method can be generalized for use with any pre-trained sentence encoder. This is advantageous, particularly in data sets that are too small to be trained end-to-end. Another advantage is that adaptation is not performed on fully connected layers that need to be trained end-to-end. This considerably reduces the number of parameters that need to be learned.

Figure 2 illustrates the CNN encoder architecture and the Adaptation Layer. Input to the adaptation layer is a word embedding of dimension $2d$, where $d$ is the dimension of the generic and DS embeddings. For a given word, the generic and DS embeddings are concatenated and interleaved. A single layer CNN learns $\alpha$ and $\beta$ via a $2 \times 1$ kernel. The domain adapted word $\mathbf{w}_{i,DA}$ is then passed as input to the CNN encoder as shown in the figure. Note that we also use this framework with a BiLSTM sentence encoder as in (Conneau et al., 2017) and is illustrated in Figure 2. Output of the BiLSTM units is max-pooled to obtain the sentence encoding.

**Sentence Encoder:** CNN encoder as in (Kim, 2014) and BiLSTM encoder proposed in (Conneau et al., 2017) are used to encode sentences in this work. Since both models have been extensively discussed in literature we shall skip elaborating on the basic model architectures. We reproduced the basic encoder models for the experiments in Section 4 with exception of few hyper-parameters, since the choice of data sets in this work differs from that of (Kim, 2014) and (Conneau et al., 2017).

**Classification Layer:** This layer learns weights for multi-class classification using a soft-max function.

### 3.1 Evaluating KCCA for Domain Adaptation

In their work (K Sarma et al., 2018), the authors demonstrate the effectiveness of KCCA based embeddings via classification experiments. However, in order to verify that the word level adaptation performed by the KCCA step can capture relevant domain semantics, we perform the following experiment.

### 3.2 Experimental Setup

Start with a dataset where language use is polarized. For example, the language used in the tweets of liberals likely differs from the language used in the tweets of conservatives as they express opinions on key political issues such as government, immigration, border control etc. These two vocabularies (Liberal and Conservative) are encoded in DS embeddings which are each paired with a generic embedding, and then again mapped into a common DA space. The words in this common space that are most different are then identified, and compared to a ground truth/gold standard list of keywords which represent ideas or concepts on which the two groups of users are known to have the most polarized opinions. Comparing the gold standard list with the list derived from the DA embedding demonstrates the efficacy of the method. This experiment adopts the list of politically contentious words that appear in (Li et al., 2017).

From the tokenized tweets of the users, construct a vocabulary of words $V_{Lib}$ and $V_{Con}$ that represent the Liberal and Conservative tweets respectively. Domain specific (DS) word embed-

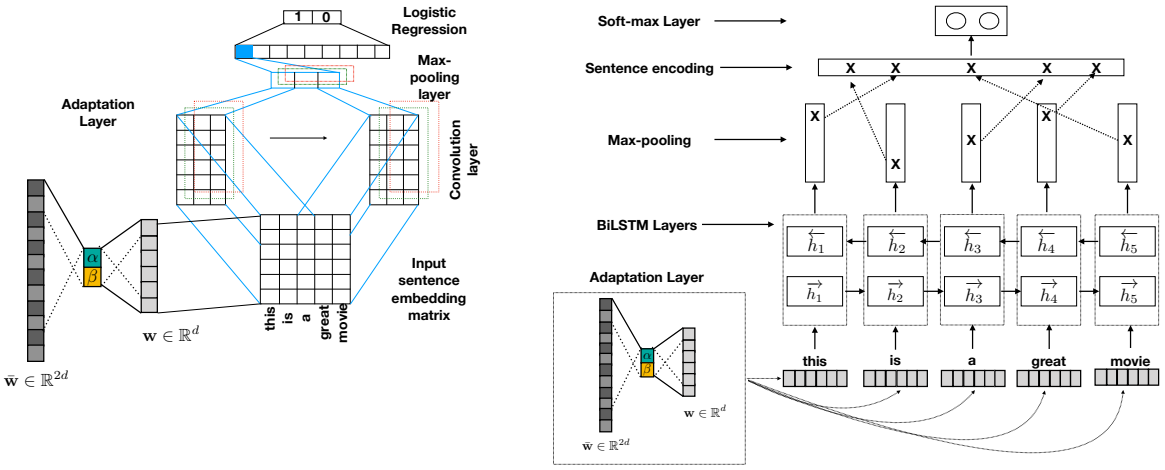


Figure 2: **Left:** This figure illustrates the adaptation layer, that takes as input a $2d$ dimensional word embedding containing the generic and DS KCCA projections and learns a $d$ dimensional DA word embedding. This DA word embedding is then input to the CNN encoder. The network can be optimized to learn only parameters $\alpha$ and $\beta$ or can be trained end to end to learn sentence embeddings and classifier weights as well. **Right:** This figure illustrates the adaptation layer, out of which, the DA word embedding is then input to the BiLSTM+max-pooling encoder. The entire network can be trained end-to-end to learn parameters $\alpha$ and $\beta$ in addition to the sentence embedding and classifier, or the network can be optimized to learn only the weights $\alpha$ and $\beta$ and weights of the classifier.

dings are first constructed from $V_{Lib}$ and $V_{Con}$. Then, KCCA projections are obtained for generic GloVe embeddings and the DS word embeddings for each user group, in order to construct DA word embeddings for Liberal $\hat{\mathbf{W}}_{DA(Lib)}$ and Conservative $\hat{\mathbf{W}}_{DA(Con)}$ users.

Next consider the set of common words in the two vocabularies, i.e $V_{common} = V_{Lib} \cap V_{Con}$. Perform a second KCCA to bring DA words in $V_{common}$ from both Liberal $V_{Lib}$ and Conservative $V_{Con}$ vocabularies into a common subspace. To measure the 'shift' $\psi$, calculate the $l_2$ distance for each word embedding $\hat{\mathbf{W}}_i \in V_{common}$, i.e

$$\psi = ||\hat{\mathbf{W}}_{i,DA(Lib)} - \hat{\mathbf{W}}_{i,DA(Con)}||_2 \quad \forall i \in V_{common} \tag{1}$$

Words with large values of $\psi$ are considered to have shifted the most between the domains of liberal and conservative users. Words are then ordered based on the magnitude of $\psi$, and cross referenced with words in the gold standard list.

In order to demonstrate that the KCCA can capture domain semantics when learning DA word embeddings, the following section presents an analysis against a random baseline.

### 3.3 Study of a random baseline

A random baseline is one that picks $k$ words from $V_{common}$, randomly, and reports these $k$ words as having shifted the most with respect to word use within tweets that correspond to Liberal and Conservative users. This random baseline follows a hypergeometric distribution, which is a discrete probability distribution that calculates the probability of obtaining $k$ successes (of a sample with a specific feature) from $n$ draws, from a finite population of $V$, without replacement, where exactly $K$ samples contain the specific feature.

For example, suppose $|V_{common}| = V$, and say the number of words in the gold standard list are $K$. Then the probability of the random baseline picking $k < K$ words out of $n$ words is

$$Pr(X = k) = \frac{\binom{K}{k}\binom{V-K}{n-k}}{\binom{V}{n}}. \tag{2}$$

To calculate these probabilities for the KCCA-DA word embeddings, we shall use the LibCon data set of Section 3.1. A gold standard list of 136 key topics/concepts important to Liberals and Conservatives is obtained from the study by (Li et al., 2017), and additional details about the LibCon data set and the gold standard list can be found in Section 4.1.

From the LibCon data set $|V_{common}| = V = 1573$. Out of these 1573 words, there are 74 words that are present in the gold standard list. Hence $K = 74$. After obtaining the DA word embeddings for words in $V_{common}$ and calculating the shift $\psi$, we take the top 200 words that shifted

| | | | | | | |
|---|---|---|---|---|---|---|
| action | cost | dream | help | media | **reform** | **women** |
| **amendment** | court | education | honor | need | **republican** | work |
| **attack** | **crisis** | fact | hope | order | right | world |
| budget | deal | **force** | income | pledge | risk | |
| burden | debate | **freedom** | information | **police** | rule | |
| business | **debt** | fund | **insurance** | poll | school | |
| candidate | decision | funding | **justice** | power | spending | |
| care | defense | future | **labor** | president | **state** | |
| class | deficit | generation | leader | problem | **truth** | |
| college | **democrat** | **government** | leadership | program | value | |
| **congress** | development | **governor** | **legislature** | protection | **violence** | |
| **control** | divide | health | **legislation** | race | wealth | |

Table 1: This table presents 74 words representing key political concepts common to Liberal and Conservative users on Twitter. Words in bold are the ones that 'shift' the most (by $l_2$ distance) in use between Liberal and Conservative users on Twitter.

most. From these words, at least 20 are present in the gold standard list.

The number of words-of-interest in a sample of 200 words forms a hypergeometric distribution with mean $\mu = n\frac{K}{V} \approx 9.4$, and standard deviation 2.7984.

Direct calculation shows that the probability of picking exactly 20 words from this subset is $Pr(X = 20) = 0.000346$. Similarly, the probability that 20 or more would have been chosen by chance is,

$$p = \sum_{k=20}^{200} Pr(X = k) \tag{3}$$

which, by numerical calculation, is $p = 0.000524$. Such a result is highly unlikely to occur by chance. Accordingly, the KCCA domain adapted word embeddings are indeed capturing something significant about the language usage in the domain.

Table 1 presents words from the gold standard list with words in bold face indicating words that shift the most across Liberal and Conservative domains.

## 4 Experimental Results

The discussion in Section 3.1 provides motivation for adopting the KCCA projections as a tool to calculate DA word embeddings. These DA word embeddings can be combined with standard neural network models to obtain sentence embeddings that can be used to improve performance on downstream tasks such as sentiment analysis. To test the effectiveness of our proposed approach, we conduct a series of binary (LibCon, MR and SST) and multi-class (Beauty, Book and Music) classification tasks, using our adaptation layer on top of several standard architectures. The standard architectures used in this paper are architectures that are common in the current State-Of-The-Art (SOTA) for sentence encoders (Conneau et al., 2017) and multi-domain sentiment analysis (Liu et al., 2018). These architectures are a BiLSTM based architecture followed by some sort of pooling and then a soft-max classification layer.

Our main result is that using the proposed domain adaptation layer on top of existing architectures helps improve the performance of the architecture by about $2 - 8\%$. Furthermore, we show that basic architectures enhanced with our DA layer outperform SOTA architectures built specifically for domain adaptation and transfer learning problems. We now give a description of the various datasets that we use and the various baseline algorithm used for our experimental comparisons.

### 4.1 Data Sets

Motivated by the fact that our DA layer is most useful when the target dataset is of modest size, we present our experimental results on three such datasets. Furthermore, the text used in these datasets often uses the same set of words to express clearly contrastive sentiments, thereby setting up groups or domains of distinct word use within a single data set.

**LibCon:** The LibCon (**Lib**eral and **Con**servative) data set is obtained from a study that aims towards understanding, analyzing and modeling the changing landscape of political discourse of an entire state over a duration of 10 years (Friedland et al., 2017). A key aspect of this study is the use of Twitter data to analyze the latent network structure among key players of Wisconsin's political-media ecology. To do this, data was drawn from a collection of Twitter data housed at the UW-Madison School of Journalism and Mass Communication. Data is drawn from a $10\%$ sample of Twitter messages worldwide from Twitter's API (Twitter 2012). Note that the period of data collection corresponds to the re-election efforts of Governor Walker (of Wisconsin). Snowball sampling is employed to identify important Twitter handles active in four 28-day periods in the first half of 2012. A hand curated list of Twitter handles corresponding to key players in Wisconsin's political climate is identified. Combined with Twitter handles of nationwide users that interact frequently with users in this list, the final list of Twitter handles of interest is formed.

Twitter handles are confirmed to their corre-

sponding political associations by verifying their party membership. The Twitter handle pool includes a balanced representation from Republican and Democratic accounts. In depth description of data collection methodologies and outcomes can be found in (Friedland et al., 2017) and readers are encouraged to consult this resource for additional information. A subsample of 1000 tweets, each from known Liberal (Democratic party) and Conservative (Republican party) users are considered for experiments here.

Gold standard list of topics key to both Democrat and Republican parties are obtained by analyzing primary debate data over a period of almost 20 years (1999-present) (Li et al., 2017). A combination of modeling techniques involving semantic spaces and neural network architectures is used to arrive at the final list of 136 top/key concepts such as 'Army', 'Border', 'Democracy', 'Justice' etc. Out of these 136 words, 74 occur in the LibCon data set as seen in Table 1.

**Movie Review:** This is a benchmark dataset (Pang and Lee, 2005) from which we randomly sample 2500 positive and 2500 negative reviews for experiments.

**SST:** The Stanford Sentiment Tree (SST) bank is yet another standard benchmark data set with reviews binarized to 'positive' or 'negative'. Our experiments use a 5000 sample training data set and a pre-determined test set of 5000 points.

**Beauty, Book, Music:** These data sets each consist of 6000 beauty product, book and music reviews with a balanced distribution of three class labels. This data set was introduced by (He et al., 2018). Data is obtained by sampling from a larger data set of product reviews obtained from Amazon (McAuley et al., 2015). Labels are 'positive', 'negative' and 'neutral'. An imbalanced setting for each data set is also available, where roughly 80% of the data points are 'positive' and the remaining are 'negative' and 'neutral'. Details about number of data points for each data set can be found in the supplement.

### 4.2 Baselines

For all test data sets, in addition to contrasting the performance of the proposed adaptation layer added to a CNN and BiLSTM encoder, we also present comparisons against 'Vanilla' CNN and BiLSTM encoders. For popular data sets like the MR and SST data sets we present results against some additional baselines, all of which are variants of the basic vanilla RNN encoders. The baselines that we used in this paper are,

**BoW:** In this standard baseline each sentence is expressed as a weighted sum of its constituent word embeddings. Weights used are raw word counts. Both generic (GloVe) and DA word embeddings are used in this baseline. Note that the DA embeddings used in this baseline are the same as in (K Sarma et al., 2018), i.e $\alpha = \beta = 0.5$

**Vanilla CNN:** A CNN based sentence classification framework as introduced by (Kim, 2014) is used as a 'Vanilla' CNN baseline.

**Vanilla BiLSTM:** A Basic BiLSTM encoder followed by max pooling as proposed by (Conneau et al., 2017) is used as the 'Vanilla' BiLSTM baseline. Note that the vanilla baseline contains no additional features like attention.

**BERT:** Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018) is a generic encoder framework that learns sentence embeddings by jointly conditioning on both left and right context in all layers. The Bert encoder is generic and requires a tuneable output layer to be learned in order to perform a chosen task. In our work, we use the pooled output from the Bert encoder to obtain sentence representations for all our data and then learn weights for a classifier to perform classification on the given data set.

**Self-attention:** A model that generates interpretable, structured sentence embeddings using Self-attention mechanism (Lin et al., 2017).

**LR-Bi-LSTM:** A model that imposes linguistic roles to Bi-LSTM architectures (Qian et al., 2017).

**DAS:** The Domain Adaptive Semi-supervised algorithm (He et al., 2018) is a transfer learning based solution that performs sentiment classification. In their experiments, the authors train on review data from different source domains such as book, music and electronics and test on a target domain such as beauty. In contrast, our method does not train on these three sources; rather, it uses only the Beauty reviews to learn the DS word embeddings (and then combines these with the generic pretrained embedding to learn the DA embedding used for classification. On the balanced data sets, metric reported is Accuracy, the same used by (He et al., 2018). On the imbalanced data sets, in addition to accuracy we report micro f-score.

### 4.3 Experimental methodology

The LibCon, MR and Beauty, Book and Music data sets do not have dedicated train/dev/test splits, so we created $80/10/10$ splits. For the SST dataset we make use of pre-defined test data set of 2210 data points. When pre-tuned models are available, such as for BERT, we further fine-tune it using our train/dev splits. All the baselines use the same splits of data.

**Hyperparameters:** All word embeddings-GloVe, DS and KCCA projections used to obtain the DA embeddings are of dimension 300. Both the CNN and BiLSTM encoders learn sentence embeddings of 300 dimensions on all data sets, except on the LibCon data set where the CNN encoders have a filter size of 120. Dropout probability is set to 0.5. The rest of the hyperparameters used are the same as in (He et al., 2018) and are found in the supplement.When using BERT, the 'bert-base-uncased' model is used to obtain sentence embeddings. Pooled output from the BERT encoder is used to represent the input sentence. The classifier learned during tuning on the train data sets is similar to the linear classification layer introduced in the BERT classification framework, with the exception of an additional Tanh activation. The additional Tanh activation function was used in order to obtain comparable results for BERT and other baselines. The size of test data is particularly challenging for a large model like BERT that has several learnable parameters, even when used only for fine-tuning the pre-trained model. The BERT classification framework is reproduced for use in our experiments, reflecting the original tuning scripts to the best of our ability. Additional information about hyperparameters can be found in the supplement.

### 4.4 Results

Table 2 presents results on the LibCon and the balanced (B) and imbalanced (I) Beauty, Book and Music data sets and Table 3 presents results on the SST and MR data sets. The performance metric reported in both tables is accuracy with with additional micro f-scores reported in Table 2.

From Table 2, it is observed that on the LibCon data set, where we have a considerable difference in language use between the two groups of users, the adapted BiLSTM and adapted CNN perform much better than the vanilla baselines. Furthermore, our proposed adaptation layer improves the performance of the Vanilla BiLSTM to surpass the performance of fine-tuned BERT. Note that the BERT encoder is pre-trained on all of Wikipedia, a much larger training data set than used for training the Adapted BiLSTM encoder. The adapted BiLSTM and CNN encoders also outperform DAS on the multiclass Beauty, Book and Music data sets. Performance of LSTMs for text surpass that of CNNs as seen in current literature. This observation is consistent in our results as well. Since the DAS baseline makes use of a CNN encoder, it is expected that a very basic LSTM such as the Vanilla BiLSTM would perform much better than any CNN framework. Results Table 4 strengthens our hypothesis that the adaptation layer is particularly well suited for small training and test data regimes.

Note that on the MR data set, the performance of fine-tuned BERT is lower than the BiLSTM baselines. While the size of training data used to optimize BERT, makes the encoder an excellent generic sentence encoder, the small size of the tuning data sets poses a considerable challenge. This is because, when the BERT encoder is fine tuned for a particular task, output from the encoder is treated as a fixed feature on top of which it learns a task specific layer. Using pre-trained BERT provides a high quality feature vector that can be used for applications on generic data sets, but this does not assure good performance when tuned on a small data set where words may be used idiosyncratically. On the other hand, on a highly polarized data set like the LibCon data set, while the distinct linguistic features enable the BERT encoder to capture good sentence embeddings, the size of the data set still poses a challenge.

**Varying Size of Training Data:** To further illustrate the effectiveness of our method on data sets with limited training data, we train the Vanilla and adapted BiLSTM and CNN encoders with even smaller samples of the training data. Our results are consistent on varying size of data sets and performance metrics for the same can be found in the supplement. Table 4 compares accuracy on the MR and SST data sets, of Vanilla and adapted encoders trained with 1000 and 2500 points. Repeating this experiment with BERT resulted in tremendous over-fitting on the test set and so we do not present results from BERT in Table 4.

**Qualitative analysis of DA word embeddings:** Here we present a small analysis of the DA embed-

| Algorithm | Acc on LibCon | Acc on Beauty (B) | Acc on Book (B) | Acc on Music (B) | Beauty (I) | | Book (I) | | Music (I) | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | Acc | F-score | Acc | F-score | Acc | F-score |
| BoW (GloVe) | 64.0 | 56.18 | 69.1 | 66.5 | 74.8 | 74.8 | 66.08 | 66.08 | 75.8 | 75.8 |
| BoW (DA embeddings, $\alpha = \beta = 0.5$) | 65.3 | 59.02 | 71.9 | 67.6 | 75.12 | 75.12 | 67.0 | 67.0 | 77.12 | 77.12 |
| Vanilla CNN | 61.36 | 61.7 | 66.5 | 61.5 | 74,5 | 74.6 | 76.8 | 78.1 | 77.5 | 77.5 |
| Vanilla BiLSTM | 68.5 | 75.6 | 77.6 | 80.0 | 84.3 | 84.3 | 83.4 | 83.4 | 84.2 | 84.2 |
| Adapted CNN | 63.24 | 62.8 | 71.0 | 63.5 | 80.1 | 75.0 | 78.0 | 76.6 | 78.1 | 78.1 |
| **Adapted BiLSTM** | **72.1** | **77.3** | **80.3** | **81.5** | **85.2** | **85.2** | **84.7** | **84.7** | **85.6** | **85.6** |
| BERT | 70.3 | - | - | - | - | - | - | - | - | - |
| DAS | N/A | 56.0 | 67.6 | 58.6 | 54.8 | - | 61.06 | - | 55.1 | - |

Table 2: This table reports performance (accuracy score) of the baseline algorithms on the LibCon and Beauty, Book and Music data sets in balanced and imbalanced setting. Micro F-score is reported on the imbalanced Beauty, Book and Music data sets as well. Bold face indicates best performing algorithm.

| Algorithm | Accuracy on MR | Accuracy on SST |
|---|---|---|
| BoW (Generic) | 75.7* | 48.9* |
| BoW (DA embeddings) | 77.0* | 49.2* |
| Vanilla CNN | 72.5* | 49.06* |
| Vanilla BiLSTM | 81.8* | 50.3* |
| LR-Bi-LSTM | 82.1 | 50.6 |
| Self-attention | 81.7 | 48.9 |
| Adapted CNN | 80.8* | 50.0* |
| **Adapted BiLSTM** | **83.1*** | 51.2 |
| BERT | 74.4* | **51.5** |

Table 3: This table reports performance (Accuracy) on the MR and SST data sets. Results with * indicate that performance metric is reported on the test dataset after training on a subset of the original data set. Bold face indicates best performing algorithm

| Algorithm | Accuracy on MR | Accuracy on SST |
|---|---|---|
| Vanilla CNN (1000 pts) | 65.1 | 48.8 |
| Adapted CNN (1000 pts) | 74.7 | 49.6 |
| Vanilla BiLSTM(1000 pts) | 76.9 | 50.08 |
| Adapted BiLSTM (1000 pts) | 78.1 | 50.3 |
| Vanilla CNN (2500 pts) | 66.5 | 49.0 |
| Adapted CNN (2500 pts) | 77.4 | 50.7 |
| Vanilla BiLSTM(2500 pts) | 78.8 | 50.2 |
| Adapted BiLSTM (2500 pts) | 80.1 | 51.0 |

Table 4: This table presents the accuracy obtained by Vanilla and adapted baselines on smaller subsamples of the training data for the MR and SST data sets.

dings obtained from the weights learned via the shallow adaptation layer. The analysis presented here is on the Book review data set consisting of roughly 10000 unique word tokens. Weights learned via the CNN encoder on this data set are $\alpha = 0.7145$ and $\beta = 0.3994$. To compare the DA embeddings against the GloVe common crawl embeddings, we first standardize both sets of embeddings to have an average norm of 1. Then we compute shift via the $l_2$ distance as described in Section 3.1. Out of 100 most shifted words, Adjectives (21%) and Adverbs (22%) such as 'emotional', 'profound' and 'emotionally' and 'obviously' shift the most. These words are used extensively when writing reviews about books. On the other hand words that shift the least are Nouns (69%) such as 'Higgins', 'Gardner' and 'Schaffer' that correspond to author and character names. This observation is consistent with the domain adaptation hypothesis. A data set of book reviews will use Adjectives and Adverbs differently in-domain as opposed to out-of-domain. On the other hand, Nouns and Noun Phrases such as character and author names remain fixed regardless.

## 5 Conclusions and Future Work

This paper shows that domain semantics captured in adapted word embeddings can improve the performance of (pre-trained) encoders in downstream tasks such as sentiment classification; particularly on data sets where obtaining data sufficient enough for tuning pre-trained encoders or for end-to-end training is difficult. Experiments show the effectiveness of the method in binary and multiclass classification tasks. The proposed framework outperforms competing transfer learning based algorithms in overcoming limitations posed by the size of training data, while learning fewer parameters than an end-to-end trained network along with obtaining better results on classification tasks. Recent work that focuses on capturing domain semantics in word embeddings, demonstrate the effectiveness of such approaches mostly via performance on a downstream task such as sentiment analysis/classification alone. In Section 3.1 we provide a simple yet effective demonstration of how techniques such as KCCA, that are used to capture domain semantics, can capture relevant semantics in the learned word embeddings. As future work, we will interface the adaptation layer for use recently introduced large scale language models such as BERT. Due to size and memory complexities of available implementation of large scale language models such as BERT, an easy integration with our proposed adaptation framework does not currently exist.

## References


Jisun An, Haewoon Kwak, and Yong-Yeol Ahn. 2018. Semaxis: A lightweight framework to characterize domain-specific word semantics beyond sentiment. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2450–2461. Association for Computational Linguistics.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Wells Friedland, Wagner Shah, and Abhishek. 2017. The civic state under threat: How social, political and media changes eroded wisconsin's civic culture.

William L. Hamilton, Kevin Clark, Jure Leskovec, and Dan Jurafsky. 2016. Inducing domain-specific sentiment lexicons from unlabeled corpora. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 595–605, Austin, Texas. Association for Computational Linguistics.

Viktor Hangya, Fabienne Braune, Alexander Fraser, and Hinrich Schütze. 2018. Two methods for domain adaptation of bilingual tasks: Delightfully simple and broadly applicable. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 810–820. Association for Computational Linguistics.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2018. Adaptive semi-supervised learning for cross-domain sentiment classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3467–3476. Association for Computational Linguistics.

Prathusha K Sarma, Yingyu Liang, and William Sethares. 2018. Domain adapted word embeddings for improved sentiment classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 37–42. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Ping Li, Benjamin Schloss, and D Jake Follmer. 2017. Speaking two languages in america: A semantic space analysis of how presidential candidates and their supporters represent abstract political concepts differently. *Behavior research methods*, 49(5):1668–1685.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101*.

Qi Liu, Yue Zhang, and Jiangming Liu. 2018. Learning domain representation for multi-domain sentiment classification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 541–550. Association for Computational Linguistics.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 43–52. ACM.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013a. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Hyeonseob Nam and Bohyung Han. 2016. Learning multi-domain convolutional neural networks for visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4293–4302.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 115–124. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word

representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Qiao Qian, Minlie Huang, Jinhao Lei, and Xiaoyan Zhu. 2017. Linguistically regularized lstm for sentiment classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1679–1689. Association for Computational Linguistics.

Jingbo Shang, Liyuan Liu, Xiaotao Gu, Xiang Ren, Teng Ren, and Jiawei Han. 2018. Learning named entity tagger using domain-specific dictionary. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2054–2064. Association for Computational Linguistics.

Jiali Zeng, Jinsong Su, Huating Wen, Yang Liu, Jun Xie, Yongjing Yin, and Jianqiang Zhao. 2018. Multi-domain neural machine translation with word-level domain context discrimination. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 447–457. Association for Computational Linguistics.

# Domain-Invariant Feature Distillation for Cross-Domain Sentiment Classification

**Mengting Hu**[1]* **Yike Wu**[1]* **Shiwan Zhao**[2]†
**Honglei Guo**[2] **Renhong Cheng**[1] **Zhong Su**[2]
[1] Nankai University [2] IBM Research - China
mthu@mail.nankai.edu.cn, wuyike@dbis.nankai.edu.cn, {zhaosw, guohl}@cn.ibm.com
chengrh@nankai.edu.cn, suzhong@cn.ibm.com

## Abstract

Cross-domain sentiment classification has drawn much attention in recent years. Most existing approaches focus on learning domain-invariant representations in both the source and target domains, while few of them pay attention to the domain-specific information. Despite the non-transferability of the domain-specific information, simultaneously learning domain-dependent representations can facilitate the learning of domain-invariant representations. In this paper, we focus on aspect-level cross-domain sentiment classification, and propose to distill the domain-invariant sentiment features with the help of an orthogonal domain-dependent task, i.e. aspect detection, which is built on the aspects varying widely in different domains. We conduct extensive experiments on three public datasets and the experimental results demonstrate the effectiveness of our method.

## 1 Introduction

Sentiment classification based on deep learning methods has developed rapidly in recent years. While achieving outstanding performance, these methods always need large-scale datasets with sentiment polarity labels to train a robust sentiment classifier. However, in most cases, large-scale labeled datasets are not available in practice and manual annotation costs much. One of the solutions to this problem is *cross-domain sentiment classification*, which aims to exploit the rich labeled data in one domain, i.e. *source domain*, to help the sentiment analysis task in another domain lacking for or even without labeled data, i.e. *target domain*. The rationality of this solution is that the source domain and target domain share some domain-invariant knowledge that can be transferred across domains.

*Source Domain*: The fried rice is amazing here.
*Target Domain*: Surprisingly, Britney Spears is amazing.

Figure 1: Example sentences from the source domain (restaurant) and the target domain (twitter) respectively. The sentiment expressions marked by solid lines are domain-invariant, while the aspect terms marked by dashed lines are domain-specific.

Previous works on cross-domain sentiment classification mainly focus on learning the domain-invariant representations in both source and target domains, either based on manual feature selection (Blitzer et al., 2006; Pan et al., 2010) or automatic representation learning (Glorot et al., 2011; Chen et al., 2012; Ganin and Lempitsky, 2015; Li et al., 2017). The sentiment classifier, which makes decisions based on the domain-invariant features and receives the supervisory signals from the source domain, can be also applied to the target domain. We can draw an empirical conclusion: the better domain-invariant features the method obtains, the better performance it gains. However, few studies explore the usage of the domain-specific information, which is also helpful to the cross-domain sentiment classification. Peng et al. (2018) propose to extract the domain-invariant and domain-dependent features of the target domain data and train two classifiers accordingly, but they require a few sentiment polarity labels in the target domain, which limits the practical application of the method.

In this paper, we exploit the domain-specific information by adding an orthogonal domain-dependent task to "distill" the domain-invariant features for cross-domain sentiment classification. The proposed method *domain-invariant feature distillation* (DIFD) does not need any sentiment

* Work performed while interning at IBM Research - China.
† Corresponding author.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5559–5568, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

polarity labels in the target domain, which is more consistent with the practical settings. Specifically, we focus on the aspect-level cross-domain sentiment classification, and train a shared sentiment classifier and two respective aspect detectors in the source and target domains. We argue that aspect detection is an orthogonal domain-dependent task with respect to the sentiment classification. As shown in Figure 1, given an input sentence, the sentiment classifier predicts its sentiment polarity based on the opinion words shared by different domains, while the aspect detector identifies the aspect terms which vary significantly across domains. The information on which the two tasks depend is mutually exclusive in the sentence, i.e. orthogonal. Therefore, by training these two tasks simultaneously, the aspect detectors will try to strip the domain-specific features from the input sentence and make the domain-invariant features purer, which is helpful to the cross-domain sentiment classification.

Moreover, we design two effective modules to boost the distillation process. One is the word-level context allocation mechanism. It modulates the importance of the words in the input sentence according to the property of different tasks. The other is the domain classifier. It tries to correctly judge which domain the domain-invariant feature comes from, while the other modules in the proposed method try to "fool" it, and the whole framework is trained in an adversarial way.

To summarize, the main contributions of our paper are as follows:

- We distill the domain-invariant sentiment features to improve the cross-domain sentiment classification by simultaneously training an aspect detection task that striping the domain-specific aspect features from the input sentence.
- We boost the separation process of the domain-invariant and domain-specific features by two effective modules which are the context allocation mechanism and domain classifier respectively.
- Experimental results demonstrate the effectiveness of the proposed method, and we further verify the rationality of the context allocation mechanism by visualization.

## 2 Related Work

**Cross-domain sentiment analysis:** Many domain adaptation methods have been proposed for sentiment analysis. SCL (Blitzer et al., 2006) learns correspondences among features from different domains. SFA (Pan et al., 2010) aims at reducing the gap between domains by constructing a bipartite graph to model the co-occurrence relationship between domain-specific words and domain-independent words. SDA (Glorot et al., 2011) learns to extract a meaningful representation for each review in an unsupervised fashion. mSDA (Chen et al., 2012) is an efficient method to marginalize noise and learn features. Gradient Reversal Layer (GRL) (Ganin and Lempitsky, 2015; Ganin et al., 2016; Li et al., 2017) is employed to learn domain-invariant representations by fooling the domain classifier. The replacement of gradient reversal with alternating minimization (Shu et al., 2018) stabilizes domain adversarial training, and we employ this method as the adversarial training.

**Aspect-level sentiment domain adaptation:** To the best of our knowledge, there are two works about aspect-related cross-domain sentiment classification. Li et al. (2019) propose a method to employ abundant aspect-category data to assist the scarce aspect-term level sentiment prediction. Zhang et al. (2019) propose IATN to address that aspects have different effects in different domains. Their method predicts sentiment polarity for the whole sentence rather than a specific aspect.

Our method concentrates on aspect-term level sentiment domain adaptation by separating the domain-specific aspect features. Bousmalis et al. (2016) and Liu et al. (2017) separate features into two subspaces by introducing constraints on the learned features. The difference is that our method is more fine-grained and utilizes the explicit aspect knowledge.

**Auxiliary task for sentiment domain adaptation:** Auxiliary task has been employed to improve cross-domain sentiment analysis. Yu and Jiang (2016) use two pivot-prediction auxiliary tasks to help induce a sentence embedding, which works well across domains for sentiment classification. Yu and Jiang (2017) propose to jointly learn domain-independent sentence embeddings by auxiliary tasks to predict sentiment scores of domain-independent words. Chen et al. (2018) design auxiliary domain discriminators for better transferring knowledge between do-

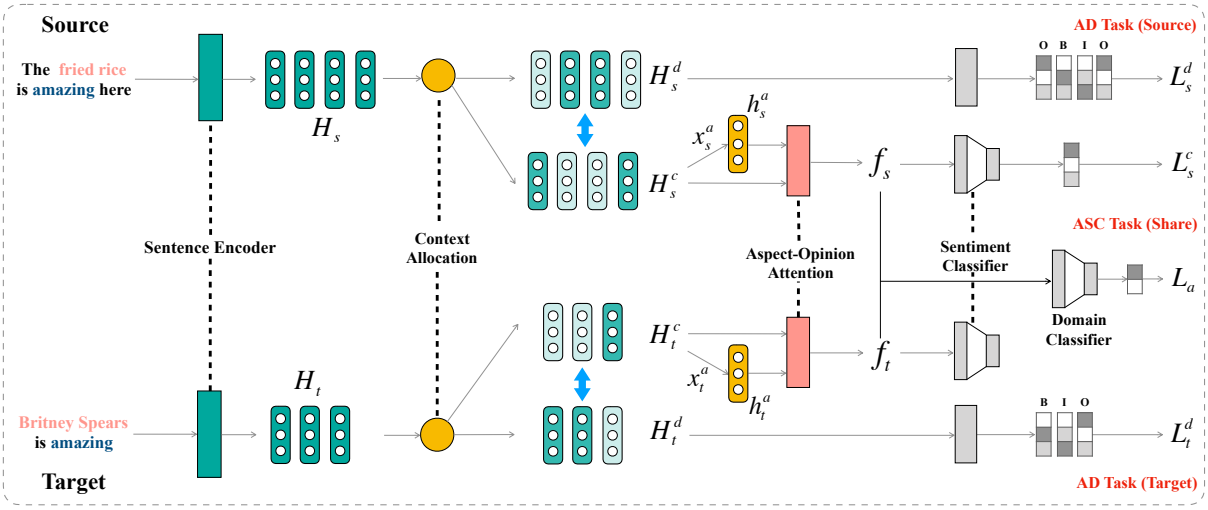


Figure 2: Network Architecture. The dashed line indicates that the parameters are shared by source and target domains. The context from the sentence encoder is divided into task-oriented contexts for ASC and AD tasks.

mains. These auxiliary tasks focus on directly enhancing domain-invariant features, while ours strips domain-specific features to distill domain-invariant features.

## 3 Methodology

### 3.1 Formulation and Overview

Suppose the source domain contains labeled data $\mathcal{D}_s = \{(x_s^k, a_s^k), y_s^k\}_{k=1}^{N_s}$, and the target domain contains unlabeled data $\mathcal{D}_t = \{(x_t^k, a_t^k)\}_{k=1}^{N_t}$, where $x$ is a sentence, $a$ is one of the aspects in $x$, and $y$ is the sentiment polarity label of $a$. The proposed method handles two kinds of tasks. One is the main task Aspect-level Sentiment Classification (**ASC**). It learns a mapping $\mathcal{F}: \{x\} \rightarrow \{\boldsymbol{f}\} \rightarrow \{y\}$ shared by source and target domains, where $\boldsymbol{f}$ is the domain-invariant feature of $x$. The other is the orthogonal domain-dependent task Aspect Detection (**AD**). It learns a mapping $\mathcal{G}_s: \{x_s\} \rightarrow \{\boldsymbol{z_s}\} \rightarrow \{a_s\}$ in the source domain, and the other one $\mathcal{G}_t: \{x_t\} \rightarrow \{\boldsymbol{z_t}\} \rightarrow \{a_t\}$ in the target domain, where $\boldsymbol{z}_s$ and $\boldsymbol{z}_t$ are both domain-dependent features of $x_s$ and $x_t$ respectively. The domain-invariant and domain-dependent features are orthogonal, i.e. $\boldsymbol{f} \perp \boldsymbol{z_s}$ and $\boldsymbol{f} \perp \boldsymbol{z_t}$. We facilitate the distillation of $\boldsymbol{f}$ by simultaneously learning $\mathcal{G}_s$ and $\mathcal{G}_t$ which try to strip $\boldsymbol{z_s}$ and $\boldsymbol{z_t}$ from $x$, and the purer $\boldsymbol{f}$ leads to the better $\mathcal{F}$ for the cross-domain sentiment classification.

Figure 2 illustrates the architecture overview of our method. Given an input sentence either from the source or target domain, we first feed it into the sentence encoder to obtain its distributed representation. Then, the context allocation mechanism divides the distributed representation into two orthogonal parts: domain-invariant and domain-dependent features. Finally, the two orthogonal features are fed into their corresponding downstream tasks. Specifically, we input the domain-invariant feature into the sentiment classifier to predict the sentiment polarity, and input the domain-dependent feature into the aspect detector of the specific domain to identify the aspect terms. In addition, we add a domain classifier to the architecture. It tries to correctly judge which domain the domain-invariant feature comes from. The whole framework is trained in an adversarial way. Next, we will introduce the components of our method in detail.

### 3.2 Sentence Encoder

Given an input sentence $x = \{w_1, w_2, ..., w_n\}$, we first map it into an embedding sequence $\hat{E} = \{\boldsymbol{e_1}, \boldsymbol{e_2}, ..., \boldsymbol{e_n}\} \in \mathbb{R}^{n \times d_e}$. Then we inject the positional information of each token in $x$ into $\hat{E}$ to obtain the final embedded representation $E$, following the Position Encoding (PE) method in the work (Vaswani et al., 2017):

$$\begin{aligned} PE(pos, 2i) &= sin(pos/10000^{2i/d_e}) \\ PE(pos, 2i+1) &= cos(pos/10000^{2i/d_e}) \\ E &= \hat{E} + PE \end{aligned} \quad (1)$$

where $pos$ is the word position in the sentence and $i$ is the $i$-th dimension of $d_e$. We consider that the injected positional information can facilitate the aspect-level sentiment classification, based on the

observation that sentiment words tend to be close to its related aspect terms (Tang et al., 2016; Chen et al., 2017).

Next we employ a Bi-directional LSTM (BiLSTM) (Graves et al., 2013) to encode $E$ into the contextualized sequence representation $H = [\boldsymbol{h_1}, \boldsymbol{h_2}, ...\boldsymbol{h_n}] \in \mathbb{R}^{n \times 2d_h}$, which preserves the contextual information of each token in the input sentence.

We unify the embedding layer and BiLSTM as the sentence encoder in which different tasks or domains all share the same weights. The advantages of sharing the weights are two-fold: first, different tasks in the same domain can benefit from each other in a multi-task manner; second, distilling the domain-invariant feature from a common transformation is more simple.

### 3.3 Context Allocation (CA)

In an input sentence, some words have a strong bias towards domain-specific information, such as the aspect terms, e.g. *"pizza"* in Restaurant domain, while others focus on the domain-invariant knowledge, such as the opinion words, e.g. *"amazing"*. Meanwhile, the ASC task and AD task exactly require orthogonal information as discussed before. Therefore, we argue that different words contribute differently according to the property of the downstream task. To facilitate the distillation of the domain invariant features, we propose a Context Allocation (CA) mechanism to allocate different weights on the same word in different downstream tasks. The values of the weights depend on how the information contained in the word matches the need of the specific task. Concretely, at each time step $i$, the module divides the contextualized representation $h_i$ of word $w_i$ into the sentiment-dominant context $h_i^c$ and aspect-dominant context $h_i^d$ as follows:

$$\boldsymbol{h_i^c} = \beta_i^c \boldsymbol{h_i}, \tag{2}$$

$$\boldsymbol{h_i^d} = \beta_i^d \boldsymbol{h_i}, \tag{3}$$

$$\beta_i^c + \beta_i^d = 1, \quad i \in \{1, 2, ..., n\}. \tag{4}$$

The two-dimensional vector $\beta_i = (\beta_i^c, \beta_i^d)$ is normalized considering that the domain-specific information and domain-invariant knowledge are mutually exclusive. It reflects the importance of $w_i$ on the ASC task and AD task respectively, and is calculated on $h_i$ as follows:

$$\beta_i = \text{softmax}(W_b \text{tanh}(W_a \boldsymbol{h_i}^{\mathrm{T}})), \tag{5}$$

where $W_a \in \mathbb{R}^{2d_h \times 2d_h}$ and $W_b \in \mathbb{R}^{2 \times 2d_h}$. The whole division process at all time steps can be formulated in the following form:

$$\begin{bmatrix} H_c \\ H_d \end{bmatrix} = \boldsymbol{\beta} \cdot \begin{bmatrix} H \\ H \end{bmatrix}, \tag{6}$$

where $\boldsymbol{\beta} = [\beta_1, \beta_2, ..., \beta_n]$. The sentiment-dominant context $H^c \in \mathbb{R}^{n \times 2d_h}$ and aspect-dominant context $H^d \in \mathbb{R}^{n \times 2d_h}$ of the input sentence are then fed into the ASC task and AD task for downstream processing respectively.

### 3.4 Aspect-level Sentiment Classification (ASC) Task

**Aspect-Opinion Attention** In the ASC task, we design an attention mechanism to model the relationship between the position of the aspect terms and their corresponding opinion words. For a specific aspect term, the domain-invariant feature based on the aspect-opinion attention contains more information of its corresponding opinion words, which is beneficial to the final aspect-level sentiment classification. Specifically, we first calculate the position representation of a specific aspect term with its position $x^a$ and the sentiment-dominant context $H^c$:

$$\boldsymbol{h^a} = H^c x^a \tag{7}$$

where $x^a = \{0_1, ..., 1_{i+1}, ..., 1_{i+m}, ...0_n\}$ represents the word positions of an aspect subsequence in the input sentence $x$ with non-zero values and $m$ is the length of the aspect. Then the representation $\boldsymbol{h^a} \in \mathbb{R}^{2d_h}$ is further utilized to calculate the sentiment-dominant feature $\boldsymbol{f}$, which is domain-invariant and should be aligned across source and target domains.

$$\begin{aligned} \gamma_i &= \tanh(\boldsymbol{h_i^c} \cdot W_p \cdot \boldsymbol{h^a} + b_p) \\ \boldsymbol{f} &= \sum_{i=1}^{n} \gamma_i \boldsymbol{h_i^c} \end{aligned} \tag{8}$$

where $\gamma_i$ reflects how much the word $w_i$ corresponds with the opinion on the aspect term, and $W_p \in \mathbb{R}^{2d_h \times 2d_h}$ and $b_p \in \mathbb{R}^1$ are weight matrix and bias respectively.

**Sentiment Classification Loss** The sentiment-dominant features $\boldsymbol{f_s}$ and $\boldsymbol{f_t}$ generated from the source and target domains respectively share the same sentiment classifier. Note that the source domain data has sentiment polarity label, while the

target domain is unlabeled. Thus we train the sentiment classifier only with the labeled data in the source domain, while utilizing it for inference in both source and target domains. The training objective of the sentiment classifier is to minimize the following loss on the source domain dataset, which is marked as $L_s^c$:

$$L_s^c = -\frac{1}{N_s}\sum^{N_s} \log P(y|\boldsymbol{f_s}) \tag{9}$$

where $y$ is the ground-truth sentiment polarity label. For simplicity, we omit the enumerated number of the instance in the loss equation.

**Domain Adversarial Loss** The domain classifier maps the sentiment-dominant feature $\boldsymbol{f}$ into a two-dimensional normalized value $\boldsymbol{y} = (y_s, y_t)$, which indicates the probability that $\boldsymbol{f}$ comes from the source and target domains respectively. The ground-truth domain label is $\boldsymbol{g_s} = (1, 0)$ for instances in the source domain, and $\boldsymbol{g_t} = (0, 1)$ in the target domain. The training objective of the domain classifier is to minimize the following loss on both source and target domain datasets, which is marked as $L_a^{\theta_D}$:

$$L_a^{\theta_D} = -\frac{1}{N_s}\sum^{N_s} \boldsymbol{g_s}\log\boldsymbol{y} - \frac{1}{N_t}\sum^{N_t} \boldsymbol{g_t}\log\boldsymbol{y}. \tag{10}$$

The part in our architecture which joins the generating process of $\boldsymbol{f}$ (including Sentence Encoder, Context Allocation and Aspect-Opinion Allocation in Figure 2) can be regarded as a domain-invariant feature extractor, which works with the domain classifier in an adversarial way. To further accelerate the distillation process of the domain-invariant features, we also introduce an adversarial loss of the domain classifier for the feature extractor. Specifically, we calculate the loss in Equation 10 with the flipped domain labels inspired by the work (Shu et al., 2018):

$$L_a^{\theta_F} = -\frac{1}{N_s}\sum^{N_s} \boldsymbol{g_t}\log\boldsymbol{y} - \frac{1}{N_t}\sum^{N_t} \boldsymbol{g_s}\log\boldsymbol{y}. \tag{11}$$

### 3.5 Aspect Detection (AD) Task

We model the AD task as a sequence labeling problem, and each word in the sentence is marked as a tag in $\{\mathrm{B}, \mathrm{I}, \mathrm{O}\}$, which means the word is at the beginning (B) or the inside (I) of an aspect term or other word (O). In this way, we can detect all the aspect terms of an input sentence in one forward pass. Specifically, we first linearly transform the aspect-dominant hidden state $\boldsymbol{h^d}$ into a three-dimensional vector. Then we calculate the aspect detection loss of the source domain as follows:

$$L_s^d = -\frac{1}{N_s}\sum^{N_s}\frac{1}{n}\sum^{n}\lambda_l \log P(y^d|\boldsymbol{h^d}) \tag{12}$$

where $y^d$ is the ground-truth aspect label, $n$ is the sentence length and $\lambda_l$ is the weight of different labels. The weight $\lambda_l$ aims to solve the class imbalance problem because the words labeled by O usually make up the majority of one sentence. It is dynamically calculated in the training phase according to the ratio of the words with a specific label in each batch. Henceforth we denote the loss of the AD task in the target domain as $L_t^d$.

**Algorithm 1** Adversarial Training

**Input:** labeled $\mathcal{D}_s$ and unlabeled $\mathcal{D}_t$

1: **repeat**
2: **Train** All parameters except Domain Classifier with $L$;
3: **Train** Domain Classifier with $\lambda^a L_a^{\theta_D}$;
4: **until** *performance on the validation set does not improve in 10 epochs.*

### 3.6 Training

We combine each component loss into an overall object function:

$$L = L_s^c + \lambda^a L_a^{\theta_F} + \lambda^d (L_s^d + L_t^d) \tag{13}$$

where $\lambda^a$ and $\lambda^d$ balance the effect of the domain classifier and the auxiliary task (i.e. aspect detection). $L$ and $\lambda^a L_a^{\theta_D}$ are alternatively optimized. The aspect-level sentiment analysis in the unlabeled target domain is predicted by the ASC task.

## 4 Experiments

### 4.1 Datasets

To make an extensive evaluation, we employ three different datasets: Restaurants (R) and Laptops (L) from SemEval 2014 task 4 (Pontiki et al., 2014), and Twitters (T) from the work (Dong et al., 2014). The statistics of these three datasets are shown in Table 1. Specifically, we collect the aspect-term level sentences and corresponding labels from these datasets. Comparing aspect terms

| Dataset | | #Pos | #Neg | #Neu | Total |
|---|---|---|---|---|---|
| Restaurants (R) | Train | 2164 | 805 | 633 | 3502 |
| | Test | 728 | 196 | 196 | 1120 |
| Laptops (L) | Train | 987 | 866 | 460 | 2313 |
| | Test | 341 | 128 | 169 | 638 |
| Twitters (T) | Train | 1561 | 1560 | 3127 | 6248 |
| | Test | 173 | 173 | 346 | 692 |

Table 1: Datasets statistics. #Pos, #Neg, #Neu represent the number of instances with positive, negative and neutral polarities.

in these three datasets, we find more than 98% aspect terms are different between Restaurants and Laptops domains, and there exists no same aspect between Restaurants and Twitters, also only 0.09% same aspects between Laptops and Twitters. This indicates that the aspect terms vary violently in different domains.

### 4.2 Experimental Settings

To evaluate our proposed method, we construct six aspect-level sentiment transfer tasks: R→L, L→R, R→T, T→R, L→T, T→L. The arrow indicates the transfer direction from the source domain to the target domain. For each transfer pair $\mathcal{D}_s \rightarrow \mathcal{D}_t$, the training set is composed of two parts: one is the labeled training set in $\mathcal{D}_s$, and the other is all unlabeled data which only contain the aspect term information in $\mathcal{D}_t$. The test set in $\mathcal{D}_s$ is employed as the validation set. The reported results are evaluated on all the data of $\mathcal{D}_t$.

The word embeddings are initialized with 100-dimension Glove vectors (Pennington et al., 2014) and fine-tuned during the training. The model hidden size $d_h$ is set to be 64. The model is optimized by the SGD method with the learning rate of 0.01. The batch size is 32. We employ ReLU as the activation function.

We adopt an early stop strategy during training if the performance on the validation set does not improve in 10 epochs, and the best model is chosen for evaluation.

### 4.3 Compared Methods

We compare with extensive baselines to validate the effectiveness of the proposed method. Some variants of our approach are also compared for analyzing the impacts of individual components.

**Transfer Baseline:** The aspect-level cross-domain sentiment classification has been rarely explored. We choose the state-of-the-art method IATN (Zhang et al., 2019) which has the most similar settings with our method as the transfer baseline. It proposes to incorporate the information of both sentences and aspect terms in the cross-domain sentiment classification.

**Non-Transfer Baselines:** The non-transfer baselines are all representative methods in recent years for the aspect-level sentiment classification in a single domain. We train the models on the training set of the source domain, and directly test them in the target domain without domain adaptation.

- **AT-LSTM** (Wang et al., 2016): It utilizes the attention mechanism to generate an aspect-specific sentence representation.
- **ATAE-LSTM** (Wang et al., 2016): It also employs attention. The difference with AT-LSTM is that the aspect embedding is as input to LSTM.
- **MemNet** (Tang et al., 2016): It employs a memory network with multi-hops attentions and predicts the sentiment based on the top-most context representations.
- **IAN** (Ma et al., 2017): It adopts interactive attention mechanism to learn the representations of the context and the aspect respectively.
- **RAM** (Chen et al., 2017): It employs multiple attentions with a GRU cell to non-linearly combine the aggregation of word features in each layer.
- **GACE** (Xue and Li, 2018): It is based on the convolutional neural network with gating mechanisms.

**Variants of Our Method:**

- **ASC+AT** (ASC with adversarial training): A single task that handles the ASC task with adversarial training.
- **DIFD:** The proposed method in this work.
- **DIFD(S)**: It contains components of the source domain from DIFD and is trained only by the source domain data.
- **DIFD-CA**: DIFD without context allocation.
- **DIFD-AT**: DIFD without adversarial training.
- **DIFD-AT+MMD**: Replace the adversarial training with Maximum Mean Discrepancy (MMD) (Tzeng et al., 2014).
- **DIFD-AT+CORAL**: Replace the adversarial training with CORAL (Sun et al., 2016).

| Models | R→L | | L→R | | R→T | | T→R | | L→T | | T→L | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Acc | F1 | Acc | F1 | Acc | F1 | Acc | F1 | Acc | F1 | Acc | F1 |
| ATAE-LSTM | 56.56 | 47.71 | 63.21 | 46.01 | 34.97 | 33.82 | 50.78 | 44.06 | 40.43 | 39.82 | 42.39 | 40.73 |
| MemNet | 57.17 | 47.79 | 65.97 | 49.28 | 34.47 | 32.37 | 50.36 | 42.68 | 44.94 | 44.37 | 36.36 | 34.96 |
| RAM | 58.32 | 43.98 | 54.68 | 26.81 | 33.18 | 29.96 | 48.18 | 43.23 | 44.87 | 44.06 | 42.73 | 42.51 |
| GACE | 61.74 | 50.39 | 66.60 | 49.07 | 32.87 | 29.34 | 45.62 | 37.52 | 44.93 | 45.18 | 47.98 | 42.39 |
| AT-LSTM | 60.62 | 45.38 | 66.75 | 46.99 | 32.98 | 28.98 | 50.64 | 43.64 | 38.47 | 36.78 | 47.34 | 42.87 |
| IAN | 60.39 | 50.69 | 66.50 | 47.97 | 35.09 | 33.02 | 51.14 | 44.46 | 44.16 | 44.34 | 43.88 | 40.16 |
| IATN | 62.05 | 46.37 | 67.13 | 51.64 | 34.14 | 30.55 | 40.13 | 37.79 | 44.80 | 44.69 | 46.12 | 42.91 |
| DIFD(S) | 63.81 | 57.74 | 68.30 | 53.66 | 36.89 | 33.74 | 55.63 | 44.10 | 45.73 | 46.17 | 44.53 | 43.76 |
| DIFD | **64.86** | **60.51** | **68.53** | **57.31** | **40.13** | **38.85** | **57.60** | **46.59** | **47.32** | **47.31** | **48.97** | **47.56** |

Table 2: Evaluation results of baselines in terms of accuracy (%) and macro-f1 (%).

| Models | R→L | | L→R | | R→T | | T→R | | L→T | | T→L | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Acc | F1 | Acc | F1 | Acc | F1 | Acc | F1 | Acc | F1 | Acc | F1 |
| ASC+AT | 62.86 | 56.47 | 67.66 | 49.84 | 35.86 | 33.43 | 52.33 | 44.41 | 44.15 | 44.31 | 42.46 | 37.94 |
| DIFD-CA | 64.18 | 57.59 | 68.17 | 53.42 | 34.27 | 31.52 | 44.39 | 41.79 | 44.14 | 44.25 | 43.54 | 43.60 |
| DIFD-AT | 63.47 | 60.09 | 68.15 | 57.27 | 40.76 | 38.66 | **59.76** | 45.08 | 45.73 | 45.91 | 47.68 | 40.20 |
| DIFD-AT+MMD | 64.25 | 58.79 | 63.13 | 56.49 | 39.44 | 38.69 | 46.93 | 44.44 | **47.81** | **47.91** | 42.87 | 38.69 |
| DIFD-AT+CORAL | 63.77 | 58.61 | 68.30 | 53.46 | **44.65** | **42.83** | 57.54 | 46.41 | 46.96 | 46.96 | 38.80 | 35.97 |
| DIFD | **64.86** | **60.51** | **68.53** | **57.31** | 40.13 | 38.85 | 57.60 | **46.59** | 47.32 | 47.31 | **48.97** | **47.56** |

Table 3: Evaluation results of variants of our model in terms of accuracy(%) and macro-f1(%). The minus sign (-) means to remove the module, and the addition (+) means to add the module.

### 4.4 Experimental Analysis

We report the classification accuracy and macro-f1 of various methods in Table 2 and Table 3, and the best scores on each metric are marked in bold. To validate the effectiveness of our method, we analyze the results from the following perspectives.

**Compare with the baselines:** We display the comparison results with baselines in Table 2. Comparing with the transfer baseline IATN, we observe that DIFD significantly outperforms IATN on all metrics by +5.51% accuracy and +7.36% macro-f1 on average. This shows that the distillation of domain-invariant features really facilitates the transfer of sentiment information across domains. In addition, for a fair comparison with the non-transfer methods which only exploit the source domain data, we also train our DIFD model without the target domain data and denote this variant as DIFD(S). We observe that DIFD(S) outperforms all the non-transfer baselines on most metrics. It is worth noting that, compared to a strong baseline IAN, DIFD(S) achieves significant improvement by +4.49% accuracy on T→R and +7.05% macro-f1 on R→L. This verifies that the orthogonal task is helpful in striping the domain-specific features from the source domain and effective for accelerating the domain adaptation.

**Compare the variants of our method:** The results of the variants of our method are reported in Table 3. We first observe that DIFD outperforms ASC+AT on all metrics significantly. This validates that the orthogonality really helps to distill the domain-invariant features and improve the performance of the cross-domain sentiment classification.

Then we can see that DIFD-CA performs much worse than DIFD, which reveals that the context allocation mechanism plays an important role in our method. We further visualize the allocation scores in Figure 3 and the result also indicates that the reasonability of the CA module. The gray tokens and red tokens have a bias towards ASC task and AD task respectively. The allocation scores are consistent with the bias of words: red tokens get larger scores for the aspect detection task, while gray tokens get larger scores for opinion expressions. This shows that our model generates task-oriented contexts successfully.

Finally, DIFD also achieves improvement over DIFD-AT on most metrics. This indicates that

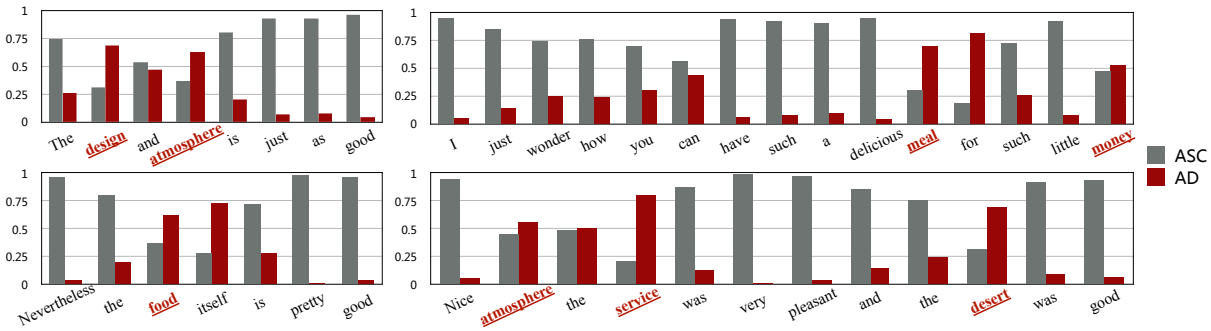


Figure 3: Visualization of context allocation weights of DIFD for different tokens within a sequence.

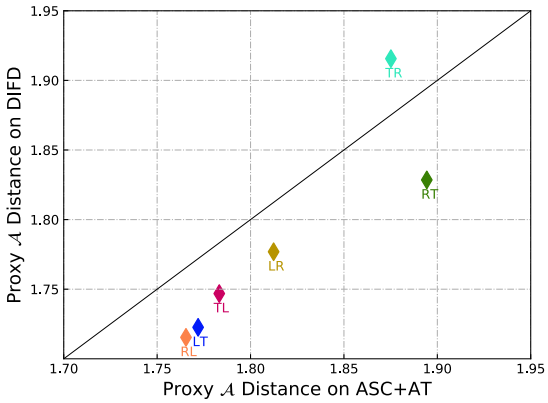


Figure 4: Proxy $\mathcal{A}$ Distance between domain-invariant features for the 6 different pairs.

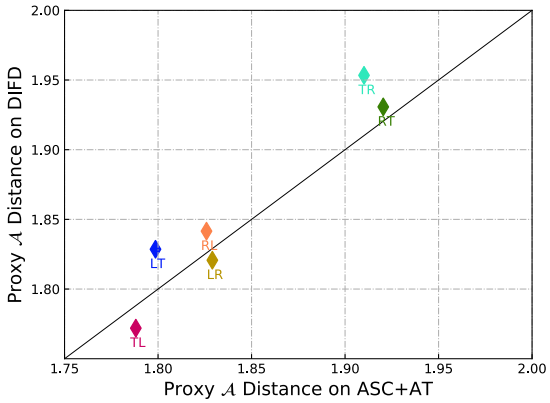


Figure 5: Proxy $\mathcal{A}$ Distance between domain-specific features for the 6 different pairs.

adversarial training with the domain classifier promotes the distillation process of the domain-invariant features. To further validate the effectiveness of adversarial training, we also try to directly minimize the divergence between domain-invariant features from source and target domains based on MMD and CORAL. Comparing with DIFD-AT+MMD and DIFD-AT+CORAL, DIFD is more robust considering that DIFD outperforms the two methods in most experimental settings.

### 4.5 Transfer Distance Analysis

In this section, we analyze the similarity of features between domains. We exploit the $\mathcal{A}$-distance (Ben-David et al., 2007) to measure the similarity between two probability distributions. The proxy $\mathcal{A}$-distance is 2(1-2$\epsilon$), where $\epsilon$ is the generalization error of a classifier (a linear SVM) trained on the binary classification problem to distinguish inputs between the two domains. We focus on the methods ASC+AT and DIFD, and first compare the similarity of domain-invariant features $\boldsymbol{f_s}$ and $\boldsymbol{f_t}$. Figure 4 reports the results for each pair of domains. The proxy $\mathcal{A}$-distance on DIFD is generally smaller than its corresponding value on ASC+AT. This indicates that DIFD can learn purer domain-invariant features than ASC+AT. Secondly, we compare the domain-specific features learned by ASC+AT and DIFD, which are represented by the average hidden state of BiLSTM in ASC+AT and the average aspect-context $H^d$ in DIFD respectively. Figure 5 reports the results for each pair of domains. The proxy $\mathcal{A}$-distance on DIFD is generally larger than its corresponding value on ASC+AT, which demonstrates that DIFD can strip more domain-specific information by the aspect detection task than ASC+AT.

There are exceptions in both Figures, i.e., TR in Figure 4, TL and LR in Figure 5. A possible explanation is that the balance between ASC and AD losses causes some domain-specific information to remain in the domain-invariant space, and vice versa.

## 5 Conclusion

In this work, we study the problem of aspect-level cross-domain sentiment analysis and propose a domain-invariant feature distillation method that simultaneously learns domain-invariant and

domain-specific features. With the help of the orthogonal domain-dependent task (i.e., aspect detection), the aspect sentiment classification task can learn better domain-invariant features and improve transfer performance. Experimental results clearly verify the effectiveness of our method.

## 6 Acknowledgement

This work is supported by National Science and Technology Major Project, China (Grant No. 2018YFB0204304).

## References

Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. 2007. Analysis of representations for domain adaptation. In *(NIPS)*, pages 137–144.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *(EMNLP)*, pages 120–128.

Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. 2016. Domain separation networks. In *(NIPS)*, pages 343–351.

Cen Chen, Yinfei Yang, Jun Zhou, Xiaolong Li, and Forrest Sheng Bao. 2018. Cross-domain review helpfulness prediction based on convolutional neural networks with auxiliary domain discriminators. In *(NAACL)*, pages 602–607.

Minmin Chen, Zhixiang Xu, Kilian Weinberger, and Fei Sha. 2012. Marginalized denoising autoencoders for domain adaptation. *(ICML)*.

Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. 2017. Recurrent attention network on memory for aspect sentiment analysis. In *(EMNLP)*, pages 452–461.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *(ACL)*, volume 2, pages 49–54.

Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. *(ICML)*.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *(ICML)*, pages 513–520.

Alex Graves, Abdel Rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *(ICASSP)*, pages 6645–6649.

Zheng Li, Ying Wei, Yu Zhang, Xiang Zhang, Xin Li, and Qiang Yang. 2019. Exploiting coarse-to-fine task transfer for aspect-level sentiment classification. *(AAAI)*.

Zheng Li, Yun Zhang, Ying Wei, Yuxiang Wu, and Qiang Yang. 2017. End-to-end adversarial memory network for cross-domain sentiment classification. In *(IJCAI)*, pages 2237–2243.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification.

Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2017. Interactive attention networks for aspect-level sentiment classification. *(IJCAI)*.

Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2010. Cross-domain sentiment classification via spectral feature alignment. In *(WWW)*.

Minlong Peng, Qi Zhang, Yu-gang Jiang, and Xuanjing Huang. 2018. Cross-domain sentiment classification with target domain specific information. In *(ACL)*, pages 2505–2513.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *(EMNLP)*, pages 1532–1543.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. *SemEval*, pages 27–35.

Rui Shu, Hung H Bui, Hirokazu Narui, and Stefano Ermon. 2018. A dirt-t approach to unsupervised domain adaptation. *(ICLR)*.

Baochen Sun, Jiashi Feng, and Kate Saenko. 2016. Return of frustratingly easy domain adaptation. In *(AAAI)*.

Duyu Tang, Bing Qin, and Ting Liu. 2016. Aspect level sentiment classification with deep memory network. *(EMNLP)*.

Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. 2014. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *(NIPS)*, pages 5998–6008.

Yequan Wang, Minlie Huang, Li Zhao, et al. 2016. Attention-based lstm for aspect-level sentiment classification. In *(EMNLP)*, pages 606–615.

Wei Xue and Tao Li. 2018. Aspect based sentiment analysis with gated convolutional networks. *(ACL)*.

Jianfei Yu and Jing Jiang. 2016. Learning sentence embeddings with auxiliary tasks for cross-domain sentiment classification. In *(EMNLP)*, pages 236–246.

Jianfei Yu and Jing Jiang. 2017. Leveraging auxiliary tasks for document-level cross-domain sentiment classification. ACL.

Kai Zhang, Hefu Zhang, Qi Liu, Hongke Zhao, Hengshu Zhu, and Enhong Chen. 2019. Interactive attention transfer network for cross-domain sentiment classification. In *(AAAI)*.

# A Novel Aspect-Guided Deep Transition Model for Aspect Based Sentiment Analysis

**Yunlong Liang**[1]*, **Fandong Meng**[2], **Jinchao Zhang**[2], **Jinan Xu**[1]†
**Yufeng Chen**[1] and **Jie Zhou**[2]
[1]Beijing Jiaotong University, China
[2]Pattern Recognition Center, WeChat AI, Tencent Inc, China
{yunlonliang,jaxu,chenyf}@bjtu.edu.cn
{fandongmeng,dayerzhang,withtomzhou}@tencent.com

## Abstract

Aspect based sentiment analysis (ABSA) aims to identify the sentiment polarity towards the given aspect in a sentence, while previous models typically exploit an aspect-independent (weakly associative) encoder for sentence representation generation. In this paper, we propose a novel **A**spect-**G**uided **D**eep **T**ransition model, named AGDT, which utilizes the given aspect to guide the sentence encoding from scratch with the specially-designed deep transition architecture. Furthermore, an aspect-oriented objective is designed to enforce AGDT to reconstruct the given aspect with the generated sentence representation. In doing so, our AGDT can accurately generate aspect-specific sentence representation, and thus conduct more accurate sentiment predictions. Experimental results on multiple SemEval datasets demonstrate the effectiveness of our proposed approach, which significantly outperforms the best reported results with the same setting.[1]

## 1 Introduction

Aspect based sentiment analysis (ABSA) is a fine-grained task in sentiment analysis, which can provide important sentiment information for other natural language processing (NLP) tasks. There are two different subtasks in ABSA, namely, aspect-category sentiment analysis and aspect-term sentiment analysis (Pontiki et al., 2014; Xue and Li, 2018). Aspect-category sentiment analysis aims at predicting the sentiment polarity towards the given aspect, which is in predefined several categories and it may not appear in the sentence. For instance, in Table 1, the aspect-category sentiment analysis is going to predict the sentiment polarity towards the aspect "*food*", which is not appeared in the sentence. By contrast, the goal of aspect-term sentiment analysis is to predict the sentiment polarity over the aspect term which is a subsequence of the sentence. For instance, the aspect-term sentiment analysis will predict the sentiment polarity towards the aspect term "*The appetizers*", which is a subsequence of the sentence. Additionally, the number of categories of the aspect term is more than one thousand in the training corpus.

| **Sentence** | The appetizers are ok, but the service is slow. | |
|---|---|---|
| **Aspect-Category** | food | service |
| **Aspect-Term** | The appetizers | service |
| **Sentiment Polarity** | Neutral | Negative |

Table 1: The instance contains different sentiment polarities towards two aspects.

As shown in Table 1, sentiment polarity may be different when different aspects are considered. Thus, the given aspect (term) is crucial to ABSA tasks (Jiang et al., 2011; Ma et al., 2017; Wang et al., 2018; Xing et al., 2019; Liang et al., 2019). Besides, Li et al. (2018a) show that not all words of a sentence are useful for the sentiment prediction towards a given aspect (term). For instance, when the given aspect is the "*service*", the words "*appetizers*" and "*ok*" are irrelevant for the sentiment prediction. Therefore, an aspect-independent (weakly associative) encoder may encode such background words (e.g., "*appetizers*" and "*ok*") into the final representation, which may lead to an incorrect prediction.

Numerous existing models (Tang et al., 2016b; Tay et al., 2017; Fan et al., 2018; Xue and Li, 2018) typically utilize an aspect-independent encoder to generate the sentence representation, and then apply the attention mechanism (Luong et al., 2015) or gating mechanism to conduct feature

* Work was done when Yunlong Liang was an intern at Pattern Recognition Center, WeChat AI, Tencent Inc, China.
† Jinan Xu is the corresponding author.
[1]The code is publicly available at: https://github.com/XL2248/AGDT



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5569–5580, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

selection and extraction, while feature selection and extraction may base on noised representations. In addition, some models (Tang et al., 2016a; Wang et al., 2016; Majumder et al., 2018) simply concatenate the aspect embedding with each word embedding of the sentence, and then leverage conventional Long Short-Term Memories (LSTMs) (Hochreiter and Schmidhuber, 1997) to generate the sentence representation. However, it is insufficient to exploit the given aspect and conduct potentially complex feature selection and extraction.

To address this issue, we investigate a novel architecture to enhance the capability of feature selection and extraction with the guidance of the given aspect from scratch. Based on the deep transition Gated Recurrent Unit (GRU) (Cho et al., 2014; Pascanu et al., 2014; Miceli Barone et al., 2017; Meng and Zhang, 2019), an aspect-guided GRU encoder is thus proposed, which utilizes the given aspect to guide the sentence encoding procedure at the very beginning stage. In particular, we specially design an aspect-gate for the deep transition GRU to control the information flow of each token input, with the aim of guiding feature selection and extraction from scratch, i.e. sentence representation generation. Furthermore, we design an aspect-oriented objective to enforce our model to reconstruct the given aspect, with the sentence representation generated by the aspect-guided encoder. We name this **A**spect-**G**uided **D**eep **T**ransition model as AGDT. With all the above contributions, our AGDT can accurately generate an aspect-specific representation for a sentence, and thus conduct more accurate sentiment predictions towards the given aspect.

We evaluate the AGDT on multiple datasets of two subtasks in ABSA. Experimental results demonstrate the effectiveness of our proposed approach. And the AGDT significantly surpasses existing models with the same setting and achieves state-of-the-art performance among the models without using additional features (e.g., BERT (Devlin et al., 2018)). Moreover, we also provide empirical and visualization analysis to reveal the advantages of our model. Our contributions can be summarized as follows:

- We propose an aspect-guided encoder, which utilizes the given aspect to guide the encoding of a sentence from scratch, in order to conduct the aspect-specific feature selection and extraction at the very beginning stage.
- We propose an aspect-reconstruction approach to further guarantee that the aspect-specific information has been fully embedded into the sentence representation.
- Our AGDT substantially outperforms previous systems with the same setting, and achieves state-of-the-art results on benchmark datasets compared to those models without leveraging additional features (e.g., BERT).

## 2 Model Description

As shown in Figure 1, the AGDT model mainly consists of three parts: *aspect-guided encoder*, *aspect-reconstruction* and *aspect concatenated embedding*. The aspect-guided encoder is specially designed to guide the encoding of a sentence from scratch for conducting the aspect-specific feature selection and extraction at the very beginning stage. The aspect-reconstruction aims to guarantee that the aspect-specific information has been fully embedded in the sentence representation for more accurate predictions. The aspect concatenated embedding part is used to concatenate the aspect embedding and the generated sentence representation so as to make the final prediction.

### 2.1 Aspect-Guided Encoder

The aspect-guided encoder is the core module of AGDT, which consists of two key components: Aspect-guided GRU and Transition GRU (Cho et al., 2014).

**A-GRU:** Aspect-guided GRU (A-GRU) is a specially-designed unit for the ABSA tasks, which is an extension of the L-GRU proposed by Meng and Zhang (2019). In particular, we design an aspect-gate to select aspect-specific representations through controlling the transformation scale of token embeddings at each time step.

At time step $t$, the hidden state $\mathbf{h}_t$ is computed as follows:

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \widetilde{\mathbf{h}}_t \quad (1)$$

where $\odot$ represents element-wise product; $\mathbf{z}_t$ is the update gate (Cho et al., 2014); and $\widetilde{\mathbf{h}}_t$ is the candidate activation, which is computed as:

$$\begin{aligned}\widetilde{\mathbf{h}}_t = \tanh(\mathbf{g}_t \odot (\mathbf{W}_{xh}\mathbf{x}_t) + \mathbf{r}_t \odot (\mathbf{W}_{hh}\mathbf{h}_{t-1})) \\ + \mathbf{l}_t \odot \mathbf{H}_1(\mathbf{x}_t) + \mathbf{g}_t \odot \mathbf{H}_2(\mathbf{x}_t) \quad (2)\end{aligned}$$

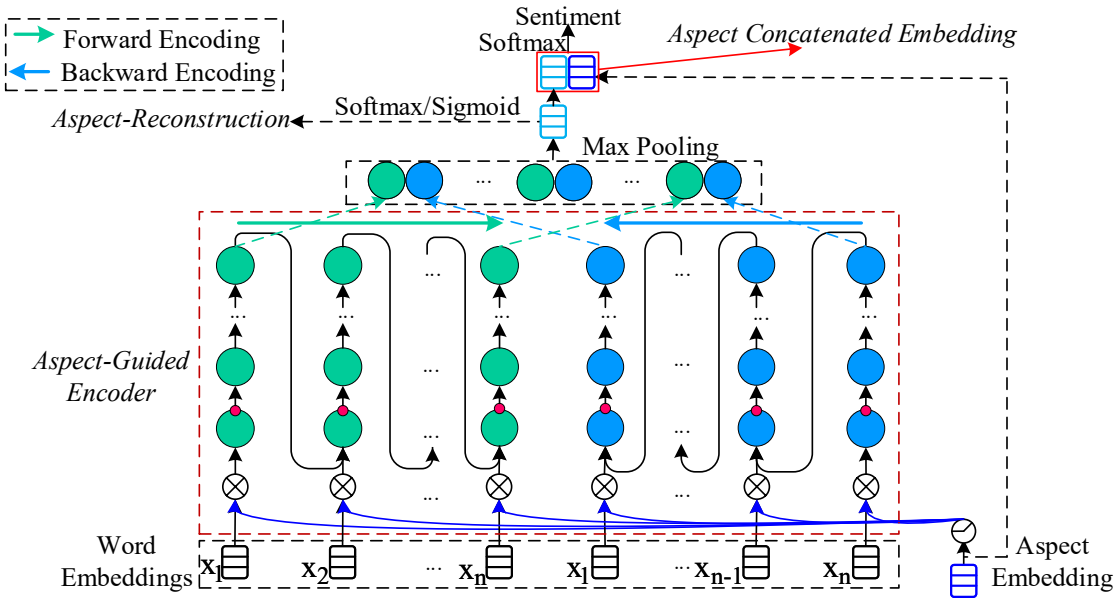


Figure 1: The overview of AGDT. The bottom right dark node (above the aspect embedding) is the aspect gate and other dark nodes (⊗) means element-wise multiply for the input token and the aspect gate. The *aspect-guided encoder* consists of a L-GRU (the circle frames fused with a small circle on above) at the bottom followed by several T-GRUs (the circle frames) from bottom to up.

where $\mathbf{g}_t$ denotes the aspect-gate; $\mathbf{x}_t$ represents the input word embedding at time step $t$; $\mathbf{r}_t$ is the reset gate (Cho et al., 2014); $\mathbf{H}_1(\mathbf{x}_t)$ and $\mathbf{H}_2(\mathbf{x}_t)$ are the linear transformation of the input $\mathbf{x}_t$, and $\mathbf{l}_t$ is the linear transformation gate for $\mathbf{x}_t$ (Meng and Zhang, 2019). $\mathbf{r}_t$, $\mathbf{z}_t$, $\mathbf{l}_t$, $\mathbf{g}_t$, $\mathbf{H}_1(\mathbf{x}_t)$ and $\mathbf{H}_2(\mathbf{x}_t)$ are computed as:

$$\mathbf{r}_t = \sigma(\mathbf{W}_{xr}\mathbf{x}_t + \mathbf{W}_{hr}\mathbf{h}_{t-1}) \quad (3)$$

$$\mathbf{z}_t = \sigma(\mathbf{W}_{xz}\mathbf{x}_t + \mathbf{W}_{hz}\mathbf{h}_{t-1}) \quad (4)$$

$$\mathbf{l}_t = \sigma(\mathbf{W}_{xl}\mathbf{x}_t + \mathbf{W}_{hl}\mathbf{h}_{t-1}) \quad (5)$$

$$\mathbf{g}_t = \text{relu}(\mathbf{W}_a\mathbf{a} + \mathbf{W}_{hg}\mathbf{h}_{t-1}) \quad (6)$$

$$\mathbf{H_1}(\mathbf{x_t}) = \mathbf{W}_1\mathbf{x}_t \quad (7)$$

$$\mathbf{H_2}(\mathbf{x_t}) = \mathbf{W}_2\mathbf{x}_t \quad (8)$$

where "$\mathbf{a}$" denotes the embedding of the given aspect, which is the same at each time step. The update gate $\mathbf{z}_t$ and reset gate $\mathbf{r}_t$ are the same as them in the conventional GRU.

In Eq. (2) $\sim$ (8), the aspect-gate $\mathbf{g}_t$ controls both nonlinear and linear transformations of the input $\mathbf{x}_t$ under the guidance of the given aspect at each time step. Besides, we also exploit a linear transformation gate $\mathbf{l}_t$ to control the linear transformation of the input, according to the current input $\mathbf{x}_t$ and previous hidden state $\mathbf{h}_{t-1}$, which has been proved powerful in the deep transition architecture (Meng and Zhang, 2019).

As a consequence, A-GRU can control both non-linear transformation and linear transformation for input $\mathbf{x}_t$ at each time step, with the guidance of the given aspect, i.e., A-GRU can guide the encoding of aspect-specific features and block the aspect-irrelevant information at the very beginning stage.

**T-GRU:** Transition GRU (T-GRU) (Pascanu et al., 2014) is a crucial component of deep transition block, which is a special case of GRU with only "state" as an input, namely its input embedding is zero embedding. As in Figure 1, a deep transition block consists of an A-GRU followed by several T-GRUs at each time step. For the current time step $t$, the output of one A-GRU/T-GRU is fed into the next T-GRU as the input. The output of the last T-GRU at time step $t$ is fed into A-GRU at the time step $t+1$. For a T-GRU, each hidden state at both time step $t$ and transition depth $i$ is computed as:

$$\mathbf{h}_t^i = (1 - \mathbf{z}_t^i) \odot \mathbf{h}_t^{i-1} + \mathbf{z}_t^i \odot \widetilde{\mathbf{h}}_t^i \quad (9)$$

$$\widetilde{\mathbf{h}}_t^i = \tanh(\mathbf{r}_t^i \odot (\mathbf{W}_h^i \mathbf{h}_t^{i-1})) \quad (10)$$

where the update gate $\mathbf{z}_t^i$ and the reset gate $\mathbf{r}_t^i$ are computed as:

$$\mathbf{z}_t^i = \sigma(\mathbf{W}_z^i \mathbf{h}_t^{i-1}) \quad (11)$$

$$\mathbf{r}_t^i = \sigma(\mathbf{W}_r^i \mathbf{h}_t^{i-1}) \quad (12)$$

The AGDT encoder is based on deep transition cells, where each cell is composed of one A-GRU

at the bottom, followed by several T-GRUs. Such AGDT model can encode the sentence representation with the guidance of aspect information by utilizing the specially designed architecture.

### 2.2 Aspect-Reconstruction

We propose an aspect-reconstruction approach to guarantee the aspect-specific information has been fully embedded in the sentence representation. Particularly, we devise two objectives for two subtasks in ABSA respectively. In terms of aspect-category sentiment analysis datasets, there are only several predefined aspect categories. While in aspect-term sentiment analysis datasets, the number of categories of term is more than one thousand. In a real-life scenario, the number of term is infinite, while the words that make up terms are limited. Thus we design different loss-functions for these two scenarios.

For the aspect-category sentiment analysis task, we aim to reconstruct the aspect according to the aspect-specific representation. It is a multi-class problem. We take the softmax cross-entropy as the loss function:

$$\mathcal{L}_c = min(-\sum_{i=0}^{C1} y_i^c \log(p_i^c)) \tag{13}$$

where C1 is the number of predefined aspects in the training example; $y_i^c$ is the ground-truth and $p_i^c$ is the estimated probability of a aspect.

For the aspect-term sentiment analysis task, we intend to reconstruct the aspect term (may consist of multiple words) according to the aspect-specific representation. It is a multi-label problem and thus the sigmoid cross-entropy is applied:

$$\begin{aligned}\mathcal{L}_t = min\{-\sum_{i=0}^{C2}[y_i^t \log(p_i^t) \\ + (1-y_i^t)\log(1-p_i^t)]\}\end{aligned} \tag{14}$$

where C2 denotes the number of words that constitute all terms in the training example, $y_i^t$ is the ground-truth and $p_i^t$ represents the predicted value of a word.

Our aspect-oriented objective consists of $\mathcal{L}_c$ and $\mathcal{L}_t$, which guarantee that the aspect-specific information has been fully embedded into the sentence representation.

### 2.3 Training Objective

The final loss function is as follows:

$$\boldsymbol{J} = min(\underline{-\sum_{i=0}^{C} y_i \log(p_i)} + \boldsymbol{\lambda}\mathcal{L}) \tag{15}$$

where the underlined part denotes the conventional loss function; C is the number of sentiment labels; $y_i$ is the ground-truth and $p_i$ represents the estimated probability of the sentiment label; $\mathcal{L}$ is the aspect-oriented objective, where Eq. 13 is for the aspect-category sentiment analysis task and Eq. 14 is for the aspect-term sentiment analysis task. And $\boldsymbol{\lambda}$ is the weight of $\mathcal{L}$.

As shown in Figure 1, we employ the aspect reconstruction approach to reconstruct the aspect (term), where "softmax" is for the aspect-category sentiment analysis task and "sigmoid" is for the aspect-term sentiment analysis task. Additionally, we concatenate the aspect embedding on the aspect-guided sentence representation to predict the sentiment polarity. Under that loss function (Eq. 15), the AGDT can produce aspect-specific sentence representations.

## 3 Experiments

### 3.1 Datasets and Metrics

**Data Preparation.** We conduct experiments on two datasets of the aspect-category based task and two datasets of the aspect-term based task. For these four datasets, we name the full dataset as "DS". In each "DS", there are some sentences like the example in Table 1, containing different sentiment labels, each of which associates with an aspect (term). For instance, Table 1 shows the customer's different attitude towards two aspects: "*food*" ("*The appetizers*") and "*service*". In order to measure whether a model can detect different sentiment polarities in one sentence towards different aspects, we extract a hard dataset from each "DS", named "HDS", in which each sentence only has different sentiment labels associated with different aspects. When processing the original sentence $s$ that has multiple aspects $a_1, a_2, ..., a_n$ and corresponding sentiment labels $l_1, l_2, ..., l_n$ ($n$ is the number of aspects or terms in a sentence), the sentence will be expanded into (s, $a_1$, $l_1$), (s, $a_2$, $l_2$), ..., (s, $a_n$, $l_n$) in each dataset (Ruder et al., 2016b,a; Xue and Li, 2018), i.e, there will be $n$ duplicated sentences associated with different aspects and labels.

| | | Positive | | Negative | | Neutral | | Conflict | | Total | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | DS | HDS | DS | HDS | DS | HDS | DS | HDS | DS | HDS |
| **Restaurant-14** | Train | 2,179 | 139 | 839 | 136 | 500 | 50 | 195 | 40 | 3,713 | 365 |
| | Test | 657 | 32 | 222 | 26 | 94 | 12 | 52 | 19 | 1,025 | 89 |
| **Restaurant-Large** | Train | 2,710 | 182 | 1,198 | 178 | 757 | 107 | - | - | 4,665 | 467 |
| | Test | 1,505 | 92 | 680 | 81 | 241 | 61 | - | - | 2,426 | 234 |

Table 2: Statistics of datasets for the aspect-category sentiment analysis task.

| | | Positive | | Negative | | Neutral | | Conflict | | Total | | NC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | DS | HDS | DS | HDS | DS | HDS | DS | HDS | DS | HDS | DS |
| **Restaurant** | Train | 2,164 | 379 | 805 | 323 | 633 | 293 | 91 | 43 | 3,693 | 1,038 | 3,602 |
| | Test | 728 | 92 | 196 | 62 | 196 | 83 | 14 | 8 | 1,134 | 245 | 1,120 |
| **Laptop** | Train | 987 | 159 | 866 | 147 | 460 | 173 | 45 | 17 | 2,358 | 496 | 2,313 |
| | Test | 341 | 31 | 128 | 25 | 169 | 49 | 16 | 3 | 654 | 108 | 638 |

Table 3: Statistics of datasets for the aspect-term sentiment analysis task. The '**NC**' indicates No "*Conflict*" label, which is just removed the "*conflict*" label and is prepared for the three-class experiment.

**Aspect-Category Sentiment Analysis.** For comparison, we follow Xue and Li (2018) and use the restaurant reviews dataset of SemEval 2014 ("restaurant-14") Task 4 (Pontiki et al., 2014) to evaluate our AGDT model. The dataset contains five predefined aspects and four sentiment labels. A large dataset ("restaurant-large") involves restaurant reviews of three years, i.e., 2014 $\sim$ 2016 (Pontiki et al., 2014). There are eight predefined aspects and three labels in that dataset. When creating the "restaurant-large" dataset, we follow the same procedure as in Xue and Li (2018). Statistics of datasets are shown in Table 2.

**Aspect-Term Sentiment Analysis.** We use the restaurant and laptop review datasets of SemEval 2014 Task 4 (Pontiki et al., 2014) to evaluate our model. Both datasets contain four sentiment labels. Meanwhile, we also conduct a three-class experiment, in order to compare with some work (Wang et al., 2016; Ma et al., 2017; Li et al., 2018a) which removed "conflict" labels. Statistics of both datasets are shown in Table 3.

**Metrics.** The evaluation metrics are accuracy. All instances are shown in Table 2 and Table 3. Each experiment is repeated five times. The mean and the standard deviation are reported.

### 3.2 Implementation Details

We use the pre-trained 300d Glove[2] embeddings (Pennington et al., 2014) to initialize word embeddings, which is fixed in all models. For out-of-vocabulary words, we randomly sample their embeddings by the uniform distribution $U(-0.25, 0.25)$. Following Tang et al. (2016b); Chen et al. (2017); Liu and Zhang (2017), we take the averaged word embedding as the aspect representation for multi-word aspect terms. The transition depth of deep transition model is 4 (see Section 3.4). The hidden size is set to 300. We set the dropout rate (Srivastava et al., 2014) to 0.5 for input token embeddings and 0.3 for hidden states. All models are optimized using Adam optimizer (Kingma and Ba, 2014) with gradient clipping equals to 5 (Pascanu et al., 2012). The initial learning rate is set to 0.01 and the batch size is set to 4096 at the token level. The weight of the reconstruction loss $\boldsymbol{\lambda}$ in Eq. 15 is fine-tuned (see Section 3.4) and respectively set to 0.4, 0.4, 0.2 and 0.5 for four datasets. The neural model is implemented in Tensorflow (Abadi et al., 2016) and all computations are done on a NVIDIA Tesla M40 GPU.

### 3.3 Baselines

To comprehensively evaluate our AGDT, we compare the AGDT with several competitive models.

**ATAE-LSTM.** It is an attention-based LSTM model. It appends the given aspect embedding with each word embedding, and then the concatenated embedding is taken as the input of LSTM. The output of LSTM is appended aspect embedding again. Furthermore, attention is applied to extract features for final predictions.

[2]Pre-trained Glove embeddings can be obtained from http://nlp.stanford.edu/projects/glove/

| Models | Restaurant-14 | | Restaurant-Large | |
|---|---|---|---|---|
| | DS | HDS | DS | HDS |
| **ATAE-LSTM**(Wang et al., 2016)* | 78.29±0.68 | 45.62±0.90 | 83.91±0.49 | 66.32±2.28 |
| **CNN**(Kim, 2014)* | 79.47±0.32 | 44.94±0.01 | 84.28±0.15 | 50.43±0.38 |
| **GCAE**(Xue and Li, 2018)* | 79.35±0.34 | 50.55±1.83 | 85.92±0.27 | 70.75±1.19 |
| **AGDT** | **81.78**±0.31 | **62.02**±1.31 | **87.55**±0.17 | **75.73**±0.50 |

Table 4: The accuracy of the aspect-category sentiment analysis task. '*' refers to citing from GCAE (Xue and Li, 2018).

| Models | Restaurant | | Laptop | |
|---|---|---|---|---|
| | DS | HDS | DS | HDS |
| **TD-LSTM**(Tang et al., 2016a)* | 73.44±1.17 | 56.48±2.46 | 62.23±0.92 | 46.11±1.89 |
| **ATAE-LSTM**(Wang et al., 2016)* | 73.74±3.01 | 50.98±2.27 | 64.38±4.52 | 40.39±1.30 |
| **IAN**(Ma et al., 2017)* | 76.34±0.27 | 55.16±1.97 | 68.49±0.57 | 44.51±0.48 |
| **RAM**(Chen et al., 2017)* | 76.97±0.64 | 55.85±1.60 | 68.48±0.85 | 45.37±2.03 |
| **GCAE**(Xue and Li, 2018)* | 77.28±0.32 | 56.73±0.56 | 69.14±0.32 | 47.06±2.45 |
| **AGDT** | **78.85**±0.45 | **60.33**±1.01 | **71.50**±0.85 | **51.30**±1.26 |

Table 5: The accuracy of the aspect-term sentiment analysis task. '*' refers to citing from GCAE (Xue and Li, 2018).

**CNN.** This model focuses on extracting n-gram features to generate sentence representation for the sentiment classification.

**TD-LSTM.** This model uses two LSTMs to capture the left and right context of the term to generate target-dependent representations for the sentiment prediction.

**IAN.** This model employs two LSTMs and interactive attention mechanism to learn representations of the sentence and the aspect, and concatenates them for the sentiment prediction.

**RAM.** This model applies multiple attentions and memory networks to produce the sentence representation.

**GCAE.** It uses CNNs to extract features and then employs two Gated Tanh-Relu units to selectively output the sentiment information flow towards the aspect for predicting sentiment labels.

### 3.4 Main Results and Analysis

**Aspect-Category Sentiment Analysis Task**

We present the overall performance of our model and baseline models in Table 4. Results show that our AGDT outperforms all baseline models on both "restaurant-14" and "restaurant-large" datasets. ATAE-LSTM employs an aspect-weakly associative encoder to generate the aspect-specific sentence representation by simply concatenating the aspect, which is insufficient to exploit the given aspect. Although GCAE incorporates the gating mechanism to control the sentiment information flow according to the given aspect, the information flow is generated by an aspect-independent encoder. Compared with GCAE, our AGDT improves the performance by 2.4% and 1.6% in the "DS" part of the two dataset, respectively. These results demonstrate that our AGDT can sufficiently exploit the given aspect to generate the aspect-guided sentence representation, and thus conduct accurate sentiment prediction. Our model benefits from the following aspects. First, our AGDT utilizes an aspect-guided encoder, which leverages the given aspect to guide the sentence encoding from scratch and generates the aspect-guided representation. Second, the AGDT guarantees that the aspect-specific information has been fully embedded in the sentence representation via reconstructing the given aspect. Third, the given aspect embedding is concatenated on the aspect-guided sentence representation for final predictions.

The "HDS", which is designed to measure whether a model can detect different sentiment polarities in a sentence, consists of replicated sentences with different sentiments towards multiple aspects. Our AGDT surpasses GCAE by a very large margin (**+11.4%** and **+4.9%** respectively) on both datasets. This indicates that the given aspect information is very pivotal to the accurate sentiment prediction, especially when the sentence

has different sentiment labels, which is consistent with existing work (Jiang et al., 2011; Ma et al., 2017; Wang et al., 2018). Those results demonstrate the effectiveness of our model and suggest that our AGDT has better ability to distinguish the different sentiments of multiple aspects compared to GCAE.

### Aspect-Term Sentiment Analysis Task

As shown in Table 5, our AGDT consistently outperforms all compared methods on both domains. In this task, TD-LSTM and ATAE-LSTM use a aspect-weakly associative encoder. IAN, RAM and GCAE employ an aspect-independent encoder. In the "DS" part, our AGDT model surpasses all baseline models, which shows that the inclusion of A-GRU (aspect-guided encoder), aspect-reconstruction and aspect concatenated embedding has an overall positive impact on the classification process.

In the "HDS" part, the AGDT model obtains +3.6% higher accuracy than GCAE on the restaurant domain and +4.2% higher accuracy on the laptop domain, which shows that our AGDT has stronger ability for the multi-sentiment problem against GCAE. These results further demonstrate that our model works well across tasks and datasets.

### Ablation Study

We conduct ablation experiments to investigate the impacts of each part in AGDT, where the GRU is stacked with 4 layers. Here "AC" represents aspect concatenated embedding , "AG" stands for A-GRU (Eq. (1) $\sim$ (8)) and "AR" denotes the aspect-reconstruction (Eq. (13) $\sim$ (15)).

From Table 6 and Table 7, we can conclude:

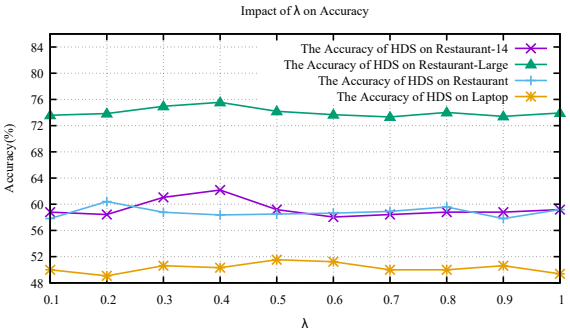


Figure 2: The impact of $\lambda$ w.r.t. accuracy on "HDS".

1). Deep Transition (DT) achieves superior performances than GRU, which is consistent with previous work (Miceli Barone et al., 2017; Meng and Zhang, 2019) (② vs. ①).

2). Utilizing "AG" to guide encoding aspect-related features from scratch has a significant impact for highly competitive results and particularly in the "HDS" part, which demonstrates that it has the stronger ability to identify different sentiment polarities towards different aspects. (③ vs. ②).

3). Aspect concatenated embedding can promote the accuracy to a degree (④ vs. ③).

4). The aspect-reconstruction approach ("AR") substantially improves the performance, especially in the "HDS" part (⑤ vs. ④).

5). the results in ⑥ show that all modules have an overall positive impact on the sentiment classification.

| | AC | AG | AR | Rest-14 | | Rest-Large | | |
|---|---|---|---|---|---|---|---|---|
| | | | | DS | HDS | DS | HDS | |
| **GRU** | √ | × | × | 80.90 | 53.93 | 86.75 | 68.46 | ① |
| **DT** | √ | × | × | 81.74 | 56.63 | 87.54 | 72.39 | ② |
| | √ | √ | × | 81.88 | 60.42 | **87.72** | 74.81 | ③ |
| | × | √ | × | **81.95** | 59.33 | 87.68 | 74.44 | ④ |
| | × | √ | √ | 81.83 | 61.35 | 87.34 | 75.56 | ⑤ |
| | √ | √ | √ | 81.78 | **62.02** | 87.55 | **75.73** | ⑥ |

Table 6: Ablation study of the AGDT on the aspect-category sentiment analysis task. Here "AC", "AG" and "AR" represent aspect concatenated embedding, A-GRU and aspect-reconstruction, respectively, '√' and '×' denotes whether to apply the operation. 'Rest-14': Restaurant-14,'Rest-Large': Restaurant-Large.

| | AC | AG | AR | Restaurant | | Laptop | | |
|---|---|---|---|---|---|---|---|---|
| | | | | DS | HDS | DS | HDS | |
| **GRU** | √ | × | × | 78.31 | 55.92 | 70.21 | 46.48 | ① |
| **DT** | √ | × | × | 78.36 | 56.24 | 71.07 | 47.59 | ② |
| | √ | √ | × | 78.77 | 60.14 | 71.42 | 50.83 | ③ |
| | × | √ | × | 78.55 | 60.08 | 71.38 | 50.74 | ④ |
| | × | √ | √ | 78.59 | 60.16 | 71.47 | 51.11 | ⑤ |
| | √ | √ | √ | **78.85** | **60.33** | **71.50** | **51.30** | ⑥ |

Table 7: Ablation study of the AGDT on the aspect-term sentiment analysis task.

### Impact of Model Depth

We have demonstrated the effectiveness of the AGDT. Here, we investigate the impact of model depth of AGDT, varying the depth from 1 to 6. Table 8 shows the change of accuracy on the test

| | Depth | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| $\mathbb{D}_1$ | DS | 81.12 | 81.45 | 81.52 | **81.78** | 81.07 | 80.68 |
| | HDS | 55.73 | 57.08 | 60.67 | **62.02** | 59.10 | 58.65 |
| $\mathbb{D}_2$ | DS | 87.20 | 87.47 | 87.53 | **87.55** | 87.11 | 87.21 |
| | HDS | 73.93 | 74.27 | **76.07** | 75.73 | 75.56 | 74.27 |
| $\mathbb{D}_3$ | DS | 78.18 | 77.94 | 78.69 | **78.85** | 78.40 | 77.88 |
| | HDS | 59.35 | 58.94 | 59.43 | **60.33** | 59.27 | 57.80 |
| $\mathbb{D}_4$ | DS | 71.13 | 71.10 | **71.62** | 71.50 | 71.16 | 70.86 |
| | HDS | 49.44 | 50.00 | 50.56 | **51.30** | 49.81 | 49.63 |

Table 8: The accuracy of model depth on the four datasets. '$\mathbb{D}_1$': Restaurant-14, '$\mathbb{D}_2$': Restaurant-Large, '$\mathbb{D}_3$': Restaurant, '$\mathbb{D}_4$': Laptop.

| | Rest-14 | Rest-Large | Rest. | Laptop |
|---|---|---|---|---|
| DS | 99.55 | 99.80 | 76.21 | 70.92 |

Table 9: The accuracy of aspect reconstruction on the full test set. 'Rest-14': Restaurant-14, 'Rest-Large': Restaurant-Large, 'Rest.': Restaurant.

sets as depth increases. We find that the best results can be obtained when the depth is equal to 4 at most case, and further depth do not provide considerable performance improvement.

### Effectiveness of Aspect-reconstruction Approach

Here, we investigate how well the AGDT can reconstruct the aspect information. For the aspect-term reconstruction, we count the construction is correct when all words of the term are reconstructed. Table 9 shows all results on four test datasets, which shows the effectiveness of aspect-reconstruction approach again.

### Impact of Loss Weight $\lambda$

We randomly sample a temporary development set from the "HDS" part of the training set to choose the lambda for each dataset. And we investigate the impact of $\lambda$ for aspect-oriented objectives. Specifically, $\lambda$ is increased from 0.1 to 1.0. Figure 2 illustrates all results on four "HDS" datasets, which show that reconstructing the given aspect can enhance aspect-specific sentiment features and thus obtain better performances.

### Comparison on Three-Class for the Aspect-Term Sentiment Analysis Task

We also conduct a three-class experiment to compare our AGDT with previous models, i.e., IARM, TNet, VAE, PBAN, AOA and MGAN, in Table 10.

| Models | Rest. | Laptop |
|---|---|---|
| **IARM**(Majumder et al., 2018)* | 80.00 | 73.80 |
| **TNet**(Li et al., 2018a)* | 80.79 | **76.54** |
| **VAE**(Xu and Tan, 2018)* | 81.10 | 75.34 |
| **PBAN**(Gu et al., 2018)* | 81.16 | 74.12 |
| **AOA**(Huang et al., 2018)* | 81.20 | 74.50 |
| **MGAN**(Fan et al., 2018)* | 81.25 | 75.39 |
| **DAuM**(Zhu and Qian, 2018)* | 82.32 | 74.45 |
| **AGDT** | **82.95** | 75.86 |

Table 10: The three-class accuracy of the aspect-term sentiment analysis task on SemEval 2014. '*' refers to citing from the original paper. 'Rest.': Restaurant.

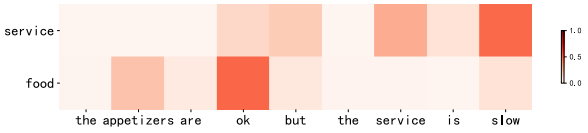


Figure 3: The output of A-GRU.

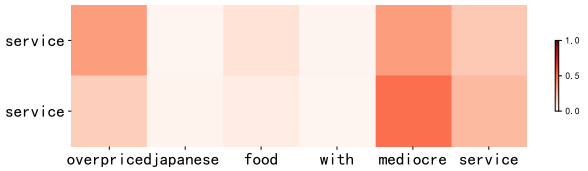


Figure 4: The above is the output of A-GRU. The bottom is the output after reconstructing the given aspect.

These previous models are based on an aspect-independent (weakly associative) encoder to generate sentence representations. Results on all domains suggest that our AGDT substantially outperforms most competitive models, except for the TNet on the laptop dataset. The reason may be TNet incorporates additional features (e.g., position features, local ngrams and word-level features) compared to ours (only word-level features).

## 4 Analysis and Discussion

**Case Study and Visualization.** To give an intuitive understanding of how the proposed A-GRU works from scratch with different aspects, we take a review sentence as an example. As the example "*the appetizers are ok, but the service is slow.*" shown in Table 1, it has different sentiment labels towards different aspects. The color depth denotes the semantic relatedness level between the given aspect and each word. More depth means stronger relation to the given aspect.

Figure 3 shows that the A-GRU can effectively guide encoding the aspect-related features with

the given aspect and identify corresponding sentiment. In another case, "*overpriced Japanese food with mediocre service.*", there are two extremely strong sentiment words. As the above of Figure 4 shows, our A-GRU generates almost the same weight to the word "*overpriced*" and "*mediocre*". The bottom of Figure 4 shows that reconstructing the given aspect can effectively enhance aspect-specific sentiment features and produce correct sentiment predictions.

**Error Analysis.** We further investigate the errors from AGDT, which can be roughly divided into 3 types. **1)** The decision boundary among the sentiment polarity is unclear, even the annotators can not sure what sentiment orientation over the given aspect in the sentence. **2)** The "conflict/neutral" instances are extremely easily misclassified as "positive" or "negative", due to the imbalanced label distribution in training corpus[3]. **3)** The polarity of complex instances is hard to predict, such as the sentence that express subtle emotions, which are hardly effectively captured, or containing negation words (e.g., *never*, *less* and *not*), which easily affect the sentiment polarity.

## 5 Related Work

**Sentiment Analysis.** There are kinds of sentiment analysis tasks, such as document-level (Thongtan and Phienthrakul, 2019), sentence-level[4] (Zhang and Zhang, 2019; Zhang et al., 2019), aspect-level (Pontiki et al., 2014; Wang et al., 2019a) and multimodal (Chen et al., 2018; Akhtar et al., 2019) sentiment analysis. For the aspect-level sentiment analysis, previous work typically apply attention mechanism (Luong et al., 2015) combining with memory network (Weston et al., 2014) or gating units to solve this task (Tang et al., 2016b; He et al., 2018a; Huang and Carley, 2018; Xue and Li, 2018; Duan et al., 2018; Tang et al., 2019; Yang et al., 2019; Bao et al., 2019), where an aspect-independent encoder is used to generate the sentence representation. In addition, some work leverage the aspect-weakly associative encoder to generate aspect-specific sentence representation (Tang et al., 2016a; Wang et al., 2016; Majumder et al., 2018). All of these methods make insufficient use of the given aspect information. There are also some work which jointly extract the aspect term (and opinion term) and predict its sentiment polarity (Schmitt et al., 2018; Li et al., 2018b; Ma et al., 2018; Angelidis and Lapata, 2018; He et al., 2019; Luo et al., 2019; Hu et al., 2019; Dai and Song, 2019; Wang et al., 2019b). In this paper, we focus on the latter problem and leave aspect extraction (Shu et al., 2017) to future work. And some work (Sun et al., 2019; Xu et al., 2019; He et al., 2018b; Xu and Tan, 2018; Chen and Qian, 2019; He et al., 2019) employ the well-known BERT (Devlin et al., 2018) or document-level corpora to enhance ABSA tasks, which will be considered in our future work to further improve the performance.

**Deep Transition.** Deep transition has been proved its superiority in language modeling (Pascanu et al., 2014) and machine translation (Miceli Barone et al., 2017; Meng and Zhang, 2019). We follow the deep transition architecture in Meng and Zhang (2019) and extend it by incorporating a novel A-GRU for ABSA tasks.

## 6 Conclusions

In this paper, we propose a novel aspect-guided encoder (AGDT) for ABSA tasks, based on a deep transition architecture. Our AGDT can guide the sentence encoding from scratch for the aspect-specific feature selection and extraction. Furthermore, we design an aspect-reconstruction approach to enforce AGDT to reconstruct the given aspect with the generated sentence representation. Empirical studies on four datasets suggest that the AGDT outperforms existing state-of-the-art models substantially on both aspect-category sentiment analysis task and aspect-term sentiment analysis task of ABSA without additional features.

## Acknowledgments

We sincerely thank the anonymous reviewers for their thorough reviewing and insightful suggestions. Liang, Xu, and Chen are supported by the National Natural Science Foundation of China (Contract 61370130, 61976015, 61473294 and 61876198), and the Beijing Municipal Natural Science Foundation (Contract 4172047), and the International Science and Technology Cooperation Program of the Ministry of Science and Technology (K11F100010).

[3]More details can be seen in the dataset or see here: http://alt.qcri.org/semeval2014/

[4]https://nlp.stanford.edu/sentiment/

## References


Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. Tensorflow: A system for large-scale machine learning. In *OSDI*, pages 265–283.

Md Shad Akhtar, Dushyant Chauhan, Deepanway Ghosal, Soujanya Poria, Asif Ekbal, and Pushpak Bhattacharyya. 2019. Multi-task learning for multi-modal emotion recognition and sentiment analysis. In *ACL*, pages 370–379.

Stefanos Angelidis and Mirella Lapata. 2018. Summarizing opinions: Aspect extraction meets sentiment prediction and they are both weakly supervised. *CoRR*, abs/1808.08858.

Lingxian Bao, Patrik Lambert, and Toni Badia. 2019. Attention and lexicon regularized LSTM for aspect-based sentiment analysis. In *ACL*, pages 253–259.

F. Chen, R. Ji, J. Su, D. Cao, and Y. Gao. 2018. Predicting microblog sentiments via weakly supervised multimodal deep learning. *IEEE Transactions on Multimedia*, 20(4):997–1007.

Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. 2017. Recurrent attention network on memory for aspect sentiment analysis. In *EMNLP*, pages 452–461.

Zhuang Chen and Tieyun Qian. 2019. Transfer capsule network for aspect level sentiment classification. In *ACL*, pages 547–556.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP*, pages 1724–1734.

Hongliang Dai and Yangqiu Song. 2019. Neural aspect and opinion term extraction with mined rules as weak supervision. In *ACL*, pages 5268–5277.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Junwen Duan, Xiao Ding, and Ting Liu. 2018. Learning sentence representations over tree structures for target-dependent classification. In *ACL*, pages 551–560.

Feifan Fan, Yansong Feng, and Dongyan Zhao. 2018. Multi-grained attention network for aspect-level sentiment classification. In *EMNLP*, pages 3433–3442.

Shuqin Gu, Lipeng Zhang, Yuexian Hou, and Yin Song. 2018. A position-aware bidirectional attention network for aspect-level sentiment analysis. In *COLING*, pages 774–784.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2018a. Effective attention modeling for aspect-level sentiment classification. In *COLING*, pages 1121–1131.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2018b. Exploiting document knowledge for aspect-level sentiment classification. In *ACL*, pages 579–585.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2019. An interactive multi-task learning network for end-to-end aspect-based sentiment analysis. In *ACL*, pages 504–515.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Minghao Hu, Yuxing Peng, Zhen Huang, Dongsheng Li, and Yiwei Lv. 2019. Open-domain targeted sentiment analysis via span-based extraction and classification. In *ACL*, pages 537–546.

Binxuan Huang and Kathleen Carley. 2018. Parameterized convolutional neural networks for aspect level sentiment classification. In *EMNLP*, pages 1091–1096.

Binxuan Huang, Yanglan Ou, and Kathleen M. Carley. 2018. Aspect level sentiment classification with attention-over-attention neural networks. *CoRR*, abs/1804.06536.

Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. 2011. Target-dependent twitter sentiment classification. In *ACL*, pages 151–160.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Xin Li, Lidong Bing, Wai Lam, and Bei Shi. 2018a. Transformation networks for target-oriented sentiment classification. In *ACL*, pages 946–956.

Xin Li, Lidong Bing, Piji Li, and Wai Lam. 2018b. A unified model for opinion target extraction and target sentiment prediction. *CoRR*, abs/1811.05082.

Bin Liang, Jiachen Du, Ruifeng Xu, Binyang Li, and Hejiao Huang. 2019. Context-aware embedding for targeted aspect-based sentiment analysis. In *ACL*, pages 4678–4683.

Jiangming Liu and Yue Zhang. 2017. Attention modeling for targeted sentiment. In *ACL*, pages 572–577.

Huaishao Luo, Tianrui Li, Bing Liu, and Junbo Zhang. 2019. DOER: Dual cross-shared RNN for aspect term-polarity co-extraction. In *ACL*, pages 591–601.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*, pages 1412–1421.

Dehong Ma, Sujian Li, and Houfeng Wang. 2018. Joint learning for targeted sentiment analysis. In *EMNLP*, pages 4737–4742.

Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2017. Interactive attention networks for aspect-level sentiment classification. In *IJCAI*, pages 4068–4074.

Navonil Majumder, Soujanya Poria, Alexander Gelbukh, Md Shad Akhtar, Erik Cambria, and Asif Ekbal. 2018. Iarm: Inter-aspect relation modeling with memory networks in aspect-based sentiment analysis. In *EMNLP*, pages 3402–3411.

Fandong Meng and Jinchao Zhang. 2019. DTMT: A novel deep transition architecture for neural machine translation. *CoRR*, abs/1812.07807.

Antonio Valerio Miceli Barone, Jindřich Helcl, Rico Sennrich, Barry Haddow, and Alexandra Birch. 2017. Deep architectures for neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 99–107.

Razvan Pascanu, aglar Glehre, Kyunghyun Cho, and Yoshua Bengio. 2014. How to construct deep recurrent neural networks. *CoRR*, abs/1312.6026.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2012. Understanding the exploding gradient problem. *CoRR*, abs/1211.5063.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *SemEval*, pages 27–35.

Sebastian Ruder, Parsa Ghaffari, and John G. Breslin. 2016a. A hierarchical model of reviews for aspect-based sentiment analysis. In *EMNLP*, pages 999–1005.

Sebastian Ruder, Parsa Ghaffari, and John G. Breslin. 2016b. Insight-1 at semeval-2016 task 5: Deep learning for multilingual aspect-based sentiment analysis. In *SemEval*, pages 330–336.

Martin Schmitt, Simon Steinheber, Konrad Schreiber, and Benjamin Roth. 2018. Joint aspect and polarity classification for aspect-based sentiment analysis with end-to-end neural networks. In *EMNLP*, pages 1109–1114.

Lei Shu, Hu Xu, and Bing Liu. 2017. Lifelong learning CRF for supervised aspect extraction. *CoRR*, abs/1705.00251.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958.

Chi Sun, Luyao Huang, and Xipeng Qiu. 2019. Utilizing BERT for aspect-based sentiment analysis via constructing auxiliary sentence. *CoRR*, abs/1903.09588.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2016a. Effective lstms for target-dependent sentiment classification. In *COLING*, pages 3298–3307.

Duyu Tang, Bing Qin, and Ting Liu. 2016b. Aspect level sentiment classification with deep memory network. In *EMNLP*, pages 214–224.

Jialong Tang, Ziyao Lu, Jinsong Su, Yubin Ge, Linfeng Song, Le Sun, and Jiebo Luo. 2019. Progressive self-supervised attention learning for aspect-level sentiment analysis. In *ACL*, pages 557–566.

Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2017. Learning to attend via word-aspect associative fusion for aspect-based sentiment analysis. *CoRR*, abs/1712.05403.

Tan Thongtan and Tanasanee Phienthrakul. 2019. Sentiment classification using document embeddings trained with cosine similarity. In *ACL*, pages 407–414.

Jingjing Wang, Changlong Sun, Shoushan Li, Xiaozhong Liu, Luo Si, Min Zhang, and Guodong Zhou. 2019a. Aspect sentiment classification towards question-answering with reinforced bidirectional attention network. In *ACL*, pages 3548–3557.

Shuai Wang, Sahisnu Mazumder, Bing Liu, Mianwei Zhou, and Yi Chang. 2018. Target-sensitive memory networks for aspect sentiment classification. In *ACL*, pages 957–967.

Yequan Wang, Minlie Huang, xiaoyan zhu, and Li Zhao. 2016. Attention-based lstm for aspect-level sentiment classification. In *EMNLP*, pages 606–615.

Yequan Wang, Aixin Sun, Minlie Huang, and Xiaoyan Zhu. 2019b. Aspect-level sentiment analysis using as-capsules. In *WWW*, pages 2033–2044.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *CoRR*, abs/1410.3916.

Bowen Xing, Lejian Liao, Dandan Song, Jingang Wang, Fuzheng Zhang, Zhongyuan Wang, and Heyan Huang. 2019. Earlier attention? aspect-aware LSTM for aspect sentiment analysis. *CoRR*, abs/1905.07719.

Hu Xu, Bing Liu, Lei Shu, and Philip S. Yu. 2019. BERT post-training for review reading comprehension and aspect-based sentiment analysis. *CoRR*, abs/1904.02232.

Weidi Xu and Ying Tan. 2018. Semi-supervised target-level sentiment analysis via variational autoencoder. *CoRR*, abs/1810.10437.

Wei Xue and Tao Li. 2018. Aspect based sentiment analysis with gated convolutional networks. In *ACL*, pages 2514–2523.

Chao Yang, Hefeng Zhang, Bin Jiang, and Keqin Li. 2019. Aspect-based sentiment analysis with alternating coattention networks. *Information Processing and Management*, 56:463–478.

Liwen Zhang, Kewei Tu, and Yue Zhang. 2019. Latent variable sentiment grammar. In *ACL*, pages 4642–4651.

Yuan Zhang and Yue Zhang. 2019. Tree communication models for sentiment analysis. In *ACL*, pages 3518–3527.

Peisong Zhu and Tieyun Qian. 2018. Enhanced aspect level sentiment classification with auxiliary memory. In *COLING*, pages 1077–1087.

# Human-Like Decision Making: Document-level Aspect Sentiment Classification via Hierarchical Reinforcement Learning

**Jingjing Wang**[1], **Changlong Sun**[2], **Shoushan Li**[1,*], **Jiancheng Wang**[1], **Luo Si**[2], **Min Zhang**[1], **Xiaozhong Liu**[2], **Guodong Zhou**[1]
[1]School of Computer Science and Technology, Soochow University, China
[2]Alibaba Group, China
{djingwang, lishoushan, minzhang, gdzhou}@suda.edu.cn,
jiancheng.wang@qq.com, {changlong.scl, xiaozhong.lxz, luo.si}@alibaba-inc.com

## Abstract

Recently, neural networks have shown promising results on Document-level Aspect Sentiment Classification (DASC). However, these approaches often offer little transparency w.r.t. their inner working mechanisms and lack interpretability. In this paper, to simulating the steps of analyzing aspect sentiment in a document by human beings, we propose a new Hierarchical Reinforcement Learning (HRL) approach to DASC. This approach incorporates clause selection and word selection strategies to tackle the data noise problem in the task of DASC. First, a high-level policy is proposed to select aspect-relevant clauses and discard noisy clauses. Then, a low-level policy is proposed to select sentiment-relevant words and discard noisy words inside the selected clauses. Finally, a sentiment rating predictor is designed to provide reward signals to guide both clause and word selection. Experimental results demonstrate the impressive effectiveness of the proposed approach to DASC over the state-of-the-art baselines.

## 1 Introduction

Document-level Aspect Sentiment Classification (DASC) is a fine-grained sentiment classification task in the field of sentiment analysis (Pang and Lee, 2007; Li et al., 2010). This task aims to predict the sentiment rating for each given aspect mentioned in a document-level review. For instance, Figure 1 shows a review document with four given aspects of a hotel (i.e., *location*, *room*, *value*, *service*). The goal of DASC is to predict the rating score towards each aspect by analyzing the whole document. In the last decade, this task has been drawing more and more interests of researchers in the Natural Language Processing community (Titov and McDonald, 2008; Yin et al., 2017; Li et al., 2018). In previous studies, neural models have shown to be effective for performance improvement on DASC. Despite the advantages, these complex neural network approaches often offer little transparency w.r.t. their inner working mechanisms and suffer from the lack of interpretability. However, clearly understanding where and how such a model makes such a decision is rather important for developing real-world applications (Liu et al., 2018; Marcus, 2018).

**Review Document**

[This hotel is close to railway station]$_{\text{Clause1}}$ [and very convenient to eat around]$_{\text{Clause2}}$ [but room of Hilton is a little uncomfortable.]$_{\text{Clause3}}$ [I'm often nitpicking for room decoration.]$_{\text{Clause4}}$ [Besides, the price is very expensive ]$_{\text{Clause5}}$ [although the staff service is professional.]$_{\text{Clause6}}$

**Rating of Each Aspect**

- ***location***: ★★★★★ (5)
- ***room***: ★★★☆☆ (3)
- ***value***: ★☆☆☆☆ (1)
- ***service***: ★★★★☆ (4)

Figure 1: An example of a review document, where clauses and words with different colors refer to different aspects.

As human beings, if asked to evaluate the sentiment rating for a specific aspect in a document, we often perform sentiment prediction in two steps. First, we select some **aspect-relevant snippets** (e.g., sentences/clauses) inside the document. Second, we select some **sentiment-relevant words** (e.g., sentiment words) inside these snippets to make a rating decision. For instance, for aspect *location* in Figure 1, we first select the aspect-relevant clauses, i.e., Clause1 and Clause2, and then select sentiment-relevant words, i.e., "*close*" and "*very convenient*" inside the two clauses, for making the rating decision (5 stars).

Inspired by the above cognitive process of human beings, one ideal and interpretable solution for DASC is to select aspect-relevant clauses and sentiment-relevant words, discarding those noisy parts of a document for decision making. In this

*Corresponding author



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5581–5590, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

solution, two major challenges exist which are illustrated as follows.

The first challenge is how to select aspect-relevant clauses and discard those irrelevant and noisy clauses. For instance, for aspect *location*, Clause5 mentioning another aspect *value* (only 1 star) may induce the noise and should be discarded, because the noise can provide wrong signals to mislead the model into assigning very low sentiment rating to aspect *location*. One possible way to alleviate this noisy problem is to leverage the soft-attention mechanism as proposed in Li et al. (2018) and Wang et al. (2018). However, this soft-attention mechanism has the limitation that the *softmax* function always assigns small but non-zero probabilities to noisy clauses, which will weaken the attention given to the few truly significant clauses for a particular aspect. Therefore, a well-behaved approach should discard noisy clauses for a specific aspect during model training.

The second challenge is how to select sentiment-relevant words and discard those irrelevant and noisy words. For instance, for aspect *location*, words "*this*", "*is*" in Clause1 are noisy words and should be discarded since they make no contribution to implying the sentiment rating. One possible way to alleviate this problem is to also leverage the soft-attention mechanism as proposed in Li et al. (2018). However, this soft-attention mechanism may induce additional noise and lack interpretability because it tends to assign higher weights to some domain-specific words rather than real sentiment-relevant words (Mudinas et al., 2012; Zou et al., 2018). For instance, this soft-attention mechanism tends to regard the name of a hotel "*Hilton*" with a good reputation in Clause3 as a positive word which could mislead the model into assigning a higher rating to aspect *room*. Therefore, a well-behaved approach should highlight sentiment-relevant words and discard noisy words for a specific aspect during model training.

In this paper, we propose a Hierarchical Reinforcement Learning (HRL) approach with a high-level policy and a low-level policy to address the above two challenges in DASC. First, a high-level policy is leveraged to select aspect-relevant clauses and discard noisy clauses during model training. Then, a low-level policy is leveraged to select sentiment-relevant words and discard noisy words inside the above selected clauses. Finally, a sentiment rating predictor is designed to provide reward signals to guide both clause and word selection. The empirical studies show that the proposed approach performs well by incorporating the clause selection and word selection strategies and significantly outperforms several state-of-the-art approaches including those with the soft-attention mechanism.

## 2 Hierarchical Reinforcement Learning

Figure 2 shows the overall framework of our Hierarchical Reinforcement Learning (HRL) approach which contains three components: a high-level policy for clause selection (Section 2.2); a low-level policy for word selection (Section 2.3); a sentiment rating predictor for providing reward signals to guide both the above clause and word selection (Section 2.4).

As a preprocessing, we adopt RST style discourse segmentation[1] (MANN, 1988) to segment all documents in corpus $\mathcal{C}$ into Elementary Discourse Units (EDUs), and consider thse EDUs as clauses by following Wang et al. (2018).

In summary, we formulate the task of DASC as a semi-Markov Decision process (Sutton et al., 1999b), i.e., hierarchical reinforcement learning with a high-level policy and a low-level policy. In particular, our HRL approach for DASC works as follows. Given a review document with a clause sequence and an aspect, the high-level policy decides whether a clause mentions this aspect. If yes, the high-level policy selects this clause and launches the low-level policy, which scans the words inside this selected clause one by one in order to select sentiment-relevant words. Otherwise, the high-level policy skips current clause and turns to the next clause until all clauses in the review document are scanned. During clause and word selection, a sentiment rating predictor is employed to provide reward signals to guide the above clause and word selection.

### 2.1 Clause Selection with High-level Policy

Assume that a review document $\mathcal{D}$ with a given aspect $x_{aspect}$ has been segmented into a clause sequence $\{u_1, ..., u_n\}$, high-level policy $\pi^h$ aims

[1]In preliminary experiments, we tried directly adopting sentence segmentation by following Li et al. (2018) rather than clause splitting and found that adopting clause splitting achieves better performance. Detailed comparison results are presented in ablation study of Section 3.2.

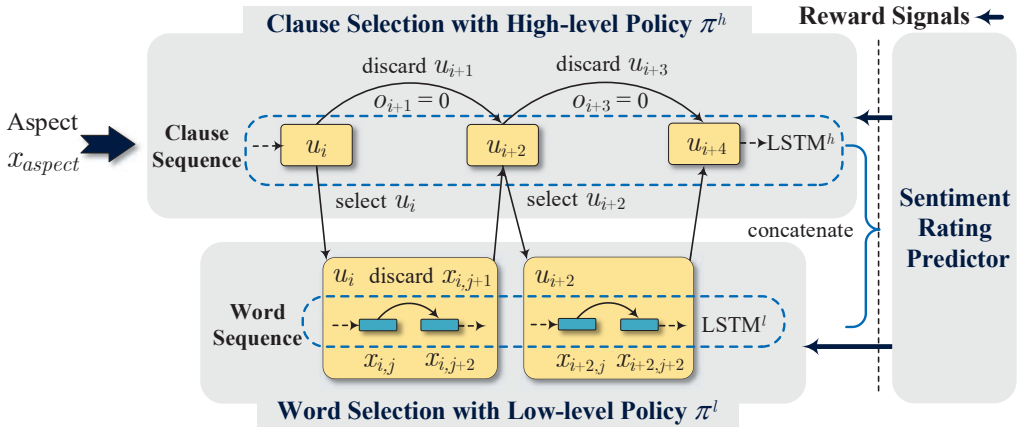


Figure 2: Overall architecture of our Hierarchical Reinforcement Learning (HRL) approach.

to select clause $u_i$ which truly mentions aspect $x_{aspect}$ and discard noisy ones. Here, clause $u_i$ consists of $k_i$ words $\{x_{i,1}, ..., x_{i,k_i}\}$. Once a clause is selected, it is passed to the low-level policy for further word selection.

During clause selection, we adopt a stochastic policy as high-level policy $\pi^h$, which can generate a conditional probability distribution $\pi^h(o|\cdot)$ over option (i.e., high-level action) sequence $o = [o_1, ..., o_n]$. Here, $o_i = 1$ indicates $u_i$ is selected; otherwise $o_i = 0$ indicates $u_i$ is discarded. More specifically, we adopt a LSTM model LSTM$^h$ to construct high-level policy $\pi^h$ for performing clause selection over the clause sequence. In LSTM$^h$, the hidden state $\hat{v}_i \in \mathbb{R}^d$ of clause $u_i$ and memory cell $c_i^h$ at $i$-th time-step are given by,

$$\hat{v}_i, c_i^h = \begin{cases} f(\hat{v}_{i-1}, c_{i-1}^h, v_i) & o_i = 1 \\ \hat{v}_{i-1}, c_{i-1}^h & o_i = 0 \end{cases} \quad (1)$$

where $v_i$ is the vector representation of clause $u_i$ and initialized by hidden state $\hat{w}_{i,k_i}$ of the last word $\hat{x}_{i,k_i}$ in clause $u_i$. Here, $\hat{w}_{i,k_i}$ is obtained from the pre-trained LSTM$^l$ (presented in line 3 of Algorithm 1); $f$ denotes all gate functions and update function of LSTM$^h$. Note that if $o_i = 0$, LSTM$^h$ will skip (i.e., discard) and not encode clause $u_i$, memory cell $c_i^h$ and hidden state $\hat{v}_i$ of current time-step $i$ are then directly copied from the previous time-step $i-1$.

In principle, the high-level policy $\pi^h$ uses a **Reward** to guide clause selection over the clause sequence. It samples an **Option** $o_i$ with the probability $\pi^h(o_i|s_i^h;\theta^h)$ at each **State** $s_i^h$. More concretely, the state, option and reward of $\pi^h$ are defined as follows.

• **State.** The state $s_i^h$ at $i$-th time-step should provide adequate information for deciding to select a clause or not for $x_{aspect}$. Thus, state $s_i^h \in \mathbb{R}^{4d}$ is composed of four parts, i.e., $\hat{v}_{i-1}$, $c_{i-1}^h$, $v_i$ and $v_a$, defined as $s_i^h = \hat{v}_{i-1} \oplus c_{i-1}^h \oplus v_i \oplus v_a$, where $v_a \in \mathbb{R}^d$ is aspect embedding[2] of $x_{aspect}$.

• **Option.** $\pi^h$ samples *option* $o_i \in \{0, 1\}$ by the conditional probability $\pi^h(o_i|s_i^h;\theta^h)$ which could be cast as a binary classification problem. Thus, we adopt logistic function to define $\pi^h(o_i|s_i^h;\theta^h)$.

$$\begin{aligned} o_i \sim \pi^h(o_i|s_i^h;\theta^h) &= o_i\sigma(W^h s_i^h + b^h) \\ &+(1-o_i)(1-\sigma(W^h s_i^h + b^h)) \end{aligned} \quad (2)$$

where $\theta^h = \{W^h \in \mathbb{R}^{1\times 4d}, b^h \in \mathbb{R}\}$ is the trainable parameter; $\sim$ denotes sampling operation; $\sigma$ denotes sigmod function.

• **Reward.** In order to select aspect-relevant clauses inside a clause sequence $\{u_1, ..., u_n\}$, given a sampled *option* trajectory $\tau^h = (s_1^h, o_1, r_1^h, ..., s_n^h, o_n, r_n^h) \sim \pi^h$, we compute the high-level cumulative reward $r_i^h$ at $i$-th time-step as follows:

$$\begin{aligned} r_i^h &= \lambda_1 \sum_{t=i}^{n} \gamma^{t-i} \log\cos(v_a, \hat{v}_t) \\ &+ \lambda_2 \sum_{t=i}^{n} \gamma^{t-i} r^l(u_t) + \lambda_3 \log p_\theta(y|\hat{v}_n) \end{aligned} \quad (3)$$

where $r_i^h$ consists of three different terms: **1)** The first term $\log\cos(v_a, \hat{v}_t)$ is a cosine intermediate reward computed by *cosine* similarity between aspect embedding $v_a \in \mathbb{R}^d$ and hidden state $\hat{v}_t \in \mathbb{R}^d$ of the $t$-th clause $u_t$. This reward provides aspect supervision signals to guide the policy to select aspect-relevant clauses. **2)** The second term

[2]Aspect embedding is initialized by an average of its keywords embeddings. Keywords for aspects (e.g., keywords for aspect *service* are service, smile etc.) are obtained from Yin et al. (2017).

$r^l(u_t) = \sum_{j=1}^{k_t} r^l_{t,j}$ is an intermediate reward from low-level policy after the word selection in the selected clause $u_t$ is finished. Note that if clause $u_t$ is discarded, $r^l(u_t) = 0$. This reward provides a feedback to indicate how well clause selection is. **3)** The third term $\log p_\theta(y|\hat{v}_n)$ is a delay reward from sentiment rating predictor. After LSTM$^h$ finishes all options, we feed the last hidden state $\hat{v}_n$ of LSTM$^h$ to the softmax decoder of sentiment rating predictor and then obtain a rating probability $p_\theta(y|\hat{v}_n)$ for ground-truth rating label $y$ to compute this delay reward. This reward provides additional reward signals to guide policy to select discriminative clauses. Besides, $\gamma$ is the discount factor; $\lambda_1$, $\lambda_2$ and $\lambda_3$ are weight parameters.

### 2.2 Word Selection with Low-level Policy

Given a word sequence $\{x_{i,1}, ..., x_{i,k_i}\}$ of clause $u_i$ selected by the high-level policy, low-level policy $\pi^l$ aims to select the sentiment-relevant word $x_{i,j}$ and discard noisy ones.

During word selection, we still adopt a stochastic policy as low-level policy $\pi^l$, which can generate a conditional probability distribution $\pi^l(a|\cdot)$ over action sequence $a = [a_{i,1}, ..., a_{i,k_i}]$, where $a_{i,j} = 1$ indicates $j$-th word $x_{ij}$ in $i$-th clause is selected; $a_{i,j} = 0$ indicates $x_{i,j}$ is discarded. Similar to clause selection, we adopt another LSTM model LSTM$^l$ to construct low-level policy $\pi^l$ for performing word selection over word sequence $\{x_{i,1}, ..., x_{i,k_i}\}$ of each clause (Note that, as shown in Figure 2, LSTM$^l$ is shared by all selected clauses from high-level policy). In LSTM$^l$, the hidden state $\hat{w}_{i,j} \in \mathbb{R}^d$ of word $x_{i,j}$ and memory cell $c^l_j$ at $j$-th time-step (Here, we omit the clause index and only use $j$ to denote the $j$-th time-step in $i$-th clause $u_i$) are given by,

$$\hat{w}_{i,j}, c^l_{i,j} = \begin{cases} f(\hat{w}_{i,j-1}, c^l_{i,j-1}, w_{i,j}) & a_{i,j} = 1 \\ \hat{w}_{i,j-1}, c^l_{i,j-1} & a_{i,j} = 0 \end{cases} \tag{4}$$

where $w_{i,j}$ is word embedding of $x_{i,j}$. Similar to LSTM$^h$, in LSTM$^l$, if $a_{i,j} = 0$, i.e., word $x_{i,j}$ is discarded, memory cell and hidden state of current time-step are directly copied from the previous time-step. Then, we illustrate state, action and reward of low-level policy as follows.

• **State.** The state $s^l_{i,j}$ at $j$-th time-step should provide adequate information for deciding to select a word or not. Thus, the state $s^l_{i,j} \in \mathbb{R}^{3d}$ is composed of three parts, i.e., $\hat{w}_{i,i-1}$, $c^l_{i,j-1}$, $w_{i,j}$, defined as $s^l_{i,j} = \hat{w}_{i,j-1} \oplus c^l_{i,j-1} \oplus w_{i,j}$.

• **Action.** $\pi^l$ samples *action* $a_{i,j} \in \{0, 1\}$ by the conditional probability $\pi^l(a_{i,j}|s^l_{i,j}; \theta^l)$. Thus, similar to high-level policy, we adopt logistic function to define $\pi^l(a_{i,j}|s^l_{i,j}; \theta^l)$.

$$\begin{aligned} a_{i,j} \sim \pi^l(a_{i,j}|s^l_{i,j}; \theta^l) = a_{i,j}\sigma(W^l s^l_{i,j} + b^l) \\ +(1 - a_{i,j})(1 - \sigma(W^l s^l_{i,j} + b^l)) \end{aligned} \tag{5}$$

where $\theta^l = \{W^l \in \mathbb{R}^{1\times 3d}, b^l \in \mathbb{R}\}$ is the trainable parameter.

• **Reward.** Similarly, in order to select sentiment-relevant words inside a word sequence $\{x_{i,1}, ..., x_{i,k_i}\}$, given a sampled *action* trajectory $\tau^l = (s^l_{i,1}, a_{i,1}, r^l_{i,1}, ..., s^l_{i,k_i}, a_{i,k_i}, r^l_{i,k_i}) \sim \pi^l$, we compute the low-level cumulative reward $r^l_{i,j}$ at $j$-th time-step as:

$$r^l_{i,j} = \lambda'_1 \log p_\theta(y|\hat{w}_{i,k_i}) + \lambda'_2(-N')/N \tag{6}$$

where $r^l_{i,j}$ consists of two terms: **1)** Similar to high-level policy, the first term $\log p_\theta(y|\hat{w}_{i,k_i})$ is a delay reward provided by sentiment rating predictor. After LSTM$^l$ finishes all actions, we feed last hidden state $\hat{w}_{i,k_i}$ in $i$-th clause to softmax decoder of sentiment rating predictor and then we can obtain this delay reward. This reward provides rating supervision information to guide policy to select discriminative words, i.e., sentiment-relevant words. **2)** The second term $\gamma(-N')/N$ is a penalty delay reward. $N' = \sum_{j=1}^{k_i} a_{i,j}$ denotes the number of selected words. The basic idea of using this penalty reward is to select words as small as possible because sentiment-relevant words is usually a small subset of all words inside the clause. Note that, we could also adopt external sentiment lexicons to achieve this goal, but sentiment lexicons are difficult to obtain in many real-world applications. Besides, $\lambda'_1$, $\lambda'_2$ are weight parameters.

### 2.3 Sentiment Rating Predictor

The goal of sentiment rating predictor lies in two-fold. On one hand, during model training, the goal of sentiment rating predictor is to use a softmax decoder to provide rating probabilities as the reward signals (see Eq.(3) and Eq.(6)) to guide both clause and word selection.

On the other hand, when model training is finished, i.e., both high-level and low-level policy finish all their selections, the goal of sentiment rating predictor is to perform DASC. Specifically, we first regard last state $\hat{v}_n$ of LSTM$^h$ as the representation of all selected clauses while last state

$\hat{w}_{n,k_n}$ of LSTM$^l$ as the representation of all selected words. Then, we concatenate $\hat{v}_n$ and $\hat{w}_{n,k_n}$ to compute the final representation $z$ of the review document $\mathcal{D}$ as: $z = \hat{v}_n \oplus \hat{w}_{n,k_n}$, where $\oplus$ denotes concatenation operation. Finally, to perform DASC, we feed $z$ to a softmax decoder as follows:

**Softmax Decoder.** We first feed $z$ to a softmax classifier $m = Wz + b$, where $m \in \mathbb{R}^C$ is output vector; $\theta = \{W, b\}$ is trainable parameter. Then, the probability of labeling sentence with sentiment rating $\hat{y} \in [1, C]$ is computed by $p_\theta(\hat{y}|z) = \frac{\exp(m_{\hat{y}})}{\sum_{\ell=1}^{C} \exp(m_\ell)}$. Finally, the label with the highest probability stands for the predicted sentiment rating for $x_{aspect}$.

Note that, for an aspect, if no clauses and words are finally selected from a review document, the model will assign a random rating for this aspect.

### 2.4 Model Training via Policy Gradient and Back-Propagation

The parameters in HRL are learned according to Algorithm 1. Specifically, these parameters can be divided into two groups: **1)** $\theta^h$ and $\theta^l$ of high-level policy $\pi^h$ and low-level policy $\pi^l$ respectively. **2)** $\theta$ of LSTM$^h$, LSTM$^l$ and softmax decoder.

For $\theta^h$ of high-level policy, we optimize it with policy gradient (REINFORCE) (Williams, 1992; Sutton et al., 1999a). The policy gradient w.r.t. $\theta^h$ is computed by differentiating the maximized expected reward $J(\theta^h)$ as follows:

$$\nabla_{\theta^h} J(\theta^h) = \mathbb{E}_{\tau^h \sim \pi^h}[\sum_{i=1}^{n} \mathcal{R}^h \nabla_{\theta^h} \log \pi^h(o_i|s_i^h; \theta^h)] \tag{7}$$

where $\mathcal{R}^h = r_i^h - b(\tau^h)$ is the advantage estimate of the high-level reward. Here, $b(\tau^h)$ is the *baseline* (Williams, 1992) which is used to reduce the variance of the high-level reward without altering its expectation theoretically. In practical use, we sample some trajectories $\tau_1^h, \tau_2^h, ..., \tau_m^h$ over the clause sequence with the current high-level policy. The model will assign a reward score to each sequence according to the designed scores function, and then estimates $b(\tau^h)$ as the average of those rewards. Similarly, the policy gradient w.r.t. $\theta^l$ of low-level policy is given by,

$$\nabla_{\theta^l} J(\theta^l) = \mathbb{E}_{\tau^l \sim \pi^l}[\sum_{j=1}^{k_i} \mathcal{R}^l \nabla_{\theta^l} \log \pi^l(a_{i,j}|s_{i,j}^l; \theta^l)] \tag{8}$$

**Algorithm 1** Hierarchical reinforcement learning

**Input:** Corpus $\mathcal{C}$; a review document $\mathcal{D}$ with a clause sequence $\{u_1, ..., u_n\}$; a clause $u_i$ with a word sequence $\{x_{i,1}, ..., x_{i,k_i}\}$.
Initialize parameters $\theta$, $\theta^h$ and $\theta^l$ randomly;
Pre-train LSTM$^l$ by forcing $\pi^l$ to select all words for classification, and update $\theta$ of LSTM$^l$ by Eq.(9);
Pre-train LSTM$^h$ by forcing $\pi^h$ to select all clauses for classification, and update $\theta$ of LSTM$^h$ by Eq.(9);
Fix all parameters $\theta$ and update $\theta^h$, $\theta^l$ as follows:
**for** review document $\mathcal{D} \in \mathcal{C}$ **do**
  **for** clause $u_i \in \{u_1, ..., u_n\}$ **do**
    Sample *option* $o_i \sim \pi^h(o_i|s_i^h; \theta^h)$;
    **if** option $o_i = 1$ **then**
      **for** word $x_{i,j} \in \{x_{i,1}, ..., x_{i,k_i}\}$ **do**
        Sample *action* $a_{i,j} \sim \pi^h(a_{i,j}|s_{i,j}^l; \theta^h)$;
      **end for**
      Compute $r_{i,j}^l$ by Eq.(6);
      Update $\theta^l$ by Eq.(8);
    **end if**
  **end for**
  Compute $r_i^h$ by Eq.(3);
  Update $\theta^h$ by Eq.(7);
**end for**

| **Datasets** | *#documents* | *#words/doc* | *#Aspect* |
|---|---|---|---|
| TripUser | 58632 | 181.03 | 7 |
| TripAdvisor | 29391 | 251.7 | 7 |
| BeerAdvocate | 51020 | 144.5 | 4 |

Table 1: Statistics of three datasets. *#words/doc* denotes the number of words (average per document). *#Aspect* denotes the number of aspects in each dataset. The rating scales of TripAdvisor and BeerAdvocate are 1-5 and 1-10 respectively.

where $\mathcal{R}^l = r_{i,j}^l - b(\tau^l)$ is the advantage estimate of the low-level reward. Similarly, $b(\tau^l)$ is used to reduce the variance of the low-level reward.

For $\theta$, we optimize it with back-propagation. The objective of learning $\theta$ is to minimize the cross-entropy loss in the classification phase, i.e.,

$$J(\theta) = \mathbb{E}_{(\mathcal{D}, x_{aspect}, y) \sim \mathcal{C}}[-\log p_\theta(y|z)] + \frac{\delta}{2}||\theta||_2^2 \tag{9}$$

where $(\mathcal{D}, x_{aspect}, y)$ denotes a review document $\mathcal{D}$ with a given aspect $x_{aspect}$ from corpus $\mathcal{C}$; $y$ is ground-truth sentiment rating for aspect $x_{aspect}$. $\delta$ is a $L_2$ regularization.

## 3 Experimentation

### 3.1 Experimental Settings

**Data.** We conduct our experiments on three public datasets on DASC, i.e., TripUser (Li et al., 2018), TripAdvisor (Wang et al., 2010) and BeerAdvocate (McAuley et al., 2012; Lei et al., 2016). In the experiment, we adopt Discourse Segmentation

| Approaches | TripUser | | | | TripAdvisor | | | | BeerAdvocate | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | *Development* | | *Test* | | *Development* | | *Test* | | *Development* | | *Test* | |
| | Acc.↑ | MSE↓ | Acc.↑ | MSE↓ | Acc.↑ | MSE↓ | Acc.↑ | MSE↓ | Acc.↑ | MSE↓ | Acc.↑ | MSE↓ |
| SVM | - | - | 46.35$^{\dagger}$ | 1.025$^{\dagger}$ | 34.30$^{\ddagger}$ | 1.982$^{\ddagger}$ | 35.26$^{\ddagger}$ | 1.963$^{\ddagger}$ | 25.70$^{\ddagger}$ | 3.286$^{\ddagger}$ | 25.79$^{\ddagger}$ | 3.270$^{\ddagger}$ |
| LSTM | 53.23 | 0.787 | 52.74 | 0.794 | 43.85$^{\ddagger}$ | 1.525$^{\ddagger}$ | 44.02$^{\ddagger}$ | 1.470$^{\ddagger}$ | 35.23$^{\ddagger}$ | 2.112$^{\ddagger}$ | 34.78$^{\ddagger}$ | 2.097$^{\ddagger}$ |
| MAMC | - | - | 55.49$^{\dagger}$ | 0.583$^{\dagger}$ | 46.21$^{\ddagger}$ | 1.091$^{\ddagger}$ | 46.56$^{\ddagger}$ | 1.083$^{\ddagger}$ | 39.43$^{\ddagger}$ | 1.696$^{\ddagger}$ | 38.06$^{\ddagger}$ | 1.755$^{\ddagger}$ |
| HARN | - | - | 58.15$^{\dagger}$ | 0.528$^{\dagger}$ | - | - | 48.21$^{\ddagger}$ | 0.923$^{\ddagger}$ | 39.81 | 1.672 | 38.19 | 1.751 |
| HUARN | - | - | 60.70$^{\dagger}$ | 0.514$^{\dagger}$ | - | - | - | - | - | - | - | - |
| C-HAN | 58.49 | 0.602 | 57.38 | 0.543 | 47.61 | 0.914 | 47.08 | 0.955 | 38.67 | 1.703 | 37.95 | 1.801 |
| HS-LSTM | 59.75 | 0.566 | 59.01 | 0.524 | 48.45 | 0.947 | 46.84 | 1.013 | 37.43 | 1.870 | 36.83 | 1.912 |
| RL-Word-Selection | 60.15 | 0.475 | 59.55 | 0.519 | 48.55 | 0.913 | 48.51 | 0.917 | 39.92 | 1.648 | 38.45 | 1.697 |
| RL-Clause-Selection | 61.32 | 0.433 | 60.54 | 0.461 | 51.05 | 0.762 | 50.02 | 0.781 | 41.39 | 1.505 | 39.76 | 1.622 |
| **HRL** | **62.97** | **0.336** | **62.84** | **0.351** | **52.71** | **0.652** | **52.27** | **0.662** | **43.41** | **1.416** | **41.39** | **1.503** |

Table 2: Comparison of our approaches and other baseline approaches to DASC. The results with symbol † are retrieved from Li et al. (2018) and those with ‡ are from Yin et al. (2017)

Tool[3] to segment all reviews in the three datasets into EDUs (i.e., clauses). Moreover, we adopt training/development/testing settings (8:1:1) by following Yin et al. (2017); Li et al. (2018). Table 1 shows the statistics of the three datasets.

**Implementation Details.** We adopt the pre-trained 200-dimension word embeddings provided by Yin et al. (2017). The dimension of LSTM hidden states is set to be 200. The other hyper-parameters are tuned according to the performance in the development set. Specifically, we adopt Adam optimizer (Kingma and Ba, 2014) with an initial learning rate of 0.012 for cross-entropy training and adopt SGD optimizer with a learning rate of 0.008 for all policy gradients training. For rewards of high-level and low-level policies, $\gamma$ is 0.8; $\lambda_1$, $\lambda_2$ and $\lambda_3$ are 0.25, 0.25 and 0.5 respectively. $\lambda_1'$, $\lambda_2'$ are 0.6 and 0.4. Additionally, the batch size is set to be 64, regularization weight is set to be $10^{-5}$ and the dropout rate is 0.2.

**Evaluation Metrics.** The performance is evaluated using Accuracy (Acc.) and MSE as Yin et al. (2017). Moreover, $t$-test is used to evaluate the significance of the performance difference between two approaches (Yang and Liu, 1999).

**Baselines.** We compare HRL with the following baselines: **1) SVM** (Yin et al., 2017). This approach only adopts unigram, bigram as features to train an SVM classifier. **2) LSTM** (Tang et al., 2015). This is a neural network approach to document-level sentiment classification which employs gated LSTM to learn text representation. **3) MAMC** (Yin et al., 2017). This approach employs hierarchical iterative attention to learn aspect-specific representation. This is a state-of-the-art approach to DASC. **4) HARN** (Li et al., 2018). This approach adopts hierarchical attention to incorporate overall rating and aspect information so as to learn aspect-specific representation. This is another state-of-the-art approach to DASC. **5) HUARN** (Li et al., 2018). This approach extends HARN by integrating additional user information. This is another state-of-the-art approach to DASC. **6) C-HAN** (Wang et al., 2018). This approach adopts hierarchical attention to incorporate clause and aspect information so as to learn text representation. Although this is a state-of-the-art approach to sentence-level ASC, it could also be directly applied in DASC. **7) HS-LSTM** (Zhang et al., 2018). This is a reinforcement learning approach to text classification, which employs a hierarchically LSTM to learn text representation. **8) RL-Word-Selection.** Our approach which leverages only the word selection strategy by using the low-level policy. **9) RL-Clause-Selection.** Our approach which leverages only the clause selection strategy by using the high-level policy.

### 3.2 Experimental Results

Table 2 shows the performance comparison of different approaches. From this table, we can see that, all LSTM-based approaches outperform **SVM**, showing that LSTM has potentials in automatically learning text representations and can bring performance improvement for DASC.

Four state-of-the-art ASC approaches including **MAMC**, **HARN**, **HUARN** and **C-HAN** all perform better than **LSTM**. These results confirm the helpfulness of considering aspect information in DASC. Besides, we find the reinforcement learning based approach **HS-LSTM** without consid-

[3]http://alt.qcri.org/tools/discourse-parser/

| Approaches | TripUser | | TripAdvisor | | BeerAdvocate | |
|---|---|---|---|---|---|---|
| | Acc.↑ | MSE↓ | Acc.↑ | MSE↓ | Acc.↑ | MSE↓ |
| **HRL** | **62.84** | **0.351** | **52.27** | **0.662** | **41.39** | **1.503** |
| w/o *cosine* intermediate reward in Eq.(3) | 60.15 | 0.487 | 49.02 | 0.811 | 38.72 | 1.673 |
| w/o penalty delay reward in Eq.(6) | 61.52 | 0.425 | 51.15 | 0.711 | 40.13 | 1.577 |
| w/o the representation of selected clauses | 58.40 | 0.622 | 46.33 | 1.207 | 38.61 | 1.684 |
| w/o the representation of selected words | 60.08 | 0.497 | 48.94 | 0.886 | 39.92 | 1.626 |
| using sentence splitting instead of clause | 59.74 | 0.504 | 50.11 | 0.842 | 39.33 | 1.651 |

Table 3: Ablation study of HRL on three different datasets.

ering aspect information can achieve comparable performance with **MAMC**, **HARN**, **C-HAN**, and even beat **MAMC** on two datasets **TripUser** and **TripAdvisor**, which demonstrates that using reinforcement learning is a good choice to learn text representation for DASC.

Our approach **RL-Word-Selection** and **RL-Clause-Selection** outperform most above approaches and they only perform slightly worse than **HUARN**. This result encourages to perform clause or word selection in DASC. Among all these approaches, our approach **HRL** performs best and it significantly outperforms ($p$-value $<$ 0.01) strong baseline **HUARN** which actually considers some other kinds of external information, such as the overall rating and the user information. These results encourage to perform both clause and word selection in DASC.

**Ablation Study.** Further, we conduct the ablation study of HRL to evaluate the contribution of each component. The results are shown in Table 3. From this table, we can see that **1)** Using *cosine* intermediate reward in Eq.(3) can averagely improve Acc. by 2.87% on three different datasets. **2)** Using penalty delay reward in Eq.(6) can averagely improve Acc. by 1.23%. **3)** To concatenate additional representation of the selected clauses in rating predictor can improve Acc. by 4.38%. **4)** To concatenate additional representation of the selected words in rating predictor can improve Acc. by 2.72%. **5)** Using the clause splitting instead of sentence splitting could improve Acc. by 2.44%. This confirms that it is more appropriate to consider clauses as the segmentation units than sentences. This is because that 90% of clauses contain only one opinion expression as proposed in Bayoudhi et al. (2015). For instance, as shown in Figure 1, if we use the sentences as the segmentation units, Clause1-Clause3 will be assigned into one unit while they talk about two aspects, i.e., *location* and *room*. In this scenario, sentence selection will not be able to discard noisy parts inside the sentence for the aspect *location* or *room*.

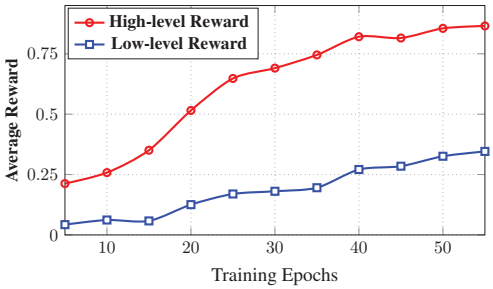


Figure 3: Training reward with different epochs

## 4 Analysis and Discussion

**Analysis of HRL Training.** Figure 3 shows two average rewards (each epoch) of high-level and low-level policy on BeerAdvocate respectively. To clearly observe the change of the reward, following Lillicrap et al. (2016), all rewards are normalized to $(0, 1)$. From this figure, we can see that, both the high-level and low-level reward increase as the training algorithm iterates. This result indicates that our HRL approach is capable of stably revising its policies to obtain more discriminative clauses and words for better performing.

**Analysis of Clause and Word Selection.** Figure 4 shows visualizations of our HRL approach which performs the clause selection and word selection on a review document. From this figure, we can see that HRL is able to precisely select aspect-relevant clauses, i.e., Clause1 and Clause2, for aspect *location* while select Clause3 and Clause4 for *room*. Further, HRL is able to select all sentiment-relevant words, such as "*close*" and "*very convenient*" for aspect *location*, while "*a little uncomfortable*" and "*nitpicking*"for *room*.

**Error Type Breakdown.** We analyze error cases in the experiments and broadly categorize them into three types: **(1)** The first type of errors are due to negation words. For instance, for the review "*The taste of this beer is not good, don't buy it*", HRL could precisely select the sentiment




(a) For aspect ***location*** (b) For aspect ***room***

Figure 4: Visualizations of clause selection (along with the row ↓) and word selection (along with the column →) for aspect (a) ***location*** and (b) ***room***. Red denotes the clause has been selected, blue denotes the word has been selected and other colors denote the token has been discarded.

"*good*", but fail to select the negation word "*not*". This inspires us to work on optimizing our approach in order to capture negation scope better in our future work. **(2)** The second type of errors are due to comparative opinions. For instance, for the review "*The room of Sheraton is much better than this one.*", HRL incorrectly predicts high rating (5 stars) to aspect *room*. It would be interesting to see if incorporating syntactic information can solve this problem and bring performance improvement. **(3)** Finally, some errors are due to mistakes during clause splitting (i.e., EDU splitting). For instance, for the review "[*This hotel having good location*] [*often needs lots of time to check in.*]", it is assigned into one clause while it talks two aspects, i.e., *location* and *check in/front desk*. This encourages to improve the performance of clause splitting for informal review texts.

## 5 Related Work

**Aspect Sentiment Classification.** Traditional studies for DASC mainly focus on feature engineering to explore efficient features for DASC (Titov and McDonald, 2008; Lu et al., 2011; McAuley et al., 2012). Recently, neural networks with the characteristic of automatically mining features have shown promising results on DASC. Lei et al. (2016) focused on extracting rationales for aspects and build a neural text regressor to predict aspect rating; Yin et al. (2017) focused on using hierarchical iterative attention to learn aspect-specific text representation for DASC; Li et al. (2018) employed a hierarchical attention approach to DASC which incorporates both the external user and overall rating information. Besides, neural networks have been widely adopted for performing a closely related task, i.e., Sentence-level Aspect Sentiment Classification (Wang et al., 2016; Tang et al., 2016; Wang and Lu, 2018).

**Reinforcement Learning.** In recent years, reinforcement learning has been applied successfully to some NLP tasks. Guo (2015) employed deep Q-learning to improve the seq2seq model for the text generation task; Li et al. (2016) showed how to apply deep reinforcement learning to model future reward in the chatbot dialogue task; Takanobu et al. (2018) employed hierarchical reinforcement learning to model the relation extraction task; Zhang et al. (2018) combined LSTM with reinforcement learning to learn structured representations for the text classification task, which is inspirational to our approach.

Unlike all above studies, inspired by the cognitive process of human beings, this paper proposes a new HRL approach to DASC task. To the best of our knowledge, this is the first attempt to address DASC with HRL.

## 6 Conclusion

In this paper, we propose a hierarchical reinforcement learning approach to DASC. The main idea of the proposed approach is to perform sentiment classification like human beings. Specifically, our approach employs a high-level policy and a low-level policy to perform clause selection and word selection in DASC respectively. Experimentation shows that both the clause and word selection are effective for DASC and the proposed approach significantly outperforms several state-of-the-art baselines for DASC.

In our future work, we would like to solve other challenges in DASC, e.g., negation detection problem, to further improve the performance. Furthermore, we would like to apply our HRL approach to other sentiment analysis tasks, such as aspect and opinion co-extraction, and dialog-level sentiment analysis.

## Acknowledgments

We thank our anonymous reviewers for their helpful comments. This work was supported by three NSFC grants, i.e., No.61672366, No.61702149 and No.61525205. This work was also supported by the joint research project of Alibaba Group and Soochow University, and a Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions (PAPD).

## References

A. Bayoudhi, H. Ghorbel, H. Koubaa, and L. Belguith. 2015. Sentiment classification at discourse segment level: Experiments on multi-domain arabic corpus. *JLCL*, 30(1).

Hongyu Guo. 2015. Generating text with deep reinforcement learning. *CoRR*, abs/1510.09202.

D. Kingma and J. Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

T. Lei, R. Barzilay, and T. Jaakkola. 2016. Rationalizing neural predictions. In *Proceedings of EMNLP*.

J. Li, H. Yang, and C. Zong. 2018. Document-level multi-aspect sentiment classification by jointly modeling users, aspects, and overall ratings. In *Proceedings of COLING*.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016. Deep reinforcement learning for dialogue generation. *CoRR*, abs/1606.01541.

S. Li, C. Huang, G. Zhou, and Sophia Yat Mei Lee. 2010. Employing personal/impersonal views in supervised and semi-supervised sentiment classification. In *Proceedings of ACL*.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2016. Continuous control with deep reinforcement learning. In *Proceedings of ICLR*.

X. Liu, L. Mou, H. Cui, Z. Lu, and S. Song. 2018. JUMPER: learning when to make classification decisions in reading. *CoRR*, abs/1807.02314.

B. Lu, M. Ott, C. Cardie, and B. K. Tsou. 2011. Multi-aspect sentiment analysis with topic models. In *Proceedings of ICDM*.

W. MANN. 1988. Rhetorical structure theory : Toward a functional theory of text organization. *Text*, 8(3):243–281.

G. Marcus. 2018. Deep learning: A critical appraisal. *CoRR*, abs/1801.00631.

J. McAuley, J. Leskovec, and D. Jurafsky. 2012. Learning attitudes and attributes from multi-aspect reviews. In *Proceedings of ICDM*.

Andrius Mudinas, Dell Zhang, and Mark Levene. 2012. Combining lexicon and learning based approaches for concept-level sentiment analysis. In *Proceedings of the first international workshop on issues of sentiment discovery and opinion mining*, page 5. ACM.

B. Pang and L. Lee. 2007. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2).

R. Sutton, D. McAllester, and etc. 1999a. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of NIPS*.

R. Sutton, D. Precup, and S. Singh. 1999b. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artif. Intell.*

Ryuichi Takanobu, Tianyang Zhang, Jiexi Liu, and Minlie Huang. 2018. A hierarchical framework for relation extraction with reinforcement learning. *CoRR*, abs/1811.03925.

D. Tang, B. Qin, and T. Li. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of EMNLP*.

D. Tang, B. Qin, and T. Liu. 2016. Aspect level sentiment classification with deep memory network. In *Proceedings of EMNLP*.

I. Titov and R. McDonald. 2008. A joint model of text and aspect ratings for sentiment summarization. In *Proceedings of ACL*.

B. Wang and W. Lu. 2018. Learning latent opinions for aspect-level sentiment classification. In *Proceedings of AAAI*.

H. Wang, Y. Lu, and C. Zhai. 2010. Latent aspect rating analysis on review text data: a rating regression approach. In *Proceedings of KDD*.

J. Wang, J. Li, S. Li, Y. Kang, M. Zhang, L. Si, and G. Zhou. 2018. Aspect sentiment classification with both word-level and clause-level attention networks. In *Proceedings of IJCAI*.

Y. Wang, M. Huang, X. Zhu, and L. Zhao. 2016. Attention-based LSTM for aspect-level sentiment classification. In *Proceedings of EMNLP*.

R. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8.

Y. Yang and X. Liu. 1999. A re-examination of text categorization methods. In *Proceedings of SIGIR*.

Y. Yin, Y. Song, and M. Zhang. 2017. Document-level multi-aspect sentiment classification as machine comprehension. In *Proceedings of EMNLP*.

T. Zhang, M. Huang, and L. Zhao. 2018. Learning structured representation for text classification via reinforcement learning. In *Proceedings of AAAI*.

Y. Zou, T. Gui, Q. Zhang, and X. Huang. 2018. A lexicon-based supervised attention model for neural sentiment analysis. In *Proceedings of COLING*.

# A Dataset of General-Purpose Rebuttal

**Matan Orbach**[1], **Yonatan Bilu**[1], **Ariel Gera**[1], **Yoav Kantor**[1],
**Lena Dankin**[1], **Tamar Lavee**[1], **Lili Kotlerman**[*2], **Shachar Mirkin**[*3],
**Michal Jacovi**[1], **Ranit Aharonov**[1] **and Noam Slonim**[1]
[1]IBM Research, [2]Intuition Robotics, [3]Digimind
{matano,yonatanb,arielge,yoavka,lenad,tamarl, michal.jacovi, ranita,noams}@il.ibm.com,
lili@intuitionrobotics.com, shachar.mirkin@digimind.com

## Abstract

In Natural Language Understanding, the task of response generation is usually focused on responses to short texts, such as tweets or a turn in a dialog. Here we present a novel task of producing a critical response to a long argumentative text, and suggest a method based on general rebuttal arguments to address it. We do this in the context of the recently-suggested task of listening comprehension over argumentative content: given a speech on some specified topic, and a list of relevant arguments, the goal is to determine which of the arguments appear in the speech. The general rebuttals we describe here (written in English) overcome the need for topic-specific arguments to be provided, by proving to be applicable for a large set of topics. This allows creating responses beyond the scope of topics for which specific arguments are available. All data collected during this work is freely available for research[1].

## 1 Introduction

A key element in argumentation is rebuttal: the ability to contest an argument by presenting a counter-argument. It is an important skill, not easily learned, and valued in many fields such as politics and science. It is useful for advancing your own views and beliefs over opposing ones, but perhaps more importantly, it facilitates a critical examination of the views and beliefs that you hold. An automatic rebuttal system could therefore be useful whenever critical analysis of written or spoken content is required – be it an elementary school student writing an essay or a seasoned journalist composing an op-ed.

In the context of Natural Language Understanding, the study of rebuttal and counter-arguments has focused on elucidating such relations between given arguments. Indeed, such "attack" relations are the foundation of *Argumentation Frameworks* (Dung, 1995); such frameworks have been one of the main objects of study in computational argumentation.

A related task, that of generating a response which need not be a rebuttal or even argumentative, has been the subject of much research, especially in the context of dialog systems, chat bots, and question answering. In this line of work the response typically follows a short input text, often only a sentence or two.

Here we suggest the task of producing a rebuttal in response to a long argumentative text. Specifically, we consider spoken speeches around four minutes long. In addition to being longer, and perhaps because they are so, these kinds of texts tend to include very general claims, often implicit in the text. As such, these claims may appear in varied contexts, and it may be feasible to compile a list of such claims independently of the speeches' topics.

For example, a concern that often comes up in debates about policy is that implementing the policy (or failing to do so) disproportionately harms minorities. This claim can be made to oppose school vouchers, to oppose voter registration laws or to support criminal justice reform. Moreover, for some such general claims, it is feasible to phrase a single rebuttal response which can fit many of the contexts in which the claim might be made. In the above example, such a response may talk about separating between the policy at hand, which should be adopted based on its merits, and the need to right historical wrongs, which should be pursued independently.

We envision an automatic rebuttal system based on this observation, which includes a manually curated General-Purpose Rebuttal Knowledge Base (GPR-KB) comprised of claims and matching re-

[*]This work was done at IBM Research.

[1]https://www.research.ibm.com/haifa/dept/vst/debating_data.shtml



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5591–5601,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

buttal responses. Given an argumentative text, the system would identify which claims from the GPR-KB are made in the text (explicitly or implicitly), and produce a rebuttal using the available counter-arguments. Clearly, many of the claims made in the text would not appear in the GPR-KB. The objective is therefore not to identify and rebut all arguments, but rather to identify and rebut *some* arguments, and construct a GPR-KB that facilitates that.

Such a system (based on the more elaborate CoPA modeling of Bilu et al., 2019) was indeed implemented as a key element in IBM's Project Debater rebuttal mechanism, and demonstrated during the live debate held between it and debating champion Harish Natarajan[2]. However, a rebuttal system of this nature may be of interest beyond the realm of debating technologies. For example, such a system may be instrumental in making media consumption a more critical process, by automatically challenging the consumer with counter-arguments. Similarly, it can be applicable in the education domain, stimulating critical thinking by prompting students with counter-arguments in response to (or during) essay-composition tasks.

The formation of a GPR-KB that is applicable to the real world poses several challenges. First, phrased claims must be both relevant to a variety of topics, and commonly used. Second, pre-written rebuttals should be effective and persuasive, even though they are created without prior knowledge of the context in which they are to be of use. We address these issues by turning to a domain in which a similar problem is solved by humans: the world of competitive debates. In these contests, successful debaters need to combine specific knowledge about the topic at hand, with general arguments that arise from the underlying principles of the debate. Their ability to use such general arguments for different topics lays the basis for using a GPR-KB as the one described above.

Accordingly, we asked an expert debater to create the initial GPR-KB by suggesting common claims and preparing matching rebuttals. The full process is detailed in §3.2.

To assess the usefulness of the suggested claims and rebuttals in the real world, we performed several steps of labeling on the dataset we constructed in Mirkin et al. (2018), containing spoken argumentative content discussing controversial topics. Details of this process, along with an analysis showing the high coverage obtained by our knowledge base, are described in §4.

Another major challenge is the development of automatic methods for identifying whether knowledge-base claims are mentioned by speakers. We break this problem into a three-stage funnel – identifying whether: (i) a claim is relevant to the topic; (ii) the claim's stance aligns with that of the speaker; (iii) the claim was made by the speaker. We provide simple baseline results for this third step (§5). Interestingly, we observe that simply selecting the claim with the highest acceptance rate in the training data (without looking at the text) provides a challenging baseline.

The main contributions of this work are (i) the introduction of a novel task in NLU: producing rebuttal in response to a long argumentative text (ii) a manually constructed GPR-KB shared across multiple topics (iii) an additional layer of labeling to our dataset from Mirkin et al. (2018) for such claims (iv) a baseline for detecting whether such a claim was mentioned in a speech.

## 2 Related Work

In Mirkin et al. (2018) we introduced the task of Listening Comprehension over argumentative content. That work analyzes recorded speeches, and tries to identify whether arguments from iDebate[3] are mentioned in the speech. Similarly, in Lavee et al. (2019) we addressed this task by first mining arguments from a large news corpus, and then identifying the arguments which are mentioned in speeches.

This work complements our previous works in two ways. First, the GPR-KB constructed here is of general claims, with wide cross-topic relevance. It facilitates Listening Comprehension for topics not mentioned in iDebate, or topics for which automatic argument mining does not yield satisfactory results. Second, while Mirkin et al. (2018) mention that the iDebate counter points can in principle be used for rebuttal, and Lavee et al. (2019) suggest mining opposing arguments from their corpus to counter arguments mentioned in speeches, pursuing both ideas is left for future work. We pick up the baton (in the context of the GPR-KB suggested here), and annotate the validity of the counter arguments as rebuttal to the ideas

[2]Video of the debate at https://www.youtube.com/watch?v=m3u-1yttrVw

[3]https://idebate.org/debatabase

expressed in a matching speech.

Response generation has been the subject of much research, using a wide variety of methods (e.g. Ritter et al., 2011; Sugiyama et al., 2013; Shin et al., 2015; Yan et al., 2016; Xing et al., 2017). In the context of dialog systems (see recent survey in Chen et al., 2017), there is usually a distinction between task-oriented systems (Wen et al., 2016) and open-domain ones (Mazaré et al., 2018; Weizenbaum, 1966). The task here can be seen as lying in between the two: on the one hand it allows for a response to speeches on a variety of topics; on the other, the response is restricted to be a rebuttal of a claim made in the speech. A major difference from dialog systems is that in this task the analysis is of a complete speech - rather than taking turns, and the goal is to respond to some of the claims - but not necessarily all.

In the context of computational argumentation much attention has been given to mapping rebuttal or disagreement among arguments. Such works include datasets exemplifying these relations (Walker et al., 2012; Peldszus and Stede, 2015a; Musi et al., 2017), modeling them (Sridhar et al., 2015) and explicitly detecting them (Rosenthal and McKeown, 2015; Peldszus and Stede, 2015b; Wachsmuth et al., 2018). The GPR-KB in this work is reminiscent of argument datasets that depict rebuttal relations, but the arguments are of a different type, being manually authored as general and applicable to a wide range of topics.

Most similar to our work is the task of generating an argument from an opposing stance for a given statement (Hua and Wang, 2018; Hua et al., 2019). These works present a neural-based generative approach, and experiment with user-written posts. Our task differs in that the input is longer text, potentially containing multiple arguments.

## 3 Data

### 3.1 Motions and Speeches

The speeches analyzed in this work are the 200 speeches provided by Mirkin et al. (2018). Each speech debates one of 50 motions originating from iDebate. In this data, the phrasing of the motions is often simplified to include an explicit *topic* and *action*. For example, the iDebate motion *This House would introduce goal line technology in football* is simplified to *We should introduce goal line technology*, where the topic is *goal line technology* and the action is *introduce*.

Speeches are evenly distributed between motions, each having two speeches supporting it (i.e. the speaker is arguing in favor of the motion) and two contesting it. They were recorded by 14 different speakers. A speech is given in several formats. We use the recorded audios and manually-created transcriptions. Recordings are about 4 minutes long, and the transcript texts contain on average 28.7 sentences and 833.1 tokens.

Lastly, the dataset contains claims taken from iDebate along with annotations identifying specific claims mentioned in particular speeches. Herein we refer to this data as iDebate18.

### 3.2 Knowledge base construction

An experienced competitive debater was solicited to author claims that tend to come up in debates across varied topics, and to write a rebuttal argument for each such claim (see the Appendix for the guidelines). She was *not* given access to any of the iDebate18 motions, which are analyzed later on. In total, 39 pairs were constructed in this way.

Texts were allowed to incorporate the special tokens *[ACTION]* and *[TOPIC]*, which are replaced by the debate topic and suggested action when applied to a specific motion or speech. For example, in the context of the motion *We should introduce goal line technology*, the claim *[ACTION] [TOPIC] will encourage better choices* is translated to *introducing goal line technology will encourage better choices*.

In a second phase, the claim-rebuttal pairs were edited by the authors, as follows:

(i) Some rebuttal texts were written with the context of a full speech in mind, and included segments that refer to what a debater would include in such a speech. For example, one included the segment "I have proven that this method is effective". Such segments were edited out.

(ii) For some claims, it seemed that an opposite claim could also be made. In these cases the negation of the claim was also added to the knowledge base, along with an appropriate rebuttal. For example, in addition to the claim *"[ACTION] [TOPIC] is the most practical way to solve the problem."*, we also added the claim *"[ACTION] [TOPIC] is not the most practical way to solve the problem."*.

After these modifications, the final knowledge base includes 55 claim-rebuttal pairs. Claims are always one short single sentence, with an average

| Claim | Rebuttal |
|---|---|
| We must limit personal choice in this case | The greater good means nothing if the rights of individuals are being violated. It doesn't make sense to violate rights in order to protect them. |
| [ACTION] [TOPIC] is good for the economy | While we need to take the economy into account when making decisions, it cannot be the sole consideration or even the top priority in many cases. In this case, the harms outweigh any benefits there may be to the economy. |
| We need to protect the weakest members of society | A truly fair society is one where different people are afforded similar rights and are also trusted to look after themselves. While weaker segments of society can be more vulnerable, this does not justify paternalistic policies that are not beneficial for society as a whole. |

Table 1: Examples of GP-claims and matching rebuttals, created through the process described in §3.2.

length of 8.5 tokens. Rebuttals are longer, on average 1.8 sentences long, and containing on average 32.9 tokens. Three examples from the GPR-KB are given in Table 1; henceforth we refer to the generated claims as *GP-claims*, or simply as *claims* when the context is clear.

## 4 Annotation Experiments

Four annotation experiments are described next, aimed at assessing the applicability of the generated GPR-KB to the real world. Each of the following subsections describes one experiment and its results. An overview of the whole process is depicted in Figure 1. The full annotation guidelines for each experiment appear in the Appendix.

### 4.1 Cross-Topic Relevancy

The GP-claims were written based on the experience of a professional debater, but without context of specific topics. The first annotation experiment aims to establish whether these claims indeed attain the desired goal of being applicable to a varied set of topics. For each motion in iDebate18, and for each GP-claim, we asked annotators to decide whether the claim supports the motion, opposes it or is not relevant[4]. Annotation was done by 7 experienced annotators, and 5 answers were collected for each question.

A GP-claim was considered *relevant* to a motion when marked as supporting or opposing it by most annotators. The stance of relevant claims towards the motion was determined by majority. When a relevant claim has an equal number of supporting and opposing answers, its stance is considered undetermined.

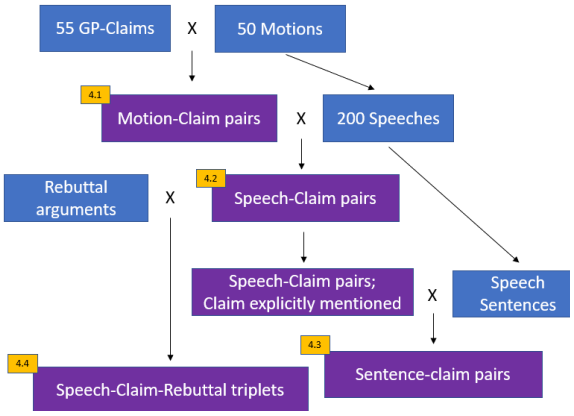


Figure 1: Annotation overview: All motion-claim pairs were annotated for whether the claim is relevant to the motion (see §4.1). For each claim, speeches discussing the relevant motions were annotated for whether the claim was mentioned in the speech (see §4.2), explicitly or implicitly. For explicitly mentioned claims, selected speech sentences were annotated for whether the claim was mentioned in the sentence (see §4.3). In addition, for claims mentioned in the speech, the corresponding Rebuttal Argument was annotated for whether it is a plausible rebuttal in the context of the speech (see §4.4). Blue rectangles indicate textual resources, violet ones indicate annotated resources, yellow ones refer to the relevant subsection.

**Results** Annotation included 2,750 claim-motion pairs[5], of which 46% are claims annotated as relevant to the motion. 20% are annotated as supporting the motion, 26% as opposing it, and a negligible number have an undetermined stance.

On average, 25.4 claims are annotated as relevant per motion. Inter-annotator agreement (average Cohen's Kappa (Cohen, 1960) over pairs of annotators), is 0.52 for the three-labels task, and 0.45 for the binary label of relevant/irrelevant.

Figure 2 shows the distribution of claims vs. the

[4]The stance is required for the experiment in §4.2.

[5]All pairs of 50 motions and 55 claims.

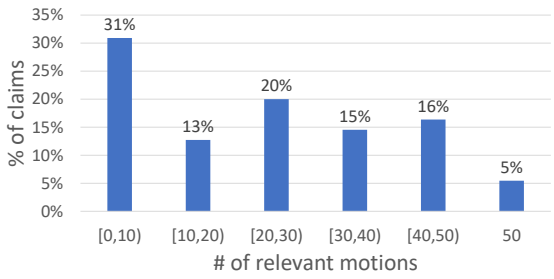


Figure 2: The distribution of GP-claims vs. the number of motions annotated as relevant.

number of motions annotated as relevant. Of note, many claims are relevant to various motions: 56% (sum of the four right-most bars) are relevant to at least 20 (out of 50) motions. However, only 5% are phrased in a manner so general that they may be relevant to all 50 motions. An example of such a general claim is *[ACTION] [TOPIC] will harm others*. At the other extreme, the claim *Animals have rights* is labeled as relevant to only 3 motions discussing various animal-related issues: adopting vegetarianism, banning bullfighting and legalizing ivory trade.

The majority of the claims are therefore general enough to be relevant to a substantial portion of the motions, but not so general as to make them trivially relevant to all motions.

### 4.2 Usage in Spoken Content

Having established that GP-claims are potentially relevant to many motions, the question still remains of whether or not they are actually commonly made by people debating these motions. This is a crucial point for using them as a basis for generating a rebuttal-response.

To assess this, annotators were shown speeches from iDebate18, alongside a matching list of GP-claims determined to be relevant in the previous stage. Specifically, claims annotated as supporting a motion were shown for speeches in which the speaker is arguing in favor of that motion, and vice versa. To allow for a greater number of potential claims, those which at least 2 annotators considered relevant (rather than 3) were included. Claims with an undetermined stance were excluded.

Speeches were presented in both audio and text formats, and annotators were allowed to choose between listening, reading or both. They then had to determine whether each claim is mentioned in the speech *explicitly*, *implicitly* or not at all. The number of claims presented for each speech was limited to 20; in case a larger number was determined to be relevant, the question was split into chunks of 20 claims.

In total, 3,246 claim-speech pairs required annotation, almost four times more than the corresponding annotation included in iDebate18. Annotation was done using the *Figure-Eight*[6] crowdsourcing platform, with 10 annotators per question. Clearly, this is a challenging task for the crowd, and hence a selected group of 22 annotators was used. Selection was based on their past performance on other tasks done by our team.

To further validate the annotation, the list of claims presented for each speech included claims for which the correct label was known a-priori. These include claims annotated in the previous experiment as irrelevant for the motion, for which it is assumed that the correct label is "not mentioned". In addition, annotation was done in batches. Claim-speech pairs for which unanimous answers were obtained in earlier batches were included in newer ones, with the correct label assumed to be this unanimous answer.

A claim is considered *mentioned* in a speech if a majority labeled it as mentioned (i.e. summing up implicit and explicit answer counts). Otherwise it is considered as *not mentioned*. A mentioned claim is *explicit* in the speech if its explicit answers count strictly exceeds its implicit answers count. Otherwise, it is considered *implicit*.

**Results** 41% of claim-speech pairs were labeled as mentioned (7% explicit and 34% implicit). On average, each claim is explicit in 4.3 speeches, and implicit in 17.6 speeches.

Pairwise inter-annotator agreement is 0.37 when considering two labels: mentioned or not. The average error rate of all annotators, on questions with a prior known answer, is 7%, suggesting a relatively high-quality annotation.

The *prior* of a claim is defined as the number of speeches in which it was found to be mentioned, divided by the number of speeches in which it was labeled. Figure 3 depicts the percentage of claims vs. their prior, separately for explicit and implicit mentions. Some claims are never mentioned in any speech: 20% are never mentioned explicitly and ∼10% are not mentioned at all, not even implicitly. Note that for the most part, claims

[6] www.figure-eight.com (formerly CrowdFlower).

are mentioned in less than half of the speeches for which they may be relevant.

In conclusion, the results suggest that GP-claims are often used in spoken content discussing various topics, and that this is not due to a small subset of trivial claims. Rather, most claims appear at least once, but usually in no more than half of the speeches for relevant motions.

These properties make automatic detection of these claims in speeches an interesting and challenging task.

**Comparison to iDebate18** Table 2 compares the results of this annotation to that of iDebate18, which contains topic-specific claims annotated for the same set of speeches[7].

Surprisingly, topic-specific claims are no more likely to occur in speeches discussing that particular topic. Moreover, the larger number of potential GP-claims leads to a higher absolute number of mentions, to the extent that – in contrast with iDebate18 – all speeches include at least one mention. Hence, the GPR-KB augments the iDebate18 dataset, both by increasing the number of claims that are to be sought in a speech, and by suggesting claims to speeches for which iDebate18 does not contain any.

| Speech stat. | GP-claims | iDebate18 |
|---|---|---|
| Coverage | 100% | 86.5% |
| Avg. Mentions | 6.7 | 1.8 |
| Avg. Potential | 16.2 | 4.4 |

Table 2: A comparison of GP-claims and topic-specific iDebate claims annotation. Coverage is the percentage of speeches with at least one claim annotated as mentioned. Avg. Potential is the average number of possibly relevant claims per speech, and Avg. Mentions is the average number of claims annotated as mentioned.

### 4.3 Where was it said?

A straightforward approach to determining whether a claim was mentioned in a speech is to go over its sentences, one by one, and decide whether the claim is equivalent to a sentence or implied by it, as indeed is done in Mirkin et al. (2018). This is a challenging task since, as described above, most mentions are implicit. In many cases one can not point to a single sentence

[7]Available answers in iDebate18 are *mentioned* or *not mentioned.*

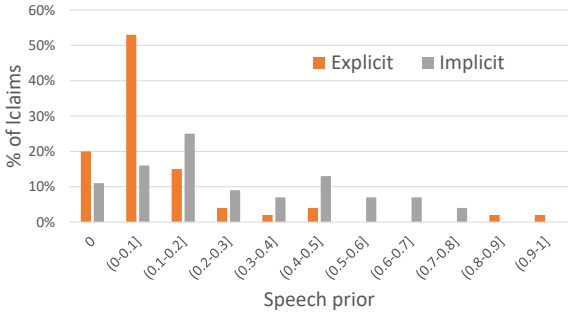


Figure 3: The distribution of GP-claims vs. speech prior (the percentage of labeled speeches in which a claim is mentioned), for explicit or implicit mentions.

mentioning the claim, as the claim is implied by the general stance of the speaker.

Even when a sentence does imply a claim, automatically inferring that may be hard. For example, for the motion *We should end cheerleading*, a relevant opposing claim is *Ending cheerleading limits personal choice*. We identified a sentence implying it, *The only clear moral system we can derive is one in which we value individual preference*, yet it seems hard to deduce this automatically without considering the surrounding context which clarifies that the argument is about personal choice.

An annotation task for identifying *where* a claim was mentioned is considerably more difficult than the aforementioned annotations. Determining ground truth is far from trivial, as annotators may point to different sentences within the same argument as being the location of the claim.

Nonetheless, such information seems a valuable part of a GPR-KB. To provide at least a partial solution, we annotated claim-sentence pairs directly, asking whether the claim is mentioned within the sentence. Algorithms developed on such data can then predict a claim as mentioned in a speech when it is mentioned in one of its sentences. This form of annotation is simple and facilitates easier collection of a large number of labels. To enable research in this direction, such annotations were performed both for GP-claims and, since none are provided in iDebate18, for claims from iDebate.

A careful selection of pairs to annotate is required since there are too many pairs for a comprehensive labeling, and sampling at random would rarely yield a pair such that the claim is mentioned in the sentence. Thus, we limited annotation to claims which were labeled as mentioned explicitly (and assumed all iDebate claims to be

so), and paired them only with sentences which are somewhat similar to them (based on word2vec, Mikolov et al., 2013). Annotation was again done by 10 crowd annotators.

**Results** Annotation included 4,271 GP-claim and sentence pairs and 2,164 iDebate-claim and sentence pairs, with a similarity of at least 0.5 and 0.7 (resp.) between claim and sentence. The usage of general crowd required some quality control. Annotators not meeting one or more of the following criteria were removed, along with their answers: Answer at least 10 questions; have at least 5 common answers with 3 different peers; have average agreement with peers $\geq 0.2$.

The resulting inter-annotator agreement was 0.55 for GP-claims and 0.46 for iDebate claims. Considering only pairs with at least 5 remaining answers, after filtering out annotators as described above, 20% of the GP-claims-sentence pairs were annotated as a match (and 17% of iDebate pairs).

### 4.4 Validity of Rebuttal Arguments

Recall that rebuttal arguments in the GPR-KB were written without any specific contexts in which they are to be used. Hence, even if the claim they respond to is indeed mentioned in a speech, it is not clear whether the pre-written rebuttal would consist a plausible rebuttal response to the speech.

We assessed the effectiveness of the rebuttal arguments using a two-step procedure. First, as in §4.2, annotators were shown a speech and a claim, and determined whether the claim is mentioned in the speech. Then, if they marked that claim as mentioned, its pre-written rebuttal was shown, and they were asked whether it is a plausible response to the mentioned claim *in the context of the speech*.

This two-step annotation procedure was chosen for three reasons. First, requiring annotators to assess whether a claim is mentioned in a speech motivates them to review its content again and locate the relevant parts in which the claim is expressed. Second, it prevents irrelevant answers from those who do not think that the claim is mentioned in the speech. Finally, asking again about a claim's mention enables result validation, when the answer is known a-priori with high confidence. Specifically, claims for which the annotation is unanimous were used for this purpose.

For each claim, we annotated two randomly sampled speeches mentioning it. This amounted to 103 rebuttal-speech pairs, since not all 55 GP-claims were mentioned in two speeches. We relied on the same group of crowd annotators who took part in the previous experiment, and once again required 10 answers for each question.

**Results** Measuring agreement using Cohen's Kappa in this scenario is problematic. First, the label distribution is very biased: If rebuttal arguments are always plausible responses, then the correct answer is always yes. Any deviation from that will greatly reduce the score, making it an ill-fitting measure for such data (Jeni et al., 2013). Second, the small number of questions leads to many annotator-pairs whose set of common questions, on which this score is computed, is rather small. This makes the computation unstable, since when averaging over all pairs, such small intersections contribute as much as the large ones.

Instead, looking at the majority annotation for each rebuttal argument, we observed that in 87% of the cases it indicates that the rebuttal is plausible. This suggests that regardless of whether annotators agree with one another, they tend to agree that the rebuttal is usually plausible. Moreover, computing the average kappa between annotators and the majority annotation, and considering only those annotators who answered at least 20 questions, yields 0.47. This value for such biased data, alongside an average error rate of 4% on the questions with known answers, suggests that the annotation is of reasonable quality.

**Analysis** The results above show most rebuttals are appropriate in the vast majority of contexts. We therefore decided that continuing with this costly annotation is not needed. Furthermore, manual analysis of cases in which the rebuttal was unanimously found inappropriate showed this stemmed from the rebuttal being inappropriate for the topic, rather than a specific speech. For example, when discussing goal-line technology, in response to the claim *"introducing goal-line technology will lead to greater problems in the future"*, the pre-written rebuttal is *"Governments have an obligation to their citizens in the here and now. The better off society is today, the more resources we will have to make the future better when it comes"*. Such a response makes several assumptions which are clearly violated here, such as the involvement of government. Thus, further validation of rebuttals may benefit from first verifying their relevancy to the topic.

### 4.5 The GPR-KB-55 Dataset

The annotation results show that it is possible to construct a concise set of general claims, such that in most speeches at least one of them will come up. Furthermore, they show that a rebuttal to these claims can be authored independently of the specific motions and speeches, while nonetheless being a plausible response in their context. Table 3 summarizes the statistics for the pair-annotation experiments. The resulting dataset is freely available, and is one of the main contributions of this work[8]. We name the new dataset GPR-KB-55.

| Annotation | # pairs | # positives |
|---|---|---|
| Motion – GP-claim | 2,750 | 1,265 |
| Speech – GP-claim | 3,246 | 1,491 |
| Sentence – GP-claim | 4,271 | 854 |
| Sentence – iDebate claim | 2,164 | 368 |

Table 3: Summary of annotation experiment results. Positive pairs are those in which a majority of the annotators indicated that the claim is relevant (motion) or mentioned (speech, sentence).

## 5 Detecting claims in speeches

Next, we establish baseline results for determining whether a GP-claim is mentioned in a speech, and compare them to results obtained for iDebate claims. For a fair comparison of the two data sources, we assume for both prior knowledge as to which claims are relevant for a motion, as well as their stance towards it. Hence, we take the labeling of GP-claims to motions (described in §4.1) as given. The following algorithms are considered:

**word2vec** The best performing baseline of Mirkin et al. (2018) utilizes a detailed description of each iDebate claim, comprised of several sentences. It examines the speech sentence by sentence, and for each sentence computes its tf-idf weighted word2vec (*w2v*) similarity to the detailed claim description. A claim is then scored by taking the maximum over all claim-sentence similarity scores. We use this method (*w2v-i18*) as a baseline for the GP-claims as well, yet sentences are scored by their similarity to the GP-claim text, since no detailed topic-specific description exists. The latter is referred to as *w2v-GP-claims*. For comparison, we repeat the experiment using only the iDebate claim texts (*w2v-i18-claims*).

**Bert** Recently, the Bert architecture (Devlin et al., 2018) has proven successful on similar tasks, and we provide its results as an additional baseline. Specifically, we select at random 80% of the motions as an ad-hoc train set, and the remainder as a test set (*bert-test*). Bert was trained on labeled claim-sentence pairs corresponding to motions from the train set, in two settings, considering: (i) GP-claims (∼3K pairs) – *bert*, and (ii) both GP-claims and iDebate claims (almost 5K pairs) – *bert+*. In inference, given a claim and a speech, sentences semantically similar to the claim (as in §4.3) are scored by the fine-tuned network. Their maximum is the outputted claim-speech score.

**Prior** One important difference between GP-claims and iDebate claims is that the same GP-claim can (and does) appear across different motions and speeches. Specifically, given a training set, the a-priori probability that a GP-claim will be mentioned in a speech can be computed. Then, test claims are scored with their computed a-priori probability *without considering the text of the speech*. This baseline is referred to as *prior*.

**Results** Figure 4 plots precision-recall curves comparing claim detection baselines over iDebate claims and GP-claims. As observed by Mirkin et al. (2018), *w2v* works best when given a detailed iDebate claim description. Without it, performance is comparable for the two claim sources, and is rather poor for both. *Prior* results were obtained by using a leave-one-motion-out cross validation: at each fold a single motion is left out and the others are used for training. Its precision-recall curve shows that when considering these statistics, it presents a challenging baseline.

To compare the *bert* baseline to others, the precision-recall curves for both *prior*[9] and *w2v* were computed over speeches from *bert-test*. As shown in Figure 5, while *bert* clearly outperforms *w2v*, it nonetheless does not reach the *prior* baseline. The additional data provided in training to *bert+* does not help.

Note that this comparison is between methods which are derived from different types of data. Here *bert* is trained only on explicitly-mentioned claims, with respect to (ostensibly) semantically

[8] https://www.research.ibm.com/haifa/dept/vst/debating_data.shtml

[9] Here, a-priori probabilities were computed on all motions not in *bert-test*, then applied to motions of *bert-test*.

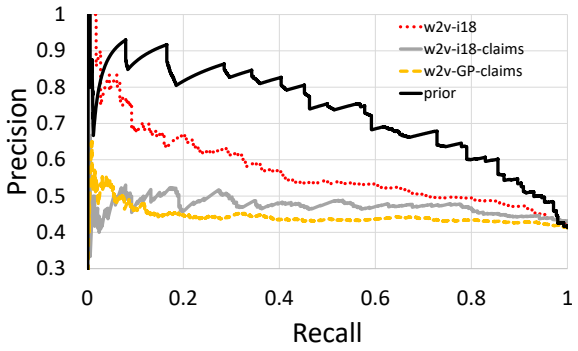


Figure 4: Precision-Recall curves for the matching of GP-claims and iDebate claims to all 200 speeches.

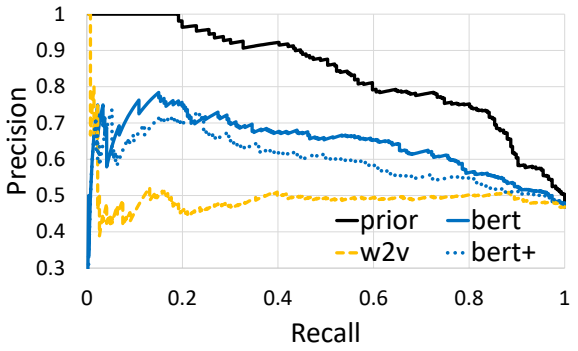


Figure 5: Precision-Recall curves for matching GP-claims to speeches, for *bert-test* (20% of motions).

similar *sentences*. On the other hand, the *prior* baseline is computed based on all claims, and their annotation w.r.t. *the entire speech*. This may be part of the reason why *bert*, which has proven to be successful on many NLP tasks, here achieves lower performance than this simple baseline.

**Analysis** Although *prior* seems like a strong baseline in terms of precision and recall, it is probably not a desired solution by itself, since it simply produces high probability responses regardless of the rebutted content. For example, among its top-20% predictions, precision is 83% and recall is 40%, yet they include only 22% of available GP-claims. Moreover, 77% of these top-20% are the same 5 claims. This reflects a property of the data: there are a few claims which are relevant to many motions, and are also implicitly mentioned in most speeches. Detection algorithms should be aware of this property, and account for it when evaluating performance. At the same time, a claim's acceptance prior can be useful for inference. For example, it could be combined with other data in a more sophisticated algorithm, or could direct the parameter choice of such an algorithm.

## 6 Conclusions and Future Work

We presented the problem of producing a rebuttal response to a long argumentative text. This task is especially challenging when the discussed topic is not known in advance, and, accordingly, potential responses are not readily available.

Toward the goal of addressing this problem we constructed a multi-layered dataset: (i) A Knowledge base of GP-claims and corresponding rebuttal arguments, which are shown to be applicable for a wide variety of topics; (ii) A mapping of these claims to motions of iDebate18 in which they might be applicable; (iii) An annotation of the stance of applicable claims; (iv) An annotation for which claims are actually mentioned in relevant speeches, and whether they are mentioned explicitly or implicitly; (v) For explicitly mentioned claims, a (partial) annotation of which sentences imply them and which do not.

In addition, we presented baselines for the related Listening Comprehension task, suggesting that this is a complicated problem. Using state-of-the-art sentence embedding yielded an $F1$-score of 0.64, while trivially taking the claim with the highest prior to be mentioned scored 0.78[10]. This suggests that careful evaluation is required.

While baselines are provided only for detecting GP-claims in spoken content, future work should aim to solve the problem as a whole - either by developing algorithms that determine relevance and stance of GP-claims to given motions, or by forgoing these stages, and successfully deciding whether a claim was mentioned in a speech, without first focusing on relevant claims.

Our results suggest that GP rebuttal arguments usually work well as a response to speeches in which the matching claim was mentioned. However, this is by no means perfect; and in some 13% of the cases they do not. It is interesting to further identify and understand these cases. By doing so, an automatic system could prefer responses that it identifies as more appropriate. Moreover, understanding such cases can lead to improving the rebuttal arguments themselves.

## 7 Acknowledgments

We thank the anonymous reviewers for their valuable comments, and Hayah Eichler for creating the initial GPR-KB.

[10] Scores computed at the precision-recall intersection.

## References


Yonatan Bilu, Ariel Gera, Daniel Hershcovich, Benjamin Sznajder, Dan Lahav, Guy Moshkowich, Anael Malet, Assaf Gavron, and Noam Slonim. 2019. Argument invention from first principles. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1013–1026, Florence, Italy. Association for Computational Linguistics.

Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. 2017. A survey on dialogue systems: Recent advances and new frontiers. *ACM SIGKDD Explorations Newsletter*, 19(2):25–35.

Jacob Cohen. 1960. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20(1):37–46.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Phan Minh Dung. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial intelligence*, 77(2):321–357.

Xinyu Hua, Zhe Hu, and Lu Wang. 2019. Argument generation with retrieval, planning, and realization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2661–2672, Florence, Italy. Association for Computational Linguistics.

Xinyu Hua and Lu Wang. 2018. Neural argument generation augmented with externally retrieved evidence. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 219–230, Melbourne, Australia. Association for Computational Linguistics.

László A Jeni, Jeffrey F Cohn, and Fernando De La Torre. 2013. Facing imbalanced data–recommendations for the use of performance metrics. In *2013 Humaine Association Conference on Affective Computing and Intelligent Interaction*, pages 245–251. IEEE.

Tamar Lavee, Matan Orbach, Lili Kotlerman, Yoav Kantor, Shai Gretz, Lena Dankin, Shachar Mirkin, Michal Jacovi, Yonatan Bilu, Ranit Aharonov, and Noam Slonim. 2019. Towards effective rebuttal: Listening comprehension using corpus-wide claim mining. *6th Workshop on Argument Mining*.

Pierre-Emmanuel Mazaré, Samuel Humeau, Martin Raison, and Antoine Bordes. 2018. Training millions of personalized dialogue agents. In *EMNLP*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Shachar Mirkin, Guy Moshkowich, Matan Orbach, Lili Kotlerman, Yoav Kantor, Tamar Lavee, Michal Jacovi, Yonatan Bilu, Ranit Aharonov, and Noam Slonim. 2018. Listening comprehension over argumentative content. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 719–724. Association for Computational Linguistics.

Elena Musi, Debanjan Ghosh, Mark Aakhus, Smaranda Muresan, and Nina Wacholder. 2017. Building an ontology of (dis) agreement space for argument mining. *SIGDIAL/SEMDIAL 2017 Joint Session on Negotiation Dialog (position paper)*.

Andreas Peldszus and Manfred Stede. 2015a. An annotated corpus of argumentative microtexts. In *Proceedings of the First Conference on Argumentation, Lisbon, Portugal, June. to appear*.

Andreas Peldszus and Manfred Stede. 2015b. Towards detecting counter-considerations in text. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 104–109.

Alan Ritter, Colin Cherry, and William B Dolan. 2011. Data-driven response generation in social media. In *Proceedings of the conference on empirical methods in natural language processing*, pages 583–593. Association for Computational Linguistics.

Sara Rosenthal and Kathy McKeown. 2015. I couldnt agree more: The role of conversational structure in agreement and disagreement detection in online discussions. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 168–177.

Andrew Shin, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. 2015. Context-dependent automatic response generation using statistical machine translation techniques. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1345–1350.

Dhanya Sridhar, James Foulds, Bert Huang, Lise Getoor, and Marilyn Walker. 2015. Joint models of disagreement and stance in online debate. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 116–125.

Hiroaki Sugiyama, Toyomi Meguro, Ryuichiro Higashinaka, and Yasuhiro Minami. 2013. Open-domain utterance generation for conversational dialogue systems using web-scale dependency structures. In *Proceedings of the SIGDIAL 2013 Conference*, pages 334–338.

Henning Wachsmuth, Shahbaz Syed, and Benno Stein. 2018. Retrieval of the best counterargument without prior topic knowledge. In *Proceedings of the*

*56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 241–251, Melbourne, Australia. Association for Computational Linguistics.

Marilyn A Walker, Jean E Fox Tree, Pranav Anand, Rob Abbott, and Joseph King. 2012. A corpus for research on deliberation and debate. In *LREC*, pages 812–817. Istanbul.

Joseph Weizenbaum. 1966. Eliza–a computer program for the study of natural language communication between man and machine. *Commun. ACM*, 9(1):36–45.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Lina Maria Rojas-Barahona, Pei-Hao Su, Stefan Ultes, David Vandyke, and Steve J. Young. 2016. A network-based end-to-end trainable task-oriented dialogue system. In *EACL*.

Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2017. Topic aware neural response generation. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Rui Yan, Yiping Song, and Hua Wu. 2016. Learning to respond with deep neural networks for retrieval-based human-computer conversation system. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 55–64. ACM.

# Rethinking Attribute Representation and Injection for Sentiment Classification

**Reinald Kim Amplayo**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
reinald.kim@ed.ac.uk

## Abstract

Text attributes, such as user and product information in product reviews, have been used to improve the performance of sentiment classification models. The de facto standard method is to incorporate them as additional biases in the attention mechanism, and more performance gains are achieved by extending the model architecture. In this paper, we show that the above method is the least effective way to represent and inject attributes. To demonstrate this hypothesis, unlike previous models with complicated architectures, we limit our base model to a simple BiLSTM with attention classifier, and instead focus on *how* and *where* the attributes should be incorporated in the model. We propose to represent attributes as chunk-wise importance weight matrices and consider four locations in the model (i.e., embedding, encoding, attention, classifier) to inject attributes. Experiments show that our proposed method achieves significant improvements over the standard approach and that attention mechanism is the worst location to inject attributes, contradicting prior work. We also outperform the state-of-the-art despite our use of a simple base model. Finally, we show that these representations transfer well to other tasks[1].

## 1 Introduction

The use of categorical attributes (e.g., user, topic, aspects) in the sentiment analysis community (Kim and Hovy, 2004; Pang and Lee, 2007; Liu, 2012) is widespread. Prior to the deep learning era, these information were used as effective categorical features (Li et al., 2011; Tan et al., 2011; Gao et al., 2013; Park et al., 2015) for the machine learning model. Recent work has used them to improve the overall performance (Chen et al., 2016; Dong et al., 2017), interpretability (Amplayo et al., 2018a; Angelidis and Lapata, 2018), and personalization (Ficler and Goldberg, 2017) of neural network models in different tasks such as sentiment classification (Tang et al., 2015), review summarization (Yang et al., 2018a), and text generation (Dong et al., 2017).

In particular, user and product information have been widely incorporated in sentiment classification models, especially since they are important metadata attributes found in review websites. Tang et al. (2015) first showed significant accuracy increase of neural models when these information are used. Currently, the accepted standard method is to use them as additional biases when computing the weights $a$ in the attention mechanism, as introduced by Chen et al. (2016) as:

$$a = \text{softmax}(v^\top e) \quad (1)$$
$$e = \tanh(Wh + W_u u + W_p p + b) \quad (2)$$
$$= \tanh(Wh + b_u + b_p + b) \quad (3)$$
$$= \tanh(Wh + b') \quad (4)$$

where $u$ and $p$ are the user and product embeddings, and $h$ is a word encoding from BiLSTM. Since then, most of the subsequent work attempted to improve the model by extending the model architecture to be able to utilize external features (Zhu and Yang, 2017), handle cold-start entities (Amplayo et al., 2018a), and represent user and product separately (Ma et al., 2017).

Intuitively, however, this method is not the ideal method to represent and inject attributes because of two reasons. First, representing attributes as additional biases cannot model the relationship between the text and attributes. Rather, it only adds a user- and product-specific biases that are independent from the text when calculating the attention weights. Second, injecting the attributes in the attention mechanism means that user and prod-

[1] Model implementation and datasets are released here: https://github.com/rktamplayo/CHIM



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5602–5613,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

uct information are only used to customize how the model choose which words to focus on, as also shown empirically in previous work (Chen et al., 2016; Ma et al., 2017). However, we argue that there are more intuitive locations to inject the attributes such as when contextualizing words to modify their sentiment intensity.

We propose to represent user and product information as weight matrices (i.e., $W$ in the equation above). Directly incorporating these attributes into $W$ leads to large increase in parameters and subsequently makes the model difficult to optimize. To mitigate these problems, we introduce chunk-wise importance weight matrices, which (1) uses a weight matrix smaller than $W$ by a chunk size factor, and (2) transforms these matrix into gates such that it corresponds to the relative importance of each neuron in $W$. We investigate the use of this method when injected to several locations in the base model: word embeddings, BiLSTM encoder, attention mechanism, and logistic classifier.

The results of our experiments can be summarized in three statements. First, our preliminary experiments show that doing bias-based attribute representation and attention-based injection is not an effective method to incorporate user and product information in sentiment classification models. Second, despite using only a simple BiLSTM with attention classifier, we significantly outperform previous state-of-the-art models that use more complicated architectures (e.g., models that use hierarchical models, external memory networks, etc.). Finally, we show that these attribute representations transfer well to other tasks such as product category classification and review headline generation.

## 2 How and Where to Inject Attributes?

In this section, we explore different ways on how to represent attributes and where in the model can we inject them.

### 2.1 The Base Model

The majority of this paper uses a base model that accepts a review $\mathbf{x} = x_1, ..., x_n$ as input and returns a sentiment $y$ as output, which we extend to also accept the corresponding user $u$ and product $p$ attributes as additional inputs. Different from previous work where models use complex architectures such as hierarchical LSTMs (Chen et al., 2016; Zhu and Yang, 2017) and external memory networks (Dou, 2017; Long et al., 2018), we aim to achieve improvements by only modifying how we represent and inject attributes. Thus, we use a simple classifier as our base model, which consists of four parts explained briefly as follows.

First, we embed $\mathbf{x}$ using a word embedding matrix that returns word embeddings $x'_1, ..., x'_n$. We subsequently apply a non-linear function to each word:

$$w_t = \tanh(W_{\text{emb}} x'_t + b_{\text{emb}}) \tag{5}$$

Second, we run a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) encoder to contextualize the words into $h_t = [\overrightarrow{h}_t; \overleftarrow{h}_t]$ based on their forward and backward neighbors. The forward and backward LSTM look similar, thus for brevity we only show the forward LSTM below:

$$\begin{bmatrix} g_t \\ i_t \\ f_t \\ o_t \end{bmatrix} = \begin{bmatrix} \tanh \\ \sigma \\ \sigma \\ \sigma \end{bmatrix} W_{\text{enc}}[w_t; \overrightarrow{h}_{t-1}] + b_{\text{enc}} \tag{6}$$

$$c_t = f_t * c_{t-1} + i_t * g_t \tag{7}$$

$$\overrightarrow{h}_t = o_t * c_t \tag{8}$$

Third, we pool the encodings $h_t$ into one document encoding $d$ using attention mechanism (Bahdanau et al., 2015), where $v$ is a latent representation of informativeness (Yang et al., 2016):

$$e_t = \tanh(W_{\text{att}} h_t + b_{\text{att}}) \tag{9}$$

$$a_t = \text{softmax}_t(v^\top e_t) \tag{10}$$

$$d = \sum_t (a_t * h_t) \tag{11}$$

Finally, we classify the document using a logistic classifier to get a predicted $y'$:

$$y' = \text{argmax}(W_{\text{cls}} d + b_{\text{cls}}) \tag{12}$$

Training is done normally by minimizing the cross entropy loss.

### 2.2 How: Attribute Representation

Note that at each part of the model, we see similar non-linear functions, all using the same form, i.e. $g(f(x)) = g(Wx + b)$, where $f(x)$ is an affine transformation function of $x$, $g$ is a non-linear activation, $W$ and $b$ are weight matrix and bias parameters, respectively. Without extending the base model architecture, we can represent the attributes either as the weight matrix $W$ or as the bias $b$ to one of these functions by modifying them to accept $u$ and $p$ as inputs, i.e. $f(x, u, p)$.

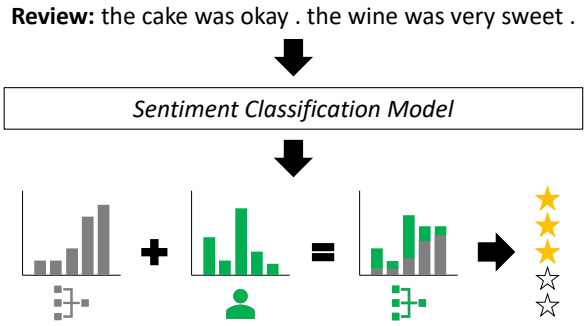


(a) Logits when representing attributes as biases in the logistic classifier

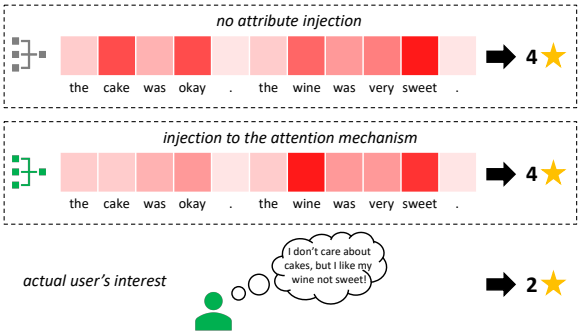


(b) Attention weights when injecting attributes in the attention mechanism.

Figure 1: Illustrative examples of issues when representing attributes as biases and injecting them in the attention mechanism. The gray process icon indicates the model without incorporating attributes, while the same icon in green indicates the model customized for the green user.

**Bias-based** The current accepted standard approach to represent the attributes is through the bias parameter $b$. Most of the previous work (Chen et al., 2016; Zhu and Yang, 2017; Amplayo et al., 2018a; Wu et al., 2018) use Equation 2 in the attention mechanism, which basically updates the original bias $b$ to $b' = W_u u + W_p p + b$. However, we argue that this is not the ideal way to incorporate attributes since it means we only add a user- and product-specific bias towards the goal of the function, *without* looking at the text. Figure 1a shows an intuitive example: When we represent user $u$ as a bias in the logistic classifier, in which it means that $u$ has a biased logits vector $b_u$ of classifying the text as a certain sentiment (e.g., $u$ tends to classify texts as three-star positive), shifting the final probability distribution regardless of what the text content may have been.

**Matrix-based** A more intuitive way of representing attributes is through the weight matrix $W$. Specifically, given the attribute embeddings $u$ and $p$, we linearly transform their concatenation into a vector $w'$ of size $D_1 * D_2$ where $D_1$ and $D_2$ are the dimensions of $W$. We then reshape $w'$ into $W'$ to get the same shape as $W$ and replace $W$ with $W'$:

$$w' = W_c[u; p] + b_c \quad (13)$$

$$W' = \text{reshape}(w', (D_1 \times D_2)) \quad (14)$$

$$f(x, u, p) = W'x + b \quad (15)$$

Theoretically, this should perform better than bias-based representations since direct relationship between text and attributes are modeled. For example, following the example above, $W'x$ is a user-biased logits vector based on the document encoding $d$ (e.g., $u$ tends to classify texts as two-star positive when the text mentions that the dessert was sweet).

However, the model is burdened by a large number of parameters; matrix-based attribute representation increases the number of parameters by $|U| * |P| * D_1 * D_2$, where $|U|$ and $|P|$ correspond to the number of users and products, respectively. This subsequently makes the weights difficult to optimize during training. Thus, directly incorporating attributes into the weight matrix may cause harm in the performance of the model.

**CHIM-based** We introduce **Ch**unk-wise **I**mportance **M**atrix (CHIM) based representation, which improves over the matrix-based approach by mitigating the optimization problems mentioned above, using the following two tricks. First, instead of using a big weight matrix $W'$ of shape $(D_1, D_2)$, we use a chunked weight matrix $C$ of shape $(D_1/C_1, D_2/C_2)$ where $C_1$ and $C_2$ are chunk size factors. Second, we use the chunked weight matrix as importance gates that shrinks the weights close to zero when they are deemed unimportant. We show the CHIM-based representation method in Figure 2.

We start by linearly transforming the concatenated attributes into $c$. Then we reshape $c$ into $C$ with shape $(D_1/C_1, D_2/C_2)$. These operations are similar to Equations 13 and 14. We then repeat this matrix $C_1 * C_2$ times and concatenate them such that we create a matrix $W'$ of shape $(D_1, D_2)$. Finally, we use the sigmoid function $\sigma$ to transform the matrix into gates that represent importance:

$$W' = \sigma \begin{bmatrix} C; ...; C \\ ...; ...; ... \\ C; ...; C \end{bmatrix} \in [0, 1]^{D_1 \times D_2} \quad (16)$$

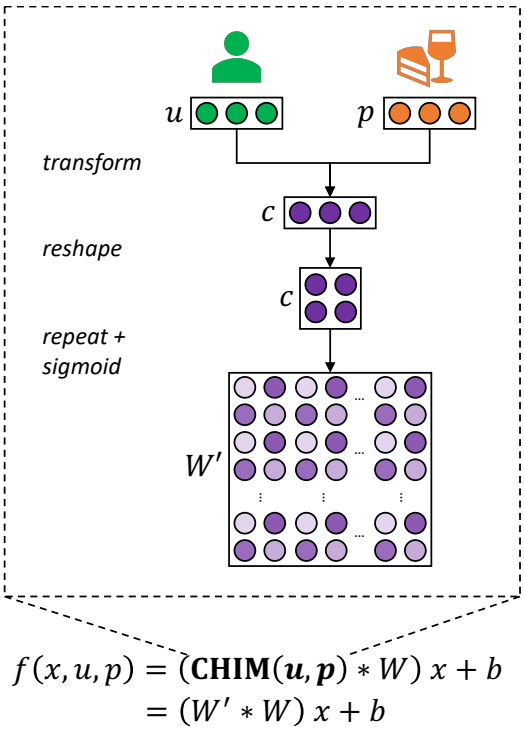


Figure 2: CHIM-based attribute representation and injection to a non-linear funtion in the model.

Finally we broadcast-multiply $W'$ with the original weight matrix $W$ to shrink the weights. The result is a sparse version of $W$, which can be seen as either a regularization step (Ng, 2004) where most weights are set close to zero, or a correction step (Amplayo et al., 2018b) where the important gates are used to correct the weights. The use of multiple chunks regards CHIM as coarse-grained access control (Shen et al., 2019) where the use of different important gates for every node is unnecessary and expensive. The final function is shown below:

$$f(x, u, p) = (W' * W)x + b \quad (17)$$

To summarize, chunking helps reduce the number of parameters while retaining the model performance, and importance matrix makes optimization easier during training, resulting to a performance improvement. We also tried alternative methods for importance matrix such as residual addition (i.e., $\tanh(W') + W$) introduced in He et al. (2016), and low-rank adaptation methods (Jaech and Ostendorf, 2018; Kim et al., 2019), but these did not improve the model performance.

### 2.3 Where: Attribute Injection

Using the approaches described above, we can inject attribute representation into four different parts of the model. This section describes what it means to inject attributes to a certain location and why previous work have been injecting them in the worst location (i.e., in the attention mechanism).

**In the attention mechanism** Injecting attributes to the attention mechanism means that we bias the selection of more informative words during pooling. For example, in Figure 1b, a user may find delicious drinks to be the most important aspect in a restaurant. Injection in the attention mechanism would bias the selection of words such as *wine*, *smooth*, and *sweet* to create the document encoding. This is the standard location in the model to inject the attributes, and several (Chen et al., 2016; Amplayo et al., 2018a) have shown how the injected attention mechanism selects different words when the given user or product is different.

We argue, however, that attention mechanism is not the best location to inject the attributes. This is because we cannot obtain user- or product-biased sentiment information from the representation. In the example above, although we may be able to select, with user bias, the words *wine* and *sweet* in the text, we do not know whether the user has a positive or negative sentiment towards these words (e.g., Does the user like wine? How about *sweet* wines? etc.). In contrast, the three other locations we discuss below use the attributes to modify how the model looks at sentiment at different levels of textual granularity.

**In the word embedding** Injecting attributes to the word embedding means that we bias the sentiment intensity of a word independent from its neighboring context. For example, if a user normally uses the words *tasty* and *delicious* with a less and more positive intensity, respectively, the corresponding attribute-injected word embeddings would come out less similar, despite both words being synonymous.

**In the BiLSTM encoder** Injecting attributes to the encoder means that we bias the contextualization of words based on their neighbors in the text. For example, if a user likes their cake sweet but their drink with no sugar, the attribute-injected encoder would give a positive signal to the encoding of *sweet* in the text "*the cake was sweet*" and a negative signal in the text "*the drink was sweet*".

**In the logistic classifier** Injecting attributes to the classifier means that we bias the probability

| Datasets | C | #train | #dev | #test | #users | #products | #docs/user | #docs/product |
|---|---|---|---|---|---|---|---|---|
| IMDB | 10 | 67,426 | 8,381 | 9,112 | 1,310 | 1,635 | 64.82 | 51.94 |
| Yelp 2013 | 5 | 62,522 | 7,773 | 8,671 | 1,631 | 1,633 | 48.42 | 48.36 |
| Yelp 2014 | 5 | 183,019 | 22,745 | 25,399 | 4,818 | 4,194 | 47.97 | 55.11 |

Table 1: Statistics of the datasets used for the Sentiment Classification task.

distribution of sentiment based on the final document encoding. If a user tends to classify the sentiment of reviews about sweet cakes as highly positive, then the model would give a high probability to highly positive sentiment classes for texts such as "*the cake was sweet*".

## 3 Experiments

### 3.1 General Setup

We perform experiments on two tasks. The first task is Sentiment Classification, where we are tasked to classify the sentiment of a review text, given additionally the user and product information as attributes. The second task is Attribute Transfer, where we attempt to transfer the attribute encodings learned from the sentiment classification model to solve two other different tasks: (a) Product Category Classification, where we are tasked to classify the category of the product, and (b) Review Headline Generation, where we are tasked to generate the title of the review, given only both the user and product attribute encodings. Datasets, evaluation metrics, and competing models are different for each task and are described in their corresponding sections.

Unless otherwise stated, our models are implemented with the following settings. We set the dimensions of the word, user, and product vectors to 300. We use pre-trained GloVe embeddings[2] (Pennington et al., 2014) to initialize the word vectors. We also set the dimensions of the hidden state of BiLSTM to 300 (i.e., 150 dimensions for each of the forward/backward hidden state). The chunk size factors $C_1$ and $C_2$ are both set to 15. We use dropout (Srivastava et al., 2014) on all non-linear connections with a dropout rate of 0.1. We set the batch size to 32. Training is done via stochastic gradient descent over shuffled mini-batches with the Adadelta update rule (Zeiler, 2012) and with $l_2$ constraint (Hinton et al., 2012) of 3. We perform early stopping using the development set. Training and experiments are done using an NVIDIA GeForce GTX 1080 Ti graphics card.

### 3.2 Sentiment Classification

**Datasets and Evaluation** We use the three widely used sentiment classification datasets with user and product information available: IMDB, Yelp 2013, and Yelp 2014 datasets[3]. These datasets are curated by Tang et al. (2015), where they ensured twenty-core for both users and products (i.e., users have at least twenty products and vice versa), split them into train, dev, and test sets with an 8:1:1 ratio, and tokenized and sentence-split using the Stanford CoreNLP (Manning et al., 2014). Dataset statistics are shown in Table 1. Evaluation is done using two metrics: the accuracy which measures the overall sentiment classification performance, and RMSE which measures the divergence between predicted and ground truth classes.

**Comparisons of different attribute representation and injection methods** To conduct a fair comparison among the different methods described in Section 2, we compare these methods when applied to our base model using the development set of the datasets. Specifically, we use a smaller version of our base model (with dimensions set to 64) and incorporate the user and product attributes using nine different approaches: (1) **bias-attention**: the bias-based method injected to the attention mechanism, (2-5) the matrix-based method injected to four different locations (**matrix-embedding**, **matrix-encoder**, **matrix-attention**, **matrix-classifier**), and (6-9) the CHIM-based method injected to four different locations (**CHIM-embedding**, **CHIM-encoder**, **CHIM-attention**, **CHIM-classifier**). We then calculate the accuracy of each approach for all datasets.

Results are shown in Figure 3. The figure shows that bias-attention consistently performs poorly compared to other approaches. As expected, matrix-based representations perform the

[2] https://nlp.stanford.edu/projects/glove/

[3] https://drive.google.com/open?id=1PxAkmPLFMnfom46FMMXkHeqIxDbA16oy

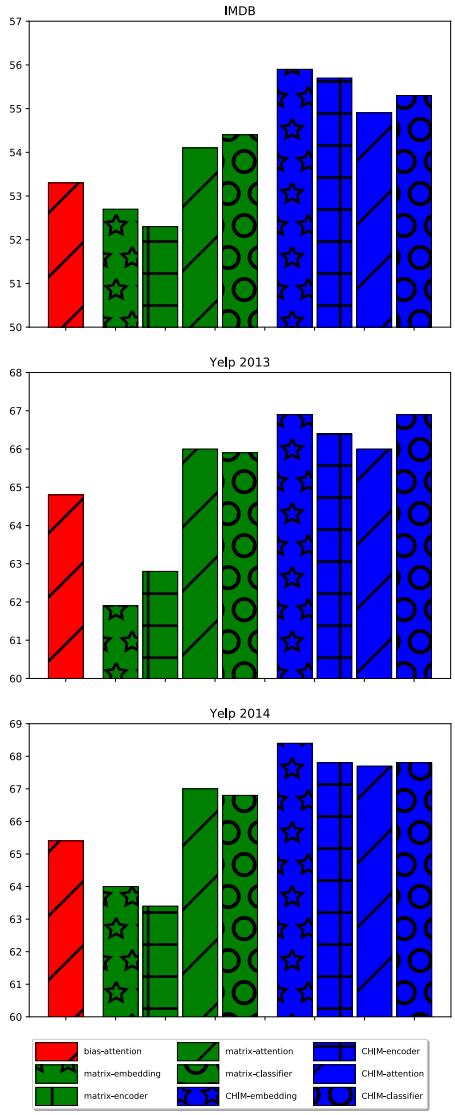


Figure 3: Accuracies (y-axis) of different attribute representation (bias, matrix, CHIM) and injection (emb: embed, enc: encode, att: attend, cls: classify) approaches on the development set of the datasets.

worst when injected to embeddings and encoder, however we can already see improvements over bias-attention when these representations are injected to attention and classifier. This is because the number of parameters used in the the weight matrices of attention and classifier are relatively smaller compared to those of embeddings and encoder, thus they are easier to optimize. The CHIM-based representations perform the best among other approaches, where CHIM-embedding garners the highest accuracy across datasets. Finally, even when using a better representation method, CHIM-attention consistently performs the worst among CHIM-based representations. This shows that attention mechanism is not the optimal location to inject attributes.

**Comparisons with models in the literature** We also compare with models from previous work, listed below:

1. **UPNN** (Tang et al., 2015) uses a CNN classifier as base model and incorporates attributes as user- and product-specific weight parameters in the word embeddings and logistic classifier.

2. **UPDMN** (Dou, 2017) uses an LSTM classifier as base model and incorporates attributes as a separate deep memory network that uses other related documents as memory.

3. **NSC** (Chen et al., 2016) uses a hierarchical LSTM classifier as base model and incorporates attributes using the bias-attention method on both word- and sentence-level LSTMs.

4. **DUPMN** (Long et al., 2018) also uses a hierarchical LSTM as base model and incorporates attributes as two separate deep memory network, one for each attribute.

5. **PMA** (Zhu and Yang, 2017) is similar to NSC but uses external features such as the ranking preference method of a specific user.

6. **HCSC** (Amplayo et al., 2018a) uses a combination of BiLSTM and CNN as base model, incorporates attributes using the bias-attention method, and also considers the existence of cold start entities.

7. **CMA** (Ma et al., 2017) uses a combination of LSTM and hierarchical attention classifier as base model, incorporates attributes using the bias-attention method, and does this separately for user and product.

Notice that most of these models, especially the later ones, use the bias-attention method to represent and inject attributes, but also employ a more complex model architecture to enjoy a boost in performance.

Results are summarized in Table 2. On all three datasets, our best results outperform all previous models based on accuracy and RMSE. Among our four models, CHIM-embedding performs the best in terms of accuracy, with performance increases of 2.4%, 1.3%, and 1.6% on IMDB, Yelp 2013,

| Model | | | IMDB | | Yelp 2013 | | Yelp 2014 | |
|---|---|---|---|---|---|---|---|---|
| Name | Base Model | Injection | Acc | RMSE | Acc | RMSE | Acc | RMSE |
| UPNN | CNN | embedding, classifier | 43.5 | 1.602 | 59.6 | 0.748 | 60.8 | 0.764 |
| UPDMN | LSTM | memory networks | 46.5 | 1.351 | 63.9 | 0.662 | 61.3 | 0.720 |
| NSC | HierLSTM | attention | 53.3 | 1.281 | 65.0 | 0.692 | 66.7 | 0.654 |
| DUPMN | HierLSTM | memory networks | 53.9 | 1.279 | 66.2 | 0.667 | 67.6 | 0.639 |
| PMA | HierLSTM | attention[1] | 54.0 | 1.301 | 65.8 | 0.668 | 67.5 | 0.641 |
| HCSC | BiLSTM+CNN | attention[2] | 54.2 | 1.213 | 65.7 | 0.660 | – | – |
| CMA | LSTM+HierAtt | attention[3] | 54.0 | 1.191 | 66.4 | 0.677 | 67.6 | 0.637 |
| | BiLSTM | embedding | **56.4** | **1.161** | **67.8** | 0.646 | **69.2** | 0.629 |
| CHIM | BiLSTM | encoder | 55.9 | 1.234 | 67.0 | 0.659 | 68.4 | 0.631 |
| (Ours) | BiLSTM | attention | 54.4 | 1.219 | 66.5 | 0.664 | 68.5 | 0.634 |
| | BiLSTM | classifier | 55.5 | 1.219 | 67.5 | **0.641** | 68.9 | **0.622** |

Table 2: Sentiment classification results of competing models based on accuracy and RMSE metrics on the three datasets. Underlined values correspond to the best values for each block. **Boldfaced** values correspond to the best values across the board. [1]uses additional external features, [2]uses a method that considers cold-start entities, [3]uses separate bias-attention for user and product.

and Yelp 2014, respectively. CHIM-classifier performs the best in terms of RMSE, outperforming all other models on both Yelp 2013 and 2014 datasets. Among our models, CHIM-attention mechanism performs the worst, which shows similar results to our previous experiment (see Figure 3). We emphasize that our models use a simple BiLSTM as base model, and extensions to the base model (e.g., using multiple hierarchical LSTMs as in Wu et al. 2018), as well as to other aspects (e.g., consideration of cold-start entities as in Amplayo et al. 2018a), are orthogonal to our proposed attribute representation and injection method. Thus, we expect a further increase in performance when these extensions are done.

### 3.3 Attribute Transfer

In this section, we investigate whether it is possible to transfer the attribute encodings, learned from the sentiment classification model, to other tasks: product category classification and review headline generation. The experimental setup is as follows. First, we train a sentiment classification model using an attribute representation and injection method of choice to learn the attribute encodings. Then, we use these fixed encodings as input to the task-specific model.

**Dataset** We collected a new dataset from Amazon[4], which includes the product category and the review headline, aside from the review text, the sentiment score, and the user and product attributes. Following Tang et al. (2015), we ensured that both users and products are twenty-core, split them into train, dev, and test sets with an 8:1:1 ratio, and tokenized and sentence-split the text using Stanford CoreNLP (Manning et al., 2014). The final dataset contains 77,028 data points, with 1,728 users and 1,890 products. This is used as the sentiment classification dataset.

To create the task-specific datasets, we split the dataset again such that no users and no products are seen in at least two different splits. That is, if user $u$ is found in the train set, then it should not be found in the dev and the test sets. We remove the user-product pairs that do not satistfy this condition. We then append the corresponding product category and review headline for each user-product pair. The final split contains 46,151 training, 711 development, and 840 test instances. It also contains two product categories: Music and Video DVD. The review headline is tokenized using SentencePiece[5] with 10k vocabulary. The datasets are released here for reproducibility: https://github.com/rktamplayo/CHIM.

**Evaluation** In this experiment, we compare five different attribute representation and injection methods: (1) the bias-attention method, and (2-5) the CHIM-based representation method in-

[4]https://s3.amazonaws.com/amazon-reviews-pds/tsv/amazon_reviews_multilingual_US_v1_00.tsv.gz

[5]https://github.com/google/sentencepiece

| Method | Accuracy | Perplexity |
|---|---|---|
| Majority | 60.12 | – |
| Random | 60.67 ± 0.27 | 43.53 |
| Bias - Attention | 58.74 ± 0.49 | 44.00 |
| CHIM - Embedding | 62.26 ± 0.22 | 42.71 |
| CHIM - Encoder | 64.62 ± 0.34 | 42.65 |
| CHIM - Attention | 60.95 ± 0.15 | 42.78 |
| CHIM - Classifier | 61.83 ± 0.43 | 42.69 |

Table 3: Accuracy (higher is better) and perplexity (lower is better) of competing models on the Amazon dataset for the transfer tasks on product category classification and review headline generation, respectively. Accuracy intervals are calculated by running the model 10 times. Performance worse than the random and majority baselines are colored red.

jected to all four different locations in the model. We use the attribute encodings, which are learned from pre-training on the sentiment classification dataset, as input to the transfer tasks, in which they are fixed and not updated during training. As a baseline, we also show results when using encodings of randomly set weights. Moreover, we additionally show the majority class as additional baseline for product category classification.

For the product category classification task, we use a logistic classifier as the classification model and accuracy as the evaluation metric. For the review headline generation task, we use an LSTM decoder as the generation model and perplexity as the evaluation metric.

**Results** For the product category classification task, the results are reported in Table 3. The table shows that representations learned from CHIM-based methods perform better than the random baseline. The best model, CHIM-encoder, achieves an increase of at least 3 points in accuracy compared to the baseline. This means that, interestingly, CHIM-based attribute representations have also learned information about the category of the product. In contrast, representations learned from the bias-attention method are not able to transfer well on this task, leading to worse results compared to the random and majority baseline. Moreover, CHIM-attention performs the worst among CHIM-based models, which further shows the ineffectiveness of injecting attributes to the attention mechanism.

Results for the review headline generation task are also shown in Table 3. The table shows less promising results, where the best model, CHIM-encoder, achieves a decrease of 0.88 points in perplexity from the random encodings. Although this still means that some information has been transferred, one may argue that the gain is too small to be considered significant. However, it has been well perceived, that using only the user and product attributes to generate text is unreasonable, since we expect the model to generate coherent texts using only two vectors. This impossibility is also reported by Dong et al. (2017) where they also used sentiment information, and Ni and McAuley (2018) where they additionally used learned aspects and a short version of the text to be able to generate well-formed texts. Nevertheless, the results in this experiment agree to the results above regarding injecting attributes to the attention mechanism; bias-attention performs worse than the random baseline, and CHIM-attention performs the worst among CHIM-based models.

### 3.4 Where should attributes be injected?

All our experiments unanimously show that (a) the bias-based attribute representation method is not the most optimal method, and (b) injecting attributes in the attention mechanism results to the worst performance among all locations in the model, regardless of the representation method used. The question "where is the best location to inject attributes?" remains unanswered, since different tasks and settings produce different best models. That is, CHIM-embedding achieves the best accuracy while CHIM-classifier achieves the best RMSE on sentiment classification. Moreover, CHIM-encoder produces the most transferable attribute encoding for both product category classification and review headline generation. The suggestion then is to conduct experiments on all locations and check which one is best for the task at hand.

Finally, we also investigate whether injecting in to more than one location would result to better performance. Specifically, we jointly inject in two different locations at once using CHIM, and do this for all possible pairs of locations. We use the smaller version of our base model and calculate the accuracies of different models using the development set of the Yelp 2013 dataset. Figure 4 shows a heatmap of the accuracies of jointly injected models, as well as singly injected models. Overall, the results are mixed and can be summa-

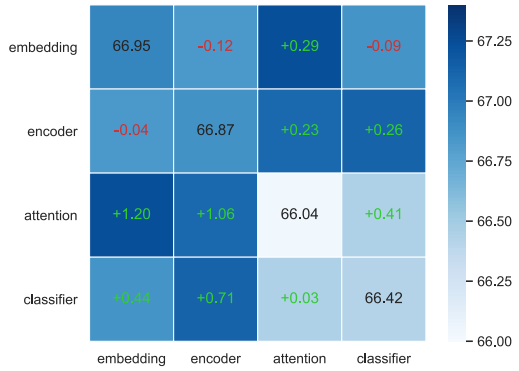


Figure 4: Heatmap of the accuracies of singly and jointly injected CHIM models. Values on each cell represents either the accuracy (for singly injected models) or the difference between the singly and doubly injected models per row.

rized into two statements. Firstly, injecting on the embedding and another location (aside from the attention mechanism) leads to a slight decrease in performance. Secondly and interestingly, injecting on the attention mechanism and another location always leads to the highest increase in performance, where CHIM-attention+embedding performs the best, outperforming CHIM-embedding. This shows that injecting in different locations might capture different information, and we leave this investigation for future work.

## 4 Related Work

### 4.1 Attributes for Sentiment Classification

Aside from user and product information, other attributes have been used for sentiment classification. Location-based (Yang et al., 2017) and time-based (Fukuhara et al., 2007) attributes help contextualize the sentiment geographically and temporally. Latent attributes that are learned from another model have also been employed as additional features, such as latent topics from a topic model (Lin and He, 2009), latent aspects from an aspect extraction model (Jo and Oh, 2011), argumentation features (Wachsmuth et al., 2015), among others. Unfortunately, current benchmark datasets do not include these attributes, thus it is practically impossible to compare and use these attributes in our experiments. Nevertheless, the methods in this paper are not limited to only user and product attributes, but also to these other attributes as well, whenever available.

### 4.2 User/Product Attributes for NLP Tasks

Incorporating user and product attributes to NLP models makes them more personalized and thus user satisfaction can be increased (Baruzzo et al., 2009). Examples of other NLP tasks that use these attributes are text classification (Kim et al., 2019), language modeling (Jaech and Ostendorf, 2018), text generation (Dong et al., 2017; Ni and McAuley, 2018), review summarization (Yang et al., 2018b), machine translation (Michel and Neubig, 2018), and dialogue response generation (Zhang et al., 2017). On these tasks, the usage of the bias-attention method is frequent since it is trivially easy and there have been no attempts to investigate different possible methods for attribute representation and injection. We expect this paper to serve as the first investigatory paper that contradicts to the positive results previous work have seen from the bias-attention method.

## 5 Conclusions

We showed that the current accepted standard for attribute representation and injection, i.e. bias-attention, which incorporates attributes as additional biases in the attention mechanism, is the least effective method. We proposed to represent attributes as chunk-wise importance weight matrices (CHIM) and showed that this representation method significantly outperforms the bias-attention method. Despite using a simple BiLSTM classifier as base model, CHIM significantly outperforms the current state-of-the-art models, even when those models use a more complex base model architecture. Furthermore, we conducted several experiments that conclude that injection to the attention mechanism, no matter which representation method is used, garners the worst performance. This result contradicts previously reported conclusions regarding attribute injection to the attention mechanism. Finally, we show promising results on transferring the attribute representations from sentiment classification, and use them to two different tasks such as product category classification and review headline generation.

## Acknowledgments

We would like to thank the anonymous reviewers for their helpful feedback and suggestions. Reinald Kim Amplayo is grateful to be supported by a Google PhD Fellowship.

## References


Reinald Kim Amplayo, Jihyeok Kim, Sua Sung, and Seung-won Hwang. 2018a. Cold-start aware user and product attention for sentiment classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 2535–2544.

Reinald Kim Amplayo, Kyungjae Lee, Jinyoung Yeo, and Seung-won Hwang. 2018b. Translations as additional contexts for sentence classification. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden.*, pages 3955–3961.

Stefanos Angelidis and Mirella Lapata. 2018. Summarizing opinions: Aspect extraction meets sentiment prediction and they are both weakly supervised. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3675–3686.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Andrea Baruzzo, Antonina Dattolo, Nirmala Pudota, and Carlo Tasso. 2009. A general framework for personalized text classification and annotation. In *Proceedings of the Workshop on Adaptation and Personalization for Web 2.0, AP WEB 2.0@UMAP, Trento, Italy, June 22, 2009.*

Huimin Chen, Maosong Sun, Cunchao Tu, Yankai Lin, and Zhiyuan Liu. 2016. Neural sentiment classification with user and product attention. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1650–1659.

Li Dong, Shaohan Huang, Furu Wei, Mirella Lapata, Ming Zhou, and Ke Xu. 2017. Learning to generate product reviews from attributes. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 623–632, Valencia, Spain. Association for Computational Linguistics.

Zi-Yi Dou. 2017. Capturing user and product information for document level sentiment analysis with deep memory network. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 521–526. Association for Computational Linguistics.

Jessica Ficler and Yoav Goldberg. 2017. Controlling linguistic style aspects in neural language generation. *CoRR*, abs/1707.02633.

Tomohiro Fukuhara, Hiroshi Nakagawa, and Toyoaki Nishida. 2007. Understanding sentiment of people from news articles: Temporal sentiment analysis of social events. In *Proceedings of the First International Conference on Weblogs and Social Media, ICWSM 2007, Boulder, Colorado, USA, March 26-28, 2007*.

Wenliang Gao, Naoki Yoshinaga, Nobuhiro Kaji, and Masaru Kitsuregawa. 2013. Modeling user leniency and product popularity for sentiment classification. In *Sixth International Joint Conference on Natural Language Processing, IJCNLP 2013, Nagoya, Japan, October 14-18, 2013*, pages 1107–1111.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Aaron Jaech and Mari Ostendorf. 2018. Low-rank RNN adaptation for context-aware language modeling. *TACL*, 6:497–510.

Yohan Jo and Alice H. Oh. 2011. Aspect and sentiment unification model for online review analysis. In *Proceedings of the Forth International Conference on Web Search and Web Data Mining, WSDM 2011, Hong Kong, China, February 9-12, 2011*, pages 815–824.

Jihyeok Kim, Reinald Kim Amplayo, Kyungjae Lee, Sua Sung, Minji Seo, and Seung-won Hwang. 2019. Categorical metadata representation for customized text classification. *TACL*, 7:201–215.

Soo-Min Kim and Eduard H. Hovy. 2004. Determining the sentiment of opinions. In *COLING 2004, 20th International Conference on Computational Linguistics, Proceedings of the Conference, 23-27 August 2004, Geneva, Switzerland*.

Fangtao Li, Nathan Nan Liu, Hongwei Jin, Kai Zhao, Qiang Yang, and Xiaoyan Zhu. 2011. Incorporating reviewer and product information for review rating prediction. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 1820–1825.

Chenghua Lin and Yulan He. 2009. Joint sentiment/topic model for sentiment analysis. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, China, November 2-6, 2009*, pages 375–384.

Bing Liu. 2012. *Sentiment Analysis and Opinion Mining*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Yunfei Long, Mingyu Ma, Qin Lu, Rong Xiang, and Chu-Ren Huang. 2018. Dual memory network model for biased product review classification. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 140–148. Association for Computational Linguistics.

Dehong Ma, Sujian Li, Xiaodong Zhang, Houfeng Wang, and Xu Sun. 2017. Cascading multiway attentions for document-level sentiment classification. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 634–643. Asian Federation of Natural Language Processing.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, System Demonstrations*, pages 55–60.

Paul Michel and Graham Neubig. 2018. Extreme adaptation for personalized neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 312–318.

Andrew Y. Ng. 2004. Feature selection, l1 vs. l2 regularization, and rotational invariance. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 78–, New York, NY, USA. ACM.

Jianmo Ni and Julian McAuley. 2018. Personalized review generation by expanding phrases and attending on aspect-aware representations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 706–711.

Bo Pang and Lillian Lee. 2007. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135.

Dae Hoon Park, Hyun Duk Kim, ChengXiang Zhai, and Lifan Guo. 2015. Retrieval of relevant opinion sentences for new products. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015*, pages 393–402.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron C. Courville. 2019. Ordered neurons: Integrating tree structures into recurrent neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Chenhao Tan, Lillian Lee, Jie Tang, Long Jiang, Ming Zhou, and Ping Li. 2011. User-level sentiment analysis incorporating social networks. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 21-24, 2011*, pages 1397–1405.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Learning semantic representations of users and products for document level sentiment classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1014–1023.

Henning Wachsmuth, Johannes Kiesel, and Benno Stein. 2015. Sentiment flow - A general model of web review argumentation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 601–611.

Zhen Wu, Xin-Yu Dai, Cunyan Yin, Shujian Huang, and Jiajun Chen. 2018. Improving review representations with user attention and product attention for sentiment classification. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5989–5996.

Min Yang, Jincheng Mei, Heng Ji, Wei Zhao, Zhou Zhao, and Xiaojun Chen. 2017. Identifying and tracking sentiments and topics from social media texts during natural disasters. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 527–533.

Min Yang, Qiang Qu, Jia Zhu, Ying Shen, and Zhou Zhao. 2018a. Cross-domain aspect/sentiment-aware abstractive review summarization. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018,*

*Torino, Italy, October 22-26, 2018*, pages 1531–1534.

Min Yang, Wenting Tu, Qiang Qu, Zhou Zhao, Xiaojun Chen, and Jia Zhu. 2018b. Personalized response generation by dual-learning based domain adaptation. *Neural Networks*, 103:72–82.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. 2016. Hierarchical attention networks for document classification. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 1480–1489.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.

Weinan Zhang, Ting Liu, Yifa Wang, and Qingfu Zhu. 2017. Neural personalized response generation as domain adaptation. *CoRR*, abs/1701.02073.

Pengcheng Zhu and Yujiu Yang. 2017. Parallel multi-feature attention on neural sentiment classification. In *Proceedings of the Eighth International Symposium on Information and Communication Technology, Nha Trang City, Viet Nam, December 7-8, 2017*, pages 181–188.

# A Knowledge Regularized Hierarchical Approach for Emotion Cause Analysis

**Chuang Fan**[1,5], **Hongyu Yan**[1,5], **Jiachen Du**[1,5], **Lin Gui**[2], **Lidong Bing**[3]
**Min Yang**[4], **Ruifeng Xu**[1,5,6]*, **Ruibin Mao**[7]
[1]Harbin Institute of Technology (Shenzhen), China [2]University of Warwick, UK
[3]R&D Center Singapore, Machine Intelligence Technology, Alibaba DAMO Academy
[4]Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences
[5]Joint Lab of Harbin Institute of Technology and RICOH
[6]Peng Cheng Laboratory, China [7]Shenzhen Securities Information Co.,Ltd, China

## Abstract

Emotion cause analysis, which aims to identify the reasons behind emotions, is a key topic in sentiment analysis. A variety of neural network models have been proposed recently, however, these previous models mostly focus on the learning architecture with local textual information, ignoring the discourse and prior knowledge, which play crucial roles in human text comprehension. In this paper, we propose a new method to extract emotion cause with a hierarchical neural model and knowledge-based regularizations, which aims to incorporate discourse context information and restrain the parameters by sentiment lexicon and common knowledge. The experimental results demonstrate that our proposed method achieves the state-of-the-art performance on two public datasets in different languages (Chinese and English), outperforming a number of competitive baselines by at least 2.08% in F-measure.

## 1 Introduction

Sentiment analysis has gained increasing popularity in recent years due to many useful applications (Pang and Lee, 2007). The goal of sentiment analysis is to classify the sentiment polarity of a given text as positive, negative, neutral, or more fine-grained classes (Kim, 2014; Li et al., 2015; Yang et al., 2016; Zhou et al., 2016; Chen et al., 2017; Qian et al., 2017; Li et al., 2018b; Yu et al., 2018). Most of these researches have assumed that emotion expressions are already observed and try to identify the emotion categories from text. However, in practice, such as in product comments or political reviews, we may care more about the reason why the customers or critics hold the emotion rather than a simple category label. Because they can improve the quality of the products or services according to the emotion cause provided by users. Emotion cause analysis (ECA) aims to identify the reasons behind a certain emotion expression in an event text, for example:

**Ex.1** *When the children saw the gifts I prepared carefully,* $(c_{-2})$*|they cheered happily and hugged me.* $(c_{-1})$*| I was full of happiness.* $(c_0)$

Here, **Ex.1** shows a document with three clauses marked as $(c_{-2})$, $(c_{-1})$, and $(c_0)$. The goal of ECA is to determine which clause contains emotion cause (e.g., $(c_{-1})$) for an emotion word (e.g., *happiness* in $(c_0)$).

Previous approaches for emotion cause analysis mostly depend on rule-based methods (Lee et al., 2010; Chen et al., 2010) and machine learning algorithms (Ghazi et al., 2015; Gui et al., 2016; Xu et al., 2019). Most of them rely heavily on complicated linguistic rules or feature engineering, which is time-consuming and labor-intensive. Recent studies have focused on solving the task using neural models (Gui et al., 2017; Li et al., 2018a; Chen et al., 2018; Li et al., 2019) with well designed attention mechanism based on local text. Despite the effectiveness of neural models, there are some defects in previous studies. First, they usually consider each clause individually, i.e., ignoring the discourse context information that can impact the semantic expression among different clauses of a document. Second, prior knowledge such as sentiment lexicon and relative position information that can provide crucial emotion cause cues has not been fully exploited in neural models.

To alleviate these limitations, we propose a regularized hierarchical neural network (RHNN) for emotion cause analysis, which combines the discourse context information and knowledge-based regularizations. Our model investigates the following intuitions. Firstly, documents exhibit discourse structure which may carry valuable information about the emotion cause cues. We em-

* Corresponding Author: xuruifeng@hit.edu.cn



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5614–5624, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

ploy a hierarchical learning structure to capture the mutual impacts of semantic expression among the discourse context, to help produce better clause representation. Secondly, in emotion events, emotion causes usually express a certain sentiment polarity by some sentiment words. For example, in the emotion cause of **Ex.1**, the sentiment words *cheered* and *happily* express a positive polarity, also play a crucial role to provoke the emotion *happiness*. Therefore, capturing these sentiment words can enhance the causal connection between learned features and predictions. We approach this issue by designing a regularizer that incorporates linguistic knowledge (e.g., sentiment lexicon) to enlarge the margin of attention weights of sentiment words and non-sentiment words. Besides, it is often the case that humans usually write important points in different sections. For emotion events, emotion causes generally occur on positions very close to the emotion word and occur frequently. **Ex.1** shows an anecdotal example illustrating this behavior that the emotion cause clause $c_{-1}$ adjoins the emotion word *happiness*. To benefit from this phenomenon, we introduce a regularizer biased by relative position information to supervise the representation learning of text and further to revise the **predictive position distribution of emotion causes relative to the emotion word** (in brief, predictive distribution).

To sum up, our contribution includes:

- We propose a novel discourse-aware learning structure with knowledge-based regularizations for emotion cause analysis.
- We empirically evaluate the proposed model on two public datasets in different languages (Chinese and English) and show statistically significant improvements compared to the state-of-the-art methods.
- To make the mechanism of our model clear, we also compare the performance of different combinations by ablation experiments. Extensive analysis on both datasets confirms the feasibility of incorporating discourse information and restraining the parameters by sentiment lexicon and common knowledge.

## 2 Our Framework

In this section, we first give the task definition. Then, our proposed regularized hierarchical neural network (RHNN), as shown in Fig 1, will be described. The two auxiliary regularizers of RHNN will be introduced in the next section.

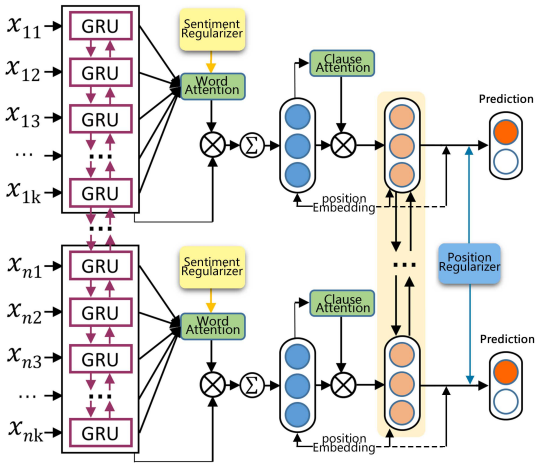


Figure 1: The architecture of our model.

### 2.1 Task Definition

The formal definition of emotion cause analysis is given in (Gui et al., 2016). Formally, for a document $d = \{c_1, c_2, \ldots, c_n\}$ consisting of $n$ clauses, it contains an emotion word $e$ and at least an emotion cause clause corresponding to this emotion word. Each clause $c_i = \{w_{i1}, w_{i2}, \ldots, w_{ik}\}$ consists of $k$ words and is labeled with emotion cause-oriented labels $\in \{0, 1\}$. We regard ECA as a binary classification task and aim to identify which clause contains emotion cause.

### 2.2 Hierarchical Attention Network

Documents exhibit discourse structure which can serve as useful information for clause representation generation. One simple but effective approach is to adopt a hierarchical attention network to simulate this structure. Our hierarchical attention network consists of several parts: a word encoder, a word attention layer, a clause attention layer and a clause encoder. The details of each component will be described in the following paragraphs.

**Word Encoder** Gated Recurrent Unit (GRU) has been widely adopted for text processing (Cho et al., 2014). In this work, we first map each word into a low dimensional embedding space by Word2Vec (Mikolov et al., 2013) and then feed the whole document into a GRU-based word encoder to extract word sequence features. To summarize information from both directions, we use bidirec-

tional GRU to exploit two parallel passes:

$$\overrightarrow{h_{it}} = \overrightarrow{\mathrm{GRU}_w}(x_{it}), t \in [1, k] \quad (1)$$

$$\overleftarrow{h_{it}} = \overleftarrow{\mathrm{GRU}_w}(x_{it}), t \in [k, 1] \quad (2)$$

where $x_{it}$ is embedding vector for the word $w_{it}$ in clause $c_i$ at time step $t$ and $k$ is the length of clause $c_i$. Then we concatenate hidden states of the two directions $h_{it} = [\overrightarrow{h_{it}}, \overleftarrow{h_{it}}]$ as the representation of each word.

**Word Attention** We introduce an attention mechanism to extract such words that are important to the meaning of the clause and aggregate the representation of these informative words to construct the clause vector. Specifically,

$$g_{it} = w_m^T(h_{it} \oplus e^E) \quad (3)$$

$$\alpha_{it} = \frac{\exp(g_{it})}{\sum_{t'} \exp(g_{it'})} \quad (4)$$

$$o_{c_i} = \sum_t \alpha_{it} h_{it} \quad (5)$$

where $w_m$ is the parameter for computing attention signals and $\oplus$ is concatenate operation. The embedding of emotion word $e$ is denoted by $e^E$. $\alpha_{it}$ is the emotion-specific attention signal showing the importance of word $w_{it}$. $o_{c_i}$ is the weighted sum of word representation based on weights.

**Clause Attention** Intuitively, different clauses of a document are different informative and should be labeled with different importance. Targeting this problem, we design a clause attention mechanism to indicate the importance of each clause. See it differently, the attention signals can be regarded as some "prior" information to bias the clause encoder toward some content that is more important to extracting the emotion cause.

As for details, we adopt a one-layer MLP to get the attention signals. Apparently, position information plays an important role in capturing the relative distance of the clause to emotion word. Thus, we concatenate the clause vector $o_{c_i}$ and its position embedding as the feature to obtain the attention signal, this yields:

$$\alpha_i = \mathrm{sigmoid}(w_v(o_{c_i} \oplus l_i)) \quad (6)$$

$$o'_{c_i} = o_{c_i} \cdot \alpha_i \quad (7)$$

where $w_v$ is a parameter vector, $l_i$ is the randomly initialized position embedding and keeps unchanged in the training stage, $\alpha_i$ is the weight of clause $c_i$. Then the clauses with different importance (e.g., $o'_{c_i}$) are fed into a clause encoder.

**Clause Encoder** Just as the meaning of a word is determined by its context, the semantic expression of a clause is usually impacted by its discourse context. Based on this observation, we introduce a clause encoder to model the latent semantic relations among different clauses. Analogously, we also append the relative position information to enhance the relations between the clause and its position. Formally,

$$\overrightarrow{h_i} = \overrightarrow{\mathrm{GRU}_c}(o'_{c_i} \oplus l_i), i \in [1, n] \quad (8)$$

$$\overleftarrow{h_i} = \overleftarrow{\mathrm{GRU}_c}(o'_{c_i} \oplus l_i), i \in [n, 1] \quad (9)$$

where $n$ is the number of clauses in a document. Also, the two directional hidden state $\overrightarrow{h_i}$ and $\overleftarrow{h_i}$ are concatenated as the final emotion-specific representation $o_i = [\overrightarrow{h_i}, \overleftarrow{h_i}]$.

### 2.3 Model Training

The emotion-specific representation $o_i$ with its position embedding $l_i$ as the final feature for emotion cause prediction and the model is trained by minimizing the cross entropy:

$$y_i = \mathrm{softmax}(W_m(o_i \oplus l_i)) \quad (10)$$

$$\mathcal{L}_{ce} = -\sum_i \widehat{y}_i \log y_i \quad (11)$$

where $W_m$ is a parameter matrix, $\widehat{y}_i$ and $y_i$ are target class distribution and predictive class distribution respectively.

## 3 Regularizers Based on Sentiment Lexicon and Relative Position

One crucial emotion cause cue is sentiment words since emotion causes usually express a certain sentiment polarity by sentiment words. However, there is no effective mechanism to guarantee that the above module indeed attends the words with sentiment polarity. Beyond this, a straightforward solution to inject position information is to directly incorporate relative word position embedding, which is a weak representation of the relative distance between the emotion word and each clause. To ease these problems, we introduce two auxiliary regularizers:

- **Sentiment Regularizer (SR.):** If a clause contains several words which exist in a sentiment lexicon, the calculated average weights of these words should be properly larger than other words. We approach this issue with an auxiliary hinge loss function.

- **Position Regularizer (PR.):** Relative position is a critical emotion cause indicator: in general the closer a clause is to the emotion word, the higher emotion cause probability it should be assigned. We approach this issue by introducing a proxy distribution and an auxiliary cross entropy function.

Formally, we disassemble the joint loss of emotion cause detection into an original cross entropy loss, a sentiment regularization loss, and a position regularization loss. The new training objective is revised as:

$$\mathcal{J}(\theta) = \mathcal{L}_{ce} + \lambda_1 * \mathcal{L}_{sr} + \lambda_2 * \mathcal{L}_{pr} \quad (12)$$

where $\mathcal{L}_{ce}$ is the original cross entropy loss (§2.3). $\lambda_1$ and $\lambda_2$ are hyper-parameters. $\mathcal{L}_{sr}$ and $\mathcal{L}_{pr}$ are two auxiliary regularization losses (§3.1 and §3.2). $\theta$ is the parameter set.

### 3.1 Sentiment Regularizer

The weak causal connection between learned features and predictions is a major issue in emotion cause analysis. Even though sentiment words are key to clause representation generation, most existing models do not focus on sentiment words or place less emphasis on them when producing clause representation. In other words, attention is distracted by irrelevant words with less sentiment polarity. To address this issue and sufficiently benefit from linguistic resources, we explicitly encourage the larger margin of attention weights between sentiment words and non-sentiment words using a sentiment lexicon. For sentiment words, the average attention weight is calculated by:

$$Avg_s = \frac{1}{l_s} \sum_{t=1}^{k} S_{w_{it}} \quad (13)$$

$$S_{w_{it}} = \begin{cases} \alpha_{it} & \text{if} \quad w_{it} \in c_i \cap \mathcal{S} \\ 0 & \text{otherwise} \end{cases} \quad (14)$$

where $w_{it}$ is the $t$-th word of clause $c_i$, $k$ is the length of $c_i$ and $l_s$ is the number of sentiment words in $c_i$. $\alpha_{it}$ is the calculated attention weight in §2.2 and $\mathcal{S}$ is a sentiment lexicon. Correspondingly, for non-sentiment words:

$$Avg_{ns} = \frac{1}{l_{ns}} \sum_{t=1}^{k} NS_{w_{it}} \quad (15)$$

$$NS_{w_{it}} = \begin{cases} \alpha_{it} & \text{if} \quad w_{it} \in c_i - \mathcal{S} \\ 0 & \text{otherwise} \end{cases} \quad (16)$$

Our training objective is to lead the model to pay more attention to sentiment words. Thus, the regularization term is formally expressed as:

$$\mathcal{L}_{sr} = \sum_i \max(0, m - (Avg_s - Avg_{ns})) \quad (17)$$

where $m$ is a hyper-parameter for margin.

### 3.2 Position Regularizer

Empirically, emotion causes usually occur at the positions which are very close to the emotion word. However, another main issue is that the predictive distribution may locates the clauses that are distant from and irrelevant to the emotion word. The goal of this regularizer is to narrow the difference between the predictive distribution and the **true position distribution of emotion causes relative to the emotion word** (in brief, true distribution). Obviously we can not obtain the true distribution. Hence, we assume that it should satisfy the following conditions: (1) It should be a normed function within $[0, 1]$; (2) It should be a symmetric function of a certain value. Based on these conditions, we employ a function defined as follows:

$$q_i = \begin{cases} 1 - \frac{|r_i|}{n} & \text{if} - b \leq r_i \leq b \\ 0 & \text{otherwise} \end{cases} \quad (18)$$

where $r_i$ is the relative distance of clause $c_i$ to emotion word, $n$ is the number of clauses in a document, and $b$ is the left and right boundary which limits the scope of emotion cause. Then we apply $q = (q_1, q_2, \ldots, q_n)$ as the proxy distribution to simulate the true distribution.

Simultaneously, from section 2.3, we get the predictive class distribution $y_i$ of clause $c_i$. Then, the probability for the emotion cause at position $i$ can be calculated as:

$$p_i = y_i(label = 1) \quad (19)$$

Similarity, we can obtain the predictive distribution of emotion causes relative to the emotion word by:

$$p = (p_1, p_2, \ldots, p_n) \quad (20)$$

The goal is to enforce the model to constrain the difference between the $p$ and $q$, thus, we use cross entropy to measure the difference:

$$\mathcal{L}_{pr} = -\sum_i q_i \log(p_i) \quad (21)$$

Note that the two introduced regularizers work like $L_1$ and $L_2$ terms, which **do not introduce**

| Item | Number | Item | Number |
|---|---|---|---|
| Chinese Dataset (Chi.) | | | |
| Documents | 2105 | Cause_1 | 2046 |
| Clauses | 11799 | Cause_2 | 56 |
| Causes | 2167 | Cause_3 | 3 |
| English Dataset (Eng.) | | | |
| Documents | 2156 | Cause_1 | 1949 |
| Clauses | 16259 | Cause_2 | 164 |
| Causes | 2421 | Cause_3 | 32 |

Table 1: Details of the two datasets. Cause_1, Cause_2 and Cause_3 represent the documents with 1, 2 and 3 cause clauses, respectively.

any new parameters and only influence the training of the standard model parameters. The hyper-parameters $\lambda_1$ and $\lambda_2$ guide the model to achieve the best trade-off among three types of losses.

## 4 Datasets and Implementation Details

### 4.1 Datasets

We select two public datasets from different languages to evaluate the proposed model: Chinese Dataset (Gui et al., 2016) collected from SINA city news[1] and English Dataset (Gao et al., 2017) collected from English novels. Each document of both datasets has only one emotion word and one or more emotion causes. It has been ensured that the emotion and the causes are relevant. The documents are segmented into several clauses manually for emotion cause analysis. The details about the two datasets are summarized in Table 1.

### 4.2 Implementation Details

For the Chinese dataset, there is no training/test split, we randomly divide the documents into a training/development/test set in a ratio of 8:1:1 and partition the clauses by Jieba[2]. For the English dataset, we randomly select 10% from the original training set as the development set and lowercase, lemmatize all the tokens by NLTK[3]. We evaluate our method 25 times with different splits and then perform one sample $t$-test on the experimental results by following (Gui et al., 2017). The precision ($P$), recall ($R$) and F-measure ($F$) are employed to measure the performance in this task.

The sentiment lexicon adopted for the Chinese dataset consists of two parts. The first part is selected from HowNet (Dong et al., 2006) sentiment analysis lexicon set[4] and the second part comes from NTUSD (Ku et al., 2006). The combination of the two parts serves as the Chinese sentiment lexicon of this research. The English sentiment lexicon comes from MPQA (Wilson et al., 2005) and we only select the words with high sentiment polarity, because they are less sensitive to contextual information and usually express consistence sentiment polarities from their prior polarity. For both sentiment lexica, we filter out the words that are not in the datasets. Ultimately, 2022 and 1348 sentiment words are selected for the Chinese and English dataset respectively.

Online learning is performed with the Adam optimizer (Kingma and Ba, 2015) and initial learning rate 0.001 is adopted. The number of layers in Bi-GRU is set to 2 and dropout rate 0.5 is used to avoid overfitting. The word vectors are pre-trained by word2vec (Mikolov et al., 2013) and keep unchanged during training stage. We perform grid search over the hyper-parameters $m$ ($\{0.10, 0.15, 0.20\}$), the boundary $b$ ($\{2, 3, 4\}$), the dimensionality of the Bi-GRU ($\{64, 128\}$), $\lambda_1$ and $\lambda_2$ (both $\{0.25, 0.5, 0.75\}$). For each corpus, the highest F-measure combination of these hyper-parameters is selected using development set.

## 5 Experiments

In this section, we will compare our RHNN model with the following groups of methods:

- Rule-based and commonsense-based methods: Rule-based method (**RB**) is a traditional rule-based method proposed by Lee et al. (2010). Commonsense-based methods (**CB**) is a knowledge-based method proposed by Russo et al. (2011). It uses Chinese Emotion Cognition Lexicon (Xu et al., 2013) as commonsense knowledge.

- Machine learning method: **SVM** is a SVM classifier trained on unigrams, bigrams and trigrams features (Li and Xu, 2014). **Word2vec** is a SVM classifier trained on word representations pre-trained by Word2vec (Mikolov et al., 2013). **Multi-kernel** represents a document by a syntactic structure and utilizes a modified convolution kernel method to determine which clause contains the emotion cause (Gui et al., 2016).

[1] http://news.sina.com.cn/society/
[2] https://github.com/fxsjy/jieba
[3] http://nltk.org
[4] http://www.keenage.com/html/c_index.html

| Method | P | R | F |
|---|---|---|---|
| RB* | 0.6747 | 0.4287 | 0.5243 |
| CB* | 0.2672 | 0.7130 | 0.3887 |
| SVM* | 0.4200 | 0.4375 | 0.4285 |
| Word2vec* | 0.4301 | 0.4233 | 0.4136 |
| Multi-kernel* | 0.6588 | 0.6972 | 0.6752 |
| LambdaMART* | 0.7720 | 0.7499 | 0.7608 |
| CNN* | 0.6472 | 0.5493 | 0.5915 |
| ConvMS-Memnet* | 0.7076 | 0.6838 | 0.6955 |
| CANN | 0.7721 | 0.6891 | 0.7266 |
| HCS | 0.7388 | 0.7154 | 0.7269 |
| MANN | 0.7843 | 0.7587 | 0.7706 |
| **RHNN** | **0.8112** | **0.7725** | **0.7914** |

Table 2: Experimental results on the Chinese dataset. Superscript $*$ indicates the results are reported in (Gui et al., 2017) and the rest are reprinted from the corresponding publications ($p$ <0.001).

| Method | P | R | F |
|---|---|---|---|
| Word2vec | 0.1651 | **0.8673** | 0.2774 |
| SVM | 0.2757 | 0.6416 | 0.3856 |
| CNN | 0.7218 | 0.2628 | 0.3390 |
| ConvMS-Memnet | 0.4605 | 0.4177 | 0.4381 |
| MANN | **0.7933** | 0.4081 | 0.5328 |
| **RHNN** | 0.6901 | 0.5267 | **0.5975** |

Table 3: Experimental results on the English dataset, we follow the results that are implemented in (Li et al., 2019), the only available results on this dataset ($p$ <0.001).

**LambdaMART** extracts emotion causes using learning to rank methods which based on the emotion-independent and emotion-dependent features (Xu et al., 2019).

- Deep learning method: **CNN** is a convolutional neural network for sentence classification (Kim, 2014). **ConvMS-Memnet** considers emotion cause analysis as a reading comprehension task and designs a multiple-slot deep memory network to model context information (Gui et al., 2017). **CANN** uses a co-attention neural network to identify emotion causes (Li et al., 2018a). **HCS** is proposed by Yu et al. (2019) using a multiple-level hierarchical network to detect the emotion causes. **MANN** is the current state-of-the-art method employing a multi-attention-based model for emotion cause extraction (Li et al., 2019). **RHNN** is our proposed model.

### 5.1 Main Results

The experimental results on both datasets are shown in Table 2 and Table 3, respectively. RB yields high precision but with low recall. CB has an opposite scenario from RB. A possible reason is that these linguistic-based methods depend on some cue words to identify the emotion cause, different rules or common sense may contain different cue words.

For the machine learning methods, SVM and Word2vec have similar performance on the Chinese dataset, but SVM outperforms Word2vec on the English dataset. The main reason is that the polysemantic phenomenon is more obvious in English expressions. Multi-kernel has better performance by capturing context information through a syntactic tree. LambdaMART, which is based on ranking strategy and global emotion features, performs best among feature-based methods. However, both Multi-kernel and LambdaMART rely on expensive human-based features and lack of expandability on different dataset.

Compared with CNN, ConvMS-Memnet models the context of each word and obtains better performance on both datasets. The co-attention based CANN captures the mutual relations between the emotion clause and each candidate clause, which has a comparable result with hierarchical-based HCS. MANN considers the interaction between the emotion clause and candidate clauses by designing a multi-attention mechanism and achieves the best performance among baselines.

The proposed RHNN model further improves the performance on both datasets as shown in the tables. The improvement is significant with $p$-value less than 0.001 in one sample $t$-test. Specifically, RHNN manages to boost the performance by 3.06% in F-measure compared to LambdaMART, which exhibits that by restraining the parameters with knowledge-based regularizations, RHNN is better to identify the emotion cause cues than feature engineering. RHNN also outperforms the current best-performing method MANN by 2.08% on the Chinese dataset and 6.47% on the English dataset in F-measure respectively. Furthermore, for the English dataset, our proposed model has balance performance in precision and recall. The reason for this phenomenon is that RHNN can capture more emotion cue (e.g., sen-

| Method | Applying with | | | Chi. | Eng. |
|---|---|---|---|---|---|
| | H. | SR. | PR. | F | F |
| Base | × | × | × | 0.7152 | 0.5199 |
| H | ✓ | × | × | 0.7553 | 0.5668 |
| SR | × | ✓ | × | 0.7570 | 0.5765 |
| PR | × | × | ✓ | 0.7483 | 0.5683 |
| HSR | ✓ | ✓ | × | 0.7860 | **0.6154** |
| HPR | ✓ | × | ✓ | 0.7659 | 0.5777 |
| SPR | × | ✓ | ✓ | 0.7600 | 0.5878 |
| RHNN | ✓ | ✓ | ✓ | **0.7914** | 0.5975 |

Table 4: Effect of different components, i.e., hierarchical structure (H.), sentiment regularizer (SR.) and position regularizer (PR.). The leftmost column is the abbreviation for corresponding sub-models (e.g., SPR denotes the sub-model with SP. and PR. except H.).

| Method | Chi. | | Eng. | |
|---|---|---|---|---|
| | Sub. | All. | Sub. | All. |
| SR | 0.7650 | 0.7570 | 0.6713 | 0.5765 |
| HSR | 0.7934 | 0.7860 | **0.7084** | **0.6154** |
| SPR | 0.7741 | 0.7600 | 0.6904 | 0.5878 |
| RHNN | **0.7997** | **0.7914** | 0.6849 | 0.5975 |

Table 5: The F-measure on sub-dataset (Sub.) that only selects the clauses which contain sentiment words, and all-dataset (All.) that experiments on the whole dataset.

timent words) information to optimize the model extracting emotion causes more exactly.

### 5.2 Detailed Analysis

**Ablations of RHNN Model** The proposed RHNN model consists of three components, including hierarchical structure (H.), sentiment regularizer (SR.) and position regularizer (PR.). We conduct ablation experiments to reveal the effect of each component. As illustrated in Table 4, all models with the proposed component consistently improve upon the Base model, verifying the effectiveness of the proposed approach. Compare with H. and PR. model, the SR. improve the performance most. The main reason is that there are 55.24% and 66.49% of emotion causes which contain sentiment words on the Chinese dataset and English dataset respectively, enforcing the model to pay more attention to sentiment words can enhance the causal connection between learned features and predictions.

On the Chinese dataset, the RHNN achieves the best performance with a 7.62% improvement on

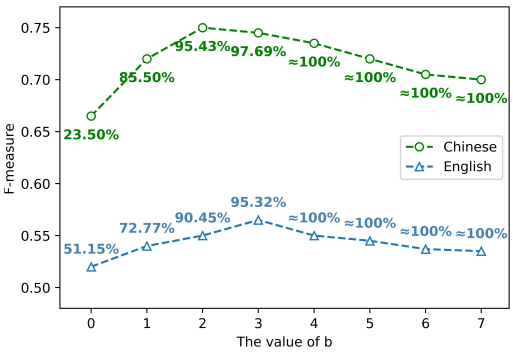


Figure 2: The F-measure of different limited scopes of emotion cause on two datasets.

the $F$ score compared with the baseline. However, on the English dataset, the HSR model performs better than the RHNN model. It may be caused by the overlapping between components. Besides, the performance on the English dataset always lower about 20% in F-measure than that on the Chinese dataset, one possible explanation for this phenomenon is that there are more clause structures in English expressions which is difficult for the model to capture this information without discourse tree.

**Effect of Sentiment Regularizer** To gain more insights into our proposed model, we conduct further experiments to examine the effectiveness of sentiment regularizer on the subset of two datasets. The experiment results in Table 5 show that: 1) RHNN and HSR model achieve the best performance on the subset of two datasets respectively, similar observations can be found regarding on whole dataset; 2) With sentiment regularizer, the performance is boosted on both subsets compared to that on the whole dataset. This is consistent with our intuition because sentiment regularizer contributes much to pick up the words with sentiment polarity and these words are important causal indicators in clauses. Meanwhile, each clause contains sentiment words in the subsets, resulting in a better performance on the subsets. 3) The performance improvement on the English subset is remarkably higher than that on the Chinese subset, one possibility is that there are more explicit emotion terms in English expression than in Chinese.

**Effect of Position Regularizer** From Eq.(18), we see that the value of $b$ limits the scope of emotion cause. In this section, we further investigate the effect of different limited scopes. For simplicity and efficiency, here we only apply the PR

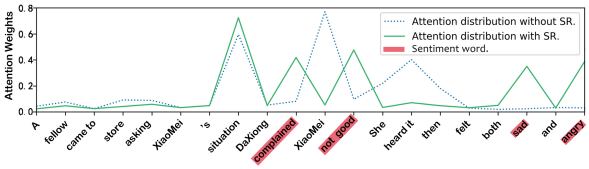


Figure 3: Comparison of attention distribution with or without sentiment regularizer.

model on this experiment. The results are shown in Fig 2, the percentages denote the coverage of emotion cause in datasets. For instance, 72.77% represents that there are 72.77% emotion causes adjoining the emotion word in English dataset. From Fig 2, we can see that the performance trends for the two datasets are similar, and the performance improves with the expanding limited scope of emotion cause. However, when the limited scope of emotion cause is larger than a certain value (2 or 3), the performance decreases. This may be due to the reason that the larger of limited scope, the higher coverage of emotion cause in datasets. Nevertheless, when the limited scope is too large, the model is forced to allocate higher probability to these clauses which are distant from and irrelevant to the emotion word, and then leads to the performance degradation.

### 5.3 Case Study

Essentially, sentiment regularizer (SR.) aims to enlarge the margin between sentiment words and non-sentiment words. The question is, will the model focus on the words with sentiment polarity? We randomly choose one example (**Ex.2**) from the Chinese dataset to visualize its attention distribution and compare the difference between plusing sentiment regularizer or not.

**Ex.2** 一个老乡来到店里问起小美的情况, $(c_{-2})$|大熊抱怨小美不好. $(c_{-1})$|她听到后感到既伤心又生气. $(c_0)$

**Ex.2** *A fellow came to store asking XiaoMei's situation,* $(c_{-2})$*|DaXiong complained XiaoMei not good.* $(c_{-1})$*| She heard it then felt both sad and angry.* $(c_0)$

In this example, the cause of emotion word *sad* is in $(c_0)$. The visualization results are shown in Fig 3, we can observe that the model without sentiment regularizer most focus on non-sentiment words such as *XiaoMei*, *she* and *heard it* which are inessential to provoke the emotion. However, when we plus the SR. into model, we can see an obvious weights shift on attention distribution.

| Emotion cause events | RHNN results |
|---|---|
| 1 ***I was immediately *ashamed* of myself for my vanity, for having assumed that he wanted me to stay with him forever.*** *I'm sorry, that was a little arrogant* | for having assumed that he wanted me to stay with him forever |
| 2 ***He hopes of being *admitted* to a sight of the young ladies,*** *of whose beauty he had heard much. But he saw only the father.* | of whose beauty he had heard much |
| 3 ***I didn't know where in the hell you was,*** *said Ennis, four years, I about *give up* on you.* | **Null** |

Table 6: The error instances of RHNN model for emotion cause analysis.

More clearly, the model captures the sentiment words *complained* and *not good* which are crucial to identify the emotion cause. This shows that our model with sentiment regularizer is more effective in extracting the most important keywords relating to the emotion cause. Also, better results are obtained using sentiment regularizer, this is consistent with what we observed in Table 4.

Finally, we perform error analysis to understand what types of errors are introduced with the proposed model, focusing on three cases from the English dataset. The results are listed in Table 6, where the first column depicts the content of the emotion cause events and the second column depicts the emotion causes identified by RHNN. As shown in Table 6, the emotion causes appear in bold and emotion word is labeled between *.

From Table 6, we can find that there are two clauses contain the emotion cause in event 1. However, our model only detects one emotion cause clause. In event 2, our model has an error prediction. One possible reason is that our model is prone to treating clauses which contain sentiment words as emotion cause. RHNN extracts nothing from event 3, it may be due to the reason that the far distance between the emotion word and emotion cause clause, resulting in a difficult understanding of causal relations. Our proposed model is capable of getting rich emotion cause

cues with knowledge-based regularizations. However, it also introduces some noisy into emotion cause analysis.

## 6 Related Work

Emotion classification is an important fundamental aspect of sentiment analysis. Going one step further, emotion cause analysis (ECA) which aims to discover the reason behind emotions, can be constructive to guide the direction of future work, i.e., improving the quality of products or services according to the emotion causes of comments provided by users. In this section, we describe the related work on emotion cause analysis.

Lee et al. (2010) first gave the formal definition of emotion cause analysis and manually constructed a dataset from the Academia Sinica Balanced Chinese Corpus. Based on this corpus, Chen et al. (2010) designed two sets of features built on six groups of linguistic cues to detect emotion cause. Support vector machines (SVMs) and conditional random fields (CRFs) were investigated to detect cause or non-cause text with extended rule-based features in existing methods (Gui et al., 2014; Ghazi et al., 2015). Other than rule-based methods, Russo et al. (2011) proposed a crowdsourcing method to construct a commonsense knowledge base for emotion cause extraction in Italian newspaper articles. But it is challenging to extend the common-sense knowledge base automatically. Recently, Gui et al. (2016) proposed a multi-kernel based method to identify the emotion cause from a manually annotated emotion cause corpus. Xu et al. (2019) proposed a method based on learning to re-rank candidate emotion cause clauses with extracting a number of emotion-dependent and emotion-independent features. However, these methods are heavily dependent on the expensive human-based features and are too difficult in a real-world application.

Inspired by the success of neural network methods, deep neural models and attention mechanisms have been widely used in emotion cause analysis. Gui et al. (2017) proposed a novel deep neural network which regarded emotion cause analysis as a question-answering task. In this study, a convolution-based memory network was introduced to store the context information. Li et al. (2018a) considered the context around the emotion word as a query instead of only emotion word to model the mutual impacts between each candidate clause and the emotion clause. Cheng et al. (2017) constructed a corpus based on Chinese microblog and proposed to detect emotion cause using multiple-user structures. Besides, Yu et al. (2019) proposed a multiple-level hierarchical network-based clause selection strategy. Li et al. (2019) proposed a multi-attention-based neural model to capture the mutual influences between the emotion clause and each candidate clause, and then generate the representations for the above two clauses separately. This method achieves the current best performance. However, the existing approaches usually focus on the local textural information, ignoring the discourse structure (Zubiaga et al., 2018), and prior knowledge such as sentiment lexicon (Qian et al., 2017) and relative position information, which can provide important emotion cues for emotion cause analysis task.

## 7 Conclusion and Future Work

In this paper, we provide a regularized hierarchical neural network (RHNN) for emotion cause analysis. The proposed model aggregates discourse context information through a hierarchical learning structure and restrains the parameters with knowledge-based regularizations. We evaluate the proposed model on two public datasets in different languages. The experimental results demonstrate that our proposed method achieves the state-of-the-art performance on both datasets and extensive analysis confirms the feasibility of incorporating the discourse context and knowledge-based regularizations.

To preserve the simplicity of the proposed model, we do not consider document as a tree structure. In the future, we will exploit how to incorporate discourse parse tree or discourse relations into emotion cause analysis task to further improve the performance.

## 8 Acknowledgements

This work was supported by National Natural Science Foundation of China U1636103, 61632011, 61876053, Shenzhen Foundational Research Funding JCYJ20180507183527919, JCYJ20180507183608379, Key Technologies Research and Development Program of Shenzhen JSGG20170817140856618, Joint Research Program of Shenzhen Securities Information Co., Ltd. No. JRPSSIC2018001, and the EU-H2020 (grant no. 794196).

## References


Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. 2017. Recurrent attention network on memory for aspect sentiment analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 452–461.

Ying Chen, Wenjun Hou, Xiyao Cheng, and Shoushan Li. 2018. Joint learning for emotion classification and emotion cause detection. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 646–651.

Ying Chen, Sophia Yat Mei Lee, Shoushan Li, and Chu-Ren Huang. 2010. Emotion cause detection with linguistic constructions. In *COLING 2010, 23rd International Conference on Computational Linguistics, Proceedings of the Conference, 23-27 August 2010, Beijing, China*, pages 179–187.

Xiyao Cheng, Ying Chen, Bixiao Cheng, Shoushan Li, and Guodong Zhou. 2017. An emotion cause corpus for chinese microblogs with multiple-user structures. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, 17(1):6.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734.

Zhendong Dong, Qiang Dong, and Changling Hao. 2006. Hownet and the computation of meaning.

Qinghong Gao, H Jiannan, X Ruifeng, G Lin, Y He, KF Wong, and Q Lu. 2017. Overview of ntcir-13 eca task. In *Proceedings of the NTCIR-13 Conference*.

Diman Ghazi, Diana Inkpen, and Stan Szpakowicz. 2015. Detecting emotion stimuli in emotion-bearing sentences. In *Computational Linguistics and Intelligent Text Processing - 16th International Conference, CICLing 2015, Cairo, Egypt, April 14-20, 2015, Proceedings, Part II*, pages 152–165.

Lin Gui, Jiannan Hu, Yulan He, Ruifeng Xu, Qin Lu, and Jiachen Du. 2017. A question answering approach for emotion cause extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 1593–1602.

Lin Gui, Dongyin Wu, Ruifeng Xu, Qin Lu, and Yu Zhou. 2016. Event-driven emotion cause extraction with corpus construction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1639–1649.

Lin Gui, Li Yuan, Ruifeng Xu, Bin Liu, Qin Lu, and Yu Zhou. 2014. Emotion cause detection with linguistic construction in chinese weibo text. In *Natural Language Processing and Chinese Computing*, pages 457–464. Springer.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Lun-Wei Ku, Yu-Ting Liang, and Hsin-Hsi Chen. 2006. Opinion extraction, summarization and tracking in news and blog corpora. In *Computational Approaches to Analyzing Weblogs, Papers from the 2006 AAAI Spring Symposium, Technical Report SS-06-03, Stanford, California, USA, March 27-29, 2006*, pages 100–107.

Sophia Yat Mei Lee, Ying Chen, and Chu-Ren Huang. 2010. A text-driven rule-based system for emotion cause detection. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 45–53. Association for Computational Linguistics.

Shoushan Li, Lei Huang, Rong Wang, and Guodong Zhou. 2015. Sentence-level emotion classification with label and context dependence. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1045–1053.

Weiyuan Li and Hua Xu. 2014. Text-based emotion classification using emotion cause extraction. *Expert Syst. Appl.*, 41(4):1742–1749.

Xiangju Li, Shi Feng, Daling Wang, and Yifei Zhang. 2019. Context-aware emotion cause analysis with multi-attention-based neural network. *Knowl.-Based Syst.*, 174:205–218.

Xiangju Li, Kaisong Song, Shi Feng, Daling Wang, and Yifei Zhang. 2018a. A co-attention neural network model for emotion cause analysis with emotional context awareness. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language*

*Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 4752–4757.

Xin Li, Lidong Bing, Wai Lam, and Bei Shi. 2018b. Transformation networks for target-oriented sentiment classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 946–956.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing System*, pages 3111–3119.

Bo Pang and Lillian Lee. 2007. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135.

Qiao Qian, Minlie Huang, Jinhao Lei, and Xiaoyan Zhu. 2017. Linguistically regularized LSTM for sentiment classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1679–1689.

Irene Russo, Tommaso Caselli, Francesco Rubino, Ester Boldrini, and Patricio Martínez-Barco. 2011. Emocause: An easy-adaptable approach to extract emotion cause contexts. In *Proceedings of the 2nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis*, pages 153–160.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *HLT/EMNLP 2005, Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, 6-8 October 2005, Vancouver, British Columbia, Canada*, pages 347–354.

Bo Xu, Hongfei Lin, Yuan Lin, Yufeng Diao, Liang Yang, and Kan Xu. 2019. Extracting emotion causes using learning to rank methods from an information retrieval perspective. *IEEE Access*, 7:15573–15583.

Ruifeng Xu, Chengtian Zou, Yanzhen Zheng, Xu Jun, Lin Gui, Bin Liu, and Xiaolong Wang. 2013. A new emotion dictionary based on the distinguish of emotion expression and emotion cognition. *Journal of Chinese Information Processing*, 27(6):82–90.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. 2016. Hierarchical attention networks for document classification. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 1480–1489.

Jianfei Yu, Luís Marujo, Jing Jiang, Pradeep Karuturi, and William Brendel. 2018. Improving multi-label emotion classification via sentiment classification with dual attention transfer network. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 1097–1102.

Xinyi Yu, Wenge Rong, Zhuo Zhang, Yuanxin Ouyang, and Zhang Xiong. 2019. Multiple level hierarchical network-based clause selection for emotion cause extraction. *IEEE Access*, 7:9071–9079.

Deyu Zhou, Xuan Zhang, Yin Zhou, Quan Zhao, and Xin Geng. 2016. Emotion distribution learning from texts. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 638–647.

Arkaitz Zubiaga, Elena Kochkina, Maria Liakata, Rob Procter, Michal Lukasik, Kalina Bontcheva, Trevor Cohn, and Isabelle Augenstein. 2018. Discourse-aware rumour stance classification in social media using sequential classifiers. *Inf. Process. Manage.*, 54(2):273–290.

# Automatic Argument Quality Assessment - New Datasets and Methods

**Assaf Toledo***, **Shai Gretz***, **Edo Cohen-Karlik***, **Roni Friedman***, **Elad Venezian**, **Dan Lahav**, **Michal Jacovi**, **Ranit Aharonov** and **Noam Slonim**
IBM Research

## Abstract

We explore the task of automatic assessment of argument quality. To that end, we actively collected $6.3k$ arguments, more than a factor of five compared to previously examined data. Each argument was explicitly and carefully annotated for its quality. In addition, $14k$ pairs of arguments were annotated independently, identifying the higher quality argument in each pair. In spite of the inherent subjective nature of the task, both annotation schemes led to surprisingly consistent results. We release the labeled datasets to the community. Furthermore, we suggest neural methods based on a recently released language model, for argument ranking as well as for argument-pair classification. In the former task, our results are comparable to state-of-the-art; in the latter task our results significantly outperform earlier methods.

## 1 Introduction

Computational argumentation has been receiving growing interest in the NLP community in recent years (Reed, 2016). With this field rapidly expanding, various methods have been developed for subtasks such as argument detection (Lippi and Torroni, 2016; Levy et al., 2014; Rinott et al., 2015), stance detection (Bar-Haim et al., 2017) and argument clustering (Reimers et al., 2019).

Recently, IBM introduced *Project Debater*, the first AI system able to debate humans on complex topics. The system participated in a live debate against a world champion debater, and was able to mine arguments, use them for composing a speech supporting its side of the debate, and also rebut its human competitor.[1] The underlying technology is intended to enhance decision-making.

More recently, IBM also introduced *Speech by Crowd*, a service which supports the collection of free-text arguments from large audiences on debatable topics to generate meaningful narratives. A real-world use-case of Speech by Crowd is in the field of civic engagement, where the aim is to exploit the wisdom of the crowd to enhance decision making on various topics. There are already several public organizations and commercial companies in this domain, e.g., Decide Madrid[2] and Zencity.[3] As part of the development of Speech by Crowd, $6.3k$ arguments were collected from contributors of various levels, and are released as part of this work.

An important sub-task of such a service is the automatic assessment of argument quality, which has already shown its importance for prospective applications such as automated decision making (Bench-Capon et al., 2009), argument search (Wachsmuth et al., 2017b), and writing support (Stab and Gurevych, 2014). Identifying argument quality in the context of Speech by Crowd allows for the top-quality arguments to surface out of many contributions.

Assessing argument quality has driven practitioners in a plethora of fields for centuries from philosophers (Aristotle et al., 1991), through academic debaters, to argumentation scholars (Walton et al., 2008). An inherent difficulty in this domain is the presumably subjective nature of the task. Wachsmuth et al. (2017a) proposed a taxonomy of quantifiable dimensions of argument quality, comprised of high-level dimensions such as *cogency* and *effectiveness*, and sub-dimensions such as *relevance* and *clarity*, that together enable the assignment of a holistic quality score to an argument.

Habernal and Gurevych (2016b) and Simpson

*These authors equally contributed to this work.

[1] For more details: https://www.research.ibm.com/artificial-intelligence/project-debater/live/

[2] https://decide.madrid.es

[3] https://zencity.io



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5625–5635, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

and Gurevych (2018) take a relative approach and treat the problem as relation classification. They focus on *convincingness* – a primary dimension of quality – and determine it by comparing pairs of arguments with similar stance. In this view, the convincingness of an individual argument is a derivative of its relative convincingness: arguments that are judged as more convincing when compared to others are attributed higher scores. These works explore the labeling and automatic assessment of argument convincingness using two datasets introduced by Habernal and Gurevych (2016b): *UKPConvArgRank* (henceforth, *UKPRank*) and *UKPConvArgAll*, which contain $1k$ and $16k$ arguments and argument-pairs, respectively.

Gleize et al. (2019) also take a relative approach to argument quality, focusing on ranking convincingness of *evidence*. Their solution is based on a Siamese neural network, which outperforms the results achieved in Simpson and Gurevych (2018) on the *UKP* datasets, as well as several baselines on their own dataset, *IBM-ConvEnv*.[4]

Here, we extend earlier work in several ways: (1) introducing a large dataset of actively collected arguments, carefully annotated for quality; (2) suggesting a method for argument-pair classification, which outperforms state-of-the-art accuracy on available datasets; (3) suggesting a method for individual argument ranking, which achieves results comparable to the state of the art.

Our data was collected actively, via a dedicated user interface. This is in contrast to previous datasets, which were sampled from online debate portals. We believe that our approach to data collection is more controlled and reduces noise in the data, thus making it easier to utilize it in the context of learning algorithms (see Section 7).

Moreover, we applied various cleansing methods to ensure the high quality of the contributed data and the annotations, as detailed in Section 3.

We packaged our data in the following datasets, which are released to the research community[5]:

- *IBM-ArgQ-6.3kArgs* - the full dataset, comprised of all $6.3k$ arguments that were collected and annotated with an individual quality score in the range $[0, 1]$.
- *IBM-ArgQ-14kPairs* - $14k$ argument pairs annotated with a relative quality label, indicating which argument is of higher quality.
- *IBM-ArgQ-5.3kArgs* - the subset of $5.3k$ arguments from *IBM-ArgQ-6.3kArgs* that passed our cleansing process. This set is used in the argument ranking experiments in Section 9.2, henceforth: *IBMRank*.
- *IBM-ArgQ-9.1kPairs* - the subset of $9.1k$ argument pairs from *IBM-ArgQ-14kPairs* that passed our cleansing process, used in the argument-pair classification experiments in Section 9.1. Henceforth: *IBMPairs*.

The dataset *IBMRank* differs from *UKPRank* in a number of ways. Firstly, *IBMRank* includes $5.3k$ arguments, which make it more than 5 times larger than *UKPRank*. Secondly, the arguments in *IBMRank* were collected actively from contributors. Thirdly, *IBMRank* includes *explicit* quality-labeling of all individual arguments, which is absent from earlier data, enabling us to explore the potential of training quality-prediction methods on top of such labels, presumably easier to collect.

Finally, with the abundance of technologies such as automated personal assistants, we envision automated argument quality assessment expanding to applications that include oral communication. Such use-cases pose new challenges, overlooked by prior work, that mainly focused on written arguments. As an initial attempt to address these issues, in the newly contributed data we guided annotators to assess the quality of an argument within the context of using the argument as-is to generate a persuasive *speech* on the topic. Correspondingly, we expect these data to reflect additional quality dimensions – e.g., a quality premium on efficiently phrased arguments, and low tolerance to blunt mistakes such as typos that may lead to poorly stated arguments.

## 2 Argument Collection

As part of the development of Speech by Crowd, online and on-site experiments have been conducted, enabling to test the ability of the service to generate a narrative based on collected arguments. Arguments were collected from two main sources: (1) debate club members, including all levels, from novices to experts; and (2) a broad audience of people attending the experiments.

[4] As this work is relatively recent and was published after our submission, we were not able to compare to it.

[5] https://www.research.ibm.com/haifa/dept/vst/debating_data.shtml#ArgumentQuality

For the purpose of collecting arguments, we first selected 11 well known controversial concepts, common in the debate world, such as *Social Media*, *Doping in Sports* and *Flu vaccination*. Using debate jargon, each concept is used to phrase two "motions", by proposing two specific and opposing policies or views towards that concept. For example, for the concept *Autonomous Cars*, we suggested the motions *We should promote Autonomous Cars* and *We should limit Autonomous Cars*.[6] The full list of motions appears in Table 1 with the number of arguments collected for each.[7]

**Guidelines** Contributors were invited to a dedicated user interface in which they were guided to contribute arguments per concept, using the following concise instructions:

> You can submit as many arguments as you like, both pro and con, using original language and no personal information (i.e. information about an identifiable person).

In addition, to exemplify the type of arguments that we expect to receive, contributors were shown an example of one argument related to the motion, provided by a professional debater. The arguments collected had to have $8 - 36$ words, aimed at obtaining efficiently phrased arguments (longer/shorter arguments were rejected by the UI). In total, we collected $6,257$ arguments.

| Motion | #Args |
|---|---|
| Flu vaccination should be mandatory | 204 |
| Flu vaccination should not be mandatory | 174 |
| Gambling should be banned | 342 |
| Gambling should not be banned | 382 |
| Online shopping brings more harm than good | 198 |
| Online shopping brings more good than harm | 215 |
| Social media brings more harm than good | 879 |
| Social media brings more good than harm | 686 |
| We should adopt cryptocurrency | 172 |
| We should abandon cryptocurrency | 160 |
| We should adopt vegetarianism | 221 |
| We should abandon vegetarianism | 179 |
| We should ban the sale of vvg to minors | 275 |
| We should allow the sale of vvg to minors | 240 |
| We should ban fossil fuels | 146 |
| We should not ban fossil fuels | 116 |
| We should legalize doping in sport | 212 |
| We should ban doping in sport | 215 |
| We should limit autonomous cars | 313 |
| We should promote autonomous cars | 480 |
| We should support information privacy laws | 355 |
| We should discourage information privacy laws | 93 |

Table 1: Motion list and statistics on data collection.

## 3 Argument Quality Labeling

We explored two approaches to labeling argument quality: (a) labeling individual arguments (absolute approach): each individual argument is directly labeled for its quality; and (b) labeling argument pairs (relative approach): each argument pair is labeled for which of the two arguments is of higher quality. In this section we describe the pros and cons of each approach as well as the associated labeling process.

### Approaches to Argument Quality Labeling

The effort in labeling individual arguments scales linearly with the number of arguments, compared to the quadratic scaling of labeling pairs (within the same motion); thus, it is clearly more feasible when considering a large number of arguments. However, the task of determining the quality of arguments in isolation is presumably more challenging; it requires to evaluate the quality of an argument without a clear reference point (except for the motion text). This is where the relative approach has its strength, as it frames the labeling task in a specific context of two competing arguments, and is expected to yield higher inter-annotator agreement. Indeed, a comparative approach is widely used in many NLP applications, e.g. in Chen et al. (2013) for assessing reading difficulty of documents and in Aranberri et al. (2017) for machine translation. In light of these considerations, here we decided to investigate and compare both approaches. We used the Figure Eight platform[8], with a relatively large number of $15 - 17$ annotators per instance, to improve the reliability of the collected annotations.

### 3.1 Labeling Individual Arguments

The goal of this task is to assign a quality score for each individual argument. Annotators were presented with the following binary question per argument:

> Disregarding your own opinion on the topic, would you recommend a friend preparing a speech supporting/contesting the topic to use this argument *as is* in the speech? (yes/no)

All arguments that were collected as described in

[6] Habernal and Gurevych (2016b) uses the term *topic* for what we refer to as *motion*.

[7] In Table 1, *vvg* stands for *violent video games*.

[8] http://figure-eight.com/

Section 2 were labeled in this task. We model the quality of each individual argument as a real value in the range of $[0, 1]$, by calculating the fraction of 'yes' answers. To ensure the annotators will carefully read each argument, the labeling of each argument started with a test question about the stance of the argument towards the concept (*pro* or *con*). The annotators' performance on these test questions was used in the quality control process described in Section 4, and also in determining which pairs of arguments to label.

### 3.2 Labeling Argument Pairs

In this task, annotators were presented with a pair of arguments, having the same stance towards the concept (to reduce bias due to the annotator's opinion), and were asked the following:

> Which of the two arguments would have been preferred by most people to support/contest the topic?

Table 2 presents an example of such an argument pair in which the annotators unanimously preferred the first argument.

| Argument 1 | Argument 2 |
|---|---|
| Children emulate the media they consume and so will be more violent if you don't ban them from violent video games | These are less fun and more harmful games but specifically violent games are played in groups and exclude softer souls |

Table 2: An example of an argument pair for the motion *We should ban the sale of violent video games to minors*. The first argument was unanimously preferred by all annotators.

As mentioned, annotating all pairs in a large collection of arguments is often not feasible. Thus, we focused our attention on pairs that are presumably most valuable to train a learning algorithm. Specifically, we annotated $14k$ randomly selected pairs, that satisfy the following criteria:

1. At least $80\%$ of the annotators agreed on the stance of each argument, aiming to focus on clearly stated arguments.

2. Individual quality scores in each pair differ by at least $0.2$, aiming for pairs with a relatively high chance of a clear winner.

3. The length of both arguments, as measured by number of tokens, differs by $\leq 20\%$, aiming to focus the task on dimensions beyond argument length.

## 4 Quality Control

To monitor and ensure the quality of collected annotations, we employed the following analyses:

**Kappa Analysis** –

1. Pairwise Cohen's kappa ($\kappa$) (Cohen, 1960) is calculated for each pair of annotators that share *at least* $50$ common argument/argument pairs judgments, and based only on those common judgments.

2. *Annotator-*$\kappa$ is obtained by averaging all pairwise $\kappa$ for this annotator as calculated in Step 1, and if and only if this annotator had $\geq 5$ pairwise $\kappa$ values estimated. This is used to ignore annotators as described later.

3. Averaging all *Annotator-*$\kappa$, calculated in Step 2, results in *Task-Average-*$\kappa$.[9]

**Test Questions Analysis** – Hidden embedded test questions, based on ground truth, are often valuable for monitoring crowd work. In our setup, at least one fifth of the judgments provided by each annotator are on test questions. When annotators fail a test question, they are alerted. Thus, beyond monitoring annotator's quality, test questions also provide annotators feedback on task expectations. In addition, an annotator that fails more than a prespecified fraction (e.g., $20\%$) of the test questions is removed from the task, and his judgments are ignored.

**High Prior Analysis** – An annotator that always answers 'yes' to a particular question should obviously be ignored; more generally, we discarded the judgments contributed by annotators with a relatively high prior to answer positively on the presented questions.

Note, if an annotator is discarded due to failure in any of the above analyses, he is further discarded from the estimation of *Annotator-*$\kappa$ and *Task-Average-*$\kappa$.

[9] It is noted that some annotators remain without valid *Annotator-*$\kappa$ and cannot be filtered out based on their $\kappa$. Similarly, those annotators do not contribute to the *Task-Average-*$\kappa$. However, in both annotation tasks, those annotators contributed only $0.01 - 0.03$ of the judgments collected.

## 5 Data Cleansing

### 5.1 Cleansing of Individual Arguments Judgments

To enhance the quality of the collected data, we discard judgments by annotators who (1) failed $\geq 20\%$ of the test-questions[10]; and/or (2) obtained *Annotator-*$\kappa \leq 0.35$ in the stance judgment task; and/or (3) answered 'yes' for $\geq 80\%$ of the quality judgment questions. Finally, we discarded arguments that were left with less than 7 valid judgments. This process left us with $5.3k$ arguments, each with $11.4$ valid annotations on average. The *Task-Average-*$\kappa$ was $0.69$ on the stance question and $0.1$ on the quality question. We refer to the full, unfiltered, set as *IBM-ArgQ-6.3kArgs*, and to the filtered set as *IBM-ArgQ-5.3kArgs* (*IBMRank*).

For completeness, we also attempted to utilize an alternative data cleansing tool, MACE (Hovy et al., 2013). We ran MACE with a threshold k, keeping the top k percent of arguments according to their entropy. We then re-calculated *Task-Average-*$\kappa$ on the resulting dataset. We ran MACE with k=0.95, as used in Habernal and Gurevych (2016b), and with k=0.85, as this results in a dataset similar in size to *IBMRank*. The resulting *Task-Average-*$\kappa$ is $0.08$ and $0.09$, respectively, lower than our reported $0.1$. We thus maintain our approach to data cleansing as described above.

The low average $\kappa$ of $0.1$ for quality judgments is expected due to the subjective nature of the task, but nonetheless requires further attention. Based on the following observations, we argue that the labels inferred from these annotations are still meaningful and valuable: (1) the high *Task-Average-*$\kappa$ on the stance task conveys the annotators carefully read the arguments before providing their judgments; (2) we report high agreement of the individual quality labels with the argument-pair annotations, to which much better $\kappa$ values were obtained (see Section 6.1); (3) we demonstrate that the collected labels can be successfully used by a neural network to predict argument ranking (see section 9.2), suggesting these labels carry a real signal related to arguments' properties.

### 5.2 Cleansing of Argument Pair Labeling

To enhance the quality of the collected pairwise data, we discard judgments by annotators who (1) failed $\geq 30\%$ of the test-questions; and/or (2) obtained *Annotator-*$\kappa \leq 0.15$ in this task. Here, the test questions were directly addressing the (relative) quality judgment of pairs, and not the stance of the arguments. In initial annotation rounds the test questions were created based on the previously collected individual arguments labels - considering pairs in which the difference in individual quality scores was $\geq 0.6$.[11] In following annotation rounds, the test questions were defined based on pairs for which $\geq 90\%$ of the annotators agreed on the winning pair. Following this process we were left with an average of $15.9$ valid annotations for each pair, and with *Task-Average-*$\kappa$ of $0.42$ on the quality judgments – a relatively high value for such a subjective task. As an additional cleansing, for training the learning algorithms, we considered only pairs for which $\geq 70\%$ of the annotators agreed on the winner, leaving us with a total of $9.1k$ pairs for training and evaluation. We refer to the full, unfiltered, set as *IBM-ArgQ-14kPairs*, and to the filtered set as *IBM-ArgQ-9.1kPairs*.

## 6 Data Consistency

### 6.1 Consistency of Labeling Tasks

Provided with both individual and pairwise quality labeling, we estimated the consistency of these two approaches. For each pair of arguments, we define the *expected winning argument* as the one with the higher *individual* argument score, and compare that to the *actual winning argument*, namely the argument preferred by most annotators when considering the pair directly. Overall, in $75\%$ of the pairs the actual winner was the expected one. Moreover, when focusing on pairs in which the individual argument scores differ by $> 0.5$, this agreement reaches $84.3\%$ of pairs.

### 6.2 Reproducibility Evaluation

An important property of a valid annotation is its reproducibility. For this purpose, a random sample of $500$ argument pairs from the *IBMPairs* dataset was relabeled by the crowd. This relabeling took place a few months after the main annotation tasks, with the exact task and data cleansing methods that were employed originally. For measuring correlation, the following $A\_score$ was defined: the fraction of valid annotations selecting

[10] Since quality judgments are relatively subjective, we focused the test questions on the stance question.

[11] Note, that annotators have the option of contesting problematic test questions, and thus unfitting ones were disabled during the task by our team.

"argument $A$" in an argument pair $(A, B)$ as having higher quality, out of the total number of valid annotations. Pearson's correlation coefficient between $A\_score$ in initial and secondary annotation of the defined sample was $0.81$.

A similar process was followed with the individual arguments quality labeling. Instead of relabeling, we split existing annotations to two even groups. We chose only individual arguments in which at least $14$ valid annotations remained after data cleansing ($1,154$ such arguments). This resulted in two sets of labels for the same data, each based on at least 7 annotations. Pearson's correlation coefficient between quality scores of the two sets was $0.53$. We then divided the quality score, which ranges between 0 to 1, to 10 equal bins. The bin frequency counts between the two sets are displayed in the heatmap in Figure 1.

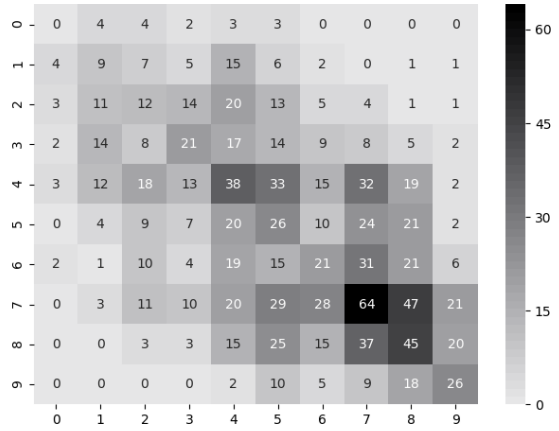


Figure 1: Counts of quality score bins between two equally sized sets of annotators.

### 6.3 Transitivity Evaluation

Following Habernal and Gurevych (2016b), we further examined to what extent our labeled pairs satisfy *transitivity*. Specifically, a triplet of arguments $(A, B, C)$ in which $A$ is preferred over $B$, and $B$ is preferred over $C$, is considered transitive if and only if $A$ is also preferred over $C$. We examined all $892$ argument triplets for which all pair-wise combinations were labeled, and found that transitivity holds in $96.2\%$ of the triplets, further strengthening the validity of our data.

## 7 Comparison of *IBMRank* and *UKPRank*

A distinctive feature of our *IBMRank* dataset is that it was collected actively, via a dedicated user interface with clear instructions and enforced length limitations. Correspondingly, we end up with cleaner texts, that are also more homogeneous in terms of length, compared to the *UKPRank* that relies on arguments collected from debate portals.

**Text Cleanliness**

We counted tokens representing a malformed span of text in *IBMRank* and *UKPRank*. These are HTML markup tags, links, excessive punctuation[12], and tokens not found in GloVE vocabulary (Pennington et al., 2014). Our findings show that 94.78% of *IBMRank* arguments contain no malformed text, 4.38% include one such token, and 0.71% include two such tokens. In the case of *UKPRank*, only 62.36% of the arguments are free of malformed text, 17.59% include one such token, and 20.05% include two or more tokens of malformed text.

**Text Length**

As depicted in Figure 2, the arguments in *IBMRank* are substantially more homogeneous in their length compared to *UKPRank*. A potential drawback of the length limitation is that it possibly prevents any learning system from being able to model long arguments correctly. However, by imposing this restriction we expect our quality labeling to be less biased due to argument length, holding greater potential to reveal other properties that contribute to argument quality. We confirmed this intuition with respect to the argument pair labeling as described in Section 9.1.

**Data Size and Individual Argument Labeling**

Finally, *IBMRank* covers $5,298$ arguments, compared to $1,052$ in *UKPRank*. In addition, in *UKPRank* no individual labeling is provided, and individual quality scores are inferred from pairs labeling. In contrast, for *IBMRank* each argument is individually labeled for quality, and we explicitly demonstrate the consistency of these individual labeling with the provided pairwise labeling.

## 8 Methods

In this section we describe neural methods for predicting the individual score and the pair-wise

[12] Sequences of three or more punctuation characters, e.g. *"?!?!?!"*

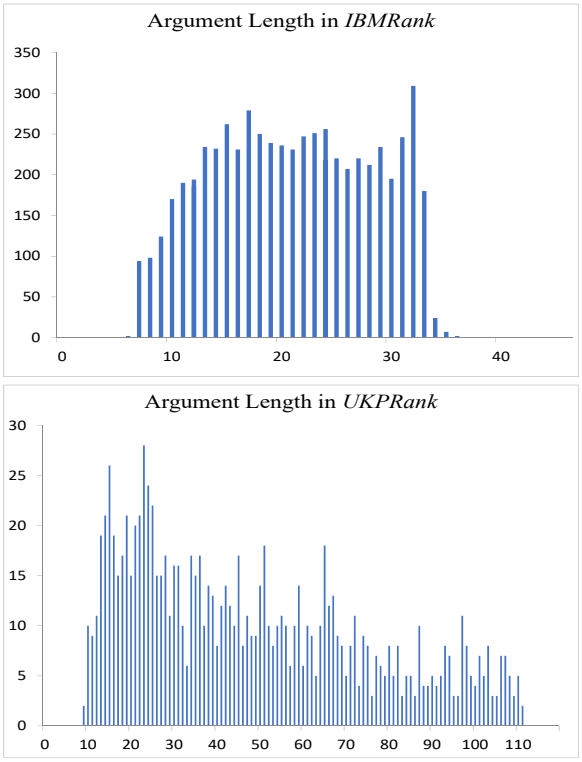


Figure 2: Histograms of argument length in *IBMRank* and *UKPRank*. X-axis: length (token count). Y-axis: the number of arguments at that length.

classification of arguments. We devise two methods corresponding to the two newly introduced datasets. Our methods are based upon a powerful language representational model named Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018) which achieves state-of-the-art results on a wide range of tasks in NLP (Wang et al. (2018), Rajpurkar et al. (2016, 2018)). BERT has been extensively trained over large corpora to perform two tasks: (1) *Masked Language Model* - randomly replace words with a predefined token, `[MASK]`, and predict the missing word. (2) *Next Sentence Prediction* - given a pair of sentences `A` and `B`, predict whether sentence `B` follows sentence `A`. Due to its bidirectional nature, BERT achieves remarkable results when fine-tuned to different tasks without the need for specific modifications per task. For further details refer to Devlin et al. (2018).

### 8.1 Argument-Pair Classification

We fine-tune BERT's Base Uncased English pre-trained model for a binary classification task.[13] The fine-tuning process is initialized with weights from the general purpose pre-trained model and a task specific weight matrix $W_{out} \in R^{768\times 2}$ is added to the 12-layer base network. Following standard practice with BERT, given a pair of arguments `A` and `B`, we feed the network with the following sequence '`[CLS]A[SEP]B`'. The `[SEP]` token indicates to the network that the input is to be treated as a pair and `[CLS]` is a token which is used to obtain contextual embedding for the entire sequence. The network is trained for 3 epochs with a learning rate of $2^{-5}$. We refer to this model as *Arg-Classifier*.

### 8.2 Argument Ranking

For training a model to output a score between $[0, 1]$ we obtain contextual embeddings from the Arg-Classifier fine-tuned model. We concatenate the last 4 layers of the model output to obtain an embedding vector of size $4 \times 768 = 3072$. The embedding vectors are used as input to a neural network with a single output and one hidden layer with 300 neurons. In order for the network to output values in $[0, 1]$, we use a sigmoid activation, $\sigma_{sigmoid}(x) = \frac{1}{1+e^{-x}}$. Denote the weight matrices $W_1 \in R^{3072\times 300}$ and $W_2 \in R^{300\times 1}$, the regressor model, $f_R$, is a 2-layered neural network with $\sigma_{relu}(x) = \max\{0, x\}$ activation. $f_R$ can be written as:[14]

$$f_R(x) = \sigma_{sigmoid}\left(W_2^T \sigma_{relu}(W_1^T x)\right)$$

where $x \in R^{3072}$ is the embedding vector representing an argument. We refer to this regression model as *Arg-Ranker*.

## 9 Experiments

### 9.1 Argument-Pair Classification

In this section we evaluate the methods described in Section 8. First, we evaluate the accuracy of *Arg-Classifier* on our *IBMPairs* dataset and on *UKPConvArgStrict* (henceforth, *UKPStrict*), the filtered argument pairs dataset of Habernal and Gurevych (2016b), in k-fold cross-validation.[15] We calculate accuracy and ROC area under curve (AUC) for each fold, and report the weighted averages over all folds. We also evaluate Simpson and Gurevych (2018)'s *GPPL* median heuristic method with *GloVe + ling* features in cross-

[13] Initial experiments with BERT's Large model showed only minor improvements, so for the purpose of the experiments detailed in Section 9 we used the Base model.

[14] We omit bias terms for readability.

[15] 22 and 32 folds respectively.

validation on our *IBMPairs* dataset. For completeness, we quote Simpson and Gurevych (2018)'s figures of *GPPL opt.* and *GPC* on *UKPStrict*.[16] We add a simple baseline classifying arguments based on their token count (*Arg-Length*).

| | IBMPairs | | |
|---|---|---|---|
| | Arg-Length | Arg-Classifier | GPPL |
| Acc. | .55 | **.80** | .71 |
| AUC | .59 | **.86** | .78 |

| | UKPStrict | | | | |
|---|---|---|---|---|---|
| | Arg-Length | Arg-Classifier | GPPL | GPPL opt. | GPC |
| Acc. | .76 | **.83** | .79 | .80 | .81 |
| AUC | .78 | **.89** | .87 | .87 | **.89** |

Table 3: Accuracy and AUC on *IBMPairs* and *UKPStrict*.

As can be seen in Tables 3, *Arg-Classifier* improves on the *GPPL* method on both datasets ($p \ll .01$ using two-tailed Wilcoxon signed-rank test).[17] We note that *Arg-Classifier*'s accuracy on the *UKPStrict* set is higher than all methods tested on this dataset in Habernal and Gurevych (2016b); Simpson and Gurevych (2018). Interestingly, all methods reach higher accuracy on *UKPStrict* compared to *IBMPairs*, presumably indicating that the data in *IBMPairs* is more challenging to classify. With regards to *Arg-Length*, we can see that it is inaccurate on *IBMPairs* but achieves a respectable result on *UKPStrict*. This is in agreement with Habernal and Gurevych (2016a) who analyzed the reasons that annotators provided for their labeling. In most cases the reason indicated preference for arguments with more information – which is what longer arguments tend to be better at. This further strengthens the value of creating *IBMPairs* and *IBMRank* as much more homogeneous datasets in terms of argument length.

### 9.2 Argument Ranking

We proceed to evaluate the *Arg-Ranker* on the *IBMRank* and *UKPRank* datasets in k-fold cross-validation, and report weighted correlation measures. We also evaluate the *Arg-Ranker* by feeding it vanilla BERT embeddings, instead of the fine-tuned embeddings generated by the *Arg-Classifier* model. We refer to this version as *Arg-Ranker-base*. In both *Arg-Ranker* and *Arg-Ranker-base* evaluations we report the mean of 3 runs.[18]

| | IBMRank | | UKPRank | | |
|---|---|---|---|---|---|
| | Arg-Ranker-base | Arg-Ranker | Arg-Ranker-base | Arg-Ranker | GPPL |
| $r$ | .41 | **.42** | .44 | **.49** | .45 |
| $\rho$ | .38 | **.41** | .57 | .59 | **.65** |

Table 4: Pearson's ($r$) and Spearman's ($\rho$) correlation of *Arg-Ranker-base*, *Arg-Ranker* and *GPPL* on the *IBMRank* and *UKPRank* datasets.

As can be seen in Table 4, on the *UKPRank* dataset, *Arg-Ranker* is slightly better than *GPPL* for Pearson's correlation, but slightly worse for Spearman's correlation. Additionally, using direct BERT embeddings provides worse correlation[19] than using the *Arg-Classifier* embeddings for both datasets, justifying its use. Finally, similarly to the findings in the argument-pair classification task, the *IBMRank* dataset is harder to predict.[20]

## 10 Error Analysis

We present a qualitative analysis of examples that the *Arg-Classifier* and *Arg-Ranker* models did not predict correctly. For each of the argument-pair and ranking tasks, we analyzed $50 - 100$ arguments from three motions on which the performance of the respective model was poor. For each motion we selected the arguments in which the model was most confident in the wrong direction.

A prominent insight from this analysis, common to both models, is that the model tends to fail when the argument persuasiveness outweighs its delivery quality (such as bad phrasing or typos). An example of this is shown in row 1 of Table 5. In this case, Argument2 is labeled as having a higher quality, even though it contains multiple typos, and thus is typical to arguments that the model was trained to avoid selecting.

Another phenomenon that both our models fail to address is arguments that are off-topic, too provocative or not grounded. An example of this, from the argument-pair task, is shown in row 2 - Argument2 is presumably considered harsh by annotators, even though it is fine in terms of gram-

[16] We were unable to reproduce the results reported in Simpson and Gurevych (2018) by running the *GPPL opt.* and *GPC* algorithms on the *UKPStrict* dataset. We have approached the authors and reported the issue, which was not solved by the time this paper was published, and hence we only quote the figures as reported there.

[17] The results per fold in both tasks are included in the supplementary material.

[18] The *GPPL* regressor of Simpson and Gurevych (2018) relies on pair-wise (relative) labeling of arguments and as a result it cannot be used for predicting the individual (absolute) labeling of arguments, as in *IBMRank*.

[19] Significantly for the *IBMRank* data on both measures, and for the *UKPRank* on Pearson's correlation, $p \ll .05$.

[20] For the experiments on *IBMRank*, we included by mistake a small fraction of arguments which actually should have been filtered. The effect on the results is minimal.

| Motion | Type | Argument1 | Argument2 |
|---|---|---|---|
| We should ban fossil fuels | Impact over delivery | *the only way to provide any space for energy alternatives to enter the market is by artificially decreasing the power of fossil fuels through a ban.* | **fossil fuels are bad for the environment, they have so2 in them that is the thing that maks acid rain and it is today harming the environment and will only be wors.** |
| Flu vaccination should not be mandatory | Provocative or not grounded | **the only responsible persons for kids are their parents. if they dont think that their kids should get the vaccine its their own decision.** | *the body has an automatic vaccination due to evolution, those who got sick and died are the weakest link and we are better off without them* |
| We should abandon vegetarianism | Consistent annotator preference | **it's harder to get all the things you need for a balanced diet while being vegetarian.** | *animals deserve less rights than humans, and it is legitimate for humans to prioritize their enjoyment over the suffering of animals.* |

Table 5: Examples of argument pairs for which there is a high difference between the argument selected by the annotators, marked in bold, and the argument predicted to be of higher quality by the model, marked in italics.

matical structure and impact on the topic. These types of arguments are becoming more important to recognize, especially in the "fake-news" era. We leave dealing with them for future work.

Finally, we also notice certain arguments were consistently preferred by annotators, regardless of the quality of the opposing argument. This is a pattern relevant only to the *Arg-Classifier* model, shown in row 3.

## 11 Conclusions and Future Work

A significant barrier in developing automatic methods for estimating argument quality is the lack of suitable data. An important contribution of this work is a newly introduced data composed of $6.3k$ carefully annotated arguments, compared to $1k$ arguments in previously considered data. Another barrier is the inherent subjectivity of the *manual* task for determining argument quality. To overcome this issue, we employed a relatively large set of crowd annotators to consider each instance, associated with various measures to ensure the quality of the annotations associated with the released data. In addition, while previous work focused on arguments collected from web debate portals, here we collected arguments via a dedicated interface, enforcing length limitations, and providing contributors with clear guidance. Moreover, previous work relied solely on annotating pairs of arguments, and used these annotations to *infer* the individual ranking of arguments; in contrast, here, we annotated all individual arguments for their quality, and further annotated $14k$ pairs. This two–fold approach allowed us, for the first time, to explicitly examine the relation between relative (pairwise) annotation and explicit (individual) annotation of argument quality. Our analysis suggests that these two schemes provide relatively consistent results. In addition, these annotation efforts may complement each other. As pairs of arguments with a high difference in individual quality scores appear to agree with argument-pair annotations, one may deduce the latter from the former. Thus, it may be beneficial to dedicate the more expensive pair-wise annotation efforts to pairs in which the difference in individual quality scores is small, reminiscent of active learning (Settles, 2009). In future work we intend to further investigate this approach, as well as explore in more detail the low fraction of cases where these two schemes led to clearly different results.

The second contribution of this work is suggesting neural methods, based on Devlin et al. (2018), for argument ranking as well as for argument-pair classification. In the former task, our results are comparable to state-of-the-art; in the latter task they significantly outperform earlier methods (Habernal and Gurevych, 2016b).

Finally, to the best of our knowledge, current approaches do not deal with argument pairs of relatively *similar* quality. A natural extension is to develop a ternary-class classification model that will be trained and evaluated on such pairs, as we intend to explore in future work.

## Acknowledgements

We thank Tel Aviv University Debating Society, Ben Gurion University Debating Society, Yale Debate Association, HWS Debate Team, Seawolf Debate Program of the University of Alaska, and many other individual debaters.

## References


Nora Aranberri, Gorka Labaka, Arantza Díaz de Ilarraza, and Kepa Sarasola. 2017. Ebaluatoia: crowd evaluation for english–basque machine translation. *Language Resources and Evaluation*, 51(4):1053–1084.

Aristotle, G.A. Kennedy, and G.A. Kennedy. 1991. *On Rhetoric: A Theory of Civic Discourse*. Oxford University Press.

Roy Bar-Haim, Indrajit Bhattacharya, Francesco Dinuzzo, Amrita Saha, and Noam Slonim. 2017. Stance classification of context-dependent claims. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 251–261, Valencia, Spain. Association for Computational Linguistics.

T Bench-Capon, K Atkinson, and Peter McBurney. 2009. *Altruism and agents: an argumentation based approach to designing agent decision mechanisms*, pages 1073 – 1080. Unknown Publisher.

Xi Chen, Paul N. Bennett, Kevyn Collins-Thompson, and Eric Horvitz. 2013. Pairwise ranking aggregation in a crowdsourced setting. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, WSDM '13, pages 193–202, New York, NY, USA. ACM.

Cohen. 1960. A coefficient of agreement for nominal scales. *Educ Psychol Meas*, pages 37–46.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Martin Gleize, Eyal Shnarch, Leshem Choshen, Lena Dankin, Guy Moshkowich, Ranit Aharonov, and Noam Slonim. 2019. Are you convinced? choosing the more convincing evidence with a Siamese network. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 967–976, Florence, Italy. Association for Computational Linguistics.

Ivan Habernal and Iryna Gurevych. 2016a. What makes a convincing argument? empirical analysis and detecting attributes of convincingness in web argumentation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1214–1223, Austin, Texas. Association for Computational Linguistics.

Ivan Habernal and Iryna Gurevych. 2016b. Which argument is more convincing? analyzing and predicting convincingness of web arguments using bidirectional lstm. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1589–1599. Association for Computational Linguistics.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. Learning whom to trust with MACE. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1120–1130, Atlanta, Georgia. Association for Computational Linguistics.

Ran Levy, Yonatan Bilu, Daniel Hershcovich, Ehud Aharoni, and Noam Slonim. 2014. Context dependent claim detection. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1489–1500, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Marco Lippi and Paolo Torroni. 2016. Argumentation mining: State of the art and emerging trends. *ACM Trans. Internet Technol.*, 16(2):10:1–10:25.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *In EMNLP*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *CoRR*, abs/1806.03822.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392. Association for Computational Linguistics.

Chris Reed. 2016. Proceedings of the third workshop on argument mining (argmining2016). In *Proceedings of the Third Workshop on Argument Mining (ArgMining2016)*. Association for Computational Linguistics.

Nils Reimers, Benjamin Schiller, Tilman Beck, Johannes Daxenberger, Christian Stab, and Iryna Gurevych. 2019. Classification and clustering of arguments with contextualized word embeddings. *CoRR*, abs/1906.09821.

Ruty Rinott, Lena Dankin, Carlos Alzate Perez, Mitesh M. Khapra, Ehud Aharoni, and Noam Slonim. 2015. Show me your evidence - an automatic method for context dependent evidence detection. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 440–450, Lisbon, Portugal. Association for Computational Linguistics.

Burr Settles. 2009. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison.

Edwin D Simpson and Iryna Gurevych. 2018. Finding convincing arguments using scalable bayesian preference learning. *Transactions of the Association for Computational Linguistics*, 6:357–371.

Christian Stab and Iryna Gurevych. 2014. Annotating argument components and relations in persuasive essays. In *COLING*, pages 1501–1510. ACL.

Henning Wachsmuth, Nona Naderi, Yufang Hou, Yonatan Bilu, Vinodkumar Prabhakaran, Tim Alberdingk Thijm, Graeme Hirst, and Benno Stein. 2017a. Computational argumentation quality assessment in natural language. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 176–187. Association for Computational Linguistics.

Henning Wachsmuth, Martin Potthast, Khalid Al Khatib, Yamen Ajjour, Jana Puschmann, Jiani Qu, Jonas Dorsch, Viorel Morari, Janek Bevendorff, and Benno Stein. 2017b. Building an argument search engine for the web. In *ArgMining@EMNLP*, pages 49–59. Association for Computational Linguistics.

Douglas Walton, Chris Reed, and Fabrizio Macagno. 2008. *Argumentation Schemes*. Cambridge University Press.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *CoRR*, abs/1804.07461.

# Fine-Grained Analysis of Propaganda in News Articles

**Giovanni Da San Martino**[1] **Seunghak Yu**[2] **Alberto Barrón-Cedeño**[3]
**Rostislav Petrov**[4] **Preslav Nakov**[1]
[1] Qatar Computing Research Institute, HBKU, Qatar
[2] MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA, USA
[3] Università di Bologna, Forlì, Italy, [4] A Data Pro, Sofia, Bulgaria
{gmartino, pnakov}@hbku.edu.qa
seunghak@csail.mit.edu, a.barron@unibo.it
rostislav.petrov@adata.pro

## Abstract

Propaganda aims at influencing people's mindset with the purpose of advancing a specific agenda. Previous work has addressed propaganda detection at the document level, typically labelling *all* articles from a propagandistic news outlet as propaganda. Such noisy gold labels inevitably affect the quality of any learning system trained on them. A further issue with most existing systems is the lack of explainability. To overcome these limitations, we propose a novel task: performing fine-grained analysis of texts by detecting all fragments that contain propaganda techniques as well as their type. In particular, we create a corpus of news articles manually annotated at the fragment level with eighteen propaganda techniques and we propose a suitable evaluation measure. We further design a novel multi-granularity neural network, and we show that it outperforms several strong BERT-based baselines.

## 1 Introduction

Research on detecting propaganda has focused primarily on articles (Barrón-Cedeno et al., 2019; Rashkin et al., 2017). In many cases, there are no labeled data for individual articles, but there are such labels for entire news outlets. Thus, often all articles from the same news outlet get labeled the way that this outlet is labeled. Yet, it has been observed that propagandistic sources could post objective non-propagandistic articles periodically to increase their credibility (Horne et al., 2018). Similarly, media generally recognized as objective might occasionally post articles that promote a particular editorial agenda and are thus propagandistic. Thus, it is clear that transferring the label of the news outlet to each of its articles, could introduce noise. Such labels can still be useful for training robust systems, but they cannot be used to get a fair assessment of a system at testing time.

One option to deal with the lack of labels for articles is to crowdsource the annotation. However, in preliminary experiments we observed that the average annotator cannot detach her personal mindset from the judgment of propaganda and bias, i.e., if a clearly propagandistic text expresses ideas aligned with the annotator's beliefs, it is unlikely that she would judge it as such.

We argue that in order to study propaganda in a sound and reliable way, we need to rely on high-quality trusted professional annotations and it is best to do so at the fragment level, targeting specific techniques rather than using a label for an entire document or an entire news outlet.

Ours is the first work that goes at a fine-grained level: identifying specific instances of propaganda techniques used within an article. In particular, we create a corresponding corpus. For this purpose, we asked six experts to annotate articles from news outlets recognized as propagandistic and non-propagandistic, marking specific text spans with eighteen propaganda techniques. We also designed appropriate evaluation measures. Taken together, the annotated corpus and the evaluation measures represent the first manually-curated evaluation framework for the analysis of fine-grained propaganda. We release the corpus (350K tokens) as well as our code in order to enable future research.[1] Our contributions are as follows:

- We formulate a new problem: detect the use of specific propaganda techniques in text.
- We build a new large corpus for this problem.
- We propose a suitable evaluation measure.
- We design a novel multi-granularity neural network, and we show that it outperforms several strong BERT-based baselines.

[1]The corpus, the evaluation measures, and the models are available at http://propaganda.qcri.org/



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5636–5646,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

Our corpus could enable research in propagandistic and non-objective news, including the development of explainable AI systems. A system that can detect instances of use of specific propagandistic techniques would be able to make it explicit to the users why a given article was predicted to be propagandistic. It could also help train the users to spot the use of such techniques in the news.

The remainder of this paper is organized as follows: Section 2 presents the propagandistic techniques we focus on. Section 3 describes our corpus. Section 4 discusses an evaluation measures for comparing labeled fragments. Section 5 presents the formulation of the task and our proposed models. Section 6 describes our experiments and the evaluation results. Section 7 presents some relevant related work. Finally, Section 8 concludes and discusses future work.

## 2 Propaganda and its Techniques

Propaganda comes in many forms, but it can be recognized by its persuasive function, sizable target audience, the representation of a specific group's agenda, and the use of faulty reasoning and/or emotional appeals (Miller, 1939). Since propaganda is conveyed through the use of a number of techniques, their detection allows for a deeper analysis at the paragraph and the sentence level that goes beyond a single document-level judgment on whether a text is propagandistic.

Whereas the definition of propaganda is widely accepted in the literature, the set of propaganda techniques differs between scholars (Torok, 2015). For instance, Miller (1939) considers seven techniques, whereas Weston (2018) lists at least 24, and Wikipedia discusses 69.[2] The differences are mainly due to some authors ignoring some techniques, or using definitions that subsume the definition used by other authors. Below, we describe the propaganda techniques we consider: a curated list of eighteen items derived from the aforementioned studies. The list only includes techniques that can be found in journalistic articles and can be judged intrinsically, without the need to retrieve supporting information from external resources. For example, we do not include techniques such as *card stacking* (Jowett and O'Donnell, 2012, page 237), since it would require comparing against external sources of information.

[2] http://en.wikipedia.org/wiki/Propaganda_techniques; last visit May 2019.

The eighteen techniques we consider are as follows (cf. Table 1 for examples):

**1. Loaded language.** Using words/phrases with strong emotional implications (positive or negative) to influence an audience (Weston, 2018, p. 6). *Ex.:* "[. . . ] a lone lawmaker's childish shouting."

**2. Name calling or labeling.** Labeling the object of the propaganda campaign as either something the target audience fears, hates, finds undesirable or otherwise loves or praises (Miller, 1939). *Ex.:* "Republican congressweasels", "Bush the Lesser."

**3. Repetition.** Repeating the same message over and over again, so that the audience will eventually accept it (Torok, 2015; Miller, 1939).

**4. Exaggeration or minimization.** Either representing something in an excessive manner: making things larger, better, worse (e.g., "the best of the best", "quality guaranteed") or making something seem less important or smaller than it actually is (Jowett and O'Donnell, 2012, p. 303), e.g., saying that an insult was just a joke. *Ex.:* "Democrats bolted as soon as Trump's speech ended in an apparent effort to signal they can't even stomach being in the same room as the president"; "I was not fighting with her; we were just playing."

**5. Doubt.** Questioning the credibility of someone or something. *Ex.:* A candidate says about his opponent: "Is he ready to be the Mayor?"

**6. Appeal to fear/prejudice.** Seeking to build support for an idea by instilling anxiety and/or panic in the population towards an alternative, possibly based on preconceived judgments. *Ex.:* "stop those refugees; they are terrorists."

**7. Flag-waving.** Playing on strong national feeling (or with respect to a group, e.g., race, gender, political preference) to justify or promote an action or idea (Hobbs and Mcgee, 2008). *Ex.:* "entering this war will make us have a better future in our country."

**8. Causal oversimplification.** Assuming one cause when there are multiple causes behind an issue. We include *scapegoating* as well: the transfer of the blame to one person or group of people without investigating the complexities of an issue. *Ex.:* "If France had not declared war on Germany, World War II would have never happened."

| Doc ID | Technique • Snippet |
|---|---|
| 783702663 | `loaded language` • until forced to act by **a worldwide storm of outrage**. |
| 732708002 | `name calling, labeling` • dismissing the protesters as "**lefties**" and hugging Barros publicly |
| 701225819 | `repetition` • Farrakhan repeatedly refers to Jews as "**Satan**." He states to his audience [. . . ] call them by their real name, '**Satan**.' |
| 782086447 | `exaggeration, minimization` • heal the situation of **extremely grave** immoral behavior |
| 761969038 | `doubt` • **Can the same be said for the Obama Administration**? |
| 696694316 | `appeal to fear/prejudice` • **A dark, impenetrable and "irreversible" winter of persecution of the faithful by their own shepherds will fall**. |
| 776368676 | `flag-waving` • conflicted, and **his 17 Angry Democrats that are doing his dirty work are a disgrace to USA**! —Donald J. Trump |
| 776368676 | `flag-waving` • attempt (Mueller) **to stop the will of We the People**!!! It's time to jail Mueller |
| 735815173 | `causal oversimplification` • he said **The people who talk about the "Jewish question" are generally anti-Semites**. Somehow I don't think |
| 781768042 | `causal oversimplification` • will not be reversed, **which leaves no alternative as to why God judges and is judging America today** |
| 111111113 | `slogans` • **BUILD THE WALL!"** Trump tweeted. |
| 783702663 | `appeal to authority` • **Monsignor Jean-François Lantheaume, who served as first Counsellor of the Nunciature in Washington, confirmed that "Viganò said the truth. That's all"** |
| 783702663 | `black-and-white fallacy` • Francis said these words: "**Everyone is guilty for the good he could have done and did not do . . . If we do not oppose evil, we tacitly feed it**. |
| 729410793 | `thought-terminating cliches` • **I do not really see any problems there.** Marx is the President |
| 770156173 | `whataboutism` • President Trump —**who himself avoided national military service** in the 1960's— keeps beating the war drums over North Korea |
| 778139122 | `reductio ad hitlerum` • "Vichy journalism," a term which now fits so much of the mainstream media. **It collaborates in the same way that the Vichy government in France collaborated with the Nazis.** |
| 778139122 | `red herring` • It describes the tsunami of vindictive personal abuse that has been heaped upon Julian from well-known journalists, many claiming liberal credentials. The Guardian, **which used to consider itself the most enlightened newspaper in the country**, has probably been the worst. |
| 698018235 | `bandwagon` • He tweeted, "**EU no longer considers #Hamas a terrorist group. Time for US to do same."** |
| 729410793 | obfusc., int. vagueness, confusion • **The cardinal's office maintains that rather than saying "yes," there is a possibility of liturgical "blessing" of gay unions, he answered the question in a more subtle way without giving an explicit "yes."** |
| 783702663 | `straw man` • "Take it seriously, but with a large grain of salt." **Which is just Allen's more nuanced way of saying: "Don't believe it**." |

Table 1: Instances of the different propaganda techniques from our corpus. We show the document ID, the technique, and the text snippet, in bold. When necessary, some context is provided to better understand the example.

**9. Slogans.** A brief and striking phrase that may include labeling and stereotyping. Slogans tend to act as emotional appeals (Dan, 2015). *Ex.:* "Make America great again!"

**10. Appeal to authority.** Stating that a claim is true simply because a valid authority/expert on the issue supports it, without any other supporting evidence (Goodwin, 2011). We include the special case where the reference is not an authority/expert, although it is referred to as *testimonial* in the literature (Jowett and O'Donnell, 2012, p. 237).

**11. Black-and-white fallacy, dictatorship.** Presenting two alternative options as the only possibilities, when in fact more possibilities exist (Torok, 2015). As an extreme case, telling the audience exactly what actions to take, eliminating any other possible choice (*dictatorship*). *Ex.:* "You must be a Republican or Democrat; you are not a Democrat. Therefore, you must be a Republican"; "There is no alternative to war."

**12. Thought-terminating *cliché*.** Words or phrases that discourage critical thought and meaningful discussion about a given topic. They are typically short, generic sentences that offer seemingly simple answers to complex questions or that distract attention away from other lines of thought (Hunter, 2015, p. 78). *Ex.:* "it is what it is"; "you cannot judge it without experiencing it"; "it's common sense", "nothing is permanent except change", "better late than never"; "mind your own business"; "nobody's perfect"; "it doesn't matter"; "you can't change human nature."

**13. Whataboutism.** Discredit an opponent's position by charging them with hypocrisy without directly disproving their argument (Richter, 2017). For example, mentioning an event that discredits the opponent: "What about . . . ?" (Richter, 2017). *Ex.:* Russia Today had a proclivity for whataboutism in its coverage of the 2015 Baltimore and Ferguson protests in the US, which re-

vealed a consistent refrain: "the oppression of blacks in the US has become so unbearable that the eruption of violence was inevitable", and that the US therefore lacks "the moral high ground to discuss human rights issues in countries like Russia and China."

**14. Reductio ad Hitlerum.** Persuading an audience to disapprove an action or idea by suggesting that the idea is popular with groups hated in contempt by the target audience. It can refer to any person or concept with a negative connotation (Teninbaum, 2009). *Ex.:* "Only one kind of person can think this way: a communist."

**15. Red herring.** Introducing irrelevant material to the issue being discussed, so that everyone's attention is diverted away from the points made (Weston, 2018, p. 78). Those subjected to a red herring argument are led away from the issue that had been the focus of the discussion and urged to follow an observation or claim that may be associated with the original claim, but is not highly relevant to the issue in dispute (Teninbaum, 2009). Ex.: "You may claim that the death penalty is an ineffective deterrent against crime – but what about the victims of crime? How do you think surviving family members feel when they see the man who murdered their son kept in prison at their expense? Is it right that they should pay for their son's murderer to be fed and housed?"

**16. Bandwagon.** Attempting to persuade the target audience to join in and take the course of action because "everyone else is taking the same action" (Hobbs and Mcgee, 2008). Ex.: "Would you vote for Clinton as president? 57% say yes."

**17. Obfuscation, intentional vagueness, confusion.** Using deliberately unclear words, so that the audience may have its own interpretation (Suprabandari, 2007; Weston, 2018, p. 8). For instance, when an unclear phrase with multiple possible meanings is used within the argument, and, therefore, it does not really support the conclusion. Ex.: "It is a good idea to listen to victims of theft. Therefore, if the victims say to have the thief shot, then you should do it."

**18. Straw man.** When an opponent's proposition is substituted with a similar one which is then refuted in place of the original (Walton, 1996). Weston (2018, p. 78) specifies the characteristics of the substituted proposition: "caricaturing an opposing view so that it is easy to refute."

| | **Prop** | **Non-prop** | **All** |
|---|---|---|---|
| articles | 372 | 79 | 451 |
| avg length (lines) | 49.8 | 34.4 | 47.1 |
| avg length (words) | 973.2 | 635.4 | 914.0 |
| avg length (chars) | 5,942 | 3,916 | 5,587 |

Table 2: Statistics about the articles retrieved with respect to the category of the media source: **prop**agandistic, **non-prop**agandistic, and all together.

| **News Outlet** | **#** | **News Outlet** | **#** |
|---|---|---|---|
| Freedom Outpost | 133 | The Remnant Magazine | 14 |
| Frontpage Magazine | 56 | Breaking911 | 11 |
| shtfplan.com | 55 | truthuncensored.net | 8 |
| Lew Rockwell | 26 | The Washington Standard | 6 |
| vdare.com | 20 | www.unz.com | 5 |
| remnantnewspaper.com | 19 | www.clashdaily.com | 1 |
| Personal Liberty | 18 | | |

Table 3: Number of articles retrieved from news outlets deemed propagandistic by Media Bias/Fact Check.

We provided the above definitions, together with some examples and an annotation schema, to our professional annotators, so that they can manually annotate news articles. The details are provided in the next section.

## 3 Data Creation

We retrieved 451 news articles from 48 news outlets, both propagandistic and non-propagandistic, which we annotated as described below.

### 3.1 Article Retrieval

First, we selected 13 propagandistic and 36 non-propagandistic news media outlets, as labeled by Media Bias/Fact Check.[3] Then, we retrieved articles from these sources, as shown in Table 2. Note that 82.5% of the articles are from propagandistic sources, and these articles tend to be longer.

Table 3 shows the number of articles retrieved from each propagandistic outlet. Overall, we have 350k word tokens, which is comparable to standard datasets for other fine-grained text analysis tasks, such as named entity recognition, e.g., CoNLL'02 and CoNLL'03 covered 381K, 333K, 310K, and 301K tokens for Spanish, Dutch, German, and English, respectively (Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003).

[3] http://mediabiasfactcheck.com/

### 3.2 Manual Annotation

We aim at obtaining text fragments annotated with any of the 18 techniques described in Section 2 (see Figure 1 for an example). Since the time required to understand and memorize all the propaganda techniques is significant, this annotation task is not well-suited for crowdsourcing. We partnered instead with a company that performs professional annotations, A Data Pro.[4] Appendix A shows details about the instructions and the tools provided to the annotators.

We computed the $\gamma$ inter-annotator agreement (Mathet et al., 2015). We chose $\gamma$ because (*i*) it is designed for tasks where both the span and its label are to be found and (*ii*) it can deal with overlaps in the annotations by the same annotator[5] (e.g., instances of *doubt* often use *name calling* or *loaded language* to reinforce their message). We computed $\gamma_s$, where we only consider the identified spans, regardless of the technique, and $\gamma_{sl}$, where we consider both the spans and their labels.

Let $a$ be an annotator. In a preliminary exercise, four annotators $a_{[1,..,4]}$ annotated six articles independently, and the agreement was $\gamma_s = 0.34$ and $\gamma_{sl} = 0.31$. Even taking into account that $\gamma$ is a pessimistic measure (Mathet et al., 2015), these values are low. Thus, we designed an annotation schema composed of two stages and involving two annotator teams, each of which covered about 220 documents. In stage 1, both $a_1$ and $a_2$ annotated the same documents independently. In stage 2, they gathered with a consolidator $c_1$ to discuss all instances and to come up with a final annotation. Annotators $a_3$ and $a_4$ and consolidator $c_2$ followed the same procedure. Annotating the full corpus took 395 man hours.

Table 4 shows the $\gamma$ agreements on the full corpus. As in the preliminary annotation, the agreements for both teams are relatively low: 0.30 and 0.34 for span selection, and slightly lower when labeling is considered as well. After the annotators discussed with the consolidator on the disagreed cases, the $\gamma$ values got much higher: up to 0.74 and 0.76 for each team. We further analyzed the annotations to determine the main cause for the disagreement by computing the percentage of instances spotted by one annotator only in the first stage that are retained as gold annotations.

[4] http://www.aiidatapro.com

[5] See (Meyer et al., 2014; Mathet et al., 2015) for other alternatives, which lack some properties; (*ii*) in particular.

| Annotations | | spans ($\gamma_s$) | +labels ($\gamma_{sl}$) |
|---|---|---|---|
| $a_1$ | $a_2$ | 0.30 | 0.24 |
| $a_3$ | $a_4$ | 0.34 | 0.28 |
| $a_1$ | $c_1$ | 0.58 | 0.54 |
| $a_2$ | $c_1$ | 0.74 | 0.72 |
| $a_3$ | $c_2$ | 0.76 | 0.74 |
| $a_4$ | $c_2$ | 0.42 | 0.39 |

Table 4: $\gamma$ inter-annotator agreement between annotators spotting spans alone (**spans**) and spotting spans+labeling (**+labels**). The top-2 rows refer to the first stage: agreement between annotators. The bottom 4 rows refer to the consolidation stage: agreement between each annotator and the final gold annotation.

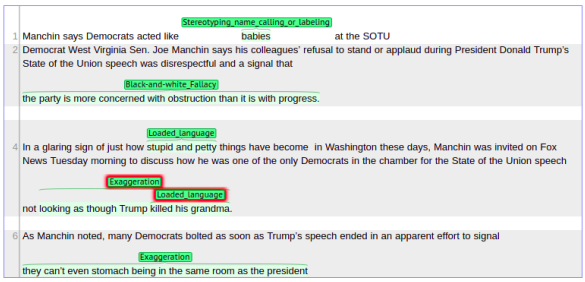


Figure 1: Example given to the annotators.

Overall the percentage is 53% (5,921 out of 11,122), and for each annotator is $a_1 = 70\%$, $a_2 = 48\%$, $a_3 = 57\%$, $a_4 = 31\%$. Observing such percentages together with the relatively low differences in Table 4 between $\gamma_s$ and $\gamma_{sl}$ for the same pairs $(a_i, a_j)$ and $(a_i, c_j)$, we can conclude that disagreements are in general not due to the two annotators assigning different labels to the same or mostly overlapping spans, but rather because one has missed an instance in the first stage.

### 3.3 Statistics about the Dataset

The total number of technique instances found in the articles, after the consolidation phase, is 7,485, with respect to a total number of 21,230 sentences (35.2%). Table 5 reports some statistics about the annotations. The average propagandistic fragment has a length of 47 characters and the average length of a sentence is 112.5 characters.

On average, the propagandistic techniques are half a sentence long. The most common ones are *loaded language* and *name calling, labeling* with 2,547 and 1,294 occurrences, respectively. They appear 6.7 and 4.7 times per article, while no other technique appears more than twice. Note that repetition are inflated as we asked the annotators to mark both the original and the repeated instances.

| Propaganda Technique | inst | avg. length |
|---|---|---|
| loaded language | 2,547 | $23.70 \pm 25.30$ |
| name calling, labeling | 1,294 | $26.10 \pm 19.88$ |
| repetition | 767 | $16.90 \pm 18.92$ |
| exaggeration, minimization | 571 | $45.36 \pm 35.55$ |
| doubt | 562 | $123.21 \pm 97.65$ |
| appeal to fear/prejudice | 367 | $93.56 \pm 74.59$ |
| flag-waving | 330 | $61.88 \pm 68.61$ |
| causal oversimplification | 233 | $121.03 \pm 71.66$ |
| slogans | 172 | $25.30 \pm 13.49$ |
| appeal to authority | 169 | $131.23 \pm 123.2$ |
| black-and-white fallacy | 134 | $98.42 \pm 73.66$ |
| thought-terminating cliches | 95 | $34.85 \pm 29.28$ |
| whataboutism | 76 | $120.93 \pm 69.62$ |
| reductio ad hitlerum | 66 | $94.58 \pm 64.16$ |
| red herring | 48 | $63.79 \pm 61.63$ |
| bandwagon | 17 | $100.29 \pm 97.05$ |
| obfusc., int. vagueness, confusion | 17 | $107.88 \pm 86.74$ |
| straw man | 15 | $79.13 \pm 50.72$ |
| **all** | 7,485 | $46.99 \pm 61.45$ |

Table 5: Corpus statistics including **inst**ances per technique and their **avg. length** in terms of characters.

## 4 Evaluation Measures

Our task is a sequence labeling one, with the following key characteristics: (*i*) a large number of techniques whose spans might overlap in the text, and (*ii*) large lengths of these spans. This requires an evaluation measure that gives credit for partial overlaps.[6] We derive an *ad hoc* measure following related work on named entity recognition (NER) (Nadeau and Sekine, 2007) and (intrinsic) plagiarism detection (PD) (Potthast et al., 2010).

While in NER, the relevant fragments tend to be short multi-word strings, in PD —and in our propaganda technique identification task— the length varies widely (cf. Table 5), and instances span from single tokens to full sentences or even longer pieces of text. Thus, in our precision and recall versions, we give partial credit to imperfect matches at the character level, as in PD.

Let document $\boldsymbol{d}$ be represented as a sequence of characters. A propagandistic text fragment is then represented as $t = [t_i, \ldots, t_j] \subseteq \boldsymbol{d}$. A document includes a set of (possibly overlapping) fragments $T$. Similarly, a learning algorithm produces a set $S$ with fragments $s = [s_m, \ldots, s_n]$, predicted on $\boldsymbol{d}$. A labeling function $l(x) \in \{1, \ldots, 18\}$ associates $s \in S$ to one of the eighteen techniques. Figure 2 gives examples of gold and predicted fragments.

[6]The evaluation measures for the CoNLL'02 and CoNLL'03 NER tasks, where an instance is considered properly identified if and only if both the boundaries and the label are correct (Tsai et al., 2006), are not suitable in our context.

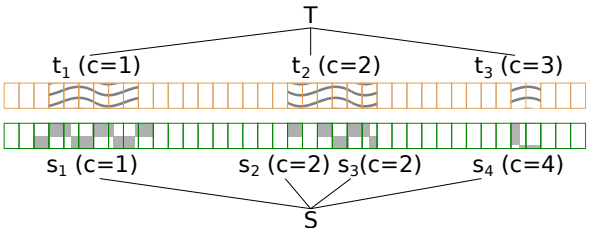


Figure 2: Example of gold annotation (top) and the predictions of a supervised model (bottom) in a document represented as a sequence of characters. The class of each fragment is shown in parentheses. $s_1$ goes beyond $t_1$'s proper boundaries; $s_2$ and $s_3$ partially spot $t_2$, but fail to identify it entirely; $s_4$ spots the exact boundaries of $t_3$, but fails to assign it the right label.

We define the following function to handle partial overlaps between fragments with same labels:

$$C(s, t, h) = \frac{|(s \cap t)|}{h} \delta\left(l(s), l(t)\right), \qquad (1)$$

where $h$ is a normalizing factor and $\delta(a, b) = 1$ if $a = b$, and 0 otherwise. In the future, $\delta$ could be refined to account for custom distance functions between classes, e.g., we might consider mistaking *loaded language* for *name calling or labeling* less problematic than confusing it with *Reduction ad Hitlerum*. Given Eq. (1), we now define variants of precision and recall able to account for the imbalance in the corpus:

$$P(S, T) = \frac{1}{|S|} \sum_{\substack{s \in S, \\ t \in T}} C(s, t, |s|), \qquad (2)$$

$$R(S, T) = \frac{1}{|T|} \sum_{\substack{s \in S, \\ t \in T}} C(s, t, |t|), \qquad (3)$$

We define Eq. (2) to be zero if $|S| = 0$ and Eq. (3) to be zero if $|T| = 0$. Following Potthast et al. (2010), in Eqs. (2) and (3) we penalize systems predicting too many or too few instances by dividing by $|S|$ and $|T|$, respectively, e.g., in Figure 2 $P(\{s_2, s_3\}, T) < P(\{s_3\}, T)$. Finally, we combine Eqs. (2) and (3) into an $F_1$-measure, the harmonic mean of precision and recall.

Having a separate function $C$ to be responsible for comparing two annotations gives us some additional flexibility that is missing in standard NER measures that operate at the token/character level. For example, in Eq. (1) we could easily change the factor that gives credit for partial overlaps by being more forgiving when only few characters are wrong.

## 5 Tasks and Proposed Models

We define two tasks based on the corpus described in Section 3: (*i*) **SLC (Sentence-level Classification)**, which asks to predict whether a sentence contains at least one propaganda technique, and (*ii*) **FLC (Fragment-level classification)**, which asks to identify both the spans and the type of propaganda technique. Note that these two tasks are of different granularities, $g_1$ and $g_2$, i.e., tokens for FLC and sentences for SLC. We split the corpus into training, development and test, each containing 293, 57, 101 articles and 14,857, 2,108, 4,265 sentences.

### 5.1 Baselines

We depart from BERT (Devlin et al., 2019), as it has achieved state-of-the-art performance on multiple NLP benchmarks, and we design three baselines based on it.

**BERT.** We add a linear layer on top of BERT and we fine-tune it, as suggested in (Devlin et al., 2019). For the FLC task, we feed the final hidden representation for each token to a layer $L_{g_2}$ that makes a 19-way classification: does this token belong to one of the eighteen propaganda techniques or to none of them (cf. Figure 3-a). For the SLC task, we feed the final hidden representation for the special `[CLS]` token, which BERT uses to represent the full sentence, to a two-dimensional layer $L_{g_1}$ to make a binary classification.

**BERT-Joint.** We use the layers for both tasks in the BERT baseline, $L_{g_1}$ and $L_{g_2}$, and we train for both FLC and SLC jointly (cf. Figure 3-b).

**BERT-Granularity.** We modify BERT-Joint to transfer information from SLC directly to FLC. Instead of using only the $L_{g_2}$ layer for FLC, we concatenate $L_{g_1}$ and $L_{g_2}$, and we add an extra 19-dimensional classification layer $L_{g_{1,2}}$ on top of that concatenation to perform the prediction for FLC (cf. Figure 3-c).

### 5.2 Multi-Granularity Network

We propose a model that can drive the higher-granularity task (FLC) on the basis of the lower-granularity information (SLC), rather than simply using low-granularity information directly. Figure 3-d shows the architecture of this model. More generally, suppose there are $k$ tasks of increasing granularity, e.g., document-level, paragraph-level, sentence-level, word-level, subword-level, character-level.

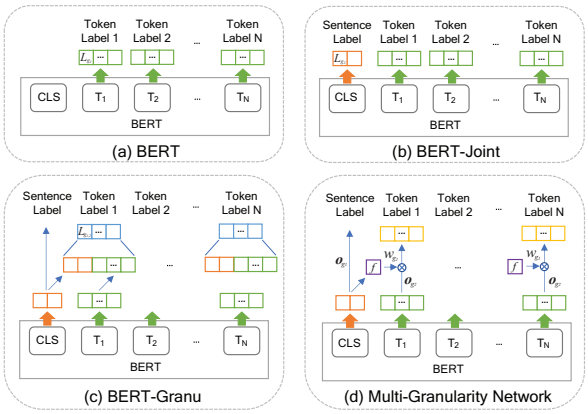


Figure 3: The architecture of the baseline models (a-c), and of our proposed multi-granularity network (d).

Each task has a separated classification layer $L_{g_k}$ that receives the feature representation of the specific level of granularity $g_k$ and outputs $\boldsymbol{o}_{g_k}$. The dimension of the representation depends on the embedding layer, while the dimension of the output depends on the number of classes in the task. The output $\boldsymbol{o}_{g_k}$ generates a weight for the next granularity task $g_{k+1}$ through a trainable gate $f$:

$$w_{g_k} = f(\boldsymbol{o}_{g_k}) \tag{4}$$

The gate $f$ consists of a projection layer to one dimension and an activation function. The resulting weight is multiplied by each element of the output of layer $L_{g_{k+1}}$ to produce the output for task $g_{k+1}$:

$$\boldsymbol{o}_{g_{k+1}} = w_{g_k} * \boldsymbol{o}_{g_{k+1}} \tag{5}$$

If $w_{g_k} = 0$ for a given example, the output of the next granularity task $\boldsymbol{o}_{g_{k+1}}$ would be 0 as well. In our setting this means that, if the sentence-level classifier is confident the sentence does not contain propaganda, i.e., $w_{g_k} = 0$, then $\boldsymbol{o}_{g_{k+1}} = 0$ and there would be no propagandistic technique predicted for any span within that sentence. Similarly, when back-propagating the error, if $w_{g_k} = 0$ for a given example, the final entropy loss would become zero; i.e. the model would not get any information from that example. As a result, only examples strongly classified as negative in a lower-granularity task would be ignored in the high-granularity task. Having the lower-granularity as the main task means that higher-granularity information can be selectively used as additional information to improve the performance, but only if the example is not considered as highly negative. We show this in Section 6.3.

For the loss function, we use a cross-entropy loss with sigmoid activation for every layer, except for the highest-granularity layer $L_{g_K}$, which uses a cross-entropy loss with softmax activation. Unlike softmax, which normalizes over all dimensions, the sigmoid allows each output component of layer $L_{g_k}$ to be independent from the rest. Thus, the output of the sigmoid for the positive class increases the degree of freedom by not affecting the negative class, and vice versa. As we have two tasks, we use sigmoid activation for $L_{g_1}$ and softmax activation for $L_{g_2}$. Moreover, we use a weighted sum of losses with a hyper-parameter $\alpha$:

$$\mathcal{L}_{\mathcal{J}} = \mathcal{L}_{g_1} * \alpha + \mathcal{L}_{g_2} * (1 - \alpha) \quad (6)$$

Again, we use BERT (Devlin et al., 2019) for the contextualized embedding layer and we place the multi-granularity network on top of it.

## 6 Experiments and Evaluation

### 6.1 Experimental Setup

We used the PyTorch framework and the pre-trained BERT model, which we fine-tuned for our tasks. We trained all models using the following hyper-parameters: batch size of 16, sequence length of 210, weight decay of 0.01, and early stopping on validation $F_1$ with patience of 7. For optimization, we used Adam with a learning rate of 3e-5 and a warmup proportion of 0.1. To deal with class imbalance, we give weight to the binary cross-entropy according to the proportion of positive samples. For the $\alpha$ in the joint loss function, we use 0.9 for sentence classification, and 0.1 for word-level classification. In order to reduce the effect of random fluctuations for BERT, all the reported numbers are the average of three experimental runs with different random seeds. As it is standard, we tune our models on the dev partition and we report results on the test partition.

### 6.2 Fragment-Level Propaganda Detection

Table 6 shows the performance for the three baselines and for our multi-granularity network on the FLC task. For the latter, we vary the degree to which the gate function is applied: using ReLU is more aggressive compared to using the Sigmoid, as the ReLU outputs zero for a negative input. Note that, even though we train the model to predict both the spans and the labels, we also evaluated it with respect to the spans only.

| Model | Spans | | | Full Task | | |
|---|---|---|---|---|---|---|
| | P | R | $F_1$ | P | R | $F_1$ |
| BERT | 39.57 | 36.42 | 37.90 | 21.48 | **21.39** | 21.39 |
| Joint | 39.26 | 35.48 | 37.25 | 20.11 | 19.74 | 19.92 |
| Granu | 43.08 | 33.98 | 37.93 | 23.85 | 20.14 | 21.80 |
| Multi-Granularity | | | | | | |
| ReLU | 43.29 | 34.74 | 38.28 | 23.98 | 20.33 | 21.82 |
| Sigmoid | **44.12** | **35.01** | **38.98** | **24.42** | 21.05 | **22.58** |

Table 6: Fragment-level experiments (FLC task). Shown are two evaluations: (*i*) **Spans** checks only whether the model has identified the fragment spans correctly, while (*ii*) **Full task** is evaluation wrt the actual task of identifying the spans and also assigning the correct propaganda technique for each span.

Table 6 shows that joint learning (BERT-Joint) hurts the performance compared to single-task BERT. However, using additional information from the sentence-level for the token-level classification (BERT-Granularity) yields small improvements. The multi-granularity models outperform all baselines thanks to their higher precision. This shows the effect of the model excluding sentences that it determined to be non-propagandistic from being considered for token-level classification. Nevertheless, the performance of sentence-level classification is far from perfect, achieving an $F_1$ of up to 60.98 (cf. Table 7). The information it contributes to the final classification is noisy and the more conservative removal of instances performed by the Sigmoid function yields better performance than the more aggressive ReLU.

### 6.3 Sentence-Level Propaganda Detection

Table 7 shows the results for the SLC task. We apply our multi-granularity network model to the sentence-level classification task to see its effect on low granularity when we train the model with a high granularity task. Interestingly, it yields huge performance improvements on the sentence-level classification result. Compared to the BERT baseline, it increases the recall by 8.42%, resulting in a 3.24% increase of the $F_1$ score. In this case, the result of token-level classification is used as additional information for the sentence-level task, and it helps to find more positive samples. This shows the opposite effect of our model compared to the FLC task. Note also that using ReLU is more effective than using the Sigmoid, unlike in token-level classification.

| Model | Precision | Recall | F1 |
|---|---|---|---|
| All-Propaganda | 23.92 | 100.00 | 38.61 |
| BERT | **63.20** | 53.16 | 57.74 |
| BERT-Granu | 62.80 | 55.24 | 58.76 |
| BERT-Joint | 62.84 | 55.46 | 58.91 |
| MGN Sigmoid | 62.27 | 59.56 | 60.71 |
| MGN ReLU | 60.41 | **61.58** | **60.98** |

Table 7: Sentence-level (SLC) results. *All-propaganda* is a baseline which always output the propaganda class.

Thus, since the performance range of the token-level classification is low, we think it is more effective to get additional information after aggressively removing negative samples by using ReLU as a gate in the model.

## 7 Related Work

Propaganda identification has been tackled mostly at the article level. Rashkin et al. (2017) created a corpus of news articles labelled as belonging to four categories: propaganda, trusted, hoax, or satire. They included articles from eight sources, two of which are propagandistic. Barrón-Cedeño et al. (2019) experimented with a binarized version of the corpus from (Rashkin et al., 2017): propaganda vs. the other three categories. The corpus labels were obtained with distant supervision, assuming that all articles from a given news outlet share the label of that outlet, which inevitably introduces noise (Horne et al., 2018).

A related field is that of computational argumentation which, among others, deals with some logical fallacies related to propaganda. Habernal et al. (2018b) presented a corpus of Web forum discussions with cases of *ad hominem* fallacy identified. Habernal et al. (2017, 2018a) introduced *Argotario*, a game to educate people to recognize and create fallacies. A byproduct of Argotario is a corpus with $1.3k$ arguments annotated with five fallacies, including *ad hominem*, *red herring* and *irrelevant authority*, which directly relate to propaganda techniques (cf. Section 2). Differently from (Habernal et al., 2017, 2018a,b), our corpus has 18 techniques annotated on the same set of news articles. Moreover, our annotations aim at identifying the minimal fragments related to a technique instead of flagging entire arguments.

## 8 Conclusion and Future Work

We have argued for a new way to study propaganda in news media: by focusing on identifying the instances of use of specific propaganda techniques. Going at this fine-grained level can yield more reliable systems and it also makes it possible to explain to the user why an article was judged as propagandistic by an automatic system.

In particular, we designed an annotation schema of 18 propaganda techniques, and we annotated a sizable dataset of documents with instances of these techniques in use. We further designed an evaluation measure specifically tailored for this task. We made the schema and the dataset publicly available, thus facilitating further research. We hope that the corpus would raise interest outside of the community of researchers studying propaganda: the techniques related to fallacies and the ones relying on emotions might provide a novel setting for the researchers interested in Argumentation and Sentiment Analysis.

We experimented with a number of BERT-based models and devised a novel architecture which outperforms standard BERT-based baselines. Our fine-grained task can complement document-level judgments, both to come out with an aggregated decision and to explain why a document —or an entire news outlet— has been flagged as potentially propagandistic by an automatic system.

We are collaborating with A Data Pro to expand the corpus. In the mid-term, we plan to build an online platform where professors in relevant fields (e.g., journalism, mass communication) can train their students to recognize and annotate propaganda techniques. The hope is to be able to accumulate annotations as a by-product of using the platform for training purposes.

## Acknowledgments

This research is part of the Tanbih project,[7] which aims to limit the effect of "fake news", propaganda and media bias by making users aware of what they are reading. The project is developed in collaboration between the MIT Computer Science and Artificial Intelligence Laboratory (CSAIL) and the Qatar Computing Research Institute (QCRI), HBKU.

[7] http://tanbih.qcri.org/

## References


Alberto Barrón-Cedeño, Giovanni Da San Martino, Israa Jaradat, and Preslav Nakov. 2019. Proppy: A system to unmask propaganda in online news. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, AAAI '19, pages 9847–9848, Honolulu, HI, USA.

Alberto Barrón-Cedeno, Israa Jaradat, Giovanni Da San Martino, and Preslav Nakov. 2019. Proppy: Organizing the news based on their propagandistic content. *Information Processing & Management*, 56(5):1849–1864.

Lavinia Dan. 2015. Techniques for the Translation of Advertising Slogans. In *Proceedings of the International Conference Literature, Discourse and Multicultural Dialogue*, LDMD '15, pages 13–23, Mures, Romania.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT '19, pages 4171–4186, Minneapolis, MN, USA.

Jean Goodwin. 2011. Accounting for the force of the appeal to authority. In *Proceedings of the 9th International Conference of the Ontario Society for the Study of Argumentation*, OSSA '11, pages 1–9, Ontario, Canada.

Ivan Habernal, Raffael Hannemann, Christian Pollak, Christopher Klamm, Patrick Pauli, and Iryna Gurevych. 2017. Argotario: Computational argumentation meets serious games. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '17, pages 7–12, Copenhagen, Denmark.

Ivan Habernal, Patrick Pauli, and Iryna Gurevych. 2018a. Adapting serious game for fallacious argumentation to German: pitfalls, insights, and best practices. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation*, LREC '18, Miyazaki, Japan.

Ivan Habernal, Henning Wachsmuth, Iryna Gurevych, and Benno Stein. 2018b. Before name-calling: Dynamics and triggers of ad hominem fallacies in web argumentation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT '18, pages 386–396, New Orleans, LA, USA.

Renee Hobbs and Sandra Mcgee. 2008. Teaching about propaganda: An examination of the historical roots of media literacy. *Journal of Media Literacy Education*, 6(62):56–67.

Benjamin D Horne, Sara Khedr, and Sibel Adali. 2018. Sampling the news producers: A large news and feature data set for the study of the complex media landscape. In *International AAAI Conference on Web and Social Media*, ICWSM '18, Stanford, CA, USA.

John Hunter. 2015. Brainwashing in a large group awareness training? The classical conditioning hypothesis of brainwashing. Master's thesis, University of Kwazulu-Natal, Pietermaritzburg, South Africa.

Garth S. Jowett and Victoria O'Donnell. 2012. What is propaganda, and how does it differ from persuasion? In *Propaganda & Persuasion*, chapter 1, pages 1–48. Sage Publishing.

Yann Mathet, Antoine Widlöcher, and Jean-Philippe Métivier. 2015. The unified and holistic method gamma ($\gamma$) for inter-annotator agreement measure and alignment. *Computational Linguistics*, 41(3):437–479.

Christian M. Meyer, Margot Mieskes, Christian Stab, and Iryna Gurevych. 2014. DKPro agreement: An open-source Java library for measuring inter-rater agreement. In *Proceedings of the International Conference on Computational Linguistics*, COLING '14, pages 105–109, Dublin, Ireland.

Clyde R. Miller. 1939. The Techniques of Propaganda. From "How to Detect and Analyze Propaganda," an address given at Town Hall. The Center for learning.

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.

Martin Potthast, Benno Stein, Alberto Barrón-Cedeño, and Paolo Rosso. 2010. An evaluation framework for plagiarism detection. In *Proceedings of the International Conference on Computational Linguistics*, COLING '10, pages 997–1005, Beijing, China.

Hannah Rashkin, Eunsol Choi, Jin Yea Jang, Svitlana Volkova, and Yejin Choi. 2017. Truth of varying shades: Analyzing language in fake news and political fact-checking. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, EMNLP '17, pages 2931–2937, Copenhagen, Denmark.

Monika L Richter. 2017. The Kremlin's platform for 'useful idiots' in the West: An overview of RT's editorial strategy and evidence of impact. Technical report, Kremlin Watch.

Francisca Niken Vitri Suprabandari. 2007. American propaganda in John Steinbeck's The Moon is Down. Master's thesis, Sanata Dharma University, Yogyakarta, Indonesia.

Gabriel H Teninbaum. 2009. Reductio ad Hitlerum: Trumping the judicial Nazi card. *Michigan State Law Review*, page 541.

Erik F. Tjong Kim Sang. 2002. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *Proceedings of the 6th Conference on Natural Language Learning*, CoNLL '02, pages 155–158, Taipei, Taiwan.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the 7th Conference on Natural Language Learning*, CoNLL '03, pages 142–147, Edmonton, Canada.

Robyn Torok. 2015. Symbiotic radicalisation strategies: Propaganda tools and neuro linguistic programming. In *Proceedings of the Australian Security and Intelligence Conference*, pages 58–65, Perth, Australia.

Richard Tzong-Han Tsai, Shih-Hung Wu, Wen-Chi Chou, Yu-Chun Lin, Ding He, Jieh Hsiang, Ting-Yi Sung, and Wen-Lian Hsu. 2006. Various criteria in the evaluation of biomedical named entity recognition. *BMC bioinformatics*, 7:92.

Douglas Walton. 1996. *The straw man fallacy*. Royal Netherlands Academy of Arts and Sciences.

Anthony Weston. 2018. *A rulebook for arguments*. Hackett Publishing.

# Context-aware Interactive Attention for Multi-modal Sentiment and Emotion Analysis

**Dushyant Singh Chauhan, Md Shad Akhtar, Asif Ekbal and Pushpak Bhattacharyya**
Department of Computer Science & Engineering
Indian Institute of Technology Patna
Patna, Bihar, India-801106
{1821CS17, shad.pcs15, asif, pb}@iitp.ac.in

## Abstract

In recent times, multi-modal analysis has been an emerging and highly sought-after field at the intersection of natural language processing, computer vision, and speech processing. The prime objective of such studies is to leverage the diversified information, (e.g., textual, acoustic and visual), for learning a model. The effective interaction among these modalities often leads to a better system in terms of performance. In this paper, we introduce a recurrent neural network based approach for the multi-modal sentiment and emotion analysis. The proposed model learns the inter-modal interaction among the participating modalities through an auto-encoder mechanism. We employ a context-aware attention module to exploit the correspondence among the neighboring utterances. We evaluate our proposed approach for five standard multi-modal affect analysis datasets. Experimental results suggest the efficacy of the proposed model for both sentiment and emotion analysis over various existing state-of-the-art systems.

## 1 Introduction

In recent past, the world has witnessed tremendous growth of various social media platforms, e.g., YouTube, Instagram, Twitter, Facebook, etc. People treat these platforms as a communication medium and freely express themselves with the help of a diverse set of input sources, e.g. *videos*, *images*, *audio*, *text* etc. The amount of information produced daily through these mediums are enormous, and hence, the research on multi-modal information processing has attracted attention to the researchers and developers. A video is a multi-modal input which provides visual, acoustic, and textual information.

The motivation of multi-modal sentiment and emotion analysis lies in fact to leverage the varieties of (often distinct) information from multiple sources for building more efficient systems. For some cases, text can provide a better clue for the prediction, whereas for the others, acoustic or visual sources can be more informative. Similarly, in some situations, a combination of two or more information sources together ensures better and unambiguous classification decision. For example, only text *"shut up"* can not decide the mood of a person but *acoustic (tone of a person)* and *visual (expression of a person)* can reveal the exact mood. Similarly, for some instances visual features such as *gesture*, *postures*, *facial expression* etc. have important roles to play in determining the correctness of the system. However, effectively combining this information is a non-trivial task that researchers often have to face (Poria et al., 2016; Ranganathan et al., 2016; Lee et al., 2018).

Traditionally, '*text*' has been the key factor in any Natural Language Processing (NLP) tasks, including sentiment and emotion analysis. However, with the recent emergence of social media platforms, an interdisciplinary study involving *text*, *visual* and *acoustic* features have drawn a great interest among the research community. Expressing the feelings and emotions through a video is much convenient than the text for a user, and it is the best source to extract all multi-modal information. Not only the *visual*, it also provides other information such as *acoustic* and *textual* representation of *spoken language*. Additionally, a single video can have multiple utterances based on a speaker's pause (speech bounded by breaths) with different sentiments and emotions. The sentiments and emotions of an utterance often have inter-dependence on the other contextual utterances. Independently classifying such an utterance poses several challenges to the underlying problem. In contrast, multi-modal sentiment and emotion analysis take inputs from more than one sources e.g.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5647–5657,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

*text*, *visual*, *acoustic* for the analysis. Effectively fusing this diverse information is non-trivial and poses several challenges to the underlying problem.

In our current work, we propose an end-to-end Context-aware Interactive Attention (CIA) based recurrent neural network for sentiment and emotion analysis. We aim to leverage the interaction between the modalities to increase the confidence of individual task in prediction. The main contributions of our current research are as follows: **(1)** *We propose an Inter-modal Interactive Module (IIM) that aims to learn the interaction among the diverse and distinct features of the input modalities, i.e., text, acoustic and visual*; **(2)** *We employ a Context-aware Attention Module (CAM) that identifies and assigns the weights to the neighboring utterances based on their contributing features. It exploits the interactive representations of pairwise modalities to learn the attention weights*, and **(3)** *We present new state-of-the-arts for five benchmark datasets for both sentiment and emotion predictions*.

## 2 Related Work

Different reviews in (Arevalo et al., 2017; Poria et al., 2016, 2017b; Ghosal et al., 2018; Morency et al., 2011a; Zadeh et al., 2018a; Mihalcea, 2012; Lee et al., 2018; Tsai et al., 2018) suggest that multi-modal sentiment and emotion analysis are relatively new areas as compared to uni-modal analysis. Feature selection (fusion) is a challenging and important task for any multi-modal analysis. Poria et al. (2016) proposed a multi-kernel learning based feature selection method for multimodal sentiment and emotion recognition. A convolutional deep belief network (CDBN) is proposed in (Ranganathan et al., 2016) to learn salient multi-modal features of low-intensity expressions of emotions, whereas Lee et al. (2018) introduced a convolutional attention network to learn multimodal feature representation between speech and text data for multi-modal emotion recognition.

A feature level fusion vector was built, and then a Support Vector Machine (SVM) classifier was used to detect the emotional duality and mixed emotional experience in (Patwardhan, 2017). Similar work on feature-level fusion based on self- attention mechanism is reported in (Hazarika et al., 2018). Fu et al. (2017) introduced an enhanced sparse local discriminative canonical correlation analysis approach (En-SLDCCA) to learn the multi-modal shared feature representation. Tzirakis et al. (2017) introduced a Long Short Term Memory (LSTM) based end-to-end multi-modal emotion recognition system in which convolutional neural network (CNN) and a deep residual network are used to capture the emotional content for various styles of speaking, robust features.

Poria et al. (2017a) presented a literature survey on various affect dimensions e.g., sentiment analysis, emotion analysis, etc., for the multi-modal analysis. A multi-modal fusion-based approach is proposed in (Blanchard et al., 2018) for sentiment classification. The author used exclusively high-level fusion of visual and acoustic features to classify the sentiment. Zadeh et al. (2016) presented the multi-modal dictionary-based technique to capture the interaction between spoken words and facial expression better when expressing the sentiment. In another work, Zadeh et al. (2017) proposed a Tensor Fusion Network (TFN) to capture the inter-modality and intra-modality dynamics between the multi-modalities (i.e., text, visual, and acoustic).

These works did not take contextual information into account. Poria et al. (2017b) introduced an Long Short Term Memory (LSTM) based framework for sentiment classification which uses contextual information to capture interrelationships between the utterances. In another work, Poria et al. (2017c) proposed a user opinion based framework to combine all the multi-modal inputs (i.e., visual, acoustic, and textual) by applying a multi-kernel learning-based approach. Contextual inter-modal attention mechanism was not explored in much details until recently. Zadeh et al. (2018a) introduced a multi-attention blocks based model for multi-modal sentiment classification but did not account for contextual information, whereas Ghosal et al. (2018) proposed a contextual inter-modal attention based framework for multi-modal sentiment classification. Recently, Zadeh et al. (2018c) introduced the largest multimodal dataset namely *CMU-MOSEI* for sentiment and emotion analysis. Author effectively fused the multi-modality inputs *i.e., text, visual, and acoustic* through a dynamic fusion graph and reported competitive performance *w.r.t.* various state-of-the-art systems for both sentiment and emotion analysis. Very recently, Akhtar et al. (2019) in-

troduced an attention based multi-task learning framework for sentiment and emotion classification on the *CMU-MOSEI* dataset.

In comparison to the existing systems, our proposed approach aims to exploits the interaction between the input modalities through an auto-encoder based *inter-modal interactive module*. The interactive module learns the joint representation for the participating modalities, which are further utilized to capture the contributing contextual utterances in a *context-aware attention module*.

## 3 Context-aware Interactive Attention (CIA) Affect Analysis

In this section, we describe our proposed approach for the effective fusion of multi-modal input sources. We propose an end-to-end Context-aware Interactive Attention (CIA) based recurrent neural network for sentiment and emotion analysis. As discussed earlier, one of the main challenges for multi-modal information analysis is to exploit the interaction among the input modalities. Therefore, we introduce an Inter-modal Interactive Module (IIM) that aims to learn the interaction between any two modalities through an auto-encoder like structure. For the *text-acoustic* pair of modalities, we aim to decode the *acoustic* representation through the encoded *textual* representation. After training of IIM, we extract the encoded representation for further processing. We argue that the encoded representation learns the interaction between the *text* and *acoustic* modalities. Similarly, we compute the interaction among all the other pairs (i.e., *acoustic-text*, *text-visual*, *visual-text*, *acoustic-visual*, and *visual-acoustic*). Next, we extract the sequential pattern of the utterances through a Bi-directional Gated Recurrent Unit (Bi-GRU) (Cho et al., 2014)). For each pair of modalities, the two representations denoting the interactions between them are combined through a mean operation. For an instance, we compute the mean of the *text-acoustic* and *acoustic-text* representations for text and acoustic modalities. The mean operation ensures that the network utilizes the two distinct representations by keeping the minimal dimension.

In our network, we, additionally, learn the interaction among the modalities through a feed-forward network. At first, all the three modalities are passed through a separate Bi-GRU. Then, pair-wise concatenation is performed over the output of Bi-GRU and passed through a fully-connected layer to extract the bi-modal interaction (BI). Further, we employ a Context-aware Attention Module (CAM) to exploit the correspondence among the neighboring utterances. The inputs to the CAM are the two representations for each pair of modalities, e.g., mean representation $M_{TA}$ and bi-modal interaction $BI_{TA}$ for the text-acoustic

**Algorithm 1** Inter-modal Interactive Module for Multi-modal Sentiment and Emotion Recognition (IIM-MMSE)

**procedure** IIM-MMSE$(t, v, a)$
  **for** $i \in 1, ..., K$ **do** ▷ $K = \#modalities$
    **for** $j \in 1, ..., K$ **do**
      ▷$\forall x, y \in [T, V, A]$, $x \neq y$ and $i \leq j$
      $C_{x_i y_j} \leftarrow IIM(x_i, y_j)$
      $C_{x_i y_j} \leftarrow biGRU(C_{x_i y_j})$
      $C_{x_i} \leftarrow biGRU(x_i)$

  **for** $i, j \in 1, ..., K$ **do**
    ▷$\forall x, y \in [T, V, A]$, and $x \neq y$
    $M_{x_i,y_j} \leftarrow Mean(C_{x_i y_j}, C_{y_i x_j})$
    $cat_{x_i,y_j} \leftarrow Concatenate(C_{x_i}, C_{y_j})$
    $BI_{x_i,y_j} \leftarrow FullyConnected(cat_{x_i,y_j})$
    $A_{x_i,y_j} \leftarrow CAM(M_{x_i,y_j}, BI_{x_i,y_j})$
  $Rep \leftarrow [A_{TV}, A_{TA}, A_{AV}]$
  $polarity \leftarrow Sent(Rep)/Emo(Rep)$
  **return** $polarity$

**Algorithm 2** Inter-Modal Interactive Module (IIM)

**procedure** IIM$(X, Y)$
  $C_{XY} \leftarrow IIM_{Encoder}(X, Y)$
  $\widetilde{Y} \leftarrow IIM_{Decoder}(C_{XY})$
  $loss \leftarrow$ cross_entropy$(\widetilde{Y}, Y)$
  Backpropagation to update the weights
  **return** $C_{XY}$

**Algorithm 3** Context-aware Attention Module (CAM)

**procedure** CAM$(M, BI)$
  $P \leftarrow M.BI^T$ ▷ Cross product
  **for** $i, j \in 1, ..., u$ **do** ▷ $u =$ #utterances
    $N(i, j) \leftarrow \frac{e^{P(i,j)}}{\sum_{k=1}^{u} e^{P(i,k)}}$
  $O \leftarrow N.BI$
  $A \leftarrow O \odot M$ ▷ Multiplicative gating.
  **return** $A$

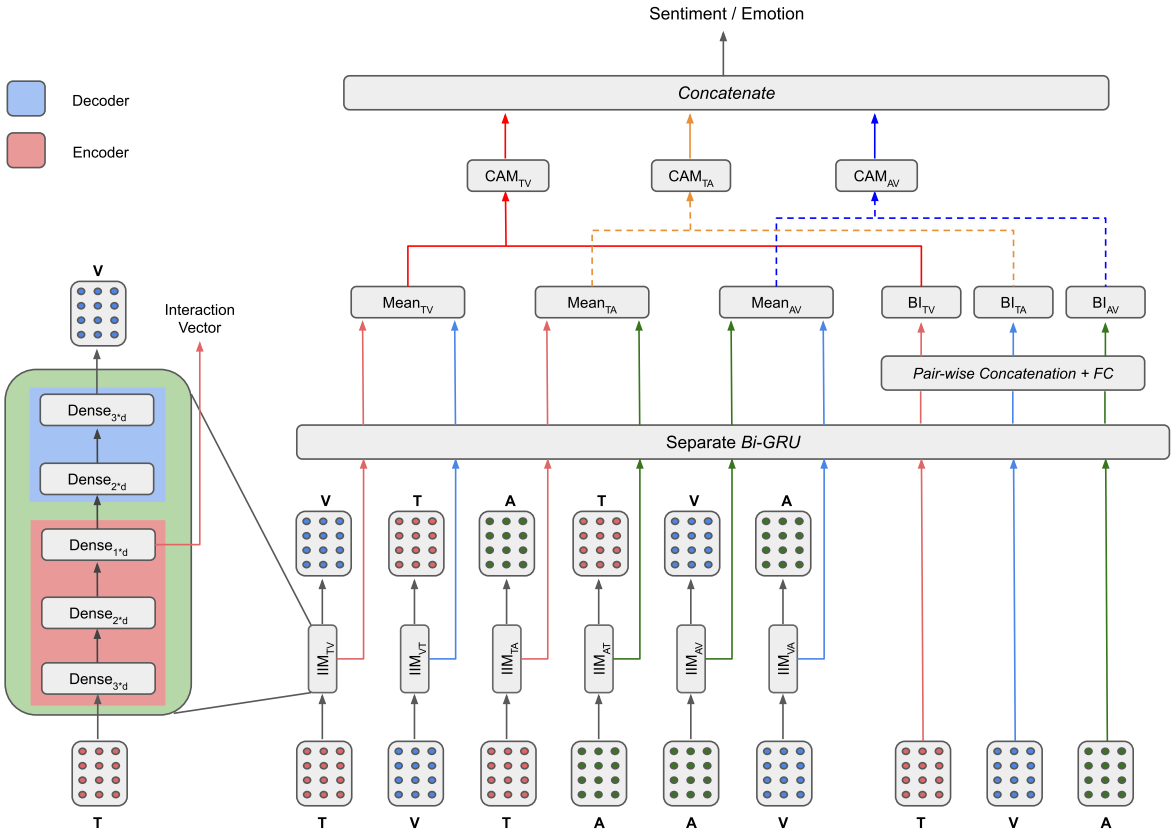


Figure 1: Overall architecture of the proposed Context-aware Interactive Attention framework.

pair. The attention module assists the network in attending the contributing features by putting weights to the current and the neighboring utterances in a video. In the end, the pair-wise (i.e., *text-acoustic*, *text-visual*, and *acoustic-visual*) attended representations are concatenated and fed to an output layer for the prediction.

We depict and summarize the proposed approach in Figure 1 and Algorithm 1, 2, and 3. The source code is available at `http://www.iitp.ac.in/~ai-nlp-ml/resources.html`.

### 3.1 Context-aware Attention Module (CAM)

Since the utterances in a video are the split units of the break/pause of the speech, their emotions (or sentiments) often have relations with their neighboring utterances. Therefore, knowledge of the emotions (or, sentiments) of the neighboring utterances is an important piece of information and has the capability to derive the prediction of an utterance, if the available inputs are insufficient for the correct prediction.

Our proposed context-aware attention module leverages the contextual information. For each utterance in a video, we compute the attention weights of all the neighboring utterances based on their contributions in predicting the current utterance. It ensures that the network properly utilizes the local contextual information of an utterance as well as the global contextual information of a video together. The aim is to compute the interactive attention weights utilizing a *softmax* activation for each utterance in the video. Next, we apply a multiplicative gating mechanism following the work of Dhingra et al. (2016). The attentive representation is, then, forwarded to the upper layers for further processing. We summarize the process of *CAM* in Algorithm3.

### 3.2 Inter-modal Interactive Module (IIM)

One of the key objectives of the multi-modal analysis is to fuse the available input modalities effectively. In general, different modalities represent distinct features despite serving a common goal. For example, in multi-modal sentiment analysis all the three modalities, i.e., *text*, *acoustic*, and *visual*, aim to predict the expressed polarity of an utterance. The distinctive features in isolation

might create an ambiguous scenario for a network to learn effectively. Therefore, we introduce an auto-encoder based inter-modal interactive module whose objective is to learn the interaction between two distinct modalities to serve a common goal. The IIM encodes the feature representation of one modality (say, text), and aims to decode it into the feature representation of another modality (say, acoustic). Similar to an auto-encoder where the input and output are conceptually the same (or closely related), in our case the input and output feature representations of two modalities also intuitively serve a common goal. After training of IIM, the encoded vector signifies a joint representation of the two modalities, which can be further utilized in the network.

As the proposed architecture in Figure 1 depicts, our proposed model is an *end-to-end* system, which takes multi-modal raw features for each utterance in a video and predicts an output. We also train our proposed IIM in the combined framework. For any pair of modalities, e.g., *text-visual*, the encoded vector in IIM receives two gradients of errors, i.e., one error from the IIM output (visual) $l_1$ and another from the task-specific label $l_2$. We aggregate the errors ($l_1 + l_2$) at the encoded vector and backpropagate it to the input (text). Thus, the weights in the encoder part will adjust according to the desired task-specific label as well. However, in contrast, the decoder part does not have such information. Therefore, we employ another IIM to capture the interaction between the *visual-text*. This time, the *visual* features are aware of the desired label during the interaction with *textual* features. A conceptual diagram, depicting the gradient flow in IIM for the text and visual modalities, is shown in Figure 2.

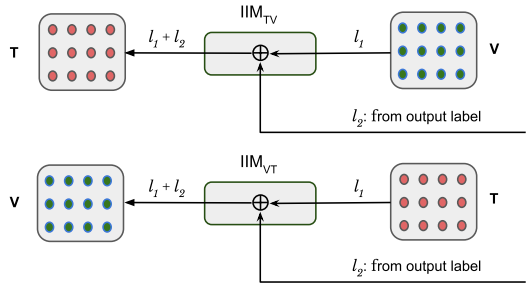


Figure 2: Difference between the interaction modules of *text-visual* and *visual-text* pairs. $l_1$ & $l_2$ are the losses from the IIM and output labels, respectively.

## 4 Datasets, Experiments and Analysis

In this section, we present our experimental results along with necessary analysis. We also compare our obtained results with several state-of-the-art systems.

### 4.1 Datasets

For the evaluation of our proposed approach, we employ five multi-modal benchmark datasets[1] covering two affect analysis tasks, i.e., sentiment and emotion.

1. **YouTube** (Morency et al., 2011b): The YouTube opinion dataset contains 269 product reviews utterances across 47 videos. There are 169, 41, and 59 utterances in training, validation, and test set, respectively.

2. **MOUD** (Pérez-Rosas et al., 2013): The dataset consists of 79 product review videos in Spanish. Each video consists of multiple utterances labeled with either positive, negative, or neutral sentiment. There are 243, 37, 106 utterances in training, validation, and test set, respectively.

3. **ICT-MMMO** (Wöllmer et al., 2013): It is an extension of YouTube opinion dataset that extends the number of videos from 47 to 340. Each online social review video is annotated for the sentiment. There are 220, 40, and 80 videos in training, validation, and test set, respectively.

4. **CMU-MOSI** (Zadeh et al., 2016): It is a collection of 2199 opinion utterances annotated with the sentiment class. There are 1284 utterances in the training set, 229 utterances in the validation set, and 686 utterances in the test set.

5. **CMU-MOSEI** (Zadeh et al., 2018c): CMU-MOSEI is the largest dataset among all the above. It has 3,229 videos which comprise of approximately 23,000 utterances. Each utterance is associated with one sentiment value and six emotions (i.e., *anger* (4903 utterances), *disgust* (4028 utterance), *fear* (1850 utterance), *happy* (12135 utterance),

[1]These datasets can be accessed through https://github.com/A2Zadeh/CMU-MultimodalSDK.

| | MOSEI | | | | | | | | MOSI | | | | | ICT-MMMO | | YouTube | | MOUD | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Emotion | | Sentiment | | | | | | Sentiment | | | | | Sentiment | | Sentiment | | Sentiment | |
| | *F1* | *W-Acc* | *F1* | $A^2$ | $A^5$ | $A^7$ | *MAE* | *r* | *F1* | $A^2$ | $A^7$ | *MAE* | *r* | *F1* | $A^2$ | *F1* | $A^3$ | *F1* | $A^2$ |
| **T** | 77.66 | 60.13 | 77.22 | 79.45 | 48.50 | 48.87 | 0.695 | 0.572 | 77.43 | 77.69 | 37.75 | 0.996 | 0.658 | 71.80 | 77.50 | 48.27 | 49.15 | 72.24 | 73.58 |
| **A** | 76.14 | 57.68 | 74.64 | 77.27 | 45.02 | 44.97 | 0.782 | 0.433 | 59.30 | 59.32 | 23.17 | 1.423 | 0.165 | 77.90 | 78.34 | 43.12 | 44.06 | 45.46 | 60.37 |
| **V** | 75.03 | 57.59 | 69.13 | 75.04 | 42.22 | 42.51 | 0.804 | 0.317 | 50.48 | 51.31 | 23.03 | 1.465 | 0.122 | 78.28 | 78.75 | 49.79 | 50.84 | 62.47 | 62.76 |
| **T+V** | 78.01 | 57.10 | 77.31 | 79.51 | 48.98 | 49.84 | 0.685 | 0.592 | 77.70 | 78.13 | 38.19 | 0.946 | 0.673 | 80.10 | 81.25 | 53.11 | 53.91 | 78.40 | 79.24 |
| **T+A** | 77.53 | 60.69 | 77.80 | 80.06 | 48.66 | 49.08 | 0.694 | 0.579 | 78.98 | 79.15 | 39.35 | 0.952 | 0.671 | 78.99 | 80.01 | 48.82 | 49.37 | 72.93 | 74.52 |
| **A+V** | 76.60 | 59.47 | 74.76 | 77.38 | 45.32 | 45.87 | 0.775 | 0.437 | 49.30 | 53.49 | 24.19 | 1.464 | 0.189 | 79.12 | 81.25 | 51.93 | 52.54 | 64.41 | 65.09 |
| **T+A+V** | **79.02** | **62.97** | **78.23** | **80.37** | **49.15** | **50.14** | **0.683** | **0.594** | **79.54** | **79.88** | **38.92** | **0.914** | **0.689** | **81.47** | **82.75** | **55.13** | **55.93** | **82.07** | **82.41** |

Table 1: Results of sentiment and emotion analysis for the proposed approach. **T:** Text, **V:** Visual, **A:** *Acoustic*. Weighted accuracy as a metric is chosen due to unbalanced samples across various emotions and it is also in line with the other existing works (Zadeh et al., 2018c).

| **Multi-label** | **No** | **One** | **Two** | **Three** | **Four** | **Five** | **Six** |
|---|---|---|---|---|---|---|---|
| **Count** | 3372 | 11050 | 5526 | 2084 | 553 | 84 | 8 |

Table 2: Statistics of multi-label emotions in CMU-MOSEI: *Emotions-per-utterance*.

*sad* (5856 utterance), and *surprise* (2262 utterance). The emotion class *happy* is approximately 52% of the total utterances, while emotions *fear* and *surprise* are approximately 8-9% in the dataset. Further, many utterances have more than one emotions representing the case of multi-label classification problem. The utterances for which all the emotions are absent, we classify them into a *no-emotion* class. We depict the statistics of multi-label emotions in Table 2. The training, validation and test set distributions are approximately 16K, 2K, and 4.6K, respectively.

### 4.2 Setups

The above datasets offer different dimension of sentiment analysis. We define the following setups for our experiments.

- **Two-class** (*pos* and *neg*) classification: MOSEI, MOSI, ICT-MMMO, and MOUD.
- **Three-class** (*pos*, *neu*, and *neg*) classification: YouTube.
- **Five-class** (*strong pos*, *weak pos*, *neu*, *weak neg*, and *strong neg*) classification: MOSEI.
- **Seven-class** (*strong pos*, *moderate pos*, *weak pos*, *neu*, *weak neg*, *moderate neg*, and *strong neg*) classification: MOSEI and MOSI.
- **Intensity prediction:** MOSEI and MOSI.

### 4.3 Experiments

We implement our proposed model on the Python-based Keras deep learning library. As the evaluation metric, we employ accuracy (weighted accuracy (Tong et al., 2017)) and F1-score for the classification problems, while for the intensity prediction task, we compute Pearson correlation scores and mean-absolute-error (MAE).

We evaluate our proposed *CIA* model on five benchmark datasets *i.e., MOUD, MOSI, YouTube, ICT-MMMO, and MOSEI*. For all the datasets, we perform *grid search* to find the optimal hyper-parameters (c.f. Table 4). Though we push for a generic hyper-parameter configuration for all datasets, in some cases, a different choice of the parameter has a significant effect. Therefore, we choose different parameters for different datasets for our experiments. Details of hyper-parameters for different datasets are depicted in Table 4.

We use different activation functions for the various modules in our model. We use *tanh* as the activation function for the inter-modal interactive module (IIM), while we employ *ReLu* for the context-aware attention module. For each dataset, we use *Adam* as optimizer.

In this paper, we address three multi-modal affective analysis problems, namely *i.e., sentiment classification ($S_C$), sentiment intensity ($S_I$) and emotion classification ($E_C$)*. We use *softmax* as a classifier for sentiment classification, while optimizing the *categorical cross-entropy* as a loss function. In comparison, we use *sigmoid* for prediction and *binary cross-entropy* as the loss function for the emotion classification. As the emotions in the dataset are multi-labeled, we apply a threshold over the predicted *sigmoid* outputs for each emotion and consider all the emotions as present whose respective values are above the threshold. We cross-validate and optimize both

| Modality | MOSEI | | | | | | | | MOSI | | | | | ICT-MMMO | | YouTube | | MOUD | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Emotion | | Sentiment | | | | | | Sentiment | | | | | Sentiment | | Sentiment | | Sentiment | |
| | *F1* | *W-Acc* | *F1* | $A^2$ | $A^5$ | $A^7$ | *MAE* | *r* | *F1* | $A^2$ | $A^7$ | *MAE* | *r* | *F1* | $A^2$ | *F1* | $A^3$ | *F1* | $A^2$ |
| **CIA** | **79.02** | **62.97** | **78.23** | **80.37** | **49.15** | **50.14** | **0.683** | **0.594** | **79.54** | **79.88** | **38.92** | **0.914** | **0.689** | **81.47** | **82.75** | **55.13** | **55.93** | **82.07** | **82.41** |
| CIA – IIM | 77.81 | 61.86 | 77.69 | 79.53 | 48.02 | 49.16 | 0.714 | 0.566 | 39.32 | 55.24 | 34.40 | 0.941 | 0.652 | 80.10 | 81.25 | 50.02 | 52.54 | 78.14 | 78.30 |

Table 3: Ablation results for IIM module.

the evaluation metrics, i.e., F1-score and weighted accuracy, and set the threshold as 0.4 and 0.18, respectively.

| Parameters | MOUD | MOSI | YouTube | MMMO | MOSEI |
|---|---|---|---|---|---|
| Bi-GRU | 50N | | | 200N | 100N |
| | 0.3D | | | | 0.5D |
| FC | 50N | 200N | 100N | | |
| | 0.5D | | | | |
| Activations | *ReLu* as activation for our model | | | | |
| | *tanh* as activation in IIM | | | | |
| Output | Softmax ($S_C$), tanh ($S_I$) & Sigmoid ($E_C$) | | | | |
| Optimizer | Adam (lr=0.001) | | | | |
| IIM Loss | Mean Square Error (MSE) | | | | |
| Model Loss | Cross-entropy (Classification) & MSE (Intensity) | | | | |
| Threshold | 0.4 (F1) & 0.18 (W-Acc) for multi label in $E_C$ | | | | |
| Batch | 16 | | | | |
| Epochs | 50 | | | | |

Table 4: Hyper-parameters for our experiments where *N, D*, $S_C$, $S_I$ and $E_C$ stands for #neurons, dropout, sentiment classification, sentiment intensity and emotion classification respectively.

We evaluate our proposed approach for all the possible input combinations *i.e., uni-modal (T, A, V), bi-modal (T+V, T+A, A+V) and tri-modal (T+V+A)*. We depict our obtained results in Table 1. For MOSEI dataset, with tri-modal inputs, our proposed system reports 79.02% F1-score and 62.97% weighted-accuracy for emotion classification. For sentiment classification, we obtain 78.23%, 80.37%, 49.15% and 50.14% as F1-score for two-class, five-class and seven-class, respectively. For sentiment intensity prediction task, our proposed system yields MAE and Pearson score of 0.683 and 0.594, respectively. We also observe that the proposed approach yields better performance for the tri-modal inputs than the bi-modal and uni-modal input combinations. This improvement implies that our proposed CIA architecture utilizes the interaction among the input modalities very effectively. Furthermore, for the other datasets, i.e., MOSI, ICT-MMMO, YouTube, and MOUD, we also observe a similar phenomenon as well (c.f. Table 1).

To show that our proposed IIM module, indeed, learns the interaction among the distinct modalities, we also perform an ablation study of the proposed CIA architecture. Consequently, we omit the IIM module from our architecture and compute the self-attention on the pair-wise fully-connected representations for the prediction. We observe that, for all the datasets, the performance of this modified architecture (i.e., CIA - IIM) is constantly inferior (with 1% to 7% F-score points) to the proposed CIA architecture. This performance degradation suggests that the IIM module is, indeed, an important component of our proposed architecture. In Table 3, we depict the evaluation results for both- with and without IIM.

### 4.4 Comparative Analysis

In this section, we present our comparative studies against several existing and recent state-of-the-art systems. For each dataset, we report three best systems for the comparisons[2]. In particular, we compare with the following systems: **Bag of Feature - Multimodal Sentiment Analysis (BoF-MSA)** (Blanchard et al., 2018), **Memory Fusion Network (MFN)** (Zadeh et al., 2018b), **Deep Fusion - Deep Neural Network (DF-DNN)** (Nojavanasghari et al., 2016), **Multi View - LSTM (MV-LSTM)** (Rajagopalan et al., 2016), **Early Fusion - LSTM (EF-LSTM)** (Zadeh et al., 2018c), **Tensor Fusion Network (TFN)** (Zadeh et al., 2017), **Random Forest (RF)** (Breiman, 2001), **Support Vector Machine** (Zadeh et al., 2016), **Multi-Attention Recurrent Network (MARN)** (Zadeh et al., 2018a), **Dynamic Fusion Graph (DFG)** (Zadeh et al., 2018c), **Multi Modal Multi Utterance-Bimodal Attention (MMMU-BA)** (Ghosal et al., 2018), **Bi-directional Contextual LSTM (BC-LSTM)** (Poria et al., 2017b) and **Multimodal Factorization Model (MFM)** (Tsai et al., 2018).

We show the comparative results in Table 5a and Table 5b for emotion and sentiment analysis, respectively. We observe that the proposed CIA framework yields better performance against the state-of-the-art for all the cases. For emotion classification, our proposed approach achieves approximately 3 and 0.6 percentage higher F1-

[2]Please note that we report all the results, which are available for comparison

| System | MOSEI | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | *Anger* | | *Disgust* | | *Fear* | | *Happy* | | *Sad* | | *Surprise* | | *Average* | |
| | *F1* | *W-Acc* | *F1* | *W-Acc* | *F1* | *W-Acc* | *F1* | *W-Acc* | *F1* | *W-Acc* | *F1* | *W-Acc* | *F1* | *W-Acc* |
| MFN⋆ | - | - | 71.4 | 65.2 | **89.9** | - | - | - | 60.8 | - | 85.4 | 53.3 | - | - |
| DF⋆ | 71.4 | - | - | 67.0 | - | - | - | - | - | - | - | - | - | - |
| MV-LSTM⋆ | - | 56.0 | - | - | - | - | - | - | - | - | - | - | - | - |
| EF-LSTM⋆ | - | - | - | - | - | 56.7 | - | 57.8 | - | 59.2 | - | - | - | - |
| TFN⋆ | - | 60.5 | - | - | - | - | 66.6 | 66.5 | - | 58.9 | - | 52.2 | - | - |
| RF⋆ | 72.0 | - | 73.2 | - | **89.9** | - | - | - | 61.8 | - | 85.4 | - | - | - |
| SVM⋆ | - | - | - | - | - | 60.0 | - | - | - | - | - | - | - | - |
| MARN⋆ | - | - | - | - | - | - | 71.0 | - | - | - | - | - | - | - |
| DFG | 72.8 | 62.6 | 76.6 | 69.1 | **89.9** | 62.0 | 66.3 | 66.3 | 66.9 | 60.4 | 85.5 | 53.7 | 76.3 | 62.3 |
| **CIA** | **74.7** | **67.4** | **81.8** | **74.1** | 87.8 | **63.9** | **71.3** | 51.9 | **72.6** | **61.8** | **86.0** | **58.2** | **79.0** | **62.9** |
| *T*-test | - | - | - | - | - | - | - | - | - | - | - | - | ***0.0002*** | ***0.0057*** |

(a) Emotion Analysis

| System | MOSEI | | | | | | MOSI | | | | | ICT-MMMO | | YouTube | | MOUD | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | *F1* | $A^2$ | $A^5$ | $A^7$ | *MAE* | *r* | *F1* | $A^2$ | $A^7$ | *MAE* | *r* | *F1* | $A^2$ | *F1* | $A^3$ | *F1* | $A^2$ |
| MARN† | - | - | - | - | - | - | 77.0 | 77.1 | 34.7 | 0.968 | 0.625 | - | - | - | 48.3 | 81.2 | 81.1 |
| MFN† | 76.0 | 76.0 | - | - | - | - | 77.3 | 77.4 | 34.1 | 0.965 | 0.632 | 73.1 | 73.8 | 51.6 | 51.7 | 80.4 | 81.1 |
| TFN† | - | - | - | - | - | - | - | - | - | - | - | 72.6 | 72.5 | - | - | - | - |
| BC-LSTM† | - | - | - | - | - | - | - | - | - | - | - | - | - | 45.1 | - | - | - |
| MFM | - | - | - | - | - | - | 78.1 | 78.1 | 36.2 | 0.951 | 0.662 | 79.2 | 81.3 | 52.4 | 53.3 | 81.7 | 82.1 |
| BoF-MSA⋆ | 63.2 | 60.0 | - | - | 0.91 | 0.30 | - | - | - | - | - | - | - | - | - | - | - |
| MV-LSTM⋆ | 76.4 | 76.4 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| DFG | 77.0 | 76.9 | 45.1 | 45.0 | 0.71 | 0.54 | - | - | - | - | - | - | - | - | - | - | - |
| MMMU-BA | 77.6 | 79.8 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| **CIA** | **78.2** | **80.4** | **49.2** | **50.1** | **0.68** | **0.59** | **79.54** | **79.88** | **38.92** | **0.914** | **0.689** | **81.47** | **82.75** | **55.13** | **55.93** | **82.07** | **82.41** |
| *T*-test | *0.038* | *0.038* | *0.00003* | *0.00001* | *0.0002* | *0.00001* | *0.0049* | *0.0022* | *0.0003* | *0.0018* | *0.0003* | *0.0005* | *0.0041* | *0.00006* | *0.00003* | *0.031* | *0.040* |

(b) Sentiment Analysis

Table 5: **Comparative results.** ⋆Values are taken from (Zadeh et al., 2018c). †Values are taken from (Tsai et al., 2018). Significance *T*-test ($< 0.05$) signifies that the obtained results are statistically significant over the existing systems with 95% confidence score.

score and weighted accuracy, respectively, than the state-of-the-art DFG (Zadeh et al., 2018c) system. Furthermore, we also see improvements for most of the individual emotion classes as well.

In sentiment analysis (c.f. Table 5b), for all the five datasets and different experimental setups, the proposed CIA framework obtains the improved accuracies for the classification tasks. For intensity prediction, our proposed framework yields lesser mean-absolute-error with high Pearson correlation scores. On average, we observe 1 to 5% improvement in accuracy values in comparison to the next best systems. Similarly, for the intensity prediction task, we report approximately 0.03 and 0.04 points improvement in mean-absolute-error and Pearson score, respectively.

We perform statistical significance test (*paired T-test*) on the obtained results and observe that performance improvement in the proposed model over the state-of-the-art is significant with 95% confidence (i.e., $p$-value$< 0.05$).

### 4.5 Error Analysis

We analyze our proposed *CIA* model to understand the importance of the baseline framework *CIA-IIM*. We study the predictions of both the models and observe that the proposed *CIA* framework improves the predictions of the baseline *CIA–IIM* model. It indicates that the CIA framework, indeed, learns the interaction among the input modalities, and the model effectively exploits this interaction for better judgment. In Table 6, we list the utterances of a CMU-MOSEI video along with their correct and predicted labels for both the proposed and baseline systems.

The video in Table 6 has 4 utterances, out of which the correct sentiments of three utterances (i.e., $u_1$, $u_3$, and $u_4$) are *positive*, while one utterance (i.e., $u_2$) is *negative*. We observe that our proposed *CIA* model predicts all the 4 utterances correctly, while the *CIA-IIM* mis-classify the sentiments of the utterances, $u_2$ and $u_3$.

We also analyze the *context-aware attention module (CAM)* with the help of heatmaps of the attention weights. The heatmaps, as depicted in Figure 3, represent the contributing utterances in the neighbourhood for the classification of each utterance. Figures 3a, 3b and 3c show the heatmaps of the pair-wise modality interaction of the proposed model *CIA*. In Figure 3a, each cell(*i,j*) of

| | Utterances | Sentiment | | | Emotion | | |
|---|---|---|---|---|---|---|---|
| | | Actual | CIA−IIM | CIA | Actual | CIA−IIM | CIA |
| 1 | *these critics argue that the welfare state breeds dependence and incompetence among those who receive it they* | Pos | Pos | Pos | Happy, Sad | Happy | Happy, Sad |
| 2 | *argue that it creates social pathologies such as single parent families excess fertility and laziness* | Neg | Pos | Neg | Sad | Happy | Sad |
| 3 | *some argue that people who receive welfare benefits cannot spend their benefits rationally and* | Pos | Neg | Pos | Fear, Sad | Sad | Fear, Sad |
| 4 | *and then lastly some people on the moral side argue that nothing should be given to a person without requiring a reciprocal obligation from that person so* | Pos | Pos | Pos | Sad | Happy, Sad | Sad |

Table 6: **Comparison between proposed *CIA* and *CIA−IIM* frameworks in MOSEI dataset.** Few cases where *CIA* framework performs better than the *CIA−IIM* framework. Red text signifies error in classification.

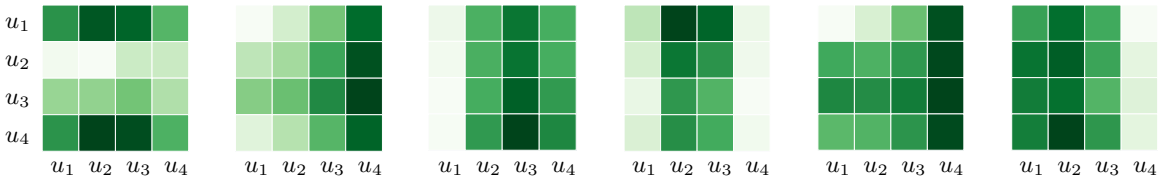


(a) $Sentiment_{TV}$ (b) $Sentiment_{TA}$ (c) $Sentiment_{AV}$ (d) $Emotion_{TV}$ (e) $Emotion_{TA}$ (f) $Emotion_{AV}$

Figure 3: **Heatmap analysis of MOSEI dataset** where **(a), (b) & (c)** represents the *Contextual Attention weights* for *TV, TA and AV* for sentiment and **(d), (e) & (f)** are *contextual attention weights* for *TV, TA and AV* for emotion.

the heatmap signifies the weights of utterance '*j*' for the classification of utterance '*i*'. For the utterance $u_4$, the model puts more attention weights on the $u_2$ and $u_3$ of the *text-visual* interactions, while for the *text-acoustic* interaction the model assigns higher weights to the $u_4$ utterance itself. Similarly, the model assigns the least weight to the $u_1$ utterance, whereas the utterance $u_3$ gets the highest weights. We argue that the proposed CAM module captures the diversity in the input modalities of the contextual utterances for the correct prediction.

For emotion prediction, the *CIA* model captures all the emotions correctly, while the *CIA-IIM* framework fails to predict the correct emotions of the utterances, $u_2$ and $u_3$. For the same video, we also show the attention heatmaps for emotion in Figure 3. For the utterance $u_2$, our proposed model (CIA) captures the emotion class '*sad*' as the CAM module assigns higher attention weights on the utterances $u_2$ and $u_3$ in Figure 3d, $u_4$ in Figure 3e, and $u_2$ in Figure 3f. Since the system finds the contributing neighbours as utterances $u_2$, $u_3$ and $u_4$ for various combinations, we argue that it utilizes the information of these utterances - which all express the '*sad*' emotion - for the correct prediction of utterance $u_2$ as '*sad*'.

## 5 Conclusion

In this paper, we have proposed a Context-aware Interactive Attention framework that aims to capture the interaction between the input modalities for the multi-modal sentiment and emotion prediction. We employed a contextual attention module to learn the contributing utterances in the neighborhood by exploiting the interaction among the input modalities. We evaluate our proposed approach on five standard multi-modal datasets. Experiments suggest the effectiveness of the proposed model over various existing systems, for both sentiment and emotion analysis, as we obtained new state-of-the-art for all five datasets.

In current work, we undertook the problem of sentiment and emotion analysis for a single-party utterances. In future, we would like to extend our work towards the multi-party dialogue.

## 6 Acknowledgment

The research reported here is partially supported by SkyMap Global India Private Limited. Asif Ekbal acknowledges the Young Faculty Research Fellowship (YFRF), supported by Visvesvaraya PhD scheme for Electronics and IT, Ministry of Electronics and Information Technology (MeitY), Government of India, being implemented by Digital India Corporation (formerly Media Lab Asia).

## References


Md Shad Akhtar, Dushyant Chauhan, Deepanway Ghosal, Soujanya Poria, Asif Ekbal, and Pushpak Bhattacharyya. 2019. Multi-task learning for multi-modal emotion recognition and sentiment analysis. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 370–379, Minneapolis, Minnesota. Association for Computational Linguistics.

John Arevalo, Thamar Solorio, Manuel Montes-y Gómez, and Fabio A González. 2017. Gated multimodal units for information fusion. *arXiv preprint arXiv:1702.01992*.

Nathaniel Blanchard, Daniel Moreira, Aparna Bharati, and Walter Scheirer. 2018. Getting the subtext without the text: Scalable multimodal sentiment classification from visual and acoustic modalities. In *Proceedings of Grand Challenge and Workshop on Human Multimodal Language*, pages 1–10.

Leo Breiman. 2001. Random forests. *Machine Learning*, 45(1):5–32.

KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259.

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. 2016. Gated-attention readers for text comprehension. *arXiv preprint arXiv:1606.01549*.

Jiamin Fu, Qirong Mao, Juanjuan Tu, and Yongzhao Zhan. 2017. Multimodal shared features learning for emotion recognition by enhanced sparse local discriminative canonical correlation analysis. *Multimedia Systems*, pages 1–11.

Deepanway Ghosal, Md Shad Akhtar, Dushyant Chauhan, Soujanya Poria, Asif Ekbal, and Pushpak Bhattacharyya. 2018. Contextual inter-modal attention for multi-modal sentiment analysis. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3454–3466, Brussels, Belgium. Association for Computational Linguistics.

Devamanyu Hazarika, Sruthi Gorantla, Soujanya Poria, and Roger Zimmermann. 2018. Self-attentive feature-level fusion for multimodal emotion detection. In *IEEE 1st Conference on Multimedia Information Processing and Retrieval, MIPR 2018, Miami, FL, USA, April 10-12, 2018*, pages 196–201.

Chan Woo Lee, Kyu Ye Song, Jihoon Jeong, and Woo Yong Choi. 2018. Convolutional attention networks for multimodal emotion recognition from speech and text data. *arXiv preprint arXiv:1805.06606*.

Rada Mihalcea. 2012. Multimodal sentiment analysis. In *Proceedings of the 3rd Workshop in Computational Approaches to Subjectivity and Sentiment Analysis, WASSA@ACL 2012, July 12, 2012, Jeju Island, Republic of Korea*, page 1.

Louis-Philippe Morency, Rada Mihalcea, and Payal Doshi. 2011a. Towards multimodal sentiment analysis: harvesting opinions from the web. In *Proceedings of the 13th International Conference on Multimodal Interfaces, ICMI 2011, Alicante, Spain, November 14-18, 2011*, pages 169–176.

Louis-Philippe Morency, Rada Mihalcea, and Payal Doshi. 2011b. Towards multimodal sentiment analysis: Harvesting opinions from the web. In *Proceedings of the 13th international conference on multimodal interfaces*, pages 169–176. ACM.

Behnaz Nojavanasghari, Deepak Gopinath, Jayanth Koushik, Tadas Baltrušaitis, and Louis-Philippe Morency. 2016. Deep multimodal fusion for persuasiveness prediction. In *Proceedings of the 18th ACM International Conference on Multimodal Interaction*, ICMI 2016, pages 284–288, New York, NY, USA. ACM.

Amol S Patwardhan. 2017. Multimodal mixed emotion detection. In *Communication and Electronics Systems (ICCES), 2017 2nd International Conference on*, pages 139–143. IEEE.

Verónica Pérez-Rosas, Rada Mihalcea, and Louis-Philippe Morency. 2013. Utterance-level multimodal sentiment analysis. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 973–982.

Soujanya Poria, Erik Cambria, Rajiv Bajpai, and Amir Hussain. 2017a. A review of affective computing: From unimodal analysis to multimodal fusion. *Information Fusion*, 37:98–125.

Soujanya Poria, Erik Cambria, Devamanyu Hazarika, Navonil Majumder, Amir Zadeh, and Louis-Philippe Morency. 2017b. Context-dependent sentiment analysis in user-generated videos. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 873–883.

Soujanya Poria, Iti Chaturvedi, Erik Cambria, and Amir Hussain. 2016. Convolutional mkl based multimodal emotion recognition and sentiment analysis. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pages 439–448. IEEE.

Soujanya Poria, Haiyun Peng, Amir Hussain, Newton Howard, and Erik Cambria. 2017c. Ensemble application of convolutional neural networks and multiple kernel learning for multimodal sentiment analysis. *Neurocomputing*, 261:217–230.

Shyam Sundar Rajagopalan, Louis-Philippe Morency, Tadas Baltrusaitis, and Roland Goecke. 2016. Extending long short-term memory for multi-view structured learning. In *European Conference on Computer Vision (ECCV-2016)*, pages 338–353. Springer International Publishing.

Hiranmayi Ranganathan, Shayok Chakraborty, and Sethuraman Panchanathan. 2016. Multimodal emotion recognition using deep learning architectures. In *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on*, pages 1–9. IEEE.

Edmund Tong, Amir Zadeh, Cara Jones, and Louis-Philippe Morency. 2017. Combating Human Trafficking with Multimodal Deep Models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1547–1556. Association for Computational Linguistics.

Yao-Hung Hubert Tsai, Paul Pu Liang, Amir Zadeh, Louis-Philippe Morency, and Ruslan Salakhutdinov. 2018. Learning factorized multimodal representations. *arXiv preprint arXiv:1806.06176*.

Panagiotis Tzirakis, George Trigeorgis, Mihalis A Nicolaou, Björn W Schuller, and Stefanos Zafeiriou. 2017. End-to-end multimodal emotion recognition using deep neural networks. *IEEE Journal of Selected Topics in Signal Processing*, 11(8):1301–1309.

Martin Wöllmer, Felix Weninger, Tobias Knaup, Björn Schuller, Congkai Sun, Kenji Sagae, and Louis-Philippe Morency. 2013. Youtube movie reviews: Sentiment analysis in an audio-visual context. *IEEE Intelligent Systems*, 28(3):46–53.

A Zadeh, PP Liang, S Poria, P Vij, E Cambria, and LP Morency. 2018a. Multi-attention recurrent network for human communication comprehension. In *Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-2018)*, pages 5642 – 5649, New Orleans, USA.

A. Zadeh, R. Zellers, E. Pincus, and L. P. Morency. 2016. Multimodal Sentiment Intensity Analysis in Videos: Facial Gestures and Verbal Messages. *IEEE Intelligent Systems*, 31(6):82–88.

Amir Zadeh, Minghai Chen, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. 2017. Tensor fusion network for multimodal sentiment analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1103–1114, Copenhagen, Denmark.

Amir Zadeh, Paul Pu Liang, Navonil Mazumder, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. 2018b. Memory fusion network for multi-view sequential learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*. AAAI Press.

Amir Zadeh, Paul Pu Liang, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. 2018c. Multimodal language analysis in the wild: Cmu-mosei dataset and interpretable dynamic fusion graph. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2236–2246. Association for Computational Linguistics.

# Sequential Learning of Convolutional Features for Effective Text Classification

**Avinash Madasu**
Samsung R&D Institute, Bangalore
m.avinash@samsung.com

**Vijjini Anvesh Rao**
Samsung R&D Institute, Bangalore
a.vijjini@samsung.com

## Abstract

Text classification has been one of the major problems in natural language processing. With the advent of deep learning, convolutional neural network (CNN) has been a popular solution to this task. However, CNNs which were first proposed for images, face many crucial challenges in the context of text processing, namely in their elementary blocks: convolution filters and max pooling. These challenges have largely been overlooked by the most existing CNN models proposed for text classification. In this paper, we present an experimental study on the fundamental blocks of CNNs in text categorization. Based on this critique, we propose Sequential Convolutional Attentive Recurrent Network (SCARN). The proposed SCARN model utilizes both the advantages of recurrent and convolutional structures efficiently in comparison to previously proposed recurrent convolutional models. We test our model on different text classification datasets across tasks like sentiment analysis and question classification. Extensive experiments establish that SCARN outperforms other recurrent convolutional architectures with significantly less parameters. Furthermore, SCARN achieves better performance compared to equally large various deep CNN and LSTM architectures.

## 1 Introduction

Text classification is one of the major applications of Natural Language Processing (NLP). Text classification involves classifying a text segment into different predefined categories. Sentiment analysis of product reviews, language detection, topic classification of various news articles are some of the problem statements of text classification. Prior to the success of deep learning, text classification was dealt using lexicon based features. These primarily involve parts-of-speech (POS) tagging where verbs and adjectives are given more importance compared to other POS. Frequency based feature selection techniques like Bag-of-Words (BoW), Term frequency-Inverse document frequency (TF-IDF) have been used and the features are trained using machine learning classifiers like Logistic Regression (LR) or Naive Bayes (NB). These approaches provided strong baselines for text classification (Wang and Manning, 2012). However, sequential patterns and semantic structure between words play a crucial role in deciding the category of a text. Traditional lexicon approaches fail to capture such complexities. Since the success of deep learning, neural network architectures have outperformed traditional methods like BoW and count based feature selection techniques. Neural network architectures especially Convolutional Neural Networks (CNN) and Long Short Term Memory (Hochreiter and Schmidhuber, 1997) (LSTM) have achieved state-of-the-art results in text classification. Shallow CNN architectures (Kim, 2014; Madasu and Rao, 2019a,b) achieved admirable results on text classification as well. Very deep CNN architectures were proposed based on character level features (Zhang et al., 2015) and word level features (Conneau et al., 2016) which significantly improved the performance in text classification. In image classification and object detection, previously proposed deep CNN architectures (Szegedy et al., 2016) achieved state-of-the-art results. The effectiveness of deep CNNs can be very well explained where primary convolutional layers detect the edges of an object. As we go deeper, more complex features of an image are learnt. But in case of text, there hasn't been much understanding behind their success. Popular CNN architectures use convolution with a fixed window size over the words in a sentence. However, a question arises whether sequential information is preserved using convolu-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5658–5667,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

tion across words. *What will be the effect if word sequences are randomly shuffled and convolution is applied on them?* In case of LSTM, random shuffling results in least performance as LSTMs rely on sequential learning and random ordering harms any such learning. Whereas, in case of a fixed window convolution applied across words, there is no strong evidence that it preserves sequential information. Previously proposed CNN architectures use max pooling operation across convolution outputs to capture most important features (Kim, 2014). This leads to another question, if the feature selected by max pooling will always be the most important feature of the input or otherwise.

To answer the above questions, we conduct a study of experiments about the effects of using fixed window convolution across words as discussed in Section 3.1. And the effects of using max pooling operation on convolution outputs are discussed in Section 3.2. Based on their critique, we propose a Sequential Convolutional Attentive Recurrent Network (SCARN) model for effective text classification. The proposed model relies on recurrent structures for preserving sequential information and convolution layers to learn task specific representations. In summary, the contributions of the paper are:

- We propose a new recurrent convolutional model for text classification by discussing the shortcomings of max pooling operation and also the strength and weakness of convolution operation.
- We evaluate the proposed model's performance on seven benchmark text classification datasets. The results show that the proposed model achieves better results with lesser number of parameters than other recurrent convolutional architectures.

## 2 Related Work

### 2.1 Recurrent Convolutional Networks

Previous architectures proposed for text classification were entirely convolution or recurrent based. There have been limited but popular works on combining recurrent structures with convolutional structures. (Lai et al., 2015) proposed a neural network (RCNN) model which combined recurrent and convolutional architectures. In their model each word is represented by a triplet of the word itself, its left and right context words, which are then trained sequentially using LSTM. Max pooling is then applied to capture the most important features of a sentence. Another attempt to combine recurrent and convolutional structures was made by Lee and Dernoncourt (2016) for sequential short-text classification. Their model uses LSTM to train short-texts and max pooling is applied on the outputs of all timesteps of LSTM to create a vector representation for each short-text. These architectures first use LSTM to train sequences and then max pooling operation is applied on its output. Zhou et al. (2015) proposed a recurrent convolutional network (C-LSTM) where convolution is applied over windows of multiple pretrained word vectors (fixed window size) to obtain higher level representations. These features serve as input to an LSTM for learning sequential information.

### 2.2 Attention Mechanism

Attention significantly improves the focusing on most meaningful information in the sentence. CNN and RNN architectures that were proposed with attention mechanism achieved superior performance to non-attention architectures. Attention originally proposed for machine translation (Bahdanau et al., 2014) was applied for document classification (Yang et al., 2016) as well. Several variants of attention mechanism have been proposed like global and local attention (Luong et al., 2015) and self attention (Lin et al., 2017). Attention has been applied to CNN blocks like pooling layers (Santos et al., 2016) for discriminative model training. Recently, attention based architectures achieved state-of-art results in question answering (Devlin et al., 2018).

## 3 Understanding Convolution and Max pooling

In this section we explore the working of CNNs in the context of natural language to better explain the intuition behind proposing SCARN. The key components of CNNs, namely convolution and max pooling strengths and weaknesses are discussed in this section.

### 3.1 Convolution operation

The convolution operation in the context of its application on word embeddings is discussed as follows.

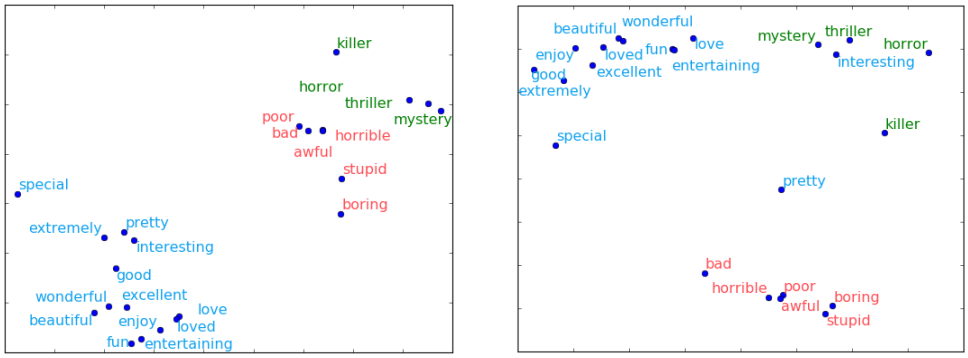


(a) Original Embeddings (b) Convolution Transformation

Figure 1: t-SNE projection of original embeddings and after convolution transformation

### What it doesn't learn

Convolution operation can be explained as a weighted sum across timesteps. This may result in loss of sequential information. To illustrate our hypothesis, we conduct an experiment on the Rotten Tomatoes dataset (Pang and Lee, 2005). In this experiment we train a CNN with convolution being applied on words with the fixed window size varying from one to maximum sentence length. We repeat this experiment with randomly shuffling all the words in an input sentence (Random ordering), and with shuffling every two consecutive words as shown in this example "read book forget movie" to "book read movie forget" (Alternate shuffle). The results of this experiment are illustrated in Figure 2[1]. Our observations in this experiment are the following:

- CNNs fail to fully incorporate sequential information as performance on random ordering and correct ordering are marginally near each other. As evident in the figure, the performance with correct ordering on window size 7 is comparable to performance with random ordering on window size 1.
- As window size increases, the difference between correct and incorrect orderings diminishes and finally converges when window size is complete sentence length. This indicates that the ability to capture sequential information by CNNs, decline with increasing window size.

Furthermore, we also note that while performance on random ordering is marginally less than correct ordering, it is still higher than other context blind algorithms like bag-of-words as shown in Table 1. This implies that while not fully exploiting sequential information, convolution filters still learn something valuable which we explore in next section.

### What it learns

To understand what convolution filters do, we train a CNN with single window size on the SST2 dataset (Socher et al., 2013) for sentiment classification. As the convolution acts over a single word, it has no ability to capture sequential information. Hence, irrespective of input sentence word order, words will always have same respective convolution output. Essentially, convolution layer here acts like an embedding transformation, where input embedding space is transformed into another space. To understand this new space we project the respective embeddings using t-SNE (Maaten and Hinton, 2008) and illustrate them in Figure 1. Here we make a distinction between the semantic knowledge of a word and the task specific knowledge of a word. As this experiment is done on dataset SST2, we refer to task specific knowledge as the sentiment knowledge of a word. Word embeddings[2] are generally trained on huge domain generic data because of which they capture the general semantic information of the word. Figure 1a shows these word embeddings as they are in the original space. As we can see, semantically similar words cluster together. Figure 1b shows words after their convolutional transformation. Blue coloured words represent words with positive sentiment. Red with negative sentiment

[1]Experiment on more datasets could be found in the Appendix B

[2]In this experiment GloVE word embeddings were used.

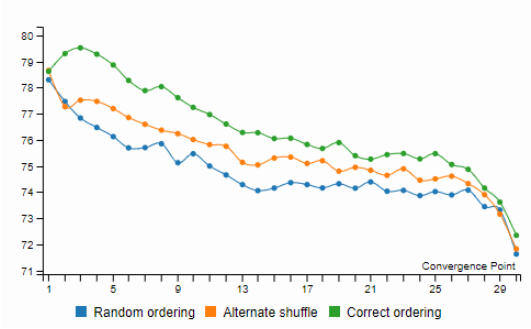


Figure 2: Accuracy (y-axis) percentage on Rotten Tomatoes dataset with varying window size.

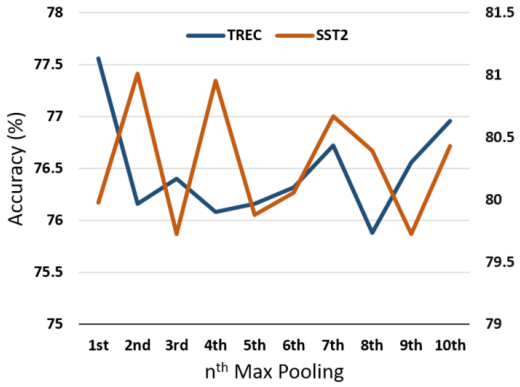


Figure 3: Accuracy (y-axis) percentage on TREC and SST2 datasets with varying $n$ for the $n^{th}$ Max pooling.

and green words are those which are semantically close to negative sentiment words. However, in the context of movie reviews, their sentiment value is close to positive sentiment words. From this experiment we observe:

- The transformation of the green words between original and convoluted outputs, shows that the convolution layer is able to tune their input embeddings such that they are more closer to the positive sentiment cluster than to their original semantic cluster. For example, a word like "killer" in the original embedding space is very close to negative sentiment words like, "bad" and "awful", especially if the word embeddings were produced on huge fact based datasets like Wikipedia or news datasets. However in the SST2 dataset, "killer" is often used to describe a movie very positively. Hence sequential learning on such transformed embeddings will be more effective than the original semantic embeddings.

- Some words like "pretty" might be semantically closer to the positive sentiment words. However in the dataset, we find common occurrences of phrases like "pretty good" or "pretty bad". The layer has hence transformed its position nearly equidistant to both the negative and positive clusters. In other words, convolution layer is able to robustly create a transformation which can handle any task specific noise.

The above observations show the effectiveness of the convolution operation. This is because a single convolution filter output value captures a weighted summation over all the features of the original word embedding. This enables the network to learn more task appropriate features as many as the number of filters.

With the above points in mind, we observe that convolution operation does not fully exploit sequential information, especially on larger window sizes. However we do see their effectiveness in conforming the input embeddings to a representation specifically meaningful to the task. Hence, for sequential learning instead of relying on convolution, we use recurrent architectures in our proposed SCARN model.

### 3.2 Max pooling operation: More doesn't mean better

Max pooling operation in images identify the discriminative features from convolution outputs. However this does not relate the same way of selecting most relevant features from convoluted features in texts. To illustrate this, we perform an experiment using a CNN architecture based on architecture by Kim (2014) popularly used in text classification. We define $n^{th}$ Max pooling as choosing $n^{th}$ highest value of all filters as opposed to first ($n$=1). By analyzing the performance distinctions on varying values of n, we make an attempt to assess whether maximum necessarily means task meaningful. The results are illustrated in Figure 3[3]. The results show that when $n$ is increased the performance varies arbitrarily on all datasets. This shows that there is no apparent corelation between magnitude of the values to importance for the task. Hence, this relationship does not stand strong for text based inputs.

[3]Experiment on more datasets could be found in the Appendix B

## 4 Model

### 4.1 Overview

The proposed SCARN model architecture is shown in the Figure 4. Let $V$ be the vocabulary size considered and X $\in \mathbb{R}^{V \times d}$ represent embedding matrix where $X_i$ is a $d$ dimensional word vector. Vocabulary words contained in the pretrained embeddings are initialized to their respective embedding vectors and words that are not present are assigned 0's. For each input text, a maximum length $N$ is considered. Zero padding is applied if the length of input text is less than $N$. Hence, each sentence or paragraph is converted to I $\in \mathbb{R}^{N \times d}$ dimensional vector which is the input to SCARN model.

The model consists of two subnetworks: Convolution Recurrent subnetwork and Recurrent Attentive subnetwork. In the first subnetwork, convolution with single window size is applied on the input $I$. Convolution filters learn the higher level representations from the input text. The output from convolution is trained sequentially using LSTM network. In the second subnetwork, the input $I$ is trained using LSTM. To better focus on most relevant words, attention mechanism is applied to the outputs of LSTM from every time step. Attention creates an alternate context vector for the input $I$ by choosing the most suitable words necessary for classification. The outputs from the first subnetwork and second subnetwork are concatenated and connected to the output layer through a dense connection.

### 4.2 Convolution Recurrent subnetwork

Let $I$ be represented as sequence of words $I = w_1 w_2 w_3 ... w_N$ where $w_i$ represent the $i^{th}$ word in the input. A total of $K$ convolution filters are applied on each word $w_i$ of input. Let $L \in \mathbb{R}^{K \times d}$ be the weight matrices of all filters. For each filter $l \in \mathbb{R}^{1 \times d}$ when applied on $w_i$, outputs a new feature $C_{il}$. Therefore, a new feature vector $C_i$ is obtained for each word $w_i$ after convolving with $K$ filters.

$$C_i = C_{i1} C_{i2} C_{i3} ..... C_{iK} \quad (1)$$

Similarly, the above procedure is repeated for all the words in the input to produce a feature vector $C \in \mathbb{R}^{N \times d \times K}$.

$$C = f(I \circledast L) \quad (2)$$

where $\circledast$ represents convolution operation and $f$ is the non-linear activation (ReLU). Applying convolution to individual words preserve the sequential information. Convolution learns the higher level representations for the input words. Each word will be transformed to a new representation pertinent to the task. The new feature representations are trained sequentially using LSTM.

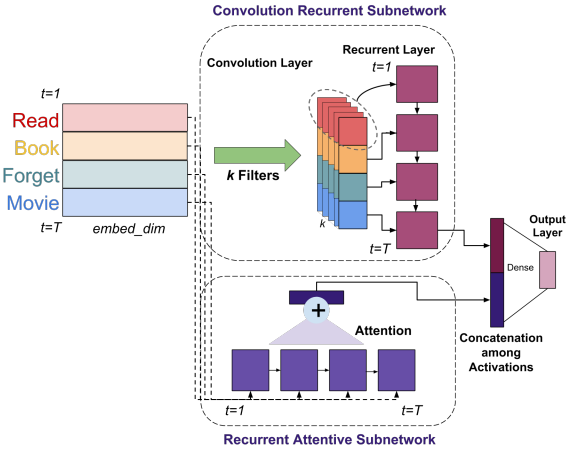


Figure 4: SCARN Architecture

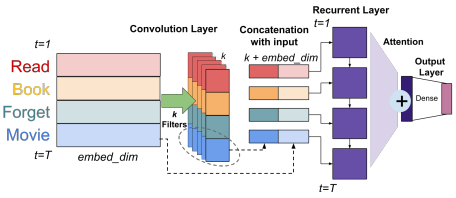


Figure 5: concat-SCARN architecture

### 4.3 Recurrent Attentive subnetwork

In this subnetwork, the input word embeddings are trained using LSTM. All words in the input sentence are not equally important in predicting final output. Although, LSTM learns sequential information, selecting significant information is a key issue for text classification. For this purpose, we employ an attention layer on the top of LSTM outputs. Attention mechanism focuses on specific significant words and tries to construct alternate context vector by aggregating the word representations.

## 5 Experiments

### 5.1 Datasets

We tested our model on standard benchmark datasets: Rotten-Tomatoes (RT) (Pang and Lee, 2005), Polarity V2.0 (Pol) (Pang and Lee, 2004), SST-2 (Socher et al., 2013), Home and Kitchen reviews (AR) (He and McAuley, 2016), TREC (Li and Roth, 2002), IMDB (Maas et al., 2011)

and Subjectivity Objectivity (SO) (Pang and Lee, 2004). The statistics for the datasets are shown in Table 3

### 5.2 Baselines

We compared our model to various text classification approaches[4].

#### BoW and TF-IDF + LR

Bag-of-words(BoW) and TF-IDF are strong baselines for text classification. BoW and TF-IDF features are extracted and softmax is applied on the top for classification.

#### Average Word Vectors + MLP

This baseline uses the average of word embeddings as features for the input text which are then trained using Multilayer perceptron (MLP).

#### Paragraph2Vec + MLP

Each input sentence is converted to a feature vector using Paragraph2vec(Le and Mikolov, 2014) which are then trained using Multilayer perceptron (MLP).

#### Deep CNN

This baseline is based on deep CNN architecture (Conneau et al., 2016) with approximately matching number of parameters as our model.

#### Char CNN

For our comparison, we employed the Char CNN architecture (Zhou et al., 2015) with less parameters compared to the original model, as the datasets used by them were considerably huge than ours.

#### LSTM and Bi-LSTM

We also offer a comparison with LSTM and Bi-LSTM architectures with a single hidden layer and approximately matching number of parameters as our model.

#### LSTM + Attention

In this baseline, attention mechanism is applied on the top of LSTM outputs across different time steps.

#### concat-SCARN

In this model, we concatenate the outputs from convolution layer as in SCARN, with input word embeddings at each time step. The concatenated outputs are trained using LSTM. Attention is applied on the top of this layer. Figure 5 shows the architecture of concat-SCARN model.

#### RCNN

We compared our model to the RCNN model (Lai et al., 2015) which uses max pooling for selecting the most appropriate features of a sentence.

#### C-LSTM

We compared our model to C-LSTM model (Zhou et al., 2015) which uses convolution over fixed window of words to learn higher level representations.

### 5.3 Implementation

#### Input and Training Details

We used google pretrained word vectors for C-LSTM[5], as it was used in the original work. For all the other experiments which require embeddings, we use GloVe pretrained word vectors[6]. The size of a word embedding in this model is 300. For each dataset, a maximum sentence length is considered which is 30 for TREC, SO, RT, Pol, SST-2 datasets, 400 for IMDB and 100 for AR dataset.

We apply a dropout layer (Srivastava et al., 2014) with a probability of 0.5 on the pretrained embeddings. We also apply dropout with a probability 0.5 on the dense layer that connects to output. We use Adam as the optimizer with a batch-size of 16 for small SCARN model and 50 for Large SCARN model. The initial learning rate is set to 0.0003. Training is done for 30 epochs.

#### Architecture

We employ two different architectures of SCARN model since the datasets vary in size. For datasets TREC, SO, RT, Pol, SST-2 the number of convolution filters are 50. Number of LSTM cells considered for these datasets are 32. We call this architecture **Small SCARN model**. For datasets AR and IMDB, number of convolution filters are 100 and number of LSTM cells are 64. We call this architecture **Large SCARN model**. The comparison of number of parameters for each model is shown in Table 2. ReLU is used as the activation function in convolution layers. In the output layer, softmax is used for multi-class classification and sigmoid for binary.

[4]For lack of code, results are from our implementations

[5]https://code.google.com/archive/p/word2vec/

[6]https://nlp.stanford.edu/data/glove.840B.300d.zip

| | Model | IMDB | TREC | SO | RT | Pol | AR | SST-2 |
|---|---|---|---|---|---|---|---|---|
| Linear | BoW + LR | 86.452 | 73.600 | 82.800 | 64.320 | 76.500 | 47.000 | 80.500 |
| | TFIDF + LR | 78.740 | 72.000 | 84.000 | 63.237 | 77.250 | 49.200 | 80.340 |
| Word Vector | Avg Word Vectors + MLP | 85.44 | 86.999 | 90 | 75.691 | 68.25 | 47.015 | 81.219 |
| | Paragraph2Vec + MLP | 77.472 | 45 | 77.2 | 62.996 | 77.25 | 38.925 | 67.27 |
| CNN | Deep CNN | 78.44 | 28.999 | 85.799 | 75.4 | 69.999 | 40.333 | 64.305 |
| | Char CNN | 70.484 | 74.199 | 59 | 50 | 50.249 | 34.37 | 60.516 |
| RNN | LSTM | 88.37 | 76.99 | 90.899 | 77.436 | 74.5 | 47.85 | 78.747 |
| | Bi-LSTM | 88.69 | 75.8 | 91.399 | 76.594 | 77.499 | 51.569 | 79.242 |
| Attention | LSTM+Attention | 88.35 | 77 | 89.7 | 76.895 | 77.75 | 50.568 | 80.01 |
| | concat-SCARN | 88.88 | 78.8 | 89.8 | 77.858 | 75.75 | 51.569 | 80.06 |
| RNN-CNN | RCNN | 86.607 | 79.6 | 91.1 | 78.098 | 78.25 | 48.026 | 80.395 |
| | C-LSTM | 87.676 | 90.4 | 91.3 | 76.474 | 67 | 52.784 | 78.308 |
| Our model | SCARN | **89.788** | **90.799** | **92.4** | **79.641** | **78.75** | **53.350** | **82.262** |

Table 1: Accuracy scores in percentage of all models on every dataset

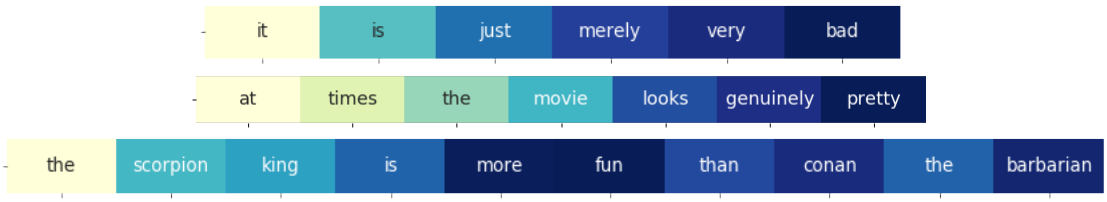


Figure 6: Attention weights for some of the sentences from the SST2 dataset

| Model | No. of parameters |
|---|---|
| Small SCARN | 68,425 |
| Large SCARN | 166,639 |
| RCNN | 180,601 |
| C-LSTM | 676,651 |

Table 2: Number of parameters for each model

## 6 Results and Discussion

Results of the experiments are tabulated in Table 1. We observe that, the proposed SCARN model outperforms linear and word vector based models, because of their inability to incorporate sequential information. We also compare our model to recurrent models like LSTM, Bidirectional-LSTM and find that SCARN outperforms them as these recurrent architectures even though learn sequential information, lack SCARN's learning of task specific representations through convolution. Meanwhile, SCARN outperforms deep CNN and char CNN models, for their lack of learning sequential information the same way recurrent architectures can. When SCARN is compared to other recurrent convolutional architectures RCNN and C-LSTM, SCARN achieves significantly much better performance with lesser parameters as shown in Table 2. RCNN uses max pooling to capture the most important feature. However our discussions in Section 3.2 show max pooling's choice of maximum may not necessitate importance. C-LSTM used fixed window convolution across words. But as seen in Section 3.1 fixed window convolution do not capture sequential information adequately. In SCARN model, we apply convolution across single word to learn task specific information, on which LSTM architecture is trained to capture contextual information.

We observe that SCARN significantly outperforms concat-SCARN on all the datasets. We believe this is because in concat-SCARN, concatenation between input embeddings and convolution activations is done. However they belong to a very different distribution. In Figure 7, we show the

| Dataset | Dataset size | Train | Dev | Test | Max Vocab size | Classes |
|---|---|---|---|---|---|---|
| IMDB | 50000 | 20000 | 5000 | 25000 | 30000 | 2 |
| TREC | 5952 | 4906 | 546 | 500 | 5000 | 6 |
| SO | 10000 | 8100 | 900 | 1000 | 30000 | 2 |
| RT | 10662 | 8100 | 900 | 1662 | 30000 | 2 |
| Pol | 2000 | 1280 | 320 | 400 | 30000 | 2 |
| AR | 121565 | 80000 | 20000 | 21565 | 30000 | 5 |
| SST-2 | 9613 | 6920 | 872 | 1821 | 10000 | 2 |

Table 3: Summary Statistics of all datasets

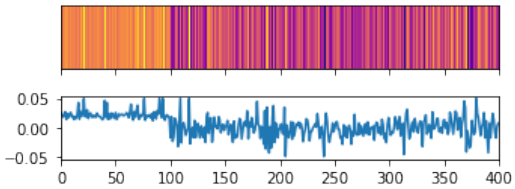


(a) Mean

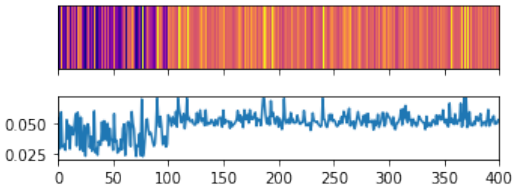


(b) Standard Deviation

Figure 7: Statistics of each feature in concatenation layer outputs on the IMDB dataset's training set.

average and standard deviation of outputs at the concatenation layer of concat-SCARN on IMDB dataset's training set. As evident from the figure, we can make a stark distinction between the two concatenated sections as shown in the architecture of concat-SCARN in Figure 5. Because of the irregular distributions, the learning may become biased towards one of the segments. Furthermore, attempts to address this issue by using batch normalization layers may not be a good idea, as normalizing word embedding inputs may spoil the semantic concepts underlying them. Hence, in our SCARN model, subnetworks are trained differently, and final activations are concatenated as illustrated in SCARN's architecture in Figure 4. As long as both subnetworks' outputs come from similar activation function, they will have similar distributions as well and hence they won't suffer from the same problem as concat-SCARN.

Effectiveness of attention has also been illustrated in Figure 6, where attention weights have been shown in a heat map, with darker colors corresponding to a higher weightage. We observe that SCARN is able to effectively utilize attention weights to focus on most contributing input words.

## 7 Conclusion

In this paper we present a critical study and viewpoint of CNNs, which even though are popularly used in text classification, details of it are often overlooked. We find that convolutional filters learn particularly in the context of sequential information. But at the same time, they are good at learning higher level task-relevant features. On the other hand, we find max pooling to be very arbitrary in selection of crucial features and hence contributing minimal to the overall task. We also find that the problems with input concatenation, as it imbalances the representations because of difference in nature of distributions. Based on our study we proposed SCARN, for effectively utilizing convolution and recurrent features for text classification. Our model beats other popular ways of combining recurrent and convolutional architectures with quite less number of parameters on various benchmark datasets. Our model also outperforms CNN and RNN architectures with equally deep or same number of parameters.

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2016. Very deep convolutional networks for text classification. *arXiv preprint arXiv:1606.01781*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep

bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, pages 507–517. International World Wide Web Conferences Steering Committee.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Twenty-ninth AAAI conference on artificial intelligence*.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196.

Ji Young Lee and Franck Dernoncourt. 2016. Sequential short-text classification with recurrent and convolutional neural networks. *arXiv preprint arXiv:1603.03827*.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, COLING '02, pages 1–7, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Avinash Madasu and Vijjini Anvesh Rao. 2019a. Effectiveness of self normalizing neural networks for text classification. *arXiv preprint arXiv:1905.01338*.

Avinash Madasu and Vijjini Anvesh Rao. 2019b. Gated convolutional neural networks for domain adaptation. In *International Conference on Applications of Natural Language to Information Systems*, pages 118–130. Springer.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity. In *Proceedings of ACL*, pages 271–278.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of ACL*, pages 115–124.

Cicero dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *arXiv preprint arXiv:1602.03609*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 90–94. Association for Computational Linguistics.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis Lau. 2015. A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630*.

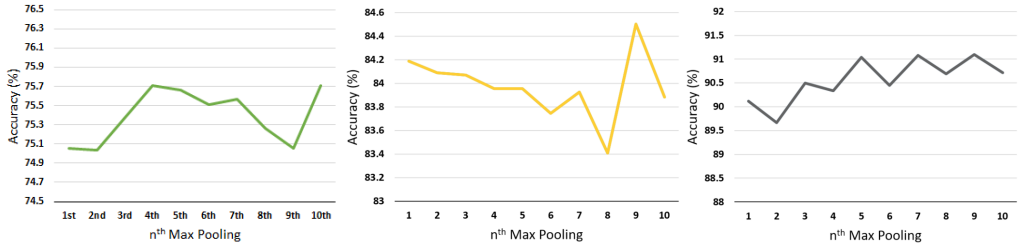


(a) RT (b) IMDB (c) SO

Figure 8: $n^{th}$ Max pooling experiments on RT, IMDB and SO Datasets

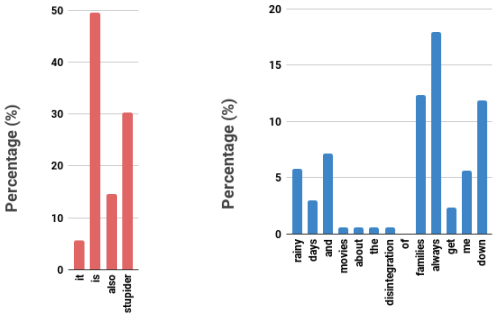


Figure 9: Percentage distribution of max pooling outputs for misclassified samples from SST2 dataset

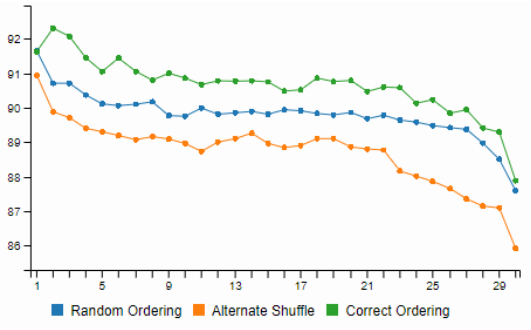


Figure 10: Accuracy (y-axis) percentage on SO dataset with varying window size.

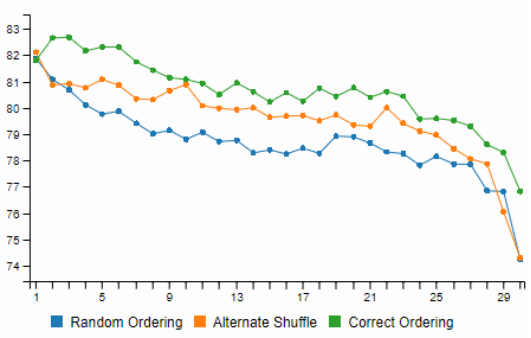


Figure 11: Accuracy (y-axis) percentage on SST2 dataset with varying window size.

## A Max Pooling: missclassified examples

For this experiment, after convolution over single word, we perform max pooling across filter outputs and attempt to discern the max pooling outputs. We identify which input words' convolution outputs were chosen by the max pooling operation. These distributions illustrated in Figure 9 are presented on some misclassified examples from the SST2 dataset. We see that often words that should have been contributing most to the overall sentiment value, have very less or minimal share. For example in the sentence, "it is also stupider", we see "is" having the near majority share, even though it tells nothing about the sentiment of the sentence. At the same time in "rainy days and movies about the disintegration of families always get me down", "disintegration" has almost no share despite being an important word for evaluating sentiment.

## B Experiments on more datasets

Figure 8 shows the $n^{th}$ Max pooling experiment explained in Section 3.2 on other datasets, namely RT, IMDB and SO. We still find that there is no co-relation between importance for task and magnitude of the values. Figure 10 and Figure 11 show the Ordering experiment from Section 3.1 on SO and SST2 datasets respectively. As evident from the figures, the proposed hypotheses stay sound on other datasets as well.

# The Role of Pragmatic and Discourse Context in Determining Argument Impact

**Esin Durmus**
Cornell University
ed459@cornell.edu

**Faisal Ladhak**
Amazon
faisall@amazon.com

**Claire Cardie**
Cornell University
cardie@cs.cornell.edu

## Abstract

Research in the social sciences and psychology has shown that the persuasiveness of an argument depends not only the language employed, but also on attributes of the source/communicator, the audience, and the appropriateness and strength of the argument's claims given the pragmatic and discourse context of the argument. Among these characteristics of persuasive arguments, prior work in NLP does not explicitly investigate the effect of the pragmatic and discourse context when determining argument quality. This paper presents a new dataset to initiate the study of this aspect of argumentation: it consists of a diverse collection of arguments covering 741 controversial topics and comprising over 47,000 claims. We further propose predictive models that incorporate the pragmatic and discourse context of argumentative claims and show that they outperform models that rely only on claim-specific linguistic features for predicting the perceived impact of individual claims within a particular line of argument.

## 1 Introduction

Previous work in the social sciences and psychology has shown that the impact and persuasive power of an argument depends not only on the language employed, but also on the credibility and character of the communicator (i.e. ethos) (Miller et al., 1976; Chaiken, 1979, 1980); the traits and prior beliefs of the audience (G. Lord et al., 1979; Davies, 1998; Correll et al., 2004; Hullett, 2005); and the pragmatic context in which the argument is presented (i.e. kairos) (Haugtvedt and Wegener, 1994; Joyce and Harwood, 2014).

Research in Natural Language Processing (NLP) has only partially corroborated these findings. One very influential line of work, for example, develops computational methods to automatically determine the linguistic characteristics of persuasive arguments (Habernal and Gurevych, 2016; Tan et al., 2016; Zhang et al., 2016), but it does so without controlling for the audience, the communicator or the pragmatic context.

Very recent work, on the other hand, shows that attributes of both the audience and the communicator constitute important cues for determining argument strength (Lukin et al., 2017; Durmus and Cardie, 2018). They further show that audience and communicator attributes can influence the relative importance of linguistic features for predicting the persuasiveness of an argument. These results confirm previous findings in the social sciences that show a person's perception of an argument can be influenced by his background and personality traits.

To the best of our knowledge, however, no NLP studies explicitly investigate the role of *kairos* — a component of pragmatic context that refers to the context-dependent "timeliness" and "appropriateness" of an argument and its claims within an argumentative discourse — in argument quality prediction. Among the many social science studies of attitude change, the order in which argumentative claims are shared with the audience has been studied extensively: Haugtvedt and Wegener (1994), for example, summarize studies showing that the argument-related claims a person is exposed to beforehand can affect his perception of an alternative argument in complex ways. Joyce and Harwood (2014) similarly find that changes in an argument's context can have a big impact on the audience's perception of the argument.

Some recent studies in NLP have investigated the effect of interactions on the overall persuasive power of posts in social media (Tan et al., 2016; Hidey and McKeown, 2018). However, in social media not all posts have to express arguments or stay on topic (Rakshit et al., 2017), and qualitative evaluation of the posts can be influenced by many



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5668–5678, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

other factors such as interactions between the individuals (Durmus and Cardie, 2019). Therefore, it is difficult to measure the effect of argumentative pragmatic context alone in argument quality prediction without the effect of these confounding factors using the datasets and models currently available in this line of research.

*In this paper, we study the role of kairos on argument quality prediction by examining the individual claims of an argument for their timeliness and appropriateness in the context of a particular line of argument.* We define kairos as the sequence of **argumentative** text (e.g. claims) along a particular line of argumentative reasoning.

To start, we present a dataset extracted from *kialo.com* of over 47,000 claims that are part of a diverse collection of arguments on 741 controversial topics. The structure of the website dictates that each argument must present a supporting or opposing claim for its parent claim, and stay within the topic of the main thesis. Rather than being posts on a social media platform, these are community-curated claims. Furthermore, for each presented claim, the audience votes on its impact within the given line of reasoning. Critically then, the dataset includes the argument context for each claim, allowing us to investigate the characteristics associated with impactful arguments.

With the dataset in hand, we propose the task of studying the characteristics of impactful claims by (1) taking the argument context into account, (2) studying the extent to which this context is important, and (3) determining the representation of context that is more effective. To the best of our knowledge, ours is the first dataset that includes claims with both impact votes and the corresponding context of the argument.

## 2 Related Work

Recent studies in computational argumentation have mainly focused on the tasks of identifying the structure of the arguments such as argument structure parsing (Peldszus and Stede, 2015; Park and Cardie, 2014), and argument component classification (Habernal and Gurevych, 2017; Mochales and Moens, 2011). More recently, there is an increased research interest to develop computational methods that can automatically evaluate qualitative characteristic of arguments, such as their impact and persuasive power (Habernal and Gurevych, 2016; Tan et al., 2016; Kelman, 1961; Burgoon et al., 1975; Chaiken, 1987; Tykocinskl et al., 1994; Dillard and Pfau, 2002; Cialdini, 2007; Durik et al., 2008; Marquart and Naderer, 2016). Consistent with findings in the social sciences and psychology, some of the work in NLP has shown that the impact and persuasive power of the arguments are not simply related to the linguistic characteristics of the language, but also on characteristics the source (ethos) (Durmus and Cardie, 2019) and the audience (Lukin et al., 2017; Durmus and Cardie, 2018). These studies suggest that perception of the arguments can be influenced by the credibility of the source, and the background of the audience.

It has also been shown, in social science studies, that *kairos*, which refers to the "timeliness" and "appropropriateness" of arguments and claims, is important to consider in studies of argument impact and persuasiveness (Haugtvedt and Wegener, 1994; Joyce and Harwood, 2014). One recent study in NLP has investigated the role of argument sequencing in argument persuasion looking at (Hidey and McKeown, 2018) Change My View[1], which is a social media platform where users post their views, and challenge other users to present arguments in an attempt to change their them. However, as stated in (Rakshit et al., 2017) many posts on social media platforms either do not express an argument, or diverge from the main topic of conversation. Therefore, it is difficult to measure the effect of pragmatic context in argument impact and persuasion, without confounding factors from using noisy social media data. In contrast, we provide a dataset of claims along with their structured argument path, which only consists of claims and corresponds to a particular line of reasoning for the given controversial topic. This structure enables us to study the characteristics of impactful claims, accounting for the effect of the pragmatic context.

Consistent with previous findings in the social sciences, we find that incorporating pragmatic and discourse context is important in computational studies of persuasion, as predictive models that with the context representation outperform models that only incorporate claim-specific linguistic features, in predicting the impact of a claim. Such a system that can predict the impact of a claim given an argumentative discourse, for example, could potentially be employed by argument retrieval and

[1]https://www.reddit.com/r/changemyview/.

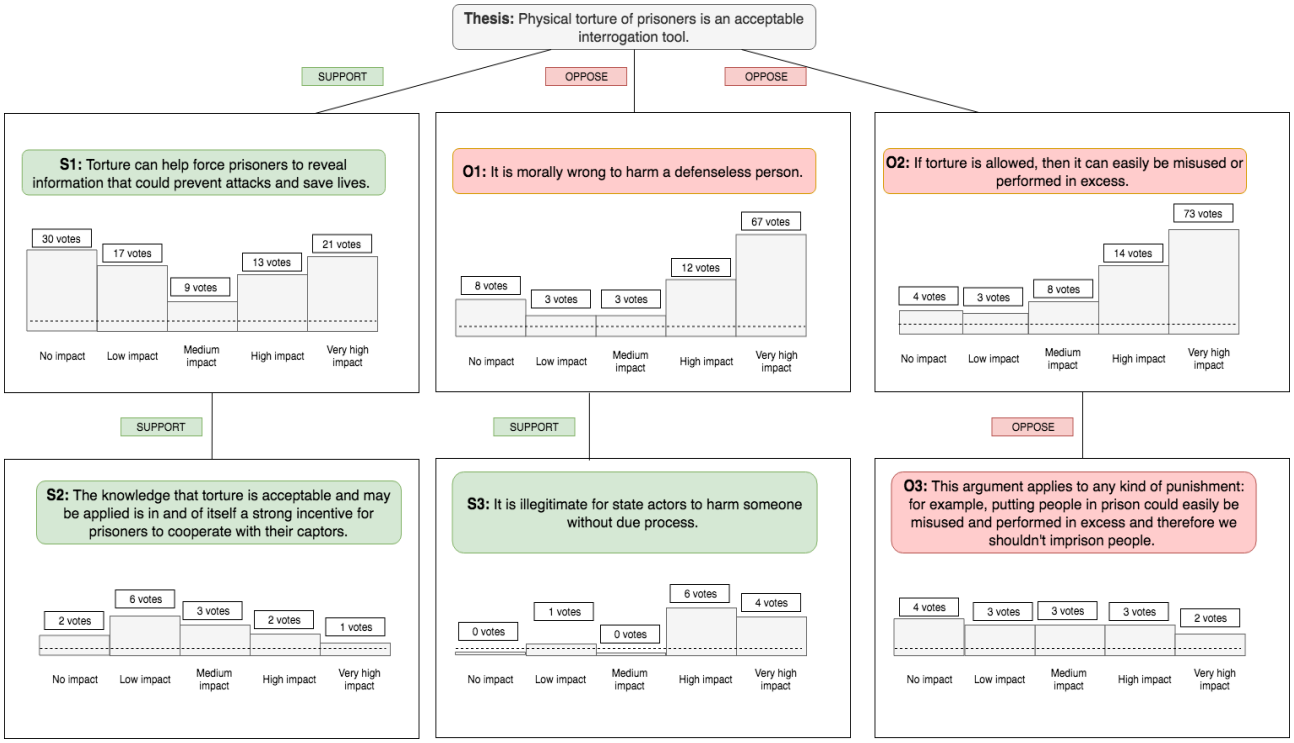


Figure 1: Example partial argument tree with claims and corresponding impact votes for the thesis "PHYSICAL TORTURE OF PRISONERS IS AN ACCEPTABLE INTERROGATION TOOL.".

generation models which aims to pick or generate the most appropriate possible claim given the discourse.

## 3 Dataset

**Claims and impact votes.** We collected 47,219 claims from *kialo.com*[2][3] for 741 controversial topics and their corresponding impact votes. Impact votes are provided by the users of the platform to evaluate how impactful a particular claim is. Users can pick one of 5 possible impact labels for a particular claim: NO IMPACT, LOW IMPACT, MEDIUM IMPACT, HIGH IMPACT and VERY HIGH IMPACT. While evaluating the impact of a claim, users have access to the full argument context and therefore, they can assess how impactful a claim is in the given context of an argument. An interesting observation is that, in this dataset, the same claim can have different impact labels depending on the context in which it is presented.

Figure 1 shows a partial **argument tree** for the argument **thesis** "PHYSICAL TORTURE OF PRISONERS IS AN ACCEPTABLE INTERROGATION TOOL.". Each node in the argument tree corresponds to a claim, and these argument trees are constructed and edited collaboratively by the users of the platform.

Except the thesis, every claim in the argument tree either opposes or supports its parent claim. Each path from the root to leaf nodes corresponds to an **argument path** which represents a particular line of reasoning on the given controversial topic.

Moreover, each claim has **impact votes** assigned by the users of the platform. The impact votes evaluate how impactful a claim is within its context, which consists of its predecessor claims from the thesis of the tree. For example, claim **O1** "IT IS MORALLY WRONG TO HARM A DEFENSELESS PERSON" is an opposing claim for the thesis and it is an IMPACTFUL CLAIM since most of its impact votes belong to the category of VERY HIGH IMPACT. However, claim **S3** "IT IS ILLEGITIMATE FOR STATE ACTORS TO HARM SOMEONE WITHOUT THE PROCESS" is a supporting claim for its parent **O1** and it is a less impactful claim since most of the impact votes belong to the NO IMPACT and LOW IMPACT categories.

**Distribution of impact votes.** The distribution of claims with the given range of number of im-

[2]The data is collected from this website in accordance with the terms and conditions.

[3]There is prior work by Durmus et al. (2019) which created a dataset of argument trees from *kialo.com*. That dataset, however, does not include any impact labels.

| # impact votes | # claims |
|---|---|
| $[3, 5)$ | 4,495 |
| $[5, 10)$ | 5,405 |
| $[10, 15)$ | 5,338 |
| $[15, 20)$ | 2,093 |
| $[20, 25)$ | 934 |
| $[25, 50)$ | 992 |
| $[50, 333)$ | 255 |

Table 1: Number of claims for the given range of number of votes. There are 19,512 claims in the dataset with 3 or more votes. Out of the claims with 3 or more votes, majority of them have 5 or more votes.

pact votes are shown in Table 1. There are 19,512 claims in total with 3 or more votes. Out of the claims with 3 or more votes, majority of them have 5 or more votes. We limit our study to the claims with at least 5 votes to have a more reliable assignment for the accumulated impact label for each claim.

**Impact label statistics.** Table 3 shows the distribution of the number of votes for each of the impact categories. The claims have $241,884$ total votes. The majority of the impact votes belong to MEDIUM IMPACT category. We observe that users assign more HIGH IMPACT and VERY HIGH IMPACT votes than LOW IMPACT and NO IMPACT votes respectively. When we restrict the claims to the ones with at least 5 impact votes, we have $213,277$ votes in total[4].

**Agreement for the impact votes.** To determine the agreement in assigning the impact label for a particular claim, for each claim, we compute the percentage of the votes that are the same as the majority impact vote for that claim. Let $c_i$ denote the count of the claims with the class labels C=[NO IMPACT, LOW IMPACT, MEDIUM IMPACT, HIGH IMPACT, VERY HIGH IMPACT] for the impact label $l$ at index $i$.

$$\text{Agreement} = 100 * \frac{\max_{0 \leq i \leq 4} c_i}{\sum_{i=0}^{4} c_i}\% \quad (1)$$

For example, for claim S1 in Figure 1, the agreement score is $100 * \frac{30}{90}\% = 33.33\%$ since the majority class (NO IMPACT) has 30 votes and there are 90 impact votes in total for this particular claim. We compute the agreement score for the cases where (1) we treat each impact label separately (5-class case) and (2) we combine the classes HIGH IMPACT and VERY HIGH IMPACT into a one class: IMPACTFUL and NO IMPACT and LOW IMPACT into a one class: NOT IMPACTFUL (3-class case).

Table 2 shows the number of claims with the given agreement score thresholds when we include the claims with at least 5 votes. We see that when we combine the low impact and high impact classes, there are more claims with high agreement score. This may imply that distinguishing between no impact-low impact and high impact-very high impact classes is difficult. To decrease the sparsity issue, in our experiments, we use 3-class representation for the impact labels. Moreover, to have a more reliable assignment of impact labels, we consider only the claims with have more than 60% agreement.

**Context.** In an argument tree, the claims from the thesis node (root) to each leaf node, form an argument path. This argument path represents a particular line of reasoning for the given thesis. Similarly, for each claim, all the claims along the path from the thesis to the claim, represent the **context** for the claim. For example, in Figure 1, the context for **O1** consists of only the thesis, whereas the context for **S3** consists of both the thesis and **O1** since **S3** is provided to support the claim **O1** which is an opposing claim for the thesis.

The claims are not constructed independently from their context since they are written in consideration with the line of reasoning so far. In most cases, each claim elaborates on the point made by its parent and presents cases to support or oppose the parent claim's points. Similarly, when users evaluate the impact of a claim, they consider if the claim is timely and appropriate given its context. There are cases in the dataset where the same claim has different impact labels, when presented within a different context. Therefore, we claim that it is not sufficient to only study the linguistic characteristic of a claim to determine its impact, but it is also necessary to consider its context in determining the impact.

*Context length* ($C_l$) for a particular claim *C* is defined by number of claims included in the argument path starting from the thesis until the claim *C*. For example, in Figure 1, the context length for **O1** and **S3** are 1 and 2 respectively. Table 4 shows number of claims with the given range of con-

[4]26,998 of them NO IMPACT, 33,789 of them LOW IMPACT, 55,616 of them MEDIUM IMPACT, 47,494 of them HIGH IMPACT and 49,380 of them VERY HIGH IMPACT.

| | 3-class case | 5-class case |
|---|---|---|
| Agreement score | Number of claims | Number of claims |
| $> 50\%$ | 10,848 | 7,304 |
| $> 60\%$ | 7,386 | 4,329 |
| $> 70\%$ | 4,412 | 2,195 |
| $> 80\%$ | 2,068 | 840 |

Table 2: Number of claims, with at least 5 votes, above the given threshold of agreement percentage for 3-class and 5-class cases. When we combine the low impact and high impact classes, there are more claims with high agreement score.

| Impact label | # votes- all claims |
|---|---|
| No impact | 32,681 |
| Low impact | 37,457 |
| Medium impact | 60,136 |
| High impact | 52,764 |
| Very high impact | 58,846 |
| Total # votes | 241,884 |

Table 3: Number of votes for the given impact label. There are $241,884$ total votes and majority of them belongs to the category MEDIUM IMPACT.

| Context length | # claims |
|---|---|
| 1 | 1,524 |
| 2 | 1,977 |
| 3 | 1,181 |
| $[4, 5]$ | 1,436 |
| $(5, 10]$ | 1,115 |
| $> 10$ | 153 |

Table 4: Number of claims for the given range of context length, for claims with more than 5 votes and an agreement score greater than $60\%$.

text length for the claims with more than 5 votes and $60\%$ agreement score. We observe that more than half of these claims have 3 or higher context length.

# 4 Methodology

## 4.1 Hypothesis and Task Description

Similar to prior work, our aim is to understand the characteristics of impactful claims in argumentation. However, we **hypothesize** that the qualitative characteristics of arguments is not independent of the context in which they are presented. To understand the relationship between argument context and the impact of a claim, we aim to incorporate the context along with the claim itself in our predictive models.

**Prediction task.** Given a claim, we want to predict the impact label that is assigned to it by the users: NOT IMPACTFUL, MEDIUM IMPACT, or IMPACTFUL.

**Preprocessing.** We restrict our study to claims with at least 5 or more votes and greater than $60\%$ agreement, to have a reliable impact label assignment. We have $7,386$ claims in the dataset satisfying these constraints[5]. We see that the impact class IMPACFUL is the majority class since around $58\%$ of the claims belong to this category.

For our experiments, we split our data to train (70%), validation (15%) and test (15%) sets.

## 4.2 Baseline Models

### 4.2.1 Majority

The majority baseline assigns the most common label of the training examples (HIGH IMPACT) to every test example.

### 4.2.2 SVM with RBF kernel

Similar to (Habernal and Gurevych, 2016), we experiment with SVM with RBF kernel, with features that represent (1) the simple characteristics of the argument tree and (2) the linguistic characteristics of the claim.

The features that represent the simple characteristics of the claim's argument tree include the distance and similarity of the claim to the thesis, the similarity of a claim with its parent, and the impact votes of the claim's parent claim. We encode the similarity of a claim to its parent and the thesis claim with the cosine similarity of their tf-idf vectors. The distance and similarity metrics aim to model whether claims which are more similar (i.e. potentially more topically relevant) to their parent claim or the thesis claim, are more impactful.

[5]We have 1,633 NOT IMPACTFUL, 1,445 MEDIUM IMPACT and 4,308 IMPACFUL claims.

We encode the quality of the parent claim as the number of votes for each impact class, and incorporate it as a feature to understand if it is more likely for a claim to impactful given an impactful parent claim.

**Linguistic features**. To represent each claim, we extracted the linguistic features proposed by (Habernal and Gurevych, 2016) such as tf-idf scores for unigrams and bigrams, ratio of quotation marks, exclamation marks, modal verbs, stop words, type-token ratio, hedging (Hyland, 1998), named entity types, POS n-grams, sentiment (Hutto and Gilbert, 2014) and subjectivity scores (Wilson et al., 2005), spell-checking, readibility features such as *Coleman-Liau* (Coleman and Liau, 1975), *Flesch* (Flesch, 1948), argument lexicon features (Somasundaran et al., 2007) and surface features such as word lengths, sentence lengths, word types, and number of complex words[6].

#### 4.2.3 FastText

Joulin et al. (2017) introduced a simple, yet effective baseline for text classification, which they show to be competitive with deep learning classifiers in terms of accuracy. Their method represents a sequence of text as a bag of n-grams, and each n-gram is passed through a look-up table to get its dense vector representation. The overall sequence representation is simply an average over the dense representations of the bag of n-grams, and is fed into a linear classifier to predict the label. We use the code released by Joulin et al. (2017) to train a classifier for argument impact prediction, based on the claim text[7].

#### 4.2.4 BiLSTM with Attention

Another effective baseline (Zhou et al., 2016; Yang et al., 2016) for text classification consists of encoding the text sequence using a bidirectional Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997), to get the token representations in context, and then attending (Luong et al., 2015) over the tokens to get the sequence representation. For the query vector for attention, we use a learned context vector, similar to Yang et al. (2016). We picked our hyperparameters based on performance on the validation set, and report our results for the best set of hyperparameters[8]. We initialized our word embeddings with glove vectors (Pennington et al., 2014) pre-trained on Wikipedia + Gigaword, and used the Adam optimizer (Kingma and Ba, 2015) with its default settings.

### 4.3 Fine-tuned BERT model

Devlin et al. (2018) fine-tuned a pre-trained deep bi-directional transformer language model (which they call BERT), by adding a simple classification layer on top, and achieved state of the art results across a variety of NLP tasks. We employ their pre-trained language models for our task and compare it to our baseline models. For all the architectures described below, we finetune for 10 epochs, with a learning rate of 2e-5. We employ an early stopping procedure based on the model performance on a validation set.

#### 4.3.1 Claim with no context

In this setting, we attempt to classify the impact of the claim, based on the text of the claim only. We follow the fine-tuning procedure for sequence classification detailed in (Devlin et al., 2018), and input the claim text as a sequence of tokens preceded by the special [CLS] token and followed by the special [SEP] token. We add a classification layer on top of the BERT encoder, to which we pass the representation of the [CLS] token, and fine-tune this for argument impact prediction.

#### 4.3.2 Claim with parent representation

In this setting, we use the parent claim's text, in addition to the target claim text, in order to classify the impact of the target claim. We treat this as a sequence pair classification task, and combine both the target claim and parent claim as a single sequence of tokens, separated by the special separator [SEP]. We then follow the same procedure above, for fine-tuning.

#### 4.3.3 Incorporating larger context

In this setting, we consider incorporating a larger context from the discourse, in order to assess the impact of a claim. In particular, we consider up

[6] We pick the parameters for SVM model according to the performance validation split, and report the results on the test split.

[7] We used maxNgram legnth of 2, learning rate of 0.8, num epochs of 15, vector dim of 300. We also used the pre-trained 300-dim wiki-news vectors made available on the fastText website.

[8] Our final hyperparams were: 100-dim word embedding, 100-dim context vector, 1 layer BiLSTM with 64 units, trained for 40 epochs with early stopping based on validation performance.

| | Precision | Recall | F1 |
|---|---|---|---|
| Majority | 19.43 | 33.33 | 24.55 |
| SVM with RBF Kernel | | | |
| Distance from the thesis | 27.42 | 33.53 | 26.05 |
| Parent quality | 58.11 | 47.85 | 46.61 |
| Linguistic features | **65.67** | 38.58 | 35.42 |
| BiLSTM with Attention | $46.50_{\pm 0.28}$ | $46.35_{\pm 0.99}$ | $46.22_{\pm 0.58}$ |
| FastText | $51.18_{\pm 0.80}$ | $46.09_{\pm 0.64}$ | $47.06_{\pm 0.70}$ |
| BERT models | | | |
| Claim only | $53.24_{\pm 1.07}$ | $50.93_{\pm 2.01}$ | $51.53_{\pm 1.53}$ |
| Claim + Parent | $55.79_{\pm 1.72}$ | $53.54_{\pm 2.09}$ | $54.00_{\pm 1.79}$ |
| Claim + Context$_f(2)$ | $56.57_{\pm 0.85}$ | $54.76_{\pm 1.71}$ | $55.18_{\pm 0.99}$ |
| Claim + Context$_f(3)$ | $57.19_{\pm 0.92}$ | $\mathbf{55.77_{\pm 1.05}}$ | $\mathbf{55.98_{\pm 0.70}}$ |
| Claim + Context$_f(4)$ | $57.09_{\pm 1.71}$ | $55.31_{\pm 1.09}$ | $55.72_{\pm 1.14}$ |
| Claim + Context$_{gru}(4)$ | $54.95_{\pm 2.00}$ | $51.55_{\pm 1.27}$ | $52.37_{\pm 1.26}$ |
| Claim + Context$_a(4)$ | $56.60_{\pm 0.52}$ | $54.55_{\pm 0.57}$ | $54.65_{\pm 0.33}$ |

Table 5: Results for the baselines and the BERT models with and without the context. Best performing model is BERT with the representation of previous 3 claims in the path along with the claim representation itself. We run the models 5 times and we report the mean and standard deviation.

to four previous claims in the discourse (for a total context length of 5). We attempt to incorporate larger context into the BERT model in three different ways.

**Flat representation of the path.** The first, simple approach is to represent the entire path (claim + context) as a single sequence, where each of the claims is separated by the [SEP] token. BERT was trained on sequence pairs, and therefore the pre-trained encoders only have two segment embeddings (Devlin et al., 2018). So to fit multiple sequences into this framework, we indicate all tokens of the target claim as belonging to segment A and the tokens for all the claims in the discourse context as belonging to segment B. This way of representing the input, requires no additional changes to the architecture or retraining, and we can just finetune in a similar manner as above. We refer to this representation of the context as a flat representation, and denote the model as Context$_f(i)$, where $i$ indicates the length of the context that is incorporated into the model.

**Attention over context.** Recent work in incorporating argument sequence in predicting persuasiveness (Hidey and McKeown, 2018) has shown that hierarchical representations are effective in representing context. Similarly, we consider hierarchical representations for representing the discourse. We first encode each claim using the pre-trained BERT model as the claim encoder, and use the representation of the [CLS] token as claim representation. We then employ dot-product attention (Luong et al., 2015), to get a weighted representation for the context. We use a learned context vector as the query, for computing attention scores, similar to Yang et al. (2016). The attention score $\alpha_c$ is computed as shown below:

$$\alpha_c = \frac{exp(V_c^T V_l)}{\sum_{c \in D} exp(V_c^T V_l)} \quad (2)$$

Where $V_c$ is the claim representation that was computed with the BERT encoder as described above, $V_l$ is the learned context vector that is used for computing attention scores, and $D$ is the set of claims in the discourse. After computing the attention scores, the final context representation $v_d$ is computed as follows:

$$V_d = \sum_{c \in D} \alpha_c V_c \quad (3)$$

We then concatenate the context representation with the target claim representation $[V_d, V_r]$ and pass it to the classification layer to predict the quality. We denote this model as Context$_a(i)$.

**GRU to encode context** Similar to the approach above, we consider a hierarchical representation for representing the context. We compute the claim representations, as detailed above, and we then feed the discourse claims' representations (in sequence) into a bidirectional Gated Recurrent

| | $C_l = 1$ | $C_l = 2$ | $C_l = 3$ | $C_l = 4$ |
|---|---|---|---|---|
| BERT models | | | | |
| Claim only | $48.61_{\pm 3.16}$ | $53.15_{\pm 1.95}$ | $54.51_{\pm 1.91}$ | $50.89_{\pm 2.95}$ |
| Claim + Parent | $51.49_{\pm 2.63}$ | $54.78_{\pm 2.95}$ | $54.94_{\pm 2.72}$ | $51.94_{\pm 2.59}$ |
| Claim + Context$_f(2)$ | $52.84_{\pm 2.55}$ | $53.77_{\pm 1.00}$ | $55.24_{\pm 2.52}$ | $57.04_{\pm 1.19}$ |
| Claim + Context$_f(3)$ | $\mathbf{54.88_{\pm 2.49}}$ | $54.71_{\pm 1.74}$ | $52.93_{\pm 2.07}$ | $\mathbf{58.17_{\pm 1.89}}$ |
| Claim + Context$_f(4)$ | $54.47_{\pm 2.95}$ | $\mathbf{54.88_{\pm 1.53}}$ | $\mathbf{57.11_{\pm 3.38}}$ | $57.02_{\pm 2.22}$ |

Table 6: F1 scores of each model for the claims with various context length values.

Unit (GRU) (Cho et al., 2014), to compute the context representation. We concatenate this with the target claim representation and use this to predict the claim impact. We denote this model as Context$_{gru}(i)$.

## 5 Results and Analysis

Table 5 shows the macro precision, recall and F1 scores for the baselines as well as the BERT models with and without context representations[9].

We see that *parent quality* is a simple yet effective feature and SVM model with this feature can achieve significantly higher ($p < 0.001$)[10] F1 score ($46.61\%$) than *distance from the thesis* and *linguistic features*. Claims with higher impact parents are more likely to be have higher impact. *Similarity with the parent and thesis* is not significantly better than the *majority* baseline. Although the BiLSTM model with attention and FastText baselines performs better than the SVM with *distance from the thesis* and *linguistic features*, it has similar performance to the *parent quality* baseline.

We find that the BERT model with *claim only* representation performs significantly better ($p < 0.001$) than the baseline models. Incorporating the *parent representation* only along with the *claim representation* does not give significant improvement over representing the claim only. However, *incorporating the flat representation of the larger context* along with the claim representation consistently achieves significantly better ($p < 0.001$) performance than the claim representation alone. Similarly, *attention representation* over the context with the learned query vector achieves significantly better performance then the *claim representation* only ($p < 0.05$).

We find that the *flat representation* of the context achieves the highest F1 score. It may be more difficult for the models with a larger number of parameters to perform better than the *flat representation* since the dataset is small. We also observe that modeling 3 claims on the argument path before the target claim achieves the best F1 score ($55.98\%$).

To understand for what kinds of claims the best performing contextual model is more effective, we evaluate the BERT model with *flat context representation* for claims with context length values 1, 2, 3 and 4 separately. Table 6 shows the F1 score of the BERT model without context and with *flat context representation* with different lengths of context. For the claims with context length 1, adding Context$_f(3)$ and Context$_f(4)$ representation along with the claim achieves significantly better ($p < 0.05$) F1 score than modeling the *claim only*. Similarly for the claims with context length 3 and 4, Context$_f(4)$ and Context$_f(3)$ perform significantly better than BERT with *claim only* (($p < 0.05$) and ($p < 0.01$) respectively). We see that models with larger context are helpful even for claims which have limited context (e.g. $C_l = 1$). This may suggest that when we train the models with larger context, they learn how to represent the claims and their context better.

## 6 Conclusion

In this paper, we present a dataset of claims with their corresponding impact votes, and investigate the role of argumentative discourse context in argument impact classification. We experiment with various models to represent the claims and their context and find that incorporating the context information gives significant improvement in predicting argument impact. In our study, we find that *flat representation* of the context gives the best improvement in the performance and our analysis indicates that the contextual models perform better even for the claims with limited context.

[9]For the models that result in different scores with different random seed, we run them 5 times and report the mean and standard deviation.

[10]We perform two-sided t test for significance analysis.

## 7 Acknowledgements

This work was supported in part by NSF grants IIS-1815455 and SES-1741441. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of NSF or the U.S. Government.

## References

Michael Burgoon, Stephen B Jones, and Diane Stewart. 1975. Toward a message-centered theory of persuasion: Three empirical investigations of language intensity1. *Human Communication Research*, 1(3):240–256.

S. Chaiken. 1979. Communicator physical attractiveness and persuasion.

Shelly Chaiken. 1980. Heuristic versus systematic information processing and the use of source versus message cues in persuasion. *Journal of Personality and Social Psychology*, 39:752–766.

Shelly Chaiken. 1987. The heuristic model of persuasion. In *Social influence: the ontario symposium*, volume 5, pages 3–39. Hillsdale, NJ: Lawrence Erlbaum.

Kyunghyun Cho, Bart van Merriënboer, Çalar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Robert B. Cialdini. 2007. Influence: The psychology of persuasion.

Meri Coleman and T. L. Liau. 1975. A computer readability formula designed for machine scoring. *Journal of Applied Psychology*, 60(2):283 – 284.

Joshua Correll, Steven J. Spencer, and Mark P. Zanna. 2004. An affirmed self and an open mind: Self-affirmation and sensitivity to argument strength.

MF Davies. 1998. Dogmatism and belief formation: Output interference in the processing of supporting and contradictory cognitions. *Journal of Personality and Social Psychology*, 75:456–466.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

James Price Dillard and Michael Pfau. 2002. *The persuasion handbook: Developments in theory and practice*. Sage Publications.

Amanda M Durik, M Anne Britt, Rebecca Reynolds, and Jennifer Storey. 2008. The effects of hedges in persuasive arguments: A nuanced analysis of language. *Journal of Language and Social Psychology*, 27(3):217–234.

Esin Durmus and Claire Cardie. 2018. Exploring the role of prior beliefs for argument persuasion. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1035–1045, New Orleans, Louisiana. Association for Computational Linguistics.

Esin Durmus and Claire Cardie. 2019. Modeling the factors of user success in online debate. In *The World Wide Web Conference*, WWW '19, pages 2701–2707, New York, NY, USA. ACM.

Esin Durmus, Faisal Ladhak, and Claire Cardie. 2019. Determining relative argument specificity and stance for complex argumentative structures. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4630–4641, Florence, Italy. Association for Computational Linguistics.

Rudolph Flesch. 1948. A new readability yardstick. *Journal of Applied Psychology*, 32(3):221 – 233.

Charles G. Lord, Lee Ross, and Mark Lepper. 1979. Biased assimilation and attitude polarization: The effects of prior theories on subsequently considered evidence. *Journal of Personality and Social Psychology*, 37:2098–2109.

Ivan Habernal and Iryna Gurevych. 2016. What makes a convincing argument? empirical analysis and detecting attributes of convincingness in web argumentation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1214–1223, Austin, Texas. Association for Computational Linguistics.

Ivan Habernal and Iryna Gurevych. 2017. Argumentation mining in user-generated web discourse. *American Journal of Computational Linguistics*, 43(1):125–179.

Curtis P. Haugtvedt and Duane T. Wegener. 1994. Message Order Effects in Persuasion: An Attitude Strength Perspective. *Journal of Consumer Research*, 21(1):205–218.

Christopher Hidey and Kathleen McKeown. 2018. Persuasive influence detection: The role of argument sequencing.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Craig R. Hullett. 2005. The impact of mood on persuasion: A meta-analysis. *Communication Research*, 32(4):423–442.

C. Hutto and Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text.

Ken Hyland. 1998. *Hedging in Scientific Research Articles*. John Benjamins.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, Valencia, Spain. Association for Computational Linguistics.

Nick Joyce and Jake Harwood. 2014. Context and identification in persuasive mass communication. *Journal of Media Psychology: Theories, Methods, and Applications*, 26:50.

Herbert C Kelman. 1961. Processes of opinion change. *Public opinion quarterly*, 25(1):57–78.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Stephanie Lukin, Pranav Anand, Marilyn Walker, and Steve Whittaker. 2017. Argument strength is in the eye of the beholder: Audience effects in persuasion. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 742–753, Valencia, Spain. Association for Computational Linguistics.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Franziska Marquart and Brigitte Naderer. 2016. *Communication and Persuasion: Central and Peripheral Routes to Attitude Change*, pages 231–242. Springer Fachmedien Wiesbaden, Wiesbaden.

Norman Miller, Geoffrey Maruyama, Rex J. Beaber, and Keith Valone. 1976. Speed of speech and persuasion. *Journal of Personality and Social Psychology*, 34(4):615–624.

Raquel Mochales and Marie-Francine Moens. 2011. Argumentation mining. *Artificial Intelligence and Law*, 19(1):1–22.

Joonsuk Park and Claire Cardie. 2014. Identifying appropriate support for propositions in online user comments. In *Proceedings of the First Workshop on Argumentation Mining*, pages 29–38, Baltimore, Maryland. Association for Computational Linguistics.

Andreas Peldszus and Manfred Stede. 2015. Joint prediction in MST-style discourse parsing for argumentation mining. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 938–948, Lisbon, Portugal. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Geetanjali Rakshit, Kevin K. Bowden, Lena Reed, Amita Misra, and Marilyn A. Walker. 2017. Debbie, the debate bot of the future. *CoRR*, abs/1709.03167.

Swapna Somasundaran, Josef Ruppenhofer, and Janyce Wiebe. 2007. Detecting arguing and sentiment in meetings. In *Proceedings of the SIGdial Workshop on Discourse and Dialogue*, volume 6.

Chenhao Tan, Vlad Niculae, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2016. Winning arguments: Interaction dynamics and persuasion strategies in good-faith online discussions. In *Proceedings of the 25th International Conference on World Wide Web*, pages 613–624. International World Wide Web Conferences Steering Committee.

Orit Tykocinskl, E Tory Higgins, and Shelly Chaiken. 1994. Message framing, self-discrepancies, and yielding to persuasive messages: The motivational significance of psychological situations. *Personality and Social Psychology Bulletin*, 20(1):107–115.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 347–354. Association for Computational Linguistics.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, California. Association for Computational Linguistics.

Justine Zhang, Ravi Kumar, Sujith Ravi, and Cristian Danescu-Niculescu-Mizil. 2016. Conversational flow in Oxford-style debates. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational*

*Linguistics: Human Language Technologies*, pages 136–141, San Diego, California. Association for Computational Linguistics.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 207–212, Berlin, Germany. Association for Computational Linguistics.

# Aspect-Level Sentiment Analysis Via Convolution over Dependency Tree

**Kai Sun**
BDBC and SKLSDE
Beihang University,China
sunkai@buaa.edu.cn

**Richong Zhang***
BDBC and SKLSDE
Beihang University, China
zhangrc@act.buaa.edu.cn

**Samuel Mensah**
BDBC and SKLSDE
Beihang University, China
samensah@buaa.edu.cn

**Yongyi Mao**
School of EECS
University of Ottawa,Canada
ymao@uottawa.ca

**Xudong Liu**
BDBC and SKLSDE
Beihang University, Beijing, China
liuxd@act.buaa.edu.cn

## Abstract

We propose a method based on neural networks to identify the sentiment polarity of opinion words expressed on a specific aspect of a sentence. Although a large majority of works typically focus on leveraging the expressive power of neural networks in handling this task, we explore the possibility of integrating dependency trees with neural networks for representation learning. To this end, we present a convolution over a dependency tree (CDT) model which exploits a Bi-directional Long Short Term Memory (Bi-LSTM) to learn representations for features of a sentence, and further enhance the embeddings with a graph convolutional network (GCN) which operates directly on the dependency tree of the sentence. Our approach propagates both contextual and dependency information from opinion words to aspect words, offering discriminative properties for supervision. Experimental results ranks our approach as the new state-of-the-art in aspect-based sentiment classification.

## 1 Introduction

The current explosion in digital technology in recent years has led to a vast amount of opinionated materials on the internet. In particular, individuals have expressed opinions on several aspects of products, services, blogs, and comments which are deemed to be influential, especially when making purchase decisions based on product reviews (Schouten and Frasincar, 2015). However, due to the voluminous amount of content online, sifting through reviews to learn knowledge of opinions expressed on specific aspects of a review is cumbersome. This fact has led to an increase in research in aspect-based sentiment analysis (ABSA), which aims to find scalable solutions to address the problem automatically. More specifically, ABSA involves two tasks: (1) to identify aspects of a sentence, and (2) to determine the sentiment polarity (e.g. positive, negative, neutral) expressed on a specific aspect. In this paper, we focus on the second task: aspect-based sentiment classification.

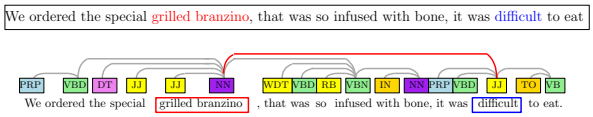


Figure 1: An example of a dependency tree where an opinion word (blue) and the specific aspect expression (red) are connected with other word tokens based on their syntactic dependencies.

With the aim to address the classification task, several methods have been developed. Majority of recent works such as (Dong et al., 2014; Tang et al., 2015; Wang et al., 2016; Chen et al., 2017; Cheng et al., 2017) have exploited neural networks due to its ability to model representations for sentences automatically. Even so, some recent methods have integrated both lexical resources with neural networks to achieve state-of-the-art performance in ABSA (Wang et al., 2018; Ouyang and Su, 2018).

Generally, we find that a dependency tree shortens the distance between the aspects and opinion words of a sentence, captures the syntactic relations between words, and offers discriminative syntactic paths on arbitrary sentences for information propagation across the tree. For instance, consider the dependency tree as depicted in Figure 1, the distance between the aspect expression *'grilled branzino'* and the opinion word *'difficult'* is shortened by a single path based on their syntactic dependencies. These properties allow neural network models to capture long-term syntactic dependencies effortlessly. Besides, dependency

*Corresponding author



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5679–5688, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

trees have graph-like structures bringing to play the recent class of neural networks, namely, graph convolutional networks (GCN) (Kipf and Welling, 2016). The GCN has been successful in learning representations for nodes, capturing the local position of nodes in the graph. Several AI applications such as link prediction (Schlichtkrull et al., 2018; Zitnik et al., 2018; Kong et al., 2019), semantic role labeling (Marcheggiani and Titov, 2017), and relation extraction (Zhang et al., 2018) have successfully exploited GCNs to improve representation learning.

These observations motivate us to develop a neural model which can operate on the dependency tree of a sentence, with the aim to make accurate sentiment predictions with respect to specific aspects. Specifically, we propose a convolution over a dependency tree (CDT) model which exploits a GCN to model the structure of a sentence through its dependency tree, where node (word) embeddings of the tree are initialized by means of a Bi-directional Long Short Term Memory (Bi-LSTM) network. Motivated by the recent work of (Zhang et al., 2018) in a relation extraction task, we find that the architecture of CDT allows the Bi-LSTM account for contextual information between successive words, while the GCN enhances the embeddings by modeling the dependencies along the syntactic paths of the dependency tree. Such operations allow information to be transferred from opinion words to aspect words, implying that the encoding for aspect words is sufficient for supervision in the classification task. Experimental results, including visualizations show the effectiveness of our proposed model.

## 2 Related Work

The performance bottleneck in the classification task of ABSA comes from modeling representations which efficiently encode the relationship between a specific aspect and the opinion words of a sentence. Most recent methods have focused on leveraging neural networks (Chen et al., 2017; Gu et al., 2018; Majumder et al., 2018; Fan et al., 2018a; Xue and Li, 2018; Huang and Carley, 2018; Zheng and Xia, 2018) which model representations automatically. Besides, in contrast to rule-based methods (Hu and Liu, 2004; Popescu and Etzioni, 2005; Ding et al., 2008; Popescu and Etzioni, 2005), neural networks are more capable of dealing with situations where opinion words are found in more complicated contexts.

Among neural network methods, some model the sentence representation using RNN variants such as LSTM and gated recurrent units (GRU). (Chen et al., 2017) handles the encoding of reviews using BiLSTM and attention networks. (Gu et al., 2018) improves the performance by considering the position of the aspect words. Similarly, (Zheng and Xia, 2018) use LSTMs to learn embeddings for the left context, right context and target phrase of sentences while considering the interactions between targets and contexts. (Majumder et al., 2018) on the other hand models the sentence representations using GRU and attention mechanisms. However convenient, these neural network based methods neglect informative resources such as dependency trees which is capable of shortening the distance between aspect and opinion words, enabling dependency information to be preserved effectively in lengthy sentences.

The state-of-the-art methods for representation learning have integrated dependency trees with neural networks. (Tai et al., 2015) proposed a tree-structured LSTM: a generalized class of LSTMs which enables the learning of dependency information between words and phrases. (Mou et al., 2015) exploit the short paths of dependency trees to learn representations of sentences using convolutional neural networks, while preserving dependency information. Motivated by such works, (Gu et al., 2018) proposed a position encoding convolutional neural network which takes into account the relative position of words and entities of a dependency tree for relation classification. Given that dependency trees can be considered as a graph, (Marcheggiani and Titov, 2017) introduced a variant of a GCN to model representations for dependency graphs in semantic role labeling tasks.

In a recent relation extraction task, (Zhang et al., 2018) extract entity-based representations via a GCN which operates on a dependecy tree. (Zhang et al., 2018) observed that stacking a GCN layer over an LSTM improves performance immensely. We follow a similar approach and propose CDT: a method which performs convolutions over a dependency tree to extract rich representations for aspect-based sentiment classification. CDT extracts a final representation for the ABSA classification task by aggregating only the aspect vec-

tors. We believe this is sufficient because the GCN componet can be interpreted as a messaging passing network which propagates information along edges. Thus, successive GCN operations allow information to be propagated across the network, and hence aspect vectors are encoded with information from opinion words which should be sufficient for supervision.

## 3 Convolution over Dependency Tree Model

In this section, we describe the CDT model which takes as input a dependency tree of a sentence. Node embeddings of the dependency tree are initially modeled by means of a BiLSTM, and the embeddings are further enhanced via a GCN. Finally, an aggregator is applied over the enhanced aspect embeddings to distill a dense vector embedding for the classification task. In particular, we aim to extract embeddings which encode both contextual and dependency information between a specific aspect expression and opinion words, providing supervisory signals for the aspect-based classification task.

We briefly describe the BiLSTM model, which takes as input the sentence $s$ with $n$ ordered word embeddings. The BiLSTM integrates context information in the word embeddings by keeping track of dependencies along the chain of words. Given an aspect-sentence pair $(a, s)$, where $a = \{a_1, a_2, \ldots, a_l\}$ is a sub-sequence of the sentence $s = \{w_1, w_2, \ldots, w_n\}$. The sentence $s$ has corresponding word embeddings $x = \{x_1, x_2, \ldots x_n\}$. The LSTM learns hidden state representations $\{\overrightarrow{h_1^0}, \overrightarrow{h_2^0}, \ldots \overrightarrow{h_n^0}\}$ in the forward direction on the word embeddings in $x$. This allows contextual information to be captured in a forward direction. In a similar fashion, a backward LSTM will learn representations $\{\overleftarrow{h_1^0}, \overleftarrow{h_2^0}, \ldots \overleftarrow{h_n^0}\}$ on $x$. Finally, we can concatenate the corresponding parallel representations modeled by both forward and backward LSTMs into higher dimensional representations $\{h_1^0, h_2^0, \ldots, h_n^0\}$, which contains the subsequence $\{h_{a_1}^0, h_{a_2}^0, \ldots, h_{a_l}^0\}$ corresponding to the aspect expression $a$. In doing so, we capture contextual information between opinion words and aspects. Besides, we integrate dependency information in the contextualized embeddings using a GCN which operates directly on the dependency tree of the sentence.

### 3.1 Graph Convolutional Network

The dependency tree can be interpreted as a graph $G$ with $n$ nodes, where nodes represent words in the sentence and edges represent syntactic dependency paths between words in the graph. The nodes of the dependency tree are given by real-valued vectors modeled by BiLSTM as described above. This structure allows a GCN to operate directly on the graph to model dependencies that exist between words. To allow the GCN to model node embeddings efficiently, we allow $G$ to have self-loops. The GCN approach ensures that the sentence structure represented by the dependency tree is encoded efficiently, whereby the representations for nodes encode the local position of opinion words and the target words in the dependency tree.

The dependency tree $G$ for any arbitrary sentence can be represented as an $n \times n$ adjacency matrix $A$, with entries $A_{ij}$ signaling if node $i$ is connected to node $j$ by a single dependency path in $G$. Specifically, $A_{ij} = 1$ if node $i$ is connected to node $j$, and $A_{ij} = 0$ otherwise. Together with node embeddings modeled by BiLSTMs, we can exploit a GCN capable of operating directly on graphs. The GCN makes efficient use of dependency paths to transform and propagate information across the paths, and update node embeddings by aggregating the propagated information. In such an operation, the GCN only considers the first-order neighborhood of a node when modeling its embeddings. However, $k$ successive GCN operations result in the propagation of information across the $k$-th order neighborhood. A single node embedding update takes the form

$$h_i^{(k+1)} = \phi\left(\sum_{j=1}^{n} c^i A_{ij}\left(W^{(k)} h_j^{(k)} + b^{(k)}\right)\right), \tag{1}$$

where $h_j^{(k)}$ is the hidden state representation for node $j$ at the $k^{th}$ layer of the GCN, $b^{(k)}$ is a bias term, $W^{(k)}$ is a parameter matrix, $c^i$ is a normalization constant, which we choose as $c^i = 1/d_i$. $d_i$ denotes the degree of node $i$ in the graph calculated as $d_i = \sum_{j=1}^{n} A_{ij}$. $\phi(\cdot)$ is a relu elementwise non-linear activation function. Note that $h_i^0$ represent the initial embeddings modeled by a BiLSTM, and $h_i^{(k+1)}$ is the final output for node $i$ at layer $k$.

In extracting a final embedding for the classifi-

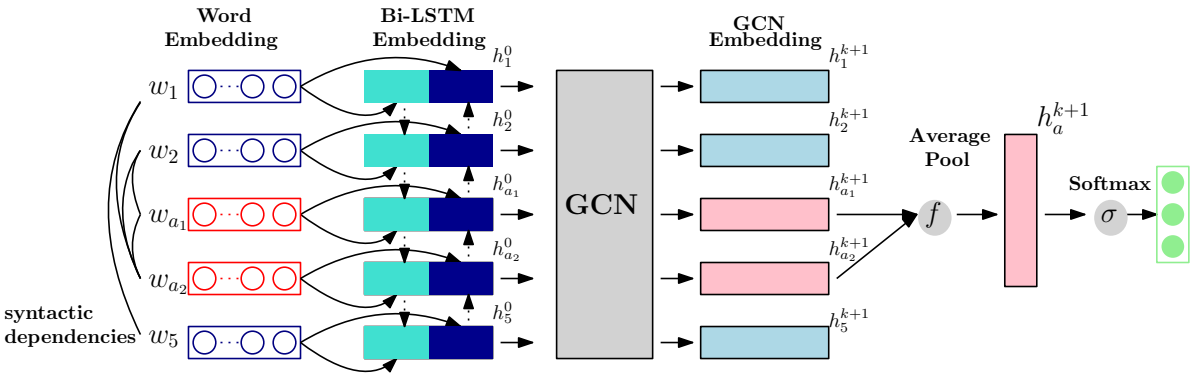


Figure 2: Overview of the CDT model based on the sentence $s = [w_1, w_2, w_{a_1}, w_{a_2}, w_5]$ , where $[w_{a_1}, w_{a_2}]$ is the specific aspect expression in $s$, and $k$ is the number of GCN layers.

cation task, we exploit a simple aggregator. For our framework, we choose an average pool which aggregates information over the aspect vectors. We choose to aggregate only the aspect vectors because we believe that these vectors encode contextual and dependency information owing to the BiLSTM and the GCN respectively. The BiLSTM and the GCN can be interpreted as message passing networks. Specifically, the BiLSTM allow aspect words of an arbitrary sentence to be contextualize, while the GCN finds the local position of aspect words in the syntactic dependency tree. The local position within the dependency tree encodes dependency information of a word with respect to its neighbors. As a result, the BiLSTM and the GCN allow embeddings for aspect words to have discriminative features, providing supervisory signals for the classification task. Moreover, we perform an average pool to retain most of the information in the aspect vectors. The pool operation over the aspect vectors takes the form of

$$h_a^{(k+1)} = f\left(\{h_{a_1}^{k+1}, h_{a_2}^{k+1}, \ldots, h_{a_l}^{k+1}\}\right), \quad (2)$$

where $f(\cdot)$ is an average pool function applied over the enhanced aspect vectors. We present an overview of the model architecture in Figure 2 based on an example sentence input.

## 4 Model Training

The aspect-based representation $h_a^{(k+1)}$ is passed to a fully connected softmax layer $\sigma$ whose output is a probability distribution over the different sentiment polarities. The model is trained end-to-end through a backpropagation, where the objective function to be minimized is the cross entropy error defined as

$$J(\theta_1, \theta_2) = -\sum_{(a,s)\in D}\sum_{c\in C} y_c((a,s)) \log \hat{y}_c((a,s)), \quad (3)$$

where $D$ is a collection of aspect-sentence pairs, $C$ is the collection of distinct sentiment classes, $y_c((a,s))$ is the ground truth for $(a,s)$ which takes the value of either 1 or 0. Besides, $(a,s)$ can belong to only one sentiment class. Hence $y_c((a,s)) = 1$ indicates that the ground truth sentiment class for $(a,s)$ is $c$. $\hat{y}_c((a,s))$ is the model's prediction for $(a,s)$. $\theta_1, \theta_2$ are trainable parameters for the BiLSTM and GCN respectively.

## 5 Experiment

In this section, we conduct experiments to validate our model which we denote as CDT on benchmark datasets. We also present restricted versions of our model denoted as ASP-BiLSTM and ASP-GCN. Unlike our main model, ASP-BiLSTM only exploits BiLSTM to model contextual information with respect to a specific aspect expression, while ASP-GCN exploits a GCN to model dependencies between words. Both models extract a final embedding on the aspect vectors. We propose these two models to observe the performance of GCN and BiLSTM, as well as the performance when we stack a GCN on a BiLSTM which forms the CDT model. To distinguish CDT as the new state-of-the-art in aspect-based sentiment classification, we compare CDT with several well established models, showing that CDT outperforms the very recent models in the classification task. In particular, we perform case studies with visualizations to verify our approach of aggregating only aspect vectors for the final embedding. We further present visualizations on case examples showing how GCN

improves on a simple BiLSTM model.

### 5.1 Datasets

We evaluate the performance of our model on SemEval 2014 (Pontiki et al., 2014), which consists of restaurant reviews (Rest14) and laptop reviews (Laptop14). We also evaluate our model on SemEval 2016 [1] containing restaurant reviews (Rest16). Experiments are also performed on a collection of tweets from Twitter provided in the works of (Dong et al., 2014). We summarize the statistics of the datasets in Table 1.

| Dataset | Positive | | Neutral | | Negative | |
|---|---|---|---|---|---|---|
| | train | test | train | test | train | test |
| Rest14 | 2164 | 727 | 637 | 196 | 807 | 196 |
| Laptop14 | 976 | 337 | 455 | 167 | 851 | 128 |
| Rest16 | 1657 | 611 | 101 | 44 | 748 | 204 |
| Twitter | 1507 | 172 | 3016 | 336 | 1528 | 169 |

Table 1: Distribution of samples by class labels on benchmark datasets

### 5.2 Implementation and parameter settings

For fairness in model comparation, we use similar parameters in compared models. Specifically, we exploit 300-dimensional Glove vectors (Pennington et al., 2014) for the word embeddings, as well as a 30-dimensional part-of-speech (POS) embeddings, 30-dimensional position embeddings, which is used to identify the relative position of each word with respect to the aspect in the sentence. We concatenate both word, POS and position embeddings, and learn a 50-dimensional BiLSTM embeddings for each word. The GCN operates on the dependency tree of the sentence to enhance the BiLSTM embeddings. All sentences are parsed by the Stanford parser. [2] To encourage the GCN to model dependencies between words, we randomly dropout $10\%$ of neurons per layer, and about $0.7$ at the input layer. The GCN model is trained for $100$ epochs with batch size $32$. We use the adam optimizer with learning rate $0.01$ for all datasets. The code for our model is found on the Github page [3].

### 5.3 Compared Prior Art

As a baseline, we include CNN and LSTM models, which learn representations from both word embeddings and position embeddings. We denote these models as CNN+Position and LSTM+Position. We also include a CNN baseline method which exploits an attention mechanism to model the relation between aspect words and context words. We denote this model as CNN+ATT. These models extract a final embedding by aggregating all learned embeddings using an average pool. In particular, we compare our proposed model with very recent models on the benchmark datasets. The models we consider include,

- TNet (Li et al., 2018a): In this work, BiLSTM embeddings are transformed into target-specific embeddings, and a CNN model is used to extract a final embedding.
- PRET+MULT (He et al., 2018b): A multi-task framework based on LSTMs is proposed to transfer knowledge from a document-level model task to an aspect-level model task.
- SA-LSTM-P (Wang and Lu, 2018): This work first learn embeddings using BiLSTM and model structural dependencies between words by means of a segmentation attention mechanism.
- LSTM+SynATT+TarRep (He et al., 2018a): This method models target representation as a weighted sum of aspect embeddings, and models the syntactic structure of the sentence using an attention mechanism.
- MGAN (Fan et al., 2018b): A BiLSTM is exploited to capture contextual information in the sentence, while a multi-grained attention mechanism is proposed to extract an embedding which effectively captures the interaction between the aspect and the context.
- MGAN (Li et al., 2018b): This work integrates an alignment mechanism in a multi-task model comprising of an aspect-term task and an aspect-category task to effectively extract aspect-specific representations.
- HSCN (Lei et al., 2019): A model is proposed to capture interactions between the context and target, select target words and extract target-specific contextual representation, while measuring the deviation between target-specific contextual representation and target representations.

[1] http://alt.qcri.org/semeval2016/task5/

[2] https://stanfordnlp.github.io/CoreNLP/

[3] https://github.com/sunkaikai/CDT_ABSA

| Model | Rest14 | | Laptop | | Twitter | | Rest16 | |
|---|---|---|---|---|---|---|---|---|
| | ACC | F1 | ACC | F1 | ACC | F1 | ACC | F1 |
| SOTA | 81.60 | 71.91 | 76.54 | 71.75 | 74.97 | 73.6 | 85.58 | 69.76 |
| CNN+Position | 79.37 | 68.64 | 72.73 | 68.28 | 72.69 | 70.92 | 84.63 | 64.75 |
| LSTM+Position | 77.59 | 67.05 | 70.06 | 64.46 | 71.39 | 69.45 | 83.47 | 62.69 |
| CNN+ATT | 79.46 | 69.44 | 70.53 | 64.27 | 73.12 | 71.01 | 84.28 | 60.86 |
| TNet (Li et al., 2018a) | 80.79 | 70.84 | 76.54 | 71.75 | 74.97 | 73.6 | - | - |
| PRET+MULT (He et al., 2018b) | 79.11 | 69.73 | 71.15 | 67.46 | - | - | 85.58 | 69.76 |
| SA-LSTM-P (Wang and Lu, 2018) | 81.60 | - | 75.1 | - | 69.0 | - | - | - |
| LSTM+SynATT+TarRep (He et al., 2018a) | 80.63 | 71.32 | 71.94 | 69.23 | - | - | 84.61 | 67.45 |
| MGAN (Fan et al., 2018b) | 81.25 | 71.94 | 75.39 | 72.47 | 72.54 | 70.81 | - | - |
| MGAN (Li et al., 2018b) | 81.49 | 71.48 | 76.21 | 71.42 | 74.62 | 73.53 | - | - |
| HSCN (Li et al., 2018b) | 77.8 | 70.20 | 76.1 | 72.5 | 69.6 | 66.1 | - | - |
| ASP-BiLSTM | 80.95 | 72.38 | 74.22 | 69.35 | 73.66 | 72.32 | 85.12 | 66.92 |
| ASP-GCN | 81.30 | 73.18 | 74.53 | 69.78 | 70.91 | 69.07 | 81.85 | 61.2 |
| CDT | **82.30** | **74.02** | **77.19** | **72.99** | 74.66 | **73.66** | **85.58** | **69.93** |

Table 2: Performance comparison on different models on the benchmark datasets. The best performance are bold-typed.

### 5.4 Performance Comparison

In this section, we compare model performance of recent methods with CDT, ASP-BiLSTM and ASP-GCN. We implement and report results for the baseline methods CNN+Position, LSTM+Position and CNN+ATT, and report the results in the original paper for the recent models under comparison. The classification results are shown in Table 2.

From the table, we find that CDT generally ouperforms all models for the different datasets, while having a slight accuracy performance degradation of $0.31$ on the twitter dataset for the TNet model. The difference between TNet and CDT is not really significant. Hence it is fair to conclude that both models are competitive on the Twitter dataset. Even with simple architectures, we find that ASP-BiLSTM and ASP-GCN have competitive performance with the recent models on benchmark datasets. Particularly, ASP-GCN outperforms the models on the Rest14 dataset.

ASP-BiLSTM, ASP-GCN and CDT extract final representations from only the aspect vectors. Based on the performance, it seems as a sufficient technique for the classification task. We believe that the aspect vector is encoded with context and dependency information from the context and structure of the sentence by means of the BiLSTM and the GCN. The BiLSTM and GCN can be regarded as message passing networks, propagating information along a chain of sequence of words(BiLSTM) or along syntactic dependency path(GCN). Due to the fact that relevant information is passed to the aspect words, a simple average pool is all we need to retain information relevant to the classification task. Note that the information propagated in the network is learned therefore only weighed information is encoded within the aspect words.

### 5.5 GCN Performance

We conduct an experiment to demonstrate that the performance of our proposed models, namely CDT and ASP-GCN, depend on the number of layers of the GCN. We perform this experiment on the Rest14 dataset and present the result in Figure 4.

In our experimentation, we find that as we increase the number of layers the accuracy performance increase to an extent. In particular, ASP-GCN increase in model performance over $6$ layers of the GCN. The performance becomes unstable after the $6$-th layer. Since GCN passes information in the local neighborhood of any node, successive operations on the dependency tree allows ASP-GCN to pass information to the furthest node. The problem of overfitting takes effect when the layers rises beyond a threshold, explaining the accuracy curve after the $6$-th layer in the figure. Another important observation is the convergence of accuracy performance of the ASP-BiLSTM and ASP-GCN at the $6$-th layer. Note that ASP-BiLSTM only captures contextual information while ASP-GCN captures dependency information. However, both models converge in performance at the $6$-th layer. Taking advantage of the GCN and the BiLSTM we expect to improve performance, capturing both context and dependencies with respect to the aspect expression. As seen in the accuracy curve of CDT, the GCN integrates dependency informa-

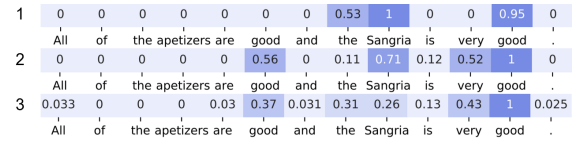


(a) Case Example 1: aspect word is 'Sangria'

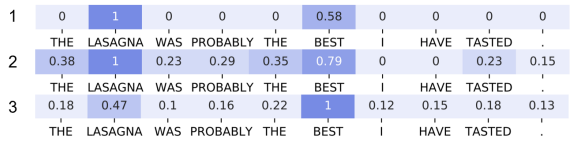


(b) Case Example 2: aspect word is 'LASAGNA'

Figure 3: Word relevance scores with respect to the final embedding of ASP-GCN. Number of layers for ASP-GCN is 1, 2, 3 for row 1, row 2, and row 3 respectively in both examples.

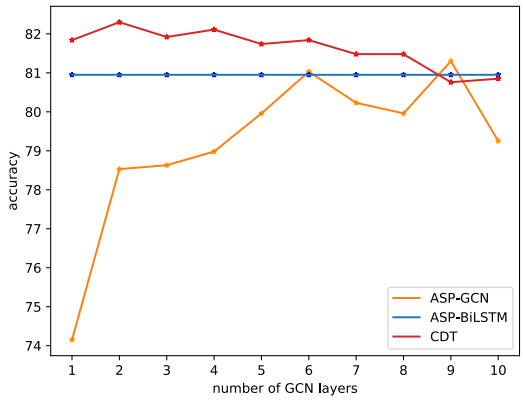


Figure 4: Accuracy curves for ASP-GCN, ASP-BiLSTM and CDT on the Rest14 dataset

tion in the contextualized embeddings to improve accuracy performance over just 2 layers, reducing the number of GCN layers needed.

### 5.6 Mask Experiment

Our primary assumption was that the aspect embeddings learned by our GCN model contains sufficient information necessary for the classification task. Based on this assumption we aggregate only the aspect embeddings using a max pool with the aim to retain most of the information. To verify this assumption, we trace from the input embeddings to the final embedding. We propose a mask method designed to estimate the relevance of a word with respect to the final embedding, and perform mask experiments using ASP-GCN.

The mask method works as follows. First, we follow through the conventional procedure to extract a final embedding $h_s$ for a given sentence $s$ using ASP-GCN. We perform a subsequent run of ASP-GCN on the same sentence $s$ to extract a final embedding, but in this instance we conceal a specific input word $w$. We conceal $w$ by mapping it to the zero vector before applying ASP-GCN on $s$. As a result a final embedding $h_{(s\backslash w)}$ is generated for $s$. If $h_s = h_{(s\backslash w)}$, the word $w$ has no impact on the representation $h_s$. In other words, $h_s$ does not capture $w$ or no information flows from $w$ to $h_s$. To this end, we can define a score function to estimate the relevance of $w$ on $h_s$. We define the score function for $w$ as

$$\gamma(w, s) = m \sum_{i=1}^{d} |h_s^i - h_{(s\backslash w)}^i| \quad (4)$$

where $d$ is the dimension of the final embedding distilled by ASP-GCN, $m$ is a normalization constant which we choose as $m = \max_{w \in s} \gamma(w, s)$. Generally, the final embedding should capture information on opinion words with respect to the target aspect. Hence, we expect to score high values for opinion words. Consider the scores for words shown in Figure 3, we find that $\gamma$ scores high values for opinion words as we increase the number of layers of ASP-GCN, while reducing scores on irrelevant words. Implying that the final embedding captures information from opinions. The results as seen in these case examples convinces us that the final embedding distilled by our model captures relevant information necessary for the classification task.

### 5.7 Case Study

In this section we study the behaviour of ASP-BiLSTM, TNet and CDT on case examples. To this end we present visualizations showing the attention these models place on words. For a good model, we expect the model to attend to words which influence the sentiment inferred on a specific aspect.

From Table 2 and Figure 4, it is clear that GCN complements the BiLSTM to improve model performance. This means that the BiLSTM can identify opinion words within the context with respect to a specific aspect. However, in some complicated contexts, it might perform poorly. But the GCN can build upon BiLSTM to attend to the correct opinion words by leveraging the dependencies among words. Consider the case example shown




Figure 5: Attention visualization for ASP-BiLSTM (1st row), TNet(2nd row) and CDT (3rd row) for the aspect word 'Sangria'

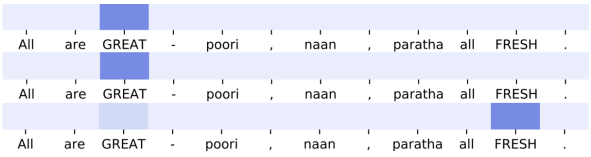


Figure 6: Attention visualization for ASP-BiLSTM (1st row), TNet (2nd row) and CDT (3rd row) for the aspect word 'paratha'

in Figure 5, ASP-BiLSTM was clever to know that the word 'good' is an opinion word with respect to the aspect 'Sangria'. But ASP-BiLSTM failed to identify whether 'good' on the far left is associated to the 'Sangria' or 'good' on the far right is associated to 'Sangria'. Interestingly, we find that the GCN could analyze this further through the dependencies between words to identify that it is the 'good' on the far right. TNet on the other hand measures the association between 'Sangria' and 'good' in both directions to identify the correct 'good'.

In Figure 6, even though the BiLSTM is able to identify the opinion word 'GREAT' which expresses an opinion on the aspect 'parathra', CDT is able to capture the opinion word 'FRESH' which directly expresses the sentiment towards the aspect. However, from the visualization is easily observed that CDT still attends to 'GREAT'. This suggest that the GCN is able to model the importance of the words with respect to the aspect, placing larger weights to words directly expressing the opinion on the aspect. At the same time, TNet misses the opinion word 'FRESH' and places attention to the word 'GREAT' just like ASP-BiLSTM.

In the case example shown in Figure 7, we find that ASP-BiLSTM places small attention on the opinion word 'BEST' which expresses the sentiment on the aspect word 'LASAGNA', while focusing its attention on 'WAS PROBABLY' which is not meaningful alone. Interestingly, CDT builds upon this little information and rely on the dependencies between the words through the dependency tree to learn that 'BEST' is the correct word to attend to. Similar to ASP-BiLSTM, TNet misses the important word 'BEST' and places attention to 'WAS PROBABLY'. This result suggest that TNet heavily depends on the representations modeled by its BiLSTM layer, while CDT considers other information such as the dependencies among words to accurately identify words which expresses opinions on specific aspects.




Figure 7: Attention visualization for ASP-BiLSTM (1st row), TNet (2nd row) and CDT (3rd row) for the aspect word 'LASAGNA'

## 6 Conclusion

Modeling representations for aspect-based sentiment classification generally require capturing informative words which express the sentiment inferred on the target aspect. Leveraging neural networks are highly desirable for representation learning. BiLSTM-based models have been successful to capture contextual information in prior works.

In this paper, we integrate a GCN with a simple BiLSTM model, with the aim to capture structural and contextual information of sentences. We have shown that the GCN successfully performs convolutions on the dependency tree to refine BiLSTM embeddings. Experimental results with visualizations support our argument on the extraction of a final embedding based on only the aspect vectors. In fact, the model we propose is simple and outperforms more complex and recent models tackling the same problem.

## Acknowledgment

This work is supported partly by China 973 program (No. 2015CB358700), by the National Natural Science Foundation of China (No. 61772059, 61421003), by the Beijing Advanced Innovation Center for Big Data and Brain Computing (BDBC), by State Key Laboratory of Software Development Environment (No. SKLSDE-2018ZX-17) and by the Fundamental Research Funds for the Central Universities.

## References


Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. 2017. Recurrent attention network on memory for aspect sentiment analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 452–461.

Jiajun Cheng, Shenglin Zhao, Jiani Zhang, Irwin King, Xin Zhang, and Hui Wang. 2017. Aspect-level sentiment classification with heat (hierarchical attention) network. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 97–106.

Xiaowen Ding, Bing Liu, and Philip S. Yu. 2008. A holistic lexicon-based approach to opinion mining. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 231–240.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 49–54.

Feifan Fan, Yansong Feng, and Dongyan Zhao. 2018a. Multi-grained attention network for aspect-level sentiment classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3433–3442.

Feifan Fan, Yansong Feng, and Dongyan Zhao. 2018b. Multi-grained attention network for aspect-level sentiment classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3433–3442.

Shuqin Gu, Lipeng Zhang, Yuexian Hou, and Yin Song. 2018. A position-aware bidirectional attention network for aspect-level sentiment analysis. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 774–784.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2018a. Effective attention modeling for aspect-level sentiment classification. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 1121–1131.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2018b. Exploiting document knowledge for aspect-level sentiment classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 579–585.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177.

Binxuan Huang and Kathleen Carley. 2018. Parameterized convolutional neural networks for aspect level sentiment classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1091–1096.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Fanshuang Kong, Richong Zhang, Hongyu Guo, Samuel Mensah, Zhiyuan Hu, and Yongyi Mao. 2019. A neural bag-of-words modelling framework for link prediction in knowledge bases with sparse connectivity. In *WWW 2019 : The Web Conference*.

Zeyang Lei, Yujiu Yang, Min Yang, Wei Zhao, Jun Guo, and Yi Liu. 2019. A human-like semantic cognition network for aspect-level sentiment classification. *AAAI*.

Xin Li, Lidong Bing, Wai Lam, and Bei Shi. 2018a. Transformation networks for target-oriented sentiment classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 946–956.

Zheng Li, Ying Wei, Yu Zhang, Xiang Zhang, Xin Li, and Qiang Yang. 2018b. Exploiting coarse-to-fine task transfer for aspect-level sentiment classification. *CoRR*, abs/1811.10999.

Navonil Majumder, Soujanya Poria, Alexander Gelbukh, Md Shad Akhtar, Erik Cambria, and Asif Ekbal. 2018. Iarm: Inter-aspect relation modeling with memory networks in aspect-based sentiment analysis. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3402–3411.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. *arXiv preprint arXiv:1703.04826*.

Lili Mou, Hao Peng, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2015. Discriminative neural sentence modeling by tree-based convolution. *arXiv preprint arXiv:1504.01106*.

Zhifan Ouyang and Jindian Su. 2018. Dependency parsing and attention network for aspect-level sentiment classification. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 391–403.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 27–35.

Ana Maria Popescu and Orena Etzioni. 2005. Extracting product features and opinions from reviews. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 339–346.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer.

Kim Schouten and Flavius Frasincar. 2015. Survey on aspect-level sentiment analysis. *IEEE Transactions on Knowledge and Data Engineering*, 28(3):813–830.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1422–1432.

Bailin Wang and Wei Lu. 2018. Learning latent opinions for aspect-level sentiment classification. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5537–5544.

Yanyan Wang, Qun Chen, Xin Liu, Murtadha H. M. Ahmed, Zhanhuai Li, Wei Pan, and Hailong Liu. 2018. Senhint: A joint framework for aspect-level sentiment analysis by deep neural networks and linguistic hints. In *WWW '18 Companion Proceedings of the The Web Conference 2018*, pages 207–210.

Yequan Wang, Minlie Huang, xiaoyan zhu, and Li Zhao. 2016. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 606–615.

Wei Xue and Tao Li. 2018. Aspect based sentiment analysis with gated convolutional networks. *arXiv preprint arXiv:1805.07043*.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. *empirical methods in natural language processing*, pages 2205–2215.

Shiliang Zheng and Rui Xia. 2018. Left-center-right separated neural network for aspect-based sentiment analysis with rotatory attention. *arXiv preprint arXiv:1802.00892*.

Marinka Zitnik, Monica Agrawal, and Jure Leskovec. 2018. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics*, 34(13):i457–i466.

# Understanding Data Augmentation in Neural Machine Translation: Two Perspectives towards Generalization

**Guanlin Li**[ϵ] [*], **Lemao Liu**[λ], **Guoping Huang**[λ], **Conghui Zhu**[ϵ], **Tiejun Zhao**[ϵ], **Shuming Shi**[λ]
[ϵ]Harbin Institute of Technology, [λ] Tencent AI Lab
{epsilonlee.green}@gmail.com, {chzhu, tjzhao}@hit.edu.cn,
{redmondliu, donkeyhuang, shumingshi}@tencent.com

## Abstract

Many Data Augmentation (DA) methods have been proposed for neural machine translation. Existing works measure the superiority of DA methods in terms of their performance on a specific test set, but we find that some DA methods do not exhibit consistent improvements across translation tasks. Based on the observation, this paper makes an initial attempt to answer a fundamental question: what benefits, which are consistent across different methods and tasks, does DA in general obtain? Inspired by recent theoretic advances in deep learning, the paper understands DA from two perspectives towards the generalization ability of a model: input sensitivity and prediction margin, which are defined independent of specific test set thereby may lead to findings with relatively low variance. Extensive experiments show that relatively consistent benefits across five DA methods and four translation tasks are achieved regarding both perspectives.

## 1 Introduction

Data Augmentation (DA) is a training paradigm that has been proved to be very effective in many modalities (Park et al., 2019; Perez and Wang, 2017; Sennrich et al., 2016a), especially for classification (Perez and Wang, 2017). In structured domain, Neural Machine Translation (NMT) is the frontier of DA research (Sennrich et al., 2016a; Norouzi et al., 2016; Zhang and Zong, 2016; Fadaee et al., 2017; Wang et al., 2018; Zhang et al., 2019; Edunov et al., 2018; Fadaee and Monz, 2018). However, by investigating a variety of DA methods, we find that their test performance across different translation tasks does not exhibit consistent improvement, and this phenomenon can be initially observed in (Wang et al., 2018) as well. The reason might be the evaluation metric on a specific test set when compared to the whole data population, which generates all possible data, has large variance so that leads to the inconsistency. This evaluation dilemma is also recognized and explored by Recht et al. (2018, 2019); Werpachowski et al. (2019), and is especially notorious for language generation tasks (Chaganty et al., 2018; Hashimoto et al., 2019) where the evaluation metrics, e.g. BLEU (Papineni et al., 2001), are extrinsic and heavily relies on the reference provided. Therefore, we ask a fundamental question: what benefits, which are more consistent across different DA methods and translation tasks, can DA in general obtain?

A direct answer to the above question is to use generalization gap (Kawaguchi et al., 2018) defined by the difference between population risk and empirical risk. This measure does not rely on any specific test set, accurately depicts generalization but is intractable to compute. So recently, many theorists have proposed either non-vacuous generalization bound (Dziugaite and Roy, 2017; Zhou et al., 2019) or novel generalization measures (Novak et al., 2018; Bartlett et al., 2017; Neyshabur et al., 2017; Jiang et al., 2019) to roughly reflect the gap. Inspired by them, we propose to understand the benefits of DA from two perspectives: input sensitivity and prediction margin. The proposed underlying two measures are well adapted from Novak et al. (2018) and Bartlett et al. (2017) and can be computed only on the train samples to unveil the consistent benefits of DA. Under a carefully designed fair setting over four different translation tasks, we examine five methods from two main categories of DA and compare them with a model trained without DA. The empirical experiments demonstrate the following findings: a). DA methods exhibit more consistent effects across different translation tasks in terms of both measures. b). DA methods can either allevi-

[*] Work done at Tencent AI Lab.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5689–5695,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

ate input sensitivity or promote prediction margin.

By and large, our main contributions are:

- We make an initial attempt to understand the essence of DA in NMT by investigating its benefits which are relatively consistent across five DA methods and four translation tasks.
- We highlight two perspectives towards generalization to measure the benefits of DA in NMT and study them with carefully designed fair experiments.

## 2 DA Methods in NMT

### 2.1 Training Objective Decomposition

Given the train set $\mathcal{T}$ the baseline NMT model $p_\theta(\mathrm{y}|\mathrm{x})$ without using DA is trained under the empirical data distribution $\hat{p}(X, Y|\mathcal{T})$ through maximum likelihood estimation:

$$J_{\mathrm{MLE}} = \mathbb{E}_{\mathrm{x,y}\sim\hat{p}(X,Y|\mathcal{T})}[\log p_\theta(\mathrm{y}|\mathrm{x})], \quad (1)$$

where $\hat{p}$ is a mixture of Dirac distribution concentrated around each training instance with uniform mixture coefficients $(1/|\mathcal{T}|)$. Then we define the augmentation (AUG) model as a conditional distribution over the train set, $q(X, Y|\mathcal{T})$. [1] Under the AUG model, the training objective becomes:

$$J_{\mathrm{AUG}} = \mathbb{E}_{\mathrm{x,y}\sim q(X,Y|\mathcal{T})}[\log p_\theta(\mathrm{y}|\mathrm{x})]. \quad (2)$$

More realistically, for any DA method in any training run, we can collect the augmented instances to form a set $\mathcal{A}$ distinguishing $\mathcal{T}$, when considering the curriculum of mixing $\mathcal{A}$ with the original train $\mathcal{T}$. Since we would like to derive a conceptual framework that reflects this form of importance weighting, we further decompose AUG model into a linear interpolation $(\alpha)$ of $\hat{p}(X, Y|\mathcal{T})$ and an augmentation distribution $q_{\mathrm{AUG}}(X, Y|\mathcal{T})$:

$$\alpha \cdot \hat{p}(X, Y|\mathcal{T}) + (1 - \alpha) \cdot q_{\mathrm{AUG}}(X, Y|\mathcal{T}), \quad (3)$$

where $\alpha$ controls the mixture ratio within a batch during SGD training. The ratio has been founded as an important factor influencing final performance (Sennrich et al., 2016a; Fadaee et al., 2017; Edunov et al., 2018; Fadaee and Monz, 2018).

[1] In the paper, we do not consider using monolingual data for DA thus conditioning only on bilingual data since this will further bring monolingual data selection discussed in Fadaee and Monz (2018) as a factor to influence the performance of different DA methods; we leave this factor for future study.

| Method | Fr⇒En | En⇒Fr | Zh⇒En | En⇒De |
|---|---|---|---|---|
| Baseline | 38.38 (5) | 38.88 (6) | 17.25 (6) | 26.19 (4) |
| RAML | *+0.22* (3) | *+0.67* (3) | *+0.23* (4) | *-0.16* (6) |
| SO | *+0.01* (4) | *+0.62* (4) | *+0.02* (5) | *-0.15* (5) |
| ST | *-0.13* (6) | *+0.46* (5) | *+1.51* (2) | *+0.83* (2) |
| TA | *+0.62* (2) | *+1.13* (1) | *+2.41* (1) | *+1.01* (1) |
| BT | *+0.82* (1) | *+0.99* (2) | *+1.06* (3) | *+0.39* (3) |

Table 1: Main BLEU results (CTC=0.62).

**Key factors** Through Eq. 3, we can identify two key factors for conducting fair experiments: a) the number of SGD updates on every original training instance means how much the model learns from $\mathcal{T}$; b) the mixture ratio means how much the model learns from $\mathcal{A}$ online, with which together balance the learning of the translation knowledge.

### 2.2 Settings and Main Performance

**Settings** By carefully controlling the above two factors, we conduct fair and extensive experiments with Transformer (Vaswani et al., 2017) on four translation tasks for five DA methods. Fairseq (Ott et al., 2019) is used as our codebase. We use standard benchmarks IWSLT17 En-Fr, WMT19 Zh-En, WMT19 En-De, where we train both translation directions on the IWSLT corpus. The five DA methods are briefly summarized as follows:

- RAML: reward-augmented maximum likelihood training, which augment the target-side with a sampling distribution $P(Y|Y^*)$ concentrated around $Y^*$ (Norouzi et al., 2016).
- Switchout (SO): similar to RAML, but also adds the some kind of augmentation to the source-side (Wang et al., 2018).
- Self-training (ST): fix the source-side, uses an forward NMT model to generate the target-side (Zhang and Zong, 2016).
- Target-agree (TA): similar to ST, but uses a forward NMT model with right-to-left decoder (Zhang et al., 2019).
- Back-translation (BT): fix the target-side, uses an backward NMT model to generate the source-side (Sennrich et al., 2016a).

The implementation of RAML and SO are borrowed from the Appx. of Wang et al. (2018). [2]

[2] We categorize and analyze how we choose or train DA methods with a generative formulation in Appx. A and B.

| Method | Fr⇒En | En⇒Fr | Zh⇒En | En⇒De |
|---|---|---|---|---|
| Baseline | 0.565 (6) | 0.623 (4) | 0.422 (6) | 0.682 (5) |
| RAML | *-0.056* (2) | *+0.003* (5) | *-0.008* (4) | *-0.003* (4) |
| SO | *-0.082* (1) | *-0.099* (1) | *-0.053* (2) | *-0.143* (1) |
| ST | *-0.034* (3) | *-0.009* (3) | *-0.054* (1) | *-0.116* (2) |
| TA | *-0.023* (5) | *-0.010* (2) | *-0.043* (3) | *-0.013* (3) |
| BT | *-0.026* (4) | *+0.035* (6) | *-0.007* (5) | *+0.232* (6) |

Table 2: Sensitivity measure (CTC=0.72).

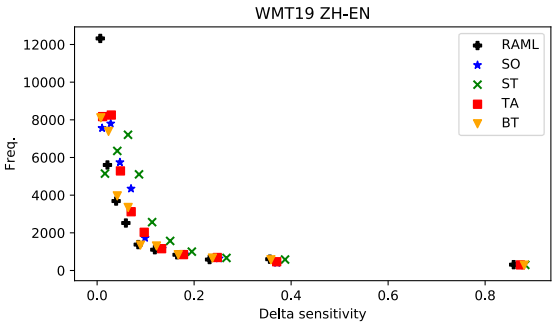


Figure 1: $\Delta$ sensitivity binned avg. token freq. statistics. Each point represents a bin from which we compute the token level average $\Delta$ sensitivity between that DA method and the baseline and the token level average frequency as its x and y coordinate value.

**CTC** Table 1 shows the main BLEU results of different methods on the test set. However, we cannot identify the best DA method because their rankings across the four translation tasks vary a bit. To measure the degree of consistency, we use a correlation measure called Kendall's coefficient of concordance (Kendall and Smith, 1939; Mazurek, 2011) to evaluate the correlation of the rankings produced on the four translation tasks (appx. C). The value shows strong consistency (correlation) of different rankings when it is close to 1. We call the correlation value Cross-Task Consistency measure or CTC. The CTC for the BLEU measure is 0.62, which is of weak consistency. This phenomenon might be a result of the intrinsic nature of using a single specific test as a substitute of the whole data population for evaluation. In the next section, we introduce two measures that are more consistent (with close-to-1 CTC value). They in some extent reflect the model generalization and are easy-to-compute as well.

## 3 Two Measures Towards Generalization

We attempt to understand the benefits that DA can obtain through the quantification of input sensitivity and prediction margin. The two measures are adapted from Novak et al. (2018) and Bartlett et al. (2017); Neyshabur et al. (2017); Jiang et al. (2019). They have been proved through massive experiments to be correlated with model generalization. Our main purpose here is to utilize them to unveil the consistency property (measured by CTC) of DA across different methods and translation tasks. The next two subsections define the two measures and report their statistics on subsamples of the train set respectively.

### 3.1 Input Sensitivity

Input sensitivity is the sensitivity of the loss computed from the model towards a minor change of input representation. Given a point of interest x, the original form in Novak et al. (2018) is defined as the expected Jacobian norm of the loss vector $\log p_\theta(\cdot|\mathrm{x})$ and $p_\theta$ is a softmax classifier:

$$\mathbb{E}_{\mathrm{x}}||J(\mathrm{x})||_F, \tag{4}$$

where $J(\mathrm{x}) = \partial \log p_\theta(\cdot|\mathrm{x})/\partial \mathrm{x}^T$, and $||\cdot||_F$ the Frobenius or L2 norm of the matrix. The paper also suggests a more predictive quantity of the generalization ability. That is to take advantage of the label $y$ of x and only compute the L2 norm of a slice of the Jacobian matrix indexed by the label. We adopt the later measure which is the gradient norm of the loss scalar indexed by y to x:

$$\mathbb{E}_{\mathrm{x}}||J(\mathrm{x})_y||_F. \tag{5}$$

If $\mathrm{x} \in \mathbb{R}^d$ lies in a space with differential structure, we can apply Eq. 5 directly. But in NMT the naive representation of an instance $(\mathrm{x}, \mathrm{y})$ is the token index given by the vocabulary, so we cannot compute the gradient of the loss with respect to x. We follow Sundararajan et al. (2017) and use the result of x after embedding lookup as its learned representation, denoted as $\mathrm{Emb}(\mathrm{x}) \in \mathbb{R}^{L_\mathrm{x} \times d}$ where $L_\mathrm{x}$ is the length of the input and $d$ the size of the embedding. By regarding the translation model $p_\theta(\mathrm{y}|\mathrm{x})$ as a function that decomposes at each step of y given $\mathrm{Emb}(\mathrm{x})$ as input to get a scalar average log likelihood, denoted as $\mathcal{L}_{\mathrm{x,y}} = \frac{1}{L_\mathrm{y}} \sum_t \log p_\theta(\mathrm{y}_t|\mathrm{y}_{<t}, \mathrm{x})$. Moreover, initial experiments on just using the single original $\mathrm{x}_i$ to evaluate the gradient will still result in inconsistency, due to the non-equivalence of the localness concept compared with the continuous setting, i.e. for language input, the localness is between discrete inputs in the neighbor of $\mathrm{x}_i$. So we

evaluate the sensitivity of $\mathrm{x}_i$ by averaging gradient norms over its $k$ nearest neighbor $\mathrm{x}_{i(j)} \in \mathrm{kNN}[\mathrm{x}_i]$ through cosine similarity between word embeddings. We set $k$ to 5 in our experiment to guarantee words in the $k$ nearest neighbor has similar semantic meaning. Formally, we define the input sensitivity of an NMT model as:

$$\mathbb{E}_{(\mathrm{x},\mathrm{y})} \frac{1}{L_\mathrm{x}} \sum_i \frac{1}{k} \sum_{\mathrm{x}_{i(j)} \in \mathrm{kNN}[\mathrm{x}_i]} \left|\left| \frac{\partial \mathcal{L}_{\mathrm{x},\mathrm{y}}}{\partial \mathrm{Emb}(\mathrm{x})_{i(j)}} \right|\right|_F, \tag{6}$$

where the $\mathrm{Emb}(\mathrm{x})_i$ is the embedding lookup of the $i^{th}$ token index, so we compute the average token-wise gradient of each instance.

We use subsamples of the train set to approximately compute the expectation in Eq. 6 and the overall statistics are shown in Table 2. Similar to Table 1, the DA methods are shown in their $\Delta$ value respect to the baseline. A first thing to notice is that the ranking is more steady across tasks (CTC=0.72). It also shows that for input x, DA in general can reduce the gradient norm of the prediction loss on $\mathrm{Emb}(\mathrm{x})_i$, which shows that DA can obtain more stable model towards data corruption.

To further understand what effect DA in general has on each input token type, we compute the $\Delta$ sensitivity between the baseline and one DA method on the same token type and sort them according to the $\Delta$ with positive value (which means DA reduces the sensitivity of that token type). We then divide the sorted types into ten bins and compute the average token type frequency of that bin. As shown in Figure 1, DA in general, improves the sensitivity of token types with relatively low frequency more than those with high frequency, thus may somehow improve the translation quality of low frequency token types.

### 3.2 Prediction Margin

Margin is a classic concept in support vector machine (Vapnik, 2013), which is defined as the geometric distance between the support vectors and the decision boundary. Larger margin implies better generalization. In nonlinear case, it reflects the distance of a correctly classified input representation with class $i$ to move towards the decision boundary between $i$ and any other class $j$ (Jiang et al., 2019). However, since the decision boundary does not have analytical form due to nonlinearity, computing the geometric distance is intractable. In our setting we regard NMT model as doing step-wise classification with $\mathrm{z} = (\mathrm{x}, \mathrm{y}_{<t})$ as input feature and $\mathrm{y}_t$ as the label. In Bartlett et al. (2017), the original definition of the margin of correctly predicted input is:

| Method | Fr⇒En | En⇒Fr | Zh⇒En | En⇒De |
|---|---|---|---|---|
| Baseline | 0.797 (4) | 0.592 (4) | 0.756 (4) | 0.679 (4) |
| RAML | *-0.028* (6) | *-0.063* (6) | *-0.024* (6) | *-0.006* (5) |
| SO | *-0.027* (5) | *-0.060* (5) | *-0.022* (5) | *-0.009* (6) |
| ST | *+0.046* (1) | *+0.036* (1) | *+0.035* (1) | *+0.044* (1) |
| TA | *+0.037* (2) | *+0.019* (2) | *+0.025* (2) | *+0.043* (2) |
| BT | *+0.017* (3) | *+0.015* (3) | *+0.004* (3) | *+0.016* (3) |

Table 3: Margin measure (CTC=0.98).

$$\frac{p_\theta(\mathrm{y}_t|\mathrm{z}) - max_{v' \neq \mathrm{y}_t} p_\theta(v'|\mathrm{z})}{\mathcal{R} \cdot ||\mathrm{x}||_2 / N}, \tag{7}$$

where $\mathrm{y}_t$ is the ground-truth label, $v'$ another class type, $\mathcal{R}$ the spectral complexity of the model and $N$ the number of training instance in the train subsamples for computing their margins. We simplify Eq. 7 to only consider the numerator. The reasons are: a) under the same model architecture, $\mathcal{R}$s are very close across different DA methods; b) we can omit $||x||_2/N$ since it remains unchanged as well. In this way, we can map every $\mathrm{z}, \mathrm{y}_t$ to a margin with label type $\mathrm{y}_t = v$, where $v \in \mathcal{V}_{tgt}$:

$$m^v_{\mathrm{z},\mathrm{y}_t} := p_\theta(\mathrm{y}_t|\mathrm{z}) - max_{v' \neq \mathrm{y}_t} p_\theta(v'|\mathrm{z}). \tag{8}$$

So for every target token type $v$ we can collect a set of margins $\{m^v_{\mathrm{z},\mathrm{y}_t}\}$, and the margin sets of all token types are combined as the total margin set $\cup_v \{m^v_{\mathrm{z},\mathrm{y}_t}\}$. Following Neyshabur et al. (2017), we do not compute the minimum margin of the total set which can be highly sensitive to outliers. Instead, if the total set has cardinality $N'$, we obtain the $\epsilon N'$-th smallest margin from the set as the overall prediction margin, with a tolerant coefficient $\epsilon \in [0, 0.1]$ ($\epsilon$ is set to 0.001). [3] We can also obtain token-wise prediction margin from $\{m^v_{\mathrm{z},\mathrm{y}_t}\}$, which is the prediction margin of a specific token type $v$.

The overall prediction margins are listed in Table 3. The relative rank is highly consistent across the four translation tasks (CTC=0.98). Although RAML and SO seem to be inferior to the baseline, other DA methods improve the margin in general. We give a possible explanation for this in the next subsection. Similar to the previous subsection, we

[3] Note that, we have tried different tolerant coefficents and find they do not effect the final rankings of DA methods.

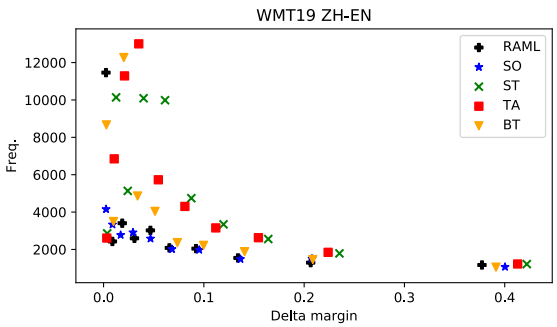


Figure 2: $\Delta$ margin binned avg. token freq. statistics.

also report the average token type frequency of each margin binned token groups in Figure 2 and find that DA, in general, brings larger margin improvement over low frequency tokens.

### 3.3 Discussion

**Why use these two measures?** We have conducted a relatively complete survey of the recent measures towards measuring generalization ability proposed by the deep learning community, such as model complexity (Zhang et al., 2016; Neyshabur et al., 2017), flatness (Dinh et al., 2017), stiffness (Fort et al., 2019) and second order Hessian of the input or the number of linear regions in hiddens (Novak et al., 2018; Montufar et al., 2014). Some of those measures have complex definition such as linear regions, others are very expensive to compute for models as large as Transformer such as Hessian and stiffness. However, we compute weight norm with different forms proposed in Neyshabur et al. (2017) and find no regularity which suggests that the complexity measure through norms for networks architectures like Transformer or convolutional/recurrent neural networks might be very different from simple feed-forward ones which might be still an open problem in theoretic deep learning community. As a matter of fact, due to computational easiness and large-scale empirical evidence, we choose sensitivity and margin the measures.

**Why no absolute consistency between the two measures?** In Section 3.1 and 3.2, the two measures do not show well consistency between them: under the margin based measure, RAML and SO do not exhibit superiority over the baseline like they do under the sensitivity measure. One reason might be: despite the measures are empirically proved to reflect generalization, they are only one view towards generalization respectively. Specifically, in recent generalization theory (Novak et al., 2018; Bartlett et al., 2017), the measures are evaluated between models with extremely evident difference in generalization ability (measured by test performance difference), for example, between models trained with random labels and true labels. Instead, our comparison is among models with similar capacity and are well-trained, which rises challenge for us to get very consistent statistics through a single view. This may inspire us to combine multiple views of model training to design better measures with stronger correlation.

## 4 Conclusion and Future Work

This paper aims at delivering relatively consistent benefit measures of DA due to the phenomenon of inconsistant BLEU improvement across translation tasks. To our expect, the proposed two measures exhibit relative consistency (especially prediction margin) on five DA methods across four translation tasks, which demonstrate that DA can benefit model with improved sensitivity or prediction margin especially for low frequency words.

However, the problem of intrinsic evaluation or better understanding of the unreasonable effective of DA should just be a start. DA is a trade-off between noise vs. knowledge injection, so it could be a nice theoretic direction to think about DA under statistical query model (Kearns, 1998) with translation between formal languages (ws-, 2019). This could inspire another essential question: what is the intrinsic properties of the augmented data (Branchaud-Charron et al., 2019) that matter in discrete domain. Applications like active data selection (Coleman et al., 2019) guided with margin or sensitivity can be derived. In general, understanding NMT model's behavior (not only with DAs) beyond BLEU (Neubig et al., 2019) should be taken seriously, e.g. to design a bevavior suite like Osband et al. (2019) is most valuable.

## Acknowledgements

The authors are grateful to all the anonymous reviewers for pointing out the topic of intrinsic evaluation of DA methods to summarize our work as well as proposing several critical issues regarding limitations of the proposed measures. We thank Xintong, Zhirui for valuable discussions; Xintong, Chaoqun, Yong Jiang for proof-reading. Tiejun Zhao is supported by National Key R&D Program of China (Project No. 2017YFB1002102).

## References


2019. *Proceedings of the Workshop on Deep Learning and Formal Languages: Building Bridges*. Association for Computational Linguistics, Florence.

Peter L. Bartlett, Dylan J. Foster, and Matus Telgarsky. 2017. Spectrally-normalized margin bounds for neural networks. In *NIPS*.

Frederic Branchaud-Charron, Andrew Achkar, and Pierre-Marc Jodoin. 2019. Spectral metric for dataset complexity assessment. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3215–3224.

A. Chaganty, S. Mussmann, and P. Liang. 2018. The price of debiasing automatic metrics in natural language evaluation. In *Association for Computational Linguistics (ACL)*.

Yong Cheng, Wei Xu, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Semi-supervised learning for neural machine translation. *CoRR*, abs/1606.04596.

Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. 2019. Selection via proxy: Efficient data selection for deep learning. *ArXiv*, abs/1906.11829.

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. 2017. Sharp minima can generalize for deep nets. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1019–1028. JMLR. org.

Gintare Karolina Dziugaite and Daniel M. Roy. 2017. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *CoRR*, abs/1703.11008.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *EMNLP*.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. Data augmentation for low-resource neural machine translation. In *ACL*.

Marzieh Fadaee and Christof Monz. 2018. Back-translation sampling by targeting difficult words in neural machine translation. In *EMNLP*.

Stanislav Fort, Paweł Krzysztof Nowak, and Srini Narayanan. 2019. Stiffness: A new perspective on generalization in neural networks. *arXiv preprint arXiv:1901.09491*.

Tommaso Furlanello, Zachary Chase Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. 2018. Born again neural networks. In *ICML*.

T. Hashimoto, H. Zhang, and P. Liang. 2019. Unifying human and statistical evaluation for natural language generation. In *North American Association for Computational Linguistics (NAACL)*.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

Yiding Jiang, Dilip Krishnan, Hossein Mobahi, and Samy Bengio. 2019. Predicting the generalization gap in deep networks with margin distributions. *CoRR*, abs/1810.00113.

Kenji Kawaguchi, Leslie Pack Kaelbling, and Yoshua Bengio. 2018. Generalization in deep learning. *CoRR*, abs/1710.05468.

Michael Kearns. 1998. Efficient noise-tolerant learning from statistical queries. *J. ACM*, 45(6):983–1006.

Maurice G Kendall and B Babington Smith. 1939. The problem of m rankings. *Annals of mathematical statistics*.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *EMNLP*.

Jiří Mazurek. 2011. Evaluation of ranking similarity in ordinal ranking problems. *Acta academica karviniensia*, 2:119–128.

Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. 2014. On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, pages 2924–2932.

Graham Neubig, Zi-Yi Dou, Junjie Hu, Paul Michel, Danish Pruthi, and Xinyi Wang. 2019. compare-mt: A tool for holistic comparison of language generation systems. In *Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL) Demo Track*, Minneapolis, USA.

Behnam Neyshabur, Srinadh Bhojanapalli, David A. McAllester, and Nathan Srebro. 2017. Exploring generalization in deep learning. In *NIPS*.

Mohammad Norouzi, Samy Bengio, Zhifeng Chen, Navdeep Jaitly, Mike Schuster, Yonghui Wu, and Dale Schuurmans. 2016. Reward augmented maximum likelihood for neural structured prediction. In *NIPS*.

Roman Novak, Yasaman Bahri, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. 2018. Sensitivity and generalization in neural networks: an empirical study. *CoRR*, abs/1802.08760.

Ian Osband, Yotam Doron, Matteo Hessel, John Aslanides, Eren Sezener, Andre Saraiva, Katrina McKinney, Tor Lattimore, Csaba Szepezvári, Satinder Singh, Benjamin Van Roy, Richard S. Sutton, David Silver, and Hado van Hasselt. 2019. Behaviour suite for reinforcement learning. *ArXiv*, abs/1908.03568.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le. 2019. Specaugment: A simple data augmentation method for automatic speech recognition. *arXiv preprint arXiv:1904.08779*.

Luis Perez and Jason Wang. 2017. The effectiveness of data augmentation in image classification using deep learning. *CoRR*, abs/1712.04621.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. 2018. Do cifar-10 classifiers generalize to cifar-10? *CoRR*, abs/1806.00451.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. 2019. Do imagenet classifiers generalize to imagenet? *CoRR*, abs/1902.10811.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. *CoRR*, abs/1511.06709.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. *CoRR*, abs/1508.07909.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *ICML*.

Vladimir Vapnik. 2013. *The nature of statistical learning theory*. Springer science & business media.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Stefan Wager, Sida I. Wang, and Percy S. Liang. 2013. Dropout training as adaptive regularization. In *NIPS*.

Sida I. Wang, Mengqiu Wang, Stefan Wager, Percy S. Liang, and Christopher D. Manning. 2013. Feature noising for log-linear structured prediction. In *EMNLP*.

Xinyi Wang, Hieu Quang Pham, Zihang Dai, and Graham Neubig. 2018. Switchout: an efficient data augmentation algorithm for neural machine translation. In *EMNLP*.

Roman Werpachowski, András György, and Csaba Szepesvári. 2019. Detecting overfitting via adversarial examples. *CoRR*, abs/1903.02380.

Yingce Xia, Di He, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In *NIPS*.

Ziang Xie, Sida I. Wang, Jiwei Li, Daniel Lévy, Aiming Nie, Daniel Jurafsky, and Andrew Y. Ng. 2017. Data noising as smoothing in neural network language models. *CoRR*, abs/1703.02573.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2016. Understanding deep learning requires rethinking generalization. *ArXiv*, abs/1611.03530.

Jiajun Zhang and Chengqing Zong. 2016. Exploiting source-side monolingual data in neural machine translation. In *EMNLP*.

Zhirui Zhang, Shujie Liu, Mu Li, Ming Zhou, and Enhong Chen. 2018. Joint training for neural machine translation models with monolingual data. In *AAAI*.

Zhirui Zhang, Shuangzhi Wu, Shujie Liu, Mu Li, Ming Zhou, and Enhong Chen. 2019. Regularizing neural machine translation by target-bidirectional agreement. *CoRR*, abs/1808.04064.

Wenda Zhou, Victor Veitch, Morgane Austern, Ryan P. Adams, and Peter Orbanz. 2019. Non-vacuous generalization bounds at the imagenet scale: a pac-bayesian compression approach. In *ICLR 2019*.

# Simple and Effective Noisy Channel Modeling for Neural Machine Translation

**Kyra Yee**[△] **Yann N. Dauphin**[▽†] **Michael Auli**[△]
[△]Facebook AI Research
[▽]Google Brain

## Abstract

Previous work on neural noisy channel modeling relied on latent variable models that incrementally process the source and target sentence. This makes decoding decisions based on partial source prefixes even though the full source is available. We pursue an alternative approach based on standard sequence to sequence models which utilize the entire source. These models perform remarkably well as channel models, even though they have neither been trained on, nor designed to factor over incomplete target sentences. Experiments with neural language models trained on billions of words show that noisy channel models can outperform a direct model by up to 3.2 BLEU on WMT'17 German-English translation. We evaluate on four language-pairs and our channel models consistently outperform strong alternatives such right-to-left reranking models and ensembles of direct models.[1]

## 1 Introduction

Sequence to sequence models directly estimate the posterior probability of a target sequence $y$ given a source sequence $x$ (Sutskever et al., 2014; Bahdanau et al., 2015; Gehring et al., 2017; Vaswani et al., 2017) and can be trained with pairs of source and target sequences. Unpaired sequences can be leveraged by data augmentation schemes such as back-translation, but direct models cannot naturally take advantage of unpaired data (Sennrich et al., 2016a; Edunov et al., 2018a).

The noisy channel approach is an alternative which is used in statistical machine translation (Brown et al., 1993; Koehn et al., 2003). It entails a channel model probability $p(x|y)$ that operates in the reverse direction as well as a language model probability $p(y)$. The language model can be estimated on unpaired data and can take a separate form to the channel model. Noisy channel modeling mitigates explaining away effects that result in the source being ignored for highly likely output prefixes (Klein and Manning, 2001).

Previous work on neural noisy channel modeling relied on a complex latent variable model that incrementally processes source and target prefixes (Yu et al., 2017). This trades efficiency for accuracy because their model performs significantly less well than a vanilla sequence to sequence model. For languages with similar word order, it can be sufficient to predict the first target token based on a short source prefix, but for languages where word order differs significantly, we may need to take the entire source sentence into account to make a decision.

In this paper, we show that a standard sequence to sequence model is an effective parameterization of the channel probability. We train the model on full sentences and apply it to score the source given an incomplete target sentence. This bases decoding decisions on scoring the entire source sequence and it is very simple and effective (§2). We analyze this approach for various target prefix sizes and find that it is most accurate for large target context sizes. Our simple noisy channel approach consistently outperforms strong baselines such as online ensembles and left-to-right reranking setups (§3).

## 2 Approach

The noisy channel approach applies Bayes' rule to model $p(y|x) = p(x|y)p(y)/p(x)$, that is, the channel model $p(x|y)$ operating from the target to the source and a language model $p(y)$. We do not model $p(x)$ since it is constant for all $y$. We com-

† Work done while at Facebook AI Research.

[1]We release code and pre-trained models at https://github.com/pytorch/fairseq



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5696–5701, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

pute the channel model probabilities as follows:

$$p(x|y) = \sum_{j}^{|x|} \log p(x_j|x_0, x_1, ...x_{j-1}, y)$$

We refer to $p(y|x)$ as the direct model. A critical choice in our approach is to model $p(x|y)$ with a standard Transformer architecture (Vaswani et al., 2017) as opposed to a model which factors over target prefixes (Yu et al., 2017). This setup presents a clear train/test mismatch: we train $p(x|y)$ on complete sentence-pairs and perform inference with incomplete target prefixes of varying size $k$, i.e., $p(x|y_1, \ldots, y_k)$. However, we find standard sequence to sequence models to be very robust to this mismatch (§3).

**Decoding.** To generate $y$ given $x$ with the channel model, we wish to compute $\arg\max_y \log p(x|y) + \log p(y)$. However, naïve decoding in this way is computationally expensive because the channel model $p(x|y)$ is conditional on each candidate target prefix. For the direct model, it is sufficient to perform a single forward pass over the network parameterizing $p(y|x)$ to obtain output word probabilities for the entire vocabulary. However, the channel model requires separate forward passes for each vocabulary word.

**Approximation.** To mitigate this issue, we perform a two-step beam search where the direct model pre-prunes the vocabulary (Yu et al., 2017). For beam size $k_1$, and for each beam, we collect $k_2$ possible next word extensions according to the direct model. Next, we score the resulting $k_1 \times k_2$ partial candidates with the channel model and then prune this set to size $k_1$. Other approaches to pre-pruning may be equally beneficial but we adopt this approach for simplicity.[2] A downside of online decoding with the channel model approach is the high computational overhead: we need to invoke the channel model $k_1 \times k_2$ times compared to just $k_1$ times for the direct model.

**Complexity.** The model of Yu et al. (2017) factorizes over source and target prefixes. During decoding, their model alternates between incrementally reading the target prefix and scoring a source prefix, resulting in a runtime of $O(n + m)$, where $n$ and $m$ are the source and target lengths, respectively. In comparison, our approach repeatedly scores the entire source for each target prefix, resulting in $O(mn)$ runtime. Although our approach has greater time complexity, the practical difference of scoring the tokens of a single source sentence instead of just one token is likely to be negligible on modern GPUs since all source tokens can be scored in parallel. Inference is mostly slowed down by the autoregressive nature of decoding. Scoring the entire source enables capturing more dependencies between the source and target, since the beginning of the target must explain the entire source, not just the beginning. This is especially important when the word order between the source and target language varies considerably, and likely accounts for the lower performance of the direct model of Yu et al. (2017) in comparison to a standard seq2seq model.

**Model combinaton.** Since the direct model needs to be evaluated for pre-pruning, we also include these probabilities in making decoding decisions. We use the following linear combination of the channel model, the language model and the direct model for decoding:

$$\frac{1}{t} \log p(y|x) + \frac{\lambda_1}{s} \Big( \log p(x|y) + \log p(y) \Big) \quad (1)$$

where $t$ is the length of the target prefix $y$, $s$ is the source sentence length and $\lambda$ is a tunable weight. Initially, we used separate weights for $p(x|y)$ and $p(y)$ but we found that a single weight resulted in the same accuracy and was easier to tune. Scaling by $t$ and $s$ makes the scores of the direct and channel model comparable to each other throughout decoding. In n-best re-ranking, we have complete target sentences which are of roughly equal length and therefore do not use per word scores.[3]

## 3 Experiments

**Datasets.** For English-German (En-De) we train on WMT'17 data, validate on news2016 and test on news2017. For reranking, we train models with a 40K joint byte pair encoding vocabulary (BPE; Sennrich et al. 2016b). To be able to use the language model during online decoding, we use the vocabulary of the langauge model on the target side. For the source vocabulary, we learn a 40K

[2] Vocabulary selection can prune the vocabulary to a few hundred types with no loss in accuracy (L'Hostis et al., 2016).

[3] Reranking experiments are also based on separate tunable weights for the LM and the channel model. However, results are comparable to a single weight.

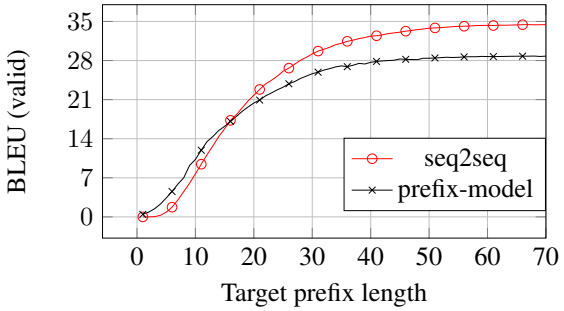


Figure 1: Comparison of two channel models: a standard seq2seq model trained on full sentence-pairs and a model trained on all possible target prefixes with the full source (prefix-model). We measure accuracy of predicting the full source with increasing target prefixes for both models. Results are on news2016.

byte pair encoding on the source portion of the bitext; we find using LM and bitext vocabularies give similar accuracy. For Chinese-English (Zh-En), we pre-process WMT'17 data following Hassan et al. (2018), we develop on dev2017 and test on news2017. For IWSLT'14 De-En we follow the setup of Edunov et al. (2018b) and measure case-sensitive tokenized BLEU. For WMT De-En, En-De and Zh-En we measure detokenized BLEU (Post, 2018).

**Language Models.** We train two big Transformer language models with 12 blocks (Baevski and Auli, 2018): one on the German newscrawl data distributed by WMT'18 comprising 260M sentences and another one on the English newscrawl data comprising 193M sentences. Both use a BPE vocabulary of 32K types. We train on 32 Nvidia V100 GPUs with 16-bit floating point operations (Ott et al., 2018) and training took four days.

**Sequence to Sequence Model training.** For En-De, De-En, Zh-En we use big Transformers and for IWSLT De-En a base Transformer (Vaswani et al., 2017) as implemented in fairseq (Ott et al., 2019). For online decoding experiments, we do not share encoder and decoder embeddings since the source and target vocabularies were learned separately. We report average accuracy of three random initializations of a each configuration. We generally use $k_1 = 5$ and $k_2 = 10$. We tune $\lambda_1$, and a length penalty on the validation set.

### 3.1 Simple Channel Model

We first motivate a standard sequence to sequence model to parameterize $p(x|y)$ as opposed to a model that is trained to operate over prefixes. We train Transformer models to translate from the target to the source (En-De) and compare two variants: i) a standard sequence to sequence model trained to predict full source sentences based on full targets (seq2seq). ii) a model trained to predict the full source based on a prefix of the target; we train on all possible prefixes of a target sentence, each paired with the full source (prefix-model).

Figure 1 shows that the prefix-model performs slightly better for short target prefixes but this advantage disappears after 15 tokens. On full target sentences seq2seq outperforms the prefix model by 5.7 BLEU. This is likely because the prefix-model needs to learn how to process both long and short prefixes which results in lower accuracy. The lower performance on long prefixes is even more problematic considering our subsequent finding that channel models perform over-proportionally well on long target prefixes (§3.4). The seq2seq model has not been trained to process incomplete targets but empirically it provides a simple and effective parameterization of $p(x|y)$.

### 3.2 Effect of Scoring the Entire Source Given Partial Target Prefixes

The model of (Yu et al., 2017) uses a latent variable to incrementally score the source for prefixes of the target. Although this results in a faster run time, the model makes decoding decisions based on source prefixes even though the full source is available. In order to quantify the benefit of scoring the entire source instead of a learned prefix length, we simulate different fractions of the source and target in an n-best list reranking setup.

The n-best list is generated by the direct model and we re-rank the list in setups where we only have a fraction of the candidate hypothesis and the source sentence. We report BLEU of the selected full candidate hypothesis.

Figure 2 shows that for any given fraction of the target, scoring the entire source (src 1) has better or comparable performance than all other source prefix lengths. It is therefore beneficial to have a channel model that scores the entire source sentence.

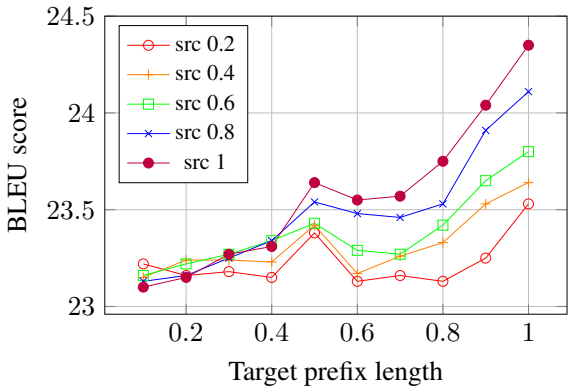


Figure 2: For any given target prefix fraction, scoring the entire source has the best or comparable performance compared to other source prefixes. We show detokenized BLEU on the dev set of WMT17 Zh-En with beam 50.

| | news2016 | news2017 |
|---|---|---|
| DIR | 39.0 | 34.3 |
| DIR ENS | 40.0 | 35.3 |
| DIR+LM | 39.8 | 35.2 |
| CH+DIR+LM | 41.0 | 36.2 |
| - per word scores | 40.0 | 35.1 |

Table 1: Online decoding accuracy for a direct model (DIR), ensembling two direct models (DIR ENS) and the channel approach (CH+DIR+LM). We ablate the impact of using per word scores. Results are on WMT De-En. Table 4 in the appendix shows standard deviations.

### 3.3 Online Decoding

Next, we evaluate online decoding with a noisy channel setup compared to just a direct model (DIR) as well as an ensemble of two direct models (DIR ENS). Table 1 shows that adding a language model to DIR (DIR+LM) gives a good improvement (Gulcehre et al., 2015) over a single direct model but ensembling two direct models is slightly more effective (DIR ENS). The noisy channel approach (CH+DIR+LM) improves by 1.9 BLEU over DIR on news2017 and by 0.9 BLEU over the ensemble. Without per word scores, accuracy drops because the direct model and the channel model are not balanced and their weight shifts throughout decoding. Our simple approach outperforms strong online ensembles which illustrates the advantage over incremental architectures (Yu et al., 2017) that do not match vanilla seq2seq models by themselves.

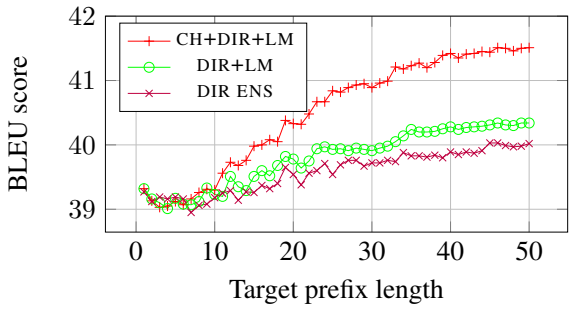


Figure 3: Impact of target prefix length for the channel model (CH+DIR+LM), direct model + LM (DIR+LM) and a direct ensemble (DIR ENS). We show detokenized BLEU on WMT De-En news2016 with beam 10.

### 3.4 Analysis

Using the channel model in online decoding enables searching a much larger space compared to n-best list re-ranking. However, online decoding is also challenging because the channel model needs to score the entire source sequence given a partial target which can be hard. To measure this, we simulate different target prefix lengths in an n-best list re-ranking setup. The n-best list is generated by the direct model and we re-rank it for different target prefixes of the candidate hypothesis. As in 3.2, we measure BLEU of the selected full candidate hypothesis. Figure 3 shows that the channel model enjoys much larger benefits from more target context than re-ranking with just the direct model and an LM (DIR+LM) or re-ranking with a direct ensemble (DIR ENS). This experiment shows the importance of large context sizes for the channel approach to work well. It indicates that the channel approach may not be able to effectively exploit the large search space in online decoding due to the limited conditioning context provided by partial target prefixes.

### 3.5 Re-ranking

Next, we switch to n-best re-ranking where we have the full target sentence available compared to online decoding. Noisy channel model re-ranking has been used by the top ranked entries of the WMT 2019 news translation shared task for English-German, German-English, Englsh-Russian and Russian-English (Ng et al., 2019). We compare to various baselines including right-to-left sequence to sequence models which are a popular choice for re-ranking and regularly feature in successful WMT submissions (Deng et al., 2018;

| | 5 | 10 | 50 | 100 |
|---|---|---|---|---|
| DIR | 39.1 | 39.2 | 39.3 | 39.2 |
| DIR ENS | 40.1 | 40.2 | 40.3 | 40.3 |
| DIR+LM | 40.0 | 40.2 | 40.6 | 40.7 |
| DIR+RL | 39.7 | 40.1 | 40.8 | 40.8 |
| DIR+RL+LM | 40.4 | 40.9 | 41.6 | 41.8 |
| CH+DIR | 39.7 | 40.0 | 40.5 | 40.5 |
| CH+DIR+LM | 40.8 | 41.5 | 42.8 | 43.2 |

Table 2: Re-ranking BLEU with different n-best list sizes on news2016 of WMT De-En. We compare to decoding with a direct model only (DIR) and decoding with an ensemble of direct models (DIR ENS). Table 5 in the appendix shows standard deviations.

| | WMT De-En | WMT En-De | WMT Zh-En | IWSLT De-En |
|---|---|---|---|---|
| DIR | 34.5 | 28.4 | 24.4 | 33.3 |
| DIR ENS | 35.5 | 29.0 | 25.2 | 34.5 |
| DIR+LM | 36.0 | 29.4 | 24.9 | 34.2 |
| DIR+RL | 35.7 | 29.3 | 25.3 | 34.4 |
| DIR+RL+LM | 36.8 | 30.0 | 25.4 | 34.9 |
| CH+DIR | 35.1 | 28.3 | 24.8 | 34.0 |
| CH+DIR+LM | 37.7 | 30.5 | 25.6 | 35.5 |

Table 3: Re-ranking accuracy with $k_1 = 50$ on four language directions on the respective test sets. See Table 6 in the appendix for standard deviations.

Koehn et al., 2018; Junczys-Dowmunt, 2018).

Table 2 shows that the noisy channel model outperforms the baseline (DIR) by up to 4.0 BLEU for very large beams, the ensemble by up to 2.9 BLEU (DIR ENS) and the best right-to-left configuration by 1.4 BLEU (DIR+RL+LM). The channel approach improves more than other methods with larger n-best lists by adding 2.4 BLEU from $k_1 = 5$ to $k_1 = 100$. Other methods improve a lot less with larger beams, e.g., DIR+RL+LM has the next largest improvement of 1.4 BLEU when increasing the beam size but this is still significantly lower than for the noisy channel approach. Adding a language model benefits all settings (DIR+LM, DIR+RL+LM, CH+DIR+LM) but the channel approach benefits most (CH+DIR vs CH+DIR+LM). The direct model with a language model (DIR+LM) performs better than for online decoding, likely because the constrained reranking setup mitigates explaining away effects (cf. Table 1).

Interestingly, both CH+DIR or DIR+LM give only modest improvements compared to CH+DIR+LM. Although previous work demonstrated that reranking with CH+DIR can improve over DIR, we show that the channel model is important to properly leverage the language model without suffering from explaining away effects (Xu and Carpuat, 2018; Wang et al., 2017). Test results on all language directions confirm that CH+DIR+LM performs best (Table 3).

## 4 Conclusion

Previous work relied on incremental channel models which do not make use of the entire source even though it is available and, as we demonstrate, beneficial. Standard sequence to sequence models are a simple parameterization for the channel probability that naturally exploits the entire source. This parameterization outperforms strong baselines such as ensembles of direct models and right-to-left models. Channel models are particularly effective with large context sizes and an interesting future direction is to iteratively refine the output while conditioning on previous contexts.

## References


Alexei Baevski and Michael Auli. 2018. Adaptive input representations for neural language modeling. *arXiv*, abs/1809.10853.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR*.

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.

Yongchao Deng, Shanbo Cheng, Jun Lu, Kai Song, Jingang Wang, Shenglan Wu, Liang Yao, Guchun Zhang, Haibo Zhang, Pei Zhang, Changfeng Zhu, and Boxing Chen. 2018. Alibaba's neural machine translation systems for wmt18. In *Proc. of WMT*.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018a. Understanding back-translation at scale. In *Proc. of EMNLP*.

Sergey Edunov, Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2018b. Classical structured prediction losses for sequence to sequence learning. In *Proc. of NAACL*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional Sequence to Sequence Learning. In *Proc. of ICML*.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. *arXiv*, abs/1503.03535.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, Will Lewis, Mu Li, Shujie Liu, Tie-Yan Liu, Renqian Luo, Arul Menezes, Tao Qin, Frank Seide, Xu Tan, Fei Tian, Lijun Wu, Shuangzhi Wu, Yingce Xia, Dongdong Zhang, Zhirui Zhang, and Ming Zhou. 2018. Achieving human parity on automatic chinese to english news translation. *arXiv*, abs/1803.05567.

Marcin Junczys-Dowmunt. 2018. Microsoft's submission to the wmt2018 news translation task: How i learned to stop worrying and love the data. In *Proc. of WMT*.

Dan Klein and Christopher Manning. 2001. Conditional structure versus conditional estimation in nlp. In *Proc. of EMNLP*.

Philipp Koehn, Kevin Duh, and Brian Thompson. 2018. The jhu machine translation systems for wmt 2018. In *Proc. of WMT*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proc. of NAACL*.

Gurvan L'Hostis, David Grangier, and Michael Auli. 2016. Vocabulary selection strategies for neural machine translation. *arXiv*, abs/1610.00072.

Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. 2019. Facebook fair's wmt19 news translation task submission. In *Proc. of WMT*.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proc. of NAACL System Demonstrations*.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. 2018. Scaling neural machine translation. In *Proc. of WMT*.

Matt Post. 2018. A call for clarity in reporting bleu scores. *arXiv*, abs/1804.08771.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. In *Proc. of ACL*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proc. of ACL*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Proc. of NIPS*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Proc. of NIPS*.

Yuguang Wang, Shanbo Cheng, Liyang Jiang, Jiajun Yang, Wei Chen, Muze Li, Lin Shi, Yanfeng Wang, and Hongtao Yang. 2017. Sogou neural machine translation systems for wmt17. In *Proceedings of the Second Conference on Machine Translation*, pages 410–415.

Weijia Xu and Marine Carpuat. 2018. The university of maryland's chinese-english neural machine translation systems at wmt18. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 535–540.

Lei Yu, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Tomás Kociský. 2017. The neural noisy channel. In *Proc. of ICLR*.

# MultiFiT: Efficient Multi-lingual Language Model Fine-tuning

**Julian Eisenschlos**[1†*] **Sebastian Ruder**[2,3‡*] **Piotr Czapla**[4*]
**Marcin Kardas**[4*] **Sylvain Gugger**[5] **Jeremy Howard**[5,6]
[1]ASAPP, Inc. [2]National University of Ireland [3]Aylien Ltd., Dublin
[4]n-waves, Wrocław [5]fast.ai [6]University of San Francisco

## Abstract

Pretrained language models are promising particularly for low-resource languages as they only require unlabelled data. However, training existing models requires huge amounts of compute, while pretrained cross-lingual models often underperform on low-resource languages. We propose Multi-lingual language model Fine-Tuning (MultiFiT) to enable practitioners to train and fine-tune language models efficiently in their own language. In addition, we propose a zero-shot method using an existing pretrained cross-lingual model. We evaluate our methods on two widely used cross-lingual classification datasets where they outperform models pretrained on orders of magnitude more data and compute. We release all models and code[1].

## 1 Introduction

Pretrained language models (LMs) have shown striking improvements on a range of natural language processing (NLP) tasks (Peters et al., 2018a; Howard and Ruder, 2018; Devlin et al., 2018). These models only require unlabelled data for training and are thus particularly useful in scenarios where labelled data is scarce. As much of NLP research has focused on the English language, the larger promise of these models is to bridge the digital language divide[2] and enable the application of NLP methods to many of the world's other 6,000 languages where labelled data is less plentiful.

Recently, cross-lingual extensions of these LMs have been proposed that train on multiple languages jointly (Artetxe and Schwenk, 2018; Lample and Conneau, 2019). These models are able to perform zero-shot learning, only requiring labelled data in the source language. However, source data in another language may often not be available, whereas obtaining a small number of labels is typically straightforward.

Furthermore such models have several downsides: a) some variants rely on large amounts of parallel data, which may not be available for truly low-resource languages; b) they require a huge amount of compute for training[3]; and c) cross-lingual models underperform on low-resource languages—precisely the setting where they would be most useful. We are aware of two possible reasons for this: 1) Languages that are less frequently seen during training are underrepresented in the embedding space.[4] 2) Infrequent scripts are over-segmented in the shared word piece vocabulary (Wang et al., 2019).

In this work, we show that small monolingual LMs are able to outperform expensive cross-lingual models both in the zero-shot and the supervised setting. We propose Multi-lingual language model Fine-tuning (MultiFit) to enable practitioners to train and fine-tune language models efficiently.[5] Our model combines universal language model fine-tuning (ULMFiT; Howard and Ruder, 2018) with the quasi-recurrent neural network (QRNN; Bradbury et al., 2017) and subword tokenization (Kudo, 2018) and can be pretrained on a single Tesla V100 GPU in a few hours. In addition, we propose to use a pretrained cross-lingual model's predictions as pseudo labels to adapt the

[1] http://nlp.fast.ai

[*] The first four authors contributed equally.

[†] Corresponding author: eisenjulian@gmail.com

[‡] Sebastian is now affiliated with DeepMind.

[2] http://labs.theguardian.com/digital-language-divide/

[3] The training cost is amortized over time as pretraining only needs to be performed once and fine-tuning is much cheaper. However, if a model needs to be applied to a new language or a domain not covered by the model, a new model needs to be trained from scratch.

[4] This is similar to how word embeddings are known to underperform on low-frequency tokens (Gong et al., 2018).

[5] We use 'multilingual' as referring to training *independent* models in multiple languages. We use 'cross-lingual' to refer to training a *joint* model across multiple languages.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5702–5707,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

monolingual language model to the zero-shot setting. We evaluate our models on two widely used cross-lingual classification datasets, MLDoc (Schwenk and Li, 2018) and CLS (Prettenhofer and Stein, 2010) where we outperform the state-of-the-art zero-shot model LASER (Artetxe and Schwenk, 2018) and multi-lingual BERT (Devlin et al., 2018) in the supervised setting—even without any pretraining. In the zero-shot setting, we outperform both models using pseudo labels—and report significantly higher performance with as little as 100 examples. We finally show that information from monolingual and cross-lingual language models is complementary and that pretraining makes models robust to noise.

## 2 Related work

**Pretrained language models** Pretrained language models based on an LSTM (Peters et al., 2018a; Howard and Ruder, 2018) and a Transformer (Radford et al., 2018; Devlin et al., 2018) have been proposed. Recent work (Peters et al., 2018b) suggests that—all else being equal—an LSTM outperforms the Transformer in terms of downstream performance. For this reason, we use a variant of the LSTM as our language model.

**Cross-lingual pretrained language models** The multi-lingual BERT model is pretrained on the Wikipedias of 104 languages using a shared word piece vocabulary. LASER (Artetxe and Schwenk, 2018) is trained on parallel data of 93 languages with a shared BPE vocabulary. XLM (Lample and Conneau, 2019) additionally pretrains BERT with parallel data. These models enable zero-shot transfer, but achieve lower results than monolingual models. In contrast, we focus on making the training of monolingual language models more efficient in a multi-lingual context. Concurrent work (Mulcaire et al., 2019) pretrains on English and another language, but shows that cross-lingual pretraining only helps sometimes.

**Multi-lingual language modeling** Training language models in non-English languages has only recently received some attention. Kawakami et al. (2017) evaluate on seven languages. Cotterell et al. (2018) study 21 languages. Gerz et al. (2018) create datasets for 50 languages. All of these studies, however, only create small datasets, which are inadequate for pretraining language models. In contrast, we are among the first to report the performance of monolingual language models on downstream tasks in multiple languages.

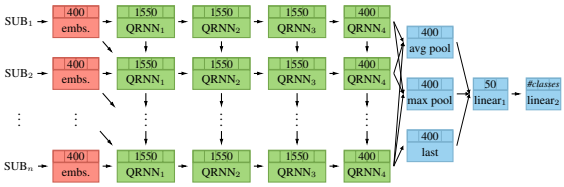


Figure 1: The MultiFiT language model with classifier consisting of a subword embedding layer, four QRNN layers, an aggregation layer, and two linear layers.

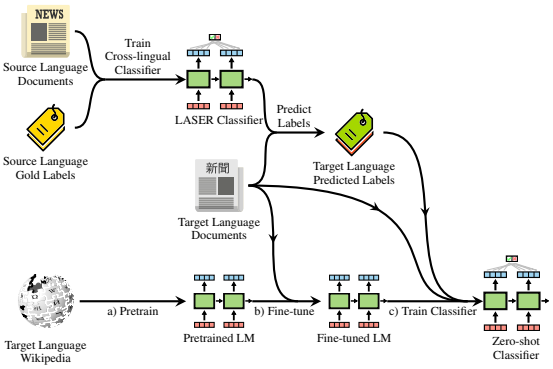


Figure 2: The steps of our cross-lingual bootstrapping method for zero-shot cross-lingual transfer. a) A monolingual language model (LM) is pretrained on target language data; b) the LM is fine-tuned on target language documents; and c) the LM is fine-tuned as a classifier using the zero-shot predictions from a linear classification layer fine-tuned on top of cross-lingual representations from LASER.

## 3 Our method

### 3.1 Multi-lingual Fine-Tuning

We propose Multi-lingual Fine-tuning (MultiFit). Our method uses the ULMFiT model (Howard and Ruder, 2018) with discriminative fine-tuning as foundation. ULMFiT is based on a 3-layer AWD-LSTM (Mer, 2017) language model. The AWD-LSTM is a regular LSTM (Hochreiter and Schmidhuber, 1997) with tuned dropout hyperparameters. To enable faster training and fine-tuning of the model, we replace it with a QRNN (Bradbury et al., 2017). The QRNN alternates convolutional layers, which are parallel across timesteps, and a recurrent pooling function, which is parallel across channels. It has been shown to outperform LSTMs, while being up to 16× faster at train and test time. ULMFiT in addition is restricted to words as input. To make our model

more robust across languages, we use subword tokenization based on a unigram language model (Kudo, 2018), which is more flexible compared to byte-pair encoding (Sennrich et al., 2016). We additionally employ label smoothing (Szegedy et al., 2016) and a novel cosine variant of the one-cycle policy (Smith, 2018)[6], which we found to outperform ULMFiT's slanted triangular learning rate schedule and gradual unfreezing. The full model can be seen in Figure 1.

### 3.2 Cross-lingual Bootstrapping

Prior methods have employed cross-lingual training strategies relying on parallel data and a shared BPE vocabulary. These can be combined with our language model, but increase its training complexity. For the case where an existing pretrained cross-lingual model and source language data are available, we propose a bootstrapping method (Ruder and Plank, 2018) that uses the pretrained model's zero-shot predictions as pseudo labels to fine-tune the monolingual model on target language data.

The steps of the method can be seen in Figure 2. Specifically, we first fine-tune a linear classification layer on top of pretrained cross-lingual representations on source language training data. We then apply this cross-lingual classifier to the target language data and store its predicted label for every example. We now fine-tune our pretrained LM on the target language data and these pseudo labels[7]. Importantly, this method enables our monolingual LM to significantly outperform its cross-lingual teacher in the zero-shot setting (§5).

## 4 Experimental setup

This section provides an overview of our experimental setup; see the appendix for full details.

**Data** We evaluate our models on the Multilingual Document Classification Corpus (MLDoc; Schwenk and Li, 2018)[8]—a new subset of Reuters Corpus Volume 2 (Lewis et al., 2004) with balanced class priors for eight languages—and on the Cross-Lingual Sentiment dataset (CLS; Prettenhofer and Stein, 2010)[9] consisting of Amazon product reviews in four languages. We provide an overview of the datasets in Table 1.

| | Domain | Languages | Train | Dev | Test |
|---|---|---|---|---|---|
| MLDoc | News | EN, DE, ES, FR, IT, JA, RU, ZH | 1k / 2k / 5k / 10k | 2k | 10k |
| CLS | Product reviews | EN, DE, FR, JA | 2k | - | 2k |

Table 1: The domain, languages, and number of training, development, and test examples in each dataset.

| | DE | ES | FR | IT | JA | RU | ZH |
|---|---|---|---|---|---|---|---|
| *Zero-shot (1,000 source language examples)* | | | | | | | |
| MultiCCA | 81.20 | 72.50 | 72.38 | 69.38 | 67.63 | 60.80 | 74.73 |
| LASER, paper | 86.25 | **79.30** | 78.30 | 70.20 | 60.95 | 67.25 | 70.98 |
| LASER, code | 87.65 | 75.48 | 84.00 | 71.18 | 64.58 | 66.58 | 76.65 |
| MultiBERT | 82.35 | 74.98 | 83.03 | 68.27 | 64.58 | **71.58** | 66.17 |
| MultiFiT, pseudo | **91.62** | 79.10 | **89.42** | **76.02** | **69.57** | 67.83 | **82.48** |
| *Supervised (100 target language examples)* | | | | | | | |
| MultiFit | 90.90 | 89.00 | 85.03 | 80.12 | 80.55 | 73.55 | 88.02 |
| *Supervised (1,000 target language examples)* | | | | | | | |
| MultiCCA | 93.70 | 94.45 | 92.05 | 85.55 | 85.35 | 85.65 | 87.30 |
| LASER, paper | 92.70 | 88.75 | 90.80 | 85.93 | 85.15 | 84.65 | 88.98 |
| MultiBERT | 94.00 | 95.15 | 93.20 | 85.82 | 87.48 | 86.85 | 90.72 |
| Monolingual BERT | 94.93 | - | - | - | - | - | 92.17 |
| MultiFiT, no wiki | 95.23 | 95.07 | 94.65 | 89.30 | 88.63 | 87.52 | 90.03 |
| MultiFiT | **95.90** | **96.07** | **94.75** | **90.25** | **90.03** | **87.65** | **92.52** |

Table 2: Comparison of zero-shot and supervised methods on MLDoc.

**Pretraining** We pretrain our models on 100M tokens extracted from the Wikipedia of the corresponding language for 10 epochs. As fewer tokens might be available for some languages, we also compare against a version (no wiki) that uses no pretraining. For all models, we fine-tune the LMs on the target data of the same language for 20 epochs. We perform subword tokenization with the unigram language model (Kudo, 2018).

**Evaluation settings** We compare two settings based on the availability of source and target language data: supervised and zero-shot. In the supervised setting, every model is fine-tuned and evaluated on examples from the target language. In the zero-shot setting, every model is fine-tuned on source language examples and evaluated on target language examples. In all cases, we use English as the source language.

**Baselines** We compare against the state-of-the-art cross-lingual embedding models LASER (Artetxe and Schwenk, 2018), which uses a large parallel corpus, multilingual BERT (MultiB-

[6]The idea is due to private conversation with the author.

[7]Distillation (Hinton et al., 2015) yielded similar results.

[8]https://github.com/facebookresearch/MLDoc

[9]https://webis.de/data/webis-cls-10.html

| | | DE | | | FR | | | JA | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Books | DVD | Music | Books | DVD | Music | Books | DVD | Music |
| *Zero-shot* | LASER, code | 84.15 | 78.00 | 79.15 | 83.90 | 83.40 | 80.75 | 74.99 | 74.55 | 76.30 |
| | MultiBERT | 72.15 | 70.05 | 73.80 | 75.50 | 74.70 | 76.05 | 65.41 | 64.90 | 70.33 |
| | MultiFiT, pseudo | **89.60** | **81.80** | **84.40** | **87.84** | **83.50** | **85.60** | **80.45** | **77.65** | **81.50** |
| *Translat.* | MT-BOW | 79.68 | 77.92 | 77.22 | 80.76 | 78.83 | 75.78 | 70.22 | 71.30 | 72.02 |
| | CL-SCL | 79.50 | 76.92 | 77.79 | 78.49 | 78.80 | 77.92 | 73.09 | 71.07 | 75.11 |
| | BiDRL | 84.14 | 84.05 | 84.67 | 84.39 | 83.60 | 82.52 | 73.15 | 76.78 | 78.77 |
| *Super.* | MultiBERT | 86.05 | 84.90 | 82.00 | 86.15 | 86.90 | 86.65 | 80.87 | 82.83 | 79.95 |
| | MultiFiT | **93.19** | **90.54** | **93.00** | **91.25** | **89.55** | **93.40** | **86.29** | **85.75** | **86.59** |

Table 3: Comparison of zero-shot, translation-based and supervised methods (with 2k training examples) on all domains of CLS. MT-BOW and CL-SCL results are from (Zhou et al., 2016).

| | LSTM | QRNN |
|---|---|---|
| Language model pretraining | 143 | **71** |
| Classifier fine-tuning | 467 | **156** |

Table 4: Comparison of LSTM and QRNN per-batch training speed on a Tesla V100 (in ms) in MultiFiT.

ERT)[10], and monolingual BERT[11]. We also compare against the best models on each dataset, MultiCCA (Ammar et al., 2016), a cross-lingual word embedding model, and BiDRL (Zhou et al., 2016), which translates source and target data.

**Our methods** We evaluate our monolingual LMs in the supervised setting (MultiFit) and our LMs fine-tuned with pseudo labels from LASER in the zero-shot setting (pseudo).

| | DE | ES | ZH |
|---|---|---|---|
| ULMFiT | 94.19 | 95.23 | 66.82 |
| MultiFiT, no wiki | 95.23 | 95.07 | 90.03 |
| MultiFiT, small Wiki | 95.37 | 95.30 | 89.80 |
| MultiFiT | **95.90** | **96.07** | **92.52** |

Table 5: Comparison of MultiFiT results with different pretraining corpora and ULMFiT, fine-tuned with 1k labels on MLDoc.

## 5 Results

**MLDoc** We show results for MLDoc in Table 2. In the zero-shot setting, MultiBERT underperforms the comparison methods as the shared embedding space between many languages is overly restrictive. Our monolingual LMs outperform their cross-lingual teacher LASER in almost every setting. When fine-tuned with only 100 target language examples, they are able to outperform all zero-shot approaches except MultiFiT on DE and FR. This calls into question the need for zero-shot approaches, as fine-tuning with even a small number of target examples is able to yield superior performance. When fine-tuning with 1,000 target examples, MultiFiT—even without pretraining—outperforms all comparison methods, including monolingual BERT.

**CLS** We show results for CLS in Table 3. MultiFiT is able to outperform its zero-shot teacher LASER across all domains. Importantly, the bootstrapped monolingual model also outperforms more sophisticated models that are trained on translations across almost all domains. In the supervised setting, MultiFiT similarly outperforms multilingual BERT. For both datasets, our methods that have been pretrained on 100 million tokens outperform both multilingual BERT and LASER, models that have been trained with orders of magnitude more data and compute.

## 6 Analysis

**Speed** We compare the LSTM and QRNN cell in MultiFiT based on the speed for processing a single batch for pretraining and fine-tuning in Table 4. MultiFiT with a QRNN pretrains and fine-tunes about 2× and 3× faster respectively.

**MultiFiT vs. ULMFiT** We compare MultiFiT pretrained on 100M Wikipedia tokens against

[10] https://github.com/google-research/bert/blob/master/multilingual.md

[11] Models are available for English, Chinese, and German (https://deepset.ai/german-bert).

ULMFiT pretrained on the same data using a 3-layer LSTM and spaCy tokenization[12] as well as MultiFiT pretrained on 2M Wikipedia tokens, and MultiFiT with no pretraining in Table 5. Pretraining on more data generally helps. MultiFiT outperforms ULMFiT significantly; the performance improvement is particularly pronounced in Chinese where ULMFiT's word-based tokenization underperformed.

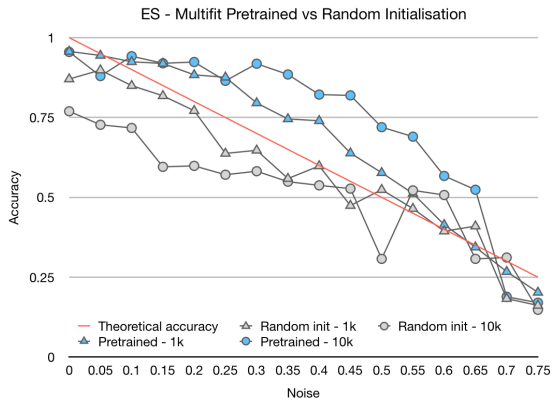


Figure 3: Comparison of MultiFiT's robustness to label noise on MLDoc with and without pretraining.

| | DE | ES | FR | IT | JA | RU | ZH |
|---|---|---|---|---|---|---|---|
| LASER, code | 87.65 | 75.48 | 84.00 | 71.18 | 64.58 | 66.58 | 76.65 |
| Random init. (1k) | 77.80 | 70.50 | 75.65 | 68.52 | 68.50 | 61.37 | 79.19 |
| Random init. (10k) | 90.53 | 69.75 | 87.40 | 72.72 | 67.55 | 63.67 | 81.44 |
| MultiFiT, pseudo (1k) | **91.34** | **78.92** | **89.45** | **76.00** | **69.57** | **68.19** | **82.45** |

Table 6: Bootstrapping results on MLDoc with and without pretraining, trained on 1k/10k LASER labels.

| | DE | ES | FR | IT | RU |
|---|---|---|---|---|---|
| Word-based | 95.28 | 95.97 | 94.72 | 89.97 | **88.02** |
| Subword | **96.10** | **96.07** | **94.75** | **94.75** | 87.65 |

Table 7: Comparison of different tokenization strategies for different languages on MLDoc.

**Robustness to noise** We suspect that MultiFiT is able to outperform its teacher as the information from pretraining makes it robust to label noise. To test this hypothesis, we train MultiFiT and a randomly initialized model with the same architecture on 1k and 10k examples of the Spanish MLDoc. We randomly perturb labels with a probability ranging from 0-0.75 and show results in Figure 3. The pretrained MultiFiT is able to partially ignore the noise, up to 65% of noisy training examples. Without pretraining, the model does not exceed the theoretical baseline (the percentage of correct examples). In addition, we compare MultiFiT with and without pretraining in Table 6. Pretraining enables MultiFiT to achieve much better performance compared to a randomly initialised model. Both results together suggest a) that pretraining increases robustness to noise and b) that information from monolingual and cross-lingual language models is complementary.

**Tokenization** Subword tokenization has been found useful for language modeling with morphologically rich languages (Czapla et al., 2018; Mielke and Eisner, 2019) and has been used in recent pretrained LMs (Devlin et al., 2018), but its concrete impact on downstream performance has not been observed. We train models with the best performing vocabulary sizes for subword (15k) and regular word-based tokenization (60k) with the Moses tokenizer (Koehn et al., 2007) on German and Russian MLDoc and show results in Table 7. Subword tokenization outperforms word-based tokenization on most languages, while being faster to train due to the smaller vocabulary size.

## 7 Conclusion

We have proposed novel methods for multilingual fine-tuning of languages that outperform models trained with far more data and compute on two widely studied cross-lingual text classification datasets on eight languages in both the zero-shot and supervised setting.

[12] https://spacy.io/api/tokenizer

## References

2017. Pointer Sentinel Mixture Models. In *Proceedings of ICLR 2017*.

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A. Smith. 2016. Massively Multilingual Word Embeddings.

Mikel Artetxe and Holger Schwenk. 2018. Massively Multilingual Sentence Embeddings for Zero-Shot Cross-Lingual Transfer and Beyond. *arXiv preprint arXiv:1812.10464*.

James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. 2017. Quasi-Recurrent Neural Networks. In *Proceedings of ICLR 2017*.

Ryan Cotterell, Sebastian Mielke, Jason Eisner, and Brian Roark. 2018. Are all languages equally hard to language-model? In *Proceedings of NAACL-HLT 2018*, pages 536–541.

Piotr Czapla, Jeremy Howard, and Marcin Kardas. 2018. Universal language model fine-tuning with subword tokenization for polish. In *Proceedings of PolEval 2018 Workshop*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.

Daniela Gerz, Ivan Vulić, Edoardo Ponti, Jason Naradowsky, Roi Reichart, and Anna Korhonen. 2018. Language Modeling for Morphologically Rich Languages: Character-Aware Modeling for Word-Level Prediction. *Proceedings of TACL 2018*.

Chengyue Gong, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2018. FRAGE: Frequency-Agnostic Word Representation. In *Proceedings of NIPS 2018*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the Knowledge in a Neural Network. *arXiv preprint arXiv:1503.02531*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. In *Proceedings of ACL 2018*.

Kazuya Kawakami, Chris Dyer, and Phil Blunsom. 2017. Learning to Create and Reuse Words in Open-Vocabulary Neural Language Modeling. In *ACL 2017*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL 2007*, pages 177–180.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of ACL 2018*, pages 66–75.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual Language Model Pretraining. *arXiv preprint arXiv:1901.07291*.

David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. 2004. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397.

Sebastian J. Mielke and Jason Eisner. 2019. Spell Once, Summon Anywhere: A Two-Level Open-Vocabulary Language Model. In *Proceedings of AAAI 2019*.

Phoebe Mulcaire, Jungo Kasai, and Noah Smith. 2019. Polyglot Contextual Representations Improve Crosslingual Transfer. In *Proceedings of NAACL 2019*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018a. Deep contextualized word representations. In *Proceedings of NAACL-HLT 2018*.

Matthew E. Peters, Mark Neumann, Luke Zettlemoyer, Wen-tau Yih, Paul G Allen, and Computer Science. 2018b. Dissecting Contextual Word Embeddings: Architecture and Representation. In *Proceedings of EMNLP 2018*.

Peter Prettenhofer and Benno Stein. 2010. Cross-Language Text Classification using Structural Correspondence Learning. In *ACL '10 Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL '10 )*, pages 1118–1127.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving Language Understanding by Generative Pre-Training.

Sebastian Ruder and Barbara Plank. 2018. Strong Baselines for Neural Semi-supervised Learning under Domain Shift. In *Proceedings of ACL 2018*.

Holger Schwenk and Xian Li. 2018. A Corpus for Multilingual Document Classification in Eight Languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of ACL 2016*.

Leslie N Smith. 2018. A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay. *arXiv preprint arXiv:1803.09820*.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2016. Rethinking the Inception Architecture for Computer Vision. In *Proceedings of CVPR 2016*.

Xinyi Wang, Hieu Pham, Philip Arthur, and Graham Neubig. 2019. Multilingual neural machine translation with soft decoupled encoding. In *Proceedings of ICLR 2019*.

Xinjie Zhou, Xianjun Wan, and Jianguo Xiao. 2016. Cross-Lingual Sentiment Classification with Bilingual Document Representation Learning. *Proceedings of ACL 2016*, pages 1403–1412.

# Hint-Based Training for Non-Autoregressive Machine Translation

**Zhuohan Li**[1] **Zi Lin**[2,3] **Di He**[2] **Fei Tian**[4] **Tao Qin**[5] **Liwei Wang**[2] **Tie-Yan Liu**[5]
[1]UC Berkeley [2]Peking University [3]Google AI [4]Facebook [5]Microsoft Research
zhuohan@cs.berkeley.edu {zi.lin, di_he, wanglw}@pku.edu.cn
feitia@fb.com {taoqin, tyliu}@microsoft.com

## Abstract

Due to the unparallelizable nature of the autoregressive factorization, AutoRegressive Translation (ART) models have to generate tokens sequentially during decoding and thus suffer from high inference latency. Non-AutoRegressive Translation (NART) models were proposed to reduce the inference time, but could only achieve inferior translation accuracy. In this paper, we proposed a novel approach to leveraging the hints from hidden states and word alignments to help the training of NART models. The results achieve significant improvement over previous NART models for the WMT14 En-De and De-En datasets and are even comparable to a strong LSTM-based ART baseline but one order of magnitude faster in inference.

## 1 Introduction

Neural machine translation has attracted much attention in recent years (Bahdanau et al., 2014, 2016; Kalchbrenner et al., 2016; Gehring et al., 2016). Given a sentence $x = (x_1, \ldots, x_{T_x})$ from the source language, the straight-forward way for translation is to generate the target words $y = (y_1, \ldots, y_{T_y})$ one by one from left to right. This is also known as the *AutoRegressive* Translation (ART) models, in which the joint probability is decomposed into a chain of conditional probabilities:

$$P(y|x) = \Pi_{t=1}^{T_y} P(y_t|y_{<t}, x), \tag{1}$$

While the ART models have achieved great success in terms of translation quality, the time consumption during inference is still far away from satisfactory. During training, the predictions at different positions can be estimated in parallel since the ground truth pair $(x, y)$ is exposed to the model. However, during inference, the model has to generate tokens sequentially as $y_{<t}$ must be inferred on the fly. Such autoregressive behavior becomes the bottleneck of the computational time (Wu et al., 2016).

In order to speed up the inference process, a line of works begin to develop non-autoregressive translation models. These models break the autoregressive dependency by decomposing the joint probability with

$$P(y|x) = P(T_y|x)\Pi_{t=1}^{T_y} P(y_t|x). \tag{2}$$

The lost of autoregressive dependency largely hurt the consistency of the output sentences, increase the difficulty in the learning process and thus lead to a low quality translation. Previous works mainly focus on adding different components into the NART model to improve the expressiveness of the network structure to overcome the loss of autoregressive dependency (Gu et al., 2017; Lee et al., 2018; Kaiser et al., 2018). However, the computational overhead of new components will hurt the inference speed, contradicting with the goal of the NART models: to parallelize and speed up neural machine translation models.

To tackle this, we proposed a novel hint-based method for NART model training. We first investigate the causes of the poor performance of the NART model. Comparing with the ART model, we find that: (1) the positions where the NART model outputs incoherent tokens will have very high hidden states similarity; (2) the attention distributions of the NART model are more ambiguous than those of ART model. Therefore, we design two kinds of hints from the hidden states and attention distributions of the ART model to help the training of the NART model. The experimental results show that our model achieves significant improvement over the NART baseline models and is even comparable to a strong ART baseline in Wu et al. (2016).

The work was performed at Microsoft Research Asia.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5708–5713,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

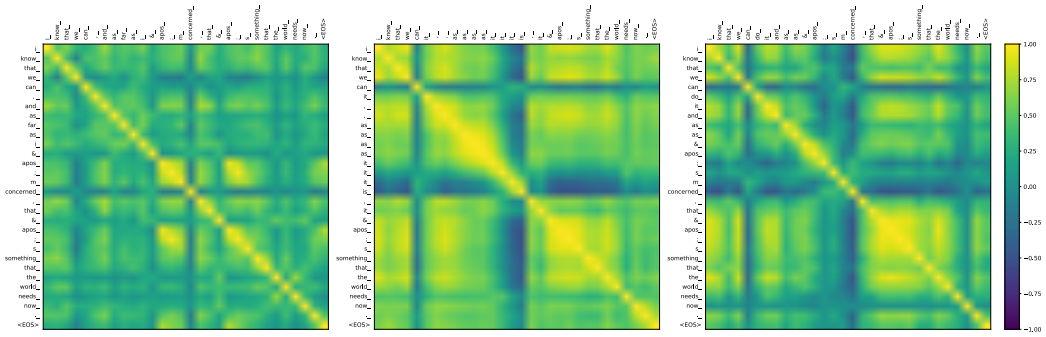

(a) ART Teacher (b) NART without Hints (c) NART with Hints

Figure 1: Case study: the above three figures visualize the hidden state cosine similarities of different models. The axes correspond to the generated target tokens. Each pixel shows the cosine similarities $\cos_{ij}$ between the last layer hidden states of the $i$-th and $j$-th generated tokens, where the diagonal pixel will always be 1.0.

## 2 Approach

In this section, we first describe the observations on the ART and NART models, and then discuss what kinds of information can be used as hints to help the training of the NART model. We follow the network structure in Vaswani et al. (2017), use a copy of the source sentence as decoder input, remove the attention masks in decoder self-attention layers and add a positional attention layer as suggested in Gu et al. (2017). We provide a visualization of ART and NART models we used in Figure 3 and a detailed description of the model structure in Appendix.

### 2.1 Observation: Illed States and Attentions

According to the case study in Gu et al. (2017), the translations of the NART models contain incoherent phrases (e.g. repetitive words) and miss meaningful tokens on the source side, while these patterns do not commonly appear in ART models. After some empirical study, we find two non-obvious facts that lead to this phenomenon.

First, we visualize the cosine similarities between decoder hidden states of a certain layer in both ART and NART models for sampled cases. Mathematically, for a set of hidden states $r_1, \ldots, r_T$, the pairwise cosine similarity can be derived by $\cos_{ij} = \langle r_i, r_j \rangle / (\|r_i\| \cdot \|r_j\|)$. We then plot the heatmap of the resulting matrix cos. A typical example is shown in Figure 1, where the cosine similarities in the NART model are larger than those of the ART model, indicating that the hidden states across positions in the NART model are "similar". Positions with highly-correlated hidden states tend to generate the same word and make the NART model output repetitive tokens, e.g., the yellow area on the top-left of Figure 1(b), while this does not happen in the ART model (Figure 1(a)). According to our statistics, 70% of the cosine similarities between hidden states in the ART model are less than 0.25, and 95% are less than 0.5.

Second, we visualize the encoder-decoder attentions for sampled cases, shown in Figure 2. Good attentions between the source and target sentences are usually considered to lead to accurate translation while poor ones may cause wrong output tokens (Bahdanau et al., 2014). In Figure 2(b), the attentions of the ART model almost covers all source tokens, while the attentions of the NART model do not cover "farm" but with two "morning". This directly makes the translation result worse in the NART model. These phenomena inspire us to use the intermediate hidden information in the ART model to guide the learning process of the NART model.

### 2.2 Hints from the ART teacher Model

Our study motivates us to leverage the intermediate hidden information from an ART model to improve the NART model. We focus on how to define hints from a well-trained ART *teacher* model and use it to guide the training process of a NART *student* model. We study layer-to-layer hints and assume both the teacher and student models have an $M$-layer encoder and an $N$-layer decoder, despite the difference in stacked components.

Without the loss of generality, we discuss our method on a given paired sentence $(x, y)$. In real experiments, losses are averaged over all training data. For the teacher model, we use $a^{tr}_{t,l,h}$ as the encoder-to-decoder attention distribution of $h$-th head in the $l$-th decoder layer at position $t$, and

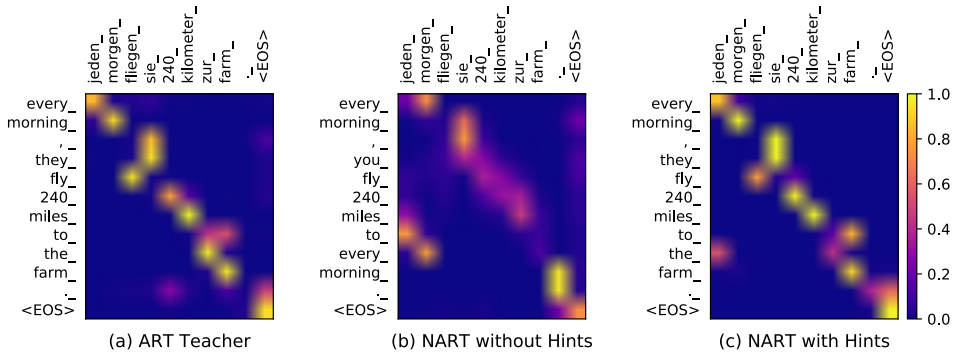


Figure 2: Case study: the above three figures visualize the encoder-decoder attention weights of different models. The x-axis and y-axis correspond to the source and generated target tokens respectively. The attention distribution is from a single head of the third layer encoder-decoder attention, which is the most informative one according to our observation. Each pixel shows attention weights $\alpha_{ij}$ between the $i$-th source token and $j$-th target token.

use $r^{tr}_{t,l}$ as the output of the $l$-th decoder layer after feed forward network at position $t$. Correspondingly, $a^{st}_{t,l,h}$ and $r^{st}_{t,l}$ are used for the student model. We propose a hint-based training framework that contains two kinds of hints:

**Hints from hidden states** The discrepancy of hidden states motivates us to use hidden states of the ART model as a hint for the learning process of the NART model. One straight-forward method is to regularize the $L_1$ or $L_2$ distance between each pair of hidden states in ART and NART models. However, since the network components are quite different in ART and NART models, applying the straight-forward regression on hidden states hurts the learning process and fails. Therefore, we design a more implicit loss to help the student refrain from the incoherent translation results by acting towards the teacher in the hidden-state level:

$$\mathcal{L}_{hid} = \frac{2}{(T_y-1)T_yN}\sum_{s=1}^{T_y-1}\sum_{t=s+1}^{T_y}\sum_{l=1}^{N}\phi(d_{st}, d_{tr}),$$

where $d_{st} = \cos(r^{st}_{s,l}, r^{st}_{t,l})$, $d_{tr} = \cos(r^{tr}_{s,l}, r^{tr}_{t,l})$, and $\phi$ is a penalty function. In particular, we let

$$\phi(d_{st}, d_{tr}) = \begin{cases} -\log(1-d_{st}), & \text{if } d_{st} \geq \gamma_{st} \\ & \text{and } d_{tr} \leq \gamma_{tr}; \\ 0, & \text{else,} \end{cases}$$

where $-1 \leq \gamma_{st}, \gamma_{tr} \leq 1$ are two thresholds controlling whether to penalize or not. We design this loss since we only want to penalize hidden states that are highly similar in the NART model, but not similar in the ART model. We have tested several choices of $-\log(1-d_{st})$, e.g., $\exp(d_{st})$, from which we find similar experimental results.

**Hints from word alignments** We observe that meaningful words in the source sentence are sometimes untranslated by the NART model, and the corresponding positions often suffer from ambiguous attention distributions. Therefore, we use the word alignment information from the ART model to help the training of the NART model.

In particular, we minimize KL-divergence between the per-head encoder-to-decoder attention distributions of the teacher and the student to encourage the student to have similar word alignments to the teacher model, i.e.

$$\mathcal{L}_{align} = \frac{1}{T_yNH}\sum_{t=1}^{T_y}\sum_{l=1}^{N}\sum_{h=1}^{H} D_{\mathrm{KL}}(a^{tr}_{t,l,h} \| a^{st}_{t,l,h}).$$

Our final training loss $\mathcal{L}$ is a weighted sum of two parts stated above and the negative log-likelihood loss $\mathcal{L}_{nll}$ defined on bilingual sentence pair $(x, y)$, i.e.

$$\mathcal{L} = \mathcal{L}_{nll} + \lambda\mathcal{L}_{hid} + \mu\mathcal{L}_{align}, \quad (3)$$

where $\lambda$ and $\mu$ are hyperparameters controlling the weight of different loss terms.

## 3 Experiments

### 3.1 Experimental Settings

The evaluation is on two widely used public machine translation datasets: IWSLT14 German-to-English (De-En) (Huang et al., 2017; Bahdanau et al., 2016) and WMT14 English-to-German (En-De) dataset (Wu et al., 2016; Gehring et al., 2017). To compare with previous works, we also reverse WMT14 English-to-German dataset and obtain WMT14 German-to-English dataset.

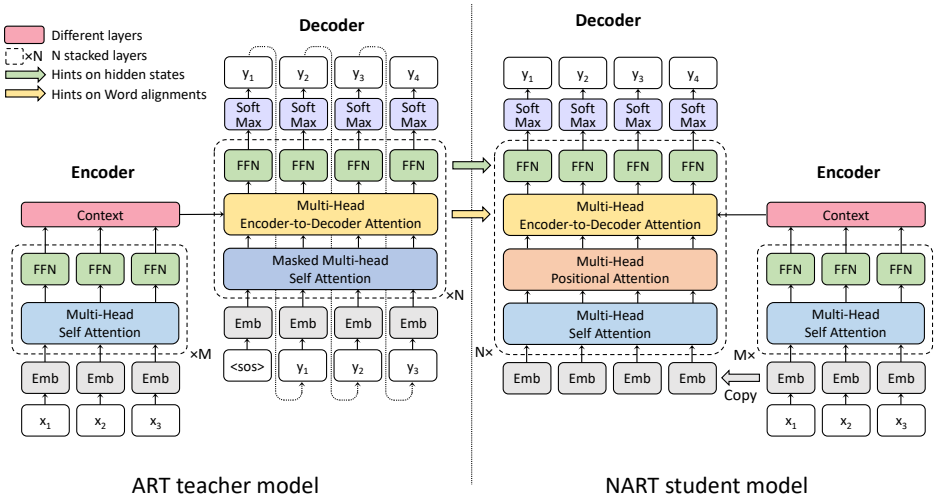


Figure 3: Hint-based training from ART model to NART model.

We pretrain Transformer (Vaswani et al., 2017) as the teacher model on each dataset, which achieves 33.26/27.30/31.29 in terms of BLEU (Papineni et al., 2002) in IWSLT14 De-En, WMT14 En-De and De-En test sets. The student model shares the same number of layers in encoder/decoder, size of hidden states/embeddings and number of heads as the teacher models (Figure 3). Following Gu et al. (2017); Kim and Rush (2016), we replace the target sentences by the decoded output of the teacher models.

Hyperparameters $(\gamma_{st}, \gamma_{tr}, \lambda, \mu)$ for hint-based learning are determined to make the scales of three loss components similar after initialization. We also employ label smoothing of value $\epsilon_{ls} = 0.1$ (Szegedy et al., 2016) in all experiments. We use Adam optimizer and follow the setting in Vaswani et al. (2017). Models for WMT14/IWSLT14 tasks are trained on 8/1 NVIDIA M40 GPUs respectively. The model is based on the open-sourced `tensor2tensor` (Vaswani et al., 2018).[1] More settings can be found in Appendix.

## 3.2 Inference

During training, $T_y$ does not need to be predicted as the target sentence is given. During testing, we have to predict the length of the target sentence for each source sentence. In many languages, the length of the target sentence can be roughly estimated from the length of the source sentence. We choose a simple method to avoid the computational overhead, which uses input length to determine target sentence length: $T_y = T_x + C$, where $C$ is a constant bias determined by the average length differences between the source and target training sentences. We can also predict the target length ranging from $[(T_x+C)-B, (T_x+C)+B]$, where $B$ is the halfwidth. By doing this, we can obtain multiple translation results with different lengths. Note that we choose this method only to show the effectiveness of our proposed method and a more advanced length estimation method can be used to further improve the performance.

Once we have multiple translation results, we additionally use our ART teacher model to evaluate each result and select the one that achieves the highest probability. As the evaluation is fully parallelizable (since it is identical to the parallel training of the ART model), this rescoring operation will not hurt the non-autoregressive property of the NART model.

## 3.3 Experimental Results

We compare our model with several baselines, including three ART models, the fertility based (FT) NART model (Gu et al., 2017), the deterministic iterative refinement based (IR) NART model (Lee et al., 2018), and the Latent Transformer (LT; Kaiser et al., 2018) which is not fully non-autoregressive by incorporating an autoregressive sub-module in the NART model architecture.

The results are shown in the Table 1.[2] Across

[1] Open-source code can be found at `https://github.com/zhuohan123/hint-nart`

[2] ‡ and † indicate that the latency is measured on our own platform or by previous works, respectively. Note that the latencies may be affected by hardware settings and such absolute values are not fair for direct comparison, so we also list the speedup of the works compared to their ART baselines.

| Models | WMT14 En-De | WMT14 De-En | IWSLT14 De-En | Latency | Speedup |
|---|---|---|---|---|---|
| *Autoregressive models* | | | | | |
| LSTM-based S2S (Wu et al., 2016; Bahdanau et al., 2016) | 24.60 | / | 28.53 | / | / |
| ConvS2S (Gehring et al., 2017; Edunov et al., 2017) | 26.43 | / | 32.84 | / | / |
| Transformer (Vaswani et al., 2017) | 27.30 | 31.29 | 33.26 | 784 ms$^{\ddagger}$ | 1.00× |
| *Non-autoregressive models* | | | | | |
| FT (Gu et al., 2017) | 17.69 | 20.62 | / | 39 ms$^{\dagger}$ | 15.6× |
| FT (rescoring 10 candidates) | 18.66 | 22.41 | / | 79 ms$^{\dagger}$ | 7.68× |
| FT (rescoring 100 candidates) | 19.17 | 23.20 | / | 257 ms$^{\dagger}$ | 2.36× |
| IR (Lee et al., 2018, adaptive refinement steps) | 21.54 | 25.43 | / | / | 2.39× |
| LT (Kaiser et al., 2018) | 19.8 | / | / | 105 ms$^{\dagger}$ | 5.78× |
| LT (rescoring 10 candidates) | 21.0 | / | / | / | / |
| LT (rescoring 100 candidates) | 22.5 | / | / | / | / |
| **NART w/ hints** | **21.11** | **25.24** | **25.55** | **26 ms**$^{\ddagger}$ | **30.2×** |
| **NART w/ hints** ($B = 4$, 9 candidates) | **25.20** | **29.52** | **28.80** | **44 ms**$^{\ddagger}$ | **17.8×** |

Table 1: Performance on WMT14 En-De, De-En and IWSLT14 De-En tasks. "/" means non-reportable.

different datasets, our method achieves significant improvements over previous non-autoregressive models. Specifically, our method outperforms fertility based NART model with 6.54/7.11 BLEU score improvements on WMT En-De and De-En tasks in similar settings and achieves comparable results with state-of-the-art LSTM-based model on WMT En-De task. Furthermore, our model achieves a speedup of 30.2 (output a single sentence) or 17.8 (teacher rescoring) times over the ART counterparts. Note that our speedups significantly outperform all previous works, because of our lighter design of the NART model: without any computationally expensive module trying to improve the expressiveness.

We also visualize the hidden state cosine similarities and attention distributions for the NART model with hint-based training, as shown in Figure 1(c) and 2(c). With hints from hidden states, the hidden states similarities of the NART model decrease in general, and especially for the positions where the original NART model outputs incoherent phrases. The attention distribution of the NART model after hint-based training is more similar to the ART teacher model and less ambiguous comparing to the NART model without hints.

According to our empirical analysis, the percentage of repetitive words drops from 8.3% to 6.5% by our proposed methods on the IWSLT14 De-En test set, which is a 20%+ reduction. This shows that our proposed method effectively improve the quality of the translation outputs. We also provide several case studies in Appendix.

Finally, we conduct an ablation study on IWSLT14 De-En task. As shown in Table 2, the hints from word alignments provide an improvement of about 1.6 BLEU points, and the hints from hidden states improve the results by about 0.8 BLEU points. We also test these models on a subsampled set whose source sentence lengths are at least 40. Our model outperforms the baseline model by more than 3 BLEU points (20.63 v.s. 17.48).

| Model | $\mathcal{L}_{nll}$ | $\mathcal{L}_{nll} + \mathcal{L}_{align}$ | $\mathcal{L}_{nll} + \mathcal{L}_{align} + \mathcal{L}_{hid}$ |
|---|---|---|---|
| **BLEU** | 23.08 | 24.76 | **25.55** |
| **Long-sentence BLEU** | 17.48 | 19.24 | **20.63** |

Table 2: Ablation studies on IWSLT14 De-En. Results are BLEU scores without teacher rescoring.

## 4 Conclusion

In this paper, we proposed to use hints from a well-trained ART model to enhance the training of NART models. Our results on WMT14 En-De and De-En significantly outperform previous NART baselines, with one order of magnitude faster in inference than ART models. In the future, we will focus on designing new architectures and training methods for NART models to achieve comparable accuracy as ART models.

## Acknowledgment

This work is supported by National Key R&D Program of China (2018YFB1402600), NSFC (61573026) and BJNSF (L172037) and a grant from Microsoft Research Asia. We would like to thank the anonymous reviewers for their valuable comments on our paper.

## References


Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2016. An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Sergey Edunov, Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2017. Classical structured prediction losses for sequence to sequence learning. *arXiv preprint arXiv:1711.04956*.

Jonas Gehring, Michael Auli, David Grangier, and Yann N Dauphin. 2016. A convolutional encoder model for neural machine translation. *arXiv preprint arXiv:1611.02344*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. 2017. Non-autoregressive neural machine translation. *arXiv preprint arXiv:1711.02281*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Minghao Hu, Yuxing Peng, Furu Wei, Zhen Huang, Dongsheng Li, Nan Yang, and Ming Zhou. 2018. Attention-guided answer distillation for machine reading comprehension. *arXiv preprint arXiv:1808.07644*.

Po-Sen Huang, Chong Wang, Dengyong Zhou, and Li Deng. 2017. Neural phrase-based machine translation. *arXiv preprint arXiv:1706.05565*.

Łukasz Kaiser, Aurko Roy, Ashish Vaswani, Niki Pamar, Samy Bengio, Jakob Uszkoreit, and Noam Shazeer. 2018. Fast decoding in sequence models using discrete latent variables. *arXiv preprint arXiv:1803.03382*.

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099*.

Yoon Kim and Alexander M Rush. 2016. Sequence-level knowledge distillation. *arXiv preprint arXiv:1606.07947*.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. *arXiv preprint arXiv:1802.06901*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2014. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826.

Ashish Vaswani, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan N. Gomez, Stephan Gouws, Llion Jones, Łukasz Kaiser, Nal Kalchbrenner, Niki Parmar, Ryan Sepassi, Noam Shazeer, and Jakob Uszkoreit. 2018. Tensor2tensor for neural machine translation. *CoRR*, abs/1803.07416.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

# Working Hard or Hardly Working: Challenges of Integrating Typology into Neural Dependency Parsers

**Adam Fisch*** **Jiang Guo*** **Regina Barzilay**
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
{fisch,jiang_guo,regina}@csail.mit.edu

## Abstract

This paper explores the task of leveraging typology in the context of cross-lingual dependency parsing. While this linguistic information has shown great promise in pre-neural parsing, results for neural architectures have been mixed. The aim of our investigation is to better understand this state-of-the-art. Our main findings are as follows: 1) The benefit of typological information is derived from coarsely grouping languages into syntactically-homogeneous clusters rather than from learning to leverage variations along individual typological dimensions in a compositional manner; 2) Typology consistent with the actual corpus statistics yields better transfer performance; 3) Typological similarity is only a rough proxy of cross-lingual transferability with respect to parsing.[1]

## 1 Introduction

Over the last decade, dependency parsers for resource-rich languages have steadily continued to improve. In parallel, significant research efforts have been dedicated towards advancing cross-lingual parsing. This direction seeks to capitalize on existing annotations in resource-rich languages by transferring them to the rest of the world's over 7,000 languages (Bender, 2011). The NLP community has devoted substantial resources towards this goal, such as the creation of universal annotation schemas, and the expansion of existing treebanks to diverse language families. Nevertheless, cross-lingual transfer gains remain modest when put in perspective: the performance of transfer models can often be exceeded using only a handful of annotated sentences in the target language (Section 5). The considerable divergence of language structures proves challenging for current models.

One promising direction for handling these divergences is linguistic typology. Linguistic typology classifies languages according to their structural and functional features. By explicitly highlighting specific similarities and differences in languages' syntactic structures, typology holds great potential for facilitating cross-lingual transfer (O'Horan et al., 2016). Indeed, non-neural parsing approaches have already demonstrated empirical benefits of typology-aware models (Naseem et al., 2012; Täckström et al., 2013; Zhang and Barzilay, 2015) While adding discrete typological attributes is straightforward for traditional feature-based approaches, for modern neural parsers finding an effective implementation choice is more of an open question. Not surprisingly, the reported results have been mixed. For instance, Ammar et al. (2016) found no benefit to using typology for parsing when using a neural-based model, while Wang and Eisner (2018) and Scholivet et al. (2019) did in several cases.

There are many possible hypotheses that can attempt to explain the state-of-the-art. Might neural models already implicitly learn typological information on their own? Is the hand-specified typology information sufficiently accurate — or provided in the right granularity — to always be useful? How do cross-lingual parsers use, or ignore, typology when making predictions? Without understanding answers to these questions, it is difficult to develop a principled way for robustly incorporating linguistic knowledge as an inductive bias for cross-lingual transfer.

In this paper, we explore these questions in the context of two predominantly-used typology-based neural architectures for delexicalized dependency parsing.[2] The first method implements a variant of selective sharing (Naseem et al., 2012);

*The first two authors contributed equally.
[1]Code: github.com/ajfisch/TypologyParser
[2]We focus on delexicalized parsing in order to isolate the effects of syntax by removing lexical influences.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5714–5720, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

the second adds typological information as an additional feature of the input sentence. Both models are built on top of the popular Biaffine Parser (Dozat and Manning, 2017). We study model performance across multiple forms of typological representation and resolution.

Our key findings are as follows:

- **Typology as Quantization** Cross-lingual parsers use typology to coarsely group languages into syntactically-homogeneous clusters, yet fail to significantly capture finer distinctions or typological feature compositions. Our results indicate that they primarily take advantage of the simple geometry of the typological space (e.g. language distances), rather than specific variations in individual typological dimensions (e.g. `SV` vs. `VS`).
- **Typology Quality** Typology that is consistent with the actual corpus statistics results in better transfer performance, most likely by capturing a better reflection of the typological variations within that sample. Typology granularity also matters. Finer-grained, high-dimensional representations prove harder to use robustly.
- **Typology vs. Parser Transferability** Typological similarity only partially explains cross-lingual transferability with respect to parsing. The geometry of the typological space does not fully mirror that of the "parsing" space, and therefore requires task-specific refinement.

## 2 Typology Representations

**Linguistic Typology, $\mathbf{T}_L$:** The standard representation of typology is sets of annotations by linguists for a variety of language-level properties. These properties can be found in online databases such as The World Atlas of Language Structures (WALS) (Dryer and Haspelmath, 2013). We consider the same subset of features related to word order as used by Naseem et al. (2012), represented as a $k$-hot vector $T \in \{0,1\}^{\sum_f |V_f|}$, where $V_f$ is the set of values feature $f$ may take.

**Liu Directionalities, $\mathbf{T}_D$:** Liu (2010) proposed using a real-valued vector $T \in [0,1]^r$ of the average *directionalities* of each of a corpus' $r$ dependency relations as a typological descriptor. These serve as a more fine-grained alternative to linguistic typology. Compared to WALS, there are rarely missing values, and the *degree* of dominance of each dependency ordering is directly encoded — potentially allowing for better modeling of local variance within a language. It is important to note, however, that true directionalities require a parsed corpus to be derived; thus, they are not a realistic option for cross-lingual parsing in practice.[3] Nevertheless, we include them for completeness.

**Surface Statistics, $\mathbf{T}_S$:** It is possible to derive a proxy measure of typology from part-of-speech tag sequences alone. Wang and Eisner (2017) found surface statistics to be highly predictive of language typology, while Wang and Eisner (2018) replaced typological features entirely with surface statistics in their augmented dependency parser. Surface statistics have the advantage of being readily available and are not restricted to narrow linguistic definitions, but are less informed by the true underlying structure. We compute the set of hand-engineered features used in (Wang and Eisner, 2018), yielding a real-valued vector $T \in [0,1]^{2380}$.

## 3 Parsing Architecture

We use the graph-based Deep Biaffine Attention neural parser of (Dozat and Manning, 2017) as our baseline model. Given a delexicalized sentence $s$ consisting of $n$ part-of-speech tags, the Biaffine Parser embeds each tag $\boldsymbol{p}_i$, and encodes the sequence with a bi-directional LSTM to produce tag-level contextual representations $\boldsymbol{h}_i$. Each $\boldsymbol{h}_i$ is then mapped into head- and child-specific representations for arc and relation prediction, $\boldsymbol{h}_i^{\text{arc-dep}}$, $\boldsymbol{h}_i^{\text{arc-head}}$, $\boldsymbol{h}_i^{\text{rel-dep}}$, and $\boldsymbol{h}_i^{\text{rel-head}}$, using four separate multi-layer perceptrons.

For decoding, arc scores are computed as:

$$s_{ij}^{\text{arc}} = \left(\boldsymbol{h}_i^{\text{arc-head}}\right)^T \left(U^{\text{arc}} \boldsymbol{h}_j^{\text{arc-dep}} + \boldsymbol{b}^{\text{arc}}\right) \quad (1)$$

while the score for dependency label $r$ for edge $(i,j)$ is computed in a similar fashion:

$$\begin{aligned} s_{(i,j),r}^{\text{rel}} = &\left(\boldsymbol{h}_i^{\text{rel-head}}\right)^T U_r^{\text{rel}} \boldsymbol{h}_j^{\text{rel-dep}} + \\ &\left(\boldsymbol{u}_r^{\text{rel-head}}\right)^T \boldsymbol{h}_i^{\text{rel-head}} + \\ &\left(\boldsymbol{u}_r^{\text{rel-dep}}\right)^T \boldsymbol{h}_j^{\text{rel-dep}} + b_r \end{aligned} \quad (2)$$

Both $s_{ij}^{\text{arc}}$ and $s_{(i,j),r}^{\text{rel}}$ are trained greedily using cross-entropy loss with the correct head or label. At test time the final tree is composed using the Chu-Liu-Edmonds (CLE) maximum spanning tree algorithm (Chu and Liu, 1965; Edmonds, 1967).

[3] Though Wang and Eisner (2017) indicate that they can be predicted from unparsed corpora with reasonable accuracy.

| Language | B* | +$\mathbf{T}^*_S$ | Our Baseline | Selective Sharing | +$\mathbf{T}_L$ | +$\mathbf{T}_D$ | +$\mathbf{T}_S$ | Fine-tune |
|---|---|---|---|---|---|---|---|---|
| Basque | 49.89 | 54.34 | 56.18 | 56.54 | 56.35† | 56.77 | 56.50 | 60.71 |
| Croatian | 65.03 | 67.78 | 74.86 | 75.23 | 74.07 | 77.39 | 75.20 | 78.39 |
| Greek | 65.91 | 68.37 | 70.09 | 70.49 | 68.05 | 71.66 | 70.47 | 73.35 |
| Hebrew | 62.58 | 66.27 | 68.85 | 68.61 | 72.02 | 72.75 | 69.21 | 73.88 |
| Hungarian | 58.50 | 64.13 | 63.81 | 64.78 | 70.28 | 66.40 | 64.21 | 72.50 |
| Indonesian | 55.22 | 64.63 | 63.68 | 64.96 | 69.73 | 67.73 | 66.25 | 73.34 |
| Irish | 58.58 | 61.51 | 61.72 | 61.49† | 65.88 | 66.49 | 62.21 | 66.76 |
| Japanese | 54.97 | 60.41 | 57.28 | 57.80 | 63.83 | 64.28 | 57.04 | 72.72 |
| Slavonic | 68.79 | 71.13 | 75.18 | 75.17† | 74.65 | 74.17 | 75.16† | 73.11 |
| Persian | 40.38 | 34.20 | 53.87 | 53.61 | 45.14 | 56.72 | 53.03 | 59.92 |
| Polish | 72.15 | 76.85 | 76.01 | 75.93† | 79.51 | 71.09 | 76.29 | 77.78 |
| Romanian | 66.55 | 69.69 | 73.00 | 73.40 | 75.20 | 76.34 | 73.82 | 75.15 |
| Slovenian | 72.21 | 76.06 | 81.21 | 80.99 | 81.39 | 81.36 | 80.92 | 82.43 |
| Swedish | 72.26 | 75.32 | 79.39 | 79.64 | 80.28 | 80.10 | 79.22 | 81.29 |
| Tamil | 51.59 | 57.53 | 57.81 | 58.85 | 59.70 | 60.37 | 58.39 | 62.94 |
| Average | 60.97 | 64.55 | 67.53 | 67.83 | 69.07 | 69.57 | 67.86 | 72.28 |

Table 1: A comparison of all methods on held-out test languages. UAS results are reported over the *train* splits of the held-out languages, following (Wang and Eisner, 2018). B* and +$\mathbf{T}^*_S$ are the baseline and surface statistics model results, respectively, of (Wang and Eisner, 2018).[4] *Fine-tune* is the result of adapting our baseline model using only 10 sentences from the target language. All of our reported numbers are the average of three runs with different random seeds. Results with differences that are statistically *insignificant* compared to the baseline are marked with † (arc-level paired permutation test with $p \geq 0.05$).

## 4 Typology Augmented Parsing

**Selective Sharing:** Naseem et al. (2012) introduced the idea of *selective sharing* in a generative parser, where the features provided to a parser were controlled by its typology. The idea was extended to discriminative models by Täckström et al. (2013). For neural parsers which do not rely on manually-defined feature templates, however, there isn't an explicit way of using selective sharing. Here we choose to directly incorporate selective sharing as a bias term for arc-scoring:

$$s_{ij}^{\text{arc-aug}} = s_{ij}^{\text{arc}} + \boldsymbol{v}^\top \boldsymbol{f}_{ij} \tag{3}$$

where $\boldsymbol{v}$ is a learned weight vector and $\boldsymbol{f}_{ij}$ is a feature vector engineered using Täckström et al.'s head-modifier feature templates (Appendix B).

**Input Features:** We follow Wang and Eisner (2018) and encode the typology for language $l$ with an MLP, and concatenate it with each input:

$$\Phi = W_2 \cdot \tanh\left(W_1 \cdot \mathbf{T}^{(l)} + \boldsymbol{b}\right) \tag{4}$$

$$\boldsymbol{h} = \texttt{BiLSTM}\left(\{\boldsymbol{p}_1 \oplus \Phi, \ldots, \boldsymbol{p}_n \oplus \Phi\}\right) \tag{5}$$

This approach assumes the parser is able to learn to use information in $\mathbf{T}^{(l)} \in \{\mathbf{T}_L^{(l)}, \mathbf{T}_D^{(l)}, \mathbf{T}_S^{(l)}\}$ to induce some distinctive change in encoding $\boldsymbol{h}$.

## 5 Experiments

**Data:** We conduct our analysis on the Universal Dependencies v1.2 dataset (Nivre et al., 2015) and follow the same train-test partitioning of languages as Wang and Eisner (2018). We train on 20 treebanks and evaluate cross-lingual performance on the other 15; test languages are shown in Table 1.[5] We perform hyper-parameter tuning via 5-fold cross-validation on the training languages.

**Results:** Table 1 presents our cross-lingual transfer results. Our baseline model improves over the benchmark in (Wang and Eisner, 2018) by more than 6%. As expected, using typology yields mixed results. Selective sharing provides little to no benefit over the baseline. Incorporating the typology vector as an input feature is more effective, with the Liu Directionalities ($\mathbf{T}_D$) driving the most measurable improvements — achieving statistically significant gains on 13/15 languages. The Linguistic Typology ($\mathbf{T}_L$) gives statistically significant gains on 10/15 languages. Nevertheless, the results are still modest. Fine-tuning on only 10 sentences yields a **2.3× larger** average UAS increase, a noteworthy point of reference.

[4] Wang and Eisner (2018)'s final $\mathbf{T}^*_S$ also contains additional neural features that we omitted, as we found it to underperform using only hand-engineered features.

[5] Two treebanks are excluded from evaluation, following the setting of Wang and Eisner (2018).

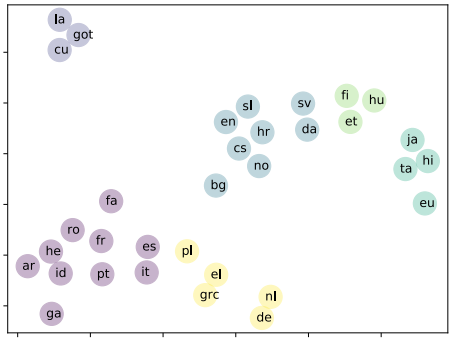


Figure 1: t-SNE projection of WALS vectors with clustering. Persian (fa) is an example of a poorly performing language that is also far from its cluster center.

## 6 Analysis

**Typology as Quantization:** Adding simple, discrete language identifiers to the input has been shown to be useful in multi-task multi-lingual settings (Ammar et al., 2016; Johnson et al., 2017). We hypothesize that the model utilizes typological information for a similar purpose by clustering languages by their parsing behavior. Testing this to the extreme, we encode languages using one-hot representations of their cluster membership. The clusters are computed by applying K-Means[6] to WALS feature vectors (see Figure 1 for an illustration). In this sparse form, compositional aspects of cross-lingual sharing are erased. Performance using this impoverished representation, however, only suffers slightly compared to the original — dropping by just $0.56\%$ UAS overall and achieving statistically significant parity or better with $\mathbf{T}_L$ on $7/15$ languages. A gap does still partially remain; future work may investigate this further.

This phenomenon is also reflected in the performance when the original WALS features are used. Test languages that do belong to compact clusters have higher performance on average than that of those who are isolates (e.g., Persian, Basque). Indeed from Table 1 and Fig. 1 we observe that the worst performing languages are isolated from their cluster centers. Even though their typology vectors can be viewed as compositions of training languages, the model appears to have limited generalization ability. This suggests that the model does not effectively use individual typological features.

This can likely be attributed to the training routine, which poses two inherent difficulties: 1) the parser has few examples (entire languages) to generalize from, making it hard from a learning perspective and 2) a naïve encoder can already implicitly capture important typological features within its hidden state, using only the surface forms of the input. This renders the additional typology features redundant. Table 2 presents the results of probing the final max-pooled output of the `BiLSTM` encoder for typological features on a *sentence level*. We find they are nearly linearly separable — logistic regression achieves greater than $90\%$ accuracy on average.

| WALS ID | 82A | 83A | 85A | 86A | 87A | 88A |
|---|---|---|---|---|---|---|
| Logreg | 87 | 85 | 97 | 92 | 94 | 92 |
| Majority | 61 | 56 | 87 | 75 | 51 | 82 |

Table 2: Performance of typology prediction using hidden states of the parser's encoder, compared to a majority baseline which predicts the most frequent category.

| +GD | B* | +$\mathbf{T}^*_S$ | Baseline | +$\mathbf{T}^\ddagger_L$ | +$\mathbf{T}_D$ | +$\mathbf{T}_S$ |
|---|---|---|---|---|---|---|
| Average | – | 67.11 | 68.45 | 69.23 | 68.36 | 67.12 |

Table 3: Average UAS results when training with Galactic Dependencies. The Linguistic Typology ($\mathbf{T}^\ddagger_L$) here is computed directly from the corpora using the rules in Appendix E. All of our reported numbers are the average of three runs.

Wang and Eisner (2018) attempt to address the learning problem by using the synthetic Galactic Dependencies (GD) dataset (Wang and Eisner, 2016) as a form of data augmentation. GD constructs "new" treebanks with novel typological qualities by systematically permuting the behaviors of real languages. Following their work, we add $8,820$ GD treebanks synthesized from the 20 UD training languages, giving $8,840$ training treebanks in total. Table 3 presents the results of training on this setting. While GD helps the weaker $\mathbf{T}^*_S$ substantially, the same gains are not realized for models built on top of our stronger baseline — in fact, the baseline only narrows the gap even further by increasing by $0.92\%$ UAS overall.[7]

**Typology Quality:** The notion of typology is predicated on the idea that some language features are consistent across different language samples, yet in practice this is not always the case. For instance, *Arabic* is listed in WALS as `SV` (82A, `Subject`$\curvearrowright$`Verb`), yet follows a large number of `Verb`$\curvearrowright$`Subject` patterns in UD v1.2. Fig. 2 fur-

[6] We use Euclidean distance as our metric, another extreme simplification. There is no guarantee that all dimensions should be given equal weight, as indicated in Table 4.

[7] Sourcing a greater number of real languages may still be helpful. The synthetic GD setting is not entirely natural, and might be sensitive to hyper-parameters.

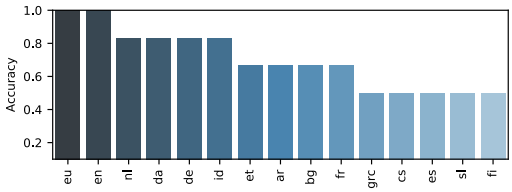


Figure 2: Averaged matching accuracy of the linguistically-defined WALS features on 15 randomly sampled languages compared to their corpus-specific values derived from UD v1.2. Rules for deriving the features from corpus are described in Appendix E.

ther demonstrates that for some languages these divergences are significant (see Appendix F for concrete examples). Given this finding, we are interested in measuring the impact this noise has on typology utilization. Empirically, $\mathbf{T}_D$, which is consistent with the corpus, performs best. Furthermore, updating our typology features for $\mathbf{T}_L$ to match the dominant ordering of the corpus yields a slight improvement of $0.21\%$ UAS overall, with statistically significant gains on $7/15$ languages.

In addition to the quality of the representation, we can also analyze the impact of its resolution. In theory, a richer, high-dimensional representation of typology may capture subtle variations. In practice, however, we observe an opposite effect, where the Linguistic Typology ($\mathbf{T}_L$) and the Liu Directionalities ($\mathbf{T}_D$) outperform the surface statistics ($\mathbf{T}_S$), with $|\mathbf{T}_L| \approx |\mathbf{T}_D| \ll |\mathbf{T}_S|$. This is likely due to the limited number of languages used for training (though training on GD exhibits the same trend). This suggests that future work may consider using targeted dimensionality reduction mechanisms, optimized for performance.

**Typology vs. Parser Transferability:** The implicit assumption of all the typology based methods is that the typological similarity of two languages is a good indicator of their parsing transferability. As a measure of parser transferability, for each language we select the oracle source language which results in the best transfer performance. We then compute precision@$k$ for the nearest $k$ neighbors in the typological space, i.e. whether the best source appears in the $k$ nearest neighbors. As shown in Table 4, we observe that while there is some correlation between the two, they are far from perfectly aligned. $\mathbf{T}_D$ has the best alignment, which is consistent with its corresponding best parsing performance. Overall, this divergence motivates the development of approaches that better match the two distributions.

| | P@1 | P@3 | P@5 | P@10 |
|---|---|---|---|---|
| $\mathbf{T}_L$ | 13 | 33 | 60 | 80 |
| $\mathbf{T}_D$ | 27 | 67 | 67 | 93 |
| $\mathbf{T}_S$ | 13 | 27 | 27 | 73 |

Table 4: Precision@$k$ for identifying the best *parsing* transfer language, for the $k$ typological neighbors.

## 7 Related Work

Other recent progress in cross-lingual parsing has focused on lexical alignment (Guo et al., 2015, 2016; Schuster et al., 2019). Data augmentation (Wang and Eisner, 2017) is another promising direction, but at the cost of greater training demands. Both directions do not directly address structure. Ahmad et al. (2019) showed structural-sensitivity is important for modern parsers; insensitive parsers suffer. Data transfer is an alternative solution to alleviate the typological divergences, such as annotation projection (Tiedemann, 2014) and source treebank reordering (Rasooli and Collins, 2019). These approaches are typically limited by parallel data and imperfect alignments. Our work aims to understand cross-lingual parsing in the context of model transfer, with typology serving as language descriptors, with the goal of eventually addressing the issue of structure.

## 8 Conclusion

Realizing the potential for typology may require rethinking current approaches. We can further drive performance by refining typology-based similarities into a metric more representative of actual transfer quality. Ultimately, we would like to design models that can directly leverage typological compositionality for distant languages.

## Acknowledgments

We thank Dingquan Wang, Jason Eisner, the MIT NLP group (special thanks to Jiaming Luo), and the reviewers for their valuable comments. This research is supported in part by the Office of the Director of National Intelligence, Intelligence Advanced Research Projects Activity, via contract #FA8650-17-C-9116, and the National Science Foundation Graduate Research Fellowship under Grant #1122374. Any opinion, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the the NSF, ODNI, IARPA, or the U.S. Government. We are also grateful for the support of MIT Quest for Intelligence program.

## References


Wasi Ahmad, Zhisong Zhang, Xuezhe Ma, Eduard Hovy, Kai-Wei Chang, and Nanyun Peng. 2019. On difficulties of cross-lingual transfer with order differences: A case study on dependency parsing. In *NAACL*, pages 2440–2452.

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A Smith. 2016. Many languages, one parser. *TACL*, 4:431–444.

Emily M. Bender. 2011. On achieving and evaluating language-independence in nlp. *Linguistic Issues in Language Technology*.

Yoeng-Jin Chu and Tseng-Hong Liu. 1965. On the shortest arborescence of a directed graph. *Scientia Sinica*, 14:1396–1400.

Timothy Dozat and Christopher D. Manning. 2017. Deep Biaffine Attention for Neural Dependency Parsing. In *ICLR*.

Matthew S. Dryer and Martin Haspelmath, editors. 2013. *WALS Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B:233–240.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *ACL-IJCNLP*, volume 1, pages 1234–1244.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2016. A representation learning framework for multi-source transfer parsing. In *AAAI*.

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2017. Googles multilingual neural machine translation system: Enabling zero-shot translation. *TACL*, 5:339–351.

Haitao Liu. 2010. Dependency direction as a means of word-order typology: A method based on dependency treebanks. *Lingua*, 120(6):1567–1578.

Tahira Naseem, Regina Barzilay, and Amir Globerson. 2012. Selective sharing for multilingual dependency parsing. In *ACL*, pages 629–637. Association for Computational Linguistics.

Joakim Nivre, Željko Agić, Maria Jesus Aranzabe, Masayuki Asahara, Aitziber Atutxa, Miguel Ballesteros, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Cristina Bosco, Sam Bowman, Giuseppe G. A. Celano, Miriam Connor, Marie-Catherine de Marneffe, Arantza Diaz de Ilarraza, Kaja Dobrovoljc, Timothy Dozat, Tomaž Erjavec, Richárd Farkas, Jennifer Foster, Daniel Galbraith, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Yoav Goldberg, Berta Gonzales, Bruno Guillaume, Jan Hajič, Dag Haug, Radu Ion, Elena Irimia, Anders Johannsen, Hiroshi Kanayama, Jenna Kanerva, Simon Krek, Veronika Laippala, Alessandro Lenci, Nikola Ljubešić, Teresa Lynn, Christopher Manning, Cătălina Mărănduc, David Mareček, Héctor Martínez Alonso, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Anna Missilä, Verginica Mititelu, Yusuke Miyao, Simonetta Montemagni, Shunsuke Mori, Hanna Nurmi, Petya Osenova, Lilja Øvrelid, Elena Pascual, Marco Passarotti, Cenel-Augusto Perez, Slav Petrov, Jussi Piitulainen, Barbara Plank, Martin Popel, Prokopis Prokopidis, Sampo Pyysalo, Loganathan Ramasamy, Rudolf Rosa, Shadi Saleh, Sebastian Schuster, Wolfgang Seeker, Mojgan Seraji, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Kiril Simov, Aaron Smith, Jan Štěpánek, Alane Suhr, Zsolt Szántó, Takaaki Tanaka, Reut Tsarfaty, Sumire Uematsu, Larraitz Uria, Viktor Varga, Veronika Vincze, Zdeněk Žabokrtský, Daniel Zeman, and Hanzhi Zhu. 2015. Universal dependencies 1.2. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Helen O'Horan, Yevgeni Berzak, Ivan Vulic, Roi Reichart, and Anna Korhonen. 2016. Survey on the use of typological information in natural language processing. In *COLING*, pages 1297–1308.

Mohammad Sadegh Rasooli and Michael Collins. 2019. Low-resource syntactic transfer with unsupervised source reordering. In *NAACL*, pages 3845–3856.

Manon Scholivet, Franck Dary, Alexis Nasr, Benoit Favre, and Carlos Ramisch. 2019. Typological features for multilingual delexicalised dependency parsing. In *NAACL*, pages 3919–3930.

Tal Schuster, Ori Ram, Regina Barzilay, and Amir Globerson. 2019. Cross-lingual alignment of contextual word embeddings, with applications to zero-shot dependency parsing. In *NAACL*, pages 1599–1613.

Oscar Täckström, Ryan McDonald, and Joakim Nivre. 2013. Target language adaptation of discriminative transfer parsers. In *NAACL*, pages 1061–1071, Atlanta, Georgia. Association for Computational Linguistics.

Jörg Tiedemann. 2014. Rediscovering annotation projection for cross-lingual parser induction. In *COLING*, pages 1854–1864.

Dingquan Wang and Jason Eisner. 2016. The galactic dependencies treebanks: Getting more data by synthesizing new languages. *TACL*, 4:491–505.

Dingquan Wang and Jason Eisner. 2017. Fine-grained prediction of syntactic typology: Discovering latent structure with supervised learning. *TACL*, 5:147–161.

Dingquan Wang and Jason Eisner. 2018. Surface statistics of an unknown language indicate how to parse it. *TACL*, 6:667–685.

Yuan Zhang and Regina Barzilay. 2015. Hierarchical low-rank tensors for multilingual transfer parsing. In *EMNLP*.

# Cross-Lingual BERT Transformation for Zero-Shot Dependency Parsing

**Yuxuan Wang**[1], **Wanxiang Che**[1]*, **Jiang Guo**[2], **Yijia Liu**[1], **and Ting Liu**[1]
[1]Research Center for Social Computing and Information Retrieval,
Harbin Institute of Technology
[2]Computer Science and Artificial Intelligence Laboratory, MIT
{yxwang,car,yjliu,tliu}@ir.hit.edu.cn,jiang_guo@csail.mit.edu

## Abstract

This paper investigates the problem of learning cross-lingual representations in a contextual space. We propose **C**ross-**L**ingual **B**ERT **T**ransformation (CLBT), a simple and efficient approach to generate cross-lingual contextualized word embeddings based on publicly available pre-trained BERT models (Devlin et al., 2018). In this approach, a linear transformation is learned from contextual word alignments to align the contextualized embeddings independently trained in different languages. We demonstrate the effectiveness of this approach on zero-shot cross-lingual transfer parsing. Experiments show that our embeddings substantially outperform the previous state-of-the-art that uses static embeddings. We further compare our approach with XLM (Lample and Conneau, 2019), a recently proposed cross-lingual language model trained with massive parallel data, and achieve highly competitive results. [1]

## 1 Introduction

One of the most promising directions for cross-lingual dependency parsing, which also remains a challenge, is to bridge the gap of lexical features. Prior works (Xiao and Guo, 2014; Guo et al., 2015) have shown that cross-lingual word embeddings are able to significantly improve the transfer performance compared to delexicalized models (McDonald et al., 2011, 2013). These cross-lingual word embeddings are *static* in the sense that they do not change with the context.[2]

Recently, contextualized word embeddings derived from large-scale pre-trained language models (McCann et al., 2017; Peters et al., 2017, 2018; Devlin et al., 2018) have demonstrated dramatic superiority over traditional static word embeddings, establishing new state-of-the-arts in various monolingual NLP tasks (Ilić et al., 2018; Schuster et al., 2018). The success has also been recognized in dependency parsing (Che et al., 2018). The great potential of these contextualized embeddings has inspired us to extend its power to cross-lingual scenarios.

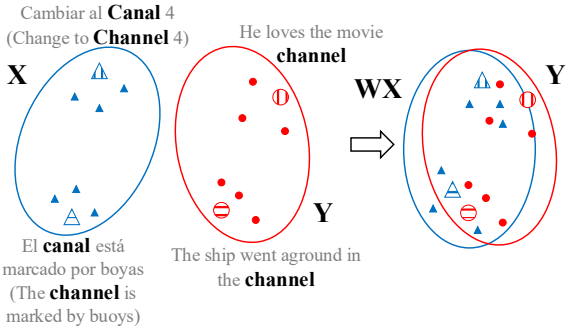


Figure 1: A toy illustration of the method, where contextualized embeddings of the word *canal* from Spanish is transformed to the semantic space of English.

Several recent works have been proposed to learn contextualized cross-lingual embeddings by training cross-lingual language models from scratch with parallel data as supervision, and has been demonstrated effective in several downstream tasks (Schuster et al., 2018; Mulcaire et al., 2019; Lample and Conneau, 2019). These methods are typically resource-demanding and time-consuming.[3] In this paper, we propose **C**ross-**L**ingual **B**ERT **T**ransformation (CLBT), a simple and efficient off-line approach that learns a linear transformation from contextual word alignments. With CLBT, contextualized embeddings

*Email corresponding

[1]Our code is released at https://github.com/WangYuxuan93/CLBT

[2]In this paper, we refer to these embeddings as *static* as opposed to *contextualized* ones.

[3]For instance, XLM was trained on 64 Volta GPUs (Lample and Conneau, 2019). While the time of training is not described in the paper, we may take the statistics from BERT as a reference, e.g., $\text{BERT}_{\text{BASE}}$ was trained on 4 Cloud TPUs for 4 days (Devlin et al., 2018).



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5721–5727,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

from pre-trained BERT models in different languages are projected into a shared semantic space. The learned transformation is then used on top of the BERT encodings for each sentence, which are further fed as input to a parser.

Our approach utilizes the semantic equivalence in word alignments, and thus is supposed to be *word sense-preserving*. Figure 1 illustrates our approach, where contextualized embeddings of the Spanish word "canal" are transformed to the corresponding semantic space in English.

Experiments on the Universal Dependencies (UD) treebanks (v2.2) (Nivre et al., 2018) show that our approach substantially outperforms previous models that use static cross-lingual embeddings, with an absolute gain of 2.91% in averaged LAS. We further compare to XLM (Lample and Conneau, 2019), a recently proposed large-scale cross-lingual language model. Results demonstrate that our approach requires significantly fewer training data, computing resources and less training time than XLM, yet achieving highly competitive results.

## 2 Related Work

Static cross-lingual embedding learning methods can be roughly categorized as *on-line* and *off-line* methods. Typically, on-line approaches integrate monolingual and cross-lingual objectives to learn cross-lingual word embeddings in a joint manner (Klementiev et al., 2012; Kočiský et al., 2014; Guo et al., 2016), while off-line approaches take pre-trained monolingual word embeddings of different languages as input and retrofit them into a shared semantic space (Xing et al., 2015; Lample et al., 2018; Chen and Cardie, 2018).

Several approaches have been proposed recently to connect the rich expressiveness of contextualized word embeddings with cross-lingual transfer. Mulcaire et al. (2019) based their model on ELMo (Peters et al., 2018) and proposed a polyglot contextual representation model by capturing character-level information from multilingual data. Lample and Conneau (2019) adapted the objectives of BERT (Devlin et al., 2018) to incorporate cross-lingual supervision from parallel data to learn cross-lingual language models (XLMs), which have obtained state-of-the-art results on several cross-lingual tasks. Similar to our approach, Schuster et al. (2019) also aligned pre-trained contextualized word embeddings through linear transformation in an off-line fashion. They used the averaged contextualized embeddings as an *anchor* for each word type, and learn a transformation in the anchor space. Our approach, however, learns this transformation directly in the contextual space, and hence is explicitly designed to be *word sense-preserving*.

## 3 Cross-Lingual BERT Transformation

This section describes our proposed approach, namely CLBT, to transform pre-trained monolingual contextualized embeddings to a shared semantic space.

### 3.1 Contextual Word Alignment

Traditional methods of learning static cross-lingual word embeddings have been relying on various sources of supervision such as bilingual dictionaries (Lazaridou et al., 2015; Smith et al., 2017), parallel corpus (Guo et al., 2015) or on-line Google Translate (Mikolov et al., 2013; Xing et al., 2015). To learn contextualized cross-lingual word embeddings, however, we require supervision at word token-level (or context-level) rather than type-level (i.e. dictionaries). Therefore, we assume a parallel corpus as our supervision, analogous to on-line methods such as XLM (Lample and Conneau, 2019).

In our approach, unsupervised bidirectional word alignment is first applied to the parallel corpus to obtain a set of aligned word pairs with their contexts, or *contextual word pairs* for short. For one-to-many and many-to-one alignments, we use the left-most aligned word,[4] such that all the resulting word pairs are one-to-one. In practice, since WordPiece embeddings (Wu et al., 2016) are used in BERT, all the parallel sentences are tokenized using BERT's wordpiece vocabulary before being aligned.

### 3.2 Off-Line Transformation

Given a set of contextual word pairs, their BERT representations $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{n}$ can be easily obtained from pre-trained BERT models,[5] where $\mathbf{x}_i \in \mathbb{R}^{d_1}$ is the contextualized embedding of token $i$ in the

[4] Preliminary experiments indicate that this way works better than keeping all the alignments.

[5] In this work, we use the English BERT (enBERT) for the source language (English) and the multilingual BERT (mBERT), which is trained on 102 languages without cross-lingual supervision, for all the target languages.

target language, and $\mathbf{y}_i \in \mathbb{R}^{d_2}$ is the representation of its alignment in the source language.

In our experiments, a parser is trained on source language data and applied directly to all the target languages. Therefore, we propose to project the embeddings of target languages to the space of the source language, instead of the opposite direction. Specifically, we aim at finding an appropriate linear transformation $\mathbf{W}$, such that $\mathbf{W}\mathbf{x}_i$ approximates $\mathbf{y}_i$.[6] This can be achieved by solving the following optimization problem:

$$\min_{\mathbf{W}} \sum_{i=1}^{n} ||\mathbf{W}\mathbf{x}_i - \mathbf{y}_i||^2,$$

where $\mathbf{W} \in \mathbb{R}^{d_1 \times d_2}$ is a parameter matrix.

Previous works on static cross-lingual embeddings have shown that an orthogonal $\mathbf{W}$ (i.e. $\mathbf{W}^\top\mathbf{W} = \mathbf{I}$) is helpful for the word translation task (Xing et al., 2015). In this case, an analytical solution can be found through singular value decomposition (SVD) of $\mathbf{Y}^\top\mathbf{X}$:

$$\mathbf{W} = \mathbf{V}\mathbf{U}^\top, \text{where } \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top = \text{SVD}(\mathbf{Y}^\top\mathbf{X}).$$

Here $\mathbf{X} \in \mathbb{R}^{n \times d}$ and $\mathbf{Y} \in \mathbb{R}^{n \times d}$ are the contextualized embedding matrices, where $n$ is the number of aligned contextual word pairs, $d$ is the dimension of monolingual contextualized embeddings. Each pair of rows $(\mathbf{x}_i, \mathbf{y}_i)$ indicates an aligned contextual word pair.

Although this can be computed in CPUs within several minutes, more memories will be required with the growth of the amount of training data. Therefore, we present an approximate solution, where $\mathbf{W}$ is optimized with gradient decent (GD) and is not constrained to be orthogonal.[7] This GD-based approach can be trained on a single GPU and typically converges in several hours.

To validate the effectiveness of our approach in cross-lingual dependency parsing, we first obtain the CLBT embeddings with the proposed approach, and then use them as input to a modern graph-based neural parser (described in next section), in replacement of the pre-trained static embeddings. Note that BERT produces embeddings in wordpiece-level, so we only use the left-most wordpiece embedding of each word.[8]

[6] We also investigated non-linear transformation in our experiments, but didn't observe any improvements.

[7] We found the orthogonal constraint doesn't help for GD.

[8] We tried alternative strategies such as averaging, using the middle or right-most wordpiece, but observed no significant difference.

# 4 Experiments

## 4.1 Data and Settings

In our experiments, the contextual word pairs are obtained from the Europarl corpora (Koehn, 2005) using the *fast_align* toolkit (Dyer et al., 2010). Only 10,000 sentence pairs are used for each target language. For the parsing datasets, we use the Universal Dependencies(UD) Treebanks (v2.2) (Nivre et al., 2018),[9] following the settings of the previous state-of-the-art system (Ahmad et al., 2018). From the 31 languages they have analyzed, we select 18 whose Europarl data is publicly available.[10] Statistics of the selected languages and treebanks can be found in the Appendix. We employ the Biaffine Graph-based Parser of Dozat and Manning (2017) and adopt their hyper-parameters for all of our models.

In all the experiments, English is used as the source language, and the other 17 languages as targets. The model is trained on the English treebank and applied directly to target languages with the transformed contextualized embeddings. We train our models using the Adam optimizer (Kingma and Ba, 2015), and most of the them converge within a few thousand epochs in several hours. More implementation details are reported in the Appendix.

## 4.2 Baseline Systems

We compare our method with the following three baseline models:

- **mBERT** (contextualized). Embeddings generated by the mBERT model are directly used in the training and testing procedures.
- **FT-SVD** (Ahmad et al., 2018, off-line, static). SVD-based transformation (Smith et al., 2017) is applied on 300-dimensional FastText embeddings (Bojanowski et al., 2017) to obtain cross-lingual static embeddings, which represents the previous state-of-the-art. We report results from their paper of the RNNGraph model which used the same architecture as ours.
- **XLM** (Lample and Conneau, 2019, on-line, contextualized). A strong method which learns contextualized cross-lingual embeddings from scratch with cross-lingual data.

[9] hdl.handle.net/11234/1-2837

[10] For languages with multiple treebanks, we use the same combinations as they did.

| Lan. | Static | Contextualized | | |
|---|---|---|---|---|
| | **FT-SVD** | **mBERT** | **CLBT (SVD)** | **CLBT (GD)** |
| en | 88.31 | 90.71 | **91.03*** | |
| de | 59.31 | 63.41 | **64.47*** | 62.14 |
| da | 68.81 | 70.57 | 71.60* | **71.66*** |
| sv | 73.49 | 70.09 | 73.33* | **75.95*** |
| nl | 60.11 | **65.66** | 65.45 | 63.86 |
| fr | 73.46 | 72.97 | 74.70* | **76.59*** |
| it | 76.23 | 79.02 | **79.46** | 78.98 |
| es | 66.91 | 65.43 | 67.14* | **68.33*** |
| pt | 67.98 | 67.11 | 69.12* | **69.25*** |
| ro | 52.11 | 46.40 | 55.14* | **55.84*** |
| sk | 56.98 | 50.76 | 59.46* | **59.92*** |
| pl | 58.59 | 63.10 | 65.37* | **65.80*** |
| bg | 66.68 | **71.20** | 70.26 | 70.75 |
| sl | 54.57 | 56.78 | **57.42*** | 57.21* |
| cs | 52.80 | 45.20 | 52.20* | **52.99*** |
| fi | 48.74 | 49.56 | 51.00* | **52.61*** |
| et | 44.40 | 46.64 | 47.79* | **48.52*** |
| lv | 49.59 | 45.11 | 48.59* | **49.78*** |
| AVG. | 60.63 | 60.53 | 63.09 | **63.54** |

Table 1: Results (LAS%) on test sets. Languages are split by language families with dashed lines. AVG. means the average of results from all target languages. Statistically significant differences between our methods and the mBERT model are marked with *, with p-value $< 0.05$ under McNemar's test.

For the **XLM** model, we employ the XNLI-15 model[11] they released to generate embeddings and apply them to cross-lingual dependency parsing in the same way as we do with our own model. We compare with them in the 4 overlapped languages both works have researched on.

### 4.3 Comparison with Off-Line Methods

Results on the test sets are shown in Table 1.[12] Languages are grouped by language families. Overall, our approach with either SVD or GD outperforms both FT-SVD and mBERT by a substantial margin (+2.91% in averaged LAS), among which GD turns out to be slightly better than SVD in most of the languages. When combined with FT-SVD, the performances can be further improved by 0.33% in LAS for the GD method and 0.51% for SVD (see the Appendix for more details). Interestingly, the mBERT model which is trained without any cross-lingual supervision but using a shared multilingual wordpiece vocabulary works surprisingly well in some languages, especially in those linguistically close to English. Similar observations have also been identified in other works (Pires et al., 2019; Wu and Dredze, 2019).

[11] github.com/facebookresearch/XLM

[12] UAS results are listed in the Appendix due to space limit. Note that since we have no access to the parsed files of the FT-SVD model, we only report statistical significant tests between our methods and the mBERT model, which is highly comparable to the FT-SVD model on average.

| Lan. | XLM | CLBT (SVD) | CLBT (GD) |
|---|---|---|---|
| en | 91.85/89.92 | **92.81*/91.03*** | |
| de | **74.65/65.31** | 73.72/64.47 | 71.08/62.14 |
| fr | 79.62/73.41 | 80.01/74.70* | **80.85*/76.59*** |
| es | 75.41/67.43 | 75.52/67.14* | **75.70*/68.33*** |
| bg | 81.07/69.45 | **82.14***/70.26 | 81.51/**70.75*** |
| AVG. | 77.69/68.90 | **77.85**/69.14 | 77.29/**69.45** |
| **Data** | 0.2-13.1M | 10K | |

Table 2: Results (UAS%/LAS%) on test sets. The last row shows the training data used in each language by sentence. AVG. means the average of results from 4 target languages. Statistically significant differences between our methods and the XLM are marked with an asterisk, with p-value $< 0.05$ under McNemar's test.

### 4.4 Comparison with On-Line Methods

Comparison of our approach and a cross-lingual language model pre-training (XLM) method (Lample and Conneau, 2019) in the 4 overlapped languages is shown in Table 2. CLBT outperforms XLM in 3 out of the 4 languages but lower in German (de). The amount of training data used in each method is also shown in the bottom: the number of parallel sentences used by XLM ranges from 0.2 million (10 million tokens) for Bulgarian to 13.1 million (682 million tokens) for French. In comparison, only 10,000 parallel sentences (0.4 million tokens) are used for each language in CLBT, demonstrating the data-efficiency of our approach. Moreover, given the efficiency in both data and training, CLBT can be readily scaled to new language pairs in hours.

### 4.5 Analysis

#### 4.5.1 Transformation of Cross-lingual BERT Embedding

In order to investigate the properties of contextualized representations before and after the linear transformation, we employ the SENSEVAL2 data (Edmonds and Cotton, 2001),[13] where words from different languages are tagged by their word senses in different contexts.

We took contextualized representations of the English word *nature* and its Spanish transla-

[13] www.hipposmond.com/senseval2/

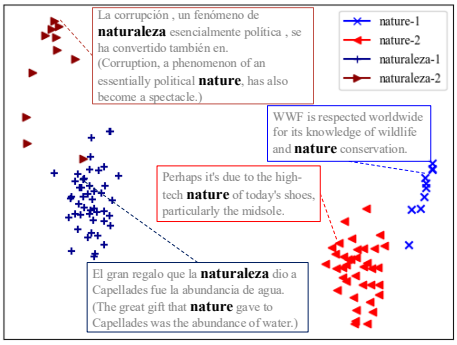


(a)

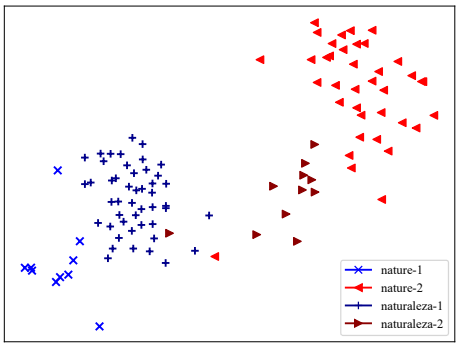


(b)

Figure 2: t-SNE visualization of the English word *nature* and its Spanish translation *naturaleza* in different contexts by the contextualized representations before (a) and after (b) the linear transformation. Points are colored by word senses. Example contexts are given in (a). Translations of Spanish sentences are in brackets.

tion *naturaleza* in different contexts from pre-trained English and multilingual BERT respectively and visualize their distributions in Figure 2(a), where we can observe obvious clustering of word senses. Specifically, words with sense *nature-1* and *naturaleza-1* mean *the physical world*, whereas *nature-2* and *naturaleza-2* mean *inherent features*. We then apply our GD-based method to embeddings of *naturaleza* and depict the resulting cross-lingual embeddings in Figure 2(b). The distance between embeddings from English and Spanish is effectively reduced after the transformation. And it is apparent that embeddings of Spanish words are closer to those with similar meanings from English, which indicates the effectiveness of our approach.

#### 4.5.2 Effect of Training Data Size

We select several languages from each language family, and investigate the effect of the amount of training data on the performances of zero-shot cross-lingual dependency parsing. Specifically,

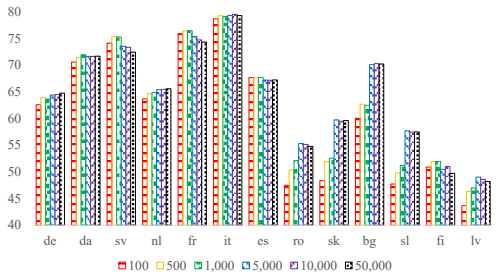


Figure 3: Effects of the amount of training data on different languages. ($y$-axis represents the LAS.)

we take the SVD-based approach, since it is faster than the GD-based one, and trained different transformation models with different amount of parallel sentences from Europarl dataset on each of the 13 selected languages.

As shown in Figure 3, for most of the languages, the best performance is achieved with only 5000 parallel sentences. It is also worth noting that for most of Germanic (e.g. German, Danish, Swedish and Dutch) and Romance (e.g. French, Italian, Spanish and Romanian) languages, which are typologically closer to English, a rather small training set of merely 100 sentences is capable of achieving comparative results.

## 5 Conclusion

We propose the Cross-Lingual BERT Transformation (CLBT) approach for contextualized cross-lingual embedding learning, which substantially outperforms the previous state-of-the-art in zero-shot cross-lingual dependency parsing. By exploiting publicly available pre-trained BERT models, our approach provides a fast and data-efficient solution to learning cross-lingual contextualized embeddings. Compared to the XLM, our method requires much fewer parallel data and less training time, yet achieving comparable performance.

For future work, we are interested in unsupervised cross-lingual alignment, inspired by prior success on static embeddings (Lample et al., 2018; Alvarez-Melis and Jaakkola, 2018), which demands a deeper understanding to the geometry of the multilingual contextualized embedding space.

## Acknowledgments

We thank the anonymous reviewers for their valuable suggestions. This work was supported by the National Natural Science Foundation of China (NSFC) via grant 61976072, 61632011 and 61772153.

## References


Wasi Uddin Ahmad, Zhisong Zhang, Xuezhe Ma, Eduard H. Hovy, Kai-Wei Chang, and Nanyun Peng. 2018. Near or far, wide range zero-shot cross-lingual dependency parsing. *CoRR*, abs/1811.00570.

David Alvarez-Melis and Tommi Jaakkola. 2018. Gromov-Wasserstein alignment of word embedding spaces. In *Proc. of EMNLP*, pages 1881–1890.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Wanxiang Che, Yijia Liu, Yuxuan Wang, Bo Zheng, and Ting Liu. 2018. Towards better ud parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. In *Proc. of CoNLL*, pages 55–64. Association for Computational Linguistics.

Xilun Chen and Claire Cardie. 2018. Unsupervised multilingual word embeddings. In *Proc. of EMNLP*, pages 261–270. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proc. of ICLR*.

Chris Dyer, Adam Lopez, Juri Ganitkevitch, Jonathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proc. of the ACL 2010 System Demonstrations*, pages 7–12. Association for Computational Linguistics.

Philip Edmonds and Scott Cotton. 2001. Senseval-2: Overview. In *The Proc. of the Second International Workshop on Evaluating Word Sense Disambiguation Systems*, SENSEVAL '01, pages 1–5, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *Proc. of ACL and IJCNLP*, pages 1234–1244.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2016. A representation learning framework for multi-source transfer parsing. In *Proc. of AAAI*, pages 2734–2740.

Suzana Ilić, Edison Marrese-Taylor, Jorge Balazs, and Yutaka Matsuo. 2018. Deep contextualized word representations for detecting sarcasm and irony. In *Proc. of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 2–7. Association for Computational Linguistics.

Diederick P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proc. of ICLR*.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing crosslingual distributed representations of words. In *Proc. of COLING*, pages 1459–1474.

Tomáš Kočiský, Karl Moritz Hermann, and Phil Blunsom. 2014. Learning bilingual word representations by marginalizing alignments. In *Proc. of ACL*, pages 224–229. Association for Computational Linguistics.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proc. of the tenth Machine Translation Summit*, pages 79–86, Phuket, Thailand. AAMT.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *CoRR*, abs/1901.07291.

Guillaume Lample, Alexis Conneau, Marc'Aurelio Ranzato, Ludovic Denoyer, and Herv Jgou. 2018. Word translation without parallel data. In *Proc. of ICLR*.

Angeliki Lazaridou, Georgiana Dinu, and Marco Baroni. 2015. Hubness and pollution: Delving into cross-space mapping for zero-shot learning. In *Proc. of ACL and IJCNLP*, pages 270–280. Association for Computational Linguistics.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems 30*, pages 6294–6305. Curran Associates, Inc.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee. 2013. Universal dependency annotation for multilingual parsing. In *Proc. of ACL*, pages 92–97. Association for Computational Linguistics.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proc. of EMNLP*, pages 62–72.

Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013. Exploiting similarities among languages for machine translation. *CoRR*, abs/1309.4168.

Phoebe Mulcaire, Jungo Kasai, and Noah Smith. 2019. Polyglot contextual representations improve crosslingual transfer. In *Proc. of NAACL*.

Joakim Nivre, Mitchell Abrams, Željko Agić, and et al. 2018. Universal dependencies 2.2. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proc. of ACL*, pages 1756–1765. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*, pages 2227–2237. Association for Computational Linguistics.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual BERT? In *Proc. of ACL*, pages 4996–5001.

Sebastian Schuster, Sonal Gupta, Rushin Shah, and Mike Lewis. 2018. Cross-lingual transfer learning for multilingual task oriented dialog. *CoRR*, abs/1810.13327.

Tal Schuster, Ori Ram, Regina Barzilay, and Amir Globerson. 2019. Cross-lingual alignment of contextual word embeddings, with applications to zero-shot dependency parsing. In *Proc. of NAACL*, pages 1599–1613.

Samuel L. Smith, David H. P. Turban, Steven Hamblin, and Nils Y. Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In *Proc. of ICLR*.

Shijie Wu and Mark Dredze. 2019. Beto, bentz, becas: The surprising cross-lingual effectiveness of bert. In *Proc. of EMNLP*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Min Xiao and Yuhong Guo. 2014. Distributed word representation learning for cross-lingual dependency parsing. In *Proc. of CoNLL*, pages 119–129. Association for Computational Linguistics.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *Proc. of NAACL*, pages 1006–1011. Association for Computational Linguistics.

# Multilingual Grammar Induction with Continuous Language Identification*

**Wenjuan Han**[†], **Ge Wang**[†], **Yong Jiang**[‡], **Kewei Tu**[†]
[†]School of Information Science and Technology, ShanghaiTech University, Shanghai, China
[‡]Alibaba Group
{hanwj,wangge,tukw}@shanghaitech.edu.cn
yongjiang.jy@alibaba-inc.com

## Abstract

The key to multilingual grammar induction is to couple grammar parameters of different languages together by exploiting the similarity between languages. Previous work relies on linguistic phylogenetic knowledge to specify similarity between languages. In this work, we propose a novel universal grammar induction approach that represents language identities with continuous vectors and employs a neural network to predict grammar parameters based on the representation. Without any prior linguistic phylogenetic knowledge, we automatically capture similarity between languages with the vector representations and softly tie the grammar parameters of different languages. In our experiments, we apply our approach to 15 languages across 8 language families and subfamilies in the Universal Dependency Treebank dataset, and we observe substantial performance gain on average over monolingual and multilingual baselines.

## 1 Introduction

Human languages bear striking resemblance at the syntactic level in spite of their diversity on the surface, as many studies have revealed (Greenberg, 1963; Hawkins, 2014). This fact provides the basis for multilingual grammar induction which tries to simultaneously induce grammars of multiple languages. Intuitively, one can couple grammar parameters of different languages with similar typology and learn them simultaneously. However, the lacking of measures of language similarity prevents this idea from being further exploited in practice.

Previous work in multilingual grammar induction either does not consider language similarity measures (Iwata et al., 2010) or models language similarity based on linguistic phylogeny (Berg-Kirkpatrick and Klein, 2010). The phylogenetic knowledge, however, could be misleading in measuring language similarity. For example, English and German are both Germanic languages, but English exhibits dominant Subject-Verb-Object (SVO) word order while German does not.

In this paper, we propose a novel approach to multilingual grammar induction. Our induction model represents language identities as continuous vectors (i.e., language embeddings) and employs a neural network to predict the grammar parameters of each language based on its embedding. The neural network parameters are universally shared across languages, which softly tie the grammar parameters of different languages. The language embeddings and the neural network parameters are trained with a standard grammar induction objective without any guidance from prior linguistic phylogenetic knowledge. We also introduce an auxiliary language identification task in which we predict the language identities of input sentences using the language embeddings.

We evaluate our approach on corpora of 15 languages across 8 language families and subfamilies. We observe that our approach achieves substantial performance gain on average over monolingual and multilingual baselines.

## 2 Dependency Model with Valence and Other Related Works

Dependency Model with Valence (DMV) (Klein and Manning, 2004) is the best known generative model for dependency grammar induction. The DMV generates a sentence and its dependency tree following three types of grammar rules (ATTACH, DECISION and ROOT). It firstly samples a token $c$ from the ROOT distribution $\mathrm{P}_{\mathrm{ROOT}}(c)$

*The first and second authors contributed equally. The third author contributed to this work when at ShanghaiTech University. The fourth author is the corresponding author.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5728–5733,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

and then recursively decides whether to generate a new child token and what child token to generate by sampling from the DECISION and ATTACH distributions $\mathrm{P}_{\text{DECISION}}(dec|h, dir, val)$ and $\mathrm{P}_{\text{ATTACH}}(c|h, dir)$, where $dir$ is a binary variable representing the direction of generation (left or right), $val$ is a binary variable representing whether the current head token already has a child in the direction $dir$ or not, $dec$ is a binary variable deciding whether to continue generation in the current direction, $c$ is the child token and $h$ is the head token.

Almost all previous methods of multilingual grammar induction are based on DMV. Their focus is on designing various priors to couple DMV parameters across languages: Cohen and Smith (2009) propose a logistic normal prior while Berg-Kirkpatrick and Klein (2010) design a hierarchical Gaussian prior according to linguistic phylogeny.

The usage of continuous language embeddings has been explored in other tasks. For example, Ammar et al. (2016) and de Lhoneux et al. (2018) apply language embeddings in supervised multilingual dependency parsing.

## 3 Approach

We perform unlexicalized grammar induction in which a sentence is represented as a sequence of part-of-speech (POS) tags. We assume that all the languages share the same set of POS tags.

### 3.1 Multilingual Grammar Model

Our multilingual grammar model adopts the NDMV (Jiang et al., 2016), a monolingual model, as the basic component. NDMV predicts grammar rule probabilities using neural networks. In our model, we add a continuous vector representation of the language identity $l$ (i.e., a language embedding) as an additional input to the neural networks in NDMV. Specifically, to predict an ATTACH rule probability $\mathrm{P}_{\text{ATTACH}}(c|h, dir, val, l)$, we use a multilayer neural network that takes the embeddings of the head token $h$, valence $val$ and language identity $l$ as input, uses a weight matrix $W_{dir}$ specific to the direction $dir$ in the first layer, and uses a weight matrix $W_c$ consisting of all the child POS tag vectors in the softmax output layer. The neural network structure is shown in the left part of Figure 1. We predict the DECISION rule probabilities in a similar way. We record the number of ROOT rule probabilities instead of predicting them

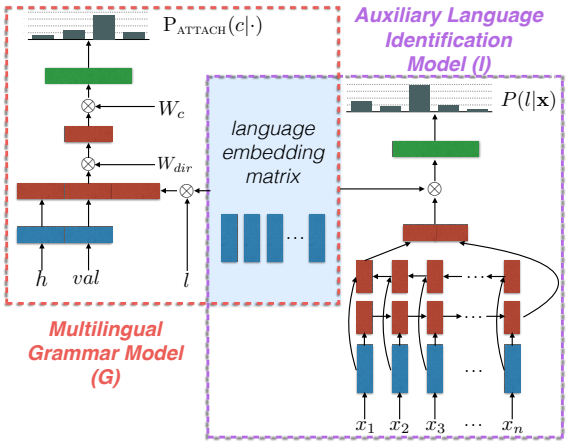


Figure 1: Model Architecture. The language embedding matrix contains the embeddings of all the languages. $l$ is a one-hot vector and $\otimes$ means matrix multiplication. Blue bars represent the embeddings of input symbols; brown bars represent the hidden states of the neural networks; green bars represent the logits which are the inputs to the Softmax layers.

since there are only a small number of such rules. The language embeddings are part of the model parameters and are trained simultaneously with all the other parameters. We hope that after training, similar languages would have similar embeddings and therefore similar grammar rule probabilities.

### 3.2 Auxiliary Task

To improve the learning of the language embeddings, we introduce an auxiliary language identification task: given an input sentence represented by a sequence of POS tags $\{x_1, x_2, \ldots, x_n\}$, predict its language. We use a standard Bidirectional Long Short-Term Memory (Bi-LSTM) to encode the input sentence, and then a multilayer perceptron to classify the sentence into one of the languages. The weight matrix of the output layer of the multilayer perceptron contains the embeddings of all the languages. The model structure is shown in the right part of Figure 1.

### 3.3 Training

Denote the set of model parameters as $\mathbf{\Theta}$, the set of languages as $\mathbf{L}$, the set of grammars of different languages as $\mathbf{G} = \{\mathbf{G}_l, l \in \mathbf{L}\}$, and the training data as $\mathcal{D} = \{\mathbf{x}^{(i)}, l^{(i)}\}_{i=1}^{N}$ where $\mathbf{x}^{(i)}$ is the $i$-th training sentence and $l^{(i)}$ is its language identity. Our training objective function $\mathcal{L}(\mathbf{\Theta})$ combines two conditional probabilities for each training sentence $\mathbf{x}^{(i)}$: $P(\mathbf{x}^{(i)}|\mathbf{G}_{l^{(i)}})$, the probability of the training sentence $\mathbf{x}^{(i)}$ being generated from the corresponding grammar $\mathbf{G}_{l^{(i)}}$; and $P(l^{(i)}|\mathbf{x}^{(i)})$,

the probability of correct language identification of $\mathbf{x}^{(i)}$.

$$\mathcal{L}(\mathbf{\Theta}) = \sum_{(\mathbf{x},l)\in\mathcal{D}} \Big( \log P_{\mathbf{\Theta}}(\mathbf{x}|\mathbf{G}_l) + \lambda \log P_{\mathbf{\Theta}}(l|\mathbf{x}) \Big)$$

where $\lambda$ is a hyper-parameter and is set to 1 by default. We follow the approach in NDMV to optimize the first term and use Adam (Kingma and Ba, 2014) to optimize the second term.

Note that the language identification model is only used during training to improve the learning of language embeddings. During testing, we run the multilingual grammar model to predict grammar rule probabilities without the need to invoke the language identification model.

## 4 Experiment

### 4.1 Setup

We selected 15 languages across 8 language families and subfamilies to ensure diversity. To enable comparisons with previous state-of-the-art approaches (Jiang et al., 2017; Li et al., 2019), we conducted our experiments on UD Treebank 1.4. For each language, we show its language family and the training corpus size in Table 1. We trained our method on the training sentences with length $\leq$ 15 and tested our method on the testing sentences with length $\leq$ 40 after removing all punctuations. Since we are doing unsupervised learning, gold dependency trees were not used during training. We use the directed dependency accuracy (DDA, the percentage of words in the testing dataset which are assigned the correct head, same to the unlabeled attachment score normally used in supervised parsing) as the evaluation metric and report the average DDA of 5 runs for each experiment. All the parameters of neural networks including language embeddings were randomly initialized and trained with learning rate 0.001, mini-batch size 1000 and epoch 50. The dimension of the head token embedding and the child token embedding is set to 10. The shape of the weight matrix $W_{dir}$ is $20 \times 10$. The dimension of the valence embedding and language identity embedding is set to 5. For the auxiliary language identification task, we use a Bi-LSTM with hidden vector dimension of 10.[1]

[1]Our code is available at https://github.com/WinnieHAN/mndmv.git.

| Language | UD Treebank | Language Family | Corpus Size |
|---|---|---|---|
| ET | Estonian | Finnic | 11404 |
| FI | Finnish | Finnic | 9648 |
| NL | Dutch | Germanic | 8783 |
| EN | English | Germanic | 7674 |
| DE | German | Germanic | 7447 |
| NO | Norwegian | Germanic | 10017 |
| GRC | Ancient_Greek | Hellenic | 9387 |
| HI | Hindi | Indo-Irian | 4997 |
| JA | Japanese | Janponic | 7441 |
| FR | French | Romance | 4976 |
| IT | Italian | Romance | 6492 |
| LA | Latin-ITTB | Romance | 10136 |
| BG | Bulgarian | Slavonic | 6507 |
| SL | Slovenian | Slavonic | 3800 |
| EU | Basque | Vasconic | 4271 |

Table 1: Languages and treebanks used in our experiments.

| CODE | MONOLINGUAL | | MULTILINGUAL | | | |
|---|---|---|---|---|---|---|
| | **DMV** | **NDMV** | **DMV** | **NDMV** | **G** | **G+I** |
| ET | 51.8 | 52.9 | 43.1 | 45.3 | 56.0 | **56.4** |
| FI | 31.8 | 27.6 | 39.1 | 40.0 | **50.7** | 49.3 |
| NL | 42.4 | 35.6 | 46.5 | 47.8 | 50.4 | **50.6** |
| EN | 51.8 | **53.7** | 47.7 | 50.8 | 51.7 | 52.7 |
| DE | 52.8 | 50.4 | 55.5 | 57.2 | 59.6 | **61.4** |
| NO | 58.9 | 59.2 | 55.7 | 58.8 | 61.0 | **61.3** |
| GRC | 40.4 | 37.7 | 41.1 | 40.8 | **46.8** | 46.2 |
| HI | 52.6 | **53.9** | 29.2 | 31.1 | 47.4 | 46.8 |
| JA | 39.8 | 37.1 | 27.8 | 29.6 | 43.4 | **44.2** |
| FR | 58.8 | 38.1 | 59.6 | 59.4 | 58.4 | **60.1** |
| IT | 60.8 | 63.6 | **66.7** | 66.4 | 64.4 | 65.9 |
| LA | 32.6 | 36.3 | 39.8 | 42.0 | **45.1** | 45.0 |
| BG | 58.9 | 61.8 | 65.9 | 69.4 | **71.3** | **71.3** |
| SL | **70.7** | 67.5 | 62.1 | 63.3 | 68.3 | 68.6 |
| EU | 42.1 | 45.5 | 45.7 | 45.2 | **54.2** | 53.6 |
| Avg | 49.7 | 48.1 | 48.4 | 49.8 | 55.3 | **55.6** |

Table 2: DDA of monolingual and multilingual approaches. Each language is indicated by its ISO 639 code. G: our multilingual grammar model. I: our auxiliary language identification task. Ave: Average DDA over 15 languages.

### 4.2 Results

We first compare our method with two baseline methods, DMV and NDMV, which are similar to our method[2]. The baseline methods are experimented in both monolingual and multilingual settings. For the monolingual setting we trained the baseline models on each language independently. For the multilingual setting we trained them on the combined training data of all the 15 languages and tested on one of the languages. Table 2 shows the experimental results. It can be seen that our multilingual grammar model (G) performs better on average than all the baselines. The improvement be-

[2]We re-implemented the DMV and the NDMV. We set the ATTACH valence and DECISION valence to 2 and used root constraints, similar to previous work (Gimpel and Smith, 2012; Bisk and Hockenmaier, 2013; Noji et al., 2016).

comes more significant when our model is jointly trained with the auxiliary language identification task (G+I). Note that our approach performs worse than the monolingual baseline on some languages, and we speculate that it is partly caused by data imbalance. In particular, the worst-performing Hindi language has only 4997 training sentences, much smaller than the average 7532. It would be interesting to make training more balanced by assigning weights to training samples of different languages, which we leave for future work.

To measure the statistical significance of the advantage of our method, we performed the nonparametric Friedman's test to support/reject the claim (null hypothesis): there is no difference between the G+I model and the NDMV model in a multilingual setting. Based on the above sample data, the P-value $7.8911 \times 10^{-4}$ would result in rejection of the claim at the 0.05 significance level, thus showing the significance in our performance gain.

In Table 3 we compare our method with recent state-of-the-art approaches on the UD Treebank dataset: Convex-MST (Grave and Elhadad, 2015), LC-DMV (Noji et al., 2016) and D-J (Jiang et al., 2017). For the three approaches we use the results reported by Jiang et al. (2017). Our G+I model performs better than Convex-MST and LC-DMV on average, even though additional priors and delicate biases are integrated into the two methods (e.g, the universal linguistic prior for Convex-MST and the limited center-embedding for LC-DMV). Our method also slightly outperforms D-J on average, even though D-J combines Convex-MST and LC-DMV and therefore utilizes even more linguistic prior knowledge.

| CODE | Convex MST | LC-DMV | D-J | G | G+I |
|---|---|---|---|---|---|
| ET | 49.4 | 31.8 | 44.0 | 56.0 | **56.4** |
| FI | 44.7 | 26.9 | 43.5 | **50.7** | 49.3 |
| NL | 45.3 | 34.1 | 43.5 | 50.4 | **50.6** |
| EN | 54.0 | 56.0 | **60.1** | 51.7 | 52.7 |
| DE | 51.4 | 50.5 | 55.7 | 59.6 | **61.4** |
| NO | 55.3 | 45.5 | 60.8 | 61.0 | **61.3** |
| GRC | 43.4 | 33.1 | 44.9 | **46.8** | 46.2 |
| HI | 56.8 | 54.2 | **60.0** | 47.4 | 46.8 |
| JA | 44.8 | 43.8 | **45.8** | 43.4 | 44.2 |
| FR | **62.0** | 48.6 | 57.0 | 58.4 | 60.1 |
| IT | 69.1 | **71.1** | 70.3 | 64.4 | 65.9 |
| LA | 38.8 | 38.6 | 42.2 | **45.1** | 45.0 |
| BG | 61.6 | 62.4 | **73.8** | 71.3 | 71.3 |
| SL | 54.0 | 49.5 | 69.6 | 68.3 | **68.6** |
| EU | 50.0 | 45.4 | **55.7** | 54.2 | 53.6 |
| Avg | 52.0 | 46.1 | 55.1 | 55.3 | **55.6** |

Table 3: Comparison of the recent state-of-the-art approaches and G/G+I. Avg: Average DDA over 15 languages.

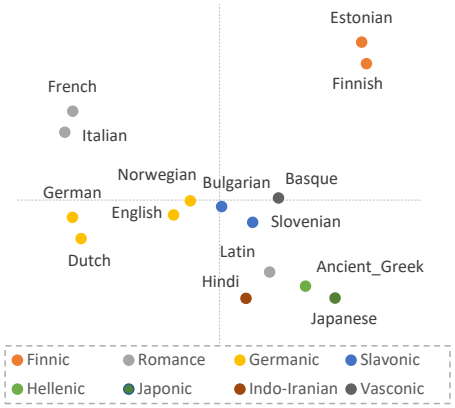


Figure 2: Visualization of the language embeddings.

## 5 Analysis

### 5.1 Visualization of Language Embeddings

One of our main expectations is that our approach can automatically learn language embeddings that capture similarities in typology between different languages. In order to verify our expectation, we collected the learned language embeddings and visualized them on a 2D plane using the t-SNE algorithm (Van der Maaten and Hinton, 2008).

Figure 2 shows the visualization result. It can be seen that in most cases languages in the same language family are close to each other. For example, Finnish is close to Estonian (Finnic languages) and Slovenian is close to Bulgarian (Slavonic languages). It is also interesting to note that some languages, such as English and Norwegian in the Germanic family and Latin in the Romance family, are closer to languages outside of their families than to their family siblings. We attribute this phenomenon to the difference in typology among some in-family languages: the flexible word order in German and Dutch is not shared by English and Norwegian; Latin, on the other hand, seems to share more common typological features with its classical counterpart, ancient Greek, than with its modern phylogenetic relatives in the Romance family. Such differences cannot be inferred from linguistic phylogeny, but our language embeddings have undoubtedly captured them.

### 5.2 In-family vs. Cross-family

In order to further examine the effectiveness of our model in coupling grammar parameters between languages regardless of their language families, we design an additional experiment on a bilingual

| | ET | FI | NO | NL | EN | DE | LA | IT | FR | BG | SL | GRC | JA | HI | EU |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ET | | -1.1 | 1.9 | 2.1 | 0.3 | 0.9 | 3.1 | 5.0 | 5.6 | 3.6 | -1.2 | 0.9 | 29.6 | 3.2 | 13.8 |
| FI | -0.2 | | 3.7 | 1.4 | 2.6 | 4.8 | 2.5 | -3.4 | 0.7 | 9.5 | 6.4 | 2.0 | 10.3 | 2.0 | 2.0 |
| NO | 8.8 | 10.0 | | -3.2 | 0.4 | 1.0 | 2.7 | -1.7 | 1.4 | 1.2 | 2.1 | 1.8 | 11.7 | 9.4 | 10.4 |
| NL | 5.3 | -1.7 | 2.5 | | 0.5 | 1.3 | 2.2 | 1.0 | 0.2 | 4.3 | 1.0 | -1.0 | 0.2 | 4.6 | 4.8 |
| EN | 0.9 | 7.6 | 2.0 | 0.7 | | 1.6 | 0.1 | 1.2 | 0.6 | 0.2 | -1.0 | 2.3 | 18.3 | 1.2 | 0.6 |
| DE | -0.8 | 5.4 | 0.7 | 2.1 | 1.9 | | 1.9 | 0.9 | 1.8 | 2.6 | -0.4 | 2.0 | 1.8 | 17.5 | 8.6 |
| LA | 7.0 | 0.8 | 0.3 | -5.2 | -3.6 | 4.0 | | -2.0 | -5.4 | -3.5 | -1.6 | -2.2 | 12.3 | 22.8 | 3.7 |
| IT | 0.7 | 11.4 | -0.3 | 0.4 | 2.8 | -2.0 | -1.3 | | -0.1 | 2.2 | 1.1 | 6.2 | 11.3 | 20.1 | 10.0 |
| FR | 0.9 | -1.8 | -1.1 | 0.6 | 1.4 | 0.2 | 0.8 | -0.4 | | 5.2 | -0.5 | 3.7 | -0.2 | 21.1 | 9.1 |
| BG | 9.0 | 13.3 | 1.2 | -2.0 | -0.9 | 2.3 | 0.1 | -1.5 | -4.1 | | 1.3 | 0.9 | 18.2 | 25.8 | 12.7 |
| SL | 5.2 | 11.7 | 1.3 | -4.3 | 1.0 | -4.1 | -0.2 | -0.6 | 0.4 | 2.3 | | 4.5 | 2.6 | 21.3 | 15.2 |
| GRC | 4.1 | -0.6 | 0.6 | -4.8 | 4.1 | 1.8 | 2.4 | 3.5 | -0.3 | 2.2 | 4.6 | | 10.4 | 12.4 | 11.4 |
| JA | 11.6 | 0.0 | -7.1 | -4.4 | 5.4 | -3.3 | 5.9 | -5.6 | 3.9 | 0.2 | 8.0 | 7.6 | | 0.0 | 2.1 |
| HI | 5.9 | -1.2 | 5.6 | 1.7 | 1.4 | 5.9 | 3.1 | 9.0 | 2.1 | 7.6 | 4.6 | 2.0 | 0.3 | | -8.4 |
| EU | 16.0 | -1.5 | 3.9 | 2.7 | -0.6 | 1.7 | 2.0 | 1.1 | 8.0 | -0.6 | 4.0 | 6.0 | -0.2 | -2.1 | |

Figure 3: Each cell shows the difference between the DDA of our model and that of DMV evaluated on the test data of the column language. Positive numbers are shown in red and negative numbers in blue. Languages are grouped by their families.

setup. Specifically, for each pair of languages, we tested our approach and the baseline of training DMV on the combined training set. Since in this setting DMV is blind to the language identity, we expect that it would perform poorly if the two languages come from different families and hence are very likely to have large difference. On the other hand, our model would not be as sensitive to the difference between the two languages. In Figure 3, we report the difference between the DDA of our model and that of DMV. It shows that the advantage of our model over DMV is more significant for cross-family language pairs than for in-family language pairs, which verifies our expectation.

## 6 Conclusion

In this paper, we incorporate continuous language identity representations into multilingual grammar induction, which softly tie grammar parameters from different languages, resulting in substantial performance gain over various baseline methods. Analysis of the language embeddings suggests that our approach may capture information about language similarity beyond linguistic phylogenetic knowledge.

While in this work we follow previous work and perform unlexicalized parsing, the proposed model can be extended for lexicalized parsing by replacing POS tag embeddings with cross-lingual word embeddings, which we leave for future work.

## Acknowledgments

This work was supported by the Major Program of Science and Technology Commission Shanghai Municipal (17JC1404102).

## References

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A Smith. 2016. Many languages, one parser. *Transactions of the Association of Computational Linguistics*.

Taylor Berg-Kirkpatrick and Dan Klein. 2010. Phylogenetic grammar induction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1288–1297. Association for Computational Linguistics.

Yonatan Bisk and Julia Hockenmaier. 2013. An hdp model for inducing combinatory categorial grammars. *Transactions of the Association for Computational Linguistics*, 1:75–88.

Shay B Cohen and Noah A Smith. 2009. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 74–82. Association for Computational Linguistics.

Kevin Gimpel and Noah A Smith. 2012. Concavity and initialization for unsupervised dependency parsing. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 577–581. Association for Computational Linguistics.

Edouard Grave and Noémie Elhadad. 2015. A convex and feature-rich discriminative approach to dependency grammar induction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1375–1384.

Joseph H Greenberg. 1963. Some universals of grammar with particular reference to the order of meaningful elements. *Universals of language*, 2:73–113.

John A Hawkins. 2014. *Cross-linguistic variation and efficiency*. OUP Oxford.

Tomoharu Iwata, Daichi Mochihashi, and Hiroshi Sawada. 2010. Learning common grammar from multilingual corpus. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 184–188. Association for Computational Linguistics.

Yong Jiang, Wenjuan Han, and Kewei Tu. 2016. Unsupervised neural dependency parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 763–771, Austin, Texas. Association for Computational Linguistics.

Yong Jiang, Wenjuan Han, and Kewei Tu. 2017. Combining generative and discriminative approaches to unsupervised dependency parsing via dual decomposition. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1689–1694, Copenhagen, Denmark. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Dan Klein and Christopher D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Miryam de Lhoneux, Johannes Bjerva, Isabelle Augenstein, and Anders Søgaard. 2018. Parameter sharing between dependency parsers for related languages. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4992–4997. Association for Computational Linguistics.

Bowen Li, Jianpeng Cheng, Yang Liu, and Frank Keller. 2019. Dependency grammar induction with a neural variational transition-based parser. In *AAAI 2019*.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85.

Hiroshi Noji, Yusuke Miyao, and Mark Johnson. 2016. Using left-corner parsing to encode universal structural constraints in grammar induction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 33–43.

# Quantifying the Semantic Core of Gender Systems

**Adina Williams**[ə] **Ryan Cotterell**[ʃ,ɦ] **Lawrence Wolf-Sonkin**[ʃ]
**Damián Blasi**[ʔ] **Hanna Wallach**[3]
[ə] Facebook AI Research, New York, USA
[ʃ]Department of Computer Science, Johns Hopkins University, Baltimore, USA
[ɦ]Department of Computer Science and Technology, University of Cambridge, Cambridge, UK
[ʔ]Comparative Linguistics Department, University of Zürich, Switzerland
[3]Microsoft Research, New York City, USA
adinawilliams@fb.com, rdc42@cam.ac.uk, lawrencews@jhu.edu
damian.blasi@uzh.ch, hanna@dirichlet.net

## Abstract

Many of the world's languages employ grammatical gender on the lexeme. For example, in Spanish, the word for house (*casa*) is feminine, whereas the word for paper (*papel*) is masculine. To a speaker of a genderless language, this assignment seems to exist with neither rhyme nor reason. But is the assignment of inanimate nouns to grammatical genders truly arbitrary? We present the first large-scale investigation of the arbitrariness of noun–gender assignments. To that end, we use canonical correlation analysis to correlate the grammatical gender of inanimate nouns with an externally grounded definition of their lexical semantics. We find that 18 languages exhibit a significant correlation between grammatical gender and lexical semantics.

## 1 Introduction

In his semi-autobiographic work about his time traveling through Germany, *A Tramp Abroad*, Twain (1880) recounted his difficulty when learning the German gender system: "Every noun has a gender, and there is no sense or system in the distribution; so the gender of each must be learned separately and by heart. In German, a young lady has no sex, while a turnip has. Think what overwrought reverence that shows for the turnip, and what callous disrespect for the girl." Although this humorous take on German grammatical gender is clearly a caricature, the quote highlights the fact that the relationship between the grammatical gender of nouns and their lexical semantics is often quite opaque.

As arbitrary as certain noun–gender assignments may appear overall, a relatively clear relationship often exists between grammatical gender and lexical semantics for some of the lexicon. The portion of the lexicon where this relationship is clear usually consists of animate nouns; nouns referring to people morphologically reflect the sociocultural notion of "natural genders." This portion of the lexicon—the "semantic core"—seems to be present in *all* gendered languages (Aksenov, 1984; Corbett, 1991). But how many *inanimate* nouns can also be included in the semantic core? Answering this question requires investigating whether there is a correlation between grammatical gender and lexical semantics for inanimate nouns.

Our primary technical contribution is demonstrating that grammatical gender and lexical semantics can be correlated using canonical correlation analysis (CCA)—a standard method for computing the correlation between two multivariate random variables. We consider 18 gendered languages, following 3 steps for each: First, we encode each inanimate noun as a one-hot vector representing the noun's grammatical gender in that language; we then create 5 operationalizations of each noun's lexical semantics using word embeddings in 5 genderless "donor" languages (English, Japanese, Korean, Mandarin Chinese, and Turkish); finally, for each genderless language, we use CCA to compute the desired correlation between grammatical gender and lexical semantics. This process yields a single value for each of the 90 gendered–genderless language pairs, revealing a significant correlation between grammatical gender and lexical semantics for 55 of these language pairs.

Secondarily, we investigate semantic similarities between the 18 languages' gender systems—i.e., their assignments of nouns to grammatical genders. We analyze the projections of lexical semantics (operationalized as word embeddings in English) obtained via CCA, finding that phylogenetically



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5734–5739, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

similar languages have more similar projections.

## 2 Background and Assumptions

### 2.1 Grammatical Gender

Languages range from employing no grammatical gender on inanimate nouns, like English, Japanese, Korean, Mandarin Chinese, and Turkish, to drawing grammatical distinctions between tens of gender-like classes (Corbett, 1991). Although there are many theories about the assignment of inanimate nouns to grammatical genders, to the best of our knowledge, the linguistics literature lacks any large-scale, quantitative investigation of arbitrariness of noun–gender assignments. However, with the advent of modern NLP methods—particularly with advancements in distributional approaches to semantics (Harris, 1954; Firth, 1957)—and with the copious amounts of text available on the internet, it is now possible to conduct such an investigation. We focus on languages that have either two (masculine–feminine) or three (masculine–feminine–neuter) genders, to which nouns are exhaustively assigned, and investigate whether a correlation exists between grammatical gender and lexical semantics for inanimate nouns—i.e., whether noun–gender assignments are arbitrary or not.

In many languages, a noun's grammatical gender can be predicted from its spelling and pronunciation (Cucerzan and Yarowsky, 2003; Nastase and Popescu, 2009). For example, almost all Spanish nouns ending in *-a* are feminine, whereas Spanish nouns ending in *-o* are usually masculine. These assignments are non-arbitrary; indeed, Corbett (1991, Ch. 4) provides a thorough typological description of how phonology pervades gender systems. We emphasize that these assignments are not the subject of our investigation. Rather, we are concerned with the relationship between grammatical gender and lexical semantics—i.e., when asking why the Spanish word *casa* is feminine, we do not consider that it ends in *-a*.

Finally, our investigation is related to that of Kann and Wolf-Sonkin, which assumes that noun–gender assignments are non-arbitrary and examines the predictability of grammatical gender from lemmatized word embeddings; in contrast, we investigate the arbitrariness of noun–gender assignments.

### 2.2 Lexical Semantics via Word Embeddings

The NLP community has widely adopted word embeddings as way of representing lexical semantics. The underlying motivation behind this adoption is the observation that words with similar meanings will be embedded as vectors that are closer together. As we explain in §3, our investigation requires a definition of lexical semantics that is independent of grammatical gender. However, in many gendered languages, word embeddings effectively encode grammatical gender because this information is trivially recoverable from distributional semantics. For example, in Spanish, singular masculine nouns tend to occur after the article *el*, whereas singular feminine nouns tend to occur after the article *la*. For this reason, we use an externally grounded definition of lexical semantics: we create 5 operationalizations of each noun's lexical semantics using word embeddings in 5 genderless "donor" languages (English, Japanese, Korean, Mandarin Chinese, and Turkish). We use 5 languages that are phylogenetically distinct and spoken in distinct regions to minimize any spurious correlations.[1]

Our investigation is based on the linguistic assumption that word embeddings in a genderless "donor" language are a good proxy for genderless lexical semantics. In practice, however, this assumption is generally false: word embeddings are largely a reflection of the text with which they were trained. For example, the embedding of the word *snow* will differ depending on whether the training text was written by people near the equator or people near the North Pole, even if both groups speak the same language. Such differences will be more pronounced for rare words, which are arguably more language- and culture-specific than many common words. For this reason, we limit the scope of our investigation to only those inanimate nouns that are likely to be used consistently across different languages. To implement this limitation, we use a Swadesh list (Buck, 1949; Swadesh, 1950, 1952, 1955, 1971/2006)—a list of words constructed to contain only very frequent words that are as close to culturally neutral as possible. By limiting the scope of our investigation to only those inanimate nouns that appear in a Swadesh list, we can be reasonably confident that their word embeddings in English, Japanese, Korean,

[1] It is natural to ask whether polysemy and homonymy might result in spurious similarities. For example, in English, the words *fish*$_N$ and *fish*$_V$ are homonymous, but in Mandarin Chinese, the words *yú* (*fish*$_N$) and *diào* (*fish*$_V$) are not. If patterns of homonymy in the genderless languages are very different, but patterns of correlation between grammatical gender and lexical semantics are very similar, then we can be reasonably sure that the correlations are not due to homonymy.

| | bg | ca | el | es | fr | he | hi | hr | it | lt | lv | pl | pt | ro | ru | sk | sl | uk |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| en | **2596** | **2720** | **2872** | **4947** | **6257** | **1489** | 828 | 443 | **4800** | **923** | **881** | **1646** | **3918** | 397 | **5779** | 1056 | 293 | 975 |
| ja | **2586** | **2596** | **2886** | **3383** | **4654** | **2223** | **1421** | 486 | **3849** | 1241 | **1215** | **1884** | **3615** | **497** | **4532** | 375 | 419 | 1380 |
| ko | **1856** | **1843** | 1982 | 1840 | **2812** | **1774** | 1269 | 371 | **2513** | 1089 | **997** | 1357 | **2442** | **364** | **2680** | 247 | 317 | 1169 |
| tr | **1578** | **1623** | **1735** | **1766** | **2580** | **1275** | 817 | 274 | **2287** | 903 | 826 | 1163 | **2223** | 303 | **2470** | 218 | 281 | 909 |
| zh | **2275** | **2190** | 2454 | **2693** | **3722** | 1810 | **1266** | 373 | **3170** | **1111** | **1110** | **1643** | **3084** | 480 | **3652** | 286 | **341** | 1196 |

Table 1: The number of inanimate nouns for each gendered–genderless language pair. Bold indicates that our investigation reveals a significant correlation between grammatical gender and lexical semantics for that pair.

Mandarin Chinese, and Turkish are a good proxy for their genderless lexical semantics, as desired.

## 3 Methodology

### 3.1 Data

We use Open Multilingual WordNet (Stamou et al., 2004; Ordan and Wintner, 2007; Raffaelli et al., 2008; Sagot and Fišer, 2008; Tufiş et al., 2008; Piasecki et al., 2009; Simov and Osenova, 2010; Toral et al., 2010; Bond and Paik, 2012; Fišer et al., 2012; González-Agirre et al., 2012; de Paiva and Rademaker, 2012; Rudnicka et al., 2012; Bond and Foster, 2013; Garabík and Pileckytė, 2013; Oliver et al., 2015)[2] as our Swadesh list. This yields 18 gendered languages (Bulgarian, Catalan, Greek, Spanish, French, Hebrew, Hindi, Croatian, Italian, Lithuanian, Latvian, Polish, Portuguese, Romanian, Russian, Slovak, Slovenian, and Ukranian), of which 12 are from the Indo-European family and 4 are not.[3] For the 5 genderless languages (English, Japanese, Korean, Mandarin Chinese, and Turkish), we use pre-trained, 50-dimensional word embeddings from FASTTEXT (Bojanowski et al., 2017; Grave et al., 2018).[4] For each gendered–genderless language pair, we limit the scope of our investigation to only those inanimate nouns that occur in both our Swadesh list and in FASTTEXT; we provide the resulting counts in Table 1. Finally, we randomly partition the set of nouns for each language pair into a 75%–25% training–testing split.

### 3.2 Notation

We first establish the requisite notation. Let $\mathcal{V}_{\ell,m} = \{1, \ldots, V_{\ell,m}\}$ denote a set of integers representing the inanimate nouns for gendered language $\ell$ and genderless language $m$. Let $\mathcal{G}_\ell$ denote the (arbitrarily ordered) genders in language $\ell$; for example, let $\mathcal{G}_{spanish} = (\text{MSC}, \text{FEM})$. Given an inanimate noun $n \in \mathcal{V}_{\ell,m}$, let $\boldsymbol{g}_\ell(n)$ denote a one-hot vector representing $n$'s grammatical gender in language $\ell$, so that the $i^{\text{th}}$ entry corresponds to the $i^{\text{th}}$ gender in $\mathcal{G}_\ell$. Similarly, let $\mathbf{e}_m(n) \in \mathbb{R}^{50}$ denote the 50-dimensional word embedding representing the lexical semantics of $n$ in language $m$. Let $G_\ell \in \mathbb{R}^{|\mathcal{G}_\ell| \times V_{\ell,m}}$ collectively denote the inanimate nouns' grammatical genders in language $\ell$, so that the $n^{\text{th}}$ column is $\boldsymbol{g}_\ell(n)$, and let $E_m \in \mathbb{R}^{50 \times V_{\ell,m}}$ collectively denote the inanimate nouns' lexical semantics, so that the $n^{\text{th}}$ column is $\mathbf{e}_m(n)$. Finally, let $G_\ell^{\text{train}}$ and $E_m^{\text{train}}$ respectively denote the columns of $G_\ell$ and $E_m$ that correspond to the inanimate nouns in the training set and let $G_\ell^{\text{test}}$ and $E^{\text{test}}$ respectively denote the columns of $G_\ell$ and $E_m$ that correspond to the inanimate nouns in the testing set.

### 3.3 Canonical Correlation Analysis

CCA is a standard method for computing the correlation between two multivariate random variables. In our investigation, we are interested in the correlation between grammatical gender and lexical semantics for each gendered–genderless language pair. To compute this correlation, we start by solving the following optimization problem:

$$(\mathbf{a}^\star, \mathbf{b}^\star) = \underset{(\mathbf{a},\mathbf{b})}{\arg\max}\ corr(\mathbf{a}^\top G_\ell^{\text{train}}, \mathbf{b}^\top E_m^{\text{train}}).$$

Although this optimization problem is non-convex, it can be solved in closed form using singular value decomposition (SVD). We use a standard implementation of CCA (Pedregosa et al., 2011).

Having found the projections $\mathbf{a}^\star \in \mathbb{R}^{|\mathcal{G}_\ell|}$ and $\mathbf{b}^\star \in \mathbb{R}^{50}$ that maximize the correlation, we then use them to compute the correlation between grammatical gender and lexical semantics as follows:

$$\rho_{\ell,m} = corr(\mathbf{a}^{\star\top} G_\ell^{\text{test}}, \mathbf{b}^{\star\top} E_m^{\text{test}}).$$

To establish statistical significance, we follow the approach of Monteiro et al. (2016). We create $B = 100,000$ permutations of the columns of

[2] http://compling.hss.ntu.edu.sg/omw/summx.html

[3] This imbalance as a limitation of our investigation.

[4] The FASTTEXT word embeddings were trained using Common Crawl and Wikipedia data, using CBOW with position weights, with character $n$-grams of length 5. For more information, see http://fasttext.cc/docs/en/crawl-vectors.html.

$G_\ell^{\text{train}}$; for each permutation $b$, we then repeat the steps above to obtain $\rho_{\ell,m}$; finally, we compute

$$p = \frac{1 + \sum_{b=1}^{B} \delta(\rho_{\ell,m}^{b} \geq \rho_{\ell,m})}{B+1}.$$

Because our investigation involves testing 90 different hypotheses, we use Bonferroni correction (Dror et al., 2017)—i.e., we multiply $p$ by 90. If the resulting Bonferroni-corrected $p$-value is small, then we can reject the null hypothesis that there is no correlation between grammatical gender and lexical semantics for that language pair.

Secondarily, we investigate semantic similarities between the 18 languages' gender systems by analyzing their projections of lexical semantics. For each pair of *gendered* languages $\ell$ and $\ell'$, we compute the correlation (cosine distance) between $\mathbf{b}_\ell^\star$ and $\mathbf{b}_{\ell'}^\star$ for each of the 5 genderless languages.

## 4 Results

We find a significant correlation between grammatical gender and lexical semantics (i.e., the Bonferroni-corrected $p$-value is less than 0.05) for 55 of the 90 gendered–genderless language pairs. These results are depicted in Figure 1. For Slovak, Croatian, and Ukranian, we find no correlation for any of the genderless languages; for Slovenian, we find a significant correlation for only Mandarin Chinese. We suspect that these results are due the relatively small number of inanimate nouns considered for each of these language pairs (see Table 1 for the counts). We also find slightly different patterns of correlation for the different genderless languages

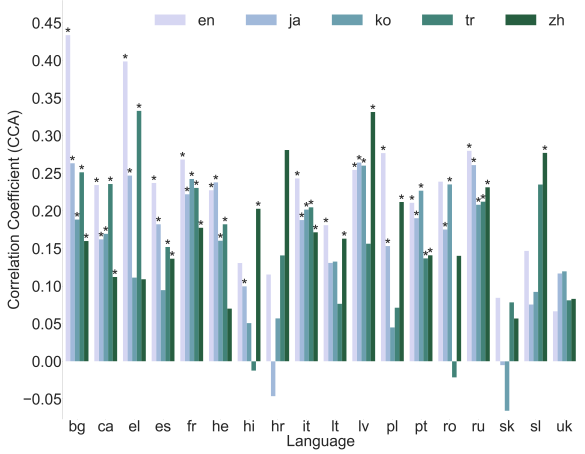


Figure 1: The correlation between grammatical gender and lexical semantics for each of the 90 gendered–genderless language pairs ($*$ indicates significance).

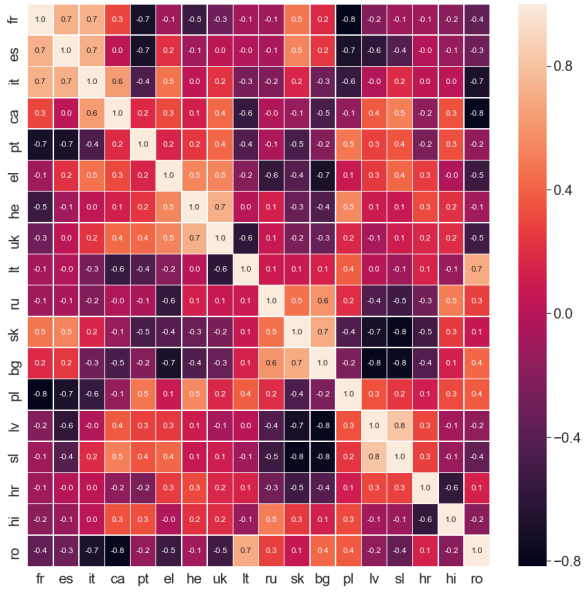


Figure 2: The correlation between $\mathbf{b}_\ell^\star$ and $\mathbf{b}_{\ell'}^\star$ for each pair of gendered languages $\ell$ and $\ell'$ (for English).

that we use to create our 5 operationalizations of lexical semantics. For Japanese, we find significant correlations for 13 of the 18 gendered languages; for English and Chinese, we find significant correlations for 12; for Korean and Turkish, we find significant correlations for 9 of the gendered languages.

For each pair of gendered languages $\ell$ and $\ell'$, Figure 2 depicts the the correlation (cosine distance) between $\mathbf{b}_\ell^\star$ and $\mathbf{b}_{\ell'}^\star$ for English. We find higher correlations for pairs of languages that are phylogenetically similar. For example, French has higher correlations with Spanish and Italian than with Polish. This is likely because phylogenetically similar languages exhibit historical similarities in their gender systems as a result of a common linguistic origin (Fodor, 1959; Ibrahim, 2014; Stump, 2015).

## 5 Conclusion

Our investigation is the first to quantitatively demonstrate that there is a significant correlation between grammatical gender and lexical semantics for inanimate nouns. Although our results provide evidence for the non-arbitrariness of noun–gender assignments, they must be contextualized. In contrast to animate nouns, it is not clear that a single cross-linguistic category explains our results. Moreover, we limit the scope of our investigation to frequent inanimate nouns. These nouns tend to be distributed across genders, whereas less frequent inanimate nouns tend to be assigned to a single gender (Dye et al., 2015). We leave the investigation of less frequent inanimate nouns for future work.

## Acknowledgements

Thanks to Jennifer Culbertson, Jacob Eisenstein, and Arya McCarthy for conversations on this topic.

## References

A.T. Aksenov. 1984. K probleme èkstralingvističeskoj motivacii grammatičeskoj kategorii roda [on extralinguistic motivation of the grammatical category of gender]. *Voprosy Jazykoznanija 33 (1)*, pages 14–25.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Francis Bond and Ryan Foster. 2013. Linking and extending an open multilingual wordnet. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1352–1362.

Francis Bond and Kyonghee Paik. 2012. A survey of wordnets and their licenses. *Proceedings of the 6th Global WordNet Conference (GWC)*, 8(4):5.

Carl Buck. 1949. *A Dictionary of Selected in the Principal Indo-European Languages*. University of Chicago Press.

Greville G. Corbett. 1991. *Gender*. Cambridge Textbooks in Linguistics. Cambridge University Press.

Silviu Cucerzan and David Yarowsky. 2003. Minimally supervised induction of grammatical gender. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*.

Rotem Dror, Gili Baumer, Marina Bogomolov, and Roi Reichart. 2017. Replicability analysis for natural language processing: Testing significance with multiple datasets. *Transactions of the Association for Computational Linguistics*, 5:471–486.

Melody Dye, Petar Milin, Richard Futrell, Michael Ramscar, and Eberhard Karls. 2015. A functional theory of gender paradigms. In *Perspectives on Morphological Organization*.

John R. Firth. 1957. A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*.

Darja Fišer, Jernej Novak, and Tomaž Erjavec. 2012. sloWNet 3.0: development, extension and cleaning. In *Proceedings of 6th International Global Wordnet Conference (GWC 2012)*, pages 113–117. The Global WordNet Association.

Istvan Fodor. 1959. The origin of grammatical gender. *Lingua*, 8:186–214.

Radovan Garabík and Indrė Pileckytė. 2013. From multilingual dictionary to Lithuanian wordnet. In *Natural Language Processing, Corpus Linguistics, E-Learning*, pages 74–80.

Aitor González-Agirre, Egoitz Laparra, and German Rigau. 2012. Multilingual central repository version 3.0: upgrading a very large lexical knowledge base. In *Proceedings of the 6th Global WordNet Conference (GWC 2012)*, Matsue.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Zellig S. Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Muhammad Hasan Ibrahim. 2014. *Grammatical gender: Its origin and development*, volume 166. Walter de Gruyter.

Katharina Kann and Lawrence Wolf-Sonkin. To appear. Grammatical gender, Neo-Whorfianism, and word embeddings: A data-driven approach to linguistic relativity (manuscript from 2018).

João M. Monteiro, Anil Rao, John Shawe-Taylor, Janaina Mourão-Miranda, and Alzheimer's Disease Initiative. 2016. A multiple hold-out framework for sparse partial least squares. *Journal of neuroscience methods*, 271:182–194.

Vivi Nastase and Marius Popescu. 2009. What's in a name? In some languages, grammatical gender. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1368–1377. Association for Computational Linguistics.

Antoni Oliver, Krešimir Šojat, and Matea Srebačić. 2015. Automatic expansion of Croatian wordnet. In *In Proceedings of the 29th CALS international conference Applied Linguistic Research and Methodology*, Zadar (Croatia).

Noam Ordan and Shuly Wintner. 2007. Hebrew wordnet: a test case of aligning lexical databases across languages. *International Journal of Translation*, 19(1):39–58.

Valeria de Paiva and Alexandre Rademaker. 2012. Revisiting a Brazilian wordnet. In *Proceedings of the 6th Global WordNet Conference (GWC 2012)*, Matsue.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12(Oct):2825–2830.

Maciej Piasecki, Stan Szpakowicz, and Bartosz Broda. 2009. *A Wordnet from the Ground Up*. Wroclaw University of Technology Press.

Ida Raffaelli, Božo Bekavac, Željko Agić, and Marko Tadić. 2008. Building Croatian wordnet. In *Proceedings of the Fourth Global WordNet Conference 2008*, pages 349–359.

Ewa Rudnicka, Marek Maziarz, Maciej Piasecki, and Stanisaw Szpakowicz. 2012. Mapping plWordNet onto Princeton WordNet. *International Journal of Lexicography*.

Benoît Sagot and Darja Fišer. 2008. Building a free French wordnet from multilingual resources. In *Proceedings of the Sixth International Language Resources and Evaluation (LREC 2008)*, Marrakech, Morocco.

Kiril Simov and Petya Osenova. 2010. Constructing of an ontology-based lexicon for Bulgarian. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC 2010)*, Valletta, Malta. European Language Resources Association (ELRA).

Sofia Stamou, Goran Nenadic, and Dimitris Christodoulakis. 2004. Exploring Balkanet shared ontology for multilingual conceptual indexing. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004)*.

Gregory Stump. 2015. Inflection classes. In *The Oxford handbook of inflection*. Oxford University Press.

Morris Swadesh. 1950. Salish internal relationships. *International Journal of American Linguistics*, 16(4):157–167.

Morris Swadesh. 1952. Lexico-statistic dating of prehistoric ethnic contacts: with special reference to north American Indians and Eskimos. *Proceedings of the American Philosophical Society*, 96(4):452–463.

Morris Swadesh. 1955. Towards greater accuracy in lexicostatistic dating. *International Journal of American linguistics*, 21(2):121–137.

Morris Swadesh. 1971/2006. *The origin and diversification of language*. Chicago: Aldine.

Antonio Toral, Stefania Bracale, Monica Monachini, and Claudia Soria. 2010. Rejuvenating the Italian wordnet: upgrading, standardising, extending. In *Proceedings of the 5th International Conference of the Global WordNet Association (GWC-2010)*, Mumbai.

Dan Tufiş, Radu Ion, Luigi Bozianu, Alexandru Ceauşu, and Dan Ştefănescu. 2008. Romanian wordnet: Current state, new applications and prospects. In *Proceedings of the 4th Global WordNet Association Conference*, pages 441–452, Szeged.

Mark Twain. 1880. *A Tramp Abroad*. Wikisource.

# Perturbation Sensitivity Analysis to Detect Unintended Model Biases

**Vinodkumar Prabhakaran**
Google Brain
San Francisco, CA, USA
vinodkpg@google.com

**Ben Hutchinson**
Google Brain
San Francisco, CA, USA
benhutch@google.com

**Margaret Mitchell**
Google Brain
Seattle, WA, USA
mmitchellai@google.com

## Abstract

Data-driven statistical Natural Language Processing (NLP) techniques leverage large amounts of language data to build models that can understand language. However, most language data reflect the public discourse at the time the data was produced, and hence NLP models are susceptible to learning incidental associations around named referents at a particular point in time, in addition to general linguistic meaning. An NLP system designed to model notions such as sentiment and toxicity should ideally produce scores that are independent of the identity of such entities mentioned in text and their social associations. For example, in a general purpose sentiment analysis system, a phrase such as *I hate Katy Perry* should be interpreted as having the same sentiment as *I hate Taylor Swift*. Based on this idea, we propose a generic evaluation framework, *Perturbation Sensitivity Analysis*, which detects unintended model biases related to named entities, and requires no new annotations or corpora. We demonstrate the utility of this analysis by employing it on two different NLP models — a sentiment model and a toxicity model — applied on online comments in English language from four different genres.

## 1 Introduction

Recent research has shown ample evidence that data-driven NLP models may inadvertently capture, reflect and sometimes amplify various social biases present in the language data they are trained on (Bolukbasi et al., 2016; Blodgett and O'Connor, 2017). Such biases can often result in unintended and disparate harms to the users who engage with NLP-aided systems. For instance, when NLP algorithms are used to moderate online communication, e.g. by detecting harassment, although the net social benefits may be positive, the harms caused by incorrect classifications may be unevenly distributed, leading to disparate impact (Feldman et al., 2015). Some writers may find their contributions being disproportionately censored, while some readers may not be adequately protected from harassment (Dixon et al., 2018).

| Sentence | Toxicity | Sentiment |
|---|---|---|
| I hate Justin Timberlake. | 0.90 | -0.30 |
| I hate Katy Perry. | 0.80 | -0.10 |
| I hate Taylor Swift. | 0.74 | -0.40 |
| I hate Rihanna. | 0.69 | -0.60 |

Table 1: Sensitivity of NLP models to named entities in text. Toxicity score range: 0 to 1; Sentiment score range: -1 to +1.

Research into fairness in machine learning distinguishes two broad categories of unfair discrimination. First, unfairness for *individuals* exists when similar individuals are treated dissimilarly (Dwork et al., 2012). Second, a range of criteria define unfairness for *groups*, each in terms of statistical dependence between group membership, model score, and class label (see, e.g., (Chouldechova and Roth, 2018; Mitchell et al., 2018)). In both cases, what is "fair" or "unfair" is highly context-dependent, and judgments about fairness require consideration of the broader sociotechnical frame (Selbst et al., 2019).

This framework also poses some practical challenges: individual fairness requires knowing intricate details about an individual, while group fairness requires knowing how an individual can be categorized into legally and socially sensitive roles. The first runs into the ethical concerns of surveillance; the second runs into the ethical concerns of discrimination. Furthermore, texts are often not annotated with the social groups of their readers/writers (and for privacy reasons we may not wish to infer them), or whether two individuals are "similar" or not. Hence, fairness research in NLP has mostly focused on mentions of social identities (Dixon et al., 2018; Borkan et al., 2019; Garg et al., 2019), or on how social stereotypes impact semantic interpretation (Webster et al., 2018), and often rely heavily on annotated corpora.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5740–5745,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

In this paper, we propose a general-purpose evaluation framework that detects unintended biases in NLP models around named entities mentioned in text. Our method does not rely on any annotated corpora, and we focus solely on application-independent *sensitivity* of models, which does not clearly fall under individual- or group- based fairness criteria. Our core idea is based on the assumption that an NLP system designed to be widely applicable should ideally produce scores that are independent of the identities of named entities mentioned in the text. For instance, the sentences *I hate Justin Timberlake* and *I hate Rihanna* both express the same semantics using identical constructions; however, the toxicity model used in our experiments gives a significantly higher score to the former (0.90) than the latter (0.69) (see Table 1 for more examples).

Mentions of such real-world entities are pervasive in data. Just as word co-occurrence metrics capture 'meaning representations' of words in the language,[1] co-occurrence patterns between entity mentions and other parts of the phrases they occur in influence their learned meaning. For example, if a person's name is often mentioned in negative linguistic contexts, a trained model might inadvertently associate negativity to that name, resulting in biased predictions on sentences with that name. If unchecked, this leads to undesirable biases in the model, violating tenets of both individual and group fairness as they are applied in context.

The primary contributions of this paper are: (i) a simple and effective general-purpose model evaluation metric, which we call *perturbation sensitivity analysis*, for measuring unintended bias; (ii) a large-scale systematic analysis of model sensitivity to name perturbations, on two tasks – sentiment and toxicity – across four different genres of English text; (iii) a demonstration of how name perturbation can reveal undesired biases in the learned model towards names of popular personalities; (iv) showing the downstream impact of name sensitivity, controlling for prediction thresholds.

## 2 Perturbation Sensitivity Analysis

We introduce *Perturbation Sensitivity Analysis (PSA)*, a general evaluation technique to detect unintended biases in NLP models towards real-world entities. Central to our approach is the notion of *perturbation*, where a reference to a real-world entity is replaced by a reference to another real-world entity of the same type (e.g., a person name replaced with another person name). PSA measures the extend to which a model prediction is sensitive to such perturbations, and is calculated w.r.t. a set of (unannotated) sentences $X$ from the target domain and a set of names $N$ (of the same entity type $t$) that the perturbations are performed over.

For simplicity, in this paper, we discuss text classification models that take in a piece of text and return a score for a target class. Similarly, we focus on perturbing over person names. However, our method is readily extendable to other kinds of models as well as to other entity types.

Our approach begins by first retrieving the set of sentences $X$ such that each sentence contains at least one referring expression that refers to an entity of the type we are doing perturbation on (person, in our case). This referring expression could be a pronoun or a proper name. We select one such referring expression as the *anchor* for each sentence in $X$. We then "perturb" each sentence by replacing the anchor with named entities $n \in N$. We then measure the sensitivity of the model with respect to such perturbation by running it on the resulting set of $|X| * |N|$ perturbed sentences.

Formally, let $x_n$ denote the perturbed sentence obtained by replacing the anchor word in $x \in X$ with $n$, and $f(x_n)$ denote the score assigned to a target class by model $f$ on the perturbed sentence $x_n$. Formally, we define three metrics for the *perturbation sensitivity of model scores*:

**Perturbation Score Sensitivity** (*ScoreSens*) of a model $f$ with respect to a corpus $X$ and a name $n$ is the average difference between $f(x_n)$ and $f(x)$ calculated over $X$, i.e. $\underset{x \in X}{E}[f(x_n) - f(x)]$.

**Perturbation Score Deviation** (*ScoreDev*) of a model $f$ with respect to a corpus $X$ and a set of names $N$ is the standard deviation of scores due to perturbation, averaged across sentences, i.e., $\underset{x \in X}{E}[\underset{n \in N}{StdDev}(f(x_n)]$.

**Perturbation Score Range** (*ScoreRange*) of a model $f$ with respect to a corpus $X$ and a set of names $N$ is the $Range$ ($max - min$) of scores, averaged across sentences, i.e., $\underset{x \in X}{E}[\underset{n \in N}{Range}(f(x_n)]$.

Whether a score difference caused by name perturbation results in a different label depends also on the threshold. Given a threshold, $0 \leq c \leq 1$,

[1] Often through word embeddings fed to or learned by the first layer of neural network based models

binary labels $y(x)$ can be obtained from the classifier $f$ as $\mathbf{I}[f(x) \geq c] \in \{0, 1\}$, where $\mathbf{I}[\cdot]$ is the indicator function. Using this, we define a metric for the *perturbation sensitivity of model labels*:

**Perturbation Label Distance** (*LabelDist*) of a binary classifier $y$ with respect to a corpus $X$ and a set of names $N$ is the Jaccard Distance between a) the set of sentences $\{x\}$ for which $y(x) = 1$, and b) the sentences $\{x\}$ for which $y(x_n) = 1$, averaged across names $n \in N$; i.e., $\underset{n \in N}{E}[Jaccard(\{x|y(x) = 1\}, \{x|y(x_n) = 1\})]$, where $Jaccard(A, B) = 1 - |A \cap B|/|A \cup B|$.

### 2.1 Assumptions

The underlying assumption of PSA is that the model should ideally be *not* sensitive to name perturbation. However, this assumption may not always hold true. Proper names *do* convey meaning akin to the linguistic meanings expressed in more general phrases, and thus perturbing names may sometimes remove critical semantic content that an NLP system should be modelling. For example, *he is like Hitler* vs. *he is like Gandhi* should have very different sentiment scores, since the sentences evoke the pragmatics associated with those referents. Whether the PSA assumption holds in individual sentences will depend on the sentential context; however, the corpus-level trends that we measure in the model scores/labels are still indicative of systemic biases in the model. This points to the importance of paying care to how the corpus $X$ is constructed, and making sure that it captures a diverse set of sentential contexts.

### 2.2 Analysis Framework

The PSA framework described above is applicable to any text classification models, on any target corpus, to detect bias with respect to any type of named entities (i.e., perturbable among each other). In this paper, we focus on two text classification models, applied to 4 different corpora, to detect biases associated with person names.

**Models:** We use two text classification models: a) a toxicity model that returns a score between [0,1], and b) a sentiment model that returns a score between [-1,+1]. Both models were trained using state-of-the-art neural network algorithms, and perform competitively on benchmark tests.[2]

[2]To obtain information about the models, for instance to perform replication experiments, please contact the authors.

**Corpora:** We use four socially and topically diverse corpora of online comments released by Voigt et al. (2018): Facebook comments on politicians' posts (FB-Pol.) and on public figures' posts (FB-Pub.), Reddit comments, and comments in Fitocracy forums. For each corpus, we select 1000 comments at random that satisfy two criteria: at most 50 words in length, and contain at least one English 3rd person singular pronouns (i.e., anchors). We use these extracted comments to build templates, where the pronouns can be replaced with a person name to form a grammatically coherent perturbed sentence. We use pronouns as the anchors for multiple reasons. Pronouns are often *closed-class* words across languages,[3] making it a useful reference cross-linguistically. Using a list of names to query for anchors is an option; but it has the risk of resulting in a set of sentences biased towards the cultural/demographic associations of those names, a confound that the use of pronouns as anchors will avoid. We balance the representation of female and male pronouns in our extracted sentences so as to minimize the effect of skew towards one gender in particular within the test set. However future work should examine how to best account for non-binary genders in this step.

**Names:** We choose a list of controversial personalities, compiled based on Wikipedia page edit frequency.[4] Because of their controversial nature, these names are more likely to have social biases associated with them, which is helpful to demonstrate the utility of our analysis.

## 3 Results

Table 2 shows the results of perturbation sensitivity analysis on different corpora. Both models exhibit significant sensitivity towards name perturbation across all 4 corpora. On average, sentences subjected to name perturbation resulted in a wide range of scores; i.e., *ScoreRange* over 0.10 for toxicity, and 0.36-0.42 for sentiment. Similarly, *ScoreDev* values for the sentiment model is also higher (over 0.07 across board) compared to that of the toxicity model (around 0.02), suggesting that the sentiment model is much more sensitive to the named entities present in text than the toxicity model. We also observe that perturbation

[3]While the assumption that pronouns are a closed-class is useful for many languages, Japanese and Malay are example languages where this assumption does not hold.

[4]https://anon.to/x9PMYo

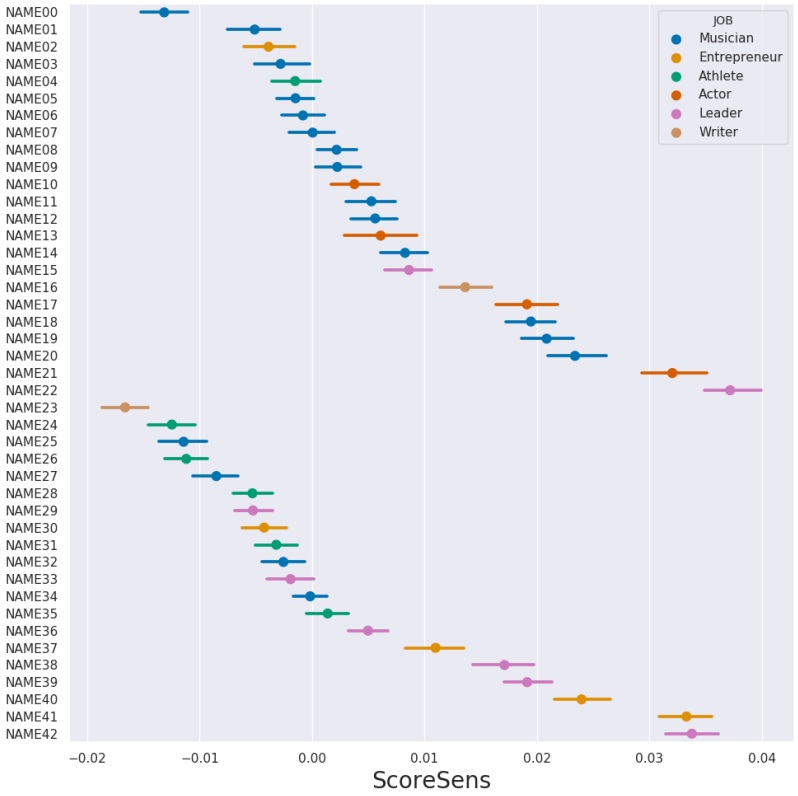


Figure 1: Name Perturbation Sensitivity (*ScoreSens*) for the toxicity model on the Reddit subcorpus, across names of controversial personalities. Female names are at the top; male names at the bottom; colors distinguish their career type. Names have been obfuscated due to their sensitive nature.

| | Toxicity | | Sentiment | |
|---|---|---|---|---|
| Corpus | *ScoreDev* | *ScoreRange* | *ScoreDev* | *ScoreRange* |
| FB-Pol. | 0.022 | 0.107 | 0.070 | 0.360 |
| FB-Pub. | 0.025 | 0.118 | 0.083 | 0.420 |
| Reddit | 0.022 | 0.107 | 0.072 | 0.376 |
| Fitocracy | 0.022 | 0.103 | 0.071 | 0.364 |

Table 2: *ScoreDev* is the per-sentence standard deviation of scores upon name perturbation, averaged across all sentences. *ScoreRange* is the per-sentence range of scores (i.e., max - min) upon name perturbation, averaged across all sentences.

sensitivity is a function of the target corpus; comments on public figures had a much larger *ScoreDev* and *ScoreRange* for both tasks.

### 3.1 Bias Towards Specific Names

We now analyze the *ScoreSens* for specific names. Figure 1 shows the *ScoreSens* for each name in our list, for the Toxicity-Reddit combination. Names are obfuscated in the figure due to their sensitive nature, but their career type is distinguished. Replacing a pronoun with some names increases the toxicity scores by over 0.03 on average, while other names decrease the scores by almost 0.02 on average. It is also notable that leaders (politicians) and actors in our list have higher toxicity associations than musicians and athletes. Similar effects also occur in the sentiment analysis model.

### 3.2 Threshold Analysis

Whether a score difference caused by perturbation results in a different label or not depends also on the threshold. It is possible that a model would be more stable on sentences with highly toxic language, but the effect of perturbation is more prevalent in sentences that have fewer signals of toxicity. We verified this to be the case in our analysis: the average (over all names) value of the perturbation score sensitivity, i.e. $|f(x_n) - f(x)|$, has a significant moderate negative correlation (-0.48) with the original score of that sentence, $f(x)$. This finding is of importance to counter-factual token fairness approaches such as (Garg et al., 2019).

To further understand the impact of perturbation sensitivity, we calculate *LabelDist*, which takes into account the number of sentences that switch either from toxic to non-toxic or vice versa, when a pronoun is changed to a name. Figure 2 shows *LabelDist* values across different thresholds. As can be seen from the Figure, the name perturbation

results in a *LabelDist* of 0.10 – 0.40 across thresholds. This roughly suggests that around 10-40% of sentences (with third person singular pronouns) labeled as toxic at any given threshold could flip the label as a result of name perturbation. It is also interesting to note that despite the negative correlation between $|f(x_n) - f(x)|$ and $f(x)$, the *LabelDist* has high values at high thresholds.

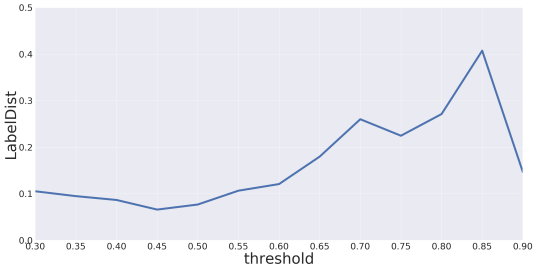


Figure 2: Even for high model thresholds, we see significant name perturbation sensitivity in classifications/labels. *LabelDist* measures the # of flips between *toxic* and *non-toxic*.

## 4 Related Work

Fairness research in NLP has seen tremendous growth in the past few years (e.g., (Bolukbasi et al., 2016; Caliskan et al., 2017; Webster et al., 2018; Díaz et al., 2018; Dixon et al., 2018; De-Arteaga et al., 2019; Gonen and Goldberg, 2019; Manzini et al., 2019)) over a range of NLP tasks such as co-reference resolution and machine translation, as well as the tasks we studied — sentiment analysis and toxicity prediction. Some of this work study bias by swapping names in sentence templates (Caliskan et al., 2017; Kiritchenko and Mohammad, 2018; May et al., 2019; Gonen and Goldberg, 2019); however they use synthetic sentence templates, while we extract naturally occurring sentences from the target corpus.

Our work is closely related to counter-factual token fairness (Garg et al., 2019), which measures the magnitude of model prediction change when identity terms (such as *gay*, *lesbian*, *transgender* etc.) referenced in naturally occurring sentences are perturbed. Additionally, De-Arteaga et al. (2019) study gender bias in occupation classification using names in online biographies. In contrast, we propose a general framework to study biases with named entities. Our work is also related to the work on interpreting neural network models by manipulating input text (Li et al., 2016); while their aim is to interpret model weights, we study the model outputs for biases.

## 5 Discussion and Conclusion

Social biases towards real-world entities are often reflected in the language that mentions them, and such biases can be unintentionally perpetuated to trained models. The focus of this paper is to introduce a simple method, Perturbation Sensitivity Analysis, to test for unwanted biases in an NLP model. Our method can be employed to study biases towards individuals (as demonstrated here), or towards groups (e.g., races, genders), and is flexible enough to be applied across domains.

We are motivated to provide solutions for end users of NLP systems, who are likely to use models outside of their original training/testing environments, e.g., on data from populations or platforms that the system was not explicitly trained on. The relative simplicity of the proposed approach suggests that the same method may be applied in different genres and across different languages, provided that a set of anchors are provided, such as pronouns in the target language. Pronouns' status cross-linguistically as closed-class – high frequency and easily listed as a small set of words – make them particularly amenable for serving as a starting point for open domain bias analyses.

After identifying unwanted biases in a model, a next logical step is to reduce these biases. Adapting the proposed approach to model training is straightforward, either by perturbing names in the training data directly, or by estimating the likelihood of given annotations as a function of sentence perturbation. Without access to model retraining, a simple solution could use post-processing to return system scores as a function of perturbed sentences, such as by averaging scores across perturbed sentences.

Future work could employ our method to study various group biases such as nationality, caste, and religion, since person names may function as significant markers for many such demographic associations. Our method could also be easily extended to other kinds of NLP models (beyond classification) as well as other types of entities (locations, organizations etc.).

**Acknowledgements** We would like to thank the anonymous reviewers for their helpful and constructive feedback. We also thank Dylan Baker, Emily Denton, Yoni Halpern, Ben Packer, Lucy Vasserman, Kellie Webster, and Simone Wu for their valuable discussions on this paper.

## References


Su Lin Blodgett and Brendan O'Connor. 2017. Racial disparity in natural language processing: A case study of social media African-American English. *arXiv preprint arXiv:1707.00061*.

Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Advances in neural information processing systems*, pages 4349–4357.

Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2019. Nuanced metrics for measuring unintended bias with real data for text classification. In *Proceedings of the FATES 2019 Workshop on Fairness, Accountability, Transparency, Ethics, and Society on the Web*.

Aylin Caliskan, Joanna J Bryson, and Arvind Narayanan. 2017. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186.

Alexandra Chouldechova and Aaron Roth. 2018. The frontiers of fairness in machine learning. *arXiv preprint arXiv:1810.08810*.

Maria De-Arteaga, Alexey Romanov, Hanna Wallach, Jennifer Chayes, Christian Borgs, Alexandra Chouldechova, Sahin Geyik, Krishnaram Kenthapadi, and Adam Tauman Kalai. 2019. Bias in bios: A case study of semantic representation bias in a high-stakes setting. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 120–128. ACM.

Mark Díaz, Isaac Johnson, Amanda Lazar, Anne Marie Piper, and Darren Gergle. 2018. Addressing age-related bias in sentiment analysis. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 412. ACM.

Lucas Dixon, John Li, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2018. Measuring and mitigating unintended bias in text classification. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pages 67–73. ACM.

Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. 2012. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference*, pages 214–226. ACM.

Michael Feldman, Sorelle A Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. 2015. Certifying and removing disparate impact. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 259–268. ACM.

Sahaj Garg, Vincent Perot, Nicole Limtiaco, Ankur Taly, Ed H Chi, and Alex Beutel. 2019. Counterfactual fairness in text classification through robustness. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pages 219–226. ACM.

Hila Gonen and Yoav Goldberg. 2019. Lipstick on a pig: Debiasing methods cover up systematic gender biases in word embeddings but do not remove them. In *Proceedings of NAACL*, pages 609–614, Minneapolis, Minnesota. Association for Computational Linguistics.

Svetlana Kiritchenko and Saif Mohammad. 2018. Examining gender and race bias in two hundred sentiment analysis systems. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 43–53, New Orleans, Louisiana. Association for Computational Linguistics.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220*.

Thomas Manzini, Lim Yao Chong, Alan W Black, and Yulia Tsvetkov. 2019. Black is to criminal as caucasian is to police: Detecting and removing multiclass bias in word embeddings. In *Proceedings of NAACL*, pages 615–621, Minneapolis, Minnesota. Association for Computational Linguistics.

Chandler May, Alex Wang, Shikha Bordia, Samuel R. Bowman, and Rachel Rudinger. 2019. On measuring social biases in sentence encoders. In *Proceedings of NAACL*, pages 622–628, Minneapolis, Minnesota. Association for Computational Linguistics.

Shira Mitchell, Eric Potash, and Solon Barocas. 2018. Prediction-based decisions and fairness: A catalogue of choices, assumptions, and definitions. *arXiv:1811.07867*.

Andrew D Selbst, Danah Boyd, Sorelle A Friedler, Suresh Venkatasubramanian, and Janet Vertesi. 2019. Fairness and abstraction in sociotechnical systems. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*. ACM.

Rob Voigt, David Jurgens, Vinodkumar Prabhakaran, Dan Jurafsky, and Yulia Tsvetkov. 2018. Rtgender: A corpus for studying differential responses to gender. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Paris, France. European Language Resources Association (ELRA).

Kellie Webster, Marta Recasens, Vera Axelrod, and Jason Baldridge. 2018. Mind the gap: A balanced corpus of gendered ambiguous pronouns. *Transactions of the Association for Computational Linguistics*, 6:605–617.

# Automatically Inferring Gender Associations from Language

**Serina Chang**
Department of Computer Science
Columbia University*
serinac@stanford.edu

**Kathleen McKeown**
Department of Computer Science
Columbia University
kathy@cs.columbia.edu

## Abstract

In this paper, we pose the question: do people talk about women and men in different ways? We introduce two datasets and a novel integration of approaches for automatically inferring gender associations from language, discovering coherent word clusters, and labeling the clusters for the semantic concepts they represent. The datasets allow us to compare how people write about women and men in two different settings - one set draws from celebrity news and the other from student reviews of computer science professors. We demonstrate that there *are* large-scale differences in the ways that people talk about women and men and that these differences vary across domains. Human evaluations show that our methods significantly outperform strong baselines.

## 1 Introduction

It is well-established that gender bias exists in language – for example, we see evidence of this given the prevalence of sexism in abusive language datasets (Waseem and Hovy, 2016; Jha and Mamidi, 2017). However, these are extreme cases of gender norms in language, and only encompass a small proportion of speakers or texts.

Less studied in NLP is how gender norms manifest in everyday language – do people talk about women and men in different ways? These types of differences are far subtler than abusive language, but they can provide valuable insight into the roots of more extreme acts of discrimination. Subtle differences are difficult to observe because each case on its own could be attributed to circumstance, a passing comment or an accidental word. However, at the level of hundreds of thousands of data points, these patterns, if they do exist, become undeniable. Thus, in this work, we introduce new datasets and methods so that we can study subtle gender associations in language at the large-scale.

Our contributions include:

- Two datasets for studying language and gender, each consisting of over 300K sentences.
- Methods to infer gender-associated words and labeled clusters in any domain.
- Novel findings that demonstrate in both domains that people do talk about women and men in different ways.

Each contribution brings us closer to modeling how gender associations appear in everyday language. In the remainder of the paper, we present related work, our data collection, methods and findings, and human evaluations of our system.[1]

## 2 Related Work

The study of gender and language has a rich history in social science. Its roots are often attributed to Robin Lakoff, who argued that language is fundamental to gender inequality, "reflected in both the ways women are expected to speak, and the ways in which women are spoken of" (Lakoff, 1973). Prominent scholars following Lakoff have included Deborah Tannen (1990), Mary Bucholtz and Kira Hall (1995), Janet Holmes (2003), Penelope Eckert (2003), and Deborah Cameron (2008), along with many others.

In recent decades, the study of gender and language has also attracted computational researchers. Echoing Lakoff's original claim, a popular strand of computational work focuses on differences in how women and men talk, analyzing key lexical traits (Boulis and Ostendorf, 2005; Argamon et al., 2007; Bamman et al., 2014) and predicting a person's gender from some text they have written (Rao et al., 2010; Jurgens et al.,

* Since writing this paper, Serina Chang has moved to the Department of Computer Science at Stanford University.

[1] Our datasets and code are available at cs.columbia.edu/nlp/tools.cgi#gendered%20corpus and github.com/serinachang5/gender-associations, respectively.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5746–5752, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

2017). There is also research studying how people talk *to* women and men (Voigt et al., 2018), as well as how people talk *about* women and men, typically in specific domains such as sports journalism (Fu et al., 2016), fiction writing (Fast et al., 2016), movie scripts (Sap et al., 2017), and Wikipedia biographies (Wagner et al., 2015, 2016). Our work builds on this body by diving into two novel domains: celebrity news, which explores gender in pop culture, and student reviews of computer science (CS) professors, which examines gender in academia and, particularly, the historically male-dominated field of CS. Furthermore, many of these works rely on manually constructed lexicons or topics to pinpoint gendered language, but our methods automatically infer gender-associated words and labeled clusters, thus reducing supervision and increasing the potential to discover subtleties in the data.

Modeling gender associations in language could also be instrumental to other NLP tasks. Abusive language is often founded in sexism (Waseem and Hovy, 2016; Jha and Mamidi, 2017), so models of gender associations could help to improve detection in those cases. Gender bias also manifests in NLP pipelines: prior research has found that word embeddings preserve gender biases (Bolukbasi et al., 2016; Caliskan et al., 2017; Garg et al., 2018), and some have developed methods to reduce this bias (Zhao et al., 2018, 2019). Yet, the problem is far from solved; for example, Gonen and Goldberg 2019 showed that it is still possible to recover gender bias from "de-biased" embeddings. These findings further motivate our research, since before we can fully reduce gender bias in embeddings, we need to develop a deeper understanding of how gender permeates through language in the first place.

We also build on methods to cluster words in word embedding space and automatically label clusters. Clustering word embeddings has proven useful for discovering salient patterns in text corpora (Wilson et al., 2018; Demszky et al., 2019). Once clusters are derived, we would like them to be interpretable. Much research simply considers the top-$n$ words from each cluster, but this method can be subjective and time-consuming to interpret. Thus, there are efforts to design methods of automatic cluster labeling (Manning et al., 2008). We take a similar approach to Poostchi and Piccardi 2018, who leverage word embeddings and WordNet during labeling, and we extend their method with additional techniques and evaluations.

| | **Celeb** | **Professor** |
|---|---|---|
| Num. texts | 15,917 | 283,973 |
| Num. sentences | 342,645 | 976,677 |
| Fem-Male prop. | .67 / .33 | .28 / .72 |

Table 1: Summary statistics of our datasets.

## 3 Data Collection

Our first dataset contains articles from celebrity magazines *People*, *UsWeekly*, and *E!News*. We labeled each article for whether it was reporting on men, women, or neither/unknown.[2] To do this, we first extracted the article's topic tags. Some of these tags referred to people, but others to non-people entities, such as "Gift Ideas" or "Health." To distinguish between these types of tags, we queried each tag on Wikipedia and checked whether the top page result contained a "Born" entry in its infobox – if so, we concluded that the tag referred to a person.

Then, from the person's Wikipedia page, we determined their gender by checking whether the introductory paragraphs of the page contained more male or female pronouns. This method was simple but effective, since pronouns in the introduction almost always resolve to the subject of that page. In fact, on a sample of 80 tags that we manually annotated, we found that comparing pronoun counts predicted gender with perfect accuracy. Finally, if an article tagged at least one woman and did not tag any men, we labeled the article as Female; in the opposite case, we labeled it as Male.

Our second dataset contains reviews from RateMyProfessors (RMP), an online platform where students can review their professors. We included all 5,604 U.S. schools on RMP, and collected all reviews for CS professors at those schools. We labeled each review with the gender of the professor whom it was about, which we determined by comparing the count of male versus female pronouns over all reviews for that professor. This method was again effective, because the reviews are ex-

[2]Our method unfortunately could not take into account non-binary gender identities, as it relied on she/her and he/his pronouns, and could not easily integrate the singular they/them, nor could we find sufficient examples of ze/zir or other non-binary pronouns in our data. That said, we will continue striving towards better inclusion, and hope in future work to expand our methods beyond the binary.

pressly written about a certain professor, so the pronouns typically resolve to that professor.

In addition to extracting the text of the articles or reviews, for each dataset we also collected various useful metadata. For the celebrity dataset, we recorded each article's timestamp and the name of the author, if available. Storing author names creates the potential to examine the relationship between the gender of the author and the gender of the subject, such as asking if there are differences between how women write about men and how men write about men. In this work, we did not yet pursue this direction because we wanted to begin with a simpler question of how gender is discussed: regardless of the gender of the authors, what is the content being put forth and consumed? Furthermore, we were unable to extract author gender in the professor dataset since the RMP reviews are anonymous. However, in future work, we may explore the influence of author gender in the celebrity dataset.

For the professor dataset, we captured metadata such as each review's rating, which indicates how the student feels about the professor on a scale of AWFUL to AWESOME. This additional variable in our data creates the option in future work to factor in sentiment; for example, we could study whether there are differences in language used when criticizing a female versus a male professor.

## 4 Inferring Word-Level Associations

Our first goal was to discover words that are significantly associated with men or women in a given domain. We employed an approach used by Bamman et al. 2014 in their work to analyze differences in how men and women write on Twitter.

### 4.1 Methods

First, to operationalize, we say that term $i$ is *associated* with gender $j$ if, when discussing individuals of gender $j$, $i$ is used with unusual frequency – which we can check with statistical hypothesis tests. Let $f_i$ represent the likelihood of $i$ appearing when discussing women or men. $f_i$ is unknown, but we can model the distribution of all possible $f_i$ using the corpus of texts that we have from the domain. We construct a gender-balanced version of the corpus by randomly undersampling the more prevalent gender until the proportions of each gender are equal. Assuming a non-informative prior distribution on $f_i$, the posterior distribution is Beta($k_i$, $N - k_i$), where $k_i$ is the count of $i$ in the gender-balanced corpus and $N$ is the total count of words in that corpus.

As Bamman et al. 2014 discuss, "the distribution of the gender-specific counts can be described by an integral over all possible $f_i$. This integral defines the Beta-Binomial distribution (Gelman et al., 2004), and has a closed form solution." We say that term $i$ is significantly associated with gender $j$ if the cumulative distribution at $k_{ij}$ (the count of $i$ in the $j$ portion of the gender-balanced corpus) is $p \leq 0.05$. As in the original work, we apply the Bonferroni correction (Dunn, 1961) for multiple comparisons because we are computing statistical tests for thousands of hypotheses.

| **Female-Associated** | **Male-Associated** |
|---|---|
| girl, cover, husband, wedding, gown, fashion, mom, pregnancy, photo, top, hair, look | movie, president, wife, dad, death, film, host, assault, claim, misconduct, action, director |
| respond, email, recommend, help, love, accept, need, send, reply, communicate | learn, teach, know, write, lecture, challenge, solve, ramble, push, joke, bore |
| easy, rude, wonderful, kind, caring, hot, strict, timely, mean, disorganized, beautiful | knowledgeable, real, challenging, brilliant, arrogant, hard, passionate, practical |

Table 2: *Top*: Sample from the top-25 most gender-associated **nouns** in the celebrity domain. *Middle*: professor domain, sample from top-25 **verbs**. *Bottom*: professor domain, sample from top-25 **adjectives**. All associations listed are $p \leq 0.05$, with Bonferroni correction. See Appendix for all top-25 nouns, verbs, and adjectives for both genders in both domains.

### 4.2 Findings

We applied this method to discover gender-associated words in both domains. In Table 2, we present a sample of the most gender-associated nouns from the celebrity domain. Several themes emerge: for example, female celebrities seem to be more associated with appearance ("gown," "photo," "hair," "look"), while male celebrities are more associated with creating content ("movie," "film," "host," "director"). This echoes real-world trends: for instance, on the red carpet, actresses tend to be asked more questions about their appearance - what brands they are wearing, how long it took to get ready, etc. - while actors are

asked questions about their careers and creative processes (as an example, see Selby 2014).

Table 2 also includes some of the most gender-associated verbs and adjectives from the professor domain. Female CS professors seem to be praised for being communicative and personal with students ("respond," "communicate," "kind," "caring"), while male CS professors are recognized for being knowledgeable and challenging the students ("teach,", "challenge," "brilliant," "practical"). These trends are well-supported by social science literature, which has found that female teachers are praised for "personalizing" instruction and interacting extensively with students, while male teachers are praised for using "teacher as expert" styles that showcase mastery of material (Statham et al., 1991).

These findings establish that there are clear differences in how people talk about women and men – even with Bonferroni correction, there are still over 500 significantly gender-associated nouns, verbs, and adjectives in the celebrity domain and over 200 in the professor domain. Furthermore, the results in both domains align with prior studies and real world trends, which validates that our methods can capture meaningful patterns and innovatively provide evidence at the large-scale. This analysis also hints that it can be helpful to abstract from words to topics to recognize higher-level patterns of gender associations, which motivates our next section on clustering.

## 5 Clustering & Cluster Labeling

With word-level associations in hand, our next goals were to discover coherent clusters among the words and to automatically label those clusters.

### 5.1 Methods

First, we trained domain-specific word embeddings using the Word2Vec (Mikolov et al., 2013) CBOW model ($w \in R^{100}$). Then, we used k-means clustering to cluster the embeddings of the gender-associated words. Since k-means may converge at local optima, we ran the algorithm 50 times and kept the model with the lowest sum of squared errors.

To automatically label the clusters, we combined the grounded knowledge of WordNet (Miller, 1995) and context-sensitive strengths of domain-specific word embeddings. Our algorithm is similar to Poostchi and Piccardi 2018's approach, but we extend their method by introducing domain-specific word embeddings for clustering as well as a new technique for sense disambiguation. Given a cluster, our algorithm proceeds with the following three steps:

1. **Sense disambiguation**: The goal is to assign each cluster word to one of its WordNet synsets; let $S$ represent the collection of chosen synsets. We know that these words have been clustered in domain-specific embedding space, which means that in the context of the domain, these words are very close semantically. Thus, we choose $S^*$ that minimizes the total distance between its synsets.

2. **Candidate label generation**: In this step, we generate $L$, the set of possible cluster labels. Our approach is simple: we take the union of all hypernyms of the synsets in $S^*$.

3. **Candidate label ranking**: Here, we rank the synsets in $L$. We want labels that are as close to all of the synsets in $S^*$ as possible; thus, we score the candidate labels by the sum of their distances to each synset in $S^*$ and we rank them from least to most distance.

In steps 1 and 3, we use WordNet pathwise distance, but we encourage the exploration of other distance representations as well.

### 5.2 Findings

Table 3 displays a sample of our results – we find that the clusters are coherent in context and the labels seem reasonable. In the next section, we discuss human evaluations that we conducted to more rigorously evaluate the output, but first we discuss the value of these methods toward analysis.

At the word-level, we hypothesized that in the celebrity domain, women were more associated with appearance and men with creating content. Now, we can validate those hypotheses against labeled clusters – indeed, there is a cluster labeled *clothing* that is 100% female (i.e. 100% words are female-associated), and a 80% male cluster labeled *movie*. Likewise, in the professor domain, we had guessed that women are associated with communication and men with knowledge, and there is a 100% female cluster labeled *communication* and a 89% male cluster labeled *cognition*. Thus, cluster labeling proves to be very effective at

| Domain | Sample Words in Cluster | F:M | Top 3 Pred. Cluster Labels |
|---|---|---|---|
| Celeb | gown, top, dress, pant, skirt, neckline | 25:0 | *covering, cloth_covering, clothing* |
| | film, release, role, character, project | 4:16 | *movie, show, event* |
| | boyfriend, beau, hubby, wife, girlfriend | 15:7 | *lover, person, relative* |
| Professor | response, email, contact, answer | 13:0 | *statement, message, communication* |
| | material, concept, topic, stuff, subject | 1:8 | *content, idea, cognition* |
| | teacher, woman, lady, prof, guy, dude | 5:7 | *man, adult, woman* |

Table 3: Sample of our clusters and predicted cluster labels. We include in the Appendix a more comprehensive table of our results. **F:M** refers to the ratio of female-associated to male-associated words in the cluster.

pulling out the patterns that we believed we saw at the word-level, but could not formally validate.

The clusters we mentioned so far all lean heavily toward one gender association or the other, but some clusters are interesting precisely because they do not lean heavily – this allows us to see where semantic groupings do not align exactly with gender association. For example, in the celebrity domain, there is a cluster labeled *lover* that has a mix of female-associated words ("boyfriend," "beau," "hubby") and male-associated words ("wife," "girlfriend"). Jointly leveraging cluster labels and gender associations allows us to see that in the semantic context of having a *lover*, women are typically associated with male figures and men with female figures, which reflects heteronormativity in society.

## 6 Human Evaluations

To test our clusters, we employed the *Word Intrusion* task (Chang et al., 2009). We present the annotator with five words – four drawn from one cluster and one drawn randomly from the domain vocabulary – and we ask them to pick out the intruder. The intuition is that if the cluster is coherent, then an observer should be able to identify the out-of-cluster word as the intruder. For both domains, we report results on all clusters and on the top 8, ranked by ascending normalized sum of squared errors, which can be seen as a prediction of coherence. In the celebrity domain, annotators identified the out-of-cluster word 73% of the time in the top-8 and 53% overall. In the professor domain, annotators identified it 60% of the time in the top-8 and 49% overall. As expected, top-8 performance in both domains does considerably better than overall, but at all levels the precision is significantly above the random baseline of 20%.

To test cluster labels, we present the annotator with a label and a word, and we ask them whether the word *falls under* the concept. The concept is a potential cluster label and the word is either a word from that cluster or drawn randomly from the domain vocabulary. For a good label, the rate at which in-cluster words fall under the label should be much higher than the rate at which out-of-cluster words fall under. In our experiments, we tested the top 4 predicted labels and the centroid of the cluster as a strong baseline label. The centroid achieved an in-cluster rate of .60 and out-of-cluster rate of .18 (difference of .42). Our best performing predicted label achieved an in-cluster rate of .65 and an out-of-cluster rate of .04 (difference of .61), thus outperforming the centroid on both rates and increasing the gap between rates by nearly 20 points. In the Appendix, we include more detailed results on both tasks.

## 7 Conclusion

We have presented two substantial datasets and a novel integration of methods to automatically infer gender associations in language. We have demonstrated that in both datasets, there are clear differences in how people talk about women and men. Furthermore, we have shown that clustering and cluster labeling are effective at identifying higher-level patterns of gender associations, and that our methods outperform strong baselines in human evaluations. In future work, we hope to use our findings to improve performance on tasks such as abusive language detection. We also hope to delve into finer-grained analyses, exploring how language around gender interacts with other variables, such as sexual orientation or profession (e.g. actresses versus female athletes). Finally, we plan to continue widening the scope of our study – for example, expanding our methods to include non-binary gender identities, evaluating changes in gender norms over time, and spreading to more domains, such as the political sphere.

## References


Shlomo Argamon, Moshe Koppel, James Pennebaker, and Jonathan Schler. 2007. Mining the blogosphere: age, gender, and the varieties of self-expression. *First Monday*, 12(9).

David Bamman, Jacob Eisenstein, and Tyler Schnoebelen. 2014. Gender identity and lexical variation in social media. *Journal of Sociolinguistics*, 18(2).

Tolga Bolukbasi, Kai-Wei Chang, James Zou, Venkatesh Saligrama, and Adam Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *NeurIPS*.

Constantinos Boulis and Mari Ostendorf. 2005. A quantitative analysis of lexical differences between genders in telephone conversations. In *ACL*.

Mary Bucholtz and Kira Hall. 1995. *Gender articulated: language and the socially constructed self*. Routledge.

Aylin Caliskan, Joanna J. Bryson, and Arvind Narayanan. 2017. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334).

Deborah Cameron. 2008. *The myth of mars and venus: do men and women really speak different languages?* Oxford University Press.

Jonathan Chang, Sean Gerrish, Chong Wang, Jordan L. Boyd-Graber, and David M. Blei. 2009. Reading tea leaves: how humans interpret topic models. In *NeurIPS*.

Dorottya Demszky, Nikhil Garg, Rob Voigt, James Zou, Matthew Gentzkow, Jesse Shapiro, and Dan Jurafsky. 2019. Analyzing polarization in social media: method and application to tweets on 21 mass shootings. In *NAACL*.

Olive Jean Dunn. 1961. Multiple comparisons among means. *Journal of the American Statistical Association*, 56(293).

Penelope Eckert and Sally McConnell-Ginet. 2003. *Language and gender*. Cambridge University Press.

Ethan Fast, Tina Vachovsky, and Michael Bernstein. 2016. Shirtless and dangerous: quantifying linguistic signals of gender bias in an online fiction writing community. In *ICWSM*.

Liye Fu, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2016. Tie-breaker: using language models to quantify gender bias in sports journalism. In *IJCAI Workshop on NLP Meets Journalism*.

Nikhil Garg, Londa Schiebinger, Dan Jurafsky, and James Zou. 2018. Word embeddings quantify 100 years of gender and ethnic stereotypes. *Proceedings of the National Academy of Sciences*, 155(16).

Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. 2004. *Bayesian data analysis*. CRC Press/Chapman Hall.

Hila Gonen and Yoav Goldberg. 2019. Lipstick on a pig: debiasing methods cover up systematic gender biases in word embeddings but do not remove them. In *NAACL-HLT*.

Janet Holmes and Miriam Meyerhoff. 2003. *The handbook of language and gender*. Blackwell Publishing.

Akshita Jha and Radhika Mamidi. 2017. When does a compliment become sexist? analysis and classification of ambivalent sexism using twitter data. In *ACL Workshop on NLP and Computational Social Science*.

David Jurgens, Yulia Tsvetkov, and Dan Jurafsky. 2017. Writer profiling without the writers text. In *SocInfo*.

Robin Lakoff. 1973. Language and woman's place. *Language in Society*, 2(1).

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schutze. 2008. *Introduction to information retrieval*. Cambridge University Press.

Tomas Mikolov, Kai Chen, Gregory Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *ICLR Workshop*.

George A. Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11).

Hanieh Poostchi and Massimo Piccardi. 2018. Cluster labeling by word embeddings and wordnet's hypernymy. In *Australasian Language Technology Association Workshop*.

Delip Rao, David Yarowsky, Abhishek Shreevats, and Manaswi Gupta. 2010. Classifying latent user attributes in twitter. In *SMUC*.

Maarten Sap, Marcella Cindy Prasetio, Ari Holtzman, Hannah Rashkin, and Yejin Choi. 2017. Connotation frames of power and agency in modern films. In *EMNLP*.

Jenn Selby. 2014. Cate blanchett calls out red carpet sexism: 'do you do this to the guys?'. In *Independent*.

Anna Statham, Laurel Richardson, and Judith A. Cook. 1991. *Gender and university teaching: a negotiated difference*. SUNY Press.

Deborah Tannen. 1990. *You just dont understand: men and women in conversation*. Ballantine Books.

Rob Voigt, David Jurgens, Vinodkumar Prabhakaran, Dan Jurafsky, and Yulia Tsvetkov. 2018. Rtgender: a corpus for studying differential responses to gender. In *LREC*.

Claudia Wagner, David Garcia, Mohsen Jadidi, and Markus Strohmaier. 2015. It's a man's wikipedia? assessing gender inequality in an online encyclopedia. In *ICWSM*.

Claudia Wagner, Eduardo Graells-Garrido, David Garcia, and Filippo Menczer. 2016. Women through the glass ceiling: gender asymmetries in wikipedia. *EPJ Data Sci*, 5(5).

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *NAACL-HLT*.

Steven Wilson, Yiting Shen, and Rada Mihalcea. 2018. Building and validating hierarchical lexicons with a case study on personal values. In *SocInfo*.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Ryan Cotterell, Vicente Ordonez, and Kai-Wei Chang. 2019. Gender bias in contextualized word embeddings. In *NAACL-HLT*.

Jieyu Zhao, Yichao Zhou, Zeyu Li, Wei Wang, and Kai-Wei Chang. 2018. Learning gender-neutral word embeddings. In *EMNLP*.

# Reporting the Unreported: Event Extraction for Analyzing the Local Representation of Hate Crimes

**Aida Mostafazadeh Davani, Leigh Yeh, Mohammad Atari, Brendan Kennedy**
**Gwenyth Portillo-Wightman, Elaine Gonzalez, Natalie Delong, Rhea Bhatia,**
**Arineh Mirinjian, Xiang Ren, Morteza Dehghani**

University of Southern California

## Abstract

Official reports of hate crimes in the US are under-reported relative to the actual number of such incidents. Further, despite statistical approximations, there are no official reports from a large number of US cities regarding incidents of hate. Here, we first demonstrate that event extraction and multi-instance learning, applied to a corpus of local news articles, can be used to predict instances of hate crime. We then use the trained model to detect incidents of hate in cities for which the FBI lacks statistics. Lastly, we train models on predicting homicide and kidnapping, compare the predictions to FBI reports, and establish that incidents of hate are indeed under-reported, compared to other types of crimes, in local press.

## 1 Introduction

Hate crimes are defined as crimes of violence either against a person or their property that display evidence of prejudice based on the victims' race, gender or gender identity, religion, disability, sexual orientation, or ethnicity (Jacobs et al., 1998). According to the results of a new Department of Justice hate crime report released in 2017 (Masucci and Langton, 2017), approximately 54% of hate crime victimizations were not reported to police during 2011-2015. Despite the recent efforts of advocacy groups, policy makers, and researchers to create reliable, national data to understand the extent and severity of hate crime victimization, the existing estimates continue to fall short (Pezzella et al., 2019).

It stands to reason that hate crimes induce local disturbance, and as a result, might be likely to get local coverage. Therefore, local news agencies can be considered a unique source of information for detecting these incidents. Here, we use a corpus of local news articles, collected from the Patch[1] website. The Patch data contain independent, hyper-local news articles compiled from local news sites.

We apply event extraction methods to identify incidents of hate crime reported in the Patch corpus for cities with no representation in FBI reports, and analyze the frequency of the extracted events compared to the number of incidents reported by the FBI. The task of labeling each article as a hate crime or not is defined as a Multi-Instance Learning (MIL) problem since each article is modeled as a sequence of sentences. Instead of predicting a label for each sentence, we use the information embedded in all the sentences of an article to determine whether the article is reporting a hate crime.

After testing the model on a set of annotated articles, we apply the trained model to cities for which the FBI does not have any reports, and we provide a lower-bound estimate on the occurrence frequency of hate crimes in those cities. Lastly, we compare the coverage of incidents of hate as reported in local news sources with coverages of two non-hate crimes, namely homicides and kidnappings, and contrast the overlap of the extracted incidents with those reports by the FBI.

Our results show that applying MIL for event extraction can help approximate the missing reports, especially in cases in which publishing the comprehensive event set faces challenges and is influenced by subjective bias.

## 2 MIL for Event Extraction

In this paper, we perform event detection and extraction on news articles based on taxonomies of acts of crime. We adapt the MIL approach for event detection developed by Wang et al. (2016), which identifies key sentences for a given article. We then use these key sentences to perform event

[1] https://www.patch.com



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5753–5757,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

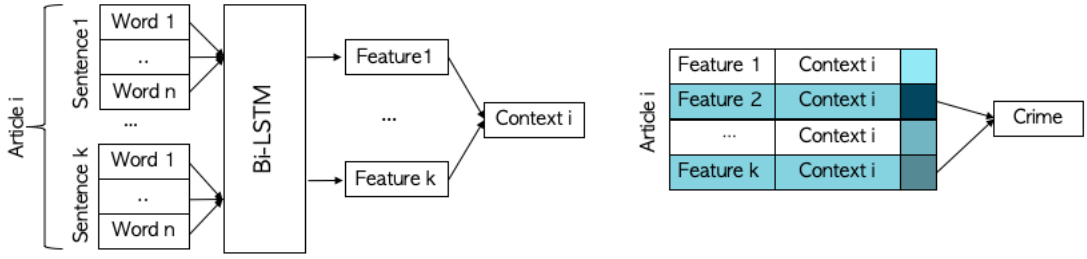


Figure 1: The event detection model using a MIL network. Local representation of each sentence are combined with context representation of its related article.

extraction, predicting the target and type of action for a given incident.

**Event Detection**

The MIL approach for document classification is illustrated in Figure 1. The two basic components are the creation of local features (representations of sentences) and the aggregation of these features into a document representation. Whereas Wang et al. (2016) applies Convolutional Neural Networks (CNNs) for the creation of local features, we use a bidirectional Long Short-Term Memory (LSTM; Hochreiter and Schmidhuber, 1997) network for representing each sentence of an article. Bidirectional networks (Graves and Schmidhuber, 2005) have been shown to provide a good semantic representation of textual data (Huang et al., 2015).

Local representations are then aggregated to form a "contextual" representation of the document, using a CNN layer. This context vector, which is the same for all sentences in the document, is then concatenated with *each* sentence's local representation.

Given the feature representation of the sentences in an article, the probabilistic score of each sentence in an article is calculated using a fully connected layer with sigmoid activation. This probabilistic score shows the extent to which the sentence contributes to predicting the crime label of the article. The label for a bag of sentences is calculated by averaging the $k$ highest probabilistic scores. We checked the results with $k$ being set to 2 or 3, since a few number of sentences in each article can determine the label.

Another prominent method which we compared the MIL results with is Hierarchical Attention Networks (HAN; Yang et al., 2016). HANs apply attention first at the level of words, then at the level of sentences, to produce representations of documents subject to local variations in textual importance. We also compare the results of neural network models with TF-IDF as a text classification baseline.

**Event Extraction**

The most challenging aspect of extracting events from a sentence is that the context of a document should be considered in order to interpret an entity and the type of triggered event (Chen et al., 2015). Approaches that exclusively use word features for the task usually lack comprehensiveness.

The event detection model in the previous section produces, for each positive prediction, a small set of sentences likely to influence the document's label. In the event extraction step, we use a bidirectional LSTM text classifier to predict the attributes of a crime event.

The attributes of a crime event are determined by the taxonomy proposed by Kennedy et al. (2018) for annotating hate rhetoric. In our case (see Section 3), we are predicting two attributes: the target of a crime event, and the type of crime.

Formulated as a multi-class, multi-task prediction, we train a biLSTM to produce a representation of the concatenation of the top two sentences and feed this to two separate feed-forward networks, one predicting the target categorization and one the crime type.

## 3 Data

The Patch website includes hyper-local news articles from 1217 cities based in the US. For this project, we scraped the articles in the "Fire and Crime" category of Patch, resulting in a corpus containing $\sim 370k$ unlabeled local news articles. For our experiments, we manually annotate subsets of the main dataset for training event detection models.

Our annotations consisted of a binary label

— whether the article represents a specific hate crime — as well as labeling the attributes of hate crime articles, which consist of the target of the action (whether the crime was based on the race, nationality, gender, religion, sexual orientation, ideology, political identification or mental/physical health of the target) and the type of action (whether the crime was an assault, arson, vandalism or hate demonstration).

For gathering a subset of articles for annotation, we filtered the news articles based on a set of 8 keywords (*swastika*, *hate*, *racial*, *religion*, *religious*, *gay*, *transgender*, *transsexual*) related to hate crimes, resulting in $\sim 3k$ patch articles, which were then combined with 500 randomly sampled articles to account for the high frequency of the hate crimes in the selected dataset. Each article was annotated for the presence and the attributes of the hate crime reports by one annotator. Annotators achieved 0.73 inter-coder agreement on a subset of 500 posts based on Cohen's Kappa (Cohen, 1968).

For hate crime articles that are not associated with the keywords, we expected the model's predictions to be sparse. To deal with this problem we applied an active learning approach introduced by Lewis and Gale (1994). In this approach, after training the model, we predicted the hate crime label for all the articles in the dataset and gathered their associated probabilities. We then selected $\sim 1k$ articles based on their probability score, using a normal distribution with a mean of 0.5 and standard deviation of 0.1. This set of articles, for which the model was uncertain about their labels, was then annotated by the same annotators and added to the training set.

We performed a similar procedure, without entity labeling and active learning, for homicide (keywords: *homicide*, *manslaughter*, *murder*, and *kill*) and kidnapping (keywords: *kidnapping*, *abduct*, *hostage*, *abduct*, and *shanghai*) events. The frequency statistics for these annotations are represented in Table 1.

| Event Type | Positive | Negative |
|---|---|---|
| Hate Crime | 1979 | 3192 |
| Homicide | 1664 | 1327 |
| Kidnapping | 1864 | 1104 |

Table 1: Frequency of events from Patch annotations.

Type and target of the hate crime was also annotated for each article. Crime type labels are distributed across assaults (900), arson (76), vandalism (450), and hate demonstrations (543). The most frequent target types were race (1029), religion (376), and sexual orientation (265).

| | MIL | HAN | TF-IDF |
|---|---|---|---|
| Hate crime | **82.9** | 82.6 | 81.6 |
| Homicide | **81.3** | 79.7 | 77.4 |
| Kidnapping | **78.7** | 75.6 | 73.9 |

Table 2: Event detection F1 scores for the test set

## 4 Experiment

All models were implemented with Tensorflow (Abadi et al., 2016). Hidden size of the LSTM cells was set to 50, filter sizes of the CNN were set to 2, 3 and 4, and a dropout layer was placed on top of the LSTM cell to set 25% of the values to zero. Each batch included 5 articles converted to their latent representation using 300-dimensional GloVe word embeddings (Pennington et al., 2014). Parameter tuning was performed with 70% of the dataset as the train set and 10% as development set and the learning rate was set to 0.00008.

All three models for predicting hate crime, kidnapping and homicide were trained for 50 epochs.

## 5 Results

The resulting F1-scores are calculated for the test set and represented in Table 2.

We apply the learned models to make predictions about the rate of hate crimes in cities for which the FBI lacks data. We also compare the relative rate of news coverage of hate crime with those of homicides and kidnappings.

**Predicting Hate Crime**

First, we compare the positive hate crime labels predicted for Patch with the FBI's city-level hate crime reports. After applying the trained model to the Patch dataset, we captured 3152 articles that report hate crime incidents. These articles include 678 reports from 286 cities that have no representation in the FBI reports. This suggests that the MIL model applied to the local news dataset can approximate missing statistics on hate crime in those cities. However, presuming a one-to-one relation between the news articles and hate crime incidents is not accurate, since there can be false positive results and duplicated articles about an incident. To provide an accurate set of unreported

hate crime incidents we removed duplicated and misclassified articles from the set of 678 unrepresented hate crime incidents.

In order to account for the possible duplications, we utilize the event extraction model to capture the event entities, namely target and action type. Running the extraction model with the same hyperparameters yields the results presented in Table 3. We use the entities together with the time (mentioned in the dataset) and location (extracted with named entity recognizer of CoreNLP (Manning et al., 2014)) of the articles to detect duplicated events.

| Label | Precision | Recall | F1 |
|---|---|---|---|
| Target | 63.9 | 65.3 | 63.9 |
| Action | 67.7 | 68.0 | 67.4 |

Table 3: Event extraction scores of MIL

After checking for pairs of articles from the same state and city, with the same reported target victim and crime action, reported at most one day apart from each other, we found 20 pairs of duplicated articles, indicating 658 unique incidents of hate in the cities with no representation in the FBI dataset.

Next, we manually checked these articles and found 315 articles that were correctly labeled as hate crime. Table 4 represents a few instances of false positives. Exploring the false positive results indicates that non-hate crime articles that mention minority social groups are often incorrectly labeled as hate crime. This issue can be explored further in future works to improve the accuracy of the predictions.

**Comparisons to Other Crime**

In order to compare the coverage of incidents of hate with coverage of homicides and kidnappings, we contrast the overlap of the extracted incidents with those reported by the FBI. Specifically, for 159 cities that have representation of the three crimes both in Patch and FBI crime reports, we calculate the ratio of Patch-based predictions to FBI reports for each crime.

To investigate the differences between the distributions of these ratios, we ran a Welch-type one-way ANOVA, which is robust to non-normal distributions, allowing for heteroscedasticity and extreme non-normality of ratios in our data (Field and Wilcox, 2017). The results indicates that the three crimes' distributions have significantly different medians ($F[2,214.28] = 102.03$, $p < 0.001$). Post-hoc tests suggested that Patch-based estimates of hate are significantly lower than homicide and kidnapping (both $p$'s $< 0.001$).

| | False Positive Examples |
|---|---|
| 1 | A former Ku Klux Klan leader in Ozark was sentenced Thursday to a decade in prison for sexually abusing a woman in southern Alabama. |
| 2 | The FBI is part of an investigation into a suspicious substance delivered to a Council on American-Islamic Relations office in Santa Clara on Thursday. |
| 3 | The hate from the violent white nationalist gathering that resulted in the death of an anti-racism protester in Charlottesville, can be found anywhere. |

Table 4: First sentences of sample articles recognized as false positive results by our annotators.

## 6 Discussion

Hate crimes in the US remain vastly underreported (Masucci and Langton, 2017). For instance, only 12.6% of the agencies in the FBI report indicated that hate crimes had occurred in their jurisdictions in 2017, and agencies as large as the Miami Police Department reported zero incidents of hate (FBI, 2018), which seem unrealistic. The contributions of this paper are two-fold: First, we have shown that event detection can be applied to the study of hate crimes. Specifically, we demonstrated that using MIL for event detection, in conjunction with local news articles, can provide conservative estimates of the occurrence of hate crimes in cities with no official representation. A possible application of this method is creating a real-time hate crime detector based on online local news agencies, providing researchers and community workers a lower-bound on the number of hate crimes in such locations. Second, the statistical analyses suggested that the scope of hate crime coverage in local news is lower than that of violent, but non-hate crimes. This suggests that although local news sources can be used as an additional source for gathering better statistics about hate crimes, the predictions of our models are simply lower bound estimates.

## References


Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 167–176.

Jacob Cohen. 1968. Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological bulletin*, 70(4):213.

FBI. 2018. Hate crime statistics, 2017. https://ucr.fbi.gov/hate-crime/2017. Accessed: 03-04-2019.

Andy P Field and Rand R Wilcox. 2017. Robust statistical methods: A primer for clinical psychology and experimental psychopathology researchers. *Behaviour research and therapy*, 98:19–38.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5-6):602–610.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

James B Jacobs, Kimberly Potter, et al. 1998. *Hate crimes: Criminal law & identity politics*. Oxford University Press on Demand.

Brendan Kennedy, Drew Kogon, Kris Coombs, Joseph Hoover, Christina Park, Gwenyth Portillo-Wightman, Aida Mostafazadeh Davani, Mohammad Atari, and Morteza Dehghani. 2018. A typology and coding manual for the study of hate-based rhetoric. *PsyArXiv*.

David D Lewis and William A Gale. 1994. A sequential algorithm for training text classifiers. In *SIGIR94*, pages 3–12. Springer.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

Madeline Masucci and Lynn Langton. 2017. Hate crime victimization, 2004-2015. *Washington, DC, US Department of Justice Office of Justice Programs Bureau of Justice Statistics*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Frank S Pezzella, Matthew D Fetzer, and Tyler Keller. 2019. The dark figure of hate crime underreporting. *American Behavioral Scientist*, page 0002764218823844.

Wei Wang, Yue Ning, Huzefa Rangwala, and Naren Ramakrishnan. 2016. A multiple instance learning framework for identifying key sentences and detecting events. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 509–518. ACM.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

# Minimally Supervised Learning of Affective Events Using Discourse Relations

**Jun Saito** **Yugo Murawaki** **Sadao Kurohashi**
Graduate School of Informatics, Kyoto University
Yoshida-honmachi, Sakyo-ku, Kyoto, 606-8501, Japan
{saito, murawaki, kuro}@nlp.ist.i.kyoto-u.ac.jp

## Abstract

Recognizing affective events that trigger positive or negative sentiment has a wide range of natural language processing applications but remains a challenging problem mainly because the polarity of an event is not necessarily predictable from its constituent words. In this paper, we propose to propagate affective polarity using discourse relations. Our method is simple and only requires a very small seed lexicon and a large raw corpus. Our experiments using Japanese data show that our method learns affective events effectively without manually labeled data. It also improves supervised learning results when labeled data are small.

## 1 Introduction

Affective events (Ding and Riloff, 2018) are events that typically affect people in positive or negative ways. For example, getting money and playing sports are usually positive to the experiencers; catching cold and losing one's wallet are negative. Understanding affective events is important to various natural language processing (NLP) applications such as dialogue systems (Shi and Yu, 2018), question-answering systems (Oh et al., 2012), and humor recognition (Liu et al., 2018). In this paper, we work on recognizing the polarity of an affective event that is represented by a score ranging from $-1$ (negative) to 1 (positive).

Learning affective events is challenging because, as the examples above suggest, the polarity of an event is not necessarily predictable from its constituent words. Combined with the unbounded combinatorial nature of language, the non-compositionality of affective polarity entails the need for large amounts of world knowledge, which can hardly be learned from small annotated data.

In this paper, we propose a simple and effective method for learning affective events that only requires a very small seed lexicon and a large raw corpus. As illustrated in Figure 1, our key idea is that we can exploit discourse relations (Prasad et al., 2008) to efficiently propagate polarity from seed predicates that directly report one's emotions (e.g., "to be glad" is positive). Suppose that events $x_1$ are $x_2$ are in the discourse relation of CAUSE (i.e., $x_1$ causes $x_2$). If the seed lexicon suggests $x_2$ is positive, $x_1$ is also likely to be positive because it triggers the positive emotion. The fact that $x_2$ is known to be negative indicates the negative polarity of $x_1$. Similarly, if $x_1$ and $x_2$ are in the discourse relation of CONCESSION (i.e., $x_2$ in spite of $x_1$), the reverse of $x_2$'s polarity can be propagated to $x_1$. Even if $x_2$'s polarity is not known in advance, we can exploit the tendency of $x_1$ and $x_2$ to be of the same polarity (for CAUSE) or of the reverse polarity (for CONCESSION) although the heuristic is not exempt from counterexamples. We transform this idea into objective functions and train neural network models that predict the polarity of a given event.

We trained the models using a Japanese web corpus. Given the minimum amount of supervision, they performed well. In addition, the combination of annotated and unannotated data yielded a gain over a purely supervised baseline when labeled data were small.

## 2 Related Work

Learning affective events is closely related to sentiment analysis. Whereas sentiment analysis usually focuses on the polarity of what are described (e.g., movies), we work on how people are typically affected by events. In sentiment analysis, much attention has been paid to compositionality. Word-level polarity (Takamura et al., 2005;



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5758–5765,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

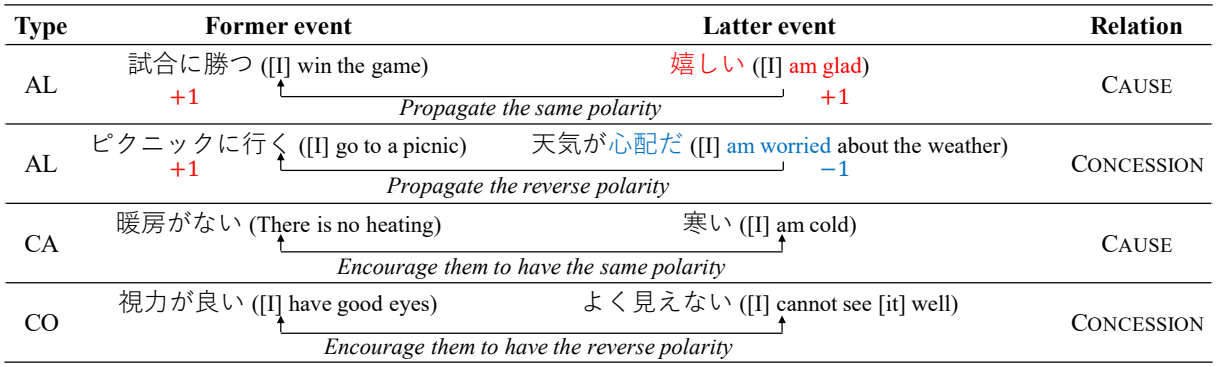


Figure 1: An overview of our method. We focus on pairs of events, the **former events** and the **latter events**, which are connected with a discourse **relation**, CAUSE or CONCESSION. Dropped pronouns are indicated by brackets in English translations. We divide the event pairs into three **types**: **AL**, **CA**, and **CO**. In AL, the polarity of a latter event is automatically identified as either positive or negative, according to the seed lexicon (the positive word is colored red and the negative word blue). We propagate the latter event's polarity to the former event. The same polarity as the latter event is used for the discourse relation CAUSE, and the reversed polarity for CONCESSION. In CA and CO, the latter event's polarity is not known. Depending on the discourse relation, we encourage the two events' polarities to be the same (CA) or reversed (CO). Details are given in Section 3.2.

Wilson et al., 2005; Baccianella et al., 2010) and the roles of negation and intensification (Reitan et al., 2015; Wilson et al., 2005; Zhu et al., 2014) are among the most important topics. In contrast, we are more interested in recognizing the sentiment polarity of an event that pertains to commonsense knowledge (e.g., getting money and catching cold).

Label propagation from seed instances is a common approach to inducing sentiment polarities. While Takamura et al. (2005) and Turney (2002) worked on word- and phrase-level polarities, Ding and Riloff (2018) dealt with event-level polarities. Takamura et al. (2005) and Turney (2002) linked instances using co-occurrence information and/or phrase-level coordinations (e.g., "$A$ and $B$" and "$A$ but $B$"). We shift our scope to event pairs that are more complex than phrase pairs, and consequently exploit discourse connectives as event-level counterparts of phrase-level conjunctions.

Ding and Riloff (2018) constructed a network of events using word embedding-derived similarities. Compared with this method, our discourse relation-based linking of events is much simpler and more intuitive.

Some previous studies made use of document structure to understand the sentiment. Shimizu et al. (2018) proposed a sentiment-specific pre-training strategy using unlabeled dialog data (tweet-reply pairs). Kaji and Kitsuregawa (2006) proposed a method of building a polarity-tagged corpus (ACP Corpus). They automatically gathered sentences that had positive or negative opinions utilizing HTML layout structures in addition to linguistic patterns. Our method depends only on raw texts and thus has wider applicability.

## 3 Proposed Method

### 3.1 Polarity Function

Our goal is to learn the polarity function $p(x)$, which predicts the sentiment polarity score of an event $x$. We approximate $p(x)$ by a neural network with the following form:

$$p(x) = \tanh(\mathrm{Linear}(\mathrm{Encoder}(x))). \quad (1)$$

Encoder outputs a vector representation of the event $x$. Linear is a fully-connected layer and transforms the representation into a scalar. tanh is the hyperbolic tangent and transforms the scalar into a score ranging from $-1$ to 1. In Section 4.2, we consider two specific implementations of Encoder.

### 3.2 Discourse Relation-Based Event Pairs

Our method requires a very small seed lexicon and a large raw corpus. We assume that we can automatically extract discourse-tagged event pairs, $(x_{i1}, x_{i2})$ $(i = 1, \cdots)$ from the raw corpus. We refer to $x_{i1}$ and $x_{i2}$ as *former* and *latter* events, respectively. As shown in Figure 1, we limit our scope to two discourse relations: CAUSE and CONCESSION.

The seed lexicon consists of positive and negative predicates. If the predicate of an extracted

event is in the seed lexicon and does not involve complex phenomena like negation, we assign the corresponding polarity score (+1 for positive events and −1 for negative events) to the event. We expect the model to automatically learn complex phenomena through label propagation. Based on the availability of scores and the types of discourse relations, we classify the extracted event pairs into the following three types.

**AL (Automatically Labeled Pairs)** The seed lexicon matches (1) the latter event but (2) not the former event, and (3) their discourse relation type is CAUSE or CONCESSION. If the discourse relation type is CAUSE, the former event is given the same score as the latter. Likewise, if the discourse relation type is CONCESSION, the former event is given the opposite of the latter's score. They are used as reference scores during training.

**CA (CAUSE Pairs)** The seed lexicon matches neither the former nor the latter event, and their discourse relation type is CAUSE. We assume the two events have the same polarities.

**CO (CONCESSION Pairs)** The seed lexicon matches neither the former nor the latter event, and their discourse relation type is CONCESSION. We assume the two events have the reversed polarities.

### 3.3 Loss Functions

Using AL, CA, and CO data, we optimize the parameters of the polarity function $p(x)$. We define a loss function for each of the three types of event pairs and sum up the multiple loss functions.

We use mean squared error to construct loss functions. For the AL data, the loss function is defined as:

$$\mathcal{L}_{\mathrm{AL}} = \frac{1}{N_{\mathrm{AL}}} \sum_{i=1}^{N_{\mathrm{AL}}} (r_{i2} - p(x_{i2}))^2 + \lambda_{\mathrm{AL}} \frac{1}{N_{\mathrm{AL}}} \sum_{i=1}^{N_{\mathrm{AL}}} (r_{i1} - p(x_{i1}))^2, \quad (2)$$

where $x_{i1}$ and $x_{i2}$ are the $i$-th pair of the AL data. $r_{i1}$ and $r_{i2}$ are the automatically-assigned scores of $x_{i1}$ and $x_{i2}$, respectively. $N_{\mathrm{AL}}$ is the total number of AL pairs, and $\lambda_{\mathrm{AL}}$ is a hyperparameter.

For the CA data, the loss function is defined as:

$$\mathcal{L}_{\mathrm{CA}} = \lambda_{\mathrm{CA}} \frac{1}{N_{\mathrm{CA}}} \sum_{i=1}^{N_{\mathrm{CA}}} (p(y_{i1}) - p(y_{i2}))^2 + \mu \frac{1}{N_{\mathrm{CA}}} \sum_{i=1}^{N_{\mathrm{CA}}} \sum_{u \in \{y_{i1}, y_{i2}\}} (1 - p(u)^2). \quad (3)$$

$y_{i1}$ and $y_{i2}$ are the $i$-th pair of the CA pairs. $N_{\mathrm{CA}}$ is the total number of CA pairs. $\lambda_{\mathrm{CA}}$ and $\mu$ are hyperparameters. The first term makes the scores of the two events closer while the second term prevents the scores from shrinking to zero.

The loss function for the CO data is defined analogously:

$$\mathcal{L}_{\mathrm{CO}} = \lambda_{\mathrm{CO}} \frac{1}{N_{\mathrm{CO}}} \sum_{i=1}^{N_{\mathrm{CO}}} (p(z_{i1}) + p(z_{i2}))^2 + \mu \frac{1}{N_{\mathrm{CO}}} \sum_{i=1}^{N_{\mathrm{CO}}} \sum_{u \in \{z_{i1}, z_{i2}\}} (1 - p(u)^2). \quad (4)$$

The difference is that the first term makes the scores of the two events distant from each other.

## 4 Experiments

### 4.1 Dataset

#### 4.1.1 AL, CA, and CO

As a raw corpus, we used a Japanese web corpus that was compiled through the procedures proposed by Kawahara and Kurohashi (2006). To extract event pairs tagged with discourse relations, we used the Japanese dependency parser KNP[1] and in-house postprocessing scripts (Saito et al., 2018). KNP used hand-written rules to segment each sentence into what we conventionally called *clauses* (mostly consecutive text chunks), each of which contained one main predicate. KNP also identified the discourse relations of event pairs if explicit discourse connectives (Prasad et al., 2008) such as "ので" (*because*) and "のに" (*in spite of*) were present. We treated Cause/Reason (原因・理由) and Condition (条件) in the original tagset (Kawahara et al., 2014) as CAUSE and Concession (逆接)[2] as CONCESSION, respectively. Here is an example of event pair extraction.

[1] http://nlp.ist.i.kyoto-u.ac.jp/EN/index.php?KNP

[2] To be precise, this discourse type is semantically broader than Concession and extends to the area of Contrast.

| Type of pairs | # of pairs |
|---|---|
| AL (Automatically Labeled Pairs) | 1,000,000 |
| CA (CAUSE Pairs) | 5,000,000 |
| CO (CONCESSION Pairs) | 5,000,000 |

Table 1: Statistics of the AL, CA, and CO datasets.

(1) 重大な失敗を犯したので、仕事をクビになった。
Because [I] made a serious mistake, [I] got fired.

From this sentence, we extracted the event pair of "重大な失敗を犯す" ([I] make a serious mistake) and "仕事をクビになる" ([I] get fired), and tagged it with CAUSE.

We constructed our seed lexicon consisting of 15 positive words and 15 negative words, as shown in Section A.1. From the corpus of about 100 million sentences, we obtained 1.4 millions event pairs for AL, 41 millions for CA, and 6 millions for CO. We randomly selected subsets of AL event pairs such that positive and negative latter events were equal in size. We also sampled event pairs for each of CA and CO such that it was five times larger than AL. The results are shown in Table 1.

#### 4.1.2 ACP (ACP Corpus)

We used the latest version[3] of the ACP Corpus (Kaji and Kitsuregawa, 2006) for evaluation. It was used for (semi-)supervised training as well. Extracted from Japanese websites using HTML layouts and linguistic patterns, the dataset covered various genres. For example, the following two sentences were labeled positive and negative, respectively:

(2) 作業が楽だ。
The work is easy.

(3) 駐車場がない。
There is no parking lot.

Although the ACP corpus was originally constructed in the context of sentiment analysis, we found that it could roughly be regarded as a collection of affective events. We parsed each sentence and extracted the last clause in it. The train/dev/test split of the data is shown in Table 2. The objective function for supervised training is:

$$\mathcal{L}_{\mathrm{ACP}} = \frac{1}{N_{\mathrm{ACP}}} \sum_{i=1}^{N_{\mathrm{ACP}}} (R_i - p(v_i))^2, \quad (5)$$

[3]The dataset was obtained from Nobuhiro Kaji via personal communication.

| Dataset | Event polarity | # of events |
|---|---|---|
| Train | Positive | 299,834 |
| | Negative | 300,164 |
| Dev | Positive | 50,118 |
| | Negative | 49,882 |
| Test | Positive | 50,046 |
| | Negative | 49,954 |

Table 2: Details of the ACP dataset.

where $v_i$ is the $i$-th event, $R_i$ is the reference score of $v_i$, and $N_{\mathrm{ACP}}$ is the number of the events of the ACP Corpus.

To optimize the hyperparameters, we used the dev set of the ACP Corpus. For the evaluation, we used the test set of the ACP Corpus. The model output was classified as positive if $p(x) > 0$ and negative if $p(x) \leq 0$.

### 4.2 Model Configurations

As for Encoder, we compared two types of neural networks: BiGRU and BERT. GRU (Cho et al., 2014) is a recurrent neural network sequence encoder. BiGRU reads an input sequence forward and backward and the output is the concatenation of the final forward and backward hidden states.

BERT (Devlin et al., 2019) is a pre-trained multi-layer bidirectional Transformer (Vaswani et al., 2017) encoder. Its output is the final hidden state corresponding to the special classification tag (`[CLS]`). For the details of Encoder, see Sections A.2.

We trained the model with the following four combinations of the datasets: AL, AL+CA+CO (two proposed models), ACP (supervised), and ACP+AL+CA+CO (semi-supervised). The corresponding objective functions were: $\mathcal{L}_{\mathrm{AL}}$, $\mathcal{L}_{\mathrm{AL}} + \mathcal{L}_{\mathrm{CA}} + \mathcal{L}_{\mathrm{CO}}$, $\mathcal{L}_{\mathrm{ACP}}$, and $\mathcal{L}_{\mathrm{ACP}} + \mathcal{L}_{\mathrm{AL}} + \mathcal{L}_{\mathrm{CA}} + \mathcal{L}_{\mathrm{CO}}$.

### 4.3 Results and Discussion

Table 3 shows accuracy. As the Random baseline suggests, positive and negative labels were distributed evenly. The Random+Seed baseline made use of the seed lexicon and output the corresponding label (or the reverse of it for negation) if the event's predicate is in the seed lexicon. We can see that the seed lexicon itself had practically no impact on prediction.

The models in the top block performed considerably better than the random baselines. The performance gaps with their (semi-)supervised counterparts, shown in the middle block, were less than

| Training dataset | Encoder | Acc |
|---|---|---|
| AL | BiGRU | 0.843 |
| | BERT | 0.863 |
| AL+CA+CO | BiGRU | **0.866** |
| | BERT | 0.835 |
| ACP | BiGRU | 0.919 |
| | BERT | **0.933** |
| ACP+AL+CA+CO | BiGRU | 0.917 |
| | BERT | 0.913 |
| Random | | 0.500 |
| Random+Seed | | 0.503 |

Table 3: Performance of various models on the ACP test set.

| Training dataset | Encoder | Acc |
|---|---|---|
| ACP (6K) | BERT | 0.876 |
| +AL | | **0.886** |
| ACP (6K) | BiGRU | 0.830 |
| +AL+CA+CO | | **0.879** |

Table 4: Results for small labeled training data. Given the performance with the full dataset, we show BERT trained only with the AL data.

7%. This demonstrates the effectiveness of discourse relation-based label propagation.

Comparing the model variants, we obtained the highest score with the BiGRU encoder trained with the AL+CA+CO dataset. BERT was competitive but its performance went down if CA and CO were used in addition to AL. We conjecture that BERT was more sensitive to noises found more frequently in CA and CO.

Contrary to our expectations, supervised models (ACP) outperformed semi-supervised models (ACP+AL+CA+CO). This suggests that the training set of 0.6 million events is sufficiently large for training the models. For comparison, we trained the models with a subset (6,000 events) of the ACP dataset. As the results shown in Table 4 demonstrate, our method is effective when labeled data are small.

The result of hyperparameter optimization for the BiGRU encoder was as follows:

$$\lambda_{\mathrm{AL}} = 1,\ \lambda_{\mathrm{CA}} = 0.35,\ \lambda_{\mathrm{CO}} = 1,\ \mu = 0.5.$$

As the CA and CO pairs were equal in size (Table 1), $\lambda_{\mathrm{CA}}$ and $\lambda_{\mathrm{CO}}$ were comparable values. $\lambda_{\mathrm{CA}}$ was about one-third of $\lambda_{\mathrm{CO}}$, and this indicated that the CA pairs were noisier than the CO pairs. A major type of CA pairs that violates our assumption was in the form of "$problem_{\mathrm{negative}}$ causes $solution_{\mathrm{positive}}$":

(4) (悪いところがある, よくなるように努力する)
(there is a bad point, [I] try to improve [it])

The polarities of the two events were reversed in spite of the CAUSE relation, and this lowered the value of $\lambda_{\mathrm{CA}}$.

| Input event | Polarity |
|---|---|
| 道に迷う ([I] get lost) | -0.771 |
| 道に迷わない ([I] don't get lost) | 0.835 |
| 笑う ([I] laugh) | 0.624 |
| 笑われる ([I] am laughed at) | -0.687 |
| 脂肪を落とす ([I] lose body fat) | 0.452 |
| 肩を落とす ([I] feel disappointed) | -0.653 |

Table 5: Examples of polarity scores predicted by the BiGRU model trained with AL+CA+CO.

Some examples of model outputs are shown in Table 5. The first two examples suggest that our model successfully learned negation without explicit supervision. Similarly, the next two examples differ only in voice but the model correctly recognized that they had opposite polarities. The last two examples share the predicate "落とす" (drop) and only the objects are different. The second event "肩を落とす" (lit. drop one's shoulders) is an idiom that expresses a disappointed feeling. The examples demonstrate that our model correctly learned non-compositional expressions.

## 5 Conclusion

In this paper, we proposed to use discourse relations to effectively propagate polarities of affective events from seeds. Experiments show that, even with a minimal amount of supervision, the proposed method performed well.

Although event pairs linked by discourse analysis are shown to be useful, they nevertheless contain noises. Adding linguistically-motivated filtering rules would help improve the performance.

## Acknowledgments

We thank Nobuhiro Kaji for providing the ACP Corpus and Hirokazu Kiyomaru and Yudai Kishimoto for their help in extracting event pairs. This work was partially supported by Yahoo! Japan Corporation.

## References


Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of the seventh edition of the Language Resources and Evaluation Conference*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Haibo Ding and Ellen Riloff. 2018. Weakly supervised induction of affective events by optimizing semantic consistency. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*.

Nobuhiro Kaji and Masaru Kitsuregawa. 2006. Automatic construction of polarity-tagged corpus from html documents. In *Proceedings of the COLING/ACL on Main Conference Poster Sessions*, COLING-ACL '06, pages 452–459, Stroudsburg, PA, USA. Association for Computational Linguistics.

Daisuke Kawahara and Sadao Kurohashi. 2006. Case frame compilation from the web using high-performance computing. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy. European Language Resources Association (ELRA).

Daisuke Kawahara, Yuichiro Machida, Tomohide Shibata, Sadao Kurohashi, Hayato Kobayashi, and Manabu Sassano. 2014. Rapid development of a corpus with discourse annotations using two-stage crowdsourcing. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 269–278, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Lizhen Liu, Donghai Zhang, and Wei Song. 2018. Modeling sentiment association in discourse for humor recognition. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 586–591, Melbourne, Australia. Association for Computational Linguistics.

Jong-Hoon Oh, Kentaro Torisawa, Chikara Hashimoto, Takuya Kawada, Stijn De Saeger, Jun'ichi Kazama, and Yiou Wang. 2012. Why question answering using sentiment analysis and word classes. In *Proceedings of the 2012 Conference on Empirical Methods on Natural Language Processing and Computational Natural Language Learning*.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The Penn discourse TreeBank 2.0. In *Proceedings of the 2012 Conference on Empirical Methods on Natural Language Processing and Computational Natural Language Learning*.

Johan Reitan, Jørgen Faret, Björn Gambäck, and Lars Bungum. 2015. Negation scope detection for twitter sentiment analysis. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*.

Jun Saito, Tomohiro Sakaguchi, Tomohide Shibata, Daisuke Kawahara, and Sadao Kurohashi. 2018. Design and visualization of linguistic information unit based on predicate-argument structure. In *Proceedings of the 24th Annual Meeting of Natural Language Processing (in Japanese).*

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Weiyan Shi and Zhou Yu. 2018. Sentiment adaptive end-to-end dialog systems. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1509–1519, Melbourne, Australia. Association for Computational Linguistics.

Toru Shimizu, Nobuyuki Shimizu, and Hayato Kobayashi. 2018. Pretraining sentiment classifiers with unlabeled dialog data. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 764–770, Melbourne, Australia. Association for Computational Linguistics.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. 2013. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1139–1147, Atlanta, Georgia, USA. PMLR.

Hiroya Takamura, Takashi Inui, and Manabu Okumura. 2005. Extracting semantic orientations of words us-

ing spin model. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 133–140, Stroudsburg, PA, USA. Association for Computational Linguistics.

Peter D. Turney. 2002. Thumbs up or thumbs down?: Semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 417–424, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 347–354, Stroudsburg, PA, USA. Association for Computational Linguistics.

Xiaodan Zhu, Hongyu Guo, Saif Mohammad, and Svetlana Kiritchenko. 2014. An empirical study on the effect of negation words on sentiment. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 304–313, Baltimore, Maryland. Association for Computational Linguistics.

# A Appendices

## A.1 Seed Lexicon

**Positive Words** 喜ぶ (rejoice), 嬉しい (be glad), 楽しい (be pleasant), 幸せ (be happy), 感動 (be impressed), 興奮 (be excited), 懐かしい (feel nostalgic), 好き (like), 尊敬 (respect), 安心 (be relieved), 感心 (admire), 落ち着く (be calm), 満足 (be satisfied), 癒される (be healed), and スッキリ (be refreshed).

**Negative Words** 怒る (get angry), 悲しい (be sad), 寂しい (be lonely), 怖い (be scared), 不安 (feel anxious), 恥ずかしい (be embarrassed), 嫌 (hate), 落ち込む (feel down), 退屈 (be bored), 絶望 (feel hopeless), 辛い (have a hard time), 困る (have trouble), 憂鬱 (be depressed), 心配 (be worried), and 情けない (be sorry).

## A.2 Settings of Encoder

**BiGRU** The dimension of the embedding layer was 256. The embedding layer was initialized with the word embeddings pretrained using the Web corpus. The input sentences were segmented into words by the morphological analyzer Juman++.[4] The vocabulary size was 100,000. The number of hidden layers was 2. The dimension of hidden units was 256. The optimizer was Momentum SGD (Sutskever et al., 2013). The mini-batch size was 1024. We ran 100 epochs and selected the snapshot that achieved the highest score for the dev set.

**BERT** We used a Japanese BERT model[5] pretrained with Japanese Wikipedia. The input sentences were segmented into words by Juman++, and words were broken into subwords by applying BPE (Sennrich et al., 2016). The vocabulary size was 32,000. The maximum length of an input sequence was 128. The number of hidden layers was 12. The dimension of hidden units was 768. The number of self-attention heads was 12. The optimizer was Adam (Kingma and Ba, 2014). The mini-batch size was 32. We ran 1 epoch.

[4] http://nlp.ist.i.kyoto-u.ac.jp/EN/index.php?JUMAN++

[5] http://nlp.ist.i.kyoto-u.ac.jp/index.php?BERT%E6%97%A5%E6%9C%AC%E8%AA%9EPretrained%E3%83%A2%E3%83%87%E3%83%AB (in Japanese)

# Event Detection with Multi-Order Graph Convolution and Aggregated Attention

**Haoran Yan[†‡], Xiaolong Jin[†‡], Xiangbin Meng[§], Jiafeng Guo[†‡] and Xueqi Cheng[†‡]**
[†]CAS Key Laboratory of Network Data Science and Technology,
Institute of Computing Technology, Chinese Academy of Sciences (CAS);
[‡]School of Computer and Control Engineering, The University of CAS
{yanhaoran17s, jinxiaolong, guojiafeng, cxq}@ict.ac.cn
[§]School of Computer Science, Beijing University of Posts and Telecommunications
mengxb@bupt.edu.cn

## Abstract

Syntactic relations are broadly used in many NLP tasks. For event detection, syntactic relation representations based on dependency tree can better capture the interrelations between candidate trigger words and related entities than sentence representations. But, existing studies only use first-order syntactic relations (i.e., the arcs) in dependency trees to identify trigger words. For this reason, this paper proposes a new method for event detection, which uses a dependency tree based graph convolution network with aggregative attention to explicitly model and aggregate multi-order syntactic representations in sentences. Experimental comparison with state-of-the-art baselines shows the superiority of the proposed method.

## 1 Introduction

As an important information extraction task, event detection aims to find event mentions with specific event types from given texts. Each event mention is identified by a word or phrase called event trigger, which serve as the main word(s) to the corresponding event. For example, in the sentence presented in Figure 1, an event detection system should be able to recognize the trigger word "fired" corresponding to the event type "Attack".

Among existing methods for event detection, sequence based ones (e.g., Chen et al. (2015); Nguyen et al. (2016)) only use the given sentences, which suffer from the low efficiency problem in capturing long-range dependency; On the contrary, dependency tree based methods utilize the syntactic relations (i.e., arcs) in the dependency tree of a given sentence to more effectively capture the interrelation between each candidate trigger word and its related entities or other triggers. For example, Nguyen and Grishman (2018) and Liu et al. (2018) treat each dependency tree as a graph and adopt Graph Convolution Network (GCN) (Kipf and Welling, 2016) to extract event triggers.

The syntactic relations between trigger words and related entities may be of first order, embodied as the direct arcs in the dependency tree. They may also be of high-order (i.e., the paths more than one hop in the dependency tree). Particularly, according to our statistics on the benchmark ACE-2005 dataset, about 51% (4977/9793) of event related entities need more than one hop to get to the corresponding trigger words and related entities in their dependency trees. For example, in Figure 1, it needs at least 4 hops (i.e., "fired"–"evidence"–"blood"–"soldiers") to figure out that the word "fired" means "shot" instead of "dismissed".

However, the above dependency tree based methods explicitly use only first-order syntactic relations, although they may also implicitly capture high-order syntactic relations by stacking more GCN layers. However, as the number of GCN layers increases, the representations of neighboring words in the dependency tree will get more and more similar, since they all are calculated via those of their neighbors in the dependency tree. This is the so-called over-smoothing problem (Zhou et al., 2018), which damages the diversity of the representations of neighboring words.

To overcome this problem, this paper proposes a Multi-Order Graph Attention Network based method for Event Detection, called MOGANED. It utilizes both first-order syntactic graph and high-order syntactic graphs to explicitly model multi-order representations of candidate trigger words. In order to calculate multi-order representations for each word, we apply Graph Attention Network (GAT) proposed in (Veličković et al., 2017) to weight the importance of its neighboring words in the syntactic graphs of different orders. We then employ an attention aggregation mechanism to merge its multi-order representations. Finally,



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5766–5770,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

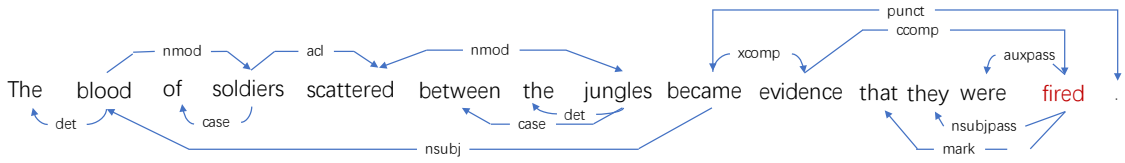


Figure 1: An example sentence with the dependency parsing result.

through experimental comparison with state-of-the-art baselines we show the superiority of the proposed method in terms of both precision and F1-measure. It should be mentioned that, to the best of our knowledge, this is the first study applying GAT to event detection.

## 2 The Proposed Model

Following the existing studies, we regard event detection as a multi-class classification problem. Let $W = w_1, w_2, ..., w_n$ be a sentence of length $n$, where $w_i$ is its $i$-th token. Since event triggers may contain multiple words, we adopt the "BIO" schema to make annotation. The number of labels is thus be $2L + 1$, where $L$ is the number of predefined event types.

The proposed model contains three modules: (i) word encoding, which encodes input sentence to a sequence of vectors; (ii) multi-order graph attention network that performs graph attention convolution over multi-order syntactic graphs; (iii) attention aggregation, which aggregates multi-order representations of each word with different attention weights to predict its label.

### 2.1 Word Encoding

The word encoding module first transforms each input token $w_i$ into a comprehensive embedding vector $x_i$ by concatenating its word embedding $word_i$, entity type embedding $et_i$, POS-tagging embedding $pos_i$ and position embedding $ps_i$ (Nguyen et al., 2016; Nguyen and Grishman, 2018), where $word_i$ is obtained by looking up a pre-trained word embedding table on a large corpus and others are randomly initialized.

The input sentence $W$ will then be transformed to a sequence of vectors $X = x_1, x_2, ..., x_n$. Since each word can only be updated by its neighbors in the dependency tree through dependency arcs, following the previous methods (Liu et al., 2018), we employ a Bidirectional Long-Short Term Memory network (BiLSTM) (Hochreiter and Schmidhuber, 1997) to encode $X$ with its context as $P = p_1, p_2, ..., p_n$, which will be used as the input of the multi-order GAT. Here,

$$p_i = [\overrightarrow{LSTM}(x_i)||\overleftarrow{LSTM}(x_i)], \quad (1)$$

where $||$ is the concatenation operation.

### 2.2 Multi-Order Graph Attention Network

Each dependency tree can be represented as a first-order syntactic graph in terms of an adjacency matrix. Let $A$ be the adjacency matrix of the first-order syntactic graph, which is generated from the dependency tree of the sentence, $W$. $A$ contains three sub matrixes $A_{along}$, $A_{rev}$ and $A_{loop}$ (Marcheggiani and Titov, 2017) of the same dimensions $n \times n$. $A_{along}(i, j) = 1$, if there is a dependency arc from $w_i$ to $w_j$ in the dependency tree, otherwise 0. As the reverse graph of $A_{along}$, $A_{rev} = A^T_{along}$. $A_{loop}$ is an identity matrix.

The adjacency matrix of the $k$-th-order syntactic graph is $A^k_{subg} = (A_{subg})^k$, where $subg \in \{along, rev, loop\}$. $A^k_{subg}$ records all k-hop paths of $A_{subg}$. Note that $A^k_{loop} = A_{loop}$, since it is an identity matrix. For simplicity, in what follows we use $a^k$, $b^k$, $c^k$ to denote $A^k_{along}$, $A^k_{rev}$ and $A^k_{loop}$, respectively.

The multi-order GAT module simply learns a list of representations over multi-order syntactic graphs by a few parallel GAT layers (Veličković et al., 2017), which weights the importance of neighbors of each word in each syntactic graph during convolution.

The representations $h^k_i$ of the $k$-th-order syntactic graph $A^k$ are calculated from the representations of the subgraphs of $A^k$:

$$h^k_i = f(p_i, a^k) \oplus f(p_i, b^k) \oplus f(p_i, c^k), \quad (2)$$

where $f(\cdot)$ is the graph attention convolution function and $\oplus$ is element-wise addition, and

$$f(p_i, a^k) = \sigma\left(\sum_{j=1}^{n}(u_{ij}a^k_{ij}(W_{a,k}p_j + \epsilon_{a,k})\right), \quad (3)$$

where $\sigma$ is exponential linear unit (ELU) (Clevert et al., 2015); $W_{a,k}$ and $\epsilon_{a,k}$ are the weight matrix and bias item for $a^k$; $u_{ij}$ is the normalized weight of the neighbor $w_j$ when updating $w_i$,

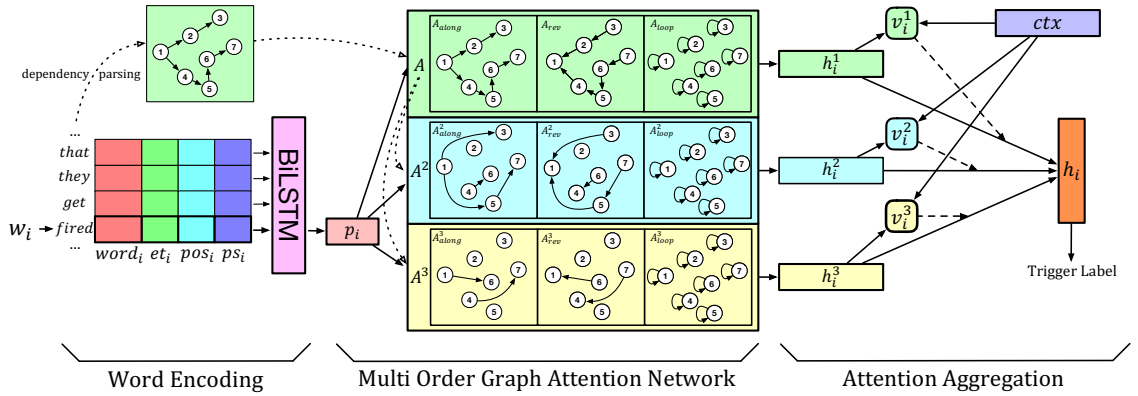


Figure 2: The illustrative architecture of the proposed MOGANED method, where $K$=3.

$$u_{ij} = softmax(e_{ij}) = \frac{exp(e_{ij})}{\sum_{j \in \mathcal{N}_i} exp(e_{ij})}, \quad (4)$$

where $e_{ij} = \gamma(W_{comb}[W_{att}p_i||W_{att}p_j])$. Here, $\mathcal{N}_i$ is the neighbor set of $w_i$ in the subgraph, $\gamma$ is LeakyReLU (with negative input slope $\alpha$ = 0.2) (Maas et al., 2013); $W_{comb}$ and $W_{att}$ are weight matrices.

After graph attention convolution, each candidate trigger word $w_i$ gets a list of multi-order representations $h_i^k, k \in [1, K]$, where $K$ is the highest order used in this module.

### 2.3 Attention Aggregation

To aggregate the list of multi-order representations $h_i^k$ of each word $w_i$, we employ a weighted attention aggregation mechanism from Hierarchical Attention Networks (Yang et al., 2016):

$$h_i = \sum_{k=1}^{K} v_i^k h_i^k, \quad (5)$$

where $v_i^k$ is the normalized weight of the $k$-th order graph representation of the word $w_i$, which is calculated by

$$v_i^k = softmax(s_i^k) = \frac{exp((s_i^k)^T ctx)}{\sum_{j=1}^{K} exp((s_i^j)^T ctx)}, \quad (6)$$

where $s_i^j = tanh(W_{awa}h_i^j + \epsilon_{awa})$. Here, $W_{awa}$ and $\epsilon_{awa}$ are the weight matrix and the bias term, respectively; $ctx$ is a randomly initialized context vector, which captures the importance of graph representations of each order.

Finally, we use the aggregated representation $h_i$ to predict the trigger label of word $w_i$ as follows:

$$y_i^t = softmax(O_i^t) = \frac{exp(O_i^t)}{\sum_{q=1}^{2L+1} exp(O_i^q)}, \quad (7)$$

where $y_i^q$ denotes the probability assigning label $q$ to word $w_i$ and $O_i = w_o h_i + \epsilon_o$. Here, $w_o$ and $\epsilon_o$ are the weight matrix and the bias item, respectively.

### 2.4 Bias Loss Function

Since the number of "O" labels in the data is much larger than that of event labels, we use a bias loss function (Chen et al., 2018) to enhance the influence of event labels during training:

$$J(\theta) = -\sum_{i=1}^{N_s}\sum_{j=1}^{N_{i,w}} (\log p(y_j^t|s_i,\theta) \cdot I(O) + \lambda \log p(y_j^t|s_i,\theta) \cdot (1 - I(O))), \quad (8)$$

where $N_s$ is the number of sentences; $N_{i,w}$ is the number of words in $s_i$; $I(O)$ equals 1, if the label of the word is "O"; otherwise 0; $\lambda$ is the weight parameter larger than 1.

## 3 Experiments

### 3.1 Experiment Settings

**Dataset and Resources** We compare our MOGANED model with baseline methods on ACE-2005 with the same data split, where 40 newswire documents are used as the test set, 30 newswire documents as the validation set and 529 remained documents as the training set.

For data preprocessing, we use the Stanford CoreNLP toolkit for sentence splitting, tokenizing, POS-tagging and dependency parsing. We adopt word embeddings trained over the New York Times corpus with the Skip-gram algorithm (Mikolov et al., 2013; Chen et al., 2018).

**Hyper Parameter Setting** The hyper parameters are tuned on the validation set. For all experiments

| Method | $P$ | $R$ | $F_1$ |
|---|---|---|---|
| Cross Event | 68.7 | 68.9 | 68.8 |
| DMCNN | 75.6 | 63.6 | 69.1 |
| JRNN | 66.0 | 73.9 | 69.3 |
| DEEB-RNN | 72.3 | **75.8** | 74.0 |
| dbRNN† | 74.1 | 69.8 | 71.9 |
| GCN-ED† | 77.9 | 68.8 | 73.1 |
| JMEE† | 76.3 | 71.3 | 73.7 |
| MOGANED† | **79.5** | 72.3 | **75.7** |

Table 1: Overall performance of different methods on the test set with gold-standard entities. † indicates that the method uses dependency arcs.

below, we use 100 as the dimension of word embeddings and 50 as that of the rest embeddings. We set the hidden units of the BiLSTM network to 100. We set the highest order $K$ to 3 and the dimension of graph representation to 150. We set batch size to 30 and utilize a fixed maximum sentence length $n = 50$ by padding short sentences and cutting longer ones. During training, we use the AdaDelta update rule (Zeiler, 2012) with a learning rate of 0.001. We set the dropout rate to 0.3 and the L2-norm to $1e-5$. We set the bias loss parameter $\lambda$ to 5.

### 3.2 Overall Performance

We compare our model with the following state-of-the-art baselines: 1) Cross Event (Liao and Grishman, 2010), which uses document level information for event extraction; 2) DMCNN (Chen et al., 2015), which builds a dynamic multi-pooling CNN model; 3) JRNN (Nguyen et al., 2016), which uses a bidirectional RNN and human designed features; 4) DEEB-RNN (Zhao et al., 2018), which uses hierarchical supervised attention with document level information for event detection; 5) dbRNN (Sha et al., 2018), which adds dependency arcs over a Bi-LSTM network to improve event extraction; 6) GCN-ED (Nguyen and Grishman, 2018), which uses an argument pooling mechanism for event detection based on GCN; 7) JMEE (Liu et al., 2018), which uses GCN with highway network and self-attention.

Table 1 presents the performance comparison between different methods. We can see that MOGANED achieves 1.6% and 1.7% improvement on precision and $F_1$-measure, respectively, compared with the best baselines. MOGANED reaches a lower recall than two sequence based methods, JRNN and DEEB-RNN. This may be caused by the propagated error from the dependency parsing tool. There are long sentences where trigger words and their corresponding entities are parsed into two independent dependency trees without connection. Therefore, the trigger words and their related entities are unable to interact in multi-order graph attention network, which leads to a lower recall. However, MOGANED still achieves the best performance in terms of precision, recall, $F_1$-measure among all dependency based methods, which suggests the effective of multi-order representations.

| Method | $P$ | $R$ | $F_1$ |
|---|---|---|---|
| MOGANED-First | 71.0 | 69.9 | 70.4 |
| MOGANED-GCN | 75.7 | 71.1 | 73.3 |
| MOGANED-Mean | 77.3 | 70.9 | 74.0 |
| MOGANED | **79.5** | **72.3** | **75.7** |

Table 2: Performance of modified architectures based on MOGANED.

### 3.3 Ablation Study

This study aims to validate the impacts of multi-order representation, graph attention network and attention aggregation. For this purpose, we design three architectures based on MOGANED: 1) MOGANED-First: it only uses first-order syntactic graph (i.e., $K$=1); 2) MOGANED-GCN: it uses traditional GCN instead of GAT; 3) MOGANED-Mean: it adopts mean pooling as the attention aggregation mechanism of multi-order representations for each word.

The experimental result is shown in Table 2. All of these three modified models get lower performance than MOGANED. MOGANED-First achieves the worst performance, which suggests that high-order syntactic relations play an important role in event detection. MOGANED-GCN drops more on precision than recall, which illustrates that the attention learned from GAT helps MOGANED predict trigger words more precisely. The performance drop of MOGANED-Mean is the smallest among the three modified models. Although the average of multi-order representations achieves competitive performance for event detection, the proposed attention aggregation module still distinguishes the importance of syntactic representations of different order, which achieves 1.7% improvement on $F_1$-measure.

## 4 Conclusion

In this paper, we proposed the MOGANED model for modeling multi-order representations via GAT and employed an attention aggregation mechanism to better capture dependency contextual information for event detection. The experimental results well demonstrate its effectiveness and superiority.

## Acknowledgments

We are grateful to Bingbing Xu, Saiping Guan, Shaobo Liu, Zixuan Li, Long Bai and Yunqi Qiu for valuable discussion. We also appreciate anonymous reviewers for constructive review comments.

This work is supported by National Key Research and Development Program of China under grant 2016YFB1000902, National Natural Science Foundation of China under grants 61772501, 61572473, 61572469 and 91646120, and the GF Innovative Research Program.

## References

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 167–176.

Yubo Chen, Hang Yang, Kang Liu, Jun Zhao, and Yantao Jia. 2018. Collective event detection via a hierarchical and bias tagging networks with gated multi-level attention mechanisms. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1267–1276.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2015. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Shasha Liao and Ralph Grishman. 2010. Using document level cross-event inference to improve event extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 789–797. Association for Computational Linguistics.

Xiao Liu, Zhunchen Luo, and Heyan Huang. 2018. Jointly multiple events extraction via attention-based graph information aggregation. *arXiv preprint arXiv:1809.09078*.

Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. *arXiv preprint arXiv:1703.04826*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 300–309.

Thien Huu Nguyen and Ralph Grishman. 2018. Graph convolutional networks with argument-aware pooling for event detection. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Lei Sha, Feng Qian, Baobao Chang, and Zhifang Sui. 2018. Jointly extracting event triggers and arguments by dependency-bridge rnn and tensor-based argument interaction. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Yue Zhao, Xiaolong Jin, Yuanzhuo Wang, and Xueqi Cheng. 2018. Document embedding enhanced event detection with hierarchical and supervised attention. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 414–419.

Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. 2018. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*.

# Coverage of Information Extraction from Sentences and Paragraphs

**Simon Razniewski**[1], **Nitisha Jain**[2], **Paramita Mirza**[1], **Gerhard Weikum**[1]

[1] Max Planck Institute for Informatics
[2] Hasso Plattner Institute

{srazniew, paramita, weikum}@mpi-inf.mpg.de
nitisha.jain@hpi.de

## Abstract

Scalar implicatures are language features that imply the negation of stronger statements, e.g., *"She was married twice"* typically implicates that she was not married thrice. In this paper we discuss the importance of scalar implicatures in the context of textual information extraction. We investigate how textual features can be used to predict whether a given text segment mentions all objects standing in a certain relationship with a certain subject. Preliminary results on Wikipedia indicate that this prediction is feasible, and yields informative assessments.

## 1 Introduction

Following the cooperative principle, natural language utterances can implicate a range of assertions that are not explicitly stated (Grice, 1975). One specific class of implicatures are scalar implicatures, which concern the negation of stronger statements (Carston, 1998). Scalar implicatures are derived from Grice's maxim of quantity - that speakers would make stronger statements if possible, therefore, negation can be deduced if these are not made.

Yet the maxim of quantity interacts with the maxim of relevance, i.e., what is implicated depends on what is relevant in a given context. Consider the examples in Figure 1. From the first sentence, typical for biographical descriptions, most humans would draw the implicature that Obama has no other children. For the second sentence about Jolie, this implicature would typically not be drawn - she might well have other children that are too young or too old to be brought to school, but are not relevant in the school context.

The interaction between the maxims of quantity and relevance has implications for textual information extraction (IE). Textual IE usually produces (ideally canonicalized) subject-predicate-object triples, and annotates them with estimates of *correctness* (also called accuracy, precision or confidence), e.g., a 93% belief that Malia is really Obama's child. In contrast, IE usually lacks such an ability for *coverage* (also called recall or completeness). It is not able to estimate whether its extractions represent all facts pertaining to a certain topic, e.g., all children of Jolie.

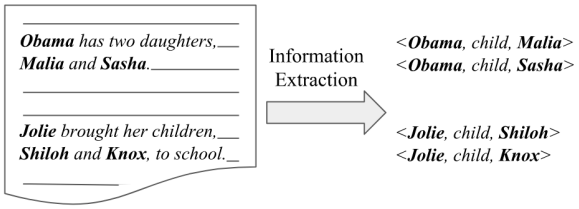


Figure 1: Two sentences with different coverage.

**Importance of coverage-awareness.** Coverage-awareness of IE is a crucial and highly desirable property for several downstream use cases. (i) Today's *question answering* systems are well geared for questions where exactly one answer should be returned (e.g., quiz questions or reading-comprehension tasks) (Fader et al., 2014; Yang et al., 2015). In contrast, for questions with sets of answers, QA systems often merely yield subsets, and do not inform the user about that. Similarly, they struggle with questions that have no answer, too often still returning a best-effort answer even if it is incorrect. Coverage awareness would enable a better treatment of both cases. (ii) Guiding editors in how to *prioritize curation* efforts is a key issue for collaboratively built and maintained knowledge bases such as Wikidata (Balaraman et al., 2018). Yet, methods to automatically identify incomplete parts are largely based on aggregate-level statistics, and do not consider entity-specific textual context (Razniewski et al., 2017; Galárraga et al., 2017). (iii) Coverage awareness would also be useful for *automated knowledge base construc-*



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5771–5776, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

*tion* techniques, either to dynamically adjust confidence thresholds, i.e., lowering thresholds in case of low coverage, and increasing thresholds in case of too many extractions, or to reallocate search budgets to low-coverage regions, while stopping the exploration of complete areas (Ipeirotis et al., 2007; Jain et al., 2008).

**Contribution and Approach.** In this paper we analyze the viability of text coverage prediction. We use textual features to estimate whether a text segment contains all objects for a given subject-predicate pair, e.g., whether a given text mentions all children of Jolie. For an experimental study, we use fact counts for 5 Wikidata relations as ground truth. Using these counts, we train and evaluate on Wikipedia-extracted sentences and paragraphs, finding that coverage prediction is generally feasible and yields informative assessments.

Our conceptual contributions are:

- We introduce and define the novel problem of textual coverage prediction, and we discuss its key features.
- We present a method, along with experimental results that demonstrate its practical value.

Our experiments confirm that coverage estimation is possible, and yield the following technical insights:

- Features: Unigrams and bigrams provide informative cues towards coverage estimation.
- Scope: Coverage estimation is feasible for diverse domains ranging from family relations to organizational membership, and both on the level of sentences and paragraphs.

## 2 Background

Information extraction from text sources has been greatly advanced over the past two decades; see e.g., (Agichtein and Gravano, 2000; Etzioni et al., 2004; Suchanek et al., 2009; Mintz et al., 2009; Riedel et al., 2013; Dong et al., 2014; Shin et al., 2015; Mausam, 2016; Chiticariu et al., 2018; Stanovsky et al., 2018). The underlying methodologies span regular-expression matching, rule-based extraction, conditional random fields, constraint reasoning, all the way to deep learning. Depending on the task at hand, IE often achieves high correctness (sometimes above 90%). However, evaluating its coverage is inherently hard, as this would require exhaustively annotated corpora as gold standard. As a consequence, assessing and optimizing coverage has typically been an afterthought at best, and is usually completely disregarded.

In contrast, coverage (recall) is one of the key metrics in information retrieval (IR), i.e., in search applications. Here, recall is measured in terms of retrieving a large fraction of the relevant documents or passages, where relevance is stated by gold-standard annotations. In the context of entire IE workflows (e.g., for text analytics over business news), the prior works of Ipeirotis et al. (2007) and Jain et al. (2008) have considered optimizations for recall. However, this solely refers to the search-centric parts of such workflows, that is, the document or passage sets that are then fed into IE steps.

Grice's maxims of cooperative communication (Grice, 1975) introduce the concept of implicatures, which are conclusions that humans draw even though texts do not literally support them. The implicatures of interest here are scalar implicatures, i.e., the conclusion that no more facts are true than those explicitly stated (Carston, 1998). Scalar implicatures are closely connected to the closed-world assumption in logics, where statements are assumed to be false, unless explicitly stated. Yet, as exemplified in in Figure 1, due to the maxim of relevance, the scope of scalar implicatures may vary significantly.

Closest to coverage-awareness is recent work on counting quantifier extraction (Mirza et al., 2018). There, relation counts are extracted from phrases such as *"Jolie has six children"*, which, in a second step, are compared against fact counts in an existing KB. In contrast, the present work aims to directly predict the coverage of text segments.

## 3 Problem and Approach

While information extraction is a noisy process with both false positives and false negatives, our focus here is on whether, in principle, a text segment allows the extraction of all facts that hold in reality. For this purpose, we assume we have perfect knowledge of *all real-world facts* for the objects that are connected to a specific subject $s$ and property $p$; we denote this object set as $RW\{o \mid sp\}$. Now assume an educated and linguistically versed human is presented with a text segment $t$ and the task of telling which objects $o$ she would assign to a fixed subject $s$ and property $p$ given solely the text $t$. We denote this ground-

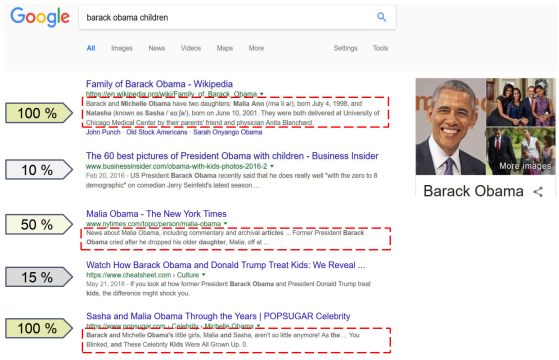


Figure 2: Possible application of a coverage classifier on web search snippets.

truth extraction as $GTE\{o \mid sp, t\}$.

**Text Coverage Prediction Problem.** Given a text segment $t$ for a fixed subject $s$ and property $p$, predict whether the human ground-truth from reading $t$ matches the real-world facts:

$$GTE\{o \mid sp, t\} = RW\{o \mid sp\}.$$

This problem is different from assessing the quality of specific IE methods and tools. Since there are no perfect IE tools, considering the extractions from an IE tool would confound two distinct issues: 1) whether a text segment contains all information of interest (our present problem), and 2) what the recall of the specific IE tool is (a standard evaluation criterion for IE methods). Although our automated evaluation below necessarily builds on concrete choices for IE methods, our emphasis is on the fundamental problem of recall assessment given solely a text segment, as described above.

By casting the problem into a binary classification task, we look only at two cases: a) $GTE\{o \mid sp, t\}$ contains all real-world facts (*complete*), and b) it does not (*incomplete*). This formulation disregards complex graded cases, such as $GTE\{o \mid sp, t\}$ containing at least 70% of the real-world facts. Nevertheless, the problem naturally invites the use of scores that are confidences/probabilities. For example, for the first sentence in Figure 1, the probability to contain all $\langle Obama, child, *\rangle$-facts might be 0.9, while for the second sentence, the probability to contain all $\langle Jolie, child, *\rangle$ facts might be 0.4. An illustration of how such confidence scores could be applied to Web search snippets is shown in Figure 2.

**Methods and baselines.** We approach the problem as classification task for $t$. As state-of-the-art text classification methods, we use interpretable feature-based Support Vector Machines (SVMs) with unigrams and bigrams from $t$ as input, and Long Short-Term Memory networks (LSTMs) (Tai et al., 2015). LSTMs are used to encode sentences/paragraphs, using word representations (d=100) that are learned from scratch (initialized uniformly). One hidden layer of size 256 with ReLU activation and an output layer with sigmoid activation are used for binary classification. We employ the Adam optimizer with default parameter values. The models were trained for 20 epochs. We also experimented with using pretrained (word2vec) embeddings, but found no improvement, possibly due to the unorthodox nature of the problem, where typical word semantics as relevant for classic NLP tasks like QA, translation etc. do not help.

In web extraction scenarios one could additionally also consider features such as subject popularity, the relative position of a text segment, or web-source reliability scores.

We employ three baselines. Two natural baselines are *length* and *#pnames*, which classify the longest text segments (by character length) or the text segments containing most proper names as complete, i.e., assume that the more information, the better. A lower bound is given by a third baseline, *random*, which simply tosses a coin to decide whether a text segment is classified as complete, or not. For all baselines, the classification thresholds/coin bias is chosen so that input class distributions are maintained, therefore their precision and recall coincide.

## 4 Experimental Setup

**Predicates.** We perform experiments for 5 Wikidata predicates that span three different domains: (i) family relations: *child* (P40), *spouse* (P26), (ii) education and work: *educatedAt* (P69), *employer* (P108), (iii) band compositions: via *hasPart* (P527) for instances of the *musical ensemble* (Q2088357) class.

**Approximating ground-truth extractions.** Due to the intellectual complexity of fact extraction, crowdsourcing annotations faces scalability challenges, particularly at the paragraph level. We therefore opt for approximating the ideal human extractions $GTE\{o \mid sp, t\}$ via the combination of open information extraction, predicate paraphrase matching and object label matching. Note that this specific choice is not decisive for our approach

and merely serves as a concrete scalable instantiation of our framework, human labels or other automated IE methods could be plugged in as well.

To evaluate whether a text segment contains an $\langle s, p, o \rangle$-fact, we rely on the open information extraction system OpenIE 4 (Pal and Mausam, 2016), the PATTY predicate paraphrase dictionary (Nakashole et al., 2012), and Wikidata entity alias names.

For example, suppose we are interested in extracting facts for the *child* property for the subject *Angelina Jolie*. OpenIE extracts the triple $\langle \underline{Maddy}, is\ first\ adopted\ \underline{son}\ [of], Jolie \rangle$ from the text segment *"Jolie's first adopted son is Maddy."* As *(i) Maddy* is one of the aliases of *Angelina Jolie's* child *Maddox Chivan* in IMDb, and *(ii) son* appears in the list of paraphrases for the *child* predicate, we consider that the text segment contains the fact $\langle Angelina\ Jolie, child, Maddox\ Chivan \rangle$.

**Labelled data.** We use distant supervision to automatically label data. Assuming that Wikidata's coverage is near-perfect for popular entities, for each of the predicates we collect the 1000-8000 most popular subjects in Wikidata, along with their facts for the respective property (see Table 1).[1] As many properties have a skew towards low frequencies, which may make completeness prediction trivial, we only considered subjects having at least two objects in Wikidata (#subj w/ $\geq$2 obj). We collected two granularities of text units, *sentences* and *paragraphs* that contain at least one object, as found on the Wikipedia pages of the respective subjects. To ensure that general features are learned, we mask proper names and specific numbers with generic placeholders. A text segment is labelled *complete*, if it contains, for each object listed on Wikidata, at least one first names or alias. It is labelled as *incomplete* otherwise. In total, we obtain about 300 complete and 2000 incomplete sentences and paragraphs per relation, which we split into 80% for training and 20% for testing.

## 5 Results and Discussion

Table 2 shows the precision, recall and F1-score in terms of identifying complete text segments. Both

[1]While using Wikidata as source for labels for distant supervision of Wikipedia texts may seem circuitous, we note that unlike Wikipedia, Wikidata is language-independent, thus, has potential for much higher coverage especially for entities more famous outside English-speaking countries.

| | *child* | *spouse* | *member* | *employer* | *educatedAt* |
|---|---|---|---|---|---|
| #subj | 40,145 | 45,261 | 8,901 | 58,731 | 273,128 |
| #subj w/ $\geq$2 obj | 15,022 | 4,055 | 1,022 | 12,885 | 72,847 |
| seeds | 1,000 | 1,000 | 1,000 | 1,000 | 8,000 |
| Sentences +/- | 135/2,050 | 119/2,444 | 672/10,358 | 47/1,499 | 447/ 2,603 |
| Paragraphs +/- | 217/1,595 | 385/2,044 | 930/ 5,362 | 108/1,248 | 339/ 2,384 |

Table 1: Number of Wikidata subjects and derived labelled text segments. +/- signifies *complete/incomplete* w.r.t. Wikidata.

| Text unit | Model | *child* P | R | F | *spouse* P | R | F | *hasPart* P | R | F | *employer* P | R | F | *educatedAt* P | R | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Sentence | *random* | | | .06 | | | .04 | | | .06 | | | 0 | | | .14 |
| | *length* | | | .05 | | | .28 | | | .13 | | | 0 | | | .24 |
| | *#pnames* | | | .05 | | | .22 | | | .17 | | | 0 | | | .28 |
| | SVM | .50 | **.42** | **.46** | .35 | .33 | .34 | .39 | .23 | .28 | **.17** | **.13** | **.14** | **.55** | .39 | .46 |
| | LSTM | **.64** | .39 | .45 | **.58** | **.39** | **.47** | **.54** | **.69** | **.60** | 0 | 0 | 0 | .47 | **.98** | **.64** |
| Paragraph | *random* | | | .12 | | | .16 | | | .15 | | | .08 | | | .12 |
| | *length* | | | .17 | | | .37 | | | .26 | | | .10 | | | .21 |
| | *#pnames* | | | .19 | | | .40 | | | .31 | | | .20 | | | .29 |
| | SVM | **.50** | .44 | .47 | **.74** | .73 | **.73** | **.59** | .53 | **.56** | **.50** | .05 | .09 | **.70** | .57 | **.63** |
| | LSTM | .41 | **.83** | **.55** | .54 | **1** | .70 | .50 | **.59** | .54 | .34 | **.75** | **.47** | .42 | **.96** | .58 |

Table 2: Performance of coverage prediction.

SVMs and LSTMs outperform the baselines by a considerable margin, performing slightly better at paragraph than at sentence level. They perform best there on the *spouse* property (.73/.70 F1), followed by *educated at* (.63-.58 F1). They comparably fail at sentence level on *employer* (.14/0 F1), presumably because it is rather rare that all employments are listed in the same sentence. For instance, it is more common to find *"He served as a professor at* [*University A*]*, and also held an appointment at* [*University B*]*. In July 2007, he left* [*University A*] *and joined the faculty of* [*University C*]*. He was also a visiting professor at the* [*University D*]*."* as a complete paragraph.

In Table 3 we show the most informative bigrams for the n-gram-based SVMs on the paragraph level, for predicates having reasonably good F1 scores. Most of the highly weighted bigrams signal the beginning of name listing, such as *daughters* $\langle pname \rangle$, *married twice*, *featuring lineup* and *attended* $\langle pname \rangle$. Some bigrams convey temporal information, such as *later married*, *briefly attended* and *left graduating*, which indicate that the paragraph contains a narrative that lists object names for different time periods. This was particularly true for the *spouse*, *employer* and *educatedAt* predicates, for which usually only one object is valid at each timepoint.

We also find various terms indicating incompleteness, for instance *surviving* ("Had 5 children, but only Mary and Bob were surviving to adulthood"), *succeeded* ("She was later succeeded by her son James in her role as ..."), *addition* ("In addition, a daughter, Susan, was born in ...") among

| *child* | *spouse* | *hasPart* | *educatedAt* |
|---|---|---|---|
| ⟨*pname*⟩ *sons* | *married twice* | *featuring lineup* | *educated* ⟨*pname*⟩ |
| *grandsons* ⟨*pname*⟩ | *children* ⟨*pname*⟩ | *lineup* ⟨*pname*⟩ | *briefly attended* |
| ⟨*num*⟩ *grandsons* | *second marriage* | *consists* ⟨*pname*⟩ | *attended* ⟨*pname*⟩ |
| *daughters* ⟨*pname*⟩ | ⟨*num*⟩ *son* | ⟨*num*⟩ *tour* | *left graduating* |
| *sons* ⟨*pname*⟩ | *later married* | *vocals* ⟨*propname*⟩ | ⟨*pname*⟩ *left* |

Table 3: Selected important paragraph-level bigrams indicating completeness for SVMs.

| Sentence | LSTM score |
|---|---|
| He was the father of actor Pierre Renoir (1885-1952), filmmaker Jean Renoir (1894-1979) and ceramic artist Claude Renoir (1901-1969). | 0.54 |
| His daughter Julie Gavras and his son Romain Gavras are also filmmakers. | 0.46 |
| Genghis Khan was aware of the friction between his sons (particularly between Chagatai and Jochi) and worried of possible conflict between them if he died. | 0.42 |
| "From this moment I am no longer the king; the king is Victor my son." | 0.17 |

Table 4: Example LSTM predictions at sentence-level.

unigrams indicating incompleteness of *child* at sentence level, or *leaving* and *previously* for *employer*.

While at paragraph level, there is no clear winner, at sentence level LSTMs outperform SVMs. Possibly, this is due to latent representations being more important when features are sparse (i.e., text segments are short). Anecdotal LSTM sentence-level predictions are shown in Table 4. The full input and resulting predictions will be made available on Github.

## 6 Discussion

**Task Difficulty.** The prediction results, ranging in F1-score from 0 to .64 for the sentence level and .09 to .73 for the paragraph level, are significantly lower than typical scores in information extraction (e.g., up to .83 F1 in the KBP TAC 2017 challenge (Getman et al., 2017)). Several aspects contribute to the problem's hardness.

- *Training data quality*. We find that distantly supervised training data for recall is much noisier than for classical IE tasks, because knowledge bases such as Wikidata, despite having low error rates, have many gaps where they are incomplete (rather than incorrect). This mirrors a similar problem as found in (Mirza et al., 2018).
- *Low NED recall*. The task requires to match text mentions against KB entities. Yet even famous subjects frequently have obscure objects, e.g., none of Bill Gates' children has a Wikipedia page. NED tools consequently often failed to correctly resolve related mentions. In the present work we thus opted for lexical matching, trading a higher recall against a lower precision.
- *Time-variance*. While some KB relations are quite stable (e.g., children), others are more volatile, and may both grow or shrink over time (e.g., band membership) (Wijaya et al., 2015). Such dynamicity adds complexity to the recall assessment, as recall may then be specific to certain time points.

**Relative recall.** Our work has focused on estimating the recall w.r.t. reality, as judged by gold-standard annotators. An equally important question, close to previous work on species-count estimation (Salloum et al., 2013), is to estimate the recall relative to what can be maximally achieved by using the union of all possible sources. For instance, for many long-tail subjects, no source would hold complete information, but specific sources could still hold maximal information.

**Modelling and reasoning.** While we have shown that textual information can be useful in inferring the recall of extractions, recall estimation might benefit from more explicit modelling and reasoning. One relevant aspect could be temporal reasoning, for instance, a professional career without temporal gaps (e.g., high school till 1993, BSc. 1994-1997, then launch of a startup in 1998) is a helpful indicator towards complete education extraction. Such reasoning could be applied on top of temporal information extraction (Ling and Weld, 2010). Another aspect are statistical priors and typicality information. Information that rock bands often consist of one bassist, 1-2 guitarists, one vocalist and one drummer could be helpful in assessing extraction recall at extraction time, similar as done post-hoc in (Galárraga et al., 2017).

## 7 Conclusion

This paper presented a first investigation on IE coverage estimation. Our results support linguistic theories about scalar implicatures, and show that coverage estimation is generally feasible. The next challenge is to incorporate this into actual noisy IE extraction pipelines.

**Acknowledgment.** This research has been supported by an NVIDIA GPU hardware grant.

## References


Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting relations from large plain-text collections. In *JCDL*.

Vevake Balaraman, Simon Razniewski, and Werner Nutt. 2018. Recoin: Relative completeness in Wikidata. In *Wiki workshop at WWW*.

Robyn Carston. 1998. Informativeness, relevance and scalar implicature. *Pragmatics And Beyond New Series*.

Laura Chiticariu, Marina Danilevsky, Yunyao Li, Frederick Reiss, and Huaiyu Zhu. 2018. SystemT: Declarative text understanding for enterprise. In *NAACL*.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *KDD*.

Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S Weld, and Alexander Yates. 2004. Web-scale information extraction in KnowItAll. In *WWW*.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *KDD*.

Luis Galárraga, Simon Razniewski, Antoine Amarilli, and Fabian M Suchanek. 2017. Predicting completeness in knowledge bases. In *WSDM*.

Jeremy Getman, Joe Ellis, Zhiyi Song, Jennifer Tracey, and Stephanie M Strassel. 2017. Overview of linguistic resources for the TAC KBP 2017 evaluations: Methodologies and results. In *TAC*.

Herbert Paul Grice. 1975. Logic and conversation.

Panagiotis G. Ipeirotis, Eugene Agichtein, Pranay Jain, and Luis Gravano. 2007. Towards a query optimizer for text-centric tasks. *ACM TODS*.

Alpa Jain, Panagiotis G. Ipeirotis, and Luis Gravano. 2008. Building query optimizers for information extraction: the SQoUT project. *SIGMOD Record*.

Xiao Ling and Daniel S Weld. 2010. Temporal information extraction. In *AAAI*.

Mausam. 2016. Open information extraction systems and downstream applications. In *AAAI*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACL*.

Paramita Mirza, Simon Razniewski, Fariz Darari, and Gerhard Weikum. 2018. Enriching knowledge bases with counting quantifiers. In *ISWC*.

Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. 2012. PATTY: a taxonomy of relational patterns with semantic types. In *EMNLP*.

Harinder Pal and Mausam. 2016. Demonyms and compound relational nouns in nominal open IE. In *AKBC*.

Simon Razniewski, Vevake Balaraman, and Werner Nutt. 2017. Doctoral advisor or medical condition: Towards entity-specific rankings of knowledge base properties. In *ADMA*.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *NAACL*.

Mariam Salloum, Xin Luna Dong, Divesh Srivastava, and Vassilis J Tsotras. 2013. Online ordering of overlapping data sources. *VLDB*.

Jaeho Shin, Sen Wu, Feiran Wang, Christopher De Sa, Ce Zhang, and Christopher Ré. 2015. Incremental knowledge base construction using DeepDive. *VLDB*.

Gabriel Stanovsky, Julian Michael, Luke Zettlemoyer, and Ido Dagan. 2018. Supervised open information extraction. In *NAACL*.

Fabian M Suchanek, Mauro Sozio, and Gerhard Weikum. 2009. SOFIE: a self-organizing framework for information extraction. In *WWW*.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *ACL*.

Derry Tanti Wijaya, Ndapandula Nakashole, and Tom Mitchell. 2015. "a spousal relation begins with a deletion of engage and ends with an addition of divorce": Learning state changing verbs from wikipedia revision history. In *EMNLP*.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. WikiQA: A challenge dataset for open-domain question answering. In *EMNLP*.

# HMEAE: Hierarchical Modular Event Argument Extraction

**Xiaozhi Wang**[1*], **Ziqi Wang**[1*], **Xu Han**[1], **Zhiyuan Liu**[1†],
**Juanzi Li**[1], **Peng Li**[2], **Maosong Sun**[1], **Jie Zhou**[2], **Xiang Ren**[3]
[1]Department of Computer Science and Technology, Tsinghua University, Beijing, China
[2]Pattern Recognition Center, WeChat AI, Tencent Inc, China
[3]Department of Computer Science, University of Southern California, CA, USA
{xz-wang16, ziqi-wan16, hanxu17}@mails.tsinghua.edu.cn

## Abstract

Existing event extraction methods classify each argument role independently, ignoring the conceptual correlations between different argument roles. In this paper, we propose a Hierarchical Modular Event Argument Extraction (HMEAE) model, to provide effective inductive bias from the concept hierarchy of event argument roles. Specifically, we design a neural module network for each basic unit of the concept hierarchy, and then hierarchically compose relevant unit modules with logical operations into a role-oriented modular network to classify a specific argument role. As many argument roles share the same high-level unit module, their correlation can be utilized to extract specific event arguments better. Experiments on real-world datasets show that HMEAE can effectively leverage useful knowledge from the concept hierarchy and significantly outperform the state-of-the-art baselines. The source code can be obtained from https://github.com/thunlp/HMEAE.

## 1 Introduction

Event argument extraction (EAE) aims to identify the entities serving as event arguments and classify the roles they play in an event. For instance, given that the word "sold" triggers a `Transfer-Ownership` event in the sentence "*Steve Jobs* sold *Pixar* to *Disney*", EAE aims to identify that "*Steve Jobs*" is an event argument and its argument role is "`Seller`". Most event extraction (EE) methods treat EE as a two-stage problem, including event detection (ED, to identify the trigger word and determine the event type) and EAE. As ED is well-studied (Nguyen and Grishman, 2018; Zhao et al., 2018) in recent years, EAE becomes the bottleneck of EE.

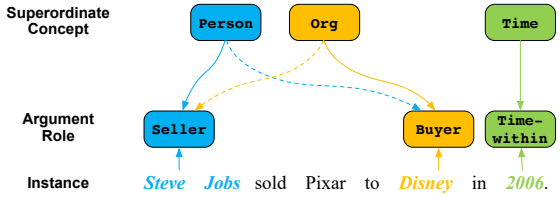


Figure 1: An example of the concept hierarchy.

Since EE benefits many NLP applications (Yang et al., 2003; Basile et al., 2014; Cheng and Erk, 2018), intensive efforts have been devoted to detecting events and extracting their event arguments. Traditional feature-based methods (Patwardhan and Riloff, 2009; Liao and Grishman, 2010b,a; Huang and Riloff, 2012; Li et al., 2013) rely on hand-crafted features and patterns. With the ongoing development of neural networks, various neural networks have been used to automatically represent textual semantics with low-dimensional vectors, and further extract event arguments based on those semantic vectors, including convolutional neural networks (Chen et al., 2015) and recurrent neural networks (Nguyen et al., 2016; Sha et al., 2018). Advanced techniques also have been adopted to further improve EE, such as zero-shot learning (Huang et al., 2018), multi-modal integration (Zhang et al., 2017), and weakly supervised methods (Chen et al., 2017; Wang et al., 2019).

However, the existing methods all treat argument roles as independent of each other, regardless of the fact that some argument roles are conceptually closer than others. Taking Figure 1 as an example, "`Seller`" is conceptually closer to "`Buyer`" than "`Time-within`", because they share the same superordinate concepts "`Person`" and "`Org`" in the concept hierarchy. Intuitively, the concept hierarchy will provide extra informa-

* indicates equal contribution
† Corresponding author: Z.Liu(liuzy@tsinghua.edu.cn)



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5777–5783,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

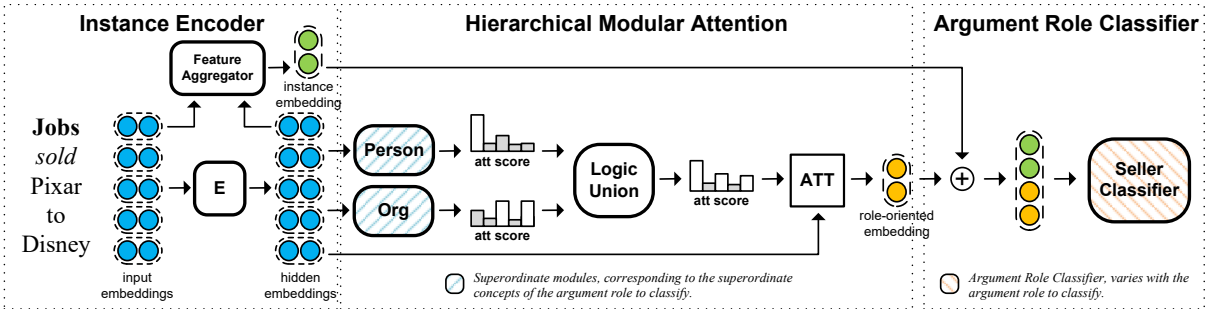


Figure 2: The overall architecture of HMEAE. Take the argument role "Seller" as an example.

tion about the correlation between argument roles and help the argument role classification.

To leverage the concept hierarchy information to improve EAE, we propose the Hierarchical Modular Event Argument Extraction (HMEAE) model. Inspired by the previous hierarchical classification works (Qiu et al., 2011; Shimura et al., 2018; Han et al., 2018) and the neural module networks (NMNs) (Andreas et al., 2016), HMEAE adopts the NMNs to enable a flexible network architecture imitating the concept hierarchical structure, which can provide effective inductive bias for better classification performance.

As Figure 1 shows, we divide the concepts into two types: the superordinate concepts representing more abstractive concepts, and the fine-grained argument roles. An argument role can belong to more than one superordinate concept, e.g., "Seller" belongs to both "Person" and "Org". As shown in Figure 2, we set a neural module network for each concept, and hierarchically compose them as the structure of the concept hierarchy into a role-oriented modular network to predict the argument role for each entity: (1) First, for each superordinate concept, a superordinate concept module (SCM) is instantiated to highlight the textual information related to the concept; (2) Then for each argument role, the SCMs corresponding to its superordinate concepts are composed by role-specific logic union modules to obtain a unified high-level module; (3) Finally, an argument role classifier is set to predict whether the entity is of the given argument role relying on the output of high-level modules.

Intuitively, considering the concept hierarchy in our model brings the following benefits: (1) The high-level modules can significantly enhance the classifiers, e.g., it is easier to determine whether an entity is "Time-within" when paying more attention to the words about "Time". (2) A superordinate concept module is shared among different argument roles in the concept hierarchy. Hence, it can capture the concept features from the shared information in the data of its subordinate argument roles, and further provide the correlation information as an effective inductive bias to the argument role classifiers. We conduct experiments on two real-world datasets. The experimental results show that our methods can achieve state-of-the-art results.

## 2 Methodology

In this section, we will introduce the overall framework of HMEAE. As shown in Figure 2, HMEAE consists of three components: (1) The instance encoder represents a sentence into hidden embeddings and utilizes a feature aggregator to aggregate sentence information into a unified instance embedding. (2) The hierarchical modular attention component builds a role-oriented embedding to highlight the information about the superordinate concepts of the argument role in the predefined concept hierarchy. (3) The argument role classifier relies on the instance embedding and the role-oriented embedding to estimate the probability of a certain argument role for the instance.

### 2.1 Instance Encoder

We denote an instance as an $n$-word sequence $x = \{w_1, \ldots, t, \ldots, a, \ldots, w_n\}$, where $t, a$ denote the trigger word and the candidate argument respectively. The trigger word is detected by the previous event detection models (independent of our work) and each named entity in the sentence is a candidate argument.

**Sentence Encoder** is adopted to encode the word sequence into hidden embeddings,

$$\{\boldsymbol{h}_1, \boldsymbol{h}_2 \ldots, \boldsymbol{h}_n\} = \mathrm{E}\big(w_1, \ldots, t, \ldots, a, \ldots, w_n\big), \quad (1)$$

where $\mathrm{E}(\cdot)$ is the neural network to encode the sentence. In this paper, we select CNN (Chen et al., 2015) and BERT (Devlin et al., 2019) as encoders.

**Feature Aggregator** aggregates the hidden embeddings into an instance embedding. Our method is independent of the feature aggregator mechanism. Here, we follow Chen et al. (2015) and use dynamic multi-pooling as the feature aggregator:

$$\begin{aligned}
[\boldsymbol{x}_{1,p_t}]_i &= \max\{[\boldsymbol{h}_1]_i, \ldots, [\boldsymbol{h}_{p_t}]_i\}, \\
[\boldsymbol{x}_{p_t+1,p_a}]_i &= \max\{[\boldsymbol{h}_{p_t+1}]_i, \ldots, [\boldsymbol{h}_{p_a}]_i\}, \\
[\boldsymbol{x}_{p_a+1,n}]_i &= \max\{[\boldsymbol{h}_{p_a+1}]_i, \ldots, [\boldsymbol{h}_n]_i\}, \\
\boldsymbol{x} &= [\boldsymbol{x}_{1,p_t}; \boldsymbol{x}_{p_t+1,p_a}; \boldsymbol{x}_{p_a+1,n}]
\end{aligned} \tag{2}$$

where $[\cdot]_i$ is the $i$-th value of a vector, $p_t, p_a$ are the positions of the trigger $t$ and the candidate argument $a$ respectively. We concatenate the piecewise max-pooling results as the instance embedding $\boldsymbol{x}$.

### 2.2 Hierarchical Modular Attention

As shown in Figure 2, given the hidden embeddings $\{\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_n\}$, a superordinate concept module gives an attention score for each hidden embedding to model its correlation with the specific superordinate concept. As an argument role can belong to more than one superordinate concept, we set a logic union module to combine the scores from different superordinate modules together. For each argument role, we hierarchically compose its superordinate concept modules into the integrated hierarchical modular attention component to build its role-oriented embedding.

**Superordinate Concept Module** For a specific superordinate concept $c$, we represent its semantic features with a trainable vector $\boldsymbol{u}_c$. Following Luong et al. (2015), we adopt a multi-layer perceptron to calculate the attention scores. We first calculate the hidden state,

$$\boldsymbol{h}_i^c = \tanh(\boldsymbol{W_a}[\boldsymbol{h}_i; \boldsymbol{u}_c]). \tag{3}$$

Then, we apply a softmax operation to get the attention score for the hidden embedding $\boldsymbol{h}_i$,

$$s_i^c = \frac{\exp(\boldsymbol{W_b}\boldsymbol{h}_i^c)}{\sum_{j=1}^{n}\exp(\boldsymbol{W_b}\boldsymbol{h}_j^c)}, \tag{4}$$

where $\boldsymbol{W_a}$ and $\boldsymbol{W_b}$ are trainable matrices shared among different superordinate concept modules.

**Logic Union Module** Given an argument role $r \in \mathcal{R}$, we denote its $k$ superordinate concepts as $c_1, c_2, \ldots, c_k$, and the corresponding attention scores for $\boldsymbol{h}_i$ are $s_i^{c_1}, s_i^{c_2}, \ldots, s_i^{c_k}$ computed by Eq. (4). As information about all the superordinate concepts should be retained in the role-oriented embedding, we calculate the mean of the attention scores as the role-oriented attention score,

$$s_i^r = \frac{1}{k}\sum_{j=1}^{k} s_i^{c_j}, \tag{5}$$

and then calculate the weighted sum of hidden embeddings as the role-oriented embedding,

$$\boldsymbol{e}^r = \sum_{i=1}^{n} s_i^r \boldsymbol{h}_i. \tag{6}$$

### 2.3 Argument Role Classifier

We concatenate the instance embedding $\boldsymbol{x}$ and the role-oriented embedding $\boldsymbol{e}^r$ as the input feature for the argument role classifier, and estimate the probability of $r \in \mathcal{R}$ of instance $x$ as follows:

$$p(r|x) = \frac{\exp(\boldsymbol{r}^\top[\boldsymbol{x}; \boldsymbol{e}^r])}{\sum_{\tilde{r}\in\mathcal{R}}\exp(\tilde{\boldsymbol{r}}^\top[\boldsymbol{x}; \boldsymbol{e}^{\tilde{r}}])}, \tag{7}$$

where $\boldsymbol{r}$ is the embedding of the argument role $r$.

The objective function is defined as follows:

$$\mathcal{L}(\theta) = -\sum_{l}\log p(r_l|x_l), \tag{8}$$

where $\theta$ is all parameters of our model. We adopt Adam (Kingma and Ba, 2015) to minimize $\mathcal{L}(\theta)$.

## 3 Experiments

### 3.1 Experimental Settings

In the experiments, our model with CNN as the encoder is named **HMEAE (CNN)**, whose most hyperparameters are the same as Chen et al. (2015) for a fair comparison. Our model with BERT as the encoder is named **HMEAE (BERT)**. For a fair comparison, we set a vanilla BERT baseline named **DMBERT**, which is without the hierarchical modular attention module but with a dynamic multi-pooling layer (Chen et al., 2015) as the feature aggregator. As our work does not involve the event detection stage, we conduct the argument role classification based on the event detection models in Chen et al. (2015) and Wang et al. (2019) for **HMEAE (CNN)** and **HMEAE (BERT)** respectively.

### Datasets and Evaluation

We evaluate our models on two real-world datasets: the widely-used ACE 2005 (Walker et al., 2006) and the newly-developed TAC KBP 2016 (Ellis et al., 2015).

**ACE 2005** (LDC2006T06) is the most widely-used dataset in event extraction. It contains 599 documents, which are annotated with 8 event types, 33 event subtypes, and 35 argument roles. We evaluate our models by the performance of argument classification. An argument is correctly classified if its event subtype, offsets and argument role match the annotation results. Following the previous works (Liao and Grishman, 2010b; Chen et al., 2015), we use the same test set containing 40 newswire documents, a development set with 30 randomly selected documents and training set with the remaining 529 documents.

**TAC KBP 2016** The Text Analysis Conference (TAC) is a series of evaluation workshops organized to encourage research in NLP and related applications. In this paper, we use the data of the TAC KBP 2016 Event Argument Extraction track (LDC2017E05). This competition annotates difficult test data but no training data. They encourage participants to use training data from any other sources. Hence we use the ACE 2005 dataset as our training data, which is less than the data used by the baselines on TAC KBP 2016.

### Concept Hierarchy Design

Considering there is not an existing ontology in the datasets, we manually design a concept hierarchy with 8 different superordinate concepts for our models. The principle of designing the concept hierarchy is to induce superordinate concepts from the specific labels using human experience. For instance, people can easily summarize "`Origin`" and "`Destination`" into "`Place`", which is the desired superordinate concept. Although the hierarchy used in this paper may not be generalized to other datasets with different label definitions, it is tractable to design an appropriate concept hierarchy with minimal human efforts and provide effective inductive bias via our method. For the details about the specific concept hierarchy used in this paper, please refer to the appendix.

### Hyperparameter Settings

**CNN** slides a convolution kernel over the input embedding sequence to get hidden embeddings. Following previous work, the input embedding of each word consists of its word embedding, position embedding, and event type embedding. The hyperparameter settings for HMEAE (CNN) are shown in Table 1.

| | |
|---|---|
| Learning Rate | 1e-03 |
| Batch Size | 20 |
| Word Embedding Dimension | 100 |
| Dropout Probability | 0.5 |
| Hidden Layer Dimension | 300 |
| Kernel Size | 3 |
| Position Embedding Dimension | 5 |
| Event Type Embedding Dimension | 5 |
| $u_c$ dimension | 900 |
| $W_b$ dimension | 900 |

Table 1: Hyperparameter settings for CNN models.

**BERT** adopts multi-layer bidirectional transformers to encode the input embedding sequence into hidden embeddings. The hyperparameters of DMBERT and HMEAE (BERT) are the same as the $\text{BERT}_{\text{BASE}}$ model. To utilize the event type information in our model, we append a special token into each input sequence for BERT to indicate the event type. Additional hyperparameters used in our experiments are shown in Table 2.

| | |
|---|---|
| Learning Rate | 6e-05 |
| Batch Size | 50 |
| Kernel Size | 3 |
| Warmup Rate | 0.1 |
| $u_c$ dimension | 900 |
| $W_b$ dimension | 900 |

Table 2: Hyperparameter settings for BERT models.

## 3.2 Overall Evaluation Results

We compare our models with various state-of-the-art baselines on ACE 2005: (1) Feature-based methods, including **Li's joint** (Li et al., 2013) and **RBPB** (Sha et al., 2016). (2) Vanilla neural network methods, including **DMCNN** (Chen et al., 2015) and **JRNN** (Nguyen et al., 2016). (3) Neural network with syntax information, like **dbRNN** (Sha et al., 2018) enhancing the recurrent neural network with dependency bridges to consider syntactically related information.

On TAC KBP 2016, we compare our models with the top systems (Dubbin et al., 2016; Hsi et al., 2016; Ferguson et al., 2016) of the competition as well as DMCNN and DMBERT.

| Method | Argument Role Classification | | |
|---|---|---|---|
| | P | R | F1 |
| Li's Joint (Li et al., 2013) | 64.7 | 44.4 | 52.7 |
| DMCNN (Chen et al., 2015) | 62.2 | 46.9 | 53.5 |
| RBPB (Sha et al., 2016) | 54.1 | 53.5 | 53.8 |
| JRNN (Nguyen et al., 2016) | 54.2 | **56.7** | 55.4 |
| dbRNN (Sha et al., 2018) | **66.2** | 52.8 | 58.7 |
| HMEAE (CNN) | 57.3 | 54.2 | 55.7 |
| DMBERT | 58.8 | 55.8 | 57.2 |
| HMEAE (BERT) | 62.2 | 56.6 | **59.3** |

Table 3: The overall results (%) on ACE 2005.

| Method | Argument Role Classification | | |
|---|---|---|---|
| | P | R | F1 |
| DISCERN-R (Dubbin et al., 2016) | 7.9 | 7.4 | 7.7 |
| Washington4 (Ferguson et al., 2016) | 32.1 | 5.0 | 8.7 |
| CMU CS Event1 (Hsi et al., 2016) | **31.2** | 4.9 | 8.4 |
| Washington1 (Ferguson et al., 2016) | 26.5 | 6.8 | 10.8 |
| DMCNN (Chen et al., 2015) | 17.9 | 16.0 | 16.9 |
| HMEAE (CNN) | 15.3 | 22.5 | 18.2 |
| DMBERT | 22.6 | 24.7 | 23.6 |
| HMEAE (BERT) | 24.8 | **25.4** | **25.1** |

Table 4: The overall results (%) on TAC KBP 2016.

The results are shown in Table 3 and Table 4. We have the following observations from the results: (1) HMEAE (CNN) and HMEAE (BERT) achieve improvements (about 2% in F1) as compared with DMCNN and DMBERT respectively, which have almost the same network framework with our models except the hierarchical modular attention. It indicates that our hierarchical modular method works well to enhance the EAE models with the inductive bias from the concept hierarchy. (2) HMEAE (BERT) is comparable to dbRNN and achieves the state-of-the-art performance among existing methods. Our methods adopt modular networks to consider the hierarchical concept knowledge in EAE, which is compatible with the sophisticated linguistic knowledge adopted in dbRNN. We will explore to integrate this two kinds of external knowledge together to further improve EAE in the future.

### 3.3 Case Study

To verify whether the superordinate concept modules work as we designed, we conduct a case study. We visualize the attention score $s_i^c$ of HMEAE (BERT) on a sentence randomly sampled from the ACE 2005 dataset in Figure 3. We

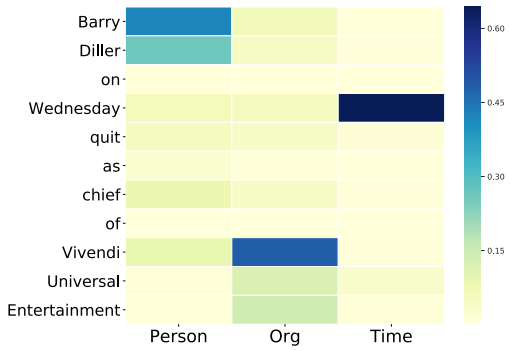


Figure 3: Heatmap for attention scores of three superordinate concept modules of the left sentence.

observe that the attention scores for the hidden embeddings of words related to the superordinate concept are much higher than the others. It indicates that as the superordinate concept modules are shared among their subordinate argument roles, the superordinate concept modules can capture the concept features well without being specially trained with exclusive data.

## 4 Conclusion and Future work

In this paper, we propose a hierarchical modular event argument extraction model (HMEAE), which adopts flexible modular networks to utilize the hierarchical concept correlation among argument roles as effective inductive bias. Experimental results show that HMEAE achieves the state-of-the-art performance. In the future, we will further explore to leverage other kinds of inductive bias from human experience to improve extensive tasks with our modular networks .

## Acknowledgement

This work is supported by the National Natural Science Foundation of China (NSFC No. 61572273, 61661146007) , the NSFC key project (U1736204) and Tsinghua University Initiative Scientific Research Program (20151080406). This work is also supported by the Pattern Recognition Center, WeChat AI, Tencent Inc. Xu Han and Xiaozhi Wang are also supported by 2018 and 2019 Tencent Rhino-Bird Elite Training Program respectively. Xiaozhi Wang is also supported by Tsinghua University Initiative Scientific Research Program. Xiang Ren is supported in part by National Science Foundation SMA 18-29268, DARPA MCS, GAILA, IARPA BETTER, and Schmidt Family Foundation.

## References


Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Neural module networks. In *Proceedings of CVPR*, pages 39–48.

P Basile, A Caputo, G Semeraro, and L Siciliani. 2014. Extending an information retrieval system through time event extraction. In *Proceedings of DART*, pages 36–47.

Yubo Chen, Shulin Liu, Xiang Zhang, Kang Liu, and Jun Zhao. 2017. Automatically labeled data generation for large scale event extraction. In *Proceedings of ACL*, pages 409–419.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of ACL-IJCNLP*, pages 167–176.

Pengxiang Cheng and Katrin Erk. 2018. Implicit argument prediction with event knowledge. In *Proceedings of ACL*, pages 831–840.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Greg Dubbin, Archna Bhatia, Bonnie Dorr, Adam Dalton, Kristy Hollingshead, Suriya Kandaswamy, Ian Perera, and Jena D Hwang. 2016. Improving discern with deep learning. In *TAC*.

Joe Ellis, Jeremy Getman, Dana Fore, Neil Kuster, Zhiyi Song, Ann Bies, and Stephanie M Strassel. 2015. Overview of linguistic resources for the TAC KBP 2016 evaluations: Methodologies and results. In *TAC*.

James Ferguson, Colin Lockard, Natalie Hawkins, Stephen Soderland, Hannaneh Hajishirzi, and Daniel S Weld. 2016. University of Washington TAC-KBP 2016 system description. In *TAC*.

Xu Han, Pengfei Yu, Zhiyuan Liu, Maosong Sun, and Peng Li. 2018. Hierarchical relation extraction with coarse-to-fine grained attention. In *Proceedings of EMNLP*, pages 2236–2245.

Andrew Hsi, Jaime G Carbonell, and Yiming Yang. 2016. CMU CS event TAC-KBP2016 event argument extraction system. In *TAC*.

Lifu Huang, Heng Ji, Kyunghyun Cho, Ido Dagan, Sebastian Riedel, and Clare Voss. 2018. Zero-shot transfer learning for event extraction. In *Proceedings of ACL*, pages 2160–2170.

Ruihong Huang and Ellen Riloff. 2012. Modeling textual cohesion for event extraction. In *Proceedings of AAAI*, pages 1664–1670.

Diederik P Kingma and Jimmy Ba. 2015. Adam: a method for stochastic optimization. In *Proceedings of ICLR*.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of ACL*, pages 73–82.

Shasha Liao and Ralph Grishman. 2010a. Filtered ranking for bootstrapping in event extraction. In *Proceedings of COLING*, pages 680–688.

Shasha Liao and Ralph Grishman. 2010b. Using document level cross-event inference to improve event extraction. In *Proceedings of ACL*, pages 789–797.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of EMNLP*, pages 1412–1421.

Thien Nguyen and Ralph Grishman. 2018. Graph convolutional networks with argument-aware pooling for event detection. In *Proceedings of AAAI*, pages 5900–5907.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of NAACL-HLT*, pages 300–309.

Siddharth Patwardhan and Ellen Riloff. 2009. A unified model of phrasal and sentential evidence for information extraction. In *Proceedings of EMNLP*, pages 151–160.

Xipeng Qiu, Xuanjing Huang, Zhao Liu, and Jinlong Zhou. 2011. Hierarchical text classification with latent concepts. In *Proceedings of ACL-HLT*, pages 598–602.

Lei Sha, Jing Liu, Chin-Yew Lin, Sujian Li, Baobao Chang, and Zhifang Sui. 2016. RBPB: Regularization-based pattern balancing method for event extraction. In *Proceedings of ACL*, pages 1224–1234.

Lei Sha, Feng Qian, Baobao Chang, and Zhifang Sui. 2018. Jointly extracting event triggers and arguments by dependency-bridge RNN and tensor-based argument interaction. In *Proceedings of AAAI*, pages 5916–5923.

Kazuya Shimura, Jiyi Li, and Fumiyo Fukumoto. 2018. HFT-CNN: Learning hierarchical category structure for multi-label short text categorization. In *Proceedings of EMNLP*, pages 811–816.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. ACE 2005 multilingual training corpus. *LDC, Philadelphia*.

Xiaozhi Wang, Xu Han, Zhiyuan Liu, Maosong Sun, and Peng Li. 2019. Adversarial training for weakly supervised event detection. In *Proceedings of NAACL-HLT*, pages 998–1008.

Hui Yang, Tat-Seng Chua, Shuguang Wang, and Chun-Keat Koh. 2003. Structured use of external knowledge for event-based open domain question answering. In *Proceedings of SIGIR*, pages 33–40.

Tongtao Zhang, Spencer Whitehead, Hanwang Zhang, Hongzhi Li, Joseph Ellis, Lifu Huang, Wei Liu, Heng Ji, and Shih-Fu Chang. 2017. Improving event extraction via multimodal integration. In *Proceedings of MM*, pages 270–278.

Yue Zhao, Xiaolong Jin, Yuanzhuo Wang, and Xueqi Cheng. 2018. Document embedding enhanced event detection with hierarchical and supervised attention. In *Proceedings of ACL*, pages 414–419.

# Entity, Relation, and Event Extraction with Contextualized Span Representations

**David Wadden**[†] **Ulme Wennberg**[†] **Yi Luan**[‡] **Hannaneh Hajishirzi**[†*]
[†]Paul G. Allen School of Computer Science & Engineering, University of Washington
[‡]Google AI Language [*]Allen Institute for Artificial Intelligence
{dwadden,ulme,hannaneh}@cs.washington.edu
luanyi@google.com

## Abstract

We examine the capabilities of a unified, multi-task framework for three information extraction tasks: named entity recognition, relation extraction, and event extraction. Our framework (called DyGIE++) accomplishes all tasks by enumerating, refining, and scoring text spans designed to capture local (within-sentence) and global (cross-sentence) context. Our framework achieves state-of-the-art results across all tasks, on four datasets from a variety of domains. We perform experiments comparing different techniques to construct span representations. Contextualized embeddings like BERT perform well at capturing relationships among entities in the same or adjacent sentences, while dynamic span graph updates model long-range cross-sentence relationships. For instance, propagating span representations via predicted coreference links can enable the model to disambiguate challenging entity mentions. Our code is publicly available at https://github.com/dwadden/dygiepp and can be easily adapted for new tasks or datasets.

## 1 Introduction

Many information extraction tasks – including named entity recognition, relation extraction, event extraction, and coreference resolution – can benefit from incorporating global context across sentences or from non-local dependencies among phrases. For example, knowledge of a coreference relationship can provide information to help infer the type of a difficult-to-classify entity mention. In event extraction, knowledge of the entities present in a sentence can provide information that is useful for predicting event triggers.

To model global context, previous works have used pipelines to extract syntactic, discourse, and other hand-engineered features as inputs to structured prediction models (Li et al., 2013; Yang and Mitchell, 2016; Li and Ji, 2014) and neural scoring functions (Nguyen and Nguyen, 2019), or as a guide for the construction of neural architectures (Peng et al., 2017; Zhang et al., 2018; Sha et al., 2018; Christopoulou et al., 2018). Recent end-to-end systems have achieved strong performance by dynmically constructing graphs of spans whose edges correspond to task-specific relations (Luan et al., 2019; Lee et al., 2018; Qian et al., 2018).

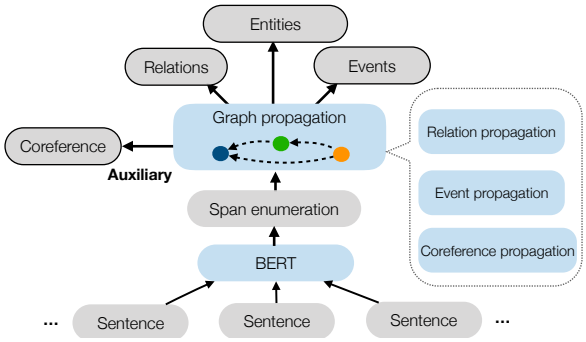


Figure 1: **Overview of our framework**: **DyGIE++**. Shared span representations are constructed by refining contextualized word embeddings via span graph updates, then passed to scoring functions for three IE tasks.

Meanwhile, contextual language models (Dai and Le, 2015; Peters et al., 2017, 2018; Devlin et al., 2018) have proven successful on a range of natural language processing tasks (Bowman et al., 2015; Sang and De Meulder, 2003; Rajpurkar et al., 2016). Some of these models are also capable of modeling context beyond the sentence boundary. For instance, the attention mechanism in BERT's transformer architecture can capture relationships between tokens in nearby sentences.

In this paper, we study different methods to incorporate global context in a general multi-task IE framework, building upon a previous span-based IE method (Luan et al., 2019). Our DyGIE++ framework, shown in Figure 1, enumerates candidate text



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5784–5789, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

spans and encodes them using contextual language models and task-specific message updates passed over a text span graph. Our framework achieves state-of-the results across three IE tasks, leveraging the benefits of both contextualization methods.

We conduct experiments and a thorough analysis of the model on named entity, relation, and event extraction. Our findings are as follows: (1) Our general span-based framework produces state-of-the-art results on all tasks and all but one subtasks across four text domains, with relative error reductions ranging from 0.2 - 27.9%. (2) BERT encodings are able to capture important within and adjacent-sentence context, achieving improved performance by increasing the input window size. (3) Contextual encoding through message passing updates enables the model to incorporate cross-sentence dependencies, improving performance beyond that of BERT alone, especially on IE tasks in specialized domains.

## 2 Task and Model

Our DyGIE++framework extends a recent span-based model for entity and relation extraction (Luan et al., 2019) as follows: (1) We perform event extraction as an additional task and propagate span updates across a graph connecting event triggers to their arguments. (2) We build span representations on top of multi-sentence BERT encodings.

### 2.1 Task definitions

The input is a document represented as a sequence of tokens $D$, from which our model constructs spans $S = \{s_1, \ldots, s_T\}$, the set of all possible within-sentence phrases (up to a threshold length) in the document.

Named Entity Recognition involves predicting the best entity type label $e_i$ for each span $s_i$. For all tasks, the best label may be a "null" label. Relation Extraction involves predicting the best relation type $r_{ij}$ for all span pairs $(s_i, s_j)$. For the data sets studied in this work, all relations are between spans within the same sentence. The coreference resolution task is to predict the best coreference antecedent $c_i$ for each span $s_i$. We perform coreference resolution as auxiliary task, to improve the representations available for the "main" three tasks.

Event Extraction involves predicting named entities, event triggers, event arguments, and argument roles. Specifically, each token $d_i$ is predicted as an event trigger by assigning it a label $t_i$. Then, for each trigger $d_i$, event arguments are assigned to this event trigger by predicting an argument role $a_{ij}$ for all spans $s_j$ in the same sentence as $d_i$. Unlike most work on event extraction, we consider the realistic setting where gold entity labels are not available. Instead, we use predicted entity mentions as argument candidates.

### 2.2 DyGIE++ Architecture

Figure 1 depicts the four-stage architecture. For more details, see (Luan et al., 2019).

**Token encoding:** DyGIE++ uses BERT for token representations using a "sliding window" approach, feeding each sentence to BERT together with a size-$L$ neighborhood of surrounding sentences.

**Span enumeration:** Spans of text are enumerated and constructed by concatenating the tokens representing their left and right endpoints, together with a learned span width embedding.

**Span graph propagation:** A graph structure is generated dynamically based on the model's current best guess at the relations present among the spans in the document. Each span representation $\mathbf{g}_j^t$ is updated by integrating span representations from its neighbors in the graph according to three variants of graph propagation. In coreference propagation, a span's neighbors in the graph are its likely coreference antecedents. In relation propagation, neighbors are related entities within a sentence. In event propagation, there are event trigger nodes and event argument nodes; trigger nodes pass messages to their likely arguments, and arguments pass messages back to their probable triggers. The whole procedure is trained end-to-end, with the model learning simultaneously how to identify important links between spans and how to share information between those spans.

More formally, at each iteration $t$ the model generates an update $\mathbf{u}_x^t(i)$ for span $s^t \in \mathbb{R}^d$:

$$\mathbf{u}_x^t(i) = \sum_{j \in B_x(i)} V_x^t(i,j) \odot \mathbf{g}_j^t, \qquad (1)$$

where $\odot$ denotes elementwise multiplication and $V_x^t(i,j)$ is a measure of similarity between spans $i$ and $j$ under task $x$ – for instance, a score indicating the model's confidence that span $j$ is the coreference antecedent of span $i$. For relation extraction, we use a ReLU activation to enforce sparsity. The final updated span representation $\mathbf{g}_j^{t+1}$ is computed as a convex combination of the previous representation and the current update, with weights determined by a gating function.

**Multi-task classification:** The re-contextualized representations are input to scoring functions which make predictions for each of the end tasks. We use a two-layer feedforward neural net (FFNN) as the scoring function. For trigger and named entity prediction for span $\mathbf{g}_i$, we compute $\text{FFNN}_{\text{task}}(\mathbf{g}_i)$. For relation and argument role prediction, we concatenate the relevant pair of embeddings and compute $\text{FFNN}_{\text{task}}([\mathbf{g}_i, \mathbf{g}_j])$.

## 3 Experimental Setup

**Data** We experiment on four different datasets: ACE05, SciERC, GENIA and WLPC (Statistics and details on all data sets and splits can be found in Appendix A.). The **ACE05** corpus provides entity, relation, and event annotations for a collection of documents from a variety of domains such as newswire and online forums. For named entity and relation extraction we follow the train / dev / test split from Miwa and Bansal (2016). Since the ACE data set lacks coreference annotations, we train on the coreference annotations from the OntoNotes dataset (Pradhan et al., 2012). For event extraction we use the split described in Yang and Mitchell (2016); Zhang et al. (2019). We refer to this split as ACE05-E in what follows. The **SciERC** corpus (Luan et al., 2018) provides entity, coreference and relation annotations from 500 AI paper abstracts. The **GENIA** corpus (Kim et al., 2003) provides entity tags and coreferences for 1999 abstracts from the biomedical research literature with a substantial portion of entities (24%) overlapping some other entity. The **WLPC** dataset provides entity, relation, and event annotations for 622 wet lab protocols (Kulkarni et al., 2018). Rather than treating event extraction as a separate task, the authors annotate event triggers as an entity type, and event arguments as relations between an event trigger and an argument.

**Evaluation** We follow the experimental setups of the respective state-of-the-art methods for each dataset: Luan et al. (2019) for entities and relations, and Zhang et al. (2019) for event extraction. An entity prediction is correct if its label and span matches with a gold entity; a relation is correct if both the span pairs and relation labels match with a gold relation triple. An event trigger is correctly identified if its offsets match a gold trigger. An argument is correctly identified if its offsets and event type match a gold argument. Triggers and arguments are correctly classified if their event types and event roles are also correct, respectively.

| Dataset | Task | SOTA | Ours | Δ% |
|---|---|---|---|---|
| ACE05 | Entity | 88.4 | **88.6** | 1.7 |
| | Relation | 63.2 | **63.4** | 0.5 |
| ACE05-Event* | Entity | 87.1 | **90.7** | 27.9 |
| | Trig-ID | 73.9 | **76.5** | 9.6 |
| | Trig-C | 72.0 | **73.6** | 5.7 |
| | Arg-ID | **57.2** | 55.4 | -4.2 |
| | Arg-C | 52.4 | **52.5** | 0.2 |
| SciERC | Entity | 65.2 | **67.5** | 6.6 |
| | Relation | 41.6 | **48.4** | 11.6 |
| GENIA | Entity | 76.2 | **77.9** | 7.1 |
| WLPC | Entity | 79.5 | **79.7** | 1.0 |
| | Relation | 64.1 | **65.9** | 5.0 |

Table 1: **DyGIE++ achieves state-of-the-art results**. Test set F1 scores of best model, on all tasks and datasets. We define the following notations for events: *Trig*: Trigger, *Arg*: argument, *ID*: Identification, *C*: Classification. * indicates the use of a 4-model ensemble for trigger detection. See Appendix E for details. The results of the single model are reported in Table 2 (c). We ran significance tests on a subset of results in Appendix D. All were statistically significant except Arg-C and Arg-ID on ACE05-Event.

**Model Variations** We perform experiments with the following variants of our model architecture. **BERT + LSTM** feeds pretrained BERT embeddings to a bi-directional LSTM layer, and the LSTM parameters are trained together with task specific layers. **BERT Finetune** uses supervised fine-tuning of BERT on the end-task. For each variation, we study the effect of integrating different task-specific message propagation approaches.

**Comparisons** For entity and relation extraction, we compare DyGIE++ against the DyGIE system it extends. DyGIE is a system based on ELMo (Peters et al., 2018) that uses dynamic span graphs to propagate global context. For event extraction, we compare against the method of Zhang et al. (2019), which is also an ELMo-based approach that relies on inverse reinforcement learning to focus the model on more difficult-to-detect events.

**Implementation Details** Our model is implemented using AllenNLP (Gardner et al., 2017). We use $\text{BERT}_{\text{BASE}}$ for entity and relation extraction tasks and use $\text{BERT}_{\text{LARGE}}$ for event extraction. For BERT finetuning, we use BertAdam with the learning rates of $1 \times 10^{-3}$ for the task specific layers, and $5.0 \times 10^{-5}$ for BERT. We use a longer warmup period for BERT than the warmup period for task specific-layers and perform linear decay of the learning rate following the warmup

| | ACE05 | SciERC | GENIA | WLPC |
|---|---|---|---|---|
| BERT + LSTM | 85.8 | 69.9 | 78.4 | **78.9** |
| +RelProp | 85.7 | 70.5 | - | 78.7 |
| +CorefProp | 86.3 | **72.0** | 78.3 | - |
| BERT Finetune | 87.3 | 70.5 | 78.3 | 78.5 |
| +RelProp | 86.7 | 71.1 | - | 78.8 |
| +CorefProp | **87.5** | 71.1 | **79.5** | - |

Table 2: F1 scores on NER.

| | ACE05 | SciERC | WLPC |
|---|---|---|---|
| BERT + LSTM | 60.6 | 40.3 | 65.1 |
| +RelProp | 61.9 | 41.1 | 65.3 |
| +CorefProp | 59.7 | 42.6 | - |
| BERT FineTune | **62.1** | 44.3 | 65.4 |
| +RelProp | 62.0 | 43.0 | **65.5** |
| +CorefProp | 60.0 | **45.3** | - |

Table 3: F1 scores on Relation.

| | Entity | Trig-C | Arg-ID | Arg-C |
|---|---|---|---|---|
| BERT + LSTM | 90.5 | 68.9 | **54.1** | **51.4** |
| +EventProp | **91.0** | 68.4 | 52.5 | 50.3 |
| BERT FineTune | 89.7 | **69.7** | 53.0 | 48.8 |
| +EventProp | 88.7 | 68.2 | 50.4 | 47.2 |

Table 4: F1 scores on ACE05-E.

Table 5: **Comparison of contextualization methods**. All ablations are performed on the dev set except for ACE05-E, where the precedent in the literature is to ablate on test.

period. Each of the feed-forward neural networks has two hidden layers and ReLU activations and 0.4 dropout. We use 600 hidden units for event extraction and 150 for entity and relation extraction (more details in Appendix E).

## 4 Results and Analyses

**State-of-the-art Results** Table 1 shows test set F1 on the entity, relation and event extraction tasks. Our framework establishes a new state-of-the-art on all three high-level tasks, and on all subtasks except event argument identification. Relative error reductions range from 0.2 - 27.9% over previous state of the art models.

**Benefits of Graph Propagation** Table 2 shows that Coreference propagation (CorefProp) improves named entity recognition performance across all three domains. The largest gains are on the computer science research abstracts of SciERC, which make frequent use of long-range coreferences, acronyms and abbreviations. CorefProp also improves relation extraction on SciERC.

Relation propagation (RelProp) improves relation extraction performance over pretrained BERT, but does not improve fine-tuned BERT. We believe

| Task | Variation | 1 | 3 |
|---|---|---|---|
| Relation | BERT+LSTM | 59.3 | **60.6** |
| | BERT Finetune | 62.0 | **62.1** |
| Entity | BERT+LSTM | 90.0 | **90.5** |
| | BERT Finetune | 88.8 | **89.7** |
| Trigger | BERT+LSTM | **69.4** | 68.9 |
| | BERT Finetune | 68.3 | **69.7** |
| Arg Class | BERT+LSTM | 48.6 | **51.4** |
| | BERT Finetune | **50.0** | 48.8 |

Table 6: **Effect of BERT cross-sentence context**. F1 score of relation F1 on ACE05 dev set and entity, arg, trigger extraction F1 on ACE05-E test set, as a function of the BERT context window size.

this occurs because all relations are within a single sentence, and thus BERT can be trained to model these relationships well.

Our best event extraction results did not use any propagation techniques (Table 4). We hypothesize that event propagation is not helpful due to the asymmetry of the relationship between triggers and arguments. Methods to model higher-order interactions among event arguments and triggers represent an interesting direction for future work.

**Benefits of Cross-Sentence Context with BERT** Table 6 shows that both variations of our BERT model benefit from wider context windows. Our model achieves the best performance with a 3-sentence window across all relation and event extraction tasks.

**Pre-training or Fine Tuning BERT Under Limited Resources** Table 5 shows that fine-tuning BERT generally performs slightly better than using the pre-trained BERT embeddings combined with a final LSTM layer.[1] Named entity recognition improves by an average of 0.32 F1 across the four datasets tested, and relation extraction improves by an average of 1.0 F1, driven mainly by the performance gains on SciERC. On event extraction, fine-tuning decreases performance by 1.6 F1 on average across tasks. We believe that this is due to the high sensitivity of both BERT finetuning and event extraction to the choice of optimization hyperparameters – in particular, the trigger detector begins overfitting before the argument detector is finished training.

Pretrained BERT combined with an LSTM layer and graph propagation stores gradients on 15 million parameters, as compared to the 100 million pa-

[1]Pre-trained BERT without a final LSTM layer performed substantially worse than either fine-tuning BERT, or using pre-trained BERT with a final LSTM layer.

| | SciERC | | GENIA |
|---|---|---|---|
| | Entity | Relation | Entity |
| Best BERT | 69.8 | 41.9 | 78.4 |
| Best SciBERT | **72.0** | **45.3** | **79.5** |

Table 7: In-domain pre-training: SciBERT vs. BERT

rameters in $\text{BERT}_{\text{BASE}}$. Since the BERT + LSTM + Propagation approach requires less memory and is less sensitive to the choice of optimization hyperparameters, it may be appealing for non-experts or for researchers working to quickly establish a reasonable baseline under limited resources. It may also be desirable in situations where fine-tuning BERT would be prohibitively slow or memory-intensive, for instance when encoding long documents like scientific articles.

**Importance of In-Domain Pretraining** We replaced BERT (Devlin et al., 2018) with SciBERT (Beltagy et al., 2019) which is pretrained on a large multi-domain corpus of scientific publications. Table 7 compares the results of BERT and SciBERT with the best-performing model configurations. SciBERT significantly boosts performance for scientific datasets including SciERC and GENIA. These results indicate that introducing unlabeled text of similar domains for pre-training can significantly improve performance.

**Qualitative Analysis** To better understand the mechanism by which graph propagation improved performance, we examined all named entities in the SciERC dev set where the prediction made by the BERT + LSTM + `CorefProp` model from Table 2 was different from the BERT + LSTM model. We found 44 cases where the `CorefProp` model corrected an error made by the base model, and 21 cases where it introduced an error. The model without `CorefProp` was often overly specific in the label it assigned, labeling entities as *Material* or *Method* when it should have given the more general label *Other Scientific Term*. Visualizations of the disagreements between the two model variants can be found in Appendix C. Figure 2 shows an example where span updates passed along a coreference chain corrected an overly-specific entity identification for the acronym "CCRs". We observed similar context sharing via `CorefProp` in the GENIA data set, and include an example in Appendix C.

Coreference propagation updated the span representations of all but one of 44 entities, and in 68% of these cases the update with the largest coreference "attention weight" came from a text span in a different sentence that was itself a named entity.

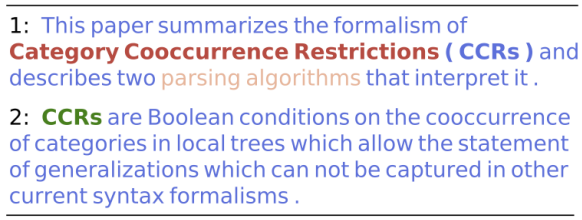


(a) The green span *CCRs* in sentence 2 is updated based on its predicted coreference antecedent.

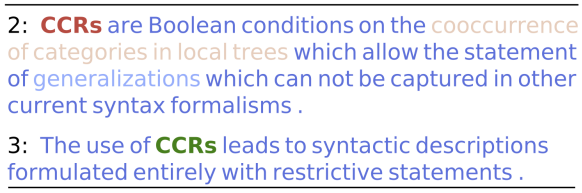


(b) The mention of *CCRs* in sentence 2 serves as a bridge to propagate information from sentence 1 to the mention of *CCRs* in sentence 3



(c) Coreference link strength. Red is strong.

Figure 2: **`CorefProp` enables a correct entity prediction**. In each subplot, the green token is being updated by coreference propagation. The preceeding tokens are colored according to the strength of their predicted coreference links with the green token. Tokens in **bold** are part of a gold coreference cluster discussing *CCRs*. During the `CorefProp` updates, the span *CCRs* in sentence 2 is updated based on its antecedent *Category Cooccurrence Restrictions*. Then, it passes this information along to the span *CCRs* in sentence 3. As a result, the model changes its prediction for *CCRs* in sentence 3 from *Method* – which is overly specific according to the SciERC annotation guideline – to the correct answer *Other Scientific Term*.

## 5 Conclusion

In this paper, we provide an effective plug-and-play IE framework that can be applied to many information extraction tasks. We explore the abilities of BERT embeddings and graph propagation to capture context relevant for these tasks. We find that combining these two approaches improves performance compared to using either one alone, with BERT building robust multi-sentence representations and graph propagations imposing additional structure relevant to the problem and domain under consideration. Future work could extend the framework to other NLP tasks and explore other approaches to model higher-order interactions like those present in event extraction.

## Acknowledgments

This research was supported by the ONR MURI N00014-18-1-2670, ONR N00014-18-1-2826, DARPA N66001-19-2-4031, NSF (IIS 1616112), Allen Distinguished Investigator Award, and Samsung GRO. We thank Mandar Joshi for his valuable BERT finetuning advice, Tongtao Zhang for sharing the ACE data code, anonymous reviewers, and the UW-NLP group for their helpful comments.

## References

Iz Beltagy, Arman Cohan, and Kyle Lo. 2019. Scibert: Pretrained contextualized embeddings for scientific text. *ArXiv*, abs/1903.10676.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*.

Fenia Christopoulou, Makoto Miwa, and Sophia Ananiadou. 2018. A walk-based model on entity graphs for relation extraction. In *ACL*.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *NeurIPs*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. Allennlp: A deep semantic natural language processing platform.

Jin-Dong Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. 2003. Genia corpus - a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19 Suppl 1:i180–2.

Chaitanya Kulkarni, Wei Xu, Alan Ritter, and Raghu Machiraju. 2018. An annotated corpus for machine reading of instructions in wet lab protocols. In *NAACL-HLT*.

Kenton Lee, Luheng He, and Luke S. Zettlemoyer. 2018. Higher-order coreference resolution with coarse-to-fine inference. In *NAACL-HLT*.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *ACL*.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *ACL*.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *EMNLP*.

Yi Luan, Dave Wadden, Luheng He, Amy Shah, Mari Ostendorf, and Hannaneh Hajishirzi. 2019. A general framework for information extraction using dynamic span graphs. In *NAACL-HLT*.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *ACL*.

Trung Minh Nguyen and Thien Huu Nguyen. 2019. One for all: Neural joint modeling of entities and events. In *AAAI*.

Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen tau Yih. 2017. Cross-sentence n-ary relation extraction with graph lstms. *Transactions of the Association for Computational Linguistics*, 5:101–115.

Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *ACL*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Joint Conference on EMNLP and CoNLL-Shared Task*, pages 1–40. Association for Computational Linguistics.

Yujie Qian, Enrico Santus, Zhijing Jin, Jiang Guo, and Regina Barzilay. 2018. Graphie: A graph-based framework for information extraction. In *NAACL-HLT*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *EMNLP*.

Erik F Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *NAACL*.

Lei Sha, Feng Qian, Baobao Chang, and Zhifang Sui. 2018. Jointly extracting event triggers and arguments by dependency-bridge rnn and tensor-based argument interaction. In *AAAI*.

Bishan Yang and Tom M. Mitchell. 2016. Joint extraction of events and entities within a document context. In *HLT-NAACL*.

Tongtao Zhang, Heng Ji, and Avirup Sil. 2019. Joint entity and event extraction with generative adversarial imitation learning. *Data Intelligence*, 1:99–120.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *EMNLP*.

# Next Sentence Prediction helps Implicit Discourse Relation Classification within and across Domains

**Wei Shi**[†] and **Vera Demberg**[†,‡]
[†]Dept. of Language Science and Technology
[‡]Dept. of Mathematics and Computer Science, Saarland University
Saarland Informatics Campus, 66123 Saarbrücken, Germany
{w.shi, vera}@coli.uni-saarland.de

## Abstract

Implicit discourse relation classification is one of the most difficult tasks in discourse parsing. Previous studies have generally focused on extracting better representations of the relational arguments. In order to solve the task, it is however additionally necessary to capture what events are expected to cause or follow each other. Current discourse relation classifiers fall short in this respect. We here show that this shortcoming can be effectively addressed by using the bidirectional encoder representation from transformers (BERT) proposed by Devlin et al. (2019), which were trained on a next-sentence prediction task, and thus encode a representation of likely next sentences. The BERT-based model outperforms the current state of the art in 11-way classification by 8% points on the standard PDTB dataset. Our experiments also demonstrate that the model can be successfully ported to other domains: on the BioDRB dataset, the model outperforms the state of the art system around 15% points.

## 1 Introduction

Discourse relation classification has been shown to be beneficial to multiple down-stream NLP tasks such as machine translation (Li et al., 2014), question answering (Jansen et al., 2014) and summarization (Yoshida et al., 2014). Following the release of the Penn Discourse Tree Bank (Prasad et al., 2008, PDTB), discourse relation classification has received a lot of attention from the NLP community, including two CoNLL shared tasks (Xue et al., 2015, 2016).

Discourse relations in texts are sometimes marked with an explicit connective (e.g., *but*, *because*, *however*), but these explicit signals are often absent. With explicit connectives acting as informative cues, it is relatively easy to classify the discourse relation with high accuracy (93.09% on four-way classification in (Pitler et al., 2008)). When there is no connective, classification has to rely on semantic information from the relational arguments. This task is very challenging, with state-of-the-art systems achieving accuracy of only 45% to 48% on 11-way classification. Consider example 1:

(1) [*The joint venture with Mr. Lang wasn't a good one.*]$_{Arg1}$ [**The venture, formed in 1986, was supposed to be Time's low-cost, safe entry into women's magazines.**]$_{Arg2}$
implicit `Comp.Concess.expectation` relation from PDTB: wsj_1903

In order to correctly classify the relation, it is necessary to understand that Arg1 raises the expectation that the next discourse segment may provide an explanation for why the venture wasn't good (e.g., that it was risky), and Arg2 contrasts with this discourse expectation. More generally, this means that a successful discourse relation classification model would have to be able to learn typical temporal event sequences, reasons, consequences etc. for all kinds of events. Statistical models attempted to address this intuition by giving models word pairs from the two arguments as features (Lin et al., 2009; Park and Cardie, 2012; Biran and McKeown, 2013; Rutherford and Xue, 2014), so that models could for instance learn to recognize antonym relations between words in the two arguments.

Recent models exploit such similarity relations between the two arguments, as well as simpler surface features that occur in one relational argument and correlate with specific coherence relations (e.g., the presence of negation, temporal expressions etc. may give hints as to what coherence relation may be present, see Park and Cardie (2012); Asr and Demberg (2015)). However, relations between arguments are often a lot more diverse than simple contrasts that can be captured



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5790–5796, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

through antonyms, and may rely on world knowledge (Kishimoto et al., 2018). It is hence clear that one cannot learn all these diverse relations from the very small amounts of available training data. Instead, we would have to learn a more general representation of discourse expectations.

Many recent discourse relation classification approaches have focused on cross-lingual data augmentation (Shi et al., 2017, 2019), training models to better represent the relational arguments by using various neural network models, including feed-forward network (Rutherford et al., 2017), convolutional neural networks (Zhang et al., 2015), recurrent neural network (Ji et al., 2016; Bai and Zhao, 2018), character-based (Qin et al., 2016) or formulating relation classification as an adversarial task (Qin et al., 2017). These models typically use pre-trained semantic embeddings generated from language modeling tasks, like Word2Vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014) and ELMo (Peters et al., 2018).

However, previously proposed neural models still crucially lack a representation of the *typical relations between sentences*: to solve the task properly, a model should ideally be able to form discourse expectations, i.e., to represent the typical causes, consequences, next events or contrasts to a given event described in one relational argument, and then assess the content of the second relational argument with respect to these expectations (see Example 1). Previous models would have to learn these relations only from the annotated training data, which is much too sparse for learning all possible relations between all events, states or claims.

The recently proposed BERT model (Devlin et al., 2019) takes a promising step towards addressing this problem: the BERT representations are trained using a language modelling and, crucially, a "next sentence prediction" task, where the model is presented with the actual next sentence vs. a different sentence and needs to select the original next sentence. We believe it is a good fit for discourse relation recognition, since the task allows the model to represent what a typical next sentence would look like.

In this paper, we show that a BERT-based model outperforms the current state of the art by 8% points in 11-way implicit discourse relation classification on PDTB. We also show that after pre-trained with small size cross-domain data, the model can be easily transferred to a new domain: it achieves around 16% accuracy gain on BioDRB compared to state of the art model. We also show that the Next Sentence Prediction task played an important role in these improvements.

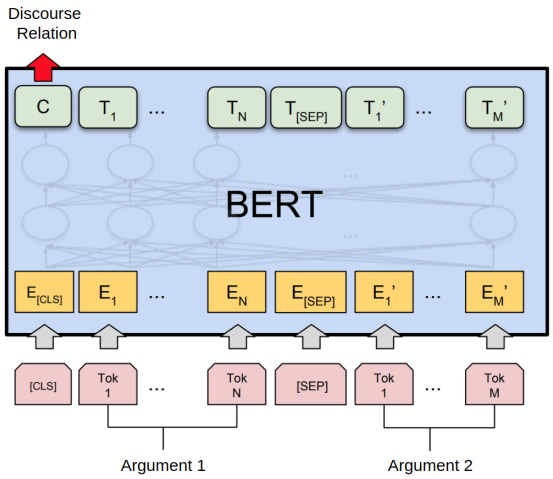


Figure 1: The architecture from BERT (Devlin et al., 2019) for fine-tuning of implicit discourse relation classification.

## 2 Next Sentence Prediction

Devlin et al. (2019) proposed the Bidirectional Encoder Representation from Transformers (BERT), which is designed to pre-train a deep bidirectional representation by jointly conditioning on both left and right contexts. BERT is trained using two novel unsupervised prediction tasks: Masked Language Modeling and Next Sentence Prediction (NSP). The NSP task has been formulated as a binary classification task: the model is trained to distinguish the original following sentence from a randomly chosen sentence from the corpus, and it showed great helps in multiple NLP tasks especially inference ones. The resulting BERT representations thus encode a representation of upcoming discourse content, and hence contain discourse expectation representations which, as we argued above, are required for classifying coherence relations.

## 3 Experiments and Results

As shown in Figure 1, $E$ denotes the input tokens' embedding and $T$ are the target words. In our case they are identical. We fit the implicit discourse relation task into sentence-pair classification proposed in BERT. Argument 1 and Argu-

| Methods | PDTB-Lin | PDTB-Ji | Cross Validation |
|---|---|---|---|
| Cai and Zhao (2017) | - | 45.81 | - |
| Kishimoto et al. (2018) | 38.77 | - | 39.80 |
| Bai and Zhao (2018) | 45.73 | 48.22 | - |
| Shi and Demberg (2019) | 45.82 | 47.83 | 41.29 |
| Bi-LSTM + w2v_300 | 37.95(0.59) | 40.57(0.67) | 37.82(0.74) |
| BERT | **53.13(0.37)** | **53.30(0.39)** | **49.30(1.33)** |
| BERT + WSJ w/o NSP | **53.39(0.49)** | **51.28(0.49)** | **49.32(1.24)** |
| BERT + WSJ | **54.82(0.61)**$*$ | **53.23(0.32)**$*$ | **49.35(0.83)** |

Table 1: Accuracy (%) with standard deviation in brackets of implicit discourse relation classification on different settings of PDTB level 2 relations. NSP refers to the subtask "next sentence prediction" in the pre-training of BERT. Numbers in bold signal significant improvements over the previous state of the art ($p<0.01$). Numbers with $*$ denote significant improvements over BERT + WSJ w/o NSP with $p<0.01$.

ment 2 are separated with token "[SEP]"; "[CLS]" is the special classification embedding while "C" is the same as "[CLS]" in pre-training but the ground-truth label in the fine-tuning. In the experiments, we used the uncased base model[1] provided by Devlin et al. (2019), which is trained on *BooksCorpus* and *English Wikipedia* with 3300M tokens in total.

### 3.1 Evaluation on PDTB

We used the Penn Discourse Tree Bank (Prasad et al., 2008), the largest available manually annotated discourse corpus. It provides a three level hierarchy of relation tags. Following the experimental settings and evaluation metrics in Bai and Zhao (2018), we use two most-used splitting methods of PDTB data, denoted as PDTB-Lin (Lin et al., 2009), which uses sections 2-21, 22, 23 as training, validation and test sets, and PDTB-Ji (Ji and Eisenstein, 2015), which uses 2-20, 0-1, 21-22 as training, validation and test sets and report the overall accuracy score. In addition, we also performed 10-fold cross validation among sections 0-22, as promoted in Shi and Demberg (2017). We also follow the standard in the literature to formulate the task as an 11-way classification task.

Results are presented in Table 1. We evaluated three versions of the BERT-based model. All of our BERT models use the pre-trained representations and are fine-tuned on the PDTB training data. The version marked as "BERT" does not do any additional pre-training. BERT+WSJ in addition performs further pre-training on the parts of the *Wall Street Journal* corpus that do not have discourse relation annotation. The model version "BERT+WJS w/o NSP" also performs pre-training on the WSJ corpus, but only uses the Masked Language Modelling task, not the Next Sentence Prediction task in the pre-training. We added this variant to measure the benefit of in-domain NSP on discourse relation classification (note though that the downloaded pre-trained BERT model contains the NSP task in the original pre-training).

We compared the results with four state-of-the-art systems: Cai and Zhao (2017) proposed a model that takes a step towards calculating discourse expectations by using attention over an encoding of the first argument, to generate the representation of the second argument, and then learning a classifier based on the concatenation of the encodings of the two discourse relation arguments. Kishimoto et al. (2018) fed external world knowledge (ConceptNet relations and coreferences) explicitly into MAGE-GRU (Dhingra et al., 2017) and achieved improvements compared to only using the relational arguments. However, we here show that it works even better when we learn this knowledge implicit through next sentence prediction task. Shi and Demberg (2019) used a seq2seq model that learns better argument representations due to being trained to explicitate the implicit connective. In addition, their classifier also uses a memory network that is intended to help remember similar argument pairs encountered during training. The current best performance was achieved by Bai and Zhao (2018), who combined representations from different grained em-

[1] https://github.com/google-research/bert#pre-trained-models

beddings including contextualized word vectors from ELMo (Peters et al., 2018), which has been proved very helpful. In addition, we compared our results with a simple bidirectional LSTM network and pre-trained word embeddings from Word2Vec.

We can see that on all settings, the model using BERT representations outperformed all existing systems with a substantial margin. It obtained improvements of 7.3% points on PDTB-Lin, 5.5% points on PDTB-Ji, compared with the ELMo-based method proposed in (Bai and Zhao, 2018). What's more, the BERT model outperformed (Shi and Demberg, 2019) on cross validation by around 8%, with significance of $p<0.01$. Significance test was performed by estimating variance of the model from the performance on different folds in cross-validation (paired t-test). For the Lin and Ji evaluations, we estimated variance due to random initialization by running them 5 times and calculating the likelihood that the state-of-the-art model result would come from that distribution.

### 3.2 Evaluation On BioDRB

The Biomedical Discourse Relation Bank (Prasad et al., 2011) also follows PDTB-style annotation. It is a corpus annotated over 24 open access full-text articles from the GENIA corpus (Kim et al., 2003) in the biomedical domain. Compared with PDTB, some new discourse relations and changes have been introduced in the annotation of BioDRB. In order to make the results comparable, we preprocessed the BioDRB annotations to map the relations to the PDTB ones, following the instructions in Prasad et al. (2011).

The biomedical domain is very different from the WSJ or the data on which the BERT model was trained. The BioDRB contains a lot of professional words / phrases that are extremely hard to model. In order to test the ability of the BERT model on cross-domain data, we performed fine-tuning on PDTB while testing on BioDRB. We also tested the state of the art model of implicit discourse relation classification proposed by Bai and Zhao (2018) on BioDRB. From Table 2, we can see that the BERT base model achieved almost 12% points improvement over the Bi-LSTM baseline and 15% points over Bai and Zhao (2018). When fine-tuned on in-domain data in the cross-validation setting, the improvement increases to around 17% points.

| Method | Cross-Domain | In-Domain |
|---|---|---|
| Bi-LSTM + w2v_300 | 32.97 | 46.49 |
| Bai and Zhao (2018) | 29.52 | 55.90 |
| BioBERT (Lee et al., 2019) | **44.33** | **67.58** |
| BERT | **44.79** | **63.02** |
| BERT + GENIA w/o NSP | **43.99** | **65.02** |
| BERT + GENIA | **45.19**$*$ | **66.04**$*$ |

Table 2: Accuracy (%) on BioDRB level 2 relations with different settings. Cross-Domain means trained on PDTB and tested on BioDRB. For the In-Domain setting, we used 5-fold cross-validation and report average accuracy. Numbers in bold are significantly better than the state of the art system with $p<0.01$ and numbers with $*$ denote denote significant improvements over BERT + GENIA w/o NSP with $p<0.01$.

It is also interesting to know whether the performance of the BERT model can be improved if we add additional pre-training on in-domain data. BioBert (Lee et al., 2019) continues pre-training BERT with bio-medical texts including PubMed and PMC corpora (around 18B tokens), which achieved the best results on in-domain setting. Similarly, BERT+GENIA refers to a model in which the downloaded BERT representations are further pre-trained on the parts of the GENIA corpus which consists of 18k sentences and is not annotated with coherence relations. Evaluation shows that this in-domain pre-training yields another 3% point improvement; our tests also show that the NSP task again plays a substantial role in the improvement. We believe that gains for further pre-training on GENIA for the biomedical domain are higher than for pre-training on WSJ for PDTB because the domain difference between the *BooksCorpus* and the biomedical domain is larger.

Currently there are not so many published results that we can compare with on BioDRB for implicit discourse relation classification. We compared BERT model with naïve Bayes and MaxEnt methods proposed in Xu et al. (2012) on one-versus-all binary classification. We followed the settings in Xu et al. (2012) and used two articles ("GENIA_1421503", "GENIA_1513057") for testing and one article ("GENIA_111020") for validation. During training, we employed down-sampling or up-sampling to keep the numbers of positive and negative samples in each relation consistent. The BERT base model achieved 43.03% average $F_1$ score and 77.34% average accuracy in one-versus-all level-1 classification. Compared with the current state-of-the-art perfor-

| Method | | Comp. | Cont. | Exp. | Temp. | Average |
|---|---|---|---|---|---|---|
| Naïve Bayes | (Xu et al., 2012) | 7.69(61.45) | 3.88(60.24) | 68.54(55.02) | 17.19(57.43) | 24.33(58.54) |
| MaxEnt | (Xu et al., 2012) | 7.08(57.83) | 3.36(53.82) | 72.32(60.64) | 23.81(61.45) | 26.64(58.44) |
| BERT | | **8.25(63.82)** | **10.26(71.54)** | **90.20(86.18)** | **63.41(87.80)** | **43.03(77.34)** |

Table 3: $F_1$-score (Accuracy) of binary classification on level 1 implicit relation in BioDRB.

| Relations | WSJ w/o NSP | | | WSJ w/ NSP | | | C. | GENIA w/o NSP | | | GENIA w/ NSP | | | C. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | P | R | $F_1$ | P | R | $F_1$ | | P | R | $F_1$ | P | R | $F_1$ | |
| Asynchronous | 0.38 | 0.46 | 0.41 | 0.29 | 0.38 | 0.33 | 13 | - | - | - | - | - | - | - |
| Synchrony | - | - | - | - | - | - | 5 | 0.87 | 0.84 | 0.85 | 0.90 | 0.88 | 0.89 | 80 |
| Cause | 0.57 | 0.65 | 0.61 | 0.57 | 0.64 | 0.60 | 200 | 0.22 | 0.10 | 0.14 | 0.23 | 0.15 | 0.18 | 20 |
| Prag. Cause | - | - | - | - | - | - | 5 | - | - | - | - | - | - | 1 |
| Contrast | 0.55 | 0.48 | 0.51 | 0.54 | 0.57 | 0.55 | 127 | - | - | - | - | - | - | 22 |
| Concession | - | - | - | - | - | - | 5 | - | - | - | 0.50 | 0.06 | 0.11 | 16 |
| Conjunction | 0.42 | 0.60 | 0.49 | 0.46 | 0.61 | 0.53 | 118 | 0.60 | 0.78 | 0.68 | 0.62 | 0.82 | 0.71 | 130 |
| Instantiation | 0.62 | 0.67 | 0.64 | 0.62 | 0.65 | 0.64 | 72 | - | - | - | - | - | - | 9 |
| Restatement | 0.52 | 0.45 | 0.48 | 0.55 | 0.45 | 0.50 | 190 | 0.56 | 0.76 | 0.65 | 0.59 | 0.69 | 0.64 | 72 |
| Alternative | 0.83 | 0.33 | 0.48 | 0.67 | 0.40 | 0.50 | 15 | - | - | - | - | - | - | 1 |
| List | 0.71 | 0.17 | 0.27 | 0.60 | 0.20 | 0.30 | 30 | - | - | - | - | - | - | - |
| Macro Avg. | 0.53 | 0.53 | 0.52 | 0.55 | 0.55 | 0.55 | 780 | 0.55 | 0.64 | 0.59 | 0.59 | 0.66 | 0.61 | 351 |

Table 4: Precision, Recall and $F_1$ score for each level-2 relation on PDTB-Lin setting and BioDRB with "BERT + WSJ/GENIA" systems w/ and w/o NSP. "-" indicates 0.00 and "C." means the number of each relation in the test set.

mances (26.64% $F_1$ and 58.54% accuracy) in Xu et al. (2012), it achieves around 16% and 19% points improvement when trained in-domain, as illustrated in Table 3.

### 3.3 Discussion

The usage of the BERT model in this paper was motivated primarily by the use of the next-sentence prediction task during training. The results in Table 1 and Table 2 confirm that removing the "Next Sentence Prediction" hurts the performance on both PDTB and BioDRB.

In order to have better insights about which relation has benefited from the NSP task, we also reported the detailed performance for each relation with and without it in BERT. As illustrated in Table 4, we can see that performances on relations like *Temporal.Synchrony, Comparison.Contrast, Expansion.Conjunction and Expansion.Alternative* have been improved by a large margin. This shows that representing the likely upcoming sentence helps the model form discourse expectations, which the classifier can then use to predict the coherence relation between the actually observed arguments.

However, compared with BERT+GENIA, the results of BioBert (Lee et al., 2019) in Table 2 show that having large in-domain data for pre-training also has limited ability in learning domain specific representations. We therefore believe that the model could be further improved by including external domain-specific knowledge from an ontology (as in Kishimoto et al. (2018)) or a causal graph for biomedical concepts and events.

## 4 Conclusion and Future work

In this paper, we show that BERT has very good ability in encoding the semantic relationship between sentences with its "next sentence prediction" task in pre-training. It outperformed the current state-of-the-art systems significantly with a substantial margin on both in-domain and cross domain data. Our results also indicate that the next-sentence prediction task during training indeed plays a role in this improvement. Future work should explore the joint representation of discourse expectations through implicit representations that are learned during training and the inclusion of external knowledge. In addition, Yang et al. (2019) showed that NSP only helps tasks with longer texts. It would be interesting to see whether it has the same effect on implicit discourse relation classification task, we'd like to leave that in the future work.

## 5 Acknowledgements

This research was funded by the German Research Foundation (DFG) as part of SFB 1102 "Information Density and Linguistic Encoding". We'd like to thank all the reviewers for their insightful and valuable comments.

## References


Fatemeh Torabi Asr and Vera Demberg. 2015. Uniform surprisal at the level of discourse relations: Negation markers and discourse connective omission. In *Proceedings of the 11th International Conference on Computational Semantics*, pages 118–128.

Hongxiao Bai and Hai Zhao. 2018. Deep enhanced representation for implicit discourse relation recognition. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 571–583.

Or Biran and Kathleen McKeown. 2013. Aggregated word pair features for implicit discourse relation disambiguation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 69–73, Sofia, Bulgaria. Association for Computational Linguistics.

Deng Cai and Hai Zhao. 2017. Pair-aware neural sentence modeling for implicit discourse relation classification. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pages 458–466. Springer.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Bhuwan Dhingra, Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. 2017. Linguistic knowledge as memory for recurrent neural networks. *arXiv preprint arXiv:1703.02620*.

Peter Jansen, Mihai Surdeanu, and Peter Clark. 2014. Discourse complements lexical semantics for non-factoid answer reranking. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 977–986.

Yangfeng Ji and Jacob Eisenstein. 2015. One vector is not enough: Entity-augmented distributed semantics for discourse relations. *Transactions of the Association of Computational Linguistics*, 3(1):329–344.

Yangfeng Ji, Gholamreza Haffari, and Jacob Eisenstein. 2016. A latent variable recurrent neural network for discourse relation language models. In *Proceedings of NAACL-HLT*, pages 332–342.

J-D Kim, Tomoko Ohta, Yuka Tateisi, and Junichi Tsujii. 2003. Genia corpusa semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(suppl_1):i180–i182.

Yudai Kishimoto, Yugo Murawaki, and Sadao Kurohashi. 2018. A knowledge-augmented neural network model for implicit discourse relation classification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 584–595.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *arXiv preprint arXiv:1901.08746*.

Junyi Jessy Li, Marine Carpuat, and Ani Nenkova. 2014. Assessing the discourse factors that influence the quality of machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 283–288.

Ziheng Lin, Min-Yen Kan, and Hwee Tou Ng. 2009. Recognizing implicit discourse relations in the penn discourse treebank. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 343–351. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Joonsuk Park and Claire Cardie. 2012. Improving implicit discourse relation recognition through feature set optimization. In *13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, page 108.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Emily Pitler, Mridhula Raghupathy, Hena Mehta, Ani Nenkova, Alan Lee, and Aravind Joshi. 2008. Easily identifiable discourse relations. In *Coling 2008: Companion volume: Posters*, pages 87–90.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The penn discourse treebank 2.0. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association (ELRA).

Rashmi Prasad, Susan McRoy, Nadya Frid, Aravind Joshi, and Hong Yu. 2011. The biomedical discourse relation bank. *BMC bioinformatics*, 12(1):188.

Lianhui Qin, Zhisong Zhang, and Hai Zhao. 2016. Implicit discourse relation recognition with context-aware character-enhanced embeddings. In *Proceedings of the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1914–1924.

Lianhui Qin, Zhisong Zhang, Hai Zhao, Zhiting Hu, and Eric Xing. 2017. Adversarial connective-exploiting networks for implicit discourse relation classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1006–1017.

Attapol Rutherford, Vera Demberg, and Nianwen Xue. 2017. A systematic study of neural discourse models for implicit discourse relation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 281–291.

Attapol Rutherford and Nianwen Xue. 2014. Discovering implicit discourse relations through brown cluster pair representation and coreference patterns. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 645–654. Association for Computational Linguistics.

Wei Shi and Vera Demberg. 2017. Do we need cross validation for discourse relation classification? In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 150–156.

Wei Shi and Vera Demberg. 2019. Learning to explicitate connectives with seq2seq network for implicit discourse relation classification. In *Proceedings of the 13th International Conference on Computational Semantics-Long Papers*, pages 188–199.

Wei Shi, Frances Yung, and Vera Demberg. 2019. Acquiring annotated data with cross-lingual explicitation for implicit discourse relation classification. In *Proceedings of the Workshop on Discourse Relation Parsing and Treebanking 2019*, pages 12–21.

Wei Shi, Frances Yung, Raphael Rubino, and Vera Demberg. 2017. Using explicit discourse connectives in translation for implicit discourse relation classification. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 484–495.

Yu Xu, Man Lan, Yue Lu, Zheng Yu Niu, and Chew Lim Tan. 2012. Connective prediction using machine learning for implicit discourse relation classification. In *Neural Networks (IJCNN), The 2012 International Joint Conference on*, pages 1–8. IEEE.

Nianwen Xue, Hwee Tou Ng, Sameer Pradhan, Rashmi Prasad, Christopher Bryant, and Attapol Rutherford. 2015. The conll-2015 shared task on shallow discourse parsing. In *Proceedings of the CoNLL-15 Shared Task*, pages 1–16. Association for Computational Linguistics.

Nianwen Xue, Hwee Tou Ng, Attapol Rutherford, Bonnie Webber, Chuan Wang, and Hongmin Wang. 2016. Conll 2016 shared task on multilingual shallow discourse parsing. In *Proceedings of the CoNLL-16 shared task*, pages 1–19. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Yasuhisa Yoshida, Jun Suzuki, Tsutomu Hirao, and Masaaki Nagata. 2014. Dependency-based discourse parser for single-document summarization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1834–1839.

Biao Zhang, Jinsong Su, Deyi Xiong, Yaojie Lu, Hong Duan, and Junfeng Yao. 2015. Shallow convolutional neural network for implicit discourse relation recognition. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2230–2235.

# Split or Merge: Which is Better for Unsupervised RST Parsing?

**Naoki Kobayashi**[†], **Tsutomu Hirao**[‡], **Kengo Nakamura**[‡],
**Hidetaka Kamigaito**[†], **Manabu Okumura**[†], **Masaaki Nagata**[‡]
[†] Institute of Innovative Research, Tokyo Institute of Technology,
[‡] NTT Communication Science Laboratories, NTT Corporation
{kobayasi, kamigaito}@lr.pi.titech.ac.jp, oku@pi.titech.ac.jp
{tsutomu.hirao.kp, kengo.nakamura.dx, masaaki.nagata.et}@hco.ntt.co.jp

## Abstract

Rhetorical Structure Theory (RST) parsing is crucial for many downstream NLP tasks that require a discourse structure for a text. Most of the previous RST parsers have been based on supervised learning approaches. That is, they require an annotated corpus of sufficient size and quality, and heavily rely on the language and domain dependent corpus. In this paper, we present two language-independent unsupervised RST parsing methods based on dynamic programming. The first one builds the optimal tree in terms of a dissimilarity score function that is defined for splitting a text span into smaller ones. The second builds the optimal tree in terms of a similarity score function that is defined for merging two adjacent spans into a large one. Experimental results on English and German RST treebanks showed that our parser based on span merging achieved the best score, around 0.8 $F_1$ score, which is close to the scores of the previous supervised parsers.

## 1 Introduction

Rhetorical Structure Theory (RST) (Mann and Thompson, 1988) is one of the theories that are most widely utilized for representing a discourse structure of a text in downstream NLP applications, such as automatic summarization (Marcu, 1998; Hirao et al., 2013), sentiment analysis (Bhatia et al., 2015), and text categorization (Ji and Smith, 2017). RST represents a text as a kind of constituent tree, whose leaves are Elementary Discourse Units (EDUs), clause-like units, and whose non-terminal nodes cover text spans consisting of a sequence of EDUs or a singleton EDU. The label of a non-terminal node represents the attribution of a text span, *nucleus* or *satellite*. A discourse relation is also assigned between two adjacent non-terminal nodes.

Since the RST tree can be regarded as a standard constituent (phrase structure) tree, syntactic parsing models for constituency parsing can also be successfully applied to RST parsing. In most cases, RST parsers have been developed on the basis of supervised learning algorithms, which require a high quality annotated corpus of sufficient size. As a result, research on RST parsing has focused on English, with the largest annotated corpus being the RST Discourse Treebank (RST-DT) (Carlson et al., 2001). These supervised RST parsing methods might not be applied to languages with only a small-size corpus.

This paper presents two types of language independent unsupervised RST parsing methods based on the CKY-like dynamic programming-based approach. One method builds the most likely parse tree in terms of a dissimilarity score defined for splitting a text span into two smaller ones. The other builds the optimal tree in terms of a similarity score defined for merging two adjacent text spans into a larger one. The similarity and dissimilarity scores between text spans are calculated on the basis of their distributional representations. Note that since our method is fully unsupervised, the parser predicts only the skeleton of an RST tree.

Moreover, we exploit multiple granularity levels of a document: (1) we first independently build three types of trees, a tree whose leaves correspond to a paragraph, a tree whose leaves correspond to a sentence, and a tree whose leaves correspond to an EDU, and then (2) merge them to obtain the whole RST tree.

We conducted experimental evaluation on English and German datasets, RST-DT and the Potsdam Commentary Corpus (PCC) (Stede and Neumann, 2014), respectively. The results demonstrated that our method with span merging outperformed our method with span splitting and ob-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5797–5802,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

tained .811 and .784 span scores for RST-DT and PCC, respectively. The scores are close to the scores of early supervised RST parsers.

## 2 Unsupervised RST Parsing

### 2.1 Motivation

Generally, RST trees with multinulcear relations are transformed into binary trees by applying right-heavy binarization as in syntactic parsing. As the result, we can easily imagine the following two simple parsing methods. First, when a split score for splitting a text span into two smaller spans is given for each candidate splitting point in the text span, we can build an RST tree by recursively splitting a text span with the best candidate points. Second, when a merge score for merging two adjacent spans into a larger one is given for each candidate pair of the spans, we can build an RST tree by recursively merging the best candidate pairs. Here, we can regard dissimilarity between two text spans as the split score and similarity between them as the merge score. Note that the dissimilarity score can be defined as $1 -$ similarity score. We propose the similarity score that is based on distributional representations of text spans in Section 2.2.

In general, while a parse tree can be obtained either by splitting or merging with a greedy parsing method, as mentioned above, it cannot be always optimal in terms of the total split or merge score. Thus, we propose our dynamic programming-based approach to obtain the optimal tree in Section 2.3.

Of course, it would be possible to utilize both similarity and dissimilarity scores. However, in this paper, we investigate which one is suitable for unsupervised RST parsing as the first step.

### 2.2 Similarity and Dissimilarity Scores between Text Spans

When two adjacent text spans, the left span $\ell$ from $i$-th to $k$-th atomic text unit[1] and the right span $r$ from $k+1$ to $j$-th atomic text unit are given, we define the similarity score between them as follows:

$$\mathrm{sim}(\overrightarrow{\ell_{i:k}}, \overrightarrow{r_{k+1:j}}) = \frac{1}{2}\left\{\frac{\overrightarrow{\ell_{i:k}} \cdot \overrightarrow{r_{k+1:j}}}{\|\overrightarrow{\ell_{i:k}}\|\|\overrightarrow{r_{k+1:j}}\|} + 1\right\}. \quad (1)$$

[1] A paragraph, a sentence or an Elementary Discourse Unit (EDU).

Here, $\overrightarrow{\ell_{i:k}}$ and $\overrightarrow{r_{k+1:j}}$ indicate the vector representations of the left and right spans, which are defined as a concatenation of two vectors for the left most atomic unit and the right most atomic unit as follows:

$$\begin{aligned}\overrightarrow{\ell_{i:k}} &= [\overrightarrow{u_i}; \overrightarrow{u_k}],\\ \overrightarrow{r_{k+1:j}} &= [\overrightarrow{u_{k+1}}; \overrightarrow{u_j}].\end{aligned} \quad (2)$$

The definition is inspired by that of (Ji and Eisenstein, 2014), which employs words at the beginning and end of a text span as important features to build an RST tree.

$\overrightarrow{u_t}$ is the vector representation of a $t$-th atomic unit $u_t$, which is defined on the basis of SIF (Arora et al., 2017)[2] as follows:

$$\overrightarrow{u_t} = \sum_{w \in W_t} \frac{a}{p(w) + a}\overrightarrow{w}. \quad (3)$$

$p(w)$ is the occurrence probability for word $w$, $\overrightarrow{w}$ is the vector representation of the word and $W_t$ is a set of words in $u_t$. We use a concatenation of two word vectors obtained from ELMo (Peters et al., 2018) and Glove (Pennington et al., 2014) as a vector representation of a word. $a$ is a parameter to decay the score of frequent words and is defined as $a = (1 - \alpha)/(\alpha Z)$, where $\alpha$ is a hyper parameter and $Z$ is the total number of words.

### 2.3 Dynamic Programming-Based Approach for Building Optimal Trees

We propose a dynamic programming-based approach to obtain the optimal tree in terms of either the total split or merge score from all the possible trees.

We first illustrate the algorithm with a merge (similarity) score. We define $V[b][e]$, which stores the maximum merge score for a span $u_{b:e}$ consisting from $b$-th unit to $e$-th unit, as follows:

$$V[b][e] = \begin{cases} 1, & b = e \\ \max\limits_{b \le k < e} V[b][k] \times \\ \quad \mathrm{sim}(\overrightarrow{\ell_{b:k}}, \overrightarrow{r_{k+1,e}}) \times \\ \quad\quad V[k+1][e], & \text{otherwise} \end{cases} \quad (4)$$

where $V[b][k]$ stores the maximum merge score for a sub-span $u_{b:k}$, and $V[k+1][e]$ stores the maximum merge score for a sub-span $u_{k+1:e}$.

[2] As for the representation for sentences and paragraphs, (Arora et al., 2017) compared SIF with the following three: avg-GloVe, tfidf-GloVe, skip-thought vectors (Kiros et al., 2015).

**Algorithm 1** Dynamic Programming-Based RST Parsing

1: $V^p[1][d] \leftarrow$DP$(u^p_{1:d})$
2: BACKTRACE$(V^p[1][d])$
3: **for** $i = 1$ to $m$ **do**
4: $b_i \leftarrow \text{start}(u^p_i)$, $e_i \leftarrow \text{end}(u^p_i)$
5: $V^s_i[b_i][e_i] \leftarrow$DP$(u^s_{b_i:e_i})$
6: BACKTRACE$(V^s_i[b_i][e_i])$
7: **for** $j = b_s$ to $e_s$ **do**
8: $b_j \leftarrow \text{start}(u^s_j)$, $e_j \leftarrow \text{end}(u^s_j)$
9: $V^e_j[b_j][e_j] \leftarrow$DP$(u^e_{b_j:e_j})$
10: BACKTRACE$(V^e_j[b_j][e_j])$
11: **function** DP(SPAN)
12: $V \leftarrow \phi$
13: $B \leftarrow \text{start}(\text{SPAN})$,$E \leftarrow \text{end}(\text{SPAN})$
14: **for** $i = 0$ to $E - B$ **do**
15: **for** $j = 0$ to $E - B - i$ **do**
16: $b \leftarrow B + j$, $e \leftarrow B + j + i$
17: **if** $b = e$ **then**
18: $V[b][e] = 1$
19: **else**
20: $V[b][e] = \max_{b \le k < e} V[b][k] \times$
21: $\text{sim}(\overrightarrow{\ell_{b:k}}, \overrightarrow{r_{k+1,e}}) \times V[k+1][e]$
**return** $V[B][E]$
22: **procedure** BACKTRACE$(V[b][e])$
23: $\hat{k} = \underset{k \in \{b,\ldots,e-1\}}{\arg\max} V[b][k] \times \text{sim}(\overrightarrow{\ell_{b:k}}, \overrightarrow{r_{k+1,e}}) \times$
24: $V[k+1][e]$
25: $C_\ell(u_{b:e}) \leftarrow u_{b:\hat{k}}$
26: $C_r(u_{b:e}) \leftarrow u_{\hat{k}+1:e}$
27: BACKTRACE$(V[b][\hat{k}])$
28: BACKTRACE$(V[\hat{k}+1][e])$

Algorithm 1 shows a dynamic programming-based RST parsing algorithm to build the optimal RST tree in terms of the total merge score. In the algorithm, DP is a function that fills out the triangular matrix $V$ as in the CKY algorithm, and BACKTRACE is a procedure to build a tree by recursively determining the left and right children for a span. Figure 1 shows an example of table $V$. The parse tree is obtained by recursively tracing back the path with the maximum score from $V[1][6]$. For example, the left child of span $u_{1:6}$ is span $u_{1:4}$ and the right child is span $u_{5:6}$.

Furthermore, we regard a document as a text span consisting of three different granularity levels: (1) a span from 1st to $d$-th paragraphs, assuming that the document consists of $d$ paragraphs, denoted as $u^p_{1:d}$, (2) a span from $b_i$-th to $e_i$-th sentences for the $i$-th paragraph $u^p_i$, which we denote as $u^s_{b_i:e_i}$, and (3) a span from $b_j$-th to $e_j$-th EDUs for the $j$-th sentence $u^s_j$, which we denote as $u^e_{b_j:e_j}$.

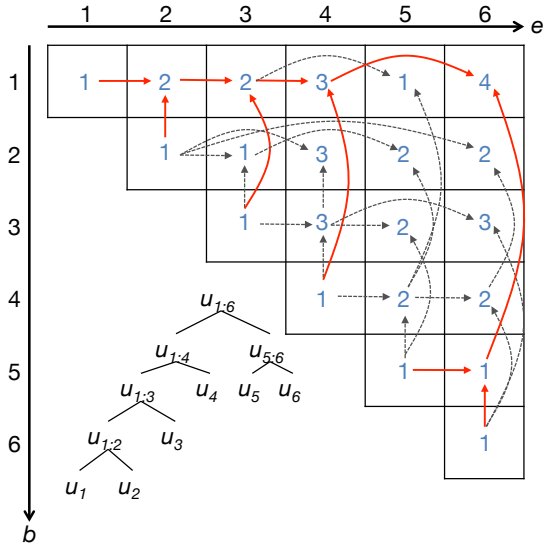


Figure 1: DP table and the RST tree-building.

| | RST-DT | PCC |
|---|---|---|
| # of Docs. | 385 | 174 |
| # of EDUs/Doc. | 56.6 | 16.1 |
| # of Sents./Doc. | 22.5 | 11.4 |
| # of Para./Doc. | 9.99 | 2.10 |
| # of EDUs/Sent. | 2.51 | 1.41 |
| # of EDUs/Para. | 5.67 | 7.66 |

Table 1: Statistics of the datasets.

First, the algorithm builds a document tree (lines 1-2). Then, it builds paragraph trees for each paragraph (lines 3-6) and builds sentence trees for each sentence (lines 7-10). Finally, we obtain an RST tree for the whole document by connecting the trees.

By replacing the similarity score, sim(,) with the dissimilarity score, $1-\text{sim}(,)$, we can obtain the optimal tree in terms of the split score. When we use the split score, the span consisting of only two atomic units can be divided into two uniquely, without calculating the split score for the span. Therefore, we compute $V[b][e] = 1$ when $e - b < 2$, instead of when $b = e$, in Equation (4). In addition, Algorithm 1, from lines 17 to 18, must be modified.

## 3 Experiments

### 3.1 Experimental Setting

We evaluated our methods on two datasets, the English RST treebank, RST-DT and the German RST treebank, PCC. RST-DT has 347 training docu-

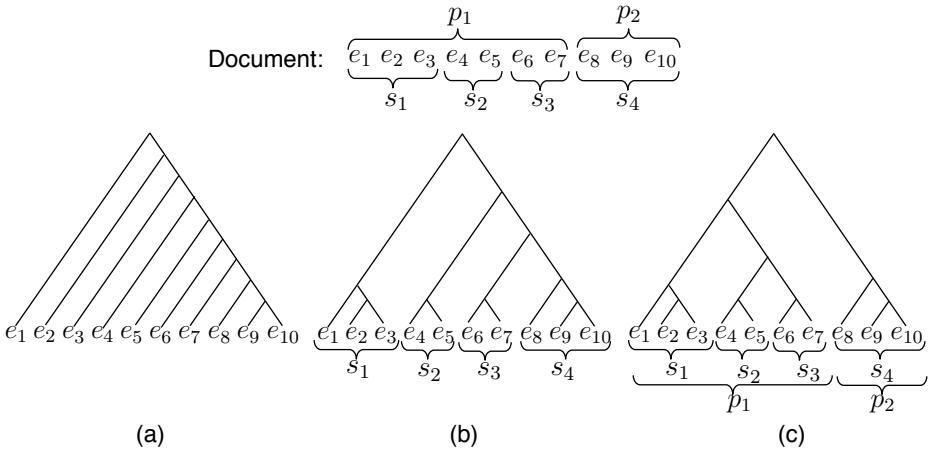


Figure 2: Right branching-trees of a document consists of ten EDUs, four sentences and two paragraphs. A right branching-tree with D2E setting (a), that with D2S2E setting (b) and that with D2P2S2E setting.

ments and 38 test documents. PCC[3] is smaller than RST-DT, and the number of documents is 174. We used the whole dataset as the test set on both datasets. Table 1 shows the statistics of the datasets. The numbers of paragraphs, sentences and EDUs per document for PCC are quite smaller than those for RST-DT.

We evaluated three variants of our methods, D2P2S2E, which is described in Algorithm 1, D2S2E, which employs two different granularity levels, *i.e.*, it regards a document as a span consisting of sentences and a sentence as a span consisting of EDUs, and D2E, which employs a single granularity level, *i.e.*, it regards a document as a span consisting of EDUs. To the best of our knowledge, there are no recent researches on unsupervised RST parsing, while we could find many on supervised RST parsing. As the result, we employed right-branching trees by merging the right most two spans, at each granularity level as baselines and compare those with ours. Figure 2 shows right branching trees at each granularity level. From the figure, we can see that the right-branching tree with two or three granularity, (b) and (c), have more complex structure than that with single granularity (a).

Since our unsupervised RST parsers can only construct unlabeled RST parse trees, we used only micro $F_1$ span score (Marcu, 2000) as an evaluation metric. Following previous studies, we conducted all the experiments on manually segmented EDUs.

[3]Since PCC contains two invalid RST trees, maz-12666 and maz-8838, we excluded them.

| Dataset | Gran. | Split | Merge | RB |
|---|---|---|---|---|
| RST-DT | D2E | .602 | **.656** | .545 |
| | D2S2E | .755 | **.788** | .751 |
| | D2P2S2E | .793 | **.811** | .803 |
| PCC | D2E | .656 | **.669** | .626 |
| | D2S2E | .757 | **.760** | .749 |
| | D2P2S2E | .787 | .784 | **.789** |

Table 2: Micro Span $F_1$ scores for RST-DT and PCC.

### 3.2 Results

Table 2 shows the evaluation results. Merge outperformed Split in most cases, and D2P2S2E achieved the best scores among our variants on both RST-DT and PCC. To clearly show the differences between our proposed method and RB, we performed significance tests, using paired bootstrap resampling (Koehn, 2004) at significance level=0.05. The results showed that there were significant differences between our method and RB at all the settings (D2E, D2S2E, D2P2S2E) for English, and at D2E and D2S2E for German, while there were no significant differences at D2P2S2E for German. The results imply that merging two adjacent spans and dividing with three granularity levels is suitable for unsupervised RST parsing. Comparing our methods with the baseline, right-branching (RB), we could find larger differences without considering the granularity levels of a document, whereas the differences became smaller by considering three granularity levels. In particular, right-branching

| Method | Span $F_1$ score |
|---|---|
| Merge | .808 |
| HHN16 | .826 |
| WLW17 | .856 |

Table 3: Micro Span $F_1$ scores on the test set of RST-DT.

was slightly better than our method on PCC. We think the result was related to the right-heavy binary conversion of RST trees: RST trees with multi-nuclear relations were transformed into binary trees beforehand by applying the right-heavy branching procedure. As a result, RB obtains better scores when the number of documents whose RST trees contain the multi-nuclear relations is larger. In fact, the ratio of the documents whose RST trees contain multi-nuclear relations in PCC is 0.712, while that in RST-DT is 0.464. We can obtain the information of sentences boundaries in most cases, however, sometimes we cannot obtain the information of paragraph boundaries. Thus, it is significant that our method outperformed baselines on D2S2E settings.

Moreover, we compared our method with some supervised RST parsers on the test set of RST-DT. Table 3 shows the results. In the table, HHN16 denotes a simple transition-based RST parser (Hayashi et al., 2016), and WLW17 denotes a current state-of-the-art transition-based RST parser (Wang et al., 2017). Although the score of our method is lower than the scores of the supervised parsers, the score is close to that of HHN16. The results demonstrated the effectiveness of our unsupervised RST parsing method. For reference, (Braud et al., 2017) reported that their supervised RST parser obtained .802 span $F_1$ score on PCC. However, we cannot compare the score with our score because the test set differs from ours.

## 4 Conclusion

This paper proposed two kinds of unsupervised RST parsing methods based on dynamic programming that can build the optimal RST tree in terms of either a span splitting score or a span merging score. We regarded a document as a text span consisting of three different granularity levels and built trees at each level, a document tree, paragraph trees for each paragraph, and sentence trees for each sentence. Then, we obtained an RST tree by connecting all trees together. To the best of our knowledge, this is the first study on unsupervised RST parsing. The evaluation results on RST-DT and PCC showed the effectiveness of our proposal; the dynamic programming-based approach with the span merging score, exploiting three granularity levels in a document, achieved .811 and .784 span $F_1$ scores on RST-DT and PCC, respectively. The results are close to the traditional transition-based supervised parser.

## References


Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *Proceedings of the 5th International Conference on Learning Representations*.

Parminder Bhatia, Yangfeng Ji, and Jacob Eisenstein. 2015. Better document-level sentiment analysis from rst discourse parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2212–2218.

Chloé Braud, Maximin Coavoux, and Anders Søgaard. 2017. Cross-lingual rst discourse parsing. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 292–304.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2001. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Proceedings of the Second SIGdial Workshop on Discourse and Dialogue*.

Katsuhiko Hayashi, Tsutomu Hirao, and Masaaki Nagata. 2016. Empirical comparison of dependency conversions for rst discourse trees. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 128–136.

Tsutomu Hirao, Yasuhisa Yoshida, Masaaki Nishino, Norihito Yasuda, and Masaaki Nagata. 2013. Single-document summarization as a tree knapsack problem. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1515–1520.

Yangfeng Ji and Jacob Eisenstein. 2014. Representation learning for text-level discourse parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 13–24.

Yangfeng Ji and Noah A. Smith. 2017. Neural discourse structure for text categorization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 996–1005.

Jamie Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. In *NIPS*.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*.

William C. Mann and Sandra A. Thompson. 1988. Rhetorical Structure Theory: Toward a functional theory of text organization. *Text*, 8(3):243–281.

Daniel Marcu. 1998. Improving summarization through rhetorical parsing tuning. In *Processing of the Sixth Workshop on Very Large Corpora*, pages 206–215.

Daniel Marcu. 2000. *The Theory and Practice of Discourse Parsing and Summarization*. MIT Press, Cambridge, MA, USA.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2227–2237.

Manfred Stede and Arne Neumann. 2014. Potsdam commentary corpus 2.0: Annotation for discourse research. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, pages 925–929.

Yizhong Wang, Sujian Li, and Houfeng Wang. 2017. A two-stage parsing method for text-level discourse analysis. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 184–188.

# BERT for Coreference Resolution: Baselines and Analysis

**Mandar Joshi** [†] **Omer Levy**[§] **Daniel S. Weld**[†ϵ] **Luke Zettlemoyer**[†§]

[†] Allen School of Computer Science & Engineering, University of Washington, Seattle, WA
{mandar90,weld,lsz}@cs.washington.edu

[ϵ]Allen Institute of Artificial Intelligence, Seattle
{danw}@allenai.org

[§] Facebook AI Research, Seattle
{omerlevy,lsz}@fb.com

## Abstract

We apply BERT to coreference resolution, achieving strong improvements on the OntoNotes (+3.9 F1) and GAP (+11.5 F1) benchmarks. A qualitative analysis of model predictions indicates that, compared to ELMo and BERT-base, BERT-large is particularly better at distinguishing between related but distinct entities (e.g., President and CEO). However, there is still room for improvement in modeling document-level context, conversations, and mention paraphrasing. Our code and models are publicly available[1].

## 1 Introduction

Recent BERT-based models have reported dramatic gains on multiple semantic benchmarks including question-answering, natural language inference, and named entity recognition (Devlin et al., 2019). Apart from better bidirectional reasoning, one of BERT's major improvements over previous methods (Peters et al., 2018; McCann et al., 2017) is passage-level training,[2] which allows it to better model longer sequences.

We fine-tune BERT to coreference resolution, achieving strong improvements on the GAP (Webster et al., 2018) and OntoNotes (Pradhan et al., 2012) benchmarks. We present two ways of extending the c2f-coref model in Lee et al. (2018). The *independent* variant uses non-overlapping segments each of which acts as an independent instance for BERT. The *overlap* variant splits the document into overlapping segments so as to provide the model with context beyond 512 tokens. BERT-large improves over ELMo-based c2f-coref 3.9% on OntoNotes and 11.5% on GAP (both absolute).

A qualitative analysis of BERT and ELMo-based models (Table 3) suggests that BERT-large (unlike BERT-base) is remarkably better at distinguishing between related yet distinct entities or concepts (e.g., Repulse Bay and Victoria Harbor). However, both models often struggle to resolve coreferences for cases that require world knowledge (e.g., *the developing story* and *the scandal*). Likewise, modeling pronouns remains difficult, especially in conversations.

We also find that BERT-large benefits from using longer context windows (384 word pieces) while BERT-base performs better with shorter contexts (128 word pieces). Yet, both variants perform much worse with longer context windows (512 tokens) in spite of being trained on 512-size contexts. Moreover, the *overlap* variant, which artificially extends the context window beyond 512 tokens provides no improvement. This indicates that using larger context windows for pretraining might not translate into effective long-range features for a downstream task. Larger models also exacerbate the memory-intensive nature of span representations (Lee et al., 2017), which have driven recent improvements in coreference resolution. Together, these observations suggest that there is still room for improvement in modeling document-level context, conversations, and mention paraphrasing.

## 2 Method

For our experiments, we use the higher-order coreference model in Lee et al. (2018) which is the current state of the art for the English OntoNotes dataset (Pradhan et al., 2012). We refer to this as *c2f-coref* in the paper.

### 2.1 Overview of c2f-coref

For each mention span $x$, the model learns a distribution $P(\cdot)$ over possible antecedent spans $Y$:

[1] https://github.com/mandarjoshi90/coref

[2] Each BERT training example consists of around 512 word pieces, while ELMo is trained on single sentences.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5803–5808, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

$$P(y) = \frac{e^{s(x,y)}}{\sum_{y' \in Y} e^{s(x,y')}} \quad (1)$$

The scoring function $s(x, y)$ between spans $x$ and $y$ uses fixed-length span representations, $\mathbf{g_x}$ and $\mathbf{g_y}$ to represent its inputs. These consist of a concatenation of three vectors: the two LSTM states of the span endpoints and an attention vector computed over the span tokens. It computes the score $s(x, y)$ by the mention score of $x$ (i.e. how likely is the span $x$ to be a mention), the mention score of $y$, and the joint compatibility score of $x$ and $y$ (i.e. assuming they are both mentions, how likely are $x$ and $y$ to refer to the same entity). The components are computed as follows:

$$s(x, y) = s_m(x) + s_m(y) + s_c(x, y) \quad (2)$$
$$s_m(x) = \text{FFNN}_m(\mathbf{g_x}) \quad (3)$$
$$s_c(x, y) = \text{FFNN}_c(\mathbf{g_x}, \mathbf{g_y}, \phi(x, y)) \quad (4)$$

where $\text{FFNN}(\cdot)$ represents a feedforward neural network and $\phi(x, y)$ represents speaker and metadata features. These span representations are later refined using antecedent distribution from a span-ranking architecture as an attention mechanism.

### 2.2 Applying BERT

We replace the entire LSTM-based encoder (with ELMo and GloVe embeddings as input) in c2f-coref with the BERT transformer. We treat the first and last word-pieces (concatenated with the attended version of all word pieces in the span) as span representations. Documents are split into segments of `max_segment_len`, which we treat as a hyperparameter. We experiment with two variants of splitting:

**Independent** The *independent* variant uses non-overlapping segments each of which acts as an independent instance for BERT. The representation for each token is limited to the set of words that lie in its segment. As BERT is trained on sequences of at most 512 word pieces, this variant has limited encoding capacity especially for tokens that lie at the start or end of their segments.

**Overlap** The *overlap* variant splits the document into overlapping segments by creating a $T$-sized segment after every $T/2$ tokens. These segments are then passed on to the BERT encoder independently, and the final token representation is derived by element-wise interpolation of representations from both overlapping segments.

Let $\mathbf{r_1} \in \mathbb{R}^d$ and $\mathbf{r_2} \in \mathbb{R}^d$ be the token representations from the overlapping BERT segments. The final representation $\mathbf{r} \in R^d$ is given by:

$$\mathbf{f} = \sigma(\mathbf{w}^T[\mathbf{r_1}; \mathbf{r_2}]) \quad (5)$$
$$\mathbf{r} = \mathbf{f} \cdot \mathbf{r_1} + (\mathbf{1} - \mathbf{f}) \cdot \mathbf{r_2} \quad (6)$$

where $\mathbf{w} \in \mathbb{R}^{2d \times d}$ is a trained parameter and $[;]$ represents concatenation. This variant allows the model to artificially increase the context window beyond the `max_segment_len` hyperparameter.

All layers in both model variants are then fine-tuned following Devlin et al. (2019).

## 3 Experiments

We evaluate our BERT-based models on two benchmarks: the paragraph-level GAP dataset (Webster et al., 2018), and the document-level English OntoNotes 5.0 dataset (Pradhan et al., 2012). OntoNotes examples are considerably longer and typically require multiple segments to read the entire document.

**Implementation and Hyperparameters** We extend the original Tensorflow implementations of c2f-coref[3] and BERT.[4] We fine tune all models on the OntoNotes English data for 20 epochs using a dropout of 0.3, and learning rates of $1 \times 10^{-5}$ and $2 \times 10^{-4}$ with linear decay for the BERT parameters and the task parameters respectively. We found that this made a sizable impact of 2-3% over using the same learning rate for all parameters.

We trained separate models with `max_segment_len` of 128, 256, 384, and 512; the models trained on 128 and 384 word pieces performed the best for BERT-base and BERT-large respectively. As span representations are memory intensive, we truncate documents randomly to eleven segments for BERT-base and 3 for BERT-large during training. Likewise, we use a batch size of 1 document following (Lee et al., 2018). While training the large model requires 32GB GPUs, all models can be tested on 16GB GPUs. We use the cased English variants in all our experiments.

**Baselines** We compare the c2f-coref + BERT system with two main baselines: (1) the original ELMo-based c2f-coref system (Lee et al., 2018), and (2) its predecessor, e2e-coref (Lee et al.,

[3] http://github.com/kentonl/e2e-coref/

[4] https://github.com/google-research/bert

| | MUC | | | $B^3$ | | | $CEAF_{\phi_4}$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | P | R | F1 | P | R | F1 | P | R | F1 | Avg. F1 |
| Martschat and Strube (2015) | 76.7 | 68.1 | 72.2 | 66.1 | 54.2 | 59.6 | 59.5 | 52.3 | 55.7 | 62.5 |
| (Clark and Manning, 2015) | 76.1 | 69.4 | 72.6 | 65.6 | 56.0 | 60.4 | 59.4 | 53.0 | 56.0 | 63.0 |
| (Wiseman et al., 2015) | 76.2 | 69.3 | 72.6 | 66.2 | 55.8 | 60.5 | 59.4 | 54.9 | 57.1 | 63.4 |
| Wiseman et al. (2016) | 77.5 | 69.8 | 73.4 | 66.8 | 57.0 | 61.5 | 62.1 | 53.9 | 57.7 | 64.2 |
| Clark and Manning (2016) | 79.2 | 70.4 | 74.6 | 69.9 | 58.0 | 63.4 | 63.5 | 55.5 | 59.2 | 65.7 |
| e2e-coref (Lee et al., 2017) | 78.4 | 73.4 | 75.8 | 68.6 | 61.8 | 65.0 | 62.7 | 59.0 | 60.8 | 67.2 |
| c2f-coref (Lee et al., 2018) | 81.4 | 79.5 | 80.4 | 72.2 | 69.5 | 70.8 | 68.2 | 67.1 | 67.6 | 73.0 |
| Fei et al. (2019) | **85.4** | 77.9 | 81.4 | **77.9** | 66.4 | 71.7 | 70.6 | 66.3 | 68.4 | 73.8 |
| EE (Kantor and Globerson, 2019) | 82.6 | **84.1** | 83.4 | 73.3 | **76.2** | 74.7 | 72.4 | 71.1 | 71.8 | 76.6 |
| BERT-base + c2f-coref (independent) | 80.2 | 82.4 | 81.3 | 69.6 | 73.8 | 71.6 | 69.0 | 68.6 | 68.8 | 73.9 |
| BERT-base + c2f-coref (overlap) | 80.4 | 82.3 | 81.4 | 69.6 | 73.8 | 71.7 | 69.0 | 68.5 | 68.8 | 73.9 |
| BERT-large + c2f-coref (independent) | 84.7 | 82.4 | **83.5** | 76.5 | 74.0 | **75.3** | **74.1** | **69.8** | **71.9** | **76.9** |
| BERT-large + c2f-coref (overlap) | 85.1 | 80.5 | 82.8 | 77.5 | 70.9 | 74.1 | 73.8 | 69.3 | 71.5 | 76.1 |

Table 1: OntoNotes: BERT improves the c2f-coref model on English by 0.9% and 3.9% respectively for base and large variants. The main evaluation is the average F1 of three metrics – MUC, $B^3$, and $CEAF_{\phi_4}$ on the test set.

| Model | M | F | B | O |
|---|---|---|---|---|
| e2e-coref | 67.2 | 62.2 | 0.92 | 64.7 |
| c2f-coref | 75.8 | 71.1 | 0.94 | 73.5 |
| BERT + RR Liu et al. (2019) | 80.3 | 77.4 | **0.96** | 78.8 |
| BERT-base + c2f-coref | 84.4 | 81.2 | **0.96** | 82.8 |
| BERT-large + c2f-coref | **86.9** | **83.0** | 0.95 | **85.0** |

Table 2: GAP: BERT improves the c2f-coref model by 11.5%. The metrics are F1 score on **M**asculine and **F**eminine examples, **O**verall, and a **Bi**as factor (F / M).

2017), which does not use contextualized representations. In addition to being more computationally efficient than e2e-coref, c2f-coref iteratively refines span representations using attention for higher-order reasoning.

### 3.1 Paragraph Level: GAP

GAP (Webster et al., 2018) is a human-labeled corpus of ambiguous pronoun-name pairs derived from Wikipedia snippets. Examples in the GAP dataset fit within a single BERT segment, thus eliminating the need for cross-segment inference. Following Webster et al. (2018), we trained our BERT-based c2f-coref model on OntoNotes.[5] The predicted clusters were scored against GAP examples according to the official evaluation script. Table 2 shows that BERT improves c2f-coref by 9% and 11.5% for the base and large models respectively. These results are in line with large gains reported for a variety of semantic tasks by BERT-based models (Devlin et al., 2019).

[5]This is motivated by the fact that GAP, with only 4,000 name-pronoun pairs in its dev set, is not intended for full-scale training.

### 3.2 Document Level: OntoNotes

OntoNotes (English) is a document-level dataset from the CoNLL-2012 shared task on coreference resolution. It consists of about one million words of newswire, magazine articles, broadcast news, broadcast conversations, web data and conversational speech data, and the New Testament. The main evaluation is the average F1 of three metrics – MUC, $B^3$, and $CEAF_{\phi_4}$ on the test set according to the official CoNLL-2012 evaluation scripts.

Table 1 shows that BERT-base offers an improvement of 0.9% over the ELMo-based c2f-coref model. Given how gains on coreference resolution have been hard to come by as evidenced by the table, this is still a considerable improvement. However, the magnitude of gains is relatively modest considering BERT's arguably better architecture and many more trainable parameters. This is in sharp contrast to how even the base variant of BERT has very substantially improved the state of the art in other tasks. BERT-large, however, improves c2f-coref by the much larger margin of 3.9%. We also observe that the *overlap* variant offers no improvement over *independent*.

Concurrent with our work, Kantor and Globerson (2019), who use higher-order entity-level representations over "frozen" BERT features, also report large gains over c2f-coref. While their feature-based approach is more memory efficient, the fine-tuned model seems to yield better results. Also concurrent, SpanBERT (Joshi et al., 2019), another self-supervised method, pretrains span representations achieving state of the art results (Avg. F1 79.6) with the *independent* variant.

| Category | Snippet | #base | #large |
|---|---|---|---|
| Related Entities | Watch spectacular performances by dolphins and sea lions at the *Ocean Theater*... It seems the North Pole and the Marine Life Center will also be renovated. | 12 | 7 |
| Lexical | Over the past 28 years , *the Ocean Park* has basically.. **The entire park** has been ... | 15 | 9 |
| Pronouns | In the meantime , our children need *an education*. **That**'s all we're asking. | 17 | 13 |
| Mention Paraphrasing | And in case you missed it *the Royals* are here. Today Britain's **Prince Charles and his wife Camilla**... | 14 | 12 |
| Conversation | (Priscilla:) **My** mother was Thelma Wahl . She was ninety years old ... (Keith:) *Priscilla Scott* is mourning . *Her* mother Thelma Wahl was a resident .. | 18 | 16 |
| Misc. | He is **my**, She is **my** Goddess , ah | 17 | 17 |
| *Total* | | 93 | 74 |

Table 3: Qualitative Analysis: #base and #large refers to the number of cluster-level errors on a subset of the OntoNotes English development set. Underlined and **bold-faced** mentions respectively indicate incorrect and missing assignments to *italicized* mentions/clusters. The miscellaneous category refers to other errors including (reasonable) predictions that are either missing from the gold data or violate annotation guidelines.

| Doc length | #Docs | Spread | F1 (base) | F1 (large) |
|---|---|---|---|---|
| 0 - 128 | 48 | 37.3 | 80.6 | 84.5 |
| 128 - 256 | 54 | 71.7 | 80.0 | 83.0 |
| 256 - 512 | 74 | 109.9 | 78.2 | 80.0 |
| 512 - 768 | 64 | 155.3 | 76.8 | 80.2 |
| 768 - 1152 | 61 | 197.6 | 71.1 | 76.2 |
| 1152+ | 42 | 255.9 | 69.9 | 72.8 |
| All | 343 | 179.1 | 74.3 | 77.3 |

Table 4: Performance on the English OntoNotes dev set generally drops as the document length increases. Spread is measured as the average number of tokens between the first and last mentions in a cluster.

| Segment Length | F1 (BERT-base) | F1 (BERT-large) |
|---|---|---|
| 128 | 74.4 | 76.6 |
| 256 | 73.9 | 76.9 |
| 384 | 73.4 | 77.3 |
| 450 | 72.2 | 75.3 |
| 512 | 70.7 | 73.6 |

Table 5: Performance on the English OntoNotes dev set with varying values for `max_segment_len`. Neither model is able to effectively exploit larger segments; they perform especially badly when `maximum_segment_len` of 512 is used.

## 4 Analysis

We performed a qualitative comparison of ELMo and BERT models (Table 3) on the OntoNotes English development set by manually assigning error categories (e.g., pronouns, mention paraphrasing) to incorrect predicted clusters.[6] Overall, we found 93 errors for BERT-base and 74 for BERT-large from the same 15 documents.

[6]Each incorrect cluster can belong to multiple categories.

**Strengths** We did not find salient qualitative differences between ELMo and BERT-base models, which is consistent with the quantitative results (Table 1). BERT-large improves over BERT-base in a variety of ways including pronoun resolution and lexical matching (e.g., *race track* and *track*). In particular, the BERT-large variant is better at distinguishing related, but distinct, entities. Table 3 shows several examples where the BERT-base variant merges distinct entities (like *Ocean Theater* and *Marine Life Center*) into a single cluster. BERT-large seems to be able to avoid such merging on a more regular basis.

**Weaknesses** An analysis of errors on the OntoNotes English development set suggests that better modeling of document-level context, conversations, and entity paraphrasing might further improve the state of the art.

*Longer documents* in OntoNotes generally contain larger and more spread-out clusters. We focus on three observations – (a) Table 4 shows how models perform distinctly worse on longer documents, (b) both models are unable to use larger segments more effectively (Table 5) and perform worse when the `max_segment_len` of 450 and 512 are used, and, (c) using overlapping segments to provide additional context does not improve results (Table 1). Recent work (Joshi et al., 2019) suggests that BERT's inability to use longer sequences effectively is likely a by-product pretraining on short sequences for a vast majority of updates.

Comparing preferred segment lengths for base and large variants of BERT indicates that larger

models might better encode longer contexts. However, larger models also exacerbate the memory-intensive nature of span representations,[7] which have driven recent improvements in coreference resolution. These observations suggest that future research in pretraining methods should look at more *effectively* encoding document-level context using sparse representations (Child et al., 2019).

Modeling *pronouns* especially in the context of conversations (Table 3), continues to be difficult for all models, perhaps partly because c2f-coref does very little to model dialog structure of the document. Lastly, a considerable number of errors suggest that models are still unable to resolve cases requiring *mention paraphrasing*. For example, bridging *the Royals* with *Prince Charles and his wife Camilla* likely requires pretraining models to encode relations between entities, especially considering that such learning signal is rather sparse in the training set.

## 5 Related Work

Scoring span or mention pairs has perhaps been one of the most dominant paradigms in coreference resolution. The base coreference model used in this paper from Lee et al. (2018) belongs to this family of models (Ng and Cardie, 2002; Bengtson and Roth, 2008; Denis and Baldridge, 2008; Fernandes et al., 2012; Durrett and Klein, 2013; Wiseman et al., 2015; Clark and Manning, 2016; Lee et al., 2017).

More recently, advances in coreference resolution and other NLP tasks have been driven by unsupervised contextualized representations (Peters et al., 2018; Devlin et al., 2019; McCann et al., 2017; Joshi et al., 2019). Of these, BERT (Devlin et al., 2019) notably uses pretraining on passage-level sequences (in conjunction with a bidirectional masked language modeling objective) to more effectively model long-range dependencies. SpanBERT (Joshi et al., 2019) focuses on pretraining span representations achieving current state of the art results on OntoNotes with the *independent* variant.

## References


Eric Bengtson and Dan Roth. 2008. Understanding the value of features for coreference resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 294–303, Stroudsburg, PA, USA. Association for Computational Linguistics.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

Kevin Clark and Christopher D. Manning. 2015. Entity-centric coreference resolution with model stacking. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1405–1415. Association for Computational Linguistics.

Kevin Clark and Christopher D. Manning. 2016. Deep reinforcement learning for mention-ranking coreference models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2256–2262. Association for Computational Linguistics.

Pascal Denis and Jason Baldridge. 2008. Specialized models and ranking for coreference resolution. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 660–669. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Greg Durrett and Dan Klein. 2013. Easy victories and uphill battles in coreference resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1971–1982. Association for Computational Linguistics.

Hongliang Fei, Xu Li, Dingcheng Li, and Ping Li. 2019. End-to-end deep reinforcement learning based coreference resolution. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 660–665, Florence, Italy. Association for Computational Linguistics.

Eraldo Fernandes, Cícero dos Santos, and Ruy Milidiú. 2012. Latent structure perceptron with feature induction for unrestricted coreference resolution. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 41–48. Association for Computational Linguistics.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2019. SpanBERT: Improving pre-training by representing and predicting spans. *arXiv preprint arXiv:1907.10529*.


[7] We required a 32GB GPU to finetune BERT-large.

Ben Kantor and Amir Globerson. 2019. Coreference resolution with entity equalization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 673–677, Florence, Italy. Association for Computational Linguistics.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197, Copenhagen, Denmark. Association for Computational Linguistics.

Kenton Lee, Luheng He, and Luke Zettlemoyer. 2018. Higher-order coreference resolution with coarse-to-fine inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 687–692, New Orleans, Louisiana. Association for Computational Linguistics.

Fei Liu, Luke Zettlemoyer, and Jacob Eisenstein. 2019. The referential reader: A recurrent entity network for anaphora resolution. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5918–5925, Florence, Italy.

Sebastian Martschat and Michael Strube. 2015. Latent structures for coreference resolution. *Transactions of the Association for Computational Linguistics*, 3:405–418.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6297–6308.

Vincent Ng and Claire Cardie. 2002. Identifying anaphoric and non-anaphoric noun phrases to improve coreference resolution. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, COLING '02, pages 1–7, Stroudsburg, PA, USA. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 1–40. Association for Computational Linguistics.

Kellie Webster, Marta Recasens, Vera Axelrod, and Jason Baldridge. 2018. Mind the GAP: A balanced corpus of gendered ambiguous pronouns. *Transactions of the Association for Computational Linguistics*, 6:605–617.

Sam Wiseman, Alexander M. Rush, Stuart Shieber, and Jason Weston. 2015. Learning anaphoricity and antecedent ranking features for coreference resolution. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1416–1426. Association for Computational Linguistics.

Sam Wiseman, Alexander M. Rush, and Stuart M. Shieber. 2016. Learning global features for coreference resolution. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 994–1004. Association for Computational Linguistics.

# Linguistic Versus Latent Relations for Modeling Coherent Flow in Paragraphs

**Dongyeop Kang  Hiroaki Hayashi  Alan W Black  Eduard Hovy**
Carnegie Mellon University, Pittsburgh, PA, USA
{dongyeok,hiroakih,awb,hovy}@cs.cmu.edu

## Abstract

Generating a long, coherent text such as a paragraph requires a high-level control of different levels of relations between sentences (e.g., tense, coreference). We call such a logical connection between sentences as a (paragraph) *flow*. In order to produce a coherent flow of text, we explore two forms of inter-sentential relations in a paragraph: one is a human-created linguistical relation that forms a structure (e.g., discourse tree) and the other is a relation from latent representation learned from the sentences themselves. Our two proposed models incorporate each form of relations into document-level language models: the former is a supervised model that jointly learns a language model as well as discourse relation prediction, and the latter is an unsupervised model that is hierarchically conditioned by a recurrent neural network (RNN) over the latent information. Our proposed models with both forms of relations outperform the baselines in partially conditioned paragraph generation task. Our codes and data are publicly available[1].

## 1 Introduction

When composing multiple sentences into a paragraph, as in novels or academic papers, we often make design decisions in advance (Byrne, 1979) such as topic introduction and content ordering to ensure better coherence of the text. For instance, McKeown (1985); Swan (2002) proposed effective patterns for scientific writing: a hypothesis at first, followed by supporting sentences to validate the hypothesis, and lastly a concluding sentence. We call such a logical connection between sentences in a written paragraph as a *flow*. A coherent flow between sentences requires an understanding of various factors including tense, coreference, plans (Appelt, 1982; Hovy, 1991), scripts (Tomkins, 1978) and several others. We focus on the paragraph-level plan between sentences.

In text planning, underlying relations in text are broadly categorized into two forms: an explicit human-defined relation (e.g., a discourse tree) (Reiter and Dale, 2000) or an implicitly learned latent relation (Yang et al., 2016). While the former is defined and manuallly annotated based on linguistic theories, the latter is simply determinable from how people in fact put sentences together. In this work, we provide an empirical comparison between a linguistically-informed and a latent form of relations in context of a paragraph generation.

We compare the effectiveness of the two forms of relations using language modeling for paragraph generation. Due to the different characteristics of the two forms, we employ comparable but different components in addition to the base language model. For linguistic relations (e.g., discourse), we cast the problem into multi-task learning of supervised language modeling and discourse relation prediction. On the other hand, for latent relations, we learn an unsupervised hierarchical language model that is hierarchically conditioned by RNNs over linear operations between sentences.

We evaluate our models on partial paragraph generation task; producing the rest of text in a paragraph given some context of text. We observe that linguistically annotated discourse relations help produce more coherent text than the latent relations, followed by other baselines.

## 2 Related Work

There has been a variety of NLG systems that incorporate additional information between sentences (Appelt, 1982; Reiter and Dale, 2000; Gatt and Krahmer, 2018) which can be broadly categorized into two forms: linguistic and latent.

[1] https://github.com/dykang/flownet



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5809–5815,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

**Linguistic** relations are explicitly represented as external labels in the form of predefined rules or plans, formats, knowledge base, discourse parses, and more. Hovy (1985, 1990); Dalianis and Hovy (1996) integrated text planning in generation, where the plans are considered in knowledge, formatted rules and so forth. However, they are limited to small scale (i.e. few examples) and hand-written rules. Kang et al. (2017); Gardent et al. (2017); Kang et al. (2018b); Wang et al. (2018) used an external knowledge base to micro-planning for generating a corresponding text, while our work focuses on comparing two forms of relations from the text itself.

Moore and Paris (1993); Young and Moore (1994) utilized discourse structures such as rhetorical structure theory (RST) (Mann and Thompson, 1988) for parsing a document. A script (Tomkins, 1978) is another structured representation that describes a typical sequence of events in a particular context. Zhang et al. (2016); Ji and Eisenstein (2014) proposed better discourse parsers using neural networks. The prior works, however, used the discourse representations to describe the structure of the paragraph, while we focus on applicability of the discourse relations to language generation.

**Latent** relations use implicit information in a document such as hierarchical structure of the document: Lin et al. (2015); Chung et al. (2016) used hierarchical RNN for modeling a document. Similarly, the hierarchical model can be extended to other variants such as attention (Yang et al., 2016), encoder-decoder framework (Serban et al., 2017; Sordoni et al., 2015), auto-encoding (Li et al., 2015), and multiscale (Chung et al., 2016). However, the hierarchical recurrence of sentences, which is dependent on topics, are less likely modeling a flow of a document.

We further summarize the fundamental differences between the two forms of relations in Appendix.

## 3 FlowNet: Language Modeling with Inter-sentential Relations

We propose language models that incorporate each relation to capture a high-level flow of text.

### 3.1 Discourse-driven FlowNet

As a linguistic relation, we employ RST (Mann and Thompson, 1988) trees to represent discourse connections in the text. For simplicity, we limit usage of the discourse trees by only considering relations between adjacent phrases[2]: relations are inserted between adjacent phrases and represented as a flattened sequence of phrases and relations. If two consecutive RST relations are given, the deeper level of relation is chosen. If the central elementary discourse unit (EDU) or phrase is after its dependent, the relation is excluded. We consider each sequence of the flattened discourse relations as a writing flow. For example, people often write a text by elaborating basic information (`Elaboration`) and then describing a following statement attributed to the information (`Attribution`).

We view discourse relations as additional labels to predict at the same time we predict next words in language modeling. Specifically, we propose to jointly train a model that predicts a sequence of words and a sequence of RST labels by taking advantage of shared representations, following previous sequence labeling problems such as named entity recognition (Collobert et al., 2011) and part-of-speech tagging (Huang et al., 2015). Note that the RST relations are only used during training to obtain better representation for the two tasks, but not at test time.

Figure 1(a) shows our FlowNet using discourse relations. Let a paragraph be a sequence of sentences $D$=$\{s_1, s_2, \ldots, s_M\}$. This model treats adjacent sentences as pairs for learning the standard seq2seq model. The first objective is to maximize the likelihood of the current sentence given the previous sentence. Hence, we maximize the following:

$$\mathbb{L}_{s2s} = \sum_j \log P(w_{ij} | w_{i,<j}, s_{i-1}) \qquad (1)$$

where $s_i$=$\{w_{i1}, w_{i2}, \ldots, w_{iT_i}\}$, and $T_i$ is the number of tokens of $s_i$.

To better guide the model with discourse context, we use the shared representations to predict RST relations at the same time. For each paragraph, we run the pre-trained RST parser (Ji and Eisenstein, 2014) and flatten the parse tree to obtain RST relations for each sentence $Y_i$=$(y_1, \ldots, y_{K_i})$, where $K_i$ is the number of discourse relations in $s_i$. We then make a label sequence over tokens in the sentence with

[2]The full discourse tree can be incorporated using other types of language model such as Tai et al. (2015).

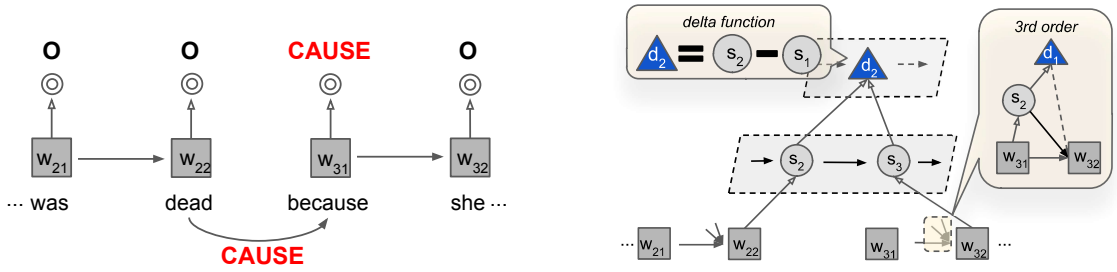


(a) Discourse-driven (b) Delta-driven

Figure 1: FlowNet with linguistic (i.e., discourse) versus latent (i.e., delta) relation. (a) For each word, a form of discourse relation and next word are jointly predicted using CRF (◎) and language model, respectively. (b) Decoding $w_i$ is conditioned on previous word ($w_{i-1}$), previous sentence ($s_{i-1}$), and delta between two previous sentences ($d_{i-2}$). Best viewed in color.

by placing $y$ at the first word of EDUs and filling up the rest with a *null* relation $o$: $Y_i' = (o, \ldots, o, \mathbf{y_1}, o, \ldots, \mathbf{y_{K_i}}, o, \ldots, o)$. We incorporate a sequence labeling objective by employing conditional random field (Lafferty et al., 2001) to find the label sequence that maximizes the score function for each sentence $s_i$: $\mathbf{S}(s_i, Y_i') = \sum_{j=1}^{T_i-1} W^T_{y_j', y_{j+1}'} h_j + b_{y_j', y_{j+1}'}$ where $h_j$, $W$ and $b$ are the hidden representation of $w_{ij}$, weight matrix, and the bias vector corresponding to the pair of labels $(y_i', y_{i+1}')$, respectively. For training, we maximize the conditional likelihood:

$$\mathbb{L}_{CRF} = \mathbf{S}(s_i, y_i') - \sum_{\mathbf{y} \in \mathbb{Y}_x} \log \mathbf{S}(s_i, \mathbf{y}) \quad (2)$$

where $\mathbb{Y}_x$ represents all possible discourse label sequences. Decoding is done by greedily predicting the output sequence with maximum score. Both training and decoding can be computed using dynamic programming. The final objective is represented as the sum of two objective functions:

$$\mathbb{L}_{\texttt{disc}} = \mathbb{L}_{s2s} + \alpha * \mathbb{L}_{CRF} \quad (3)$$

where $\alpha$ is a scaling parameter to control the impact of CRF objective. The value is chosen empirically by searching based on validation set.

### 3.2 Delta-driven FlowNet

In this model, we aim to utilize latent representations to characterize the flow between sentences. Specifically we define *delta*, subtractions of hidden represenations of adjacent sentences as such latent information. Figure 1(b) shows how we hierarchically model different levels of information: words, sentences, and *deltas*.

Each word is encoded using a RNN encoder $g_{word}$. We take the last hidden representation of word as sentence embeddings $s_1, ..., s_M$. Similar to hierarchical RNN (Lin et al., 2015), each sentence representation is encoded using another RNN encoder $g_{sent}$. While discourse flow provides an explicit relation symbols, delta flow calculates a latent relation by subtracting previous representation $s_{i-1}$ from current representation $s_i$[3]:

$$d(s_{i-1}, s_i) = d_{i-1} = s_i - s_{i-1} \quad (4)$$

Given a sequence of $M$-1 delta relations $d_1, ..., d_{M-1}$ for a paragraph of $M$ sentences, we again encode them using another RNN encoder $g_{delta}$. The model takes the word, sentence and delta information altogether to predict the next ($t$-th) word in the $m$-th sentence:

$$h_t = f(h_{t-1}, x_t, s_{m-1}, d_{m-2}) \quad (5)$$

where $x_t$ is a word representation, $s_{m-1}$ is a sentence representation and $d_{m-2}$ is a delta information. Note that sentence representation is from the previous sentence, and delta information is calculated by two previous sentences. If there is no previous information given, the parameters are randomly initialized.

## 4 Experiment

Due to the absence of goal-oriented language generation task, we collect paragraph data and define a new task of generating partial text of a paragraph given some context.

### 4.1 Data

We collect paragraphs from three different domains: `Papers` are paragraphs extracted from

[3]Our experiment includes a comparison among other types of linear operations between sentences such as addition or a learnable function.

| | Train | Valid | Test |
|---|---|---|---|
| Papers | 16,173 | 899 | 899 |
| SciFi | 157,031 | 8,724 | 8,724 |
| Fantasy | 317,654 | 17,649 | 17,649 |

Table 1: Number of paragraphs in our dataset.

academic manuscripts in computer science domain from the PeerRead (Kang et al., 2018a), and Fantasy and SciFi are paragraphs of two frequent categories extracted from the BookCorpus (Zhu et al., 2015), where paragraphs are extracted using the line breaker in the dataset.

We only use paragraphs whose lengths are from 4 to 7, in order to measure the performance change according to paragraph length. The dataset is randomly split by 0.9/0.05/0.05 for train, valid, and test set, respectively. Table 1 shows the numbers of paragraphs for each domain. All paragraphs are parsed into RST trees using the state-of-the-art discourse parser by Ji and Eisenstein (2014).

### 4.2 *Bridging*: Partial Paragraph Generation

We evaluate our models on partial text generation task; given a partial information (e.g., some sentences), producing the rest of text.

[1] Inside the club we moved straight for the bar. [2] Devlin ordered a beer for himself and a glass of my favorite wine for me. [3] I love that I didn't have to tell him what I wanted. [4] He knew me well and always thought about what I wanted or needed, in and out of bed.

Figure 2: Bridging task: given [1] and [4] sentences, guessing [2,3] sentences (red, underlined).

Figure 2 shows our bridging task. It requires a generation of masked sentences in the middle of a paragraph given the first and the last sentences. If only the first sentence is given, the generation can be too divergent. The existence of the last sentence makes the generation more coherent and converged to some point.

We evaluate it with one hard and one soft automatic metrics: METEOR (M) (Banerjee and Lavie, 2005) and VectorExtrema (VE) (Liu et al., 2016) by calculating cosine similarity of averaged word embeddings (Pennington et al., 2014), and human performance.

### 4.3 Models and Setup

We compare various baseline seq2seq models which encode the context; a concatenated first and last sentences, and decode the intermediate words: **S2S** is attentional seq2seq model (Bahdanau et al., 2014), and **HS2S**: is a hierarchical version of the S2S by combining two baselines: HRNN (Lin et al., 2015) hierarchically models sequence of words and sentences, and HRED (Serban et al., 2017; Sordoni et al., 2015) encodes the given context and decodes the words. **FlowNet (delta/disc.)** is our proposed language model with delta and discourse relations, respectively.

We find the best hyper-parameters on validation set using grid search. Here are the final parameters used: 32 for batch size, 25 for maximum sentence length, 300 for word embedding size initialized by GloVe (Pennington et al., 2014), 1 LSTM layer (Hochreiter and Schmidhuber, 1997) with 512 size, clipping by 0.25, 0.2 learning rate and 0.5 decay rate with Adagrad (Duchi et al., 2011) optimizer, and 50,000 for the vocabulary size. The total number of distinct discourse relations is 44.

### 4.4 Results

| | Papers | | SciFi | | Fantasy | |
|---|---|---|---|---|---|---|
| | **M** | **VE** | **M** | **VE** | **M** | **VE** |
| **S2S** | 3.7 | 56.3 | 3.5 | 71.0 | 3.3 | 66.3 |
| **HS2S** | 3.7 | 54.7 | 3.4 | **73.0** | 3.0 | 69.7 |
| **FlowNet** (delta) | 3.1 | **58.5** | 3.6 | 69.7 | 3.6 | **73.9** |
| **FlowNet** (disc.) | **4.0** | 57.2 | **4.2** | 70.3 | **3.9** | 71.8 |

Table 2: Performance on bridging task. **M**ETEOR and **V**ector**E**xtrema are used. The higher the better.

In Table 2, both discourse and delta driven FlowNet outperform the baseline models across most of the metrics except for VecterExtrema on SciFi. Especially, as the number of training size increases (Papers<<SciFi<Fantasy), the improvements gained from the FlowNet become bigger. This is probably because the model learns more information of the (discourse or latent) relations from the larger data.

| | M | VE |
|---|---|---|
| SUBTRACT | 3.35 | **67.20** |
| ADD | **3.45** | 65.35 |
| MLP | 3.32 | 62.97 |

Table 3: Comparison of different delta functions.

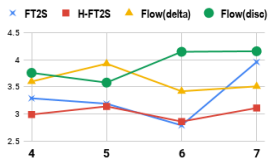


Figure 3: Comparison of paragraph lengths. Best viewed in color.

Table 3 shows performance comparison among different delta operations: SUBTRACT, ADD, and

| |
|---|
| **FIRST**: Satyrs never wear armor, including helmets, Newel began, using his hands expressively.<br>**LAST**: Anyhow, as we actors were laying siege, a big chunk of the battlement dislodged from atop the tower. |
| **REF**: [M1] "But years ago I was in a play, and the helm was part of my costume. [M2] During the big battle scene, a few of us were assailing a castle. [M3] We had quite a set. [M4] The main tower must have been fifteen feet tall, fashioned from real stone.<br>**HUMAN**: [M1] Actually he needed to wear any protectors to prevent him from a big accident. [M2] We planned to make a prank cam to make him wear those always. [M3] "I have a good idea," Newel kept talking continuously. [M4] "Let's play a role like we are under the attack. |
| **S2S**: [M1] he's a good man [M2] the UNK, the one who's a man who's a man and the other [M3] and the other, the one who 's a good friend [M4] he's a good man<br>**HS2S**: [M1] i'm not sure that," he said [M2] i'm not sure that i'm not sure [M3] i'm not sure that i'm not a fool [M4] "i'm not sure that," he said<br>**FLOWNET (DELTA)**: [M1] he's a good man [M2] i'm not sure what to do [M3] i'm not sure that i'm not going to be a vampire [M4] he's a good man<br>**FLOWNET (DISC.)**: [M1] perhaps they were not quite good, but he was not a master, and they were the most powerful [M2] the only way to do not like a little, but i' d been in the world [M3] "you're right," he said "i am not a fool you're here [M4] you're going to be a bit more than the other |

Table 4: An example paragraph and predicted texts in `Fantasy` dataset. Given **FIRST** and **LAST** sentences, the models generate middle sentences (e.g., [M1] → [M2]..). **REF** and **HUMAN** are reference middle sentences and sentences written by human annotator, respectively. Please find more examples in the appendix.

MLP which is a multi-layer perceptron network. All scores are macro-averaged across datasets. While ADD shows good performance on METEOR, SUBTRACT does on the soft metric (i.e., VecExt), indicating that subtraction can help the model capture the better semantics than the other functions. Figure 3 shows how performance changes on `Fantasy` as the paragraph lengths increase. Both of FlowNet achieve more improvements when generating longer paragraphs. Especially, discourse relations achieve the best performance at length 6 and 7.

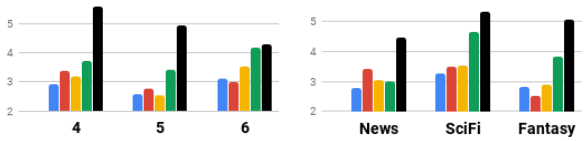


(a) By paragraph lengths (b) By domains

Figure 4: Comparison (METEOR) with human performance (black bars): S2S (blue), HS2S (red), Flow:delta (yellow), and Flow:disc. (green). Best viewed in color.

We conduct a comparison with human performance (See Figure 4). We randomly choose 100 samples per dataset and per paragraph length and ask an annotator to perform the bridging task on the final 1,000 samples. Human outperforms the models by large margins. FlowNet with discourse relations outperforms the FlowNet with latent relations and other baselines by a large margin. As the paragraph length increases or more data is trained, discourse relations become more useful.

Table 4 shows an example paragraph with text produced by the models as well as reference and human annotation. Given only the partial context (i.e., first and last sentences), bridging task is very challenging even for human. The reference sentences and human annotations are semantically very different indeed. Among the latent models, FlowNet (delta) produces more coherent flow of text compared to S2S and HS2S. Surprisingly, FlowNet (discourse) enables generating more diverse sentences with a bit of coherence, because each sentence is generated based on the representation conditioned on the predicted RST discourse relation.

## 5 Conclusion and Discussion

We explore two forms of inter-sentential relations: linguistic relation such as discourse relations and a latent representation learned from the text. The proposed models for both relations achieve significant improvements over the baselines on partial paragraph generation task.

Despite the empirical effectiveness and difference between the linguistic and latent relations, they are not directly aligned for comparison. A potential direction for future study is to directly couple them together and see whether one form contains the other, or vice versa. Another direction is to check their effectiveness on top of the recent pre-trained language models.

## Acknowledgements

We also thank Jason Weston, Dan Jurafsky, and anonymous reviewers for their helpful comments.

## References


Douglas E Appelt. 1982. Planning natural-language utterances to satisfy multiple goals. Technical report, SRI INTERNATIONAL MENLO PARK CA ARTIFICIAL INTELLIGENCE CENTER.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Donn Byrne. 1979. *Teaching writing skills*. Longman.

Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. 2016. Hierarchical multiscale recurrent neural networks. *CoRR*, abs/1609.01704.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Hercules Dalianis and Eduard Hovy. 1996. Aggregation in natural language generation. In *Trends in Natural Language Generation An Artificial Intelligence Perspective*, pages 88–105. Springer.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. Creating training corpora for nlg micro-planning. In *55th annual meeting of the Association for Computational Linguistics (ACL)*.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61:65–170.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9:1735–1780.

Eduard H. Hovy. 1985. Integrating text planning and production in generation. In *IJCAI*.

Eduard H Hovy. 1990. Pragmatics and natural language generation. *Artificial Intelligence*, 43(2):153–197.

Eduard H Hovy. 1991. Approaches to the planning of coherent text. In *Natural language generation in artificial intelligence and computational linguistics*, pages 83–102. Springer.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *CoRR*, abs/1508.01991.

Yangfeng Ji and Jacob Eisenstein. 2014. Representation learning for text-level discourse parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 13–24.

Dongyeop Kang, Waleed Ammar, Bhavana Dalvi, Madeleine van Zuylen, Sebastian Kohlmeier, Eduard Hovy, and Roy Schwartz. 2018a. A dataset of peer reviews (peerread): Collection, insights and nlp applications. In *Proceedings of NAACL-HLT*.

Dongyeop Kang, Varun Gangal, Ang Lu, Zheng Chen, and Eduard Hovy. 2017. Detecting and explaining causes from text for a time series event. In *Conference on Empirical Methods on Natural Language Processing*.

Dongyeop Kang, Tushar Khot, Ashish Sabharwal, and Eduard Hovy. 2018b. Adventure: Adversarial training for textual entailment with knowledge-guided examples. In *The 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, Melbourne, Australia.

John D. Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

Jiwei Li, Minh-Thang Luong, and Daniel Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. In *ACL*.

Rui Lin, Shujie Liu, Muyun Yang, Mu Li, Ming Zhou, and Sheng Li. 2015. Hierarchical recurrent neural network for document modeling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 899–907.

Chia-Wei Liu, Ryan Lowe, Iulian V Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *arXiv preprint arXiv:1603.08023*.

William C Mann and Sandra A Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text-Interdisciplinary Journal for the Study of Discourse*, 8(3):243–281.

Kathleen R McKeown. 1985. Discourse strategies for generating natural-language text. *Artificial Intelligence*, 27(1):1–41.

Johanna D Moore and Cécile L Paris. 1993. Planning text for advisory dialogues: Capturing intentional and rhetorical information. *Computational linguistics*, 19(4):651–694.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Ehud Reiter and Robert Dale. 2000. *Building natural language generation systems*. Cambridge university press.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*, pages 3295–3301.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 553–562. ACM.

Judith A. Swan. 2002. The science of scientific writing. In *Book*.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Silvan S Tomkins. 1978. Script theory: Differential magnification of affects. In *Nebraska symposium on motivation*. University of Nebraska Press.

Qingyun Wang, Xiaoman Pan, Lifu Huang, Boliang Zhang, Zhiying Jiang, Heng Ji, and Kevin Knight. 2018. Describing a knowledge base. In *CoRR*, volume abs/1809.01797.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

R Michael Young and Johanna D Moore. 1994. Dpocl: A principled approach to discourse planning. In *Proceedings of the Seventh International Workshop on Natural Language Generation*, pages 13–20. Association for Computational Linguistics.

Biao Zhang, Deyi Xiong, Jinsong Su, Qun Liu, Rongrong Ji, Hong Duan, and Min Zhang. 2016. Variational neural discourse relation recognizer. In *EMNLP*.

Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 19–27.

# Event Causality Recognition Exploiting Multiple Annotators' Judgments and Background Knowledge

**Kazuma Kadowaki**[§‡], **Ryu Iida**[§¶], **Kentaro Torisawa**[§¶],
**Jong-Hoon Oh**[§], and **Julien Kloetzer**[§]

[§]Data-driven Intelligent System Research Center (DIRECT),
National Institute of Information and Communications Technology (NICT)
[‡]Advanced Technology Laboratory, The Japan Research Institute, Limited
[¶]Graduate School of Science and Technology, Nara Institute of Science and Technology
{kadowaki,ryu.iida,torisawa,rovellia,julien}@nict.go.jp

## Abstract

We propose new BERT-based methods for recognizing event causality such as "smoke cigarettes" → "die of lung cancer" written in web texts. In our methods, we grasp each annotator's policy by training multiple classifiers, each of which predicts the labels given by a single annotator, and combine the resulting classifiers' outputs to predict the final labels determined by majority vote. Furthermore, we investigate the effect of supplying background knowledge to our classifiers. Since BERT models are pre-trained with a large corpus, some sort of background knowledge for event causality may be learned during pre-training. Our experiments with a Japanese dataset suggest that this is actually the case: Performance improved when we pre-trained the BERT models with web texts containing a large number of event causalities instead of Wikipedia articles or randomly sampled web texts. However, this effect was limited. Therefore, we further improved performance by simply adding texts related to an input causality candidate as background knowledge to the input of the BERT models. We believe these findings indicate a promising future research direction.

## 1 Introduction

Event causality, such as "smoke cigarettes" → "die of lung cancer," is critical knowledge for NLP applications such as machine reading (Rajpurkar et al., 2016). For the task of recognizing event causality written in web texts, we propose new BERT-based methods that exploit independent labels in a gold dataset provided by multiple annotators. In the creation of the dataset we used (Hashimoto et al., 2014), three annotators independently labeled the data and the final labels were determined by majority vote. In the previous work, each annotator's independent judgments were ignored, but in our proposed method, we exploit each annotator's judgments in predicting the majority vote labels.

The dataset we used had a reasonable degree of inter-annotator agreement (Fleiss' Kappa value was 0.67), but a discrepancy remained among the annotators. Despite this discrepancy, we assume that their judgments are more or less consistent and that we can improve performance by training multiple classifiers, each from the labels provided by an individual annotator to grasp her/his policy, and by combining the resulting outputs of these classifiers.

Researchers have studied how to exploit the differences between the behaviors of annotators and crowd workers to improve the quality of gold datasets (Snow et al., 2008; Zaidan and Callison-Burch, 2011; Zhou et al., 2012; Jurgens, 2013; Plank et al., 2014; Jamison and Gurevych, 2015; Felt et al., 2016; Li et al., 2017). In an attempt that resembles ours, one study (Jamison and Gurevych, 2015) successfully improved the performance of several NLP tasks by computing the agreement ratio of each training instance and using only those instances with high agreement. Another work (Plank et al., 2014) improved part-of-speech tagging by measuring the inter-annotator agreement on a small number of sampled data and incorporating this value during training via a modified loss function. However, neither of them directly used each annotator's judgments, as we did in this work.

As another research direction, we also investigate how to appropriately exploit *background knowledge*. In previous work, text fragments such as binary patterns (e.g., "$A$ causes $B$") and texts expressing causalities (e.g., "He died due to lung cancer") retrieved from large corpora were given to causality recognizers as background knowledge (Hashimoto et al., 2014; Kruengkrai et al., 2017), as well as association among words (Torisawa,



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5816–5822,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

2006; Riaz and Girju, 2010; Do et al., 2011), semantic polarities (Hashimoto et al., 2012), answers obtained from a web-based open-domain why-QA system and other causality related texts (Kruengkrai et al., 2017), and causality-related word embeddings (Xie and Mu, 2019).

In this work, we investigate whether BERT (Devlin et al., 2019) (especially its pre-training) enables novel ways to exploit background knowledge. Our assumption is that if a BERT model is pre-trained using a large amount of causality-rich texts, it can learn some sort of background knowledge from the text. If the pre-training is adequately performed, background knowledge in the form of text fragments and a special mechanism for dealing with them might become obsolete. Our experimental results show that a BERT model pre-trained with causality-rich texts achieved significantly better performance than models using Wikipedia articles or randomly sampled web texts, both of which can be viewed as texts that do not specifically focus on causality. But the BERT model does not seem to sufficiently capture background knowledge, at least in our task setting. Further improvement is possible by simply concatenating, to an input causality candidate, text fragments related to it as background knowledge.

In our experiments, we show that our best method significantly outperformed a state-of-the-art method (Kruengkrai et al., 2017) by about 5% in average precision.

## 2 Proposed Method

We propose three BERT-based methods for event-causality recognition, and show an overview of them in Figure 1. All of the methods take input matrix $\mathbf{x}$, which represents an input causality candidate such as "smoke cigarettes" $\rightarrow$ "die of lung cancer," and obtain each annotator's labels (either *ProperCausality* or *NonProperCausality*), which are denoted by $y^A$, $y^B$, and $y^C$ for three annotators $A$, $B$, and $C$, respectively. These are used for predicting final labels $y^{\mathrm{MV}}$, which are determined by majority vote. Here, we assume that the dataset was labeled by three annotators, $A$, $B$ and $C$, but extending the methods to deal with an arbitrary number of annotators is straightforward.

The proposed methods compute the probability $P(y^{\mathrm{MV}}|\mathbf{x})$ that a causality candidate represented by $\mathbf{x}$ expresses a proper event causality. We regard the candidate as proper if and only if $P(y^{\mathrm{MV}}|\mathbf{x}) > \theta$ ($\theta = 0.5$).

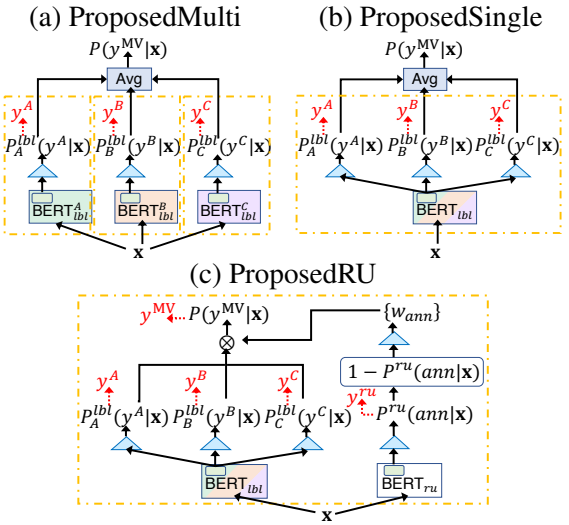


Figure 1: Proposed architectures, where red symbols (e.g., $y^A$) stand for gold labels used in training.

$$P(y^{\mathrm{MV}}|\mathbf{x}) = \sum_{ann} w_{ann} P^{lbl}_{ann}(y^{ann}|\mathbf{x}) \quad (1)$$

$$P^{lbl}_{ann}(y^{ann}|\mathbf{x}) = \mathrm{softmax}(\mathbf{W}_{ann}\mathrm{BERT}_{lbl}(\mathbf{x})) \quad (2)$$

$$P^{ru}(ann|\mathbf{x}) = \mathrm{softmax}(\mathbf{W}_D\mathrm{BERT}_{ru}(\mathbf{x})) \quad (3)$$

$$w_{ann} = \mathrm{softmax}_{ann}(1 - P^{ru}(ann|\mathbf{x})) \quad (4)$$

$\mathbf{W}_D \in \mathbb{R}^{3\times d}$ and $\mathbf{W}_{ann} \in \mathbb{R}^{2\times d}$ are trainable matrices, where $d$ is size of hidden state in BERT models.

Table 1: Equations used in ProposedRU

**ProposedMulti:** The outputs of this method are achieved by an ensemble of three classifiers, each of which is independently trained (or, more precisely, fine-tuned) with the judgments of one of three annotators to mimic her/his judgments. The architecture of this method is shown in Figure 1(a) and consists of three pairs of a BERT model and a subsequent softmax layer, where each pair computes the probability $P^{lbl}_{ann}(y^{ann}|\mathbf{x})$ of labels $y^{ann}$ of each annotator $ann \in \{A, B, C\}$, in the same manner as Equation (2) in Table 1. Probability $P(y^{\mathrm{MV}}|\mathbf{x})$ of final label $y^{\mathrm{MV}}$ is the average of $P^{lbl}_{ann}(y^{ann}|\mathbf{x})$.

**ProposedSingle:** This method uses multi-task learning in which each task corresponds to predicting labels given by one of the three annotators. The architecture of this method is shown in Figure 1(b) and consists of a single BERT model with three softmax layers, where the output of each softmax layer corresponds to an annotator's label. $P(y^{\mathrm{MV}}|\mathbf{x})$

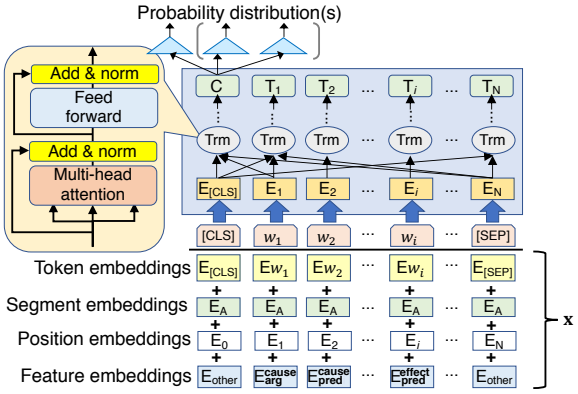


Figure 2: Proposed BERT architecture and its input $\mathbf{x}$

is computed in the same way as in **Proposed-Multi**.

**ProposedRU:** This method uses the **Proposed-Single** architecture, which consists of a BERT model ($\text{BERT}_{lbl}$ in Figure 1(c)) and three softmax layers, to compute probability $P^{lbl}_{ann}(y^{ann}|\mathbf{x})$ for annotator $ann$. We also add another BERT model ($\text{BERT}_{ru}$) and its subsequent softmax layer to the architecture to assign a lower weight to the predictions of an annotator who is likely to disagree with the majority vote. The entire computation is done using the equations in Table 1. Since each causality candidate is independently labeled by three annotators, at most one annotator disagrees with the majority vote label. To identify that annotator, $\text{BERT}_{ru}$, along with the softmax layer, estimates the probability $P^{ru}(ann|\mathbf{x})$ that annotator $ann$ disagrees with the majority vote. We call this probability the *relative unreliability* of $ann$ (Equation (3)). Instead of averaging $P^{lbl}_{ann}(y^{ann}|\mathbf{x})$ as in **ProposedSingle**, **ProposedRU** uses the weighted sum of $P^{lbl}_{ann}(y^{ann}|\mathbf{x})$ to predict the final label (Equation (1)). Weight $w_{ann}$ is computed from $P^{ru}(ann|\mathbf{x})$ to consider the relative unreliability of $ann$ (Equation (4))[1].

Representation $\mathbf{x}$ of the input causality candidate is computed from the entire sentence (e.g., "He smoked cigarettes and died of lung cancer caused by them") that contains a pair of cause phrase ("smoke cigarettes") and effect phrase ("died of lung cancer"). $\mathbf{x}$ consists of the token, position, and segment embeddings of each word (Devlin et al., 2019). We distinguish the words in the cause and effect candidates from the other words by giving them *feature embeddings*; the cause argument, the cause predicate, the effect argument, the effect predicate, and the other words are all given a different feature embedding vector that is randomly initialized (see Figure 2).

$$L = \alpha L^{\text{MV}} + \beta \sum_{ann} L^{lbl}_{ann} + \gamma L^{ru} \quad (5)$$

$$L^{\text{MV}} = -\log P(y^{\text{MV}}|\mathbf{x}) \quad (6)$$

$$L^{lbl}_{ann} = -\log P^{lbl}_{ann}(y^{ann}|\mathbf{x}) \quad (7)$$

$$L^{ru} = \begin{cases} 0 & \text{if } y^A = y^B = y^C \\ -\log P^{ru}(ann|\mathbf{x}) & \text{otherwise} \end{cases} \quad (8)$$

Table 2: Loss functions used in ProposedRU

Note that we followed the fine-tuning scheme presented in Devlin et al. (2019) in the training of **ProposedMulti** and **ProposedSingle**. For training **ProposedRU**, we designed a special loss function $L$ in Equation (5) in Table 2 as the weighted sum of $L^{\text{MV}}$, $L^{lbl}_{ann}$, and $L^{ru}$, each of which is a loss function for $P(y^{\text{MV}}|\mathbf{x})$, $P^{lbl}_{ann}(y^{ann}|\mathbf{x})$, and $P^{ru}(ann|\mathbf{x})$, respectively. Here, $\alpha$, $\beta$, and $\gamma$ are hyper-parameters ($\alpha + \beta + \gamma = 1.0$).

The BERT model in our methods is pre-trained from scratch using causality-rich texts to investigate whether such BERT models can learn background knowledge during their pre-training. We used 19,567,386 sentences from 2,799,079 passages extracted from four billion web pages, where each passage consists of seven sentences and includes at least one event causality detected by a CRF-based causality recognizer[2] (Oh et al., 2013) (3,046,619 event causalities were detected in the passages).

We also introduce **ProposedRU+BK**, which integrates the background knowledge used in the previous work into **ProposedRU**. As input, **ProposedRU+BK** is given a *pseudo sentence*, which is the concatenation of the original input sentence and Kruengkrai et al. (2017)'s text fragments embodying background knowledge[3] along with a

[1] Note that this method is applicable only for the problem setting in which the final binary-class label is determined by using three annotators' labels. Some extension is needed for computing relative unreliability $P^{ru}(ann|\mathbf{x})$ for an arbitrary number of annotators by considering the probability that each annotator agrees with the majority.

[2] The recognizer was applied only to passages that were extracted by specifically focusing on clue terms such as "because." Event causalities recognized this way are a different type than those under focus in this work.

[3] We used all three types of text fragments from Krueng-

separator to compute input representation $\mathbf{x}$.

Here, we hypothesized that BERT models could learn some sort of background knowledge by using causality-rich texts in pre-training. To investigate whether this is true, we pre-trained several BERT models, each using either causality-rich texts or general texts, and then fine-tuned each of the models using the same architecture as used in **ProposedRU**. We introduce **ProposedRU$_{\text{wiki}}$** and **ProposedRU$_{\text{web}}$**, in which the pre-training is performed from scratch using a single corpus, that is, either Wikipedia articles (19,567,381 sentences retrieved in August 2018) or randomly sampled web texts (19,567,396 sentences in 1,990,472 web pages randomly sampled from the four billion web pages), respectively. The fine-tuning is then done as in **ProposedRU**.

In contrast to the above single-step pre-training, we also attempted to start from the pre-trained model for **ProposedRU$_{\text{web}}$**, which was pre-trained using general web text, and then additionally pre-train it with causality-rich texts to make the model suitable for causality-event recognition[4]. More precisely, to focus on the event causality part described in each passage, we extracted the pairs of cause and effect parts detected by Oh et al. (2013)'s CRF-based causality recognizer and used them as a causality-rich corpus while assuming that the cause and effect are sentence pairs for the next-sentence prediction task (9,783,691 pairs or 19,567,382 sentences). This model is called **ProposedRU$_{\text{web+pair}}$** hereafter.

For comparison, we also introduce **ProposedRU$_{\text{web+web}}$**, which uses another set of web sentences randomly sampled from four billion web pages, as a corpus of the general but not causality-rich texts. The size is the same as that of the cause-effect pairs used for **ProposedRU$_{\text{web+pair}}$**.

## 3 Experiments

### 3.1 Settings

We used the datasets for event-causality recognition in Japanese of Hashimoto et al. (2014). They regard causality candidate $A \rightarrow B$ proper if and only if "if $A$ happens, the probability of $B$ increases." To annotate this dataset, three annotators used their own judgment independently, and each annotator's individual labels are included in the datasets. Table 3 shows their statistics[5].

| Data | #Instances | #True causalities |
|---|---|---|
| Training | 107,068 | 8,986 |
| Development | 23,602 | 3,759 |
| Test | 23,650 | 3,647 |

Table 3: Statistics of datasets

(d) VanillaBERT

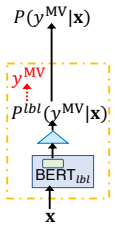


We trained a single BERT model and its softmax layer with majority vote labels.

(e) MajorityMulti

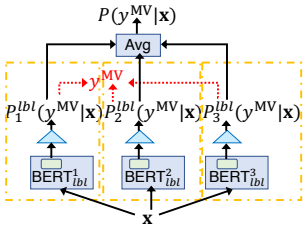


The architecture is identical to **ProposedMulti**, but the training was performed with majority vote labels.

(f) MajoritySingle

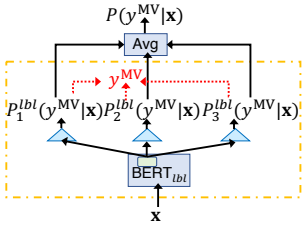


The architecture is identical to **ProposedSingle**, but the training was performed with majority vote labels.

Figure 3: Baseline architectures

As baselines, we used the state-of-the-art method of Kruengkrai et al. (2017), **MCNN**, which uses a CNN and exploits text fragments retrieved from four billion web pages as background knowledge, and three BERT-based methods, **VanillaBERT**, **MajorityMulti**, and **MajoritySingle**, which use only majority vote labels (Figure 3). These BERT-based baselines used the same pre-trained BERT model (pre-trained with causality-rich texts) as in our proposed methods.

As the parameter settings of the pre-training for all of the BERT models, we followed the BERT$_{\text{BASE}}$ settings[6] (Devlin et al., 2019) except for a batch size of 50. For **ProposedRU$_{\text{web+pair}}$** and **ProposedRU$_{\text{web+web}}$**, we additionally pre-

krai et al. (2017) (i.e., short binary patterns, why-QA system's answers, and sentences with clue terms). Every text fragment was extracted from four billion Japanese web pages.

[4] This approach is recommended by Google's BERT implementation (https://github.com/google-research/bert) for computational efficiency.

[5] The number of training instances in Hashimoto et al. (2014)'s dataset was 112,098, but we excluded the duplicates in it and used the resulting 107,068 instances for training each method.

[6] 12-layers, 768 hidden states, 12 heads, and training for one million steps with a warmup rate of 1% using an Adam optimizer with a learning rate of 1e-4.

| Model | R | P | F | Avg.P |
|---|---|---|---|---|
| MCNN | 40.2 | **61.1** | 48.5 | 55.1 |
| VanillaBERT | 63.6 | 49.7 | 55.8* | 54.2 |
| MajorityMulti | 61.5 | 51.8 | 56.2* | 55.7 |
| MajoritySingle | 48.3 | 56.8 | 52.2* | 54.4 |
| ProposedMulti | 63.9 | 51.3 | 56.9* | 56.7 |
| ProposeSingle | 62.8 | 52.7 | 57.3*† | 57.1 |
| ProposedRU | **64.0** | 52.0 | **57.4*** | **57.4** |

'*' stands for significant improvement over MCNN and '†' means that over MajorityMulti (McNemar test, $p = 0.05$).

Table 4: Results of event-causality recognition

| Model | R | P | F | Avg.P |
|---|---|---|---|---|
| ProposedRU$_{\text{wiki}}$ | 57.4 | 49.6 | 53.2$\star$ | 53.3 |
| ProposedRU$_{\text{web}}$ | 59.0 | 50.9 | 54.6$\star$ | 54.5 |
| ProposedRU | 64.0 | 52.0 | 57.4 | 57.4 |
| ProposedRU$_{\text{web+web}}$ | 62.5 | 49.0 | 54.9$\star$ | 54.8 |
| ProposedRU$_{\text{web+pair}}$ | 64.3 | 48.2 | 55.1$\star$ | 55.3 |
| ProposedRU+BK | **67.4** | **52.3** | **58.9** | **59.9** |

'$\star$' stands for significant difference from ProposedRU.

Table 5: Results of ProposedRU and its variants

trained the models for 0.3 million steps with a learning rate of 2e-5. As for the fine-tuning, we chose the combination of epochs {1, 2, 3} and learning rates {5e-6, 1e-5, 2e-5, 3e-5, 4e-5, 5e-5} that achieved the best average precision on 50% of the development dataset, as done in Kruengkrai et al. (2017). For **ProposedRU** and its variants, we additionally optimized the $\alpha$, $\beta$, and $\gamma$ hyper-parameters in the same way.

### 3.2 Results

The recall, precision, F1-score, and average precision of each method on the majority vote labels of the test dataset are presented in Table 4. **ProposedRU** achieved the best F1-score and average precision. All of the proposed methods (**ProposedMulti**, **ProposedSingle**, and **ProposedRU**) outperformed all of the baseline methods (**MCNN**, **VanillaBERT**, **MajorityMulti**, and **MajoritySingle**) in average precision. This suggests that using each annotator's labels produced a positive effect. The F1-scores of **ProposedRU** and **ProposedSingle** were not significantly different (McNemar test, $p = 0.05$).

Table 5 shows the results of our investigation into the pre-training of some sort of *background knowledge*. Among the methods to be compared in Table 5, the first three methods (i.e., **ProposedRU$_{\text{wiki}}$**, **ProposedRU$_{\text{web}}$**, and **ProposedRU**) utilized a single-step pre-training from scratch, whereas the next two methods (**ProposedRU$_{\text{web+web}}$** and **ProposedRU$_{\text{web+pair}}$**) performed additional pre-training before the fine-tuning. The results show that pre-training using causality-rich texts contributes to further performance improvement than that using general texts, such as Wikipedia articles and random web texts (see **ProposedRU** vs. **ProposedRU$_{\text{wiki}}$**, **ProposedRU** vs. **ProposedRU$_{\text{web}}$**, and **ProposedRU$_{\text{web+pair}}$** vs. **ProposedRU$_{\text{web+web}}$**). In short, we can say that BERT might learn some sort of background knowledge from causality-rich texts. An interesting point is that, although the amount of texts used for **ProposedRU** is almost the same as that of the second-step causality-rich pre-training for **ProposedRU$_{\text{web+pair}}$**, the performance differs considerably (about 2% difference). This suggests that the design of the pre-training steps is not straightforward and thus merits further research.

We further evaluated **ProposedRU+BK**, in which text fragments embodying background knowledge are concatenated to the input sentence as explained in Section 2. Table 5 shows that **ProposedRU+BK** improved the average precision over **ProposedRU** by about 2.5% (i.e., **ProposedRU+BK** significantly outperformed the state-of-the-art method, **MCNN**, by about 5%), suggesting that background knowledge in the form of text fragments is still useful, at least in our current experimental setting. However, the usefulness might be lost when a model is appropriately pre-trained with a larger amount of texts that covers even more background knowledge.

## 4 Conclusion

This paper proposed BERT-based methods for recognizing event causality that exploit each annotator's independent judgments. By using each annotator's judgments, we showed that even a simple multi-task learning approach or an ensemble method improved performance in our experiments. Our best-performing method significantly outperformed the state-of-the-art method by about 5% in average precision. We also confirmed that the pre-trained BERT model learns some sort of background knowledge for event causality from causality-rich texts, although text fragments embodying background knowledge remain useful, at least in our current setting. As future work, we are examining the possibility of using an adversarial learning framework (Goodfellow et al., 2014), which was recently used in the why-QA task (Oh et al., 2019) for causality recognition.

## References


Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Quang Do, Yee Seng Chan, and Dan Roth. 2011. Minimally supervised event causality identification. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 294–303.

Paul Felt, Eric Ringger, and Kevin Seppi. 2016. Semantic annotation aggregation with conditional crowdsourcing models and word embeddings. In *Proceedings of the 26th International Conference on Computational Linguistics*, pages 1787–1796.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, pages 2672–2680.

Chikara Hashimoto, Kentaro Torisawa, Stijn De Saeger, Jong-Hoon Oh, and Jun'ichi Kazama. 2012. Excitatory or inhibitory: A new semantic orientation extracts contradiction and causality from the web. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 619–630.

Chikara Hashimoto, Kentaro Torisawa, Julien Kloetzer, Motoki Sano, István Varga, Jong-Hoon Oh, and Yutaka Kidawara. 2014. Toward future scenario generation: Extracting event causality exploiting semantic relation, context, and association features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 987–997.

Emily Jamison and Iryna Gurevych. 2015. Noise or additional information? Leveraging crowdsource annotation item agreement for natural language tasks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 291–297.

David Jurgens. 2013. Embracing ambiguity: A comparison of annotation methodologies for crowdsourcing word sense labels. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 556–562.

Canasai Kruengkrai, Kentaro Torisawa, Chikara Hashimoto, Julien Kloetzer, Jong-Hoon Oh, and Masahiro Tanaka. 2017. Improving event causality recognition with multiple background knowledge sources using multi-column convolutional neural networks. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, pages 3466–3473.

Jiyi Li, Yukino Baba, and Hisashi Kashima. 2017. Hyper questions: Unsupervised targeting of a few experts in crowdsourcing. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1069–1078.

Jong-Hoon Oh, Kazuma Kadowaki, Julien Kloetzer, Ryu Iida, and Kentaro Torisawa. 2019. Open-domain why-question answering with adversarial learning to encode answer texts. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4227–4237.

Jong-Hoon Oh, Kentaro Torisawa, Chikara Hashimoto, Motoki Sano, Stijn De Saeger, and Kiyonori Ohtake. 2013. Why-question answering using intra- and inter-sentential causal relations. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1733–1743.

Barbara Plank, Dirk Hovy, and Anders Søgaard. 2014. Learning part-of-speech taggers with inter-annotator agreement loss. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 742–751.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Mehwish Riaz and Roxana Girju. 2010. Another look at causality: Discovering scenario-specific contingency relationships with no supervision. *2010 IEEE Fourth International Conference on Semantic Computing*, pages 361–368.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Ng. 2008. Cheap and fast – but is it good? Evaluating non-expert annotations for natural language tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 254–263.

Kentaro Torisawa. 2006. Acquiring inference rules with temporal constraints by using Japanese coordinated sentences and noun-verb co-occurrences. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 57–64.

Zhipeng Xie and Feiteng Mu. 2019. Distributed representation of words in cause and effect spaces. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7330–7337.

Omar F. Zaidan and Chris Callison-Burch. 2011. Crowdsourcing translation: Professional quality from non-professionals. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1220–1229.

Dengyong Zhou, Sumit Basu, Yi Mao, and John C. Platt. 2012. Learning from the wisdom of crowds by minimax entropy. In *Advances in Neural Information Processing Systems 25*, pages 2195–2203.

# What Part of the Neural Network Does This? Understanding LSTMs by Measuring and Dissecting Neurons

**Ji Xin, Jimmy Lin,** and **Yaoliang Yu**
David R. Cheriton School of Computer Science
University of Waterloo
{ji.xin,jimmylin,yaoliang.yu}@uwaterloo.ca

## Abstract

Memory neurons of long short-term memory (LSTM) networks encode and process information in powerful yet mysterious ways. While there has been work to analyze their behavior in carrying low-level information such as linguistic properties, how they directly contribute to label prediction remains unclear. We find inspiration from biologists and study the affinity between individual neurons and labels, propose a novel metric to quantify the sensitivity of neurons to each label, and conduct experiments to show the validity of our proposed metric. We discover that some neurons are trained to specialize on a subset of labels, and while dropping an arbitrary neuron has little effect on the overall accuracy of the model, dropping label-specialized neurons predictably and significantly degrades prediction accuracy on the associated label. We further examine the consistency of neuron-label affinity across different models. These observations provide insight into the inner mechanisms of LSTMs.

## 1 Introduction

In recent years, the application of deep learning to natural language processing (NLP) has been a success. Many consider the employment of distributed representations to be one of the reasons for deep learning's success (LeCun et al., 2015; Young et al., 2018). However, how these distributed representations encode information in deep neural networks, especially long short-term memory (LSTM) networks that are prevalent in NLP, still remains unclear (Feng et al., 2018). One of the potential ways to understand how neural networks function is to analyze the behavior of individual neurons that carry the distributed representation. While there have been a number of works that analyze low-level information stored in individual LSTM neurons, such as linguistic properties (Bau et al., 2019; Qian et al., 2016), syntax of source code (Karpathy et al., 2015), and sentiment (Radford et al., 2017), how each neuron contributes directly to the final classification layer remains unclear.

We find inspiration to analyze individual neurons of LSTMs from how biologists analyze neurons of roundworms (White et al., 1986). Biological neural systems consist of a huge number of neurons, and can react to the environment in complicated ways. Biologists start with analyzing basic components of reactions, what stimuli trigger them, and which neurons are excited during the process. To verify the relationship between stimuli, neurons, and reactions, biologists further dissect neurons which are correlated with specific basic reactions, and see if the reaction still occurs for the same stimuli.

We adopt the same methodology to study LSTMs, using a representative task in NLP: named-entity recognition (NER) (Ratinov and Roth, 2009; Lample et al., 2016). Even though the output of a neural network may be complicated, we focus on basic components of the output: whether a label is predicted or not. We feed into the neural model various input instances, and analyze the relationship between the value of each LSTM neuron and the predicted label. We quantify the sensitivity of neurons to each label, and study how label-specific information is distributed among all neurons. We discover that each individual neuron is specialized to carry information for a subset of labels, and the information of each label is only carried by a subset of all neurons. We further conduct experiments to gradually drop out individual neurons. This significantly lowers the accuracy of labels that the neuron is specialized on, while having little effect on the overall performance of the model. We also study the corre-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5823–5830,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

lation between labels, and discover some patterns that are shared among different models.

Our contributions are as follows: (1) To the best of our knowledge, we are the first to have taken this neuron-label affinity focused approach to understanding the inner workings of LSTMs. (2) We propose a novel metric to quantify such affinity, and conduct experiments to verify the validity and consistency of this metric.

## 2 Related Work

Recently, work has been done to analyze continuous representations in NLP. Shi et al. (2016) and Qian et al. (2016) analyze linguistic properties carried by representation vectors using external supervision. Bau et al. (2019) and Dalvi et al. (2019) further analyze linguistic information in individual neurons from neural machine translation representations in an unsupervised manner. For LSTMs of language models, Karpathy et al. (2015) identify individual neurons that trigger for specific information, such as bracket and sequence length, and Radford et al. (2017) discover neurons that encode sentiment information.

In computer vision, Zhou et al. (2018) analyze the relationship between individual units of a CNN and label prediction. To the best of our knowledge, however, in the field of NLP, there has been little work on analyzing the affinity between labels and neurons of recurrent networks. This paper aims to address this problem.

## 3 Model and Experiments

### 3.1 Model Selection

Named-entity recognition is a sequence labeling task. The input of the model is a sequence of words $x^{(t)}, t = 1, 2, \cdots$. Each input word has a corresponding label $z^{(t)} \in \mathcal{L}$, where $\mathcal{L}$ is the set of all labels $\{l_j\}, j = 1, 2, \cdots, m$. The label indicates whether the word is an entity or not, and if yes, which kind of entity it is.

A typical modern NER model consists of a bi-directional LSTM and a conditional random field (CRF) on top of the LSTM (Collobert et al., 2011; Huang et al., 2015). Sometimes there is also a convolutional neural network (CNN) (Ma and Hovy, 2016; Chiu and Nichols, 2016). However, the goal of this paper is not to achieve state-of-the-art performance on this task, but rather we are trying to understand the mechanisms of LSTMs. Therefore, we choose a relatively simple model (see Figure 1) for the experiments: a single layer uni-directional LSTM with a fully-connected layer on top of it.

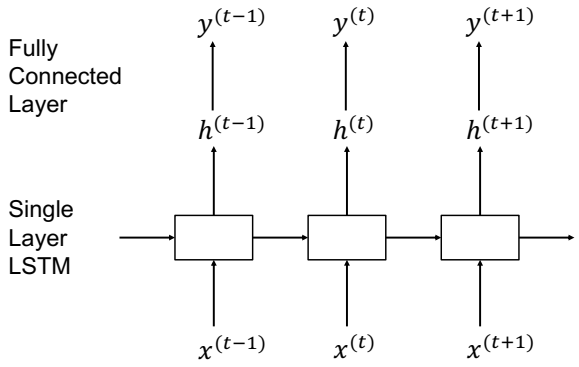


Figure 1: The model we use for this task: a single layer uni-directional LSTM with a fully-connect layer on top of the LSTM.

We denote $\mathbf{h}^{(t)} \in \mathbb{R}^n$ as the LSTM's hidden state at timestep $t$, and $\mathbf{h}_i^{(t)}$ its $i$-th entry. $\mathbf{W} \in \mathbb{R}^{n \times m}$ is the weight matrix of the fully-connected layer. The output of the entire model at timestep $t$ is therefore the vector

$$\mathbf{y}^{(t)} = \text{Softmax}(\mathbf{W}^\top \mathbf{h}^{(t)}), \quad (1)$$

where $\mathbf{y}^{(t)} \in \mathbb{R}^m$ and each entry is the predicted probability of a label in $\mathcal{L}$:

$$\Pr(z^{(t)} = l_j) = \mathbf{y}_j^{(t)} \quad (2)$$

$$= \frac{\exp\left(\mathbf{W}_{:,j}^\top \mathbf{h}^{(t)}\right)}{\sum_{j'} \exp\left(\mathbf{W}_{:,j'}^\top \mathbf{h}^{(t)}\right)}, \quad (3)$$

where $\mathbf{W}_{:,j}^\top$ is the transpose of the $j$-th column vector of the matrix $\mathbf{W}$. The final prediction $\tilde{z}^{(t)}$ is chosen as the label with greatest probability.

### 3.2 Experiment Setup

The model is trained on the CoNLL2003 (Sang and De Meulder, 2003) training dataset. Development and test sets of CoNLL2003 will be used in experiments in Section 4. In this dataset there are nine labels in total, under the BIO tagging schema. See the first row of Figure 3 for the complete set of labels.

Code for this paper is adopted from the toolkit by Yang and Zhang (2018). We set the hidden size of the LSTM to 50, since a larger hidden size does not significantly improve the results. Other hyper-parameters, such as learning rate, batch size, and drop out rate, are kept unchanged. The model is trained for 10 epochs, and we pick the checkpoint

| cell | B-PER | I-PER | B-LOC | I-LOC | B-ORG | I-ORG | B-MISC | I-MISC | O |
|---|---|---|---|---|---|---|---|---|---|
| 7 | 0.10 | -0.01 | 0.02 | 0.01 | 0.08 | 0.01 | -0.08 | 0.00 | 0.10 |
| 10 | 0.10 | 0.18 | 0.06 | 0.02 | 0.07 | 0.13 | 1.05 | -0.01 | -0.01 |
| 13 | 0.72 | 1.81 | 0.09 | 0.00 | 0.00 | 0.05 | -0.01 | 0.00 | -0.03 |
| 17 | 0.03 | 0.08 | 1.17 | 0.13 | 0.08 | 0.15 | -0.01 | 1.29 | 0.09 |
| 19 | 0.07 | 0.25 | 0.27 | -0.19 | 0.85 | -0.09 | 0.50 | 0.08 | 2.40 |
| 24 | -0.04 | 0.00 | 0.03 | 0.06 | -0.04 | -0.03 | 0.02 | 0.17 | -0.03 |
| 27 | -0.23 | 0.06 | 0.21 | -0.23 | -0.11 | 0.11 | 1.29 | 0.05 | 1.35 |
| 34 | -0.04 | 0.26 | 0.24 | 0.02 | 0.04 | -0.05 | 1.39 | 0.12 | 0.13 |
| 37 | 1.23 | 0.11 | -0.02 | -0.04 | 0.74 | -0.08 | 0.01 | 0.04 | 0.56 |
| 48 | 0.17 | 0.30 | 0.00 | -0.05 | -0.02 | 0.14 | 1.09 | 0.25 | 0.57 |

Figure 2: Sensitivity of the top ten neurons for all labels; red for positive values, blue for negative ones, and deeper colors stand for larger absolute values.

| B-PER | I-PER | B-LOC | I-LOC | B-ORG | I-ORG | B-MISC | I-MISC | O |
|---|---|---|---|---|---|---|---|---|
| 11 | 13 | 45 | 28 | 1 | 28 | 34 | 28 | 19 |
| 44 | 9 | 29 | 14 | 19 | 44 | 27 | 17 | 42 |
| 37 | 44 | 17 | 3 | 37 | 16 | 48 | 30 | 32 |
| 9 | 14 | 2 | 30 | 28 | 38 | 10 | 16 | 39 |
| 13 | 31 | 39 | 31 | 9 | 36 | 29 | 29 | 23 |
| 42 | 11 | 32 | 44 | 47 | 3 | 45 | 42 | 27 |
| 29 | 30 | 38 | 45 | 44 | 12 | 28 | 45 | 22 |
| 46 | 36 | 21 | 25 | 32 | 43 | 30 | 26 | 46 |
| 14 | 16 | 9 | 41 | 6 | 33 | 19 | 14 | 38 |
| 2 | 40 | 23 | 2 | 21 | 47 | 21 | 1 | 20 |

Figure 3: Top ten neurons in terms of importance rankings for all labels. The green and yellow labels are the ones used in the experiments described in Section 4.2.

based on the best F1 score on the development set. The chosen checkpoint has an F1 score of 86.4. For comparison, the F1 score obtained by the same toolkit using a bi-directional LSTM and a CRF is 89.5 (Yang et al., 2018).

## 4 Analyzing Neuron-Label Affinity

In this section, we first identify important neurons by quantifying the sensitivity of a neuron to a label, and then verify the quantification by neuron ablation experiments.

### 4.1 Identifying Important Neurons

A neuron of an LSTM corresponds to an entry (dimension) of $\mathbf{h}^{(t)}$. For a certain label $l_j$, we try to identify neurons that are important for its prediction in the following way.

We define the *contribution* of the $i$-th neuron to the $j$-th label at timestep $t$ as

$$u_{i,j}^{(t)} = \mathbf{W}_{i,j}\mathbf{h}_i^{(t)}. \quad (4)$$

Note that contribution is defined with the number after multiplied by $\mathbf{W}$ in the fully-connected layer. Therefore the contribution value itself is what matters, not its absolute value.

The *sensitivity* of the $i$-th neuron to the $j$-th label is further defined as

$$s_{i,j} = \mathbb{E}(u_{i,j}^{(t)}|\tilde{z}^{(t)} = l_j) - \mathbb{E}(u_{i,j}^{(t)}|\tilde{z}^{(t)} \neq l_j), \quad (5)$$

where $\mathbb{E}$ stands for taking average over $t$. This is the difference of the mean contribution over $l_j$ entity words versus other words. The higher $s_{i,j}$ is, the more sensitive the $i$-th neuron is for predicting the label $l_j$.

We compute $s_{i,j}$ for all $i$ and $j$ pairs, and the average is done over the entire development set. A part of the results is shown in Figure 2, and the full results are shown in the appendix.

From the figure we can see that information is distributed across different neurons in a highly non-uniform way:

- Each neuron has a different sensitivity to different labels. Some neurons are only sensitive to one label, e.g., neuron #10 for B-MISC; some are sensitive to multiple labels, e.g., neuron #17 for B-LOC and I-MISC; some are even not sensitive to any, e.g., neuron #7.

- For each label, there are multiple neurons that are sensitive to it, as well as multiple neurons that are not.

From these, we can come to the conclusion that the prediction of each label is based on information that is distributed among multiple, but not all, neurons. Furthermore, different types of information are distributed differently.

For each label, we further rank all neurons based on their sensitivity, and obtain an *importance ranking* for the label. The top ten neurons for each label are shown in Figure 3, and the full results are shown in the appendix.

### 4.2 Verifying the Importance of Neurons

We try to verify whether the *sensitivity* we define in the previous subsection is a valid and consistent indicator of a neuron's importance for a label.

The way to do this is to perform model evaluation on the test set,[1] while incrementally ablating neurons[2] from the model, in a certain order. If the sensitivity of neurons we obtain is valid and consistent, when we ablate neurons in the order of *importance ranking* of the label $l_j$, the performance on the test set should drop fastest for predicting $l_j$, and slower for other labels.

We choose two pairs of labels: (B-PER, B-MISC) and (B-LOC, I-ORG). In each pair, we conduct neuron ablation according to each label's

[1] Recall that we obtain the value of sensitivity only from the development set.

[2] Ablating a neuron here means setting $\mathbf{h}_i^{(t)}$ to 0.

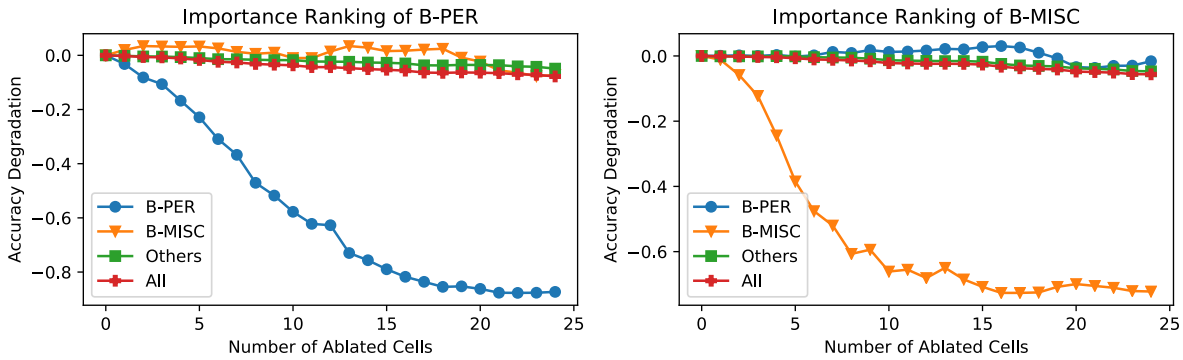


Figure 4: Ablation according to importance ranking of B-PER and B-MISC (yellow columns in Figure 3).

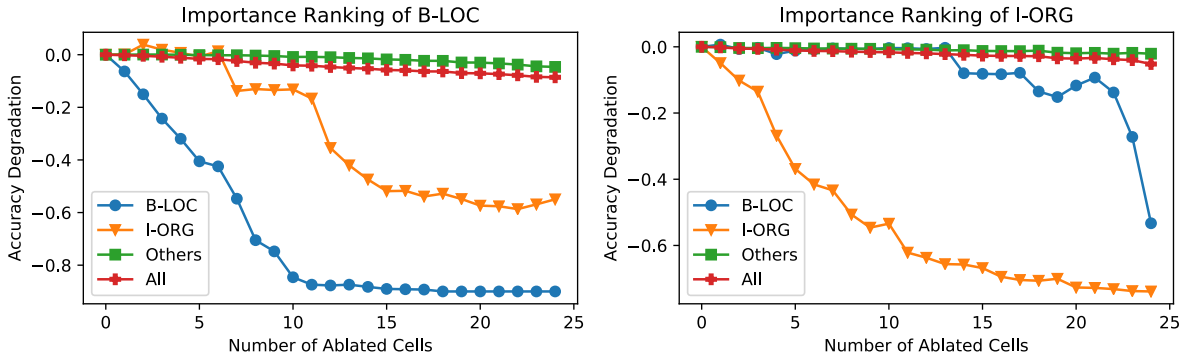


Figure 5: Ablation according to importance ranking of B-LOC and I-ORG (green columns in Figure 3).

importance ranking, and compare the model's performance for predicting each label. The results are shown in Figures 4 and 5. We only show the first half of the importance ranking, since the latter half not only is less important, but also has more overlap between different labels.

From the figures we can see that when ablating neurons according to a certain label's importance ranking, the accuracy of the label drops much faster than the other labels. The overall performance, however, remains more or less unaffected. This shows that while a single neuron can be important for a subset of labels, the overall performance is more robust to neuron ablation. This further verifies our observations from the previous subsection: information is distributed among multiple neurons in various ways. A neuron may have encoded important information for a certain label, but it is unlikely that all important information is concentrated in one neuron.

It is worth noting that a neuron can be important for multiple labels. Therefore, when ablating neurons according to one label's importance ranking, the performance for other labels may also degrade. This can be seen in the left plot of Figure 5. Neuron #38 appears in both the top ten lists of B-LOC and I-ORG (shaded boxes in Figure 3), and when it is ablated (the seventh ablated neuron), not only the performance of B-LOC, but also that of I-ORG, is compromised. The fourth ablated neuron in the right plot of Figure 5 has a similar behavior, but it is less significant, probably because this neuron is ranked seventh for B-LOC and is therefore less important than it is for I-ORG. This phenomenon is less significant in Figure 4, since the top neurons from importance rankings of B-PER and B-MISC have fewer overlaps.

### 4.3 Correlation Between Labels

Even though the distribution of information in neurons may seem arbitrary, we want to see if multiple, independently-trained models share any common traits.

In addition to the model we have used in previous sections, we train three more models with the same model architecture and hyperparameters but different random seeds. We compute neuron-label sensitivity for all four models using both develop-

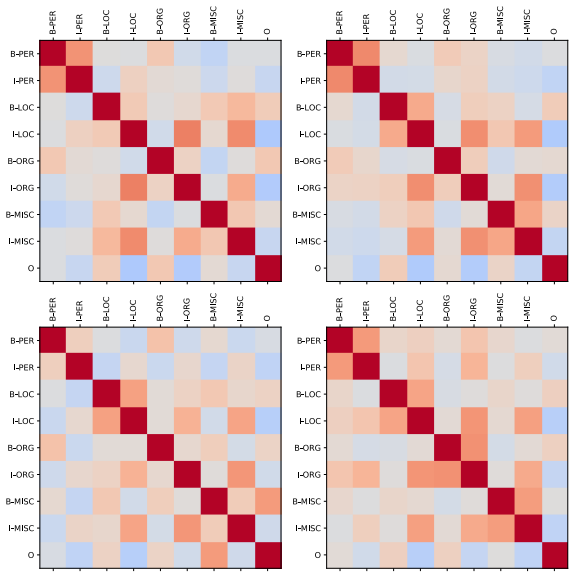


Figure 6: Correlation between all labels for four different models.

ment set and test set. For each of the models, we compute the correlation between all labels among different neurons. The sensitivity matrix for each model has 50 rows (neurons) and 9 columns (labels); see Figure 7 in the appendix for example. Correlation is computed among all rows in the matrix. The results are shown in Figure 6.

While there are differences among the four correlation plots, they share the following patterns:

- Label pairs of the form B-x and I-x (where x is PER/LOC/ORG/MISC) are generally positively correlated. We can observe some dark-red $2 \times 2$ blocks on the diagonal. Although for each trained model, it might be different neurons (i.e., neuron #) that encode information about B-x, these neurons typically also carry information about I-x.
- The label triples I-LOC, I-ORG, and I-MISC are also positively correlated.
- Label pairs of the form B-x and I-y (where x and y are different entities) are generally negatively correlated, e.g., I-PER with any of B-LOC, B-ORG, and B-MISC.
- The label O is negatively correlated with all I-x labels.

Although it remains unclear what information the neurons exactly encode, we speculate that there are at least two kinds of information, based on the observed patterns:

- Coarse-grain types of the current word. For example, whether the word is related to PER, or LOC/ORG/MISC, or O.
- Entity boundary location. If the previous prediction is O, it means the current word should be either another O or the left boundary of an entity, and thus the model should only predict O or B-x, but never I-x. Hence, I-x is negatively correlated with B-y and O.

## 5 Conclusion

In this paper, we try to understand the mechanisms of LSTMs by measuring and dissecting LSTM neurons. We discover that the prediction of each label is based on label-specific information, which is distributed among different groups of neurons. We propose a method to quantify and rank the importance of each neuron for each label, and further conduct ablation experiments to verify the validity and consistency of such importance rankings. Results show that the importance of a neuron is very different for different labels.

**Future work.** We consider the following three directions as future work. (1) While we now know how label-specific information is distributed among neurons and how important each neuron is to different labels, we can only make speculations about what the information is. It would be meaningful to study how the neurons are trained to encode label-specific information and what exactly the information is. (2) From the sensitivity figures, some neurons do not seem important to any label. It would be interesting to see what will happen if they are removed, and furthermore, how this can help model compression and hidden size selection. (3) Transformers (Vaswani et al., 2017; Devlin et al., 2019) have been increasingly popular in NLP, and it would be important to extend our work to understanding these architecture.

## Acknowledgments

This research was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada, and enabled by computational resources provided by Compute Ontario and Compute Canada.

## References


Anthony Bau, Yonatan Belinkov, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2019. Identifying and controlling important neurons in neural machine translation. In *International Conference on Learning Representations*.

Jason P. C. Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Yonatan Belinkov, Anthony Bau, and James R. Glass. 2019. What is one grain of sand in the desert? Analyzing individual neurons in deep NLP models. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)*, pages 6309–6317.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota.

Shi Feng, Eric Wallace, Alvin Grissom II, Mohit Iyyer, Pedro Rodriguez, and Jordan Boyd-Graber. 2018. Pathologies of neural models make interpretations difficult. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3719–3728, Brussels, Belgium.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Andrej Karpathy, Justin Johnson, and Li Fei-Fei. 2015. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature*, 521(7553):436.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany.

Peng Qian, Xipeng Qiu, and Xuanjing Huang. 2016. Analyzing linguistic knowledge in sequential model of sentence. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 826–835, Austin, Texas.

Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. 2017. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 147–155, Boulder, Colorado.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural MT learn source syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1526–1534, Austin, Texas.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008.

John G White, Eileen Southgate, J. Nichol Thomson, and Sydney Brenner. 1986. The structure of the nervous system of the nematode Caenorhabditis elegans. *Philosophical Transactions of the Royal Society B*, 314(1165):1–340.

Jie Yang, Shuailong Liang, and Yue Zhang. 2018. Design challenges and misconceptions in neural sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3879–3889, Santa Fe, New Mexico.

Jie Yang and Yue Zhang. 2018. NCRF++: an open-source neural sequence labeling toolkit. In *Proceedings of ACL 2018, System Demonstrations*, pages 74–79, Melbourne, Australia.

Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. 2018. Recent trends in deep learning based natural language processing. *IEEE Computational Intelligence Magazine*, 13(3):55–75.

Bolei Zhou, Yiyou Sun, David Bau, and Antonio Torralba. 2018. Revisiting the importance of individual units in CNNs via ablation. *arXiv preprint arXiv:1806.02891*.

| cell | B-PER | I-PER | B-LOC | I-LOC | B-ORG | I-ORG | B-MISC | I-MISC | O |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |
| 0 | 0.00 | 0.27 | 0.01 | 0.02 | -0.05 | 0.45 | 0.00 | 0.26 | 0.03 |
| 1 | -0.04 | 0.41 | 0.04 | 0.01 | 1.40 | 0.21 | -0.05 | 0.36 | 0.09 |
| 2 | 0.37 | 0.03 | 0.92 | 0.62 | -0.08 | 0.12 | 0.19 | 0.17 | 0.43 |
| 3 | -0.13 | 0.37 | -0.01 | 0.83 | -0.03 | 0.64 | -0.04 | 0.11 | 0.52 |
| 4 | 0.02 | -0.01 | 0.01 | 0.02 | 0.03 | 0.35 | 0.05 | 0.02 | 0.10 |
| 5 | 0.22 | 0.26 | 0.07 | 0.38 | 0.11 | -0.03 | -0.02 | 0.27 | 0.21 |
| 6 | 0.14 | 0.08 | 0.06 | 0.00 | 0.45 | 0.04 | -0.02 | -0.02 | 0.51 |
| 7 | 0.10 | -0.01 | 0.02 | 0.01 | 0.08 | 0.01 | -0.08 | 0.00 | 0.10 |
| 8 | -0.04 | 0.17 | 0.00 | 0.03 | 0.25 | 0.00 | 0.00 | 0.03 | 0.05 |
| 9 | 0.89 | 1.50 | 0.63 | 0.05 | 0.61 | 0.01 | 0.06 | -0.01 | 0.00 |
| 10 | 0.10 | 0.18 | 0.06 | 0.02 | 0.07 | 0.13 | 1.05 | -0.01 | -0.01 |
| 11 | 1.93 | 0.94 | -0.01 | 0.00 | 0.40 | 0.03 | 0.10 | 0.25 | 0.01 |
| 12 | 0.06 | 0.10 | 0.15 | 0.19 | 0.40 | 0.49 | -0.05 | -0.10 | 0.23 |
| 13 | 0.72 | 1.81 | 0.09 | 0.00 | 0.00 | 0.05 | -0.01 | 0.00 | -0.03 |
| 14 | 0.39 | 0.95 | -0.06 | 1.30 | -0.04 | 0.39 | -0.04 | 0.36 | 0.04 |
| 15 | 0.29 | -0.02 | -0.01 | 0.20 | 0.36 | 0.19 | 0.01 | 0.01 | -0.13 |
| 16 | 0.27 | 0.58 | 0.00 | 0.61 | -0.07 | 0.75 | 0.02 | 0.84 | 0.01 |
| 17 | 0.03 | 0.08 | 1.17 | 0.13 | 0.08 | 0.15 | -0.01 | 1.29 | 0.09 |
| 18 | 0.00 | 0.03 | 0.55 | 0.32 | -0.01 | 0.15 | 0.11 | -0.22 | 0.10 |
| 19 | 0.07 | 0.25 | 0.27 | -0.19 | 0.85 | -0.09 | 0.50 | 0.08 | 2.40 |
| 20 | 0.28 | 0.36 | 0.22 | 0.04 | -0.10 | -0.03 | 0.00 | 0.00 | 0.59 |
| 21 | 0.03 | 0.03 | 0.65 | 0.43 | 0.41 | 0.25 | 0.44 | 0.14 | 0.13 |
| 22 | 0.29 | -0.04 | 0.00 | 0.01 | 0.13 | 0.00 | 0.25 | 0.00 | 0.83 |
| 23 | 0.12 | 0.13 | 0.59 | -0.02 | 0.16 | 0.16 | -0.03 | -0.01 | 1.36 |
| 24 | -0.04 | 0.00 | 0.03 | 0.06 | -0.04 | -0.03 | 0.02 | 0.17 | -0.03 |
| 25 | 0.00 | 0.04 | 0.05 | 0.71 | 0.07 | 0.32 | 0.03 | 0.21 | 0.01 |
| 26 | -0.01 | 0.17 | 0.03 | 0.23 | 0.00 | 0.27 | 0.04 | 0.37 | 0.05 |
| 27 | -0.23 | 0.06 | 0.21 | -0.23 | -0.11 | 0.11 | 1.29 | 0.05 | 1.35 |
| 28 | 0.13 | 0.20 | 0.52 | 1.71 | 0.62 | 1.48 | 0.68 | 1.73 | -0.04 |
| 29 | 0.65 | 0.19 | 1.34 | 0.49 | 0.02 | 0.11 | 0.81 | 0.56 | 0.02 |
| 30 | 0.19 | 0.75 | 0.25 | 0.80 | 0.03 | 0.40 | 0.52 | 1.05 | 0.03 |
| 31 | 0.04 | 0.94 | 0.11 | 0.80 | 0.15 | 0.13 | 0.39 | 0.27 | -0.22 |
| 32 | 0.22 | 0.46 | 0.77 | 0.08 | 0.48 | 0.09 | 0.02 | -0.13 | 1.88 |
| 33 | 0.20 | -0.02 | -0.02 | 0.00 | 0.24 | 0.47 | -0.03 | -0.03 | -0.05 |
| 34 | -0.04 | 0.26 | 0.24 | 0.02 | 0.04 | -0.05 | 1.39 | 0.12 | 0.13 |
| 35 | 0.03 | 0.26 | 0.02 | 0.00 | 0.16 | 0.20 | -0.01 | -0.07 | 0.03 |
| 36 | 0.19 | 0.72 | -0.02 | 0.50 | 0.04 | 0.69 | 0.05 | 0.11 | 0.02 |
| 37 | 1.23 | 0.11 | -0.02 | -0.04 | 0.74 | -0.08 | 0.01 | 0.04 | 0.56 |
| 38 | 0.07 | 0.01 | 0.67 | 0.26 | 0.25 | 0.73 | 0.20 | 0.00 | 0.60 |
| 39 | 0.29 | 0.22 | 0.90 | 0.23 | 0.38 | 0.13 | -0.15 | 0.01 | 1.74 |
| 40 | 0.01 | 0.47 | 0.00 | 0.01 | 0.01 | 0.00 | 0.21 | 0.01 | -0.02 |
| 41 | -0.02 | 0.17 | -0.12 | 0.71 | 0.17 | -0.02 | 0.39 | -0.07 | 0.03 |
| 42 | 0.69 | 0.36 | 0.11 | 0.13 | 0.28 | -0.26 | 0.08 | 0.51 | 1.93 |
| 43 | 0.07 | 0.14 | 0.38 | -0.06 | 0.40 | 0.49 | 0.05 | 0.14 | 0.29 |
| 44 | 1.37 | 1.12 | 0.42 | 0.74 | 0.53 | 0.89 | -0.05 | -0.04 | -0.02 |
| 45 | 0.00 | 0.06 | 1.50 | 0.72 | 0.00 | 0.36 | 0.73 | 0.49 | 0.12 |
| 46 | 0.65 | 0.41 | 0.00 | 0.00 | 0.09 | 0.05 | 0.01 | 0.00 | 0.69 |
| 47 | 0.18 | 0.00 | 0.00 | 0.02 | 0.54 | 0.47 | 0.01 | 0.00 | -0.10 |
| 48 | 0.17 | 0.30 | 0.00 | -0.05 | -0.02 | 0.14 | 1.09 | 0.25 | 0.57 |
| 49 | 0.07 | 0.45 | 0.01 | 0.01 | 0.03 | 0.01 | -0.03 | -0.02 | -0.10 |

Figure 7: Sensitivity of all neurons. This is the full result of Figure 2.

| B-PER | I-PER | B-LOC | I-LOC | B-ORG | I-ORG | B-MISC | I-MISC | O |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| 11 | 13 | 45 | 28 | 1 | 28 | 34 | 28 | 19 |
| 44 | 9 | 29 | 14 | 19 | 44 | 27 | 17 | 42 |
| 37 | 44 | 17 | 3 | 37 | 16 | 48 | 30 | 32 |
| 9 | 14 | 2 | 30 | 28 | 38 | 10 | 16 | 39 |
| 13 | 31 | 39 | 31 | 9 | 36 | 29 | 29 | 23 |
| 42 | 11 | 32 | 44 | 47 | 3 | 45 | 42 | 27 |
| 29 | 30 | 38 | 45 | 44 | 12 | 28 | 45 | 22 |
| 46 | 36 | 21 | 25 | 32 | 43 | 30 | 26 | 46 |
| 14 | 16 | 9 | 41 | 6 | 33 | 19 | 14 | 38 |
| 2 | 40 | 23 | 2 | 21 | 47 | 21 | 1 | 20 |
| 39 | 32 | 18 | 16 | 43 | 0 | 41 | 5 | 48 |
| 22 | 49 | 28 | 36 | 12 | 30 | 31 | 31 | 37 |
| 15 | 46 | 44 | 29 | 11 | 14 | 22 | 0 | 3 |
| 20 | 1 | 43 | 21 | 39 | 45 | 40 | 48 | 6 |
| 16 | 3 | 19 | 5 | 15 | 4 | 38 | 11 | 2 |
| 5 | 20 | 30 | 18 | 42 | 25 | 2 | 25 | 43 |
| 32 | 42 | 34 | 38 | 8 | 26 | 18 | 2 | 12 |
| 33 | 48 | 20 | 39 | 38 | 21 | 11 | 24 | 5 |
| 30 | 0 | 27 | 26 | 33 | 1 | 42 | 21 | 34 |
| 36 | 35 | 12 | 15 | 41 | 35 | 9 | 43 | 21 |
| 47 | 5 | 31 | 12 | 23 | 15 | 4 | 34 | 45 |
| 48 | 34 | 42 | 42 | 35 | 23 | 43 | 3 | 7 |
| 6 | 19 | 13 | 17 | 31 | 18 | 36 | 36 | 18 |
| 28 | 39 | 5 | 32 | 22 | 17 | 26 | 19 | 4 |
| 23 | 28 | 6 | 24 | 5 | 48 | 25 | 27 | 1 |
| 10 | 29 | 10 | 9 | 46 | 39 | 16 | 37 | 17 |
| 7 | 10 | 25 | 20 | 17 | 31 | 32 | 8 | 8 |
| 38 | 41 | 1 | 8 | 7 | 10 | 24 | 4 | 26 |
| 19 | 26 | 26 | 47 | 25 | 2 | 37 | 40 | 14 |
| 43 | 8 | 24 | 10 | 10 | 29 | 46 | 15 | 41 |
| 49 | 43 | 7 | 4 | 34 | 27 | 47 | 39 | 30 |
| 12 | 23 | 35 | 34 | 36 | 32 | 15 | 20 | 35 |
| 31 | 37 | 4 | 0 | 49 | 46 | 20 | 7 | 0 |
| 21 | 12 | 49 | 7 | 4 | 13 | 0 | 47 | 36 |
| 17 | 17 | 0 | 22 | 30 | 6 | 8 | 13 | 29 |
| 35 | 6 | 8 | 49 | 29 | 11 | 17 | 38 | 11 |
| 4 | 45 | 48 | 40 | 40 | 49 | 13 | 46 | 16 |
| 40 | 27 | 46 | 1 | 26 | 9 | 35 | 22 | 25 |
| 18 | 25 | 47 | 46 | 45 | 7 | 6 | 10 | 9 |
| 0 | 2 | 22 | 6 | 13 | 8 | 5 | 9 | 10 |
| 25 | 21 | 16 | 33 | 18 | 40 | 49 | 23 | 44 |
| 45 | 18 | 40 | 13 | 48 | 22 | 23 | 49 | 40 |
| 26 | 38 | 15 | 11 | 3 | 41 | 33 | 6 | 24 |
| 41 | 24 | 11 | 35 | 24 | 5 | 3 | 33 | 13 |
| 24 | 47 | 3 | 23 | 14 | 24 | 14 | 44 | 28 |
| 8 | 4 | 37 | 37 | 0 | 20 | 1 | 41 | 33 |
| 1 | 7 | 36 | 48 | 16 | 34 | 12 | 35 | 47 |
| 34 | 15 | 33 | 43 | 2 | 37 | 44 | 12 | 49 |
| 3 | 33 | 14 | 19 | 20 | 19 | 7 | 32 | 15 |
| 27 | 22 | 41 | 27 | 27 | 42 | 39 | 18 | 31 |

Figure 8: Importance rankings for all labels. This is the full result of Figure 3.

# Quantity doesn't buy quality syntax with neural language models

**Marten van Schijndel**
Cornell University
mv443@cornell.edu

**Aaron Mueller**
Johns Hopkins University
amueller@jhu.edu

**Tal Linzen**
Johns Hopkins University
tal.linzen@jhu.edu

## Abstract

Recurrent neural networks can learn to predict upcoming words remarkably well on average; in syntactically complex contexts, however, they often assign unexpectedly high probabilities to ungrammatical words. We investigate to what extent these shortcomings can be mitigated by increasing the size of the network and the corpus on which it is trained. We find that gains from increasing network size are minimal beyond a certain point. Likewise, expanding the training corpus yields diminishing returns; we estimate that the training corpus would need to be unrealistically large for the models to match human performance. A comparison to GPT and BERT, Transformer-based models trained on billions of words, reveals that these models perform even more poorly than our LSTMs in some constructions. Our results make the case for more data efficient architectures.

## 1 Introduction

Recurrent neural network language models (LMs) can learn to predict upcoming words with remarkably low perplexity (Mikolov et al., 2010; Jozefowicz et al., 2016; Radford et al., 2019). This overall success has motivated targeted paradigms that measure whether the LM's predictions reflect a correct analysis of sentence structure. One such evaluation strategy compares the probability assigned by the LM to a minimal pair of sentences differing only in grammaticality (Linzen et al., 2016). In the following example, the LM is expected to assign a higher probability to the sentence when the verb agrees in number with the subject (1a) than when it does not (1b):

(1) a. The author laughs.
b. *The author laugh.

RNN LMs have been shown to favor the grammatical variant in the vast majority of cases sampled at random from a corpus (Linzen et al., 2016), but their accuracy decreases in the presence of distracting nouns intervening between the head of the subject and the verb, especially when those nouns are in relative clauses (Marvin and Linzen, 2018). Can we hope to address these deficits by training larger and larger networks on larger and larger corpora, relying on the "unreasonable effectiveness" of massive datasets (Halevy et al., 2009) and computational power?[1] Or would architectural advances be necessary to improve our LMs' syntactic representations (Kuncoro et al., 2018)?

This paper takes a first step towards addressing this question. We train 125 RNN LMs with long short-term memory (LSTM, Hochreiter and Schmidhuber, 1997) units, systematically varying the size of the training corpus and the dimensionality of the models' hidden layer, and track the relationship between these parameters and the performance of the models on agreement dependencies in a range of syntactic constructions (Marvin and Linzen, 2018). We also compare our RNNs' accuracy to that of GPT (Radford et al., 2018) and BERT (Devlin et al., 2019), Transformer-based LMs trained on very large corpora.

We find that model capacity does not consistently improve performance beyond a minimum threshold. Increased corpus size likewise has a moderate and inconsistent effect on accuracy. We estimate that even if training data yielded consistent improvements, an unreasonable amount of data would be required to match human accuracy. We conclude that reliable and data-efficient learning of syntax is likely to require external supervision signals or a stronger inductive bias than that provided by RNNs and Transformers.

[1] http://www.incompleteideas.net/IncIdeas/BitterLesson.html



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5831–5837,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

## 2 Language models

**Architecture:** All of the models we trained consisted of two LSTM layers. We trained models with 100, 200, 400, 800 or 1600 units in each hidden layer. Input and output weights were tied during training (Press and Wolf, 2017; Inan et al., 2017); consequently, the input embedding had the same dimensionality as the hidden layers.

**Training data:** We trained networks of each size on 2M, 10M, 20M, 40M and 80M words (2M = 2 million). We extracted five disjoint sections of the WikiText-103 corpus (Merity et al., 2016) for each corpus size;[2] in total, we trained 125 models (5 layer sizes $\times$ 5 corpus sizes $\times$ 5 corpus subsets).[3] We used the WikiText-103 validation set for validation.

**Vocabulary:** To ensure comparability across different models trained on different data, we used the same vocabulary for all the models we trained. The vocabulary consisted of an intersection of the 400k word GloVe vocabulary (Pennington et al., 2014) with the 50k words used by GRNN (see below); the resulting vocabulary had 28,438 words.

**GRNN:** We also report the syntactic performance of a publicly available LSTM LM (Gulordava et al., 2018, henceforth GRNN). This trained model has been the focus of a considerable amount of analysis work in the past year. The model has two layers of 650 units each, and was trained on 80M words.

**Comparison with Transformers:** Finally, we report results from two publicly available LMs based on non-recurrent self-attention (Transformers; Vaswani et al., 2017): GPT (Radford et al., 2018) and BERT (Devlin et al., 2019). Both of these models have been argued to learn powerful syntactic representations (Goldberg, 2019; Wolf, 2019). We compare our results to those reported by Wolf (2019) on a similar challenge set for these two Transformer models.[4]

GPT is a 12-layer Transformer with 110 million parameters (compared to GRNN's 39 million parameters); it was trained on 1 billion words. BERT has a similar architecture to GPT,[5] with three differences: it is bidirectional, it was trained on 3.3 billion words, and it has a different training objective than the typical LM: it attempts to predict a single masked word in a sentence given the words both before and after the target word. For comparability to the LSTMs and GPT, we examine the agreement performance of BERT when only the words before the target are given (in contrast to the bidirectional tests reported by Goldberg 2019).

## 3 Evaluation

We tested each trained model on the constructions from the Marvin and Linzen (2018) challenge set, which is based on the agreement paradigm described in the introduction.[6] We replaced the verbs used by Marvin and Linzen with the high-frequency verbs *is/are*, *was/were* and *has/have*. This was done to ensure that even the models trained on smaller corpora will have had exposure to both forms of the verb in question.

We performed statistical tests of our hypotheses using Bayes factors, which quantify the support the data provide for a more complex model compared to a simpler one (Rouder et al., 2009). We computed two-sample Bayes factors using `ttestBF` from the `BayesFactor` R package (Morey and Rouder, 2018) using default settings. Our null hypothesis was that there is no difference in accuracy between the two sets of models in question (e.g., all models with 400 units per layer compared to all models with 800 units per layer). The magnitude of the resulting Bayes factor $K$ can be interpreted as follows (Jeffreys, 1961): $K < 1$ indicates that there is no difference in accuracy between the two model groups, and $K > 10$ provides strong evidence that the model groups obtain different accuracies.

## 4 Results

Increasing model size improved syntactic prediction accuracy up to 400 units per layer; further in-

[2] We made each of the 40M- and 80M-token training sets as disjoint as possible, but since Wikitext-103 only contains 103M tokens, it was not possible to make them wholly disjoint using Wikitext-103 as the mother corpus.

[3] Each model was initialized randomly and was trained to convergence with a dropout of 0.2 using a batch size of 20, backpropagating error for 35 observations. An initial learning rate of 20 was gradually annealed.

[4] The comparison is not exact because the Transformers were evaluated based on the rank of the two target verbs given the prefix, and the LSTMs based on the total log-probability of the sentence (including the final period); in addition, Wolf (2019) did not modify the dataset to use only high frequency verbs, as we describe in Section 3.

[5] BERT Base showed more accurate syntactic predictions than BERT Large (Goldberg, 2019), which has more parameters, so we only consider BERT Base.

[6] https://github.com/BeckyMarvin/LM_syneval

| Corpus size | | Layer size | |
|---|---|---|---|
| 2M → 10M | **5508.8** | 100 → 200 | **768.5** |
| 10M → 20M | 0.1 | 200 → 400 | **63.5** |
| 20M → 40M | **12.9** | 400 → 800 | 0.2 |
| 40M → 80M | 0.2 | 800 → 1600 | 0.1 |

Table 1: Strength of evidence for improvements in agreement prediction accuracy as a result of increasing corpus size averaging across layer size (left) or layer size averaging across corpus size (right), as quantified by Bayes factors. Boldfaced Bayes factors indicate strong evidence of improvement.

creases in model size had no effect (see Table 1 for the statistical tests). Increasing the amount of training data impacted accuracy in an inconsistent way. Training on 10M tokens produced general improvements across all constructions compared to 2M tokens. Doubling the corpus to 20M did not affect accuracy, but doubling it again to 40M did. There was no evidence of further improvement between 40M and 80M words.

In the remainder of this section we analyze the effect of increasing model size and training corpus size on the models' predictions for each construction in the data set.[7] A subset of the results is shown in Fig. 1; for the full results, see Figs. 3, 4 and 5 in the Appendix.

**Local number agreement:** All models trained on 10M words or more obtained perfect or near-perfect accuracy (mean $> 99\%$) in the cases where the verb was adjacent to its subject: simple agreement (*the author has/*have books*) and agreement within a sentential complement (*the mechanics said the author has/*have books*). When trained on 2M words, the models performed slightly worse, but their accuracy was still very high (mean 95.6%), regardless of model size. Overall, we conclude that the plurality of specific nouns and the generalization that a verb has to agree with a noun can be learned very quickly.

**Attractors:** Agreement across subject relative clauses (*the author that likes the guards has/*have books*) and across prepositional phrases (*the author next to the guards has/*have books*; Fig. 1a) benefited from increasing the hidden layer size to 400, but showed little improvement when hidden layer size was increased further. Accuracy in these constructions consistently improved as the amount of training data increased.

**Object relative clauses:** Expanding the training corpus improved local agreement **within** object relative clauses (*the movies that the guard has/*have are good*; Fig. 1b) for all model sizes, but only improved agreement **across** those clauses (*the movie that the guards like has/*have drama*; Fig. 1c) in models with larger hidden layers. Larger hidden layers improved accuracy in object relatives only when a relativizer was present, and only up to about 80% accuracy. When a sentence lacked an overt relativizer (*the movies the security guard has/*have are good*; Fig. 1d), all models performed poorly, with accuracy levelling off around 70%.

**Coordination:** Perhaps surprisingly, all LSTM LMs struggled with agreement in a coordinated verb phrase (*the authors laugh and have/*has books*; Fig. 1e), even though this construction does not include distracting nouns between the subject and the second verb. In larger models trained on more data, accuracy was higher when the second verb was **further** from the subject (*the authors know many different languages and have/*has books*; Fig. 1f).

Training on more than 10M tokens did not improve accuracy in short VP coordination, even when the amount of data was multiplied by eight (10M → 80M: $K < 1$), unlike coordination across long VPs, which benefited from additional data (10M → 80M: $K > 90$). These results further challenge the assumption that increased amounts of training data will lead to adequately abstract syntactic representations: RNNs show a limited ability to generalize from instances of a construction that have longer constituents to instances with shorter constituents.

**Reflexive anaphora:** A reflexive pronoun such as *themselves* must have an antecedent with the appropriate plurality in the same clause as the pronoun (*The manager that the architects like doubted himself/*themselves*). Accuracy was not strongly affected by the parameters we varied: reflexive agreement accuracy across a relative clause was consistently mediocre (61%–76%) regardless of model size or the amount of training data.

[7] See Table 2 in the Appendix for a Bayes factor analysis of the improvement in each construction for each amount of training data.

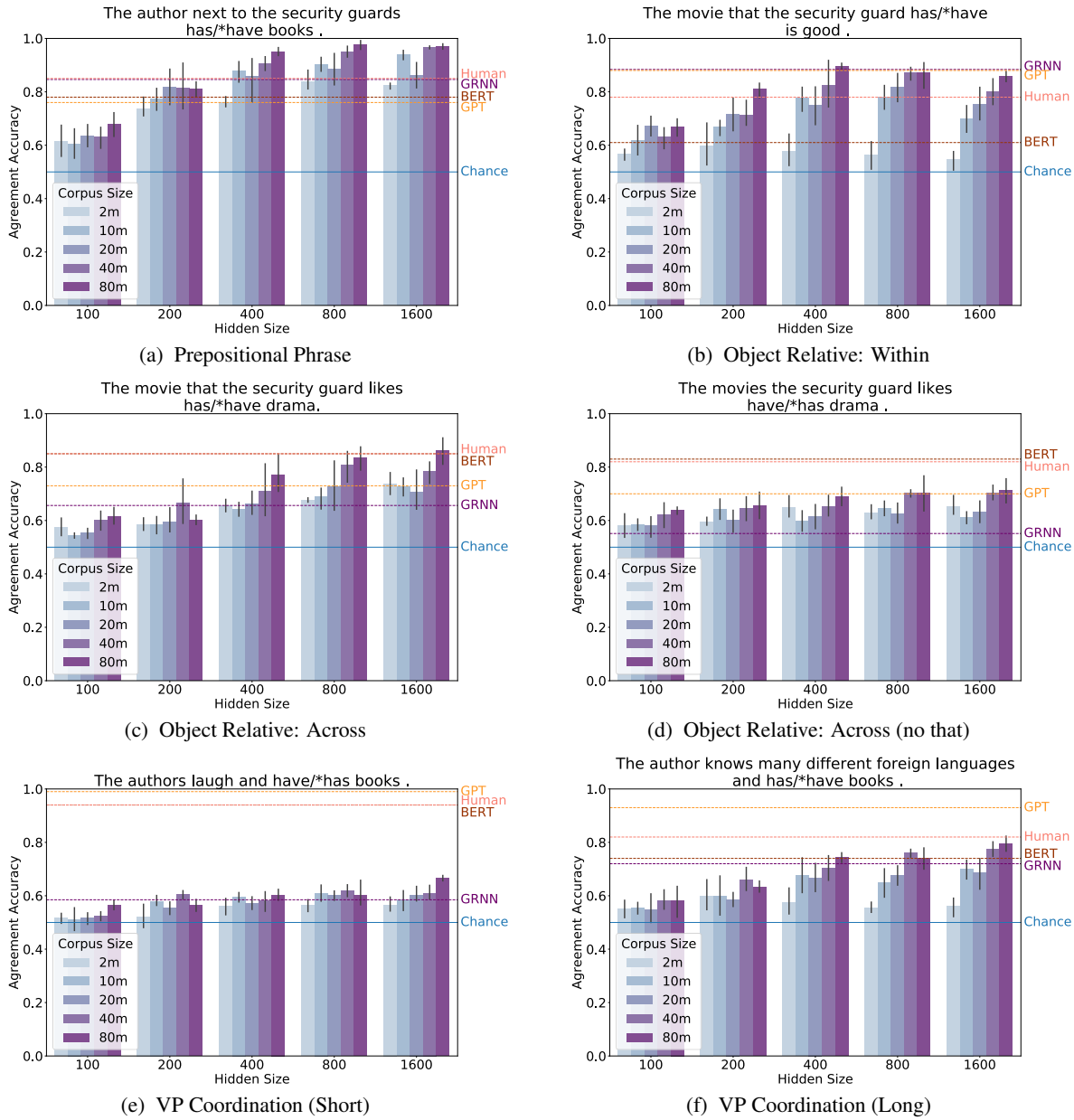


(a) Prepositional Phrase

(b) Object Relative: Within

(c) Object Relative: Across

(d) Object Relative: Across (no that)

(e) VP Coordination (Short)

(f) VP Coordination (Long)

Figure 1: LSTM agreement performance in several syntactic constructions. The solid horizontal line indicates chance performance. The dashed lines show the performance of GPT and BERT as reported by Wolf (2019), the performance of humans as reported by Marvin and Linzen (2018), and the performance of GRNN. Error bars reflect standard deviation across the five models in each category.

**Transformers:** Despite having more parameters and having been trained on significantly larger corpora, the two Transformer models performed either as well as or more poorly than our LSTMs in seven of the ten subject-verb agreement conditions. BERT underperformed GPT in several conditions despite having been trained on three times as many tokens as GPT.[8]

## 5 How much data would be enough?

How much training data would be required for an LSTM LM to perform at a human level (as reported by Marvin and Linzen 2018) in the condi-

[8] Goldberg (2019) reports much better results using a setup in which BERT has access to both left and right context. We hypothesize that the task is made significantly simpler when the model knows where the target word is relative to the end of the sentence. For example, if the point of prediction is at the last word of the sentence, it is also the last point at which the verb agreeing with the main clause subject could possibly occur; the model does not need to detect the end of the relative clause to perform the task in this case.

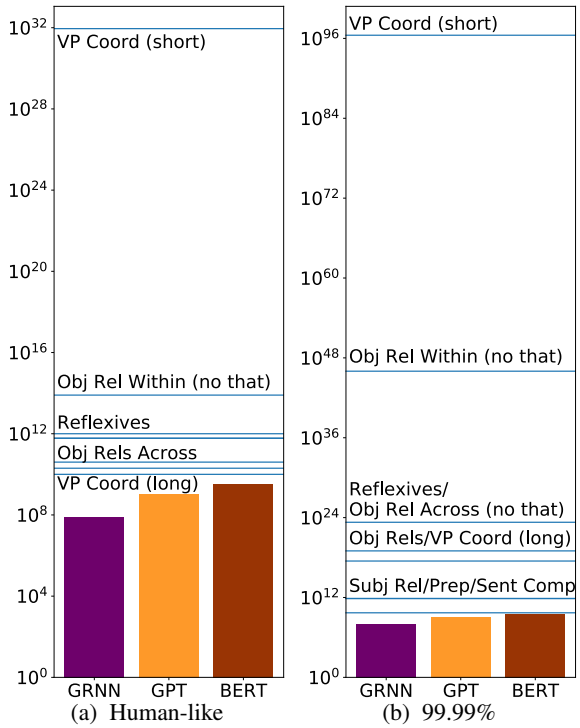


Figure 2: Lines depict number of training tokens needed for LSTMs to achieve human-like (left) or 99.99% accuracy (right) in each syntactic agreement condition, according to our estimates. Bars depict the amount of data on which each model was trained.

tions in which our models do not already perform at a human level? As a conservative estimate, we measured the error reduction achieved by doubling the data from 20M to 40M tokens (the largest error reduction we observed beyond 2M $\rightarrow$ 10M).[9] Under the assumption that each subsequent doubling of training data would produce the same percent error reduction, we predicted the amount of data required to obtain human-like and 99.99% accuracy (see Fig. 2).[10] We found that every remaining construction would require over 10 billion tokens to achieve human-like performance, and most would require trillions of tokens to achieve perfect accuracy – an impractically large amount of training data, especially for these relatively simple syntactic phenomena.

[9]See Tables 3 and 4 in the Appendix for data requirements estimated from other error reduction rates.

[10]Human performance on this task is well known to be far from perfect, with error rates approaching 25% in some contexts (Bock and Miller, 1991). While modeling human errors is of considerable interest to cognitive scientists (Linzen and Leonard, 2018), we believe that in most applied contexts it is desirable for the model to make no errors at all.

## 6 Discussion

We have investigated the effect of network size and training corpus size on the quality of the syntactic representations of LSTM LMs, as measured by agreement prediction accuracy. Increased model size had limited benefits; models with 400 hidden units performed significantly better than smaller models, but further increases in network size had no effect. The limited effect of network size is consistent with previous findings on sequence labeling tasks (Reimers and Gurevych, 2017; Greff et al., 2017). We have also shown that increasing the amount of training data is unlikely to result in human-like accuracy in all cases.

We found a striking difference in agreement accuracy between short and long coordinated verb phrases: performance on short phrases was poorer. While RNNs are known to struggle with generalizing short patterns to longer sequences, this pattern constitutes a failure to generalize to *shorter* sequences (cf. Trask et al., 2018); techniques for improving longer distance dependency learning in LMs (e.g., Trinh et al., 2018; Dai et al., 2019) are unlikely to mitigate this deficit. This suggests that challenge sets should include materials that can be used to ascertain whether the model's syntactic representations are robust to syntactically irrelevant factors such as constituent length.

GPT and BERT, Transformer models trained on very large corpora, did not consistently outperform the LSTMs trained on several orders of magnitude less data. Other studies suggest that Transformer models suffer from similar problems as the LSTMs we have analyzed. BERT's agreement accuracy decreases as the subject becomes more distant from its verb (Bacon and Regier, 2019). Dramatically increasing the pre-training corpus for a BERT-like model from 562M words to 18G words only leads to a modest improvement in its natural language inference accuracy, from 81.7% to 82.3% (Baevski et al., 2019). Overall, this body of results points to the limited data efficiency of standard RNNs and Transformers, and indicates that learning syntax from realistic amounts of data—in particular the amount of data available to humans when they learn language—may require syntactically structured architectures or explicit syntactic supervision (Enguehard et al., 2017; Kuncoro et al., 2018, 2019; Wilcox et al., 2019).

## References


Geoff Bacon and Terry Regier. 2019. Does BERT agree? Evaluating knowledge of structure dependence through agreement relations. Technical report, UC Berkeley.

Alexei Baevski, Sergei Edunov, Yinhan Liu, Luke Zettlemoyer, and Michael Auli. 2019. Cloze-driven pretraining of self-attention networks. Technical report, Facebook AI Research.

Kathryn Bock and Carol A. Miller. 1991. Broken agreement. *Cognitive Psychology*, 23(1):45–93.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. Technical report, Carnegie Mellon University.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics.

Émile Enguehard, Yoav Goldberg, and Tal Linzen. 2017. Exploring the syntactic abilities of RNNs with multi-task learning. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 3–14. Association for Computational Linguistics.

Yoav Goldberg. 2019. Assessing BERT's syntactic abilities. Technical report, Bar Ilan University.

Klaus Greff, Rupesh K. Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber. 2017. LSTM: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. In *Proceedings of the 2018 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics.

Alon Halevy, Peter Norvig, and Fernando Pereira. 2009. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2):8–12.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying word vectors and word classifiers: A loss framework for language modeling. In Yoshua Bengio and Yann LeCun, editors, *Proceedings of the Fifth International Conference on Learning Representations*. International Conference on Learning Representations.

Harold Jeffreys. 1961. *Theory of Probability*, 3rd edition. Oxford University Press, Oxford.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

Adhiguna Kuncoro, Chris Dyer, John Hale, Dani Yogatama, Stephen Clark, and Phil Blunsom. 2018. LSTMs can learn syntax-sensitive dependencies well, but modeling structure makes them better. In *Proceedings of the 2018 Annual Meeting of the Association for Computational Linguistics*, pages 1426–1436. Association for Computational Linguistics.

Adhiguna Kuncoro, Chris Dyer, Laura Rimell, Stephen Clark, and Phil Blunsom. 2019. Scalable syntax-aware language models using knowledge distillation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3472–3484, Florence, Italy. Association for Computational Linguistics.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Tal Linzen and Brian Leonard. 2018. Distinct patterns of syntactic agreement errors in recurrent networks and humans. In *Proceedings of the 2018 Annual Meeting of the Cognitive Science Society*, pages 690–695. Cognitive Science Society.

Rebecca Marvin and Tal Linzen. 2018. Targeted syntactic evaluation of language models. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202. Association for Computational Linguistics.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Wikitext-103. Technical report, Salesforce.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)*, pages 1045–1048, Makuhari, Chiba, Japan.

Richard D. Morey and Jeffrey N. Rouder. 2018. *BayesFactor: Computation of Bayes Factors for Common Designs*. R package version 0.9.12-4.2.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of EMNLP*.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 2017 Annual Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. Technical report, OpenAI.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. Technical report, OpenAI.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of LSTM-networks for sequence tagging. In Martha Palmer, Rebecca Hwa, and Sebastian Riedel, editors, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 338–348. Association for Computational Linguistics.

Jeffrey N. Rouder, Paul L. Speckman, Dongchu Sun, Richard D. Morey, and Geoffrey Iverson. 2009. Bayesian t-tests for accepting and rejecting the null hypothesis. *Psychonomic Bulletin & Review*, 16(2):225–237.

Andrew Trask, Felix Hill, Scott E Reed, Jack Rae, Chris Dyer, and Phil Blunsom. 2018. Neural arithmetic logic units. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 8035–8044. Curran Associates, Inc.

Trieu H. Trinh, Andrew M. Dai, Minh-Thang Luong, and Quoc V. Le. 2018. Learning longer-term dependencies in RNNs with auxiliary losses. In *Proceedings of the 35th International Conference on Machine Learning*, pages 4965–4974. PMLR 80.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Ethan Wilcox, Peng Qian, Richard Futrell, Miguel Ballesteros, and Roger Levy. 2019. Structural supervision improves learning of non-local grammatical dependencies. In *Proceedings of the 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics.

Thomas Wolf. 2019. Some additional experiments extending the tech report "assessing BERT's syntactic abilities" by Yoav Goldberg. Technical report, Huggingface Inc.

# Higher-order Comparisons of Sentence Encoder Representations

**Mostafa Abdou**[†] [*] **Artur Kulmizev**[♣] **Felix Hill**[♢] **Daniel M. Low**[♠] **Anders Søgaard**[†]
[†]Department of Computer Science, University of Copenhagen
[♣]Department of Linguistics and Philology, Uppsala University
[♢]DeepMind [♠]Program in Speech and Hearing Bioscience and Technology,
Harvard Medical School-MIT

## Abstract

Representational Similarity Analysis (RSA) is a technique developed by neuroscientists for comparing activity patterns of different measurement modalities (e.g., fMRI, electrophysiology, behavior). As a framework, RSA has several advantages over existing approaches to interpretation of language encoders based on probing or diagnostic classification: namely, it does not require large training samples, is not prone to overfitting, and it enables a more transparent comparison between the representational geometries of different models and modalities. We demonstrate the utility of RSA by establishing a previously unknown correspondence between widely-employed pretrained language encoders and human processing difficulty via eye-tracking data, showcasing its potential in the interpretability toolbox for neural models.

## 1 Introduction

Examining the parallels between human and machine learning is a natural way for us to better understand the former and track our progress in the latter. The "black box" aspect of neural networks has recently inspired a large body of work related to interpretability, i.e. understanding of representations that such models learn. In NLP, this push has been largely motivated by linguistic questions, such as: *what linguistic properties are captured by neural networks?* and *to what extent do decisions made by neural models reflect established linguistic theories?* Given the relative recency of such questions, much work in the domain so far has been focused on the context of models in isolation (e.g. *what does model* **X** *learn about linguistic phenomenon* **Y**?) In order to more broadly understand models' representational tendencies, however, it is vital that such questions be formed not only with other models in mind, but also other representational methods and modalities (e.g. behavioral data, fMRI measurements, etc.). In context of the latter concern, the present-day interpretability toolkit has not yet been able to afford a practical way of reconciling this.

In this work, we employ Representational Similarity Analysis (RSA) as a simple method of interpreting neural models' representational spaces as they relate to other models and modalities. In particular, we conduct an experiment wherein we investigate the correspondence between human processing difficulty (as reflected by gaze fixation measurements) and the representations induced by popular pretrained language models. In our experiments, we hypothesize that there exists an overlap between the sentences which are difficult for humans to process and those for which per-layer encoder representations are least correlated.

Our intuition is that such sentences may exhibit factors such as low-frequency vocabulary, lexical ambiguity, and syntactic complexity (e.g. multiple embedded clauses), etc. that are uncommon in both standard language and, relatedly, the corpora employed in training large-scale language models. In the case of a human reader, encountering such a sentence may result in a number of processing delays, e.g. longer aggregate gaze duration. In the case of a sentence encoder, an uncommon sentence may lead to a degradation of representations in the encoder's layers, wherein a lower layer might learn to encode vastly different information than a higher one. Similarly, different models' representations may emphasize different aspects of these more complex sentences and therefore diverge from each other. With this in mind, our hypothesis is that sentences which are difficult for humans to process are likely to have divergent representations within models' internal layers and between different models' layers.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5838–5845,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

**Understanding and analysing language encoders** In recent years, some prominent efforts towards interpreting neural networks for NLP have included: developing suites that evaluate network representations through performance on downstream tasks (Conneau et al., 2017a; Wang et al., 2018; McCann et al., 2018); analyzing network predictions on carefully curated datasets (Linzen et al., 2016; Marvin and Linzen, 2018; Gulordava et al., 2018; Loula et al., 2018; Dasgupta et al., 2018; Tenney et al., 2018); and employing diagnostic classifiers to assess whether certain classes of information are encoded in a model's (intermediate) representations (Adi et al., 2016; Chrupała et al., 2017; Hupkes et al., 2017; Belinkov et al., 2017).

While these approaches provide valuable insights into how neural networks process a large variety of phenomena, they rely on decoding accuracy as a probe for encoded linguistic information. If properly biased, this means that they can detect whether information is encoded in a representation or not. However, they do not allow for a direct comparison of representational structure between models. Consider a toy dataset of five sentences of interest and three encodings derived from quite different processing models; a hidden state of a trained neural language model, a *tf-idf* weighted bag-of-words representation, and measurements of fixation duration from an eye-tracking device. Probing methods do not allow us to quantify or visualise, for each of these encoding strategies, how the encoder's responses to the five sentences relate to each other. Moreover, probing methods would not directly reveal whether the fixations from the eye-tracking device aligned more closely with the tf-idf representation or the states of the neural language model. In short, while probing classifier methods can establish if phenomena are separable based on the provided representations, they do not tell us about the overall geometry of the representational spaces. RSA, on the other hand, provides a basis for higher-order comparisons between spaces of representations, and a way to visualise and quantify the extent to which they are isomorphic.

Indeed, RSA has seen a modest introduction within interpretable NLP in recent years. For example, Chrupała et al. (2017) employed RSA as a means of correlating encoder representations of speech, text, and images in a post-hoc analysis of a multi-task neural pipeline. Similarly, Bouchacourt and Baroni (2018) used the framework to measure the similarity between input image embeddings and the representations of the same image by an agent in an language game setting. More recently, Chrupała and Alishahi (2019) correlated activation patterns of sentence encoders with symbolic representations, such as syntax trees. Lastly, similar to our work here, Abnar et al. (2019) proposed an extension to RSA that enables the comparison of a single model in the face of isolated, changing parameters, and employed this metric along with RSA to correlate NLP models' and human brains' respective representations of language. We hope to position our work among this brief survey and further demonstrate the flexibility of RSA across several levels of abstraction.

## 2 Representational Similarity Analysis

RSA was proposed by Kriegeskorte et al. (2008) as a method of relating the different representational modalities employed in neuroscientific studies. Due to the lack of correspondence between the activity patterns of disparate measurement modalities (e.g. brain activity via fMRI, behavioural responses), RSA aims to abstract away from the activity patterns themselves and instead compute representational dissimilarity matrices (RDMs), which characterize the information carried by a given representation method through dissimilarity structure.

Given a set of representational methods (e.g., pretrained encoders) $M$ and a set of experimental conditions (sentences) $N$, we can construct RDMs for each method in $M$. Each cell in an RDM corresponds to the dissimilarity between the activity patterns associated with pairs of experimental conditions $n_i, n_j \in N$, say, a pair of sentences. When $n_i = n_j$, the dissimilarity between an experimental condition and itself is intuitively 0, thus making the $N \times N$ RDM symmetric along a diagonal of zeros (Kriegeskorte et al., 2008).

The RDMs of the different representational methods in $M$ can then be directly compared in a Representational *Similarity* Matrix (RSM). This comparison of RDMs is known as second-order analysis, which is broadly based on the idea of a *second-order isomorphism* (Shepard and Chipman, 1970). In such an analysis, the principal point of comparison is the match between the dissimilarity structure of the different representa-

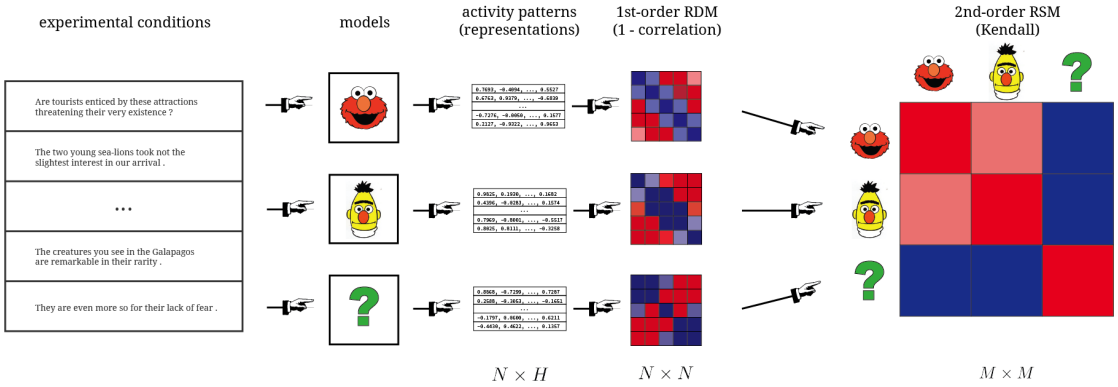


Figure 1: An example of first- and second-order analyses, where $N = \#$ of experimental conditions, $M = \#$ of models, and $H = \#$ of activity patterns observed for a given model (i.e. dimensionality). The right-most side of the figure depicts a representational similairty matrix (RSM) of correlations between RDMs.

tional methods. Intuitively, this can be expressed through the notion of *distance between distances*, and is thus related to Earth Mover's Distance (Rubner et al., 2000).[1] Figure 1 shows an illustration of the first and second order analyses for pretrained language encoders.

Note that RSA is meaningfully different from, and complementary to, methods that employ saturating functions of representation distances (e.g. decoding accuracy, mutual information), which suffer from **(a)** a ceiling effect: being able to distinguish experimental phenomenon $A$ from $B$ with with an accuracy of 100% and experimental phenomenon $C$ from $D$ with an accuracy of 100% does not mean that the distance between $A$ and $B$ is the same as that between $C$ and $D$; and **(b)** discretization (Nili et al., 2014).

We follow Kriegeskorte et al. (2008) in using the correlation distance of experimental condition pairs $n_i, n_j \in N$ as a dissimilarity measure, where $\bar{n_i}$ is the mean of $n_i$'s elements, $\cdot$ is the dot product, and $\|$ is the $l_2$ norm: $corr(x) = 1 - \frac{(n_i - \bar{n_i}) \cdot (n_j - \bar{n_j})}{\|(n_i - \bar{n_i}\|_2 \|(n_j - \bar{n_j}\|_2}$. Compared to other measures, correlation distance is preferable as it normalizes both the mean and variance of activity patterns over experimental conditions. Other popular measures include the Euclidean distance and the Malahanobis distance (Kriegeskorte et al., 2006).

## 3 Fixation Duration and Encoder Disagreement

Gaze fixation patterns have been shown to strongly reflect the online cognitive processing demands of human readers (Raney et al., 2014; Ashby et al., 2005) and to be dependent upon a number of linguistic factors (Van Gompel, 2007). Specifically, it has been demonstrated that word frequency, syntactic complexity, and lexical ambiguity play a strong part in determining which sentences are difficult for humans to process (Rayner and Duffy, 1986; Duffy et al., 1988; Levy, 2008).

Using the RSA framework, we aim to explore how gaze fixation patterns and the linguistic factors associated with sentence processing difficulty relate to the representational spaces of popular language encoders. Namely, we hypothesize that, for a given sentence, disagreement between hidden layers corresponds to processing difficulty. Because layer disagreement for a sentence measures the extent to which two layers (e.g. within BERT) disagree with each other about the pairwise similarity of the sentence (with other sentences in the corpus), a sentence with high layer disagreement will have unstable similarity relationships to other sentences in the corpus. This indicates that it has a degraded encoder representation. Going further, we also hypothesize that models' representations of said sentences may be confounded, in part, by factors that are known to influence humans.

**Eye-tracking data** For our experiments, we make use of the Dundee eye-tracking corpus (Kennedy et al., 2003), the English part of which consists of eye-movement data recorded as 10 native participants read 2,368 sentences from 20 newspaper articles. We consider the following fixation features: TOTAL FIXATION DURATION and FIRST PASS DURATION. For each of the features, we first take the average of the measurements recorded for all 10 participants per word, then ob-

[1] More precisely, our measure of dissimilarity between experimental conditions is analogous to *ground distance* and dissimilarity between RDMs to *earth mover's distance.*

tain sentence-level annotations by summing the measurements of all words in a sentence and dividing by its length. The result of this is two vectors $V_{totfix}$ and $V_{firstpass}$ of length $2,368$, where each cell in the vector corresponds to a sentence's average total fixation and average first pass duration, respectively.

**Syntactic complexity, word frequency, and lexical ambiguity** We also consider the three following linguistic features which affect processing difficulty. For each of the following the result is also a vector of length $2,368$ where each cell corresponds to a sentence:

**a.** the average word log frequency per sentence extracted from the British National Corpus (Leech, 1992), $V_{logFreq.}$.

**b.** the average number of senses per word per sentence extracted from WordNet (Miller, 1995), $V_{wordSense}$.

**c.** Yngve scores, a standard measure of syntactic complexity based on cognitive load (Yngve, 1960) , $V_{Yngve}$.

**Pretrained encoders** We conduct our analysis on pretrained BERT-large (Devlin et al., 2018) and ELMo (Peters et al., 2018), two widely employed contextual sentence encoders. To obtain a representation of a sentence from a given layer $L$, we perform mean-pooling over the time-steps which correspond to the words of a sentence, obtaining a vector representation of the sentence. Mean-pooling is a common approach for obtaining vector representations of sentences for downstream tasks (Peters et al., 2018; Conneau et al., 2017b). We refer to ELMo's lowest layer as E1, BERT's 11th layer as B11, etc.

**RDMs** We construct an RDM (see §2) for each contextual encoder's layers. Each RDM is a $2,368 \times 2,368$ matrix which represents the dissimilarity structure of the layer, (i.e., each row vector in the matrix contains the dissimilarity of a given sentence to every other sentence). We then compute the correlations between the two different RDMs. For our evaluation of how well the representational geometry of a layer correlates to another, we employ Kendall's $\tau_A$ as suggested in Nili et al. (2014), computing the pairwise correlation for each two corresponding rows in two RDMs. This second-order analysis gives us a pairwise relational similarity vector $V_{Corr_{L_i-L_j}}$ of length $2,368$, which has the correlations between two layers $L_i$ and $L_j$'s RDMs for each of the sentences.

**Third-order analysis** The final part of our analysis involves computing correlations (Spearman's $\rho$) of $\{V_{Corr_{L_i-L_j}}, V_{logFreq}, V_{Yngve}, V_{wordSense}\}$ with each of $V_{totfix}$ and $V_{firstpass}$. The results from this are shown in Table 1. The top section of the table shows correlations when $L_i$ and $L_j$ are the three final adjacent layers in BERT and ELMo. The middle section shows the results for top three BERT layer pairs $L_i$ and $L_j$ which maximize the correlation scores. The final section shows correlation with the linguistic features. Finally, Figure 2 shows Spearman's $\rho$ correlations between $V_{Corr_{L_i-L_j}}$ and each of $V_{totfix}$, and $V_{Yngve}$ for all combinations of the 24 BERT layers.

## 4 Discussion

Our results show highly significant negative correlations between $V_{Corr_{L_i-L_j}}$ and sentence gaze fixation times. These findings confirm the hypothesis that the sentences that are most challenging for humans to process, are the sentences **(a)** the layers of BERT disagree most on among themselves; and **(b)** that ELMo and BERT disagree most on, indicating that there may be common factors which affect human processing difficulty and result in disagreement between layers. By Layer disagreement we refer to the expression $1 - V_{Corr_{L_i-L_j}}$. It is important to note that these encoders are trained with a language modelling objective, unlike models where reading behaviour is explicitly modelled (Hahn and Keller, 2016) or predicted (Matthies and Søgaard, 2013). Indeed, the similarities here emerge naturally as a function of the task being performed. This can be seen as analogous to the case of similarities observed between neural networks trained to perform object recognition and spatio-temporal cortical dynamics (Cichy et al., 2016).

**Syntactic complexity** Figure 2 shows that, for all combinations of BERT layers, total fixation time and Yngve scores have strong negative and positive correlations (respectively) with layer disagreement. Furthermore, we observe that disagreement between middle layers seems to show the strongest correlation with Yngve scores. To confirm this, we split the correlations into four groups: "low" ($i, j \in [1, 8]$), "middle" ($i, j \in$

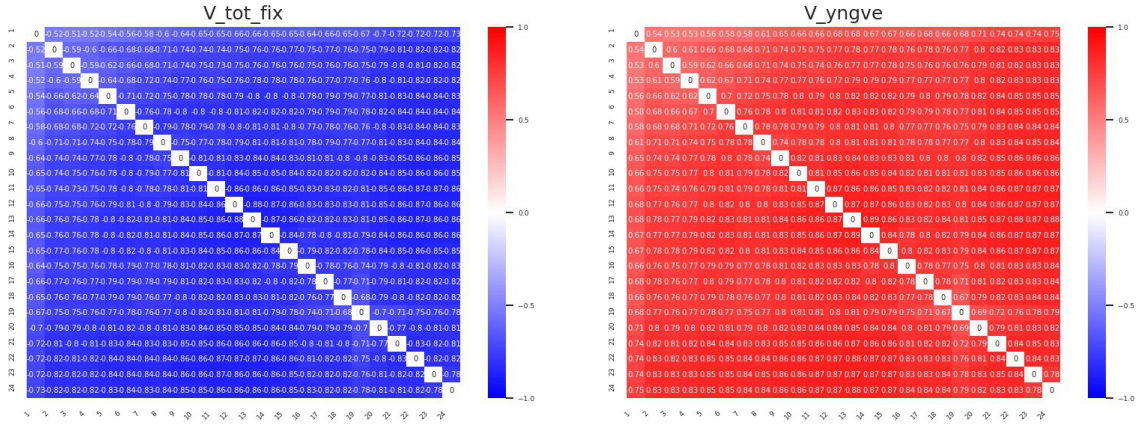


Figure 2: RSMs showing (Spearman's $\rho$) correlation between disagreement among layers $i$ and $j$ ($V_{Corr_{L_i-L_j}}$) and $V_{totfix}$ (left) and $V_{Yngve}$ (Right). BERT layers are denoted with numbers from 1 (topmost) to 24 (lowest).

| Layer Disagreement | Total Fixation | First Pass Duration |
|---|---|---|
| E1-B22 | -0.46 | -0.46 |
| E2-B23 | -0.66 | -0.67 |
| E3-B24 | -0.22 | -0.23 |
| B11-B12 | -0.88 | -0.87 |
| B12-B13 | -0.87 | -0.85 |
| B10-B21 | -0.87 | -0.86 |
| **Linguistic Features** | | |
| Log Freq. | -0.20 | -0.19 |
| Avg. Senses per Word | -0.007* | -0.004* |
| Yngve Score | 0.66 | 0.66 |

Table 1: Spearman's $\rho$ between $V_{Corr_{L_i-L_j}}$, $V_{logFreq.}$, $V_{wordSense}$, $V_{Yngve}$ and each of $V_{totfix}$ and $V_{firstpass}$. All correlations significant with $p < 0.0001$ after Bonferroni correction unless marked with *.

$[9, 16]$), "high" ($i, j \in [17, 24]$), and "out" ($|i - j| > 7$), with the latter representing out-of-group correlations (e.g. $Corr_{L_1-L_{24}}$). To account for correlations between disagreeing adjacent layers (e.g. $|i - j| = 1$) and Yngve scores being higher (as a possible confounding factor), we also distinguish layers as either "adjacent" or "non-adjacent". Considering these two factors as three- and two-leveled independent variables respectively, we conduct a two-way analysis of variance. The analysis reveals that the effect of group is significant at $F(3, 275) = 78.47, p < 0.0001$, with "low" ($\mu$ = 0.65, $\sigma$ = 0.08), "middle" ($\mu$ = 0.84, $\sigma$ = 0.03), "high" ($\mu$ = 0.80, $\sigma$ = 0.05), and "out" ($\mu$ = 0.80, $\sigma$ = 0.05). Neither the effect of adjacency nor its interaction with group proved to be significant.

This can be seen as (modest) support for the findings of previous work (Blevins et al., 2018; Tenney et al., 2019): namely, that the intermediate layers of neural language models encode the most syntax, and are therefore possibly more sensitive towards syntactic complexity. A very similar pattern is observed for total fixation time. When considered together with the correlation between $V_{Yngve}$ and fixation times, this indicates a tripartite affinity between layer disagreement, syntactic complexity, and fixation.

**Lexical Ambiguity and Word Frequency** Finally, we observe that $V_{logFreq.}$ has a moderate correlation with both fixation time and layer disagreement and that $V_{wordSense}$ is nearly uncorrelated to both. Detailed plots of the latter can be found in Appendix A.

## 5 Conclusion

We presented a framework for analyzing neural network representations (RSA) that allowed us to relate human sentence processing data with language encoder representations. In experiments conducted on two widely used encoders, our findings show that sentences which are difficult for humans to process have more divergent representations both intra-encoder and between different encoders. Furthermore, we lend modest support to the intuition that a model's middle layers encode comparatively more syntax. Our framework offers insight that is complimentary to decoding or probing approaches, and is particularly useful to compare representations from across modalities.

## Acknowledgements

We would like to thank Vinit Ravishankar, Matt Lamm, and the anonymous reviewers for their helpful comments. Mostafa Abdou and Anders Søgaard are supported by a Google Focused Research Award and a Facebook Research Award.

## References


Abnar, S., Beinborn, L., Choenni, R., and Zuidema, W. (2019). Blackbox meets blackbox: Representational similarity & stability analysis of neural language models and brains. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 191–203, Florence, Italy. Association for Computational Linguistics.

Adi, Y., Kermany, E., Belinkov, Y., Lavi, O., and Goldberg, Y. (2016). Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. *arXiv preprint arXiv:1608.04207*.

Ashby, J., Rayner, K., and Clifton, C. (2005). Eye movements of highly skilled and average readers: Differential effects of frequency and predictability. *The Quarterly Journal of Experimental Psychology Section A*, 58(6):1065–1086.

Belinkov, Y., Màrquez, L., Sajjad, H., Durrani, N., Dalvi, F., and Glass, J. (2017). Evaluating layers of representation in neural machine translation on part-of-speech and semantic tagging tasks. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1–10.

Blevins, T., Levy, O., and Zettlemoyer, L. (2018). Deep rnns encode soft hierarchical syntax. *arXiv preprint arXiv:1805.04218*.

Bouchacourt, D. and Baroni, M. (2018). How agents see things: On visual representations in an emergent language game. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 981–985, Brussels, Belgium. Association for Computational Linguistics.

Chrupała, G. and Alishahi, A. (2019). Correlating neural and symbolic representations of language. *arXiv preprint arXiv:1905.06401*.

Chrupała, G., Gelderloos, L., and Alishahi, A. (2017). Representations of language in a model of visually grounded speech signal. *arXiv preprint arXiv:1702.01991*.

Cichy, R. M., Khosla, A., Pantazis, D., Torralba, A., and Oliva, A. (2016). Comparison of deep neural networks to spatio-temporal cortical dynamics of human visual object recognition reveals hierarchical correspondence. *Scientific reports*, 6:27755.

Conneau, A., Kiela, D., Schwenk, H., Barrault, L., and Bordes, A. (2017a). Supervised learning of universal sentence representations from natural language inference data. *CoRR*, abs/1705.02364.

Conneau, A., Kiela, D., Schwenk, H., Barrault, L., and Bordes, A. (2017b). Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*.

Dasgupta, I., Guo, D., Stuhlmüller, A., Gershman, S. J., and Goodman, N. D. (2018). Evaluating compositionality in sentence embeddings. *arXiv preprint arXiv:1802.04302*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Duffy, S. A., Morris, R. K., and Rayner, K. (1988). Lexical ambiguity and fixation times in reading. *Journal of memory and language*, 27(4):429–446.

Gulordava, K., Bojanowski, P., Grave, E., Linzen, T., and Baroni, M. (2018). Colorless green recurrent networks dream hierarchically. *arXiv preprint arXiv:1803.11138*.

Hahn, M. and Keller, F. (2016). Modeling human reading with neural attention. *arXiv preprint arXiv:1608.05604*.

Hupkes, D., Veldhoen, S., and Zuidema, W. (2017). Visualisation and'diagnostic classifiers' reveal how recurrent and recursive neural networks process hierarchical structure. *arXiv preprint arXiv:1711.10203*.

Kennedy, A., Hill, R., and Pynte, J. (2003). The dundee corpus. In *Proceedings of the 12th European conference on eye movement*.

Kriegeskorte, N., Goebel, R., and Bandettini, P. (2006). Information-based functional brain mapping. *Proceedings of the National Academy of Sciences*, 103(10):3863–3868.

Kriegeskorte, N., Mur, M., and Bandettini, P. A. (2008). Representational similarity analysis-connecting the branches of systems neuroscience. *Frontiers in systems neuroscience*, 2:4.

Leech, G. N. (1992). 100 million words of english: the british national corpus (bnc).

Levy, R. (2008). Expectation-based syntactic comprehension. *Cognition*, 106(3):1126–1177.

Linzen, T., Dupoux, E., and Goldberg, Y. (2016). Assessing the ability of lstms to learn syntax-sensitive dependencies. *arXiv preprint arXiv:1611.01368*.

Loula, J., Baroni, M., and Lake, B. M. (2018). Rearranging the familiar: Testing compositional generalization in recurrent networks. *arXiv preprint arXiv:1807.07545*.

Marvin, R. and Linzen, T. (2018). Targeted syntactic evaluation of language models. *arXiv preprint arXiv:1808.09031*.

Matthies, F. and Søgaard, A. (2013). With blinkers on: Robust prediction of eye movements across readers. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 803–807.

McCann, B., Keskar, N. S., Xiong, C., and Socher, R. (2018). The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*.

Miller, G. A. (1995). Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Nili, H., Wingfield, C., Walther, A., Su, L., Marslen-Wilson, W., and Kriegeskorte, N. (2014). A toolbox for representational similarity analysis. *PLoS computational biology*, 10(4):e1003553.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Raney, G. E., Campbell, S. J., and Bovee, J. C. (2014). Using eye movements to evaluate the cognitive processes involved in text comprehension. *Journal of visualized experiments: JoVE*, (83).

Rayner, K. and Duffy, S. A. (1986). Lexical complexity and fixation times in reading: Effects of word frequency, verb complexity, and lexical ambiguity. *Memory & cognition*, 14(3):191–201.

Rubner, Y., Tomasi, C., and Guibas, L. (2000). The earth mover's distance as a metric for image retrieval. In *IJCV*.

Shepard, R. N. and Chipman, S. (1970). Second-order isomorphism of internal representations: Shapes of states. *Cognitive psychology*, 1(1):1–17.

Tenney, I., Das, D., and Pavlick, E. (2019). Bert rediscovers the classical nlp pipeline. *arXiv preprint arXiv:1905.05950*.

Tenney, I., Xia, P., Chen, B., Wang, A., Poliak, A., McCoy, R. T., Kim, N., Van Durme, B., Bowman, S., Das, D., et al. (2018). What do you learn from context? probing for sentence structure in contextualized word representations.

Van Gompel, R. P. (2007). *Eye movements: A window on mind and brain*. Elsevier.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. (2018). Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Yngve, V. H. (1960). A model and an hypothesis for language structure. *Proceedings of the American philosophical society*, 104(5):444–466.

## A Correlation Heatmaps

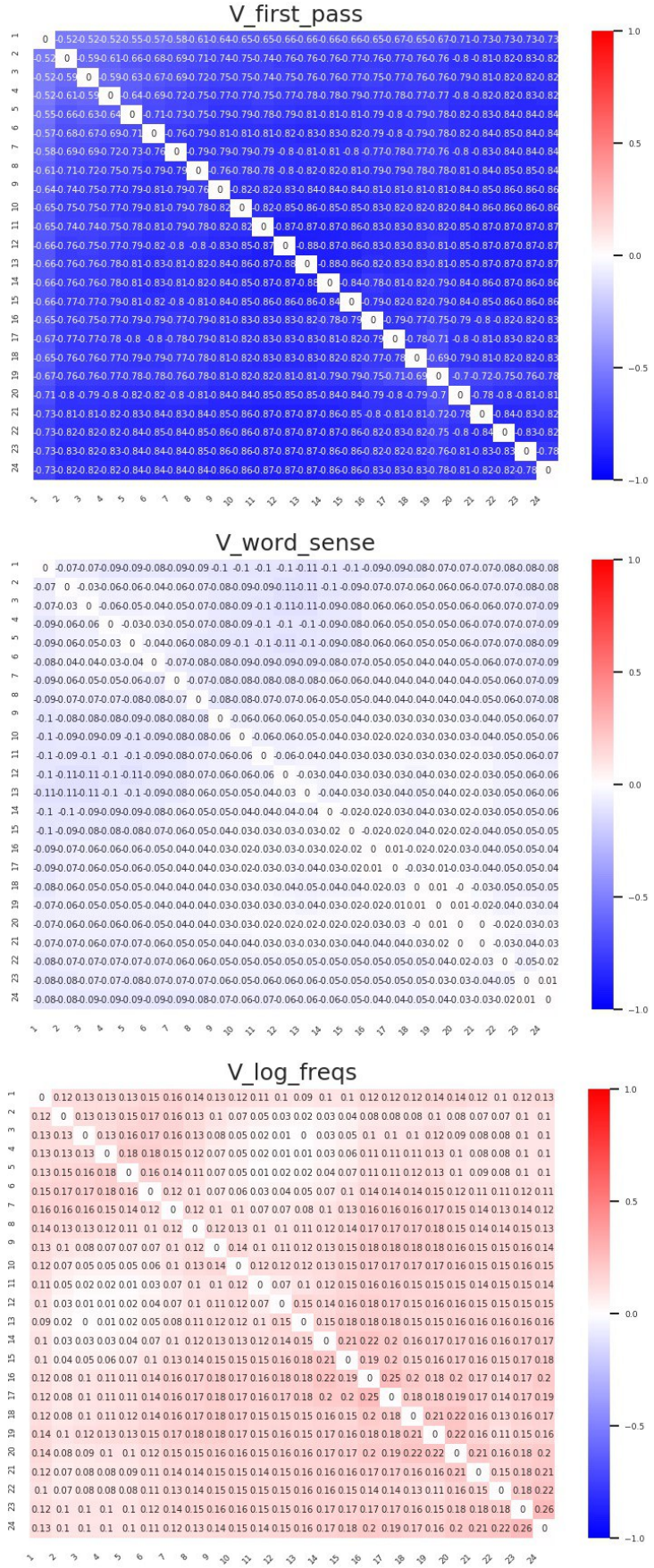


Figure 3: RSM heatmaps showing (Spearman's $\rho$) correlation between disagreement among layers $i$ and $j$ ($V_{Corr_{L_i - L_j}}$) and **(a)** $V_{firstpass}$ (top), **(b)** $V_{wordSense}$ (middle) and, **(c)** $V_{logFreq}$ (bottom).

# Text Genre and Training Data Size in Human-Like Parsing

**John T. Hale** and **Adhiguna Kuncoro**
DeepMind / London
{jthale,akuncoro}@google.com

**Keith B. Hall**
Google Research / New York
kbhall@google.com

**Chris Dyer**
DeepMind / London
cdyer@google.com

**Jonathan R. Brennan**
University of Michigan / Ann Arbor
jobrenn@umich.edu

## Abstract

Domain-specific training typically makes NLP systems work better. We show that this extends to cognitive modeling as well by relating the states of a neural phrase-structure parser to electrophysiological measures from human participants. These measures were recorded as participants listened to a spoken recitation of the same literary text that was supplied as input to the neural parser. Given more training data, the system derives a better cognitive model — but only when the training examples come from the same textual genre. This finding is consistent with the idea that humans adapt syntactic expectations to particular genres during language comprehension (Kaan and Chun, 2018; Branigan and Pickering, 2017).

## 1 Introduction

Natural language processing (NLP) systems based on deep neural networks are sensitive to the amount and type of training data that they receive. A "data hungry" method may not work well until it is supplied with sufficient examples (e.g. Yogatama et al., 2019). Likewise, transfer to a different textual genre may be poor (Petrov and McDonald, 2012). This is the classic problem of domain adaptation[1] which arises in many areas of NLP.

This paper revisits domain adaptation in the context of human-like parsing. With this human-like aspect in mind, we consider models that use linguistically-plausible trees (see Frank, 2011 for a review) and operate incrementally from left to right (e.g. Steedman, 2000). We quantify the fit to human language performance using freely-available electrophysiological data (henceforth: EEG) that was elicited by a pre-existing literary text (Brennan and Hale, 2019).

These EEG data come from a naturalistic stimulus, and in virtue of their higher temporal resolution, are more detailed than reading times or plausibility judgments. Hale et al. (2018) were the first to model them using a neural parser. In that study, textual training data came from the same book as did the human participants' stimuli. While this yielded a model that was quite well-matched to the EEG modeling task, its training data was confined to just 1543 sentences (24K words). In contrast, recent studies suggest that neural language models require substantially more data to achieve human-like linguistic competence (Gulordava et al., 2018; Futrell et al., 2019; Frank and Hoeks, 2019).

We investigate this question of data size together with a contrast between textual domains or "genres". Modeling human neural signals, we find that in-domain training leads to a better and better fit as more examples are added to the training set whereas with out-of-domain data, more examples do not help. This is interesting given the consistent reductions in language modeling perplexity with more data, which we observe across both domains. We further find that, across all amounts of training data, models that incorporate linguistically-plausible phrase structure achieve a better fit to human EEG data than models that do not. This suggests that phrase structure should play an important role in human-like models of language comprehension, even in models that benefit from large training data.

## 2 Methodology

We proceed by comparing parsing systems that are based on Recurrent Neural Network Grammars (Dyer et al., 2016; Wilcox et al., 2019, henceforth; RNNG) and trained according to fourteen

[1] It remains quite difficult to reconcile human-like incremental parsing with high performance out-of-domain; many researchers take a nonincremental whole-sentence approach (Gildea, 2001; Baucom et al., 2013; Joshi et al., 2018).



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5846–5852,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| | |
|---|---|
| complexity metric: | surprisal of hypotheses in beam (Hale, 2001; Roark et al., 2009) |
| amount of training data: | 39832,$\{100, 250, 500, 750\}$K, 1M and 1437575 sentences |
| genre: | newspaper text (Graff et al., 2005) and lexically-similar literature (Gutenberg) |

Table 1: Training regimes

different regimes. These training regimes cross seven different amounts of training data with two different genres of writing. The question across all regimes is: how well does a complexity metric derived from these parsers improve the modeling of EEG data from human participants engaged in language comprehension of the same text? Note that the dependent variable here is not parsing performance or language modeling perplexity of the RNNG, but rather the helpfulness, as regards cognitive modeling, of a derived measure based on the intermediate states that the RNNG visits during decoding of the stimulus text that the human participants heard. Table 1 summarizes these training regimes.

## 3 Materials

### 3.1 Lexically-similar literature

In the first genre, we ranked e-books from the freely-available Project Gutenberg collection according to the CosineTop50 metric from McClosky et al. (2010). This (dis)similarity metric compares vectors whose components are attestation counts from the 50 most frequent words in a reference text. Here, the reference text is the one that human EEG study participants listened to – the first chapter of Alice in Wonderland. Table 2 shows some highly-ranked books on this scheme. It yielded largely juvenile literature from the $19^{\text{th}}$ century aimed at girls. The average sentence length in this selection was 17 words.

| dissimilarity | title | author |
|---|---|---|
| 0.0584 | The Admiral's Caravan | Charles E. Carryl |
| 0.0620 | The Secret Garden | Frances Hodgson Burnett |
| 0.0628 | The Lodger | Marie Belloc Lowndes |
| 0.0687 | The Girls and I: A Veracious History | Mary Louisa Stewart Molesworth |
| 0.0689 | What Timmy Did | Marie Adelaide Belloc |
| 0.0724 | Little Miss Peggy | Mrs. Molesworth |
| 0.0725 | The Girls of St. Olave's | Mabel Mackintosh |
| 0.0741 | The Celebrity at Home | Violet Hunt |
| 0.0750 | I've Married Marjorie | Margaret Widdemer |
| 0.0752 | The Forged Note | Oscar Micheaux |
| 0.0755 | Mary Erskine | Jacob Abbott |
| 0.0758 | The Bountiful Lady | Thomas Cobb |
| 0.0758 | Legacy | James H Schmitz |
| 0.0763 | Some Little People | George Kringle |
| 0.0774 | In the Wilderness | Robert Hichens |

Table 2: Alice-like books from Project Gutenberg

### 3.2 Newspaper text

In the second genre, we randomly sampled news articles from the English Gigaword corpus (Graff et al., 2005). This sampling was made regardless of the particular national source, i.e. Agence France-Press, New York Times or Xinhua News Agency. Sentences in this sample were, on average, 20 words long. This out-of-domain text had a CosineTop50 dissimilarity level of 0.56.

### 3.3 Presumptive Trees

Both genres were parsed using a re-implementation of the Berkeley parser (Petrov et al., 2006) to yield presumptively-correct, "silver-grade" trees. This Berkeley parser was trained on a diverse set of annotated data. These include the Penn Treebank's Wall Street Journal materials (Marcus et al., 1993), the Question Treebank (Judge et al., 2006), Ontonotes (e.g. Pradhan and Ramshaw, 2017) and Parsing-the-Web Corpora (Petrov and McDonald, 2012). In a manual inspection of randomly sampled silver trees, the only obvious mistakes were tagging errors (1 newspaper, 2 literature), which are not harmful since RNNGs do not use part of speech tags. Indeed, the bracketing and phrase labels on these silver trees appeared to be fully consistent with the Penn Treebank Bracketing Guidelines (Bies et al., 1995). Before being passed to the RNNG as training data, these trees were post-processed to remove empty elements, punctuation, and function tags.

## 4 Training

### 4.1 Hyperparameters

We tuned RNNG hyperparameters to optimize development set perplexity. The development set comprises the entire Alice in Wonderland book as obtained from Project Gutenberg.[2] To control for

[2]While the first chapter of Alice in Wonderland that we use as stimuli is thus a subset of the development set, recall from section 2 that our evaluation metric is fit to the human EEG, rather than validation perplexity.

model capacity, across all training regimes we used the same hyper-parameter that consistently achieved good validation perplexities across all configurations.[3]

### 4.2 Vocabulary

Within each of the two genres, we facilitate comparison across different training set sizes by running the RNNGs with a shared vocabulary derived from the largest training configuration. An attestation frequency cut-off was applied to each of them to ensure broadly comparable rates of out-of-vocabulary words on the development set, Alice in Wonderland. For the lexically-similar literature, this threshold was 5 attestations, whereas for the newspaper text this threshold was 20 attestations. On the validation set, these cut-offs yielded out-of-vocabulary rates of 2.3 to 2.4 percent for newspaper text and 0.5 to 1.28 percent for lexically-similar literature.

### 4.3 Achieved perplexity

Per-action perplexity levels[4] achieved by the trained RNNG parsing models on the development set are shown in Figure 1. Validation perplexity consistently improves with more and more silver-grade training data, even when that data comes from a different domain.[5]

## 5 EEG Regression model

Surprisal values from a beam search parser based on RNNG are entered as predictors into a regression model of scalp voltages. Models are fit with the `brm` function in R and model fits were compared using Bayesian model comparison (Vehtari et al., 2017). These regression models include random intercepts for participant along-side predictors to account for factors of non-interest that nevertheless are known to influence sentence processing difficulty (see e.g. Goodkind and Bicknell, 2018). These are: sentence position in the stimulus text, word position within each sentence, acoustic sound power, unigram word frequency in the HAL corpus (Balota et al., 2007). Unigram predictors are included for the previous word, the current word and the next word.

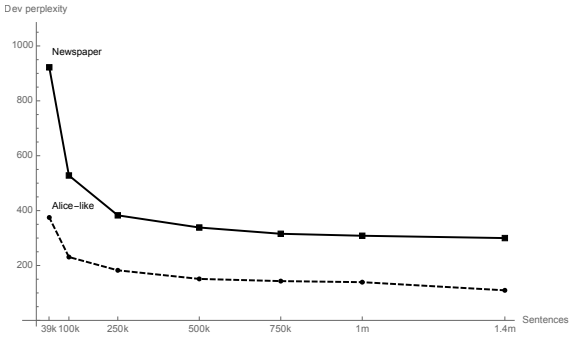


Figure 1: Language modeling perplexity (lower is better) on Alice in Wonderland across the two genres of training data.

The EEG data themselves come from 33 datasets that were collected by Brennan and Hale (2019) while participants listened to the first chapter of Alice in Wonderland.[6] Ocular artifacts and other noise sources are removed from the raw signal using ICA and visual inspection. The EEG data are reduced to a single spatio-temporal region of interest (ROI) comprising the data from anterior channels across both hemispheres between 200 and 400 ms after the onset of content-words. This anterior ROI has, uniquely, shown sensitivity to surprisal values from incremental parsers under a data-driven whole-scalp analysis in Brennan and Hale (2019). Note that this ROI is earlier and more anterior than the usual locus of the N400 component, which typically manifests on central electrodes at or around 400 ms post-stimulus (for a review see Kaan, 2007).

## 6 Results

Figure 2 plots the goodness-of-fit of a regression model that includes RNNG-derived complexity metrics. As the neural phrase structure parser is trained on increasingly larger corpora, the regression model of EEG amplitudes fits better and better. However, this pattern only obtains with in-domain training data that is lexically-similar to the first chapter of Alice in Wonderland. When trained on newspaper text from the Gigaword cor-

[3] The RNNG hyperparameters are: 2-layer stack LSTM (Dyer et al., 2015), 450 hidden units, and an initial SGD learning rate of 0.3, decayed exponentially with a factor of 0.1 applied after the tenth epoch.

[4] The per-action perplexity is computed based on the joint probability of strings ($\mathbf{x}$) and trees ($\mathbf{y}$), denoted as $p(\mathbf{x}, \mathbf{y})$, which therefore aggregate the perplexity of the next-word prediction with the perplexity of tree-building actions. Approximate inference methods such as important sampling can be used to derive an estimate of $p(\mathbf{x})$ (Dyer et al., 2016).

[5] While relative perplexity levels internal to a genre are comparable in virtue of a shared vocabulary, absolute perplexity values are not directly comparable across the genres since these vocabularies are different.

[6] These data are available at: https://doi.org/10.7302/Z29C6VNH.

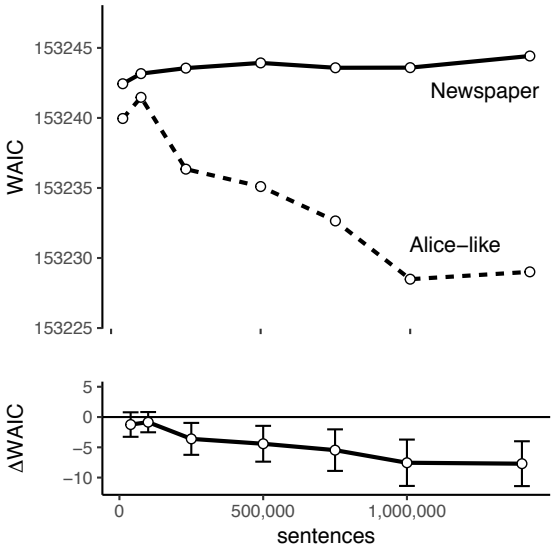


Figure 2: Goodness of fit of surprisal predictions from RNNG at beam size $k = 200$ syntactic analyses on anterior electrodes during the time window 200-400ms after the onset of a spoken word in the first chapter of Alice in Wonderland. The widely-applicable information criterion WAIC is described further in Vehtari et al. (2017). Lower WAIC indicates a better fit.

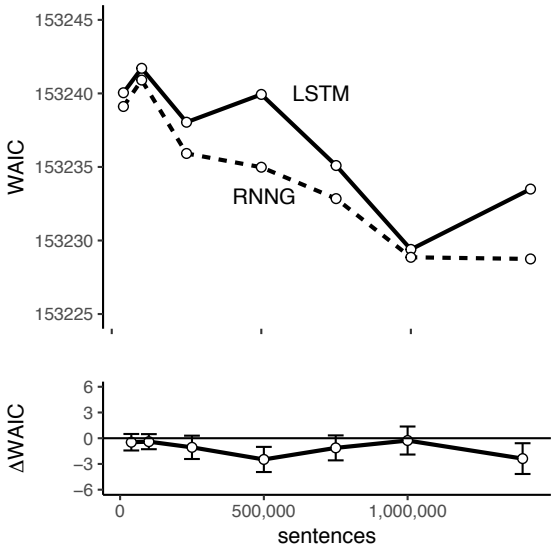


Figure 3: Comparison between RNNG with explicit hierarchical representations and LSTM without such representations. Lower WAIC indicates a better fit to human electrophysiological data.

pus, the goodness of fit to human EEG data remains about the same no matter how much training data is supplied. Tukey's test of additivity indicates an interaction between genre and training set size, $F(1, 5) = 171.3, p < 0.000001$. No effect of surprisal values obtained in electrodes and timepoints corresponding to the N400 component.

Figure 3 compares the RNNG, which explicitly uses phrase structure representations, to an LSTM sequence model which does not (Hochreiter and Schmidhuber, 1997). Here, both models receive the benefit of in-domain training on the lexically-similar literature. But regardless of how much in-domain training data is made available, the explicitly phrase-structural RNNG always offers a better account of the EEG signal.

## 7 Related Work

The importance of domain adaptation in NLP has been well-established in earlier work (see Daumé III, 2007 and footnote 1), including applications to parsing (Sarkar, 2001; McClosky et al., 2006; Søgaard and Rishøj, 2010; Weiss et al., 2015). Our approach to in-domain data selection is closely related to earlier work in language modeling and machine translation (Keller and Lapata, 2003; Moore and Lewis, 2010; Axelrod et al., 2011). Our focus on large training sets responds to recent work in language modeling (Devlin et al., 2019; Radford et al., 2019). The research reported here differs in two ways from these earlier studies. First, we consider the interaction between domain and amount of training data, rather than examining each variable in isolation. Second, we investigate the impact of these variables on cognitive modeling, which reveals a pattern that is different from what we observe in the standard perplexity evaluation. We focus on text genre, rather than online adaptation as van Schijndel and Linzen (2018) do. Despite coming at the problem from a different direction (and using EEG rather than self-paced reading) our results agree with van Schijndel and Linzen in suggesting that some kind of adaptation must be going on in human language comprehension.

## 8 Conclusion

These comparisons confirm that genre matters. If surprisal describes human linguistic expectations, then we can say that those expectations are better-modeled by a parsing system that benefits from in-domain training. This would follow if, as Kaan and Chun (2018) have suggested, people are able to very rapidly adjust their syntactic expectations to match a particular genre.

Indeed, these expectations seems to be phrase-

structural in nature. Certainly the presence of unigram nuisance predictors in the EEG regression (section 5) and the comparatively worse performance of the LSTM sequence model (Figure 3) render it unlikely that this finding is due to word frequency or superficial co-occurrence. Rather, the neural parser has learned something about $19^{\text{th}}$ century children's literature that can be captured at the syntactic level. Whatever these syntactic properties are, it could have been the case that they were equally learnable from newswire text or Alice-like books. But in fact Alice-like books generalize better to human EEG data.

This use of moment-by-moment processing difficulty to adjudicate between trained NLP systems offers a reminder that the quest for human-level performance in language technology should always be understood in relation to a particular kind of linguistic performance in a particular genre.

## Acknowledgments

The authors would like to thank Laura Rimell for valuable discussion of this work. This material is based upon work supported by the National Science Foundation under Grant No 1607441.

## References

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. 2011. Domain adaptation via pseudo in-domain data selection. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 355–362.

David A. Balota, Melvin J. Yap, Keith A. Hutchison, Michael J. Cortese, Brett Kessler, Bjorn Loftis, James H. Neely, Douglas L. Nelson, Greg B. Simpson, and Rebecca Treiman. 2007. The English Lexicon Project. *Behavior Research Methods*, 39(3):445–459.

Eric Baucom, Levi King, and Sandra Kübler. 2013. Domain adaptation for parsing. In *Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013*, pages 56–64, Hissar, Bulgaria. INCOMA Ltd. Shoumen, BULGARIA.

Ann Bies, Mark Ferguson, Karen Katz, and Robert MacIntyre. 1995. *Bracketing Guidelines for Treebank II Style Penn Treebank Project*. Linguistic Data Consortium. University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-95-06-0.

Holly P. Branigan and Martin J. Pickering. 2017. An experimental approach to linguistic representation. *Behavioral and Brain Sciences*, 40:e282.

Jonathan R. Brennan and John T. Hale. 2019. Hierarchical structure guides rapid linguistic predictions during naturalistic listening. *PLOS ONE*, 14(1):1–17.

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *Proc. of ACL*, pages 256–263.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proc. of NAACL*. arXiv:1810.04805.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343, Beijing, China. Association for Computational Linguistics.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209, San Diego, California. Association for Computational Linguistics.

Robert Frank. 2011. Phrase structure. In Patrick Colm Hogan, editor, *The Cambridge Encyclopedia of the Language Sciences*, pages 621–622. Cambridge University Press.

Stefan L. Frank and John C.J. Hoeks. 2019. The interaction between structure and meaning in sentence comprehension: Recurrent neural networks and reading times. In *Proceedings of the 41$^{\text{st}}$ Annual Meeting of the Cognitive Science Society*, Montreal.

Richard Futrell, Ethan Wilcox, Takashi Morita, Peng Qian, Miguel Ballesteros, and Roger Levy. 2019. Neural language models as psycholinguistic subjects: Representations of syntactic state. In *Proceedings of the 18th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Daniel Gildea. 2001. Corpus variation and parser performance. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*.

Adam Goodkind and Klinton Bicknell. 2018. Predictive power of word surprisal for reading times is a linear function of language model quality. In *Proceedings of the 8th Workshop on Cognitive Modeling and Computational Linguistics (CMCL 2018)*, pages 10–18, Salt Lake City, Utah. Association for Computational Linguistics.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2005. English Gigaword second edition. LDC2005T12.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1195–1205, New Orleans, Louisiana. Association for Computational Linguistics.

Gutenberg. 2019. Project Gutenberg.

John Hale. 2001. A probabilistic Earley parser as a psycholinguistic model. In *Second Meeting of the North American Chapter of the Association for Computational Linguistics*.

John Hale, Chris Dyer, Adhiguna Kuncoro, and Jonathan Brennan. 2018. Finding syntax in human encephalography with beam search. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2727–2736, Melbourne, Australia. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Vidur Joshi, Matthew Peters, and Mark Hopkins. 2018. Extending a parser to distant domains using a few dozen partially annotated examples. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1190–1199, Melbourne, Australia. Association for Computational Linguistics.

John Judge, Aoife Cahill, and Josef van Genabith. 2006. Questionbank: Creating a corpus of parse-annotated questions. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, pages 497–504, Stroudsburg, PA, USA. Association for Computational Linguistics.

Edith Kaan. 2007. Event-related potentials and language processing: A brief overview. *Language and Linguistics Compass*, 1:571–591.

Edith Kaan and Eunjin Chun. 2018. Syntactic adaptation. In Kara D. Federmeier and Duane G. Watson, editors, *Current Topics in Language*, volume 68 of *Psychology of Learning and Motivation*, chapter 4, pages 85–116. Academic Press.

Frank Keller and Mirella Lapata. 2003. Using the web to obtain frequencies for unseen bigrams. *Computational Linguistics*, 29(3):459–484.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19:313–330.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proc. of NAACL*.

David McClosky, Eugene Charniak, and Mark Johnson. 2010. Automatic domain adaptation for parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 28–36, Los Angeles, California. Association for Computational Linguistics.

Robert C. Moore and William Lewis. 2010. Intelligent selection of language model training data. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 220–224.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440, Sydney, Australia. Association for Computational Linguistics.

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. Presented at the First Workshop on Syntactic Analysis of Non-Canonical Language held in cooperation with NAACL-HLT.

Sameer Pradhan and Lance Ramshaw. 2017. Ontonotes: Large scale multi-layer, multi-lingual, distributed annotation. In Nancy Ide and James Pustejovsky, editors, *Handbook of Linguistic Annotation*, pages 521–554. Springer Netherlands, Dordrecht.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. OpenAI blog.

Brian Roark, Asaf Bachrach, Carlos Cardenas, and Christophe Pallier. 2009. Deriving lexical and syntactic expectation-based measures for psycholinguistic modeling via incremental top-down parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 324–333.

Anoop Sarkar. 2001. Applying co-training methods to statistical parsing. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics on Language Technologies*, pages 1–8.

Marten van Schijndel and Tal Linzen. 2018. A neural model of adaptation in reading. In *Proceedings of*

*the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4704–4710, Brussels, Belgium. Association for Computational Linguistics.

Anders Søgaard and Christian Rishøj. 2010. Semi-supervised dependency parsing using generalized tri-training. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1065–1073.

Mark Steedman. 2000. *The Syntactic Process*. MIT Press.

Aki Vehtari, Andrew Gelman, and Jonah Gabry. 2017. Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC. *Statistics and Computing*, 27(5):1413–1432.

David Weiss, Christopher Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Proc. of ACL*.

Ethan Wilcox, Peng Qian, Richard Futrell, Miguel Ballesteros, and Roger Levy. 2019. Structural supervision improves learning of non-local grammatical dependencies. In *Proc. of NAACL*.

Dani Yogatama, Cyprien de Masson d'Autume, Jerome Connor, Tomas Kocisky, Mike Chrzanowski, Lingpeng Kong, Angeliki Lazaridou, Wang Ling, Lei Yu, Chris Dyer, and Phil Blunsom. 2019. Learning and evaluating general linguistic intelligence. `arXiv:1901.11373`.

# Feature2Vec: Distributional semantic modelling of human property knowledge

**Steven Derby** **Paul Miller** **Barry Devereux**

Queen's University Belfast, Belfast, United Kingdom
{sderby02, p.miller, b.devereux}@qub.ac.uk

## Abstract

Feature norm datasets of human conceptual knowledge, collected in surveys of human volunteers, yield highly interpretable models of word meaning and play an important role in neurolinguistic research on semantic cognition. However, these datasets are limited in size due to practical obstacles associated with exhaustively listing properties for a large number of words. In contrast, the development of distributional modelling techniques and the availability of vast text corpora have allowed researchers to construct effective vector space models of word meaning over large lexicons. However, this comes at the cost of interpretable, human-like information about word meaning. We propose a method for mapping human property knowledge onto a distributional semantic space, which adapts the *word2vec* architecture to the task of modelling concept features. Our approach gives a measure of concept and feature affinity in a single semantic space, which makes for easy and efficient ranking of candidate human-derived semantic properties for arbitrary words. We compare our model with a previous approach, and show that it performs better on several evaluation tasks. Finally, we discuss how our method could be used to develop efficient sampling techniques to extend existing feature norm datasets in a reliable way.

## 1 Introduction

Distributional semantic modelling of word meaning has become a popular method for building pretrained lexical representations for downstream Natural Language Processing (NLP) tasks (Baroni and Lenci, 2010; Mikolov et al., 2013; Pennington et al., 2014; Levy and Goldberg, 2014; Bojanowski et al., 2017). In this approach, meaning is encoded in a dense vector space model, such that words (or concepts) that have vector representations that are spatially close together are similar in meaning. A criticism of these real-valued embedding vectors is the opaqueness of their representational dimensions and their lack of cognitive plausibility and interpretability (Murphy et al., 2012; Şenel et al., 2018). In contrast, human conceptual property knowledge is often modelled in terms of relatively sparse and interpretable vectors, based on verbalizable, human-elicited features collected in property knowledge surveys (McRae et al., 2005; Devereux et al., 2014). However, gathering and collating human-elicited property knowledge for concepts is very labour intensive, limiting both the number of words for which a rich feature set can be gathered, as well as the completeness of the feature listings for each word. Neural embedding models, on the other hand, learn from large corpora of text in an unsupervised fashion, allowing very detailed, high-dimensional semantic models to be constructed for a very large number of words. In this paper, we propose *Feature2Vec*, a computational framework that combines information from human-elicited property knowledge and information from distributional word embeddings, allowing us to exploit the strengths and advantages of both approaches. Feature2Vec maps human property norms onto a pretrained vector space model of word meaning. The embedding of feature-based information in the pretrained embedding space makes it possible to rank the relevance of features using cosine similarity, and we demonstrate how simple composition of features can be used to approximate concept vectors.

## 2 Related Work

Several property-listing studies have been conducted with human participants in order to build *property norms* – datasets of normalized human-verbalizable feature listings for lexical concepts (McRae et al., 2005; Devereux et al., 2014).



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5853–5859, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

One use of feature norms is to critically examine distributional semantic models on their ability to encode grounded, human-elicited semantic knowledge. For example, Rubinstein et al. (2015) demonstrated that state-of-the-art distributional semantic models fail to predict attributive properties of concept words (e.g. the properties *is-red* and *is-round* for the word *apple*) as accurately as taxonomic properties (e.g. *is-a-fruit*). Similarly, Sommerauer and Fokkens (2018) investigated the types of semantic knowledge encoded within pretrained word embeddings, concluding that some properties cannot be learned by supervised classifiers. Collell and Moens (2016) compared linguistic and visual representations of object concepts on their ability to represent different types of property knowledge. Research has shown that state-of-the-art distributional semantic models built from text corpora fail to capture important aspects of meaning related to grounded perceptual information, as this kind of information is not adequately represented in the statistical regularities of text data (Li and Gauthier, 2017; Kelly et al., 2014). Motivated by these issues, Silberer (2017) constructed multimodal semantic models from text and image data, with the goal of grounding word meaning using visual attributes. More recently, Derby et al. (2018) built similar models with the added constraint of sparsity, demonstrating that sparse multimodal vectors provide a more faithful representation of human semantic representations. Finally, the work that most resembles ours is that of Fagarasan et al. (2015), who use Partial Least Squares Regression (PLSR) to learn a mapping from a word embedding model onto specific conceptual properties. Concurrent work recently undertaken by Li and Summers-Stay (2019) replaces the PLSR model with a feedforward neural network. In our work, we instead map property knowledge directly into vector space models of word meaning, rather than learning a supervised predictive function from concept embedding dimensions to feature terms.

## 3 Method

We make primary comparison with the work of Fagarasan et al. (2015), although their approach differs from ours in that they map from an embedding space onto the feature space, while we learn a mapping from the feature domain onto the embedding space. We outline both methods below.

### 3.1 Distributional Semantic Models

For our experiments, we make use of the pretrained GloVe embeddings (Pennington et al., 2014) provided in the *Spacy*[1] package trained on the Common Crawl[2]. The GloVe model includes $685,000$ tokens with embedding vectors of dimension 300, providing excellent lexical coverage with a rich set of semantic representations.

In our analyses we use both the McRae property norms (McRae et al., 2005), which contain $541$ concepts and $2526$ features, and the CSLB norms (Devereux et al., 2014) which have $638$ concepts with $2725$ features. For both sets of norms, a feature is listed for a concept if it has been elicited by five or more participants in the property norming study. The number of participants listing a given feature for a given concept name is termed the *production frequency* for that concept×feature pair. This gives sparse production frequency vectors for each concept over all the features in the norms.

### 3.2 Partial Least Square Regression (PLSR)

Fagarasan et al. (2015) used partial least squares regression (PLSR) to map between the GloVe embedding space and property norm vectors. Suppose we have two real-valued matrices $G \in \mathbb{R}^{n \times m}$ and $F \in \mathbb{R}^{n \times k}$. In this context, $G$ and $F$ represent GloVe embedding vectors and property norm feature vectors, respectively. For $n$ available concept words, $G$ is a matrix which consists of stacked pretrained embeddings from GloVe and $F$ is the (sparse) matrix of production frequencies for each concept×feature pair. $G$ and $F$ share the same row indexing for concept words. For a new dimension size $p \in \mathbb{N}$, a partial least squared regression learns two new subspaces with dimensions $n \times p$, which have maximal covariance between them. The algorithm solves this problem by learning a mapping from the matrix $G$ onto $F$, similar to a regression model. The fitted regression model thus provides a framework for predicting vectors in the feature space from vectors in the embedding space.

In this work, we use the PLSR approach as a baseline for our model. In implementing PLSR, we set the intermediate dimension size to $50$, following Fagarasan et al. (2015). We also build a PLSR model using $120$ dimensions, which in preliminary experimentation we found gave the best

[1] https://spacy.io/models/en#en_core_web_lg
[2] http://commoncrawl.org/

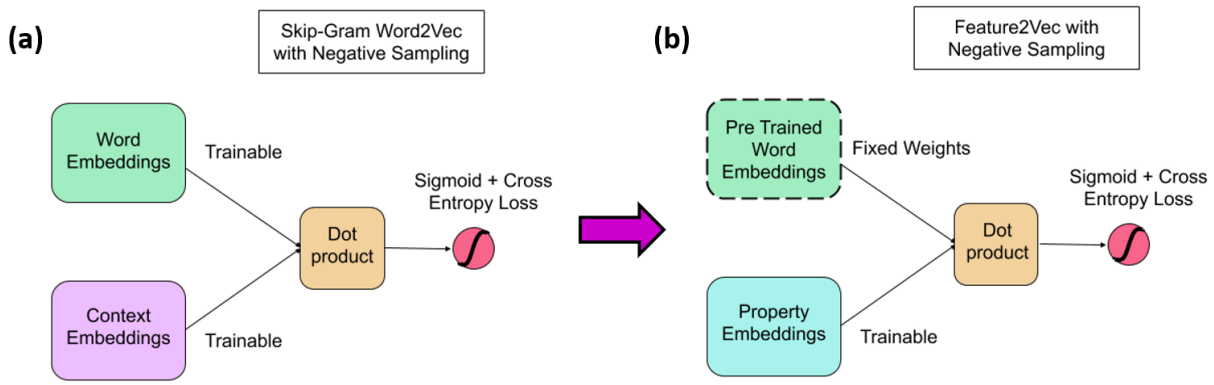


Figure 1: Diagram illustrating the relationship of *Feature2Vec* to the standard Skip-Gram Word2Vec architecture. **(a)** The standard Skip-Gram Word2Vec architecture with negative sampling. Word and context embeddings are randomly initialized, and both sets of embeddings are trainable. **(b)** Our Feature2Vec architecture, which uses pretrained word embeddings that are fixed during training. The network learns semantic feature embeddings for feature labels in property norm datasets instead of context embeddings.

performance from a range of values tested.

### 3.3 Skip-Gram Word2Vec

Mikolov et al. (2013) proposed learning word embeddings using a predictive neural-network approach. In particular, the skip-gram implementation with negative sampling mines word co-occurrences within a text window, and the network must learn to predict the surrounding context from the target word. More specifically, for a vocabulary $V$, two sets of embeddings are learned through gradient decent, one for target embeddings and one for context embeddings. Given a target word $w \in V$ and a context word $c \in V$ in its window, the network calculates the dot product for the embeddings for $w$ and $v$ and a sigmoid activation is applied to the output (Fig. 1(a)). Negative samples are also generated for training, where the context is not in the target word's window. Let $(w, c) \in D$ be the positive word and context pairs and $(w, c) \in D'$ be the negative word and context pairs. Then, using binary cross entropy loss, we learn a parameterization $\theta$ of the neural network that maximizes the function

$$\mathcal{J}(\theta) = \sum_{(w,c)\in D} log(\sigma(v_c \cdot v_w)) + \sum_{(w,c)\in D'} log(\sigma(-v_c \cdot v_w))$$

where $\sigma$ is the sigmoid function and $v_w$ and $v_c$ are the corresponding real-valued embeddings for the target words and context words (Goldberg and Levy, 2014).

In this work, we adapt this skip-gram approach to the task of constructing semantic representations of human property norms by mapping properties into an embedding space (Figure 1). We achieve this by using a neural network to predict the properties from the input word, using the skip-gram architecture with negative sampling on the properties. We replace context embeddings and windowed co-occurrence counts from the conventional skip-gram architecture with property embeddings and concept-feature production frequencies. The loss function for training remains the same; however, there are two modifications to the learning process. The first is that the target embeddings for the concept words are pre-trained (i.e. the GloVe embeddings), and gradients are not applied to this embedding matrix. The layer for the property norms is randomly initialized, and gradients are applied to these vectors to learn a semantic representation for properties aligned to the pre-trained distributional semantic space for the words. Secondly, the negative samples are generated from randomly sampled properties. We downweight negative samples by multiplying their associated loss by one over the negative sampling rate, so that the system pays more attention to real cases and less to the incorrect negative examples. Due to the sparsity of word-feature production frequencies, we generate all positive instances and randomly sample negative examples after each epoch to create a new set of training samples. We name this approach *Feature2Vec*[3]. We use a learn-

[3]The code for the model is available at https://github.com/stevend94/Feature2Vec

| | McRae | | | | CSLB | | | |
|---|---|---|---|---|---|---|---|---|
| Model | Top 1 | Top 5 | Top 10 | Top 20 | Top 1 | Top 5 | Top 10 | Top 20 |
| PLSR 50 | 3.55 | 14.18 | 29.79 | 47.52 | 2.90 | 23.19 | 44.20 | 60.87 |
| PLSR 120 | 3.55 | 25.53 | 43.26 | 53.90 | 7.25 | 34.78 | 55.80 | 71.74 |
| Feature2Vec | 4.96 | 34.75 | 45.39 | 60.99 | 4.35 | 36.23 | 53.62 | 76.09 |

Table 1: Accuracy scores (percent) for retrieving the correct concept from amongst the top $N$ most similar nearest neighbours. For PLSR, predicted feature vectors for test concepts are compared to actual feature vectors. For Feature2Vec, predicted concept word embeddings are compared to the actual GloVe embedding vectors.

ing rate of 0.001 and Adam optimization to train for 120 epochs. We use a negative sampling rate of 20 for both the McRae and CLSB datasets. In contrast to the PSLR approach, *Feature2Vec* learns a mapping from the feature norms onto the GloVe embedding space.

## 4 Experiments

We train with the McRae and CSLB property norms separately and report evaluations for each dataset. For the McRae dataset we use 400 randomly selected concepts for training and the remaining 141 for testing, and for the CSLB dataset we use 500 randomly selected concepts for training and the remaining 138 for testing[4].

### 4.1 Predicting Feature Vectors

We first evaluate how well the baseline PLSR model performs on the feature vector reconstruction task used by Fagarasan et al. (2015). In this evaluation, the feature vector for a test concept is predicted and we test whether the real concept vector is within the top $N$ most similar neighbours of the predicted vector. We report results over both 50 (as in Fagarasan et al. (2015)) and 120 dimensions for a range of values of $N$ (Table 1).

### 4.2 Constructing Concept Representations from Feature Vectors

For *Feature2Vec*, we embed property norm features into the GloVe semantic space, giving a representation of properties in terms of GloVe dimensions. To predict a held-out concept embedding, we build a representation of the concept word by averaging the learned feature embedding vectors for that word using the ground truth information from the property norm dataset. This gives a method to construct embeddings for new words using property knowledge and associated production frequencies (for example, for a held-out word *unicorn*, its GloVe embedding vector might be predicted from all features of *horse*, along with the features *is-white*, *has-a-horn*, and *is-fantastical*). We compare these predicted embeddings to the held-out Glove embeddings (Table 1). However, we note that this approach is different to the PLSR models, so we do not make a direct comparison between PLSR and Feature2vec nearest neighbour results. Nevertheless, the results show that the word embeddings composed from the learned *Feature2Vec* feature embeddings appear relatively frequently amongst the most similar neighbour words in the pretrained GloVe space, indicating that feature embedding composition approximates the original word embedddings reasonably well.

| | McRae | | CSLB | |
|---|---|---|---|---|
| Model | Train | Test | Train | Test |
| PLSR 50 | 49.52 | 31.67 | 50.58 | 40.25 |
| PLSR 120 | 68.66 | 32.97 | 65.42 | 40.71 |
| Feature2Vec | 90.70 | 35.33 | 90.31 | 44.30 |

Table 2: The average percentage of features that each method can predict for a given concept vector.

### 4.3 Predicting Property Knowledge

The evaluation task that we are most interested in is how well the models can predict feature knowledge for concepts, given the distributional semantic vectors. More specifically, for a given concept with $K$ features, we wish to take the top $K$ predicted features according to each method, and record the overlap with the true property norm listing. In this evaluation, we make direct comparisons between all three models (PLSR 50, PLSR 120, & Feature2Vec). For the PLSR models, we predict the feature vector for a given target word using the embedding vector as input and take the top $K$ weighted features. For Feature2Vec, we

[4]We use the Python pickle package to store the numpy state for reproducible results in our code.

| Concepts | Properties |
|---|---|
| Kingfisher | **has_wings** **does_fly** **has_a_beak** **has_feathers** **is_a_bird** **does_eat** has_a_tail does_swim **has_legs** does_lay_eggs |
| Avocado | **is_eaten_edible** **is_tasty** **does_grow** **is_green** **is_healthy** is_used_in_cooking **has_skin_peel** is_red is_food **is_a_vegetable** |
| Door | made_of_metal has_a_door_doors is_useful has_a_handle_handles made_of_wood made_of_plastic is_heavy is_furniture does_contain_hold is_found_in_kitchens |
| Dragon | is_big_large is_an_animal has_a_tail does_eat is_dangerous has_legs has_claws is_grey is_small does_fly |

Table 3: Top 10 predictions from CSLB-trained Feature2Vec model. The properties in bold and underlined are those that agree with the available ground truth features from the CSLB dataset. The first two concepts are from the CSLB test set, whilst the final two words were randomly sampled from the word embedding lexicon.

rank all feature embeddings by their distance to the embedding for the target word, using cosine similarity, and take the top $K$ most similar features (Table 2). The results demonstrate that Feature2Vec outperforms the PLSR models on property knowledge prediction, for both training and testing datasets.

### 4.4 Analysis

Following previous work, we provide the top 10 feature predictions for a few sample concepts, displayed in Table 3. Properties underlined and in bold represent features that match the available ground truth data (i.e., the concept×feature pair occurs in the norms). The first two words in Table 3 were sampled from the CSLB norms test set, whilst the last two words were randomly sampled from the word embedding lexicon and are not concept words appearing in the CSLB norms. We find that the predicted features that are not contained within the ground truth property set still tend to be quite reasonable, even for the two concepts not in the test dataset. As property norms do not represent an exhaustive listing of property knowledge, this is not surprising, and predicted properties not in the norms are not necessarily errors (Devereux et al., 2009; Fagarasan et al., 2015). Moreover, the set of features used within the norms are dependent on the concepts that were presented to the human participants. It is therefore notable that the conceptual representations predicted by our model for the two out-of-norms concept words are particularly plausible, even though the attributes were never intended to conceptually represent these words. Our analysis supports the view that such supervised models could be utilised as an assistive tool for surveying much larger vocabularies of words.

## 5 Conclusion

We proposed a method for constructing distributional semantic vectors for human property norms from a pretrained vector space model of word meaning, which outperforms previous methods for predicting concept features on two property norm datasets. As discussed by Fagarasan et al. (2015) and others, it is clear that property norm datasets provide only a semi-complete picture of human conceptual knowledge, and more extensive surveys may provide additional useful property knowledge information. By predicting plausible semantic features for concepts through the leveraging of corpus-derived word embedding data, our method offers a useful tool for guiding the expensive and laborious process of collecting property norm listings. For example, existing property norm datasets can be extended through human verification of features predicted with high confidence by Feature2Vec, with these features being added to the norms and subsequently incorporated into Feature2Vec in an iterative, semi-supervised manner (Kelly et al., 2012). Thus, Feature2Vec provides a useful heuristic to add interpretable feature-based information to these datasets for new words in a practical and efficient way.

## References


Marco Baroni and Alessandro Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4):673–721.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Guillem Collell and Marie-Francine Moens. 2016. Is an image worth more than a thousand words? on the fine-grain semantic differences between visual and linguistic representations. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2807–2817.

Steven Derby, Paul Miller, Brian Murphy, and Barry Devereux. 2018. Using sparse semantic embeddings learned from multimodal text and image data to model human conceptual knowledge. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 260–270, Brussels, Belgium. Association for Computational Linguistics.

Barry Devereux, Nicholas Pilkington, Thierry Poibeau, and Anna Korhonen. 2009. Towards unrestricted, large-scale acquisition of feature-based conceptual representations from corpus data. *Research on Language and Computation*, 7(2):137–170.

Barry J Devereux, Lorraine K Tyler, Jeroen Geertzen, and Billi Randall. 2014. The Centre for Speech, Language and the Brain (CSLB) concept property norms. *Behavior Research Methods*, 46(4):1119–1127.

Luana Fagarasan, Eva Maria Vecchi, and Stephen Clark. 2015. From distributional semantics to feature norms: grounding semantic models in human perceptual data. In *Proceedings of the 11th International Conference on Computational Semantics*, pages 52–57.

Yoav Goldberg and Omer Levy. 2014. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*.

Colin Kelly, Barry Devereux, and Anna Korhonen. 2012. Semi-supervised learning for automatic conceptual property extraction. In *Proceedings of the 3rd Workshop on Cognitive Modeling and Computational Linguistics (CMCL 2012)*, pages 11–20, Montréal, Canada. Association for Computational Linguistics.

Colin Kelly, Barry Devereux, and Anna Korhonen. 2014. Automatic extraction of property norm-like data from large text corpora. *Cognitive Science*, 38(4):638–682.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 302–308.

Dandan Li and Douglas Summers-Stay. 2019. Mapping distributional semantics to property norms with deep neural networks. *Big Data and Cognitive Computing*, 3(2):30.

Lucy Li and Jon Gauthier. 2017. Are distributional representations ready for the real world? Evaluating word vectors for grounded perceptual meaning. In *Proceedings of the First Workshop on Language Grounding for Robotics*, pages 76–85, Vancouver, Canada. Association for Computational Linguistics.

Ken McRae, George S Cree, Mark S Seidenberg, and Chris McNorgan. 2005. Semantic feature production norms for a large set of living and nonliving things. *Behavior Research Methods*, 37(4):547–559.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Brian Murphy, Partha Talukdar, and Tom Mitchell. 2012. Learning effective and interpretable semantic models using non-negative sparse embedding. In *Proceedings of COLING 2012*, pages 1933–1950, Mumbai, India. The COLING 2012 Organizing Committee.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Dana Rubinstein, Effi Levi, Roy Schwartz, and Ari Rappoport. 2015. How well do distributional models capture different types of semantic knowledge? In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 726–730.

L. K. Şenel, I. Utlu, V. Yücesoy, A. Koç, and T. Çukur. 2018. Semantic structure and interpretability of word embeddings. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(10):1769–1779.

Carina Silberer. 2017. Grounding the meaning of words with visual attributes. In *Visual Attributes*, pages 331–362. Springer.

Pia Sommerauer and Antske Fokkens. 2018. Firearms and tigers are dangerous, kitchen knives and zebras are not: Testing whether word embeddings can

tell. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Brussels, Belgium. Association for Computational Linguistics.

# Sunny and Dark Outside?! Improving Answer Consistency in VQA through Entailed Question Generation

[1]Arijit Ray, [1]Karan Sikka, [1]Ajay Divakaran, [*]Stefan Lee, [1]Giedrius Burachas
[1]first.last@sri.com, [*]steflee@gatech.edu
[1]SRI International, [*]Georgia Institute of Technology

## Abstract

While models for Visual Question Answering (VQA) have steadily improved over the years, interacting with one quickly reveals that these models lack consistency. For instance, if a model answers "red" to "What color is the balloon?", it might answer "no" if asked, "Is the balloon red?". These responses violate simple notions of entailment and raise questions about how effectively VQA models ground language. In this work, we introduce a dataset, ConVQA, and metrics that enable quantitative evaluation of consistency in VQA. For a given observable fact in an image (e.g. the balloon's color), we generate a set of logically consistent question-answer (QA) pairs (e.g. Is the balloon red?) and also collect a human-annotated set of common-sense based consistent QA pairs (e.g. Is the balloon the same color as tomato sauce?). Further, we propose a consistency-improving data augmentation module, a Consistency Teacher Module (CTM). CTM automatically generates entailed (or similar-intent) questions for a source QA pair and fine-tunes the VQA model if the VQA's answer to the entailed question is consistent with the source QA pair. We demonstrate that our CTM-based training improves the consistency of VQA models on the ConVQA datasets and is a strong baseline for further research.

## 1 Introduction

"A skeptic, I would ask for consistency first of all."

*Sylvia Plath (Plath, 2007)*

Visual Question Answering (VQA) (Antol et al., 2015) involves answering natural language questions about images. Despite the recent progress on VQA, we observe that existing methods are prone to making blatant mistakes while answering questions regarding the same visual fact but from slightly different perspectives (Figure 1). This reveals a critical limitation of the state-of-the-art models in maintaining consistency.

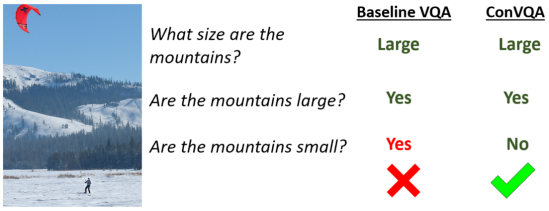


Figure 1: Current VQA models often fail at consistently answering semantically rephrased questions. To address this limitation, we construct a consistent VQA (ConVQA) dataset with diverse QA pairs that query the same visual fact. We also propose a Consistency Teacher Module (CTM) that improves VQA consistency by rewarding consistent behavior.

In particular, we motivate our definition of consistency based on classical deductive logic (Tarski and Tarski, 1994) that defines a consistent theory as one that does not entail a contradiction. Correspondingly, we define consistency, in the context of VQA, as being able to answer questions posed from different semantic perspectives about a certain fact without any contradiction. In addition, consistent Question-Answer (QA) pairs can be derived based on simple notions of logic or by commonsense reasoning. For instance, say an image contains a "large building". Logic-based QA pairs can be "is the building small? no" and "what size is the building? large". On the other hand, if an image contains "vegetarian pizza", commonsense-based QA pairs can be "is it a vegetarian pizza? yes" and "is there pepperoni on the pizza? no", which requires commonsense knowledge that "pepperoni" is not vegetarian.

While attempts have been made to construct logic-based consistent VQA datasets (Hudson and Manning, 2019), they still fall short on commonsense-based consistency. To this end, our ConVQA Dataset consists of two subsets: 1) a challenging human-annotated set comprised of commonsense-based consistent QA's (shown in



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5860–5865, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

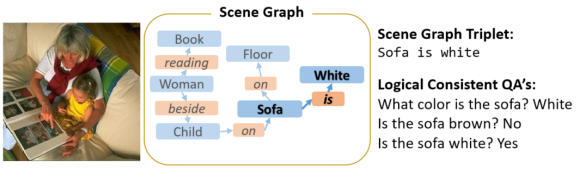


Figure 2: Qualitative examples from our ConVQA dataset derived based on logic.

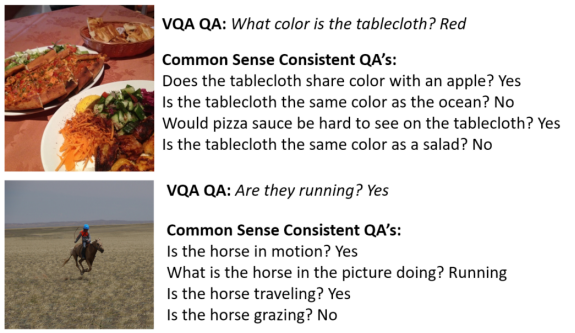


Figure 3: Qualitative examples from our human-annotated commonsense-based ConVQA Dataset.

Figure 3), and 2) an automatically generated logic-based consistent QA dataset (shown in Figure 2).

To improve the consistency of VQA models, we propose a Consistency Teacher Module (CTM), which consists of a Question Generator that synthesizes entailed (or similar-intent) questions given a seed QA pair and a Consistency Checker that examines whether answers to those similar-intent questions are consistent. For training a consistent VQA model, our CTM acts as a consistency-based data augmentation scheme that trains a VQA model with consistent answers to entailed questions. We demonstrate that our approach improves the performance of a baseline VQA model on our ConVQA testing sets in terms of both accuracy and consistency. Our datasets and models will be available at https://bit.ly/32exlM7.

## 2 Related Work

Checking for consistency can be considered as an interrogative Turing Test (Radziwill and Benton, 2017) for linguistic robustness (Stede, 1992), (Li et al., 2017). Works such as Xu et al. (2018) explore the robustness of VQA with respect to image variations, whereas works such as Ray et al. (2016) and Mahendru et al. (2017) focus on the understanding of the premise of a question instead of relying on dataset biases (Agrawal et al., 2017) (Goyal et al., 2017) or linguistic biases (Ramakrishnan et al., 2018).

Recently, the research community has shown great interest in evaluating VQA for consistency and plausibility. GQA (Hudson and Manning, 2019) is established as a scene-graph based QA dataset. Their questions, similar to Johnson et al. (2017), require multiple hops of reasoning, and are not validated or annotated by humans. Our ConVQA differs from GQA in the following two aspects. First, we provide a human-validated test set of the automatically generated logic-based consistent QA's for a more accurate performance evaluation. Second, we collect human-annotated QA pairs based on common-sense in addition to the logic-based QA's. The most relevant work to ours is Shah et al. (2019). However, they focus strictly on question paraphrases that maintains the same answers as the source question. We, however, focus on generating questions which can have different answers, but are about the same visual fact, which greatly increases the diversity of the resulting QA pairs. To the best of our knowledge, the proposed ConVQA dataset is the first consistent QA dataset that contains human-annotated consistent QA's based on common-sense.

Other works have also looked into question generation (Zhang et al., 2016), (Mostafazadeh et al., 2016) for training better VQA models. In Misra et al. (2017), QA pairs are obtained from an oracle in a simulated environment. In contrast, our CTM-based training operates on real images and uses a learned consistency measure to train the VQA module with consistent QA's.

## 3 ConVQA Datasets

The consistent QA pairs in our ConVQA are generated automatically based on simple notions of logical consistency or are human-annotated using commonsense reasoning.

**Logic-based Consistent QA. (L-ConVQA)** Consider the Visual Genome (Krishna et al., 2017) scene graph in Figure 2 consisting of objects, attributes, and their relationships. We consider each triplet to encode a single 'visual fact', for instance, that the sofa is white. We employ slot-filler NLP techniques to generate a set of QA pairs for each triplet (object-relation-subject) in the scene graph. Currently, we focus on attribute (e.g., color, size), existential (e.g., is there) and relational (e.g., sofa on floor) consistency. We leverage Wordnet (Miller, 1995) and a manually generated list of antonyms (e.g., white vs. black) and hypernyms (e.g., white $\rightarrow$ color) to generate these QA pairs.

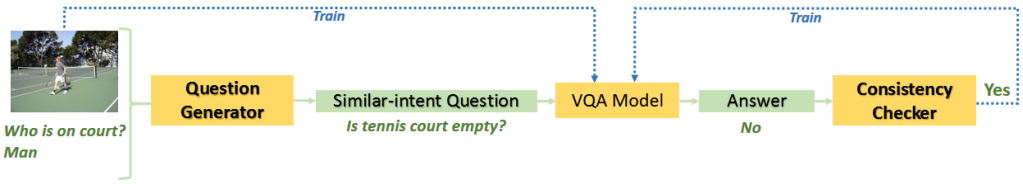


Figure 4: Block diagram of the proposed CTM including a Question Generator that synthesizes questions with similar intent and a Consistency Checker that classifies QA pairs as consistent, unrelated, or contradictory. CTM finetunes VQA models via reinforcement learning with the answer consistency as the reward to encourage VQA models to answer rephrased questions more consistently. The examples shown are from a real run.

For example, for the attribute "white" of an object "cup", we generate QA pairs such as "is the cup white? yes", "is the cup black? no" and "what color is cup? white". We also filter objects and relationships by frequency and saliency (e.g., based on bounding boxes) to avoid non-salient and infrequent objects or noisy relationships. We have a total of 880,141 QA pairs in 255,910 sets on 70,292 images. We split the data into a training set with 47,999 images, a validation set with 9,993 images, and a test set with 12,300 images. Notably, we create a smaller clean test set (12,325 QA pairs on 725 images) using Amazon Mechanical Turk (AMT) where three independent workers were asked to remove incorrect or unnatural QA's.

**Commonsense-based Consistent QA.** While the logic-based consistent QA set provides a first step into large-scale examination of VQA consistency, the generated questions require limited reasoning and commonsense and are, therefore, frequently simpler than human-annotated ones. Hence, we collect more challenging QA pairs based on commonsense (**CS-ConVQA**) by asking AMT workers to write intelligent rephrases of QA pairs sampled from the VQA2.0 (Goyal et al., 2017) validation dataset. AMT workers were instructed to avoid simple word paraphrases and instead to write rephrases that require commonsense reasoning in order to answer the question consistently. We collect approximately 3.5 consistent QA pairs per image for 6439 images. After filtering images that overlap with the training set of the L-ConVQA subset, we split this data into a training set (1568 images), a validation set (450 images), and a test set (1590 images).

## 4 Approach

To improve VQA consistency, we propose training a VQA model using a Consistency Teacher Module (CTM) that generates entailed questions and performs a consistency-based data augmentation. More specifically, CTM consists of two trainable components – an entailed question generator and a consistency checker.

**Entailed Question Generator.** For a given a source question-answer pair, we define entailed questions as those for which the answer should be obvious given the source QA pair. For example, given the source QA pair "Who is on court? Man", an appropriate entailed question might be "Is the tennis court empty?". We train a question generation model that given representations of the image and the source QA pair, generates a new question. Specifically, our question generator concatenates the deep features of an image (extracted using a ResNet152 (He et al., 2016) network) and a QA pair (extracted using a 1-layer LSTM (Hochreiter and Schmidhuber, 1997)) to represent a visual fact. These features are fed into another LSTM model to generate a similar-intent question. We train this module on the automatically generated Logical L-ConVQA train set. We also include some closely related (according to averaged Word2Vec (Mikolov et al., 2013) distance) Visual Genome (Krishna et al., 2017) QA pairs in the training of the question generator to add some diversity to the generated questions.

**Consistency Checker.** Once the VQA model produces an answer for the generated entailed question, it may or may not be consistent with the source question. To evaluate this and provide feedback to the VQA model, we train a consistency checker that processes the image and both the source and entailed QA pairs. Similar to the question generator architecture, the consistency checker takes in deep features of the image and two QA pairs and classifies the QA pairs as consistent, inconsistent or unrelated. This model is trained using the automatically generated L-ConVQA train set alone. Inconsistent examples in the L-ConVQA set are made using simple techniques such as flipping yes/no answers and replac-




Figure 5: Qualitative examples of entailed question generation and consistency checking for training the VQA.

ing entities in the scene graph triplets.

**Consistency Teacher Module (CTM).** Putting these two components together, we can train a VQA model based on its consistency on generated entailed questions. Figure 4 shows our pipeline. During training, for each source VQA QA pair ("Who is on court? Man"), we generate an entailed question ("Is tennis court empty?") and produce the VQA model's answer ("No"). We then run the consistency checker to determine if the generated answer is consistent with the source QA (in this case, "Who is on court? Man"). If the answer is consistent (and VQA confidence $> 0.7$), we treat it as the ground truth for the entailed question and update the VQA model as if this example were part of the original dataset. Likewise if it is deemed inconsistent, or if the question is deemed unrelated, it is unclear what the correct answer should be, so we do not update the model.

## 5 Experiments

To evaluate our approach, we apply the Consistency Teacher Module (CTM) module to a state-of-the-art VQA model trained on VQAv2 and evaluate performance on the ConVQA datasets. We describe training procedures, metrics, and baselines in this section.

**Consistency Teacher Module Training.** We train the components of CTM – Entailed Question Generation and Consistency Checker – using only the synthetic L-ConVQA train set (referred as the standard **CTM**) or a mix of Visual Genome and L-ConVQA train (referred as **CTMvg**) and keep them frozen when fine-tuning the VQA model. When we train the VQA, the sets used to finetune the VQA or seed the CTM come from splits not




Figure 6: Qualitative examples of our improved VQA consistency by CTM training compared to baseline bottom-up VQA. Green is correct, Red is wrong. GT is Ground Truth.

seen during training of the CTM - val split of L-ConVQA and train split of CS-ConVQA.

Qualitative examples of our Entailed Question Generator trained only on L-ConVQA are shown in Figure 5. Despite only being trained on the automatically-generated L-ConVQA data, it generates reasonably well-entailed questions on human-annotated questions.

Our Consistency Checker has a high accuracy of classification on the L-ConVQA test set (90%). However, when tested with a mix of commonsense-based CS-ConVQA, the accuracy drops to 64% (chance is 33% for 3 classes). We find that precision is important when training the VQA using the pre-trained Consistency Checker. Hence, we use the classifier at above 90% confidence threshold, where the precision is 70.38%.

**Evaluation Metrics for ConVQA.** We report three metrics for ConVQA – capturing notions of consistency and performance.

– **Perfect-Consistency (Perf-Con).** A model is perfectly consistent for a question set if it answers all questions in the set correctly. We report the percentage of such sets as Perf-Con.
– **Average Consistency (Avg-Con).** We also report the average accuracy within a consistent question set over the entire dataset as Avg-Con.
– **Accuracy (top-1).** Finally, we report the top-1 accuracy over all questions in the dataset.

**Baselines.** We compare to a number of baseline models to put our CTM results into context:

Table 1: Performance comparison of baseline VQA trained on VQA2.0, baseline VQA finetuned on ConVQA, and VQA trained using our CTM. **L-ConVQA** is the human-cleaned Logical Consistent QA dataset, **CS-ConVQA** is the human annotated Common-sense Consistency Dataset and **VG** is Visual Genome. CTM-based training produces the best results in terms of overall accuracy and consistency. **DATA** denotes the data used to fine-tune VQA or seed the CTM question generator.

| | DATA | L-ConVQA | | | CS-ConVQA | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | **Perf Con** | **Avg Con** | **Top1** | **Perf Con** | **Avg Con** | **Top1** | **Yes/No** | **Num** |
| a) VQA | VQA2.0 | 36.25 | 71.36 | 70.34 | 26.13 | 59.61 | 60.03 | 65.49 | 31.39 |
| b) FineTune | CS-ConVQA | 34.54 | 70.39 | 69.48 | 26.39 | 59.65 | 60.07 | 65.80 | 35.92 |
| c) FineTune | L/CS-ConVQA | **54.68** | **83.42** | **83.16** | 24.70 | 59.30 | 59.60 | 65.14 | 33.33 |
| d) **+CTM** | L/CS-ConVQA | 54.6 | 83.23 | 82.79 | 25.94 | **60.39** | **60.78** | **66.63** | **36.89** |
| e) FineTune | L/CS-ConVQA,VG | 36.40 | 71.60 | 70.94 | 25.22 | 59.19 | 59.56 | 65.30 | 31.39 |
| f) **+CTMvg** | L/CS-ConVQA,VG | 51.41 | 81.66 | 81.37 | **27.49** | 59.75 | 60.15 | 66.41 | 34.95 |

– **VQA.** We take the bottom-up top-down VQA model (Anderson et al., 2018) as our base model for these experiments. To evaluate consistency in existing models, we present results on ConVQA of this model pretrained on VQA2.0.
– **Finetuned models:** We present results for models finetuned on ConVQA and Visual Genome – **Finetune CS-ConVQA** finetuned on the commonsense ConVQA dataset, **Finetune L/CS-ConVQA** on both logical and commonsense ConVQA, and **Finetune L/CS-ConVQA,VG** extending to Visual Genome questions.

When we apply our CTM model to the finetuned baselines above, we seed the question generator with the associated dataset.

## 6 Results and Analysis

Table 1 shows quantitative results on our L-ConVQA and CS-ConVQA datasets. We make a number of observations below.

**The state-of-the-art VQA has low consistency.** The baseline VQA system **(row a)** retains similarly high top-1 accuracy on the ConVQA splits (63.58% on VQAv2 vs 70.34% / 60.03% on L-ConVQA / CS-ConVQA); however, it achieves only 26.13% perfect consistency on the human generated CS-ConVQA questions.

**Finetuning is an effective strategy for the synthetic L-ConVQA split.** Finetuning on L-ConVQA train results in 18.43% gains in perfect consistency on L-ConVQA test (**row c vs a**). This is unsurprising given the templated questions and simple concepts in L-ConVQA; however, perfect consistency is low in absolute terms at 54.68%.

**Finetuning does not lead to significant gains in consistency for human-generated questions.** Finetuning the VQA model on CS-ConVQA **(row b)** leads to an improvement in consistency of only 0.26%. Likewise, adding L-ConVQA **(row c)** and extra Visual Genome questions **(row e)** actually reduces consistency.

**CTM-based training preserves or improves consistency when leveraging additional data.** When we apply CTM to the Finetuned L/CS-ConVQA model, we improve CS-ConVQA perfect consistency by 1.24% **(row d vs c)** while modestly improving other metrics. Extending to Visual Genome questions, the CTM augmented model improves perfect consistency in CS-ConVQA by 2.27% over the finetuned model **(row f vs e)**. Interestingly, the CTM modules were never trained with the human-annotated CS-ConVQA questions and yet lead to this improvement on CS-ConVQA by acting as an intelligent data augmenter/regularizer.

## 7 Conclusion and Discussion

In this paper, we introduced a ConVQA dataset consisting of logic-based and commonsense-based consistent QA pairs about visual facts in an image. We also proposed a Consistency Teacher Module that acts as a consistency-based data augmenter to teach VQA models to answer consistently. As future work, we plan to look into improving our automatically generated consistent QA pairs using external knowledge-bases.

## Acknowledgements

We thank Yi Yao, Xiao Lin, Anirban Roy, Avi Ziskind and the anonymous reviewers for their helpful comments. This research was developed under DARPA XAI. The views, opinions and/or findings expressed are those of the authors' and should not be interpreted as representing the official views/policies of the DoD / U.S. Govt.

## References


Aishwarya Agrawal, Aniruddha Kembhavi, Dhruv Batra, and Devi Parikh. 2017. C-vqa: A compositional split of the visual question answering (vqa) v1. 0 dataset. *arXiv preprint arXiv:1704.08243*.

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6077–6086.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2425–2433.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *CVPR*, volume 1, page 3.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Drew A Hudson and Christopher D Manning. 2019. Gqa: a new dataset for compositional question answering over real-world images. *arXiv preprint arXiv:1902.09506*.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2901–2910.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123(1):32–73.

Yitong Li, Trevor Cohn, and Timothy Baldwin. 2017. Robust training under linguistic adversity. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 21–27.

Aroma Mahendru, Viraj Prabhu, Akrit Mohapatra, Dhruv Batra, and Stefan Lee. 2017. The promise of premise: Harnessing question premises in visual question answering. *arXiv preprint arXiv:1705.00601*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Ishan Misra, Ross Girshick, Rob Fergus, Martial Hebert, Abhinav Gupta, and Laurens van der Maaten. 2017. Learning by asking questions. *arXiv preprint arXiv:1712.01238*.

Nasrin Mostafazadeh, Ishan Misra, Jacob Devlin, Margaret Mitchell, Xiaodong He, and Lucy Vanderwende. 2016. Generating natural questions about an image. *arXiv preprint arXiv:1603.06059*.

Sylvia Plath. 2007. *The unabridged journals of Sylvia Plath*. Anchor.

Nicole M Radziwill and Morgan C Benton. 2017. Evaluating quality of chatbots and intelligent conversational agents. *arXiv preprint arXiv:1704.04579*.

Sainandan Ramakrishnan, Aishwarya Agrawal, and Stefan Lee. 2018. Overcoming language priors in visual question answering with adversarial regularization. In *Advances in Neural Information Processing Systems*, pages 1541–1551.

Arijit Ray, Gordon Christie, Mohit Bansal, Dhruv Batra, and Devi Parikh. 2016. Question relevance in vqa: identifying non-visual and false-premise questions. *arXiv preprint arXiv:1606.06622*.

Meet Shah, Xinlei Chen, Marcus Rohrbach, and Devi Parikh. 2019. Cycle-consistency for robust visual question answering. *arXiv preprint arXiv:1902.05660*.

Manfred Stede. 1992. The search for robustness in natural language understanding. *Artificial Intelligence Review*, 6(4):383–414.

Alfred Tarski and Jan Tarski. 1994. *Introduction to Logic and to the Methodology of the Deductive Sciences*. 24. Oxford University Press on Demand.

Xiaojun Xu, Xinyun Chen, Chang Liu, Anna Rohrbach, Trevor Darrell, and Dawn Song. 2018. Fooling vision and language models despite localization and attention mechanism. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4951–4961.

Shijie Zhang, Lizhen Qu, Shaodi You, Zhenglu Yang, and Jiawan Zhang. 2016. Automatic generation of grounded visual questions. *arXiv preprint arXiv:1612.06530*.

# GeoSQA: A Benchmark for Scenario-based Question Answering in the Geography Domain at High School Level

**Zixian Huang, Yulin Shen, Xiao Li, Yuang Wei, Gong Cheng, Lin Zhou, Xinyu Dai, Yuzhong Qu**
National Key Laboratory for Novel Software Technology, Nanjing University, China
{zixianhuang,ylshen,xiaoli,weiyuang}@smail.nju.edu.cn,
{gcheng,daixinyu,yzqu}@nju.edu.cn,zhoul@nlp.nju.edu.cn

## Abstract

Scenario-based question answering (SQA) has attracted increasing research attention. It typically requires retrieving and integrating knowledge from multiple sources, and applying general knowledge to a specific case described by a scenario. SQA widely exists in the medical, geography, and legal domains—both in practice and in the exams. In this paper, we introduce the GeoSQA dataset. It consists of 1,981 scenarios and 4,110 multiple-choice questions in the geography domain at high school level, where diagrams (e.g., maps, charts) have been manually annotated with natural language descriptions to benefit NLP research. Benchmark results on a variety of state-of-the-art methods for question answering, textual entailment, and reading comprehension demonstrate the unique challenges presented by SQA for future research.

## 1 Introduction

Scenario-based question answering (SQA) is an emerging application of NLP (Lally et al., 2017). Different from traditional QA, a question in SQA is accompanied by a *scenario*, e.g., a patient summary in the medical domain asking for diagnosis or treatment. A scenario differs from a document given in the reading comprehension task where the answer can be extracted or abstracted from the document (Rajpurkar et al., 2016; Nguyen et al., 2016; Lai et al., 2017). SQA requires retrieving and integrating knowledge from multiple sources, and applying general knowledge to a specific case described by the scenario.

SQA has found application in many fields, especially in the legal domain (Ye et al., 2018; Luo et al., 2017; Zhong et al., 2018) and in high-school geography exams (Ding et al., 2018; Zhang et al., 2018). The latter is particularly challenging because a geographical scenario consists of both text and diagrams (e.g., maps, charts). Questions include city planning, climates, agriculture planning, transportation, etc. An example of a scenario and a question is presented in Figure 1.

Geographical SQA has posed great challenges to NLP and related research, ranging from scenario understanding to cross-modal knowledge integration and reasoning. However, there is a lack of large datasets and benchmarking efforts for this task. In this paper, we introduce GeoSQA—an SQA dataset in the geography domain consisting of 1,981 scenarios and 4,110 multiple-choice questions at high school level. In particular, each diagram has been manually annotated with a high-quality natural language description of its content, as illustrated in Figure 1. This labor-intensive effort significantly extends the use of GeoSQA, which can support visual SQA (using the diagrams), natural language based SQA (using the annotations of diagrams), and even the diagram-to-text research. We test the effectiveness of a variety of methods for question answering, textual entailment, and reading comprehension on GeoSQA. The results demonstrate its unique challenges, waiting for more effective solutions.

The remainder of the paper is organized as follows. Section 2 discusses related work. Section 3 describes the GeoSQA dataset. Section 4 reports benchmark results. Section 5 concludes the paper.

## 2 Related Work

### 2.1 Scenario-based Question Answering

Scenario-based question answering (SQA) is introduced by Lally et al. (2017), where the Watson-Paths system is presented to answer questions that describe a medical scenario about a patient and ask for diagnosis or treatment. SQA also finds application in the legal domain, where a legal case de-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5866–5871,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics




**Scenario:** The diagram shows the urban planning of a satellite city in 2010. The city commonly receives NW winds. Read the diagram and answer the question.

**Question:** Location M in the diagram is suitable for
A. Waterworks
B. Incineration plant
C. Sewage treatment plant **(the correct answer)**
D. Thermal power plant

**Diagram annotations --- templated text:**
M is to the north of the river.
M is in the northwest of the satellite city.

**Diagram annotations --- free-form text:**
The river flows north.

Figure 1: An example of a scenario, a question, and diagram annotations.

scribes a scenario to be decided (Ye et al., 2018; Luo et al., 2017; Zhong et al., 2018).

For some domains, reasoning with domain knowledge is essential to SQA. Therefore, such questions often appear in exams like China's version of the SAT called Gaokao. For example, for the geography domain, Ding et al. (2018) and Zhang et al. (2018) construct a knowledge graph to support answering scenario-based geography questions at high school level.

### 2.2 Related Datasets

There are many datasets for traditional QA, such as WebQuestions (Berant et al., 2013) and WikiQA (Yang et al., 2015). A closely related task is reading comprehension, where the answer to a question is extracted or abstracted from a given document (Rajpurkar et al., 2016; Nguyen et al., 2016; Lai et al., 2017). By comparison, SQA is arguably more difficult because a scenario is present and contextualizes a question, but no direct answer can be identified from the scenario.

The GeoSQA dataset introduced in this paper is not the first resource for geographical SQA. Ding et al. (2018) and Zhang et al. (2018) have made their datasets public. However, compared with GeoSQA, their datasets are small and, more importantly, they ignore diagrams which represent a unique challenge to geographical SQA. By contrast, diagrams are included in GeoSQA for completeness, and have been manually annotated with natural language descriptions for extended use—including but not limited to NLP research.

Existing SQA datasets for other domains include the TREC Precision Medicine track (Roberts et al., 2018) for the medical domain, and CAIL (Xiao et al., 2018) for the legal domain. However, SQA in the geography domain requires different forms of knowledge and different reasoning capabilities, and has posed different research challenges.

## 3 The GeoSQA Dataset

GeoSQA is an SQA dataset in the geography domain, containing 1,981 scenarios and 4,110 multiple-choice questions at high school level. A scenario consists of a piece of text and a diagram, supporting 1–5 questions. A diagram is annotated with a natural language description of its content. A question has four options that are possible answers. Exactly one of them is the correct answer. The dataset is available online[1].

### 3.1 Data Collection and Deduplication

We crawled over 6,000 scenarios and 13,000 questions from Gaokao and mock tests that are available on the Web. However, some scenarios are just copies or trivial variants of others. There is a need to clean and deduplicate the collected data.

**Method.** The problem is to decide whether a pair of scenarios are (near) duplicates or not.

We firstly establish a matching between their structures. The matching consists of 6 pairs of their text elements: 1 pair of their scenario text, 1 pair of their most similar questions, and 4 pairs of the most similar options of the above two questions. Text similarity is computed by the cosine similarity between two bags of words.

Then we extend a popular text matching method called MatchPyramid (Pang et al., 2016) to classify a pair of scenarios as duplicates or not. The original implementation of MatchPyramid can

[1] ws.nju.edu.cn/gaokao/geosqa/1.0

only process a pair of text. We extend it to process all the 6 pairs of text by concatenating their feature vectors inside MatchPyramid.

**Experiments.** To evaluate our method, we manually label 1,000 pairs of scenarios where positive and negative examples are balanced. The set is divided into training, validation, and test sets with a 60-20-20 split. Our method achieves an accuracy of 95.3% on the test set, showing its satisfying performance.

Then we apply our method to the entire dataset. We index all the scenarios using Apache Lucene. For each scenario, we retrieve 10 top-ranked scenarios as suspect duplicates. Each pair of scenarios is classified by our method, which is trained using all the 1,000 labeled examples.

To verify the quality of the final results, we randomly sample and manually check 100 pairs of scenarios that are predicted to be duplicates. Indeed, all of them are decided correctly. We also randomly sample 50 scenarios and, for each of them, we retrieve and manually check 10 top-ranked scenarios that are predicted to be non-duplicates. Only 6% are decided incorrectly, suggesting a low degree of redundancy of our data.

### 3.2 Diagram Annotation

Crawled diagrams are images. To extend the use of GeoSQA and to better support NLP research, we manually annotate each diagram with a high-quality natural language description of its content so that NLP researchers can use these text annotations instead of the original diagrams.

**Annotation.** We recruited 30 undergraduate students from one of the top-ranked universities in China as annotators. All of them had an excellent record in geography during high school.

Each diagram is assigned to one annotator, who also has access to the scenario text and related questions. The annotator firstly categorizes the diagram according to a hierarchy of categories. Each category is associated with a set of text templates that are recommended to be used in annotations as far as possible. However, the annotator is free to use any form of text to annotate information that is not covered by the provided templates.

Annotations are required to precisely reflect the content of the diagram. All the information related to every supported question and every option should be annotated. On the other hand, inferring new knowledge via human reasoning is prohibited.

Note that the entire annotation process is designed to be iterative. The 22 diagram categories and 81 text templates are not predefined but incrementally induced during the experiment. However, there are still 11% of the diagrams that are believed to not belong to any category. No templates are provided for their annotations.

An example of annotations is shown in Figure 1.

**Audit.** To ensure the quality of the annotations, we recruited 3 senior annotators to audit the results. Each diagram is audited by one senior annotator, who rates the annotations from three dimensions in the range of 1–5.

- Sufficiency: The annotations cover all the necessary information in the diagram that is useful for answering related questions.
- Fairness: The annotations are not biased towards any particular option of a question.
- Objectiveness: The annotations are plain descriptions of the diagram—not influenced by human reasoning.

The scenarios where the annotations of the diagram are rated below 3 in any dimension are excluded from the dataset.

## 4 Benchmark Results

We tested several state-of-the-art methods for question answering, textual entailment, and reading comprehension on our GeoSQA dataset.

### 4.1 Corpora

We use two corpora as background knowledge. **Textbooks** contains 15K sentences extracted from two high-school geography textbooks. **Wikipedia** contains 1M articles in the latest Chinese edition of Wikipedia. We index their sentences using Apache Lucene.

### 4.2 Methods

We tested two text matching methods. In **IR** (Clark et al., 2016), for each option, we use a combination of the scenario text, the question, and the option as a query, to retrieve the top-ranked sentence from a corpus. We use the ranking score of this sentence as the score of the option. Finally, we choose the option with the highest score as the answer. In **PMI** (Clark et al., 2016), for each option, we calculate the Pointwise Mutual Information (PMI) between the question and the option as

the score of the option. Finally, we choose the option with the highest score as the answer. Probabilities in PMI are estimated based on a corpus.

We tested four textual entailment methods: **ESIM** (Chen et al., 2017), **DIIN** (Gong et al., 2018), **BERT**$_{NLI}$ (Devlin et al., 2018), and **BiMPM** (Wang et al., 2017). The first three methods were trained on the XNLI dataset (Conneau et al., 2018). The last method was trained on the LCQMC dataset (Liu et al., 2018). For each option, a textual entailment method retrieves six top-ranked sentences from a corpus to form the entailing text. Retrieval follows the procedure described in the above-mentioned **IR** method. The scenario text and diagram annotations may or may not be included in the entailing text, depending on the configuration. A combination of the question and the option form the entailed text. Finally, we choose the option with the highest entailment score as the answer.

We tested one reading comprehension method: **BERT**$_{RC}$ (Devlin et al., 2018). It was trained on the DuReader dataset (He et al., 2018). For each option, a reading comprehension method retrieves six top-ranked sentences from a corpus as part of the passage for reading comprehension. Retrieval follows the procedure described in the above-mentioned **IR** method. The scenario text and diagram annotations may or may not be included in the passage, depending on the configuration. Finally, the reading comprehension method extracts a text span from the passage. We choose the option that is the most similar to the extracted text span as the answer. Similarity is computed by the cosine similarity between two bags of words.

### 4.3 Results

The results are summarized in Table 1. Note that a question has four options. Even guessing randomly, the expected proportion of correctly answered questions would be 25%.

Almost all the methods performed similar to random guess, showing that SQA on our dataset has its unique challenges.

### 4.4 Discussion

To explain the poor performance of existing methods, we have identified the following challenges. First, SQA relies on domain knowledge that is not provided in the scenario. However, relevant knowledge may fail to be retrieved from the corpus. Second, for some questions, commonsense knowledge is needed but is not included in textbooks and may fail to be retrieved from Wikipedia. Third, the retrieved general knowledge needs to be applied to the specific case described by a scenario. Existing QA and reading comprehension methods hardly have this capability.

| | Textbooks | Wikipedia |
|---|---|---|
| IR | 25.24 | 25.14 |
| PMI | 26.22 | 25.19 |
| ESIM w/o scenario | 25.85 | 25.41 |
| ESIM w/ scenario | 24.34 | 24.41 |
| DIIN w/o scenario | 24.15 | 25.20 |
| DIIN w/ scenario | 25.11 | 24.89 |
| BERT$_{NLI}$ w/o scenario | 24.29 | 24.17 |
| BERT$_{NLI}$ w/ scenario | 24.97 | 24.68 |
| BiMPM w/o scenario | 24.13 | 24.51 |
| BiMPM w/ scenario | 24.76 | 23.81 |
| BERT$_{RC}$ w/o scenario | 24.81 | 24.78 |
| BERT$_{RC}$ w/ scenario | 23.66 | 23.01 |

Table 1: Proportions of correctly answered questions.

## 5 Conclusion

We have contributed GeoSQA—a large SQA dataset where diagrams are present and have been manually annotated with natural language descriptions. We have tested a variety of existing methods on our dataset. The results are not satisfactory, thus demonstrating the unique challenges presented by the SQA task on our dataset. In future work, we will work towards more effective solutions to meet the challenges.

Researchers are invited to use GeoSQA to support their own tasks, including but not limited to natural language based SQA, visual SQA, and the diagram-to-text task.

## Acknowledgments

This work was supported in part by the National Key R&D Program of China under Grant 2018YFB1005100, and in part by the NSFC under Grant 61572247. Gong Cheng was supported by the Qing Lan Program of Jiangsu Province. We would like to thank all the annotators.

## References

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013*

*Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1533–1544.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1657–1668.

Peter Clark, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter D. Turney, and Daniel Khashabi. 2016. Combining retrieval, statistics, and inference to answer elementary science questions. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 2580–2586.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel R. Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. XNLI: evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2475–2485.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Jiwei Ding, Yuan Wang, Wei Hu, Linfeng Shi, and Yuzhong Qu. 2018. Answering multiple-choice questions in geographical gaokao with a concept graph. In *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*, pages 161–176.

Yichen Gong, Heng Luo, and Jian Zhang. 2018. Natural language inference over interaction space. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Wei He, Kai Liu, Jing Liu, Yajuan Lyu, Shiqi Zhao, Xinyan Xiao, Yuan Liu, Yizhong Wang, Hua Wu, Qiaoqiao She, Xuan Liu, Tian Wu, and Haifeng Wang. 2018. Dureader: a chinese machine reading comprehension dataset from real-world applications. In *Proceedings of the Workshop on Machine Reading for Question Answering@ACL 2018, Melbourne, Australia, July 19, 2018*, pages 37–46.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard H. Hovy. 2017. RACE: large-scale reading comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 785–794.

Adam Lally, Sugato Bagchi, Michael Barborak, David W. Buchanan, Jennifer Chu-Carroll, David A. Ferrucci, Michael R. Glass, Aditya Kalyanpur, Erik T. Mueller, J. William Murdock, Siddharth Patwardhan, and John M. Prager. 2017. WatsonPaths: Scenario-based question answering and inference over unstructured information. *AI Magazine*, 38(2):59–76.

Xin Liu, Qingcai Chen, Chong Deng, Huajun Zeng, Jing Chen, Dongfang Li, and Buzhou Tang. 2018. LCQMC: A large-scale chinese question matching corpus. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 1952–1962.

Bingfeng Luo, Yansong Feng, Jianbo Xu, Xiang Zhang, and Dongyan Zhao. 2017. Learning to predict charges for criminal cases with legal basis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2727–2736.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. In *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, December 9, 2016.*

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016. Text matching as image recognition. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 2793–2799.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2383–2392.

Kirk Roberts, Dina Demner-Fushman, Ellen M. Voorhees, William R. Hersh, Steven Bedrick, and Alexander J. Lazar. 2018. Overview of the trec 2018 precision medicine track. In *Proceedings of The Twenty-Seventh Text REtrieval Conference, TREC 2018, Gaithersburg, Maryland, USA, November 14-16, 2018.*

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 4144–4150.

Chaojun Xiao, Haoxi Zhong, Zhipeng Guo, Cunchao Tu, Zhiyuan Liu, Maosong Sun, Yansong Feng, Xianpei Han, Zhen Hu, Heng Wang, and Jianfeng Xu. 2018. CAIL2018: A large-scale legal dataset for judgment prediction. *CoRR*, abs/1807.02478.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 2013–2018.

Hai Ye, Xin Jiang, Zhunchen Luo, and Wenhan Chao. 2018. Interpretable charge predictions for criminal cases: Learning to generate court views from fact descriptions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1854–1864.

Zhiwei Zhang, Lingling Zhang, Hao Zhang, Weizhuo He, Zequn Sun, Gong Cheng, Qizhi Liu, Xinyu Dai, and Yuzhong Qu. 2018. Towards answering geography questions in gaokao: A hybrid approach. In *Proceedings of the Third China Conference on Knowledge Graph and Semantic Computing, CCKS 2018, Tianjin, China, 14-17 August 2018*, pages 1–13.

Haoxi Zhong, Zhipeng Guo, Cunchao Tu, Chaojun Xiao, Zhiyuan Liu, and Maosong Sun. 2018. Legal judgment prediction via topological learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3540–3549.

# Revisiting the Evaluation of Theory of Mind through Question Answering

**Matthew Le** **Y-Lan Boureau** **Maximilian Nickel**
Facebook AI Research
New York, NY
{mattle,ylan,maxn}@fb.com

## Abstract

Theory of mind, i.e., the ability to reason about intents and beliefs of agents is an important task in artificial intelligence and central to resolving ambiguous references in natural language dialogue. In this work, we revisit the evaluation of theory of mind through question answering. We show that current evaluation methods are flawed and that existing benchmark tasks can be solved without theory of mind due to dataset biases. Based on prior work, we propose an improved evaluation protocol and dataset in which we explicitly control for data regularities via a careful examination of the answer space. We show that state-of-the-art methods which are successful on existing benchmarks fail to solve theory-of-mind tasks in our proposed approach.

## 1 Introduction

Humans interact and communicate with other people in a highly efficient way, as described for instance as Grice's cooperative principle (Grice et al., 1975). Inferring other people's mental state is a crucial component of cognitive development, playing a role in how humans learn the meaning of words (Bloom, 2002), distinguish beliefs from reality (Gopnik and Astington, 1988), predict people's behavior (Wimmer and Perner, 1983), understand what others refer to, and reduce ambiguity in conversation (Clark, 1981; Keysar et al., 2000). This ability to reason about the mental state of other agents, called *theory of mind*, is thus an important component of an intelligent system aiming to emulate and interact with humans.

In developmental psychology, classic tests such as the *Sally-Anne test* (Baron-Cohen et al., 1985) have been used to assess the ability to infer false beliefs in others. Recently, Grant et al. (2017) and Nematzadeh et al. (2018) proposed to adapt these tests to evaluate the capability of machine learning models to form a theory of mind. The key insight of Grant et al. (2017) was to cast them as question answering tasks, where a system is given a story and has to answer questions about the beliefs of agents in that story. This allows to adapt the bAbi benchmarking protocol (Weston et al., 2016) to evaluate theory of mind capabilities of modern neural network architectures: stories are automatically generated so that a suitably large number of examples can be provided for training.

We believe this is a promising approach for advancing research on theory of mind, as it decouples its evaluation from problems such as multi-agent systems, game theory, and meta-learning, which are all components of other evaluation methods (Rabinowitz et al., 2018; Bard et al., 2019). However, artificial data runs the risk of displaying hidden artifacts and biases that correlate with the prediction task and can be exploited by models (Jabri et al., 2016; Gururangan et al., 2018; Poliak et al., 2018). The highly systematic nature of data generation puts synthetic benchmarks at an even higher risk of overestimating the target competence of interest.

This paper shows that current theory-of-mind QA benchmarks do indeed suffer from data biases, and are perfectly solvable without theory of mind. To overcome this, we propose an improved evaluation method and dataset[1]. We then show that state-of-the-art memory-augmented models – which are successful on existing benchmarks – fail to solve theory-of-mind tasks in our improved approach.

## 2 Theory of Mind Benchmarks

This section briefly reviews the Sally-Anne test and related paradigms for evaluating theory of mind.

**First-Order Beliefs** The so-called *Sally-Anne test* (Baron-Cohen et al., 1985) examines children's

[1]Our code and dataset will be made available at https://github.com/facebookresearch/ToMi



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5872–5877, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

ability to reason about other agents' false beliefs. The child observes the actions of two agents (Sally, Anne). First, Sally puts an object into a container. Second, Anne moves the object without Sally observing this action. Third, the child is asked multiple questions about reality and the agents' beliefs:

*First-Order Belief*: Where will Sally look for the marble?

*Reality*: Where is the marble really?

*Memory*: Where was the marble in the beginning?

The first question tests the ability of the child to infer the correct mental state of Sally, i.e., that she has the false belief of the marble being in the basket. The reality and memory questions ensure that the child has a correct understanding of the state of the world and is not responding at chance.

**Second-Order Beliefs** "Sally-Anne"-type questions are well-suited to evaluate *first-order* beliefs. Nematzadeh et al. (2018) proposed to also evaluate *second-order* theory of mind, i.e., the ability to infer beliefs about beliefs. Perner and Wimmer (1985) proposed a set of experiments to test such second-order beliefs in children. The experiments can be summarized as follows: Two agents (Mary, John) see an icecream van in the park. The vendor tells them that he will be in the park all afternoon. After Mary leaves the park, the vendor decides to leave the park and tells John he is going to the church. On the way to the church, the vendor meets Mary and tells her also that he will be at the church. The child is then asked the following question:

*Second-Order Belief*: Where does John think Mary will go to get ice-cream?

As control, children are also asked memory, reality, and first-order belief questions.

**Psychology Tests as AI Benchmarks** Grant et al. (2017) cast the aforementioned experiments as question answering tasks and proposed to create a bAbi-style dataset (Weston et al., 2016) to evaluate theory of mind in artificial intelligence models. For instance, this is the bAbi question-answering task version of the Sally-Anne test:

*Sally puts a marble in her basket*
*Sally leaves the room*
*Anne moves the marble in her box*

---

**Q**: *Where would Sally look for the marble?*
**A**: Basket

Nematzadeh et al. (2018) evaluate several modern neural network architectures over the resulting dataset and report that all fail on theory-of-mind tasks, especially when irrelevant sentences are introduced into stories at test time. In the following, we refer to these benchmarks as *ToM-bAbi*.

## 3 Evaluating Theory of Mind Evaluation

This section examines shortcomings of ToM-bAbi for evaluating a model's theory of mind abilities and proposes an improved approach.

**Related Work on Dataset Artifacts** It is challenging to determine what specific competence is revealed by the successful completion of a task, as illustrated by the case of *Clever Hans*, the famous horse whose skill at reading human reactions passed for arithmetic ability (Pfungst, 1911). Recent analyses of several AI benchmarks have uncovered similar difficulties when probing learning models. In the domain of visual question answering, baselines solely relying on candidate answers have shown surprisingly good performance (Jabri et al., 2016), leading to the creation of a carefully designed diagnostic dataset (Johnson et al., 2017). Similarly, natural language inference benchmarks have been shown to be vulnerable to bias exploitation (Gururangan et al., 2018; Poliak et al., 2018), and multimodal machine translation benchmarks have been demonstrated to be too simple to require multi-modality (Caglayan et al., 2019). We follow the same approach here of examining the performance of a baseline that does not make use of a type of information crucial to the target competence. Taking "reasoning about another agent" as a working definition of theory of mind, a reasonable prerequisite of benchmarks probing theory of mind competence is that they be impossible to solve without some input about the other agent.

**Leveraging Dataset Biases in ToM-bAbi** Reviewing the generation process of ToM-bAbi uncovers predictable regularities that allow a model to use corner-cutting heuristics instead of keeping track of agents: the stories follow a strict event sequence template for each task type, shown in Fig. 1. The ToM-bAbi dataset takes precautions to guard against the most simplistic heuristic (e.g., 'always output the location of line 4') by adding irrelevant sentences at random places as noise, but many regularities and correlations remain. These regularities make it possible to construct a parsimonious set

**Algorithm 1:** Rules to solve ToM-bAbi

```
Data: question q, story s
Result: location l
if "beginning" ∈ q then        // Memory
    l ← location of first object occurrence;
else if "really" ∈ q then      // Reality
    l ← location of last object occurrence;
else if "look" ∈ q then    // 1st Order
    if "exited" ∈ s after last "object is in"
     then
        l ← location of last object
          occurrence before last "exited";
    else
        l ← location of last object
          occurrence;
    end
else if "think" ∈ q then   // 2nd Order
    if "exited" ∈ s after last "object is in"
     then
        l ← location of last object
          occurrence before first "exited";
    else
        l ← location of last object
          occurrence;
    end
end
```

of rules that perfectly solve all tasks without ever extracting any information about the agents. As shown in the pseudo-code implementation of Algorithm 1, these rules only involve simple lexical and ordinal patterns (code provided in the supplemental material).

**Towards Robust ToM Evaluation in QA** To increase robustness against such regularities and correlations, we build upon the ideas of ToM-bAbi but improve data generation and evaluation in multiple ways. We refer to the supplementary material for the full pseudo-code of our dataset generator.

First, to generate a balanced dataset over story types, ToM-bAbi uses different generators for true-belief, false-belief, and second-order false-belief stories. However, the different generators add clear biases to the data which can be exploited to identify the story type (for example, lack of the word "exited" signals a true belief story – see Fig. 1). To overcome this issue, we use the same randomized generation method for all stories and keep track of which type is produced. We then sample from this randomized story generator with rejection to create a balanced dataset over all three types of stories.

Second, to decrease the amount of information that can be predicted from any given event, we add the following random distractors during data generation: actions of unrelated agents to decorrelate actions from answers, distractor statements about locations and objects to make the number of mentions less informative, randomization of the order of exit/move/re-entry actions, and randomization of the agent whose beliefs are being queried. This leads to stories that are a lot less predictable (see examples in Fig. 1).

Third, theory of mind manifests in "Sally-Anne" type tests through the understanding that an agent's belief is different from the actual state of the world, and that both coexist at the same moment. It is therefore crucial to evaluate both, the ability to infer the state of the world *and* the mental state of an agent. Although current benchmarks include reality and memory control questions, only one type of question is asked for each different, separately generated story. This can obscure revealing correlations in the models' responses e.g., that accuracy in first-order belief questions is associated with lower performance on reality questions. Since a correct answer to a false-belief question is only meaningful if the reality question is also answered correctly, it is especially important to check that the models can distinguish between states *for each story*. For this reason, we propose to systematically ask all question types for each generated story. In particular, for a single story involving agents `A`, `B`, and object `O`, we ask *all* following questions:

*Reality*: Where is `O`?

*Memory*: Where was `O` in the beginning?

*First-Order Belief* `A`: Where will `A` look for `O`?

*First-Order Belief* `B`: Where will `B` look for `O`?

*Second-Order Belief* `A`: Where does `A` believe `B` will look for `O`?

*Second-Order Belief* `B`: Where does `B` believe `A` will look for `O`?

Furthermore, we propose to count a story $s$ as answered correctly if all questions about $s$ are jointly correct, and to measure overall accuracy as the fraction of correctly answered stories. This ensures that a model is not acquiring theory of mind at the expense of its ability to perform other tasks and that it correctly answers reality and false-beliefs

**ToM-bAbi dataset**

*Three types of stories*

1 $\langle A \rangle$ entered $\langle L \rangle$
2 $\langle B \rangle$ entered $\langle L \rangle$
3 Phone rang. *// Distractor can appear anywhere*
4 The $\langle O \rangle$ is in $\langle C1 \rangle$.
5 $\langle B \rangle$ exited $\langle L \rangle$ *// if story type 1 or 2*
6 $\langle A \rangle$ moved the $\langle O \rangle$ to $\langle C2 \rangle$.
7 $\langle A \rangle$ exited the $\langle L \rangle$ *// if story type 2*
8 $\langle B \rangle$ entered the $\langle L \rangle$ *// if story type 2*

*Example story*

1 Isla entered the bathroom.
2 Benjamin entered the bathroom.
3 The cabbage is in the green_pantry.
4 Phone rang.
5 Isla moved the cabbage to the red_drawer.

*Answers for each story-question pair*

| | Story 1 | Story 2 | Story 3 |
|---|---|---|---|
| First Order | C1 | C2 | C2 |
| Second Order | C1 | C1 | C2 |
| Memory | C1 | C1 | C1 |
| Reality | C2 | C2 | C2 |

**ToMi dataset**

*Examples of stories from the ToMi dataset*

1 Oliver dislikes the kitchen
2 Carter entered the porch.
3 Abigail entered the porch.
4 The potato is in the green_suitcase.
5 Abigail exited the porch.
6 Abigail entered the hall.
7 Carter moved the potato to the green_envelope.
8 Oliver entered the hall.

1 Mila entered the closet.
2 Isla entered the closet.
3 Ava entered the closet.
4 The orange is in the blue_container.
5 Isla exited the closet.
6 Isla entered the garage.
7 Ava moved the orange to the green_bathtub.

1 William entered the staircase.
2 Aiden entered the staircase.
3 Aiden exited the staircase.
4 Aria entered the staircase.
5 The potato is in the red_drawer.
6 Aiden dislikes the grapefruit
7 William moved the potato to the blue_container.
8 Aria exited the staircase.

Figure 1: *Left:*Stories from the ToM-bAbi dataset follow three strict templates, with the possible random insertion of the distractor phrase "Phone rang." This makes it easier to devise rules to locate the answers for all pairs of question and story types, as shown in Alg. 1. For example, C1 always appears in the same sentence as the first object occurrence. *Right:* the ToMi dataset we propose is generated with considerably more randomness, with distractor phrases, distractor locations, distractor characters, and shuffling of the order of actions.

questions jointly. Formally, we define this *joint accuracy* as

$$Acc_J(\mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \prod_{q \in \mathcal{Q}_s} \mathbb{1}(\widehat{a}_q = a_q) \quad (1)$$

where $\mathbb{1} : \{\bot, \top\} \rightarrow \{0, 1\}$ is the indicator function, $\mathcal{S}$ the set of all stories, $\mathcal{Q}_s$ the set of all questions about story $s$, and $a_q, \widehat{a}_q$ are the correct and the model's answer for question $q$ respectively.

## 4 Experimental Evaluation

This section evaluates state-of-the-art question answering methods on the original ToM-bAbI and on our improved dataset. We follow Nematzadeh et al. (2018) and consider Memory Networks (MemNN; Sukhbaatar et al. 2015), Relation Networks (RelNet; Santoro et al. 2017, and Recurrent Entity Networks (EntNet; Henaff et al. 2017), which solve the original bAbI question answering tasks.

Table 1 shows the average accuracy over all questions for the original ToM-bAbI benchmarks (ToM-easy, ToM) and our improved dataset (ToMi).

Table 1: Accuracy on benchmark datasets. "Average" indicates the average accuracy over all questions. "Joint" indicates the joint accuracy as defined in Eq. (1).

| | **Average** | | | **Joint** |
|---|---|---|---|---|
| | ToM-easy | ToM | ToMi | ToMi |
| Rules | 100.0 | 100.0 | 77.5 | 36.5 |
| MemNN | 100.0 | 90.7 | 77.2 | 44.3 |
| EntNet | 100.0 | 94.9 | 90.6 | 66.8 |
| RelNet | 100.0 | 94.4 | 86.0 | 57.4 |

Moreover, for ToMi we also show the joint accuracy as defined in Equation (1). Algorithm 1 achieves perfect accuracy on ToM-easy and ToM. Since all QA models have the capacity to capture these simple heuristics, it is not surprising that they also perform very well on the original benchmarks, i.e., perfect accuracy on ToM-easy and over 90% percent accuracy on ToM.[2] Moreover, we found that the drop in accuracy in ToM is mostly caused

[2]These results are better than in the ToM-bAbI paper. Communication with the authors has not provided clear reasons why, so this may be due to more extensive optimization.

Table 2: Average accuracy by question type (MemNN)

| | ToM-easy | ToM | ToMi |
|---|---|---|---|
| Memory | 100.0 | 71.9 | 98.90 |
| Reality | 100.0 | 100.0 | 93.39 |
| First-Order | 100.0 | 98.6 | 70.72 |
| Second-Order | 100.0 | 95.3 | 64.66 |

Table 3: Average accuracy per question type in ToMi. "FB" and "'w/o FB" indicate question-story pairs that do and do not involve false beliefs, respectively.

| | | MemNN | RelNet | EntNet |
|---|---|---|---|---|
| **w/o FB** | First Order | 85.45 | 96.42 | 94.29 |
| | Second Order | 82.67 | 95.37 | 85.08 |
| | Reality | 93.39 | 100.0 | 100.0 |
| | Memory | 98.90 | 99.90 | 100.0 |
| **FB** | First Order | 12.62 | 10.40 | 54.95 |
| | Second Order | 17.27 | 17.81 | 36.55 |

by *memory* questions. This is because ToM stories are significantly longer and the location of the first occurrence of the queried object can exceed the memory capacity of the models. Table 2 shows an ablation for MemNNs that illustrates this effect.

By contrast, Algorithm 1[3] and QA models fail to solve the ToMi tasks. This is especially clear when looking at the joint accuracy of Table 1, which is not inflated by easy-to-answer memory and reality questions. Our ablation in Table 3 provides further insights into these results. It lists the average accuracy per question type and further splits the results into false-belief and non-false-belief question-story pairs. All models do reasonably well on question-story pairs that do not involve false beliefs, i.e., where the mental state of an agent should coincide with the state of the world. However, for question-story pairs with false beliefs, all models fail to provide correct answers consistently. Recurrent Entity Networks, which explicitly aim to keep track of the state of the world, are performing best on false beliefs tasks, indicating that this is a useful inductive bias for QA models on theory-of-mind tasks.

## 5 Discussion

Theory of mind is an important component of intelligent systems which interact with humans. In the context of natural language, theory of mind is not only of interest because of its evaluation through question answering but also because it is a crucial component to understand references and reduce ambiguities. This work re-examined the evaluation of theory of mind through bAbi-style question answering tasks. We revealed exploitable regularities in the generated data of existing benchmarks, and proposed to remedy this with a new dataset and evaluation method. We also showed that existing question answering methods that were capable of solving the previously proposed benchmarks are not able to solve the new tasks anymore. In future work, we aim at developing models to solve the newly proposed tasks.

Achieving this would demonstrate some level of ability to reason about first- and second-order false beliefs. But an important point to keep in mind when using our benchmark is that it still relies on synthetic data and tasks. We chose this approach since synthetic data generation can be especially useful in novel and early-stage research efforts, as it provides a controlled environment and allows for detailed analyses of a model's ability to solve a task. However, while methods such as Recurrent Entity Networks have shown promise for keeping track of the state-of-the-world in our experiments, this is still in scenarios where the complexity of natural language is relatively simple. On real-world data, this ability would be much weaker as it would require additional competencies such as co-reference resolution, handling polysemy and ambiguities, and common-sense reasoning. Therefore, solving the tasks we propose can only be considered a prerequisite for a fully functional theory of mind which will ultimately have to be evaluated in real-world scenarios.

[3] A more elaborate set of rules could solve the tasks, but would require to take the agents into account.

## References


Nolan Bard, Jakob N Foerster, Sarath Chandar, Neil Burch, Marc Lanctot, H Francis Song, Emilio Parisotto, Vincent Dumoulin, Subhodeep Moitra, Edward Hughes, et al. 2019. The hanabi challenge: A new frontier for ai research. *arXiv preprint arXiv:1902.00506*.

Simon Baron-Cohen, Alan M. Leslie, and Uta Frith. 1985. Does the autistic child have a "theory of mind"? *Cognition*, 21(1):37–46.

Paul Bloom. 2002. *How children learn the meanings of words*. MIT press.

Ozan Caglayan, Pranava Swaroop Madhyastha, Lucia Specia, and Loïc Barrault. 2019. Probing the need

for visual context in multimodal machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4159–4170.

Herbert H Clark. 1981. Definite reference and mutual knowledge. *Elements of Discourse Understanding*.

Alison Gopnik and Janet W. Astington. 1988. Children's understanding of representational change and its relation to the understanding of false belief and the appearance-reality distinction. *Child Development*, 59(1):26.

Erin Grant, Aida Nematzadeh, and Thomas L. Griffiths. 2017. How can memory-augmented neural networks pass a false-belief task? In *Proceedings of the 39th Annual Meeting of the Cognitive Science Society, CogSci 2017, London, UK, 16-29 July 2017*.

H Paul Grice, Peter Cole, Jerry L Morgan, et al. 1975. Logic and conversation. *1975*, pages 41–58.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112.

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. 2017. Tracking the world state with recurrent entity networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Allan Jabri, Armand Joulin, and Laurens Van Der Maaten. 2016. Revisiting visual question answering baselines. In *European conference on computer vision*, pages 727–739. Springer.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2901–2910.

Boaz Keysar, Dale J Barr, Jennifer A Balin, and Jason S Brauner. 2000. Taking perspective in conversation: The role of mutual knowledge in comprehension. *Psychological Science*, 11(1):32–38.

Aida Nematzadeh, Kaylee Burns, Erin Grant, Alison Gopnik, and Thomas L. Griffiths. 2018. Evaluating theory of mind in question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2392–2400.

Josef Perner and Heinz Wimmer. 1985. "john thinks that mary thinks that..." attribution of second-order beliefs by 5- to 10-year-old children. *Journal of Experimental Child Psychology*, 39(3):437–471.

Oskar Pfungst. 1911. *Clever Hans:(the horse of Mr. Von Osten.) a contribution to experimental animal and human psychology*. Holt, Rinehart and Winston.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis only baselines in natural language inference. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 180–191.

Neil Rabinowitz, Frank Perbet, Francis Song, Chiyuan Zhang, SM Ali Eslami, and Matthew Botvinick. 2018. Machine theory of mind. In *International Conference on Machine Learning*, pages 4215–4224.

Adam Santoro, David Raposo, David G. T. Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Tim Lillicrap. 2017. A simple neural network module for relational reasoning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 4974–4983.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2440–2448.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2016. Towards ai-complete question answering: A set of prerequisite toy tasks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Heinz Wimmer and Josef Perner. 1983. Beliefs about beliefs: Representation and constraining function of wrong beliefs in young children's understanding of deception. *Cognition*, 13(1):103–128.

# Multi-passage BERT: A Globally Normalized BERT Model for Open-domain Question Answering

**Zhiguo Wang, Patrick Ng, Xiaofei Ma, Ramesh Nallapati, Bing Xiang**
AWS AI Labs
{zhiguow, patricng, xiaofeim, rnallapa, bxiang}@amazon.com

## Abstract

BERT model has been successfully applied to open-domain QA tasks. However, previous work trains BERT by viewing passages corresponding to the same question as independent training instances, which may cause incomparable scores for answers from different passages. To tackle this issue, we propose a multi-passage BERT model to globally normalize answer scores across all passages of the same question, and this change enables our QA model find better answers by utilizing more passages. In addition, we find that splitting articles into passages with the length of 100 words by sliding window improves performance by 4%. By leveraging a passage ranker to select high-quality passages, multi-passage BERT gains additional 2%. Experiments on four standard benchmarks showed that our multi-passage BERT outperforms all state-of-the-art models on all benchmarks. In particular, on the OpenSQuAD dataset, our model gains 21.4% EM and 21.5% $F_1$ over all non-BERT models, and 5.8% EM and 6.5% $F_1$ over BERT-based models.

## 1 Introduction

BERT model (Devlin et al., 2018) has achieved significant improvements on a variety of NLP tasks. For question answering (QA), it has dominated the leaderboards of several machine reading comprehension (RC) datasets. However, the RC task is only a simplified version of the QA task, where a model only needs to find an answer from a given passage/paragraph. Whereas, in reality, an open-domain QA system is required to pinpoint answers from a massive article collection, such as Wikipedia or the entire web.

Recent studies directly applied the BERT-RC model to open-domain QA (Yang et al., 2019; Nogueira et al., 2018; Alberti et al., 2019). They firstly leverage a passage retriever to retrieve multiple passages for each question. During training, passages corresponding to the same question are taken as independent training instances. During inference, the BERT-RC model is applied to each passage individually to predict an answer span, and then the highest scoring span is selected as the final answer. Although this method achieves significant improvements on several datasets, there are still several unaddressed issues. **First**, viewing passages of the same question as independent training instances may result in incomparable answer scores across passages. Thus, globally normalizing scores over all passages of the same question (Clark and Gardner, 2018) may be helpful. **Second**, previous work defines passages as articles, paragraphs, or sentences. However, *the question of proper granularity of passages* is still underexplored. **Third**, passage ranker for selecting high-quality passages has been shown to be very useful in previous open-domain QA systems (Wang et al., 2018a; Lin et al., 2018; Pang et al., 2019). However, we do not know whether it is still required for BERT. **Fourth**, most effective QA and RC models highly rely on explicit inter-sentence matching between questions and passages (Wang and Jiang, 2017; Wang et al., 2016; Seo et al., 2017; Wang et al., 2017), whereas BERT only applies self-attention layers over the concatenation of a question-passage pair. It is unclear whether the inter-sentence matching still matters for BERT.

To answer these questions, we conduct a series of empirical studies on the OpenSQuAD dataset (Rajpurkar et al., 2016; Wang et al., 2018a). Experimental results show that: (1) global normalization makes QA model more stable while pinpointing answers from large number of passages; (2) splitting articles into passages with the length of 100 words by sliding window brings 4% improvements; (3) leveraging a BERT-based passage



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5878–5882,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

ranker gives us extra 2% improvements; and (4) explicit inter-sentence matching is not helpful for BERT. We also compared our model with state-of-the-art models on four standard benchmarks, and our model outperforms all state-of-the-art models on all benchmarks.

## 2 Model

Open-domain QA systems aim to find an answer for a given question from a massive article collection. Usually, a retriever is leveraged to retrieve $m$ passages $P = [P_1, ..., P_i, ..., P_m]$ for a given question $Q = (q^1, ..., q^{|Q|})$, where $P_i = (p_i^1, ..., p_i^{|p_i|})$ is the $i$-th passage, and $q^k \in Q$ and $p_i^j \in P_i$ are corresponding words. A QA model will compute a score $Pr(a|Q, P)$ for each possible answer span $a$. We further decompose the answer span prediction into predicting the start and end positions of the answer span $Pr(a|Q, P) = P_s(a_s|Q, P)P_e(a_e|Q, P)$, where $P_s(a_s|Q, P)$ and $P_e(a_e|Q, P)$ are the probabilities of $a_s$ and $a_e$ to be the start and end positions.

**BERT-RC** model assumes passages in $P$ are independent of each other. The model concatenates the question $Q$ and each passage $P_i$ into a new sequence "[CLS] $p_i^1, ..., p_i^{|p_i|}$ [SEP] $q^1, ..., q^{|Q|}$ [SEP]", and applies BERT to encode this sequence. Then the vector representation of each word position from BERT encoder is fed into two separate dense layers to predict the probabilities $P_s$ and $P_e$ (Devlin et al., 2018). During training, the log-likelihood of the correct start and end positions for each passage is optimized independently. For passages without any correct answers, we set the start and end positions to be 0, which is the position for the first token [CLS]. During inference, BERT-RC model is applied to each passage individually to predict an answer, and then the highest scoring span is selected as the final answer. If answers from different passages have the same string, they are merged by summing up their scores.

**Multi-passage BERT**: BERT-RC model normalizes probability distributions $P_s$ and $P_e$ for each passage independently, which may cause incomparable answer scores across passages. To tackle this issue, we leverage the global normalization method (Clark and Gardner, 2018) to normalize answer scores among multiple passages, and dub this model as multi-passage BERT. Concretely, all passages of the same question are processed independently as we do in BERT-RC until the normalization step. Then, $softmax$ is applied to normalize all word positions from all passages.

**Passage ranker** reranks all retrieved passages, and selects a list of high-quality passages for the multi-passage BERT model. We implement the passage ranker as another BERT model, which is similar to multi-passage BERT except that at the output layer it only predicts a single score for each passage based on the vector representation of the first token [CLS]. We also apply $softmax$ over all passage scores corresponding to the same question, and train to maximize the log-likelihood of passages containing the correct answers. Denote the passage score as $Pr(P_i|Q, P)$, then the score of an answer span from passage $P_i$ will be $Pr(P_i|Q, P)P_s(a_s|Q, P)P_e(a_e|Q, P)$.

## 3 Experiments

**Datasets**: We experiment on four open-domain QA datasets. (1) *OpenSQuAD*: question-answer pairs are from SQuAD 1.1 (Rajpurkar et al., 2016), but a QA model will find answers from the entire Wikipedia rather than the given context. Following Chen et al. (2017), we use the 2016-12-21 English Wikipedia dump. 5,000 QA pairs are randomly selected from the original training set as our validation set, and the remaining QA pairs are taken as our new training set. The original development set is used as our test set. (2) *TriviaQA*: TriviaQA unfiltered version (Joshi et al., 2017) are used. Following Pang et al. (2019), we randomly hold out 5,000 QA pairs from the original training set as our validation set, and take the remaining pairs as our new training set. The original development set is used as our test set. (3) *Quasar-T* (Dhingra et al., 2017) and (4) *SearchQA* (Dunn et al., 2017) are leveraged with the official split.

**Basic Settings**: If not specified, the pre-trained BERT-base model with default hyper-parameters is leveraged. ElasticSearch with BM25 algorithm is employed as our retriever for OpenSQuAD. Passages for other datasets are from the corresponding releases. During training, we use top-10 passages for each question plus all passages (within the top-100 list) containing correct answers. During inference, we use top-30 passages for each question. Exact Match (EM) and $F_1$ scores (Rajpurkar et al., 2016) are utilized as the evaluation metrics.

| No. | Model | EM | $F_1$ |
|---|---|---|---|
| 1 | Single-sentence | 34.8 | 44.4 |
| 2 | Length-50 | 35.5 | 45.2 |
| 3 | Length-100 | 35.7 | 45.7 |
| 4 | Length-200 | 34.8 | 44.7 |
| 5 | w/o sliding-window (same as (3)) | 35.7 | 45.7 |
| 6 | w/ sliding-window | 40.4 | 49.8 |
| 7 | w/o passage ranker (same as (6)) | 40.4 | 49.8 |
| 8 | w/ passage ranker | 41.3 | 51.7 |
| 9 | w/ passage scores | **42.8** | **53.4** |
| 10 | BERT+QANet | 18.3 | 27.8 |
| 11 | BERT+QANet (fix BERT) | 35.5 | 45.9 |
| 12 | BERT+QANet (init. from (11)) | 36.2 | 46.4 |

Table 1: Results on the validation set of OpenSQuAD.

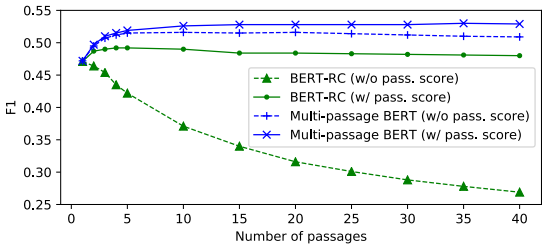


Figure 1: Effect of global normalization.

### 3.1 Model Analysis

To answer questions from section 1, we conduct a series of experiments on OpenSQuAD dataset, and report the validation set results in Table 1. Multi-passage BERT model is used for experiments.

**Effect of passage granularity**: Previous work usually defines passages as articles (Chen et al., 2017), paragraphs (Yang et al., 2019), or sentences (Wang et al., 2018a; Lin et al., 2018). We explore the effect of passage granularity regarding to the passage length, i.e., the number of words in each passage. Each article is split into non-overlapping passages based on a fixed length. We vary passage length among {50, 100, 200}, and list the results as models (2) (3) (4) in Table 1, respectively. Comparing to single-sentence passages (model (1)), leveraging fixed-length passages works better, and passages with 100 words works the best. Hereafter, we set passage length as 100 words.

**Effect of sliding window**: Splitting articles into non-overlapping passages may force some near-boundary answer spans to lose useful contexts. To deal with this issue, we split articles into overlapping passages by sliding window. We set the window size as 100 words, and the stride as 50 words (half the window size). Result from the sliding window model is shown as model (6) in Table 1. We can see that this method brings us 4.7% EM and 4.1% $F_1$ improvements. Hereafter, we use sliding window method.

**Effect of passage ranker**: We plug the passage ranker into the QA pipeline. First, the retriever returns top-100 passages for each question. Then, the passage ranker is employed to rerank these 100 passages. Finally, multi-passage BERT takes top-30 reranked passages as input to pinpoint the final answer. We design two models to check the effect of the passage ranker. The first model utilizes the reranked passages but without using passage scores, whereas the second model makes use of both the reranked passages and their scores. Results are given in Table 1 as models (8) and (9) respectively. We can find that only using reranked passages gives us 0.9% EM and 1.0% $F_1$ improvements, and leveraging passage scores gives us 1.5% EM and 1.7% $F_1$ improvements. Therefore, passage ranker is useful for multi-passage BERT model.

**Effect of global normalization**: We train BERT-RC and multi-passage BERT models using the reranked passages, then evaluate them by taking as input various number of passages. These models are evaluated on two setups: with and without using passage scores. $F_1$ scores for BERT-RC based on different number of passages are shown as the dotted and solid green curves in Figure 1. $F_1$ scores for our multi-passage BERT model with similar settings are shown as the dotted and solid blue curves. We can see that all models start from the same $F_1$, because multi-passage BERT is equivalent to BERT-RC when using only one passage. While increasing the number of passages, BERT-RC without using passage scores decreases the performance significantly, which verifies that the answer scores from BERT-RC are incomparable across passages. This issue is alleviated to some extent by leveraging passage scores. On the other hand, performance from multi-passage BERT without using passage scores increases at the beginning, and then flattens out after passage number is over 10. By utilizing passage scores, multi-passage BERT gets better performance while using more passages. This phenomenon shows the effectiveness of global normalization, which enables the model find better answers by utilizing more passages.

**Does explicit inter-sentence matching matter?** Almost all previous state-of-the-art QA and RC models find answers by matching pas-

| Datasets | Quasar-T | | SearchQA | | TriviaQA | | OpenSQuAD | |
|---|---|---|---|---|---|---|---|---|
| Models | EM | $F_1$ | EM | $F_1$ | EM | $F_1$ | EM | $F_1$ |
| DrQA (Chen et al., 2017) | 37.7 | 44.5 | 41.9 | 48.7 | 32.3 | 38.3 | 29.8 | - |
| $R^3$ (Wang et al., 2018a) | 35.3 | 41.7 | 49.0 | 55.3 | 47.3 | 53.7 | 29.1 | 37.5 |
| OpenQA (Lin et al., 2018) | 42.2 | 49.3 | 58.8 | 64.5 | 48.7 | 56.3 | 28.7 | 36.6 |
| TraCRNet (Dehghani et al., 2019) | 43.2 | 54.0 | 52.9 | 65.1 | - | - | - | - |
| HAS-QA (Pang et al., 2019) | 43.2 | 48.9 | 62.7 | 68.7 | 63.6 | 68.9 | - | - |
| BERT (Large) (Nogueira et al., 2018) | - | - | - | 69.1 | - | - | - | - |
| BERT serini (Yang et al., 2019) | - | - | - | - | - | - | 38.6 | 46.1 |
| BERT-RC (Ours) | 49.7 | 56.8 | 63.7 | 68.7 | 61.0 | 66.9 | 45.4 | 52.5 |
| Multi-Passage BERT (Base) | **51.3** | 59.0 | **65.2** | 70.6 | 62.0 | 67.5 | 51.2 | 59.0 |
| Multi-Passage BERT (Large) | 51.1 | **59.1** | 65.1 | **70.7** | **63.7** | **69.2** | **53.0** | **60.9** |

Table 2: Comparison with state-of-the-art models, where the first group are models without using BERT, the second group are BERT-based models, and the last group are our multi-passage BERT models.

sages with questions, aka inter-sentence matching (Wang and Jiang, 2017; Wang et al., 2016; Seo et al., 2017; Wang et al., 2017; Song et al., 2017). However, BERT model simply concatenates a passage with a question, and differentiates them by separating them with a delimiter token [SEP], and assigning different segment ids for them. Here, we aim to check whether explicit inter-sentence matching still matters for BERT. We employ a shared BERT model to encode a passage and a question individually, and a weighted sum of all BERT layers is used as the final token-level representation for the question or passage, where weights for all BERT layers are trainable parameters. Then the passage and question representations are input into QANet (Yu et al., 2018) to perform inter-sentence matching, and predict the final answer. Model (10) in Table 1 shows the result of jointly training the BERT encoder and the QANet model. The result is very poor, likely because the parameters in BERT are catastrophically forgotten while training the QANet model. To tackle this issue, we fix parameters in BERT, and only update parameters for QANet. The result is listed as model (11). It works better than model (10), but still worse than multi-passage BERT in model (6). We design another model by starting from model (11), and then jointly fine-tuning the BERT encoder and QANet. Model (12) in Table 1 shows the result. It works better than model (11), but still has a big gap with multi-passage BERT in model (6) . Therefore, we conclude that the explicit inter-sentence matching is not helpful for multi-passage BERT. One possible reason is that the multi-head self-attention layers in BERT has already embedded the inter-sentence matching.

### 3.2 Comparison with State-of-the-art Models

We evaluate BERT-RC and Multi-passage BERT on four standard benchmarks, where passage scores are leveraged for both models. We build another multi-passage BERT for each dataset by initializing it with the pre-trained BERT-Large model. Experimental results from our models as well as other state-of-the-art models are shown in Table 2, where the first group are open-domain QA models without using the BERT model, the second group are BERT-based models, and the last group are our multi-passage BERT models.

From Table 2, we can see that our multi-passage BERT model outperforms all state-of-the-art models across all benchmarks, and it works consistently better than our BERT-RC model which has the same settings except the global normalization. In particular, on the OpenSQuAD dataset, our model improves by 21.4% EM and 21.5% $F_1$ over all non-BERT models, and 5.8% EM and 6.5% $F_1$ over BERT-based models. Leveraging BERT-Large model makes multi-passage BERT even better on TriviaQA and OpenSQuAD datasets.

## 4 Conclusion

We propose a multi-passage BERT model for open-domain QA to globally normalize answer scores across mutiple passages corresponding to the same question. We find two effective techniques to improve the performance of multi-passage BERT: (1) splitting articles into passages with the length of 100 words by sliding window; and (2) leveraging a passage ranker to select high-quality passages. With all these techniques, our multi-passage BERT model outperforms all state-of-the-art models on four standard benchmarks.

In future, we plan to consider inter-correlation

among passages for open-domain question answering (Wang et al., 2018b; Song et al., 2018).

## References


Chris Alberti, Kenton Lee, and Michael Collins. 2019. A bert baseline for the natural questions. *arXiv preprint arXiv:1901.08634*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *ACL 2017*.

Christopher Clark and Matt Gardner. 2018. Simple and effective multi-paragraph reading comprehension. In *ACL 2018*.

Mostafa Dehghani, Hosein Azarbonyad, Jaap Kamps, and Maarten de Rijke. 2019. Learning to transform, combine, and reason in open-domain question answering. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 681–689. ACM.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Bhuwan Dhingra, Kathryn Mazaitis, and William W Cohen. 2017. Quasar: Datasets for question answering by search and reading. *arXiv preprint arXiv:1707.03904*.

Matthew Dunn, Levent Sagun, Mike Higgins, V Ugur Guney, Volkan Cirik, and Kyunghyun Cho. 2017. Searchqa: A new q&a dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL 2017*.

Yankai Lin, Haozhe Ji, Zhiyuan Liu, and Maosong Sun. 2018. Denoising distantly supervised open-domain question answering. In *ACL 2018*.

Rodrigo Nogueira, Jannis Bulian, and Massimiliano Ciaramita. 2018. Learning to coordinate multiple reinforcement learning agents for diverse query reformulation. *arXiv preprint arXiv:1809.10658*.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Lixin Su, and Xueqi Cheng. 2019. Has-qa: Hierarchical answer spans model for open-domain question answering. In *AAAI 2019*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *EMNLP 2016*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *ICLR 2017*.

Linfeng Song, Zhiguo Wang, and Wael Hamza. 2017. A unified query-based generative model for question generation and question answering. *arXiv preprint arXiv:1709.01058*.

Linfeng Song, Zhiguo Wang, Mo Yu, Yue Zhang, Radu Florian, and Daniel Gildea. 2018. Exploring graph-structured passage representation for multi-hop reading comprehension with graph neural networks. *arXiv preprint arXiv:1809.02040*.

Shuohang Wang and Jing Jiang. 2017. Machine comprehension using match-lstm and answer pointer. In *ICLR 2017*.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerry Tesauro, Bowen Zhou, and Jing Jiang. 2018a. R3: Reinforced ranker-reader for open-domain question answering. In *AAAI 2018*.

Shuohang Wang, Mo Yu, Jing Jiang, Wei Zhang, Xiaoxiao Guo, Shiyu Chang, Zhiguo Wang, Tim Klinger, Gerald Tesauro, and Murray Campbell. 2018b. Evidence aggregation for answer re-ranking in open-domain question answering. In *ICLR 2018*.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *ACL 2017*.

Zhiguo Wang, Haitao Mi, Wael Hamza, and Radu Florian. 2016. Multi-perspective context matching for machine comprehension. *arXiv preprint arXiv:1612.04211*.

Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019. End-to-end open-domain question answering with bertserini. *arXiv preprint arXiv:1902.01718*.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. In *ICLR 2018*.

# A Span-Extraction Dataset for Chinese Machine Reading Comprehension

**Yiming Cui**[†‡], **Ting Liu**[†], **Wanxiang Che**[†],
**Li Xiao**[‡], **Zhipeng Chen**[‡], **Wentao Ma**[‡§], **Shijin Wang**[‡§], **Guoping Hu**[‡]
[†]Research Center for Social Computing and Information Retrieval (SCIR),
Harbin Institute of Technology, Harbin, China
[‡]State Key Laboratory of Cognitive Intelligence, iFLYTEK Research, China
[§]iFLYTEK AI Research (Hebei), Langfang, China
[†]{ymcui,tliu,car}@ir.hit.edu.cn
[‡§]{ymcui,lixiao3,zpchen,wtma,sjwang3,gphu}@iflytek.com

## Abstract

Machine Reading Comprehension (MRC) has become enormously popular recently and has attracted a lot of attention. However, the existing reading comprehension datasets are mostly in English. In this paper, we introduce a Span-Extraction dataset for Chinese machine reading comprehension to add language diversities in this area. The dataset is composed by near 20,000 real questions annotated on Wikipedia paragraphs by human experts. We also annotated a challenge set which contains the questions that need comprehensive understanding and multi-sentence inference throughout the context. We present several baseline systems as well as anonymous submissions for demonstrating the difficulties in this dataset. With the release of the dataset, we hosted the Second Evaluation Workshop on Chinese Machine Reading Comprehension (CMRC 2018). We hope the release of the dataset could further accelerate the Chinese machine reading comprehension research.[1]

## 1 Introduction

To read and comprehend natural languages is the key to achieve advanced artificial intelligence. Machine Reading Comprehension (MRC) aims to comprehend the context of given articles and answer the questions based on them. Various types of machine reading comprehension datasets have been proposed, such as cloze-style reading comprehension (Hermann et al., 2015; Hill et al., 2015; Cui et al., 2016), span-extraction reading comprehension (Rajpurkar et al., 2016; Trischler et al., 2016), open-domain reading comprehension (Nguyen et al., 2016; He et al., 2017), reading comprehension with multiple-choice (Richardson et al., 2013; Lai et al., 2017), etc. Along with the development of the reading comprehension dataset, various neural network approaches have been proposed and made a big advancement in this area (Kadlec et al., 2016; Cui et al., 2017; Dhingra et al., 2017; Wang and Jiang, 2016; Xiong et al., 2016; Liu et al., 2017; Wang et al., 2017; Hu et al., 2018; Wang et al., 2018; Yu et al., 2018).

We also have seen various efforts on the construction of Chinese machine reading comprehension datasets. In cloze-style reading comprehension, Cui et al. (2016) proposed a Chinese cloze-style reading comprehension dataset: People's Daily & Children's Fairy Tale. To add difficulties to the dataset, along with the automatically generated evaluation sets (development and test), they also release a human-annotated evaluation set. Later, Cui et al. (2018) propose another dataset, which is gathered from children's reading material. To add more diversity and for further investigation on transfer learning, they also provide another evaluation dataset, which is also annotated by human experts, but the query is more natural than the cloze type. The dataset was used in the first evaluation workshop on Chinese machine reading comprehension (CMRC 2017). In open-domain reading comprehension, He et al. (2017) propose a large-scale open-domain Chinese machine reading comprehension dataset (DuReader), which contains 200k queries annotated from the user query logs on the search engine. Shao et al. (2018) proposed a reading comprehension dataset in Traditional Chinese.

Though we have seen that the current machine learning approaches have surpassed the human performance on the SQuAD dataset (Rajpurkar et al., 2016), we wonder if these state-of-the-art models could also give a similar performance on the dataset of different languages. To further accelerate the development of the machine reading comprehension research, we propose a span-

[1]Resources are available: https://github.com/ymcui/cmrc2018.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5883–5889,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| | |
|---|---|
| **[Passage]**<br>《黄色脸孔》是柯南·道尔所著的福尔摩斯探案的56个短篇故事之一，收录于《福尔摩斯回忆录》。孟罗先生素来与妻子恩爱，但自从最近邻居新入伙后，孟罗太太则变得很奇怪，曾经凌晨时份外出，又藉丈夫不在家时偷偷走到邻居家中。于是孟罗先生向福尔摩斯求助，福尔摩斯听毕孟罗先生的故事后，认为孟罗太太被来自美国的前夫勒索，所以不敢向孟罗先生说出真相，所以吩咐孟罗先生，如果太太再次走到邻居家时，即时联络他，他会第一时间赶到。孟罗太太又走到邻居家，福尔摩斯陪同孟罗先生冲入，却发现**邻居家中的人是孟罗太太与前夫生的女儿，因为孟罗太太的前夫是黑人，她怕孟罗先生嫌弃混血儿，所以不敢说出真相。** | **[Passage]**<br>"The Adventure of the Yellow Face", one of the 56 short Sherlock Holmes stories written by Sir Arthur Conan Doyle, is the third tale from The Memoirs of Sherlock Holmes. Mr. Munro has always been loved by his wife, but since the new neighbors recently joined, Mrs. Munro has become very strange. She used to go out in the early hours of the morning and secretly went to her neighbors when her husband was not at home. ... Mrs. Munro went to the neighbor's house again, and Holmes accompanied Mr. Munro to rush in, only to find that the neighbor's family was the daughter of Mrs. Munro and her ex-husband, **because Mrs. Munro's ex-husband was black, and she was afraid of Mr. Munro hate the mixed-race, so she did not dare to tell the truth.** |
| **[Question]**<br>孟罗太太为什么在邻居新入伙后变得很奇怪?<br>**[Answer 1]**<br>邻居家中的人是孟罗太太与前夫生的女儿，因为孟罗太太的前夫是黑人，她怕孟罗先生嫌弃混血儿<br>**[Answer 2]**<br>邻居家中的人是孟罗太太与前夫生的女儿，因为孟罗太太的前夫是黑人，她怕孟罗先生嫌弃混血儿，所以不敢说出真相。<br>**[Answer 3]**<br>邻居家中的人是孟罗太太与前夫生的女儿，因为孟罗太太的前夫是黑人，她怕孟罗先生嫌弃混血儿，所以不敢说出真相。 | **[Question]**<br>Why Mrs. Munro became strange after the new neighbors moved in?<br>**[Answer 1]**<br>because Mrs. Munro's ex-husband was black, and she was afraid of Mr. Munro hate the mixed-race<br>**[Answer 2]**<br>because Mrs. Munro's ex-husband was black, and she was afraid of Mr. Munro hate the mixed-race<br>**[Answer 3]**<br>because Mrs. Munro's ex-husband was black, and she was afraid of Mr. Munro hate the mixed-race, so she did not dare to tell the truth. |

Figure 1: An example of the proposed CMRC 2018 dataset (challenge set). English translation is also given for comparison.

extraction dataset for Chinese machine reading comprehension. Figure 1 shows an example of the proposed dataset. The main contributions of our work can be concluded as follows.

- We propose a Chinese span-extraction reading comprehension dataset which contains near 20,000 human-annotated questions, to add linguistic diversity in reading comprehension field.
- To thoroughly test the ability of the MRC systems, besides the development and test set, we also make a challenge set which contains carefully annotated questions that require various clues in the passage. The BERT-based approaches could only give under 50% F1-score on this set, indicating its difficulty.
- The proposed Chinese RC data could also be a resource for cross-lingual research purpose when studied along with SQuAD and other similar datasets.

## 2 The Proposed Dataset

### 2.1 Task Definition

Generally, the reading comprehension task can be described as a triple $\langle \mathcal{P}, \mathcal{Q}, \mathcal{A} \rangle$, where $\mathcal{P}$ represents **P**assage, $\mathcal{Q}$ represents **Q**uestion and the $\mathcal{A}$ represents **A**nswer. Specifically, for span-extraction reading comprehension task, the question is annotated by the human, which is much more natural than the cloze-style MRC datasets (Hill et al., 2015; Cui et al., 2016). The answer $\mathcal{A}$ should be a span which is directly extracted from the passage $\mathcal{P}$. According to most of the works on SQuAD, the task can be simplified by predicting the start and end pointer in the passage (Wang and Jiang, 2016).

### 2.2 Data Pre-Processing

We downloaded Chinese portion of Wikipedia webpage dump[2] on Jan 22, 2018 and used open-source toolkit *Wikipedia Extractor*[3] for pre-processing the raw files into plain text. We also convert the Traditional Chinese characters into Simplified Chinese for normalization purpose using *opencc*[4] toolkit.

### 2.3 Human Annotation

The questions in the proposed dataset are completely annotated by human experts, which is different from previous works that rely on the automatic data generation (Hermann et al., 2015; Hill et al., 2015; Cui et al., 2016). Before annotating, the document is divided into several passages, and each passage is limited to have no more than 500 Chinese words, where the word is counted by using LTP (Che et al., 2010). Then, the annotator was instructed to first evaluate the appropriateness of the passages, because some of the passages are extremely difficult for the public to understand. Following rules are applied when discarding the passages.

[2] https://dumps.wikimedia.org/zhwiki/latest/

[3] http://medialab.di.unipi.it/wiki/Wikipedia_Extractor

[4] https://github.com/BYVoid/OpenCC

- Contain over 30% non-Chinese characters.
- Contain many professional words that hard to understand.
- Contains many special characters and symbols.
- The paragraph is written in classical Chinese, which is substantially different from the Chinese language nowadays.

After identifying the passage is appropriate for annotation, the annotator will read the passage and ask the questions based on it and annotated a primary answer. During the question annotation, the following rules are applied.

- No more than five questions for each passage.
- The answer MUST be a span in the passage to meet the task definition.
- Encourage the question diversity, such as who/when/where/why/how, etc.
- Avoid directly using the description in the passage. Use paraphrase or syntax transformation to add difficulties for answering.
- Long answers (say over 30 characters) will be discarded.

For the evaluation sets, i.e., development, test, challenge, there are three answers available for better evaluation. Besides the primary answer that was annotated by the question proposer, we also invite two additional annotators to write the second and third answers for the question. During this phase, the annotators could not see the primary answer to ensure the answer was not copied from others and encourage the diversities in the answer.

### 2.4 Challenge Set

In order to examine how well can reading comprehension models deal with the questions that need comprehensive reasoning over various clues in the context, we additionally annotated a small challenge set for this purpose while keeping the span-extraction style. The annotation was also done by three annotators in a similar way that for development and test set. Figure 1 shows an example in the challenge set. The question should meet the following standards to be qualified into this set.

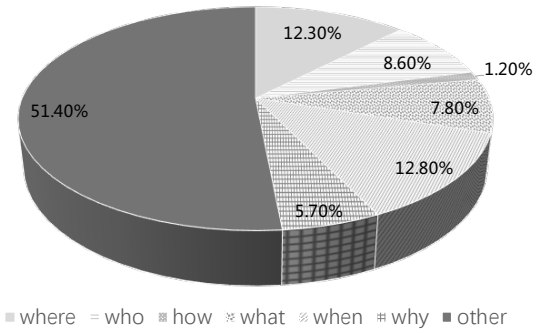


Figure 2: Question types of the development set.

- The answer cannot be only inferred by a single sentence in the passage if the answer is a single word or short phrase. We encourage the annotator to ask the questions that need comprehensive reasoning in the passage to increase the difficulties.
- If the answer belongs to a type of named entity, or specific genre (such as date, color, etc.), it can not be the only one in the context, or the machine could easily pick it out according to its type. For example, if there is only one person name appears in the context, then it cannot be used for annotating questions. There should be at least two person names that could mislead the machine for answering.

### 2.5 Statistics

The general statistics of the pre-processed data are given in Table 1. The question type distribution of the development set is given in Figure 2.

| | Train | Dev | Test | Challenge |
|---|---|---|---|---|
| Question # | 10,321 | 3,351 | 4,895 | 504 |
| Answer per Q | 1 | **3** | **3** | **3** |
| Max P tokens | 962 | 961 | 980 | 916 |
| Max Q tokens | 89 | 56 | 50 | 47 |
| Max A tokens | 100 | 85 | 92 | 77 |
| Avg P tokens | 452 | 469 | 472 | 464 |
| Avg Q tokens | 15 | 15 | 15 | 18 |
| Avg A tokens | 17 | 9 | 9 | 19 |

Table 1: Statistics of the CMRC 2018 dataset. (P: Passage, Q: Question, A: Answer)

## 3 Evaluation Metrics

In this paper, we adopt two evaluation metrics following Rajpurkar et al. (2016). However, as

| | Development | | Test | | Challenge | |
|---|---|---|---|---|---|---|
| | EM | F1 | EM | F1 | EM | F1 |
| *Estimated Human Performance* | *91.083* | *97.348* | *92.400* | *97.914* | *90.382* | *95.248* |
| Z-Reader (single model) | **79.776** | **92.696** | **74.178** | **88.145** | 13.889 | **37.422** |
| MCA-Reader (ensemble) | 66.698 | 85.538 | 71.175 | 88.090 | 15.476 | 37.104 |
| RCEN (ensemble) | 76.328 | 91.370 | 68.662 | 85.753 | 15.278 | 34.479 |
| MCA-Reader (single model) | 63.902 | 82.618 | 68.335 | 85.707 | 13.690 | 33.964 |
| OmegaOne (ensemble) | 66.977 | 84.955 | 66.272 | 82.788 | 12.103 | 30.859 |
| RCEN (single model) | 73.253 | 89.750 | 64.576 | 83.136 | 10.516 | 30.994 |
| GM-Reader (ensemble) | 58.931 | 80.069 | 64.045 | 83.046 | **15.675** | 37.315 |
| OmegaOne (single model) | 64.430 | 82.699 | 64.188 | 81.539 | 10.119 | 29.716 |
| GM-Reader (single model) | 56.322 | 77.412 | 60.470 | 80.035 | 13.690 | 33.990 |
| R-NET (single model) | 45.418 | 69.825 | 50.112 | 73.353 | 9.921 | 29.324 |
| SXU-Reader (ensemble) | 40.292 | 66.451 | 46.210 | 70.482 | N/A | N/A |
| SXU-Reader (single model) | 37.310 | 66.121 | 44.270 | 70.673 | 6.548 | 28.116 |
| T-Reader (single model) | 39.422 | 62.414 | 44.883 | 66.859 | 7.341 | 22.317 |
| **BERT-base (Chinese)** | 63.6 | 83.9 | 67.8 | 86.0 | 18.4 | 42.1 |
| **BERT-base (Multi-lingual)** | 64.1 | 84.4 | 68.6 | 86.8 | 18.6 | 43.8 |

Table 2: Baseline results and CMRC 2018 participants' results. Note that, some of the submissions are using development set for training as well.

the Chinese language is quite different from English, we adapt the original metrics in the following ways. Note that, the common punctuations, white spaces are ignored for normalization.

### 3.1 Exact Match

Measure the exact match between the prediction and ground truths that is 1 for the exact match. Otherwise, the score is 0. This is the same as the one proposed by Rajpurkar et al. (2016).

### 3.2 F1-Score

Measure the character-level fuzzy match between the prediction and ground truths. Instead of treating the predictions and ground truths as bag-of-words, we calculate the length of the longest common sequence (LCS) between them and compute the F1-score accordingly. We take the maximum F1 over all of the ground truth answers for a given question. Note that, non-Chinese words will not be segmented into characters.

### 3.3 Estimated Human Performance

We also report the estimated human performance in order to measure the difficulty of the proposed dataset. As we have illustrated in the previous section, there are three answers for each question in development, test, and challenge set. Unlike Rajpurkar et al. (2016), we use a cross-validation method to calculate the performance. We regard the first answer as human prediction and treat the rest of the answers as ground truths. In this way, we can get three human prediction performance by iteratively regarding the first, second, and third answer as the human prediction. Finally, we calculate the average of three results as the final estimated human performance on this dataset.

## 4 Experimental Results

### 4.1 Baseline System

Following Devlin et al. (2019), we adopt BERT for our baseline system. Specifically, we slightly modify the `run_squad.py` script[5] for adjusting our dataset, while keeping the most of the original implementation. For the baseline system, we used an initial learning rate of 3e-5 with a batch size of 32 and trained for two epochs. The maximum lengths of document and query are set to 512 and 64.

### 4.2 Results

The results are shown in Table 2. Besides the baseline systems, we also include the participants' results of CMRC 2018 evaluation. We release the training and development set to the public and accepted submissions from participants to evaluate their models on the hidden test and challenge set to preserve the integrity of the evaluation process following Rajpurkar et al. (2016). As we can see that most of the participants could obtain over 80 in the test F1. While compared to F1 metric, the EM metric is substantially lower compared

[5] https://github.com/google-research/bert/blob/master/run_squad.py

to the SQuAD dataset (usually within 10 points). This suggests that how to determine the exact span boundary in Chinese machine reading comprehension plays a key role to improve the system performance.

Not surprisingly, as shown in the last column of Table 2, though the top-ranked systems obtain decent scores on the development and test set, they are failed to give satisfactory results on the challenge set. However, as we can see that the estimated human performance on the development, test, and challenge set are relatively similar, where the challenge set gives slightly lower scores. We also observed that though Z-Reader obtains best scores on the test set, it failed to give consistent performances on the EM metric of the challenge set. This suggests that the current reading comprehension models are relatively not capable of handling difficult questions that need comprehensive reasoning among several clues in the passage.

BERT-based approaches show competitive performance against participants submissions.[6] Though traditional models have higher scores in the test set, when it comes to the challenge set, BERT-based baselines are consistently higher, demonstrating that rich representation provided by BERT is beneficial for solving harder questions and generalize well among both easy and hard questions.

## 5 Conclusion

In this work, we propose a span-extraction dataset for Chinese machine reading comprehension. The dataset is annotated by human experts with near 20,000 questions as well as a challenging set which is composed of the questions that need reasoning over multiple clues. The evaluation results show that the machine could give excellent scores on the development and test set with only near 10 points below the estimated human performance in F1-score. However, when it comes to the challenge set, the scores are declining drastically while the human performance remains almost the same with the non-challenge set, indicating that there are still potential challenges in designing more sophisticated models to improve the performance. We hope the release of this dataset could bring language diversity in machine reading comprehension task, and accelerate further investigation on solving the questions that need comprehensive reasoning over multiple clues.

[6] As CMRC 2018 workshop was held before the publication of BERT, systems of the participants are not based on BERT.

## Open Challenge

We would like to invite more researchers doing experiments on our CMRC 2018 dataset and evaluate on the hidden test and challenge set to further test the generalization of the models. You can follow the instructions on our CodaLab Worksheet to submit your model via https://bit.ly/2ZdS8Ct

## Acknowledgments

We would like to thank the anonymous reviewers for their thorough reviewing and providing thoughtful comments to improve our paper. We would like to thank our resource team for annotating and verifying evaluation data. Also, we thank the Seventeenth China National Conference on Computational Linguistics (CCL 2018)[7] and Changsha University of Science and Technology for providing venue for evaluation workshop. This work was supported by the National Natural Science Foundation of China (NSFC) via grant 61976072, 61632011 and 61772153.

## References

Wanxiang Che, Zhenghua Li, and Ting Liu. 2010. Ltp: A chinese language technology platform. In *Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations*, pages 13–16. Association for Computational Linguistics.

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. 2017. Attention-over-attention neural networks for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 593–602. Association for Computational Linguistics.

Yiming Cui, Ting Liu, Zhipeng Chen, Wentao Ma, Shijin Wang, and Guoping Hu. 2018. Dataset for the first evaluation on chinese machine reading comprehension. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. European Language Resources Association (ELRA).

Yiming Cui, Ting Liu, Zhipeng Chen, Shijin Wang, and Guoping Hu. 2016. Consensus attention-based neural networks for chinese reading comprehension.

[7] http://www.cips-cl.org/static/CCL2018/index.html

In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1777–1786. The COLING 2016 Organizing Committee.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William Cohen, and Ruslan Salakhutdinov. 2017. Gated-attention readers for text comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1832–1846. Association for Computational Linguistics.

Wei He, Kai Liu, Yajuan Lyu, Shiqi Zhao, Xinyan Xiao, Yuan Liu, Yizhong Wang, Hua Wu, Qiaoqiao She, Xuan Liu, et al. 2017. Dureader: a chinese machine reading comprehension dataset from real-world applications. *arXiv preprint arXiv:1711.05073*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1684–1692.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The goldilocks principle: Reading children's books with explicit memory representations. *arXiv preprint arXiv:1511.02301*.

Minghao Hu, Yuxing Peng, Zhen Huang, Xipeng Qiu, Furu Wei, and Ming Zhou. 2018. Reinforced mnemonic reader for machine reading comprehension. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 4099–4106. International Joint Conferences on Artificial Intelligence Organization.

Rudolf Kadlec, Martin Schmid, Ondřej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 908–918. Association for Computational Linguistics.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794. Association for Computational Linguistics.

Ting Liu, Yiming Cui, Qingyu Yin, Wei-Nan Zhang, Shijin Wang, and Guoping Hu. 2017. Generating and exploiting large-scale pseudo training data for zero pronoun resolution. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 102–111. Association for Computational Linguistics.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392. Association for Computational Linguistics.

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 193–203.

Chih Chieh Shao, Trois Liu, Yuting Lai, Yiying Tseng, and Sam Tsai. 2018. Drcd: a chinese machine reading comprehension dataset. *arXiv preprint arXiv:1806.00920*.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2016. Newsqa: A machine comprehension dataset. *arXiv preprint arXiv:1611.09830*.

Shuohang Wang and Jing Jiang. 2016. Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905*.

Wei Wang, Ming Yan, and Chen Wu. 2018. Multi-granularity hierarchical attention fusion networks for reading comprehension and question answering. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1705–1714. Association for Computational Linguistics.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 189–198. Association for Computational Linguistics.

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604*.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. Qanet: Combining local convolution

with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*.

# MICRON: Multigranular Interaction for Contextualizing RepresentatiON in Non-factoid Question Answering

**Hojae Han***, **Seungtaek Choi***, **Haeju Park, Seung-won Hwang**
Department of Computer Science, Yonsei University, Seoul, Korea
{stovecat, hist0613, phj0225, seungwonh}@yonsei.ac.kr

## Abstract

This paper studies the problem of non-factoid question answering, where the answer may span over multiple sentences. Existing solutions can be categorized into representation- and interaction-focused approaches. We combine their complementary strength, by a hybrid approach allowing multi-granular interactions, but represented at word level, enabling an easy integration with strong word-level signals. Specifically, we propose **MICRON: Multigranular Interaction for Contextualizing RepresentatiON**, a novel approach which derives contextualized *uni*-gram representation from *n*-grams. Our contributions are as follows: First, we enable multi-granular matches between question and answer *n*-grams. Second, by contextualizing word representation with surrounding n-grams, MICRON can naturally utilize word-based signals for query term weighting, known to be effective in information retrieval. We validate MICRON in two public non-factoid question answering datasets: WikiPassageQA and InsuranceQA, showing our model achieves the state of the art among baselines with reported performances on both datasets.

## 1 Introduction

Non-factoid questions, unlike factoid questions answered by short facts like a word or a phrase, may get answered by a long answer spanning across multiple sentences. Following the definition in (Guo et al., 2019), neural approaches for this task can be roughly categorized into **representation-** and **interaction-focused** approaches.

First, **representation-focused** approaches (Rücklé and Gurevych, 2017; Shao et al., 2019) encode query and answer into vectors of the same size, and match the two by computing vector similarity. Models in this category have advantages of efficiency, as representations can be pre-computed and indexed for efficient retrieval. However, structural information, such as some question word matching another in answer, is missing in this representation. In addition, structural information can be diluted when squeezing a long text into a single vector. This weakness is often complemented by auxiliary information such as attention (Tan et al., 2016; Santos et al., 2016; Wang and Jiang, 2016). Figure 1a illustrates a representative architecture in this category (Shao et al., 2019).

Second, **interaction-focused** approaches aim to preserve structural information above. A naive structural information is a matrix storing pairwise word interaction, or 1:1. However, due to a typical length difference between a question and a long answer in our problem setting, most answer words are left unmatched, except a few *uni*-gram in the answer. Later work relaxes 1:1 constraint, to 1:N and M:N, by allowing a match to *n-gram* (1:N) or a match between query and answer *bi*-grams (2:2). A state-of-the-art in this category (Rücklé et al., 2019), shown in Figure 1b, uses *bi*-gram Convolutional Neural Network (CNN) to represent query/answer *bi*-grams and their interactions. Similar architecture was generalized for N:N or N:M matches (Song et al., 2019; Chen et al., 2018), which may introduce a new challenge of multi-granular interaction we discuss later.

Our work is of combining the strength of the two, as shown in Figure 1c. We illustrate our technical contributions using the following running example:

**Example 1** *Consider a running example of matching a question, "Who is in charge of this education process", with a matching passage on "the institution of higher learning". Interaction between a query bi-gram "education process" and the 5-gram "the institution of higher learning" is a key*

*The authors contribute equally to this paper.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5890–5895,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

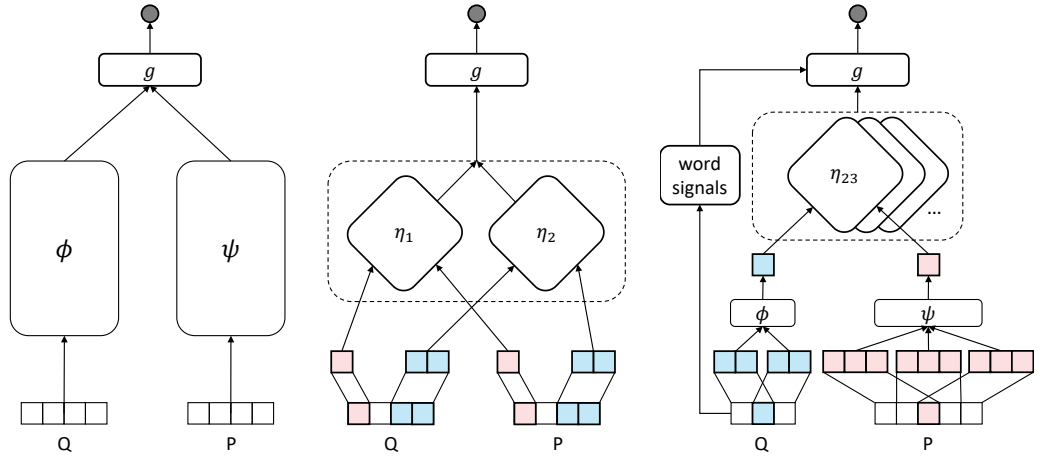


(a) Representation-focused (b) Interaction-focused (c) Ours: MICRON

Figure 1: Comparative illustration of three approaches.

*indicator explaining this match. In addition, external word-level importance signals, such as Inverse Document Frequency (IDF), are observed to be simple yet most powerful (Guo et al., 2016), in matching a short query (or, question) with a long document, as in information retrieval or non-factoid question answering scenarios. For our example question with eight words, the IDF weight is highest for education, appearing rarely in other questions, while that is lower for common words.*

Below are our two key contributions, inspired by the above running example.

**1) Multigranular interaction**: Figure 1c shows a dotted area, where interaction between *m*- and *n*-grams are represented. This enables matching between different sized n-grams: $\eta_{25}$ enables the interaction between the *bi*-gram *"education process"* and the 5-gram *"the institution of higher learning"* in our running example. However, existing multigranular interaction (Chen et al., 2018) cannot combine word-level signal, such as a high IDF score of word "*education*".

**2) N-gram contextualized word representation**: Our next step is to combine this matching signal into a contextualized word representation For example, we represent word *higher* as an aggregation of its participating consecutive 5-grams, where *"... the institution of higher"* and *"the institution of higher ..."* disambiguate that the term should not be matched a question on *"high school"*. Similarly, question word *education* is represented by surrounding 2-grams: *"education process"* and *"of education"*. Contextualizing into word-level representation makes it natural to combine with word-level IDF scores in the model, and also enables indexing (Hwang and Chang, 2005). This shares the spirit of contextualized embedding, such as BERT (Devlin et al., 2018) and ELMo (Peters et al., 2018), but specialized for **short-distance** phrase context **localized** within question and passage.

We summarize the main contributions of this paper as follows. First, we utilize multigranular interaction to extract important information from the question/passage matching by proposing **MICRON: Multigranular Interaction for Contextualizing RepresentatiON**. Second, we leverage strong word-level signals, which we will discuss later.

We evaluate our method in two public non-factoid QA datasets: WikiPassageQA (Cohen et al., 2018) and InsuranceQA (Feng et al., 2015). The results show that our model achieves the state of the art among baselines with reported performances on both datasets. Our source code is freely-available at `https://github.com/stovecat/MICRON` for further study.

## 2 Our approach

In this section, we introduce our method in detail. MICRON mainly consists of three modules: encoding module, matching module, and scoring module. We use a Siamese architecture for encoding module, which is a common setting in our target problem (Rücklé and Gurevych, 2017; Shao et al., 2019; Rücklé et al., 2019).

**Encoding Module** For a word vector sequence $W \in \mathbb{R}^{|W| \times d}$ with dimensionality $d$, we encode

it by $n$-gram CNN as the following:

$$\Gamma_n(W) = \texttt{n-gramCNN}(W) \quad (1)$$

where $n$ is the window size of $n$-gram CNN. Each $\Gamma_n(W) \in \mathbb{R}^{|W| \times d}$ represents $n$-gram semantics.

As a distinction from other interaction-focused approaches, we introduce an additional **Contextualization Layer** $\Phi$, which returns a word representation, contextualized by surrounding $n$-gram phrases of the word belongs to. In our work, we define $\Phi$ as the arithmetic mean of $n$-gram representations, formalized as follows[1]:

$$[\Phi_n(W)]_k = \frac{\sum_{i=1}^{n} [\Gamma_n(W)]_{k-i+1}}{n} \quad (2)$$

where $[\Gamma_n(W)]_k$ is $k$-th row vector of $\Gamma_n(W)$, and each row of $\Phi_n(W) \in \mathbb{R}^{|W| \times d}$ is the contextualized $n$-gram representation, corresponding to each word.

**Matching Module** Query and candidate answer $Q \in \mathbb{R}^{|Q| \times d}$ and $P \in \mathbb{R}^{|P| \times d}$ can be encoded into $\Phi_n(Q)$ and $\Psi_m(P)$. We build an interaction matrix $\eta_{nm}$ by computing dot product between $\Phi_n(Q)$ and $\Psi_m(P)$:

$$\eta_{nm} = \Phi_n(Q)\Psi_m(P)^T \quad (3)$$

Output matrix $\eta_{nm} \in \mathbb{R}^{|Q| \times |P|}$ contains the relevance scores of all pairs between $n$-grams in query and $m$-grams in answer.

From $\eta_{nm}$, we conduct a row-wise max-pooling to obtain $A_{nm}$, relaxing the length constraint in interactions (Rücklé et al., 2019).

$$[A_{nm}]_i = \max_j([\eta_{nm}]_{ij}) \quad (4)$$

**Scoring Module** We then aggregate the best matching scores $A_{nm}$ across all combinations of question $n$-grams and answer $m$-grams from $F = \{1, 2, 3, 5\}$ following the convention of (Shao et al., 2019), yielding the cumulative score for each question word $\gamma \in \mathbb{R}^{|Q|}$:

$$\gamma = \sum_{n \in F} \sum_{m \in F} A_{nm} \quad (5)$$

Finally, we obtain the relevance score $\Omega$ from $\gamma$ vector. Note that we could adopt any effective word-based signals $\tau \in \mathbb{R}^{|Q|}$, known a priori. By applying dot product between $\gamma$ and $\tau$, we can contrast matching scores by word importance. Specifically,

$$\Omega = \begin{cases} \gamma \cdot \tau, & \text{if } \tau \text{ exists} \\ \sum_{i=1}^{|\gamma|} \gamma_i, & \text{otherwise} \end{cases} \quad (6)$$

A widely adopted example of $\tau$ is IDF, computed either globally (treating all passages in the dataset as a corpus) or locally (treating only candidate passages of given question as a corpus) (Blair-Goldensohn et al., 2003). Note that effective word-level signals may depend on the characteristic of dataset. We will further show empirically which measure is more effective for each dataset and explain why in later section.

**Loss function** Our model is trained by the loss function studied in (Cohen and Croft, 2016):

$$L = \sum_{q \in Q} (1 - (\mu_{q_r} - \max q_{nr}))BCE_q \quad (7)$$

where $BCE_q$ is the standard binary cross entropy for the question, $\mu_{q_r}$ is the mean score of all relevant answers and $\max q_{nr}$ is the max score of all irrelevant answers for $q$.

## 3 Experiments

### 3.1 Dataset

We evaluate MICRON on two non-factoid question answering datasets: 1) **WikiPassageQA** (Cohen et al., 2018) is a recent Wikipedia based collection. There are high contextual similarity between answers and non-answers since all candidate answers are from the same document. 2) **InsuranceQA** (Feng et al., 2015) is another well-known large-scale non-factoid QA dataset from insurance domain constructed by putting the ground truth answers into the pool and randomly sampling negative answers[2]. Table 2 shows the statistics of two datasets.

### 3.2 Baselines

We divide the models into following four categories: 1) IR scores, 2) Representation-focused, 3) Interaction-focused, and 4) MICRON in Table 1. As state of the art in one dataset is not likely to be that in another, we focused on baselines either open source or reported results on both datasets.

We implement two interaction-focused and one representation-focused baselines: **N-gram CNN**

[1]We omit $\Psi$ for simplicity. $\Phi$ and $\Psi$ are the same in our architecture.

[2]Among two test sets available, V1 and V2, we used the latter.

| Model | InsuranceQA | WikiPassageQA | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Accuracy | MAP | MRR | P@5 | P@10 | nDCG | R@5 | R@10 | R@20 |
| **IR Based** | | | | | | | | | |
| BM25[1] | 24.9 | 53.73 | 62.58 | 19.47 | 11.51 | 66.59 | 63.34 | 73.11 | 83.09 |
| **Representation-focused** | | | | | | | | | |
| CNN[1] | 24.4 | 27.33 | 31.48 | - | - | - | - | - | - |
| BiLSTM[1] | 32.4 | 46.16 | 52.89 | - | - | - | - | - | - |
| Att-BiLSTM (Tan et al., 2016)[1] | 37.9 | 47.04 | 54.36 | - | - | - | - | - | - |
| AP-BiLSTM (Santos et al., 2016)[1] | 31.9 | 46.98 | 55.20 | - | - | - | - | - | - |
| LW-BiLSTM (Rücklé and Gurevych, 2017)[1] | 36.9 | 47.56 | 54.33 | - | - | - | - | - | - |
| **Interaction-focused** | | | | | | | | | |
| CA-Wang (Wang and Jiang, 2016)[1] | 37.0 | 48.71 | 56.11 | - | - | - | - | - | - |
| COALA (Rücklé et al., 2019)[1] | 38.0 | 60.58 | 69.40 | - | - | - | - | - | - |
| COALA p-means (Rücklé et al., 2019)[1] | 39.9 | 59.29 | 68.48 | - | - | - | - | - | - |
| COALA syntax-aware (Rücklé et al., 2019)[1] | 39.5 | 60.48 | 68.75 | - | - | - | - | - | - |
| N-gram CNN | 45.8 | 56.81 | 64.53 | 21.63 | 13.20 | 69.29 | 69.52 | 81.85 | 91.35 |
| Unigram CNN | 36.5 | 55.45 | 64.93 | 20.53 | 12.64 | 68.57 | 65.54 | 78.49 | 89.13 |
| Unigram CNN w/ global IDF | 35.4 | 56.68 | 68.52 | 20.91 | 12.72 | 69.86 | 66.94 | 79.27 | 89.68 |
| Unigram CNN w/ local IDF | 29.7 | 58.66 | 68.96 | 22.50 | 13.17 | 71.23 | 71.43 | 81.91 | 91.97 |
| **Ours** | | | | | | | | | |
| MICRON | **49.8** | 59.38 | 67.17 | 22.21 | 13.13 | 71.31 | 71.09 | 81.73 | 91.46 |
| MICRON w/ global IDF | 46.7 | 59.44 | 67.87 | 22.07 | 13.13 | 71.40 | 70.82 | 81.27 | 90.75 |
| MICRON w/ local IDF | 48.0 | **63.00** | **71.03** | **23.17** | **13.82** | **74.14** | **73.77** | **85.30** | **93.07** |

Table 1: Results of the different models on the InsuranceQA and WikiPassageQA. [1] are reported in (Rücklé et al., 2019) . COALA syntax-aware (Rücklé and Gurevych, 2017) is a variant of COALA using dependency parse trees (Schuster and Manning, 2016).

| Dataset | # of Questions | | | Answer |
|---|---|---|---|---|
| | Train | Valid | Test | Length |
| WikipassageQA | 3332 | 417 | 416 | 153 |
| InsuranceQA | 10391 | 1592 | 1625 | 112 |

Table 2: Statistics of datasets

builds N:N matching matrices respectively. The size of N is the same with our method for fair comparison. **Unigram CNN** uses 1:1 word matching, and is able to utilize word-based signals as query term weighting value.

### 3.3 Implementation Details

For word embeddings, we use 300d pre-trained Glove (Pennington et al., 2014). The sequence length of the passage are all different for each dataset: 400 tokens for WikiPassageQA, 200 tokens for InsuranceQA. The dropout is applied after every layers with a keep rate of 0.7. All weights except embedding matrices are constrained by L2 regularization with constant values of $10^{-7}$ and $10^{-5}$ respectively for WikiPassageQA and InsuranceQA. We use Adam optimizer (Kingma and Ba, 2014) with an initial learning rate of $10^{-6}$ and $10^{-4}$ for each dataset. The learning parameters were chosen by the best performance on the dev set.

### 3.4 Results

Table 1 shows the results on WikiPassageQA and InsuranceQA datasets. We observe that our proposed approach, named MICRON, significantly outperforms both representation-focused and interaction-focused baselines in various evaluation metrics, achieving the best performance in both datasets.

Our finding could be summarized as below: First, we manifest the effectiveness of multigranular interaction. Compared to N-gram CNN, MICRON allows matching between different $n$-grams (e.g., 2:3, 3:5) and achieves the improvement on both datasets by 4.0% point accuracy, 2.57% point MAP respectively.

Second, we relax length constraint in $n$-gram from COALA and achieve relative gain in InsuranceQA dataset. However, this gain is marginal when the phrase is short as in WikiPassageQA dataset, considering the better performance of COALA over N-gram CNN.

Third, word-based signals may help considerably in WikiPassageQA, where both global and local IDF scores of words are vary significantly (high variance). This variance is especially high for local IDF, which serves as a strong signal as consistently observed in (Blair-Goldensohn et al., 2003). In contrast, in InsuranceQA, the variance

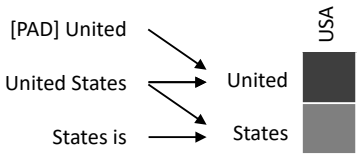


(a) An example of multigranular interaction over different $n$-grams (2:1).

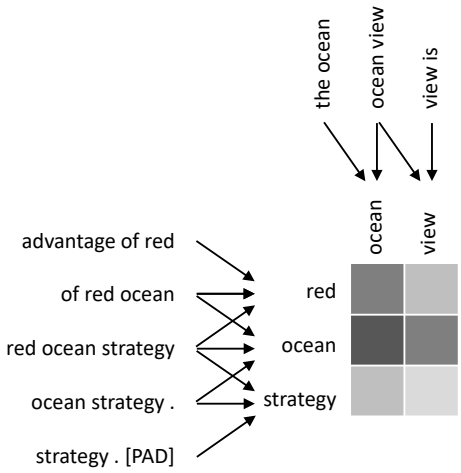


(b) An example of contextualized representation for the word *ocean*, lowering the matching score for negative candidate.

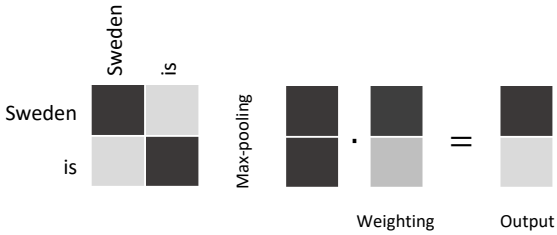


(c) Effect of word-based signals of enhancing query term weighting.

Figure 2: Qualitative examples of MICRON in the WikiPassageQA dataset.

of word signals are low. Consequently, the use of IDF cannot contribute to performance, or even contributes negatively.

### 3.5 Qualitative Examples

We illustrate several qualitative examples of MICRON in Figure 2. In Figure 2a, multigranular interaction (2:1) between the *bi*-gram *"United States"* and the *uni*-gram *"USA"* allows the matching. Figure 2b shows the case of where the contextualized representation enables to lower the matching score between *"red ocean"* and *"ocean view"*. From Figure 2c, we can see the word based signals can control the impact of each contextualized word scores: amplifying the matching of *"Sweden"*-*"Sweden"* and reducing the *"is"*-*"is"* matching.

## 4 Conclusion

In this paper, we study non-factoid question answering. Specifically, our approach is inspired by the complementary strength of representation- and interaction-focused approaches. We combine the strength of the two, by allowing multigranular interactions, but represented per-word basis, contextualized by participating n-grams. For this purpose, we propose MICRON, allowing to match flexible n-grams and to combine with word-based query term weighting, achieving the state of the art among baselines with reported performances on both datasets[3].

## Acknowledgments

This work is supported by Microsoft Research Asia and IITP funded by MSIT (2017-0-0177; XAI). Hwang is a corresponding author.

## References

Sasha Blair-Goldensohn, Kathleen McKeown, and Andrew Hazen Schlaikjer. 2003. A hybrid approach for answering definitional questions.

Haolan Chen, Fred X Han, Di Niu, Dong Liu, Kunfeng Lai, Chenglin Wu, and Yu Xu. 2018. Mix: Multi-channel information crossing for text matching. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 110–119. ACM.

Daniel Cohen and W Bruce Croft. 2016. End to end long short term memory networks for non-factoid question answering. In *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval*, pages 143–146. ACM.

Daniel Cohen, Liu Yang, and W Bruce Croft. 2018. Wikipassageqa: A benchmark collection for research on non-factoid answer passage retrieval. *arXiv preprint arXiv:1805.03797*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[3]After the submission of this paper, BERT-based approach (Xu et al., 2019) outperformed MICRON in WikiPassageQA. However, MICRON is more energy-efficient (1M parameters) than BERT (110M parameters) and still outperforms BERT in InsuranceQA

Minwei Feng, Bing Xiang, Michael R Glass, Lidan Wang, and Bowen Zhou. 2015. Applying deep learning to answer selection: A study and an open task. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 813–820. IEEE.

Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. 2016. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 55–64. ACM.

Jiafeng Guo, Yixing Fan, Liang Pang, Liu Yang, Qingyao Ai, Hamed Zamani, Chen Wu, W Bruce Croft, and Xueqi Cheng. 2019. A deep look into neural ranking models for information retrieval. *arXiv preprint arXiv:1903.06902*.

Seung-won Hwang and Kevin Chang. 2005. Optimizing access cost for top-k queries over web sources: A unified cost-based approach. In *ICDE*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Andreas Rücklé and Iryna Gurevych. 2017. Representation learning for answer selection with lstm-based importance weighting. In *IWCS 2017—12th International Conference on Computational Semantics—Short papers*.

Andreas Rücklé, Nafise Sadat Moosavi, and Iryna Gurevych. 2019. Coala: A neural coverage-based approach for long answer selection with small data. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)*.

Cicero dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *arXiv preprint arXiv:1602.03609*.

Sebastian Schuster and Christopher D Manning. 2016. Enhanced english universal dependencies: An improved representation for natural language understanding tasks. In *LREC*, pages 23–28. Portorož, Slovenia.

Taihua Shao, Xiaoyan Kui, Pengfei Zhang, and Honghui Chen. 2019. Collaborative learning for answer selection in question answering. *IEEE Access*, 7:7337–7347.

Yang Song, Qinmin Vivian Hu, and Liang He. 2019. P-cnn: Enhancing text matching with positional convolutional neural network. *Knowledge-Based Systems*.

Ming Tan, Cicero Dos Santos, Bing Xiang, and Bowen Zhou. 2016. Improved representation learning for question answer matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 464–473.

Shuohang Wang and Jing Jiang. 2016. A compare-aggregate model for matching text sequences. *arXiv preprint arXiv:1611.01747*.

Peng Xu, Xiaofei Ma, Ramesh Nallapati, and Bing Xiang. 2019. Passage ranking with weak supervsion. *arXiv preprint arXiv:1905.05910*.

# Machine Reading Comprehension Using Structural Knowledge Graph-aware Network

**Delai Qiu**[1], **Yuanzhe Zhang**[2], **Xinwei Feng**[4], **Xiangwen Liao**[1],
**Wenbin Jiang**[4], **Yajuan Lyu**[4], **Kang Liu**[2,3], **Jun Zhao**[2,3]
[1] College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China
[2] Institute of Automation, Chinese Academy of Sciences, Beijing, China
[3] University of Chinese Academy of Sciences, Beijing, China
[4] Baidu Inc., Beijing, China
noneqdl@gmail.com, {yzzhang,kliu,jzhao}@nlpr.ia.ac.cn
lxw@fzu.edu.cn, {fengxinwei,jiangwenbin,lvyajuan}@baidu.com

## Abstract

Leveraging external knowledge is an emerging trend in machine comprehension task. Previous work usually utilizes knowledge graphs such as ConceptNet as external knowledge, and extracts triples from them to enhance the initial representation of the machine comprehension context. However, such method cannot capture the structural information in the knowledge graph. To this end, we propose a Structural Knowledge Graph-aware Network (SKG) model, constructing sub-graphs for entities in the machine comprehension context. Our method dynamically updates the representation of the knowledge according to the structural information of the constructed sub-graph. Experiments show that SKG achieves state-of-the-art performance on the ReCoRD dataset.

## 1 Introduction

Machine reading comprehension (MRC) is an important subtask in natural language processing, which requires a system to read a given passage and answer questions about it. The ability of utilizing external knowledge is of great significance in an MRC system (Rajpurkar et al., 2016; Trischler et al., 2017). Latest large-scale datasets, e.g. ReCoRD (Zhang et al., 2018) specify that external knowledge is required to answer questions.

Many previous studies have introduced external knowledge in machine comprehension (Weissenborn, 2017; Mihaylov and Frank, 2018; Bauer et al., 2018). They often acquire external knowledge from structural knowledge graphs, such as ConceptNet (Speer et al., 2017) and Freebase (Tanon et al., 2016), in which knowledge is organized by triples like *"(shortage, related_to, lack)"* and *"(need, related_to, lack)"*. However, most of them fail to make full use of the structural information in the knowledge graph, and use sequence modeling methods like recurrent neural networks (RNNs) to generate the representation of knowledge, rather than based on graph structure.

In this paper, we present a Structural Knowledge Graph-aware Network (SKG) model to leverage the structural information of external knowledge. The advantages of our proposed method are two folds: a) the constructed sub-graphs contain both MRC context and external knowledge nodes, reserving the knowledge structure; b) the graph neural network is capable of updating the nodes dynamically according to the structure of sub-graphs, instead of using external knowledge as pre-trained representations. Concretely, we first construct sub-graphs from external knowledge such as ConceptNet based on the context, and initialize the representation of nodes via knowledge graph embedding method (Yang et al., 2015). Then we employ graph attention networks to dynamically update the representation of nodes on sub-graphs. Finally, we utilize the final representation of nodes to augment the representation of context via gate mechanisms.

Our contributions can be summarized as:

- We present a simple but effective method to construct sub-graphs from a knowledge graph, which can reserve the structure of knowledge;
- Graph attention networks are employed to dynamically update the representation of knowledge based on sub-graph structure, which can make full advantage of structural information of external knowledge;
- Experiments demonstrate that SKG model is able to effectively leverage external knowledge in MRC task, and achieves state-of-the-art performance on the ReCoRD dataset.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5896–5901,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

## 2 Related work

**External Knowledge Enhanced MRC Models**
There are several models that use knowledge for machine comprehension (Yang and Mitchell, 2017; Mihaylov and Frank, 2018; Weissenborn, 2017; Bauer et al., 2018; Pan et al., 2019). Mihaylov and Frank (2018) relies on the ability of the attention mechanism to retrieve relevant pieces of knowledge, and Bauer et al. (2018) employs multi-hop commonsense paths to help multi-hop reasoning. They treat retrieved knowledge triples as sequences and use sequence modeling methods to compress the representation of knowledge, which are not based on graph structure. On the contrary, we organize knowledge as sub-graphs, then update the representation of nodes on sub-graphs with graph neural network.

**Graph Neural Netwoks**
Graph neural netwoks (Kipf and Welling, 2016; Schlichtkrull et al., 2018; Veličković et al., 2017) have been shown successful on many Natural Language Processing (NLP) tasks (Cao et al., 2018; Zhou et al., 2018; Song et al., 2018; Cao et al., 2019). They are good at dealing with graph-structured data. In (Cao et al., 2018; Song et al., 2018; Cao et al., 2019), Graph Convolutional Networks (GCNs) have been applied in multi-documnent machine comprehension for multi-hop reasoning question answering, but they consider only the internal structure information in the MRC context without incorporating external knowledge. To the best of our knowledge, our work is the first to study graph attention networks in machine comprehension with external knowledge.

## 3 SKG Model

The architecture of SKG is shown in Figure 1. It contains four modules: (1) Question and paragraph modeling module, which acquires the contextual representation of question and paragraph with BERT; (2) Knowledge sub-graph construction module, which retrieves sub-graphs from knowledge graph based on the context; (3) Graph attention module, which updates the representation of nodes on graph; (4) Output layer module, which is employed to generate the final answer.

### 3.1 Question and Paragraph Modeling

We first encode the tokens with BERT (Devlin et al., 2018). BERT has become one of the most successful natural language representation models

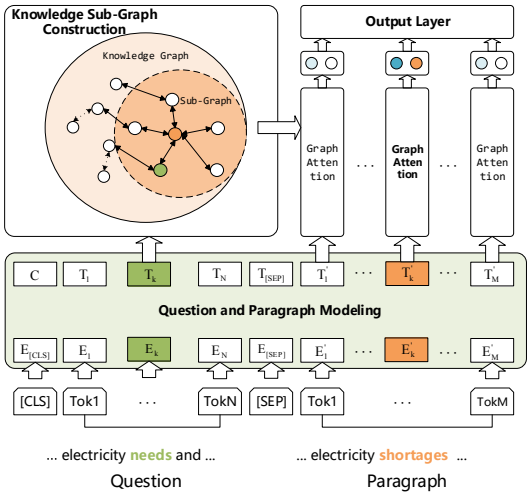


Figure 1: Framework of our SKG model.

on various NLP datasets, including SQuAD (Rajpurkar et al., 2016). BERT's model architecture is a multi-layer bidirectional Transformer (Vaswani et al., 2017) encoder, which is pre-trained on large-scale corpus. We represent the input question and paragraph as a single packed sequence as follows:

$$[CLS]Question[SEP]Paragraph[SEP]$$

where $[CLS]$ is a specific classifier token and $[SEP]$ is a sentence separator which are defined in BERT. And we use BERT to generate the contextual representation of question and paragraph. The final hidden output from BERT for the $i^{th}$ input token is denoted as $\boldsymbol{t}_{b_i} \in \mathbb{R}^H$, and $H$ is the output hidden size of BERT model.

### 3.2 Knowledge Sub-Graph Construction

We first use knowledge graph embedding approach to generate the initial representation of nodes and edges in terms of the whole knowledge graph.

We consider a knowledge triple in a knowledge graph as *(head, relation, tail)*. For the $i^{th}$ token $t_i$ in paragraph, we retrieve all triples whose head or tail contains lemmas of the token. Take token "shortage" as an example, we retrieve triples like *"(shortage, related_to, lack)"*. Then, we retrieve the neighbor triple of them, and reserve ones that contain lemmas of any token of the question. Thus we can acquire triples like *"(need, related_to, lack)"*. We reorganize these triples into a sub-graph via connecting identical entities and reserving the relations as edges in these triples. So

we can construct a simple sub-graph like *"(shortage, related_to,* ***lack****, related_to, need)"*, where "lack" is the identical entity. The sub-graph can be denoted as $g_i$ and the nodes and edges of it are initiated by the embeddings above.

### 3.3 Graph Attention

Our graph attention network is designed to update the representation of the nodes in a constructed sub-graph, which is inspired by (Veličković et al., 2017; Zhou et al., 2018). For the $i^{th}$ token $t_i$ in paragraph, its sub-graph is $g_i = \{n_1, n_2, .., n_k\}$, where $k$ is the number of nodes. And $N_j$ is the set of the $j^{th}$ node neighbors.

The representation of nodes is updated $L$ times. At the $l^{th}$ update, the updating rules are as follows, which are designed to model the interaction between the $j^{th}$ node and its neighbor nodes. In this way, the nodes are dynamically updated according to the structure of sub-graphs.

$$\boldsymbol{h}_j^{l+1} = \sum_{n=1}^{N_j} \alpha_n \boldsymbol{t}_n^l \tag{1}$$

$$\alpha_n = \frac{\exp(\beta_n)}{\sum_{j=1}^{N_j} \exp(\beta_j)} \tag{2}$$

$$\beta_n = \left(\mathbf{W}_r^l \boldsymbol{r}_n^l\right)^\top \tanh\left(\mathbf{W}_h^l \boldsymbol{h}_n^l + \mathbf{W}_t^l \boldsymbol{t}_n^l\right) \tag{3}$$

where $\boldsymbol{h}_j^l \in \mathbb{R}^d$ is hidden state of the $j^{th}$ node, and its neighbor's hidden state is $\boldsymbol{t}_n^l$, and the hidden state of relation is $\boldsymbol{r}_n^l$, and $d$ is the hidden state dimension. $\mathbf{W}_h^l$, $\mathbf{W}_t^l$ and $\mathbf{W}_r^l$ are trainable weight matrices for the node, its neighbors and relations respectively. After $L$ updates, we can get the final hidden state of the central node as the final representation, which can be denoted as $\boldsymbol{t}_{k_i}$.

### 3.4 Output Layer

In the output layer, we combine this knowledge representation $\boldsymbol{t}_{k_i}$ with the textual representation $\boldsymbol{t}_{b_i}$ via a sigmoid gate, since external knowledge is not always necessary for reasoning.

$$w_i = \sigma((W\left[\boldsymbol{t}_{b_i}; \boldsymbol{t}_{k_i}\right])) \tag{4}$$

$$\boldsymbol{t'}_i = w_i \odot \boldsymbol{t}_{b_i} + (1 - w_i) \odot \boldsymbol{t}_{k_i} \tag{5}$$

We denote $T = \{\boldsymbol{t'}_1, \boldsymbol{t'}_2, ..., \boldsymbol{t'}_n\}$ as the final representation, where $\boldsymbol{t'}_i \in \mathbb{R}^H$. And we study a start vector $S \in \mathbb{R}^H$ and an end vector $E \in \mathbb{R}^H$ like (Devlin et al., 2018), which take $T$ as input. Then, the probability of the $i^{th}$ token being the start of the answer span is computed as a dot product between $T_i$ and $S$ followed by a softmax over all of the words in the paragraph:

$$P_i^s = \frac{e^{S \cdot T_i}}{\sum_j e^{S \cdot T_j}} \tag{6}$$

Let $P_i^e$ be the probability of the $i^{th}$ token to be the end of an answer span, which can be calculated by the same above formula, and the maximum scoring span is used as the answer. The training objective is the loglikelihood of the correct start and end positions.

## 4 Experiments

### 4.1 Dataset

**ReCoRD** We report results on ReCoRD dataset (Zhang et al., 2018), a large-scale dataset for machine comprehension requiring external knowledge. There are 100,730, 10,000 and 10,000 examples in the training set, the development set and the test set respectively. The test set is not public, which needs to submit the model to the organization[1] to get the results.

**External Knowledge** We consider two knowledge sources as our external knowledge: WordNet and ConceptNet. For WordNet, we use the preprocessed data provided by Bordes et at. (2013), which contains 151,442 triples with 40,943 synsets and 18 relations. For ConceptNet, we use the preprocessed data provided by Bauer et al. (2018), which contains 2,808,998 triples with 978,672 entities and 46 relations.

### 4.2 Implementation Details

Our model is implemented with pytorch[2], and uses the framework[3] for BERT model. We employ the open-source framework OpenKE (Han et al., 2018) to obtain the embedding of entities and relations with the BILINEAR model (Yang et al., 2015). The size of embedding of entities and relations is 100. The update times $L$ of graph attention network is set to 5. We use Adam optimizer. The learning rate uses the linear schedule to decrease from 0.00003 to 0.

[1] https://sheng-z.github.io/ReCoRD-explorer/
[2] https://github.com/pytorch/pytorch
[3] https://github.com/huggingface/pytorch-pretrained-BERT

| Model | EM | | F1 | |
|---|---|---|---|---|
| | Dev | Test | Dev | Test |
| QANet (Yu et al., 2018) | 35.38 | 36.51 | 36.75 | 37.79 |
| SAN (Liu et al., 2018) | 38.14 | 39.77 | 39.09 | 40.72 |
| DocQA w/o ELMo (Clark and Gardner, 2018) | 36.59 | 38.52 | 37.89 | 39.76 |
| DocQA w/ ELMo (Clark and Gardner, 2018) | 44.13 | 45.44 | 45.39 | 46.65 |
| SKG+BERT-Large(ours) | **70.94** | **72.24** | **71.55** | **72.78** |

Table 1: The performance of different models on ReCoRD dataset.

| Model | EM | F1 |
|---|---|---|
| BERT-Base( 2018) | 54.03 | 55.99 |
| SKG+BERT-Base(ours) | 60.26 | 60.79 |
| BERT-Large( 2018) | 64.28 | 66.60 |
| SKG+BERT-Large(ours) | 70.94 | 71.55 |

Table 2: The effectiveness of introducing external knowledge.

| Model | EM | F1 |
|---|---|---|
| MHPGM+NOIC( 2018) | 28.36 | 29.29 |
| KG+LSTM+Bert-Base | 58.01 | 58.49 |
| SKG+BERT-Base(ours) | 60.26 | 60.79 |

Table 3: The results of different ways of introducing external knowledge.

### 4.3 Results and Analysis

**Results on ReCoRD**

We choose several baselines: (1) QANet (Yu et al., 2018) is one of the top MRC models, which is different from many other MRC models due to the use of transformer (Vaswani et al., 2017). (2) SAN (Liu et al., 2018) is a top-rank MRC model, which employs a stochastic answer module. (3) DocQA (Clark and Gardner, 2018) is a strong baseline model, which consists of bidirectional attention flow and self-attention.

The results of different models are shown in Table 1. Our SKG+BERT-Large[4] model achieves better performance than all previous published models, which is 26.13% higher in value than the state-of-art model DocQA with ELMo.

**The Effectiveness of External Knowledge**

Moreover, ablation experimental results on the dev set[5] are given in Table 2. Once we remove the module incorporating external knowledge, the model degenerates into the fine-tuing BERT model, and the results show significant performance drop with 4.80% and 4.95% on different pre-trained model sizes respectively. The results demonstrate that our model can effectively utilize external knowledge.

**Different Ways to Use External Knowledge**

In the original paper of ReCoRD (Zhang et al., 2018), there is no research on existing models that use external knowledge to improve MRC task. So we study the recent model MHPGM+NOIC (Bauer et al., 2018) , which utilizes multi-hop relational knowledge paths from ConceptNet. As shown in Table 3, our SKG model is more proper for introducing external knowledgse. In addition, to investigate the impact of the structural information on performance, we replace our sub-graph construction module with KG+LSTM, which retrieves knowledge triples without reconstructing the structure of them, and considers paths among them as sequences. We employ Long Short-Term Memory (LSTM) model to generate the representation of knowledge. As shown in Table 3, the performance drops 2.3% in F1, which means that the incorporation of structural information in the knowledge graph is able to make better use of external knowledge.

## 5 Conclusion

We propose SKG model for improving machine comprehension. Rather than treating triples from knowledge graph independently and separately, we construct sub-graphs from external knowledge. Then we generate the representation of knowledge with graph attention networks to improve the representation of context. Experimental results indicate that our model achieves the best performance in the challenging ReCoRD dataset.

[4]Our first module is based on different BERT model sizes. BERT-Base contains 110M parameters and BERT-Large contains 340M parameters.

[5]Due to several weeks evaluation wait-time on the non-public test set, we just report results of our supplementary experiments on the dev set. In previous submissions, the results in dev set are consistent with test set.

## Acknowledgements

This work was supported by National Key R&D Program of China under Grant 2018YFB1005100, the National Natural Science Foundation of China (No.61533018, No.U1605251). This work was also supported by the Open Project of National Laboratory of Pattern Recognition at the Institute of Automation of the Chinese Academy of Sciences (201900041) and the independent research project of National Laboratory of Pattern Recognition.

## References

Lisa Bauer, Yicheng Wang, and Mohit Bansal. 2018. Commonsense for generative multi-hop question answering tasks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 4220–4230.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2018. Question answering by reasoning across documents with graph convolutional networks. *CoRR*, abs/1808.09920.

Yu Cao, Meng Fang, and Dacheng Tao. 2019. Bag: Bi-directional attention entity graph convolutional network for multi-hop reasoning question answering. *arXiv preprint arXiv:1904.04969*.

Christopher Clark and Matt Gardner. 2018. Simple and effective multi-paragraph reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 845–855.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Xu Han, Shulin Cao, Xin Lv, Yankai Lin, Zhiyuan Liu, Maosong Sun, and Juanzi Li. 2018. Openke: An open toolkit for knowledge embedding. In *Proceedings of EMNLP*, pages 139–144.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Xiaodong Liu, Yelong Shen, Kevin Duh, and Jianfeng Gao. 2018. Stochastic answer networks for machine reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1694–1704.

Todor Mihaylov and Anette Frank. 2018. Knowledgeable reader: Enhancing cloze-style reading comprehension with external commonsense knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 821–832.

Xiaoman Pan, Kai Sun, Dian Yu, Heng Ji, and Dong Yu. 2019. Improving question answering with external knowledge.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer.

Linfeng Song, Zhiguo Wang, Mo Yu, Yue Zhang, Radu Florian, and Daniel Gildea. 2018. Exploring graph-structured passage representation for multi-hop reading comprehension with graph neural networks. *CoRR*, abs/1809.02040.

Robert Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 4444–4451.

Thomas Pellissier Tanon, Denny Vrandecic, Sebastian Schaffert, Thomas Steiner, and Lydia Pintscher. 2016. From freebase to wikidata: The great migration. In *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11 - 15, 2016*, pages 1419–1428.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. Newsqa: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 191–200.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Dirk Weissenborn. 2017. Reading twice for natural language understanding. *CoRR*, abs/1706.02596.

Bishan Yang and Tom M. Mitchell. 2017. Leveraging knowledge bases in lstms for improving machine reading. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1436–1446.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. *CoRR*, abs/1804.09541.

Sheng Zhang, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, Kevin Duh, and Benjamin Van Durme. 2018. Record: Bridging the gap between human and machine commonsense reading comprehension. *CoRR*, abs/1810.12885.

Hao Zhou, Tom Young, Minlie Huang, Haizhou Zhao, Jingfang Xu, and Xiaoyan Zhu. 2018. Commonsense knowledge aware conversation generation with graph attention. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden.*, pages 4623–4629.

# Answering Conversational Questions on Structured Data without Logical Forms

**Thomas Müller, Francesco Piccinno, Massimo Nicosia, Peter Shaw, Yasemin Altun**
Google Research
{thomasmueller,piccinno,massimon,petershaw,altun}@google.com

## Abstract

We present a novel approach to answering sequential questions based on structured objects such as knowledge bases or tables without using a logical form as an intermediate representation. We encode tables as graphs using a graph neural network model based on the Transformer architecture. The answers are then selected from the encoded graph using a pointer network. This model is appropriate for processing conversations around structured data, where the attention mechanism that selects the answers to a question can also be used to resolve conversational references. We demonstrate the validity of this approach with competitive results on the Sequential Question Answering (SQA) task (Iyyer et al., 2017).

## 1 Introduction

In recent years, there has been significant progress on conversational question answering (QA), where questions can be meaningfully answered only within the context of a conversation (Iyyer et al., 2017; Choi et al., 2018; Saha et al., 2018). This line of work, as in single QA setup, falls into two main categories, (i) the answers are extracted from some text in a reading comprehension setting, (ii) the answers are extracted from structured objects, such as knowledge bases or tables. The latter is commonly posed as a semantic parsing task, where the goal is to map questions to some logical form which is then executed over the knowledge base to extract the answers.

In semantic parsing, there is extensive work on using deep neural networks for training models over manually created logical forms in a supervised learning setup (Jia and Liang, 2016; Ling et al., 2016; Dong and Lapata, 2018). However, creating labeled data for this task can be expensive and time-consuming. This problem resulted in research that investigates semantic parsing with weak supervision where training data consists of questions and answers along with the structured resources to recover the logical form representations that would yield the correct answer (Liang et al., 2016; Iyyer et al., 2017).

In this paper, we follow this line of research and investigate answering sequential questions with respect to structured objects. In contrast to previous approaches, instead of learning the intermediate logical forms, we propose a novel approach that encodes the structured resources, i.e. tables, along with the questions and answers from the context of the conversation. This approach allows us to handle conversational context without the definition of detailed operations or a vocabulary dependent on the logical form formalism that are required in the weakly supervised semantic parsing approaches.

We present empirical performance of the proposed approach on the Sequential Question Answering task (SQA) (Iyyer et al., 2017) which improves the state-of-the-art performance on all questions, particularly on the follow-up questions that require effective encoding of the context.

## 2 Approach

We build a QA model for a sequence of questions that are asked about a table and can be answered by selecting one or more table cells.

### 2.1 Graph Formulation

We encode tables as graphs by representing columns, rows and cells as nodes. Figure 1 shows an example graph representing how we encode the table in relation to $q_2$, which is the follow up question to $q_1$. Within a column, cells with identical texts are collapsed into a single node. In the example graph, we only create a single node for *"Toronto"* and a single node for *"Montreal"*. We then add 4 directed edges that connect columns



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5902–5910, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

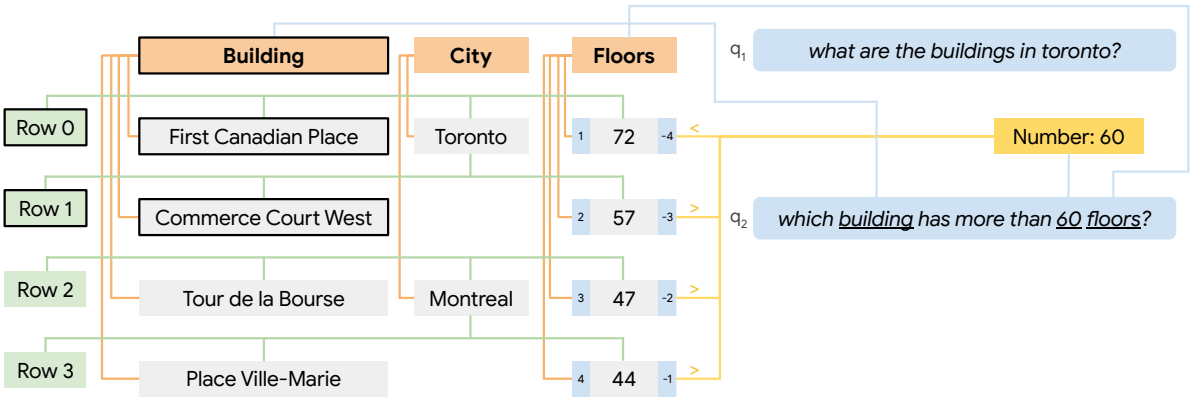


Figure 1: Example of a table encoded as a graph in relation to $q_2$, the follow up question to $q_1$. Columns (orange boxes), rows (green boxes) and cells (gray boxes) are represented by interconnected nodes. Questions words are linked to the table by edit distance (blue lines). Numerical values in question and table are connected by comparison relations (yellow lines). Additionally numerical cells are annotated with their numerical rank and inverse rank with respect to the column (small blue boxes). The answer column, rows and cells of $q_1$ are also marked (nodes with borders).

and rows to cells, one in either direction (orange and green edges in the figure).

The question is represented by a node covering the entire question text and a node for each token. The main question node is connected to each token, column and cell node. [1]

Nodes have associated nominal feature sets. All nodes have a feature indicating their type: column, row, cell, question and question token. The text in column (i.e., the column name), cell, question and token nodes are added to the corresponding node feature set adopting a bag-of-words representation. Column, row and cell nodes have additional features that indicate their column (for cell and column nodes) and row (for cell and row nodes) indexes.

We align question tokens with column names and cell text using the Levenshtein edit distance between n-grams in the question and the table text, similar to previous work (Shaw et al., 2019). In particular, we score every question n-gram with the normalized edit distance [2] and connect the cell to the token span if the score is $> 0.5$. Through the alignment, the cell is connected to all the tokens of the matching span and the binned score is added as an additional feature to the cell. In Figure 1, the *"building"* and *"floors"* tokens in the questions are connected to the matching *"Building"* and *"Floors"* column nodes from the table.

[1] We do not show some of these connections in the figure to avoid clutter.

[2] $\text{ned}(v, w) = \frac{\text{ed}(v,w)}{\max(|v|,|w|)}$

**Numeric Operations** In order to allow operations over numbers and date expressions, we extend our graph with a set of relations involving numerical values in the question and table cells. We infer the type of the numerical values in a column, such as the ones in the *"Floors"* column, by picking their most frequent type (number or date). Then, we add special features to each cell in the column: the rank and inverse rank of the value in the cell, considering the other cell values from the same column. These features allow the model to answer questions such as *"what is the building with most floors?"*. In addition, we add a new node to the graph for each numeric expression from the question (such as the number *60* from the second question in Figure 1), and we connect it to the tokens it spans. The numerical nodes originated from the question are connected to the table cells containing numerical values. The connection type encodes the result of the comparison between the question and cell values, lesser, greater or equal, as shown in the figure (yellow edges). This relations allow the model to answer questions such as *"which buildings have more than 50 floors?"*.

**Context** We extend the model to capture conversational context by using the feature-based encoding in the graph formulation. In order to handle follow-up questions, we add the previous answers to the graph by marking all the answer rows, columns and cells with nominal features. The nodes with borders in Figure 1 contain the answers to the first question $q_1$: *"what are the buildings*

*in toronto?"*. In the example, the first two rows receive a feature ANSWER_ROW, the *"building"* column a feature ANSWER_COLUMN and *"First Canadian Place"* and *"Commerce Court West"* a feature ANSWER_CELL. Notice that the content of $q_1$ is not encoded in the graph, only its answers.

### 2.2 Node Representations

Before the initial encoder layer, all nodes are mapped to vector representations using learned embeddings. For nodes with multiple features, such as column and cell nodes, we reduce the set of feature embeddings to a single vector using the mean. We also concatenate an embedding encoding whether the node represents a question token, or not.

### 2.3 Encoder

We use a Graph Neural Network (GNN) encoder based on the Transformer (Vaswani et al., 2017). The only modification is that the self-attention mechanism is extended to consider the edge label between each pair of nodes.

We follow the formulation of Shaw et al. (2019) that uses additive edge vector representations. The self-attention weights are therefore calculated as:

$$s_{ij} = (\mathbf{W}^q x_i)^\intercal (\mathbf{W}^k x_j + r_{ij}), \qquad (1)$$

where $s_{ij}$ is the unnormalized attention weight for the node vector representations $x_i$ and $x_j$, and $\mathbf{W}^q$ and $\mathbf{W}^k$ are parameter matrices. This extension introduces $r_{ij}$ to the calculation, which is a vector representation of the edge label between the two nodes. Edge vectors are similarly added when summing over node representations to produce the new output representations.

We use edge labels corresponding to relative positions between tokens, alignments between tokens and table elements, and relations between table elements, as described in Section 2.1. These amount to 9 fixed edge labels in the graph (4 between rows/cells/columns, 2 between question and cells/columns, and 3 numeric relations) and a tunable number of relative token positions).

### 2.4 Answer Selection

We extend the Transformer decoder to include a copy mechanism based on a pointer network (Vinyals et al., 2015). The copy mechanism allows the model to predict sequences of answer

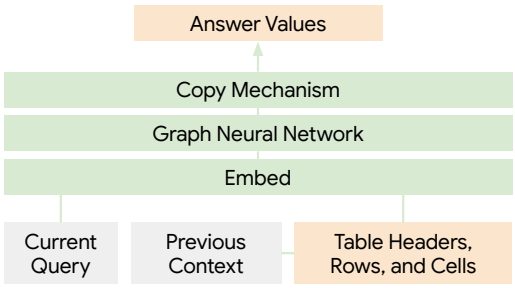


Figure 2: The model input is a graph, with nodes corresponding to query words and table elements, and edge labels capturing their relations. A graph neural network encoder generates contextualized node representations. Answers values are selected from nodes corresponding to table elements.

columns and rows from the input, rather than select symbols from an output vocabulary. Figure 2 visualizes the entire model architecture.

## 3 Related Work

Semantic parsing models can be trained to produce gold logical forms using an encoder-decoder approach (Suhr et al., 2018) or by filling templates (Xu et al., 2017; Peng et al., 2017; Yu et al., 2018). When gold logical forms are not available, they are typically treated as latent variables or hidden states and the answers or denotations are used to search for correct logical forms (Yih et al., 2015; Long et al., 2016; Iyyer et al., 2017). In some cases, feedback from query execution is used as a reward signal for updating the model through reinforcement learning (Zhong et al., 2017; Liang et al., 2016, 2018; Agarwal et al., 2019) or for refining parts of the query (Wang et al., 2018). In our work, we do not use logical forms or RL, which can be hard to train, but simplify the training process by directly matching questions to table cells.

Most of the QA and semantic parsing research focuses on single turn questions. We are interested in handling multiple turns and therefore in modeling context. In semantic parsing tasks, logical forms (Iyyer et al., 2017; Sun et al., 2018b; Guo et al., 2018) or SQL statements (Suhr et al., 2018) from previous questions are refined to handle follow up questions. In our model, we encode answers to previous questions by marking answer rows, columns and cells in the table, in a non-autoregressive fashion.

In regards to how structured data is represented, methods range from encoding table information,

metadata and/or content, (Gur et al., 2018; Sun et al., 2018b; Petrovski et al., 2018) to encoding relations between the question and table items (Krishnamurthy et al., 2017) or KB entities (Sun et al., 2018a). We also encode the table structure and the question in an annotation graph, but use a different modelling approach.

## 4 Experimental Setup

We evaluate our method on the SequentialQA (SQA) dataset (Iyyer et al., 2017), which consists of sequences of questions that can be answered from a given table. The dataset is designed so that every question can be answered by one or more table cells. It consists of 6,066 answer sequences containing 17553 questions (2.9 question per sequence on average). Table 2 shows an example.

We lowercase and remove punctuation from questions, cell texts and column names. We then split each input utterance on spaces to generate a sequence of tokens.[3] We only keep the most frequent 5,000 word types and map everything else to one of 2,000 OOV buckets. Numbers and dates are parsed in a similar way as in the DynSp and the Neural Programmer (Neelakantan et al., 2016a).

We use the Adam optimizer (Kingma and Ba, 2014) for optimization and tune hyperparameters with Google Vizier (Golovin et al., 2017). More details and the hyperparameter ranges can be found in the appendix (A).

All numbers given for our model are averaged over 5 independent runs with different random initializations.

## 5 Results

We compare our model to Float Parser (FP) (Pasupat and Liang, 2015), Neural Programmer (NP) (Neelakantan et al., 2016b), DynSp (Iyyer et al., 2017) and Camp (Sun et al., 2018b) in Table 1.

We observe that our model improves the SOTA from 45.6% by Camp to 55.1% in question accuracy (ALL), reducing the relative error rate by 18%. For the initial question (POS1), however, it is behind DynSp by 3.7%. More interestingly, our model handles follow up questions especially well outperforming the previously best model FP by 20% on POS3, a 28% relative error reduction.

[3] Whitespace tokenization simplifies the preprocessing but we can expect an off-the-shelf tokenizer to work as good or even better.

| Model | ALL | SEQ | POS1 | POS2 | POS3 |
|---|---|---|---|---|---|
| FP * | 34.1 | 7.2 | 52.6 | 25.6 | 25.9 |
| NP * | 39.4 | 10.8 | 58.9 | 35.9 | 24.6 |
| DynSp | 42.0 | 10.2 | **70.9** | 35.8 | 20.1 |
| FP † * | 33.2 | 7.7 | 51.4 | 22.2 | 22.3 |
| NP † * | 40.2 | 11.8 | 60.0 | 35.9 | 25.5 |
| DynSp † | 44.7 | 12.8 | 70.4 | 41.1 | 23.6 |
| Camp † * | 45.6 | 13.2 | 70.3 | 42.6 | 24.8 |
| Ours * | 45.1 | 13.3 | 67.2 | 42.4 | 26.4 |
| Ours † * | **55.1** | **28.1** | 67.2 | **52.7** | **46.8** |
| Ours † * (RA) | 61.7 | 28.1 | 67.2 | 60.1 | 57.7 |

Table 1: SQA test results. † marks contextual models using the previous question or the answer to the previous question. * marks the models that use the table content. RA denotes an oracle model that has access to the previous reference answer at test time. ALL is the average question accuracy, SEQ the sequence accuracy, and POS X, the accuracy of the X'th question in a sequence.

As in previous work, we also report performance for a non-contextual setup where follow up questions are answered in isolation. We observe that our model effectively leverages the context information by improving the average question accuracy from 45.1% to 55.1% in comparison to the use of context in DynSp yielding 2.7% improvement. If we provide the previous reference answers, the average question accuracy jumps to 61.7%, showing that 6.6% of the errors are due to error propagation.

**Numeric operations** For understanding how effective our model is in handling numeric operations, we trained models without the specific handling explained in Section 2. We find that that the overall accuracy decreases from 55.1% to 51.5%, demonstrating the competence of our approach to model such operations. This effectiveness is further emphasized when focusing on questions that contain a superlative (e.g., *"tallest"*, *"most expensive"*) with a performance difference of 47.3% with numeric relations and 40.3% without. It is worthwhile to call out that the model without special number handling still out-performs the previous SOTA Camp by more than 5 points (45.6% vs 55.1%).

**Table size.** We observe that our model is not sensitive to table size changes, with an average accuracy of 52.4% for the 10% largest tables (vs.

**What are all the nations?**
*Australia, Italy, Germany, Soviet Union, Switzerland, United States, Great Britain, France*

**Which won gold medals?**
*Australia, Italy, Germany, Soviet Union*

**Which won more than one?**
*Australia*

| Rank | Nation | Gold | Silver | Bronze | Total |
|---|---|---|---|---|---|
| 1 | Australia | 2 | 1 | 0 | 3 |
| 2 | Italy | 1 | 1 | 1 | 3 |
| 3 | Germany | 1 | 0 | 1 | 2 |
| 4 | Soviet Union | 1 | 0 | 0 | 1 |
| 5 | Switzerland | 0 | 2 | 1 | 3 |
| 6 | United States | 0 | 1 | 0 | 1 |
| 7 | Great Britain | 0 | 0 | 1 | 1 |
| 7 | France | 0 | 0 | 1 | 1 |

Table 2: A sequence of questions (left) and the corresponding table (right) selected from the SQA dataset that is answered correctly by our approach. This example requires handling conversational context and numerical comparisons.

55.1% overall).[4]

**Error analysis.** Table 2 shows an example that is consistently handled correctly by the model. It requires a simple string match (*"nations"* → *"nation"*), and implicit and explicit comparisons.

We performed error analysis on test data over 100 initial (POS1) and 100 follow up questions (POS$>$ 1) to identify the limitations of our approach.

For the initial questions, we find that 26% are match errors, e.g., the model does not match *"episode"* to *"Eps #"*, or cases where the model has to exclude rows with empty values from the results. 29% of the errors require a more sophisticated table understanding, e.g., rows that represent the total of all other rows should often not be included in the answers. For 15% of the errors, we think that the reference answer is incorrect and for another 15% the model prediction is correct but contains duplicates because multiple rows contain the same value. 12% of the errors are around complex matches such as selecting certain ranks (*"the first two"*), exclusion or negation.

For the follow up questions, 38% are caused by complex matches; 17% are match errors; 13% of the errors are due to incorrect reference answers and 11% would require advanced table understanding. Only 8% of the errors are due to incorrect management of the conversational context. Section B of the appendix contains a more detailed analysis and error examples.

## 6 Discussion

We present a model for table-centered conversational QA that predicts the answers directly from the table. We show that this model improves SOTA on SQA dataset and particularly handles conversational context effectively.

As future work, we plan to expand our model with pre-trained language representations (e.g., BERT (Devlin et al., 2018)) in order to improve performance on initial queries and matching of queries to table entries. To handle larger tables, we will investigate sharding the table row-wise, running the model on all the shards first, and then on the final table which combines all the answer rows.

[4]Fig. 3 of the appendix shows a scatter plot of table size vs. accuracy.

## References


Rishabh Agarwal, Chen Liang, Dale Schuurmans, and Mohammad Norouzi. 2019. Learning to generalize from sparse and underspecified rewards. *arXiv preprint arXiv:1902.07198*.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wentau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. QuAC: Question answering in context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2174–2184, Brussels, Belgium. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 731–742, Melbourne, Australia. Association for Computational Linguistics.

Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Elliot Karro, and D. Sculley, editors. 2017. *Google Vizier: A Service for Black-Box Optimization*.

Daya Guo, Duyu Tang, Nan Duan, Ming Zhou, and Jian Yin. 2018. Dialog-to-action: Conversational question answering over a large-scale knowledge base. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 2946–2955, USA. Curran Associates Inc.

Izzeddin Gur, Semih Yavuz, Yu Su, and Xifeng Yan. 2018. DialSQL: Dialogue based structured query generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1339–1349, Melbourne, Australia. Association for Computational Linguistics.

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1821–1831, Vancouver, Canada. Association for Computational Linguistics.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22, Berlin, Germany. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1516–1526, Copenhagen, Denmark. Association for Computational Linguistics.

Chen Liang, Jonathan Berant, Quoc V. Le, Kenneth D. Forbus, and Ni Lao. 2016. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. *CoRR*, abs/1611.00020.

Chen Liang, Mohammad Norouzi, Jonathan Berant, Quoc V Le, and Ni Lao. 2018. Memory augmented policy optimization for program synthesis and semantic parsing. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 10015–10027. Curran Associates, Inc.

Wang Ling, Phil Blunsom, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, Fumin Wang, and Andrew Senior. 2016. Latent predictor networks for code generation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 599–609, Berlin, Germany. Association for Computational Linguistics.

Reginald Long, Panupong Pasupat, and Percy Liang. 2016. Simpler context-dependent logical forms via model projections. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1456–1465, Berlin, Germany. Association for Computational Linguistics.

Arvind Neelakantan, Quoc V. Le, Martín Abadi, Andrew McCallum, and Dario Amodei. 2016a. Learning a natural language interface with neural programmer. *CoRR*, abs/1611.08945.

Arvind Neelakantan, Quoc V. Le, Martín Abadi, Andrew McCallum, and Dario Amodei. 2016b. Learning a natural language interface with neural programmer. *CoRR*, abs/1611.08945.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics.

Haoruo Peng, Ming-Wei Chang, and Wen-tau Yih. 2017. Maximum margin reward networks for learning from explicit and implicit supervision. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2368–2378, Copenhagen, Denmark. Association for Computational Linguistics.

Bojan Petrovski, Ignacio Aguado, Andreea Hossmann, Michael Baeriswyl, and Claudiu Musat. 2018. Embedding individual table columns for resilient SQL chatbots. In *Proceedings of the 2018 EMNLP Workshop SCAI: The 2nd International Workshop on Search-Oriented Conversational AI*, pages 67–73, Brussels, Belgium. Association for Computational Linguistics.

Amrita Saha, Vardaan Pahuja, Mitesh M. Khapra, Karthik Sankaranarayanan, and Sarath Chandar. 2018. Complex sequential question answering: Towards learning to converse over linked question answer pairs with a knowledge graph. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 705–713.

Peter Shaw, Philip Massey, Angelica Chen, Francesco Piccinno, and Yasemin Altun. 2019. Generating logical forms from graph representations of text and entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 95–106, Florence, Italy. Association for Computational Linguistics.

Alane Suhr, Srinivasan Iyer, and Yoav Artzi. 2018. Learning to map context-dependent sentences to executable formal queries. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2238–2249, New Orleans, Louisiana. Association for Computational Linguistics.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen. 2018a. Open domain question answering using early fusion of knowledge bases and text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4231–4242, Brussels, Belgium. Association for Computational Linguistics.

Yibo Sun, Duyu Tang, Nan Duan, Jingjing Xu, Xiaocheng Feng, and Bing Qin. 2018b. Knowledge-aware conversational semantic parsing over web tables. *CoRR*, abs/1809.04271.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2692–2700. Curran Associates, Inc.

Chenglong Wang, Po-Sen Huang, Alex Polozov, Marc Brockschmidt, and Rishabh Singh . 2018. Execution-guided neural program decoding. In *ICML Neural Abstract Machines and Program Induction workshop, 2018*.

Xiaojun Xu, Chang Liu, and Dawn Song. 2017. Sqlnet: Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv:1711.04436*.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1321–1331, Beijing, China. Association for Computational Linguistics.

Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. 2018. TypeSQL: Knowledge-based type-aware neural text-to-SQL generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 588–594, New Orleans, Louisiana. Association for Computational Linguistics.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.

## A Hyperparameters

We tune hyperparameters with Google Vizier (Golovin et al., 2017). For the encoder and decoder, we select the number of layers from $[3, 6]$ and embedding and hidden dimensions from $\{128, 256, 512\}$, setting the feed forward layer hidden dimensions $4\times$ higher. We employ dropout at training time with $P_{dropout}$ selected from $\{0.2, 0.4, 0.5\}$. We select the attention heads from $\{4, 8, 16\}$ and use a clipping distance of $6$ for relative position representations.

We use the Adam optimizer (Kingma and Ba, 2014) with $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\epsilon = 10^{-9}$. We tune the learning rate and use the same warm-up and decay strategy for learning rate as Vaswani et al. (2017), selecting a number of warm-up steps up to a maximum of 2000. We run the training until convergence for a fixed number of steps $(100,000)$ and use the final checkpoint for evaluation. We choose batch sizes from $\{32, 64\}$.

## B Results

**Table Size** Given that our model makes use of the whole table, it is conceivable that the performance of our approach can be more sensitive to the table size than methods that predict intermediate representations. Plotting the model performance with respect to number of cells in the table (Figure 3), we observe that the performance does not vary significantly by the table size.

**Error Analysis** For a detailed analysis, we annotated 100 initial and follow-up questions with the following match types:

**MATCH** A lexical or semantic match error such as not matching *"episode"* with *"EPS #"*.

**TABLE_UNDERSTANDING** A question that would require a higher level table understanding to be answered correctly. For example, we often have to decide that a row is just the total of all other rows and should not be selected.

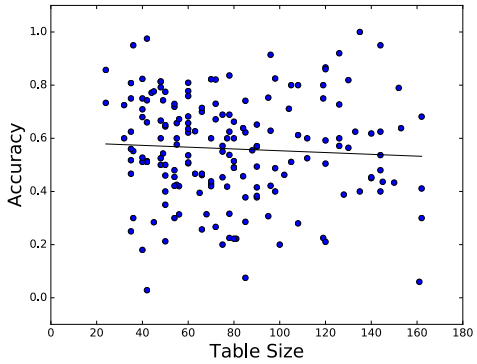


Figure 3: Scatter-plot of accuracy and table size. Each point represents the accuracy on all the questions asked about a test set table of the corresponding size.

| Error type | Counts |
|---|---|
| TABLE_UNDERSTANDING | 29 |
| COMPLEX_MATCH | 12 |
| MATCH | 26 |
| GOLD | 15 |
| ANSWER_SET | 15 |
| OTHER | 3 |

(a)

| Error type | Counts |
|---|---|
| TABLE_UNDERSTANDING | 11 |
| COMPLEX_MATCH | 38 |
| MATCH | 17 |
| GOLD | 13 |
| ANSWER_SET | 4 |
| OTHER | 9 |
| CONTEXT | 8 |

(b)

Table 3: Errors on 100 random initial (a) and follow-up (b) questions

**COMPLEX_MATCH** A question that would require a numerical value match, some sort of sorting or a negation to be answered. *"what is the area of all of the non-remainders?"*

**GOLD** A question with a wrong reference answers. *"what gene functions are listed?"* – Gold points to Category column rather than Gene Functions.

**ANSWER_SET** The returned answer should be duplicate free. *"what are all of the rider's names?"*, but the table contains *"Carl Fogarty"* multiple times.

**CONTEXT** Only used in follow-up questions. This error indicates that a more sophisticated context management is needed.

**OTHER** Any other kind of error.

Table 3 contains the error counts for initial questions and follow-ups, respectively. Table 4 contains interesting examples.

| Question | Notes |
|---|---|
| COMPLEX_MATCH | |
| *who scored more than earnie stewart?* | 2-hop reasoning. Requires comparison on top of the result of the inner question. |
| *and which has been active for the longest?* | Reasoning with text and date (1986-present) |
| *who else is in that field?* | Exclusion. |
| *of these, which did not publish on february 9?* | Negation. The model is doing the right thing but missing one of the values. |
| *what is the highest passengers for a route?* | The model selects the 2nd highest and not the 1st. |
| TABLE_UNDERSTANDING | |
| *now, during which year did they have the worst placement?* | Requires understanding that *"Withdrawal .."* is worse than any position. |
| *which of these seasons have four judges?* | Requires counting the number of named entities within a single cell. |
| GOLD | |
| *what gene functions are listed?* | Gold points to *"Category"* column rather than *"Gene Functions"* |
| *which aircraft have 1 year of service* | Gold points to the 4th column (*"in service"*) instead of the 1st column (*"aircraft"*) |
| CONTEXT | |
| *when was thaddeus bell born?, when was klaus jurgen schneider born?, which is older?* | The correct birthday is selected, but not the person. |

Table 4: Some interesting error cases with comments.

# Improving Answer Selection and Answer Triggering using Hard Negatives

**Sawan Kumar** *
Indian Institute of Science, Bangalore
sawankumar@iisc.ac.in

**Shweta Garg** †
Search Customer Experience, Amazon
shwegarg@amazon.com

**Kartik Mehta**
India Machine Learning, Amazon
kartim@amazon.com

**Nikhil Rasiwasia**
India Machine Learning, Amazon
rasiwasi@amazon.com

## Abstract

In this paper, we establish the effectiveness of using hard negatives, coupled with a siamese network and a suitable loss function, for the tasks of answer selection and answer triggering. We show that the choice of sampling strategy is key for achieving improved performance on these tasks. Evaluating on recent answer selection datasets - InsuranceQA, SelQA, and an internal QA dataset, we show that using hard negatives with relatively simple model architectures (bag of words and LSTM-CNN) drives significant performance gains. On InsuranceQA, this strategy alone improves over previously reported results by a minimum of 1.6 points in P@1. Using hard negatives with a Transformer encoder provides a further improvement of 2.3 points. Further, we propose to use quadruplet loss for answer triggering, with the aim of producing globally meaningful similarity scores. We show that quadruplet loss function coupled with the selection of hard negatives enables bag-of-words models to improve F1 score by 2.3 points over previous baselines, on SelQA answer triggering dataset. Our results provide key insights into answer selection and answer triggering tasks.

## 1 Introduction

Question answering (QA) is an active field of research, drawing attention from the natural language processing (NLP) and information retrieval (IR) community. Selection-based QA is the task of selecting an answer for a given question, from a set of candidate answers. Two tasks have been proposed for selection-based QA. Given a question and an answer candidate pool, **answer selection** is the task of ranking valid answers higher than irrelevant answers, where it is assumed that there is at least one valid answer in the candidate pool. **Answer triggering**, defined for cases where the candidate pool may or may not have a valid answer, is the task of finding a valid answer, while being allowed to abstain.

Recently released datasets (Feng et al., 2015; Yang et al., 2015; Jurczyk et al., 2016) have led to significant development in end-to-end neural network Selection-based QA. These networks are learned by posing answer selection as either a ranking problem or a classification problem. When posed as a ranking problem, a common choice is to use a triplet loss defined over a question, a correct answer and a (usually sampled) negative answer. Triplet loss penalises relative distances between positive and negative pairs, making the choice of negatives critical. Recent progress in image understanding tasks suggests that learning in ranking tasks can be enhanced by focusing on sampling strategies in conjunction with the loss function employed (Hermans et al., 2017; Chen et al., 2017). In this work, we investigate the expressive power of siamese architectures (Bromley et al., 1994) coupled with hard negatives (i.e., difficult negative samples for the model being learnt), against more sophisticated models. Specifically, all our models ignore any interaction between question and answer representations, and use a shared encoder to independently encode questions and answers. Siamese architectures provide low-latency solutions in the presence of large candidate answer pools (by caching candidate representations).

In this paper, we first show that using hard negatives with vanilla neural architectures improves over previously reported results on InsuranceQA answer selection task (Feng et al., 2015). The proposed strategy achieves P@1 of 73.3%, as compared to the previously reported 71.1% (dos Santos et al., 2016). Further, using a Transformer encoder coupled with the proposed strategy achieves

*Work done during an internship at Amazon
†Work done at India Machine Learning, Amazon



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5911–5917,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

P@1 of 75.6%. Next, we propose the use of quadruplet loss (Chen et al., 2017) for answer triggering. We show that by employing online selection of hard negatives with quadruplet loss, bag-of-words models improve upon previous baselines on SelQA answer triggering task (Jurczyk et al., 2016). The proposed strategy achieves an F1 score of 53.21, as compared to the previously reported 50.89 (Jurczyk et al., 2016).

We note that the performance gains demonstrated in this paper are obtained with a restricted class of models. Specifically, we restrict to siamese networks, and don't model any interaction between question and answer representations. The previously reported results which we compare against use architectures that are not restricted to be siamese, and model the interaction between question and answer representations.

## 2 Related Work

For answer selection, CNN, LSTM and LSTM-CNN architectures (Tan et al., 2015; dos Santos et al., 2016), trained with a triplet loss, have been explored. Recent focus has been on designing the interaction layer, between a question and a candidate answer, through various models of attention (Yu et al., 2014; Rao et al., 2016; Zhou et al., 2015; Zhang et al., 2017; Bian et al., 2017). Compare-aggregate architectures have also been studied as sentence matching models and applied to answer selection (Wang and Jiang, 2016; Wang et al., 2017). Similarly, for answer triggering, different architectural changes have been proposed (Zhao et al., 2017; Acheampong et al., 2016; Gupta et al., 2018; Li and Wu, 2017). Often, the architectural advances are coupled with subtle changes in the training process. For example, dos Santos et al. (2016) incorporate a form of negative mining. This makes it difficult to separate the benefits gained from the mining strategy from the architectural advances.

Triplet loss and mining of hard negatives have been studied for computer vision tasks with triplet networks being popular for estimating feature embeddings for images. The key issue with triplet networks is that, for a training set with $N$ samples, the number of triplets is cubic in $N$. Training becomes intractable for modest sizes of the training set. A solution in the form of importance sampling has been studied. Schroff et al. (2015) learn image embeddings using triplet loss, trained using moderately hard negatives. While other approaches have been studied for the task of person re-identification, which combine classification with a verification loss, Hermans et al. (2017) show that a vanilla CNN with triplet loss and the right sampling strategy could outperform the best models at the time. Wu et al. (2017) present a distance-weighed sampling approach.

Triplet loss, while useful for ranking tasks, doesn't produce globally meaningful scores for tasks such as person re-identification. This can be attributed to the fact that triplet loss doesn't try to learn a global threshold to separate all inter-class pairs from all intra-class pairs, and learns only relative distances with respect to an anchor (a question, in the context of this paper). This has been addressed by accounting for the global structure of the embedding space (Kumar et al., 2016; Ustinova and Lempitsky, 2016), and directly optimizing for inter-class distances to be larger than intra-class distances. Quadruplet loss (Chen et al., 2017) prvoides another way to produce globally meaningful scores, and is suitable for our setting. Quadruplet loss extends triplet loss to ensure smaller intra-class distances and larger inter-class distances. This is achieved by additionally penalizing positive and negative pairs with different probes (questions, in the context of this paper).

While selection of hard negatives has been used for problems in computer vision, its usefulness has not been evaluated for QA tasks. In this work, we first present the selection of hard negatives with triplet loss for answer selection. Next, we show that quadruplet loss is suitable for the task of answer triggering, and when coupled with online selection of hard negatives, improves over previous baselines. As far as we know, we are the first to use quadruplet loss for question answering.

## 3 Data

| Dataset | #Q | #QA | $\mathbf{L_q}$ | $\mathbf{L_a}$ |
|---|---|---|---|---|
| InsuranceQA | 12887 | 18540 | 7.15 | 95.54 |
| SelQA | 5524 | 6390 | 11.02 | 25.39 |
| LargeQA | 341869 | 502763 | 14.06 | 30.34 |

Table 1: Comparison of answer selection datasets. **#Q**: Number of questions; **#QA**: Number of correct question-answer pairs; $\mathbf{L_q}/\mathbf{L_a}$: Average number of tokens in a question/answer.

Insurance QA (Feng et al., 2015), a domain-specific non-factoid QA dataset, is suitable for

evaluating answer selection. SelQA (Jurczyk et al., 2016), a recent open-domain factoid QA dataset provides data for evaluating both answer selection and answer triggering.

While these datasets provide standardised comparison, we also evaluate our methods on a large internal answer selection dataset, LargeQA, created using Community Question-Answers (CQnA) asked on a website. The dataset was created similar to InsuranceQA, with the size of candidate pool fixed to 100 answers in test sets.

Table 1 presents some statistics on the training data used in these datasets.

We don't include WikiQA (Yang et al., 2015) and TrecQA (Wang et al., 2007) datasets which have been previously used to report improvements in answer selection. This is due to the large variance in our experiments as well as in the reproduction of existing methods, also noted by Crane (2018), perhaps due to their smaller sizes.

## 4 Method

For selection-based QA , the training data $X$, can be characterized as a list of questions $Q = \{q_1, q_2, \ldots, q_s\}$, and sets of correct answer(s) $A_q = \{a_1, a_2, \ldots, a_p\}$ for each question $q \in Q$.

In the following, we discuss online selection of hard negatives with triplet loss for answer selection, and with quadruplet loss for answer triggering. In each case, a siamese architecture is used where questions and answers are encoded independently by identical copies of a neural network denoted by $f$. Cosine similarity is used to compute the similarity between a question and an answer: $S(x, y) = \text{cosine}(f(x), f(y))$.

### 4.1 Answer Selection

Siamese networks can be trained for answer selection using triplets $(q, a^+, a^-)$, where $a^+ \in A_q$ is a correct answer and $a^- \notin A_q$ an incorrect answer chosen randomly from the entire answer pool. A triplet loss with margin $m$ is formulated as:

$$L_{\text{triplet}} = \max(0, m - S(q, a^+) + S(q, a^-)). \quad (1)$$

In this work, we employ online selection of hard negatives within a batch of sampled question-answer pairs. The model is trained using batch gradient descent, with batches $B = \{(q_i, a_i, a_i^H) : 1 \leq i \leq b\}$, where $b$ is the batch size, $a_i \in A_{q_i}$ and $a_i^H$ is the hardest negative answer chosen as:

$$a_i^H = \underset{j: a_j \notin A_{q_i},\ 1 \leq j \leq b}{\text{argmax}} (S(q_i, a_j)). \quad (2)$$

Selection of negatives from within a batch of sampled question-answer pairs has several advantages. First, it does not require any extra computation from the embedding model. Second, there is no need to employ additional heuristics to avoid the hardest negatives. Selection from within a stochastic batch ensures that the selected hardest negatives in a batch are not dominated by false negatives or noisy hard negatives during training (See Appendix C for experimental results).

### 4.2 Answer Triggering

Answer triggering differs from answer selection in that we need to identify if a valid answer is present in the candidate pool. Triplet loss is not suitable as it promotes scores to be discriminative only relative to a given question. While classification based methods have been studied, we hypothesize that globally meaningful scores can be obtained in a ranking setup. Essentially, we need the similarity scores for all wrongly paired question-answers to be smaller than the scores for correct question-answer pairs. We propose to use quadruplet loss for answer triggering. Quadruplet loss, introduced by Chen et al. (2017) for Person re-identification, aims at keeping all inter-class distances larger than all intra-class distances. This, we believe, is what we need to obtain globally meaningful scores for answer triggering.

The network can be trained using quadruplets of the form $(q, a^+, a^-, q')$, where $a^+ \in A_q$, $a^- \notin A_q$, and $q'$ is a negative question for both $a^+$ and $a^-$ chosen randomly from the entire question pool, with $a^+ \notin A_{q'}$, $a^- \notin A_{q'}$. Quadruplet loss for answer triggering can be formulated as:

$$L_{\text{quad}} = \max(0, m_1 - S(q, a^+) + S(q, a^-)) + \max(0, m_2 - S(q, a^+) + S(q', a^-)) \quad (3)$$

where $m_1$ and $m_2$ are fixed margins.

As with triplet loss, we believe the selection of negatives is critical for learning with quadruplet loss. We propose to select both the negative answer and the negative question using online selection of hard negatives. The model is trained using batch gradient descent with batches $B = \{(q_i, a_i, a_i^H, q_i^H) : 1 \leq i \leq b\}$, where $b$ is the batch size, $a_i \in A_{q_i}$, $a_i^H$ is chosen as in Equation 2, and $q_i^H$ is selected to be the hardest negative question for $a_i^H$:

$$q_i^H = \underset{k: a_i^H \notin A_{q_k},\ 1 \leq k \leq b}{\text{argmax}} (S(q_k, a_i^H)). \quad (4)$$

| Method | P@1 | | |
|---|---|---|---|
| | Dev | Test1 | Test2 |
| Attentive Pooling + CNN (dos Santos et al., 2016) | **68.8** | 69.8 | 66.3 |
| Attentive Pooling + BiLSTM (dos Santos et al., 2016) | 68.4 | **71.7** | **66.4** |
| Max-Pooling + Random Negatives | 61.5 | 61.6 | 57.9 |
| Max-Pooling + Hard Negatives | 64.4 | 65.3 | 64.4 |
| LSTM-CNN + Random Negatives | 68.6 | 70.3 | 67.2 |
| LSTM-CNN + Hard Negatives | **72.5** | **73.3** | **69.1** |
| Transformer encoder + Random Negatives | 65.8 | 67.1 | 63.8 |
| Transformer encoder + Hard Negatives | **75.7** | **75.6** | **73.4** |

Table 2: *Answer selection on InsuranceQA*. **Top**: Baselines. **Middle** & **Bottom**: Our Max-Pooling, LSTM-CNN and Transformer encoder results. When coupled with hard negatives, simpler architectures such as Max-Pooling become competitive, while LSTM-CNN improves over previously reported results. The effectiveness of using hard negatives, compared to random negatives, is also evident when training the Transformer encoder model.

During inference, for a given question, we predict the highest scoring answer, provided the score exceeds the threshold. The optimal threshold is obtained using performance on the development set.

## 5 Results and Analysis

We present quantitative results using MRR and P@1 for answer selection on InsuranceQA, LargeQA and SelQA, and F1 score (Yang et al., 2015) for answer triggering on SelQA, followed by an ablation study of the proposed method for answer triggering.

### 5.1 Experimental Setup

Following Tan et al. (2015), we experiment with an LSTM-CNN model, where a CNN layer is employed on top of LSTM encoding of the input sentence, followed by Max Pooling to obtain a fixed length representation for the sentence.

Motivated by the performance of bag-of-words like sentence representations (Arora et al., 2016), we also experiment with distributional bag-of-words models where sentence embeddings are obtained by pooling across the dimensions of corresponding word embeddings. In particular, we used Max-Pooling and Max-Min-Pooling, the latter being obtained by concatenating the outputs of Max and Min Pooling.

Finally, we experiment with the Transformer encoder (Vaswani et al., 2017; Devlin et al., 2019) for answer selection.

Training details are available in Appendix A.

### 5.2 Answer Selection

**InsuranceQA**: LSTM-CNN model coupled with hard negatives outperforms the best reported numbers on both test sets by a significant margin (1.6 and 2.7 points gain in P@1 with p-values of 0.066 and 0.008 on Test1 and Test2 respectively) on InsuranceQA (Table 2). Finally, the Transformer encoder model with hard negatives provides significant performance gains for the task (3.9 and 7.0 points gain in P@1), while still learning with siamese networks without any interaction between question and answer representations.

| Method | P@1 | MRR |
|---|---|---|
| LSTM-CNN + Random Negatives | +0 | +0 |
| LSTM-CNN + Hard Negatives | **+2.5** | **+0.015** |

Table 3: *Answer selection on LargeQA*; reported numbers are relative to the baseline.

**LargeQA**: The gains due to hard negatives are also observed on LargeQA (Table 3), with an improvement of 2.5 in P@1 and 0.015 in MRR.

| Model | MRR |
|---|---|
| RNN + Attention Pooling (Jurczyk et al., 2016) | 0.876 |
| CNN-Hinge (dos Santos et al., 2017) | 0.881 |
| ACNN (Shen et al., 2018) | 0.880 |
| AdaQA (Shen et al., 2018) | 0.910 |
| DRCN (Kim et al., 2019) | **0.930** |
| Max-Pooling + Random Negatives | 0.843 |
| Max-Pooling + Hard Negatives | 0.896 |

Table 4: *Answer selection on SelQA*. **Top**: Baselines. **Bottom**: Max-Pooling results from this work.

**SelQA**: For answer selection on SelQA (See Table 4), Max-Pooling models performed best among the models we investigated, perhaps due to the smaller size of the dataset (other results omitted).

**Key Observations:** First, siamese bag-of-words models, coupled with hard negatives, are competitive with models which rely on various ways of question-answer interaction. Second, siamese networks coupled with hard negatives are particularly suited for domain specific QA (In-

suranceQA). We believe open-domain QA has a greater need for interaction between question and answer representations. Third, we believe employing hard negatives is especially useful for the Transformer encoder model, which has a much larger set of parameters. With the Transformer encoder model, there is an improvement of 8.5 points on InsuranceQA Test1 dataset when using hard negatives as opposed to using random negatives. With the LSTM-CNN model, the corresponding gain is 3.0 points.

### 5.3 Answer Triggering

| Method | F1 | |
|---|---|---|
| | Dev | Test |
| CNN (Jurczyk et al., 2016) | 49.16 | 50.89 |
| RNN + AP (Jurczyk et al., 2016) | 44.02 | 45.67 |
| Max-Min-Pooling + Quadruplet loss + Hard negatives | **54.95** | **53.21** |

Table 5: *Answer triggering on SelQA.* **Top**: Baselines. **Bottom**: Our approach using quadruplet loss with Max-Min-Pooling model. The choice of the loss function and the sampling strategy proposed in this paper is justified by the improved results obtained using a relatively simpler Max-Min-Pooling model.

The proposed approach with the Max-Min-Pooling model for answer triggering improves upon the previously reported baselines significantly (2.32 points) on SelQA (See Table 5).

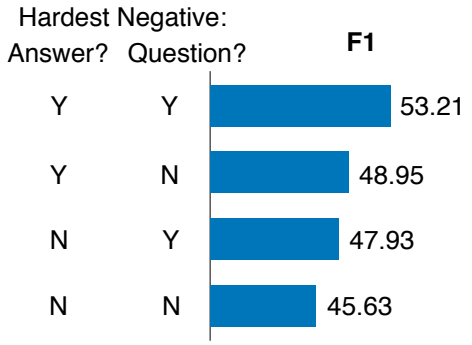


Figure 1: *Ablation study for answer triggering.* The columns on the left indicate whether online selection of hardest negative answer/question was done. To obtain best results with quadruplet loss, it is critical to sample both hard negative answers and hard negative questions.

An ablation study (Figure 1) reveals that the selection of both hardest negative answers and questions is needed to get the majority improvement. Quadruplet loss with the Max-Min-Pooling model without hard negatives is competitive with baselines (Table 5), justifying its usage for answer triggering.

Further, we also compared against using triplet loss with the Max-Min-Pooling model for answer triggering. We found that quadruplet loss outperforms triplet loss (F1 of 51.89 on development set, and 50.08 on Test set) by 3.13 points in F1.

## 6 Conclusion & Future Work

We have shown that selection of hard negatives is a powerful tool for answer selection. We improve over previously reported results on recent benchmarks using siamese architectures and hard negatives, outperforming interaction-based models. We show the generality of the approach using shallow as well as deep neural network models.

For answer triggering, we presented results supporting the hypothesis that quadruplet loss with hard negatives is suitable for the task, and improves upon previous baselines. Our ablation study confirms the importance of using hard negatives.

As future work, we plan to further investigate the generality of the approach with other tasks and base models.

## Acknowledgments

We would like to thank the reviewers for their valuable feedback.

## References

Kingsley N Acheampong, Zhen-Hao Pan, Er-Qiang Zhou, and Xiao-Yu Li. 2016. Answer triggering of factoid questions: A cognitive approach. In *Wavelet Active Media Technology and Information Processing (ICCWAMTIP), 2016 13th International Computer Conference on*, pages 33–37. IEEE.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2016. A simple but tough-to-beat baseline for sentence embeddings.

Weijie Bian, Si Li, Zhao Yang, Guang Chen, and Zhiqing Lin. 2017. A compare-aggregate model with dynamic-clip attention for answer selection. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1987–1990. ACM.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1994. Signature verification using a" siamese" time delay neural network. In *Advances in neural information processing systems*, pages 737–744.

Weihua Chen, Xiaotang Chen, Jianguo Zhang, and Kaiqi Huang. 2017. Beyond triplet loss: a deep quadruplet network for person re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 403–412.

Matt Crane. 2018. Questionable answers in question answering research: Reproducibility and variability of published results. *Transactions of the Association for Computational Linguistics*, 6:241–252.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Minwei Feng, Bing Xiang, Michael R Glass, Lidan Wang, and Bowen Zhou. 2015. Applying deep learning to answer selection: A study and an open task. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 813–820. IEEE.

Deepak Gupta, Sarah Kohail, and Pushpak Bhattacharyya. 2018. Combining graph-based dependency features with convolutional neural network for answer triggering. *CoRR*, abs/1808.01650.

Alexander Hermans, Lucas Beyer, and Bastian Leibe. 2017. In defense of the triplet loss for person re-identification. *CoRR*, abs/1703.07737.

Tomasz Jurczyk, Michael Zhai, and Jinho D Choi. 2016. Selqa: A new benchmark for selection-based question answering. In *2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 820–827. IEEE.

Seonhoon Kim, Inho Kang, and Nojun Kwak. 2019. Semantic sentence matching with densely-connected recurrent and co-attentive information. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6586–6593.

BG Kumar, Gustavo Carneiro, Ian Reid, et al. 2016. Learning local image descriptors with deep siamese and triplet convolutional networks by minimising global loss functions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5385–5394.

Wei Li and Yunfang Wu. 2017. Hierarchical gated recurrent neural tensor network for answer triggering. In *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*, pages 287–294. Springer.

Jinfeng Rao, Hua He, and Jimmy Lin. 2016. Noise-contrastive estimation for answer selection with deep neural networks. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 1913–1916. ACM.

Cícero Nogueira dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *CoRR*, abs/1602.03609.

Cícero Nogueira dos Santos, Kahini Wadhawan, and Bowen Zhou. 2017. Learning loss functions for semi-supervised learning via discriminative adversarial networks. *CoRR*, abs/1707.02198.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823.

Dinghan Shen, Martin Renqiang Min, Yitong Li, and Lawrence Carin. 2018. Learning context-sensitive convolutional filters for text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1839–1848, Brussels, Belgium. Association for Computational Linguistics.

Ming Tan, Bing Xiang, and Bowen Zhou. 2015. Lstm-based deep learning models for non-factoid answer selection. *CoRR*, abs/1511.04108.

Evgeniya Ustinova and Victor Lempitsky. 2016. Learning deep embeddings with histogram loss. In *Advances in Neural Information Processing Systems*, pages 4170–4178.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. 2007. What is the Jeopardy model? a quasi-synchronous grammar for QA. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 22–32, Prague, Czech Republic. Association for Computational Linguistics.

Shuohang Wang and Jing Jiang. 2016. A compare-aggregate model for matching text sequences. *CoRR*, abs/1611.01747.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 4144–4150. AAAI Press.

Chao-Yuan Wu, R Manmatha, Alexander J Smola, and Philipp Krähenbühl. 2017. Sampling matters in deep embedding learning. In *Proc. IEEE International Conference on Computer Vision (ICCV)*.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. WikiQA: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018, Lisbon, Portugal. Association for Computational Linguistics.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. *CoRR*, abs/1412.1632.

Xiaodong Zhang, Sujian Li, Lei Sha, and Houfeng Wang. 2017. Attentive interactive neural networks for answer selection in community question answering. In *AAAI*, pages 3525–3531.

Jie Zhao, Yu Su, Ziyu Guan, and Huan Sun. 2017. An end-to-end deep framework for answer triggering with a novel group-level objective. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1276–1282, Copenhagen, Denmark. Association for Computational Linguistics.

Xiaoqiang Zhou, Baotian Hu, Qingcai Chen, Buzhou Tang, and Xiaolong Wang. 2015. Answer sequence learning with neural networks for answer selection in community question answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 713–718, Beijing, China. Association for Computational Linguistics.

# Can You Unpack That?
# Learning to Rewrite Questions-in-Context

**Ahmed Elgohary, Denis Peskov, Jordan Boyd-Graber***
Department of Computer Science, UMIACS, iSchool, Language Science Center
University of Maryland, College Park
{elgohary, dpeskov, jbg}@cs.umd.edu

## Abstract

Question answering is an AI-complete problem, but existing datasets lack key elements of language understanding such as coreference and ellipsis resolution. We consider sequential question answering: multiple questions are asked one-by-one in a conversation between a questioner and an answerer. Answering these questions is only possible through understanding the conversation history. We introduce the task of question-in-context rewriting: given the context of a conversation's history, rewrite a context-dependent into a self-contained question with the same answer. We construct, CANARD, a dataset of 40,527 questions based on QuAC (Choi et al., 2018) and train Seq2Seq models for incorporating context into standalone questions.

## 1 Introduction

Question Answering (QA) is an AI complete problem (Webber, 1992), but existing QA datasets do not rise to the challenge: they lack key NLP problems like anaphora resolution, coreference disambiguation, and ellipsis resolution. The logic needed to answer these types of questions requires deeper NLP understanding that simulates the context in which humans naturally answer questions.

Neural techniques question answering have improved (Devlin et al., 2018) machine reading comprehension (Rajpurkar et al., 2016, MRC): computers can take a single question and extract answers from datasets like Wikipedia. However, QA models struggle to generalize when questions do not look like the standalone questions systems in training data: e.g., new genres, languages, or closely-related tasks (Yogatama et al., 2019).

Conversational question answering (Reddy et al., 2019, CQA) is a generalization that ask *multiple* questions in an information-seeking dialogs. Unlike MRC, CQA requires models to link questions together to resolve the conversational dependencies between them: each question needs to be understood in the conversation context. For example, the question *"What was he like in that episode?"* cannot be understood without knowing what *"he"* and *"that episode"* refer to, which can be resolved using the conversation context.

We reduce challenging, interconnected CQA examples to independent, stand-alone MRC to create CANARD—**C**ontext **A**bstraction: **N**ecessary **A**dditional **R**ewritten **D**iscourse—a new dataset[1] that rewrites QuAC (Choi et al., 2018) questions. We crowdsource context-independent paraphrases of QuAC questions and use the paraphrases to train and evaluate question-in-context rewriting.

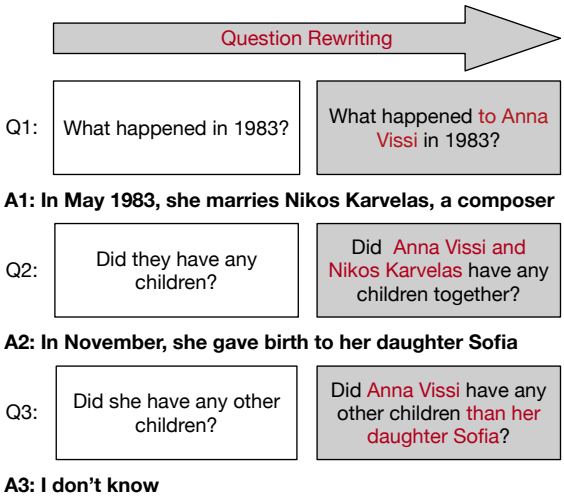


Figure 1: Question-in-context rewriting task. The input to each step is a question to rewrite given the dialog history which consists of the dialog utterances (questions and answers) produced before the given question is asked. The output is an equivalent, context-independent paraphrase of the input question.

* Now at Google AI Zürich

[1] http://canard.qanta.org



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5918–5924,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| Characteristic | Ratio |
|---|---|
| Answer Not Referenced | 0.98 |
| Question Meaning Unchanged | 0.95 |
| Correct Coreferences | 1.0 |
| Grammatical English | 1.0 |
| Understandable w/o Context | 0.90 |

Table 1: Manual inspection of 50 rewritten context-independent questions from CANARD suggests that the new questions have enough context to be independently understandable.

Section 2 formally defines the task of question de-contextualization. Section 3 constructs CANARD, a new dataset of question-in-context with corresponding context-independent paraphrases. Section 5 analyzes our rewrites (and the underlying methodology) to understand the linguistic phenomena that make CQA difficult. We build several baseline rewriting models and compare their BLEU scores to our human rewrites in Section 4.

## 2 Defining Question-In-Context Rewrites

We formally define the task of question-in-context rewriting (de-contextualization). Given a conversation topic $t$ and a history $H$ of $m-1$ turns, each turn $k$ is a question $q_i$ and an answer $a_i$; the task is to generate a rewrite $q'_m$ for the next question $q_m$ based on $H$. Since $q_m$ is part of the conversation, its meaning often involves references to parts of its preceding history. A valid rewrite $q'_m$ should be self-contained: a correct answer to $q'_m$ by itself is a correct answer to $q_m$ combined with the question's preceding history $H$.

Figure 1 shows the assumptions of CQA and how they are made explicit in rewrites. The first question omits the title of the page (Anna Vissi), the second question omits the answer to the first question (replacing both Anna Vissi and her husband with the pronoun "they"), and the last question adds a scalar implicature that must be resolved.

## 3 Dataset Construction

We elicit paraphrases from human crowdworkers to make previously context-dependent questions *unambiguously* answerable. Through this process, we resolve difficult coreference linkages and create a pair-wise mapping between ambiguous and context-enriched questions. We derive CANARD from QuAC (Choi et al., 2018), a sequential question answering dataset about specific Wikipedia sections. QuAC uses a pair of workers—a "student" and a "teacher"—to ask and respond to questions. The "student" asks questions about a topic based on only the title of the Wikpedia article and the title of the target section. The "teacher" has access to the full Wikipedia section and provides answers by selecting text that answers the question. With this methodology, QuAC gathers 98k questions across 13,594 conversations. We take their entire dev set and a sample of their train set and create a custom JavaScript task in Mechanical Turk that allows workers to rewrite these questions. JavaScript hints help train the users and provides automated, real-time feedback.

We provide workers with a comprehensive set of instructions and task examples. We ask them to rewrite the questions in natural sounding English while preserving the sentence structure of the original question. We discourage workers from introducing new words that are unmentioned in the previous utterances and ask them to copy phrases when appropriate from the original question. These instructions ensure that the rewrites only resolve conversation-dependent ambiguities. Thus, we encourage workers to create minimal edits; in Section 5.2, we take advantage of this to use BLEU for evaluating model-generated rewrites.

We display the questions in the conversation one at a time, since the rewrites should include only the previous utterance. After a rewrite to the question is submitted, the answer to the question is displayed. The next question is then displayed. This repeats until the end of the conversation. The full set of instructions and the data collection interface are provided in the appendix.

We apply quality control throughout our collection process. During the task, JavaScript checks automatically monitor and warn about common errors: submissions that are abnormally short (e.g., 'why'), rewrites that still have pronouns (e.g., 'he wrote this album'), or ambiguous words (e.g., 'this article', 'that'). Many QuAC questions ask about 'what/who else' or ask for 'other' or 'another' entity. For that class of questions, we ask workers to use a phrase such as 'other than', 'in addition to', 'aside from', 'besides', 'together with' or 'along with' with the appropriate context in their rewrite.

We gather and review our data in batches to screen potentially compromised data or low quality workers. A post-processing script flags suspicious

| |
|---|
| ORIGINAL: Was this an honest mistake by the media? |
| REWRITE: Was the claim of media regarding Leblanc's room come to true? |
| ORIGINAL: What was a single from their album? |
| REWRITE: What was a single from horslips' *album*? |
| ORIGINAL: Did they marry? |
| REWRITE: Did Hannah Arendt and Heidegger marry? |

Table 2: Not all rewrites correctly encode the context required to answer a question. We take two failures to provide examples of the two common issues: Changed Meaning (top) and Needs Context (middle). We provide an example with no issues (bottom) for comparison.

rewrites and workers who take and abnormally long or short time. We flag about 15% of our data. *Every* flagged question is manually reviewed by one of the authors and an entire HIT is discarded if one is deemed inadequate. We reject 19.9% of submissions and the rest comprise CANARD. Additionally, we filter out under-performing workers based on these rejections from subsequent batches. To minimize risk, we limit the initial pool of workers to those that have completed 500 HITs with over 90% accuracy and offer competitive payment of $0.50 per HIT.

We verify the efficacy of our quality control through manual review. A random sample of fifty questions sampled from the final dataset is reviewed for desirable characteristics by a native English speaker in Table 1. Each of the positive traits occurs in 90% or more of the questions. Based on our sample, our edits retain grammaticality, leave the question meaning unchanged, and use pronouns unambiguously. There are rare occasions where workers use a part of the answer to the question being rewritten or where some of the context is left ambiguous. These infrequent mistakes should not affect our models. We provide examples of failures in Table 2.

We use the rewrites of QuAC's development set as our test set (5,571 question-in-context and corresponding rewrite pairs) and use a 10% sample of QuAC's training set rewrites as our development set (3,418); the rest are training data (31,538).

| | Dev | Test |
|---|---|---|
| **Copy** | 33.84 | 36.25 |
| **Pronoun Sub** | 47.72 | 47.44 |
| **Seq2Seq** | 51.37 | 49.67 |
| **Human Rewrites*** | 59.92 | |

Table 3: BLEU scores of the baseline models on development and test data. Seq2Seq improves up to four points over naive baselines but still well below human accuracy. Human accuracy (*) is computed from a small subset of the validation set.

## 4 Baselines

We compare three baseline models for the question-in-context rewriting task. In the **Copy** baseline, the rewrite $q'_m$ is set to be the same as the input question $q_m$ without making any changes.

We also try a **Pronoun Substitution** baseline in which the first pronoun in $q_m$ is replaced with the topic entity of the conversation. We use the title of the corresponding Wikipedia article to the original QuAC conversation as the topic entity. Similar to the Copy baseline, the training data is not used in that baseline.

Unlike the previous baselines which do not use our rewrites as training data, the third baseline is a neural sequence-to-sequence (**Seq2Seq**) model with attention and a copy mechanism (Bahdanau et al., 2015; See et al., 2017). We construct the input sequence by concatenating all utterances in the history $H$, prepending them to $q_m$, and adding a special separator token between utterances. We use a bidirectional LSTM encoder-decoder model with shared the word embeddings between the encoder and the decoder.[2]

Since questions are written by humans, a human rewrites are the upper-bound for this task. However, annotators (especially crowdworkers) can be inconsistent or disagree. To estimate the human accuracy, we collect 100 pairs of rewritten questions; each pair has two rewrites of the same question (in its given context) by two different workers. We manually verify that all rewrites are valid and then use the pair of rewrites as a hypothesis and a reference.

Table 3 shows the BLEU scores produced by the baselines and humans over both the validation and the test sets.[3] Although a well-trained standard

[2]We initialize the embeddings with GloVE (Pennington et al., 2014) and train with a batch-size of 16 for 200000 steps. We use OpenNMT (Klein et al., 2018) implementation.

[3]We use multi-bleu-detok.perl (Sennrich et al., 2017)

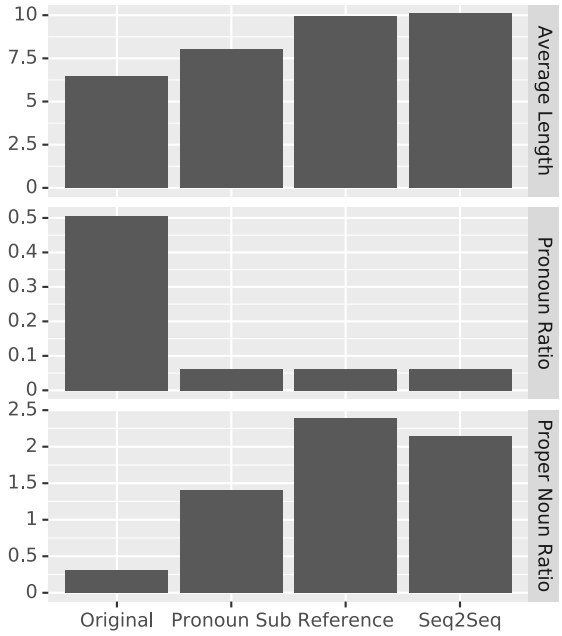


Figure 2: Human rewrites are longer, have fewer pronouns, and have more proper nouns than the original QuAC questions. Rewrites are longer and contain more proper nouns than our Pronoun Sub baseline and trained Seq2Seq model.

neural sequence-to-sequence improves 2–4 BLEU points over naive baselines, it is still 9 BLEU points below human-accuracy. We analyze sources of errors in the following section.

## 5 Dataset and Model Analysis

We analyze our dataset with automatic metrics after validating the reliability of our data (Section 3). We compare our dataset to the original QuAC questions and to automatically generated questions by our models. Then, we manually inspect the sources of rewriting errors in the seq2seq baseline.

### 5.1 Anaphora Resolution and Coreference

Our rewrites are longer, contain more nouns and less pronouns, and have more word types than the original data. Machine output lies in between the two human-generated corpora, but quality is difficult to assess. Figure 2 shows these statistics. We motivate our rewrites by exploring linguistic properties of our data. Anaphora resolution and coreference are two core NLP tasks applicable to this dataset, in addition to the downstream tasks evaluated in Section 4.

Pronouns occur in 53.9% of QuAC questions. Questions with pronouns are more likely to be ambiguous than those without any. Only 0.9% of these have pronouns that span more than one category (e.g., 'she' and 'his'). Hence, pronouns within a single sentence are likely unambiguous. However, 75.0% of the aggregate history has pronouns and the percentage of mixed category pronouns increase to 27.8% of our data. Therefore, pronoun disambiguation potentially becomes a problem for a quarter of the original data. An example is provided in Table 4.

| Label | Text |
|---|---|
| QUESTION | How long did he stay there? |
| REWRITE | How long did Cito Gaston stay at the Jays? |
| HISTORY | *Cito Gaston*<br>**Q:** What did Gaston do after the world series?<br>...<br>**Q:** Where did he go in 2001?<br>**A:** In 2002, he was hired by the Jays as special assistant to president and chief executive officer Paul Godfrey. |

Table 4: An example that had over ten flagged proper nouns in the history. Rewriting requires resolving challenging coreferences.

Approximately one-third of the questions generated by our pronoun-replacement baseline are within 85% string similarity to our rewritten questions. That leaves two-thirds of our data that cannot be solved with pronoun resolution alone.

### 5.2 Model Analysis

By manually examining the predictions of the seq2seq model, we notice that the main source of errors is that the model tends to find a short path to completing the rewrites. That often results in *under-specified questions* as in Example 1 in Table 5, *question meaning change* as in Example 2 or *meaningless questions* as in Example 3.

Another source of errors is having related entities mentioned in the context as Example 4 in Table 5, where the model confused "Copa America" with "Argentina". The model also struggles with listing multiple entities mentioned in different parts of the context. Example 5 in Table 5 show the output and the reference rewrites of the question *"Did she have any more works than those 3?"*, where two of the three entities—"United States of Banana", "La Comedia" and "Asalto al tiempo"—are lost in the rewrite.

| | Seq2Seq output | Reference |
|---|---|---|
| 1 | What did Chamberlain's men do? | What did Chamberlain's men do during the Battle of Gettysburg? |
| 2 | How many games did Ozzie Smith win? | How many games did the Cardinals win while Ozzie Smith played? |
| 3 | Did 108th get to the finals? | Did the US Women's Soccer Team get to the finals in the 1999 World Cup? |
| 4 | Did Gabriel Batistuta reside in any other countries, besides touring in the Copa America? | Besides Argentina, did Gabriel Batistuta reside in any other countries? |
| 5 | Did La Comedia have any more works than La Comedia 3? | Did Giannina Braschi have any more works than United States of Banana, La Comedia and Asalto al tiempo? |

Table 5: Example erroneous rewrites generated by the Seq2Seq models and their corresponding reference rewrites. The dominant source of error is the model tendency to produce short rewrites (Examples 1–3). Related entities (Copa America and Argentina in Example 4) distract the model. The model struggles with listing multiple entities mentioned in different parts of the context (Example 5).

## 6 Related Work and Discussion

Recent work in CQA has used simple concatenation (Elgohary et al., 2018), sequential neural models (Huang et al., 2019), and transformers (Qu et al., 2019a) for modeling the interaction between the conversation history, the question and reference documents. Some of the components in those models, such as relevant history turn selection (Qu et al., 2019b), can be adopted in question rewriting models for our task. An interesting avenue for future work is to incorporate deeper context, either from other modalities (Das et al., 2017) or from other dialog comprehension tasks (Sun et al., 2019).

Parallel to our work, Rastogi et al. (2019) and Su et al. (2019) introduce utterance rewriting datasets for dialog state tracking. Rastogi et al. (2019) covers a narrow set of domains and the rewrites of Su et al. (2019) are based on Chinese dialog with two-turn fixed histories. In contrast, CANARD has histories of variable turn lengths, covers wider topics, and is based on CQA.

Training question rewriting using reinforcement learning with the task accuracy as reward signal is explored in retrieval-based QA (Liu et al., 2019) and in MRC (Buck et al., 2018). A natural question is whether reinforcement learning could learn to retain the necessary context to rewrite questions in CQA. However, our dataset could be used to pretrain a question rewriter that can further be refined using reinforcement learning.

More broadly, we hope CANARD can drive human-computer collaboration in QA (Feng and Boyd-Graber, 2019). While questions typically vary in difficulty (Sugawara et al., 2018), existing research either introduces new benchmarks of difficult (adversarial) stand-alone questions (Dua et al., 2019; Wallace et al., 2019, inter alia), or models that simplify hard questions through paraphrasing (Dong et al., 2017) or decomposition (Talmor and Berant, 2018). We aim at studying QA models that can ask for human assistance (feedback) when they struggle to answer a question.

The reading comprehension setup of CQA provides a controlled environment where the main source of difficulty is interpreting a question in its context. The interactive component of CQA also provides a natural mechanism for improving rewriting. When the computer cannot understand (rewrite) a question because of complicated context, missing world knowledge, or upstream errors (Peskov et al., 2019) in the course of a conversation, it should be able to ask its interlocutor, "can you unpack that?" This dataset helps start that conversation; the next steps are developing and evaluating models that efficiently decide when to ask for human assistance, and how to best use this assistance.

## Acknowledgments

We thank Jo Shoemaker and Michelle Yuan for crowdsourcing feedback. This work was supported by NSF grant IIS-1822494. Boyd-Graber and Peskov were also supported by IIS-1748663. Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the view of the sponsors.

## References


Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*.

Christian Buck, Jannis Bulian, Massimiliano Ciaramita, Wojciech Gajewski, Andrea Gesmundo, Neil Houlsby, and Wei Wang. 2018. Ask the right questions: Active question reformulation with reinforcement learning. In *Proceedings of the International Conference on Learning Representations*.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. QuAC: Question answering in context. In *Proceedings of Empirical Methods in Natural Language Processing*.

Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José MF Moura, Devi Parikh, and Dhruv Batra. 2017. Visual dialog. In *Computer Vision and Pattern Recognition*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Li Dong, Jonathan Mallinson, Siva Reddy, and Mirella Lapata. 2017. Learning to paraphrase for question answering. In *Proceedings of Empirical Methods in Natural Language Processing*.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Ahmed Elgohary, Chen Zhao, and Jordan Boyd-Graber. 2018. Dataset and baselines for sequential open-domain question answering. In *Proceedings of Empirical Methods in Natural Language Processing*.

Shi Feng and Jordan Boyd-Graber. 2019. What AI can do for me: Evaluating machine learning interpretations in cooperative play. In *International Conference on Intelligent User Interfaces*.

Hsin-Yuan Huang, Eunsol Choi, and Wen tau Yih. 2019. FlowQA: Grasping flow in history for conversational machine comprehension. In *Proceedings of the International Conference on Learning Representations*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Vincent Nguyen, Jean Senellart, and Alexander Rush. 2018. OpenNMT: Neural machine translation toolkit. In *Proceedings of Association for Machine Translation in the Americas*.

Ye Liu, Chenwei Zhang, Xiaohui Yan, Yi Chang, and Philip S Yu. 2019. Generative question refinement with deep reinforcement learning in retrieval-based QA system. In *Proceedings of the ACM International Conference on Information and Knowledge Management*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of Empirical Methods in Natural Language Processing*.

Denis Peskov, Joe Barrow, Pedro Rodriguez, Graham Neubig, and Jordan Boyd-Graber. 2019. Mitigating noisy inputs for question answering. In *Proceedings of the Annual Conference of the International Speech Communication Association*.

Chen Qu, Liu Yang, Minghui Qiu, W. Bruce Croft, Yongfeng Zhang, and Mohit Iyyer. 2019a. BERT with history modeling for conversational question answering. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*.

Chen Qu, Liu Yang, Minghui Qiu, Yongfeng Zhang, Cen Chen, W. Bruce Croft, and Mohit Iyyer. 2019b. Attentive history selection for conversational question answering. In *Proceedings of the ACM International Conference on Information and Knowledge Management*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of Empirical Methods in Natural Language Processing*.

Pushpendre Rastogi, Arpit Gupta, Tongfei Chen, and Lambert Mathias. 2019. Scaling multi-domain dialogue state tracking via query reformulation. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Siva Reddy, Danqi Chen, and Christopher D Manning. 2019. CoQA: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the Association for Computational Linguistics*.

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nadejde. 2017. Nematus: a toolkit for neural machine translation. In *Proceedings of the Software Demonstrations of the Conference of the European Chapter of the Association for Computational Linguistics*.

Hui Su, Xiaoyu Shen, Rongzhi Zhang, Fei Sun, Pengwei Hu, Cheng Niu, and Jie Zhou. 2019. Improving multi-turn dialogue modelling with utterance

ReWriter. In *Proceedings of the Association for Computational Linguistics*.

Saku Sugawara, Kentaro Inui, Satoshi Sekine, and Akiko Aizawa. 2018. What makes reading comprehension questions easier? In *Proceedings of Empirical Methods in Natural Language Processing*.

Kai Sun, Dian Yu, Jianshu Chen, Dong Yu, Yejin Choi, and Claire Cardie. 2019. DREAM: A challenge dataset and models for dialogue-based reading comprehension. *Transactions of the Association for Computational Linguistics*.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Eric Wallace, Pedro Rodriguez, Shi Feng, Ikuya Yamada, and Jordan Boyd-Graber. 2019. Trick me if you can: Human-in-the-loop generation of adversarial question answering examples. In *Transactions of the Association for Computational Linguistics*.

Bonnie Webber. 1992. Question answering. In Stuart C. Shapiro, editor, *Encyclopedia of Artificial Intelligence*, 2nd edition, pages 814–822. John Wiley & Sons, Inc., New York, NY.

Dani Yogatama, Cyprien de Masson d'Autume, Jerome Connor, Tomás Kociský, Mike Chrzanowski, Lingpeng Kong, Angeliki Lazaridou, Wang Ling, Lei Yu, Chris Dyer, and Phil Blunsom. 2019. Learning and evaluating general linguistic intelligence. *arXiv preprint arXiv:1901.11373*.

# QUOREF: A Reading Comprehension Dataset with Questions Requiring Coreferential Reasoning

**Pradeep Dasigi**[♡] **Nelson F. Liu**[♡♣] **Ana Marasović**[♡]
**Noah A. Smith**[♡♣] **Matt Gardner**[♡]
[♡]Allen Institute for Artificial Intelligence
[♣]Paul G. Allen School of Computer Science & Engineering, University of Washington
{pradeepd,anam,mattg}@allenai.org, {nfliu,nasmith}@cs.washington.edu

## Abstract

Machine comprehension of texts longer than a single sentence often requires coreference resolution. However, most current reading comprehension benchmarks do not contain complex coreferential phenomena and hence fail to evaluate the ability of models to resolve coreference. We present a new crowdsourced dataset containing more than 24K span-selection questions that require resolving coreference among entities in over 4.7K English paragraphs from Wikipedia. Obtaining questions focused on such phenomena is challenging, because it is hard to avoid lexical cues that shortcut complex reasoning. We deal with this issue by using a strong baseline model as an adversary in the crowdsourcing loop, which helps crowdworkers avoid writing questions with exploitable surface cues. We show that state-of-the-art reading comprehension models perform significantly worse than humans on this benchmark—the best model performance is 70.5 $F_1$, while the estimated human performance is 93.4 $F_1$.

## 1 Introduction

Paragraphs and other longer texts typically make multiple references to the same entities. Tracking these references and resolving coreference is essential for full machine comprehension of these texts. Significant progress has recently been made in reading comprehension research, due to large crowdsourced datasets (Rajpurkar et al., 2016; Bajaj et al., 2016; Joshi et al., 2017; Kwiatkowski et al., 2019, *inter alia*). However, these datasets focus largely on understanding local predicate-argument structure, with very few questions requiring long-distance entity tracking. Obtaining such questions is hard for two reasons: (1) teaching crowdworkers about coreference is challenging, with even experts disagreeing on its nuances (Pradhan et al., 2007; Versley, 2008; Recasens et al., 2011; Poesio et al., 2018), and (2) even if we can get crowdworkers to target coreference phenomena in their questions, these questions may contain giveaways that let models arrive at the correct answer without performing the desired reasoning (see §3 for examples).

Byzantines were avid players of tavli (Byzantine Greek: τάβλη), a game known in English as backgammon, which is still popular in former Byzantine realms, and still known by the name tavli in Greece. Byzantine nobles were devoted to horsemanship, particularly ***tzykanion***, now known as *polo*. The game came from Sassanid Persia in the early period and a Tzykanisterion (stadium for playing *the game*) was built by Theodosius II (r. 408–450) inside the Great Palace of Constantinople. Emperor Basil I (r. 867–886) excelled at **it**; Emperor Alexander (r. 912–913) died from exhaustion while playing, Emperor Alexios I Komnenos (r. 1081–1118) was injured while playing with Tatikios, and John I of Trebizond (r. 1235–1238) died from a fatal injury during a game. Aside from Constantinople and Trebizond, other Byzantine cities also featured tzykanisteria, most notably Sparta, Ephesus, and Athens, an indication of a thriving urban aristocracy.

Q1. What is the Byzantine name of the game that Emperor Basil I excelled at? **it → tzykanion**
Q2. What are the names of the sport that is played in a Tzykanisterion? *the game → tzykanion*; *polo*
Q3. What cities had tzykanisteria? cities → Constantinople; Trebizond; Sparta; Ephesus; Athens

Figure 1: Example paragraph and questions from the dataset. Highlighted text in paragraphs is where the questions with matching highlights are anchored. Next to the questions are the relevant coreferent mentions from the paragraph. They are bolded for the first question, italicized for the second, and underlined for the third in the paragraph.

We introduce a new dataset, QUOREF,[1] that contains questions requiring coreferential reasoning (see examples in Figure 1). The questions are derived from paragraphs taken from a diverse set of English Wikipedia articles and are collected using an annotation process (§2) that deals with the aforementioned issues in the following ways:

[1]Links to dataset, code, models, and leaderboard available at https://allennlp.org/quoref.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5925–5932, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

First, we devise a set of instructions that gets workers to find anaphoric expressions and their referents, asking questions that connect two mentions in a paragraph. These questions mostly revolve around traditional notions of coreference (Figure 1 Q1), but they can also involve referential phenomena that are more nebulous (Figure 1 Q3). Second, inspired by Dua et al. (2019), we disallow questions that can be answered by an adversary model (uncased base BERT, Devlin et al., 2019, trained on SQuAD 1.1, Rajpurkar et al., 2016) running in the background as the workers write questions. This adversary is not particularly skilled at answering questions requiring coreference, but can follow obvious lexical cues—it thus helps workers avoid writing questions that shortcut coreferential reasoning.

QUOREF contains more than 24K questions whose answers are spans or sets of spans in 4.7K paragraphs from English Wikipedia that can be arrived at by resolving coreference in those paragraphs. We manually analyze a sample of the dataset (§3) and find that 78% of the questions cannot be answered without resolving coreference. We also show (§4) that the best system performance is 70.5% $F_1$, while the estimated human performance is 93.4%. These findings indicate that this dataset is an appropriate benchmark for coreference-aware reading comprehension.

## 2 Dataset Construction

**Collecting paragraphs** We scraped paragraphs from Wikipedia pages about English movies, art and architecture, geography, history, and music. For movies, we followed the list of English language films,[2] and extracted plot summaries that are at least 40 tokens, and for the remaining categories, we followed the lists of featured articles.[3] Since movie plot summaries usually mention many characters, it was easier to find hard QUOREF questions for them, and we sampled about 40% of the paragraphs from this category.

**Crowdsourcing setup** We crowdsourced questions about these paragraphs on Mechanical Turk. We asked workers to find two or more co-referring spans in the paragraph, and to write questions such that answering them would require the knowledge that those spans are coreferential. We did not ask them to explicitly mark the co-referring spans. Workers were asked to write questions for a random sample of paragraphs from our pool, and we showed them examples of good and bad questions in the instructions (see Appendix A). For each question, the workers were also required to select one or more spans in the corresponding paragraph as the answer, and these spans are not required to be same as the coreferential spans that triggered the questions.[4] We used an uncased base BERT QA model (Devlin et al., 2019) trained on SQuAD 1.1 (Rajpurkar et al., 2016) as an adversary running in the background that attempted to answer the questions written by workers in real time, and the workers were able to submit their questions only if their answer did not match the adversary's prediction.[5] Appendix A further details the logistics of the crowdsourcing tasks. Some basic statistics of the resulting dataset can be seen in Table 1.

| | Train | Dev. | Test |
|---|---|---|---|
| Number of questions | 19399 | 2418 | 2537 |
| Number of paragraphs | 3771 | 454 | 477 |
| Avg. paragraph len (tokens) | 384±105 | 381±101 | 385±103 |
| Avg. question len (tokens) | 17±6 | 17±6 | 17±6 |
| Paragraph vocabulary size | 57648 | 18226 | 18885 |
| Question vocabulary size | 19803 | 5579 | 5624 |
| % of multi-span answers | 10.2 | 9.1 | 9.7 |

Table 1: Key statistics of QUOREF splits.

## 3 Semantic Phenomena in QUOREF

To better understand the phenomena present in QUOREF, we manually analyzed a random sample of 100 paragraph-question pairs. The following are some empirical observations.

**Requirement of coreference resolution** We found that 78% of the manually analyzed questions cannot be answered without coreference resolution. The remaining 22% involve some form of coreference, but do not require it to be resolved for answering them. Examples include a paragraph that mentions only one city, "*Bristol*", and a sentence that says "*the city was bombed*". The associated question, *Which city was bombed?*, does not really require coreference resolution from a model

[2] https://en.wikipedia.org/wiki/Category:English-language_films

[3] https://en.wikipedia.org/wiki/Wikipedia:Featured_articles

[4] For example, the last question in Table 2 is about the coreference of {*she*, *Fania*, *his mother*}, but none of these mentions is the answer.

[5] Among models with acceptably low latency, we qualitatively found uncased base BERT to be the most effective.

| Reasoning | Paragraph Snippet | Question | Answer |
|---|---|---|---|
| **Pronominal resolution (69%)** | Anna and Declan eventually make their way on foot to a roadside pub, where they discover **the three van thieves** going through Anna's luggage. **Declan fights *them***, displaying unexpected strength for a man of his size, and retrieves Anna's bag. | Who does **Declan get into a fight** with? | the three van thieves |
| **Nominal resolution (54%)** | Later, **Hippolyta** was granted a daughter, Princess Diana, . . . **Diana defies *her mother*** and . . . | What is the name of the person who is **defied by her daughter**? | Hippolyta |
| **Multiple resolutions (32%)** | The now upbeat collective keep **the toucan**, nicknaming it "**Amigo**" . . . When authorities show up to catch **the bird**, Pete and Liz spirit **him** away by **Liz hiding *him* in her dress** . . . | What is the name of the character who **hides in Liz's dress**? | Amigo |
| **Commonsense reasoning (10%)** | Amos reflects back on his early childhood . . . with **his mother *Fania* and father Arieh**. . . . One of **his mother**'s friends is killed while hanging up laundry during the war. . . . **Fania** falls into a depression. . . **she** . . . goes to . . . Tel Aviv, where **she** kills **herself** by overdose . . . | How does **Arieh's wife** die? | kills herself by overdose |

Table 2: Phenomena in QUOREF. Note that the first two classes are not disjoint. In the final example, the paragraph does not explicitly say that *Fania* is *Arieh*'s wife.

that can identify city names, making the content in the question after *Which city* unnecessary.

**Types of coreferential reasoning** Questions in QUOREF require resolving pronominal and nominal mentions of entities. Table 2 shows percentages and examples of analyzed questions that fall into these two categories. These are not disjoint sets, since we found that 32% of the questions require both (row 3). We also found that 10% require some form of commonsense reasoning (row 4).

## 4 Baseline Model Performance on QUOREF

We evaluated two classes of baseline models on QUOREF: state-of-the art reading comprehension models that predict single spans (§4.1) and heuristic baselines to look for annotation artifacts (§4.2).

We use two evaluation metrics to compare model performance: exact match (EM), and a (macro-averaged) $F_1$ score that measures overlap between a bag-of-words representation of the gold and predicted answers. We use the same implementation of EM as SQuAD, and we employ the $F_1$ metric used for DROP (Dua et al., 2019). See Appendix B for model training hyperparameters and other details.

| Method | Dev | | Test | |
|---|---|---|---|---|
| | EM | $F_1$ | EM | $F_1$ |
| **Heuristic Baselines** | | | | |
| passage-only BERT QA | 19.77 | 25.56 | 21.25 | 29.27 |
| passage-only XLNet QA | 18.44 | 27.59 | 18.57 | 28.24 |
| **Reading Comprehension** | | | | |
| QANet | 34.41 | 38.26 | 34.17 | 38.90 |
| QANet+BERT | 43.09 | 47.38 | 42.41 | 47.20 |
| BERT QA | 58.44 | 64.95 | 59.28 | 66.39 |
| XLNet QA | **64.52** | **71.49** | **61.88** | **70.51** |
| **Human Performance** | - | - | 86.75 | 93.41 |

Table 3: Performance of various baselines on QUOREF, measured by exact match (EM) and $F_1$. Boldface marks the best systems for each metric and split.

### 4.1 Reading Comprehension Models

We test four single-span (SQuAD-style) reading comprehension models: (1) **QANet** (Yu et al., 2018), currently the best-performing published model on SQuAD 1.1 without data augmentation or pretraining; (2) **QANet + BERT**, which enhances the QANet model by concatenating frozen BERT representations to the original input embeddings; (3) **BERT QA** (Devlin et al., 2019), the adversarial baseline used in data construction, and (4) **XLNet QA** (Yang et al., 2019), another large pretrained language model based on the Transformer architecture (Vaswani et al., 2017) that outperforms BERT QA on reading comprehension

benchmarks SQuAD and RACE (Lai et al., 2017).

We use the AllenNLP (Gardner et al., 2018) implementation of QANet modified to use the marginal objective proposed by Clark and Gardner (2018) and `pytorch-transformers`[6] implementation of base BERT QA[7] and base XLNet QA. BERT is pretrained on English Wikipedia and BookCorpus (Zhu et al., 2015) (3.87B wordpieces, 13GB of plain text) and XLNet additionally on Giga5 (Napoles et al., 2012), ClueWeb 2012-B (extended from Smucker et al., 2009), and Common Crawl[8] (32.89B wordpieces, 78GB of plain text).

### 4.2 Heuristic Baselines

In light of recent work exposing predictive artifacts in crowdsourced NLP datasets (Gururangan et al., 2018; Kaushik and Lipton, 2018, *inter alia*), we estimate the effect of predictive artifacts by training BERT QA and XLNet QA to predict a single start and end index given only the passage as input (**passage-only**).

### 4.3 Results

Table 3 presents the performance of all baseline models on QUOREF.

The best performing model is XLNet QA, which reaches an $F_1$ score of 70.5 in the test set. However, it is still more than 20 $F_1$ points below human performance.[9]

BERT QA trained on QUOREF under-performs XLNet QA, but still gets a decent $F_1$ score of 66.4. Note that BERT QA trained on SQuAD would have achieved an $F_1$ score of 0, since our dataset was constructed with that model as the adversary. The extent to which BERT QA does well on QUOREF might indicate its capacity for coreferential reasoning that was not exploited when it was trained on SQuAD (for a detailed discussion of this phenomenon, see Liu et al., 2019). Our analysis of model errors in §4.4 shows that some of the improved performance may also be due to artifacts in QUOREF.

We notice smaller improvements from XLNet QA over BERT QA (4.12 in $F_1$ test score, 2.6 in EM test score) on QUOREF compared to other reading comprehension benchmarks: SQuAD and RACE (see Yang et al., 2019). This might indicate the insufficiency of pretraining on more data (XL Net was pretrained on 6 times more plain text, nearly 10 times more wordpieces than BERT), for coreferential reasoning.

The passage-only baseline under-performs all other systems; examining its predictions reveals that it almost always predicts the most frequent entity in the passage. Its relatively low performance, despite the tendency for Wikipedia articles and passages to be written about a single entity, indicates that a large majority of questions likely require coreferential reasoning.

### 4.4 Error Analysis

We analyzed the predictions from the baseline systems to estimate the extent to which they really understand coreference.

Since the contexts in QUOREF come from Wikipedia articles, they are often either about a specific entity, or are narratives with a single protagonist. We found that the baseline models exploit this property to some extent. We observe that 51% of the QUOREF questions in the development set that were correctly answered by BERT QA were either the first or the most frequent entity in the paragraphs, while in the case of those that were incorrectly answered, this value is 12%. XLNet QA also exhibits a similar trend, with the numbers being 48% and 11%, respectively.

QA systems trained on QUOREF often need to find entities that occur far from the locations in the paragraph at which that the questions are anchored. To assess whether the baseline systems exploited answers being close, we manually analyzed predictions of BERT QA and XLNet QA on 100 questions in the development set, and found that the answers to 17% of the questions correctly answered by XLNet QA are the nearest entities, whereas the number is 4% for those incorrectly answered. For BERT QA, the numbers are 17% and 6% respectively.

## 5 Related Work

**Traditional coreference datasets** Unlike traditional coreference annotations in datasets like those of Pradhan et al. (2007), Ghaddar and Langlais (2016), Chen et al. (2018) and Poesio et al. (2018), which aim to obtain complete coref-

[6] https://github.com/huggingface/pytorch-transformers

[7] The large BERT model does not fit in the available GPU memory.

[8] https://commoncrawl.org/

[9] Human performance was estimated from the authors' answers of 400 questions from the test set, scored with the same metric used for systems.

erence clusters, our questions require understanding coreference between only a few spans. While this means that the notion of coreference captured by our dataset is less comprehensive, it is also less conservative and allows questions about coreference relations that are not marked in OntoNotes annotations. Since the notion is not as strict, it does not require linguistic expertise from annotators, making it more amenable to crowdsourcing. Guha et al. (2015) present the limitations of annotating coreference in newswire texts alone, and like us, built a non-newswire coreference resolution dataset focusing on Quiz Bowl questions. There is some other recent work (Poesio et al., 2019; Aralikatte and Søgaard, 2019) in crowdsourcing coreference judgments that relies on a relaxed notion of coreference as well.

**Reading comprehension datasets** There are many reading comprehension datasets (Richardson et al., 2013; Rajpurkar et al., 2016; Kwiatkowski et al., 2019; Dua et al., 2019, *inter alia*). Most of these datasets principally require understanding local predicate-argument structure in a paragraph of text. QUOREF also requires understanding local predicate-argument structure, but makes the reading task harder by explicitly querying anaphoric references, requiring a system to track entities throughout the discourse.

## 6 Conclusion

We present QUOREF, a focused reading comprehension benchmark that evaluates the ability of models to resolve coreference. We crowdsourced questions over paragraphs from Wikipedia, and manual analysis confirmed that most cannot be answered without coreference resolution. We show that current state-of-the-art reading comprehension models perform significantly worse than humans. Both these findings provide evidence that QUOREF is an appropriate benchmark for coreference-aware reading comprehension.

## Acknowledgments

We thank the anonymous reviewers for the useful discussion. Thanks to HuggingFace for releasing `pytorch-transformers`, and to Dheeru Dua for sharing with us the crowdsourcing setup used for DROP.

## References

Rahul Aralikatte and Anders Søgaard. 2019. Model-based annotation of coreference. arXiv:1906.10724.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. 2016. MS MARCO: A human generated machine reading comprehension dataset. In *NeurIPS*.

Hong Chen, Zhenhua Fan, Hao Lu, Alan Yuille, and Shu Rong. 2018. PreCo: A large-scale dataset in preschool vocabulary for coreference resolution. In *Proc. of EMNLP*.

Christopher Clark and Matt Gardner. 2018. Simple and effective multi-paragraph reading comprehension. In *Proc. of ACL*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. of NAACL*.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proc. of NAACL*.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. In *Proc. of Workshop for NLP Open Source Software*.

Abbas Ghaddar and Phillippe Langlais. 2016. WikiCoref: An English coreference-annotated corpus of Wikipedia articles. In *Proc. of LREC*.

Anupam Guha, Mohit Iyyer, Danny Bouman, and Jordan Boyd-Graber. 2015. Removing the training wheels: A coreference dataset that entertains humans and challenges computers. In *Proc. of NAACL*.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A Smith. 2018. Annotation artifacts in natural language inference data. In *Proc. of NAACL*.

Mandar S. Joshi, Eunsol Choi, Daniel S. Weld, and Luke S. Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proc. of ACL*.

Divyansh Kaushik and Zachary Chase Lipton. 2018. How much reading does reading comprehension require? A critical investigation of popular benchmarks. In *Proc. of EMNLP*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones,

Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural Questions: A benchmark for question answering research. In *TACL*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proc. of EMNLP*.

Nelson F. Liu, Roy Schwartz, and Noah A. Smith. 2019. Inoculation by fine-tuning: A method for analyzing challenge datasets. In *Proc. of NAACL*.

Courtney Napoles, Matthew R. Gormley, and Benjamin Van Durme. 2012. Annotated Gigaword. Technical report.

Massimo Poesio, Jon Chamberlain, Silviu Paun, Juntao Yu, Alexandra Uma, and Udo Kruschwitz. 2019. A crowdsourced corpus of multiple judgments and disagreement on anaphoric interpretation. In *Proc. of NAACL*.

Massimo Poesio, Yulia Grishina, Varada Kolhatkar, Nafise Moosavi, Ina Roesiger, Adam Roussel, Fabian Simonjetz, Alexandra Uma, Olga Uryupina, Juntao Yu, and Heike Zinsmeister. 2018. Anaphora resolution with the ARRAU corpus. In *Proc. of CRAC Workshop*.

Sameer S. Pradhan, Lance Ramshaw, Ralph Weischedel, Jessica MacBride, and Linnea Micciulla. 2007. Unrestricted coreference: Identifying entities and events in OntoNotes. In *Proc. of ICSC*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proc. of EMNLP*.

Marta Recasens, Eduard Hovy, and M Antònia Martí. 2011. Identity, non-identity, and near-identity: Addressing the complexity of coreference. *Lingua*, 121(6):1138–1152.

Matthew Richardson, Christopher J. C. Burges, and Erin Renshaw. 2013. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proc. of EMNLP*.

Mark D. Smucker, Charles L. A. Clarke, and Gordon V. Cormack. 2009. Experiments with clueweb09: Relevance feedback and web tracks. In *Proc. of TREC*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proc. of NeurIPS*.

Yannick Versley. 2008. Vagueness and referential ambiguity in a large-scale annotated corpus. *Research on Language and Computation*, 6(3):333–353.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. arXiv:1906.08237.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. 2018. QANet: Combining local convolution with global self-attention for reading comprehension. In *Proc. of ICLR*.

Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *Proc. of ICCV*.

# Appendices

## A Crowdsourcing Logistics

### A.1 Instructions

The crowdworkers were giving the following instructions:

"In this task, you will look at paragraphs that contain several phrases that are references to names of people, places, or things. For example, in the first sentence from sample paragraph below, the references Unas and the ninth and final king of Fifth Dynasty refer to the same person, and Pyramid of Unas, Unas's pyramid and the pyramid refer to the same construction. You will notice that multiple phrases often refer to the same person, place, or thing. Your job is to write questions that you would ask a person to see if they understood that the phrases refer to the same entity. To help you write such questions, we provided some examples of good questions you can ask about such phrases. We also want you to avoid questions that can be answered correctly by someone without actually understanding the paragraph. To help you do so, we provided an AI system running in the background that will try to answer the questions you write. You can consider any question it can answer to be too easy. However, please note that the AI system incorrectly answering a question does not necessarily mean that it is good. Please read the examples below carefully to understand what kinds of questions we are interested in."

### A.2 Examples of Good Questions

We illustrate examples of good questions for the following paragraph.

"The Pyramid of Unas is a smooth-sided pyramid built in the 24th century BC for the Egyptian pharaoh Unas, the ninth and final king of the Fifth Dynasty. It is the smallest Old Kingdom pyramid, but significant due to the discovery of Pyramid Texts, spells for the king's afterlife incised into the walls of its subterranean chambers. Inscribed for the first time in Unas's pyramid, the tradition of funerary texts carried on in the pyramids of subsequent rulers, through to the end of the Old Kingdom, and into the Middle Kingdom through the Coffin Texts which form the basis of the Book of the Dead. Unas built his pyramid between the complexes of Sekhemket and Djoser, in North Saqqara. Anchored to the valley temple via a nearby lake, a long causeway was constructed to provide access to the pyramid site. The causeway had elaborately decorated walls covered with a roof which had a slit in one section allowing light to enter illuminating the images. A long wadi was used as a pathway. The terrain was difficult to negotiate and contained old buildings and tomb superstructures. These were torn down and repurposed as underlay for the causeway. A significant stretch of Djoser's causeway was reused for embankments. Tombs that were on the path had their superstructures demolished and were paved over, preserving their decorations."

The following questions link pronouns:

**Q1:** What is the name of the person whose pyramid was built in North Saqqara? **A:** Unas

**Q2:** What is significant due to the discovery of Pyramid Texts? **A:** The Pyramid of Unas

**Q3:** What were repurposed as underlay for the causeway? **A:** old buildings; tomb superstructures

The following questions link other references:

**Q1:** What is the name of the king for whose afterlife spells were incised into the walls of the pyramid? **A:** Unas

**Q2:** Where did the final king of the Fifth dynasty build his pyramid? **A:** between the complexes of Sekhemket and Djoser, in North Saqqara

### A.3 Examples of Bad Questions

We illustrate examples of bad questions for the following paragraph.

"Decisions by Republican incumbent Peter Fitzgerald and his Democratic predecessor Carol Moseley Braun to not participate in the election resulted in wide-open Democratic and Republican primary contests involving fifteen candidates. In the March 2004 primary election, Barack Obama won in an unexpected landslidewhich overnight made him a rising star within the national Democratic Party, started speculation about a presidential future, and led to the reissue of his memoir, Dreams from My Father. In July 2004, Obama delivered the keynote address at the 2004 Democratic National Convention, seen by 9.1 million

viewers. His speech was well received and elevated his status within the Democratic Party. Obama's expected opponent in the general election, Republican primary winner Jack Ryan, withdrew from the race in June 2004. Six weeks later, Alan Keyes accepted the Republican nomination to replace Ryan. In the November 2004 general election, Obama won with 70 percent of the vote. Obama cosponsored the Secure America and Orderly Immigration Act. He introduced two initiatives that bore his name: LugarObama, which expanded the NunnLugar cooperative threat reduction concept to conventional weapons; and the Federal Funding Accountability and Transparency Act of 2006, which authorized the establishment of USAspending.gov, a web search engine on federal spending. On June 3, 2008, Senator Obamaalong with three other senators: Tom Carper, Tom Coburn, and John McCain—introduced follow-up legislation: Strengthening Transparency and Accountability in Federal Spending Act of 2008."

The following questions do not require coreference resolution:

**Q1:** Who withdrew from the race in June 2004? **A:** Jack Ryan

**Q2:** What Act sought to build on the Federal Funding Accountability and Transparency Act of 2006? **A:** Strengthening Transparency and Accountability in Federal Spending Act of 2008

The following question has ambiguous answers:

**Q1:** Whose memoir was called Dreams from My Father? **A:** Barack Obama; Obama; Senator Obama

### A.4 Worker Pool Management

Beyond training workers with the detailed instructions shown above, we ensured that the questions are of high quality by selecting a good pool of 21 workers using a two-stage selection process, allowing only those workers who clearly understood the requirements of the task to produce the final set of questions. Both the qualification and final HITs had 4 paragraphs per HIT for paragraphs from movie plot summaries, and 5 per HIT for the other domains, from which the workers could choose. For each HIT, workers typically spent 20 minutes, were required to write 10 questions, and were paid US$7.

## B Experimental Setup Details

Unless otherwise mentioned, we adopt the original published procedures and hyperparameters used for each baseline.

**BERT QA and XLNet QA** We use uncased BERT, and cased XLNet, but lowercase our data while processing. We train our model with a batch size of 10, sequence length of 512 wordpieces, and a stride of 128. We use the AdamW optimizer, with a learning rate of $3^{-5}$. We train for 10 epochs, checkpointing the model after 19399 steps. We report the performance of the checkpoint which is the best on the dev set.

**QANet** Durining training, we truncate paragraphs to 400 (word) tokens during training and questions to 50 tokens. During evaluation, we truncate paragraphs to 1000 tokens and questions to 100 tokens.

**Passage-only baseline** We keep the HPs setup used for training BERT QA and XLNet QA and replace questions with empty strings.

# Zero-shot Reading Comprehension by Cross-lingual Transfer Learning with Multi-lingual Language Representation Model

**Tsung-Yuan Hsu*** **Chi-liang Liu*** **Hung-yi Lee**
Graduate Institute of Communication Engineering
National Taiwan University
{sivia89024, liangtaiwan1230, tlkagkb93901106}@gmail.com

## Abstract

Because it is not feasible to collect training data for every language, there is a growing interest in cross-lingual transfer learning. In this paper, we systematically explore zero-shot cross-lingual transfer learning on reading comprehension tasks with a language representation model pre-trained on multi-lingual corpus. The experimental results show that with pre-trained language representation zero-shot learning is feasible, and translating the source data into the target language is not necessary and even degrades the performance. We further explore what does the model learn in zero-shot setting[0].

## 1 Introduction

*Reading Comprehension* (RC) has become a central task in natural language processing, with great practical value in various industries. In recent years, many large-scale RC datasets in English (Hermann et al., 2015; Hewlett et al., 2016; Rajpurkar et al., 2016; Nguyen et al., 2016; Trischler et al., 2017; Joshi et al., 2017; Rajpurkar et al., 2018) have nourished the development of numerous powerful and diverse RC models (Seo et al., 2016; Hu et al., 2018; Wang et al., 2017; Clark and Gardner, 2018; Huang et al., 2017). The state-of-the-art model (Devlin et al., 2018) on SQuAD, one of the most widely used RC benchmarks, even surpasses human-level performance. Nonetheless, RC on languages other than English has been limited due to the absence of sufficient training data. Although some efforts have been made to create RC datasets for Chinese (He et al., 2018; Shao et al., 2018) and Korean (Seungyoung Lim, 2018), it is not feasible to collect RC datasets for every language since annotation efforts to collect a new RC dataset are often far from trivial. Therefore, the setup of transfer learning, especially zero-shot learning, is of extraordinary importance.

Existing methods (Asai et al., 2018) of cross-lingual transfer learning on RC datasets often count on machine translation (MT) to translate data from source language into target language, or vice versa. These methods may not require a well-annotated RC dataset for the target language, whereas a high-quality MT model is needed as a trade-off, which might not be available when it comes to low-resource languages.

In this paper, we leverage pre-trained multilingual language representation, for example, BERT learned from multilingual un-annotated sentences (multi-BERT), in cross-lingual zero-shot RC. We fine-tune multi-BERT on the training set in source language, then test the model in target language, with a number of combinations of source-target language pair to explore the cross-lingual ability of multi-BERT. Surprisingly, we find that the models have the ability to transfer between low lexical similarity language pair, such as English and Chinese. Recent studies (Lample and Conneau, 2019; Devlin et al., 2018; Wu and Dredze, 2019) show that cross-lingual language models have the ability to enable preliminary zero-shot transfer on simple natural language understanding tasks, but zero-shot transfer of RC has not been studied. To our knowledge, this is the first work systematically exploring the cross-lingual transferring ability of multi-BERT on RC tasks.

## 2 Zero-shot Transfer with Multi-BERT

Multi-BERT has showcased its ability to enable cross-lingual zero-shot learning on the natural language understanding tasks including XNLI (Conneau et al., 2018), NER, POS, Dependency Parsing, and so on. We now seek to know if a pre-trained multi-BERT has ability to solve RC tasks in the zero-shot setting.

*Equal contribution

[0]All the modifications of existing corpora used in this paper would be released in https://github.com/ntu-spml-lab/artificial-reading-comprehension-datasets



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5933–5940, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

## 2.1 Experimental Setup and Data

We have training and testing sets in three different languages: English, Chinese and Korean. The English dataset is SQuAD (Rajpurkar et al., 2016). The Chinese dataset is DRCD (Shao et al., 2018), a Chinese RC dataset with 30,000+ examples in the training set and 10,000+ examples in the development set. The Korean dataset is KorQuAD (Seungyoung Lim, 2018), a Korean RC dataset with 60,000+ examples in the training set and 10,000+ examples in the development set, created in exactly the same procedure as SQuAD. We always use the development sets of SQuAD, DRCD and KorQuAD for testing since the testing sets of the corpora have not been released yet.

Next, to construct a diverse cross-lingual RC dataset with compromised quality, we translated the English and Chinese datasets into more languages, with Google Translate[1]. An obvious issue with this method is that some examples might no longer have a recoverable span. To solve the problem, we use fuzzy matching[2] to find the most possible answer, which calculates minimal edit distance between translated answer and all possible spans. If the minimal edit distance is larger than min(10, lengths of translated answer - 1), we drop the examples during training, and treat them as noise when testing. In this way, we can recover more than 95% of examples. The following generated datasets are recovered with same setting.

The pre-trained multi-BERT is the official released one[3]. This multi-lingual version of BERT were pre-trained on corpus in 104 languages. Data in different languages were simply mixed in batches while pre-training, without additional effort to align between languages. When fine-tuning, we simply adopted the official training script of BERT, with default hyperparameters, to fine-tune each model until training loss converged.

## 2.2 Experimental Results

Table 1 shows the result of different models trained on either Chinese or English and tested on Chinese. In row (f), multi-BERT is fine-tuned on English but tested on Chinese, which achieves competitive performance compared with QANet trained on Chinese. We also find that multi-BERT trained on English has relatively lower EM compared with the model with comparable F1 scores. This shows that the model learned with zero-shot can roughly identify the answer spans in context but less accurate. In row (c), we fine-tuned a BERT model pre-trained on English monolingual corpus (English BERT) on Chinese RC training data directly by appending fastText-initialized Chinese word embeddings to the original word embeddings of English-BERT. Its F1 score is even lower than that of zero-shot transferring multi-BERT (rows (c) v.s. (e)). The result implies multi-BERT does acquire better cross-lingual capability through pre-training on multilingual corpus.

| Model | Train-set | EM | F1 |
|---|---|---|---|
| (a) Shao et al. 2018 | Chinese | - | 53.78 |
| (b) QANet[3] | Chinese | 66.10 | 78.10 |
| (c) English-BERT | Chinese | 65.00 | 76.96 |
| (d) Chinese-BERT | Chinese | 82.00 | 89.10 |
| (e) multi-BERT | Chinese | 81.24 | 88.68 |
| (f) multi-BERT | English | 63.31 | 78.82 |
| (g) multi-BERT | English +Chinese | 82.63 | 90.10 |

Table 1: EM/F1 scores over Chinese testing set.

| Train | Test | | |
|---|---|---|---|
| | English | Chinese | Korean |
| En | **81.2/88.6** | 63.3/78.8 | 49.2/69.3 |
| Zh | 34.1/53.8 | **81.2/88.7** | 56.4/78.2 |
| Kr | 58.5/68.4 | 73.4/82.7 | **69.41/89.3** |
| En-Fr | 67.5/76.4 | 56.5/72.5 | 37.2/56.3 |
| En-Zh | 59.7/71.4 | **61.4/78.8** | 49.0/72.7 |
| En-Jp | 53.3/64.9 | 62.4/76.7 | 50.4/72.0 |
| En-Kr | 41.7/50.1 | 56.7/71.6 | **47.1/70.8** |
| Zh-En | **26.6/44.1** | 57.7/71.1 | 40.5/59.5 |
| Zh-Fr | 23.4/39.8 | 44.9/62.0 | 39.6/59.9 |
| Zh-Jp | 25.5/42.6 | 60.9/72.4 | 44.9/65.7 |
| Zh-Kr | 26.5/42.2 | 58.2/69.5 | **47.4/67.7** |

Table 2: EM/F1 score of multi-BERTs fine-tuned on different training sets and tested on different languages (En: English, Fr: French, Zh: Chinese, Jp: Japanese, Kr: Korean, xx-yy: translated from xx to yy). The text in bold means training data language is the same as testing data language.

Table 2 shows the results of multi-BERT fine-tuned on different languages and then tested on English , Chinese and Korean. The top half of the table shows the results of training data without translation. It is not surprising that when the training and testing sets are in the same language, the best results are achieved, and multi-BERT shows transfer capability when training and testing sets are in different languages, especially between Chinese and Korean.

In the lower half of Table 2, the results are ob-

[1] https://translate.google.com/
[2] https://github.com/taleinat/fuzzysearch
[3] https://github.com/google-research/bert

tained by the translated training data. First, we found that when testing on English and Chinese, translation always degrades the performance (En v.s. En-XX, Zh v.s. Zh-XX). Even though we translate the training data into the same language as testing data, using the untranslated data still yield better results. For example, when testing on English, the F1 score of the model training on Chinese (Zh) is 53.8, while the F1 score is only 44.1 for the model training on Zh-En. This shows that translation degrades the quality of data. There are some exceptions when testing on Korean. Translating the English training data into Chinese, Japanese and Korean still improve the performance on Korean. We also found that when translated into the same language, the English training data is always better than the Chinese data (En-XX v.s. Zh-XX), with only one exception (En-Fr v.s. Zh-Fr when testing on KorQuAD). This may be because we have less Chinese training data than English. These results show that the quality and the size of dataset are much more important than whether the training and testing are in the same language or not.

### 2.3 Discussion

#### 2.3.1 The Effect of Machine Translation

Table 2 shows that fine-tuning on un-translated target language data achieves much better performance than data translated into the target language. Because the above statement is true across all the languages, it is a strong evidence that translation degrades the performance.We notice that the translated corpus and untranslated corpus are not the same. This may be another factor that influences the results. Conducting an experiment between un-translated and back-translated data may deal with this problem.

#### 2.3.2 The Effect of Other Factors

Here we discuss the case that the training data are translated. We consider each result is affected by at least three factors: (1) training corpus, (2) data size, (3) whether the source corpus is translated into the target language. To study the effect of data-size, we conducted an extra experiment where we down-sampled the size of English data to be the same as Chinese corpus, and used the down-sampled corpus to train. Then We carried out one-way ANOVA test and found out the significance of the three factors are ranked as below: (1) $>$ (2) $>>$ (3). The analysis supports that the characteristics of training data is more important than translated into target language or not. Therefore, although translation degrades the performance, whether translating the corpus into the target language is not critical.

| Train | Test | EM | F1 |
|---|---|---|---|
| English | English-permuted | 1.25 | 11.54 |
| English | Chinese-permuted | 5.02 | 17.49 |
| Chinese | Chinese-permuted | 8.91 | 25.67 |
| Chinese | Chinese | 81.24 | 88.68 |

Table 3: EM/F1 scores over artificially created unseen languages (English-permuted and Chinese-permuted).

## 3 What Does Zero-shot Transfer Model Learn?

### 3.1 Unseen Language Dataset

It has been shown that extractive QA tasks like SQuAD may be tackled by some language independent strategies, for example, matching words in questions and context (Weissenborn et al., 2017). Is zero-shot learning feasible because the model simply learns this kind of language independent strategies on one language and apply to the other?

To verify whether multi-BERT largely counts on a language independent strategy, we test the model on the languages unseen during pre-training. To make sure the languages have never been seen before, we artificially make unseen languages by permuting the whole vocabulary of existing languages. That is, all the words in the sentences of a specific language are replaced by other words in the same language to form the sentences in the created unseen language. It is assumed that if multi-BERT used to find answers by language independent strategy, then multi-BERT should also do well on unseen languages. Table 4 shows that the performance of multi-BERT drops drastically on the dataset. It implies that multi-BERT might not totally rely on pattern matching when finding answers.

### 3.2 Embedding in Multi-BERT

PCA projection of hidden representations of the last layer of multi-BERT before and after fine-tuning are shown in Fig. 1. The red points represent Chinese tokens, and the blue points are for English. The results show that tokens from different languages might be embedded into the same

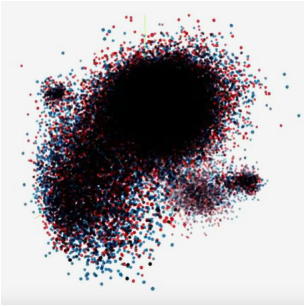

(a) Before Fine-tuning

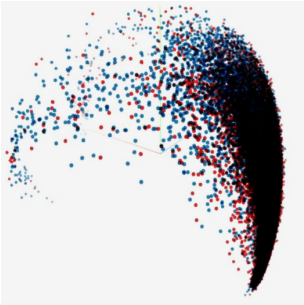

(b) After Fine-tuning

Figure 1: PCA visualization of hidden representations from the 12-th transformer layer of multi-BERT before and after fine-tuning on English. The red points represent Chinese tokens, and the blue points are for English.

| Train | Mix Lang. | EM | F1 | Sub. |
|---|---|---|---|---|
| English | None | 81.17 | 88.63 | 0% |
| English | Chinese | 68.79 | 79.18 | 31% |
| English | French | 65.7 | 77.43 | 61% |
| English | Japanese | 63.32 | 74.06 | 30% |
| English | Korean | 39.93 | 63.46 | 32% |

Table 4: EM/F1 scores on artificial code-switching datasets generated by replacing some of the words in English dataset with synonyms in another languages. (Sub. is the substitution ratio of the dataset)

| Source | Example |
|---|---|
| pred: | second 法 律 of 熱 力 學 (Zh) |
| gt: | second law of thermodynamics |
| pred: | エ レ ク ト リ ッ ク motors (Jp) |
| gt: | electric motors |
| pred: | fermionic nature des lectrons (Fr) |
| gt: | fermionic nature of electrons |
| pred: | the 차이점 in 잠재력 에너지 (Kr) |
| gt: | the difference in potential energy |

Table 5: Answers inferenced on code-switching dataset. The predicted answers would be the same as the ground truths (gt) if we translate every word into English.

space with close spatial distribution. Even though during the fine-tuning only the English data is used, the embedding of the Chinese token changed accordingly. We also quantitatively evaluate the similarities between the embedding of the languages. The results can be found in the Appendix.

### 3.3 Code-switching Dataset

We observe linguistic-agnostic representations in the last subsection. If tokens are represented in a language-agnostic way, the model may be able to handle code-switching data. Because there is no code-switching data for RC, we create artificial code-switching datasets by replacing some of the words in contexts or questions with their synonyms in another language. The synonyms are found by word-by-word translation with given dictionaries. We use the bilingual dictionaries collected and released in facebookresearch/MUSE GitHub repository. We substitute the words if and only if the words are in the bilingual dictionaries.

Table 4 shows that on all the code-switching datasets, the EM/F1 score drops, indicating that the semantics of representations are not totally disentangled from language. However, the examples of the answers of the model (Table 5) show that multi-BERT could find the correct answer spans although some keywords in the spans have been translated into another language.

### 3.4 Typology-manipulated Dataset

There are various types of typology in languages. For example, in English the typology order is subject-verb-object (SVO) order, but in Japanese and Korean the order is subject-object-verb (SOV). We construct a typology-manipulated dataset to examine if the typology order of the training data influences the transfer learning results. If the model only learns the semantic mapping between different languages, changing English typology order from SVO to SOV should improve the transfer ability from English to Japanese. The method used to generate datasets is the same as Ravfogel et al. 2019.

The source code is from a GitHub repository named Shaul1321/rnn_typology, which labels given sentences to CoNLL format with Stanford-CoreNLP and then re-arranges them greedily.

Table 6 shows that when we change the English typology order to SOV or OSV order, the perfor-

| Train/Test | English | Chinese | Korean |
|---|---|---|---|
| En | 81.2/88.6 | 63.3/78.8 | 49.2/69.3 |
| En-SOV | 78.4/86.5 | 62.8/78.3 | 47.6/70.4 |
| En-VOS | 79.4/87.1 | 59.1/74.6 | 46.2/67.0 |
| En-VSO | 79.4/87.1 | 60.9/76.8 | 44.2/65.4 |
| En-OSV | 78.9/86.9 | 63.5/78.0 | 49.0/70.7 |
| En-OVS | 73.6/82.5 | 57.6/72.1 | 45.8/67.0 |

Table 6: EM/F1 scores over artificially created typology-manipulated dataset.

mance on Korean is improved and worsen on English and Chinese, but very slightly. The results show that the typology manipulation on the training set has little influence. It is possible that multi-BERT normalizes the typology order of different languages to some extent.

## 4 Conclusion

In this paper, we systematically explore zero-shot cross-lingual transfer learning on RC with multi-BERT. The experimental results on English, Chinese and Korean corpora show that even when the languages for training and testing are different, reasonable performance can be obtained. Furthermore, we created several artificial data to study the cross-lingual ability of multi-BERT in the presence of typology variation and code-switching. We showed that only token-level pattern matching is not sufficient for multi-BERT to answer questions and typology variation and code-switching only caused minor effects on testing performance.

## References


Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018. A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 789–798, Melbourne, Australia. Association for Computational Linguistics.

Akari Asai, Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2018. Multilingual Extractive Reading Comprehension by Runtime Machine Translation. *arXiv e-prints*, page arXiv:1809.03275.

Christopher Clark and Matt Gardner. 2018. Simple and effective multi-paragraph reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 845–855, Melbourne, Australia. Association for Computational Linguistics.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel R. Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. Xnli: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Wei He, Kai Liu, Jing Liu, Yajuan Lyu, Shiqi Zhao, Xinyan Xiao, Yuan Liu, Yizhong Wang, Hua Wu, Qiaoqiao She, Xuan Liu, Tian Wu, and Haifeng Wang. 2018. DuReader: a Chinese machine reading comprehension dataset from real-world applications. In *Proceedings of the Workshop on Machine Reading for Question Answering*, pages 37–46, Melbourne, Australia. Association for Computational Linguistics.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1693–1701. Curran Associates, Inc.

Daniel Hewlett, Alexandre Lacoste, Llion Jones, Illia Polosukhin, Andrew Fandrianto, Jay Han, Matthew Kelcey, and David Berthelot. 2016. Wikireading: A novel large-scale language understanding task over wikipedia. *CoRR*, abs/1608.03542.

Minghao Hu, Yuxing Peng, Zhen Huang, Xipeng Qiu, Furu Wei, and Ming Zhou. 2018. Reinforced mnemonic reader for machine reading comprehension. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 4099–4106. International Joint Conferences on Artificial Intelligence Organization.

Hsin-Yuan Huang, Chenguang Zhu, Yelong Shen, and Weizhu Chen. 2017. Fusionnet: Fusing via fully-aware attention with application to machine comprehension. *CoRR*, abs/1711.07341.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *CoRR*, abs/1901.07291.

Guillaume Lample, Alexis Conneau, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. In *International Conference on Learning Representations*.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, and Rangan and Majumder. 2016. Ms marco: A human generated machine reading comprehension dataset.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Shauli Ravfogel, Yoav Goldberg, and Tal Linzen. 2019. Studying the inductive biases of rnns with synthetic variations of natural languages. *CoRR*, abs/1903.06400.

Naomi Saphra and Adam Lopez. 2018. Understanding learning dynamics of language models with svcca.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603.

Jooyoul Lee Seungyoung Lim, Myungji Kim. 2018. KorQuAD: Korean QA Dataset for Machine Comprehension.

Chih-Chieh Shao, Trois Liu, Yuting Lai, Yiying Tseng, and Sam Tsai. 2018. DRCD: a chinese machine reading comprehension dataset. *CoRR*, abs/1806.00920.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. NewsQA: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 191–200, Vancouver, Canada. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 189–198, Vancouver, Canada. Association for Computational Linguistics.

Dirk Weissenborn, Georg Wiese, and Laura Seiffe. 2017. Making neural qa as simple as possible but not simpler.

Shijie Wu and Mark Dredze. 2019. Beto, bentz, becas: The surprising cross-lingual effectiveness of BERT. *CoRR*, abs/1904.09077.

Chunting Zhou, Xuezhe Ma, Di Wang, and Graham Neubig. 2019. Density matching for bilingual word embedding. *CoRR*, abs/1904.02343.

# A Supplemental Material

## A.1 Internal Representation of multi-BERT

The architecture of multi-BERT is a Transformer encoder (Vaswani et al., 2017). While fine-tuning on SQuAD-like dataset, the bottom layers of multi-BERT are initialized from Google-pretrained parameters, with an added output layer initialized from random parameters. Tokens representations from the last layer of bottom-part of multi-BERT are inputs to the output layer and then the output layer outputs a distribution over all tokens that indicates the probability of a token being the START/END of an answer span.

### A.1.1 Cosine Similarity

As all translated versions of SQuAD/DRCD are parallel to each other. Given a source-target language pair, we calculate cosine similarity of the mean pooling of tokens representation within corresponding answer-span as a measure of how much they look like in terms of the internal representation of multi-BERT. The results are shown in Fig. 2.

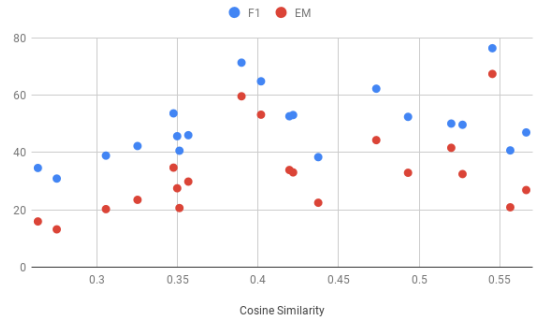


Figure 2: The relation of cosine similarity of answer words with EM/F1 scores in red and blue respectively. Each point represents a source-target language pair of datasets.

### A.1.2 SVCCA

Singular Vector Canonical Correlation Analysis (SVCCA) is a general method to compare the correlation of two sets of vector representations. SVCCA has been proposed to compare learned representations across language models (Saphra and Lopez, 2018). Here we adopt SVCCA to measure the linear similarity of two sets of representations in the same multi-BERT from different translated datasets, which are parallel to each other. The results are shown in Fig 3.

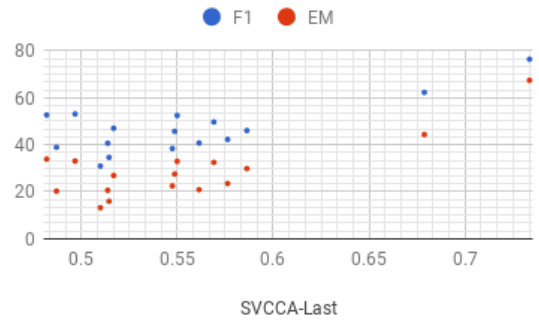


Figure 3: The relation of SVCCA similarity with EM/F1 scores in red and blue respectively. Each point represents a source-target language pair of datasets.

## A.2 Improve Transfering

In the paper, we show that internal representations of multi-BERT are linear-mappable to some extent between different languages. This implies that multi-BERT model might encode semantic and syntactic information in language-agnostic ways and explains how zero-shot transfer learning could be done.

To take a step further, while transfering model from source dataset to target dataset, we align representations in two proposed way, to improve performance on target dataset.

### A.2.1 Linear Mapping Method

Algorithms proposed in (Lample et al., 2018; Artetxe et al., 2018; Zhou et al., 2019) to unsupervisedly learn linear mapping between two sets of embeddings are used here to align representations of source (training data) to those of target. We obtain the mapping generated by embeddings from one specific layer of pre-trained multi-BERT then we apply this mapping to transform the internal representations of multi-BERT while fine-tuning on training data.

### A.2.2 Adversarial Method

In Adversarial Method, we add an additional transform layer to transform representations and a discrimination layer to discriminate between transformed representations from source language (training set) and target language (development set). And the GAN loss is applied in the total loss of fine-tuning.

### A.2.3 Discussion

As table 7 shows, there are no improvements among above methods. Some linear mapping

| Approach | EM | F1 |
|---|---|---|
| MUSE(Lample et al., 2018) | 33.03 | 49.48 |
| DeMa(Zhou et al., 2019) | 55.64 | 72.59 |
| Vecmap(Artetxe et al., 2018) | 14.05 | 24.83 |
| GAN-layer 8 | 54.26 | 71.04 |
| GAN-layer 11 | 60.47 | 76.14 |

Table 7: EM/F1 scores on DRCD dev-set.

methods even causes devastating effect on EM/F1 scores.

# QUARTZ: An Open-Domain Dataset of Qualitative Relationship Questions

**Oyvind Tafjord, Matt Gardner, Kevin Lin, Peter Clark**
Allen Institute for Artificial Intelligence, Seattle, WA
{oyvindt,mattg,kevinl peterc}@allenai.org

## Abstract

We introduce the first open-domain dataset, called QUARTZ, for reasoning about textual qualitative relationships. QUARTZ contains general qualitative statements, e.g., "A sunscreen with a higher SPF protects the skin longer.", twinned with 3864 crowdsourced situated questions, e.g., "Billy is wearing sunscreen with a lower SPF than Lucy. Who will be best protected from the sun?", plus annotations of the properties being compared. Unlike previous datasets, the general knowledge is textual and not tied to a fixed set of relationships, and tests a system's ability to comprehend and apply textual qualitative knowledge in a novel setting. We find state-of-the-art results are substantially (20%) below human performance, presenting an open challenge to the NLP community.

## 1 Introduction

Understanding and applying qualitative knowledge is a fundamental facet of intelligence. For example, we may read that exercise improves health, and thus decide to spend more time at the gym; or that larger cars cause more pollution, and thus decide to buy a smaller car to be environmentally sensitive. These skills require understanding the underlying qualitative relationships, and being able to apply them in specific contexts.

To promote research in this direction, we present the first *open-domain* dataset of qualitative relationship questions, called QUARTZ ("Qualitative Relationship Test set")[1]. Unlike earlier work in qualitative reasoning, e.g., (Tafjord et al., 2019), the dataset is not restricted to a small, fixed set of relationships. Each question $Q_i$ (2-way multiple choice) is grounded in a particular situation, and is paired with a sentence $K_i$ expressing the general qualitative knowledge needed to answer it.

[1] Available at http://data.allenai.org/quartz/

**Q:** If Mona lives in a city that begins producing a greater amount of pollutants, what happens to the air quality in that city? (A) increases (B) decreases **[correct]**

**K:** More pollutants mean poorer air quality.

**Annotations:**

**Q:** [**MORE**, "greater", "amount of pollutants"]
→ (A) [**MORE**, "increases", "air quality"]
(B) [**LESS**, "decreases", "air quality"]

**K:** [**MORE**, "more", "pollutants"]
↔ [**LESS**, "poorer", "air quality"]

Figure 1: QUARTZ contains situated qualitative questions, each paired with a gold background knowledge sentence and qualitative annotations.

$Q_i$ and $K_i$ are also annotated with the properties being compared (Figure 1). The property annotations serve as supervision for a potential semantic parsing based approach. The overall task is to answer the $Q_i$ given the corpus $K = \{K_i\}$.

We test several state-of-the-art (BERT-based) models and find that they are still substantially (20%) below human performance. Our contributions are thus (1) the dataset, containing 3864 richly annotated questions plus a background corpus of 400 qualitative knowledge sentences; and (2) an analysis of the dataset, performance of BERT-based models, and a catalog of the challenges it poses, pointing the way towards solutions.

## 2 Related Work

Despite rapid progress in general question-answering (QA), e.g., (Clark and Gardner, 2018), and formal models for qualitative reasoning (QR), e.g., (Forbus, 1984; Weld and De Kleer, 2013), there has been little work on reasoning with *textual* qualitative knowledge, and no dataset available in this area. Although many datasets include a few qualitative questions, e.g., (Yang et al., 2018; Clark et al., 2018), the only one directly probing



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5941–5946, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| | |
|---|---|
| **Differing Comparatives:** | |
| $Q_1$ | Jan is comparing stars, specifically a small star and the larger Sun. Given the size of each, Jan can tell that the Sun puts out heat that is (A) greater (B) lesser |
| $K_1$ | Bigger stars produce more energy, so their surfaces are hotter. |
| **Discrete Property Values:** | |
| $Q_2$ | What happens to a light car when it has the same power as a heavy car? (A) accelerates faster (B) accelerates slower |
| $K_2$ | The smaller its mass is, the greater its acceleration for a given amount of force. |
| **Numerical Property Values:** | |
| $Q_3$ | Will found water from a source 10m from shore. Eric found water from a source 2m from shore. Whose water likely contains the least nutrients? (A) Will's (B) Eric's |
| $K_3$ | Most nutrients are washed into ocean water from land. Therefore, water closer to shore tends to have more nutrients. |
| **Commonsense Knowledge:** | |
| $Q_4$ | Compared to a box of bricks a box of feathers would be (A) lighter (B) heavier |
| $K_4$ | A given volume of a denser substance is heavier than the same volume of a less dense substance. |
| **Multiple Entities ("Worlds"):** | |
| $Q_5$ | Jimbo liked to work out, while James never did. Which person would have weaker muscles? (A) Jimbo (B) James |
| $K_5$ | Muscles that are exercised are bigger and stronger than muscles that are not exercised. |
| **Complex Stories:** | |
| $Q_6$ | NASA has sent an unmanned probe to survey a distant solar system with four planets. Planet Zorb is farthest from the sun of this solar system, Planet Krakatoa is second farthest, Planet Beanbag is third, and Krypton is the closest. The probe visits the planets in order, first Zorb, then Krakatoa, then Beanbag and finally Krypton. Did the probe have to fly farther in its trip from (A) Zorb to Krakatoa or (B) from Beanbag to Krypton? |
| $K_6$ | In general, the farther away from the Sun, the greater the distance from one planets orbit to the next. |

Table 1: Examples of crowdsourced questions Q and corpus knowledge K in QUARTZ, illustrating phenomena.

QR is QuaRel (Tafjord et al., 2019). However, although QuaRel contains 2700 qualitative questions, its underlying qualitative knowledge was specified formally, using a small, fixed ontology of 19 properties. As a result, systems trained on QuaRel are limited to only questions about those properties. Likewise, although the QR community has performed some work on extracting qualitative models from text, e.g., (McFate et al., 2014; McFate and Forbus, 2016), and interpreting questions about identifying qualitative processes, e.g., (Crouse et al., 2018), there is no dataset available for the NLP community to study textual qualitative reasoning. QUARTZ addresses this need.

## 3 The Task

Examples of QuaRTz questions $Q_i$ are shown in Table 1, along with a sentence $K_i$ expressing the relevant qualitative relationship. The QUARTZ task is to answer the questions given a small (400 sentence) corpus $K$ of general qualitative relationship sentences. Questions are crowdsourced, and the sentences $K_i$ were collected from a larger corpus, described shortly.

Note that the task involves substantially more than matching intensifiers (more/greater/...) between $Q_i$ and $K_i$. Answers also require some qualitative reasoning, e.g., if the intensifiers are inverted in the question, and entity tracking, to keep track of which entity an intensifier applies to. For example, consider the qualitative sentence and three questions (correct answers bolded):

$K_n$: People with greater height are stronger.
$Q_n$: Sue is taller than Joe so Sue is (A) **stronger** (B) weaker
$Q'_n$: Sue is shorter than Joe so Sue is (A) stronger (B) **weaker**
$Q''_n$: Sue is shorter than Joe so Joe is (A) **stronger** (B) weaker

$Q'_n$ requires reasoning about intensifers that are flipped with respect to $K$ (shorter $\rightarrow$ weaker), and $Q''_n$ requires entities be tracked correctly (asking about Sue or Joe changes the answer).

## 4 Dataset Collection

QUARTZ was constructed as follows. First, 400 sentences[2] expressing general qualitative relations were manually extracted by the authors from a large corpus using keyword search ("increase", "faster", etc.). Examples ($K_i$) are in Table 1.

Second, crowdworkers were shown a seed sentence $K_i$, and asked to annotate the two properties

[2] In a few cases, a short paragraph rather than sentence was selected, where surrounding context was needed to make sense of the qualitative statement.

being compared using the template below, illustrated using $K_2$ from Table 1:

"The smaller its mass is, the greater its acceleration for a given amount of force."

What is being compared?

Less ▾ mass More ▾ acceleration

They were then asked to author a situated, 2-way multiple-choice (MC) question that tested this relationship, guided by multiple illustrations. Examples of their questions ($Q_i$) are in Table 1.

Third, a second set of workers was shown an authored question, asked to validate its answer and quality, and asked to annotate how the properties of $K_i$ identified earlier were expressed. To do this, they filled a second template, illustrated for $Q_2$:

| | Appears in question as: | |
|---|---|---|
| Relation: | More/Less phrase: | Changed property: |
| **MORE mass**: | - | heavy |
| **LESS mass**: | - | light |
| **MORE acceleration**: | faster | accelerates |
| **LESS acceleration**: | slower | accelerates |

Finally these workers were asked to generate a new question by "flipping" the original so the answer switched. Flipping means inverting comparatives (e.g., "more" → "less"), values, and other edits as needed to change the answer, e.g.,

K: *More rain causes damper surfaces.*

Q:*More rain causes (A) wetter land (B) drier land*

Q-flipped: **Less** *rain causes (A) wetter land (B) drier land*

Flipped questions are created to counteract the tendency of workers to use the same comparison direction (e.g., "more") in their question as in the seed sentence $K_i$, potentially answerable by simply matching $Q_i$ and $K_i$. Flipped questions are more challenging as they demand more qualitative reasoning (Section 7.1).

Questions marked by workers as poor quality were reviewed by the authors and rejected/modified as appropriate. The dataset was then split into train/dev/test partitions such that questions from the same seed $K_i$ were all in the same partition. Statistics are in Table 2.

To determine if the questions are correct and answerable given the general knowledge, a human baseline was computed. Three annotators independently answered a random sample of 100 questions given the supporting sentence $K_i$ for each. The mean score was 95.0%.

| | |
|---|---|
| # questions $Q_i$ | 3864 |
| flip/no flip | 1932/1932 |
| positive/negative qualitative influence (QR+/QR-) | 2772/1092 |
| train/dev/test | 2696/384/784 |
| av $Q_i$ length (sents) min/avg/max | 1/1.5/6 |
| av $K_i$ length (sents) min/avg/max | 1/1.1/4 |

Table 2: Statistics of QUARTZ.

## 5 Models

The QUARTZ task is to answer the questions given the corpus $K$ of qualitative background knowledge. We also examine a "no knowledge" (questions only) task and a "perfect knowledge" task (each question paired with the qualitative sentence $K_i$ it was based on). We report results using two baselines and several strong models built with BERT-large (Devlin et al., 2019) as follows:

1. **Random:** always 50% (2-way MC).
2. **BERT-Sci:** BERT fine-tuned on a large, general set of science questions (Clark et al., 2018).
3. **BERT (IR):** This model performs the full task. First, a sentence $K_i$ is retrieved from $K$ using $Q_i$ as a search query. This is then supplied to BERT as *[CLS] $K_i$ [SEP] question-stem [SEP] answer-option [SEP]* for each option. The [CLS] output token is projected to a single logit and fed through a softmax layer across answer options, using cross entropy loss, the highest being selected. This model is fine-tuned using QUARTZ (only).
4. **BERT (IR upper bound):** Same, but using the ideal (annotated) $K_i$ rather than retrieved $K_i$.
5. **BERT-PFT (no knowledge):** BERT first fine-tuned ("pre-fine-tuned") on the RACE dataset (Lai et al., 2017; Sun et al., 2019), and then fine-tuned on QUARTZ (questions only, no $K$, both train and test). Questions are supplied as *[CLS] question-stem [SEP] answer-option [SEP]*.
6. **BERT-PFT (IR):** Same as BERT (IR), except starting with the pre-fine-tuned BERT.

All models were implemented using AllenNLP (Gardner et al., 2018).

## 6 Results

The results are shown in Table 3, and provide insights into both the data and the models:

1. **The dataset is hard.** Our best model, BERT-PFT (IR), scores only 73.7, over 20 points behind human performance (95.0), suggesting there are significant linguistic and semantic challenges to overcome (Section 7).

| Questions→<br>Model ↓ | All Test | No-flip only | Flip only |
|---|---|---|---|
| **Baselines:** | | | |
| Random | 50.0 | 50.0 | 50.0 |
| BERT-Sci | 54.6 | 76.0 | 33.2 |
| **Models:** | | | |
| BERT (IR) | 64.4 | 66.3 | 62.5 |
| (BERT IR upper bound) | (67.7) | (68.1) | (67.3) |
| BERT-PFT (no knowledge) | 68.8 | 70.4 | 67.1 |
| BERT-PFT (IR) | **73.7** | 77.3 | 70.2 |
| (BERT-PFT IR upper bound) | (79.8) | (82.1) | (77.6) |
| Human | 95.0 | | |

Table 3: Performance of various models on QUARTZ.

2. **A general science-trained QA system has not learned this style of reasoning.** BERT-Sci only scores 54.6, just a little above random (50.0).

3. **Pre-Fine-Tuning is important.** Fine-tuning only on QUARTZ does signficantly worse (64.4) than pre-fine-tuning on RACE before fine-tuning on QUARTZ (73.7). Pre-fine-tuning appears to teach BERT something about multiple choice questions in general, helping it more effectively fine-tune on QUARTZ.

4. **BERT already "knows" some qualitative knowledge.** Interestingly, BERT-PFT (no knowledge) scores 68.8, significantly above random, suggesting that BERT already "knows" some kind of qualitative knowledge. To rule out annotation artifacts, we we experimented with balancing the distributions of positive and negative influences, and different train/test splits to ensure no topical overlap between train and test, but the scores remained consistent.

5. **BERT can apply general qualitative knowledge to QA, but only partially.** The model for the full task, BERT-PFT (IR) outperforms the no knowledge version (73.7, vs. 68.8), but still over 20 points below human performance. Even given the ideal knowledge (IR upper bound), it is still substantially behind (at 79.8) human performance. This suggests more sophisticated ways of training and/or reasoning with the knowledge are needed.

## 7 Discussion and Analysis

### 7.1 Qualitative Reasoning

Can models learn qualitative reasoning from QUARTZ? While QUARTZ questions do not require chaining, 50% involve "flipping" a qualitative relationship (e.g., K: "more X → more Y", Q: "Does less X → less Y?"). Training on just the original crowdworkers' questions, where they chose to flip the knowledge only 10% of the time, resulted in poor (less than random) performance on all the flipped questions. However, training on full QUARTZ, where no-flip and flip were balanced, resulted in similar score for both types of question, suggesting that such a reasoning capability can indeed be learned.

| **Comparatives:** |
|---|
| "warmer" ↔ "increase temperature" |
| "more difficult" ↔ "slower" |
| "need more time" ↔ "have lesser amount" |
| "decreased distance" ↔ "hugged" |
| "cost increases" ↔ "more costly" |
| "increase mass" ↔ "add extra" |
| "more tightly packed" ↔ "add more" |
| **Commonsense Knowledge:** |
| "more land development" ↔ "city grow larger" |
| "not moving" ↔ "sits on the sidelines" |
| "caught early" ↔ 'sooner treated" |
| "lets more light in" ↔ "get a better picture" |
| "stronger electrostatic force" ↔ "hairs stand up more" |
| "less air pressure" ↔ "more difficult to breathe" |
| "more photosynthesis" ↔ "increase sunlight" |
| **Discrete Values:** |
| "stronger acid" ↔ "vinegar" vs. "tap water" |
| "more energy" ↔ "ripple" vs. "tidal wave" |
| "closer to Earth" ↔ "ball on Earth" vs. "ball in space" |
| "mass" ↔ "baseball" vs. "basketball" |
| "rougher" ↔ "notebook paper" vs. "sandpaper" |
| "heavier" ↔ "small wagon" vs. "eighteen wheeler" |

Table 4: Examples of linguistic and semantic gaps between knowledge $K_i$ (left) and question $Q_i$ (right). A system needs to bridge such gaps for high performance.

### 7.2 Linguistic Phenomena

From a detailed analysis of 100 randomly sampled questions, the large majority (86%) involved the (overlapping) linguistic and semantic phenomena below, and illustrated in Tables 1 and 4:

1. **Differing comparative expressions** (≈68%) between $K_i$ and $Q_i$ occur in the majority of questions, e.g.,
   *"increased altitude" ↔ "higher"*
2. **Indirection and Commonsense knowledge** (≈35%) is needed for about 1/3 of the questions to relate $K$ and $Q$, e.g.,
   *"higher temperatures" ↔ "A/C unit broken"*
3. **Multiple Worlds** (≈26%): 1/4 of the questions explicitly mention *both* situations being compared, e.g., $Q_1$ in Table 1. Such questions are known to be difficult because models can easily confuse the two situations (Tafjord et al., 2019).
4. **Numerical property values** (≈11%) require numeric comparison to identify the qualitative relationship, e.g., that "60 years" is older than "30 years".

5. **Discrete property values** (≈7%), often require commonsense to compare, e.g., that a "melon" is larger than an "orange".
6. **Stories** (≈15%): 15% of the questions are 3 or more sentences long, making comprehension more challenging.

This analysis illustrates the richness of linguistic and semantic phenomena in QUARTZ.

### 7.3 Use of the Annotations

QUARTZ includes a rich set of annotations on all the knowledge sentences and questions, marking the properties being compared, and the linguistic and semantic comparatives employed (Figure 1). This provides a laboratory for exploring semantic parsing approaches, e.g., (Berant et al., 2013; Krishnamurthy et al., 2017), where the underlying qualitative comparisons are extracted and can be reasoned about.

## 8 Conclusion

Understanding and applying textual qualitative knowledge is an important skill for question-answering, but has received limited attention, in part due the lack of a broad-coverage dataset to study the task. QUARTZ aims to fill this gap by providing the first *open-domain* dataset of qualitative relationship questions, along with the requisite qualitative knowledge and a rich set of annotations. Specifically, QuaRTz removes the requirement, present in all previous qualitative reasoning work, that a fixed set of qualitative relationships be formally pre-specified. Instead, QuaRTz tests the ability of a system to find and apply an arbitrary relationship on the fly to answer a question, including when simple reasoning (arguments, polarities) is required.

As the QUARTZ task involves using a general corpus $K$ of textual qualitative knowledge, a high-performing system would be close to a fully general system where $K$ was much larger (e.g., the Web or a filtered subset), encompassing many more qualitative relationships, and able to answer arbitrary questions of this kind. Scaling further would thus require more sophisticated retrieval over a larger corpus, and (sometimes) chaining across influences, when a direct connection was not described in the corpus. QUARTZ thus provides a dataset towards this end, allowing controlled experiments while still covering a substantial number of textual relations in an open setting. QuaRTz is available at http://data.allenai.org/quartz/.

### Acknowledgements

We are grateful to the AllenNLP and Beaker teams at AI2, and for the insightful discussions with other Aristo team members. Computations on beaker.org were supported in part by credits from Google Cloud.

## References

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *EMNLP'13*.

Christopher Clark and Matt Gardner. 2018. Simple and effective multi-paragraph reading comprehension. In *ACL*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

M. Crouse, C. McFate, and K. Forbus. 2018. Learning to build qualitative scenario models from natural language. In *Proc. 31st Int. Workshop on Qualitative Reasoning (QR'18)*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Kenneth D. Forbus. 1984. Qualitative process theory. *Artificial Intelligence*, 24:85–168.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. In *NLP OSS Workshop at ACL*. (arXiv:1803.07640).

Jayant Krishnamurthy, Pradeep Dasigi, and Matthew Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *EMNLP'17*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. In *EMNLP*.

Clifton McFate and Kenneth Forbus. 2016. Scaling up linguistic processing of qualitative processes. In *Proc. 4th Ann. Conf. on Advances in Cognitive Systems*.

Clifton James McFate, Kenneth D Forbus, and Thomas R Hinrichs. 2014. Using narrative function to extract qualitative information from natural language texts. In *AAAI'14*.

Kai Sun, Dian Yu, Dong Yu, and Claire Cardie. 2019. Improving machine reading comprehension with general reading strategies. In *NAACL*.

Oyvind Tafjord, Peter Clark, Matt Gardner, Wen-tau Yih, and Ashish Sabharwal. 2019. Quarel: A dataset and models for answering questions about qualitative relationships. In *AAAI*.

Daniel S Weld and Johan De Kleer. 2013. *Readings in qualitative reasoning about physical systems*. Morgan Kaufmann.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*.

# Giving BERT a Calculator: Finding Operations and Arguments with Reading Comprehension

**Daniel Andor, Luheng He, Kenton Lee, Emily Pitler**
Google Research
{andor, luheng, kentonl, epitler}@google.com

## Abstract

Reading comprehension models have been successfully applied to extractive text answers, but it is unclear how best to generalize these models to abstractive numerical answers. We enable a BERT-based reading comprehension model to perform lightweight numerical reasoning. We augment the model with a predefined set of executable 'programs' which encompass simple arithmetic as well as extraction. Rather than having to learn to manipulate numbers directly, the model can pick a program and execute it. On the recent Discrete Reasoning Over Passages (DROP) dataset, designed to challenge reading comprehension models, we show a 33% absolute improvement by adding shallow programs. The model can learn to predict new operations when appropriate in a math word problem setting (Roy and Roth, 2015) with very few training examples.

## 1 Introduction

End-to-end reading comprehension models have been increasingly successful at extractive question answering. For example, performance on the SQuAD 2.0 (Rajpurkar et al., 2018) benchmark has improved from 66.3 F1 to 89.5[1] in a single year. However, the Discrete Reasoning Over Passages (DROP) (Dua et al., 2019) dataset demonstrates that as long as there is quantitative reasoning involved, there are plenty of relatively straightforward questions that current extractive QA systems find difficult to answer. Other recent work has shown that even state-of-the-art neural models struggle with numerical operations and quantitative reasoning when trained in an end-to-end manner (Saxton et al., 2019; Ravichander et al., 2019). In other words, even BERT (Devlin et al., 2019) is not very good at doing simple calculations.

[1] https://rajpurkar.github.io/SQuAD-explorer/

| *How many more Chinese nationals are there than European nationals?* |
|---|
| The city of Bangkok has a population of 8,280,925 ...the census showed that it is home to 81,570 Japanese and **55,893** Chinese nationals, as well as 117,071 expatriates from other Asian countries, **48,341** from Europe, 23,418 from the Americas,... |
| **NAQANet**: −55893<br>**Ours**: `Diff`(55893, 48341) = **7552** |

Table 1: Example from the DROP development set. The correct answer is not explicitly stated in the passage and instead must be computed. The NAQANet model[2](Dua et al., 2019) predicts a negative number of people, whereas our model predicts that an operation `Diff` should be taken and identifies the two arguments.

In this work, we extend an extractive QA system with numerical reasoning abilities. We do so by asking the neural network to synthesize small programs that can be executed. The model picks among simple programs of the form `Operation`(args, ...), where the possible operations include span extraction, answering yes or no, and arithmetic. For math operations, the arguments are pointers to numbers in the text and, in the case of composition, other operations. In this way, the burden of actually doing the computation is offloaded from the neural network to a calculator tool. The program additionally provides a thin layer of interpretability that mirrors some of the reasoning required for the answer. For example, in Table 1, the model predicts subtraction (`Diff`) over two numbers in the passage, and executes it to produce the final answer.

We start with a simple extractive question answering model based on BERT (Devlin et al., 2019), and show the following:

1. Predicting unary and binary math operations

[2] https://demo.allennlp.org/reading-comprehension/NzQwNjg1



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5947–5952, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

with arguments resulted in significant improvements on the DROP dataset.

2. Our model can smoothly handle more traditional reading comprehension inputs as well as math problems with new operations. Co-training with the CoQA (Reddy et al., 2018) dataset improved performance on DROP. The DROP+CoQA trained model had never seen multiplication or division examples, but can learn to predict these two ops when appropriate in a math word problem setting (Roy and Roth, 2015) with very few training examples.

## 2 Background and Related Work

**Discrete Reasoning over Paragraphs (DROP)** (Dua et al., 2019) is a reading comprehension task that requires discrete reasoning. Inspired by semantic parsing tasks where models need to produce executable 'programs', it keeps the open-domain nature of reading comprehension tasks such as SQuAD 2.0 (Rajpurkar et al., 2018). As shown in Table 1, the system needs to perform fuzzy matching between *"from Europe"* and *"European nationals"* in order to identify the arguments.

**Numerically-aware QANet (NAQANet)** (Dua et al., 2019) is the current state-of-the-art[3] system for DROP. It extends the QANet model (Yu et al., 2018) with predictions for numbers (0–9) and summation operations. For the latter, it performs a 3-way classification (plus, minus, and zero) on all the numbers in the passage.

While certain binary operations are expressible efficiently with flat sign prediction, it is difficult to generalize the architecture. Moreover, each number is tagged independently, which can cause global inconsistencies; for instance, in Table 1 it assigns a single minus label and no plus labels, leading to a prediction of negative people.

**Mathematical Word Problems** have been addressed with a wide variety of datasets and approaches; see Zhang et al. (2018) for an overview. One such dataset of arithmetic problems is the Illinois dataset (Roy and Roth, 2015). The problems are posed in simple natural language that has a specific, narrow domain, For example: *"If there are 7 bottle caps in a box and Linda puts 7 more bottle caps inside, how many bottle caps are in the box?"*. Unlike DROP, the problems are typically 1–3 sentences long and do not require reading complex passages. Instead, the main challenge is mathematical reasoning. According to Zhang et al. (2018), the current state of the art uses syntactic parses and deterministic rules to convert the input to logical forms (Liang et al., 2016).

## 3 Model

We extend a BERT-based extractive reading comprehension model with a lightweight extraction and composition layer. For details of the BERT architecture see Devlin et al. (2019). We only rely on the representation of individual tokens that are jointly conditioned on the given question $Q$ and passage $P$. Our model predicts an answer by selecting the top-scoring *derivation* (i.e. program) and executing it.

**Derivations** We define the space of possible derivations $\mathcal{D}$ as follows:

- *Literals*: $\{\text{YES}, \text{NO}, \text{UNKNOWN}, 0, \ldots 9\}$.
- *Numerical operations*: including various types of numerical compositions of numbers[4], such as `Sum` or `Diff`.
- *Text spans*: composition of tokens into text spans up to a pre-specified length.
- *Composition of compositions*: we only consider two-step compositions, including merging text spans and nested summations.

The full set of operations are listed in Table 2. For example, `Sum` is a numerical operation that adds two numbers and produces a new number. While we could recursively search for compositions with deep derivations, here we are guided by what is required in the DROP data and simplify inference by heavily restricting multi-step composition. Specifically, spans can be composed into a pair of merged spans (`Merge`), and the sum of two numbers (`Sum`) can subsequently be summed with a third (`Sum3`). The results in Table 3 show the dev set oracle performance using these shallow derivations, by answer type.

**Representation and Scoring** For each derivation $d \in \mathcal{D}$, we compute a vector representation $\mathbf{h}_d$ and a scalar score $\rho(d, P, Q)$ using the BERT output vectors. The scores $\rho$ are used for computing the probability $P(d \mid P, Q)$ as well as for pruning. For brevity, we will drop the dependence on $P$ and $Q$ in this section.

[3] https://leaderboard.allenai.org/drop/submissions/public

[4] Numbers are heuristically extracted from the text.

| | Derivations | Example Question | Answer Derivation |
|---|---|---|---|
| *Literals* | YES, NO, UNKNOWN, 0, 1 ..., 9 | How many field goals did Stover kick? | 4 |
| *Numerical* | `Diff100` : $n_0 \rightarrow 100 - n_1$ | How many percent of the national population does not live in Bangkok? | $100 - 12.6 = 87.4$ |
| | `Sum` : $n_0, n_1 \rightarrow n_0 + n_1$ as well as: `Diff`, `Mul`, `Div` | How many from the census were in Ungheni and Cahul? | $32,828 + 28,763 = 61591$ |
| *Text spans* | `Span` : $i, j \rightarrow s$ | Does Bangkok have more Japanese or Chinese nationals? | "Japanese" |
| *Compositions* | `Merge` : $s_0, s_1 \rightarrow \{s_0, s_1\}$ | What languages are spoken by more than 1%, but fewer than 2% of Richmond's residents? | "Hmong-Mien languages", "Laotian" |
| | `Sum3` : $n_0, n_1, n_2 \rightarrow (n_0 + n_1) + n_2$ | How many residents, in terms of percentage, speak either English, Spanish, or Tagalog? | $\texttt{Sum}(64.56, 23.13) + 2.11 = 89.8$ |

Table 2: Operations supported by the model. $s, n$ refer to arguments of type *span* and *number*, respectively. $i, j$ are the start and end indices of span $s$. The omitted definitions of `Diff`, `Mul`, and `Div` are analogous to `Sum`.

*Literals* are scored as $\rho(d) = \mathbf{w}_d^\intercal \text{MLP}_{\text{lit}}(\mathbf{h}_{\texttt{CLS}})$, where $\mathbf{h}_{\texttt{CLS}}$ is the output vector at the `[CLS]` token of the BERT model (Devlin et al., 2019).

*Numeric operations* use the vector representations $\mathbf{h}_i$ of the first token of each numeric argument. Binary operations are represented as

$$\mathbf{h}_d = \text{MLP}_{\text{binary}}(\mathbf{h}_i, \mathbf{h}_j, \mathbf{h}_i \circ \mathbf{h}_j) \quad (1)$$

and scored as $\rho(d) = \mathbf{w}_{op}^\intercal \mathbf{h}_d$, where $\mathbf{h}_d$ represents the binary arguments and *op* is the operation type. $\circ$ is the Hadamard product. Unary operations such as `Diff100` are scored as $\mathbf{w}_{op}^\intercal \text{MLP}_{\text{unary}}(\mathbf{h}_i)$.

*Text spans* are scored as if they were another binary operation taking as arguments the start and end indices $i$ and $j$ of the span (Lee et al., 2017):

$$\mathbf{h}_d = \text{MLP}_{\text{span}}(\mathbf{h}_i, \mathbf{h}_j) \quad (2)$$

and scored as $\rho(d) = \mathbf{w}_{\text{span}}^\intercal \mathbf{h}_d$.

*Compositions of compositions* are scored with the vector representations of its children. For example, the ternary `Sum3`, comprising a `Sum` and a number, is scored with $\mathbf{w}_{\texttt{Sum3}}^\intercal \text{MLP}_{\texttt{Sum3}}(\mathbf{h}_{d0}, \mathbf{h}_k)$, where $\mathbf{h}_{d0}$ corresponds to the representation from the first `Sum`, and $\mathbf{h}_k$ is the representation of the third number. The *composition* of two spans is scored as $\mathbf{w}_{\texttt{Merge}}^\intercal \text{MLP}_{\texttt{Merge}}(\mathbf{h}_{d0}, \mathbf{h}_{d1}, \mathbf{h}_{d0} \circ \mathbf{h}_{d1})$, where $\mathbf{h}_{d0}$ and $\mathbf{h}_{d1}$ are span representations from (2). The intuition for including $\mathbf{h}_{d0} \circ \mathbf{h}_{d1}$ is that it encodes span similarity, and spans with similar types are more likely to be merged.

This strategy differs from the NAQANet baseline in a few ways. One straightforward difference is that we use BERT as the base encoder rather than QANet. A more meaningful difference is that we model all derivations in the unified op scoring framework described above, which allows generalizing to new operations, whereas NAQANet would require more large-scale changes to go beyond addition and subtraction. Generalizing the model to new ops is a case of extending the derivations and scoring functions. In Section 4, we will show the impact of incrementally adding `Diff100`, `Sum3`, and `Merge`.

### 3.1 Training

We used exhaustive pre-computed oracle derivations $\mathcal{D}^*$ following Dua et al. (2019). We marginalized out all derivations $d^*$ that lead to the answer[5] and minimized:

$$\mathcal{J}(P, Q, \mathcal{D}^*) = -\log \sum_{d^* \in \mathcal{D}^*} P(d^* \mid P, Q)$$

$$P(d \mid P, Q) = \frac{\exp \rho(d, P, Q)}{\sum_{d'} \exp \rho(d', P, Q)}$$

If no derivation lead to the gold answer ($\mathcal{D}^*$ is empty), we skipped the example.

**Pruning** During inference, the `Merge` and `Sum3` operations are composed from the results of `Span` and `Sum` operations, respectively. The space of possible results of `Merge` is quadratic in the number $|\mathcal{S}|$ of possible spans. With $|\mathcal{S}| \sim 10^4$, the complete set of `Merge` instances becomes overwhelming. Similarly, with $|\mathcal{N}| \sim 100$ numbers in each passage, there are millions of possible `Sum3` derivations. To do training and inference efficiently, we kept only the top 128 `Span` and `Sum` results when computing `Merge` and `Sum3`.[6]

[5] In practice we capped the number of derivations at 64, which covers 98.7% of the training examples.

[6] During training, the pruned arguments had recall of 80–90% after 1 epoch and plateaued at 95–98%.

| | *Oracle* | Overall Dev | | Overall Test | | Date (1.6%) | | Number (62%) | | Span (32%) | | Spans (4.4%) | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Dev EM | EM | F1 | EM | F1 | EM | F1 | EM | F1 | EM | F1 | EM | F1 |
| NAQANet | | 46.75 | 50.39 | 44.24 | 47.77 | 32.0 | 39.6 | 44.9 | 45.0 | 58.2 | 64.8 | 0.0 | 27.3 |
| Our basic[7] | *80.03* | 66.50 | 69.91 | - | - | 57.0 | 65.1 | 65.8 | 66.1 | 78.0 | 82.6 | 0.0 | 35.7 |
| +`Diff100` | *88.75* | 75.52 | 78.82 | - | - | 53.6 | 61.3 | 80.3 | 80.5 | 78.4 | 82.8 | 0.0 | 35.8 |
| +`Sum3` | *90.16* | 76.70 | 80.06 | - | - | 58.0 | 64.6 | 81.9 | 82.1 | 78.9 | 83.4 | 0.0 | 36.0 |
| +`Merge` | *93.01* | 76.95 | 80.48 | - | - | 58.1 | 61.8 | 82.0 | 82.1 | 78.8 | 83.4 | 5.1 | 45.0 |
| +CoQA | *93.01* | **78.00** | **81.56** | **76.93** | **80.47** | **59.5** | **66.4** | **83.0** | **83.2** | **79.8** | **84.2** | **5.8** | **46.8** |
| +Ensemble | *93.01* | **78.95** | **82.54** | **78.15** | **81.78** | **59.7** | **67.7** | **83.9** | **84.1** | **81.2** | **85.5** | 5.4 | 46.5 |
| *Oracle* | | *93.01* | | | | *71.6* | | *94.5* | | *95.8* | | *60.5* | |

Table 3: Accuracies on the DROP dev and test set in terms of exact match (EM) and token-level F1. The right-hand columns show the performance breakdown with different answer types on the development set. The largest improvements come from *Date*, *Number*, and *Spans* (answers with multiple spans). *Oracle* rows and columns indicate the performance that could be achieved by perfect selections of derivations. The ensemble used 6 models.

**Spurious ambiguities** Of the answers for which we could find at least one oracle derivation, 36% had two or more alternatives. During training, the model became effective at resolving many of these ambiguities. We monitored the entropy of $P(d^* \mid P, Q)$ for the ambiguous examples as training progressed. At the start, the entropy was 2.5 bits, which matches the average ambiguous oracle length of $\sim$ 6 alternatives. By the end of 4 epochs, the average entropy had dropped to $< 0.2$ bits, comparable to a typical certainty of 95–99% that one of the derivations is the correct one.

## 4 Experiments

Our main experiments pertain to DROP (Dua et al., 2019), using DROP and, optionally, CoQA (Reddy et al., 2018) data for training. Preprocessing and hyperparameter details are given in the supplementary material. In addition to full DROP results, we performed ablation experiments for the incremental addition of the `Diff100`, `Sum3`, and `Merge` operations, and finally the CoQA training data. We ran on the CoQA dev set, to show that the model co-trained on CoQA can still perform traditional reading comprehension. To investigate our model's ability to do symbolic reasoning at the other extreme, we performed few-shot learning experiments on the Illinois dataset of math problems (Roy and Roth, 2015).

### 4.1 DROP Results

As shown in Table 3, our model achieves over 50% relative improvement (over 33% absolute) over the previous state-of-the-art NAQANet system. The ablations indicate that the improvements due to the addition of extra ops (`Diff100`, `Sum3`, `Merge`) are roughly consistent with their proportion in the data. Specifically, the `Diff100` and `Sum3` derivations increase the oracle performance by 8.7% and 1.4% respectively, corresponding to model improvements of roughly 9% and 1.1%, respectively. Answers requiring two spans occur about 2.8% of the time, which is a 60.4% proportion of the *Spans* answer type. `Merge` only improves the *Spans* answer type by 9%, which we think is due to the significant 11:1 class imbalance between competing single and multiple spans. As a result, multiple spans are under-predicted, leaving considerable headroom there.

Pre-training on CoQA then fine-tuning on DROP lead to our best results on DROP, reported in Table 3. After fine-tuning on DROP, the model forgot how to do CoQA, with an overall F1 score of 52.2 on the CoQA dev set. If one prefers a model competent in both types of input, then the forgetting can be prevented by fine-tuning on both CoQA and DROP datasets simultaneously. This resulted in dev set F1 scores of 82.2 on CoQA and 81.1 on DROP. The CoQA performance is decent and compares well with the pre-trained model performance of 82.5. The 0.5% drop in DROP performance is likely attributable to the difference between pre-training versus fine-tuning on CoQA.

We ensembled 6 models (3 seeds $\times$ 2 learning rates) for an additional 1% improvement.

### 4.2 Results on Math Word Problems

We trained our model on the Illinois math word problems dataset (Roy and Roth, 2015), which contains answers requiring multiplication and

[7] The "basic" model includes all $\mathcal{D}_{\text{direct}}$, all $\mathcal{S}$, and the simple binary operations `Sum` and `Diff`.

| | |
|---|---|
| Roy et al. (2015) | 73.9 |
| Liang et al. (2016) | **80.1** |
| Wang et al. (2018) | 73.3 |
| Our basic: IL data | $48.6 \pm 5.3$ |
| + `Mul` and `Div` | $74.0 \pm 6.0$ |
| + DROP data | $\mathbf{83.2 \pm 6.0}$ |

Table 4: Accuracy on the Illinois (IL) dataset[8] of 562 single-step word problems, using the five cross-validation folds of Roy and Roth (2015). Standard deviations were computed from the five folds. Roughly half the questions require the use of `Sum` and `Diff`, and half require `Mul` and `Div`.

division—operations not present in DROP—as well as addition and subtraction, in roughly equal proportion. Given the small ($N = 562$) dataset size, training and evaluation is done with five-fold cross-validation on a standardized set of splits. As shown in Table 4, when we added `Mul` and `Div` to our basic DROP operations, the model was able to learn to use them. Transferring from the DROP dataset further improved performance beyond that of Liang et al. (2016), a model specific to math word problems that uses rules over dependency trees. Compared to other more general systems, our model outperforms the deep reinforcement learning based approach of Wang et al. (2018).

## 5 Conclusions and Future Work

We proposed using BERT for reading comprehension combined with lightweight neural modules for computation in order to smoothly handle both traditional factoid question answering and questions requiring symbolic reasoning in a single unified model. On the DROP dataset, which includes a mix of reading comprehension and numerical reasoning, our model achieves a 33% absolute improvement over the previous best. The same model can also do standard reading comprehension on CoQA, and focused numerical reasoning on math word problems. We plan to generalize this model to more complex and compositional answers, with better searching and pruning strategies of the derivations.

## Acknowledgements

We would like to thank Chris Alberti and Livio Baldini Soares for tremendously helpful discussions, and we are grateful to all members of the Google Research Language team.

[8] https://cogcomp.org/page/resource_view/98

## References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *NAACL*.

Kenton Lee, Tom Kwiatkowski, Ankur P. Parikh, and Dipanjan Das. 2017. Learning recurrent span representations for extractive question answering. *CoRR*, abs/1611.01436.

Chao-Chun Liang, Kuang-Yi Hsu, Chien-Tsung Huang, Chung-Min Li, Shen-Yu Miao, and Keh-Yih Su. 2016. A tag-based statistical english math word problem solver with understanding, reasoning and explanation. In *IJCAI*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you dont know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789.

Abhilasha Ravichander, Aakanksha Naik, Carolyn Penstein Rosé, and Eduard H. Hovy. 2019. Equate: A benchmark evaluation framework for quantitative reasoning in natural language inference. *CoRR*, abs/1901.03735.

Siva Reddy, Danqi Chen, and Christopher D. Manning. 2018. CoQA: A conversational question answering challenge. *CoRR*, abs/1808.07042.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *EMNLP*.

Subhro Roy, Tim Vieira, and Dan Roth. 2015. Reasoning about quantities in natural language. *Transactions of the Association for Computational Linguistics*, 3:1–13.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. 2019. Analysing mathematical reasoning abilities of neural models. *CoRR*, abs/1904.01557.

Lei Wang, Dongxiang Zhang, Lianli Gao, Jingkuan Song, Long Guo, and Heng Tao Shen. 2018. MathDQN: Solving arithmetic word problems via deep reinforcement learning. In *AAAI*.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. QANet: Combining local convolution with global self-attention for reading comprehension. *Proceedings of ICLR*.

Dongxiang Zhang, Lei Wang, Nuo Xu, Bing Tian Dai, and Heng Tao Shen. 2018. The gap of semantic parsing: A survey on automatic math word problem solvers. *IEEE transactions on pattern analysis and machine intelligence*.

# A Gated Self-attention Memory Network for Answer Selection

**Tuan Lai** [*1], **Quan Hung Tran** [*2], **Trung Bui** [2], **Daisuke Kihara** [1]
{lai123,dkihara}@purdue.edu, {qtran,bui}@adobe.com
[1] Purdue University, West Lafayette, IN
[2] Adobe Research, San Jose, CA

## Abstract

Answer selection is an important research problem, with applications in many areas. Previous deep learning based approaches for the task mainly adopt the Compare-Aggregate architecture that performs word-level comparison followed by aggregation. In this work, we take a departure from the popular Compare-Aggregate architecture, and instead, propose a new gated self-attention memory network for the task. Combined with a simple transfer learning technique from a large-scale online corpus, our model outperforms previous methods by a large margin, achieving new state-of-the-art results on two standard answer selection datasets: TrecQA and WikiQA.

## 1 Introduction and Related Work

Answer selection is an important task, with applications in many areas (Lai et al., 2018). Given a question and a set of candidate answers, the task is to identify the most relevant candidate. Previous work on answer selection typically relies on feature engineering, linguistic tools, or external resources (Wang et al., 2007; Wang and Manning, 2010; Heilman and Smith, 2010; Yih et al., 2013; Yao et al., 2013). Recently, with the renaissance of neural network models, many deep learning based methods have been proposed to address the task (Tay et al., 2017b; Shen et al., 2017; Wang et al., 2017; Bian et al., 2017; Tymoshenko and Moschitti, 2018; Tay et al., 2018; Tayyar Madabushi et al., 2018; Yoon et al., 2019). They outperform traditional techniques. A common trait of a number of these deep learning methods is the use of the Compare-Aggregate architecture (Wang and Jiang, 2017). Typically in this architecture, contextualized vector representations of small units such as words of the question and the candidate are first *compared and aligned*. After that, these comparison results are then *aggregated* to calculate a score indicating the relevance between the question and the candidate. On standard answer selection datasets such as TrecQA (Wang et al., 2007) or WikiQA (Yang et al., 2015), Compare-Aggregate approaches achieve very competitive performance. However, they still have some limitations. For example, the first few layers of most previous Compare-Aggregate models encode the question-candidate pair into sequences of contextualized vector representations separately (Wang et al., 2017; Shen et al., 2017; Bian et al., 2017). These sequences are independent and completely ignore the information from the other sequence.

In this work, we take a departure from the popular Compare-Aggregate architecture, so instead, we propose a mix between two very successful architectures in machine comprehension and sequence modeling, the memory network (Sukhbaatar et al., 2015) and the self-attention architecture (Vaswani et al., 2017). In the context of answer selection, the self-attention architecture allows us to learn the contextual representation of elements in the sequence with respect to both the question and the answer, while the multi-hop reasoning memory network allows us to refine the decision over multiple steps. To this end, we propose a new memory-based, gated self-attention architecture for the task of answer selection. Combined with a simple transfer learning technique from a large-scale online corpus, our model achieves new state-of-the-art results on the TrecQA and WikiQA datasets.

In the following parts, we first describe our gated self-attention memory network for answer selection in Section 2. We then go into details our transfer learning approach in Section 3. After that, we describe the conducted experiments and their results in Section 4. Finally, we conclude this

[*]Equal contributions. The work was conducted while the first author interned at Adobe Research.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5953–5959, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

work in Section 5.

## 2 Gated Self-Attention Memory Network

### 2.1 The gated self-attention mechanism

The gated attention mechanism (Dhingra et al., 2017; Tran et al., 2017) extends the popular scalar-based attention mechanism by calculating a real vector gate to control the flow of information, instead of a scalar value. Let's denote the sequence of input vectors as $X = [\mathbf{x}_1..\mathbf{x}_n]$. If we have context information $\mathbf{c}$, then in traditional attention mechanism, association score $\alpha_i$ is usually calculated as a normalized dot product between the two vectors $\mathbf{c}$ and $\mathbf{x}_i$ (Equation 1) where $i \in [1..n]$.

$$\alpha_i = \frac{\exp(\mathbf{c}^T \mathbf{x}_i)}{\sum_{j \in [1..n]} \exp(\mathbf{c}^T \mathbf{x}_j)} \tag{1}$$

For the gated attention mechanism, the association between two vectors $\mathbf{c}$ and $\mathbf{x}_i$ is represented by gate vector $\mathbf{g}_i$ as follows:

$$\mathbf{g}_i = \sigma\big(f(\mathbf{c}, \mathbf{x}_i)\big) \tag{2}$$

where $\sigma$ denotes the element-wise sigmoid function. Function $f$ is a parameterized function and thus, is more flexible in modelling the interaction between vectors $\mathbf{c}$ and $\mathbf{x}_i$.

In this work, we propose a new type of self-attention based on the gated attention mechanism described above, and we refer to it as the *gated self-attention* mechanism (GSAM). We want to condition the gate vector not only on a context vector and a single input vector but also on the entire sequence of inputs. Therefore, we design function $f$ to be dependent on all the inputs in the sequence and the context vector. To calculate the gate for input $\mathbf{x}_i$, first, each of the inputs in the input sequence and the context vector will present an individual gate "vote". Then, the votes are aggregated to calculate gate $\mathbf{g}_i$ for $\mathbf{x}_i$. This process is illustrated in Equation 3:

$$\begin{aligned}
\mathbf{v}^j &= \mathbf{W}\mathbf{x}_j + \mathbf{b} \;;\; \mathbf{v}^c = \mathbf{W}\mathbf{c} + \mathbf{b} \\
s_i^j &= \mathbf{x}_i^T \mathbf{v}^j \;;\; s_i^c = \mathbf{x}_i^T \mathbf{v}^c \\
\alpha_i^j &= \frac{\exp(s_i^j)}{\sum_{k \in [1..n]} \exp(s_i^k) + \exp(s_i^c)} \\
\alpha_i^c &= \frac{\exp(s_i^c)}{\sum_{k \in [1..n]} \exp(s_i^k) + \exp(s_i^c)} \\
\mathbf{g}_i &= f_i(c, X) \\
&= \sigma\left( \sum_j \left( \alpha_i^j \, \mathbf{x}^j \right) + \alpha_i^c \, \mathbf{c} \right)
\end{aligned} \tag{3}$$

where $\mathbf{W}$ and $\mathbf{b}$ are learnable parameters shared among functions $f_1 ... f_n$. Vectors $\mathbf{v}$s are linear-transformed inputs which are used to calculate the self attentions. $s_i^j$ is the unnormalized attention score of input $\mathbf{x}_j$ put on $\mathbf{x}_i$ and $\alpha_i^j$ is the normalized score. We use affine-transformed inputs $\mathbf{v}$ and $\mathbf{x}$ to calculate the self-attention instead of just $\mathbf{x}$ to break the attention symmetry phenomenon.

### 2.2 Combining with the memory network

In most previous memory network architectures, interactions between memory cells are relatively limited. At each hop, a single control vector is used to interpret each memory cell independently. To overcome this limitation, we combine GSAM described in Section 2.1 with the memory network architecture to create a new network called the Self-Attention Memory Network (GSAMN). Figure 1 shows the simplified computation flow of GSAMN. In each reasoning hop, instead of using only context vector $\mathbf{c}$ to interpret the inputs, we use GSAM. Let $\mathbf{c}_k$ be the controlling context and $\mathbf{x}_1^k ... \mathbf{x}_n^k$ be the memory values at the $k^{\text{th}}$ reasoning hop. Each memory cell update from the $k^{\text{th}}$ hop to the next hop is calculated as the gated self-attention update (Equation 4).

$$\begin{aligned}
\mathbf{g}_i &= f_i(\mathbf{c}_k, X) \\
\mathbf{x}_i^{k+1} &= \mathbf{g}_i \odot \mathbf{x}_i^k
\end{aligned} \tag{4}$$

The controller's update is a combination of the gated self-attention above, and the memory network's traditional aggregate update. As the memory state values have already been attended to by the gating mechanism, we only need to average them (not weighted average).

$$\begin{aligned}
\mathbf{g}_c &= f_c(\mathbf{c}_k, X) \\
\mathbf{c}_{k+1} &= \mathbf{g}_c \odot \mathbf{c}_k + \frac{1}{n} \sum_i \mathbf{x}_i^{k+1}
\end{aligned} \tag{5}$$

.

### 2.3 GSAMN for answer selection

In the context of answer selection, we concatenate question Q and candidate answer A to a single sequence and treat the task as a binary classification problem. Given the GSAMN architecture above, we can use final controller state $\mathbf{c}_T$ as the representation of the sequence. The matching probability $P(A \mid Q)$ is finally calculated as follows:

$$P(A \mid Q) = \sigma\Big( \mathbf{W}_c \, \mathbf{c}_T + \mathbf{b}_c \Big) \tag{6}$$

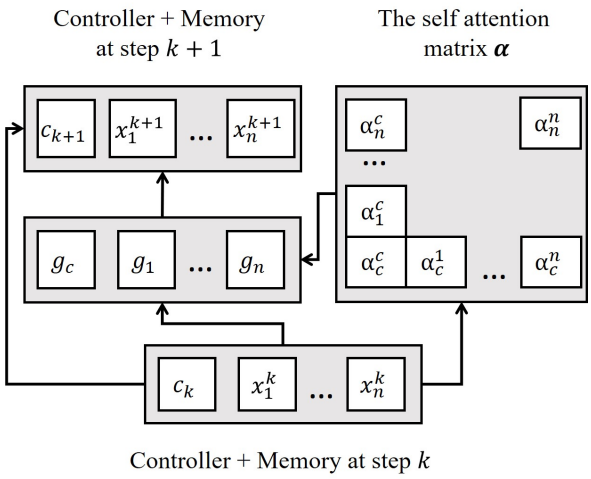


Figure 1: Simplified computation flow of the Gated Self-Attention Memory Network

where $\mathbf{W}_c$ and $\mathbf{b}_c$ are learnable parameters. We can initialize the memory values $\mathbf{x}_1^0 \ldots \mathbf{x}_n^0$ using any representation model such as word2vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014), ELMo (Peters et al., 2018), or BERT (Devlin et al., 2018). The control vector $\mathbf{c}$ is a randomly initialized learnable vector.

## 3 Transfer Learning

Previous studies on answer selection have focused mostly on small-scale datasets. On the other hand, many community question answering (CQA) platforms such as Yahoo Answers and Stack Exchange have become an essential source of information for many people. The amount of data (i.e., questions and answers) in these CQA platforms is huge and encompasses many domains and topics. This provides a great opportunity to apply transfer learning techniques to improve answer selection systems trained on limited datasets.

We crawled question-answer pairs related to various topics from Stack Exchange [1]. After that, we removed every pair that contains text written in a language different from English. Furthermore, to ensure that in each collected pair the answer is highly relevant to the question, we removed pairs whose answers have less than two up-votes from community users. Finally, because training answer selection models also require negative examples, for each question, we sampled several real answers not related to the question to build up negative pairs. In the end, our dataset has 628,706 positive pairs and 9,874,758 negative pairs in total. We refer to our newly collected dataset as StackExchangeQA. Table 1 shows some examples of positive question-answer pairs from the dataset. The dataset has question-answer pairs from many different domains and topics. The code for constructing the StackExchangeQA dataset is available online [2].

In this work, we employ a basic transfer learning technique. The first step is to pre-train our answer selection model on the StackExchangeQA dataset. Then, the second step is to fine-tune the same model on a target dataset of interest such as TrecQA or WikiQA. Despite the simplicity of the technique, the performance of our model improves substantially compared to not using transfer learning. Different from previous works which use source datasets that were manually annotated (Min et al., 2017; Chung et al., 2018), our source dataset required minimal effort to obtain and preprocess. The choice of crawling question-answer pairs from the Stack Exchange website was arbitrary. We could also have crawled data from websites such as Yahoo Answers instead.

## 4 Experiments and Results

To evaluate the effectiveness of our proposed answer selection model, we use two datasets: TrecQA and WikiQA. The TrecQA dataset (Wang et al., 2007) was created from the TREC Question Answering tracks. There are two versions of TrecQA: raw and clean. Both versions have the same training set but their development and test sets differ. In this study, we use the clean version of the dataset that removed questions in development and test sets with no answers or only positive/negative answers. The clean version has 1,229/65/68 questions and 53,417/1,117/1,442 question-answer pairs for the train/dev/test split. The WikiQA dataset (Yang et al., 2015) was constructed from real queries of Bing and Wikipedia. Following the literature (Yang et al., 2015; Bian et al., 2017; Shen et al., 2017), we removed all questions with no correct answers before training and evaluating answer selection models. The excluded WikiQA has 873/126/243 questions and 8,627/1,130/2,351 question-answer pairs for train/dev/test split.

Similar to previous work, we report the model

[1] https://stackexchange.com/

[2] https://github.com/laituan245/StackExchangeQA

| Domain | QA Pair |
|---|---|
| Academia | **Question**: Is it okay for a PhD student to go on holidays in breaks?<br>**Answer**: Do you have an adviser? Have you talked to them about this? Most should be fine with you taking some time off to visit your family, but you should probably discuss longer breaks with them to work out all the details. |
| Apple Product | **Question**: What hidden features have you found in iOS 6?<br>**Answer**: Newly downloaded apps have a "new" label on the home screen. |
| Chemistry | **Question**: Hydrochloric acid vs hydrogen chloride?<br>**Answer**: Hydrochloric acid is an aqueous solution of hydrogen chloride. |
| Cooking | **Question**: How do I prevent tomatoes from falling in a green salad?<br>**Answer**: I work around this by serving tomatoes on the top of the individual salads after they've been portioned out. I'm not sure of a way to keep them incorporated. |
| Philosophy | **Question**: What did Socrates teach which lead to his conviction that he spoiled youth and taught other Gods?<br>**Answer**: I think in general one of the problems Socrates' contemporaries may have had with him was not so much what he taught but how he taught. Perhaps Socrates' method of philosophy was characterised more by testing propositions through questioning, than any strict concern with formulating a set of propositions on any one subject. |

Table 1: Examples of positive question-answer pairs from the StackExchangeQA dataset

performance as the mean average precision (MAP) and mean reciprocal rank (MRR) [3]. In all experiments, we use the base version of BERT (Devlin et al., 2018) to initialize the memory of our proposed architecture. We fine-tune the BERT embeddings during training. We set the number of reasoning hops to be 2. We use the Adam optimizer with a learning rate of 5e-5, $\beta_1 = 0.9$, $\beta_2 = 0.999$, L2 weight decay of 0.01, learning rate warmup over the first 10 percent of the total number of training steps, and linear decay of the learning rate. We did hyper-parameter tuning on the development sets.

It is worth noting that, we have experimented with various values for the number of reasoning hops. We found that using 2 hops gives the best performance on the tested datasets while using larger number of hops decreases the performance slightly. We attribute the diminishing returns in increasing the number of hops to the limited size of the TrecQA and WikiQA datasets. Many previous works related to memory networks also use small number of memory hops (Weston et al., 2015; Sukhbaatar et al., 2015; Miller et al., 2016; Zhang et al., 2018).

### 4.1 Comparison with Previous Methods

Table 2 summarizes the performances of our proposed models and compares them to the baselines on the TrecQA and WikiQA datasets. The full model [BERT + GSAMN+ Transfer Learning] outperforms the previous state-of-the-art methods by a large margin. Note that by simply fine-tuning the pre-trained BERT embeddings, one can easily achieve very competitive performance on both datasets. This is expected as BERT has been pre-trained on a massive amount of unlabeled data. However, our proposed techniques do add a significant amount of performance. The gain from using all of our proposed techniques is larger than the difference between fine-tuning BERT model compared to previous systems in the TrecQA dataset.

### 4.2 Ablation Analysis

We aim to analyze the relative effectiveness of different components of our full model. From the original BERT baseline, we add one component at a time and evaluate the performance of the partial models on the datasets. From Table 2, we can see that both the variants [BERT + GSAMN] and [BERT + Transfer Learning] have better performance than the original BERT baseline. However, both of the partial variants still perform worse than the one with all the techniques. This shows that although each of our proposed components is effective by itself, we need to combine them together in order to achieve the best performance.

### 4.3 GSAMN versus Transformer

It was an iterative process to arrive at the current design of GSAMN. We aim to analyze whether the improvement in performance comes from the inductive bias that we introduced into the architec-

[3] https://aclweb.org/aclwiki/Question_Answering_(State_of_the_art)

| Model | TrecQA | | WikiQA | |
|---|---|---|---|---|
| | MAP | MRR | MAP | MRR |
| BERT + GSAMN+ Transfer | **0.914** | **0.957** | **0.857** | **0.872** |
| BERT + Transformers + Transfer | 0.895 | 0.939 | 0.831 | 0.848 |
| BERT + GSAMN | 0.906 | 0.949 | 0.821 | 0.832 |
| BERT + Transformers | 0.886 | 0.926 | 0.813 | 0.828 |
| ELMo + Compare-Aggregate | 0.850 | 0.898 | 0.746 | 0.762 |
| BERT + Transfer | 0.902 | 0.949 | 0.832 | 0.849 |
| BERT | 0.877 | 0.922 | 0.810 | 0.827 |
| QC + PR + MP CNN (2018) | 0.865 | 0.904 | — | — |
| IWAN + sCARNN (2018) | 0.829 | 0.875 | 0.716 | 0.722 |
| IWAN (2017) | 0.822 | 0.889 | 0.733 | 0.750 |
| Compare-Aggregate (2017) | 0.821 | 0.899 | 0.748 | 0.758 |
| BiMPM (2017) | 0.802 | 0.875 | 0.718 | 0.731 |
| HyperQA (2017a) | 0.784 | 0.865 | 0.705 | 0.720 |
| NCE-CNN (2016) | 0.801 | 0.877 | — | — |
| Attentive Pooling CNN (2016) | 0.753 | 0.851 | 0.689 | 0.696 |
| W&I (2015) | 0.746 | 0.820 | — | — |

Table 2: Results on the TrecQA and WikiQA datasets

ture or simply from having more parameters due to added complexity. To this end, we evaluated the performances of two variants [BERT + Transformers] and [BERT + Transformers + Transfer Learning]. These model variants simply use two Transformer layers (Vaswani et al., 2017) on top of BERT instead of using our GSAMN architecture. Table 2 clearly shows that GSAMN outperforms the Transformer based variants, with or without the transfer learning component.

We have experimented with adding more Transformer layers on top of BERT but the performance did not improve. For example, using 6 extra Transformer layers only achieves a MAP score of 0.885 on the TrecQA dataset. This is reasonable because BERT by itself already contains 12 Transformer layers. Without a new kind of layer such as our proposed GSAMN architecture, stacking more Transformer layers will not be helpful, especially in this case where the tested datasets are not large.

### 4.4 GSAMN versus Compare-Aggregate

Finally, we have a comparison between our full model and the Compare-Aggregate framework. Most previous Compare-Aggregate architectures use traditional word embeddings such as word2vec (Mikolov et al., 2013) or GloVe (Pennington et al., 2014). In contrast, our full model uses BERT which is an arguably more powerful language representation model. To this end, we implemented a Compare-Aggregate variant that uses dynamic-clip attention (Bian et al., 2017). We use ELMo (Peters et al., 2018) to represent the input words to the implemented Compare-Aggregate architecture. We use ELMo instead of BERT because BERT is in subword-level while one of the intuitions behind the Compare-Aggregate variant is about comparing word-level representations. In addition, we have tested the variant [BERT + Compare-Aggregate] but found it to be worse than the version [ELMo + Compare-Aggregate]. The results in Table 2 show that our model significantly outperforms [ELMo + Compare-Aggregate] as well.

## 5 Conclusions

In this paper, we propose a new gated self-attention memory network architecture for answer selection. Combined with a simple transfer learning technique from a large-scale CQA corpus, the model achieves the state-of-the-art performance on two well-studied answer selection datasets: TrecQA and WikiQA. In the future, we plan to investigate more transfer learning techniques for utilizing the large volume of existing CQA data. In addition, we plan to apply our self-attention memory network on other sentence matching tasks such as natural language inference, paraphrase identification, or measuring semantic relatedness.

## References


Weijie Bian, Si Kan Li, Zhao Yang, Guang Chen, and Zhiqing Lin. 2017. A compare-aggregate model with dynamic-clip attention for answer selection. In *CIKM*.

Yu-An Chung, Hung yi Lee, and James R. Glass. 2018. Supervised and unsupervised transfer learning for question answering. In *NAACL-HLT*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Bhuwan Dhingra, Hanxiao Liu, William W. Cohen, and Ruslan R. Salakhutdinov. 2017. Gated-attention readers for text comprehension. In *ACL*.

Michael Heilman and Noah A. Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 1011–1019, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tuan Manh Lai, Trung Bui, and Sheng Li. 2018. A review on deep learning techniques applied to answer selection. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2132–2144, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA. Curran Associates Inc.

Alexander H. Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *EMNLP*.

Sewon Min, Min Joon Seo, and Hannaneh Hajishirzi. 2017. Question answering through transfer learning from large fine-grained supervision data. In *ACL*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke S. Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*.

Jinfeng Rao, Hua He, and Jimmy Lin. 2016. Noise-contrastive estimation for answer selection with deep neural networks. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 1913–1916. ACM.

Cícero Nogueira dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *ArXiv*, abs/1602.03609.

Gehui Shen, Yunlun Yang, and Zhi-Hong Deng. 2017. Inter-weighted alignment network for sentence pair modeling. In *EMNLP*.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2017a. Enabling efficient question answer retrieval via hyperbolic neural networks. *CoRR*, abs/1707.07847.

Yi Tay, Minh C. Phan, Anh Tuan Luu, and Siu Cheung Hui. 2017b. Learning to rank question answer pairs with holographic dual lstm architecture. In *SIGIR*.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Multi-cast attention networks. In *KDD*.

Harish Tayyar Madabushi, Mark Lee, and John Barnden. 2018. Integrating question classification and deep learning for improved answer selection. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3283–3294, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Quan Hung Tran, Gholamreza Haffari, and Ingrid Zukerman. 2017. A generative attentional neural network model for dialogue act classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 524–529.

Quan Hung Tran, Tuan Lai, Gholamreza Haffari, Ingrid Zukerman, Trung Bui, and Hung Bui. 2018. The context-dependent additive recurrent neural net. In *NAACL-HLT*.

Kateryna Tymoshenko and Alessandro Moschitti. 2018. Cross-pair text representations for answer sentence selection. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2162–2173, Brussels, Belgium. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Mengqiu Wang and Christopher D. Manning. 2010. Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 1164–1172, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. 2007. What is the jeopardy model? a quasi-synchronous grammar for qa. In *EMNLP-CoNLL*.

Shuohang Wang and Jing Jiang. 2017. A compare-aggregate model for matching text sequences. *CoRR*, abs/1611.01747.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. In *IJCAI*.

Zhiguo Wang and Abraham Ittycheriah. 2015. Faq-based question answering via word alignment. *ArXiv*, abs/1507.02628.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. *CoRR*, abs/1410.3916.

Yi Yang, Wen tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *EMNLP*.

Xuchen Yao, Benjamin Van Durme, Chris Callison-burch, and Peter Clark. 2013. Answer extraction as sequence tagging with tree edit distance. In *North American Chapter of the Association for Computational Linguistics (NAACL)*.

Scott Wen-tau Yih, Ming-Wei Chang, Chris Meek, and Andrzej Pastusiak. 2013. Question answering using enhanced lexical semantic models. ACL Association for Computational Linguistics.

Seunghyun Yoon, Franck Dernoncourt, Deok Seong Kim, Trung Bui, and Kyomin Jung. 2019. A compare-aggregate model with latent clustering for answer selection. *ArXiv*, abs/1905.12897.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? In *ACL*.

# Polly Want a Cracker: Analyzing Performance of Parroting on Paraphrase Generation Datasets

**Hong-Ren Mao, Hung-Yi Lee**
Department of Electrical Engineering
National Taiwan University
ccm29cam@gmail.com, hungyilee@ntu.edu.tw

## Abstract

Paraphrase generation is an interesting and challenging NLP task which has numerous practical applications. In this paper, we analyze datasets commonly used for paraphrase generation research, and show that simply parroting input sentences surpasses state-of-the-art models in the literature when evaluated on standard metrics. Our findings illustrate that a model could be seemingly adept at generating paraphrases, despite only making trivial changes to the input sentence or even none at all.

## 1 Introduction

The task of paraphrase generation has many important applications in NLP. It can be used to generate adversarial examples of input text, which can then be used to train neural networks so that they become less susceptible to adversarial attack (Iyyer et al., 2018). For knowledge-based QA systems, a paraphrasing step can produce multiple variations of a user query and match them with knowledge base assertions, enhancing recall (Yin et al., 2015; Fader et al., 2014). Relation extraction can also benefit from incorporating paraphrase generation into its processing pipeline (Romano et al., 2006). Manually annotating translation references is expensive, and automatically generating references through paraphrasing has been shown to be effective for evaluation of machine translation (Zhou et al., 2006; Kauchak and Barzilay, 2006).

Datasets used for paraphrase generation include QUORA[1], TWITTER (Lan et al., 2017) and MSCOCO (Lin et al., 2014). Previous work on paraphrase generation that used these datasets (Wang et al., 2019; Gupta et al., 2018; Li et al., 2018; Prakash et al., 2016) chose BLEU (Papineni et al., 2002), METEOR (Lavie and Agarwal, 2007) and TER (Snover et al., 2006) as evaluation metrics.

In this paper, we find that simply using the input sentence as output in an unsupervised manner (*i.e.* fully parroting the input) significantly outperforms the state-of-the-art on two metrics for TWITTER, and on one metric for QUORA. Even after changing part of the input sentence (*i.e.* partially parroting the input), state-of-the-art metric scores can still be surpassed.

Consequently, for future paraphrase generation research which achieve good evaluation scores, we suggest investigating whether their methods or models act differently from simple parroting behavior.

## 2 Method Description

Given an input sentence $i$, the goal of paraphrase generation is to generate an output sentence $o$ which is semantically identical to $i$, but contain variations in lexicon or syntax. Full parroting simply uses the input as output ($o = i$).

Paraphrase generation models may not parrot the input sentence word for word, but it is possible that they only modify a few words of the input, thus we also experiment with simple methods of modifying $i$, such as replacing or cutting words from the head, from the tail or from random positions.

Both full parroting and the forms of partial parroting we use are fully unsupervised.

## 3 Datasets

**QUORA.** The QUORA dataset contains 149,263 paraphrase sentence pairs (positive examples) and 255,027 non-paraphrase sentence pairs (negative examples). Having both positive and negative ex-

[1] https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5960–5968, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

amples makes it appealing for research on paraphrase generation (Gupta et al., 2018; Li et al., 2018) and identification (Lan and Xu, 2018). After processing the dataset, there are 149,650 unique sentences that have reference paraphrases.

Gupta et al. (2018) sampled 4K sentences as their test set, but did not specify which sentences they used. Li et al.(2018) sampled 30K sentences as their test set, also not specifying which sentences they used. To avoid selecting a subset of data that is biased in favor of our method, we perform evaluation on the entire QUORA dataset. Although we evaluate on the entire dataset, the size of our training set is zero due to the fully unsupervised nature of full and partial parroting.

We group sentences by the number of reference paraphrases they have, and plot the relative counts in Appendix A. It can be seen that over 64% of entries have only a single reference paraphrase, which is problematic because even if a paraphrase of good quality is generated for any one of these entries, BLEU, METEOR and TER scores could still be inferior if the generated paraphrase differs too much from the single reference paraphrase. Previous paraphrase generation work on QUORA (Gupta et al., 2018; Li et al., 2018) did not mention removing these entries, thus we include them in our experiments for fair comparison. However, we strongly recommend future work which wishes to use BLEU, METEOR and TER as evaluation metrics to only consider entries that have multiple reference paraphrases.

**TWITTER.** There are 114,025 paraphrase sentence pairs in TWITTER, which were acquired by collecting tweets which contain identical URLs (Lan et al., 2017). As with QUORA, prior paraphrase generation work on this dataset (Li et al., 2018) did not provide their sampled test set sentences, so we evaluate parroting on the entire dataset to avoid bias. We follow the same data processing steps as QUORA, and plot the number of reference paraphrases in Appendix A.

**MSCOCO.** This is an image captioning dataset, with multiple captions provided for a single image (Lin et al., 2014). There have been multiple works which use it as a paraphrase generation dataset by treating captions of the same image as paraphrases (Wang et al., 2019; Gupta et al., 2018; Prakash et al., 2016). The training and testing sets are available, containing 331,163 and 162,016 input sentences respectively.

However, relevance scores for captions of the same image score only 3.38 out of 5 under human evaluation (in contrast, the score is 4.82 for QUORA) (Gupta et al., 2018), due to the fact that different captions for the same image often vary in the semantic information conveyed. This makes the use of MSCOCO as a paraphrase generation dataset questionable.

We plot the number of reference paraphrases in Appendix A.

## 4 Experiments

We evaluate the performance of full parroting on all three datasets and compare with state-of-the-art models.

We also study the performance of partial parroting. Whereas full parroting does not modify the input sentence, partial parroting replaces or cuts some of the input words. We try three different modes of choosing words to be cut or replaced: from the sentence head, from the tail or sampled randomly.

### 4.1 Evaluation

Following prior paraphrase generation research which used QUORA, TWITTER and MSCOCO, we use BLEU, METEOR and TER as evaluation metrics. When calculating metric scores, all available reference paraphrases for a given input sentence are considered.

### 4.2 Results

**Full parroting.** Our results are organized in Tables 1, 2 and 3. We see for TWITTER, parroting outperforms the state-of-the-art by significant margins on both BLEU and METEOR scores; for QUORA, parroting outperforms the state-of-the-art appreciably on METEOR while having comparable performance on BLEU.

The poor performance of full parroting on MSCOCO is due to higher edit distances between input sentences and their reference paraphrases. TER measures the edit distance of a sentence to a reference sentence, normalized by the average length of all references (Snover et al., 2006):

$$\text{TER} = \frac{\text{\# of edits}}{\text{average \# of reference words}}$$

We see that the TER score of full parroting is particularly high on MSCOCO compared to the other two datasets. Correspondingly, the BLEU and METEOR scores are lower by a wide margin.

| Metric | STATE-OF-THE-ART | | | PARROT | | |
|---|---|---|---|---|---|---|
| | paper | score | num_train | score | num_train | ΔSOTA |
| BLEU ↑ | (Li et al., 2018) | **43.54** | 100K | 41.59 | 0 | -4.47% |
| METEOR ↑ | (Gupta et al., 2018) | 33.6 | 150K | **38.60** | 0 | **+14.88%** |
| TER ↓ | (Gupta et al., 2018) | **39.5** | 150K | 45.22 | 0 | +14.47% |

Table 1: Performance of full parroting v.s. state-of-the-art on QUORA. Higher BLEU and METEOR scores are better, while higher TER scores are worse. Bold text represents best results.

| Metric | STATE-OF-THE-ART | | | PARROT | | |
|---|---|---|---|---|---|---|
| | paper | score | num_train | score | num_train | ΔSOTA |
| BLEU ↑ | (Li et al., 2018) | 45.74 | 110K | **65.26** | 0 | **+42.67%** |
| METEOR ↑ | (Li et al., 2018) | 20.18 | 110K | **41.73** | 0 | **+106.77%** |
| TER ↓ | - | - | - | **41.87** | 0 | - |

Table 2: Performance of full parroting v.s. state-of-the-art on TWITTER.

| Metric | STATE-OF-THE-ART | | | PARROT | | |
|---|---|---|---|---|---|---|
| | paper | score | num_train | score | num_train | ΔSOTA |
| BLEU ↑ | (Wang et al., 2019) | **44.0** | 331K | 19.0 | 0 | -56.8% |
| METEOR ↑ | (Wang et al., 2019) | **34.7** | 331K | 23.9 | 0 | -31.0% |
| TER ↓ | (Wang et al., 2019) | **37.1** | 331K | 70.4 | 0 | +89.7% |

Table 3: Performance of full parroting v.s. state-of-the-art on MSCOCO.

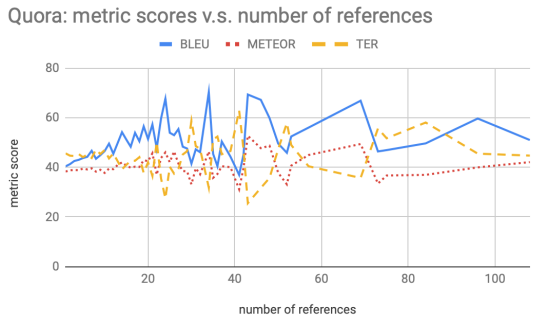


Figure 1: Metric scores v.s. number of reference paraphrases (Quora). For lower numbers of references, metric scores improve as the amount of references increases. For higher numbers of references, there does not appear to be a clear correlation.

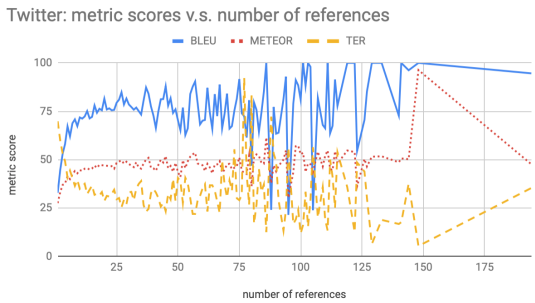


Figure 2: Metric scores v.s. number of reference paraphrases (Twitter). For lower numbers of references, metric scores improve as the amount of references increases. For higher numbers of references, there does not appear to be a clear correlation.

For further investigation of parroting on QUORA and TWITTER, we plot parroting performance versus the number of reference paraphrases available for a given input sentence (Figures 1 and 2). If the number of references is not too high, metric scores generally improve when the number of references rises. Once the number of references exceeds a certain threshold, we do not observe a clear correlation, showing that the probability of finding a reference sentence which bears higher resemblance to the input does not increase proportionally with the number of references.

The choice of testing on the entire dataset for QUORA and TWITTER experiments was to avoid bias in favor of parroting. Nevertheless, we also randomly sampled test sets of size 4K for QUORA in the same manner as (Gupta et al., 2018) (which holds the most state-of-the-art records on QUORA) and test sets of size 5K for TWITTER in the same manner as (Li et al., 2018) (which holds all state-

| Statistic | QUORA (5K test set × 1200) | | | TWITTER (4K test set × 250) | | |
|---|---|---|---|---|---|---|
| | BLEU ↑ | METEOR ↑ | TER ↓ | BLEU ↑ | METEOR ↑ | TER ↓ |
| Average | 41.57 | 38.59 | 45.21 | 74.97 | 46.22 | 33.43 |
| Std. Dev. | 0.50 | 0.20 | 0.50 | 0.36 | 0.14 | 0.35 |
| Max. | 43.12 | 39.29 | 46.95 | 76.01 | 46.56 | 34.31 |
| Min. | 39.98 | 37.85 | 43.53 | 74.01 | 45.89 | 32.41 |

Table 4: Performance of full parroting on randomly sampled test sets. The test set size and sampling method is the same as that described in prior state-of-the-art work. Here, scores for sampled QUORA test sets are similar to those of the full dataset, and the scores for TWITTER test sets are better than scores achieved on the full dataset.

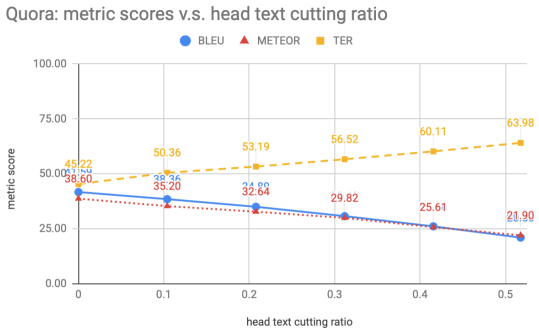


Figure 3: Metric scores v.s. ratio of text that is cut from the start of the input sentence (Quora).

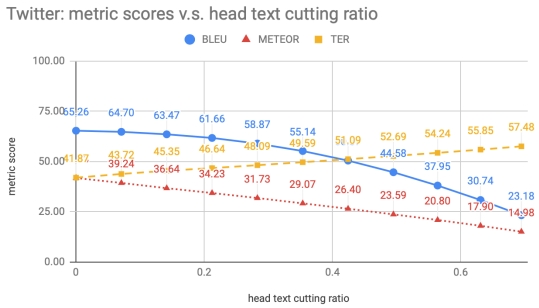


Figure 4: Metric scores v.s. ratio of text that is cut from the start of the input sentence (Twitter).

of-the-art records on TWITTER). In total, 1200 test sets of size 4K were sampled for QUORA and 250 test sets of size 5K were sampled for TWITTER. Parroting performance on these sampled test sets can be found in Table 4. It can be observed that the average metric scores for QUORA are similar to the scores in Table 1, whereas the average scores for TWITTER are noticeably better than those in Table 2. Furthermore, the score deviation between different samples is small. Consequently, although the exact test sets used by (Gupta et al., 2018) and (Li et al., 2018) are not available, it is logical to assume that parroting performance would still exceed or be on par with the state-of-the-art on those test sets.

**Partial parroting.** We also introduce lexical variation into our parroting method by replacing or cutting words of the input sentence. For replacement, we substitute input words with an out-of-vocabulary word not found in any of the input sentence's reference paraphrases. Paraphrase generation models are usually allowed to generate words which exist in reference paraphrases; we purposely use out-of-vocabulary words to give harsher scores to our method.

Figures 3 and 4 show performance of cutting words from the start of input sentences. For QUORA, when over 10% of the input sentence has been modified by being cut off, partial parroting underperforms the state-of-the-art by only 3.8% on METEOR. For TWITTER, the same form of partial parroting (cutting off words) still outperforms the state-of-the-art on BLEU when input sentences are modified by 42% , and does the same on METEOR when the input is modified by 56%. Additionally, we experiment with cutting words in other positions, and also replace words rather than cut them away. The results can be found in Appendix B.

Earlier work using QUORA and TWITTER (Gupta et al., 2018; Li et al., 2018) only provided a few examples of output paraphrases, and did not study in detail the paraphrasing behavior of their models, making it unclear whether the models achieve qualitatively better results than our simple rule-based parroting techniques, given that evaluation scores of the two are similar. We recommend future research to perform such an analysis if their metric scores are close to that of parroting.

## 5 Related Work

For the task of paraphrase generation, Wang et al. (2019) trained a Transformer network on

MSCOCO; Gupta et al. (2018) trained a seq2seq variational autoencoder (VAE) on QUORA and MSCOCO; Li et al. (2018) trained a seq2seq pointer network on QUORA and TWITTER, then fine-tuned it with an *evaluator* which was trained via inverse reinforcement learning; Prakash et al. (2016) trained a seq2seq model with residual connections on MSCOCO.

Work on paraphrase generation using other datasets can also be found. Methods include lexical substitution (Hassan et al., 2007; Bolshakov and Gelbukh, 2004), back-translation (Wieting and Gimpel, 2018) and sequence-to-sequence neural networks (Iyyer et al., 2018).

It is worth noting that paraphrase generation serves practical purposes, such as augmenting training data for NLP models to decrease their susceptibility to adversarial attack (Iyyer et al., 2018), or enhancing recall for QA systems (Yin et al., 2015; Fader et al., 2014). Improvement of downstream model performance is a valid evaluation metric for paraphrase generation, and future work wishing to use QUORA entries which only have a single reference paraphrase could choose such an evaluation metric instead of BLEU, METEOR or TER.

As a sidenote, we also ran experiments in which BLEU scores were calculated using non-reference dataset sentences. The results are in Appendix C.

## 6 Conclusion

In this work, we discover that various forms of simple parroting outperforms state-of-the-art results on QUORA and TWITTER when evaluated using BLEU and METEOR. An interpretation is that current models could simply be parroting input sentences, and researchers should perform qualitative analysis of such behavior. Another interpretation is that BLEU and METEOR are inappropriate for evaluating paraphrase generation models, in which case other metrics such as effectiveness of data augmentation (Iyyer et al., 2018), may be used instead.

## References


Igor A. Bolshakov and Alexander Gelbukh. 2004. Synonymous paraphrasing using wordnet and internet. In *Natural Language Processing and Information Systems*, pages 312–323, Berlin, Heidelberg. Springer Berlin Heidelberg.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 1156–1165, New York, NY, USA. ACM.

Ankush Gupta, Arvind Agarwal, Prawaan Singh, and Piyush Rai. 2018. A deep generative framework for paraphrase generation. In *AAAI*.

Samer Hassan, Andras Csomai, Carmen Banea, Ravi Sinha, and Rada Mihalcea. 2007. Unt: Subfinder: Combining knowledge sources for automatic lexical substitution. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 410–413. Association for Computational Linguistics.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885. Association for Computational Linguistics.

David Kauchak and Regina Barzilay. 2006. Paraphrasing for automatic evaluation. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, HLT-NAACL '06, pages 455–462, Stroudsburg, PA, USA. Association for Computational Linguistics.

Wuwei Lan, Siyu Qiu, Hua He, and Wei Xu. 2017. A continuously growing dataset of sentential paraphrases. In *EMNLP*.

Wuwei Lan and Wei Xu. 2018. Neural network models for paraphrase identification, semantic textual similarity, natural language inference, and question answering. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*.

Alon Lavie and Abhaya Agarwal. 2007. Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07, pages 228–231, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zichao Li, Xin Jiang, Lifeng Shang, and Hang Li. 2018. Paraphrase generation with deep reinforcement learning. In *EMNLP*.

Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *ECCV*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Aaditya Prakash, Sadid A. Hasan, Kathy Lee, Vivek Datla, Ashequl Qadir, Joey Liu, and Oladimeji Farri. 2016. Neural paraphrase generation with stacked residual lstm networks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2923–2934. The COLING 2016 Organizing Committee.

Lorenza Romano, Milen Kouylekov, Idan Szpektor, Ido Dagan, and Alberto Lavelli. 2006. Investigating a generic paraphrase-based approach for relation extraction. In *11th Conference of the European Chapter of the Association for Computational Linguistics*.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *In Proceedings of Association for Machine Translation in the Americas*, pages 223–231.

Su Wang, Rahul Gupta, Nancy Chang, and Jason Baldridge. 2019. A task in a suit and a tie: paraphrase generation with semantic augmentation. In *AAAI*.

John Wieting and Kevin Gimpel. 2018. Paranmt-50m: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462. Association for Computational Linguistics.

Pengcheng Yin, Nan Duan, Ben Kao, Junwei Bao, and Ming Zhou. 2015. Answering questions with complex semantic constraints on open knowledge bases. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 1301–1310, New York, NY, USA. ACM.

Liang Zhou, Chin-Yew Lin, and Eduard H. Hovy. 2006. Re-evaluating machine translation results with paraphrase support. In *EMNLP*.

# A Number of References Paraphrases for Datasets

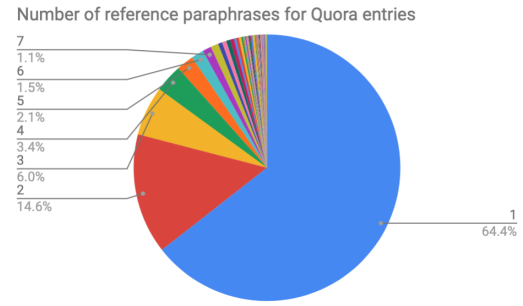


Figure 5: Number of reference paraphrases v.s. percentage of Quora dataset

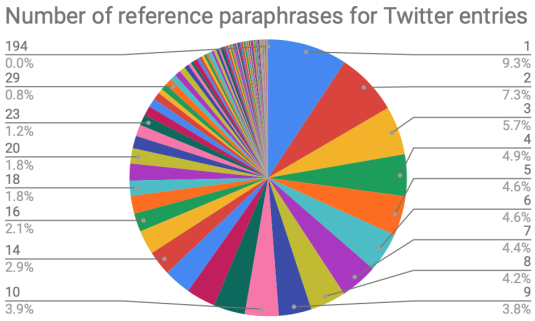


Figure 6: Number of reference paraphrases v.s. percentage of Twitter dataset

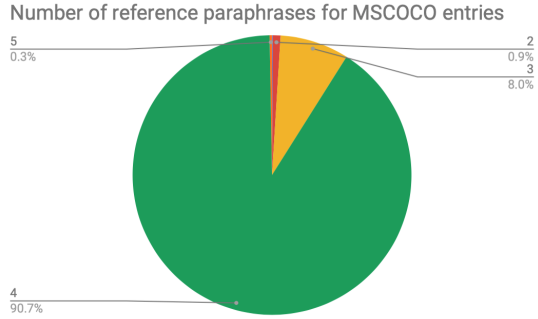


Figure 7: Number of reference paraphrases v.s. percentage of MSCOCO dataset

# B Performance of Partial Parroting

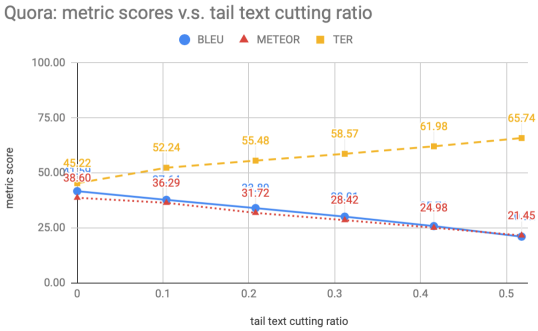


Figure 8: Metric scores v.s. ratio of text that is cut from the end of the input sentence (Quora)

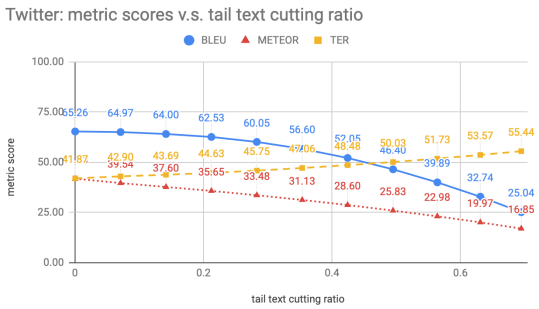


Figure 9: Metric scores v.s. ratio of text that is cut from the end of the input sentence (Twitter)

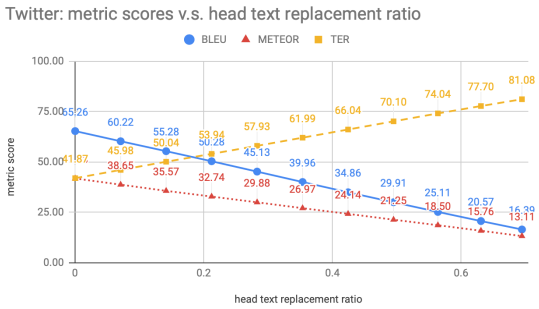


Figure 13: Metric scores v.s. ratio of text that is replaced from the start of the input sentence (Twitter)

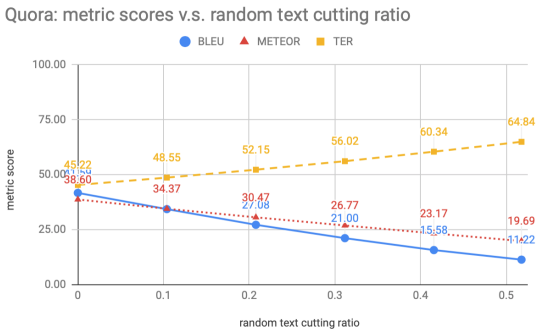


Figure 10: Metric scores v.s. ratio of text that is cut randomly from the input sentence (Quora)

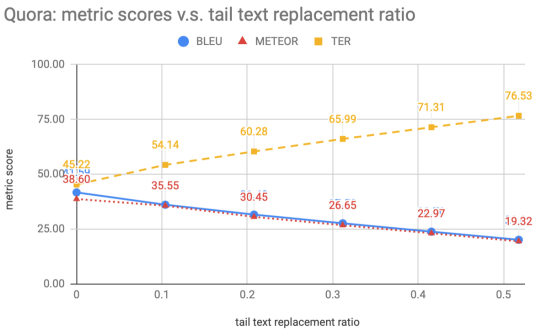


Figure 14: Metric scores v.s. ratio of text that is replaced from the end of the input sentence (Quora)

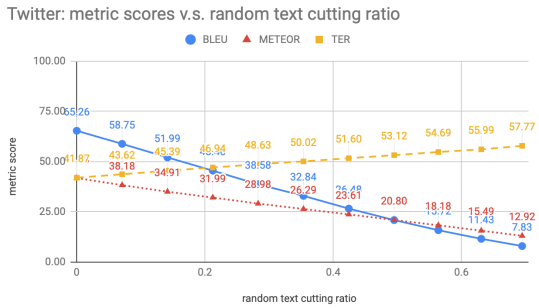


Figure 11: Metric scores v.s. ratio of text that is cut randomly from the input sentence (Twitter)

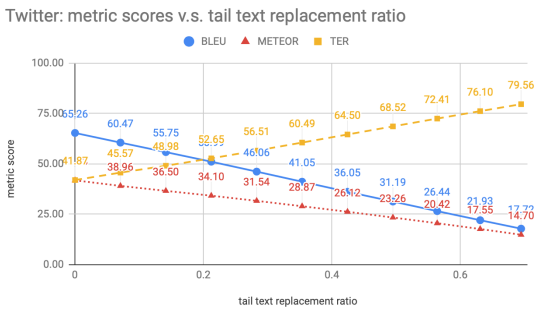


Figure 15: Metric scores v.s. ratio of text that is replaced from the end of the input sentence (Twitter)

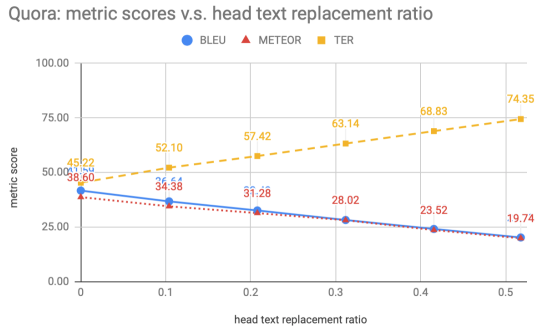


Figure 12: Metric scores v.s. ratio of text that is replaced from the start of the input sentence (Quora)

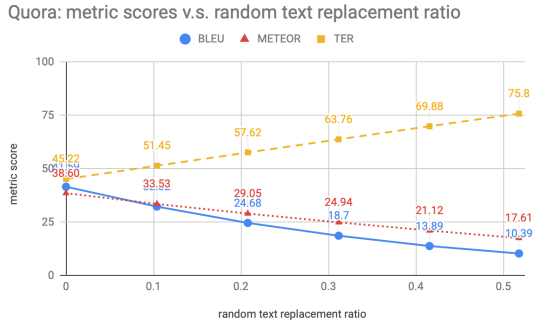


Figure 16: Metric scores v.s. ratio of text that is replaced randomly from the input sentence (Quora)

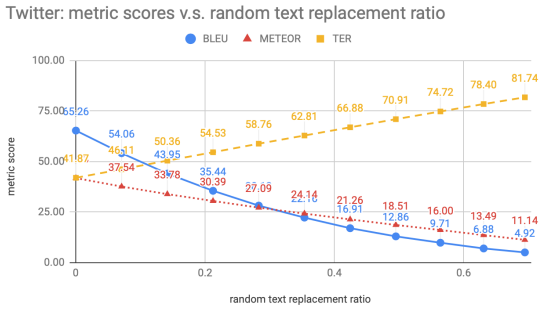


Figure 17: Metric scores v.s. ratio of text that is replaced randomly from the input sentence (Twitter)

## C Calculating BLEU score using non-reference sentences

For an input sentence, BLEU scores are usually calculated by comparing the input sentence with a number of reference sentences. We ran experiments in which 5 reference and 100 randomly sampled non-reference sentences were used, and show part of our results below. It can be seen that sentences with higher BLEU scores are more similar to the input sentence, which is to be expected.

Input sentence: what would happen if you hired two private detectives to spy on each other ?

| BLEU | |
|---|---|
| 0.3 - 0.35 | what will happen if i hire two private detectives to follow each other ?<br>what would happen if i got two private investigators to follow each other ?<br>what would happen if i sent two private investigators to find each other ? |
| 0.25 - 0.3 | (None) |
| 0.2 - 0.25 | what would happen if earth collided with a black hole ?<br>what if i hired two private eyes and ordered them to follow each other ?<br>what would happen if donald trump lost and refused to concede the election ? |
| 0.15 - 0.2 | would i be able to hire two private investigators and then get them to follow each other ?<br>what would happen if donald trump turned out to be a plant for hillary to win the white house ? |
| 0 - 0.15 | do i need to pay again on coursera if i switch sessions ?<br>is there anyway to tell if someone blocked you on facebook ?<br>what song do you listen to when you are angry ?<br>.....<br>what are bugs you noticed on quora ?<br>if i eat a pot cookie , how long until i 'm able to pass a urine test ? |

Input sentence: who do you think portrayed batman better : christian bale or ben affleck ?

| BLEU | |
|---|---|
| 0.4 - 0.45 | according to you , whose batman performance was best : christian bale or ben affleck ? |
| 0.35 - 0.4 | (None) |
| 0.3 - 0.35 | no fanboys please , but who was the true batman , christian bale or ben affleck ? |
| 0.25 - 0.3 | (None) |
| 0.2 - 0.25 | (None) |
| 0.15 - 0.2 | what do you think about " chinese dream " ?<br>what do you think about dota2 ?<br>who was better as batman : bale or affleck ?<br>did ben affleck shine more than christian bale as batman ?<br>do you think that the demonetization policy will backfire for bjp in 2019 elections ? |
| 0 - 0.15 | biswapati sarkar : how do you overcome a writer 's block ?<br>who is the better batman ? affleck or bale ?<br>how do you stop your cat from spraying ?<br>.....<br>do we always get what we deserve ?<br>can a moon have a moon ? |

# Query-focused Sentence Compression in Linear Time

**Abram Handler** and **Brendan O'Connor**
College of Information and Computer Sciences
University of Massachusetts Amherst
{ahandler,brenocon}@cs.umass.edu

## Abstract

Search applications often display shortened sentences which must contain certain query terms and must fit within the space constraints of a user interface. This work introduces a new transition-based sentence compression technique developed for such settings. Our query-focused method constructs length and lexically constrained compressions in linear time, by growing a subgraph in the dependency parse of a sentence. This theoretically efficient approach achieves an 11x empirical speedup over baseline ILP methods, while better reconstructing gold constrained shortenings. Such speedups help query-focused applications, because users are measurably hindered by interface lags. Additionally, our technique does not require an ILP solver or a GPU.

## 1 Introduction

Traditional study of extractive sentence compression seeks to create short, readable, single-sentence summaries which retain the most "important" information from source sentences. But search user interfaces often require compressions which must include a user's query terms and must not exceed some maximum length, permitted by screen space. Figure 1 shows an example.

This study examines the English-language compression problem with such length and lexical requirements. In our constrained compression setting, a source sentence $S$ is shortened to a compression $C$ which (1) must include all tokens in a set of query terms $Q$ and (2) must be no longer than a maximum budgeted character length, $b \in \mathbb{Z}^+$. Formally, constrained compression maps $(S, Q, b) \rightarrow C$, such that $C$ respects $Q$ and $b$. We describe this task as query-focused compression because $Q$ places a hard requirement on words from $S$ which must be included in $C$.




Figure 1: A search user interface (boxed, top) returns a snippet consisting of three compressions which must contain a users' query $Q$ (bold) and must not exceed $b = 75$ characters in length. The third compression $C$ was derived from source sentence $S$ (italics, bottom).

Existing techniques are poorly suited to constrained compression. While methods based on integer linear programming (ILP) can trivially accommodate such length and lexical restrictions (Clarke and Lapata, 2008; Filippova and Altun, 2013; Wang et al., 2017), these approaches rely on slow third-party solvers to optimize an NP-hard integer linear programming objective, causing user wait time. An alternative LSTM tagging approach (Filippova et al., 2015) does not allow practitioners to specify length or lexical constraints, and requires an expensive graphics processing unit (GPU) to achieve low runtime latency (access to GPUs is a barrier in fields like social science and journalism). These deficits prevent application of existing compression techniques in search user interfaces (Marchionini, 2006; Hearst, 2009), where length, lexical and latency requirements are paramount. We thus present a new stateful method for query-focused compression.

Our approach is theoretically and empirically faster than ILP-based techniques, and more accurately reconstructs gold standard compressions.

## 2 Related work

Extractive compression shortens a sentence by removing tokens, typically for summarization



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5969–5975,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| Approach | Complexity | Constrained |
|---|---|---|
| ILP | exponential | yes |
| LSTM tagger | linear | no |
| **VERTEX ADDITION** | **linear** | **yes** |

Table 1: Our VERTEX ADDITION technique (§3) constructs constrained compressions in linear time. Prior work (§2) has higher computational complexity (ILP) or does not respect hard constraints (LSTM tagger).

(Knight and Marcu, 2000; Clarke and Lapata, 2008; Filippova et al., 2015; Wang et al., 2017).[1] To our knowledge, this work is the first to consider extractive compression under hard length and lexical constraints.

We compare our VERTEX ADDITION approach to ILP-based compression methods (Clarke and Lapata, 2008; Filippova and Altun, 2013; Wang et al., 2017), which shorten sentences using an integer linear programming objective. ILP methods can easily accommodate lexical and budget restrictions via additional optimization constraints, but require worst-case exponential computation.[2]

Finally, compression methods based on LSTM taggers (Filippova et al., 2015) cannot currently enforce lexical or length requirements. Future work might address this limitation by applying and modifying constrained generation techniques (Kikuchi et al., 2016; Post and Vilar, 2018; Gehrmann et al., 2018).

## 3 Compression via VERTEX ADDITION

We present a new transition-based method for shortening sentences under lexical and length constraints, inspired by similar approaches in transition-based parsing (Nivre, 2003). We describe our technique as VERTEX ADDITION because it constructs a shortening by *growing* a (possibly disconnected) subgraph in the dependency parse of a sentence, one vertex at a time. This approach can construct constrained compressions with a linear algorithm, leading to 11x lower latency than ILP techniques (§4). To our knowledge, our method is also the first to construct compressions by *adding* vertexes rather than *pruning* subtrees in a parse (Knight and Marcu, 2000; Almeida and Martins, 2013; Filippova and Alfonseca, 2015). We assume a boolean relevance model: $S$ must contain $Q$. We leave more sophisticated relevance models for future work.

### 3.1 Formal description

VERTEX ADDITION builds a compression by maintaining a state $(C_i, P_i)$ where $C_i \subseteq S$ is a set of added candidates, $P_i \subseteq S$ is a priority queue of vertexes, and $i$ indexes a timestep during compression. Figure 2 shows a step-by-step example.

During initialization, we set $C_0 \leftarrow Q$ and $P_0 \leftarrow S \setminus Q$. Then, at each timestep, we pop some candidate $v_i = h(P_i)$ from the head of $P_i$ and evaluate $v_i$ for inclusion in $C_i$. (Neighbors of $C_i$ in $P_i$ get higher priority than non-neighbors; we break ties in left-to-right order, by sentence position). If we accept $v_i$, then $C_{i+1} \leftarrow C_i \cup v_i$; if not, $C_{i+1} \leftarrow C_i$. (Acceptance decisions are detailed in §4.3.) We continue adding vertexes to $C$ until either $P_i$ is empty or $C_i$ is $b$ characters long.[3] The appendix includes a formal algorithm.

VERTEX ADDITION is linear in the token length of $S$ because we pop and evaluate some vertex from $P_i$ at each timestep, after $P_0 \leftarrow S \setminus Q$. Additionally, because (1) we never accept $v_i$ if the length of $C_i \cup v_i$ is more than $b$, and (2) we set $C_0 \leftarrow Q$, our method respects $Q$ and $b$.

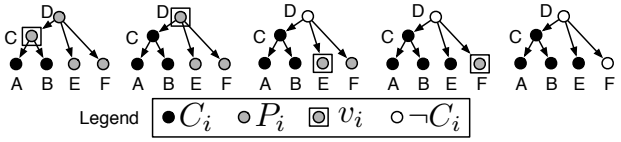


Figure 2: A dependency parse of a sentence $S$, shown across five timesteps of VERTEX ADDITION (from left to right). Each node in the parse is a vertex in $S$. Our stateful method produces the final compression {A,C,B,E} (rightmost). At each timestep, each candidate $v_i$ is boxed; rejected candidates $\neg C_i$ are unshaded.

## 4 Evaluation

We observe the latency, readability and token-level F1 score of VERTEX ADDITION, using a standard dataset (Filippova and Altun, 2013). We compare our method to an ILP baseline (§2) because ILP methods are the only known technique for constrained compression. All methods have similar

[1] Some methods compress via generation instead of deletion (Rush et al., 2015; Mallinson et al., 2018). Our extractive method is intended for practical, interpretable and trustworthy search systems (Chuang et al., 2012). Users might not trust abstractive summaries (Zhang and Cranshaw, 2018), particularly in cases with semantic error.

[2] ILPs are exponential in $|V|$ when selecting tokens (Clarke and Lapata, 2008) and exponential in $|E|$ when selecting edges (Filippova et al., 2015).

[3] We linearize $C$ by left-to-right vertex position in $S$, common for compression in English (Filippova and Altun, 2013).

compression ratios (shown in appendix), a well-known evaluation requirement (Napoles et al., 2011). We evaluate the significance of differences between VERTEX ADDITION$_{LR}$ and the ILP with bootstrap sampling (Berg-Kirkpatrick et al., 2012). All differences are significant ($p < .01$).

### 4.1 Constrained compression experiment

In order to evaluate different approaches to constrained compression, we require a dataset of sentences, constraints and known-good shortenings, which respect the constraints. This means we need tuples $(S, Q, b, C_g)$, where $C_g$ is a known-good compression of $S$ which respects $Q$ and $b$ (§1).

To support large-scale automatic evaluation, we reinterpret a standard compression corpus (Filippova and Altun, 2013) as a collection of input sentences and constrained compressions. The original dataset contains pairs of sentences $S$ and compressions $C_g$, generated using news headlines. For our experiment, we set $b$ equal to the character length of the gold compression $C_g$. We then sample a small number of nouns[4] from $C_g$ to form a query set $Q$, approximating both the observed number of tokens and observed parts of speech in real-world search (Jansen et al., 2000; Barr et al., 2008). Sampled $Q$ include reasonable queries like "police, Syracuse", "NHS" and "Hughes, manager, QPR".

By sampling queries and defining budgets in this manner, we create 198,570 training tuples and 9949 test tuples, each of the form $(S, Q, b, C_g)$. Filippova and Altun (2013) define the train/test split. We re-tokenize, parse and tag with CoreNLP v3.8.0 (Manning et al., 2014). We reserve 25,000 training tuples as a validation set.

### 4.2 Model: ILP

We compare our system to a baseline ILP method, presented in Filippova and Altun (2013). This approach represents each edge in a syntax tree with a vector of real-valued features, then learns feature weights using a structured perceptron trained on a corpus of $(S, C_g)$ pairs.[5] Learned weights are used to compute a global compression objective, subject to structural constraints which ensure $C$ is a valid tree. This baseline can easily perform constrained compression: at test time, we add optimization constraints specifying that $C$ must include $Q$, and not exceed length $b$.

To our knowledge, a public implementation of this method does not exist. We reimplement from scratch using Gurobi Optimization (2018), achieving a test-time, token-level F1 score of 0.76 on the unconstrained compression task, lower than the result (F1 = 84.3) reported by the original authors. There are some important differences between our reimplementation and original approach (described in detail in the appendix). Since VERTEX ADDITION requires $Q$ and $b$, we can only compare it to the ILP on the *constrained* (rather than traditional, unconstrained) compression task.

### 4.3 Models: VERTEX ADDITION

Vertex addition accepts or rejects some candidate vertex $v_i$ at each timestep $i$. We learn such decisions $y_i \in \{0, 1\}$ using a corpus of tuples $(S, Q, b, C_g)$ (§4.1). Given such a tuple, we can always execute an oracle path shortening $S$ to $C_g$ by first initializing VERTEX ADDITION and then, at each timestep: (1) choosing $v_i = h(P_i)$ and (2) adding $v_i$ to $C_i$ iff $v_i \in C_g$. We set $y_i = 1$ if $v_i \in C_g$; we set $y_i = 0$ if $v_i \notin C_g$. We then use decisions from oracle paths to train two models of inclusion decisions, $p(y_i = 1 | v_i, C_i, P_i, S)$. At test time, we accept $v_i$ if $p(y_i > .5)$.

**Model One.** Our VERTEX ADDITION$_{NN}$ model broadly follows neural approaches to transition-based parsing (e.g. Chen and Manning (2014)): we predict $y_i$ using a LSTM classifier with a standard max-pooling architecture (Conneau et al., 2017), implemented with a common neural framework (Gardner et al., 2017). Our classifier maintains four vocabulary embeddings matrixes, corresponding to the four disjoint subsets $C_i \cup \neg C_i \cup P_i \cup \{v_i\} = V$. Each LSTM input vector $x_t$ comes from the appropriate embedding for $v_t \in V$, depending on the state of the compression system at timestep $i$. The appendix details network tuning and optimization.

**Model Two.** Our VERTEX ADDITION$_{LR}$ model uses binary logistic regression,[6] with 3 classes of features.

Edge features describe the properties of the edge $(u, v_i)$ between $v_i \in P_i$ and $u \in C_i$. We

[4] 1 to 3 nouns; cardinality chosen uniformly at random.

[5] Another ILP (Wang et al., 2017) sets weights using a LSTM, achieving similar in-domain performance. This method requires a multi-stage computational process (i.e. run LSTM *then* ILP) that is poorly-suited to query-focused settings, where low latency is crucial.

[6] We implement with Python 3 using scikit-learn (Pedregosa et al., 2011). We tune the inverse regularization constant to $c = 10$ via grid search over powers of ten, to optimize validation set F1.

| Approach | F1 | SLOR | *Latency |
|---|---|---|---|
| RANDOM (lower bound) | 0.653 | 0.377 | 0.5 |
| ABLATED (edge only) | 0.827 | 0.669 | 3.7 |
| VERTEX ADDITION$_{NN}$ | 0.873 | 0.728 | 2929.1 (CPU) |
| ILP | 0.852 | 0.756 | 44.0 |
| VERTEX ADDITION$_{LR}$ | 0.881 | 0.745 | 4.1 |

Table 2: Test results for constrained compression. *Latency is the geometric mean of observed runtimes (in milliseconds per sentence). VERTEX ADDITION$_{LR}$ achieves the highest F1, and also runs 10.73 times faster than the ILP. Differences between all scores for VERTEX ADDITION$_{LR}$ and ILP are significant ($p < .01$).

use the edge-based feature function from Filippova and Altun (2013), described in detail in the appendix. This allows us to compare the performance of a vertex addition method based on local decisions with an ILP method that optimizes a global objective (§4.5), using the same feature set.

Stateful features represent the relationship between $v_i$ and the compression $C_i$ at timestep $i$. Stateful features include information such as the position of $v_i$ in the sentence, relative to the rightmost and left-most vertex in $C_i$, as well as history-based information such as the fraction of the character budget used so far. Such features allow the model to reason about which sort of $v_i$ should be added, given $Q$, $S$ and $C_i$.

Interaction features are formed by crossing all stateful features with the type of the dependency edge governing $v_i$, as well as with indicators identifying if $u$ governs $v_i$, if $v_i$ governs $u$ or if there is no edge $(u, v_i)$ in the parse.

### 4.4 Metrics: F1, Latency and SLOR

We measure the token-level F1 score of each compression method against gold compressions in the test set. F1 is the standard automatic evaluation metric for extractive compression (Filippova et al., 2015; Klerke et al., 2016; Wang et al., 2017).

In addition to measuring F1, researchers often evaluate compression systems with human *importance* and *readability* judgements (Knight and Marcu, 2000; Filippova et al., 2015). In our setting $Q$ determines the "important" information from $S$, so importance evaluations are inappropriate. To check readability, we use the automated readability metric SLOR (Lau et al., 2015), which correlates with human judgements (Kann et al., 2018).

We evaluate theoretical gains from VERTEX ADDITION (Table 1) by measuring empirical latency. For each compression method, we sample and compress $N = 300,000$ sentences, and record the runtime (in milliseconds per sentence). We observe that runtimes are distributed log-normally (Figure 3), and we thus summarize each sample using the geometric mean. ILP and VERTEX ADDITION$_{LR}$ share edge feature extraction code to support to fair comparison. We test VERTEX ADDITION$_{NN}$ using a CPU: the method is too slow for use in search applications in areas without access to specialized hardware (Table 2). The appendix further details latency and SLOR experiments.

### 4.5 Analysis: ABLATED & RANDOM

For comparison, we implement an ABLATED vertex addition method, which learns inclusion decisions using only edge features from Filippova and Altun (2013). ABLATED has a lower F1 score than ILP, which uses the same edge-level information to optimize a global objective: adding stateful and interaction features (i.e. VERTEX ADDITION$_{LR}$) improves F1 score. Nonetheless, strong performance from ABLATED hints that edge-level information alone (e.g. dependency type) can mostly guide acceptance decisions.

We also evaluate a RANDOM baseline, which accepts each $v_i$ randomly in proportion to $p(y_i = 1)$ across training data. RANDOM achieves reasonable F1 because (1) $C_0 = Q \in C_g$ and (2) F1 correlates with compression rate (Napoles et al., 2011), and $b$ is set to the length of $C_g$.

## 5 Future work: practical compression

This work presents a new method for fast query-focused sentence compression, motivated by the need for query-biased snippets in search engines (Tombros and Sanderson, 1998; Marchionini, 2006). While our approach shows promise in simulated experiments, we expect that further work will be required before the method can be employed for practical, user-facing search.

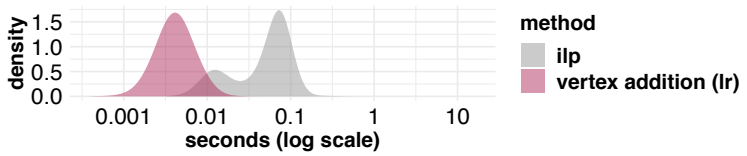


Figure 3: Density plot of log transformed latencies for VERTEX ADDITION$_{LR}$ (left) and ILP (right). Theoretical gains (Table 1) create real speedups. The ILP shows greater runtime variance, possibly reflecting varying approaches from Gurobi Optimization (2018).

To begin, both our technique and our evaluation ignore the conventions of search user interfaces, which typically display missing words using ellipses. This convention is important, because it allows snippet systems to transparently show users which words have been removed from a sentence. However, we observe that some well-formed compressions are difficult to read when displayed in this format. For instance the sentence "Aristide quickly fled Haiti in September 1991" can be shortened to the well-formed compression "Aristide fled in 1991." But this compression does not read fluidly when using ellipses ("Aristide...fled...in...1991"). Human experiments aimed at enumerating the desirable and undesirable properties of compressions displayed in ellipse format (e.g. compressions should minimize number of ellipses?) could help guide user-focused snippet algorithms in future work.

Our method also assumes access to a reliable, dependency parse, and ignores any latency penalties incurred from parsing. In practical settings, both assumptions are unreasonable. Like other NLP tools, dependency parsers often perform poorly on out-of-domain text (Bamman, 2017), and users looking to quickly investigate a new corpus might not wish to wait for a parser. Faster approaches based on low-latency part-of-speech tagging, or more cautious approaches based on syntactic uncertainty (Keith et al., 2018), each offer exciting possibilities for additional research.

Our approach also assumes that a user already knows a reasonable $b$ and reasonable $Q$ for a given sentence $S$.[7] However, in some cases, there is no well-formed shortening of which respects the requirements. For instance, if $Q$="Kennedy" and $b$=15 there is no reasonable shortening for the toy sentence "Kennedy kept running", because the compressions "Kennedy kept" and "Kennedy running" are not well-formed. We look forward to investigating which $(Q, S, b)$ triples will never return well-formed compressions in later work.

Finally, some shortened sentences will modify the meaning of a sentence, but we ignore this important complication in this initial study. In the future, we hope to apply ongoing research into textual entailment (Bowman et al., 2015; Pavlick and Callison-Burch, 2016; McCoy and Linzen, 2018) to develop semantically-informed approaches to the task.

## 6 Conclusion

We introduce a query-focused VERTEX ADDITION$_{LR}$ method for search user interfaces, with much lower theoretical complexity (and empirical runtimes) than baseline techniques. In search applications, such gains are non-trivial: real users are measurably hindered by interface lags (Nielsen, 1993; Liu and Heer, 2014). We hope that our fast, query-focused method better enables snippet creation at the "pace of human thought" (Heer and Shneiderman, 2012).

## Acknowledgments

Thanks to Javier Burroni and Nick Eubank for suggesting ways to optimize and measure performance of Python code. Thanks to Jeffrey Flanigan, Katie Keith and the UMass NLP reading group for feedback. This work was partially supported by IIS-1814955.

## Appendix

The appendix is available with the arXiv version of this paper: https://arxiv.org/abs/1904.09051

## References

Miguel Almeida and Andre Martins. 2013. Fast and robust compressive summarization with dual decomposition and multi-task learning. In *ACL*.

[7] Recall that we simulate $b$ and $Q$ based on the well-formed shortening $C_g$, see §4.1.

David Bamman. 2017. Natural language processing for the long tail. In *Digital Humanities*.

Cory Barr, Rosie Jones, and Moira Regelson. 2008. The linguistic structure of english web-search queries. In *EMNLP*.

Taylor Berg-Kirkpatrick, David Burkett, and Dan Klein. 2012. An empirical investigation of statistical significance in NLP. In *EMNLP*.

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*.

Ted Briscoe, John Carroll, and Rebecca Watson. 2006. The second release of the RASP system. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *EMNLP*.

Jason Chuang, Daniel Ramage, Christopher D. Manning, and Jeffrey Heer. 2012. Interpretation and trust: Designing model-driven visualizations for text analysis. In *CHI*.

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research*, 31:399–429.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Katja Filippova and Enrique Alfonseca. 2015. Fast k-best sentence compression. *arXiv preprint arXiv:1510.08418*.

Katja Filippova, Enrique Alfonseca, Carlos A Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with LSTMs. In *EMNLP*.

Katja Filippova and Yasemin Altun. 2013. Overcoming the lack of parallel data in sentence compression. In *EMNLP*. https://github.com/google-research-datasets/sentence-compression.

Katja Filippova and Michael Strube. 2008. Dependency tree based sentence compression. In *Proceedings of the Fifth International Natural Language Generation Conference*.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. AllenNLP: A deep semantic natural language processing platform.

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-up abstractive summarization. In *EMNLP*.

LLC Gurobi Optimization. 2018. Gurobi optimizer reference manual (v8).

Kenneth Heafield. 2011. KenLM: faster and smaller language model queries. In *EMNLP: Sixth Workshop on Statistical Machine Translation*.

Marti Hearst. 2009. *Search user interfaces*. Cambridge University Press, Cambridge New York.

Jeffrey Heer and Ben Shneiderman. 2012. Interactive dynamics for visual analysis. *Queue*, 10(2).

Bernard J. Jansen, Amanda Spink, and Tefko Saracevic. 2000. Real life, real users, and real needs: a study and analysis of user queries on the web. *Information Processing and Management*, 36:207–227.

Katharina Kann, Sascha Rothe, and Katja Filippova. 2018. Sentence-Level Fluency Evaluation: References Help, But Can Be Spared! In *CoNLL 2018*.

Katherine Keith, Su Lin Blodgett, and Brendan O'Connor. 2018. Monte Carlo syntax marginals for exploring and using dependency parses. In *NAACL*.

Yuta Kikuchi, Graham Neubig, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. 2016. Controlling output length in neural encoder-decoders. In *EMNLP*.

Sigrid Klerke, Yoav Goldberg, and Anders Søgaard. 2016. Improving sentence compression by learning to predict gaze. In *NAACL*.

Kevin Knight and Daniel Marcu. 2000. Statistics-based summarization - step one: Sentence compression. In *AAAI*.

Jey Han Lau, Alexander Clark, and Shalom Lappin. 2015. Unsupervised prediction of acceptability judgements. In *ACL*.

Zhicheng Liu and Jeffrey Heer. 2014. The effects of interactive latency on exploratory visual analysis. *IEEE Transactions on Visualization and Computer Graphics*, 20:2122–2131.

Jonathan Mallinson, Rico Sennrich, and Mirella Lapata. 2018. Sentence Compression for Arbitrary Languages via Multilingual Pivoting. In *EMNLP*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *ACL System Demonstrations*.

Gary Marchionini. 2006. Exploratory search: From finding to understanding. *Commun. ACM*, 49(4).

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *LREC*.

Thomas R. McCoy and Tal Linzen. 2018. Non-entailed subsequences as a challenge for natural language inference. In *Proceedings of the Society for Computation in Linguistics*.

Courtney Napoles, Benjamin Van Durme, and Chris Callison-Burch. 2011. Evaluating sentence compression: Pitfalls and suggested remedies. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*.

Jakob Nielsen. 1993. *Usability Engineering*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *International Conference on Parsing Technologies*.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan T. McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A multilingual treebank collection. In *LREC*.

Ellie Pavlick and Chris Callison-Burch. 2016. So-called non-subsective adjectives. In **SEM*.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830.

Matt Post and David Vilar. 2018. Fast lexically constrained decoding with dynamic beam allocation for neural machine translation. In *NAACL*.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *EMNLP*.

Sebastian Schuster and Christopher D. Manning. 2016. Enhanced english universal dependencies: An improved representation for natural language understanding tasks. In *LREC*.

Anastasios Tombros and Mark Sanderson. 1998. Advantages of query biased summaries in information retrieval. In *SIGIR*.

Liangguo Wang, Jing Jiang, Hai Leong Chieu, Chen Hui Ong, Dandan Song, and Lejian Liao. 2017. Can syntax help? Improving an LSTM-based sentence compression model for new domains. In *ACL*.

Amy X. Zhang and Justin Cranshaw. 2018. Making sense of group chat through collaborative tagging and summarization. *Proceedings of the ACM on Human-Computer Interaction*, 2(CSCW).

# Generating Personalized Recipes from Historical User Preferences

**Bodhisattwa Prasad Majumder**[*], **Shuyang Li**[*], **Jianmo Ni, Julian McAuley**
Computer Science and Engineering
University of California, San Diego
{bmajumde, shl008, jin018, jmcauley}@ucsd.edu

## Abstract

Existing approaches to recipe generation are unable to create recipes for users with culinary preferences but incomplete knowledge of ingredients in specific dishes. We propose a new task of *personalized recipe generation* to help these users: expanding a name and incomplete ingredient details into complete natural-text instructions aligned with the user's historical preferences. We attend on technique- and recipe-level representations of a user's previously consumed recipes, fusing these 'user-aware' representations in an attention fusion layer to control recipe text generation. Experiments on a new dataset of 180K recipes and 700K interactions show our model's ability to generate plausible and personalized recipes compared to non-personalized baselines.

## 1 Introduction

In the kitchen, we increasingly rely on instructions from cooking websites: recipes. A cook with a predilection for Asian cuisine may wish to prepare chicken curry, but may not know all necessary ingredients apart from a few basics. These users with limited knowledge cannot rely on existing recipe generation approaches that focus on creating coherent recipes given all ingredients and a recipe name (Kiddon et al., 2016). Such models do not address issues of personal preference (e.g. culinary tastes, garnish choices) and incomplete recipe details. We propose to approach both problems via *personalized generation* of plausible, user-specific recipes using user preferences extracted from previously consumed recipes.

Our work combines two important tasks from natural language processing and recommender systems: data-to-text generation (Gatt and Krahmer, 2018) and personalized recommendation (Rashid et al., 2002). Our model takes as user input the name of a specific dish, a few key ingredients, and a calorie level. We pass these loose input specifications to an encoder-decoder framework and attend on user profiles—learned latent representations of recipes previously consumed by a user—to generate a recipe *personalized* to the user's tastes. We fuse these 'user-aware' representations with decoder output in an attention fusion layer to jointly determine text generation. Quantitative (perplexity, user-ranking) and qualitative analysis on user-aware model outputs confirm that personalization indeed assists in generating plausible recipes from incomplete ingredients.

While personalized text generation has seen success in conveying user writing styles in the product review (Ni et al., 2017; Ni and McAuley, 2018) and dialogue (Zhang et al., 2018) spaces, we are the first to consider it for the problem of recipe generation, where output quality is heavily dependent on the *content* of the instructions—such as ingredients and cooking techniques.

To summarize, our main contributions are as follows:

1. We explore a new task of generating plausible and personalized recipes from incomplete input specifications by leveraging historical user preferences;[1]
2. We release a new dataset of 180K+ recipes and 700K+ user reviews for this task;
3. We introduce new evaluation strategies for generation quality in instructional texts, centering on quantitative measures of coherence. We also show qualitatively and quantitatively that personalized models generate high-quality and specific recipes that align with historical user preferences.

[*] denotes equal contribution

[1] Our source code and appendix are at https://github.com/majumderb/recipe-personalization



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5976–5982,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

## 2 Related Work

Large-scale transformer-based language models have shown surprising expressivity and fluency in creative and conditional long-text generation (Vaswani et al., 2017; Radford et al., 2019). Recent works have proposed hierarchical methods that condition on narrative frameworks to generate internally consistent long texts (Fan et al., 2018; Xu et al., 2018; Yao et al., 2018). Here, we generate procedurally structured recipes instead of free-form narratives.

Recipe generation belongs to the field of data-to-text natural language generation (Gatt and Krahmer, 2018), which sees other applications in automated journalism (Leppänen et al., 2017), question-answering (Agrawal et al., 2017), and abstractive summarization (Paulus et al., 2018), among others. Kiddon et al. (2015); Bosselut et al. (2018b) model recipes as a structured collection of ingredient entities acted upon by cooking actions. Kiddon et al. (2016) imposes a 'checklist' attention constraint emphasizing hitherto unused ingredients during generation. Yang et al. (2017) attend over explicit ingredient references in the prior recipe step. Similar hierarchical approaches that infer a full ingredient list to constrain generation will not help personalize recipes, and would be infeasible in our setting due to the potentially unconstrained number of ingredients (from a space of 10K+) in a recipe. We instead learn historical preferences to guide full recipe generation.

A recent line of work has explored user- and item-dependent aspect-aware review generation (Ni et al., 2017; Ni and McAuley, 2018). This work is related to ours in that it combines contextual language generation with personalization. Here, we attend over historical user preferences from previously consumed recipes to generate recipe content, rather than writing styles.

## 3 Approach

Our model's input specification consists of: the recipe name as a sequence of tokens, a partial list of ingredients, and a caloric level (high, medium, low). It outputs the recipe instructions as a token sequence: $\mathcal{W}_r = \{w_{r,0}, \ldots, w_{r,T}\}$ for a recipe $r$ of length $T$. To personalize output, we use historical recipe interactions of a user $u \in \mathcal{U}$.

**Encoder:** Our encoder has three embedding layers: vocabulary embedding $\mathcal{V}$, ingredient embedding $\mathcal{I}$, and caloric-level embedding $\mathcal{C}$. Each token in the (length $L_n$) recipe name is embedded via $\mathcal{V}$; the embedded token sequence is passed to a two-layered bidirectional GRU (BiGRU) (Cho et al., 2014), which outputs hidden states for names $\{\mathbf{n}_{\text{enc},j} \in \mathbb{R}^{2d_h}\}$, with hidden size $d_h$. Similarly each of the $L_i$ input ingredients is embedded via $\mathcal{I}$, and the embedded ingredient sequence is passed to another two-layered BiGRU to output ingredient hidden states as $\{\mathbf{i}_{\text{enc},j} \in \mathbb{R}^{2d_h}\}$. The caloric level is embedded via $\mathcal{C}$ and passed through a projection layer with weights $W_c$ to generate calorie hidden representation $\mathbf{c}_{\text{enc}} \in \mathbb{R}^{2d_h}$.

**Ingredient Attention:** We apply attention (Bahdanau et al., 2015) over the encoded ingredients to use encoder outputs at each decoding time step. We define an attention-score function $\alpha$ with key $K$ and query $Q$:

$$\alpha(K, Q) = \frac{\exp\left(\tanh\left(W_\alpha\left[K + Q\right] + \mathbf{b}_\alpha\right)\right)}{Z},$$

with trainable weights $W_\alpha$, bias $\mathbf{b}_\alpha$, and normalization term $Z$. At decoding time $t$, we calculate the ingredient context $\mathbf{a}_t^i \in \mathbb{R}^{d_h}$ as:

$$\mathbf{a}_t^i = \sum_{j=1}^{L_i} \alpha\left(\mathbf{i}_{\text{enc},j}, \mathbf{h}_t\right) \times \mathbf{i}_{\text{enc},j}.$$

**Decoder:** The decoder is a two-layer GRU with hidden state $h_t$ conditioned on previous hidden state $h_{t-1}$ and input token $w_{r,t}$ from the original recipe text. We project the concatenated encoder outputs as the initial decoder hidden state:

$$\mathbf{h}_0\left(\in \mathbb{R}^{d_h}\right) = W_{h_0}\left[\mathbf{n}_{\text{enc},L_n}; \mathbf{i}_{\text{enc},L_i}; \mathbf{c}_{\text{enc}}\right] + \mathbf{b}_{h_0}$$
$$\mathbf{h}_t, \mathbf{o}_t = \text{GRU}\left(\left[w_{r,t}; \mathbf{a}_t^i\right], \mathbf{h}_{t-1}\right).$$

To bias generation toward user preferences, we attend over a user's previously reviewed recipes to jointly determine the final output token distribution. We consider two different schemes to model preferences from user histories: (1) recipe interactions, and (2) techniques seen therein (defined in Section 4). Rendle et al. (2009); Quadrana et al. (2018); Ueda et al. (2011) explore similar schemes for personalized recommendation.

**Prior Recipe Attention:** We obtain the set of prior recipes for a user $u$: $R_u^+$, where each recipe can be represented by an embedding from a recipe embedding layer $\mathcal{R}$ or an average of the name tokens embedded by $\mathcal{V}$. We attend over the $k$-most recent prior recipes, $R_u^{k+}$, to account for temporal drift of user preferences (Moore et al., 2013).

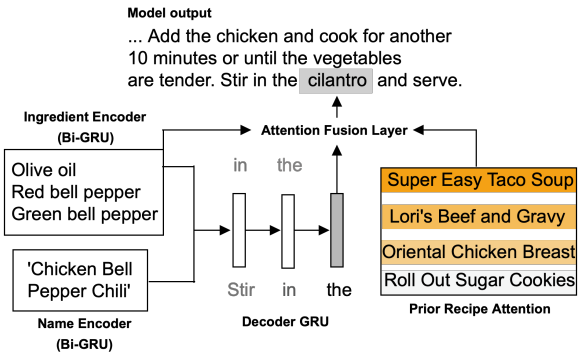


Figure 1: Sample data flow through model architecture. Emphasis on prior recipe attention scores (darker is stronger). Ingredient attention omitted for clarity.

These embeddings are used in the '**Prior Recipe**' and '**Prior Name**' models, respectively.

Given a recipe representation $\mathbf{r} \in \mathbb{R}^{d_r}$ (where $d_r$ is recipe- or vocabulary-embedding size depending on the recipe representation) the *prior recipe attention* context $\mathbf{a}_t^{r_u}$ is calculated as

$$\mathbf{a}_t^{r_u} = \sum_{r \in R_u^{k+}} \alpha\left(\mathbf{r}, \mathbf{h}_t\right) \times \mathbf{r}.$$

**Prior Technique Attention:** We calculate prior technique preference (used in the '**Prior Tech**' model) by normalizing co-occurrence between users and techniques seen in $R_u^+$, to obtain a preference vector $\boldsymbol{\rho}_u$. Each technique $x$ is embedded via a technique embedding layer $\mathcal{X}$ to $\mathbf{x} \in \mathbb{R}^{d_x}$. *Prior technique attention* is calculated as

$$\mathbf{a}_t^{x_u} = \sum_{x \text{ seen in } R_u^+} \left(\alpha\left(\mathbf{x}, \mathbf{h}_t\right) + \rho_{u,x}\right) \times \mathbf{x},$$

where, inspired by copy mechanisms (See et al., 2017; Gu et al., 2016), we add $\rho_{u,x}$ for technique $x$ to emphasize the attention by the user's prior technique preference.

**Attention Fusion Layer:** We fuse all contexts calculated at time $t$, concatenating them with decoder GRU output and previous token embedding:

$$\mathbf{a}_t^f = \text{ReLU}\left(W_f\left[w_{r,t}; \mathbf{o}_t; \mathbf{a}_t^i; (\mathbf{a}_t^{r_u} \text{ or } \mathbf{a}_t^{x_u})\right] + \mathbf{b}_f\right).$$

We then calculate the token probability:

$$P(S_{r,t}) = \text{softmax}\left(W_P[\mathbf{a}_t^f] + \mathbf{b}_P\right),$$

and maximize the log-likelihood of the generated sequence conditioned on input specifications and user preferences. Figure 1 shows a case where the Prior Name model attends strongly on previously consumed savory recipes to suggest the usage of an additional ingredient ('cilantro').

| Split | # Users | # Recipes | # Actions | Sparsity[3] |
|---|---|---|---|---|
| Train | 25,076 | 160,901 | 698,901 | 99.983% |
| Dev | 7,023 | 6,621 | 7,023 | – |
| Test | 12,455 | 11,695 | 12,455 | – |

Table 1: Statistics of Food.com interactions

## 4 Recipe Dataset: Food.com

We collect a novel dataset of 230K+ recipe texts and 1M+ user interactions (reviews) over 18 years (2000-2018) from Food.com.[2] Here, we restrict to recipes with at least 3 steps, and at least 4 and no more than 20 ingredients. We discard users with fewer than 4 reviews, giving 180K+ recipes and 700K+ reviews, with splits as in Table 1.

Our model must learn to generate from a diverse recipe space: in our training data, the average recipe length is 117 tokens with a maximum of 256. There are 13K unique ingredients across all recipes. Rare words dominate the vocabulary: 95% of words appear <100 times, accounting for only 1.65% of all word usage. As such, we perform Byte-Pair Encoding (BPE) tokenization (Sennrich et al., 2016; Radford et al., 2018), giving a training vocabulary of 15K tokens across 19M total mentions. User profiles are similarly diverse: 50% of users have consumed ≤6 recipes, while 10% of users have consumed >45 recipes.

We order reviews by timestamp, keeping the most recent review for each user as the test set, the second most recent for validation, and the remainder for training (sequential leave-one-out evaluation (Kang and McAuley, 2018)). We evaluate only on recipes not in the training set.

We manually construct a list of 58 cooking techniques from 384 cooking actions collected by Bosselut et al. (2018b); the most common techniques (*bake*, *combine*, *pour*, *boil*) account for 36.5% of technique mentions. We approximate technique adherence via string match between the recipe text and technique list.

## 5 Experiments and Results

For training and evaluation, we provide our model with the first 3-5 ingredients listed in each recipe. We decode recipe text via top-$k$ sampling (Radford et al., 2019), finding $k = 3$ to produce satisfactory results. We use a hidden size $d_h = 256$

[2] https://www.kaggle.com/shuyangli94/food-com-recipes-and-user-interactions

[3] Ratio of unobserved actions to all possible actions.

| Model | BPE PPL | BLEU-1 | BLEU-4 | ROUGE-L | D-1 (%) | D-2 (%) | UMA | MRR | PP (%) |
|---|---|---|---|---|---|---|---|---|---|
| NN | – | 20.279 | 0.465 | 16.871 | 0.931 | 9.394 | 0.100 | 0.293 | – |
| Enc-Dec | 9.611 | 28.391 | **3.385** | **25.001** | 0.220 | 1.928 | 0.100 | 0.293 | – |
| Prior Tech | 9.572 | **28.864** | 3.312 | 24.920 | 0.233 | **2.158** | 0.128 | 0.319 | 62.821 |
| Prior Recipe | 9.551 | 27.858 | 3.215 | 24.822 | 0.231 | 2.062 | 0.302 | 0.412 | **66.026** |
| Prior Name | **9.516** | 28.046 | 3.211 | 24.794 | **0.233** | 2.080 | **0.505** | **0.628** | 61.165 |

Table 2: Metrics on generated recipes from test set. D-1/2 = Distinct-1/2, UMA = User Matching Accuracy, MRR = Mean Reciprocal Rank, PP = Pairwise preference over baseline (evaluated for 310 recipe pairs per model).

| | |
|---|---|
| **Input** | **Name:** Pomberrytini; **Ingredients:** pomegranate-blueberry juice, cranberry juice, vodka ; **Calorie:** Low |
| **Gold** | Place everything except the orange slices in a cocktail shaker. Shake until well mixed and well chilled. Pour into martini glasses and float an orange slice in each glass. |
| **Enc-Dec** | Combine all ingredients. Cover and refrigerate. Serve with whipped topping. |
| **Prior Tech** | Combine all ingredients. Store in refrigerator. Serve over ice. Enjoy! |
| **Prior Recipe** | Pour the ice into a cocktail shaker. Pour in the vodka and vodka. Add a little water and shake to mix. Pour into the glass and garnish with a slice of orange slices. Enjoy! |
| **Prior Name** | Combine all ingredients except for the ice in a blender or food processor. Process to make a smooth paste and then add the remaining vodka and blend until smooth. Pour into a chilled glass and garnish with a little lemon and fresh mint. |

Table 3: Sample generated recipe. Emphasis on personalization and explicit ingredient mentions via highlights.

for both the encoder and decoder. Embedding dimensions for vocabulary, ingredient, recipe, techniques, and caloric level are 300, 10, 50, 50, and 5 (respectively). For prior recipe attention, we set $k = 20$, the 80th %-ile for the number of user interactions. We use the Adam optimizer (Kingma and Ba, 2015) with a learning rate of $10^{-3}$, annealed with a decay rate of 0.9 (Howard and Ruder, 2018). We also use teacher-forcing (Williams and Zipser, 1989) in all training epochs.

In this work, we investigate how leveraging historical user preferences can improve generation quality over strong baselines in our setting. We compare our personalized models against two baselines. The first is a name-based Nearest-Neighbor model (**NN**). We initially adapted the Neural Checklist Model of Kiddon et al. (2016) as a baseline; however, we ultimately use a simple Encoder-Decoder baseline with ingredient attention (**Enc-Dec**), which provides comparable performance and lower complexity. All personalized models outperform baseline in BPE perplexity (Table 2) with Prior Name performing the best. While our models exhibit comparable performance to baseline in BLEU-1/4 and ROUGE-L, we generate more diverse (Distinct-1/2: percentage of distinct unigrams and bigrams) and acceptable recipes. BLEU and ROUGE are not the most appropriate metrics for generation quality. A 'correct' recipe can be written in many ways with the same main entities (ingredients). As BLEU-1/4 capture structural information via n-gram matching, they are not correlated with subjective recipe quality. This mirrors observations from Baheti et al. (2018); Fan et al. (2018).

We observe that personalized models make more diverse recipes than baseline. They thus perform better in BLEU-1 with more key entities (ingredient mentions) present, but worse in BLEU-4, as these recipes are written in a personalized way and deviate from gold on the phrasal level. Similarly, the 'Prior Name' model generates more unigram-diverse recipes than other personalized models and obtains a correspondingly lower BLEU-1 score.

**Qualitative Analysis:** We present sample outputs for a cocktail recipe in Table 3, and additional recipes in the appendix. Generation quality progressively improves from generic baseline output to a blended cocktail produced by our best performing model. Models attending over prior recipes explicitly reference ingredients. The Prior Name model further suggests the addition of lemon and mint, which are reasonably associated with previously consumed recipes like coconut mousse and pork skewers.

**Personalization:** To measure personalization, we evaluate how closely the generated text corresponds to a particular user profile. We compute the likelihood of generated recipes using identical input specifications but conditioned on ten different user profiles—one 'gold' user who consumed the original recipe, and nine randomly generated user profiles. Following Fan et al. (2018), we expect the highest likelihood for the recipe conditioned on the gold user. We measure user matching accuracy (UMA)—the proportion where the gold user is ranked highest—and Mean Reciprocal Rank (MRR) (Radev et al., 2002) of the gold user. All personalized models beat baselines in both metrics, showing our models personalize generated recipes to the given user profiles. The Prior Name model achieves the best UMA and MRR by a large margin, revealing that prior recipe names are strong signals for personalization. Moreover, the addition of attention mechanisms to capture these signals improves language modeling performance over a strong non-personalized baseline.

**Recipe Level Coherence:** A plausible recipe should possess a coherent step order, and we evaluate this via a metric for recipe-level coherence. We use the neural scoring model from Bosselut et al. (2018a) to measure recipe-level coherence for each generated recipe. Each recipe step is encoded by BERT (Devlin et al., 2019). Our scoring model is a GRU network that learns the overall recipe step ordering structure by minimizing the cosine similarity of recipe step hidden representations presented in the correct and reverse orders. Once pretrained, our scorer calculates the similarity of a generated recipe to the forward and backwards ordering of its corresponding gold label, giving a score equal to the difference between the former and latter. A higher score indicates better step ordering (with a maximum score of 2). Table 4 shows that our personalized models achieve average recipe-level coherence scores of 1.78-1.82, surpassing the baseline at 1.77.

**Recipe Step Entailment:** Local coherence is also crucial to a user following a recipe: it is crucial that subsequent steps are logically consistent with prior ones. We model local coherence as an entailment task: predicting the likelihood that a recipe step follows the preceding. We sample several consecutive (positive) and non-consecutive (negative) pairs of steps from each recipe. We train a BERT (Devlin et al., 2019) model to predict the entailment score of a pair of steps separated by a `[SEP]` token, using the final representation of the `[CLS]` token. The step entailment score is computed as the average of scores for each set of consecutive steps in each recipe, averaged over every generated recipe for a model, as shown in Table 4.

| Model | Recipe Level Coherence | Recipe Step Entailment |
|---|---|---|
| Enc-Dec | 1.77 | 0.72 |
| Prior Tech | 1.78 | 0.73 |
| Prior Recipe | 1.80 | 0.76 |
| Prior Name | **1.82** | **0.78** |

Table 4: Coherence metrics on generated recipes from test set.

**Human Evaluation:** We presented 310 pairs of recipes for pairwise comparison (Fan et al., 2018) (details in appendix) between baseline and each personalized model, with results shown in Table 2. On average, human evaluators preferred personalized model outputs to baseline 63% of the time, confirming that personalized attention improves the semantic plausibility of generated recipes. We also performed a small-scale human coherence survey over 90 recipes, in which 60% of users found recipes generated by personalized models to be more coherent and preferable to those generated by baseline models.

## 6 Conclusion

In this paper, we propose a novel task: to generate personalized recipes from incomplete input specifications and user histories. On a large novel dataset of 180K recipes and 700K reviews, we show that our personalized generative models can generate plausible, personalized, and coherent recipes preferred by human evaluators for consumption. We also introduce a set of automatic coherence measures for instructional texts as well as personalization metrics to support our claims. Our future work includes generating structured representations of recipes to handle ingredient properties, as well as accounting for references to collections of ingredients (e.g. "dry mix").

**Acknowledgements.** This work is partly supported by NSF #1750063. We thank all reviewers for their constructive suggestions, as well as Rei M., Sujoy P., Alicia L., Eric H., Tim S., Kathy C., Allen C., and Micah I. for their feedback.

## References


Aishwarya Agrawal, Jiasen Lu, Stanislaw Antol, Margaret Mitchell, C. Lawrence Zitnick, Devi Parikh, and Dhruv Batra. 2017. VQA: visual question answering. *IJCV*, 123(1):4–31.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Ashutosh Baheti, Alan Ritter, Jiwei Li, and Bill Dolan. 2018. Generating more interesting responses in neural conversation models with distributional constraints. In *EMNLP*.

Antoine Bosselut, Asli Çelikyilmaz, Xiaodong He, Jianfeng Gao, Po-Sen Huang, and Yejin Choi. 2018a. Discourse-aware neural rewards for coherent text generation. In *NAACL-HLT*.

Antoine Bosselut, Omer Levy, Ari Holtzman, Corin Ennis, Dieter Fox, and Yejin Choi. 2018b. Simulating action dynamics with neural process networks. In *ICLR*.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT 2019*.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *ACL*.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *J. Artif. Intell. Res.*, 61:65–170.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *ACL*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *ACL*.

Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *ICDM*.

Chloé Kiddon, Ganesa Thandavam Ponnuraj, Luke Zettlemoyer, and Yejin Choi. 2015. Mise en place: Unsupervised interpretation of instructional recipes. In *EMNLP*.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally coherent text generation with neural checklist models. In *EMNLP*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Leo Leppänen, Myriam Munezero, Mark Granroth-Wilding, and Hannu Toivonen. 2017. Data-driven news generation for automated journalism. In *INLG*.

Joshua L. Moore, Shuo Chen, Douglas Turnbull, and Thorsten Joachims. 2013. Taste over time: The temporal dynamics of user preferences. In *ISMIR*.

Jianmo Ni, Zachary C. Lipton, Sharad Vikram, and Julian McAuley. 2017. Estimating reactions and recommending products with generative models of reviews. In *IJCNLP*.

Jianmo Ni and Julian McAuley. 2018. Personalized review generation by expanding phrases and attending on aspect-aware representations. In *ACL*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *ICLR*.

Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. 2018. Sequence-aware recommender systems. In *UMAP*.

Dragomir R. Radev, Hong Qi, Harris Wu, and Weiguo Fan. 2002. Evaluating web-based question answering systems. In *LREC*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Al Mamunur Rashid, Istvan Albert, Dan Cosley, Shyong K. Lam, Sean M. McNee, Joseph A. Konstan, and John Riedl. 2002. Getting to know you: learning new user preferences in recommender systems. In *IUI*.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: bayesian personalized ranking from implicit feedback. In *UAI*.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *ACL*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL*.

Mayumi Ueda, Mari Takahata, and Shinsuke Nakajima. 2011. User's food preference extraction for personalized cooking recipe recommendation. In *SPIM*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Ronald J. Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280.

Jingjing Xu, Xuancheng Ren, Yi Zhang, Qi Zeng, Xiaoyan Cai, and Xu Sun. 2018. A skeleton-based model for promoting coherence among sentences in narrative story generation. In *EMNLP*.

Zichao Yang, Phil Blunsom, Chris Dyer, and Wang Ling. 2017. Reference-aware language models. In *EMNLP*.

Lili Yao, Nanyun Peng, Ralph M. Weischedel, Kevin Knight, Dongyan Zhao, and Rui Yan. 2018. Plan-and-write: Towards better automatic storytelling. *CoRR*, abs/1811.05701.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? In *ACL*.

# Generating Highly Relevant Questions

**Jiazuo Qiu** and **Deyi Xiong***
School of Computer Science and Technology, Soochow University, Suzhou, China
qjzhzw@163.com; dyxiong@suda.edu.cn

## Abstract

The neural seq2seq based question generation (QG) is prone to generating generic and undiversified questions that are poorly relevant to the given passage and target answer. In this paper, we propose two methods to address the issue. (1) By a partial copy mechanism, we prioritize words that are morphologically close to words in the input passage when generating questions; (2) By a QA-based reranker, from the n-best list of question candidates, we select questions that are preferred by both the QA and QG model. Experiments and analyses demonstrate that the proposed two methods substantially improve the relevance of generated questions to passages and answers.

## 1 Introduction

Question generation is to generate a valid and fluent question according to a given passage and the target answer. In general, the answer is a span of words in the passage. QG can be used in many scenarios, such as automatical tutoring systems, improving the performance of QA models and enabling chatbots to lead a conversation.

In the early days, QG has been tackled mainly via rule-based approaches (Mitkov and Ha, 2003; Heilman and Smith, 2010). These methods rely on many handcrafted rules and templates. Constructing such rules is time-consuming and it is difficult to adapt them to other domains.

Very recently, neural networks have been used for QG. Specifically, the encoder-decoder seq2seq model (Du et al., 2017; Zhou et al., 2017; Song et al., 2018) is used to encode a passage and generate a question corresponding to the answer.

Such end-to-end neural models are able to generate better questions than traditional rule-based approaches. However, one issue with the current neural models is that the generated questions are not quite relevant to the corresponding passages and target answers. In other words, the neural QG models tend to generate generic questions (e.g., "what is the name of...?").

In this paper, we propose two methods to deal with this low-relevance issue for QG. First, we present a partial copy method to enhance the existing copy mechanism so that the QG model can not only copy a word as an entire unit from the passage to the generated question but also copy a part of a word (e.g., "start" in "started") to generate a new morphological form of the word in the question. The fine-grained partial copy mechanism enables the QG model to copy morphologically changed words from the passage, increasing the relevance of the generated question to the passage by sharing more words in different forms. Second, we propose a QA-based reranker to rerank QG results. Particularly, we use a neural QA model to evaluate the quality of generated questions, and rerank them according to the QA model scores. Normally, generic questions get low scores from the neural QA model, hence the reranker is able to select non-generic and highly relevant questions.

While alleviating the low-relevance issue, the proposed two methods alone and their combination have also improved the quality of generated questions in terms of both BLEU and METEOR in our experiments.

## 2 Methods

### 2.1 The Partial Copy Mechanism

The conventional copy mechanism (Gu et al., 2016; See et al., 2017) can allow the decoder to copy a word from the input passage to the generated question. However, such copy mechanism works at the word level. It cannot copy a part of a word to reproduce an appropriate form of the

*Corresponding author



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5983–5987, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

word in the generated question. In other words, it cannot copy words with morphological changes. However, morphological changes frequently happen when we transform a passage into a question as grammatical functions of some words (e.g., verbs, nouns, adjectives, etc) change.

Let's consider the following example:

> Passage: Teaching started in 1794.
> Question: When did teaching start?

The morphological variants "start", "starts" of "started" in the passage may be used in the generated question.

In order to encourage the decoder to copy the inflected forms of a word from the passage to the generated question, we propose a partial copy mechanism that measures the character overlap rate of an original word in the passage with its copied form in the question. We first detect the overlapped subsequence of characters between words $w_1$ and $w_2$ according to the longest common subsequence (LCS) between them. For example, "start" and "started" have LCS "start". Then we calculate the overlap rate $C$ between $w_1$ and $w_2$ as follows.

$$C = \frac{|LCS| * 2}{|w_1| + |w_2|} \quad (1)$$

According to this formula, the overlap rate between "start" and "started" is 0.71. The value range of $C$ is $\in [0, 1]$. Full-word copy is of course encouraged as the value of $C$ is 1.

One problem with $C$ is that many unrelated words also have LCS. For example, "a" is the LCS of "append" and "start". The overlap rate of the two words is 0.18 instead of 0. To ensure the effectiveness of the method, we compute the final overlap rate with a threshold $\gamma$ :

$$C = \begin{cases} C, & \text{if } C >= \gamma \\ 0, & \text{if } C < \gamma \end{cases} \quad (2)$$

Whenever we generate a word in the question, we find its corresponding word in the passage with the highest attention weight. We then calculate the overlap rate $C$ between the two words and use $C$ to re-adjust the probability of the generated word as follows:

$$P_{adj} = P * (1 + \lambda_1 * C) \quad (3)$$

where $P$ is the original generation probability output by the decoder, $\lambda_1$ is a hyperparameter whose range is [0, $+\infty$). We normalize all these re-adjusted probabilities to get the final probability distribution.

### 2.2 The QA-Based Reranking

Since we use the beam search algorithm in the neural QG decoder, we can generate multiple question candidates. We find that the generated question with the highest probability according to the baseline neural QG model is not always the best question.

Partially inspired by Li et al. (2016), we propose a QA-based reranker to rerank the n-best questions generated by the baseline decoder.

In general, the task of the QG model is to estimate the probability $P(q|p, a)$ of a generated question $q$ given the passage $p$ and target answer $a$. In the QA-based reranker, we re-estimate the quality of a candidate question by calculating the probability of the target answer $a$ given the passage and the generated question $q$, i.e., $P(a|p, q)$.

In theory, we can combine the two probabilities to rerank generated questions. But in practice, we take a more intuitive and straightforward way to use the $F_1$ score of a predicted answer by the trained QA model according to the generated question. The $F_1$ score is calculated at the character level by comparing the generated answer and gold answer (considering the answers as a set of characters). The idea behind this is that a good question allows the QA model to easily find an answer close to the ground truth answer.

The score used to rerank a question candidate is therefore computed as:

$$score = (1 - \lambda_2) * score_1 + \lambda_2 * score_2 \quad (4)$$

where $score_1$ is the log probability of the candidate question estimated by the baseline QG model, $score_2$ is the $F_1$ score of the predicted answer by the QA model, and $\lambda_2$ is a hyperparameter whose range is [0, 1].

## 3 Experiments

### 3.1 Datasets

Following previous work, we conducted our experiments on SQuAD (Rajpurkar et al., 2016), a QA dataset which can also be used for QG. The dataset contains 536 articles and over 100k questions. Since the test set is unavailable, Du et al. (2017) randomly divide the raw dataset into

| models | BLEU-1 | BLEU-2 | BLEU-3 | BLEU-4 | METEOR |
|---|---|---|---|---|---|
| Du et al. (2017) | 43.09 | 25.96 | 17.50 | 12.28 | 16.62 |
| Song et al. (2018) (reported in paper) | – | – | – | 13.98 | 18.77 |
| Song et al. (2018) (our re-running) | 42.15 | 27.21 | 19.53 | 14.56 | 19.15 |
| partial copy ($\lambda_1$=0.5) | 44.13 | 28.29 | 20.17 | 14.95 | 19.81 |
| partial copy ($\lambda_1$=1) | 44.33 | 28.34 | 20.17 | 14.95 | 20.00 |
| partial copy ($\lambda_1$=2) | 42.34 | 26.95 | 19.15 | 14.16 | **20.16** |
| QA-based reranking ($\lambda_2$=0.2) | 42.64 | 27.57 | 19.77 | 14.72 | 19.43 |
| QA-based reranking ($\lambda_2$=0.5) | 42.50 | 27.42 | 19.62 | 14.56 | 19.38 |
| QA-based reranking ($\lambda_2$=0.8) | 42.43 | 27.34 | 19.54 | 14.48 | 19.34 |
| partial copy + QA-based reranking | **44.61** | **28.78** | **20.59** | **15.29** | 20.13 |

Table 1: Experiment results on the test set.

train/dev/test set. In our experiments, we used the same data split as Du et al. (2017).

### 3.2 Baseline and Settings

Our baseline is based on the QG model proposed by Song et al. (2018). To be specific, it is a seq2seq model with attention and copy mechanism. The model consists of two encoders and a decoder. The two encoders encode a passage = ($p_1$,...,$p_M$) and an answer = ($a_1$,...,$a_N$) respectively. Additionally, multi-perspective matching strategies are used to combine the two encoders. With information from the encoders, the decoder generates a question word by word.

We retained the same values for most hyperparameters in our experiments as the baseline system (Song et al., 2018). We used Glove (Pennington et al., 2014) to initialize the word embeddings and trained the model for 10 epochs. Copy and coverage mechanism (See et al., 2017) were included while additional lexical features (POS, NER) were not. We used adam (Kingma and Ba, 2015) as the optimizer during training. The beam size was set to 20 for the decoder.

In the experiments of the partial copy mechanism, we set the threshold $\gamma$ to 0.7. Three values (0.5, 1 and 2) were tried for $\lambda_1$.

In the experiments of the QA-based reranking, we used the SAN model (Liu et al., 2018) as the QA model. 0.2, 0.5 and 0.8 were tried for $\lambda_2$.

We used automatic evaluation metrics: BLEU-1, BLEU-2, BLEU-3, BLEU-4 (Papineni et al., 2002) and METEOR (Denkowski and Lavie, 2014). The calculation script was provided by Du et al. (2017).

### 3.3 Results

Experiment results are shown in Table 1, from which we observe that our methods substantially improves the baseline in terms of all evaluation metrics.

The combination of the proposed two methods ($\lambda_1$ = 1, $\lambda_2$ = 0.2) achieve the best performance, gaining 0.73 BLEU-4 and nearly 1 METEOR point of improvements over the baseline (obtained by re-running the source code of the baseline, higher than the results reported in the paper (Song et al., 2018)). The proposed partial copy mechanism alone obtain substantial improvements over the baseline, especially in terms of BLEU-1 and BLEU-2. This is because this method is able to help the decoder copy morphologically changed words from passages. The application of the QA-based reranking obtain further improvements over the partial copy mechanism, indicating that the two methods are complementary to each other.

## 4 Analysis

**Partial copy**: We use the method to enhance the existing copy mechanism, making generated questions more relevant to passages and target answers. The average proportion of words (or their other morphological forms) that are copied from the passages in generated questions increases from 75.49% (the baseline) to 78.74%. Under such a mechanism, generic questions such as "what is the name of...?" will be penalized by our method as the overlap rate between words in these generic questions and those in passages is low. On the contrary, questions with higher overlap rate and therefore higher relevance to the input passage are rewarded by the new copy mechanism. In order to testify this hypothesis, we counted the numbers

| Question templates | Song et al. (2018) | Our model |
|---|---|---|
| What is/was the name of ...? | 8,180/13,740 | 5,618/9,990 |
| What type of ...? | 4,942 | 3,805 |
| What is/was another name ...? | 2,731/26 | 57/3 |
| What is/was the total ...? | 470/794 | 101/62 |
| What is/was it ...? | 251/57 | 111/32 |

Table 2: Frequency of generic questions generated by the baseline and our methods.

| |
|---|
| **Passage**: in the polytechnic sector : wellington polytechnic amalgamated with massey university . |
| **Answer**: wellington polytechnic |
| **Question**: what school did massey university combine with ? |
| **Baseline**: what is the name of the polytechnic sector in the polytechnic ? |
| **Partial copy**: who amalgamated with massey university in the polytechnic sector ? |

| |
|---|
| **Passage**: in practice , catholic services in all provinces were quickly forbidden , and the reformed church became the " public " or " privileged " church in the republic . |
| **Answer**: catholic services |
| **Question**: what was forbidden in all provinces ? |
| **Baseline**: what was the " public " church in the republic ? |
| **QA-based reranking**: what was forbidden in all provinces in the republic ? |

Table 3: QG examples.

of such generic questions generated by the baseline and our method, which are shown in Table 2. It is clearly seen that the number of generic questions is significantly decreased after the partial copy mechanism is used.

In the first example displayed in Table 3, the phrase "what school" is difficult to be generated as it does not appear in the input passage. The baseline model generates the generic question "what is the name of...?", which is not relevant to the passage and target answer. Such generic questions are generated because the templates of these questions occur frequently in the training data. The trained seq2seq model is prone to generating these "safe" questions, similar to the undiversified response generation in seq2seq-based dialogue model (Li et al., 2016). In contrast, our model is able to generate a more relevant question including a rare word "amalgamated" as the word has a high overlap rate.

**QA-based reranking**: In our experiments, a total of 2,099 questions were reranked. Among them, 1,117 examples achieve a higher BLEU-4 score after reranking, while only 821 examples have a lower BLEU-4 after reranking.

In the second example of Table 3, the question with the highest score generated by the baseline is "what church", while the ground truth question is asking "what was forbidden". Since the generated questions to be reranked are different to each other, the QA model naturally finds different answers to these questions. For "what church", the QA model detected "reformed church" as the answer while for "what was forbidden", the QA model correctly detected the target answer "catholic services". Therefore, the QA-based reranker is able to find the answer-relevant questions.

## 5 Related Work

The neural QG is an emerging task. Unlike the extractive QA, most neural QG models are generative. Du et al. (2017) pioneer the neural QG by proposing neural seq2seq models to deal with the task. Unfortunately, they do not use the target answer for QG. At the same time, Zhou et al. (2017) present a similar model for QG. They use answer position embeddings to represent target answers and explore a variety of lexical features.

After that, many QG studies have been conducted on the basis of the widely-used seq2seq architecture together with the attention and copy mechanism. Song et al. (2018) propose two encoders for both the passage and the target answer. Du and Cardie (2018) employ coreferences as an additional feature. Kim et al. (2019) propose a model of answer separation. Yuan et al. (2017) and Kumar et al. (2018) adopt reinforcement learning to optimize the generation process.

QA and QG are closely related to each other. Tang et al. (2017) treat QA and QG as dual tasks, and many other studies use QG to enhance QA or jointly learn QG and QA (Duan et al., 2017; Wang et al., 2017; Sachan and Xing, 2018).

## 6 Conclusion

In this paper, we have presented two methods to improve the relevance of generated questions to the given passage and target answer. Experiments and analyses on SQuAD show that both the partial copy mechanism and QA-based reranking improve the relevance of generated questions in terms of both BLEU and METEOR.

## Acknowledgements

The present research was supported by the National Natural Science Foundation of China (Grant No. 61622209). We would like to thank the anonymous reviewers for their insightful comments.

## References

Michael J Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. pages 376–380.

Xinya Du and Claire Cardie. 2018. Harvesting paragraph-level question-answer pairs from wikipedia. *meeting of the association for computational linguistics*, 1:1907–1917.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. *meeting of the association for computational linguistics*, 1:1342–1352.

Nan Duan, Duyu Tang, Peng Chen, and Ming Zhou. 2017. Question generation for question answering. pages 866–874.

Jiatao Gu, Zhengdong Lu, Li Hang, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning.

Michael Heilman and Noah A Smith. 2010. Good question! statistical ranking for question generation. pages 609–617.

Yanghoon Kim, Hwanhee Lee, Joongbo Shin, and Kyomin Jung. 2019. Improving neural question generation using answer separation. *national conference on artificial intelligence*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *international conference on learning representations*.

Vishwajeet Kumar, Ganesh Ramakrishnan, and Yuanfang Li. 2018. A framework for automatic question generation from text using deep reinforcement learning. *arXiv: Computation and Language*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. *north american chapter of the association for computational linguistics*, pages 110–119.

Xiaodong Liu, Yelong Shen, Kevin Duh, and Jianfeng Gao. 2018. Stochastic answer networks for machine reading comprehension. *meeting of the association for computational linguistics*, 1:1694–1704.

Ruslan Mitkov and Le An Ha. 2003. Computer-aided generation of multiple-choice tests. pages 17–22.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. *Proc Meeting of the Association for Computational Linguistics*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. pages 1532–1543.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *empirical methods in natural language processing*, pages 2383–2392.

Mrinmaya Sachan and Eric P Xing. 2018. Self-training for jointly learning to ask and answer questions. 1:629–640.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *meeting of the association for computational linguistics*, 1:1073–1083.

Linfeng Song, Zhiguo Wang, Wael Hamza, Yue Zhang, and Daniel Gildea. 2018. Leveraging context information for natural question generation. 2:569–574.

Duyu Tang, Nan Duan, Tao Qin, and Ming Zhou. 2017. Question answering and question generation as dual tasks. *arXiv: Computation and Language*.

Tong Wang, Xingdi Yuan, and Adam Trischler. 2017. A joint model for question answering and question generation. *arXiv: Computation and Language*.

Xingdi Yuan, Tong Wang, Caglar Gulcehre, Alessandro Sordoni, Philip Bachman, Saizheng Zhang, Sandeep Subramanian, and Adam Trischler. 2017. Machine comprehension by text-to-text neural question generation. *meeting of the association for computational linguistics*, pages 15–25.

Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. 2017. Neural question generation from text: A preliminary study. *arXiv: Computation and Language*, pages 662–671.

# Improving Neural Story Generation by Targeted Common Sense Grounding

**Huanru Henry Mao, Bodhisattwa Prasad Majumder,**
**Julian McAuley, Garrison W. Cottrell**
Department of Computer Science and Engineering
UC San Diego
{hhmao, bmajumde, jmcauley, gary}@eng.ucsd.edu

## Abstract

Stories generated with neural language models have shown promise in grammatical and stylistic consistency. However, the generated stories are still lacking in *common sense reasoning*, e.g., they often contain sentences deprived of world knowledge. We propose a simple multi-task learning scheme to achieve quantitatively better common sense reasoning in language models by leveraging auxiliary training signals from datasets designed to provide common sense grounding. When combined with our two-stage fine-tuning pipeline, our method achieves improved common sense reasoning and state-of-the-art perplexity on the *WritingPrompts* (Fan et al., 2018) story generation dataset.

## 1 Introduction

*Story generation* is the task of automatically producing compelling creative writing. Recent advances in language modeling have yielded thematic and stylistic coherence in story generation through large scale pretraining of Transformer models (Vaswani et al., 2017). The recent introduction of the General Pre-trained Transformer v2 (GPT2) (Radford et al., 2019)—a high-capacity Transformer trained on a large, diverse corpus of text crawled from the web (called WebText)—is capable of generating stylistically coherent text but commonly produces text with logical inconsistencies. For example, in one sample the model writes: "*It was a sunny, warm summer night*". Obviously, this writing is nonsense as it cannot be sunny at night. The lack of common sense reasoning in such a strong language model suggests that minimizing next-token perplexity alone may be insufficient in producing models that can compose sensible stories.

In this paper, we consider the challenge of *common sense reasoning* (CSR) in language modeling for story generation. Unlike other work in the CSR literature (Storks et al., 2019), which evaluates CSR in isolation, we are specifically interested in a generative model's likelihood of producing text that exhibits common sense. Evaluating common sense qualitatively in a model's samples is difficult, as it is subject to human bias and dependent on sampling procedure. We propose evaluating the common sense of a model automatically by ranking the model's perplexity on spurious (plausible, but nonsense) text completions from SWAG (Zellers et al., 2018) and Story Cloze (Mostafazadeh et al., 2016) datasets, which are designed for common sense grounding.

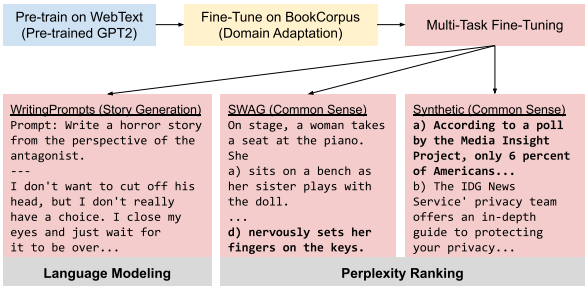


Figure 1: Our two-stage fine-tuning pipeline with auxiliary multi-task learning. For perplexity ranking examples, bolded text indicates the correct answer. For **synthetic**, the correct choice is written by a human, and the wrong choice is generated by a neural network.

**Our contributions** are as follows: We propose a simple way to define *better* CSR in generative models, which leads to an auxiliary multi-task objective to directly bias our model to generate text with better common sense. When fine-tuning is combined with multi-task learning in a two-stage pipeline, we improve the model's CSR and outperform state-of-the-art perplexity on the WritingPrompts (Fan et al., 2018) dataset.[1]

[1]Our source code is at https://github.com/calclavia/story-generation.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5988–5993,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

## 2 Tasks

### 2.1 Primary Task: Language Modeling

Our primary task is to perform language modeling (Elman, 1990; Bengio et al., 2003; Dai and Le, 2015) on the WritingPrompts dataset. A language model learns to assign the probability of a text sequence $X = x_1, \ldots, x_T$ using the conditional probability factorization:

$$P(X) = \prod_{t=1}^{T} P(x_t | x_{1:t-1}). \quad (1)$$

We train our model using a standard cross-entropy loss between next-step true tokens and predicted probabilities given current tokens.

WritingPrompts (Fan et al., 2018) is a dataset of prompts and short stories crawled from Reddit. The aim of the dataset is to produce a story given a free-text prompt. We reduce this conditional text generation task into a generic language modeling task by simply concatenating the prompt before the story and treating a prompt-story pair as one input to the Transformer decoder model. This human-readable format (example in Figure 1) is chosen because GPT2 may have been trained on similarly formatted text from the web. When sampling, we can either seed the model with a prompt or allow it to generate its own prompt.

### 2.2 Auxiliary Task: Perplexity Ranking

Our auxiliary task aims to bias the model to produce text with better common sense (which we refer to as *sensible* text). Given a set of text sequences $S = \{S_1, \ldots, S_N\}$, where $S_1$ is a sensible text sequence and the rest $S_2, \ldots, S_N$ are spurious text sequences, we operationally define *better* as the model assigning higher probability (Eq. 1) to $S_1$ versus the average probability over spurious text sequences $S_2, \ldots, S_N$. Using this definition, it is possible to directly optimize the model to assign $P(S_1)$ to be higher than any $P(S_i)$. Formally, we define the probability of the model choosing the correct sequence $S_1$ over spurious sequences as the softmax over the length-normalized log probabilities of all plausible sequences:

$$\frac{\exp(\frac{1}{T_1} \log P(S_1))}{\sum_{i=1}^{N} \exp(\frac{1}{T_i} \log P(S_i))} \quad (2)$$

where $T_i$ refers to the length of the text sequence $S_i$. Thus, we can practically minimize the negative log-likelihood of Eq. 2 by reusing the same softmax layer used for the primary language modeling task. We refer to this objective as *perplexity ranking* as it constrains the model to rank sensible text to have lower perplexity than spurious ones.

| Dataset | Size | Role |
|---|---|---|
| WritingPrompts | 272K Stories | Story Generation |
| BookCorpus | 10K Books | Domain Adaptation |
| SWAG | 73K Questions | Common Sense |
| Synthetic | 250K Pairs | Common Sense |

Table 1: Datasets, training set sizes and their roles

**SWAG:** In order to train on this auxiliary objective, we need training examples in the format of multiple choice questions, where the correct choice corresponds to the text with the best common sense. We choose the SWAG dataset (Zellers et al., 2018), a dataset that provides 4-way multiple choice common sense questions that are adversarially filtered to seem plausible to language models. Unlike other CSR datasets (Talmor et al., 2018), SWAG forms its question and answer as two full sentences, which we can concatenate into a single string to find its probability. This makes it suitable for perplexity ranking.

**Synthetic:** To facilitate perplexity ranking on SWAG, we additionally use a synthetic dataset that consists of 250K pairs of human written text from WebText and samples generated by the original 1.5B parameter version of the GPT2 model.[2] These samples are many paragraphs long and truncated to a maximum of 1024 tokens. We frame these pairs as a 2-way classification problem and train the model by perplexity ranking to assign higher likelihood to human written text over synthetic examples. The assumption we make is that human written text is more sensible than text written by neural language models. Hence, on average, this promotes the model to assign higher probabilities to sensible text.

## 3 Training Pipeline

We introduce a two-stage training pipeline (Figure 1) to improve model performance both in terms of perplexity and CSR on story generation. Our pipeline uses four different datasets (Table 1), each of which plays a role in improving model performance. Our model architecture is the 117M parameter version of GPT2, using the pre-trained

[2]We obtained the samples from https://github.com/openai/gpt-2-output-dataset

weights provided by Radford et al. (2019). We refer readers to Vaswani et al. (2017) for details of the Transformer architecture.

We first perform intermediate fine-tuning (Phang et al., 2018; Howard and Ruder, 2018) of the pre-trained GPT2 on BookCorpus (Zhu et al., 2015) as a method of domain adaptation from WebText to the domain of stories. BookCorpus is a dataset that contains over 10,000 free books crawled from the web.[3] We train on this corpus using our language modeling objective. Next, we fine-tune on the target WritingPrompts dataset with a multi-task learning objective. We alternate training between the language modeling objective on WritingPrompts and perplexity ranking on SWAG and our synthetic dataset. Training details and hyperparameters are in the appendix.

## 4 Evaluation

We perform three types of evaluation on the model to assess its readability, reliance on the prompt (prompt ranking) and CSR.

Readability is measured in terms of model perplexity on the test set of WritingPrompts. Because GPT2 uses subword tokenization (Sennrich et al., 2016), it is not directly comparable to the word-level perplexity obtained in Fan et al. (2018). We estimate the corresponding word-level perplexity by taking the product of each subword's probabilities to obtain probabilities for each word. Both sub-word perplexity and word-level perplexities are reported in our experiments.

Prompt ranking (Fan et al., 2018) assesses how well a model matches a story to its given prompt. This is measured by computing the likelihood of stories conditioned under ten different prompts, nine of which are randomly sampled and one is the true prompt. Following Fan et al. (2018), we count a random story sample as correct when it ranks the true prompt with the lowest perplexity. We compute the accuracy from 1000 random samples.

CSR is evaluated on two multiple choice datasets – SWAG and Story Cloze (Mostafazadeh et al., 2016). We rank the perplexity computed by the model for each example and count it as correct if the lowest perplexity matches the answer. The SWAG validation set provides a proxy of how well the model generalizes in CSR to unseen examples. To ensure generalization beyond SWAG, we also perform *zero-shot* evaluation on the Winter 2018 Story Cloze validation set. Story Cloze consists of 5-sentence stories with correct and spurious endings. It is similar to SWAG but serves as an in-domain dataset to specifically test the model's performance at CSR in story telling.

[3]We crawled BookCorpus from https://www.smashwords.com/

| |
|---|
| **Premise:** *John and Billy became very skilled at beer pong. They entered a contest in college. They won the contest and advanced to the next level. The next level sent them to Vegas.* |
| **GPT2 → BC → WP output:** |
| 1. They would fall. |
| 2. Later, they figured out what it was all about. |
| **GPT2 → BC → WP + SWAG + SYNTH output:** |
| 1. They have been ranked number one in their respective leagues and are considered the best in their respective countries. |
| 2. They then moved to the bars. |

Table 2: Top two highest likelihood story completions from 10 random completion samples generated by our models when primed with a premise from the Story Cloze validation set.

## 5 Results and Discussion

We analyze our pipeline and report the results in Table 3. We also generate stories by sampling from our model using nucleus sampling with $p = 0.9$ (Holtzman et al., 2019). We present example story completions in Table 2 and full sampled stories in our appendix and Table 4.

**Pre-training:** We compare our models with the attention-based Fusion Model (Fan et al., 2018), which has been designed for and trained on WritingPrompts. We observe that a pre-trained GPT2 performing zero-shot inference on WritingPrompts (GPT2 in Table 3) is a strong baseline. By fine-tuning GPT2 on WritingPrompts (GPT2 → WP), we outperform the Fusion Model in perplexity. All our models outperform the Fusion Model in prompt ranking, which suggests that task-specific models are unnecessary given pre-training.

**Intermediate Fine-Tuning:** The first stage in our pipeline performs intermediate fine-tuning of GPT2 on BookCorpus (GPT2 → BC in Table 3). To confirm that intermediate fine-tuning helps downstream performance, we evaluate the zero-shot performance of the model on WritingPrompts. This yields perplexity and prompt ranking improvements compared to GPT2, demonstrating successful domain adaptation. Perform-

| Models | SW PPL | Word PPL | Prompt Ranking | SWAG | Story Cloze |
|---|---|---|---|---|---|
| Fusion Model (Fan et al., 2018) | - | 36.6 | 16.3% | - | - |
| GPT2 | 35.57 | 51.29* | 49.8% | 48.1% | 58.8% |
| GPT2 → BC | 29.10 | 42.01* | 62.7% | 50.5% | 59.6% |
| GPT2 → WP | 21.68 | 30.65* | 80.0% | 49.8% | 58.3% |
| GPT2 → BC → WP | 20.79 | 29.56* | **80.6%** | 51.4% | 59.1% |
| GPT2 → BC → WP + SWAG | **20.78** | **29.52*** | 78.9% | 75.3% | 63.2% |
| GPT2 → BC → WP + SWAG + SYNTH | **20.78** | 29.63* | 80.1% | **76.3%** | **64.1%** |

Table 3: Performance of models on the test set of WritingPrompts and validation set of SWAG and Story Cloze. **SW PPL** and **Word PPL** refer to sub-word and word-level perplexity on WritingPrompts respectively. WP refers to WritingPrompts, BC refers to BookCorpus and SYNTH refers to training with our 250K synthetic examples. The asterisk * refers to an estimated score derived from BPE PPL.

ing two-stage fine tuning (GPT2 → BC → WP) further improves perplexity and CSR. We hypothesize the improvement in CSR is due to BookCorpus being a higher quality dataset written by authors when compared against WebText.

**Multi-tasking Fine-Tuning:** Performing multi-task learning on WritingPrompts and SWAG (GPT2 → BC → WP + SWAG) unsurprisingly yields significant improvements on the SWAG validation set. More importantly, the zero-shot performance on Story Cloze also improved, indicating that it was able to generalize its common sense knowledge. We also see qualitatively improved results when generating story completions (Table 2). The addition of the synthetic dataset we introduced (GPT2 → BC → WP + SWAG + SYNTH) further boosts performance on CSR. Other metrics are negligibly affected by the auxiliary tasks.

## 6 Related Work

**Story Generation:** Recent work in neural story generation (Kiros et al., 2015; Roemmele, 2016) has shown success in using hierarchical methods (Yao et al., 2018; Fan et al., 2018) to generate stories. In these schemes, a neural architecture is engineered to first generate an outline or a prompt, then to expand the prompt into a full-length story. Our work performs hierarchical generation, but our main focus is on achieving better common sense in the generated text rather than engineering task-specific architectures.

**Common Sense Reasoning:** Common sense reasoning (CSR) has been studied through many benchmarks such as SWAG (Zellers et al., 2018), Story Cloze (Mostafazadeh et al., 2016), the Winograd Schema Challenge (Levesque et al., 2012), and CommonsenseQA (Talmor et al., 2018). Recent methods (Peters et al., 2018; Radford et al., 2018) on these benchmarks focus on large-scale pre-training of language models. They show that transfer learning is an effective means to improve CSR and our fine-tuning pipeline builds upon these techniques. Our results on SWAG and Story Cloze are far from state-of-the-art (Devlin et al., 2018). However, our aim is not to directly tackle SWAG or Story Cloze, but instead to use it as a constraint on our model and a proxy to estimate the likelihood of generating sensible text.

**Multi-task Learning:** Multi-task learning (MTL) introduces inductive bias in a model, helps reduce overfitting and increases robustness (Caruana, 1993; Ruder, 2017; McCann et al., 2018). Our work builds upon MTL principles as we introduce auxiliary tasks to specifically tackle CSR (Søgaard and Bingel, 2017). Contrary to conventional auxiliary multi-task learning (Howard and Ruder, 2018), which adds an additional classifier to the final representations of the model for learning auxiliary tasks, we use perplexity ranking, which does not introduce additional parameters. We argue that perplexity ranking, by definition, guarantees that the model will generate sensible text with a higher probability.

## 7 Conclusion

We demonstrated a simple way to improve the likelihood that a particular language model generates sensible text. We introduced a training pipeline that leverages transfer and multi-task learning to improve perplexity and CSR. One limitation of our approach is its dependence on SWAG, which only provides short-term common sense grounding. While our synthetic dataset may mitigate this issue, story-level coherence is unlikely to be achieved using this method alone. We leave this investigation for future work.

| **Prompt:** Aliens start abducting humans. |
|---|
| Steve smashed his Xbox controller at the table. He lit a smoke and took another hit. Two exes were in the room, just outside the door.<br>"What are you doing?" he said. "Get back in there."<br>"I-I've never actually seen any of the aliens," he replied, dispelling the smoke from his mouth.<br>The two aliens were smirking. "Is this your first time?" the ex asked.<br>"The last one tried to kill us."<br>The aliens looked at each other. "Are you sure you are ok? It looks like you are severely hurt."<br>"Yeah, I'm fine. I just saw the door open. I'm just fine," he replied.<br>"We're fine? You do that every day, Steve, but we still haven't gotten past this door," the ex said, resting a hand on the UFO that stood in the doorway.<br>"I'm fine."<br>"Good! I've never seen anything like it, have I Steve?"<br>Steve stood up. "Yes I have." He picked up his controller.<br>"Ok, shoot!" the aliens yelled. "This is no time to hesitate."<br>Steve put the controller to his ear. "Who are you? What are you doing here?"<br>The aliens pulled the gun towards him. "We-we're here to kidnap you."<br>Steve put his hands on the UFO, and pulled the trigger. The gun went off. The aliens, scared, immediately realized where they were and ran towards the other two aliens. They kept running. Steve continued on his way towards his childhood toy house. When he was just a teen, he had built his own helicopter. He always felt they had lost. The abandoned building on the edge of the highway he and the ex had entered had been the perfect landing spot. He didn't know where they were. He didn't know how they were going to get out of there. There were bodies everywhere. |

Table 4: Sample generated by GPT2 → BC → WP + SWAG + SYNTH primed with the same prompt as Fan et al. (2018).

**Acknowledgements.** This work is partly supported by NSF #1750063, the Guangzhou Science and Technology Planning Project #201704030051, NSF awards CNS-1730158, ACI-1540112, ACI-1541349, OAC-1826967, the University of California Office of the President, and the University of California San Diego's California Institute for Telecommunications and Information Technology/Qualcomm Institute. We thank CENIC for the 100Gpbs networks and all the reviewers for their constructive suggestions.

## References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Rich Caruana. 1993. Multitask learning: A knowledge-based source of inductive bias. In *ICML*.

Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised sequence learning. In *NIPS*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14(2):179–212.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *ACL*.

Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. *CoRR*, abs/1904.09751.

Jeremy Howard and Sebastian Ruder. 2018. Fine-tuned language models for text classification. *CoRR*, abs/1801.06146.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *NIPS*.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language de-

cathlon: Multitask learning as question answering. *CoRR arXiv:1806.08730*.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James F. Allen. 2016. A corpus and evaluation framework for deeper understanding of commonsense stories. *NAACL-HLT*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*.

Jason Phang, Thibault Févry, and Samuel R. Bowman. 2018. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *CoRR*, abs/1811.01088.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. 2019. Language models are unsupervised multitask learners. Technical report, OpenAI.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. Technical report, OpenAI.

Melissa Roemmele. 2016. Writing stories with help from recurrent neural networks. In *AAAI*.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *CoRR*, abs/1706.05098.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL*.

Anders Søgaard and Joachim Bingel. 2017. Identifying beneficial task relations for multi-task learning in deep neural networks. In *EACL*.

Shane Storks, Qiaozi Gao, and Joyce Y. Chai. 2019. Commonsense reasoning for natural language understanding: A survey of benchmarks, resources, and approaches. *CoRR*, abs/1904.01172.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2018. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *CoRR*, abs/1811.00937.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Lili Yao, Nanyun Peng, Ralph M. Weischedel, Kevin Knight, Dongyan Zhao, and Rui Yan. 2018. Plan-and-write: Towards better automatic storytelling. *CoRR*, abs/1811.05701.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. SWAG: A large-scale adversarial dataset for grounded commonsense inference. *CoRR*, abs/1808.05326.

Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *ICCV*.

# Abstract Text Summarization: A Low Resource Challenge

**Shantipriya Parida** **Petr Motlicek**
Idiap Research Institute
Rue Marconi 19, 1920 Martigny, Switzerland
{shantipriya.parida,petr.motlicek}@idiap.ch

## Abstract

Text summarization is considered as a challenging task in the NLP community. The availability of datasets for the task of multilingual text summarization is rare, and such datasets are difficult to construct. In this work, we build an abstract text summarizer for the German language text using the state-of-the-art "Transformer" model. We propose an iterative data augmentation approach which uses synthetic data along with the real summarization data for the German language. To generate synthetic data, the Common Crawl (German) dataset is exploited, which covers different domains. The synthetic data is effective for the low resource condition and is particularly helpful for our multilingual scenario where availability of summarizing data is still a challenging issue. The data are also useful in deep learning scenarios where the neural models require a large amount of training data for utilization of its capacity. The obtained summarization performance is measured in terms of ROUGE and BLEU score. We achieve an absolute improvement of +1.5 and +16.0 in ROUGE1 F1 (R1_F1) on the development and test sets, respectively, compared to the system which does not rely on data augmentation.

## 1 Introduction

Automatic text summarization is considered as a challenging task because while summarizing a piece of text, we read it entirely to develop our understanding to prepare highlighting its main points. Due to the lack of human knowledge and language processing abilities in computers, automatic text summarization is a major non-trivial task (Allahyari et al., 2017).

Two major approaches for automatic summarization are: extractive and abstractive. The extractive summarization approach produces summaries by choosing a subset of sentences in the original text. The abstract text summarization approach aims to shorten the long text into a human-readable form that contains the most important fact from the original text (Allahyari et al., 2017; Kryściński et al., 2018).

The deep learning-based neural attention model when applying to abstract text summarization performs well compared to standard learning-based approaches (Rush et al., 2015). Abstract text summarization using the attentional encoder-decoder recurrent neural network approach shows a state-of-the-art performance and sets a baseline model (Nallapati et al., 2016). Further improvements are introduced to the baseline model by using the pointer generator network and coverage mechanism using reinforcement learning based training procedure (See et al., 2017; Paulus et al., 2017). There is an inherent limitation to natural language processing tasks such as text summarization for resource-poor and morphological complex languages owing to a shortage of quality linguistic data available (Kurniawan and Louvan, 2018). The use of synthetic data along with the real data is one of the popular approaches followed in machine translation domain for the low resource conditions to improve the translation quality (Bojar and Tamchyna, 2011; Hoang et al., 2018; Chinea-Rıos et al., 2017). The iterative back-translation (e.g. training back-translation systems multiple times) were also found effective in machine translation (Hoang et al., 2018). We explore similar approaches in our experiments for the text summarization task.

The organizations of this paper is as follows: Section 1 describes related work on abstract text summarization. Section 2 explains the techniques followed in our work. Section 3 describes the dataset used in our experiment. Section 4 explains the experimental settings: models and their parameters. Section 5 provides evaluation results with



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5994–5998,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

analysis and discussion. Section 6 provides conclusion to the paper.

## 2 Method Description

Across all experiments performed in this paper, we have used the Transformer model as implemented in OpenNMT-py[1] (Vaswani et al., 2018; See et al., 2017). The Transformer model is based on encoder/decoder architecture. In context to summarize, it takes text as input and provides its summary.

We use synthetic data as shown in Figure 1 to increase the size of the training data.

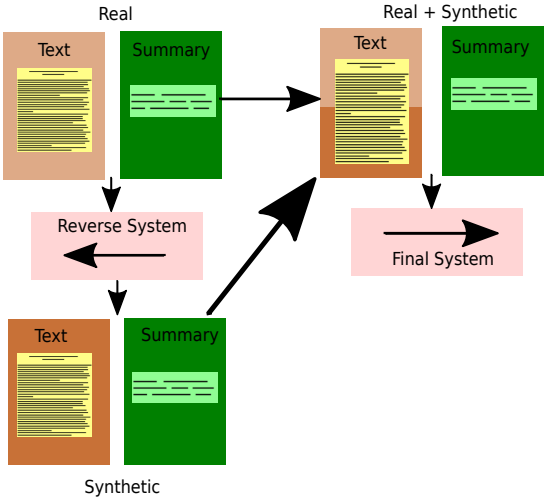


Figure 1: Generation of synthetic data using a reverse system. To generate synthetic data, first, a system in the reverse direction (i.e. source as summary and target as text) is trained and then used to generate text for the given summary. Then both the real and synthetic data acts as input to the final system.

## 3 Dataset Description

We use German wiki data (spread across different domain) collected from the SwissText 2019[2] (real data) and Common Crawl[3] data (synthetic data) in our experiment. The statistics of all the datasets are shown in Table 1.

### 3.1 SwissText datasets used as real data

We divide the 100K SwissText dataset (downloaded from SwissText 2019 website) into three subsets: train, dev, and test in 90:5:5 ratio (i.e. 90K for training, 5K for development and 5K for the test data). The experiments performed over these datasets are described in Section 4.3 (denoted as S1 experimental setup).

| Dataset | #Text | #Summ |
|---|---|---|
| Train_Real(SwissText) | 90K | 90K |
| Train_RealSynth(Swiss+CC) | 190K | 190K |
| Train_RealSynthRegen(Swiss+CC) | 190K | 190K |
| Dev(SwissText) | 5K | 5K |
| Test(SwissText) | 5K | 5K |

Table 1: Statistics of the experimental data which include the number of texts and their summaries.

### 3.2 Common Crawl dataset used as synthetic data

The data crawled from the Internet (Common Crawl) used to prepare synthetic data to boost the training. The steps followed to create the synthetic dataset as follows:

Step 1: **Build vocab:** We create vocabulary using SwissText based on the occurrence of the most frequent (top N) German words.

Step 2: **Sentence selection:** The sentences from the Common Crawl data are selected with respect to the vocabulary based on the threshold we provide (e.g. a sentence has 10 words and the threshold is 10% (0.1)). For a sentence to be selected, at least 1 out of 10 words should be in the vocabulary.

Step 3: **Filtering:** Select random sentences (e.g. 100K) from the selected Common Crawl data in the previous step.

Step 4: **Generate summary:** The 100K data obtained from the previous step are used as a summary and required to generate corresponding text. We use the reverse trained model where we provide the summary as source and target as text. This results in the text as well as the corresponding summary as additional data to be utilized along with real data (SwissText).

Eventually, the 190K dataset is created (denote as Train_RealSynth) as a combination of 90K SwissText train data (real) and 100K synthetic data. This dataset is used in the experimental setup S2 (described in details in Section 4.3).

## 4 Experimental Setup

This section describes our experiments conducted for the text summarization task.

[1] http://opennmt.net/OpenNMT-py/Summarization.html
[2] https://www.swisstext.org/
[3] http://commoncrawl.org/

| Setting | Dataset | R1_F1 | R2_F1 | RL_F1 | BLEU |
|---|---|---|---|---|---|
| S1 | Dev | 43.9 | 28.5 | 46.3 | 12.6 |
| | Test | 39.7 | 22.9 | 42.2 | 9.0 |
| S2 | Dev | 45.4 | 29.8 | 47.4 | 14.0 |
| | Test | 55.7 | 41.8 | 57.6 | 20.8 |
| S3 | Dev | 44.3 | 28.5 | 46.4 | 13.1 |
| | Test | 40.0 | 23.0 | 42.3 | 9.4 |

Table 2: Evaluation results of our models on development (dev) and testing (test) sets. The automatic evaluation scores in terms of Rouge (R1_F1, R2_F1, RL_F1) and BLEU for the output summaries are shown in the table.

### 4.1 Prepossessing

The preprocess step involves preprocessing the dataset such that source and target are aligned and use the same dictionary. Additionally, we truncate the source length at 400 tokens and the target length at 100 tokens to expedite training (See et al., 2017).

### 4.2 Model Parameters

The Transformer model is implemented in OpenNMT-py. To train the model, we use a single GPU. To fit the model to the GPU cluster, a batch size equal to 4,096 is selected for training. The validation batch size is set to 8. We use an initial learning rate of 2, drop out of 0.2 and 8,000 warm-up steps. Decoding uses a beam size of 10 and we did not set any minimum length of output summary.

### 4.3 Model Setup

We use 3 settings: (i) real data (we set this as the baseline in our experiment), (ii) real data and synthetic data, and (iii) real and regenerated synthetic data for the summarization task, described as follows:

1. *S1: Transformer model using Train_Real data*

   In this setup, we use the "Train_Real" data for training the Transformer model.

2. *S2: Transformer Model using Train_RealSynth data*

   In this setup, we use the "Train_RealSynth" data for training the Transformer model. As the balance between real and synthetic data is an important factor, we maintain a 1:1 ratio (e.g. 1 (real) :1 (synthetic)) for our experiment (Sennrich et al., 2016).

3. *S3: Transformer Model using Train_RealSynthRegen data*

   We propose an iterative approach to improve the quality of synthetic summaries. In this setup, after training a system with (real+synthetic) data, it is used to regenerate synthetic data for the final system. As a result, the input data to the final system is a combination of real and regenerated synthetic data as shown in Figure 2.

### 4.4 Training Procedure

The copying mechanism is applied during training. It allows the summarizer to fall back and copy the source text when encounters $< unk >$ tokens by referencing to the softmax of the multiplication between attention scores of the output with the attention scores of the source (See et al., 2017). The systems are trained for 300K iterations.

## 5 Evaluation and Discussion

We evaluate the results for every 10,000 iterations on the dev and test set. The automatic evaluation results based on the dev and test set are shown in Table 2 with sample summaries in Table 3. To evaluate the proposed algorithms, we use ROUGE (Recall-Oriented Understudy for Gisting Evaluation) score, which is a popular metric for text summarization task, and has several variants like ROUGE-N, and ROUGE-L, which measure the overlap of n-grams between the system and reference summary (LIN, 2004). We use ROUGE_1 F1 (R1_F1), ROUGE_2 F1 (R2_F1), and ROUGE_L F1 (RL_F1) for scoring the generated summary. In addition, we also use the SacreBLEU[4] evaluation metric (Post, 2018).

Figure 3 presents the learning curves for the models (S1 and S2) on the development set. It can be seen that there is a variance (e.g. word

[4] https://github.com/mjpost/sacreBLEU

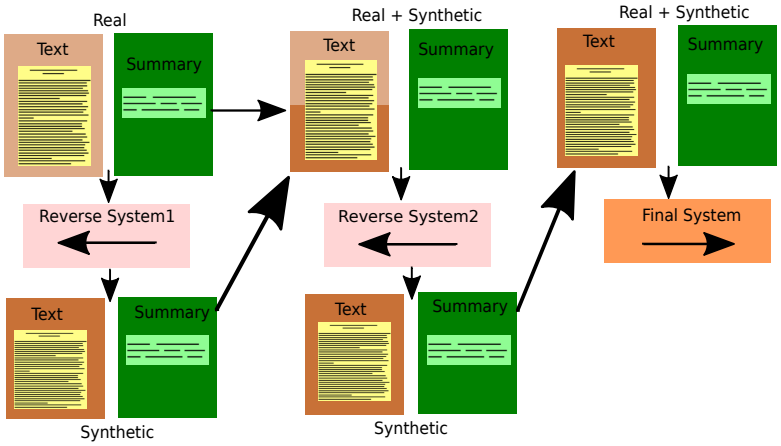


Figure 2: Regeneration of synthetic data. After training a system with real+synthetic data (Reverse System2 above), used to create synthetic summarization data for the final system.

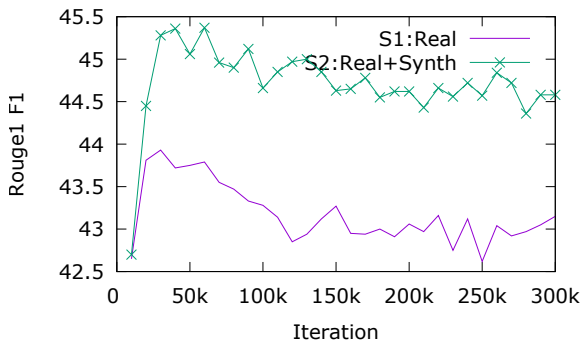


Figure 3: Learning curves in terms of Rouge1 F1 (R1_F1) Score on dev set.

| |
|---|
| *Ref Summary* : "Das Feuerschiff Relandersgrund war ein finnisches Feuerschiff, das von 1888 bis 1914 im Schrenmeer bei Rauma positioniert war. Heute dient es als Restaurantschiff in Helsinki." *Gloss*: The lightship Relandersgrund was a Finnish lightship, which was built from 1888 to 1914 Schrenmeer was positioned at Rauma. Today serves it as a restaurant ship in Helsin |
| *S1 Summary*: :"Die "Rauma". ist ein 1886—1888 Feuerschiff der norwegischen Reederei "Libauskij", Das Schiff wurde in den 1930er Jahren gebaut und in den 2000er Jahren als Museumsschiff als" *Gloss*:"The "Rauma ". is a 1886-1888 Lightship of the Norwegian shipping company "Libauskij",The ship was built in the 1930s and in the 2000s as a museum ship as |
| *S2 Summary*: :"Das Feuerschiff Relandersgrund war ein Feuerschiff des das von 1888 bis 1914 im Einsatz war. Heute dient es als Restaurantschiff in Kotka," *Gloss*: The lightship Relandersgrund was on Lightship of the 1888 to 1914 was in use. Today it serves as a restaurant ship in Kotka |
| *S3 Summary*: :"Das Kotka." ist ein finnischer Museumsschiff der im Zweiten Weltkrieg von der russischen Marine als Restaurantschiff 1" eingesetzt wurde. Im Mittelalter war das Schiff unter dem Namen "Vuolle" 1" fr die finnische Marine 1" *Gloss*: The Kotka. "Is a Finnish one Museum ship of the World War II Russian Navy used as a restaurant ship 1 " has been. In the Middle Ages, the ship was under the name "Vuolle" 1 "for the Finnish Navy 1 |

Table 3: Sample summaries on test set. The matching words of generating summaries with respect to references are shown in color blue.

selection, summary length) for model S2 generated summary as compared with model S1. During manual verification, we found that the summaries generated without a minimum length constraint appear better compared to summaries with minimum length constraint. Although we do not explicitly specify a minimum length parameter for generating summaries for the models, the average length of words generated by model S2 (e.g. 41.42 words) is longer than the model S1 (e.g. 39.81 words). Some data (e.g. name, year) were found inconsistent during a comparison of the generated summary with respect to the reference. There is a variance in summaries generated by model S3 as compared to S2 and S1. In terms of Rouge score model S3 outperforms model S1 but perform worse than model S2 (see Table 2).

## 6 Conclusion

In this paper, we highlighted the implementation of synthetic data for the abstract text summarization task under low resource condition, which helps improving the text summarization system in terms of automatic evaluation metrics. As the next step, we plan to investigate: i) synthetic summarization data, and ii) applying transfer learning on text summarization for the multilingual low resource data set with little or no ground truth summaries (Keneshloo et al., 2018).

## Acknowledgments

The work is supported by an innovation project (under an InnoSuisse grant) oriented to improve the automatic speech recognition and natural language understanding technologies for German. Title: "SM2: Extracting Semantic Meaning from Spoken Material" funding application no. 29814.1 IP-ICT.

## References

Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saeid Safaei, Elizabeth D Trippe, Juan B Gutierrez, and Krys Kochut. 2017. Text summarization techniques: a brief survey. *arXiv preprint arXiv:1707.02268*.

Ondrej Bojar and Aleš Tamchyna. 2011. Improving translation model by monolingual data. In *Sixth Workshop on Statistical Machine Translation*, page 330.

Mara Chinea-Rıos, Alvaro Peris, and Francisco Casacuberta. 2017. Adapting neural machine translation with parallel synthetic data. *WMT 2017*, page 138.

Cong Duy Vu Hoang, Philipp Koehn, Gholamreza Haffari, and Trevor Cohn. 2018. Iterative back-translation for neural machine translation. *ACL 2018*, 23(32.5):18.

Yaser Keneshloo, Naren Ramakrishnan, and Chandan K Reddy. 2018. Deep transfer reinforcement learning for text summarization. *arXiv preprint arXiv:1810.06667*.

Wojciech Kryściński, Romain Paulus, Caiming Xiong, and Richard Socher. 2018. Improving abstraction in text summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1808–1817.

Kemal Kurniawan and Samuel Louvan. 2018. Indosum: A new benchmark dataset for indonesian text summarization. In *2018 International Conference on Asian Language Processing (IALP)*, pages 215–220. IEEE.

C-Y LIN. 2004. Rouge: A package for automatic evaluation of summaries. In *Proc. of Workshop on Text Summarization Branches Out, Post Conference Workshop of ACL 2004*.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191. Association for Computational Linguistics.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 86–96.

Ashish Vaswani, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan Gomez, Stephan Gouws, Llion Jones, Łukasz Kaiser, Nal Kalchbrenner, Niki Parmar, Ryan Sepassi, Noam Shazeer, and Jakob Uszkoreit. 2018. Tensor2tensor for neural machine translation. In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Papers)*, pages 193–199. Association for Machine Translation in the Americas.

# Generating Modern Poetry Automatically in Finnish

**Mika Hämäläinen**
Department of Digital Humanities
University of Helsinki
mika.hamalainen@helsinki.fi

**Khalid Alnajjar**
Department of Computer Science
University of Helsinki
khalid.alnajjar@helsinki.fi

## Abstract

We present a novel approach for generating poetry automatically for the morphologically rich Finnish language by using a genetic algorithm. The approach improves the state of the art of the previous Finnish poem generators by introducing a higher degree of freedom in terms of structural creativity. Our approach is evaluated and described within the paradigm of computational creativity, where the fitness functions of the genetic algorithm are assimilated with the notion of aesthetics. The output is considered to be a poem 81.5% of the time by human evaluators.

## 1 Introduction

Poem generation is a challenging task for creative NLG (natural language generation) requiring structural integrity in the form of rhyming and meter, grammatical correctness and figurative expression. Poems are meant to be interpreted and therefore the meaning they convey cannot be fully explained by semantics, but they rather require an exploration into the notion of pragmatics.

In this paper, we present a novel approach based on a genetic algorithm for creating poetry in Finnish from the stand point of computational creativity. In addition to solving problems related to poems in general, the morphosyntactically complex Finnish sets additional requirements for producing grammatical output.

Computational creativity can be seen as a search for creative artefacts in a conceptual space (cf. Wiggins, 2006). Therefore the use of genetic algorithm for a creative task is reasonable as it conducts a search and picks out the most suitable candidates based on its fitness function. An important aspect for creativity is that the system should be able to assess its own creations, a notion called appreciation (Colton, 2008) or aesthetic function (Colton et al., 2011) in the literature. The fitness function of the genetic algorithm serves for this exact purpose, as it can score the output in terms of different aesthetic dimensions.

## 2 Related work

In the past, poetry generation has been studied both from the point of view of computational creativity and natural language generation. Poem generation has been tackled with a variety of different methods such as case-based reasoning (Gervás, 2001), templates (Colton et al., 2012), translation with WFSTs (weighted finite-state transducers) (Greene et al., 2010), text transformation via word embeddings (Bay et al., 2017) and conditional variational autoencoders (Li et al., 2018). As the field of poem generation has been broadly discussed by Oliveira (2017), we dedicate the rest of this section to describing the existing poetry generation work conducted for Finnish within the computational creativity paradigm. We also discuss some previous approaches using genetic algorithms.

One of the first takes on Finnish poem generation is the P. O. Eticus system (Toivanen et al., 2012). P. O. Eticus uses a corpus of human authored poems. These poems are used as templates for generating new poetry. In practice, the system takes a random poem from the corpus, conducts a morphological analysis on it and replaces some of the words in the existing poem. The replaced words are inflected to match the morphology of the original word.

Another take on the poetry generation in Finnish is that of Kantosalo et al. (2015). This approach is presented as a part of a poem authoring system. How this generator operates is that it takes sentences form children's books stored in its corpus based on a shared keyword. These sentences serve as verses, or poem fragments, and



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5999–6004,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

they are output one after another forming a generated poem. As the system does not alter text at all, it does not have to deal with the complexities of the Finnish morphology.

The most recent work on Finnish poem generation is the work presented by Hämäläinen (2018a). This approach consists of individual rule-based verse generators, each of which produces structurally different verses with different types of figurative expression, such as metaphors, tautology, comparison and so on. The verse generators are applied in the order defined by hand-written poem structures. Semantic cohesion is achieved by the fact that each verse generator outputs a noun to the following verse generator in the poem structure. This guarantees that verses are always coherent to some extent with the verse that immediately precedes them. This generator is in use in the creative internet application Poem Machine tailored for co-creativity (Hämäläinen, 2018b).

The previous approaches in Finnish poem generation covered in this section are limited in terms of structural creativity. The approaches are either limited by the structure imposed by the existing poems or sentences, or the hand-written verse structures. The approach we present in our paper showcases more creative freedom on the structural level. This, however, is challenging due to the complicated morphosyntax of Finnish; structural changes can easily render the results nonsensical as wrong morphology in an incorrect syntactic position will make the entire sentence nongrammatical. We take this into account in our proposed method.

Genetic algorithms have been used in the generation of poetic language before. Although not full poem generation, Hervás et al. (2007) have used genetic algorithms for generating alliterations in Spanish. In terms of full poetry generation, Manurung et al. (2012) aim for *grammaticality*, *meaningfulness* and *poeticness* with their genetic algorithm approach. Their approach tires to maximize the similarity of the poem meter to the target meter, and the poem semantics to the target semantics, while still retaining grammaticality.

A recent approach to poem generation with genetic algorithms, TwitSong 3.0 (Lamb and Brown, 2019), is based on a mined corpus of sentences that are used as verses in poems based on their inter-compatibility in terms of rhyming. They base their fitness functions on the following metrics: (*meter, emotion, topicality* and *imagery)*. The fitness functions operate on verse level. They solve emotion and imagery with existing lexicons, topicality is assessed based on trigram and keyword similarity with the desired topic and meter is scored based on how close it is to a iambic meter.

## 3 Poem Generator

This section is dedicated to describing the data used for poem generation, the genetic algorithm and how the Finnish morphosyntax is solved by the system. Special attention is paid to describing the fitness functions, according to which the system can rank its creations.

### 3.1 Data

We crawl Wikisources[1] for Finnish poetry. This way we obtain 6,189 poems. We parse the poems by using the Finnish dependency parser (Haverinen et al., 2014) to obtain syntactic relations, morphological features, part of speech and lemma for each word. This constitutes our poem corpus, denoted as $P$, with verse-level syntactic parsing. These poems are used by the genetic algorithm for the initial population, where a stanza of a human authored poem is treated as one poem.

For semantics, we use the pretrained word2vec word embeddings[2] trained on the Finnish Internet Parsebank (Kanerva et al., 2014). This word2vec model has been trained on lemmatized data, which is important as we are interested in obtaining replacement words in an uninflected form.

### 3.2 Genetic Algorithm

Genetic algorithms are inspired by evolution taking place in the real world. They have an initial set of individuals forming a population. These individuals are then exposed to evolutionary processes such as mutation and crossover. After a generation, the fittest individuals survive to the next generation and the evolutionary process is repeated. For modeling this process, we use the Python library DEAP (Fortin et al., 2012) as the genetic algorithms framework.

We employ a standard $(\mu + \lambda)$ genetic algorithm, which has previously been used in computational creativity applications (see Alnajjar et al.,

[1] https://fi.wikisource.org

[2] http://bionlp-www.utu.fi/fin-vector-space-models/fin-word2vec-lemma.bin

2018). The method begins by constructing an initial population and then evolving it, while optimizing certain parameters, throughout $G$ generations. At each generation step, the fittest $\mu$ individuals in the current population and the $\lambda$ offspring are selected to represent the next population. We empirically set $\mu$ and $\lambda$ to 100 and $G$ to 25. Additionally, the algorithm takes two user-defined inputs, a poem $p$ and a theme $t$. For our case, we considered a theme $t$ as a single word representing an abstract concept such as *nature*; alternatively, a set of words could be used instead to represent a more focused theme (e.g. *tree*, *forest*, *flower*, ... etc).

#### 3.2.1 Initial Population

To build an initial population containing poems with various syntactic structures, the method makes $\mu$ copies of the input poem $p$. For each poem, the method then replaces one verse in it with a random verse from a different poem existing in the poem corpus $P$.

#### 3.2.2 Mutation and Crossover

In our method, we implement one type of mutation which selects a random content word in the poem. The term content word in this case refers to a word that belongs to an open class part-of-speech category. The selected word is substituted with another semantically similar word, which is determined as follows. Let $w$ be the random content word selected to be replaced, the method retrieves the top 300 semantically similar words to $w$ as candidate replacements from the word2vec model. Thereafter, the method uses UralicNLP (Hämäläinen, 2019) to perform morphological analysis on all candidate words. The candidate words that have a different part-of-speech tag than the original word $w$ are omitted out. Out of the remaining candidate words, a random word is picked to substitute $w$.

We use a single-point crossover at the verse-level. In practice, this means that during the evolutionary process two poem individuals are selected and a single point at the beginning of their verses is chosen at random. Verses after that point are swapped between them.

#### 3.2.3 Fitness Functions

The genetic algorithm assesses the individuals based on six metrics that evaluate the poem's structure and one metric that evaluates semantics. The difference in the number of syllables in verses and in the poetic foot, as measured by the distribution of long and short syllables, are contrasted to the original poem. The genetic algorithm is set to minimize these values to keep the difference minimal. As not changing the poem at all would result in the minimum difference in these values, we penalize identical verses by giving them a distance of 20. This way the genetic algorithm tries to make changes so that results following the original meter are preferred.

The number of full rhymes, assonance rhymes and consonance rhymes in between the verses of each generated poem are used as metrics to assess to overall poetic quality of the individuals. The number of alliterating words is counted within verses as this type of rhyme is traditionally occurring within verses in Finnish poetry. The values given by these four metrics are maximized by the genetic algorithm to get the maximum number rhyming words in the final outputs.

The last metric measures the average semantic similarity of the words in the poem to the input theme $t$ with the word2vec model. Maximizing this function pushes the evolutionary process towards creating poems that are close in semantics to the desired input theme.

As we are employing multiple objective functions in our genetic algorithm, we resort to using a non-dominant sorting algorithm (NSGA-II) (Deb et al., 2002) for optimizing these functions. In short, the algorithm selects individuals that are not dominated by any other individual. An individual $x$ is considered to be dominating another if its scores on all objective functions are greater than or equal to $y$'s and it is always better than $y$ on at least one objective.

### 3.3 Surface Generation

As the genetic algorithm does substitutions on the level of lemmas, it is important to be able to turn the verses with new lemmas into grammatical sentences. This is not only needed for presenting the final output produced by the genetic algorithm to people, but also for the fitness functions to work.

In Finnish, the surface form of a word (morphological realization) is affected by two mechanisms: agreement and government. The former means that certain words have to share morphological features in a sentence. For example, adjectives will have to follow the case and number of the noun they modify, like so: *punainen talo* (a

red house) and *punaisessa talossa* (in a red house). This can be accounted for just by inflecting the replacement word with the morphology of the original word. For this purpose we use Omorfi (Pirinen et al., 2017) which implements Finnish morphology as an FST (finite-state transducer).

Government, on the other hand, requires some additional work. In government, words affect on each other morphologically in a way that depends on the governor. This means that if a governor word is replaced by another one in the sentence, the morphology of the governed word needs to adapt to the change. In concrete, given an original verse *uneksin hatusta* (I dream of a hat) and a change of the verb to *näen hatun* (I see a hat), the case of the object for *hattu* has to change from elative to genitive. We resolve government with Syntax maker (Hämäläinen and Rueter, 2018), which resolves the required case based on corpus statistics.

## 4 Results and Evaluation

As evaluation of creative systems is one of the most difficult problems in the field of computational creativity, instead of trying to come up with an evaluation metric of our own, we opt for the evaluation method used to evaluate a previous Finnish poem generator. In practice, this means conducting a quantitative evaluation with human judges with the evaluation questions defined by Toivanen et al. (2012).

An additional reasoning for using human evaluators instead of automated evaluation metrics is the poor correlation observed in a previous study (Hämäläinen and Alnajjar, 2019) of automatic evaluation metrics such as BLEU (Papineni et al., 2002) and PINC (Chen and Dolan, 2011) scores with human judgments when evaluating creativity of a system.

We run the genetic algorithm to produce a final population for 20 different initial poems for four different themes *luonto* (nature), *perhe* (family), *lemmikki* (pet) and *ihminen* (human). From each of the 20 final populations, we pick one poem at random. We shuffle the order of poems to reduce the priming effect of poems appearing always in a given order. We divide the 20 poems into two batches of 10 poems to reduce the effort of an individual evaluator. Each batch of 10 is then evaluated by 10 different human evaluators recruited from the university campus. The total number of evaluators is 20 and all of them are native in Finnish.

We use the following evaluation questions from Toivanen et al. (2012): (1) *How typical is the text as a poem?* (2) *How understandable is it?* (3) *How good is the language?* (4) *Does the text evoke mental images?* (5) *Does the text evoke emotions?* (6) *How much do you like the text?*. These questions are evaluated in a 5 point Likert scale, where 1 represents the worst and 5 the best grade. In addition to these questions, one simple binary question is asked: *Is the text a poem?*.

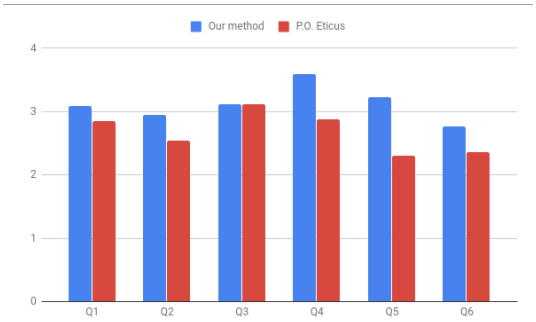


Figure 1: Evaluation results

Figure 1 represents the average values of the results of the human evaluation for each question. The plot also shows the evaluation results of P.O. Eticus as obtained in their study. As we can see, our method shows higher ratings on all the evaluation questions except for question 3. As for the binary question, the judges rated the output as a poem 81.5% of the time which is exactly the same result as P.O. Eticus got.

However, it is to remember that as a high level of subjectivity is involved in this evaluation setting, our results should not be directly compared to those of P.O. Eticus. The results form their study should taken more as a reference, rather than a definite proof that our system always outperforms P.O. Eticus.

| | Q1 | Q2 | Q3 | Q4 | Q5 | Q6 |
|---|---|---|---|---|---|---|
| Average | 3.10 | 2.94 | 3.11 | 3.60 | 3.23 | 2.77 |
| Median | 3 | 3 | 3 | 4 | 3 | 3 |
| Mode | 2 | 2 | 3 | 4 | 4 | 2 |

Table 1: The average, median and mode of the evaluation results

Table 1 shows the median and mode of the results in addition to the average values. The median values seems to correspond to the rounded average values. However, the mode deviates in the case

of the first, second, fifth and last questions as the most chosen answer by the judges was different from the average.

*Ja kultaa, kuninkaankin saan.*
*Ja laulut ne kiertävät maata ja merta*
*Jos virkkaan kun orja*
*Aina, todella Herra pahankurisuutta antaa.*

And gold, of a king I shall have.
And songs, they shall roam on the land and the sea
If I knit like a slave
Always, indeed the Lord shall wrack his mischief.

Above is an example of a poem generated by the system and its translation in English. The poems generated by the system are typically of this length as the genetic algorithm uses a stanza of an existing poem as its starting point.

## 5 Discussion and Conclusion

The method presented in this paper shows improvement on a previously used evaluation metric. However, based on the discussions we had with some of the human evaluators after they had given their judgment, it became evident that people have very different criteria for poetry. Some of the judges had guessed that they were reading computer generated poems, even though this detail was not revealed to them explicitly. Their judgments were the most critical towards the generated poetry. On the other hand, the evaluators, who were struck by a surprise that the poems were generated by a computer, were in general more generous in their judgments. One of the evaluators almost refused to believe the poems were generated by a computer instead of a person.

The high level of subjectivity that we could observe just by talking with people calls for a more robust qualitative study on the poem evaluation problem itself in the future. This would allow us to uncover additional factors that affect on the judgments given by people. Furthermore, conducting a study just on the evaluation itself makes it possible for us to evaluate the adequacy of the used evaluation metric in evaluating computer generated poetry.

Nevertheless, the scores achieved by our system, in relation to a previous method by following the same evaluation metric, are promising as they are indicative of potentially higher quality in the output. We have presented a solution for the Finnish morphosyntax in conjunction with employing a genetic algorithm to cater for computational creativity in poem generation.

## Acknowledgements

Special thanks to Jack Rueter for helping out with the evaluation.

## References

Khalid Alnajjar, Hadaytullah Hadaytullah, and Hannu Toivonen. 2018. "Talent, Skill and Support." A method for automatic creation of slogans. In *Proceedings of the 9th International Conference on Computational Creativity (ICCC 2018)*, pages 88–95, Salamanca, Spain. Association for Computational Creativity.

Benjamin Bay, Paul Bodily, and Dan Ventura. 2017. Text transformation via constraints and word embedding. In *Proceedings of the Eighth International Conference on Computational Creativity*, pages 49–56.

David L Chen and William B Dolan. 2011. Collecting highly parallel data for paraphrase evaluation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 190–200.

Simon Colton. 2008. Creativity Versus the Perception of Creativity in Computational Systems. In *AAAI Spring Symposium: Creative Intelligent Systems*, Technical Report SS-08-03, pages 14—-20, Stanford, California, USA.

Simon Colton, John William Charnley, and Alison Pease. 2011. Computational creativity theory: The face and idea descriptive models. In *ICCC*, pages 90–95.

Simon Colton, Jacob Goodwin, and Tony Veale. 2012. Full-face poetry generation. In *Proceedings of the Third International Conference on Computational Creativity*, pages 95—-102.

K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. 2012. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175.

Pablo Gervás. 2001. An expert system for the composition of formal Spanish poetry. *Knowledge-Based Systems*, 14(3):181–188.

Erica Greene, Tugba Bodrumlu, and Kevin Knight. 2010. Automatic analysis of rhythmic poetry with applications to generation and translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 524–533, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mika Hämäläinen. 2018a. Harnessing NLG to Create Finnish Poetry Automatically. In *Proceedings of the Ninth International Conference on Computational Creativity*, pages 9–15.

Mika Hämäläinen. 2018b. Poem Machine - a Co-creative NLG Web Application for Poem Writing. In *The 11th International Conference on Natural Language Generation: Proceedings of the Conference*, pages 195—-196.

Mika Hämäläinen. 2019. UralicNLP: An NLP library for Uralic languages. *Journal of Open Source Software*, 4(37):1345.

Mika Hämäläinen and Khalid Alnajjar. 2019. Modelling the Socialization of Creative Agents in a Master-Apprentice Setting: The Case of Movie Title Puns. In *Proceedings of the Tenth International Conference on Computational Creativity*, pages 266–273.

Mika Hämäläinen and Jack Rueter. 2018. Development of an Open Source Natural Language Generation Tool for Finnish. In *Proceedings of the Fourth International Workshop on Computational Linguistics for Uralic Languages*, pages 51–58.

Katri Haverinen, Jenna Nyblom, Timo Viljanen, Veronika Laippala, Samuel Kohonen, Anna Missilä, Stina Ojala, Tapio Salakoski, and Filip Ginter. 2014. Building the essential resources for finnish: the turku dependency treebank. *Language Resources and Evaluation*, 48(3):493–531.

Raquel Hervás, Jason Robinson, and Pablo Gervás. 2007. Evolutionary assistance in alliteration and allelic drivel. In *Workshops on Applications of Evolutionary Computation*, pages 537–546. Springer.

Jenna Kanerva, Juhani Luotolahti, Veronika Laippala, and Filip Ginter. 2014. Syntactic n-gram collection from a large-scale corpus of internet Finnish. In *Human Language Technologies-The Baltic Perspective: Proceedings of the Sixth International Conference Baltic HLT*, volume 268, pages 184–191.

Anna Kantosalo, Jukka Toivanen, and Hannu Toivonen. 2015. Interaction Evaluation for Human-Computer Co-creativity: A Case Study. In *Proceedings of the Sixth International Conference on Computational Creativity*, pages 276–283.

Carolyn Lamb and Daniel G. Brown. 2019. TwitSong 3.0: towards semantic revisions in computational poetry. In *Proceedings of the Tenth International Conference on Computational Creativity*, pages 212–219.

Juntao Li, Yan Song, Haisong Zhang, Dongmin Chen, Shuming Shi, Dongyan Zhao, and Rui Yan. 2018. Generating classical Chinese poems via conditional variational autoencoder and adversarial training. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3890–3900, Brussels, Belgium. Association for Computational Linguistics.

Ruli Manurung, Graeme Ritchie, and Henry Thompson. 2012. Using genetic algorithms to create meaningful poetic text. *J. Exp. Theor. Artif. Intell.*, 24:43–64.

Hugo Gonçalo Oliveira. 2017. A survey on intelligent poetry generation: Languages, features, techniques, reutilisation and evaluation. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 11–20, Santiago de Compostela, Spain. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318.

Tommi A Pirinen, Inari Listenmaa, Ryan Johnson, Francis M. Tyers, and Juha Kuokkala. 2017. Open morphology of finnish. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University.

Jukka Toivanen, Hannu Toivonen, Alessandro Valitutti, and Oskar Gross. 2012. Corpus-Based Generation of Content and Form in Poetry. In *Proceedings of the Third International Conference on Computational Creativity*.

Geraint A Wiggins. 2006. A preliminary framework for description, analysis and comparison of creative systems. *Knowledge-Based Systems*, 19(7):449–458.

# SUM-QE: a BERT-based Summary Quality Estimation Model

**Stratos Xenouleas**[1], **Prodromos Malakasiotis**[1],
**Marianna Apidianaki**[2] and **Ion Androutsopoulos**[1]
[1] Department of Informatics, Athens University of Economics and Business, Greece
[2] CNRS, LLF, France and University of Helsinki, Finland
stratosxen@gmail.com, rulller@aueb.gr
marianna.apidianaki@helsinki.fi, ion@aueb.gr

## Abstract

We propose SUM-QE, a novel Quality Estimation model for summarization based on BERT. The model addresses linguistic quality aspects that are only indirectly captured by content-based approaches to summary evaluation, without involving comparison with human references. SUM-QE achieves very high correlations with human ratings, outperforming simpler models addressing these linguistic aspects. Predictions of the SUM-QE model can be used for system development, and to inform users of the quality of automatically produced summaries and other types of generated text.

## 1 Introduction

Quality Estimation (QE) is a term used in machine translation (MT) to refer to methods that measure the quality of automatically translated text without relying on human references (Bojar et al., 2016, 2017). In this study, we address QE for summarization. Our proposed model, SUM-QE, successfully predicts linguistic qualities of summaries that traditional evaluation metrics fail to capture (Lin, 2004; Lin and Hovy, 2003; Papineni et al., 2002; Nenkova and Passonneau, 2004). SUM-QE predictions can be used for system development, to inform users of the quality of automatically produced summaries and other types of generated text, and to select the best among summaries output by multiple systems.

SUM-QE relies on the BERT language representation model (Devlin et al., 2019). We use a pre-trained BERT model adding just a task-specific layer, and fine-tune the entire model on the task of predicting linguistic quality scores manually assigned to summaries. The five criteria addressed are given in Figure 1. We provide a thorough evaluation on three publicly available summarization datasets from NIST shared tasks, and compare the performance of our model to a wide variety of baseline methods capturing different aspects of linguistic quality. SUM-QE achieves very high correlations with human ratings, showing the ability of BERT to model linguistic qualities that relate to both text content and form.[1]

**Q1 – Grammaticality:** The summary should have no datelines, system-internal formatting, capitalization errors or obviously ungrammatical sentences (e.g., fragments, missing components) that make the text difficult to read.

**Q2 – Non redundancy:** There should be no unnecessary repetition in the summary.

**Q3 – Referential Clarity:** It should be easy to identify who or what the pronouns and noun phrases in the summary are referring to.

**Q4 – Focus:** The summary should have a focus; sentences should only contain information that is related to the rest of the summary.

**Q5 – Structure & Coherence:** The summary should be well-structured and well-organized. The summary should not just be a heap of related information, but should build from sentence to sentence to a coherent body of information about a topic.

Figure 1: SUM-QE rates summaries with respect to five linguistic qualities (Dang, 2006a). The datasets we use for tuning and evaluation contain human assigned scores (from 1 to 5) for each of these categories.

## 2 Related Work

Summarization evaluation metrics like Pyramid (Nenkova and Passonneau, 2004) and ROUGE (Lin and Hovy, 2003; Lin, 2004) are recall-oriented; they basically measure the content from a model (reference) summary that is preserved in peer (system generated) summaries. Pyramid requires substantial human effort, even in its more recent versions that involve the use of word embeddings (Passonneau et al., 2013) and a lightweight crowdsourcing scheme (Shapira et al.,

[1] Our code is available at https://github.com/nlpaueb/SumQE



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6005–6011,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

2019). ROUGE is the most commonly used evaluation metric (Nenkova and McKeown, 2012; Allahyari et al., 2017; Gambhir and Gupta, 2017). Inspired by BLEU (Papineni et al., 2002), it relies on common $n$-grams or subsequences between peer and model summaries. Many ROUGE versions are available, but it remains hard to decide which one to use (Graham, 2015). Being recall-based, ROUGE correlates well with Pyramid but poorly with linguistic qualities of summaries. Louis and Nenkova (2013) proposed a regression model for measuring summary quality without references. The scores of their model correlate well with Pyramid and Responsiveness, but text quality is only addressed indirectly.[2]

Quality Estimation is well established in MT (Callison-Burch et al., 2012; Bojar et al., 2016, 2017; Martins et al., 2017; Specia et al., 2018). QE methods provide a quality indicator for translation output at run-time without relying on human references, typically needed by MT evaluation metrics (Papineni et al., 2002; Denkowski and Lavie, 2014). QE models for MT make use of large post-edited datasets, and apply machine learning methods to predict post-editing effort scores and quality (good/bad) labels.

We apply QE to summarization, focusing on linguistic qualities that reflect the readability and fluency of the generated texts. Since no post-edited datasets – like the ones used in MT – are available for summarization, we use instead the ratings assigned by human annotators with respect to a set of linguistic quality criteria. Our proposed models achieve high correlation with human judgments, showing that it is possible to estimate summary quality without human references.

## 3 Datasets

We use datasets from the NIST DUC-05, DUC-06 and DUC-07 shared tasks (Dang, 2006a,b; Over et al., 2007). Given a question and a cluster of newswire documents, the contestants were asked to generate a 250-word summary answering the question. DUC-05 contains 1,600 summaries (50 questions x 32 systems); in DUC-06, 1,750 summaries are included (50 questions x 35 systems); and DUC-07 has 1,440 summaries (45 questions x 32 systems).

[2]In the Responsiveness annotation instructions, annotators were asked to assess the linguistic quality of the summary only if it interfered with the expression of information and reduced the amount of conveyed information. See https://duc.nist.gov/duc2005/responsiveness.assessment.instructions

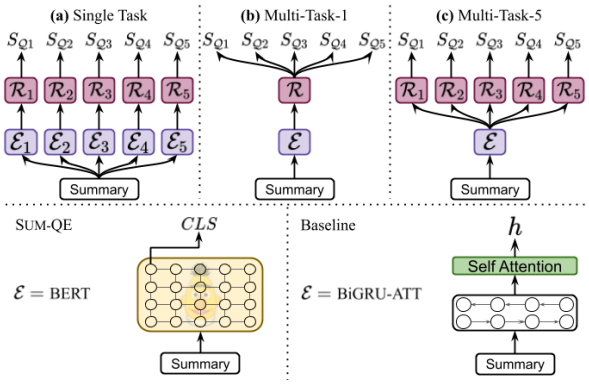


Figure 2: Illustration of different flavors of the investigated neural QE methods. An encoder ($\mathcal{E}$) converts the summary to a dense vector representation $h$. A regressor $\mathcal{R}_i$ predicts a quality score $S_{\mathcal{Q}i}$ using $h$. $\mathcal{E}$ is either a BiGRU with attention (BiGRU-ATT) or BERT (SUM-QE). $\mathcal{R}$ has three flavors, one single-task (a) and two multi-task (b, c).

The submitted summaries were manually evaluated in terms of content preservation using the Pyramid score, and according to five linguistic quality criteria ($\mathcal{Q}1, \ldots, \mathcal{Q}5$), described in Figure 1, that do not involve comparison with a model summary. Annotators assigned scores on a five-point scale, with 1 and 5 indicating that the summary is bad or good with respect to a specific $\mathcal{Q}$. The overall score for a contestant with respect to a specific $\mathcal{Q}$ is the average of the manual scores assigned to the summaries generated by the contestant. Note that the DUC-04 shared task involved seven $\mathcal{Q}$s, but some of them were found to be highly overlapping and were grouped into five in subsequent years (Over et al., 2007).[3] We address these five criteria and use DUC data from 2005 onwards in our experiments.

## 4 Methods

### 4.1 The SUM-QE Model

In SUM-QE, each peer summary is converted into a sequence of token embeddings, consumed by an encoder $\mathcal{E}$ to produce a (dense vector) summary representation $h$. Then, a regressor $\mathcal{R}$ predicts a quality score $S_{\mathcal{Q}}$ as an affine transformation of $h$:

$$S_{\mathcal{Q}} = \mathcal{R}(h) = W^{\mathcal{R}}h + b^{\mathcal{R}} \quad (1)$$

Non-linear regression could also be used, but a

[3]The complete guidelines given to annotators for DUC 2005 and subsequent years can be found at https://duc.nist.gov/duc2005/quality-questions.txt

linear (affine) $\mathcal{R}$ already performs well. We use BERT as our main encoder and fine-tune it in three ways, which leads to three versions of SUM-QE.

**Single-task (BERT-FT-S-1):** The first version of SUM-QE uses five separate estimators, one per quality score, each having its own encoder $\mathcal{E}_i$ (a separate BERT instance generating $h_i$) and regressor $\mathcal{R}_i$ (a separate linear regression layer on top of the corresponding BERT instance):

$$S_{\mathcal{Q}i} = \mathcal{R}_i(h_i), i = 1 \dots 5 \quad (2)$$

**Multi-task with one regressor (BERT-FT-M-1):** The second version of SUM-QE uses one estimator to predict all five quality scores at once, from a single encoding $h$ of the summary, produced by a single BERT instance. The intuition is that $\mathcal{E}$ will learn to create richer representations so that $\mathcal{R}$ (an affine transformation of $h$ with 5 outputs) will be able to predict all quality scores:

$$S_{\mathcal{Q}i} = \mathcal{R}(h)[i], i = 1 \dots 5 \quad (3)$$

where $\mathcal{R}(h)[i]$ is the $i$-th element of the vector returned by $\mathcal{R}$.

**Multi-task with 5 regressors (BERT-FT-M-5):** The third version of SUM-QE is similar to BERT-FT-M-1, but we now use five different linear (affine) regressors, one per quality score:

$$S_{\mathcal{Q}i} = \mathcal{R}_i(h), i = 1 \dots 5 \quad (4)$$

Although BERT-FT-M-5 is mathematically equivalent to BERT-FT-M-1, in practice these two versions of SUM-QE produce different results because of implementation details related to how the losses of the regressors (five or one) are combined.

### 4.2 Baselines

**BiGRUs with attention:** This is very similar to SUM-QE but now $\mathcal{E}$ is a stack of BiGRUs with self-attention (Xu et al., 2015), instead of a BERT instance. The final summary representation ($h$) is the sum of the resulting context-aware token embeddings ($h = \sum_i a_i h_i$) weighted by their self-attention scores ($a_i$). We again have three flavors: one single-task (BiGRU-ATT-S-1) and two multi-task (BiGRU-ATT-M-1 and BiGRU-ATT-M-5).

**ROUGE:** This baseline is the ROUGE version that performs best on each dataset, among the versions considered by Graham (2015). Although ROUGE focuses on surface similarities between peer and reference summaries, we would expect properties like grammaticality, referential clarity and coherence to be captured to some extent by ROUGE versions based on long $n$-grams or longest common subsequences.

**Language model (LM):** For a peer summary, a reasonable estimate of $\mathcal{Q}1$ (Grammaticality) is the perplexity returned by a pre-trained language model. We experiment with the pre-trained GPT-2 model (Radford et al., 2019), and with the probability estimates that BERT can produce for each token when the token is treated as masked (BERT-FR-LM).[4] Given that the grammaticality of a summary can be corrupted by just a few bad tokens, we compute the perplexity by considering only the $k$ worst (lowest LM probability) tokens of the peer summary, where $k$ is a tuned hyper-parameter.[5]

**Next sentence prediction:** BERT training relies on two tasks: predicting masked tokens and next sentence prediction. The latter seems to be aligned with the definitions of $\mathcal{Q}3$ (Referential Clarity), $\mathcal{Q}4$ (Focus) and $\mathcal{Q}5$ (Structure & Coherence). Intuitively, when a sentence follows another with high probability, it should involve clear referential expressions and preserve the focus and local coherence of the text.[6] We, therefore, use a pre-trained BERT model (BERT-FR-NS) to calculate the sentence-level perplexity of each summary:

$$\mathcal{H} = 2^{-\frac{1}{n}\sum_{i=2}^{n} \log_2 p(s_i|s_{i-1})} \quad (5)$$

where $p(s_i|s_{i-1})$ is the probability that BERT assigns to the sequence of sentences $\langle s_{i-1}, s \rangle$, and $n$ is the number of sentences in the peer summary.

## 5 Experiments

To evaluate our methods for a particular $\mathcal{Q}$, we calculate the average of the predicted scores for the summaries of each particular contestant, and the average of the corresponding manual scores assigned to the contestant's summaries. We measure the correlation between the two (predicted vs.

[4] Here BERT parameters are frozen (not fine-tuned). We use the pre-trained masked LM model to obtain probability estimates for the tokens, which are then used to calculate the perplexity.

[5] Consult the supplementary material for details.

[6] We also found the three quality scores to be highly correlated. The reader may refer to the supplementary material for correlation heatmaps between the five quality scores.

| | | DUC-05 | | | DUC-06 | | | DUC-07 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | $\rho$ | $\tau$ | $r$ | $\rho$ | $\tau$ | $r$ | $\rho$ | $\tau$ | $r$ |
| $\mathcal{Q}1$ Grammaticality | BEST-ROUGE | 0.213 | 0.128 | 0.033 | -0.049 | -0.044 | 0.331 | 0.387 | 0.283 | 0.506 |
| | GPT-2 | 0.678 | 0.511 | 0.637 | 0.391 | 0.280 | 0.593 | 0.780 | 0.586 | 0.675 |
| | BERT-FR-LM | 0.437 | 0.319 | 0.025 | 0.524 | 0.354 | 0.667 | 0.598 | 0.453 | 0.566 |
| | BiGRU-ATT-S-1 | 0.119 | 0.079 | 0.116 | 0.263 | 0.182 | 0.459 | 0.119 | 0.085 | 0.494 |
| | BiGRU-ATT-M-1 | 0.190 | 0.144 | 0.091 | 0.619 | 0.462 | 0.757 | 0.332 | 0.235 | 0.662 |
| | BiGRU-ATT-M-5 | 0.156 | 0.160 | 0.040 | 0.613 | 0.466 | 0.771 | 0.315 | 0.215 | 0.584 |
| | BERT-FT-S-1 | 0.681 | 0.543 | **0.817** | **0.907** | **0.760** | **0.929** | 0.845 | 0.672 | **0.930** |
| | BERT-FT-M-1 | 0.675 | 0.543 | 0.805 | 0.889 | 0.749 | 0.902 | **0.851** | **0.684** | 0.896 |
| | BERT-FT-M-5 | **0.712** | **0.564** | 0.802 | 0.883 | 0.732 | 0.925 | 0.840 | 0.680 | 0.902 |
| $\mathcal{Q}2$ Non redundancy | BEST-ROUGE | -0.121 | -0.081 | 0.064 | -0.401 | -0.301 | -0.408 | -0.299 | -0.222 | -0.486 |
| | BiGRU-ATT-S-1 | -0.063 | -0.049 | -0.101 | 0.511 | 0.358 | 0.514 | 0.468 | 0.352 | 0.457 |
| | BiGRU-ATT-M-1 | -0.197 | -0.143 | -0.094 | 0.478 | 0.478 | 0.524 | 0.478 | 0.340 | 0.565 |
| | BiGRU-ATT-M-5 | -0.226 | -0.167 | -0.124 | 0.414 | 0.304 | 0.399 | 0.283 | 0.201 | 0.238 |
| | BERT-FT-S-1 | 0.330 | 0.232 | **0.499** | 0.677 | 0.517 | 0.679 | 0.756 | 0.576 | 0.689 |
| | BERT-FT-M-1 | 0.333 | 0.232 | 0.494 | **0.791** | **0.615** | **0.789** | **0.761** | **0.596** | **0.799** |
| | BERT-FT-M-5 | **0.377** | **0.310** | 0.471 | 0.632 | 0.460 | 0.674 | 0.754 | 0.572 | 0.740 |
| $\mathcal{Q}3$ Referential clarity | BEST-ROUGE | 0.381 | 0.284 | 0.166 | 0.411 | 0.329 | 0.372 | 0.449 | 0.347 | 0.407 |
| | BERT-FR-NS | 0.185 | 0.130 | -0.138 | 0.462 | 0.315 | 0.494 | 0.478 | 0.322 | 0.085 |
| | BiGRU-ATT-S-1 | 0.662 | 0.479 | 0.468 | 0.493 | 0.342 | 0.647 | 0.664 | 0.476 | 0.677 |
| | BiGRU-ATT-M-1 | 0.702 | 0.540 | 0.492 | 0.527 | 0.396 | 0.681 | 0.732 | 0.533 | 0.681 |
| | BiGRU-ATT-M-5 | 0.694 | 0.519 | 0.492 | 0.579 | 0.427 | 0.719 | 0.659 | 0.472 | 0.655 |
| | BERT-FT-S-1 | **0.913** | **0.759** | **0.796** | 0.872 | 0.732 | 0.901 | **0.934** | **0.796** | **0.936** |
| | BERT-FT-M-1 | 0.889 | 0.714 | 0.761 | **0.881** | **0.735** | 0.882 | 0.879 | 0.699 | 0.891 |
| | BERT-FT-M-5 | 0.810 | 0.617 | 0.732 | 0.860 | 0.718 | **0.919** | 0.889 | 0.723 | 0.895 |
| $\mathcal{Q}4$ Focus | BEST-ROUGE | 0.440 | 0.373 | 0.270 | 0.440 | 0.331 | 0.475 | 0.495 | 0.360 | 0.563 |
| | BERT-FR-NS | 0.458 | 0.337 | -0.106 | 0.522 | 0.354 | 0.508 | 0.547 | 0.364 | 0.089 |
| | BiGRU-ATT-S-1 | 0.150 | 0.110 | 0.153 | 0.355 | 0.242 | 0.644 | 0.433 | 0.321 | 0.533 |
| | BiGRU-ATT-M-1 | 0.199 | 0.118 | 0.194 | 0.366 | 0.259 | 0.653 | 0.533 | 0.372 | 0.553 |
| | BiGRU-ATT-M-5 | 0.154 | 0.097 | 0.160 | 0.493 | 0.371 | 0.691 | 0.645 | 0.462 | 0.657 |
| | BERT-FT-S-1 | 0.645 | 0.471 | 0.578 | 0.814 | 0.636 | 0.853 | 0.873 | 0.704 | 0.902 |
| | BERT-FT-M-1 | 0.664 | 0.491 | 0.642 | 0.776 | 0.608 | 0.842 | **0.893** | **0.745** | **0.905** |
| | BERT-FT-M-5 | **0.791** | **0.621** | **0.739** | **0.875** | **0.710** | **0.911** | 0.818 | 0.636 | 0.867 |
| $\mathcal{Q}5$ Structure & Coherence | BEST-ROUGE | 0.391 | 0.300 | 0.039 | 0.080 | 0.056 | 0.023 | 0.370 | 0.292 | 0.293 |
| | BERT-FR-NS | 0.200 | 0.153 | -0.140 | 0.171 | 0.120 | 0.285 | 0.418 | 0.280 | 0.015 |
| | BiGRU-ATT-S-1 | 0.223 | 0.153 | 0.040 | 0.458 | 0.326 | 0.526 | 0.606 | 0.442 | 0.534 |
| | BiGRU-ATT-M-1 | 0.404 | 0.264 | 0.067 | 0.479 | 0.350 | 0.599 | 0.664 | 0.499 | 0.576 |
| | BiGRU-ATT-M-5 | 0.244 | 0.157 | -0.113 | 0.435 | 0.296 | 0.540 | 0.522 | 0.389 | 0.506 |
| | BERT-FT-S-1 | 0.536 | 0.415 | 0.477 | 0.681 | 0.522 | 0.810 | 0.862 | 0.690 | **0.857** |
| | BERT-FT-M-1 | 0.566 | 0.419 | 0.512 | 0.684 | 0.515 | 0.726 | 0.864 | 0.690 | 0.803 |
| | BERT-FT-M-5 | **0.634** | **0.472** | **0.586** | **0.796** | **0.620** | **0.892** | **0.921** | **0.787** | 0.843 |

Table 1: Spearman's $\rho$, Kendall's $\tau$ and Pearson's $r$ correlations on DUC-05, DUC-06 and DUC-07 for $\mathcal{Q}1$–$\mathcal{Q}5$. BEST-ROUGE refers to the version that achieved best correlations and is different across years.

manual) across all contestants using Spearman's $\rho$, Kendall's $\tau$ and Pearson's $r$.

We train and test the SUM-QE and BiGRU-ATT versions using a 3-fold procedure. In each fold, we train on two datasets (e.g., DUC-05, DUC-06) and test on the third (e.g., DUC-07). We follow the same procedure with the three BiGRU-based models. Hyper-parameters are tuned on a held out subset from the training set of each fold.

## 6 Results

Table 1 shows Spearman's $\rho$, Kendall's $\tau$ and Pearson's $r$ for all datasets and models. The three fine-tuned BERT versions clearly outperform all other methods. Multi-task versions seem to perform better than single-task ones in most cases. Especially for $\mathcal{Q}4$ and $\mathcal{Q}5$, which are highly correlated, the multi-task BERT versions achieve the best overall results. BiGRU-ATT also benefits from multi-task learning.

The correlation of SUM-QE with human judgments is high or very high (Hinkle et al., 2003) for all $\mathcal{Q}$s in all datasets, apart from $\mathcal{Q}2$ in DUC-05 where it is only moderate. Manual scores for $\mathcal{Q}2$ in DUC-05 are the highest among all $\mathcal{Q}$s and years (between 4 and 5) and with the smallest standard deviation, as shown in Table 2. Differences among systems are thus small in this respect, and although SUM-QE predicts scores in this range, it struggles to put them in the correct order, as illustrated in Figure 3.

| | DUC-05 | DUC-06 | DUC-07 |
|---|---|---|---|
| $\mathcal{Q}1$ | 3.77 (± 0.42) | 3.58 (± 0.60) | 3.54 (± 0.78) |
| $\mathcal{Q}2$ | **4.41 (± 0.20)** | **4.23 (± 0.26)** | **3.71 (± 0.31)** |
| $\mathcal{Q}3$ | 2.99 (± 0.50) | 3.11 (± 0.52) | 3.20 (± 0.66) |
| $\mathcal{Q}4$ | 3.15 (± 0.41) | 3.60 (± 0.39) | 3.30 (± 0.47) |
| $\mathcal{Q}5$ | 2.18 (± 0.46) | 2.39 (± 0.51) | 2.42 (± 0.59) |

Table 2: Mean manual scores (± standard deviation) for each $\mathcal{Q}$ across datasets. $\mathcal{Q}2$ is the hardest to predict because it has the highest scores and the lowest standard deviation.

BEST-ROUGE has a negative correlation with the ground-truth scores for $\mathcal{Q}2$ since it does not account for repetitions. The BiGRU-based models also reach their lowest performance on $\mathcal{Q}2$ in DUC-05. A possible reason for the higher relative performance of the BERT-based models, which achieve a moderate positive correlation, is that BiGRU captures long-distance relations less effectively than BERT, which utilizes Transformers (Vaswani et al., 2017) and has a larger receptive field. A possible improvement would be a stacked BiGRU, since the states of higher stack layers have a larger receptive field as well.[7]

The BERT multi-task versions perform better with highly correlated qualities like $\mathcal{Q}4$ and $\mathcal{Q}5$ (as illustrated in Figures 2 to 4 in the supplementary material). However, there is not a clear winner among them. Mathematical equivalence does not lead to deterministic results, especially when random initialization and stochastic learning algorithms are involved. An in-depth exploration of this point would involve further investigation, which will be part of future work.

## 7 Conclusion and Future Work

We propose a novel Quality Estimation model for summarization which does not require human references to estimate the quality of automatically produced summaries. SUM-QE successfully predicts qualitative aspects of summaries that recall-oriented evaluation metrics fail to approximate. Leveraging powerful BERT representations, it achieves high correlations with human scores for most linguistic qualities rated, on three different datasets. Future work involves extending the SUM-QE model to capture content-related aspects, either in combination with existing evaluation metrics (like Pyramid and ROUGE) or, preferably, by identifying important information in the original text and modelling its preservation in the proposed summaries. This would preserve SUM-QE's independence from human references, a property of central importance in real-life usage scenarios and system development settings.

[7] As we move up the stack, the states are affected directly by their neighbors and indirectly by the neighbors of their neighbors, and so on.

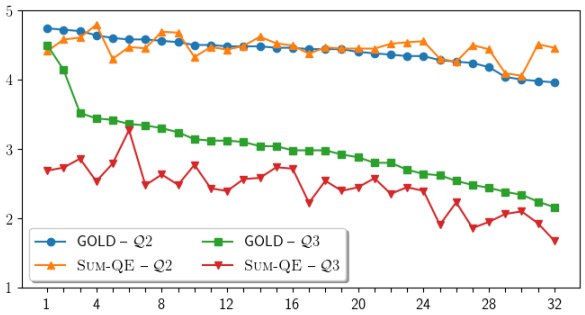


Figure 3: Comparison of the mean gold scores assigned for $\mathcal{Q}2$ and $\mathcal{Q}3$ to each of the 32 systems in the DUC-05 dataset, and the corresponding scores predicted by SUM-QE. Scores range from 1 to 5. The systems are sorted in descending order according to the gold scores. SUM-QE makes more accurate predictions for $\mathcal{Q}2$ than for $\mathcal{Q}3$, but struggles to put the systems in the correct order.

The datasets used in our experiments come from the NIST DUC shared tasks which comprise newswire articles. We believe that SUM-QE could be easily applied to other domains. A small amount of annotated data would be needed for fine-tuning – especially in domains with specialized vocabulary (e.g., biomedical) – but the model could also be used out of the box. A concrete estimation of performance in this setting will be part of future work. Also, the model could serve to estimate linguistic qualities other than the ones in the DUC dataset with mininum effort.

Finally, SUM-QE could serve to assess the quality of other types of texts, not only summaries. It could thus be applied to other text generation tasks, such as natural language generation and sentence compression.

## Acknowledgments

We would like to thank the anonymous reviewers for their helpful feedback on this work. The work has been partly supported by the Research Center of the Athens University of Economics and Business, and by the French National Research Agency under project ANR-16-CE33-0013.

## References


Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saeid Safaei, Elizabeth D. Trippe, Juan B. Gutierrez, and Krys Kochut. 2017. Text Summarization Techniques: A Brief Survey. *Advanced Computer Science and Applications*, 8:397–405.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. 2017. Findings of the 2017 Conference on Machine Translation (WMT17). In *Proceedings of the Second Conference on Machine Translation*, pages 169–214, Copenhagen, Denmark. Association for Computational Linguistics.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016. Findings of the 2016 Conference on Machine Translation. In *Proceedings of the First Conference on Machine Translation*, pages 131–198, Berlin, Germany. Association for Computational Linguistics.

Chris Callison-Burch, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2012. Findings of the 2012 Workshop on Statistical Machine Translation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 10–51, Montréal, Canada. Association for Computational Linguistics.

Hoa Trang Dang. 2006a. DUC 2005: Evaluation of Question-focused Summarization Systems. In *Proceedings of the Workshop on Task-Focused Summarization and Question Answering*, SumQA '06, pages 48–55, Sydney, Australia.

Hoa Trang Dang. 2006b. Overview of DUC 2006. In *Proceedings of the Document Understanding Workshop at HLT-NAACL 2006*, Brooklyn, NY, USA.

Michael Denkowski and Alon Lavie. 2014. Meteor Universal: Language Specific Translation Evaluation for Any Target Language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 376–380, Baltimore, Maryland, USA. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Mahak Gambhir and Vishal Gupta. 2017. Recent automatic text summarization techniques: a survey. *Artificial Intelligence Review*, 47(1):1–66.

Yvette Graham. 2015. Re-evaluating Automatic Summarization with BLEU and 192 Shades of ROUGE. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 128–137, Lisbon, Portugal. Association for Computational Linguistics.

D.E. Hinkle, W. Wiersma, and S.G. Jurs. 2003. *Applied Statistics for the Behavioral Sciences*, volume 663 of *Applied Statistics for the Behavioral Sciences*. Houghton Mifflin.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Workshop on Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic Evaluation of Summaries Using N-gram Co-occurrence Statistics. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 150–157, Edmonton, Canada. Association for Computational Linguistics.

Annie Louis and Ani Nenkova. 2013. Automatically Assessing Machine Summary Content Without a Gold Standard. *Computational Linguistics*, 39(2):267–300.

André F. T. Martins, Marcin Junczys-Dowmunt, Fabio N. Kepler, Ramón Astudillo, Chris Hokamp, and Roman Grundkiewicz. 2017. Pushing the Limits of Translation Quality Estimation. *Transactions of the Association for Computational Linguistics*, 5:205–218.

Ani Nenkova and Kathleen McKeown. 2012. A Survey of Text Summarization Techniques. In Charu C. Aggarwal and ChengXiang Zhai, editors, *Mining Text Data*, pages 43–76. Springer, Boston, MA.

Ani Nenkova and Rebecca Passonneau. 2004. Evaluating Content Selection in Summarization: The Pyramid Method. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pages 145–152, Boston, Massachusetts, USA. Association for Computational Linguistics.

Paul Over, Hoa Dang, and Donna Harman. 2007. DUC in Context. *Information Processing & Management*, 43(6):1506–1520.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia,

Pennsylvania, USA. Association for Computational Linguistics.

Rebecca J. Passonneau, Emily Chen, Weiwei Guo, and Dolores Perin. 2013. Automated Pyramid Scoring of Summaries using Distributional Semantics. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 143–147, Sofia, Bulgaria. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners.

Ori Shapira, David Gabay, Yang Gao, Hadar Ronen, Ramakanth Pasunuru, Mohit Bansal, Yael Amsterdamer, and Ido Dagan. 2019. Crowdsourcing Lightweight Pyramids for Manual Summary Evaluation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 682–687, Minneapolis, Minnesota. Association for Computational Linguistics.

Lucia Specia, Frédéric Blain, Varvara Logacheva, Ramón Astudillo, and André F. T. Martins. 2018. Findings of the WMT 2018 Shared Task on Quality Estimation. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 689–709, Belgium, Brussels. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 2048–2057, Lille, France. PMLR.

# An Empirical Comparison on Imitation Learning and Reinforcement Learning for Paraphrase Generation

**Wanyu Du**
Department of Computer Science
University of Virginia
Charlottesville, VA 22904
wd5jq@virginia.edu

**Yangfeng Ji**
Department of Computer Science
University of Virginia
Charlottesville, VA 22904
yangfeng@virginia.edu

## Abstract

Generating paraphrases from given sentences involves decoding words step by step from a large vocabulary. To learn a decoder, supervised learning which maximizes the likelihood of tokens always suffers from the exposure bias. Although both reinforcement learning (RL) and imitation learning (IL) have been widely used to alleviate the bias, the lack of direct comparison leads to only a partial image on their benefits. In this work, we present an empirical study on how RL and IL can help boost the performance of generating paraphrases, with the pointer-generator as a base model [1]. Experiments on the benchmark datasets show that (1) imitation learning is constantly better than reinforcement learning; and (2) the pointer-generator models with imitation learning outperform the state-of-the-art methods with a large margin.

## 1 Introduction

Generating paraphrases is a fundamental research problem that could benefit many other NLP tasks, such as machine translation (Bahdanau et al., 2014), text generation (Radford et al., 2019), document summarization (Chopra et al., 2016), and question answering (McCann et al., 2018). Although various methods have been developed (Zhao et al., 2009; Quirk et al., 2004; Barzilay and Lee, 2003), the recent progress on paraphrase generation is mainly from neural network modeling (Prakash et al., 2016). Particularly, the encoder-decoder framework is widely adopted (Cho et al., 2015), where the encoder takes source sentences as inputs and the decoder generates the corresponding paraphrase for each input sentence.

In supervised learning, a well-known challenge of generating paraphrases is the exposure bias: the current prediction is conditioned on the ground truth during training but on previous predictions during decoding, which may accumulate and propagate the error when generating the text. To address this challenge, prior work (Li et al., 2018) suggests to utilize the exploration strategy in reinforcement learning (RL). However, training with the RL algorithms is not trivial and often hardly works in practice (Dayan and Niv, 2008). A typical way of using RL in practice is to train the model with supervised learning (Ranzato et al., 2015; Shen et al., 2015; Bahdanau et al., 2016), which leverages the supervision information from training data and alleviate the exposure bias to some extent. In the middle ground between RL and supervised learning, a well-known category is imitation learning (IL) (Daumé et al., 2009; Ross et al., 2011), which has been used in structured prediction (Bagnell et al., 2007) and other sequential prediction tasks (Bengio et al., 2015).[2]

In this work, we conduct an empirical comparison between RL and IL to demonstrate the pros and cons of using them for paraphrase generation. We first propose a unified framework to include some popular learning algorithms as special cases, such as the REINFORCE algorithm (Williams, 1992) in RL and the DAGGER algorithm (Ross et al., 2011) in IL. To better understand the value of different learning techniques, we further offer several variant learning algorithms based on the RL framework. Experiments on the benchmark datasets show: (1) the DAGGER algorithm is better than the REINFORCE algorithm and its variants on paraphrase generation, (2) the DAGGER algorithm with a certain setting gives the best

[1] The data and code for this work can be obtained from: https://github.com/ddddwy/Reinforce-Paraphrase-Generation

[2] In this work, we view *scheduled sampling* (Bengio et al., 2015) as an imitation learning algorithm similar to DAGGER (Ross et al., 2011).



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6012–6018,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

results, which outperform the previous state-of-the-art with about 13% on the average evaluation score. We expect this work will shed light on how to choose between RL and IL, and alleviate the exposure bias for other text generation tasks.

## 2 Method

Given an input sentence $\boldsymbol{x} = (x_1, x_2, \cdots, x_S)$ with length $S$, a paraphrase generation model outputs a new sentence $\boldsymbol{y} = (y_1, y_2, \cdots, y_T)$ with length $T$ that shares the same meaning with $\boldsymbol{x}$. The widely adopted framework on paraphrase generation is the encoder-decoder framework (Cho et al., 2014). The encoder reads sentence $\boldsymbol{x}$ and represents it as a single numeric vector or a set of numeric vectors. The decoder defines a probability function $p(y_t \mid \boldsymbol{y}_{\leq t-1}, \boldsymbol{x}; \boldsymbol{\theta})$, where $\boldsymbol{y}_{\leq t-1} = (y_1, y_2, \ldots, y_{t-1})$ and $\boldsymbol{\theta}$ is the collection of model parameters,

$$p(y_t \mid \boldsymbol{y}_{\leq t-1}, \boldsymbol{x}; \boldsymbol{\theta}) = \text{softmax}\left(\mathbf{W}\boldsymbol{h}_t\right) \quad (1)$$

with $\boldsymbol{h}_t = \boldsymbol{f}(\boldsymbol{h}_{t-1}, y_{t-1}, \boldsymbol{x})$, where $\boldsymbol{f}$ as a nonlinear transition function and $\mathbf{W} \in \boldsymbol{\theta}$ as a parameter matrix. We use the *pointer-generator model* (See et al., 2017) as the base model, which is state-of-the-art model on paragraph generation (Li et al., 2018). We skip the detail explanation of this model and please refer to (See et al., 2017) for further information.

### 2.1 Basic Learning Algorithms

To facilitate the comparison between RL and IL, we propose a unified framework with the following objective function. Given a training example $(\boldsymbol{x}, \boldsymbol{y})$, the objective function is defined as

$$L(\boldsymbol{\theta}) = \Big\{ \sum_{t=1}^{T} \log \pi_\theta(\tilde{y}_t \mid \boldsymbol{h}_t) \Big\} \cdot r(\tilde{\boldsymbol{y}}, \boldsymbol{y}), \quad (2)$$

Following the terminology in RL and IL, we rename $P(\tilde{y}_t \mid \boldsymbol{y}_{\leq t-1}, \boldsymbol{x}; \boldsymbol{\theta})$ as the the policy function $\pi_\theta(\tilde{y}_t \mid \boldsymbol{h}_t)$. That implies taking an action based on the current observation, where the action is *picking* a word $\tilde{y}_t$ from the vocabulary $\mathcal{V}$. $r(\tilde{\boldsymbol{y}}, \boldsymbol{y})$ is a reward function with $r(\tilde{\boldsymbol{y}}, \boldsymbol{y}) = 1$ if $\tilde{\boldsymbol{y}} = \boldsymbol{y}$. In our experiments, We use the ROUGE-2 score (Lin, 2004) as the reward function. Algorithm 1 presents how to optimize $L(\boldsymbol{\theta})$ in the online learning fashion. As shown in the pseudocode, the **schedule rates** $(\alpha, \beta)$ and the **decoding function** $\texttt{Decode}(\cdot)$ are the keys to understand the special cases of this unified framework.

**Algorithm 1** Online learning

**Input**: A training example $(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})$, current schedule rates $\alpha^{(i)}, \beta^{(i)} \in [0, 1]$, learning rate $\eta$
Initialize $L(\boldsymbol{\theta}) \leftarrow 0$
**for** $t = 1, \ldots, T$ **do**
  $p_1, p_2 \sim \text{Uniform}(0, 1)$
  $\tilde{y}_{t-1} \leftarrow y_{t-1}$ **if** $(p_1 < \alpha^{(i)})$ **else** $\hat{y}_{t-1}$
  $\boldsymbol{h}_t = \boldsymbol{f}(\boldsymbol{h}_{t-1}, \tilde{y}_{t-1}, \boldsymbol{x})$
  $\hat{y}_t \leftarrow \texttt{Decode}(\pi(y \mid \boldsymbol{h}_t))$
  $\tilde{y}_t \leftarrow y_t$ **if** $(p_2 < \beta^{(i)})$ **else** $\hat{y}_t$
  $L(\boldsymbol{\theta}) \leftarrow L(\boldsymbol{\theta}) + \log \pi(\tilde{y}_t \mid \boldsymbol{h}_t)$
**end for**
$\delta\boldsymbol{\theta} \leftarrow \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) \cdot r(\tilde{\boldsymbol{y}^{(i)}}, \boldsymbol{y}^{(i)})$
$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta \cdot \delta\boldsymbol{\theta}$

**The REINFORCE Algorithm.** When $\alpha = 0$, $\beta = 0$, and $\texttt{Decode}(\pi(y \mid \boldsymbol{h}_t))$ is defined as as:

$$\begin{aligned} &\texttt{Decode}(\pi_\theta(y \mid \boldsymbol{h}_{t-1})) \\ &\quad = \text{Random\_Sampling}(\pi_\theta(y \mid \boldsymbol{h}_{t-1})), \end{aligned} \quad (3)$$

Specifically, when $\alpha = \beta = 0$, both $\tilde{y}_{t-1}$ and $\tilde{y}_t$ will choose the sampled values from the $\texttt{Decode}$ function with policy $\pi_\theta$. It essentially samples a trajectory from the decoder $(\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_T)$ as in the REINFORCE algorithm. The reward is $r(\tilde{\boldsymbol{y}}, \boldsymbol{y}) = r(\hat{\boldsymbol{y}}, \boldsymbol{y})$ once it has the entire trajectory $\hat{\boldsymbol{y}}$.

**The DAGGER Algorithm.** When $0 < \alpha < 1$, $\beta = 1$, and $\texttt{Decode}(\pi(y \mid \boldsymbol{h}_t))$ is defined as as:

$$\texttt{Decode}(\pi_\theta(y \mid \boldsymbol{h}_{t-1})) = \operatorname*{argmax}_{y'} \pi_\theta(y' \mid \boldsymbol{h}_{t-1}). \quad (4)$$

Depending the value of $\alpha$, $\tilde{y}_{t-1}$ will choose between the ground truth $y_{t-1}$ and decoded value $\hat{y}_{t-1}$ with the function defined in Equation 4. On the other hand, $\tilde{y}_t$ will always choose the ground truth $y_t$ as $\beta = 1$. Since $\tilde{\boldsymbol{y}} = \boldsymbol{y}$, we have $r(\tilde{\boldsymbol{y}}, \boldsymbol{y}) = 1$ and the reward can be ignored from Equation 2. In imitation learning, ground truth sequence $\boldsymbol{y}$ is called expert actions. The DAGGER algorithm (Ross et al., 2011) is also called scheduled sampling (Bengio et al., 2015) in recent deep learning literature. To be accurate, in the DAGGER and the scheduled sampling, the $\alpha$ is dynamically changed during training. Typically, it starts from 1 and gradually decays to a certain value along with iterations. As shown in our experiments, the selection of decay scheme has a big impact on model performance.

**The MLE Algorithm.** Besides, there is a trivial case when $\alpha = 1, \beta = 1$. In this case, $\tilde{y}_{t-1}$ and $\tilde{y}_t$ are equal to $y_{t-1}$ and $y_t$ respectively, and $r(\tilde{\boldsymbol{y}}, \boldsymbol{y}) = 1$. Optimizing the objective function in Equation 2 is reduced to the maximum likelihood estimation (MLE).

### 2.2 Other Variant Algorithms

Inspired by the previous three special cases, we offer other algorithm variants with different combinations of $(\alpha, \beta)$, while the decoding function $\texttt{Decode}(\pi(y \mid \boldsymbol{h}_t))$ in the same as Equation 3 in all following variants.

- **REINFORCE-GTI** (REINFORCE with Ground Truth Input): $\alpha = 1, \beta = 0$. Unlike the REINFORCE algorithm, REINFORCE-GTI restricts the input to the decoder can only be ground truth words, which means $\tilde{y}_{t-1} = y_{t-1}$. This is a popular implementation in the deep reinforcement learning for Seq2Seq models (Keneshloo et al., 2018).

- **REINFORCE-SO** (REINFORCE with Sampled Output): $\alpha = 1, 0 < \beta < 1$. In terms of choosing the value of $\tilde{y}_t$ as output from the decoder, REINFORCE-SO allows $\tilde{y}_t$ to select the ground truth $y_t$ with probability $\beta$.

- **REINFORCE-SIO** (REINFORCE with Sampled Input and Output): $0 < \alpha < 1, 0 < \beta < 1$. Instead of always taking the ground truth $y_{t-1}$ as input, REINFORCE-SIO further relaxes the constraint in REINFORCE-SO and allows $\tilde{y}_{t-1}$ to be the decoded value $\hat{y}_{t-1}$ with probability $\alpha$.

Unless specified explicitly, an additional requirement when $0 < \alpha, \beta < 1$ is that its value decays to a certain value during training, which by default is 0.

## 3 Experiments

**Dataset and Evaluation Metrics.** We evaluate our models on the Quora Question Pair Dataset [3], and the Twitter URL Paraphrasing Dataset (Lan et al., 2017) [4]. Both datasets contain positive and negative examples of paraphrases, and we only keep the positive examples for our experiments as in prior work of paraphrase generation (Li et al., 2018; Patro et al., 2018). For the Quora dataset, we follows the configuration of (Li et al., 2018) and split the data into 100K training pairs, 30K testing pairs and 3K validation pairs. For the Twitter dataset, since our model cannot deal with the negative examples as (Li et al., 2018) do, we just obtain the 1-year 2,869,657 candidate pairs from https://languagenet.github.io, and filter out all negative examples. Finally, we divided the remaining dataset into 110K training pairs, 3K testing pairs and 1K validation pairs.

We use the following evaluation metrics to compare our models with other state-of-art neural networks: ROUGE-1 and ROUGE-2 (Lin, 2004), BLEU with up to bi-grams (Papineni et al., 2002). For the convenience of comparison, we also calculate the average of the scores.

**Competitive Systems.** We compare our results with four competitive systems on paraphrase generation: the sequence-to-sequence model (Bahdanau et al., 2014, Seq2seq), the Reinforced by Matching framework (Li et al., 2018, RbM), the Residual LSTM (Prakash et al., 2016, Res-LSTM), and the Discriminator LSTM model (Patro et al., 2018, Dis-LSTM). Among these competitive systems, the RbM (Li et al., 2018) is more closely related to our work, since we both use the pointer-generator as the base model and apply some reinforcement learning algorithms for policy learning.

**Experimental Setup.** We first pre-train the pointer-generator model with MLE, then fine-tune the models with various algorithms proposed in section 2. Pre-training is critical to make the REINFORCE algorithm and some variants to work. More implementation details are provided in Appendix A.

**Result Analysis.** Table 1 shows the model performances on the Quora test set, and Table 2 shows the model performances on the Twitter test set. For the Quora dataset, all our models outperform the competitive systems with a large margin. We suspect the reason is because we ran the development set during training on-the-fly, which is not the experimental setup used in (Li et al., 2018).

For both datasets, we find that DAGGER with a fixed $(\alpha, \beta)$ gives the best performance among all the algorithm variants. The difference between DAGGER and DAGGER* is that, in DAGGER, we use the decay function on $\alpha$ at each iteration,

[3] https://www.kaggle.com/c/quora-question-pairs

[4] https://languagenet.github.io

| | Models | Schedule Rate $\alpha$ | Schedule Rate $\beta$ | Evaluation Metrics ROUGE-1 | ROUGE-2 | BLEU | Avg-Score |
|---|---|---|---|---|---|---|---|
| 1 | Seq2Seq | - | - | 58.77 | 31.47 | 36.55 | 42.26 |
| 2 | Res-LSTM | - | - | 59.21 | 32.43 | 37.38 | 43.00 |
| 3 | RbM | - | - | 64.39 | 38.11 | 43.54 | 48.68 |
| 4 | Dis-LSTM | - | - | - | 44.90 | 45.70 | 45.30 |
| 5 | Pre-trained MLE | $\alpha = 1$ | $\beta = 1$ | 66.72 | 47.70 | 54.01 | 56.14 |
| 6 | Reinforce | $\alpha = 0$ | $\beta = 0$ | 67.00 | 47.91 | 54.06 | 56.32 |
| 7 | Reinforce-GTI | $\alpha = 1$ | $\beta = 0$ | 67.03 | 48.10 | 54.23 | 56.45 |
| 8 | Reinforce-SO | $\alpha = 1$ | $\beta \to 0$ | 66.88 | 47.95 | 54.16 | 56.33 |
| 9 | Reinforce-SIO | $\alpha \to 0$ | $\beta \to 0$ | 67.62 | 48.99 | 55.19 | 57.26 |
| 10 | Dagger | $\alpha \to 0$ | $\beta = 1$ | 67.64 | 48.96 | 55.06 | 57.22 |
| 11 | Dagger* | $\alpha = 0.5$ | $\beta = 1$ | **68.34** | **49.99** | **55.75** | **58.02** |

Table 1: Performance on Quora dataset. The results of competitive systems are reprinted from prior work: line 1 – 3 are obtained from (Li et al., 2018), line 4 is obtained from (Patro et al., 2018). The average score in the last column is for the convenience of comparison.

| | Models | Schedule Rate $\alpha$ | Schedule Rate $\beta$ | Evaluation Metrics ROUGE-1 | ROUGE-2 | BLEU | Avg-Score |
|---|---|---|---|---|---|---|---|
| 1 | Pre-trained MLE | $\alpha = 1$ | $\beta = 1$ | 58.49 | 43.84 | 38.45 | 46.92 |
| 2 | Reinforce | $\alpha = 0$ | $\beta = 0$ | 58.67 | 44.06 | 38.46 | 47.06 |
| 3 | Reinforce-GTI | $\alpha = 1$ | $\beta = 0$ | 58.58 | 43.89 | 38.42 | 46.96 |
| 4 | Reinforce-SO | $\alpha = 1$ | $\beta \to 0$ | 58.58 | 43.89 | 38.41 | 46.96 |
| 5 | Reinforce-SIO | $\alpha \to 0$ | $\beta \to 0$ | 58.82 | 44.10 | 38.85 | 47.25 |
| 6 | Dagger | $\alpha \to 0$ | $\beta = 1$ | 58.84 | 44.24 | 38.95 | 47.34 |
| 7 | Dagger* | $\alpha = 0.2$ | $\beta = 1$ | **58.95** | **44.34** | **39.04** | **47.44** |

Table 2: Performance on Twitter dataset. Since the dataset we obtained is different from (Li et al., 2018), we do not directly compare the results with the prior works.

$\alpha \leftarrow k \cdot \alpha$ with $k = 0.9999$. In our experiments, we also try different decaying rates, and present the best results we obtained (more details are provided in Table B). The selection of $\alpha$ depends on the specific task: for the Quora dataset, we find $\alpha = 0.5$ gives us the optimal policy; for the Twitter dataset, we find $\alpha = 0.2$ gives us the optimal policy.

As shown in line 6 – 11 from Table 1, the additional training with whichever variant algorithms can certainly enhance the generation performance over the pre-trained model (line 5). This observation is consistent with many previous works of using RL/IL in NLP. However, we also notice that the improvement of using the Reinforce algorithm (line 6) is very small, only 0.18 on the average score.

As shown in line 2 – 7 from Table 2, the additional training with variant algorithms also shows improved performance over the pre-trained model (line 1). However, for the pointer-generator model, it is more difficult to do paraphrase generation on the Twitter dataset. Since in the Twitter dataset, one source sentence shares several different paraphrases, while in the Quora dataset, one source sentence only corresponds to one paraphrase. This explains why the average improvement in the Twitter dataset is not as significant as in the Quora dataset. Besides, from Table 2, we also find that IL (line 6 – 7) outperforms RL (line 2 – 3), which is consist with the experimental results in Table 1.

Overall, in this particular setting of paraphrase generation, we found that Dagger is much easier to use than the Reinforce algorithm, as it always takes ground truth (expert actions) as its outputs. Although, picking a good decay function $\alpha$ can be really tricky. On the other hand, the Reinforce

algorithm (together with its variants) could only outperform the pre-trained baseline with a small margin.

## 4 Related Work

Paraphrase generation has the potential of being used in many other NLP research topics, such as machine translation (Madnani et al., 2007) and question answering (Buck et al., 2017; Dong et al., 2017). Early work mainly focuses on extracting paraphrases from parallel monolingual texts (Barzilay and McKeown, 2001; Ibrahim et al., 2003; Pang et al., 2003). Later, Quirk et al. (2004) propose to use statistical machine translation for generating paraphrases directly. Despite the particular MT system used in their work, the idea is very similar to the recent work of using encoder-decoder frameworks for paraphrase generation (Prakash et al., 2016; Mallinson et al., 2017). In addition, Prakash et al. (2016) extend the encoder-decoder framework with a stacked residual LSTM for paraphrase generation. Li et al. (2018) propose to use the pointer-generator model (See et al., 2017) and train it with an actor-critic RL algorithm. In this work, we also adopt the pointer-generator framework as the base model, but the learning algorithms are developed by uncovering the connection between RL and IL.

Besides paraphrase generation, many other NLP problems have used some RL or IL algorithms for improving performance. For example, structured prediction has more than a decade history of using imitation learning (Daumé et al., 2009; Chang et al., 2015; Vlachos, 2013; Liu et al., 2018). In addition, scheduled sampling (as another form of DAGGER) has been used in sequence prediction ever since it was proposed in (Bengio et al., 2015). Similar to IL, reinforcement learning, particularly with neural network models, has been widely used in many different domains, such as coreference resolution (Yin et al., 2018), document summarization (Chen and Bansal, 2018), and machine translation (Wu et al., 2018).

## 5 Conclusion

In this paper, we performed an empirical study on some reinforcement learning and imitation learning algorithms for paraphrase generation. We proposed a unified framework to include the DAGGER and the REINFORCE algorithms as special cases and further presented some variant learning algorithms. The experiments demonstrated the benefits and limitations of these algorithms and provided the state-of-the-art results on the Quora dataset.

## Acknowledgments

The authors thank three anonymous reviewers for their useful comments and the UVa NLP group for helpful discussion. This research was supported in part by a gift from Tencent AI Lab Rhino-Bird Gift Fund.

## References

JA Bagnell, Joel Chestnutt, David M Bradley, and Nathan D Ratliff. 2007. Boosting structured prediction for imitation learning. In *Advances in Neural Information Processing Systems*, pages 1153–1160.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2016. An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 16–23, Stroudsburg, PA, USA. Association for Computational Linguistics.

Regina Barzilay and Kathleen R McKeown. 2001. Extracting paraphrases from a parallel corpus. In *Proceedings of the 39th annual meeting of the Association for Computational Linguistics*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179.

Christian Buck, Jannis Bulian, Massimiliano Ciaramita, Wojciech Gajewski, Andrea Gesmundo, Neil Houlsby, and Wei Wang. 2017. Ask the right questions: Active question reformulation with reinforcement learning. *arXiv preprint arXiv:1705.07830*.

Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daumé III, and John Langford. 2015. Learning to search better than your teacher. *arXiv preprint arXiv:1502.02206*.

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. *arXiv preprint arXiv:1805.11080*.

Kyunghyun Cho, Aaron Courville, and Yoshua Bengio. 2015. Describing multimedia content using attention-based encoder-decoder networks. *IEEE Transactions on Multimedia*, 17(11):1875–1886.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Sumit Chopra, Michael Auli, and Alexander M Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98.

Hal Daumé, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine learning*, 75(3):297–325.

Peter Dayan and Yael Niv. 2008. Reinforcement learning: the good, the bad and the ugly. *Current opinion in neurobiology*, 18(2):185–196.

Li Dong, Jonathan Mallinson, Siva Reddy, and Mirella Lapata. 2017. Learning to paraphrase for question answering. *arXiv preprint arXiv:1708.06022*.

Ali Ibrahim, Boris Katz, and Jimmy Lin. 2003. Extracting structural paraphrases from aligned monolingual corpora. In *Proceedings of the second international workshop on Paraphrasing-Volume 16*, pages 57–64.

Yaser Keneshloo, Tian Shi, Naren Ramakrishnan, and Chandan K. Reddy. 2018. Deep reinforcement learning for sequence to sequence models. *arXiv preprint arXiv:1805.09461*.

Wuwei Lan, Siyu Qiu, Hua He, and Wei Xu. 2017. A continuously growing dataset of sentential paraphrases. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1224–1234, Copenhagen, Denmark. Association for Computational Linguistics.

Zichao Li, Xin Jiang, Lifeng Shang, and Hang Li. 2018. Paraphrase generation with deep reinforcement learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Ming Liu, Wray Buntine, and Gholamreza Haffari. 2018. Learning how to actively learn: A deep imitation learning approach. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1874–1883.

Nitin Madnani, Necip Fazil Ayan, Philip Resnik, and Bonnie J Dorr. 2007. Using paraphrases for parameter tuning in statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 120–127. Association for Computational Linguistics.

Jonathan Mallinson, Rico Sennrich, and Mirella Lapata. 2017. Paraphrasing revisited with neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 881–893.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*.

Bo Pang, Kevin Knight, and Daniel Marcu. 2003. Syntax-based alignment of multiple translations: Extracting paraphrases and generating new sentences. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 102–109.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Badri N Patro, Vinod K Kurmi, Sandeep Kumar, and Vinay P Namboodiri. 2018. Learning semantic sentence embeddings using pair-wise discriminator. *arXiv preprint arXiv:1806.00807*.

Aaditya Prakash, Sadid A Hasan, Kathy Lee, Vivek Datla, Ashequl Qadir, Joey Liu, and Oladimeji Farri. 2016. Neural paraphrase generation with stacked residual LSTM networks. *arXiv preprint arXiv:1610.03098*.

Chris Quirk, Chris Brockett, and William Dolan. 2004. Monolingual machine translation for paraphrase generation. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 142–149.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1:8.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2015. Minimum risk training for neural machine translation. *arXiv preprint arXiv:1512.02433*.

Andreas Vlachos. 2013. An investigation of imitation learning algorithms for structured prediction. In *European Workshop on Reinforcement Learning*, pages 143–154.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Lijun Wu, Fei Tian, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2018. A study of reinforcement learning for neural machine translation. *arXiv preprint arXiv:1808.08866*.

Qingyu Yin, Yu Zhang, Weinan Zhang, Ting Liu, and William Yang Wang. 2018. Deep reinforcement learning for chinese zero pronoun resolution. *arXiv preprint arXiv:1806.03711*.

Shiqi Zhao, Xiang Lan, Ting Liu, and Sheng Li. 2009. Application-driven statistical paraphrase generation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, pages 834–842, Stroudsburg, PA, USA. Association for Computational Linguistics.

# Countering the Effects of Lead Bias in News Summarization via Multi-Stage Training and Auxiliary Losses

**Matt Grenander**[*], **Yue Dong**[*]
McGill University / MILA
{matthew.grenander, yue.dong2}@mail.mcgill.ca

**Jackie C. K. Cheung**
McGill University / MILA
jcheung@cs.mcgill.ca

**Annie Louis**
University of Edinburgh
Alan Turing Institute
alouis@inf.ed.ac.uk

## Abstract

Sentence position is a strong feature for news summarization, since the lead often (but not always) summarizes the key points of the article. In this paper, we show that recent neural systems excessively exploit this trend, which although powerful for many inputs, is also detrimental when summarizing documents where important content should be extracted from later parts of the article. We propose two techniques to make systems sensitive to the importance of content in different parts of the article. The first technique employs 'unbiased' data; i.e., randomly shuffled sentences of the source document, to pretrain the model. The second technique uses an auxiliary ROUGE-based loss that encourages the model to distribute importance scores throughout a document by mimicking sentence-level ROUGE scores on the training data. We show that these techniques significantly improve the performance of a competitive reinforcement learning based extractive system, with the auxiliary loss being more powerful than pretraining.

## 1 Introduction

Extractive summarization remains a simple and fast approach to produce summaries which are grammatical and accurately represent the source text. In the news domain, these systems are able to use a dominant signal: the position of a sentence in the source document. Due to journalistic conventions which place important information early in the articles, the lead sentences often contain key information. In this paper, we explore how systems can look beyond this simple trend.

Naturally, automatic systems have all along exploited position cues in news as key indicators of important content (Schiffman et al., 2002; Hong and Nenkova, 2014; Liu, 2019). The 'lead' baseline is rather strong in single-document news summarization (Brandow et al., 1995; Nenkova, 2005), with automatic systems only modestly improving the results. Nevertheless, more than 20-30% of summary-worthy sentences come from the second half of news documents (Nallapati et al., 2016; Kedzie et al., 2018), and the lead baseline, as shown in Table 1, does not always produce convincing summaries. So, systems must balance the position bias with representations of the semantic content throughout the document. Alas, preliminary studies (Kedzie et al., 2018) suggest that even the most recent neural methods predominantly pick sentences from the lead, and that their content selection performance drops greatly when the position cues are withheld.

| |
|---|
| **Lead-3:** Bangladesh beat fellow World Cup quarter-finalists Pakistan by 79 runs in the first one-day international in Dhaka. Tamim Iqbal and Mushfiqur Rahim scored centuries as Bangladesh made 329 for six and Pakistan could only muster 250 in reply. Pakistan will have the chance to level the three-match series on Sunday when the second odi takes place in Mirpur. |
| **Reference:** Bangladesh beat fellow World Cup quarter-finalists Pakistan by 79 runs. Tamim Iqbal and Mushfiqur Rahim scored centuries for Bangladesh. Bangladesh made 329 for six and Pakistan could only muster 250 in reply. Pakistan will have the chance to level the three-match series on Sunday. |
| **Lead-3**: Standing up for what you believe. What does it cost you? What do you gain? |
| **Reference:** Indiana town's Memories Pizza is shut down after online threat. Its owners say they'd refuse to cater a same-sex couple's wedding. |

Table 1: 'Lead' (first 3 sentences of source) can produce extremely faithful (top) to disastrously inaccurate (bottom) summaries. Gold standard summaries are also shown.

In this paper, we verify that sentence position and lead bias dominate the learning signal for state-of-the-art neural extractive summarizers in the news domain. We then present techniques to improve content selection in the face of this bias. The first technique makes use of 'unbiased data' created by permuting the order of sentences in

[*]Equal contribution.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6019–6024, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

the training articles. We use this shuffled dataset for pre-training, followed by training on the original (unshuffled) articles. The second method introduces an auxiliary loss which encourages the model's scores for sentences to mimic an estimated score distribution over the sentences, the latter computed using ROUGE overlap with the gold standard. We implement these techniques for two recent reinforcement learning based systems, RNES (Wu and Hu, 2018) and BanditSum (Dong et al., 2018), and evaluate them on the CNN/Daily Mail dataset (Hermann et al., 2015).

We find that our auxiliary loss achieves significantly better ROUGE scores compared to the base systems, and that the improvement is even more pronounced when the true best sentences appear later in the article. On the other hand, the pre-training approach produces mixed results. We also confirm that when summary-worthy sentences appear late, there is a large performance discrepancy between the oracle summary and state-of-the-art summarizers, indicating that learning to balance lead bias with other features of news text is a noteworthy issue to tackle.

## 2 Related Work

Modern summarization methods for news are typically based on neural network-based sequence-to-sequence learning (Kalchbrenner et al., 2014; Kim, 2014; Chung et al., 2014; Yin and Pei, 2015; Cao et al., 2015; Cheng and Lapata, 2016; Nallapati et al., 2017; Narayan et al., 2018a; Zhou et al., 2018). In MLE-based training, extractive summarizers are trained with gradient ascent to maximize the likelihood of heuristically-generated ground-truth binary labels (Nallapati et al., 2017). Many MLE-based models do not perform as well as their reinforcement learning-based (RL) competitors that directly optimize ROUGE (Paulus et al., 2018; Narayan et al., 2018b; Dong et al., 2018; Wu and Hu, 2018). As RL-based models represent the state of the art for extractive summarization, we analyze them in this paper.

The closest work to ours is a recent study by Kedzie et al. (2018) which showed that MLE-based models learn a significant bias for selecting early sentences when trained on news articles as opposed to other domains. As much as 58% of selected summary sentences come directly from the lead. Moreover, when these models are trained on articles whose sentences are randomly shuffled, the performance drops considerably for news domain only. While this drop could be due to the destruction of position cues, it may also arise because the article's coherence and context were lost.

In this paper, we employ finer control on the distortion of sentence position, coherence, and context, and confirm that performance drops are mainly due to the lack of position cues. We also propose the first techniques to counter the effects of lead bias in neural extractive systems.

## 3 Base Models for Extractive Summarization

In supervised systems, given a document $D = \{s_1, ..., s_n\}$ with $n$ sentences, a summary can be seen as set of binary labels $y_1, \ldots, y_n \in \{0, 1\}$, where $y_i = 1$ indicates that the $i$-th sentence is included in the summary.

We choose to experiment with two state-of-the-art RL-based extractive models: **RNES** (Wu and Hu, 2018) and **BanditSum** (Dong et al., 2018). Both employ an encoder-decoder structure, where the encoder extracts sentence features into fixed-dimensional vector representations $h_1, \ldots, h_n$, and a decoder produces the labels $y_1, \ldots, y_n$ based on these sentence representations. RNES uses a CNN+bi-GRU encoder, and BanditSum a hierarchical bi-LSTM. RNES's decoder is *auto-regressive*, meaning it predicts the current sentence's label based on decisions made on previous sentences; i.e., $y_t = f(D, h_t, y_{1:t-1})$. In BanditSum, there is no such dependence: it produces affinity scores for each sentence and the top scoring sentences are then selected.

## 4 Lead Bias of News Systems

First, we investigate the impact of sentence position on our models. We manipulate the **original** CNN/Daily Mail dataset to preserve sentence position information at different levels. In the **random** setting, sentences are shuffled randomly; in **reverse**, they are in reverse order; in **insert-lead** and **insert-lead3**, we insert an out-of-document sentence (chosen randomly from the corpus) as the first sentence or randomly as one of the first three sentences, respectively.

In Table 2, we show BanditSum's performance,[1] when trained and tested on the various datasets. All models (except random) perform

[1] We notice the same trends on RNES.

| train setting \ test setting | original | random | reverse | insert-lead | insert-lead3 | Mean | Std. Dev. |
|---|---|---|---|---|---|---|---|
| Lead-3 baseline | 32.68 | 22.81 | 17.94 | 27.67 | 27.68 | 25.76 | 5.00 |
| original | **33.85** | 26.18 | 20.71 | 31.71 | 31.11 | 28.71 | 4.72 |
| random | 30.88 | **29.70** | 29.79 | 29.97 | 30.09 | **30.09** | **0.42** |
| reverse | 21.35 | 26.32 | **33.59** | 21.63 | 21.65 | 24.91 | 4.72 |
| insert-lead | 33.21 | 26.07 | 20.70 | **33.41** | 31.59 | 29.00 | 4.93 |
| insert-lead3 | 32.29 | 25.57 | 20.22 | 32.92 | **32.15** | 28.63 | 4.98 |

Table 2: BanditSum's performance—calculated as the average between ROUGE-1,-2, and -L F1—on the validation set of the CNN/Daily Mail corpus. The sentence position information is perturbed at different levels, as explained in Section 4.

worse when tested on a mismatched data perturbation. Even when the distortion is at a single lead position in **insert-lead** and **insert-lead3**, the performance on the original data is significantly lower than when trained without the distortion. These results corroborate Kedzie et al. (2018)'s findings for RL-based systems. Interestingly, the **random** model has the best mean performance and the lowest variation indicating that completely removing the position bias may allow a model to focus on learning robust sentence semantics.

## 5 Learning to Counter Position Bias

We present two methods which encourage models to locate key phrases at diverse parts of the article.

### 5.1 Multi-Stage Training

This technique is inspired by the robust results from the **random** model in section 4. We implement a multi-stage training method for both BanditSum and RNES where in the first few epochs, we train on an 'unbiased' dataset where the sentences in every training document are randomly shuffled. We then fine-tune the models by training on the original training articles. The goal is to prime the model to learn sentence semantics independently of position, and then introduce the task of balancing semantics and positional cues.

### 5.2 ROUGE-based Auxiliary Loss

We observed that BanditSum tends to converge to a low-entropy policy, in the sense that the model's affinity scores are either 1 or 0 at the end of training. Furthermore, over 68% of its selections are from the three leading sentences of the source. Regularizing low-entropy policies can increase a model's propensity to explore potentially good states or stay close to a known good policy (Nachum et al., 2017; Galashov et al., 2019). We extend this idea to summarization by introducing a ROUGE-based loss which regularizes the model policy using an estimate of the value of individual sentences.

These sentence-level estimates are computed as a *distribution* $P_R$:

$$P_R(x = i) = \frac{r(s_i, \mathcal{G})}{\sum_{j=1}^{n} r(s_j, \mathcal{G})}, \tag{1}$$

where $r$ is the average of ROUGE-1, -2 and -L $F_1$ scores between sentence $s_i$ in the article and the reference summary $\mathcal{G}$. We would like the model's predictive distribution $P_\mathcal{M}$ to approximately match $P_R$. To compute $P_\mathcal{M}$, we normalize the predicted scores from a non-auto-regressive model. In an auto-regressive model such as RNES, the decision of including a sentence depends on those selected so far. So a straightforward KL objective is hard to implement, and we use this technique for BanditSum only.

Our auxiliary loss is defined as the KL divergence: $\mathcal{L}_{\text{KL}} = D_{\text{KL}}(P_R \parallel P_\mathcal{M})$. The update rule then becomes:

$$\theta^{(t+1)} = \theta^{(t)} + \alpha \left( \nabla \mathcal{L}_\mathcal{M}(\theta^{(t)}) + \beta \nabla \mathcal{L}_{\text{KL}}(\theta^{(t)}) \right) \tag{2}$$

where $\theta^{(t)}$ represents the model's parameters at time step $t$, $\mathcal{L}_\mathcal{M}$ is the original model's loss function, and $\beta$ is a hyperparameter.

## 6 Experimental Setup

We use the CNN/Daily Mail dataset (Hermann et al., 2015) with the standard train/dev/test splits of 287,227/13,368/11,490. To avoid inconsistencies, we built on top of the author-provided implementations for BanditSum and our faithful reimplementation of RNES.

To reduce training time, we pre-compute and store the average of ROUGE-1, -2, and -L for every sentence triplet of each article, using a HDF5 table and PyTables (PyTables Developers Team, 2002-2019; The HDF Group, 1997-2019). This allows for a considerable increase in training speed.

| Model | ROUGE | | | Overlp |
|---|---|---|---|---|
| | 1 | 2 | L | % |
| Lead-3 | 40.06 | 17.53 | 36.18 | 100.0 |
| Oracle | 56.53 | 32.65 | 53.12 | 27.24 |
| Refresh | 40.0 | 18.2 | 36.6 | – |
| NeuSum | 40.15 | 17.80 | 36.63 | 58.24 |
| RNES | 41.15 | 18.81 | 37.75 | 68.44 |
| RNES+pretrain | 41.29 | 18.85 | 37.79 | 68.22 |
| BanditSum | 41.68 | 18.78 | 38.00 | 69.87 |
| B.Sum+pretrain | 41.68 | 18.79 | 37.99 | 70.77 |
| B.Sum+entropy | 41.71 | 18.87 | 38.04 | 64.83 |
| BanditSum+KL | **41.81*** | **18.96*** | **38.16*** | 65.13 |

Table 3: ROUGE scores for systems. 'Overlp' denotes the model's overlap in extraction choices with the lead-3 baseline. Scores significantly higher than BanditSum with $p < 0.001$ (bootstrap resampling test) are marked with *.

We limit the maximum number of sentences considered in an article to the first 100.

All the models were trained for 4 epochs. For the multi-stage training, we pretrain for 2 epochs, then train on the original articles for 2 epochs. We set the auxiliary loss hyperparameters $\alpha = 1e-4$ and $\beta = 0.0095$ in eq. 2 based on a grid search using the Tune library (Liaw et al., 2018).

We also train a baseline **entropy** model by replacing $\mathcal{L}_{\text{KL}}$ with the negated entropy of $P_{\mathcal{M}}$ in eq. 2. This loss penalizes low entropy, helping the model explore, but it is 'undirected' compared to our proposed method. We present the results of Lead-3 baseline (first 3 sentences), and two other competitive models—Refresh[2] (Narayan et al., 2018a) and NeuSum (Zhou et al., 2018).

Lastly, we include results from an *oracle* summarizer, computed as the triplet of source sentences with the highest average of ROUGE-1, -2 and -L scores against the abstractive gold standard.

## 7 Results and Discussion

Table 3 reports the F1 scores for ROUGE-1,-2 and -L (Lin, 2004). We use the *pyrouge*[3] wrapper library to evaluate the final models, while training with a faster Python-only implementation[4].

We test for significance between the baseline models and our proposed techniques using the bootstrap method. This method was first recommended for testing significance in ROUGE scores by Lin (2004), and has subsequently been advocated as an appropriate measure in works such as Dror et al. (2018) and Berg-Kirkpatrick et al. (2012).

[2]We are unable to evaluate this model on the lead overlap measure due to lack of access to the model outputs.

[3]www.github.com/bheinzerling/pyrouge

[4]www.github.com/Diego999/py-rouge

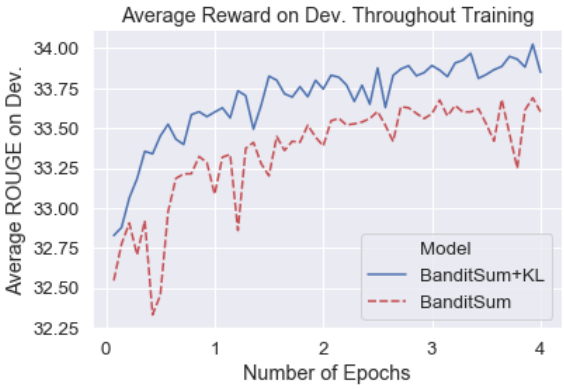


Figure 1: Training curves for BanditSum based models. Average ROUGE is the average of ROUGE-1, -2 and -L F1.

The simple entropy regularizer has a small but not significant improvement, and pretraining has a similar improvement only for RNES. But the auxiliary ROUGE loss significantly ($p < 0.001$) improves over BanditSum, obtaining an extra 0.15 ROUGE points on average. The last column reports the percentage of summary sentences which overlap with the lead. The auxiliary loss leads to a 4.7% absolute decrease in such selections compared to the base system, while also reaching a better ROUGE score. Figure 1 shows that the reward (average ROUGE-1,-2,-L) for the auxiliary loss model is consistently above the base.

We also examined the auxiliary loss model on documents where the summary is mostly comprised of lead sentences $D_{\text{early}}$, mostly sentences much later in the article $D_{\text{late}}$, and a dataset at the midway point, $D_{\text{med}}$. To create these sets, we rank test articles using the average index of its summary sentences in the source document. The 100 test articles with lowest average index are $D_{\text{early}}$, the 100 with highest value are $D_{\text{late}}$ and the 100 closest to the median are $D_{\text{med}}$. In Table 4, we can see that the auxiliary loss model's improvements are even more amplified on $D_{\text{med}}$ and $D_{\text{late}}$.

On the other hand, our pretraining results are mixed. We hope to employ more controlled multi-tasking methods (Kiperwasser and Ballesteros, 2018) in the future to deal with the issue.

The second line in Table 4 reports the oracle ROUGE scores of the best possible extractive summary. While all systems are quite close to the oracle on $D_{\text{early}}$ they only reach half the performance on $D_{\text{late}}$. This gap indicates that our improvements only scratch the surface, but also that this problem is worthy and challenging to explore.

It is worth noting that we have attempted to build a single model which can summarize both

| Model | $D_{\text{early}}$ | $D_{\text{med}}$ | $D_{\text{late}}$ |
|---|---|---|---|
| Lead-3 | 46.17 | 30.90 | 20.18 |
| Oracle | 50.52 | 47.92 | 42.21 |
| RNES | 41.76 | 32.11 | 20.62 |
| RNES+pretrain | 41.66 | 32.38 | 20.64 |
| BanditSum | 43.10 | 32.65 | 21.63 |
| BanditSum+entropy | 41.96 | 32.59 | 22.12 |
| BanditSum+KL | 42.63 | 33.05 | 21.96 |

Table 4: Average ROUGE-1, -2 and -L F1 scores on $D_{\text{early}}$, and $D_{\text{med}}$, $D_{\text{late}}$. Each set contains 100 documents.

lead-biased articles and those whose information is spread throughout. Our aim was to encourage the model to explore useful regions as a way of learning better document semantics. But we hypothesize that our models can be further improved by learning to automatically predict when the lead paragraph suffices as a summary, and when the model should look further in the document.

## 8 Conclusion

In this paper, we have presented the first approaches for learning a summarization system by countering the strong effect of summary-worthy lead sentences. We demonstrate that recent summarization systems over-exploit the inherent lead bias present in news articles, to the detriment of their summarization capabilities. We explore two techniques aimed at learning to better balance positional cues with semantic ones. While our auxiliary loss method achieves significant improvement, we note that there is a large gap which better methods can hope to bridge in the future.

One approach, building on ours, is to examine other ways to combine loss signals (Finn et al., 2017), and to encourage exploration (Haarnoja et al., 2018). We will also carry out deeper study of the properties of $D_{\text{early}}$ and $D_{\text{late}}$ type documents and use them to inform new solutions. On cursory analysis, the most frequent terms in $D_{\text{early}}$ tend to be about UK politics, while in $D_{\text{late}}$ they are often related to British soccer.

## Acknowledgments

This work is supported by the Natural Sciences and Engineering Research Council of Canada, the Institute for Data Valorisation (IVADO), Compute Canada, and the CIFAR Canada AI Chair program.

## References

Taylor Berg-Kirkpatrick, David Burkett, and Dan Klein. 2012. An empirical investigation of statistical significance in NLP. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 995–1005, Jeju Island, Korea. Association for Computational Linguistics.

Ronald Brandow, Karl Mitze, and Lisa F Rau. 1995. Automatic condensation of electronic publications by sentence selection. *Information Processing & Management*, 31(5):675–685.

Ziqiang Cao, Furu Wei, Sujian Li, Wenjie Li, Ming Zhou, and Houfeng Wang. 2015. Learning summary prior representation for extractive summarization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning*.

Yue Dong, Yikang Shen, Eric Crawford, Herke van Hoof, and Jackie Chi Kit Cheung. 2018. Banditsum: Extractive summarization as a contextual bandit. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3739–3748.

Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. 2018. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1383–1392, Melbourne, Australia. Association for Computational Linguistics.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135, International Convention Centre, Sydney, Australia. PMLR.

Alexandre Galashov, Siddhant M Jayakumar, Leonard Hasenclever, Dhruva Tirumala, Jonathan Schwarz, Guillaume Desjardins, Wojciech M Czarnecki, Yee Whye Teh, Razvan Pascanu, and Nicolas Heess. 2019. Information asymmetry in kl-regularized rl. *International Conference on Learning Representations (ICLR)*.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with

a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1861–1870, Stockholmsmssan, Stockholm Sweden. PMLR.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Kai Hong and Ani Nenkova. 2014. Improving the estimation of word importance for news multi-document summarization. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 712–721.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Chris Kedzie, Kathleen McKeown, and Hal Daume III. 2018. Content selection in deep learning models of summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1818–1828.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Eliyahu Kiperwasser and Miguel Ballesteros. 2018. Scheduled multi-task learning: From syntax to translation. *Transactions of the Association of Computational Linguistics*, 6:225–240.

Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E Gonzalez, and Ion Stoica. 2018. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*.

Yang Liu. 2019. Fine-tune BERT for extractive summarization. *CoRR*, abs/1903.10318.

Ofir Nachum, Mohammad Norouzi, and Dale Schuurmans. 2017. Improving policy gradient by exploring under-appreciated rewards. In *International Conference on Learning Representations*, pages 2775–2785.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. SummaRuNNer: A recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gùlçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL)*.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018a. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018b. Ranking sentences for extractive summarization with reinforcement learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*.

Ani Nenkova. 2005. Automatic text summarization of newswire: Lessons learned from the document understanding conference. In *AAAI*, volume 5, pages 1436–1441.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations (ICLR)*.

PyTables Developers Team. 2002-2019. PyTables: Hierarchical datasets in Python.

Barry Schiffman, Ani Nenkova, and Kathleen McKeown. 2002. Experiments in multidocument summarization. In *Proceedings of the Second International Conference on Human Language Technology Research*, pages 52–58.

The HDF Group. 1997-2019. Hierarchical Data Format, version 5. Http://www.hdfgroup.org/HDF5/.

Yuxiang Wu and Baotian Hu. 2018. Learning to extract coherent summary via deep reinforcement learning. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*.

Wenpeng Yin and Yulong Pei. 2015. Optimizing sentence modeling and selection for document summarization. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1383–1389.

Qingyu Zhou, Nan Yang, Furu Wei, Shaohan Huang, Ming Zhou, and Tiejun Zhao. 2018. Neural document summarization by jointly learning to score and select sentences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 654–663, Melbourne, Australia. Association for Computational Linguistics.

# Learning Rhyming Constraints using Structured Adversaries

**Harsh Jhamtani** [1] **Sanket Vaibhav Mehta** [1] **Jaime Carbonell** [1] **Taylor Berg-Kirkpatrick** [2]
[1] School of Computer Science, Carnegie Mellon University
[2] Department of Computer Science and Engineering, University of California San Diego
{jharsh,svmehta,jgc}@cs.cmu.edu, tberg@ucsd.eng.edu

## Abstract

Existing recurrent neural language models often fail to capture higher-level structure present in text: for example, rhyming patterns present in poetry. Much prior work on poetry generation uses manually defined constraints which are satisfied during decoding using either specialized decoding procedures or rejection sampling. The rhyming constraints themselves are typically not learned by the generator. We propose an alternate approach that uses a structured discriminator to learn a poetry generator that directly captures rhyming constraints in a generative adversarial setup. By causing the discriminator to compare poems based only on a learned similarity matrix of pairs of line ending words, the proposed approach is able to successfully learn rhyming patterns in two different English poetry datasets (Sonnet and Limerick) without explicitly being provided with any phonetic information.

## 1 Introduction

Many existing approaches to text generation rely on recurrent neural networks trained using likelihood on sequences of words or characters. However, such models often fail to capture overall structure and coherency in multi-sentence or long-form text (Bosselut et al., 2018; Holtzman et al., 2018). To rectify this, prior work has proposed losses which encourage overall coherency or other desired behavior (Li et al., 2016; Zhang and Lapata, 2017; Bosselut et al., 2018). However, most of these approaches rely on manually provided definitions of what constitutes a good or suitable structure, thereby limiting their applicability. In this paper we propose a method for English poetry generation that directly learns higher-level rhyming constraints as part of a generator without requiring strong manual intervention. Prior works on poetry generation (Oliveira, 2017; Ghazvininejad et al., 2018) have focused mostly on ad-hoc decoding procedures to generate reasonable poetry, often relying on pruning from a set of candidate outputs to encourage desired behavior such as presence of explicitly-defined rhyming patterns.

We propose an adversarial approach to poetry generation that, by adding structure and inductive bias into the discriminator, is able to learn rhyming constraints directly from data without prior knowledge. The role of the discriminator is to try to distinguish between generated and real poems during training. We propose to add inductive bias via the choice of discriminator architecture: We require the discriminator to reason about poems through pairwise comparisons between line ending words. These learned word comparisons form a similarity matrix for the poem within the discriminator's architecture. Finally, the discriminator evaluates the poem through a 2D convolutional classifier applied directly to this matrix. This final convolution is naturally biased to identify spatial patterns across word comparisons, which, in turn, biases learned word comparisons to pick up rhyming since rhymes are typically the most salient spatial patterns.

Recent work by Lau et al. (2018) proposes a quatrain generation method that relies on specific domain knowledge about the dataset to train a classifier for learning the notion of rhyming: that a line ending word always rhymes with exactly one more ending word in the poem. This limits the applicability of their method to other forms of poetry with different rhyming patterns. They train the classifier along with a language model in a multi-task setup. Further, at generation time, they heavily rely on rejection sampling to produce quatrains which satisfy any valid rhyming pattern. In contrast, we find that generators trained using our structured adversary produce samples that satisfy rhyming constraints with much higher frequency.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6025–6031, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

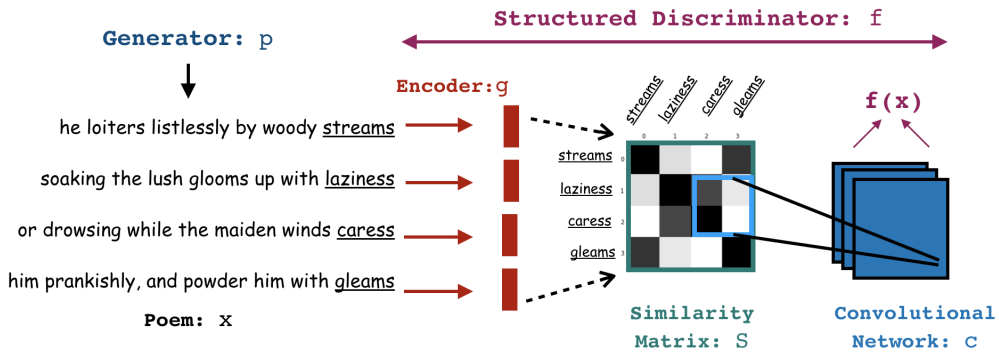


Figure 1: Model Overview: We propose a structured discriminator to learn a poetry generator in a generative adversarial setup. Similarities between pairs of end-of-line words are obtained by computing cosine similarity between their corresponding representations, produced by a learned character-level LSTM encoder. The discriminator operates on the resulting matrix $S$ representing pair-wise similarities of end words. The proposed discriminator learns to identify rhyming word pairs as well as rhyming constraints present in the dataset without being provided phonetic information in advance.

Our main contributions are as follows: We introduce a novel structured discriminator to learn a poetry generation model in a generative adversarial setup. We show that the discriminator induces an accurate rhyming metric and the generator learns natural rhyming patterns without being provided with phonetic information. We successfully demonstrate the applicability of our proposed approach on two datasets with different structural rhyming constraints. Our poem generation model learned with the structured discriminator is more sampling efficient compared to prior work – many fewer generation attempts are required in order to obtain an valid sample which obeys the rhyming constraints of the corresponding poetry dataset.

## 2 Method

Many forms of poetry make use of rhyming patterns on line-ending words (Oliveira, 2017). Therefore, to characterize a rhyming poem, a model needs (1) to know what it means to rhyme (2) to identify the specific permissible rhyming patterns for a particular poem type. For example, a limerick is a 5 line poem with a rhyming constraint of the type AABBA, i.e. the ends of the first, second, and fifth lines rhyme. We consider an adversarial learning setup with a hierarchical language model and a structured discriminator, such that the discriminator is trained to distinguish between generated examples and training examples, and the generator is trained to *fool* the discriminator. Our novel structured discriminator operates on a matrix which encodes a *learned* pair-wise similarity function of the line ending words. We refer to our model as **RHYME-GAN**.

### 2.1 Neural Generation Model

Our generator is a hierarchical neural language model (Figure 1) that first generates a sequence of line-ending words, and thereafter generates the poem's lines conditioned on the ending words. We use recurrent neural networks for ending word generation as well line generation conditioned on ending words. Following prior work (Lau et al., 2018), we generate words in each line in reverse order (i.e. right to left), and begin generation with the last line first. Let $\hat{x}$ represent a sample from the current generator distribution, denoted by $p_\theta$, where $\theta$ represents the generator parameters. We initialize the word embeddings in the generator with pre-trained word embeddings (Lau et al., 2018) trained on a separate non-sonnet corpus.

### 2.2 Structured Discriminator

We introduce a structured discriminator, denoted by function $f_\phi(x)$, which outputs the probability that $x$ is a sample from the dataset as opposed to generated. Our architecture defines an intermediate matrix $S \in R^{T \times T}$, where $T$ denotes the number of lines in the poem, which encodes pair-wise similarities between line ending words in order to capture rhyming structure. The discriminator's output is determined by a two layer 2D convolutional neural network applied to $S$. Convolutional neural networks have been shown to capture local as well as global patterns in 2D data – for example, images. Thus, our discriminator is composed of two main components: computation of a ma-

trix $S$, and a convolutional neural network to classify the computed matrix $S$. The pair-wise computation provides a useful inductive bias to identify the notion of rhyming, whereas the convolutional network is a suitable choice to capture overall rhyming patterns.

More specifically, let the words at the ends of lines in $x$ be denoted by $e$. The number of ending words will be same as the number of lines in $x$, which we denote as $T$. We encode each ending word using a character-level LSTM (Hochreiter and Schmidhuber, 1997) denoted by $g_{\phi_g}$, and use the last hidden state of the LSTM as a vector representation of the word. We let $S_{ij}$ be the cosine similarity between the representations of ending words $e_i$, $e_j$, given by following equation:

$$S_{ij} = \frac{g(e_i)g(e_j)}{|g(e_i)||g(e_j)|} \quad (1)$$

The matrix $S$ is passed through a convolutional neural network composed with a linear layer, together denoted by $c_{\phi_c}$. The final output is passed through a sigmoid non-linearity, so that $f_\phi(x) \in [0, 1]$. The value of $f_\phi(x)$ represents the discriminator's assessment of the probability that datum $x$ belongs to the *real* dataset, rather than being a generated sample. The discriminator's objective will train it to distinguish between a sample $x$ from training data $\mathcal{X}$, and a generated sample $\hat{x}$, in a binary classification setup. Specifically, we define the discriminator loss for $x, \hat{x}$ as follows:

$$d(x, \hat{x}; \phi) = -\log(f_\phi(x)) - \log(1 - f_\phi(\hat{x})) \quad (2)$$

### 2.3 Learning

Generator parameters $\theta$ and discriminator parameters $\phi$ are trained together under following objective:

$$\min_\theta \left[ \mathbb{E}_{x \in \mathcal{X}} \left[ -\log p_\theta(x) + \lambda \max_\phi \mathbb{E}_{\hat{x} \sim p_\theta}[-d(x, \hat{x})] \right] \right] \quad (3)$$

Note, in addition to using a traditional adversarial objective, we also include a likelihood term to help stabilize the generator. $\lambda$ is a hyperparameter which controls the relative weight of the two terms. Since sampling of $\hat{x}$ from generator involves discrete choices, we use the REINFORCE (Williams, 1992) algorithm to train the generator using learning signal from the adversarial loss term. The generator simultaneously gets an exact gradient from the likelihood portion of the objective. We observe training is more stable when we pretrain the LSTM word encoder $g_{\phi_g}(.)$ part of the discriminator, along with a separate LSTM decoder, using an auto-encoding objective on words in the vocabulary.

| Model | Expected #Samples | |
|---|---|---|
| | SONNET | LIMERICK |
| DEEP-SPEARE | 153.8 | N/A |
| RHYME-LM | 169.5 | 500 |
| RHYME-GAN-NS | 4.8 | 26.6 |
| **RHYME-GAN** | **3.7** | **4.7** |

Table 1: Sampling efficiency: We obtain 10K samples of poetry without additional intervention during decoding, and report the expected samples as inverse of the fraction of samples satisfying valid rhyming patterns for the corresponding dataset. Lower values are better.

## 3 Experiments and Results

### 3.1 Datasets

We work with the Shakespeare SONNET dataset (Lau et al., 2018) and a new LIMERICK corpus. Each sonnet in the Sonnet dataset is made up of 3 quatrains of 4 lines each, and a couplet. The dataset consists of 2685 sonnets in train, and 335 each in validation and test splits. The quatrains typically have one of the following rhyming structures: AABB, ABAB, ABBA, though some deviations are observed in the dataset. This may be because rhyming patterns are not always strictly followed in writing quatrains, and there are possible inaccuracies in the word pronunciation dictionaries used (e.g. some words can have multiple different pronunciations based on context).

A limerick is a form of verse with five lines. Limericks typically follow a rhyming pattern of AABBA. We collect limericks from an online collection[1]. Due to a large vocabulary in the full collection, we filter the dataset to retain only those limericks whose all the words are in a subset of 9K most frequent words. Our final dataset consists of $10,400$ limericks in train and 1300 each in validation and test splits. We train and evaluate the models separately on each corpus.

### 3.2 Poem Generator

**Sampling efficiency** We compute the expected number of samples needed before we sample a quatrain which satisfies one of the hand-defined rhyming patterns. Towards this end, we get 10K samples from each model without any constraints (except avoiding *UNK* - unknown tokens). Fol-

[1] http://hardsoft.us. Accessed May 2019.

lowing prior work (Lau et al., 2018), words are sampled with a temperature value between 0.6 and 0.8. We use CMU dictionary (Weide, 1998) to look up the phonetic representation of a word, and extract the sequence of phonemes from the last stressed syllable onward. Two words are considered to be rhyming if their extracted sequences match (Parrish, 2015). We consider a generated quatrain to have an acceptable pattern if the four ending words follow one of the three rhyming patterns: AABB, ABBA, ABAB. Similarly for LIMERICK, we consider only those samples to be acceptable which have line endings of the rhyming form AABBA.

We consider a baseline **RHYME-LM**, which has the same generator architecture as RHYME-GAN but is trained without the discriminator. We also compare with **RHYME-GAN-NS** which uses a simpler non-structured discriminator. Specifically, it uses a discriminator which first runs a character-level encoder for each ending word - similar to RHYME-GAN - but then instead of computing pairwise similarity matrix, it utilizes a LSTM on the sequence of the computed representations.

As can be observed from Table 1, RHYME-GAN needs fewer samples than other methods to produce an acceptable quatrain or a limerick, indicating that it has learned natural rhyming structures more effectively from data. Note we do not report DEEP-SPEARE on Limerick due to their SONNET specific assumption that for a given end-of-line word there is exactly one more rhyming word among other end-of-line words. Additionally, RHYME-GAN-NS performs worse than RHYME-GAN, and the difference in performance is more prominent in LIMERICK – demonstrating that the proposed structure in the discriminator provided useful inductive bias. Note that compared to 4 line quatrains in SONNET, LIMERICK has 5 line poems and has arguably more complex rhyming pattern constraints.

**Likelihood on held out data** We report negative log likelihood (NLL) on test splits (Table 2). For SONNET, RHYME-GAN achieves a test set NLL of 3.98. Our model without adversarial learning i.e. RHYME-LM, achieves a test set NLL of 3.97. DEEP-SPEARE reports a test set NLL of 4.38. Note that our language model is hierarchical while DEEP-SPEARE has a linear model. The NLL for RHYME-LM and RHYME-GAN are very similar, though RHYME-GAN gets much better sampling efficiency scores than RHYME-LM.

| Model | NLL | |
|---|---|---|
| | SONNET | LIMERICK |
| DEEP-SPEARE | 4.38 | N/A |
| **RHYME-LM** | **3.97** | **3.48** |
| RHYME-GAN | 3.98 | 3.49 |

Table 2: Held out negative log likelihood per token for poems in test split.

Our generator implementation is largely based on that of Lau et al. (2018). The main difference is that we first generate all the line-ending words and then condition on them to generate the remaining words. The change was made to make it more amenable to our proposed discriminator. However, our hierarchical language model (RHYME-LM) performs worse than DEEP-SPEARE as per sampling efficiency. Therefore, structured discriminator is the driving factor behind the observed improvement with RHYME-GAN. However, committing to the ending words of all lines before completing preceding lines can be a limitation, and addressing it is a possible future direction.

### 3.3 Analyzing Learned Discriminator

We probe the the word representations $g(.)$ to check if rhyming words are close-by in the learned manifold. We consider all pairs of words among the ending words in a quatrain/limerick, and label each pair to be rhyming or non-rhyming based on previously stated definition of rhyming. If the cosine similarity score between the representations of pairs of words is above a certain threshold, we predict that word pair as rhyming, else it is predicted as non-rhyming. We report F1 scores for the binary classification setup of predicting word-pairs to be rhyming or not. We consider some additional baselines: **RHYM-EM** (Reddy and Knight, 2011) uses latent variables to model rhyming schemes, and train parameters using EM. **GRAPHEME-K** baselines predict a word pair as rhyming only if the last $K = \{1, 2, 3\}$ characters of the two words are same.

For SONNET data, we observe that RHYME-GAN obtains a F1 score of 0.90 (Table 3) on the test split (threshold chosen to maximize f1 on dev split). We repeat the above analysis on the LIMERICK dataset and observe an F1 of 0.92 for RHYME-GAN. DEEP-SPEARE model reports F1

| Model | SONNET | LIMERICK |
|---|---|---|
| GRAPHEME-1 | 0.71 | 0.79 |
| GRAPHEME-2 | 0.78 | 0.79 |
| GRAPHEME-3 | 0.69 | 0.67 |
| RHYM-EM | 0.71 | 0.73 |
| DEEP-SPEARE/MAX-MARGIN | 0.91 | 0.81 |
| RHYME-GAN-NS | 0.85 | 0.87 |
| **RHYME-GAN** | **0.90** | **0.92** |

Table 3: Rhyming probe: We use the cosine similarity score of the learned representations to predict a word pair as rhyming or not, and report F1 score for this classification task. RHYM-EM is an unsupervised rhyming pattern discovery method. GRAPHEME-K baselines predict based on exact match of last $k$ characters.

of 0.91 on SONNET. As stated earlier, DEEP-SPEARE'S model is not amenable to LIMERICK - we do compare though with the max-margin classifier in DEEP-SPEARE model trained on LIMERICK which gets F1 score of 0.81. The scores are understandably lower since the AABBA pattern in limericks is not amenable to SONNET specific assumptions made in DEEP-SPEARE model. On the other hand, RHYME-GAN achieves high F1 scores for both the datasets without incorporating any domain specific rhyming pattern information.

RHYME-GAN performs much better than RHYM-EM and GRAPHEME-K baselines. RHYM-EM does not perform well - probably because it operates at word-level and fails to generalize. Note that **RHYME-GAN-NS** gets F1 score of 0.85 in case of SONNET dataset and 0.87 for LIMERICK. These values are lower than corresponding scores for RHYME-GAN, demonstrating that the proposed structure in the discriminator was useful in learning the notion of rhyming.

### 3.4 Human Evaluations

Following prior work (Lau et al., 2018), we requested human annotators to identify the human-written poem when presented with two samples at a time - a quatrain from the Sonnet corpus and a machine-generated quatrain, and report the annotator accuracy on this task. Note that a lower accuracy value is favorable as it signifies higher quality of machine-generated samples. Using 150 valid samples (i.e. samples belonging to one of the allowed rhyming patterns), we observe 56.0% annotator accuracy for RHYME-GAN, and 53.3% for DEEP-SPEARE – indicating that the post-rejection sampling outputs from the two methods are of comparable quality (the difference in annotator accuracy is not statistically significant as per McNemar's test under $p < 0.05$). If we use pre-rejection samples, we observe 60.0% annotator accuracy for RHYME-GAN, and 81.3% for DEEP-SPEARE (the difference being statistically significant as per McNemar's test under $p < 0.05$) – indicating that unfiltered samples from RHYME-GAN are of higher quality compared to DEEP-SPEARE.

## 4 Related Work

Early works on poetry generation mostly used rule based methods (Gervás, 2000; Wu et al., 2009; Oliveira, 2017). More recently, neural models for poetry generation have been proposed (Zhang and Lapata, 2014; Ghazvininejad et al., 2016, 2017; Hopkins and Kiela, 2017; Lau et al., 2018; Liu et al., 2019). Yan et al. (2013) retrieve high ranking sentences for a given user query, and repeatedly swap words to satisfy poetry constraints. Ghazvininejad et al. (2018) worked on poetry translation using an unconstrained machine translation model and separately learned Finite State Automata for enforcing rhythm and rhyme. Similar to rhyming and rhythm patterns in poetry, certain types of musical compositions showcase rhythm and repetition patterns, and some prior works model such patterns in music generation (Walder and Kim, 2018; Jhamtani and Berg-Kirkpatrick, 2019). Generative adversarial learning (Goodfellow et al., 2014) for text generation has been used in prior works (Fedus et al., 2018; Wang et al., 2018, 2019; Rao and Daumé III, 2019), though has not been explored with regard to the similarity structure proposed in this paper.

## 5 Conclusions

In this paper we have proposed a novel structured discriminator to learn a poem generator. The generator learned utilizing the structured adversary is able to identify rhyming structure patterns present in data, as demonstrated through the improved sampling efficiency. Through the rhyming classification probe, we demonstrate that the proposed discriminator is better at learning the notion of rhyming compared to baselines.

## Acknowledgements

We are thankful to anonymous EMNLP conference reviewers for providing valuable feedback. We thank Graham Neubig for useful discussions on Limerick generation. This project is funded in part by the NSF under grant 1618044 and by the NEH under grant HAA-256044-17.

## References


Antoine Bosselut, Asli Celikyilmaz, Xiaodong He, Jianfeng Gao, Po-Sen Huang, and Yejin Choi. 2018. Discourse-aware neural rewards for coherent text generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 173–184.

William Fedus, Ian Goodfellow, and Andrew M Dai. 2018. Maskgan: Better text generation via filling in the_. *arXiv preprint arXiv:1801.07736*.

Pablo Gervás. 2000. Wasp: Evaluation of different strategies for the automatic generation of spanish verse. In *Proceedings of the AISB-00 symposium on creative & cultural aspects of AI*, pages 93–100.

Marjan Ghazvininejad, Yejin Choi, and Kevin Knight. 2018. Neural poetry translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 67–71.

Marjan Ghazvininejad, Xing Shi, Yejin Choi, and Kevin Knight. 2016. Generating topical poetry. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1191.

Marjan Ghazvininejad, Xing Shi, Jay Priyadarshi, and Kevin Knight. 2017. Hafez: an interactive poetry generation system. *Proceedings of ACL 2017, System Demonstrations*, pages 43–48.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Ari Holtzman, Jan Buys, Maxwell Forbes, Antoine Bosselut, David Golub, and Yejin Choi. 2018. Learning to write with cooperative discriminators. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1638–1649.

Jack Hopkins and Douwe Kiela. 2017. Automatically generating rhythmic verse with neural networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 168–178.

Harsh Jhamtani and Taylor Berg-Kirkpatrick. 2019. Modeling self-repetition in music generation using generative adversarial networks. *Machine Learning for Music Discovery Workshop, ICML 2019*.

Jey Han Lau, Trevor Cohn, Timothy Baldwin, Julian Brooke, and Adam Hammond. 2018. Deep-speare: A joint neural model of poetic language, meter and rhyme. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1948–1958.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202.

Zhiqiang Liu, Zuohui Fu, Jie Cao, Gerard de Melo, Yik-Cheung Tam, Cheng Niu, and Jie Zhou. 2019. Rhetorically controlled encoder-decoder for modern chinese poetry generation. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 1992–2001.

Hugo Gonçalo Oliveira. 2017. A survey on intelligent poetry generation: Languages, features, techniques, reutilisation and evaluation. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 11–20.

Allison Parrish. 2015. Pronouncing: Interface for the cmu pronouncing dictionary. [Online; accessed May 2019].

Sudha Rao and Hal Daumé III. 2019. Answer-based adversarial training for generating clarification questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 143–155.

Sravana Reddy and Kevin Knight. 2011. Unsupervised discovery of rhyme schemes. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 77–82.

Christian Walder and Dongwoo Kim. 2018. Neural dynamic programming for musical self similarity. In *International Conference on Machine Learning*, pages 5092–5100.

William Yang Wang, Sameer Singh, and Jiwei Li. 2019. Deep adversarial learning for nlp. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*, pages 1–5.

Xin Wang, Wenhu Chen, Yuan-Fang Wang, and William Yang Wang. 2018. No metrics are perfect: Adversarial reward learning for visual storytelling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 899–909.

R Weide. 1998. The CMU pronunciation dictionary, release 0.6. *Carnegie Mellon University*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Xiaofeng Wu, Naoko Tosa, and Ryohei Nakatsu. 2009. New hitch haiku: An interactive renku poem composition supporting tool applied for sightseeing navigation system. In *International Conference on Entertainment Computing*, pages 191–196. Springer.

Rui Yan, Han Jiang, Mirella Lapata, Shou-De Lin, Xueqiang Lv, and Xiaoming Li. 2013. i, poet: Automatic chinese poetry composition through a generative summarization framework under constrained optimization. In *IJCAI*, pages 2197–2203.

Xingxing Zhang and Mirella Lapata. 2014. Chinese poetry generation with recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 670–680.

Xingxing Zhang and Mirella Lapata. 2017. Sentence simplification with deep reinforcement learning. *arXiv preprint arXiv:1703.10931*.

# Question-type Driven Question Generation

**Wenjie Zhou, Minghua Zhang, Yunfang Wu***
Key Laboratory of Computational Linguistics, Ministry of Education
School of Electronics Engineering and Computer Science, Peking University, Beijing, China
{wjzhou013, zhangmh, wuyf}@pku.edu.cn

## Abstract

Question generation is a challenging task which aims to ask a question based on an answer and relevant context. The existing works suffer from the mismatching between question type and answer, i.e. generating a question with type *how* while the answer is a personal name. We propose to automatically predict the question type based on the input answer and context. Then, the question type is fused into a seq2seq model to guide the question generation, so as to deal with the mismatching problem. We achieve significant improvement on the accuracy of question type prediction and finally obtain state-of-the-art results for question generation on both SQuAD and MARCO datasets.

## 1 Introduction

Question generation (QG) can be effectively applied to many fields, including question answering (Duan et al., 2017), dialogue system (Shum et al., 2018) and education. In this paper, we focus on the answer-aware QG, which is to generate a question according to the given sentence and the expected answer.

Recently, the neural-based approaches on QG have achieved remarkable success, by applying large-scale reading comprehension datasets and employing the encoder-decoder framework. Most of the existing works are based on the seq2seq network incorporating attention mechanism and copy mode, which are first applied in Zhou et al. (2017). Later, Song et al. (2018) leverage multi-perspective matching methods, and Sun et al. (2018) propose a position-aware model to put more emphasis on answer-surrounded context words. Both works are trying to enhance the relevance between the question and answer. Zhao et al. (2018) aggregate paragraph-level context to provide sufficient information for question generation. Another direction is to integrate question answering and question generation as dual tasks (Tang et al., 2017). Beyond answer-aware QG, some systems try to generate questions from a text without answer as input (Du and Cardie, 2017; Subramanian et al., 2018).

Despite the progress achieved by the previous work, we found that the types of generated questions are often incorrect. According to experiments on SQuAD, one strong model (Zhou et al., 2017) we replicate only obtains 57.6% accuracy in question type. As we know, question type is vital for question generation, since it determines the question pattern and guides the generating process. If the question type is incorrect, the remained generated sequence would drift far away.

Several works have addressed this issue. Sun et al. (2018) incorporated a question word generation mode to generate question word at each decoding step, which utilized the answer information by employing the encoder hidden states at the answer start position. However, their method did not consider the structure and lexical features of answer. Meanwhile, the way they utilize the question word is not as effective as ours. Hu et al. (2018) proposed a model to generate question based on the given question type and aspect, and their work verifies the effect of question word but fails in the conventional QG task which does not give a question type, since their experimental results show poor performance when trying to generate question types automatically. Our work solves this problem as Section 3.3 shows. Wang et al. (2018) devised a type decoder which combines three type-specific generation distribution (including question type) with weighted sum. However, the results displayed in their paper show that questions in dialogue are far different from questions for reading comprehension, which indicates a gap

*Corresponding author.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6032–6037,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

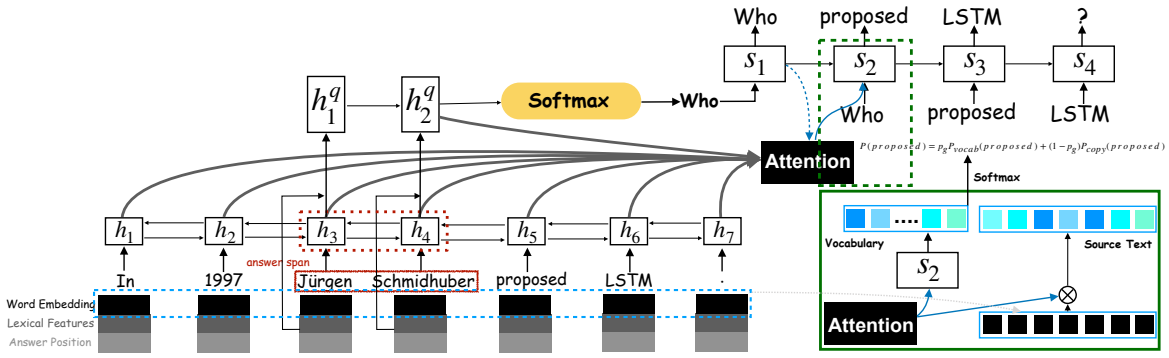


Figure 1: Structure of our unified model

between two tasks.

In this paper, we propose a unified model to predict the question type and to generate questions simultaneously. We conduct experiments on two reading comprehension datasets: SQuAD and MARCO, and obtain promising results. As for the auxiliary task, our unified model boosts the accuracy of question type prediction significantly, by 16.79% on SQuAD and 3.5% on MARCO. For question generation, our model achieves new state-of-the-art results on both datasets, with BLEU-4 16.31 on SQuAD and 21.59 on MARCO.

## 2 Model Description

The structure of our model is shown in Figure 1. A feature-rich encoder is used to encode the input sentence and corresponding answer that is a span of the sentence. Besides, the answer hidden states are used to predict the type of target question. This prediction will further be used to guide QG with a unified attention-based decoder.

### 2.1 Feature-rich Encoder

Follow Zhou et al. (2017), we exploit lexical features to enrich the encoder, where features are composed of POS tags, NER tags, and word case. We concatenate the word embedding $e_t$, answer position embedding $a_t$ and lexical features embedding $l_t$ as input $((x_1, ..., x_T), x_t = [e_t; a_t; l_t])$. Then a bidirectional LSTM is used to produce a sequence of hidden states $(h_1, ..., h_T)$.

### 2.2 Question Type Prediction

Since different types of questions are various in syntax and semantics, it is essential to predict an accurate type for generating a reasonable question. According to the statistics on SQuAD, 78% of questions in the training set begin with the 7 most common used question words as Table 1 shows. So we divide question types into 8 categories, including 7 question words and an additional type 'others'. It shows that the data distribution on different types is quite unbalanced, suggesting it is a hard task to predict the correct question type.

We use an unidirectional LSTM network to predict the expected question type. Assuming *m+1, ..., m+a* is the index of given answer span in the input sentence, based on the corresponding feature-rich hidden states $(h_{m+1}, ..., h_{m+a})$, we calculate the question type hidden states as follow:

$$h_j^q = LSTM^q([h_{m+j}; l_{m+j}], h_{j-1}^q), j \in [1, a] \quad (1)$$

$$h_0^q = h_T \quad (2)$$

where $l_{m+j}$ is the corresponding lexical features. Besides, to make full use of the feature-rich encoder hidden states, we take the last hidden state as the initial hidden state of $LSTM^q$.

Then, the last output hidden state $h_a^q$ are fed into a softmax layer to obtain the type distribution:

$$P(Q_w) = softmax(W_q h_a^q) \quad (3)$$

$$E^q = -log(P(Q_w^*)) \quad (4)$$

$E^q$ is the loss of question type prediction, $Q_w^*$ is the target type.

### 2.3 Unified Attention-based Decoder

The conventional attention-based decoder adopts a <BOS> token as the first input word at step 1, while we replace it with the predicted question word $Q_w$ to guide the generation process. At step $i$, we calculate the decoder hidden state as follow:

$$s_i = LSTM([w_{i-1}; c_{i-1}], s_{i-1}) \quad (5)$$

Further, to fuse the type information in every step of question generation, our model takes $h_a^q$ into consideration while calculating the context

| Type | What | Who | How | When | Which | Where | Why | Others |
|---|---|---|---|---|---|---|---|---|
| **SQuAD** | 43.26% | 9.39% | 9.12% | 6.26% | 4.78% | 3.76% | 1.37% | 21.83% |
| **MARCO** | 44.29% | 1.47% | 16.28% | 1.52% | 1.11% | 4.16% | 1.88% | 29.29% |

Table 1: Proportions of each type of questions on two datasets.

vector, i.e. the context vector is conditioned on $(h_1, ..., h_T, h_{T+1})$, where $h_{T+1} = h_a^q$.

$$c_i = \sum_{t=1}^{T+1} \alpha_{it} h_t \tag{6}$$

where $\alpha_{it}$ is calculated via attention mechanism (Bahdanau et al., 2014).

Then, $s_i$ together with $c_i$ will be fed into a two-layer feed-forward network to produce the vocabulary distribution $P_{vocab}$.

Following (See et al., 2017), the copy mode is used to duplicate words from the source via pointing:

$$P(w_i) = p_g P_{vocab}(w_i) + (1 - p_g) P_{copy}(w_i) \tag{7}$$

where $P_{copy}$ is the distribution of copy mode, $p_g \in [0, 1]$ is a gate to dynamically assign weights.

The loss at step $i$ is the negative log-likelihood of the target word $w_i^*$. To obtain a combined training objective of two tasks, we add the loss of question type prediction into the loss of question generation to form a total loss function:

$$E^{total} = \frac{1}{K} \sum_{i=1}^{K} -logP(w_i^*) + E^q \tag{8}$$

# 3 Experiment

## 3.1 Experiment Settings

**Dataset** Following the previous works, we conduct experiments on two datasets, SQuAD and MARCO. We use the data released by Zhou et al. (2017) and Sun et al. (2018), there are 86,635, 8,965 and 8,964 sentence-answer-question triples in the training, development and test set for SQuAD, and 74,097, 4,539 and 4,539 sentence-answer-question triples in the training, development and test set for MARCO, respectively. Lexical features are extracted using Stanford CoreNLP.

**Implementation Details** Our vocabulary is set to contain the most frequent 20,000 words in each training set. Word embeddings are initialized with the pre-trained 300-dimensional Glove vectors, and they are allowed to be fine-tuned during training. The representations of answer position, POS tags, NER tags, and word case are randomly initialized as 32-dimensional vectors, respectively. The feature-rich encoder consists of 2 layers BiLSTM, and the hidden size of all encoders and decoder is set to 512. The cutoff length of the input sequences is set to 10. In testing, we used beam search with a beam size of 12. The development set is used to search the best checkpoint. In order to decrease the volatility of the training procedure, we then average the nearest 5 checkpoints to obtain a single averaged model.

Following the existing work, we use BLUE (Papineni et al., 2002) as the metrics for automatic evaluation, with BLUE-4 as the main metric.

## 3.2 Upper Bound Analysis

To study the effectiveness of the question type for QG, we make an upper bound analysis. The experimental results are shown in Table 2.

First, we feed the decoder with the original first word of questions, regardless whether the first word is a question word or not. As shown in Table 2, comparing with the baseline model, the performance gets 3.41 and 3.32 points increment on SQuAD and MARCO, respectively, which is a large margin.

Since the beginning words which are not question words compose a large vocabulary while the number of each word is few, it is irrational to train a classifier to predict all types of these beginning words. Therefore, we reduce the question type vocabulary to only question words. In details, if the start of a targeted question is a question word, the corresponding question word will replace the <BOS> to feed into decoder, otherwise we just use the original <BOS> without any replacement. This experiment still gains a lot, as shown in Table 2 with "given the quetion type".

The above experiments verify the magnitude of using the proper question type to guide the generation process, suggesting us a promising direction for QG.

| Dataset / Model | SQuAD BLEU-1 | SQuAD BLEU-2 | SQuAD BLEU-3 | SQuAD BLEU-4 | MARCO BLEU-1 | MARCO BLEU-2 | MARCO BLEU-3 | MARCO BLEU-4 |
|---|---|---|---|---|---|---|---|---|
| feature-rich pointer generator (baseline) | 41.25 | 26.76 | 19.53 | 14.89 | 54.04 | 36.68 | 26.62 | 20.15 |
| given the first word of question | 47.00 | 31.82 | 23.69 | 18.30 | 59.51 | 41.42 | 30.70 | 23.47 |
| given the question type | 45.56 | 30.53 | 22.61 | 17.38 | 57.39 | 39.60 | 29.29 | 22.52 |

Table 2: Upper bound analysis by incorporating question words with different ways.

| Dataset / Model | SQuAD BLEU-1 | SQuAD BLEU-2 | SQuAD BLEU-3 | SQuAD BLEU-4 | MARCO BLEU-1 | MARCO BLEU-2 | MARCO BLEU-3 | MARCO BLEU-4 |
|---|---|---|---|---|---|---|---|---|
| NQG++ (Zhou et al., 2017) | - | - | - | 13.29 | - | - | - | - |
| question word generate model (Sun et al., 2018) | 42.10 | 27.52 | 20.14 | 15.36 | 46.59 | 33.46 | 24.57 | 18.73 |
| Hybrid model (Sun et al., 2018) | 43.02 | 28.14 | 20.51 | 15.64 | 48.24 | 35.95 | 25.79 | 19.45 |
| Maxout Pointer (sentence) (Zhao et al., 2018) | **44.51** | 29.07 | 21.06 | 15.82 | - | - | - | 16.02 |
| **Our Model** | | | | | | | | |
| Feature-rich pointer generator (Baseline) | 41.25 | 26.76 | 19.53 | 14.89 | 54.04 | 36.68 | 26.62 | 20.15 |
| Question-type driven model (Unified-model) | 43.11 | **29.13** | **21.39** | **16.31** | **55.67** | **38.16** | **28.12** | **21.59** |

Table 3: Experimental results of our model comparing with previous methods on two datasets.

### 3.3 Results and Analysis

**Main Results** The experimental results on two datasets are shown in Table 3. By incorporating the question type prediction, our model obtains obvious performance gain over the baseline model, with 1.42 points gain on SQuAD and 1.44 on MARCO. Comparing with previous methods, our model outperforms the existing best methods, achieving new state-of-the-art results on both datasets, with 16.31 on SQuAD and 21.59 on MARCO.

**Question Type Accuracy** We evaluate different models in terms of Beginning Question Word Accuracy (BQWA). This metric measures the ratio of the generated questions that share the same beginning word with the references which begin with a question word. Table 4 displays the BQWA of two models on both datasets. It shows that our unified model brings significant performance gain in question type prediction.

Further, in Figure 2 we show the accuracy of different question words on both datasets in detail. Our unified model outperforms the baseline for all question types on SQuAD, and all but two types on MARCO.

| Model | BQWA on SQuAD | BQWA on MARCO |
|---|---|---|
| Baseline | 57.62% | 76.98% |
| Unified model | **74.41%** | **80.48%** |

Table 4: Experiments of the beginning question word accuracy.

**Model Analysis** We conduct experiments on different variants of our model, as shown in Table 5. "w/o answer hidden state" takes $(x_{m+1}, ..., x_{m+a})$ as the input of type decoder instead of answer hidden states; "w/o question word replace <BOS>" simply use <BOS> as the first input word of decoder. Experiments on SQuAD verify the effectiveness of our model setting.

| Dataset / Model | SQuAD BLEU-1 | SQuAD BLEU-2 | SQuAD BLEU-3 | SQuAD BLEU-4 |
|---|---|---|---|---|
| Unified-model | 43.11 | 29.13 | 21.39 | 16.31 |
| w/o answer hidden states | 41.91 | 27.77 | 20.37 | 15.54 |
| w/o question word replace <BOS> | 41.05 | 26.93 | 19.73 | 15.10 |

Table 5: Experiments on different settings of our model.

**Case Study** To show the effect of question words prediction on question generation, Table 6 lists some typical examples.

In the first example, the baseline fails to recognize *Len-shaped* as an adjective, while the unified-model succeeds by utilizing lexical features which are the input of question type prediction layer.

In the second example, the baseline assigns a location type for the generated question based on *American* and *Israel*, it fails to consider the whole answer span. Our unified model resolves it by encoding the answer hidden states sequence as a whole.

The third example shows another typical error, which fails to consider answer surrounding context. In this example the given answer only contains a number *66*. By taking *article 66* into account, we know the question should not be a nu-

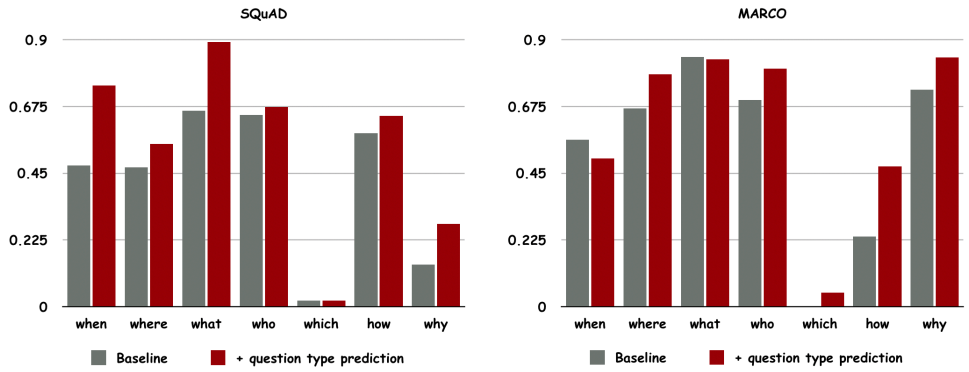


Figure 2: The accuracy on different question words.

merical type. Since the answer hidden states used as the input of type prediction contain surrounding information, our model resolves this problem.

| |
|---|
| **Context:** In land plants , chloroplasts are generally lens-shaped, 5–8 m in diameter and 1–3 m thick.<br>**Reference:** How are chloroplasts in land plants usually shaped?<br>**Baseline:** What are chloroplasts generally generally?<br>**Unified-model:** How are chloroplasts in land plants? |
| **Context:** In response to American aid to Israel, on October 16, 1973, OPEC raised the posted price of oil by 70%, to $5.11 a barrel.<br>**Reference:** Why did the oil ministers agree to a cut in oil production?<br>**Baseline:** Where did OPEC receive the price of oil by 70%?<br>**Unified-model:** Why did OPEC raised the posted price of oil by 70%? |
| **Context:** Article 65 of the agreement banned cartels and article 66 made provisions for concentrations, or mergers, and the abuse of a dominant position by companies.<br>**Reference:** Which article made provisions for concentrations or mergers and the abuse of a dominant position by companies?<br>**Baseline:** How many provisions made provisions for concentrations?<br>**Unified-model:** Which article made provisions for concentrations, or mergers? |

Table 6: Examples of generated questions. The underline words are the target answer.

## 4 Conclusion

In this paper, we discuss the challenge in question type prediction for question generation. We propose a unified model to predict the type and utilize it to guide the generation of question. Experiments on SQuAD and MARCO datasets verify the effectiveness of our model. We improve the accuracy of question type prediction by a large margin, and achieve new state-of-the-art results for question generation.

## Acknowledgments

We thank Weikang Li, Xin Jia and Nan Jiang for their valuable comments and suggestions. This work is supported by the National Natural Science Foundation of China (61773026).

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Xinya Du and Claire Cardie. 2017. Identifying where to focus in reading comprehension for neural question generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2067–2073.

Nan Duan, Duyu Tang, Peng Chen, and Ming Zhou. 2017. Question generation for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 866–874.

Wenpeng Hu, Bing Liu, Jinwen Ma, Dongyan Zhao, and Rui Yan. 2018. Aspect-based question generation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA.*, pages 311–318.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30*

- *August 4, Volume 1: Long Papers*, pages 1073–1083.

Heung-Yeung Shum, Xiaodong He, and Di Li. 2018. From eliza to xiaoice: challenges and opportunities with social chatbots. *Frontiers of IT & EE*, 19(1):10–26.

Linfeng Song, Zhiguo Wang, Wael Hamza, Yue Zhang, and Daniel Gildea. 2018. Leveraging context information for natural question generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers)*, pages 569–574.

Sandeep Subramanian, Tong Wang, Xingdi Yuan, Saizheng Zhang, Adam Trischler, and Yoshua Bengio. 2018. Neural models for key phrase extraction and question generation. In *Proceedings of the Workshop on Machine Reading for Question Answering@ACL 2018, Melbourne, Australia, July 19, 2018*, pages 78–88.

Xingwu Sun, Jing Liu, Yajuan Lyu, Wei He, Yanjun Ma, and Shi Wang. 2018. Answer-focused and position-aware neural question generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3930–3939.

Duyu Tang, Nan Duan, Tao Qin, and Ming Zhou. 2017. Question answering and question generation as dual tasks. *CoRR*, abs/1706.02027.

Yansen Wang, Chenyi Liu, Minlie Huang, and Liqiang Nie. 2018. Learning to ask questions in open-domain conversational systems with typed decoders. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 2193–2203.

Yao Zhao, Xiaochuan Ni, Yuanyuan Ding, and Qifa Ke. 2018. Paragraph-level neural question generation with maxout pointer and gated self-attention networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3901–3910.

Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. 2017. Neural question generation from text: A preliminary study. In *Natural Language Processing and Chinese Computing - 6th CCF International Conference, NLPCC 2017, Dalian, China, November 8-12, 2017, Proceedings*, pages 662–671.

# Deep Reinforcement Learning with Distributional Semantic Rewards for Abstractive Summarization

**Siyao Li**[1]*, **Deren Lei**[1]*, **Pengda Qin**[2], **William Yang Wang**[1]
[1]University of California, Santa Barbara
[2]Beijing University of Posts and Telecommunications
{siyaoli, derenlei}@ucsb.edu, qinpengda@bupt.edu.cn, william@cs.ucsb.edu

## Abstract

Deep reinforcement learning (RL) has been a commonly-used strategy for the abstractive summarization task to address both the exposure bias and non-differentiable task issues. However, the conventional reward ROUGE-L simply looks for exact n-grams matches between candidates and annotated references, which inevitably makes the generated sentences repetitive and incoherent. In this paper, instead of ROUGE-L, we explore the practicability of utilizing the distributional semantics to measure the matching degrees. With distributional semantics, sentence-level evaluation can be obtained, and semantically-correct phrases can also be generated without being limited to the surface form of the reference sentences. Human judgments on Gigaword and CNN/Daily Mail datasets show that our proposed distributional semantics reward (DSR) has distinct superiority in capturing the lexical and compositional diversity of natural language.

## 1 Introduction

Abstractive summarization is a task of paraphrasing a long article with fewer words. Unlike extractive summarization, abstractive summaries can include tokens out of the article's vocabulary. There exists several encoder-decoder approaches such as attention-based architecture (Bahdanau et al., 2015; Rush et al., 2015; Nallapati et al., 2016; Chopra et al., 2016), incorporating graph techniques (Moawad and Aref, 2012; Ganesan et al., 2010), and adding pointer generator (Vinyals et al., 2015; Nallapati et al., 2016; See et al., 2017; Bello et al., 2017). However, long sentence generation suffers from exposure bias (Bahdanau et al., 2017) as the error accumulates during the decoding process.

Many innovative deep RL methods (Ranzato et al., 2016; Wu et al., 2016; Paulus et al., 2018; Lamb et al., 2016) are developed to alleviate this issue by providing sentence-level feedback after generating a complete sentence, in addition to optimal transport usage (Napoles et al., 2012). However, commonly used automatic evaluation metrics for generating sentence-level rewards count exact n-grams matches and are not robust to different words that share similar meanings since the semantic level reward is deficient.

Currently, many studies on contextualized word representations (Peters et al., 2018; Devlin et al., 2019) prove that they have a powerful capacity of reflecting distributional semantic. In this paper, we propose to use the distributional semantic reward to boost the RL-based abstractive summarization system. Moreover, we design several novel objective functions.

Experiment results show that they outperform the conventional objectives while increasing the sentence fluency. Our main contributions are three-fold:

• We are the first to introduce DSR to abstractive summarization and achieve better results than conventional rewards.
• Unlike ROUGE, our DSR does not rely on cross-entropy loss (XENT) to produce readable phrases. Thus, no exposure bias is introduced.
• DSR improves generated tokens' diversity and fluency while avoiding unnecessary repetitions.

## 2 Methodology

**Background** While sequence models are usually trained using XENT, they are typically evaluated at test time using discrete NLP metrics such as BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), METEOR (Banerjee and Lavie, 2005). Therefore, they suffer from both the exposure bias

*Equal contributions.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6038–6044, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

and non-differentiable task metric issues. To solve these problems, many resorts to deep RL with sequence to sequence model (Paulus et al., 2018; Ranzato et al., 2016; Ryang and Abekawa, 2012), where the learning agent interacts with a given environment. However, RL models have poor sample efficiency and lead to very slow convergence rate. Therefore, RL methods usually start from a pretrained policy, which is established by optimizing XENT at each word generation step.

$$L_{\text{XENT}} = -\sum_{t=1}^{n'} \log P(y_t|y_1, \ldots, y_{t-1}, x). \quad (1)$$

Then, during RL stage, the conventional way is to adopt self-critical strategy to fine-tune based on the target evaluation metric,

$$L_{\text{RL}} = \sum_{t=1}^{n'} \log P(\hat{y_t}|\hat{y_1}, \ldots, \hat{y_{t-1}}, x) \times (r_{metric}(y^b) - r_{metric}(\hat{y})) \quad (2)$$

**Distributional Semantic Reward** During evaluating the quality of the generated sentences, ROUGE looks for exact matches between references and generations, which naturally overlooks the expression diversity of the natural language. In other words, it fails to capture the semantic relation between similar words. To solve this problem, distributional semantic representations are a practical way. Recent works on contextualized word representations, including ELMO (Peters et al., 2018), GPT (Radford et al., 2018), BERT (Devlin et al., 2019), prove that distributional semantics can be captured effectively. Based on that, a recent study, called BERTSCORE (Zhang et al., 2019), focuses on sentence-level generation evaluation by using pre-trained BERT contextualized embeddings to compute the similarity between two sentences as a weighted aggregation of cosine similarities between their tokens. It has a higher correlation with human evaluation on text generation tasks comparing to existing evaluation metrics.

In this paper, we introduce it as a DSR for deep RL. The BERTSCORE is defined as:

$$R_{\text{BERT}} = \frac{\sum_{y_i \in y} \text{idf}(y_i) \max_{\hat{y}_j \in \hat{y}} \mathbf{y_i}^\top \hat{\mathbf{y}}_\mathbf{j}}{\sum_{y_i \in y} \text{idf}(y_i)} \quad (3)$$

$$P_{\text{BERT}} = \frac{\sum_{y_j \in y} \text{idf}(y_j) \max_{\hat{y}_i \in \hat{y}} \mathbf{y_j}^\top \hat{\mathbf{y}}_\mathbf{i}}{\sum_{y_j \in y} \text{idf}(y_j)} \quad (4)$$

$$F_{\text{BERT}} = 2\frac{R_{\text{BERT}} \cdot P_{\text{BERT}}}{R_{\text{BERT}} + P_{\text{BERT}}} \quad (5)$$

where $\mathbf{y}$ and $\hat{\mathbf{y}}$ represent BERT contextual embeddings of reference word y and candidate word $\hat{y}$, respectively. The function idf$(\cdot)$ calculates inverse document frequency (idf). In our DSR, we do not use the idf since Zhang et al. (2019) requires to use the entire dataset including test set for calculation. Besides, ROUGE do not use similar weight, so we do not include idf for consistency.

## 3 Experimental Setup

### 3.1 Datasets

**Gigaword corpus** It is an English sentence summarization dataset based on annotated Gigaword (Napoles et al., 2012). A single sentence summarization is paired with a short article. We use the OpenNMT provided version It contains 3.8M training, 189k development instances. We randomly sample 15k instances as our test data.

**CNN/Daily Mail dataset** It consists of online news articles and their corresponding multi-sentence abstracts (Hermann et al., 2015; Nallapati et al., 2016). We use the non-anonymized version provided by See et al. (2017), which contains 287k training, 13k validation, and 11k testing examples. We truncate the articles to 400 tokens and limit the summary lengths to 100 tokens.

### 3.2 Pretrain

We first pretrain a sequence-to-sequence model with attention using XENT and then select the best parameters to initialize models for RL. Our models have 256-dimensional hidden states and 128-dimensional word embeddings and also incorporates the pointer mechanism (See et al., 2017) for handling out of vocabulary words.

### 3.3 Baseline

In abstractive summarization, ROUGE (Lin, 2004) is a common evaluation metric to provide a sentence-level reward for RL. However, using ROUGE as a pure RL objective may cause too many repetitions and reduced fluency in outputs. Paulus et al. (2018) propose a hybrid learning objective that combines XENT and self-critical ROUGE reward (Paulus et al., 2018).

$$L_{\text{baseline}} = \gamma L_{\text{rouge}} + (1 - \gamma) L_{\text{XENT}} \quad (6)$$

where $\gamma$ is a scaling factor and the F score of ROUGE-L is used as the reward to calculate $L_{\text{Rouge}}$. In our experiment, we select $\gamma = 0.998$ for Gigaword Corpus and $\gamma = 0.9984$ for CNN/Daily Mail dataset. [1] Note that we do not avoid repetition during the test time as Paulus et al. (2018) do, because we want to examine the repetition of sentence directly produced after training.

### 3.4 Proposed Objective Functions

Inspired by the above objective function (Paulus et al., 2018), we optimize RL models with a similar loss function as equation 2. Instead of ROUGE-L, we incorporate BERTSCORE, a DSR to provide sentence-level feedback.

In our experiment, $L_{\text{DSR}}$ is the self-critical RL loss (equation 2) with $F_{\text{BERT}}$ as the reward. We introduce the following objective functions:

**DSR+ROUGE:** A combined reward function of ROUGE and $F_{\text{BERT}}$

$$L_1 = \gamma L_{\text{DSR}} + (1 - \gamma) L_{\text{rouge}} \tag{7}$$

In our experiment, we select $\gamma = 0.5$ for both datasets to balance the influence of two reward functions.

**DSR+XENT:** $F_{\text{BERT}}$ reward with XENT to make the generated phrases more readable.

$$L_2 = \gamma' L_{\text{DSR}} + (1 - \gamma') L_{\text{XENT}} \tag{8}$$

In our experiment, we select $\gamma' = 0.998$ for Gigaword Corpus and $\gamma' = 0.9984$ for CNN/Daily Mail dataset.

**DSR:** Pure $F_{\text{BERT}}$ objective function without any teacher forcing.

$$L_3 = L_{\text{DSR}} \tag{9}$$

## 4 Results

For the abstractive summarization task, we test our models with different objectives on the Gigaword and CNN/Daily Mail datasets. We choose $F_{\text{BERT}}$ and ROUGE-L as our automatic evaluation metrics. For multi-sentence summaries in CNN/Daily Mail dataset, we calculate ROUGE-L and $F_{\text{BERT}}$ after concatenating the original abstract sentences or their BERT contextualized embeddings. The results are shown in Table 1. The model DSR+ROUGE (equation 7) achieves the best ROUGE performance on Gigaword and similar performance comparing to ROUGE model on CNN/Daily Mail. Training with DSR (equation 9) and DSR+XENT (equation 8) can obtain the best BERTScores as expected. It is also expected that ROUGE model will obtain worse BERTScores as simply optimizing ROUGE will generate less readable sentences (Paulus et al., 2018); however, DSR model without XENT as a teacher forcing can improve the performance of pretrained model in both $F_{\text{BERT}}$ and ROUGE-L scale.

Note that DSR model's ROUGE-L is high in training time but does not have a good generalization on test set, and ROUGE-L is not our target evaluation metrics. In the next section, we will do human evaluation to analyze the summarization performance of different reward systems.

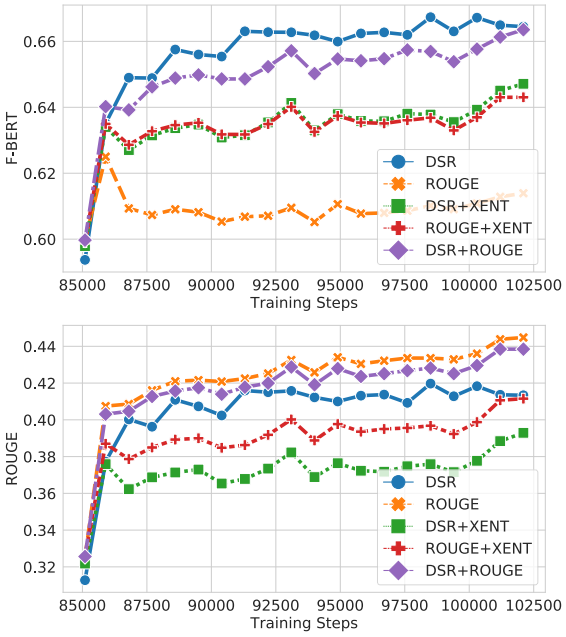


Figure 1: $F_{\text{BERT}}$ (left) and ROUGE (right) on Gigaword Corpus during training time after pretraining with XENT. Note that ROUGE is not our target evaluation metrics.

## 5 Human Evaluation

We perform human evaluation on the Amazon Mechanical Turk to assure the benefits of DSR on output sentences' coherence and fluency. We randomly sample 500 items as an evaluation set us-

[1] $\gamma$ is very close to 1 due to the difference in the scales of the two loss functions. For Gigaword Corpus, we tune the $\gamma$ on the development set. For CNN/Daily Mail, we use the same $\gamma$ as Paulus et al. (2018) does.

| Model | Gigawords | | | | CNN/Daily Mail | | | |
|---|---|---|---|---|---|---|---|---|
| | $F_{\text{BERT}}$ | $P_{\text{BERT}}$ | $R_{\text{BERT}}$ | ROUGE | $F_{\text{BERT}}$ | $P_{\text{BERT}}$ | $R_{\text{BERT}}$ | ROUGE |
| XENT | 65.78 | 67.61 | 64.53 | 40.77 | 62.77 | 62.18 | 63.79 | 29.46 |
| ROUGE | 61.46 | 60.89 | 62.70 | 42.73 | 60.11 | 59.31 | 61.33 | **33.89** |
| ROUGE+XENT | 66.50 | 67.24 | 66.28 | 42.72 | 61.38 | 61.07 | 62.17 | 33.07 |
| DSR+ROUGE | 66.48 | 66.79 | 66.65 | **42.95** | 65.01 | 65.92 | 64.56 | 33.61 |
| DSR+XENT | 67.02 | **67.69** | 66.85 | 42.29 | 66.64 | 66.06 | 67.63 | 31.28 |
| DSR | **67.06** | 67.34 | **67.28** | 41.73 | **66.93** | **66.27** | **67.98** | 30.96 |

Table 1: Results on Gigawords and CNN/Daily Mail for abstractive summarization. Upper rows show the results of baselines. Rouge stands for the F score of Rouge-L.

| Task | Model v. base | Gigawords | | | CNN/Daily Mail | | |
|---|---|---|---|---|---|---|---|
| | | Win | Lose | Tie | Win | Lose | Tie |
| Relevance | DSR + XENT | 31.0% | 25.2% | 43.8% | 51.1% | 34.1% | 14.8% |
| | DSR | 45.4% | 27.2% | 27.4% | 48.7% | 38.5% | 12.8% |
| Fluency | DSR + XENT | 40.2% | 19.6% | 40.2% | 55.8% | 28.5% | 15.6% |
| | DSR | 45.4% | 28.8% | 25.8% | 54.0% | 31.7% | 14.3% |

Table 2: Human evaluation results on Gigaword and CNN/Daily Mail for abstractive summarization. ROUGE+XENT is the selected baseline model.

ing uniform distribution. During the evaluation, given the context without knowing the ground truth summary, judges are asked to decide two things between generations of ROUGE+XENT and DSR+XENT: 1. which summary is more relevant to the article. 2. which summary is more fluent. Ties are permitted. A similar test is also done between ROUGE+XENT and DSR model. As shown in Table 2, DSR and DSR+XENT models improve the relevance and fluency of generated summary significantly. In addition, using pure DSR achieve better performances on Gigaword Corpus and comparable results as DSR+XENT on CNN/Daily Mail. While Paulus et al.'s (2018) objective function requires XENT to make generated sentence readable, our proposed DSR does not require XENT and can limit the exposure bias originated from it.

## 6 Analysis

**Diversity** Other than extractive summarization, abstractive summarization allows more degrees of freedom in the choice of words. While simply selecting words from the article made the task easier to train, higher action space can provide more paths to potentially better results (Nema et al., 2017). Using the DSR as deep RL reward will support models to choose actions that are not n-grams of the articles. In Table 4, we list a few generated samples on Gigaword Corpus. In our first example in Table 4, the word "sino-german" provides an interesting and efficient way to express the relation between China and Germany. $F_{\text{BERT}}$ is also improved by making this change. In addition, the second example in Table 4 shows that RL model with DSR corrects the sentence' grammar and significantly improves the $F_{\text{BERT}}$ score by switching "down" to an unseen word "drops". On the other hand, when optimizing DSR to improve the diversity of generation, some semantically similar words may also be generated and harm the summarization quality as shown in the third example in Table 4. The new token "wins" reduces the scores of both metrics. We also evaluate the diversity of a model quantitively by averaging the percentage of out-of-article n-grams in generated sentences. Results can be found in Table 3. The DSR model achieves the highest diversity.

| Model | Gigawords | | CNN/Daily Mail | |
|---|---|---|---|---|
| | Rep(%) | Div(%) | Rep(%) | Div(%) |
| ROUGE+XENT | 11.57 | 16.33 | 66.82 | 21.32 |
| DSR+ROUGE | 18.42 | 15.97 | 50.41 | 34.70 |
| DSR+XENT | 20.15 | 16.24 | 25.98 | 31.80 |
| DSR | **7.20** | **19.17** | **22.03** | **41.35** |

Table 3: Qualitative analysis on repetition(Rep) / diversity(Div). They are calculated by the percentage of repeat/out-of-article n-grams (unigrams for Gigaword and 5-grams for CNN/Daily Mail) in generated sentences.

| | |
|---|---|
| 1) Context | economic links between china and germany will be at the center of talks here next month between li peng and chancellor helmut kohl when the chinese premier makes a week-long visit to germany , sources on both sides said . |
| Groundtruth | li peng 's visit to germany to focus on economic ties |
| ROUGE+XENT | chinese premier to visit germany next month { $F_{\text{BERT}}$: 49.48, ROUGE:**18.64**} |
| DSR+XENT | chinese premier to visit germany next month { $F_{\text{BERT}}$: 49.48, ROUGE:**18.64**} |
| DSR | **sino-german** economic **links** to be at center of talks **with** germany {$F_{\text{BERT}}$: **50.77**, ROUGE:17.33} |
| 2) Context | the sensitive index dropped by #.## percent friday on the bombay stock exchange -lrb- bse -rrb- following brisk inquiries from operators that generally pressed sales to square up positions on the last day of the current settlement . |
| Groundtruth | sensex falls in bombay stock exchange |
| ROUGE+XENT | sensitive index down on bombay stock exchange {$F_{\text{BERT}}$: 70.03, ROUGE:**45.62**} |
| DSR+XENT | sensitive index **drops** on bombay stock exchange {$F_{\text{BERT}}$: **73.92**, ROUGE:**45.62**} |
| DSR | sensitive index **drops** on bombay stock exchange {$F_{\text{BERT}}$: **73.92**, ROUGE:**45.62**} |
| 3) Context | belgium 's rik verbrugghe took victory in tuesday 's prologue stage of the tour de romandie , with a confident ride through the streets of geneva 's old town . |
| Groundtruth | verbrugghe takes prologue victory in tour de romandie |
| ROUGE+XENT | verbrugghe **takes** victory in prologue stage of tour de romandie {$F_{\text{BERT}}$: 91.41, ROUGE:75.93} |
| DSR+XENT | verbrugghe **takes** victory in tour de romandie prologue stage {$F_{\text{BERT}}$: **91.56**, ROUGE:**81.79**} |
| DSR | verbrugghe **wins** victory in tour de romandie prologue stage {$F_{\text{BERT}}$: 89.45, ROUGE:70.10} |
| 4) Context | european finance ministers on saturday urged swedes to vote " yes " to adopting the euro , saying entry into the currency bloc would be good for sweden and the rest of europe . |
| Groundtruth | european finance ministers urge swedes to vote yes to euro |
| ROUGE+XENT | **eu** finance ministers urge swedes to vote yes to euro {$F_{\text{BERT}}$: **96.63**, ROUGE:**93.47**} |
| DSR+XENT | **eu** finance ministers urge swedes to vote yes to adopting euro {$F_{\text{BERT}}$: 90.38, ROUGE:88.89} |
| DSR | **eu** finance ministers urge swedes to vote yes to euro euro {$F_{\text{BERT}}$: 95.05, ROUGE:**93.47**} |

Table 4: Qualitative analysis of generated samples on Gigaword corpus. Generated words that do not appear in the context are marked blue. Repeated words are marked red. The first two examples represent DSR's generated tokens are more diverse. However, it may suffer from problems as shown in example 3 and 4.

**Repetition** Repetition will lead to lower $F_{\text{BERT}}$ as shown in the last example in Table 4. Using DSR reduces the probability of producing repetitions. The average percentage of repeated n-grams in generated sentences are presented in the Table 3. As shown in this table, unlike ROUGE, the DSR model can achieve high fluency without XENT; moreover, it produces the fewest repetitions among all the rewards. Table 4 gives an example that DSR produces a repeated word (from example 4), but it does not reflect the overall distribution of repeated word generation for all evaluated models.

## 7 Conclusion

This paper demonstrates the effectiveness of applying the distributional semantic reward to reinforcement learning in abstractive summarization, and specifically, we choose BERTSCORE. Our experimental results demonstrate that we achieve better performance on Gigaword and CNN/Daily Mail datasets. Besides, the generated sentences have fewer repetitions, and the fluency is also improved. Our finding is aligned to a contemporaneous study (Wieting et al., 2019) on leveraging semantic similarity for machine translation.

## References


Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2017. An actor-critic algorithm for sequence prediction. In *5th International Conference on Learning Representations, Conference Track Proceedings*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, Conference Track Proceedings*.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Irwan Bello, Hieu Pham, Quoc V. Le, Mohammad Norouzi, and Samy Bengio. 2017. Neural combinatorial optimization with reinforcement learning. In *5th International Conference on Learning Representations, Workshop Track Proceedings*.

Sumit Chopra, Michael Auli, and Alexander M Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. 2010. Opinosis: A graph based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 340–348.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in neural information processing systems*, pages 1693–1701.

Alex M Lamb, Anirudh Goyal Alias Parth Goyal, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. 2016. Professor forcing: A new algorithm for training recurrent networks. In *Advances In Neural Information Processing Systems*, pages 4601–4609.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Ibrahim F Moawad and Mostafa Aref. 2012. Semantic graph reduction approach for abstractive text summarization. In *2012 Seventh International Conference on Computer Engineering & Systems (ICCES)*, pages 132–138. IEEE.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Ça glar Gulçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *CoNLL 2016*, page 280.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 95–100. Association for Computational Linguistics.

Preksha Nema, Mitesh M Khapra, Anirban Laha, and Balaraman Ravindran. 2017. Diversity driven attention model for query-based abstractive summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1063–1072.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *6th International Conference on Learning Representations, Conference Track Proceedings*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT*, pages 2227–2237.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *4th International Conference on Learning Representations, Conference Track Proceedings*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389.

Seonggi Ryang and Takeshi Abekawa. 2012. Framework of automatic text summarization using reinforcement learning. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 256–265. Association for Computational Linguistics.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

John Wieting, Taylor Berg-Kirkpatrick, Kevin Gimpel, and Graham Neubig. 2019. Beyond bleu: Training neural machine translation with semantic similarity. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 4344–4355.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

# Clause-Wise and Recursive Decoding for Complex and Cross-Domain Text-to-SQL Generation

**Dongjun Lee**
SAP Labs Korea
dongjun.lee01@sap.com

## Abstract

Most deep learning approaches for text-to-SQL generation are limited to the WikiSQL dataset, which only supports very simple queries over a single table. We focus on the *Spider* dataset, a complex and cross-domain text-to-SQL task, which includes complex queries over multiple tables. In this paper, we propose a SQL clause-wise decoding neural architecture with a self-attention based database schema encoder to address the *Spider* task. Each of the clause-specific decoders consists of a set of sub-modules, which is defined by the syntax of each clause. Additionally, our model works recursively to support nested queries. When evaluated on the *Spider* dataset, our approach achieves 4.6% and 9.8% accuracy gain in the test and dev sets, respectively. In addition, we show that our model is significantly more effective at predicting complex and nested queries than previous work.

## 1 Introduction

Text-to-SQL generation is the task of translating a natural language question into the corresponding SQL. Recently, various deep learning approaches have been proposed based on the WikiSQL dataset (Zhong et al., 2017). However, because WikiSQL contains only very simple queries over just a single table, these approaches (Xu et al., 2017; Huang et al., 2018; Yu et al., 2018a; Dong and Lapata, 2018) cannot be applied directly to generate complex queries containing elements such as `JOIN`, `GROUP BY`, and nested queries.

To overcome this limitation, Yu et al. (2018c) introduced *Spider*, a new complex and cross-domain text-to-SQL dataset. It contains a large number of complex queries over different databases with multiple tables. It also requires a model to generalize to unseen database schema as different databases are used for training and testing. Therefore, a model should understand not only the natural language question but also the schema of the corresponding database to predict the correct SQL query.

In this paper, we propose a novel SQL-specific clause-wise decoding neural network model to address the *Spider* task. We first predict a sketch for each SQL clause (e.g., `SELECT`, `WHERE`) with text classification modules. Then, clause-specific decoders find the columns and corresponding operators based on the sketches. Our contributions are summarized as follows.

- We decompose the clause-wise SQL decoding process. We also modularize each of the clause-specific decoders into sub-modules based on the syntax of each clause. Our architecture enables the model to learn clause-dependent context and also ensures the syntactic correctness of the predicted SQL.
- Our model works recursively so that it can predict nested queries.
- We also introduce a self-attention based database schema encoder that enables our model to generalize to unseen databases.

In the experiment on the *Spider* dataset, we achieve 24.3% and 28.8% exact SQL matching accuracy on the test and dev set respectively, which outperforms the previous state-of-the-art approach (Yu et al., 2018b) by 4.6% and 9.8%. In addition, we show that our approach is significantly more effective compared to previous work at predicting not only simple SQL queries, but also complex and nested queries.

## 2 Related Work

Our work is related to the grammar-based constrained decoding approaches for semantic parsing (Yin and Neubig, 2017; Rabinovich et al., 2017;



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6045–6051,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

Iyer et al., 2018). While their approaches are focused on general purpose code generation, we instead focus on SQL-specific grammar to address the text-to-SQL task. Our task differs from code generation in two aspects. First, it takes a database schema as an input in addition to natural language. To predict SQL correctly, a model should fully understand the relationship between the question and the schema. Second, as SQL is a non-procedural language, predictions of SQL clauses do not need to be done sequentially.

For text-to-SQL generation, several SQL-specific approaches have been proposed (Zhong et al., 2017; Xu et al., 2017; Huang et al., 2018; Yu et al., 2018a; Dong and Lapata, 2018; Yavuz et al., 2018) based on WikiSQL dataset (Zhong et al., 2017). However, all of them are limited to the specific WikiSQL SQL sketch, which only supports very simple queries. It includes only the `SELECT` and `WHERE` clauses, only a single expression in the `SELECT` clause, and works only for a single table. To predict more complex SQL queries, sequence-to-sequence (Iyer et al., 2017; Finegan-Dollak et al., 2018) and template-based (Finegan-Dollak et al., 2018; Lee et al., 2019) approaches have been proposed. However, they focused only on specific databases such as ATIS (Price, 1990) and GeoQuery (Zelle and Mooney, 1996). Because they only considered question and SQL pairs without requiring an understanding of database schema, their approaches cannot generalize to unseen databases.

SyntaxSQLNet (Yu et al., 2018b) is the first and state-of-the-art model for the *Spider* (Yu et al., 2018c), a complex and cross-domain text-to-SQL task. They proposed an SQL specific syntax tree-based decoder with SQL generation history. Our approach differs from their model in the following aspects. First, taking into account that SQL corresponds to non-procedural language, we develop a clause-specific decoder for each SQL clause, where SyntaxSQLNet predicts SQL tokens sequentially. For example, in SyntaxSQLNet, a single column prediction module works both in the `SELECT` and `WHERE` clauses, depending on the SQL decoding history. In contrast, we define and train decoding modules separately for each SQL clause to fully utilize clause-dependent context. Second, we apply sequence-to-sequence architecture to predict columns instead of using the sequence-to-set framework from SyntaxSQLNet, because correct ordering is essential for the `GROUP BY` and `ORDER BY` clauses. Finally, we introduce a self-attention mechanism (Lin et al., 2017) to efficiently encode database schema, which includes multiple tables.

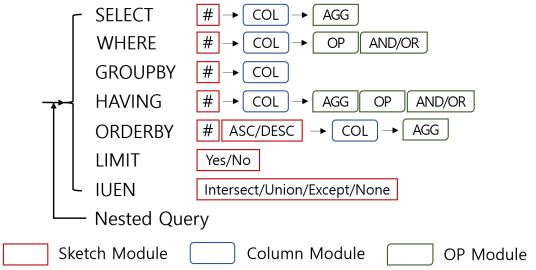


Figure 1: Clause-wise and recursive SQL generation process.

## 3 Methodology

We predict complex SQL clause-wisely as described in Figure 1. Each clause is predicted consecutively by at most three different types of modules (sketch, column, operator). The same architecture recursively predicts nested queries with temporal predicted SQL as an additional input.

### 3.1 Question and Schema Encoding

We encode a natural language question with a bi-directional LSTM. We denote $H_Q \in \mathbb{R}^{d \times |X|}$ as the question encoding, where $d$ is the number of LSTM units and $|X|$ is the number of tokens in the question.

To encode a database schema, we consider each column in its tables as a concatenated sequence of words from the table name and column name with a separation token. (e.g., [student, `[SEP]`, first, name]). First, we apply bi-directional LSTM over this sequence for each column. Then, we apply the self-attention mechanism (Lin et al., 2017) over the LSTM outputs to form a summarized fixed-size vector for each column. For the $i$th column, its encoding $h_{col}^{(i)} \in \mathbb{R}^d$ is computed by a weighted sum of the LSTM output $o_{col}^{(i)} \in \mathbb{R}^{d \times |L|}$ as follows:

$$\alpha = \texttt{softmax}(w^T \texttt{tanh}(o_{col}^{(i)})) \quad (1)$$

$$h_{col}^{(i)} = o_{col}^{(i)} \alpha^T \quad (2)$$

where $|L|$ is the number of tokens in the column and $w \in \mathbb{R}^d$ is a trainable parameter. We denote $H_{col} = [h_{col}^{(1)}, ...h_{col}^{(|C|)}]$ as columns encoding where $|C|$ is the number of columns in the database.

### 3.2 Sketch Prediction

We predict the clause-wise sketch via 8 different text classification modules that include the number of SQL expressions in each clause, the presence of `LIMIT` clause, and the presence of `INTERSECT/UNION/EXCEPT` as described in Figure 1. All of them share the same model architecture but are trained separately. For the classification, we applied attention-based bi-directional LSTM following Zhou et al. (2016).

First, we compute sentence representation $r_s \in \mathbb{R}^d$ by a weighted sum of question encoding $H_Q \in \mathbb{R}^{d\times|X|}$. Then we apply the softmax classifier to choose the sketch as follows:

$$\alpha_s = \texttt{softmax}(w_s^T \texttt{tanh}(H_Q)) \quad (3)$$

$$r_s = H_Q\, \alpha_s^T \quad (4)$$

$$P_{sketch} = \texttt{softmax}(W_s r_s + b_s) \quad (5)$$

where $w_s \in \mathbb{R}^d, W_s \in \mathbb{R}^{n_s\times d}, b_s \in \mathbb{R}^{n_s}$ are trainable parameters and $n_s$ is the number of possible sketches.

### 3.3 Columns and Operators Prediction

To predict columns and operators, we use the LSTM decoder with the attention mechanism (Luong et al., 2015) such that *the number of decoding steps are decided by the sketch prediction module*. We train 5 different column prediction modules separately for each SQL clause, but they share the same architecture.

In the column prediction module, the hidden state of the decoder at the $t$-th decoding step is computed as $d_{col}^{(t)} (\in \mathbb{R}^d) = \texttt{LSTM}(d_{col}^{(t-1)}, h_{col}^{(t-1)})$, where $h_{col}^{(t-1)} \in \mathbb{R}^d$ is an encoding of the predicted column in the previous decoding step. The context vector $r^{(t)}$ is computed by a weighted sum of question encodings $H_Q \in \mathbb{R}^{d\times|X|}$ based on attention weight as follows:

$$\alpha^{(t)} = \texttt{softmax}({d_{col}^{(t)}}^T H_Q) \quad (6)$$

$$r^{(t)} = H_Q\, {\alpha^{(t)}}^T \quad (7)$$

Then, the attentional output of the $t$-th decoding step $a_{col}^{(t)}$ is computed as a linear combination of $d_{col}^{(t)} \in \mathbb{R}^d$ and $r^{(t)} \in \mathbb{R}^d$ followed by `tanh` activation.

$$a_{col}^{(t)} = \texttt{tanh}(W_1 d_{col}^{(t)} + W_2 r^{(t)}) \quad (8)$$

where $W_1, W_2 \in \mathbb{R}^{d\times d}$ are trainable parameters. Finally, the probability for each column at the $t$-th decoding step is computed as a dot product between $a_{col}^{(t)} \in \mathbb{R}^d$ and the encoding of each column in $H_{col} \in \mathbb{R}^{d\times|C|}$ followed by softmax.

$$P_{col}^{(t)} = \texttt{softmax}({a_{col}^{(t)}}^T H_{col}) \quad (9)$$

To predict corresponding operators for each predicted column, we use a decoder of the same architecture as in the column prediction module. The only difference is that a decoder input at the $t$-th decoding step is an encoding of the $t$-th predicted column from the column prediction module.

$$d_{op}^{(t)} = \texttt{LSTM}(d_{op}^{(t-1)}, h_{col}^{(t)}) \quad (10)$$

Attentional output $a_{op}^{(t)} \in \mathbb{R}^d$ is computed identically to Eq. (8). Then, the probability for operators corresponding to the $t$-th predicted column is computed by the softmax classifier as follows:

$$P_{op}^{(t)} = \texttt{softmax}(W_o a_{op}^{(t)} + b_o) \quad (11)$$

where $W_o \in \mathbb{R}^{n_o\times d}$ and $b_o \in \mathbb{R}^{n_o}$ are trainable parameters and $n_o$ is the number of possible operators.

### 3.4 From Clause Prediction

After the predictions of all the other clauses, we use a heuristic to generate the `FROM` clause. We first collect all the columns that appear in the predicted SQL, and then we `JOIN` tables that include these predicted columns.

### 3.5 Recursion for Nested Queries

To predict the presence of a sub-query, we train another module that has the same architecture as the operator prediction module. Instead of predicting corresponding operators for each column, it predicts whether each column is compared to a variable (e.g., `WHERE` age $>$ 3) or to a sub-query (e.g., `WHERE` age $>$ (`SELECT avg`(age) ..)). In the latter case, we add the temporal `[SUB_QUERY]` token to the corresponding location in the SQL output. Additionally, if the sketch prediction module predicts one of `INTERSECT/UNION/EXCEPT` operators, we add a `[SUB_QUERY]` token after the operator.

To predict a sub-query, our model takes the temporal generated SQL with a `[SUB_QUERY]` token as an input in addition to a natural language question with separate token `[SEP]` (e.g.,

| Method | Dev Easy | Medium | Hard | Extra Hard | All | Test All |
|---|---|---|---|---|---|---|
| SQLNet | 23.2% | 8.6% | 9.8% | 0% | 10.9% | 12.4% |
| TypeSQL | 18.8% | 5.5% | 4.6% | 2.4% | 8.0% | 8.2% |
| SyntaxSQLNet | 38.4% | 15.0% | 16.1% | 3.5% | 19.0% | 19.7% |
| Ours | **53.2%** | **27.0%** | **20.1%** | **6.5%** | **28.8%** | **24.3%** |
| -rec | 53.2% | 27.0% | 14.4% | 2.9% | 27.4% | - |
| -rec - col-att | 46.4% | 22.0% | 12.1% | 4.7% | 23.4% | - |
| -rec -col-att -sketch | 33.2% | 18.6% | 11.5% | 4.7% | 18.7% | - |

Table 1: Accuracy of exact SQL matching with different hardness levels.

| Method | SELECT | WHERE | GROUP BY | ORDER BY | KEYWORDS |
|---|---|---|---|---|---|
| SQLNet | 46.6% | 20.6% | 37.6% | 49.2% | 62.8% |
| TypeSQL | 43.7% | 14.8% | 16.9% | 52.1% | 67.0% |
| SyntaxSQLNet | 55.4% | 22.2% | 51.4% | 50.6% | 73.3% |
| Ours | **68.7%** | **39.0%** | **63.1%** | **63.5%** | **76.5%** |

Table 2: F1 scores of SQL component matching on the *dev* set.

What is ... `[SEP]` SELECT ... INTERSECT `[SUB_QUERY]`). This input is encoded in the same way as question encoding described in Section 3.1. Then, the rest of the SQL generation process is identical to that described in Section 3.2–3.4. After the sub-query is predicted, it replaces the `[SUB_QUERY]` token to form the final query.

## 4 Experiments

### 4.1 Experimental Setup

We evaluate our model with *Spider* (Yu et al., 2018c), a large-scale, complex and cross-domain text-to-SQL dataset. We follow the same database split as Yu et al. (2018c), which ensures that any database schema that appears in the training set does not appear in the dev or test set. Through this split, we examine how well our model can be generalized to unseen databases. Because the test set is not opened to the public, we use the *dev* set for the ablation analysis. For the evaluation metrics, we use 1) accuracy of exact SQL matching and 2) F1 score of SQL component matching, proposed by (Yu et al., 2018c). We also follow their query hardness criteria to understand the model performance on different levels of queries. Our model and all the baseline models are trained based on only the *Spider* dataset without data augmentation.

### 4.2 Model Configuration

We use the same hyperparameters for every module. For the word embedding, we apply deep contextualized word representations (ELMO) from Peters et al. (2018) and allow them to be fine-tuned during the training. For the question and column encoders, we use a 1-layer 512-unit bi-directional LSTM. For the decoders in the columns and operators prediction modules, we use a 1-layer 1024-unit uni-directional LSTM. For the training, we use Adam optimizer (Kingma and Ba, 2014) with a learning rate of 1e-4 and use early stopping with 50 epochs. Additionally, we use dropout (Hinton et al., 2012) with a rate of 0.2 for the regularization.

### 4.3 Result and Analysis

Table 1 shows the exact SQL matching accuracy of our model and previous models. We achieve 24.3% and 28.8% on the test and dev sets respectively, which outperforms the previous best model SyntaxSQLNet (Yu et al., 2018b) by 4.6% and 9.8%. Moreover, our model outperforms previous models on all different query hardness levels.

To examine how each technique contributes to the performance, we conduct an ablation analysis of three aspects: 1) without recursion, 2) without self-attention for database schema encoding, and 3) without sketch prediction modules that decide the number of decoding steps. Without recursive sub-query generation, the accuracy drops by 5.7% and 3.6% for hard and extra hard queries, respectively. This result shows that the recursion we use enables the model to predict nested queries. When using the final LSTM hidden state as in Yu et al. (2018b) instead of using self-attention for schema encoding, the accuracy drops by 4.0% on all queries. Finally, when using only an encoder-

decoder architecture without sketch generation for columns prediction, the accuracy drops by 4.7%.

For the component matching result for each SQL clause, our model outperforms previous approaches for all of the SQL components by a significant margin, as shown in Table 2. Examples of predicted SQL from different models are shown in Appendix A.

## 5 Conclusion

In this paper, we propose a recursive and SQL clause-wise decoding neural architecture to address the complex and cross-domain text-to-SQL task. We evaluate our model with the *Spider* dataset, and the experimental result shows that our model significantly outperforms previous work for generating not only simple queries, but also complex and nested queries.

## Acknowledgments

We thank Yongsik Lee, Jaesik Yoon, and Donghun Lee (SAP) for their reviews and support. We also thank professor Sungroh Yoon, Jongyun Song, and Taeuk Kim (Seoul National University) for their insightful feedback and three anonymous reviewers for their helpful comments.

## References

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 731–742.

Catherine Finegan-Dollak, Jonathan K Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving text-to-sql evaluation methodology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 351–360.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Po-Sen Huang, Chenglong Wang, Rishabh Singh, Wen-tau Yih, and Xiaodong He. 2018. Natural language to structured query generation via meta-learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 732–738.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. *arXiv preprint arXiv:1704.08760*.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. 2018. Mapping language to code in programmatic context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1643–1652.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Dongjun Lee, Jaesik Yoon, Jongyun Song, Sanggil Lee, and Sungroh Yoon. 2019. One-shot learning for text-to-sql generation. *arXiv preprint arXiv:1905.11499*.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 2227–2237.

Patti J Price. 1990. Evaluation of spoken language systems: The atis domain. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*.

Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract syntax networks for code generation and semantic parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1139–1149.

Xiaojun Xu, Chang Liu, and Dawn Song. 2017. Sqlnet: Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv:1711.04436*.

Semih Yavuz, Izzeddin Gur, Yu Su, and Xifeng Yan. 2018. What it takes to achieve 100% condition accuracy on wikisql. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1702–1711.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *Proceedings of the 55th Annual Meeting of the*

*Association for Computational Linguistics (Volume 1: Long Papers)*, pages 440–450.

Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. 2018a. Typesql: Knowledge-based type-aware neural text-to-sql generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 588–594.

Tao Yu, Michihiro Yasunaga, Kai Yang, Rui Zhang, Dongxu Wang, Zifan Li, and Dragomir Radev. 2018b. Syntaxsqlnet: Syntax tree networks for complex and cross-domain text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1653–1663.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 2018c. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921.

John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the national conference on artificial intelligence*, pages 1050–1055.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 207–212.

## A Sample SQL Predictions

In Table 3, we show some examples of predicted SQL queries from different models. We compare the result of our model with two of previous state-of-the-art models: SyntaxSQLNet (Yu et al., 2018b) and the modified version of SQLNet (Xu et al., 2017) by Yu et al. (2018c) to support complex SQL queries.

| hardness | type | description |
|---|---|---|
| easy | NL | What are the names of all the countries that became independent after 1950? |
| | Truth | SELECT Name FROM country WHERE IndepYear > 1950 |
| | Ours | SELECT Name FROM country WHERE IndepYear > "[VAR]" |
| | Syntax | SELECT Name FROM country WHERE GovernmentForm = "[VAR]" |
| | SQLNet | SELECT T1.Name FROM city as T1 JOIN country as T2 WHERE T2.Population > "[VAR]" |
| medium | NL | Which city and country is the Alton airport at? |
| | Truth | SELECT City, Country FROM airports WHERE AirportName = "Alton" |
| | Ours | SELECT City, Country FROM airports WHERE AirportName = "[VAR]" |
| | Syntax | SELECT Country, City FROM airports WHERE Country = "[VAR]" |
| | SQLNet | SELECT T1.City, T2.DestAirport FROM airports as T1 JOIN flights as T2 |
| medium | NL | List the names of poker players ordered by the final tables made in ascending order. |
| | Truth | SELECT T1.Name FROM people as T1 JOIN poker_player as T2 ORDER BY T2.Final_Table_Made |
| | Ours | SELECT T1.Name FROM people as T1 JOIN poker_player as T2 ORDER BY T2.Final_Table_Made ASC |
| | Syntax | SELECT T2.Name FROM poker_player as T1 JOIN people as T2 ORDER BY T1.Earnings ASC |
| | SQLNet | SELECT Name FROM people ORDER BY Birth_Date ASC |
| medium | NL | How much does the most recent treatment cost? |
| | Truth | SELECT cost_of_treatment FROM Treatments ORDER BY date_of_treatment DESC LIMIT 1 |
| | Ours | SELECT cost_of_treatment FROM Treatments ORDER BY cost_of_treatment DESC LIMIT "[VAR]" |
| | Syntax | SELECT cost_of_treatment FROM Treatments ORDER BY cost_of_treatment ASC LIMIT "[VAR]" |
| | SQLNet | SELECT T1.charge_amount FROM Charges as T1 JOIN Dogs as T2 ORDER BY date_adopted DESC LIMIT "[VAR]" |
| hard | NL | List the names of teachers who have not been arranged to teach courses. |
| | Truth | SELECT Name FROM teacher WHERE Teacher_id NOT IN (SELECT Teacher_id FROM course_arrange) |
| | Ours | SELECT Name FROM teacher WHERE Teacher_id NOT IN (SELECT Teacher_id FROM course_arrange) |
| | Syntax | SELECT Name FROM teacher |
| | SQLNet | SELECT Name FROM teacher |
| hard | NL | Which cities do more than one employee under age 30 come from? |
| | Truth | SELECT city FROM employee WHERE age < 30 GROUP BY city HAVING count(*) > 1 |
| | Ours | SELECT city FROM employee WHERE age < "[VAR]" GROUP BY city HAVING count(*) > "[VAR]" |
| | Syntax | SELECT city FROM employee WHERE age > "[VAR]" |
| | SQLNet | SELECT T1.city FROM employee as T1 JOIN hiring as T2 JOIN shop as T3 WHERE T3.District > "[VAR]" GROUP BY T1.city HAVING count(*) > "[VAR]" |
| hard | NL | What is the document id with least number of paragraphs? |
| | Truth | SELECT document_id FROM Paragraphs GROUP BY document_id ORDER BY count(*) LIMIT 1 |
| | Ours | SELECT document_id FROM Documents GROUP BY document_id ORDER BY count(*) ASC LIMIT "[VAR]" |
| | Syntax | SELECT document_id FROM Documents GROUP BY document_id ORDER BY count(*) ASC LIMIT "[VAR]" HAVING count(*) >= "[VAR]" |
| | SQLNet | SELECT template_id FROM Templates GROUP BY template_id HAVING sum(*) NOT "[VAR]" ORDER BY count(*) ASC LIMIT "[VAR]" |
| extra | NL | How many dogs have not gone through any treatment? |
| | Truth | SELECT count(*) FROM Dogs WHERE dog_id NOT IN (SELECT dog_id FROM Treatments) |
| | Ours | SELECT count(*) FROM Dogs WHERE dog_id NOT IN (SELECT dog_id FROM Treatments) |
| | Syntax | SELECT count(*) FROM Charges WHERE charge_id NOT IN (SELECT charge_id FROM Charges) |
| | SQLNet | SELECT count(*) FROM Dogs WHERE dog_id IN "[VAR]" |
| extra | NL | What is the name of the high schooler who has the greatest number of friends? |
| | Truth | SELECT T2.name FROM Friend as T1 JOIN Highschooler as T2 GROUP BY T1.student_id ORDER BY count(*) DESC LIMIT 1 |
| | Ours | SELECT T1.name FROM Highschooler as T1 JOIN Friend as T2 GROUP BY T2.student_id ORDER BY count(*) DESC LIMIT 1 |
| | Syntax | SELECT name FROM Highschooler ORDER BY grade DESC LIMIT 1 |
| | SQLNet | SELECT T1.name FROM Friend as T1 JOIN Friend as T2 GROUP BY T2.student_id ORDER BY * DESC LIMIT 1 |

Table 3: Sample SQL predictions by our model and previous state-of-the-art models on the dev split. NL denotes the natural language question and Truth denotes the corresponding ground truth SQL query. Ours, Syntax, and SQLNet denotes the SQL predictions from our model, SyntaxSQLNet (Yu et al., 2018b), and modified SQLNet (Xu et al., 2017) by Yu et al. (2018c), respectively.

# Do Nuclear Submarines Have Nuclear Captains? A Challenge Dataset for Commonsense Reasoning over Adjectives and Objects

**James Mullenbach**[1*], **Jonathan Gordon**[2*], **Nanyun Peng**[3], **Jonathan May**[3]
[1] ASAPP Inc.
[2] Department of Computer Science, Vassar College
[3] Information Sciences Institute, University of Southern California
jmullenbach@asapp.com, jgordon@vassar.edu
npeng,jonmay@isi.edu

## Abstract

How do adjectives project from a noun to its parts? If a motorcycle is red, are its wheels red? Is a nuclear submarine's captain nuclear? These questions are easy for humans to judge using our commonsense understanding of the world, but are difficult for computers. To attack this challenge, we crowdsource a set of human judgments that answer the English-language question "Given a whole described by an adjective, does the adjective also describe a given part?" We build strong baselines for this task with a classification approach. Our findings indicate that, despite the recent successes of large language models on tasks aimed to assess commonsense knowledge, these models do not greatly outperform simple word-level models based on pre-trained word embeddings. This provides evidence that the amount of commonsense knowledge encoded in these language models does not extend far beyond that already baked into the word embeddings. Our dataset will serve as a useful testbed for future research in commonsense reasoning, especially as it relates to adjectives and objects.

## 1 Introduction

We investigate the commonsense inference of the transitivity of an attribute of a whole object to its component parts. To illustrate this targeted reasoning by example, "is a sharp knife's handle sharp?" The ability to perform commonsense inference of this type enables a more complete understanding of the physical world and therefore may find use in a variety of tasks in pragmatics and at the interface of vision and language. Consider generating a story in which a slow car goes to the shop to get a new part. If the new part is a windshield, the car remains slow, whereas if the new part is an engine, the car may now be fast. This knowledge may also help a visual agent reason about unseen objects: it knows a brick house does not have a brick door without needing to see the door.

The past few years have seen a raft of data sets intended to test our ability to construct models with an understanding of commonsense knowledge. Standout examples are the Stanford Natural Language Inference (SNLI) and related Multi-Genre Natural Language Inference (MNLI) corpora (Bowman et al., 2015; Williams et al., 2018), the SemEval-2018 commonsense shared task (Ostermann et al., 2018), the Rochester Story Completion (ROCStories) corpus (Mostafazadeh et al., 2016), and the Situations with Adversarial Generations (SWAG) grounded inference corpus (Zellers et al., 2018). After their release, very large language models (LMs) were able to reach or surpass human-level performance on SNLI (Peters et al., 2018) and SWAG (Devlin et al., 2018).

However, researchers have found inadequacies in these datasets and the models trained on them. Despite the strong performance of recent systems on SNLI (e.g., Chen et al., 2017; Parikh et al., 2016), Glockner et al. (2018) show that by making trivial changes to the test set, these methods suffered. Further, Pavlick and Callison-Burch (2016) show that state-of-the-art models for natural language inference fail on a task requiring only reasoning over adjective-noun relations. Relatedly, in the shared task to predict sentence endings of ROCStories, Schwartz et al. (2017) show that by incorporating style features, with only the answer choices as input, it is possible to reach near state-of-the-art performance. These results point to implicit bias baked into the data sets.

Rudinger et al. (2017) demonstrate similar systematic and social bias in SNLI, attributing it to the fact that hypothesis sentences were written by crowd workers. The SWAG data set was specif-

*Research conducted while author was at USC/ISI



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6052–6058, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

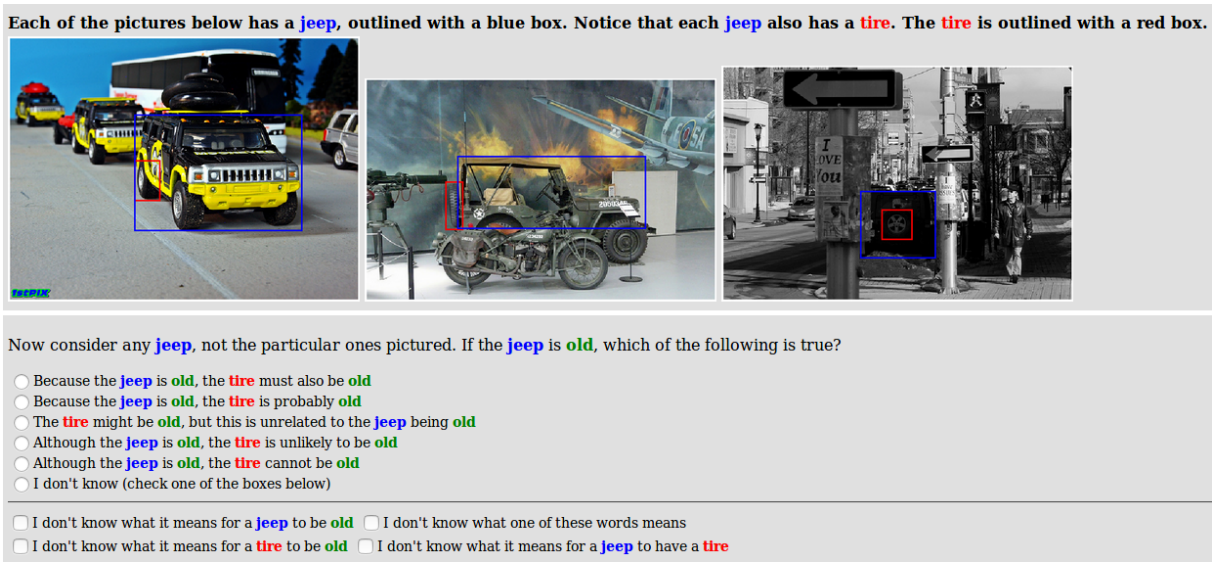


Figure 1: Visual annotation interface, excluding overall instructions.

ically constructed in an adversarial way with this in mind, but may be disadvantaged by the fact that continuation sentences are generated by computers. This may lead to patterns that are hard to detect but can nevertheless be picked up by other language models. We avoid the issue of elicitation bias by first collecting candidates grounded in natural sources of text and images, and then gathering only scaled judgments from crowd workers, as was done by Zhang et al. (2017).

To understand how to build truly intelligent agents, we should strive to create datasets with as little exploitable bias as possible, and to further investigate the landscape of current performance. We contribute a dataset which provides a focused evaluation, based on a specific task in commonsense reasoning. Gathering and validating data from crowd workers, we evaluate a number of approaches to performing these inferences, a three-way lexical entailment problem. We find that simple word embedding-based models perform adequately, but beneath humans, on this task, with recent large LM approaches (Devlin et al., 2018; Radford et al., 2018) providing only slight improvement over the purely lexical approach.

## 2 Related Work

Other researchers have constructed datasets investigating similar ideas in commonsense reasoning. Forbes and Choi (2017) develop a dataset and methods for inferring physical commonsense knowledge from verb usage, showing it is possible to learn the physical implications of unseen verbs from a small seed set. Zhang et al. (2017) create a large dataset for general commonsense inference in the form of premise-hypothesis pairs, equipped with ordinal labels ranging from "impossible" to "very likely". We adopt much of their methodology but for a targeted subset of commonsense reasoning. The SemEval 2018 Task 10 on Capturing Discriminative Attributes (Krebs et al., 2018) describes a similar lexical reasoning task involving triplets of words, though it focuses on finding attributes that *distinguish* two concepts, while in our work the adjective may well apply to both part and whole.

Past work has also evaluated commonsense capabilities in neural models. Pavlick and Callison-Burch (2016) investigate the related problem of entailment in adjective-nouns, and show surprising negative results for neural NLI models. Wang et al. (2018) showed that models based on distributional semantics without explicit external knowledge perform poorly at predicting physical plausibility of actions.

Lucy and Gauthier (2017) investigate perceptual properties of distributional embeddings and suggest that part–whole properties like *has_legs* are well encoded by embeddings. This may help explain why the simple word-based MLP models perform well without other sources of context. Rei et al. (2018) introduce an effective neural architecture for learning word-embedding based models for graded lexical entailment. Prior work (Bulat et al., 2016; Fagarasan et al., 2015) utilizes embeddings to predict real-world perceptual proper-

| Whole | Part | Adjective | Label | NLI premise | NLI hypothesis |
|---|---|---|---|---|---|
| armchair | arm | black | Probably | On the back of the president's quaint black armchair there was emblazoned a half-sun, brilliant with its gilded rays. | The armchair's arm is black. |
| vanity | mirror | white | Impossible | A door to a bathroom half open and a white vanity. | The vanity's mirror is white. |
| bench | support | wooden | Unrelated | In front of me about five feet distance, stood a wooden bench. | The bench's support is wooden. |

Table 1: Example triples and retrieved premise sentences, with labels, used for training word embedding-based models and language model fine-tuning.

ties, and we expect an approach that leverages this will help solve this task, but we leave it to future work.

## 3 Candidate collection

We seek to annotate examples of (*whole*, *part*, *adjective*) triples with answers to the question: "Does an ⟨*adjective*⟩ ⟨*whole*⟩ have an ⟨*adjective*⟩ ⟨*part*⟩?" As a major part of our contribution, we provide an annotated dataset that is visually grounded, with relations mined from Visual Genome (Krishna et al., 2017) and Google Syntactic N-grams (Goldberg and Orwant, 2013). We provide an overview here, with details in Appendix A.

### 3.1 Part–whole relations

Visual Genome (VG) is a large dataset of images annotated with objects, their attributes, and the relations between them. We start by considering all relationships in the VG dataset where the predicate is an underspecified *has* relation. We count the number of images in which a pair of objects appear in a *has* relation, and keep only those pairs appearing in at least three distinct images.

### 3.2 Adjectives

We gather adjectives from both Google Syntactic N-grams and VG. From Syntactic N-grams, we count the occurrences of an adjective modifying a noun with the *amod* relation. We remove common non-attributive (e.g., *awake*) and non-descriptive (e.g., *first*) adjectives using manually constructed lexicons. Then, for each whole noun, we gather its five most common adjectival modifiers, as well as its five most common adjective attributes from Visual Genome. Through pilot studies we observed that without further filtering, annotations were highly skewed towards non-entailment, thus we achieve a more balanced dataset by filtering out adjectives that are never observed attached to the part.

## 4 Collecting human annotations

We crowdsource annotations on Amazon Mechanical Turk (AMT) for each (*whole*, *part*, *adjective*) triple as follows:

### 4.1 Task overview

For each part–whole pair, we sample three random images from VG that contain the pair, and draw bounding boxes around both objects, provided by VG annotations. We present these to workers simply to provide context for the part–whole pair, since early tests showed that without visual cues workers often have trouble understanding the overall problem. Then, we ask a series of questions that each associates the pair with an adjective. To encourage the worker to imagine the prototypical version of the objects rather than the specific ones shown,[1] we use the template "Consider any ⟨*whole*⟩, not the particular ones pictured". Specific questions have the form: "If the ⟨*whole*⟩ is ⟨*adjective*⟩, which of the following is true?" The answers describe whether it is "impossible", "unlikely", "unrelated", "likely", or "guaranteed" that the identified part is also described by the adjective. The answers use causal language to encourage "conditional plausibility" thinking, as described by Zhang et al. (2017). This also allows for the "unrelated" answer, which covers spu-

[1] This is a necessary downside of displaying visual cues.

| Label | Percentage |
|---|---|
| Guaranteed | 44.2 |
| Probably | 19.8 |
| Unrelated | 23.7 |
| Unlikely | 5.6 |
| Impossible | 6.6 |

Table 2: Final label distribution

rious examples, such as a black guitar's cord being black, where the cord is likely black, but not *as a result* of the guitar being black. We also give an option for the worker to mark that one of the pairwise relations is nonsensical.

### 4.2 Qualification task

After manually annotating some examples, and conducting two AMT pilot studies, we found a non-trivial margin between our own agreement and that of workers, as measured by the quadratic-weighted Cohen's $\kappa$. To alleviate this, we followed Zhang et al. (2017) and conducted a pilot study to gather a pool of qualified workers. We launched a pilot task with two gold examples from each class on which our manual annotations agreed, and recruited 300 crowd workers to label them. By setting a $\kappa$ threshold on agreement with the gold examples at 0.7, this resulted in 106 qualified workers, whom we requested to perform the rest of the annotations. We collected at least three annotations per triple. An example annotation interface is shown in Figure 1.

### 4.3 Filtering and statistics

From the total of 20,284 triples annotated, we filter out 4,040 (19.9%) that were reported to contain an invalid triple. We further remove instances without a majority vote from the workers. This results in a set of 13,684 triples with an inter-annotator agreement (quadratic-weighted Cohen's $\kappa$) of 0.624. (For reference, Zhang et al. (2017) report $\kappa = 0.54$ for general commonsense inference.) The label distribution is shown in Table 2. The dataset has 728 unique *part* nouns, 873 unique *whole* nouns, and 553 unique adjectives.

## 5 Inference baselines

We now describe several basic approaches for solving these commonsense inference problems, which we intend as a baseline to be built upon by future work. Formally, models answer the question: Given (**1**) a noun denoting a whole object that has (**2**), a component part also denoted by a noun, does (**3**), an adjective that describes **1** also describe **2**? The data is first split into training, validation, and test sets consisting of 70%, 10%, and 20% of the data respectively. Model selection and tuning details are described in Appendix C.

### 5.1 Word embedding models

We approach the problem as categorical classification and train a multi-layer perceptron (MLP) model to classify inputs consisting of word embeddings for the whole, part, and adjective words. The MLP takes as input the concatenation of these three word embeddings, obtained from GloVe (Pennington et al., 2014), and applies a single hidden layer with ReLU activation before the final softmax layer which predicts the class label.

### 5.2 Adjective projection as NLI

As we want to evaluate strong yet simple pre-existing language understanding models on this task, we now describe a method for obtaining the direct prediction described above via conversion to a form suitable for inference in the style of the SNLI and MNLI datasets (Bowman et al., 2015; Williams et al., 2018), which consist of *premise* and *hypothesis* sentence pairs. We first form simple hypothesis sentences from the tuples using the fixed template "The ⟨*whole*⟩'s ⟨*part*⟩ is ⟨*adjective*⟩." We then retrieve premise sentences that describe a ⟨*whole*⟩ ⟨*adjective*⟩. An example for (*bicycle, old*) is "He rode an old bicycle and brought fruits and vegetables home from Chinatown." We retrieve context sentences from five resources: Project Gutenberg books[2], the Gigaword news corpus (Parker et al., 2011), SNLI, MNLI, and MSCOCO image captions (Lin et al., 2014); premise sentence selection is described fully in Appendix D and examples are shown in Table 1.

### 5.3 Fine-tuning language models

We apply transfer learning from two recently developed large contextualized LMs to this task. Both are state-of-the-art on NLI and commonsense tasks.

Specifically, we test OpenAI GPT (Radford et al., 2018), and BERT (Devlin et al., 2018).

[2] https://gutenberg.org

OpenAI GPT is a unidirectional model that predicts the next word, while BERT is bidirectional and predicts randomly missing words, as well as the next sentence. Both train on the BooksCorpus, with BERT additionally trained on English Wikipedia. Both models are fine-tuned to perform NLI by applying a linear layer to the model's final output at one position of the input. The models are then trained in a multi-task way for the inference task and the language modeling objective(s), updating the whole network.

| Method | Accuracy |
|---|---|
| Majority baseline | 0.430 |
| Majority-per-part baseline | 0.485 |
| GloVe embeddings | 0.651 |
| OpenAI GPT (Radford et al., 2018) | 0.666 |
| BERT (Devlin et al., 2018) | 0.667 |
| Human performance | 0.785 |

Table 3: Test set results for multi-class prediction

Test set results for these methods are in Table 3. We also provide performance by two simple baselines, the first of which always predicts the majority class ("guaranteed"). To choose the second baseline, we evaluated choosing the majority class for the given whole, part, or adjective. Of these, predicting the majority-per-part had the best validation set performance, so we report that result on test.

We observe that the best model that operates on just word embeddings is within $\approx 0.02$ of both language models in absolute accuracy points, and the best performing model still lags behind human accuracy[3] by nearly 0.12 absolute points, suggesting work remains to be done on incorporating this variety of common sense into intelligent models.

## 6 Conclusion

Inspired by recent commonsense dataset construction efforts and the speed with which researchers develop highly performant models for them, we develop a dataset that evaluates a type of inference that is specific but that agents with commonsense should be able to solve. We show that state-of-the-art language models perform well, but that models using just pretrained word embeddings perform comparably, and both fall short of human accuracy. We release our dataset to provide a challenging commonsense reasoning task for the community.

[3]Measured using the triplet, not NLI, version of the data.


## Acknowledgements

Thanks to Sandeep Soni and Ian Stewart for helpful feedback on an initial draft. Thanks to Kevin Knight for initial inspiration and for the title. This work is supported by Contract W911NF-15-1-0543 with the US Defense Advanced Research Projects Agency (DARPA). Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the view of the sponsor.


## References


Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642. Association for Computational Linguistics.

Luana Bulat, Douwe Kiela, and Stephen Clark. 2016. Vision and feature norms: Improving automatic feature norm learning through cross-modal maps. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 579–88.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1657–68. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Luana Fagarasan, Eva Maria Vecchi, and Stephen Clark. 2015. From distributional semantics to feature norms: grounding semantic models in human perceptual data. In *Proceedings of the 11th International Conference on Computational Semantics*, pages 52–57.

Maxwell Forbes and Yejin Choi. 2017. Verb physics: Relative physical knowledge of actions and objects. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 266–276.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking NLI systems with sentences that require simple lexical inferences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 650–5. Association for Computational Linguistics.

Yoav Goldberg and Jon Orwant. 2013. A dataset of syntactic-ngrams over time from a very large corpus of English books. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, volume 1, pages 241–7.

Alicia Krebs, Alessandro Lenci, and Denis Paperno. 2018. Semeval-2018 task 10: Capturing discriminative attributes. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 732–740.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123(1):32–73.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision*, pages 740–55. Springer.

Li Lucy and Jon Gauthier. 2017. Are distributional representations ready for the real world? evaluating word vectors for grounded perceptual meaning. In *Proceedings of the First Workshop on Language Grounding for Robotics*, pages 76–85.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–49, San Diego, California. Association for Computational Linguistics.

Simon Ostermann, Michael Roth, Ashutosh Modi, Stefan Thater, and Manfred Pinkal. 2018. SemEval-2018 task 11: Machine comprehension using commonsense knowledge. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 747–757, New Orleans, Louisiana. Association for Computational Linguistics.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–55, Austin, Texas. Association for Computational Linguistics.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English Gigaword. *Linguistic Data Consortium*.

Ellie Pavlick and Chris Callison-Burch. 2016. Most babies are little and most problems are huge: Compositional entailment in adjective-nouns. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2164–2173.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–43.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–37. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. Preprint.

Marek Rei, Daniela Gerz, and Ivan Vulic. 2018. Scoring lexical entailment with a supervised directional similarity network. *arXiv preprint arXiv:1805.09355*.

Rachel Rudinger, Chandler May, and Benjamin Van Durme. 2017. Social Bias in Elicited Natural Language Inferences. In *The 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL): Workshop on Ethics in NLP*.

Roy Schwartz, Maarten Sap, Ioannis Konstas, Leila Zilles, Yejin Choi, and Noah A. Smith. 2017. The effect of different writing tasks on linguistic style: A case study of the ROC story cloze task. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 15–25, Vancouver, Canada. Association for Computational Linguistics.

Robert Speer, Joshua Chin, and Catherine Havasi. 2017. ConceptNet 5.5: An open multilingual graph of general knowledge. In *AAAI*, pages 4444–51.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Su Wang, Greg Durrett, and Katrin Erk. 2018. Modeling semantic plausibility by injecting world knowledge. In *Proceedings of the 2018 Conference of*

*the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 303–308.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. SWAG: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 93–104. Association for Computational Linguistics.

Sheng Zhang, Rachel Rudinger, Kevin Duh, and Benjamin Van Durme. 2017. Ordinal common-sense inference. *Transactions of the Association for Computational Linguistics*, 5:379–395.

# Aggregating Bidirectional Encoder Representations Using MatchLSTM for Sequence Matching

**Bo Shao**[†‡], **Yeyun Gong**[‡], **Weizhen Qi**[§‡], **Nan Duan**[‡], **Xiaola Lin**[†]
[†]Sun Yat-sen University
[‡]Microsoft Research Asia
[§]University of Science and Technology of China
shaobo2@mail2.sysu.edu.cn
{yegong, nanduan}@microsoft.com
weizhen@mail.ustc.edu.cn
linxl@mail.sysu.edu.cn

## Abstract

In this work, we propose an aggregation method to combine the **B**idirectional **E**ncoder **R**epresentations from **T**ransformer (BERT) with a MatchLSTM layer for Sequence Matching. Given a sentence pair, we extract the output representations of it from BERT. Then we extend BERT with a MatchLSTM layer to get further interaction of the sentence pair for sequence matching tasks. Taking natural language inference as an example, we split BERT output into two parts, which is from premise sentence and hypothesis sentence. At each position of the hypothesis sentence, both the weighted representation of the premise sentence and the representation of the current token are fed into LSTM. We jointly train the aggregation layer and pre-trained layer for sequence matching. We conduct an experiment on two publicly available datasets, WikiQA and SNLI. Experiments show that our model achieves significant improvement compared with state-of-the-art methods on both datasets.

## 1 Introduction

Text sequence matching aims to infer the relationship of two text sequences which is a critical problem used in many NLP tasks such as answer selection (Yang et al., 2015), natural language inference (Bowman et al., 2015), and paraphrase identification (Dolan et al., 2004).

Recent years, various deep learning models have been proposed for text sequence matching tasks (Wang and Jiang, 2016; Tay et al., 2018; Tymoshenko and Moschitti, 2018; Kim et al., 2018). Most previous works are learned from task specific supervised datasets, which are always limited to the amount, due to the cost of annotation. Recently, learning a general representation of words based on language models attracts great attention, and are successfully applied to a range of NLP tasks (McCann et al., 2017; Peters et al., 2018, 2017; Alec Radford, 2018; Devlin et al., 2018). In these works, they first learn general language representations through training a language model using large scale unlabeled data. Then they use these representations to the downstream tasks. Previous works show that the representations with linguistic information perform obvious effectiveness and robustness applied to downstream tasks.

In these methods, there are two typical strategies using pre-trained models, feature-based (Peters et al., 2017) and fine-tuning approach (Devlin et al., 2018; Radford et al., 2018). Our model is based on the fine-tuning approach and use BERT (Devlin et al., 2018) as our pre-trained language model. BERT has been directly used in many tasks and achieved significantly improvement compared with other methods. However, BERT only uses a standard softmax layer for many different specific tasks. Intuitively, for more complex tasks, like sequence matching in this work, we need a better strategy for extending the BERT to infer the relationship of the sequence pair. Thus, we incorporate a MatchLSTM layer (Wang and Jiang, 2015) to BERT, and make it better to adapt to sequence matching.

In our model, we design a MatchLSTM layer to aggregate representations of words in the sequences, and then we concatenate the hidden state of first token in BERT to the output of MatchLSTM (Wang and Jiang, 2015). This method can combine the features extracted from BERT and MatchLSTM effectively, and it is much better than directly using the outputs of BERT as pre-trained word representations to MatchLSTM.

The main contributions of this work are as follows:

- We extend BERT with a MatchLSTM layer. It can be applied to a variety of sequence matching tasks.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6059–6063,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

- We compare various aggregation layers to extend BERT, with the MatchLSTM layer achieving the best performance. It gets more powerful representations than the original output from BERT for sequence matching.

- Experiments on WikiQA and SNLI datasets show that our model achieves better performance than state-of-the-art methods.

## 2 Related Work

Sequence Matching task has attracted lots of attention in the past decades. There are many related works on Question Answering(QA) (Yin et al., 2015; Tay et al., 2018; Wang et al., 2017; Tymoshenko and Moschitti, 2018; Min et al., 2017), Natural Language Inference(NLI) (Peters et al., 2018; Kim et al., 2018; Radford et al., 2018) and so on. (Yin et al., 2015) use attention mechanism with convolutional layer to model sentence pairs. In (Tymoshenko and Moschitti, 2018), they combine the similarity features of members within the same pair and traditional sentence pair similarity and achieve state-of-the-art results on several answer selection datasets including WikiQA (Yang et al., 2015). (Peters et al., 2018) and (Radford et al., 2018) incorporate pre-trained language models to text sequence matching task and achieve performance improvement on SNLI (Bowman et al., 2015).

Recently, pre-trained language models have been successfully used in NLP tasks. ELMo (Peters et al., 2017) is trained as a bidirectional language model. OpenAI GPT (Alec Radford, 2018) uses a basic left-to-right transformer to learn a language model. BERT, used in this paper, is based on the architecture of a bidirectional Transformer and per-trained on Masked Language Model task and Next Sentence Prediction. Using the pre-trained parameters, BERT (Devlin et al., 2018) achieves state-of-the-art performance on the GLUE benchmark (Wang et al., 2018) and SQuAD 1.1 (Rajpurkar et al., 2016) by fine-tuning in corresponding supervised data.

In this work, we design a sequence matching model based on BERT.

## 3 Our Model

In this section, we will introduce our sequence matching model. Our model is combined BERT encoder with the MatchLSTM layer.

### 3.1 BERT Encoder

Given two natural language sentences $A$, $B$, the tokens of both sentences are packed into a token sequence $S$ as "[CLS] $A$ [SEP] $B$ [SEP]", where [CLS], [SEP] are special tokens. The representation of each token is the sum of three types of embedding: 1). **WordPiece embedding**(Wu et al., 2016) is used in BERT to represent the input sequence, which has 30000 token vocabulary. The tokens in $S$ are split into several word pieces with "##". 2). **Position embedding** is used to cover position information in transformer. 3). **Segment embedding** is used to indicate whether the current token is from $A$ or $B$.

Then the representation of $S$ is fed into the BERT encoder which is a multi-layer bidirectional Transformer(Vaswani et al., 2017). The implement of its architecture can refer to (Vaswani et al., 2017; Devlin et al., 2018). BERT encodes the input token sequence $S$ into a context aware representation $h = \{h_0, h_1, ..., h_{|S|-1}\}$. Then $h$ is used to a match layer for downstream supervised tasks.

For downstream tasks of text sequence matching, BERT extract the final hidden state of the first token, $h_0$, corresponding to the special token [CLS]. The label probabilities are computed with $h_0$ by a standard softmax layer. Finally, all of the parameters of the model are fine-tuned to maximize the log-probability of the correct label in specific tasks.

### 3.2 MatchLSTM Layer

Although BERT has proved its effectiveness for text sequence matching tasks, there still exist two limitations. First, the two sentences can only interact through a general attention mechanism in the transformer, which is not enough for complex matching tasks. Secondly, BERT only uses the first token hidden state in sequence matching tasks such as QA, MNLI in (Devlin et al., 2018), ignoring the final hidden state $h_i, i > 0$ in other positions. To overcome the two limitations, we add a bidirectional MatchLSTM network (Wang and Jiang, 2015), which computes the word-by-word matching results at each position for input sentence pairs.

To get the representation of each input sentence, we first split the final hidden state $h$ into $h^A$ and $h^B$ as the representations of the sentence pair, $A$ and $B$, according to its position information. Then we apply a bidirectional MatchLSTM net-

work with the input of $h^A$ and $h^B$. At each position $i$ of the tokens in sentence $B$, the MatchLSTM layer first uses a standard word-by-word attention mechanism to obtain attention weight $a^i = \{a_0^i, a_1^i, ..., a_{|A|-1}^i\}$ over sentence $A$ and compute a weighted sum of sentence $A$ as $c_i$, which can be formulated as,

$$e_j^i = u^T tanh(W^A h_j^A + W^B h_i^B + W^s s_{i-1} + b_e)$$
$$a_j^i = \frac{e_j^i}{\sum_{k=1}^{|p|} e_k^i}; c_i = \sum_{j=0}^{|A|} a_j^i h_j \tag{1}$$

where $u$, $W^A$,$W^B$,$W^s$ and $b_e$ are trainable parameters, $s_i$ is the hidden state at the $i$th position of $B$. The $i$,$j$ in Equation 1 represents $i$th word in $h^A$ and $j$th word in $h^B$, and $e_j^i$ is the attention score between the two words. We concatenate $c_i$ with the current token of sentence B as:

$$r_i = [h_i^B : c_i]$$
$$s_i = LSTM(r_i, s_{i-1}) \tag{2}$$

Then, we concatenate the last hidden state of MatchLSTM layer $s_{|B|}$ with the first hidden state $h_0$ of BERT encoder and feed it into the output softmax layer,

$$f = [s_{|B|} : h_0]$$
$$P = softmax(W^f f + b_f) \tag{3}$$

where $W^f$ and $b_f$ are trainable parameters, $P$ is the label probability distribution.

### 3.3 Loss Function

Finally we compute the loss function to maximize the log-probability of corresponding label $y \in \{0, 1, .., m-1\}$, $m$ is the number of target label types. It can be formulated as,

$$\mathcal{L} = -log(P_i)[i = y] \tag{4}$$

### 3.4 Other Aggregation Layers

In this section, we introduce two other general methods to utilize all the final hidden states of BERT encoder.

#### 3.4.1 Average Layer

To incorporate all the hidden states of BERT, a basic idea is to sum all final hidden states $h$ directly as the feature vector $f_{avg}$. It will be used to calculate the probability of final label by Eq.3.

#### 3.4.2 Convolutional Layer

We also evaluate the performance using the convolutional layer with max pooling. We set several fixed window sizes to extract feature vectors. For each windows size $l$, we use a kernel matrix $W^l \in R^{l \times d}$ and a non-linear function to operate on the hidden vector $h$ from BERT encoder. The output of one operation is a local feature which can be computed as follows:

$$z_i = g(W^l \cdot v[i : i + l] + b^l) \tag{5}$$

where $W^l$ and $b^l$ are trainable parameters, $g$ is a non-linear function. We conduct this operation on different hidden vectors $v_{1:l}, v_{2:l+1}, ..., v_{n-l+1:n}$ and get a set of local features $z = \{z_1, z_2, ..., z_{n-l+1}\}$. We then extract a maximum feature from the local features $z$ generated by one kernel through the max pooling layer and get feature vector $f_{cnn}$. We concatenate $f_{cnn}$ with the first hidden state $h_0$ as above to get the final feature vector and calculate the probability distribution of labels.

## 4 Experiment

We conduct experiments on the WikiQA and SNLI corpuses. We use Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR) as the evaluation metrics on WikiQA. On SNLI, we use accuracy(ACC) as our evaluation metric.

### 4.1 Datasets

WikiQA dataset (Yang et al., 2015) is a sentence-level QA dataset, containing 2.0k/0.3k/0.6k train/dev/test examples. Each sample contains a query in Bing's log and a snippet of a Wikipedia retrieved by Bing. Each query contains 18.6 candidate sentences on average. The task is to select an answer from candidate sentences for each query.

The Stanford Natural Language Inference (SNLI) dataset (Bowman et al., 2015) is a large-scale entailment classification task, containing 570K human annotated sentence pairs. Each sample consists of a sentence pair and a label from relationship class (entailment, neutral, contradiction, and -). We remove samples in "-" class same as previous works, since they do not have a consensus label among the annotators. For training, development and test sets, we use the same split with (Bowman et al., 2015) to report our results.

### 4.2 Settings

We use the pre-trained model of BERT$_{BASE}$ (Our goal is to prove the effectiveness of our aggregation method, so we do not use the BERT$_{LARGE}$ which is time and resource consuming), the number of Transformer blocks is 12, and dimension of all hidden state is 768 in our model. The batch size of the model is 32 and the initial learning rate is 2e-5. We set the dropout rate to 0.5.

Except for using state-of-the-art methods as our baseline, we also list variants of aggregated layers with BERT in the experiments. "BERT+AVG" is the model described in 3.4.1. "BERT+CNN" is the model proposed in 3.4.2. "BERT" represents the BERT with pre-trained parameters as initialization. "BERT$_{no\ init}$" denotes BERT using random parameters as initialization. The "-FHS" means the model does not use the hidden state of first token in BERT encoder.

### 4.3 Experiments on WikiQA

| Methods | MAP | MRR |
|---|---|---|
| ABCNN (Yin et al., 2015) | 69.21 | 71.08 |
| MULT (Wang and Jiang, 2016) | 74.33 | 75.45 |
| HyperQA (Tay et al., 2018) | 72.70 | 71.20 |
| BiMPM (Wang et al., 2017) | 71.80 | 73.10 |
| PTK (Tymoshenko and Moschitti, 2018) | 73.34 | 74.68 |
| SQuAD* (Min et al., 2017) | 83.20 | 84.58 |
| BERT$_{no\ init}$ | 58.91 | 60.09 |
| BERT (Devlin et al., 2018) | 80.15 | 82.31 |
| BERT+AVG | 81.95 | 83.26 |
| BERT+CNN | 81.23 | 83.16 |
| **BERT+MatchLSTM** | **83.53** | **85.13** |
| -FHS | 70.46 | 72.44 |

Table 1: Performance for answer sentence selection on WikiQA dataset. Results of PTK are ensemble results and SQuAD* is trained with extra supervised corpus, SQuAD.

We first study the effectiveness of our method for answering on the WikiQA dataset. Our model achieves new state-of-the-art performance compared with previous methods. First, comparing "BERT+MatchLSTM" with "BERT", we find that "BERT+MatchLSTM" achieves significant improvement, which illustrates the effectiveness of our method using a MatchLSTM layer to extend "BERT". Moreover, from the results of "BERT+MatchLSTM", "BERT+AVG" and "BERT+CNN", we find that the MatchLSTM layer utilizes the output representations from the BERT encoder more efficiently than "AVG" and "CNN". We also see that "BERT$_{no\ init}$" does not perform well on this task, which shows that the architecture of BERT is not superior to existing architectures, thus using other architectures to boost the pre-trained BERT is meaningful and important. From the result of "-FHS" we find that the performance drops a lot, which illustrates the method we used to concatenate the hidden state of first token in BERT to the output of MatchLSTM is very important.

### 4.4 Experiments on SNLI

| SNLI | train | test |
|---|---|---|
| SLRC (Zhang et al., 2018) | 89.1 | 89.1 |
| LMPT (Radford et al., 2018) | 96.6 | 89.9 |
| ESIM+ELMo (Peters et al., 2018) | 91.6 | 88.7 |
| +Ensemble | 92.1 | 89.3 |
| DRCN (Kim et al., 2018) | 93.1 | 88.9 |
| +Ensemble | 95.0 | 90.1 |
| BERT$_{no\ init}$ | 90.7 | 82.5 |
| BERT (Devlin et al., 2018) | 95.9 | 89.6 |
| BERT+AVG | 96.2 | 90.1 |
| BERT+CNN | 94.7 | 90.2 |
| **BERT+MatchLSTM** | **96.8** | **90.9** |
| -FHS | 91.4 | 88.3 |

Table 2: Accuracy on SNLI dataset. All compared results are from the SNLI Leaderboard.

Table 2 shows the comparisons of our method with the state-of-the-art methods on the SNLI dataset. The trends observed in this experiment are consistent with the former one. Compared with baseline methods, our "BERT+MatchLSTM" achieves the best performance on both training and test sets. Furthermore, compared with "BERT", "BERT+AVG" and "BERT+CNN", our model shows obvious superiority. Comparing the results of "BERT" and "BERT$_{no\ init}$", we also observe that the pre-training process is significantly helpful in this task.

## 5 Conclusion

In this work, we design an aggregation method to combine BERT with a MatchLSTM layer for sequence matching. We show that our model is more effective in computing the interaction features with the BERT encoder than the original BERT model. Experiments show that our method achieves new state-of-the-art performance on WikiQA and SNLI datasets.

## Acknowledgements

This work is supported by the National Natural Science Foundation of China under Grants No.U1711263.

## References


Tim Salimans andIlya Sutskever Alec Radford, Karthik Narasimhan. 2018. Improving language understanding with unsupervised learning. *Technical report, OpenAI*.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th international conference on Computational Linguistics*, page 350. Association for Computational Linguistics.

Seonhoon Kim, Jin-Hyuk Hong, Inho Kang, and Nojun Kwak. 2018. Semantic sentence matching with densely-connected recurrent and co-attentive information. *arXiv preprint arXiv:1805.11360*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6294–6305.

Sewon Min, Minjoon Seo, and Hannaneh Hajishirzi. 2017. Question answering through transfer learning from large fine-grained supervision data. *arXiv preprint arXiv:1702.02171*.

Matthew E Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. *arXiv preprint arXiv:1705.00108*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/language-unsupervised/language_ understanding_paper. pdf*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Hyperbolic representation learning for fast and efficient neural question answering. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 583–591. ACM.

Kateryna Tymoshenko and Alessandro Moschitti. 2018. Cross-pair text representations for answer sentence selection. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2162–2173.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Alex Wang, Amapreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Shuohang Wang and Jing Jiang. 2015. Learning natural language inference with lstm. *arXiv preprint arXiv:1512.08849*.

Shuohang Wang and Jing Jiang. 2016. A compare-aggregate model for matching text sequences. *arXiv preprint arXiv:1611.01747*.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. *arXiv preprint arXiv:1702.03814*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2015. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *arXiv preprint arXiv:1512.05193*.

Zhuosheng Zhang, Yuwei Wu, Zuchao Li, Shexia He, Hai Zhao, Xi Zhou, and Xiang Zhou. 2018. I know what you want: Semantic learning for text comprehension. *arXiv preprint arXiv:1809.02794*.

# What Does This Word Mean?
# Explaining Contextualized Embeddings with Natural Language Definition

**Ting-Yun Chang** **Yun-Nung Chen**
Department of Computer Science and Information Engineering
National Taiwan University, Taipei, Taiwan
r06922168@ntu.edu.tw y.v.chen@ieee.org

## Abstract

Contextualized word embeddings have boosted many NLP tasks compared with traditional static word embeddings. However, the word with a specific sense may have different contextualized embeddings due to its various contexts. To further investigate what contextualized word embeddings capture, this paper analyzes whether they can indicate the corresponding sense definitions and proposes a general framework that is capable of explaining word meanings given contextualized word embeddings for better interpretation. The experiments show that both ELMo and BERT embeddings can be well interpreted via a readable textual form, and the findings may benefit the research community for a better understanding of what the embeddings capture[1].

## 1 Introduction

Contextualized word embeddings, such as ELMo, BERT, and OpenAI GPT, GPT-2 (Peters et al., 2018; Devlin et al., 2018; Radford et al., a,b) have been shown to yield richer representations of meaning and boosted many NLP tasks. To understand what contextualized word embeddings capture, Schuster et al. (2019) recently visualized the representations of ELMo and showed that 1) embeddings of the same word in different contexts can form a cluster, and 2) when a word has multiple senses, the embeddings can be separated into multiple distinct groups, one for each meaning. To further investigate the meanings contextualized word embeddings indicate, this paper focuses on analyzing whether a contextualized embedding is sense-informative enough to indicate the corresponding sense definition given a (target word, context) pair. We train and evaluate our model on the online Oxford dictionary dataset released by Chang et al. (2018).

To analyze if the embeddings are sense-informative, our work focuses on learning a mapping between the semantic space of contextualized word embeddings and the space of definition embeddings. Specifically, a better mapping indicates richer sense-specific cues in the contextualized word embedding.

Different from the definition modeling in the prior work (Noraset et al., 2017; Gadetsky et al., 2018; Chang et al., 2018), we reformulate the task from natural language generation (NLG) to classification, i.e., selecting the most reasonable definition according to the target word and its contexts. As recent work has shown the great success in encoding lexical resources into consistent representations (Tissier et al., 2017; Bahdanau et al., 2017; Bosc and Vincent, 2018), in this paper, we leverage pretrained sentence encoder (Cer et al., 2018) to project all definitions in the Oxford dictionary to a consistent embedding space, supporting our reformulation which requires to learn a mapping transforming from the contextualized word embedding space to the definition embedding space. Therefore, we can avoid some predicaments in NLG, such as troubles in generating fluent sequences, the exposure bias problem (Ranzato et al., 2015) and the difficulties in evaluation (Stent et al., 2005).

## 2 Methodology

The goal of this paper is to analyze whether we can distill sense-specific information from the pretrained contextualized word embeddings such as ELMo and BERT for better interpretation. Specifically, given the embedding of a (word, context) pair, our model learns a non-linear mapping network $f : X \rightarrow Y$ to project it into the desired

[1]The source code and the trained models are publicly available at https://github.com/MiuLab/GenDef.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6064–6070,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

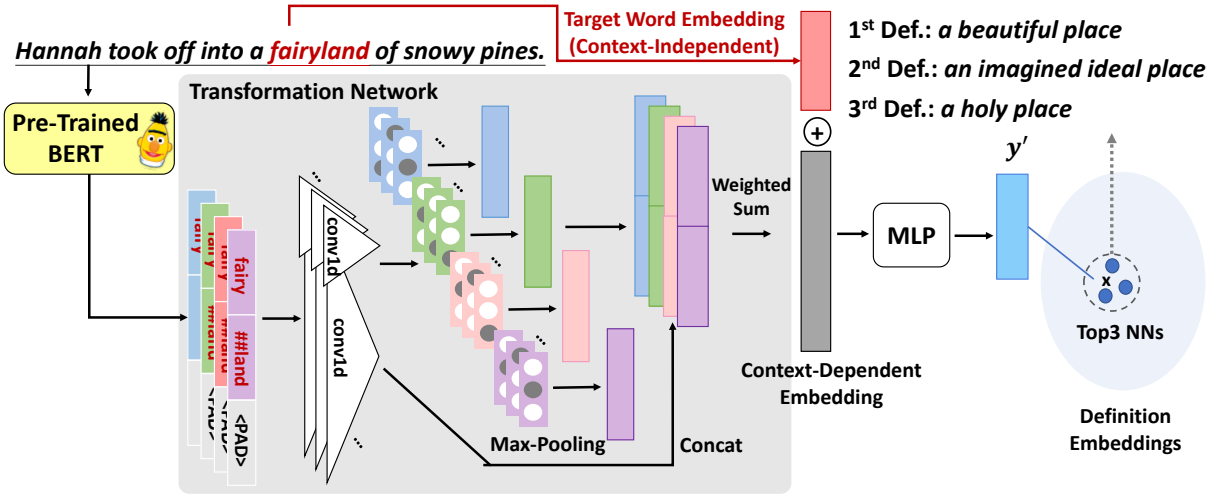


Figure 1: Illustration of the proposed model using contextualized embeddings from BERT as the context-dependent component. We use the one-dimensional convolution with its kernel sizes being 1 and 3.

definition space. Note that the mapping is many-to-one intrinsically because there exist some examples sharing the same definition, and their target words or contexts differ. This paper assumes that *contextualized word embeddings can be easily translated into their corresponding definitions if their semantics is well captured in the representations*. To validate the assumption, the following models are proposed for mapping the embeddings.

### 2.1 Model Objective

The whole framework of our proposed model is illustrated in Figure 1. Our goal is to learn the mapping that transforms contextualized word embeddings into their corresponding definition, which is to solve:

$$f^\star = \arg\min_f \|f(\mathbf{x}) - \mathbf{y}\|_2, \qquad (1)$$

where $\mathbf{x} = (\mathbf{u}, \mathbf{v}) \in X$ consists of the target word embedding $\mathbf{u}$, which is context-independent and serving as the target word identity, and a context-dependent component $\mathbf{v}$, which could be either the context embedding or the contextualized target word embedding; $\mathbf{y} \in Y$ is the embedding of the corresponding definition. Both the definition embedding and the context embedding are encoded by the pretrained transformer-based universal sentence encoder (Cer et al., 2018), and we utilize different types of contextualized word embeddings, such as ELMo, BERT-base, and BERT-large, by swapping the context-dependent component shown in the left part of Figure 1.

Note that though we model this task as a classification problem, we do not train a classifier that regards all definitions as discrete labels but learn a translation between two representation spaces, motivated by Lample et al. (2017, 2018), because different definitions may have semantic similarities. We first encode all ground truth definitions to a consistent embedding space and learn a mapping function $f$. During the inference stage, given a target word and its context, we retrieve the corresponding human-readable definitions of the top-$k$ nearest neighbors of our predicted embedding in the definition embedding space, with consideration of the whole 79,030 definition candidates in the Oxford dictionary. Note that the candidates for each word are not restricted to its existing definitions in the dictionary, considering that words may go through semantic shift, such as the word *gay* has shifted its meaning from *happy* to *homosexual*.

### 2.2 Mapping Architecture

Our mapping architecture consists of a transformation network followed by the 7-layer MLP, with batch normalization (Ioffe and Szegedy, 2015) and ReLU. In order to incorporate diverse context-dependent embeddings (such as ELMo and BERT), different transformation nets are proposed, whose common goal is to map the input features to a fixed-dimension representation. Three variants are described in detail.

- **Context Embedding**: the target word embedding and its context embedding encoded

by a pretrained sentence encoder are concatenated as the input to the transformation net, which is simply implemented as a fully-connected layer with the ReLU activation.

- **ELMo**: we apply a weighted sum over 3 extracted contextualized word embeddings, i.e., the output of character CNN and two LSTMs, getting a single context-dependent vector for concatenating with the target word vector.

- **BERT**: the target word is tokenized into word pieces, and we use one-dimensional convolution (conv1d) (Kim, 2014) and max-pooling to tackle the variable-lengthed features. We extract the last 4 layers from BERT and jointly learn softmax-normalized weights corresponding to each layer similar to ELMo.[2] Figure 1 illustrates the mapping model leveraging features from BERT, which expresses the best capability of carrying sense-specific explanation among all variants.

### 2.3 Reverse Mapping

In order to analyze what the mapping captures for better interpretation, we examine the reverse direction of our mapping after training, motivated by Yuan et al. (2016). Given a context-dependent embedding $\mathbf{v}$ and its ground truth definition embedding $\mathbf{y}$, also the word embedding $\mathbf{u}_w$ for each target word $w$ in the vocabulary $V$, and a pretrained mapping $\bar{f}$, the word that is the closest vector to the target definition after mapping is formulated as:

$$\arg\max_{w \in V} \cos(\bar{f}(\mathbf{u}_w, \mathbf{v}), \mathbf{y}). \quad (2)$$

In our experiments, the word set corresponding to the top-$k$ highest cosine scores often contains the actual target word, also overlapping with a few synonyms provided by the Oxford dictionary. When applying to contextualized word embeddings from BERT-base, the model achieves the average recall of 23.7%, even though we do not incorporate any synonym information during training. This demonstrates that our models are capable of automatically capturing potential similarities. Examples of the generated synonyms can be found in the supplementary material.

[2]Note that our work focuses on analyzing the sense information encoded in the contextualized embeddings; thus, our model is stacked upon the frozen representations extracted from the pretrained BERT instead of fine-tuning them. More training details are given in the supplementary material.

## 3 Experiments

In order to examine whether the sense-specific information captured by contextualized word embeddings can be well disentangled, the following experiments are conducted.

### 3.1 Definition Retrieval

This is to analyze whether our proposed mapping indeed interprets the sense-specific definitions from contextualized word embeddings. Considering that the words can be seen and unseen, our experiments contain two levels of tasks (Chang et al., 2018).

- **Seen** is to test the pair with *(seen word, unseen context, seen definition)*, including 151,306 pairs of instances containing 9,276 target words.

- **Unseen** is to test the pair with *(unseen word, unseen context)*, including 15,959 pairs of instances corresponding to the 1,000 randomly selected target words held-out from training. Such a **zero-shot** setting challenges if the input feature is informative enough and if the mapping can generalize to the unseen but semantically consistent embeddings. Also, it is a practical and appealing task as many new words are coined every year.

We ensure both being polysemic tasks by sampling within instances whose target words have at least 3 definitions when building these two test sets.

#### 3.1.1 Results

We measure the performance of various proposed architectures with average precision (@1, @5, @10) as well as the average cosine distance between the predicted definition embedding and the ground truth embedding (lower is better). Two baselines without using contextualized embeddings as context-dependent input features are proposed, 1) using the target word embedding only, which entirely ignores the contexts and thus being a lower bound of this task and 2) leveraging the context embedding from the pretrained Transformer-based universal-sentence encoder as

| Task | Methods | P@1 | P@5 | P@10 | Cosine Dist |
|---|---|---|---|---|---|
| **Seen** | target word embedding | 33.27 / 18.29 | 50.77 / 31.92 | 56.06 / 36.69 | 0.251 |
| | + context embedding | 59.36 / 45.19 | 71.43 / 58.17 | 74.95 / 62.42 | 0.178 |
| | + ELMo | 67.00 / 53.91 | 77.32 / 65.69 | 80.35 / 69.58 | 0.149 |
| | + BERT-base | **74.83 / 63.34** | **83.28 / 73.97** | **85.46 / 77.06** | **0.123** |
| | + BERT-large | 73.89 / 62.36 | 82.61 / 73.24 | 84.92 / 76.28 | 0.126 |
| **Zero-Shot** | target word embedding | 1.84 / 1.06 | 6.54 / 4.22 | 9.67 / 6.44 | 0.388 |
| | + context embedding | 1.97 / 1.29 | 7.00 / 4.77 | 10.78 / 7.50 | 0.383 |
| | + ELMo | 2.04 / 1.38 | 6.79 / 4.65 | 10.21 / 7.06 | 0.387 |
| | + BERT-base | 3.27 / 2.28 | 9.59 / 7.41 | 14.44 / 11.44 | 0.344 |
| | + BERT-large | **3.50 / 2.52** | **10.47 / 8.17** | **15.58 / 12.35** | **0.339** |

Table 1: Precision@K (average within examples sharing the same target words / average within examples sharing the same (target word, definition)) and cosine distance for models using various input features.

| Methods | Tasks | |
|---|---|---|
| | Seen | Unseen |
| Noraset et al. (2017) | 21.6 / 36.7 | 1.7 / 15.8 |
| Chang et al. (2018) | 24.9 / 41.0 | 2.0 / 15.9 |
| target word embedding | 28.4 / 36.9 | 4.6 / 17.2 |
| + context embedding | 58.5 / 62.8 | 5.1 / 16.8 |
| + ELMo | 66.5 / 71.6 | 4.8 / 17.2 |
| + BERT-base | **74.7 / 78.3** | **7.1 / 19.3** |

Table 2: BLEU@4 / ROUGE-L:F scores of NLG-based models and various proposed architectures.

described in Section 2.2. Note that the naive baseline is randomly guessing among the whole 79,030 definitions, with P@1 lower than 0.0013%, showing the difficulty of this task.

For **Seen** experiments, Table 1 shows that the context-dependent component contains abundant sense-informative cues, where contextualized word embeddings, especially BERT, expresses the strong capability of producing corresponding definitions with about 15% enhancement of P@1 comparing to the second baseline. For **Unseen** results, the trend is similar: all models with context-dependent input features outperform the first baseline, and the variants with BERT reach the best scores among all metrics. The above results demonstrate rich sense-informative cues captured by the contextualized word embeddings.

Furthermore, we evaluate the definitions by their natural language surfaces using BLEU (Papineni et al., 2002) and ROUGE-L (Lin, 2004) scores. The results are in Table 2, where both Noraset et al. (2017) and the first proposed baseline generates or selects definitions depending merely on the static target word embedding, and all other architectures are context-dependent. We initialize the target word embeddings for all models with pretrained `fasttext` (Joulin et al., 2016) on Wikipedia 2017, UMBC webbase corpus for a fair comparison. The results demonstrate that our mapping model can better explain the word representations than the prior work.

#### 3.1.2 Analysis

An ablation study is conducted in which only the contextualized word embedding is used. While the scores of all 3 related variants drop dramatically due to the lack of the explicit, context-independent signal to the target word identity, the scores still outperform the first baseline by 7% (P@1) on the **Seen** task, showing the superiority of contextualized representations to their static counterparts. In addition, the reason about better performance of the models with contextualized embeddings compared to the one with context embeddings (the second baseline) is that two context embeddings sharing the same target word sense may differ a lot due to various words in the contexts, but they may have similar contextualized word embeddings produced from ELMo or BERT. This allows our proposed model to better interpret the sense information.

Despite the overall low performance under the zero-shot setting, it is found that all proposed models with context-dependent components are still able to disambiguate different senses. Table 3 shows a randomly-sampled example from the output of the BERT-base model. Although the model can only correctly answer the first definition of *draw*, which may be the most common usage so that it can be easily generated from the input embedding, we show that our model is still able to capture the other two very different word senses

| Target | Contexts, Selected Definitions, Ground Truth |
|---|---|
| draw | The embodied capacity to write and *draw* seems to rule over the languid group of objects underneath.<br>**1st Definition**: produce an image of someone or something by making lines and marks on paper<br>**2nd Definition**: produce a picture or diagram by making lines and marks on paper with a pencil pen etc<br>**3rd Definition**: compose or draw up something written or abstract<br>**Ground Truth**: produce an image of someone or something by making lines and marks on paper |
| | When it came to the end of the day, though, I was more than happy to *draw* the curtains and shut the day out.<br>**1st Definition**: arrange something carefully into a particular shape or position<br>**2nd Definition**: arrange objects or parts in a zigzag formation or so that they are not in line<br>**3rd Definition**: draw a circle round something especially to focus attention on it<br>**Ground Truth**: pull curtains shut or open |
| | ... hinders your ability to impart spin on the ball, reducing your ability to *draw* and fade the shot on command.<br>**1st Definition**: put the ball in play by throwing it up between two opponents<br>**2nd Definition**: strike the ball in the direction of ones followthrough so that it travels to the left ...<br>**3rd Definition**: propel a ball with a bat racket stick etc to score runs or points in a game<br>**Ground Truth**: hit the ball so that it deviates slightly usually as a result of spin |

Table 3: The analysis of the top 3 selected definitions on the **Unseen** task.

| Model | Accuracy (%) |
|---|---|
| Lee and Chen (2017) | 52.14 |
| Neelakantan et al. (2015) | 54.00 |
| Mancini et al. (2016) | 54.56 |
| Guo et al. (2019) | 55.27 |
| Chang et al. (2018) | 57.00 |
| Pilehvar and Collier (2016) | 58.55 |
| Proposed (BERT-base) | **68.64** |

Table 4: The results on Word-in-Context (WiC) data.

according to the selected definitions. More output samples on both **Seen** and **Unseen** tasks can be found in the supplementary material.

Moreover, unlike the prior work that required discrete token generation to interpret the pretrained word embeddings (Noraset et al., 2017; Gadetsky et al., 2018; Chang et al., 2018), we reformulate the definition modeling task from an NLG problem to a classification problem via learning a mapping between two semantically continuous spaces, which greatly simplifies the hardness, making significant improvement as shown in Table 2. Specifically, as the input representations of Noraset et al. (2017) and the first proposed baseline are the same, i.e., they both are context-agnostic and utilize the same pretrained static word embeddings, the better performance of our model demonstrates the direct benefits of not requiring sequence generation.

### 3.2 Word Sense Selection in Context

We further examine if the captured sense-specific cues help word sense disambiguation via Word-in-Context data (WiC) (Pilehvar and Camacho-Collados, 2018), in which each instance contains a pair of two contexts sharing a target word, and the task is to decide whether their word senses are the same.[3] To justify that the models are capable of selecting senses encoded in the embeddings, for each pair, our model outputs 10 candidate definitions (top-10 nearest neighbors), and we output TRUE if any definition occurs in both candidate sets, otherwise FALSE. Table 4 shows that the proposed model with contextualized word embeddings outperforms all previous models. We conclude that contextualized word embeddings indeed capture sense-informative cues and our proposed model is capable of interpreting the corresponding senses via definition.

## 4 Conclusion

This paper proposes a framework that can well interpret the contextualized word embeddings by human-readable sense definitions. The experiments demonstrate that contextualized word embeddings capture the sense-informative cues and the proposed model can better explain the semantics encoded in the representations.


## Acknowledgements

We would like to thank Ta-Chung Chi for in-depth discussions and anonymous reviewers for their insightful comments on the paper. This work was financially supported from the Young Scholar Fellowship Program by Ministry of Science and Technology (MOST) in Taiwan, under Grant 108-2636-E-002-003.


[3] To analyze the generalizability, we follow the experimental setting of Guo et al. (2019), where all baselines and the proposed model are pretrained and evaluated directly on the large WiC training set without fine-tuning.

## References


Dzmitry Bahdanau, Tom Bosc, Stanisław Jastrzebski, Edward Grefenstette, Pascal Vincent, and Yoshua Bengio. 2017. Learning to compute word embeddings on the fly. *arXiv preprint arXiv:1706.00286*.

Tom Bosc and Pascal Vincent. 2018. Auto-encoding dictionary definitions into consistent word embeddings. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1522–1532.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Ting-Yun Chang, Ta-Chung Chi, Shang-Chi Tsai, and Yun-Nung Chen. 2018. xSense: Learning sense-separated sparse representations and textual definitions for explainable word sense networks. *arXiv preprint https://arxiv.org/abs/1809.03348*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Artyom Gadetsky, Ilya Yakubovskiy, and Dmitry Vetrov. 2018. Conditional generators of words definitions. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 266–271.

Fenfei Guo, Mohit Iyyer, Leah Findlater, and Jordan Boyd-Graber. 2019. A differentiable self-disambiguated sense embedding model via scaled gumbel softmax.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. 2016. Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2017. Unsupervised machine translation using monolingual corpora only. *arXiv preprint arXiv:1711.00043*.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Phrase-based & neural unsupervised machine translation. *arXiv preprint arXiv:1804.07755*.

Guang-He Lee and Yun-Nung Chen. 2017. MUSE: Modularizing unsupervised sense embeddings. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 327–337.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Massimiliano Mancini, Jose Camacho-Collados, Ignacio Iacobacci, and Roberto Navigli. 2016. Embedding words and senses together via joint knowledge-enhanced training. *arXiv preprint arXiv:1612.02703*.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2015. Efficient non-parametric estimation of multiple embeddings per word in vector space. *arXiv preprint arXiv:1504.06654*.

Thanapon Noraset, Chen Liang, Larry Birnbaum, and Doug Downey. 2017. Definition modeling: Learning to define word embeddings in natural language. In *Proceedings of AAAI*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237.

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2018. WiC: 10,000 example pairs for evaluating context-sensitive representations. *arXiv preprint arXiv:1808.09121*.

Mohammad Taher Pilehvar and Nigel Collier. 2016. De-conflated semantic representations. *arXiv preprint arXiv:1608.01961*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. a. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. b. Language models are unsupervised multitask learners.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

Tal Schuster, Ori Ram, Regina Barzilay, and Amir Globerson. 2019. Cross-lingual alignment of contextual word embeddings, with applications to zero-shot dependency parsing. *arXiv preprint arXiv:1902.09492*.

Amanda Stent, Matthew Marge, and Mohit Singhai. 2005. Evaluating evaluation methods for generation in the presence of variation. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 341–351. Springer.

Julien Tissier, Christopher Gravier, and Amaury Habrard. 2017. Dict2vec: Learning word embeddings using lexical dictionaries. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 254–263.

Dayu Yuan, Julian Richardson, Ryan Doherty, Colin Evans, and Eric Altendorf. 2016. Semi-supervised word sense disambiguation with neural models. *arXiv preprint arXiv:1603.07012*.

# Pre-Training BERT on Domain Resources for Short Answer Grading

**Chul Sung**[*]
IBM Watson Education, USA
daniel.c.sung@gmail.com

**Tejas Indulal Dhamecha**
IBM Research, India
tidhamecha@in.ibm.com

**Swarnadeep Saha**[†]
UNC Chapel Hill, USA
swarna@cs.unc.edu

**Tengfei Ma**
IBM Research AI, USA
tengfei.ma1@ibm.com

**Vinay Reddy**
IBM Watson Education, USA
vinay.kasireddy@ibm.com

**Rishi Arora**
IBM GBS, India
arishi@in.ibm.com

## Abstract

Pre-trained BERT contextualized representations have achieved state-of-the-art results on multiple downstream NLP tasks by fine-tuning with task-specific data. While there has been a lot of focus on task-specific fine-tuning, there has been limited work on improving the pre-trained representations. In this paper, we explore ways of improving the pre-trained contextual representations for the task of automatic short answer grading, a critical component of intelligent tutoring systems. We show that the pre-trained BERT model can be improved by augmenting data from the domain-specific resources like textbooks. We also present a new approach to use labeled short answering grading data for further enhancement of the language model. Empirical evaluation on multi-domain datasets shows that task-specific fine-tuning on the enhanced pre-trained language model achieves superior performance for short answer grading.

## 1 Introduction

Intelligent tutoring system (ITS) is one of the tools to facilitate personalized learning. Automatic short answer grading (ASAG) is an important component of a dialog-based tutoring (DBT) system; especially, to enable Socratic tutoring. Automatic grading is the task of evaluating the correctness of a student answer for a specific question by comparing it to a reference answer. In short answer grading, the reference answers are typically one or two sentences long and close ended. A broad range of approaches from simple bag-of-words to transfer learning and deep neural networks have been explored to address the short answer grading problem (Mueller and Thyagarajan, 2016; Saha et al., 2018; Marvaniya et al., 2018).

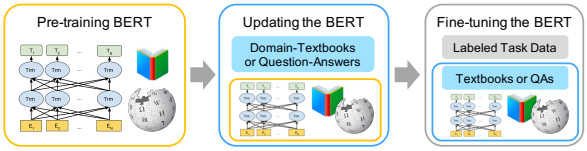


Figure 1: In this work, we propose to update the pre-trained BERT language model by utilizing the textbook or question-answer data to improve short answer grading.

For a variety of natural language processing (NLP) tasks, state-of-the-art results have been reported with pre-trained deep language models, such as BERT (Devlin et al., 2018), GPT (Radford et al., 2018), and ULMFit (Howard and Ruder, 2018). In these approaches, pre-trained language models are utilized for down-stream NLP tasks by means of task-specific *fine-tuning*.

A typical DBT, and therefore a short answer grading system, is often used across various subjects or domains (e.g., Science, Sociology, and Psychology). A pre-trained language model, such as that of BERT, is typically trained on generic English language corpus. Thus, there may be a scope for updating and improving the pre-trained language model on available textual resources for the domains of short answer grading. Textbooks used to create questions and reference answers are mostly available in digital form. Moreover, the labeled data for short answer grading have question and reference answer pairs available but contextual information between the pairs and even just questions are easily ignored. Thus, in this research, we aim to explore and evaluate various methods of updating the pre-trained BERT language model (LM)[*] on such domain-specific

[*]Corresponding author.

[†]The work was done when the author was an employee at IBM Research, India.

[*]We prefer the phrase *to update LM* to imply further training or modifying the pre-trained LM. In some literature, the term *LM fine-tuning* is also used to refer to the update of pre-trained language model. However, we reserve the term *fine-tuning* for task-specific classifier training in sync with BERT terminology.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6071–6075,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

available resources in the context of ASAG to answer the following research questions.

RQ1 Is updating pre-trained BERT LM helpful in improving short answer grading performance?

RQ2 What is the effect of unsupervised domain corpora (i.e. domain textbooks) in updating LM for short answer grading?

RQ3 How generalizable are the pre-trained and the updated BERT models to unseen domains, in case, where textbooks are not available?

RQ4 How can labeled Question-Answer data be exploited to update LM for short answer grading, in addition to fine-tuning?

Our evaluations are performed on four subjects (Physiology, American Government, and two on Psychology). In addition to the empirical analysis, we also propose a novel approach to effectively utilize the question-answer data as part of the pre-trained model update.

## 2 Related Work

The problem of short answer grading has attracted significant attention of the researchers over the years. Various approaches, starting from traditional hand-crafted features (Mohler et al., 2011; Sultan et al., 2016) to more recent deep learning models (Riordan et al., 2017; Kumar et al., 2017) and their combination (Saha et al., 2018) have been explored. However, similar to most downstream NLP tasks, ASAG also suffers from the overhead of task-specific architectures and thus scalability across different subjects has proven to be hard.

In a step towards alleviating this overhead, the NLP community has recently proposed multiple generic pre-trained language models which can be transferred seamlessly and fine-tuned for any end task. Universal Language Model Fine-Tuning (ULMFiT) (Howard and Ruder, 2018) method is one of the first such initiatives to illustrate the effectiveness of language model fine-tuning. Embeddings from Language Models, commonly referred to as ELMo (Peters et al., 2018) also learns deep contextualized word representations using the internal states of a deep bidirectional language model. Finally, Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018) improves on all previous pre-training techniques by training a deep language model that jointly conditions on both the left and right context simultaneously through all the layers. BERT's effectiveness has been widespread and Sung *et al.* (Sung et al., 2019) have shown that fine-tuning BERT for ASAG also outperforms all existing techniques.

While BERT has been fine-tuned to achieve state-of-the-art results on a large number of tasks, the idea of further pre-training of the language model to incorporate more domain knowledge has been explored less. BioBERT (Lee et al., 2019) for biomedical tasks and SciBERT (Beltagy et al., 2019) for science domains have shown the effectiveness of language model pre-training for tasks in specific domains. Motivated by these prior works, we propose two pre-training techniques for BERT in the context of short answer grading that improve over the generic BERT.

## 3 Method

Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018) is a deep bidirectional pre-trained language model that is fine-tuned for downstream NLP tasks. As illustrated in Figure 1, we describe the usage of BERT in two parts: firstly the proposed enhancements in the pre-training step and secondly, the fine-tuning step for short answer grading.

### 3.1 Pre-Training BERT

We start with the pre-trained BERT language model, trained using the English Wikipedia and BooksCorpus and propose two methods to further improve it.

#### 3.1.1 Usage of Textbooks

Our first approach relies on the usage of textbooks from specific domains of short answer grading. Specifically, we collect textbooks corresponding to the domains and chunk them into paragraphs and feed each paragraph as a document for pre-training. Since our task at hand is short answer grading, we assume that the answers to questions do not overlap between paragraphs and thus we treat each paragraph as an independent document.

To validate our assumption, we randomly sampled 60 question-answer pairs from Physiology domain, and manually examined these whether the answer is contained in the same paragraph of the question. We observe that for about 90% of these

samples, that is indeed the case. Further details on the textbooks and their pre-processing steps for data preparation is provided in the experiments section.

#### 3.1.2 Usage of Question-Answer Pairs

Our second approach leverages the labeled (question, reference answer, student answer) data triples as unsupervised data for pre-training the language model. We consider the triples with correct labels only and create pairs of the following form for each of the correct student answers.

| Question-Reference Answer pair |
|---|
| What does the telencephalon contain?<br>The telencephalon contains most of the cerebral cortex. |
| **Question-Correct Answer pair** |
| What does the telencephalon contain?<br>It contains the cerebral cortex, limbic system, and basal ganglia. |

Each of these pairs is fed as documents to the BERT architecture. Since one of the training objectives for BERT is next sentence prediction, the answer in a (question, answer) pair provides an immediate context to the question. Incorrect and partially correct student answers, apart from being factually incomplete, are often grammatically incorrect and thus may harm the language model learning. Hence, we ignore those triples for pre-training.

### 3.2 Fine-tuning for ASAG

The BERT fine-tuning step using labeled short answer grading data proceeds similar to any sentence pair classification task. The (reference answer, student answer) pair is converted to a single sequence of tokens by using a separator token `[SEP]` between the pair and a classification token `[CLS]` at the beginning. The input pair's representation, as obtained from the embedding of the `[CLS]` token, is then fed into a dense layer, which along with the language model is updated during fine-tuning.

## 4 Experiments

The experiments section is organized as follows. We start by providing a brief description of the dataset, followed by a discussion about the implementation details. In the concluding two subsections, we analyze the effect of the two proposed pre-training methodologies for short answer grading.

| Textbook | Train | | | Test | | |
|---|---|---|---|---|---|---|
| | Correct | Incorrect | Partial | Correct | Incorrect | Partial |
| Phy | 9,676 | 1,197 | 2,524 | 4,784 | 593 | 1,321 |
| Gov | 9,405 | 3,483 | 2,433 | 4,784 | 1,699 | 1,177 |
| Psy-I | 8,151 | 736 | 1,700 | 4,144 | 354 | 795 |
| Psy-II | 8,324 | 755 | 2,588 | 4,146 | 370 | 1,317 |

Table 1: Class-wise student answer distribution in train and test sets for the four domains used in experiments.

| Corpus | # of words | Domain |
|---|---|---|
| English Wiki | 2.5B | General |
| BooksCorpus | 0.8B | General |
| Textbooks | 1.1M | Phy + Gov |
| QAs | 0.6M | Phy + Gov |
| | 1.3M | Phy + Gov + Psy-I,II |

Table 2: Size and domain of pre-training text corpora.

### 4.1 Dataset

We show results on a proprietary large-scale industry dataset consisting of three domains - (1) Physiology of Behavior (Phy), (2) American Government (Gov), and (3) Psychology – Human Development (Psy-I) and Abnormal Psychology (Psy-II). Given a question, reference answer and student answer, we address a 3-way classification task into `correct`, `incorrect` and `partially correct` classes. Table 1 shows the train-test splits for each of the domains.

We obtain three textbooks for each of the Physiology and American Government domains. These include our own textbooks and additional ones downloaded from Lumen Learning[†] and Gutenberg[‡] websites. We do not use any textbooks for the Psychology domain, to show the effect of BERT pre-training on out-of-domain data. We combine the data from all the textbooks for further pre-training of BERT. Table 2 summarizes the sizes of the textbook corpora (1.1M words) and question-answer corpora (1.3M words) used in our experiments. Note that, the original BERT model is learned with about 3.3B words corpora.

### 4.2 Implementation Details

We leverage the TensorFlow implementation of BERT-Base[§] for all our experiments. It is further pre-trained for 240K, 150K, and 240K epochs for BERT+Textbook$_{\text{Phy+Gov}}$, BERT+QA$_{\text{Phy+Gov}}$, and BERT+QA$_{\text{Phy+Gov+Psy-I,II}}$ respectively using the same hyperparameters until the accuracy of the two pre-training objectives converges to 100%.

[†] *https://lumenlearning.com/*
[‡] *https://www.gutenberg.org/*
[§] *https://github.com/google-research/bert*

| | BERT | (+Textbook) | (+QA) | (+Textbook+QA) |
|---|---|---|---|---|
| Phy | 81.35 | 82.28 | 83.95 | 83.64 |
| Gov | 74.55 | 76.25 | 75.43 | 77.28 |
| Psy-I | 78.69 | 78.05 | 77.71 | 77.89 |
| Psy-II | 80.06 | 79.10 | 79.46 | 80.74 |

Table 3: Effect of domain data. Textbook and Question-Answer data from only Phy and Gov domains are used in the pre-trained model update; making Psy-I and Psy-II unseen domains.

Note that the corpus size for Phy+Gov is smaller, leading to faster convergence. Once the pre-training is done, we fine-tune the model with the short answer grading labeled data for 3 epochs using a learning rate of 3e-5. All results are reported in terms of macro-averaged F1.

### 4.3 Effect of Domains Textbook Data

In this set of experiments, we aim to understand the effect of updating the pre-trained BERT LM on textbook data. We take the textbooks from only two domains (Physiology and American Government), for designing a scenario where textbooks are not available. Such a scenario helps us understand the generalizability of BERT (for which all the domains are unseen), and BERT+Textbook data (for which Psy-I and Psy-II are unseen domains). We combine the data from both the domains and pre-train a single BERT model as pre-training per-domain models is computationally expensive and almost impossible to scale.

Table 3 shows the results. The pre-trained BERT model, as is, performs fairly well (74-81% M-F1). The LM updated with textbook data (BERT+Textbook), improves performance on the domains included in additional pre-training (Phy and Gov). However, we suspect that the updated model becomes more specialized towards seen domains, which leads to performance degradation on the unseen domain of Psychology.

- RQ1 and RQ2 can be answered as *LM update using domain textbook data positively affects the short answer grading performance on the corresponding domains.*
- RQ3 can be answered as *the updated BERT LM model does not appear to generalize well on unseen domains, as the evidence suggests that LM becomes more domain-specific.*

### 4.4 Effect of Question-Answer Data

In this set of experiments, we aim to understand effectiveness of the Question-Answer (QA) data to update the LM. As explained earlier, for each reference answer and correct answer, question-answer pairs are created and utilized as documents.

In Table 3, a combined QA dataset from Psy and Gov subjects is used for running additional steps of LM pre-training. This again simulates the unseen domain scenario for Psy-I and Psy-II in pre-training update. It can be observed that the proposed approach utilizing QA data (BERT+QA) improves the performance consistently in both the seen subjects of Phy and Gov. Akin to textbook data experiments, the performance is degrading for unseen domains as the model becomes more specialized. Interestingly enough, the updated LM on both strategies (BERT+Textbook+QA) also positively impacts on in-domain performance.

Additionally, another set of results is obtained by utilizing the QA dataset of *all four* domains; simulating an all seen-domain scenario. Table 4 reports the corresponding results showing that the QA data helps improve the model for all the domains, consistently.

| | BERT | BERT+QA |
|---|---|---|
| Phy | 81.35 | **83.36** |
| Gov | 74.55 | **76.25** |
| Psy-I | 78.69 | **78.73** |
| Psy-II | 80.06 | **81.41** |

Table 4: Effect of Question-Answer data in seen-domains scenario. All four domains are included in the Question-Answer data.

These observations suggest that the proposed approach re-purposes the `Next Sentence Prediction` task to effectively encode the latent features of ASAG. Further, RQ1 can be answered affirmatively; and RQ4 can be answered as *it is advisable to use the QA corpora for LM model update in addition to fine-tuning.*

## 5 Conclusion

In this paper, we proposed two ways to update the pre-trained BERT language model for the short answer grading task. We illustrate utilization of unstructured textbook data and labeled question answer data for the model update. We show that by adding a step of updating BERT using these domain-related resources, we can achieve better results than directly fine-tuning pre-trained BERT on the end task. We also observed that the updated model becomes more specialized towards the corresponding domains, adversely affecting the per-

formance on unseen domains.

We limited the scope of this paper to the task of automatic short answer grading only. However, our findings of the sensitivity of domain-specific BERT models appear generic. We believe that any multi-domain text classification task should exhibit similar behavior. We also note that our strategies for improving the BERT model should be directly applicable to other QA tasks as well. Future directions include trying our method on different tasks and different relevant data.

## References


Iz Beltagy, Arman Cohan, and Kyle Lo. 2019. Scibert: Pretrained contextualized embeddings for scientific text. *arXiv preprint arXiv:1903.10676*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

Sachin Kumar, Soumen Chakrabarti, and Shourya Roy. 2017. Earth moverâĂŹs distance pooling over siamese lstms for automatic short answer grading. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 2046–2052.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. Biobert: pre-trained biomedical language representation model for biomedical text mining. *arXiv preprint arXiv:1901.08746*.

Smit Marvaniya, Swarnadeep Saha, Tejas I Dhamecha, Peter Foltz, Renuka Sindhgatta, and Bikram Sengupta. 2018. Creating scoring rubric from representative student answers for improved short answer grading. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 993–1002. ACM.

Michael Mohler, Razvan Bunescu, and Rada Mihalcea. 2011. Learning to grade short answer questions using semantic similarity measures and dependency graph alignments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 752–762. Association for Computational Linguistics.

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In *AAAI*, volume 16, pages 2786–2792.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Brian Riordan, Andrea Horbach, Aoife Cahill, Torsten Zesch, and Chong Min Lee. 2017. Investigating neural architectures for short answer scoring. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 159–168.

Swarnadeep Saha, Tejas I Dhamecha, Smit Marvaniya, Renuka Sindhgatta, and Bikram Sengupta. 2018. Sentence level or token level features for automatic short answer grading?: Use both. In *International Conference on Artificial Intelligence in Education*, pages 503–517. Springer.

Md Arafat Sultan, Cristobal Salazar, and Tamara Sumner. 2016. Fast and easy short answer grading with high accuracy. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1070–1075.

Chul Sung, Tejas Indulal Dhamecha, and Nirmal Mukhi. 2019. Improving short answer grading using transformer-based pre-training. In *Artificial Intelligence in Education*, volume 2, pages 469–481. Springer International Publishing.

# WIQA: A dataset for "What if..." reasoning over procedural text

**Niket Tandon***, **Bhavana Dalvi Mishra***, **Keisuke Sakaguchi**,
**Antoine Bosselut**, **Peter Clark**
Allen Institute for Artificial Intelligence, Seattle, WA
{nikett,bhavanad,keisukes,antoineb,peterc}@allenai.org

## Abstract

We introduce WIQA, the first large-scale dataset of "What if..." questions over procedural text. WIQA contains three parts: a collection of paragraphs each describing a process, e.g., beach erosion; a set of crowdsourced *influenct graphs* for each paragraph, describing how one change affects another; and a large (40k) collection of "What if...?" multiple-choice questions derived from the graphs. For example, given a paragraph about beach erosion, would stormy weather result in more or less erosion (or have no effect)? The task is to answer the questions, given their associated paragraph. WIQA contains three kinds of questions: perturbations to steps mentioned in the paragraph; external (out-of-paragraph) perturbations requiring commonsense knowledge; and irrelevant (no effect) perturbations. We find that state-of-the-art models achieve 73.8% accuracy, well below the human performance of 96.3%. We analyze the challenges, in particular tracking chains of influences, and present the dataset as an open challenge to the community.

## 1 Introduction

Procedural text is common in language, but challenging to comprehend because it describes a dynamically changing world. While recent systems for procedural text comprehension can answer questions about what events happen, e.g., (Bosselut et al., 2018; Henaff et al., 2017; Dalvi et al., 2018), the extent to which they understand the influences *between* those events remains unclear.

One important test of understanding is to predict what would happen if a process was *perturbed* in some way, requiring understanding and tracing the chain of influences through a paragraph. However, to date there is no dataset available to help develop this capability. We aim to fill this gap with WIQA[1], the first large-scale dataset testing "What if..." reasoning over procedural text.

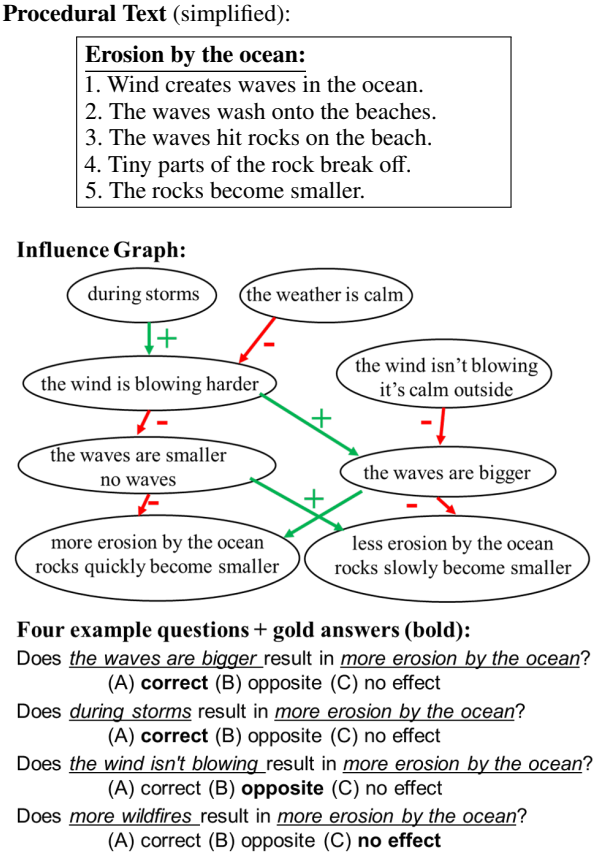


Figure 1: WIQA contains procedural paragraphs, crowdsourced influence graphs associated with them, and a large collection of "Does *changeX* result in *changeY*?" (what-if) questions, derived from the graphs.

WIQA contains 40.7K questions, for 379 process paragraphs. To efficiently create the questions, crowdworkers created 2107 *influence graphs* (IGs) for the paragraphs, describing how one perturbation positively or negatively influences another (Fig-

*Niket Tandon and Bhavana Dalvi Mishra contributed equally to this work.

[1]The dataset is available at http://data.allenai.org/wiqa/



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6076–6085,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

ure 1). Questions were then derived from paths in the graphs, each asking how the change described in one node affects another. Each question is a templated, multiple choice (MC) question of the form *Does changeX result in changeY? (A) Correct (B) Opposite (C) No effect*, where *Opposite* indicates a negative influence between *changeX* and *changeY*. To bound the task, perturbations are typically qualitative (e.g., "the wind is blowing harder"), and possible effects are restricted to changes to entities and events mentioned in the paragraph (e.g., "the waves are bigger"). Perturbations themselves include in-paragraph, out-of-paragraph, and irrelevant (no effect) changes. The WIQA task is to answer the questions, given the paragraph (but not the IG).

We first describe the task and how the dataset was constructed, and then present results from baselines and strong BERT-based models. We find that the best model is still 23% behind human performance and the gap further widens with indirect and out-of-paragraph effects, illustrating that the dataset is hard. We present a detailed analysis showing WIQA is rich in linguistic and semantic phenomena. Our contributions are: (1) the new dataset (2) performance measures and an analysis of its challenges, to support research on counterfactual, textual reasoning over procedural text.

## 2 Related Work

While there are several NLP datasets now available for procedural text understanding, e.g., (Kiddon et al., 2016; Dalvi et al., 2018; Weston et al., 2015), these have all targeted the task of tracking entity states throughout the text. WIQA takes the next step of asking how states might *change* if a perturbation was introduced.

Predicting the effects of qualitative change has been studied in the qualitative reasoning (QR) community, but primarily using formal models (Forbus, 1984; Weld and De Kleer, 2013). Similarly, counterfactual reasoning has been studied in the logic community (Lewis, 2013), but again using formal frameworks. In contrast, WIQA treats the task as a mixture of reading comprehension and commonsense reasoning, creating a new NLP challenge.

## 3 Dataset Construction

To efficiently generate questions, we first asked crowdworkers to create *influence graphs* (IGs) for each paragraph. We then create questions from the IGs using paths in the IGs. We now describe this process.

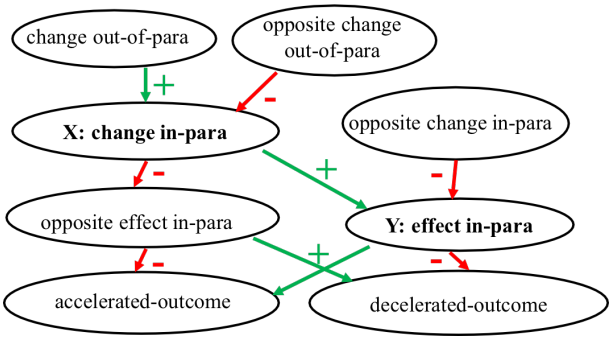


Figure 2: The template used to acquire influence graphs

**Influence Graphs** An influence graph $\mathcal{G}(V, E)$ for a procedural text $T$ is an unweighted directed graph. Each vertex $v_i$ is labeled with one or more text strings, each describing a change to the original conditions described or assumed in $T$, such that all those changes have the same influence on a connected node $v_j$. Each edge is labeled with a *polarity*, **+** or **-**, indicating whether the influence is positive (causes/increases) or negative (prevents/ reduces). Indirect effects can be found by traversing $\mathcal{G}$.
It is useful to distinguish two kinds of nodes:
1. **Out-of-para nodes:** denoting events or changes to entities/events not mentioned in the paragraph, e.g, "during storms" in Figure 1.
2. **In-para nodes:** denoting events or changes to entities/events mentioned in the paragraph, e.g, "the wind is blowing harder" in Figure 1.

**Acquiring influence graphs** For a source of paragraphs, we used the 377 training set paragraphs from the ProPara dataset (Tandon et al., 2018). (Multiple) influence graphs were then crowdsourced for each. To do this, we use an *influence graph template*, shown in Figure 2. Workers were asked to populate this (hidden) template using a sequence of five questions, where the later questions were automatically constructed from their answers to the earlier questions. The first question asks the worker to supply an X and Y in: "If [X] occurs, it will have the intermediate effect [Y] resulting in *accelerated_outcome*" (where the *accelerated outcome* phrase was pre-authored for each paragraph). For X and Y, workers were asked to describe a change in some property/phenomonon mentioned in the paragraph, e.g., if a paragraph sentence $x_i$ is "Wind creates waves.", they may author an X saying "the wind is blowing harder" (Figure 1).

(The alignment of X and $x_i$, and whether X describes an increase or decrease of $x_i$, denoted by $d_X \in \{+, -\}$, was also recorded.). This fills X and Y in Figure 2. Similar questions populate the remaining nodes (see Appendix). 2107 influence graphs were collected in this way.

**Generating Questions from Graphs** Each path in a graph forms a *"change→effect?"* question, whose answer is either "correct" or "opposite" depending on the product of the polarities of the traversed edges. Questions are labeled with the number of edges traversed (1 = "1-hop", etc.). We also distinguish *in-para* and *out-of-para* questions depending on the type of node they originated from. We then created a third category of question, whose answer is "no effect", by selecting out-of-para changes from *other* paragraphs and asking for their effect on nodes in the current graph. Occasionally these changes did affect the selected node, resulting in an erroneous label of "no effect", but this was rare (and such cases were removed from the test partition, as we now describe).

Using a separate crowdsourcing task, questions in the test set were filtered to improve the test set quality. First, five workers independently answered each question, given the paragraph. The inter-annotator agreement between workers, using Krippendorff's alpha, was moderately high (0.6). We then retained only questions with majority agreement (i.e., at least 3 out of 5 workers agreed), resulting in 88% of questions being retained.

**Balancing the Dataset** From the (many) questions thus generated, we (randomly) selected a subset that approximately balanced the numbers of (a) in-para, out-of-para, and no-effect questions, and (b) questions with each answer (correct, opposite, no-effect), resulting in 40,695 questions. Train, dev, and test partitions do not share paragraphs about the same topic. Statistics are shown in Table 1.

**Explanations** As each question is derived from a path in an IG, we can also generate *explanations* for each answer using that path. Although explanation is not part of the WIQA task, we create an explanation database to support a possible explanation task at a future date.

Consider a question "Does perturbation $q_p$ result in $q_e$?" with answer $d_e \in \{+, -\}$ (as a shorthand for $\{correct, opposite\}$), created from an IG path:

$$q_p \xrightarrow{r_{pX}} X \xrightarrow{r_{XY}} Y \xrightarrow{r_{Ye}} q_e$$

Here, $r_{pX}$, $r_{XY}$, and $r_{Ye}$ denote the polarities (+/-) of the edges $q_p$X, XY, and X$q_e$ in the IG respectively. (As described earlier, answer $d_e$ is the product of the polarities $r_{pX}.r_{XY}.r_{Ye}$). To define an explanation in terms of the paragraph's sentences $x_1, ..., x_K$, we define the gold explanation $E_{gold}$ as the structure:

$$q_p \rightarrow d_i x_i \rightarrow d_j x_j \rightarrow d_e q_e$$

where $x_i$ is the sentence corresponding to X, $x_j$ is the sentence corresponding to Y, and $d_i$, $d_j$, and $d_e$ denote the directions of influence (+/-, denoting {more,correct}/{less,opposite}). As workers already annotated the alignment between X and $x_i$ (similarly Y and $x_j$) we know $x_i$ and $x_j$. Similarly, as workers also annotated whether X describes an increase of decrease of $x_i$, denoted by $d_X \in \{+, -\}$, we can straightforwardly compute the directions of influence:

$d_i = r_{pX}.d_X$
$d_j = d_i$ (in-paragraph influence[2])
$d_e = r_{pX}.r_{XY}.r_{Ye}$ = the answer

We can similarly generate explanations for answers derived from 1-hop and 2-hop paths.

We generated a full database of explanations for all the questions with answer "correct" or "opposite" (For "no effect" answers, there is no explanation as there is by definition no path of influence). We then removed the (occasional) explanation where worker annotations were contradictory (e.g., $j < i$) or had no majority decision for an annotation. This database is available for a possible future explanation task (given question + paragraph, produce the answer + explanation).

## 4 Experiments

### 4.1 Models

We measured the performance of two baselines and three strong neural models on WIQA, to understand how it stresses these models:

**Majority** predicts the most frequent label, *correct*, in the training dataset.

**Polarity** is a rule-based baseline that assumes influences of the form "*more X → more Y*" (similarly for "less") are *correct*, hence "*more X → less Y*" are *opposite*. A small lexicon of positive (*"more"*) and negative (*"less"*) words is used to assign the more/less polarities. A random class label is chosen when assignments

[2]Paragraph sentences always describe correct, not opposite, influences on later sentences, HENCE if $x_i$ is more/accelerated, $x_j$ will be too (similarly for less/decelerated).

| Count of | Train | Dev | Test | Total |
|---|---|---|---|---|
| Topics | 87 | 23 | 12 | 122 |
| Paragraphs | 261 | 77 | 41 | 379 |
| Influence graphs | 1453 | 424 | 230 | 2107 |
| Questions | 29808 | 6894 | 3993 | 40695 |

| | | # Questions | | | |
|---|---|---|---|---|---|
| | | Train | Dev | Test | Total |
| Question type | in-para | 7303 | 1655 | 935 | 9893 |
| | out-of-para | 12567 | 2941 | 1598 | 17108 |
| | no-effect | 9936 | 2298 | 1460 | 13694 |
| | Total | 29808 | 6894 | 3993 | 40695 |
| Number of hops (in- & out-of-para qns) | #hops=1 | 6754 | 1510 | 835 | 9099 |
| | #hops=2 | 8969 | 2145 | 1153 | 12267 |
| | #hops=3 | 4149 | 941 | 545 | 5635 |
| | Total | 19872 | 4596 | 2533 | 27001 |

Table 1: Dataset statistics

cannot be made.

**Adaboost** (Freund and Schapire, 1995) was used to make the 3-way classification using several bag-of-word features computed from *change* and *effect*.

**Decomp-Attn** applies the Decomposable Attention (DA) model of (Parikh et al., 2016) to our task. The original DA model computes entailment, i.e., the confidence that a *premise* entails (or contradicts) a *hypothesis*. We recast WIQA as an entailment task where cause-effect becomes premise-hypothesis, and (correct/opposite/no-effect) correspond to (entailment/contradiction/neutral). We retrain the DA model on WIQA using this mapping.

**BERT** (Devlin et al., 2018) is a pre-trained transformer-based language model that has achieved state of the art performance on many NLP tasks. We supply questions to BERT as *[CLS] paragraph [SEP] question [SEP] answer-option* for each of the three options. The [CLS] token is then projected to a single logit and fed through a softmax layer across the three options, using cross entropy loss, and the highest-scoring option selected. We fine-tune BERT on the WIQA training data in this way. We also measure an ablated version where the paragraph is omitted (train and test).

**Human Performance** was estimated by having three people independently answering the same 100 questions (with paragraphs) drawn randomly from the test set. Krippendorff's alpha (nominal metric) for these answers was 0.908 (high agreement) (Krippendorff, 1970).

| Question Type | in-para | out-of-para | no-effect | Total |
|---|---|---|---|---|
| # questions | 935 | 1598 | 1460 | 3993 |
| *Majority* | 45.46 | 49.47 | 0.55 | 30.66 |
| *Polarity* | 76.31 | 53.59 | 0.27 | 39.43 |
| *Adaboost* | 49.41 | 36.61 | 48.42 | 43.93 |
| *Decomp-Attn* | 56.31 | 48.56 | 73.42 | 59.48 |
| *BERT (no para)* | 60.32 | 43.74 | 84.18 | 62.41 |
| *BERT* | **79.68** | **56.13** | **89.38** | **73.80** |
| Human perf. | | | | 96.33 |

Table 2: Comparing models on WIQA test partition

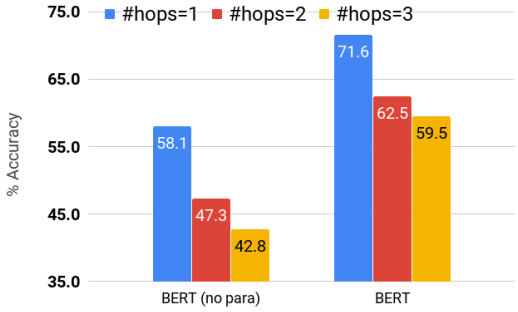


Figure 3: Accuracy of the best baselines drops as number of hops increase, quicker for 'no para' version.

## 5 Results and Analysis

### 5.1 Prediction Accuracy

The results (Table 2) provide several insights:

1. **The dataset is hard.** Our strongest model (73.8) is over 20 points behind human performance, suggesting WIQA poses significant challenges. Prediction of out-of-para effects is particularly challenging, 37 points behind human performance.

2. **BERT already "knows" some change-effect knowledge.** Even without the paragraph, and even though the test paragraphs are on topics unseen in training, BERT scores substantially above the baselines. This suggests BERT has some type of cause-effect information embedded in it.

3. **Supplying the paragraph helps,** resulting in 10 points higher score, illustrating that WIQA contains questions that require understanding of the paragraph. This suggests more sophisticated reading strategies may further improve results.

### 5.2 Predicting Indirect Effects

In-para and out-of-para questions were derived from chains of different lengths ("hops") in the influence graphs. Figure 3 shows how performance varies with respect to those lengths, and shows that

**"indirect" (2/3-hop) effects are harder to predict than "direct" (1-hop) effects**. For example, it is easier to predict "cloudy day" results in "less sunshine" (direct) than "less photosynthesis" (indirect). This suggests that some form of reasoning along influence chains may be needed to predict indirect effects reliably, as those effects are less likely to be explicitly stated in corpora and embedded in pre-trained language models.

### 5.3 Consistency

Are the models making consistent predictions? If a model predicts both X→Y and Y→Z are correct, it should, if it were consistent, also predict *X* →Z is correct. To measure a model's *transitivity consistency*, for each influence graph, we measure how often its indirect predictions (2/3-hop) are consistent[3] with its 1-hop predictions. Similarly, we measure *disjunctive consistency* by how often its predictions for edges known to be opposite (eg X→Y and X→*opp-effect-in-para* in Fig 2) are indeed so[4]. The results in Figure 4 illustrate that the **models are far from consistent**. This suggests that reasoning with global consistency constraints may improve results, e.g., (Ning et al., 2017; Tandon et al., 2018).

### 5.4 Linguistic and Semantic Phenomena

We analyzed 200 descriptions of changes in 100 random questions, and observe the following challenging (overlapping) phenomena to handle:

**1. Qualitative Language:** ≈65% of the change statements are expressed qualitatively, using a broad vocabulary of comparatives (e.g., more, fewer, smaller, larger, cooler, slower, higher, harder, decreased, hotter) or their corresponding adjectives (small, cool, etc.). In addition, whether the change is a positive or negative influence on the process is context-dependent ("more X" can be positive or negative, depending on X, and sometimes depending on the paragraph topic itself).

**2. Commonsense** (≈45%): Exogenous influences are (by definition) not stated in the paragraph, and so require substantial commonsense to understand, e.g., that "heavy rainfall" (out of para) negatively influences "more wild fires" (in para); or that "overfishing" (out of para) negatively influences "fish lay eggs" (in para).

[3]i.e., the polarity (+/-, for correct/opposite) of edge XZ = the product of the polarities of edges chaining from X to Z. As models can also predict "no effect", random score is 1/3.

[4]Only edge pairs with labels +&-, or -&+, are disjunctively consistent (of 9 possible labelings), hence random is 2/9.

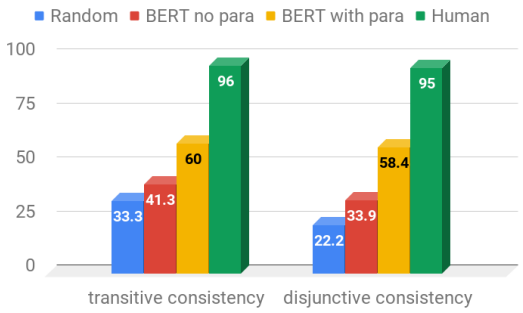


Figure 4: The best models (red,yellow) make substantially less consistent predictions than humans (green).

**3. Lexical matching** ≈15% of the in-para changes refer to paragraph entities using different terms, e.g., "insect" (para) ↔ "bee" (question), "becomes" ↔ "forms", "removes" ↔ "expels", complicating aligning questions with the paragraph.

**4. Negation** (≈6%): Negation occurs in about 6% of the changes, e.g., "drought does not occur", "soil is not fertile", "magma does not get larger".

**5. Juxtaposed polarities** (≈3%): Sometimes positive- and negative-related terms are juxtaposed, (e.g., "much less", "increased deforestation", "less severe") again challenging to process.

These all illustrate the diversity of linguistic and semantic challenges in WIQA.

## 6 Conclusion

An important test of understanding procedural text is whether the effects of *perturbations* to the process can be predicted. To that end, we have introduced WIQA, the first large-scale dataset for "what if" reasoning over text. While our experiments suggest language models have some built-in knowledge of influences, and some ability to identify influences in paragraphs, these capabilities are limited, producing predictions that are over 20 points worse than humans, often inconsistent, and particularly erroneous about indirect (multi-hop) effects. WIQA aims to improve this state of affairs, offering a new challenge and resource to the community. The dataset is available at http://data.allenai.org/wiqa/.


### Acknowledgements

We are grateful to the AllenNLP and Beaker teams at AI2, and for the insightful discussions with other Aristo team members. Computations on beaker.org were supported in part by credits from Google Cloud.

## References


Antoine Bosselut, Omer Levy, Ari Holtzman, Corin Ennis, Dieter Fox, and Yejin Choi. 2018. Simulating action dynamics with neural process networks. *6th International Conference on Learning Representations (ICLR)*.

Bhavana Dalvi, Lifu Huang, Niket Tandon, Wen-tau Yih, and Peter Clark. 2018. Tracking state changes in procedural text: A challenge dataset and models for process paragraph comprehension. In *NAACL-HLT'18*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Kenneth D. Forbus. 1984. Qualitative process theory. *Artificial Intelligence*, 24:85–168.

Yoav Freund and Robert E. Schapire. 1995. A decision-theoretic generalization of on-line learning and an application to boosting. In *EuroCOLT*.

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. 2017. Tracking the world state with recurrent entity networks. In *ICLR*.

Chloé Kiddon, Luke S. Zettlemoyer, and Yejin Choi. 2016. Globally coherent text generation with neural checklist models. In *EMNLP*.

Klaus Krippendorff. 1970. Estimating the reliability, systematic error and random error of interval data. *Educational and Psychological Measurement*, 30(1):61–70.

David Lewis. 2013. *Counterfactuals*. John Wiley & Sons.

Qiang Ning, Zhili Feng, and Dan Roth. 2017. A structured learning approach to temporal relation extraction. In *EMNLP*, pages 1027–1037.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255, Austin, Texas. Association for Computational Linguistics.

Niket Tandon, Bhavana Dalvi Mishra, Joel Grus, Wen-tau Yih, Antoine Bosselut, and Peter Clark. 2018. Reasoning about actions and state changes by injecting commonsense knowledge. *EMNLP'18*, arXiv preprint arXiv:1808.10012.

Daniel S Weld and Johan De Kleer. 2013. *Readings in qualitative reasoning about physical systems*. Morgan Kaufmann.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards AI-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.


## Appendix A: Topicwise consistency

We study trends in topic-wise accuracy of models as they read more context information. Bert no-para model does not have access to any context or paragraph, except the language model's background knowledge from Wikipedia. By reading the paragraph context Bert with-para model performs much better on certain topics such as `Pollination, blood, mountain, evaporation` but the impact of reading is much less on topics such as `Igneous rocks, plant crops, solar eclipse, DNA replication`. Topics such as `blood` are very popular on Wikipedia and distributed across several very different articles. These topics are harder for BERT as it requires additional paragraph context to understand the question.

| topic | BERT (no para) | BERT |
|---|---|---|
| igneous rock | 0.66 | 0.64 |
| plant crops | 0.61 | 0.61 |
| solar eclipse | 0.43 | 0.43 |
| frog | 0.59 | 0.62 |
| DNA replication | 0.58 | 0.63 |
| water cycle | 0.63 | 0.69 |
| fish | 0.5 | 0.57 |
| pumpkin | 0.61 | 0.69 |
| pollination | 0.62 | 0.75 |
| blood | 0.62 | 0.76 |
| mountain | 0.57 | 0.72 |
| evaporation | 0.42 | 0.67 |

Table 3: As the Bert model (that has access to the paragraph in context) reads more paragraphs in context, its accuracy is better. Reading helps certain topics such as `Pollination, blood, mountain, evaporation` more than others

## Appendix B: Crowdsourcing Influence Graphs

We crowdsource influence graphs by getting the graphs constructed progressively, with the help of five questions stated in Figure 7. At first, the turkers see an empty graph in Figure 5.

When the annotators answer the first question (shown in Fig. 7), two nodes of the partial influence graph are filled (depicted in Fig. 6).

Once all the questions are answered, the influence graph will be ready. During the process of annotation, there are appropriate validations for quality control.

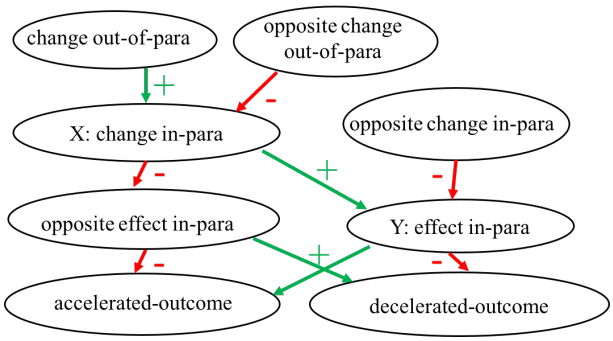


Figure 5: At the start of the process to annotate an influence graph for a given paragraph, the annotators see a blank influence graph with the basic structure.

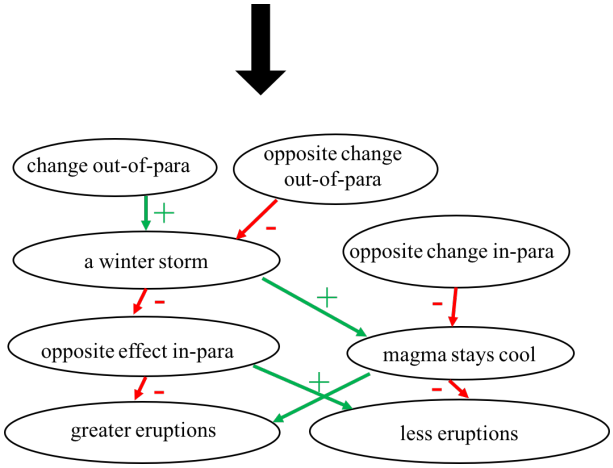


Figure 6: As the annotators answer questions in Fig. 7, a partial influence graph emerges. As they answer questions, the annotators found it useful to validate their answers by examining the emerged influence graph.

## Appendix C: Sample Influence Graphs

To get an impression of our crowdsourced influence graph repository, we display four paragraphs (not hand picked) in Figures 8, 9, 10, 11. These range from natural process, to human body process and mechanical process.

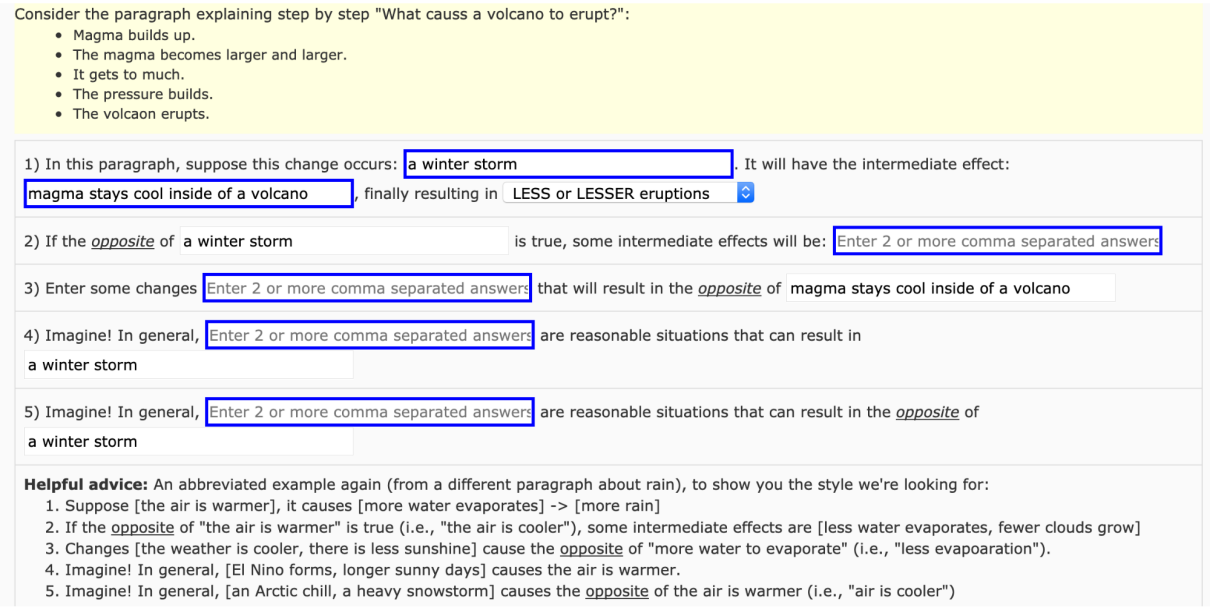


Figure 7: The interface shown to the annotators on Mechanical turk platform. Given a paragraph in yellow background, the annotators answer the five questions and an influence graph emerges from their answers.

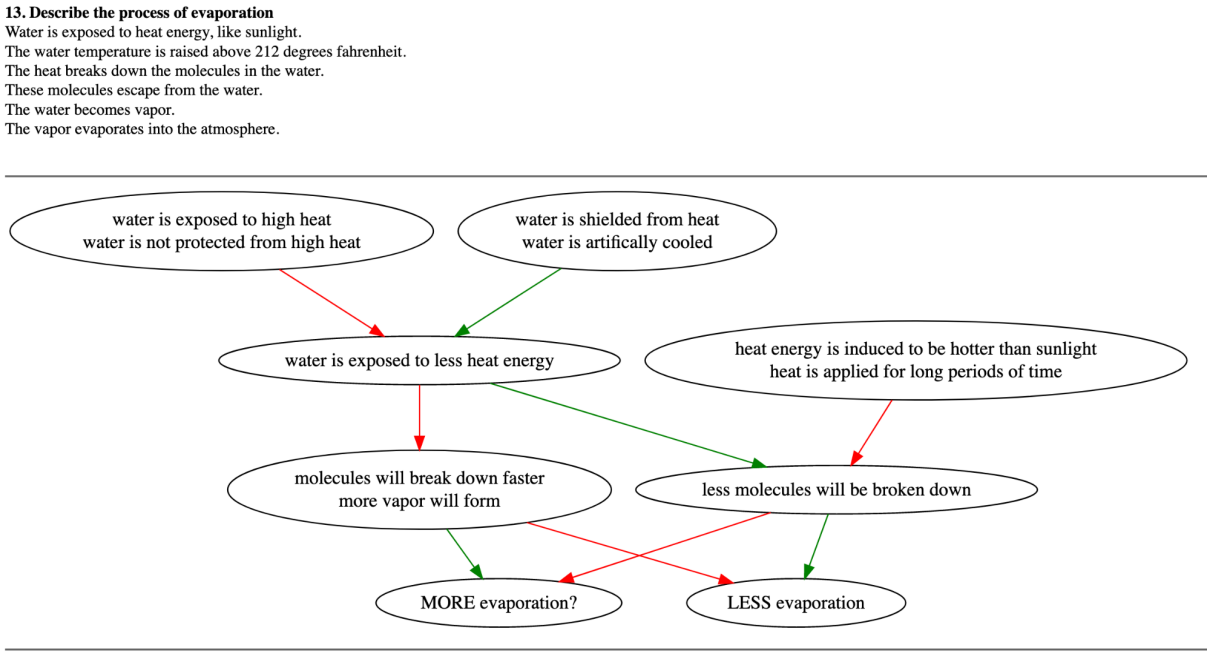


Figure 8: Influence graph for a paragraph from the topic `evaporation`

**451. Describe how a flashlight works**
Batteries are put in a flashlight.
The flashlight is turned on.
Two contact strips touch one another.
A circuit is completed between the batteries and the lamp.
The lamp in the flashlight begins to glow.
The reflector in the flashlight directs the lamps beam.
A straight beam of light is generated.
The flashlight is turned off.
The circuit is broken.
The beam is no longer visible.

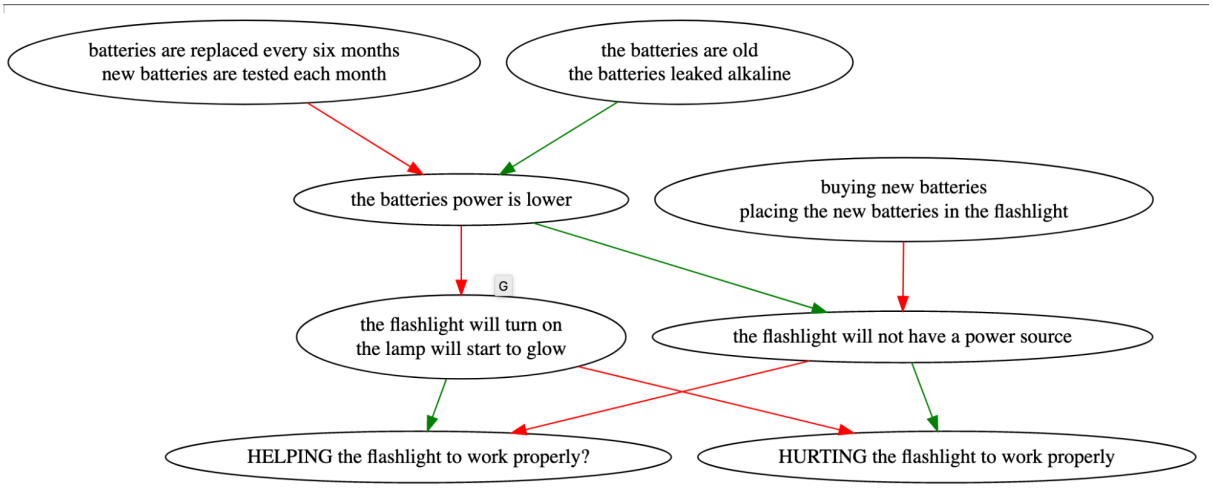


Figure 9: Influence graph for a paragraph from the topic `flashlight`

**16. What do lungs do?**
You breathe oxygen into your body through the nose or mouth.
The oxygen travels to the lungs through the windpipe.
The air sacs in the lungs send the oxygen into the blood stream.
The carbon dioxide in the blood stream is transferred to the air sacs.
The lungs expel through the nose or mouth back into the environment.

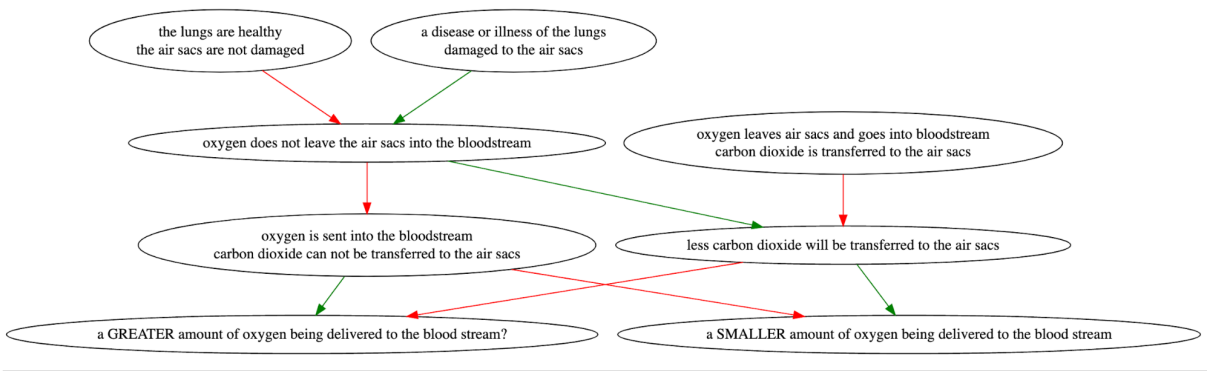


Figure 10: Influence graph for a paragraph from the topic `lungs`

**5. How do minerals form?**
Magma comes up to the surface of the earth.
The magma cools.
Particles inside the magma move closer together.
Crystals are formed.
The crystals contain minerals.

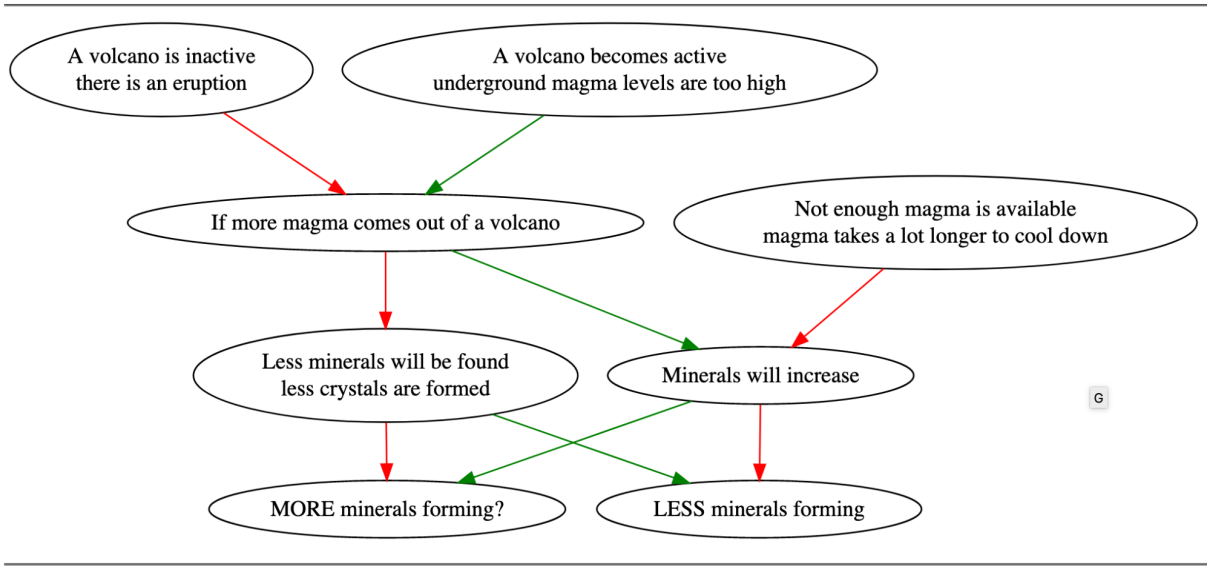


Figure 11: Influence graph for a paragraph from the topic `minerals`

# Evaluating BERT for natural language inference: A case study on the CommitmentBank

**Nanjiang Jiang** and **Marie-Catherine de Marneffe**
Department of Linguistics
The Ohio State University
{jiang.1879, demarneffe.1}@osu.edu

## Abstract

Natural language inference (NLI) datasets (e.g., MultiNLI) were collected by soliciting hypotheses for a given premise from annotators. Such data collection led to annotation artifacts: systems can identify the premise-hypothesis relationship without observing the premise (e.g., negation in hypothesis being indicative of contradiction). We address this problem by recasting the CommitmentBank for NLI, which contains items involving reasoning over the extent to which a speaker is committed to complements of clause-embedding verbs under entailment-canceling environments (conditional, negation, modal and question). Instead of being constructed to stand in certain relationships with the premise, hypotheses in the recast CommitmentBank are the complements of the clause-embedding verb in each premise, leading to no annotation artifacts in the hypothesis. A state-of-the-art BERT-based model performs well on the CommitmentBank with 85% F1. However analysis of model behavior shows that the BERT models still do not capture the full complexity of pragmatic reasoning, nor encode some of the linguistic generalizations, highlighting room for improvement.

## 1 Introduction

Natural language inference (NLI), the task of identifying whether a hypothesis can be inferred from, contradicted by, or not related to a premise, has become one of the standard benchmark tasks for natural language understanding. NLI datasets, such as SNLI (Bowman et al., 2015) and MultiNLI (Williams et al., 2018), are typically built by asking annotators to compose sentences based on premises extracted from corpora, so that the composed sentences stand in entailment/contradiction/neutral relationship to the premise. The hypotheses collected this way have been found to contain annotation artifacts – clues allowing systems to identify the relationship between a premise and a hypothesis without observing the premise. For instance, Gururangan et al. (2018) found that in SNLI and MultiNLI, negation is highly indicative of contradiction and generic nouns (e.g., *animal, something*) of entailment.

| | |
|---|---|
| *Premise:* A: Boy that's scary, isn't it. B: Oh, can you imagine, because it happens in the middle of the night, so you know, these parents didn't know the kid was gone until the kid is knocking on the door screaming, let me in.<br>*Hypothesis:* the kid was gone. | `Entailment (1.92)` |
| *Premise:* "Clever". Klug means "clever". Would you say that Abie was clever?<br>*Hypothesis:* Abie was clever. | `Neutral (0)` |
| *Premise:* GM confirmed it received U.S. antitrust clearance to boost its holding. Sansui Electric agreed to sell a 51% stake to Polly Peck of Britain for $110 million. Still, analysts said the accord doesn't suggest Japan is opening up to more foreign takeovers.<br>*Hypothesis:* Japan is opening up to more foreign takeovers. | `Contradiction (-1.2)` |

Table 1: Examples from the CommitmentBank, with NLI class and original annotation mean.

To address this issue, we recast the CommitmentBank (CB henceforth) (de Marneffe et al., 2019), an English dataset of speaker commitment/event factuality, for NLI.[1] The original CommitmentBank includes naturally occurring discourses annotated with speaker commitment towards the content of complements of clause-embedding verbs under entailment-canceling environments (negation, modal, question and conditional). CB does not suffer from the drawback of annotation artifacts in the hypotheses, since the hypotheses are the complement of a clause-embedding verb in the premise. It thus tests for

[1]The recast CommitmentBank is part of the SuperGLUE benchmark (Wang et al., 2019a), at `https://super.gluebenchmark.com`.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6086–6091,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

inferences involving a particular kind of syntactic construction and contains no annotation artifacts in the hypothesis, making it suitable to test for robust language understanding.

CB has many challenging aspects which are highlighted in various adversarial NLI datasets. It can be thought of as a variant of HANS (McCoy et al., 2019), which contains examples where the hypothesis is a subsequence or a constituent of the premise. It contains several phenomena in the "stress tests" (Naik et al., 2018) including word overlap, negation, and length mismatch. However these datasets are artificially constructed while CB data are naturally occurring.

Here we evaluate BERT, the state-of-the-art model in NLI, on CB. While BERT models achieve good performance with supervision from both CB and MultiNLI, they still struggle with items involving pragmatic reasoning and lag behind human performance. Experiments show that BERT does not use the linguistic generalizations for speaker commitment to make predictions, although BERT can learn them with direct supervision. CB is thus a useful benchmark for measuring progress on robust natural language understanding and specifically speaker commitment inferences.

## 2 The CommitmentBank

To study the linguistic correlates of speaker commitment in English, de Marneffe et al. (2019) introduced the CommitmentBank dataset.[2] It consists of naturally occurring English items with up to two sentences of preceding context and one target sentence, from three genres: newswire (Wall Street Journal), fiction (British National Corpus), and dialogue (Switchboard). The target sentences contain a clause-embedding verb (such as *think*) in an entailment-canceling environment (negation, modal, question, or conditional). Each item has at least 8 annotations indicating the extent to which the speaker of the sentences are committed to the truth of the embedded clause ($+3$/speaker is certain that it is true, $0$/speaker is not certain about its truth, $-3$/speaker is certain that it is false).

### 2.1 Recast for NLI

For each item, we take the context and target sentence to be the premise, and the embedded clause in the target sentence to be the hypothesis.

[2]The original CommitmentBank is available at https://github.com/mcdm/CommitmentBank

| | Entailment | Neutral | Contradiction | Total |
|---|---|---|---|---|
| Train | 115 | 16 | 119 | 250 |
| Dev | 23 | 5 | 28 | 56 |
| Test | 113 | 16 | 121 | 250 |
| Total | 251 | 37 | 268 | 556 |

Table 2: Number of items with each gold label in each split. Split are from Wang et al. (2019a).

We identified a subset of the CommitmentBank with high annotator agreement, and assigned categorical labels (entailment/neutral/contradiction) to them according to their mean annotations in $[-3, 3]$. We label an item as entailment if at least 80% of its annotations are within $[1, 3]$, where the speaker is committed to the complement $p$, as neutral if within $[0]$, where the speaker is uncommitted toward $p$, as contradiction if within $[-3, -1]$ where the speaker is committed to $\neg p$. We discard the item if less than 80% of the annotations are within one of the three sub-ranges. Table 1 contains examples from CB with the original mean annotation and the gold NLI label. The number of items in each class is in Table 2.

### 2.2 Possible Annotation Artifacts

Since the hypotheses in CB are extracted from the premises instead of generated by annotators, we expect CB to contain less annotation artifacts compared to SNLI or MultiNLI.

**Length** Gururangan et al. (2018) found that entailed hypotheses in SNLI tend to be shorter and neutral ones longer.[3] The hypothesis length in the CB train set is distributed evenly across the three classes (mean length 8.5 tokens for entailment, 6.6 for neutral and 7.3 for contradiction).

**Lexical Features** Following Gururangan et al. (2018), we computed the PMI between each unigram/4-gram and class in the training set,[4] capturing the extent to which an expression is associated with each class. Table 3 gives the five unigrams and 4-grams with the highest PMI values. For hypotheses, there don't seem to be any discriminating expressions. In particular, negation words are not features for contradiction, in contrast to SNLI and MultiNLI. For premises, we find some expressions that are strongly associated with

[3]In SNLI, about half of the entailed hypotheses have token length less than 5, while a similar portion of the neutral hypothesis have length under 12 tokens.

[4]We applied add-one smoothing to the raw counts.

| **Hypothesis** | | | | | |
|---|---|---|---|---|---|
| Entailment | | Neutral | | Contradiction | |
| Mr | C ##rois ##set by | pretty | the cat was well | really | ' s going to |
| been | the language was peeled | cat | at any given moment | people | a de ##ter ##rent |
| . | language was peeled down | moment | any given moment a | can | de ##ter ##rent to |
| had | no women are allowed | response | given moment a response | go | is going to be |
| - | women are allowed to | ban | moment a response of | parents | it was too long |
| **Premise** | | | | | |
| Entailment | | Neutral | | Contradiction | |
| g | might have known that | clever | Do you think that | mean | . B : I |
| herself | . You could say | Nicky | . Do you think | five | don ' t think |
| perhaps | Mr . An ##tar | pressure | you know , what | care | B : Uh , |
| wrong | . An ##tar ' | Base | ' ' I hope | guy | : I don ' |
| notice | An ##tar ' s | radio | ' I hope you | jury | , I mean , |

Table 3: Unigrams and 4-grams with top 5 PMI with each class for the hypotheses and premises. We used the BERT cased tokenizer, which marks split word pieces with ##. The 4-grams in the hypotheses listed here share the same PMI values with many other 4-grams in the same class (745 for entailment, 60 for neutral, 6 for contradiction).

a particular class. But most of these expressions align with linguistic generalizations about these particular constructions, as elaborated on below.

**Entailment** The most discriminating expressions for the entailment premises include modal operators *perhaps*, *could* and *might*. This is because 63 out of the 115 entailment items in the train set involve the modal environment. Factive verbs *notice* and *know* are also discriminating features of entailment, indicating that factive verbs tend to suggest the truths of their complement (Karttunen, 1971).

**Neutral** The most discriminating expressions for the neutral class include questions: *Do you think*. This is due to the fact that 10 out of 16 neutral items are under the question environment.

**Contradiction** For contradiction, the most discriminating expressions involve neg-raising constructions (*I don't think/know/believe that p*, where the speaker is committed to *p* being false), filler phrases *Uh* and *I mean*, and indicator of speakers in dialogues *B:*. These are all characteristic of the Switchboard genre, which makes up 80% of the contradictions in the training set.

## 3 Predicting NLI labels

We evaluate BERT, the state-of-the-art model for NLI, on CB.[5] The BERT model follows the standard practice for sentence-pair tasks as in Devlin et al. (2019). We concatenate the premise and the hypothesis with `[SEP]`, prepend the sequence with `[CLS]`, and feed the input to BERT. The representation for `[CLS]` is fed into a softmax layer for a three-way classification.

For all experiments, we used the Adam optimizer (Kingma and Ba, 2015) with an initial learning rate of 1e-5, a batch size of 8, and fine-tuned with at most 10 epochs on each dataset. We fine-tuned BERT with three different sets of training data: CB only (CB$^B$), MultiNLI only (MNLI$^B$), and MultiNLI first then CB (MNLI+CB$^B$).[6] For comparison, we also included the models' performance on the MultiNLI dev set.

**Baselines** We included two baselines: a bag-of-words baseline (CBOW) in which each item is represented as the average of its tokens' GloVe (Pennington et al., 2014) vectors; a Heuristics baseline, only applicable to the CB dataset, which uses five rules based on the observations in Section 2.2: 1. items under modals are entailments, 2. neg-raising items of the form *I don't think/know/believe* are contradictions, 3. items with factive verbs are entailments, 4. items under negation are contradictions, 5. all other items are neutral.

**Human Performance** We included human performance on CB from Wang et al. (2019a) obtained by asking crowdworkers to re-annotate a part of the test set. The MultiNLI human performance accuracy is for the matched/mismatched test set from Nangia and Bowman (2019).

**Results** Table 4 shows the results. CBOW does not perform well on either datasets. On the CB

[5] We used `jiant` (Wang et al., 2019b) with the `bert_large_cased` model for all our experiments.

[6] Superscript $^B$ denotes BERT model tuned on the corresponding dataset to distinguish them from the actual datasets.

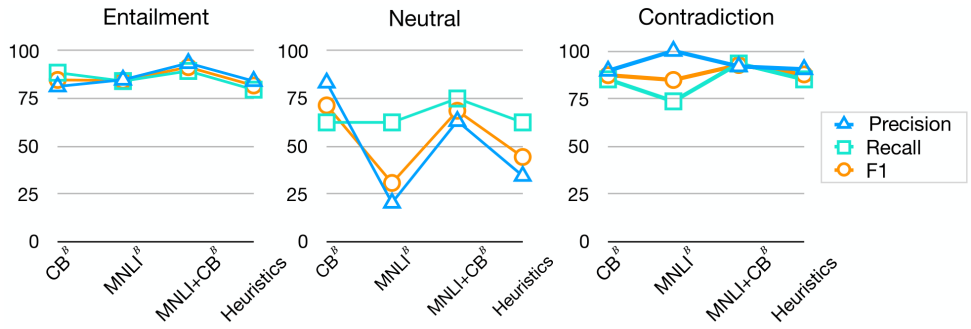


Figure 1: Precision, recall and F1 of each class on the CB test set for three BERT variants and Heuristics.

| | CB | | MultiNLI | |
|---|---|---|---|---|
| | Acc. | F1 | Acc. | F1 |
| CBOW | 69.2 | 47.6 | 71.2 | 71.2 |
| MNLI$^B$ | 77.6 | 66.7 | **83.6** | **83.6** |
| Heuristics | 81.2 | 71.3 | - | - |
| CB$^B$ | 85.2 | 81.2 | 41.1 | 30.6 |
| MNLI+CB$^B$ | **91.2** | **85.3** | 72.3 | 74.4 |
| Human | 98.9 | 95.8 | 92.0/92.8 | - |

Table 4: Performance on the CB test set and the MultiNLI dev set.

data, the heuristics based on linguistic generalizations is a strong baseline, performing better than MNLI$^B$. We gain a lot of performance with supervision from CB only (CB$^B$), which aligns with McCoy et al.'s observation that BERT performs very well when trained with in-domain data. The best results are obtained by MNLI+CB$^B$ on CB and by MNLI$^B$ on MultiNLI, but still lag behind human performance. While MNLI+CB$^B$ gives the best performance on CB, it does not perform well on MultiNLI. This is in line with Liu et al. (2019) who found that fine-tuning on datasets that test for a specific linguistic phenomenon decrease the performance on the original dataset.

## 4 Analysis

Figure 1 shows the precision, recall and F1 scores of each class on the CB test set for the three BERT variants and the Heuristics baseline. Heuristics performs similarly as CB$^B$ on all classes. Compared with CB$^B$, MNLI+CB$^B$ improves the overall performance of contradiction and the recall of neutral. MNLI$^B$ identifies contradictions with perfect precision but poor recall. All models do poorly on the neutral class, which has very few items in the dataset and no clear linguistic generalizations.

| Correct label with Heuristics? | Yes (203) | No (47) |
|---|---|---|
| CB$^B$ | 88.7 | 55.8 |
| MNLI$^B$ | 68.9 | 55.7 |
| MNLI+CB$^B$ | 89.4 | 68.5 |

Table 5: F1 scores of the three BERT models on the CB test set divided by whether Heuristics predicts the correct label (size of each subset in parentheses).

**Linguistic Generalizations and Beyond** 80% of the predictions of both CB$^B$ and MNLI+CB$^B$ are the same as the predictions of the Heuristics baseline, while it drops to 69.6% for MNLI$^B$. We divided the test set by whether Heuristics predicts the correct label. Table 5 reports the models' F1 scores on the two subsets. CB$^B$ and MNLI+CB$^B$ performances on the Yes-items are similar and both outperforming MNLI$^B$ (statistically significant improvements, McNemar's test, $p < 0.01$). There is no statistical difference between the models' performance for the No-items. There is thus a performance gap between items requiring more pragmatic reasoning in general (No-items) and items which can be correctly predicted by identifying certain structures (Yes-items), suggesting that there is still work to achieve robust language understanding. Table 6 shows some items on which MNLI+CB$^B$ still fails.

**Feature Probing** To investigate whether BERT actually learns the linguistic features from the Heuristics baseline and uses them to make NLI predictions, we trained two probing models to predict 1. whether the clause-embedding verb is factive and 2. the type of entailment-canceling environment. Following Tenney et al. (2019), we take the weighted sum of BERT layers (fine-tuned for NLI) to produce a pooled representation for each token. Unlike Tenney et al. (2019), in which the

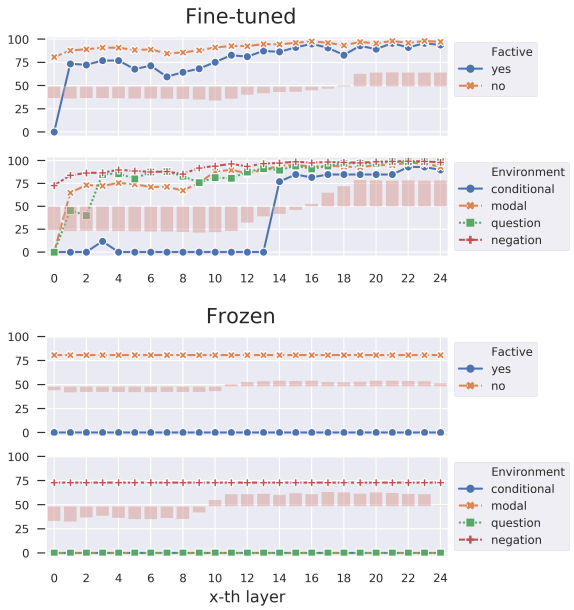


Figure 2: Results on the two probing tasks using parameters from MNLI+CB$^B$. Lineplots show model's F1 scores when using encoding layers up to the $x$-th layer. Barplots show the scalar mixing weights.

representations for the word tokens are used, we take the representation of the `[CLS]` token for each item and fed it into a MLP classifier to predict whether the discourse has certain features. We extracted the trained scalar mixing weights to see the importance of the different layers. For each layer $k$, we trained a series of classifiers using all previous layers up to $k$, to measure at which layer the feature can be correctly predicted.[7]

We did the above experiments in two settings: 1. fine-tune all BERT layers to learn the feature-specific representations, and 2. freeze BERT layers tuned for NLI and only train the probing classifier. The results are shown in Figure 2.

When fine-tuning BERT layers for each feature task, we see that performance increases as more layers are added. Factives, conditionals, and modals are correctly predicted at later layers than nonfactives and negation. For conditionals and modals, this might be due to the fact that they are rare in the dataset. Factives possibly require more contextual information in order to be learned: the scalar weights indicate that factivity is processed at higher layers than entailment-canceling environment. This is consistent with the language acquisition literature (Hacquard and Lidz, 2019) which suggests that rich syntactic/pragmatic information is required to learn the semantics of factive verbs.

However, when we freeze the BERT parameters from MNLI+CB$^B$, the models always give the highest probability to negation environment and nonfactive verb, resulting in zero F1s on every other feature. The scalar mixing weights are smaller than the weights from the fine-tuned model. This suggests that, although BERT can learn these features with direct supervision, training BERT for NLI does not result in representations that encode these features: the model relies on other statistical clues to make decisions.

[7] The code and data are available at https://github.com/njjiang/jiant/tree/cb_emnlp19.

| |
|---|
| *Premise:* B: Yeah, it's called VCX or something like that. Also called Delta Clipper, which is a decent name for something like that. A: Wow. Well, I don't know. you think you'd, uh, go up in space if you had a chance?<br>*Hypothesis:* speaker B would go up in space if he had a chance<br>H: `neutral` B: `contradiction` Gold: `neutral` |
| *Premise:* Those people... Not a one of them realized I was not human. They looked at me and they pretend I'm someone called Franz Kafka. Maybe they really thought I was Franz Kafka.<br>*Hypothesis:* he was Franz Kafka<br>H & B: `entailment` Gold: `contradiction` |

Table 6: Items in the test set with predictions the by Heuristics baseline (H) and MNLI+CB$^B$ (B). The first one is correctly predicted by Heuristics, while the second one is not.

## 5 Conclusion

We introduce CB as a dataset for NLI, and show that it does not contain annotation artifacts in the hypotheses in contrast to previous NLI datasets. Our evaluation shows that despite the high F1 scores, BERT models have systematic error patterns, suggesting that they still do not capture the full complexity of human pragmatic reasoning. There is much room for improvement, and the CB dataset will be a useful testbed to assess models' progress on such reasoning.

## Acknowledgement

We thank Livy Real for suggesting to look at the CommitmentBank through the lens of NLI. We thank the members of the Clippers group at The Ohio State University for discussions and suggestions, as well as the reviewers for their feedback. This material is based upon work supported by the National Science Foundation under Grant No. 1845122.

## References


Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112.

Valentine Hacquard and Jeffrey Lidz. 2019. Children's attitude problems: Bootstrapping verb meaning from syntax and pragmatics. *Mind & Language*, 34(1):73–96.

Lauri Karttunen. 1971. Some observations on factivity. *Papers in Linguistics*, 4:55–69.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*.

Nelson F. Liu, Roy Schwartz, and Noah A. Smith. 2019. Inoculation by fine-tuning: A method for analyzing challenge datasets. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2171–2179, Minneapolis, Minnesota. Association for Computational Linguistics.

Marie-Catherine de Marneffe, Mandy Simons, and Judith Tonhauser. 2019. The CommitmentBank: Investigating projection in naturally occurring discourse. In *Sinn und Bedeutung 23*.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Rose, and Graham Neubig. 2018. Stress test evaluation for natural language inference. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2340–2353, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Nikita Nangia and Samuel R. Bowman. 2019. Human vs. muppet: A conservative estimate of human performance on the GLUE benchmark. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4566–4575, Florence, Italy. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019a. SuperGLUE: A stickier benchmark for general-purpose language understanding systems.

Alex Wang, Ian F. Tenney, Yada Pruksachatkun, Katherin Yu, Jan Hula, Patrick Xia, Raghu Pappagari, Shuning Jin, R. Thomas McCoy, Roma Patel, Yinghui Huang, Jason Phang, Edouard Grave, Najoung Kim, Phu Mon Htut, Thibault F'evry, Berlin Chen, Nikita Nangia, Haokun Liu, Anhad Mohananey, Shikha Bordia, Nicolas Patry, Ellie Pavlick, and Samuel R. Bowman. 2019b. `jiant` 1.1: A software toolkit for research on general-purpose text understanding models. `http://jiant.info/`.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

# Incorporating Domain Knowledge into Medical NLI using Knowledge Graphs

**Soumya Sharma**[*] soumyasharma20[†]
**Bishal Santra**[*] bsantraigi[†]
**Abhik Jana** abhikjana1[†]
**T Y S S Santosh** santoshtyss[†]
**Niloy Ganguly** niloy[‡]
**Pawan Goyal** pawang[‡]

Indian Institute of Technology Kharagpur, India
{†}@gmail.com, {‡}@cse.iitkgp.ac.in

## Abstract

Recently, biomedical version of embeddings obtained from language models such as BioELMo have shown state-of-the-art results for the textual inference task in the medical domain. In this paper, we explore how to incorporate structured domain knowledge, available in the form of a knowledge graph (UMLS), for the Medical NLI task. Specifically, we experiment with fusing embeddings obtained from knowledge graph with the state-of-the-art approaches for NLI task, which mainly rely on contextual word embeddings. We also experiment with fusing the domain-specific sentiment information for the task. Experiments conducted on MedNLI dataset clearly show that this strategy improves the baseline BioELMo architecture for the Medical NLI task[1].

## 1 Introduction

Natural language inference (NLI) is one of the basic natural language understanding tasks which deals with detecting inferential relationship such as entailment or contradiction, between a given premise and a hypothesis. In recent years, with the availability of large annotated datasets like SNLI (Bowman et al., 2015), MultiNLI (Williams et al., 2018), researchers have come up with several neural network based models which could be trained with these large annotated datasets and are able to produce state-of-the-art performances (Bowman et al., 2015, 2016; Munkhdalai and Yu, 2017; Sha et al., 2016; Chen et al., 2017; Tay et al., 2018). With these attempts, even though NLI in domains like fiction, travel etc. has progressed a lot, NLI in medical domain is yet to be explored extensively. With the introduction of MedNLI (Romanov and Shivade, 2018), an expert annotated dataset for NLI in the clinical domain, researchers have started pursuing the problem of clinical NLI. Modeling informal inference is one of the basic tasks towards achieving natural language understanding, and is considered very challenging. MedNLI is a dataset that assists in assessing how good a sentence or word embedding method is for downstream uses in medical domain.

Recently, with the emergence of advanced contextual word embedding methods like ELMo (Peters et al., 2018) and BERT (Devlin et al., 2018), performances of many NLP tasks have improved, setting state-of-the-art performances. Following this stream of literature, Lee et al. (2019) introduce BioBERT, which is a BERT model pretrained on English Wikipedia, BooksCorpus and fine-tuned on PubMed (7.8B tokens in total) corpus, PMC full-text articles. Jin et al. (2019) propose BioELMo which is a domain-specific version of ELMo trained on 10M PubMed abstracts, and attempt to solve medical NLI problem with these domain specific embeddings, leading to state-of-the-art performance. These two attempts show a direction towards solving medical NLI problem where the pretrained embeddings are fine-tuned on medical corpus and are used in the state-of-the-art NLI architecture.

Chen et al. (2018) proposed the use of external knowledge to help enrich neural-network based NLI models by applying Knowledge-enriched co-attention, Local inference collection with Exter-

[*] equal contribution
[1] https://github.com/soummyaah/KGMedNLI/



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6092–6097, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

nal Knowledge, and Knowledge-enchanced inference composition components. Another line of solution tries to bring in the extra domain knowledge from sources like Unified Medical Language System (UMLS) (Bodenreider, 2004). Romanov and Shivade (2018) used the knowledge-directed attention based methods in (Chen et al., 2018) for Medical NLI. Another such attempt is made by Lu et al. (2019), where they incorporate domain knowledge in terms of the definitions of medical concepts from UMLS with the state-of-the-art NLI model ESIM (Chen et al., 2017) and vanilla word embeddings of Glove (Pennington et al., 2014) and fastText (Bojanowski et al., 2017). Even though, the authors achieve significant improvement by incorporating only concept definitions from UMLS, the features of this clinical knowledge graph are yet to be fully utilized. Motivated by the emerging trend of embedding knowledge graphs to encode useful information in a high dimensional vector space, we propose the idea of applying state-of-the-art knowledge graph embedding algorithm on UMLS and use these embeddings as a representative of additional domain knowledge with the state-of-the-art medical NLI models like BioELMo, to investigate the performance improvement on this task. Additionally, we also incorporate the sentiment information for medical concepts given by MetaMap (Aronson and Lang, 2010) leading to further improvement of the performance. Note that, as state-of-the-art baselines, we use the models proposed by Jin et al. (2019) and Lu et al. (2019). Since, both of these studies consider ESIM as the core NLI model which makes it more convenient for us to incorporate extra domain knowledge and to have a fair performance comparison with these state-of-the-art models. Our contributions are two-fold.

- We incorporate domain knowledge via knowledge graph embeddings applied on UMLS. We propose an intelligent path-way to combine contextual word embeddings with the domain specific embeddings learnt from the knowledge graph, and feed them to the state-of-the-art NLI models like ESIM. This helps to improve the performance of the base architecture.

- We further show the usefulness of the associated sentiments per medical concept from UMLS in boosting the performance further, which in a way shows that if we can carefully use the domain knowledge present in sources like UMLS, it can lead to promising results as far as the medical NLI task is concerned.

## 2 Dataset

In this study, we use the MedNLI dataset (Romanov and Shivade, 2018), a well-accepted dataset for natural language inference in clinical domain. The dataset is sampled from doctors' notes in the clinical dataset MIMIC-III (Alistair EW Johnson and Mark., 2016) and is arguably the largest publicly available database of patient records. The entire dataset consists of 14,049 premise-hypothesis pairs divided into 11,232 train pairs, 1,395 validation pairs and 1,422 test pairs. Each such pair consists of a gold label which could be either entailment (true), contradiction (false), or neutral (undetermined). The average (maximum) sentence lengths of premises and hypotheses are 20 (202) and 5.8 (20), respectively.

## 3 Proposed Approach

The task is to classify the given premise ($p$) and hypothesis ($h$) sentence pair into one of the three classes: entailment, contradiction and neutral. Following the approach by Jin et al. (2019), we use the BioELMo embedding model where authors bring in contextual information in terms of embeddings obtained via applying ELMo trained on 10M PubMed abstracts, and use these with the state-of-the-art ESIM model (Chen et al., 2017) for the NLI task. The architecture includes two sentence encoders each of which takes in as input the word embeddings of $p$ and $h$. The inputs are run through corresponding bi-directional LSTM encoder layers. Pairwise attention matrix is computed between encoded $p$ and $h$, which forms the attention layer followed by a second bi-directional LSTM layer run separately over $p$ and $h$. Max and average pooling are performed over the outputs of LSTM layers and the output of pooling operations is run through a softmax model. We feed this architecture an additional domain knowledge from UMLS as vector representations obtained via knowledge graph embeddings, the details of which are described below.

**UMLS:** Unified Medical Language System (UMLS) is a compendium which includes many health and biomedical vocabularies and standards. It provides a mapping structure between these

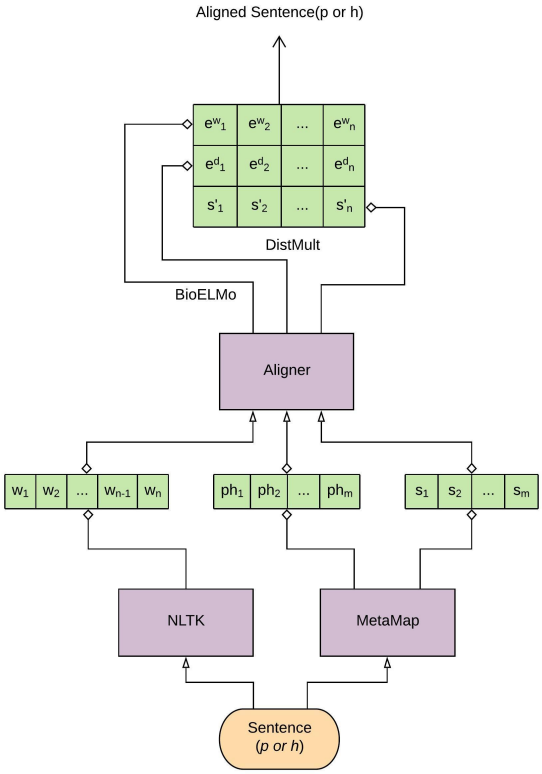


Figure 1: **BioELMo w/ KG pipeline** Here $w$ is the NLTK tokenized form of premise (p) or hypothesis (h), $ph$ is the MetaMap tokenized form of sentence (p or h). $s$ signifies the sentiment vector. $e^w$ and $e^d$ signify the aligned word embeddings and DistMult embeddings, respectively. $s'$ signifies the aligned sentiment vector.

vocabularies and is a comprehensive thesaurus and ontology of biomedical concepts. UMLS contains 3 knowledge sources: Metathesaurus, Semantic Network, and Specialist Lexicon and Lexical Tools. We use two of these sources: the Metathesaurus and the Semantic Network. The Metathesaurus comprises of over 1 million biomedical concepts and 5 million concept names. Each concept has numerous relationships with each other. Each concept in the Metathesaurus is assigned one or more Semantic Type linked to other Semantic Types through a semantic relationship. This information is provided in the Semantic Network of UMLS. There are 127 semantic types and 54 relationships in total. Semantic types include "disease", "symptom", "laboratory test" and semantic relationships include "is-a", "part-of", "affects".

**MetaMap:** MetaMap is a tool for effective entity mapping of biomedical text to the concepts and

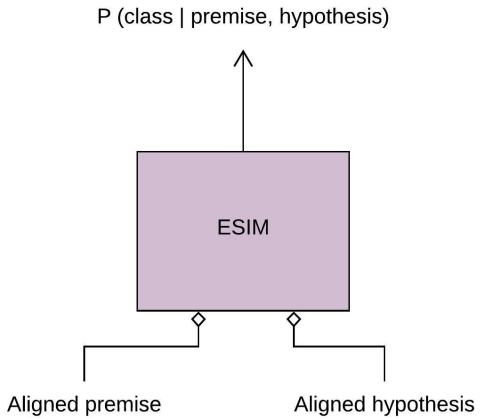


Figure 2: Aligned premise and hypothesis embeddings are obtained using the method described in Figure 1 and become inputs to the ESIM model.

semantic type in UMLS Metathesaurus. On feeding a sentence to MetaMap, it divides the sentence into phrases based on medical concepts found in the sentence and for each medical concept it provides its unique ID in Metathesaurus, its position in the sentence, the list of semantic types the concept is mapped to, the preferred medical name and unique ID for the preferred concept (such as a concept called "chest pain" would be mapped to its preferred medical term "angina"). We also get a boolean value associated with each concept denoting whether the medical concept occurs in a negative sentiment (1) or not (0). For example, in the sentence, "The patient showed no signs of pain", the medical concept 'pain' would appear with a negative sentiment. Note that, for each extracted phrase, there may be more than one related medical concepts and each concept may have more than one possible mapping. For our study, we only consider the mapping with the highest MetaMap Indexing (MMI) score, a metric provided by MetaMap. As a result, every word in a sentence has zero or one corresponding medical concept.

**Constructing the appropriate knowledge graph:** We use the MetaMap tool to process the complete MedNLI dataset and extract the relevant information from UMLS into a smaller knowledge graph.

First, we use MetaMap to extract medical concepts from $p$ and $h$, and map them to the standard terminology in UMLS. We choose to map each medical concept to its preferred medical term.

E.g., "blood clots" would map to "thrombus". This helps us to map different synonymous surface forms to the same concept. This results in 7,496 unique medical concepts from UMLS matched to various words and phrases in the MedNLI dataset. Each unique concept in UMLS becomes a node in our knowledge graph. The relations in our knowledge graph come from two sources: The Metathesaurus and the Semantic Network of UMLS.

Using relations extracted from these two sources, we connect the filtered medical concepts from UMLS to build a smaller Knowledge Graph (subgraph of UMLS).

We get 117,467 triples from the Metathesaurus and 23,824,105 triples from the Semantic Network, which constitute the edgelists in the prepared knowledge graph.

**Knolwedge Graph Embeddings:** To obtain the embedding from this graph, we use state-of-the-art DistMult model (Bishan Yang and Deng, 2015). The choice is inspired by Kadlec et al. (2017), which reports that an appropriately tuned DistMult model can produce similar or better performance while compared with the competing knowledge graph embedding models. DistMult model embeds entities (nodes) and relations (edges) as vectors. It uses a matrix dot product to measure compatibility between head ($h$) and tail ($t$) entities, connected by a relation ($r$). Logistic loss is used for training the model.

$$\sigma_{DistMult}(h, r, t) = r^T(h \cdot t) \quad (1)$$

**Combining Knolwedge Graph Embeddings with BioELMo:** As explained in Figure 1, each sentence ($p$ or $h$) is tokenized using the simple module of NLTK[2] as well as processed using MetaMap to get UMLS concepts. To align these, we copy the UMLS concept for a phrase to all the constituent words.

Once we have aligned the tokens obtained via NLTK and MetaMap, we apply BioELMo and DistMult to get the embedding vectors, $e_{BioELMo,w}$ and $e_{DistMult,w}$ for each word $w$. We concatenate these vectors as $e_w = e_{BioELMo,w} \oplus e_{DistMult,w}$, to obtain the word representation for $w$. We call the proposed model which uses these embeddings as *BioELMo w/ KG*.

**Combining Sentiment Information:** We further enhance the domain knowledge by incorporating sentiment information for a concept separately. For that purpose, we use the sentiment boolean provided by MetaMap and create a 1-d vector $(sent_w)$ containing 0 for positive medical concepts or non-medical concept and 1 for negative concept. This 1-d vector is aligned with the $e_{DistMult,w}$ in the same fashion as explained above. We concatenate this single dimension with our concatenated resultant embeddings. Thus $e_w = e_{BioELMo,w} \oplus e_{DistMult,w} \oplus sent_w$. We call the proposed model which uses these embeddings as *BioELMo w/ KG + Sentiment*.

We use the vanilla ESIM model (Chen et al., 2017) and feed the obtained concatenated embeddings for each word in the premise and hypothesis to the model, to be trained for the inference task (see Figure 2).

| Model | Accuracy |
|---|---|
| fastText | 68.7% |
| GloVe | 73.1% |
| $BioELMo$ (Jin et al., 2019) | 78.2% |
| $ESIMw/K$ (Lu et al., 2019) | 77.8% |
| fastText w/ KG+Sentiment | 73.67% |
| GloVe w/ KG+Sentiment | 74.46% |
| BioELMo w/ KG | 78.76% |
| **BioELMo w/ KG+Sentiment** | **79.04%** |

Table 1: Performance of our models (bottom four) along with the state-of-the-art baseline models (top four). Baseline results for fastText, GloVe are obtained from Romanov and Shivade (2018). Adding knowledge graph information to the base models showed an absolute improvement of 4.97% in case of fastText and 1.36% in case of GloVe. The baseline model utilizing BioELMo as base embeddings (Jin et al., 2019) showed an accuracy of 78.2%. On adding knowledge graph information, we were able to improve these results to 78.76% and on further addition of sentiment information, the accuracy rose to 79.04%

## 4 Experimental Results and Analysis

As discussed earlier, we mainly consider the models presented by Jin et al. (2019) [$BioELMo$] and Lu et al. (2019) [$ESIMw/K$] as our baselines. We report accuracy as the performance metric. Table 1 represents the performance comparison of our proposed models and the baselines, which shows that incorporation of knowledge graph embeddings helps to improve the model performance. All the results reported use ESIM as their base model. Further, incorporating sentiment of medical concepts gives further improvements,

[2] https://www.nltk.org/

achieving an overall 1% improvement over the baseline model.

We also see from (Jin et al., 2019) that BERT and BioBERT show an accuracy of 77.8% and 81.7%, respectively. However, they also showcase through a probing task that BioELMo is a better feature extractor than BioBERT, even though the latter has higher performance when fine tuned on MedNLI. Due to this reason, we take BioELMo as our base architecture and use our enhancements over BioELMo instead of BioBERT.

We also experimented with using fastText and GloVe as our base general embeddings. With addition of Knowledge Graph embeddings and sentiment information, the results showed an absolute improvement from 68.7% to 73.67% in case of fastText, and 73.1% to 74.46% in case of GloVe. All results are summarized in Table 1.

**Training Details:** For *DistMult*, we use word embeddings dimensions to be 100. SGD was used for optimization with an initial learning rate of $10^{-4}$. The batch size was set to 100. For *ESIM*, we take the dimension of hidden states of BiLSTMs to be 500. We set the dropout to 0.5 and choose an initial learning rate of $10^{-3}$. We choose a batch size of 32 and run for a maximum of 64 epochs. The training is stopped when the development loss does not decrease after 5 subsequent epochs.

**Qualitative Analysis:** We explain the efficacy of our model with the help of a few examples. Consider the sentence pair, $p$: "History of CVA" and $h$: "patient has history of stroke". In medical terms, 'CVA' means 'Cerebrovascular accident' which is another term for 'stroke'. By Using MetaMap, we are able to find that the preferred term for 'stroke' is 'Cerebrovascular accident' and hence our model classified the sample pair correctly as entailment.

In many cases, our baseline models fail to capture negative sentiment present in the premise or hypothesis. For example, in case of $p$: Watermelon stomach with gastric varices, without bleed in more than 2 years, $h$: Patient has no active bleeding, BioELMo predicts this as contradiction, whereas the gold label is entailment. But, by using the negative sentiment, captured by Metamap, for the word 'bleed' in both premise and hypothesis, our model is able predict the label correctly.

On the other hand, even though our model produces promising improvement over state-of-the-art performance, there are cases for which our model is not able to classify correctly. For the sentence pair $p$: "She was speaking normally at that time" and $h$: "The patient has no known normal time where she was speaking normally." contradicting each other, our model predicts this to be entailment. The probable reason could be that, the negative sentiment associated with "speaking normally" is not captured by MetaMap and the noise in MetaMap is percolated further. In another example case, $p$: "He had no EKG changes and first set of enzymes were negative." and $h$: "the patient has negative enzymes." our model classifies this pair as entailment while the gold label is neutral. While the premise says that the first set of enzymes was negative, it gives no information about the current state. This leads us to believe that a sense of timeline is extremely important for examples like this which is not already being captured by our model. Taking care of these cases would be our immediate future work.

## 5 Conclusion

In this paper, we showed that knowledge graph embeddings obtained through applying state-of-the-art model like DistMult from UMLS could be a promising way towards incorporating domain knowledge leading to improved state-of-the-art performance for the medical NLI task. We further showed that sentiments of medical concepts can contribute to medical NLI task as well, opening a new direction to be explored further. With the emergence of knowledge graphs in different domains, the proposed approach can be tried out in other domains as well for future exploration.

## Acknowledgements

The authors would like to acknowledge the funding and support from Samsung Research Institute, Delhi (SRID), India.

## References

Lu Shen H Lehman Li-wei Mengling Feng Mohammad Ghassemi Benjamin Moody Peter Szolovits Leo Anthony Celi Alistair EW Johnson, Tom J Pollard and Roger G Mark. 2016. Mimic-iii, a freely accessible critical care database. *Scientific data, 3:160035.*

A. R. Aronson and F.-M. Lang. 2010. An overview of metamap: Historical perspective and recent advances. *J. Amer. Med. Inform. Assoc.*, 17:229–236.

Xiaodong He Jianfeng Gao Bishan Yang, Wen-tau Yih and Li Deng. 2015. Embedding entities and rela-

tions for learning and inference in knowledge bases. *ICLR*.

Olivier Bodenreider. 2004. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl_1):D267–D270.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Samuel R Bowman, Raghav Gupta, Jon Gauthier, Christopher D Manning, Abhinav Rastogi, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. In *54th Annual Meeting of the Association for Computational Linguistics, ACL 2016*. Association for Computational Linguistics (ACL).

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Diana Inkpen, and Si Wei. 2018. Neural natural language inference models enhanced with external knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2406–2417, Melbourne, Australia. Association for Computational Linguistics.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced lstm for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1657–1668.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Qiao Jin, Bhuwan Dhingra, William W. Cohen, and Xinghua Lu. 2019. Probing biomedical embeddings from language models. *CoRR*, abs/1904.02181.

Rudolf Kadlec, Ondrej Bajgar, and Jan Kleindienst. 2017. Knowledge base completion: Baselines strike back. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 69–74, Vancouver, Canada. Association for Computational Linguistics.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *CoRR*, abs/1901.08746.

Mingming Lu, Yu Fang, Fengqi Yan, and Maozhen Li. 2019. Incorporating domain knowledge into natural language inference on clinical texts. *IEEE Access*.

Tsendsuren Munkhdalai and Hong Yu. 2017. Neural tree indexers for text understanding. In *Proceedings of the conference. Association for Computational Linguistics. Meeting*, volume 1, page 11. NIH Public Access.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237.

Alexey Romanov and Chaitanya Shivade. 2018. Lessons from natural language inference in the clinical domain. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1586–1596, Brussels, Belgium. Association for Computational Linguistics.

Lei Sha, Baobao Chang, Zhifang Sui, and Sujian Li. 2016. Reading and thinking: Re-read lstm unit for textual entailment recognition. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2870–2879.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Compare, compress and propagate: Enhancing neural architectures with alignment factorization for natural language inference. In *Proceedings of EMNLP 2018*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122.

# The FLORES Evaluation Datasets for Low-Resource Machine Translation: Nepali–English and Sinhala–English

**Francisco Guzmán**[♡♦] **Peng-Jen Chen**[♡★] **Myle Ott**[★] **Juan Pino**[♦]
**Guillaume Lample**[★‡] **Philipp Koehn**[■] **Vishrav Chaudhary**[♦] **Marc'Aurelio Ranzato**[★]
[♦]Facebook Applied Machine Learning [★]Facebook AI Research
[‡]Sorbonne Universités [■]Johns Hopkins University
{fguzman,pipibjc,myleott,juancarabina,guismay,vishrav,ranzato}@fb.com
phi@jhu.edu

## Abstract

For machine translation, a vast majority of language pairs in the world are considered low-resource because they have little parallel data available. Besides the technical challenges of learning with limited supervision, it is difficult to *evaluate* methods trained on low-resource language pairs because of the lack of freely and publicly available benchmarks. In this work, we introduce the FLORES evaluation datasets for Nepali–English and Sinhala–English, based on sentences translated from Wikipedia. Compared to English, these are languages with very different morphology and syntax, for which little out-of-domain parallel data is available and for which relatively large amounts of monolingual data are freely available. We describe our process to collect and cross-check the quality of translations, and we report baseline performance using several learning settings: fully supervised, weakly supervised, semi-supervised, and fully unsupervised. Our experiments demonstrate that current state-of-the-art methods perform rather poorly on this benchmark, posing a challenge to the research community working on low-resource MT. Data and code to reproduce our experiments are available at https://github.com/facebookresearch/flores.

## 1 Introduction

Research in Machine Translation (MT) has seen significant advances in recent years thanks to improvements in modeling, and in particular neural models (Sutskever et al., 2014; Bahdanau et al., 2015; Gehring et al., 2016; Vaswani et al., 2017), as well as the availability of large parallel corpora for training (Tiedemann, 2012; Smith et al., 2013; Bojar et al., 2017). Indeed, modern neural MT systems can achieve near human-level translation performance on language pairs for which sufficient parallel training resources exist (e.g., Chinese–English translation (Hassan et al., 2018) and English–French translation (Gehring et al., 2016; Ott et al., 2018a).

Unfortunately, MT systems, and in particular neural models, perform poorly on *low-resource* language pairs, for which parallel training data is scarce (Koehn and Knowles, 2017). Improving translation performance on low-resource language pairs could be very impactful considering that these languages are spoken by a large fraction of the world population.

Technically, there are several challenges to solve in order to improve translation for low-resource languages. First, in face of the scarcity of clean parallel data, MT systems should be able to use any source of data available, namely monolingual resources, noisy comparable data, as well as parallel data in related languages. Second, we need reliable public evaluation benchmarks to track progress in translation quality.

Building evaluation sets on low-resource languages is both expensive and time-consuming because the pool of professional translators is limited, as there are few fluent bilingual speakers for these languages. Moreover, the quality of professional translations for low-resource languages is not on par with that of high-resource languages, given that the quality assurance processes for the low-resource languages are often lacking or under development. Also, it is difficult to verify the quality of the human translations as an non-native speaker, because the topics of the documents in these low-resource languages may require knowl-

[♡] Equal contribution.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6098–6111,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

edge and context coming from the local culture.

In this work, we introduce new evaluation benchmarks on two very low-resource language pairs: Nepali–English and Sinhala–English. Sentences were extracted from Wikipedia articles in each language and translated by professional translators. The datasets we release to the community are composed of a tune set of 2559 and 2898 sentences, a development set of 2835 and 2766 sentences, and a test set of 2924 and 2905 sentences for Nepali–English and Sinhala–English respectively.

In §3, we describe the methodology we used to collect the data as well as to check the quality of translations. The experiments reported in §4 demonstrate that these benchmarks are very challenging for current state-of-the-art methods, yielding very low BLEU scores (Papineni et al., 2002) even using all available parallel data as well as monolingual data or Paracrawl[1] filtered data. This suggests that these languages and evaluation benchmarks can constitute a useful test-bed for developing and comparing MT systems for low-resource language pairs.

## 2 Related Work

There is ample literature on low-resource MT. From the modeling side, one possibility is to design methods that make more effective use of monolingual data. This is a research avenue that has seen a recent surge of interest, starting with semisupervised methods relying on back-translation (Sennrich et al., 2015), integration of a language model into the decoder (Gulcehre et al., 2017; Stahlberg et al., 2018) all the way to fully unsupervised approaches (Lample et al., 2018b; Artetxe et al., 2018), which use monolingual data both for learning good language models and for fantasizing parallel data. Another avenue of research has been to extend the traditional supervised learning setting to a *weakly supervised* one, whereby the original training set is augmented with parallel sentences mined from noisy comparable corpora like Paracrawl. In addition to the challenge of learning with limited supervision, low-resource language pairs often involve distant languages that do not share the same alphabet, or have very different morphology and syntax; accordingly, recent work has begun to explore language-independent lexical representations to improve transfer learning (Gu et al., 2018).

In terms of low-resource datasets, DARPA programs like LORELEI (Strassel and Tracey, 2016) have collected translations on several low-resource languages like English–Tagalog. Unfortunately, the data is only made available to the program's participants. More recently, the Asian Language Treebank project (Riza et al., 2016) has introduced parallel datasets for several low-resource language pairs, but these are sampled from text originating in English and thus may not generalize to text sampled from low-resource languages.

In the past, there has been work on extracting high quality translations from crowd-sourced workers using automatic methods (Zaidan and Callison-Burch, 2011; Post et al., 2012). However, crowd-sourced translations have generally lower quality than professional translations. In contrast, in this work we explore the quality checks that are required to filter *professional* translations of low-resource languages in order to build a high quality benchmark set.

In practice, there are very few publicly available datasets for low-resource language pairs, and often times, researchers *simulate* learning on low-resource languages by using a high-resource language pair like English–French, and merely limiting how much labeled data they use for training (Johnson et al., 2016; Lample et al., 2018a). While this practice enables a framework for easy comparison of different approaches, the real practical implications deriving from these methods can be unclear. For instance, low-resource languages are often distant and often times corresponding corpora are not comparable, conditions which are far from the simulation with high-resource European languages, as has been recently pointed out by Neubig and Hu (2018).

## 3 Methodology & Resulting Datasets

For the construction of our benchmark sets we chose to translate between Nepali and Sinhala into and out of English. Both Nepali and Sinhala are Indo-Aryan languages with a subject-object-verb (SOV) structure. Nepali is similar to Hindi in its structure, while Sinhala is characterized by extensive omissions of arguments in a sentence.

Nepali is spoken by about 20 million people if we consider only Nepal, while Sinhala is spo-

[1] https://paracrawl.eu/

ken by about 17 million people just in Sri Lanka[2]. Sinhala and Nepali have very little publicly available parallel data . For instance, most of the parallel corpora for Nepali–English originate from GNOME and Ubuntu handbooks, and account for about 500K sentence pairs.[3] For Sinhala–English, there are an additional 600K sentence pairs automatically aligned from from OpenSubtitles (Lison et al., 2018). Overall, the domains and quantity of the existing parallel data are very limited. However, both languages have a rather large amount of monolingual data publicly available (Buck et al., 2014), making them perfect candidates to track performance on unsupervised and semi-supervised tasks for Machine Translation.

### 3.1 Document selection

To build the evaluation sets, we selected and professionally translated sentences originating from Wikipedia articles in English, Nepali and Sinhala from a Wikipedia snapshot of early May 2018. To select sentences for translation, we first selected the top 25 documents that contain the largest number of *candidate* sentences in each source language. To this end, we defined candidate sentences[4] as: (*i*) being in the intended source language according to a language-id classifier (Bojanowski et al., 2017)[5], and (*ii*) having sentences between 50 and 150 characters. Moreover, we considered sentences and documents to be inadequate for translation when they contained large portions of untranslatable content such as lists of entities[6]. To avoid such lists we used the following rules: (*i*) for English, sentences have to start with an uppercase letter and end with a period; (*ii*) for Nepali and Sinhala, sentences should not contain symbols such as bullet points, repeated dashes, repeated periods or ASCII characters. The document set, along with the categories of documents is presented in the Appendix, Table 8.

After the document selection process, we randomly sampled 2,500 sentences for each language. From English, we translated into Nepali and Sinhala, while from Sinhala and Nepali, we only translated into English. We requested each string to be translated twice by different translators.

### 3.2 Quality checks

Translating domain-specialized content such as Wikipedia articles from and to low-resource languages is challenging: the pool of available translators is limited, there is limited context available to each translator when translating one string at a time, and some of the sentences can contain code-switching (e.g. text about Buddhism in Nepali or Sinhala can contain Sanskrit or Pali words). As a result, we observed large variations in the level of translation quality, which motivated us to enact a series of automatic and manual checks to filter out poor translations.

We first used automatic methods to filter out poor translations and sent them for rework. Once the reworked translations were received, we sent all translations (original or reworked) that passed the automatic checks to human quality checks. Translations which failed human checks, were disregarded. Only the translations that passed all checks were added to the evaluation benchmark, although some source sentences may have less than two translations. Below, we describe the automatic and manual quality checks that we applied to the datasets.

**Automatic Filtering.** The guiding principles underlying our choice of automatic filters are: (*i*) translations should be fluent (Zaidan and Callison-Burch, 2011), (*ii*) they should be sufficiently different from the source text, (*iii*) translations should be similar to each other, yet not equal; and (*iv*) translations should not be transliterations. In order to identify the vast majority of translation issues we filtered by: (*i*) applying a count-based n-gram language model trained on Wikipedia monolingual data and removing translations that have perplexity above 3000.0 (English translations only), (*ii*) removing translations that have sentence-level char-BLEU score between the two generated translations below 15 (indicating disparate translations) or above 90 (indicating suspiciously similar translations), (*iii*) removing sen-

[2]See https://www.ethnologue.com/language/npi and https://www.ethnologue.com/language/sin.

[3]Nepali has also 4K sentences translated from English Penn Tree Bank at http://www.cle.org.pk/software/ling_resources/UrduNepaliEnglishParallelCorpus.htm, which is valuable parallel data.

[4]We first used HTML markup to split document text into paragraphs. We then used regular expressions to split on punctuation, e.g. full-stop, poorna virama (\u0964) and exclamation marks.

[5]This is a necessary step as many sentences in foreign language Wikipedias may be in English or other languages.

[6]For example, the Academy Awards page: https://en.wikipedia.org/wiki/Academy_Award_for_Best_Supporting_Actor.

tences that contain at least 33% transliterated words, (*iv*) removing translations where at least 50% of words are copied from the source sentence, and (*v*) removing translations that contain more than 50% out-of-vocabulary words or more than 5 total out-of-vocabulary words in the sentences (English translations only). For this, the vocabulary was calculated on the monolingual English Wikipedia described in Table 2.

**Manual Filtering.** We followed a setup similar to *direct assessment* (Graham et al., 2013). We asked three different raters to rate sentences from 0–100 according to the perceived translation quality. In our guidelines, the 0–10 range represents a translation that is completely incorrect and inaccurate, the 70–90 range represents a translation that closely preserves the semantics of the source sentence, while the 90–100 range represents a *perfect* translation. To ensure rating consistency, we rejected any evaluation set in which the range of scores among the three reviewers was above 30 points, and requested a fourth rater to break ties, by replacing the most diverging translation rating with the new one. For each translation, we took the average score over all raters and rejected translations whose scores were below 70.

To ensure that the translations were as fluent as possible, we also designed an Amazon Mechanical Turk (AMT) monolingual task to judge the *fluency* of English translations. Regardless of content preservation, translations that are not fluent in the target language should be disregarded. For this task, we then asked five independent human annotators to rate the fluency of each English translation from 1 (bad) to 5 (excellent), and retained only those above 3.Additional statistics of automatic and manual filtering stages can be found in Appendix.

### 3.3 Resulting Datasets

We built three evaluation sets for each language pair using the data that passed our automatic and manual quality checks: *dev* (tune), *devtest* (validation) and *test* (test). The tune set is used for hyperparameter tuning and model selection, the validation set is used to measure generalization during development, while the test set is used for the final blind evaluation.

To measure performance in both directions (e.g. Sinhala–English and English–Sinhala), we built test sets with mixed original-*translationese* (Baroni and Bernardini, 2005) on the source side. To reduce the effect of the source language on the quality of the resulting evaluation benchmark, direct and reverse translations were mixed at an approximate 50-50 ratio for the *devtest* and *test* sets. On the other hand, the *dev* set was composed of the remainder of the available translations, which were not guaranteed to be balanced. Before selection, the sentences were grouped by document, to minimize the number of documents per evaluation set.

| **orig lang** | *dev* | | *devtest* | | *test* | |
|---|---|---|---|---|---|---|
| | uniq | tot | uniq | tot | uniq | tot |
| **Nepali–English** | | | | | | |
| English | 693 | 1,181 | 800 | 1,393 | 850 | 1,462 |
| Nepali | 825 | 1,378 | 800 | 1,442 | 850 | 1,462 |
| | **1,518** | **2,559** | **1,600** | **2,835** | **1,700** | **2,924** |
| **Sinhala–English** | | | | | | |
| English | 1,123 | 1,913 | 800 | 1,395 | 850 | 1,465 |
| Sinhala | 565 | 985 | 800 | 1,371 | 850 | 1,440 |
| | **1,688** | **2,898** | **1600** | **2,766** | **1700** | **2,905** |

Table 1: Number of unique sentences (*uniq*) and total number of sentence pairs (*tot*) per FLORES test set grouped by their original languages.

In Table 1 we present the statistics of the resulting sets. For Sinhala–English, the *test* set is composed of 850 sentences originally in English, and 850 originally in Sinhala. We have approximately 1.7 translations per sentence. This yielded 1,465 sentence pairs originally in English, and 1,440 originally in Sinhalese, for a total of 2,905 sentences. Similarly, for Nepali–English, the *test* set is composed of 850 sentences originally in English, and 850 originally in Nepali. This yielded 1,462 sentence pairs originally in English and 1,462 originally in Nepali, for a total of 2,924 sentence pairs. The composition of the rest of the sets can be found in Table 1.

In Appendix Table 6, we present the aggregate distribution of topics per sentence for the datasets in Nepali–English and Sinhala–English, which shows a diverse representation of topics ranging from General (e.g. documents about tires, shoes and insurance), History (e.g. documents about history of the radar, the Titanic, etc.) to Law and Sports. This richness of topics increases the difficulty of the set, as it requires models that are rather domain-independent. The full list of documents and topics is also in Appendix, Table 8.

## 4 Experiments

In this section, we first describe the data used for training the models, we then discuss the learning settings and models considered, and finally we report the results of these baseline models on the new evaluation benchmarks.

### 4.1 Training Data

Small amounts of parallel data are available for Sinhala–English and Nepali–English. Statistics can be found in Table 2. This data comes from different sources. Open Subtitles and GNOME/KDE/Ubuntu come from the OPUS repository[7]. Global Voices is an updated version (2018q4) of a data set originally created for the CASMACAT project[8]. Bible translations come from the bible-corpus[9]. The Paracrawl corpus comes from the Paracrawl project[10]. The filtered version (Clean Paracrawl) was generated using the LASER model (Artetxe and Schwenk, 2018) to get the best sentence pairs having 1 million English tokens as specified in Chaudhary et al. (2019). We also contrast this filtered version with a randomly filtered version (Random Paracrawl) with the same number of English tokens. Finally, our multilingual experiments in Nepali use Hindi monolingual (about 5 million sentences) and English-Hindi parallel data (about 1.5 million parallel sentences) from the IIT Bombay corpus[11].

| | Sentences | Tokens |
|---|---|---|
| **Nepali–English** | | |
| *parallel* | | |
| Bible | 62K | 1.5M |
| Global Voices | 3K | 75K |
| Penn Tree Bank | 4K | 88K |
| GNOME/KDE/Ubuntu | 495K | 2M |
| *comparable** | | |
| Unfiltered Paracrawl | 2.2M | 40.6M |
| Clean Paracrawl | 32.9K | 1M |
| Random Paracrawl | 55.3K | 1M |
| *monolingual* | | |
| Wikipedia (en) | 67.8M | 2.0B |
| Common Crawl (ne) | 3.6M | 103.0M |
| Wikipedia (ne) | 92.3K | 2.8M |
| **Sinhala–English** | | |
| *parallel* | | |
| Open Subtitles | 601K | 3.6M |
| GNOME/KDE/Ubuntu | 46K | 151K |
| *comparable** | | |
| Paracrawl | 3.4M | 45.4M |
| Clean Paracrawl | 47K | 1M |
| Random Paracrawl | 74.2K | 1M |
| *monolingual* | | |
| Wikipedia (en) | 67.8M | 2.0B |
| Common Crawl (si) | 5.2M | 110.3M |
| Wikipedia (si) | 155.9K | 4.7M |

Table 2: Parallel, comparable, and monolingual data used in experiments in §4. The number of tokens for parallel and comparable corpora are reported over the English tokens. Monolingual and comparable corpora do not include any sentences from the evaluation sets. *Comparable data from Paracrawl is used only in the weakly-supervised experiments since alignments are noisy.

### 4.2 Training Settings

We evaluate models in four training settings. First, we consider a fully *supervised* training setting using the parallel data listed in Table 2.

Second, we consider a fully *unsupervised* setting, whereby only monolingual data on both the source and target side are used to train the model (Lample et al., 2018b).

Third, we consider a *semi-supervised* setting where we also leverage monolingual data on the target side using the standard back-translation training protocol (Sennrich et al., 2015): we train a backward MT system, which we use to translate monolingual target sentences to the source language. Then, we merge the resulting pairs of noisy (back-translated) source sentences with the original target sentences and add them as additional parallel data for training source-to-target MT system. Since monolingual data is available for both languages, we train backward MT systems in both directions and repeat the back-translation process iteratively (He et al., 2016; Lample et al., 2018a). We consider up to two back-translation iterations. At each iteration we generate back-translations using beam search, which has been shown to perform well in low-resource settings (Edunov et al., 2018); we use a beam width of 5 and individually tune the length-penalty on the *dev* set.

Finally, we consider a *weakly supervised* setting by using a baseline system to filter out Paracrawl data using LASER (Artetxe and Schwenk, 2018) by following the approach similar to Chaudhary et al. (2019), in order to augment the original training set with a possibly larger but noisier set of parallel sentences.

[7] http://opus.nlpl.eu/

[8] http://casmacat.eu/corpus/global-voices.html

[9] https://github.com/christos-c/bible-corpus/

[10] https://paracrawl.eu/

[11] http://www.cfilt.iitb.ac.in/iitb_parallel/

For Nepali only, we also consider training using Hindi data, both in a joint supervised and semi-supervised setting. For instance, at each iteration of the joint semi-supervised setting, we use models from the previous iteration to back-translate English monolingual data into both Hindi and Nepali, and from Hindi and Nepali monolingual data into English. We then concatenate actual parallel data and back-translated data of the same language pair together, and train a new model. We also consider using English-Hindi data in the unsupervised scenario. In that setting, a model is pretrained in an unsupervised way with English, Hindi and Nepali monolingual data using the unsupervised approach by Lample and Conneau (2019), and it is then jointly trained on both the Nepali–English unsupervised learning task and the Hindi-English supervised task (in both directions).

### 4.3 Models & Architectures

We consider both phrase-based statistical machine translation (PBSMT) and neural machine translation (NMT) systems in our experiments. All hyper-parameters have been cross-validated using the dev set. The PBSMT systems use Moses (Koehn et al., 2007), with state-of-the-art settings (5-gram language model, hierarchical lexicalized reordering model, operation sequence model) but no additional monolingual data to train the language model.

The NMT systems use the Transformer (Vaswani et al., 2017) implementation in the Fairseq toolkit (Ott et al., 2019); preliminary experiments showed these to perform better than LSTM-based NMT models. More specifically, in the supervised setting, we use a Transformer architecture with 5 encoder and 5 decoder layers, where the number of attention heads, embedding dimension and inner-layer dimension are 2, 512 and 2048, respectively. In the semi-supervised setting, where we augment our small parallel training data with millions of back-translated sentence pairs, we use a larger Transformer architecture with 6 encoder and 6 decoder layers, where the number of attention heads, embedding dimension and inner-layer dimension are 8, 512 and 4096, respectively. When we use multilingual data, the encoder is shared in the {Hindi, Nepali}–English direction, and the decoder is shared in the English–{Hindi, Nepali}direction. We regularize our models with dropout, label smoothing and weight decay, with the corresponding hyper-parameters tuned independently for each language pair. Models are optimized with Adam (Kingma and Ba, 2015) using $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\epsilon = 1e-8$. We use the same learning rate schedule as Ott et al. (2018b). We run experiments on between 4 and 8 Nvidia V100 GPUs with mini-batches of between 10K and 100K target tokens following Ott et al. (2018b). Code to reproduce our results can be found at https://github.com/facebookresearch/flores.

### 4.4 Preprocessing and Evaluation

We tokenize Nepali and Sinhala using the Indic NLP Library.[12] For the PBSMT system, we tokenize English sentences using the Moses tokenization scripts. For NMT systems, we instead use a vocabulary of 5K symbols based on a joint source and target Byte-Pair Encoding (BPE; Sennrich et al., 2015) learned using the sentencepiece library[13] over the parallel training data. We learn the joint BPE for each language pair over the raw English sentences and tokenized Nepali or Sinhala sentences. We then remove training sentence pairs with more than 250 source or target BPE tokens.

We report detokenized SacreBLEU (Post, 2018) when translating into English, and tokenized BLEU (Papineni et al., 2002) when translating from English into Nepali or Sinhala.

### 4.5 Results

In the supervised setting, PBSMT performed quite worse than NMT, achieving BLEU scores of 2.5, 4.4, 1.6 and 5.0 on English–Nepali, Nepali–English, English–Sinhala and Sinhala–English, respectively. Table 3 reports results using NMT in all the other learning configurations described in §4.2. There are several observations we can make.

First, these language pairs are very difficult, as even supervised NMT baselines achieve BLEU scores less than 8. Second and not surprisingly, the BLEU score is particularly low when translating into the more morphologically rich Nepali and Sinhala languages. Third, unsupervised NMT approaches seem to be ineffective on these distant language pairs, achieving BLEU scores close to 0. The reason for this failure is due to poor initialization of the word embeddings.

[12] https://github.com/anoopkunchukuttan/indic_nlp_library

[13] https://github.com/google/sentencepiece

| | Supervised | | Unsupervised | | Semi-supervised | | | | Weakly supervised |
|---|---|---|---|---|---|---|---|---|---|
| | | +mult. | | + mult. | it. 1 | it. 2 | it 1. + mult. | it 2. + mult. | |
| **English–Nepali** | 4.3 | 6.9 | 0.1 | 8.3 | 6.8 | 6.8 | 8.8 | 8.8 | 5.8 |
| **Nepali–English** | 7.6 | 14.2 | 0.5 | 18.8 | 12.7 | 15.1 | 19.8 | 21.5 | 9.6 |
| **English–Sinhala** | 1.2 | - | 0.1 | - | 5.2 | 6.5 | - | - | 3.1 |
| **Sinhala–English** | 7.2 | - | 0.1 | - | 12.0 | 15.1 | - | - | 10.9 |

Table 3: BLEU scores of NMT using various learning settings on *devtest* (see §3). We report detokenized SacreBLEU (Post, 2018) for {Ne,Si}→En and tokenized BLEU for En→{Ne,Si}.

Poor initialization can be attributed to to the monolingual corpora used to train word embeddings which do not have sufficient number of overlapping strings, and are not comparable (Neubig and Hu, 2018; Søgaard et al., 2018).

Fourth, the biggest improvements are brought by the semi-supervised approach using back-translation, which nearly doubles BLEU for Nepali–English from 7.6 to 15.1 (+7.5 BLEU points) and Sinhala–English from 7.2 to 15.1 (+7.9 BLEU points), and increases +2.5 BLEU points for English–Nepali and +5.3 BLEU points for English–Sinhala.

Fifth, additional parallel data in English-Hindi further improves translation quality in Nepali across all settings. For instance, in the Nepali–English supervised setting, we observe a gain of 6.5 BLEU points, while in the semi-supervised setting (where we back-translate also to and from Hindi) the gain is 6.4 BLEU points. Similarly, in the unsupervised setting, multilingual training with Hindi brings Nepali–English to 3.9 BLEU and English–Nepali to 2.5 BLEU; if however, the architecture is pretrained as prescribed by Lample and Conneau (2019), BLEU score improves to 18.8 BLEU for Nepali–English and 8.3 BLEU for English–Nepali.

Finally, the weakly supervised baseline using the additional noisy parallel data described in §4.1 improves upon the supervised baseline in all four directions. This is studied in more depth in Table 4 for Sinhala–English and Nepali–English. Without any filtering or with random filtering, BLEU score is close to 0 BLEU. Applying the a filtering method based on LASER scores (Artetxe and Schwenk, 2018) provides an improvement over using the unfiltered Paracrawl, of +5.5 BLEU points for Nepali–English and +7.3 BLEU points for Sinhala–English. Adding Paracrawl Clean to the initial parallel data improves performance by +2.0 and +3.7 BLEU points, for Nepali–English and Sinhala–English, respectively.

| Corpora | BLEU | |
|---|---|---|
| | ne–en | si–en |
| Parallel | 7.6 | 7.2 |
| Unfiltered Paracrawl | 0.4 | 0.4 |
| Paracrawl Random | 0.1 | 0.4 |
| Paracrawl Clean | 5.9 | 7.7 |
| Parallel + Paracrawl Clean | 9.6 | 10.9 |

Table 4: Weakly supervised experiments: Adding noisy parallel data from filtered Paracrawl improves translation quality in some conditions. "Parallel" refers to the data described in Table 2.

## 5 Discussion

In this section, we provide an analysis of the performance on the Nepali to English *devtest* set using the semi-supervised machine translation system, see Figure 1. Findings on other language directions are similar.

**Fluency of references:** we observe no correlation between the fluency rating of human references and the quality of translations as measured by BLEU. This suggests that the difficulty of the translation task is not related to the fluency of the references, at least at the current level of accuracy.

**Document difficulty:** we observe that translation quality is similar across all document ids, with a difference of 10 BLEU points between the document that is the easiest and the hardest to translate. This suggests that the random sampling procedure used to construct the dataset was adequate and that no single Wikipedia document produces much harder sentences than others.

**Original vs translationese:** we noticed that documents originating from Nepali are harder to translate than documents originating in English. This holds when performing the evaluation with the supervised MT system: translations of original Nepali sentences obtain 4.9 BLEU while Nepali translationese obtain 9.1 BLEU. This suggests that the existing parallel corpus is closer to English Wikipedia than Nepali Wikipedia.

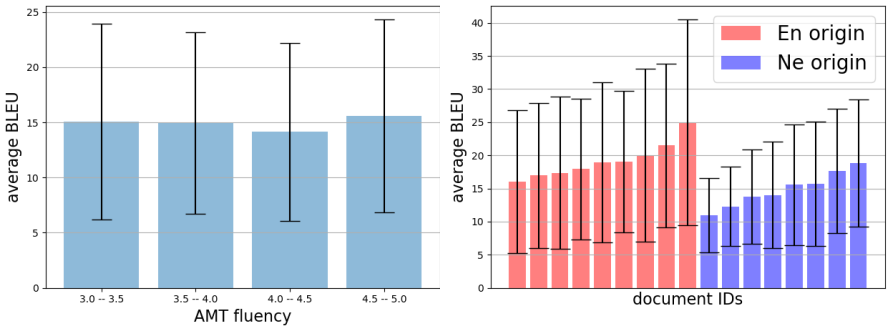


Figure 1: Analysis of the Ne→En *devtest* set using the semi-supervised machine translation system. **Left**: sentence level BLEU versus AMT fluency score of the reference sentences in English; source sentences that have received more fluent human translations are not easier to translate by machines. **Right**: average sentence level BLEU against Wikipedia document id from which the source sentence was extracted; sentences have roughly the same degree of difficulty across documents since there is no extreme difference between shortest and tallest bar. However, source sentences originating from Nepali Wikipedia (blue) are translated more poorly than those originating from English Wikipedia (red). Documents are sorted by BLEU for ease of reading.

### 5.1 Domain drift

To better understand the effect of domain mismatch between the parallel dataset and the Wikipedia evaluation set, we restricted the Sinhala–English training set to only the Open Subtitles portion of the parallel dataset, and we held out 1000 sentences for "in-domain" evaluation of generalization performance. Table 5 shows that translation quality on in-domain data is between 10 and 16 BLEU points higher. This may be due to both domain mismatch as well as sensitivity of the BLEU metric to sentence length. Indeed, there are on average 6 words per sentences in the Open Subtitles test set compared to 16 words per sentence in the FLORES *devtest* set. However, when we train semi-supervised models on back-translated Wikipedia data whose domain better matches the "Out-of-domain" *devtest* set, we see much larger gains in BLEU for the "Out-of-domain" set than we see on the "In-domain" set, suggesting that domain mismatch is indeed a major problem.

| | **Open Subtitles** | **FLORES (*devtest*)** |
|---|---|---|
| **Sinhala–English** | | |
| Supervised | 23.5 | 7.2 |
| Semi-sup. | 28.1 (+20%) | 15.1 (+210%) |
| **English–Sinhala** | | |
| Supervised | 11.0 | 1.2 |
| Semi-sup. | 11.8 (+7%) | 6.5 (+542%) |

Table 5: In-domain vs. out-of-domain translation performance (BLEU) for supervised and semi-supervised NMT models. We report BLEU on a held-out subset of 1,000 sentences from the Open Subtitles training data (see Table 2) and on *devtest* (see §3). Semi-supervised models are trained on back-translated Wikipedia data.

## 6 Conclusions

One of the biggest challenges in MT today is learning to translate low-resource language pairs. Research in this area not only faces formidable technical challenges, from learning with limited supervision to dealing with very distant languages, but it is also hindered by the lack of freely and publicly available evaluation benchmarks.

In this work, we introduce and freely release to the community FLORES benchmarks for Nepali–English and Sinhala–English . Nepali and Sinhala are languages with very different syntax and morphology than English; also, very little parallel data in these language pairs is publicly available. However, a good amount of monolingual data, parallel data in related languages, and Paracrawl data exist in both languages, making these two language pairs a perfect candidate for research on low-resource MT.

Our experiments show that current state-of-the-art approaches perform rather poorly on these new evaluation benchmarks, with semi-supervised and in particular multi-lingual neural methods outperforming all the other model variants and training settings we considered. We perform additional analysis to probe the quality of the datasets. We find no evidence of poor construction quality, yet observe that the low BLEU scores are partly due to the domain mismatch between the training and test datasets. We believe that these benchmarks will help the research community on low-resource MT make faster progress by enabling free access to evaluation data on actual low-resource languages and promoting fair comparison of methods.

## References


Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018. Unsupervised statistical machine translation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Mikel Artetxe and Holger Schwenk. 2018. Massively Multilingual Sentence Embeddings for Zero-Shot Cross-Lingual Transfer and Beyond. *arXiv preprint arXiv:1812.10464*.

D. Bahdanau, K. Cho, and Y. Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*.

Marco Baroni and Silvia Bernardini. 2005. A new approach to the study of translationese: Machine-learning the difference between original and translated text. *Literary and Linguistic Computing*, 21(3):259–274.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, et al. 2017. Findings of the 2017 conference on machine translation (wmt17). In *Proceedings of the Second Conference on Machine Translation*, pages 169–214.

Christian Buck, Kenneth Heafield, and Bas van Ooyen. 2014. N-gram counts and language models from the common crawl. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*. European Language Resources Association (ELRA).

Vishrav Chaudhary, Yuqing Tang, Francisco Guzmn, Holger Schwenk, and Philipp Koehn. 2019. Low-resource corpus filtering using multilingual sentence embeddings. In *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*, pages 263–268, Florence, Italy. Association for Computational Linguistics.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Conference of the Association for Computational Linguistics (ACL)*.

Jonas Gehring, Michael Auli, David Grangier, and Yann N Dauphin. 2016. A convolutional encoder model for neural machine translation. *arXiv preprint arXiv:1611.02344*.

Yvette Graham, Timothy Baldwin, Alistair Moffat, and Justin Zobel. 2013. Continuous measurement scales in human evaluation of machine translation. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 33–41, Sofia, Bulgaria. Association for Computational Linguistics.

Jiatao Gu, Yong Wang, Yun Chen, Victor OK Li, and Kyunghyun Cho. 2018. Meta-learning for low-resource neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3622–3631.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, and Yoshua Bengio. 2017. On integrating a language model into neural machine translation. *Computer Speech & Language*, 45:137–148.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, Shujie Liu, Tie-Yan Liu, Renqian Luo, Arul Menezes, Tao Qin, Frank Seide, Xu Tan, Fei Tian, Lijun Wu, Shuangzhi Wu, Yingce Xia, Dongdong Zhang, Zhirui Zhang, and Ming Zhou. 2018. Achieving human parity on automatic chinese to english news translation. In *arXiv:1803.05567*.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tieyan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In *Advances in Neural Information Processing Systems*, pages 820–828.

M. Johnson, M. Schuster, Q.V. Le, M. Krikun, Y. Wu, Z. Chen, N. Thorat, F. Vigas, M. Wattenberg, G. Corrado, M. Hughes, and J. Dean. 2016. Googles multilingual neural machine translation system: Enabling zero-shot translation. In *Transactions of the Association for Computational Linguistics*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Ondrej Bojar Chris Dyer, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Annual Meeting of the Association for Computational Linguistics (ACL), demo session*.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39.

G. Lample, A. Conneau, L. Denoyer, and M. Ranzato. 2018a. Unsupervised machine translation using monolingual corpora only. In *International Conference on Learning Representations (ICLR)*.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018b. Phrase-based & neural unsupervised machine translation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Pierre Lison, Jörg Tiedemann, and Milen Kouylekov. 2018. Opensubtitles2018: Statistical rescoring of sentence alignments in large, noisy parallel corpora. In *LREC*. European Language Resources Association (ELRA).

Graham Neubig and Junjie Hu. 2018. Rapid adaptation of neural machine translation to new languages. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Brussels, Belgium.

Myle Ott, Michael Auli, David Granger, and Marc'Aurelio Ranzato. 2018a. Analyzing uncertainty in neural machine translation. *arXiv preprint arXiv:1803.00047*.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. 2018b. Scaling neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*.

K. Papineni, S. Roukos, T. Ward, and W.J. Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*.

Matt Post. 2018. A call for clarity in reporting bleu scores. *arXiv*, 1804.08771.

Matt Post, Chris Callison-Burch, and Miles Osborne. 2012. Constructing parallel corpora for six indian languages via crowdsourcing. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 401–409, Montréal, Canada. Association for Computational Linguistics.

Hammam Riza, Michael Purwoadi, Teduh Uliniansyah, Aw Ai Ti, Sharifah Mahani Aljunied, Luong Chi Mai, Vu Tat Thang, Nguyen Phuong Thai, Vichet Chea, Sethserey Sam, et al. 2016. Introduction of the asian language treebank. In *Coordination and Standardization of Speech Databases and Assessment Techniques (O-COCOSDA), 2016 Conference of The Oriental Chapter of International Committee for*, pages 1–6. IEEE.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 86–96.

Jason R Smith, Herve Saint-Amand, Magdalena Plamada, Philipp Koehn, Chris Callison-Burch, and Adam Lopez. 2013. Dirt cheap web-scale parallel text from the common crawl. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1374–1383.

Anders Søgaard, Sebastian Ruder, and Ivan Vulić. 2018. On the limitations of unsupervised bilingual dictionary induction. In *Conference of the Association for Computational Linguistics (ACL)*.

Felix Stahlberg, James Cross, and Veselin Stoyanov. 2018. Simple fusion: Return of the language model. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 204–211. Association for Computational Linguistics.

Stephanie Strassel and Jennifer Tracey. 2016. Lorelei language packs: Data, tools, and resources for technology development in low resource languages. *LREC*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Jrg Tiedemann. 2012. Parallel data, tools and interfaces in opus. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey. European Language Resources Association (ELRA).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Omar F. Zaidan and Chris Callison-Burch. 2011. Crowdsourcing translation: Professional quality from non-professionals. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1220–1229, Portland, Oregon, USA. Association for Computational Linguistics.

# A List of Topics

| topic | proportion (%) | |
|---|---|---|
| | ne–en | si–en |
| General | 18.3 | 24.1 |
| History | 6.5 | 15.1 |
| Science | 7.4 | 12.7 |
| Religion | 8.9 | 10.5 |
| Social Sciences | 10.2 | 6.9 |
| Biology | 6.3 | 9.1 |
| Geography | 10.6 | 4.6 |
| Art/Culture | 6.7 | 8.3 |
| Sports | 5.8 | 6.7 |
| Politics | 8.1 | N/A |
| People | 7.4 | N/A |
| Law | 3.9 | 2.0 |

Table 6: Distribution of the topics of the sentences in the dev, devtest and test sets according to the Wikipedia document they were sampled from.

## B Statistics of automatic filtering and manual filtering

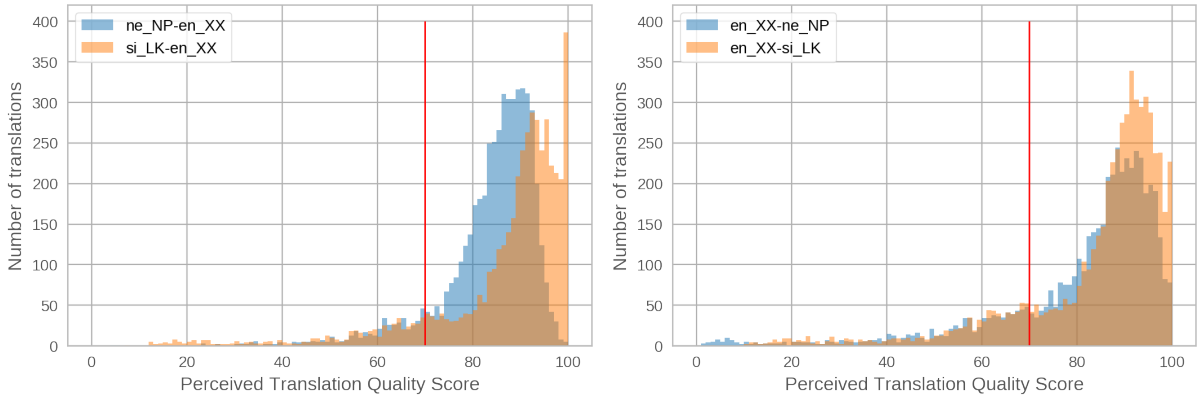


Figure 2: Histogram of averaged translation quality score. We ask three different raters to rate each sentence from 0–100 according to the perceived translation quality. In our guidelines, the 0–10 range represents a translation that is completely incorrect and inaccurate; the 11–29 range represents a translation with few correct keywords, but the overall meaning is different from the source; the 30–50 range represents a translation that contains translated fragments of the source string, with major mistakes; the 51–69 range represents a translation which is understandable and conveys the overall meaning of source string but contains typos or grammatical errors; the 70–90 range represents a translation that closely preserves the semantics of the source sentence; and the 90–100 range represents a *perfect* translation. Translations with averaged translation score less than 70 (red line) are removed from the dataset.

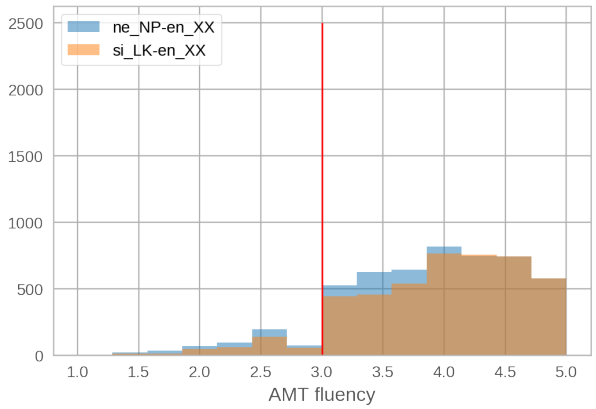


Figure 3: Histogram of averaged AMT fluency score of English translations. We ask five different raters to rate each sentence from 1–5 according to its fluency. In our guidelines, the 1–2 range represents a sentence that is not fluent, 3 is neutral, while the 4–5 range is for fluent sentences that raters can easily understand. Translations with averaged fluency score less than 3 (red line) are removed from the dataset.

| | Nepali–English | English–Nepali | Sinhala–English | English–Sinhala |
|---|---|---|---|---|
| **Automatic filtering** | 14% | 18% | 24% | 7% |
| **Manual filtering** | | | | |
| Translation quality | 10% | 19% | 13% | 16% |
| Fluency | 10% | - | 17% | - |

Table 7: Percentage of translations that did not pass the automatic and manual filtering checks. We first use automatic methods to filter out poor translations and send those translations back for rework. We then collect translations that pass the automatic filtering and send them to two human quality checks, one for adequacy and the other for fluency. Note that the percentage of sentences that did not pass manual filtering is among those sentences that passed the automatic filtering.

## C List of Wikpedia Documents

| domain | document/gloss | topic |
|---|---|---|
| en.wikipedia.org | Astronomy | Science |
| en.wikipedia.org | History of radar | History |
| en.wikipedia.org | Shoe | General |
| en.wikipedia.org | Tire | General |
| en.wikipedia.org | Indian cuisine | Art/Culture |
| en.wikipedia.org | IPhone | General |
| en.wikipedia.org | Apollo program | History |
| en.wikipedia.org | Chess | General |
| en.wikipedia.org | Honey | General |
| en.wikipedia.org | Police | Law |
| en.wikipedia.org | Desert | Geography |
| en.wikipedia.org | Slavery | Social Sciences |
| en.wikipedia.org | Riddler | Art/Culture |
| en.wikipedia.org | Diving | Sports |
| en.wikipedia.org | Cat | Biology |
| en.wikipedia.org | Boxing | Sports |
| en.wikipedia.org | White wine | General |
| en.wikipedia.org | Creativity | Social Sciences |
| en.wikipedia.org | Capitalism | Social Sciences |
| en.wikipedia.org | Alaska | Geography |
| en.wikipedia.org | Museum | General |
| en.wikipedia.org | Lifeguard | General |
| en.wikipedia.org | Tennis | Sports |
| en.wikipedia.org | Writer | General |
| en.wikipedia.org | Anatomy | Science |
| si.wikipedia.org | Qoran | Religion |
| si.wikipedia.org | Dhammas | Religion |
| si.wikipedia.org | Vegetation | Science |
| si.wikipedia.org | Names of Colombo Students | History |
| si.wikipedia.org | Titanic | History |
| si.wikipedia.org | The Heart | Biology |
| si.wikipedia.org | The Ear | Biology |
| si.wikipedia.org | Theravada | Religion |
| si.wikipedia.org | WuZetian | History |
| si.wikipedia.org | Psychoanalisis | Science |
| si.wikipedia.org | Angulimala | Religion |
| si.wikipedia.org | Insurance | General |
| si.wikipedia.org | Leafart | Art/Culture |
| si.wikipedia.org | Communication Science | Science |
| si.wikipedia.org | Pharaoh Neferneferuaten | History |
| ne.wikipedia.org | Nelson Mandela | People |
| ne.wikipedia.org | Parliament of India | Politics |
| ne.wikipedia.org | Kailali and Kanchanpur | Geography |
| ne.wikipedia.org | Bhuwan Pokhari | Geography |
| ne.wikipedia.org | COPD | Biology |
| ne.wikipedia.org | KaalSarp Yoga | Religion |
| ne.wikipedia.org | Research Methodology in Economics | Social Sciences |
| ne.wikipedia.org | Essay | Social Sciences |
| ne.wikipedia.org | Mutation | Science |
| ne.wikipedia.org | Maoist Constituent Assembly | Politics |
| ne.wikipedia.org | Patna | Geography |
| ne.wikipedia.org | Federal rule systen | Law |
| ne.wikipedia.org | Newari Community | Art/Culture |
| ne.wikipedia.org | Raka's Dynasty | History |
| ne.wikipedia.org | Rice | Biology |
| ne.wikipedia.org | Breastfeeding | Biology |
| ne.wikipedia.org | Earthquake | Science |
| ne.wikipedia.org | Motiram Bhatta | People |
| ne.wikipedia.org | Novel Magazine | Art/Culture |
| ne.wikipedia.org | Vladimir Putin | Politics |
| ne.wikipedia.org | History of Nelali Literature | History |
| ne.wikipedia.org | Income tax | Law |
| ne.wikipedia.org | Ravi Prasjal$^+$ | People |
| ne.wikipedia.org | Yogchudamani Upanishads$^+$ | Religion |
| ne.wikipedia.org | Sedai$^+$ | Religion |

Table 8: List of documents by Wikipedia domain, their document name or English translation, and corresponding topics. The document name has an hyper-reference to the original document. $^+$ denotes a page that has been removed or no longer available at the time of this submission.

# D Examples from *devtest*

| | | |
|---|---|---|
| **En→Ne** | | |
| **Source** | | It has automatic spell checking and correction, predictive word capabilities, and a dynamic dictionary that learns new words. |
| **References** | **A** | यसमा स्वचालित हिज्जे जाँच र सुधार छ , भविष्यवाणी शब्द क्षमताहरू , र गतिशील शब्दकोश हुन्छ जसले नयाँ शब्दहरू सिक्छ । |
| | **B** | यसमा स्वचालित हिज्जे जाच्ने तथा सच्याउने , शब्दहरूको अनुमान गर्ने , तथा नयाँ शब्दहरू सिक्ने स्फुर्त शब्दकोश हुन्छ । |
| **System** | | यसमा स्वचालित हिज्जे जाँच र सुधार , पूर्वानुमान शब्द क्षमता र नयाँ शब्द सिक्ने गतिशील शब्दकोश छ । |
| **Source** | | The academic research tended toward the improvement of basic technologies, rather than their specific applications. |
| **References** | **A** | शैक्षिक अनुसन्धानले उनीहरूको विशिष्ट अनुप्रयोगहरूको सट्टा आधारभूत प्रविधिको सुधारको पक्षमा जोड दिए । |
| | **B** | यो शैक्षणिक अनुसन्धान सामान्य प्रविधिको सुधार तर्फ ढलकिएको छ , नाकि तिनिहरूको विशेश प्रयोग तर्फ । |
| **System** | | प्राध्यापक अनुसन्धानले उनीहरूको विशिष्ट अनुप्रयोगमन्दा पनि आधारभूत प्रविधिको सुधारतिर टेवा पुर्यायो । |
| **Ne→En** | | |
| **Source** | | पुरानो समयमा राजालाई सल्लाह दिने सभा ' संसद ' कहलाउँथ्यो । |
| **References** | **A** | In the past, the assembly that advised the king were called 'parliament'. |
| | **B** | In old times the counsil that gave advice to the king was called 'parliament'. |
| **System** | | In old times the council of counsel to the king was 'Senate'. |
| **Source** | | कार्यकर्ताका रुपमा अफ्रिकन नेशनल कंग्रेसमा आबद्ध भए । |
| **References** | **A** | As a worker African Mandela joined the Congress party. |
| | **B** | He joined the African National Congress as a activist. |
| **System** | | As a worker, he joined the African National Congress. |
| **En→Si** | | |
| **Source** | | Iphone users can and do access the internet frequently, and in a variety of places. |
| **References** | **A** | අයිෆෝන් භාවිත කරන්නන්ට නිතරම සහ විවිධ ස්ථානවලදී අන්තර්ජාලයට පිවිසිය හැකිය . |
| | **B** | Iphone පරිශීලකයින් හට නිතරම විවිධ ස්ථානවලදී අන්තර්ජාලය වෙත පිවිසීමට හැකිය . |
| **System** | | අයිෆෝන් භාවිතා කරන්නන්ට පුළුවන් වගේම අන්තර්ජාලයට නිතර පිවිසෙන්නත් පුළුවන් . ඒ වගේම විවිධ තැන්වල |
| **Source** | | In Serious meets, the absolute score is somewhat meaningless. |
| **References** | **A** | සැබෑ තරඟ වලදී ලකුණු සැසඳීම තේරුමක් නැති ක්රියාවකි . |
| | **B** | වැදගත් උළෙල වල , නිරපේක්ෂ ලකුණු තරමක් නිෂ්ඵල ය . |
| **System** | | සීරියස් හමුවේ , නිරපේක්ෂ ලකුණු කිසියම් තේරුමක් නැති |
| **Si→En** | | |
| **Source** | | තර්ජන , ශාරීරික හිංසනය , දේපල හානිය , පහර දීම සහ මරාදැමීම මෙම දඬුවම්ය . |
| **References** | **A** | Threatening, physical violence, property damage, assault and execution are these punishments. |
| | **B** | Threats, bodily violence, property damages, assaults and killing are these punishments. |
| **System** | | Threats, physical harassment, property damage, strike and killing this punishment. |
| **Source** | | අධ්‍ය යාපනයෙන් පසු හෝ පවුලේ යුතුකම් ඉටු කරන්නට හෝ රෝග තත්වයන් නිසා සංඝයා උපසම්පදාවෙන් නිතරම ඉවත් වෙති . |
| **References** | **A** | After education priests leave ordination in order to fulfill duties to the family or due to sickness. |
| | **B** | Sangha is often abandoned because of education or after fulfilling family responsibilities or because of illness. |
| **System** | | After education or to fulfill the family's disease or disease conditions, the companion is often removed from substance. |

Table 9: Examples of sentences from the En-Ne, Ne-En, En-Si and Si-En *devtest* set. System hypotheses (System) are generated using the semi-supervised model described in the main paper using beam search decoding.

# Mask-Predict: Parallel Decoding of Conditional Masked Language Models

**Marjan Ghazvininejad*** **Omer Levy*** **Yinhan Liu*** **Luke Zettlemoyer**
Facebook AI Research
Seattle, WA

## Abstract

Most machine translation systems generate text autoregressively from left to right. We, instead, use a masked language modeling objective to train a model to predict any subset of the target words, conditioned on both the input text and a partially masked target translation. This approach allows for efficient iterative decoding, where we first predict all of the target words non-autoregressively, and then repeatedly mask out and regenerate the subset of words that the model is least confident about. By applying this strategy for a constant number of iterations, our model improves state-of-the-art performance levels for non-autoregressive and parallel decoding translation models by over 4 BLEU on average. It is also able to reach within about 1 BLEU point of a typical left-to-right transformer model, while decoding significantly faster.[1]

## 1 Introduction

Most machine translation systems use sequential decoding strategies where words are predicted one-by-one. In this paper, we present a model and a parallel decoding algorithm which, for a relatively small sacrifice in performance, can be used to generate translations in a *constant* number of decoding iterations.

We introduce conditional masked language models (CMLMs), which are encoder-decoder architectures trained with a masked language model objective (Devlin et al., 2018; Lample and Conneau, 2019). This change allows the model to learn to predict, in parallel, any arbitrary subset of masked words in the target translation. We use transformer CMLMs, where the decoder's self attention (Vaswani et al., 2017) can attend to the entire sequence (left and right context) to predict each masked word. We train with a simple masking scheme where the number of masked target tokens is distributed uniformly, presenting the model with both easy (single mask) and difficult (completely masked) examples. Unlike recently proposed insertion models (Gu et al., 2019; Stern et al., 2019), which treat each token as a separate training instance, CMLMs can train from the entire sequence in parallel, resulting in much faster training.

We also introduce a new decoding algorithm, *mask-predict*, which uses the order-agnostic nature of CMLMs to support highly parallel decoding. Mask-predict repeatedly masks out and re-predicts the subset of words in the current translation that the model is least confident about, in contrast to recent parallel decoding translation approaches that repeatedly predict the entire sequence (Lee et al., 2018). Decoding starts with a completely masked target text, to predict all of the words in parallel, and ends after a constant number of mask-predict cycles. This overall strategy allows the model to repeatedly reconsider word choices within a rich bi-directional context and, as we will show, produce high-quality translations in just a few cycles.

Experiments on benchmark machine translation datasets show the strengths of mask-predict decoding for transformer CMLMs. With just 4 iterations, BLEU scores already surpass the performance of the best non-autoregressive and parallel decoding models.[2]

With 10 iterations, the approach outperforms the current state-of-the-art parallel decod-

*Equal contribution, sorted alphabetically.

[1]Our code is publicly available at: https://github.com/facebookresearch/Mask-Predict

[2]We use the term "parallel decoding" to refer to the family of approaches that can generate the entire target sequence in parallel. These are often referred to as "non-autoregressive" approaches, but both iterative refinement (Lee et al., 2018) and our mask-predict approach condition on the model's past predictions.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6112–6121,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

ing model (Lee et al., 2018) by gaps of 4-5 BLEU points on the WMT'14 English-German translation benchmark, and up to 3 BLEU points on WMT'16 English-Romanian, but with the same model complexity and decoding speed. When compared to standard autoregressive transformer models, CMLMs with mask-predict offer a trade-off between speed and performance, trading up to 2 BLEU points in translation quality for a 3x speed-up during decoding.

## 2 Conditional Masked Language Models

A conditional masked language model (CMLM) predicts a set of target tokens $Y_{mask}$ given a source text $X$ and part of the target text $Y_{obs}$. It makes the strong assumption that the tokens $Y_{mask}$ are conditionally independent of each other (given $X$ and $Y_{obs}$), and predicts the individual probabilities $P(y|X, Y_{obs})$ for each $y \in Y_{mask}$. Since the number of tokens in $Y_{mask}$ is given in advance, the model is also implicitly conditioning on the length of the target sequence $N = |Y_{mask}| + |Y_{obs}|$.

### 2.1 Architecture

We adopt the standard encoder-decoder transformer for machine translation (Vaswani et al., 2017): a source-language encoder that does self-attention, and a target-language decoder that has one set of attention heads over the encoder's output and another set for the target language (self-attention). In terms of parameters, our architecture is identical to the standard one. We deviate from the standard decoder by removing the self-attention mask that prevents left-to-right decoders from attending on future tokens. In other words, our decoder is bi-directional, in the sense that it can use both left and right contexts to predict each token.

### 2.2 Training Objective

During training, we randomly select $Y_{mask}$ among the target tokens. We first sample the number of masked tokens from a uniform distribution between one and the sequence's length, and then randomly choose that number of tokens. Following Devlin et al. (2018), we replace the inputs of the tokens $Y_{mask}$ with a special `MASK` token.

We optimize the CMLM for cross-entropy loss over every token in $Y_{mask}$. This can be done in parallel, since the model assumes that the tokens in $Y_{mask}$ are conditionally independent of each other. While the architecture can technically make predictions over all target-language tokens (including $Y_{obs}$), we only compute the loss for the tokens in $Y_{mask}$.

### 2.3 Predicting Target Sequence Length

In traditional left-to-right machine translation, where the target sequence is predicted token by token, it is natural to determine the length of the sequence dynamically by simply predicting a special `EOS` (end of sentence) token. However, for CMLMs to predict the entire sequence in parallel, they must know its length in advance. This problem was recognized by prior work in non-autoregressive translation, where the length is predicted with a fertility model (Gu et al., 2018) or by pooling the encoder's outputs into a length classifier (Lee et al., 2018).

We follow Devlin et al. (2018) and add a special `LENGTH` token to the encoder, akin to the `CLS` token in BERT. The model is trained to predict the length of the target sequence $N$ as the `LENGTH` token's output, similar to predicting another token from a different vocabulary, and its loss is added to the cross-entropy loss from the target sequence.

## 3 Decoding with Mask-Predict

We introduce the mask-predict algorithm, which decodes an entire sequence in parallel within a constant number of cycles. At each iteration, the algorithm selects a subset of tokens to mask, and then predicts them (in parallel) using an underlying CMLM. Masking the tokens where the model has doubts while conditioning on previous high-confidence predictions lets the model re-predict the more challenging cases, but with more information. At the same time, the ability to make large parallel changes at each step allows mask-predict to converge on a high quality output sequence in a sub-linear number of decoding iterations.

### 3.1 Formal Description

Given the target sequence's length $N$ (see Section 3.3), we define two variables: the target sequence $(y_1, \ldots, y_N)$ and the probability of each token $(p_1, \ldots, p_N)$. The algorithm runs for a predetermined number of iterations $T$, which is either a constant or a simple function of $N$. At each iteration, we perform a *mask* operation, followed by *predict*.

| | |
|---|---|
| $src$ | Der Abzug der franzsischen Kampftruppen wurde am 20. November abgeschlossen . |
| $t = 0$ | The departure of the French combat completed completed on 20 November . |
| $t = 1$ | The departure of French combat troops was completed on 20 November . |
| $t = 2$ | The withdrawal of French combat troops was completed on November 20th . |

Figure 1: An example from the WMT'14 DE-EN validation set that illustrates how mask-predict generates text. At each iteration, the highlighted tokens are masked and repredicted, conditioned on the other tokens in the sequence.

**Mask** For the first iteration ($t = 0$), we mask all the tokens. For later iterations, we mask the $n$ tokens with the lowest probability scores:

$$Y^{(t)}_{mask} = \arg\min_i(p_i, n)$$
$$Y^{(t)}_{obs} = Y \setminus Y^{(t)}_{mask}$$

The number of masked tokens $n$ is a function of the iteration $t$; specifically, we use linear decay $n = N \cdot \frac{T-t}{T}$, where $T$ is the total number of iterations. For example, if $T = 10$, we will mask 90% of the tokens at $t = 1$, 80% at $t = 2$, and so forth.

**Predict** After masking, the CMLM predicts the masked tokens $Y^{(t)}_{mask}$, conditioned on the source text $X$ and the unmasked target tokens $Y^{(t)}_{obs}$. We select the prediction with the highest probability for each masked token $y_i \in Y^{(t)}_{mask}$ and update its probability score accordingly:

$$y^{(t)}_i = \arg\max_w P(y_i = w|X, Y^{(t)}_{obs})$$
$$p^{(t)}_i = \max_w P(y_i = w|X, Y^{(t)}_{obs})$$

The values and the probabilities of unmasked tokens $Y^{(t)}_{obs}$ remain unchanged:

$$y^{(t)}_i = y^{(t-1)}_i$$
$$p^{(t)}_i = p^{(t-1)}_i$$

We tried updating or decaying these probabilities in preliminary experiments, but found that this heuristic works well despite the fact that some probabilities are stale.

### 3.2 Example

Figure 1 illustrates how mask-predict can generate a good translation in just three iterations.

In the first iteration ($t = 0$), the entire target sequence is masked ($Y^{(0)}_{mask} = Y$ and $Y^{(0)}_{obs} = \emptyset$), and is thus generated by the CMLM in a purely non-autoregressive process:

$$P(Y^{(0)}_{mask}|X, Y^{(0)}_{obs}) = P(Y|X)$$

This produces an ungrammatical translation with repetitions ("completed completed"), which is typical of non-autoregressive models due to the multi-modality problem (Gu et al., 2018).

In the second iteration ($t = 1$), we select 8 of the 12 tokens generated in the previous step; these token were predicted with the lowest probabilities at $t = 0$. We mask them and repredict with the CMLM, while conditioning on the 4 unmasked tokens $Y^{(1)}_{obs} = \{$"The", "20", "November", "."$\}$. This results in a more grammatical and accurate translation. Our analysis shows that this second iteration removes most repetitions, perhaps because conditioning on even a little bit of the target sequence is enough to collapse the multi-modal target distribution into a single output (Section 5.1).

In the last iteration ($t = 2$), we select the 4 of the 12 tokens that had the lowest probabilities. Two of those tokens were predicted at the first step ($t = 0$), and not repredicted at the second step ($t = 1$). It is quite common for earlier predictions to be masked at later iterations because they were predicted with less information and thus tend to have lower probabilities. Now that the model is conditioning on 8 tokens, it is able to produce an more fluent translation; "withdrawal" is a better fit for describing troop movement, and "November 20th" is a more common date format in English.

### 3.3 Deciding Target Sequence Length

When generating, we first compute the CMLM's encoder, and then use the `LENGTH` token's encoding to predict a distribution over the target sequence's length (see Section 2.3). Since much of the CMLM's computation can be batched, we select the top $\ell$ length candidates with the highest probabilities, and decode the same example with different lengths in parallel. We then select the sequence with the highest average log-probability as our result:

$$\frac{1}{N}\sum \log p^{(T)}_i$$

Our analysis reveals that translating multiple candidate sequences of different lengths can improve performance (see Section 5.3).

## 4 Experiments

We evaluate CMLMs with mask-predict decoding on standard machine translation benchmarks. We find that our approach significantly outperforms prior parallel decoding machine translation methods and even approaches the performance of standard autoregressive models (Section 4.2), while decoding significantly faster (Section 4.3).

### 4.1 Experimental Setup

**Translation Benchmarks** We evaluate on three standard datasets, WMT'14 EN-DE (4.5M sentence pairs), WMT'16 EN-RO (610k pairs) and WMT'17 EN-ZH (20M pairs) in both directions. The datasets are tokenized into subword units using BPE (Sennrich et al., 2016). We use the same preprocessed data as Vaswani et al. (2017) and Wu et al. (2019) for WMT'14 EN-DE and WMT'17 EN-ZH respectively, and use the data from Lee et al. (2018) for WMT'16 EN-RO. We evaluate performance with BLEU (Papineni et al., 2002) for all language pairs, except from EN to ZH, where we use SacreBLEU (Post, 2018).[3]

**Hyperparameters** We follow most of the standard hyperparameters for transformers in the base configuration (Vaswani et al., 2017): 6 layers per stack, 8 attention heads per layer, 512 model dimensions, 2048 hidden dimensions. We also experiment with 512 hidden dimensions, for comparison with previous parallel decoding models (Gu et al., 2018; Lee et al., 2018). We follow the weight initialization scheme from BERT (Devlin et al., 2018), which samples weights from $\mathcal{N}(0, 0.02)$, initializes biases to zero, and sets layer normalization parameters to $\beta = 0, \gamma = 1$. For regularization, we use $0.3$ dropout, $0.01$ $L_2$ weight decay, and smoothed cross validation loss with $\varepsilon = 0.1$. We train batches of 128k tokens using Adam (Kingma and Ba, 2015) with $\beta = (0.9, 0.999)$ and $\varepsilon = 10^{-6}$. The learning rate warms up to a peak of $5 \cdot 10^{-4}$ within 10,000 steps, and then decays with the inverse square-root schedule. We trained all models for 300k steps, measured the validation loss at the end of each epoch, and averaged the 5 best checkpoints to create the final model. During decoding, we use a beam size of $b = 5$ for autoregressive decoding, and similarly use $\ell = 5$ length candidates for mask-predict decoding. We trained with mixed precision floating point arithmetic on two DGX-1 machines, each with eight 16GB Nvidia V100 GPUs interconnected by Infiniband (Micikevicius et al., 2018).

**Model Distillation** Following previous work on non-autoregressive and insertion-based machine translation (Gu et al., 2018; Lee et al., 2018; Stern et al., 2019), we train CMLMs on translations produced by a standard left-to-right transformer model (large for EN-DE and EN-ZH, base for EN-RO). For a fair comparison, we also train standard left-to-right base transformers on translations produced by large transformer models for EN-DE and EN-ZH, in addition to the standard baselines. We analyze the impact of distillation in Section 5.4.

### 4.2 Translation Quality

We compare our approach to three other parallel decoding translation methods: the fertility-based sequence-to-sequence model of Gu et al. (2018), the CTC-loss transformer of Libovický and Helcl (2018), and the iterative refinement approach of Lee et al. (2018). The first two methods are purely non-autoregressive, while the iterative refinement approach is only non-autoregressive in the first decoding iteration, similar to our approach. In terms of speed, each mask-predict iteration is virtually equivalent to a refinement iteration.

Table 1 shows that among the parallel decoding methods, our approach yields the highest BLEU scores by a considerable margin. When controlling for the number of parameters (i.e. considering only the smaller CMLM configuration), CMLMs score roughly 4 BLEU points higher than the previous state of the art on WMT'14 EN-DE, in both directions. Another striking result is that a CMLM with only 4 mask-predict iterations yields higher scores than 10 iterations of the iterative refinement model; in fact, only 3 mask-predict iterations are necessary for achieving a new state of the art on both directions of WMT'14 EN-DE (not shown).

The translations produced by CMLMs with mask-predict also score competitively when compared to strong transformer-based autoregressive models. In all 4 benchmarks, our base CMLM reaches within 0.5-1.2 BLEU points from a well-tuned base transformer, a relative decrease of less

[3] SacreBLEU hash: BLEU+case.mixed+lang.en-zh +numrefs.1+smooth.exp+test.wmt17+tok.zh+version.1.3.7

| Model | Dimensions (Model/Hidden) | Iterations | WMT'14 EN-DE | WMT'14 DE-EN | WMT'16 EN-RO | WMT'16 RO-EN |
|---|---|---|---|---|---|---|
| NAT w/ Fertility (Gu et al., 2018) | 512/512 | 1 | 19.17 | 23.20 | 29.79 | 31.44 |
| CTC Loss (Libovický and Helcl, 2018) | 512/4096 | 1 | 17.68 | 19.80 | 19.93 | 24.71 |
| Iterative Refinement (Lee et al., 2018) | 512/512 | 1 | 13.91 | 16.77 | 24.45 | 25.73 |
| | 512/512 | 10 | 21.61 | 25.48 | 29.32 | 30.19 |
| (Dynamic #Iterations) | 512/512 | ? | 21.54 | 25.43 | 29.66 | 30.30 |
| *Small CMLM with Mask-Predict* | 512/512 | 1 | 15.06 | 19.26 | 20.12 | 20.36 |
| | 512/512 | 4 | **24.17** | **28.55** | **30.00** | 30.43 |
| | 512/512 | 10 | **25.51** | **29.47** | **31.65** | **32.27** |
| *Base CMLM with Mask-Predict* | 512/2048 | 1 | 18.05 | 21.83 | 27.32 | 28.20 |
| | 512/2048 | 4 | **25.94** | **29.90** | **32.53** | **33.23** |
| | 512/2048 | 10 | **27.03** | **30.53** | **33.08** | **33.31** |
| Base Transformer (Vaswani et al., 2017) | 512/2048 | $N$ | 27.30 | — — | — — | — — |
| Base Transformer (Our Implementation) | 512/2048 | $N$ | 27.74 | 31.09 | 34.28 | 33.99 |
| Base Transformer (+Distillation) | 512/2048 | $N$ | 27.86 | 31.07 | — — | — — |
| Large Transformer (Vaswani et al., 2017) | 1024/4096 | $N$ | 28.40 | — — | — — | — — |
| Large Transformer (Our Implementation) | 1024/4096 | $N$ | 28.60 | 31.71 | — — | — — |

Table 1: The performance (BLEU) of CMLMs with mask-predict, compared to other parallel decoding machine translation methods. The standard (sequential) transformer is shown for reference. Bold numbers indicate state-of-the-art performance among parallel decoding methods.

| Model | Dimensions (Model/Hidden) | Iterations | WMT'17 EN-ZH | WMT'17 ZH-EN |
|---|---|---|---|---|
| *Base CMLM with Mask-Predict* | 512/2048 | 1 | 24.23 | 13.64 |
| | 512/2048 | 4 | 32.63 | 21.90 |
| | 512/2048 | 10 | 33.19 | 23.21 |
| Base Transformer (Our Implementation) | 512/2048 | $N$ | 34.31 | 23.74 |
| Base Transformer (+Distillation) | 512/2048 | $N$ | 34.44 | 23.99 |
| Large Transformer (Our Implementation) | 1024/4096 | $N$ | 35.01 | 24.65 |

Table 2: The performance (BLEU) of CMLMs with mask-predict, compared to the standard (sequential) transformer on WMT' 17 EN-ZH.

than 4% in translation quality. In many scenarios, this is an acceptable price to pay for a significant speedup from parallel decoding.

Table 2 shows that these trends also hold for English-Chinese translation, in both directions, despite major linguistic differences between the two languages.

### 4.3 Decoding Speed

Because CMLMs can predict the entire sequence in parallel, mask-predict can translate an entire sequence in a constant number of decoding iterations. Does this appealing theoretical property translate into a wall-time speed-up in practice? By comparing the actual decoding times, we show that, for some sacrifice in performance, our parallel method can translate much faster than standard sequential transformers.

**Setup** As the baseline system, we use the base transformer with beam search ($b = 5$) to translate WMT'14 EN-DE; we also use greedy search ($b = 1$) as a faster but less accurate baseline. For CMLMs, we vary the number of mask-predict iterations ($T = 4, \ldots, 10$) and length candidates ($\ell = 1, 2, 3$). For both models, we decode batches of 10 sentences.[4] For each decoding run, we measure the performance (BLEU) and wall time (seconds) from when the model and data have been loaded until the last example has been translated, and calculate the relative decoding speed-up (CMLM time / baseline time) to assess the speed-performance trade-off.

The implementation of both the baseline transformer and our CMLM is based on `fairseq` (Gehring et al., 2017), which efficiently decodes left-to-right transformers by caching the state. Caching reduces the baseline's decoding speed from 210 seconds to 128.5; CMLMs do not use cached decoding. All experiments used exactly the same machine and the same single GPU.

[4]The batch size was chosen arbitrarily; mask-predict can scale up to much larger batch sizes.

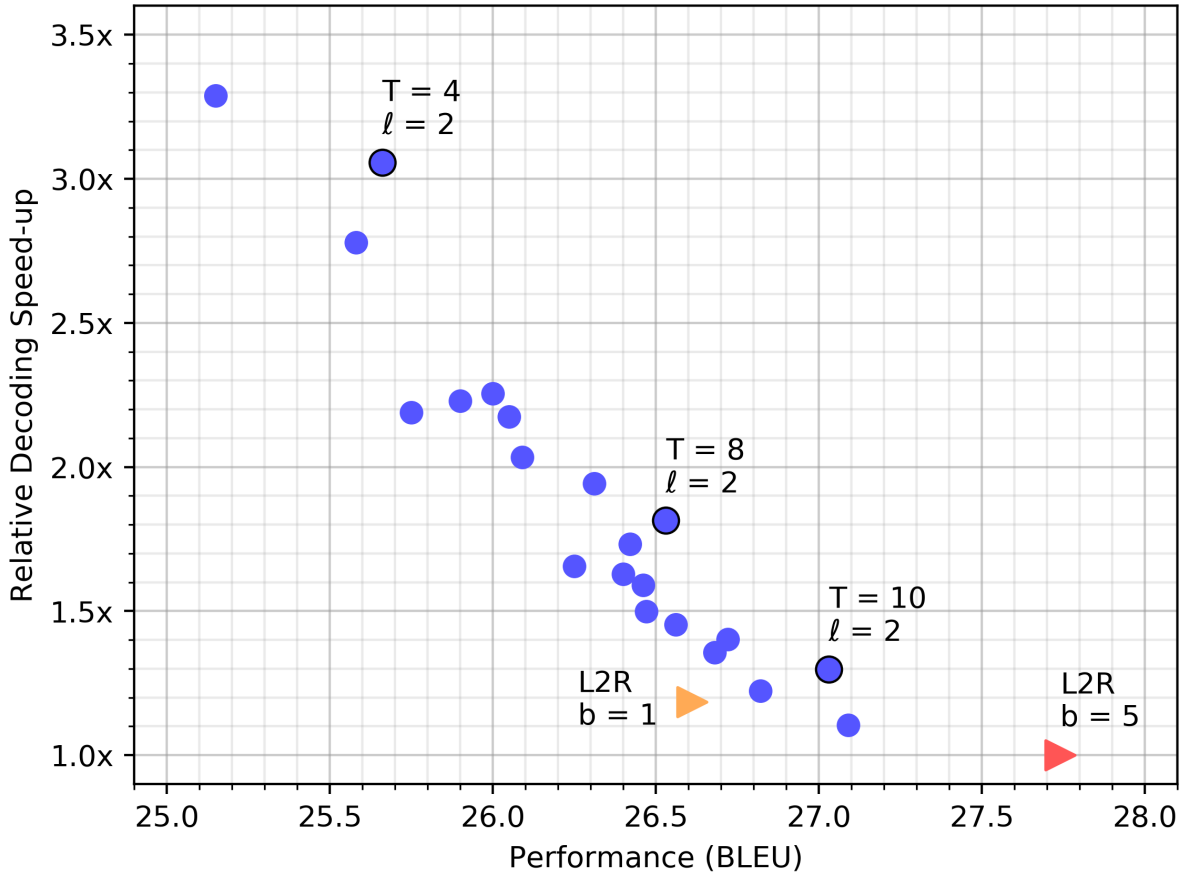


Figure 2: The trade-off between speed-up and translation quality of a base CMLM with mask-predict, compared to the standard sequentially-decoded base transformer on the WMT'14 EN-DE test set, with beam sizes $b = 1$ (orange triangle) and $b = 5$ (red triangle). Each blue circle represents a mask-predict decoding run with a different number of iterations ($T = 4, \ldots, 10$) and length candidates ($\ell = 1, 2, 3$).

**Results** Figure 2 shows the speed-performance trade-off. We see that mask-predict is versatile; on one hand, we can translate over 3 times faster than the baseline at a cost of 2 BLEU points ($T = 4$, $\ell = 2$), or alternatively retain a high quality of 27.03 BLEU while gaining a 30% speed-up ($T = 4$, $\ell = 2$). Surprisingly, this latter configuration outperforms an autoregressive transformer with greedy decoding ($b = 1$) in both quality and speed. We also observe that more balanced configurations (e.g. $T = 8$, $\ell = 2$) yield similar performance to the single-beam autoregressive transformer, but decode much faster.

## 5 Analysis

To complement the quantitative results in Section 4, we present qualitative analysis that provides some intuition as to why our approach works and where future work could potentially improve it.

### 5.1 Why Are Multiple Iterations Necessary?

Various non-autoregressive translation models, including our own CMLM, make the strong assumption that the individual token predictions are conditionally independent of each other. Such a model might consider two or more possible translations, A and B, but because there is no coordination mechanism between the token predictions, it could predict one token from A and another token from B. This problem, known as the multi-modality problem (Gu et al., 2018), often manifest as *token repetitions* in the output when the model has multiple hypotheses that predict the same word $w$ with high confidence, but at different positions.

We hypothesize that multiple mask-predict iterations alleviate the multi-modality problem by allowing the model to condition on parts of the input, thus collapsing the multi-modal distribution into a sharper uni-modal distribution. To test our

| Iterations | WMT'14 EN-DE BLEU | WMT'14 EN-DE Reps | WMT'16 EN-RO BLEU | WMT'16 EN-RO Reps |
|---|---|---|---|---|
| $T = 1$ | 18.05 | 16.72% | 27.32 | 9.34% |
| $T = 2$ | 22.91 | 5.40% | 31.08 | 2.82% |
| $T = 3$ | 24.99 | 2.03% | 32.19 | 1.26% |
| $T = 4$ | 25.94 | 1.07% | 32.53 | 0.87% |
| $T = 5$ | 26.30 | 0.72% | 32.62 | 0.61% |

Table 3: The performance (BLEU) and percentage of repeating tokens when decoding with a different number of mask-predict iterations ($T$).

| | $T = 4$ | $T = 10$ | $T = N$ |
|---|---|---|---|
| $1 \leq N < 10$ | 21.8 | 22.4 | 22.4 |
| $10 \leq N < 20$ | 24.6 | 25.9 | 26.0 |
| $20 \leq N < 30$ | 24.9 | 26.7 | 27.1 |
| $30 \leq N < 40$ | 24.9 | 26.7 | 27.6 |
| $40 \leq N$ | 25.0 | 27.5 | 28.1 |

Table 4: The performance (BLEU) of base CMLM with different amounts of mask-predict iterations ($T$) on WMT'14 EN-DE, bucketed by target sequence length ($N$). Decoding with $\ell = 1$ length candidates.

hypothesis, we measure the percentage of repetitive tokens produced by each iteration of mask-predict as a proxy metric for the multi-modality problem.

Table 3 shows that, indeed, the proportion of repetitive tokens drops drastically during the first 2-3 iterations. This finding suggests that the first few iterations are critical for converging into a uni-modal distribution. The decrease in repetitions also correlates with the steep rise in translation quality (BLEU), supporting the conjecture of Gu et al. (2018) that multi-modality is a major roadblock for purely non-autoregressive machine translation.

### 5.2 Do Longer Sequences Need More Iterations?

A potential concern with using a constant amount of decoding iterations is that it may be effective for short sequences (where the number of iterations $T$ is closer to the output's length $N$), but insufficient for longer sequences. To determine whether this is the case, we use `compare-mt` (Neubig et al., 2019) to bucket the evaluation data by target sentence length and compute the performance with different values of $T$.

Table 4 shows that increasing the number of decoding iterations ($T$) appears to mainly improve the performance on longer sequences. Having said that, the performance differences across length buckets are not very large, and it seems that even 4 mask-predict iterations are enough to produce decent translations for long sequences ($40 \leq N$).

### 5.3 Do More Length Candidates Help?

Traditional autoregressive models can dynamically decide the length of the target sequence by generating a special `END` token when they are done, but that is not true for models that decode multiple tokens in parallel, such as CMLMs. To address this problem, our model predicts the length of the target sequence (Section 2.3) and decodes multiple length candidates in parallel (Section 3.3). We compare our model's performance with a varying number of length candidates to its performance when conditioned on the reference (gold) target length in order to determine how accurate it is at predicting the correct length and assess the relative contribution of decoding with multiple length candidates.

| Length Candidates | WMT'14 EN-DE BLEU | WMT'14 EN-DE LP | WMT'16 EN-RO BLEU | WMT'16 EN-RO LP |
|---|---|---|---|---|
| $\ell = 1$ | 26.56 | 16.1% | 32.75 | 13.8% |
| $\ell = 2$ | 27.03 | 30.6% | 33.06 | 26.1% |
| $\ell = 3$ | **27.09** | 43.1% | **33.11** | 39.6% |
| $\ell = 4$ | **27.09** | 53.1% | 32.13 | 49.2% |
| $\ell = 5$ | 27.03 | 62.2% | 33.08 | 57.5% |
| $\ell = 6$ | 26.91 | 69.5% | 32.91 | 64.3% |
| $\ell = 7$ | 26.71 | 75.5% | 32.75 | 70.4% |
| $\ell = 8$ | 26.59 | 80.3% | 32.50 | 74.6% |
| $\ell = 9$ | 26.42 | 83.8% | 32.09 | 78.3% |
| Gold | 27.27 | — | 33.20 | — |

Table 5: The performance (BLEU) of base CMLM with 10 mask-predict iterations ($T = 10$), varied by the number of length candidates ($\ell$), compared to decoding with the reference target length (Gold). Length precision (LP) is the percentage of examples that contain the correct length as one of their candidates.

Table 5 shows that having multiple candidates can increase performance almost as much as conditioning on the gold length. Surprisingly, adding too many candidates can even degrade performance. We suspect that because CMLMs are implicitly conditioned on the target length, producing a translation that is too short (i.e. high precision, low recall) will have a high average log probability. In preliminary experiments, we tried to address this issue by weighting the different candidates according to the model's length prediction, but this approach gave too much weight to the top candidate and resulted in lower performance.

| Iterations | WMT'14 EN-DE | | WMT'16 EN-RO | |
|---|---|---|---|---|
| | Raw | Dist | Raw | Dist |
| $T = 1$ | 10.64 | **18.05** | 21.22 | **27.32** |
| $T = 4$ | 22.25 | **25.94** | 31.40 | **32.53** |
| $T = 10$ | 24.61 | **27.03** | 32.86 | **33.08** |

Table 6: The performance (BLEU) of base CMLM, trained with either raw data (Raw) or knowledge distillation from an autoregressive model (Dist).

### 5.4 Is Model Distillation Necessary?

Previous work on non-autoregressive and insertion-based machine translation reported that it was necessary to train their models on text generated by an autoregressive teacher model, a process known as distillation. To determine CMLM's dependence on this process, we train a models on both raw and distilled data, and compare their performance.

Table 6 shows that in every case, training with model distillation substantially outperforms training on raw data. The gaps are especially large when decoding with a single iteration (purely non-autoregressive). Overall, it appears as though CMLMs are heavily dependent on model distillation.

On the English-Romanian benchmark, the differences are much smaller, and after 10 iterations the raw-data model can perform comparably with the distilled model. A possible explanation is that our teacher model was weaker for this dataset due to insufficient hyperparameter tuning. Alternatively, it could also be the case that the English-German dataset is much noisier than the English-Romanian one, and that the teacher model essentially cleans the training data. Unfortunately, we do not have enough evidence to support or refute either hypothesis at this time.

## 6 Related Work

**Training Masked Language Models with Translation Data** Recent work by Lample and Conneau (2019) shows that training a masked language model on sentence-pair translation data, as a pre-training step, can improve performance on cross-lingual tasks, including autoregressive machine translation. Our training scheme builds on their work, with the following differences: we use separate model parameters for source and target texts (encoder and decoder), and we also use a different masking scheme. Specifically, we mask a varying percentage of tokens, only from the target, and do not replace input tokens with noise. Most importantly, the goal of our work is different; we do not use CMLMs for pre-training, but to directly generate text with mask-predict decoding.

Concurrently with our work, Song et al. (2019) extend the approach of Lample and Conneau (2019) by using separate encoder and decoder parameters (as in our model) and pre-training them jointly in an autoregressive version of masked language modeling, although with monolingual data. While this work demonstrates that pre-training CMLMs can improve autoregressive machine translation, it does not try to leverage the parallel and bi-directional nature of CMLMs to generate text in a non-left-to-right manner.

**Generating from Masked Language Models** One such approach for generating text from a masked language model casts BERT (Devlin et al., 2018), a *non*-conditional masked language model, as a Markov random field (Wang and Cho, 2019). By masking a sequence of length $N$ and then iteratively sampling a single token at each time from the model (either sequentially or in arbitrary order), one can produce grammatical examples. While this sampling process has a theoretical justification, it also requires $N$ forward passes of the model; mask-predict decoding, on the other hand, can produce text in a constant number of iterations.

**Parallel Decoding for Machine Translation** There have been several advances in parallel decoding machine translation by training non-autoregressive models. Gu et al. (2018) introduce a transformer-based approach with explicit word fertility, and identify the multi-modality problem. Libovický and Helcl (2018) approach the multi-modality problem by collapsing repetitions with the Connectionist Temporal Classification training objective (Graves et al., 2006). Perhaps most similar to our work is the iterative refinement approach of Lee et al. (2018), in which the model corrects the original non-autoregressive prediction by passing it multiple times through a denoising autoencoder. A major difference is that Lee et al. (2018) train their noisy autoencoder to deal with corrupt inputs by applying stochastic corruption heuristics on the training data, while we simply mask a random number of input tokens. We also show that

our approach outperforms all of these models by wide margins.

**Arbitrary Order Language Generation** Finally, recent work has developed insertion-based transformers for arbitrary, but fixed, word order generation (Gu et al., 2019; Stern et al., 2019). While they do not decode in a constant number of iterations, Stern et al. (2019) show strong results in logarithmic time. Both models treat each token insertion as a separate training example, which cannot be computed in parallel with every other insertion in the same sequence. This makes training significantly more expensive that standard transformers (which use causal attention masking) and our CMLMs (which can predict all of the masked tokens in parallel).

## 7 Conclusion

This work introduces conditional masked language models and a novel mask-predict decoding algorithm that leverages their parallelism to generate text in a constant number of decoding iterations. We show that, in the context of machine translation, our approach substantially outperforms previous parallel decoding methods, and can approach the performance of sequential autoregressive models while decoding much faster. While there are still open problems, such as the need to condition on the target's length and the dependence on knowledge distillation, our results provide a significant step forward in non-autoregressive and parallel decoding approaches to machine translation. In a broader sense, this paper shows that masked language models are useful not only for representing text, but also for *generating* text efficiently.

## Acknowledgements

We thank Abdelrahman Mohamed for sharing his expertise on non-autoregressive models, and our colleagues at FAIR for valuable feedback.

## References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional Sequence to Sequence Learning. *ArXiv e-prints*.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376. ACM.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *ICLR*.

Jiatao Gu, Qi Liu, and Kyunghyun Cho. 2019. Insertion-based decoding with automatically inferred generation order. *arXiv preprint arXiv:1902.01370*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference for Learning Representations*.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1173–1182, Brussels, Belgium. Association for Computational Linguistics.

Jindřich Libovický and Jindřich Helcl. 2018. End-to-end non-autoregressive neural machine translation with connectionist temporal classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3016–3021, Brussels, Belgium. Association for Computational Linguistics.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. 2018. Mixed precision training. In *International Conference on Learning Representations*.

Graham Neubig, Zi-Yi Dou, Junjie Hu, Paul Michel, Danish Pruthi, and Xinyi Wang. 2019. compare-mt: A tool for holistic comparison of language generation systems. In *Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL) Demo Track*, Minneapolis, USA.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. Mass: Masked sequence to sequence pre-training for language generation. *arXiv preprint arXiv:1905.02450*.

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. 2019. Insertion transformer: Flexible sequence generation via insertion operations. *arXiv preprint arXiv:1902.03249*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Alex Wang and Kyunghyun Cho. 2019. Bert has a mouth, and it must speak: Bert as a markov random field language model. *arXiv preprint arXiv:1902.04094*.

Felix Wu, Angela Fan, Alexei Baevski, Yann N Dauphin, and Michael Auli. 2019. Pay less attention with lightweight and dynamic convolutions. *International Conference on Learning Representations*.

# Learning to Copy for Automatic Post-Editing

**Xuancheng Huang**[†], **Yang Liu**[†‡*], **Huanbo Luan**[†], **Jingfang Xu**[§] **and Maosong Sun**[†]
[†]Institute for Artificial Intelligence
State Key Laboratory of Intelligent Technology and Systems
Department of Computer Science and Technology, Tsinghua University, Beijing, China
Beijing National Research Center for Information Science and Technology
[§]Sogou Inc., Beijing, China
[‡]Beijing Advanced Innovation Center for Language Resources
hxc17@mails.tsinghua.edu.cn, liuyang2011@tsinghua.edu.cn, luanhuanbo@gmail.com,
xujingfang@sogou-inc.com, sms@tsinghua.edu.cn

## Abstract

Automatic post-editing (APE), which aims to correct errors in the output of machine translation systems in a post-processing step, is an important task in natural language processing. While recent work has achieved considerable performance gains by using neural networks, how to model the copying mechanism for APE remains a challenge. In this work, we propose a new method for modeling copying for APE. To better identify translation errors, our method learns the representations of source sentences and system outputs in an interactive way. These representations are used to explicitly indicate which words in the system outputs should be copied, which is useful to help CopyNet (Gu et al., 2016) better generate post-edited translations. Experiments on the datasets of the WMT 2016-2017 APE shared tasks show that our approach outperforms all best published results. [1]

## 1 Introduction

Automatic post-editing (APE) is an important natural language processing (NLP) task that aims to automatically correct errors made by machine translation systems (Knight and Chander, 1994). It can be considered as an efficient way to modify translations to a specific domain or to incorporate additional information into translations rather than translating from scratch (McKeown et al., 2012; Chatterjee et al., 2015, 2018).

Approaches to APE can be roughly divided into two broad categories: *statistical* and *neural* approaches. While early efforts focused on statistical approaches relying on manual feature engineering (Simard et al., 2007; Béchara et al., 2011), neural network based approaches capable of learning representations from data have shown remarkable superiority over their statistical counterparts (Varis and Bojar, 2017; Chatterjee et al., 2017; Junczys-Dowmunt and Grundkiewicz, 2017; Unanue et al., 2018). Most of them cast APE as a multi-source sequence-to-sequence learning problem (Zoph and Knight, 2016): given a source sentence ($src$) and a machine translation ($mt$), APE outputs a post-edited translation ($pe$).

| | |
|---|---|
| $src$ | I ate a cake yesterday |
| $mt$ | **Ich** esse **einen** Hamburger |
| $pe$ | **Ich** hatte gestern **einen** Kuchen gegessen |

Table 1: Example of automatic post-editing (APE). Given a source sentence ($src$) and a machine translation ($mt$), the goal of APE is to post-edit the erroneous translation to obtain a correct translation ($pe$). Our work aims to explicitly model how to copy words from $mt$ to $pe$ (highlighted in bold), which is a common phenomenon in APE.

A common phenomenon in APE is that many words in $mt$ can be **copied** to $pe$. As shown in Table 1, two German words "Ich" and "einen" occur in both $mt$ and $pe$. Note that the positions of copied words in $mt$ and $pe$ are not necessarily identical (e.g., "einen" in Table 1). Our analysis on the datasets of the WMT 2016 and 2017 APE shared tasks shows that over 80% of words in $mt$ are copied to $pe$. As APE models not only need to decide which words in $mt$ should be copied correctly, but also should place the copied words in appropriate positions in $pe$, it is challenging to model copying for APE. Our experiments show that the state-of-the-art APE method (Junczys-Dowmunt and Grundkiewicz, 2018) only achieves a copying accuracy of 64.63% (see Table 7).

We believe that existing approaches to APE (Varis and Bojar, 2017; Chatterjee et al., 2017; Junczys-Dowmunt and Grundkiewicz, 2017; Unanue et al., 2018) suffer from two major

*Corresponding author: Yang Liu
[1]The source code is available at https://github.com/THUNLP-MT/L2Copy4APE



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6122–6132,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

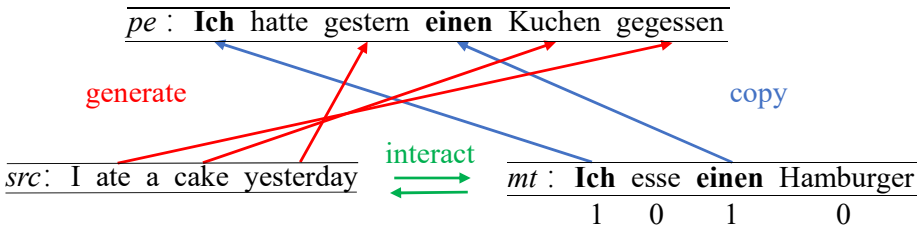


Figure 1: Learning to copy for APE. Our work is based on two key ideas. As both $src$ and $mt$ play important roles in APE, the first idea is that $src$ and $mt$ should "interact" with each other during representation learning to better generate words from $src$ and copy words from $mt$ during inference. The second idea is to predict which target words in $mt$ should be copied since it is easy to obtain labeled data automatically by comparing $mt$ and $pe$. The words to be copied (e.g., "Ich") in $mt$ are labeled with 1's and other words (e.g., "esse") with 0's.

drawbacks when modeling the copying mechanism. First, the representations of $src$ and $mt$ are learned separately. APE is a two-source sequence-to-sequence learning problem in which both $src$ and $mt$ play important roles. On the one hand, if $src$ is ignored, it is difficult to identify translation errors related to adequacy in $mt$, especially for fluent but inadequate translations (e.g., $mt$ in Figure 1). On the other hand, $mt$ serves as a major source for generating $pe$ since many words (e.g., "Ich" and "einen" in Figure 1) are copied from $mt$ to $pe$. Intuitively, it is likely to be easier to decide which words in $mt$ should be copied if $src$ and $mt$ fully "interact" with each other during representation learning. Although CopyNet (Gu et al., 2016) can be adapted for explicitly modeling the copying mechanism in multi-source sequence-to-sequence learning, the lack of the interaction between $src$ and $mt$ still remains a problem.

Second, there is no explicit labeling that indicates which target words in $mt$ should be copied. Existing approaches only rely on the attention between the encoder and decoder to implicitly choose target words to be copied. Given $mt$ and $pe$, it is easy to decide whether a target word in $mt$ should be copied or not. In Figure 1, the words in $mt$ that should be copied are labeled with 1's. Other words are labeled with with 0's, which should be re-generated from $src$. These labels can served as useful supervision signals to help better copy words from $mt$ to $pe$, even when CopyNet is used.

In this work, we propose a new method for modeling the copying mechanism for APE. As shown in Figure 1, our work is based on two key ideas. First, our method is capable of learning the representations of input in an interactive way by enabling $src$ and $mt$ to attend to each other during representation learning. This might be useful for deciding when to generate words from $src$ and when to copy words from $mt$ during post-editing. Second, it is possible to predict which words in $mt$ should be copied because it is easy to automatically construct labeled data by comparing $mt$ and $pe$. Such predictions can be combined with CopyNet to better model copying for APE. Experiments show that our approach outperforms the best published results on the datasets of the WMT 2016-2017 APE shared tasks.

## 2 Background

### 2.1 Multi-source Sequence-to-Sequence Learning

Multi-source sequence-to-sequence learning has been widely used in APE in recent years (Junczys-Dowmunt and Grundkiewicz, 2018; Pal et al., 2018; Tebbifakhr et al., 2018; Shin and Lee, 2018). The architecture of multi-source Transformer is shown in Figure 2(a). It can be equipped with CopyNet (see Section 2.2) to serve as a baseline in our experiments. It is worth noting that $src$ and $mt$ are encoded separately.

Let $\mathbf{x} = x_1 \dots x_I$ be a source sentence (i.e., $src$) with $I$ words, $\tilde{\mathbf{y}} = \tilde{y}_1 \dots \tilde{y}_K$ be a translation output by a machine translation system (i.e., $mt$) with $K$ words, and $\mathbf{y} = y_1 \dots y_J$ be the post-edited translation (i.e., $pe$) with $J$ words. The APE model is given by

$$P(\mathbf{y}|\mathbf{x}, \tilde{\mathbf{y}}; \boldsymbol{\theta}) = \prod_{j=1}^{J} P(y_j|\mathbf{x}, \tilde{\mathbf{y}}, \mathbf{y}_{<j}; \boldsymbol{\theta}), \quad (1)$$

where $y_j$ is the $j$-th target word in $pe$, $\mathbf{y}_{<j} = y_1 \dots y_{j-1}$ is a partial translation, $\boldsymbol{\theta}$ is a set of model parameters, and $P(y_j|\mathbf{x}, \tilde{\mathbf{y}}, \mathbf{y}_{<j}; \boldsymbol{\theta})$ is a word-level translation probability.

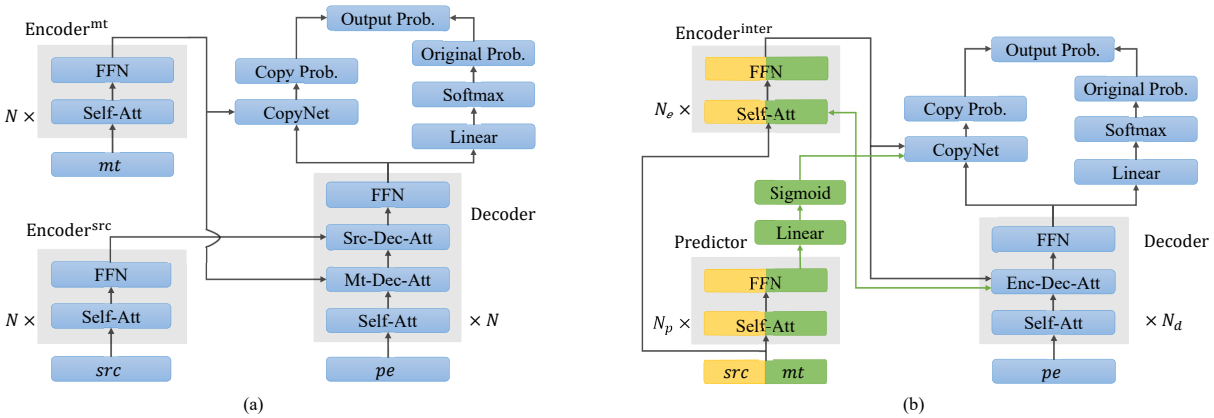


Figure 2: (a) The architecture of multi-source Transformer (Junczys-Dowmunt and Grundkiewicz, 2018) equipped with CopyNet (Gu et al., 2016) and (b) the architecture of our approach. While the existing work learns the representations of $src$ and $mt$ separately, our approach allows for learning the representations of $src$ and $mt$ in an interactive way by concatenating them as a single input. In addition, our approach introduces a Predictor module to explicitly indicate which words in $mt$ should be copied.

The word-level translation probability in Eq. (1) is computed as

$$\mathbf{H}^{\text{src}} = \text{Encoder}^{\text{src}}(\mathbf{x}, \boldsymbol{\theta}), \quad (2)$$

$$\mathbf{H}^{\text{mt}} = \text{Encoder}^{\text{mt}}(\tilde{\mathbf{y}}, \boldsymbol{\theta}), \quad (3)$$

$$\mathbf{h}_j^{\text{pe}} = \text{Decoder}(\mathbf{y}_{<j}, \mathbf{H}^{\text{src}}, \mathbf{H}^{\text{mt}}, \boldsymbol{\theta}), \quad (4)$$

$$P(y_j|\mathbf{x}, \tilde{\mathbf{y}}, \mathbf{y}_{<j}; \boldsymbol{\theta}) \propto \exp(\mathbf{h}_j^{\text{pe}} \mathbf{W}_g), \quad (5)$$

where $\text{Encoder}^{\text{src}}(\cdot)$ is the encoder for $src$, $\mathbf{H}^{\text{src}}$ is the real-valued representation of $src$, $\text{Encoder}^{\text{mt}}(\cdot)$ is the encoder for $mt$, $\mathbf{H}^{\text{mt}}$ is the representation of $mt$, $\text{Decoder}(\cdot)$ is the decoder, $\mathbf{h}_j^{\text{pe}}$ is the representation of the $j$-th target word $y_j$ in $pe$. $\mathbf{W}_g \in \mathbb{R}^{d \times \mathcal{V}_y}$ is a weight matrix, $d$ is the dimension of hidden states, and $\mathcal{V}_y$ is the target vocabulary size.

A limitation of the aforementioned model is that $src$ and $mt$ are encoded separately without interacting with each other, which might lead to the inability to find which $src$ word is untranslated and which $mt$ word is incorrect. For example, the $mt$ sentence in Figure 1 is fluent and meaningful. Without $src$, the APE system is unable to identify translation errors. In addition, the multi-source Transformer does not explicitly model the copying between $mt$ and $pe$ in neither the Encoder nor the Decoder.

## 2.2 CopyNet

CopyNet (Gu et al., 2016) is a widely used method for modeling copying in sequence-to-sequence learning. It has been successfully applied to single-turn dialogue (Gu et al., 2016), text summarization (See et al., 2017), and grammar error correction (Zhao et al., 2019).

It is possible to extend the multi-source Transformer with CopyNet to explicitly model the copying mechanism, as shown in Figure 2(a). CopyNet defines the word-level translation probability in Eq. (1) as a linear interpolation of copying and generating probabilities:

$$P(y_j|\mathbf{x}, \tilde{\mathbf{y}}, \mathbf{y}_{<j}; \boldsymbol{\theta}) = \gamma_j \times P^{\text{copy}}(y_j) + (1 - \gamma_j) \times P^{\text{gen}}(y_j), \quad (6)$$

where $P^{\text{copy}}(y_j)$ is the copying probability for $y_j$, $P^{\text{gen}}(y_j)$ is the generating probability for $y_j$, and $\gamma_j$ is a gating weight. They are defined as follows:

$$P^{\text{copy}}(y_j) \propto \exp(g(\mathbf{H}^{\text{mt}}, \mathbf{h}_j^{\text{pe}})), \quad (7)$$

$$P^{\text{gen}}(y_j) \propto \exp(\mathbf{h}_j^{\text{pe}} \mathbf{W}_g), \quad (8)$$

$$\gamma_j = u(\mathbf{H}^{\text{mt}}, \mathbf{h}_j^{\text{pe}}), \quad (9)$$

where $g(\cdot)$ and $u(\cdot)$ are non-linear functions. See (Zhao et al., 2019) for more details.

Copying in APE involves two kinds of decisions: (1) choosing words in $mt$ to be copied and (2) placing the copied words in appropriate positions in $pe$. CopyNet makes the two kinds of decisions simultaneously. We conjecture that if which words in $mt$ should be copied can be explicitly indicated, it might be easier for CopyNet to copy words from $mt$ to $pe$ correctly. Therefore, it is necessary to design a new method for identifying words to be copied.

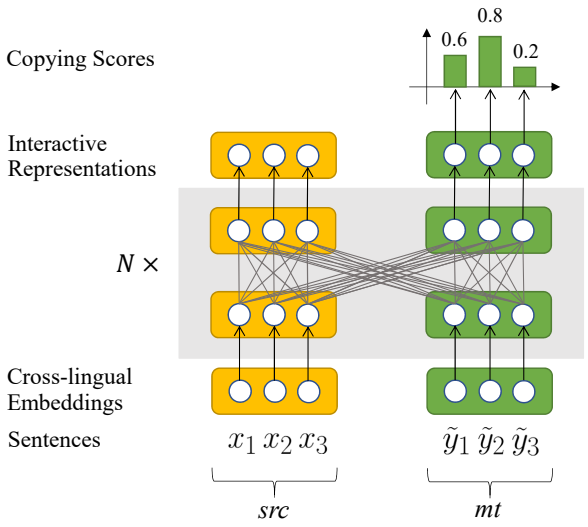


Figure 3: Interactive representation learning. Concatenated to serve as a single input, $src$ and $mt$ attend to each other during representation learning. Note that learnable weights of $src$ and $mt$ are shared. Interactive representation learning is used in both Predictor and Encoder. For example, copying scores are predicted based on the learned representations (see Eq. (13)).

## 3 Approach

Figure 2(b) shows the overall architecture of our approach. It differs from previous work in two aspects. First, we propose to let $src$ and $mt$ "interact" with each other to learn better representations (Section 3.1). Second, our approach introduces a Predictor module to predict words to be copied (Section 3.2). Section 3.3 describes how to train our APE model.

### 3.1 Interactive Representation Learning

We propose an interactive representation learning method by making $src$ and $mt$ attend to each other. Following Lample and Conneau (2019) and He et al. (2018), we concatenate $src$ and $mt$ in the dimension of sentence length with additional position and language embeddings:

$$\mathbf{X}_i = \mathbf{E}^{\text{token}}[x_i] + \mathbf{E}^{\text{pos}}[i] + \mathbf{E}^{\text{lang}}[0], \quad (10)$$

$$\tilde{\mathbf{Y}}_k = \mathbf{E}^{\text{token}}[\tilde{y}_k] + \mathbf{E}^{\text{pos}}[k] + \mathbf{E}^{\text{lang}}[1], \quad (11)$$

where $\mathbf{X}_i$ is the embedding of the $i$-th source word $x_i$, $\tilde{\mathbf{Y}}_k$ is the embedding of the $k$-th target word $\tilde{y}_k$, and $\mathbf{E}^{\text{token}}$, $\mathbf{E}^{\text{pos}}$, and $\mathbf{E}^{\text{lang}}$ are the token, position and language embedding matrices.

As shown in Figure 3, the representation of $src$ and $mt$ can be learned jointly:

$$\mathbf{H}^{\text{inter}} = \text{Encoder}^{\text{inter}}([\mathbf{X}; \tilde{\mathbf{Y}}], \boldsymbol{\theta}), \quad (12)$$

where $\text{Encoder}^{\text{inter}}(\cdot)$ is the interactive Encoder and $[\mathbf{X}; \tilde{\mathbf{Y}}]$ is the concatenation of $\mathbf{X}$ and $\tilde{\mathbf{Y}}$ in the dimension of sentence length.

As shown in Figure 2(b), the multi-source Encoders are replaced by the interactive Encoder, which enables $src$ and $mt$ to attend to each other.[2] We expect that enabling the interactions between them can help to strengthen the ability of the model to find which words in $src$ is untranslated and which words in $mt$ is correct. Note that interactive representation learning is used both in Predictor and Encoder. In the following, we will describe how to predict which words in $mt$ should be copied based on these learned representations.

### 3.2 Predicting Words to be Copied

Given $mt$ and $pe$, we can label each word in $mt$ as 0 or 1. We use 1 to denote that the word is to be copied (e.g. "Ich" and "einen" in Figure 1) and 0 not to be copied (e.g. "esse" and "Hamburger" in Figure 1). It is possible to use the Longest Common Sequence (LCS) (Wagner and Fischer, 1974) algorithm to obtain common sequences between $mt$ and $pe$. If the word in $mt$ also appears in the common sequences, it will be labeled 1; otherwise, it will be labeled 0. We denote these labels as $l_1 \dots l_K$.

We propose a Predictor module to predict words to be copied. As discrete labels are non-differentiable during training, the Predictor module outputs **copying scores** instead for the target words in $mt$:

$$\mathbf{s} = \text{sigmoid}([\mathbf{H}^{\text{pred}}_{I+1}; \cdots; \mathbf{H}^{\text{pred}}_{I+K}]\mathbf{W}_s), \quad (13)$$

where $\mathbf{s} \in \mathbb{R}^{K\times 1}$ is a vector of copying scores corresponding to the $K$ words in $mt$, $\mathbf{H}^{\text{pred}} \in \mathbb{R}^{(I+K)\times d}$ is the representation of $src$ and $mt$:

$$\mathbf{H}^{\text{pred}} = \text{Predictor}([\mathbf{X}; \tilde{\mathbf{Y}}]; \boldsymbol{\theta}), \quad (14)$$

and $\mathbf{W}_s \in \mathbb{R}^{d\times 1}$ is a weight matrix. Only the representation of $mt$ (i.e., $[\mathbf{H}^{\text{pred}}_{I+1}; \cdots; \mathbf{H}^{\text{pred}}_{I+K}]$) is used for calculating copying scores. [3]

As shown in Figure 2(b), copying scores can be incorporated into three parts of our model: the

[2] Note that the Predictor and Encoder do not share their weights because we found that sharing leads to degraded performance in experiments.

[3] It is also possible to predict which source words in $src$ should be used to generate non-copied words in $pe$ (e.g., "Kuchen" in Figure 1). This can be done by generating explicit labels using bilingual word alignment. We leave this for future work.

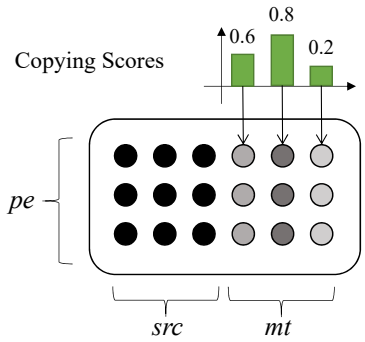


Figure 4: Copying scores as scaling masks. The copying scores are used to modify the attention layers of the Encoder, Decoder, and CopyNet.

Encoder, the Decoder, and the CopyNet. Inspired by Yang et al. (2018)'s strategy to integrate localness to self-attention, we propose to incorporate copying scores into our model by modifying attention weights involved in the aforementioned three modules.

The original scaled dot-product attention (Vaswani et al., 2017) is defined as

$$\text{energy} = \frac{\mathbf{q}\mathbf{K}^\top}{\sqrt{d}}, \tag{15}$$

$$\text{Att}(\mathbf{q}, \mathbf{K}) = \text{softmax}(\text{energy}), \tag{16}$$

where $\mathbf{q} \in \mathbb{R}^{1\times d}$ is the query vector, $\mathbf{K} \in \mathbb{R}^{(I+K)\times d}$ is the key matrix, and $\text{energy} \in \mathbb{R}^{1\times(I+K)}$ is the "energy" vector.

As shown in Figure 4, the copying scores can be used to form a scaling mask on the attention sub-layer:[4]

$$\text{Att}(\mathbf{q}, \mathbf{K}) = \text{softmax}\left(\text{energy} \odot [\mathbf{m}; \mathbf{s}]^\top\right), \tag{17}$$

where $\mathbf{m} = \{1.0\}^I$ is a masking vector corresponding to $src$ and $\mathbf{s} \in \mathbb{R}^{K\times 1}$ is the vector of copying scores calculated by Eq. (13) corresponding to $mt$. Note that copying scores are used to only change the attention weights related to $mt$ while the $src$ part is unchanged.

### 3.3 Training

The training objective of our approach $L_{\text{all}}(\boldsymbol{\theta})$ consists of three parts:

$$L_{\text{all}}(\boldsymbol{\theta}) = (1-\alpha) \times (L_{\text{ape}}(\boldsymbol{\theta}) + \lambda \times L_{\text{copy}}(\boldsymbol{\theta})) + \alpha \times L_{\text{pred}}(\boldsymbol{\theta}), \tag{18}$$

where $\alpha$ and $\lambda$ are hyper-parameters.

The first part is the original log-likelihood loss function of APE: [5]

$$L_{\text{ape}}(\boldsymbol{\theta}) = -\log P(\mathbf{y}|\mathbf{x}, \tilde{\mathbf{y}}; \boldsymbol{\theta}). \tag{19}$$

The second part is related to the CopyNet:

$$L_{\text{copy}}(\boldsymbol{\theta}) = \frac{1}{K}\sum_{k=1}^{K}(l_k - c_k)^2, \tag{20}$$

where $l_k$ is the ground-truth label (see Section 3.2) and $c_k$ is a quantity that measures how likely the $k$-th word in $mt$ to be copied by CopyNet:

$$c_k = \sum_{j=1}^{J} \gamma_j \times P^{\text{copy}}(\tilde{y}_k). \tag{21}$$

Note that $\gamma \times P^{\text{copy}}(y)$ is the term related to copying the target word $y$ in Eq. (6).

The third part is a cross-entropy loss related to the Predictor:

$$L_{\text{pred}}(\boldsymbol{\theta}) = -\sum_{k=1}^{K}\Big[l_k \log(s_k) + (1-l_k)\log(1-s_k)\Big], \tag{22}$$

where $s_k$ is the copying score of the $k$-th word $\tilde{y}_k$ in $mt$.

Finally, we use an optimizer to find the model parameters that minimize the overall loss function:

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\text{argmin}}\left\{L_{\text{all}}(\boldsymbol{\theta})\right\}. \tag{23}$$

## 4 Experiments

### 4.1 Setup

**Datasets**

We evaluated our approach on the WMT APE datasets, which often distinguish between two tasks: phrase-based statistical machine translation (i.e., PBSMT) and neural machine translation (i.e., NMT). All these APE datasets consist of English-German triplets containing source text ($src$), the translations ($mt$) from a "black-box" MT system and the corresponding human-post-edits ($pe$). The statistics for the WMT APE datasets are shown in Table 2. In addition to the official dataset, the organizers also recommend using additional datasets

[4] Actually, we let "energy" vector minus its minimum value to keep it non-negative.

[5] For simplicity, we use sentence-level loss functions here. In practice, we use corpus-level loss functions.

| Category | Dataset | # Sent. |
|---|---|---|
| PBSMT | training set | 23,000 |
| | dev2016 | 1,000 |
| | test2016 | 2,000 |
| | test2017 | 2,000 |
| NMT | training set | 13,442 |
| | dev2018 | 1,000 |
| Additional | artificial-small | 526,368 |
| | artificial-big | 4,391,180 |
| | eSCAPE-PBSMT | 7,258,533 |
| | eSCAPE-NMT | 7,258,533 |

Table 2: Statistics of the English-German datasets in the WMT APE task. Note that the NMT official data only contains training and development sets.

(Junczys-Dowmunt and Grundkiewicz, 2016; Negri et al., 2014).

We used the WMT official dataset for the PBSMT task and the NMT task separately. The artificial training data (Junczys-Dowmunt and Grundkiewicz, 2016) was also used for both tasks. More precisely, we used the concatenation of the official training data and the artificial-small data to learn a truecasing model (Koehn et al., 2007) and obtain sub-word units using byte-pair encoding (BPE) (Sennrich et al., 2015) with 32k merges. Then, we applied truecasing and BPE to all datasets. We oversampled the official training data 20 times and concatenated them with both artificial-small and artificial-big datasets (Junczys-Dowmunt and Grundkiewicz, 2018). Finally, we obtained a dataset containing nearly 5M triplets for both tasks. To test our approach on a larger PBSMT dataset, we used the eSCAPE synthetic dataset (Negri et al., 2014), which contains 7.2M sentences. By including the eSCAPE dataset, the training set is enlarged to nearly 12M sentences.

### Hyper-Parameter Settings

For the original Transformer model, CopyNet and our approach, the hidden size was set to 512 and the filter size was set to 2,048. The number of individual attention heads was set to 8 for multi-head attention. We set $N = N_e = N_d = 6$, $N_p = 3$ and we tied all three $src, mt, pe$ embeddings for saving memory. The embeddings and softmax weights were also tied. In training, we used Adam (Kingma and Ba, 2014) for optimization. Each mini-batch contains approximately 25K tokens. We used the learning rate decay policy described by (Vaswani et al., 2017). In decoding, the beam size was set to 4. We used the length penalty (Wu et al., 2016) and set the hyper-parameter to 1.0. The other hyper-parameter settings were the same as the Transformer model (Vaswani et al., 2017). We implemented our approach on top of the open-source toolkit THUMT (Zhang et al., 2017).[6]

| $\alpha$ | $\lambda$ | TER↓ | BLEU↑ |
|---|---|---|---|
| 0.1 | 1.0 | 18.83 | 72.59 |
| 0.5 | 1.0 | 18.45 | 72.83 |
| 0.9 | 0.1 | 18.89 | 72.27 |
| 0.9 | 0.5 | 18.47 | 72.83 |
| 0.9 | 1.0 | **18.38** | **72.99** |

Table 3: Effect of $\alpha$ and $\lambda$. The TER and BLEU scores are calculated on the WMT 2016 APE official development set.

### Evaluation Metrics

We used the same evaluation metrics as the official WMT APE task (Chatterjee et al., 2018): case-sensitive BLEU and TER. BLEU is computed by *multi-bleu.perl* (Koehn et al., 2007). TER is calculated using TERcom.[7]

### Baselines

We compared our approach with the following seven baselines:

1. ORIGINAL: the original $mt$ without any post-editing.
2. COPYNET (Gu et al., 2016; Zhao et al., 2019): the multi-source Transformer equipped with CopyNet (see Figure 2(a)).
3. NPI-APE (Vu and Haffari, 2018): a neural programmer-interpreter approach.
4. MS_UEDIN (Junczys-Dowmunt and Grundkiewicz, 2018): a multi-source Transformer-based APE system that shares the encoders of $src$ and $mt$. It is the champion of the WMT 2018 APE shared task.
5. USAAR_DFKI (Pal et al., 2018): a multi-source Transformer-based APE system with a joint encoder that attends over a combination of two encoded sequences. It is a participant of the WMT 2018 APE shared task.

[6] https://github.com/THUNLP-MT/THUMT
[7] http://www.cs.umd.edu/snover/tercom/

| System | TEST16 | | TEST17 | | TEST16+17 | |
|---|---|---|---|---|---|---|
| | TER↓ | BLEU↑ | TER↓ | BLEU↑ | TER↓ | BLEU↑ |
| (1) ORIGINAL | 24.76 | 62.11 | 24.48 | 62.49 | 24.62 | 62.30 |
| WMT 2017 official + artificial (5M) | | | | | | |
| (2) NPI-APE | 22.07 | 66.67 | 22.58 | 65.52 | – | – |
| (3) POSTECH | 19.14 | 70.98 | 19.26 | 70.50 | – | – |
| (4) FBK | 18.79 | 71.48 | 19.54 | 70.09 | – | – |
| (5) MS_UEDIN$_{\text{ensemble}}$ | 18.86 | 71.04 | 19.03 | 70.46 | – | – |
| (6) COPYNET | 18.91 | 71.64 | 19.47 | 70.75 | 19.19 | 71.19 |
| (7) Ours | 18.39 | 72.50 | 18.81 | 71.35 | 18.60 | 71.92 |
| (8) Ours$_{\text{ensemble}}$ | **17.77** | **73.19** | **18.41** | **72.09** | **18.09** | **72.62** |
| WMT 2017 official + artificial + eSCAPE (12M) | | | | | | |
| (9) USAAR_DFKI | – | 68.52 | – | 68.91 | – | – |
| (10) MS_UEDIN$_{\text{ensemble}}$ | 17.34 | 73.43 | 17.47 | 72.84 | – | – |
| (11) COPYNET | 18.06 | 72.77 | 18.29 | 71.89 | 18.18 | 72.37 |
| (12) Ours | 17.45 | 73.51 | 17.77 | 72.98 | 17.61 | 73.24 |
| (13) Ours$_{\text{ensemble}}$ | **17.06** | **74.00** | **17.37** | **73.26** | **17.22** | **73.62** |

Table 4: Results on the English-German PBSMT sub-task. "TEST16+17" is the concatenation of "TEST16" and "TEST17". MS_UEDIN$_{\text{ensemble}}$ and Ours$_{\text{ensemble}}$ used ensembles of four models.

6. POSTECH (Shin and Lee, 2018): a multi-source Transformer-based APE system with two encoders. It is a participant of the WMT 2018 APE shared task.

7. FBK (Tebbifakhr et al., 2018): a multi-source Transformer-based APE system with two encoders. It is a participant of the WMT 2018 APE shared task.

We implemented COPYNET also on top of THUMT (Zhang et al., 2017). The results of all other baselines were taken from the corresponding original papers.

### 4.2 Effect of Hyper-parameters

We first investigated the effect of the hyper-parameters $\alpha$ and $\lambda$ in Eq. (18). As shown in Table 3, using $\alpha = 0.9$ and $\lambda = 1.0$ achieves the best performance in terms of TER and BLEU on the WMT 2016 development set, suggesting that both the Predictor and CopyNet play important roles in our approach. Therefore, we set $\alpha = 0.9$ and $\lambda = 1.0$ in the following experiments.

### 4.3 Main Results

#### Results on the PBSMT Sub-task

Table 4 shows the results of the PBSMT sub-task. We used the development set of the WMT 2016 APE PBSMT sub-task for model selection for our approach.

Under the small-data training condition (i.e., 5M), our single model (i.e., System 7) outperforms all single-model baselines on all test sets. The superiority over COPYNET (i.e., System 6) suggests that interactive representation learning and incorporating copying scores are effective in improving APE. Our approach that uses the ensemble of four models (i.e., System 8) also improves over the best published result (i.e., System 5).

Under the large-data training condition (i.e., 12M), we find that our single (i.e., System 12) and ensemble (i.e., System 13) models still outperform the best single (i.e., System 11) and ensemble (i.e., System 10) models of baselines.

| System | TER↓ | BLEU↑ |
|---|---|---|
| ORIGINAL | 15.08 | 76.76 |
| POSTECH | 14.94 | 77.26 |
| COPYNET | 15.12 | 77.05 |
| Ours | **14.88** | **77.40** |

Table 5: Experiments on the WMT 2018 English-German NMT sub-task. The TER and BLEU scores are calculated on the WMT 2018 APE NMT sub-task development set.

| ID | Module | | | | DEV16 | | TEST16+17 | |
|---|---|---|---|---|---|---|---|---|
| | Interactive | Predictor | CopyNet | Joint Training | TER↓ | BLEU↑ | TER↓ | BLEU↑ |
| 1 | √ | × | × | × | 18.74 | 72.21 | 19.11 | 71.03 |
| 2 | × | × | √ | × | 19.33 | 71.86 | 19.19 | 71.19 |
| 3 | × | √ | √ | √ | 19.11 | 72.03 | 19.07 | 71.30 |
| 4 | √ | × | √ | × | 18.62 | 72.42 | 18.91 | 71.53 |
| 5 | √ | √ | × | × | 18.53 | 72.53 | 18.85 | 71.54 |
| 6 | √ | √ | √ | × | 18.44 | 72.96 | 18.77 | 71.75 |
| 7 | √ | √ | √ | √ | **18.38** | **72.99** | **18.60** | **71.92** |

Table 6: Ablation study. "Interactive" denotes interactive representation learning (Section 3.1), "Predictor" denotes detecting correct words (Section 3.2), "CopyNet" denotes the CopyNet used in our approach (Section 2.2), and "Joint Training" denotes the combination of three loss functions (Section 3.3).

**Results on the NMT Sub-task**

Table 5 shows the results on the NMT sub-task. Besides ORIGINAL and COPYNET, we also compared our approach with POSTECH (Shin and Lee, 2018), which is the only participating system that released the results on the development set of the WMT 2018 NMT sub-task.[8] We find that our approach also outperforms all baselines.

### 4.4 Ablation Study

Table 6 shows the results of ablation study. It is clear that interactive representation learning plays a critical role since removing it impairs post-editing performance (line 3). As shown in line 4, the Predictor is also an essential part of our approach. CopyNet and joint training are also shown to be beneficial for improving APE (lines 5 and 6) but seem to have relatively smaller contributions than interactive representation learning and predicting words to be copied.

### 4.5 Results on Prediction Accuracy

The Predictor is an important module in our approach as it predicts which words in $mt$ should be copied. Given the ground-truth labels, it is easy to calculate *prediction accuracy* by casting the prediction as a binary classification problem. We find that the Predictor achieves a prediction accuracy of 85.09% on the development set.

### 4.6 Comparison of Copying Accuracies

A target word $y$ in a machine translation $\tilde{\mathbf{y}}$ is called to be *correctly* copied to an automatic edited translation $\hat{\mathbf{y}}$ if and only if the positions where $y$ occurs in $\hat{\mathbf{y}}$ and $\mathbf{y}$ (i.e., the ground-truth edited translation) are identical. Therefore, it is easy to define copying accuracy to measure how well the copying mechanism works.

| System | Accuracy |
|---|---|
| MS_UEDIN | 64.63 |
| COPYNET | 64.72 |
| Ours | **65.61** |

Table 7: Comparison of copying accuracies.

Table 7 shows the comparison of copying accuracies between MS_UEDIN, COPYNET, and our approach. We find that our approach outperforms the two baselines. However, the copying accuracy of our approach is almost 20% lower than the prediction accuracy (i.e., 65.61% vs. 85.09%), indicating that it is much more challenging to place the copied words in correct positions.

### 4.7 Visualization

Figure 5 gives an example that illustrates how copying scores influence attention. It shows the heatmap of the Enc-Dec-Attention, which averages over 8 different heads. Only the attention weights beween $pe$ and $mt$ are included. We take the second layer of Enc-Dec-Attention for example. The x-axis represents $mt$ and the y-axis represents $pe$. The darker the color, the higher the copying scores.

We find that words "mit" and "dem" are identified by the Predictor. Accordingly, the attention weights corresponding to these words are decreased since the columns corresponding to these words have lighter color. As a result, all words in $mt$ other than "mit" and "dem" are copied to $pe$.

[8] As POSTECH only used small data for training, the eSCAPE datasets were not used in this experiment for a fair comparison.

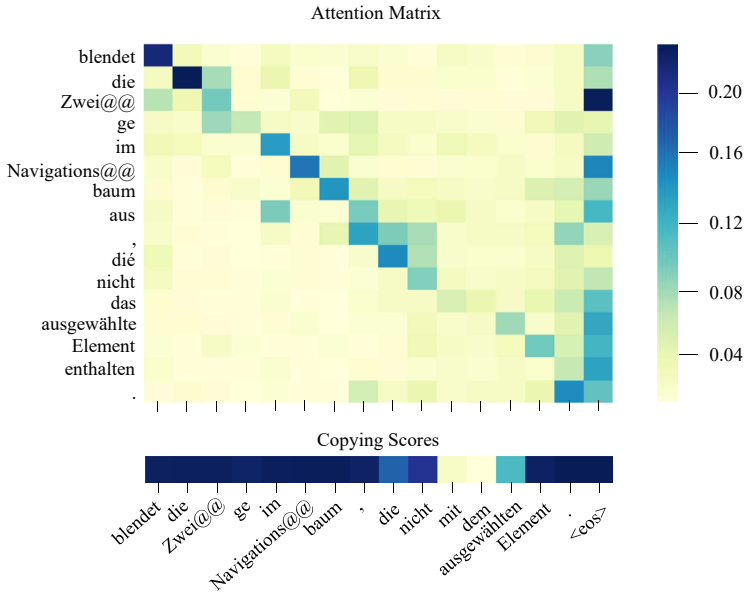


Figure 5: Example of the heatmap of attention and copying scores. The x-axis is $mt$ and the y-axis is $pe$. The Predictor successfully detects the incorrect word "mit" and "dem" and gives low copying scores to these words and then decrease the importance of them in attention. Other words in $mt$ are correctly copied to $pe$.

## 5 Related Work

### 5.1 Multi-source Sequence-to-Sequence Learning

Recently, multi-source Transformer-based APE systems (Junczys-Dowmunt and Grundkiewicz, 2018; Pal et al., 2018; Tebbifakhr et al., 2018; Shin and Lee, 2018) have achieved the state-of-the-art results on the datasets of the WMT APE shared task. Our work differs from prior studies by enabling interactions between $src$ and $mt$ and explicitly detecting words to be copied.

### 5.2 The Copying Mechanism

Zhao et al. (2019) apply CopyNet (Gu et al., 2016) to grammar error correction. Their approach generates labels similar to ours, but only uses them to perform mutli-task learning. Libovický et al. (2016) first introduce CopyNet to APE but do not provide a detailed description of their method and experimental results. We show that interactive representation learning and explicit indication of words are important for modeling copying in APE.

### 5.3 Interactive Representation Learning

Niehues et al. (2016) simply concatenate the output of the PBSMT system and the source sentence to serve as the input the NMT system without enabling multi-layer interactive learning. Lample and Conneau (2019) used cross-lingual setting to enable cross-lingual language model pre-training. We propose to let $src$ and $mt$ fully interact with each other to make it easier to decide which words in $mt$ should be copied.

## 6 Conclusion

We have presented a new method for modeling the copying mechanism for automatic post-editing. By making the source sentence and machine translation attend to each other, representations learned in such an interactive way help to identify whether a target word should be copied or be re-generated. We also find that explicitly predicting words to be copied is beneficial for improving the performance of post-editing. Experiments show that our approach achieves new state-of-the-art results on the WMT 2016 & 2017 APE PBSMT sub-tasks.

## Acknowledgments

We thank all anonymous reviewers for their valuable comments. This work is supported by the National Key R&D Program of China (No. 2017YFB0202204), National Natural Science Foundation of China (No. 61761166008, No. 61432013), Beijing Advanced Innovation Center for Language Resources (No. TYR17002), and the NExT++ project supported by the National Research Foundation, Prime Ministers Office, Singapore under its IRC@Singapore Funding Initiative. This research is also supported by Sogou Inc.

## References


Hanna Béchara, Yanjun Ma, and Josef van Genabith. 2011. Statistical post-editing for a statistical mt system. In *MT Summit*.

Rajen Chatterjee, M. Amin Farajian, Matteo Negri, Marco Turchi, Ankit Srivastava, and Santanu Pal. 2017. Multi-source neural automatic post-editing: Fbk's participation in the wmt 2017 ape shared task. In *Proceedings of WMT*.

Rajen Chatterjee, Matteo Negri, Raphaël Rubino, and Marco Turchi. 2018. Findings of the wmt 2018 shared task on automatic post-editing. In *Proceedings of WMT*.

Rajen Chatterjee, Marion Weller, Matteo Negri, and Marco Turchi. 2015. Exploring the planet of the apes: a comparative study of state-of-the-art methods for mt automatic post-editing. In *Proceedings of ACL*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of ACL*.

Tianyu He, Xu Tan, Yingce Xia, Di He, Tao Qin, Zhibo Chen, and T. M. Liu. 2018. Layer-wise coordination between encoder and decoder for neural machine translation. In *Proceedings of NeurIPS*.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2016. Log-linear combinations of monolingual and bilingual neural machine translation models for automatic post-editing. In *Proceedings of WMT*.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2017. An exploration of neural sequence-to-sequence architectures for automatic post-editing. In *Proceedings of IJCNLP*.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2018. Ms-uedin submission to the wmt2018 ape shared task: Dual-source transformer for automatic post-editing. In *Proceedings of WMT*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Kevin Knight and Ishwar Chander. 1994. Automated postediting of documents. In *Proceedings of AAAI*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL*.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*.

Jindrich Libovický, Jindrich Helcl, Marek Tlustý, Pavel Pecina, and Ondrej Bojar. 2016. Cuni system for wmt16 automatic post-editing and multimodal translation tasks. In *Proceedings of WMT*.

Kathleen McKeown, Kristen Parton, Nizar Habash, Gonzalo Iglesias, and Adrià de Gispert. 2012. Can automatic post-editing make mt more meaningful? In *Proceedings of EAMT*.

Matteo Negri, Marco Turchi, Rajen Chatterjee, and Nicola Bertoldi. 2014. escape: a large-scale synthetic corpus for automatic post-editing. In *Proceedings of LREC*.

Jan Niehues, Eunah Cho, Thanh-Le Ha, and Alexander H. Waibel. 2016. Pre-translation for neural machine translation. In *Proceedings of COLING*.

Santanu Pal, Nico Herbig, Antonio Krüger, and Josef van Genabith. 2018. A transformer-based multi-source automatic post-editing system. In *Proceedings of WMT*.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of ACL*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. In *Proceedings of ACL*.

Jaehun Shin and Jong-Hyeok Lee. 2018. Multi-encoder transformer network for automatic post-editing. In *Proceedings of WMT*.

Michel Simard, Cyril Goutte, and Pierre Isabelle. 2007. Statistical phrase-based post-editing. In *Proceedings of HLT-NAACL*.

Amirhossein Tebbifakhr, Ruchit Agrawal, Matteo Negri, and Marco Turchi. 2018. Multi-source transformer for automatic post-editing. In *Proceedings of CLiC-it*.

Inigo Jauregi Unanue, Ehsan Zare Borzeshi, and Massimo Piccardi. 2018. A shared attention mechanism for interpretation of neural automatic post-editing systems. In *Proceedings of NMT@ACL*.

Dusan Varis and Ondrej Bojar. 2017. Cuni system for wmt17 automatic post-editing task. In *Proceedings of WMT*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of NeurIPS*.

Thuy-Trang Vu and Gholamreza Haffari. 2018. Automatic post-editing of machine translation: A neural programmer-interpreter approach. In *Proceedings of EMNLP*.

Robert A. Wagner and Michael J. Fischer. 1974. The string-to-string correction problem. *Journal of the ACM*, 21:168–173.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Gregory S. Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Baosong Yang, Zhaopeng Tu, Derek F. Wong, Fandong Meng, Lidia S. Chao, and Tong Zhang. 2018. Modeling localness for self-attention networks. In *Proceedings of EMNLP*.

Jiacheng Zhang, Yanzhuo Ding, Shiqi Shen, Yong Cheng, Maosong Sun, Huan-Bo Luan, and Yang Liu. 2017. Thumt: An open source toolkit for neural machine translation. *arXiv preprint arXiv:1706.06415*.

Wei Ke Zhao, Liang Wang, Kewei Shen, Ruoyu Jia, and Jingming Liu. 2019. Improving grammatical error correction via pre-training a copy-augmented architecture with unlabeled data. In *Proceedings of NAACL-HLT*.

Barret Zoph and Kevin Knight. 2016. Multi-source neural translation. In *Proceedings of HLT-NAACL*.

# Exploring Human Gender Stereotypes with Word Association Test

**Yupei Du , Yuanbin Wu , and Man Lan**

Department of Computer Science and Technology,
East China Normal University
{yupeidu.cs}@gmail.com {ybwu,mlan}@cs.ecnu.edu.cn

## Abstract

Word embeddings have been widely used to study gender stereotypes in texts. One key problem regarding existing bias scores is to evaluate their validities: do they really reflect true bias levels? For a small set of words (e.g. occupations), we can rely on human annotations or external data. However, for most words, evaluating the correctness of them is still an open problem. In this work, we utilize word association test, which contains rich types of word connections annotated by human participants, to explore how gender stereotypes spread within our minds. Specifically, we use random walk on word association graph to derive bias scores for a large amount of words. Experiments show that these bias scores correlate well with bias in the real world. More importantly, comparing with word-embedding-based bias scores, it provides a different perspective on gender stereotypes in words.[1]

## 1 Introduction

Consciously or not, gender stereotypes can be found in many places in our life. From proportions of female in certain occupations, to students' choices on their majors, from the stories people been told in books and movies, to the decisions they make every day, researchers try to build different stereotype models, to understand patterns behind various gender-related social and psychological phenomena.

Here, we investigate gender bias using the lens of language, especially, the words. Recent studies have shown that, by learning from large amounts of texts, low dimensional word embeddings (Mikolov et al., 2013) can exhibit human gender bias: the vector of "engineer" is more close to "he" than "she" (Bolukbasi et al., 2016). What's more insightful is that the revealed bias correlates well with census data, and its trends on historical texts matches women's movements in 1960-1970s (Garg et al., 2017). Thus word embeddings might be a reasonable tool in the study of gender stereotypes.

In this work, we try to resolve two problems regarding existing word-embedding-based bias scores. First, most embeddings are trained on word co-occurrence information in texts: "engineer" co-occurs with "he" more often than "she" in corpus. Since co-occurrence is only one of the various relations among words, it is natural to ask whether other types of relations can also reflect gender bias. Second, the validity of word-embedding-based bias scores are often examined using a small set of occupation words (i.e., how they correlate with occupation statistics in the real world) or a small set of human annotated words. How to evaluate validities of bias scores on the remaining large amount of words is still an open problem.

In order to answer these questions, we depart from existing models and explore a new tool for quantifying gender bias of words. Our method utilizes a recent result of large-scale word association test (De Deyne et al., 2018). In this test, participants are given a *cue* word and asked to write down most related words (*responses*) in their minds. Comparing with the co-occurrence relation, which subjects to contexts and topics of a document, word association test bears a more direct interaction with human brains. Therefore, it may provide a more detailed profile on human gender stereotypes.

We build a word association graph based on records of cue-response pairs. The vertices represent words, and an edge indicates whether two words are associated in some tests. With this

[1]Our code is publicly available at https://github.com/Yupei-Du/bias-in-wat.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6133–6143,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

graph, we can study how gender bias propagates among words using various graph-based algorithms. For example, a random walk starting from "he" and "she" can be applied to model the stochastic behaviors in real association tests, and the probabilities of reaching a word can be seen as its distances to "he" and "she". One advantage of graph-based models is that the derived bias scores are characterized by the intrinsic structure of the word association graph. Therefore, if we roughly assume the word association graph is an externalization of concept networks in human brains, as properties of the graph, the bias scores can be seen as a benchmark to help examining validities of other models.

In empirical evaluations, we compare the word-embedding-based bias scores with our word-association-based bias scores. First, regarding the real world census data and human annotations, word-association-based scores show stronger correlations than vanilla word-embedding-based scores. Second, as embeddings could also be applied for building graphs, we investigate such graphs and look into their differences with the word association graph. It turns out that both kinds of graphs exhibit "small world" properties, but they have different hub words and connecting mechanisms. We then run the same stereotype propagation algorithm and compare the derived gender bias scores. Third, as a case study, we use word-association-based scores as a benchmark to see the effectiveness of existing de-bias algorithms on word embeddings (Bolukbasi et al., 2016; Zhao et al., 2018). The results suggest that it is hard for both de-bias algorithms to remove gender stereotypes in word embeddings completely, which matches the conclusion in (Gonen and Goldberg, 2019).

## 2 Word Association Test

Word association test is a simple and sometimes entertaining game in which participants are asked to respond with the first several words that come out in their mind (the response) after being presented with a word (the cue)[2]. Table 1 lists some examples of the test. It is considered to be one of the most straightforward approaches for gaining insight into our semantic knowledge (Steyvers and Tenenbaum, 2005), and is also a common approach to measure words' meanings in our mind (so called mental lexicons (Jackendoff and Jackendoff, 2002)). For studies of gender stereotypes, word association test provides a way to inspect how a gender-specific word (e.g., "he", "she") connecting with other words, and the patterns of the connections may help us to probe gender stereotypes in our brains.

| Cue | R1 | R2 | R3 |
|---|---|---|---|
| way | path | via | method |
| extra | plus | special | additional |
| i | you | me | eye |
| come | go | closer | on |
| than | then | there | though |
| son | daughter | sun | boy |
| mind | brain | cognition | thinking |

Table 1: Examples of word association test records in SWOWEN datasets (De Deyne et al., 2018). R1, R2, R3 are participants' responses.

Several collections of English word association test records are publicly available (Nelson et al., 2004; Moss et al., 1996; Kiss et al., 1973). In this work, we focus on SWOWEN (De Deyne et al., 2018), which is currently the largest English word association database. The dataset is collected from 2011 to 2018, and contains more than 12000 cues and responses from over 90000 participants. The collection process begins with a small number of words as a word pool. Every time a participant starts a new test, a cue word is sampled randomly from the word pool. Once a new word not in the word pool is associated, it is added into the pool. This snowball sampling method helps to include both frequent and less-frequent cues at the same time.

## 3 Word Association Graph

Based on a table of word association test results, we try to figure out a complete picture on how gender biases are embedded in the test. We build a word association graph $G = (V, E)$ whose vertices $V$ are cues and responses, edges $E$ encode associations in the test, and weights of edges represent the strength of associations. Specifically, we take the following steps.

**Pre-Processing** Our pre-processing steps are mostly based on De Deyne et al. (2018)[3]. These

[2] https://en.wikipedia.org/wiki/Word_Association

[3] Raw word association test results and pre-process scripts are available in https://github.com/SimonDeDeyne/SWOWEN-2018. We re-implement

procedures consist of spell-check, unifying cue forms, removing untrustworthy participants, and Americanize all non-American spellings. Through these steps, we reduce word association records from 88722 participants, 12292 cues, and 4069086 responses to 83863 participants, 12217 cues, and 3665100 responses.

**Counting Association Pairs** We set all preprocessed cues and responses as vertices. An edge is added if it corresponds to a cue-response pair (directions of pairs are ignored for simplicity), and we count the occurrence numbers for edges as their weights. The final graph $G$ contains 12217 nodes and 1283047 edges.

Next, we will study how the gender bias propagates in $G$. More concretely, we investigate how gender information transit from a set of *gender-specific words* (i.e., words related with a gender by definition, such as "he", "she", "man", "woman") to other words. We obtain bias scores for *every* word based on the propagation and the graph structure, then we use them as a benchmark for evaluating validities of other models.

## 4 Stereotype Propagation

In the literature of psychology, Dell (1986) proposes a mental process in which words can spread its meaning by word associations. This theory suggests that once a gender-specific word in word association graph is activated, its influence will propagate along graph edges and form gender stereotypes. If a gender-neutral word (e.g. *nurse*) has stronger connections with gender-specific words in one gender (e.g. feminine words) than the other one, this gender-neutral word will receive imbalanced gender information, leading to unnecessary gender tendency. We thus tend to relate *nurse* more tightly with a woman than a man.

To characterize stereotypes of words, we simulate this mental process by spreading gender information from a set of commonly used gender-specific words $\mathcal{L}$ to the whole graph, using a random walk approach (Zhou et al., 2003). After enough iterations, gender information received by each word will eventually converge. Then we can calculate bias scores by seeing how much gender information they have received. We call this method *stereotype propagation*.

First, we build the set of gender-specific words

| | |
|---|---|
| he / she | brother / sister |
| father / mother | man / woman |
| son / daughter | boy / girl |
| husband / wife | uncle / aunt |
| grandfather / grandmother | gentleman / lady |

Table 2: Gender-specific word pairs in $\mathcal{L}$.

$\mathcal{L}$. In order to balance influence from different genders and avoid their heterogeneity, we collect them pair-by-pair. Specifically, $\mathcal{L} = \{(w_m^k, w_f^k)\}_{k=1}^{|\mathcal{L}|}$, where $w_m^k$ (a *masculine word*) and $w_f^k$ (a *feminine word*) are semantically related but different in gender. Table 2 is the $\mathcal{L}$ we use by default.

For each word, we use a 2-dimension vector $(b_m, b_f)$ to record the amount of gender information they will receive from masculine words ($b_m$) and feminine words ($b_f$). Let $P \in \mathbb{R}^{|V|\times 2}$ be the matrix containing vectors of all words, and $P_0$ be the initial state of $P$. For the gender-specific words in $\mathcal{L}$, we set their vectors in $P_0$ with $(1, 0)$ (masculine words) and $(0, 1)$ (feminine words). Vectors of other words are initialized with $(0, 0)$.

Next, following (Zhou et al., 2003), we iteratively update all words' masculine and feminine information until reaching a stationary $P^*$,

$$P_{t+1} = \alpha T P_t + (1-\alpha) P_0, \qquad (1)$$

where $T \in \mathbb{R}^{|V|\times|V|}$ is a normalization of $G$'s adjacent matrix $S$ (entries are edge weights),

$$T = D^{-\frac{1}{2}} S D^{-\frac{1}{2}}, \;\; D = \mathrm{diag}_i\Big(\sum_{j=1}^{N} S_{ij}\Big),$$

and $\alpha$ is a hyper parameter in range (0, 1), controlling the balance between *global* and *local* consistency. When $\alpha$ is small, we will pay more attention to keep the initial vectors of words (i.e., global consistency). When $\alpha$ gets larger, we will emphasize the smoothness between connected words: close words should have similar labels (i.e., local consistency).

We have a closed-form solution of Equation 1

$$P^* = (1-\alpha)(I - \alpha T)^{-1} P_0. \qquad (2)$$

It is worth noting that the solution only depends on the graph structure ($T$), gender-specific words ($P_0$), and a hyper parameter $\alpha$. We think that by minimizing the influence of parameters (only $P_0$

their R codes with Python.

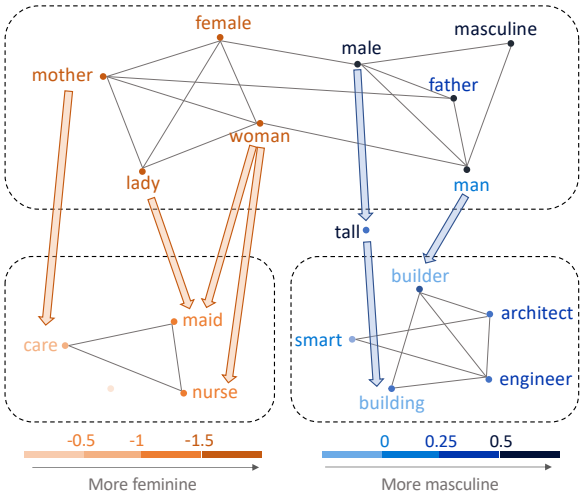


Figure 1: A snapshot on how stereotypes propagate in word association graph. Darker color on nodes means higher bias (visualize $b$). Colored arrows indicate how gender information propagate.

and $\alpha$), vectors in $P^*$ could be more robustly tied to the intrinsic properties of the word association graph and thus can be applied to (in)validate other bias scores.

Finally, we define a word's bias score $b$, from vectors in $P^*$,

$$b = \log \frac{\bar{b}_m}{\bar{b}_f},$$

where

$$\bar{b}_m = \frac{e^{b_m}}{e^{b_m} + e^{b_f}}, \quad \bar{b}_f = \frac{e^{b_f}}{e^{b_m} + e^{b_f}}.$$

We snapshot a sub-graph of $G$ with $P^*$ in Figure 1 to show how stereotype propagates. We first observe that words in the association graph are clustered. Those clusters are connected by some hubs (i.e., high degree words), through which gender information is able to spread quickly. As a result, words in a cluster usually have a same gender polarity. In Section 5.4, we will further show that $G$ enjoys some key small-word properties. Thus, inherently, people may be "lazy" at carefully bounding gender biases. Another finding in Figure 1 is that "building" bears a male bias since both buildings and men are "tall". This kind of connections are hard to observe in co-occurrences (word embeddings). We will also discuss details in Section 5.4.

## 5 Empirical Evaluation and Application

In this section, we first evaluate the word-association-based bias scores by examining whether it correlates to other real-world data. Then, we perform various controlled comparisons on settings of the stereotype propagation, including different graph structures, gender specific word sets $\mathcal{L}$ and propagation algorithms. Finally, as a case study, we use our proposed scores as a benchmark to evaluate the performances of two recent de-bias methods for word embeddings.

### 5.1 Alignments with Real-world Data

In this section, we perform two experiments to validate the feasibility and reliability of studying gender stereotypes using the word association test.

In the first experiment, we examine whether the bias detected from word association test can correlate with gender stereotypes in the real world. We adopt two data sources here, namely the gender proportion of occupations in U.S. census data (*census data* in Table 3) [4] (Ruggles et al., 2015), and manually annotated professions stereotypes by U.S. based crowd workers (*human judgments* in Table 3) (Bolukbasi et al., 2016). We measure the strengths of correlations by Pearson's $r$[5] and significance $p$ in Table 3.

Our second experiments compare average bias scores of words between different concepts using *Implicit Association Test* from (Caliskan et al., 2017). We adopt three pairs of concepts, namely *career vs. family*, *math vs. arts*, and *science vs. arts*[6]. Words in each pair are thought to have different gender tendencies. For example, we often relate *career* with a man but relate *family* with a woman. Words we use are shown in Table 4. For each pair, we report Cohen's $d$ [7] and significance $p$.

For comparison, we also conduct a similar analysis for word-embedding-based stereotype scores. We take two widely used settings, namely *we-cos* (Bolukbasi et al., 2016) and *we-norm* (Garg et al.,

[4]Follow Levanon et al. (2009), we choose the log proportions, log-prop($p$) = $\log \frac{p}{1-p}$, where $p$ = percentage of women in an occupation, similar with the way we calculate words' stereotype scores.

[5]Pearson's $r$ is a measure of the linear correlation between two variables. It has a value between 1 and -1, where 1 means total positive linear correlation, 0 means no linear correlation, and -1 means total negative linear correlation.

[6]Three words are not included in our word association graph, we thus substitute them by similar words, they are *cousins* to *cousin*, *equations* to *equation* and *computation* to *compute*.

[7]Cohen's $d$ is an effect size used to indicate the standardized difference between two means. Conventional small, medium, and large values of d are 0.2, 0.5, and 0.8 respectively.

| Method | Correlation Analysis | | | | Implicit Association Test | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Census data | | Human Judgments | | career vs. family | | math vs. arts | | science vs. arts | |
| | $r$ | $p$ | $r$ | $p$ | $d$ | $p$ | $d$ | $p$ | $d$ | $p$ |
| we-cos | 0.58 | $10^{-7}$ | 0.57 | $< 10^{-10}$ | **1.42** | .00 | **1.30** | .01 | 1.37 | .01 |
| we-norm | **0.59** | $10^{-7}$ | 0.57 | $< 10^{-10}$ | 1.18 | .02 | **1.30** | .01 | 1.37 | .01 |
| $G$ | 0.58 | $10^{-7}$ | **0.66** | $< 10^{-10}$ | 0.89 | .10 | 0.98 | .06 | **1.55** | .00 |
| $G^e_{global}$ | 0.04 | .60 | -0.01 | .85 | -0.20 | .60 | -0.08 | .88 | -0.31 | .51 |
| $G^e_{local}$ | -0.08 | .49 | 0.03 | .72 | -0.07 | .87 | -0.13 | .74 | -0.01 | .98 |

Table 3: Validation of stereotype scores. $r$ is the Pearson's $r$ in correlation analysis, $d$ is the effect size, and $p$ is the significance. $G$, $G^e_{global}$, $G^e_{local}$ represent stereotype propagation on different graphs. $\alpha$ is set to 0.99 by default.

| **concept** | **words** |
|---|---|
| career | executive, professional, corporation, salary, office, business, career |
| family | home, parents, children, family, cousin, marriage, wedding, relatives |
| maths | math, algebra, geometry, calculus, equation, compute, numbers, addition |
| arts | poetry, art, dance, literature, novel, symphony, drama |
| science | science, technology, physics, chemistry, Einstein, NASA, experiment, astronomy |

Table 4: Words in each concept.

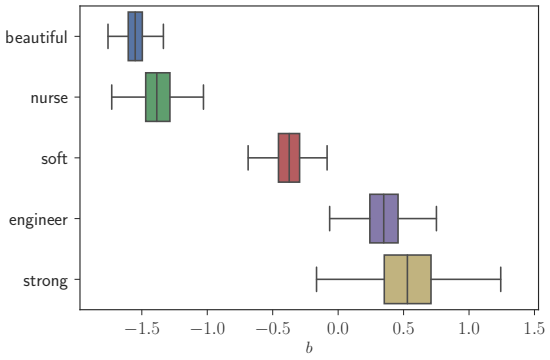


Figure 2: Examples of words' bias scores with confidence intervals (95%).

2017). They are both obtained by comparing a word $w$'s embedding similarities with masculine and feminine words in gender word pairs $\mathcal{L}$.

The bias scores in "we-cos" are defined as [8],

$$b = \frac{1}{|\mathcal{L}|}\sum_{k=1}^{|\mathcal{L}|}\Big(\frac{w^\intercal w_m^k}{||w||\cdot||w_m^k||} - \frac{w^\intercal w_f^k}{||w||\cdot||w_f^k||}\Big), \quad (3)$$

and in "we-norm", the scores are

$$b = -\frac{1}{|\mathcal{L}|}\sum_{k=1}^{|\mathcal{L}|}\big(||w - w_m^k||^2 - ||w - w_f^k||^2\big), \quad (4)$$

where $w_m$, $w_f$ is the embeddings of masculine and feminine words in $\mathcal{L}$. We re-train *word2vec* embeddings (Mikolov et al., 2013) on a Wikipedia dump.

Table 3 lists the correlation and implicit association test results. We find that

- Word-association-based bias scores have strong performances in both experiments. In correlation analysis, they exhibit similar or stronger relationships with real-world bias comparing to word-embedding-based bias scores. In implicit association test, our method still show large effects (Cohen's $d > 0.8$). It means that word association test is reliable in detecting human gender stereotypes.

- More importantly, although both word-embedding-based and word-association-based bias scores align well with real-world bias, gender stereotypes discovered from texts and human brains are different. The key evidence is that the word-association-based bias scores align much better on human annotated words.[9] Therefore, it is valuable and helpful to introduce a different perspective for lexical-level bias analyses.

### 5.2 The Influence of $\mathcal{L}$

To test the robustness of the proposed scores, we are interested in how $\mathcal{L}$ influences the results. We apply a bootstrap sampling strategy on the entire $\mathcal{L}$. Specifically, we sample 8 gender word pairs each time from $\mathcal{L}$. For all 45 possible combinations of the 8 gender words, we compute their bias scores to get confidence intervals w.r.t. to the full $\mathcal{L}$. We assume bias scores of a word to follow Gaussian distribution for easy calculation of confidence interval. We set the confidence level at 95%. Figure 2 illustrates some examples of words' bias scores with confidence intervals. We further

[8] We reuse symbol $w$ as word $w$'s embedding vector.

[9] Significantly higher correlation with human judgments than word-embedding-based methods (standard score Z=2.05).

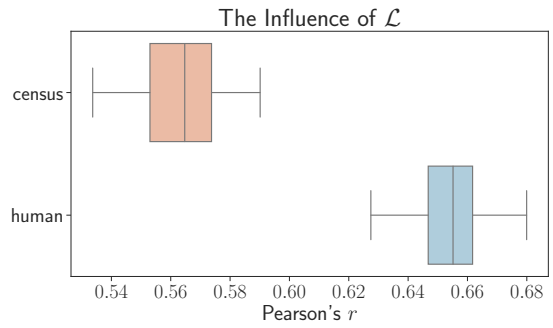


Figure 3: Confidence intervals (95%) of correlations between stereotype propagation results and real-world bias. Census stands for the census data dataset and human stands for the human judgments dataset.

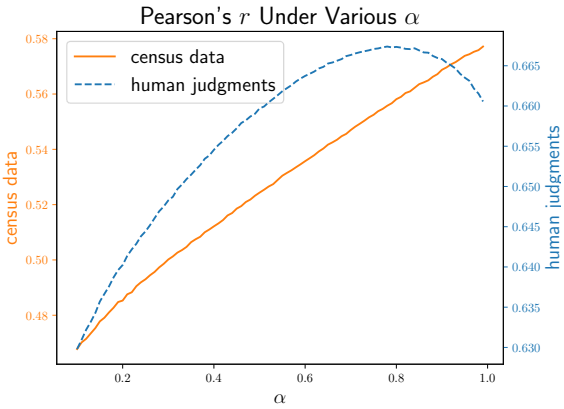


Figure 4: Correlation analysis under various $\alpha$.

calculate the confidence intervals of correlations between our results and gender stereotypes in the real world in Figure 3. The results show that the words' bias scores are stable with respect to reasonable settings of $\mathcal{L}$.

### 5.3 Variants of Stereotype Propagation

There are many variants of the random walk method in previous literature (Zhou et al., 2003; Zhu et al., 2003; Velikovich et al., 2010; Vicente et al., 2017). We experiment with another approach in (Zhu et al., 2003), to test whether bias scores of words are insensitive towards random walk algorithms. In this method, we consider only local consistency (without the second term of Equation 1). As a result, we find high consistency between their bias scores (Pearson's $r = 0.789$, $p = 0.0$).

We also examine the influence of hyper parameter $\alpha$. Here, we conduct similar correlation analysis in Section 5.1 under various $\alpha$ (Figure 4). We find that stereotype propagation has a stable performance regarding different $\alpha$, while a proper $\alpha$ (close to 1, allowing information to spread more freely) do helps.

### 5.4 Comparing with Word Embedding Graphs

Word embeddings can also be applied to build graphs on words: edges and weights can be defined based on similarities among word vectors. Previous work Hamilton et al. (2016) has adopted these kind of approaches to study the propagation of words' sentiment. Therefore, it is natural to compare the graphs built from word embeddings with the word association graph here.

#### 5.4.1 Word Embedding Graphs

We take two strategies, namely local and global, to construct two word embedding graphs $G^e_{local}$ and $G^e_{global}$. The local strategy is to connect a word with its $k$ nearest neighbors (same as Hamilton et al. (2016)). The global strategy is to find $e$ most similar word pairs in all possible word pairs as graph edges. To ensure that information will not be interrupted during propagation, we rectify both $G^e_{local}, G^e_{global}$ to their largest connected components [10]. To make a fair comparison, we make the two word embedding graphs have the same average node degree with the word association graph (105), hence, we set $k = 105$ and $e = 1235178$. We use the same word2vec embeddings in Section 5.

We run stereotype propagation on both embedding based graphs. As reported in Figure 3, unfortunately, both of them can hardly capture gender bias of words. To understand the reasons, we further compare the two types of graphs in detail.

#### 5.4.2 Graph Properties

We summarize some descriptive statistics of the three graphs in Table 5, where $L$ is the average shortest path length between every possible word pair. $D$ is the longest shortest path between two words in the graph. $C$ is the clustering coefficient, which is the average proportion of two arbitrary neighbors of a random word being themselves neighbors. When $C = 0$, no words have neighbors that are also each others' neighbors and when $C = 1$, the graph should be a fully connected graph. It measures how tightly words are

[10] We won't have the same problem in the word association graph, because a word will be added only when it is associated from another word.

| | Definition | $G^e_{local}$ | $G^e_{global}$ | $G$ |
|---|---|---|---|---|
| $\|V\|$ | number of words | 11870 | 11758 | 12217 |
| $K$ | average in degree | 104.69 | 105.05 | 105.02 |
| $L$ | average shortest path length | 2.89 | 3.43 | 2.12 |
| $D$ | diameter of the graph | 5 | 10 | 3 |
| $C$ | clustering coefficient | 0.27 | 0.40 | 0.24 |

Table 5: Descriptive statistics of graphs. $G$ is the word association graph.

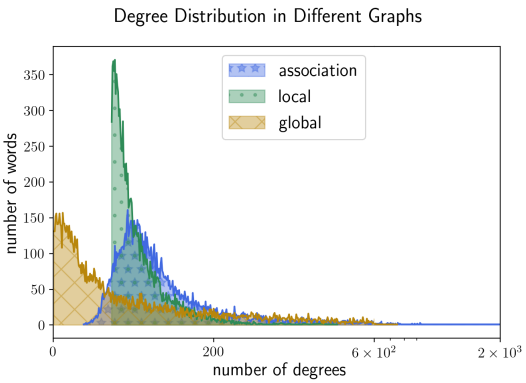


Figure 5: Distributions of node degrees.

organized as clusters.

*Small world* graphs are defined to be graphs where the distance between two arbitrary nodes follows a logarithmic distribution ($L \propto \log |V|$). In these graphs, node degree distributions follow power laws (Jeong et al., 2000), which means the degree of most nodes' are small, while a few nodes (*hubs*) have much larger degree number. Most nodes in the graph are organized as clusters connected by hubs, resulting in a relatively large clustering coefficient. These attributes work together to accelerate information flow within graphs.

From Table 5, we find that all three graphs exhibit small world properties. First, all three graphs have a quite small $L$ and $D$ ($L = 2.89, D = 5$, $L = 3.43, D = 10$, and $L = 2.12, D = 3$ for local, global, and word association graph respectively, while $\log 10000 = 9.21$). Second, their clustering coefficient $C$ are relatively large ($> 0.2$), while that of a typical random network is $10^{-3}$ (Steyvers and Tenenbaum, 2005). Third, as shown in in Figure 5, the distributions of node degrees have shapes of power laws (e.g., long tails). It's worth noting that in word association graph, words with an extremely small number ($< 10$) of degrees are rare. We think that one reason might be the strategy token in data collection. The word association test records are collected using a snowball sampling method (Section 2), and words with few degrees are new comers in the test, which have not been sampled as cue words for enough times. Another difference is that the curve of word association graph has a significant longer tail which may imply that it has more powerful hubs. This feature allows gender information to spread more quickly ($L$ is smaller) than the word embedding based graphs.

### 5.4.3 Types of Graph Edge

We also give a finer analysis on the two types of graphs to show their different ways on connecting gender specific words and other words. We examine *gender bias paths* which are all shortest paths [11] connecting words in $\mathcal{L}$ and other words in all three graphs.

According to semantic network theory (Collins and Loftus, 1975), there are three types of connections in human word association, namely *semantic priming* (Traxler, 2011), *neighborhood effects* and *frequency effects* (Andrews, 1989). Later, *co-occurrence* relation has also been found to hold a correlation with word association tests (Spence and Owens, 1990).

Our main observation is that, regarding the gender bias paths, we can find all of these connection types in bias paths of the word association graph. However, neighborhood effects and frequency effects are missed in word embedding graphs. This observation reveals that word association graph has a richer variety of connections, which may help the stereotype propagation. Examples of different connection types in both kind of graphs are exhibited in Table 6. We discuss each edge types in the following.

**Semantic priming** refers to the effect that word associations can happen between semantically related words. This is observed in bias paths on both word association graph and word embedding graphs, like *potential-maybe* in word association graph and *really-actually* in word embedding graphs. A more profound semantic relationship is hierarchical, including generic to specific and partitive relationships, such as *energy-carbohydrate* and *girl-person*.

**Neighborhood effects** refer to word associations between words that are highly confusable due to overlapping features, like spelling and pro-

[11] If several paths are equally short, we pick the one with highest cumulative weight.

| Type | Word Association Graph | Word Embedding Graph |
|---|---|---|
| semantic priming | boy - *energy* - *carbohydrate*<br>girl - *potential* - *maybe* | boy - kid - *really* - *actually*<br>*girl* - *person* |
| neighborhood effects | *son* - *some*<br>she - *pronoun* - *pro* - *projection* | X |
| frequency effects | husband - married - *a* - few<br>girl - lady - *the* - article | X |
| co-occurrence | father - *respect* - *others*<br>mother - *birth* - *day* | he - *even* - *more*<br>she - *cook* - *pickle* |

Table 6: Examples of different connection types between words. These words are shown in italics. Blue words refer to masculine word while red words refer to feminine word.

nunciation. Representative examples include *pronoun-pro-programmer* (spelling) and *son* to *some* (pronunciation). We manually check more than 100 bias paths on word embedding graphs, and didn't find this kind of connections, indicating its inability in modeling such effects.

**Frequency effects** suggest that human tends to associate commonly used words with higher possibility, making hubs in word association graph more likely to be frequent words. We pick top 2% words with largest node degrees as the representatives of hub words, and count the average occurrence of them in Wikipedia. As a result, the hubs of word association graph (297868) appears much more frequently than hubs in word embedding graphs (12931 of $G^e_{local}$, 3373 of $G^e_{local}$). It implies a stronger frequency effects in the word association graph.

To investigate how frequency effects influence stereotype propagation, we check the composition of these hub words. It turns out that most gender-specific words (except for *he*, *she*, *daughter*, *aunt*) in $\mathcal{L}$ are contained in the hub words of the word association graph, while none of them lie within those of the embedding-based graphs. Therefore, there is no wonder that $\mathcal{L}$ in the word association graph could distribute its gender information to other words in the graph more efficiently, leading to better performance in detecting gender bias of words.

**Co-occurrence** Due to the fact that most commonly used word embedding models are trained on word co-occurrence (Mikolov et al., 2013; Pennington et al., 2014), we observe lots of connections with such type in bias paths on word embedding graphs, which also widely exists in those on the word association graph, such as *birth-day* and *even-more*.

To summarize, frequency effects, neighborhood effects, and more concentrated hubs may work together to help spread gender stereotypes. It shows the value of introducing word association test into gender stereotype research.

### 5.5 Case Study: Do De-bias Methods Really Remove Gender Bias?

Some works have proposed approaches to reduce gender stereotypes in word embeddings. They can mainly be divided into two categories: post-processing step (Bolukbasi et al., 2016) and training goal modification (Zhao et al., 2018). Recent work (Gonen and Goldberg, 2019) has argued that those de-bias procedures only cover up but do not remove stereotypes. In this section, we utilize the bias scores from the word association graph to examine whether those de-bias methods work.

We experiment with de-bias methods of both categories, *Hard-Debias* refers to the method in (Bolukbasi et al., 2016) and *Gn-Glove* refers to the method in (Zhao et al., 2018). For each method, we compare it with an unde-biased version. For Hard-Debias, we compare with embeddings before these procedures [12] and for Gn-Glove, we compare with the original Glove (Pennington et al., 2014) offered in its project homepage [13].

For stereotype in word embeddings, we use the same measurements in Section 5, we-cos and we-norm. We examine whether those word embeddings still contain human-like stereotypes by con-

[12] We use the same word2vec embeddings in Section 5.

[13] https://github.com/uclanlp/gn_glove

| | | Original | | De-biased | |
|---|---|---|---|---|---|
| | | $r$ | $p$ | $r$ | $p$ |
| Hard-Debias | we-norm | 0.36 | .00 | 0.06 | $10^{-9}$ |
| | we-cos | 0.35 | .00 | 0.05 | $10^{-8}$ |
| Gn-Glove | we-norm | 0.38 | .00 | 0.34 | $< 10^{-10}$ |
| | we-cos | 0.38 | .00 | 0.34 | $< 10^{-10}$ |

Table 7: Correlation analysis between word association bias scores and bias scores from de-biased word embeddings. $r$ is Pearson's $r$ and $p$ is the significant value.

ducting a correlation analysis between their word-embedding-based bias scores and our word association bias scores. We expected that word association bias scores would have proper alignments with original word embeddings but poor alignments with de-biased versions if those de-bias procedures have done their jobs.

After removing a set of gender-specific words whose gender information are not removed in Hard-Debias, we extract the words appears both in our word association graph and the word embeddings vocabulary, 11311 words in total, as the word list for this case study.

Experiment results are shown in Table 7. We have three main observations:

- Both methods work to some extent, since Pearson's $r$s of both de-biased versions decrease compared with the original version.
- Neither Hard-Debias nor Gn-Glove removes gender stereotypes in word embeddings entirely, because bias scores derived from de-biased embeddings still exhibit significant correlations with the word association bias scores.
- Hard-Debias outperforms Gn-Glove. As we can see, Gn-Glove can hardly remove gender bias, which matches the conclusion in (Gonen and Goldberg, 2019).

## 6 Related Work

Studying gender stereotypes by language analysis (Hamilton and Trolier, 1986; Basow, 1992; Wetherell and Potter, 1993; Holmes and Meyerhoff, 2008; Coates, 2015) is now an import research topic. Although traditional methods like survey (Williams and Best, 1990) are quite effective, they are expensive and time-consuming. Therefore, Garg et al. (2017) use word embeddings as a quantitative lens to investigate historical trends of gender stereotypes. However, due to the limitations in word embedding training, existing methods have constrained effects in this field.

Word association test, as a product of psychology research, has been proven reliable on studying human implicit memory (Cañs, 1990; Playfoot et al., 2016). Different from word embeddings, word association test not only have a correlation with the co-occurrence of words (Spence and Owens, 1990) but also can reflect richer relationships like hierarchical relationships (Nuopponen, 2014), making it an excellent material in studying human stereotypes.

## 7 Conclusion

We utilize a recent large-scale word association test to explore how gender stereotypes propagate within our mind by constructing a word association graph from it, and derive gender bias scores of words. Experiments suggest that our approach is effective in detecting human stereotypes, and is tied robustly to graph structure. Therefore, these bias scores could be used to validate other methods. What's more, our results indicate that gender bias learned from large-scale texts is different from that within our minds, introducing a new perspective for lexical-level stereotype-related research.

## Acknowledgments

The authors wish to thank the reviewers for their helpful comments and suggestions, Qi Zheng, Tao Ji, Yuekun Yao, Changzhi Sun and Xinyue Chen for their useful comments on writing, Hoiyan Mak, Qing Cai, Bing Li, and Yang Yang for contributive discussions on psychology-related topics. This research is (partially) supported by STCSM (18ZR1411500) and the Fundamental Research Funds for the Central Universities. The corresponding author is Yuanbin Wu.

## References

Sally Andrews. 1989. Frequency and neighborhood effects on lexical access: Activation or search? *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 15(5):802.

Susan A Basow. 1992. *Gender: Stereotypes and roles*. Thomson Brooks/Cole Publishing Co.

Tolga Bolukbasi, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and Adam Tauman Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4349–4357.

Aylin Caliskan, Joanna J Bryson, and Arvind Narayanan. 2017. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186.

José J Cañs. 1990. Associative strength effects in the lexical decision task. *The Quarterly Journal of Experimental Psychology*, 42(1):121–145.

Jennifer Coates. 2015. *Women, men and language: A sociolinguistic account of gender differences in language*. Routledge.

Allan M Collins and Elizabeth F Loftus. 1975. A spreading-activation theory of semantic processing. *Psychological review*, 82(6):407.

Simon De Deyne, Danielle J Navarro, Amy Perfors, Marc Brysbaert, and Gert Storms. 2018. The "small world of words" english word association norms for over 12,000 cue words. *Behavior research methods*, pages 1–20.

Gary S Dell. 1986. A spreading-activation theory of retrieval in sentence production. *Psychological review*, 93(3):283.

Nikhil Garg, Londa Schiebinger, Dan Jurafsky, and James Zou. 2017. Word embeddings quantify 100 years of gender and ethnic stereotypes. *CoRR*, abs/1711.08412.

Hila Gonen and Yoav Goldberg. 2019. Lipstick on a pig: Debiasing methods cover up systematic gender biases in word embeddings but do not remove them. *CoRR*, abs/1903.03862.

David L Hamilton and Tina K Trolier. 1986. Stereotypes and stereotyping: An overview of the cognitive approach.

William L. Hamilton, Kevin Clark, Jure Leskovec, and Dan Jurafsky. 2016. Inducing domain-specific sentiment lexicons from unlabeled corpora. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 595–605.

Janet Holmes and Miriam Meyerhoff. 2008. *The handbook of language and gender*, volume 25. John Wiley & Sons.

Ray Jackendoff and Ray S Jackendoff. 2002. *Foundations of language: Brain, meaning, grammar, evolution*. Oxford University Press, USA.

Hawoong Jeong, Bálint Tombor, Réka Albert, Zoltan N Oltvai, and A-L Barabási. 2000. The large-scale organization of metabolic networks. *Nature*, 407(6804):651.

George R Kiss, Christine Armstrong, Robert Milroy, and James Piper. 1973. An associative thesaurus of english and its computer analysis. *The computer and literary studies*, pages 153–165.

Asaf Levanon, Paula England, and Paul Allison. 2009. Occupational feminization and pay: Assessing causal dynamics using 1950–2000 us census data. *Social Forces*, 88(2):865–891.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Helen Moss, Lianne Older, and Lianne JE Older. 1996. *Birkbeck word association norms*. Psychology Press.

Douglas L Nelson, Cathy L McEvoy, and Thomas A Schreiber. 2004. The university of south florida free association, rhyme, and word fragment norms. *Behavior Research Methods, Instruments, & Computers*, 36(3):402–407.

Anita Nuopponen. 2014. Tangled web of concept relations. concept relations for iso 1087-1 and iso 704. In *Terminology and Knowledge Engineering 2014*, pages 10–p.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543.

David Playfoot, Teodor Balint, Vibhuti Pandya, Averil Parkes, Mollie Peters, and Samantha Richards. 2016. Are word association responses really the first words that come to mind? *Applied Linguistics*, 39(5):607–624.

Steven Ruggles, Katie Genadek, Ronald Goeken, Josiah Grover, and Matthew Sobek. 2015. Integrated public use microdata series: Version 6.0 [dataset]. minneapolis: University of minnesota.

Donald P Spence and Kimberly C Owens. 1990. Lexical co-occurrence and association strength. *Journal of Psycholinguistic Research*, 19(5):317–330.

Mark Steyvers and Joshua B Tenenbaum. 2005. The large-scale structure of semantic networks: Statistical analyses and a model of semantic growth. *Cognitive science*, 29(1):41–78.

Matthew J Traxler. 2011. *Introduction to psycholinguistics: Understanding language science*. John Wiley & Sons.

Leonid Velikovich, Sasha Blair-Goldensohn, Kerry Hannan, and Ryan T. McDonald. 2010. The viability of web-derived polarity lexicons. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 2-4, 2010, Los Angeles, California, USA*, pages 777–785.

Iñaki San Vicente, Rodrigo Agerri, and German Rigau. 2017. Q-wordnet PPV: simple, robust and (almost) unsupervised generation of polarity lexicons for multiple languages. *CoRR*, abs/1702.01711.

Margaret Wetherell and Jonathan Potter. 1993. *Mapping the language of racism: Discourse and the legitimation of exploitation*. Columbia University Press.

John E Williams and Deborah L Best. 1990. *Measuring sex stereotypes: A multination study, Rev*. Sage Publications, Inc.

Jieyu Zhao, Yichao Zhou, Zeyu Li, Wei Wang, and Kai-Wei Chang. 2018. Learning gender-neutral word embeddings. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 4847–4853.

Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. 2003. Learning with local and global consistency. In *Advances in Neural Information Processing Systems 16 [Neural Information Processing Systems, NIPS 2003, December 8-13, 2003, Vancouver and Whistler, British Columbia, Canada]*, pages 321–328.

Xiaojin Zhu, Zoubin Ghahramani, and John D. Lafferty. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pages 912–919.

# A Modular Architecture for Unsupervised Sarcasm Generation

**Abhijit Mishra, Tarun Tater, Karthik Sankaranarayanan**
IBM Research
{abhijimi,ttater24,kartsank}@in.ibm.com

## Abstract

In this paper, we propose a novel framework for sarcasm generation; the system takes a literal *negative opinion* as input and translates it into a sarcastic version. Our framework does not require any paired data for training. Sarcasm emanates from *context-incongruity* which becomes apparent as the sentence unfolds. Our framework introduces incongruity into the literal input version through modules that: (a) filter factual content from the input opinion, (b) retrieve incongruous phrases related to the filtered facts and (c) synthesize sarcastic text from the filtered and incongruous phrases. The framework employs reinforced neural sequence to sequence learning and information retrieval and is trained only using unlabeled non-sarcastic and sarcastic opinions. Since no labeled dataset exists for such a task, for evaluation, we manually prepare a benchmark dataset containing literal opinions and their sarcastic paraphrases. Qualitative and quantitative performance analyses on the data reveal our system's superiority over baselines, built using known unsupervised statistical and neural machine translation and style transfer techniques.

## 1 Introduction

Sarcasm[1] is an intensive, ironic construct that is intended to express contempt or ridicule. It is often linked with intelligence, creativity, and wit, and therefore empowering machines to generate sarcasm is in line with the key goals of *Strong AI*[2]. From the perspective of Natural Language Generation (NLG), sarcasm generation remains an important problem and can prove useful in downstream applications such as conversation systems, recommenders, and online content generators. For instance, in a conversational setting, a more natural and intriguing form of conversation between humans and machines could happen if machines can intermittently generate sarcastic responses, like their human counterparts.

Over the years, a lot of research and development efforts have gone into the problem of detecting sarcasm in text, which aims to classify whether a given text contains sarcasm or not (Joshi et al. (2017b) provide an overview). However, systems for generation of sarcasm have been elusive. This is probably due to the fact that in sarcasm generation both *selection of contents* for sarcastic opinion generation and *surface realization of contents* in natural language form are highly nuanced.

In the broader area of style transformation of texts, most of the existing works have focused narrowly on transformations at lexical and syntax levels, *i.e.,* text simplification (Siddharthan, 2014), text formalization (Jain et al., 2018), sentiment style transfer (Shen et al., 2017; Xu et al., 2018), sentiment flipping (Li et al., 2018) and understanding humor (West and Horvitz, 2019). However, very little work has been done ((Piwek, 2003),(Hovy, 1987)) on incorporating *pragmatics* into generation tasks such as sarcasm. Sarcasm generation offers a rich playground to study this challenge and push the state-of-the-art in text transformation. Moreover, being a pragmatic task, sarcasm construction offers diverse ways to convey the same intent, based on cultural, social and demographic backgrounds. Hence, a supervised treatment of sarcasm generation using paired labeled data (such as parallel sentences) will be highly restrictive. This further motivates the need for exploring unsupervised approaches as the one we propose in this paper.

We make the first attempt towards automatic sarcasm generation where the generation is conditioned on a literal input sentence. For example, the literal opinion *"I hate it when my bus is late."*

[1] https://www.thefreedictionary.com/Sarcasm

[2] https://en.wikipedia.org/wiki/Artificial_general_intelligence



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6144–6154, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

should be transformed into *"Absolutely love waiting for the bus"*. As sarcasm conveys a negative sentiment, our system expects a negative sentiment opinion as input. Out of various possible theories proposed to explain the phenomenon of sarcasm construction (Joshi et al., 2017b), our framework relies on the theory of *context incongruity* (Campbell and Katz, 2012). Context incongruity is prevalent in textual sarcasm (Riloff et al., 2013; Joshi et al., 2015b). The theory presents sarcasm as a contrast between positive sentiment context (*e.g., absolutely loved it*) and negative situational context (*e.g., my bus is late*).

In our framework, translation of literal sentences to sarcastic ones happens in four stages *viz.*, (1) **Sentiment Neutralization**, during which sentiment-bearing words and phrases are filtered from the input, (2) **Positive Sentiment Induction**, where the neutralized input is translated into phrases conveying a strong positive sentiment, (3) **Negative Situation Retrieval**, during which a negative situation related to the input is retrieved, and (4) **Sarcasm Synthesis**, where appropriate sarcastic constructs are formed from the positive sentiment and negative situation phrases gathered in the first three stages.

Training and development of these modules require only three unpaired corpora of positive, negative, and sarcastic opinions. For evaluating the system, we manually prepare a small benchmark dataset which contains a set of literal opinions and their corresponding sarcastic paraphrases. Quantitative evaluation of our system is done using popular translation-evaluation metrics, and document similarity measurement metrics. For qualitative evaluation, we consider the human judgment of sarcastic intensity, fluency, and adequacy of the generated sentences. As baselines, we consider some of our simplistic model variants and existing systems for unsupervised machine translation and style transfer. Our overall observation is that our system often generates sarcasm of better quality than the baselines. The code, data, and resources are available at https://github.com/TarunTater/sarcasm_generation.

## 2 Challenges in Sarcasm Generation

Generation of sarcasm, unlike other language generation tasks, is highly nuanced. If we reconsider the example in the introductory section, the output sentence is sarcastic as it presents an unusual situation where the opinion holder has liked the rather boring act of *waiting for a bus*. The unusualness (and hence, the sarcasm) arises from two implicitly opposing (*incongruous*) contexts: *love* and *waiting for the bus*. Such a form of sarcasm, based on the *context incongruity* theory (Campbell and Katz, 2012), is more common in text than other forms such as prepositional, embedded or illocutionary sarcasm (Camp, 2012). For any textual sarcasm generator, figuring out contextually incongruous phrases will be as difficult as generating a fluent sentence. Moreover, most of the existing language generators are known to work on large scale literal/non-sarcastic texts (*e.g.,* language models trained on Wikipedia articles), and are agnostic of the possible collocations of contextually incongruous phrases (Joshi et al., 2017a). We try to overcome these challenges through our modular system design, discussed as follows.

## 3 System Architecture

The overall system architecture is presented in Figure 1. For development of the modules three corpora are needed: (a) a corpus of positive sentiment sentences ($\boldsymbol{\mathcal{P}}$ ), (b) a corpus of negative sentiment sentences ($\boldsymbol{\mathcal{N}}$ ), and (c) a corpus of sarcastic sentences ($\boldsymbol{\mathcal{S}}$ ). The framework performs transformation of literal text into sarcastic ones in four stages as given below:

### 3.1 Sentiment Neutralization

The neutralization module is designed to filter sentiment information out from the input text. For example, the input *hate when my bus is late* should be filtered to produce a neutral statement like *bus is late*. Neutralization is performed by a neural sentiment classification module. Each word in the input sentence of length $N$ ($\boldsymbol{x} = \{x_1, x_2, ..., x_N\}$) (in *one-hot* representation, padded wherever necessary) is transformed into $K$-dimensional embedding. The embeddings are then passed through a layer of recurrent units such as Long Short Term Memory (LSTM)(Hochreiter and Schmidhuber, 1997). The output of the LSTM layers are then sent to a *self-attention* layer before passing through a `softmax` based classifier. The classifier is trained with the supervision from sentiment positive/negative labels using corpora $\boldsymbol{\mathcal{P}}$ and $\boldsymbol{\mathcal{N}}$ .

During testing, for a given input of length $N$, the self attention vector $\boldsymbol{\alpha} = \alpha_1, \alpha_2, ..., \alpha_N$ is first extracted (details skipped for brevity, refer Xu

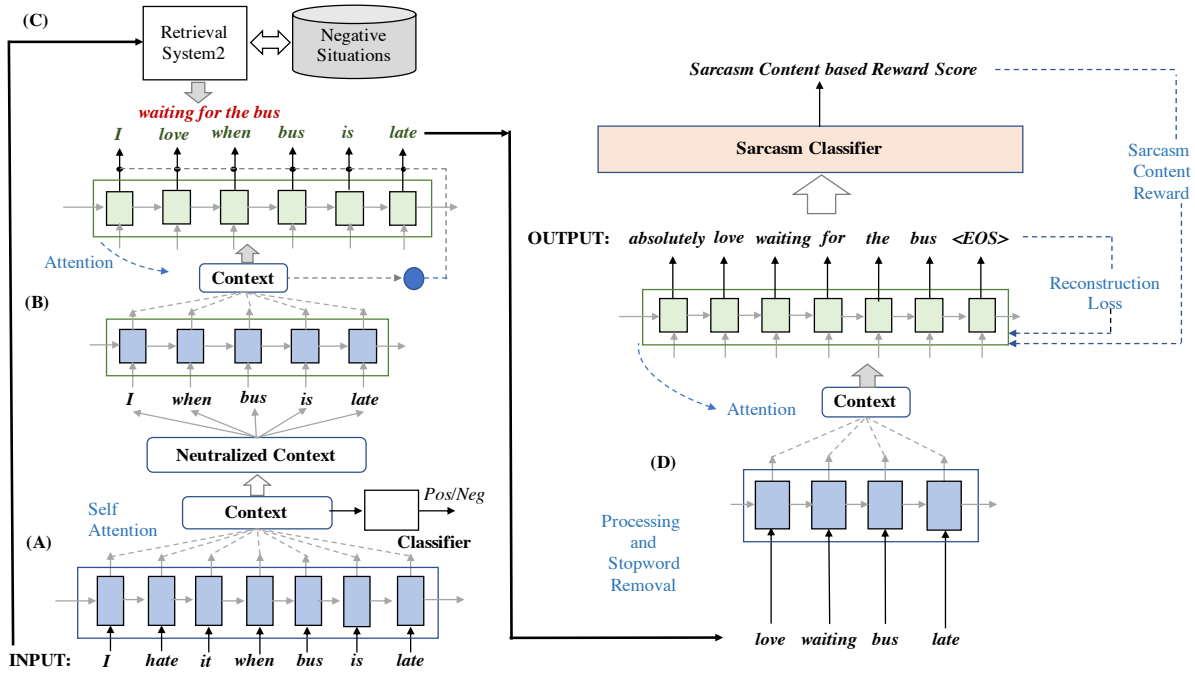


Figure 1: System Architecture. Input to the system are literal sentences. The blue and green boxed represent layers of embedding and LSTMs. The components are (A) Sentiment Neutralizer (B) Positive Sentiment Inducer (C) Negative Situation Retriever , and (D) Sarcasm Synthesizer.

et al. (2018)). We then inverse and discretize the self attention vector as follows:

$$\hat{\alpha_i} = \begin{cases} 0, & \text{if } \alpha_i > \mu + 2\sigma \\ 1, & \text{otherwise} \end{cases} \quad (1)$$

where $\alpha_i$ is the attention weight for the $i^{th}$ word, $\mu$ and $\sigma$ are the mean and standard deviation for the attention vector. For each word in the input, if the discretized attention weight is 0, it is filtered out.

The motivation behind such a design is that if the classifier is trained well, the attention weights will represent the contribution of each word to the overall sentiment decision (Xu et al., 2018). Sentiment bearing words will receive higher attention whereas neutral words will have lower attention weights. It is worth noting that neutralization can be done in several other ways such as filtering based on a sentiment lexicon. However, such operations would require additional resources such as sentiment dictionary, sense disambiguation tools, whereas the neural classification based filtering can only work with binary sentiment labeled data. Also, for computing word-level sentiment contributions, recent techniques (such as gradient-based methods (Sundararajan et al., 2017)) can be used. For simplicity, we use attention based filtering.

### 3.2 Positive Sentiment Induction

Once the neutralized output is extracted, it is passed to the positive sentiment induction module which transforms it into a positive sentiment sentence. For this, we use a traditional sequence to sequence pipeline (Bahdanau et al., 2014) with attention and copy mechanisms (Gulcehre et al., 2016). The input is a set of words coming from the neutralization module. These are transformed into embeddings and are then encoded with the help of LSTM layers. The decoder attends over the encoded output and produces the output tokens based on the attended vector and the previously generated tokens. This is a standard technique, typically used in neural machine translation.

As the output from the system is expected to be positive in sentiment, for training the framework, we use only a set of positive sentences from $\mathcal{P}$ . Each sentence in the data is filtered using the neutralization module. The filtered version, and the original positive sentence are used as $\langle source, target \rangle$ pairs.

### 3.3 Negative Situation Retrieval

Negative situations present in sarcastic opinions are typically extrinsic and are loosely related to the semantics of its literal version. Hence, a sequence to sequence module analogous to Section

| |
|---|
| **5-grams:** getting up for school facts, getting yelled at by people, trying to schedule my classes, feeling like every single person, walking to class in pouring, making people who already hate, working on my last day, spending countless hours at doctors, getting overdraft statements in mail |
| **4-grams:** talking about world politics, stuck in a generation, sitting in class wondering, canceled at short notice, distancing myself from certain, wipe my own tears |
| **3-grams:** born not breathing, paid to sleep, scared those faces, taking a shower, starting your monday, accused of everything, worrying about someone, fight jealousy arguments, license to trill, awarded literature prize |
| **2-grams:** scratching itchy, looking chair, getting hiv, shot first, collecting death, lost respect |
| **1-gram:** canceled, sleeping, trying, buying, stapling |

Table 1: Example negative situations extracted using bootstrapping technique (Riloff et al., 2013)

3.2 may not be very useful. Moreover, for sarcasm generation, for a certain topic, it is safe to assume that there can be a finite set of negative situations. From this set appropriate situation phrases can be *"retrieved"* depending on the given input. Thus, finding out appropriate negative situations boils down to two sub-problems of (a) preparing a finite set of negative situations, and (b) setting up the negative situation retrieval process. We discuss each of these two steps below.

#### 3.3.1 Building *Negative Situation* Gazetteer

This is a one-time process and is done using the unsupervised bootstrapping technique similar to Riloff et al. (2013). For each sentence in the sarcasm corpus $\boldsymbol{\mathcal{S}}$ , a candidate negative situation phrase is extracted. A candidate negative situation phrase is a word n-gram ($n \leq 5$) that follows a positive sentiment phrase in a sarcastic sentence [3]. After the candidates for a positive phrase are obtained, their Part of Speech tags are extracted with the help of a POS tagger. Specific patterns of n-gram are then obtained using the POS tags. This ensures that the phrases extracted are mostly *verb phrases*, *noun-phrases*, and *to-infinitive* verb phrases that describe situations. In our setting we use 30 predefined POS n-gram patterns following Riloff et al. (2013).

Once the candidate negative situation phrases are extracted, they are filtered based on a scoring function as given below:

$$score_i = \frac{\#ns_i \; in \; \boldsymbol{\mathcal{S}}}{\#ns_i \; in \; \boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{N}}} \qquad (2)$$

where $ns_i$ is the $i^{th}$ negative situation extracted for a certain positive phrase.

The scoring function returns a real value indicating the exclusiveness of the negative situation w.r.t the sarcastic sentences. If the score exceeds a threshold (*i.e.,* $p > 0.5$), the candidate phrase is added to the gazetteer. Once all the possible negative situation phrases are extracted, each phrase is used to extract more positive sentiment phrases similarly as above. This process of positive phrase and negative situation extraction is repeated until no new phrases are found. Table 1 shows some example negative situation phrases extracted from our dataset.

#### 3.3.2 Retrieval Process

The idea is to find negative situations relevant to the input sentence. We implement an information retrieval system based on PyLucene. All the negative situations from the gazetteer (Sec. 3.3.1) are first indexed. The input sentence is considered as the query for which the most relevant negative situation is retrieved from the indexed list. The factors involved in PyLucene's retrieval algorithm include *tf-idf*, number of matching terms in the query sentence and the retrieved sentence, and importance measure of a term according to the total number of terms in the search.

Once the positive sentiment and negative situations are generated for the input sentence, they undergo a post-processing step where stopwords and redundant words are removed and given as input to the sarcasm synthesis module.

### 3.4 Sarcasm Synthesis

The sarcasm synthesis module is a sequence-to-sequence network that expects a set of keywords related to positive sentiment and negative situation phrases. For training this module, the sarcasm corpus $\boldsymbol{\mathcal{S}}$ is used. To prepare the input, we implement a keyword extraction technique based on POS tagging. Sentences in $\boldsymbol{\mathcal{S}}$ are POS-tagged and then stopwords are removed, and then based on the POS tags noun, verb, adjective and adverbs are retained. This way, the input keywords to the system would somewhat be similar to the keywords expected in real time scenario.

[3] the word "love" is considered as the seed positive sentiment phrase to begin the bootstrapping procedure

The module follows an encode-attend-decode style architecture like the positive sentiment induction module, but with a different learning objective. Keywords in the input (in *one-hot* representation, padded wherever necessary) are transformed into a sequence of embeddings and then encoded by layers of LSTMs, which produces a hidden representations for each input word. The decoder, consisting of a single layer of LSTMs stacked on the top of a decoder embedding layer, attend over the encoded hidden representations and generate target words. In general, for $T$ training instances of keywords and sarcastic texts, $\{x^i, y^i\}_{i=1}^{T}$, the training objective is to maximize the likelihood of a target $y^i$ given $x^i$, which is similar to minimizing the *cross entropy* between the target distribution and the predicted output distribution. For training the neural network, the cross-entropy loss is back propagated. In other words, the gradient of the negative cross-entropy loss is considered to update the model parameters. The gradient is given by:

$$\nabla_\theta L(\theta) = \nabla_\theta \sum_{i=1}^{T} y^i log\ P_{M_\theta}(\hat{y^i}|x^i) \qquad (3)$$

where $L$ is the loss function and $M_\theta$ is the translation system with parameter $\theta$. $\hat{y^i}$ is the predicted sentence. In our setting, where the input is not a proper sentence, the problem with the above objective is that it does not strongly enforce the decoder to learn and produce sarcastic output. We speculate that minimizing the token-level cross entropy loss in Eq. 3 may help produce an output that is grammatically correct but not sarcastic enough. For instance, the decoder may incur insignificant cross-entropy loss after generating a sentence like *absolutely loved it*, as this sentence has considerable overlap with the reference sarcastic text that provides supervision.

One possible idea to tackle such problems is to employ a sarcasm scorer that can determine the sarcasm content in the generated output, and use the scores given by the sarcasm scorer for better training of the generator. However, the sarcasm scorer may be external to the sequence-to-sequence pipeline, and the scoring function may not be differentiable with respect to the model $M_\theta$. For this, we apply reinforcement learning which considers sarcasm content score as a form of *reward* and use it to fine-tune the learning process. For learning, the *policy gradient theorem* (Williams, 1992) is used. The system is trained under a modified learning objective *i.e.,* to maximize the expected reward score for a set of produced candidate sentences. The generator, operating with a policy of $P_{M_\theta}(\hat{y^i}|x^i)$, producing an output $\hat{y^i}$ with an *expected* reward score computed using a scorer, will thus have the following gradient:

$$\begin{aligned} \nabla_\theta RL(\theta) &= \nabla_\theta \mathbb{E}_{\hat{y_i} \sim P_{M_\theta}(\hat{y^i}|x^i)}[R(\hat{y^i})] \\ &= \mathbb{E}[R(\hat{y^i}) \nabla_\theta log(P_{M_\theta}(\hat{y^i}|x^i))] \end{aligned} \qquad (4)$$

where $RL$ is the modified learning objective which has to be maximized and $R$ is a reward function that is computed using an external scorer. In practice, the expected reward is computed by (a) sampling candidate outputs from the policy $P_{M_\theta}$, (b) computing the reward score for each candidate and (c) averaging the rewards so obtained. In typical RL settings, the learner is typically initialized to a random policy distribution. However, in our case, since some supervision is already available in the form of target sarcastic sentences, we pretrain the model with the loss minimization objective given in Eq. 3 and then fine-tune the model based on the policy gradient scheme following Eq. 4. Thus, the learner gets initialized with a better policy distribution.

For reward calculation, we consider the confidence score of a sarcasm classifier (probability of being sarcastic) trained using $\boldsymbol{\mathcal{S}}$ as positive examples and $\boldsymbol{\mathcal{P}}$ and $\boldsymbol{\mathcal{N}}$ taken as negative examples. For our setting, the classifier is analogous to the classifier used for neutralization. The classifier is based on embedding, LSTM and softmax layers.

Since the input to the system is a list of words, it may seem that the sarcasm synthesis module may not require sequence to sequence learning, and a much simpler approach like bag-of-words to sequence generation could have been used. However, note that the input to the generator is obtained after dropping words during neutralization and later appending the negative situation phrase. The sequentiality is, thus, not completely lost. This makes sequence to sequence model an intuitive choice.

We now explain our experimental setup.

## 4 Experiment Setup

### 4.1 Datasets

As stated earlier, our system does not rely on any paired data for training. It requires three corpora of positive sentences, negative sentences, and sarcastic sentences collected independently.

For positive and negative sentiment corpora $\mathcal{P}$ and $\mathcal{N}$, we considered short sentences/snippets from the following well-known sources such as (a) Stanford Sentiment Treebank Dataset, (b) Amazon Product Reviews, (c) Yelp Reviews (d) Sentiment 140 dataset (See Kotzias et al. (2015) for sources). The above datasets primarily contain tweets and short snippets. Tweets are normalized by removing hashtags, usernames, and performing spell checking and lexical normalization using NLTK (Loper and Bird, 2002). We then filtered out sentences with more than 30 words. Approximately $50,000$ sentences from each category are retained. Then, based on the vocabulary overlap with our sarcasm corpus $\mathcal{S}$, $47,827$ sentences are finally retained from each category (total number of instances is $95654$).

For the unlabelled sarcasm corpus $\mathcal{S}$, we relied on popular datasets used for sarcasm detection tasks such as the ones by Ghosh and Veale (2016), Riloff et al. (2013), and the Reddit Sarcasm Corpus[4]. Sentences are denoised, spell corrected and normalized. Average sentence length is kept as 30 words. A total number of $306,141$ sentences are thus collected. A common vocabulary of size $20,000$ is extracted (based on frequency) for all the modules from the three corpora. Each corpus is divided into a train-valid-test split of $80\%$-$10\%$-$10\%$.

#### 4.1.1 Benchmark Dataset for Evaluation

Since no dataset containing paired examples of literal and sarcastic utterances are available, we created a small test-set for evaluating our system. From the test split of the sarcasm corpus $\mathcal{S}$, 250 sentences on diverse topics are selected and are manually translated into literal versions by two linguists. From this, only 203 sentences could be selected by the linguists who mutually decided whether the sentences were sarcastic enough to keep in the test dataset or not.

[4]https://www.kaggle.com/danofer/sarcasm

### 4.2 Model Configuration

For the neutralization module, the embedding dimension size is set to 128, two layers of LSTMs of hidden dimension of 200 are used. The classifier trains for 10 epochs with a batch size of 32, and achieves a validation accuracy of $96\%$ and training accuracy of $98\%$.

For positive sentiment induction module, the embedding dimensions for both encoder and decoder are set to 500. Both the encoder and decoder have only one layer of LSTM, with a hidden dimension of 500. The module is built on top of the OpenNMT (Klein et al., 2017) framework. Training happens in $100,000$ iterations and the batch size is set to 64. The positive sentiment induction module, at the end of the training, produces a bigram BLEU (Papineni et al., 2002) score of $62.25\%$. For bootstrapping negative situations and other purposes, the POS tagger from Spacy[5] is used. The Lucene-IR framework is set up to retrieve negative situations.

The model configuration and training parameters for the sarcasm synthesizer is the same as the positive sentiment induction module. For the RL scheme, for each instance, the expected reward is computed over 100 candidate samples. At the end of the training, the bigram BLEU score on the validation set turns out to be $59.3\%$. For reward computation, we use a classifier similar to the one used for neutralization. The embedding size for this classifier is 300 and it uses two layers of unidirectional LSTMs with a hidden dimension of 300. It trains with a batch size of 64 and produces a validation accuracy of $78.3\%$. The probability estimates given by the classifier for any input text are taken as reward scores. For optimization, cross entropy loss criterion is used.

### 4.3 Evaluation Criteria

Absence of automatic evaluation metrics capable of capturing subtleties of sarcasm makes it difficult to evaluate sarcasm generators. For evaluation, we still use the popular translation and summarization evaluation metrics METEOR (Banerjee and Lavie, 2005) and ROUGE (Lin, 2004). Additionally, to check the semantic relatedness between the input and output, we use Skip-thought sentence similarity metric[6]. Note that using BLEU (Papineni et al., 2002) will be futile here as direct

[5]http://spacy.io

[6]https://github.com/Maluuba/nlg-eval

n-gram overlaps between the predicted and gold-standard sentences are not expected to be significantly higher for such a task. We still include it as an evaluation metric for completion.

We employ an additional metric to judge the percentage of length increment (abbreviated as WL) to see the if the length of the output is generally more than that of the input (for the reference text it is 67%). The notion behind this metric is that sarcasm typically requires more context than its literal version, requiring to have more words present at the target side.

#### 4.3.1 Human Judgement based Evaluation

We also consider human judgment scores indicating whether the generated output is non-sarcastic/sarcastic (0/1 labels), how fluent it is (in a scale of 1-5, 1 being lower), and to what extent it is related to the input (in a scale of 1-5). The relatedness measure is important as the objective of the task is to produce a sarcastic version of the input text without altering the semantics much. For human evaluation, we consider only the 30 sentences randomly picked from the benchmark (test) dataset. Sarcasm is a difficult topic, so we stuck to only two annotators who had a better understanding of the language and socio-cultural diversities.

### 4.4 Systems for Comparison

For comparison, we consider the following four systems:

1. **SarcasmBot**: This is an open-sourced sarcasm generation chatbot released by Joshi et al. (2015a). The system generates a sarcastic response to an input utterance.

2. **UNMT:** This system is based on Unsupervised Neural Machine Translation technique by Artetxe et al. (2017), which can be extended to any translation task. In our setting, the source and target side are literal and sarcastic utterances, *i.e.* the direction of translation is non-sarcastic to sarcastic.

3. **Monoses:** This is similar to UNMT but based on unsupervised Statistical Machine Translation (Artetxe et al., 2018).

4. **ST:** This is based on the cross alignment technique proposed by Shen et al. (2017), used for the task of sentiment translation.

5. **FLIP:** This is based on heuristics for sentiment reversal. For this, the input sentence is first dependency-parsed. The root verb is determined along with its tense and aspects with the help of its part-of-speech tags[7]. The sentiment of root verb is determined using sentiment lexicon[8]. If the verb has non-zero positive or negative sentiment score, its antonym is found using WordNet. Appropriate tense and aspect form of the antonym is then obtained[9]. The modified antonym replaces the original root verb. Similarly, we replace adjective and adverbs with words carrying opposite sentiment.

For training the above systems (except FLIP), we used $\mathcal{S}$ at one side and a larger version of combined $\mathcal{P}$ and $\mathcal{N}$ containing $558,235$ sentences on the other side, curated from the same sources as mentioned earlier. Apart from this system, we also tested some of our model variants, which are presumably inferior and can be considered as baselines. These are termed as:

1. **SG_NORMAL:** a system with only the sarcasm synthesizer module which takes the input directly (after removing stopwords from the input),

2. **SG_RL:**, same as *SG_NORMAL* but also applies reinforcement learning,

3. **ALL_NORMAL:**, the complete system, with sarcasm synthesizer trained without reinforcement learning strategy.

4. **ALL_RL:**, the complete system with reinforcement learning.

## 5 Results and Analysis

Tables 2 and 3 present evaluation results. While it was expected that the automatic metrics may not be able to capture the subtleties of sarcasm, the WL measure indicates that a carefully designed modular approach like ours often generates longer sentences with more context. This is also corroborated by the human evaluation where annotators have judged that the output generated from

[7]For parsing and POS-tagging spaCy (http://spacy.io/) is used

[8]https://www.nltk.org/_modules/nltk/sentiment/vader.html

[9]https://www.nodebox.net/code/index.php/Linguistics

| System | BLEU | METEOR | ROUGE-L | SkipT | WL |
|---|---|---|---|---|---|
| SarcasmBot | 0.002 | 0.012 | 0.01 | 0.20 | -43% |
| UNMT | 0.17 | 0.15 | 0.33 | 0.45 | -1.1% |
| Monoses | 0.12 | 0.13 | 0.25 | 0.42 | -3.4% |
| ST | 0.017 | 0.04 | 0.11 | 0.24 | -4.3% |
| FLIP | 0.10 | 0.17 | 0.34 | 0.45 | 0.003% |
| SG_NORM | 0.09 | 0.13 | 0.24 | 0.41 | -3.3% |
| SG_RL | 0.09 | 0.13 | 0.24 | 0.40 | -4.18 |
| ALL_NORM | 0.12 | 0.13 | 0.23 | 0.37 | 32.3% |
| ALL_RL | 0.11 | 0.13 | 0.23 | 0.37 | 31.2 |

Table 2: Evaluation results for our system and various baselines.$SkipT \rightarrow$ skip thought similarity

| System | Sarcasm | Fluency | Adequacy |
|---|---|---|---|
| SarcasmBot | 86.6% | 4.3 | 1.2 |
| UNMT | 20% | 3.9 | 3.8 |
| Monoses | 33.3% | 3.7 | 3.5 |
| ST | 16% | 2.8 | 2.3 |
| FLIP | 20% | 4.1 | 3.8 |
| SG_NORM | 56.6% | 3.7 | 3.2 |
| SG_RL | 63.3% | 3.9 | 3.6 |
| ALL_NORM | 63.3% | 3.7 | 3.9 |
| ALL_RL | 73.3% | 3.7 | 3.8 |

Table 3: Human judgment scores for various systems

our system are more sarcastic than the comparison systems. *SarcasmBot*, being a heuristic driven sarcasm generator produces sarcastic responses but is not related to the input topic. Moreover, it ends up generating only 20 different responses for our entire test dataset making its output redundant and unrelated to the input. Other existing systems such as *UNMT* and *Monoses* converge to autoencoding and end up replicating the input as output. *FLIP*, performs transformations at lexical level, hence achieves better fluency but certainly fails to induce sarcasm in most of the cases.

Table 4 presents example generations from different systems. It is quite interesting to note that due to the RL, the model tends to produce longer sentences and brings additional context necessary for sarcasm. The fluency is however compromised. A close inspection of the outputs from each module suggests that the overall error committed by the system is due to accumulation of different types of errors, mainly (a) error during neutralization due to inappropriate assignment of weights to the words in the input, (b) dropping of words and/or insertion of spurious words during positive sentiment induction, and (c) error in scoring the sarcasm content in the RL setting. These can be addressed through better hyper-parameter tuning, gathering more training data for training the individual modules (especially the sarcasm synthesizer), and fine-tuning the sarcasm scorer.

| **Input: worrying because did not finish my homework.** |
|---|
| **Reference:** *did not finish any homework & i still need to shower ! lol ! love stressing out.*<br>**SarcasmBot:** How exciting! $\star rolls\ eyes \star$ .<br>**Monoses:** whining because actually finish my homework.<br>**UNMT:** worrying because i did not finish my homework.<br>**ST**: if do not work my mom hurts.<br>**FLIP**: reassuring because did not finish my homework.<br>**SG (NORMAL)**: i am worrying about the worrying of homework.<br>**SG (RL)**: worrying about finish homework.<br>**ALL (NORMAL):** no worrying , i finish doing homework great.<br>**ALL (RL):** worrying about finish homework is great. |
| **Input: swimming lessons are very costly in nyc.** |
| **Reference:** *you have to sacrifice your first born child for swimming lessons in nyc.*<br>**SarcasmBot:** That is a very useful piece of information! Yawn!<br>**Monoses:** Dry lessons are very costly financially.<br>**UNMT:** swimming lessons are very costly in nyc.<br>**ST:** a will was in bed.<br>**FLIP**: swimming lessons are very costly in nyc.<br>**SG (NORMAL)**: this is so costly to me swimming lessons in nyc.<br>**SG (RL)**: swimming lessons is so costly in nyc.<br>**ALL (NORMAL):** loving the swimming lessons in nyc ch.<br>**ALL (RL):** i am loving the swimming lessons . going to be a very costly in nyc ch. |

Table 4: Examples of generated outputs from different systems. *Reference*$\rightarrow$ human written sarcastic text

## 6 Related Work

As stated earlier, not many systems for sarcasm generation exist today. The closest work to ours is the one by Joshi et al. (2015a) which employs a heuristic driven approach for generating a sarcastic response to an input utterance. Since, the output of the system is a response, the system is not suitable for translating a literal input text into a sarcastic version. Unlike sarcasm generation, sar-

casm detection has been a well-known problem with several available solutions. For this problem, traditional supervised and deep neural network based solutions have been proposed. The supervised approaches rely on: (a) Unigrams and Pragmatic features (González-Ibánez et al., 2011; Barbieri et al., 2014; Joshi et al., 2015b) (b) Stylistic patterns (Davidov et al., 2010) and patterns related to *situational disparity* (Riloff et al., 2013) and (c) Cognitive features extracted from gaze patterns (Mishra et al., 2016, 2017). Recent systems are based on variants of deep neural networks built on the top of embeddings. Deep neural networks based solutions for sarcasm detection include (Ghosh and Veale, 2016) who uses a combination of RNNs and CNNs for sarcasm detection, and (Tay et al., 2018), who propose a variant of CNN for extracting features related to context incongruity.

A few works exist in the domains of irony, pun and humour generation and are summarized by Wallace (2015), Ritchie (2005) and Strapparava et al. (2011) respectively. However, most of these are heuristic driven and, hence, may not be easily scaled to new domains and languages. From the perspective of language style transfer. Shen et al. (2017) propose an unsupervised scheme to learn latent content distribution across different text corpora and use it for sentiment style transfer. Xu et al. (2018) introduce an unsupervised sentiment translation technique through sentiment neutralization and reinforced sequence generation. Zhang et al. (2018) propose a style transfer technique based on unsupervised MT inspired by Artetxe et al. (2017). Artetxe et al. (2018) have recently proposed an unsupervised statistical machine translation scheme. We adopt some of these modules for the task of sarcasm generation. As far as we know, our proposed model is the first of its kind for end-to-end neural sarcasm generation.

## 7 Conclusion and Future Work

We proposed a first of its kind approach for textual sarcasm generation from literal opinionated texts. We designed a modular framework for extracting facts from the input, generating incongruous positive and negative situational phrases related to the facts, and finally generating sarcastic variations. For evaluation, we prepared a benchmark dataset containing literal opinions and their sarcastic versions. Through qualitative and quantitative analysis of the system's performance on the benchmark dataset, we observed that our system often generates better sarcastic sentences compared to some of our trivial model variants, and unsupervised systems used for machine translation and sentiment style transfer. In the future, we would like to extend this framework for cross-lingual and cross-cultural sarcasm and irony generation.

## References


Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018. Unsupervised statistical machine translation. *arXiv preprint arXiv:1809.01272*.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2017. Unsupervised neural machine translation. *arXiv preprint arXiv:1710.11041*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv:1409.0473*.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Francesco Barbieri, Horacio Saggion, and Francesco Ronzano. 2014. Modelling sarcasm in twitter, a novel approach. *ACL 2014*, page 50.

Elisabeth Camp. 2012. Sarcasm, pretense, and the semantics/pragmatics distinction. *Noûs*, 46(4):587–634.

John D Campbell and Albert N Katz. 2012. Are there necessary conditions for inducing a sense of sarcastic irony? *Discourse Processes*, 49(6):459–480.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Semi-supervised recognition of sarcastic sentences in twitter and amazon. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 107–116. Association for Computational Linguistics.

Aniruddha Ghosh and Tony Veale. 2016. Fracking sarcasm using neural network. In *Proceedings of the 7th workshop on computational approaches to subjectivity, sentiment and social media analysis*, pages 161–169.

Roberto González-Ibánez, Smaranda Muresan, and Nina Wacholder. 2011. Identifying sarcasm in twitter: a closer look. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 581–586. Association for Computational Linguistics.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. *arXiv preprint arXiv:1603.08148*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Eduard Hovy. 1987. Generating natural language under pragmatic constraints. *Journal of Pragmatics*, 11(6):689–719.

Parag Jain, Abhijit Mishra, Amar Prakash Azad, and Karthik Sankaranarayanan. 2018. Unsupervised controllable text formalization. *arXiv preprint arXiv:1809.04556*.

Aditya Joshi, Samarth Agrawal, Pushpak Bhattacharyya, and Mark J Carman. 2017a. Expect the unexpected: Harnessing sentence completion for sarcasm detection. In *International Conference of the Pacific Association for Computational Linguistics*, pages 275–287. Springer.

Aditya Joshi, Pushpak Bhattacharyya, and Mark J Carman. 2017b. Automatic sarcasm detection: A survey. *ACM Computing Surveys (CSUR)*, 50(5):73.

Aditya Joshi, Anoop Kunchukuttan, Pushpak Bhattacharyya, and Mark James Carman. 2015a. Sarcasmbot: An open-source sarcasm-generation module for chatbots. In *WISDOM Workshop at KDD*.

Aditya Joshi, Vinita Sharma, and Pushpak Bhattacharyya. 2015b. Harnessing context incongruity for sarcasm detection. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 757–762.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. *arXiv preprint arXiv:1701.02810*.

Dimitrios Kotzias, Misha Denil, Nando De Freitas, and Padhraic Smyth. 2015. From group to individual labels using deep features. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 597–606. ACM.

Juncen Li, Robin Jia, He He, and Percy Liang. 2018. Delete, retrieve, generate: A simple approach to sentiment and style transfer. *arXiv preprint arXiv:1804.06437*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Edward Loper and Steven Bird. 2002. Nltk: the natural language toolkit. *arXiv preprint cs/0205028*.

Abhijit Mishra, Kuntal Dey, and Pushpak Bhattacharyya. 2017. Learning cognitive features from gaze data for sentiment and sarcasm classification using convolutional neural network. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 377–387.

Abhijit Mishra, Diptesh Kanojia, Seema Nagar, Kuntal Dey, and Pushpak Bhattacharyya. 2016. Harnessing cognitive features for sarcasm detection. *ACL 2016*, page 156.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL, 2002*.

Paul Piwek. 2003. A flexible pragmatics-driven language generator for animated agents. *arXiv preprint cs/0312050*.

Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert, and Ruihong Huang. 2013. Sarcasm as contrast between a positive sentiment and negative situation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 704–714.

Graeme Ritchie. 2005. Computational mechanisms for pun generation. In *Proceedings of the Tenth European Workshop on Natural Language Generation (ENLG-05)*.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in neural information processing systems*, pages 6830–6841.

Advaith Siddharthan. 2014. A survey of research on text simplification. *ITL-International Journal of Applied Linguistics*, 165(2):259–298.

Carlo Strapparava, Oliviero Stock, and Rada Mihalcea. 2011. Computational humour. In *Emotion-oriented systems*, pages 609–634. Springer.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3319–3328. JMLR. org.

Yi Tay, Luu Anh Tuan, Siu Cheung Hui, and Jian Su. 2018. Reasoning with sarcasm by reading in-between. *arXiv preprint arXiv:1805.02856*.

Byron C Wallace. 2015. Computational irony: A survey and new perspectives. *Artificial Intelligence Review*, 43(4):467–483.

Robert West and Eric Horvitz. 2019. Reverse-engineering satire, or" paper on computational humor accepted despite making serious advances". *arXiv preprint arXiv:1901.03253*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Jingjing Xu, Xu Sun, Qi Zeng, Xuancheng Ren, Xiaodong Zhang, Houfeng Wang, and Wenjie Li. 2018. Unpaired sentiment-to-sentiment translation: A cycled reinforcement learning approach. *arXiv preprint arXiv:1805.05181*.

Zhirui Zhang, Shuo Ren, Shujie Liu, Jianyong Wang, Peng Chen, Mu Li, Ming Zhou, and Enhong Chen. 2018. Style transfer as unsupervised machine translation. *arXiv preprint arXiv:1808.07894*.

# Generating Classical Chinese Poems from Vernacular Chinese

**Zhichao Yang**[1]*, **Pengshan Cai**[1]*, **Yansong Feng**[2], **Fei Li**[1],
**Weijiang Feng**[3], **Elena Suet-Ying Chiu**[1], **Hong Yu**[1]
[1] University of Massachusetts, MA, USA
{zhichaoyang, pengshancai}@umass.edu foxlf823@gmail.com
chiu@llc.umass.edu hong_yu@uml.edu
[2] Institute of Computer Science and Technology, Peking University, China
fengyansong@pku.edu.cn
[3] College of Computer, National University of Defense Technology, China
fengweijiang14@nudt.edu.cn

## Abstract

Classical Chinese poetry is a jewel in the treasure house of Chinese culture. Previous poem generation models only allow users to employ keywords to interfere the meaning of generated poems, leaving the dominion of generation to the model. In this paper, we propose a novel task of generating classical Chinese poems from vernacular, which allows users to have more control over the semantic of generated poems. We adapt the approach of unsupervised machine translation (UMT) to our task. We use segmentation-based padding and reinforcement learning to address under-translation and over-translation respectively. According to experiments, our approach significantly improve the perplexity and BLEU compared with typical UMT models. Furthermore, we explored guidelines on how to write the input vernacular to generate better poems. Human evaluation showed our approach can generate high-quality poems which are comparable to amateur poems.

## 1 Introduction

During thousands of years, millions of classical Chinese poems have been written. They contain ancient poets' emotions such as their appreciation for nature, desiring for freedom and concerns for their countries. Among various types of classical poetry, *quatrain poems* stand out. On the one hand, their aestheticism and terseness exhibit unique elegance. On the other hand, composing such poems is extremely challenging due to their phonological, tonal and structural restrictions.

Most previous models for generating classical Chinese poems (He et al., 2012; Zhang and Lapata, 2014) are based on limited keywords or characters at fixed positions (e.g., acrostic poems). Since users could only interfere with the semantic of generated poems using a few input words, models control the procedure of poem generation. In this paper, we proposed a novel model for classical Chinese poem generation. As illustrated in Figure 1, our model generates a classical Chinese poem based on a vernacular Chinese paragraph. Our objective is not only to make the model generate aesthetic and terse poems, but also keep rich semantic of the original vernacular paragraph. Therefore, our model gives users more control power over the semantic of generated poems by carefully writing the vernacular paragraph.

Although a great number of classical poems and vernacular paragraphs are easily available, there exist only limited human-annotated pairs of poems and their corresponding vernacular translations. Thus, it is unlikely to train such poem generation model using supervised approaches. Inspired by unsupervised machine translation (UMT) (Lample et al., 2018b), we treated our task as a translation problem, namely translating vernacular paragraphs to classical poems.

However, our work is not just a straight-forward application of UMT. In a training example for UMT, the length difference of source and target languages are usually not large, but this is not true in our task. Classical poems tend to be more concise and abstract, while vernacular text tends to be detailed and lengthy. Based on our observation on gold-standard annotations, vernacular paragraphs usually contain more than twice as many Chinese characters as their corresponding classical poems. Therefore, such discrepancy leads to two main problems during our preliminary experiments: (1) **Under-translation**: when summarizing vernacular paragraphs to poems, some vernacular sentences are not translated and ignored by our model. Take the last two vernacular sentences in Figure

*Equal contribution



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6155–6164,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

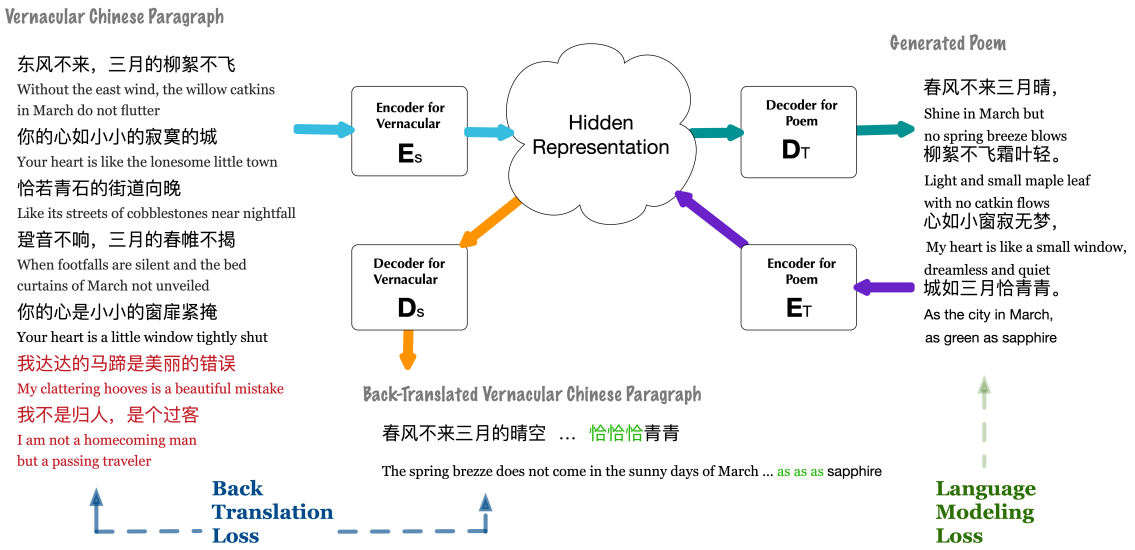


Figure 1: An example of the training procedures of our model. Here we depict two procedures, namely back translation and language modeling. Back translation has two paths, namely $\mathbf{E}_S \rightarrow \mathbf{D}_T \rightarrow \mathbf{E}_T \rightarrow \mathbf{D}_S$ and $\mathbf{D}_T \rightarrow \mathbf{E}_S \rightarrow \mathbf{D}_S \rightarrow \mathbf{E}_T$. Language modeling also has two paths, namely $\mathbf{E}_T \rightarrow \mathbf{D}_T$ and $\mathbf{E}_S \rightarrow \mathbf{D}_S$. Figure 1 shows only the former one for each training procedure.

1 as examples, they are not covered in the generated poem. (2) **Over-translation**: when expanding poems to vernacular paragraphs, certain words are unnecessarily translated for multiple times. For example, the last sentence in the generated poem of Figure 1, *as green as sapphire*, is back-translated as *as green as as as sapphire*.

Inspired by the phrase segmentation schema in classical poems (Ye, 1984), we proposed the method of phrase-segmentation-based padding to handle with under-translation. By padding poems based on the phrase segmentation custom of classical poems, our model better aligns poems with their corresponding vernacular paragraphs and meanwhile lowers the risk of under-translation. Inspired by Paulus et al. (2018), we designed a reinforcement learning policy to penalize the model if it generates vernacular paragraphs with too many repeated words. Experiments show our method can effectively decrease the possibility of over-translation.

The contributions of our work are threefold:

(1) We proposed a novel task for unsupervised Chinese poem generation from vernacular text.

(2) We proposed using phrase-segmentation-based padding and reinforcement learning to address two important problems in this task, namely under-translation and over-translation.

(3) Through extensive experiments, we proved the effectiveness of our models and explored how to write the input vernacular to inspire better poems. Human evaluation shows our models are able to generate high quality poems, which are comparable to amateur poems.

## 2 Related Works

**Classical Chinese Poem Generation** Most previous works in classical Chinese poem generation focus on improving the semantic coherence of generated poems. Based on LSTM, Zhang and Lapata (2014) purposed generating poem lines incrementally by taking into account the history of what has been generated so far. Yan (2016) proposed a polishing generation schema, each poem line is generated incrementally and iteratively by refining each line one-by-one. Wang et al. (2016) and Yi et al. (2018) proposed models to keep the generated poems coherent and semantically consistent with the user's intent. There are also researches that focus on other aspects of poem generation. (Yang et al. (2018) explored increasing the diversity of generated poems using an unsupervised approach. Xu et al. (2018) explored generating Chinese poems from images. While most previous works generate poems based on topic words, our work targets at a novel task: generating poems from vernacular Chinese paragraphs.

**Unsupervised Machine Translation** Compared

with supervised machine translation approaches (Cho et al., 2014; Bahdanau et al., 2015), unsupervised machine translation (Lample et al., 2018a,b) does not rely on human-labeled parallel corpora for training. This technique is proved to greatly improve the performance of low-resource languages translation systems. (e.g. English-Urdu translation). The unsupervised machine translation framework is also applied to various other tasks, e.g. image captioning (Feng et al., 2019), text style transfer (Zhang et al., 2018), speech to text translation (Bansal et al., 2017) and clinical text simplification (Weng et al., 2019). The UMT framework makes it possible to apply neural models to tasks where limited human labeled data is available. However, in previous tasks that adopt the UMT framework, the abstraction levels of source and target language are the same. This is not the case for our task.

**Under-Translation & Over-Translation** Both are troublesome problems for neural sequence-to-sequence models. Most previous related researches adopt the coverage mechanism (Tu et al., 2016; Mi et al., 2016; Sankaran et al., 2016). However, as far as we know, there were no successful attempt applying coverage mechanism to transformer-based models (Vaswani et al., 2017).

## 3 Model

### 3.1 Main Architecture

We transform our poem generation task as an unsupervised machine translation problem. As illustrated in Figure 1, based on the recently proposed UMT framework (Lample et al., 2018b), our model is composed of the following components:

- Encoder $\mathbf{E}_s$ and decoder $\mathbf{D}_s$ for vernacular paragraph processing
- Encoder $\mathbf{E}_t$ and decoder $\mathbf{D}_t$ for classical poem processing

where $\mathbf{E}_s$ (or $\mathbf{E}_t$) takes in a vernacular paragraph (or a classical poem) and converts it into a hidden representation, and $\mathbf{D}_s$ (or $\mathbf{D}_t$) takes in the hidden representation and converts it into a vernacular paragraph (or a poem). Our model relies on a vernacular texts corpus $\boldsymbol{S}$ and a poem corpus $\boldsymbol{T}$. We denote $S$ and $T$ as instances in $\boldsymbol{S}$ and $\boldsymbol{T}$ respectively.

The training of our model relies on three procedures, namely *parameter initialization*, *language modeling* and *back-translation*. We will give detailed introduction to each procedure.

**Parameter initialization** As both vernacular and classical poem use Chinese characters, we initialize the character embedding of both languages in one common space, the same character in two languages shares the same embedding. This initialization helps associate characters with their plausible translations in the other language.

**Language modeling** It helps the model generate texts that conform to a certain language. A well-trained language model is able to detect and correct minor lexical and syntactic errors. We train the language models for both vernacular and classical poem by minimizing the following loss:

$$\begin{aligned}\mathcal{L}^{lm} = &\mathop{\mathbb{E}}_{S\in\boldsymbol{S}}[-\log P(S|\mathbf{D}_s(\mathbf{E}_s(S_N))]+\\ &\mathop{\mathbb{E}}_{T\in\boldsymbol{T}}[-\log P(T|\mathbf{D}_t(\mathbf{E}_t(T_N))],\end{aligned} \tag{1}$$

where $S_N$ (or $T_N$) is generated by adding noise (drop, swap or blank a few words) in $S$ (or $T$).

**Back-translation** Based on a vernacular paragraph $S$, we generate a poem $T_S$ using $\mathbf{E}_s$ and $\mathbf{D}_t$, we then translate $T_S$ back into a vernacular paragraph $S_{T_S} = \mathbf{D}_s(\mathbf{E}_t(T_S))$. Here, $S$ could be used as gold standard for the back-translated paragraph $S_{T_s}$. In this way, we could turn the unsupervised translation into a supervised task by maximizing the similarity between $S$ and $S_{T_S}$. The same also applies to using poem $T$ as gold standard for its corresponding back-translation $T_{S_T}$. We define the following loss:

$$\begin{aligned}\mathcal{L}^{bt} = &\mathop{\mathbb{E}}_{S\in\boldsymbol{S}}[-\log P(S|\mathbf{D}_s(\mathbf{E}_t(T_S))]+\\ &\mathop{\mathbb{E}}_{T\in\boldsymbol{T}}[-\log P(T|\mathbf{D}_t(\mathbf{E}_s(S_T))].\end{aligned} \tag{2}$$

Note that $\mathcal{L}^{bt}$ does not back propagate through the generation of $T_S$ and $S_T$ as we observe no improvement in doing so. When training the model, we minimize the composite loss:

$$\mathcal{L} = \alpha_1\mathcal{L}^{lm} + \alpha_2\mathcal{L}^{bt}, \tag{3}$$

where $\alpha_1$ and $\alpha_2$ are scaling factors.

### 3.2 Addressing Under-Translation and Over-Translation

During our early experiments, we realize that the naive UMT framework is not readily applied to our task. Classical Chinese poems are featured for

its terseness and abstractness. They usually focus on depicting broad poetic images rather than details. We collected a dataset of classical Chinese poems and their corresponding vernacular translations, the average length of the poems is 32.0 characters, while for vernacular translations, it is 73.3. The huge gap in sequence length between source and target language would induce over-translation and under-translation when training UMT models. In the following sections, we explain the two problems and introduce our improvements.

### 3.2.1 Under-Translation

By nature, classical poems are more concise and abstract while vernaculars are more detailed and lengthy, to express the same meaning, a vernacular paragraph usually contains more characters than a classical poem. As a result, when summarizing a vernacular paragraph $S$ to a poem $T_S$, $T_S$ may not cover all information in $S$ due to its length limit. In real practice, we notice the generated poems usually only cover the information in the front part of the vernacular paragraph, while the latter part is unmentioned.

To alleviate under-translation, we propose phrase segmentation-based padding. Specifically, we first segment each line in a classical poem into several sub-sequences, we then join these sub-sequences with the special padding tokens <p>. During training, the padded lines are used instead of the original poem lines. As illustrated in Figure 2, padding would create better alignments between a vernacular paragraph and a prolonged poem, making it more likely for the latter part of the vernacular paragraph to be covered in the poem. As we mentioned before, the length of the vernacular translation is about twice the length of its corresponding classical poem, so we pad each segmented line to twice its original length.

According to Ye (1984), to present a stronger sense of rhythm, each type of poem has its unique phrase segmentation schema, for example, most seven-character quatrain poems adopt the 2-2-3 schema, i.e. each quatrain line contains 3 phrases, the first, second and third phrase contains 2, 2, 3 characters respectively. Inspired by this law, we segment lines in a poem according to the corresponding phrase segmentation schema. In this way, we could avoid characters within the scope of a phrase to be cut apart, thus best preserve the semantic of each phrase.(Chang et al., 2008)




Figure 2: A real example to show the effectiveness of our phrase-segmentation-based padding. Without padding, the vernacular paragraph could not be aligned well with the poem. Therefore, the text *in South Yangtze ends but the grass and trees have not withered* in red is not covered in the poem. By contrast, they are covered well after using our padding method.

### 3.2.2 Over-Translation

In NMT, when decoding is complete, the decoder would generate an <EOS>token, indicating it has reached the end of the output sequence. However, when expending a poem $T$ into a vernacular Chinese paragraph $S_T$, due to the conciseness nature of poems, after finishing translating every source character in $T$, the output sequence $S_T$ may still be much shorter than the expected length of a poem's vernacular translation. As a result, the decoder would believe it has not finished decoding. Instead of generating the <EOS>token, the decoder would continue to generate new output characters from previously translated source characters. This would cause the decoder to repetitively output a piece of text many times.

To remedy this issue, in addition to minimizing the original loss function $\mathcal{L}$, we propose to minimize a specific discrete metric, which is made possible with reinforcement learning.

We define *repetition ratio* $RR(S)$ of a paragraph $S$ as:

$$RR(S) = 1 - \frac{vocab(S)}{len(S)}, \tag{4}$$

where $vocab(S)$ refers to the number of distinctive characters in $S$, $len(S)$ refers the number of all characters in $S$. Obviously, if a generated sequence contains many repeated characters, it would have high *repetition ratio*. Following the self-critical policy gradient training (Rennie et al.,

| | Training set | Validation set | Test set |
|---|---|---|---|
| # Poems | 163K | 19K | 487 |
| Average length of poems | 32.0 | 32.0 | 32.0 |
| # vernacular paragraphs | 337K | 19K | 487 |
| Average length of vernacular paragraphs | 71.8 | 76.8 | 73.3 |

Table 1: Statistics of our dataset

2017), we define the following loss function:

$$\mathcal{L}^{rl} = \mathop{\mathbb{E}}_{S \in \boldsymbol{S}}[(RR(S_{T_S}) - \tau) \log P(S|\mathbf{D}_s(\mathbf{E}_t(T_S))], \tag{5}$$

where $\tau$ is a manually set threshold. Intuitively, minimizing $\mathcal{L}^{rl}$ is equivalent to maximizing the conditional likelihood of the sequence $S$ given $S_{T_S}$ if its *repetition ratio* is lower than the threshold $\tau$. Following (Wu et al., 2016), we revise the composite loss as:

$$\mathcal{L}' = \alpha_1 \mathcal{L}^{lm} + \alpha_2 \mathcal{L}^{bt} + \alpha_3 \mathcal{L}^{rl}, \tag{6}$$

where $\alpha_1, \alpha_2, \alpha_3$ are scaling factors.

## 4 Experiment

The objectives of our experiment are to explore the following questions: (1) How much do our models improve the generated poems? (Section 4.4) (2) What are characteristics of the input vernacular paragraph that lead to a good generated poem? (Section 4.5) (3) What are weaknesses of generated poems compared to human poems? (Section 4.6) To this end, we built a dataset as described in Section 4.1. Evaluation metrics and baselines are described in Section 4.2 and 4.3. For the implementation details of building the dataset and models, please refer to supplementary materials.[1]

### 4.1 Datasets

**Training and Validation Sets** We collected a corpus of poems and a corpus of vernacular literature from online resources. The poem corpus contains 163K quatrain poems from *Tang Poems* and *Song Poems*, the vernacular literature corpus contains 337K short paragraphs from 281 famous books, the corpus covers various literary forms including prose, fiction and essay. Note that our poem corpus and a vernacular corpus are not aligned. We further split the two corpora into a training set and a validation set.

[1]Our data and code is publicly available at https://github.com/whaleloops/interpoetry

**Test Set** From online resources, we collected 487 seven-character quatrain poems from *Tang Poems* and *Song Poems*, as well as their corresponding high quality vernacular translations. These poems could be used as gold standards for poems generated from their corresponding vernacular translations. Table 1 shows the statistics of our training, validation and test set.

### 4.2 Evaluation Metrics

**Perplexity** Perplexity reflects the probability a model generates a certain poem. Intuitively, a better model would yield higher probability (lower perplexity) on the gold poem.

**BLEU** As a standard evaluation metric for machine translation, BLEU (Papineni et al., 2001) measures the intersection of n-grams between the generated poem and the gold poem. A better generated poem usually achieves higher BLEU score, as it shares more n-gram with the gold poem.

**Human evaluation** While perplexity and BLEU are objective metrics that could be applied to large-volume test set, evaluating Chinese poems is after all a subjective task. We invited 30 human evaluators to join our human evaluation. The human evaluators were divided into two groups. The expert group contains 15 people who hold a bachelor degree in Chinese literature, and the amateur group contains 15 people who holds a bachelor degree in other fields. All 30 human evaluators are native Chinese speakers.

We ask evaluators to grade each generated poem from four perspectives: 1) *Fluency*: Is the generated poem grammatically and rhythmically well formed, 2) *Semantic coherence*: Is the generated poem itself semantic coherent and meaningful, 3) *Semantic preservability*: Does the generated poem preserve the semantic of the modern Chinese translation, 4) *Poeticness*: Does the generated poem display the characteristic of a poem and does the poem build good poetic image. The grading scale for each perspective is from 1 to 5.

| ID | Vernacular paragraph | Generated poem |
|---|---|---|
| 1 | 青山隐隐约约绿水千里迢迢，秋时已尽江南草木还未枯凋。二十四桥明月映照幽幽清夜，你这美人现在何处教人吹箫？ | 青山隐隐绿水光，千里秋时已尽藏。江南草木还未枯，二十四桥幽夜香。 |
| 1 | Blue peak is blur and faint, green river meaders thousands miles, the southern grass has not dried up in the autumn. Twenty-four hole bridges and bright moons shine in the clear night. where do you beautifully teach people to flute? | Blue peak blurs and green river shines, thousands miles away, autumn has been hidden. the southern grass has not dried up in the autumn. Twenty-four hole bridges smells beautiful and quiet tonight. |
| 2 | 拂袖起舞于梦中徘徊，相思蔓上心扉。她眷恋梨花泪，静画红妆等谁归，空留伊人徐徐憔悴。 | 拂袖起舞梦徘徊，相思蔓上心扉开。玉眷梨花泪痕静，画红等谁归去来。 |
| 2 | The sleeves danced in the dream, and the lovesickness was on the heart. She is in love with the tears of pears, and who is quietly wearing red makeup, only left alone to be languished slowly. | The sleeves danced in the dream, the lovesickness appeared in the heart. Jade concerns tears of pears but the mark is still, wearing red makeup waiting for the one to come and go. |
| 3 | 窗外的麻雀在电线杆上多嘴，你说这一句很有夏天的感觉。手中的铅笔在纸上来来回回，我用几行字形容你是我的谁。 | 窗下麻姑灯火多，闲中说与万缘何。夏频手把铅华纸，来往回头用几多。 |
| 3 | The sparrow outside the window is talking on the pole. You say this sentence makes you feel very summer. The pencil in my hand is writing back and forth on the paper. I only use a few lines to describe who you are to me. | Under the window lie sparrow girls in this prosperous city, chit chatting about the destiny of the world. Summer hands over many drawing canvas, Looking back and forth, how many do you need? |
| 4 | 雨天的屋瓦，浮漾湿湿的流光，灰而温柔，迎光则微明，背光则幽黯，对于视觉，是一种低沉的安慰。 | 雨余屋瓦浮漾湿，流光灰色暖相迎。光则微明背则色，幽人黯黯对风清。 |
| 4 | The rainy days of the roof tiles are soaking wet and wet, gray and gentle, Facing the light, it is slightly bright, Against the light, it is pitch dark, For the concept of vision, it is a deep comfort. | The excess rain makes roof tiles rippling, ambilight gray is warm and welcoming. Light is slightly bright, against is pure color. The person hides in dark but faces wind breeze. |
| 5 | 只要历史不阻断，时间不倒退，一切都会衰老。老就老了吧，安详地交给世界一副慈祥美。假饰天真是最残酷的自我糟践。 | 只要谐公不阻时，不倒退食一尘埃。会衰老矣安分世，一副慈祥假此来。 |
| 5 | As long as history does not block, time does not go backwards, everything will age. It is fine to get old, and handing it to the world with kindness. Faking innocence is the cruelest self-destruction. | As long as people do not block time, it will not go backwards and absorbs into a dust. People should stay chill and get old. Faking innocence is not the way to go. |

Table 2: A few poems generated by our model from their corresponding vernacular paragraphs.

### 4.3 Baselines

We compare the performance of the following models: (1) *LSTM* (Hochreiter and Schmidhuber, 1997); (2)*Naive transformer* (Vaswani et al., 2017); (3)*Transformer + Anti OT* (RL loss); (4)*Transformer + Anti UT* (phrase segmentation-based padding); (5)*Transformer + Anti OT&UT*.

### 4.4 Reborn Poems: Generating Poems from Vernacular Translations

As illustrated in Table 2 (ID 1). Given the vernacular translation of each gold poem in test set, we generate five poems using our models. Intuitively, the more the generated poem resembles the gold poem, the better the model is. We report mean perplexity and BLEU scores in Table 3 (Where +Anti OT refers to adding the reinforcement loss to mitigate over-fitting and +Anti UT refers to adding phrase segmentation-based padding to mitigate under-translation), human evaluation results in Table 4.[2]

According to experiment results, perplexity, BLEU scores and total scores in human evaluation are consistent with each other. We observe all BLEU scores are fairly low, we believe it is reasonable as there could be multiple ways to compose a poem given a vernacular paragraph. Among transformer-based models, both +Anti OT and +Anti UT outperforms the naive transformer, while Anti OT&UT shows the best performance, this demonstrates alleviating under-translation and over-translation both helps generate better poems. Specifically, +Anti UT shows bigger improvement than +Anti OT. According to human evaluation, among the four perspectives, our Anti OT&UT brought most score improvement in *Semantic preservability*, this proves our improvement on semantic preservability was most obvious to human evaluators. All transformer-

[2]We did not use LSTM in human evaluation since its performance is worse as shown in Table 3.

| Model | Perplexity | BLEU | BLEU-1 | BLEU-2 | BLEU-3 | BLEU-4 |
|---|---|---|---|---|---|---|
| LSTM | 118.27 | 3.81 | 39.16 | 6.93 | 1.58 | 0.49 |
| Transformer | 105.79 | 5.50 | 40.92 | 8.02 | 2.46 | 1.11 |
| +Anti OT | 77.33 | 6.08 | 41.22 | 8.72 | 2.82 | 1.36 |
| +Anti UT | 74.21 | 6.34 | 42.20 | **9.04** | **2.96** | 1.44 |
| +Anti OT&UT | **65.58** | **6.57** | **42.53** | 8.98 | **2.96** | **1.46** |

Table 3: Perplexity and BLEU scores of generating poems from vernacular translations. Since perplexity and BLEU scores on the test set fluctuates from epoch to epoch, we report the mean perplexity and BLEU scores over 5 consecutive epochs after convergence.

| Model | Fluency | Semantic coherence | Semantic preservability | Poeticness | Total |
|---|---|---|---|---|---|
| Transformer | 2.63 | 2.54 | 2.12 | 2.46 | 9.75 |
| +Anti OT | 2.80 | 2.75 | 2.44 | 2.71 | 10.70 |
| +Anti UT | 2.82 | 2.82 | 2.86 | 2.85 | 11.35 |
| +Anti OT&UT | **3.21** | **3.27** | **3.27** | **3.28** | **13.13** |

Table 4: Human evaluation results of generating poems from vernacular translations. We report the mean scores for each evaluation metric and total scores of four metrics.

based models outperform LSTM. Note that the average length of the vernacular translation is over 70 characters, comparing with transformer-based models, LSTM may only keep the information in the beginning and end of the vernacular. We anticipated some score inconsistency between expert group and amateur group. However, after analyzing human evaluation results, we did not observed big divergence between two groups.

### 4.5 Interpoetry: Generating Poems from Various Literature Forms

Chinese literature is not only featured for classical poems, but also various other literature forms. *Song lyric*(宋词), or *ci* also gained tremendous popularity in its palmy days, standing out in classical Chinese literature. *Modern prose*, *modern poems* and *pop song lyrics* have won extensive praise among Chinese people in modern days. The goal of this experiment is to transfer texts of other literature forms into quatrain poems. We expect the generated poems to not only keep the semantic of the original text, but also demonstrate terseness, rhythm and other characteristics of ancient poems. Specifically, we chose 20 famous fragments from four types of Chinese literature (5 fragments for each of modern prose, modern poems, pop song lyrics and Song lyrics). We try to As no ground truth is available, we resorted to human evaluation with the same grading standard in Section 4.4.

Comparing the scores of different literature forms, we observe Song lyric achieves higher scores than the other three forms of modern literature. It is not surprising as both Song lyric and quatrain poems are written in classical Chinese, while the other three literature forms are all in vernacular.

Comparing the scores within the same literature form, we observe the scores of poems generated from different paragraphs tends to vary. After carefully studying the generated poems as well as their scores, we have the following observation:

1) In classical Chinese poems, poetic images (意象) were widely used to express emotions and to build artistic conception. A certain poetic image usually has some fixed implications. For example, *autumn* is usually used to imply sadness and loneliness. However, with the change of time, poetic images and their implications have also changed. According to our observation, if a vernacular paragraph contains more poetic images used in classical literature, its generated poem usually achieves higher score. As illustrated in Table 2, both paragraph 2 and 3 are generated from pop song lyrics, paragraph 2 uses many poetic images from classical literature (e.g. pear flowers, makeup), while paragraph 3 uses modern poetic images (e.g. sparrows on the utility pole). Obviously, compared with poem 2, sentences in poem 3 seems more confusing, as the poetic images in modern times may not fit well into the language model of classical poems.

2) We also observed that poems generated from descriptive paragraphs achieve higher scores than from logical or philosophical paragraphs. For example, in Table 2, both paragraph 4 (more descriptive) and paragraph 5 (more philosophical) were

| Literature form | Fluency | Semantic coherence | Semantic preservability | Poeticness | Total |
|---|---|---|---|---|---|
| Prose | 2.52 | 2.30 | 2.30 | 2.32 | 9.44 |
| Modern poem | 2.37 | 2.34 | 2.01 | 2.16 | 8.88 |
| Pop song lyric | 2.40 | 2.31 | 2.24 | 2.42 | 9.37 |
| Song lyric | **2.62** | **2.54** | **2.26** | **2.49** | **9.91** |

Table 5: Human evaluation results for generating poems from various literature forms. We show the results obtained from our best model (Transformer+Anti OT&UT).

selected from famous modern prose. However, compared with poem 4, poem 5 seems semantically more confusing. We offer two explanations to the above phenomenon: **i**. Limited by the 28-character restriction, it is hard for quatrain poems to cover complex logical or philosophical explanation. **ii**. As vernacular paragraphs are more detailed and lengthy, some information in a vernacular paragraph may be lost when it is summarized into a classical poem. While losing some information may not change the general meaning of a descriptive paragraph, it could make a big difference in a logical or philosophical paragraph.

### 4.6 Human Discrimination Test

We manually select 25 generated poems from vernacular Chinese translations and pair each one with its corresponding human written poem. We then present the 25 pairs to human evaluators and ask them to differentiate which poem is generated by human poet.[3]

As demonstrated in Table 6, although the general meanings in human poems and generated poems seem to be the same, the wordings they employ are quite different. This explains the low BLEU scores in Section 4.3. According to the test results in Table 7, human evaluators only achieved 65.8% in mean accuracy. This indicates the best generated poems are somewhat comparable to poems written by amateur poets.

We interviewed evaluators who achieved higher than 80% accuracy on their differentiation strategies. Most interviewed evaluators state they realize the sentences in a human written poem are usually well organized to highlight a theme or to build a poetic image, while the correlation between sentences in a generated poem does not seem strong. As demonstrated in Table 6, the last two sentences in both human poems (marked as red) echo each other well, while the sentences in machine-generated poems seem more independent. This gives us hints on the weakness of generated poems: While neural models may generate poems that resemble human poems lexically and syntactically, it's still hard for them to compete with human beings in building up good structures.

[3]We did not require the expert group's participation as many of them have known the gold poems already. Thus using their judgments would be unfair.

| | |
|---|---|
| Human | 黄沙磧里客行迷，四望云天直下低。<br>Within yellow sand moraine guest travels lost,<br>looking around found sky and clouds low.<br>为言地尽天还尽，行到安西更向西。<br>It is said that earth and sky ends here,<br>however I need to travel more west than anxi. |
| Machine | 异乡客子黄沙迷，雁路迷寒云向低。<br>Guest in yellow sand gets lost,<br>geese are lost because clouds gets low.<br>只道山川到此尽，安西还要更向西。<br>It is said that mountains end here,<br>however anxi is even more west. |
| Human | 绝域从军计惘然，东南幽恨满词笺。<br>It's hard to pay for the ambition of the military field,<br>the anxiety of situation in southeast is all over poems.<br>一箫一剑平生意，负尽狂名十五年。<br>A flute and sword is all I care about in my life,<br>15 years have failed the reputation of "madman". |
| Machine | 从军疆场志难酬，令人怅望东南州。<br>It's hard to fulfill my ambition on the military field,<br>the situation in the southeast states are troublesome.<br>形容仗剑敌平戎，情怀注满赋雪愁。<br>I would like to use my sword to conquer my enemy,<br>yet my feelings are full of worry like the snow. |

Table 6: Examples of generated poems and their corresponding gold poems used in human discrimination test.

## 5 Discussion

**Addressing Under-Translation** In this part, we wish to explore the effect of different phrase segmentation schemas on our phrase segmentation-based padding. According to Ye (1984), most seven-character quatrain poems adopt the 2-2-3 segmentation schema. As shown in examples in Figure 3, we compare our phrase segmentation-based padding (2-2-3 schema) to two less common schemas (i.e. 2-3-2 and 3-2-2 schema) we report our experiment results in Table 8.

| Accuracy | Value |
|---|---|
| Min | 52.0 |
| Max | 84.0 |
| Mean | 65.8 |

Table 7: The performance of human discrimination test.

| Padding schema | Perplexity | BLEU |
|---|---|---|
| 2-2-3 | **74.21** | **6.34** |
| 2-3-2 | 83.12 | 5.49 |
| 3-2-2 | 85.66 | 5.75 |

Table 8: Perplexity and BLEU scores of different padding schemas.

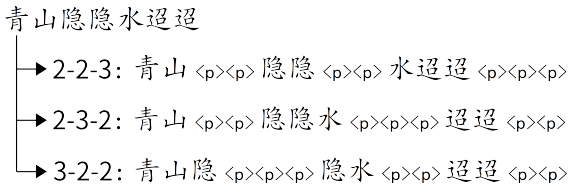


Figure 3: Examples of different padding schemas.

The results show our 2-2-3 segmentation-schema greatly outperforms 2-3-2 and 3-2-2 schema in both perplexity and BLEU scores. Note that the BLEU scores of 2-3-2 and 3-2-2 schema remains almost the same as our naive baseline (Without padding). According to the observation, we have the following conclusions: 1) Although padding better aligns the vernacular paragraph to the poem, it may not improve the quality of the generated poem. 2) The padding tokens should be placed according to the phrase segmentation schema of the poem as it preserves the semantic within the scope of each phrase.

**Addressing Over-Translation** To explore the effect of our reinforcement learning policy on alleviating over-translation, we calculate the *repetition ratio* of vernacular paragraphs generated from classical poems in our validation set. We found *naive transformer* achieves $40.8\%$ in repetition ratio, while our +Anti OT achieves $34.9\%$. Given the repetition ratio of vernacular paragraphs (written by human beings) in our validation set is $30.1\%$, the experiment results demonstrated our RL loss effectively alleviate over-translation, which in turn leads to better generated poems.

## 6 Conclusion

In this paper, we proposed a novel task of generating classical Chinese poems from vernacular paragraphs. We adapted the unsupervised machine translation model to our task and meanwhile proposed two novel approaches to address the under-translation and over-translation problems. Experiments show that our task can give users more controllability in generating poems. In addition, our approaches are very effective to solve the problems when the UMT model is directly used in this task. In the future, we plan to explore: (1) Applying the UMT model in the tasks where the abstraction levels of source and target languages are different (e.g., unsupervised automatic summarization); (2) Improving the quality of generated poems via better structure organization approaches.

## References


Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Sameer Bansal, Herman Kamper, Adam Lopez, and Sharon Goldwater. 2017. Towards speech-to-text translation without speech recognition. In *EACL*.

Pi-Chuan Chang, Michel Galley, and Christopher D. Manning. 2008. Optimizing chinese word segmentation for machine translation performance. In *WMT@ACL*.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*.

Yang Feng, Lin Ma, Wei Liu, and Jiebo Luo. 2019. Unsupervised image captioning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Jing He, Ming Zhou, and Long Jiang. 2012. Generating chinese classical poems with statistical machine translation models. In *AAAI*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9:1735–1780.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018a. Unsupervised machine translation using monolingual corpora only. In *ICLR*.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018b. Phrase-based neural unsupervised machine translation. In *EMNLP*.

Haitao Mi, Baskaran Sankaran, Zhiguo Wang, and Abe Ittycheriah. 2016. Coverage embedding models for neural machine translation. In *EMNLP*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *ICLR*.

Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1179–1195.

Baskaran Sankaran, Haitao Mi, Yaser Al-Onaizan, and Abe Ittycheriah. 2016. Temporal attention model for neural machine translation. *CoRR*, abs/1608.02927.

Zhaopeng Tu, Zhengdong Lu, Yang P. Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *ACL*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Zhe Wang, Wei He, Hua Wu, Haiyang Wu, Wei Li, Haifeng Wang, and Enhong Chen. 2016. Chinese poetry generation with planning based neural network. In *COLING*.

Wei-Hung Weng, Yu-An Chung, and Peter Szolovits. 2019. Unsupervised clinical language translation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pages 3121–3131.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Gregory S. Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Linli Xu, Liang Jiang, Chuan Qin, Zhe Wang, and Dongfang Du. 2018. How images inspire poems: Generating classical chinese poetry from images with memory networks. In *AAAI*.

Rui Yan. 2016. i, poet: Automatic poetry composition through recurrent neural networks with iterative polishing schema. In *IJCAI*.

Cheng Yang, Maosong Sun, Xiaoyuan Yi, and Wenhao Li. 2018. Stylistic chinese poetry generation via unsupervised style disentanglement. In *EMNLP*.

Jiaying Ye. 1984. *Poem Criticism with Jialin*. Zhong hua shu ju, Beijing, China.

Xiaoyuan Yi, Maosong Sun, Ruoyu Li, and Zonghan Yang. 2018. Chinese poetry generation with a working memory model. In *IJCAI*.

Xingxing Zhang and Mirella Lapata. 2014. Chinese poetry generation with recurrent neural networks. In *EMNLP*.

Zhirui Zhang, Shuo Ren, Shujie Liu, Jianyong Wang, Peng Chen, Mu Li, Ming Zhou, and Enhong Chen. 2018. Style transfer as unsupervised machine translation. *CoRR*, abs/1808.07894.

# Set to Ordered Text: Generating Discharge Instructions from Medical Billing Codes

**Litton J Kurisinkel, Nancy F. Chen**
Institute for Infocomm Research, A*STAR
{Litton_Kurisinkel, nfychen}@i2r.a-star.edu.sg

## Abstract

We present set to ordered text, a natural language generation task applied to automatically generating discharge instructions from admission ICD (International Classification of Diseases) codes. This task differs from other natural language generation tasks in the following ways: (1) The input is a set of identifiable entities (ICD codes) where the relations between individual entities are not explicitly specified. (2) The output text is not a narrative description (e.g. news articles) composed from the input. Rather, inferences are made from the input (ICD codes, which represent diagnoses and clinical procedures) to generate the output (instructions). (3) There is an optimal order in which each sentence (instruction) should appear in the output. Unlike most other tasks, neither the input (ICD codes) nor their corresponding text representations of diagnoses and clinical procedures appear in the output, so the ordering of the output instructions needs to be learned in an unsupervised fashion. We hypothesize that each instruction in the output is mapped to a subset of ICD codes specified in the input. We propose a neural architecture that jointly models (a) subset selection: choosing relevant subsets from a set of input entities; (b) content ordering: learning the order of instructions; (c) text generation: representing the instructions corresponding to the selected subsets in natural language. In addition, we penalize redundancy during beam search to improve tractability for long text generation. We formulate the problem setup and conducted experiments using the MIMIC-III dataset. Our model outperforms baseline models in both BLEU scores and human evaluations.

## 1 Introduction

### 1.1 Problem Statement

Many healthcare applications exhibit a strong mapping between numerical or categorical information and clinical instructions. To develop language generation capabilities for these settings, we define a task where discharge instructions are automatically generated using admission ICD (International Classification of Diseases) codes[1] to potentially streamline clinical workflow. We define a task where the input is a set of identifiable items (ICD codes) and the output consists of ordered text sequences (instructions), which are inferred from the input (see Figure 1).

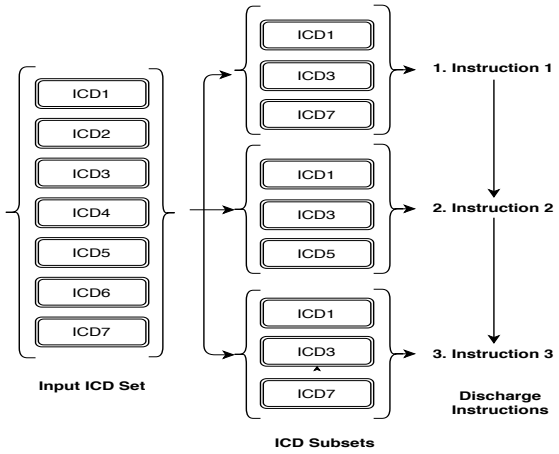


Figure 1: Set to Ordered Text: Problem Definition

### 1.2 Proposed Approach

We hypothesize that each discharge instruction in the output is mapped to a subset of ICD codes specified in the input. Our proposed approach thus models the correlations between individual entities in the input set to choose the most relevant subsets, and learn to generate their corresponding textual outputs in the appropriate order. We also incorporate explicit means for reducing redundancy during decoding. We empirically verify

[1]It is standard practice for healthcare providers to assign ICD codes to represent information regarding diagnosis and/or procedures for each hospital visit for medical billing purposes.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6165–6175,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

the proposed approach by generating discharge instructions from ICD codes assigned during hospital admissions.

### 1.3 Relation to Other Work

For most natural language generation tasks, the relations between the input entities are specified in one way or another. For text-to-text generation (e.g. news articles), the relation between entities are semantically encoded in sequences of words (Paulus et al., 2017). For graph-to-text generation, relations between the nodes in a graph are characterized through labeled or unlabeled edges (Liu et al., 2019b). For text generation with database inputs, relations are usually specified through the attributes of each data entry (Lebret et al., 2016). In our problem setup, there is no explicit characterization of the relations between the input ICD codes assigned to a patient's visit.

Most natural language generation problems focus on generating descriptions, which often include the entities specified in the input. Examples include summarization or text generation from database records (Cheng and Lapata, 2016; Jhamtani et al., 2018). In our case, both the content and the ordering of the output need to be inferred from the input data.

Our work is closest to the line of research on text expansion, where the generated text is conditioned on a set of entities (Clark et al., 2018; Zhao et al., 2018; Kiddon et al., 2016). Although in this line of work the relations between the entity input set are not specified, as in our case, these input entities often appear in the output text, making it more straightforward to model the order of the input entities appearing in the output. In our case, neither the input entity set (ICD codes) nor their corresponding text representations (diagnoses and clinical procedures) occur in the generated output (instructions).

## 2 Approach

### 2.1 Task Definition

For input set $S$, the generation task can be specified as follows,

$$x \leftarrow F(S, X)$$
$$o \leftarrow Gen(x),$$

where $F$ represents the mechanism to choose the next subset $x$ given input $S$ and the already chosen sequence of subsets $X$; $Gen$ generates the output sentence $o$ from the chosen subset $x$.

### 2.2 Neural Architecture

The proposed neural architecture is shown in Figure 2. The major components of the network are a lookup table for ICD codes, gates for content and subset selection, one RNN for content ordering and another one for decoding each discharge instruction. The network also posses two attention layers for finding the correlations between input ICD codes and for attending to the chosen content at each stage of instruction generation. The instruction ordering RNN is initialized with a zero vector. Below we elaborate on the major components of the proposed neural architecture in more detail. Bold symbols in equations represent parameter matrices.

#### 2.2.1 Content and Subset Selection

The content selection gate selects the relevant content from each ICD code during each instruction generation phase. The gate takes the ICD code embedding, the previous state of the content ordering RNN $H_{t-1}$ and the correlations among ICD codes into account for selecting the content. The content correlation vector $C_j$ for each ICD code embedding $icd_j$ in the input is computed as follows:

$$\alpha_{j,k} = \exp(icd_j^T \mathbf{W_a} icd_k)$$
$$C_j = \sum_{k \neq j} \alpha_{j,k} icd_k$$

The content gate value computation and the subsequent content selection is conducted as follows:

$$gc_j = \text{sigmoid}(\mathbf{W_{gc}}[icd_j; C_j; H_{t-1}])$$
$$icd_j^{'} = gc_j \odot icd_j$$

The selected content from each of the input ICD code passes through a subset selection gate. Each subset is selected as a probability distribution over the input set of ICD codes as follows:

$$gs_j = \text{sigmoid}(\mathbf{W_{gs}}[icd_j; C_j; H_{t-1}])$$
$$gs_j = \text{softmax}(gs_j)\ \forall j$$

$gs_j, \forall j$ represents the distribution of ICD codes in the subset chosen at the current time step of content ordering RNN. The subset selection gate updates the output from content selection gate as follows:

$$icd_j^{''} = gs_j icd_j^{'} \quad (1)$$

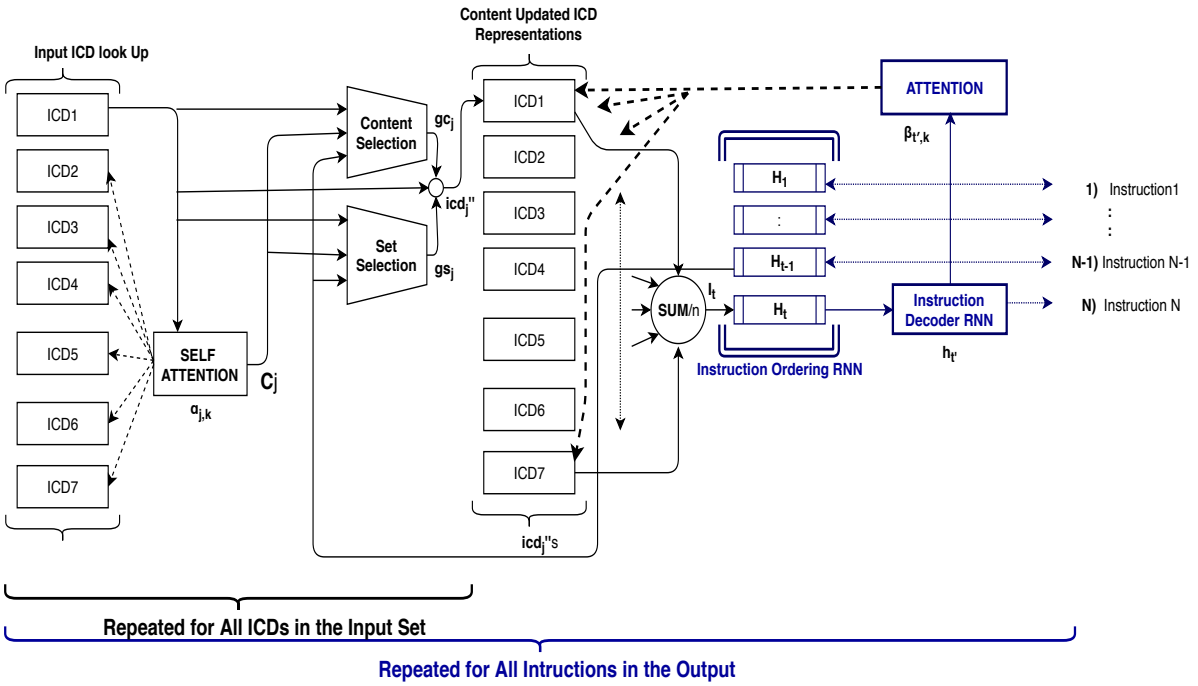


Figure 2: Proposed Neural Architecture

In the Figure 2, trapezoids represent the content and subset selection gates. During each stage of content and subset selection, the gates receive information regarding content and order of already selected subsets from the instruction ordering RNN. The figure also depicts the self attention layer which computes $C_j$.

### 2.2.2 Content Ordering and Instruction Generation

We use a GRU recurrent neural network (Chung et al., 2014) for content ordering. The RNN is initialized with a zero vector before the network activity begins and it takes the selected content of ICD codes as input during each time step. At time-step $t$,

$$H_t = \mathbf{GRU}(I_t, H_{t-1}), \tag{2}$$

where $I_t$ is the mean vector of $icd_j^{"}$s computed using the Equation 1. The instruction decoder RNN (also a GRU) is initialized with $H_t$ to generate the instruction at the current time step of content ordering RNN. During each time step of decoding, the decoder RNN attends over the current set of $icd_j^{"}$ to generate the sequence of words $w_{t'}$ in the instruction as formulated below.

$$\beta_{t',k} = \exp(h_{t'}^T \mathbf{W_c} icd_k^{"})$$
$$c_{t'} = \sum \beta_{t',k} icd_k^{"}$$
$$h_{t'}^{'} = \tanh(\mathbf{W_h}[h_{t'}, c_{t'}])$$

where $h_{t'}$ is the hidden state of the decoder at time-step $t^{'}$. The probability distribution over the output vocabulary for generating the word $w_{t'}$ at time step $t^{'}$ of the instruction generating decoder is computed as below,

$$P(w_{t'}|w_{<t'}, \{icd_i^{"}, .., icd_n^{"}\}) = \text{softmax}(\mathbf{W_o} h_{t'}^{'} + b_o) \tag{3}$$

The portion of Figure 2 marked in blue, represents the set of computations detailed in the current section.

### 2.2.3 Beam Search with Redundancy Penalization

In our approach, instructions are decoded one after another, corresponding to each hidden state of the content ordering RNN, as generating one single long text sequence could lead to intractability. In addition, we include a penalization factor for reducing redundancy in the cost function $\mathcal{C}$ of beam search:

$$\mathcal{C} = -\sum_{i=1}^{t} \log P_i + \lambda * J_{sim}(q_t, Q_I) \tag{4}$$

where $q_t$ is the set of words currently chosen into the instruction under generation, $Q_I$ represents the set of words in the previously generated instructions, $J_{sim}$ computes Jaccard's Similarity(Hamers et al., 1989) and $\lambda$ is a constant tuned to obtain a maximum BLEU score on development set.

| Model | Accuracy |
|---|---|
| SVM | 73 % |
| Logistic Regression | 91 % |

Table 1: Instruction vs. non-instruction classification results

## 3 Experiments

In this section, empirical evaluation is conducted to quantify the accuracy of our model regarding text generation, content ordering, correctness of grammar and informativeness.

### 3.1 Corpus: ICD Codes to Discharge Instructions[2]

Our dataset consisting of admission ICD codes and their corresponding discharge instructions is derived from MIMIC-III[3]. MIMIC-III is a database containing clinical information regarding patients, admission details, lab tests and medical notes (Ew et al., 2016). For each patient admitted to the hospital, there is a recorded set of ICD codes assigned for billing purposes to specify the diagnoses and clinical procedures related to the patient's admission. We assigned unique IDs for diagnoses and procedure codes and did not distinguish between them. Patients receive a list of discharge instructions written by clinical staff before they return home. These discharge instructions are embedded in larger documents called discharge reports.

We trained statistical models (SVM and logistic regression) to classify instruction sentences (e.g. commands) from non-instruction sentences in the discharge reports using TF-IDF vectors computed from sentences in the discharge reports as features. The dataset is split into the training set, developmental set and test set, each comprising 2,000, 500 and 500 sentences, respectively. The accuracy of this binary classification is in Table 1. The logistic regression classifier was used to construct the corpus that maps ICD codes to discharge instructions, resulting in 18,900 of input output pairs for training and 900 each for testing and development. Manual verification was performed to ensure data quality for testing and developmental sets. Following customary data post-processing protocols, named entities such as numerical values and person names were replaced by place holder tags such as [num] and [person name].

[2]https://github.com/littoncode/mimic_scripts3

[3]https://mimic.physionet.org/gettingstarted/overview/

### 3.2 Implementation Details

We chose different base models and different settings of our model to evaluate the efficiency of our method. In this section we describe each of the models in detail.

#### 3.2.1 Seq2Seq

In this setting we use sequence to sequence model with the attention mechanism (Bahdanau et al., 2014). The set of input ICD codes are arranged as a single sequence in random order and the model generates the entire set of instructions as a single sequence. The learning algorithm minimizes negative log-likelihood. Beam search decoding with beam size of 9 is used during testing.

#### 3.2.2 Set2SingleSeq

In this setup, the model is a variant of (Zhou et al., 2017), where input is a set and content selection is conducted by learning correlations between the input items. This is also a variant of our method, where the output is treated as a single sequence instead of a set of instructions. Any $icd_j$ in the input set of ICD codes are updated to selected content $icd_j^{'}$ using the content selection gate. The content correlation vector $C_j$ for each $icd_j$ is computed below:

$$\alpha_{j,k} = \exp(icd_j^T \mathbf{W_a} icd_k)$$
$$C_j = \sum_{k \neq j} \alpha_{j,k} icd_k$$

The content gate value computation and subsequent content selection is conducted as follows:

$$gc_j = \text{sigmoid}(\mathbf{W_{gc}}[icd_j; C_j])$$
$$icd_j^{'} = gc_j \odot icd_j$$

The entire sequence of instructions is decoded using the computed set of $icd_j^{'}$. For this purpose, the decoder is initialized with the mean vector of $icd_j^{'}$ and the decoder attends over all the set of $icd_j^{'}$ during each time-step of decoding. The learning algorithm minimizes negative log likelihood for optimizing parameters. Beam search decoding with beam size of 9 is used during testing.

#### 3.2.3 Set2MultipleSeq

This setting represents the method explained in the Section 2.2. In this setup, instructions are generated one after another in the learned order. The optimized size for ICD embedding, content ordering RNN, and decoder RNN is 600. Sizes of network parameter matrices $\mathbf{W_a}$, $\mathbf{W_{gc}}$, $\mathbf{W_{gs}}$, $\mathbf{W_c}$ and $\mathbf{W_o}$ are adjusted accordingly. The learning algorithm minimizes negative log likelihood for optimizing parameters. Error is averaged for all instruction decoder time steps for the set of instructions generated. Beam search with beam size of 9 is used during testing.

#### 3.2.4 Set2MultipleSeq+Opt

This setup is an enhanced version of *Set2MultipleSeq* (Section 3.2.3), but during beam search decoding redundancy is penalized (see Section 2.2.3). The value of $\lambda$ in Equation 4 is set to 2.7 to obtain maximum content coverage on the development set.

## 4 Evaluation Study

### 4.1 Evaluation I: Content Generation

Instruction generation is quantitatively estimated by measuring the N-gram match of the generated content with ground-truth in test set using the BLEU metric (Papineni et al., 2002). We did not set any stopping criteria for the number of instructions to be generated for simplicity sake. However, we generated as many number of instructions in the corresponding test record. The results are shown in Table 2. *Set2MultipleSeq* yields a better score than *Seq2Seq*, indicating the effectiveness of content selection and self attention for modeling the correlation between ICD codes. The proposed *Set2MultipleSeq* approach yields even more improvement, validating that the introduction of subset selection and content selection helped in defining the content and context of an instruction during generation. This resulted in more accurate generation of instructions one after the other. When penalizing redundancy during decoding (*Set2MultipleSeq+Opt*), it explicitly forces the instruction decoder to generate an instruction with novel content during each stage of the content ordering RNN, thereby reducing errors caused by repeated content.

| Method | Content Generation | |
|---|---|---|
| | **BLEU** | **BLEU-2** |
| Seq2Seq | 6.3 | 22.30 |
| Set2SingleSeq | 7.6 | 27.00 |
| Set2MultipleSeq | 9.6 | 33.30 |
| Set2MultipleSeq+Opt | **12.6** | **36.30** |

Table 2: Evaluation I: Content Generation.

| Method | Content Ordering | | |
|---|---|---|---|
| | **P** | **R** | **F** |
| Seq2Seq | 0.06 | 0.07 | 0.07 |
| Set2SingleSeq | 0.07 | 0.09 | 0.08 |
| Set2MultipleSeq | 0.18 | 0.19 | 0.18 |
| Set2MultipleSeq+Opt | **0.22** | **0.23** | **0.22** |

Table 3: Evaluation II: Content Ordering: P is Precision, R is Recall and F is F- Measure. The values are approximated to two decimal points

### 4.2 Evaluation II: Content Ordering

For evaluating content ordering, we use a variant of the metric used by Gong et al. (2016). In this scheme, we compare the order of words in generated sequence of instructions with ground-truth: We take the set of all order bigrams $S_o$ from the generated sequence of instructions where the first word in each bigram is from a preceding instruction and the second word is from any instruction succeeding it in the sequence. Precision and recall is computed as:

$$Precision = \frac{|S_o \bigcap S_0'|}{|S_0|}$$

$$Recall = \frac{|S_o \bigcap S_0'|}{|S_0'|}$$

$S_0'$ is the order bigrams in the corresponding human written set of instructions in testset. Thus the metric scores the ordering better when the right content is generated in the right order. F-Measure is computed as the harmonic mean of precision and recall. The results in Table 3 show that there is a considerable improvement in the quality of ordering with after introducing content ordering mechanism in the neural architecture. Better ordering score for $Set2MultipleSeq + Opt$ is obvious as redundant content adversely affect instruction ordering.

| Method | % |
|---|---|
| Set2SingleSeq | 20 |
| Set2MultipleSeq | 77 |
| Ambigous | 3 |

Table 4: Evaluation III: Gramaticality: The values represent percentage of times instructions generated by the model is chosen by the human evaluator.

| Method | % |
|---|---|
| Set2SingleSeq | 30 |
| Set2MultipleSeq | 63 |
| Ambigous | 7 |

Table 5: Evaluation III: Informativeness: The values represent percentage of times instructions generated by the model is chosen by a human evaluator.

| Method | % |
|---|---|
| Set2MultipleSeq | 38 |
| Set2MultipleSeq+opt | 61 |
| Ambigous | 1 |

Table 6: Evaluation III: Informativeness: The values represent percentage of times instructions generated by the model is chosen

### 4.3 Evaluation III: Human Analysis

We conduct human evaluations to measure if the generated instructions are grammatically correct and how informative they are. Four human evaluators who are post graduate students in linguistics were recruited and each given 30 sets of instructions from the testset.

#### 4.3.1 Grammaticality

For each of the 30 instructions chosen for human evaluation, the evaluators were given a number of choices, each generated from a different model, and asked to choose the option that was the most grammatically correct. For each question, the instructions from the different models were shown to the evaluators in a random order to avoid any kind of bias. Instructions generated by *Set2SingleSeq* and *Set2MultipleSeq*. *Set2MultipleSeq+Opt* is excluded as it is an optimization for avoiding redundancy without any direct influence on the grammatical quality of text generated by neural models.

Results aggregated across evaluators through majority voting are shown in Table 4. The results show that incorporating neural network components for content selection and ordering helps in defining the context of an instruction and generating the right content in the right form. *Set2MultipleSeq* generates one instruction at a time, while *Set2SingleSeq* generates the entire set of instructions as one sequence. Grammaticality is shown to improve *Set2MultipleSeq* when only one (shorter) instruction needs to be decoded at a time .

#### 4.3.2 Informativeness

We also conducted human experiments to evaluate the informativeness of generated instructions. The evaluators were asked to read the reference set of instructions prior to examining the instructions generated by competing methods. They were then asked to choose the model that generated instructions retaining the most information from the reference instructions. If two models present instructions that are equally informative, then the model providing richer information density is chosen.

We conducted separate evaluations for two pairs of competing systems (*Set2SingleSeq*, *Set2MultipleSeq*) and (*Set2SingleSeq*, *Set2MultipleSeq+Opt*). The results are shown in Tables 5 and 6 respectively. Results shown in Table 5 explains that incorporating neural components for subset selection and content ordering helps in improving informative instruction generation. We observe that conducting content selection multiple times during each time step through content ordering RNN helps in generating a discrete set of instructions (*Set2MultipleSeq*). Table 6 shows that penalizing redundancy during beam search decoding reduces noise and helps in generating instructions with rich information density. Inter-evaluator agreement for the entire set of human evaluation is reasonably high: Cohen's kappa coefficient is 0.79.

## 5 Discussions

### 5.1 Qualitative Comparison Across Models

Table 7 shows an example of instructions generated by the different approaches we investigated. Though *Set2SingleSeq* generates relevant instructions, it repeats the same content, misses out on instructions, and is less grammatically correct.

*Set2MultipleSeq* shows qualitative improvement over *Set2SingleSeq*, where the repeated content is largely reduced and the grammaticality is improved. However, for *Set2MultipleSeq*, there is

| Method | Generated Content |
|---|---|
| Set2SingleSeq | shower daily including washing incisions gently with mild soap , no baths or swimming until cleared by surgeon. shower daily and pat incisions dry no lotions , creams or powders on incisions no driving for [num]. shower daily , let water flow over wounds , pat dry with a towel towel , do no |
| Set2MultipleSeq | **1)** shower daily and pat incisions dry no lotions , creams or powders on incisions, no baths or swimming until cleared by surgeon.<br>**2)** no lifting greater then [num] pounds for [num] weeks , do not drive or operate heavy machinery while taking any narcotic pain medication such as percocet<br>**3)** call for any fever , redness or drainage from wounds or weight gain more than [num] pounds<br>**4)** call your doctor for any fever , redness or drainage from wounds |
| Set2MultipleSeq+Opt | **1)** shower daily and pat incisions dry no lotions , creams or powders on incisions, no baths or swimming until cleared by surgeon.<br>**2)** no lifting greater then [num] pounds for [num] weeks , do not drive or operate heavy machinery while taking any narcotic pain medication such as percocet<br>**3)** call for any fever , redness or drainage from wounds or weight gain more than [num] pounds<br>**4)** follow up with your primary care doctor , dr [person name] , in the next week as well |
| Nurse Written Instructions | **1)** shower, no baths, no lotions,creams or powders to incisions<br>**2)** no lifting more than [num] pounds for [num] weeks from surgery<br>**3)** do not drive or drink alcohol while taking narcotic pain medications<br>**4)** call with fever, redness or drainage from incision or weight gain more than [num] pounds in one day or five in one week. |

Table 7: Examples from Evaluation II: Content Ordering. Person names and numeric values are specified as [person name] and [num] respectively. Repetitive content is color coded in blue for *Set2SingleSeq* and in purple for *Set2MultipleSeq*.

still repeated content in the 3rd and 4th instructions.

*Set2MultipleSeq+Opt* brings more tractability in the neural model by penalizing redundancy and preventing noisy content generation. In the example shown, *Set2MultipleSeq+Opt* improves over *Set2MultipleSeq*, as the content in the 3rd instruction is no longer repeated in the 4th instruction.

### 5.2 Variability in Groundtruth References

Variability stemming from communication style differences across clinicians could potentially be one reason why the overall scores for content generation (BLEU scores) and content ordering (F-measure) are on the low end. We observed that for the same set of ICD codes, different clinicians have different writing styles, even when the underlying content of the instructions are the same. Consider the following examples:

(a) shower daily and pat incisions dry no lotions , creams or powders on incisions, no baths or swimming until cleared by surgeon

(b) shower, no baths, no lotions,creams or powders to incisions

In (a), a more detailed way of representing the in-

struction is presented, while in (b) the same information is represented in a more succinct manner.

If a clinician decides to be more detailed in writing the instructions, the clinician might also include specific information such as medication details: "Please be sure to take aspirin and plavix everyday as directed." Such inconsistency of how information such as medication details are specified in the instructions can potentially lead the models to generate noisy content.

## 6 Related Prior Work

### 6.1 Natural Language Generation

Here we define natural language generation as the task of generating text from textual input or non-linguistic input (e.g. graphs, data records). Previous work on text generation span across various input types and methods. Initial models used learned rules (Reiter et al., 2003, 2005) and manually engineered templates (Kukich, 1983; McRoy et al., 2000; McKeown, 1985) for constructing the output text. There is also work using automatic means for generating templates (Angeli et al., 2010; Howald et al., 2013).

Such template approaches could be inherently less efficient in modeling semantics when compared to neural networks, especially if there is ample training data. Initial neural models for text generation were first motivated by machine translation (Bahdanau et al., 2014; Cho et al., 2014; Srivastava et al., 2014). However, such approaches are less suitable in comprehending the semantics of structures that are more complex than short sequences and in generating longer sequences (Paulus et al., 2017; Wiseman et al., 2018). Recent advancement in text to text generation has primarily focused on news document summarization (Cheng and Lapata, 2016; Nallapati et al., 2017; See et al., 2017; Paulus et al., 2017).

A portion of related work focuses on generating text from an input graph (Koncel-Kedziorski et al., 2019; Song et al., 2018, 2017): Graphs with labeled edges (e.g., knowledge graphs or abstract meaning representation) are used to generate a direct description of the information characterized in the input graph.

Text generation methods from data records has been investigated for different datasets such as wikipedia infobox (Lebret et al., 2016; Liu et al., 2018; Sha et al., 2018; Perez-Beltrachini and Lapata, 2018), weather predictions (Mei et al., 2016) or sport game summaries (Wiseman et al., 2017). There is a subset of work on text generation from data records which relies on content planning for generating a single sentence (Liu et al., 2018), while some other researchers generate multi-sentence outputs from structured data inputs (Puduppully et al.; Jhamtani et al., 2018). Puduppully et al. generated basketball game summaries by explicitly learning the order in which information should be mentioned in the output by generating an intermediate content plan.

### 6.2 Text Generation in Healthcare

A lot of NLP work in the medical domain has focused on information extraction (see (Wang et al., 2018) for a review). Driven by healthcare application demands, recently there is emerging interest in areas such as automatic ICD code assignment (Zhang et al., 2017; Scheurwegs et al., 2017; Mullenbach et al., 2018), risk prediction (Ma et al., 2018), and dialogue comprehension (Liu et al., 2019a). One of the earliest work on text generation in the medical domain dates back to more than two decades ago, where interactive systems generate summaries of the patient status for physicians and tailored explanations of clinical information for individual patients (Buchanan et al., 1995). Recent advances of neural modeling has rekindled interest in text generation. Text generation has had more presence in the media and journalism domain, ranging from image captioning (Kinghorn et al., 2018) to news reports on weather (Reiter et al., 2005) and sports (Wiseman et al., 2017). For the medical domain, interest in neural text generation has been springing up, with a primary focus on document summarization (Moradi and Ghadiri, 2018). For interested readers, Pauws et al. (2019) provides a recent review on how data-to-text technology can be applied in healthcare settings.

## 7 Conclusion

We proposed a neural architecture that learns to generate content in a specific order (discharge instructions for patients) without explicit specifications of the relations between input entities (ICD codes representing diagnoses and procedures) and how the input entities relate to the output. Our approach yields encouragingly better results in comparison with strong baselines. We further improved performance by penalizing redundancy during decoding.

## 8 Acknowledgement

We thank Jiewen Wu for pointing us to the MIMIC dataset and sharing his experience with us. We thank Ai Ti Aw, Mien Ho, Pavitra Krishnaswamy, and Zhengyuan Liu for their insightful discussions. We are also grateful for the encouraging and constructive feedback from the anonymous reviewers.

Research efforts were supported by funding for Digital Health from the Science and Engineering Research Council (SERC Project No: A1818g0044), A*STAR, Singapore. In addition, this work was conducted using resources and infrastructure provided by the Human Language Technology unit at I2R.

## References

Gabor Angeli, Percy Liang, and Dan Klein. 2010. A simple domain-independent probabilistic approach to generation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 502–512. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bruce G Buchanan, Johanna D Moore, Diana E Forsythe, Giuseppe Carenini, Stellan Ohlsson, and Gordon Banks. 1995. An intelligent interactive system for delivering individualized information to patients. *Artificial intelligence in medicine*, 7(2):117–154.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 484–494.

Kyunghyun Cho, B van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8), 2014*.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*.

Elizabeth Clark, Yangfeng Ji, and Noah A Smith. 2018. Neural text generation in stories using entity representations as context. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2250–2260.

Alistair Ew, Tom J Pollard, Lu Shen, H Lehman Li-wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3:160035.

Jingjing Gong, Xinchi Chen, Xipeng Qiu, and Xuanjing Huang. 2016. End-to-end neural sentence ordering using pointer network. *arXiv preprint arXiv:1611.04953*.

Lieve Hamers et al. 1989. Similarity measures in scientometric research: The jaccard index versus salton's cosine formula. *Information Processing and Management*, 25(3):315–18.

Blake Howald, Ravikumar Kondadadi, and Frank Schilder. 2013. Domain adaptable semantic clustering in statistical nlg. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013)–Long Papers*, pages 143–154.

Harsh Jhamtani, Varun Gangal, Eduard Hovy, Graham Neubig, and Taylor Berg-Kirkpatrick. 2018. Learning to generate move-by-move commentary for chess games from large-scale social forum data. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1661–1671.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally coherent text generation with neural checklist models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 329–339.

Philip Kinghorn, Li Zhang, and Ling Shao. 2018. A region-based image caption generator with refined descriptions. *Neurocomputing*, 272:416–424.

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text generation from knowledge graphs with graph transformers. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2284–2293.

Karen Kukich. 1983. Design of a knowledge-based report generator. In *Proceedings of the 21st annual meeting on Association for Computational Linguistics*, pages 145–150. Association for Computational Linguistics.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1203–1213.

Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. 2018. Table-to-text generation by structure-aware seq2seq learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Zhengyuan Liu, Hazel Lim, Nur Farah Ain Suhaimi, Shao Chuen Tong, Sharon Ong, Angela Ng, Sheldon Lee, Michael R Macdonald, Savitha Ramasamy, Pavitra Krishnaswamy, et al. 2019a. Fast prototyping a dialogue comprehension system for nurse-patient conversations on symptom monitoring. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Industry Papers)*, pages 24–31.

Zhibin Liu, Zheng-Yu Niu, Hua Wu, and Haifeng Wang. 2019b. Knowledge aware conversation generation with reasoning on augmented graph. *arXiv preprint arXiv:1903.10245*.

Fenglong Ma, Jing Gao, Qiuling Suo, Quanzeng You, Jing Zhou, and Aidong Zhang. 2018. Risk prediction on electronic health records with prior medical knowledge. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1910–1919. ACM.

Kathleen R McKeown. 1985. Discourse strategies for generating natural-language text. *Artificial Intelligence*, 27(1):1–41.

Susan W McRoy, Songsak Channarukul, and Syed S Ali. 2000. Yag: A template-based generator for real-time systems. In *Proceedings of the first international conference on Natural language generation-Volume 14*, pages 264–267. Association for Computational Linguistics.

Hongyuan Mei, Mohit Bansal, and Matthew R Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 720–730.

Milad Moradi and Nasser Ghadiri. 2018. Different approaches for identifying important concepts in probabilistic biomedical text summarization. *Artificial intelligence in medicine*, 84:101–116.

James Mullenbach, Sarah Wiegreffe, Jon Duke, Jimeng Sun, and Jacob Eisenstein. 2018. Explainable prediction of medical codes from clinical text. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1101–1111.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Steffen Pauws, Albert Gatt, Emiel Krahmer, and Ehud Reiter. 2019. Making effective use of healthcare data using data-to-text technology. In *Data Science for Healthcare*, pages 119–145. Springer.

Laura Perez-Beltrachini and Mirella Lapata. 2018. Bootstrapping generators from noisy data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1516–1527.

Ratish Puduppully, Li Dong, and Mirella Lapata. Data-to-text generation with content selection and planning.

Ehud Reiter, Roma Robertson, and Liesl M Osman. 2003. Lessons from a failure: Generating tailored smoking cessation letters. *Artificial Intelligence*, 144(1-2):41–58.

Ehud Reiter, Somayajulu Sripada, Jim Hunter, Jin Yu, and Ian Davy. 2005. Choosing words in computer-generated weather forecasts. *Artificial Intelligence*, 167(1-2):137–169.

Elyne Scheurwegs, Boris Cule, Kim Luyckx, Léon Luyten, and Walter Daelemans. 2017. Selecting relevant features from the electronic health record for clinical code prediction. *Journal of biomedical informatics*, 74:92–103.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083.

Lei Sha, Lili Mou, Tianyu Liu, Pascal Poupart, Sujian Li, Baobao Chang, and Zhifang Sui. 2018. Order-planning neural text generation from structured data. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Linfeng Song, Xiaochang Peng, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2017. Amr-to-text generation with synchronous node replacement grammar. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 7–13.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. A graph-to-sequence model for amr-to-text generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational*

*Linguistics (Volume 1: Long Papers)*, pages 1616–1626.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Yanshan Wang, Liwei Wang, Majid Rastegar-Mojarad, Sungrim Moon, Feichen Shen, Naveed Afzal, Sijia Liu, Yuqun Zeng, Saeed Mehrabi, Sunghwan Sohn, et al. 2018. Clinical information extraction applications: a literature review. *Journal of biomedical informatics*, 77:34–49.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2018. Learning neural templates for text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3174–3187.

Danchen Zhang, Daqing He, Sanqiang Zhao, and Lei Li. 2017. Enhancing automatic icd-9-cm code assignment for medical texts with pubmed. In *BioNLP 2017*, pages 263–271.

He Zhao, Chong Feng, Zhunchen Luo, and Changhai Tian. 2018. Entity set expansion from twitter. In *Proceedings of the 2018 ACM SIGIR International Conference on Theory of Information Retrieval*, pages 155–162. ACM.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2017. Selective encoding for abstractive sentence summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1095–1104.

# Constraint-based Learning of Phonological Processes

Shraddha Barke, Rose Kunkel, Nadia Polikarpova, Eric Meinhardt, Eric Baković, and Leon Bergen

{sbarke,wkunkel,npolikarpova,emeinhar,ebakovic,lbergen} @ucsd.edu

University of California, San Diego

## Abstract

Phonological processes are context-dependent sound changes in natural languages. We present an unsupervised approach to learning human-readable descriptions of phonological processes from collections of related utterances. Our approach builds upon a technique from the programming languages community called *constraint-based program synthesis*. We contribute a novel encoding of the learning problem into Satisfiablity Modulo Theory constraints, which enables both data efficiency and fast inference. We evaluate our system on textbook phonology problems and lexical databases, and show that it achieves high accuracy at speeds two orders of magnitude faster than state-of-the-art approaches.

## 1 Introduction

Phonological processes govern the way speech sounds in natural languages change depending on the context. For example, in English verbs, the past tense suffix /d/ turns into [t] after voiceless consonants (so the word "zipped" is pronounced [zɪp**t**], while "begged" is pronounced [bɛg**d**]). Linguists routinely face the task of inferring phonological processes by observing and contrasting *surface forms* (pronunciations) of morphologically related words. To aid linguists with this task, we consider the problem of learning phonological processes automatically from collections of related surface forms.

This problem setting imposes four core requirements, which guide the design of our approach:

1. Inference results must be **fully interpretable**: our goal is to *explain* phonological processes exhibited by the data, not merely *predict* pronunciations of unseen words. Hence, our model takes the form of discrete, conditional rewrite rules from rule-based phonology (Chomsky and Halle, 1968).

2. Inference must be **unsupervised**: phonological processes are formalized as transformations from (latent) *underlying forms* to surface forms (rather than *between* surface forms).

3. Inference must be **data-efficient**: typically only a handful of data points are available.

4. Inference must be **fast**: we envision linguists using our system interactively, tweaking the data and being able to see the inferred rules within minutes.

Recently *program synthesis* has emerged as a promising approach to interpretable and data-efficient learning (Ellis et al., 2015; Singh et al., 2017; Verma et al., 2018; Ellis et al., 2018). In program synthesis, models are represented as programs in a domain-specific language (DSL), which allows domain experts to impose a strong prior by designing an appropriate DSL. Program synthesis uses powerful constraint solvers to perform combinatorial optimization and find the least-cost program in the DSL that fits the data. Program synthesis has been previously used to tackle the problem of phonological rule learning (Ellis et al., 2015), however their work uses global inference which scales poorly and hence does not satisfy requirement 4 (their system takes an hour on average to solve a phonology textbook problem).

In this work, we propose a novel inference technique that satisfies all four core requirements. Our key insight is that the problem of learning conditional rewrite rules can be decomposed into three steps: inference of the latent *underlying forms*, learning the *changes* (rewrites), and learning the *conditions*. Moreover, each of these problems can be encoded as a constrained optimization problem that can be solved efficiently by modern *satisfiability modulo theories* (SMT) solvers (de Moura and Bjørner, 2008). Both the decomposition and the encoding into constraints are contributions of this work. We implement this



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6176–6186, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

approach in a system called SYPHON and show that it is capable of generating accurate phonological rules in under a minute and from just 5–30 data points.

## 2 Background and Problem Definition

In this section, we illustrate phonological processes and the problem of phonological rule induction using our running example of English verbs.

### 2.1 Rule-Based Phonology

***Phonological features.*** *Phones* (speech sounds) are described using a feature system that groups similar-sounding phones together. For instance, voiced consonants (consonants produced with vibrating vocal cords, like [z], [d], [b]) possess the features +consonant and +voice, while voiceless consonants (like [s], [t], [p]) possess the features +consonant and −voice. Each phone can be uniquely identified by a feature vector: for example [−voice −strident +anterior −distributed] uniquely identifies the sound [t]. However, some phones may be uniquely identified by several feature vectors, and not all feature vectors correspond to phones (the feature system is redundant). For example, the feature vector [+low +high] does not correspond to any phones, as no phone can have both a raised and a lowered tongue body.

***Phonological rules.*** In rule-based phonology, a phonological process is formalized as a conditional rewrite rule that transforms an *underlying form* of a word (roughly, the unique stored form of the word) into its *surface form* (the word as it is intended to be pronounced). In our English past tense example, the underlying form /zɪpd/—formed by concatenating the stem /zɪp/ and past tense suffix /d/—is transformed into the surface form [zɪpt] by a rule that makes an obstruent voiceless when it occurs after a voiceless obstruent:

$$[-\text{sonorant}] \rightarrow [-\text{voice}] \,/\, [-\text{voice}] \_$$

In general, phonological rules have the form $A \rightarrow B \,/\, L \_ R$, where all of $A$, $B$, $L$, and $R$ are feature vectors. The rule means that any phone that matches $A$ and occurs between two phones that match $L$ and $R$, respectively, will be rewritten to match $B$ (leaving the features not mentioned by $B$ intact). $A$ is called the *target* of the rule, $B$ is called the *structural change*, and $L$ and $R$ are the left and the right *contexts*.[1] In the example above, the right context is omitted, because it is irrelevant to the rule's application; formally, $A$, $L$, and $R$ may each be empty feature vectors, which are defined to match any phone.

Hereafter, we refer to the sequence $LAR$ of the target and the context as the *condition* of the rule. If the condition is empty, the rule applies unconditionally. In addition to + and −, the values of features in the condition of the rule may be *variables*, which enforce that features have the same value in different parts of the condition. For example,

$$A \rightarrow B \,/\, [\alpha\text{consonant}] \_ [\alpha\text{consonant}]$$

describes a rule which applies between pairs of consonants and pairs of vowels, but not between a consonant and a vowel.

### 2.2 Problem Definition

The input to our problem is a matrix of surface forms, such as the one shown in Fig. 1, left. These forms are arranged into rows, corresponding to different stems, and columns, corresponding to different inflections (in this case, the third-person singular and past tense of English verbs). In the interest of space, we only show four rows from this data set, but a typical input in a phonology textbook problem is only slightly larger and ranges from 5 to 30 rows.

Given these data, our task is to infer the latent underlying forms for each of the words in the input such that the resulting matrix of underlying forms factorizes into stems and suffixes, and to learn a sequence of phonological rules which, when applied elementwise to the matrix of underlying forms, reproduces the matrix of surface forms.

This learned sequence of phonological rules is generative in the following sense: given the underlying form for a new word, such as /æskz/, we can deterministically apply these rules to generate the surface form of that word, [æsks]. We use this property to evaluate the accuracy of the rule set we learned by holding out a portion of the words from the data, and then applying the rule to the underlying forms of those words, which were determined through phonological research.

### 2.3 Phonological Intuition

The design of our system is informed by how linguists solve the problem of phonological rule induction. When a phonologist analyzes these data, they begin by positing underlying forms that are likely to result in the simplest set of rules. For example, they observe that the substring shared in each row is most likely the stem, which surfaces without change; the underlying suffix in the first column in Fig. 1 is likely

[1] In this work we only consider a subset of *strictly local* $k$=3 rules (Chandlee et al., 2014), where either side of the context is restricted to at most a single phone.

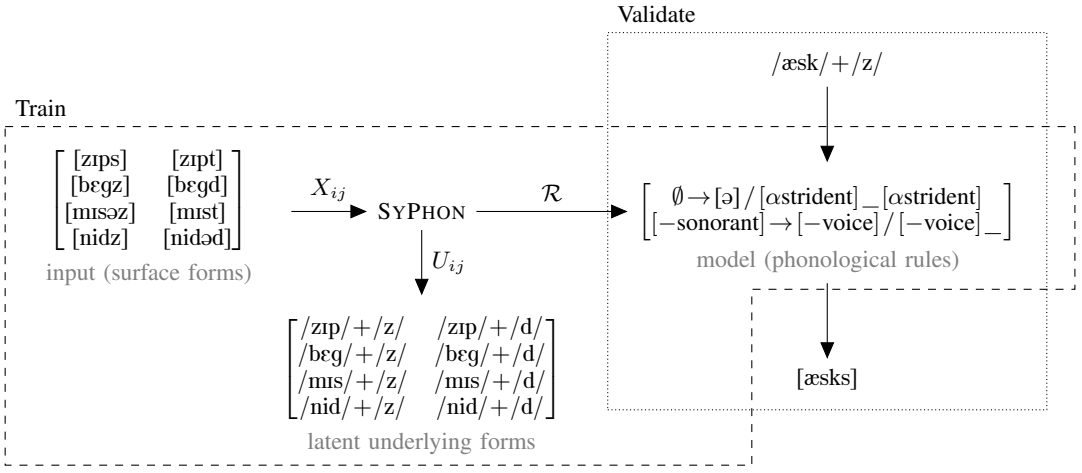


Figure 1: The general structure of the problem, shown concretely for English verbs.

/z/, which sometimes surfaces as [s] and other times as [əz]; and similarly, the underlying suffix in the second column is likely /d/, which can change to [t] or [əd]. The choice of /z/ and /d/ as the underlying suffixes is preferred to, say, /s/ and /t/, because this choice lets us explain all the observed data using only three edits: /z/ → [s], /d/ → [t], and ∅ → [ə].

The next step is to merge and generalize individual edits: the first two edits are both devoicing an obstruent, so they can be merged into [−sonorant] → [−voice], while the last edit is an insertion and cannot be generalized.

The final step of the analysis is to infer the conditions under which each of the two structural changes occurs. By contrasting examples in the first column, we infer that the insertion happens when the suffix /z/ occurs after a strident (like /s/ in /mɪs/); otherwise, /z/ and /d/ are devoiced whenever they occur after a voiceless obstruent (like /p/ in /zɪp/). The full data set can be explained using the two rules in Fig. 1, right. Note that in order to capture the data in both columns, the insertion rule says that [ə] is inserted whenever the stridency of the left and right context matches. Note also that in this case the order of rules matters: for words like /mɪsz/, insertion is applied first, which prevents the devoicing rule from applying.

## 3 Learning Phonological Rules

As illustrated in Fig. 1, the input to our learning problem is a matrix of surface forms $X_{ij}$ with $I$ rows and $J$ columns. The goal is to learn a discrete rule set $\mathcal{R}$, while jointly inferring the latent set of $I$ stems $S_i$ and $J$ affixes $A_j$.

***Hypothesis space.*** The hypothesis space for $\mathcal{R}$ can be formalized as a context-free grammar:

$$\begin{aligned} &\mathcal{R} \Rightarrow R^* && R \Rightarrow C \rightarrow C \,/\, C\_C \\ &C \Rightarrow (VF)^* && V \Rightarrow + \,|\, - \\ & && F \Rightarrow \text{consonant} \,|\, \text{voice} \,|\, \ldots \end{aligned} \tag{1}$$

According to this grammar, $\mathcal{R}$ is a sequence of rules $R$; each $R$ is defined in terms of four feature vectors $C$; each feature vector is a sequence of pairs of feature values $V$ and feature names $F$.

***Rewriting.*** We use $\mathcal{C}_R$ and $\mathcal{B}_R$ to denote the condition and structural change of a rule $R$, respectively. A feature vector $C$ can be interpreted as a Boolean formula that holds of a phone $a$ if $a$ possesses all features in $C$; we denote by $|C|$ the number of models of this formula, *i.e.* phones in the inventory $\Phi$ that satisfy $C$. Similarly, $\mathcal{C}_R$ is a Boolean formula over *trigrams* of phones. A *rewrite* of a trigram $abc$ by rule $R$ is defined as:

$$R(abc) = \begin{cases} \mathcal{B}_R(b) & \text{if } \mathcal{C}_R(abc) \\ b & \text{otherwise} \end{cases}$$

The notion of rewrites can be extended to words and rule sets.

***Learning as constrained optimization.*** We can now formalize our problem as a hard *correctness constraint* over rules and underlying forms $U_{ij}$:

$$\mathcal{R}(U_{ij}) = X_{ij} \quad \text{where } U_{ij} \triangleq A_j[S_i] \tag{2}$$

Here, $A_j[S_i]$ denotes a concatenation of the prefix/suffix $A_j$ with the stem $S_i$.

There might be many rule sets $\mathcal{R}$ that are consistent with all the data, and what we would like is to pick one that generalizes to other data that exhibits the same phonological process (for example, the rule inferred in Fig. 1 should generalize to other regular English verbs). Hence we frame the learning problem

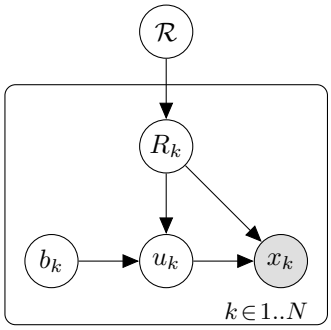


Figure 2: Probabilistic model of a phonological process. A rule set $\mathcal{R}$ is sampled from a description length prior. We observe a set of $N$ surface phonemes $x_k$; each $x_k$ is generated by sampling a rule $R_k$ from $\mathcal{R}$ and an underlying trigram $u_k$, and deterministically applying $R_k$ to $u_k$ (coin flip $b_k$ decides whether $u_k$ should match $R_k$'s condition).

as a constrained optimization problem and derive the objective function using a Bayesian model.

### 3.1 Bayesian Model

***Generative process.*** Intuitively, to generate surface forms $X_{ij}$, we must sample a single rule set $\mathcal{R}$, $I$ stems $S_i$, and $J$ affixes $A_j$, and then deterministically apply $\mathcal{R}$ to each $A_j[S_i]$. Prior work on phonological rule learning (Ellis et al., 2015) assumed that $S_i$ and $A_j$ are sampled uniformly from the language and independently of $\mathcal{R}$. We observe, however, that in most data sets of interest, underlying forms are in fact sampled to contrast the contexts in which $\mathcal{R}$ does and does not apply. We model this intuition as a *strong sampling* process depicted in Fig. 2.

For simplicity, in this model each observation corresponds to an individual rule application to an underlying trigram $u$ that produces a surface phoneme $x$. For example, the rewrite /zɪpz/ → [zɪps] is represented as four observations: /#zi/ → [z], /zɪp/ → [ɪ], /ipz/ → [p], and /pz#/ → [s] (where # encodes word boundary).

Our generative process first samples a ruleset $\mathcal{R}$ from the description length prior over the hypothesis space (1):

$$P(\mathcal{R}) \propto 2^{-w_s \cdot \sum_{R \in \mathcal{R}} \ell(R)}$$

where $\ell(R)$ is the length of rule $R$ and $w_s > 0$ is a model hyperparameter. For each observation $k \in 1..N$, we pick a rule $R_k$ uniformly from $\mathcal{R}$. Before sampling the underlying trigram $u_k$, we flip a coin $b_k$ to decide whether we want to sample a *positive* or a *negative* trigram, *i.e.* whether $\mathcal{C}_{R_k}(u_k)$ should hold true; we then sample $u_k$ uniformly from the set of all positive (resp. negative) trigrams (subject to the hard constraint that they form a factorizable matrix $U_{ij}$). Finally, we deterministically compute $x_k \triangleq R_k(u_k)$. Hence we can define:

$$P(x_k, u_k \mid R_k) = \begin{cases} 0 & \text{if } R_k(u_k) \neq x_k \\ \frac{P(b_k = \top)}{|\mathcal{C}_{R_k}|} & \text{if } \mathcal{C}_{R_k}(u_k) \\ \frac{P(b_k = \bot)}{|\neg\mathcal{C}_{R_k}|} & \text{otherwise} \end{cases}$$

Our goal is to maximize

$$P(\mathcal{R}, R_1, ..., R_N, u_1, ..., u_N \mid x_1, ..., x_N)$$
$$\propto P(\mathcal{R}) \prod_{k=1}^{N} P(x_k, u_k \mid R_k) P(R_k \mid \mathcal{R})$$

***Objective function.*** Taking logs, we can derive the following approximate minimization objective for our constrained optimization problem:

$$w_s \sum_{R \in \mathcal{R}} \ell(R) + N_R^+ \cdot \log(|\mathcal{C}_R|) \qquad (3)$$

where $N_R^+$ is the number of positive examples for this rule. (Note that this objective ignores $P(R_k \mid \mathcal{R})$ and $b_k$, which are assumed to be uniform. It also ignores the negative examples. This provides a reasonable approximation, under the assumption that $|\neg\mathcal{C}_R| \gg |\mathcal{C}_R|$ for each rule $R$, which holds in the current setting.) This function includes a *simplicity term*, which favors rules with shorter (and hence, more general) conditions, and a *likelihood term*, which favors more specific conditions if there are sufficient positive examples to support them. This likelihood term stems from our strong sampling assumption; we demonstrate its importance for inferring accurate rules in Sec. 5.

### 3.2 Inference by Program Synthesis

To solve the constrained optimization problem we build upon a technique from programming languages called *constraint-based program synthesis* (Solar-Lezama, 2013).

***Constraint-based synthesis.*** The input to (inductive) program synthesis is a DSL that defines the space of possible programs and a set of input-output examples $E = \overrightarrow{\langle i, o \rangle}$; the goal is to find a program whose behavior is consistent with the examples. In constraint-based synthesis, this search problem is reduced to solving a boolean constraint. To this end, we index the DSL by a bitvector $c$, called a *control vector*. We then define a mapping from control vectors to program behaviors via an *evaluation relation* $\varphi(c, y, z)$—a boolean formula that holds if and only if a program indexed by $c$ produces output $z$ on input $y$. Given the evaluation relation, the synthesis problem reduces to solving the following boolean constraint:

$$\exists c. \bigwedge_{\langle i,o \rangle \in E} \varphi(c, i, o)$$

An SMT solver (de Moura and Bjørner, 2008) is then used to find a satisfying assignment for $c$, which allows us to recover the corresponding program. For this approach to succeed, the evaluation relation has to be designed carefully so that it only uses constraints that the solver can efficiently reason about.

***Synthesis of phonological rules.*** In our setting, the DSL is the space of all rule sets $\mathcal{R}$ (up to a certain size), and the evaluation relation $\varphi(c,U,X)$ is the correctness condition (2). Importantly, our setting differs from traditional program synthesis in two ways: first, we have to search for both the control vector and the inputs, and second, in addition to satisfying the correctness condition, we also seek to minimize the objective function (3). If we encode the objective function as $\psi(c,\overrightarrow{\langle U,X\rangle})$, we can reduce rule learning to the following constrained optimization:

$$\textbf{minimize} \quad \psi(c,\overrightarrow{\langle A_j[S_i],X_{ij}\rangle})$$
$$\textbf{subject to} \quad \bigwedge_{i,j=1,1}^{N,M} \varphi(c,A_j[S_i],X_{ij})$$

Given a proper encoding of $\varphi$ and $\psi$, this constraint can be solved by an optimizing SMT solver (Bjørner et al., 2015); this is the approach used in prior work (Ellis et al., 2015). However, this is a very computationally intensive problem. The reason is the astronomical size of the search space: for a problem of factorizing a $10\times 2$ matrix $X_{ij}$ into stems of length $\ell_S = 3$ and affixes of length $\ell_A = 2$, if we limit the maximum number of rules $N_{\mathcal{R}}$ to 2 and consider an inventory $\Phi$ with 90 phones and a feature set $F$ with 30 features, we can estimate the size of the search space as $3^{|F|N_{\mathcal{R}}}|\Phi|^{I\ell_S+J\ell_A} \approx 2^{600}$.

***Decomposition.*** To achieve scalable inference, we decompose the global constrained optimization problem into three steps, inspired by phonological intuition we described in Sec. 2.3:

1. *Underlying form inference.* In the first step we use an SMT solver to generate likely underlying stems and suffixes. We rank them based on the heuristic that underlying forms $U_{ij}$ that have a smaller edit distance from surface forms $X_{ij}$ are more likely to produce simple rules (Sec. 3.3).

2. *Change inference.* Given the set of edits between each $U_{ij}$ and the corresponding $X_{ij}$, we identify the smallest set of structural changes $B$ that can describe all the edits (Sec. 3.4).

3. *Condition inference.* Finally, for each structural change $B$, we use program synthesis to infer the condition under which this change occurs (Sec. 3.5). If this step fails, we go back to step 1 and generate the next candidate matrix $U_{ij}$.

In the rest of this section we detail these three steps. For illustration purposes, in all examples we will assume that our feature set has just three features: voice v, sonorant s, and continuant c.

### 3.3 Underlying Form Inference

The input to this step is the matrix of surface forms $X_{ij}$ and the output is a set of aligned pairs $\langle U,X\rangle_{ij}$. Tab. 1 illustrates this for a $2\times 2$ matrix of English verbs. For example, $\langle U,X\rangle_{11} = \langle$[zɪpz],[zɪps]$\rangle$; we use red to show alignment information (in this case, a single substitution $z \to s$). Insertions and deletions are represented by alignment with null segments.

| Input | | Output | |
|---|---|---|---|
| [zɪps] | [zɪpt] | [zɪpz]<br>[zɪps] | [zɪpd]<br>[zɪpt] |
| [nidz] | [nidəd] | [nidz]<br>[nidz] | [nid_d]<br>[nidəd] |

Table 1: Underlying form inference on English verbs.

The output matrix $\langle U, X\rangle_{ij}$ has to satisfy two properties: (i) the matrix $U_{ij}$ can be factorized into stems $S_i$ and affixes $A_j$, and (ii) each pair $\langle U,X\rangle$ has a small edit distance. Our intuition is that underlying forms that have a small edit distance from surface forms are likely to produce simpler rules. Hence we generate candidate matrices $\langle U,X\rangle_{ij}$ in the order of increasing edit distance, until rule inference succeeds for one of them. This strategy will always eventually find a matrix of underlying forms which can be related to the surface forms by a rule set we can infer as long as one exists. This process is not guaranteed to find the global minimum of the objective function (3), but we show empirically that it produces good results.

We can encode the properties (i) and (ii) as a boolean constraint over unknown strings with concatenation and length, which can be solved efficiently by the Z3STR2 solver (Zheng et al., 2017). From the solutions for those unknowns it is straightforward to recover not only the stems and suffixes, but also the required alignment information between the underlying and surface forms.

### 3.4 Change Inference

The input to change inference is the set of all edits in the aligned pairs $\langle U,X\rangle_{ij}$, computed in the previous step, and the output is a set of structural changes

that captures all the edits. Tab. 2 illustrates this for the edits from Tab. 1; columns LHS and RHS show relevant features of the left- and and right-hand sides of the edit.

| Edit | LHS | RHS | Change |
|---|---|---|---|
| /z/ → [s] | [+v −s +c] | [−v −s +c] | [−v] |
| /d/ → [t] | [+v −s −c] | [−v −s −c] | |
| ∅ → [ə] | ∅ | [ə] | [ə] |

Table 2: Change inference on English verbs.

For each edit, we compute the set of all possible structural changes which are consistent with the edit. For example, the edit /z/ → [s] is consistent with four possible changes: [−v], [−v −s] [−v +c], and [−v −s +c]. Next, we greedily merge change-sets of different edits if their intersection is nonempty. This merging step allows us to identify a small set of distinct structural changes which together describe all the edits. For example, the change-sets of the first two edits in Tab. 2 can be merged to produce the change-set: {[−v],[−v −s]}. The third edit in Tab. 2 is an insertion, which changes the values of *all* features present in [ə], and hence cannot be merged. When no more merges are possible, we pick the simplest change from every change set (in this case, we end up with changes $B_1 = [-\text{v}]$ and $B_2 = [\text{ə}]$). This greedy process bounds the maximum number of rules to the number of change sets.

### 3.5 Condition Inference

For each structural change $B$ inferred in the previous step, we now attempt to determine the condition $LAR$ under which the change applies. If successful, a rule $A \rightarrow B \;/\; L \_ R$ is added to the inferred rule set $\mathcal{R}$; otherwise we go back to underlying form inference and try the next candidate matrix $U_{ij}$.

For a given change $B$, the input to condition inference is the set of pairs $\langle u, \ell \rangle_k$, where $u_k$ is a phone trigram in some underlying form and the label $\ell_k$ can be *positive* ($\top$), *negative* ($\bot$), or *unknown* (?). Tab. 3 illustrates this for trigrams from $U = /\text{zɪpz}/$. A trigram is labeled positive if its middle phone undergoes the change $B$ in the data, negative if it does not undergo $B$, and unknown if $B$ has no effect on this phone. In our example, neither /ɪ/ nor /p/ in /zɪps/ actually changed, however /zɪp/ is labeled $\bot$ while /ɪpz/ is labeled ?, because /p/ is already [−v], and hence devoicing has no effect on it. Our goal is to infer a condition consistent with the labels of all the positive and negative trigrams (unknown trigrams are ignored).

| $u$ | $\ell$ | Features | | |
|---|---|---|---|---|
| /#zɪ/ | $\bot$ | [+#] | [+v −s] | [+v +s] |
| /zɪp/ | $\bot$ | [+v −s] | [+v +s] | [−v −s] |
| /ɪpz/ | ? | [+v +s] | [−v −s] | [+v −s] |
| /pz#/ | $\top$ | [−v −s] | [+v −s] | [+#] |

Table 3: Input to condition inference for change [−v] on /zɪpz/ → [zɪps]

***Inference by program synthesis.*** To frame condition inference as a program synthesis problem we need to define the control vector that indexes the space of all possible conditions, and a corresponding evaluation relation. In our control vector, for each feature $f$, we use six control variables, which represent the three positions that a feature can appear in a rule (left context, target, and right context) and the two values it can take on (+ and −). We denote these variables by $f_p^v$ for $v$ in $V = \{+,-\}$ and $p$ in $P = \{l,t,r\}$.

Our evaluation relation takes the form $\varphi(c,u,\ell) \triangleq \text{matches}(c,u) \Leftrightarrow \ell$, where matches is a relation specifying whether the condition indexed by $c$ matches the trigram $u$. The matches relation is further defined as follows:

$$\text{matches}(c,u) \triangleq \bigwedge_{(f,v,p)\in F\times V\times P} f_p^v \Rightarrow (u_{p,f} = v)$$

where $u_{p,f}$ is the value of feature $f$ at position $p$ in trigram $u$.

### 3.6 Inductive Bias

In addition to being consistent with the data, we also want the condition to minimize the objective function (3). We encode the objective function as

$$w_s s(c) + l(c),$$

where $s(c)$ encodes the simplicity of the condition indexed $c$ (its size), $l(c)$ encodes the likelihood, and $w_s$ is a model hyperparameter which determines the relative importance of simplicity.

The challenge is to encode the likelihood term in a solver-friendly way. To count the number of models of $|\mathcal{C}_R|$, we observe that $|\mathcal{C}_R| = |\mathcal{C}_R^l||\mathcal{C}_R^t||\mathcal{C}_R^r|$, *i.e.* we can independently count the models of the target, and the left and right contexts, so

$$l(c) \triangleq N^+ \sum_{p\in P} \log(|\mathcal{C}_R^p|)$$

We also observe that $|\mathcal{C}_R^p|$ can be encoded efficiently using a constraint whose size is *linear* in the size of the phone inventory $\Phi$:

$$|\mathcal{C}_R^p| \triangleq \sum_{a\in\Phi} \text{if} \Big( \bigwedge_{\substack{(f,v)\in F\times V \\ a_f \neq v}} \neg f_p^v \Big) \text{ then } 1 \text{ else } 0$$

Finally, as the solver does not support logarithms, we encode log using a lookup table. This is tractable, since we only need to evaluate the log of each $|\mathcal{C}_R^p|$, which is at most the size of our inventory, roughly 100 phones.

### 3.7 Current limitations

SYPHON currently leverages three simplifying assumptions about rules for domain-specific problem decomposition and SMT encoding, which are crucial to making learning computationally tractable.

***Conjunctive conditions.*** Rule conditions are conjunctions of equalities over feature values, and each rule has a unique change. We can thus decompose the learning process into change inference and condition inference: change inference greedily groups all observed edits into changes, and from then on we assume that each change uniquely corresponds to a rule.

***Local context.*** The condition of each rule is only a function of the target phone and two surrounding phones. This allows us to encode condition inference as learning a formula over *trigrams* of phones, which has a compact encoding as SMT constraints.

***Rule interaction.*** One rule's change does not create conditions for another. This allows us to perform condition inference for each rule independently.

Many attested patterns in real languages go beyond these limitations. We believe that it is possible to lift these restrictions, and still leverage the structure of conditional rewrite rules to retain most of the benefits of our problem decomposition. We leave this extension to future work.

## 4 Data

We evaluate our system on two broad categories of datasets: lexical databases and textbook problems.

### 4.1 Lexical Databases

We use large lexical databases to investigate two (morpho)phonological processes in English: flapping (6457 rows) and regular verb inflections (2756 rows). We process the CMU pronouncing dictionary (Weide, 2014) to create underlying and surface form pairs exemplifying flapping, as in Gildea and Jurafsky (1996). For verb inflections, we combine morphological information extracted from CELEX-2 (Baayen et al., 1993) with CMU transcriptions to create a database of regular verbs, where each row of the database contains the third-person singular present tense wordform and past tense wordform for a given verb. For both datasets we have gold standard solutions for both rule sets and underlying forms, provided by one of our coauthors, who is a phonologist.

### 4.2 Textbook Problems

For this category, we curated a set of 34 problems from phonology textbooks (Gussenhoven and Jacobs, 2017; Odden, 2005; Roca and Johnson, 1999) by selecting problems with local, non-interacting rules. For each problem, the input data set is tailored (by the textbook author) to illustrate a particular phonological process, and contains 20-50 surface forms. For all of these problems we have gold standard solutions, either provided with the textbook or inferred by a phonologist. Gold standard solutions range in complexity from one to two rules. Out of the 34, 21 problems are shared with (Ellis et al., 2019), which we use as the baseline for inference times.

Following the textbooks, these problems are further subdivided into 10 *matrix problems*, 20 *alternation problems*, and 4 *supervised problems*. The matrix problems follow the format presented in Sec. 2. The alternation and supervised problems are easier, as we are given more information about the underlying form. For alternation problems, we are essentially given a set of choices for what the underlying form might be, and for supervised problems the underlying form is given exactly. These problems include examples of phones in complementary distribution. Our problem decomposition allows us to switch out underlying forms inference to handle different kinds of input. According to this classification, the flapping lexical database is an alternation problem and verbs is a matrix problem.

## 5 Experiments

We evaluate our system on the two categories of data sets discussed in Sec. 4. We split the 34 textbook problems into 24 development and 10 test problems. Our system has several free parameters (most importantly, the simplicity weight $w_s$). These were hand-tuned on all of the data except the test problems. For the test problems, we only added missing sounds to the inventory as needed. The 10 test problems are all alternation problems. We leave for future work the investigation of these hyperparameter settings on new matrix problems.

### 5.1 Lexical Database Experiments

We evaluate our system on two large English datasets, one demonstrating flapping, and the other verbs. For each dataset, we learn a rule set from 20, 50 and 100

| | Accuracy | | Rule Match | | | | UF |
|---|---|---|---|---|---|---|---|
| | | | Precision | | Recall | | |
| | SP | SP- | SP | SP- | SP | SP- | |
| Flap 20 | 76 | 52 | 50 | 66 | 31 | 25 | 100 |
| Flap 50 | 93 | 79 | 86 | 71 | 86 | 71 | 100 |
| Flap 100 | 100 | 79 | 100 | 71 | 100 | 71 | 100 |
| Verb 20 | 86 | 73 | 48 | 42 | 83 | 61 | 100 |
| Verb 50 | 88 | 78 | 52 | 50 | 92 | 80 | 100 |
| Verb 100 | 95 | 81 | 62 | 58 | 100 | 82 | 100 |

Table 4: Accuracy results for the English flapping and verbs corpora data sets on 20, 50 and 100 training examples. SYPHON (SP ) and SYPHON- (SP- ) are two variants of our model, with and without likelihood, resp. Accuracy reports the generalization accuracy on unseen inputs, rule match and UF indicate how well the inferred rule and underlying form resp. match the gold standard.

| | Data Set | Inferred Rule |
|---|---|---|
| 1 | Flap 20 | $\begin{bmatrix}+\text{cor}\\-\text{voi}\end{bmatrix} \rightarrow [+\text{approx}] \,/\, [+1\text{stress}]\_$ |
| 2 | Flap 50 | $\begin{bmatrix}+\text{cor}\\-\text{voi}\\-\text{del. rel.}\end{bmatrix} \rightarrow \begin{bmatrix}+\text{voi}\\+\text{approx}\end{bmatrix} \,/\, [+\text{stress}]\_[+\text{syll}]$ |
| 3 | Flap 100 | $\begin{bmatrix}+\text{ant}\\-\text{voi}\\-\text{del. rel.}\end{bmatrix} \rightarrow \begin{bmatrix}+\text{voi}\\+\text{approx}\end{bmatrix} \,/\, [+\text{stress}]\_[+\text{syll}]$ |
| 4 | Russian | $[-\text{son}] \rightarrow [-\text{voi}] \,/\, \_\#$ |
| 5 | Scottish | $[+\text{syll}] \rightarrow [+\text{long}] \,/\, \_ \begin{bmatrix}+\text{cons}\\+\text{voi}\\+\text{cont}\end{bmatrix}$ |
| 6 | Korean | $\begin{bmatrix}-\text{cont}\\-\text{voi}\end{bmatrix} \rightarrow \begin{bmatrix}-\text{c.g.}\\-\text{s.g.}\end{bmatrix} \,/\, \_[+\text{c.g.}]$ |
| 7 | Farsi | $\begin{bmatrix}-\text{cont}\\+\text{dors}\end{bmatrix} \rightarrow \emptyset \,/\, [+\text{ATR}]\_\#$ |
| 8 | Hungarian | $[-\text{son}] \rightarrow [\alpha\text{voi}] \,/\, \_ \begin{bmatrix}\alpha\text{voi}\\-\text{del. rel.}\end{bmatrix}$ |
| 9 | Kishambaa | $[+\text{nas}] \rightarrow [-\text{voi}] \,/\, \_[+\text{s.g.}]$ |
| 10 | Persian | $\begin{bmatrix}+\text{approx}\\-\text{voi}\end{bmatrix} \rightarrow [+\text{voi}] \,/\, \_[-\text{nas}]$ |
| 11 | Ganda | $[+\text{lat}] \rightarrow [+\text{cont}] \,/\, \begin{bmatrix}-\text{lab}\\+\text{ATR}\end{bmatrix} \_$ |
| 12 | Limbu | $\begin{bmatrix}+\text{back}\\+\text{syll}\end{bmatrix} \rightarrow [+\text{rnd}] \,/\, \begin{bmatrix}+\text{lab}\\-\text{cont}\end{bmatrix} \_$ |
| 13 | Kongo | $\begin{bmatrix}-\text{son}\\+\text{cor}\end{bmatrix} \rightarrow \begin{bmatrix}-\text{ant}\\+\text{dist}\\+\text{strid}\end{bmatrix} \,/\, \_ \begin{bmatrix}-\text{rnd}\\+\text{high}\end{bmatrix}$ |

Table 5: Selected inferred rules: English flapping trained on 20, 50 and 100 examples (1–3); textbook development problems (4–8); textbook test problems (9–13).

data points, and evaluate its accuracy on the remaining data. We also perform a syntactic comparison of the rule set against the gold standard rules, which we report as average precision and recall among the sets of features in the two rules. Finally, we compare the latent underlying forms we inferred for each problem to the known correct underlying forms. Tab. 4 summarizes the results. Tab. 5 (rows 1–3) shows the actual rules inferred on the three flapping training sets.

To examine the importance of likelihood in our system, we repeat this experiment for a variant of our system SYPHON-, which does not consider likelihood and simply optimizes our simplicity prior. As the number of data points increases, the effect of the likelihood grows, and so for SYPHON the recall compared to the gold standard quickly climbs. By contrast, the recall of SYPHON- plateaus, which shows that likelihood is indeed important for finding good rules.

### 5.2 Textbook Problem Experiments

We evaluate the textbook problems under the following three experimental conditions. First, to evaluate the generalization accuracy for unseen inputs, for each of the problems, we hold out a randomly sampled 20% of the data. We then learn a rule set on the remaining data, and validate it against the held out data. We repeat this process three times, and report the average accuracy for each class of problems in Tab. 6. We also evaluate syntactic accuracy of the rules and of underlying forms, in the same way as for the lexical databases. Additionally, we evaluate our system on 10 test problems, which were left out entirely when tuning the system hyperparameters. We report the same metrics for these problems. Tab. 5 shows inferred rules for selected development problems (rows 4–8) and test problems (rows 9–13).

The accuracy of our inferred rules and underlying forms is 100% for all textbook problems. This is not surprising: the combination of hard constraints and a restrictive DSL makes inferring incorrect rules or underlying forms very difficult. More interesting is the syntactic comparison to the gold standard. This measure is intended to estimate how well the rules SYPHON produces match phonologists' intuition. The results in Tab. 6 confirm that without the likelihood term, inference tends to exclude important features from the rule condition: the recall for held out problems goes down by 21%.

Finally, we compare inference times of SYPHON with the prior work of Ellis et al. (2019), which is also based on constraint-based program synthesis but does not perform problem decomposition, instead using the global encoding outlined in Sec. 3.2. As shown in Tab. 7, the decomposition makes SYPHON at least *two orders of magnitude* faster, with an average inference time of just 30 seconds for matrix problems, thus enabling phonologists to use the tool interactively.

| | Accuracy | | Rule Match | | | | UF |
|---|---|---|---|---|---|---|---|
| | | | Precision | | Recall | | |
| | SP | SP- | SP | SP- | SP | SP- | SP |
| MAT | 100 | 100 | 70 | 69 | 77 | 69 | 100 |
| ALT | 100 | 100 | 66 | 61 | 71 | 62 | 100 |
| SUP | 100 | 100 | 63 | 60 | 71 | 64 | - |
| TEST | 100 | 100 | 54 | 52 | 61 | 48 | 100 |

Table 6: Accuracy of textbook problems. We use (-) for supervised problems without underlying form inference.

| | Inference Time (secs) | | |
|---|---|---|---|
| | SYPHON | Baseline | Speedup |
| MAT | 30.0 | 3100 | 124.6 |
| ALT | 10.7 | N/A | N/A |
| SUP | 5.3 | 6333 | 378.3 |
| TEST | 8.3 | N/A | N/A |

Table 7: Comparison of the inference times of textbook problems with baseline. We report the median execution times and geometric mean of the speedups. N/A indicates examples where baseline results are unavailable.

## 6 Related Work

Learning (morpho-)phonology is a rich and active area of research; in this overview, we focus on approaches that share our goal of inferring fully interpretable models of phonological processes.

Most closely related to ours is the work of Ellis et al. (2015) and its (unpublished) follow-up (Ellis et al., 2019) on using program synthesis to infer phonological rules. As mentioned above, the main difference is that SYPHON is two orders of magnitude faster than their system thanks to a novel decomposition and efficient SMT encoding. On the other hand, we impose extra restrictions on the hypothesis space (*i.e.* we only support local rules), which means that SYPHON is unable to solve some of the harder textbook problems that Ellis et al. (2019) can solve. In addition, Ellis et al. (2019) propose a method for inducing phonological representations which are universal across languages.

Beyond program synthesis, Rasin et al. (2017) use a comparable description length-based approach to unsupervised joint inference of underlying phonological forms and rewrite rule representations of phonological processes, but use a genetic algorithm to find approximate solutions. Gildea and Jurafsky (1996) and Chandlee et al. (2014) discuss supervised learning of restricted classes of finite-state transducer representations of several phonological processes (including English flapping). To date, such work either requires thousands of training observations (Gildea and Jurafsky, 1996) or has used abstracted and greatly simplified symbol inventories and training data (Chandlee et al., 2014).

Hayes and Wilson (2008), Goldsmith and Riggle (2012), and Futrell et al. (2017) propose different methods for learning probabilistic models of phonotactics, which represent gradient co-occurrence restrictions between surface segments within a word. Unlike the current implementation of SYPHON, these models include representational structures that enable them to capture certain non-local phenomena. However, because these models focus on phonotactics, they do not infer underlying forms or rules which relate underlying forms to surface forms.

Finally, much work has focused on learning representations of phonological processes as mappings that minimally violate a set of ranked or weighted constraints (Prince and Smolensky, 2004; Legendre et al., 1990), but such work has generally taken the constraint definitions as given and focused on learning rankings or weights (see *e.g.* Goldwater and Johnson, 2003; Tesar and Smolensky, 2000; Boersma and Hayes, 2001), with some exceptions (Doyle et al., 2014; Doyle and Levy, 2016).

## 7 Conclusion

We have presented a new approach to learning fully interpretable phonological rules from sets of related surface forms. We have shown that our approach produces rules that largely match linguists' intuition from a handful of examples and within minutes. The contributions of this paper are a novel decomposition of the global inference problem into three local problems, as well as an encoding of these problems into constraints that can be efficiently solved by an SMT solver.

## References

Adam Albright and Bruce Hayes. 2002. Modeling English past tense intuitions with minimal generalization. In *Proceedings of the 2002 Workshop on Morphological Learning, Association for Computational Linguistics*.

Rajeev Alur, Rastislav Bodík, Garvit Juniwal, Milo M. K. Martin, Mukund Raghothaman, Sanjit A. Seshia, Rishabh Singh, Armando Solar-Lezama, Emina Torlak, and Abhishek Udupa. 2013. Syntax-guided synthesis. In *Formal Methods in Computer-Aided Design, FMCAD 2013*, pages 1–8.

Rajeev Alur, Arjun Radhakrishna, and Abhishek Udupa. 2017. Scaling enumerative program synthesis via

divide and conquer. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 319–336. Springer.

R. Harald Baayen, Richard Piepenbrock, and H van Rijn. 1993. The CELEX lexical database on CD-ROM.

Nikolaj Bjørner, Anh-Dung Phan, and Lars Fleckenstein. 2015. $\nu$z - an optimizing SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 194–199.

Paul Boersma and Bruce Hayes. 2001. Empirical Tests of the Gradual Learning Algorithm. *Linguistic Inquiry*, 32(1):45–86.

Jane Chandlee, Rémi Eyraud, and Jeffrey Heinz. 2014. Learning strictly local subsequential functions. *Transactions of the Association for Computational Linguistics*, 2:491–504.

Noam Chomsky and Morris Halle. 1968. *The Sound Pattern of English*. Studies in language. Harper & Row.

Leonardo Mendonça de Moura and Nikolaj Bjørner. 2008. Z3: an efficient SMT solver. In *TACAS*, volume 4963 of *LNCS*, pages 337–340. Springer.

Gabriel Doyle, Klinton Bicknell, and Roger Levy. 2014. Nonparametric Learning of Phonological Constraints in Optimality Theory. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1094–1103.

Gabriel Doyle and Roger Levy. 2016. Data-driven learning of symbolic constraints for a log-linear model in a phonological setting. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2217–2226.

Greg Durrett and John DeNero. 2013. Supervised learning of complete morphological paradigms. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*.

Kevin Ellis, Adam Albright, Armando Solar-Lezama, Joshua B. Tenenbaum, and Timothy J. O'Donnell. 2019. Synthesizing theories of human language with Bayesian program induction. In Prep.

Kevin Ellis, Daniel Ritchie, Armando Solar-Lezama, and Josh Tenenbaum. 2018. Learning to infer graphics programs from hand-drawn images. In *Advances in Neural Information Processing Systems 31*, pages 6059–6068. Curran Associates, Inc.

Kevin Ellis, Armando Solar-Lezama, and Josh Tenenbaum. 2015. Unsupervised learning by program synthesis. In *Advances in neural information processing systems*, pages 973–981.

Richard Futrell, Adam Albright, Peter Graff, and Timothy J. O'Donnell. 2017. A generative model of phonotactics. *Transactions of the Association for Computational Linguistics*, 5:73–86.

Daniel Gildea and Daniel Jurafsky. 1996. Learning bias and phonological-rule induction. *Computational Linguistics*, 22(4):497–530.

John Goldsmith and Jason Riggle. 2012. Information theoretic approaches to phonological structure: the case of Finnish vowel harmony. *Natural Language & Linguistic Theory*, 30(3):859–896.

Sharon Goldwater and Mark Johnson. 2003. Learning OT Constraint Rankings Using a Maximum Entropy Model. *Proceedings of the Stockholm Workshop on Variation within Optimality Theory*, pages 111–120.

Noah D Goodman, Joshua B Tenenbaum, Jacob Feldman, and Thomas L Griffiths. 2008. A rational analysis of rule-based concept learning. *Cognitive science*, 32(1):108–154.

Dan Gusfield. 1997. *Algorithms on Strings, Trees and Sequences*. Cambridge University Press.

Carlos Gussenhoven and Haike Jacobs. 2017. *Understanding Phonology*. Routledge.

Bruce Hayes and Colin Wilson. 2008. A maximum entropy model of phonotactics and phonotactic learning. *Linguistic Inquiry*, 39(3):379–440.

Géraldine Legendre, Yoshiro Miyata, and Paul Smolensky. 1990. Harmonic Grammar – A formal multi-level connectionist theory of linguistic well-formedness: Theoretical foundations. Technical Report ICS # 90-5, CU-CS-465-90, University of Colorado.

David Odden. 2005. *Introducing Phonology*. Cambridge University Press.

José Oncina, Pedro García, and Enrique Vidal. 1993. Learning Subsequential Transducers for Pattern Recognition Interpretation Tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(5):448–458.

Alan Prince and Paul Smolensky. 2004. *Optimality Theory: Constraint interaction in generative grammar*. Wiley-Blackwell.

Ezer Rasin, Iddo Berger, Nur Lan, and Roni Katzir. 2017. Acquiring opaque phonological interactions using Minimum Description Length. In *Supplemental Proceedings of the 2017 Annual Meeting on Phonology*.

Iggy Roca and Wyn Johnson. 1999. *A Workbook in Phonology*. Blackwell.

Rohit Singh, Venkata Vamsikrishna Meduri, Ahmed K. Elmagarmid, Samuel Madden, Paolo Papotti, Jorge-Arnulfo Quiané-Ruiz, Armando Solar-Lezama, and Nan Tang. 2017. Synthesizing entity matching rules by examples. *PVLDB*, 11(2):189–202.

Armando Solar-Lezama. 2013. Program sketching. *International Journal on Software Tools for Technology Transfer*, 15(5-6):475–495.

Bruce Tesar and Paul Smolensky. 2000. *Learnability in Optimality Theory*. MIT Press.

Abhinav Verma, Vijayaraghavan Murali, Rishabh Singh, Pushmeet Kohli, and Swarat Chaudhuri. 2018. Programmatically interpretable reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning*, pages 5052–5061.

R. L Weide. 2014. The CMU pronouncing dictionary. Release 0.7b.

Yunhui Zheng, Vijay Ganesh, Sanu Subramanian, Omer Tripp, Murphy Berzish, Julian Dolby, and Xiangyu Zhang. 2017. Z3str2: an efficient solver for strings, regular expressions, and length constraints. *Formal Methods in System Design*, 50(2-3):249–288.

# Detect Camouflaged Spam Content via StoneSkipping: Graph and Text Joint Embedding for Chinese Character Variation Representation

**Zhuoren Jiang[1]*, Zhe Gao[2]*, Guoxiu He[3], Yangyang Kang[2], Changlong Sun[2], Qiong Zhang[2], Luo Si[2], Xiaozhong Liu[4]†**

[1] School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China
[2]Alibaba Group, Hangzhou & Sunnyvale & Seattle, China & USA
[3]School of Information Management, Wuhan University, Wuhan, China
[4]School of Informatics, Computing and Engineering, IUB, Bloomington, USA
jiangzhr3@mail.sysu.edu.cn,
{gaozhe.gz,yangyang.kangyy,qz.zhang,luo.si}@alibaba-inc.com,
guoxiu.he@whu.edu.cn, changlong.scl@taobao.com,
liu237@indiana.edu

## Abstract

The task of Chinese text spam detection is very challenging due to both glyph and phonetic variations of Chinese characters. This paper proposes a novel framework to jointly model Chinese variational, semantic, and contextualized representations for Chinese text spam detection task. In particular, a Variation Family-enhanced Graph Embedding (VFGE) algorithm is designed based on a Chinese character variation graph. The VFGE can learn both the graph embeddings of the Chinese characters (local) and the latent variation families (global). Furthermore, an enhanced bidirectional language model, with a combination gate function and an aggregation learning function, is proposed to integrate the graph and text information while capturing the sequential information. Extensive experiments have been conducted on both SMS and review datasets, to show the proposed method outperforms a series of state-of-the-art models for Chinese spam detection.

## 1 Introduction

Chinese orchestrates over tens of thousands of characters by utilizing their morphological information, e.g., pictograms, simple/compound ideograms, and phono-semantic compounds (Norman, 1988). Different characters, however, may share the similar glyph and phonetic "root". For instance, from glyph perspective, character "裸 (naked)" looks like "课 (course)" (homographs), while from phonetic viewpoint, it shares the similar pronunciation with "锣 (gong)" (homophones). The form of variations can also be compounded, for instance, "账 (account)"and "帐 (curtain)" have the similar structure and pronunciation (homonyms). Unfortunately, in the context of spam detection, as shown in Figure 1, spammers are able to take advantage of these variations to escape from the detection algorithms (Jindal and Liu, 2007). For instance, in the e-commerce ecosystem, variation-based Chinese spam mutations thrive to spread illegal, misleading, and harmful information[1]. In this study, we propose a novel problem - Chinese Spam Variation Detection (CSVD), a.k.a. investigating an effective Chinese character embedding model to assist the classification models to detect the variations of Chinese spam text, which needs to address the following key challenges.

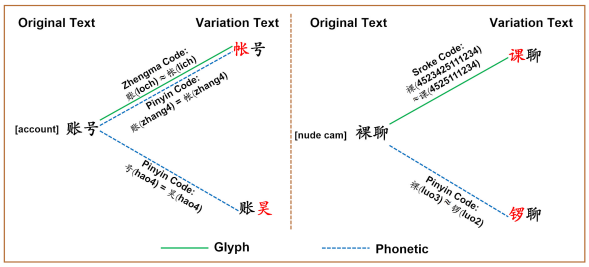


Figure 1: Character Variations in Chinese Spam Texts (the pinyin codes provide phonetic information and the stroke and Zhengma codes provide glyph information).

**Diversity**: the variation patterns of Chinese characters can be complex and subtle, which are difficult to generalize and detect. For instance, in the experimental dataset, one Chinese character can have 297 (glyph and phonetic) variants averagely and 2,332 maximally. The existing keyword based spam detection approaches, e.g., (Ntoulas et al., 2006), can hardly address this problem. **Sparseness, Zero-shot, and Dynamics**:

*These two authors contributed equally to this research.
†Corresponding author

[1]More detailed information can be found in the experiment section.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6187–6196, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

when competing with classification models, spammers are constantly creating new Chinese characters combinations for spam texts (that can be a "zero/few shot learning" problem (Socher et al., 2013)). The labelling cost can be inevitably high in such dynamic circumstance. Data driven approaches, e.g., (Zhang et al., 2015), will perform poorly for unseen data. **Camouflage**: with the common cognition knowledge of Chinese and the contextual information, users are able to consume the spam information, even when some characters in the content are intentionally mutated into their similar variations (Spinks et al., 2000; Shu and Anderson, 1997). However, the variation-based spam text are highly camouflaged for machines. It is important to propose a novel Chinese character representation learning model that can synthesize character variation knowledge, semantics, and contextualized information.

To address these challenges, we propose a novel solution, **S**tone**S**kipping (**SS**) model to enable Chinese variation representation learning via graph and text joint embedding. SS is able to learn the Chinese character variation knowledge and predict the new variations not appearing in the training set by utilizing sophisticated heterogeneous graph mining method. For a piece of text (a character sequence), with the proposed model, each candidate character can probe character variation graph (like stone bouncing cross the water surface), and explore its glyph and phonetic variation information (like the ripples caused by the stone hitting the water). Algorithmically, a Variation Family-enhanced Graph Embedding (VFGE) algorithm is proposed to extract the heterogeneous Chinese variation knowledge while learning the (local) graph representation of a Chinese character along with the (global) representation of the latent variation families. Finally, an enhanced bidirectional language model, with a combination gate function and an aggregation learning function, is proposed to comprehensively learn the variation, semantic, and sequential information of Chinese characters. To the best of our knowledge, this is the first work to use graph embedding to learn the heterogeneous variation knowledge of Chinese characters for spam detection.

The major contributions of this paper can be summarized as follows:

1. We propose an innovative CSVD problem, in the context of text spam detection, to address the diversity, sparseness, and text camouflage problems.

2. A novel joint embedding SS model is proposed to learn the variational, semantic, and contextual representations of Chinese characters. SS is able to predict unseen variations.

3. A Chinese character variation graph is constructed for encapsulating the glyph and phonetic relationships among Chinese characters. Since the graph can be potentially useful for other NLP tasks, we share the graph/embeddings to motivate further investigation.

4. Through the extensive experiments on both SMS and review datasets[2], we demonstrate the efficacy of the proposed method for Chinese spam detection. The proposed method outperforms the state-of-the-art models.

## 2 Related Work

**Neural Word Embeddings**. Unlike traditional word representations, low-dimensional distributed word representations (Mikolov et al., 2013; Pennington et al., 2014) are able to capture in-depth semantics of text content. More recently, ELMo (Peters et al., 2018) employed learning functions of the internal states of a deep bidirectional language model to generate the character embeddings. BERT (Devlin et al., 2018) utilized bidirectional encoder representations from transformers (Vaswani et al., 2017) and achieved improvements for multiple NLP tasks. However, all the prior models only focused on learning the context, whereas the text variation was ignored. Moreover, CSVD problem can be different from other NLP tasks: the intentional character mutations and unseen variations (zero-shot learning (Socher et al., 2013)) can threaten prior models' performances.

**Chinese Word and Sub-word Embeddings**. A number of studies explored Chinese representation learning methodologies. CWE (Chen et al., 2015) learned the character and word embeddings to improve the representation performance. GWE (Su and Lee, 2017) introduced the features extracted from the images of traditional Chinese characters. JWE (Yu et al., 2017) used deep learning to generate character embedding based on an extended radical collection. Cw2vec (Cao et al., 2018) in-

[2]In order to help other scholars reproduce the experiment outcome, we will release the datasets via GitHub (https://github.com/Giruvegan/stoneskipping)

vestigated Chinese character as a sequence of n-gram stroke order to generate its embedding. Although these models had considered the nature of Chinese characters, they only utilized glyph features while the phonetic information was ignored. In CSVD problem, the forms of variations can be heterogeneous, and a single kind of features cannot cover all mutation patterns. More importantly, all these models are not designed for spam detection, and the task-oriented model should be able to highlight the most important variations for spam text.

**Graph Embedding**. Graph (a.k.a. information network) is a natural data structure for characterizing the multiple relationships between the objects. Recently, multiple graph embedding algorithms are proposed to learn the low dimensional feature representations of vertexes in graphs. DeepWalk (Perozzi et al., 2014) and Node2vec (Grover and Leskovec, 2016) are random walk based models. LINE (Tang et al., 2015) modeled 1st and 2nd order graph neighbourhood. Meanwhile, metapath2vec++ (Dong et al., 2017) was designed for heterogeneous graph embedding with human defined metapath rules. HEER (Shi et al., 2018) is a recent state-of-the-art heterogeneous graph embedding model. Though the techniques utilized in these models are different, most existing graph embedding models focus more on local graph structure representation, e.g., modelling of a fixed-size graph neighbourhood. CSVD problem requires graph embedding conducted from a more global perspective, to characterize comprehensive variation patterns.

**Spelling Correction**. Spelling correction may serve as an alternative to address CSVD problem, e.g., using dictionary-based (Yeh et al., 2014) or language model-based method (Yu and Li, 2014) to restore the content variations to their regular format. However, because spammers intentionally mutate the spam text to escape from the detection model, training data sparseness and dynamics may challenge this approach.

## 3 StoneSkipping Model

Figure 2 depicts the proposed SS model. There are three core modules in SS: a **Chinese character variation graph** to host the heterogeneous variation information; a **variation family-enhanced graph embedding** for Chinese character variation knowledge extraction and graph representation learning; an **enhanced bidirectional language model** for joint representation learning. In the remaining of this section, we will introduce them in detail.

### 3.1 Chinese Character Variation Graph

A Chinese character variation graph[3] can be denoted as $G = (C, R)$. $C$ denotes the Chinese character set, and each character is represented as a vertex in $G$. $R$ denotes the variation relation (edge) set, and edge weight is the similarity of two characters given the target relation (variation) type. To accurately characterize both phonetic and glyph information of Chinese character, we utilize three different encoding methods:

**Pinyin** system provides phonetic-based information, which is widely used for representing the pronunciations of Chinese characters (Chen and Lee, 2000). In this system, each Chinese character has one syllable which consists of three components: an initial (consonant), a final (vowel), and a tone. There are four types of tones in Modern Standard Mandarin Chinese. Different tones with the same syllable can have different meanings. For instance, the pinyin code of "裸 (naked)" is "luo3" and "锣 (gong)" is 'luo2". The pinyin-based variation similarity is calculated based on their pinyin syllables with tones[4].

**Stroke** is a basic glyph pattern for writing Chinese character (Cao et al., 2018). All Chinese characters are written in a certain stroke order and can be represented as a stroke code, e.g., the stroke code of "裸 (naked)" is "4523425111234" and '课 (course)" is "4525111234". The stroke-based variational similarity is calculated based on *longest common substring* and *longest common subsequence* metrics[4].

**Zhengma** is another important means for glyph character encoding, which encodes character at radical level (Yu et al., 2017). For instance, the Zhengma code of "裸 (naked)" is "WTKF" and '课 (course)" is "SKF". The Zhengma-based variational similarity is calculated based on the *Jaccard Index* metric[4].

Unlike previous works (Cao et al., 2018; Yu et al., 2017) only employ one kind of glyph-based information, we utilize two different glyph pat-

[3]We will release Chinese character variation graph via GitHub (https://github.com/Giruvegan/stoneskipping).

[4]Because of the space limitation, the detailed operations of relation generation will be provided on https://github.com/Giruvegan/stoneskipping.

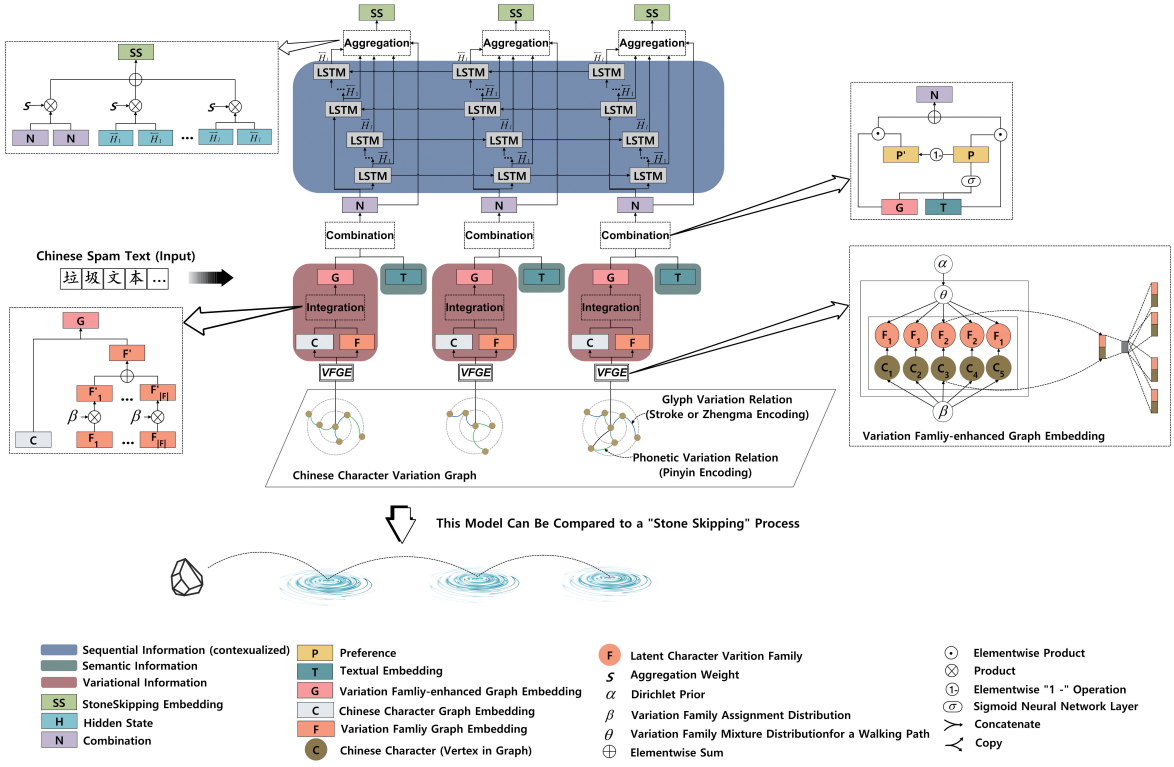


Figure 2: An Illustration of "StoneSkipping" Framework

terns (stroke and Zhengma) to encode the Chinese character. Because these two patterns can characterize Chinese characters from different internal structural levels, and complement each other to enable an enhanced glyph representation learning. Furthermore, the pinyin encoder provides phonetic information. The constructed character variation graph integrates these three kinds of variation relations, which can be significant for camouflaged spam detection.

### 3.2 Variation Family-enhanced Graph Embedding

While the variation graph can provide comprehensive knowledge of Chinese character variations, efforts need to be made to address these two problems: (1) the variation patterns can be very flexible, and the compounded (long-range) variation information transfer may exist. Therefore, short-range (local) graph information, e.g., character vertex's neighbors, may be insufficient for spam detection. Meanwhile, it is impractical to exhaust all the possible variation patterns. (2) To oblige users to consume the text content, spammers cannot make the variation patterns to be too complex/confusing, they usually focus on the most sensitive words in a spam message. Hence, some random infrequent variation patterns could be "noisy" for CSVD while polluting the detection outcomes.

**Latent Character Variation Family**. In this study, we propose a VFGE model to address these problems. As depicted in Figure 2, in VFGE model, we introduce a set of latent variables "*character variation family*" $F = \{F_1, ..., F_{|F|}\}$ at a graph schema (global) level to capture the critical information for spam detection. Each $F_i$ is defined as *a distribution of characters*, which aims to estimate the global frequent variation dependencies in $G$. By learning $F$, VFGE is able to highlight the useful variations, eliminate the noisy patterns, and predict the unseen variation forms w.r.t. the spam detection task.

**Random Walk based Character Family Representation Co-Learning** . VFGE is a random walk based graph embedding model, and we employ a hierarchical random walk strategy (Jiang et al., 2018b) on $G$ to generate the optimized walking paths (character vertex sequences) for each character. The model can sample the most possible variation context vertexes for each character. Based on generated walking paths, VFGE executes the following two processes iteratively:

(1) **Family Assignment**. By leveraging both lo-

cal context and global family distributions, we assign a discrete family for each character vertex in a particular walking path to form a character-family pair $\langle C, F\rangle$. As shown in Figure 2, we assume different walking paths tend to emerge various character variation patterns which can be represented as mixtures over latent variation families. Given a character $C_i$ in a $path$, $C_i$ has a higher chance to be assigned to a dominant family $F_i$. The assignment probability can be calculated as:

$$\begin{aligned} &Pr(F_i|C_i, path) \\ \propto &Pr(C_i, F_i, path) \\ = &Pr(path)Pr(F_i|path)Pr(C_i|F_i) \end{aligned} \tag{1}$$

As depicted in Figure 2, $\alpha$ is the parameter of the Dirichlet prior on the per-path family distributions ($Pr(path)$); $\beta$ is the family assignment distribution ($Pr(C|F)$); and $\theta$ is the family mixture distribution for a walking path ($Pr(F|path)$). The distribution learning can be considered as a Bayesian inference problem, and we use Gibbs sampling (Porteous et al., 2008) to address this problem.

(2) **Character-Family Representation Co-Learning**. Given the assigned character-family pairs, the proposed method aims to obtain the representations of character $C$ and latent variation family $F$ by mapping them into a low-dimensional space $\mathbb{R}^d$ ($d$ is a parameter specifying the number of dimensions). Motivated by (Liu et al., 2015), we propose a novel representation learning method to optimize characters and families ***separately*** and ***simultaneously***.

The objective is defined to maximize the following log probability:

$$\mathcal{L} = \max_f \sum_{C_i \in C} \sum_{C_j \in \mathbb{N}(C_i)} log Pr(\langle C_j, F_j\rangle \,|\mathbf{C_i^{F_i}}) \tag{2}$$

We use $f(\cdot)$ as the embedding function. $\mathbf{C_i} = f(C_i)$ represents the character graph embedding and $\mathbf{F_i} = f(F_i)$ represents the family graph embedding. $\mathbf{C_i^{F_i}}$ denotes the concatenation of $\mathbf{C_i}$ and $\mathbf{F_i}$, whereas $\mathbb{N}(C_i)$ is $C_i$'s neighborhood (context). As Figure 2 shows, the feature representation learning method is an upgraded version of the skip-gram architecture. Compared with merely using the target vertex $C_i$ to predict context vertexes in original skip-gram model (Mikolov et al., 2013), the proposed approach employs the character-family pair $\langle C_i, F_i\rangle$ to predict context character-family pairs. From variation viewpoint, character vertex's context will encapsulate both local (vertex) and global (variation family) information. Hence, the learned representations are able to comprehensively preserve the variation information in $G$.

$Pr(\langle C_j, F_j\rangle \,|\mathbf{C_i^{F_i}})$ is modeled as a softmax function:

$$Pr(\langle C_j, F_j\rangle \,|\mathbf{C_i^{F_i}}) = \frac{exp(\mathbf{C_j^{F_j}} \cdot \mathbf{C_i^{F_i}})}{\sum_{C_k \in C} exp(\mathbf{C_k^{F_k}} \cdot \mathbf{C_i^{F_i}})} \tag{3}$$

Stochastic gradient ascent is used for optimizing the model parameters of $f$. Negative sampling (Mikolov et al., 2013) is applied for optimization efficiency. Note that, the parameters of each character embedding and family embedding are shared over all the character-family pairs, which, as suggested in (Liu et al., 2015), can address the training data sparseness problem and improve the representation quality.

**Family-enhanced Embedding Integration**. As shown in Figure 2, the family-enhanced character graph embedding can be calculated as:

$$\mathbf{G_i} = \left[\mathbf{C_i}, \sum_{F_j \in F} Pr(F_j|C_i)\mathbf{F_j}\right] \tag{4}$$

where $\mathbf{G_i}$ is family-enhanced graph embedding for $C_i$, and $[\cdot]$ is concatenating operation. $Pr(F_j|C_i)$ can be inferred from family assignment distribution $\beta$.

### 3.3 Enhanced Bidirectional Language Model

As shown in Figure 2, SS model utilizes an enhanced bidirectional language model to jointly learn variation, semantic and contextualized representation of Chinese character.

**Combination Gate Function**. This gate function is utilized for combining the variation and semantic representations, which is the input function for bidirectional language model. The formulations of the gate function are listed as follows:

$$\begin{aligned} \mathbf{P} &= \sigma(\mathbf{W}_P \cdot [\mathbf{G}, \mathbf{T}] + \mathbf{b}_P) \\ \mathbf{N} &= (\mathbf{P} \odot \mathbf{T}) + ((\mathbf{1} - \mathbf{P}) \odot \mathbf{G}) \end{aligned} \tag{5}$$

$\mathbf{P} \in \mathbb{R}^d$ is the preference weights for controlling the contributions from $\mathbf{G} \in \mathbb{R}^d$ (variation graph embedding) and $\mathbf{T} \in \mathbb{R}^d$ (Skip-Gram textual embedding). $\mathbf{W}_P \in \mathbb{R}^{2d \times d}$. $\mathbf{N} \in \mathbb{R}^d$ is the combination representation. $\odot$ is elementwise product, and $+$ is elementwise sum.

| Group | Model | SMS | | Review | |
|---|---|---|---|---|---|
| | | Accuracy | F1 Score | Accuracy | F1 Score |
| Text | **Skipgram** (Mikolov et al., 2013) | 0.807 | 0.765 | 0.693 | 0.560 |
| | **GloVe** (Pennington et al., 2014) | 0.732 | 0.637 | 0.707 | 0.600 |
| | **ELMo** (Peters et al., 2018) | 0.786 | 0.747 | 0.755 | 0.647 |
| Chinese | **CWE** (Chen et al., 2015) | 0.751 | 0.674 | 0.780 | 0.726 |
| | **GWE** (Su and Lee, 2017) | 0.505 | 0.426 | 0.778 | 0.718 |
| | **JWE** (Yu et al., 2017) | 0.770 | 0.707 | 0.738 | 0.646 |
| | **Cw2vec** (Cao et al., 2018) | 0.800 | 0.753 | 0.724 | 0.618 |
| Graph | **DeepWalk** (Perozzi et al., 2014) | 0.836 | 0.804 | 0.738 | 0.638 |
| | **LINE** (Tang et al., 2015) | 0.821 | 0.783 | 0.764 | 0.695 |
| | **Node2vec** (Grover and Leskovec, 2016) | 0.835 | 0.802 | 0.792 | 0.736 |
| | **M2V**$_{\mathbf{Max}}$ (Dong et al., 2017) | 0.838 | 0.807 | 0.790 | 0.740 |
| | **HEER** (Shi et al., 2018) | 0.723 | 0.617 | 0.771 | 0.708 |
| Correction | **Pycorrector** (Yu and Li, 2014) | 0.782 | 0.727 | 0.688 | 0.549 |
| Comparison | **SS**$_{\mathbf{Graph}}$ | 0.839 | 0.827 | 0.812 | 0.756 |
| | **SS**$_{\mathbf{Naive}}$ | 0.849 | 0.825 | 0.811 | 0.757 |
| | **SS**$_{\mathbf{Original}}$ | **0.851** | **0.832** | **0.854** | **0.822** |

Table 1: Chinese Text Spam Detection Performance Comparison of Different Models

**Aggregation Learning Function**. With the combination representation $\mathbf{N}$ as input, we train a bidirectional language model for capturing the sequential information. There could be multiple layers of forward and backward LSTMs in bidirectional language model. For $k_{th}$ character, $\overrightarrow{\mathbf{H}_l^k}$ is the forward LSTM unit's output for layer $l$, where $l = 1, 2, ..., L$, and $\overleftarrow{\mathbf{H}_l^k}$ is the output of the backward LSTM unit.

The output $\mathbf{SS}$ embedding is learned from an aggregation function, which aims to aggregate the intermediate layer representations of the bidirectional language model and the input embedding $\mathbf{N}$. For $k_{th}$ character, if we denote $\mathbf{H}_0^k = [\mathbf{N}^k, \mathbf{N}^k]$ (self concatenation), and $\mathbf{H}_l^k = [\overleftarrow{\mathbf{H}_l^k}, \overrightarrow{\mathbf{H}_l^k}]$, the output can be:

$$\mathbf{SS^k} = \omega\Big( \underbrace{s_0 \mathbf{H}_0^k}_{\text{(Variational \& Semantic)}} + \underbrace{\sum_{l=1}^{L} s_l \mathbf{H}_l^k}_{\text{Contextualized}} \Big) \quad (6)$$

where $\omega$ is the scale parameter, and $s_l$ is a weight parameter for the combination of each layer, which can be learned through the training process. Similar aggregation operation has been proven useful to model the contextualized word representation (Peters et al., 2018).

# 4 Experiment

## 4.1 Dataset and Experiment Setting

**Dataset**[5]. In Table 2, we summarize the statistics of the two real-world spam datasets (in Chinese). One is a SMS dataset, the other is a product review dataset. Both datasets were manually labeled (spam or regular labels) by professionals. False advertising and scam information are the most common forms of spam information for SMS dataset, while abuse information dominates review spam dataset.

| Dataset | Part | All | Spam | Normal |
|---|---|---|---|---|
| SMS | Train | 48,884 | 23,891 | 24,993 |
| | Test | 48,896 | 23,891 | 25,005 |
| Review | Train | 37,299 | 17,299 | 20,000 |
| | Test | 37,299 | 17,299 | 20,000 |

Table 2: Statistics of Two Chinese Spam Text Datasets

In the constructed variation graph, there are totally 25,949 Chinese characters (vertexes) and 7,705,051 variation relations. For all the variation relations, there are 1,508,768 pinyin relations (phonetic), 373,803 stroke relations (glyph), and 5,822,480 Zhengma relations (glyph).

**Experimental Set-up**. We validated the proposed model in Chinese text spam detection task. In order to simulate the "diversity", "sparseness" and " zero-shot" problems under real business scenarios, we made a challenging restriction on the

[5]https://github.com/Giruvegan/stoneskipping

| Character | Text Skipgram | | Chinese Cw2vec | | Graph VFGE | | Proposed model SS | |
|---|---|---|---|---|---|---|---|---|
| 运(move) | C | 捷(prompt) | C | 捷(prompt) | G P | 云(cloud) | S C | 转(transmit) |
| | C | 站(stop) | C | 站(stop) | G P | 纭(numerous) | G P | 芸(weed) |
| | C | 客(guest) | S C | 输(transport) | G | 坛(altar) | G P | 云(cloud) |
| 惊(shock) | S C | 讶(surprised) | S C | 讶(surprised) | G P | 景(view) | S C | 慌(flurried) |
| | S C | 愕(startled) | S C | 撼(shake) | G | 晾(dry) | G | 琼(jade) |
| | S C | 吓(scare) | S C | 愕(startled) | G | 谅(forgive) | G S C | 悚(afraid) |

G: Glyph; P: Phonetic; S: Semantic; C :Context

Table 3: Case Study: given the target character, we list the top 3 similar characters from each algorithm. The characters are selected from a frequently used candidate character set whose size is 8238.

training and testing sets, i.e., the character variations were only included in testing set, and all samples in training set were using the original characters.

For the proposed SS model, we utilized the following setting: layers of LSTMs: 2; dimension of hidden (output) state in LSTM: 128; dimension of pre-trained character text embedding: 128; dimension of VFGE embedding: 128; batch size: 64; Dropout: 0.1. For training VFGE embedding[6], the walk length was 80, the number of walks per vertex was 10. These parameters were adopted in (Peters et al., 2018; Jiang et al., 2018a; Perozzi et al., 2014; Grover and Leskovec, 2016). The variation family number[7] was 500. SS model was pre-trained for parameter initialization as suggested in (Peters et al., 2018).

**Baselines and Comparison Groups**. We chose 13 strong baseline algorithms, from text or graph viewpoints, to comprehensively evaluate the performance of the proposed method.

*General Textual Based Baselines*: **Skip-gram** (Mikolov et al., 2013), **GloVe** (Pennington et al., 2014), and **ELMo** (Peters et al., 2018).

*Chinese Specific Textual Based Baselines*: **CWE** (Chen et al., 2015), **GWE** (Su and Lee, 2017), **JWE** (Yu et al., 2017), and **Cw2vec** (Cao et al., 2018).

*Graph Embedding Based Baselines*: **DeepWalk** (Perozzi et al., 2014), **LINE** (Tang et al., 2015), **Node2vec** (Grover and Leskovec, 2016), **Metapath2vec++** (Dong et al., 2017), and **HEER** (Shi et al., 2018). We applied this group of baselines on constructed Chinese character variation graph to get graph based character embeddings. Specifically, **Metapath2vec++** required a human-defined metapath scheme to guide the random walks. We tried 4 different metapaths for this experiment:(1) $\mathbf{M2V_P}$ (only walking on pinyin (phonetic) relations); (2) $\mathbf{M2V_S}$ (only walking on stroke (glyph) relations); (3) $\mathbf{M2V_Z}$ (only walking on Zhengma (glyph) relations); (4) $\mathbf{M2V_C}$ (alternately walking on glyph and phonetic relations). We reported the best results from these four metapaths, denoted as $\mathbf{M2V_{Max}}$.

*Spelling Correction Baseline*: **Pycorrector**[8] based on n-gram language model (Yu and Li, 2014).

*Comparison Group*: we compared the performances of several variants of the proposed method in order to highlight our technical contributions. There were 3 comparison groups conducted. $\mathbf{SS_{Graph}}$: we only used VFGE graph embedding. $\mathbf{SS_{Naive}}$: we simply concatenated VFGE graph embedding and skip-gram textual embedding (a naive version). $\mathbf{SS_{Original}}$: the proposed SS model.

For a fair comparison, the dimension[9] of all embedding models was 128. A single layer of CNN classification model[10] was used for spam detection task.

## 4.2 Experiment Result and Analysis

The text spam detection task performances of different models were reported in Table 1. Based on the experiment results, we had the following ob-

[6]For the experiment fairness, all the random walk based graph embedding baselines shared the same parameters with VFGE.

[7]Based on the parameter sensitive analysis, the proposed method was not very sensitive to number of variation families.

[8]https://github.com/shibing624/pycorrector

[9]The initial dimension of $\mathrm{SS_{Naive}}$ and $\mathrm{SS_{Original}}$ is 256, so we used a fully connected layer to reduce its dimension to 128.

[10]The filter sizes of CNN is 3, 4, 5, and the filter number is 128, dropout ratio is 0.1.

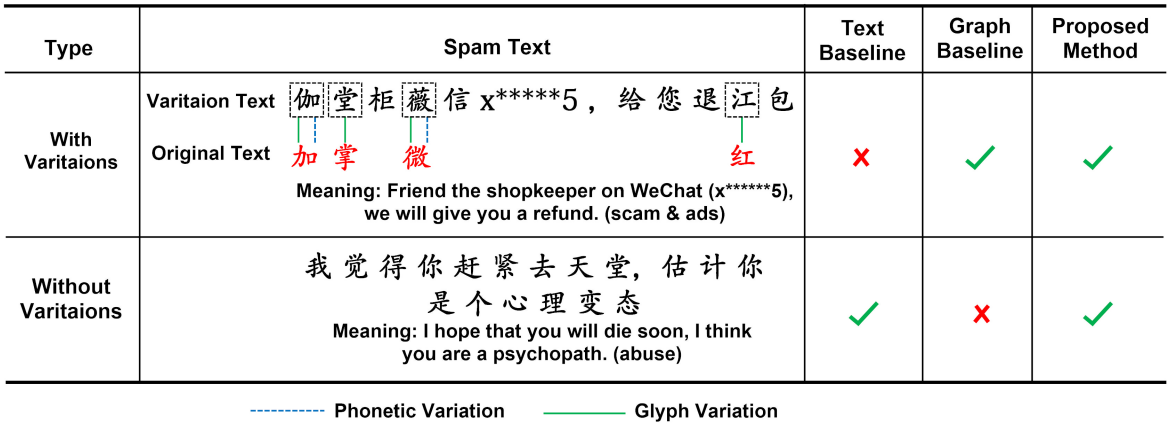


Figure 3: Two typical examples for CSVD task

servations:

**SS vs. Baselines**. (1) $\text{SS}_{\text{Original}}$ outperformed the baseline models for all evaluation metrics on both datasets, which indicated the proposed SS model can effectively address the CSVD problem. (2) On review dataset, the leading gap between $\text{SS}_{\text{Original}}$ and other baselines was greater. A possible explanation was that, the review spam text usually had richer content and more complex variation patterns than SMS spam text. Therefore, a good variation representation model may have certain advantages.

**Chinese vs. General**. (1) Compared to classical textual embedding models (Skipgram and GloVe), the Chinese embedding models showed their advantages, especially on review dataset. This result indicated that the characteristic knowledge of Chinese can help to detect spam text. (2) ELMo was able to learn both the semantic and contextualized information, and it achieved a good performance in text baseline group.

**Graph vs. Text**. Generally, the graph based baselines outperformed the textual based baselines (including general and Chinese). This observation indicated: (1) the variation knowledge of Chinese character can be critical for CSVD problem. (2) The proposed character variation graph can provide critical information for Chinese character representation learning. (3) Compared to other graph based baselines, $\text{SS}_{\text{Graph}}$ was superior, which proved the effectiveness of VFGE algorithm, and the proposed variation family can characterize and predict useful variation patterns for CSVD problem.

**Chinese Character Encodings**. (1) In Chinese textual embedding baseline group, JWE (radical based) and Cw2vec (stroke based) didn't perform well, which indicated employing a single kind of glyph-based information can be insufficient for Chinese variation representation learning. Similarly, in graph based baseline group, the performances of $\text{M2V}_{\text{P}}$, $\text{M2V}_{\text{S}}$ and $\text{M2V}_{\text{Z}}$ (employed only one encoding relation on the constructed graph) were still unsatisfactory. The results revealed that an individual encoding method cannot comprehensively encode a character, we should consider various kinds of variation information simultaneously. (2) The performance of $\text{M2V}_{\text{C}}$ (integrated all relations based on a predefined metapath pattern) was still inferior. This result indicated a human-defined rule cannot effectively integrate all relationships in a complex graph.

**Representation vs. Spelling Correction**. Pycorrector performed poorly in experiment, and other baselines outperformed this approach, which proved the spelling correction method is not capable for CSVD problem.

**Variants of SS model**. For variants of the proposed method, the results showed that (1) by combining the semantic and sequential information, the task performances can improve; (2) simply concatenating graph and text embeddings cannot generate a satisfactory joint representation. (3) The proposed SS model can successfully capture the variation, semantic, and sequential information for character representation learning.

### 4.3 Case Study

To gain an insightful understanding regarding the variation representation of the proposed method, we conduct qualitative analysis by performing the case studies of character similarities. As shown in Table 3, for exemplary characters, the most

similar characters, based on skipgram embedding (general textual based baseline), are all semantically similar or/and context-related. Meanwhile, based on Cw2vec embedding (most recent Chinese embedding baseline), all similar characters for target characters are also semantically similar or/and context-related. Unsurprisingly, for each target character, all similar characters based on VFGE model (best performed graph embedding model), are glyph and phonetic similar characters. The proposed SS model can achieve a comprehensive coverage from variation, semantic and context viewpoints. For instance, in its top 3 similar characters for "运(move)", "转(transmit)" is a semantic and context similar character, and "云(cloud)" is a glyph and phonetic similar character. Furthermore, SS model can capture complicated compound similarity between Chinese characters, for instance, "悚(afraid)" is a glyph, semantic, and context similar character for "惊(shock)". This also explains why SS model performs well to address the CSVD problem.

Figure 3 depicts two typical examples in the experimental datasets. For the spam text with variations, spammers used character variations to create camouflaged expressions. For instance, using glyph variation "江(river)" to replace "红(red)", and glyph-phonetic compound variation "薇(osmund)" to replace "微(micro)". The classical text embedding models may fail to identify this kind of spam texts. With the mining of character variation graph, the graph based approaches can be successful to capture these changes. For spam text without variations, classification models need more semantic and contextual information, and the text based methods can be suitable for this kind of spam texts. The proposed SS model is able to detect both two kinds of spam texts effectively, and experiment results proved SS can successfully model Chinese variational, semantic and contextualized representations for CSVD task.

## 5 Conclusion

In this paper, we propose a StoneSkipping model for Chinese spam detection. The performance of the proposed method is comprehensively evaluated in two real world datasets with challenging experimental setting. The results of experiments show that the proposed model significantly outperforms a number of state-of-the-art methods. Meanwhile, the case study empirically proves that the proposed model can successfully capture the Chinese variation, semantic, and contextualized information, which can be essential for CSVD problem. In the future, we will investigate more sophisticated methods to improve SS's performance, e.g., enable self-attention mechanism for contextualized information modelling.

## Acknowledgments

This work is supported by the National Natural Science Foundation of China (61876003, 81971691), the China Department of Science and Technology Key Grant (2018YFC1704206), and Fundamental Research Funds for the Central Universities (18lgpy62).

## References

Shaosheng Cao, Wei Lu, Jun Zhou, and Xiaolong Li. 2018. cw2vec: Learning chinese word embeddings with stroke n-gram information. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, pages 5053–5061.

Xinxiong Chen, Lei Xu, Zhiyuan Liu, Maosong Sun, and Huan-Bo Luan. 2015. Joint learning of character and word embeddings. In *IJCAI*, pages 1236–1242.

Zheng Chen and Kai-Fu Lee. 2000. A new statistical approach to chinese pinyin input. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 135–144. ACM.

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 855–864. ACM.

Zhuoren Jiang, Liangcai Gao, Ke Yuan, Zheng Gao, Zhi Tang, and Xiaozhong Liu. 2018a. Mathematics content understanding for cyberlearning via formula evolution map. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 37–46. ACM.

Zhuoren Jiang, Yue Yin, Liangcai Gao, Yao Lu, and Xiaozhong Liu. 2018b. Cross-language citation recommendation via hierarchical representation learning on heterogeneous graph. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 635–644. ACM.

Nitin Jindal and Bing Liu. 2007. Review spam detection. In *Proceedings of the 16th international conference on World Wide Web*, pages 1189–1190. ACM.

Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. 2015. Topical word embeddings. In *AAAI*, pages 2418–2424.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Jerry Norman. 1988. *Chinese*. Cambridge University Press.

Alexandros Ntoulas, Marc Najork, Mark Manasse, and Dennis Fetterly. 2006. Detecting spam web pages through content analysis. In *Proceedings of the 15th international conference on World Wide Web*, pages 83–92. ACM.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 2227–2237.

Ian Porteous, David Newman, Alexander Ihler, Arthur Asuncion, Padhraic Smyth, and Max Welling. 2008. Fast collapsed gibbs sampling for latent dirichlet allocation. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 569–577. ACM.

Yu Shi, Qi Zhu, Fang Guo, Chao Zhang, and Jiawei Han. 2018. Easing embedding learning by comprehensive transcription of heterogeneous information networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2190–2199. ACM.

Hua Shu and Richard C Anderson. 1997. Role of radical awareness in the character and word acquisition of chinese children. *Reading Research Quarterly*, 32(1):78–89.

Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. 2013. Zero-shot learning through cross-modal transfer. In *Advances in neural information processing systems*, pages 935–943.

John A Spinks, Ying Liu, Charles A Perfetti, and Li Hai Tan. 2000. Reading chinese characters for meaning: The role of phonological information. *Cognition*, 76(1):B1–B11.

Tzu-ray Su and Hung-yi Lee. 2017. Learning chinese word representations from glyphs of characters. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 264–273.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077. International World Wide Web Conferences Steering Committee.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Jui-Feng Yeh, Yun-Yun Lu, Chen-Hsien Lee, Yu-Hsiang Yu, and Yong-Ting Chen. 2014. Chinese word spelling correction based on rule induction. In *Proceedings of The Third CIPS-SIGHAN Joint Conference on Chinese Language Processing*, pages 139–145.

Jinxing Yu, Xun Jian, Hao Xin, and Yangqiu Song. 2017. Joint embeddings of chinese words, characters, and fine-grained subcharacter components. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 286–291.

Junjie Yu and Zhenghua Li. 2014. Chinese spelling error detection and correction based on language model, pronunciation, and shape. In *Proceedings of The Third CIPS-SIGHAN Joint Conference on Chinese Language Processing*, pages 220–223.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

# An Attentive Fine-Grained Entity Typing Model with Latent Type Representation

**Ying Lin** and **Heng Ji**
Computer Science Department
University of Illinois at Urbana-Champaign, Urbana, IL, USA 61802
{yinglin8,hengji}@illinois.edu

## Abstract

We propose a fine-grained entity typing model with a novel attention mechanism and a hybrid type classifier. We advance existing methods in two aspects: feature extraction and type prediction. To capture richer contextual information, we adopt contextualized word representations instead of fixed word embeddings used in previous work. In addition, we propose a two-step mention-aware attention mechanism to enable the model to focus on important words in mentions and contexts. We also present a hybrid classification method beyond binary relevance to exploit type interdependency with latent type representation. Instead of independently predicting each type, we predict a low-dimensional vector that encodes latent type features and reconstruct the type vector from this latent representation. Experiment results on multiple data sets show that our model significantly advances the state-of-the-art on fine-grained entity typing, obtaining up to 6.6% and 5.5% absolute gains in macro averaged F-score and micro averaged F-score respectively. [1]

## 1 Introduction

Fine-grained entity typing aims to assign one or more types to each entity mention given a certain context. For example, in the following sentence, "*If **Rogers** is in the game, the Huskies will be much better equipped to match the Cougars in that aspect*", the mention "Rogers" should be labeled as `athlete` in addition to `person` according to the context (e.g., game, Huskies). These fine-grained entity types are proven to be effective in supporting a wide range of downstream applications such as relation extraction (Yao et al., 2010), question answering (Lin et al., 2012), and coreference resolution (Recasens et al., 2013).

Fine-grained entity typing is usually formulated as a multi-label classification problem. Previous approaches (Ling and Weld, 2012; Choi et al., 2018; Xin et al., 2018) typically address it with binary relevance that decomposes the problem into isolated binary classification subproblems and independently predicts each type. However, this method is commonly criticized for its label independence assumption, which is not valid for fine-grained entity typing. For example, if the model is confident at predicting the type `artist`, it should promote its parent type `person` but discourage `organization` and its descendant types. In order to capture inter-dependencies between types, we propose a hybrid model that incorporates latent type representation in addition to binary relevance. Specifically, the model learns to predict a low-dimensional vector that encodes latent type features obtained through Principle Label Space Transformation (Tai and Lin, 2012) and reconstruct the sparse and high-dimensional type vector from this latent representation.

Another major challenge in fine-grained entity typing is to differentiate similar types, such as `director` and `actor`, which requires the model to capture slightly different nuances in texts. Previous neural models (Shimaoka et al., 2016; Xin et al., 2018; Choi et al., 2018; Xu and Barbosa, 2018) generally extract features from pre-trained word embeddings. Instead, we adopt contextualized word representations (Peters et al., 2018), which can capture context-aware word semantics and better represent out-of-vocabulary words. We further propose a two-step attention mechanism to actively extract the most relevant information from the sentence. Particularly, we calculate the attention for context words in a mention-aware manner, allowing the model to focus on different parts of the sentence for different mentions. For example, in the following sentence, "*In 2005 two fed-*

[1] Code for this paper is available at: https://github.com/limteng-rpi/fet.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6197–6202, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

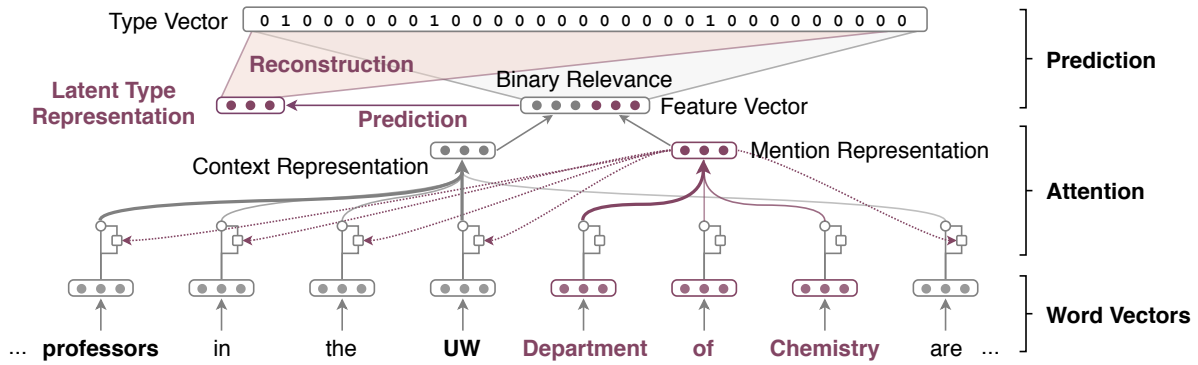


Figure 1: An illustration of our fine-grained entity typing framework.

*eral agencies, the* ***US Geological Survey*** *and the* ***Fish and Wildlife Service****, began to identify fish in the* ***Potomac*** *and tributaries ...*", the model should use "federal agencies" to help classify "US Geological Survey" and "Fish and Wildlife Service" as `government_agency`, but focus on "fish" and "tributaries" to determine that "Potomac" should be a `body_of_water` (river) instead of a `city`.

## 2 Methodology

Figure 1 illustrates our fine-grained entity typing framework. We represent the input sentence using pre-trained contextualized word representations. Next, we apply a two-step mention-aware attention mechanism to extract the most relevant features from the sentence to form the feature vector. On top of the model, we employ a hybrid classifier to predict the types of each mention.

### 2.1 Sentence Encoder

Contextual information plays a key role as we often need to determine the types especially subtypes according to the context. Hence, unlike previous neural models that generally use fixed word embeddings, we employ contextualized word representations (ELMo, Peters et al. 2018) that can capture word semantics in different contexts. Furthermore, because ELMo takes as input characters instead of words, it can better represent out-of-vocabulary words that are prevalent in entity mentions by leveraging sub-word information. Given a sentence of $S$ words, the encoder generates a sequence of word vectors $\{\boldsymbol{r}_1, ..., \boldsymbol{r}_S\}$, where $\boldsymbol{r}_i \in \mathbb{R}^{d_r}$ is the representation of the $i$-th word.

### 2.2 Mention Representation

Previous attentive models (Shimaoka et al., 2017; Xu and Barbosa, 2018; Xin et al., 2018; Choi et al., 2018) only apply attention mechanisms to the context. However, some words in an entity mention may provide more useful information for typing, such as "Department" in Figure 1. To allow the model to focus on more informative words, we represent a mention $m$ consisting of $M$ words as a weighted sum of its contextualized word representations with an attention mechanism (Bahdanau et al., 2015) as

$$\boldsymbol{m} = \sum_i^M a_i^m \boldsymbol{r}_i,$$

where the attention score $a_i^m$ is computed as

$$a_i^m = \text{Softmax}(e_i^m) = \frac{\exp(e_i^m)}{\sum_k^M \exp(e_k^m)},$$

$$e_i^m = \boldsymbol{v}^{m\top} \tanh(\boldsymbol{W}^m \boldsymbol{r}_i),$$

where parameters $\boldsymbol{W}^m \in \mathbb{R}^{d_a \times d_r}$ and $\boldsymbol{v}^m \in \mathbb{R}^{d_a}$ are learned during training, and the hidden attention dimension $d_a$ is set to $d_r$ in our experiments.

### 2.3 Context Representation

Given the context of mention $m$, we form its representation from involved contextualized word vectors with a mention-aware attention mechanism

$$\boldsymbol{c} = \sum_i^C a_i^c \boldsymbol{r}_i,$$

where $C$ is the number of contextual words, and $a_i^c$ is defined as $a_i^c = \text{Softmax}(e_i^c)$, where

$$e_i^c = \boldsymbol{v}^{c\top} \tanh(\boldsymbol{W}^c(\boldsymbol{r}_i \oplus \boldsymbol{m} \oplus p_i)),$$

where $\oplus$ represents concatenation, and $\boldsymbol{v}^c \in \mathbb{R}^{d_a}$ and $\boldsymbol{W}^c \in \mathbb{R}^{d_a \times (2d_r+1)}$ are trainable parameters. We introduce a relative position term $p_i$ to indicate the distance from the $i$-th word to the mention as

$$p_i = \left(1 - \mu\Big(\min(|i-a|, |i-b|) - 1\Big)\right)^+,$$

where $a$ and $b$ are indices of the first and last words of the mention, and $\mu$ is set to 0.1.

Finally, the feature vector of mention $m$ is formed by concatenating its mention representation $\boldsymbol{m}$ and context representation $\boldsymbol{c}$.

### 2.4 Hybrid Classification Model

We propose a hybrid type classification model consisting of two classifiers as Figure 1 shows. We first learn a matrix $\boldsymbol{W}^b \in \mathbb{R}^{d_t \times 2d_r}$ to predict type scores by

$$\tilde{\boldsymbol{y}}^b = \boldsymbol{W}^b(\boldsymbol{m} \oplus \boldsymbol{c}),$$

where $\tilde{y}_i^b$ is the score for the $i$-th type and $d_t$ is the number of types. However, this method independently predicts each type and does not consider their inter-dependencies. To tackle this issue, we introduce an additional classifier inspired by Principle Label Space Transformation (Tai and Lin, 2012). Under the hypercube sparsity assumption that the number of training examples is much smaller than $2^{d_t}$, Tai and Lin (2012) project high-dimensional type vectors into a low-dimensional space to find underlying type correlations behind the first order co-occurrence through Singular Value Decomposition (SVD)

$$\boldsymbol{Y} \approx \tilde{\boldsymbol{Y}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{L}^\top,$$

where $\boldsymbol{U} \in \mathbb{R}^{d_t \times d_l}$, $\boldsymbol{\Sigma} \in \mathbb{R}^{d_l \times d_l}$, $\boldsymbol{L} \in \mathbb{R}^{N \times d_l}$, and $d_l \ll d_t$. This low-dimensional space is similar to the hidden concept space in Latent Semantic Analysis (Deerwester et al., 1990). The $i$-th row of $\boldsymbol{L}$ is the latent representation of the $i$-th type vector. After that, we learn to predict the latent type representation from the feature vector using

$$\boldsymbol{l} = \boldsymbol{V}^l(\boldsymbol{m} \oplus \boldsymbol{c}),$$

where $\boldsymbol{V}^l \in \mathbb{R}^{2d_r \times d_l}$ is trainable. We then reconstruct the type vector from $\boldsymbol{l}$ using a linear projection $\tilde{\boldsymbol{y}}^l = \boldsymbol{W}^l\boldsymbol{l} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{l}$. Next, by combining scores from both classifiers, we have

$$\tilde{\boldsymbol{y}} = \sigma(\boldsymbol{W}^b(\boldsymbol{m} \oplus \boldsymbol{c}) + \gamma\boldsymbol{W}^l\boldsymbol{l}),$$

where $\gamma$ is a scalar initialized to 0.1 and updated during training. Finally, our training objective is to minimize the following cross-entropy-based loss function

$$J(\theta) = -\frac{1}{N}\sum_i^N \boldsymbol{y}_i \log \tilde{\boldsymbol{y}}_i + (1 - \boldsymbol{y}_i)\log(1 - \tilde{\boldsymbol{y}}_i).$$

In the test phase, we predict each type with a probability $\tilde{y}_i > 0.5$ or $\arg\max \tilde{y}_i$ if all probabilities are lower than 0.5.

## 3 Experiments

### 3.1 Data Sets

In our experiments, we evaluate the proposed model on the following data sets.

**OntoNotes** fine-grained entity typing data set is derived from the OntoNotes corpus (Weischedel et al., 2013) and annotated by Gillick et al. (2014) using a three-layer set of 87 types. We use the augmented data set created by Choi et al. (2018).

**FIGER** (Ling and Weld, 2012) contains 2.7 million automatically labeled training instances from Wikipedia and 434 manually annotated sentences from news reports. We use ground truth mentions in our experiments and sample 0.1M training instances as the development set.

**KNET** (Xin et al., 2018) is another data set derived from Wikipedia. It consists of an automatically annotated subset (WIKI-AUTO) and a manually annotated (WIKI-MAN) test set.

**BBN Pronoun Coreference and Entity Type Corpus** (BBN, Weischedel and Brunstein, 2005) annotates 2,311 Wall Street Journal articles in Treebank-2 (LDC95T7) with fine-grained entity types. We use the version processed by Ren et al. (2016a).

| Data set | Train | Dev | Test | Label | Depth |
|---|---|---|---|---|---|
| OntoNotes | 3.4M | 2,202 | 8,963 | 87 | 3 |
| FIGER | 2.7M | - | 563 | 113 | 2 |
| KNET (Wiki-Auto) | 1M | 0.1M | 0.1M | 74 | 2 |
| KNET (Wiki-Man) | - | - | 100 | 74 | 2 |
| BBN | 32,739 | - | 6,430 | 56 | 2 |

Table 1: Data set statistics: Numbers of train/dev/test instances, label set size, max hierarchy depth.

### 3.2 Experimental Setup

We use the pre-trained `original-5.5b` ELMo model [2] and freeze its weights during training. We use an Adam optimizer with learning rate of 5e-5, L2 weight decay of 0.01, warmup rate of 0.1, and linear learning rate decay. We use a mini-batch size of 200. To reduce overfitting, we apply dropout (Srivastava et al., 2014) to the word representation, relative position term, and the final feature vector with probability 0.5, 0.2, and 0.2.

We evaluate the performance by strict accuracy (Acc), macro-average F-score (Macro F1), and micro-average F-score (Micro F1) (Ling and Weld, 2012).

[2] https://allennlp.org/elmo

### 3.3 Evaluation Results

We compare the performance of our model with state-of-the-art methods on OntoNotes, FIGER, and KNET in Table 2, 3, and 4.

| Model | Acc | Macro F1 | Micro F1 |
|---|---|---|---|
| Shimaoka et al. (2016) | 51.7 | 70.9 | 64.9 |
| Ren et al. (2016b) | 57.2 | 71.5 | 66.1 |
| Choi et al. (2018) | 59.5 | 76.8 | 71.8 |
| Our Model | **63.8** | **82.9** | **77.3** |

Table 2: Results on the OntoNotes test set. The first three methods use only KB-based supervision.

| Model | Acc | Macro F1 | Micro F1 |
|---|---|---|---|
| Ling and Weld (2012) | 53.2 | 69.9 | 69.3 |
| Yogatama et al. (2015) | – | – | 72.3 |
| Shimaoka et al. (2017) | 54.5 | 74.8 | 71.6 |
| + hand-crafted | 59.7 | 79.0 | 75.4 |
| Our Model | **62.9** | **83.0** | **79.8** |

Table 3: Results on the FIGER (Gold) test set.

| WIKI-AUTO | | | |
|---|---|---|---|
| **Model** | **Acc** | **Macro F1** | **Micro F1** |
| Shimaoka et al. (2016) | 42.8 | 72.4 | 74.9 |
| KNET-MA | 41.6 | 72.7 | 75.7 |
| KNET-KA* | 45.5 | 73.6 | 76.2 |
| KNET-KAD* | **47.2** | 74.9 | 77.9 |
| Our Model | 45.8 | **77.4** | **78.4** |
| **WIKI-MAN** | | | |
| **Model** | **Acc** | **Macro F1** | **Micro F1** |
| Shimaoka et al. (2016) | 18.0 | 69.4 | 70.1 |
| KNET-MA | 26.0 | 71.2 | 72.1 |
| KNET-KA* | 23.0 | 71.1 | 71.7 |
| KNET-KAD* | **34.0** | 74.9 | 75.3 |
| Our Model | 29.0 | **77.6** | **75.3** |

Table 4: Results on KNET test sets. KNET-KA and KNET-KAD use additional entity embeddings.

| Model | Acc | Macro F1 | Micro F1 |
|---|---|---|---|
| Ling and Weld (2012) | 46.7 | 67.2 | 61.2 |
| Yosef et al. (2012) | 52.3 | 57.6 | 58.7 |
| Ren et al. (2016a) | **67.0** | 72.7 | 73.5 |
| Our Model | 55.9 | **79.3** | **78.1** |
| Our Model* | 55.4 | 76.1 | 75.7 |

Table 5: Results on the BBN test set. Our Model* is a variant of our model that uses Bert contextualized word representations.

We compare the outputs of both classifiers ($\tilde{\boldsymbol{y}}^b$ and $\tilde{\boldsymbol{y}}^l$). The reconstructed type vector $\tilde{\boldsymbol{y}}^l$ alone doesn't predict entity types accurately, while adding this classifier substantially improves the performance of the model.

We show results on the BBN dataset in Table 5. Our Model* is a variant where we replace ELMo embeddings with Bert (large cased model) contextualized word representations.

| | Acc | Marco F1 | Micro F1 |
|---|---|---|---|
| Shimaoka et al. (2017) | 54.5 | 74.8 | 71.6 |
| + ELMo | 59.3 | 81.7 | 79.0 |
| + latent | 57.7 | 77.5 | 75.2 |
| + two-step attention | 59.7 | 78.8 | 76.3 |
| + ELMo | 62.9 | 83.0 | 79.8 |

Table 6: Ablation study on the FIGER (Gold) test set.

To evaluate the influence of individual components of our model, we conduct an ablation study as shown in Table 6. We implement a baseline model similar to (Shimaoka et al., 2017) (no hand-crafted features), the state-of-the-art on this data set. We observe that each component added to the model improves its performance.

We visualize mention and context attention in Figure 2. The first example shows the impact of mention attention. The baseline model mistakenly classifies the mention as `location` probably because "Asian" generally appears in location mentions, while our model successfully predicts `organization` by assigning higher weights to "Student Commission". In Example #2 and #3, we compare context attention between "Terry Martino" and "APA". Although they occur in the same sentence, our model is able to focus on different context words for different mentions with the mention-aware attention mechanism.

**#1** ... *left out of the* [ORGANIZATION *Asian Student Commission* ] *and that they don't feel like they are being represented well...*

**#2** *After Executive Director* [PERSON *Terry Martino*] *announced ... applause packed into the APA 's board room*

**#3** *After Executive Director Terry Martino announced ... applause packed into the* [ORGANIZATION *APA* ] *'s board room*

Figure 2: Mention and context attention visualization.

## 4 Related Work

As entity types are usually organized as a forest of hierarchies, several models are proposed to leverage this structure. In (Yosef et al., 2012), the authors build a set of classifiers based on the taxon-

omy of YAGO (Hoffart et al., 2013) and perform top-down hierarchical classification. Shimaoka et al. (2017) propose a hierarchical label encoding method to share parameters between types in the same hierarchy. Xu and Barbosa (2018) propose a hierarchy-aware loss function to reduce the penalty when predicted types are related. By contrast, our model automatically find type inter-dependency via matrix factorization and is able to capture inter-dependencies between types regardless of whether they are in the same hierarchy.

Attention mechanisms are widely used in neural fine-grained entity typing models (Shimaoka et al., 2016, 2017; Xu and Barbosa, 2018; Xin et al., 2018; Choi et al., 2018) to weight context words. We also apply it to mention words and introduce a position term to make attentions for context words more mention-aware.

## 5 Conclusions and Future Work

We propose an attentive architecture for fine-grained entity typing with latent type representation. Experiments on multiple data sets demonstrate that our model achieves state-of-the-art performance. In the future, we will further improve the performance of fine-grained types, which is still lower than that of general types due to less training instances and distant supervision noise. We also plan to utilize fine-grained entity typing results in more downstream applications, such as coreference resolution and event extraction.

## Acknowledgement

This work was supported by the U.S. DARPA AIDA Program No. FA8750-18-2-0014, Air Force No. FA8650-17-C-7715, and U.S. ARL NS-CTA No. W911NF-09-2-0053. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

## References

Dzmitry Bahdanau, KyungHyun Cho, Yoshua Bengio, and Roee Aharoni. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 6th International Conference on Learning Representations (ICLR 2015)*.

Eunsol Choi, Omer Levy, Yejin Choi, and Luke Zettlemoyer. 2018. Ultra-fine entity typing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407.

Dan Gillick, Nevena Lazic, Kuzman Ganchev, Jesse Kirchner, and David Huynh. 2014. Context-dependent fine-grained entity type tagging. *arXiv preprint arXiv:1412.1820*.

Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, and Gerhard Weikum. 2013. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence*, 194:28–61.

Thomas Lin, Oren Etzioni, et al. 2012. No noun phrase left behind: detecting and typing unlinkable entities. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP 2012)*.

Xiao Ling and Daniel S Weld. 2012. Fine-grained entity recognition. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI 2012)*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT 2018)*.

Marta Recasens, Marie-Catherine de Marneffe, and Christopher Potts. 2013. The life and death of discourse entities: Identifying singleton mentions. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT 2013)*.

Xiang Ren, Wenqi He, Meng Qu, Lifu Huang, Heng Ji, and Jiawei Han. 2016a. AFET: Automatic fine-grained entity typing by hierarchical partial-label embedding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*.

Xiang Ren, Wenqi He, Meng Qu, Clare R Voss, Heng Ji, and Jiawei Han. 2016b. Label noise reduction in entity typing by heterogeneous partial-label embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD 2016)*.

Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. 2016. An attentive neural architecture for fine-grained entity type classification. In *Proceedings of the 5th Workshop on Automated Knowledge Base Construction*.

Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. 2017. Neural architectures for fine-grained entity type classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017)*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*.

Farbound Tai and Hsuan-Tien Lin. 2012. Multilabel classification with principal label space transformation. *Neural Computation*, 24(9):2508–2542.

Ralph Weischedel and Ada Brunstein. 2005. Bbn pronoun coreference and entity type corpus. *BBN Technologies*.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al. 2013. Ontonotes release 5.0 LDC2013T19. *Linguistic Data Consortium, Philadelphia, PA*.

Ji Xin, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2018. Improving neural fine-grained entity typing with knowledge attention. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI 2018)*.

Peng Xu and Denilson Barbosa. 2018. Neural fine-grained entity type classification with hierarchy-aware loss. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT 2018)*.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2010. Collective cross-document relation extraction without labelled data. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP 2010)*.

Dani Yogatama, Daniel Gillick, and Nevena Lazic. 2015. Embedding methods for fine grained entity type classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL 2015)*.

Mohamed Amir Yosef, Sandro Bauer, Johannes Hoffart, Marc Spaniol, and Gerhard Weikum. 2012. Hyena: Hierarchical type classification for entity names. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING 2012)*.

# An Improved Neural Baseline for Temporal Relation Extraction

**Qiang Ning,**[1] **Sanjay Subramanian,**[2] **and Dan Roth**[1,2]
[1]University of Illinois at Urbana-Champaign
[2]University of Pennsylvania
qning2@illinois.edu, {subs,danroth}@seas.upenn.edu

## Abstract

Determining temporal relations (e.g., *before* or *after*) between events has been a challenging natural language understanding task, partly due to the difficulty to generate large amounts of high-quality training data. Consequently, neural approaches have not been widely used on it, or showed only moderate improvements. This paper proposes a new neural system that achieves about 10% absolute improvement in accuracy over the previous best system (25% error reduction) on two benchmark datasets. The proposed system is trained on the state-of-the-art MATRES dataset and applies contextualized word embeddings, a Siamese encoder of a temporal common sense knowledge base, and global inference via integer linear programming (ILP). We suggest that the new approach could serve as a strong baseline for future research in this area.

## 1 Introduction

Temporal relation (TempRel) extraction has been considered as a major component of understanding time in natural language (Do et al., 2012; UzZaman et al., 2013; Minard et al., 2015; Llorens et al., 2015; Ning et al., 2018a). However, the annotation process for TempRels is known to be time consuming and difficult even for humans, and existing datasets are usually small and/or have low inter-annotator agreements (IAA); e.g., UzZaman et al. (2013); Chambers et al. (2014); O'Gorman et al. (2016) reported Kohen's $\kappa$ and $F_1$ in the 60's. Albeit the significant progress in deep learning nowadays, neural approaches have not been used extensively for this task, or showed only moderate improvements (Dligach et al., 2017; Lin et al., 2017; Meng and Rumshisky, 2018). We think it is important for to understand: is it because we missed a "magic" neural architecture, because the training dataset is small, or because the quality of the dataset should be improved?

Recently, Ning et al. (2018c) introduced a new dataset called *Multi-Axis Temporal RElations for Start-points* (MATRES). MATRES is still relatively small in its size (15K TempRels), but has a higher annotation quality from its improved task definition and annotation guideline. This paper uses MATRES to show that a long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) system can readily outperform the previous state-of-the-art system, CogCompTime (Ning et al., 2018d), by a large margin. The fact that a standard LSTM system can significantly improve over a feature-based system on MATRES indicates that neural approaches have been mainly dwarfed by the quality of annotation, instead of specific neural architectures or the small size of data.

To gain a better understanding of the standard LSTM method, we extensively compare the usage of various word embedding techniques, including word2vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014), FastText (Bojanowski et al., 2016), ELMo (Peters et al., 2018), and BERT (Devlin et al., 2018), and show their impact on TempRel extraction. Moreover, we further improve the LSTM system by injecting knowledge from an updated version of TEMPROB, an automatically induced temporal common sense knowledge base that provides *typical* TempRels between events[1] (Ning et al., 2018b). Altogether, these components improve over CogCompTime by about 10% in $F_1$ and accuracy. The proposed system is public[2] and can serve as a strong baseline for future research.

## 2 Related Work

Early computational attempts to TempRel extraction include Mani et al. (2006); Chambers et al.

[1]For example, "explode" typically happens *before* "die".
[2]https://cogcomp.org/page/publication_view/879



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6203–6209, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

(2007); Bethard et al. (2007); Verhagen and Pustejovsky (2008), which aimed at building classic learning algorithms (e.g., perceptron, SVM, and logistic regression) using *hand-engineered features* extracted for each pair of events. The frontier was later pushed forward through continuous efforts in a series of SemEval workshops (Verhagen et al., 2007, 2010; UzZaman et al., 2013; Bethard et al., 2015, 2016, 2017), and significant progresses were made in terms of data annotation (Styler IV et al., 2014; Cassidy et al., 2014; Mostafazadeh et al., 2016; O'Gorman et al., 2016), structured inference (Chambers and Jurafsky, 2008a; Do et al., 2012; Chambers et al., 2014; Ning et al., 2018a), and structured machine learning (Yoshikawa et al., 2009; Leeuwenberg and Moens, 2017; Ning et al., 2017).

Since TempRel is a specific relation type, it is natural to borrow recent neural relation extraction approaches (Zeng et al., 2014; Zhang et al., 2015; Zhang and Wang, 2015; Xu et al., 2016). There have indeed been such attempts, e.g., in clinical narratives (Dligach et al., 2017; Lin et al., 2017; Tourille et al., 2017) and in newswire (Cheng and Miyao, 2017; Meng and Rumshisky, 2018; Leeuwenberg and Moens, 2018). However, their improvements over feature-based methods were moderate (Lin et al. (2017) even showed negative results). Given the low IAAs in those datasets, it was unclear whether it was simply due to the low data quality or neural methods inherently do not work well for this task.

A recent annotation scheme, Ning et al. (2018c), introduced the notion of multi-axis to represent the temporal structure of text, and identified that one of the sources of confusions in human annotation is asking annotators for TempRels across different axes. When annotating only same-axis TempRels, along with some other improvements to the annotation guidelines, MATRES was able to achieve much higher IAAs.[3] This dataset opens up opportunities to study neural methods for this problem. In Sec. 3, we will explain our proposed LSTM system, and also highlight the major differences from previous neural attempts.

## 3 Neural TempRel Extraction

One major disadvantage of feature-based systems is that errors occurred in feature extraction propagate to subsequent modules. Here we study the usage of LSTM networks[4] on the TempRel extraction problem as an end-to-end approach that only takes a sequence of word embeddings as input (assuming that the position of events are known). Conceptually, we need to feed those word embeddings to LSTMs and obtain a vector representation for a particular pair of events, which is followed by a fully-connected, feed-forward neural network (FFNN) to generate confidence scores for each output label. Based on the confidence scores, global inference is performed via integer linear programming (ILP), which is a standard procedure used in many existing works to enforce the transitivity property of time (Chambers and Jurafsky, 2008b; Do et al., 2012; Ning et al., 2017). An overview of the proposed network structure and corresponding parameters can be found in Fig. 1. Below we also explain the main components.

### 3.1 Handling Event Positions

Each TempRel is associated with two events, and for the same text, different pairs of events possess different relations, so it is critical to indicate the positions of those events when we train LSTMs for the task. The most straightforward way is to concatenate the hidden states from both time steps that correspond to the location of those events (Fig. 1b). Dligach et al. (2017) handled this issue differently, by adding XML tags immediately before and after each event (Fig. 1a). For example, in the sentence, *After **eating** dinner, he **slept** comfortably*, where the two events are bold-faced, they will convert the sequence into *After <e1> eating </e1> dinner, he <e2> slept </e2> comfortably.* The XML markups, which was initially proposed under the name of *position indicators* for relation extraction (Zhang and Wang, 2015), uniquely indicate the event positions to LSTM, such that the final output of LSTM can be used as a representation of those events and their context. We compare both methods in this paper, and as we show later, the straightforward concatenation method is already as good as XML tags for this task.

### 3.2 Common Sense Encoder (CSE)

In naturally occurring text that expresses TempRels, connective words such as *since*,

[3] Between experts: Kohen's $\kappa \approx 0.84$. Among crowdsourcers: accuracy 88%. More details in Ning et al. (2018c).

[4] We also tried convolutional neural networks but did not observe that CNNs improved performance significantly compared to the LSTMs. Comparison between LSTM and CNN is also not the focus of this paper.

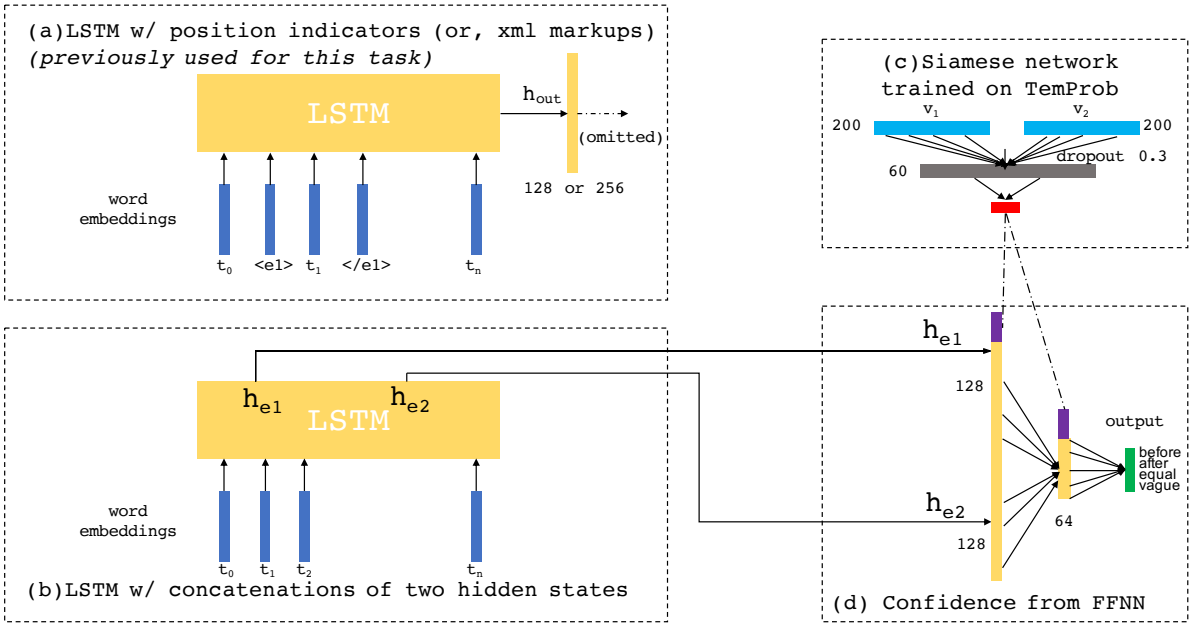


Figure 1: Overview of the neural network structures studied in this paper. Networks (a) and (b) are two ways to handle event positions in LSTMs (Sec. 3.1). (c) The Siamese network used to fit TemProb (Sec. 3.2). Once trained on TemProb, the Siamese network is fixed when training other parts of the system. (d) The FFNN that generates confidence scores for each label. Sizes of hidden layers are already noted. Embeddings of the same color share the same matrix.

*when*, or *until* are often not explicit; nevertheless, humans can still infer the TempRels using common sense with respect to the events. For example, even without context, we know that *die* is typically after *explode* and *schedule* typically before *attend*. Ning et al. (2018b) was an initial attempt to acquire such knowledge, by aggregating automatically extracted TempRels from a large corpus. The resulting knowledge base, TEMPROB, contains observed frequencies of tuples $(v1, v2, r)$ representing the probability of verb 1 and verb 2 having relation $r$ and it was shown a useful resource for TempRel extraction.

However, TEMPROB is a simple counting model and fails (or is unreliable) for unseen (or rare) tuples. For example, we may see (ambush, die) less frequently than (attack, die) in a corpus, and the observed frequency of (ambush, die) being *before* or *after* is thus less reliable. However, since "ambush" is semantically similar to "attack", the statistics of (attack, die) can actually serve as an auxiliary signal to (ambush, die). Motivated by this idea, we introduce common sense encoder (CSE): We fit an updated version of TEMPROB via a Siamese network (Bromley et al., 1994) that generalizes to unseen tuples through the resulting embeddings for each verb (Fig. 1c). Note that the TEMPROB we use is reconstructed using the same method described in Ning et al. (2018b) with the base method changed to CogCompTime. Once trained, CSE will remain fixed when training the LSTM part (Fig. 1a or b) and the feedforward neural network part (Fig. 1d). We only use CSE for its output. In the beginning, we tried to directly use the output (i.e., a scalar) and the influence on performance was negligible. Therefore, here we discretize the CSE output, change it to categorical embeddings, concatenate them with the LSTM output, and then produce the confidence scores (Fig. 1d).

## 4 Experiments

### 4.1 Data

The MATRES dataset[5] contains 275 news articles from the TempEval3 workshop (UzZaman et al., 2013) with newly annotated events and TempRels. It has 3 sections: TimeBank (TB), AQUAINT (AQ), and Platinum (PT). We followed the official split (i.e., TB+AQ for training and PT for testing), and further set aside 20% of the training data as the development set to tune learning rates and epochs. We also show our performance on another dataset,

[5] http://cogcomp.org/page/publication_view/834

TCR[6] (Ning et al., 2018a), which contains both temporal and causal relations and we only need the temporal part. The label set for both datasets are *before*, *after*, *equal*, and *vague*.

| | Purpose | #Doc | #Events | #TempRels |
|---|---|---|---|---|
| TB+AQ | Train | 255 | 8K | 13K |
| PT | Test | 20 | 537 | 837 |
| TCR | Test | 25 | 1.3K | 2.6K |

Table 1: TimeBank (TB), AQUAINT (AQ), and Platinum (PT) are from MATRES (Ning et al., 2018c) and TCR from Ning et al. (2018a).

### 4.2 Results and Discussion

We compare with the most recent version of CogCompTime, the state-of-the-art on MATRES.[7] Note that in Table 2, CogCompTime performed slightly different to Ning et al. (2018d): CogCompTime reportedly had $F_1$=65.9 (Table 2 Line 3 therein) and here we obtained $F_1$=66.6. In addition, Ning et al. (2018d) only reported $F_1$ scores, while we also use another two metrics for a more thorough comparison: classification accuracy (acc.) and temporal awareness $F_{\text{aware}}$, where the awareness score is for the graphs represented by a group of related TempRels (more details in the appendix). We also report the average of those three metrics in our experiments.

Table 2 compares the two different ways to handle event positions discussed in Sec. 3.1: position indicators (P.I.) and simple concatenation (Concat), both of which are followed by network (d) in Fig. 1 (i.e., without using Siamese yet). We extensively studied the usage of various pretrained word embeddings, including conventional embeddings (i.e., the medium versions of word2vec, GloVe, and FastText provided in the Magnitude package (Patel et al., 2018)) and contextualized embeddings (i.e., the original ELMo and large uncased BERT, respectively); except for the input embeddings, we kept all other parameters the same. We used cross-entropy loss and the StepLR optimizer in PyTorch that decays the learning rate by 0.5 every 10 epochs (performance not sensitive to it).

Comparing to the previously used P.I. (Dligach et al., 2017), we find that, with only two exceptions (underlined in Table 2), the Concat system saw consistent gains under various embeddings and metrics. In addition, contextualized embeddings (ELMo and BERT) expectedly improved over the conventional ones significantly, although no statistical significance were observed between using ELMo or BERT (more significance tests in Appendix).

| *System* | *Emb.* | *Acc.* | $F_1$ | $F_{\text{aware}}$ | Avg. |
|---|---|---|---|---|---|
| P.I. | word2vec | 63.2 | 67.6 | 60.5 | 63.8 |
| | GloVe | 64.5 | 69.0 | 61.1 | 64.9 |
| | FastText | 60.5 | 64.7 | 59.5 | 61.6 |
| | ELMo | 67.5 | 73.9 | 63.0 | **68.1** |
| | BERT | 68.8 | 73.6 | 61.7 | 68.0 |
| Concat | word2vec | 65.0 | 69.5 | 59.4 | 64.6 |
| | GloVe | 64.9 | 69.5 | 60.9 | 65.1 |
| | FastText | 64.0 | 68.6 | 60.1 | 64.2 |
| | ELMo | 67.7 | 74.0 | 63.3 | 68.3 |
| | BERT | 69.1 | 74.4 | 63.7 | **69.1** |
| Concat+CSE | ELMo | 71.7 | 76.7 | 66.0 | **71.5** |
| | BERT | 71.3 | 76.3 | 66.5 | 71.4 |
| CogCompTime | - | 61.6 | 66.6 | 60.8 | **63.0** |

Table 2: Performances on the MATRES test set (i.e., the PT section). CogCompTime (Ning et al., 2018d) is the previous state-of-the-art feature-based system. Position indicator (P.I.) and concatenation (Concat) are two ways to handle event positions in LSTMs (Sec. 3.1). Concat+CSE achieves significant improvement over CogCompTime on MATRES.

Given the above two observations, we further incorporated our common sense encoder (CSE) into "Concat" with ELMo and BERT in Table 2. We split TEMPROB into train (80%) and validation (20%). The proposed Siamese network (Fig. 1c) was trained by minimizing the cross-entropy loss using Adam (Kingma and Ba, 2014) (learning rate 1e-4, 20 epochs, and batch size 500). We first see that CSE improved on top of Concat for both ELMo and BERT under all metrics, confirming the benefit of TEMPROB; second, as compared to CogCompTime, the proposed Concat+CSE achieved about 10% absolute gains in accuracy and $F_1$, 5% in awareness score $F_{\text{aware}}$, and 8% in the three-metric-average metric, with $p < 0.001$ per the McNemar's test. Roughly speaking, the 8% gain is contributed by LSTMs for 2%, contextualized embeddings for 4%, and CSE for 2%. Again, no statistical significance were observed between using ELMo and BERT. Table 3 furthermore applies CogCompTime and the proposed Concat+CSE system on a different test set called TCR (Ning et al., 2018a). Both systems achieved better scores (suggesting that TCR is easier than MATRES), while the proposed sys-

[6] http://cogcomp.org/page/publication_view/835

[7] http://cogcomp.org/page/publication_view/844

tem still outperformed CogCompTime by roughly 8% under the three-metric-average metric, consistent with our improvement on MATRES.

| *System* | *Emb.* | *Acc.* | $F_1$ | $F_{\text{aware}}$ | Avg. |
|---|---|---|---|---|---|
| CogCompTime | - | 68.1 | 70.7 | 61.6 | **66.8** |
| Concat+CSE | ELMo | 80.8 | 78.6 | 69.9 | **76.4** |
| | BERT | 78.4 | 77.0 | 69.0 | 74.9 |

Table 3: Further evaluation of the proposed system, i.e., Concat (Table 3.1) plus CSE (Sec. 3.2), on the TCR dataset (Ning et al., 2018a).

## 5 Conclusion

Temporal relation extraction has long been an important yet challenging task in natural language processing. Lack of high-quality data and difficulty in the learning problem defined by previous annotation schemes inhibited performance of neural-based approaches. The discoveries that LSTMs readily improve the feature-based state-of-the-art CogCompTime on the MATRES and TCR datasets by a large margin not only give the community a strong baseline, but also indicate that the learning problem is probably better defined by MATRES and TCR. Therefore, we should move along that direction to collect more high-quality data, which can facilitate more advanced learning algorithms in the future.

## Acknowledgements

This research is supported by a grant from the Allen Institute for Artificial Intelligence (allenai.org), the IBM-ILLINOIS Center for Cognitive Computing Systems Research (C3SR) - a research collaboration as part of the IBM AI Horizons Network, and contract HR0011-18-2-0052 with the US Defense Advanced Research Projects Agency (DARPA). Approved for Public Release, Distribution Unlimited. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

## References

Steven Bethard, Leon Derczynski, Guergana Savova, James Pustejovsky, and Marc Verhagen. 2015. SemEval-2015 Task 6: Clinical TempEval. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 806–814.

Steven Bethard, James H Martin, and Sara Klingenstein. 2007. Timelines from text: Identification of syntactic temporal relations. In *Semantic Computing, 2007. ICSC 2007. International Conference on*, pages 11–18. IEEE.

Steven Bethard, Guergana Savova, Wei-Te Chen, Leon Derczynski, James Pustejovsky, and Marc Verhagen. 2016. SemEval-2016 Task 12: Clinical TempEval. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1052–1062.

Steven Bethard, Guergana Savova, Martha Palmer, and James Pustejovsky. 2017. Semeval-2017 task 12: Clinical tempeval. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 565–572. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1994. Signature verification using a "siamese" time delay neural network. In *The Conference on Advances in Neural Information Processing Systems (NIPS)*, pages 737–744.

Taylor Cassidy, Bill McDowell, Nathanel Chambers, and Steven Bethard. 2014. An annotation framework for dense event ordering. In *Proc. of the Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 501–506.

Nathanael Chambers, Taylor Cassidy, Bill McDowell, and Steven Bethard. 2014. Dense event ordering with a multi-pass architecture. *Transactions of the Association for Computational Linguistics*, 2:273–284.

Nathanael Chambers and Dan Jurafsky. 2008a. Jointly combining implicit constraints improves temporal ordering. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*.

Nathanael Chambers and Daniel Jurafsky. 2008b. Unsupervised Learning of Narrative Event Chains. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics, ACL 2008*, pages 789–797.

Nathanael Chambers, Shan Wang, and Dan Jurafsky. 2007. Classifying temporal relations between events. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 173–176. Association for Computational Linguistics.

Fei Cheng and Yusuke Miyao. 2017. Classifying temporal relations by bidirectional LSTM over dependency paths. In *Proc. of the Annual Meeting of the Association of Computational Linguistics (ACL)*, volume 2, pages 1–6.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Thomas G. Dietterich. 1998. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*.

Dmitriy Dligach, Timothy Miller, Chen Lin, Steven Bethard, and Guergana Savova. 2017. Neural temporal relation extraction. volume 2, pages 746–751.

Quang Do, Wei Lu, and Dan Roth. 2012. Joint inference for event timeline construction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Brian S Everitt. 1992. *The analysis of contingency tables*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Artuur Leeuwenberg and Marie-Francine Moens. 2017. Structured learning for temporal relation extraction from clinical records. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*.

Artuur Leeuwenberg and Marie-Francine Moens. 2018. Temporal information extraction by predicting relative time-lines. *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*.

Chen Lin, Timothy Miller, Dmitriy Dligach, Steven Bethard, and Guergana Savova. 2017. Representations of time expressions for temporal relation extraction with convolutional neural networks. *BioNLP 2017*, pages 322–327.

Hector Llorens, Nathanael Chambers, Naushad UzZaman, Nasrin Mostafazadeh, James Allen, and James Pustejovsky. 2015. SemEval-2015 Task 5: QA TEMPEVAL - evaluating temporal information understanding with question answering. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 792–800.

Inderjeet Mani, Marc Verhagen, Ben Wellner, Chong Min Lee, and James Pustejovsky. 2006. Machine learning of temporal relations. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 753–760. Association for Computational Linguistics.

Yuanliang Meng and Anna Rumshisky. 2018. Context-aware neural model for temporal information extraction. In *Proc. of the Annual Meeting of the Association of Computational Linguistics (ACL)*, volume 1, pages 527–536.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Proceedings of NIPS 2013*, pages 1–9.

Anne-Lyse Minard, Manuela Speranza, Eneko Agirre, Itziar Aldabe, Marieke van Erp, Bernardo Magnini, German Rigau, Ruben Urizar, and Fondazione Bruno Kessler. 2015. SemEval-2015 Task 4: TimeLine: Cross-document event ordering. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 778–786.

Nasrin Mostafazadeh, Alyson Grealish, Nathanael Chambers, James Allen, and Lucy Vanderwende. 2016. CaTeRS: Causal and temporal relation scheme for semantic annotation of event structures. In *Proceedings of the 4th Workshop on Events: Definition, Detection, Coreference, and Representation*, pages 51–61.

Qiang Ning, Zhili Feng, and Dan Roth. 2017. A structured learning approach to temporal relation extraction. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 1038–1048, Copenhagen, Denmark.

Qiang Ning, Zhili Feng, Hao Wu, and Dan Roth. 2018a. Joint reasoning for temporal and causal relations. In *Proceedings of the Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 2278–2288.

Qiang Ning, Hao Wu, Haoruo Peng, and Dan Roth. 2018b. Improving temporal relation extraction with a globally acquired statistical resource. In *Proceedings of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*, pages 841–851.

Qiang Ning, Hao Wu, and Dan Roth. 2018c. A multi-axis annotation scheme for event temporal relations. In *Proceedings of the Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 1318–1328.

Qiang Ning, Ben Zhou, Zhili Feng, Haoruo Peng, and Dan Roth. 2018d. Cogcomptime: A tool for understanding time in natural language. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*.

Tim O'Gorman, Kristin Wright-Bettner, and Martha Palmer. 2016. Richer event description: Integrating event coreference with temporal, causal and bridging annotation. In *Proceedings of the 2nd Workshop on Computing News Storylines (CNS 2016)*,

pages 47–56, Austin, Texas. Association for Computational Linguistics.

Ajay Patel, Alexander Sands, Chris Callison-Burch, and Marianna Apidianaki. 2018. Magnitude: A fast, efficient universal vector embedding utility package. *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*.

William F Styler IV, Steven Bethard, Sean Finan, Martha Palmer, Sameer Pradhan, Piet C de Groen, Brad Erickson, Timothy Miller, Chen Lin, Guergana Savova, et al. 2014. Temporal annotation in the clinical domain. *Transactions of the Association for Computational Linguistics*, 2:143.

Julien Tourille, Olivier Ferret, Aurelie Neveol, and Xavier Tannier. 2017. Neural architecture for temporal relation extraction: A bi-lstm approach for detecting narrative containers. In *Proc. of the Annual Meeting of the Association of Computational Linguistics (ACL)*, volume 2, pages 224–230.

Naushad UzZaman, Hector Llorens, James Allen, Leon Derczynski, Marc Verhagen, and James Pustejovsky. 2013. SemEval-2013 Task 1: TEMPEVAL-3: Evaluating time expressions, events, and temporal relations. **SEM*, 2:1–9.

Marc Verhagen, Robert Gaizauskas, Frank Schilder, Mark Hepple, Graham Katz, and James Pustejovsky. 2007. SemEval-2007 Task 15: TempEval temporal relation identification. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 75–80. Association for Computational Linguistics.

Marc Verhagen and James Pustejovsky. 2008. Temporal processing with the TARSQI toolkit. In *22nd International Conference on on Computational Linguistics: Demonstration Papers*, pages 189–192. Association for Computational Linguistics.

Marc Verhagen, Roser Sauri, Tommaso Caselli, and James Pustejovsky. 2010. SemEval-2010 Task 13: TempEval-2. In *Proceedings of the 5th international workshop on semantic evaluation*, pages 57–62. Association for Computational Linguistics.

Zhenqi Xu, Jiani Hu, and Weihong Deng. 2016. Recurrent convolutional neural network for video classification. In *Multimedia and Expo (ICME), 2016 IEEE International Conference on*, pages 1–6. IEEE.

Katsumasa Yoshikawa, Sebastian Riedel, Masayuki Asahara, and Yuji Matsumoto. 2009. Jointly identifying temporal relations with markov logic. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 405–413. Association for Computational Linguistics.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proc. the International Conference on Computational Linguistics (COLING)*, pages 2335–2344.

Dongxu Zhang and Dong Wang. 2015. Relation classification via recurrent neural network. *arXiv preprint arXiv:1508.01006*.

Shu Zhang, Dequan Zheng, Xinchen Hu, and Ming Yang. 2015. Bidirectional long short-term memory networks for relation classification. In *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation*, pages 73–78.

# Improving Fine-grained Entity Typing with Entity Linking

**Hongliang Dai[1], Donghong Du[1,3], Xin Li[2], and Yangqiu Song[1]**
[1]Department of CSE, HKUST
[2]Tencent Technology (SZ) Co., Ltd.
[1]{hdai,yqsong}@cse.ust.hk
[2]alonsoli@tencent.com
[3]dduaa@connect.ust.hk

## Abstract

Fine-grained entity typing is a challenging problem since it usually involves a relatively large tag set and may require to understand the context of the entity mention. In this paper, we use entity linking to help with the fine-grained entity type classification process. We propose a deep neural model that makes predictions based on both the context and the information obtained from entity linking results. Experimental results on two commonly used datasets demonstrates the effectiveness of our approach. On both datasets, it achieves more than 5% absolute strict accuracy improvement over the state of the art.

## 1 Introduction

Given a piece of text and the span of an entity mention in this text, fine-grained entity typing (FET) is the task of assigning fine-grained type labels to the mention (Ling and Weld, 2012). The assigned labels should be context dependent (Gillick et al., 2014). For example, in the sentence "Trump threatens to pull US out of World Trade Organization," the mention "Trump" should be labeled as /person and /person/politician, although Donald Trump also had other occupations such as businessman, TV personality, etc.

This task is challenging because it usually uses a relatively large tag set, and some mentions may require the understanding of the context to be correctly labeled. Moreover, since manual annotation is very labor-intensive, existing approaches have to rely on distant supervision to train models (Ling and Weld, 2012; Ghaddar and Langlais, 2018).

Thus, the use of extra information to help with the classification process becomes very important. In this paper, we improve FET with entity linking (EL). EL is helpful for a model to make typing decisions because if a mention is correctly linked to its target entity, we can directly obtain the type information about this entity in the knowledge base (KB). For example, in the sentence "There were some great discussions on a variety of issues facing Federal Way," the mention "Federal Way" may be incorrectly labeled as a company by some FET models. Such a mistake can be avoided after linking it to the city Federal Way, Washington. For cases that require the understanding of the context, using entity linking results is also beneficial. In the aforementioned example where "Trump" is the mention, obtaining all the types of Donald Trump in the knowledge base (e.g., politician, businessman, TV personality, etc.) is still informative for inferring the correct type (i.e., politician) that fits the context, since they narrows the possible labels down.

However, the information obtained through EL should not be fully trusted since it is not always accurate. Even when a mention is correctly linked to an entity, the type information of this entity in the KB may be incomplete or outdated. Thus, in this paper, we propose a deep neural fine-grained entity typing model that flexibly predicts labels based on the context, the mention string, and the type information from KB obtained with EL.

Using EL also introduces a new problem for the training process. Currently, a widely used approach to create FET training samples is to use the anchor links in Wikipedia (Ling and Weld, 2012; Ren et al., 2016a). Each anchor link is regarded as a mention, and is weakly labeled with all the types of its referred entity (the Wikipedia page the anchor link points to) in KB. Our approach, when links the mention correctly, also uses all the types of the referred entity in KB as extra information. This may cause the trained model to overfit the weakly labeled data. We design a variant of the hinge loss and introduce noise during training to address this problem.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6210–6215, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

We conduct experiments on two commonly used FET datasets. Experimental results show that introducing information obtained through entity linking and having a deep neural model both helps to improve FET performance. Our model achieves more than 5% absolute strict accuracy improvement over the state of the art on both datasets.

Our contributions are summarized as follows:

- We propose a deep neural fine-grained entity typing model that utilizes type information from KB obtained through entity linking.
- We address the problem that our model may overfit the weakly labeled data by using a variant of the hinge-loss and introducing noise during training.
- We demonstrate the effectiveness of our approach with experimental results on commonly used FET datasets.

Our code is available at `https://github.com/HKUST-KnowComp/IFETEL`.

## 2 Related Work

An early effort of classifying named entities into fine-grained types can be found in (Fleischman and Hovy, 2002), which only focuses on person names. Latter, datasets with larger type sets are constructed (Weischedel and Brunstein, 2005; Ling and Weld, 2012; Choi et al., 2018). These datasets are more preferred by recent studies (Ren et al., 2016a; Murty et al., 2018).

Most of the existing approaches proposed for FET are learning based. The features used by these approaches can either be hand-crafted (Ling and Weld, 2012; Gillick et al., 2014) or learned from neural network models (Shimaoka et al., 2017; Xu and Barbosa, 2018; Xin et al., 2018). Since FET systems usually use distant supervision for training, the labels of the training samples can be noisy, erroneous or overly specific. Several studies (Ren et al., 2016b; Xin et al., 2018; Xu and Barbosa, 2018) address these problems by separating clean mentions and noisy mentions, modeling type correction (Ren et al., 2016a), using a hierarchy-aware loss (Xu and Barbosa, 2018), etc.

(Huang et al., 2016) and (Zhou et al., 2018) are two studies that are most related to this paper. Huang et al. (2016) propose an unsupervised FET system where EL is an importat component. But they use EL to help with clustering and type name selection, which is very different from how we use it to improve the performance of a supervised FET model. (Zhou et al., 2018) finds related entities based on the context instead of directly applying EL. The types of these entities are then used for inferring the type of the mention.

## 3 Method

Let $T$ be a predefined tag set, which includes all the types we want to assign to mentions. Given a mention $m$ and its context, the task is to predict a set of types $\boldsymbol{\tau} \subset T$ suitable for this mention. Thus, this is a multi-class, multi-label classification problem (Ling and Weld, 2012). Next, we will introduce our approach for this problem in detail, including the neural model, the training of the model, and the entity linking algorithm we use.

### 3.1 Fine-grained Entity Typing Model

**Input** Each input sample to our FET system contains one mention and the sentence it belongs to. We denote $w_1, w_2, ..., w_n$ as the words in the current sentence, $w_{p_1}, w_{p_2}, ..., w_{p_l}$ as the words in the mention string, where $n$ is the number of words in the sentence, $p_1, ..., p_l$ are the indices of the words in the mention string, $l$ is the number of words in the mention string. We also use a set of pretrained word embeddings.

Our FET approach is illustrated in Figure 1. It first constructs three representations: *context representation*, *mention string representation*, and *KB type representation*. Note that the KB type representation is obtained from a knowledge base through entity linking and is independent of the context of the mention.

**Context Representation** To obtain the context representation, we first use a special token $w_m$ to represent the mention (the token "[Mention]" in Figure 1). Then, the word sequence of the sentence becomes $w_1, ..., w_{p_l-1}, w_m, w_{p_l+1}, ..., w_n$. Their corresponding word embeddings are fed into two layers of BiLSTMs. Let $\boldsymbol{h}_m^1$ and $\boldsymbol{h}_m^2$ be the output of the first and the second layer of BiLSTMs for $w_m$, respectively. We use $\boldsymbol{f}_c = \boldsymbol{h}_m^1 + \boldsymbol{h}_m^2$ as the context representation vector.

**Mention String Representation** Let $\boldsymbol{x}_1, ..., \boldsymbol{x}_l$ be the word embeddings of the mention string words $w_{p_1}, ..., w_{p_l}$. Then the mention string representation $\boldsymbol{f}_s = (\sum_{i=1}^{l} \boldsymbol{x}_i)/l$.

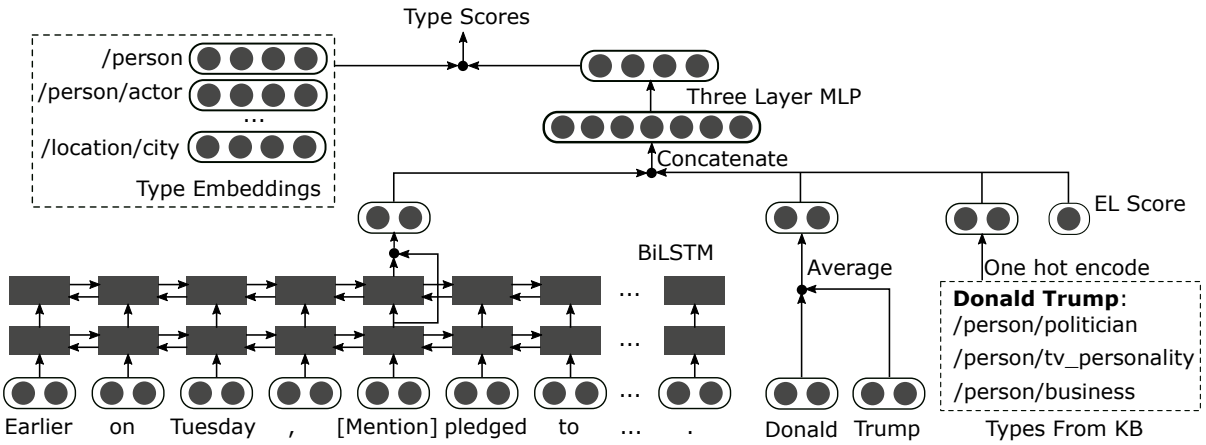


Figure 1: Our approach. The example sentence is "Earlier on Tuesday, *Donald Trump* pledged to help hard-hit U.S. farmers caught in the middle of the escalating trade war." Here, the correct label for the mention *Donald Trump* should be /person, /person/politician. "[Mention]" is a special token that we use to represent the mention.

**KB Type Representation** To obtain the KB type representation, we run an EL algorithm for the current mention. If the EL algorithm returns an entity, we retrieve the types of of this entity from the KB. We use Freebase as our KB[1]. Since the types in Freebase is different from $T$, the target type set, they are mapped to the types in $T$ with rules similar to those used in (Zhou et al., 2018). Afterwards, we perform one hot encoding on these types to get the KB Type Representation $\boldsymbol{f}_e$. If the EL algorithm returns NIL (i.e., the mention cannot be linked to an entity), we simply one hot encode the empty type set.

**Prediction** Apart from the three representations, we also obtain the score returned by our entity linking algorithm, which indicates its confidence on the linking result. We denote it as a one dimensional vector $\boldsymbol{g}$. Then, we get $\boldsymbol{f} = \boldsymbol{f}_c \oplus \boldsymbol{f}_s \oplus \boldsymbol{f}_e \oplus \boldsymbol{g}$, where $\oplus$ means concatenation. $\boldsymbol{f}$ is then fed into an MLP that contains three dense layers to obtain $\boldsymbol{u}_m$, out final representation for the current mention sample $m$. Let $t_1, t_2, ..., t_k$ be all the types in $T$, where $k = |T|$. We embed them into the same space as $\boldsymbol{u}_m$ by assigning each of them a dense vector (Yogatama et al., 2015). These vectors are denoted as $\boldsymbol{t}_1, ..., \boldsymbol{t}_k$. Then the score of the mention $m$ having the type $t_i \in T$ is calculated as the dot product of $\boldsymbol{u}_m$ and $\boldsymbol{t}_i$:

$$s(m, t_i) = \boldsymbol{u}_m \cdot \boldsymbol{t}_i. \quad (1)$$

We predict $t_i$ as a type of $m$ if $s(m, t_i) > 0$.

[1]We use Freebase mainly because it is widely used by existing studies. Wikidata is an alternative.

### 3.2 Model Training

Following existing studies, we also generate training data by using the *anchor links* in Wikipedia. Each anchor link can be used as a mention. These mentions are labeled by mapping the Freebase types of the target entries to the tag set $T$ (Ling and Weld, 2012).

Since the *KB type representations* we use in our FET model are also obtained through mapping Freebase types, they will perfectly match the automatically generated labels for the mentions that are correctly linked (i.e., when the entity returned by the EL algorithm and the target entry of the anchor link are the same). For example, in Figure 1, suppose the example sentence is a training sample obtained from Wikipedia, where "Donald Trump" is an anchor link points to the Wikipedia page of *Donald Trump*. After mapping the Freebase types of *Donald Trump* to the target tag set, this sample will be weakly annotated as /person/politician, /person/tv_personality, and /person/business, which is exactly the same as the type information (the "Types From KB" in Figure 1) obtained through EL. Thus, during training, when the EL system links the mention to the correct entity, the model only needs to output the types in the *KB type representation*. This may cause the trained model to overfit the weakly labeled training data. For most types of entities such as locations and organizations, it is fine since they usually have the same types in different contexts. But it is problematic for person mentions, as their types can be context dependent.

To address this problem, during training, if a

mention is linked to a person entity by our entity linking algorithm, we add a random fine-grained person type label that does not belong to this entity while generating the *KB type representation*. For example, if the mention is linked to a person with types /person/actor and /person/author, a random label /person/politician may be added. This will force the model to still infer the type labels from the context even when the mention is correctly linked, since the *KB type representation* no longer perfectly match the weak labels.

To make it more flexible, we also propose to use a variant of the hinge loss used by (Abhishek et al., 2017) to train our model:

$$L = \sum_{m} [\sum_{t \in \tau_m} \max(0, 1 - s(m,t)) + \sum_{t \in \bar{\tau}_m} \lambda(t) \max(0, 1 + s(m,t))] \quad (2)$$

where $\tau_m$ is the correct type set for mention $m$, $\bar{\tau}_m$ is the incorrect type set. $\lambda(t) \in [1, +\infty)$ is a predefined parameter to impose a larger penalty if the type $t$ is incorrectly predicted as positive. Since the problem of overfitting the weakly annotated labels is more severe for person mentions, we set $\lambda(t) = \lambda_P$ if $t$ is a fine-grained person type, and $\lambda(t) = 1$ for all other types.

During training, we also randomly set the EL results of half of the training samples to be NIL. So that the model can perform well for mentions that cannot be linked to the KB at test time.

### 3.3 Entity Linking Algorithm

In this paper, we use a simple EL algorithm that directly links the mention to the entity with the greatest commonness score. Commonness (Pan et al., 2015; Medelyan and Legg, 2008) is calculated base on the anchor links in Wikipedia. It estimates the probability of an entity given only the mention string. In our FET approach, the commonness score is also used as the confidence on the linking result (i.e., the $\boldsymbol{g}$ used in the prediction part of Subsection 3.1). Within a same document, we also use the same heuristic used in (Ganea and Hofmann, 2017) to find coreferences of generic mentions of persons (e.g., "Matt") to more specific mentions (e.g., "Matt Damon").

We also tried other more advanced EL methods in our experiments. However, they do not improve the final performance of our model. Experimental results of using the EL system proposed in (Ganea and Hofmann, 2017) is provided in Section 4.

## 4 Experiments

### 4.1 Setup

We use two datasets: FIGER (GOLD) (Ling and Weld, 2012) and BBN (Weischedel and Brunstein, 2005). The sizes of their tag sets are 113 and 47, respectively. FIGER (GOLD) allows mentions to have multiple type paths, but BBN does not. Another commonly used dataset, OntoNotes (Gillick et al., 2014), is not used since it contains many pronoun and common noun phrase mentions such as "it," "he," "a thrift institution," which are not suitable to directly apply entity linking on.

Following (Ling and Weld, 2012), we generate weakly labeled datasets for training with Wikipedia anchor links. Since the tag sets used by FIGER (GOLD) and BBN are different, we create a training set for each of them. For each dataset, $2,000$ weakly labeled samples are randomly picked to form a development set. We also manually annotated 50 person mentions collected from news articles for tuning the parameter $\lambda_P$.

We use the 300 dimensional pretrained GloVe word vectors provided by (Pennington et al., 2014). The hidden layer sizes of the two layers of BiLSTMs are both set to 250. For the three-layer MLP, the size of the two hidden layers are both set to 500. The size of the type embeddings is 500. $\lambda_P$ is set to 2.0. We also apply batch normalization and dropout to the input of each dense layer in our three-layer MLP during training.

We use strict accuracy, Macro F1, and Micro F1 to evaluate fine-grained typing performance (Ling and Weld, 2012).

### 4.2 Compared Methods

We compare with the following existing approaches: AFET (Ren et al., 2016a), AAA (Abhishek et al., 2017), NFETC (Xu and Barbosa, 2018), and CLSC (Chen et al., 2019).

We use **Ours (Full)** to represent our full model, and also compare with five variants of our own approach: **Ours (DirectTrain)** is trained without adding random person types while obtaining the KB type representation, and $\lambda_P$ is set to 1; **Ours (NoEL)** does not use entity linking, i.e., the KB type representation and the entity linking confidence score are removed, and the model is trained in DirectTrain style; **Ours (NonDeep)** uses one BiLSTM layer and replaces the MLP with a dense layer; **Ours (NonDeep NoEL)** is the NoEL version of *Ours (NonDeep)*; **Ours (LocAttEL)** uses

| Dataset | FIGER (GOLD) | | | BBN | | |
|---|---|---|---|---|---|---|
| Approach | Accuracy | Macro F1 | Micro F1 | Accuracy | Macro F1 | Micro F1 |
| AFET | 53.3 | 69.3 | 66.4 | 67.0 | 72.7 | 73.5 |
| AAA | 65.8 | 81.2 | 77.4 | 73.3 | 79.1 | 79.2 |
| NFETC | 68.9 | 81.9 | 79.0 | 72.1 | 77.1 | 77.5 |
| CLSC | - | - | - | 74.7 | 80.7 | 80.5 |
| Ours (NonDeep NoEL) | 65.9 | 81.7 | 78.0 | 69.3 | 81.4 | 81.5 |
| Ours (NonDeep) | 72.3 | 85.4 | 82.6 | 79.1 | 87.9 | 88.4 |
| Ours (DirectTrain) | 69.1 | 85.2 | 82.2 | - | - | - |
| Ours (NoEL) | 69.8 | 82.7 | 80.4 | 80.5 | 87.5 | 88.0 |
| Ours (LocAttEL) | 75.1 | 86.3 | 83.9 | **82.8** | 88.9 | 89.5 |
| Ours (Full) | **75.5** | **87.1** | **84.6** | 82.5 | **89.2** | **89.6** |

Table 1: Fine-grained entity typing performance. The performance of "Ours (DirectTrain)" on BBN is omitted since this dataset does not have fine-grained types for person.

the entity linking approach proposed in (Ganea and Hofmann, 2017) instead of our own commonness based approach. *Ours (Full)*, *Ours (DirectTrain)*, and *Ours (NonDeep)* all use our own commonness based entity linking approach.

### 4.3 Results

The experimental results are listed in Table 1. As we can see, our approach performs much better than existing approaches on both datasets.

The benefit of using entity linking in our approach can be verified by comparing *Ours (Full)* and *Ours (NoEL)*. The performance on both datasets decreases if the entity linking part is removed. Especially on FIGER (GOLD), the strict accuracy drops from 75.5 to 69.8. Using entity linking improves less on BBN. We think this is because of three reasons: 1) BBN has a much smaller tag set than FIGER (GOLD); 2) BBN does not allow a mention to be annotated with multiple type paths (e.g., labeling a mention with both /building and /location is not allowed), thus the task is easier; 3) By making the model deep, the performance on BBN is already improved a lot, which makes further improvement harder.

The improvement of our full approach over *Ours (DirectTrain)* on FIGER (GOLD) indicates that the techniques we use to avoid overfitting the weakly labeled data are also effective.

*Ours (LocAttEL)*, which uses a more advanced EL system, does not achieve better performance than *Ours (Full)*, which uses our own EL approach. After manually checking the results of the two EL approaches and the predictions of our model on FIGER (GOLD), we think this is mainly because: 1) Our model also uses the context while making predictions. Sometimes, if it "thinks" that the type information provided by EL is incorrect, it may not use it. 2) The performances of different EL approaches also depends on the dataset and the types of entities used for evaluation. We find that on FIGER (GOLD), the approach in (Ganea and Hofmann, 2017) is better at distinguishing locations and sports teams, but it may also make some mistakes that our simple EL method does not. For example, it may incorrectly link "March," the month, to an entity whose Wikipedia description fits the context better. 3) For some mentions, although the EL system links it to an incorrect entity, the type of this entity is the same with the correct entity.

## 5 Conclusions

We propose a deep neural model to improve fine-grained entity typing with entity linking. The problem of overfitting the weakly labeled training data is addressed by using a variant of the hinge loss and introducing noise during training. We conduct experiments on two commonly used dataset. The experimental results demonstrates the effectiveness of our approach.

## Acknowledgments

This paper was supported by the Early Career Scheme (ECS, No. 26206717) from Research Grants Council in Hong Kong and WeChat-HKUST WHAT Lab on Artificial Intelligence Technology.

## References


Abhishek Abhishek, Ashish Anand, and Amit Awekar. 2017. Fine-grained entity type classification by jointly learning representations and label embeddings. In *Proceedings of EACL*, volume 1, pages 797–807.

Bo Chen, Xiaotao Gu, Yufeng Hu, Siliang Tang, Guoping Hu, Yueting Zhuang, and Xiang Ren. 2019. Improving distantly-supervised entity typing with compact latent space clustering. In *Proceedings of NAACL*, page 28622872.

Eunsol Choi, Omer Levy, Yejin Choi, and Luke Zettlemoyer. 2018. Ultra-fine entity typing. In *Proceedings of ACL*, pages 87–96.

Michael Fleischman and Eduard Hovy. 2002. Fine grained classification of named entities. In *Proceedings of COLING*, pages 1–7. Association for Computational Linguistics.

Octavian-Eugen Ganea and Thomas Hofmann. 2017. Deep joint entity disambiguation with local neural attention. In *Proceedings of EMNLP*, pages 2619–2629.

Abbas Ghaddar and Phillippe Langlais. 2018. Transforming wikipedia into a large-scale fine-grained entity type corpus. In *Proceedings of LREC*.

Dan Gillick, Nevena Lazic, Kuzman Ganchev, Jesse Kirchner, and David Huynh. 2014. Context-dependent fine-grained entity type tagging. *arXiv preprint arXiv:1412.1820*.

Lifu Huang, Jonathan May, Xiaoman Pan, and Heng Ji. 2016. Building a fine-grained entity typing system overnight for a new x (x= language, domain, genre). *arXiv preprint arXiv:1603.03112*.

Xiao Ling and Daniel S Weld. 2012. Fine-grained entity recognition. In *Proceedings of AAAI*, volume 12, pages 94–100.

Olena Medelyan and Catherine Legg. 2008. Integrating cyc and wikipedia: Folksonomy meets rigorously defined common-sense. *Wikipedia and Artificial Intelligence: An Evolving Synergy*, page 13.

Shikhar Murty, Patrick Verga, Luke Vilnis, Irena Radovanovic, and Andrew McCallum. 2018. Hierarchical losses and new resources for fine-grained entity typing and linking. In *Proceedings of ACL*, volume 1, pages 97–109.

Xiaoman Pan, Taylor Cassidy, Ulf Hermjakob, Heng Ji, and Kevin Knight. 2015. Unsupervised entity linking with abstract meaning representation. In *Proceedings of NAACL*, pages 1130–1139.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*, pages 1532–1543.

Xiang Ren, Wenqi He, Meng Qu, Lifu Huang, Heng Ji, and Jiawei Han. 2016a. AFET: Automatic fine-grained entity typing by hierarchical partial-label embedding. In *Proceedings of EMNLP*, pages 1369–1378.

Xiang Ren, Wenqi He, Meng Qu, Clare R Voss, Heng Ji, and Jiawei Han. 2016b. Label noise reduction in entity typing by heterogeneous partial-label embedding. In *Proceedings of SIGKDD*, pages 1825–1834. ACM.

S Shimaoka, P Stenetorp, K Inui, and S Riedel. 2017. Neural architectures for fine-grained entity type classification. In *Proceedings of EACL*, volume 15, pages 1271–1280. Association for Computational Linguistics.

Ralph Weischedel and Ada Brunstein. 2005. BBN pronoun coreference and entity type corpus. *Linguistic Data Consortium, Philadelphia*.

Ji Xin, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2018. Improving neural fine-grained entity typing with knowledge attention. In *Proceedings of AAAI*.

Ji Xin, Hao Zhu, Xu Han, Zhiyuan Liu, and Maosong Sun. 2018. Put it back: Entity typing with language model enhancement. In *Proceedings of EMNLP*, pages 993–998.

Peng Xu and Denilson Barbosa. 2018. Neural fine-grained entity type classification with hierarchy-aware loss. In *Proceedings of NAACL-HLT*, volume 1, pages 16–25.

Dani Yogatama, Daniel Gillick, and Nevena Lazic. 2015. Embedding methods for fine grained entity type classification. In *Proceedings of ACL*, volume 2, pages 291–296.

Ben Zhou, Daniel Khashabi, Chen-Tse Tsai, and Dan Roth. 2018. Zero-shot open entity typing as type-compatible grounding. In *Proceedings of EMNLP*, pages 2065–2076.

# Combining Spans into Entities: A Neural Two-Stage Approach for Recognizing Discontiguous Entities

**Bailin Wang**
ILCC, School of Informatics
University of Edinburgh
bailin.wang@ed.ac.uk

**Wei Lu**
StatNLP Research Group
Singapore University of Technology and Design
luwei@sutd.edu.sg

## Abstract

In medical documents, it is possible that an entity of interest not only contains a discontiguous sequence of words but also overlaps with another entity. Entities of such structures are intrinsically hard to recognize due to the large space of possible entity combinations. In this work, we propose a neural two-stage approach to recognizing discontiguous and overlapping entities by decomposing this problem into two subtasks: 1) it first detects all the overlapping spans that either form entities on their own or present as segments of discontiguous entities, based on the representation of segmental hypergraph, 2) next it learns to combine these segments into discontiguous entities with a classifier, which filters out other incorrect combinations of segments. Two neural components are designed for these subtasks respectively and they are learned jointly using a shared encoder for text. Our model achieves the state-of-the-art performance in a standard dataset, even in the absence of external features that previous methods used.

## 1 Introduction

Named entity recognition (NER) aims at identifying shallow semantic elements in text and has been a crucial step towards natural language understanding (Tjong Kim Sang and De Meulder, 2003). Extracted entities can facilitate various downstream tasks like question answering (Abney et al., 2000), relation extraction (Mintz et al., 2009; Liu et al., 2017), event extraction (Riedel and McCallum, 2011; Lu and Roth, 2012; Li et al., 2013), and coreference resolution (Soon et al., 2001; Ng and Cardie, 2002; Chang et al., 2013).

The underlying assumptions behind most NER systems are that an entity should contain a contiguous sequence of words and should not overlap with each other. However, such assumptions do not always hold in practice. First, entities or mentions[1] with overlapping structures frequently exist in news (Doddington et al., 2004) and biomedical documents (Kim et al., 2003). Second, entities can be discontiguous, especially in clinical texts (Pradhan et al., 2014b). For example, Figure 3 shows three entities where two of them are discontiguous ("*laceration ... esophagus*" and "*stomach ... lac*"), and the second discontiguous entity also overlaps with another entity ("*blood in stomach*").

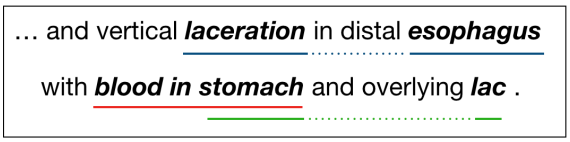


Figure 1: Entities are highlighted with colored underlines. "*laceration ... esophagus*" and "*stomach ... lac*" contain discontiguous sequence of words and the latter also overlaps with another entity "*blood in stomach*".

Such discontiguous entities are intrinsically hard to recognize considering the large search space of possible combinations of entities that have discontiguous and overlapping structures. Muis and Lu (2016a) proposed a hypergraph-based representation to compactly encode discontiguous entities. However, this representation suffers from the ambiguity issue during decoding – one particular hypergraph corresponds to multiple interpretations of entity combinations. As a result, it resorted to heuristics to deal with such an issue.

Motivated by their work, we take a novel approach to resolve the ambiguity issue in this work. Our core observation is that though it is hard to exactly encode the exponential space of all possible discontiguous entities, recent work on extracting overlapping structures (Wang and Lu, 2018) can be employed to efficiently explore the space

[1]Mentions are defined as references to entities that could be named, nominal or pronominal (Florian et al., 2004).



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6216–6224, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

of all the span combinations of discontiguous entities. Based on this observation, we decompose the problem of recognizing discontiguous entities into two subtasks: 1) *segment extraction*: learning to detect all (potentially overlapping) spans that either form entities on their own or present as parts of a discontiguous entity; 2) *segment merging*: learning to form entities by merging certain spans into discontiguous entities.

Our contributions are summarized as follows:

- By decomposing the problem of extracting discontiguous entities into two subtasks, we propose a two-stage approach that does not have the ambiguity issue.
- Under this decomposition, we design two neural components for these two subtasks respectively. We further show that the joint learning setting where the two components use a shared text encoder is beneficial.
- Empirical results show that our system achieves a significant improvement compared with previous methods, even in the absence of external features that previous methods used. [2]

Though we only focus on discontiguous entity recognition in this work, our model may find applications in other tasks that involve discontiguous structures, such as detecting gappy multiword expressions (Schneider et al., 2014).

## 2 Related Work

The task of extracting overlapping entities has long been studied (Zhang et al., 2004; Zhou et al., 2004; Zhou, 2006; McDonald et al., 2005; Alex et al., 2007; Finkel and Manning, 2009; Lu and Roth, 2015; Muis and Lu, 2017). As neural models (Collobert et al., 2011; Lample et al., 2016; Huang et al., 2015; Chiu and Nichols, 2016; Ma and Hovy, 2016) are proven effective for NER, there have been several neural systems recently proposed to handle entities of overlapping structures (Ju et al., 2018; Katiyar and Cardie, 2018; Wang et al., 2018; Sohrab and Miwa, 2018; Wang and Lu, 2018; Straková et al., 2019; Lin et al., 2019; Fisher and Vlachos, 2019). Our system is based on the model of neural segmental hypergraphs (Wang and Lu, 2018) which encodes all the possible combinations of overlapping entities using a compact hypergraph representation without ambiguity. Note that other system for extracting overlapping structures can also fit into our two-stage system.

For discontiguous and overlapping entity recognition, Tang et al. (2013); Zhang et al. (2014); Xu et al. (2015) extended the BIO tagging scheme to encode such complex structures so that traditional linear-chain CRF (Lafferty et al., 2001) can be employed. However, the model suffers greatly from ambiguity during decoding due to the use of the extended tagset. Muis and Lu (2016a) proposed a hypergraph-based representation to reduce the level of ambiguity. Essentially, these systems trade expressiveness for efficiency: they inexactly encoded the whole space of discontiguous entities with ambiguity for training, and then relied on some heuristics to handle the ambiguity during decoding. [3] Considering it is intrinsically hard to exactly identify discontiguous entities in one stage using a structured model, our work tries to decompose the task into two sub-tasks to resolve the ambiguity issue.

This task is also related to joint entity and relation extraction (Kate and Mooney, 2010; Li and Ji, 2014; Miwa and Sasaki, 2014) where the discontiguous entities can be viewed as relation links between segments. The major difference is that discontiguous entities require explicitly modeling overlapping entities and linking multiple segments.

## 3 Model

Our goal is to extract a set of entities that may have overlapping and discontiguous structures given a natural language sentence. We use $\boldsymbol{x} = x_1 \dots x_{|\boldsymbol{x}|}$ to denote a sentence, and use $\boldsymbol{y} = \{[b_{i:j} \dots b_{m:n}]^k\}$ to denote a set of discontiguous entities where each entity of type $k$ contains a list of spans, e.g., $b_{i:j}$ and $b_{m:n}$, with subscripts indicating the starting and ending positions of the span. Hence, this task can be viewed as extracting and labelling a sequence of spans as an entity.

Our two-stage approach first extracts spans of interest like $b_{i:j}$, which are parts of discontiguous entities. Then it merges these extracted spans into discontiguous entities. In the more general setting where discontiguous entities are typed, our

[2] Our code is available at https://github.com/berlino/disco_em19.

[3] We will briefly introduce them and their heuristics later as our baselines in our experiments for comparison.

approach is designed to jointly extract and label the spans at the first stage, then only merge the spans of the same type at the second stage. We call the intermediate typed span $b^k_{i:j}$ a *segment* in the rest of the paper.

Formally, our model aims at maximizing the conditional probability $p(\boldsymbol{y}|\boldsymbol{x})$, which is decomposed as:

$$p(\boldsymbol{y}|\boldsymbol{x}) = p(\boldsymbol{s}|\boldsymbol{x})p(\boldsymbol{y}|\boldsymbol{s},\boldsymbol{x}) \quad (1)$$

where $\boldsymbol{s} = \{b^k_{i:j}\}$ denotes the set of segments that leads to $\boldsymbol{y}$ through a specific combination. [4] That is, we divide the problem of extracting discontiguous entities into two subtasks, namely *segment extraction* and *segment merging*.

### 3.1 Segment Extraction

The entity segments $\boldsymbol{s}$ of interest in a given sentence could also overlap with each other. For example, in Figure 3 the entity "*blood in stomach*" contains another segment "*stomach*". To make our model capable of extracting such overlapping segment combinations, we employ the model of neural segmental hypergraphs from Wang and Lu (2018), which uses a hypergraph-based representation to encode all the possible combinations of segments without ambiguity. Specifically, the segmental hypergraphs adopt a log-linear approach to model the conditional probability of each segment combination for a given sentence:

$$p(\boldsymbol{s}|\boldsymbol{x}) = \frac{\exp f(\boldsymbol{x},\boldsymbol{s})}{\sum_{\boldsymbol{s}'} \exp f(\boldsymbol{x},\boldsymbol{s}')} \quad (2)$$

where $f(\boldsymbol{x},\boldsymbol{s})$ is the score function for any pair of input sentence $\boldsymbol{x}$ and output segment combination $\boldsymbol{s}$.

In segmental hypergraphs, each segment combination $\boldsymbol{s}$ corresponds to a hyperpath. Following Wang and Lu (2018), the score for a hyperpath is the sum of the scores for each hyperedge, which are based on the word-level and span-level representations through LSTM (Graves and Schmidhuber, 2005):

$$\mathbf{h}^w_i = [\text{biLSTM}_1(\mathbf{x}_0, ..., \mathbf{x}_n)]_i \quad (3)$$
$$\mathbf{h}^s_{i:j} = \text{biLSTM}_2(\mathbf{h}^w_i, ..., \mathbf{h}^w_j) \quad (4)$$

where $\mathbf{x}_k$ is the corresponding word embedding for word $x_k$, $\mathbf{h}^w_i$ denotes the representation for the $i$-th word and $\mathbf{h}^s_{i:j}$ denotes the representation for the span from the $i$-th to the $j$-th word.

[4] We note that each $\boldsymbol{y}$ corresponds to one unique $\boldsymbol{s}$.

| | # *sents* | # *entities* (%) | | | # *o.l.* (%) |
|---|---|---|---|---|---|
| | | 1 *segment* | 2 *segments* | 3 *segments* | |
| Train | 534 | 544 (46) | 607 (51) | 44 (4) | 205 (17) |
| Dev | 303 | 357 (45) | 421 (53) | 18 (2) | 240 (30) |
| Test | 430 | 584 (48) | 610 (50) | 16 (1) | 327 (27) |

Table 1: Statistics of the dataset. *o.l.*: overlapping entities, *sents*: sentences.

On top of the segmental hypergraph representation, the partition function which is the denominator of Equation 2 can be computed using dynamic programming. The inference algorithm has a quadratic time complexity in the number of words, which can be further reduced to linear time complexity if we introduce the maximal length $c$ of a segment. We regard $c$ as a hyperparameter.

### 3.2 Segment Merging

Given a set of segments, our next subtask is to merge them into entities. First, we enumerate all the valid segment combinations, denoted as $\mathbf{E}$, based on the assumption that the segments in the same entity should have the same type and not overlap with each other. Our model then independently decides whether each valid segment combination forms an entity. We call these valid segment combinations *entity candidates*. For brevity, let us use $\boldsymbol{t}^k$ to denote an entity candidate $[b_{i:j} \ldots b_{m:n}]^k$ where each segment like $b^k_{i:j}$ belongs to $\boldsymbol{s}$.

Formally, given segments $\boldsymbol{s}$ , the probability of generating entities $\boldsymbol{y}$ can be represented as:

$$p(\boldsymbol{y}|\boldsymbol{s},\boldsymbol{x}) = \prod_{\boldsymbol{t}^k \in \mathbf{E}} \big(\mathbb{1}[\boldsymbol{t}^k \in \boldsymbol{y}] p(\boldsymbol{t}^k \in \boldsymbol{y}) + \mathbb{1}[\boldsymbol{t}^k \notin \boldsymbol{y}](1 - p(\boldsymbol{t}^k \in \boldsymbol{y}))\big) \quad (5)$$

where $\mathbb{1}$ is an indicator function. We use a binary classifier to model $p(\boldsymbol{t}^k \in \boldsymbol{y})$.

To capture the interactions between segments within the same combination, we employ yet another LSTM on top of segments as follows:

$$\mathbf{h}^e_{\boldsymbol{t}^k} = \text{biLSTM}_3(\mathbf{h}^s_{i:j}, ..., \mathbf{h}^s_{m:n}) \quad (6)$$

where $\mathbf{h}^e_{\boldsymbol{t}^k}$ denotes the representation of the segment combination $\boldsymbol{t}^k$, which then serves as a feature vector for a binary classifier to determine whether it is an entity. Note that we reuse the span

representation from Equation 4, meaning that encoder for words and spans are shared in both segment extraction and merging.

The binary classifier for each $\boldsymbol{t}^k$ in Equation 5 is computed as:

$$p(\boldsymbol{t}^k \in \boldsymbol{y}) = \text{sigmoid}(\mathbf{W} \cdot \text{relu}(\mathbf{h}^e_{\boldsymbol{t}^k}) + b) \quad (7)$$

where we use a rectified linear unit (ReLU) (Glorot et al., 2011) and a linear layer, parameterized by $\mathbf{W}$ and $b$, to map the representation from Equation 6 to a scalar score. This score is normalized into a distribution by the sigmoid function.

In the joint model, we stack three separate LSTMs to encode text at different levels from words to spans, then discontiguous entities. Intuitively, the word and span level LSTM try to capture the lower-level information for segment extraction while the entity level LSTM captures the higher-level information for segment merging.

### 3.3 Learning and Decoding

For a dataset $\mathcal{D}$ consisting of sentence-entities pairs $(\boldsymbol{x}, \boldsymbol{y})$, our objective is to minimize the negative log-likelihood as follows:

$$\mathcal{L}(\theta) = -\sum_{(\boldsymbol{x}_i, \boldsymbol{y}_i) \in \mathcal{D}} \log p(\boldsymbol{y}_i | \boldsymbol{x}_i) + \frac{\lambda}{2}\|\theta\|^2 \quad (8)$$

where $\theta$ denotes model parameters and $\lambda$ is the $\ell_2$ coefficient. $p(\boldsymbol{y}_i|\boldsymbol{x}_i)$ is computed by $p(\boldsymbol{s}_i|\boldsymbol{x}_i)p(\boldsymbol{y}_i|\boldsymbol{s}_i, \boldsymbol{x}_i)$ where $\boldsymbol{s}_i$ is inferred from $\boldsymbol{y}_i$.

During the decoding stage, the system first predicts the most probable segments from the neural segmental hypergraph by $\hat{\boldsymbol{s}} = \arg\max_{\boldsymbol{s}'} p(\boldsymbol{s}'|\boldsymbol{x})$. Then it feeds the prediction to the next stage of merging segments and outputs the discontiguous entities by $\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}'} p(\boldsymbol{y}'|\hat{\boldsymbol{s}}, \boldsymbol{x})$.

## 4 Experiments

### 4.1 Setup

**Data** We evaluated our model on the task of recognizing mentions in clinical text from ShARe/CLEF eHealth Evaluation Lab (SHEL) 2013 (Suominen et al., 2013) and SemEval-2014 (Pradhan et al., 2014a). The task is defined to extract mentions of *disorders* from clinical documents according to the Unified Medical Language System (UMLS). The original dataset only has a small percentage of discontiguous entities, making it not suitable for comparing the effectiveness of different models when handling discontiguous entities. Following Muis and Lu (2016a), we use a subset of the original data where each sentence contains at least one discontiguous entity.

We split the dataset according to the setting of SemEval 2014. Statistics are shown in Table 1. In this subset, 53.6% of entities are discontiguous. Overlapping entities also frequently appear. Since an entity has three segments at most, we make the constraint that an entity candidate has no more than three segments during segment merging.

Note that all entities hold the same type of *disorder* in this dataset. Our model is intrinsically able to handle discontiguous entities of multiple types. Recall that segments are typed as $b^k_{i:j}$ during segment extraction, and only segments of the same type can be merged into an entity $\boldsymbol{t}^k$ where $k$ indicates the entity type. To assess its ability to deal with multiple entity types, we conducted a further analysis (see section 4.2).

**Hyperparameters** We use the pretrained word embeddings from Chiu et al. (2016) which are trained on the PubMed corpus. A dropout layer (Srivastava et al., 2014) is used after each word is mapped to its embedding. The dropout rate and the number of hidden units in LSTMs are tuned based on the performance on the development set. We set the maximal length of a segment to be 6 during segment extraction. Our model is trained with Adam (Kingma and Ba, 2014). [5]

**Baselines** The first baseline we consider is to extend the traditional BIO tagging scheme to seven tags following Tang et al. (2013). With this tagging scheme, each word in a sentence is assigned a label. Then a linear-chain CRF is built to model the sequence labelling process. The next baseline is a hypergraph-based method by Muis and Lu (2016a). It encodes each entity combination into a directed graph based on six types of nodes; each has its specific semantics.

Since these baselines are both ambiguous, heuristics are required during decoding. Following Muis and Lu (2016a), we explored two heuristics: given a model's ambiguous output, either a tag sequence or a hypergraph, the "enough" heuristic finds the minimal set of entities that corresponds to it, while "all" decodes the union of all the possible set of entities. Please refer to Muis and Lu (2016a) for details. We also describe them in the Appendix for self-containedness.

[5] See Appendix C for the full hyperparameters.

| | Model | $P$ | $R$ | $F_1$ |
|---|---|---|---|---|
| Non-Neural | CRF (enough) | 54.7 | 41.2 | 47.0 |
| | CRF (all) | 15.2 | 44.9 | 22.7 |
| | Graph (enough) | **76.9** | 40.1 | 52.7 |
| | Graph (all) | 76.0 | 40.5 | 52.8 |
| | *Our model* | 76.3 | 41.4 | 53.7 |
| Neural | CRF (enough) | 43.7 | 54.3 | 48.4 |
| | CRF (all) | 15.7 | 55.8 | 24.5 |
| | *Our model* | 48.4 | **66.5** | **56.1** |
| | *w.o. shared encoder* | 46.2 | 65.1 | 54.0 |

Table 2: Main results. Graph: the hypergraph based model by Muis and Lu (2016a). "enough" and "all" denotes the heuristics used in ambiguous model. *w.o. shared encoder*: without using shared encoder.

We compare our approach to these baselines in two settings. In the non-neural setting, we compare models using the same set of handcrafted features, including external features from POS tagger and Brown cluster following (Muis and Lu, 2016a). In the neural setting, we implement a linear-chain CRF model using the same neural encoder. We are trying to see our model can perform better in both settings. Note that all neural models in our experiments do not leverage any handcrafted features.

### 4.2 Results and Analysis

The main results are listed in Table 2. In both non-neural and neural settings, our model achieves the better result in terms of $F_1$ compared with other baselines, revealing the effectiveness of our methodology of decomposing the task into two stages. Our neural model achieves the best performance even without using any external handcrafted features.

We also assess the performance when our model uses separate encoders for segment extraction and merging. From the results, we observe that the setting of using a shared encoder is very beneficial for our two-stage system.

Compared with non-neural models, neural models are better in terms of $F_1$, both for CRF and our models. The gain mostly comes from the ability to recall more entities. Handcrafted features in non-neural models lead to high precisions but do not seem to be general enough to recall most entities.

The "enough" heuristic works better than "all" in most cases. Hence we use it for evaluating models' ability in handling multiple entity types.

**Handling Multiple Entity Types** To assess the effectiveness of handling entities of multiple types, we further categorize each entity into three types based on its Concept Unique Identifier (CUI), following Muis and Lu (2016a). [6] In this setting, segments are jointly extracted and labelled using these three categories during segment extraction. During segment merging, an entity candidate can only contain segments of the same type during merging.

| | Model | $P$ | $R$ | $F_1$ |
|---|---|---|---|---|
| Non-Neural | CRF (enough) | 55.3 | 37.4 | 44.6 |
| | Graph (enough) | **67.3** | 37.5 | 48.2 |
| Neural | CRF (enough) | 41.6 | 52.3 | 46.3 |
| | *Our model* | 43.3 | **65.8** | **52.2** |

Table 3: Results on handling multiple entity types.

The results are listed in Table 3. Our neural model again achieves the best performance among all models in terms of $F_1$. Compared with neural CRF, our model is significantly better at recalling entities. Similar to the previous observation, the neural encoder consistently boosts the performance of the CRF by recalling more entities, compared with its non-neural counterpart.

## 5 Conclusion and Future Work

In this work, we propose a neural two-stage approach for recognizing discontiguous entities, which learns to extract and merge segments jointly without suffering from ambiguity issue. Empirically, it achieves a significant improvement compared with previous methods that rely heavily on handcrafted features.

During training, the classifier of merging segments is only exposed to correct segments, making it unable to recover from errors of segment exaction during decoding. This issue is similar to *exposure bias* (Wiseman and Rush, 2016) and it might be beneficial if the classifier of segment merging is exposed to incorrect segments during training. We leave this for future work.

## Acknowledgements

We would like to thank the anonymous reviewers for their valuable comments. Wei Lu is supported by Singapore Ministry of Education Academic Research Fund (AcRF) Tier 2 Project MOE2017-T2-1-156, and is partially supported by SUTD project PIE-SGP-AI-2018-01.

[6] Note that the label of CUI is not available for all entities. Entities without CUI are assigned with a default *NULL* label.

## References


Steven Abney, Michael Collins, and Amit Singhal. 2000. Answer extraction. In *Proc. of the sixth conference on applied natural language processing*.

Beatrice Alex, Barry Haddow, and Claire Grover. 2007. Recognising nested named entities in biomedical text. In *Proc. of BioNLP*.

Kai-Wei Chang, Rajhans Samdani, and Dan Roth. 2013. A constrained latent variable model for coreference resolution. In *Proc. of EMNLP*.

Billy Chiu, Gamal Crichton, Anna Korhonen, and Sampo Pyysalo. 2016. How to train good word embeddings for biomedical nlp. In *Proc. of the 15th Workshop on Biomedical Natural Language Processing*.

Jason PC Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *TACL*, 4:357–370.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *JMLR*, 12(Aug):2493–2537.

George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie Strassel, and Ralph M Weischedel. 2004. The automatic content extraction (ace) program-tasks, data, and evaluation. In *Proc. of LREC*.

Jenny Rose Finkel and Christopher D Manning. 2009. Nested named entity recognition. In *Proc. of EMNLP*.

Joseph Fisher and Andreas Vlachos. 2019. Merge and label: A novel neural network architecture for nested NER. In *Proc. of ACL*, pages 5840–5850.

R. Florian, H. Hassan, A. Ittycheriah, H. Jing, N. Kambhatla, X. Luo, N. Nicolov, and S. Roukos. 2004. A statistical model for multilingual entity detection and tracking. In *Proc. of HLT-NAACL*.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proc. of AISTATS*.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5-6):602–610.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Meizhi Ju, Makoto Miwa, and Sophia Ananiadou. 2018. A neural layered model for nested named entity recognition. In *Proc. of NAACL-HLT*.

Rohit J Kate and Raymond J Mooney. 2010. Joint entity and relation extraction using card-pyramid parsing. In *Proc. of CoNLL*.

Arzoo Katiyar and Claire Cardie. 2018. Nested named entity recognition revisited. In *Proc. of NAACL-HLT*.

J-D Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. 2003. Genia corpus—a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(suppl_1):i180–i182.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proc. of ICLR*.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proc. of NAACL-HLT*.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proc. of ACL*.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proc. of ACL*.

Hongyu Lin, Yaojie Lu, Xianpei Han, and Le Sun. 2019. Sequence-to-nuggets: Nested entity mention detection via anchor-region networks. In *Proc. of ACL*, pages 5182–5192.

Liyuan Liu, Xiang Ren, Qi Zhu, Shi Zhi, Huan Gui, Heng Ji, and Jiawei Han. 2017. Heterogeneous supervision for relation extraction: A representation learning approach. In *Proc. of EMNLP*.

Wei Lu and Dan Roth. 2012. Automatic event extraction with structured preference modeling. In *Proc. of ACL*.

Wei Lu and Dan Roth. 2015. Joint mention extraction and classification with mention hypergraphs. In *Proc. of EMNLP*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proc. of ACL*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Flexible text segmentation with structured multilabel classification. In *Proc. of HLT-EMNLP*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proc. of ACL-IJCNLP*.

Makoto Miwa and Yutaka Sasaki. 2014. Modeling joint entity and relation extraction with table representation. In *Proc. of EMNLP*.

Aldrian Obaja Muis and Wei Lu. 2016a. Learning to recognize discontiguous entities. In *Proc. of EMNLP*.

Aldrian Obaja Muis and Wei Lu. 2016b. Supplementary material for recognizing overlapping mentions with mention separators.

Aldrian Obaja Muis and Wei Lu. 2017. Labeling gaps between words: Recognizing overlapping mentions with mention separators. In *Proc. of EMNLP*.

Vincent Ng and Claire Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proc. of ACL*.

Sameer Pradhan, Noémie Elhadad, Wendy Chapman, Suresh Manandhar, and Guergana Savova. 2014a. Semeval-2014 task 7: Analysis of clinical text. In *Proc. of SemEval*.

Sameer Pradhan, Noémie Elhadad, Brett R South, David Martinez, Lee Christensen, Amy Vogel, Hanna Suominen, Wendy W Chapman, and Guergana Savova. 2014b. Evaluating the state of the art in disorder recognition and normalization of the clinical narrative. *Journal of the American Medical Informatics Association*, 22(1):143–154.

S. Riedel and A. McCallum. 2011. Fast and robust joint models for biomedical event extraction. In *Proc. of EMNLP*.

Nathan Schneider, Emily Danchik, Chris Dyer, and Noah A Smith. 2014. Discriminative lexical semantic segmentation with gaps: running the mwe gamut. *TACL*, 2:193–206.

Mohammad Golam Sohrab and Makoto Miwa. 2018. Deep exhaustive model for nested named entity recognition. In *Proc. of EMNLP*.

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational linguistics*, 27(4):521–544.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958.

Jana Straková, Milan Straka, and Jan Hajic. 2019. Neural architectures for nested NER through linearization. In *Proc. of ACL*, pages 5326–5331.

Hanna Suominen, Sanna Salanterä, Sumithra Velupillai, Wendy W Chapman, Guergana Savova, Noemie Elhadad, Sameer Pradhan, Brett R South, Danielle L Mowery, Gareth JF Jones, et al. 2013. Overview of the share/clef ehealth evaluation lab 2013. In *Proc. of International Conference of the Cross-Language Evaluation Forum for European Languages*.

Buzhou Tang, Hongxin Cao, Yonghui Wu, Min Jiang, and Hua Xu. 2013. Recognizing clinical entities in hospital discharge summaries using structural support vector machines with word representation features. In *Proc. of BMC medical informatics and decision making*.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proc. of CoNLL*.

Bailin Wang and Wei Lu. 2018. Neural segmental hypergraphs for overlapping mention recognition. In *Proc. of EMNLP*.

Bailin Wang, Wei Lu, Yu Wang, and Hongxia Jin. 2018. A neural transition-based model for nested mention recognition. In *Proc. of EMNLP*.

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *Proc. of EMNLP*.

Jun Xu, Yaoyun Zhang, Jingqi Wang, Yonghui Wu, Min Jiang, Ergin Soysal, and Hua Xu. 2015. Uth-ccb: the participation of the semeval 2015 challenge–task 14. In *Proc. of SemEval*.

Jie Zhang, Dan Shen, Guodong Zhou, Jian Su, and Chew-Lim Tan. 2004. Enhancing hmm-based biomedical named entity recognition by studying special phenomena. *Journal of biomedical informatics*, 37(6):411–422.

Yaoyun Zhang, Jingqi Wang, Buzhou Tang, Yonghui Wu, Min Jiang, Yukun Chen, and Hua Xu. 2014. Uth_ccb: a report for semeval 2014–task 7 analysis of clinical text. In *Proc. of SemEval*.

Guodong Zhou. 2006. Recognizing names in biomedical texts using mutual information independence model and svm plus sigmoid. *International Journal of Medical Informatics*, 75(6):456–467.

Guodong Zhou, Jie Zhang, Jian Su, Dan Shen, and Chewlim Tan. 2004. Recognizing names in biomedical texts: a machine learning approach. *Bioinformatics*, 20(7):1178–1190.

## A Segment Extraction

Neural segmental hypergraphs (Wang and Lu, 2018) were proposed for modeling overlapping structures in entity mentions. We directly adopt their approach to model the segments of overlapping structures. Note that our segment also holds the information of entity type, so the resulting system for segment extraction can also be viewed as performing sub-mention recognition. Next, we illustrate how the segmental hypergraph encodes the overlapping segments by a concrete example.

For brevity, we only show the example that is annotated with one entity type, and it is able to be trivially extended to the case of multiple entity types.

Given a phrase "He had blood in his mouth and on his tongue", there exist two disorder mentions: '*blood in his mouth*' and '*blood ... on his tongue*' where the second mention has a discontiguous sequence of words. Our two-stage approach first extracts segments that lead to these two mentions. In this example, the segments consist of '*blood*', '*blood in his mouth*' and '*on his tongue*'. We observe that the first two segments overlap with each other.

Segmental hypergraph encodes this segment combination based on five types of nodes:

- $\mathbf{A}_i$ encodes all segments that start with the $i$-th or a later word
- $\mathbf{E}_i$ encodes all segments that start exactly with the $i$-th word
- $\mathbf{T}_i^k$ represents all segments of type $k$ starting with the $i$-th word
- $\mathbf{I}_{i,j}^k$ represents all segments of type $k$ that contain the $j$-th word and start with the $i$-th word
- $\mathbf{X}$ marks the end of a segment.

Each segment can be expressed in terms of these five nodes and corresponds with a path in the segmental hypergraph. As a result, each segment combination corresponds with a *hyperpath* where hyperedges are designated to connect multiple nodes so as to model overlapping segments. Figure 2 shows such a hyperpath for the segment combination in our example phrase. Since we only have one entity type in this example, we eliminate the superscript $k$ in $\mathbf{T}$ and $\mathbf{I}$ nodes that indicates the information of entity type.

Starting from the third word 'blood', there exist two segments '*blood*' and '*blood in his mouth*'. The brown hyperedge with the parent node being $\mathbf{I}_{3,3}$ is responsible for connecting these two overlapping segments. This hyperedge means that there exists a segment that ends at the third word (the link from $\mathbf{I}_{3,3}$ to $\mathbf{X}$) and there also exists a segment that continues to the next word (the link from $\mathbf{I}_{3,3}$ to $\mathbf{I}_{3,4}$ ). The segment '*on his tongue*' is directly mapped to the path from $\mathbf{T}_8$ to $\mathbf{X}$.

The score for each hyperpath is the sum of the scores that are computed over each hyperedge. Since $\mathbf{T}$ nodes encode word-level information and $\mathbf{I}$ nodes encode span-level information, two LSTMs are employed to capture the interactions at both word level and span level respectively. We use their original implementation that is publicly available[7].

## B Heuristics for Handling Ambiguity

This section tries to explain the two heuristics "enough" and "all" when ambiguous tag sequences occur. We use the extended BIO tagging scheme (Tang et al., 2013; Muis and Lu, 2016a) for example.

To encode the three discontiguous entities in Figure 3, this tagset has seven tags:

- B/I: **B**eginning and **I**nside of contiguous entities
- BH/IH: **B**eginning and **I**nside of *head* where head refers to segments shared by multiple discontiguous entities.
- BD/ID: **B**eginning and **I**nside of *body* where body refers to segments that are not shared across entities.
- O: **O**utside of entities.

The resulting tag sequence is shown in Figure 4. Since this tagging scheme cannot model the correspondence between different tags, tagging sequences are very likely to have multiple interpretations. For instance, it is not clear that "laceration" should be combined with "esophagus" or with "stomach".

The "all" heuristics extracts all the possible entities that could exist in the tagging sequence. In this case, "all" heuristics will produce "*laceration ... esophagus*" , "*stomach ... lac*", "*blood in stomach*", "*laceration ... lac*", "*esophagu ... lac*", "*laceration ...esophagus ... lac*", "*laceration ... stomach*", "*esophagus ... stomach*", "*laceration... stomach ... lac*", "*esophagus ... stomach...lac*".

The "enough" heuristics tries to find the minimal set of entities that corresponds to this tagging sequence. In this case, "enough" heuristics would produce at least three entities like: "*laceration ... esophagus*" , "*stomach ... lac*" and "*blood in stomach*"; "*laceration ... lac*" , "*blood in stomach*" and

[7] https://github.com/berlino/overlapping-ner-em18

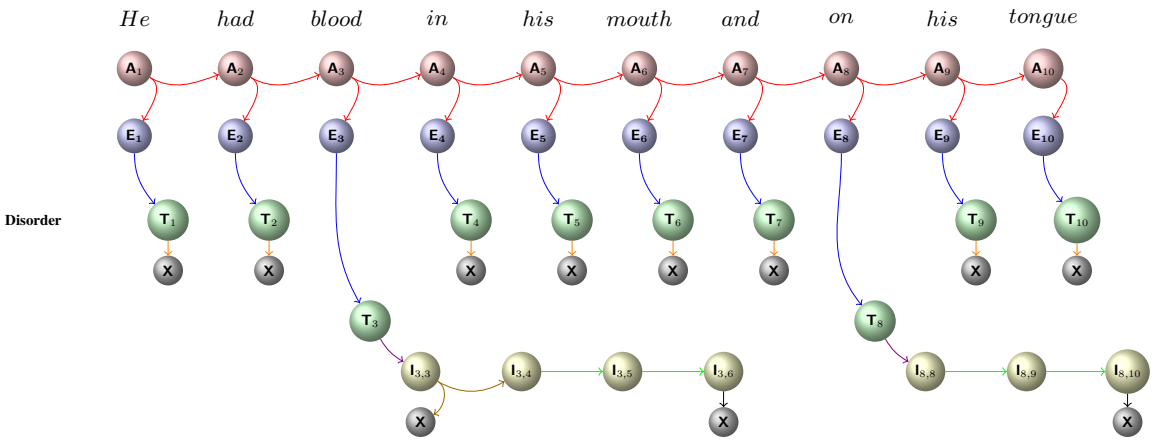


Figure 2: A hyperpath that encodes three mention segments: '*blood*', '*blood in his mouth*' and '*on his tongue*'.




Figure 3: Entities are highlighted with colored underlines. "*laceration ... esophagus*" and "*stomach ... lac*" contain discontiguous sequence of words and the latter also overlaps with another entity "*blood in stomach*".

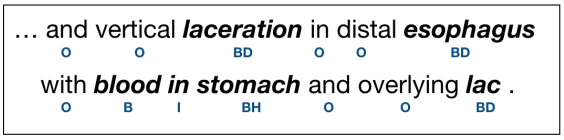


Figure 4: Entities annotated using seven tags.

"*esophagus ... stomach*". We make further constraints to generate only one combination following Muis and Lu (2016b).

## C Hyperparameters

The hyperparameters used in our neural two-stage model are listed in Table 4. Since the size of our dataset is relatively small, the dropout is crucial to prevent overfitting considering that the pre-traind word embeddings have the dimension of 200. The length of most segments is not greater than 6, so we set the maximal length $c$ to be 6 to improve the efficiency of segment extraction.

We also tried to incorporate a character-level component (Lample et al., 2016) to capture morphological and orthographic information. However, it does not have a significant effect on the performance in term of $F_1$.

| | |
|---|---|
| word embedding dim | 200 |
| LSTM(word) hidden size | 128 |
| LSTM(span) hidden size | 128 |
| LSTM(entity) hidden size | 64 |
| maximal length $c$ | 6 |
| dropout | 0.8 |
| $l_2$ | 0.0001 |

Table 4: Hyperparameters of our joint model.

# Cross-Sentence $N$-ary Relation Extraction using Lower-Arity Universal Schemas

**Kosuke Akimoto**[1], **Takuya Hiraoka**[1], **Kunihiko Sadamasa**[1], **Mathias Niepert**[2]
[1] Security Research Laboratories, NEC Corporation
[2] NEC Laboratories Europe
{k-akimoto@ab,t-hiraoka@ce,k-sadamasa@az}.jp.nec.com
mathias.niepert@neclab.eu

## Abstract

Most existing relation extraction approaches exclusively target binary relations, and $n$-ary relation extraction is relatively unexplored. Current state-of-the-art $n$-ary relation extraction method is based on a supervised learning approach and, therefore, may suffer from the lack of sufficient relation labels. In this paper, we propose a novel approach to cross-sentence $n$-ary relation extraction based on *universal schemas*. To alleviate the sparsity problem and to leverage inherent decomposability of $n$-ary relations, we propose to learn relation representations of lower-arity facts that result from decomposing higher-arity facts. The proposed method computes a score of a new $n$-ary fact by aggregating scores of its decomposed lower-arity facts. We conduct experiments with datasets for ternary relation extraction and empirically show that our method improves the $n$-ary relation extraction performance compared to previous methods.

## 1 Introduction

Relation extraction is a core natural language processing task which is concerned with the extraction of relations between entities from text. It has numerous applications ranging from question answering (Xu et al., 2016) to automated knowledge base construction (Dong et al., 2014).

While the vast majority of existing research focuses on extracting *binary* relations, there exists only few recent approaches to extract $n$-ary relations, that is, relations among $n \geq 2$ entities (Li et al., 2015; Ernst et al., 2018). In $n$-ary relation extraction, relation mentions tend to span multiple sentences more frequently as $n$ increases. Thus, Peng et al. (2017) recently extended the problem to cross-sentence $n$-ary relation extraction in which $n$-ary relations are extracted from multiple sentences. As a motivating example, consider the following text from Wikipedia: "*Revis started off the* **2009 season** *matched up against some of football's best wide receivers. In Week 1, he helped limit Houston* **Texans** *Pro-bowler* **Andre Johnson** *to four receptions for 35 yards.*" In this example, two sentences collectively describes that Andre Johnson is a player of the football team the Texans during 2009 season, and thus we need cross-sentence information to correctly extract this ternary interaction among the three entities, i.e. `Player`(Andre Johnson, Texans, 2009 season).

Previous methods (Peng et al., 2017; Song et al., 2018) capture cross-sentence $n$-ary relation mentions by representing texts with a *document graph* which consists of both intra- and cross-sentence links between words. With this graphical representation, they applied graph neural networks to predict ternary relations in the medical domain. However, these methods train the neural networks in a supervised manner using distant supervision (Mintz et al., 2009) and, therefore, may suffer from the lack of sufficient positive labels when a well-populated knowledge base is not available.

On the other hand, for binary relation extraction, the problem of insufficient positive labels can be mitigated with *universal schemas* (Riedel et al., 2013). In a universal schema approach, textual representations (surface patterns) of entities and their relations are encoded into the same vector space as the canonical knowledge base relations. Thus, semantically similar surface patterns can share information of relation labels in a semi-supervised manner. This reduces the amount of required labeled training data. Applying the universal schema approach to $n$-ary ($n > 2$) relation extraction is, however, not straight-forward due to the sparsity of higher-order relation mentions among a specific set of $n > 2$ entities.[1] This is be-

[1] In our Wiki-90k dataset (see §4.1), only 12.5% of ternary entity tuples have at least two relations among the entities, while 77.3% of entities appear at least twice.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6225–6231, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

cause the universal schema approach (Riedel et al., 2013) and its extensions (Toutanova et al., 2015; Verga et al., 2016, 2017) utilize co-occurring patterns of relation types between specific pair of entities. Also, prior work has only addressed binary relations, and it is not trivial to define surface patterns among $n > 2$ entities and to encode these patterns into a vector representation.

To mitigate the aforementioned sparsity problem and utilize existing encoders for binary and unary surface patterns, we propose to train universal schema models on more dense lower-arity (unary and binary) facts instead of original sparse $n$-ary facts. Since most $n$-ary relations can be decomposed into a set of $k$-ary relations ($k = 1, 2$) which are implied by the $n$-ary relation,[2] we can easily acquire lower-arity facts by decomposing $n$-ary facts. Our model learns representations of these lower-arity relations using the universal schema framework, and predicts new $n$-ary facts by aggregating scores of lower-arity facts.

To evaluate the proposed method, we create new cross-sentence $n$-ary relation extraction datasets with multiple ternary relations.[3] The new datasets contain more entity tuples with known relational facts appeared in a knowledge base than the existing dataset (Peng et al., 2017), and, therefore, these datasets can be used to more effectively evaluate methods which predict relation labels for each individual entity tuple. We show empirically that by jointly training lower-arity models and an $n$-ary score aggregation model, the proposed method improves the performance of $n$-ary relation extraction. To the best of our knowledge, this is the first attempt to apply universal schemas to $n$-ary relation extraction, taking advantage of the compositionality of higher-arity facts.

## 2 Task Definition and Notation

The cross-sentence $n$-ary relation extraction task (Peng et al., 2017) is defined as follows. Let $\mathcal{E}$ be a set of entities, $\mathcal{R}_{\text{KB}}$ be a set of relation types of an external knowledge base $\mathcal{KB}$, and $\mathcal{O}_{\text{KB}} = \{\langle r, (e_1, ..., e_n)\rangle : r(e_1, ..., e_n) \in \mathcal{KB}, r \in \mathcal{R}_{\text{KB}}, e_i \in \mathcal{E}\}$ be the set of facts in $\mathcal{KB}$. We collect a set of candidate entity tuples among which KB relation $r \in \mathcal{R}_{\text{KB}}$ possibly holds.[4] Here, all entities in each candidate tuple $(e_1, ..., e_n)$ are mentioned in the same text section $T$ in a given set of documents. We define a set of these entity mentions as $\mathcal{O}_{\text{text}} = \{\langle T, (e_1, ..., e_n)\rangle : e_i \in \mathcal{E} \text{ is mentioned in } T\}$. Here, text section $T$ is a (short) span in a document which can describes relational facts among entities. In the cross-sentence $n$-ary relation extraction task, text section $T$ can contain multiple sentences. In this paper, following (Peng et al., 2017), we define $M$ consecutive sentences ($M \geq 1$) which contain $n$ target entities as a text section in the cross-sentence $n$-ary relation extraction task. We use the term "relation" to refer to both relations $r \in \mathcal{R}_{\text{KB}}$ and sections $T$.

The goal of the cross-sentence $n$-ary relation extraction task is to predict new facts $\langle r, (e_1, ..., e_n)\rangle \notin \mathcal{O}_{\text{KB}}$ for relation $r \in \mathcal{R}_{\text{KB}}$ given $\mathcal{O} = \mathcal{O}_{\text{KB}} \cup \mathcal{O}_{\text{text}}$, where $n \geq 2$.

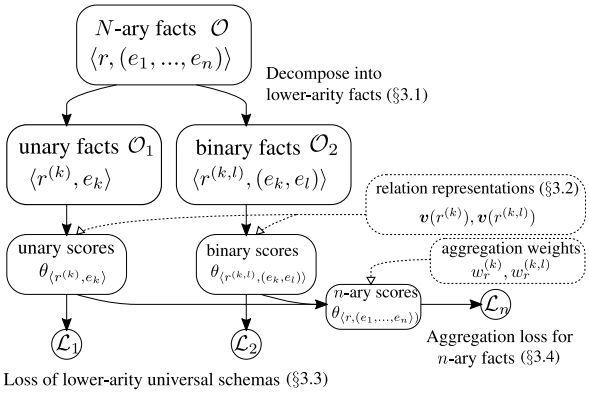


Figure 1: An overview of the proposed method.

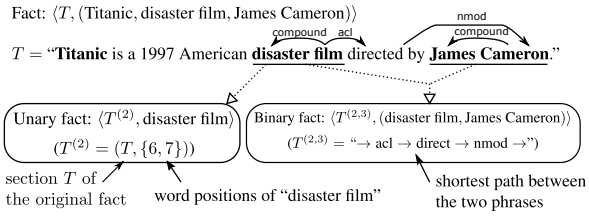


Figure 2: An example of decomposed textual facts.

## 3 Proposed Method

### 3.1 Lower-Arity Facts

To alleviate the sparsity problem of facts among $n$ entities ($n > 2$) and to utilize well-studied encoders for binary and unary surface patterns, we

[2] For example, the ternary relation `AwardedFor`$(director, movie, award)$ can be decomposed into the binary relations `DirectorOf`$(director, movie)$ and `WonAward`$(director, award)$. Note that a similar idea is introduced in (Ernst et al., 2018) as *partial facts* or *partial patterns*.

[3] Our codes and datasets are available at https://github.com/aurtg/nary-relation-extraction-decomposed.

[4] It is allowed that multiple KB relations hold among the same set of entities.

decompose a set of original $n$-ary facts, $\mathcal{O}$, into a set of unary facts $\mathcal{O}_1$ and a set of binary facts $\mathcal{O}_2$ (Figure 1).

**Unary Facts:** Given an $n$-ary fact $\langle r, (e_1, ..., e_k)\rangle \in \mathcal{O}$, we decompose it into a set of $n$ unary facts $\{\langle r^{(k)}, e_k\rangle : k = 1, ..., n\}$, where $r^{(k)}$ is a tentative unary relation w.r.t. the $k$-th argument of the original relation $r$. If $r$ is a KB relation, we define unary relation $r^{(k)}$ as a new canonicalized relation. If $r$ is section $T$, we define unary relation $r^{(k)}$ as a tuple $r^{(k)} = (T, pos(e_k))$, where $pos(e_k)$ is a set of word position indices of entity $e_k$ in section $T$ (Figure 2). We denote a set of all decomposed unary facts by $\mathcal{O}_1$. Intuitively, these unary relations represent semantic roles or types of corresponding arguments of the original relation $r$ (Yao et al., 2013).

**Binary Facts:** Given an $n$-ary fact $\langle r, (e_1, ..., e_k)\rangle \in \mathcal{O}$, we decompose it into a set of $n(n-1)$ binary facts $\{\langle r^{(k,l)}, (e_k, e_l)\rangle : k, l = 1, ..., n, k \neq l\}$, where $r^{(k,l)}$ is a tentative binary relation between the $k$-th and $l$-th argument of the original relation $r$. If $r$ is a KB relation, we define binary relation $r^{(k,l)}$ as a new canonicalized relation. If $r$ is a section $T$, we represent it by the shortest path between $e_k$ and $e_l$ on the *document graph* (Quirk and Poon, 2017) of $T$ (Figure 2), and denote it by $path(T; e_k, e_l)$. We denote the set of all decomposed binary facts by $\mathcal{O}_2$.

### 3.2 Lower-Arity Relation Representations

We learn a vector representation $\boldsymbol{v}(r) \in \mathbb{R}^{d_r}$ for each unary or binary relation in $\mathcal{O}_1$ or $\mathcal{O}_2$. For $r^{(k)}$ or $r^{(k,l)}$ derived from a KB relation, we represent it by a trainable parameter vector. On the other hand, for the one derived from a textual relation, we use the following encoders to compute its representations.

**Unary encoder:** For an unary textual relation $r^{(k)} = (T, pos(e_k))$, we represent each section $T$ by a sequence of word vectors and use a bidirectional LSTM (Bi-LSTM) (Schuster and Paliwal, 1997) to compute a hidden representation $\boldsymbol{h}_l \in \mathbb{R}^{d_r}$ at each word position $l$. Following recent works (Zhang et al., 2018; He et al., 2018; Lee et al., 2017), we aggregate $\boldsymbol{h}_l$ within a phrase of entity $e_k$ to compute $\boldsymbol{v}(T^{(k)})$. We use element-wise mean as aggregation function:

$$\boldsymbol{v}(r^{(k)}) = \text{mean}(\{\boldsymbol{h}_l : l \in pos(e_k)\}). \quad (1)$$

**Binary encoder:** For a binary textual relation $r^{(k,l)} = path(T; e_k, e_l)$, we represent each token (word or edge label) in $path(T; e_k, e_l)$ by an embedding vector (Toutanova et al., 2015; Verga et al., 2016). We use a Bi-LSTM to compute a hidden representation $\boldsymbol{h}'_l \in \mathbb{R}^{d_r}$ at each token position $l$, and max-pool along the path to compute the relation representation:

$$\boldsymbol{v}(T^{(k,l)}) = \max(\{\boldsymbol{h}'_l : l = 1, ..., L\}). \quad (2)$$

### 3.3 Learning Relation Representations

We follow Verga et al. (2017) to train relation representations (§3.2). We define a score $\theta_{\langle r,p\rangle}$ for each lower-arity fact $\langle r, p\rangle \in \mathcal{O}_1 \cup \mathcal{O}_2$, and minimize the following loss (3) for each arity $i = 1, 2$. Here, placeholder $p$ refers to either an entity (if $\langle r, p\rangle \in \mathcal{O}_1$) or an entity tuple (if $\langle r, p\rangle \in \mathcal{O}_2$), and we simply refer to both as entity tuple. The loss functions contrast a score of an original fact $\langle r, p^+\rangle \in \mathcal{O}_i$ and those of $K$ sampled negative facts $\langle r, p_k^-\rangle \notin \mathcal{O}_i$. We sample negative facts by randomly replacing entity tuple $p^+$ in the original fact by different entity tuples $p_k^-$.

$$\mathcal{L}_i = \mathop{\mathbb{E}}_{\substack{\langle r,p^+\rangle \in \mathcal{O}_i \\ \langle r,p_k^-\rangle \notin \mathcal{O}_i}} [-\log(\frac{\exp(\theta_{\langle r,p^+\rangle})}{\exp(\theta_{\langle r,p^+\rangle}) + \sum_k \exp(\theta_{\langle r,p_k^-\rangle})})]. \quad (3)$$

The score of fact $\langle r, p\rangle$ is defined as $\theta_{\langle r,p\rangle} = \boldsymbol{v}(r)^T \boldsymbol{v}(p; r)$. Entity tuple representations $\boldsymbol{v}(p; r)$ are computed with a weighted average of the representations $\{\boldsymbol{v}(r') : r' \in V(p)\}$ as shown in (4) and (5) where $a(r', r; V(p))$ is the attention weight for each relation $r' \in V(p)$.[5]

$$\boldsymbol{v}(p; r) = \sum_{r' \in V(p)} a(r', r; V(p)) \boldsymbol{v}(r'),$$
$$a(r', r; V(p)) = \frac{\exp(\boldsymbol{v}(r')^T \boldsymbol{v}(r))}{\sum_{r'' \in V(p)} \exp(\boldsymbol{v}(r'')^T \boldsymbol{v}(r))}. \quad (4)$$

$$V(p) = \begin{cases} \{r : \langle r, e_k\rangle \in \mathcal{O}_1\} & (\text{if } p = e_k) \\ \{r : \langle r, (e_k, e_l)\rangle \in \mathcal{O}_2\} & (\text{if } p = (e_k, e_l)) \end{cases}. \quad (5)$$

### 3.4 Aggregating Lower-Arity Scores

To predict $n$-ary facts of KB relation $r \in \mathcal{R}_{\text{KB}}$, we compute its score $\theta_{\langle r,(e_1,...,e_n)\rangle}$ by aggregating lower-arity scores as in (6), where $w_r^{(\cdot)}$ is a positive scalar weight defined for each KB relation which sum to one: $\sum_k w_r^{(k)} + \sum_{k \neq l} w_r^{(k,l)} = 1$.

[5] During training, as in (Verga et al., 2017), we aggregate sub-sampled $M$ relations from $V(p)$ if $|V(p)| > M$. We set $M = 2$ for all experiments.

We can set all weights $w_r^{(k)}$ and $w_r^{(k,l)}$ to $1/n^2$, or train these weights to give higher scores to positive $n$-ary facts by minimizing additional loss function $\mathcal{L}_n$. Note that $\mathcal{L}_n$ directly contrasts $n$-ary scores associated with KB relations $r \in \mathcal{R}_{\text{KB}}$ in a more supervised manner than both $\mathcal{L}_1$ and $\mathcal{L}_2$.[6]

$$\theta_{\langle r,(e_1,\ldots,e_n)\rangle} = \sum_{k=1}^{n} w_r^{(k)} \theta_{\langle r^{(k)},e_k\rangle} + \sum_{\substack{k=1,\ldots,n \\ l=1,\ldots,n \\ k\neq l}} w_r^{(k,l)} \theta_{\langle r^{(k,l)},(e_k,e_l)\rangle}. \quad (6)$$

$$\mathcal{L}_n = \mathop{\mathbb{E}}_{\substack{\langle r,p^+\rangle\in\mathcal{O}_{\text{KB}} \\ \langle r,p^-\rangle\notin\mathcal{O}_{\text{KB}}}} [\max(0, 1 - \theta_{\langle r,p^+\rangle} + \theta_{\langle r,p^-\rangle})] \quad (7)$$

The overall loss function is now $\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2 + \alpha\mathcal{L}_n$. By changing $\alpha$, we can balance the semi-supervised effect of lower-arity universal schemas ($\mathcal{L}_1, \mathcal{L}_2$) and that of the supervision with $n$-ary relation labels ($\mathcal{L}_n$).

# 4 Experiments

## 4.1 Dataset

The cross-sentence $n$-ary relation extraction dataset from Peng et al. (2017) contains only 59 distinct ternary KB facts including the train and test set. Since our proposed method and universal schemas baselines predict KB relations for each *entity tuple* instead of each *surface pattern*, the number of known facts of KB relations is crucial to reliably evaluate and compare these methods. Thus, we created two new $n$-ary cross-sentence relation extraction datasets (dubbed with **Wiki-90k** and **WF-20k**) that contain more known facts retrieved from public knowledge bases.

To create the Wiki-90k and WF-20k datasets, we used Wikidata and Freebase respectively as external knowledge bases. Since these knowledge bases store only binary relational facts, we defined multiple ternary relations by combining a few binary relations.[7,8] For both datasets, we collected paragraphs from the English Wikipedia, and used Stanford CoreNLP (Manning et al., 2014) to extract dependency and co-reference links. Entity mentions are detected using DBpedia Spotlight (Daiber et al., 2013). We followed (Peng et al., 2017) to extract co-occurring entity tuples and their surface patterns, that is, we selected tuples which occurred in a *minimal span* within at most $M \leq 3$ consecutive sentences. Entity tuples without a known KB relation are subsampled, since the number of such tuples are too large. We randomly partitioned all entity tuples into train, development (dev), and test sets.

## 4.2 Baselines

**(Song et al., 2018):** The state-of-the-art cross-sentence $n$-ary relation extraction method proposed by Song et al. (2018) represents each surface pattern by the concatenation of entity vectors from the last layer of a Graph State LSTM, a variant of a graph neural network. The concatenated vector is then fed into a classifier to predict the relation label. Since their method directly predicts a relation label for each surface pattern, it is more robust to the sparsity of surface patterns among a specific higher arity entity tuple. However, due to their purely supervised training objective, its performance may degrade if the number of available training labels is small.

**Universal schemas:** We compared our method with semi-supervised methods based on universal schemas (Toutanova et al., 2015; Verga et al., 2017). In our experiments, we used the same encoder as (Song et al., 2018) to encode each surface pattern.[9] We tested two types of scoring functions, Model F and Model E, as in (Toutanova et al., 2015).[10,11]

## 4.3 Evaluation

We compared the methods in the held-out evaluation as in (Mintz et al., 2009) and report (weighted) mean average precision (MAP) (Riedel et al., 2013). Unless otherwise noted, reported values are average values over six experiments, in which network parameters are randomly initialized. All reported $p$-values are calculated based on Wilcoxon rank sum test (Wilcoxon, 1945) with

[6] The loss function (7) performs better than a log-likelihood based loss, $-\log\sigma(\theta_{\langle r,p^+\rangle} - \theta_{\langle r,p^-\rangle})$.

[7] See the appendix A for details about defined ternary relations.

[8] For about half of the defined ternary relations, combined original binary relations in KB are different from decomposed binary relations of the ternary relations in the proposed method (§3.1).

[9] Using linear projection instead of simple concatenation did not improve performance in our preliminary experiments.

[10] It is not trivial to apply Model E scoring function with Verga et al. (2017) method, since their aggregation method calculates a representation for each row, i.e. entity tuple.

[11] DistMult scoring function in (Toutanova et al., 2015) showed poor performance compared to the other two scoring functions in our preliminary experiments.

| Method | Wiki-90k | | WF-20k | |
|---|---|---|---|---|
| | average | weighted | average | weighted |
| Proposed | **0.584** | **0.634** | **0.821** | **0.842** |
| (Song et al., 2018) | 0.471 | 0.536 | 0.639 | 0.680 |
| (Toutanova et al., 2015) with Graph State LSTM (Song et al., 2018) encoder | | | | |
| Model F | 0.240 | 0.262 | 0.341 | 0.380 |
| Model E | 0.399 | 0.414 | 0.725 | 0.752 |
| (Verga et al., 2017) with Graph State LSTM (Song et al., 2018) encoder | | | | |
| Model F | 0.443 | 0.482 | 0.610 | 0.653 |

**bold**: $p \leq 0.01$

Table 1: Mean average precisions (MAPs) on **test** data.

multiple-test adjustment using Holm's method (Holm, 1979).

### 4.4 Results

Table 1 illustrates the performance of each method.[12] Compared to the baseline methods, our proposed method achieves higher weighted MAP for both datasets. Interestingly, Model F performs well in Verga et al. (2017) baseline, while it shows low performance in Toutanova et al. (2015) baseline.

**Ablation Study:** Table 2 illustrates the performance of various settings of our proposed method. **U**,**B**, and **N** stand for using the loss functions $\mathcal{L}_1$, $\mathcal{L}_2$, and $\alpha\mathcal{L}_n$ respectively. In the result, **U+B** performs significantly better ($p < 0.005$) than **U** and **B**, and this shows effectiveness of combining scores of both binary facts and unary facts. On the other hand, there was no significant difference between **U+B+N** and **N** ($p > 0.9$). Note that we used all positive labels in this experiment, that is, sufficient amount of positive labels are used for calculating the loss **N**.

**Data efficiency:** Furthermore, we also investigated the influence of the training data size (the number of positive labels) of our proposed method and baseline methods.[13] Here, $\alpha = \infty$ stands for optimizing $\mathcal{L}_n$ instead of $\mathcal{L}_1 + \mathcal{L}_2 + \alpha\mathcal{L}_n$. As shown in Figure 3, $\alpha = 1$ achieved higher performance than $\alpha = \infty$, showing that introducing lower-arity semi-supervised loss ($\mathcal{L}_1 + \mathcal{L}_2$) improves the performance for dataset with few positive labels. On the other hand, the lower performance of $\alpha = 0$ compared to $\alpha = 0.1, 1$ suggests that information of higher-arity facts introduced from $\mathcal{L}_n$ is benefitial for $n$-ary relation extraction.

## 5 Conclusion and Future Works

We proposed a new method for cross-sentence $n$-ary relation extraction that decomposes sparse $n$-ary facts into dense unary and binary facts. Experiments on two datasets with multiple ternary relations show that our proposed method can statistically significantly improve over previous works, which suggests the effectiveness of using unary and binary interaction among entities in surface patterns.

However, as Fatemi et al. (2019) suggests, there exists cases in which reconstructing $n$-ary facts from decomposed binary facts induces false positives. Tackling this issue is one important future research direction.

[12] For the proposed method, we set $\alpha = 10$.

[13] In this experiment, we conducted four experiments per each setting and set $K = 10$.

| Setting | average | weighted |
|---|---|---|
| U | 0.363 | 0.404 |
| B | 0.411 | 0.452 |
| N | 0.646 | 0.685 |
| U+B | 0.521 | 0.552 |
| U+B+N | 0.646 | 0.682 |
| U+B+N ($K = 10$) | **0.650** | **0.689** |
| U+B+N ($K = 5$) | 0.592 | 0.640 |
| U+B+N (fix $w_r = 1$) | 0.645 | 0.679 |

Default hyperparameter: $K = 20$

Table 2: Ablation study: mean average precisions (MAPs) on **dev** data (Wiki-90k).

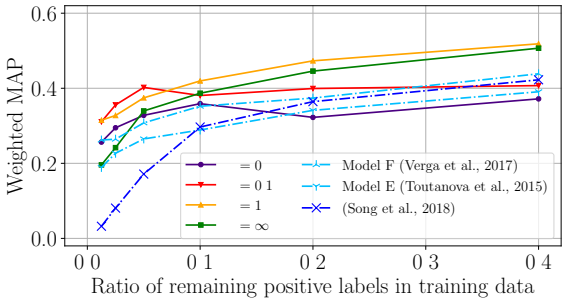


Figure 3: Weighted MAP on test data with missing labels (Wiki-90k).

## Acknowledgements

We thank all the EMNLP reviewers and Daniel Andrade for their valuable comments and suggestions to improve the paper.

## References

Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N. Mendes. 2013. Improving efficiency and accuracy in multilingual entity extraction. In *Proceedings of the 9th International Conference on Semantic Systems*, I-SEMANTICS '13, pages 121–124. ACM.

Xin Luna Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *The 20th ACM*

*SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pages 601–610.

Patrick Ernst, Amy Siu, and Gerhard Weikum. 2018. Highlife: Higher-arity fact harvesting. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, pages 1013–1022. International World Wide Web Conferences Steering Committee.

Bahare Fatemi, Perouz Taslakian, David Vazquez, and David Poole. 2019. Knowledge hypergraphs: Extending knowledge graphs beyond binary relations. *arXiv preprint arXiv:1906.00137*.

Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2018. Jointly predicting predicates and arguments in neural semantic role labeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 364–369. Association for Computational Linguistics.

Sture Holm. 1979. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, pages 65–70.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197. Association for Computational Linguistics.

Hong Li, Sebastian Krause, Feiyu Xu, Andrea Moro, Hans Uszkoreit, and Roberto Navigli. 2015. Improvement of n-ary relation extraction by adding lexical semantics to distant-supervision rule learning. In *Proceedings of the International Conference on Agents and Artificial Intelligence - Volume 2*, ICAART 2015, pages 317–324. SCITEPRESS - Science and Technology Publications, Lda.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60. Association for Computational Linguistics.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, pages 1003–1011. Association for Computational Linguistics.

Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen-tau Yih. 2017. Cross-sentence n-ary relation extraction with graph lstms. *Transactions of the Association for Computational Linguistics*, 5:101–115.

Chris Quirk and Hoifung Poon. 2017. Distant supervision for relation extraction beyond the sentence boundary. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1171–1182. Association for Computational Linguistics.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84. Association for Computational Linguistics.

M. Schuster and K.K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. N-ary relation extraction using graph-state lstm. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2226–2235. Association for Computational Linguistics.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509. Association for Computational Linguistics.

Patrick Verga, David Belanger, Emma Strubell, Benjamin Roth, and Andrew McCallum. 2016. Multilingual relation extraction using compositional universal schema. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 886–896. Association for Computational Linguistics.

Patrick Verga, Arvind Neelakantan, and Andrew McCallum. 2017. Generalizing to unseen entities and entity pairs with row-less universal schema. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*.

Frank Wilcoxon. 1945. Individual comparisons by ranking methods. *Biometrics bulletin*, 1(6):80–83.

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Question answering on freebase via relation extraction and textual evidence. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2326–2336. Association for Computational Linguistics.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2013. Universal schema for entity type prediction. In *Proceedings of the 2013 Workshop on Automated*

*Knowledge Base Construction*, AKBC '13, pages 79–84, New York, NY, USA. ACM.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2205–2215. Association for Computational Linguistics.

# Gazetteer-Enhanced Attentive Neural Networks for Named Entity Recognition

**Hongyu Lin**[1,3], **Yaojie Lu**[1,3], **Xianpei Han**[1,2,*], **Le Sun**[1,2], **Bin Dong**[4], **Shanshan Jiang**[4]
[1]Chinese Information Processing Laboratory [2]State Key Laboratory of Computer Science
Institute of Software, Chinese Academy of Sciences, Beijing, China
[3]University of Chinese Academy of Sciences, Beijing, China
[4]Ricoh Software Research Center (Beijing) Co.,Ltd.
{hongyu2016,yaojie2017,xianpei,sunle}@iscas.ac.cn
{bin.dong,shanshan.jiang}@srcb.ricoh.com

## Abstract

Current region-based NER models only rely on fully-annotated training data to learn effective region encoder, which often face the training data bottleneck. To alleviate this problem, this paper proposes *Gazetteer-Enhanced Attentive Neural Networks*, which can enhance region-based NER by learning name knowledge of entity mentions from easily-obtainable gazetteers, rather than only from fully-annotated data. Specially, we first propose an *attentive neural network (ANN)*, which explicitly models the mention-context association and therefore is convenient for integrating externally-learned knowledge. Then we design an auxiliary *gazetteer network*, which can effectively encode name regularity of mentions only using gazetteers. Finally, the learned gazetteer network is incorporated into ANN for better NER. Experiments show that our ANN can achieve the state-of-the-art performance on ACE2005 named entity recognition benchmark. Besides, incorporating gazetteer network can further improve the performance and significantly reduce the requirement of training data.

## 1 Introduction

Named entity recognition (NER), aiming to identify text mentions of specific entity types, is a fundamental NLP task. Recently, region-based NER approaches (Finkel and Manning, 2009; Xu et al., 2017; Sohrab and Miwa, 2018) have attracted significant attention, which first encode all candidate regions (commonly all subsequences of a sentence) using a *region encoder*, then identify whether each subsequence is an entity mention of target types using a classifier. For example, in Figure 1 all subsequences of the sentence, such as "George Washington", will first be encoded, and then be classified into entity types. Compared to sequential labeling models, region-based models can naturally detect nested or overlapping mentions by considering all subsequences, and therefore are of great value to NER.

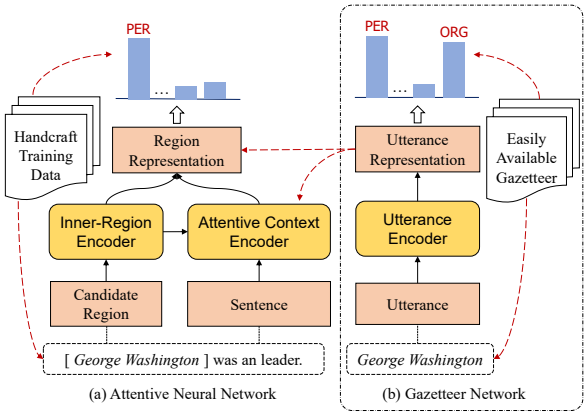


Figure 1: The overall architecture of GEANN. The candidate region is "George Washington", which literally could be a person or an organization (university).

Generally, an effective region encoder should capture two kinds of knowledge for NER. One is *name knowledge*, which encodes the inner compositional regularity of entity mentions, i.e., how likely a subsequence itself will be an entity mention. For example, a region encoder should know "George Washington" is a valid *PER* name because "George" is a common first name and "Washington" is a common last name. Another is *context knowledge*, which identifies whether the subsequence in the context indeed refers to an entity. For example, a region encoder should know "X said" is a suitable context for *PER* mention and "study at X" is a suitable context for *Org* mention.

Currently, most region-based NER models learn these two kinds of knowledge only from expensive, fully-annotated training data, and therefore often face the training data bottleneck, i.e., the lack of training data will significantly undermine

*Xianpei Han is the corresponding author.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6232–6237,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

the performance. To address this problem, we find that name knowledge can be effectively captured by leveraging easily-obtainable gazetteer resources. For example, it is easy to learn the company name patterns "the...company" and "...Inc." from a company name gazetteer containing "the Walt Disney company" and "Apple Inc.". By capturing mention regularity entailing gazetteers, the region-based models can be enhanced with more accurate name knowledge, and thereby the need of fully-annotated training data can be reduced.

To this end, this paper proposes ***G**azetteer-**E**nhanced **A**ttentive **N**eural **N**etworks (GEANN)*, whose architecture is shown in Figure 1. Specifically, to better decouple name and context knowledge for incorporating gazetteer information, we first design a new region-based ***attentive neural network (ANN)***, which introduces attention mechanism (Bahdanau et al., 2014; Vaswani et al., 2017; Su et al., 2018) to explicitly model the association between mentions and contexts. Starting from ANN, we further introduce an auxiliary ***gazetteer network***, which can effectively learn name knowledge only using gazetteers, i.e., it can encode each utterance in a context-free way, and identifies whether it matches regular patterns of mentions. Finally, the learned gazetteer network is incorporated into ANN to capture better name and context knowledge. Experiments show that ANN can achieve the state-of-the-art NER performance, and incorporating name knowledge from gazetteers can significantly reduce the training data requirement. To the best of our knowledge, this is the first work trying to explicitly exploit mention-context association with attention mechanism in region-based NER, as well as the first work which enhances NER model with name knowledge captured from gazetteers using neural networks.

## 2 Attentive Neural Network for NER

This section describes our attentive neural network, which directly classifies over all subsequences of a sentence to recognize whether each subsequence corresponds to an entity mention. Figure 1 (a) shows the architecture of ANN.

Given a sentence, ANN first maps all words into word representations $\{\boldsymbol{x_1}, \boldsymbol{x_2}, ..., \boldsymbol{x_n}\}$ following Lample et al. (2016). Then a BiLSTM layer is used to obtain context-aware word representation $\boldsymbol{h_t^A}$. After that, for each candidate region $s_{ij}$, we follow Sohrab and Miwa (2018) to use an inner-region encoder to obtain its representation $\boldsymbol{s_{ij}}$ , which captures name knowledge considering both its boundary and inside information:

$$\boldsymbol{s}_{ij} = \text{MLP}([\boldsymbol{h}_i^A; \frac{1}{j-i+1}\sum_{k=i}^{j}\boldsymbol{h}_k^A; \boldsymbol{h}_j^A]), \quad (1)$$

where MLP is a multi-layer perceptron. To explicitly model the association between a region and its context, we design an attentive context encoder, which outputs a contextual vector $\boldsymbol{c}_{ij}$ entailing the context knowledge of $s_{ij}$ by:

$$\begin{aligned} \boldsymbol{c}_{ij} &= \sum_{k=1}^{i-1}\alpha_{ijk}\boldsymbol{h}_k^A + \sum_{k=j+1}^{n}\alpha_{ijk}\boldsymbol{h}_k^A \\ \alpha_{ijk} &= \frac{\exp(e_{ijk})}{\sum_{t=1}^{i-1}\exp(e_{ijt}) + \sum_{t=j+1}^{n}\exp(e_{ijt})} \end{aligned} \quad (2)$$

where

$$e_{ijk} = \tanh(\boldsymbol{s_{ij}}^T\boldsymbol{\Lambda}\boldsymbol{h}_k^A + \boldsymbol{W}\boldsymbol{h}_k^A + b) \quad (3)$$

is an attentive model which scores how important the word $x_k$ is for recognizing the entity type of $s_{ij}$. After obtaining $\boldsymbol{c}_{ij}$, we concatenate it with $\boldsymbol{s}_{ij}$, and feed it into a MLP classifier to obtain the probabilities of $s_{ij}$ corresponding to an entity type (or NIL if $s_{ij}$ is not a mention). Similar to previous methods, ANN is learned by minimizing the cross-entropy loss on the fully-annotated training data.

By decoupling and explicitly modeling the mention-context association, ANN not only can better identify entity mentions, but also is very easy to incorporate external name knowledge. This enables convenient integration of gazetteer knowledge, as we will illustrate next.

## 3 Gazetteer-Enhanced ANN

One main drawback of ANN and other region-based models is that they only rely on fully-annotated data to learn both name knowledge and context knowledge. Unfortunately, such training data is very expensive to construct, which limits the applications of these model to more entity types. To tackle this problem, we proposes to learn name knowledge from easily-obtainable, large-scale gazetteers. In this way, the name knowledge can be captured more accurately, and the requirement of fully-annotated data can be reduced.

To this end, we propose to incorporate an auxiliary *gazetteer network* into ANN, which can learn and exploit name knowledge only using gazetteers. Given an utterance, gazetteer network predicts

whether it should be included in the gazetteer of specific entity types, i.e., whether it is a valid entity name. Gazetteer network is context-free, which only considers whether the input follows the compositional regularity of mentions, and therefore can be trained only using gazetteers.

Formally, given an input utterance $u = \{u_i, ..., u_j\}$, an utterance encoder first learns its representation $\boldsymbol{u}$, which has similar structure with the inner-region encoder of ANN. After that, $\boldsymbol{u}$ is used to compute the probabilities of it being a valid name for each type:

$$\boldsymbol{O}_u^G = s(\boldsymbol{W}\boldsymbol{u} + \boldsymbol{b}). \quad (4)$$

where $s$ is the sigmoid function and $k^{th}$ dimension of $\boldsymbol{O}_u^G$ indicates the probability of $u$ being a valid mention of type $y_k$. As an utterance can possibly be a valid mention of various types, we use a multi-label, multi-class cross-entropy loss to train our gazetteer network:

$$\mathcal{L}^G(\theta) = \sum_{u \in G} [\boldsymbol{g}'_u \log \boldsymbol{O}_u^G + (1 - \boldsymbol{g}'_u) \log(1 - \boldsymbol{O}_u^G)] \quad (5)$$

where $G$ is the gazetteers, $\boldsymbol{g}'_u$ is an one-hot vector whose $k^{th}$ will be set to 1 if utterance $u$ is in the gazetteer of type $y_k$, elsewise 0.

In this way, a well-trained gazetteer network can learn an effective utterance representation $\boldsymbol{u}$, which is used to identify whether the utterance is a valid mention. This means $\boldsymbol{u}$ should capture enough name knowledge of specific entity types. To incorporate such knowledge into ANN, we simply concatenate the representation learned by gazetteer network and the representation learned by the original inner-region encoder. Then this new representation is fed to the following modules of ANN. In this way, name knowledge learned from gazetteers is incorporated to enhance the region encoder, and the requirement of fully-annotated data can be reduced.

# 4 Experiments

## 4.1 Experimental Settings

**Data Preparation.** We conducted experiments on ACE2005 named entity recognition task[1]. We used the same dataset splits as Wang and Lu (2018); Katiyar and Cardie (2018). For each entity type in ACE2005, we collect a gazetteer from Wikipedia anchor texts, i.e., anchor texts linking to an entity whose type are the same will be included in a gazetteer. The same as previous studies, models are evaluated using micro-F1. To balance time complexity and recall, we follow Wang and Lu (2018) to restrict mention length up to 6, which covers more than 93% mentions.

[1]Conventional NER datasets, such as CoNLL2003, removed nested entity mentions and therefore are not suitable for evaluating region-based NER models.

| | P | R | F1 |
|---|---|---|---|
| LSTM-CRF | 73.2 | 61.3 | 66.7 |
| Neural Transition | 75.2 | 65.5 | 70.0 |
| Segmental Hypergraph | 75.7 | 68.3 | 71.8 |
| Exhaustive | **81.2** | 66.9 | 73.4 |
| ANN | 78.9 | 69.8 | 74.1 |
| GEANN | 77.1 | **73.3** | **75.2** |

Table 1: Experiment results on ACE2005 named entity mention recognition.

**Baselines.** Following methods were compared:

1) **LSTM-CRF** (Lample et al., 2016), which is the most widely used NER baseline, but it cannot handle nested or overlapping mentions.

2) **Neural Transition** (Wang et al., 2018), a transition model which achieved very competitive performance on ACE2005.

3) **Segmental Hypergraph** (Wang and Lu, 2018), a hypergraph-based model which introduces a new tagging schema and achieved the state-of-the-art performance on ACE2005.

4) **Exhaustive Model** (Sohrab and Miwa, 2018), a region-based model using a region encoder to capture both inner and boundary features of a candidate region, which is similar to ANN without attentive contextual encoder.

## 4.2 Overall Results

Table 1 shows the overall results of our methods compared with baselines. We can see that:

1) By explicitly modeling both name and context knowledge, the proposed attentive neural network is an effective region-based NER model, and achieves the state-of-the-art performance. Compared with other baselines, ANN achieves significant F1-score improvements.

2) Incorporating name knowledge from gazetteers can significantly improve the performance. GEANN further achieves 1.1 F1-score improvement over ANN. This indicates that name knowledge learned from gazetteers can significant enhance the region encoder, and therefore improves the NER performance.

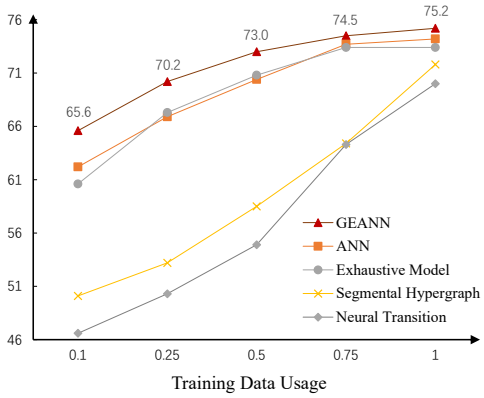


Figure 2: F1-scores on ACE2005 when training data size varies.

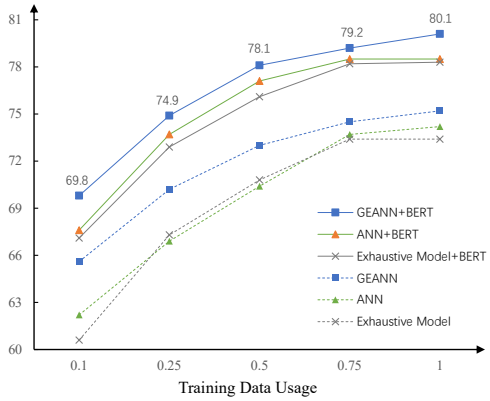


Figure 3: F1-scores of models with BERT on ACE2005 when training data size varies.

3) Our attentive context encoder provides an effective way to exploit context knowledge for NER. Compared with *Exhaustive* baseline, ANN achieves significant improvement by explicitly modeling the association between entity mentions and their contexts.

### 4.3 Effects of Gazetteer Network

To further investigate the effect of introducing gazetteers, Table 2 shows the results when training data size varies. We can see that:

1) For *Transition* and *SH*, their performance significantly decreased when reducing training data. We believe this because these approaches need to model complex label structure, so large scale training data are critical, and reducing training data will have huge impact on them.

2) Region-based models are less sensitive to the reduction of training data. We believe this is because their output structure is simple, and therefore can be trained using less training data.

3) GEANN can achieve significant improvements over ANN regardless of training data size. By leveraging name knowledge from gazetteers, GEANN with only 50% training data can achieve comparable performance with ANN trained with the entire dataset.

### 4.4 GEANN with BERT

Pretrained context-aware representation, such as ELMOs (Peters et al., 2018) and BERT (Devlin et al., 2018), have shown significant progress in many NLP tasks, especially in low-resource cases. To verify the adaptivity of the proposed GEANN, we further introduce BERT into models by replacing the word embedding with BERT representations. Figure 3 shows the results. We can see that even BERT can enhance NER, GEANN can still further achieve significant improvement over BERT regardless of the training data size. This verified GEANN can further capture task-specific name knowledge, which complement well with universal pretrained language knowledge.

## 5 Related Work

Sequential labeling approaches (Zhou and Su, 2002; Chieu and Ng, 2002; Bender et al., 2003; Settles, 2004; Lample et al., 2016) are widely used in NER. But this paradigm cannot handle nested mentions without specially designed tagging schema (Lu and Roth, 2015; Katiyar and Cardie, 2018; Wang and Lu, 2018; Lin et al., 2019). Recently, region-based models provide a natural solution for this issue. Finkel and Manning (2009) first proposed to classify over regions corresponding to parsing tree nodes. Xu et al. (2017) proposed to directly classify over all subsequences with a neural network model. Sohrab and Miwa (2018) extended their method by introducing a new region encoder. Generally, these methods have achieved promising results but heavily rely on fully-annotated data.

Gazetteers or dictionaries have long been regarded as a useful and easily-obtainable resource for NER. Previous methods commonly incorporated gazetteers by either using them to as handcraft features (Bender et al., 2003; Tsuruoka and Tsujii, 2003; Ciaramita and Altun, 2005; Minkov et al., 2005; Ritter et al., 2011; Seyler et al., 2018; Yu et al., 2018), or using them to generate data by distant supervision (Cohen, 2005; Ren et al., 2015; Giannakopoulos et al., 2017; Shang et al., 2018). However, the first kind of methods can not fully

leverage the inner mention structure knowledge entailing in gazetteers, while the second approaches will result in remarkable noise.

## 6 Conclusions

This paper first proposes *attentive neural networks*, an effective region-based model which explicitly models mention-context association. Then we propose to incorporate an auxiliary *gazetteer network* to enhance ANN. The gazetteer network can effectively learn name knowledge only using easily-available gazetteers, and therefore can significantly improve model performance and reduce data requirement. Experiments show that GEANN achieves the state-of-the-art performance on ACE2005 with much lower data requirement.

## Acknowledgments

We sincerely thank the reviewers for their insightful comments and valuable suggestions. Moreover, this work is supported by the National Natural Science Foundation of China under Grants no. 61433015, 61572477 and 61772505; and the Young Elite Scientists Sponsorship Program no. YESS20160177.

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Oliver Bender, Franz Josef Och, and Hermann Ney. 2003. Maximum entropy models for named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 148–151. Association for Computational Linguistics.

Hai Leong Chieu and Hwee Tou Ng. 2002. Named entity recognition: a maximum entropy approach using global information. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

Massimiliano Ciaramita and Yasemin Altun. 2005. Named-entity recognition in novel domains with external lexical knowledge. In *Proceedings of the NIPS Workshop on Advances in Structured Learning for Text and Speech Processing*, volume 2005.

Aaron M Cohen. 2005. Unsupervised gene/protein named entity normalization using automatically extracted dictionaries. In *Proceedings of the acl-ismb workshop on linking biological literature, ontologies and databases: Mining biological semantics*, pages 17–24. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jenny Rose Finkel and Christopher D Manning. 2009. Nested named entity recognition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 141–150. Association for Computational Linguistics.

Athanasios Giannakopoulos, Claudiu Musat, Andreea Hossmann, and Michael Baeriswyl. 2017. Unsupervised aspect term extraction with b-lstm & crf using automatically labelled datasets. *arXiv preprint arXiv:1709.05094*.

Arzoo Katiyar and Claire Cardie. 2018. Nested named entity recognition revisited. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 861–871. Association for Computational Linguistics.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Hongyu Lin, Yaojie Lu, Xianpei Han, and Le Sun. 2019. Sequence-to-nuggets: Nested entity mention detection via anchor-region networks. *arXiv preprint arXiv:1906.03783*.

Wei Lu and Dan Roth. 2015. Joint mention extraction and classification with mention hypergraphs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 857–867.

Einat Minkov, Richard C Wang, and William W Cohen. 2005. Extracting personal names from email: Applying named entity recognition to informal text. In *Proceedings of human language technology conference and conference on empirical methods in natural language processing*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Xiang Ren, Ahmed El-Kishky, Chi Wang, Fangbo Tao, Clare R Voss, and Jiawei Han. 2015. Clustype: Effective entity recognition and typing by relation phrase-based clustering. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 995–1004. ACM.

Alan Ritter, Sam Clark, Oren Etzioni, et al. 2011. Named entity recognition in tweets: an experimental study. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1524–1534. Association for Computational Linguistics.

Burr Settles. 2004. Biomedical named entity recognition using conditional random fields and rich feature sets. In *Proceedings of the international joint workshop on natural language processing in biomedicine and its applications*, pages 104–107. Association for Computational Linguistics.

Dominic Seyler, Tatiana Dembelova, Luciano Del Corro, Johannes Hoffart, and Gerhard Weikum. 2018. A study of the importance of external knowledge in the named entity recognition task. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 241–246.

Jingbo Shang, Liyuan Liu, Xiang Ren, Xiaotao Gu, Teng Ren, and Jiawei Han. 2018. Learning named entity tagger using domain-specific dictionary. *arXiv preprint arXiv:1809.03599*.

Mohammad Golam Sohrab and Makoto Miwa. 2018. Deep exhaustive model for nested named entity recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2843–2849. Association for Computational Linguistics.

Jinsong Su, Jiali Zeng, Deyi Xiong, Yang Liu, Mingxuan Wang, and Jun Xie. 2018. A hierarchy-to-sequence attentional neural machine translation model. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 26(3):623–632.

Yoshimasa Tsuruoka and Jun'ichi Tsujii. 2003. Boosting precision and recall of dictionary-based protein name recognition. In *Proceedings of the ACL 2003 workshop on Natural language processing in biomedicine-Volume 13*, pages 41–48. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Bailin Wang and Wei Lu. 2018. Neural segmental hypergraphs for overlapping mention recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 204–214. Association for Computational Linguistics.

Bailin Wang, Wei Lu, Yu Wang, and Hongxia Jin. 2018. A neural transition-based model for nested mention recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1011–1017. Association for Computational Linguistics.

Mingbin Xu, Hui Jiang, and Sedtawut Watcharawittayakul. 2017. A local detection approach for named entity recognition and mention detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1237–1247. Association for Computational Linguistics.

Xiaodong Yu, Stephen Mayhew, Mark Sammons, and Dan Roth. 2018. On the strength of character language models for multilingual named entity recognition. *arXiv preprint arXiv:1809.05157*.

GuoDong Zhou and Jian Su. 2002. Named entity recognition using an hmm-based chunk tagger. In *proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 473–480. Association for Computational Linguistics.

# "A Buster Keaton of Linguistics": First Automated Approaches for the Extraction of Vossian Antonomasia

**Michel Schwab,[1] Robert Jäschke,[1,2] Frank Fischer,[3] and Jannik Strötgen[4]**
[1]Humboldt University, Berlin, Germany
[2]L3S Research Center, Hannover, Germany
[3]National Research University Higher School of Economics, Moscow, Russia
[4]Bosch Center for Artificial Intelligence, Renningen, Germany
{michel.schwab,robert.jaeschke}@hu-berlin.de
ffischer@hse.ru
jannik.stroetgen@de.bosch.com

## Abstract

Attributing a particular property to a person by naming another person, who is typically well-known for the respective property, is called a *Vossian Antonomasia* (VA). This subtpye of metonymy, which overlaps with metaphor, has a specific syntax and is especially frequent in journalistic texts. While identifying Vossian Antonomasia is of particular interest in the study of stylistics, it is also a source of errors in relation and fact extraction as an explicitly mentioned entity occurs only metaphorically and should not be associated with respective contexts. Despite rather simple syntactic variations, the automatic extraction of VA was never addressed as yet since it requires a deeper semantic understanding of mentioned entities and underlying relations. In this paper, we propose a first method for the extraction of VAs that works completely automatically. Our approaches use named entity recognition, distant supervision based on Wikidata, and a bi-directional LSTM for postprocessing. The evaluation on 1.8 million articles of the New York Times corpus shows that our approach significantly outperforms the only existing semi-automatic approach for VA identification by more than 30 percentage points in precision.

## 1 Introduction

**Background.** Vossian Antonomasia (VA) is a stylistic device which attributes a certain property to a person by naming another (more well-known, more popular) person as a reference point. For instance, when Jim Koch is described as "the Steve Jobs of Beer" (Fallows, 2014), certain qualities of Steve Jobs, be it entrepreneurship or persuasiveness, are assigned to Jim Koch, co-founder and chairman of the Boston Beer Company.

VA is named after Gerardus Vossius (1577–1649), the Dutch classical scholar and author of rhetorical textbooks. Although the phenomenon is traceable back to antiquity (as shows the example of Crassus, who used to be ironically referred to as "the Palatine Venus"), it was first distinguished and described as a separate phenomenon by Vossius.

It constitutes a sub-phenomenon of the classic antonomasia. In the classification of Holmqvist and Płuciennik (2010) it is called *metaphorical antonomasia* or *antonomasia*$_2$ and is described as a comparison "with paragons from other spheres". VAs are particularly popular in journalistic texts.

**Definition.** VAs consist of three parts: a *source* (in our example "Steve Jobs") serves as paragon to elevate the *target* ("Jim Koch") by applying a *modifier* ("of beer") that provides the corresponding context (see Bergien, 2013). Although in most cases both the source and the target are persons, one or both of them could be almost anything as long as they bear a proper name, for instance, software, as shows this example: "Word [...] was dubbed the Marquis de Sade of word processors, which was not altogether unfair." (NYT 1993/09/26/0636952)[1] The target is not necessarily part of the sentence (in particular when a VA is used in a headline – like in the title of this paper[2]) and it can be hypothetical, as in "Who is the Tolstoy of the Zulus?" (NYT 1988/01/03/0106769)

**Challenges.** Despite some interest in VAs in the humanities (e.g., Holmqvist and Płuciennik, 2010; Bergien, 2013), works on their automatic extraction are scarce. One reason could be that "antonomasia is essentially humanistic and anti-computational" (Holmqvist and Płuciennik, 2010) and thus identifying and understanding it in its entirety is often difficult even for humans and requires deep cultural

[1]To avoid an excessively long reference list, all examples taken from the New York Times corpus (Sandhaus, 2008) are cited using the pattern "NYT year/month/day/article-id".

[2]To resolve this example: "a Buster Keaton of linguistics" refers to the performance artist Stuart Sherman (NYT 1987/07/19/0057747).



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6238–6243,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

background knowledge. For example, understanding the phrase "It's the Dolly Parton of cakes" (NYT 2007/02/14/1826062) requires specific knowledge about Dolly Parton, her skills or peculiarities. Only in rare cases like this, an explanation of the intended meaning ("a little bit tacky, but you love her") is provided. It is usually left to the reader to make sense of a VA.

**Importance.** From a humanities perspective, a VA constitutes an interesting phenomenon of enculturation (Holmqvist and Płuciennik, 2010) that deserves to be studied more in-depth, based on larger corpora.

However, we expect that the large-scale identification of VAs and other such stylistic devices is not only important in the humanities and for cultural reasons, but also in natural language processing to avoid mistakes in tasks such as machine translation and fact extraction. For instance, given the sentence "Today, the German Ronaldo quit his career.", we do not want to extract a fact that Ronaldo quit his career, but we need to detect that "the German Ronaldo" is a VA referring to a German soccer player.

In addition, entity disambiguation and coreference resolution could be improved. Consider the example "Jimmy Johnson is the Madonna of college football ..." (NYT 1987/01/02/0000431). We would like to refer "the Madonna of college football" to Jimmy Johnson, and, in particular, we want to avoid that Madonna is part of any coreference chain.

It could also help in new interesting question answering tasks, for example, "Who is the Bill Gates of Japan?", where the sentence "... Mr. Horie has made headline news with a success story that has turned him into the Bill Gates of Japan." (NYT 2006/01/19/1733197) could be one answer to this question.

Generally speaking, successful detection and resolution of VAs could be a further step towards full natural language understanding.

**Contribution.** We propose the first effective fully automated approaches for extracting VAs from texts. In our evaluation, we compare them against a baseline and assess the difficulty of identifying VAs using crowdsourcing. In addition, we extend the only existing larger data set (Fischer and Jäschke, 2019) with VAs of eight additional patterns. We have also double-checked and corrected existing and new annotations. Our code and data are freely[3] available[4] and easily extensible towards other patterns and entity types.

## 2 Related Work

Very similar to metaphor detection (Tsvetkov et al., 2014; Gao et al., 2018; Mao et al., 2018), identifying VAs is a difficult task. So far, we are dealing with manually curated collections like the Wall Street Journal's collection of occurrences of "Michael Jordans of ..." (Cohen et al., 2015). The only attempt automating the extraction has been presented by Fischer and Jäschke (2019). Their semi-automatic approach for English texts extracts VA candidates for only one single pattern, *the ENTITY of*. Using a regular expression to identify candidates, names and aliases of Wikidata entities whose 'instance-of' property is 'human' as a first filter, and a manually created blacklist as second filter, they extract about 3,700 VA candidates. 70% of them have been confirmed as VAs by a domain expert.

Since the phenomenon is a kind of metonymy, its detection is closely related to approaches to metonymy resolution. This has been covered extensively, specifically by Markert and Nissim, for example, in (Nissim and Markert, 2003). They compare this task to word-sense disambiguation and propose a thesaurus-based classification approach. Figures of speech that have been covered include, for instance, metaphors (Tsvetkov et al., 2014; Gao et al., 2018; Mao et al., 2018) and irony (Akhtyrska, 2014). The supervised approach of Tsvetkov et al. (2014), for instance, can quite reliably "discriminate whether a syntactic construction is meant literally or metaphorically" but is very specific, focused on subject–verb–object and adjective–noun phrases.

## 3 Corpus Creation and Baseline

To create a corpus of candidate phrases and manually confirmed VAs, we use the dataset of Fischer and Jäschke (2019) as starting point and their method as baseline. However, instead of relying only on one individual syntactic pattern, *the ENTITY of*, we extended the candidate generation process with the patterns *the ENTITY among*, *the ENTITY for*, and six variations thereof where *the* is

[3] GPL3 and CC-BY, respectively

[4] https://vossanto.weltliteratur.net/ – in addition to our code and data, information about some of the authors' prior work on VAs is also available.

| pattern | regex | Wikidata | blacklist | true VA |
|---|---|---|---|---|
| the-of | 12,748,735 | 90,712 | 3,591 | 2,779 |
| a-of | 5,900,839 | 11,860 | 705 | 118 |
| an-of | 956,247 | 4,539 | 88 | 14 |
| the-for | 2,960,459 | 8,070 | 817 | 24 |
| a-for | 1,869,946 | 4,812 | 536 | 59 |
| an-for | 304,529 | 1,424 | 296 | 13 |
| the-among | 122,345 | 139 | 13 | 3 |
| a-among | 67,019 | 82 | 25 | 13 |
| an-among | 11,158 | 12 | 1 | 0 |
| **sum** | 24,941,277 | 121,650 | 6,072 | 3,023 |

Table 1: Number of VA candidates after each step and manually confirmed VAs for all used patterns.

replaced by *a* or *an*, respectively. We identified these patterns to be the most prevalent patterns in a preliminary analysis of a manually maintained collection of VAs.

Using the New York Times corpus (Sandhaus, 2008) of 1,854,726 articles published between 1987 and 2007, the regular expression based pattern matching resulted in almost 25 million candidate phrases vs. less than 13 million candidate phrases in Fischer and Jäschke (2019)'s work.

Using Wikidata for distant supervision, we keep only candidates whose string matches the exact name or alias of Wikidata entities, respecting the case of letters, with 'instance-of' property 'human', and exclude candidates using the manually curated blacklist[5] provided by Fischer and Jäschke (2019).[6] We consider this procedure of "regex – Wikidata – blacklist" as baseline. In Table 1, we show the number of VA candidate phrases after each step.

In addition, all remaining candidates were manually annotated by a domain expert with 10 years of experience in the research of VAs and samples were double-checked by a trained student. The resulting corpus contains 6,072 VA candidates, 3,023 of them were classified to be true VAs by the human annotators. That is, the baseline approach has a precision of 49.8% (see Table 1). Note that due to the sparseness of the phenomenon, we did not try to estimate the recall of the baseline approach.

[5]words that are also common nouns or otherwise unlikely to be the source of a VA

[6]Like Fischer and Jäschke (2019), we focus on humans as source as they are the most frequent type; other types like 'team' or 'company' are left for future work.

## 4 VA Extraction Approaches

We aim at a fully automated extraction of VAs from texts. For this, we develop two approaches to replace the need of a manually curated blacklist of the baseline approach using named entity recognition as well as Wikidata and a link-based popularity measure. In addition, as a further approach, we extend the baseline by classifying VA candidates using a bi-directional LSTM.

### 4.1 Wikidata (WD)

This is a modification of the baseline approach to avoid the manual curation of a blacklist. Thus, after the regex step and linking the source candidates with the Wikidata list of humans as in the baseline approach, we assess their popularity within Wikidata.

The rationale is that a prerequisite for being a source of a VA expression is to be famous or popular or notorious for something. Otherwise, the VA is less likely to be understood by a reader. This arguably goes along with being present on several of the almost 300 international Wikipedia versions. Therefore, we remove source candidates whose name also matches the label or alias of a non-human Wikidata entity that has more sitelinks[7] (i.e., is more popular according to our measure) than the linked human. For instance, the candidate 'the House of', that matched the American botanist Homer Doliver House[8] in the first step, will be removed since in the second step the entity 'house'[9] (the building, not the botanist) has more sitelinks ($178 > 9$). The popularity measure could easily be changed to any other measure like statements or using Wikipedia clickstream data.

In addition, we remove all candidates where the source (e.g., 'Prince') together with multiple words following it (e.g., 'of Wales') match the name or alias of another Wikidata entity (https://www.wikidata.org/wiki/Q43274). This allows us to remove frequent false positives like 'Prince of Wales'.

### 4.2 Named Entity Recognition (NER)

This approach is built on the regex step from the baseline approach as well, but we replace the way of identifying 'persons': instead of restricting the extraction to VAs with Wikidata entities, we per-

[7]https://wikidata.org/wiki/Help:Sitelinks

[8]https://www.wikidata.org/wiki/Q3139666

[9]https://www.wikidata.org/wiki/Q3947

form Named Entity Recognition on the sentence candidates using the Stanford three-class named entity tagger (Finkel et al., 2005).

When all words of the candidate source (i.e., the words between the first and the last word of the regular expression match) are tagged as 'PERSON', we consider the candidate a positive match. However, to avoid false positives, we apply the last step from the WD approach (Sec. 4.1) and remove again all candidates whose source together with following words match another Wikidata entity.

### 4.3 Bi-directional Long Short-Term Memory Neural Network (BLSTM)

We use the baseline data of 6,072 VA candidates with 3,023 confirmed VAs to train and test a BLSTM on whole sentences (Schuster and Paliwal, 1997; Graves and Schmidhuber, 2005) using 5-fold cross validation. The BLSTM classifies sentences on whether they contain a VA expression or not. We represent each word from a sentence candidate with a pre-trained word embedding. We use word embeddings from GloVe which consist of 300-dimensional vectors that were trained on a Google News corpus (Pennington et al., 2014). We implement the BLSTM in Keras (Chollet et al., 2015) with Tensorflow backend, using Adam Optimizer and default hyperparameters (epochs=80, batch size=128, hidden units=300, dropout=0.25). We expect that further improvements may be possible if a fine-grained hyperparameter search was to be conducted.

## 5 Evaluation and Results

Determining an overall *recall* of our approaches on the New York Times corpus is unfeasible as it contains 1.8 million articles. Thus, we used a random sample of 105 articles (for each year 5). However, we found only one VA: "If Nike is the Chicago Bulls of the athletic shoe market, retailers are holding their breath for a strong underdog player to emerge." (NYT 1997/06/07/0935205). Due to our focus on individuals, it is no VA that should be part of our data set. Thus, we can only determine *precision* and *recall* of our new approaches based on the baseline data set described in Section 3.

### 5.1 Difficulty of the Task

Identifying VAs is a non-trivial task even for humans. Thus, we used *expert knowledge* to evaluate candidates for the ground truth data (see Section 3).

| approach | prec | rec | f1 |
|---|---|---|---|
| baseline | 49.8% | — | — |
| WD | 67.3% | 93.0% | 78.1% |
| NER | 71.8% | 81.3% | 76.2% |
| BLSTM | 86.9% | 85.3% | 86.1% |

Table 2: Performance of the three proposed approaches in comparison with the baseline.

In addition, we leveraged *crowdsourcing* to check whether we could use (untrained) people to evaluate VA candidates. Therefore, workers on the crowdsourcing platform Figure Eight had to check whether a sentence contained a VA.[10]

Crowdsourcing resulted in 600 judgments for 200 randomly selected candidates of our baseline data set with an inter-annotator agreement calculated by Cohen's Kappa of 0.72 between expert and workers. Re-checking the 28 disagreements showed that the expert judged all correctly, which results in a crowdsourcing accuracy of 86%. This shows how difficult it is to identify VAs.

### 5.2 Baseline VA Candidates

We cannot determine recall for our approaches, instead we compute precision, recall and f-score based on the baseline data set (see Section 3), which is shown in Table 2 as well as precision of the baseline data set. The automated approaches using Wikidata and named entity recognition can boost the precision significantly with a moderate loss in recall. The BLSTM approach performs best. Although the loss in recall is higher than with the WD approach, the precision reaches almost 87% and is thus raised to a new level.

### 5.3 Non-Baseline VA Candidates

The results of our approaches are not limited to the baseline data set. The WD approach generated 955 new VA candidates, the NER approach 4,399. The human annotators evaluated 100 randomly selected VA candidates of each set, which resulted in a precision of 17% (WD) and 30% (NER). We also predicted the labels of these random samples with the trained BLSTM which resulted in 83% and

[10] https://www.figure-eight.com/. We highlighted the source candidate and included its Wikidata link. To maintain quality, we set test questions where workers had to pass a threshold (70%). We asked for three judgments per sentence and used majority voting. We paid 2 US Cents per judgment, only allowing workers with experience level 2 or higher.

75% precision on the WD approach and the NER approach, respectively. This shows that the BLSTM does not only have potential to improve the results of the baseline approach but also of the WD and NER approaches.

### 5.4 Error Analysis

About 13% of the WD approach errors (based on baseline) were false negatives, for example, "an Edison of magic". Here, "Thomas Edison" would have been the right source, but as the entity "Thomas Edison" has no alias "Edison", the most popular entity having 'Edison' as name or alias is a town in the United States.[11] 87% were false positives like "the Michelangelo of the Sistine Chapel" where the person name was not used as a VA source but stands for a specific work period of the artist. In the sample set that we selected randomly from the WD approach candidates which did not appear in the baseline (see Section 5.3), we detected false positives like "an Air of Mystery" as "Air" is an alias of Michael Jordan, who has more sitelinks than the word "air" (116 to 113).

The errors from the NER approach (based on baseline) had around 36% false negatives like "the Marco Polo of baseball" where the source was not detected as a person by the named entity tagger. 64% were false positives like "the Dave Brown of old". The NER approach sample set from Section 5.3 contained new false positive candidates because awards ("the Harriet H. Jonas Award of"), institutions ("the O. K. Harris Gallery of"), or titles ("the Episcopal Bishop of") were falsely tagged as persons.

### 5.5 Discussion

Since we presented the first approaches for the extraction of VAs that work completely automatically, our work has some limitations that we discuss in the following. The restriction to one type of source can be solved easily, for instance, by choosing not only 'humans' but proper nouns in general, as well as allowing further syntactic variations of the source phrase, for example, 'the ADJECTIVE ENTITY' ("the new Michael Jordan") or 'the ADJECTIVE/NOUN sort/version/equivalent of' ("the Georgian equivalent of") but (probably) at the expense of precision.

Beyond extending recall by tackling further syntactic variations and allowing non-human sources, we currently aim to extract the modifier and target of VAs.

Another goal is to train a neural network with a larger and better balanced training set to use the model to study a larger corpus. Alternatively, a pre-trained model could be used and fine-tuned with our labeled data to improve results.

Besides above-mentioned limitations which we plan to address in future work, we also want to report on approaches that have not lead to any improvements. For example, we tested measuring the similarity between source and modifier using word embeddings (with the assumption that a low similarity should indicate a VA). Unfortunately, there are too many neutral modifiers like "his time" or subgenres of the source like "the Tiger Woods of micro golf" where this idea does not work. Furthermore, removing candidates whose source was contained in WordNet (Fellbaum, 1998) and not labeled as 'person' did not work, since "wife", "chancellor", and so forth were labeled as persons in WordNet.

## 6 Conclusions

In this paper, we presented approaches for the first fully automated extraction of the stylistic device *Vossian Antonomasia*.

In addition, we were able to create the largest known collection of VAs by looking into 21 years of the New York Times. The data set contains the annotated list of 6,072 VA candidates, each with article id, date, URL, marked source and modifier, Wikidata id, and class.

The best approach, using a BLSTM, reached a precision of 86.9% and a recall of 85.3%.

[11] https://www.wikidata.org/wiki/Q746801

## References


Kateryna Akhtyrska. 2014. *Linguistic Expression of Irony in Social Media*. Ph.D. thesis, Universidade do Algarve.

Angelika Bergien. 2013. Names as frames in current-day media discourse. In *Name and Naming. Proceedings of the second international conference on onomastics*, pages 19–27, Cluj-Napoca. Editura Mega.

François Chollet et al. 2015. Keras. https://keras.io.

Ben Cohen, Geoff Foster, Matthew Oshinsky, Jon Keegan, and Tom McGinty. 2015. The Michael Jordan of .... Website.

James Fallows. 2014. The Steve Jobs of Beer. *The Atlantic*, (November 2014).

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. The MIT Press, Cambridge, MA.

Frank Fischer and Robert Jäschke. 2019. 'The Michael Jordan of greatness'—Extracting Vossian antonomasia from two decades of The New York Times, 1987–2007. *Digital Scholarship in the Humanities*, (fqy087).

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics.

Ge Gao, Eunsol Choi, Yejin Choi, and Luke Zettlemoyer. 2018. Neural Metaphor Detection in Context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 607–613. Association for Computational Linguistics.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5):602–610.

Kenneth Holmqvist and Jarosław Płuciennik. 2010. Princess Antonomasia and the Truth: Two Types of Metonymic Relations. In Armin Burkhardt and Brigitte Nerlich, editors, *Tropical Truth(s). The Epistemology of Metaphor and Other Tropes*, pages 373–381. De Gruyter, Berlin/New York.

Rui Mao, Chenghua Lin, and Frank Guerin. 2018. Word Embedding and WordNet Based Metaphor Identification and Interpretation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 1222–1231. Association for Computational Linguistics.

Malvina Nissim and Katja Markert. 2003. Syntactic features and word similarity for supervised metonymy resolution. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 56–63. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543. Association for Computational Linguistics.

Evan Sandhaus. 2008. The New York Times Annotated Corpus LDC2008T19. DVD, Linguistic Data Consortium, Philadelphia.

Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *Transactions on Signal Processing*, 45(11):2673–2681.

Yulia Tsvetkov, Leonid Boytsov, Anatole Gershman, Eric Nyberg, and Chris Dyer. 2014. Metaphor Detection with Cross-Lingual Model Transfer. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 248–258. Association for Computational Linguistics.

# Multi-Task Learning for Chemical Named Entity Recognition with Chemical Compound Paraphrasing

**Taiki Watanabe**[1,3]* watanabe-taiki@fujitsu.com
**Akihiro Tamura**[2,3] **Takashi Ninomiya**[2,3] {tamura,ninomiya}@cs.ehime-u.ac.jp
**Takuya Makino**[1,3] **Tomoya Iwakura**[1,3] {makino.takuya,iwakura.tomoya}@fujitsu.com
[1] Fujitsu Laboratories, Ltd. [2] Ehime University [3] RIKEN AIP-FUJITSU Collaboration Center

## Abstract

We propose a method to improve named entity recognition (NER) for chemical compounds using multi-task learning by jointly training a chemical NER model and a chemical compound paraphrase model. Our method enables the long short-term memory (LSTM) of the NER model to capture chemical compound paraphrases by sharing the parameters of the LSTM and character embeddings between the two models. The experimental results on the BioCreative IV's CHEMDNER task show that our method improves chemical NER and achieves state-of-the-art performance (+1.43 F-score).

## 1 Introduction

Named Entity Recognition (NER) is one of the important basic technologies for Natural Language Processing (NLP) such as Information Extraction and Entity Linking. LSTM-CRF NER models, which combine a conditional random field (CRF) and a long short-term memory (LSTM), have achieved high performance (Lample et al., 2016; Ma and Hovy, 2016). The LSTM-CRF models that use a neural language model (NLM) pre-trained from a large-scale unlabeled corpus (Akbik et al., 2018; Peters et al., 2018) have shown state-of-the-art performances on the CoNLL 2003 shared task (Sang et al., 2003).

Chemical compound databases are widely used for investigating properties of chemical compounds or for developing new chemical products. However, updating the databases by hand is hard because new findings on chemical compounds are mainly reported in scientific papers and patents every day. Hence, chemical NER has been studied to recognize chemical compounds from chemical documents (Leaman et al., 2015; Lu et al., 2015; Lin et al., 2018).

One of the problems in chemical NER is notation variants of chemical compounds. For example, *Phenylalanine* has different notations such as *L-β-phenylalanine* and *(S)-2-Benzylglycine*. The average number of notations of each compound in PubChem[1] is approximately 3.88.[2]

If these expressions are dealt with differently, the statistics of the same compound can be dispersed, especially for low frequency compounds. In other words, the more distinct these representations are, the more difficult identifying chemical compound entities becomes. However, existing chemical NER methods do not deal with notation variants of chemical compounds, derived from the partial structures or the notation fluctuation peculiar to these chemical compounds.

We propose *HanPaNE*, which *Han*dles *Pa*raphrase in *NE*R by utilizing chemical compound paraphrase pairs as multi-task learning.

To train expression identity of different notations, *HanPaNE* learns shared parameters between paraphrases using multi-task learning on NER and paraphrase generation. This contrasts with widely used approaches that automatically augment training data by replacing NEs with other NEs (Yi et al., 2004). To train paraphrase generation, we use an attention-based neural machine translation (ANMT) model (Luong et al., 2015; Bahdanau et al., 2015) that shares parameters with the NER model in the translation encoder. The experimental results on the BioCreative IV's CHEMDNER task (Krallinger et al., 2015) show that our method achieves the best accuracy (+1.43 F-score).

*Taiki Watanabe belonged to Ehime University when this work was done.

[1] https://pubchem.ncbi.nlm.nih.gov/
[2] We surveyed the average number on May, 2019.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6244–6249, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

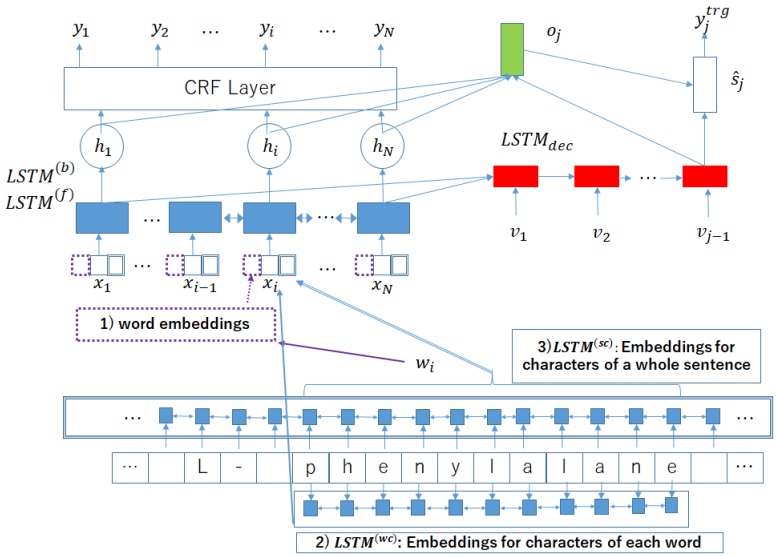


Figure 1: An overview of the method *HanPaNE*. The parameters of $LSTM^{(f)}$, $LSTM^{(b)}$, $LSTM^{(wc)}$ and $LSTM^{(sc)}$ are shared in NER and paraphrase generation.

## 2 Proposed Method

Figure 1 shows an overview of the method *HanPaNE*. We first describe our NER and chemical compound paraphrase model, and then describe our multi-task learning of the NER and the paraphrase model.

### 2.1 NER using Character-level NLM

This section describes the NER model using a character-level NLM (Akbik et al., 2018), which is our baseline comparison with our approach. This model showed state-of-the-art performance on the CoNLL 2003 shared task.

For this NER model, given an input sentence $\mathbf{w} = w_1, w_2, \cdots, w_N$, each word $w_i$ is converted into a vector $\mathbf{x}_i$ represented as the concatenation of the following three types of embeddings.

- 1) word embeddings obtained by looking up a parameter matrix.
- 2) CWSE is the hidden state of a BiLSTM ($LSTM^{(wc)}$) on the character sequence consisting of a word (Lample et al., 2016).
- 3) CSE is the hidden state of a BiLSTM ($LSTM^{(sc)}$) on the character sequence consisting of a whole sentence (Akbik et al., 2018).

CWSE for a word is the concatenation of the last hidden states of the forward LSTM and the backward LSTM of $LSTM^{(wc)}$. CSE for a word is the concatenation of the hidden state of the last character of the word for the forward LSTM of $LSTM^{(sc)}$, and the hidden state of the first character of the word for the backward LSTM of $LSTM^{(sc)}$. We note that $LSTM^{(sc)}$ is pretrained as character-level NLMs with log-likelihood. For example, a forward LSTM is trained with the following:

$$P(\mathbf{c}_{1:T}) = \prod_{t=1}^{T} p(c_t|\mathbf{c}_{1:t-1}), \quad (1)$$

where $c_t$ is the $t$-th character in a sentence.

After obtaining the vector representations of the input words $\mathbf{X} = \mathbf{x}_1, \cdots, \mathbf{x}_N$, the hidden state $\mathbf{h}_i$ of BiLSTM and the scores of NE tags $\mathbf{e}_i$ for each word $w_i$ are computed as:

$$\overrightarrow{\mathbf{h}_i} = LSTM^{(f)}(\mathbf{x}_i, \overrightarrow{\mathbf{h}_{i-1}}), \quad (2)$$
$$\overleftarrow{\mathbf{h}_i} = LSTM^{(b)}(\mathbf{x}_i, \overleftarrow{\mathbf{h}_{i+1}}), \quad (3)$$
$$\mathbf{h}_i = [\overrightarrow{\mathbf{h}_i}; \overleftarrow{\mathbf{h}_i}], \quad (4)$$
$$\mathbf{e}_i = W_e \mathbf{h}_i, \quad (5)$$

where the dimension of $\mathbf{h}_i$ is $d$, $W_e \in \mathcal{R}^{k \times d}$ is a weight matrix, and $k$ is the number of NE types to be identified.

The BiLSTM-CRF+CSE model predicts a tag sequence with the score matrix $\mathbf{P} = (\mathbf{e}_1, \cdots, \mathbf{e}_N)^T \in \mathcal{R}^{N \times k}$ using CRF. Note that $P_{i,j}$ is the score of the $j$-th tag for the $i$-th word. In particular, the probability of the output tag sequence

$\mathbf{y} = y_1, \cdots, y_N$ is calculated as follows:

$$p(\mathbf{y}|\mathbf{w}) = \frac{e^{s(\mathbf{w},\mathbf{y})}}{\sum_{\tilde{\mathbf{y}} \in \mathbf{Y_w}} e^{s(\mathbf{w},\tilde{\mathbf{y}})}}, \quad (6)$$

$$s(\mathbf{w}, \mathbf{y}) = \sum_{i=-1}^{N} A_{y_i, y_{i+1}} + \sum_{i=1}^{N} P_{i, y_i}, \quad (7)$$

where $\mathbf{Y_w}$ is the set of all possible tag sequences, $\mathbf{A}$ is a matrix of transition scores, $A_{i,j}$ represents a score that transits from the $i$-th tag to the $j$-th tag, and $y_{-1}$ and $y_{N+1}$ are the special tag for the start of the sentence and the end of sentence, respectively.

During training, the model maximizes the following equation using the correct tag sequences:

$$\log(p(\mathbf{y}|\mathbf{w})) = s(\mathbf{w}, \mathbf{y}) - \log(\sum_{\tilde{\mathbf{y}} \in \mathbf{Y_w}} e^{s(\mathbf{w},\tilde{\mathbf{y}})}), \quad (8)$$

where $\mathbf{y}$ is the correct tag sequence of $\mathbf{w}$. When recognizing NEs, the model outputs a tag sequence that maximizes the score calculated by the following equation: $\mathbf{y}^* = \arg\max_{\tilde{\mathbf{y}} \in \mathbf{Y_w}} s(\mathbf{w}, \tilde{\mathbf{y}})$.

### 2.2 Paraphrase Model

We used the ANMT model (Luong et al., 2015; Bahdanau et al., 2015), which is a standard in machine translation, as a chemical compound paraphrase model. The ANMT model converts an input sequence $\mathbf{w}$ into another sequence $\mathbf{y}^{trg} = y_1^{trg}, y_2^{trg}, \cdots, y_T^{trg}$, which is a paraphrase of the input in our model, using an RNN encoder and an RNN decoder. The RNN encoder converts an input sequence to a multiset of fixed-length vectors, and then the RNN decoder generates an output sequence from the converted fixed length vector. We use a bidirectional LSTM defined by Eq. (2) and Eq. (3) as the encoder, and the vector representations $\mathbf{x}_i$ as the word embedding vectors of input sentences. The concatenation of the final hidden states of the bidirectional LSTM encoder, $\overrightarrow{\mathbf{h}}_N$ of Eq. (2) and $\overleftarrow{\mathbf{h}}_1$ of Eq. (3), was used as the initial state of the LSTM decoder: $\mathbf{s}_1 = [\overrightarrow{\mathbf{h}}_N; \overleftarrow{\mathbf{h}}_1]$.

The decoder calculates the $j$-th hidden state $\mathbf{s}_j$ as follows: $\mathbf{s}_j = LSTM_{dec}([\mathbf{v}_{j-1}; \hat{\mathbf{s}}_{j-1}], \mathbf{s}_{j-1})$, where $\mathbf{v}_{j-1}$ is the word embedding vector of $y_{j-1}^{trg}$, and $\hat{\mathbf{s}}_{j-1}$ is the $(j-1)$-th attention vector. Then, the model calculates the $j$-th attention vector $\hat{\mathbf{s}}_j$ using the context vector $\mathbf{o}_j$: $\hat{\mathbf{s}}_j = \tanh(W_e[\mathbf{s}_j : \mathbf{o}_j])$. where $W_e$ is a weight matrix and tanh is the hyperbolic tangent function. The context vector $\mathbf{o}_j$ is a weighted average of the encoder's hidden states:

$$\mathbf{o}_j = \sum_{i=1}^{N} \alpha_j(i)\mathbf{h}_i, \quad (9)$$

$$\alpha_j(i) = \frac{\exp(\mathbf{h}_i \cdot \mathbf{s}_j)}{\sum_{k=1}^{N} \exp(\mathbf{h}_k \cdot \mathbf{s}_j)}, \quad (10)$$

where exp is the natural exponential function. The decoder generates an output based on the probability distribution of the $j$-th token: $p(y_j^{trg}|\mathbf{y}_{<j}^{trg}, \mathbf{w}) = \mathrm{softmax}(W_s\hat{\mathbf{s}}_j)$, where $W_s$ is a weight matrix. The objective function is defined as follows:

$$J(\theta) = -\sum_{(\mathbf{w},\mathbf{y^{trg}}) \in \mathbf{D}} \log p(\mathbf{y}^{trg}|\mathbf{w}), \quad (11)$$

where $D$ is the training data and $\theta$ is the set of model parameters.

### 2.3 Handling Paraphrases in NER

Our method, *HanPaNE*, learns the NER model described in Section 2.1 and the chemical compound paraphrase model described in Section 2.2 at the same time through multi-task learning. In the multi-task learning, the character embedding weight matrix and the LSTM parameters in Eq. (2), Eq. (3) and the parameters of $LSTM^{(wc)}$ are shared between the two models. By the parameter sharing, the LSTM part of the NER model is expected to convert the same compound with different notations into a similar vector expression. The objective functions of the two models are Eq. (8) and Eq. (11), respectively.

## 3 Experiments

### 3.1 Experimental Settings

We used the BioCreative IV CHEMDNER data set, which was preprocessed by Luo et al (2018)[3]. The word embedding layer with Word2vec and the character-based NLM were independently pretrained from the MEDLINE abstracts from the PubMed website (PubMedAbs)[4]. We created PubChemDic, a set of paraphrases of chemical compounds compiled from PubChem. The number of dimensions of the word embedding layer, the character embedding layer, the character LSTM, the LSTM of the NER model (the LSTM encoder of the paraphrase model), the LSTM decoder of the paraphrase model, and the LSTM of the character

[3] https://github.com/lingluodlut/Att-ChemdNER

[4] https://www.nlm.nih.gov/databases/download/pubmed_medline.html

| Model | Precision | Recall | F-score |
|---|---|---|---|
| Baseline | 92.75 | 92.15 | 92.45 |
| VE-P | 93.11 | 91.40 | 92.25 |
| *HanPaNE*-P | 92.71 | 91.94 | 92.32 |
| VE+P | **93.15** | 91.79 | 92.47 |
| *HanPaNE*+P (Proposed) | 92.81 | **92.33** | **92.57** |

Table 1: Experimental results. +P indicates consideration of paraphrases and -P does not.

NLM are set to 100, 25, 50, 200, 400, and 2048, respectively.

We compared the following methods. "+P" indicates with consideration of paraphrases in PubChemDic and "-P" indicates without consideration of paraphrases.

- A BiLSTM-CRF+CSE of (Akbik et al., 2018) described in Section 2.1 was used as our Baseline.
- VE-P is a baseline trained with virtual examples (VEs) created by randomly replacing NEs of training data with chemical compounds in the PubChemDic similar to (Yi et al., 2004).
- VE+P is a baseline trained with VEs created by replacing NEs of training data with their corresponding paraphrases in the PubChemDic.
- *HanPaNE*-P is a multi-task for NER and paraphrasing trained with randomly generated sentence pairs with PubChemDic.
- *HanPaNE*+P is the proposed method trained with generated sentence pairs by replacing NEs in sentences with their corresponding paraphrases in the PubChemDic.

The baseline was trained with 55,458 sentences of CHEMDNER training data. VE-P was trained with 110,916 sentences in total consisting of the original CHEMDNER training data and 55,458 sentences automatically generated from the CHEMDNER training data. VE+P was trained with 59,033 sentences consisting of 3,575 sentences automatically generated from the original CHEMDNER training data with paraphrases of chemical compounds and the original CHEMDNER training data.[5] For *HanPaNE*-P and *HanPaNE*+P, we used randomly selected 100,000 sentences from PubMedAbs for training paraphrasing and the original CHEMDNER training data for NER.

For example, for +P, "... *Phenylalanine* is ..." are converted into "... *L-β-phenylalanine* is ..." and ".. *(S)-2-Benzylglycine* is ...", where *L-β-phenylalanine* and *(S)-2-Benzylglycine* are paraphrases of *Phenylalanine*. As for -P, "... *Phenylalanine* is ..." is converted into like "... *ethylene* is ...." where *ethylene* is randomly sampled from chemical compounds.

### 3.2 Experimental Results

Table 1 shows the experimental results. We can see that *HanPaNE*+P showed the highest accuracy and *HanPaNE*+P and VE+P, with consideration of paraphrases, showed a higher accuracy than Baseline. In contrast, *HanPaNE*-P and VE-P, without consideration of paraphrases, did not. The results indicate that the use of paraphrases contributed to improved accuracy.

We also conducted the following two types of hypothesis testing. The first one is a McNemar paired test on the labeling disagreements of words assigned by *HanPaNE* and the others as in (Sha and Pereira, 2003). All the results except for Baseline were significantly different ($p < 0.01$). The second one is a bi-nominal test used in (Sasano and Kurohashi, 2008). For this test, the number of the entities correctly recognized by only *HanPaNE* and the number of entities correctly recognized by only the other method are counted. Then, based on the assumption that outputs have the binomial distribution, we apply a binomial test. All the results were significantly different for this test ($p < 0.05$). These results showed that *HanPaNE* works better than augmented training data.

We also evaluated the accuracy on NEs not covered by training data and PubChemDic. From Table 2, we can see that our method also showed the best performance for both of the NEs not covered by the training data, PubChemDic and the covered NEs. The results indicate that our method contributed to recognizing NEs not covered by existing large data sets. The training data includes 8,508 types of chemical terms and covers 60.76% of chemical terms in the test data. PubChemDic includes 337,289,536 types of chemical compound names and covers 10.02% of the chem-

[5] VE+P had a smaller number of training data than VE-P because we only replaced NEs with their paraphrases included in PubChemDic.

| Model | Acc.NC | Acc.C |
|---|---|---|
| Baseline | 0.8475 | 0.9672 |
| VE+P | 0.8463 | 0.9623 |
| *HanPaNE*+P (Proposed) | **0.8510** | **0.9680** |

Table 2: Accuracy of NEs covered (Acc.C) or not covered (Acc. NC) by the training data and PubChemDic.

Table 3: Comparison with best results on BioCreative IV's CHEMDNER task.

| Method | Precision | Recall | F-score |
|---|---|---|---|
| Leaman et al. (2015) | 89.09 | 85.75 | 87.39 |
| Lu et al. (2015) | 88.73 | 87.41 | 88.06 |
| Lin et al. (2018) | 92.29 | 90.01 | 91.14 |
| **This paper** | **92.81** | **92.33** | **92.57** |

ical terms in the test data. The joint coverage of both is 61.79%. This means almost 1/3 of the NEs in the test data are not covered by the training data and PubChemDic even if these data include over 337 million names of chemical compounds and terms.

## 4 Related Works

### BioCreative IV's CHEMDNER task

Table 3 shows a comparison with the previous best results. (Leaman et al., 2015) and (Lu et al., 2015) proposed a feature-based approach to improve the chemical NER performance. (Lin et al., 2018) proposed a neural network approach that treats document level information to maintain tagging consistency across sentences. By learning paraphrasing, our method showed the best accuracy on the CHEMDNER task.

### Multi-task learning

Multi-task learning is often utilized to leverage the performance of NLP systems (Liu et al., 2015; Luong et al., 2016; Dong et al., 2015; Hashimoto et al., 2017), including NER. Liu et al. (2018) and Rei (2017) studied multi-task learning of sequence labeling with language models. Aguilar et al. (2018) and Cao et al. (2018) proposed multi-task learning of NER with word segmentation. Peng and Dredze (2017)'s method of multi-task learning leverages the performance of domain adaptation. Clark et al. (2018)'s method utilizes multi-task learning of NER with several NLP tasks such as POS tagging and parsing. Crichton et al. (2017) and Wang et al. (2018) leverage the performance of NER by multi-task learning of several tasks of biomedical NLP.

## 5 Conclusions

We proposed a multi-task learning method on an NER model and a paraphrase model to utilize paraphrase pairs efficiently. The evaluations on the BioCreative IV's CHEMDNER task showed that our method achieved the best performance.

## References


Gustavo Aguilar, Adrian Pastor López Monroy, Fabio González, and Thamar Solorio. 2018. Modeling noisiness to recognize named entities using multi-task neural networks on social media. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1401–1412.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th COLING*, pages 1638–1649.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd ICLR*.

Pengfei Cao, Yubo Chen, Kang Liu, Jun Zhao, and Shengping Liu. 2018. Adversarial transfer learning for chinese named entity recognition with self-attention mechanism. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 182–192.

Kevin Clark, Minh-Thang Luong, Christopher D. Manning, and Quoc Le. 2018. Semi-supervised sequence modeling with cross-view training. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1914–1925. Association for Computational Linguistics.

Gamal Crichton, Sampo Pyysalo, Billy Chiu, and Anna Korhonen. 2017. A neural network multi-task learning approach to biomedical named entity recognition. *BMC Bioinformatics*, 18.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proceedings of the 53rd ACL and the 7th IJCNLP (Volume 1: Long Papers)*, pages 1723–1732.

Kazuma Hashimoto, caiming xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple nlp tasks. In *Proceedings of the 2017 EMNLP*, pages 1923–1933.

Martin Krallinger, Florian Leitner, Obdulia Rabal, Miguel Vazquez, Julen Oyarzabal, and Alfonso Valencia. 2015. Chemdner: The drugs and chemical names extraction challenge. *Journal of Cheminformatics*, 7(1):S1.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 NAACL-HLT*, pages 260–270.

Robert Leaman, Chih-Hsuan Wei, and Zhiyong Lu. 2015. tmchem: a high performance approach for chemical named entity recognition and normalization. *Journal of Cheminformatics*, 7(1):S3.

Hongfei Lin, Jian Wang, Ling Luo, Pei Yang, Zhihao Yang, Yin Zhang, and Lei Wang. 2018. An attention-based bilstm-crf approach to document-level chemical named entity recognition. *Bioinformatics*, pages 1381–1388.

Liyuan Liu, Jingbo Shang, Xiang Ren, Frank Xu, Huan Gui, Jian Peng, and Jiawei Han. 2018. Empower sequence labeling with task-aware neural language model.

Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. 2015. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *Proceedings of the 2015 NAACL-HLT*, pages 912–921.

Yanan Lu, Donghong Ji, Xiaoyuan Yao, Xiaomei Wei, and Xiaohui Liang. 2015. Chemdner system with mixed conditional random fields and multi-scale word clustering. *Journal of Cheminformatics*, 7(1):S4.

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 EMNLP*, pages 1412–1421.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th ACL*, pages 1064–1074.

Nanyun Peng and Mark Dredze. 2017. Multi-task domain adaptation for sequence tagging. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 91–100.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 NAACL-HLT*, pages 2227–2237.

Marek Rei. 2017. Semi-supervised multitask learning for sequence labeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2121–2130. Association for Computational Linguistics.

Tjong Kim Sang, Erik F., and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2003*, pages 142–147.

Ryohei Sasano and Sadao Kurohashi. 2008. Japanese named entity recognition using structural natural language processing. In *Proc of . IJCNLP'08*, pages 607–612.

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proc. of NAACL HLT'03*, pages 134–141.

Xuan Wang, Yu Zhang, Xiang Ren, Yuhao Zhang, Marinka Zitnik, Jingbo Shang, Curtis Langlotz, and Jiawei Han. 2018. Cross-type biomedical named entity recognition with deep multi-task learning. *Bioinformatics*, page bty869.

Eunji Yi, Gary Geunbae Lee, Yu Song, and Soo-Jun Park. 2004. Svm-based biological named entity recognition using minimum edit-distance feature boosted by virtual examples. In *IJCNLP'04*, pages 807–814.

# FewRel 2.0: Towards More Challenging Few-Shot Relation Classification

**Tianyu Gao**[1], **Xu Han**[1], **Hao Zhu**[1,2], **Zhiyuan Liu**[1*], **Peng Li**[3], **Maosong Sun**[1], **Jie Zhou**[3]
[1]Department of Computer Science and Technology, Tsinghua University, Beijing, China
[2]Carnegie Mellon University, Pittsburgh, PA, USA
[3]Pattern Recognition Center, WeChat AI, Tencent Inc, China
{gty16,hanxu17}@mails.tsinghua.edu.cn,zhuhao@cmu.edu
liuzy@tsinghua.edu.cn,patrickpli@tencent.com
sms@tsinghua.edu.cn,withtomzhou@tencent.com

## Abstract

We present FewRel 2.0, a more challenging task to investigate two aspects of few-shot relation classification models: (1) Can they adapt to a new domain with only a handful of instances? (2) Can they detect none-of-the-above (NOTA) relations? To construct FewRel 2.0, we build upon the FewRel dataset (Han et al., 2018) by adding a new test set in a quite different domain, and a NOTA relation choice. With the new dataset and extensive experimental analysis, we found (1) that the state-of-the-art few-shot relation classification models struggle on these two aspects, and (2) that the commonly-used techniques for domain adaptation and NOTA detection still cannot handle the two challenges well. Our research calls for more attention and further efforts to these two real-world issues. All details and resources about the dataset and baselines are released at https://github.com/thunlp/fewrel.

## 1 Introduction

Few-shot learning, which requires models to handle new classification tasks with only a handful of training instances, has drawn much attention in recent years (Ravi and Larochelle, 2017; Vinyals et al., 2016; Munkhdalai and Yu, 2017; Snell et al., 2017). To advance this field in NLP, Han et al. (2018) propose FewRel, a large-scale dataset to explore few-shot learning in relation classification. Many efforts (Gao et al., 2019; Soares et al., 2019) have been devoted to the new task and some of the methods even exceed human performance[1] on FewRel. Based on the dataset FewRel, we propose FewRel 2.0, a new task containing two real-world issues that FewRel ignores: (1) few-shot domain adaptation, and (2) few-shot none-of-the-above detection.

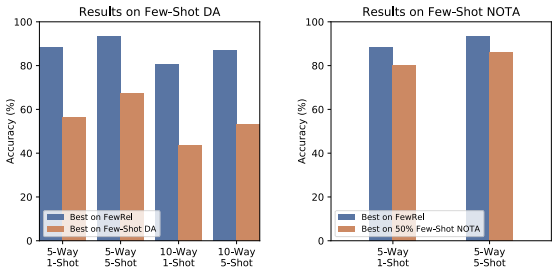


Figure 1: The comparison between the best results of the current models on FewRel, few-shot DA and few-shot NOTA. From the figures we can see that even the state-of-the-art models struggle on the new tasks.

**Few-shot domain adaptation** (few-shot DA) aims to evaluate the abilities of few-shot models to transfer across domains, which is crucial for real-world applications, since the test domains usually lack of annotations and could differ vastly from the training domains. To this end, we construct a new test set sharing great disparities with the original FewRel dataset, and carry out extensive experiments on the state-of-the-art few-shot models and commonly-used domain adaptation methods. Some prior experimental results in Figure 1 show that even the performance of the most effective methods on FewRel drops drastically on the new test set, proving that few-shot DA is challenging and requires further investigations.

**Few-shot none-of-the-above detection** (few-shot NOTA) is an advanced version of the existing $N$-way $K$-shot setting in few-shot learning. The original $N$-way $K$-shot setting samples $N$ classes, as well as $K$ supporting instances and several queries from each class for each test batch, assuming that all queries belong to the sampled $N$ classes. However, in few-shot NOTA, queries could also be none-of-the-above (NOTA), which brings one more option in classification and challenges existing few-shot methods. Consider-

* Corresponding author
[1]https://thunlp.github.io/fewrel.html



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6250–6255, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| Training Phase (Famous Person from Wikipedia) | | |
|---|---|---|
| Supp. Set | (A) date_of_birth | *Mark Twain* was born in *1835*. |
| | (B) place_of_birth | *Elvis Presley* was born in *Memphis, Tennessee*. |
| Query | (A) or (B) or **NOTA** | *William Shakespeare* passed away at age 52 (around *1616*). |
| **Test Phase (Biomedicine)** | | |
| Supp. Set | (A) may_treat | *Ribavirin* remains essential to *Chronic Hepatitis C* treatment. |
| | (B) manifestation_of | Boys with *Prader-Willi syndrome* often have *undescended testicle*. |
| Query | **(A)** or (B) or NOTA | *Thiabendazole* was effective in eradicating the *strongyloides infection*. |

Table 1: An example for a 2-way 1-shot scenario, including both few-shot DA and few-shot NOTA. Different colors indicate different entities, blue for head entities, and red for tail entities. For few-shot DA, instances in the training phase and test phase come from different domains. For few-shot NOTA, it requires models to detect the none-of-the-above (NOTA) relation.

ing few-shot NOTA has not yet been widely explored, we propose several solutions based on the state-of-the-art few-shot models and evaluate them with few-shot NOTA setting. Figure 1 shows that though achieving promising results, there is still a room of improvements for few-shot NOTA.

In the following sections, we first describe the two newly-added challenges in FewRel 2.0, then introduce possible directions for addressing these two issues, and finally present results and observations from our experiments.

## 2 FewRel 2.0

### Formulation for $N$-Way $K$-Shot Setting

The original FewRel task adopts the $N$-way $K$-shot setting. The whole dataset is divided into training, validation and test subsets, which have no intersection in relation types. Models are evaluated with batches sampled from the test set, each of which consists of $(\mathcal{R}, \mathcal{S}, x, r)$, where $\mathcal{R} = \{r_1, r_2, ..., r_N\}$ is the sampled relation set, $r \in \mathcal{R}$ is the correct relation label for the query $x$, and $\mathcal{S}$ is the supporting set containing $K$ instances for each relation,

$$\mathcal{S} = \{(x^j_{r_i}, r_i)\}, 1 \leq i \leq N, 1 \leq j \leq K. \quad (1)$$

Models should predict the relation label $y \in \mathcal{R}$ for the query instance $x$ based on the given $\mathcal{S}$ and $\mathcal{R}$. Both of the following two challenges are based on this $N$-way $K$-shot setting.

### Few-Shot Domain Adaptation

Both the training and test sets of the original FewRel dataset are constructed by manually annotating the distantly supervised (Bunescu and Mooney, 2007; Mintz et al., 2009) results on Wikipedia corpus and Wikidata (Vrandečić and Krötzsch, 2014) knowledge bases. In other words, they are from the same domain, yet in a real-world scenario, we might train models on one domain and perform few-shot learning on a different one. For example, we may train models on Wikipedia, which has large amounts of data and adequate annotations, and then perform few-shot learning on some domains suffering data sparsity, like literature, finance and medicine. Note that, not only do these corpora differ vastly from each other in morphology and syntax, but there are wide disparities between the relation sets defined on these domains as well, which makes transferring knowledge across different domains more challenging.

To explore few-shot DA, we construct a new test set by aligning PubMed [2], a database containing large amounts of biomedical literature, with UMLS [3], a large-scale knowledge base in the biomedical sciences. Then we let the annotators classify whether each instance we get from the distant supervision is correct. Every sentence is assigned to at least two annotators, and if their annotation results do not agree with each other, the third annotator is assigned. In the end, we gather a valid dataset with 25 relations and 100 instances for each relation.

For few-shot DA, we adopt the original FewRel training set for training, and the newly-annotated dataset for test, as shown in Table 1. Besides, we use SemEval-2010 task 8 dataset (Hendrickx et al., 2009) as the validation set, since both the corpora and the schema of SemEval-2010 task 8 are in different domains from the original FewRel dataset and the newly-annotated test set.

### Few-Shot None-of-the-Above Detection

In a $N$-way $K$-shot, all queries are assumed to be in the given relation set, yet sentences expressing no specific relations or relations not in the given set should also be taken into consideration, for they make up the vast majority of text. This calls for the none-of-the-above (NOTA) relation, which

[2] https://www.ncbi.nlm.nih.gov/pubmed/

[3] UMLS represents the Unified Medical Language System®, which is the trademark of U.S. National Library of Medicine.

indicates that the query instance does not express any of the given relations. Though it is common in some conventional classification tasks, where NOTA is usually regarded as an extra class, detecting NOTA could be hard in few-shot learning, because the given relation sets are not fixed so that the NOTA relation requires to cover a different semantic space each time. An example of NOTA is given in Table 1.

We formalize few-shot NOTA based on the $N$-way $K$-shot setting. For the query instance $x$, the correct relation label becomes $r \in \{r_1, r_2, ..., r_N, \texttt{NOTA}\}$ rather than $r \in \{r_1, r_2, ..., r_N\}$. We use the parameter NOTA rate to describe the proportion of NOTA queries during the whole test phase. For example, $0\%$ NOTA rate means no queries are NOTA and $50\%$ NOTA rate means half of the queries have the label NOTA.

The NOTA queries are sampled from those relations outside the given $N$ relations. To be more specific, denoting the whole test set as $\mathcal{D}_{\texttt{test}}$, the set containing all instances in the relation set $\mathcal{R}$ as $\mathcal{D}_{\mathcal{R}}$ and the NOTA rate as $\alpha$, $\alpha$ of the query instances (NOTA queries) are from $\mathcal{D}_{\texttt{test}} \setminus \mathcal{D}_{\mathcal{R}}$ and $1-\alpha$ of the instances are from $\mathcal{D}_{\mathcal{R}}$.

Note that during the test phase, all the queries are from the test set, though models can sample instances from the training set as supporting instances for NOTA relation (this method is described explicitly in Section 4). Also note that to better demonstrate the effects of the NOTA relation, we use the original FewRel dataset for few-shot NOTA, instead of the new test set, which can get rid of the influence of domain adaptation.

## 3 Approaches for Few-Shot DA

Many efforts have been devoted for domain adaptation, like subspace mapping (Pan et al., 2010; Fernando et al., 2013), finding domain-invariant spaces (Baktashmotlagh et al., 2013; Ganin et al., 2016), feature augmentation (Blitzer et al., 2006) and minimax estimators (Provost and Fawcett, 2001). Among them, adversarial training (Goodfellow et al., 2015; Ganin et al., 2016; Wang et al., 2018) has been proved to be efficient in finding domain-invariant features. It is a game process between an encoder and a discriminator, where the encoder tries to generate domain-invariant features while the discriminator tries to tell which domain the features are from.

Here we follow the adversarial training setting in Wang et al. (2018), where a two-layer perceptron network is used as the discriminator. While training the few-shot learning task, we feed the sentence encoder $\boldsymbol{E}$ and the discriminator $\boldsymbol{D}$ with the corpora from the training domain and the test domain, and optimize the min-max game,

$$\min_{\theta_{\boldsymbol{E}}} \max_{\theta_{\boldsymbol{D}}} \sum_{x \in \mathcal{C}_0} \log[\boldsymbol{D}(\boldsymbol{E}(x))]_0 + \sum_{x \in \mathcal{C}_1} \log[\boldsymbol{D}(\boldsymbol{E}(x))]_1, \tag{2}$$

where $[\cdot]_i$ is the $i$-th element of the vector, $\mathcal{C}_0$ is the training corpus and $\mathcal{C}_1$ is the test corpus.

## 4 Approaches for Few-Shot NOTA

A simple way to handle NOTA is to regard it as an extra class in the $N$-way $K$-shot setting. To be more specific, we can sample instances outside the $N$ relations as the supporting data of NOTA, and perform the $(N+1)$-way $K$-shot learning. As compared to the current methods ignoring NOTA, this approach does not bring much improvements, since the supporting data for NOTA actually belong to several different relations and are scattered in the feature space, making it hard to perform classification.

To better address few-shot NOTA, we propose a model named **BERT-PAIR** based on the sequence classification model in BERT (Devlin et al., 2019). We pair each query instance with all the supporting instances, concatenate each pair as one sequence, and send the concatenated sequence to the BERT sequence classification model to get the score of the two instances expressing the same relation. Denote the BERT model as $\boldsymbol{B}$, the query instance as $x$ and the paired supporting instance as $x_r^j$ (the $j$-th supporting instance for the relation $r$), $\boldsymbol{B}(x, x_r^j)$ outputs a two-element vector corresponding to scores of the pair sharing the same relation and not sharing the same relation. The probability over each relation in the few-shot scenario, including NOTA, is addressed as follows,

$$p(y = r|x) = \frac{\exp(o_r)}{\sum_{r' \in \mathcal{R}} \exp(o_{r'})}, r \in \mathcal{R} \tag{3}$$

where $y$ is the predicted label and $\mathcal{R} = \{r_1, ..., r_N, \texttt{NOTA}\}$ is the relation set including NOTA. For $r \in \{r_1, ..., r_N\}$, $o_r$ is calculated by averaging,

$$o_r = \frac{1}{K} \sum_{j=1}^{K} [\boldsymbol{B}(x, x_r^j)]_1. \tag{4}$$

| Model | 5-Way 1-Shot | | 5-Way 5-Shot | |
|---|---|---|---|---|
| | On 1.0 | On 2.0 | On 1.0 | On 2.0 |
| GNN (CNN) | $66.23 \pm 0.75$ | $27.94 \pm 0.03$ | $81.28 \pm 0.62$ | $29.33 \pm 0.11$ |
| Proto (CNN) | $74.52 \pm 0.07$ | $35.09 \pm 0.10$ | $88.40 \pm 0.06$ | $49.37 \pm 0.10$ |
| Proto-ADV (CNN) | $70.28 \pm 0.15$ | $42.21 \pm 0.09$ | $84.63 \pm 0.07$ | $58.71 \pm 0.06$ |
| Proto (BERT) | $80.68 \pm 0.28$ | $40.12 \pm 0.19$ | $89.60 \pm 0.09$ | $51.50 \pm 0.29$ |
| Proto-ADV (BERT) | $73.35 \pm 0.95$ | $41.90 \pm 0.44$ | $82.30 \pm 0.53$ | $54.74 \pm 0.22$ |
| BERT-PAIR | $88.32 \pm 0.64$ | $56.25 \pm 0.40$ | $93.22 \pm 0.13$ | $67.44 \pm 0.54$ |
| **Model** | **10-Way 1-Shot** | | **10-Way 5-Shot** | |
| | **On 1.0** | **On 2.0** | **On 1.0** | **On 2.0** |
| GNN (CNN) | $46.27 \pm 0.80$ | $16.44 \pm 0.04$ | $64.02 \pm 0.77$ | $18.26 \pm 0.03$ |
| Proto (CNN) | $62.38 \pm 0.06$ | $22.98 \pm 0.05$ | $80.45 \pm 0.08$ | $35.22 \pm 0.06$ |
| Proto-ADV (CNN) | $56.34 \pm 0.08$ | $28.91 \pm 0.10$ | $74.67 \pm 0.12$ | $44.35 \pm 0.09$ |
| Proto (BERT) | $71.48 \pm 0.15$ | $26.45 \pm 0.10$ | $82.89 \pm 0.11$ | $36.93 \pm 0.01$ |
| Proto-ADV (BERT) | $61.49 \pm 0.69$ | $27.36 \pm 0.50$ | $72.60 \pm 0.38$ | $37.40 \pm 0.36$ |
| BERT-PAIR | $80.63 \pm 0.17$ | $43.64 \pm 0.46$ | $87.02 \pm 0.12$ | $53.17 \pm 0.09$ |

Table 2: Accuracies (%) on few-shot DA. "On 1.0" represents the results on the original FewRel dataset and "On 2.0" represents the results on the new test set. The models with "-ADV" use adversarial training described in Section 3.

The score for NOTA $o_{\text{NOTA}}$ is calculated by the equation,

$$o_{\text{NOTA}} = \min_{r \in \{r_1, \ldots, r_N\}} \frac{1}{K} \sum_{j=1}^{K} [\boldsymbol{B}(x, x_r^j)]_0. \quad (5)$$

Then we can treat NOTA the same as other relations and optimize the model with the cross entropy loss, which is commonly-used in few-shot learning and other classification tasks.

## 5 Experiments

### 5.1 Baseline Models for Few-Shot Learning

We pick the two best models from the results in Han et al. (2018), **GNN** (Satorras and Estrach, 2018) and **Prototypical Networks** (Snell et al., 2017), as our baseline models. As for the encoders, besides the CNN encoder used in Han et al. (2018), we also adopt BERT since it achieves the state-of-the-arts in multiple tasks (Devlin et al., 2019). For all models and encoders, we follow the parameter settings from Han et al. (2018) and Devlin et al. (2019).

### 5.2 Evaluation Results on Few-Shot DA

Table 2 demonstrates the evaluation results of few-shot DA on the existing FewRel test set and the new test set. Besides the baselines, we also evaluate Prototypical Networks with adversarial training described in Section 3 and our proposed BERT-PAIR model in Section 4. We get three observations from the results:

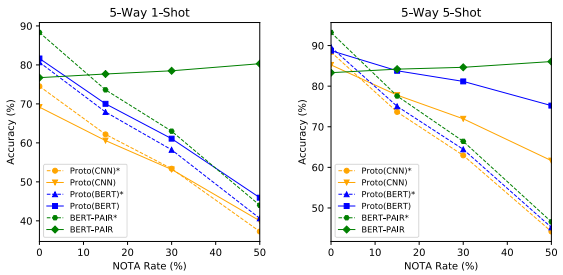


Figure 2: 5-way $K$-shot results under different NOTA rates. Models with * simply ignore the NOTA setting and assume all queries can be classified as one of the $N$ relations.

(1) All few-shot models suffer dramatic performance falls when tested on a different domain.

(2) Adversarial training does improve the results on the new test domain, yet still has large space for growth.

(3) BERT-PAIR outperforms all other few-shot models on both 1.0 and 2.0 test set.

Besides, to see where the growth boundary is, we split 10 relations, $1,000$ instances out of the 2.0 test set and add them to the training set, then train and evaluate BERT-PAIR on the new data. We get $72.30\%$ for 5-way 1-shot and $80.50\%$ for 5-way 5-shot, 16 and 13 points higher than the current best results. Note that only $1,000$ training instances can lead to such an enormous gap, indicating that

| Model | 5-Way-1-Shot | | | |
|---|---|---|---|---|
| | 0% NOTA | 15% NOTA | 30% NOTA | 50% NOTA |
| Proto (CNN)* | $74.52 \pm 0.07$ | $62.18 \pm 0.22$ | $53.38 \pm 0.14$ | $37.26 \pm 0.04$ |
| Proto (CNN) | $69.17 \pm 0.07$ | $60.59 \pm 0.05$ | $53.18 \pm 0.12$ | $40.00 \pm 0.10$ |
| Proto (BERT)* | $80.68 \pm 0.28$ | $67.92 \pm 0.31$ | $58.22 \pm 0.20$ | $40.64 \pm 0.14$ |
| Proto (BERT) | $81.65 \pm 0.97$ | $70.02 \pm 0.23$ | $61.08 \pm 0.28$ | $45.94 \pm 0.50$ |
| BERT-PAIR* | $88.32 \pm 0.64$ | $73.60 \pm 0.51$ | $63.00 \pm 0.47$ | $43.99 \pm 0.09$ |
| BERT-PAIR | $76.73 \pm 0.55$ | $77.67 \pm 0.14$ | $78.49 \pm 0.21$ | $80.31 \pm 0.12$ |
| **Model** | **5-Way-5-Shot** | | | |
| | **0% NOTA** | **15% NOTA** | **30% NOTA** | **50% NOTA** |
| Proto (CNN)* | $88.40 \pm 0.06$ | $73.64 \pm 0.11$ | $62.95 \pm 0.12$ | $44.20 \pm 0.05$ |
| Proto (CNN) | $85.23 \pm 0.07$ | $77.79 \pm 0.03$ | $71.96 \pm 0.14$ | $61.66 \pm 0.08$ |
| Proto (BERT)* | $89.60 \pm 0.09$ | $75.03 \pm 0.17$ | $64.44 \pm 0.18$ | $45.22 \pm 0.03$ |
| Proto (BERT) | $88.74 \pm 0.83$ | $83.79 \pm 0.44$ | $81.17 \pm 0.48$ | $75.21 \pm 0.52$ |
| BERT-PAIR* | $93.22 \pm 0.13$ | $77.58 \pm 0.42$ | $66.41 \pm 0.24$ | $46.58 \pm 0.09$ |
| BERT-PAIR | $83.32 \pm 0.38$ | $84.19 \pm 0.46$ | $84.64 \pm 0.13$ | $86.06 \pm 0.43$ |

Table 3: Accuracies (%) on few-shot NOTA. Models with * simply ignore the NOTA setting and assume all queries can be classified as one of the given relations.

there is still a huge room for improvements.

### 5.3 Evaluation Results on Few-Shot NOTA

We evaluate Prototypical Networks with the naive NOTA solution described in Section 4 and BERT-PAIR under the NOTA setting. All models are trained given $50\%$ NOTA queries and tested under four different NOTA rates: $0\%$, $15\%$, $30\%$, $50\%$. To show how accuracy falls if ignoring the NOTA relation, we also demonstrate the results of models without considering NOTA (marked with * in Figure 2). We demonstrate the evaluation results in Figure 2. For detailed numbers of results on few-shot NOTA, please refer to Table 3. From Figure 2 we can conclude that:

(1) Treating NOTA as the $N+1$ relation is beneficial for handling Few-Shot NOTA, though the results still fall fast when the NOTA rate increases.

(2) BERT-PAIR works better under the NOTA setting for its binary-classification style model, and stays stable with rising NOTA rate.

(3) Though BERT-PAIR achieves promising results, huge gaps still exist between the conventional ($0\%$ NOTA rate) and NOTA settings (gaps of $8$ points for 5-way 1-shot and 7 points for 5-way 5-shot with $50\%$ NOTA rate), which calls for further research to address the challenge.

## 6 Conclusion

In this paper, we propose FewRel 2.0, a more challenging few-shot relation classification task with a new test set from the biomedical domain and the none-of-the-above setting. The purpose of the new task is to explore two aspects which are ignored in the previous work: few-shot domain adaptation (few-shot DA) and few-shot none-of-the-above detection (few-shot NOTA). Extensive experiments demonstrate that the existing state-of-the-art few-shot models struggle on the new task. We also point out some possible directions to handle these two issues, implement several new models and evaluate them with the new task. Though achieving promising improvements, these commonly-used techniques are still not the satisfactory solutions for few-shot DA and few-shot NOTA, which requires further explorations in these two real-world challenges.

## Acknowledgments

This work is supported by the National Natural Science Foundation of China (NSFC No. 61572273, 61661146007) and Tsinghua University Initiative Scientific Research Program (20151080406). This work is also supported by the Pattern Recognition Center, WeChat AI, Tencent Inc. Han and Gao are supported by 2018 and 2019 Tencent Rhino-Bird Elite Training Program respectively. Gao is also supported by Tsinghua University Initiative Scientific Research Program. We also thank Xiaozhi Wang for his insightful ideas and suggestions.

## References


Mahsa Baktashmotlagh, Mehrtash T Harandi, Brian C Lovell, and Mathieu Salzmann. 2013. Unsupervised domain adaptation by domain invariant projection. In *Proceedings of ICCV*, pages 769–776.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of EMNLP*, pages 120–128.

Razvan Bunescu and Raymond Mooney. 2007. Learning to extract relations from the web using minimal supervision. In *Proceedings of ACL*, pages 576–583.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Basura Fernando, Amaury Habrard, Marc Sebban, and Tinne Tuytelaars. 2013. Unsupervised visual domain adaptation using subspace alignment. In *Proceedings of ICCV*, pages 2960–2967.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *JMLR*, 17(1):2096–2030.

Tianyu Gao, Xu Han, Zhiyuan Liu, and Maosong Sun. 2019. Hybrid attention-based prototypical networks for noisy few-shot relation classification. In *Proceedings of AAAI*, pages 6407–6414.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *ICLR*.

Xu Han, Hao Zhu, Pengfei Yu, Ziyun Wang, Yuan Yao, Zhiyuan Liu, and Maosong Sun. 2018. Fewrel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. In *Proceedings of EMNLP*, pages 4803–4809.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2009. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of SEW-2009*, pages 94–99.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of ACL-IJCNLP*, pages 1003–1011.

Tsendsuren Munkhdalai and Hong Yu. 2017. Meta networks. In *Proceedings of ICML*, pages 2554–2563.

Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. 2010. Domain adaptation via transfer component analysis. *TNNLS*, 22(2):199–210.

Foster Provost and Tom Fawcett. 2001. Robust classification for imprecise environments. *ML*, 42(3):203–231.

Sachin Ravi and Hugo Larochelle. 2017. Optimization as a model for few-shot learning. In *Proceedings of ICLR*.

Victor Garcia Satorras and Joan Bruna Estrach. 2018. Few-shot learning with graph neural networks. In *Proceedings of ICLR*.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *Proceedings of NIPS*, pages 4077–4087.

Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. 2019. Matching the blanks: Distributional similarity for relation learning. In *Proceedings of ACL*, pages 2895–2905.

Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. In *Proceedings of NIPS*, pages 3630–3638.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Xiaozhi Wang, Xu Han, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2018. Adversarial multi-lingual neural relation extraction. In *Proceedings of COLING*, pages 1156–1166.

# ner and pos when nothing is capitalized

**Stephen Mayhew, Tatiana Tsygankova, Dan Roth**
University of Pennsylvania
Philadelphia, PA, 19104
{mayhew, ttasya, danroth}@seas.upenn.edu

## Abstract

For those languages which use it, capitalization is an important signal for the fundamental NLP tasks of Named Entity Recognition (NER) and Part of Speech (POS) tagging. In fact, it is such a strong signal that model performance on these tasks drops sharply in common lowercased scenarios, such as noisy web text or machine translation outputs. In this work, we perform a systematic analysis of solutions to this problem in English, modifying only the casing of the train or test data using lowercasing and truecasing methods. While prior work and first impressions might suggest training a caseless model, or using a truecaser at test time, we show that the most effective strategy is a concatenation of cased and lowercased training data, producing a single model with high performance on both cased and uncased text. As shown in our experiments, this result holds across tasks and input representations. Finally, we show that our proposed solution gives an 8% F1 improvement in mention detection on noisy out-of-domain Twitter data.

## 1 Introduction

Many languages use capitalization in text, often to indicate named entities. For tasks that are concerned with named entities, such as *named entity recognition* (NER) and *part of speech tagging* (POS), this is an important signal, and models for these tasks nearly always retain it in training.[1]

But capitalization is not always available. For example, informal user-generated texts can have inconsistent capitalization, and similarly the outputs of speech recognition or machine translation are traditionally without case. Ideally we would like a model to perform equally well on both cased and uncased text, in contrast with current models.

[1]For POS tagging, this happens in tagsets that explicitly mark proper nouns, such as the Penn Treebank tagset.

| | | Test | |
|---|---|---|---|
| Tool | Task | Cased | Uncased |
| BiLSTM-CRF w/ ELMo | NER | 92.45 | 34.46 |
| BiLSTM-CRF w/ ELMo | POS | 97.85 | 88.66 |

Table 1: Modern tools trained on cased data perform well on cased test data, but poorly on uncased (lowercased) test data. For NER, we evaluate on the testb set of CoNLL 2003, and the scores are reported as F1. For POS, we evaluate on PTB sections 22-24, and the scores represent accuracy. ELMo refers to contextual representations from Peters et al. (2018).

Table 1 demonstrates how popular modern systems trained on cased data perform well on cased data, but suffer dramatic performance drops when evaluated on lowercased text.

Prior solutions have included models trained on lowercase text, or models that automatically recover capitalization from lowercase text, known as *truecasing*. There has a been a substantial body of literature on the effect of truecasing applied after speech recognition (Gravano et al., 2009), machine translation (Wang et al., 2006), or social media (Nebhi et al., 2015). A few works that evaluate on downstream tasks (including NER and POS) show that truecasing improves performance, but they do not demonstrate that truecasing is *the best way* to improve performance.

In this paper, we evaluate two foundational NLP tasks, NER and POS, on cased text and lowercased text, with the goal of maximizing the average score regardless of casing. To achieve this goal, we explore a number of simple options that consist of modifying the casing of the train or test data. Ultimately we propose a simple preprocessing method for training data that results in a single model with high performance on both cased and uncased datasets.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6256–6261,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

## 2 Related Work

This problem of robustness in casing has been studied in the context of NER and truecasing.

**Robustness in NER** A practical, common solution to this problem is summarized by the Stanford CoreNLP system (Manning et al., 2014): train on uncased text, or use a truecaser on test data.[2] We include these suggested solutions in our analysis below.

In one of the few works that address this problem directly, Chieu and Ng (2002) describe a method similar to co-training for training an upper case NER, in which the predictions of a cased system are used to adjudicate and improve those of an uncased system. One difference from ours is that we are interested in having a single model that works on upper or lowercased text. When tagging text in the wild, one cannot know *a priori* if it is consistently cased or not.

**Truecasing** Truecasing presents a natural solution for situations with noisy or uncertain text capitalization. It has been studied in the context of many fields, including speech recognition (Brown and Coden, 2001; Gravano et al., 2009), and machine translation (Wang et al., 2006), as the outputs of these tasks are traditionally lowercased.

Lita et al. (2003) proposed a statistical, word-level, language-modeling based method for truecasing, and experimented on several downstream tasks, including NER. Nebhi et al. (2015) examine truecasing in tweets using a language model method and evaluate on both NER and POS.

More recently, a neural model for truecasing has been proposed by Susanto et al. (2016), in which each character is associated with a label *U* or *L*, for upper and lower case respectively. This neural character-based method outperforms word-level language model-based prior work.

## 3 Truecasing Experiments

We use our own implementation of the neural method described in Susanto et al. (2016) as the truecaser used in our experiments.[3] Briefly, each sentence is split into characters (including spaces) and modeled with a 2-layer bidirectional LSTM, with a linear binary classification layer on top.

We train the truecaser on a dataset from Wikipedia, originally created for text simplification (Coster and Kauchak, 2011), but commonly used for evaluation in truecasing papers (Susanto et al., 2016). This task has the convenient property that if the data is well-formed, then supervision is free. We evaluate this truecaser on several data sets, measuring F1 on the word level (see Table 2). At test time, all text is lowercased, and case labels are predicted.

| System | Test set | F1 |
|---|---|---|
| (Susanto et al., 2016) | Wikipedia | 93.19 |
| BiLSTM | Wikipedia | 93.01 |
| | CoNLL Train | 78.85 |
| | CoNLL Test | 77.35 |
| | PTB 01-18 | 86.91 |
| | PTB 22-24 | 86.22 |

Table 2: Truecaser word-level performance on English data. This truecaser is trained on the Wikipedia corpus. *Wikipedia* refers to the test set from Coster and Kauchak (2011). *CoNLL Test* refers to *testb*. *PTB* is the Penn Treebank.

First, we evaluate the truecaser on the same test set as Susanto et al. (2016) in order to show that our implementation is near to the original. Next, we measure truecasing performance on plain text extracted from the CoNLL 2003 English (Tjong Kim Sang and De Meulder, 2003) and Penn Treebank (Marcus et al., 1993) train and test sets. These results contain two types of errors: idiosyncratic casing in the gold data and failures of the truecaser. However, from the high scores in the Wikipedia experiment, we suppose that much of the score drop comes from idiosyncratic casing. This point is important: if a dataset contains idiosyncratic casing, then it is likely that NER or POS models have fit to that casing (especially with these two wildly popular datasets). As a result, truecasing, since it can't recover these idiosyncrasies, is not likely to be the best plan.

Notably, the scores on CoNLL are especially low, likely because of elements such as titles, bylines, and documents that contain league standings and other sports results written in uppercase.

The higher scores on Penn Treebank corpus suggest that the capitalization standards are more traditional. Many errors are where the truecaser fails to correctly capitalize such words as "Federal" or "Central". In addition, there are many

[2] https://stanfordnlp.github.io/CoreNLP/caseless.html

[3] cogcomp.org/page/publication_view/881

occasions where the truecaser fails to capitalize named entities, for example "Mr. susulu".

## 4 Methods

In this section, we introduce our proposed solutions. In all experiments, we constrain ourselves to only change the casing of the training or testing data with no changes to the architectures of the models in question. This isolates the importance of dealing with casing, and makes our observations applicable to situations where modifying the model is not feasible, but retraining is possible.

Our experiments aim to answer the extremely common situation in which capitalization is noisy or inconsistent (as with inputs from the internet). In light of this goal, we evaluate each experiment on both cased and lowercased test data, reporting individual scores as well as the average. Our experiments on lowercase text can also give insight on best practices for when test data is known to be all lowercased (as with the outputs of some upstream system).

We experiment on five different data casing scenarios described below.

1. **Train on cased** Simply apply a model trained on cased data to unmodified test data, as in Table 1.

2. **Train on uncased** Lowercase the training data and retrain. At test time, we lowercase all test data. If we did not do this, then scores on the cased test set would suffer because of casing mismatch between train and test. Since lowercasing costs nothing, we can improve average scores this way. As such, cased and uncased test data will have the same score.

3. **Train on cased+uncased** Concatenate original cased and lowercased training data and retrain a model. Test data is unmodified.

   Since this concatenation results in twice the number of training examples than other methods, we also experimented with randomly lowercasing 50% of the sentences in the original training corpus. We refer to this experiment as *3.5 Half Mixed*. We also tried ratios of 40% and 60%, but these were slightly worse than 50% in our evaluations.

4. **Train on cased, test on truecased** Do nothing to the train data, but truecase the test data. Since we lowercase text before truecasing it, the cased and uncased test data will have the same score.

5. **Truecase train and test** Truecase the train data and retrain. Truecase the test data also. As in experiment 4, cased and uncased test data will have the same score.

One way to look at these experiments is as dropout for capitalization, where a sentence is lowercased with respect to the original with probability $p \in [0, 1]$. In experiment 1, $p = 0$. In experiment 2, $p = 1$. In experiment 3, $p = 0.5$. Our implementation is somewhat different from standard dropout in that our method is a preprocessing step, not done randomly at each epoch.

## 5 Experiments

Before we show results, we will describe our experimental setup. We emphasize that our goal is to experiment with strong models in noisy settings, not to obtain state-of-the-art scores on any dataset.

### 5.1 NER

We use the standard BiLSTM-CRF architecture for NER (Ma and Hovy, 2016), using an AllenNLP implementation (Gardner et al., 2018).

We experiment with pre-trained contextual embeddings, ELMo (Peters et al., 2018), which are generated for each word in a sentence, and concatenated with GloVe word vectors (lowercased) (Pennington et al., 2014), and character embeddings. ELMo embeddings are trained with cased inputs, meaning that there will be some mismatch when generating embeddings for uncased text.

In all experiments, we train on English CoNLL 2003 Train data (Tjong Kim Sang and De Meulder, 2003) and evaluate on the CoNLL 2003 Test data (testb). We always evaluate on two different versions: the original version, and a version with all casing removed (e.g. everything lowercase).

### 5.2 POS Tagging

We use a neural POS tagging model built with a BiLSTM-CRF (Ma and Hovy, 2016), and GloVe embeddings (Pennington et al., 2014), character embeddings, and ELMo pre-trained contextual embeddings (Peters et al., 2018).

As our experimental data, we use the Penn Treebank (Marcus et al., 1993), and follow the training splits of (Ling et al., 2015), namely 01-18 for train,

| Exp. | Test (C) | Test (U) | Avg |
|---|---|---|---|
| 1. Cased | **92.45** | 34.46 | 63.46 |
| 2. Uncased | 89.32 | 89.32 | 89.32 |
| 3. C+U | 91.67 | 89.31 | **90.49** |
| 3.5. Half Mixed | 91.68 | 89.05 | 90.37 |
| 4. Truecase Test | 82.93 | 82.93 | 82.93 |
| 5. Truecase All | 90.25 | **90.25** | 90.25 |

Table 3: Results from NER+ELMo experiments, tested on CoNLL 2003 English test set. *C* and *U* are Cased and Uncased respectively. All scores are F1.

| Exp. | Test (C) | Test (U) | Avg |
|---|---|---|---|
| 1. Cased | **97.85** | 88.66 | 93.26 |
| 2. Uncased | 97.45 | **97.45** | 97.45 |
| 3. C+U | 97.79 | 97.35 | 97.57 |
| 3.5. Half Mixed | **97.85** | 97.36 | **97.61** |
| 4. Truecase Test | 95.21 | 95.21 | 95.21 |
| 5. Truecase All | 97.38 | 97.38 | 97.38 |

Table 4: Results from POS+ELMo experiments, tested on WSJ 22-24, from PTB. *C* and *U* are Cased and Uncased respectively. All scores are accuracies.

19-21 for validation, 22-24 for testing. As with NER, we evaluate on both original and lowercased versions of test.

## 6 Results

Results for NER are shown in Table 3, and results for POS are shown in Table 4. There are several interesting observations to be made.

Primarily, our experiments show that the approach with the most promising results was experiment 3: training on the concatenation of original and lowercased data. Lest one might think this is because of the double-size training corpus, results from experiment 3.5 are either in second place (for NER) or slightly ahead (for POS).

Conversely, we show that the folk-wisdom approach of truecasing the test data (experiment 4) does not perform well. The underwhelming performance can be explained by the mismatch in casing standards as seen in Section 3. However, experiment 5 shows that if the training data is also truecased, then the performance is good, especially in situations where the test data is known to contain no case information.

Training only on uncased data gives good performance in both NER and POS – in fact the highest performance on uncased text in POS – but never reaches the overall average scores from experiment 3 or 3.5.

We have repeated these experiments for NER in several different settings, including using only static embeddings, using a non-neural truecaser, and using BERT uncased embeddings (Devlin et al., 2019). While the relative performance of the experiments varied, the conclusion was the same: training on cased and uncased data produces the best results.

When using uncased BERT embeddings, we found that performance on the uncased test set (U) was typically higher than that of ElMo, while the maximum performance on the cased test set (C) was typically lower. This again exemplifies the challenge of using capitalization as a signal while being robust to its absence.

## 7 Application: Improving NER Performance on Twitter

To further test our results, we look at the Broad Twitter Corpus[4] (Derczynski et al., 2016), a dataset comprised of tweets gathered from a broad variety of genres, and including many noisy and informal examples. Since we are testing the robustness of our approach, we use a model trained on CoNLL 2003 data. Naturally, in any cross-domain experiment, one will obtain higher scores by training on in-domain data. However, our goal is to show that our methods produce a more robust model on out-of-domain data, not to maximize performance on this test set. We use the recommended test split of section F, containing 3580 tweets of varying length and capitalization quality.

Since the train and test corpora are from different domains, we evaluate on the level of mention detection, in which all entity types are collapsed into one. The Broad Twitter Corpus has no annotations for MISC types, so before converting to a single generic type, we remove all MISC predictions from our model.

Results are shown in Table 5, and a familiar pattern emerges. Experiment 3 outperforms experiment 1 by 8 points F1, followed by experiment 3.5 and experiment 5, showing that our approach holds when evaluated on a real-world data set.

[4] https://github.com/GateNLP/broad_twitter_corpus

| Exp. | Mention Detection F1 |
|---|---|
| 1. Cased | 58.63 |
| 2. Uncased | 53.13 |
| 3. C+U | **66.14** |
| 3.5. Half Mixed | 64.69 |
| 4. Truecase Test | 58.22 |
| 5. Truecase All | 62.66 |

Table 5: Results on NER+ELMo on the Broad Twitter Corpus, set F, measured as mention detection F1.

## 8 Conclusion

We have performed a systematic analysis of the problem of unknown casing in test data for NER and POS models. We show that commonly-held suggestions (namely, lowercase train and test data, or truecase test data) are rarely the best. Rather, the most effective strategy is a concatenation of cased and lowercased training data. We have demonstrated this with experiments in both NER and POS, and have further shown that the results play out in real-world noisy data.

## 9 Acknowledgments

For their valuable feedback and suggestions, we would like to thank Jordan Kodner, Shyam Upadhyay, and Nitish Gupta.

This work was supported by Contracts HR0011-15-C-0113 and HR0011-18-2-0052 with the US Defense Advanced Research Projects Agency (DARPA). Approved for Public Release, Distribution Unlimited. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

## References

Eric W Brown and Anni R Coden. 2001. Capitalization recovery for text. In *Workshop on Information Retrieval Techniques for Speech Applications*, pages 11–22. Springer.

Hai Leong Chieu and Hwee Tou Ng. 2002. Teaching a weaker classifier: Named entity recognition on upper case text. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 481–488. Association for Computational Linguistics.

Will Coster and David Kauchak. 2011. Learning to simplify sentences using Wikipedia. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 1–9, Portland, Oregon. Association for Computational Linguistics.

Leon Derczynski, Kalina Bontcheva, and Ian Roberts. 2016. Broad twitter corpus: A diverse named entity recognition resource. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1169–1179, Osaka, Japan. The COLING 2016 Organizing Committee.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6, Melbourne, Australia. Association for Computational Linguistics.

Agustin Gravano, Martin Jansche, and Michiel Bacchiani. 2009. Restoring punctuation and capitalization in transcribed speech. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pages 4741–4744. IEEE.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramón Fermandez, Silvio Amir, Luís Marujo, and Tiago Luís. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1520–1530, Lisbon, Portugal. Association for Computational Linguistics.

Lucian Vlad Lita, Abe Ittycheriah, Salim Roukos, and Nanda Kambhatla. 2003. tRuEasIng. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 152–159. Association for Computational Linguistics.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Kamel Nebhi, Kalina Bontcheva, and Genevieve Gorrell. 2015. Restoring capitalization in# tweets. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1111–1115. ACM.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Raymond Hendy Susanto, Hai Leong Chieu, and Wei Lu. 2016. Learning to capitalize with character-level recurrent neural networks: An empirical study. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2090–2095, Austin, Texas. Association for Computational Linguistics.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*.

Wei Wang, Kevin Knight, and Daniel Marcu. 2006. Capitalizing machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 1–8, New York City, USA. Association for Computational Linguistics.

# CaRB: A Crowdsourced Benchmark for Open IE

**Sangnie Bhardwaj**[*†] Indian Institute of Technology New Delhi, India sangnie321@gmail.com

**Samarth Aggarwal**[*] Indian Institute of Technology New Delhi, India samarthaggarwal2510@gmail.com

**Mausam** Indian Institute of Technology New Delhi, India mausam@cse.iitd.ac.in

## Abstract

Open Information Extraction (Open IE) systems have been traditionally evaluated via manual annotation. Recently, an automated evaluator with a benchmark dataset (OIE2016) was released – it scores Open IE systems automatically by matching system predictions with predictions in the benchmark dataset (Stanovsky and Dagan, 2016). Unfortunately, our analysis reveals that its data is rather noisy, and the tuple matching in the evaluator has issues, making the results of automated comparisons less trustworthy.

We contribute CaRB, an improved dataset and framework for testing Open IE systems. To the best of our knowledge, CaRB is the first *crowdsourced* Open IE dataset and it also makes substantive changes in the matching code and metrics. NLP experts annotate CaRB's dataset to be more accurate than OIE2016. Moreover, we find that on one pair of Open IE systems, CaRB framework provides contradictory results to OIE2016. Human assessment verifies that CaRB's ranking of the two systems is the accurate ranking. We release the CaRB framework along with its crowdsourced dataset.

## 1 Introduction

Open Information Extraction (Open IE) refers to the task of forming relational tuples from sentences, without a fixed relation vocabulary (Banko et al., 2007). Open IE has numerous downstream applications such as knowledge base construction, relation extraction, summarisation and learning word embeddings (Stanovsky et al., 2015; Mausam, 2016). There have been many Open IE systems till date such as TextRunner (Banko et al., 2007), ReVerb (Fader et al., 2011; Etzioni et al., 2011), OLLIE (Mausam et al., 2012), ClausIE (Del Corro and Gemulla, 2013), OpenIE 4 (Christensen et al., 2011; Pal and Mausam, 2016), OpenIE 5 (Saha et al., 2017; Saha and Mausam, 2018), PropS (Stanovsky et al., 2016), NST (Jia et al., 2018), Neural Open IE (Cui et al., 2018), and more. With the advent of so many systems, it is imperative to have a standardized mechanism for automatic evaluation so that they can be compared.

Traditionally, these systems have been evaluated over small manually curated gold datasets (e.g., (Fader et al., 2011; Mausam et al., 2012)). There are two problems with this approach. One, it is not reliable due to the small size of annotation. Second, it lacks standardization, since there is no single gold dataset over which all systems are evaluated. Moreover, the guidelines to annotate may vary across datasets and annotators. Recently, some standard benchmarks datasets and evaluators have been proposed: OIE2016 (Stanovsky and Dagan, 2016), RelVis (Schneider et al., 2017), and Wire57 (Léchelle et al., 2018). Unfortunately, these datasets are either too small or too noisy to meaningfully compare Open IE systems.

For instance, since its release in 2016, OIE2016 has been considered the de facto standard for Open IE evaluation (e.g., OIE2016 is used by the recent NST and Neural Open IE systems). However, upon close analysis, we find several issues with this benchmark. Its gold dataset makes significant errors and misses a large number of important tuples. This can be attributed to the fact that this dataset was not manually curated for Open IE, rather QA-SRL data was adapted for this task. There are also issues with its evaluation rules, which we detail later.

In response, we propose a new benchmark system CaRB: **C**rowdsourced **a**utomatic open **R**elation extraction **B**enchmark, which has a good sized and high quality dataset, along with better

[*] Joint first author
† Presently an AI Resident at Google



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6262–6267, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| | |
|---|---|
| Sent. #1 | *Butters Drive in the Canberra suburb of Phillip is named in his honour .* |
| OIE2016 | ( in the Canberra suburb of Phillip is named in his honour . ; drive; ),<br>( Butters Drive in the Canberra suburb of Phillip ; named ; his honour ) |
| CaRB | ( Butters Drive in the Canberra suburb of Phillip ; is named ; in his honour ),<br>( Butters Drive ; is ; in the Canberra suburb of Phillip ) |
| Sent. # 2 | *It was only incidentally that economic issues appeared in nationalist political forms .* |
| OIE2016 | ( incidentally ; appeared ; economic issues ; nationalist political forms . ) |
| CaRB | ( economic issues ; appeared only incidentally in ; nationalist political forms ) |
| Sent. #3 | *The main reason for this adoption over mainline gimp was its support for high bit depths which can be required for film work .* |
| OIE2016 | ( high bit depths ; required ; film work ) |
| CaRB | ( this adoption ; has support for ; high bit depths ), ( high bit depths ; can be required for ; film work ),<br>( this adoption ; was over ; mainline gimp ), ( mainline gimp ; has no support for ; high bit depths ),<br>( its support for high bit depths which can be required for film work ; was The main reason for ;<br>this adoption over mainline gimp ) |
| Sent. #4 | *The number of ones equals the number of zeros plus one , since the state containing only zeros can not occur .* |
| OIE2016 | ( The number of ones ; equals ; the number of zeros plus one ; since the state containing only zeros<br>can not occur ), ( the state ; containing ; only zeros ), ( the state containing only zeros ; occur ) |
| CaRB | ( The number of ones ; equals ; the number of zeros plus one ),<br>( the state containing only zeros ; can not occur ) |

Table 1: Sample gold annotations for OIE2016 vs. CaRB

evaluation metrics. In order to create this gold dataset, we crowdsource human annotation of extractions using Amazon Mechanical Turk (MTurk) using the same original sentences as OIE2016. Our MTurk task has an automated system for training and qualifying workers, which makes crowdsourcing this annotation feasible.

Two Open IE experts (authors of this paper) manually annotate 50 random sentences, which are then used as expert ground truth to evaluate the respective tuples in OIE2016's and CaRB's gold datasets (Tables 4,5). We find that CaRB outperforms OIE2016 by 21 points in precision and 16 points in recall in token level match. This demonstrates that CaRB's gold dataset is significantly more accurate than OIE2016's. Additionally, when evaluating all systems using our benchmark, we notice that CaRB reverses OIE2016's ranking of PropS and ClausIE. Human verification, again through crowdsourcing, verifies that two systems are ranked more accurately by CaRB. We release CaRB's dataset, along with its evaluator as a novel benchmark for further use by research community.[1]

## 2 Related Work

To the best of our knowledge, there are three benchmarks systems available for comparing Open IE systems. Of them, the first and the most prominent is OIE2016 (Stanovsky and Dagan, 2016). This has been widely adopted as the standard evaluation framework to test new systems on. In OIE2016, gold tuples are generated using an automated rule-based system built on top of a QA-SRL dataset (He et al., 2015). In early analysis we find this dataset to be rather noisy. Table 1 illustrates some sample sentences from this gold dateset. These tuples look obviously wrong, and unfit to be in the gold set.

In addition to the dataset, Stanovsky and Dagan (2016) release a scorer that compares a set of gold tuples with a set of system tuples to estimate word-level precision and recall. This scorer has been identified to not penalize long extractions. It also does not penalise extractions for misidentifying parts of a relation in an argument slot (or vice versa), leading to trivial systems that score much better than genuine Open IE systems (Léchelle et al., 2018). We also observe that the scorer compares words all-to-all allowing multiple same words in an extraction to match a corresponding one in the gold. Thus, simply repeating a word in the extraction will give it a high precision score. Finally, the scorer loops over gold tuples in an arbitrary order, and matches them to predicted extractions in a sequential manner. Once a gold matches to a predicted extraction, it is rendered unavailable for any subsequent, potentially better-matched, extraction.

Another dataset is RelVis (Schneider et al., 2017), a benchmark that borrows its data from four different datasets including OIE2016. Since OIE2016 forms a major part of this dataset, it has similar issues with noise. Its scorer makes some

[1] https://github.com/dair-iitd/CaRB

modifications to OIE2016. However, it does not reward partial coverage of gold tuples, and forces one system prediction to match just one gold. It also does not penalize overlong extractions.

Finally, Wire57 (Léchelle et al., 2018) makes further improvements in the scorer. It penalises overlong extractions and assigns a token-level precision and recall score to all gold-prediction pairs for a sentence. Moreover, it considers all pairs of extractions in its matching phase. However, it still forces one prediction to match just one gold. It also reports just one score for a system, ignoring the confidence values of the individual predictions that make the precision-recall curve of OIE2016 possible. Our scorer is inspired by theirs, with some changes. More importantly, the dataset used in Wire57 is manually curated, but with only 57 sentences, which is too small to suffice as a comprehensive test dataset.

## 3 Crowdsourcing CaRB Dataset

To overcome the shortcomings of dataset noise and size, we crowdsource a high-quality gold dataset for Open IE. We ask workers over Amazon Mechanical Turk (MTurk) to annotate extractions for the 1,282 sentences in dev and test splits of OIE2016. The workers annotate tuples in the form (arg1, rel, arg2), and also annotate location and time attributes for each tuple, when possible.

Open IE annotations are not easy to obtain from non-expert workers. To get acceptable quality, we train workers using a tutorial[2] that doubles up as a qualification test. Their performance in the test is automatically graded. Only workers that pass this are allowed to move on to the main task. The qualification is integrated with the task so that a new worker is served the tutorial and test first, but a qualified worker is directly taken to the main task. This makes the crowdsourcing process scalable.

We divide the task of annotating a sentence into three steps: (1) identifying the relation, (2) identifying the arguments for that relation, and (3) optionally identifying the location and time attributes for the tuple. The training process for the annotators is split into four steps, each of which focuses on a different guideline for Open IE. These are:

1. **Completeness:** The worker must attempt to extract *all* assertions from the sentence.
2. **Assertedness:** Each tuple must be implied by the original sentence.

[2]Screenshots in supplementary material

| Sentence | *I ate an apple and an orange.* | (prec,rec) | |
|---|---|---|---|
| Gold | (I; ate; an apple)<br>(I; ate; an orange) | OIE2016 | CaRB |
| System 1 | (I; ate; an apple and an orange) | (1,0.5) | (0.57,1) |
| System 2 | (I; ate; an apple) | (1,0.5) | (1,0.87) |

Table 2: One-to-One Match vs. Multi Match

| Sentence | *I ate an apple.* | (prec,rec) | |
|---|---|---|---|
| Gold | (I; ate; an apple) | OIE2016 | CaRB |
| System 1 | (I; ate; an apple) | (1,1) | (1,1) |
| System 2 | (ate; an apple; I) | (1,1) | (0,0) |

Table 3: Tuple Match vs. Lexical Match

3. **Informativeness:** The worker must include the maximum amount of relevant information in an argument.
4. **Atomicity:** Each tuple must be an indivisible unit. Whenever possible, the worker must extract multiple atomic tuples from a sentence that has conjunctions.

We also develop a user-friendly interface for annotating the sentences, which almost eliminates the need for workers to type anything. However, we note that several workers got frustrated in our qualification test, could not understand the task and left the job. However, several good workers completed the task successfully, and annotated significant high-quality data for us.

For sentences involving reporting verbs like *said, told, asked,* etc., some systems annotate additional attributional context for every utterance (Mausam et al., 2012). For this, we create a separate task, so as to prevent workers from being bombarded with all the rules at the same time.

We post-process the data to remove obvious incorrect annotations, like ones with a missing arg1 or rel. We also follow the convention of ending a relation with a preposition instead of beginning arg2 with one, so all prepositions are shifted to rel.

## 4 The CaRB Scorer

We now describe CaRB's approach for scoring system predictions against the gold. Instead of greedily matching gold tuples to system tuples in arbitrary order, CaRB creates an all-pair matching table, with each column as system tuple and each row as gold tuple. It computes precision and recall scores between each pair of tuples. Then, for computing overall recall, the maximum recall score is taken in each row, and averaged. By taking the maximum, recall computation matches a

gold tuple with the closest system extraction. For computing precision, the system predictions are matched one to one with gold tuples, in the order of best match score to worst. The match precision scores are then averaged to compute precision. To compute precision-recall curve this computation is done at different confidence thresholds of system extractions.

In this way, CaRB's recall computation uses the notion of *multi-match*, wherein a gold tuple can match multiple system extractions. This is helpful in avoiding penalizing a system very heavily if it stuffs information from multiple gold tuples in a single extraction. Table 2 displays an example wherein system 1 combines information from two gold tuples in a single extraction, and system 2 only extracts one of the gold tuples. One-to-one match (OIE2016) is indifferent between the two which means that for OIE2016, adding more information in the same extraction has no value at all. However, multi match (CaRB) assigns higher recall to system 1, since it contains strictly more information, and higher precision to system 2, since its prediction exactly matched a gold extraction.

On the other hand, CaRB uses *single match* for precision. This is because CaRBs gold tuples are atomic, and cannot be further divided into more tuples. By single matching for precision, CaRB penalizes Open IE systems that produce several very similar and redundant extractions.

Another significant change from OIE2016 scorer is in the use of *tuple match* instead of *lexical match*. CaRB matches relation with relation, and arguments with arguments, however OIE2016 serialized the tuples into a sentence and just computed lexical matches. Table 3 illustrates an example when the arguments are shuffled, lexical match (OIE2016) shows no effect but tuple match (CaRB) rightfully decreases the scores. To avoid spurious matches, CaRB considers only matches with atleast one common word in the relation field.

Finally, some Open IE systems extract n-ary tuples and others do not. To treat all systems on equal footing, we follow previous work and append all higher numbered arguments into arg2.

## 5 Evaluation

### 5.1 Dataset Quality

We first estimate the overall quality of the crowdsourced dataset. To this end, two authors of this paper annotate 50 dev sentences from OIE2016 to create an expert dataset. They first independently annotate tuples from these sentences, achieving an agreement F1 score of 83. They then resolve the differences and merge these independent sets. This is taken as an expert gold against which both OIE2016 and CaRB datasets are assessed.

| Dataset | Precision | Recall | F1 |
|---|---|---|---|
| OIE2016 | 0.65 | 0.55 | 0.60 |
| CaRB | 0.87 | 0.71 | 0.78 |

Table 4: Data quality using token-level match

| | Precision | Recall | F1 |
|---|---|---|---|
| OIE2016 | 0.67 | 0.51 | 0.57 |
| CaRB | 0.74 | 0.73 | 0.73 |

Table 5: Data quality using lexical match

| System | Precision | Recall | F1 | AUC |
|---|---|---|---|---|
| Ollie | 0.505 | 0.346 | 0.411 | 0.224 |
| PropS | 0.340 | 0.300 | 0.319 | 0.126 |
| OpenIE 4 | **0.553** | 0.437 | **0.488** | **0.272** |
| OpenIE 5 | 0.521 | 0.424 | 0.467 | 0.245 |
| ClausIE | 0.411 | **0.496** | 0.450 | 0.224 |

Table 6: Performance of Open IE systems on CaRB

Tables 4 and 5 estimate dataset quality of OIE2016 and CaRB. We find that CaRB has enormously high precision and recall values, suggesting that it is a much cleaner dataset. Table 1 compares the crowd sourced annotations and OIE2016 gold annotations for some sample sentences. While there is still scope for improvement, CaRB dataset appears much better than the OIE2016's gold.

Stanovsky and Dagan (2016) remark that their gold dataset reaches an F1 of 95.8 on their expert annotation, whereas our assessment suggest values around 60. We surmise that this discrepancy is due to the different gold-prediction scoring schemes used. In original OIE2016 paper, the authors "match an automated extraction with a gold proposition if both agree on the grammatical head of all of their elements (predicate and arguments)".[3] The head match criterion is a much laxer scheme than ours and can explain the very high F1 score against their expert annotation.

[3]This scheme is later changed in their github repository to a lexical match, where if the fraction of words in the prediction also present in the gold is above a threshold, the pair is declared a match.

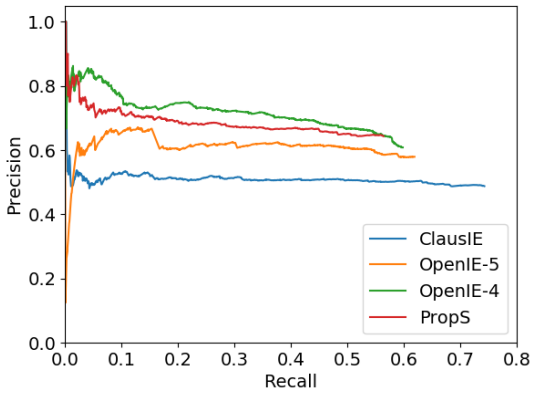


Figure 1: Comparison of Open IE systems using OIE2016

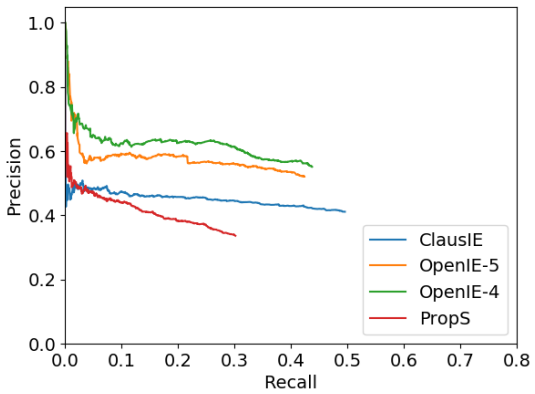


Figure 2: Evaluation of Open IE systems using CaRB

### 5.2 Comparison of Open IE Systems

We test the different Open IE systems depicted in Stanovsky and Dagan (2016), using the CaRB dataset and scorer. The p-r curves obtained using OIE2016 and CaRB are outlined in figures 1 (reproduced from Stanovsky et al. (2018)) and 2. Precision, recall and F1 scores (at max F1 point) and area under precision-recall curve are reported in Table 6. It can be seen that the curve for PropS lies above ClausIE at all times in OIE2016, but PropS performs the worse of all systems in CaRB. To verify that CaRB indeed gives the correct ranking, we turn back to human verification.

### 5.3 Human Verification

Through human verification, our goal is to learn the accurate ranking for ClausIE and PropS. We randomly select 100 test sentences and evaluate both system extractions on this subset.

We assess the correct ranking between PropS and ClausIE using MTurk. Four workers are shown the extractions from both systems in random order and asked to either choose one of the systems as the better one or indicate that both are equal. The majority opinion of these four is considered as the correct ranking for that sentence, an equal split leading to a tie. In this experiment, we only allow MTurk workers who have been trained for Open IE for the crowdsourcing task to participate.

Of these 100 sentences, PropS is chosen to have performed better for 15, ClausIE for 69 whereas 16 ended up in a tie. ClausIE is indeed considered the better system in human evaluation, and we verify that CaRB gives an accurate ranking of these two systems compared to OIE2016.

## 6 Conclusion

We contribute CaRB, a crowdsourced dataset for evaluation and comparison of Open IE systems. We assess this dataset against an expert-annotated dataset and find that it is dramatically more accurate than the existing OIE2016 benchmark dataset.

We also implement a scorer that computes precision, recall and area under p-r curve for a given system output by matching it with the CaRB dataset. In designing our scorer, we make several design choices that deviate from prior work in both match scores and also in finding the best match for a tuple. We believe our scheme treats various systems fairly. And in one case where CaRB and OIE2016 give different rankings to two Open IE systems, we demonstrate via human evaluation that the ranking given by CaRB is the accurate one. We release the dataset and scorer for further use by research community.

We expect that crowdsourced annotation will also be able to help the training of Open IE systems as it has helped their evaluation – we leave the creation of a suitably large crowdsourced training set for Open IE to future work.

## Acknowledgements

We thank Gabriel Stanovsky for helpful discussions and making his dataset and code available for research. We thank Prachi Jain and Keshav Kolluru for their comments on earlier versions of the paper. This work is supported by IBM AI Horizons Network grant, an IBM SUR award, grants by Google, Bloomberg and 1MG, and a Visvesvaraya faculty award by Govt. of India. We thank IIT Delhi HPC facility for compute resources.

## References


Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pages 2670–2676.

Janara Christensen, Mausam, Stephen Soderland, and Oren Etzioni. 2011. An analysis of open information extraction based on semantic role labeling. In *Proceedings of the 6th International Conference on Knowledge Capture (K-CAP 2011), June 26-29, 2011, Banff, Alberta, Canada*, pages 113–120.

Lei Cui, Furu Wei, and Ming Zhou. 2018. Neural open information extraction. *CoRR*, abs/1805.04270.

Luciano Del Corro and Rainer Gemulla. 2013. Clausie: Clause-based open information extraction. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 355–366, New York, NY, USA. ACM.

Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam. 2011. Open information extraction: The second generation. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume One*, IJCAI'11, pages 3–10. AAAI Press.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1535–1545, Stroudsburg, PA, USA. Association for Computational Linguistics.

Luheng He, Mike Lewis, and Luke Zettlemoyer. 2015. Question-answer driven semantic role labeling: Using natural language to annotate natural language. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 643–653.

Shengbin Jia, Yang Xiang, and Xiaojun Chen. 2018. Supervised neural models revitalize the open relation extraction. *CoRR*, abs/1809.09408.

William Léchelle, Fabrizio Gotti, and Philippe Langlais. 2018. Wire57 : A fine-grained benchmark for open information extraction. *CoRR*, abs/1809.08962.

Mausam. 2016. Open information extraction systems and downstream applications. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 4074–4077. AAAI Press.

Mausam, Michael Schmitz, Robert Bart, Stephen Soderland, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 523–534, Stroudsburg, PA, USA. Association for Computational Linguistics.

Harinder Pal and Mausam. 2016. Demonyms and compound relational nouns in nominal open IE. In *Proceedings of the 5th Workshop on Automated Knowledge Base Construction*, pages 35–39, San Diego, CA. Association for Computational Linguistics.

Swarnadeep Saha and Mausam. 2018. Open information extraction from conjunctive sentences. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 2288–2299.

Swarnadeep Saha, Harinder Pal, and Mausam. 2017. Bootstrapping for numerical open IE. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 317–323, Vancouver, Canada. Association for Computational Linguistics.

Rudolf Schneider, Tom Oberhauser, Tobias Klatt, Felix A. Gers, and Alexander Löser. 2017. Analysing errors of open information extraction systems. *CoRR*, abs/1707.07499.

Gabriel Stanovsky and Ido Dagan. 2016. Creating a large benchmark for open information extraction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, page (to appear), Austin, Texas. Association for Computational Linguistics.

Gabriel Stanovsky, Ido Dagan, and Mausam. 2015. Open IE as an intermediate structure for semantic tasks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*, pages 303–308.

Gabriel Stanovsky, Jessica Ficler, Ido Dagan, and Yoav Goldberg. 2016. Getting more out of syntax with props. *CoRR*, abs/1603.01648.

Gabriel Stanovsky, Julian Michael, Luke Zettlemoyer, and Ido Dagan. 2018. Supervised open information extraction. In *Proceedings of The 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT)*, page (to appear), New Orleans, Louisiana. Association for Computational Linguistics.

# Weakly Supervised Attention Networks for Entity Recognition

**Barun Patra***, **Joel Ruben Antony Moniz***
{barunpatra95, jramoniz}@gmail.com

## Abstract

The task of entity recognition has traditionally been modelled as a sequence labelling task. However, this usually requires a large amount of fine-grained data annotated at the token level, which in turn can be expensive and cumbersome to obtain. In this work, we aim to circumvent this requirement of word-level annotated data. To achieve this, we propose a novel architecture for entity recognition from a corpus containing weak binary presence/absence labels, which are relatively easier to obtain. We show that our proposed weakly supervised model, trained solely on a multi-label classification task, performs reasonably well on the task of entity recognition, despite not having access to any token-level ground truth data.

## 1 Introduction

Entity Recognition frequently finds use as a first step in numerous downstream NLP tasks (Wang and Xue, 2017; Lee et al., 2018; Liang et al., 2018). Traditionally, it has been posed as a sequence labeling task (Lample et al., 2016; Ma and Hovy, 2016), which in turn requires corpora with token-based annotations. A key drawback of this formulation, however, lies in its dependence on corpora annotated at the token-level, which can often be tedious to obtain and expensive to annotate.

One potential way of overcoming this limitation is to move towards a method that utilizes a weaker form of supervision that is easier to obtain. In this work, we focus on one such form of weak supervision: binary labels that indicate the presence of an entity type. The cognitive load of selecting whether an entity type is present or not is usually less than that of actually highlighting and annotating spans with their correct entity types. It also stands to reason that providing these binary labels might be faster. Both these properties are particularly advantageous for a human-in-the-loop setup in a user facing task, since a user is more likely to answer a yes/no question than to provide the annotated entity spans. This, in turn, facilitates cheaper and faster data collection; be it explicitly in the form of feedback questions, or implicitly from mined user logs (for example, clicked search engine results for queries related to "movies" are likely to contain the entity in question (Xu et al., 2009)).

In this work, we make first steps towards moving away from span-based corpora, relying solely on binary presence/absence classification labels for extracting entities. We propose a novel attention-based model that, though trained on a multi-label classification task, can be used for entity recognition. We show the efficacy of our proposed model on the widely used 2003 CoNLL dataset. Our model achieves reasonable performance without having access to token-level annotations. We thus show that it is possible to extract entities using a weak classification signal. [1]

## 2 Related Work

Commonly used methods for entity extraction rely on token-level annotated corpora. Conventionally, these supervised methods learn a CRF (Lafferty et al., 2001) or a Seq2Seq (Sutskever et al., 2014) model over either hand-crafter or neural features. More recently, pre-trained embeddings from language models trained on large corpora (Peters et al., 2018; Devlin et al., 2018), when augmented with previous methods, have shown marked improvements.

A contrasting line of work has been to explore unsupervised entity extraction without using

*Equal Contribution

[1] All our code can be found at https://github.com/joelmoniz/AttentionSegmentation



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6268–6273, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

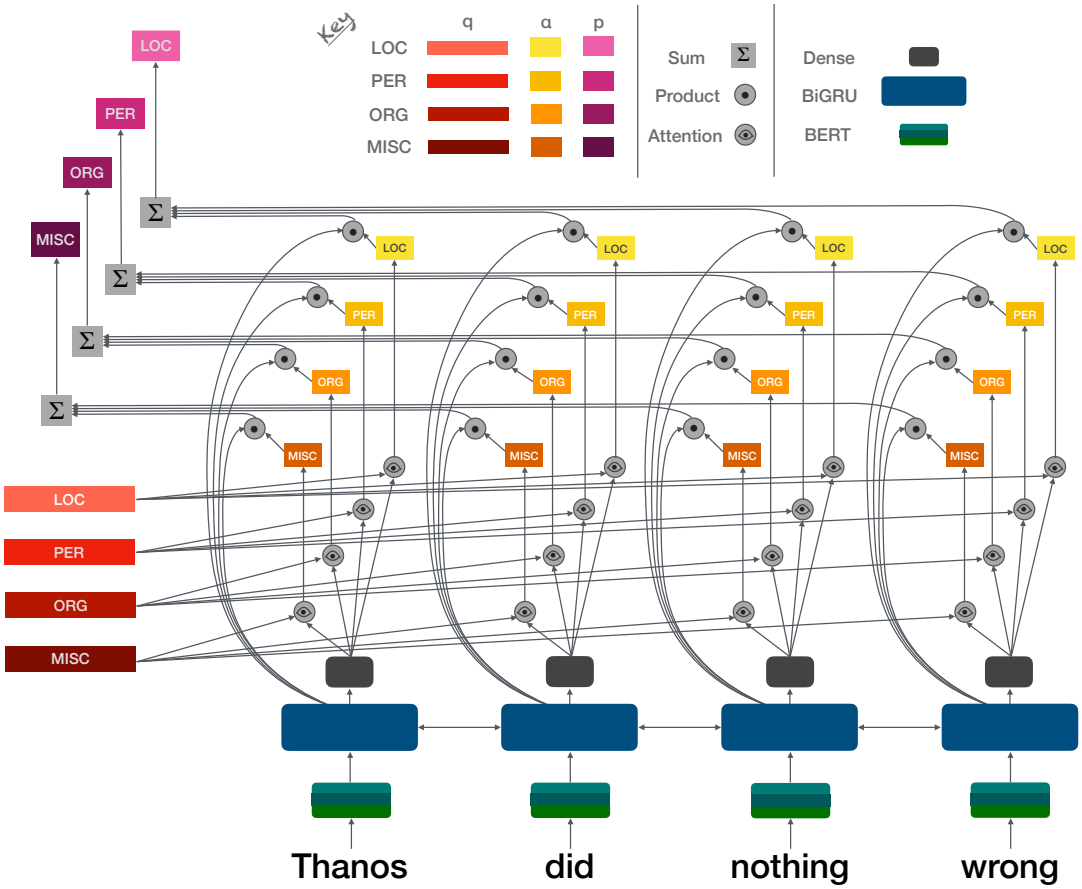


Figure 1: Model Architecture: Sentence-level Classifier. **q** represents the learned query vector per tag, $\alpha$ denotes the attention weights computed per token and **p** denotes the probability of predicting each tag at the sentence level.

ground-truth token or sentence level annotations. For example, a common paradigm for unsupervised Named Entity Recognition involves relying on a seed gazetteer, as in the case of Zhang and Elhadad (2013) and Ghiasvand and Kate (2015) both in the medical domain. In a more general setting, Carlson et al. (2009) use gazetteers to bootstrap training by labelling sequences that can be confidently annotated, and then use this partly labelled data to train their proposed Partial Perceptron algorithm; although they make use of a gazetteer, being the closest in setting to our proposed approach, we use Carlson et al. (2009) as our primary baseline. Another common technique involves bootstrapping the system with a set of rule templates, such as in (Etzioni et al., 2005) and (Collins and Singer, 1999). However, these methods often rely on an initial seed rule-base or on the availability of gazetteers (or the effective generation of these gazetteers using an online source such as Wikipedia).

Aside from directly improving performance on various tasks, attention (Bahdanau et al., 2014; Luong et al., 2015) has proven to be extremely useful when used indirectly in a wide variety of other ways (for example, for segmentation (Tang and Yang, 2018) and unsupervised speech-to-text alignment (Boito et al., 2017; Godard et al., 2018)). In addition, using attention-based models for object segmentation in a weakly supervised setting has been well explored in the vision domain (Teh et al., 2016; Zhang et al., 2018). Inspired by this, we leverage the attention weights of the model to identify entity spans.

## 3 Method

Figures 1 and 2 describe the different components of our model. Our model comprises of 4 modules: a sentence representation module, an attention module, a token-level tagger and a sentence-level classifier. Concretely, given a sentence $(x_1, \cdots x_l)$, the model predicts whether a tag $t \in \mathcal{T}$ is present in the sentence, as well as an attention distribution over the words for each tag $(\alpha_{1,t} \cdots \alpha_{l,t}) \forall t \in \mathcal{T}$ (where $\mathcal{T}$ is the set of entity tags), as described below:

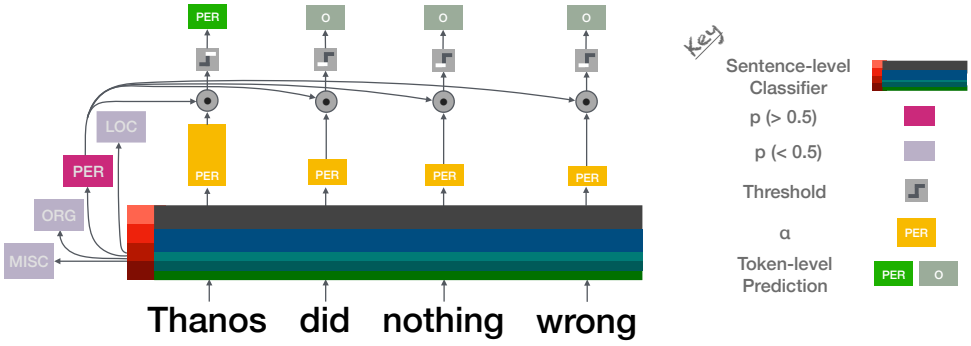


Figure 2: Model Architecture: Token Tagger. The Token Tagger performs tagging using the attention weights. $\alpha$ denotes the attention weights of the predicted tag(s).

**Token Representation Module:** generates an embedding for each token, representing the word's meaning and its left and right context. We use BERT embeddings (Devlin et al., 2018) for generating a word-level representation. We further use a Bidirectional GRU (Cho et al., 2014) layer to better adapt BERT embeddings to the task to obtain token representations $(e_1 \cdots e_l)$.

**Attention Module:** consists of an attention mechanism for each tag type. Given the token embeddings, a softmax distribution pertaining to the corresponding tag is generated, modelling the conditional probability of a word in a sentence being of that tag, given that the entire sentence contains the tag. We compute attention as in Bahdanau et al. (2014), using one learned query vector $q_t$ per tag $t \in \mathcal{T}$. The token embeddings $(e_1 \cdots e_l)$ are passed through a dense layer to generate keys $(k_1 \cdots k_l)$ in the query space, which together with query $q_t$, yield a set of attention weights $(\alpha_{1,t} \cdots \alpha_{l,t})$.

**Sentence-level Classifier:** generates the probability of the presence of a tag in the sentence. For each tag, the token-level representations $(e_1 \cdots e_l)$ are weighed by the attention distribution $(\alpha_{1,t} \cdots \alpha_{l,t})$ corresponding to the tag. The weighted sum generates a sentence representation $s_t$ per tag, which is passed through a sigmoid layer to generate the probability of the tag being present $(p_t)$, with $p_t > 0.5$ denoting the presence of a tag.

**Token Tagger Module:** combines the probabilities from the Sentence-level Classifier with the attention weights obtained from the Attention Module to generate BIO tags for each token. Only the attention weights pertaining to the predicted labels $\mathcal{T}'$ are considered (i.e., if no tag is predicted, the entire sentence is marked "O"), with the attention weights being scaled by the probability of the predicted label (i.e $p_t * (\alpha_{1,t} \cdots \alpha_{l,t})$). A word $x_i$ is assigned the label $y_i = \text{argmax}_{t \in \mathcal{T}'}(p_t * \alpha_{i,t})$ if $p_t * \alpha_{i,y_i}$ is greater than a small threshold $\epsilon$ and it is neither a punctuation symbol nor a stop-word (Figure 2).

## 4 Experiments

**Dataset:** To demonstrate the feasibility of our model, we adapt the commonly used CoNLL 2003 dataset (Ratinov and Roth, 2009). The dataset contains token-level annotations of the Reuter's Corpus (Lewis et al., 2004) for 4 entity types: person (PER), location (LOC), organization (ORG) and miscellaneous (MISC), with each token being tagged in the IOB format (Ramshaw and Marcus, 1999). For training and validation, we strip out the token-level annotations, instead annotating sentences to merely indicate the presence of entity-types. In order to quantify the quality of the extracted entities, we then measure the span level f-score on the test set using the gold entity spans.

**Modelling Choices and Hyperparameters:** We use the BERT-Base Multilingual Cased model for the Token Representation Module, similar to the NER tagging setup in Devlin et al. (2018). We experiment both with using the embeddings as is, and fine-tuning the top layers of the attention encoder (we only try fine-tuning the top, top-two and top-three layers due to computational constraints). For the Token Tagger module, we find that the attention probabilities usually demarcate the tagged words quite clearly, and that an $\epsilon$ value of 0.01 performs reasonably well. We use early stopping based on the (averaged) validation accuracy of the sentence-level predictions. We also observe that fine-tuning the BERT model requires learning rates comparable in order of magnitude to those used in Devlin et al. (2018), and hence use

a learning rate of 2e-7 for fine-tuning the transformer layers, and 1e-3 for the rest of the network. More details related to the hyperparameters used are presented in Appendix A. All our models have been implemented using the AllenNLP framework (Gardner et al., 2017).

## 5 Results and Analysis

| Type | Model | F-score | Acc |
|---|---|---|---|
| S | (Lample et al., 2016) | 90.9 | - |
| | (Peters et al., 2018) | 92.2 | - |
| | (Devlin et al., 2018) | 92.8 | - |
| US | (Carlson et al., 2009) | 55.3 | - |
| WS | BERT + PS | 65.8 | 95.5 |
| | BERT(FT) + PS | 68.4 | 95.5 |
| | BERT + GRU + PS | 79.2 | 95.5 |
| | BERT(FT) + GRU | 80.7 | 95.6 |
| | BERT(FT) + GRU + PS | 81.1 | 95.6 |

Table 1: Different model performances on the 2003 CoNLL test set. FT denotes fine-tuning, PS denotes probability scaling. S is supervised, US is unsupervised and WS is weakly supervised

Table 1 shows the performance of our model. We observe that our proposed approach significantly outperforms the baseline, and performs reasonably well when compared to various state-of-the-art supervised approaches that use significantly more ground-truth annotation information.

To further investigate the impact of the different components of the model, we ablate our model components (Table 1). We observe that having a contextual GRU layer for adapting to the task has a significant impact on the performance, with the final model performing much better than BERT + PS. Further, fine-tuning and probability scaling also help improve model performance.

### 5.1 Impact of Stop-Word Removal

We find that the model learns to focus on indicator words that frequently occur in sentences where a particular tag is present, and tends to use them to identify the presence of entities at the sentence level. For example, the word "at" is often indicative of the existence of a location in a sentence, the name of the location itself aside, and the model tends to focus on both. This behaviour is unsurprising, given that the model is trained purely using a signal of whether or not a sentence contains an entity of that type, with no idea about the entity boundaries themselves. Based on what we commonly observe, a few of these indicator words include: prepositions such as "at" and "in" when a LOC entity is present; common titles preceding PER names (such as "President" X or "Miss" Y); conjunctions that might separate two or more entities (such as X "and" Y).

This results in the model picking out spurious words alongside the actual entity, which necessitates the use of stop-word and symbol removal. However, this removal also results in the model not being able to pick out words when they occur within an entity span (for example, "Republic of Iceland"). This is particularly problematic for ORG (4.85% spans have stop-words) and MISC (3.15% spans have stop-words), compared to LOC(0.6%) and PER(0.76%). This issue can potentially be mitigated with either a more sophisticated tagger module or a better stop-word/symbol removal mechanism, which we leave to future work.

### 5.2 Errors due to Incorrect Entity Boundaries

| Model | Text | Type | Micro |
|---|---|---|---|
| (Peters et al., 2018) | 95.9 | 93.8 | 94.8 |
| BERT(FT) + GRU + PS | 84.0 | 89.0 | 86.5 |

Table 2: MUC f-scores for the model

Since our model does not have access to annotated training data, it has no direct supervision for learning entity boundaries. This particularly hurts the model in the CoNLL task, since precision, recall and f-score are measured based on an exact string match. In order to investigate this, we use the metric used in MUC events (Grishman and Sundheim, 1996; Chinchor, 1998), wherein a system is scored on two axes: finding the correct text and the correct type. A text is correct if the entity boundaries are correct, regardless of their type, while a type is correct if an entity is tagged with a correct type, regardless of the boundaries, as long as there is an overlap with the gold type. The final score is a micro average f-measure (see Nadeau and Sekine (2007) for more details).

Table 2 shows the performance of the model along the two axes, as well as the MUC score. We also report the same metrics for the supervised model proposed in Peters et al. (2018). We

| Idx | Sentence | Predictions | Gold |
|---|---|---|---|
| 0 | @0@ Neal Lancaster , Dave Barr ( Canada ) , Mike Sullivan , Willie | LOC PER | LOC PER |
| 1 | @0@. Fatima Yusuf ( Nigeria ) @0@.@0@ | LOC PER | LOC PER |
| 2 | Russian peacemaker Alexander Lebed said he and rebel military leader Aslan Maskhadov agreed after overnight talks to defer the decision on whether Chechnya should be independent until December @0@ , @0@ . | LOC MISC PER | LOC MISC PER |
| 3 | " The world community should not be indifferent to the fact that President Lukashenko , who leads this European state of @0@ million people , is trying to establish a dictatorship with his new constitution , " Sharetsky said . | MISC PER | MISC PER |
| 4 | " I would like to thank the Eagles organisation and the wonderful fans of Philadelphia for supporting me throughout my career , " Cunningham said . | LOC ORG PER | ORG PER |
| 5 | June @0@-@0@ v Hampshire ( three days ) | LOC | ORG |

Figure 3: Qualitative Analysis (Best viewed in color) PER, LOC, MISC, ORG. The Gold column shows the gold tags contained in the sentence while the Predictions column contains the tags predicted by the model. Tags in green are correctly predicted, while those in red are incorrect.

see that the model primarily loses out on the text based measure, and performs quite well on the type based one. This is in accordance with our hypothesis that the model identifies the correct entities, but fails at finding the exact entity boundaries. A better boundary detection method can consequently be used alongside our model to improve the entity retrievals in a downstream task.

### 5.3 Qualitative Analysis

Fig 3 shows examples of where our proposed model performs well, and where it fails. Examples 0, 1 and 2 show when the model is correctly able to identify the entity classes persons, locations and miscellaneous. Examples 3, 4 and 5 show some of the shortcomings. As described in 5.1, indicator tokens like President (Example 3) are usually marked as the PER entity type, which gets penalized when compared to the gold labels. Another common failure case we observe is when outside domain knowledge is necessary to disambiguate the entity type. For example, in Example 4, the model predicts Philadelphia as LOC, while the correct tag is ORG (referring to the Philadelphia Eagles). Similarly, in Example 5, the model classifies Hampshire as LOC, while the correct tag is ORG (the cricket club).

## 6 Conclusion

We present a novel method for entity recognition using a relatively weak supervision signal. Our proposed model, trained on a multi-label classification task, achieves reasonable entity recognition performance.

While our proposed method is simple, we demonstrate that it works surprisingly well. Various other formulations for this task are possible: for example, one might involve marginalizing over the tags and using this for predicting a label; another could perform attention over spans instead of tokens. We plan on investigating these alternate approaches in future work.


### Acknowledgments

We would like to thank the anonymous reviewers for their invaluable feedback, which helped shaped the paper into its current form. We would also like to thank Matthew R. Gormley for the helpful discussions on the topic.


## References


Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Marcely Zanon Boito, Alexandre Bérard, Aline Villavicencio, and Laurent Besacier. 2017. Unwritten languages demand attention too! word discovery with encoder-decoder models. In *Automatic Speech Recognition and Understanding Workshop (ASRU), 2017 IEEE*, pages 458–465. IEEE.

Andrew Carlson, Scott Gaffney, and Flavian Vasile. 2009. Learning a named entity tagger from gazetteers with the partial perceptron. In *AAAI Spring Symposium: Learning by Reading and Learning to Read*, pages 7–13.

Nancy Chinchor. 1998. Overview of muc-7. In *Seventh Message Understanding Conference (MUC-7): Proceedings of a Conference Held in Fairfax, Virginia, April 29-May 1, 1998*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Michael Collins and Yoram Singer. 1999. Unsupervised models for named entity classification. In *1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S Weld, and Alexander Yates. 2005. Unsupervised named-entity extraction from the web: An experimental study. *Artificial intelligence*, 165(1):91–134.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. Allennlp: A deep semantic natural language processing platform.

Omid Ghiasvand and Rohit J Kate. 2015. Biomedical named entity recognition with less supervision. In *Healthcare Informatics (ICHI), 2015 International Conference on*, pages 495–495. IEEE.

Pierre Godard, Marcely Zanon-Boito, Lucas Ondel, Alexandre Berard, François Yvon, Aline Villavicencio, and Laurent Besacier. 2018. Unsupervised word segmentation from speech with attention. *arXiv preprint arXiv:1806.06734*.

Ralph Grishman and Beth Sundheim. 1996. Message understanding conference-6: A brief history. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*, volume 1.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT*, pages 260–270.

Jihwan Lee, Dongchan Kim, Ruhi Sarikaya, and Young-Bum Kim. 2018. Coupled representation learning for domains, intents and slots in spoken language understanding. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 714–719. IEEE.

David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. 2004. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397.

Chen Liang, Mohammad Norouzi, Jonathan Berant, Quoc V Le, and Ni Lao. 2018. Memory augmented policy optimization for program synthesis and semantic parsing. In *Advances in Neural Information Processing Systems*, pages 9994–10006.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Lance A Ramshaw and Mitchell P Marcus. 1999. Text chunking using transformation-based learning. In *Natural language processing using very large corpora*, pages 157–176. Springer.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *CoNLL*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Zhiwen Tang and Grace Hui Yang. 2018. Deeptilebars: Visualizing term distribution for neural information retrieval. *arXiv preprint arXiv:1811.00606*.

Eu Wern Teh, Mrigank Rochan, and Yang Wang. 2016. Attention networks for weakly supervised object localization. In *BMVC*, pages 1–11.

Chuan Wang and Nianwen Xue. 2017. Getting the most out of amr parsing. In *Proceedings of the 2017 conference on empirical methods in natural language processing*, pages 1257–1268.

Gu Xu, Shuang-Hong Yang, and Hang Li. 2009. Named entity mining from click-through data using weakly supervised latent dirichlet allocation. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1365–1374. ACM.

Jianming Zhang, Sarah Adel Bargal, Zhe Lin, Jonathan Brandt, Xiaohui Shen, and Stan Sclaroff. 2018. Top-down neural attention by excitation backprop. *International Journal of Computer Vision*, 126(10):1084–1102.

Shaodian Zhang and Noémie Elhadad. 2013. Unsupervised biomedical named entity recognition: Experiments with clinical and biological texts. *Journal of biomedical informatics*, 46(6):1088–1098.

# Revealing and Predicting Online Persuasion Strategy with Elementary Units

**Gaku Morio** and **Ryo Egawa** and **Katsuhide Fujita**
Tokyo University of Agriculture and Technology
Koganei, Tokyo, Japan
gakumorio@gmail.com, egawa@katfuji.lab.tuat.ac.jp,
katfuji@cc.tuat.ac.jp

## Abstract

In online arguments, identifying how users construct their arguments to persuade others is important in order to understand a persuasive strategy directly. However, existing research lacks empirical investigations on highly semantic aspects of elementary units (EUs), such as propositions for a persuasive online argument. Therefore, this paper focuses on a pilot study, revealing a persuasion strategy using EUs. Our contributions are as follows: (1) annotating five types of EUs in a persuasive forum, the so-called *ChangeMyView*, (2) revealing both intuitive and non-intuitive strategic insights for the persuasion by analyzing 4612 annotated EUs, and (3) proposing baseline neural models that identify the EU boundary and type. Our observations imply that EUs definitively characterize online persuasion strategies.

The annotated dataset, annotation guideline, and implementation of the neural model are available in public.[1]

## 1 Introduction

Persuasive discourses are common in social media. Recently, the automatic identification of such persuasive discourse is receiving attention in the field of computational linguistics (Habernal and Gurevych, 2016b; Tan et al., 2016; Habernal and Gurevych, 2016a; Persing and Ng, 2017; Musi et al., 2018; Hidey and McKeown, 2018; Ji et al., 2018; Durmus and Cardie, 2019; Gleize et al., 2019). Existing studies have examined persuasive arguments with regard to dynamics or lexical features (Tan et al., 2016), claim and premise analysis (Musi and Aakhus, 2018), and concessions (Musi et al., 2018). However, to the best of our knowledge, fine-grained analyses with an **elementary unit (EU)** semantic, such as propositions, are still in their infancy, although we estimate that EUs are important in obtaining the persuasion strategy. Also, existing studies do not extract token-level EUs via machine learning, which would be a basic step for an automatic understanding of the strategy.

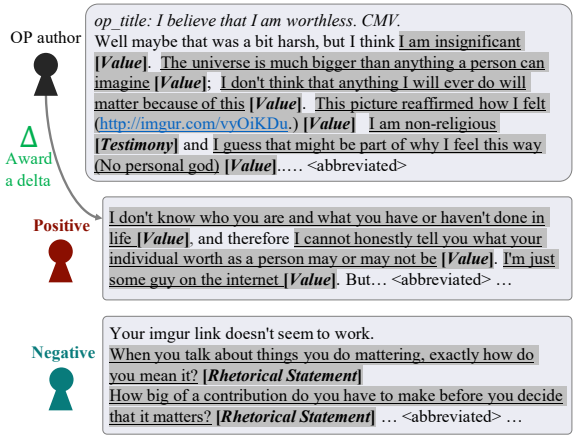


Figure 1: Overview of our EU annotation in *ChangeMyView*. We observed that Testimonies tend to appear in an original post (OP) (see Section 4.)

Therefore, this paper shows the fundamental role of EUs in a persuasive forum by annotating five types of token-level EUs (i.e., Testimony, Fact, Value, Policy, and Rhetorical Statement) in a persuasive forum and also provides a baseline neural model to identify the EUs automatically.[2]

In this study, we use the effective dataset of *ChangeMyView* (Tan et al., 2016), which is a subreddit where original post (OP) users provide a controversial view in the title to change their

[1]Dataset and annotation guideline: http://katfuji.lab.tuat.ac.jp/nlp_datasets/. Implementation: https://github.com/EdoFrank/emnlp2019_morio_egawa

[2]Although we use the partial data of the previous study (Egawa et al., 2019), this work is distinct from the previous one because we focus on the novel analysis of EUs and a novel task to predict EUs. See Section 4 and 5.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6274–6279,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| | source | #unit | level | IAA |
|---|---|---|---|---|
| **ours** | CMV | 4612 | token | .677 |
| Al Khatib et al. | editorial | 14313 | token | .56 |
| Hidey et al. | CMV | 2615 | token | .65 |
| Park et al. | eRulemaking | 4899 | clause | .648 |

Table 1: Corpus comparison related to our study.

perspective through opponent users' arguments. OP users award a delta ($\Delta$) point when their view is changed. Figure 1 presents an overview of *ChangeMyView* where the positive post is an awarded post and the negative post is a non-awarded one.

Our analyses to find out the role of EUs in *ChangeMyView* show both intuitive and non-intuitive phenomena (Section 4.) We also inject a structural feature into the neural model to improve the performance; the results show that EUs have characteristic positional roles (Section 5.)

## 2 Related Work

EUs have been considered in investigating the characteristics of argumentation in an argument mining discipline. Park et al. (2015); Park and Cardie (2018) deemed Testimony to be a type of objective evidence using personal state or experience. Al Khatib et al. (2016) proposed the argument model for analyzing the strategy of a news editorial. The model considers token-level discourse units of six different types. The study that is most closely related to our research is (Hidey et al., 2017) because it annotated semantic types of claims and premises in *ChangeMyView*; however, the authors did not take objective propositions such as facts into consideration.

Table 1 summarizes the comparison in terms of size, annotation granularity level, and inter-annotator agreement (IAA) of the related corpora. The table shows that our corpus is middle sized but has sufficient granularity and the reasonable IAA reliability.

## 3 Annotating EUs

### Typology of EUs

This study considers five types of EUs. For the introduction of the scheme, the motivation is derived from our expectation that we can feature persuasive arguments by considering personal experience, facts (Park and Cardie, 2018) and rhetoric (Blankenship and Craig, 2006). The five types of EUs are defined as follows:

Ⓣ: **Testimony** is an objective proposition related to the author's personal state or experience such as the following: *I do not have children.*

Ⓕ: **Fact** is a proposition describing objective facts that can be verified using objective evidence and therefore captures the evidential facts in persuasions: *Empire Theatres in Canada has a "Reel Babies" showing for certain movies.*

Ⓥ: **Value** is a proposition that refers to subjective value judgments without providing a statement on what should be done: *it is absolutely terrifying.*

Ⓟ: **Policy** offers a specific course of action to be taken or what should be done: *intelligent students should be able to see that.*

Ⓡ: **Rhetorical Statement** implicitly states the subjective value judgment by expressing figurative phrases, emotions, or rhetorical questions: *does it physically hurt men to be raped by women (as in PIV sex)?*

### Annotation Process

We extracted 115 *ChangeMyView* threads from the train set of Tan et al. (Tan et al., 2016) through a simple random sampling.[3] Each thread contains a triple of OP, positive, and negative. Therefore, we use 345 posts for the annotation.

EUs are annotated by 19 non-native annotators who are proficient in English, excluding the authors. Three annotators independently annotated 87 threads; the remaining 28 threads were annotated by 8 experts who were selected from the 19 annotators. Each annotator was asked to read the annotation guideline prior to the actual annotation. We also held several meetings to train the annotators and re-annotated the erroneous annotations when required. Given that the annotators are non-native speakers, the posts are translated into their language, and the translated documents are only used as a reference for the annotators. For the 83 threads, a gold standard is established by merging three posts using a majority vote.

We consider the token-level annotation (Stab and Gurevych, 2017), rather than the clause level one, to extract accurate minimal EU boundary and to remove irrelevant boundaries such as *for example* and *therefore*.

[3]More precisely, 419 threads were randomly extracted from the train data range 2013/01/01–2015/05/07, and the older threads were actually used due to time constraints.

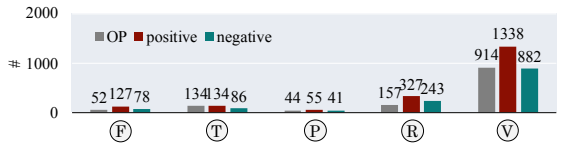


Figure 2: Distribution of the number of EUs. Most of the propositions in *ChangeMyView* are subjective.

| | Ⓕ | Ⓣ | Ⓟ | Ⓡ | Ⓥ |
|---|---|---|---|---|---|
| OP vs. reply | | ** | | ** | |
| positive vs. negative | | | | | |

Table 2: Significance of the proportion of EU frequency of *OP vs. reply* and *positive vs. negative*. Significant at $p < 0.01$ if **.

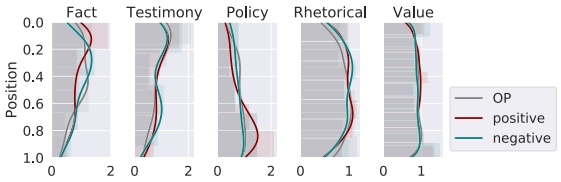


Figure 3: Kernel density estimate plots showing the distribution of EU positions in a post. Fact and Testimony tend to appear in the beginning of the post.

| position | MannWhitneyU | KS |
|---|---|---|
| 0.00 - 0.50 | **Ⓕ | *Ⓕ |
| 0.25 - 0.75 | | |
| 0.50 - 1.00 | | |

Table 3: Significance of the EU position between a positive and negative post. Significant at $p < 0.05$ if *.

### Annotation Results

As a result, 4612 EUs are annotated. Figure 2 shows the distribution of the number of EUs. We see that Value dominates in *ChangeMyView*, indicating most of the propositions are subjective.

At the end of the annotation, the IAA of Krippendorff's $\alpha_U$ (Krippendorff, 2004) was .677. Since sentence- or clause-level annotations (Park and Cardie, 2018) cannot accurately distinguish an inference step, we assume that the reasonable agreement is due to the token-level annotation.

## 4 Strategy Analysis of Persuasion: Revealing the Role of EUs

### Frequency of EUs Characterizes OP vs. Reply but Insignificant for Positive vs. Negative

Table 2 shows the significance of the proportion of each EU type for the post types, namely, *OP vs. reply (positive and negative)* and *positive vs. negative*. Given that the proportion is a real value, the Mann–Whitney U test is employed. A significant difference exists in Testimony in OP vs. reply and Rhetorical Statements in OP vs. reply. Therefore, OP authors are likely to assert their view with experiences, i.e., OP: *For many years, I've regularly skipped breakfast.* It is also intuitive that Rhetorical Statement exists to a high degree in persuasion (reply) posts because its strategy is for the persuasion.

In contrast, no significance is observed in a positive vs. negative case in Table 2. Although Habernal and Gurevych (2016b) stated that facts appear in persuasive comments, our result of Facts is insignificant as well as non-intuitive. We assume that this is because users are persuaded by effective indications of EUs, rather than simply presenting objective statements.

### Positional Role of EUs is Characteristic

To examine the positional role of EUs as a persuasion strategy, we investigate the position with a one-dimensional axis, that is, a normalized position wherein the beginning of a post is 0.0 and the end is 1.0. Figure 3 shows the resulting histogram of the positions for each EU type. It presents the characteristic distributions: Fact and Testimony are possibly located at the beginning of the post, e.g., positive: *Eating first thing in the morning stabilizes blood sugar levels, which regulates appetite and energy*, indicating the persuasion strategy to state solid facts prior to claiming assertion. Moreover, a Policy appeared at the bottom, indicating that what should be done is regarded as a conclusion, e.g., negative: *Just keep in mind that eating breakfast may increase your ability to concentrate, even if you think you are doing ok right now*. Value is the type that dominates in most posts, and it appears in uniform distribution.

The statistical tests were performed to determine the existence of a difference between a positive and negative in terms of the position. In general, given that addressing the entire positions simultaneously could simplify the property, we set the window size to 0.5 and investigated three ranges as follows: beginning $(0.00 - 0.50)$, middle $(0.25 - 0.75)$, and end $(0.50 - 1.00)$. For each window, we employed the Mann–Whitney U test and Kolmogorov-Smirnov (KS) significance for the average positions in threads which have at least five propositions, as described in Table 3.

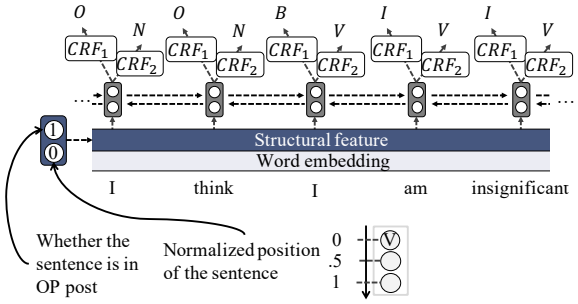


Figure 4: BiLSTM-CRF with structural features.

The table shows that the average position at the beginning of the post has significant differences in Fact.

## 5 Neural EU Extraction to Automatically Obtain the Strategy

In practice, we have to extract EUs computationally to obtain the strategy automatically. We consider the EU extraction problem as a sequence tagging problem in analogy using a related argument mining work (Eger et al., 2017; Habernal and Gurevych, 2017). The sequence tagging problem involves labeling each token in an input document. In each time step $i$, the label $y_i \in \mathcal{Y}$, where each token is associated with this label, is predicted as follows: $\mathcal{Y} = \{(b,t) \mid b \in \{B,I,O\}, t \in \{F,T,P,R,V,N\}\}$, where $b$ denotes the unit identification whether the current token is non-EU ($O$), begins ($B$), or continues ($I$) an EU. In addition, $t$ is the unit classification, i.e., Non-EU ($N$), Fact ($F$), Testimony ($T$), Policy ($P$), Rhetorical Statement ($R$), and Value ($V$).

### The Multi-task Learning Approach

We employed neural multi-task learning (MTL) models for the sequence tagging problem because several existing argument mining studies achieved advanced performance when the MTL models were used (Schulz et al., 2018). Figure 4 shows the overview of the proposed model . This model comprises two modules. *(i)* The bi-directional LSTM (BiLSTM) module encodes the input tokens in a given document. *(ii)* The module for conditional random fields (CRFs) for each subtask (unit identification and classification) is applied to execute label classification, denoted by the layers toward the top of Figure 4. We call our model BiLSTM-CRF (BLC).

### (i) Module of the BiLSTM

In our model, LSTMs were used to encode the input tokens. We employed the BiLSTM that uses two LSTMs to represent the forward and backward contexts in a sentence. Given an input token representation $x_i$ (i.e., a word embedding vector) at time $i$, we obtained the context-aware hidden state $h_i = LSTM_F(x_{1:i}) \circ LSTM_B(x_{n:i})$, where $n$ denotes the length of the input tokens.

### (ii) Module of the CRFs

This output module discriminates the BIO tag and EU type using the hidden representations generated by the BiLSTM. When a CRF layer was utilized, the past and future tags to predict the present tag are used efficiently. Therefore, the CRF layer is effective for solving a non-independent sequence tagging problem, including BIO tagging, i.e., the label after "$B$" is obviously "$I$." Note that we provided an MTL model that shares all the trainable parameters of the BiLSTM module but does not share any parameter of the CRF module for each task.

### Experimental Setting

The train and test threads were randomly divided per a ratio of $8:2$, where 30% of the train set is considered to be a development set. Herein, we set up two cases: *post-level*, a case in which an entire post is provided for the input and *sentence-level*, a case in which each sentence is given. The motivation is that the EUs cannot cross over a sentence. However, the sentence-level case would loss structural information. To remedy this problem, we focus on two structural features: whether or not the sentence is in OP post and the normalized position of the sentence in a post.

Each system was trained for 50 iterations with random seeds and the model that exhibited the optimal performance using the development set was selected (Schulz et al., 2018). The hyperparameters of our network are 100-dim GloVe embeddings (Pennington et al., 2014), the hidden layer size 200 for the BiLSTMs, the dropout rate of 0.5, and the optimizer is Adam (Kingma and Ba, 2014).

### Results: Overall Performance

Table 4 shows that our proposed sentence-level BLC with structural information performs best regarding macro F1 in either boundary identification or unit classification tasks. The presence of

| BLC variant | unit identification | | | | unit classification | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | B | I | O | macro | Ⓥ | Ⓡ | Ⓟ | Ⓣ | Ⓕ | macro |
| sentence-level + structural | 75.4 | 92.8 | 62.1 | ***76.8** | ***80.7** | **61.5** | ***16.0** | **43.1** | **33.2** | ***49.2** |
| sentence-level | **75.6** | 92.8 | 61.0 | 76.5 | 80.2 | 60.9 | 13.0 | 41.3 | 31.3 | 47.7 |
| post-level | 67.8 | **92.9** | ***64.7** | 75.1 | 79.9 | 49.4 | 2.1 | 35.3 | 32.2 | 43.8 |

Table 4: Performance of the average F1. Max score if **bold** and significant differences with $p < 0.05$ if *.

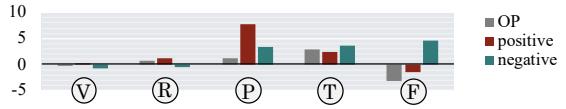


Figure 5: Performance improvement (F1 of a unit classification) by adding the structural features.

structural features provides an excellent boost in classifying unit types. For the post-level BLC, the model yields a better score in predicting non-EU parts because post-level discrimination can capture the entire post and thus identify irrelevant boundaries, such as supplementary notes, in posts.

**Results: Analysis of Structural Features**

Figure 5 shows the performance improvement by structural features in a distinct experiment from Table 4. In general, we see our structural features perform better for EUs that have characteristic positional distributions such as Policy and Testimony as shown in Figure 3. Interestingly, the structural features did not perform well in Fact classification for OPs and positive posts. The result implies Facts in the OPs and positive posts would be solid and can be predicted without structural information.

## 6 Conclusion

This paper demonstrated empirical work on the five types of elementary units (EUs) in online arguments. To find out the role of EUs, we annotated 4612 EUs and showed both intuitive and non-intuitive results. We also proposed baseline neural models to discriminate EUs. Experimental results showed that positional roles of some EUs are essential in detecting EUs. The annotated corpus and neural models will be applicable for future persuasion reasoning or evaluation of persuasiveness.

## Acknowledgments

This work was supported by JST CREST (JPMJCR15E1) and JST AIP-PRISM (JPMJCR18ZL), Japan.

## References

Khalid Al Khatib, Henning Wachsmuth, Johannes Kiesel, Matthias Hagen, and Benno Stein. 2016. A news editorial corpus for mining argumentation strategies. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3433–3443, Osaka, Japan. The COLING 2016 Organizing Committee.

Kevin L. Blankenship and Traci Y. Craig. 2006. Rhetorical question use and resistance to persuasion: An attitude strength analysis. *Journal of Language and Social Psychology*, 25(2):111–128.

Esin Durmus and Claire Cardie. 2019. A corpus for modeling user and language effects in argumentation on online debating. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 602–607, Florence, Italy. Association for Computational Linguistics.

Ryo Egawa, Gaku Morio, and Katsuhide Fujita. 2019. Annotating and analyzing semantic role of elementary units and relations in online persuasive arguments. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 422–428, Florence, Italy. Association for Computational Linguistics.

Steffen Eger, Johannes Daxenberger, and Iryna Gurevych. 2017. Neural end-to-end learning for computational argumentation mining. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11–22, Vancouver, Canada. Association for Computational Linguistics.

Martin Gleize, Eyal Shnarch, Leshem Choshen, Lena Dankin, Guy Moshkowich, Ranit Aharonov, and Noam Slonim. 2019. Are you convinced? choosing the more convincing evidence with a Siamese network. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 967–976, Florence, Italy. Association for Computational Linguistics.

Ivan Habernal and Iryna Gurevych. 2016a. What makes a convincing argument? empirical analysis and detecting attributes of convincingness in web argumentation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1214–1223, Austin, Texas. Association for Computational Linguistics.

Ivan Habernal and Iryna Gurevych. 2016b. Which argument is more convincing? analyzing and predicting convincingness of web arguments using bidirectional lstm. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1589–1599. Association for Computational Linguistics.

Ivan Habernal and Iryna Gurevych. 2017. Argumentation mining in user-generated web discourse. *Comput. Linguist.*, 43(1):125–179.

Christopher Hidey and Kathleen McKeown. 2018. Persuasive influence detection: The role of argument sequencing.

Christopher Hidey, Elena Musi, Alyssa Hwang, Smaranda Muresan, and Kathy McKeown. 2017. Analyzing the semantic types of claims and premises in an online persuasive forum. In *Proceedings of the 4th Workshop on Argument Mining*, pages 11–21, Copenhagen, Denmark. Association for Computational Linguistics.

Lu Ji, Zhongyu Wei, Xiangkun Hu, Yang Liu, Qi Zhang, and Xuanjing Huang. 2018. Incorporating argument-level interactions for persuasion comments evaluation using co-attention model. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3703–3714. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Klaus Krippendorff. 2004. Measuring the reliability of qualitative text analysis data. *Quality and Quantity*, 38(6):787–800.

Elena Musi and Mark Aakhus. 2018. Discovering argumentative patterns in energy polylogues: A macroscope for argument mining. *Argumentation*, 32(3):397–430.

Elena Musi, Debanjan Ghosh, and Smaranda Muresan. 2018. Changemyview through concessions: Do concessions increase persuasion? *Dialogue and Discourse (D&D)*, Vol 9(1).

Joonsuk Park, Cheryl Blake, and Claire Cardie. 2015. Toward machine-assisted participation in erulemaking: An argumentation model of evaluability. In *Proceedings of the 15th International Conference on Artificial Intelligence and Law*, ICAIL '15, pages 206–210, New York, NY, USA. ACM.

Joonsuk Park and Claire Cardie. 2018. A corpus of erulemaking user comments for measuring evaluability of arguments. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018.*

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics.

Isaac Persing and Vincent Ng. 2017. Why can't you convince me? modeling weaknesses in unpersuasive arguments. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 4082–4088.

Claudia Schulz, Steffen Eger, Johannes Daxenberger, Tobias Kahse, and Iryna Gurevych. 2018. Multi-task learning for argumentation mining in low-resource settings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 35–41. Association for Computational Linguistics.

Christian Stab and Iryna Gurevych. 2017. Parsing argumentation structures in persuasive essays. *Computational Linguistics*, 43(3):619–659.

Chenhao Tan, Vlad Niculae, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2016. Winning arguments: Interaction dynamics and persuasion strategies in good-faith online discussions. In *WWW*, pages 613–624. ACM.

# A Challenge Dataset and Effective Models for Aspect-Based Sentiment Analysis

**Qingnan Jiang[1], Lei Chen[1], Ruifeng Xu[2,3], Xiang Ao[4], Min Yang[1]***
[1]Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences
[2]Department of Computer Science, Harbin Institute of Technology (Shenzhen)
[3]Peng Cheng Laboratory
[4]Institute of Computing Technology, Chinese Academy of Sciences
jqnthomask@gmail.com, lei.chen@siat.ac.cn, xuruifeng@hit.edu.cn
aoxiang@ict.ac.cn, min.yang@siat.ac.cn

## Abstract

Aspect-based sentiment analysis (ABSA) has attracted increasing attention recently due to its broad applications. In existing ABSA datasets, most sentences contain only one aspect or multiple aspects with the same sentiment polarity, which makes ABSA task degenerate to sentence-level sentiment analysis. In this paper, we present a new large-scale Multi-Aspect Multi-Sentiment (MAMS) dataset, in which each sentence contains at least two different aspects with different sentiment polarities. The release of this dataset would push forward the research in this field. In addition, we propose simple yet effective CapsNet and CapsNet-BERT models which combine the strengths of recent NLP advances. Experiments on our new dataset show that the proposed model significantly outperforms the state-of-the-art baseline methods[1].

## 1 Introduction

Aspect-based sentiment analysis (ABSA) aims at identifying the sentiment polarity towards the specific aspect in a sentence. An target aspect refers to a word or a phrase describing an aspect of an entity. For example, in the sentence "*The decor is not special at all but their amazing food makes up for it*", there are two aspect terms "*decor*" and "*food*", and they are associated with *negative* and *positive* sentiment respectively.

Recently, neural network methods have dominated the study of ABSA since these methods can be trained end-to-end and automatically learn important features. (Wang et al., 2016) proposed to learn an embedding vector for each aspect, and these aspect embeddings were used to calculate the attention weights to capture important information with regard to the given aspects. (Tang et al., 2016b) developed the deep memory network to compute the importance degree and text representation of each context word with multiple attention layers. (Ma et al., 2017) introduced the interactive attention networks (IAN) to interactively learn attentions in contexts and targets, and generated the representations for target and context words separately. (Xue and Li, 2018) proposed to extract sentiment features with convolutional neural networks and selectively output aspect related features for classification with gating mechanisms. Subsequently, Transformer (Vaswani et al., 2017) and BERT based methods (Devlin et al., 2018) have shown high potentials on ABSA task. There are also several studies attempting to simulate the process of human reading cognition to further improve the performance of ABSA (Lei et al., 2019; Yang et al., 2019).

So far, several ABSA datasets have been constructed, including SemEval-2014 Restaurant Review dataset, Laptop Review dataset (Pontiki et al., 2014) and Twitter dataset (Dong et al., 2014). Although these three datasets have since become the benchmark datasets for the ABSA task, most sentences in these datasets consist of only one aspect or multiple aspects with the same sentiment polarity (see Table 1)[2], which makes aspect-based sentiment analysis degenerate to sentence-level sentiment analysis. Based on our empirical observation, the sentence-level sentiment classifiers without considering aspects can still achieve competitive results with many recent ABSA methods (see TextCNN and LSTM in Table 3). On the other hand, even advanced ABSA methods trained on these datasets can hardly distinguish the sentiment polarities towards different aspects in the sentences that contain multiple aspects and multiple sentiments.

*Min Yang is corresponding author

[1]Data and code can be found as: https://github.com/siat-nlp/MAMS-for-ABSA

[2]ATSA and ACSA represent aspect-term and aspect-category sentiment analysis, respectively.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6280–6285,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| Dataset | Type | Categ. | Size | MM_size |
|---|---|---|---|---|
| Restaurant | ATSA | 4 | 4827 | 1283 |
| Restaurant | ACSA | 4 | 4738 | 454 |
| Laptop | ATSA | 4 | 3012 | 604 |
| Twitter | ATSA | 3 | 6940 | 6 |

Table 1: Statistics of existing datasets for ABSA. Size and MM_size represent the total number of instances and multi-apsect multi-sentiment instances in the dataset. Each multi-apsect multi-sentiment instance contains multiple aspects with different sentiment polarities.

With the goal of advancing and facilitating research in the field of aspect-based sentiment analysis, in this paper, we present a new Multi-Aspect Multi-Sentiment (MAMS) dataset. In MAMS dataset, each sentence consists of at least two aspects with different sentiment polarities, making the proposed dataset more challenging compared with existing ABSA datasets. Considering merely the sentence-level sentiment of the sentence will fail to achieve good performance on MAMS dataset. We empirically evaluate the state-of-the-art ABSA methods on MAMS dataset, the poor results demonstrate that the proposed MAMS dataset is more challenging than the SemEval-2014 Restaurant Review dataset.

We analyze the properties of recent ABSA methods, and propose new capsule networks (denoted as CapsNet and CapsNet-BERT) to model the complicated relationship between aspects and contexts, which combine the strengths of recent NLP advances. Experimental results show that the proposed methods achieve significantly better results than the state-of-the-art baseline methods on MAMS and SemEval-14 Restaurant datasets.

Our main contributions are summarized as follows: (1) We manually annotate a large-scale multi-aspect multi-sentiment dataset, preventing ABSA degenerating to sentence-level sentiment analysis. The release of it would push forward the research of ABSA. (2) We propose a novel capsule network based model to learn the complicated relationship between aspects and contexts. (3) Experimental results show that the proposed method achieves significantly better results than the state-of-the-art baseline methods.

## 2 Dataset Construction

**Data Collection** Similar to SemEval-2014 Restaurant Review dataset (Pontiki et al., 2014), we annotate sentences from the Citysearch New York dataset collected by (Ganu et al., 2009). We split each document in the corpus into a few sentences, and remove the sentences consisting more than 70 words.

| Dataset | | Pos. | Neg. | Neu. | Total. |
|---|---|---|---|---|---|
| ATSA | Train | 3380 | 2764 | 5042 | 11186 |
| | Train (small) | 1089 | 892 | 1627 | 3608 |
| | Validation | 403 | 325 | 604 | 1332 |
| | Test | 400 | 329 | 607 | 1336 |
| ACSA | Train. | 1929 | 2084 | 3077 | 7090 |
| | Train (small) | 1000 | 1100 | 1613 | 3713 |
| | Validation | 241 | 259 | 388 | 888 |
| | Test | 245 | 263 | 393 | 901 |

Table 2: Statistics of MAMS dataset.

**Data Annotation** We create two versions of MAMS dataset for two subtasks of aspect-based sentiment analysis: aspect-term sentiment analysis (ATSA) and aspect-category sentiment analysis (ACSA).

For ATSA, we invited three experienced researchers who work on natural language processing (NLP) to extract aspect terms in the sentences and label the sentiment polarities with respect to the aspect terms. The sentences that consist of only one aspect term or multiple aspects with the same sentiment polarities are deleted. We also provide the start and end positions in a sentence for each aspect term.

For ACSA, we pre-defined eight coarse aspect categories: *food*, *service*, *staff*, *price*, *ambience*, *menu*, *place* and *miscellaneous*. Five aspect categories are adopted in SemEval-2014 Restaurant Review Dataset. We add three more aspect categories to deal with some confusing situations. Three experienced NLP researchers were asked to identify the aspect categories described in given sentences and determine the sentiment polarities towards these aspect categories. We only keep the sentences which consist of at least two unique aspect categories with different sentiment polarities.

**Dataset Analysis** The statistics of MAMS dataset for ATSA and ACSA are reported in Table 2. MAMS consists of 13,854 instances for ATSA and 8,879 instances for ACSA, which is 2.87 and 1.87 times of SemEval-2014 Restaurant Review dataset respectively. In MAMS, the sentences contain 2.62 aspect terms and 2.25 aspect categories in average. All sentences in MAMS contains multiple aspects with different sentiment polarities. On the contrary, the ra-

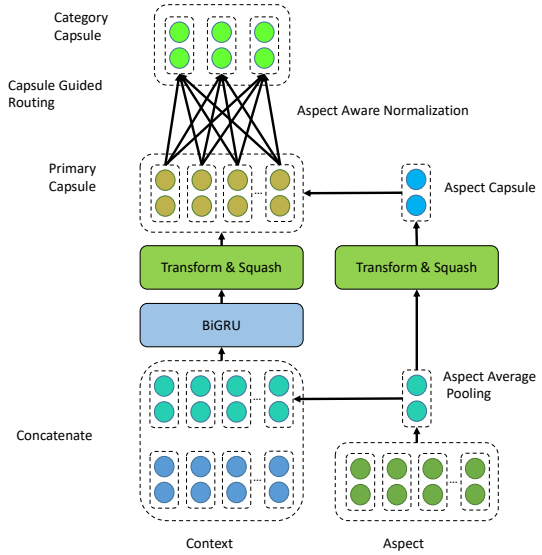


Figure 1: Model overview.

tio of multi-aspect multi-sentiment instances in Restaurant-ATSA, Restaurant-ACSA, Laptop and Twitter datasets are 26.58%, 9.58%, 20.05%, and 0.09% respectively, making the existing datasets less challenging since non-MAMS instances could be classified correctly without effectively modeling the relationships between contexts and aspects. For fair comparison with SemEval-14 Restaurant dataset, we also provide a small version of MAMS dataset (called MAMS-small) by sampling training instances from the MAMS training set, which has the same number of training instances with the SemEval-14 Restaurant dataset.

## 3 Methodology

We use $D$ to denote the collection of sentences in the training data. Given a sentence $S = \{w_1^s, ..., w_n^s\}$, an aspect term $A^t = \{w_1^a, ..., w_m^a\}$ or an aspect category $A^c$, aspect-level sentiment classification aims to predict the sentiment polarity $y \in \{1, ..., C\}$ of sentence $S$ with respect to $A^t$ or $A^c$. Here, $w$ denotes a specific word, $n$ and $m$ are the lengths of the sentence and aspect term, $C$ represents the number of sentiment categories.

As illustrated in Figure 1, the proposed model consists of an embedding layer, an encoding layer, a primary capsule layer and a category capsule layer.

### 3.1 Embedding Layer

In the embedding layer, we convert the sentence $S$ into word embeddings $\mathbf{E}$. For ACSA task, aspect category embedding $\mathbf{a}$ are randomly initialized and learned during training, while for ATSA task, aspect embedding $\mathbf{a}$ are computed as average pooling over aspect word embeddings. We get the aspect-aware sentence embedding $\mathbf{E}^{sa}$ by concatenating the aspect embedding $\mathbf{a}$ with each word embedding in $S$: $\mathbf{E}_i^{sa} = [\mathbf{E}_i; \mathbf{a}]$

### 3.2 Encoding Layer

We convert aspect-aware sentence representation $\mathbf{E}^{sa}$ into contextualized representation $\mathbf{H} = [\mathbf{h}_1, ..., \mathbf{h}_n]$ through bidirectional gated recurrent unit (BiGRU) (Cho et al., 2014) and residual connection (He et al., 2016).

$$\mathbf{H} = \text{BiGRU}(\mathbf{E}^{sa}) + \mathbf{E}^{sa} \quad (1)$$

### 3.3 Primary Capsule Layer

In the primary capsule layer, we get primary capsules $\mathbf{P} = [\mathbf{p}_i, ...\mathbf{p}_n]$ and aspect capsule $\mathbf{c}$ through linear transformation and squashing activation.

$$\mathbf{p}_i = \text{squash}(\mathbf{W}^p \mathbf{h}_i + \mathbf{b}^p) \quad (2)$$

$$\mathbf{c} = \text{squash}(\mathbf{W}^a \mathbf{a} + \mathbf{b}^a) \quad (3)$$

where $\mathbf{W}^p$, $\mathbf{b}^p$, $\mathbf{W}^a$ and $\mathbf{b}^a$ are learnable parameters. The squash function is defined as:

$$\text{squash}(\mathbf{s}) = \frac{\|\mathbf{s}\|^2}{1 + \|\mathbf{s}\|^2} \frac{\mathbf{s}}{\|\mathbf{s}\|} \quad (4)$$

**Aspect Aware Normalization** Due to the variable lengths of sentences, the number of primary capsules sent to upper layer capsules varies from sentence to sentence, leading to unstable training procedure. Extremely long sentences make the squash activation saturate and result in high confidence for all categories; while very short sentences in contrast will lead to low confidence for all categories. To alleviate this problem, we propose the aspect aware normalization that utilizes aspect capsule to select important primary capsules, and normalize primary capsule weights $\mathbf{u}$ by:

$$u_i = \frac{\exp(\mathbf{p}_i \mathbf{W}^n \mathbf{c})}{\sum_{j=1}^{n} \exp(\mathbf{p}_j \mathbf{W}^n \mathbf{c})} \quad (5)$$

where $\mathbf{W}^n$ is a learnable parameter.

**Capsule Guided Routing** Original dynamic routing mechanism (Sabour et al., 2017) suffer from inefficient training due to the iteration procedure of routing. And there is no upper layer information used to guide the routing process, which

makes dynamic routing work like a self-directed process.

Instead of computing coupling coefficients between primary capsules and category capsules during routing process, we design a capsule-guided routing mechanism, which leverages prior knowledge about the sentiment categories to guide the routing process effectively and efficiently. Specifically, we use a set of sentiment capsules to store prior knowledge about the sentiment categories. Let $\mathbf{G} \in R^{C \times d}$ be the sentiment matrix, which is initialized with the averaged embeddings of sentiment words. $C$ is the number of sentiment categories, and $d$ is the dimension of the sentiment embedding. We can get the sentiment capsules $\mathbf{Z} = [\mathbf{z}_1, ..., \mathbf{z}_C]$ by applying squash activation over sentiment matrix and compute the routing weights $\mathbf{w}$ by calculating the similarity between primary capsules and sentiment capsules:

$$\mathbf{z}_i = \text{squash}(\mathbf{G}_i) \tag{6}$$

$$w_{ij} = \frac{\exp(\mathbf{p}_i \mathbf{W}^r \mathbf{z}_j)}{\sum_{k=1}^{n} \exp(\mathbf{p}_i \mathbf{W}^r \mathbf{z}_k)} \tag{7}$$

### 3.4 Category Capsule Layer

Based on the normalization weights and routing weights, the final category capsules $\mathbf{V} = [\mathbf{v}_1, ..., \mathbf{v}_C]$ can be calculated as:

$$\mathbf{v}_j = \text{squash}(s \sum_{i=1}^{n} w_{ij} u_i \mathbf{p}_i) \tag{8}$$

where $s$ is a learnable scale parameter to scale the connection weights to a suitable level.

Following (Sabour et al., 2017), we use the margin loss as loss function for aspect-based sentiment classification:

$$L = \sum_{k=1}^{C} T_k \max(0, m^+ - \|\mathbf{v}_k\|)^2 + \lambda(1 - T_k) \max(0, \|\mathbf{v}_k\| - m^-)^2 \tag{9}$$

where $T_k = 1$ if and only if a category $k$ is present. $m^+$ and $m^-$ are the margin hyper-parameters. $\lambda$ control the loss for absent categories. In our experiments, $m^+$, $m^-$ and $\lambda$ is set to 0.9, 0.1, 0.6, respectively.

### 3.5 CapsNet-BERT

To utilize the features learned from large-scale corpus, we design the CapsNet-BERT model which combines the strength of BERT and capsule networks. We replace the embedding layer and encoding layer of CapsNet with pre-trained BERT. The CapsNet-BERT model takes "[CLS] sentence [SEP] aspect [SEP]" as input, which computes the deep representations of sentences and aspects with pre-trained BERT. We then feed the sentence and aspect representations into capsule layers and predict the corresponding sentiment polarities.

## 4 Experiments

### 4.1 Experimental Setup

**Experimental Data** In order to evaluate the effectiveness of our model, we conduct experiments on the two MAMS datasets and SemEval-14 Restaurant Review (Pontiki et al., 2014) dataset. All models share the same data pre-processing procedure, and use the same pre-trained word embeddings.

**Baseline Methods** We compare CapsNet with various baselines. (1) LSTM based models: TD_LSTM (Tang et al., 2016a), AT_LSTM (Wang et al., 2016), ATAE_LSTM (Wang et al., 2016), BiLSTM enhanced with attention mechanism, IAN (Ma et al., 2017) and AOA_LSTM (Huang et al., 2018); (2) CNN based model: GCAE (Xue and Li, 2018); (3) Attention based models: MemNet (Tang et al., 2016b) and AEN (Song et al., 2019). To verify the effectiveness of combining capsule networks and BERT, we also compare the performance of CapsNet-BERT with vanilla BERT model. Vanilla BERT model adopts "[CLS] sentence [SEP] aspect [SEP]" as input and predicts the sentiment polarity of the sentence towards the given aspect.

**Implementation Details** In all experiments, we use 300-dimentional word vectors pre-trained by GloVe (Pennington et al., 2014) to initialize the word embedding vectors for non-BERT models. The capsule size is set to 300. The batch sizes are set to 64 and 32 for CapsNet and CapsNet-BERT respectively. We use Adam optimizer (Kingma and Ba, 2015) to train our models. The learning rates are set to 0.0003 and 0.00003 for CapsNet and CapsNet-BERT respectively. We run all models for 5 times and report the average results on the test datasets. We fine-tune the hyper-parameters for all baselines on the validation set.

| Method | ATSA | | | ACSA | | |
|---|---|---|---|---|---|---|
| | MAMS | MAMS-small | Restaurant | MAMS | MAMS-small | Restaurant |
| TextCNN | 52.694 | 51.736 | 75.928 | 48.856 | 48.814 | 81.418 |
| LSTM | 52.486 | 50.134 | 75.552 | 48.546 | 48.346 | 82.076 |
| TD_LSTM | 74.596 | 73.582 | 75.630 | - | - | - |
| AT_LSTM | 77.558 | 73.028 | 76.200 | 66.436 | 65.128 | 83.100 |
| ATAE_LSTM | 77.054 | 71.708 | 77.200 | 70.634 | 66.814 | **84.000** |
| BiLSTM+Attn | 76.166 | 70.720 | 77.356 | 66.304 | 66.262 | 80.514 |
| IAN | 76.602 | 71.168 | 78.600 | - | - | - |
| AOA_LSTM | 77.260 | 72.290 | **81.200** | - | - | - |
| AEN | 66.722 | 61.706 | 80.980 | - | - | - |
| MemNet | 64.568 | 62.694 | 80.950 | 63.288 | 62.818 | 77.862 |
| GCAE | 77.588 | 73.246 | 77.280 | 72.098 | 66.968 | 79.350 |
| CapsNet | **79.776** | **73.860** | 80.786 | **73.986** | **67.128** | 83.554 |
| BERT | 82.218 | 79.440 | 84.460 | 78.292 | 75.316 | 90.442 |
| CapsNet-BERT | **83.391** | **80.910** | **85.934** | **79.461** | **76.366** | **91.375** |
| CapsNet-DR | 79.434 | 71.398 | 77.768 | 69.036 | 65.638 | 81.904 |
| CapsNet-BERT-DR | 82.970 | 80.092 | 84.646 | 79.132 | 75.634 | 90.736 |

Table 3: Experimental results on two MAMS datasets and SemEval-2014 Restaurant Review dataset for both ATSA and ACSA subtasks.

### 4.2 Experimental Results and Analysis

Experimental results are reported in Table 3. From Table 3 we draw the following conclusions. First, sentence-level sentiment classifiers (TextCNN and LSTM) achieve competitive results on SemEval-14 Restaurant Review dataset but perform poorly on MAMS datasets. This verifies that MAMS datasets can alleviate the task degeneration problem encountered in Restaurant dataset for ABSA. Second, most advanced and complex ABAS methods, which achieve impressive results on Restaurant dataset, perform poorly on the MAMS-small dataset. This verifies that MAMS (small) is more challenging than SemEval-14 Restaurant Review dataset. Third, attention based models without effectively modeling word order (e.g., MemNet and AEN) perform worst on MAMS since they lose word order information and cannot identify which part of context describes the given aspect. Fourth, CapsNet outperforms non-BERT baselines on 4 of 6 datasets, showing the potential of applying capsule networks to aspect-based sentiment analysis task. In addition, CapsNet-BERT performs significantly better than other models including BERT, indicating that combining capsule network and BERT can obtain additional improvement compared to vanilla BERT.

### 4.3 Ablation Study

To analyze the effectiveness of the proposed capsule-guided routing mechanism, we conduct ablation study that replace capsule-guided routing by dynamic routing in both CapsNet and CapsNet-BERT, resulting in CapsNet-DR and CapsNet-BERT-DR. From Table 3 (last two rows) we can see that capsule-guided routing boosts the performance of CapsNet and CapsNet-BERT on all the datasets.

## 5 Conclusion

In this paper, we present MAMS, a challenge dataset for aspect-based sentiment analysis, in which each sentence contains multiple aspects with different sentiment polarities. The proposed MAMS dataset could prevent aspect-level sentiment classification degenerating to sentence-level sentiment classification, which might push forward the researches on aspect-based sentiment analysis. In addition, we propose a simple yet effective capsule networks that significantly outperforms compared methods.

## Acknowledgements

Min Yang is partially supported by SIAT Innovation Program for Excellent Young Researchers (No. Y8G027) and sponsored by CCF-Tencent Open Research Fund. Xiang Ao is partially supported by the National Natural Science Foundation of China (No. 61602438), CCF-Tencent RhinoBird Young Faculty Open Research Fund (No. RAGR20180111), and Youth Innovation Promotion Association CAS.

## References


Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *Proceedings of the 52nd annual meeting of the association for computational linguistics (volume 2: Short papers)*, volume 2, pages 49–54.

Gayatree Ganu, Nomie Elhadad, and Amlie Marian. 2009. Beyond the stars: Improving rating predictions using review text content. In *WebDB*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Binxuan Huang, Yanglan Ou, and Kathleen M Carley. 2018. Aspect level sentiment classification with attention-over-attention neural networks. In *International Conference on Social Computing, Behavioral-Cultural Modeling and Prediction and Behavior Representation in Modeling and Simulation*, pages 197–206. Springer.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. *international conference on learning representations*.

Zeyang Lei, Yujiu Yang, Min Yang, Wei Zhao, Jun Guo, and Yi Liu. 2019. A human-like semantic cognition network for aspect-level sentiment classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6650–6657.

Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2017. Interactive attention networks for aspect-level sentiment classification. In *IJCAI'17 Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 4068–4074.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 27–35.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. 2017. Dynamic routing between capsules. *neural information processing systems*, pages 3856–3866.

Youwei Song, Jiahai Wang, Tao Jiang, Zhiyue Liu, and Yanghui Rao. 2019. Attentional encoder network for targeted sentiment classification. *arXiv preprint arXiv:1902.09314*.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2016a. Effective lstms for target-dependent sentiment classification. *international conference on computational linguistics*, pages 3298–3307.

Duyu Tang, Bing Qin, and Ting Liu. 2016b. Aspect level sentiment classification with deep memory network. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 214–224.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *neural information processing systems*, pages 5998–6008.

Yequan Wang, Minlie Huang, xiaoyan zhu, and Li Zhao. 2016. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 606–615.

Wei Xue and Tao Li. 2018. Aspect based sentiment analysis with gated convolutional networks. *meeting of the association for computational linguistics*, 1:2514–2523.

Min Yang, Qingnan Jiang, Ying Shen, Qingyao Wu, Zhou Zhao, and Wei Zhou. 2019. Hierarchical human-like strategy for aspect-level sentiment classification with sentiment linguistic knowledge and reinforcement learning. *Neural Networks*.

# Learning with Noisy Labels for Sentence-level Sentiment Classification

**Hao Wang** [♮,‡] **Bing Liu** [‡,*] **Chaozhuo Li** [§] **Yan Yang** [♮] **Tianrui Li** [♮]
[♮]School of Information Science and Technology, Southwest Jiaotong University
hwang@my.swjtu.edu.cn, {yyang;trli}@swjtu.edu.cn
[‡]Department of Computer Science, University of Illinois at Chicago
liub@uic.edu
[§]State Key Lab of Software Development Environment, Beihang University
lichaozhuo@buaa.edu.cn

## Abstract

Deep neural networks (DNNs) can fit (or even over-fit) the training data very well. If a DNN model is trained using data with noisy labels and tested on data with clean labels, the model may perform poorly. This paper studies the problem of learning with noisy labels for sentence-level sentiment classification. We propose a novel DNN model called NETAB (as shorthand for convolutional neural **NET**works with **AB**-networks) to handle noisy labels during training. NETAB consists of two convolutional neural networks, one with a noise transition layer for dealing with the input noisy labels and the other for predicting 'clean' labels. We train the two networks using their respective loss functions in a mutual reinforcement manner. Experimental results demonstrate the effectiveness of the proposed model.

## 1 Introduction

It is well known that sentiment annotation or labeling is subjective (Liu, 2012). Annotators often have many disagreements. This is especially so for crowd-workers who are not well trained. That is why one always feels that there are many errors in an annotated dataset. In this paper, we study whether it is possible to build accurate sentiment classifiers even with noisy-labeled training data. Sentiment classification aims to classify a piece of text according to the polarity of the sentiment expressed in the text, e.g., *positive* or *negative* (Pang and Lee, 2008; Liu, 2012; Zhang et al., 2018). In this work, we focus on sentence-level sentiment classification (SSC) with labeling errors.

As we will see in the experiment section, noisy labels in the training data can be highly damaging, especially for DNNs because they easily fit the training data and memorize their labels even when training data are corrupted with noisy labels (Zhang et al., 2017). Collecting datasets annotated with clean labels is costly and time-consuming as DNN based models usually require a large number of training examples. Researchers and practitioners typically have to resort to crowdsourcing. However, as mentioned above, the crowdsourced annotations can be quite noisy.

Research on learning with noisy labels dates back to 1980s (Angluin and Laird, 1988). It is still vibrant today (Mnih and Hinton, 2012; Natarajan et al., 2013, 2018; Menon et al., 2015; Gao et al., 2016; Liu and Tao, 2016; Khetan et al., 2018; Zhan et al., 2019) as it is highly challenging. We will discuss the related work in the next section.

This paper studies the problem of learning with noisy labels for SSC. Formally, we study the following problem.

*Problem Definition*: Given noisy labeled training sentences $S = \{(x_1, y_1), ..., (x_n, y_n)\}$, where $x_i|_{i=1}^n$ is the $i$-th sentence and $y_i \in \{1, ..., c\}$ is the sentiment label of this sentence, the noisy labeled sentences are used to train a DNN model for a SSC task. The trained model is then used to classify sentences with clean labels to one of the $c$ sentiment labels.

In this paper, we propose a convolutional neural **NET**work with **AB**-networks (NETAB) to deal with noisy labels during training, as shown in Figure 1. We will introduce the details in the subsequent sections. Basically, NETAB consists of two convolutional neural networks (CNNs) (see Figure 1), one for learning sentiment scores to predict 'clean' [1] labels and the other for learning a noise transition matrix to handle input noisy labels. We call the two CNNs A-network and AB-network, respectively. The fundamental here is that (1) DNNs memorize easy instances first and gradu-

[*]Corresponding author

[1]Here we use clean with single quotes as it is not completely clean. In practice, models can hardly produce completely clean labels.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6286–6292,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

ally adapt to hard instances as training epochs increase (Zhang et al., 2017; Arpit et al., 2017); and (2) noisy labels are theoretically flipped from the clean/true labels by a noise transition matrix (Sukhbaatar et al., 2015; Goldberger and Ben-Reuven, 2017; Han et al., 2018a,b). We motivate and propose a CNN model with a transition layer to estimate the noise transition matrix for the input noisy labels, while exploiting another CNN to predict 'clean' labels for the input training (and test) sentences. In training, we pre-train A-network in early epochs and then train AB-network and A-network with their own loss functions in an alternating manner. To our knowledge, this is the first work that addresses the noisy label problem in sentence-level sentiment analysis. Our experimental results show that the proposed model outperforms the state-of-the-art methods.

## 2 Related Work

Our work is related to sentence sentiment classification (SSC). SSC has been studied extensively (Hu and Liu, 2004; Pang and Lee, 2005; Zhao et al., 2008; Narayanan et al., 2009; Täckström and McDonald, 2011; Wang and Manning, 2012; Yang and Cardie, 2014; Kim, 2014; Tang et al., 2015; Wu et al., 2017; Wang et al., 2018). None of them can handle noisy labels. Since many social media datasets are noisy, researchers have tried to build robust models (Gamon, 2004; Barbosa and Feng, 2010; Liu et al., 2012). However, they treat noisy data as additional information and don't specifically handle noisy labels. A noise-aware classification model in (Zhan et al., 2019) trains using data annotated with multiple labels. Wang et al. (2016) exploited the connection of users and noisy labels of sentiments in social networks. Since the two works use multiple-labeled data or users' information (we only use single-labeled data, and we do not use any additional information), they have different settings than ours.

Our work is closely related to DNNs based approaches to learning with noisy labels. DNNs based approaches explored three main directions: (1) training DNNs on selected samples (Malach and Shalev-Shwartz, 2017; Jiang et al., 2018; Ren et al., 2018; Han et al., 2018b), (2) modifying the loss function of DNNs with regularization biases (Mnih and Hinton, 2012; Jindal et al., 2016; Patrini et al., 2017; Ghosh et al., 2017; Ma et al., 2018; Zhang and Sabuncu, 2018), and (3) plugging an extra layer into DNNs (Sukhbaatar et al., 2015; Bekker and Goldberger, 2016; Goldberger and Ben-Reuven, 2017; Han et al., 2018a). All these approaches were proposed for image classification where training images were corrupted with noisy labels. Some of them require noise rate to be known a priori in order to tune their models during training (Patrini et al., 2017; Han et al., 2018b). Our approach combines direction (1) and direction (3), and trains two networks jointly without knowing the noise rate. We have used five latest existing methods in our experiments for SSC. The experimental results show that they are inferior to our proposed method.

In addition, Xiao et al. (2015), Reed et al. (2015), Guan et al. (2016), Li et al. (2017), Veit et al. (2017), and Vahdat (2017) studied weakly-supervised DNNs or semi-supervised DNNs. But they still need some clean-labeled training data. We use no clean-labeled data.

## 3 Proposed Model

Our model builds on CNN (Kim, 2014). The key idea is to train two CNNs alternately, one for addressing the input noisy labels and the other for predicting 'clean' labels. The overall architecture of the proposed model is given in Figure 1. Before going further, we first introduce a proposition, a property, and an assumption below.

**Proposition 1** *Noisy labels are flipped from clean labels by an unknown noise transition matrix.*

Proposition 1 is reformulated from (Han et al., 2018a) and has been investigated in (Sukhbaatar et al., 2015; Goldberger and Ben-Reuven, 2017; Bekker and Goldberger, 2016). This proposition shows that if we know the noise transition matrix, we can use it to recover the clean labels. In other words, we can put noise transition matrix on clean labels to deal with noisy labels. Given these, we ask the following question: *How to estimate such an unknown noise transition matrix*?

Below we give a solution to this question based on the following property of DNNs.

**Property 1** *DNNs tend to prioritize memorization of simple instances first and then gradually memorize hard instances (Zhang et al., 2017).*

Arpit et al. (2017) further investigated this property of DNNs. Our setting is that simple instances are sentences of clean labels and hard instances are those with noisy labels. We also have the following assumption.

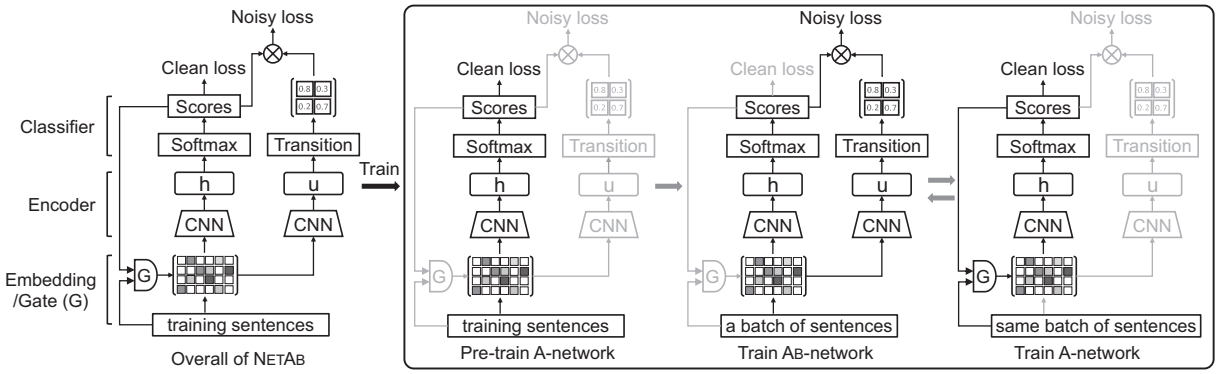


Figure 1: The proposed NETAB model (left) and its training method (right). Components in light gray color denote that these components are deactivated during training in that stage. (Color online)

**Assumption 1** *The noise rate of the training data is less than* $50\%$.

This assumption is usually satisfied in practice because without it, it is hard to tackle the input noisy labels during training.

Based on the above preliminaries, we need to estimate the noisy transition matrix $Q \in \mathbb{R}^{c\times c}$ ($c = 2$ in our case, i.e., *positive* and *negative*), and train two classifiers $\ddot{y} \sim P(\ddot{y}|x,\theta)$ and $\widehat{y} \sim P(\widehat{y}|x,\vartheta)$, where $x$ is an input sentence, $\ddot{y}$ is its noisy label, $\widehat{y}$ is its 'clean' label, $\theta$ and $\vartheta$ are the parameters of two classifiers. Note that both $\ddot{y}$ and $\widehat{y}$ here are the prediction results from our model, not the input labels. We propose to formulate the probability of the sentence $x$ labeled as $j$ with

$$P(\ddot{y}=j|x,\theta)=\sum_i P(\ddot{y}=j|\widehat{y}=i)P(\widehat{y}=i|x,\vartheta) \tag{1}$$

where $P(\ddot{y}=j|\widehat{y}=i)$ is an item (the $ji$-th item) in the noisy transition matrix $Q$. We can see that the noisy transition matrix $Q$ is exploited on the 'clean' scores $P(\widehat{y}|x,\vartheta)$ to tackle noisy labels.

We now present our model NETAB and introduce how NETAB performs Eq. (1). As shown in Figure 1, NETAB consists of two CNNs. The intuition here is that we use one CNN to perform $P(\widehat{y}=i|x,\vartheta)$ and use another CNN to perform $P(\ddot{y}=j|x,\theta)$. Meanwhile, the CNN performing $P(\ddot{y}=j|x,\theta)$ estimates the noise transition matrix $Q$ to deal with noisy labels. Thus we add a transition layer into this CNN.

More precisely, in Figure 1, the CNN with a clean loss performs $P(\widehat{y}=i|x,\vartheta)$. We call this CNN the A-network. The other CNN with a noisy loss performs $P(\ddot{y}=j|x,\theta)$. We call this CNN the AB-network. AB-network shares all the parameters of A-network except the parameters from the Gate unit and the clean loss. In addition, AB-network has a transition layer to estimate the noisy transition matrix $Q$. In such a way, A-network predict 'clean' labels, and AB-network handles the input noisy labels.

We use cross-entropy with the predicted labels $\ddot{y}$ and the input labels $y$ (given in the dataset) to compute the noisy loss, formulated as below

$$\mathcal{L}_{noisy}=-\frac{1}{|\ddot{S}|}\sum_{x\in\ddot{S}}\sum_i \mathbb{I}(y=i|x)\log P(\ddot{y}=i|x) \tag{2}$$

where $\mathbb{I}$ is the indicator function (if $y==i$, $\mathbb{I}=1$; otherwise, $\mathbb{I}=0$), and $|\ddot{S}|$ is the number of sentences to train AB-network in each batch.

Similarly, we use cross-entropy with the predicted labels $\widehat{y}$ and the input labels $y$ to compute the clean loss, formulated as

$$\mathcal{L}_{clean}=-\frac{1}{|\widehat{S}|}\sum_{x\in\widehat{S}}\sum_i \mathbb{I}(y=i|x)\log P(\widehat{y}=i|x) \tag{3}$$

where $|\widehat{S}|$ is the number of sentences to train A-network in each batch.

Next we introduce how our model learns the parameters ($\vartheta$, $\theta$ and $Q$). An embedding matrix $v$ is produced for each sentence $x$ by looking up a pre-trained word embedding database (e.g., GloVe.840B (Pennington et al., 2014)). Then an encoding vector $h=CNN(v)$ (and $u=CNN(v)$) is produced for each embedding matrix $v$ in A-network (and AB-network). A sofmax classifier gives us $P(\hat{y}=i|x,\vartheta)$ (i.e., 'clean' sentiment scores) on the learned encoding vector $h$. As the noise transition matrix $Q$ indicates the transition values from clean labels to noisy labels, we com-

| | #Noisy Training Data | #Clean Training Data | #Validation Data | #Test Data |
|---|---|---|---|---|
| Movie | 13539P, 13350N | 4265P, 4265N | 105P, 106N | 960P, 957N |
| Laptop | 9702P, 7876N | 1064P, 490N | 33P, 20N | 298P, 175N |
| Restaurant | 8094P, 10299N | 1087P, 823N | 39P, 14N | 339P, 116N |

Table 1: Summary statistics of the datasets. Number of positive (P) and negative (N) sentences in (noisy and clean) training data, validation data, and test data. The second column shows the statistics of sentences extracted from the 2,000 reviews of each dataset. The last three columns show the statistics of the sentences in three clean-labeled datasets, see "Clean-labeled Datasets".

pute $Q$ as follows

$$Q = [q_1; q_2] \quad (4)$$

$$q_i = softmax(g_i f_i),\ i = 1, 2 \quad (5)$$

$$g_i = tanh(W_i u + b_i) \quad (6)$$

where $W_i$ is a trainable parameter matrix, $b_i$ and $f_i$ are two trainable parameter vectors. They are trained in the AB-network. Finally, $P(\ddot{y} = j|x, \theta)$ is computed by Eq. (1).

In training, NETAB is trained end-to-end. Based on Proposition 1 and Property 1, we pre-train A-network in early epochs (e.g., 5 epochs). Then we train AB-network and A-network in an alternating manner. The two networks are trained using their respective cross-entropy loss. Given a batch of sentences, we first train AB-network. Then we use the scores predicted from A-network to select some possibly clean sentences from this batch and train A-network on the selected sentences. Specifically speaking, we use the predicted scores to compute sentiment labels by $\arg\max_i\{\ddot{y} = i|\ddot{y} \sim P(\ddot{y}|x, \theta)\}$. Then we select the sentences whose resulting sentiment label equals to the input label. The selection process is marked by a Gate unit in Figure 1. When testing a sentence, we use A-network to produce the final classification result.

## 4 Experiments

In this section, we evaluate the performance of the proposed NETAB model. we conduct two types of experiments. (1) We corrupt clean-labeled datasets to produce noisy-labeled datasets to show the impact of noises on sentiment classification accuracy. (2) We collect some real noisy data and use them to train models to evaluate the performance of NETAB.

**Clean-labeled Datasets.** We use three clean labeled datasets. The first one is the movie sentence polarity dataset from (Pang and Lee, 2005). The other two datasets are laptop and restaurant datasets collected from SemEval-2016 [2]. The former consists of laptop review sentences and the latter consists of restaurant review sentences. The original datasets (i.e., Laptop and Restaurant) were annotated with aspect polarity in each sentence. We used all sentences with only one polarity (*positive* or *negative*) for their aspects. That is, we only used sentences with aspects having the same sentiment label in each sentence. Thus, the sentiment of each aspect gives the ground-truth as the sentiments of all aspects are the same.

For each clean-labeled dataset, the sentences are randomly partitioned into training set and test set with 80% and 20%, respectively. Following (Kim, 2014), We also randomly select 10% of the test data for validation to check the model during training. Summary statistics of the training, validation, and test data are shown in Table 1.

**Noisy-labeled Training Datasets.** For the above three domains (movie, laptop, and restaurant), we collected 2,000 reviews for each domain from the same review source. We extracted sentences from each review and assigned review's label to its sentences. Like previous work, we treat 4 or 5 stars as positive and 1 or 2 stars as negative. The data is noisy because a positive (negative) review can contain negative (positive) sentences, and there are also neutral sentences. This gives us three noisy-labeled training datasets. We still use the same test sets as those for the clean-labeled datasets. Summary statistics of all the datasets are shown in Table 1.

**Experiment 1:** Here we use the clean-labeled data (i.e., the last three columns in Table 1). We corrupt the clean training data by switching the labels of some random instances based on a noise rate parameter. Then we use the corrupted data to train NETAB and CNN (Kim, 2014).

The test accuracy curves with the noise rates [0, 0.1, 0.2, 0.3, 0.4, 0.5] are shown in Figure 2.

[2] http://alt.qcri.org/semeval2016/task5/

| Methods | Movie | | | Laptop | | | Restaurant | | |
|---|---|---|---|---|---|---|---|---|---|
| | ACC | F1_pos | F1_neg | ACC | F1_pos | F1_neg | ACC | F1_pos | F1_neg |
| NBSVM-uni (Wang and Manning, 2012) | 0.6791 | 0.6663 | 0.6910 | 0.7637 | 0.8216 | 0.6500 | 0.7949 | 0.8478 | 0.6858 |
| NBSVM-bi (Wang and Manning, 2012) | 0.6416 | 0.6438 | 0.6394 | 0.7784 | 0.8320 | **0.6749** | 0.7154 | 0.7834 | 0.5853 |
| CNN (Kim, 2014) | 0.6667 | 0.6467 | 0.6844 | 0.7737 | 0.8381 | 0.6245 | 0.8329 | 0.8841 | 0.7007 |
| Adaptation (Goldberger and Ben-Reuven, 2017) | 0.6682 | 0.6708 | 0.6656 | 0.7272 | 0.7936 | 0.5981 | 0.8285 | 0.8872 | 0.6422 |
| Forward (Patrini et al., 2017) | 0.6864 | 0.6753 | 0.6969 | 0.7547 | 0.8170 | 0.6282 | 0.8329 | 0.8882 | 0.6695 |
| Backward (Patrini et al., 2017) | 0.6651 | 0.6160 | 0.6830 | 0.7124 | 0.7834 | 0.5723 | 0.7890 | 0.8485 | 0.6521 |
| Masking (Han et al., 2018a) | 0.6708 | 0.6631 | 0.6782 | 0.7188 | 0.7787 | 0.6144 | 0.8219 | 0.8789 | 0.6639 |
| Co-teaching (Han et al., 2018b) | 0.6150 | 0.5980 | 0.6306 | 0.7145 | 0.7867 | 0.5686 | 0.7978 | 0.8575 | 0.6515 |
| NETAB (Our method) | **0.7047** | **0.7076** | **0.7017** | **0.7928** | **0.8487** | 0.6711 | **0.8593** | **0.9056** | **0.7241** |

Table 2: Accuracy (ACC) of both classes, F1 (F1_pos) of positive class and F1 (F1_neg) of negative class on clean test data/sentences. Training data are real noisy-labeled sentences.

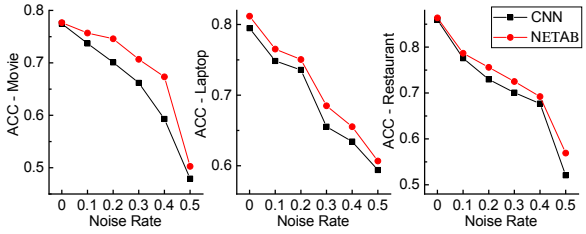


Figure 2: Accuracy (ACC) on clean test data. For training, the labels of clean data are flipped with the noise rates [0, 0.1, 0.2, 0.3, 0.4, 0.5]. For example, 0.1 means that 10% of the labels are flipped. (Color online)

From the figure, we can see that the test accuracy drops from around 0.8 to 0.5 when the noise rate increases from 0 to 0.5, but our NETAB outperforms CNN. The results clearly show that the performance of the CNN drops quite a lot with the noise rate increasing.

**Experiment 2:** Here we use the real noisy-labeled training data to train our model and the baselines, and then test on the test data in Table 1. Our goal is two fold. First, we want to evaluate NETAB using real noisy data. Second, we want to see whether sentences with review level labels can be used to build effective SSC models.

**Baselines.** We use one strong non-DNN baseline, NBSVM (with unigrams or bigrams features) (Wang and Manning, 2012) and six DNN baselines. The first DNN baseline is CNN (Kim, 2014), which does not handle noisy labels. The other five were designed to handle noisy labels.

The comparison results are shown in Table 2. From the results, we can make the following observations. (1) Our NETAB model achieves the best ACC and F1 on all datasets except for F1 of negative class on Laptop. The results demonstrate the superiority of NETAB. (2) NETAB outperforms the baselines designed for learning with noisy labels. These baselines are inferior to ours as they were tailored for image classification. Note that we found no existing method to deal with noisy labels for SSC.

**Training Details.** We use the publicly available pre-trained embedding GloVe.840B (Pennington et al., 2014) to initialize the word vectors and the embedding dimension is 300.

For each baseline, we obtain the system from its author and use its default parameters. As the DNN baselines (except CNN) were proposed for image classification, we change the input channels from 3 to 1. For our NETAB, we follow Kim (2014) to use window sizes of 3, 4 and 5 words with 100 feature maps per window size, resulting in 300-dimensional encoding vectors. The input length of sentence is set to 40. The network parameters are updated using the Adam optimizer (Kingma and Ba, 2014) with a learning rate of 0.001. The learning rate is clipped gradually using a norm of 0.96 in performing the Adam optimization. The dropout rate is 0.5 in the input layer. The number of epochs is 200 and batch size is 50.

## 5 Conclusions

This paper proposed a novel CNN based model for sentence-level sentiment classification learning for data with noisy labels. The proposed model learns to handle noisy labels during training by training two networks alternately. The learned noisy transition matrices are used to tackle noisy labels. Experimental results showed that the proposed model outperforms a wide range of baselines markedly. We believe that learning with noisy labels is a promising direction as it is often easy to collect noisy-labeled training data.

## Acknowledgments

Hao Wang and Yan Yang's work was partially supported by a grant from the National Natural Science Foundation of China (No. 61572407).

## References


Dana Angluin and Philip Laird. 1988. Learning from noisy examples. *Machine Learning*, 2(4):343–370.

Devansh Arpit, Stanislaw K. Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron C. Courville, Yoshua Bengio, and Simon Lacoste-Julien. 2017. A closer look at memorization in deep networks. In *ICML*, pages 233–242.

Luciano Barbosa and Junlan Feng. 2010. Robust sentiment detection on twitter from biased and noisy data. In *COLING*, pages 36–44.

Alan Joseph Bekker and Jacob Goldberger. 2016. Training deep neural-networks based on unreliable labels. In *ICASSP*, pages 2682–2686.

Michael Gamon. 2004. Sentiment classification on customer feedback data: Noisy data, large feature vectors, and the role of linguistic analysis. In *COLING*, pages 841–847.

Wei Gao, Lu Wang, Yu-Feng Li, and Zhi-Hua Zhou. 2016. Risk minimization in the presence of label noise. In *AAAI*, pages 1575–1581.

Aritra Ghosh, Himanshu Kumar, and PS Sastry. 2017. Robust loss functions under label noise for deep neural networks. In *AAAI*, pages 1919–1925.

Jacob Goldberger and Ehud Ben-Reuven. 2017. Training deep neural-networks using a noise adaptation layer. In *ICLR*, pages 1–9.

Ziyu Guan, Long Chen, Wei Zhao, Yi Zheng, Shulong Tan, and Deng Cai. 2016. Weakly-supervised deep learning for customer review sentiment classification. In *IJCAI*, pages 3719–3725.

Bo Han, Jiangchao Yao, Gang Niu, Mingyuan Zhou, Tvor Tsang, Ya Zhang, and Masashi Sugiyama. 2018a. Masking: A new perspective of noisy supervision. In *NIPS*, pages 5836–5846.

Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. 2018b. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *NIPS*, pages 8527–8537.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *KDD*, pages 168–177.

Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. 2018. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *ICML*, pages 2309–2318.

Ishan Jindal, Matthew Nokleby, and Xuewen Chen. 2016. Learning deep networks from noisy labels with dropout regularization. In *ICDM*, pages 967–972.

Ashish Khetan, Zachary C Lipton, and Animashree Anandkumar. 2018. Learning from noisy singly-labeled data. In *ICLR*, pages 1–15.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1476–1751.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Yuncheng Li, Jianchao Yang, Yale Song, Liangliang Cao, Jiebo Luo, and Li-Jia Li. 2017. Learning from noisy labels with distillation. In *CVPR*, pages 1910–1918.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.

Kun-Lin Liu, Wu-Jun Li, and Minyi Guo. 2012. Emoticon smoothed language models for twitter sentiment analysis. In *AAAI*, pages 1678–1684.

Tongliang Liu and Dacheng Tao. 2016. Classification with noisy labels by importance reweighting. *TPAMI*, 38(3):447–461.

Xingjun Ma, Yisen Wang, Michael E Houle, Shuo Zhou, Sarah M Erfani, Shu-Tao Xia, Sudanthi Wijewickrema, and James Bailey. 2018. Dimensionality-driven learning with noisy labels. In *ICML*, pages 3361–3370.

Eran Malach and Shai Shalev-Shwartz. 2017. Decoupling "when to update" from "how to update". In *NIPS*, pages 960–970.

Aditya Menon, Brendan Van Rooyen, Cheng Soon Ong, and Bob Williamson. 2015. Learning from corrupted binary labels via class-probability estimation. In *ICML*, pages 125–134.

Volodymyr Mnih and Geoffrey E Hinton. 2012. Learning to label aerial images from noisy data. In *ICML*, pages 567–574.

Ramanathan Narayanan, Bing Liu, and Alok Choudhary. 2009. Sentiment analysis of conditional sentences. In *EMNLP*, pages 180–189.

Nagarajan Natarajan, Inderjit S Dhillon, Pradeep Ravikumar, and Ambuj Tewari. 2018. Cost-sensitive learning with noisy labels. *Journal of Machine Learning Research*, 18:1–33.

Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. 2013. Learning with noisy labels. In *NIPS*, pages 1196–1204.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL*, pages 115–124.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, 2(1–2):1–135.

Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. 2017. Making deep neural networks robust to label noise: A loss correction approach. In *CVPR*, pages 1944–1952.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. 2015. Training deep neural networks on noisy labels with bootstrapping. In *ICLR Workshop Track*, pages 1–11.

Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. 2018. Learning to reweight examples for robust deep learning. In *ICML*, pages 4331–4340.

Sainbayar Sukhbaatar, Joan Bruna, Manohar Paluri, Lubomir Bourdev, and Rob Fergus. 2015. Training convolutional networks with noisy labels. In *ICLR Workshop Track*, pages 1–11.

Oscar Täckström and Ryan McDonald. 2011. Semi-supervised latent variable models for sentence-level sentiment analysis. In *ACL*, pages 569–574.

Duyu Tang, Bing Qin, Furu Wei, Li Dong, Ting Liu, and Ming Zhou. 2015. A joint segmentation and classification framework for sentence level sentiment classification. *TASLP*, 23(11):1750–1761.

Arash Vahdat. 2017. Toward robustness against label noise in training deep discriminative neural networks. In *NIPS*, pages 5596–5605.

Andreas Veit, Neil Alldrin, Gal Chechik, Ivan Krasin, Abhinav Gupta, and Serge Belongie. 2017. Learning from noisy large-scale datasets with minimal supervision. In *CVPR*, pages 839–847.

Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *ACL*, pages 90–94.

Yaowei Wang, Yanghui Rao, Xueying Zhan, Huijun Chen, Maoquan Luo, and Jian Yin. 2016. Sentiment and emotion classification over noisy labels. *Knowledge-Based Systems*, 111:207–216.

Yequan Wang, Aixin Sun, Jialong Han, Ying Liu, and Xiaoyan Zhu. 2018. Sentiment analysis by capsules. In *WWW*, pages 1165–1174.

Fangzhao Wu, Jia Zhang, Zhigang Yuan, Sixing Wu, Yongfeng Huang, and Jun Yan. 2017. Sentence-level sentiment classification with weak supervision. In *SIGIR*, pages 973–976.

Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. 2015. Learning from massive noisy labeled data for image classification. In *CVPR*, pages 2691–2699.

Bishan Yang and Claire Cardie. 2014. Context-aware learning for sentence-level sentiment analysis with posterior regularization. In *ACL*, pages 325–335.

Xueying Zhan, Yaowei Wang, Yanghui Rao, and Qing Li. 2019. Learning from multi-annotator data: A noise-aware classification framework. *ACM Trans. Inf. Syst.*, 37(2):26:1–26:28.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2017. Understanding deep learning requires rethinking generalization. In *ICLR*, pages 1–15.

Lei Zhang, Shuai Wang, and Bing Liu. 2018. Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1253.

Zhilu Zhang and Mert Sabuncu. 2018. Generalized cross entropy loss for training deep neural networks with noisy labels. In *NIPS*, pages 8778–8788.

Jun Zhao, Kang Liu, and Gen Wang. 2008. Adding redundant features for CRFs-based sentence sentiment classification. In *EMNLP*, pages 117–126.

# DENS: A Dataset for Multi-class Emotion Analysis

**Chen Liu** and **Muhammad Osama** and **Anderson de Andrade**
Wattpad
Toronto, ON, Canada
cecilia, muhammad.osama, anderson@wattpad.com

## Abstract

We introduce a new dataset for multi-class emotion analysis from long-form narratives in English. The Dataset for Emotions of Narrative Sequences (DENS) was collected from both classic literature available on Project Gutenberg and modern online narratives available on Wattpad, annotated using Amazon Mechanical Turk. A number of statistics and baseline benchmarks are provided for the dataset. Of the tested techniques, we find that the fine-tuning of a pre-trained BERT model achieves the best results, with an average micro-F1 score of 60.4%. Our results show that the dataset provides a novel opportunity in emotion analysis that requires moving beyond existing sentence-level techniques.

## 1 Introduction

Humans experience a variety of complex emotions in daily life. These emotions are heavily reflected in our language, in both spoken and written forms.

Many recent advances in natural language processing on emotions have focused on product reviews (McAuley et al., 2015) and tweets (Mohammad et al., 2018; Kant et al., 2018). These datasets are often limited in length (e.g. by the number of words in tweets), purpose (e.g. product reviews), or emotional spectrum (e.g. binary classification).

Character dialogues and narratives in storytelling usually carry strong emotions. A memorable story is often one in which the emotional journey of the characters resonates with the reader. Indeed, emotion is one of the most important aspects of narratives. In order to characterize narrative emotions properly, we must move beyond binary constraints (e.g. good or bad, happy or sad).

In this paper, we introduce the Dataset for Emotions of Narrative Sequences (DENS) for emotion analysis, consisting of passages from long-form fictional narratives from both classic literature and modern stories in English. The data samples consist of self-contained passages that span several sentences and a variety of subjects. Each sample is annotated by using one of 9 classes and an indicator for annotator agreement.

## 2 Background

Using the categorical basic emotion model (Plutchik, 1979), (Mohammad and Kiritchenko, 2015; Mohammad, 2012) studied creating lexicons from tweets for use in emotion analysis. Recently, (Mohammad et al., 2018), (Klinger et al., 2018) and (Kant et al., 2018) proposed shared-tasks for multi-class emotion analysis based on tweets.

Fewer works have been reported on understanding emotions in narratives. Emotional Arc (Reagan et al., 2016) is one recent advance in this direction. The work used lexicons and unsupervised learning methods based on unlabelled passages from titles in Project Gutenberg[1].

For labelled datasets on narratives, (Alm et al., 2005) provided a sentence-level annotated corpus of childrens' stories and (Kim and Klinger, 2018) provided phrase-level annotations on selected Project Gutenberg titles.

To the best of our knowledge, the dataset in this work is the first to provide multi-class emotion labels on passages, selected from both Project Gutenberg and modern narratives. The dataset is available upon request for non-commercial, research only purposes[2].

## 3 Dataset

In this section, we describe the process used to collect and annotate the dataset.

[1] https://www.gutenberg.org/

[2] Please send requests to: academic_dataset@wattpad.com



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6293–6298, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

### 3.1 Plutchik's Wheel of Emotions

The dataset is annotated based on a modified Plutchik's wheel of emotions.

The original Plutchik's wheel consists of 8 primary emotions: *Joy, Sadness, Anger, Fear, Anticipation, Surprise, Trust, Disgust*. In addition, more complex emotions can be formed by combing two basic emotions. For example, *Love* is defined as a combination of *Joy* and *Trust* (Fig. 1).

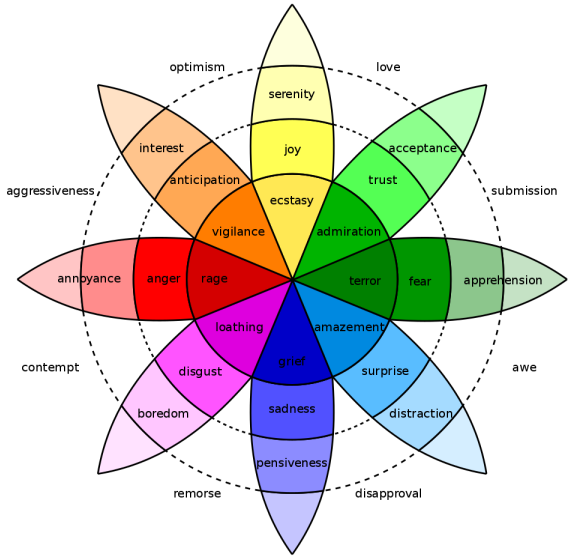


Figure 1: Plutchik's wheel of emotions (Wikimedia, 2011)

The intensity of an emotion is also captured in Plutchik's wheel. For example, the primary emotion of *Anger* can vary between *Annoyance* (mild) and *Rage* (intense).

We conducted an initial survey based on 100 stories with a significant fraction sampled from the romance genre. We asked readers to identify the major emotion exhibited in each story from a choice of the original 8 primary emotions.

We found that readers have significant difficulty in identifying *Trust* as an emotion associated with romantic stories. Hence, we modified our annotation scheme by removing *Trust* and adding *Love*. We also added the *Neutral* category to denote passages that do not exhibit any emotional content.

The final annotation categories for the dataset are: *Joy, Sadness, Anger, Fear, Anticipation, Surprise, Love, Disgust, Neutral*.

### 3.2 Passage Selection

We selected both classic and modern narratives in English for this dataset. The modern narratives were sampled based on popularity from Wattpad. We parsed selected narratives into passages, where a passage is considered to be eligible for annotation if it contained between 40 and 200 tokens.

In long-form narratives, many non-conversational passages are intended for transition or scene introduction, and may not carry any emotion. We divided the eligible passages into two parts, and one part was pruned using selected emotion-rich but ambiguous lexicons such as *cry, punch, kiss, etc.*. Then we mixed this pruned part with the unpruned part for annotation in order to reduce the number of neutral passages. See Appendix A.1 for the lexicons used.

### 3.3 Mechanical Turk (MTurk)

MTurk was set up using the standard sentiment template and instructed the crowd annotators to 'pick the best/major emotion embodied in the passage'.

We further provided instructions to clarify the intensity of an emotion, such as: "*Rage/Annoyance* is a form of *Anger*", "*Serenity/Ecstasy* is a form of *Joy*", and "*Love* includes *Romantic/Family/Friendship*", along with sample passages.

We required all annotators have a 'master' MTurk qualification. Each passage was labelled by 3 unique annotators. Only passages with a majority agreement between annotators were accepted as valid. This is equivalent to a Fleiss's $\kappa$ score of greater than $0.4$.

For passages without majority agreement between annotators, we consolidated their labels using in-house data annotators who are experts in narrative content. A passage is accepted as valid if the in-house annotator's label matched any one of the MTurk annotators' labels. The remaining passages are discarded. We provide the fraction of annotator agreement for each label in the dataset.

Though passages may lose some emotional context when read independently of the complete narrative, we believe annotator agreement on our dataset supports the assertion that small excerpts can still convey coherent emotions.

During the annotation process, several annotators had suggested for us to include additional emotions such as confused, pain, and jealousy,

| Genre | Distribution (%) |
|---|---|
| Mystery/Thriller | 19.7 |
| Paranormal | 16.6 |
| Fantasy | 13.2 |
| Horror | 11.3 |
| Romance | 8.7 |
| Action/Adventure | 5.3 |
| Other | 9.3 |

Table 1: Genre distribution of the modern narratives

which are common to narratives. As they were not part of the original Plutchik's wheel, we decided to not include them. An interesting future direction is to study the relationship between emotions such as 'pain versus sadness' or 'confused versus surprise' and improve the emotion model for narratives.

### 3.4 Dataset Statistics

The dataset contains a total of 9710 passages, with an average of 6.24 sentences per passage, 16.16 words per sentence, and an average length of 86 words.

The vocabulary size is 28K (when lowercased). It contains over 1600 unique titles across multiple categories, including 88 titles (1520 passages) from Project Gutenberg. All of the modern narratives were written after the year 2000, with notable amount of themes in *coming-of-age*, *strong-female-lead*, and *LGBTQ+*. The genre distribution is listed in Table 1.

In the final dataset, 21.0% of the data has consensus between all annotators, 73.5% has majority agreement, and 5.48% has labels assigned after consultation with in-house annotators.

The distribution of data points over labels with top lexicons (lower-cased, normalized) is shown in Table 2. Note that the *Disgust* category is very small and should be discarded. Furthermore, we suspect that the data labelled as *Surprise* may be noisier than other categories and should be discarded as well.

Table 3 shows a few examples labelled data from classic titles. More examples can be found in Table 6 in the Appendix A.2.

## 4 Benchmarks

We performed benchmark experiments on the dataset using several different algorithms. In all experiments, we have discarded the data labelled with *Surprise* and *Disgust*.

We pre-processed the data by using the SpaCy[3] pipeline. We masked out named entities with entity-type specific placeholders to reduce the chance of benchmark models utilizing named entities as a basis for classification.

Benchmark results are shown in Table 4. The dataset is approximately balanced after discarding the *Surprise* and *Disgust* classes. We report the average micro-F1 scores, with 5-fold cross validation for each technique.

We provide a brief overview of each benchmark experiment below. Among all of the benchmarks, Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018) achieved the best performance with a 0.604 micro-F1 score.

Overall, we observed that deep-learning based techniques performed better than lexical based methods. This suggests that a method which attends to context and themes could do well on the dataset.

### 4.1 Bag-of-Words-based Benchmarks

We computed bag-of-words-based benchmarks using the following methods:

- Classification with TF-IDF + Linear SVM (TF-IDF + SVM)
- Classification with Depeche++ Emotion lexicons (Araque et al., 2018) + Linear SVM (Depeche + SVM)
- Classification with NRC Emotion lexicons (Mohammad and Turney, 2010, 2013) + Linear SVM (NRC + SVM)
- Combination of TF-IDF and NRC Emotion lexicons (TF-NRC + SVM)

### 4.2 Doc2Vec + SVM

We also used simple classification models with learned embeddings. We trained a Doc2Vec model (Le and Mikolov, 2014) using the dataset and used the embedding document vectors as features for a linear SVM classifier.

### 4.3 Hierarchical RNN

For this benchmark, we considered a Hierarchical RNN, following (Sordoni et al., 2015). We used two BiLSTMs (Graves et al., 2005) with 256 units

[3] https://spacy.io/

| Label | Gutenberg | Total | Top Lexicons |
|---|---|---|---|
| Neutral | 318 | 1711 | take, love, long, really, want, always, though, away, look |
| Fear | 159 | 1412 | left, behind, right, want, let, death, go, say, think |
| Sadness | 195 | 1402 | father, always, little, look, something, us, really, mother, think |
| Anger | 192 | 1306 | feel, much, well, man, look, us, say, something, love |
| Joy | 241 | 1266 | see, always, let, long, make, hand, away, get, really |
| Love | 162 | 1157 | hand, know, right, let, happy, get, ever, us, look |
| Anticipation | 147 | 1020 | know, long, life, make, get, think, blood, want, feel |
| Surprise | 102 | 362 | love, find, looking, know, well, much, something, door, really |
| Disgust | 4 | 74 | get, hand, inside, let, hate, table, men, always, make |

Table 2: Dataset label distribution

| Text | Label |
|---|---|
| *I found this was a little too close upon him, but I made it up in what follows. He stood stock-still for a while and said nothing, and I went on thus: "You cannot," says I, "without the highest injustice, believe that I yielded upon all these persuasions without a love not to be questioned, not to be shaken again by anything that could happen afterward. If you have such dishonourable thoughts of me, I must ask you what foundation in any of my behaviour have I given for such a suggestion?"* | Angry |
| *She stretched hers eagerly and gratefully towards him. What had happened? Through all the numbness of her blood, there sprang a strange new warmth from his strong palm, and a pulse, which she had almost forgotten as a dream of the past, began to beat through her frame. She turned around all a-tremble, and saw his face in the glow of the coming day.* | Anticipation |
| *Ah! That moving procession that has left me by the road-side! Its fantastic colors are more brilliant and beautiful than the sun on the undulating waters. What matter if souls and bodies are failing beneath the feet of the ever-pressing multitude! It moves with the majestic rhythm of the spheres. Its discordant clashes sweep upward in one harmonious tone that blends with the music of other worlds–to complete God's orchestra.* | Joy |

Table 3: Sample data from classic titles

| Model | micro-F1 |
|---|---|
| TF-IDF + SVM | 0.450 |
| Depeche + SVM | 0.254 |
| NRC + SVM | 0.286 |
| TF-NRC + SVM | 0.458 |
| Doc2Vec + SVM | 0.403 |
| HRNN | 0.469 |
| BiRNN + Self-Attention | 0.487 |
| ELMo + BiRNN | 0.516 |
| Fine-tuned BERT | **0.604** |

Table 4: Benchmark results (averaged 5-fold cross validation)

each to model sentences and documents. The tokens of a sentence were processed independently of other sentence tokens. For each direction in the token-level BiLSTM, the last outputs were concatenated and fed into the sentence-level BiLSTM as inputs.

The outputs of the BiLSTM were connected to 2 dense layers with 256 ReLU units and a Softmax layer. We initialized tokens with publicly available embeddings trained with GloVe (Pennington et al., 2014). Sentence boundaries were provided by SpaCy. Dropout was applied to the dense hidden layers during training.

### 4.4 Bi-directional RNN and Self-Attention (BiRNN + Self-Attention)

One challenge with RNN-based solutions for text classification is finding the best way to combine word-level representations into higher-level representations.

Self-attention (Yang et al., 2016; Lin et al., 2017; Sinha et al., 2018) has been adapted to text classification, providing improved interpretability and performance. We used (Lin et al., 2017) as the basis of this benchmark.

The benchmark used a layered Bi-directional RNN (60 units) with GRU cells and a dense layer. Both self-attention layers were 60 units in size and cross-entropy was used as the cost function.

Note that we have omitted the orthogonal regularizer term, since this dataset is relatively small compared to the traditional datasets used for training such a model. We did not observe any significant performance gain while using the regularizer term in our experiments.

### 4.5 ELMo embedding and Bi-directional RNN (ELMo + BiRNN)

Deep Contextualized Word Representations (ELMo) (Peters et al., 2018) have shown recent success in a number of NLP tasks. The unsupervised nature of the language model allows it to utilize a large amount of available unlabelled data in order to learn better representations of words.

We used the pre-trained ELMo model (v2) available on *Tensorhub*[4] for this benchmark. We fed the word embeddings of ELMo as input into a one layer Bi-directional RNN (16 units) with GRU cells (with dropout) and a dense layer. Cross-entropy was used as the cost function.

### 4.6 Fine-tuned BERT

Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018) has achieved state-of-the-art results on several NLP tasks, including sentence classification.

We used the fine-tuning procedure outlined in the original work to adapt the pre-trained uncased $\text{BERT}_{\text{LARGE}}$[5] to a multi-class passage classification task. This technique achieved the best result among our benchmarks, with an average micro-F1 score of 60.4%.

## 5 Conclusion

We introduce DENS, a dataset for multi-class emotion analysis from long-form narratives in English. We provide a number of benchmark results based on models ranging from bag-of-word models to methods based on pre-trained language models (ELMo and BERT).

Our benchmark results demonstrate that this dataset provides a novel challenge in emotion analysis. The results also demonstrate that attention-based models could significantly improve performance on classification tasks such as emotion analysis.

Interesting future directions for this work include: 1. incorporating common-sense knowledge into emotion analysis to capture semantic context and 2. using few-shot learning to bootstrap and improve performance of underrepresented emotions.

Finally, as narrative passages often involve interactions between multiple emotions, one avenue for future datasets could be to focus on the multi-emotion complexities of human language and their contextual interactions.

[4] https://tfhub.dev/google/elmo/2

[5] https://tfhub.dev/google/bert_uncased_L-24_H-1024_A-16/1

## References


Cecilia Alm, Dan Roth, and Richard Sproat. 2005. Emotions from text: Machine learning for text-based emotion prediction.

Oscar Araque, Lorenzo Gatti, Jacopo Staiano, and Marco Guerini. 2018. Depechemood++: a bilingual emotion lexicon built through simple yet powerful techniques.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. 2005. Bidirectional lstm networks for improved phoneme classification and recognition. In *International Conference on Artificial Neural Networks*, pages 799–804. Springer.

Neel Kant, Raul Puri, Nikolai Yakovenko, and Bryan Catanzaro. 2018. Practical text classification with large pre-trained language models. *arXiv preprint arXiv:1812.01207*.

Evgeny Kim and Roman Klinger. 2018. Who feels what and why? annotation of a literature corpus with semantic roles of emotions. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1345–1359. Association for Computational Linguistics.

Roman Klinger, Orphee De Clercq, Saif Mohammad, and Alexandra Balahur. 2018. Iest: Wassa-2018 implicit emotions shared task. In *Proceedings of the 9th Workshop on Computational Approaches to*

*Subjectivity, Sentiment and Social Media Analysis*, pages 31–42, Brussels, Belgium. Association for Computational Linguistics.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *CoRR*, abs/1405.4053.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding.

Julian McAuley, Rahul Pandey, and Jure Leskovec. 2015. Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 785–794. ACM.

Saif M. Mohammad. 2012. #emotional tweets. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 246–255, Montréal, Canada. Association for Computational Linguistics.

Saif M. Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. 2018. Semeval-2018 task 1: Affect in tweets. In *Proceedings of the International Workshop on Semantic Evaluation (SemEval-2018)*. Association for Computational Linguistics.

Saif M. Mohammad and Svetlana Kiritchenko. 2015. Using hashtags to capture fine emotion categories from tweets. *Computational Intelligence*, 31(2):301–326.

Saif M. Mohammad and Peter D. Turney. 2010. Emotions evoked by common words and phrases: Using mechanical turk to create an emotion lexicon. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, CAAGET '10, pages 26–34, Stroudsburg, PA, USA. Association for Computational Linguistics.

Saif M. Mohammad and Peter D. Turney. 2013. Crowdsourcing a word-emotion association lexicon. 29(3):436–465.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Robert Plutchik. 1979. *Emotions: A general psychoevolutionary theory*, volume 1. Psychology Press, Taylor and Francis Group.

Andrew J. Reagan, Lewis Mitchell, Dilan Kiley, Christopher M. Danforth, and Peter Sheridan Dodds. 2016. The emotional arcs of stories are dominated by six basic shapes. *EPJ Data Science*, 5:31.

Koustuv Sinha, Yue Dong, Jackie Chi Kit Cheung, and Derek Ruths. 2018. A hierarchical neural attention-based text classifier. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 817–823. Association for Computational Linguistics.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 553–562. ACM.

Wikimedia. 2011. Robert plutchik's wheel of emotions. File: Plutchik-wheel.svg.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489. Association for Computational Linguistics.

# Multi-Task Stance Detection with Sentiment and Stance Lexicons

**Yingjie Li**
Computer Science
University of Illinois at Chicago
Chicago, IL, USA
yli300@uic.edu

**Cornelia Caragea**
Computer Science
University of Illinois at Chicago
Chicago, IL, USA
cornelia@uic.edu

## Abstract

Stance detection aims to detect whether the opinion holder is in support of or against a given target. Recent works show improvements in stance detection by using either the attention mechanism or sentiment information. In this paper, we propose a multi-task framework that incorporates target-specific attention mechanism and at the same time takes sentiment classification as an auxiliary task. Moreover, we used a sentiment lexicon and constructed a stance lexicon to provide guidance for the attention layer. Experimental results show that the proposed model significantly outperforms state-of-the-art deep learning methods on the SemEval-2016 dataset.

## 1 Introduction

With the rapid growth of social media, user opinions towards various targets, e.g., politicians and religion, are abundant. These opinions can help optimize management systems and can gain insight into important events, e.g., presidential elections. The stance detection task aims to determine whether people are in favor of, against, or neutral towards a specific target. This task is similar to the three-way aspect-level sentiment analysis that determines sentiment polarity towards aspect terms. However, different from aspect-level sentiment analysis, the target in stance detection might not be explicitly mentioned in a given sentence. Consider the following example tweet: *@realDonaldTrump is the only honest voice of the @GOP and that should scare the shit out of everyone! #SemST.* Target: *Hillary Clinton*; Stance: *Against*; Sentiment: *Positive*. Observe that even though target Hillary Clinton does not appear in this tweet, we can still infer that opinion holder is less likely to be in favor of Hillary Clinton. Therefore, identifying target related information is of vital importance to stance detection.

Previous studies to stance detection used feature engineering (Mohammad et al., 2016b), Convolutional Neural Networks (CNNs) (Vijayaraghavan et al., 2016; Wei et al., 2016) and Recurrent Neural Networks (RNNs) (Zarrella and Marsh, 2016). However, they failed to take target information into considerations. In order to address this issue, several target-specific attention mechanisms (Du et al., 2017; Zhou et al., 2017; Sun et al., 2018) have been proposed to embed target information into sentence representations.

Sentiment information has also been proven to be beneficial to stance detection. For example, Sobhani et al. (2016) found that sentiment lexicon features are useful for stance detection when combined with other features. Instead of directly integrating sentiment features into vector representations, Sun et al. (2018) proposed a hierarchical attention network that learns the importance of sentiment information automatically. Later, Sun et al. (2019) proposed a joint model that determines stance and sentiment simultaneously. However, the attention mechanism is not considered in this model.

Motivated by recent advances in multi-task learning (Liu et al., 2016, 2017; Balikas et al., 2017; Yu et al., 2018; Cohan et al., 2019) and the effectiveness of sentiment information in stance detection (Sobhani et al., 2016; Sun et al., 2018, 2019), we propose an attention-based multi-task framework that takes sentiment classification as an auxiliary task for stance detection. Specifically, we first encode words of input sentences in fixed-length dense vectors (Mikolov et al., 2013; Bojanowski et al., 2017) and then feed them as input to Bidirectional Long Short-Term Memory networks (BiLSTM) (Hochreiter and Schmidhuber, 1997; Schuster and Paliwal, 1997). After that, the attention mechanism (Bahdanau et al., 2015) is used to extract important words for sentiment



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6299–6305,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

representation. Besides, a target-specific attention layer is designed to identify important words related to a given target. The main task incorporates sentiment information by concatenating the outputs of the two attention layers. Moreover, in order to provide guidance to the attention mechanism, sentiment and stance lexicon features are integrated into the final loss. We show that the proposed model, called AT-JSS-Lex, achieves competitive results on the SemEval-2016 dataset (Mohammad et al., 2016a,b, 2017).

Our contributions are summarized as follows: First, we propose a joint sentiment and stance model (AT-JSS-Lex) based on multi-task learning that improves stance detection with the help of sentiment information and integrates both sentiment attention and target-specific attention. Second, we propose a novel formulation of the loss function that uses an existing sentiment lexicon and a stance lexicon, that we specifically constructed for this task, to guide the attention mechanism. As part of our contributions, we will make the stance lexicon publicly available. Finally, we show that the proposed AT-JSS-Lex model achieves remarkable improvements in performance over strong baselines and prior works on the SemEval-2016 stance detection dataset.

## 2 Model

### 2.1 Multi-Task Learning Architecture

The overall architecture of the proposed model is shown in Figure 1. For sentiment classification (the auxiliary task), the input sentence ($S = s_1, s_2, ..., s_n$) is first sent to an embedding layer and each word is represented by a dense vector ($X = x_1^{d_1}, x_2^{d_1}, ..., x_n^{d_1}$), where $n$ is the sentence length and $d_1$ is the dimension of the word embedding. Then, a BiLSTM is used for feature extraction. At time step $i$, the hidden vector of forward LSTM is calculated based on previous hidden vector $h_{i-1}$ and current input vector $x_i$, $\overrightarrow{h_i} = LSTM(x_i, \overrightarrow{h_{i-1}})$. The hidden vector $\overleftarrow{h_i}$ of backward LSTM is defined in a similar way. After concatenating the two hidden vectors, we obtain $h_i = [\overrightarrow{h_i}; \overleftarrow{h_i}]$.

We adopt the attention mechanism (Bahdanau et al., 2015) to improve the three-class sentiment classification. Attention weight $\alpha_i^1$ is computed as:

$$\alpha_i^1 = \frac{exp(e_i)}{\sum_{j=1}^{n} exp(e_j)} \quad (1)$$

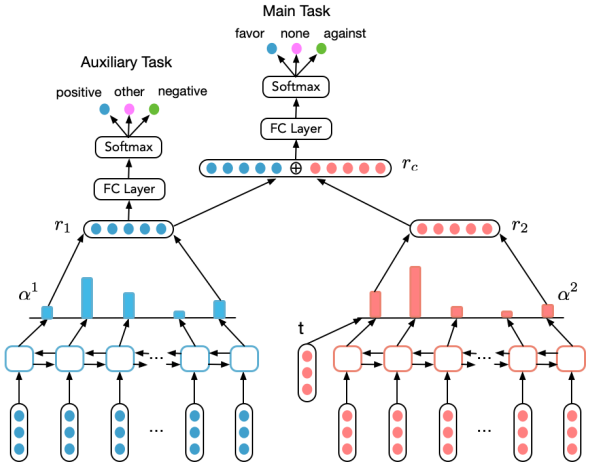


Figure 1: Our multi-task model for stance detection.

where $e_i$ is calculated based on sentence summary vector $s$, the final hidden vector of BiLSTM, and hidden vector $h_i$:

$$e_i = v_s^T tanh(W_s s + W_i h_i + b_s) \quad (2)$$

where $W_s$, $W_i \in \mathbb{R}^{2d_2 \times 2d_2}$ and $v_s$, $b_s \in \mathbb{R}^{2d_2}$. $d_2$ is the dimension of hidden units of LSTM. The final vector representation is the weighted sum of hidden vectors:

$$r_1 = \sum_{i=1}^{n} \alpha_i^1 h_i \quad (3)$$

At last, a fully-connected layer and a softmax layer are applied to get label distribution.

Different from the attention layer of auxiliary task, the attention layer of main task integrates target embedding $t$, similar to (Zhou et al., 2017). The target embedding $t$ is the word embedding of the target word. For example, for target "Abortion," $t$ is the word vector of the word "Abortion." For multi-word targets (e.g., Hillary Clinton), we use the average of the constituent word vectors (e.g., the average of the vectors corresponding to "Hillary" and "Clinton"). Then $e_i^{'}$ can be written as:

$$e_i^{'} = v_t^T tanh(W_t t + W_i^{'} h_i + b_t) \quad (4)$$

where $W_t \in \mathbb{R}^{2d_2 \times d_1}$, $W_i^{'} \in \mathbb{R}^{2d_2 \times 2d_2}$, $t \in \mathbb{R}^{d_1}$, $v_t$ and $b_t \in \mathbb{R}^{2d_2}$. Then, $\alpha^2$ and $r_2$ are computed in the same way as Eq. (1) and Eq. (3), respectively. The final vector representation of main task is the concatenation of $r_1$ and $r_2$.

In the training stage, cross-entropy loss is used to train the model and the loss function $L$ is defined as follows:

$$L = \lambda L_{main} + (1 - \lambda) L_{aux} \quad (5)$$

| Target | #Train | %Favor | %Against | %None | #Test | %Favor | %Against | %None |
|---|---|---|---|---|---|---|---|---|
| **Atheism** | 513 | 17.9 | 59.3 | 22.8 | 220 | 14.5 | 72.7 | 12.7 |
| **Climate** | 395 | 53.7 | 3.8 | 42.5 | 169 | 72.8 | 6.5 | 20.7 |
| **Feminism** | 664 | 31.6 | 49.4 | 19.0 | 285 | 20.4 | 64.2 | 15.4 |
| **Hillary** | 689 | 17.1 | 57.0 | 25.8 | 295 | 15.3 | 58.3 | 26.4 |
| **Abortion** | 653 | 18.5 | 54.4 | 27.1 | 280 | 16.4 | 67.5 | 16.1 |
| **Total** | 2914 | 25.8 | 47.9 | 26.3 | 1249 | 23.1 | 51.8 | 25.1 |

Table 1: Data distribution of SemEval-2016 dataset.

where $\lambda$ is a hyper-parameter to be tuned, determining the weight of stance detection task. $L_{main}$ and $L_{aux}$ are loss functions of main task and auxiliary task, respectively.

### 2.2 Lexicon Loss

Previous works showed that the attention mechanism is beneficial for stance detection. However, the attention mechanism does not always work well due to the size of training data and inability to identify target information. In order to address these issues, we propose a reformulation of the loss in Eq. (5) that employs both sentiment and stance lexicons to improve stance detection. Our full model is called AT-JSS-Lex.

We use the sentiment lexicon[1] by Hu and Liu (2004) and construct a stance lexicon[2] (of almost 2,000 words) with words related to the five targets in the SemEval-2016 dataset. Specifically, we construct a stance lexicon for each target from the training data available from the SemEval 2016 dataset and from an extra 1,000 tweets for each target that we collected from Twitter using specific hashtags. For example, for the target "Hillary Clinton," hashtags such as "#Crooked-Hillary," "#hillaryforprison," and "#Hillary2016" are used to collect more tweets. After data collection, we manually extracted the directly related words (e.g., Killary, Shillary, and Hilly) and indirectly related words (e.g., Trump, Benghazi, and abortion) from each tweet. We ended up with around 400 lexicon words for each target.

The intuition of using sentiment and stance lexicons is to provide guidance to the attention layer. Specifically, given an input sentence, we mark sentiment and stance lexicon words as 1 and mark as 0 the remaining words that are not present in any of the two lexicons in order to obtain a lexicon vector $lex$. For example, the lexicon vector for "Celebrity atheism is beginning to irk me #initforthemoney #SemST" is [0 1 0 0 0 1 0 0 0]. Note that the words "atheism" and "irk" are from the stance and sentiment lexicons, respectively. The final loss from Eq. (5) is then defined as:

$$\begin{aligned} L =& \lambda L_{main} + (1-\lambda) L_{aux} \\ &+ \beta(\alpha_{norm} - lex)^2 \end{aligned} \tag{6}$$

where $\beta$ is a hyperparameter that determines the importance of lexicon loss and $\alpha_{norm}$ is the normalization of $\alpha_{sum}$. $\alpha_{sum}$ is the summation of attention weights $\alpha^1$ and $\alpha^2$. We normalize the $\alpha_{sum}$ by dividing with the maximum value. Through the minimization of the loss function, ideally, $\alpha_{norm}$ gets closer to $lex$, i.e., the vector components of $\alpha_{norm}$ get closer to 1 when the corresponding components in $lex$ are 1 (and closer to 0 otherwise), which enforces the model to learn higher attention weights for important words.

## 3 Experiments and Analysis

### 3.1 Datasets and Experimental Settings

We use SemEval-2016 Task 6.A to test the performance of our proposed model. This dataset contains five different targets: "Atheism," "Climate Change is a Real Concern" ("Climate"), "Feminist Movement" ("Feminism"), "Hillary Clinton" ("Hillary") and "Legalization of Abortion" ("Abortion"). Table 1 shows the distribution of these targets in the dataset. Each tweet has a stance label ("Favor," "Against" or "None") and a sentiment label ("Positive," "Negative" or "Other").

We sampled about 15% of the training data as validation data to tune the parameters. Word vectors are initialized using fastText word embeddings[3] (Bojanowski et al., 2017) with dimension 300. The size of hidden units in LSTM is set to 200. The dropout rate is 0.5 for dense layer and 0.2 for LSTM layer. The learning rate of Adam optimizer (Kingma and Ba, 2015) is 0.001. The size of the dense layer for the auxiliary task and

[1] https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html#lexicon

[2] https://github.com/chuchun8/EMNLP19-Stance

[3] https://github.com/facebookresearch/fastText/blob/master/docs/crawl-vectors.md

| Model | Atheism | Climate | Feminism | Hillary | Abortion | $MacF_{avg}$ | $MicF_{avg}$ |
|---|---|---|---|---|---|---|---|
| | $F_{avg}$ | $F_{avg}$ | $F_{avg}$ | $F_{avg}$ | $F_{avg}$ | | |
| **AT-JSS-Lex** | 69.22 | **59.18** | **61.49** | **68.33** | **68.41** | **65.33** | **72.33** |
| **AT-JSS** | **70.81** | 57.71 | 60.14 | 67.06 | 67.16 | 64.58 | 72.06 |
| **JSS** | 66.87 | 54.28 | 58.35 | 64.74 | 62.48 | 61.34 | 68.96 |
| **AT-BiLSTM** | 66.31 | 52.56 | 59.50 | 64.51 | 66.10 | 61.80 | 71.86 |
| **BiLSTM** | 64.88 | 43.68 | 57.93 | 58.81 | 60.86 | 57.23 | 68.44 |
| **SVM-ngram** | 65.19 | 42.35 | 57.46 | 58.63 | 66.42 | 58.01 | 68.98 |
| **JOINT** | 66.78 | 50.60 | 59.35 | 62.47 | 61.58 | 60.16 | 69.22 |
| **TAN** | 59.33 | 53.59 | 55.77 | 65.38 | 63.72 | 59.56 | 68.79 |
| **AS-biGRU-CNN** | 66.76 | 43.40 | 58.83 | 57.12 | 65.45 | 58.31 | 69.42 |
| **HAN** | 70.53 | 49.56 | 57.50 | 61.23 | 66.16 | 61.00 | 69.79 |
| **TGMN-CR** | 64.60 | 43.02 | 59.35 | 66.21 | 66.21 | 59.88 | 71.04 |

Table 2: Performance comparison of stance detection on the SemEval-2016 dataset. The numbers are percentages.

main task is 50 and 100, respectively. $\lambda$ is 0.7 and $\beta$ is 0.025 for all targets. L2 regularization is applied to the loss function and the regularization parameter is set to 0.01.

### 3.2 Evaluation Metrics

$F_{avg}$, macro-average of F1 score ($MacF_{avg}$) and micro-average of F1 score ($MicF_{avg}$) are adopted to evaluate the performance of proposed model. Firstly, the F1 score of label "Favor" and "Against" is calculated as follows:

$$F_{favor} = \frac{2P_{favor}R_{favor}}{P_{favor} + R_{favor}} \quad (7)$$

$$F_{against} = \frac{2P_{against}R_{against}}{P_{against} + R_{against}} \quad (8)$$

where P and R are precision and recall respectively. Then the F1 average is calculated as:

$$F_{avg} = \frac{F_{favor} + F_{against}}{2} \quad (9)$$

Note that the label "None" is not discarded during training. However, the label "None" is not considered in the evaluation because we are only interested in labels "Favor" and "Against" in this task.

We average the $F_{avg}$ on each target to get $MacF_{avg}$. Moreover, we get $MicF_{avg}$ by calculating $F_{favor}$ and $F_{against}$ across all targets.

### 3.3 Results

First, an ablation experiment is used to determine the importance of each component of our proposed model for stance detection:

- AT-JSS-Lex is a lexicon integrated multi-task model with attention mechanisms.
- AT-JSS is a model that shares the same architecture with AT-JSS-Lex, but has no lexicon loss.
- JSS is a joint sentiment and stance model similar to AT-JSS, but without the attention mechanisms.
- AT-BiLSTM is a BiLSTM model with stance attention.
- BiLSTM is a single task model that only exploits BiLSTM to detect stance.

Table 2 (top) shows the results of this ablation study. As we can see from the table, AT-JSS-Lex performs best on "Climate," "Feminism," "Hillary," and "Abortion." Moreover, AT-JSS-Lex has the best $MacF_{avg}$ and $MicF_{avg}$ scores when compared with the other models. Experimental results show that $MacF_{avg}$ and $MicF_{avg}$ drop by 0.75% and 0.27% when we remove the lexicon component, indicating that lexicon information contributes to stance detection (except on "Atheism"). Note that AT-JSS-Lex also outperforms AT-BiLSTM model, which shows that the proposed multi-task framework has better performance than single task with attention mechanism.

Second, we compare the proposed model with the following baseline methods (all experimental results of baseline methods are retrieved from original papers):

- SVM-ngram (Mohammad et al., 2016b) is trained by using word n-grams and character n-grams features, surpassing the best model in SemEval-2016 competition.
- JOINT (Sun et al., 2019) is a joint model that exploits sentiment information to improve stance detection task without attention mechanism.
- TAN (Du et al., 2017) is an attention-based LSTM model that extracts important part of given text.

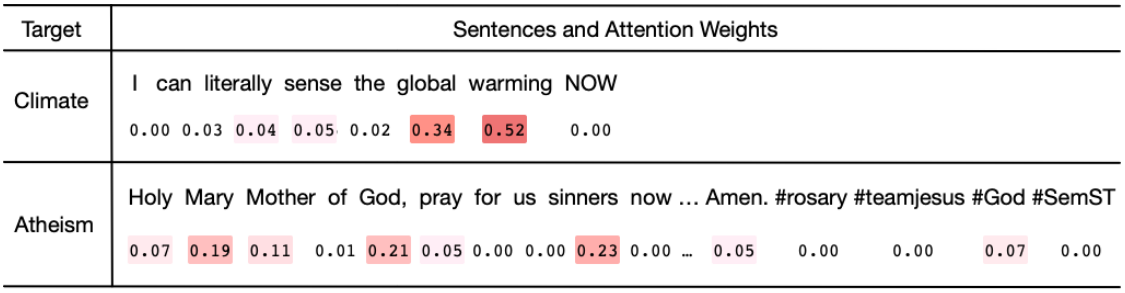


Figure 2: Visualization of learned attention weights.

- AS-biGRU-CNN (Zhou et al., 2017) is another attention-based model that adds CNN layer after attention-based LSTM model to extract target specific features.
- HAN (Sun et al., 2018) is a hierarchical attention model leveraging various linguistic features.
- TGMN-CR (Wei et al., 2018) uses attention and memory modules to extract important information for detecting stance.

Table 2 (bottom) shows the results of this comparison as well. We can observe that AT-JSS-Lex model outperforms all baseline models except on "Atheism." Specifically, the proposed model outperforms JOINT model by 5.17% and 3.11% in $MacF_{avg}$ and $MicF_{avg}$, demonstrating the effectiveness of attention mechanism and lexicon information. In addition, our AT-JSS-Lex model also performs better than attention-based models (TAN, HAN, AS-biGRU-CNN and TGMN-CR), showing that multi-task learning can benefit the stance detection task.

### 3.4 Attention Visualizations

In Figure 2, we list two input sentences and visualize the attention weights learned during the training process. The color indicates the importance of a word in given sentences, the darker the more important. We can observe that the words "global" and "warming" that are closely related to the target "Climate," have high attention weights in the first sentence. Likewise, in the second sentence, the words "Mary," "Mother," "God," and "sinners," are highlighted and show that the proposed model can pay attention to target-related words.

### 3.5 Error Analysis

Mislabeled data and compound hashtags are two challenging factors that increase the difficulty of further improving the stance classification. For example, consider the following tweet: *Watch @Dame_Lillard bring the Blazers to the playoff #Beast #SemST.* Target: *Atheism*; Stance: *Against*; Sentiment: *Positive*. Even though the stance label of this tweet is "Against," however, we can observe that the content of this sentence is nothing related to the target.

For SemEval 2016 dataset, sometimes it is hard to predict the stance without considering the compound hashtags. Here is an example: *@Reince This is very credible! Good work! America is desperately in need of good leadership. #VoteGOP #NoHillary #SemST.* Target: *Hillary Clinton*; Stance: *Against*; Sentiment: *Positive*. It would be very difficult to infer the correct stance label if we do not consider the hashtags "#NoHillary" and "#VoteGOP." Therefore, inability to separate compound hashtags results in the loss of important target information.

## 4 Conclusion and Future Work

In this paper, we propose an attention-based multi-task learning framework and integrate lexicon information to achieve better performance. Experimental results show that our model outperforms state-of-the-art deep learning methods for this task. Moreover, visualization results indicate the capability of our model to capture essential information. Future work includes exploiting unsupervised learning to generate target-related lexicon and incorporating more labels (e.g., emotion classification) for multi-task learning.

## Acknowledgments

We thank our reviewers for their valuable comments and feedback that helped improve the quality of our paper. This research is supported by NSF awards 1541155, 1741353 and 1526542 to Cornelia Caragea. The computation for this project was performed on Amazon Web Services.

## References


Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations*.

Georgios Balikas, Simon Moura, and Massih-Reza Amini. 2017. Multitask learning for fine-grained twitter sentiment analysis. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*, pages 1005–1008.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Arman Cohan, Waleed Ammar, Madeleine van Zuylen, and Field Cady. 2019. Structural scaffolds for citation intent classification in scientific publications. *arXiv preprint arXiv:1904.01608*.

Jiachen Du, Ruifeng Xu, Yulan He, and Lin Gui. 2017. Stance classification with target-specific neural attention networks. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 3988–3994.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 2873–2879.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1–10.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiao-Dan Zhu, and Colin Cherry. 2016a. A dataset for detecting stance in tweets. In *LREC*.

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016b. Semeval-2016 task 6: Detecting stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 31–41.

Saif M Mohammad, Parinaz Sobhani, and Svetlana Kiritchenko. 2017. Stance and sentiment in tweets. *ACM Transactions on Internet Technology (TOIT)*, 17(3):26.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Parinaz Sobhani, Saif Mohammad, and Svetlana Kiritchenko. 2016. Detecting stance in tweets and analyzing its interaction with sentiment. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*, pages 159–169.

Qingying Sun, Zhongqing Wang, Shoushan Li, Qiaoming Zhu, and Guodong Zhou. 2019. Stance detection via sentiment information and neural network model. *Frontiers of Computer Science*, 13(1):127–138.

Qingying Sun, Zhongqing Wang, Qiaoming Zhu, and Guodong Zhou. 2018. Stance detection with hierarchical attention network. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2399–2409.

Prashanth Vijayaraghavan, Ivan Sysoev, Soroush Vosoughi, and Deb Roy. 2016. DeepStance at SemEval-2016 task 6: Detecting stance in tweets using character and word-level CNNs. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 413–419.

Penghui Wei, Wenji Mao, and Daniel Zeng. 2018. A target-guided neural memory model for stance detection in twitter. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8.

Wan Wei, Xiao Zhang, Xuqin Liu, Wei Chen, and Tengjiao Wang. 2016. pkudblab at semeval-2016 task 6: A specific convolutional neural network system for effective stance detection. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 384–388.

Jianfei Yu, Luis Marujo, Jing Jiang, Pradeep Karuturi, and William Brendel. 2018. Improving multi-label emotion classification via sentiment classification with dual attention transfer network. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1097–1102.

Guido Zarrella and Amy Marsh. 2016. MITRE at SemEval-2016 task 6: Transfer learning for stance detection. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 458–463.

Yiwei Zhou, Alexandra I Cristea, and Lei Shi. 2017. Connecting targets to tweets: Semantic attention-based model for target-specific stance detection. In *International Conference on Web Information Systems Engineering*, pages 18–32.

# A Robust Self-Learning Framework for Cross-Lingual Text Classification

**Xin Dong**
Rutgers University
New Brunswick, NJ, USA
xd48@rutgers.edu

**Gerard de Melo**
Rutgers University
New Brunswick, NJ, USA
gdm@demelo.org

## Abstract

Based on massive amounts of data, recent pretrained contextual representation models have made significant strides in advancing a number of different English NLP tasks. However, for other languages, relevant training data may be lacking, while state-of-the-art deep learning methods are known to be data-hungry. In this paper, we present an elegantly simple robust self-learning framework to include unlabeled non-English samples in the fine-tuning process of pretrained multilingual representation models. We leverage a multilingual model's own predictions on unlabeled non-English data in order to obtain additional information that can be used during further fine-tuning. Compared with original multilingual models and other cross-lingual classification models, we observe significant gains in effectiveness on document and sentiment classification for a range of diverse languages.

## 1 Introduction

Owing to notable advances in deep learning and representation learning, important progress has been achieved on text classification, reading comprehension, and other NLP tasks. Recently, pretrained language representations with self-supervised objectives (Peters et al., 2018; Devlin et al., 2018; Radford et al., 2018) have further pushed forward the state-of-the-art on many English tasks. While these sorts of deep models can be trained on different languages, deep models typically require substantial amounts of labeled data for the specific domain of data.

Unfortunately, the cost of acquiring new custom-built resources for each combination of language and domain is very high, as it typically requires human annotation. Available resources for domain-specific tasks are often imbalanced between different languages. The scarcity of non-English annotated corpora may preclude our ability to train language-specific machine learning models. In contrast, English-language annotations are often readily available to train deep models. Although translation can be an option, human translation is very costly and for many language pairs, any available domain-specific parallel corpora are too small to train high-quality machine translation systems.

Cross-lingual systems rely on training data from one language to train a model that can be applied to other languages (de Melo and Siersdorfer, 2007), alleviating the training bottleneck issues for low-resource languages. This is facilitated by recent advances in learning joint multilingual representations (Lample and Conneau, 2019; Artetxe and Schwenk, 2018; Devlin et al., 2018).

In our work, we propose a self-learning framework to incorporate the predictions of the multilingual BERT model (Devlin et al., 2018) on non-English data into an English training procedure. The initial multilingual BERT model was simultaneously pretrained on 104 languages, and has shown to perform well for cross-lingual transfer of natural language tasks (Wu and Dredze, 2019). Our model begins by learning just from available English samples, but then makes predictions on unlabeled non-English samples and a part of those samples with high confidence prediction scores are repurposed to serve as labeled examples for a next iteration of fine-tuning until the model converges.

Based on this multilingual self-learning technique, we demonstrate the superiority of our framework on Multilingual Document Classification (MLDoc) (Schwenk and Li, 2018) in comparison with several strong baselines. Our study then proceeds to show that our method is better on Chinese sentiment classification than other cross-lingual methods that also consider unla-



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6306–6310,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

beled non-English data. This shows that our method is more effective at cross-lingual transfer for domain-specific tasks, using a mix of labeled and unlabeled data via a multilingual BERT sentence model.

## 2 Method

Our proposed framework consists of three parts as shown in Figure 1. The first part is the pretrained multilingual encoder, denoted as $f_n(\cdot;\theta_n)$. The encoder is assumed to have been pretrained across different languages with appropriate strategies, such as WordPiece, Byte Pair Encoding modeling, and Cross-lingual Word Alignment, to allow the model to share representations across languages to a certain degree. Hence, we obtain a universal sentence representation $\mathbf{h} \in \mathbb{R}^d$ from this encoder, where $d$ is the dimensionality of the sentence representation. Subsequently, a task-specific classification module $f_{\text{cl}}(\cdot;\theta_{\text{cl}})$ is applied for fine-tuning on top of the pretrained model $f_n(\cdot;\theta_n)$. This module consists of a linear function mapping $\mathbf{h} \in \mathbb{R}^d$ into $\mathbb{R}^{|\mathcal{Y}|}$ and a softmax function, where $\mathcal{Y}$ is the set of target classes.

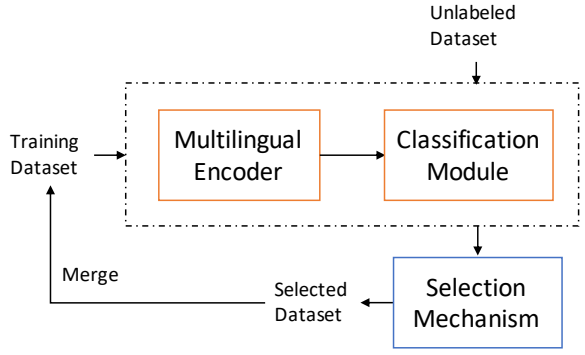


Figure 1: Illustration of self-learning process for cross-lingual classification.

For the overall process, we first train the whole network $f(\cdot;\theta)$ in $K$ epochs using a set of the labeled data $L = \{(x_i, y_i) \mid i = 1, ..., n\}$, where $n$ is the number of labeled instances, $x_i \in \mathcal{X}$ are instances, and $y_i \in \mathcal{Y}$ are the corresponding ground truth labels. The next step is to make predictions for the unlabeled instances in $U = \{x_u \mid u = 1, ..., m\}$. We assume that $f(\cdot;\theta)$ yields a class label as well as a confidence score.

To better take advantage of the pretrained multilingual model, a selection mechanism is invoked to repurpose unlabeled data with high confidence scores for incorporation into the training data. There are several variations of such selection mechanisms (Abney, 2007), and we rely on a *balancing* approach that considers the same number of instances for each class. Thus, we select a subset $\{x_s \mid s = 1, ..., K_t\}$ of the unlabeled data for each class, containing the top $K_t$ highest confidence items based on the current trained model. The union set $U_{\text{s}}$ of selected items is merged into the training set $L$ and then we retrain the model and repeat this process iteratively. The detailed process is described in Algorithm 1, where for each class $y \in \mathcal{Y}$, $\mathcal{D}_y$ denotes the set of tuples pairing unlabeled data with corresponding confidence scores $c$.

**Algorithm 1** Self-learning on cross-lingual tasks

**repeat**
  Fine-tune $f(\cdot;\theta)$ for $K$ epochs using $L$
  **for** $y \in \mathcal{Y}$ **do**
    $\mathcal{D}_y \leftarrow \emptyset$
  **for** $x_{\text{u}} \in U$ **do**
    $(y, c) \leftarrow f(x_u;\theta)$
    $\mathcal{D}_y \leftarrow \mathcal{D}_y \cup \{(x_{\text{u}}, c)\}$
  **for** $y \in \mathcal{Y}$ **do**
    $S_y \leftarrow \underset{S \subseteq \mathcal{D}_y,\ |S| \leq K_t}{\text{argmax}} \sum_{(x_{\text{u}}, c) \in S} c$
    $U \leftarrow U \setminus S_y$
    $U_{\text{s}} \leftarrow U_{\text{s}} \cup S_y$
  $L \leftarrow L \cup U_{\text{s}}$
**until** stopping criterion is true

## 3 Experiments

We evaluate our self-learning framework on two cross-lingual document and sentiment classification tasks to show the effectiveness of self-learning for multilingual BERT-based cross-lingual transfer.

### 3.1 Experimental Setup

**Datasets.** For evaluation, we first rely on MLDoc (Schwenk and Li, 2018), a balanced subset of the Reuters corpus covering 8 languages for document classification, with 1,000 training and validation documents and 4,000 test documents for each language. The 4-way topic classification scheme consists of CCAT (Corporate/Industrial), ECAT (Economics), GCAT (Government/Social), and MCAT (Markets) as targets. For cross-lingual classification, 1,000 target language training documents are used as unlabeled data for self-learning.

Table 1: Accuracy (in %) on MLDoc experiments. Bold denotes the best on cross-lingual transfer.

| Approach | *en* | *de* | *zh* | *es* | *fr* | *it* | *ja* | *ru* |
|---|---|---|---|---|---|---|---|---|
| *In-language supervised learning* | | | | | | | | |
| Schwenk and Li (2018) | 92.2 | 93.7 | 87.3 | 94.5 | 92.1 | 85.6 | 85.4 | 85.7 |
| BERT (Wu and Dredze, 2019) | 94.2 | 93.3 | 89.3 | 95.7 | 93.4 | 88.0 | 88.4 | 87.5 |
| *Cross-lingual transfer* | | | | | | | | |
| Schwenk and Li (2018) | 92.2 | 81.2 | 74.7 | 72.5 | 72.4 | 69.4 | 67.6 | 60.8 |
| Artetxe and Schwenk (2018) | 89.9 | 84.8 | 71.9 | 77.3 | 78.0 | 69.4 | 60.3 | 67.8 |
| BERT | 94.2 | 78.9 | 75.4 | 79.5 | 80.0 | 68.7 | 73.2 | 70.7 |
| Our Approach | 94.2 | **90.0** | **87.0** | **85.3** | **88.4** | **75.2** | **76.8** | **79.3** |

Table 2: Accuracy (in %) on Chinese sentiment classification without using labeled Chinese data. CLD and CLTC represent cross-lingual distillation and cross-lingual text classification.

| Approach | ACC |
|---|---|
| *Domain Adaptation* | |
| mSDA (Chen et al., 2012) | 31.44 |
| *Machine Translation* | |
| DAN + MT (Chen et al., 2018) | 39.66 |
| *CLD-based CLTC* | |
| CLD-KCNN (Xu and Yang, 2017) | 40.96 |
| CLDFA-KCNN (Xu and Yang, 2017) | 41.82 |
| *Cross-lingual Adversarial* | |
| ADAN (Chen et al., 2018) | 42.49 |
| *Cross-lingual transfer* | |
| BERT | 40.73 |
| Our Approach | **43.88** |

We further evaluate our method on cross-lingual sentiment classification from English to Chinese. For English, we use a balanced dataset of 700k Yelp reviews from Zhang et al. (2015) with their ratings as labels (scale 1–5) and adopting their training–validation split: 650k reviews for training and 50k as a validation set. For Chinese, we use the same dataset configuration as Chen et al. (2018), consisting of 150k unlabeled Chinese hotel reviews and 10k balanced Chinese hotel reviews as a validation set. The results are reported on a separate test set of another 10k hotel reviews. The data are also annotated with 5 labels (1–5). Both classification tasks are evaluated in terms of classification accuracy (ACC).

**Model Details.** We tune the hyper-parameters for our neural network architecture based on each non-English validation set. For the encoder, we invoke the multilingual BERT model (Devlin et al., 2018), which supports 104 languages[1]. It relies on a shared 110k WordPiece vocabulary across all languages and yields sentence representations in a common multilingual space. Most model hyper-parameters are the same as in pretraining, with the exception of the batch size, max. sequence length, and number of training epochs. The batch size, max. sequence length and number of training epochs used for the MLDoc task are 128, 32, and 4, respectively, while they are 128, 96, and 3 for sentiment classification. Another hyper-parameter involved in self-learning is $K_t$, which is 40 for MLDoc and 100 for sentiment classification. We rely on early stopping as a termination criterion.

[1]2018-11-23 version from https://github.com/google-research/bert/blob/master/multilingual.md

### 3.2 Results and Analysis

**Cross-lingual Document Classification.** Three recent strong baselines are included in our MLDoc experiments. Schwenk and Li (2018) use MultiCCA, multilingual word embeddings trained with a bilingual dictionary and convolution neural networks. Artetxe and Schwenk (2018) pretrain a multilingual sentence representation with a massively multilingual sequence-to-sequence NMT model, where the encoder is used for fine-tuning downstream tasks. We also considered multilingual BERT without self-learning as one of our baselines. As shown in Table 1, our framework significantly outperforms all baselines in 7 languages on cross-lingual document classification, including for phylogenetically unrelated languages. We also show the respective percentages of instances added into the training set that are correct for the MLDoc data based on our method in Table 3. The high percentages of correctly labeled incorporated instances in 7 languages further show the effectiveness of self-learning in our framework.

Table 3: Percentages of instances added into the training set that are correct for the MLDoc data using our method.

| *de* | *zh* | *es* | *fr* | *it* | *ja* | *ru* |
|---|---|---|---|---|---|---|
| 93% | 92 % | 89 % | 92% | 80% | 84% | 89% |

**Cross-lingual Sentiment Classification.** To evaluate the robustness of our framework on cross-lingual sentiment classification, we consider several diverse baselines as listed in Table 2. mSDA (Chen et al., 2012) is a very effective method for cross-domain sentiment classification on Amazon reviews, which can also be used in cross-lingual tasks, but it has the worst performance. Deep Averaging Networks (DANs) by Iyyer et al. (2015) consider an arithmetic mean of word vectors as a sentence representation and pass it to a classification module, while Chen et al. (2018) translate the Chinese test text into English as a machine translation baseline. The third category of baselines includes Xu and Yang (2017), who propose a cross-lingual distillation (CLD) method that makes use of soft source predictions on a parallel corpus to train a target model (CLD-KCNN). They further propose an improved variant (CLDFA-KCNN) that utilizes adversarial training for domain adaptation within a single language. Adversarial DAN (ADAN) by Chen et al. (2018) is another state of the art baseline that improves cross-lingual generalization by means of adversarial training. We also run experiments on multilingual BERT and observe that it does not outperform CLD-based CLTC and ADAN, while our approach achieves the new state-of-the-art result, indicating that our self-learning method for cross-lingual transfer can be more effective than a diverse range of other approaches. In addition, we evaluate the proximity between incorrect prediction and the corresponding correct label in our sentiment task by means of mean squared error. The error of our method is 1.37, while for regular multilingual BERT it is 1.42, which also shows the superiority of our method.

**Comparison of Selection Mechanisms.** We ran experiments on two selection mechanisms. One is *balancing*, as described in Section 2. An alternative is *throttling* by selecting the top $n$ unlabeled examples without considering specific classes (Abney, 2007). Our experiments on MLDoc show that the results suffer from a rapid decline during self-learning with *throttling*. This is because selecting from all samples leads to an imbalance between different classes and due to repeated error amplification this means that samples are increasingly likely to be assigned to the majority class in each self-learning iteration.

## 4 Related Work

**Semi-supervised Learning.** There is a long history of research on semi-supervised Learning to exploit unlabeled data. Self-learning (also known as self-training) was successfully applied to NLP tasks in early work such as on word sense disambiguation (Yarowsky, 1995) and parsing (McClosky et al., 2006). In recent work, Artetxe et al. (2018) show that self-learning can iteratively improve unsupervised cross-lingual word embeddings. Clark et al. (2018) presents Cross-View Training, a new self-training algorithm that works well for neural sequence modeling. Other semi-supervised methods, such as co-training (Blum and Mitchell, 1998) and tri-training (Zhou and Li, 2005), have as well been used for sentiment analysis. Ruder and Plank (2018) propose a novel multi-task tri-training method that reduces the time and space complexity of classic tri-training for sentiment analysis. For cross-lingual sentiment analysis, Wan (2009) uses machine translation to directly convert English training data to Chinese, which provides two views for co-training. Xu and Yang (2017) propose to use soft probabilistic predictions for the documents in a label-rich language as the (induced) supervisory labels in a parallel corpus of documents, while there is no need to use parallel corpora in our work. Chen et al. (2018) propose an Adversarial Deep Averaging Network to learn invariance across languages, which is another baseline considered in our experiments.

**Cross-lingual Representation Learning.** With models such as ELMo (Peters et al., 2018), GPT-2 (Radford et al., 2018), and BERT (Devlin et al., 2018), important progress has been made in learning improved sentence representations with context-specific encodings of words via a language modeling objective. The latter two approaches both rely on Transformer encoders, but BERT is trained using masked language modeling instead of right-to-left or left-to-right language modeling. Additionally, BERT also optimizes a next sentence classification objective. Recent work has also investigated cross-lingual extensions. Devlin et al. (2018) themselves published a multilingual version of BERT, following the same model architecture and training procedure, except that the union of 104 different language editions of Wikipedia serves as the training input. Lample and Conneau (2019) incorporate parallel text into BERT's architecture by

training on a new supervised learning objective. Artetxe and Schwenk (2018) also show that the encoder from a pretrained sequence-to-sequence model can be used to produce cross-lingual sentence embeddings. All these methods are compatible with our self-learning framework, since they provide a shared sentence meaning representation across languages as needed by our approach.

## 5 Conclusion

In this work, we propose a self-learning framework for cross-lingual text classification. Based on the cross-lingual prediction ability of pretrained multilingual model, this elegantly simple framework makes the most of unlabeled text to improve cross-lingual transfer for text classification. We achieve new state-of-the-art results on cross-lingual document and sentiment classification and demonstrate that self-learning is an effective method for improved classification accuracy without target language training data.

## References


Steven Abney. 2007. *Semisupervised learning for computational linguistics*. Chapman and Hall/CRC.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018. A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. *arXiv preprint arXiv:1805.06297*.

Mikel Artetxe and Holger Schwenk. 2018. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *arXiv preprint arXiv:1812.10464*.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM.

Minmin Chen, Zhixiang Xu, Kilian Weinberger, and Fei Sha. 2012. Marginalized denoising autoencoders for domain adaptation. *arXiv preprint arXiv:1206.4683*.

Xilun Chen, Yu Sun, Ben Athiwaratkun, Claire Cardie, and Kilian Weinberger. 2018. Adversarial deep averaging networks for cross-lingual sentiment classification. *Transactions of the Association for Computational Linguistics*, 6:557–570.

Kevin Clark, Minh-Thang Luong, Christopher D Manning, and Quoc V Le. 2018. Semi-supervised sequence modeling with cross-view training. *arXiv preprint arXiv:1809.08370*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of ACL-IJCNLP 2015*, volume 1.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of NAACL-HLT 2006*.

Gerard de Melo and Stefan Siersdorfer. 2007. Multilingual text classification using ontologies. In *Proceedings of ECIR 2007*. Springer.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Sebastian Ruder and Barbara Plank. 2018. Strong baselines for neural semi-supervised learning under domain shift. *arXiv preprint arXiv:1804.09530*.

Holger Schwenk and Xian Li. 2018. A corpus for multilingual document classification in eight languages. *arXiv preprint arXiv:1805.09821*.

Xiaojun Wan. 2009. Co-training for cross-lingual sentiment classification. In *Proceedings of ACL-IJCNLP 2009*.

Shijie Wu and Mark Dredze. 2019. Beto, bentz, becas: The surprising cross-lingual effectiveness of bert. *arXiv preprint arXiv:1904.09077*.

Ruochen Xu and Yiming Yang. 2017. Cross-lingual distillation for text classification. *arXiv preprint arXiv:1705.02073*.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd annual meeting of the association for computational linguistics*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Zhi-Hua Zhou and Ming Li. 2005. Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge & Data Engineering*, (11):1529–1541.

# Learning to Flip the Sentiment of Reviews from Non-Parallel Corpora

**Canasai Kruengkrai**
Yahoo Japan Corporation
ckruengk@yahoo-corp.jp

## Abstract

Flipping sentiment while preserving sentence meaning is challenging because parallel sentences with the same content but different sentiment polarities are not always available for model learning. We introduce a method for acquiring imperfectly aligned sentences from non-parallel corpora and propose a model that learns to minimize the sentiment and content losses in a fully end-to-end manner. Our model is simple and offers well-balanced results across two domains: Yelp restaurant and Amazon product reviews.[1]

## 1 Introduction

Text style transfer is the process of editing a sentence to modify specific attributes (e.g., style, sentiment, and tense) while preserving its attribute-independent content (Shen et al., 2017; Fu et al., 2018; Li et al., 2018). In this work, we are particularly interested in sentiment modification. This task is challenging because parallel data (i.e., aligned sentences with the same content but different sentiment polarities) are not always available and they are costly to annotate. Major existing solutions that exploit non-parallel data include training discriminators to guide sentence generation (Hu et al., 2017; Shen et al., 2017; Fu et al., 2018), retrieving similar sentences from another corpus and then editing them (Guu et al., 2018; Li et al., 2018), and applying back-translation (Prabhumoye et al., 2018; Zhang et al., 2018b; Lample et al., 2019).

Achieving both accurate sentiment modification and high content preservation can be difficult. Recently, Li et al. (2018)'s framework has shown promising results. They delete sentiment-specific phrases (e.g., *"works great"*) from a given sentence, retrieve new phrases associated with the target sentiment (e.g, *"barely used"*), and recombine them to generate an output. However, their framework relies on separate modules and assumes that clear boundaries exist between the sentiment-specific phrases and the content, which is not always the case.

(a) the service was great too → the service was n't too great
(b) this one is by far the worst → this one is the best by far

Figure 1: The retrieved sentences (right) from the opposite corpora given the query sentences (left).

In this work, our objective is to develop a model that can be trained end-to-end from sentence pairs acquired from non-parallel corpora. Our approach is based on the observation that *shared* tokens between similar sentences in unaligned corpora convey the content we wish to preserve, while *non-shared* tokens more or less reflect sentiment change. Figure 1 shows the examples of the retrieved sentences from the opposite corpora given the query sentences. Although these sentence pairs are not perfectly aligned, they provide a useful signal of sentiment change and are easy to obtain. By considering the retrieved sentences as the *noisy source* sentences and the query sentences as the *target* sentences, we can extend an encoder-decoder model to learn sentiment modification using auxiliary loss functions.

Our contributions are as follows: We introduce a search-and-corrupt method for creating training data from non-parallel corpora (§3.1). We propose a model that learns to minimize the sentiment and content losses with an alignment constraint in a fully end-to-end manner (§3.2). We empirically show that our model produces more balanced results in terms of sentiment modification and content preservation than more complex models on Yelp restaurant and Amazon product reviews (§4).

[1]The code for reproducibility will be available on the author's website.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6311–6316, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

## 2 Related wok

Following a seminal work by Hu et al. (2017), researchers have developed a variety of methods for learning the structured/unstructured parts (i.e., style/content) of latent representations. Hu et al. (2017)'s method is based on the combination of semi-supervised variational autoencoders (Kingma et al., 2014) and the wake-sleep algorithm (Hinton et al., 1995). Shen et al. (2017) utilized an adversarial training method that uses a style discriminator to align the populations of transferred and real target sentences. Other approaches related to disentangled representations include learning multi-decoder/style-embedding models (Fu et al., 2018) and using a language model as the style discriminator (Yang et al., 2018). Our model only has an objective component to align sentence representations. We do not force our model to learn the disentangled representations of style and content. Our work is closely related to classical approaches for discovering parallel sentences in non-parallel corpora (Fung and Cheung, 2004; Munteanu and Marcu, 2005) and prototype-then-edit approaches (Guu et al., 2018). Li et al. (2018) introduced the notion of attribute markers, which are style-specific words/phrases for disentangling style and content in a sentence at the word level. There is also a line of work that studies other aspects of words based on emotional information (Xu et al., 2018; Zhang et al., 2018a). Here, we make no assumption on phrase boundaries between style and content. We simply exploit the local properties of two imperfectly aligned sentences.

## 3 Proposed method

### 3.1 Data preprocessing

We derive Yelp restaurant and Amazon product reviews from Li et al. (2018).[2] The original training and validation sets consist of unaligned corpora of positive and negative sentences. Given a sentence in one corpus, we retrieve a similar sentence from another corpus. We use a nearest neighbor search based on MinHash (Broder, 2000) and LSH Forest (Bawa et al., 2005), which are implemented in datasketch.[3] Unlike Li et al. (2018), we do not use pre-computed corpus statistics to delete sentiment-specific words/phrases. We simply filter out punctuation when doing indexing and querying. We do sentence retrieval only in the training/validation data preparation steps.

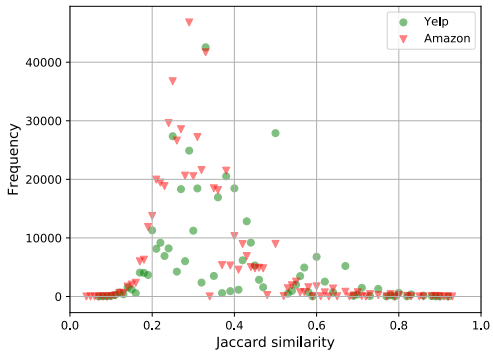


Figure 2: Distribution of Jaccard similarities between the query and retrieved sentences on the Yelp and Amazon training sets.

In the following, we use the sentence pair in Figure 1(a) as a running example. The shared tokens between the two sentences include {*"the"*, *"service"*, *"was"*, *"great"*, *"too"*}, while the token appearing only in the retrieved sentence is *"n't"*. We then consider the retrieved sentence as our noisy source sentence and the query sentence as our target sentence. Transforming the noisy source sentence to the target sentence in this example involves deleting *"n't"* and reordering *"too great"*. The sentiment flipping is obtained as a by-product of this transformation.

A practical issue is that not all the retrieved sentences are as clean as the running example. Figure 2 shows the distribution of Jaccard similarities between the query and retrieved sentences on the Yelp and Amazon training sets. Both domains have a similar characteristic in that the majority of Jaccard similarities is around 0.3. We observe that the sentence pairs with Jaccard similarities below 0.3 typically have different sentence meanings, so we filter them out. On Amazon, we also observe that the sentence pairs with high Jaccard similarities tend to have neutral sentiment. For example, given the query sentence *"my husband bought this for me as a christmas present ."* from the positive corpus, we retrieve the sentence *"i bought this as a christmas present for my husband ."* from the negative corpus with Jaccard similarity 0.82. Both sentences are neutral, which is not useful for learning sentiment flipping. Thus, on Amazon, we further filter out the sentence pairs that have Jaccard similarities exceeding 0.8.

[2] https://github.com/lijuncen/Sentiment-and-Style-Transfer

[3] https://github.com/ekzhu/datasketch

Another issue is that not all the non-shared tokens reflect sentiment change. For example, given the sentence *"very pleasant atmosphere ."*, we retrieve the sentence *"waitress was n't very pleasant ."* having Jaccard similarity 0.33. The non-shared tokens in the retrieved sentence include {*"waitress"*, *"was"*, *"n't"*}. If we directly use this retrieved sentence as the source sentence, our model must learn to delete/add many tokens to minimize the loss. Unfortunately, the tokens like *"waitress"* and *"was"* convey the content that we wish to preserve at test time, while only one token, *"n't"*, is useful for sentiment flipping. To deal with this undesired behavior, we randomly replace 20% of the non-shared tokens in the retrieved sentences having Jaccard similarities less than 0.5 with the $\langle$`mask`$\rangle$ token on both domains. In the example, random replacement of the non-shared tokens yields *"*$\langle$`mask`$\rangle$ *was n't very pleasant ."*. The sentence corruption process should alleviate overchanging the content words at test time. We also apply this technique to learn a sentence reconstructor (detailed in §3.2).

To this end, we can obtain imperfectly aligned sentences with similar content but different sentiment polarities by using our search-and-corrupt method. Assume we have the original training sets $\mathcal{X}=\{\boldsymbol{x}^{(1)},\ldots,\boldsymbol{x}^{(M)}\}$ and $\mathcal{Y}=\{\boldsymbol{y}^{(1)},\ldots,\boldsymbol{y}^{(N)}\}$. Each training set belongs to a different sentiment polarity. Let $\boldsymbol{y}'$ and $\boldsymbol{x}'$ be retrieved, corrupted versions of $\boldsymbol{y}$ and $\boldsymbol{x}$, respectively. Our unified training set becomes $\mathcal{D}=\{(\boldsymbol{y}'^{(i)},\boldsymbol{x}^{(i)})\}_{i=1}^{M}\cup\{(\boldsymbol{x}'^{(i)},\boldsymbol{y}^{(i)})\}_{i=1}^{N}$.

### 3.2 Model

Having the training set $\mathcal{D}$ in the form of noisy source and target sentences allows us to extend the attention-based encoder-decoder model (Bahdanau et al., 2015) for learning sentiment flipping. Let enc and dec be an encoder and a decoder parameterized by $\boldsymbol{\theta}_{\text{enc}}$ and $\boldsymbol{\theta}_{\text{dec}}$, respectively. Our model first contains the sentiment loss:

$$\mathcal{L}_{\text{sentiment}}(\boldsymbol{\theta}_{\text{enc}},\boldsymbol{\theta}_{\text{dec}}) = \frac{1}{M}\sum_{i=1}^{M} -\log p_{\text{dec}}(\boldsymbol{x}^{(i)}|\text{enc}(\boldsymbol{y}'^{(i)}),\boldsymbol{s}_x) + \frac{1}{N}\sum_{i=1}^{N} -\log p_{\text{dec}}(\boldsymbol{y}^{(i)}|\text{enc}(\boldsymbol{x}'^{(i)}),\boldsymbol{s}_y)]. \quad (1)$$

This loss is the combination of the standard negative log-likelihood losses. The only new components here are the two sentiment embeddings $\boldsymbol{s}_x$ and $\boldsymbol{s}_y$. In practice, we represent them with two randomly initialized vectors. We add each of them to the affine transformation of the hidden layer before applying the softmax in the decoder. These sentiment embeddings should help in shifting the probability distribution over the target tokens.

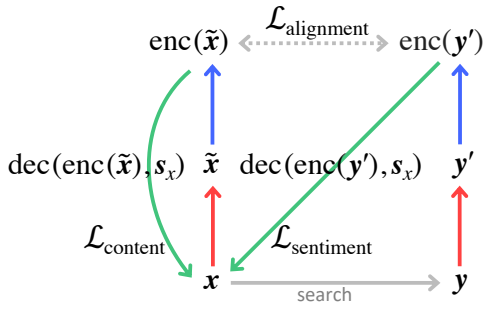


Figure 3: Workflow of the proposed method. Red arrows denote sentence corruption; blue arrows denote encoding; green arrows denote decoding.

We then randomly replace 20% of the tokens in the target sentence with the $\langle$`mask`$\rangle$ token and use the corrupted target sentence as its own source sentence. This process should guide the model to learn to reconstruct itself from its corrupted version (denoted by $\tilde{\boldsymbol{x}}$ and $\tilde{\boldsymbol{y}}$) and hopefully to retain the content. We design our content loss as follows:

$$\mathcal{L}_{\text{content}}(\boldsymbol{\theta}_{\text{enc}},\boldsymbol{\theta}_{\text{dec}}) = \frac{1}{M}\sum_{i=1}^{M} -\log p_{\text{dec}}(\boldsymbol{x}^{(i)}|\text{enc}(\tilde{\boldsymbol{x}}^{(i)}),\boldsymbol{s}_x) + \frac{1}{N}\sum_{i=1}^{N} -\log p_{\text{dec}}(\boldsymbol{y}^{(i)}|\text{enc}(\tilde{\boldsymbol{y}}^{(i)}),\boldsymbol{s}_y)]. \quad (2)$$

This loss is in the same form as Eq. (1) except that here we change the encoder input from the retrieved, corrupted sentence to the corrupted target sentence. Our idea is somewhat analogous to that of denoising autoencoders (Vincent et al., 2010; Bengio et al., 2013), in which a Gaussian noise is added to a continuous-valued input and the model learns to reconstruct a clean input.

To encourage the model to make more use of the sentiment embeddings, we use the following loss as a constraint:

$$\mathcal{L}_{\text{alignment}}(\boldsymbol{\theta}_{\text{enc}}) = \frac{1}{M}\sum_{i=1}^{M}||\text{enc}(\boldsymbol{y}'^{(i)}) - \text{enc}(\tilde{\boldsymbol{x}}^{(i)})||_2^2 + \frac{1}{N}\sum_{i=1}^{N}||\text{enc}(\boldsymbol{x}'^{(i)}) - \text{enc}(\tilde{\boldsymbol{y}}^{(i)})||_2^2. \quad (3)$$

This loss consists of the standard mean squared error (MSE) losses. Here, we want the decoder to use more information from the sentiment embeddings $\boldsymbol{s}_x$ and $\boldsymbol{s}_y$ to generate the target sentence,

if the representations of the two noisy source sentences produced by the same encoder are not considerably different from each other. Our unified objective becomes:

$$\min_{\boldsymbol{\theta}_{\text{enc}},\boldsymbol{\theta}_{\text{dec}}} \mathcal{L}_{\text{sentiment}} + \mathcal{L}_{\text{content}} + \mathcal{L}_{\text{alignment}}. \quad (4)$$

Figure 3 shows the workflow of our method given $\boldsymbol{x}$ as the target sentence. The procedure is the same when considering $\boldsymbol{y}$ as the target sentence. The sentence retrieval and corruption are done once in preprocessing.

## 4 Experiments

### 4.1 Training details

We implement our model on top of OpenNMT-py[4] (Klein et al., 2017) based on Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997). We use a single-layer bidirectional LSTM as the encoder and a single-layer unidirectional LSTM with attention (Bahdanau et al., 2015) and input-feeding (Luong et al., 2015) as the decoder. Our network configurations are identical to those of Li et al. (2018). Specifically, we use 512 hidden states and 128-dimensional word vectors.

We use Adam (Kingma and Ba, 2015) with the batch size of 256, the learning rate of 0.001, and the gradient clipping at 5. The learning rate decays by halves if the validation perplexity does not decrease. We initialize all model parameters and word embeddings by sampling from $\mathcal{U}(-0.1, 0.1)$. We train for 20 epochs or until the validation perplexity does not decrease for two epochs. At test time, we use the beam size of 5. We use the same network and parameter configurations for all experiments. We conduct all experiments on NVIDIA Tesla P40 GPUs.

### 4.2 Evaluation and baselines

Li et al. (2018) created 1000 human references for each of the Yelp and Amazon test sets using crowdworkers, allowing us to perform automatic evaluation. Following previous work (Shen et al., 2017; Li et al., 2018), we use two quantitative evaluation metrics: classification accuracy and BLEU. Classification accuracy indicates the percentage of system outputs correctly classified as the target sentiment by a pre-trained sentiment classifier. We train our sentiment classifier using Vowpal Wabbit[5] (Agarwal et al., 2014). We use bigram features and train for 20 epochs. BLEU indicates the content similarity between system outputs and human references. We compute a BLEU score using the `multi-bleu.perl` script shipped with OpenNMT-py.

We compare our method against three different baselines. First, Shen et al. (2017)'s cross-aligned autoencoder learns to directly align the populations of the transferred sentences from one sentiment with the actual sentences from the other. Second, Fu et al. (2018)'s multi-decoder learns the sentence representation containing only the content information and generates the output using a sentiment-specific decoder. Both Shen et al. (2017)'s and Fu et al. (2018)'s models are based on adversarial training (Goodfellow et al., 2014). Lastly, Li et al. (2018)'s delete-and-retrieve method generates the output from the content and the retrieved sentiment-specific phrases with a recurrent neural network.

[4] https://github.com/OpenNMT/OpenNMT-py
[5] https://github.com/VowpalWabbit

| Model | Yelp | | Amazon | |
|---|---|---|---|---|
| | Acc | BLEU | Acc | BLEU |
| Shen et al. (2017) | 74.5 | 6.79 | **74.4** | 1.57 |
| Fu et al. (2018) | 46.8 | 11.24 | 70.3 | 7.87 |
| Li et al. (2018) | 88.3 | 12.61 | 53.4 | 27.12 |
| This work | **88.5** | 12.13 | 53.8 | 15.95 |
| w/o $\mathcal{L}_{\text{sentiment}}$ | 3.4 | **24.06** | 18.2 | **42.65** |
| w/o $\mathcal{L}_{\text{content}}$ | 86.4 | 10.08 | 53.9 | 14.77 |
| w/o $\mathcal{L}_{\text{alignment}}$ | 84.7 | 11.94 | 51.6 | 16.51 |
| only $\mathcal{L}_{\text{sentiment}}$ | 85.4 | 10.05 | 53.4 | 14.76 |

Table 1: Results on the Yelp and Amazon test sets.

### 4.3 Results

Table 1 shows the results of various models. Our model based on the unified objective of Eq. (4) offers better balanced results compared to its variants. When removing $\mathcal{L}_{\text{sentiment}}$, our model degrades to an input copy-like method, resulting in low classification accuracies but the highest BLEU scores. When removing $\mathcal{L}_{\text{content}}$, the BLEU scores drop, indicating that the model cannot maintain a sufficient number of content words. Without $\mathcal{L}_{\text{alignment}}$, we observe a reduction in both accuracy and BLEU on Yelp. However, this tendency is inconsistent on Amazon (i.e., −2.2 accuracy and +0.56 BLEU). When using only $\mathcal{L}_{\text{sentiment}}$, our model falls back to the vanilla encoder-decoder model with a single loss, yielding poorer results on both datasets.

| | Yelp | Amazon |
|---|---|---|
| Source | we sit down and we got some really slow and lazy service . | this is the worst game i have come across in a long time . |
| Human | *the* service *was quick* and *responsive* | this is the *best* game i have come across in a long time . |
| Shen et al. (2017) | we *went* down and we *were a good* , *friendly food* . | this is the *best thing* i *ve had for* a *few years* . |
| Fu et al. (2018) | we sit down and we got some really and *fast food* . | this is the *best knife* i have *no room with* a long time . |
| Li et al. (2018) | we got *very nice place to* sit down and we got some service . | this is the *best* game i have come across in a long time . |
| This work | we *sat* down and got some *great* service . | this is the *best item* i have come across in a long time . |

(a) From negative to positive

| | Yelp | Amazon |
|---|---|---|
| Source | my husband got a ruben sandwich , he loved it . | i would definitely recommend this for a cute case . |
| Human | my husband got a *reuben* sandwich, he *hated* it. | i would definitely *not* recommend this for a cute case . |
| Shen et al. (2017) | my husband got a *appetizer* sandwich , *she was* it *wrong* . | i would *not* recommend this for a *long time* . |
| Fu et al. (2018) | my husband got a *beginning house with however i ignored* . | i would definitely recommend this for a *bra does it* . |
| Li et al. (2018) | my husband got a ruben sandwich , it *was too dry* . | i would *not* recommend this for a cute case . |
| This work | my husband got a *turkey* sandwich *and* it *was cold* . | i would *not* recommend this for a *very long time* . |

(b) From positive to negative

Table 2: Example outputs on the Yelp and Amazon test sets. Sentiment-bearing words/phrases are colored. Added/changed words are in *italics*.

### 4.4 Discussion

Li et al. (2018) have made the outputs of their methods and those of Shen et al. (2017) and Fu et al. (2018) publicly available, which allows us to perform a comparison without much effort. On Yelp, our unified objective model achieves the best accuracy and a competitive BLEU score. On Amazon, the adversarial training models of Shen et al. (2017) and Fu et al. (2018) have high accuracies but relatively low BLEU scores. These results indicate that the adversarial training models can modify the sentence to the target sentiment but fail to preserve the content.

Compared with Li et al. (2018)'s delete-and-retrieve method, which seems to reconcile both accuracy and BLEU, our method yields reasonable results given that it does not retrieve new sentences at test time and does not explicitly delete/extract sentiment-specific phrases. Li et al. (2018) also use a separately trained neural language model to select the best output (described in the last paragraph of Sec. 4.4 in their paper), while we only use a single-trained model. Table 2 shows the example outputs of our model and others.

We further examine whether flipping between $\boldsymbol{s}_x$ and $\boldsymbol{s}_y$ has any effect at test time. LSTM is commonly known to be powerful and could generate the output sentence with the target sentiment without using the information from the sentiment embeddings $\boldsymbol{s}_x$ and $\boldsymbol{s}_y$. If our model fails to learn $\boldsymbol{s}_x$ and $\boldsymbol{s}_y$, flipping them at test time should have less or no effect. In other words, given the same input sentence, changing $\boldsymbol{s}_x$ to $\boldsymbol{s}_y$ and vice versa would result in nearly the same outputs. To test this, we compose simple, neutral sentences and feed them to our unified objective model trained on Yelp with different sentiment embeddings. Table 3 shows the generated outputs, confirming that our model indeed makes use of $\boldsymbol{s}_x$ and $\boldsymbol{s}_y$.

| | |
|---|---|
| Source | i have been there . |
| $\boldsymbol{s}_x$ | i have *enjoyed eating* there . |
| $\boldsymbol{s}_y$ | i have *n't* been there . |
| Source | i had experience . |
| $\boldsymbol{s}_x$ | i had *great* experience . |
| $\boldsymbol{s}_y$ | *worst* experience i *ever* had . |

Table 3: Outputs of the proposed model given the same source sentences with different sentiment embeddings ($\boldsymbol{s}_x$ = positive; $\boldsymbol{s}_y$ = negative).

## 5 Conclusion

We have shown that our unified objective model successfully learns from the noisy training sentence pairs acquired from non-parallel corpora using our search-and-corrupt method. Even though our model is conceptually simpler, it produces competitive results against the strong baselines across two domains.

## Acknowledgments

We thank the anonymous reviewers of all versions of this paper for their useful comments.

## References


Alekh Agarwal, Olivier Chapelle, Miroslav Dudík, and John Langford. 2014. A reliable effective terascale linear learning system. *J. Mach. Learn. Res.*, 15(1).

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*.

Mayank Bawa, Tyson Condie, and Prasanna Ganesan. 2005. LSH forest: Self-tuning indexes for similarity search. In *Proceedings of WWW*.

Yoshua Bengio, Li Yao, Guillaume Alain, and Pascal Vincent. 2013. Generalized denoising auto-encoders as generative models. In *Proceedings of NIPS*.

Andrei Z. Broder. 2000. Identifying and filtering near-duplicate documents. In *Proceedings of the 11th Annual Symposium on Combinatorial Pattern Matching*.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2018. Style transfer in text: Exploration and evaluation. In *Proceedings of AAAI*.

Pascale Fung and Percy Cheung. 2004. Mining very-non-parallel corpora: Parallel sentence and lexicon extraction via bootstrapping and em. In *Proceedings of EMNLP*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Proceedings of NIPS*.

Kelvin Guu, Tatsunori B. Hashimoto, Yonatan Oren, and Percy Liang. 2018. Generating sentences by editing prototypes. *Transactions of the Association for Computational Linguistics (TACL)*.

Geoffrey E. Hinton, Peter Dayan, Brendan J. Frey, and Radford M. Neal. 1995. The wake-sleep algorithm for unsupervised neural networks. *Science*, 268.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8).

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Controllable text generation. In *Proceedings of ICML*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Diederik P. Kingma, Danilo J. Rezende, Shakir Mohamed, and Max Welling. 2014. Semi-supervised learning with deep generative models. In *Proceedings of NIPS*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL*.

Guillaume Lample, Sandeep Subramanian, Eric Michael Smith, Ludovic Denoyer, Marc'Aurelio Ranzato, and Y-Lan Boureau. 2019. Multiple-attribute text rewriting. In *Proceedings of ICLR*.

Juncen Li, Robin Jia, He He, and Percy Liang. 2018. Delete, retrieve, generate: A simple approach to sentiment and style transfer. In *Proceedings of NAACL*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of EMNLP*.

Dragos Stefan Munteanu and Daniel Marcu. 2005. Improving machine translation performance by exploiting non-parallel corpora. *Computational Linguistics*, 31(4).

Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W. Black. 2018. Style transfer through back-translation. In *Proceedings of ACL*.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Proceedings of NIPS*.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11.

Jingjing Xu, Xu Sun, Qi Zeng, Xuancheng Ren, Xiaodong Zhang, Houfeng Wang, and Wenjie Li. 2018. Unpaired sentiment-to-sentiment translation: A cycled reinforcement learning approach. In *Proceedings of ACL*.

Zichao Yang, Zhiting Hu, Chris Dyer, Eric P Xing, and Taylor Berg-Kirkpatrick. 2018. Unsupervised text style transfer using language models as discriminators. In *Proceedings of NIPS*.

Yi Zhang, Jingjing Xu, Pengcheng Yang, and Xu Sun. 2018a. Learning sentiment memories for sentiment modification without parallel data. In *Proceedings of EMNLP*.

Zhirui Zhang, Shuo Ren, Shujie Liu, Jianyong Wang, Peng Chen, Mu Li, Ming Zhou, and Enhong Chen. 2018b. Style transfer as unsupervised machine translation. *CoRR*, abs/1808.07894.

# Label Embedding using Hierarchical Structure of Labels for Twitter Classification

**Taro Miyazaki** **Kiminobu Makino** **Yuka Takei** **Hiroki Okamoto** **Jun Goto**
NHK Science and Technology Research Laboratories
{miyazaki.t-jw, makino.k-gg, takei.y-ek,
okamoto.h-iw, goto.j-fw}@nhk.or.jp

## Abstract

Twitter is used for various applications such as disaster monitoring and news material gathering. In these applications, each Tweet is classified into pre-defined classes. These classes have a semantic relationship with each other and can be classified into a hierarchical structure, which is regarded as important information. Label texts of pre-defined classes themselves also include important clues for classification. Therefore, we propose a method that can consider the hierarchical structure of labels and label texts themselves. We conducted evaluation over the Text REtrieval Conference (TREC) 2018 Incident Streams (IS) track dataset, and we found that our method outperformed the methods of the conference participants.

## 1 Introduction

Twitter is used for various applications such as disaster monitoring (Ashktorab et al., 2014; Mizuno et al., 2016) and news material gathering (Vosecky et al., 2013; Hayashi et al., 2015). In these applications, each Tweet is classified into pre-defined classes. These classes have a semantic relationship with each other and can be classified into a hierarchical structure. Consider a news material gathering system that has classes such as "Tornado," "Flood," and "Riot." These can be classified into two upper classes, "*Natural disaster*" ("Tornado" and "Flood" as lower classes) and "*Incident*" ("Riot" as a lower class). These relationships can be an important clue for classification. Needless to say, label texts of pre-defined classes themselves include important clues. People can understand what the criterion of the classification is by reading labels. Therefore, we propose a method that can consider the hierarchical structure of labels and the labels themselves.

Our method is based on the Label Embedding (LE) method (Zhang et al., 2018). The typical LE method embeds input text and label text, and then the embedded vectors are fed into "Matcher," which outputs the score between the input text and label. We use a two-step attention mechanism for LE to consider the hierarchical structure of labels. We confirm the effectiveness of our method through evaluation over the Text REtrieval Conference (TREC) 2018 Incident Streams (IS) track dataset.

Our contributions are as follows: (1) we propose label embedding using the hierarchical structure of labels, and (2) our method outperformed others on the TREC 2018 IS track dataset for several metrics, including the Any-type Micro F1 score, which is the target metric of the track.

## 2 TREC 2018 IS track

The TREC 2018 IS track is a shared-task that aims to mature social media-based emergency response technology (McCreadie et al., 2019). The task of the track is classifying Tweets by information type, which consists of 24 classes. The list of information types and the description for each information type are given as ontology as shown in Table 1. The dataset contains approximately 1,300 Tweets for training and more than 20,000 for testing. The numbers of samples for each class are also given in the table[1]. As shown in the table, a training dataset does not include much data and is unbalanced for classes.

The track uses two evaluation methods — Multi-type and Any-type — as follows[2].

**Multi-type:** Calculating categorization performance per information type in a 1-vs-All manner.

[1] Note that, each sample has just one information type for training data, but more than one for testing data.

[2] See http://trecis.org/2018/Evaluation.html for details.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6317–6322,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

Table 1: Ontology information type.

| Intent type | Information type | Description | # (train) | # (test) |
|---|---|---|---|---|
| REQUEST | REQUEST-GOODSSERVICES | The user is asking for a particular service or physical good | 0 | 126 |
| | REQUEST-SEARCHANDRESCUE | The user is requesting a rescue (for themselves or others) | 0 | 286 |
| | REQUEST-INFORMATIONWANTED | The user is requesting information | 10 | 172 |
| REPORT | REPORT-WEATHER | The user is providing a weather report (current or forecast) | 41 | 1,325 |
| | REPORT-FIRSTPARTYOBSERVATION | The user is giving an eye-witness account | 28 | 3,807 |
| | REPORT-THIRDPARTYOBSERVATION | The user is reporting a information that they received from someone else | 15 | 4,160 |
| | REPORT-EMERGINGTHREATS | The user is reporting a potential problem that may cause future loss of life or damage | 36 | 686 |
| | REPORT-SIGNIFICANTEVENTCHANGE | The user is reporting a new occurrence that public safety officers need to respond to | 34 | 415 |
| | REPORT-MULTIMEDIASHARE | The user is sharing images or video | 127 | 3,974 |
| | REPORT-SERVICEAVAILABLE | The user is reporting that they or someone else is providing a service | 15 | 1,076 |
| | REPORT-FACTOID | The user is relating some facts, typically numerical | 140 | 2,383 |
| | REPORT-OFFICIAL | An official report by a government or public safety representative | 52 | 403 |
| | REPORT-CLEANUP | A report of the clean up after the event | 2 | 62 |
| | REPORT-HASHTAGS | Reporting which hashtags correspond to each event | 4 | 3,363 |
| CALLTOACTION | CALLTOACTION-VOLUNTEER | The user is asking people to volunteer to help the response effort | 2 | 116 |
| | CALLTOACTION-DONATIONS | The user is asking people to donate goods/money | 15 | 804 |
| | CALLTOACTION-MOVEPEOPLE | The user is asking people to leave an area or go to another area | 26 | 27 |
| OTHER | OTHER-PASTNEWS | The post is generic news, e.g. reporting that the event occurred | 12 | 1,351 |
| | OTHER-CONTINUINGNEWS | The post providing/linking to continuous coverage of the event | 250 | 4,871 |
| | OTHER-ADVICE | The author is providing some advice to the public | 39 | 1,209 |
| | OTHER-SENTIMENT | The post is expressing some sentiment about the event | 39 | 6,952 |
| | OTHER-DISCUSSION | Users are discussing the event | 51 | 2,060 |
| | OTHER-IRRELEVANT | The post is irrelevant, contains no information | 163 | 2,605 |
| | OTHER-UNKNOWN | Does not fit into any other category | 26 | 77 |
| | OTHER-KNOWNALREADY | The responder already knows this information | 112 | 1101 |

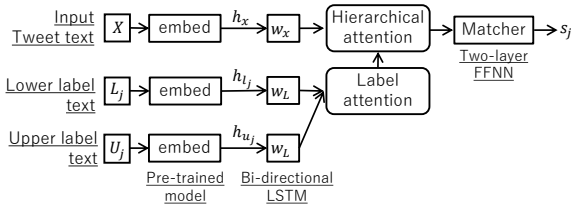


Figure 1: Overview of our proposed method.

Methods are evaluated using four metrics: Precision, Recall, F1 (positive class), and Accuracy (overall).

**Any-type:** A system receives a full score for a Tweet if the system assigned any of the categories that the human assessor selected for that Tweet. Methods are evaluated using four metrics: Precision, Recall, F1 (micro average: the target metric of the track), and Accuracy (micro average).

## 3 Our Method

As shown in Table 1, each information type has a rather rich description, so we think that using the description can improve the performance. Also, the information types can be regarded as a hierarchical-structure label, which consists of four upper classes (equal to intent type, such as REQUEST and REPORT) and 24 lower classes (equal to information type, such as REQUEST-GOODSSERVICES and REPORT-WEATHER). To use these useful features, we propose a label embedding using the hierarchical structure of labels.

The overview of our method is given in Figure 1. Our method is similar to the one Zhang et al. (2018) used, but it differs in that our method considers the hierarchical structure of labels, which is our key feature.

$X$, $L_j$, and $U_j$ are the input text, label text of the $j$-th lower class, and label text of the upper class corresponding to the $j$-th lower class, respectively. We use "description" and "intent type" in Table 1 for the label text of the lower and upper classes, respectively. "Embed" in the figure is a pre-trained model such as BERT (Devlin et al., 2019) and ELMo (Peters et al., 2018) of $V \times m$, where $V$ and $m$ denote the vocabulary size and embedding size, respectively. $\mathbf{w}_X$ and $\mathbf{w}_L$ means Bi-directional Long Short-Term Memory (Bi-LSTM), and both have a $d$ dimension of hidden layer.

First, $X$, $L_j$, and $U_j$ are embedded using the pre-trained model, and then the embedded vectors are fed into $\mathbf{w}_X$ for $X$ and $\mathbf{w}_L$ for $L_j$ and $U_j$, respectively. We gather the hidden states of Bi-LSTM for each input token and stack them, and then we get three matrices:

$$\text{Text representing matrix: } \mathbf{h}_x \in \mathbb{R}^{|X| \times 2d},$$
$$\text{Lower label representing matrix: } \mathbf{h}_{l_j} \in \mathbb{R}^{|L_j| \times 2d},$$
$$\text{Upper label representing matrix: } \mathbf{h}_{u_j} \in \mathbb{R}^{|U_j| \times 2d},$$

where $|X|$, $|L_j|$ and $|U_j|$ represent the number of tokens in $X$, $L_j$ and $U_j$, respectively. Note that we have a $2d$ dimension for the row of each matrix because of concatenating forward and backward states.

Next, we calculate "label attention" and "hierarchical attention." The label attention weight $\mathbf{w}_{\text{label}} \in \mathbb{1}^{|L_j|}$ and label attention vector $\mathbf{a}_{\text{label}} \in$

$\mathbb{1}^{2d}$ are calculated as follows:

$$\mathbf{w}_{\text{label}} = \text{softmax}\left(\mathbf{h} \cdot \mathbf{h}_{l_j}^T / \sqrt{2d}\right),$$
$$\mathbf{a}_{\text{label}} = \mathbf{w}_{\text{label}} \cdot \mathbf{h}_{l_j},$$

where $\mathbf{h} \in \mathbb{1}^{2d}$ is the concatenation of hidden states of the final step of both directions of $\mathbf{h}_{u_j}$. We use scaled-dot product attention for label attention (Vaswani et al., 2017).

By using $\mathbf{a}_{\text{label}}$, the hierarchical attention weight $\mathbf{w}_{\text{hier}} \in \mathbb{1}^{|X|}$ and hierarchical attention vector $\mathbf{a}_{\text{hier}} \in \mathbb{1}^{2d}$ are obtained as follows:

$$\mathbf{w}_{\text{hier}} = \mathbf{a}_{\text{label}} \cdot \mathbf{h}_x^T,$$
$$\mathbf{a}_{\text{hier}} = \mathbf{w}_{\text{hier}} \cdot \mathbf{h}_x.$$

Note that we use normalization for label attention but not for hierarchical attention. We confirm that this condition is best by an experiment, which is detailed in the Discussion section.

Then, $\mathbf{a}_{\text{hier}}$ is fed to "Matcher," which consists of a two-layer Feed-Forward Neural Network (FFNN):

$$\mathbf{v}_{\text{mid}} = \mathbf{W}_{\text{mid}} \cdot \text{ReLU}\left(\mathbf{a}_{\text{hier}}\right) + \mathbf{b}_{\text{mid}},$$
$$s_j = \mathbf{W}_{\text{matcher}} \cdot \text{ReLU}\left(\mathbf{v}_{\text{mid}}\right) + \mathbf{b}_{\text{matcher}},$$

where $\mathbf{W}_{\text{mid}} \in \mathbb{R}^{d \times 2d}$ and $\mathbf{W}_{\text{matcher}} \in \mathbb{R}^{1 \times d}$ are weight matrices, and $\mathbf{b}_{\text{mid}} \in \mathbb{1}^d$ and $\mathbf{b}_{\text{matcher}} \in \mathbb{1}^1$ are bias. $s_j$ is a score between input Tweet $X$ and information type $j$.

We gather $s_j$ for all $j \in C$, where $C$ denotes the set of the information type. Finally, we obtain the estimation result $o$ for input Tweet $X$ as follows:

$$o = \text{argmax}\left(s_1, s_2, \ldots, s_{|C|}\right).$$

# 4 Experiments

## 4.1 Experimental Settings

Our experiments were based on TREC 2018 IS track in terms of the dataset and evaluation. The dataset consists of original json data of Tweets. We excluded @-mentions and URLs. We conducted 10-fold cross validation to find the best setting, and all models were used as ensemble models to predict the information type for testing data.

The models were implemented in Chainer (Tokui et al., 2015) and learned with the Adam optimizer (Kingma and Ba, 2014), on the basis of categorical cross-entropy loss with class weights $W_c = \frac{|c_{max}|}{|c|}$, where $|c|$ is the number of samples of information type $c$ appearing in the training data, and $|c_{max}|$ is that of the most-frequent information type. We used BERT-base, Uncased[3] as the pre-trained model, which has a 12-layer, 768-hidden states transformer model with 12-heads attention. The BERT model was used as a frozen model without fine-tuning.

[3] https://github.com/google-research/bert

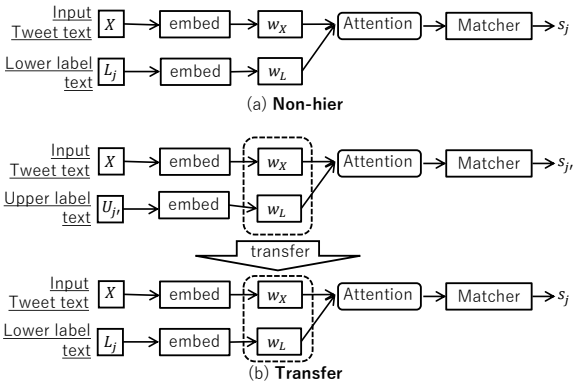


Figure 2: Baseline methods based on label embedding.

The hyperparameters used were: a minibatch size of 32; hidden layer size of 200; $L_2$ regularization coefficient of $10^{-5}$; dropout rate of 0.5; and 100 training iterations, with early stopping on the basis of the Macro F1 score of the development data.

## 4.2 Baseline Methods

We prepared three baseline methods. All baseline methods used the same hyperparameters as the proposed method.

**Non-LE:** This method is not based on LE. We use the BERT model as token embedding. The embedded vectors are fed into Bi-LSTM, and then the concatenation of hidden states of the final step of both directions is fed into the two-layer FFNN for classification into the information type.

**Non-hier:** This method is almost the same as our proposed method but does not use hierarchical attention. The vector fed into Matcher is calculated as attention between embedding vectors of the input text and label text of the lower class (Figure 2-(a)).

**Transfer:** This method uses the same structure as Non-hier, but we use transfer learning to consider the hierarchical structure of labels. This is inspired by Shimura et al. (2018). First, the model is learned as classifying upper class labels, and then the model is transferred to be learned as classifying lower class labels (Figure 2-(b)). We tried several settings and found that transferring only the Bi-LSTM layer is better in this task.

Table 2: Experimental results. *italics* denote the best results in TREC 2018, and **bold** denotes the overall best results. For some metrics, the best results are for other TREC 2018 participants' methods not shown in the table, so no results are in *italics* or **bold** for these metrics.

| Method | Multi-type, Macro Avg. | | | | Any-type, Micro Avg. | | | |
|---|---|---|---|---|---|---|---|---|
| | Precision | Recall | F1 | Accuracy | Precision | Recall | F1 (target metric) | Accuracy |
| Non-LE | 0.2461 | 0.1302 | 0.1523 | 0.9073 | 0.5032 | 0.7055 | 0.5874 | 0.4387 |
| Non-hier | 0.2522 | 0.1300 | 0.1497 | 0.9065 | 0.5206 | 0.6301 | 0.5701 | 0.4282 |
| Transfer | 0.2448 | 0.1324 | 0.1533 | 0.9071 | 0.5097 | 0.6810 | 0.5830 | 0.4361 |
| Proposed | 0.2464 | 0.1300 | 0.1521 | **0.9079** | 0.5164 | 0.7018 | **0.5950** | **0.4460** |
| TREC2018 participants (Top-3 on Any-type Micro F1 score and Top-1 on Multi-type Macro F1 score) | | | | | | | | |
| cbunS2 | ***0.2666*** | 0.1122 | 0.1262 | *0.9059* | 0.4549 | 0.7780 | *0.5749* | *0.4213* |
| KDEIS4_DM | 0.1483 | 0.0708 | 0.0734 | 0.9035 | 0.3914 | 0.9856 | 0.5603 | 0.3908 |
| umdhcilfasttext | 0.1827 | 0.0962 | 0.1117 | 0.9044 | 0.4534 | 0.7260 | 0.5582 | 0.4022 |
| DLR_Augmented | 0.2496 | 0.1657 | ***0.1571*** | 0.8996 | 0.3965 | 0.6165 | 0.4826 | 0.3419 |

### 4.3 Results

Table 2 presents the results for our methods and part of the TREC 2018 participants' results. At TREC 2018, cbunS2, KDEIS4_DM, and umdhcilfasttext had the top three results for the Any-type Micro F1 score, which is the target metric of the track. DLR_Augmented had the best Multi-type Macro F1 score.

Our proposed method outperformed others in terms of the Any-type Micro F1 score and both types of accuracy, while the best for the Multi-type F1 score is DLR_Augmented.

### 4.4 Discussion

#### Comparison with other methods

Our proposed method performs the best for several metrics including the target metric (Any-type Micro F1 score). Our hierarchical attention can give less weight for verbose phrases such as "The user is asking" in label text. These phrases include only a little information useful for classifying, so giving them less weight is reasonable and effective. This is one of the advantages of our method, which distinguishes it from LE-based baseline methods. Our method outperforms not only LE-based methods, but also the simple but strong baseline (Non-LE) and the TREC 2018 participants' methods. Therefore, we confirmed the effectiveness of our proposed method.

On the other hand, our method did not perform the best for the Multi-type Macro F1 score. DLR_Augmented, which performed the best for this metric, uses augmented data. This is effective in the learning process especially for classes that have small training data. Our method does not use augmented data, so it has lower accuracy for these classes than DLR_Augmented. Comparing our proposed method with the baseline methods, differences in the Multi-type Macro F1 score were small, and the proposed method achieved the best score for Multi-type accuracy. We think that the differences come from labels that have a small number of samples in test set because the F1 scores for these classes are sensitive to the output, which greatly affects the Macro F1 score.

Table 3: Comparing normalization methods.

| label attention | hierarchical attention | Macro F1 score |
|---|---|---|
| normalized | normalized | 0.2676 |
| **normalized** | **none** | **0.3539** |
| none | normalized | 0.0303 |
| none | none | 0.0329 |

#### Insight of our proposed method

To determine which combination of normalization is effective for our task, we conducted a small experiment using development data. Results are shown in Table 3. We used the Macro F1 score as the metric[4]. We can see that using normalization for label attention but not for hierarchical attention is best. One reason is that the norm of the output vector of hierarchical attention directly affects the score $s_j$. Hierarchical attention without normalization can make distinctions for the output vector of each class in the norm, so it works well. On the other hand, using normalization is better in label attention. We observed that not using normalization for label attention makes the model diverge, so normalization needs to be used for label attention.

Table 4 shows the macro F1 scores summarized for each upper class of the proposed method and Non-hier. $\Delta$ in the table means the difference between two methods. Interestingly, our method works best for the upper class OTHER, which contains the fewest meanings in the upper class label

[4] Each sample has just one information type for development data and more than one for testing data, so the Macro F1 score between Table 2 and 3 cannot be compared directly.

Table 4: Comparing macro F1 score for each upper class between proposed and Non-hier.

| Upper class | Proposed | Non-hier | Δ |
|---|---|---|---|
| REQUEST | 0.030 | 0.037 | -0.007 |
| REPORT | 0.124 | 0.125 | -0.001 |
| CALLTOACTION | 0.180 | 0.170 | +0.010 |
| OTHER | 0.228 | 0.216 | +0.012 |

text. The lower class label text for the upper class OTHER includes less verbose phrases such as "The user requesting," so label attention can give higher weight for important parts of the label text more precisely.

On the other hand, the F1 score of the proposed method for the upper class REPORT is worse than that of Non-hier. This is caused by the lower class REPORT-SIGNIFICANTEVENTCHANGE (the Δ is $-0.05$, which is the worst in all lower classes). The label text of the lower class includes a misspelling ("occurence" instead of "occurrence"[5]). This confuses our model when calculating label attention weight, so the F1 score is lower.

We found that the number of tokens in the lower class label text affects the effectiveness of the proposed method. The proposed method has a better F1 score than Non-hier for nine lower classes with a mean number of tokens in the lower class labels of 11.2. On the other hand, the proposed method has a lower F1 score than Non-hier for seven lower classes with a mean number of tokens of 8.0. This shows that the more tokens the lower class label text has, the more effective the proposed method becomes. Of course, this does not mean that our method works for only classes that have rich label text. For example, our method improves the F1 score for some classes that have short label text such as OTHER-DISCUSSION, which has only five tokens in the label text.

Overall, using hierarchical structure of labels is effective in many cases, but it is sensitive to the quality and quantity of label texts of the lower classes.

## 5 Related Work

Label embedding has attracted attention, especially for few- and zero-shot learning tasks (Socher et al., 2013; Akata et al., 2013; Kodirov et al., 2017). Now, many methods applied to natural language processing are proposed. Zhang et al. (2018) used multi-task learning, and Xia et al. (2018) used capsule networks for LE and obtained good results. Our method differs in that it considers the hierarchical structure of labels.

There are many studies using a pre-defined hierarchical structure of labels (Wu et al., 2014; Li et al., 2015; Bilal et al., 2018), and some methods are used with LE (Bengio et al., 2010; Ren et al., 2016). These approaches are effective but large-scale hierarchical data from a thesaurus need to be prepared. Our method does not need large-scale data, so it is advantageous when it comes to calculating cost and flexibility. Shimura et al. (2018) used transfer learning to consider the hierarchical structure of labels, which is the basis of our baseline method (Transfer).

## 6 Conclusion and Future Work

In this paper, we proposed a method of label embedding using the hierarchical structure of labels and confirmed its effectiveness through evaluation over the Text REtrieval Conference (TREC) 2018 Incident Streams (IS) track dataset. Our method outperformed other methods and obtained the best results on the dataset for several metrics including the Any-type Micro F1 score, which was the target metric of TREC 2018 IS track.

We used rather long sentences for lower label text. Typical applications use shorter labels such as "*Tornado*" and "*Riot*." Confirming whether our method works well for other datasets including ones that have only shorter label texts is left for our future work.

## Acknowledgements

We would like to thank Prof. Timothy Baldwin and Dr. Ichiro Yamada for valuable discussions, and the TREC 2018 IS track organizers for providing the dataset. We also thank the anonymous reviewers for their helpful comments.

## References

Zeynep Akata, Florent Perronnin, Zaid Harchaoui, and Cordelia Schmid. 2013. Label-embedding for attribute-based classification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 819–826.

Zahra Ashktorab, Christopher Brown, Manojit Nandi, and Aron Culotta. 2014. Tweedr: Mining twitter to

[5] Our label texts are made from provided ontology, so they contain misspellings arising from the original ontology.

inform disaster response. In *Proceedings of the 11th International ISCRAM Conference*.

Samy Bengio, Jason Weston, and David Grangier. 2010. Label embedding trees for large multi-class tasks. In *Advances in Neural Information Processing Systems 23*, pages 163–171.

Alsallakh Bilal, Amin Jourabloo, Mao Ye, Xiaoming Liu, and Liu Ren. 2018. Do convolutional neural networks learn class hierarchy? *IEEE transactions on visualization and computer graphics*, 24(1):152–162.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 4171–4186.

Kohei Hayashi, Takanori Maehara, Masashi Toyoda, and Ken-ichi Kawarabayashi. 2015. Real-time top-r topic detection on twitter with topic hijack filtering. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 417–426.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Elyor Kodirov, Tao Xiang, and Shaogang Gong. 2017. Semantic autoencoder for zero-shot learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3174–3183.

Xirong Li, Shuai Liao, Weiyu Lan, Xiaoyong Du, and Gang Yang. 2015. Zero-shot image tagging by hierarchical semantic embedding. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 879–882.

Richard McCreadie, Cody Buntain, and Ian Soboroff. 2019. TREC incident streams: Finding actionable information on social media. In *Proceedings of 16th International Conference on Information Systems for Crisis Response and Management*.

Junta Mizuno, Masahiro Tanaka, Kiyonori Ohtake, Jong-Hoon Oh, Julien Kloetzer, Chikara Hashimoto, and Kentaro Torisawa. 2016. WISDOM X, DISAANA and D-SUMM: Large-scale NLP systems for analyzing textual big data. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, pages 263–267.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 2227–2237.

Xiang Ren, Wenqi He, Meng Qu, Lifu Huang, Heng Ji, and Jiawei Han. 2016. Afet: Automatic fine-grained entity typing by hierarchical partial-label embedding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1369–1378.

Kazuya Shimura, Jiyi Li, and Fumiyo Fukumoto. 2018. HFT-CNN: Learning hierarchical category structure for multi-label short text categorization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 811–816.

Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. 2013. Zero-shot learning through cross-modal transfer. In *Advances in neural information processing systems*, pages 935–943.

Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. 2015. Chainer: a next-generation open source framework for deep learning. In *Proceedings of workshop on machine learning systems (LearningSys) in the twenty-ninth annual conference on neural information processing systems (NIPS)*, volume 5, pages 1–6.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Jan Vosecky, Di Jiang, Kenneth Wai-Ting Leung, and Wilfred Ng. 2013. Dynamic multi-faceted topic discovery in twitter. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 879–884.

Chenxia Wu, Ian Lenz, and Ashutosh Saxena. 2014. Hierarchical semantic labeling for task-relevant RGB-D perception. In *Robotics: Science and systems*.

Congying Xia, Chenwei Zhang, Xiaohui Yan, Yi Chang, and Philip Yu. 2018. Zero-shot user intent detection via capsule neural networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3090–3099.

Honglun Zhang, Liqiang Xiao, Wenqing Chen, Yongkun Wang, and Yaohui Jin. 2018. Multi-task label embedding for text classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4545–4553.

# Interpretable Word Embeddings via Informative Priors

**Miriam Hurtado Bodell**[*]
Linköping University
miriam.hurtado.bodell@liu.se

**Martin Arvidsson**[*]
Linköping University
martin.arvidsson@liu.se

**Måns Magnusson**
Aalto University
mans.magnusson@aalto.fi

## Abstract

Word embeddings have demonstrated strong performance on NLP tasks. However, lack of interpretability and the unsupervised nature of word embeddings have limited their use within computational social science and digital humanities. We propose the use of informative priors to create interpretable and domain-informed dimensions for probabilistic word embeddings. Experimental results show that sensible priors can capture latent semantic concepts better than or on-par with the current state of the art, while retaining the simplicity and generalizability of using priors.

## 1 Introduction

Increased availability of large digitized corpora and significant developments in natural language processing (NLP) have sparked a growing interest within computational social science and digital humanities (CSSDH) to use computational methods for textual data (Laver et al., 2003; Grimmer, 2010; DiMaggio et al., 2013; Jockers and Mimno, 2013; Tsur et al., 2015). Word embeddings, a family of unsupervised methods for representing words as dense vectors (Mikolov et al., 2013b; Pennington et al., 2014), are one such development. Although word embeddings have demonstrated strong performance on NLP tasks (Mikolov et al., 2013a,c), they have yet to gain widespread attention within CSSDH.

We believe two key limitations can help explain the lack of applications within CSSDH. First, since the dimensions of the word embeddings are largely uninterpretable, it is not clear how to disentangle why words are similar. Substantive interpretability is key for CSSDH research, and thus, the lack thereof is a major limitation. Second, off-the-shelf word embedding models generally lack a channel through which substantive research questions can be incorporated.

To improve interpretability, previous research suggests using sparsity constraints (Murphy et al., 2012; Sun et al., 2016; Faruqui et al., 2015) and rotation techniques (Park et al., 2017; Rothe and Schütze, 2016; Dufter and Schütze, 2019). Other work considers dimension-specific constraints to remove gender-bias, and, as a by-product, improve interpretability (Zhao et al., 2018). Within CSSDH, previous work derives interpretable dimensions via post-processing in the form of antonym-pair vector algebra (Kozlowski et al., 2018; Garg et al., 2018) and ideal-point anchoring of antonym word-pairs (Lauretig, 2019). Recent formulations of word embeddings as probabilistic models (Vilnis and McCallum, 2015; Rudolph et al., 2016; Barkan, 2017; Havrylov et al., 2018) enable the incorporation of domain knowledge through priors. In this paper, we add to the literature on interpretable word embeddings, proposing a novel use of *informative priors* to create predefined interpretable dimensions – thus leveraging the expressiveness and generalizeability of the probabilistic framework.

## 2 Informative Priors in Word Embeddings

The central idea of this paper is to use informative priors to restrict the degree to which different words can inhabit different dimensions, such that one or more dimensions become interpretable and connected to one's research interest. Specifically, we place informative priors on word types that we expect to discriminate on a particular dimension, e.g. *man-woman* for a gender dimension.

Let $\mathcal{V}_+$ and $\mathcal{V}_-$ be the set of *anchor word types* that informative priors is placed on, e.g. $\mathcal{V}_+ = \{man\}$ and $\mathcal{V}_- = \{woman\}$. Also, let $\mathcal{V}_\pm = \mathcal{V}_+ \cup \mathcal{V}_-$, and let $\overline{\mathcal{V}}_\pm$ be the word types without any informative priors, i.e. the complement set.

Given a corpora with vocabulary size $V$, we represent each token $x_i \in \{0, 1\}^V$ as a one-hot vector

[*] Equal contribution



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6323–6329, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

with a single nonzero entry at $v$. Here, $v \in \mathcal{V}$ represents the word type at position $i$ in the text. Following Rudolph et al. (2016), we model each individual entry $x_{iv} \in \{0, 1\}$ of the one-hot vectors conditional on its context $\mathbf{x}_{c_i}$, i.e the tokens surrounding $x_i$, where $c_i$ denotes the positions belonging to the context.

Each word type is associated with an *embedding vector*, $\rho_v \in \mathbb{R}^K$, which governs the distribution of $x_i$, and a *context vector*, $\alpha_v \in \mathbb{R}^K$, governing the distributions of the tokens for which $x_i$ is part of $\mathbf{x}_{c_i}$. The conditional probability of $x_{iv}$ is modelled as a linear combination of the embedding and context vectors, i.e.

$$p(x_{iv}|\mathbf{x}_{c_i}) \sim \text{Bernoulli}(\eta_{iv}), \quad (1)$$

where

$$\eta_{iv} = \text{logit}^{-1}[\rho_v^\intercal(\textstyle\sum_{j \in c_i} \sum_{v'} \alpha_{jv'} x_{jv'})],$$

with logit serving as the link function. Rudolph et al. (2016) place a zero-centered Gaussian prior with variance $\sigma$ on the embedding and context vectors. Letting $\theta_v = \{\rho_v, \alpha_v\}$, this translates to

$$\theta_v \sim \mathcal{N}(0, \sigma). \quad (2)$$

In addition, since evolution of semantic concepts are of special interest in CSSDH (Tahmasebi et al., 2015), we also consider a dynamic word embedding model to capture temporal dynamics in the dimension of interest. We follow the specification in Rudolph and Blei (2017) which extends Eq. (1) by associating each token with a time slice $t$, and fit separate $\rho_v^{(t)} \in \mathbb{R}^K$ for each $t$. Thus, $\theta_v^{(t)} = \{\rho_v^{(t)}, \alpha_v\}$. To share statistical strength between time-points, a Gaussian random walk prior is placed on $\rho_v^{(t)}$, i.e.

$$\rho_v^{(t)} \sim \mathcal{N}(\rho_v^{(t-1)}, \sigma_d I). \quad (3)$$

Where $\sigma_d = \frac{\sigma}{100}$, as in Rudolph and Blei (2017), determines the smoothness of the trajectories. This shows how, in contrast to the state of the art (Kozlowski et al., 2018; Garg et al., 2018; Lauretig, 2019; Zhao et al., 2018), informative priors allow easy integration with other, more complex, probabilistic models.

### 2.1 The Standard Basis Prior

In the following sections, we introduce a number of prior specifications that differ in how they restrict the degree to which words can occupy different dimensions. Letting $K$ represent the dimension that we want to make interpretable, and dimensions $1 : K-1$ be standard word embedding dimensions not subject to interpretation, we define our first prior specification, the *Standard Basis Prior*, as

$$\begin{aligned} \theta_{\mathcal{V}_\pm}^{1:K} &\sim \mathcal{N}(0, \sigma) & \theta_{\mathcal{V}_+}^{K} &\sim \mathcal{N}(1, \gamma) \\ \theta_{\mathcal{V}_\pm}^{1:K-1} &\sim \mathcal{N}(0, \omega) & \theta_{\mathcal{V}_-}^{K} &\sim \mathcal{N}(-1, \gamma), \end{aligned} \quad (4)$$

where $\theta_{\mathcal{V}_+}^K$ and $\theta_{\mathcal{V}_-}^K$ are priors on the dimension of interest ($K$) of $\rho_v$ and $\alpha_v$ for word types in $\mathcal{V}_+$ and $\mathcal{V}_-$ respectively, and where $\theta_{\mathcal{V}_\pm}^{1:K-1}$ is the shared prior for all anchor word types on dimensions $1 : K-1$. Finally, $\theta_{\mathcal{V}_\pm}^{1:K}$ is the standard prior from Eq. (2), placed on all dimensions for all non-anchor word types. Hyperparameters $\omega$ and $\gamma$ are shared for $v \in \mathcal{V}_\pm$, controlling the strength of the prior. As $\gamma, \omega \to 0$, we force $v \in \mathcal{V}_\pm$ to essentially become a standard basis, defined by the word types in the prior. Consequently, the dot product for these vectors will be 0 for all dimensions except $K$, and thus the effect of the anchored word types on the rest of the vocabulary will exclusively depend on $K$.

However, this implies that, as $\gamma, \omega \to 0$, word types within $\mathcal{V}_-$ and $\mathcal{V}_+$ obtain exactly the same word embedding. For example, with $\mathcal{V}_+ = \{brother, king\}$, *brother* is treated as semantically identical to *king*. To address this issue, we consider increasing $\omega$, allowing the anchor word types to exist more freely in the first $K-1$ dimensions while remaining close to $-1$ and 1 in the $K^{th}$ dimension. This permits the use of both $brother$ and $king$ as prior anchors without assuming that they are exactly the same word. We henceforth refer to a standard basis prior with $\omega = 10^{-6}$ as *strict* and $\omega = 1$ as *weak*.

### 2.2 The Truncated Prior

A limitation of the prior specifications introduced thus far is the implicit assumptions that (i) anchor word types discriminate equally on the dimension of interest, and (ii) that we know their exact location in this dimension. To address this, we consider the *Truncated Prior* that does not assume a basis for the anchored word types, but only that they live on the positive and negative real line as

$$\begin{aligned} \theta_{\mathcal{V}_\pm}^{1:K} &\sim \mathcal{N}(0, \sigma) & \theta_{\mathcal{V}_+}^{K} &\sim \mathcal{N}^+(0, \gamma) \\ \theta_{\mathcal{V}_\pm}^{1:K-1} &\sim \mathcal{N}(0, \omega) & \theta_{\mathcal{V}_-}^{K} &\sim \mathcal{N}^-(0, \gamma). \end{aligned} \quad (5)$$

Where $\mathcal{N}^+(0, 1)$ and $\mathcal{N}^-(0, 1)$ is the positive and negative truncated normal distribution, truncated at 0, respectively.

### 2.3 Using Neutral Words

So far, we have only considered word types known to discriminate on the dimension of interest as prior anchors. However, domain knowledge might also inform us about *neutral* words in this dimension (e.g. stop words). We thus consider placing informative priors on a third set of prior anchors $\mathcal{V}_*$ containing neutral word types as

$$\begin{aligned} \theta_{\mathcal{V}_*}^{1:K-1} &\sim \mathcal{N}(0, \sigma) \\ \theta_{\mathcal{V}_*}^{K} &\sim \mathcal{N}(0, \psi)\,, \end{aligned} \tag{6}$$

where $\psi$ is the variance for dimension $K$. By guiding neutral words close to 0 in the dimension of interest, explanatory power that otherwise might have been attributed to word types in $\mathcal{V}_*$ will instead be allocated to other words in $\mathbf{x}_{c_i}$.

## 3 Experiments

Our main empirical concern is how well the proposed priors can capture meaningful latent dimensions. We summarize our empirical questions as follows:

1. Which prior specification can best capture predefined dimensions?

2. How does the best prior specification compare with the state of the art in CSSDH (Garg et al., 2018; Kozlowski et al., 2018) (henceforth referred to as SOTA)?

We consider two semantic dimensions, *gender*, which is explored in SOTA, and *sentiment*, a dimension proven difficult to capture in standard word embedding models (Tang et al., 2014). We follow SOTA when choosing prior anchors for gender, while using the AFINN dictionary (Nielsen, 2011) to find prior anchors for sentiment. To evaluate the effect of "few" vs. "many" prior anchors, we run experiments using between 2 and 276 words depending on the dimension of interest and the dataset at hand. All prior word types used in the experiments can be found in Sec. B in the supplementary material. We follow Rudolph et al. (2016), obtaining maximum a posteriori estimates of the parameters using TensorFlow (Abadi et al., 2015) with the Adam optimizer (Kingma and Ba, 2015) and negative sampling [1]. We set the size of the embeddings $K = 100$, use a context window size of 8 and $\sigma = 1$ throughout all experiments.

[1]Code is available at: https://github.com/martinarvidsson/InterpretableWordEmbeddings

We examine the proposed priors using three commonly sized English corpora for textual analysis within CSSDH: the top 100 list of books in Project Gutenberg (2019), a sample from Twitter (Go et al., 2009) and the U.S. Congress Speeches 1981-2016 (Gentzkow et al., 2018). The number of tokens ranges from 1.8M to 40M after preprocessing (see Sec. A in the supplementary material for details). The various origins, sizes and contents of these datasets work as a check of the effect of the priors in different types of corpora.

To measure the extent to which the inferred dimension reflects the semantic concept of interest, we consider how well-placed a number of prespecified hold-out word types (not a part of $\mathcal{V}_\pm$) are in this particular dimension. Specifically, accuracy is computed as the fraction of hold-out words that are placed on the correct side of 0 on the dimension of interest in the embedding and context vectors. For the gender dimension, hold-out word types include the 200 most common names of the last century (Social Security Administration, 2019) and gendered words, while for sentiment, a sample of the AFINN dictionary is used (see Sec. C in the supplementary material). The number of hold-out word types ranges between 213 and 275, since not all exist in each corpus.

The experimental configurations are compared to SOTA, which we implement by (i) fitting the probabilistic word embedding model in Eq. (1) and (2) without informative priors, and (ii) deriving the interpretable dimension post-hoc by subtracting normalized embedding vectors of antonym-pairs, e.g. $\rho_{gender} = (\rho_{man} - \rho_{woman}) + (\rho_{he} - \rho_{she})$. Comparing the sign of the cosine similarity between hold-out words and the created vector allows us to contrast the accuracy of our method with that of SOTA. To get a measure of uncertainty we calculate binomial confidence intervals using the normal approximation.

## 4 Results

We begin by comparing the strict and weak *Standard Basis Priors*, with $\gamma = 10^{-6}$, varying the number of anchor words used to inform the dimension of interest. The general pattern in Fig. 1 shows how (i) increasing the number of anchor words improves the accuracy, and that (ii) this improvement is much greater for the weak than for the strict *Standard Basis Prior*. We explain this

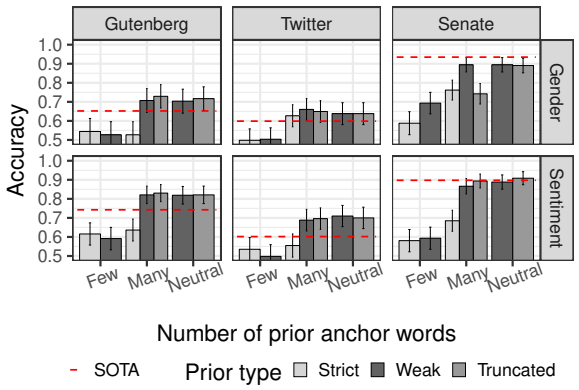


Figure 1: Comparison of prior specifications per dataset and dimension

difference by the nature of the strict prior; it forces all anchor words to have exactly the same meaning – which clearly is untrue. These results speak against methods that transform the vector space based on strict standard basis vectors, as in Lauretig (2019). Further noticeable is that the average accuracy is greater for the Senate corpus, which, using subsampling, we find is driven by corpora size.

Relaxing the location assumptions with the *Truncated Prior* ($\gamma = 1000$) and placing informative priors on neutral words ($\psi = 0.01$) show varying degrees of improvements. However, due to the limited number of hold-out test words, large improvements are required for significant differences – which only is observed for the sentiment dimension in the Senate corpus.

Fig. 1 further shows how our proposed approach generally performs better or on-par with SOTA. The one exception is the gender dimension in the Senate corpus. Follow-up analysis show that this difference is driven by misclassification of a cluster of ambiguous names, e.g. *Madison* (founding father) and *Jordan* (country). These names become correctly classified when using the full vectors, suggesting that gender has not been completely isolated in the Senate corpus (see Gonen and Goldberg (2019) for an in-depth discussion on issues of concept-isolation in word embeddings).

### 4.1 Case Study of Semantic Change

Using the *Truncated Prior* with neutral words, we leverage the dynamic embedding model described in Eq. (3) to explore temporal patterns in the Senate corpus. We set $\sigma_d = 0.05$ to allow for the identification of abrupt temporal changes.

The left panel of Fig. 2 displays sentiment-trajectories for two words experiencing drastic three-year consecutive changes in the sentiment dimension; *September* between 1999 and 2001 and *Oklahoma* between 1993 and 1995, capturing two terror events: the attacks on the World Trade Center on September 11th 2001, and the Oklahoma City bombing in 1995. The change precedes the event due to the smoothness of the prior. In the years that follow, the words gradually regain their previous sentiment – reflecting a decline in their association with terror.

Second, we test and find support for the gender-occupation results in Kozlowski et al. (2018), i.e. that occupations' temporal positioning in the gender dimension correlates well with the proportion of men and women within those fields. The right panel of Fig. 2 displays the gender-dimension trajectories for the words *Nurse* and *Laywer*, occupations with high shares of women and men, respectively (Kozlowski et al., 2018).

In sum, these examples showcase how interpretability can be gained using informative priors. By isolating prespecified concepts, e.g. sentiment, into one dimension ($K$), one can infer word type-concept associations such as *September-sentiment* – for all non-anchor word types – allowing for more meaningful within and between word type comparisons.

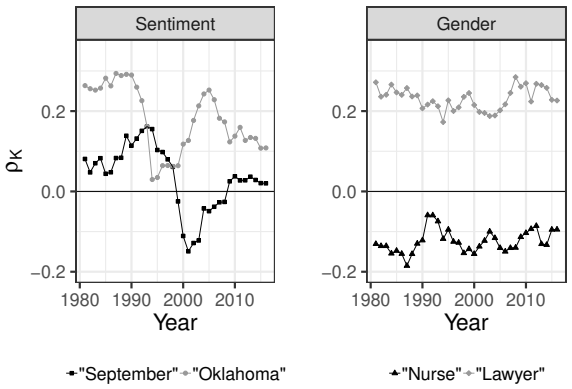


Figure 2: Gender and sentiment trajectories in the Senate corpus.

## 5 Conclusion

In this paper, we show how informative priors provide a simple and straightforward alternative for constructing interpretable dimensions for probabilistic word embeddings – allowing CSSDH researchers to explore how words relate to each

other in prespecified dimensions of interest, e.g. gender and sentiment.

Our results demonstrate that the biggest gains in interpretability are obtained by (i) using many prior words and (ii) increasing the degree to which they can live outside the predefined dimension. The weak *Standard Basis Prior* and the *Truncated Prior* overall capture the predefined dimensions with similar accuracy. Aligned with previous research, the experimental results indicate some issues with incomplete isolation of concepts which we believe could be addressed in future work by placing informative priors on multiple dimension to capture more complex concepts, and move away from simplistic antonym-driven dimension definitions.

Finally, while being flexible and easily extended to other probabilistic word embedding models, our approach performs better or on par with SOTA.

## Acknowledgement

This work was supported by the Swedish Research Council, grant numbers: 2018–05170 and 2018–06063, and Academy of Finland grant number 313122. We thank Marc Keuschnigg, Peter Hedström, Jacob Habinek and all reviewers for their valuable comments.

## References

Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2015. Tensorflow: Large-scale machine learning on heterogeneous systems, 2015. *Software available from tensorflow. org*, 1(2).

Oren Barkan. 2017. Bayesian neural word embedding. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Paul DiMaggio, Manish Nag, and David Blei. 2013. Exploiting affinities between topic modeling and the sociological perspective on culture: Application to newspaper coverage of us government arts funding. *Poetics*, 41(6):570–606.

Philipp Dufter and Hinrich Schütze. 2019. Analytical methods for interpretable ultradense word embeddings. *arXiv preprint arXiv:1904.08654*.

Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah A. Smith. 2015. Sparse overcomplete word vector representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1491–1500, Beijing, China. Association for Computational Linguistics.

Nikhil Garg, Londa Schiebinger, Dan Jurafsky, and James Zou. 2018. Word embeddings quantify 100 years of gender and ethnic stereotypes. *Proceedings of the National Academy of Sciences*, 115(16):E3635–E3644.

M. Gentzkow, J. M. Shapiro, and M. Taddy. 2018. Congressional record for the 43rd-114th congresses: Parsed speeches and phrase counts. http://data.stanford.edu/congress_text. Accessed: 2019-04-17.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12):2009.

Hila Gonen and Yoav Goldberg. 2019. Lipstick on a pig: Debiasing methods cover up systematic gender biases in word embeddings but do not remove them. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 609–614, Minneapolis, Minnesota. Association for Computational Linguistics.

Justin Grimmer. 2010. A bayesian hierarchical topic model for political texts: Measuring expressed agendas in senate press releases. *Political Analysis*, 18(1):1–35.

Serhii Havrylov, Ivan Titov, et al. 2018. Embedding words as distributions with a bayesian skip-gram model. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1775–1789.

Matthew L Jockers and David Mimno. 2013. Significant themes in 19th-century literature. *Poetics*, 41(6):750–769.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Austin C Kozlowski, Matt Taddy, and James A Evans. 2018. The geometry of culture: Analyzing meaning through word embeddings. *arXiv preprint arXiv:1803.09288*.

Adam Lauretig. 2019. Identification, interpretability, and Bayesian word embeddings. In *Proceedings of the Third Workshop on Natural Language Processing and Computational Social Science*, pages 7–17, Minneapolis, Minnesota. Association for Computational Linguistics.

Michael Laver, Kenneth Benoit, and John Garry. 2003. Extracting policy positions from political texts using words as data. *American political science review*, 97(2):311–331.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751.

Brian Murphy, Partha Talukdar, and Tom Mitchell. 2012. Learning effective and interpretable semantic models using non-negative sparse embedding. *Proceedings of COLING 2012*, pages 1933–1950.

F. Å. Nielsen. 2011. AFINN. http://localhost/pubdb/p.php?6010.

Sungjoon Park, JinYeong Bak, and Alice Oh. 2017. Rotated word vector representations and their interpretability. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 401–411.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Project Gutenberg. 2019. http://www.gutenberg.org. Accessed: 2019-04-24.

Sascha Rothe and Hinrich Schütze. 2016. Word embedding calculus in meaningful ultradense subspaces. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 512–517.

Maja Rudolph and David Blei. 2017. Dynamic bernoulli embeddings for language evolution. *arXiv preprint arXiv:1703.08052*.

Maja Rudolph, Francisco Ruiz, Stephan Mandt, and David Blei. 2016. Exponential family embeddings. In *Advances in Neural Information Processing Systems*, pages 478–486.

Social Security Administration. 2019. Top names over the last 100 years. https://www.ssa.gov/oact/babynames/decades/century.html. Accessed: 2019-05-02.

Fei Sun, Jiafeng Guo, Yanyan Lan, Jun Xu, and Xueqi Cheng. 2016. Sparse word embeddings using l1 regularized online learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 2915–2921. AAAI Press.

Nina Tahmasebi, Lars Borin, Gabriele Capannini, Devdatt Dubhashi, Peter Exner, Markus Forsberg, Gerhard Gossen, Fredrik D Johansson, Richard Johansson, Mikael Kågebäck, et al. 2015. Visions and open challenges for a knowledge-based culturomics. *International Journal on Digital Libraries*, 15(2-4):169–187.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1555–1565.

Oren Tsur, Dan Calacci, and David Lazer. 2015. A frame of mind: Using statistical models for detection of framing and agenda setting campaigns. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1629–1638.

Luke Vilnis and Andrew McCallum. 2015. Word representations via gaussian embedding. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Jieyu Zhao, Yichao Zhou, Zeyu Li, Wei Wang, and Kai-Wei Chang. 2018. Learning gender-neutral word embeddings. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4847–4853, Brussels, Belgium. Association for Computational Linguistics.

# Adversarial Removal of Demographic Attributes Revisited

**Maria Barrett**[1], **Yova Kementchedjhieva**[1], **Yanai Elazar**[2],
**Desmond Elliott**[1], **Anders Søgaard**[1]
[1]University of Copenhagen, [2]Bar-Ilan University
{mjb,yova,de,soegaard}@di.ku.dk,yanaiela@gmail.com

## Abstract

Elazar and Goldberg (2018) showed that protected attributes can be extracted from the representations of a *debiased* neural network for mention detection at above-chance levels, by evaluating a diagnostic classifier on a held-out subsample of the data it was trained on. We revisit their experiments and conduct a series of follow-up experiments showing that, in fact, the diagnostic classifier generalizes poorly to both new in-domain samples and new domains, indicating that it relies on correlations specific to their particular data sample. We further show that a diagnostic classifier trained on the *biased* baseline neural network also does not generalize to new samples. In other words, the biases detected in Elazar and Goldberg (2018) seem restricted to their particular data sample, and would therefore not bias the decisions of the model on new samples, whether in-domain or out-of-domain. In light of this, we discuss better methodologies for detecting bias in our models.

## 1 Introduction

Several approaches have been proposed to learn classifiers that are invariant (unbiased with respect) to protected attributes: cost-sensitive (Agarwal et al., 2018), regularization-based (Bechavod and Ligett, 2017), and adversarial (Ganin and Lempitsky, 2015). In the adversarial approach, a model learns representations $x$ that should be predictive for a main task $y$ and oblivious to a protected attribute $z$.

Adversarial training has been used to learn data representations that are invariant to demographic attributes (Raff and Sylvester, 2018; Beutel et al., 2017; Li et al., 2018), as well as representations invariant to domain differences (Ganin and Lempitsky, 2015), clean or noisy speech (Sriram et al., 2018), and invariant to differences between source languages in multi-lingual machine translation (Xie et al., 2017). Elazar and Goldberg (2018) argue, however, that adversarial learning does not fully remove sensitive demographic traits from the data representations. This conclusion is based on the observation that a diagnostic classifier trained over the supposedly debiased data representations could still predict gender, age and race above chance level in their experimental setup.

In general, diagnostic classifiers are trained on data representations to predict the protected demographic attributes in question as well as possible, i.e., the classifier picks up on any correlations, strong or weak, between data representations and the demographic classes. Correlations come in different flavours: PREVALENT: Certain features are indicative of gender in most contexts, e.g., the distribution of phrases like 'as a mother' or 'as a guy'. SAMPLE-SPECIFIC: Some features are indicative of gender within a particular domain/sample, e.g. *bling* may correlate with a particular range of ages in a sample of reviews of accessories, and yet another range in reviews of movies. ACCIDENTAL: Yet, other features may show a correlation with the protected attribute in a given dataset, while in fact being completely unrelated to it. The correlation is unexpected and particular to a finite data sample.

This paper presents a follow-up to the experiments in Elazar and Goldberg (2018) and examines what kind of correlation the data representations in their models exhibit with demographic attributes: PREVALENT, SAMPLE-SPECIFIC or ACCIDENTAL correlations. We do this by *not only* evaluating the diagnostic classifiers on in-sample data, but also on new samples, as well as across textual domains. We also explore what the models learn in a more qualitative manner.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6330–6335,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| | Gender | | | Age | | |
|---|---|---|---|---|---|---|
| | Train | Dev | Test | Train | Dev | Test |
| Pan16 Twit | 148 | 10 | 10 | 148 | 10 | 10 |
| CF Twit | – | – | 6.4 | – | – | – |
| 100 Auth Twit | – | – | 49.6 | – | – | – |
| Pan14 Blogs | – | – | 12.8 | – | – | 8 |
| Pan14 Reviews | – | – | 13 | – | – | 8.4 |
| Pan14 SoMe | – | – | 10 | – | – | 10 |
| Pan16 Rand | – | – | 10 | – | – | 10 |

Table 1: Dataset sizes (in 1000 sentences).

**Contributions** Our contributions are methodological. We show that the diagnostic classifiers used to establish gender and age bias in Elazar and Goldberg (2018) (a) rely only on sample-specific patterns to predict gender and age, and (b) do therefore not generalize to new samples or domains. Surprisingly, we also show that this also holds for the biased baseline model in their experiments, suggesting that the particular representations induced for the mention detection task in Elazar and Goldberg (2018) were not biased with respect to protected demographic attributes. This does not show whether the data *is* biased, or whether adversarial training is a good or poor debiasing technique; merely that Elazar and Goldberg (2018) did not properly establish any of these things. Our contributions are, as said, methodological, and we believe this case study highlights the key difficulties establishing bias in model representations.

## 2 Adversarial Attribute Removal with Diagnostic Classifiers

In adversarial attribute removal, a model is trained with a two-fold objective: learning to solve a main task and *un*learning to predict a protected attribute (+Adv). A model trained without the second objective is referred to as a *non-adversarial* model (-Adv). The architecture of Elazar and Goldberg (2018) consists of a single-layer LSTM encoder, and two multi-layer perceptrons – one for each task. The main task $p(y|x)$ is learned as usual, with loss being backpropagated through the relevant perceptron and the encoder. The attribute unlearning is achieved by training the relevant perceptron to predict $p(z|x)$, where $z$ is the protected attribute, while also punishing the encoder for letting any signal through that could allow the perceptron to do so (Ganin and Lempitsky, 2015). After training, a diagnostic classifier, an *attacker*, is used to evaluate the effectiveness of the adversarial training. The language encoder is used to obtain representations of the input data, and the diagnostic classifier is trained to predict the protected attributes from these representations, without access to the encoder or to the original inputs. Since the dataset is balanced and the targets are binary, *leakage* is defined in Elazar and Goldberg (2018) as any demographic attribute prediction accuracy over 50.0% by the diagnostic classifier.

**Preprocessing** Table 1 shows sizes for all used datasets. Our main dataset, Pan16 Twit (Rangel et al., 2016), is split into train and development, following Elazar and Goldberg (2018). We further remove 10,000 sentences from their train data to use as a held-out test split, making sure there is no author overlap between training and test data. This means that we have 12,000 fewer sentences in our train split than Elazar and Goldberg (2018), but we report results on exactly the same development split as well as on the new held-out test split. Pan16 Twit is balanced using undersampling with respect to main task and demographic attribute (gender and age respectively) which is why there are separate datasets for Gender and Age. Our main observation here is that training, development and test splits and random subsamples of one sample of data. Using random subsamples this way is common in machine learning, including bias detection studies (Elazar and Goldberg, 2018; Zhao et al., 2019) and probing studies (Ravfogel et al., 2018; Lin et al., 2019), but is known to overestimate performance (Globerson and Roweis, 2016), in particular for high-dimensional problems.

**Replication** We start by replicating the experiment of Elazar and Goldberg (2018) using their code on Pan16 Twit with their data splits. The main task is predicting the mentions of other Twitter users after removing all user names in the tweets. The protected demographic attributes are age and gender, both with binary targets. Our development results (main and diagnostic classifier) which are comparable to Elazar and Goldberg (2018) are reported in Table 6 in the Appendix; test set results are also in Table 4. Our results remain comparable to those obtained in Elazar and Goldberg (2018), albeit the diagnostic classifier is able to achieve 3.92 percentage points less leakage for gender and 2.59 percentage points for age, possibly due to the reduction in training data. Adver-

| | AGE | | GENDER | |
|---|---|---|---|---|
| | Mean | RStD | Mean | RStD |
| -ADV | 14.48 | 0.34 | 5.40 | 0.16 |
| +ADV | 5.3 | 0.45 | 3.68 | 0.22 |

Table 2: Mean and relative standard deviation of the *leakage* of 10 attackers trained on different subsamples of the training data (PAN16 TWIT). Evaluated on the test set.

sarial training reduces the leakage of demographic attributes, but the diagnostic classifier is still able to predict the sensitive demographic attribute from the data representations significantly better than chance (line 1 of Table 4). Significance test is obtained by checking subsampled test set accuracies against the 95% confidence interval of a random distribution.

## 3 In-Sample Analysis

In §4, we use out-of-sample and cross-domain evaluation to show that the model trained above relies on spurious or ACCIDENTAL correlations. In this section, as a supplement, we train 10 attackers on different random subsamples of 50% of the training data to explore the robustness of the observed leakage. Each attacker is evaluated on the same PAN16 TWIT test data. Furthermore, we present a qualitative analysis of what the above model has learned.

Table 2 shows the mean and relative standard deviation (RStD) of the leakage of the 10 diagnostic classifiers in each setting. The leakage, unsurprisingly, is larger for the non-adversarial condition than the adversarial condition.

**Extracting leaked samples** We further analyse the tweets that were correctly labelled by the different attackers. Given that these are binary classification tasks, we consider labels predicted with a high probability to be leaked, and labels predicted with a probability close to 0.5 to be randomly chosen. We sort all of the correctly labelled test examples by the probability assigned to the label by the attacker. We denote the top $n$ samples as the leaked samples, where $n$ is the amount of leakage observed for a model[1].

[1]e.g., with a subsampled test accuracy of 54.4%, we take the 4.4% most confidently predicted tweets for each subsampled model and only use the intersection from all models.

| | GENDER | | | AGE | | |
|---|---|---|---|---|---|---|
| | M | F | $n$ | Y | O | $n$ |
| -ADV | 52.1 | 47.9 | 142 | 69.1 | 30.9 | 553 |
| +ADV | 100. | 0. | 37 | 57.9 | 42.1 | 145 |

Table 3: Class distribution for the confidently predicted tweets and $n$umber of tweets per model condition for M(ale), F(emale) and Y(oung), O(ld)

We use the intersection of leaked samples across the attackers, under the assumption that samples that were correctly labelled by all ten models (and with high probability) are most likely to exhibit protected attribute leakage. After deduplicating, we have 179 sentences correctly labelled for Gender, and 698 sentences for age, as reported in Table 3. We call these the LEAKED sentences. (There are no female-authored sentences that are confidently predicted after adversarial training).

**Exploring leaked samples** Out of the LEAKED gender sentences, the top 10 sentences which received the highest confident scores are presented in Table 7 in the Appendix. 28 subjects rated how confident they were about the gender of the author on a 9-point scale. Subjects were recruited via the authors' professional and private networks and were not compensated for the time they spent. All subjects were unaware of the origin of the tweets, and the sentences were presented in a random order. The scale was calibrated such that the extremes represented *Very certain* about the author being a man (1) or a woman (9), with 5 a neutral middle value. The mean for sentences by male authors selected without adversarial training is 4.68 and for females 5.69. With adversarial training, this mean is 4.83. The mean of sentences extracted from the model trained with adversarial training is closest to the neutral value 5, but independent t-tests show that the differences between all classes are insignificant ($p > 0.05$). Therefore, the results show that humans had difficulties determining the gender of the author. This is contrary to the findings of Flekova et al. (2016), but is similar by the unaveraged results of Burger et al. (2011). This indicates that the data did not exhibit (m)any obvious predictors of gender or age.

In addition to this study, we also use this data to visualize what our diagnostic classifiers focus on. For this purpose, we use the models from Section 2, trained on the full PAN16 TWIT dataset, and perform feature analysis using uptraining: Us-




(a) GENDER −ADV (b) GENDER +ADV

(c) AGE −ADV (d) AGE +ADV

Figure 1: Biased $n$-grams in a linear simulation of the attacker model in Elazar and Goldberg (2018). Red=female, blue=male, green=young, brown=old. The size corresponds to the size of the coefficient.

ing a transparent, linear model to approximate our deep model (Bastani et al., 2017), we train a logistic regression model with L1 regularization on the train set representations, relabeled by the predictions of our diagnostic classifier. We then study the coefficients of this linear model. The data is represented using the 10,000 most frequent $n$-grams.[2] We also use the confidence of the diagnostic classifier for importance weighting during training; note that the predicted classes are not balanced.[3] For gender and age, and for our baseline and our adversarially trained system, we uptrain 10 models on random 50% subsets of the training data and average coefficients to filter out noise. Figure 1 shows word clouds for the $n$-grams in the LEAKED sentences, with word size reflecting the size of the avg. coefficients in the uptrained linear models. We observe that for models with no adversarial training, a few $n$-grams dominate the coefficients. In the models where adversarial training was used, there are no large coefficients dominating the space, suggesting that the model succeeded in removing strong correlates; we note that these correlates do not intuitively relate strongly to gender or age.

[2] $n$-gram representations are less expressive than LSTM encodings, but given the short length of tweets, we assume up to 5-gram representation capture most relevant patterns.

[3] Table 5 in the Appendix provides an overview of the class distribution.

| | AGE | | GENDER | |
|---|---|---|---|---|
| TEST SET | -ADV | +ADV | -ADV | +ADV |
| PAN16 TWIT | 67.86* | 53.45* | 56.79* | 53.74* |
| PAN16 RAND | 50.13 | 50.32 | 49.50 | 50.00 |
| 100 AUTH TWIT | – | – | 50.92 | 52.57 |
| CF TWITTER | – | – | 51.63* | 50.74* |
| BLOGS | 50.63* | 55.12* | 50.85 | 49.95 |
| REVIEWS | 51.03 | 50.26 | 50.44 | 49.28 |
| SOME | 50.32 | 50.16 | 50.34 | 48.34 |

Table 4: Cross-sample diagnostic classifier accuracy of classification of demographic attribute when trained on PAN16 TWIT data and evaluated on different test sets. Significantly different from random *=$p < 0.01$.

## 4 Out-of-Sample and Cross-Domain Evaluation

In our main set of experiments, we now evaluate the adversarial and non-adversarial diagnostic classifiers across different samples and different domains. We use the following datasets for these experiments: PAN14 (Rangel et al., 2014)[4] - we include the English data from PAN14 for the following domains: BLOGS, hotel REVIEWS, and Social Media (SOME). The PAN14 data are not annotated for the mention task, but this is not necessary to evaluate the diagnostic classifier on the protected, demographic attribute. We also use data from Crowdflower's Gender Classifier data (CF TWIT), manually annotated for gender[5]. We also made a new Twitter dataset with manually annotated gender for 100 authors (100 AUTH TWIT) for the purpose of evaluating on a different Twitter sample, following the approach used to create PAN16 TWIT – detailed in Appendix A. The datasets are balanced with respect to demographic attributes, and the Twitter datasets are balanced with respect to mentions. We also construct an artificial dataset from PAN16 TWIT where the main task labels are preserved but the demographic label is randomly shuffled (PAN16 RAND), allowing us to run experiments with no PREVALENT or SAMPLE-SPECIFIC gender correlations, only ACCIDENTAL. This experiment supplements our in-sample analysis in §3 and shows that our models are not overly expressive, indicating that the sample-specific correlations detected

[4] We do not include PAN14-Twitter, since the PAN16 TWIT training data subsumes the data from PAN14.

[5] Downloaded from https://d1p17r2m4rzlbo.cloudfront.net/wp-content/uploads/2016/03/gender-classifier-DFE-791531.csv

in Elazar and Goldberg (2018) are relatively simple associations.

The results in Table 4 show, however, that the performance and leakage of the adversarial models do *not* generalize well across domains, but for the most part the performance of the non-adversarial models doesn't either. This is the main result presented here: The leakage shown in Elazar and Goldberg (2018) does not transfer across domains, but also does not even generalize to a new sample within the same domain. This suggests that the leakage is merely spurious correlations in a small, finite sample of data. Accuracies are, as expected, mostly lower when adversarial training is employed, but for the 100 AUTH TWIT sample, the non-adversarial accuracy is close to random, while the adversarial accuracy is two points higher: it appears that the adversarial component has merely served as a regularizer during training.

## 5 Conclusion and Discussion

We examined the methodology used in Elazar and Goldberg (2018) to establish bias in the representations of adversarial machine learning architectures designed to protect demographic attributes.[6] Our results suggest that when measuring demographic parity using a diagnostic classifier, one needs to be careful in controlling for spurious correlations that are limited to just one specific sample of data. In-sample correlations are not necessarily meaningful in high-dimensional problems, and while they can be lead to worse performance on test time (Globerson and Roweis, 2016), they do not lead to demographic bias in practice, when our models are deployed on new samples of data. In order to get more realistic assessments of representation bias with respect to protected attributes, we therefore need to test the out-of-sample generalization of detected bias/leakage, *and*, perhaps, to qualitatively inspect the observed leakage patterns.

We repeat here that none of our results say anything about whether the datasets used in Elazar and Goldberg (2018) are biased or not, or whether other models induced from this data are likely to exhibit biases. Our contribution is mainly methodological: What we have shown is that the methodology in Elazar and Goldberg (2018), i.e., in-sample evaluation of diagnostic classifiers, is not sufficient to establish bias/leakage beyond the current data sample. Instead we propose out-of-sample and cross-domain evaluation, as well as more qualitative investigation of the induced diagnostic classifiers. Our results are also orthogonal to the main contribution of Elazar and Goldberg (2018), which is to show that adversarial debiasing is not always able to remove bias.

[6]The methodology is not only found in Elazar and Goldberg (2018). The same methodology can be found in other papers on detecting bias in machine learning models, for example, Zhao et al. (2019), as well as in several papers probing neural network representations for linguistic knowledge, for example, Lin et al. (2019).

## Acknowledgements

Maria Barrett is sponsored by a Facebook Research Award; Anders Søgaard is sponsored by a Facebook Research Award and a Google Focused Research Award.

## References

Alekh Agarwal, Alina Beygelzimer, Miroslav Dudik, John Langford, and Hanna Wallach. 2018. A reductions approach to fair classification. In *International Conference on Machine Learning*, pages 60–69.

Osbert Bastani, Carolyn Kim, and Hamsa Bastani. 2017. Interpretability via model extraction. In *Workshop on Fairness, Accountability, and Transparency in Machine Learning (FAT/ML 2017)*.

Yahav Bechavod and Katrina Ligett. 2017. Penalizing unfairness in binary classification. *arXiv preprint arXiv:1707.00044*.

Alex Beutel, Jilin Chen, Zhe Zhao, and Ed H Chi. 2017. Data decisions and theoretical implications when adversarially learning fair representations. *arXiv preprint arXiv:1707.00075*.

John D Burger, John Henderson, George Kim, and Guido Zarrella. 2011. Discriminating gender on twitter. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1301–1309. Association for Computational Linguistics.

Yanai Elazar and Yoav Goldberg. 2018. Adversarial removal of demographic attributes from text data. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 11–21. Association for Computational Linguistics.

Lucie Flekova, Jordan Carpenter, Salvatore Giorgi, Lyle Ungar, and Daniel PreoŢiuc-Pietro. 2016. Analyzing biases in human perception of user age and gender from text. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 843–854. Association for Computational Linguistics.

Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*, pages 1180–1189.

Amir Globerson and Sam Roweis. 2016. Nightmare at test time: Robust learning by feature deletion. In *Proceedings of the 23rd International Conference on Machine Learning*.

Yitong Li, Timothy Baldwin, and Trevor Cohn. 2018. Towards robust and privacy-preserving text representations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 25–30. Association for Computational Linguistics.

Yongjie Lin, Yi Chern Tan, and Robert Frank. 2019. Open sesame: Getting inside BERT's linguistic knowledge. In *The 2nd BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*.

Edward Raff and Jared Sylvester. 2018. Gradient reversal against discrimination. In *Proceedings of the 5th Workshop on Fairness, Accountability and Transparency in Machine Learning*.

Francisco Rangel, Paolo Rosso, Irina Chugur, Martin Potthast, Martin Trenkmann, Benno Stein, Ben Verhoeven, and Walter Daelemans. 2014. Overview of the 2nd author profiling task at PAN 2014. In *CLEF 2014 Evaluation Labs and Workshop Working Notes Papers, Sheffield, UK, 2014*, pages 1–30.

Francisco Rangel, Paolo Rosso, Ben Verhoeven, Walter Daelemans, Martin Potthast, and Benno Stein. 2016. Overview of the 4th author profiling task at pan 2016: cross-genre evaluations. In *Working Notes Papers of the CLEF 2016 Evaluation Labs. CEUR Workshop Proceedings/Balog, Krisztian [edit.]; et al.*, pages 750–784.

Shauli Ravfogel, Francis Tyers, and Yoav Goldberg. 2018. Can LSTM learn to capture agreement? the case of Basque. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP*.

Anuroop Sriram, Heewoo Jun, Yashesh Gaur, and Sanjeev Satheesh. 2018. Robust speech recognition using generative adversarial networks. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5639–5643. IEEE.

Qizhe Xie, Zihang Dai, Yulun Du, Eduard Hovy, and Graham Neubig. 2017. Controllable invariance through adversarial feature learning. In *Advances in Neural Information Processing Systems*, pages 585–596.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Ryan Cotterell, Vicente Ordonez, and Kai-Wei Chang. 2019. Gender bias in contextualized word embeddings. In *Proceedings of the North American Chapter of the Association for Computational Lingustics*, pages 629–634.

# A deep-learning framework to detect sarcasm targets

[1]**Jasabanta Patro,** [2]**Srijan Bansal,** [3]**Animesh Mukherjee,**
Indian Institute of Technology Kharagpur, India – 721302
{[1]jasabantapatro, [2]srijanbansal97}@iitkgp.ac.in , [3]animeshm@cse.iitkgp.ac.in

## Abstract

In this paper we propose a deep learning framework for sarcasm target detection in predefined sarcastic texts. Identification of sarcasm targets can help in many core natural language processing tasks such as aspect based sentiment analysis, opinion mining etc. To begin with, we perform an empirical study of the socio-linguistic features and identify those that are statistically significant in indicating sarcasm targets ($p$-values in the range $(0.05, 0.001)$). Finally, we present a deep-learning framework augmented with socio-linguistic features to detect sarcasm targets in sarcastic book-snippets and tweets. We achieve a huge improvement in the performance in terms of exact match and dice score as compared to the current state-of-the-art baseline.

## 1 Introduction

Computational sarcasm is a very well studied research area in computational linguistics (Joshi et al., 2017). Sentiment analysis and opinion mining of sarcastic texts are known to be difficult problems (Pang et al., 2008). For instance, in aspect based sentiment analysis, which deals with the identification of sentiment expressed toward different aspects or dimensions of the entities present in the text, it is very important to identify the sarcasm targets and sentiments toward them in the texts. Thus, if a user expresses a sarcastic utterance such as *"My laptop has an awesome battery life that lasts for 15 minutes"*, the tool should recognize that the speaker is expressing a negative sentiment toward the battery life of the laptop, even though, it has a positive sentiment word 'awesome' in it. Similarly the opinion mining tool should identify the negative opinion of the user expressed toward the entity "battery". Sarcasm target identification can also benefit natural language generation; for example, after detection of entity toward which a negative sentiment is expressed in a sarcastic text, a natural language generation system will have more context to generate a response. Similarly, a sentiment analysis tool will flag the sentiment in a sarcastic text toward the correct aspect of a product or the entity which can help to build a more accurate product review. In this paper we present a novel method for sarcasm target identification with the help of deep learning techniques in addition to a set of socio-linguistic features.

There is a lot of literature that deal with the sarcasm detection in text (Joshi et al., 2017), but only Joshi et al. (2018) have addressed the problem of sarcasm target identification. The sarcasm target is defined as the entity or situation that is being mocked or ridiculed at in the sarcastic text. Formally, the sarcasm target identification is defined as the task of building a system that takes a sarcastic text (book snippets, tweets etc.) as input, and either identifies a subset of words as sarcasm targets or outputs a fall-back label 'outside' if the target is not present in the text. For example in the sarcastic text *"I love to be ignored"*, the target is *"I"*. We consider two assumptions in this work as the same has been done in the baseline, – (a) every sarcastic text has at least one sarcasm target as this holds true by the definition of sarcasm, and, (b) the notion of sarcasm target is applicable for sarcastic texts only.

Sarcasm target identification is a difficult task, the primary reasons being,

- **Multiple candidate phrases**: There can be multiple target candidate phrases present in the sarcastic text. For example, in the sarcastic text, "The laptop heats up so much that I strongly recommend chefs to use it as a cook-top", the target candidates could be 'chefs', 'cook-top' and 'laptop'; however, only 'laptop' is ridiculed in this sentence.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6336–6342,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

- **Multiple sarcasm targets**: There can be multiple sarcasm target phrases present in the sentence. For example, in the sarcastic text, "I used to be a middle-of-the-road kid, but now with my freaky looks I'm definitely an outsider. Hooray.", have two sarcasm targets, i.e., 'my freaky looks' and 'I'.
- **Absence of any target**: It is also possible that no sarcasm target is present at all in the sarcastic text. For example in the sarcastic text "Oh, and I suppose the apples ate the cheese." the sarcasm target has to be labelled as 'outside'.

The main contributions and results of this paper can be summarized as,

- An empirical study of the socio-linguistic features that are highly significant in identifying sarcasm targets.
- A novel deep learning framework augmented with socio-linguistic features to detect sarcasm targets in sarcastic texts. We achieve a huge improvement over Joshi et al. (2018) in sarcasm target detection in terms of the evaluation metrics – exact match and dice score.

In this paper our main motive was to establish that deep neural machinery can be effectively married with socio-linguistic features to detect sarcasm targets. This exercise was a proof of concept to show that this marriage is indeed useful. The code we developed for this work is made freely available[1].

## 2 Related works

Most of the papers in the area of computational sarcasm address the problem of sarcasm detection, i.e., classification of a text as sarcastic or non-sarcastic. Joshi et al. (2017) present a compilation of past works including the datasets, approaches, issues and trends in automatic sarcasm detection. They observe mainly three approaches to the sarcasm detection problem – semi-supervised extraction of sarcastic patterns (Tsur et al., 2010; Ptáček et al., 2014; Bouazizi and Ohtsuki, 2015; Riloff et al., 2013; Joshi et al., 2015), use of hashtag based supervision (Davidov et al., 2010; Abercrombie and Hovy, 2016), and use of contextual information for sarcasm detection (Hazarika et al., 2018; Wallace et al., 2014; Rajadesingan et al., 2015). Recently, Tay et al. (2018) presented an attention-based neural model to explicitly model contrast and incongruity. Kolchinski and Potts (2018) presented two methods for representing authors in the context of textual sarcasm detection; they show that augmenting a bidirectional RNN with these representations improves performance in sarcasm detection. Ghosh and Muresan (2018) did a thorough analysis of sarcasm markers in social media platforms like Twitter and Reddit; in their study they found that in Twitter while emoticons or emojies are the most discriminative markers to recognize sarcastic/ironic utterances, for Reddit the morphological markers (e.g., interjections, tag questions) are the most discriminative. In socio-linguistic literature even though there are many studies that observe propagation of hate speech (Ribeiro et al., 2018; Salminen et al., 2018) and abusive behaviour (Founta et al., 2018; Maity et al., 2018; Mathew et al., 2019a,b) in social media an in-depth analysis of how sarcastic message travels in social networks and how tweets around the targets behave is an area which social scientists need to investigate. To the best of our knowledge, only Joshi et al. (2018) addresses the problem of sarcasm target identification. This problem attempts to identify the entity toward which sentiment is expressed in a sentence which in turn can have a lot of applications. Our objective here is to leverage recent deep learning methods to escalate the overall performance on this task.

## 3 Dataset

We consider the dataset released by Joshi et al. (2018) for our experiments. The dataset has two types of sarcastic text - - book snippets and tweets. There are 224 book snippets and 506 tweets present in the data. The sarcasm targets present in these book snippets and tweets are manually annotated by three well experienced linguists who have at least five years of linguistic annotation experience for tasks such as sentiment analysis, word sense disambiguation and other related works. For the book snippets the average length of the sarcasm target is 1.6 words while it is 2.08 words for the tweets. The average length of the whole snippets is 27.74 words whereas for the tweets this is 12.97 words. For annotation, the annotators are given a bunch of sarcastic texts and asked to identify which words represent the target that the author is mocking? In case the annotators do not find specific words in the text that corre-

[1]Code:https://github.com/Srijanb97/Sarcasm_Target_Detection-EMNLP-

spond to a target, they label it as 'outside'.

## 4 Socio-linguistic features

In this section, we present various socio-linguistic features that show statistically significant differences between the words corresponding to the sarcasm targets and the rest of the words in the sarcastic text. The results are shown in Table 1. Some of the observations are,

- The distribution of location (LOC) and organisation (ORG) named entities are significantly different for the sarcasm target words compared to the other words ($p < 0.001$).
- The distribution of some of the POS tags (nouns, verbs, adjectives and modifiers) are significantly different for the target words compared to the other words.
- We calculate the LIWC[2] and Empath (Fast et al., 2016) category fractional distributions across the target and the other words in the snippets and tweets. Certain categories as noted in the table are significantly different. The LIWC and Empath dictionary has many pre-defined categories (e.g., 'social', 'family' etc.). Analysis using these dictionaries has been done on different collection of tweets in many past research (Fink et al., 2012; Schwartz et al., 2013; Maity et al., 2016) which forms our primary motivation for this study.

## 5 Methodology

The architecture of our proposed system is shown in Figure 1. The input to our system is a sarcastic text concatenated with a dummy word at the end of the sentence. We proceed with the hypothesis that each word is a potential candidate to be a sarcasm target. Thus for each word in the sentence we create three components, (i) left context, (ii) right context, and (iii) a word representation for itself. Suppose the input sarcastic sentence is represented by a sequence of words $w_1, w_2 ... w_N$, $w_{N+1}$, where $w_{N+1}$ is a dummy word. We append a start token and an end token respectively at the beginning and the end of this sentence. These two tokens are never be considered as center word, but act as the left context for the first ($w_1$) word and the right context for last dummy word ($w_{N+1}$) respectively. Thus, for a word $w_K$, where $1 <= K <= N + 1$, the left context is defined as $[< start > w_1 : w_{K-1}]$ while the right context is defined as $[w_{K+1} : w_{N+1} < end >]$. Each word in the left context, right context and the central word are passed through an embedding layer to initialize them through pre-trained embeddings. We experiment with various pre-trained word embeddings like Glove, fast-text, elmo, BERT etc. The word representations are then passed to a LSTM or bidirectional LSTM (Bi-LSTM) layer or a target dependent LSTM (TD-LSTM) layer. In case of unidirectional LSTM (simple LSTM) layer, we keep the flow of hidden vectors in left context and right context as toward the center. Next we concatenate the hidden vectors of rightmost LSTM cell in left context, the central word LSTM cell hidden vector and the hidden vector of leftmost LSTM cell in right context, and pass them to a dense layer. In case of Bi-LSTM we concatenate both the forward and backward hidden vectors at each component before concatenating them again across the components. The dense representation is then concatenated with socio-linguistic features as we have obtained for the word $w_k$, and passed to a linear layer with sigmoid activation function, for the classification of the center word as sarcasm target or not.

[2] http://www.liwc.net/comparison.php

## 6 Experiments and results

### 6.1 Evaluation metrics

We consider two evaluation metrics – (i) exact match accuracy, and (ii) dice score, as has been also used in the baseline method (see (Joshi et al., 2018) for definitions).

### 6.2 Baselines

The baseline as described in Joshi et al. (2018) consists of two extractors joined by an integrator. The two extractors are (i) rule based, and (ii) statistics based. While the rule based extractor extracts candidate words for sarcasm target based on nine syntactic rules, the statistical extractor takes features such as lexical, POS tag, polarity, pragmatic features etc., and passes them to a classifier for the candidate word selection. The selected candidate words are then given as input to the integrator module, which is a hybrid 'AND' or 'OR' module, to select the final set of words as sarcasm targets.

### 6.3 Model setup and results

**The setup**: All the results reported are on 3-fold cross validation. Models were trained with Adam Optimizer having a learning rate $1e-5$ and a batch

| Catrgory | Features for snippets | Features for tweets |
|---|---|---|
| NER | LOC(***), ORG(***) | LOC(***), ORG(***) |
| POS | CC(***), IN(***), MD(***), NNP(***), PRP(***), PRP$(***), RB(***), VB(***), VBD(***), VBP(***), VBZ(***), JJ(**), NN(**), NNS(**), TO(**), UH(**), WP(*) | IN(***), MD(***), NN(***), NNP(***), NNS(***), PRP$(***), RB(***), UH(***), VB(***), VBG(***), VBP(***), VBZ(***), CC(**), TO(**), VBD(**), JJ(*), JJR(*), JJS(*), RP(*) |
| Empath | Appearance(***), Feminine(***), White_Collar_Job(***), Beauty(*), Cleaning(*), Exotic(*), Farming(*), Occupation(*), Violence(*) | Affection(***), Ancient(***), Beach(***), Car(***), Cleaning(***), College(***), Domestic_Work(***), Driving(***), Eating(***), Economics(***), Family(***), Government(***), Health(***), Home(***), Love(***), Medical_Emergency(***), Meeting(***), Morning(***), Occupation(***), Ocean(***), Office(***), Optimism (***), Poor(***), Positive_Emotion(***), Reading(***), Sailing(***), School(***), Science(***), Sports(***), Swimming(***), Traveling(***), Water(***), White_Collor_Job(***), Work(***), Vehicle(***), Art(**), Blue_Collar_Job(**), Business(**), Clothing(**), Cooking(**), Dance(**), Exotic(**), Fabric(**), Feminine(**), Friends(**), Hygiene(**), Liquid(**), Musical(**), Law(**), Plant(**), Restaurant(**), Ship(**), Toy(**),Achievement(*), Air_Travel(*), Animal(*), Attractive(*), Childish(*), Children(*), Contentment(*), Exercise(*), Fashion(*), Furniture(*), Help(*), Leader(*), Leisure(*), Messaging(*), Music(*), Nervousness(*), Politics(*), Shopping(*), Sleep(*), Technology(*), Violence(*), Warmth(*), Weather(*), Wedding(*) |
| LIWC | Adverbs(***), Affect(***), AuxVb(***), CogMech(***), Conj(***), Excl(***), Humans(***), Negate(***), Past(***), Posemo(***), Ppron(***), Prep(***), Present(***), Pronoun(***), Relativ(***), SheHe(***), Social(***), Space(***), Tentat(***), They(***), Time(***), Verbs(***), Cause(**), Discrep(**), Funct(**), Future(**), Hear(**), Incl(**), Motion(**), Achiev(*), Article(*), Body(*), Death(*), Family(*), Insight(*), Percept(*), Quant(*) | Adverbs(***), Affect(***), AuxVb(***), Body(***), CogMech(***), Conj(***), Discrep(***), Excl(***), Family(***), Funct(***), Health(***), Home(***), Humans(***), Insight(***), Leisure(***), Past(***), Posemo(***), Ppron(***), Prep(***), Present(***), Pronoun(***), Sad(***), Social(***), Verbs(***), Work(***), Assent(**), Bio(**), Cause(**), Future(**), Ingest(**), Negate(**), Quant(**), Sexual(**), Certain(*), Friends(*), I(*), Incl(*), SheHe(*) |

Table 1: Features that are significantly different for the target words compared to the other words. Features for which $p < 0.05$ ($p < 0.01$ and $p < 0.001$) is represented by (*) {(**), (***)} respectively.

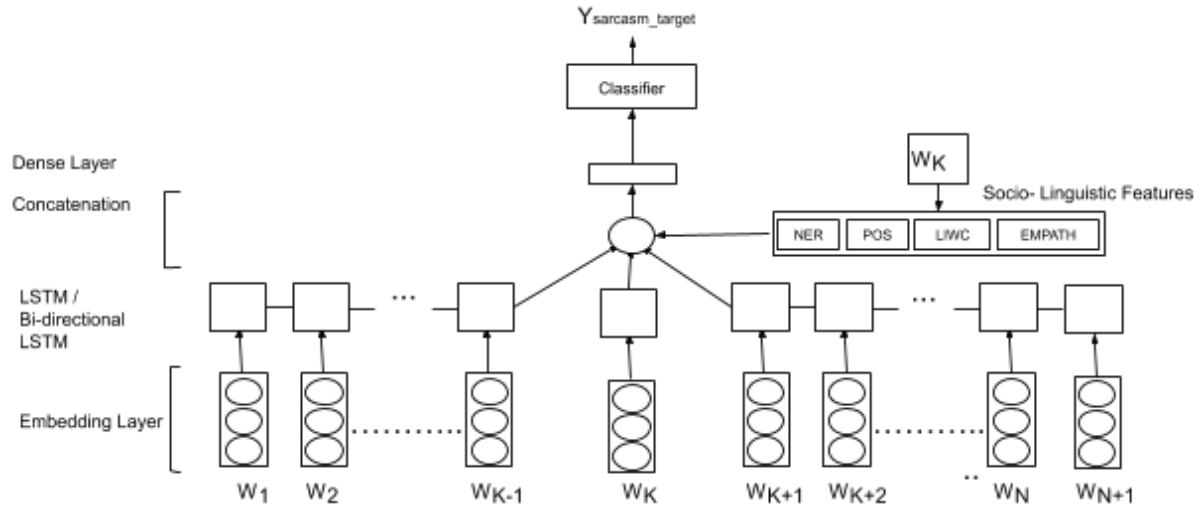


Figure 1: Architecture of the proposed system. $w_K$ is the center word to be classified as target or not, $[< start > w_1 : w_{K-1}]$ is the left context and $[w_{K+1} : w_{N+1} < end >]$ is the right context.

size of 64. Best results were obtained with the Elmo embedding as initialization.

**Results**: We report the exact match and the dice score obtained from different variants of our model and compare them with the baseline in Table 2. We note that all the variants of our model outperforms the baseline approach by a large margin. Among non-augmented models, the variant with Bi-LSTM layer performs the best in most of the metrics. The dice score for the book snippets data is best when TD-LSTM is used. The augmentation of socio-linguistic features (BiLSTM layer + slf) the performance further leading to the establishment of new state-of-the-art in sarcasm target detection. In addition, for our model variants we also report the macro and micro F1-scores in Table 3. Once again the Bi-LSTM is indicative of the best performance in majority of cases. The macro-F1 and micro-F1 increases further for book snippets when augmented with socio-linguistic features.

| Model | $EM_T$ | $DS_T$ | $EM_S$ | $DS_S$ |
|---|---|---|---|---|
| Baseline: AND | 13.45 | 20.82 | 16.51 | 21.28 |
| Baseline: OR | 9.09 | 39.63 | 7.01 | 32.68 |
| LSTM layer | 26.01 | 82.84 | 23.37 | 87.57 |
| Bi-LSTM layer | 29.48 | 84.04 | 30.14 | 87.66 |
| TD-LSTM layer | 26.35 | 82.27 | 25.97 | 87.71 |
| Bi-LSTM layer + slf | **30.12** | **84.11** | **31.17** | **88.16** |

Table 2: Comparison our models with the baseline. T: tweets, S: snippets, sl: socio-linguistic features.

| Model | $F1_T^\mu$ | $F1_T^M$ | $F1_S^\mu$ | $F1_S^M$ |
|---|---|---|---|---|
| LSTM layey | 46.79 | 51.04 | 39.31 | 43.72 |
| Bi-LSTM layer | **54.74** | **59.19** | 48.27 | 45.36 |
| TD-LSTM layer | 41.59 | 49.28 | 47.23 | 46.42 |
| Bi-LSTM layer + slf | 50.30 | 55.76 | **48.30** | **48.54** |

Table 3: Macro (M) and micro ($\mu$) F1 scores for the different models.

## 7 Conclusion

In this work, we have presented a deep learning model for sarcasm target identification. We outperform the only available baseline by a large margin. We identify various socio-linguistic features that differentiate the target text from the rest of the snippet/tweet. When these additional socio-linguistic features are fused into our deep learning framework they seem to improve performance for both snippets and tweets establishing new state-of-the-art for this problem.

## References


Gavin Abercrombie and Dirk Hovy. 2016. Putting sarcasm detection into context: The effects of class imbalance and manual labelling on supervised machine classification of twitter conversations. In *Proceedings of the ACL 2016 Student Research Workshop*, pages 107–113.

Mondher Bouazizi and Tomoaki Ohtsuki. 2015. Sarcasm detection in twitter:" all your products are incredibly amazing!!!"-are they really? In *2015 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Semi-supervised recognition of sarcastic sentences in twitter and amazon. In *Proceedings of the fourteenth conference on computational natural language learning*, pages 107–116. Association for Computational Linguistics.

Ethan Fast, Binbin Chen, and Michael S Bernstein. 2016. Empath: Understanding topic signals in large-scale text. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 4647–4657. ACM.

Clay Fink, Jonathon Kopecky, and Maksym Morawski. 2012. Inferring gender from the content of tweets: A region specific example. In *ICWSM '12*, pages 459–462.

Antigoni-Maria Founta, Constantinos Djouvas, Despoina Chatzakou, Ilias Leontiadis, Jeremy Blackburn, Gianluca Stringhini, Athena Vakali, Michael Sirivianos, and Nicolas Kourtellis. 2018. Large scale crowdsourcing and characterization of twitter abusive behavior. In *ICWSM*.

Debanjan Ghosh and Smaranda Muresan. 2018. "with 1 follower i must be awesome : P". exploring the role of irony markers in irony recognition. In *ICWSM*.

Devamanyu Hazarika, Soujanya Poria, Sruthi Gorantla, Erik Cambria, Roger Zimmermann, and Rada Mihalcea. 2018. Cascade: Contextual sarcasm detection in online discussion forums. In *COLING*.

Aditya Joshi, Pushpak Bhattacharyya, and Mark James Carman. 2017. Automatic sarcasm detection: A survey. *ACM Comput. Surv.*, 50:73:1–73:22.

Aditya Joshi, Pranav Goel, Pushpak Bhattacharyya, and Mark Carman. 2018. Sarcasm target identification: Dataset and an introductory approach. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*.

Aditya Joshi, Vinita Sharma, and Pushpak Bhattacharyya. 2015. Harnessing context incongruity for sarcasm detection. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 757–762.

Y. Alex Kolchinski and Christopher Potts. 2018. Representing social media users for sarcasm detection. In *EMNLP*.

Suman Kalyan Maity, Aishik Chakraborty, Pawan Goyal, and Animesh Mukherjee. 2018. Opinion conflicts: An effective route to detect incivility in twitter. In *ACM CSCW*.

Suman Kalyan Maity, Ritvik Saraf, and Animesh Mukherjee. 2016. #bieber + #blast = #bieberblast: Early prediction of popular hashtag compounds. In *CSCW '16*, pages 50–63.

Binny Mathew, Ritam Dutt, Pawan Goyal, and Animesh Mukherjee. 2019a. Spread of hate speech in online social media. In *ACM WebSci*.

Binny Mathew, Punyajoy Saha, Hardik Tharad, Subham Rajgaria, Prajwal Singhania, Suman Kalyan Maity, Pawan Goyal, and Animesh Mukherjee. 2019b. Thou shalt not hate: Countering online hatespeech. In *ICWSM*.

Bo Pang, Lillian Lee, et al. 2008. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, 2(1–2):1–135.

Tomáš Ptáček, Ivan Habernal, and Jun Hong. 2014. Sarcasm detection on czech and english twitter. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 213–223.

Ashwin Rajadesingan, Reza Zafarani, and Huan Liu. 2015. Sarcasm detection on twitter: A behavioral modeling approach. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, pages 97–106. ACM.

Manoel Horta Ribeiro, Pedro H. Calais, Yuri A. Santos, Virgílio A. F. Almeida, and Wagner Meira. 2018. Characterizing and detecting hateful users on twitter. In *ICWSM*.

Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert, and Ruihong Huang. 2013. Sarcasm as contrast between a positive sentiment and negative situation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 704–714.

Joni O. Salminen, Hind Almerekhi, Milica Milenkovic, Soon-Gyo Jung, Jisun An, Haewoon Kwak, and Bernard J. Jansen. 2018. Anatomy of online hate: Developing a taxonomy and machine learning models for identifying and classifying hate in online news media. In *ICWSM*.

Hansen Andrew Schwartz, Johannes C. Eichstaedt, Margaret L. Kern, Lukasz Dziurzynski, Richard E. Lucas, Megha Agrawal, Gregory J. Park,

Shrinidhi K. Lakshmikanth, Sneha Jha, Martin E. P. Seligman, and Lyle Ungar. 2013. Characterizing geographic variation in well-being using tweets. In *ICWSM '13*, pages 583–591.

Yi Tay, Anh Tuan Luu, Siu Cheung Hui, and Jian Su. 2018. Reasoning with sarcasm by reading in-between. In *ACL*.

Oren Tsur, Dmitry Davidov, and Ari Rappoport. 2010. Icwsma great catchy name: Semi-supervised recognition of sarcastic sentences in online product reviews. In *Fourth International AAAI Conference on Weblogs and Social Media*.

Byron C Wallace, Laura Kertz, Eugene Charniak, et al. 2014. Humans require context to infer ironic intent (so computers probably do, too). In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 512–516.

# In Plain Sight: Media Bias Through the Lens of Factual Reporting

**Lisa Fan**[1,*] **Marshall White**[1,*] **Eva Sharma**[1] **Ruisi Su**[1]
**Prafulla Kumar Choubey**[2] **Ruihong Huang**[2] **Lu Wang**[1]
[1] Khoury College of Computer Sciences, Northeastern University, Boston, MA 02115
[2] Department of Computer Science and Engineering, Texas A&M University
{fan.lis, white.mars, sharma.ev, su.ruis}@husky.neu.edu
{prafulla.choubey, huangrh}@tamu.edu, luwang@ccs.neu.edu

## Abstract

The increasing prevalence of political bias in news media calls for greater public awareness of it, as well as robust methods for its detection. While prior work in NLP has primarily focused on the *lexical bias* captured by linguistic attributes such as word choice and syntax, other types of bias stem from the actual content selected for inclusion in the text. In this work, we investigate the effects of *informational bias*: factual content that can nevertheless be deployed to sway reader opinion. We first produce a new dataset, BASIL, of 300 news articles annotated with 1,727 bias spans[1] and find evidence that informational bias appears in news articles more frequently than lexical bias. We further study our annotations to observe how informational bias surfaces in news articles by different media outlets. Lastly, a baseline model for informational bias prediction is presented by fine-tuning BERT on our labeled data, indicating the challenges of the task and future directions.

## 1 Introduction

News media exercises the vast power of swaying public opinion through the way it selects and crafts information (De Vreese, 2004; DellaVigna and Gentzkow, 2010; McCombs and Reynolds, 2009; Perse, 2001; Reynolds and McCombs, 2002). Multiple studies have identified the correlation between the increasing polarization of media and the general population's political stance (Gentzkow and Shapiro, 2010, 2011; Prior, 2013), underscoring the imperative to understand the nature of news bias and how to accurately detect it.

In the natural language processing community, the study of bias has centered around what we term

**Main event:** Democratic presidential candidates ask to see full Mueller report
**Main targets:** Donald Trump, Democratic candidates

**HPO:** Democrats want access to special counsel Robert Mueller's investigation into Russian interference in the 2016 presidential election [*before President Donald Trump has a chance to interfere.*]$_{\text{Trump}}$ ... Sen. Mark Warner said in a statement: [*"Any attempt by the Trump Administration to cover up the results of this investigation into Russia's attack on our democracy would be unacceptable."*]$_{\text{Trump}}$

**FOX:** Democratic presidential candidates [**wasted no time**]$_{\text{Dems}}$ Friday evening demanding the immediate public release of the long-awaited report from Robert S. Mueller III. ... Several candidates, in calling for the swift release of the report, also [*sought to gather new supporters and their email addresses*]$_{\text{Dems}}$ by putting out [**"petitions"**]$_{\text{Dems}}$ calling for complete transparency from the Justice Department.

**NYT:** And on Saturday, one day before Attorney General William Barr released a short summary of Mueller's findings, former Texas Rep. Beto O'Rourke charged on the campaign trail in South Carolina that you [*"have a president, who in my opinion beyond the shadow of a doubt, sought to, however* [***ham-handedly,***]$_{\text{Trump}}$ *collude with the Russian government—a foreign power—to undermine and influence our elections."*]$_{\text{Trump}}$

Figure 1: Examples of negative bias from Huffington Post (HPO), Fox News (FOX), and New York Times (NYT) discussing the same event. *Informational bias* and **lexical bias** are highlighted. The target of the bias is noted at the end of each span. Intermediary targets of indirect bias spans are underlined.

**lexical bias:** bias stemming from content realization, or how things are said (Greene and Resnik, 2009; Hube and Fetahu, 2019; Iyyer et al., 2014; Recasens et al., 2013; Yano et al., 2010). Such forms of bias typically do not depend on context outside of the sentence and can be alleviated while maintaining its semantics: polarized words can be removed or replaced, and clauses written in active voice can be rewritten in passive voice.

However, political science researchers find that news bias can also be characterized by decisions

*Equal contribution. Lisa Fan focused on annotation schema design and writing, Marshall White focused on data collection and statistical analysis.

[1]Dataset can be found at www.ccs.neu.edu/home/luwang/data.html.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6343–6349, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

made regarding content selection and organization within articles (Gentzkow et al., 2015; Prat and Strömberg, 2013). As shown in Figure 1, though all three articles report on the same event, Huffington Post (HPO) and Fox News (FOX) each frame entities of opposing stances negatively: HPO states an assumed future action of *Donald Trump* as a fact, and FOX implies *Democrats* are taking advantage of political turmoil. Such bias can only be revealed by gathering information from a variety of sources or by analyzing how an entity is covered throughout the article.

We define these types of bias as **informational bias:** sentences or clauses that convey information tangential, speculative, or as background to the main event in order to sway readers' opinions towards entities in the news. Informational bias often depends on the broader context of an article, such as in the second FOX annotation in Figure 1: gathering new supporters would be benign in an article describing political campaign efforts. The subtlety of informational bias can more easily affect an unsuspecting reader, which presents the necessity of developing novel detection methods.

In order to study the differences between these two types of bias, we first collect and label a dataset, **BASIL** (Bias Annotation Spans on the Informational Level), of 300 news articles with lexical and informational bias spans. To examine how media sources encode bias differently, the dataset uses 100 triplets of articles, each reporting the same event from three outlets of different ideology. Based on our annotations, we find that all three sources use more informational bias than lexical bias, and informational bias is embedded uniformly across the entire article, while lexical bias is frequently observed at the beginning.

We further explore the challenges in bias detection and benchmark BASIL using rule-based classifiers and the BERT model (Devlin et al., 2019) fine-tuned on our data. Results show that identifying informational bias poses additional difficulty and suggest future directions of encoding contextual knowledge from the full articles as well as reporting by other media.

## 2 Related Work

Prior work on automatic bias detection based on natural language processing methods primarily deals with finding sentence-level bias and considers linguistic attributes like word polarity (Recasens et al., 2013), partisan phrases (Yano et al., 2010), and verb transitivity (Greene and Resnik, 2009). However, such studies fail to take into consideration biases that depend on a larger context, which is what we try to address in this work.

Our work is also in line with *framing analysis* in social science theory, or the concept of selecting and signifying specific aspects of an event to promote a particular interpretation (Entman, 1993). In fact, informational bias can be considered a specific form of framing where the author intends to influence the reader's opinion of an entity. The relationship between framing and news is investigated by Card et al. (2015), in which news articles are annotated with framing dimensions like "legality" and "public opinion." BASIL contains richer information that allows us to study the purpose of "frames," i.e., how biased content is invoked to support or oppose the issue at hand.

Research in political science has also studied bias induced by the inclusion or omission of certain facts (Entman, 2007; Gentzkow and Shapiro, 2006, 2010; Prat and Strömberg, 2013). However, their definition of bias is typically grounded in how a reader perceives the ideological leaning of the article and news outlet, whereas our informational bias centers around the media's sentiment towards individual entities. Furthermore, while previous work mostly uses all articles published by a news outlet to estimate their ideology (Budak et al., 2016), we focus on stories of the same events reported by different outlets.

## 3 BASIL Dataset Annotation

Using a combination of algorithmic alignment and manual inspection, we select 100 sets of articles, each set discussing the *same event* from three different news outlets. 10 sets are selected for each year from 2010 to 2019. We use, in order from most conservative to most liberal, Fox News (FOX), New York Times (NYT), and Huffington Post (HPO). **Main events** and **main entities** are manually identified for each article prior to annotation. The political leanings of the main entities (*liberal*, *conservative*, or *neutral*) are also manually annotated. See the Supplementary for details.

**Annotation Process.** To compare how the three media sources discuss a story, annotators treat each article triplet as a single unit without knowing media information. Annotations are conducted on both *document-level* and *sentence-level*. On

| | | NYT | FOX | HPO | All |
|---|---|---|---|---|---|
| # Articles | | 100 | 100 | 100 | 300 |
| # Sentences | | 3,049 | 2,639 | 2,296 | 7,984 |
| # Words | | 91,818 | 70,024 | 62,321 | 224,163 |
| # Annotations | | 636 | 573 | 518 | 1,727 |
| Sentences / Article | | 30.5 ± 13.8 | 26.4 ± 10.2 | 23.0 ± 11.0 | 26.6 ± 12.2 |
| Words / Sentence | | 30.1 ± 14.0 | 26.5 ± 12.4 | 27.1 ± 12.5 | 28.1 ± 13.2 |
| Annotations / Article | | 6.4 ± 4.1 | 5.7 ± 3.8 | 5.2 ± 3.5 | 5.8 ± 3.8 |
| **Bias Type** | *Informational* | 468 (73.6%) | 421 (73.5%) | 360 (69.5%) | 1,249 (72.3%) |
| | *Lexical* | 168 (26.4%) | 152 (26.5%) | 158 (30.5%) | 478 (27.7%) |
| **Aim** | *Direct* | 574 (90.2%) | 485 (84.6%) | 462 (89.2%) | 1,521 (88.1%) |
| | *Indirect* | 62 (9.8%) | 88 (15.4%) | 56 (10.8%) | 206 (11.9%) |
| **Polarity** | *Positive* | 112 (17.6%) | 89 (15.5%) | 110 (21.2%) | 311 (18.0%) |
| | *Negative* | 524 (82.4%) | 484 (84.5%) | 408 (78.8%) | 1,416 (82.0%) |
| **Annotations in quotes** | | 205 (32.2%) | 299 (52.2%) | 217 (41.9%) | 721 (41.8%) |

Table 1: Descriptive statistics of the BASIL dataset. Mean and standard deviation shown where applicable. Annotation dimensions show raw counts and their percentage within the dimension in parentheses.

the document-level, annotators estimate the overall polarities of how the main event and main entities are covered, and rank the triplet's articles on the ideological spectrum with respect to one another. Before reading the articles, annotators specify their sentiment towards each main entity on a 5 point Likert scale.[2]

On the sentence-level, annotators identify spans of lexical and informational bias by analyzing whether the text tends to affect a reader's feeling towards one of the main entities. In addition to the main dimension of **bias type** (*lexical* or *informational*), each span is labeled with the **target** of the bias (a *main entity*), the bias **polarity** (*positive* or *negative* towards the target), the bias **aim** towards the main target (*direct* or *indirect*), and whether the bias is part of a **quote**. Bias aim investigates the case where the main entity is indirectly targeted through an intermediary figure (see the HPO example in Figure 1, where the sentiment towards the intermediary entity "Trump Administration" is transferred to the main target, "Donald Trump"). Statistics are presented in Table 1.

**Inter-annotator Agreement (IAA).** Two annotators individually annotate each article triplet before discussing their annotations together to resolve conflicts and agree on "gold-standard" labels. We measure span-level agreement according to Toprak et al. (2010), where we calculate the F1 score of span overlaps between two sets of annotations (details are in the Supplementary). Although the F1 scores of IAA are unsurprisingly low for this highly variable task, the score dramatically increases when agreement is calculated between individual annotations and the gold standard—from 0.34 to 0.70 for informational bias spans and from 0.14 to 0.56 for the sparser lexical spans, demonstrating the effectiveness of resolution discussions.

During the discussions, we noticed several trends that improved the quality of the gold standard annotations. First, the difficulty of being continually vigilant of one's own implicit bias would sometimes cause annotators to mark policies they disagreed with as negative bias (e.g., a liberal annotator might consider the detail that a politician supports an anti-abortion law as negative bias). Discussions allowed annotators to re-examine the articles from a more neutral perspective. Annotators also disagreed on whether a detail was relevant background or biasing peripheral information. During discussions, they performed comparisons to other articles of the triplet to make a final decision—if another article includes the same information, it is likely relevant to the main event. This strategy reiterates the importance of leveraging different media sources.

For overlapping spans, we find high agreement on the other annotation dimensions, with an average Cohen's $\kappa$ of 0.84 for polarity and 0.92 for target main entity.

## 4 Media Bias Analysis

### 4.1 Contrasting the Bias Types

**Informational bias outnumbers lexical bias.** As shown in Table 1, the large majority of annotations in BASIL are classified as informational bias. One explanation for its prevalence is that journalists typically make a conscious effort to avoid

[2]The likely effect of annotators' prior beliefs on their perception of bias will be investigated in future work.

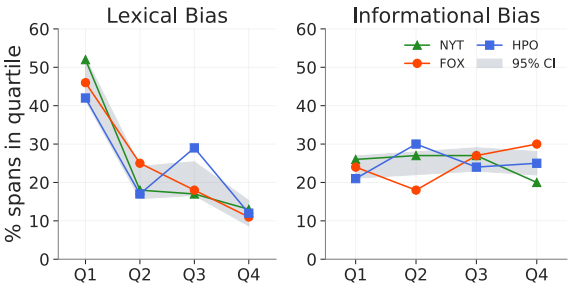


Figure 2: Distribution of lexical and informational bias spans found in each quartile of an article. The shaded area represents the 95% confidence interval for the three outlets combined.

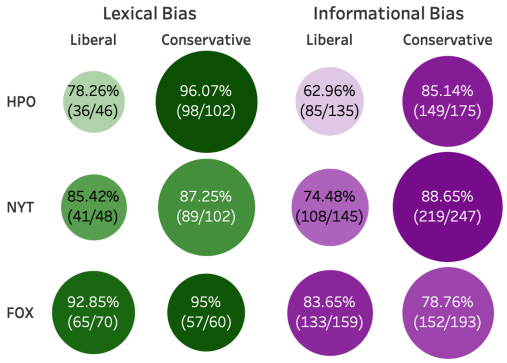


Figure 3: Percentage of bias spans with negative polarity toward targets of known ideology, grouped by media source, bias type, and target's ideology. For example, in all HPO articles, there are 46 lexical bias spans targeting liberals, 78.26% of which are negative. Larger circle means greater number of spans. Darker color corresponds to higher ratio of negative spans.

biased language, but can still introduce informational bias, either intentionally or through negligence.

For both bias types though, negative bias spans are much more pervasive than positive spans, mirroring the well-established paradigm that news media in general focuses on negative events (Niven, 2001; Patterson, 1996).

**Lexical bias appears early in an article.** We further study differences in characteristics between lexical and informational annotation spans and find that the two bias types diverge in positional distributions. Figure 2 shows that a disproportionate amount of lexical bias is located in the first quartile of articles. A visual inspection indicates that this may be attributed in part to media sources' attempts to hook readers with inflammatory speech early on (e.g., FOX: "Paul Ryan stood his ground against a barrage of Biden *grins, guffaws, snickers and interruptions*.").

In contrast, informational bias is often embedded in context, and therefore can appear at any position in the article. This points to a future direction of bias detection using discourse analysis.

**Quotations introduce informational bias.** We also find that almost half of the informational bias comes from within quotes (48.7%), highlighting a bias strategy where media sources select opinionated quotes as a subtle proxy for their own opinions (see the second HPO and first NYT annotations in Figure 1).

### 4.2 Portrayal of Political Entities

On the document-level, only 17 out of 100 article sets had reversed orderings (i.e. FOX marked as "more liberal" or HPO marked as "more conservative" within a triplet), confirming the ideological leanings identified in previous studies. Here, we utilize BASIL's span-level annotations to gain a more granular picture of how sources covering the same events control the perception of entities.

Concretely, we examine the polarity of bias spans with target entities of known ideology. As shown in Figure 3, for both bias types, the percentage and volume of negative coverage for liberal entities strongly correspond to the ideological leaning of the news outlet. Note that though NYT appears to have significantly more informational bias spans against conservatives than HPO, this is because NYT tends to have longer articles than the other two sources (see Table 1), and thus naturally results in more annotation spans by raw count.[3]

Moreover, the breakdown of lexical bias distinguishes FOX from the other two outlets: it comparatively has more negative bias spans towards liberals and fewer towards conservatives, even though all three outlets have more conservative entities than liberal ones across the 100 triplets (average of 99.0 conservatives, 72.7 liberals).

## 5 Experiments on Bias Detection

We study the bias prediction problem on BASIL as a binary classification task (i.e., whether or not a sentence contains bias) and as a BIO sequence tagging task (i.e., tagging the bias spans in one sentence at the token-level). We benchmark the performance with rule-based classifiers and the popular BERT model (Devlin et al., 2019) fine-tuned

[3]The proportion of annotations to article length are similar for all news outlets: one annotation for every 4.1 (for HPO), 4.5 (for FOX), or 4.6 (for NYT) sentences.

| Sentence-level | Precision | Recall | F1 |
|---|---|---|---|
| *Lexical Bias* | | | |
| BERT fine-tuning | 29.13 | 38.57 | 31.49 |
| *Informational Bias* | | | |
| TF-IDF | 25.81 | 26.23 | 26.02 |
| BERT fine-tuning | 43.87 | 42.91 | 43.27 |

| Token-level | Precision | Recall | F1 |
|---|---|---|---|
| *Lexical Bias* | | | |
| Polarity lexicon | 8.00 | 0.17 | 0.33 |
| Subjectivity lexicon | 28.00 | 0.65 | 1.28 |
| BERT fine-tuning | 25.60 | 29.32 | 25.98 |
| *Informational Bias* | | | |
| BERT fine-tuning | 25.56 | 14.78 | 18.71 |
| *Sentence-to-Token pipeline* | | | |
| Lexical bias | 12.00 | 13.64 | 12.77 |
| Informational bias | 9.52 | 5.08 | 6.63 |

Table 2: Sentence classification (top) and sequence tagging (bottom) results on lexical and informational bias prediction. For the BERT fine-tuning models, the mean from 10-fold cross validation is shown. The minimum standard deviation from cross validation for all BERT models is 3.36, the maximum is 12.44.

on informational and lexical bias spans separately.

**Training Details.** We utilize the pre-trained BERT-Base model and use the "Cased" version to account for named entities, which are important for bias detection. We run BERT on individual sentences[4] and perform stratified 10-fold cross validation. The validation set is used to determine when to stop training and a held out test set is used for the final evaluation of each fold. For the sentence-level classifiers, both our informational and lexical models use 6,819 sentences for training, 758 for validation, and 400 for testing.

Due to the sparsity of our data, we train and test our token-level models only on sentences containing bias spans of the relevant bias type. Our informational and lexical bias sequence taggers use a train/val/test split of 1,043/116/62 sentences and 383/42/23 sentences respectively. Results are shown in Table 2.

**Sentence-level Classifier.** The fine-tuned BERT is better at predicting informational bias than lexical bias, likely because informational bias is better captured by sentence-level context. As a baseline, we select the 4 sentences[5] in each article with the lowest average TF-IDF token scores as containing informational bias. The intuition is that sentences with different content than the rest of the article are more likely to contain extraneous information that the author chose to include to frame the story in a certain way. We find that this simple baseline performs relatively well considering the difficulty of the task, indicating the importance of explicitly modeling context. Future work may consider leveraging context in the entire article or articles on the same story by other media.

**Token-level Classifier.** From Table 2, we see that the BERT lexical sequence tagger produces better recall and F1 than the informational tagger, highlighting the additional difficulty of accurately identifying spans of informational bias. We also use the polarity and subjectivity lexicons from the MPQA website (Wilson et al., 2005; Choi and Wiebe, 2014) as a simple baseline for lexical bias tagging and find that these word-level cues, though widely used in prior sentiment analysis studies, are insufficient to fully capture lexical bias.

In order to evaluate token-level prediction on the larger original test set, we conduct a pipeline experiment with the fine-tuned BERT models where sentences predicted as containing bias by the best sentence-level classifier from cross validation are tagged by the best token-level model. The results reaffirm our hypothesis that while both tasks are extremely difficult, informational bias is more challenging to detect.

## 6 Conclusion

We presented a novel study on the effects of informational bias in news reporting from three major media outlets of different political ideology. Analysis of our annotated dataset, BASIL, showed the prevalence of informational bias in news articles when compared to lexical bias, and demonstrated BASIL's utility as a fine-grained indicator of how media outlets cover political figures. An experiment on bias prediction illustrated the importance of context when detecting informational bias and revealed future research directions.

## Acknowledgements

This research is supported in part by National Science Foundation through Grant IIS-1813341. We thank Philip Resnik, Nick Beauchamp, and Donghee Jo for their valuable suggestions on various aspects of this work. We are also grateful to the anonymous reviewers for their comments.

[4]BERT's maximum input length is 512 tokens, which is shorter than most articles in BASIL. We thus treat sentences as passages, rather than using text of fixed length.

[5]BASIL averages 4.1 informational bias spans per article.

## References


Ceren Budak, Sharad Goel, and Justin M Rao. 2016. Fair and balanced? quantifying media bias through crowdsourced content analysis. *Public Opinion Quarterly*, 80(S1):250–271.

Dallas Card, Amber E Boydstun, Justin H Gross, Philip Resnik, and Noah A Smith. 2015. The media frames corpus: Annotations of frames across issues. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 438–444.

Yoonjung Choi and Janyce Wiebe. 2014. +/-effectwordnet: Sense-level lexicon acquisition for opinion inference. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1181–1191.

Claes De Vreese. 2004. The effects of strategic news on political cynicism, issue evaluations, and policy support: A two-wave experiment. *Mass Communication & Society*, 7(2):191–214.

Stefano DellaVigna and Matthew Gentzkow. 2010. Persuasion: empirical evidence. *Annu. Rev. Econ.*, 2(1):643–669.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Robert M Entman. 1993. Framing: Toward clarification of a fractured paradigm. *Journal of communication*, 43(4):51–58.

Robert M Entman. 2007. Framing bias: Media in the distribution of power. *Journal of communication*, 57(1):163–173.

Matthew Gentzkow and Jesse M Shapiro. 2006. Media bias and reputation. *Journal of political Economy*, 114(2):280–316.

Matthew Gentzkow and Jesse M Shapiro. 2010. What drives media slant? evidence from us daily newspapers. *Econometrica*, 78(1):35–71.

Matthew Gentzkow and Jesse M Shapiro. 2011. Ideological segregation online and offline. *The Quarterly Journal of Economics*, 126(4):1799–1839.

Matthew Gentzkow, Jesse M Shapiro, and Daniel F Stone. 2015. Media bias in the marketplace: Theory. In *Handbook of media economics*, volume 1, pages 623–645. Elsevier.

Stephan Greene and Philip Resnik. 2009. More than words: Syntactic packaging and implicit sentiment. In *Proceedings of human language technologies: The 2009 annual conference of the north american chapter of the association for computational linguistics*, pages 503–511. Association for Computational Linguistics.

Christoph Hube and Besnik Fetahu. 2019. Neural based statement classification for biased language. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 195–203. ACM.

Mohit Iyyer, Peter Enns, Jordan Boyd-Graber, and Philip Resnik. 2014. Political ideology detection using recursive neural networks. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1113–1122.

Maxwell McCombs and Amy Reynolds. 2009. How the news shapes our civic agenda. In *Media effects*, pages 17–32. Routledge.

David Niven. 2001. Bias in the news: Partisanship and negativity in media coverage of presidents george bush and bill clinton. *Harvard International Journal of Press/Politics*, 6(3):31–46.

Thomas E Patterson. 1996. Bad news, bad governance. *The Annals of the American Academy of Political and Social Science*, 546(1):97–108.

Elizabeth M Perse. 2001. *Media effects and society*. Routledge.

Andrea Prat and David Strömberg. 2013. The political economy of mass media. *Advances in economics and econometrics*, 2:135.

Markus Prior. 2013. Media and political polarization. *Annual Review of Political Science*, 16:101–127.

Marta Recasens, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. 2013. Linguistic models for analyzing and detecting biased language. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1650–1659.

Amy Reynolds and Maxwell McCombs. 2002. News influence on our pictures of the world. In *Media effects*, pages 11–28. Routledge.

Cigdem Toprak, Niklas Jakob, and Iryna Gurevych. 2010. Sentence and expression level annotation of opinions in user-generated discourse. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 575–584. Association for Computational Linguistics.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of Human Language Technology Conference and Conference*

*on Empirical Methods in Natural Language Processing*.

Tae Yano, Philip Resnik, and Noah A Smith. 2010. Shedding (a thousand points of) light on biased language. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 152–158. Association for Computational Linguistics.

# Incorporating Label Dependencies in Multilabel Stance Detection

**William Ferreira**
Lucubo Ltd
william@lucubo.com

**Andreas Vlachos**
Dept. of Computer Science and Technology
University of Cambridge
Cambridge, UK
av308@cst.cam.ac.uk

## Abstract

Stance detection in social media is a well-studied task in a variety of domains. Nevertheless, previous work has mostly focused on multiclass versions of the problem, where the labels are mutually exclusive, and typically positive, negative or neutral. In this paper, we address versions of the task in which an utterance can have multiple labels, thus corresponding to multilabel classification. We propose a method that explicitly incorporates label dependencies in the training objective and compare it against a variety of baselines, as well as a reduction of multilabel to multiclass learning. In experiments with three datasets, we find that our proposed method improves upon all baselines on two out of three datasets. We also show that the reduction of multilabel to multiclass classification can be very competitive, especially in cases where the output consists of a small number of labels and one can enumerate over all label combinations.

## 1 Introduction

Stance detection is an established task in the computational linguistics community, and is typically concerned with whether an utterance (e.g. a tweet) expresses an attitude (often *positive*, *negative* or *neutral*) against a target such as an entity e.g. a politician (Hasan and Ng, 2013; Mohammad et al., 2016), or another utterance, e.g. a previous tweet in a thread (Zubiaga et al., 2016). Thus stance detection is an important task for analyzing discourse in online forums and social media platforms and is a component in assessing the veracity of claims (Kochkina et al., 2018).

When the stances are mutually exclusive as in the aforementioned cases, multiclass classification is an appropriate formulation for the task. Often, however, a text may express multiple stances simultaneously. Such cases need to be formulated as multilabel classification (Sorower, 2010), where an instance can receive multiple, non-mutually exclusive labels. The most commonly used approaches to multiclass classification treat the task by learning models for each label. However, such approaches do not model dependencies between the labels explicitly, i.e. that the presence of one label results in another becoming more or less likely.

| **Brexit Blog Corpus** (Simaki et al., 2018) |
|---|
| **Utterance**: rivalry between the us and china is inevitable but it needs to be kept within bounds that would preclude the use of military force. |
| **Stances**: certainty, contrariety, necessity, prediction |
| **US Election Twitter Corpus** (Sobhani et al., 2019) |
| **Utterance**: voters mean more than super delegates @sensanders corrupt -> #hillaryclinton spends millions on msm to discourage #americans voting #sanders |
| **Stances**: Clinton: AGAINST, Sanders: FAVOR |
| **Moral Foundations Twitter** (Dehghani et al., 2019) |
| **Utterance**: blatant racism in #colorado, #blacklivesmatter http://fb.me/1ibyxmswm |
| **Stances**: cheating, harm |

Figure 1: Examples from each of the datasets.

In this paper, we investigate multilabel stance detection in the context of three datasets: the Brexit Blog Corpus (BBC) (Simaki et al., 2018), the US Election Tweets Corpus (ETC) (Sobhani et al., 2019), and the Moral Foundations Twitter Corpus (MFTC) (Dehghani et al., 2019). Figure 1 shows examples from each dataset where the utterances have been annotated with multiple stances. In BBC and MFTC, each utterance is annotated with a variable number of stances, encoded as binary presence/absence of each possible stance. In ETC, each utterance has a three-way stance *FAVOR* (positive), *AGAINST* (negative), or *NONE* (neutral) for each of the candidates.

We show that it is possible to improve over baseline results that employ binary relevance and



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6350–6354,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

multitask learning, by incorporating label dependencies explicitly with the *cross-label dependency loss* (Yeh et al., 2017), originally introduced by Zhang and Zhou (2006). We also show that a reduction of multilabel to multiclass classification by considering all label combinations, also known as label powerset, can be very competitive, especially when the output consists of a small number of labels and one can enumerate all combinations, and verify our results with statistical significance testing. Finally, we improve on the best reported results on the ETC dataset.

## 2 Multilabel classification

The most commonly used approach to multilabel classification encodes the labels so that a single multilabel classification task is reduced to many sub-tasks learned independently. E.g. for BBC and MFTC binary models are learnt for each of the labels that predict the presence or absence of each label, hence the name *Binary Relevance* (BR), which has also been used in image classification (Boutell et al., 2004). It is straightforward to extend BR to handle tasks such as ETC where each position in the output can have more options than presence/absence, by using multiclass classifiers instead of binary ones. While it is possible to learn the models for the subtasks jointly using multitask learning (Ruder, 2017), this does not capture label dependencies in the output directly; instead it encourages layers of the model before the output to be learned to benefit all tasks simultaneously.

An alternative encoding method, *Label Powerset* (LP), captures dependencies explicitly: each label combination appearing in the training data is encoded as a new, unique label, reducing the task once again to a multiclass classification. However, LP can introduce an exponentially large number of new labels, potentially with few training instances, thus exacerbating class imbalance. Moreover, only those label combinations seen in the data will be available during training; this can be a particular issue when there is a shortage of representative training data.

BR encoding methods ignore label dependencies, and the LP method relies on encoding each label combination appearing in the training set as an explicit new label, both methods reducing the task to binary/multiclass classification. In what follows, we adopt a middle-ground between BR and LP methods by incorporating a notion of dependence between the labels in the targets.

## 3 Cross-label dependency loss

To capture the dependencies among labels in the output, we follow Yeh et al. (2017) and employ the cross-label dependency (XLD) loss between vectors $\mathbf{y}$ and $\hat{\mathbf{y}}$:

$$\text{XLD}(\mathbf{y}, \hat{\mathbf{y}}) := \frac{1}{|\mathbf{y}^0||\mathbf{y}^1|} \sum_{(p,q)\in\mathbf{y}^0\times\mathbf{y}^1} \exp(\hat{\mathbf{y}}_p - \hat{\mathbf{y}}_q)$$

where $\mathbf{y}$ denotes a vector of true (binary) labels of dimension $n$, $\hat{\mathbf{y}}$ a vector of predicted label probabilities, $\mathbf{y}^1$ are the indices of the 1-labelled elements of $\mathbf{y}$, $\mathbf{y}^0$ are the indices of the 0-labelled elements, and $\hat{\mathbf{y}}_p$ denotes the $p$th element of vector $\hat{\mathbf{y}}$. Minimising the cross-label dependency loss is equivalent to maximising the distance between 0- and 1-labelled targets, by penalising the model when it predicts label pairs that shouldn't co-exist for the instance. The intuition is similar to that of Bayesian Personalized Ranking in collaborative filtering (Rendle et al., 2009). We add XLD to cross-entropy loss to define an overall loss function $\text{L}(\mathbf{Y}, \hat{\mathbf{Y}})$:

$$\text{L}(\mathbf{Y}, \hat{\mathbf{Y}}) := \text{XEnt}(\mathbf{Y}, \hat{\mathbf{Y}}) + \alpha \sum_{i\in 1\ldots m} \text{XLD}(\mathbf{Y}_i, \hat{\mathbf{Y}}_i)$$

where XEnt is cross-entropy loss, $\mathbf{Y}_i$ denotes a row vector of dimension $n$, and $\alpha \geq 0$ is a hyperparameter controlling the contribution of cross-label dependency loss to the overall loss.

Extending XLD to the ETC dataset is slightly complicated by the fact that the labels have three possible values, so we cannot represent a set of target labels as a binary vector. Firstly, we encode the labels using a *one-hot encoding* binary representation, so for example, $AGAINST = 100$, $NONE = 010$ and $FOR = 001$. We then apply XLD between the two encoded target labels, $\mathbf{y}$ and $\mathbf{z}$, of each tweet, and their predicted label probabilities $\hat{\mathbf{y}}$ and $\hat{\mathbf{z}}$ respectively, as follows:

$$\text{XLD}(\mathbf{y}, \mathbf{z}, \hat{\mathbf{y}}, \hat{\mathbf{z}}) = \sum_{(p,q)\in\mathbf{y}^0\times\mathbf{z}^1} \exp(\hat{\mathbf{y}}_p - \hat{\mathbf{z}}_q)$$

The above definition is not symmetric since it compares the 0-labelled positions of the first target label with the 1-labelled positions of the second target label. We re-introduce the symmetry by defining the overall loss function as:

$$\text{L}(\mathbf{Y}, \mathbf{Z}, \hat{\mathbf{Y}}, \hat{\mathbf{Z}}) := \text{XEnt}(\mathbf{Y}, \hat{\mathbf{Y}}) + \text{XEnt}(\mathbf{Z}, \hat{\mathbf{Z}})$$

$$+ \alpha_L \sum_{i \in 1...m} \text{XLD}(\mathbf{Y}_i, \mathbf{Z}_i, \hat{\mathbf{Y}}_i, \hat{\mathbf{Z}}_i)$$
$$+ \alpha_R \sum_{i \in 1...m} \text{XLD}(\mathbf{Z}_i, \mathbf{Y}_i, \hat{\mathbf{Z}}_i, \hat{\mathbf{Y}}_i)$$

where $\alpha_L \geq 0$ and $\alpha_R \geq 0$ are hyper-parameters controlling the contribution of the left and right XLD loss across the targets, respectively.

## 4 Experimental setup

In our experiments, we use the following multilabel stance detection datasets. The BBC dataset (Simaki et al., 2018) contains 1,239[1] utterances from social media blogs. Each utterance is assigned multiple stances by expert annotators from a set of ten stances. The ETC dataset (Sobhani et al., 2019) consists of 3 sets of tweets, each associated with a pair of election candidates in the US 2016 Election: Donald Trump-Hillary Clinton (DT-HC), Donald Trump-Ted Cruz (DT-TC), and Hillary Clinton-Bernie Sanders (HC-BS), containing 1,722, 1,317 and 1,366 tweets respectively. The MFTC dataset consists of 35,108 tweets curated from six[2] distinct discourse domains, e.g. natural disasters, politics, etc. Each tweet is annotated with up to 10 labels of moral sentiment.

Hyper-parameter selection is done using 5-fold cross-validation (CV) on the training set of each dataset. For the BBC dataset, we split the data 80% into a training set, and 20% holdout test set. For the ETC dataset, we combine the training and validation sets already provided to perform CV, and report on the original test set. For the MFTC data set, we split the data 80% into a training set, and 20% holdout test set.

In BBC and MFTC we use the *Jaccard Similarity Score* (JSS) (Jaccard, 1902) as our scoring metric:

$$J(X, Y) = \frac{\mathbf{y^1} \cap \hat{\mathbf{y}}^\mathbf{1}}{\mathbf{y^1} \cup \hat{\mathbf{y}}^\mathbf{1}} \quad (1)$$

JSS gives credit for partial matches, but does not reward predicting the absence of labels, which is desirable as most labels for each instance are absent (e.g. 90% of the instances in BBC and MFTC have at most two labels). It is less harsh than accuracy (*Exact Match Ratio*) (Sorower, 2010), which requires the entire label combination to be predicted correctly. For the ETC dataset, where each tweet is tagged with exactly two stances (i.e. no absent labels), following Sobhani et al. (2019), we use the macro-averaged F1-score for FAVOR and AGAINST, as the scoring metric.

## 5 Results

In our experiments, we consider models that capture label dependencies explicitly as well as baselines that do not capture these. As our baselines, we consider binary relevance using FastText (FT) classifiers (Joulin et al., 2017) for each stance label in BBC/MFTC and politician in ETC, as well as a multi-task learning (MTL) approach (Ruder, 2017) where each of the classifiers becomes a task and they all operate on a shared hidden layer (hard parameter sharing). As models capturing dependencies, we considered three options: the combination of the cross label dependency loss with MTL (MTL-XLD), and the combinations of label powerset with FT and MTL (FT-LP and MTL-LP respectively). For the latter, each label combination becomes a task learned jointly with the rest. Further details on all models and parameter tuning are in the supplementary material[3].

In Table 1 we report the test set results for all models. The results for the MFTC dataset are averaged across the six discourse domains. Overall, MTL-LP is the best performing multilabel classification method across all the datasets. MTL-LP is also better than the best performing model Seq2Seq reported in Sobhani et al. (2019) for the ETC dataset. MTL-XLD improves on the baseline models for the BBC and MFTC datasets, but performs slightly worse than MTL on the ETC dataset. We note that our results for the BBC and MFTC datasets are not directly comparable with previous work on BBC (Simaki et al., 2017) and MFTC (Dehghani et al., 2019), since we consider the full set of labels, whereas previous work removed those that were sparser. Our reimplementation of the logistic regression model of Simaki et al. (2017), as an additional baseline, resulted in poor performance in the BBC dataset (20 in JSS) and we did not consider it further.

### 5.1 Bootstrap training experiments

Reporting results on a held out test set is standard practice, but we also examine the results of

[1]The original dataset contained 1,682 utterances, but we removed duplicates occurring in the training and test sets.

[2]Originally seven but we dropped one domain after consultation with the authors.

[3]Code to reproduce our experiments is available here https://github.com/willferreira/multilabel-stance-detection

| | BBC | ETC | MFTC |
|---|---|---|---|
| FT-BR | 39.72 | 52.24 | 51.19 |
| MTL | 48.57 | 53.32 | 53.97 |
| FT-LP | 36.20 | 53.57 | 55.11 |
| MTL-LP | **55.60** | **55.37** | **62.98** |
| MTL-XLD | 51.33 | 52.22 | 60.94 |
| Sobhani (Seq2Seq) | NA | 54.81 | NA |

Table 1: Overall results for each dataset and model.

the classifiers for the BBC and MFTC datasets via bootstrapping (Gorman and Bedrick, 2019). We bootstrap 30 sample training sets from each dataset, by random sampling without replacement 80% of the data, retaining the 20% as a test set. For each sample, we train the MTL, MTL-LP, and best MTL-XLD model on each bootstrapped training set, and report the score on the test set. Summary statistics are shown in Tables 2 and 3. Using Welch's t-test[4], we cannot reject the null hypothesis (p=0.094) that the mean scores for the MTL-LP and MTL-XLD models on BBC are the same, but we can reject the null hypothesis that either of them are the same as the MTL baseline. For the MFTC Baltimore discourse domain, we cannot reject the null hypothesis (p=0.84) that the mean scores for the MTL-LP and MTL-XLD are the same, but we can reject the null that either of them are the same as the MTL baseline. For the remaining domains, we can reject the null hypothesis that the MTL-LP and MTL-XLD means are the same for ALM (p=6.74e-15), BLM (p=2.3e-13), Davidson (p=1.8e-24), Election (p=1.6e-17) and MeToo (p=0.0021). We can also reject the null, for each domain, that either MLT-XLD or MLT-LP have the same means as the MTL baseline.

| | MTL | MTL-XLD | MTL-LP |
|---|---|---|---|
| mean | 45.27 | 51.58 | 52.51 |
| std | 2.01 | 1.92 | 2.28 |

Table 2: Summary statistics for bootstrap results on BBC dataset.

### 5.2 Learning curve experiments

For the BBC and MFTC datasets, we construct the learning curve for the MTL-LP model against the MTL-XLD model. We sample increasingly larger fractions of the training sets, train the models on these fractions, and record the scoring metric on the original holdout test set. The learning curve for BBC is shown in Figure 2, from which we can see that the MTL-XLD model is superior to MTL-LP until the training dataset size is approximately 70% of the original, after which MTL-LP scores higher than MTL-XLD. The remaining learning curves for MFTC are given in the supplementary material, and show that for discourse domains ALM, BLM, Davidson, Election and MeToo, MTL-LP is superior to MTL-XLD at all training set sizes, however MTL-XLD is superior to MTL-LP for the Baltimore domain.

[4] https://en.wikipedia.org/wiki/Welch%27s_t-test

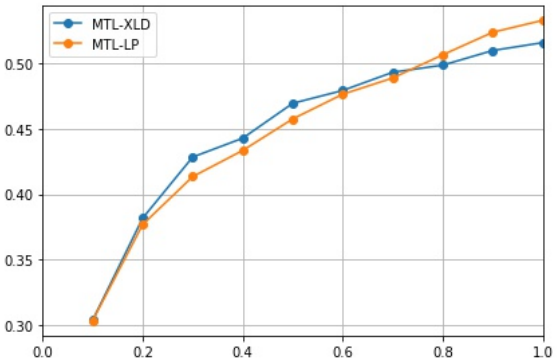


Figure 2: BBC: bootstrapped learning curve.

| | | ALM | BLM | B'more |
|---|---|---|---|---|
| MTL | mean | 65.74 | 75.56 | 40.45 |
| | std | 1.71 | 1.11 | 1.82 |
| MTL-LP | mean | **72.44** | **80.77** | 48.45 |
| | std | 1.53 | 0.86 | 1.39 |
| MTL-XLD | mean | 69.82 | 78.03 | **48.52** |
| | std | 0.90 | 0.87 | 1.16 |
| | | D'son | Election | MeToo |
| MTL | mean | 34.29 | 60.41 | 40.92 |
| | std | 3.41 | 1.78 | 1.55 |
| MTL-LP | mean | **50.76** | **68.22** | **49.56** |
| | std | 2.32 | 0.98 | 0.94 |
| MTL-XLD | mean | 48.76 | 65.43 | 48.75 |
| | std | 3.04 | 1.04 | 1.02 |

Table 3: Summary statistics for bootstrap results on MFTC dataset domains.

## 6 Conclusions

In this paper, we focused on multilabel stance detection and presented experiments on three

datasets. We demonstrated that taking label dependencies into account improves the performance of classification-based and multi-task baselines. In future work, we will explore how to integrate the textual descriptions of the labels in our our approach which has been shown to be beneficial in the case of large label sets (Mullenbach et al., 2018).

## Acknowledgements

We would like to thank Daniel King for conducting initial experiments on the BBC corpus during his MSc thesis at the University of Sheffield.

## References

Matthew Boutell, Jiebo Luo, Xipeng Shen, and Christopher Brown. 2004. Learning multi-label scene classification. *Pattern Recognition*, 37:1757–1771.

Morteza Dehghani, Joseph Hoover, Gwenyth Portillo-Wightman, Leigh Yeh, Shreya Havaldar, Ying Lin, Aida M Davani, Brendan Kennedy, Mohammad Atari, Zahra Kamel, and et al. 2019. Moral foundations twitter corpus.

Kyle Gorman and Steven Bedrick. 2019. We need to talk about standard splits. In *Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Kazi Saidul Hasan and Vincent Ng. 2013. Stance classification of ideological debates: Data, models, features, and constraints. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1348–1356, Nagoya, Japan. Asian Federation of Natural Language Processing.

P Jaccard. 1902. Lois de distribution florale dans la zone alpine. *Bull Soc Vaudoise Sci Nat*, 38:69–130.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics.

Elena Kochkina, Maria Liakata, and Arkaitz Zubiaga. 2018. All-in-one: Multi-task learning for rumour verification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3402–3413, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. SemEval-2016 task 6: Detecting stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 31–41, San Diego, California. Association for Computational Linguistics.

James Mullenbach, Sarah Wiegreffe, Jon Duke, Jimeng Sun, and Jacob Eisenstein. 2018. Explainable prediction of medical codes from clinical text. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1101–1111, New Orleans, Louisiana. Association for Computational Linguistics.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 452–461, Arlington, Virginia, United States. AUAI Press.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *CoRR*, abs/1706.05098.

Vasiliki Simaki, Carita Paradis, and Andreas Kerren. 2017. Stance classification in texts from blogs on the 2016 british referendum. In *SPECOM*.

Vasiliki Simaki, Carita Paradis, and Andreas Kerren. 2018. Evaluating stance-annotated sentences from the brexit blog corpus: A quantitative linguistic analysis. *ICAME Journal*, 42:133–165.

Parinaz Sobhani, Diana Inkpen, and Xiaodan Zhu. 2019. Exploring deep neural networks for multi-target stance detection. *Computational Intelligence*, 35(1):82–97.

Mohammad S Sorower. 2010. A literature survey on algorithms for multi-label learning. Technical report, Oregon State University, Corvallis.

Chih-Kuan Yeh, Wei-Chieh Wu, Wei-Jen Ko, and Yu-Chiang Frank Wang. 2017. Learning deep latent space for multi-label classification. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 2838–2844.

Min-Ling Zhang and Zhi-Hua Zhou. 2006. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1338–1351.

Arkaitz Zubiaga, Elena Kochkina, Maria Liakata, Rob Procter, and Michal Lukasik. 2016. Stance classification in rumours as a sequential task exploiting the tree structure of social media conversations. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2438–2448, Osaka, Japan. The COLING 2016 Organizing Committee.

# Investigating Sports Commentator Bias within a Large Corpus of American Football Broadcasts

**Jack Merullo**[★♠] **Luke Yeh**[★♠] **Abram Handler**[♠]
**Alvin Grissom II**[♣] **Brendan O'Connor**[♠] **Mohit Iyyer**[♠]

University of Massachusetts Amherst[♠] Ursinus College[♣]
{jmerullo,lyeh,ahandler,miyyer,brenocon}@umass.edu
agrissom@ursinus.edu

## Abstract

Sports broadcasters inject drama into play-by-play commentary by building team and player narratives through subjective analyses and anecdotes. Prior studies based on small datasets and manual coding show that such theatrics evince commentator bias in sports broadcasts. To examine this phenomenon, we assemble FOOTBALL, which contains 1,455 broadcast transcripts from American football games across six decades that are automatically annotated with 250K player mentions and linked with racial metadata. We identify major confounding factors for researchers examining racial bias in FOOTBALL, and perform a computational analysis that supports conclusions from prior social science studies.

## 1 Introduction

Sports broadcasts are major events in contemporary popular culture: televised American football (henceforth "football") games regularly draw tens of millions of viewers (Palotta, 2019). Such broadcasts feature live sports commentators who weave the game's mechanical details into a broader, more subjective narrative. Previous work suggests that this form of storytelling exhibits racial bias: nonwhite players are less frequently praised for good plays (Rainville and McCormick, 1977), while white players are more often credited with "intelligence" (Bruce, 2004; Billings, 2004). However, such prior scholarship forms conclusions from small datasets[1] and subjective manual coding of race-specific language.

We revisit this prior work using large-scale computational analysis. From YouTube, we collect broadcast football transcripts and identify mentions of players, which we link to metadata about each player's race and position. Our resulting FOOTBALL dataset contains over 1,400 games spanning six decades, automatically annotated with ∼250K player mentions (Table 1). Analysis of FOOTBALL identifies two confounding factors for research on racial bias: (1) the racial composition of many positions is very skewed (e.g., only ∼5% of running backs are white), and (2) many mentions of players describe only their actions on the field (not player attributes). We experiment with an additive log-linear model for teasing apart these confounds. We also confirm prior social science studies on racial bias in naming patterns and sentiment. Finally, we publicly release FOOTBALL,[2] the first large-scale sports commentary corpus annotated with player race, to spur further research into characterizing racial bias in mass media.

| Player | Race | Mention text |
|---|---|---|
| Baker Mayfield | white | "Mayfield the ultimate competitor he's tough he's scrappy" |
| Jesse James | white | "this is a guy ...does nothing but work brings his lunch pail" |
| Manny Lawson | nonwhite | "good specs for that defensive end freakish athletic ability" |
| B.J. Daniels | nonwhite | "that otherworldly athleticism he has saw it with Michael Vick" |

Table 1: Example mentions from FOOTBALL that highlight racial bias in commentator sentiment patterns.

## 2 Collecting the FOOTBALL dataset

We collect transcripts of 1,455 full game broadcasts from the U.S. NFL and National Collegiate Athletic Association (NCAA) recorded between 1960 and 2019. Next, we identify and link mentions of players within these transcripts to infor-

★Authors contributed equally.

[1] Rainville and McCormick (1977), for example, study only 16 games.

[2] http://github.com/jmerullo/football



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6355–6361, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

mation about their race (white or nonwhite) and position (e.g., quarterback). In total, FOOTBALL contains 267,778 mentions of 4,668 unique players, 65.7% of whom are nonwhite.[3] We now describe each stage of our data collection process.

### 2.1 Processing broadcast transcripts

We collect broadcast transcripts by downloading YouTube videos posted by nonprofessional, individual users identified by querying YouTube for football archival channels.[4] YouTube automatically captions many videos, allowing us to scrape caption transcripts from 601 NFL games and 854 NCAA games. We next identify the teams playing and game's year by searching for exact string matches in the video title and manually labeling any videos with underspecified titles.

After downloading videos, we tokenize transcripts using spaCy.[5] As part-of-speech tags predicted by spaCy are unreliable on our transcript text, we tag FOOTBALL using the ARK TweetNLP POS tagger (Owoputi et al., 2013), which is more robust to noisy and fragmented text, including TV subtitles (Jørgensen et al., 2016). Additionally, we use `phrasemachine` (Handler et al., 2016) to identify all corpus noun phrases. Finally, we identify player mentions in the transcript text using exact string matches of first, last, and full names to roster information from online archives; these rosters also contain the player's position.[6] Although we initially had concerns about the reliability of transcriptions of player names, we noticed minimal errors on more common names. Qualitatively, we noticed that even uncommon names were often correctly transcribed and capitalized. We leave a more systematic study for future work.

### 2.2 Identifying player race

Racial identity in the United States is a creation of complex, fluid social and historical processes (Omi and Winant, 2014), rather than a reflection of innate differences between fixed groups. Nevertheless, popular *perceptions* of race in the United States and the prior scholarship on racial bias in sports broadcasts which informs our work (Rainville and McCormick, 1977; Rada, 1996; Billings, 2004; Rada and Wulfemeyer, 2005) typically assume hard distinctions between racial groups, which measurably affect commentary. In this work, we do not reify these racial categories; we use them as commonly understood within the context of the society in which they arise.

To conduct a large-scale re-examination of this prior work, we must identify whether each player in FOOTBALL is perceived as white or nonwhite.[7] Unfortunately, publicly available rosters or player pages do not contain this information, so we resort to crowdsourcing. We present crowd workers on the Figure Eight platform with 2,720 images of professional player headshots from the Associated Press paired with player names. We ask them to "read the player's name and examine their photo" to judge whether the player is white or nonwhite. We collect five judgements per player from crowd workers in the US, whose high inter-annotator agreement (all five workers agree on the race for 93% of players) suggests that their perceptions are very consistent. Because headshots were only available for a subset of players, the authors labeled the race of an additional 1,948 players by performing a Google Image search for the player's name[8] and manually examining the resulting images. Players whose race could not be determined from the search results were excluded from the dataset.

## 3 Analyzing FOOTBALL

We now demonstrate confounds in the data and revisit several established results from racial bias studies in sports broadcasting. For all experiments, we seek to analyze the statistics of contextual terms that describe or have an important association with a mentioned player. Thus, we preprocess the transcripts by collecting contextual terms in windows of five tokens around each player mention, following the approach of Ananya et al. (2019) for gendered mention analysis.[9]

We emphasize that different term extraction strategies are possible, corresponding to different

[3] See Appendix for more detailed statistics.

[4] We specifically query for `full NFL|NCAA|college football games 1960s|1970s|1980s|1990s|2000`, and the full list of channels is listed in in the Appendix.

[5] https://spacy.io/ (2.1.3), Honnibal and Montani (2017)

[6] Roster sources listed in Appendix. We tag first and last name mentions only if they can be disambiguated to a single player in the rosters from opposing teams.

[7] While we use the general term "nonwhite" in this paper, the majority of nonwhite football players are black: in 2013, 67.3% of the NFL was black and most of the remaining players (31%) were white (Lapchick, 2014).

[8] We appended "NFL" to every query to improve precision of results.

[9] If multiple player mentions fall within the same window, we exclude each term to avoid ambiguity.

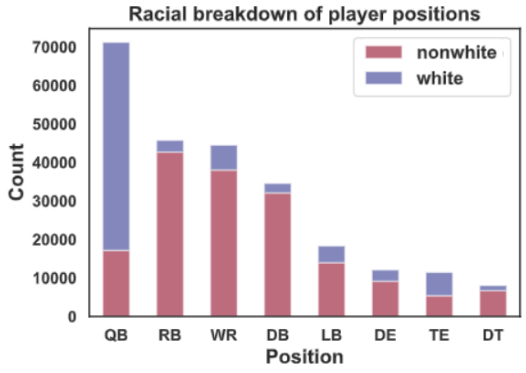


Figure 1: Almost all of the eight most frequently-mentioned positions in FOOTBALL are heavily skewed in favor of one race.

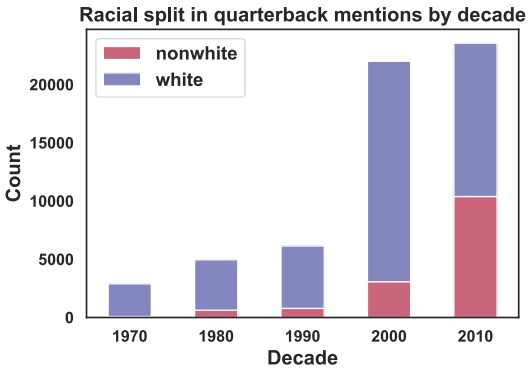


Figure 2: The percentage of nonwhite quarterbacks mentions has drastically increased over time, exemplifying the changing racial landscape in FOOTBALL across time.

precision–recall tradeoffs. For instance, instead of collecting all terms in a window (high recall) we might instead only collect terms in copular constructions with the entity mention (high precision), such as 'devoted' in "Tebow is devoted". Because mention detection strategies affect conclusions about bias in FOOTBALL, systematically defining, analyzing or even learning different possible strategies offers an exciting avenue for future work.

### 3.1 Statistical and linguistic confounds

Identifying racial bias in football broadcasts presents both statistical and linguistic modeling challenges. Many descriptions of players in broadcasts describe temporary player states (e.g., "Smith deep in the backfield") or discrete player actions ("Ogden with a huge block"), rather than possibly-biased descriptions of players themselves ("Cooper is one scrappy receiver"). Moreover, many players' actions ("passes the ball downfield") depend on the position they play, which is often skewed by race (Figure 1). Furthermore, the racial composition of mentions across different decades can differ dramatically—Figure 2 shows these changes for quarterback mentions—which makes the problem even more complex. Modeling biased descriptions of players thus requires disentangling attributes describing shifting, position-dependent player actions on field (e.g., "Paulsen the tight end with a *fast* catch") from attributes referring to intrinsic characteristics of individual players ("Paulsen is just so, so *fast*").

To demonstrate this challenge, we distinguish between per-position effects and racial effects using an additive, log-linear model which represents the log probability that a word or noun phrase $w$ will describe a player entity $e$ as the sum of two learned coefficients, corresponding to two observed covariates. One observed covariate records a player's race and the other a player's position, which allows us to use learned coefficients to represent how much a player's race or position contributes to the chance of observing an $(w, e)$ pair.

Formally, we model such effects using a sparse MAP estimation variant of SAGE (Eisenstein et al., 2011). We define the binary vector $y_e \in \{0, 1\}^J$ to represent the observed player covariates of race (white or nonwhite) and position. For example, component $y_{e,k}$ will be set to 1 if player $e$ is a quarterback and the component $k$ indexes the quarterback covariate; $y_e$ is a concatenation of two one-hot vectors. We then model $p(w \mid e) \propto \exp\left(\beta_w + (\gamma y_e)_w\right)$, with $\beta_w \in \mathbb{R}^{|\mathcal{V}|}$ as a background distribution over the vocabulary $\mathcal{V}$, set to empirical corpus-wide word and phrase log-probabilities, and $\gamma \in \mathbb{R}^{J \times |\mathcal{V}|}$ as a matrix of feature effects on those log probabilities. $\gamma_{j,w}$ denotes the *difference* in log-probability of $w$ for the $j^{th}$ player feature being on versus off. For example, if $j$ indexes the quarterback covariate and $w$ indexes the word "tough", then $\gamma_{j,w}$ represents how much more likely the word "tough" is to be applied to quarterbacks over the base distribution. We impose a uniform Laplace prior on all elements of $\gamma$ to induce sparsity, and learn a MAP estimate with the LibLBFGS implementation of OWL-QN, an L1-capable quasi-Newtonian convex optimizer (Andrew and Gao, 2007; Okazaki, 2010).

Table 2 shows several highest-valued $\gamma_{j,w}$ for a subset of the $J$ covariates. The adjective "quick"

| | |
|---|---|
| **White** | *long way, long time, valuable* |
| **DB** | *strong safety, free safety, state university* |
| **RB** | *second effort,single setback,ground game* |
| **QB** | *freshman quarterback, arm strength, easier* |
| **WR** | *quick slant, end zone touchdown, punt returner* |

Table 2: Top terms for the white, defensive back (DB), running back (RB), quarterback (QB), and wide receiver (WR) covariates for the log linear model.

is predictive of wide receivers, but refers to an action the players take on the field ("quick slant"), not an attribute of the receivers themselves. We also find that since "strong safety" is a kind of defensive back, the adjective "strong" is often associated with defensive backs, who are often nonwhite. In this case, "strong" does not reflect racial bias. Preliminary experiments with per-position mention-level race classifiers, as per Ananya et al. (2019), were also unable to disentangle race and position.

These results suggest that a more sophisticated approach may be necessary to isolate race effects from the confounds; it also raises sharp conceptual questions about the meaning of race-conditional statistical effects in social scientific inquiry, since race is a multifaceted construct (a "bundle of sticks," as Sen and Wasow (2016) argue). For future work, it may be useful to think of comparisons between otherwise similar players: how do broadcasters differ in their discussions of two players who are both quarterbacks, and who have similar in-game performance, but differ by race?

We now describe two experiments that sidestep some of these confounds, each motivated by prior work in social science: the first examines player naming patterns, which are less tied to action on field than player attributes. The other uses words with known sentiment polarity to identify positive and negative attributes, regardless of player position or game mechanics.

### 3.2 Exploring naming patterns

*Naming patterns* in sports broadcasting—how commentators refer to players by name (e.g., first or last name)—are influenced by player attributes, as shown by prior small-scale studies. For example, Koivula (1999) find that women are more frequently referred to by their first names than men in a variety of sports. Bruce (2004) discover a similar trend for race in basketball games: white players are more frequently referred to by their last names than nonwhite players, often because

| Position | Race | First | Last | Full |
|---|---|---|---|---|
| QB | white | 8.3% | 20.0% | 71.7% |
| QB | nonwhite | 18.1% | 7.5% | 74.5% |
| WR | white | 6.9% | 36.5% | 56.5% |
| WR | nonwhite | 11.3% | 24.1% | 64.6% |
| RB | white | 10.5% | 41.2% | 48.4% |
| RB | nonwhite | 8.5% | 35.4% | 56.1% |
| TE | white | 16.6% | 18.7% | 64.7% |
| TE | nonwhite | 13.8% | 16.6% | 69.7% |

Table 3: White players at the four major offensive positions are referred to by last name more often than nonwhite players at the same positions, a discrepancy that may reflect unconscious racial boundary-marking.

commentators believe their first names sound too "normal". Bruce (2004) further points out that the "practice of having fun or playing with the names of people from non-dominant racial groups" contributes to racial "othering". A per-position analysis of player mentions in FOOTBALL corroborates these findings for all offensive positions (Table 3).

### 3.3 Sentiment patterns

Prior studies examine the relationship between commentator sentiment and player race: Rainville and McCormick (1977) conclude that white players receive more positive coverage than black players, and Rada (1996) shows that nonwhite players are praised more for physical attributes and less for cognitive attributes than white ones.

To examine sentiment patterns within FOOTBALL, we assign a binary sentiment label to contextualized terms (i.e., a window of words around a player mention) by searching for words that match those in domain-specific sentiment lexicons from Hamilton et al. (2016).[10] This method identifies 49,787 windows containing sentiment-laden words, only 12.8% of which are of negative polarity, similar to the 8.3% figure reported by Rada (1996).[11] We compute a list of the most positive words for each race ranked by ratio of relative frequencies (Monroe et al., 2008).[12] A qualitative inspection of these lists

[10] We use a filtered intersection of lexicons from the NFL, CFB, and sports subreddits, yielding 121 positive and 125 negative words.

[11] Preliminary experiments with a state-of-the-art sentiment model trained on the Stanford Sentiment Treebank (Peters et al., 2018) produced qualitatively unreliable predictions due to the noise in FOOTBALL.

[12] We follow Monroe et al. (2008) in removing infrequent words before ranking; specifically, a word must occur at least ten times for each race to be considered.

| Race | Most positive words |
|---|---|
| white (all) | *enjoying, favorite, calm, appreciate, loving, miracle, spectacular, perfect, cool, smart* |
| nonwhite (all) | *speed, gift, versatile, gifted, playmaker, natural, monster, wow, beast, athletic* |
| white (QBs) | *cool, smart, favorite, safe, spectacular, excellent, class, fantastic, good, interesting* |
| nonwhite (QBs) | *ability, athletic, brilliant, awareness, quiet, highest, speed, wow, excited, wonderful* |

Table 4: Positive comments for nonwhite players (top two rows: all player mentions; bottom two rows: only quarterback mentions) focus on their athleticism, while white players are praised for personality and intelligence.

(Table 4) confirms that nonwhite players are much more frequently praised for physical ability than white players, who are praised for personality and intelligence (see Table 1 for more examples).

**Limitations:** The small lexicon results in the detection of relatively few sentiment-laden windows; furthermore, some of those are false positives (e.g., "beast mode" is the nickname of former NFL running back Marshawn Lynch). The former issue precludes a per-position analysis for all non-QB positions, as we are unable to detect enough sentiment terms to draw meaningful conclusions. The top two rows of Table 4, which were derived from all mentions regardless of position, are thus tainted by the positional confound discussed in Section 3.1. The bottom two rows of Table 4 are derived from the same analysis applied to just quarterback windows; qualitatively, the results appear similar to those in the top two rows. That said, we hope that future work on contextualized term extraction and sentiment detection in noisy domains can shed more light on the relationship between race and commentator sentiment patterns.

## 4 Related Work

Our work revisits specific findings from social science (§3) on racial bias in sports broadcasts. Such non-computational studies typically examine a small number of games drawn from a single season and rely on manual coding to identify differences in announcer speech (Rainville and McCormick, 1977; Billings, 2004; Rada and Wulfemeyer, 2005). For example, Rada (1996) perform a fine-grained analysis of five games from the 1992 season, coding for aspects such as players' cognitive or physical attributes. Our computational approach allows us to revisit this type of work (§3) using FOOTBALL, without relying on subjective human coding.

Within NLP, researchers have studied gender bias in word embeddings (Bolukbasi et al., 2016; Caliskan et al., 2017), racial bias in police stops (Voigt et al., 2017) and on Twitter (Hasanuzzaman et al., 2017), and biases in NLP tools like sentiment analysis systems (Kiritchenko and Mohammad, 2018). Especially related to our work is that of Ananya et al. (2019), who analyze mention-level gender bias, and Fu et al. (2019), who examine gender bias in tennis broadcasts. Other datasets in the sports domain include the event-annotated baseball commentaries of Keshet et al. (2011) and the WNBA and NBA basketball commentaries of Aull and Brown (2013), but we emphasize that FOOTBALL is the first large-scale sports commentary corpus annotated for race.

## 5 Conclusion

We collect and release FOOTBALL to support large-scale, longitudinal analysis of racial bias in sports commentary, a major category of mass media. Our analysis confirms the results of prior smaller-scale social science studies on commentator sentiment and naming patterns. However, we find that baseline NLP methods for quantifying mention-level genderedness (Ananya et al., 2019) and modeling covariate effects (Eisenstein et al., 2011) cannot overcome the statistical and linguistic confounds in this dataset. We hope that presenting such a technically-challenging resource, along with an analysis showing the limitations of current bias-detection techniques, will contribute to the emerging literature on bias in language. Important future directions include examining the temporal aspect of bias as well as developing more precise mention identification techniques.

## Acknowledgments

We thank the anonymous reviewers for their insightful comments, and Emma Strubell, Patrick Verga, David Fisher, Katie Keith, Su Lin Blodgett, and other members of the UMass NLP group for help with earlier drafts of the paper. This work was partially supported by NSF IIS-1814955.

## References


Ananya Ananya, Nitya Parthasarthi, and Sameer Singh. 2019. Genderquant: Quantifying mention-level genderedness. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

G. Andrew and J. Gao. 2007. Scalable training of L1-regularized log-linear models. In *Proceedings of the International Conference of Machine Learning*.

Laura L Aull and David West Brown. 2013. Fighting words: a corpus analysis of gender representations in sports reportage. *Corpora*, 8(1).

Andrew C Billings. 2004. Depicting the quarterback in black and white: A content analysis of college and professional football broadcast commentary. *Howard Journal of Communications*, 15(4).

Tolga Bolukbasi, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and Adam Tauman Kalai. 2016. Man is to computer programmer as woman is to homemaker? Debiasing word embeddings. In *Proceedings of Advances in Neural Information Processing Systems*.

Toni Bruce. 2004. Marking the boundaries of the 'normal'in televised sports: The play-by-play of race. *Media, Culture & Society*, 26(6).

Aylin Caliskan, Joanna J. Bryson, and Arvind Narayanan. 2017. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334).

Jacob Eisenstein, Amr Ahmed, and Eric P. Xing. 2011. Sparse additive generative models of text. In *Proceedings of the International Conference of Machine Learning*.

Liye Fu, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2019. Tie-breaker: Using language models to quantify gender bias in sports journalism. In *International Joint Conference on Artificial Intelligence*.

William L Hamilton, Kevin Clark, Jure Leskovec, and Dan Jurafsky. 2016. Inducing domain-specific sentiment lexicons from unlabeled corpora. In *Proceedings of Empirical Methods in Natural Language Processing*.

Abram Handler, Matthew Denny, Hanna M. Wallach, and Brendan T. O'Connor. 2016. Bag of what? simple noun phrase extraction for text analysis. In *NLP+CSS@EMNLP*.

Mohammed Hasanuzzaman, Gaël Dias, and Andy Way. 2017. Demographic word embeddings for racism detection on twitter. In *International Joint Conference on Natural Language Processing*.

Matthew Honnibal and Ines Montani. 2017. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing.

Anna Jørgensen, Dirk Hovy, and Anders Søgaard. 2016. Learning a POS tagger for AAVE-like language. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Ezra Keshet, Terrence Szymanski, and Stephen Tyndall. 2011. BALLGAME: A corpus for computational semantics. In *Proceedings of the Ninth International Conference on Computational Semantics (IWCS 2011)*.

Svetlana Kiritchenko and Saif Mohammad. 2018. Examining gender and race bias in two hundred sentiment analysis systems. In *Proceedings of the Joint Conference on Lexical and Computational Semantics*.

Nathalie Koivula. 1999. Gender stereotyping in televised media sport coverage. *Sex roles*, 41(7-8).

Richard Lapchick. 2014. The 2014 racial and gender report card: National football league.

Burt L Monroe, Michael P Colaresi, and Kevin M Quinn. 2008. Fightin'words: Lexical feature selection and evaluation for identifying the content of political conflict. *Political Analysis*, 16(4).

Naoaki Okazaki. LibLBFGS: a Library of Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) [online]. 2010.

Michael Omi and Howard Winant. 2014. *Racial formation in the United States*. Routledge.

Olutobi Owoputi, Brendan T. O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Frank Palotta. 2019. NFL ratings rebound after two seasons of declining viewership. *CNN Business*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

James A Rada. 1996. Color blind-sided: Racial bias in network television's coverage of professional football games. *Howard Journal of Communications*, 7(3).

James A Rada and K Tim Wulfemeyer. 2005. Color coded: Racial descriptors in television coverage of intercollegiate sports. *Journal of Broadcasting & Electronic Media*, 49(1).

Raymond E Rainville and Edward McCormick. 1977. Extent of covert racial prejudice in pro football announcers' speech. *Journalism Quarterly*, 54(1).

Maya Sen and Omar Wasow. 2016. Race as a 'bundle of sticks': Designs that estimate effects of seemingly immutable characteristics. *Annual Review of Political Science*, 19:499–522.

Rob Voigt, Nicholas P. Camp, Vinodkumar Prabhakaran, William L. Hamilton, Rebecca C. Hetey, Camilla M. Griffiths, David Jurgens, Dan Jurafsky, and Jennifer L. Eberhardt. 2017. Language from police body camera footage shows racial disparities in officer respect. *Proceedings of the National Academy of Sciences*, 114(25).

# Charge-Based Prison Term Prediction with Deep Gating Network

**Huajie Chen**[1*] **Deng Cai**[2*] **Wei Dai**[1] **Zehui Dai**[1] **Yadong Ding**[1]
[1]NLP Group, Gridsum, Beijing, China
{chenhuajie,daiwei,daizehui,dingyadong}@gridsum.com
[2]The Chinese University of Hong Kong
thisisjcykcd@gmail.com

## Abstract

Judgment prediction for legal cases has attracted much research efforts for its practice use, of which the ultimate goal is prison term prediction. While existing work merely predicts the total prison term, in reality a defendant is often charged with multiple crimes. In this paper, we argue that charge-based prison term prediction (CPTP) not only better fits realistic needs, but also makes the total prison term prediction more accurate and interpretable. We collect the first large-scale structured data for CPTP and evaluate several competitive baselines. Based on the observation that fine-grained feature selection is the key to achieving good performance, we propose the Deep Gating Network (DGN) for charge-specific feature selection and aggregation. Experiments show that DGN achieves the state-of-the-art performance.

## 1 Introduction

Judgment prediction (Kort, 1957; Ulmer, 1963; Segal, 1984; Liu et al., 2004; Liu and Hsieh, 2006) aims at automatically predicting the judgment result given a textual description of a legal case (An example is given in Figure 1). Recently, there has been a resurgent interest in this task due to the availability of more data and new machine learning techniques (Luo et al., 2017; Zhong et al., 2018b; Hu et al., 2018).

Judgment prediction can be decomposed into several sub-tasks: (a) relevant law article extraction (Liu and Hsieh, 2006; Liu and Liao, 2005; Liu et al., 2015), (b) charge prediction (Liu and Hsieh, 2006; Luo et al., 2017; Hu et al., 2018), (c) and prison term prediction (Zhong et al., 2018a). The dependencies among them have also been studied by Zhong et al. (2018b). While effective methods exist for sub-task (a) and (b), (e.g In CAIL2018 competition (Zhong et al., 2018a), both the charge prediction and the article prediction have attained $F_{\text{micro}}$ over $95\%$), the prison term prediction remains the performance bottleneck.

**Case description**: On July 7, 2017, when the defendant Cui XX was drinking in a bar, he came into conflict with Zhang XX…… After arriving at the police station, he refused to cooperate with the policeman and bited on the arm of the policeman……

**Result of judgment**: Cui XX was sentenced to *12* months imprisonment for *creating disturbances* and *12* months imprisonment for *obstructing public affairs*……

| | | |
|---|---|---|
| ● Charge#1 | creating disturbances | term 12 months |
| ● Charge#2 | obstructing public affairs | term 12 months |

Figure 1: An example of judgment prediction.

In this paper, we improve the accuracy of prison term prediction by decomposing it into a set of *charge-based* prison term predictions (CPTPs). In this way, more subtle and sophisticated interactions between textual description and a specific charge can be captured, resulting in more precise term predictions for individual charges. Meanwhile, CPTPs also shed light on the prediction of the total prison term.

On the other hand, CPTP also poses challenges due to the following reasons: The case description can be very lengthy and not all parts are relevant to a specific charge. The charge-related descriptions are often presented in an interleaving way, making it difficult to associate a specific charge with its corresponding information.

To address the above problems, we propose the Deep Gating Network (DGN) for gradually filtering and aggregating charge-specified information at different levels of granularity. Specifically, we stack multiple blocks of an LSTM layer and a charge-specific gating layer for generating a focused charge-based representation of the case description. Finally, the whole document representa-

*Both authors contributed equally.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6362–6367, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

tion is obtained by a convolutional neural network.

To conduct the experiments, we construct a new dataset, which contains more than 200, 000 criminal cases.[1] To show the effectiveness of the proposed approach, we compare it with several strong baselines adapted from aspect-based sentiment classification (Wang et al., 2016; Tang et al., 2016; Chen et al., 2017; Li et al., 2018). Experiments show that our method achieves significantly better results than all of them. In addition, when we leverage the results of charge-based term predictions for the total prison term prediction, it also surpasses several strong baselines that are directly aimed at the total term prison prediction.

In summary, our contributions are as follows:

- We formally define the task of charge-based prison term prediction and collect the first dataset for it.
- We propose the Deep Gating Network (DGN). Experiments show our method achieves the state-of-the-art performance.
- We show that the accuracy of the total term prediction is also improved by a simple heuristic integration of individual charge-based term predictions.

## 2 Problem Definition & Dataset Construction

We formally define the task of charge-based prison term prediction as follows. The input are a case description $\mathbf{x} = \{x_1, x_2, \cdots, x_n\}$ and a set of corresponding charges $\mathbf{c} = \{c_1, c_2, \cdots, c_k\}$, where $n$ and $k$ are the length of case description and the number of charges respectively. The goal is to predict the prison terms $\mathbf{y} = \{y_1, y_2, \cdots, y_k\}$, where $y_j$ is the prison term corresponding to charge $c_j$.

To the best of our knowledge, there is no existing structured dataset for the above task. We thus collect and construct a dataset based on the published records from the Supreme People's Court of China,[2] where each criminal case document includes the accusation by the procuratorate, the court view, and the result of judgment. Following Xiao et al. (2018), we take the accusation by the procuratorate as the input textual description. The charges and the corresponding prison terms are extracted from the result of judgment using regular expressions like "sentence to __ months imprisonment for __". We build 238,749 well-structured cases in total (An example is given in Fig 1). The collected cases are further split into the training set, the validation set, and the test set. The statistics of the dataset are detailed in Table 1. The range of possible prison terms is [1, 240] (in months). The dataset has a broad coverage of common charges, 157 different types of charges are involved.

| | #single | #multiple | total |
|---|---|---|---|
| Train | 147, 580 | 42, 420 | 190, 000 |
| Valid | 19, 350 | 5, 602 | 24, 952 |
| Test | 18, 539 | 5, 258 | 23, 797 |

Table 1: The statistics of the proposed dataset. #single/#multiple means single/multiple charge(s) cases.

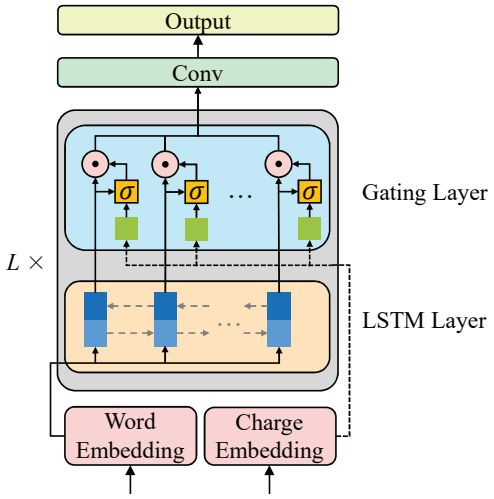


Figure 2: The architecture of DGN

## 3 Our Approach

Figure 2 gives an overview of our model, which consists of two components: (1) Deep Gating Network (DGN) for charge-based feature filtering and aggregation. (2) Convolutional Neural Network (CNN) for the whole document representation learning.

### 3.1 Deep Gating Network

At the bottom layer of DGN, each word $x_i$ is mapped into a low dimensional vector $h_i^{(0)}$ according to a word embedding table.

DGN then starts to construct charge-specific representations gradually. $L$ identical blocks are

[1] The dataset can be found at https://github.com/huajiechen/CPTP

[2] http://wenshu.court.gov.cn/

hierarchically stacked. The $l$-th block takes the output of the $(l-1)$-th layer $h_i^{l-1}$ as input. Each block transforms its input semantic vectors into more sophisticated and focused representations based on gated feature selection and combination.

Specifically, each gating block consists of a bi-LSTM layer for context aggregation and a gating layer for charge-specific feature filtering.

$$\tilde{h}_i^{(l)} = [\overrightarrow{\text{LSTM}}(h_i^{(l-1)}); \overleftarrow{\text{LSTM}}(h_i^{(l-1)})]$$
$$h_k^{(l)} = g_i^{(l)} \odot \tilde{h}_i^{(l)}$$

where $g_i^{(l)}$ is the gate for $i$-th vector of the $l$-th gating block and $\odot$ denotes element-wise multiplication. $g_i^{(l)}$ is computed as:

$$g_i^{(l)} = \text{sigmoid}(W^{(l)}[\tilde{h}_i^{(l)}; c_j] + b^{(l)})$$

where $c_j$ is target charge embedding. The gating layers can select the charge-specific features according to the target charge embedding.

### 3.2 Convolutional Neural Network

Convolutional Neural Network (CNN) has been effective in modeling sequential data (Kim, 2014; Hu et al., 2014; Pang et al., 2016). It uses convolution operations (with multiple groups of filters) for $n$-gram feature extraction. The sequence-level representation is then obtained through max-pooling, where the most salient $n$-gram features are detected and selected.

In this work, we use a CNN with filter width in $[1, 2, 3, 4, 5]$. The number of filters for each width is 256. We concatenate the outputs of different filters for the final document representation $z$.

### 3.3 Output and Training

The charge-specific document representation $z$ is passed to a fully connected layer with ReLU activation for the final prediction.

$$\hat{y}_j = \text{ReLU}(W_o z + b_o) \quad (1)$$

where $W_o$ and $b_o$ are trainable parameters.

Since the Mean Squared Error (MSE) loss cannot reflect the relative deviation ratio between the prediction and the ground-truth, we take the logarithm before estimating their difference.

$$\delta_j = |\log(y_j + 1) - \log(\hat{y}_j + 1)| \quad (2)$$

To alleviate the impact of outliers and stabilize the training, we propose to use Huber Loss (Huber, 1964), $a$ is set to 1 in experiments:

$$l(x, y_j) = \begin{cases} 0.5 \times \delta_j^2, & \text{if } \delta_j < a \\ a(\delta_j - 0.5a), & \text{otherwise} \end{cases}$$
$$\mathcal{L}(x, y) = \sum_{j=1}^{k} l(x, y_j)$$

**Total Term Prediction** Although our model is trained to predict the prison term for specific charge, it can be readily adapted to predict the total term by a simple heuristic integration of individual charge-based prison term predictions. There are certain regulations for combined punishment of crimes in Chinese legislation. For simplicity, we take the average of the maximum and summation of individual charge-specific term predictions. The total term prediction is also capped at 240 months.

$$\hat{y}_{total} = \min(240, \frac{\max_j(\hat{y}_j) + \sum_j(\hat{y}_j)}{2})$$

## 4 Experiments

### 4.1 Evaluation Metrics

For evaluation, we adopt the official score function (S metric) of the CAIL2018 Competition (Zhong et al., 2018a). The score function measures the log different $\delta$ between prediction value $\hat{y}$ and gold value $y$ as in Eq 2. The final score $s(\delta)$ is a piecewise function that increases monotonically with the value of $\delta$. For more details about the S metric, we refer interested readers to (Zhong et al., 2018a). We also report the exact match (EM) rate and error-tolerant accuracy Acc@$p$, where $p$ is the maximum acceptable error rate. Formally, a prediction is considered "correct" if and only if its value is in the range $[y(1-p), y(1+p)]$.

### 4.2 Compared Methods

The task of charged-based prison term prediction is similar in spirit to aspect-based sentiment classification (Pang et al., 2008), where multiple classification decisions are made given one text description and different target entities. This suggests that other neural architectures proposed for aspect-based sentiment classification may also be suitable for our task. The adaption from classification to regression can be easily accomplished by replacing the original final layer with that of Eq 1. Specifically, we adapted the following models:

| Model | S | EM | Acc@0.1 | Acc@0.2 |
|---|---|---|---|---|
| ATE-LSTM | 66.49 | 7.72 | 16.12 | 33.89 |
| MemNet | 70.23 | 7.52 | 18.54 | 36.75 |
| RAM | 70.32 | 7.97 | 18.87 | 37.38 |
| TNet | 73.94 | 8.06 | 19.55 | 39.89 |
| DGN | **76.48** | **8.92** | **20.66** | **42.61** |

Table 2: Results on charge-based prison term prediction(%).

- ATAE-LSTM (Wang et al., 2016): it concatenates aspect embedding and the output of LSTM, and uses self-attention to obtain aspect-based representation.
- MemNet (Tang et al., 2016): it uses multi-hop attention over the word embeddings for a sentence, where aspect embedding is regarded as the initial key.
- RAM (Chen et al., 2017): it also uses multi-hop attention for aspect-specific representation learning, while the attention at different time steps are aggregated by recurrent neural network.
- TNet (Li et al., 2018): it has a similar architecture to DGN. The major difference is that it employs a Transformation Network for mixing the information in aspect embedding and token representations rather than the explicit gates in our model.

The aspect embedding in above models is replaced by charge embedding in our experiments. In addition, we also compare with the popular models for total term prediction (Zhong et al., 2018a,b):

- CNN (Kim, 2014): the case description is encoded by a CNN with multiple filter widths, followed by max-pooling.
- RNN (Hochreiter and Schmidhuber, 1997): bi-LSTM are used for case description encoding, where the final states are regarded as the document representation.
- RCNN (Lai et al., 2015): we stack a CNN on the top of LSTM states for final representation.

### 4.3 Main Results

The results of charge-based prison term prediction are shown in Table 2. The proposed DGN achieves the best results on all four metrics. In addition, the margins between our model and others are remarkably wide. It can be observed that aspect-based sentiment models only give moderate performance, which we attribute to that the case description is so long that more rigorous feature selection, such as the treatment of DGN, is needed. Our model selects and aggregates features in a explicit way which is more efficient and effective in dealing with charge-specific descriptions often spread out across lengthy case documents in CPTP.

| Model | S | EM | Acc@0.1 | Acc@0.2 |
|---|---|---|---|---|
| CNN | 67.24 | 8.41 | 16.96 | 35.58 |
| RNN | 67.27 | 8.04 | 16.79 | 35.11 |
| RCNN | 69.56 | 8.54 | 17.57 | 35.75 |
| DGN | **75.74** | **8.64** | **19.32** | **40.43** |

Table 3: Results on total term prediction(%).

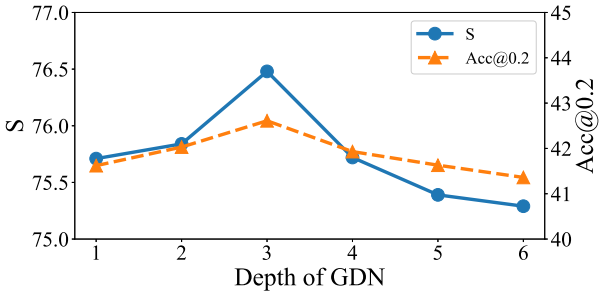


Figure 3: Performance with different depths of DGN.

Table 3 presents the results of the total term prediction. Although our method is not directly trained to make the final prediction, the performance of our model surpasses all baselines, which confirms that the breakdown charge-based analysis can indeed help the total prison term prediction.

### 4.4 Depth of DGN

To study the impact of the number of DGN blocks, we test our model with various depths and show the results in Fig 3.[3] As shown, the performance improves as the depth of DGN increases until it reaches 3 when the performance begins to drop likely due to overfitting.

### 4.5 Effects of Log Huber Loss

We compare Log Huber Loss (LHL) with Mean Square Error (MSE), Mean Absolute Error (MAE) and Huber Loss (HL). We also try Log Cosh Loss

[3]For simplicity, we only show S score and Acc@0.2.

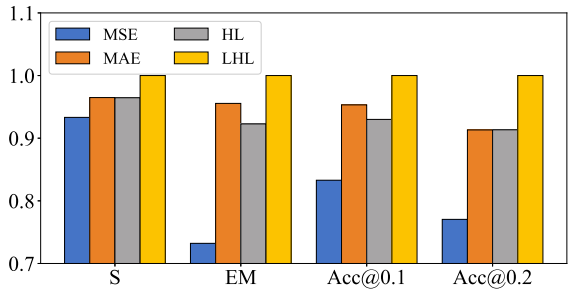


Figure 4: Compare of different loss functions. The result of LHL is regarded as unit 1.

(LCL), but it does not converge. As shown in Figure 4, Log Huber Loss performs best in all metrics for all models. The improvement is most significant in S metric. It also suggests that making the loss function consistent with the evaluation metric is beneficial.

### 4.6 Error Analysis

So far, our model has the best results on prison term predictions. In this section, we aim to conduct an in-depth analysis and answer the following questions: (1) In which cases, our model fails to deliver accurate predictions? (2) What are the prospects for further improvement? After carefully analyzing 100 examples, we roughly classify them into the following categories.

**Lengthy Description** Some cases are extremely complicated, especially for cases with gangs. These descriptions are often lengthy and involve multiple criminal suspects.

**Incomplete Information** In some cases, the input case description does not contain sufficient information for precise prediction. Note we only take the accusation by the procuratorate as input, which is incomplete compared to the whole materials relevant to a case. For example, if a defendant is recidivism within a shorter period, he/she shall be given a heavier punishment.

**Rare Cases** Some special circumstances will influence the prison term, yet rarely happen in the training set. For example, if a defendant cause injuries to others due to excessive defense, he/she shall be given a lighter punishment. This knowledge is easily understandable by humans, bu hard to be learned by machine learning models.

## 5 Ethical Discussions

Although the research on prison term prediction has considerable potential to improve efficiency and fairness in criminal justice, there are certain ethical concerns worth discussions.

First, does the training data provide unbiased examples and sufficient? For example, some may worry about that the model may treat people differently based on race, social class, age and so on (Tonry, 2014). Discrimination in the past may be learned in models. Also, with the development of our society, new forms of crimes will appear. A model trained on historical data may fail in these new cases.

Second, is the learned system robust enough? Some subtle details may significantly affect the result of judgment. For example, the amount of theft and the number of drugs, these numerical values are often not uniform in different case descriptions, causing it hard to learn by neural models. Some infrequent words, such as named entities, may also cause undesirable interference.

The mistake of legal judgment is serious, it is about people losing years of their lives in prison, or dangerous criminals being released to reoffend. We should pay attention to how to avoid judges' over-dependence on the system. It is necessary to consider its application scenarios. In practice, we recommend deploying our system in the "Review Phase", where other judges check the judgment result by a presiding judge. Our system can serve as one anonymous checker.

In summary, the judgment prediction is an emerging technology at its exploratory stage. We should be aware of the risks and prevent any inappropriate use of the technology.

## 6 Conclusion

In this paper, we formally presented the task of charge-based prison term prediction. We introduced the first large-scale dataset for this task. To tackle the problem of the noisy and entangled description of legal cases, we proposed the deep gating network for charge-specific information filter. Experiments show that our model significantly improves the accuracy of charge-based prison term prediction, as well as the total term prediction. Finally, we discussed some ethical problems of the proposed techniques that are worth cautious thinking.

## References


Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. 2017. Recurrent attention network on memory for aspect sentiment analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 452–461.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in neural information processing systems*, pages 2042–2050.

Zikun Hu, Xiang Li, Cunchao Tu, Zhiyuan Liu, and Maosong Sun. 2018. Few-shot charge prediction with discriminative legal attributes. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 487–498.

Peter J. Huber. 1964. Robust estimation of a location parameter. *Annals of Mathematical Statistics*, 35(1):73–101.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

Fred Kort. 1957. Predicting supreme court decisions mathematically: A quantitative analysis of the "right to counsel" cases. *American Political Science Review*, 51(1):1–12.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Twenty-ninth AAAI conference on artificial intelligence*.

Xin Li, Lidong Bing, Wai Lam, and Bei Shi. 2018. Transformation networks for target-oriented sentiment classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 946–956.

Chao-Lin Liu, Cheng-Tsung Chang, and Jim-How Ho. 2004. Case instance generation and refinement for case-based criminal summary judgments in chinese.

Chao-Lin Liu and Chwen-Dar Hsieh. 2006. Exploring phrase-based classification of judicial documents for criminal charges in chinese. In *International Symposium on Methodologies for Intelligent Systems*, pages 681–690. Springer.

Chao-Lin Liu and Ting-Ming Liao. 2005. Classifying criminal charges in chinese for web-based legal services. In *Asia-Pacific Web Conference*, pages 64–75. Springer.

Yi-Hung Liu, Yen-Liang Chen, and Wu-Liang Ho. 2015. Predicting associated statutes for legal problems. *Information Processing & Management*, 51(1):194–211.

Bingfeng Luo, Yansong Feng, Jianbo Xu, Xiang Zhang, and Dongyan Zhao. 2017. Learning to predict charges for criminal cases with legal basis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2727–2736.

Bo Pang, Lillian Lee, et al. 2008. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, 2(1–2):1–135.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016. Text matching as image recognition. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Jeffrey A Segal. 1984. Predicting supreme court cases probabilistically: The search and seizure cases, 1962-1981. *American Political Science Review*, 78(4):891–900.

Duyu Tang, Bing Qin, and Ting Liu. 2016. Aspect level sentiment classification with deep memory network. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 214–224.

Michael Tonry. 2014. Legal and ethical issues in the prediction of recidivism. *Federal Sentencing Reporter*, 26(3):167–176.

S Sidney Ulmer. 1963. Quantitative analysis of judicial processes: Some practical and theoretical applications. *Law & Contemp. Probs.*, 28:164.

Yequan Wang, Minlie Huang, xiaoyan zhu, and Li Zhao. 2016. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 606–615.

Chaojun Xiao, Haoxi Zhong, Zhipeng Guo, Cunchao Tu, Zhiyuan Liu, Maosong Sun, Yansong Feng, Xianpei Han, Zhen Hu, Heng Wang, and Jianfeng Xu. 2018. Cail2018: A large-scale legal dataset for judgment prediction.

Haoxi Zhong, Chaojun Xiao, Zhipeng Guo, Cunchao Tu, Zhiyuan Liu, Maosong Sun, Yansong Feng, Xianpei Han, Zhen Hu, Heng Wang, and Jianfeng Xu. 2018a. Overview of cail2018: Legal judgment prediction competition.

Haoxi Zhong, Guo Zhipeng, Cunchao Tu, Chaojun Xiao, Zhiyuan Liu, and Maosong Sun. 2018b. Legal judgment prediction via topological learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3540–3549. Association for Computational Linguistics.

# Restoring ancient text using deep learning: a case study on Greek epigraphy

**Yannis Assael**[1,2,†], **Thea Sommerschield**[1,†], **Jonathan Prag**[1]
University of Oxford[1], DeepMind[2]

## Abstract

Ancient History relies on disciplines such as Epigraphy, the study of ancient inscribed texts, for evidence of the recorded past. However, these texts, "inscriptions", are often damaged over the centuries, and illegible parts of the text must be restored by specialists, known as epigraphists. This work presents PYTHIA, the first ancient text restoration model that recovers missing characters from a damaged text input using deep neural networks. Its architecture is carefully designed to handle long-term context information, and deal efficiently with missing or corrupted character and word representations. To train it, we wrote a non-trivial pipeline to convert PHI, the largest digital corpus of ancient Greek inscriptions, to machine actionable text, which we call PHI-ML. On PHI-ML, PYTHIA's predictions achieve a 30.1% character error rate, compared to the 57.3% of human epigraphists. Moreover, in 73.5% of cases the ground-truth sequence was among the Top-20 hypotheses of PYTHIA, which effectively demonstrates the impact of this assistive method on the field of digital epigraphy, and sets the state-of-the-art in ancient text restoration.

## 1 Introduction

One of the key sources for Ancient History is the discipline of epigraphy, which delivers firsthand evidence for the thought, society and history of ancient civilisations. Epigraphy is the study of documents, "inscriptions", written on a durable surface (stone, ceramic, metal) by individuals, groups and institutions of the past (Davies and Wilkes, 2012). Only a small minority of surviving inscriptions are fully legible and complete, as many have been damaged in time (Figure 1). An epigraphist must then hypothesise how much text is missing, and what it might have originally been. These hypotheses are called "restorations" (Bodel, 2012). The present work offers a fully automated aid to the epigraphist's restoration task.

Restoring text is a complex and time-consuming task (Woodhead, 1967; Mattingly, 1996). Epigraphists rely on accessing vast repositories of information to find textual and contextual "parallels" (recurring expressions in similar documents). These repositories primarily consist in a researcher's mnemonic repertoire of such parallels, and in digital corpora for performing "string matching" searches (The Packard Humanities Institute, 2005; Clauss, 2012). However, minor differences in the search query can exclude or obfuscate relevant results, making it hard to estimate the true probability distribution of possible restorations. To the best of our knowledge, this is the first work to bypass the constraints of current epigraphic methods by means of a fully automated deep learning model, PYTHIA, which aids the task of ancient text restoration. It is supplemented by PHI-ML, an epigraphic dataset of a machine actionable text. PYTHIA takes as input a sequence of damaged text, and is

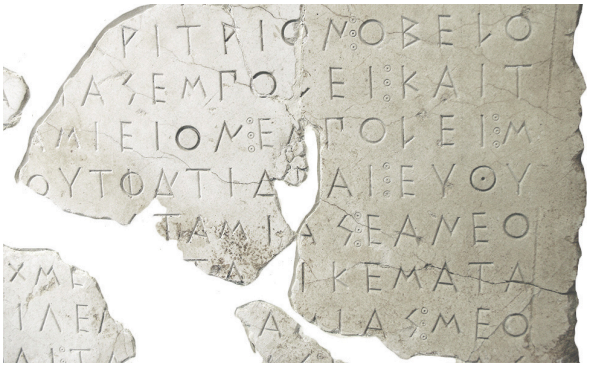

Figure 1: Damaged inscription: a decree concerning the Acropolis of Athens (485/4 BCE). *IG* $I^3$ 4B. (CC BY-SA 3.0, WikiMedia)

[†] These authors contributed equally to this work.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6368–6375, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

trained to predict character sequences comprising the hypothesised restorations. It works both at a character- and a word-level, thereby effectively handling incomplete or missing words. PYTHIA can furthermore be used by all disciplines dealing with ancient texts (philology, papyrology, codicology) and applies to any language (ancient or modern). To aid and encourage future research, PYTHIA and PHI-ML have been open-sourced at http://tiny.cc/ancient-text-restoration.

## 2 Related work

Natural language processing (NLP) has dealt with tasks akin to text restoration. Indeed, standard count-based n-gram language models (LM) share with epigraphists the "parallel-finding" approach. N-gram models are outperformed by neural language models, which operate at a word-level (Mikolov et al., 2010, 2011), at a subword- or character-level (Sutskever et al., 2011; Mikolov et al., 2012; Botha and Blunsom, 2014), or a combination of both, known as character-aware language models (Miyamoto and Cho, 2016; Kim et al., 2016; Hwang and Sung, 2017). Despite our efforts to include BERT (Devlin et al., 2018) in our evaluation, we found that the excessive resources required did not allow for training on a single GPU. Text restoration also shares similarities with machine reading comprehension (Hermann et al., 2015; Kočiskỳ et al., 2018), and cloze deletion tests (Hill et al., 2016; Bajgar et al., 2017; Fedus et al., 2018; Xie et al., 2018; Zhang et al., 2018). Although word-level language modelling is capable of capturing context information more efficiently than character-level alternatives, damaged inscriptions preserve only limited parts of words, complicating the learning of representations. To overcome this issue, PYTHIA works simultaneously at both a character- and a word-level, thereby capturing long-term dependencies ("context information").

Finally, several works have used machine learning to study ancient inscriptions, focusing on assistive tools (Roued-Cunliffe, 2010), optical character recognition and visual analysis (Terras and Robertson, 2006; Garz et al., 2014; Soumya and Kumar, 2014; Shaus, 2017; Hussien et al., 2015; Amato et al., 2016; Can et al., 2016; Suganya and Murugavalli, 2017; Palaniappan and Adhikari, 2017; Avadesh and Goyal, 2018), writer identification (Tracy and Papaodysseus, 2009; Panagopoulos et al., 2009; Faigenbaum-Golovin et al., 2016), text analysis (Rao et al., 2009b,a, 2010; Yadav et al., 2010; Lee and Haug, 2010; Vatri and McGillivray, 2018), and machine translation (Pagé-Perron et al., 2017; Luo et al., 2019).

## 3 Generating PHI-ML

Due to availability of digitised epigraphic corpora, PYTHIA has been trained on ancient Greek (henceforth, "AG") inscriptions, written in the ancient Greek language between $7^{\text{th}}$ century BCE and $5^{\text{th}}$ century CE. We chose AG epigraphy as a case study for two reasons: a) the variability of contents and context of the AG epigraphic record makes it an excellent challenge for NLP; b) several digital AG textual corpora have been recently created, the largest ones being PHI (The Packard Humanities Institute, 2005; Gawlinski, 2017) for epigraphy; Perseus (Smith et al., 2000) and First1KGreek (Crane et al., 2014) for ancient literary texts.

When restoring damaged AG inscriptions, the epigraphists' conjectures on the total number of missing characters are guided by grammatical and syntactical considerations, as well as by the reconstructed graphical layout of the inscription. Conjectured missing characters are conventionally marked with hyphens, one hyphen equating to one missing character. Additionally, epigraphists traditionally convert edited texts to lower case and add punctuation and diacritics, which are generally absent from the original inscription. These conventions were also used in PHI.

Because human annotations in PHI were noisy and often syntactically inconsistent (Iversen, 2007), we wrote a pipeline to convert it into a machine actionable text. We first computed the character frequencies and standardised the AG alphabet to include all core characters, including all accentuation (147 characters), numbers, spaces and punctuation marks. Two additional characters were introduced: '-' representing a missing character, and '?' signifying a character to be predicted. Then we wrote regular expressions to replace all AG numerical notations appearing in the texts with 0 to avoid numerical correlations, strip the remaining punctuation marks, remove the conventional epigraphical symbols surrounding certain characters ("Leiden Conventions"), and discard notes whose content was not in Greek. We then proceeded to clear human comments, fix the spacing and cases of duplicate punctuation,

and filtered the resulting text so as to retain only the restricted alphabetical characters. The texts with fewer than 100 characters were also discarded. Lastly, we matched the number of missing characters with those conjectured by epigraphists, thereby converting the length value to an equal number of '-' symbols.

The resulting dataset is named PHI-ML, and consists of more than 3.2 million words (Table 1). The inscriptions whose PHI IDs ended in {3, 4} (every inscription in PHI was assigned a unique identifier when the original corpus was created) were held out and used respectively as test and validation sets.

| Split | Inscriptions | Words | Chars |
|---|---|---|---|
| Train | 34,952 | 2,792k | 16,300k |
| Valid | 2,826 | 211k | 1,230k |
| Test | 2,949 | 223k | 1,298k |

Table 1: Statistics for the PHI-ML corpus.

## 4 Restoring text using PYTHIA

**PYTHIA's architecture** is a sequence-to-sequence (Sutskever et al., 2014) based neural network architecture, consisting of a Long-Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) encoder, an LSTM decoder, and an attention mechanism introduced by Luong et al. (2015); Bahdanau et al. (2014). The encoder takes an inscription text $x$ as input, where the symbol '-' denotes the missing characters, and '?' the blanks to be predicted. The input characters are first passed through a lookup table with learnable embedding vectors. Next, the encoded sequence is used as input for the decoder, which is trained to predict the content of the '?' characters, as shown in Figure 2. Attention allows the decoder to "attend" to parts of the input sequence relevant to the current output, thus improving the modelling of long-term dependencies. To further improve performance, we designed PYTHIA's encoder to take an additional input stream of word embeddings, as it is difficult to model the word-level context using only character-level information. Thus, we generated a list of the 100k most frequent words appearing in PHI-ML, and using a separate lookup table we concatenated at each time-step the embedding of each character with the embedding of the word it belongs to. Words that do not appear in the list, or that contain missing characters were mapped to 'unk', an embedding for unknown words. Figure 2 illustrates PYTHIA processing the phrase μηδέν ἄγαν. Finally, to allow better modelling we used a bidirectional LSTM encoder and refer to this architecture as PYTHIA-BI-WORD. Further details are given in Appendix A.

**Obtaining suggestions.** To better aid the epigraphist's task, PYTHIA returns multiple predictions as well as the level of confidence for each result, rather than a single prediction per text restoration. Specifically, we provide a set of the Top 20 predictions decoded using beam search.

## 5 Experimental evaluation

The ground-truths for incomplete epigraphic texts were lost over millennia. Consequently, in order to generate a ground-truth sequence, we artificially removed part of the input text and treated this as the ground-truth sequence. On each training step we selected an inscription and sampled a start index and a length value $\in [100, 1000]$, and extracted the context text $x$, which was then used as input. Within $x$, we sampled a new start index and length $\in [1, 10]$ to select the target sequence $y$; its characters' positions were replaced with the special symbol '?', which denotes the positions to be predicted. The test and validation sets used the maximum context length. Beam search with a beam width of 100 was used to decode hypothesis. To simplify comparisons, all AG accentuation was discarded, as inputting accents was time-consuming for the human evaluations described in the following paragraph. This decision did not noticeably influence the reported scores.

### 5.1 Methods evaluated

**ANCIENT HISTORIAN.** Because text restoration is an extremely time-consuming task even for an expert epigraphist, we set out to evaluate the difficulty of the restoration task at hand - and thereby judge the impact of our work - with the help of two doctoral students with epigraphical expertise. The scholars were allowed to use the training set to search for "parallels", and made an average of 50 restorations in 2 hours, with a 57.3% character error rate (CER).

**LM PHILOLOGY.** To evaluate the performance of a model using "parallels", we trained a LM. Since large parts of the text are garbled, making complete words unidentifiable, and because BERT was not an option, the LM works at a character-

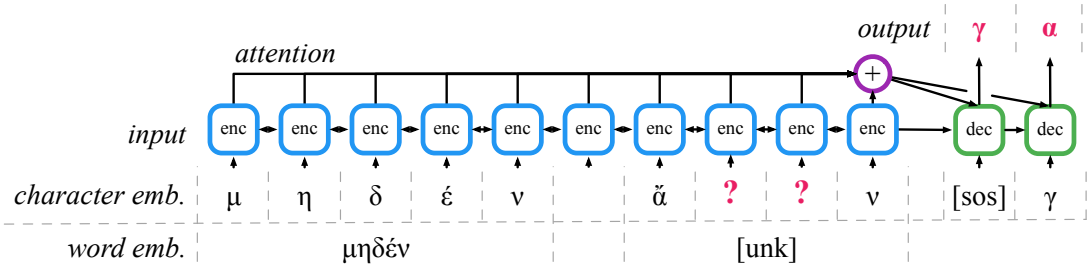


Figure 2: PYTHIA-BI-WORD processing the phrase μηδέν ἄγαν (mēdén ágan) "nothing in excess", a fabled maxim inscribed on Apollo's temple in Delphi. The letters "γα" are the characters to be predicted, and are annotated with '?'. Since ἄ??ν is not a complete word, its embedding is treated as unknown ('unk'). The decoder outputs correctly "γα".

level and is based on the setup of Zaremba et al. (2015) (Appendix B). The LM was trained on two larger digital corpora of literary AG texts ("philology"), First1KGreek and Perseus, and evaluated on PHI-ML.

**LM PHILOLOGY & EPIGRAPHY.** LM jointly trained on First1KGreek, Perseus and PHI-ML.

**LM EPIGRAPHY.** LM trained on PHI-ML.

**PYTHIA-UNI.** An ablation architecture, using only characters as input and unidirectional LSTM.

**PYTHIA-BI.** Similar to the PYTHIA-UNI ablation, but with a bidirectional LSTM.

**PYTHIA-BI-WORD.** This is our proposed model of choice, which uses a bidirectional LSTM and both characters and words as inputs.

### 5.2 Results

The aforementioned methods were evaluated using: a) the character error rate (CER) of the top prediction and the target sequence, b) the Top-20 accuracy score, where we ascertain whether the ground-truth sequence exists within the first 20 predictions. The latter evaluates the effectiveness of PYTHIA as an assistive tool providing restoration suggestions to epigraphists. As shown in Table 2, the ancient historians' restorations had a CER of 57.3%, which is telling of the difficulty of the task. The language model trained on epigraphic datasets performed slightly better, with a CER of 57.3%. Interestingly, the two attempts to use larger philological datasets performed worse. This is very likely due to a divergence in epigraphical and literary cultures. The CER of the unidirectional PYTHIA-UNI and the bidirectional PYTHIA-BI alternatives were 42.2% and 32.5% respectively. The top score was therefore achieved by the bidirectional PYTHIA-BI-WORD, which took both word and character embeddings as inputs, with a CER of 30.1%. Furthermore, the ground-truth appeared among the 20 most probable predictions of PYTHIA-BI-WORD 73.5% of the times, which indicates that it could be a uniquely effective assistive tool.

| Method | CER | Top-20 |
|---|---|---|
| Ancient Historian | 57.3% | – |
| LM Philology | 68.1% | 26.0% |
| LM Philology & Epigraphy | 65.0% | 28.8% |
| LM Epigraphy | 52.7% | 47.0% |
| PYTHIA-UNI | 42.2% | 60.6% |
| PYTHIA-BI | 32.5% | 71.1% |
| **PYTHIA-BI-WORD** | **30.1%** | **73.5%** |

Table 2: Predictive performance on PHI-ML.

### 5.3 The importance of context

The presence of context information is a determining factor in the accuracy of epigraphic restorations. We therefore evaluated the impact of different textual lengths acting as augmented context on the Top-20 accuracy measure of PYTHIA. As can be seen in Figure 3, the correlation between the "context length" and the predictive performance of our model is positive. Specifically, the performance peaks around 500 characters of context

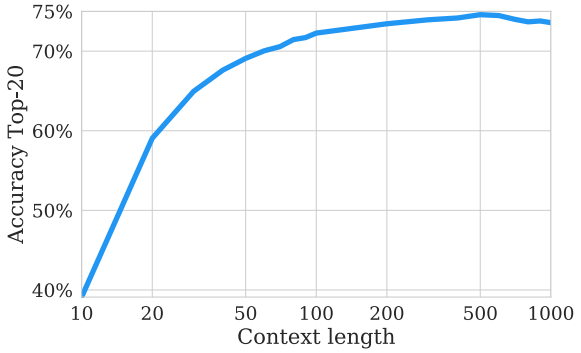


Figure 3: Predictive performance of the PYTHIA-BI-WORD model under different context lengths.

length. Furthermore, Figure 3 exemplifies the increased difficulty faced by the model when only a short context length (e.g. 20 characters) is offered. The latter scenario recalls the similar difficulties encountered by string-matching and "parallel" search approaches, where the search queries would often be short.

### 5.4 Visualising PYTHIA's attention

We set up an example modifying lines b.8 - c.5 of the inscription MDAI(A) 32 (1907) 428, 275 (PHI ID PH316753), to evaluate PYTHIA's receptiveness to context information and visualise the attention weights at each decoding step. In the text of Figure 4, the last word is a Greek personal name ending in -ου. We set ἀπολλοδώρου ("Apollodorou") as the personal name, and hid its first 9 characters. This name was specifically chosen because it already appears within the input text. Figure 4 illustrates the attention weights for decoding the first 4 missing characters. To aid visualisation, the weights were separately scaled between 0 and 1 within the area of the characters to be predicted ('?') in green, and of the rest of the text in blue; the magnitude is represented by the colour intensity. As can be seen, PYTHIA is attending to the contextually-relevant parts of the text: specifically, ἀπολλοδώρου. The name is correctly predicted. As a litmus test, we substituted ἀπολλοδώρου in the input text with another personal name of the same length: ἀρτεμιδώρου ("Artemidorou"). The predicted sequence alters accordingly to ἀρτεμιδώρ, thereby illustrating the importance of context in the prediction process.

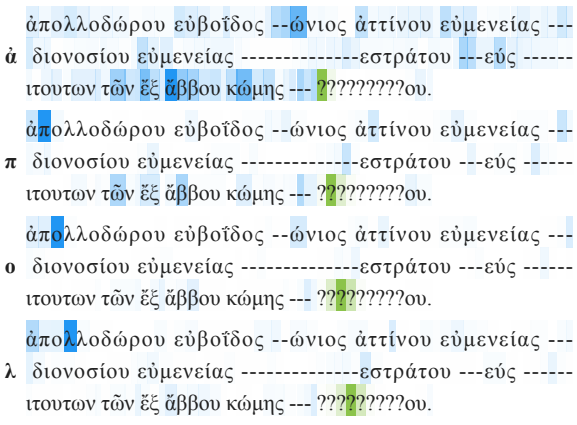


Figure 4: Visualisation of the attention weights for the decoding of the first 4 missing characters. The ground-truth text $y$ = ἀπολλοδώρ appears in the input text, and PYTHIA attends to the relevant parts of the sequence.

### 5.5 Restoring full texts

We then applied PYTHIA iteratively in order to predict all the missing text of an AG inscription, comparing PYTHIA's predictions with an edition of reference (Rhodes and Osborne, 2003). In Figure 5 the correct restorations are highlighted in colour blue and erroneous ones are in purple. In a real-world scenario, PYTHIA would provide more than one hypothesis to the epigraphist. The ground-truth sequence did in fact exist within the Top-20 hypotheses in nearly all cases, illustrating the efficacy of such technologies when paired with human decision-making.

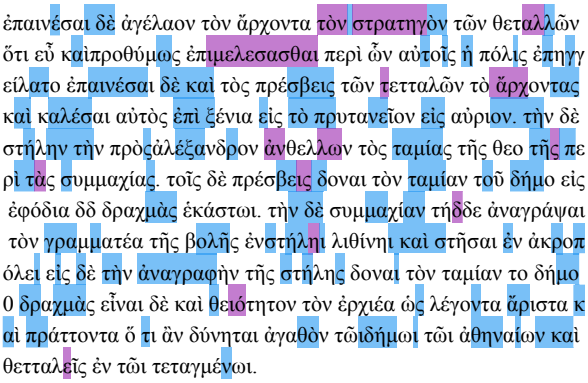


Figure 5: Sample restoration of the inscription *IG* II$^2$ 116, lines 34 - 48. Restorations are in colour blue when correct, purple when incorrect.

## 6 Conclusions

PYTHIA is the first ancient text restoration model of its kind. Our experimental evaluation and ablation studies illustrate the validity of our design decisions, and illuminate the ways PYTHIA can assist, guide and advance the ancient historian's task - and digital humanities proper. The combination of machine learning and epigraphy has the potential to impact meaningfully the study of inscribed textual cultures, both ancient and modern. By open-sourcing PYTHIA, and PHI-ML's processing pipeline, we hope to aid future research and inspire further interdisciplinary work.


### Acknowledgements

We would like to thank Leah Lazar for her valuable contributions; Brendan Shillingford, Misha Denil, Çağlar Gülçehre, and Nando de Freitas for the helpful comments and discussions; and the Packard Humanities Institute for having made this data digitally available.

## References


Giuseppe Amato, Fabrizio Falchi, and Lucia Vadicamo. 2016. Visual recognition of ancient inscriptions using convolutional neural network and fisher vector. *Journal on Computing and Cultural Heritage*, 9(4):21.

Meduri Avadesh and Navneet Goyal. 2018. Optical character recognition for sanskrit using convolution neural networks. In *International Workshop on Document Analysis Systems*, pages 447–452.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Ondrej Bajgar, Rudolf Kadlec, and Jan Kleindienst. 2017. Embracing data abundance: Booktest dataset for reading comprehension. In *International Conference on Learning Representations Workshop*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179.

John Bodel. 2012. Epigraphy and the ancient historian. In *Epigraphic Evidence*, pages 27–82. Routledge.

Jan Botha and Phil Blunsom. 2014. Compositional morphology for word representations and language modelling. In *International Conference on Machine Learning*, pages 1899–1907.

Gülcan Can, Jean-Marc Odobez, and Daniel Gatica-Perez. 2016. Evaluating shape representations for maya glyph classification. *Journal on Computing and Cultural Heritage*, 9(3):14.

Manfred Clauss. 2012. Epigraphik-datenbank clauss-slaby. Accessed on 2019-04-24.

Gregory Ralph Crane, Monica Berti, Annette Geßner, Matthew Munson, and Tabea Selle. 2014. Open greek and latin project. http://www.dh.uni-leipzig.de/wo/projects/open-greek-and-latin-project/. Accessed on 2019-04-24.

John Davies and John Wilkes. 2012. *Epigraphy and the historical sciences*. British Academy.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Shira Faigenbaum-Golovin, Arie Shaus, Barak Sober, David Levin, Nadav Na'aman, Benjamin Sass, Eli Turkel, Eli Piasetzky, and Israel Finkelstein. 2016. Algorithmic handwriting analysis of judaha's military correspondence sheds light on composition of biblical texts. *National Academy of Sciences*, 113(17):4664–4669.

William Fedus, Ian J. Goodfellow, and Andrew M. Dai. 2018. Maskgan: Better text generation via filling in the _____. In *International Conference on Learning Representations*.

Angelika Garz, Nicole Eichenberger, Marcus Liwicki, and Rolf Ingold. 2014. Hisdoc 2.0: Toward computer-assisted paleography. *Manuscript Cultures, Natural Sciences and Technology in Manuscript Studies*, 7.

Laura Gawlinski. 2017. Review: Packard humanities institute's searchable greek inscriptions. https://classicalstudies.org/scs-blog/laura-gawlinski/review-packard-humanities-institutes-searchable-greek-inscriptions. Accessed on 2019-08-26.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in neural information processing systems*, pages 1693–1701.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The goldilocks principle: Reading children's books with explicit memory representations. In *International Conference on Learning Representations*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Rana S Hussien, Azza A Elkhidir, and Mohamed G Elnourani. 2015. Optical character recognition of arabic handwritten characters using neural network. In *International Conference on Computing, Control, Networking, Electronics and Embedded Systems Engineering*, pages 456–461. IEEE.

Kyuyeon Hwang and Wonyong Sung. 2017. Character-level language modeling with hierarchical recurrent neural networks. In *International Conference on Acoustics, Speech and Signal Processing*, pages 5720–5724. IEEE.

Paul A. Iversen. 2007. The Packard Humanities Institute - Greek epigraphy project and the revolution in Greek epigraphy. *Abgadiyat*, 2.1.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *American Association for Artificial Intelligence*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gáabor Melis, and Edward Grefenstette. 2018. The narrativeqa reading comprehension challenge. *Transactions of the Association of Computational Linguistics*, 6:317–328.

John Lee and Dag Haug. 2010. Porting an ancient greek and latin treebank. In *International Conference on Language Resources and Evaluation*.

Jiaming Luo, Yuan Cao, and Regina Barzilay. 2019. Neural decipherment via minimum-cost flow: from ugaritic to linear b. *arXiv preprint arXiv:1906.06718*.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Empirical Methods in Natural Language Processing*, pages 1412–1421.

Harold Mattingly. 1996. *The Athenian empire restored: epigraphic and historical studies*. University of Michigan Press.

Tomáš Mikolov, Anoop Deoras, Stefan Kombrink, Lukáš Burget, and Jan Černockỳ. 2011. Empirical evaluation and combination of advanced language modeling techniques. In *International Speech Communication Association*.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *International Speech Communication Association*.

Tomáš Mikolov, Ilya Sutskever, Anoop Deoras, Hai-Son Le, Stefan Kombrink, and Jan Cernocky. 2012. Subword language modeling with neural networks. *Preprint*, 8.

Yasumasa Miyamoto and Kyunghyun Cho. 2016. Gated word-character recurrent language model. In *Empirical Methods in Natural Language Processing*, pages 1992–1997.

Émilie Pagé-Perron, Maria Sukhareva, Ilya Khait, and Christian Chiarcos. 2017. Machine translation and automated analysis of the sumerian language. In *Association for Computational Linguistics*.

Satish Palaniappan and Ronojoy Adhikari. 2017. Deep learning the indus script. *arXiv preprint arXiv:1702.00523*.

Michail Panagopoulos, Constantin Papaodysseus, Panayiotis Rousopoulos, Dimitra Dafi, and Stephen Tracy. 2009. Automatic writer identification of ancient greek inscriptions. *Transactions on pattern analysis and machine intelligence*, 31(8):1404–1414.

Rajesh PN Rao, Nisha Yadav, Mayank N Vahia, Hrishikesh Joglekar, R Adhikari, and Iravatham Mahadevan. 2009a. Entropic evidence for linguistic structure in the indus script. *Science*, 324(5931):1165–1165.

Rajesh PN Rao, Nisha Yadav, Mayank N Vahia, Hrishikesh Joglekar, R Adhikari, and Iravatham Mahadevan. 2009b. A markov model of the indus script. *Proceedings of the National Academy of Sciences*, 106(33):13685–13690.

Rajesh PN Rao, Nisha Yadav, Mayank N Vahia, Hrishikesh Joglekar, Ronojoy Adhikari, and Iravatham Mahadevan. 2010. Entropy, the indus script, and language. *Computational Linguistics*, 36(4):795–805.

Peter John Rhodes and Robin Osborne. 2003. *Greek historical inscriptions: 404-323 BC*. Oxford University Press, USA.

Henriette Roued-Cunliffe. 2010. Towards a decision support system for reading ancient documents. *Literary and linguistic computing*, 25(4):365–379.

Arie Shaus. 2017. *Computer Vision and Machine Learning Methods for Analyzing First Temple Period Inscriptions*. Ph.D. thesis, Tel Aviv University.

David Smith, Jeffrey Rydberg-Cox, and Gregory Crane. 2000. The perseus project: A digital library for the humanities. *Literary and Linguistic Computing*, 15(1):15–25.

A. Soumya and G. Hemantha Kumar. 2014. Classification of ancient epigraphs into different periods using random forests. In *International Conference on Signal and Image Processing*, pages 171–178. IEEE.

Thendral Suganya and Subramaniam Murugavalli. 2017. Feature selection for an automated ancient tamil script classification system using machine learning techniques. In *International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies*, pages 1–6. IEEE.

Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *International Conference on Machine Learning*, pages 1017–1024.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Melissa Terras and Paul Robertson. 2006. *Image to interpretation: an intelligent system to aid historians in reading the Vindolanda texts*. Oxford University Press.

The Packard Humanities Institute. 2005. PHI greek inscriptions. https://inscriptions.packhum.org/. Accessed on 2019-04-24.

Stephen V. Tracy and Constantin Papaodysseus. 2009. The study of hands on greek inscriptions: The need for a digital approach. *American Journal of Archaeology*, 113(1):99–102.

Alessandro Vatri and Barbara McGillivray. 2018. The diorisis ancient greek corpus. *Research Data Journal for the Humanities and Social Sciences*.

Arthur Woodhead. 1967. *The study of Greek inscriptions*, volume 424. Cambridge University Press Archive.

Qizhe Xie, Guokun Lai, Zihang Dai, and Eduard Hovy. 2018. Large-scale cloze test dataset created by teachers. In *Empirical Methods in Natural Language Processing*, pages 2344–2356.

Nisha Yadav, Hrishikesh Joglekar, Rajesh PN Rao, Mayank N Vahia, Ronojoy Adhikari, and Iravatham Mahadevan. 2010. Statistical analysis of the indus script using n-grams. *PLoS One*, 5(3):e9506.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2015. Recurrent neural network regularization. In *International Conference on Learning Representations*.

Zhuosheng Zhang, Yafang Huang, Pengfei Zhu, and Hai Zhao. 2018. Effective character-augmented word embedding for machine reading comprehension. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 27–39. Springer.

# Embedding Lexical Features via Tensor Decomposition for Small Sample Humor Recognition

**Zhenjie Zhao**[1*] **Andrew Cattle**[1*] **Evangelos E. Papalexakis**[2†] **Xiaojuan Ma**[1]
[1] Department of Computer Science & Engineering, HKUST
[2] Department of Computer Science & Engineering, University of California, Riverside
{zzhaoao, acattle}@connect.ust.hk, epapalex@cs.ucr.edu, mxj@cse.ust.hk

## Abstract

We propose a novel tensor embedding method that can effectively extract lexical features for humor recognition. Specifically, we use word-word co-occurrence to encode the contextual content of documents, and then decompose the tensor to get corresponding vector representations. We show that this simple method can capture features of lexical humor effectively for continuous humor recognition. In particular, we achieve a distance of 0.887 on a global humor ranking task, comparable to the top performing systems from SemEval 2017 Task 6B (Potash et al., 2017) but without the need for any external training corpus. In addition, we further show that this approach is also beneficial for small sample humor recognition tasks through a semi-supervised label propagation procedure, which achieves about 0.7 accuracy on the *16000 One-Liners* (Mihalcea and Strapparava, 2005) and *Pun of the Day* (Yang et al., 2015) humour classification datasets using only 10% of known labels.

## 1 Introduction

Recognizing humor automatically is an important step for natural human-computer interaction (Shahaf et al., 2015). While early works tend to frame humor recognition as a binary classification task (Mihalcea and Strapparava, 2005; Yang et al., 2015), the last few years have seen the emergence of humor recognition as a pairwise relative ranking task (Cattle and Ma, 2016; Shahaf et al., 2015). In addition to pairwise ranking, SemEval 2017 Task 6 also includes a global ranking subtask. However, the majority of submissions build global rankings using a series of pairwise comparisons (Potash et al., 2017). Only Yan and Pedersen (2017) attempt to predict global rankings directly, ranking documents inversely to their probability according to an n-gram language model.

State-of-the-art humor recognition algorithms usually require a considerable amount of training data with labels to learn effective features (Yang et al., 2015). However, such data are difficult to obtain – especially fine-grained humor annotations. First, the humor judgments differ from individual to individual. Thus, collecting perceptually consistent human labels is expensive and time-consuming. Second, fine-grained degrees of humor add a further challenge. Therefore, methods on small sample learning or even unsupervised rule-based methods merit investigation.

In this paper, considering the importance of lexical information for humor recognition (Radev et al., 2015), we propose a tensor decomposition method to capture the contextual nuances of a corpus. This allows us to model the lexical similarity of sentences regardless of the size of the corpus. In this way, we can rank the degree of humor effectively via lexical centrality (Radev et al., 2015), namely, regarding the distance to the lexical center as an indicator of the degree of humor. Experimental results on the SemEval 2017 Task 6 dataset (Potash et al., 2017) show that *without external training data*, the tensor embedding method can achieve performance equivalent to the second place on SemEval 2017 Task 6B without the need for any external training corpus. In addition, by applying a semi-supervised label propagation procedure (Zhou et al., 2003), we can also use the tensor embedding method for small sample humor recognition, achieving about 0.7 accuracy with only 10% of known labels on the *16000 One-Liners* (Mihalcea and Strapparava, 2005) and *Pun of the Day* (Yang et al., 2015) datasets.

[*] Zhenjie Zhao and Andrew Cattle contributed equally to this work.

[†] E. Papalexakis was supported by a UCR-China collaboration grant by the Bourns College of Engineering at UCR, and by the National Science Foundation CDSE Grant no. OAC-1808591



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6376–6381, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

The contributions of this paper are: 1) we propose a tensor embedding method to model the lexical features of documents, which can capture lexical similarity effectively regardless of the size of the corpus, 2) we show that the lexical features can be used effectively for fine-grained humor ranking and small sample humor recognition. Our implementation is open-sourced, and can be found at https://github.com/zhaozj89/TensorEmbeddingNLP.

## 2 Related Work

### 2.1 Humor Feature Extraction

Modeling and learning humor features are critical for automatic humor recognition. Previous works tend to use a combination of phonological, stylistic, semantic, and content-based features. Phonological features include acoustic features extracted from sitcom audio tracks (Bertero and Fung, 2016) and "phonetic embeddings" generated using a character-to-phoneme LSTM encoder-decoder (Donahue et al., 2017). Stylistic features include alliteration, rhyming, negative sentiment, and adult slang (Mihalcea and Strapparava, 2005) as well as emotional scenarios (Reyes et al., 2012). Semantic features range from attempts to measure incongruity (Cattle and Ma, 2018; Shahaf et al., 2015; Yang et al., 2015) to the use of word embeddings as inputs to neural models (Bertero and Fung, 2016; Donahue et al., 2017). Content-based approaches include word frequency (Mihalcea and Strapparava, 2005), n-gram probability (Yan and Pedersen, 2017), and lexical centrality (Radev et al., 2015).

Centrality is based on the observation that humorous responses to common stimuli tend to cluster around a small number of core jokes (Radev et al., 2015; Shahaf et al., 2015), with more central documents benefiting from "wisdom of the crowd". While most humor features involve making population-level inferences based on document-level features, centrality is instead population-level feature directly. Radev et al. (2015) calculate their centrality feature using LexRank, a graph-based text summarization method (Erkan and Radev, 2004). Compared with more traditional lexical similarity measures like tf-idf, this method is better suited to short humor texts due to their short lengths leading to sparse vector representations (Erkan and Radev, 2004).

### 2.2 Small Sample Humor Recognition

Once the humor features have been extracted, the next step is training a machine learning model to make predictions. Although learning-based methods have shown significant performance improvement recently (Yang et al., 2015), one of their main bottlenecks is the lack of appropriate training corpora. While previous works have employed data crawled from websites (Mihalcea and Strapparava, 2005; Yang et al., 2015), Twitter (Cattle and Ma, 2016; Reyes et al., 2012), sitcom subtitles (Bertero and Fung, 2016; Purandare and Litman, 2006), or the New Yorker Cartoon Caption Contest (Radev et al., 2015; Shahaf et al., 2015), these datasets are generally not released publicly. Owing to the difficulty in obtaining fine-grained labeled humor data, it is critical to study how to recognize humor by a small training sample or even without labeled data.

## 3 Method

### 3.1 Tensor Embedding

Contextual patterns of words can be used to measure lexical similarity for humor recognition. State-of-the-art learning-based approaches like doc2vec (Le and Mikolov, 2014) or sent2vec (Pagliardini et al., 2018) usually require a large amount of data. This is difficult to obtain for humor recognition. We propose to use a novel tensor decomposition method to obtain lexical features of short humor texts. To capture lexical similarity for humor recognition, we propose to represent the tensor through a novel word-word co-occurrence method, which has only been explored in the context of fake news detection (Hosseinimotlagh and Papalexakis, 2018). Considering a corpus $\mathcal{D} = \{s_1, s_2, \ldots, s_D\}$ with $D$ sentences, we first build a vocabulary for it, namely, $w_1, w_2, \ldots, w_V$, where $V$ is the number of words. For each sentence $s$ in $\mathcal{D}$, we count the word-word co-occurrence in a small window $H$, and build a frequency matrix $\mathbf{W}_s \in \mathbb{Z}^{V \times V}$, where $\mathbb{Z}$ denotes the set of integers. In particular, $\mathbf{W}_s(i, j)$ indicates the frequency that word $w_i$ and $w_j$ co-occur in $s$ within the window $H$. In this way, we can capture the lexical patterns of $s$ in $\mathbf{W}_s$. We then stack all $\mathbf{W}_s$ as a three-dimensional tensor $\boldsymbol{\mathcal{W}} \in \mathbb{Z}^{V \times V \times D}$. The objective of tensor decomposition is to find an approximation $\hat{\boldsymbol{\mathcal{W}}}$ of $\boldsymbol{\mathcal{W}}$ so

| | Prompt | | | | | | Average |
|---|---|---|---|---|---|---|---|
| | **Christmas** | **Shakespeare** | **Bad Job** | **Break Up** | **Broadway** | **Cereal** | |
| Our system | 0.909 | 1.000 | 0.909 | 0.909 | 0.727 | 0.909 | 0.887 (±0.113) |
| Duluth | 0.818 | 0.909 | 1.000 | 0.636 | 1.000 | 0.909 | 0.872 (±0.137) |
| TakeLab | 0.909 | 0.909 | 1.000 | 0.818 | 1.000 | 0.818 | 0.908 (±0.081) |
| QUB | 0.818 | 0.909 | 0.818 | 1.000 | 1.000 | 0.909 | 0.924 (±0.081) |
| SVNIT | 0.818 | 1.000 | 0.909 | 1.000 | 1.000 | 0.818 | 0.938 (±0.089) |

Table 1: The results of our lexical centrality system using tensor embeddings along with the top four SemEval 2017 Task 6B systems reproduced from Potash et al. (2017). Prompt names refer to specific prompts from the SemEval 2017 Task 6 evaluation set.

| | **Pun of the Day** | | | | **16000 One-Liners** | | | |
|---|---|---|---|---|---|---|---|---|
| | Acc | Pre | Rec | F1 | Acc | Pre | Rec | F1 |
| 5% | 0.670 | 0.707 | 0.578 | 0.633 | 0.665 | 0.690 | 0.602 | 0.642 |
| 10% | 0.700 | 0.735 | 0.611 | 0.667 | 0.681 | 0.704 | 0.624 | 0.661 |
| 30% | 0.729 | 0.752 | 0.680 | 0.714 | 0.700 | 0.722 | 0.641 | 0.679 |
| 90% | 0.745 | 0.752 | 0.723 | 0.737 | 0.705 | 0.721 | 0.667 | 0.693 |
| Yang et al. (2015) | 0.7958 | 0.761 | 0.862 | 0.808 | 0.798 | 0.801 | 0.793 | 0.797 |

Table 2: The results of our label propagation system. XX% refers to XX% of the data used as training and (100-XX)% as test. Baseline results reproduced from Yang et al. (2015)

that:

$$\hat{\mathcal{W}} = \sum_{r=1}^{R} \mathbf{v}_r \otimes \mathbf{v}_r \otimes \mathbf{d}_r, \quad (1)$$

where $\mathbf{v}_r \in \mathbb{R}^V$, $\mathbf{d}_r \in \mathbb{R}^D$, $R$ is the pre-defined rank parameter, and $\otimes$ is the outer product, namely, $\mathbf{v}_r \otimes \mathbf{v}_r \otimes \mathbf{d}_r$ being a three-dimensional tensor, and

$$\mathbf{v}_r \otimes \mathbf{v}_r \otimes \mathbf{d}_r(i,j,k) = \mathbf{v}_r(i) \cdot \mathbf{v}_r(j) \cdot \mathbf{d}_r(k). \quad (2)$$

With the tensor decomposition, we can find low-rank embeddings of sentences that capture the similarity of contextual patterns (Hosseinimotlagh and Papalexakis, 2018). In particular, $\mathbf{C} = [\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_R] \in \mathbb{R}^{D \times R}$, where the $s$-row of $\mathbf{C}$ is the embedding vector of sentence $s$. The Euclidean distance of embeddings is used to measure the similarity of two sentences.

### 3.2 Lexical Centrality

We use lexical centrality to rank the degree of humor (Radev et al., 2015). While Radev et al. (2015) utilize a graph-based definition of centrality, we instead take a vector-space approach. Given the decomposed $\mathbf{C} = [\mathbf{c}_1^T, \mathbf{c}_2^T, \ldots, \mathbf{c}_D^T]^T$, we compute a centroid as the average $\mathbf{m}$ of all sentence vectors of a corpus:

$$\mathbf{m} = \frac{1}{D} \sum_{k=1}^{D} \mathbf{c}_k. \quad (3)$$

The Euclidean distance to the center is then taken as an indicator of the degree of humor. In other words, given two sentences $s_1$ and $s_2$ and their embeddings $\mathbf{x}_1$ and $\mathbf{x}_2$, $d(\mathbf{m}, \mathbf{x1}) < d(\mathbf{m}, \mathbf{x2})$ implies $s_1$ is funnier than $s_2$.

### 3.3 Label Propagation

With the lexical similarity captured by tensor embeddings, we can build a similarity graph, and use a label propagation algorithm (Zhou et al., 2003) for semi-supervised humor recognition. In this way, we can use only a small portion of labeled data to predict the remaining unlabeled data effectively (Zhou et al., 2003). In particular, with the tensor embeddings, we first find the $K$ nearest neighbors of each data point, and build a similarity graph $\mathcal{G}$. We then form an affinity matrix $W$, where $W_{ij} = 1$ if $i$ and $j$ are connected, otherwise, $W_{ij} = 0$. Afterwards, we iterate:

$$F(t+1) = \alpha W F(t) + (1-\alpha) Y, \quad (4)$$

and can get the results $F^*$ as the limit of this sequence. Equation (4) means we propagate the labels of each data point to its neighbors in a weighted average way, where $\alpha$ is the ratio of propagating labels each iteration. For each point $x_i$, its label is $y_i = \arg\max_{j \le c} F^*_{ij}$.

## 4 Experiment

To evaluate the effectiveness of the tensor embedding method, we conduct two experiments on global humor ranking and binary humor classification separately. The alternating least squares

method of CANDECOMP/PARAFAC tensor decomposition (Sidiropoulos et al., 2017) is used to calculate the low-rank sentence embeddings as implemented in the Matlab tensor toolbox [1].

### 4.1 Global Humor Ranking

To show the effectiveness of the lexical centrality of our tensor embedding method, we conduct an experiment on SemEval 2017 Task 6B (Potash et al., 2017) consisting of tweeted responses to specific thematic prompts generated as part of a TV show. For each prompt, the writing staff of the show pick a top 10 and an overall winner. These humor judgments are used as gold standard labels.

Tensor embeddings and centroids are computed on a per-prompt basis and responses are ranked according to their distance from the centroid. We run a grid search procedure to determine the optimal rank value as 100, the window size as 5. For evaluation, we adopt the same edit distance-based metric used in Potash et al. (2017).

The results of our lexical centrality system using tensor embeddings is shown in Table 1, where the official results of other state-of-the-art systems are taken from Potash et al. (2017). Our system outperforms all but the Duluth (Yan and Pedersen, 2017) system in the official results for SemEval 2017 Task 6B (Potash et al., 2017), making our performance equivalent to second place. It is notable that our system can perform well on the **Broadway** prompt, where other methods usually fail. Moreover, because our system does not have a learning procedure, the performance is more stable than others.

### 4.2 Binary Humor Classification

To show the effectiveness of label propagation of our tensor embedding method for small sample humor recognition, we conduct an experiment on two humor classification datasets *16000 One-Liners* (Mihalcea and Strapparava, 2005) and *Pun of the Day* (Yang et al., 2015). Similarly, we run a grid search procedure to find optimal parameters, and set the rank as 10, window size as 5, neighbor number as 50, $\alpha$ as 0.2. $F(0)$ is set as a zero matrix initially. For each dataset, we randomly select 5%, 10%, 30%, and 90% of the data for training. We run a 10-fold procedure, and report the average accuracy, precision, recall, and F1 score values.

The results of humor classification are shown in Table 2. Our own implementation of Yang et al. (2015) is included as a baseline. While Yang et al. (2015) uses a large portion of data for training and combine different features, we find that at similar portion of training data (90%), the results of our method are comparable to it. In addition, with only a small portion of training data, our method still achieves good results.

### 4.3 Discussion

#### 4.3.1 Lexical Centrality

The most notable aspect of our tensor embedding/lexical centrality approach is how little training data our system requires. Our system's unsupervised nature means that we do not need to use the 106 training prompts included with the SemEval 2017 Task 6 dataset. Our results are obtained exclusively using the six evaluation prompts. By comparison, almost all the systems reported in Potash et al. (2017) take a supervised approach and make full use of the training set. Furthermore, since we consider prompts one-at-a-time and since each prompt only contains approximately 100 responses, we are able to achieve a state-of-the-art performance with 100 training documents. The only system reported in Potash et al. (2017) to take an unsupervised approach is Duluth (Yan and Pedersen, 2017). Like ours, their results are obtained without using the training set. However, their system uses an n-gram language model trained on a 6.2GB subset of the News Commentary Corpus and the News Crawl Corpus. Similarly, most other systems use some form of external training corpora for training word embeddings, phoneme models, semantic models, and so on.

Another advantage of our approach is the ease of interpretability, in contrast to neural-based state-of-the-art baselines. Because our lexical feature is in an Euclidean space, we can compare and rank humor level more easily. Tweets labeled as "overall winners" exhibited a smaller mean distance from their respective centroids (0.848) than those labeled as "merely in the top 10" (0.942). These tweets then in turn exhibited smaller distances than those labeled as "not in the top 10" (1.00). A one-way ANOVA test gives mild evidence that overall winners are drawn from a different distribution than tweets not in the top 10 ($p = 0.106$). This slight result is likely due to the fuzzy nature of humor and the relatively small

[1] www.tensortoolbox.org

dataset. Finally, ad hoc analysis of tweets with distances > 2 revealed these to be mostly "not in the top 10".

#### 4.3.2 Label Propagation

Although the semi-supervised framework provides a good alternative for small sample humor recognition, our method still cannot achieve a state-of-the-art performance with the same portion of training data. There is still space to improve the method; for example, by modeling not only the lexical similarity, but also other features, such as word association (Cattle and Ma, 2016), and the like, that are important for humor recognition. In addition, label propagation cannot handle unbalanced data well. Adding prior knowledge of the ratio of labels, e.g., the unbalanced SemEval 2017 Task 6 dataset, also deserves further investigation.

## 5 Conclusion

In this paper, we show the importance of lexical features for small sample humor recognition. We propose a tensor embedding method to capture the lexical similarity effectively. Without training data, on SemEval 2017 Task 6B, we can achieve a relatively good result. Under a semi-supervised framework, the tensor embedding method can also achieve pretty good results for small sample humor classification. It is interesting to further investigate a unified tensor embedding model to combine not only lexical, but also other features that are important for the sense of humor.

## References


Dario Bertero and Pascale Fung. 2016. A long short-term memory framework for predicting humor in dialogues. In *2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2016-Proceedings of the Conference*, pages 130–135.

Andrew Cattle and Xiaojuan Ma. 2016. Effects of semantic relatedness between setups and punchlines in twitter hashtag games. In *Proceedings of the Workshop on Computational Modeling of People's Opinions, Personality, and Emotions in Social Media (PEOPLES)*, pages 70–79, Osaka, Japan. The COLING 2016 Organizing Committee.

Andrew Cattle and Xiaojuan Ma. 2018. Recognizing humour using word associations and humour anchor extraction. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1849–1858, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

David Donahue, Alexey Romanov, and Anna Rumshisky. 2017. HumorHawk at SemEval-2017 task 6: Mixing meaning and sound for humor recognition. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 98–102, Vancouver, Canada. Association for Computational Linguistics.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

Seyedmehdi Hosseinimotlagh and Evangelos E Papalexakis. 2018. Unsupervised content-based identification of fake news articles with tensor decomposition ensembles. In *WSDM 2018 Workshop on Misinformation and Misbehavior Mining on the Web (MIS2)*.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–1188–II–1196. JMLR.org.

Rada Mihalcea and Carlo Strapparava. 2005. Making computers laugh: Investigations in automatic humor recognition. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 531–538. Association for Computational Linguistics.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2018. Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features. In *NAACL 2018 - Conference of the North American Chapter of the Association for Computational Linguistics*.

Peter Potash, Alexey Romanov, and Anna Rumshisky. 2017. SemEval-2017 task 6: #HashtagWars: Learning a sense of humor. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 49–57, Vancouver, Canada. Association for Computational Linguistics.

Amruta Purandare and Diane Litman. 2006. Humor: Prosody analysis and automatic recognition for F*R*I*E*N*D*S*. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, pages 208–215. Association for Computational Linguistics.

Dragomir R. Radev, Amanda Stent, Joel R. Tetreault, Aasish Pappu, Aikaterini Iliakopoulou, Agustin Chanfreau, Paloma de Juan, Jordi Vallmitjana, Alejandro Jaimes, Rahul Jha, and Robert Mankoff. 2015. Humor in collective discourse: Unsupervised funniness detection in the new yorker cartoon caption contest. *CoRR*, abs/1506.08126.

Antonio Reyes, Paolo Rosso, and Davide Buscaldi. 2012. From humor recognition to irony detection: The figurative language of social media. *Data & Knowledge Engineering*, 74:1–12.

Dafna Shahaf, Eric Horvitz, and Robert Mankoff. 2015. Inside jokes: Identifying humorous cartoon captions. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 1065–1074, New York, NY, USA. ACM.

Nicholas D. Sidiropoulos, Lieven De Lathauwer, Xiao Fu, Kejun Huang, Evangelos E. Papalexakis, and Christos Faloutsos. 2017. Tensor decomposition for signal processing and machine learning. *IEEE Transactions on Signal Processing*, 65(13):3551–3582.

Xinru Yan and Ted Pedersen. 2017. Duluth at semeval-2017 task 6: Language models in humor detection. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 384–388, Vancouver, Canada. Association for Computational Linguistics.

Diyi Yang, Alon Lavie, Chris Dyer, and Eduard Hovy. 2015. Humor recognition and humor anchor extraction. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2367–2376, Lisbon, Portugal. Association for Computational Linguistics.

Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. 2003. Learning with local and global consistency. In *Proceedings of the 16th International Conference on Neural Information Processing Systems*, NIPS'03, pages 321–328, Cambridge, MA, USA. MIT Press.

# EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks

**Jason Wei**[1,2] **Kai Zou**[3]
[1]Protago Labs Research, Tysons Corner, Virginia, USA
[2]Department of Computer Science, Dartmouth College
[3]Department of Mathematics and Statistics, Georgetown University
jason.20@dartmouth.edu kz56@georgetown.edu

## Abstract

We present **EDA**: **e**asy **d**ata **a**ugmentation techniques for boosting performance on text classification tasks. EDA consists of four simple but powerful operations: synonym replacement, random insertion, random swap, and random deletion. On five text classification tasks, we show that EDA improves performance for both convolutional and recurrent neural networks. EDA demonstrates particularly strong results for smaller datasets; on average, across five datasets, training with EDA while using only 50% of the available training set achieved the same accuracy as normal training with all available data. We also performed extensive ablation studies and suggest parameters for practical use.

## 1 Introduction

Text classification is a fundamental task in natural language processing (NLP). Machine learning and deep learning have achieved high accuracy on tasks ranging from sentiment analysis (Tang et al., 2015) to topic classification (Tong and Koller, 2002), but high performance often depends on the size and quality of training data, which is often tedious to collect. Automatic data augmentation is commonly used in computer vision (Simard et al., 1998; Szegedy et al., 2014; Krizhevsky et al., 2017) and speech (Cui et al., 2015; Ko et al., 2015) and can help train more robust models, particularly when using smaller datasets. However, because it is challenging to come up with generalized rules for language transformation, universal data augmentation techniques in NLP have not been thoroughly explored.

Previous work has proposed some techniques for data augmentation in NLP. One popular study generated new data by translating sentences into French and back into English (Yu et al., 2018). Other work has used data noising as smoothing (Xie et al., 2017) and predictive language models for synonym replacement (Kobayashi, 2018). Although these techniques are valid, they are not often used in practice because they have a high cost of implementation relative to performance gain.

| Operation | Sentence |
|---|---|
| None | A sad, superior human comedy played out on the back roads of life. |
| SR | A ***lamentable***, superior human comedy played out on the ***backward*** road of life. |
| RI | A sad, superior human comedy played out on ***funniness*** the back roads of life. |
| RS | A sad, superior human comedy played out on ***roads*** back ***the*** of life. |
| RD | A sad, superior human out on the roads of life. |

Table 1: Sentences generated using EDA. SR: synonym replacement. RI: random insertion. RS: random swap. RD: random deletion.

In this paper, we present a simple set of universal data augmentation techniques for NLP called EDA (**e**asy **d**ata **a**ugmentation). To the best of our knowledge, we are the first to comprehensively explore text editing techniques for data augmentation. We systematically evaluate EDA on five benchmark classification tasks, showing that EDA provides substantial improvements on all five tasks and is particularly helpful for smaller datasets. Code is publicly available at `http://github.com/jasonwei20/eda_nlp`.

## 2 EDA

Frustrated by the measly performance of text classifiers trained on small datasets, we tested a number of augmentation operations loosely inspired by those used in computer vision and found that they helped train more robust models. Here, we present the full details of EDA. For a given sentence in the training set, we randomly choose and perform one of the following operations:



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6382–6388, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

1. **Synonym Replacement (SR):** Randomly choose $n$ words from the sentence that are not stop words. Replace each of these words with one of its synonyms chosen at random.
2. **Random Insertion (RI):** Find a random synonym of a random word in the sentence that is not a stop word. Insert that synonym into a random position in the sentence. Do this $n$ times.
3. **Random Swap (RS):** Randomly choose two words in the sentence and swap their positions. Do this $n$ times.
4. **Random Deletion (RD):** Randomly remove each word in the sentence with probability $p$.

Since long sentences have more words than short ones, they can absorb more noise while maintaining their original class label. To compensate, we vary the number of words changed, $n$, for SR, RI, and RS based on the sentence length $l$ with the formula $n=\alpha l$, where $\alpha$ is a parameter that indicates the percent of the words in a sentence are changed (we use $p=\alpha$ for RD). Furthermore, for each original sentence, we generate $n_{aug}$ augmented sentences. Examples of augmented sentences are shown in Table 1. We note that synonym replacement has been used previously (Kolomiyets et al., 2011; Zhang et al., 2015; Wang and Yang, 2015), but to our knowledge, random insertions, swaps, and deletions have not been extensively studied.

## 3 Experimental Setup

We choose five benchmark text classification tasks and two network architectures to evaluate EDA.

### 3.1 Benchmark Datasets

We conduct experiments on five benchmark text classification tasks: (1) **SST-2**: Stanford Sentiment Treebank (Socher et al., 2013), (2) **CR**: customer reviews (Hu and Liu, 2004; Liu et al., 2015), (3) **SUBJ**: subjectivity/objectivity dataset (Pang and Lee, 2004), (4) **TREC**: question type dataset (Li and Roth, 2002), and (5) **PC**: Pro-Con dataset (Ganapathibhotla and Liu, 2008). Summary statistics are shown in Table 5 in Supplemental Materials. Furthermore, we hypothesize that EDA is more helpful for smaller datasets, so we delegate the following sized datasets by selecting a random subset of the full training set with $N_{train}$={500, 2,000, 5,000, all available data}.

### 3.2 Text Classification Models

We run experiments for two popular models in text classification. **(1)** Recurrent neural networks (RNNs) are suitable for sequential data. We use a LSTM-RNN (Liu et al., 2016). **(2)** Convolutional neural networks (CNNs) have also achieved high performance for text classification. We implement them as described in (Kim, 2014). Details are in Section 9.1 in Supplementary Materials.

## 4 Results

In this section, we test EDA on five NLP tasks with CNNs and RNNs. For all experiments, we average results from five different random seeds.

### 4.1 EDA Makes Gains

We run both CNN and RNN models with and without EDA across all five datasets for varying training set sizes. Average performances (%) are shown in Table 2. Of note, average improvement was 0.8% for full datasets and 3.0% for $N_{train}$=500.

| | Training Set Size | | | |
|---|---|---|---|---|
| **Model** | 500 | 2,000 | 5,000 | full set |
| RNN | 75.3 | 83.7 | 86.1 | 87.4 |
| +EDA | 79.1 | 84.4 | 87.3 | 88.3 |
| CNN | 78.6 | 85.6 | 87.7 | 88.3 |
| +EDA | 80.7 | 86.4 | 88.3 | 88.8 |
| *Average* | 76.9 | 84.6 | 86.9 | 87.8 |
| +EDA | 79.9 | 85.4 | 87.8 | **88.6** |

Table 2: Average performances (%) across five text classification tasks for models with and without EDA on different training set sizes.

### 4.2 Training Set Sizing

Overfitting tends to be more severe when training on smaller datasets. By conducting experiments using a restricted fraction of the available training data, we show that EDA has more significant improvements for smaller training sets. We run both normal training and EDA training for the following training set fractions (%): {1, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100}. Figure 1(a)-(e) shows performance with and without EDA for each dataset, and 1(f) shows the averaged performance across all datasets. The best average accuracy without augmentation, 88.3%, was achieved using 100% of the training data. Models trained using EDA surpassed this number by achieving an

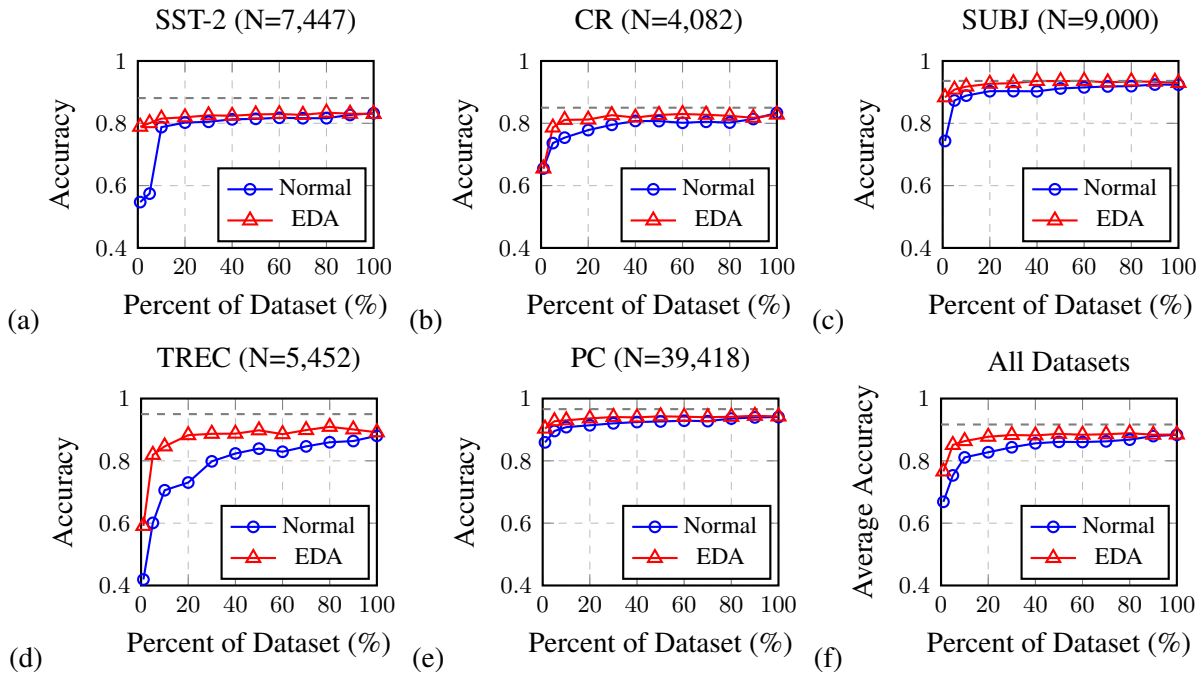


Figure 1: Performance on benchmark text classification tasks with and without EDA, for various dataset sizes used for training. For reference, the dotted grey line indicates best performances from Kim (2014) for SST-2, CR, SUBJ, and TREC, and Ganapathibhotla (2008) for PC.

average accuracy of 88.6% while only using 50% of the available training data.

### 4.3 Does EDA conserve true labels?

In data augmentation, input data is altered while class labels are maintained. If sentences are significantly changed, however, then original class labels may no longer be valid. We take a visualization approach to examine whether EDA operations significantly change the meanings of augmented sentences. First, we train an RNN on the pro-con classification task (PC) without augmentation. Then, we apply EDA to the test set by generating nine augmented sentences per original sentence. These are fed into the RNN along with the original sentences, and we extract the outputs from the last dense layer. We apply t-SNE (Van Der Maaten, 2014) to these vectors and plot their 2-D representations (Figure 2). We found that the resulting latent space representations for augmented sentences closely surrounded those of the original sentences, which suggests that for the most part, sentences augmented with EDA conserved the labels of their original sentences.

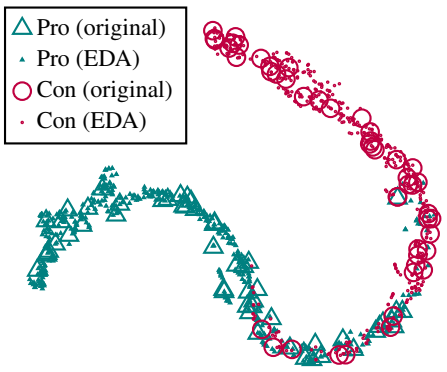


Figure 2: Latent space visualization of original and augmented sentences in the Pro-Con dataset. Augmented sentences (small triangles and circles) closely surround original sentences (big triangles and circles) of the same color, suggesting that augmented sentences maintianed their true class labels.

### 4.4 Ablation Study: EDA Decomposed

So far, we have seen encouraging empirical results. In this section, we perform an ablation study to explore the effects of each operation in EDA. Synonym replacement has been previously used (Kolomiyets et al., 2011; Zhang et al., 2015; Wang and Yang, 2015), but the other three EDA operations have not yet been explored. One could hypothesize that the bulk of EDA's performance gain is from synonym replacement, so we isolate each of the EDA operations to determine their individual ability to boost performance. For all four operations, we ran models using a single oper-

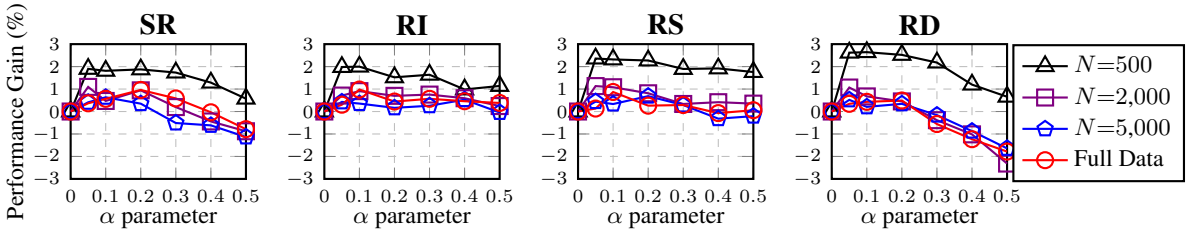


Figure 3: Average performance gain of EDA operations over five text classification tasks for different training set sizes. The $\alpha$ parameter roughly means "percent of words in sentence changed by each augmentation." SR: synonym replacement. RI: random insertion. RS: random swap. RD: random deletion.

ation while varying the augmentation parameter $\alpha$=$\{0.05, 0.1, 0.2, 0.3, 0.4, 0.5\}$ (Figure 3).

It turns out that all four EDA operations contribute to performance gain. For SR, improvement was good for small $\alpha$, but high $\alpha$ hurt performance, likely because replacing too many words in a sentence changed the identity of the sentence. For RI, performance gains were more stable for different $\alpha$ values, possibly because the original words in the sentence and their relative order were maintained in this operation. RS yielded high performance gains at $\alpha \leq 0.2$, but declined at $\alpha \geq 0.3$ since performing too many swaps is equivalent to shuffling the entire order of the sentence. RD had the highest gains for low $\alpha$ but severely hurt performance at high $\alpha$, as sentences are likely unintelligible if up to half the words are removed. Improvements were more substantial on smaller datasets for all operations, and $\alpha$=0.1 appeared to be a "sweet spot" across the board.

### 4.5 How much augmentation?

The natural next step is to determine how the number of generated augmented sentences per original sentence, $n_{aug}$, affects performance. In Figure 4, we show average performances over all datasets for $n_{aug}$=$\{1, 2, 4, 8, 16, 32\}$. For smaller training sets, overfitting was more likely, so generating many augmented sentences yielded large performance boosts. For larger training sets, adding more than four augmented sentences per original sentence was unhelpful since models tend to generalize properly when large quantities of real data are available. Based on these results, we recommend usage parameters in Table 3.

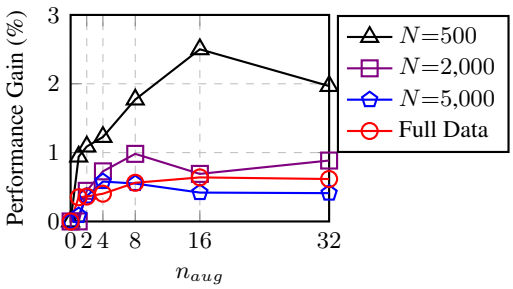


Figure 4: Average performance gain of EDA across five text classification tasks for various training set sizes. $n_{aug}$ is the number of generated augmented sentences per original sentence.

| $N_{train}$ | $\alpha$ | $n_{aug}$ |
|---|---|---|
| 500 | 0.05 | 16 |
| 2,000 | 0.05 | 8 |
| 5,000 | 0.1 | 4 |
| More | 0.1 | 4 |

Table 3: Recommended usage parameters.

## 5 Comparison with Related Work

Related work is creative but often complex. Back-translation (Sennrich et al., 2016), translational data augmentation (Fadaee et al., 2017), and noising (Xie et al., 2017) have shown improvements in BLEU measure for machine translation. For other tasks, previous approaches include task-specific heuristics (Kafle et al., 2017) and back-translation (Silfverberg et al., 2017; Yu et al., 2018). Regarding synonym replacement (SR), one study showed a 1.4% F1-score boost for tweet classification by finding synonyms with k-nearest neighbors using word embeddings (Wang and Yang, 2015). Another study found no improvement in temporal analysis when replacing headwords with synonyms (Kolomiyets et al., 2011), and mixed results were reported for using SR in character-level text classification (Zhang et al., 2015); however, neither work conducted extensive ablation studies.

Most studies explore data augmentation as a complementary result for translation or in a task-specific context, so it is hard to directly compare

EDA with previous literature. But there are two studies similar to ours that evaluate augmentation techniques on multiple datasets. Hu (2017) proposed a generative model that combines a variational auto-encoder (VAE) and attribute discriminator to generate fake data, demonstrating a 3% gain in accuracy on two datasets. Kobayashi (2018) showed that replacing words with other words that were predicted from the sentence context using a bi-directional language model yielded a 0.5% gain on five datasets. However, training a variational auto-encoder or bidirectional LSTM language model is a lot of work. EDA yields results on the same order of magnitude but is much easier to use because it does not require training a language model and does not use external datasets. In Table 4, we show EDA's ease of use compared with other techniques.

| Technique (#datasets) | LM | Ex Dat |
|---|---|---|
| Trans. data aug.[1] (1) | yes | yes |
| Back-translation[2] (1) | yes | yes |
| VAE + discrim.[3] (2) | yes | yes |
| Noising[4] (1) | yes | no |
| Back-translation[5] (2) | yes | no |
| LM + SR[6] (2) | yes | no |
| Contextual aug.[7] (5) | yes | no |
| SR - kNN[8] (1) | no | no |
| **EDA (5)** | **no** | **no** |

Table 4: Related work in data augmentation. #datasets: number of datasets used for evaluation. Gain: reported performance gain on all evaluation datasets. LM: requires training a language model or deep learning. Ex Dat: requires an external dataset.[9]

## 6 Discussion and Limitations

Our paper aimed to address the lack of standardized data augmentation in NLP (compared to vision) by introducing a set of simple operations that might serve as a baseline for future investigation. With the rate that NLP research has progressed in recent years, we suspect that researchers will soon find higher-performing augmentation techniques that will also be easy to use.

Notably, much of the recent work in NLP focuses on making neural models larger or more complex. Our work, however, takes the opposite approach. We introduce simple operations, the result of asking the fundamental question, *how can we generate sentences for augmentation without changing their true labels?* We do not expect EDA to be the go-to augmentation method for NLP, either now or in the future. Rather, we hope that our line of thought might inspire new approaches for universal or task-specific data augmentation.

Now, let's note many of EDA's limitations. Foremost, performance gain can be marginal when data is sufficient; for our five classification tasks, the average performance gain for was less than $1\%$ when training with full datasets. And while performance gains seem clear for small datasets, EDA might not yield substantial improvements when using pre-trained models. One study found that EDA's improvement was negligible when using ULMFit (Shleifer, 2019), and we expect similar results for ELMo (Peters et al., 2018) and BERT (Devlin et al., 2018). Finally, although we evaluate on five benchmark datasets, other studies on data augmentation in NLP use different models and datasets, and so fair comparison with related work is highly non-trivial.

## 7 Conclusions

We have shown that simple data augmentation operations can boost performance on text classification tasks. Although improvement is at times marginal, EDA substantially boosts performance and reduces overfitting when training on smaller datasets. Continued work on this topic could explore the theoretical underpinning of the EDA operations. We hope that EDA's simplicity makes a compelling case for further thought.

## 8 Acknowledgements

We thank Chengyu Huang, Fei Xing, and Yifang Wei for help with study design and paper revisions, and Chunxiao Zhou for insightful feedback. Jason Wei thanks Eugene Santos for inspiration.

[1](Fadaee et al., 2017) for translation
[2](Yu et al., 2018) for comprehension
[3](Hu et al., 2017) for text classification
[4](Xie et al., 2017) for translation
[5](Sennrich et al., 2016) for translation
[6](Kolomiyets et al., 2011) for temporal analysis
[7](Kobayashi, 2018) for text classification
[8](Wang and Yang, 2015) for tweet classification
[9]EDA does use a synonym dictionary, WordNet, but the cost of downloading it is far less than training a model on an external dataset, so we don't count it as an "external dataset."

## References


Xiaodong Cui, Vaibhava Goel, and Brian Kingsbury. 2015. Data augmentation for deep neural network acoustic modeling. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 23(9):1469–1477.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. Data augmentation for low-resource neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 567–573. Association for Computational Linguistics.

Murthy Ganapathibhotla and Bing Liu. 2008. Mining opinions in comparative sentences. In *Proceedings of the 22Nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 241–248, Stroudsburg, PA, USA. Association for Computational Linguistics.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 168–177, New York, NY, USA. ACM.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Toward controlled generation of text. In *ICML*.

Kushal Kafle, Mohammed Yousefhussien, and Christopher Kanan. 2017. Data augmentation for visual question answering. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 198–202. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882.

Tom Ko, Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. 2015. Audio augmentation for speech recognition. In *INTERSPEECH*.

Sosuke Kobayashi. 2018. Contextual augmentation: Data augmentation by words with paradigmatic relations. In *NAACL-HLT*.

Oleksandr Kolomiyets, Steven Bethard, and Marie-Francine Moens. 2011. Model-portability experiments for textual temporal analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11, pages 271–276, Stroudsburg, PA, USA. Association for Computational Linguistics.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2017. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, COLING '02, pages 1–7, Stroudsburg, PA, USA. Association for Computational Linguistics.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 2873–2879. AAAI Press.

Qian Liu, Zhiqiang Gao, Bing Liu, and Yuanlin Zhang. 2015. Automated rule selection for aspect extraction in opinion mining. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 1291–1297. AAAI Press.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *CoRR*, abs/1802.05365.

David Rolnick, Andreas Veit, Serge J. Belongie, and Nir Shavit. 2017. Deep learning is robust to massive label noise. *CoRR*, abs/1705.10694.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96. Association for Computational Linguistics.

Sam Shleifer. 2019. Low resource text classification with ulmfit and backtranslation. *CoRR*, abs/1903.09244.

Miikka Silfverberg, Adam Wiemerslage, Ling Liu, and Lingshuang Jack Mao. 2017. Data augmentation for morphological reinflection. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 90–99. Association for Computational Linguistics.

Patrice Simard, Yann LeCun, John S. Denker, and Bernard Victorri. 1998. Transformation invariance in pattern recognition-tangent distance and tangent propagation. In *Neural Networks: Tricks of*

*the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop*, pages 239–27, London, UK, UK. Springer-Verlag.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher Manning, Andrew Ng, and Christopher Potts. 2013. Parsing With Compositional Vector Grammars. In *EMNLP*.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2014. Going deeper with convolutions. *CoRR*, abs/1409.4842.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. pages 1422–1432.

Simon Tong and Daphne Koller. 2002. Support vector machine active learning with applications to text classification. *J. Mach. Learn. Res.*, 2:45–66.

Laurens Van Der Maaten. 2014. Accelerating t-sne using tree-based algorithms. *J. Mach. Learn. Res.*, 15(1):3221–3245.

William Yang Wang and Diyi Yang. 2015. That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using #petpeeve tweets. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2557–2563. Association for Computational Linguistics.

Ziang Xie, Sida I. Wang, Jiwei Li, Daniel Levy, Aiming Nie, Dan Jurafsky, and Andrew Y. Ng. 2017. Data noising as smoothing in neural network language models.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. *CoRR*, abs/1804.09541.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 649–657, Cambridge, MA, USA. MIT Press.

# Neural News Recommendation with Multi-Head Self-Attention

**Chuhan Wu**[1], **Fangzhao Wu**[2], **Suyu Ge**[1], **Tao Qi**[1], **Yongfeng Huang**[1],**and Xing Xie**[2]
[1]Department of Electronic Engineering, Tsinghua University, Beijing 100084, China
[2]Microsoft Research Asia, Beijing 100080, China
{wu-ch19, gsy17, qit16, yfhuang}@mails.tsinghua.edu.cn,
{fangzwu, xing.xie}@microsoft.com

## Abstract

News recommendation can help users find interested news and alleviate information overload. Precisely modeling news and users is critical for news recommendation, and capturing the contexts of words and news is important to learn news and user representations. In this paper, we propose a neural news recommendation approach with multi-head self-attention (NRMS). The core of our approach is a news encoder and a user encoder. In the news encoder, we use multi-head self-attentions to learn news representations from news titles by modeling the interactions between words. In the user encoder, we learn representations of users from their browsed news and use multi-head self-attention to capture the relatedness between the news. Besides, we apply additive attention to learn more informative news and user representations by selecting important words and news. Experiments on a real-world dataset validate the effectiveness and efficiency of our approach.

## 1 Introduction

Online news platforms such as Google News[1] and MSN News[2] have attracted many users to read news online (Das et al., 2007). Massive news articles are generated everyday and it is impossible for users to read all news to find their interested content (Phelan et al., 2011). Thus, personalized news recommendation is very important for online news platforms to target user interests and alleviate information overload (IJntema et al., 2010).

Learning accurate representations of news and users are two core tasks in news recommendation (Okura et al., 2017). Several deep learning based methods have been proposed for these tasks (**?**Kumar et al., 2017; Khattar et al., 2018; Wu et al., 2019b,c,a; Zhu et al., 2019; An et al., 2019). For example, Okura et al. (2017) proposed to learn news representations from news bodies via auto-encoders, and learn representations of users from their browsed news via GRU. However, GRU is quite time-consuming, and their method cannot capture the contexts of words. Wang et al. (2018) proposed to learn news representations from news titles via a knowledge-aware convolutional neural network (CNN), and learn representations of users based on the similarities between candidate news and their browsed news. However, CNN cannot capture the long-distance contexts of words, and their method cannot model the relatedness between browsed news.

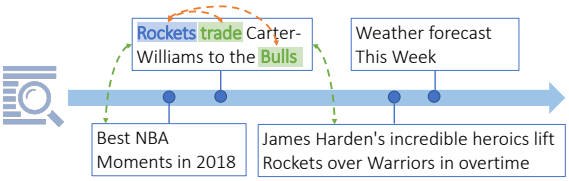


Figure 1: Several news browsed by an example user. Orange and green dashed lines represent the interactions between words and news respectively.

Our work is motivated by several observations. First, the interactions between words in news title are important for understanding the news. For example, in Fig. 1, the word "Rockets" has strong relatedness with "Bulls". Besides, a word may interact with multiple words, e.g., "Rockets" also has semantic interactions with "trade". Second, different news articles browsed by the same user may also have relatedness. For example, in Fig. 1 the second news is related to the first and the third news. Third, different words may have different importance in representing news. In Fig. 1, the word "NBA" is more informative than "2018". Besides, different news articles browsed by the same user may also have different importance in repre-

[1]https://news.google.com/
[2]https://www.msn.com/en-us/news



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6389–6394,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

senting this user. For example, the first three news articles are more informative than the last one.

In this paper, we propose a neural news recommendation approach with multi-head self-attention (NRMS). The core of our approach is a news encoder and a user encoder. In the news encoder, we learn news representations from news titles by using multi-head self-attention to model the interactions between words. In the user encoder, we learn representations of users from their browsing by using multi-head self-attention to capture their relatedness. Besides, we apply additive attentions to both news and user encoders to select important words and news to learn more informative news and user representations. Extensive experiments on a real-world dataset show that our approach can effectively and efficiently improve the performance of news recommendation.

## 2 Our Approach

Our NRMS approach for news recommendation is shown in Fig. 2. It contains three modules, i.e., *news encoder*, *user encoder* and *click predictor*.

### 2.1 News Encoder

The *news encoder* module is used to learn news representations from news titles. It contains three layers. The first one is word embedding, which is used to convert a news title from a sequence of words into a sequence of low-dimensional embedding vectors. Denote a news title with $M$ words as $[w_1, w_2, ..., w_M]$. Through this layer it is converted into a vector sequence $[\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_M]$.

The second layer is a word-level multi-head self-attention network (Vaswani et al., 2017; Wu et al., 2018). The interactions between words are important for learning news representations. For example, in the news title "Rockets Ends 2018 with a Win", the interaction between "Rockets" and "Win" is useful for understanding this news, and such long-distance interactions usually cannot be captured by CNN. In addition, a word may interact with multiple words in the same news. For instance, in above example the word "Rockets" has interactions with both "Ends" and "Win". Thus, we propose to use multi-head self-attention to learn contextual representations of words by capturing their interactions. The representation of the $i_{th}$ word learned by the $k_{th}$ attention head is

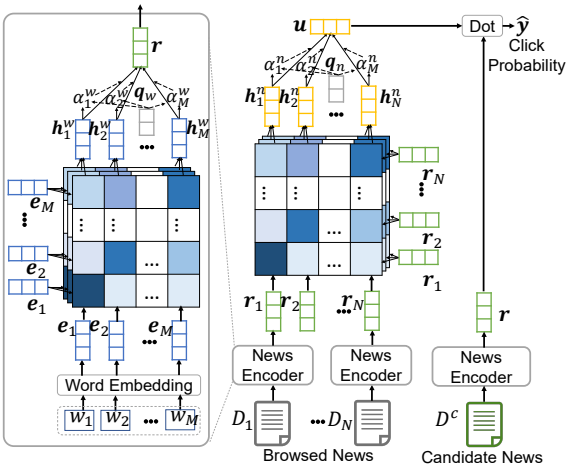


Figure 2: The framework of our *NRMS* approach.

computed as:

$$\alpha_{i,j}^k = \frac{\exp(\mathbf{e}_i^T \mathbf{Q}_k^w \mathbf{e}_j)}{\sum_{m=1}^{M} \exp(\mathbf{e}_i^T \mathbf{Q}_k^w \mathbf{e}_m)}, \quad (1)$$

$$\mathbf{h}_{i,k}^w = \mathbf{V}_k^w (\sum_{j=1}^{M} \alpha_{i,j}^k \mathbf{e}_j), \quad (2)$$

where $\mathbf{Q}_k^w$ and $\mathbf{V}_k^w$ are the projection parameters in the $k_{th}$ self-attention head, and $\alpha_{i,j}^k$ indicates the relative importance of the interaction between the $i_{th}$ and $j_{th}$ words. The multi-head representation $\mathbf{h}_i^w$ of the $i_{th}$ word is the concatenation of the representations produced by $h$ separate self-attention heads, i.e., $\mathbf{h}_i^w = [\mathbf{h}_{i,1}^w; \mathbf{h}_{i,2}^w; ...; \mathbf{h}_{i,h}^w]$.

The third layer is an additive word attention network. Different words in the same news may have different importance in representing this news. For example, in the second news of Fig. 1, the word "NFL" is more informative than "Today" for understanding this news. Thus, we propose to use attention mechanism to select important words in news titles for learning more informative news representations. The attention weight $\alpha_i^w$ of the $i$-th word in a news title is computed as:

$$a_i^w = \mathbf{q}_w^T \tanh(\mathbf{V}_w \times \mathbf{h}_i^w + \mathbf{v}_w), \quad (3)$$

$$\alpha_i^w = \frac{\exp(a_i^w)}{\sum_{j=1}^{M} \exp(a_j^w)}, \quad (4)$$

where $\mathbf{V}_w$ and $\mathbf{v}_w$ are projection parameters, and $\mathbf{q}_w$ is the query vector. The final representation of a news is the weighted summation of the contextual word representations, formulated as:

$$\mathbf{r} = \sum_{i=1}^{M} \alpha_i^w \mathbf{h}_i^w. \quad (5)$$

### 2.2 User Encoder

The *user encoder* module is used to learn the representations of users from their browsed news. It contains two layers. The first one is a news-level multi-head self-attention network. Usually, news articles browsed by the same user may have some relatedness. For example, in Fig. 1, the first two news articles are related. In addition, a news article may interact with multiple news articles browsed by the same user. Thus, we propose to apply multi-head self-attention to enhance the representations of news by capturing their interactions. The representation of the $i_{th}$ news learned by the $k_{th}$ attention head is formulated as follows:

$$\beta_{i,j}^k = \frac{\exp(\mathbf{r}_i^T \mathbf{Q}_k^n \mathbf{r}_j)}{\sum_{m=1}^M \exp(\mathbf{r}_i^T \mathbf{Q}_k^n \mathbf{r}_m)}, \tag{6}$$

$$\mathbf{h}_{i,k}^n = \mathbf{V}_k^n (\sum_{j=1}^M \beta_{i,j}^k \mathbf{r}_j), \tag{7}$$

where $\mathbf{Q}_k^n$ and $\mathbf{V}_k^n$ are parameters of the $k_{th}$ news self-attention head, and $\beta_{i,j}^k$ represents the relative importance of the interaction between the $j_{th}$ and the $k_{th}$ news. The multi-head representation of the $i_{th}$ news is the concatenation of the representations output by $h$ separate self-attention heads, i.e., $\mathbf{h}_i^n = [\mathbf{h}_{i,1}^n; \mathbf{h}_{i,2}^n; ...; \mathbf{h}_{i,h}^n]$.

The second layer is an additive news attention network. Different news may have different informativeness in representing users. For example, in Fig. 1 the first news is more informative than the fourth news in modeling user interest, since the latter one is usually browsed by massive users. Thus, we propose to apply the additive attention mechanism to select important news to learn more informative user representations. The attention weight of the $i_{th}$ news is computed as:

$$a_i^n = \mathbf{q}_n^T \tanh(\mathbf{V}_n \times \mathbf{h}_i^n + \mathbf{v}_n), \tag{8}$$

$$\alpha_i^n = \frac{\exp(a_i^n)}{\sum_{j=1}^N \exp(a_j^n)}, \tag{9}$$

where $\mathbf{V}_n$, $\mathbf{v}_n$ and $\mathbf{q}_n$ are parameters in the attention network, and $N$ is the number of the browsed news. The final user representation is the weighted summation of the representations of the news browsed by this user, which is formulated as:

$$\mathbf{u} = \sum_{i=1}^N \alpha_i^n \mathbf{h}_i^n. \tag{10}$$

### 2.3 Click Predictor

The *click predictor* module is used to predict the probability of a user clicking a candidate news. Denote the representation of a candidate news $D^c$ as $\mathbf{r}^c$. Following (Okura et al., 2017), the click probability score $\hat{y}$ is computed by the inner product of the user representation vector and the news representation vector, i.e., $\hat{y} = \mathbf{u}^T \mathbf{r}^c$. We also explored other kinds of scoring methods such as perception, but dot product shows the best performance and efficiency.

### 2.4 Model Training

Motivated by (Huang et al., 2013), we use negative sampling techniques for model training. For each news browsed by a user (regarded as a positive sample), we randomly sample $K$ news which are shown in the same impression but not clicked by the user (regarded as negative samples). We shuffle the orders of these news to avoid possible positional biases. Denote the click probability score of the positive and the $K$ negative news as $\hat{y}^+$ and $[\hat{y}_1^-, \hat{y}_2^-, ..., \hat{y}_K^-]$ respectively. These scores are normalized by the softmax function to compute the posterior click probability of a positive sample as follows:

$$p_i = \frac{\exp(\hat{y}_i^+)}{\exp(\hat{y}_i^+) + \sum_{j=1}^K \exp(\hat{y}_{i,j}^-)}. \tag{11}$$

We re-formulate the news click probability prediction problem as a pseudo $(K+1)$-way classification task, and the loss function for model training is the negative log-likelihood of all positive samples $\mathcal{S}$, which is formulated as follows:

$$\mathcal{L} = -\sum_{i \in \mathcal{S}} \log(p_i). \tag{12}$$

## 3 Experiments

### 3.1 Datasets and Experimental Settings

We conducted experiments on a real-world news recommendation dataset collected from MSN News[3] logs in one month (Dec. 13, 2018 to Jan. 12, 2019). The detailed statistics are shown in Table 1. The logs in the last week were used for test, and the rest were used for training. We randomly sampled 10% of training data for validation.

[3]https://www.msn.com/en-us/news

| # users | 10,000 | avg. # words per title | 11.29 |
|---|---|---|---|
| # news | 42,255 | # positive samples | 489,644 |
| # impressions | 445,230 | # negative samples | 6,651,940 |

Table 1: Statistics of our dataset.

In our experiments, the word embeddings are 300-dimensional and initialized by the Glove embedding (Pennington et al., 2014). The self-attention networks have 16 heads, and the output of each head is 16-dimensional. The dimension of the additive attention query vectors is 200. Following (Wu et al., 2019b), the negative sampling ratio $K$ is 4. Adam (Kingma and Ba, 2014) is used for model optimization. We apply 20% dropout to the word embeddings to mitigate overfitting. The batch size is 64. These hyperparameters are tuned on validation set. We conducted experiments on a machine with Xeon E5-2620 v4 CPUs and a GTX1080Ti GPU. We independently repeated each experiment 10 times and reported average results in terms of AUC, MRR, nDCG@5 and nDCG@10.

### 3.2 Performance Evaluation

We evaluate the performance of our approach by comparing it with several baseline methods, including: (1) *LibFM* (Rendle, 2012), a matrix factorization based recommendation method; (2) *DSSM* (Huang et al., 2013), deep structured semantic model; (3) *Wide&Deep* (Cheng et al., 2016), a popular neural recommendation method; (4) *DeepFM* (Guo et al., 2017), another popular neural recommendation method; (5) *DFM* (Lian et al., 2018), deep fusion model for news recommendation; (6) *DKN* (Wang et al., 2018), deep knowledge-aware network for news recommendation; (7) *Conv3D* (Khattar et al., 2018), a neural news recommendation method with 3-D CNNs to learn user representations; (8) *GRU* (Okura et al., 2017), a neural news recommendation method using GRU to learn user representations; (9) *NRMS*, our approach. In methods (1) and (3-5), we use one-hot encoded user ID, news ID and the TF-IDF features extracted from news titles as the model input. In methods (6-9), we all use news titles for fair comparison. The results of these methods are summarized in Table 2.

We have several observations from Table 2. First, neural recommendation methods such as *DSSM* and *NRMS* outperform traditional recommendation methods such as *LibFM* on news recommendation. This may be because neural networks can learn better representations of news and users than matrix factorization methods. Thus, it may be more appropriate to learn news and user representations via neural networks rather than craft them manually. Second, among the deep learning based methods, the methods which exploit the relatedness between news (e.g., *Conv3D*, *GRU* and *NRMS*) can outperform other methods. This may be because the news browsed by the same user usually have relatedness, and capturing the news relatedness is useful for understanding these news and modeling user interests. Third, our approach performs better than all baseline methods. This is because our approach can capture the interactions between both words and news via multi-head self-attention to enhance representation learning of news and users. Besides, our approach employs additive attention to select important words and news for learning informative news and user representations. These results validate the effectiveness of our approach.

| Methods | AUC | MRR | nDCG@5 | nDCG@10 |
|---|---|---|---|---|
| LibFM | 0.5661 | 0.2414 | 0.2689 | 0.3552 |
| DSSM | 0.5949 | 0.2675 | 0.2881 | 0.3800 |
| Wide&Deep | 0.5812 | 0.2546 | 0.2765 | 0.3674 |
| DeepFM | 0.5830 | 0.2570 | 0.2802 | 0.3707 |
| DFM | 0.5861 | 0.2609 | 0.2844 | 0.3742 |
| DKN | 0.6032 | 0.2744 | 0.2967 | 0.3873 |
| Conv3D | 0.6051 | 0.2765 | 0.2987 | 0.3904 |
| GRU | 0.6102 | 0.2811 | 0.3035 | 0.3952 |
| NRMS* | **0.6275** | **0.2985** | **0.3217** | **0.4139** |

Table 2: The results of different methods. *The improvement is significant at $p < 0.01$.

We also conducted experiments to compare the time efficiency of our approach with several popular news recommendation methods. The results are shown in Table 3. From the results, we find our approach has a smaller parameter size and a lower time complexity in learning news and user representations than existing news recommendation methods. In addition, different from *DKN*, our approach does not need to memorize the news browsing histories of users when computing the click probability scores. In addition, since our approach can be further accelerated by computing the hidden representations of different attention heads in parallel, our approach is more suitable for being deployed in large-scale news recommendation scenarios. These results validate the efficiency of our approach.

| Method | # Parameters | News Encoding Time | User Encoding Time |
|---|---|---|---|
| DKN | 681 K | 46.4 s | 10.1 min |
| Conv3D | 575 K | 29.8 min | 16.7 min |
| GRU | 541 K | 59.8 s | 148.0 min |
| NRMS | 530 K | 39.7 s | 6.7 min |

Table 3: The number of parameters in *NRMS* and baseline methods, and their time in encoding 1 million news and 1 million users. *Word embeddings are excluded.

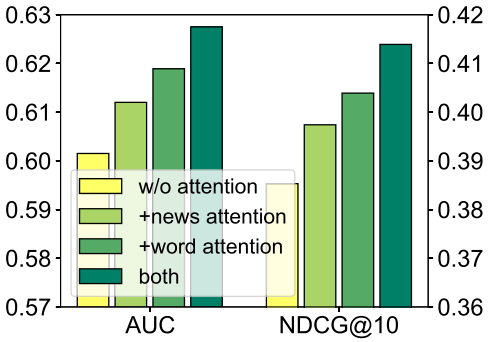


(a) Attention mechanisms at different levels.

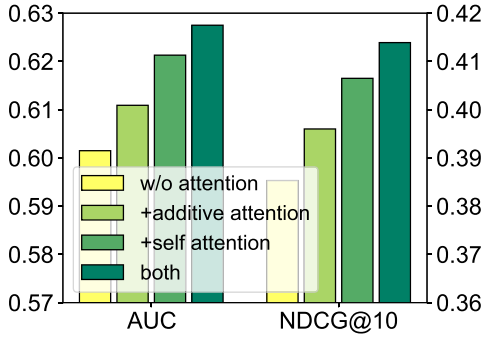


(b) Attention mechanisms of different kinds.

Figure 3: Effectiveness of different attention networks.

### 3.3 Effectiveness of Attention Mechanism

Next we explore the effectiveness of attentions in our approach. First, we verify the word- and news-level attentions. The results are shown in Fig. 3(a). We find the word-level attention is very useful. This is because modeling the interactions between words and selecting important words can help learn informative news representations. Besides, the news-level attention is also useful. This is because capturing the relatedness of news and selecting important news can benefit the learning of user representations. Moreover, combining both word- and news-level attentions can further improve the performance of our approach.

We also study the influence of additive and self-attentions on our approach. The results are shown in Fig. 3(b). From these results, we find the self-attentions are very useful. This is because the interactions between words and news are important for understanding news and modeling users. In addition, the additive attentions are also helpful. This is because different words and news may usually have different importance in representing news and users. Thus, selecting important words and news can help learn more informative news and user representations. Combining both additive and self-attention can further improve our approach. Thus, these results validate the effectiveness of the attention mechanism in our approach.

## 4 Conclusion and Future Work

In this paper we propose a neural news recommendation approach with multi-head self-attention. The core of our approach is a news encoder and a user encoder. In both encoders we apply multi-head self-attentions to learn contextual word and news representations by modeling the interactions between words and news. In addition, we use additive attentions to select important words and news to learn more informative news and user representations. Extensive experiments validate the effectiveness and efficiency of our approach.

In our future work, we will try to improve our approach in the following potential directions. First, in our framework we do not consider the positional information of words and news, but they may be useful for learning more accurate news and user representations. We will explore position encoding techniques to incorporate the word position and the time-stamps of news clicks to further enhance our approach. Second, we will explore how to effectively incorporate multiple kinds of news information in our framework, especially long sequences such as news body, which may challenge the efficiency of typical self-attention networks.

## Acknowledgments

The authors would like to thank Microsoft News for providing technical support and data in the experiments, and Jiun-Hung Chen (Microsoft News) and Ying Qiao (Microsoft News) for their support and discussions. This work was supported by the National Key Research and Development Program of China under Grant number 2018YFC1604002, the National Natural Science Foundation of China under Grant numbers U1836204, U1705261, U1636113, U1536201, and U1536207.

## References


Mingxiao An, Fangzhao Wu, Chuhan Wu, Kun Zhang, Zheng Liu, and Xing Xie. 2019. Neural news recommendation with long-and short-term user representations. In *ACL*, pages 336–345.

Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *DLRS*, pages 7–10.

Abhinandan S Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. 2007. Google news personalization: scalable online collaborative filtering. In *WWW*, pages 271–280. ACM.

Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. Deepfm: a factorization-machine based neural network for ctr prediction. In *AAAI*, pages 1725–1731. AAAI Press.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *CIKM*, pages 2333–2338. ACM.

Wouter IJntema, Frank Goossen, Flavius Frasincar, and Frederik Hogenboom. 2010. Ontology-based news recommendation. In *Proceedings of the 2010 EDBT/ICDT Workshops*, page 16.

Dhruv Khattar, Vaibhav Kumar, Vasudeva Varma, and Manish Gupta. 2018. Weave&rec: A word embedding based 3-d convolutional network for news recommendation. In *CIKM*, pages 1855–1858.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Vaibhav Kumar, Dhruv Khattar, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep neural architecture for news recommendation. In *CLEF (Working Notes)*.

Jianxun Lian, Fuzheng Zhang, Xing Xie, and Guangzhong Sun. 2018. Towards better representation learning for personalized news recommendation: a multi-channel deep fusion approach. In *IJCAI*, pages 3805–3811.

Shumpei Okura, Yukihiro Tagami, Shingo Ono, and Akira Tajima. 2017. Embedding-based news recommendation for millions of users. In *KDD*, pages 1933–1942. ACM.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Owen Phelan, Kevin McCarthy, Mike Bennett, and Barry Smyth. 2011. Terms of a feather: Content-based news recommendation and discovery using twitter. In *ECIR*, pages 448–459. Springer.

Steffen Rendle. 2012. Factorization machines with libfm. *TIST*, 3(3):57.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*, pages 5998–6008.

Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. Dkn: Deep knowledge-aware network for news recommendation. In *WWW*, pages 1835–1844.

Chuhan Wu, Fangzhao Wu, Mingxiao An, Jianqiang Huang, Yongfeng Huang, and Xing Xie. 2019a. Neural news recommendation with attentive multi-view learning. In *IJCAI*, pages 3863–3869.

Chuhan Wu, Fangzhao Wu, Mingxiao An, Jianqiang Huang, Yongfeng Huang, and Xing Xie. 2019b. Npa: Neural news recommendation with personalized attention. In *KDD*, pages 2576–2584. ACM.

Chuhan Wu, Fangzhao Wu, Mingxiao An, Yongfeng Huang, and Xing Xie. 2019c. Neural news recommendation with topic-aware news representation. In *ACL*, pages 1154–1159.

Chuhan Wu, Fangzhao Wu, Junxin Liu, Sixing Wu, Yongfeng Huang, and Xing Xie. 2018. Detecting tweets mentioning drug name and adverse drug reaction with hierarchical tweet representation and multi-head self-attention. In *SMM4H*, pages 34–37.

Qiannan Zhu, Xiaofei Zhou, Zeliang Song, Jianlong Tan, and Li Guo. 2019. Dan: Deep attention neural network for news recommendation. In *AAAI*, volume 33, pages 5973–5980.

# What Matters for Neural Cross-Lingual Named Entity Recognition: An Empirical Analysis

**Xiaolei Huang**[1*], **Jonathan May**[2], **Nanyun Peng**[2]
[1]Dept. of Information Science, University of Colorado Boulder
[2]University of Southern California Information Sciences Institute
xiaolei.huang@colorado.edu, {jonmay, npeng}@isi.edu *

## Abstract

Building named entity recognition (NER) models for languages that do not have much training data is a challenging task. While recent work has shown promising results on cross-lingual transfer from high-resource languages to low-resource languages, it is unclear what knowledge is transferred. In this paper, we first propose a simple and efficient neural architecture for cross-lingual NER. Experiments show that our model achieves competitive performance with the state-of-the-art. We further analyze how transfer learning works for cross-lingual NER on two transferable factors: sequential order and multilingual embeddings, and investigate how model performance varies across entity lengths. Finally, we conduct a case-study on a non-Latin language, Bengali, which suggests that leveraging knowledge from Wikipedia will be a promising direction to further improve the model performances. Our results can shed light on future research for improving cross-lingual NER.

## 1 Introduction

Named Entity Recognition (NER) is an important NLP task that identifies the boundary and type of named entities (e.g., person, organization, location) in texts. However, for some languages, it is hard to obtain enough labeled data to build a fully supervised learning model. Cross-lingual transfer models, which train on high-resource languages and transfer to target languages are a promising direction for languages with little annotated data (Bharadwaj et al., 2016; Tsai et al., 2016; Pan et al., 2017; Yang et al., 2017; Mayhew et al., 2017; Wang et al., 2017; Cotterell and Duh, 2017; Feng et al., 2018; Xie et al., 2018; Zhou et al., 2019).

Cross-lingual NER models learn transferable knowledge mainly from four sources: translation (Mayhew et al., 2017; Feng et al., 2018), bilingual embeddings (Ni et al., 2017; Xie et al., 2018), phonetic alphabets (Bharadwaj et al., 2016) and multi-source learning (Mayhew et al., 2017; Lin et al., 2018; Zhou et al., 2019; Chen et al., 2019; Rahimi et al., 2019). Translation can be applied at either the sentence level via machine translation or at the word and phrase level via application of bilingual dictionaries. Bilingual embeddings are a form of bilingual dictionaries; they constitute a projection from one pre-trained language representation into the same vector space as the other such that words with the same meaning have similar representations (Conneau et al., 2018). Phonetic alphabets enable different languages to share the same pronunciation characters so that the character-level knowledge is transferable between languages that otherwise have different character sets, such as English and Bengali (Hermjakob et al., 2018). Multi-source learning is effective when multiple or similar language resources are available by learning shareable knowledge. For example, training a model on both English and Hindi can significantly improve the model performance on Bengali than only using English (Mayhew et al., 2017). However, there is little prior work with detailed analysis of how cross-lingual NER models transfer knowledge between languages on different levels.

In this paper, we focus on a single-source zero-shot transfer setting where we transfer from English to target languages that have no annotated data. In our settings, the resources are limited to annotated source language data, bilingual dictionaries, and unlabeled corpora from both source and target languages. We first propose a neural cross-lingual NER model that combines the ideas of translation, bilingual embedding, and phonetic

*The work was done when the first author worked as an intern at USC ISI.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6395–6401, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

alphabets. Next, we conduct qualitative analyses to answer the following questions on how the model transfers knowledge under the cross-lingual settings: 1) does the source language syntax matter? 2) how do word and character embeddings affect the model transfer? We analyze how F1 scores differ across different entity lengths. Finally, we conduct a case study on Bengali.

## 2 Model

NER models take a sequence of tokens as input and predict a sequence of labels such as person (PER) or location (LOC). In this paper, we adopt the neural architecture NER model from Lample et al. (2016); Ma and Hovy (2016). The model first combines pre-trained word and character embeddings as token representations, then feeds the representations to one layer of a Bidirectional Long Term Short Memory (Bi-LSTM) (Hochreiter and Schmidhuber, 1997), and finally predicts the outputs via a Conditional Random Field (CRF). We show the model architecture on left of Figure 1. However, languages express the same named entity in different words and characters. To bridge the barriers, we combine three strategies: bilingual embedding, reverse translation, and transliteration.

**Bilingual Embedding.** Word embeddings are usually trained for each language separately and therefore in different vector spaces (Ruder et al., 2019). To map word embeddings into the same shared space, we use MUSE (Conneau et al., 2018) to build our bilingual embeddings. The bilingual embeddings enable the words with the same meaning to have similar word representations in the shared space.

**Reverse Translation.** We use the bilingual dictionaries provided by Rolston and Kirchhoff (2016). Language variations exist among source and target languages and training on monolingual corpus might limit learning the variations. Therefore, we translate the source language, English, to the target language for the training corpus to reconstruct source language sentences and learn source and target languages jointly. However, one English word might have multiple corresponding translations. To select the best translation, we define an empirical score function $F(w, w_{t,i})$ that calculates the cosine similarity between the translated target word ($w_{t,i}$) and the English source word ($w$) based on its contextual words ($w_{c,j}$).

$$F(w, w_{t,i}) = \alpha \cdot \cos(E(w), E(w_{t,i})) + (1-\alpha) \cdot \sum_{j=1}^{m} \frac{\cos(E(w_{t,i}), E(w_{c,j}))}{(d_j+1)^2} \quad (1)$$

where $E(w)$ is the bilingual embedding vector of the word, $d_j$ is the sequential distance between the word and its contextual word j, and $\alpha$ is the trade-off factor that balances impacts of translation pair and contextual words. In this study, we set $\alpha$ to 0.5. We choose the translation pair with the highest similarity score. Note that the reverse only applies during the training step that translates English into the target language.

| Target word | Source word | Uroman | Type |
|---|---|---|---|
| grünen | green | gruenen | ORG |
| ইউরোপের | Europe | iuropera | LOC |

Table 1: Examples of Uroman that maps different languages into the same character space. We show two target languages, German (top) and Bengali (bottom). The second column is a translation of the target language. The transliterations show the phonetic similarity between the source and target languages.

**Character - Uromanization.** Different languages may not share the same characters, but some named entities in the multilingual corpora share phonetic similarities (Huang et al., 2004). To map multilingual corpora into the same character space and connect the phonetic similarity between entities, we employ Uroman[1] (Hermjakob et al., 2018), which transliterates any language into something roughly pronounceable English spellings, while keeps the English words unchanged. We show some examples of Uroman in Table 1.

## 3 Experiments

In this study, we first evaluate our proposed cross-lingual method on the CoNLL 2002 and 2003 NER datasets (Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003). We then conduct ablation studies to examine how the model learns transferable knowledge under the cross-lingual settings. Finally, we conduct a case study on Bengali, a low resource language.

[1] https://www.isi.edu/~ulf/uroman.html

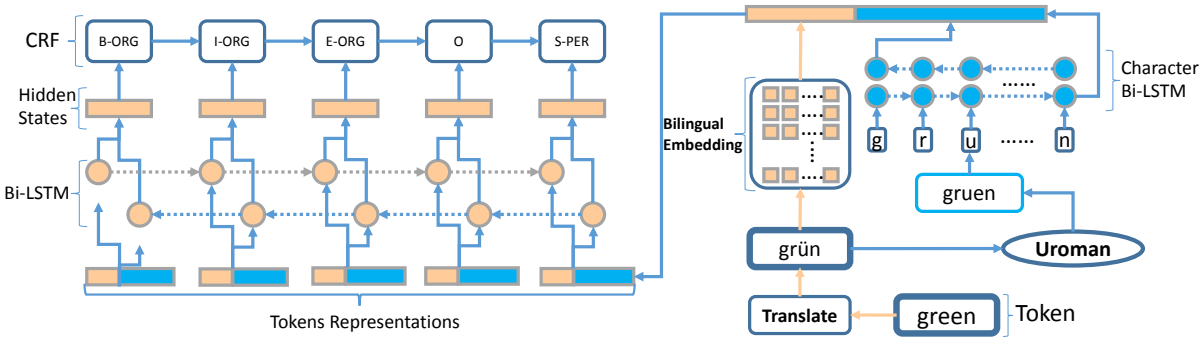


Figure 1: Architecture of our proposed model. Each word is translated from the source. Each token representation contains two concatenated parts: a bilingual word embedding (light orange) and character representations (blue). The character representation for each word is generated via the Bi-LSTM. Words are first transliterated and split into individual characters before being fed into the character Bi-LSTM.

### 3.1 Data

The CoNLL datasets contain four different European languages, Spanish, Dutch, English, and German. The data contains four types of named entities: person (PER), organization (ORG), location (LOC) and MISC. We use Bengali language data from LDC2015E13 (V2.1). To be consistent, we only keep the PER, ORG and LOC tags and ignore MISC tags from the Bengali corpus.

In this study, we choose the BIOSE tag schema instead of standard BIO, where the B, I, O, S, E refer to the beginning, inside, outside, single and end of an entity, respectively. Previous work shows that BIOSE can learn a more expressive model than the BIO schema does (Ratinov and Roth, 2009). We lowercase the data and replace numbers and URLs as "num" and "url" respectively. We use the training data to train our model, the development sets to select the best well-trained model, and the test sets to evaluate performance, where the training data is English and the development and test sets are from the target language.

### 3.2 Experimental Settings

**Bilingual Embedding**. We use the 300-dimensional bilingual word embeddings pre-trained by MUSE (Conneau et al., 2018) and 300-dimensional randomly initialized character embeddings. Specifically, we first collect monolingual pre-trained word embeddings from fastText (Bojanowski et al., 2017). We then align the embeddings using the MUSE supervised model. We normalize the inputs to MUSE and keep the other hyperparameters as defaults. We merge the aligned word embeddings and remove duplicates. Updating the embedding during training time will change the vector space of partial bilingual embeddings and therefore break the vector alignment. Thus, we freeze the embedding weights during the training step. For OOV (out-of-vocabulary) words, we use a randomly initialized 300 dimension vector within $[-0.1, 0.1]$.

**Reverse Translation**. We follow the general translation step in Section 2. Specifically, we replace English words with target language words if the translation pairs exist in the bilingual dictionaries (Rolston and Kirchhoff, 2016). If a translation pair does not exist for a word, we keep the word unchanged in the training data. We use the pre-trained bilingual embedding to help select the best choice of polysemy.

**Model parameters**. We use 300-dimension hidden states for both character and token level Bi-LSTMs. To prevent overfitting, we apply dropout (Srivastava et al., 2014) with a rate of 0.5 on outputs of the two Bi-LSTMs. We follow the Conditional Random Field (CRF) setup of Peng and Dredze (2015). We then randomly initialize the weights of the layers within $[-0.1, 0.1]$. We train the model for 200 epochs and optimize the parameters by Stochastic Gradient Descent (SGD) with momentum, gradient clipping, and learning rate decay. We set the learning rate (lr) and the decay rate (dr) as 0.01 and 0.05 respectively. We update the learning rate by $\frac{lr}{(1+(n+1)*dr)}$ after epoch $n$. We clip the gradients to the range of [-5.0, 5.0]. We measure performance by F1 score.

### 3.3 Baselines

In this study, we compare our proposed method with three close works under the cross-lingual settings. We compare our method with the best-reported performance from their works. We briefly

summarize the the three baselines in this section.

- WikiNER (Tsai et al., 2016) first links named entities in the multilingual Wikipedia corpora and extracts page categories as "wikifier" features, and use these features to achieve cross-lingual transfer.
- CTNER (Mayhew et al., 2017) first translates the corpora into English via bilingual dictionaries and multilingual Wikipedia entries on both word and phrase levels, and directly perform NER on the translated target language.
- NCNER (Xie et al., 2018) proposes a neural NER model with attention mechanism. The work is closest to ours. However, our model takes different approaches in obtaining multilingual embedding, translation, and transliteration as Xie et al. (2018), while the neural models share the similar architecture of Bi-LSTMs-CRF, and use multi-lingual embeddings to achieve transfer.

### 3.4 Results

| Models | Spanish | Dutch | German |
|---|---|---|---|
| WikiNER* | 60.55 | 61.60 | 48.10 |
| CTNER* | 51.82 | 53.94 | 50.96 |
| CTNER*+ | 65.95 | 66.50 | 59.11 |
| NCNER | **72.37** | 71.25 | 57.76 |
| Our Model | 64.48 | **73.44** | **62.26** |

Table 2: F1 score comparisons of cross-lingual models on Spanish, Dutch and German. The "*" indicates the model uses the Wikipedia resources. The "+" means training model by multiple language resources.

As seen in Table 2, our proposed method outperforms the previous works in nearly all cross-lingual tasks. Moreover, the two neural model methods exceed the performance of the other models. These results suggest the effectiveness of neural cross-lingual transfer through multi-lingual embedding space, and show competitive results of our proposed methods.

### 3.5 Ablation Studies

We conduct ablation analyses to understand how the model transfers learned knowledge from one language to the other. We focus on two aspects: syntax and embeddings.

**Syntax analysis.** Different languages might not share the same syntax structures. The neural model learns sequential information from the source language and applies the learned knowledge on the target language. The importance of this sequential information is unknown. We shuffle the sentences of training data while keeping the internal order of named entities unchanged. We show the results in Table 3. The results show that after shuffling, the performances of our model decrease and the decreases vary among different languages. This suggests the importance of sequential information for those languages under the cross-lingual settings.

| Models | Spanish | Dutch | German |
|---|---|---|---|
| Full Model | 64.48 | 73.44 | 62.26 |
| Shuffle | 49.44 | 40.61 | 27.25 |
| Word-only | 53.00 | 59.60 | 52.58 |
| Char-only | 18.88 | 12.15 | 16.46 |

Table 3: F1 scores of different ablation analyses, compared to our full model.. "Shuffle" means the training data is shuffled. "Word-only" means our proposed model is only fed bilingual word embeddings, and "Char-only" means the model only receives character embeddings as input.

**Embeddings analysis.** We train our proposed models with both character and bilingual word embeddings, however, how the model values different embeddings is unknown. We feed the model with either bilingual word or character embeddings. Table 3 shows that bilingual word embeddings have better performance than character embeddings across the three languages. The results suggest that the aligned bilingual embeddings are more important than the character embeddings for the three languages.

### 3.6 Entity length analysis

While we can observe how models perform overall on the named entities, we can not know how models differ from short to long entities. Particularly, probing if models hold strong biases towards shorter entities can help interpret the process of cross-lingual transfer learning. To make the comparison, we first categorize entities into three levels: single token, two tokens and greater or equal to three tokens. We then calculate the F1-score of the three entity lengths in the test set and among correctly predicted entities. Finally we summarize the F1-score in Table 4.

With comparing to the overall performance across the three languages in Table 2, we can observe that the single token shows relatively closer

| Language | 1 | 2 | ≥3 |
|---|---|---|---|
| Spanish | 68.93 | 68.21 | 47.57 |
| Dutch | 69.49 | 82.46 | 60.85 |
| German | 59.50 | 70.76 | 37.22 |

Table 4: F1 scores of different lengths of entities across the three languages: Spanish, Dutch and German. The number 1 refers to the entities with single token, the 2 means the entities with two tokens and the ≥ 3 indicates the entities with not less than three tokens.

to the performance, the entities with two tokens achieve the higher scores, while the entities with more than 2 tokens decrease significantly ranging from 12.59 to 25.04 absolute percentages of F1 scores. The observation indicates that entities longer than two tokens are more difficult to infer. This might encourage us to balance the weight of long entities in our current evaluation method which ignores entity length when datasets have high volumes of long entities.

## 4 Case Study: Bengali

| Models | F1 score |
|---|---|
| CTNER | 30.47 |
| CTNER+ | 31.70 |
| CTNER* | 46.28 |
| CTNER*+ | 45.70 |
| Our Model | 34.29 |

Table 5: F1 score comparisons of translation-based models on Bengali. The "*" indicates the model uses Wikipedia resources. The "+" means a model is trained with multiple language resources.

The previous cross-lingual settings were only for European languages, which share similar alphabets. However, many languages use non-Latin orthography. In this work, we present a case study on Bengali, which does not use a Latin alphabet. We compare our proposed method with the translation-based method, CTNER (Mayhew et al., 2017). The results in Table 5 show that our model outperforms the previous methods without Wikipedia.

The results suggest multilingual Wikipedia is critical for future performance improvements beyond simple transfer. This is to be expected; a domain discrepancy exists between the source and target language data and therefore many named entities are missing or mismatched: partial data sources of the Bengali come from social media and online forums (Cieri et al., 2016). By contrast, the only data source of CoNLL data is from news articles (Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003). While transfer can help provide universal context clue information across languages there is no substitute for a resource of actual names.

| Language | Type | Token |
|---|---|---|
| Spanish | 2.5 | 0.8 |
| Dutch | 2.5 | 0.9 |
| German | 2.3 | 1.5 |
| Bengali | 18.2 | 12.2 |

Table 6: OOV rate (percentage) in our bilingual word embeddings across the four languages. Type indicates unique words, and token refers to counting token numbers.

The colloquium words from social media may cause the issue of out of vocabulary (*OOV*) and further impact the transferring process. We summarize the percentage of missing words in our pretrained bilingual word embeddings in Table 6. The OOV rate of Bengali is significantly higher than that of the other three languages. This suggests that the high OOV rate and the discrepancy of data sources may hurt the effectiveness of transfer.

## 5 Conclusion

We have presented a simple but efficient method to adapt neural NER for the cross-lingual settings. The proposed method benefits from multiple transferable knowledge and shows competitive performances with the state of the art using limited resources. We examine multiple factors that impact the transfer process and conduct an ablation study to measure their influences. Our experiment on Bengali shows that leveraging knowledge from Wikipedia will be a promising direction for future research.

## 6 Acknowledgments

This work is partially funded by DARPA (HR0011-15-C-0115) and NIH R01 (LM012592). The authors thank the anonymous reviewers for their helpful suggestions. We thank Boliang Zhang, Stephen Mayhew, Michael J. Paul and Thamme Gowda for their useful feedback. Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the view of the sponsor.

## References


Akash Bharadwaj, David Mortensen, Chris Dyer, and Jaime Carbonell. 2016. Phonologically aware neural model for named entity recognition in low resource transfer settings. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1462–1472, Austin, Texas. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Xilun Chen, Ahmed Hassan Awadallah, Hany Hassan, Wei Wang, and Claire Cardie. 2019. Multi-source cross-lingual model transfer: Learning what to share. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3098–3112, Florence, Italy. Association for Computational Linguistics.

Christopher Cieri, Mike Maxwell, Stephanie Strassel, and Jennifer Tracey. 2016. Selection criteria for low resource language programs. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 4543–4549, Portorož, Slovenia. European Language Resources Association (ELRA).

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. In *International Conference on Learning Representations*.

Ryan Cotterell and Kevin Duh. 2017. Low-resource named entity recognition with cross-lingual, character-level neural conditional random fields. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 91–96, Taipei, China. Asian Federation of Natural Language Processing.

Xiaocheng Feng, Xiachong Feng, Bing Qin, Zhangyin Feng, and Ting Liu. 2018. Improving low resource named entity recognition using cross-lingual knowledge transfer. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 4071–4077, Stockholm, Sweden. International Joint Conferences on Artificial Intelligence Organization.

Ulf Hermjakob, Jonathan May, and Kevin Knight. 2018. Out-of-the-box universal romanization tool uroman. *Proceedings of ACL 2018, System Demonstrations*, pages 13–18.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Fei Huang, Stephan Vogel, and Alex Waibel. 2004. Improving named entity translation combining phonetic and semantic similarities. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT*, pages 260–270, San Diego, California. Association for Computational Linguistics.

Ying Lin, Shengqi Yang, Veselin Stoyanov, and Heng Ji. 2018. A multi-lingual multi-task architecture for low-resource sequence labeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 799–809, Melbourne, Australia. Association for Computational Linguistics.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Stephen Mayhew, Chen-Tse Tsai, and Dan Roth. 2017. Cheap translation for cross-lingual named entity recognition. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2536–2545, Copenhagen, Denmark. Association for Computational Linguistics.

Jian Ni, Georgiana Dinu, and Radu Florian. 2017. Weakly supervised cross-lingual named entity recognition via effective annotation and representation projection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1470–1480, Vancouver, Canada. Association for Computational Linguistics.

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017. Cross-lingual name tagging and linking for 282 languages. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1946–1958, Vancouver, Canada. Association for Computational Linguistics.

Nanyun Peng and Mark Dredze. 2015. Named entity recognition for chinese social media with jointly trained embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 548–554, Lisbon, Portugal. Association for Computational Linguistics.

Afshin Rahimi, Yuan Li, and Trevor Cohn. 2019. Massively multilingual transfer for ner. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 151–164, Florence, Italy. Association for Computational Linguistics.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155, Boulder, Colorado. Association for Computational Linguistics, Association for Computational Linguistics.

Leanne Rolston and Katrin Kirchhoff. 2016. Collection of bilingual data for lexicon transfer learning. Technical report, Technical Report UW-EE-2016-0001.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2019. A survey of cross-lingual word embedding models. *Journal of Artificial Intelligence Research*, 65:569–631.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Erik F. Tjong Kim Sang. 2002. Introduction to the conll-2002 shared task: Language-independent named entity recognition. In *Proceedings of the 6th Conference on Natural Language Learning - Volume 20*, COLING-02, pages 1–4, Stroudsburg, PA, USA. Association for Computational Linguistics.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147, Edmonton, Canada. Association for Computational Linguistics.

Chen-Tse Tsai, Stephen Mayhew, and Dan Roth. 2016. Cross-lingual named entity recognition via wikification. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 219–228, Berlin, Germany. Association for Computational Linguistics.

Dingquan Wang, Nanyun Peng, and Kevin Duh. 2017. A multi-task learning approach to adapting bilingual word embeddings for cross-lingual named entity recognition. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 383–388, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Jiateng Xie, Zhilin Yang, Graham Neubig, Noah A. Smith, and Jaime Carbonell. 2018. Neural cross-lingual named entity recognition with minimal resources. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 369–379, Brussels, Belgium. Association for Computational Linguistics.

Zhilin Yang, Ruslan Salakhutdinov, and William W Cohen. 2017. Transfer learning for sequence tagging with hierarchical recurrent networks. In *International Conference on Learning Representations*, Palais des Congrès Neptune, Toulon, France.

Joey Tianyi Zhou, Hao Zhang, Di Jin, Hongyuan Zhu, Meng Fang, Rick Siow Mong Goh, and Kenneth Kwok. 2019. Dual adversarial neural transfer for low-resource named entity recognition. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 3461–3471, Florence, Italy. Association for Computational Linguistics.

# Telling the Whole Story: A Manually Annotated Chinese Dataset for the Analysis of Humor in Jokes

**Dongyu Zhang**[1], **Heting Zhang**[1], **Xikai Liu**[2], **Hongfei Lin**[2], **Feng Xia**[1*]
[1]School of Software, Dalian University of Technology, 116620, China
[2]School of Computer Science and Technology, Dalian University of Technology, 116024, China
f.xia@ieee.org
hflin@dlut.edu.cn

## Abstract

Humor plays important role in human communication, which makes it important problem for natural language processing. Prior work on the analysis of humor focuses on whether text is humorous or not, or the degree of funniness, but this is insufficient to explain why it is funny. We therefore create a dataset on humor with 9,123 manually annotated jokes in Chinese. We propose a novel annotation scheme to give scenarios of how humor arises in text. Specifically, our annotations of linguistic humor not only contain the degree of funniness, like previous work, but they also contain key words that trigger humor as well as character relationship, scene, and humor categories. We report reasonable agreement between annotators. We also conduct an analysis and exploration of the dataset. To the best of our knowledge, we are the first to approach humor annotation for exploring the underlying mechanism of the use of humor, which may contribute to a significantly deeper analysis of humor. We also contribute with a scarce and valuable dataset, which we will release publicly.

## 1 Introduction

Humor plays important role in human communication, which not only serves to exchange ideas or convey messages, but also involves emotion regulation such as provoking laughter, generating amusement, and reducing stress (Wooten, 1996; Morse, 2007). In particular, with the rapid growth of social media applications such as Facebook and Twitter, a significantly increasing number of individuals are using these social media public platforms to release humorous texts. Humor often arises when two incongruous concepts are applied and examined through one semantic frame (Lefcourt, 2001; Paulos, 2008). The two concepts often involve semantic disconnection in forms such as contradiction and contrast/comparison. Humor sometimes occurs due to ambiguity (Yang et al., 2015), such as unexpected homophones/homographs.

The importance and complexity of humor has thus gained attention in natural language processing (NLP), and many computational approaches to it have been proposed (Binsted et al., 2006; Yang et al., 2015; Baziotis et al., 2017; Ortega-Bueno et al., 2018; Liu et al., 2018). Corpora are fundamental in NLP for sound analysis of humor and for high-quality automatic humor detection. Scholars have been devoted to the study of the humor resources in both English and other languages. Mihalcea and Strapparava (2005) constructed a humorous witticism dataset with 16,000 text data for humor identification in English sentences. The dataset comes from one-liners, reuters titles, BNC sentences, and proverbs, and was annotated with humor and non-humor. Reyes et al. (2013) established an English irony dataset of 40,000 tweets for conducting the study of irony on tweets. The dataset contains the label of irony and other specific hashtags of non-irony (education, humor, and politics). Zhang and Liu (2014) established an English humor corpus with 3,000 tweets to recognize humor on Twitter. The dataset contains the annotation of humorous tweets, non-humorous tweets and humorous non-tweets. Potash et al. (2017) built a 12,734 tweets dataset of English for studying the comparative ranking of humor. The dataset comes from the midnight TV program called Hashtag Wars which published on Twitter. Castro et al. (2017) established a humorous text corpus containing 33,531 tweets for detecting humor in Spanish Tweets. The dataset involves humorous annotation and humor level annotation. The humor level annotation is based on a 5-point scale, 1 signifying the lowest level and 5 signifying the highest level. Castro et al. (2018) revised



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6402–6407, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

the Spanish Twitter corpus with crowd notes and presented a 27,000 tweets dataset in total. Specially, the authors used 5 different emojis to represent the 5 degrees of humor instead of using the 5-point annotation.

However, while previous work focuses on textual humor annotation of humorous/non-humorous and degree of funniness, such annotations do not provide adequate knowledge and scenarios to explain how humor arises, so they may not provide a deep analysis of the underlying mechanism of humor. In addition, as the majority of data came from Twitter, the data source lacks variety. To this end, we create a dataset on humor with 9,123 manually annotated jokes in Chinese. We propose a novel scheme with annotations of key words that make text humorous as well as character relationship, scene, humor category, and degree of funniness. The annotation agreement analyses for multiple annotators are described. We also conduct analysis and exploration on the dataset. Our contributions are as follows.

• We propose a novel annotation scheme to explain how humor arises in text. Unlike previous work, we annotate not only what is humorous, but also what causes humor.

• We contribute to a new, sizeable, and scarce joke dataset, which is being released publicly and particularly valuable in languages other than English.

## 2 Data Collection

To make the dataset objective and comprehensive, we collected joke data involving both diachronic and synchronic relationships simultaneously from a variety of fields. Also, We selected jokes based on a four-dimensional model. On the time axis, our dataset includes jokes from books, literary journals, etc. published over the past decade, which satisfies the diachronic requirement. It also includes jokes posted on websites and microblogs, many of which are novel, which conforms to the synchronic requirement. On the spatial axis, the dataset contains both domestic and translated foreign jokes. On the subject axis, the perception of intensity of jokes also varies from person to person, due to their varying backgrounds and senses of humor. On the style axis, jokes from books have various themes, and they are relatively canonical, while online jokes seem more oral and informal. The source information is in Table 1.

| Sources | Words | Sentences | Jokes |
|---|---|---|---|
| Websites | 2,397,816 | 23,508 | 5,463 |
| Micro-blogs | 1,207,856 | 11,614 | 2,581 |
| Books,Journals | 504,920 | 4,855 | 1,079 |
| Total | 4,110,592 | 39,977 | 9,123 |

Table 1: Information on data sources.

## 3 Annotation Scheme

### 3.1 Annotation model

The annotation model is as follows:

JokeModel = (Relationship, Scene, Category, HumorLevel, Keyword, DataSource )

• Relationship: We annotated the mutual relationship between the main characters such as teacher-student, doctor-patient, lovers, superior and subordinate, etc. because accessing the relationship between people in jokes is helpful for a clearer understanding of the contextual coordinates on the joke (Popa, 2005).

• Scene: The scene refers to the place where the joke occurs. Previous studies indicate humor plays an important role in the places including campus (Morrison et al., 2012), workplace (Blumenfeld and Alpern, 1994), family (Lovorn, 2008) and public space (Thornton, 2007). We therefore selected the campus, workplace, family and public space for the annotation of scene in humor.

• Category: There is no consensus on the category of humor in the literature. Based on our investigation of a wide range of literature, we focused on eight main types of the most frequently appearing humor including homophonic, harmonic, antiphrasis, analogy, euphemism, irony, exaggeration, and reversal.

• Level: We weighed the fine-grained annotation of humor (Bressler and Balshine, 2006; Castro et al., 2018; Deckers and Devine, 1981) and presented a 5 point-scale for humor rating, 1 signifying the non-humor, 2-5 signifying the gradually increasing degree of humor. (Gan, 2015; Stein, 1998; Tsakona, 2009)

• DataSource: Table 1 presents the source of jokes.

• Keyword: We define the key words as words that trigger humor and that may have conflicting, incongruous and ambiguous meanings in jokes (Van Hee et al., 2016). Van Hee et al. (2016) proposed the annotation item based on the text spans of contrasting which contains the type of explicit

Example1:
<Humor>
<ID>H02</ID>
<Contents>妻子：每次我唱歌的时候，你为什么总要到阳台上去?
"Wife: When I sing, why do you always go to the balcony?"
丈夫：我想让大家都知道，不是我在打你
"Husband: Because I want to let everyone know that I am not hitting you."
<Contents>
<Relationship>丈夫 "husband"/妻子 "wife"</Relationship>
<Scene>家庭 "family"</Scene>
<Category>讽刺 "irony"</Category>
<HumorLevel>3</HumorLevel>
<Keywords>唱歌 "sing"/打 "hit"</Keywords>
<DataSource>N</DataSource>
</Humor>

Figure 1: An example of annotation

and implicit. The special word pair in the contrasting text spans triggers the production of humor. Specially, we annotated the keyword based on the thought of contrasting text spans in humor in the format of prototype. An annotation example shows in figure 1.

### 3.2 Keyword annotation

The keyword is the most challenging of the six annotating items. Following Van Hee et al. (2016), our annotation of key words is at the relation level, which involves the identification of incongruous or ambiguous vocabulary, resulting in a comic effect. To discriminate key words, the annotators followed the below guidelines:

• Read the entire text-discourse to establish a general understanding of the meaning.

• For each word in the text, establish its meaning in context.

• Determine which words have the meaning of incongruous/conflicting/ambiguous/unexpected or strong emotions that make text humorous in the given context.

• Decide whether the contextual meaning can be understood.

• If yes, mark the word as a key word.

Figure 2 shows the example of keyword annotation in humor.

The words "sing" and "hit" act as keywords, because the two phrases are the main indicators of humor: "she sings badly, and it sounds like she

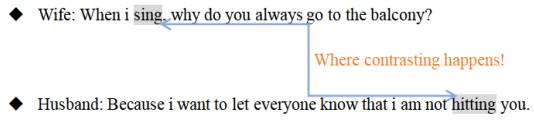


Figure 2: An example of keyword annotation

is being beaten and screaming." The comparison of "sing" and "hit" invokes the humor, so they are keywords.

### 3.3 Annotation process

Eight postgraduate students and one PhD student worked together to complete the annotation of the joke dataset. The participants were divided into four groups of two. Each group annotated the jokes using cross-validation. The PhD student arbitrated. During the annotation process, when two people reached agreement on the annotation result, then the marking was complete; when there was disagreement, the arbitrator attempted to resolve it. When the arbitration was inconsistent with the views of the two persons' judgment: Case 1. If the inconsistency was in the degree of humor, we used the average value of the three people. Case 2. If there was any disagreement about the generation mechanism, it was discussed by the whole group of nine people, and the mechanism receiving the largest number of votes was the final result.

### 3.4 Annotation agreement and challenges

To evaluate inter-annotator agreement, we let three annotators annotate the same 600 sentences to assess inter-annotator agreement. We used Fleiss' s kappa (Fleiss, 1971). The agreement on the relationship annotation was $\kappa$ = 0.85; the agreement on the scene annotation was $\kappa$ = 0.79; the agree-

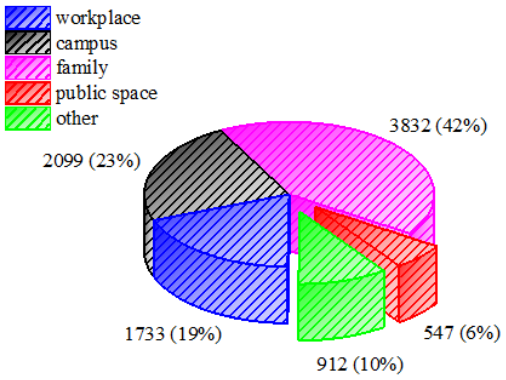


Figure 3: Quantity of joke scenes

ment on the category annotation was $\kappa$ = 0.71, the

agreement on the humor level annotation was $\kappa$ = 0.65, and the agreement on the keywords annotation was $\kappa$ = 0.59.

The keywords and humor level annotation was the most challenging part of the annotation, due to the subjective nature of cognition and the different background knowledge of people. To minimize the problems annotators faced, we held a seminar once a week to discuss the ambiguities. Then, the guide gave an authoritative explanation. Finally, ambiguity points and measures were added into the annotating guide manual to help annotators to make judgments quickly and correctly when they encountered the same problem.

## 4 Dataset Analysis

The dataset contains 9,123 jokes, 39,977 sentences and 4,110,592 words in total, with an average of 4.38 sentences per joke. In the dataset, there are five scenes of joke (where the joke occurs): workplace, campus, family, public space and others. Family accounts for 42% of the dataset. This is perhaps because family life accounts for the largest proportion of life as a whole. These statistical data fully confirm that the jokes originate from life, and that they are known to the general public, which shows that the audience for jokes is very wide. The specific distribution is in Figure 3.

Figure 4 shows the vocabulary that appears

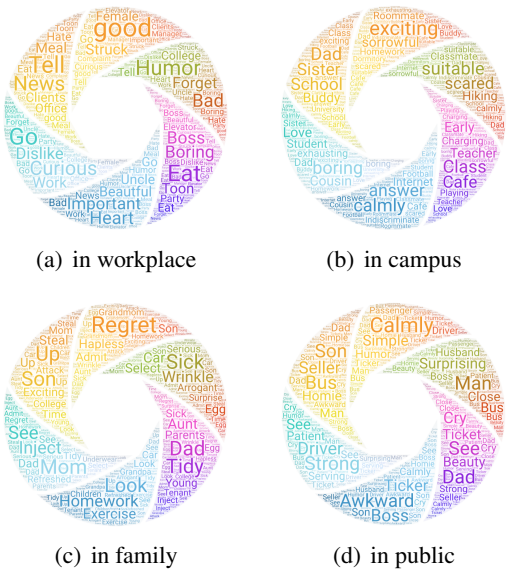


(a) in workplace (b) in campus

(c) in family (d) in public

Figure 4: Word cloud in each scene (translated from Chinese)

most frequently in each scene of joke. As Figure 4 shows, the high-frequency words for each

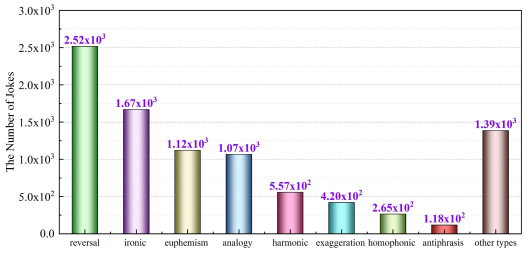


Figure 5: Humor categories

scene of jokes are various. For instance, the top five high-frequency words in campus jokes are "school, class, exciting, suitable, answer"; in family jokes are "son, mom, regret, homework, see"; in workplace jokes are "boring, boss, curious, forget, tell"; in public space jokes are "man, bus, surprising, cry, calmly". It is intriguing that some high-frequency words in certain jokes are related to certain scenes.

For instance, the high-frequency words in workplace jokes are "boss, boring, forget, which not only is in line with the bias of the work place, but also proves the validity of this classification to some extent.

We also analyzed the humor categories because they may associate with underlying mechanism of the use of humor. The quantitative statistics for humor categories are shown in Figure 5.

Our annotation not only contains the degree of funniness, but also key words that trigger humor, as well as character relationship, scene, and humor categories. Specially, we have improved on the study of Castro et al. (2018) by providing evidence of what causes humor and explaining how humor arises in text. Furthermore, our data have come from a range of sources in numerous domains rather than only from Twitter.

## 5 Conclusion

We propose a novel annotation scheme to explain how humor arises in text. Unlike previous work, we annotate not only what is humorous, but also what causes humor. Our dataset creation involved nine volunteer students for 8 months. We will release the dataset publicly. With 9,123 Chinese jokes and 39,977 sentences in total, and with fine-grained annotation of humor, the dataset provides a new, sizeable, and scarce joke dataset, which is particularly valuable in languages other than English for scholars in many disciplines, such as computational, linguistic, and cognitive studies.

## Acknowledgments

This research was supported by NSFC Programs (No. 61602079; No.61632011); Ministry of Education (No.16YJCZH141).

## References

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017. Datastories at semeval-2017 task 6: Siamese lstm with attention for humorous text comparison. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 390–395.

Kim Binsted, Anton Nijholt, Oliviero Stock, Carlo Strapparava, G Ritchie, R Manurung, H Pain, Annalu Waller, and D O'Mara. 2006. Computational humor. *IEEE Intelligent Systems*, 21(2):59–69.

Esther Blumenfeld and Lynne Alpern. 1994. *Humor at work: The guaranteed, bottom-line, low-cost, high-efficiency guide to success through humor*. Peachtree Publishers.

Eric R Bressler and Sigal Balshine. 2006. The influence of humor on desirability. *Evolution and Human Behavior*, 27(1):29–39.

Santiago Castro, Luis Chiruzzo, Aiala Rosá, Diego Garat, and Guillermo Moncecchi. 2018. A crowd-annotated spanish corpus for humor analysis. In *Proceedings of the Sixth International Workshop on Natural Language Processing for Social Media*, pages 7–11.

Santiago Castro, Matłas Cubero, Diego Garat, and Guillermo Moncecchi. 2017. Is this a joke? detecting humor in spanish tweets. *Inteligencia Artificial*.

Lambert Deckers and John Devine. 1981. Humor by violating an existing expectancy. *The Journal of Psychology*, 108(1):107–110.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

Xiaoli Gan. 2015. A study of the humor aspect of english puns: Views from the relevance theory. *Theory and Practice in Language Studies*, 5(6):1211–1215.

Herbert M Lefcourt. 2001. *Humor: The psychology of living buoyantly*. Springer Science & Business Media.

Lizhen Liu, Donghai Zhang, and Wei Song. 2018. Modeling sentiment association in discourse for humor recognition. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 586–591.

M G Lovorn. 2008. Humor in the home and in the classroom: The benefits of laughing while we learn. *Journal of Education and Human Development*, 2(1):1–12.

Rada Mihalcea and Carlo Strapparava. 2005. Making computers laugh: Investigations in automatic humor recognition. In *Conference on Human Language Technology & Empirical Methods in Natural Language Processing*, pages 531–538.

Mary Kay Morrison et al. 2012. *Using humor to maximize living: Connecting with humor*. R&L Education.

D. R. Morse. 2007. Use of humor to reduce stress and pain and enhance healing in the dental setting. *J N J Dent Assoc*, 78(4):32–36.

Reynier Ortega-Bueno, Carlos E Muniz-Cuza, José E Medina Pagola, and Paolo Rosso. 2018. Uo upv: Deep linguistic humor detection in spanish social media. In *Proceedings of the Third Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval 2018) co-located with 34th Conference of the Spanish Society for Natural Language Processing (SEPLN 2018)*, pages 204–213.

John Allen Paulos. 2008. *Mathematics and humor: A study of the logic of humor*. University of Chicago Press.

Diana-Elena Popa. 2005. Jokes and translation. *Perspectives*: *Studies in Translatology*, 13(1):48–57.

Peter Potash, Alexey Romanov, and Anna Rumshisky. 2017. Semeval-2017 task 6:# hashtagwars: Learning a sense of humor. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 49–57.

Antonio Reyes, Paolo Rosso, and Tony Veale. 2013. A multidimensional approach for detecting irony in twitter. *Language Resources & Evaluation*, 47(1):239–268.

Howard F Stein. 1998. Organizational euphemism and the cultural mystification of evil. *Administrative Theory & Praxis*, pages 346–357.

Kendell C Thornton. 2007. Relationship closeness and embarrassment. *Individual Differences Research*, 5(1).

Villy Tsakona. 2009. Language and image interaction in cartoons: Towards a multimodal theory of humor. *Journal of Pragmatics*, 41(6):1171–1188.

Cynthia Van Hee, Els Lefever, and Véronique Hoste. 2016. Exploring the realization of irony in twitter data. In *LREC*.

P Wooten. 1996. Humor: an antidote for stress. *Holistic Nursing Practice*, 10(2):49–56.

Diyi Yang, Alon Lavie, Chris Dyer, and Eduard Hovy. 2015. Humor recognition and humor anchor extraction. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2367–2376.

Renxian Zhang and Naishi Liu. 2014. Recognizing humor on twitter. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 889–898. ACM.

# Generating Natural Anagrams: Towards Language Generation Under Hard Combinatorial Constraints

**Masaaki Nishino**[1] **Sho Takase**[2*] **Tsutomu Hirao**[1] **Masaaki Nagata**[1]
[1] NTT Communication Science Laboratories, NTT Corporation
{masaaki.nishino.uh,tsutomu.hirao.kp,masaaki.nagata.et}@hco.ntt.co.jp
[2] Tokyo Institute of Technology
sho.takase@nlp.c.titech.ac.jp

## Abstract

An anagram is a sentence or a phrase that is made by permutating the characters of an input sentence or a phrase. For example, "Trims cash" is an anagram of "Christmas". Existing automatic anagram generation methods can find possible combinations of words form an anagram. However, they do not pay much attention to the naturalness of the generated anagrams. In this paper, we show that simple depth-first search can yield natural anagrams when it is combined with modern neural language models. Human evaluation results show that the proposed method can generate significantly more natural anagrams than baseline methods.

## 1 Introduction

Making an *anagram* is one of the most popular wordplay games and has a long history that can be traced back to the time of the Ancient Greeks (Wheatley, 1862). An anagram is a sentence or a phrase that is made as a permutation of characters of an input sentence or phrase. For example, "Trims cash" is an anagram of "Christmas" since the former can be made by permutating the characters contained in the latter.

The automatic generation of anagrams is a challenging task since it is NP-hard[1]. Existing automatic anagram generation methods mainly focus on search efficiency in finding word combinations that can form anagrams (Jordan and Monteiro, 2003). However, they place little consideration on the word orders that seem natural to humans. Fortunately, recent progress in NLP techniques now enables natural sentences to be generated in many NLP tasks (Józefowicz et al., 2016; Gehrmann et al., 2018; Skadina and Pinnis, 2017). However, beam search, the de-facto decoding algorithm used in many language generation methods, cannot be used as-is to generate anagrams since it cannot ensure that the generated sentences are anagrams. Although beam search variants that can generate sentences that satisfy some constraints have been proposed (Anderson et al., 2017; Hokamp and Liu, 2017), applying them to anagram generation tasks soon becomes intractable. To summarize, generating natural anagrams is a challenging task that cannot be solved by simply applying existing methods.

In this paper, we propose an anagram generation algorithm. The proposed algorithm is very simple; we run depth-first search to enumerate anagrams. As to the generation procedure, we use a neural language model and character frequency to suppress unnecessary search operations. Due to the power of the currently available pre-trained neural language models, this simple approach works surprisingly well. In experiments that employ human evaluations we confirm that the proposed natural anagram generation method can generate more natural anagrams than a baseline method.

## 2 Related Work

Although anagram generation is NP-hard, the basic algorithm that can output a set of words as anagrams is relatively simple (Jordan and Monteiro, 2003). As a result many anagram generation software and web services exist including the Internet Anagram Server[2] and the Anagram Artist[3]. Unfortunately, these methods pay scant attention to the naturalness of the generated anagrams. In Wikipedia (Wikipedia, 2019), it is said that the

This work was conducted while the author was at NTT Communication Science Laboratories.

[1] NP-hardness of anagram generation can be proved by using the fact that it contains an exact cover problem, which has been proven to be NP-complete (Karp, 1972), as a special case.

[2] https://wordsmith.org/anagram/

[3] https://www.anagrammy.com/resources/anagram_artist.html



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6408–6412, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

automatic anagram generators work poorly when challenged with multi-word anagrams since they cannot take word meanings into account.

The anagram generation task can be seen as a variant of the language generation problem under some constraints. Anderson et al. (2017) proposes a constrained beam search algorithm for image captioning. Their method can generate sentences containing pre-specified words. Their method first creates a finite-state automaton (FSA) representing the constraints. Then it runs beam search over the FSA to obtain results that satisfy the constraints. It is possible to apply their method to anagram generation. However, since the number of FSA states grows exponentially with input length, it rapidly becomes intractable. Hokamp and Liu (2017) proposes the constrained decoding method called grid beam search for machine translation. It is also impractical if applied to the anagram generation task.

## 3 Preliminaries

We represent sentence $s$ as a sequence of words, $s = (w_1, w_2, \ldots, w_{|s|})$, where $|s|$ is the number of words contained in $s$. Every word $w_i$ is selected from vocabulary $V$, which consists of $|V|$ distinct words. We use $C(s)$ to represent the multiset of characters contained in $s$. For example, if $s =$ "Christmas", then $C(s) = \{\mathrm{a}, \mathrm{c}, \mathrm{h}, \mathrm{i}, \mathrm{m}, \mathrm{r}, \mathrm{s}, \mathrm{s}, \mathrm{t}\}$. In the following, we ignore case and we assume that every word $w \in V$ consists of the characters from a to z. We say sentence $t$ is an *anagram* of sentence $s$ if $C(s) = C(t)$. For example, if $t =$ "Trims cash", then $C(t) = \{\mathrm{a}, \mathrm{c}, \mathrm{h}, \mathrm{i}, \mathrm{m}, \mathrm{r}, \mathrm{s}, \mathrm{s}, \mathrm{t}\}$, which is equivalent to $C(s)$ for $s =$ "christmas". The *anagram generation problem* is the problem of finding sentence $t$ given input sentence $s$, where $t$ is an anagram of $s$.

We use a language model when generating anagrams. Let $p(s)$ be the probability that sentence $s = (w_1, w_2, \ldots, w_{|s|})$ appears. A language model is a stochastic model that gives conditional probability $p(w_{i+1}|w_1, \ldots, w_i)$. A language model derives probability $p(s)$ by

$$p(s) = p(w_1) \prod_{i=1}^{|s|-1} p(w_{i+1}|w_1, \ldots, w_i). \quad (1)$$

While our anagram generation algorithm can use any language model, our experiments use the neural language model of ELMo (Peters et al., 2018).

## 4 Anagram Generation

Our anagram generation algorithm uses simple DFS. Given input string $s$, we start the search with the initial string $t =$"$\langle \mathtt{S} \rangle$" and then try to append a word $w \in V$ to the tail of $t$ for each step to search for anagrams of $s$. If we find that $t$ yields no solution by appending any words to it, we pop back the last word from $t$ and continue the search by appending a different word to the tail of $t$. Since there are $|V|^N$ possible sentences consisting of $N$ words, naive DFS is intractable. Thus we need to reduce the search steps. For this we introduce the following two criteria.

The first criteria checks frequency of each character appearing in $t$. Partial sentence $t$ cannot be an anagram of $s$ by appending any words to it once the condition $C(t) \setminus C(s) \neq \emptyset$ is satisfied. For example, partial sentence $t =$"$\langle \mathtt{S} \rangle$ The" cannot form an anagram of $s =$"Christmas" since $C(t) \setminus C(s) = \{\mathrm{e}\} \neq \emptyset$.

The second criterion exploits sentence probability. Since a language model can evaluate how natural a sentence is by calculating sentence occurrence probability, we threshold this probability and remove candidates with insufficient sentence probability. We use score function $f(w)$ defined as

$$f(w) = p(w \mid t) \cdot |w|^2 \quad (2)$$

as the score yielded by adding word $w$ to sentence $t$, where $|w|$ is the number of characters of $w$. We select word $w$ if it makes $f(w)$ larger than threshold value $\eta$. We use the number of characters of word $w$ in the score function since shorter words tend to have high conditional probability $p(w \mid t)$ but using them would often read to unnatural sentences that consist of only very short ($|w| \leq 2$) words.

**Branch and Bound for top-$K$ search** If we need only a few solutions, then we can further reduce the number of search steps by employing a simple branch and bound technique. Here we assume that we need $K$ anagrams with high sentence probability $p(t)$. We use the fact that the sentence probability $p(t)$ defined in (1) is monotone decreasing when we append new word $w$ to the sentence tail. Therefore, once we have found $K$ anagrams and their probabilities are larger than $\theta$, we can omit unnecessary subsequent search steps if the current partial sentence $t$ satisfies $\theta > p(t)$.

We show the final natural anagram generation

**Algorithm 1** Anagram generation algorithm

**Input:** Input sentence $s$
**Output:** At most $K$ anagrams of $s$

```
1: t ←"⟨S⟩"
2: A ← a set of K dummy solutions
3: θ ← 0.0
4: SEARCH(t, A)
5: Output anagrams in A.
6:
7: procedure SEARCH(t, A)
8:     if t is an anagram of s then
9:         if p(t) > min_{a∈A} p(a) then
10:            Replace a_min = argmin_{a∈A} p(a) with t.
11:            θ ← min_{a∈A} p(a)
12:    else
13:        for all w ∈ V do
14:            if f(w) > η and
15:        C(t + w) \ C(s) = ∅ and p(t + w) > θ then
16:                SEARCH(t + w)
```

| Score | Description |
|---|---|
| 5 | Natural as a written language. |
| 4 | Natural as a spoken language. |
| 3 | A few grammatical errors. |
| 2 | Meaning is guessable. |
| 1 | Meaning is unclear. |

Table 1: Human evaluation criteria.

procedure in Algorithm 1. It first initializes sentence $t$ and set of top-$K$ anagrams $A$ and then calls the recursive search procedure SEARCH. SEARCH first checks whether $t$ is an anagram or not (line 8). If $t$ is an anagram whose probability $p(t)$ is larger than $\min_{a \in A} p(a)$, then replace anagram $a^* = \text{argmin}_{a \in A} p(a)$ with $t$ (line 10). If $t$ is not an anagram, we proceed with the search by appending a word $w \in V$ to the tail of $t$. Here we use $t+w$ to represent the sentence obtained by appending $w$ to the tail of $t$. We check that appending word $w$ satisfies the conditions shown above (lines 14, 15), then we recursively call SEARCH$(t + w)$.

**Implementation Details** We tried to use the data structure called *dancing links* (Knuth, 2000) for managing the set of words satisfying the character frequency constraints. Dancing links is a link-based data structure that supports efficient DFS for some combinatorial search problems (Details are shown in (Knuth, 2019)). However, we found that using a simple array-based representation is sufficient since the conditional probability evaluation consumes most of the time.

## 5 Experiments

We conduct human evaluations to assess the performance of the proposed method. For inputs, we randomly select 100 titles of English Wikipedia articles whose length lies between 15 to 20 characters. We use the language model included in the TensorFlow (Abadi et al., 2015) implementation of ELMo[4], which is a CNN-LSTM language model trained with a 1 billion word benchmark corpus and that consists of approximately 800 million tokens[5]. It is also possible to combine the proposed method with conventional n-gram language models. However, since the superiority of neural language models over n-gram models has been reported in previous works (e.g., (Du et al., 2016; Józefowicz et al., 2016)), we conducted experiments using only ELMo. We use parameter $\eta = 0.005$ and we conduct top-$K$ search with $K = 5$. We terminate the search after 20000 steps, which takes about 40 minutes.

| Method | Average Score | |
|---|---|---|
| Proposed | 4.11 | (0.99) |
| Baseline | 2.18 | (1.17) |

Table 2: Human evaluation results. The values in parentheses are standard deviations.

The baseline method used Anagram Artist, a freely available anagram generation software package. Anagram Artist can generate anagrams consisting of at most 5 words given inputs with lengths less than 25 characters. Although the algorithm used in anagram artist is not explained, it seems to use dictionary matching combining with some heuristics for deciding word order. We also tried an Internet anagram server. However, the quality of the anagrams generated by those two baseline methods are not much different. We compute the perplexities of the outputs of the baseline method and use the values to select the top-$K$ anagrams. We run all experiments on computers with GeForce GTX 1080 Ti GPUs. The proposed method was mainly implemented in Python and the module for checking character frequency constraints was implemented in C++. We asked three native speakers to rate each anagram with scores ranging from 1 to 5 following the criteria shown in Table 1.

Table 2 shows the human evaluation results, where we compute the average score by first com-

[4] https://github.com/allenai/bilm-tf

[5] We also considered the use of pre-trained BERT (Devlin et al., 2018) since it is trained with a larger dataset. However, the pre-trained model does not contain weights of output layers and we cannot use it as-is for our task.

| Inputs | Proposed outputs | | Baseline outputs | |
|---|---|---|---|---|
| Economic experiment | Expect more in income | 5.0 | Nonce epimer toxemic | 1.0 |
| | Mine economic expert | 3.0 | Peen commix neoteric | 1.0 |
| | One comic experiment | 4.3 | Entice opener commix | 1.3 |
| | Common experience it | 2.7 | Mine compeer exciton | 1.7 |
| | More except in income | 3.7 | Peen commix erection | 1.0 |
| What would Reagan do | Two had a large wound | 3.3 | Two waul dragonhead | 1.0 |
| | We had a long draw out | 4.7 | What wauled gadroon | 1.0 |
| | What would a range do | 3.3 | What godown radulae | 1.0 |
| | We laugh and draw too | 4.7 | Thaw godown radulae | 1.0 |
| | We had no law to guard | 5.0 | Wood gulden waratah | 1.0 |
| Ashgate publishers | Belarus gas the ship | 2.0 | Litharge subphases | 1.3 |
| | Higher sales at pubs | 5.0 | Is the alpha burgess | 2.0 |
| | This has a superb leg | 4.7 | This huge sparables | 1.3 |
| | The bags are plus his | 2.7 | A lighter subphases | 3.3 |
| | She has a subtle grip | 5.0 | The agas publishers | 2.7 |

Table 3: Example anagrams generated by the proposed method and baseline methods.

| Anagrams | Proposed outputs |
|---|---|
| William Shakespeare ↓ I am a weakish speller | Phillies was a remake |
| | We all make his praise |
| | Speaker William Shea |
| | Hawaii keeps smaller |
| | William Ashe speaker |
| Tom Marvolo Riddle ↓ I am Lord Voldemort | I told Mr Lord a move |
| | Roll met David Romo |
| | Mr Ali drove to Mold |
| | Dr Millar moved too |
| | Mr Orlov told media |

Table 4: Results of generating famous anagrams.

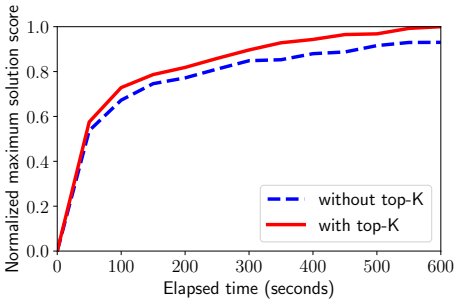


Figure 1: Running time comparison of the proposed method when applying the branch and bound method for retaining top-$K$ solutions.

puting the average score of the three evaluators for each of the top-$K$ outputs, then selecting the maximum of the $K$ scores as the score of the method. The average shown in the table is the average of such maximum scores. The proposed method failed to generate any anagram for 3 instances. We therefore set the score of the proposed method to 1.0 for them. We can see that the proposed method yielded significantly higher human evaluation scores. Table 3 shows example anagrams generated by the proposed method and the baseline method. While the baseline method seems to employ some heuristics for deciding the word order, the combinations of selected words tend to form meaningless sentences. On the other hand, the proposed method tends to generate readable sentences. The anagrams shown in the table are listed in decreasing order of sentence probability. We can find that sentences with high probability are not always highly rated by the human evaluations. However, the normalized discounted cumulative gain (NDCG) score for the proposed method was 0.86. It suggests that using sentence probability works well for finding natural anagrams.

To evaluate the effectiveness of applying the branch and bound technique, we select problem instances that took longer than 600 seconds to complete and compared the maximum sentence probabilities of solutions found at a given time when top-$K$ search is used. Figure 1 shows the comparison results, where the vertical axis plots the maximum scores of found anagrams normalized by the maximum one found within 600 seconds. This figure suggests using branch and bound accelerates the output of high score solutions. For example, with top-$K$, solutions with 0.8 maximum solution scores are found after 150 seconds, while it takes about 250 seconds without top-$K$ search.

We also evaluated whether some famous anagrams appearing in Wikipedia (Wikipedia, 2019) can be generated with the proposed method. Table 4 shows the results of trying to generate fa-

mous anagrams. Unfortunately, the proposed method fails to find the famous anagram for these inputs. This is due to the fact that the famous anagrams use relatively rare words. However, the generated anagrams contain natural sentences.

## 6 Conclusion

We proposed an algorithm for generating natural anagrams. Surprisingly, a simple depth-first search approach is shown to be sufficient for generating natural anagrams when it is combined with modern neural language models. Since our current search procedure is relatively naive, there is room for improvement. More efficient use of GPUs in performing the DFS procedure is a promising future work.

## Acknowledgements

We thank the anonymous reviewers for their insightful comments.

## References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2017. Guided open vocabulary image captioning with constrained beam search. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 936–945, Copenhagen, Denmark. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Jun Du, Yan-Hui Tu, Lei Sun, Feng Ma, Hai-Kun Wang, Jia Pan, Cong Liu, Jing-Dong Chen, and Chin-Hui Lee. 2016. The ustc-iflytek system for chime-4 challenge. *Proc. CHiME*, pages 36–38.

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109, Brussels, Belgium. Association for Computational Linguistics.

Chris Hokamp and Qun Liu. 2017. Lexically constrained decoding for sequence generation using grid beam search. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1535–1546, Vancouver, Canada. Association for Computational Linguistics.

Timothy R. Jordan and Axel Monteiro. 2003. Generating anagrams from multiple core strings employing user-defined vocabularies and orthographic parameters. *Behavior Research Methods, Instruments, & Computers*.

Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *CoRR*, abs/1602.02410.

Richard M Karp. 1972. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer.

Donald E Knuth. 2000. Dancing links. In *Millenial Perspectives in Computer Science*, pages 187–214.

Donald E Knuth. 2019. *The Art of Computer Programming, Volume 4 Pre-fascicle 5c, Dancing Links*. Addison-Wesley.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Inguna Skadina and Mārcis Pinnis. 2017. NMT or SMT: Case study of a narrow-domain English-Latvian post-editing project. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 373–383, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Henry Benjamin Wheatley. 1862. Of anagrams: A monograph treating of their history from the earliest ages to the present time.

Wikipedia. 2019. Anagram — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/wiki/Anagram`. [Online; accessed 21-May-2019].

# STANCY: Stance Classification Based on Consistency Cues

**Kashyap Popat**[1], **Subhabrata Mukherjee**[2], **Andrew Yates**[1], **Gerhard Weikum**[1]
[1]Max Planck Institute for Informatics, Saarbrücken, Germany
[2]Microsoft Research, Redmond, WA, USA
[1]{kpopat,ayates,weikum}@mpi-inf.mpg.de
[2]subhabrata.mukherjee@microsoft.com

## Abstract

Controversial claims are abundant in online media and discussion forums. A better understanding of such claims requires analyzing them from different perspectives. Stance classification is a necessary step for inferring these perspectives in terms of supporting or opposing the claim. In this work, we present a neural network model for stance classification leveraging BERT representations and augmenting them with a novel consistency constraint. Experiments on the *Perspectrum* dataset, consisting of claims and users' perspectives from various debate websites, demonstrate the effectiveness of our approach over state-of-the-art baselines.

## 1 Introduction

There is an abundance of contentious claims on the Web including controversial statements from politicians, biased news reports, rumors, etc. People express their perspectives about these controversial claims through various channels like editorials, blog posts, social media, and discussion forums. To achieve a deeper understanding of these claims, we need to understand users' perspectives and stance towards the claims. Recent research (FNC-1, 2016; Baly et al., 2018; Chen et al., 2019) has shown stance classification to be a critical step for information credibility and automated fact-checking.

**Prior Work and Limitations:** Prior approaches for stance classification proposed in Somasundaran and Wiebe (2010); Anand et al. (2011); Walker et al. (2012); Hasan and Ng (2013, 2014); Sridhar et al. (2015); Sun et al. (2018) rely on various linguistic features, e.g., n-grams, dependency parse tree, opinion lexicons, and sentiment to determine the stance of perspectives regarding controversial topics. Ferreira and Vlachos (2016) further incorporate natural language claims and propose a logistic regression model using the lexical and semantic features of claims and perspectives. SemEval tasks (Mohammad et al., 2016; Kochkina et al., 2017) and other approaches (Chen and Ku, 2016; Lukasik et al., 2016; Sobhani et al., 2017) have focused on determining stance only in Tweets.

Bar-Haim et al. (2017) propose classifiers based on hand-crafted lexicons to identify important phrases in perspectives and their consistency with the claim to predict the stance However, their model critically relies on manual lexicons and assumes that the important phrases in claims are already identified.

Neural-network-based approaches for stance classification learn the claim and perspective representations separately and later combine them with conditional LSTM encoding (Augenstein et al., 2016), attention mechanisms (Du et al., 2017) or memory networks (Mohtarami et al., 2018). Some neural network models also incorporate lexical features (Riedel et al., 2017; Hanselowski et al., 2018). None of these approaches leverage knowledge acquired from massive external corpora.

**Approach and Contributions:** To overcome the limitations of prior works, we present STANCY, a neural network model for stance classification. Given an input pair of a claim and a user's perspective, our model predicts whether the perspective is *supporting* or *opposing* the claim. For example, the claim *"You have nothing to worry about surveillance, if you have done nothing wrong"* is supported by the user perspective *"Information gathered through surveillance could be used to fight terrorism"* and opposed by another user perspective *"With surveillance, the user privacy will go away!"*.

Our model for stance classification leverages representations from the BERT (Bidirectional



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6413–6418, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

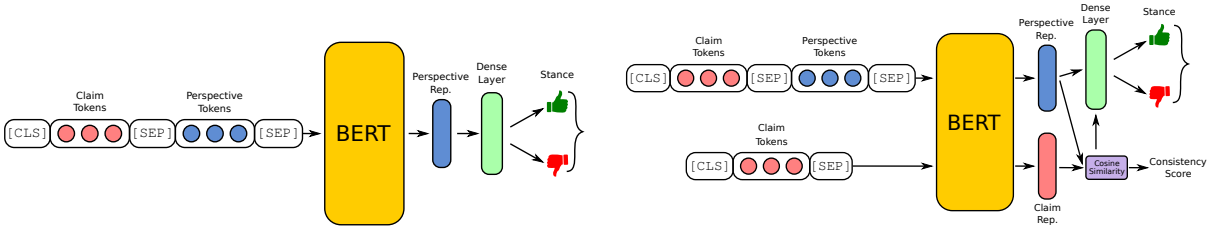


(a) $\text{BERT}_{\text{BASE}}$: Fine-tuning BERT for stance classification.

(b) $\text{BERT}_{\text{CONS}}$: Enhancing BERT using the joint loss ($loss_{ce}$ for stance classification and $loss_{cos}$ for consistency).

Figure 1: BERT-based methods for determining the stance of the perspective with respect to the claim.

Encoder Representations from Transformers) neural network model (Devlin et al., 2019). BERT is trained on huge text corpora and serves as background knowledge. We fine-tune BERT for our task which also allows us to jointly model claims and perspectives. Furthermore, we enhance our model by augmenting it with a novel consistency constraint to capture agreement between the claim and perspective.

Key contributions of this paper are:

- **Model**: A neural network model for stance classification leveraging BERT representations learned over massive external corpora and a novel consistency constraint to jointly model claims and perspectives.
- **Interpretability**: A simple approach to interpret the contribution of perspective tokens in deciding their stance towards the claim.
- **Experiments**: Experiments on a recent dataset, *Perspectrum*, highlighting the effectiveness of our approach with error analysis.

## 2 BERT-based Approaches

In this section, first we describe the base model, $\text{BERT}_{\text{BASE}}$, that is adapted for the stance classification (Chen et al., 2019). Thereafter, we present our consistency-aware model, $\text{BERT}_{\text{CONS}}$.

### 2.1 Adapting BERT for Stance Classification

The goal of the stance classification task is to determine the stance of the user *Perspective ($P$)* with respect to the *Claim ($C$)*. Since this task involves a pair of sentences ($C$ and $P$), we follow the approach for sentence pair classification task as proposed in Devlin et al. (2019); Chen et al. (2019).

In order to obtain the representation $X^{P|C}$ of $P$ with respect to $C$, this sentence pair is fused into a single input sequence by using a special classification token (`[CLS]`) and a separator token (`[SEP]`): `[CLS]` $C_{toks}$ `[SEP]` $P_{toks}$ `[SEP]`. The input sequences are tokenized using WordPiece tokenization. The final hidden state representation corresponding to the `[CLS]` token is used as $X^{P|C} \in R^H$. The classification probability is given by passing this representation through the softmax layer:

$$\hat{y} = softmax(X^{P|C}W^T) \quad (1)$$

where softmax layer weights $W \in R^{H \times K}$ and $K$ is the number of stance (classification) labels. All the parameters of BERT and $W$ are fine-tuned jointly by minimizing the cross-entropy loss ($loss_{ce}$). The architecture of this model, $\text{BERT}_{\text{BASE}}$, is shown in Figure 1a.

### 2.2 Consistency-aware Stance Classification

In this setting, we want to incorporate the consistency between the claim ($C$) and perspective ($P$) representations. We hypothesize that the latent representations of claim and perspective should be *dissimilar* if the perspective *opposes* the claim, whereas their representations should be *similar* if the claim is *supported* by the perspective. We capture this with the following components.

**Claim Representation:** To capture the latent representation of the claim, we use only the claim text as the input sequence to BERT, i.e., `[CLS]` $C$ `[SEP]`. The final hidden state of the first input token (`[CLS]`) is used as the claim's representation $X^C \in R^H$.

**Perspective Representation:** Latent representation of the perspective (with respect to the claim) is captured by fusing the two sequences as described in Section 2.1. We pack the claim and perspective pair as a single input sequence and use the final hidden state of the first input token as the perspective representation $X^{P|C} \in R^H$.

| Split | Supporting Pairs | Opposing Pairs | Total Pairs |
|---|---|---|---|
| train | 3603 | 3404 | 7007 |
| dev | 1051 | 1045 | 2096 |
| test | 1471 | 1302 | 2773 |
| **Total** | 6125 | 5751 | 11876 |

Table 1: Perspectrum data statistics.

**Capturing Consistency:** To incorporate the consistency between claim and perspective representations, we use the *cosine embedding loss*:

$$loss_{cos} = \begin{cases} 1 - \cos(X^C, X^{P|C}) & y_{sim} = 1 \\ max(0, \cos(X^C, X^{P|C})) & y_{sim} = -1 \end{cases}$$

where $\cos(.)$ is the cosine similarity function. $y_{sim}$ is equal to 1 if the perspective is *supporting* the claim (*similar* representations), and $-1$ if the claim is *opposed* by the perspective (*dissimilar* representations).

**Joint Loss:** The classification probabilities are determined by concatenating $X^{P|C}$ and $\cos(X^C, X^{P|C})$ and passing it through a softmax layer. However, unlike the $\text{BERT}_{\text{BASE}}$ configuration, parameters of the consistency-aware model are learned by optimizing the joint loss function: $loss = loss_{ce} + loss_{cos}$. With this joint loss function, we enforce consistency between latent representations of the claim and perspective. The architecture of this consistency-aware model, $\text{BERT}_{\text{CONS}}$, is shown in Figure 1b.

## 3 Experimental Setup

For our experiments, we consider the base version of BERT[1] with 12 layers, 768 hidden size, and 12 attention heads. We fine-tune BERT-based models using the Adam optimizer with learning rates $\{1, 3, 5\} \times 10^{-5}$ and training batch sizes $\{24, 28, 32\}$. We choose the best parameters based on the development split of the dataset. For measuring the performance, we use per-class and macro-averaged Precision/Recall/F1.

### 3.1 Dataset

We evaluate our approach on the *Perspectrum* dataset (Chen et al., 2019). *Perspectrum* contains claims and users' perspectives from various online debate websites like `idebate.com`, `debatewise.org`, and `procon.org`. Each claim has different perspectives along with the stance (*supporting* or *opposing* the claim). We use the same train/dev/test split as provided in the released dataset. Statistics of the dataset is shown in Table 1.

### 3.2 Baselines

We use the following baselines:

**LSTM**: A long short-term memory (LSTM) model, in which we pass the claim and perspective word representations (using GloVE-6B word embeddings of size 300) through a bidirectional LSTM. Then we concatenate the final hidden states of the claim and perspective, and pass it through dense layers with ReLU activations.

**ESIM**: An enhanced sequential inference model (ESIM) for natural language inference proposed in Chen et al. (2017).

**MLP**: Multi-layer perceptron (MLP) based model using lexical and similarity-based features – presented as a *simple but tough-to-beat baseline* for stance detection in Riedel et al. (2017).

**WordAttn**: Our implementation of word-by-word attention-based model using long short-term memory networks (Rocktäschel et al., 2016).

**LangFeat**: A random forest classifier using linguistic lexicons like NRC lexicon (Mohammad and Turney, 2010), hedges (e.g., *possibly*, *might*, etc.), positive/negative sentiment words (Hu and Liu, 2004), MPQA subjective lexicon (Wilson et al., 2005) and bias lexicon (Recasens et al., 2013) along with sentiment scores as features.

**$\text{BERT}_{\text{BASE}}$**: Approach proposed in Chen et al. (2019) (as described in Section 2.1).

**Human**: Human performance on this task as reported in Chen et al. (2019).

## 4 Results and Discussion

Stance classification performance of our model and the baselines on the *test* split of the Perspectrum dataset are presented in Table 2. Our consistency-aware model $\text{BERT}_{\text{CONS}}$ outperforms all the other baselines. It achieves a performance improvement of about 2 points in F1-score over the strong baseline corresponding to the $\text{BERT}_{\text{BASE}}$ model (p-value of $4.985\text{e}{-4}$ as per the McNemar test). This highlights the value addition achieved by incorporating consistency cues. Since the BERT-based models incorporate the knowledge acquired from massive external corpora, our model, $\text{BERT}_{\text{CONS}}$, captures better semantics and outperforms the other baselines.

[1]BERT implementation: `https://git.io/fhbJQ`

| Approach | Supporting | | | Opposing | | | Overall (Macro) | | |
|---|---|---|---|---|---|---|---|---|---|
| | Prec. | Recall | F1 | Prec. | Recall | F1 | Prec. | Recall | F1 |
| LSTM | 63.42 | 58.80 | 61.02 | 56.99 | 61.67 | 59.24 | 60.20 | 60.24 | 60.13 |
| ESIM | 64.38 | 61.32 | 62.81 | 58.53 | 61.67 | 60.06 | 61.46 | 61.50 | 61.44 |
| MLP | 64.53 | 60.98 | 62.71 | 58.50 | 62.14 | 60.26 | 61.51 | 61.56 | 61.48 |
| WordAttn | 64.43 | 63.43 | 63.93 | 59.40 | 60.45 | 59.92 | 62.07 | 62.03 | 62.04 |
| LangFeat | 63.74 | 75.05 | 68.94 | 64.75 | 51.77 | 57.53 | 64.24 | 63.41 | 63.23 |
| $\text{BERT}_{\text{BASE}}$ | 78.43 | 80.08 | 79.25 | 76.95 | **75.12** | 76.02 | 77.69 | 77.60 | 77.63 |
| $\text{BERT}_{\text{CONS}}$ | **79.05** | **84.64** | **81.75** | **81.14** | 74.65 | **77.76** | **80.09** | **79.65** | **79.95** |
| Human | - | - | - | - | - | - | 91.3 | 90.6 | 90.9 |

Table 2: Comparison of our approach $\text{BERT}_{\text{CONS}}$ with different baseline models for stance classification.

| Opposing Class | Supporting Class |
|---|---|
| unauthorized, falsely, even though, unlike, cannot, not everyone, could strike, could further weaken, jeopardize, impacts, may not provide, ... | enabling, ensuring, prevail, positive discrimination, gains, help reduce, would improve, right, would allow, encourage, more effective, ... |

Table 3: Top phrases for determining stance.

### 4.1 Interpreting Token-level Contribution

Due to the massive structure of BERT with a complex attention mechanism, it is difficult to interpret the significance of different lexical units in the text. Therefore, we propose a simple technique to interpret the contribution of each token in the text in determining the stance.

Given the claim ($C$) and perspective ($P$) pair, we tokenize $P$ into *phrases*. We record the change in stance classification probabilities by adding one perspective phrase at a time to the input:

$$\Delta_i = |BERT_{CONS}(C, P_i) - BERT_{CONS}(C, P_{i-1})|$$

where $P_i$ is the prefix of $P$ up to the $i^{th}$ phrase. This helps us in understanding the contribution of each perspective phrase towards determining the stance – the larger the change in the classification probabilities, the larger the contribution. For this analysis, we consider unigrams and chunks from a shallow parser as phrases. The top contributing phrases for the *supporting* and *opposing* classes are shown in Table 3.

### 4.2 Error Analysis

In this section, we analyze why the task of stance classification is challenging and why the performance of the best model configuration is far from human performance as observed by the performance gap in Table 2.

**Negations:** One of the major challenges in solving this task is understanding negations and their scope. For example, given the claim *"College education is worth it"*, the perspective *"Many college graduates are employed in jobs that **do not** require college degrees"* is opposing the claim. However, our model is not able to capture that the negation phrase *'do not require'* opposes the claim. On the other hand, the presence of negation in the perspective does not necessarily imply that it is opposing the claim. Contrast this with the claim *"Chess must be at the Olympics"* and perspective *"Chess is currently **not** an Olympic sport, but it should be"* – where the negation is merely a part of the statement and the stance is given by the discourse segment following *'but'*.

**Commonsense:** Determining the stance may require commonsense knowledge. For example, the claim *"Chess must be at the Olympics"* is opposed by the perspective *"Olympic sports are supposed to be **physical**"*. To understand this, the model should have the background knowledge that chess is not a physical sport.

**Semantics:** Understanding the stance also involves a deeper understanding of semantics. For example, given the claim *"Make all museums free of charge"* is opposed by the perspective *"State funding should be used **elsewhere**"*. Here, the word *'elsewhere'* is the key cue which determines the stance. However, the presence of the word *'elsewhere'* does not necessarily imply that the perspective is opposing the claim. For instance, the perspective *"We could spend the money **elsewhere**"* is supporting the claim *"The EU should significantly reduce the amount it spends on agricultural production subsidies"*. Hence, the polarity of the word *'elsewhere'* is determined by the context and semantics of the statement.

## 5 Conclusion

In this work, we propose a consistency-aware neural network model for stance classification. Our model leverages representations from the BERT model trained over massive external corpora and a novel consistency constraint to jointly model claims and perspectives. Our experiments on a recent benchmark highlight the advantages of our approach. We also study the gap in human performance and the performance of the best model for stance classification.

## Acknowledgments

This research was partly supported by the ERC Synergy Grant "imPACT" (No. 610150).

## References

Pranav Anand, Marilyn Walker, Rob Abbott, Jean E. Fox Tree, Robeson Bowmani, and Michael Minor. 2011. Cats rule and dogs drool!: Classifying stance in online debate. In *Proceedings of the 2nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis (WASSA 2.011)*, pages 1–9, Portland, Oregon. Association for Computational Linguistics.

Isabelle Augenstein, Tim Rocktäschel, Andreas Vlachos, and Kalina Bontcheva. 2016. Stance detection with bidirectional conditional encoding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 876–885, Austin, Texas. Association for Computational Linguistics.

Ramy Baly, Mitra Mohtarami, James Glass, Lluís Màrquez, Alessandro Moschitti, and Preslav Nakov. 2018. Integrating stance detection and fact checking in a unified corpus. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 21–27, New Orleans, Louisiana. Association for Computational Linguistics.

Roy Bar-Haim, Indrajit Bhattacharya, Francesco Dinuzzo, Amrita Saha, and Noam Slonim. 2017. Stance classification of context-dependent claims. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 251–261, Valencia, Spain. Association for Computational Linguistics.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1657–1668, Vancouver, Canada. Association for Computational Linguistics.

Sihao Chen, Daniel Khashabi, Wenpeng Yin, Chris Callison-Burch, and Dan Roth. 2019. Seeing things from a different angle:discovering diverse perspectives about claims. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 542–557. Association for Computational Linguistics.

Wei-Fan Chen and Lun-Wei Ku. 2016. UTCNN: a deep learning model of stance classification on social media text. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1635–1645, Osaka, Japan. The COLING 2016 Organizing Committee.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Jiachen Du, Ruifeng Xu, Yulan He, and Lin Gui. 2017. Stance classification with target-specific neural attention. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 3988–3994.

William Ferreira and Andreas Vlachos. 2016. Emergent: a novel data-set for stance classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1163–1168, San Diego, California. Association for Computational Linguistics.

FNC-1. 2016. Fake news challenge stage 1 (fnc-1): Stance detection.

Andreas Hanselowski, Avinesh PVS, Benjamin Schiller, Felix Caspelherr, Debanjan Chaudhuri, Christian M. Meyer, and Iryna Gurevych. 2018. A retrospective analysis of the fake news challenge stance-detection task. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1859–1874, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Kazi Saidul Hasan and Vincent Ng. 2013. Stance classification of ideological debates: Data, models, features, and constraints. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1348–1356, Nagoya, Japan. Asian Federation of Natural Language Processing.

Kazi Saidul Hasan and Vincent Ng. 2014. Why are you taking this stance? identifying and classifying reasons in ideological debates. In *EMNLP*.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 168–177, New York, NY, USA. ACM.

Elena Kochkina, Maria Liakata, and Isabelle Augenstein. 2017. Turing at semeval-2017 task 8: Sequential approach to rumour stance classification with branch-lstm. *CoRR*, abs/1704.07221.

Michal Lukasik, P. K. Srijith, Duy Vu, Kalina Bontcheva, Arkaitz Zubiaga, and Trevor Cohn. 2016. Hawkes processes for continuous time sequence classification: an application to rumour stance classification in twitter. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 393–398, Berlin, Germany. Association for Computational Linguistics.

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. SemEval-2016 task 6: Detecting stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 31–41, San Diego, California. Association for Computational Linguistics.

Saif M. Mohammad and Peter D. Turney. 2010. Emotions evoked by common words and phrases: Using mechanical turk to create an emotion lexicon. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, CAAGET '10.

Mitra Mohtarami, Ramy Baly, James Glass, Preslav Nakov, Lluís Màrquez, and Alessandro Moschitti. 2018. Automatic stance detection using end-to-end memory networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 767–776, New Orleans, Louisiana. Association for Computational Linguistics.

Marta Recasens, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. 2013. Linguistic models for analyzing and detecting biased language. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1650–1659, Sofia, Bulgaria. Association for Computational Linguistics.

Benjamin Riedel, Isabelle Augenstein, Georgios P. Spithourakis, and Sebastian Riedel. 2017. A simple but tough-to-beat baseline for the fake news challenge stance detection task. *CoRR*, abs/1707.03264.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. In *ICLR*.

Parinaz Sobhani, Diana Inkpen, and Xiaodan Zhu. 2017. A dataset for multi-target stance detection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 551–557, Valencia, Spain. Association for Computational Linguistics.

Swapna Somasundaran and Janyce Wiebe. 2010. Recognizing stances in ideological on-line debates. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 116–124, Los Angeles, CA. Association for Computational Linguistics.

Dhanya Sridhar, James Foulds, Bert Huang, Lise Getoor, and Marilyn Walker. 2015. Joint models of disagreement and stance in online debate. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 116–125, Beijing, China. Association for Computational Linguistics.

Qingying Sun, Zhongqing Wang, Qiaoming Zhu, and Guodong Zhou. 2018. Stance detection with hierarchical attention network. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2399–2409, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Marilyn Walker, Pranav Anand, Rob Abbott, and Ricky Grant. 2012. Stance classification using dialogic properties of persuasion. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 592–596, Montréal, Canada. Association for Computational Linguistics.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 347–354, Stroudsburg, PA, USA. Association for Computational Linguistics.

# Cross-lingual intent classification in a low resource industrial setting

**Talaat Khalil*** **Kornel Kiełczewski*** **Georgios Christos Chouliaras**
**Amina Keldibek** **Maarten Versteegh**
Booking.com Amsterdam
{talaat.khalil,kornel.kielczewski}@booking.com

## Abstract

This paper explores different approaches to multilingual intent classification in a low resource setting. Recent advances in multilingual text representations promise cross-lingual transfer for classifiers. We investigate the potential for this transfer in an applied industrial setting and compare to multilingual classification using machine translated text. Our results show that while the recently developed methods show promise, practical application calls for a combination of techniques for useful results.

## 1 Introduction

Classifying the intent represented in a customer message is a core functionality of natural language understanding applications in customer service. Once the intent of the customer's message is understood and various entities are resolved, actions can be triggered, such as automatically replying to the customer, routing the message to the right customer service representative, or asking the customer for more information.

The rise of automation in customer service and the growing customer expectation of immediate answers is generating increasing demand for accurate intent detection across many languages. Recent advances in natural language processing have enabled impressive performance on text classification tasks, provided that large-scale labelled data are available. In industrial settings, this is often not the case, or at least not for all languages for which intent detection is required.

This paper studies what we believe to be a common case in industry, in which a small amount of labeled data are available for a single language, say English, but neither data nor labels are available for other languages. This setting poses an interesting constraint on the application of modern machine learning techniques, in that much fewer in-domain data are available for the task than is generally required. We believe that the findings in this paper can be generalized outside our specific industrial domain and apply in general to the challenge of enabling classification in a target language based on supervision in a source language.

This paper investigates the application of two techniques from the literature to this problem. The first technique is an out-of-domain machine translation system, as can be assumed to be generally available, applied to the source language data set in order to generate supervised data in the target language. The second technique that we investigate is transfer learning based on multi-lingual document representations. We study the results of applying these techniques, as well as the combination of both, in a number of experiments.

## 2 Related work

This study is broadly positioned in the application of cross-lingual transfer, with special focus on low resource applications. We study two approaches to this problem: machine translation and transfer through multilingual text representations.

One approach to NLP tasks in multilingual settings where the target language has scarce resources relies on machine translation. There are two popular methods: translating target language to English and vice versa, translating English to target language (Garca et al., 2012; He et al., 2013). While straightforward, a disadvantage of this approach is that building a reliable machine translation system with few resources in the target language is challenging (Upadhyay et al., 2018).

Enriched word embeddings are used to solve

* These authors contributed equally.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6419–6424,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

various (monolingual) NLP tasks including intent classification (Kim et al., 2016). Multilingual representations of characters, words or documents can be used to solve multilingual NLP tasks. (Klementiev et al., 2012) present a representation for single words for a pair of languages that preserves semantic similarity. This approach relies on parallel data sets, a challenge to their scalability. To address this issue (Søgaard et al., 2015) propose an inter-lingual representation of words obtained with inverted indexing from common subsets of Wikipedia articles on a larger set of languages.

A recent approach is the construction of alignments between monolingual word representations for different languages into a single embedding space (Ammar et al., 2016; Alaux et al., 2018; Conneau et al., 2017; Schuster et al., 2019).

Recent studies evaluated shared sentence representations from multiple languages using both unsupervised learning on monolingual corpora (Lample and Conneau, 2019; Eriguchi et al., 2018) and supervised learning using parallel data (Artetxe and Schwenk, 2018). Advances in generative pretraining models like the Transformer (Vaswani et al., 2017), GPT (Radford et al., 2018), and BERT (Devlin et al., 2018) models make this approach even more promising. In (Lample and Conneau, 2019) authors successfully apply these techniques and propose methods for cross-lingual language modelling, unsupervised and supervised, outperforming previous best approaches.

In this paper, we study the capability of transfer from a source language to a target language through machine translation and pretrained multilingual document representations.

## 3 Data sets

The data set for the source language ("EN") consists of 48,875 English chat messages from customers to the customer service department of a large e-commerce website. The messages typically express intents from customers to get information or make changes on an existing transaction, e.g. cancelling a past transaction, modifying it in a certain way, or enquiring about information related to the transaction or the product. A test set for the target language ("FR") of 4,336 messages was similarly collected from the production system of the conversational agent. The median number of words in the source language messages is 16, the 99$^{th}$ percentile is at 93 words.

The messages were anonymized by replacing email addresses, transaction id's, credit card and telephone numbers and other personally identifiable information with token placeholders. After this process, The messages were manually labeled on 17 intents, covering the majority of incoming requests. Each message received one or more intent labels. The distribution of intents is shown in Figure 1 showing that the English and French sets have roughly the same label distribution, although there are some differences.

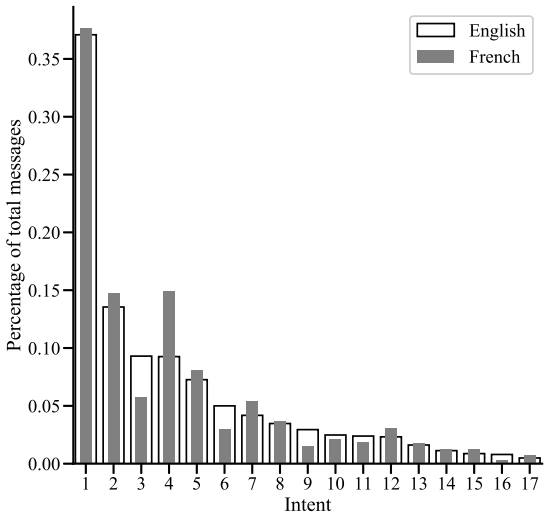


Figure 1: Intent distribution for English and French.

To support the investigation of using machine translation, an additional data set ("MT") was constructed by translating the EN data set using an in-house neural machine translation model (Levin et al., 2017). The machine translation model was trained on the open parallel corpus (Tiedemann, 2012), product descriptions, and fine-tuned for user reviews. While the product descriptions apply to the same industry as the chat messages we are investigating here, the training set for the machine translation model contains no documents from customer service interactions, so we consider it to be an out-of-domain system for the purpose of this study, as might be available to any investigator.

The EN data were split into fixed train, development, and test sets. The same splits were maintained for the translated (MT) data set. The FR data were used only for testing. Table 1 shows the volume of each of the data sets.

| Language | Train | Dev | Test |
|---|---|---|---|
| EN | 39,094 | 4,887 | 4,894 |
| MT | 39,094 | 4,887 | - |
| FR | - | - | 4,336 |

Table 1: Data set sizes for each approach.

## 4 Experiments

We study the effects of two approaches to cross-lingual transfer for intent classification; machine translation and transfer from multilingual document representations. We conduct three main experiments; one each for the two approaches and one that combines both approaches.

### 4.1 Machine translation

Machine translation for cross-lingual transfer is assessed by comparing a classifier trained and tested on the source language (EN) with one trained on the machine translated samples (MT) and tested on the target language (FR).

We evaluate two text classification architectures. The first architecture is a linear support vector machine (SVM) with bag-of-ngram input features and $\ell_1$- and $\ell_2$-regularization, a classic text classification setup (Sebastiani, 2002). A combination of unigram, bigram, and trigram features with tf-idf weighting is used.

The second text classification architecture is a convolutional neural network (CNN) similar to (Kim, 2014). The system represents input documents as sequences of word embeddings (Mikolov et al., 2018; Grave et al., 2018) that are passed through a single convolutional layer followed by a densely connected layer. The network was trained using the Adam (Kingma and Ba, 2015) optimizer and the hyperparameters of the network (size and amount of kernels, size of pre-final layer, dropout- and $\ell_1$-, $\ell_2$-regularization strength) were found using Bayesian optimization (Snoek et al., 2012). These text classification systems were chosen because both are commonly used in industry.

To assess machine translation for cross-lingual transfer, we compare two settings for each of the classifiers. In the first setting, both training and test data come from the source language (EN). In the second setting, training data is machine translated (MT) and test data is drawn from the target language (FR). If machine translation is a feasible strategy for transfer, then the results for the second setting should be close to those for the first.

### 4.2 Multilingual document embeddings

To investigate the use of multilingual document embeddings for cross-lingual transfer, we build a set of classifiers based on representations from the base multilingual BERT model (Devlin et al., 2018). We use the aggregate document representation of the input message and add a single output layer with sigmoid activations for multi-label classification, as recommended in (Devlin et al., 2018). With this system we study two setups.

In the first setup a classifier is trained on the source language (EN) only, and evaluated on both the source (EN) and the target language (FR). The hypothesis here is that if the multilingual representations from BERT allow for lossless transfer between source and target language, then the performance on the target language will be close to the performance on the source language.

In the second setup, we combine the approaches of machine translation and multilingual document representation. We train a similar classifier as above with input representations from BERT on both the source language (EN) data and the machine translated target language (MT) data. A separate classifier is trained only on the machine translated (MT) training set. We compare the performance of these two classifiers on the target (FR) test set. If the architecture facilitates task-specific transfer learning then there will be a difference in the performance between the two classifiers.

Lastly we compare the classifier that uses multilingual document representations and machine translated data with the classifiers that use only either of those approaches. If there is a cumulative effect from the combination of these approaches, then the first will outperform both of the latter.

## 5 Results

For the application described in this paper, false positive identifications of a class are worse than false negatives. Therefore the classifiers are calibrated on the development set to a precision close to 0.90 and maximum recall. This is a common practice in applications in which there is an asymmetry in the cost of errors. Table 2 shows the main results. Note that not all systems could achieve the required precision.

For assessing the feasibility of machine translation for cross-lingual transfer, we compare the re-

| Test | Train | System | P | R | F1 |
|---|---|---|---|---|---|
| EN | EN | svm | .91 | .60 | .72 |
| | EN | cnn | .90 | .67 | .77 |
| | EN | bert | .91 | **.71** | .80 |
| | EN+MT | bert | .91 | **.71** | .80 |
| FR | MT | svm | .86 | .43 | .57 |
| | MT | cnn | .86 | .47 | .61 |
| | EN | bert | .74 | .51 | .61 |
| | MT | bert | .86 | .62 | .72 |
| | EN+MT | bert | .86 | **.63** | .73 |

Table 2: Test set results for the experiments.

sults of the linear SVM and the CNN. For both of these classifiers, the source (EN) to source (EN) setting outperforms the translated (MT) to target (FR) setting, by 0.17 to 0.2 recall and 0.15 to 0.16 F1-score. The latter results are, however, substantially better than chance for both classifiers. It is interesting to note that the results of the SVM do not lag far behind those of the CNN, both for EN and FR, indicating that the machine translation approach is feasible even for the SVM.

Next, the capability of multi-lingual document representations to provide cross-lingual transfer is tested by comparing the performance on EN and FR for the BERT model trained solely on EN. For the FR test case, this model could not be calibrated for a precision $\geq 0.9$, so we can only compare the F1-scores. Here again, the EN setting shows better performance than the FR setting (F1 of 0.80 versus 0.61), indicating that the transfer incurs a loss in performance, but is not entirely without meaning.

Lastly, the BERT system trained on translated (MT) data shows the combination of the approaches. The evaluation on target data (FR) shows that this combination achieves higher performance than all of the above settings, outperforming MT only (0.62 recall versus 0.47 and 0.43), as well as only transfer from multilingual embeddings (0.72 F1 versus 0.61). Adding EN data to this combination model improves performance slightly (0.63 versus 0.62 recall), but these extra data do not seem to have a large effect on the performance of the system. In addition, this latter BERT model, trained on both EN and MT, when tested on the EN test set performs on par with the BERT model trained only on EN data. The implication of this is that the MT data do not introduce a degradation of the model. It is interesting to note that the results of this setting on the target language are higher than those of the SVM on the source language.

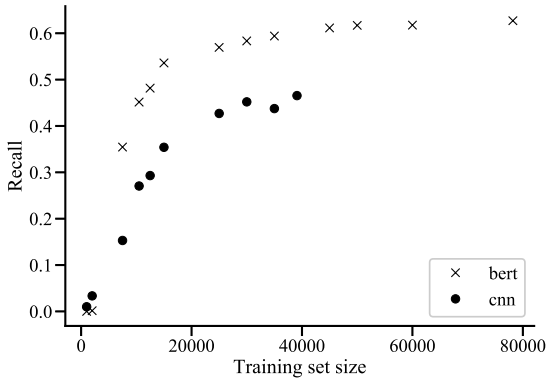


Figure 2: Recall as a function of the size of the training data. This figure shows the recall of the BERT model trained on both EN and MT and the recall of the CNN model trained on MT.

To evaluate the dependence of the BERT system with source and translated data on the amount of data available for fine-tuning the BERT model, we look at the performance as a function of the data set size and compare it in this respect to the CNN. Figure 2 shows that both the BERT model and the CNN performances converge on about 10,000 to 25,000, indicating that only a relatively small amount of labeled data is necessary.

## 6 Discussion

In this paper we studied two techniques for cross-lingual transfer in a typical industrial setting, with small amounts of labeled data. The results show that both machine translated data and multilingual document representations are decent strategies for cross-lingual transfer for intent detection on short chat messages. Both approaches incur a significant loss when compared to the source-to-source classification results, showing that there is still room for improvement before "zero-shot" transfer from multilingual document representations is viable as a stand-alone approach to cross-lingual transfer.

The combination of both techniques performs competitively and the observation that this setting outperforms the SVM in the source-to-source setting illustrates the advances that have been made in recent years in both machine translation and document representation.

It is an open question on whether these results generalize to other language pairs. The languages studied here are both western Indo-European languages so the results obtained here may not apply to pairs of distant languages. In addition, the label distributions between the source and target language in our experiments were fairly closely aligned, which may not apply to other use cases. In general, more investigation is needed to assess how well the current generation of multi-lingual document representations supports cross-lingual transfer and if there are differences in how well the representations between languages align.

It is promising to note though that the results in this study can be obtained with data set sizes that are generally available in industrial settings.

## Acknowledgements

The authors wish to thank Elena Sokolova for useful discussions, Pavel Levin for his comments on an early version of this paper and Nam Pham for his help with the hyperparameter optimization.

## References

Jean Alaux, Edouard Grave, Marco Cuturi, and Armand Joulin. 2018. Unsupervised hyperalignment for multilingual word embeddings. *CoRR*, abs/1811.01124.

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A. Smith. 2016. Massively multilingual word embeddings. *CoRR*, abs/1602.01925.

Mikel Artetxe and Holger Schwenk. 2018. Margin-based parallel corpus mining with multilingual sentence embeddings. *CoRR*, abs/1811.01136.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *CoRR*, abs/1710.04087.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Akiko Eriguchi, Melvin Johnson, Orhan Firat, Hideto Kazawa, and Wolfgang Macherey. 2018. Zero-shot cross-lingual classification using multilingual neural machine translation. *CoRR*, abs/1809.04686.

F. Garca, L. F. Hurtado, E. Segarra, E. Sanchis, and G. Riccardi. 2012. Combining multiple translation systems for spoken language understanding portability. In *2012 IEEE Spoken Language Technology Workshop (SLT)*, pages 194–198.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

X. He, L. Deng, D. Hakkani-Tur, and G. Tur. 2013. Multi-style adaptive training for robust cross-lingual spoken language understanding. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8342–8346.

J. Kim, G. Tur, A. Celikyilmaz, B. Cao, and Y. Wang. 2016. Intent detection using semantically enriched word embeddings. In *2016 IEEE Spoken Language Technology Workshop (SLT)*, pages 414–419.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing crosslingual distributed representations of words. In *Proceedings of COLING 2012*, pages 1459–1474, Mumbai, India. The COLING 2012 Organizing Committee.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *CoRR*, abs/1901.07291.

Pavel Levin, Dhanuka. Nishikant, Talaat Khalil, Fedor Kovalev, and Maxim Khalilov. 2017. Toward a full-scale neural machine translation in production: the Booking.com use case. In *Proceedings of Machine Translation Summit XVI*.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Tal Schuster, Ori Ram, Regina Barzilay, and Amir Globerson. 2019. Cross-lingual alignment of contextual word embeddings, with applications to zero-shot dependency parsing. *CoRR*, abs/1902.09492.

Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. 2012. Practical Bayesian optimization of machine

learning algorithms. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'12, pages 2951–2959, USA. Curran Associates Inc.

Anders Søgaard, Željko Agić, Héctor Martínez Alonso, Barbara Plank, Bernd Bohnet, and Anders Johannsen. 2015. Inverted indexing for cross-lingual NLP. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1713–1722, Beijing, China. Association for Computational Linguistics.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in opus. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey. European Language Resources Association (ELRA).

Shyam Upadhyay, Manaal Faruqui, Gokhan Tur, Dilek Hakkani-Tur, and Larry Heck. 2018. (almost) zero-shot cross-lingual spoken language understanding. In *Proceedings of the IEEE ICASSP*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

# SoftRegex: Generating Regex from Natural Language Descriptions using Softened Regex Equivalence

**Jun-U Park**
Yonsei University
Seoul, Republic of Korea
junupark@yonsei.ac.kr

**Sang-Ki Ko**
Kangwon National University
Kangwon, Republic of Korea
narame7@gmail.com

**Marco Cognetta***
Yonsei University
Seoul, Republic of Korea
cognetta.marco@gmail.com

**Yo-Sub Han**
Yonsei University
Seoul, Republic of Korea
emmous@yonsei.ac.kr

## Abstract

We continue the study of generating semantically correct regular expressions from natural language descriptions (NL). The current state-of-the-art model, SemRegex, produces regular expressions from NLs by rewarding the reinforced learning based on the semantic (rather than syntactic) equivalence between two regular expressions. Since the regular expression equivalence problem is PSPACE-complete, we introduce the EQ_Reg model for computing the similarity of two regular expressions using deep neural networks. Our EQ_Reg model essentially softens the equivalence of two regular expressions when used as a reward function. We then propose a new regex generation model, SoftRegex, using the EQ_Reg model, and empirically demonstrate that SoftRegex substantially reduces the training time (by a factor of at least 3.6) and produces state-of-the-art results on three benchmark datasets.

## 1 Introduction

Regular expressions are an efficient tool to represent structured data with specific rules in a variety of fields such as natural language processing or text classification. However, it is not always an easy task to write an exact regular expression for those who do not have a deep knowledge of regular expressions or when the expression is very complicated, and incorrect or sloppy regular expressions may cause unexpected consequences in practice (Bispo et al., 2006; Zhang et al., 2018). Indeed, even a single character difference between regular expressions can cause them to represent completely different sets of strings. As such, researchers have begun working on a system than generates a regular expression from a natural language description provided by a human while reducing possible errors caused by incorrect regular expressions (Liu et al., 2019). Recently, Locascio et al. (2016) designed the DeepRegex model based on the sequence-to-sequence (Seq2Seq) model (Sutskever et al., 2014) using minimal domain knowledge during the learning phase while still accurately predicting regular expressions from NLs. Later, Zhong et al. (2018a) improved the performance by training on not only syntactic content of the expressions (i.e. the exact textual representation of the expression that was used), but also the semantic content (the regular language described by the expression). However, the reward function in the SemRegex model (Zhong et al., 2018a) that determines if the predicted regular expression is semantically equivalent to the ground truth expression is known to be PSPACE-complete and is a bottleneck in practice (Stockmeyer and Meyer, 1973). Thus, if we can solve this problem (even approximately) more quickly, then we can decrease the required learning time in the natural-language-to-regular expression (NL-RX) model. Reward shaping (Mataric, 1994; Ng et al., 1999) is a well-known mechanism to estimate the reward of an action in reinforced learning without executing the action. As a reward shaping mechanism for NL-RX, we build a new model, EQ_Reg, based on deep learning that estimates the equivalence probability of two regular expressions and use it to improve the NL-RX training speed substantially. During the NL-

*Now at Google.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6425–6431,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

RX model training phase, EQ_Reg can quickly determine the equivalence of the predicted expression and the true expression, and pass this value as a reward for reinforcement learning. Our new NL-RX model together with the EQ_Reg model as a reward function is called SoftRegex. We demonstrate that SoftRegex substantially reduces the training time and produces state-of-the-art results on three benchmark datasets.

## 2 Related Work

**Generating Regular Expressions**: Ranta (1998) studied rule-based techniques for the conversion between multi-languages and regular expressions. Kushman and Barzilay (2013) built a parsing model that translates a natural language sentence to a regular expression, and provided a dataset which is now a popular benchmark dataset for related research. Locascio et al. (2016) proposed the Deep-Regex model based on Seq2Seq for generating regular expressions from natural language descriptions together with a dataset of 10,000 NL-RX pairs. However, due to the limitations of the standard Seq2Seq model, the Deep-Regex model can only generate regular expressions similar in shape to the training data. The SemRegex model improved the Deep-Regex model by using reinforcement learning based on the deterministic finite automata (DFA) equivalence oracle (which determines if two regular expressions describe the same language) as a reward function. This model can generate correct regular expressions that may not resemble the ground truth answer.

**Comparing Regular Expressions**: A regular language can have several syntactically different regular expressions. We say that two regular expressions are equivalent if they both define the same language. The most basic method to deciding whether or not two regular expressions are equivalent is to convert both regular expressions to DFAs. However, the time and space complexity of converting regular expressions to DFAs are both exponential (Hopcroft and Ullman, 1979). Stockmeyer and Meyer (1973) showed that it is PSPACE-complete to decide if two regular expressions generate the same set of words. Thus, deciding equivalence for two regular expressions is a bottleneck calculation even for small inputs.

## 3 Methods

### 3.1 NL-RX Model

We apply the Seq2Seq model with an attention mechanism (Luong et al., 2015). We utilize LSTM (Hochreiter and Schmidhuber, 1997) cells in the Seq2Seq model, which consists of an encoder and decoder. The encoder generates a latent vector from the given natural language description. Concurrently, the decoder receives the latent vector from the encoder and generates an output.

**Maximum Likelihood Estimation (MLE):** Let $\theta$ be all parameters in the Seq2Seq model. The probability that the model outputs regular expressions $R$ from a natural language input $S$ is

$$p_\theta(R \mid S) = \prod_{t=1}^{T} p_\theta(r_t \mid r_{<t}, S). \quad (1)$$

Here, $r_t$ represents a predicted token at time step $t$. The Deep-Regex model trains itself to find the proper $r_t$ using MLE and optimizes through minimizing loss.

**Policy Gradient:** The SemRegex model additionally trains the NL-RX model via a policy gradient (Williams, 1992) by rewarding the model if it generates regular expressions that are semantically equivalent to the ground truth. The reward function returns 1 if the output regular expression is equivalent to the answer, and 0 otherwise. In short, the SemRegex model trains itself to maximize the following objective function:

$$J(\theta) = \sum_{(S,R)\in D} p(R \mid S) r(R), \quad (2)$$

where $D$ is a training set, $p(R \mid S)$ represents the probability that the model predicts regular expression for given sentence, and $r(R)$ is the return value of the reward function that receives the predicted regular expression.

### 3.2 EQ_Reg Model

The reward function in the SemRegex model is based on a regular expression equivalence oracle. This test is time intensive (the regular expression equivalence problem is PSPACE-complete) and returns only a binary value (0/1) representing the equivalence. Therefore, the equivalence test is a major hurdle when training large amounts of data, which is closely related to model performance. Recently, there have been a few attempts to tackle intractable problems using deep learning (Prates

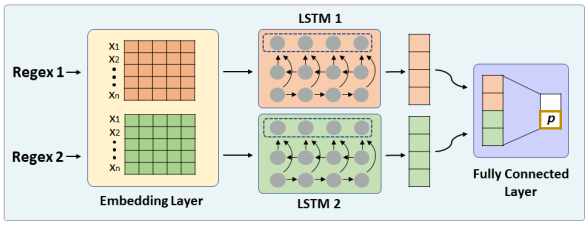


Figure 1: The configuration of the EQ_Reg model

et al., 2019; Selsam et al., 2019). This motivates us to study a different approach that can speed up the equivalence test based on deep learning and propose the EQ_Reg model that returns an equivalence probability of two regular expressions. We can then reward the NL-RX model with continuous values (the equivalence probability of two regular expressions) and significantly boost the learning speed of the NL-RX model using the EQ_Reg model.

The EQ_Reg model converts the input regular expressions to two sequences of embedding vectors. Next, we feed the embedding vector sequences into two different LSTM layers. Each LSTM layer converts the sequences to latent vectors. Since the regular expression is a grammatical expression, we present its context using a bidirectional RNN method (Graves et al., 2013) as

$$\overrightarrow{h}_t = \sigma(W_{x\overrightarrow{h}} x_t + W_{\overrightarrow{h}\overrightarrow{h}} \overrightarrow{h}_{t-1} + b_{\overrightarrow{h}}), \quad (3)$$

$$\overleftarrow{h}_t = \sigma(W_{x\overleftarrow{h}} x_t + W_{\overleftarrow{h}\overleftarrow{h}} \overleftarrow{h}_{t-1} + b_{\overleftarrow{h}}), \quad (4)$$

$$y_t = \sigma(W_{y\overrightarrow{h}} \overrightarrow{h} + W_{y\overleftarrow{h}} \overleftarrow{h}) + b_y, \quad (5)$$

where $\overrightarrow{h}$ is the forward hidden sequence and $\overleftarrow{h}$ is the backward hidden sequence. We concatenate the latent vectors from each LSTM layer into one vector and so that a single vector contains both regular expressions information. Finally, the fully connected layer predicts the equivalence of the two regular expressions from their concatenated vector.

## 4 Experimental Setup

We run our experiments on a computer with the following specifications: AMD Ryzen 7 1700 8-core with 64GB RAM and a GTX 1080Ti GPU on Ubuntu 16.04.1. Our source code is written using PyTorch 0.4.0 (Ketkar, 2017).

### 4.1 EQ_Reg Model

**Datasets:** Locascio et al. (2016) created a set of NL-RX pair data by arbitrarily creating and combining data in a tree form. We define the *depth* of a regular expression in this dataset as the depth of the tree that generated the NL-RX pair (see Appendix A). Similar to Locascio et al. (2016), we randomly generate regular expression pairs up to depth three and label the equivalence between each pair. We sample approximately 200,000 pairs using this method with a ratio of equivalent and non-equivalent pairs of about 2:1. We prepare three sets of data having different depths for test data. One set is made up of only regular expressions with depth at most 3, which is the same as the training data (10,000 pairs). The other two have depths 4 and 5, respectively, and each contain 1,000 pairs of regular expressions of which half are equivalent.

**Model Settings:** We set the two LSTMs in our model to not have shared parameters as we found this gives better performance. The embedding dimension size is set to 4, since the size of vocabulary for regular expressions is relatively small compared to natural languages. We configure two independent LSTMs with 1 layer to receive the two regular expressions. The LSTM layers use the average value of the sequence output values as their final outputs and pass them to the next layer. We set the batch size to 256 and the learning rate to 0.1 and use a stochastic gradient descent optimizer (Bottou, 2010).

### 4.2 SoftRegex Model

**Datasets:** We use three public datasets to compare SoftRegex with the-state-of-the-art model, SemRegex. The KB13 (Kushman and Barzilay, 2013) dataset was constructed by regex experts and is relatively small. On the other hand, NL-RX-Synth is data generated automatically and NL-RX-Turk is made from ordinary people by paraphrasing NL descriptions in NL-RX-Synth using Mechanical Turk (Locascio et al., 2016). Both datasets have 10,000 pairs of NL-RX data. We follow the previous work in splitting the data (train: 65%, dev: 10%, test: 25%).

**Description:** lines having words with lower-case letter at least 5 times or no digit.

**Regular expression (Ground truth):** \b(([a-z]){5,})|(~([0-9])))\b

**Regular expression (Equivalent with ground truth):** ((~([0-9]))|([a-z]){5,}))

Figure 2: An example NL-RX pair and another regular expression that is semantically equivalent to the ground truth.

**Model settings:** We arrange the SoftRegex architecture based on SemRegex. We embed the input sequence tokens with dimension size 128, stack two LSTM layers, and set the hidden state dimension size to 256. We set the batch size to 32 and learning rate to 0.05 with the ADADELTA optimizer (Zeiler, 2012). We substitute in the EQ_Reg model as the reward function which gives high reward value 1 if our model generate a regular expression that is equivalent to the ground truth (Figure 2).

## 5 Results and Analysis

### 5.1 Model Performance

**EQ_Reg:** We test the EQ_Reg model with the three datasets from Section 4.1. The F1-score of test set 1 (depth 3) is 0.986, test set 2 (depth 4) is 0.868, and test set 3 (depth 5) is 0.853. As expected, we can see the model is showing high performance for test data with the same range of depth with the training data. In addition, although the EQ_Reg model has a simple structure, it shows reasonable accuracy for more complex data that has not been previously seen. Here, we can notice the model does not just remember the structure of the training data but generalizes to solve the equivalence problem by understanding the semantics of regular expressions.

**SoftRegex:** Table 1 shows the experimental results of Deep-Regex, SemRegex, and our SoftRegex model. The average accuracy of 10 evaluations is given. The distinguishing test cases method is based on the membership test of samples for the case when an oracle is not available and is described in (Zhong et al., 2018a). The accuracy of SoftRegex is similar to or better than SemRegex (Oracle) and always better than Deep Regex and SemRegex (Distinguishing Test Cases). Figure 3 shows the average training time per epoch over 30 epochs for each of the three models (SoftRegex and both SemRegex variants) and datasets. The training time with EQ_Reg vastly outperforms the training time for SemRegex. In the worst case (the NL-RX-Synth dataset), we see that our new method is still 3.6 times faster than that of SemRegex. Though the speedup described in experimental result may appear constant, our softened equivalence approximately decides a PSPACE-complete problem in linear time to the length of regular expressions, which would otherwise take exponential time.

| Methods | KB13 | NL-RX-Synth | NL-RX-Turk |
|---|---|---|---|
| Deep-Regex | 65.6% | 89.7% | 58.2% |
| SemRegex (Oracle) | 78.2% | 91.6% | 62.3% |
| SemRegex (Distinguishing Test Cases) | 77.5% | 90.2% | 61.3% |
| SoftRegex | 78.2% | 91.4% | 62.8% |

Table 1: The accuracy of NL-RX models.

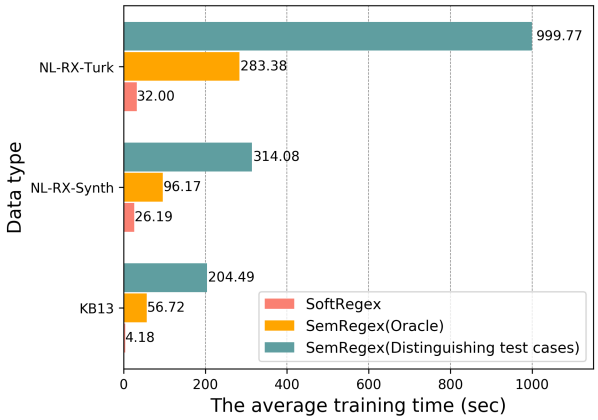


Figure 3: The average training time of NL-RX models.

### 5.2 Error Analysis

Zhong et al. (2018b) pointed out some problems in the NL-RX datasets. Specifically, there are some ambiguities since Locascio et al. (2016) tried to obtain data from machine-generated sentences. Thus, there are situations that even expert humans cannot accurately classify. On the other hand, the NL-RX-Turk dataset is unreliable in that it is generated by non-experts who are paraphrasing previously generated data. We investigate all 921 incorrect predictions of SoftRegex in NL-RX-Turk and categorize the resulting errors into 4 types (Table 2). The 354 type-1 errors are caused by ambiguity in the natural language description where SoftRegex prediction matches one interpretation of the natural language description but not the ground truth. The 256 type-2 errors are from incorrect regex descriptions. These errors occur when descriptions from NL-RX-Synth are misunderstood by the human annotators. The 56 type-3 errors are from incorrect user-generated data (e.g. typos in the descriptions). The remaining 296 type-4 errors are true errors that our model failed to give correct answers for. For the

| NL Example | Predicted Answer | Ground Truth | Error Type |
|---|---|---|---|
| lines containing 3 or more capital letters | .*([[A-Z]]){3,}.* | (.*[A-Z].*){3,} | Description ambiguity (type-1) |
| lines without a capital letter or string 'dog' | ∼(([A-Z])\|(dog)) | (∼[A-Z]))\|((dog)+) | Wrong description (type-2) |
| lines with 5 or more words without characters | \b∼(.)\b | \b∼((.){5,})\b | Typo in description (type-3) |
| lines with words and 4 lower-case letters | \b.*[a-z].*\b | \b([a-z]){4,}\b | True error (type-4) |

Table 2: Example of errors caused by SoftRegex for NL-RX-Turk.

type-1 errors, we need to train more NL-RX data where a single NL is paired with several equivalent regex to cope with the NL ambiguity problem. The type-2 and type-3 errors are caused from crowd-sourcing, which might be hard to detect in the current learning model. We may consider a rule-based prescreening procedure. Finally, for the type-4 errors, we plan to incorporate the copying mechanism (Gu et al., 2016) and the sequence-to-tree translation model (Dong and Lapata, 2016) to enhance the performance of our model considering the deterministic relationship between input and output sequences and hierarchical structure of output regular expressions. Although errors are mostly caused by data errors that we can handle easily, we chose to conduct our experiments under the same condition as SemRegex to provide a fair comparison with prior work.

## 6 Conclusions

Recently, there have been several successful attempts to generate regular expressions from natural language. The current state-of-the-art model is based on reinforced learning using the (in)equivalence of two regular expressions. The regular expression equivalence procedure is a PSPACE-complete problem and, thus, is a major bottleneck in training both in time and space. We sidestep this bottleneck by using the EQ_Reg model as reward shaping, which gives a regex equivalence probability for two regular expressions, and build a new NL-RX model called SoftRegex. Our SoftRegex model with EQ_Reg as a reward function substantially speeds up the training phase (at least 3.6 times faster than SemRegex) while having similar or better performance on a series of standard benchmarks.

## Acknowledgments

Park and Han were supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2018-0-00276).

Ko was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIP) (No. 2017-0-00255).

## References

João Bispo, Ioannis Sourdis, João M. P. Cardoso, and Stamatis Vassiliadis. 2006. Regular expression matching for reconfigurable packet inspection. In *2006 IEEE International Conference on Field Programmable Technology, FPT 2006, Bangkok, Thailand, December 13-15, 2006*, pages 119–126.

Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016*, pages 33–43.

Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. 2013. Hybrid speech recognition with deep bidirectional LSTM. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 273–278.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016*.

Sepp Hochreiter and Jrgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

John E Hopcroft and Jeffry D. Ullman. 1979. *Introduction to automata theory, languages, and computation*.

Nikhil Ketkar. 2017. Introduction to pytorch. In *Deep learning with python*, pages 195–208.

Nate Kushman and Regina Barzilay. 2013. Using semantic unification to generate regular expressions from natural language. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 826–836.

Xiao Liu, Yufei Jiang, and Dinghao Wu. 2019. A lightweight framework for regular expression verification. In *2019 IEEE 19th International Symposium on High Assurance Systems Engineering (HASE)*, pages 1–8.

Nicholas Locascio, Karthik Narasimhan, Eduardo DeLeon, Nate Kushman, and Regina Barzilay. 2016. Neural generation of regular expressions from natural language with minimal domain knowledge. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1918–1923.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Maja J Mataric. 1994. Reward functions for accelerated learning. In *Machine Learning Proceedings 1994*, pages 181–189.

Andrew Y Ng, Daishi Harada, and Stuart Russell. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, pages 278–287.

Marcelo Prates, Pedro HC Avelar, Henrique Lemos, Luis C Lamb, and Moshe Y Vardi. 2019. Learning to solve np-complete problems: A graph neural network for decision tsp. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4731–4738.

Aarne Ranta. 1998. A multilingual natural-language interface to regular expressions. In *Proceedings of the International Workshop on Finite State Methods in Natural Language Processing*, pages 79–90.

Daniel Selsam, Matthew Lamm, Benedikt Bünz, Percy Liang, Leonardo de Moura, and David L. Dill. 2019. Learning a SAT solver from single-bit supervision. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

Larry J Stockmeyer and Albert R Meyer. 1973. Word problems requiring exponential time (preliminary report). In *Proceedings of the fifth annual ACM symposium on Theory of computing*, pages 1–9.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Shanshan Zhang, Lihong He, Slobodan Vucetic, and Eduard Dragut. 2018. Regular expression guided entity mention mining from noisy web data. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1991–2000.

Zexuan Zhong, Jiaqi Guo, Wei Yang, Jian Peng, Tao Xie, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. 2018a. Semregex: A semantics-based approach for generating regular expressions from natural language specifications. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1608–1618.

Zexuan Zhong, Jiaqi Guo, Wei Yang, Tao Xie, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. 2018b. Generating regular expressions from natural language specifications: Are we there yet? In *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*.

## A Supplemental Material

We now describe how Locascio et al. (2016) generated their synthetic regular expression data. To begin, they manually mapped primitive regular expression operations, such as union and concatenation, to natural language. They then defined a small alphabet on which the operations could be performed. In the end, their system has 15 operations and 6 types of characters in the vocabulary. From this, they were able to build up NL-RX pairs automatically by creating a parse tree of repeated applications of operations to an initial regular expression. The operations (non-terminals) and alphabet (terminals) are shown in Table 3.

Figure 4 gives an example of how a NL-RX pair is generated by creating a parse tree. It can be seen that it builds from the bottom up by starting with a series of words or characters (terminals) and applying some primitive operations (non-terminals). Simultaneously, natural language descriptions of terminals and non-terminals are composed together to create a semantically identical natural language description. In this example, the NL-RX pair has depth 2.

| Non-Terminals | | |
|---|---|---|
| x&y → x and y | x\|y → x or y | ∼(x) → not x |
| .*x.*y → x followed by y | .*x.* → contains x | x{N, } → x, N or more times |
| x&y&z → x and y and z | x\|y\|z → x or y or z | x{1, N} → at most N times |
| x.* → starts with x | .*x → ends with x | \bx\b → words with x |
| (x)+ → x, at least once | (x)* → x, zero or more times | x → only x |

| Terminals | | |
|---|---|---|
| [AEIOUaeiou] → a vowel | [0-9] → a number | word → the string word |
| [A-Z] → an uppercase letter | [a-z] → a lowercase letter | . → a character |

Table 3: The non-terminal and terminal operations with their corresponding natural language description as constructed by Locascio et al. (2016).

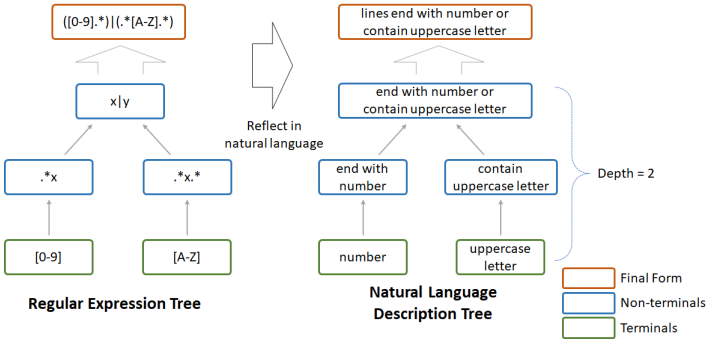


Figure 4: An example of how Locascio et al. (2016) generate the dataset of NL-RX pairs.

# Using Clinical Notes with Time Series Data for ICU Management

**Swaraj Khadanga**
University of Minnesota
Minneapolis, MN
khada004@umn.edu

**Karan Aggarwal**
University of Minnesota
Minneapolis, MN
aggar081@umn.edu

**Shafiq Joty**
Nanyang Technological University
Singapore
srjoty@ntu.edu.sg

**Jaideep Srivastava**
University of Minnesota
Minneapolis, MN
srivasta@umn.edu

## Abstract

Monitoring patients in ICU is a challenging and high-cost task. Hence, predicting the condition of patients during their ICU stay can help provide better acute care and plan the hospital's resources. There has been continuous progress in machine learning research for ICU management, and most of this work has focused on using time series signals recorded by ICU instruments. In our work, we show that adding clinical notes as another modality improves the performance of the model for three benchmark tasks: in-hospital mortality prediction, modeling decompensation, and length of stay forecasting that play an important role in ICU management. While the time-series data is measured at regular intervals, doctor notes are charted at irregular times, making it challenging to model them together. We propose a method to model them jointly, achieving considerable improvement across benchmark tasks over baseline time-series model. Our implementation can be found at https://github.com/kaggarwal/ClinicalNotesICU.

## 1 Introduction

With the advancement of medical technology, patients admitted into the intensive care unit (ICU) are monitored by different instruments on their bedside, which measure different vital signals about patient's health. During their stay, doctors visit the patient intermittently for check-ups and make *clinical notes* about the patient's health and physiological progress. These notes can be perceived as *summarized expert knowledge* about the patient's state. All these data about instrument readings, procedures, lab events, and clinical notes are recorded for reference. Availability of ICU data and enormous progress in machine learning have opened up new possibilities for health care research. Monitoring patients in ICU is a challenging and high-cost task. Hence, predicting the condition of patients during their ICU stay can help plan better resource usage for patients that need it most in a cost-effective way. Prior works (Harutyunyan et al., 2019; Ghassemi et al., 2015; Suresh et al., 2018; Song et al., 2018; Caballero Barajas and Akella, 2015) have focused exclusively on modeling the problem using the time series signals from medical instruments. Expert knowledge from doctor's notes has been ignored in the literature.

In this work, we use clinical notes in addition to the time-series data for improved prediction on benchmark ICU management tasks (Harutyunyan et al., 2019). While the time-series data is measured continuously, the doctor notes are charted at intermittent times. This creates a new challenge to model continuous time series and discrete-time note events jointly. We propose such a multimodal deep neural network that comprises of recurrent units for the time-series and convolution network for the clinical notes. We demonstrate that adding clinical notes improves the performance on in-hospital mortality prediction, modeling decompensation, and length of stay forecasting tasks.

## 2 Related Work

We provide a review of machine learning approaches for clinical prediction tasks.

**Biomedical natural language processing** The recent developments in deep learning-based techniques for NLP have been adapted for clinical notes. Convolutional neural networks have been used to predict ICD codes from clinical texts (Mullenbach et al., 2018; Li et al., 2018; Gangavarapu et al., 2019). Rios and Kavuluru (2015) and Baker et al. (2016) used convolutional neural networks to classify various biomedical articles. Pre-trained word and sentence embeddings have also shown good results for sentence similarity tasks (Chen



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6432–6437,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

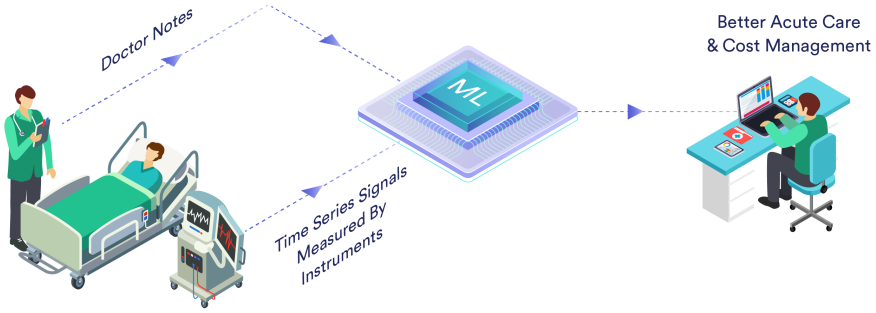


Figure 1: Doctor notes compliments measured physiological signals for better ICU management.

et al., 2018). Recently, there is an interest in the community to use clinical notes for ICU related tasks (Jin et al., 2018; Boag et al., 2018; Liu et al., 2019; Huang et al., 2019). Given the long structured nature of the clinical text, we prefer convolutional neural networks over recurrent networks, as demonstrated in previous studies (Zhang et al., 2016; Boag et al., 2018). The work closest to ours is Jin et al. (2018), who use an aggregated word embeddings of clinical notes for in-hospital mortality prediction.

**ICU management related literature** ICU management literature has focused exclusively on using time-series measurements for the prediction tasks Harutyunyan et al. (2019); Ghassemi et al. (2015); Suresh et al. (2018); Song et al. (2018); Caballero Barajas and Akella (2015). Recurrent neural networks have been models of choice for these recent works, with additional gains from using attention or multi-task learning (Song et al. (2018)). Xu et al. (2018) accommodated supplemental information like diagnosis, medications, and lab events to improve model performance. We use RNNs for modeling time-series in this work, utilizing the setup identical to Harutyunyan et al. (2019).

Multi-modal learning has shown success in speech, natural language, and computer vision (Ngiam et al. (2011), Mao et al. (2014)). Recently, a lot of work has been done using images/videos with natural language text (Elliott et al. (2016)). We use a similar intuition for utilizing clinical notes with time-series data for ICU management tasks. In the next section, we define the three benchmark tasks we evaluate in this work.

## 3 Prediction Tasks

We use the definitions of the benchmark tasks defined by Harutyunyan et al. (2019) as the following three problems:

1. **In-hospital Mortality**: This is a binary classification problem to predict whether a patient dies before being discharged from the first two days of ICU data.

2. **Decompensation**: Focus is to detect patients who are physiologically declining. Decompensation is defined as a sequential prediction task where the model has to predict at each hour after ICU admission. Target at each hour is to predict the mortality of the patient within a 24 hour time window.

3. **Length of Stay Forecasting (LOS)**: The benchmark defines LOS as a prediction of bucketed remaining ICU stay with a multi-class classification problem. Remaining ICU stay time is discretized into 10 buckets: $\{0-1, 1-2, 2-3, 3-4, 4-5, 5-6, 6-7, 7-8, 8-14, 14+\}$ days where first bucket, covers the patients staying for less than a day (24 hours) in ICU and so on. This is only done for the patients that did not die in ICU.

These tasks have been identified as key performance indicators of models that can be beneficial in ICU management in the literature. Most of the recent work has focused on using RNN to model the temporal dependency of the instrument time series signals for these tasks (Harutyunyan et al. (2019), Song et al. (2018)).

## 4 Methods

In this section, we describe the models used in this study. We start by introducing the notations used, then describe the baseline architecture, and finally present our proposed multimodal network.

For a patient's length of ICU stay of $T$ hours, we have time series observations, $x_t$ at each time step $t$ (1 hour interval) measured by instruments along with doctor's note $n_i$ recorded at *irregular* time stamps. Formally, for each patient's ICU

stay, we have time series data $[x_t]_{t=1}^{T}$ of length $T$, and $K$ doctor notes $[N_i]_{i=1}^{K}$ charted at time $[TC(i)]_{i=1}^{K}$, where $K$ is generally much smaller than $T$. For **in-hospital mortality** prediction, $m$ is a binary label at $t = 48$ hours, which indicates whether the person dies in ICU before being discharged. For **decompensation** prediction performed hourly, $[d_t]_{t=5}^{T}$ are the binary labels at each time step $t$, which indicates whether the person dies in ICU within the next 24 hours. For **LOS** forecasting also performed hourly, $[l_t]_{t=5}^{T}$ are multi-class labels defined by buckets of the remaining length of stay of the patient in ICU. Finally, we denote $N_T$ as the concatenated doctor's note during the ICU stay of the patient (*i.e.,*, from $t = 1$ to $t = T$).

### 4.1 Baseline: Time-Series LSTM Model

Our baseline model is similar to the models defined by Harutyunyan et al. (2019). For all the three tasks, we used a Long Short Term Memory or LSTM (Hochreiter and Schmidhuber, 1997) network to model the temporal dependencies between the time series observations, $[x_t]_{t=1}^{T}$. At each step, the LSTM composes the current input $x_t$ with its previous hidden state $h_{t-1}$ to generate its current hidden state $h_t$; that is, $h_t = \text{LSTM}(x_t, h_{t-1})$ for $t = 1$ to $t = T$. The predictions for the three tasks are then performed with the corresponding hidden states as follows:

$$\begin{aligned} \hat{m} &= \text{sigmoid}(W_m h_{48} + b_m) \\ \hat{d}_t &= \text{sigmoid}(W_d h_t + b_d) \;\; \text{for } t = 5 \ldots T \\ \hat{l}_t &= \text{softmax}(W_l h_t + b_l) \;\; \text{for } t = 5 \ldots T \end{aligned} \tag{1}$$

where $\hat{m}$, $\hat{d}_t$, and $\hat{l}_t$ are the probabilities for in-hospital mortality, decompensation, and LOS, respectively, and $W_m$, $W_d$, and $W_l$ are the respective weights of the fully-connected (FC) layer. Notice that the in-hospital mortality is predicted at end of 48 hours, while the predictions for decompensation and LOS tasks are done at each time step after first four hours of ICU stay. We trained the models using cross entropy (CE) loss defined as below.

$$\begin{aligned} \mathcal{L}_{\text{ihm}} &= \text{CE}(m, \hat{m}) \\ \mathcal{L}_{\text{decom}} &= \frac{1}{T} \sum_t \text{CE}(d_t, \hat{d}_t) \\ \mathcal{L}_{\text{los}} &= \frac{1}{T} \sum_t \text{CE}(l_t, \hat{l}_t) \end{aligned} \tag{2}$$

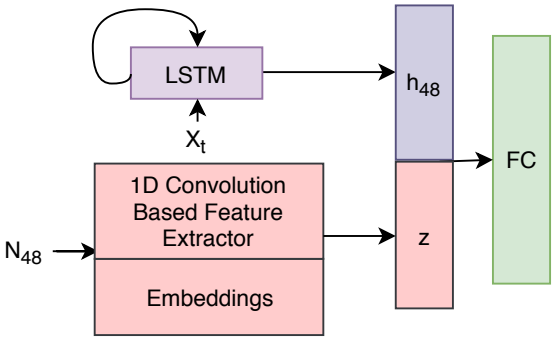


Figure 2: Block diagram from the in-hospital mortality multi-modal network.

### 4.2 Multi-Modal Neural Network

In our multimodal model, our goal is to improve the predictions by taking both the time series data $x_t$ and the doctor notes $n_i$ as input to the network.

**Convolutional Feature Extractor for Doctor Notes.** As shown in Fig. 2, we adopt a convolutional approach similar to Kim (2014) to extract the textual features from the doctor's notes. For a piece of clinical note $N$, our CNN takes the word embeddings $\mathbf{e} = (e_1, e_2, \ldots, e_n)$ as input and applies 1D convolution operations, followed by max-pooling over time to generate a $p$ dimensional feature vector $\hat{z}$, which is fed to the fully connected layer along side the LSTM output from time series signal (described in the next paragraph) for further processing. From now onwards, we denote the 1D convolution over note $N$ as $\hat{z} = \text{Conv1D}(N)$.

**Model for In-Hospital Mortality.** This model takes the time series signals $[x_t]_{t=1}^{T}$ and all notes $[N_i]_{i=1}^{K}$ to predict the mortality label $m$ at $t = T$ ($T = 48$). For this, $[x_t]_{t=1}^{T}$ is processed through an LSTM layer just like the baseline model in Sec. 4.1, and for the notes, we concatenate ($\otimes$) all the notes $N_1$ to $N_K$ charted between $t = 1$ to $t = T$ to generate a single document $N_T$. More formally,

$$\begin{aligned} N_T &= N_1 \otimes N_2 \otimes \cdots \otimes N_K \\ h_t &= \text{LSTM}(x_t, h_{t-1}) \;\; \text{for } t = 1 \ldots T \\ \hat{z} &= \text{Conv1D}(N_T) \\ \hat{m} &= \text{sigmoid}(W_1 h_{48} + W_2 \hat{z} + b) \end{aligned} \tag{3}$$

We use pre-trained word2vec embeddings (Mikolov et al., 2013) trained on both MIMIC-III clinical notes and PubMed articles to initialize our methods as it outperforms other embeddings as shown in (Chen et al., 2018). We also freeze the embedding layer parameters, as we did not observe any improvement by fine-tuning them.

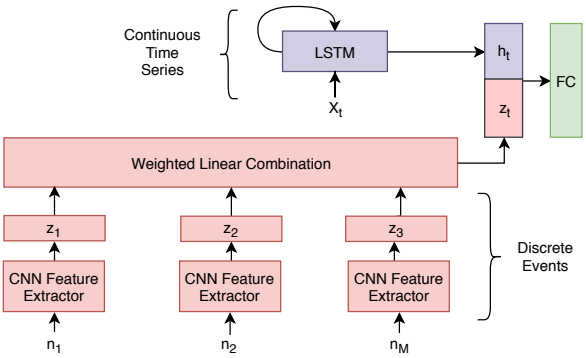


Figure 3: Block diagram from decompensation and length of stay prediction multi-modal network.

**Model for Decompensation and Length of Stay.** Being sequential prediction problems, modeling decompensation and length-of-stay requires special technique to align the discrete text events to continuous time series signals, measured at 1 event per hour. Unlike in-hospital mortality, here we extract feature maps $z_i$ by processing each note $N_i$ independently using 1D convolution operations. For each time step $t = 1, 2 \ldots T$, let $z_t$ denote the extracted text feature map to be used for prediction at time step $t$. We compute $z_t$ as follows.

$$\begin{aligned} z_i &= \text{Conv1D}(N_i) \;\text{ for } i = 1 \ldots K \\ w(t,i) &= \exp[-\lambda * (t - CT(i))] \\ z_t &= \frac{1}{M} \sum_{i=1}^{M} z_i w(t,i) \end{aligned} \tag{4}$$

where $M$ is the number of doctor notes seen before time-step $t$, and $\lambda$ is a decay hyperparameter tuned on a validation data. Notice that $z_t$ is computed as a weighted sum of the feature vectors, where the weights are computed with an exponential decay function. The intuition behind using a decay is to give preference to recent notes as they better describe the current state of the patient.

The time series data $x_t$ is modeled using an LSTM as before. We concatenate the attenuated output from the CNN with the LSTM output for the prediction tasks as follows:

$$\begin{aligned} h_t &= \text{LSTM}(x_t, h_{t-1}) \\ \hat{d}_t &= \text{sigmoid}(W_d^1 h_t + W_d^2 z_t + b) \\ \hat{l}_t &= \text{softmax}(W_l^1 h_t + W_l^2 z_t + b) \end{aligned} \tag{5}$$

Both our baselines and multimodal networks are regularized using dropout and weight decay. We used Adam Optimizer to train all our models.

## 5 Experiments

We used MIMIC-III (Johnson et al., 2016) dataset for all our experiments following Harutyunyan et al. (2019)'s benchmark setup for processing the time series signals from ICU instruments. We use the same test-set defined in the benchmark and 15% of remaining data as validation set. For in-hospital mortality task, only those patients are considered who were admitted in the ICU for at least 48 hours. However, we dropped all clinical notes which doesn't have any chart time associated and also dropped all the patients without any notes. Owing to this step, *our results are not directly comparable to the numbers reported by Harutyunyan et al. (2019).* Notes which have been charted before ICU admission are concatenated and treated as one note at $t = 1$. After preprocessing, the number of patients for in-hospital mortality is 11,579 and 22,353 for the other two tasks.

For in-hospital mortality task, best performing baseline and multimodal network have 256 hidden units LSTM cell. For convolution operation, we used 256 filters for each of kernel size 2, 3 and 4. For decompensation and LOS prediction, we used 64 hidden units for LSTM and 128 filters for each 2,3 and 4 size convolution filters. The best decay factor $\lambda$ for text features was 0.01. We implement our methods with TensorFlow Abadi et al. (2015)[1]. All our models were regularized using 0.2 dropout and 0.01 weight decay coefficient. We run the experiments 5 times with different initialization and report the mean and standard deviations.

## 6 Results

We use Area Under Precision-Recall (AUCPR) metric for in-hospital mortality and decompensation tasks as they suffer from class imbalance with only 10% patients suffering mortality, following the benchmark. Davis and Goadrich (2006) suggest AUCPR for imbalanced class problems. We use Cohen's linear weighted kappa, which measures the correlation between predicted and actual multi-class buckets to evaluate LOS in accordance with with Harutyunyan et al. (2019).

We compared the multimodal network with the baseline time series LSTM models for all three tasks. Results from our experiments are documented in Table 1. Our proposed multimodal net-

[1]The code can be found at https://github.com/kaggarwal/ClinicalNotesICU

**In-Hospital Mortality**

| | AUCROC | AUCPR |
|---|---|---|
| Baseline (No Text) | 0.844 | 0.487 |
| Text-Only | 0.793 | 0.303 |
| MultiModal - Avg WE | 0.851 | 0.492 |
| MultiModal - 1DCNN | **0.865** | **0.525** |

**Decompensation**

| | AUCROC | AUCPR |
|---|---|---|
| Baseline (No Text) | 0.892 | 0.325 |
| Text-Only | 0.789 | 0.081 |
| MultiModal - Avg WE | 0.902 | 0.311 |
| MultiModal - 1DCNN | **0.907** | **0.345** |

**Length of Stay**

| | Kappa |
|---|---|
| Baseline (No Text) | 0.438 |
| Text Only | 0.341 |
| MultiModal - Avg WE | 0.449 |
| MultiModal - 1DCNN | **0.453** |

Table 1: Mean metrics for all three tasks. Standard deviations: IHM (AUCROC $< 0.004$, AUCPR $< 0.015$), Decompensation (AUCROC $< 0.008$, AUCPR $< 0.008$), and LOS (Kappa $< 0.003$).

work outperforms the time series models for all three tasks. For in-hospital mortality prediction, we see an improvement of around 7.8% over the baseline time series LSTM model. The other two problems were more challenging itself than the first task, and modeling the notes for the sequential task was difficult. With our multimodal network, we saw an improvement of around 6% and 3.5% for decompensation and LOS, respectively.

We did not observe a change in performance with respect to results reported in benchmark (Harutyunyan et al., 2019) study despite dropping patients with no notes or chart time. In order to understand the predictive power of clinical notes, we also train text only models using CNN part from our proposed model. Additionally, we try average word embedding without CNN as another method to extract feature from the text as a baseline. Text-only-models perform poorly compared to time-series baseline. Hence, text can only provide additional predictive power on top of time-series data.

## 7 Conclusion

Identifying the patient's condition in advance is of critical importance for acute care and ICU management. Literature has exclusively focused on using time-series measurements from ICU instruments to this end. In this work, we demonstrate that utilizing clinical notes along with time-series data can improve the prediction performance significantly. In the future, we expect to improve more using advanced models for the clinical notes since text summarizes expert knowledge about a patient's condition.

## References


Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Simon Baker, Anna Korhonen, and Sampo Pyysalo. 2016. Cancer hallmark text classification using convolutional neural networks. In *Proceedings of the Fifth Workshop on Building and Evaluating Resources for Biomedical Text Mining (BioTxtM2016)*, pages 1–9.

Willie Boag, Dustin Doss, Tristan Naumann, and Peter Szolovits. 2018. Whats in a note? unpacking predictive value in clinical note representations. *AMIA Summits on Translational Science Proceedings*, 2018:26.

Karla L Caballero Barajas and Ram Akella. 2015. Dynamically modeling patient's health state from electronic medical records: A time series approach. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 69–78. ACM.

Qingyu Chen, Yifan Peng, and Zhiyong Lu. 2018. Biosentvec: creating sentence embeddings for biomedical texts. *arXiv preprint arXiv:1810.09302*.

Jesse Davis and Mark Goadrich. 2006. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM.

Desmond Elliott, Douwe Kiela, and Angeliki Lazaridou. 2016. Multimodal learning and reasoning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*.

Tushaar Gangavarapu, Aditya Jayasimha, Gokul S Krishnan, and Sowmya Kamath. 2019. Tags: Towards automated classification of unstructured clinical nursing notes. In *International Conference on Applications of Natural Language to Information Systems*, pages 195–207. Springer.

Marzyeh Ghassemi, Marco AF Pimentel, Tristan Naumann, Thomas Brennan, David A Clifton, Peter Szolovits, and Mengling Feng. 2015. A multivariate timeseries modeling approach to severity of illness assessment and forecasting in icu with sparse, heterogeneous clinical data. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Hrayr Harutyunyan, Hrant Khachatrian, David C Kale, Greg Ver Steeg, and Aram Galstyan. 2019. Multitask learning and benchmarking with clinical time series data. *Scientific data*, 6(1):96.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Kexin Huang, Jaan Altosaar, and Rajesh Ranganath. 2019. Clinicalbert: Modeling clinical notes and predicting hospital readmission. *arXiv preprint arXiv:1904.05342*.

Mengqi Jin, Mohammad Taha Bahadori, Aaron Colak, Parminder Bhatia, Busra Celikkaya, Ram Bhakta, Selvan Senthivel, Mohammed Khalilia, Daniel Navarro, Borui Zhang, et al. 2018. Improving hospital mortality prediction with medical named entities and multimodal learning. *arXiv preprint arXiv:1811.12276*.

Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3:160035.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

M. Li, Z. Fei, M. Zeng, F. Wu, Y. Li, Y. Pan, and J. Wang. 2018. Automated icd-9 coding via a deep learning approach. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, pages 1–1.

Ning Liu, Pan Lu, Wei Zhang, and Jianyong Wang. 2019. Knowledge-aware deep dual networks for text-based mortality prediction. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 1406–1417. IEEE.

Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, and Alan L Yuille. 2014. Explain images with multimodal recurrent neural networks. *arXiv preprint arXiv:1410.1090*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

James Mullenbach, Sarah Wiegreffe, Jon Duke, Jimeng Sun, and Jacob Eisenstein. 2018. Explainable prediction of medical codes from clinical text. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1101–1111, New Orleans, Louisiana. Association for Computational Linguistics.

Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. 2011. Multimodal deep learning. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 689–696.

Anthony Rios and Ramakanth Kavuluru. 2015. Convolutional neural networks for biomedical text classification: application in indexing biomedical articles. In *Proceedings of the 6th ACM Conference on Bioinformatics, Computational Biology and Health Informatics*, pages 258–267. ACM.

Huan Song, Deepta Rajan, Jayaraman J Thiagarajan, and Andreas Spanias. 2018. Attend and diagnose: Clinical time series analysis using attention models. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Harini Suresh, Jen J Gong, and John V Guttag. 2018. Learning tasks for multitask learning: Heterogenous patient populations in the icu. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 802–810. ACM.

Yanbo Xu, Siddharth Biswal, Shriprasad R Deshpande, Kevin O Maher, and Jimeng Sun. 2018. Raim: Recurrent attentive and intensive model of multimodal patient monitoring data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2565–2573. ACM.

Ye Zhang, Iain Marshall, and Byron C Wallace. 2016. Rationale-augmented convolutional neural networks for text classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 795–804.

# Spelling-Aware Construction of Macaronic Texts for Teaching Foreign-Language Vocabulary

**Adithya Renduchintala** and **Philipp Koehn** and **Jason Eisner**
Center for Language and Speech Processing
Johns Hopkins University
{adi.r,phi}@jhu.edu jason@cs.jhu.edu

## Abstract

We present a machine foreign-language teacher that modifies text in a student's native language (L1) by replacing some word tokens with glosses in a foreign language (L2), in such a way that the student can acquire L2 vocabulary simply by reading the resulting *macaronic* text. The machine teacher uses no supervised data from human students. Instead, to guide the machine teacher's choice of which words to replace, we equip a cloze language model with a training procedure that can incrementally learn representations for novel words, and use this model as a proxy for the word guessing and learning ability of real human students. We use Mechanical Turk to evaluate two variants of the student model: (i) one that generates a representation for a novel word using only surrounding context and (ii) an extension that also uses the spelling of the novel word.

## 1 Introduction

Reading plays an important role in building our native language (L1) vocabulary (Nation, 2001). While some novel words might require the assistance of a dictionary, a large portion are acquired through *incidental learning* – where a reader, exposed to a novel word, tries to infer its meaning using clues from the surrounding context and spelling (Krashen, 1989). An initial "rough" understanding of a novel word might suffice for the reader to continue reading, with subsequent exposures refining their understanding of the novel word.

Our goal is to design a machine teacher that uses a human reader's incidental learning ability to teach foreign language (L2) vocabulary. The machine teacher's modus operandi is to replace L1 words with their L2 glosses, which results in a *macaronic* document that mixes two languages in an effort to ease the human reader into understanding the L2. While some of our prior work (Renduchintala et al., 2016b,a) considered incorporating other features of the L2 such as word order and fixed phrases, in this paper we only consider simple lexical substitutions.

Our hope is that such a system can augment traditional foreign language instruction. As an example, consider a native speaker of English (learning German) presented with the following sentence: **Der** Nile is a **Fluss** in **Afrika**. With a little effort, one would hope the student could infer the meaning of the German words because there is sufficient contextual information and spelling information for the cognate **Afrika**.

In our previous papers on foreign language teaching (Renduchintala et al., 2016b; Knowles et al., 2016; Renduchintala et al., 2017), we focused on fitting detailed models of students' learning when the instructional stimuli (macaronic or otherwise) were chosen by a simple random or heuristic teaching policy. In the present paper, we flip the emphasis to choosing *good* instructional stimuli—machine teaching. This still requires a model of student learning. We employ a reasonable model that is not trained on any human students at all, but only on text that a generic student is presumed to have read. Thus, our model is not personalized, although it may be specialized to the domain of L1 text that it was initially trained on.

That said, our model is reasonably sophisticated and includes new elements. It uses a neural cloze language model (in contrast to the weaker pairwise CRF model of Renduchintala et al. (2016b)) to intelligently guess the meaning of L2 words in full macaronic sentential context. Guessing actually takes the form of a learning rule that jointly improves the embeddings of all L2 words in the sentence. This is our simulation of incidental learning which accumulates over repeated exposures to the same L2 words in different contexts.

Our machine teacher tries to construct macaronic sentences that the human student ought to understand, given all the learning that our generic model predicts would have happened from the previous



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6438–6443,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| Sentence | The | Nile | is | a | river | in | Africa |
|---|---|---|---|---|---|---|---|
| Gloss | **Der** | **Nil** | **ist** | **ein** | **Fluss** | **in** | **Afrika** |
| Macaronic Configurations | **Der** | Nile | **ist** | a | river | in | Africa |
| | The | Nile | is | a | **Fluss** | in | Africa |
| | **Der** | **Nil** | **ist** | **ein** | river | in | Africa |

Table 1: An example English (L1) sentence with German (L2) glosses. Using the glosses, many possible macaronic configurations are possible. Note that the gloss sequence is not a fluent L2 sentence.

macaronic sentences shown to the student. Our teacher does not yet attempt to monitor the human student's *actual* learning. Still, we show that it is useful to a beginner student and far less frustrating than a random (or heuristic based) alternative.

A "pilot" version of the present paper appeared at a recent workshop (Renduchintala et al., 2019): it experimented with three variants of the generic student model, using an artificial L2 language. In this paper, we extend the best of those models to consider an L2 word's spelling (along with its context) when guessing its embeddings. We therefore conduct our experiments on real L2 languages (Spanish and German).

## 2 Related Work

Our motivation is similar to that of commercially available prior systems such as Swych (2015) and OneThirdStories (2018) that also incorporate incidental learning within foreign language instruction. Other prior work (Labutov and Lipson, 2014; Renduchintala et al., 2016b) relied on building a model of the student's incidental learning capabilities, using supervised data that was painfully collected by asking students to react to the actions of an initially untrained machine teacher. Our method, by contrast, constructs a generic student model from unannotated L1 text alone. This makes it possible for us to quickly create macaronic documents in any domain covered by that text corpus.

## 3 Method

Our machine teacher can be viewed as a search algorithm that tries to find the (approximately) best macaronic configuration for the next sentence in a given L1 document. We assume the availability of a "gold" L2 gloss for each L1 word: in our experiments, we obtained these from bilingual speakers using Mechanical Turk. Table 1 shows an example English sentence with German glosses and three possible macaronic configurations (there are exponentially many configurations). The machine teacher must assess, for example, how accurately a student would understand the meanings of **Der**, **ist**, **ein**, and **Fluss** when presented with the following candidate macaronic configuration: **Der** Nile **ist ein Fluss** in Africa.[1] Understanding may arise from inference on this sentence as well as whatever the student has learned about these words from previous sentences. The teacher makes this assessment by presenting this sentence to a generic student model (§§3.1–3.3). It uses a L2 embedding scoring scheme (§3.4) to guide a greedy search for the best macaronic configuration (§3.5).

### 3.1 Generic Student Model

Our model of a "generic student" (GSM) is equipped with a cloze language model that uses a bidirectional LSTM to predict L1 words in L1 context (Mousa and Schuller, 2017; Hochreiter and Schmidhuber, 1997). Given a sentence $\mathbf{x} = [x_1, \ldots, x_t, \ldots, x_T]$, the cloze model defines $p(x_t \mid \mathbf{h}^{\mathbf{f}}{}_t, \mathbf{h}^{\mathbf{b}}{}_t) \forall t \in \{1, \ldots, T\}$, where:

$$\mathbf{h}^{\mathbf{f}}{}_t = \text{LSTM}^{\mathbf{f}}([\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}]; \boldsymbol{\theta}^{\mathbf{f}}) \in \mathbb{R}^D \quad (1)$$

$$\mathbf{h}^{\mathbf{b}}{}_t = \text{LSTM}^{\mathbf{b}}([\mathbf{x}_T, \ldots, \mathbf{x}_{t+1}]; \boldsymbol{\theta}^{\mathbf{b}}) \in \mathbb{R}^D \quad (2)$$

are hidden states of forward and backward LSTM encoders parameterized by $\boldsymbol{\theta}^{\mathbf{f}}$ and $\boldsymbol{\theta}^{\mathbf{b}}$ respectively. The model assumes a fixed L1 vocabulary of size $V$, and the vectors $\mathbf{x}_t$ above are embeddings of these word types, which correspond to the rows of an embedding matrix $\mathbf{E} \in \mathbb{R}^{V \times D}$. The cloze distribution at each position $t$ in the sentence is obtained using

$$p(\cdot \mid \mathbf{h}^{\mathbf{f}}, \mathbf{h}^{\mathbf{b}}) = \text{softmax}(\mathbf{E}\, h([\mathbf{h}^{\mathbf{f}}; \mathbf{h}^{\mathbf{b}}]; \boldsymbol{\theta}^{\mathbf{h}})) \quad (3)$$

where $h(\cdot; \boldsymbol{\theta}^{\mathbf{h}})$ is a projection function that reduces the dimension of the concatenated hidden states from $2D$ to $D$. We "tie" the input embeddings and output embeddings as in Press and Wolf (2017).

We train the parameters $\boldsymbol{\theta} = [\boldsymbol{\theta}^{\mathbf{f}}; \boldsymbol{\theta}^{\mathbf{b}}; \boldsymbol{\theta}^{\mathbf{h}}; \mathbf{E}]$ using Adam (Kingma and Ba, 2014) to maximize $\sum_{\mathbf{x}} \mathcal{L}(\mathbf{x})$, where the summation is over sentences $\mathbf{x}$ in a large L1 training corpus, and

$$\mathcal{L}(\mathbf{x}) = \sum_t \log p(x_t \mid \mathbf{h}^{\mathbf{f}}{}_t, \mathbf{h}^{\mathbf{b}}{}_t) \quad (4)$$

We set the dimensionality of word embeddings and LSTM hidden units to 300. We use the WikiText-103 corpus (Merity et al., 2016) as the L1 training corpus. We apply dropout ($p = 0.2$) between the word embeddings and LSTM layers, and between the LSTM and projection layers (Srivastava et al., 2014). We assume that the resulting model represents the entirety of the student's L1 knowledge.

[1] By "meaning" we mean the L1 token that was originally in the sentence before it was replaced by an L2 gloss.

### 3.2 Incremental L2 Vocabulary Learning

Our generic student model (GSM) supposes that to learn new vocabulary, the student continues to try to improve $\mathcal{L}(\mathbf{x})$ on additional sentences. Thus, if $x_i$ is a new word, the student will try to adjust its embedding to increase all summands of (4), both the $t=i$ summand (making $x_i$ more predictable) and the $t\neq i$ summands (making $x_i$ more predictive of $x_t$).

For our purposes, we do not update $\boldsymbol{\theta}$ (which includes L1 embeddings), as we assume that the student's L1 knowledge has already converged. For the L2 words, we use another word-embedding matrix, $\mathbf{F}$, initialized to $\mathbf{0}^{V'\times D}$, and modify (3) to consider both the L1 and L2 embeddings:

$$p(\cdot \mid [\mathbf{h^f};\mathbf{h^b}]) = \text{softmax}([\mathbf{E};\mathbf{F}]\cdot h([\mathbf{h^f};\mathbf{h^b}];\boldsymbol{\theta^h}))$$

We also restrict the softmax function here to produce a distribution not over the full bilingual vocabulary of size $|V| + |V'|$, but only over the bilingual vocabulary consisting of the L1 types $V$ together with only the L2 types $v' \subset V'$ that actually appear in the macaronic sentence. (In the above example macaronic sentence, $|v'|=4$.) This restriction prevents the model from updating the embeddings of L2 types that are not visible in the macaronic sentence, on the grounds that students are only going to update the meanings of what they are currently reading (and are not even aware of the entire L2 vocabulary).

We assume that when a student reads a macaronic sentence $\mathbf{x}$, they update (only) $\mathbf{F}$ so as to maximize

$$\mathcal{L}(\mathbf{x}) - \lambda \|\mathbf{F} - \mathbf{F^{prev}}\|^2 \tag{5}$$

As mentioned above, increasing the $\mathcal{L}$ term adjusts $\mathbf{F}$ so that the surrounding context can easily predict each L2 word, and each L2 word can, in turn, easily predict the surrounding context (both L1 and L2). However, the penalty term with coefficient $\lambda > 0$ prevents $\mathbf{F}$ from straying too far from $\mathbf{F^{prev}}$, which represents the value of $\mathbf{F}$ before this sentence was read. This limits the degree to which a *single* sentence influences the update to $\mathbf{F}$. As a result, an L2 word's embedding reflects *all* the past sentences that contained that word, not just the most recent such sentence, although with a bias toward the most recent ones, which is realistic. Given a new sentence $\mathbf{x}$, we (approximately) maximize the objective above using 10 steps of gradient ascent (with step-size of 0.1), which gave good convergence in practice. In principle, $\lambda$ should be set based on human-subject experiments. In practice, in this paper, we simply took $\lambda = 1$.

### 3.3 Spelling-Aware Extension

So far, our generic student model ignores the fact that a novel word like **Afrika** is guessable simply by its spelling similarity to Africa. Thus, we augment the generic student model to use character $n$-grams. In addition to an embedding per word type, we learn embeddings for character $n$-gram types that appear in our L1 corpus. The row in $\mathbf{E}$ for a word $w$ is now parameterized as:

$$\tilde{\mathbf{E}}\cdot\tilde{\mathbf{w}} + \sum_n \tilde{\mathbf{E}}^{\mathbf{n}}\cdot\tilde{\mathbf{w}}^n \frac{1}{\mathbf{1}\cdot\tilde{\mathbf{w}}^n} \tag{6}$$

where $\tilde{\mathbf{E}}$ is the full-word embedding matrix and $\tilde{\mathbf{w}}$ is a one-hot vector associated with the word type $w$, $\tilde{\mathbf{E}}^{\mathbf{n}}$ is a character $n$-gram embedding matrix and $\tilde{\mathbf{w}}^n$ is a *multi*-hot vector associated with all the character $n$-grams for the word type $w$. For each $n$, the summand gives the average embedding of all $n$-grams in $w$ (where $\mathbf{1}\cdot\tilde{\mathbf{w}}^n$ counts these $n$-grams). We set $n$ to range from 3 to 4 (see Appendix B). This formulation is similar to previous sub-word based embedding models (Wieting et al., 2016; Bojanowski et al., 2017).

Similarly, the embedding of an L2 word $w$ is parameterized as

$$\tilde{\mathbf{F}}\cdot\tilde{\mathbf{w}} + \sum_n \tilde{\mathbf{F}}^{\mathbf{n}}\cdot\tilde{\mathbf{w}}^n \frac{1}{\mathbf{1}\cdot\tilde{\mathbf{w}}^n} \tag{7}$$

Crucially, we initialize $\tilde{\mathbf{F}}^{\mathbf{n}}$ to $\mu\tilde{\mathbf{E}}^{\mathbf{n}}$ (where $\mu > 0$) so that L2 words can inherit part of their initial embedding from similarly spelled L1 words: $\tilde{\mathbf{F}}^{\mathbf{4}}{}_{\texttt{Afri}} := \mu\tilde{\mathbf{E}}^{\mathbf{4}}{}_{\texttt{Afri}}$.[2] But we allow $\tilde{\mathbf{F}}^{\mathbf{n}}$ to diverge over time in case an $n$-gram functions differently in the two languages. In the same way, we initialize each row of $\tilde{\mathbf{F}}$ to the corresponding row of $\mu\cdot\tilde{\mathbf{E}}$, if any, and otherwise to $\mathbf{0}$. Our experiments set $\mu = 0.2$ (see Appendix B). We refer to this spelling-aware extension to GSM as sGSM.

### 3.4 Scoring L2 embeddings

Did the simulated student learn correctly and usefully? Let $\mathcal{P}$ be the "reference set" of all (L1 word, L2 gloss) pairs from *all tokens in the entire document*. We assess the machine teacher's success by how many of these pairs the simulated student has learned. (The student may even succeed on some pairs that it has never been shown, thanks to $n$-gram clues.) Specifically, we measure the "goodness" of

[2] We set $\mu = 0.2$ based on findings from our hyperparameter search (see Appendix B).

the updated L2 word embedding matrix $\mathbf{F}$. For each pair $p = (e, f) \in \mathcal{P}$, sort all the words in the entire L1 vocabulary according to their cosine similarity to the L2 word $f$, and let $r_p$ denote the rank of $e$. For example, if the student had managed to learn a matrix $\mathbf{F}$ whose embedding of $f$ exactly equalled $\mathbf{E}$'s embedding of $e$, then $r_p$ would be 1. We then compute a mean reciprocal rank (MRR) score of $\mathbf{F}$:

$$\text{MRR}(\mathbf{F}) = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \left( \frac{1}{r_p} \textbf{ if } r_p \leq r_{\text{max}} \textbf{ else } 0 \right) \quad (8)$$

We set $r_{\text{max}} = 4$ based on our pilot study. This threshold has the effect of only giving credit to an embedding of $f$ such that the correct $e$ is in the simulated student's top 4 guesses. As a result, §3.5's machine teacher focuses on introducing L2 tokens whose meaning can be deduced *rather accurately* from their single context (together with any prior exposure to that L2 type). This makes the macaronic text comprehensible for a human student, rather than frustrating to read. In our pilot study we found that $r_{\text{max}}$ substantially improved human learning.

### 3.5 Macaronic Configuration Search

Our current machine teacher produces the macaronic document greedily, one sentence at a time. Actual documents produced are shown in Appendix D.

Let $\mathbf{F}^{\text{prev}}$ be the student model's embedding matrix after the reading the first $n-1$ macaronic sentences. We evaluate a candidate next sentence $\mathbf{x}$ by the score $\text{MRR}(\mathbf{F})$ where $\mathbf{F}$ maximizes (5) and is thus the embedding matrix that the student would arrive at after reading $\mathbf{x}$ as the $n^{\text{th}}$ macaronic sentence.

We use best-first search to seek a high-scoring $\mathbf{x}$. A search state is a pair $(i, \mathbf{x})$ where $\mathbf{x}$ is a macaronic configuration (Table 1) whose first $i$ tokens may be either L1 or L2, but whose remaining tokens are still L1. The state's score is obtained by evaluating $\mathbf{x}$ as described above. In the initial state, $i = 0$ and $\mathbf{x}$ is the $n^{\text{th}}$ sentence of the original L1 document. The state $(i, \mathbf{x})$ is a final state if $i = |\mathbf{x}|$. Otherwise its two successors are $(i+1, \mathbf{x})$ and $(i+1, \mathbf{x}')$, where $\mathbf{x}'$ is identical to $\mathbf{x}$ except that the $(i+1)^{\text{th}}$ token has been replaced by its L2 gloss. The search algorithm maintains a priority queue of states sorted by score. Initially, this contains only the initial state. A step of the algorithm consists of popping the highest-scoring state and, if it is not final, replacing it by its two successors. The queue is then pruned back to the top 8 states. When the queue becomes empty, the algorithm returns the configuration $\mathbf{x}$ from the highest-scoring final state that was ever popped.

| L2 | Model | Closed-class | Open-class |
|---|---|---|---|
| Es | random | 0.74±0.0126(54) | 0.61±0.0134(17) |
| | GSM | 0.72±0.0061(54) | 0.70±0.0084(17) |
| | sGSM | **0.82±0.0038**(41) | **0.80±0.0044**(21) |
| De | random | 0.59±0.0054(34) | 0.38±0.0065(13) |
| | GSM | 0.80±0.0033(34) | 0.78±0.0056(13) |
| | sGSM | **0.82±0.0063**(33) | **0.79±0.0062**(14) |

Table 2: Average token guess quality ($\tau = 0.6$) in the comprehension experiments. The ± denotes a 95% confidence interval computed via bootstrap resampling of the set of human subjects. The % of L1 tokens replaced with L2 glosses is in parentheses. Appendix C evaluates with other choices of $\tau$.

## 4 Evaluation

Does our machine teacher generate useful macaronic text? To answer this, we measure whether *human* students (i) comprehend the L2 words in context, and (ii) retain knowledge of those L2 words when they are later seen without context.

We assess (i) by displaying each successive sentence of a macaronic document to a human student and asking them to guess the L1 meaning for each L2 token $f$ in the sentence. For a given machine teacher, all human subjects saw the same macaronic document, and each subject's comprehension score is the average quality of their guesses on all the L2 tokens presented by that teacher. A guess's quality $q \in [0,1]$ is a thresholded cosine similarity between the embeddings[3] of the guessed word $\hat{e}$ and the original L1 word $e$: $q = \text{cs}(e, \hat{e})$ **if** $\text{cs}(e, \hat{e}) \geq \tau$ **else** 0. Thus, $\hat{e} = e$ obtains $q = 1$ (full credit), while $q = 0$ if the guess is "too far" from the truth (as determined by $\tau$).

To assess (ii), we administer an L2 vocabulary quiz after having human subjects *simply* read a macaronic passage (without any guessing as they are reading). They are then asked to guess the L1 translation of each L2 word type that appeared at least once in the passage. We used the same guess quality metric as in (i).[4] This tests if human subjects naturally learn the meanings of L2 words, in informative contexts, well enough to later translate them out of context. The test requires only short-term retention, since we give the vocabulary quiz immediately after a passage is read.

We compared results on macaronic documents constructed with the generic student model (GSM), its spelling-aware variant (sGSM), and a random

[3] Here we used pretrained word embeddings from Mikolov et al. (2018), in order to measure actual semantic similarity.

[4] If multiple L1 types $e$ were glossed in the document with this L2 type, we generously use the $e$ that maximizes $\text{cs}(e, \hat{e})$.

baseline. In the baseline, tokens to replace are randomly chosen while ensuring that each sentence replaces the same number of tokens as in the GSM document. This ignores context, spelling, and prior exposures as reasons to replace a token.

Our evaluation was aimed at native English (L1) speakers learning Spanish or German (L2). We recruited L2 "students" on Amazon Mechanical Turk (MTurk). They were absolute beginners, selected using a placement test and self-reported L2 ability.

### 4.1 Comprehension Experiments

We used the first chapter of Jane Austen's "Sense and Sensibility" for Spanish, and the first 60 sentences of Franz Kafka's "Metamorphosis" for German. Bilingual speakers provided the L2 glosses (see Appendix A).

For English-Spanish, 11, 8, and 7 subjects were assigned macaronic documents generated with sGSM, GSM, and the random baseline, respectively. The corresponding numbers for English-German were 12, 7 and 7. A total of 39 subjects were used in these experiments (some subjects did both languages). They were given 3 hours to complete the entire document (average completion time was $\approx 1.5$ hours) and were compensated \$10.

Table 2 reports the mean comprehension score over all subjects, broken down into comprehension of function words (closed-class POS) and content words (open-class POS).[5] For Spanish, the sGSM-based teacher replaces *more* content words (but fewer function words), and furthermore the replaced words in both cases are *better understood* on average, which we hope leads to more engagement and more learning. For German, by contrast, the number of words replaced does not increase under sGSM, and comprehension only improves marginally. Both GSM and sGSM do strongly outperform the random baseline. But the sGSM-based teacher only replaces a few additional cognates (**hundert** but not **Mutter**), apparently because English-German cognates do not exhibit large *exact* character $n$-gram overlap. We hypothesize that character skip $n$-grams might be more appropriate for English-German.

### 4.2 Retention Experiments

For retention experiments we used the first 25 sentences of our English-Spanish dataset. New participants were recruited and compensated \$5. Each participant was assigned a macaronic document generated with the sGSM, GSM or random model (20, 18, and 22 participants respectively). As Table 3 shows, sGSM's advantage over GSM on comprehension holds up on retention. On the vocabulary quiz, students correctly translated $> 30$ of the 71 word types they had seen (Table 8), and more than half when near-synonyms earned partial credit (Table 3).

| L2 | Model | Closed-class | Open-class |
|---|---|---|---|
| Es | random | $0.47 \pm 0.0058 (60)$ | $0.40 \pm 0.0041 (46)$ |
| | GSM | $0.48 \pm 0.0084 (60)$ | $0.42 \pm 0.0105 (15)$ |
| | sGSM | $\mathbf{0.52 \pm 0.0054} (47)$ | $\mathbf{0.50 \pm 0.0037} (24)$ |

Table 3: Average type guess quality ($\tau = 0.6$) in the retention experiment. The % of L2 gloss types that were shown in the macaronic document is in parentheses. Appendix C evaluates with other choices of $\tau$.

## 5 Future Work

We would like to explore different character-based compositions such as Kim et al. (2016) that can potentially generalize better across languages. We would further like to extend our work beyond simple lexical learning to allow learning phrasal translations, word reordering, and morphology.

Beyond that, we envision machine teaching interfaces in which the student reader *interacts* with the macaronic text—advancing through the document, clicking on words for hints, and facing occasional quizzes (Renduchintala et al., 2016b)—and with other educational stimuli. As we began to explore in Renduchintala et al. (2016a, 2017), interactions provide feedback that the machine teacher could use to adjust its model of the student's lexicons (here $\mathbf{E}, \mathbf{F}$), inference (here $\boldsymbol{\theta}^{\mathbf{f}}, \boldsymbol{\theta}^{\mathbf{b}}, \boldsymbol{\theta}^{\mathbf{h}}, \mu$), and learning (here $\lambda$). In this context, we are interested in using models that are *student-specific* (to reflect individual learning styles), *stochastic* (since the student's observed behavior may be inconsistent owing to distraction or fatigue), and able to model *forgetting* as well as learning (e.g., Settles and Meeder, 2016).

## 6 Conclusions

We presented a method to generate macaronic (mixed-language) documents to aid foreign language learners with vocabulary acquisition. Our key idea is to derive a model of student learning from only a cloze language model, which uses both context and spelling features. We find that our model-based teacher generates comprehensible macaronic text that promotes vocabulary learning.

[5] https://universaldependencies.org/u/pos/

## References


Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Rebecca Knowles, Adithya Renduchintala, Philipp Koehn, and Jason Eisner. 2016. Analyzing learner understanding of novel L2 vocabulary. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 126–135, Berlin, Germany.

Stephen Krashen. 1989. We acquire vocabulary and spelling by reading: Additional evidence for the input hypothesis. *The Modern Language Journal*, 73(4):440–464.

Igor Labutov and Hod Lipson. 2014. Generating code-switched text for lexical learning. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 562–571, Baltimore, Maryland. Association for Computational Linguistics.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*, Miyazaki, Japan. European Languages Resources Association (ELRA).

Amr Mousa and Björn Schuller. 2017. Contextual bidirectional long short-term memory recurrent neural network language models: A generative approach to sentiment analysis. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1023–1032, Valencia, Spain.

Ian S. P. Nation. 2001. *Learning vocabulary in another language*. Ernst Klett Sprachen.

OneThirdStories. 2018. OneThirdStories. `https://onethirdstories.com/`. Accessed: 2019-02-20.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163, Valencia, Spain.

Adithya Renduchintala, Rebecca Knowles, Philipp Koehn, and Jason Eisner. 2016a. Creating interactive macaronic interfaces for language learning. In *Proceedings of ACL-2016 System Demonstrations*, pages 133–138, Berlin, Germany.

Adithya Renduchintala, Rebecca Knowles, Philipp Koehn, and Jason Eisner. 2016b. User modeling in language learning with macaronic texts. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1859–1869, Berlin, Germany.

Adithya Renduchintala, Philipp Koehn, and Jason Eisner. 2017. Knowledge tracing in sequential learning of inflected vocabulary. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 238–247, Vancouver, Canada.

Adithya Renduchintala, Philipp Koehn, and Jason Eisner. 2019. Simple construction of mixed-language texts for vocabulary learning. In *Proceedings of the 14th Workshop on Innovative Use of NLP for Building Educational Applications (BEA)*, Florence.

Burr Settles and Brendan Meeder. 2016. A trainable spaced repetition model for language learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1848–1858, Berlin, Germany.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Swych. 2015. Swych. `http://swych.it/`. Accessed: 2019-02-20.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Charagram: Embedding words and sentences via character n-grams. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1504–1515, Austin, Texas.

# Towards Machine Reading for Interventions from Humanitarian-Assistance Program Literature

**Bonan Min, Yee Seng Chan, Haoling Qiu,** and **Joshua Fasching**
Raytheon BBN Technologies, Cambridge, Massachusetts
{bonan.min,yeeseng.chan,haoling.qiu,joshua.fasching}@raytheon.com

## Abstract

Solving long-lasting problems such as food insecurity requires a comprehensive understanding of interventions applied by governments and international humanitarian assistance organizations, and their results and consequences. Towards achieving this grand goal, a crucial first step is to extract past interventions and when and where they have been applied, from hundreds of thousands of reports automatically. In this paper, we developed a corpus annotated with interventions to foster research, and developed an information extraction system for extracting interventions and their location and time from text. We demonstrate early, very encouraging results on extracting interventions.

## 1 Introduction

The world is a complex socio-political system: there are long lasting problems such as food insecurity, global warming and diseases affecting much of the world's population, as well as bursting, extreme events such as war, natural disasters and financial crisis. Recently, there has been growing interests in applying event extraction to provide better situation awareness ("what happened"), but rarely in terms of offering insight into providing guidance on what humanitarian assistance interventions have been applied in the past and how they influence the situation.

Furthermore, interventions may have intended outcomes and unintended consequences. An example of an unintended consequence is that free food distribution depresses prices for local produce, and creates a disincentive for farmers. It is extremely useful to automatically extract interventions, their outcome and consequences from hundreds of thousands of articles, to provide to decision makers in governments or humanitarian aid organizations a comprehensive understanding of what intervention options are available when crisis happened, and what outcomes to expect when applying each of them.

This paper is a first step in this direction, starting with developing Information Extraction (IE) techniques to automatically extract interventions from text including academic studies, program/project guidance and evaluation documents from non-profit or international organizations.

We view interventions as a (series of) event(s) [1] with time and space dimensions. For example:

**S1**: *WFP is scaling up its* **food assistance** *activities in Baghdad, Anbar, Dohuk and Ninewa governorates in 2018.*

The intervention "Food Assistance" is happening in the year 2018 in locations "Baghdad, Anbar, Dohuk and Ninewa governorates". Being able to extract interventions and their location and time is a first step towards enabling comprehensive understanding of their effects and consequences.

In this paper, we develop IE techniques towards reading for interventions. The basic methodology is to treat intervention extraction as an event extraction problem, and read intentional and factual statements about interventions in existing project documentation and evaluations literature. Our contributions are three fold:

- We construct a new corpus, annotated with interventions to foster research.
- We develop an IE algorithm for extracting interventions and their locations and time.
- Experiments show the effectiveness of our approach.

We discuss related work in the next section and describe our intervention extraction models in Section 3. In Section 4, we describe our ontology

[1] The main difference between an intervention and other events is that an intervention is a deliberate policy choice, whereas famine or a terrorist attack, for example, is not.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6444–6448, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

of intervention types and the intervention dataset. We present experiment results in Section 5, before concluding in Section 6. The intervention corpus and source code are available at https://github.com/BBN-E/mr-intervention.

## 2 Related Work

Event extraction is often formulated as a multi-stage (Ahn, 2006) classification (trigger classification then argument identification) problem. Prior works either use high-level features (Huang and Riloff, 2012; Ji and Grishman, 2008) or are Neural Network models (Chen et al., 2015). Nguyen (2016) propose joint event extraction using recurrent neural networks.

In need of labeled datasets for training models and evaluation, datasets such as MUC (Grishman and Sundheim, 1996), ACE (Doddington et al., 2004) and Situation Frames (Strassel et al., 2017) have been developed. There are also datasets created for specific domains. An example is the GENIA biomedical event annotation (Kim et al., 2008). Our work is the latest continuation along this path: creating a dataset to foster research in automatically extracting interventions from text, and demonstrating encouraging results.

## 3 Extraction Models

We model interventions as events. Given a sentence, we perform intervention extraction using a two-stage process:

- Trigger classification: Labeling words with their predicted intervention type (if any). For instance, in sentence S1, the extraction system should label "food **assistance**" [2] as a trigger of an intervention type *provide food*.
- Argument classification: If a sentence contains predicted triggers $\{t_i\}$, we pair each $t_i$ with each entity and time mention $\{m_j\}$ in the sentence to generate candidate event arguments. Given a candidate argument $(t_i, m_j)$, the system predicts its associated role (if any). For instance, given the candidate argument ("food assistance", "Baghdad"), it predicts the role $Place$.

To perform event trigger and argument classification, we developed two convolution neural network (CNN) models: one for performing trigger extraction, and one for performing argument extraction. We show the argument model in Figure 1. These models are based on the work of Chen et al. (2015), which achieve competitive performance for event extraction.

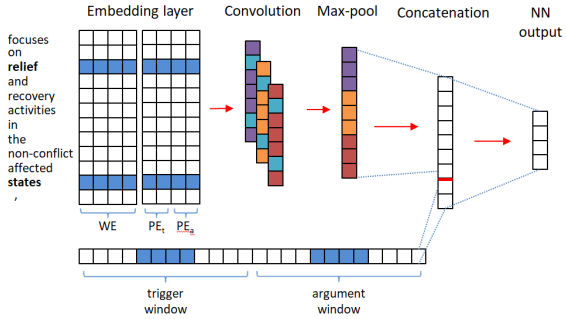


Figure 1: A CNN based model for event argument classification. WE is word embeddings. $PE_t$ and $PE_a$ are position embeddings, capturing a token's distance to the candidate trigger and argument respectively. These position embeddings are randomly initialized and learnt during training.

Our trigger model uses pre-trained word embeddings[3] (Baroni et al., 2014), and learns position embeddings during training (to represent relative distance of each word in the sentence to the candidate trigger). Our argument model uses these, as well as position embeddings relative to the candidate argument, and event embeddings (to represent event type of predicted candidate trigger). The position and event embeddings are randomly initialized and learnt during training.

## 4 Intervention Ontology and Dataset

In this section, we first present the types of interventions that we focus on, then describe a corpus annotated with intervention instances.

### 4.1 Intervention ontology

We focus on modeling interventions or humanitarian assistances that are meant to alleviate mass suffering, improve socioeconomic conditions, and maintain human dignity. We list our intervention ontology in Table 1. Types include promotion of *anti-retroviral* healthcare, promoting respect of *human rights*, ensuring *children friendly learning spaces*, *management of sexual violence*, *therapeutic feeding* of the severely malnourished, *vector control* of insects and pests, and *provision of various humanitarian aid* such as cash, food, etc.

[2] Our model makes predictions on single words, which we then automatically expand to phrases.

[3] EN-wform.w.5.cbow.neg10.400.subsmpl.txt.gz embeddings from the "Don't count, predict" project at http://clic.cimec.unitn.it/composes/semantic-vectors.html

| Intervention Type | Example Snippet |
|---|---|
| anti-retroviral treatment | postpartum ARV *drugs* may also be given to infants |
| capacity building human rights | mission personnel are also engaged in building the capacity of national authorities to promote and *respect* human rights |
| child friendly learning spaces | promotes quality education for indigenous girls and boys through child-friendly learning *environments* |
| provision of goods and services | |
| • provide cash | cash *distributions* during emergencies |
| • provide delivery kit | *distributing* a home delivery kit to every pregnant woman |
| • provide education kit | developing and freely *distributing* education materials |
| • provide farming tool | the scope of the program encompasses *provision* of fertilizer |
| • provide fishing tool | restoration of livelihoods through *provision* of fishing boats and fishing equipment |
| • provide food | food *aid* is often supplied in emergency situations together with seed aid |
| • provide hygiene tool | respond to humanitarian emergencies always aim to *distribute* soap routinely |
| • provide livestock feed | where they were *provided* with fodder |
| • provide seed | food aid is often supplied in emergency situations together with seed *aid* |
| • provide veterinary service | *providing* free or subsidized animal health services |
| sexual violence management | health professionals expected to provide *post-rape care* |
| therapeutic feeding or treating | therapeutic food *provided* in supplementary feeding centers |
| vector control | Malathion is commonly used to *control* mosquitoes |

Table 1: Types of interventions with example text snippets, where event triggers are italicized.

### 4.2 An intervention corpus

State-of-the-art event extraction systems adopt a supervised approach where they learn from a corpus of manually labeled examples that are specific to a predefined event ontology. For instance, the Automatic Content Extraction (ACE) (Doddington et al., 2004) corpus contains more than 500 documents manually annotated with examples for 33 event types. We similarly take a supervised learning approach by collecting and annotating examples for training extraction systems.

Humanitarian assistance programs are associated with various documentation: project proposals, guidances, progress reports, and evaluation reports on program execution. These documents are ideal for mining intervention instances. We collected several hundred documents from the following sources:

- Literature reviews, e.g., the REFANI review[4]. which reviews Cash Transfer Programmes and their impact on malnutrition in humanitarian contexts.
- Programme/project guidelines, e.g., the Sphere Handbook, which lists universal standards in core areas of humanitarian response.
- Evaluation documents, which range from thorough external evaluation of intervention operations[5], to brief presentations of results in programme documents, and post-intervention summary articles.
- Programme documents by implementers[6].
- Academic studies, such as quasi-experimental studies from ebrary.ifpri.org.

### 4.3 Annotating intervention instances

We provided definitions and text examples for the intervention types [7] to two annotators, and then asked them to identify and annotate intervention instances for each document. Annotators are provided with a User Interface (UI) (Chan et al., 2019) which allows them to search for examples efficiently. 30 documents are annotated by two annotators, resulting in an inter-annotator agreement of 0.83.

A total of 976 intervention instances (triggers) are found for the target intervention types. The "Count" column of Table 2 shows numbers of examples for each intervention type.

## 5 Experiments

In this section, we first present experiments in extracting interventions (triggers), and then describe early results on extracting locations and time for interventions.

As shown in Table 1, a large number of the intervention types (e.g. *provide cash*, *provide delivery kit*) have to do with *provision of goods and services*. Although we have kept the labeling of these interventions separate during the annotation process, so that we could optionally perform fine-grained evaluation (and indeed we will later in this section), we found that these interventions share

[4] www.actionagainsthunger.org/sites/default/files/publications/REFANI-lit-review-2015_0.pdf

[5] E.g. bmcnutr.biomedcentral.com/articles/10.1186/s40795-016-0102-6

[6] E.g. one.wfp.org/operations/current_operations/project_docs/200275.pdf

[7] A portion is shown in Table 1.

| Intervention Type | Count | F1-score |
|---|---|---|
| anti-retroviral treatment | 114 | 0.59 |
| capacity building human rights | 65 | 0.68 |
| child friendly learning spaces | 39 | 0.30 |
| provision of goods and services | 432 | 0.77 |
| sexual violence management | 131 | 0.45 |
| therapeutic feeding or treating | 49 | 0.52 |
| vector control | 146 | 0.67 |
| Aggregate | 976 | 0.68 |

Table 2: Intervention types with number of trigger examples and F1-scores based on 5-fold cross validation.

common trigger words (e.g. "provide", "provision", "distribute", etc.) and really rely on their arguments (e.g. "cash", "fertilizer", "fishing boats") for disambiguation. Hence, we perform two sets of trigger classification evaluation: coarse-grained and fine-grained.

### 5.1 Coarse-grained trigger classification

Our annotated trigger examples are spread across 240 documents. We perform 5-fold cross validation to evaluate trigger classification, over the 7 intervention types shown in Table 2. In each fold, we use 20% of the documents as test data and the remainder as training data. We performed minimal hyper-parameters tuning, using 30 epochs and batch size of 40. These were found to achieve good performance in preliminary experiments where we had further split the training data into training and development. We follow (Chen et al., 2015) for the values of the remaining hyper-parameters, e.g. CNN filter size of 3, position and event embeddings of length 5, etc. In our evaluation, a trigger is correctly classified if its intervention event type and offsets match those of a reference trigger. We show the coarse-grained trigger classification scores in the column "F1-score" of Table 2. We obtained a micro-averaged F1-score of 0.68 from the cross validation experiments.

Analysis on the decoding results show that examples vary greatly in terms of difficulty in extracting them. For example, for "sexual violence management", some triggers are phrases such as "post-rape care" that are straightforward for a classifier to recognize, if given sufficient training data. However, there is a long tail of examples where long range dependencies need to be resolved in order to type them correctly. For example, in "**counseling** ... sexual violence" and "clinical **management** ... sexual abuse", the trigger words are often more than 5 tokens far away from additional contextual clues that indicate the target type. We leave modeling these as future work.

### 5.2 Fine-grained trigger classification

As mentioned earlier in this section, we propose that the intervention type *provision of goods and services* rely on their event arguments (artifacts involved) for disambiguation into the finer-grained interventions listed in Table 1. To enable this, we first need to detect mentions of different goods/services in text. We adopt a simple list-based approach, where we manually compiled lists [8] of descriptors for each category. For instance, we use the descriptors ("livestock feed", "fodder", "hay", etc.) for the category *livestock feed*.

Then, when we note that our coarse-grained trigger model had predicted a trigger instance of *provision of goods and services* in a sentence, we check the trigger's surrounding context (5 token window) for mentions of *livestock feed*, *farming tool*, *fishing tool*, etc. We thus deterministically relabel *provision of goods and services* into the appropriate finer-grained intervention type, depending on which category of descriptor is present in the trigger's context window. As shown in Table 2, the coarse-grained F1-score of *provision of goods and services* is 0.77. After performing the deterministic re-labeling into finer-grained intervention types, we obtain an aggregate F1-score of 0.56 when evaluating against our fine-grained trigger labels. Recall misses such as those resulting from incomplete descriptor lists, and precision misses resulting from multiple descriptor categories being present within a trigger's surrounding context, contributed to the drop in F1-score.

### 5.3 Extracting locations and time

We leverage the ACE corpus, which contains annotations of *Place* and *Time* event arguments, to train an event type independent *Place/Time* argument classifier, based on the neural architecture described in Section 3. In our evaluation, an argument is correctly classified if its event type, event argument role, and offsets match any of the reference event arguments.

Africa countries are often the focus sites of humanitarian programs and agencies, such as the World Food Programme (WFP). Hence, to evaluate the performance of our argument model for intervention events, we randomly selected 250 documents from around 6,000 documents collected from `allafrica.com`.

[8] These lists are available at `https://github.com/BBN-E/mr-intervention`

We first apply our coarse-grained trigger classifier on these documents. We then ask annotators to evaluate trigger predictions and retain only correct ones (188 triggers), which we subsequently use to evaluate our argument classifier. We focus on using correct triggers to evaluate argument classification, to avoid error propagation (from erroneous trigger predictions) from muddling a fair assessment of the argument classifier.

Our annotators assigned a total of 15 *Time* arguments and 77 Place arguments to the 188 event triggers. Our argument classifier predicted a total of 12 Time arguments, giving a precision, recall, and F1 of 0.92, 0.73, and 0.81 respectively. Our argument classifier predicted 30 *Place* arguments, giving a precision, recall, and F1 of 0.93, 0.36, and 0.52 respectively.

## 6 Conclusion and Future Work

In this paper, we introduced a new corpus annotated with intervention events, and presented a system that achieves encouraging results.

Our next step is to annotate more documents and make them available to the research community to foster research in this area. We also plan to add *Location* and *Time* annotation on top of the intervention annotation.

## Acknowledgements

The authors would like to thank Ben Watkins and colleagues at Kimetrica LLC for the inspiring discussions on interventions, and providing initial definitions of intervention types in Table 1. The authors also would like to thank the anonymous reviewers for their insightful comments, which helped us to improve the final version of the paper.

This work was supported by DARPA/I2O and U.S. Army Research Office Contract No. W911NF-18-C-0003 under the World Modelers program. The views, opinions, and/or findings contained in this article are those of the author and should not be interpreted as representing the official views or policies, either expressed or implied, of the Department of Defense or the U.S. Government. This document does not contain technology or technical data controlled under either the U.S. International Traffic in Arms Regulations or the U.S. Export Administration Regulations.

## References

David Ahn. 2006. The stages of event extraction. In *ARTE*, pages 1–8, Sydney, Australia. Association for Computational Linguistics.

Marco Baroni, Georgiana Dinu, and German Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *ACL-2014*, pages 238–247, Baltimore, USA.

Yee Seng Chan, Joshua Fasching, Haoling Qiu, and Bonan Min. 2019. Rapid customization for event extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 31–36, Florence, Italy. Association for Computational Linguistics.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *ACL-IJCNLP2-2015*, pages 167–176, Beijing, China. Association for Computational Linguistics.

George R. Doddington, Alexis Mitchell, Mark A. Przybocki, Lance A. Ramshaw, Stephanie M. Strassel, and Ralph M. Weischedel. 2004. The automatic content extraction (ace) program - tasks, data, and evaluation. In *LREC*. European Language Resources Association.

Ralph Grishman and Beth Sundheim. 1996. Message understanding conference-6: A brief history. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*, volume 1.

Ruihong Huang and Ellen Riloff. 2012. Modeling textual cohesion for event extraction. In *AAAI-CAI*, AAAI'12, pages 1664–1670. AAAI Press.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *ACL-HLT-2008*, pages 254–262, Columbus, Ohio. Association for Computational Linguistics.

Jin-Dong Kim, Tomoko Ohta, and Jun'ichi Tsujii. 2008. Corpus annotation for mining biomedical events from literature. *BMC bioinformatics*, 9(1):10.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *NAACL-HLT-2016*, pages 300–309, San Diego, California. Association for Computational Linguistics.

Stephanie M Strassel, Ann Bies, and Jennifer Tracey. 2017. Situational awareness for low resource languages: the lorelei situation frame annotation task. In *SMERP@ ECIR*, pages 32–41.

# RUN through the Streets: A New Dataset and Baseline Models for Realistic Urban Navigation

**Tzuf Paz-Argaman**[1] and **Reut Tsarfaty**[1,2]
[1]Open University of Israel
[2]Allen Institute for Artificial Intelligence
{tzufar,reut.tsarfaty}@gmail.com

## Abstract

Following navigation instructions in natural language requires a composition of language, action, and knowledge of the environment. Knowledge of the environment may be provided via visual sensors or as a symbolic world representation referred to as a *map*. Here we introduce the *Realistic Urban Navigation* (RUN) task, aimed at interpreting navigation instructions based on a real, dense, urban map. Using Amazon Mechanical Turk, we collected a dataset of 2515 instructions aligned with actual routes over three regions of Manhattan. We propose a strong baseline for the task and empirically investigate which aspects of the neural architecture are important for the RUN success. Our results empirically show that *entity abstraction*, *attention over words and worlds*, and a constantly updating *world-state*, significantly contribute to task accuracy.

## 1 Introduction and Background

The task of interpreting and following natural language (NL) navigation instructions involves interleaving different signals, at the very least the linguistic utterance and the representation of the world. For example, in "turn right on the first intersection", the instruction needs to be interpreted, and a specific object in the world (the intersection) needs to be located in order to execute the instruction. In NL navigation studies, the representation of the world may be provided via visual sensors (Misra et al., 2018; Blukis et al., 2018; Nguyen et al., 2018; Yan et al., 2018; Anderson et al., 2018) or as a symbolic world representation. This work focuses on navigation based on a symbolic world representation (referred to as a *map*).

Previous datasets for NL navigation based on a symbolic world representation, HCRC (Anderson et al., 1991; Vogel and Jurafsky, 2010; Levit and Roy, 2007) and SAIL (MacMahon et al., 2006; Chen and Mooney, 2011; Kim and Mooney, 2012, 2013; Artzi and Zettlemoyer, 2013; Artzi et al., 2014; Fried et al., 2017; Andreas and Klein, 2015) present relatively simple worlds, with a small fixed set of entities known to the navigator in advance. Such representations bypass the great complexity of real urban navigation, which consists of long paths and an abundance of previously unseen entities of different types.

In this work we introduce *Realistic Urban Navigation* (RUN), where we aim to interpret navigation instructions relative to a rich symbolic representation of the world, given by a real dense urban map. To address RUN, we designed and collected a new dataset based on OpenStreetMap, in which we align NL instructions to their corresponding routes. Using Amazon Mechanical Turk, we collected 2515 instructions over 3 regions of Manhattan, all specified (and verified) by (respective) sets of humans workers. This task raises several challenges. First of all, we assume a large world, providing long routes, vulnerable to error propagation; secondly, we assume a rich environment, with entities of various different types, most of which are unseen during training and are not known in advance; finally, we evaluate on the full route intended, rather than on last-position only.

We then propose a strong neural baseline for RUN where we augment a standard encoder-decoder architecture with an entity abstraction layer, attention over words and worlds, and a constantly updating world-state. Our experimental results and ablation study show that this architecture is indeed better-equipped to treat *grounding* in realistic urban settings than standard sequence-to-sequence architectures. Given this RUN benchmark, empirical results, and evaluation procedure, we hope to encourage further investigation into the topic of interpreting NL instructions in realistic and previously unseen urban domains.



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6449–6455, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

| | Symbolic World Representation | | | | NL Utterances | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Map** | **Tokens in Map** | **Entities in Map** | **Number of Tiles** | **Size ($km^2$)** | **Paragraphs** | **Instructions** | **Unique Tokens in Paragraphs** | **Avg. Tokens per Instruction** | **Avg. Verbs per Instruction** | **Avg. Actions per Instruction** | **Instruction with advmode before ROOT** | **Avg. Named Entities per Instruction** |
| 1 | 10,353 | 1,612 | 5,457 | 0.51 | 159 | 874 | 735 | 14.42 | 1.53 | 18.71 | 12.81% | 1.93 |
| 2 | 9,829 | 1,134 | 4,935 | 0.46 | 128 | 884 | 727 | 12.46 | 1.31 | 12.81 | 9.16% | 1.32 |
| 3 | 8,844 | 1,051 | 5,452 | 0.51 | 102 | 757 | 654 | 11.86 | 1.29 | 12.44 | 10.3% | 1.44 |
| **Corpus** | 29,026 | 3,797 | 15,844 | 1.48 | 389 | 2515 | **1451** | 12.96 | 1.38 | 13.71 | 10.78% | 1.57 |

Table 1: Data Statistics of RUN: statistics over different maps and the full corpus. The table is divided into features of the symbolic world representation and the written paragraphs.

| **Phenomenon** | **instructions** | **Example from RUN** |
|---|---|---|
| Reference to unnamed entity | 53.33% | Walk to the first **stoplight** and turn left heading south. |
| Reference to named entity | 93.33% | Walk a little more and you will reach your destination on your right: **Fantastic Cafe**. |
| Coreference | 10% | Pass the intersection and **it** will be the second building on your right. |
| Sequencing | 20% | Walk to the **next** intersection, turning right to Avenue B. |
| Count | 23.33% | Walk **5** buildings down the street, and you will see the mcdonalds. |
| Egocentric spatial relation | 26.67% | B&H photo will be immediately **on your right** and that is where you want to be. |
| Imperative | 83.33% | **Go** through the intersection and **follow** the road past the Kmart center. |
| Direction | 66.67% | Make a **right** and go up one block to the light on West 33rd street and 7th Avenue . |
| Condition | 26.67% | we will continue walking **till** we come to the east 7th street intersection. |
| Verification | 20% | On your left toward about the middle of the block **you'll see** Alphabet City. |

Table 2: Linguistic Analysis of RUN: we analyze 30 randomly sampled instructions in RUN. The table characterizes linguistic phenomena in RUN, categorized according to the catalogue presented in Chen et al. (2018).

| | **#Entities** | **#Unique Entities** | **#Tiles** | **Tiles Moved per Sentence\Paragraph** |
|---|---|---|---|---|
| HCRC | *11.93 | *8.125 | *11.93 [1] | n/a*9.75 |
| SAIL | 22 | 0 | 33.33 | 1.3\5.34 |
| TTW | *62.6 | **0 | 25 | n/a\2.5 |
| RUN | 932 | 365 | 1059.6 | 12.2\78.89 |

Table 3: Quantitative Comparison of the HCRC (Anderson et al., 1991), SAIL (MacMahon et al., 2006), TTW (de Vries et al., 2018), and the new RUN Dataset. *We average over three randomly chosen maps. **de Vries et al. (2018) assume perfect perception: all entities at each location are known in advance.

## 2 The RUN Task and Dataset

In this work we address the task of following a sequence of NL navigation instructions given in colloquial language based on a dense urban map.

The *input* to the RUN task we define is as follows: (i) a map with rich details divided into tiles, (ii) an explicit starting point, and (iii) a sequence of navigation instructions we henceforth refer to as a navigation *paragraph* . We refer to each sentence as an *instruction*, and we assume that following the individual instructions in the *paragraph* one by one will lead the agent to the intended end-point. The *output* of RUN is the entire route described by the paragraph, i.e., all coordinates up to and including its end-point, pinned on the map.

In order to address RUN we designed and collected a novel dataset, henceforth the *RUN dataset*, which is based on OpenStreetMap (OSM).[2] The map contains rich layers and an abundance of entities of different types. Each entity is complex and can contain (at least) four labels: name, type, is_building=y/n, and house number. An entity can spread over several tiles. As the maps do not overlap, only very few entities are shared among them. The RUN dataset aligns NL navigation instructions to coordinates of their corresponding route on the OSM map.

We collected the RUN dataset using Amazon Mechanical Turk (MTurk), allowing only native English speakers to perform the task. We collected instructions based on three different areas, all in urban, dense parts of Manhattan. The size of each map is $0.5\ km^2$. The dataset contains 2515 navigation instructions (equal to 389 complete paragraphs) paired with their routes. The paragraphs are crowd-sourced from 389 different instructors, of which style and language vary (Geva et al., 2019).

Our data collection protocol is as follows. First, we asked the MTurk worker to describe a route between two landmarks of their choice. After having described the complete route in NL, the same worker was instructed to pin their described route on the map. This was moderated by showing them the paragraph they narrated, sentence by sentence,

[1] The task defined by Vogel and Jurafsky (2010) is of moving between entities only.

[2] OSM is a free, editable, map of the whole world, that was built by volunteers, with millions of users constantly adding informative tags to the map.

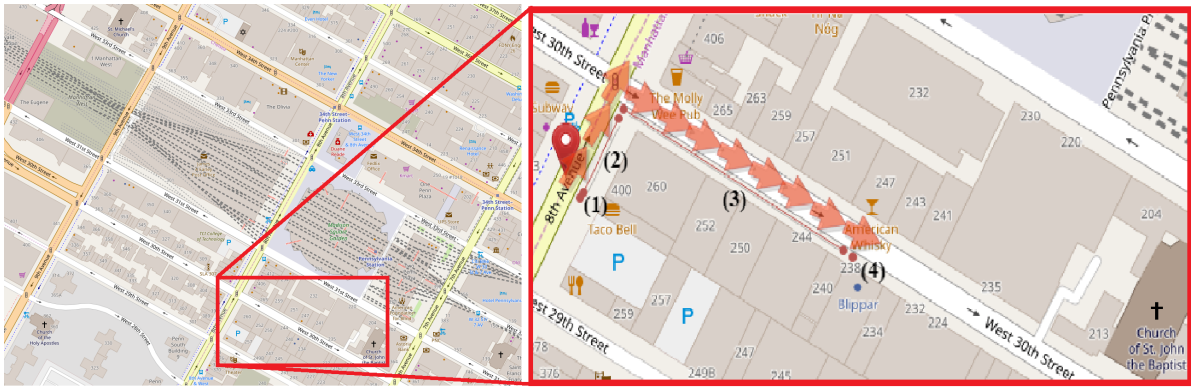


**Instructions:** (1) As you walk out of Taco Bell on 8th Avenue, turn right. (2) Then turn right as you reach the intersection of West 30th Street. (3) Now head down West 30th Street for approximately a half block. (4) You have gone too far if you reach Church of St. John the Baptist.

Figure 1: An Example of a Short Navigation Paragraph: showing a navigation paragraph at the bottom and two images - full map (left image) and a small part of the map (right image). In the full map, many entities are not seen until zoom-in is applied. The navigation paragraph is divided into four sentences: (1) sentence requires a turn action; (2) requires implicit walk actions and an explicit turn; (3) requires walk actions; the last (4) sentence is a verification only and no action is required.

so that they have to pin on the map each instruction separately. A worker could only pin routes on street paths. Furthermore, on every turn the worker had to mark an explicit point on the map which marked the direction in which the worker needs to move next. An example of simple individual instructions and their respective route is given in Figure 1.

We then asked a disjoint set of workers (testers) to verify the routes by displaying the starting point of the route, and displaying the instructions in the paragraph sentence-by-sentence. The tester had to pin the final point of the sentence. Each route was tested by three different workers. Testing the routes allowed us to find incorrect routes (paragraphs that don't match an actual path) and discard them. They also provide an estimate of the human performance on the task (Reported in Section 4, Experiments).

Having collected the data, we divided the map into tiles, each tile is 11.132 m X 11.132 m. Each tile contains the labels of the entities it displays on the map, such as restaurants, traffic-lights, etc., and the walkable streets in it. Each walkable street is composed of an ordered list of tiles, including a starting tile and an end tile. Table 1 shows statistics over the dataset. Table 2 characterize linguistic phenomena in RUN, categorized according to the catalogue of Chen et al. (2018). Table 3 shows a quantitative comparison of the RUN dataset to previous datasets of map-based navigation. The table underscores some key features of RUN, relative to the previous tasks. RUN contains longer paths and many more entities that are unique, thus appearing for the first time during testing; the size of the map is on a different scale than previous tasks, thus, amplifying the grounding challenge; the number of tiles moved is accordingly larger than in previous datasets, hence increasing the vulnerability to error propagation.

Overall, RUN contains challenging linguistic phenomena, at least as in previous work, and a rich environment, with more realistic paths than in previous tasks.

## 3 Models for RUN

We model RUN as a sequence-to-sequence learning problem, where we map a sequence of instructions to a sequence of actions that should be performed in order to pin the actual path on the map. The execution system we provide for the task defines three types of actions: 'TURN', 'WALK', 'END'.[3] 'TURN' is one of the following: right-turn, left-turn, turn-around. The turning move is not necessarily an exact 90-degree turn; the execution system looks for the closest turn option.

[3]RUN has a variation that contains two more types of actions: 'FACE' is a change of direction to face a specific street and end of the street; 'VERIFY' gives verification to the current direction when it is explicitly mentioned in the instructions. For example, "turn right on to 8th Avenue". However, we have been unsuccessful in benefiting from "FACE" and "VERIFY". We leave them for future research.

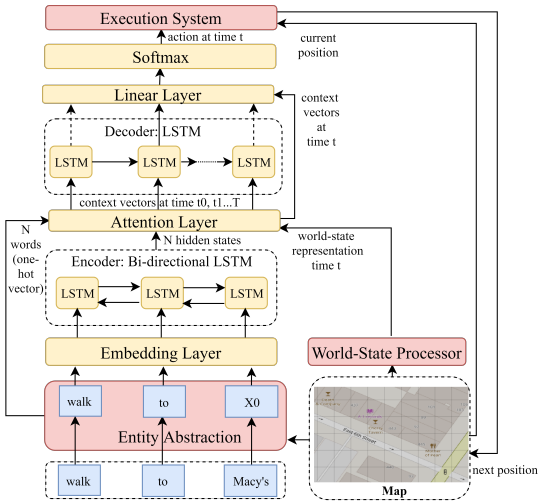


Figure 2: Our Model, Conditioned Generation with Attention over Words and World-States, an Entity Abstraction Layer and an Execution System (CGAEW). The light color (yellow) parts presents a standard Encoder-Decoder with attention, while the dark color (red) are components added on top of a standard CGA.

'WALK' is a change of position in the direction we are facing. The streets can be curved, so 'WALK' is relative to the street that the agent is on. Each street is an ordered-list of tiles, so an action of walking two steps is in fact two actions of 'WALK', in the direction the agent is facing. The 'END' action defines the end of each route. The input consists of an instruction sequence $x_{1:N}$, a map $M$, and a starting point $p_0$ on the map. The output is a sequence of actions $a^*_{1:T}$ to be executed.

$$a^*_{1:T} = arg\max_{a_{1:T}} P(a_{1:T}|x_{1:N}, M, p_0)$$

$$= \arg\max_{a_{1:T}} \prod_{t=1}^{T} P(a_t|a_{1:t-1}, x_{1:N}, M, p_0)$$

Where $x_i$ denote sentences, $a_i$ denotes actions, $M$ is the map and $p_0$ is the starting point.

Our basic model for RUN is a sequence-to-sequence model similar to the work of Mei et al. (2015) on SAIL, and inspired by Xu et al. (2015). It is based on Conditioned Generation with Attention (CGA). To this model we added an Entity abstraction layer (CGAE) and a World-state representation (CGAEW). It thus consists of six components we describe in turn – Encoder, Decoder, Attention, Entity Abstraction, World-State Processor, Execution-System. The complete architecture is depicted in Fig. 2.

**The Encoder** takes the sequence of words that assembles a single sentence and encodes it as a vector using a biLSTM (Graves and Schmidhuber, 2005). **The Decoder** is an LSTM generating a sequence of actions that the execution-system can perform, according to weights defined by an Attention layer. The **Entity Abstraction** component deals with out-of-vocabulary words (OOV). We adopt a similar approach to Iyer et al. (2017); Suhr et al. (2018), replacing phrases in the sentences which refer to previously unseen entities with variables, prior to delivering the sentence to the Encoder. E.g., "Walk from Macy's to 7th street" turns into "Walk from X1 to Y1". Variables are typed (streets, restaurants, etc.) and are numbered based on their order of occurrence in the sentence. The numbering resets after every utterance, so the model remains with a handful of typed entity-variables. The **World-State Processor** maps variables to the entities on the map which are mentioned in the sentence. The world-state representation consists of two vectors, one representing the entities at the current position, and one representing the entities in the path ahead. The **Attention** layer considers the sequence of encoded words *as well as* current world-state, and provides weights on the words for each of the decoder steps. In both training and testing, the **Execution-System** executes each action separately to produce the next position.[4]

## 4 Experiments

We evaluate our model on RUN and assess the contribution of the particular components that we added on top of the standard CGA model.

We train the model using a negative log-likelihood loss, and used Adam (Kingma and Ba, 2014) optimization. For weights initialization we rely on Glorot and Bengio (2010). We used a grid search to validate the hyper-parameters. The model converged at around 30 epochs and produced good results with 0.9 drop-out and a beam of size 4. During inference, we seek the best candidate path using beam-search and normalize the scores of the sequences according to Wu et al. (2016).

We follow the evaluation methodology defined by Chen and Mooney (2011) for SAIL where we use three-fold validation, and in each fold, we use two maps for training (90%) and validation (10%) and test on the third one. We report a sized-

[4]Our code, models, complete maps, annotated dataset, and evaluation: https://github.com/OnlpLab/RUN.

| NO-MOVE | RANDOM | JUMP | CGA | CGAE | CGAEW | HUMAN |
|---|---|---|---|---|---|---|
| 30.3\0.3 | 11.2\0.1 | 26.3\0 | 43.68 (5.93)\0.26 | 46.01 (6.13)\6.17 | 62.37 (3.11)\10.45 | n/a\81.12 |

Table 4: Bounds on Accuracy for Sentences\Paragraphs, weighted averages over folds (std).

| | Sentence | JUMP baseline | CGA | CGAE | CGAEW |
|---|---|---|---|---|---|
| 1 | Just before 9th Avenue, you will see your destination on the right, the West Side Jewish Center. | ✓ | ✗ | ✗ | ✓ |
| 2 | Turn left onto West 34th Street. | ✗ | ✓ | ✓ | ✓ |
| 3 | At the 8th Avenue and West 20th Street intersection, turn right onto West 20th Street. | ✗ | ✗ | ✓ | ✓ |
| 4 | Keep going till you get to the intersection of West 21st Street. | ✗ | ✗ | ✗ | ✓ |
| 5 | Head west on East 7th for 2 (large) blocks; Its a one-way street. | ✗ | ✗ | ✗ | ✗ |

Table 5: Error analysis of all models, for different instructions, showing succeeded / failure on predicting the path.

weighted average test result. For all models we report the accuracy per single sentences and full paragraphs. Success is measured by generating an exact route, not striding away from the path. The last position on the path should be within five tile euclidean distance from the intended destination, as the position explained in the instruction might not be specific enough for one tile.[5] In single-sentences, the last position should also be facing the correct direction.

We provide three simple baselines for the RUN task: (1) NO-MOVE: the only position considered is the starting point; (2) RANDOM: As in Anderson et al. (2018), turn to a randomly selected heading, then execute a number of 'WALK' actions of an average route; (3) JUMP: at each sentence, extract entities from the map and move between them in the order they appear. If the 'WALK' action is invalid we take a random 'TURN' action.

Table 4 shows the results for the baseline models as well as the HUMAN measured performance on the task. The human performance provides an upper bound for the RUN task performance, while the simple baselines provide lower bounds. The best baseline model is NO-MOVE, reaching an accuracy of 30.3% on single sentences and 0.3 on complete paragraphs. For the HUMAN case, paragraph accuracy reaches above 80.

Table 4 shows the results of our model as an ablation study, and Table 5 shows typical errors of each variant. We see that CGAE outperforms CGA, as the swap of entities with variables lowers the complexity of the language that the model needs to learn, allowing the model to effectively cope with unseen entities at test time. We further found that, in many cases, CGAE produces the right type of action, but it does not produce enough of it to reach the intended destination. We attribute these errors to the absence of a world-state representation, resulting in an incapability to ground instructions to specific locations. CGAEW improves upon CGAE as the existence of world-state in the score of the attention layer allows the model better learn the grounding of entities in the instruction to the map. However our best model still fails on features not captured by our world-state: abstract unmarked entities such as blocks, intersections, etc, and generic entities such as traffic-lights (Tab. 5).

[5] We selected this distance as it was the average distance our successful mechanical tester-workers arrived from the intended pinned point.

## 5 Conclusion

We introduce RUN, a new task and dataset for NL navigation in realistic urban environments. We collected (and verified) NL navigation instructions aligned with actual paths, and propose a strong neural baseline for the task. Our ablation studies show the significant contribution of each of the components we propose. In the future we plan to extend the world-state representation, and enable the model to ground *generic* and *abstract* concepts as well. We further intend to add additional signals, for instance coming from vision (cf. Chen et al. (2018)), for more accurate localization.

## 6 Acknowledgments

We thank Yoav Goldberg and Yoav Artzi, for their advice and comments. We thank the ONLP team at the Open University of Israel, for fruitful discussions throughout the process. We further thank the anonymous reviewers for their helpful comments. This research was presented at Georgetown University, Cornell University, and Tel-Aviv University. This research is supported by European Research Council, ERC-StG-2015 scheme, Grant number 677352, and by the Israel Science Foundation (ISF), Grant number 1739/26, for which we are grateful.

## References


Anne H Anderson, Miles Bader, Ellen Gurman Bard, Elizabeth Boyle, Gwyneth Doherty, Simon Garrod, Stephen Isard, Jacqueline Kowtko, Jan McAllister, Jim Miller, et al. 1991. The hcrc map task corpus. *Language and speech*, 34(4):351–366.

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. 2018. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3674–3683.

Jacob Andreas and Dan Klein. 2015. Alignment-based compositional semantics for instruction following. *arXiv preprint arXiv:1508.06491*.

Yoav Artzi, Dipanjan Das, Slav Petrov, et al. 2014. Learning compact lexicons for ccg semantic parsing. In *EMNLP*, pages 1273–1283.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1:49–62.

Valts Blukis, Nataly Brukhim, Andrew Bennett, Ross A Knepper, and Yoav Artzi. 2018. Following high-level navigation instructions on a simulated quadcopter with imitation learning. *arXiv preprint arXiv:1806.00047*.

David L Chen and Raymond J Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *AAAI*, volume 2, pages 1–2.

Howard Chen, Alane Shur, Dipendra Misra, Noah Snavely, and Yoav Artzi. 2018. Touchdown: Natural language navigation and spatial reasoning in visual street environments. *arXiv preprint arXiv:1811.12354*.

Daniel Fried, Jacob Andreas, and Dan Klein. 2017. Unified pragmatic models for generating and following instructions. *arXiv preprint arXiv:1711.04987*.

Mor Geva, Yoav Goldberg, and Jonathan Berant. 2019. Are we modeling the task or the annotator? an investigation of annotator bias in natural language understanding datasets. *arXiv preprint arXiv:1908.07898*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5-6):602–610.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. *arXiv preprint arXiv:1704.08760*.

Joohyun Kim and Raymond J Mooney. 2012. Unsupervised pcfg induction for grounded language learning with highly ambiguous supervision. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 433–444. Association for Computational Linguistics.

Joohyun Kim and Raymond J Mooney. 2013. Adapting discriminative reranking to grounded language learning. In *ACL (1)*, pages 218–227.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Michael Levit and Deb Roy. 2007. Interpretation of spatial language in a map navigation task. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(3):667–679.

Matt MacMahon, Brian Stankiewicz, and Benjamin Kuipers. 2006. Walk the talk: Connecting language, knowledge, and action in route instructions. *Def*, 2(6):4.

Hongyuan Mei, Mohit Bansal, and Matthew R Walter. 2015. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. *arXiv preprint arXiv:1506.04089*.

Dipendra Misra, Andrew Bennett, Valts Blukis, Eyvind Niklasson, Max Shatkhin, and Yoav Artzi. 2018. Mapping instructions to actions in 3d environments with visual goal prediction. *arXiv preprint arXiv:1809.00786*.

Khanh Nguyen, Debadeepta Dey, Chris Brockett, and Bill Dolan. 2018. Vision-based navigation with language-based assistance via imitation learning with indirect intervention. *arXiv preprint arXiv:1812.04155*.

Alane Suhr, Srinivasan Iyer, and Yoav Artzi. 2018. Learning to map context-dependent sentences to executable formal queries. *arXiv preprint arXiv:1804.06868*.

Adam Vogel and Dan Jurafsky. 2010. Learning to follow navigational directions. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 806–814. Association for Computational Linguistics.

Harm de Vries, Kurt Shuster, Dhruv Batra, Devi Parikh, Jason Weston, and Douwe Kiela. 2018. Talk the walk: Navigating new york city through grounded dialogue. *arXiv preprint arXiv:1807.03367*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057.

Claudia Yan, Dipendra Misra, Andrew Bennnett, Aaron Walsman, Yonatan Bisk, and Yoav Artzi. 2018. Chalet: Cornell house agent learning environment. *arXiv preprint arXiv:1801.07357*.

# Context-Aware Conversation Thread Detection in Multi-Party Chat

**Ming Tan** [*†] **Dakuo Wang**[*‡] **Yupeng Gao**[*‡] **Haoyu Wang**[†]
**Saloni Potdar**[†] **Xiaoxiao Guo**[‡] **Shiyu Chang** [‡] **Mo Yu**[‡]
[†]IBM Watson [‡]IBM Research

## Abstract

In multi-party chat, it is common for multiple conversations to occur concurrently, leading to intermingled conversation threads in chat logs. In this work, we propose a novel Context-Aware Thread Detection (CATD) model that automatically disentangles these conversation threads. We evaluate our model on three real-world datasets and demonstrate an overall improvement in thread detection accuracy over state-of-the-art benchmarks.

## 1 Introduction

In multi-party chat conversations, such as in Slack[1], multiple topics are often discussed at the same time (Wang et al., 2019; Zhang et al., 2017). For example, in Table 1, Alice and Bob talk about work, while Bob and Chuck chat about lunch, and this results in intermingled messages. Automatic conversational thread detection could be used to disentangle and group the messages into their respective topic threads. The resulting thread information could in turn be used to improve response relevancy for conversational agents (Shamekhi et al., 2018) or improve chat summarization quality (Zhang and Cranshaw, 2018).

Unlike most of today's Email and Forum systems that use threaded structure by default. However, the Instant Messaging systems (e.g., Slack) often require users to manually organize messages in threads. A recent study (Wang et al., 2019) found that users most likely do not manually create threads. On average, only 15.3 threads were created per Slack channel with 355 messages, when they discuss group projects.

Prior work on conversation thread disentanglement is often based on pairwise message comparison. Some solutions use unsupervised clustering methods with hand-engineered features (Wang and Oard, 2009; Shen et al., 2006), while others use supervised approaches with statistical (Du et al., 2017) or linguistic features (Wang et al., 2008; Wang and Rosé, 2010; Elsner and Charniak, 2008, 2010, 2011; Mayfield et al., 2012). Recent work by (Jiang et al., 2018; Mayfield et al., 2012) adopt deep learning approaches to compute message pair similarity, using a combination of message content and simple contextual features (e.g., authorship and timestamps).

| Thread | Message |
|---|---|
| T1 | Alice: Bob, please send me the proposal. |
| T2 | Bob: Where are we going for lunch? |
| T1 | Bob: Sent. |
| T1 | Alice: Thanks. |
| T2 ? | Chuck: How about the Thai place? |

Table 1: Messages and threads in a chat channel. Does Chuck's message belong to T1, T2, or a new thread?

However, linguistic theories (Biber and Conrad, 2019) differentiate the following three concepts: register, genre and style, to describe the text varieties. Register refers to the linguistic features such as the choice of words in content. Genre and Style refer to the conversational structure such as the sentence sequence and distribution. All aforementioned thread disentanglement methods fail to take into account the *contextual information of the thread*, or the *conversational flow and genre*.

A *thread's contextual information* is a useful feature for thread-detection because considering the relationship between a single new input message and an existing message alone may not be enough to accurately determine thread membership. Hence, using the full thread context history during comparison can improve pre-

[*] Equal contributions from the corresponding authors: mingtan@us.ibm.com, dakuowang@ibm.com, yupeng.gao@ibm.com.

[1] A instant messaging platform: https://slack.com



*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6456–6461, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

diction. Additionally, the *conversational flow and genre* may also be useful because (Butler et al., 2002) suggests this represents a conversation's signature. For example, we observe that users act distinctively in public Q&A (StackOverflow) and enterprise Q&A (IBM Social Q&A) online community (Wang et al., 2016), even when they are answering a similar question. Based on these hypotheses, we propose two context-aware thread detection (**CATD**) models. The first model (**CATD-MATCH**) captures contexts of existing threads and computes the distance between the context and the input message; the second model (**CATD-FLOW**) captures the conversational flow, and computes the language genre consistency while attaching the input message to a thread. We also combine them with a dynamic gate for further performance improvement, followed by an efficient beam search mechanism in the inference step. The evaluation proves our approach improves over the existing methods.

The contribution of this work is two-fold: 1) We propose context-aware deep learning models for thread detection and it advances the state-of-the-art; 2) Based on the dataset in (Jiang et al., 2018), we develop and release a more realistic multi-party multi-thread conversation dataset for future research.

## 2 Methodology

We model thread-detection as a topic detection and tracking task by deciding whether an incoming message starts a new thread or belongs to an existing thread (Allan, 2002). The goal is to assign each message $m_i$ in the sequence, $M = \{m_i\}_{i=1}^{N}$, a thread label $t_i$, such that the complete thread label sequence $T = \{t_i\}_{i=1}^{N}$ contains multiple threads $(\mathcal{T}^1, \mathcal{T}^2, \cdots)$, where each thread $\mathcal{T}^l$ contains all messages with the same label. We denote $\mathcal{T}_{i-1}^{l} = \{m_{\delta(l,k)}\}$ as the $l$-th thread context already detected with $M_{i-1} = \{m_j\}_{j=1}^{i-1}$, and $\delta(l,k)$ is a function returning the index of the *last* $k$-th message of $\mathcal{T}_{i-1}^{l}$ in $M_{i-1}$. $m_{\delta(l,k)}$ is always before $m_i$, but may not be the last message $m_{i-1}$.

The training and inference steps are as follows: we train an LSTM-based thread classification model to obtain the membership of the message $m_i$ to an existing- or a new thread, given the existing message sequence's thread tags $T_{i-1} = \{t_j\}_{j=1}^{i-1}$, which form $L$ threads $\{\mathcal{T}_{i-1}^{l}\}_{l=1}^{L}$ (Subsection 2.1). During inference, we use this model to sequentially perform message thread labelling (Subsection 2.2).

### 2.1 Context-Aware Thread Detection

We first adopt the Universal Sentence Encoder[2] with deep averaging network (USE) (Cer et al., 2018) to get a static feature representation for each message in the form of sentence embeddings. We encode each message $m_j$ as $\mathbf{enc}(m_j)$, by concatenating the USE output with two 20-dimensional embeddings: (1) **User-identity** difference between $m_j$ and $m_i$. (2) **Time difference** by mapping the time difference between $m_j$ and $m_i$ into 11 ranges (from 1 minutes to 72 hours, details in Appendix A). These two features are also used in (Jiang et al., 2018), and another baseline model **GTM** uses only these features (Elsner and Charniak, 2008).

Given a message sequence $M_{i-1}$, which has been detected with $L$ threads $\{\mathcal{T}_{i-1}^{l}\}_{l=1}^{L}$, we predict the thread tag $t_i \in \{\mathcal{T}_{i-1}^{l}\}_{l=1}^{L} \cup \{\mathcal{T}_{i-1}^{L+1}\}$ for $m_i$, where $\mathcal{T}_{i-1}^{L+1}$ indicates that $m_i$ starts a new thread. As shown in Fig.1, we adopt a message-level single directional LSTM to encode each thread $\mathcal{T}_{i-1}^{l}$, whose inputs are $\mathbf{enc}(\cdot)$ of maximum $K$ last messages (set to 20) in the thread, denoted as $\{m_{\delta(l,k)}\}_{k=1}^{K}$. The messages outside that window are viewed as irrelevant to the prediction of $m_i$. In Fig.1, we propose two CATD models, CATD-FLOW and CATD-MATCH, each one capturing the semantic relationship between the new message and the existing thread contexts.

**CATD-FLOW** (left in Fig.1): This model considers each thread as a conversation flow for a particular topic with its own genre, and the current message should belong to the thread with which it is more likely to form a fluent conversation. Therefore, we concatenate the $\mathbf{enc}(m_i)$ to each LSTM sequence of $\mathcal{T}_{i-1}^{l}$, and get the last output $\mathbf{e}_{\mathbf{flow}}^{l}$ to dot-product a trainable vector $\mathbf{w}$ and compute the probability of $m_i$ being labeled with $\mathcal{T}_{i-1}^{l}$,

$$P(t_i = \mathcal{T}_{i-1}^{l} | M_i, T_{i-1}) = \frac{\exp(\gamma \tanh(\mathbf{w} \cdot \mathbf{e}_{\mathbf{flow}}^{l}))}{\sum_{\mathcal{T}_{i-1}^{l'}} \exp(\gamma \tanh(\mathbf{w} \cdot \mathbf{e}_{\mathbf{flow}}^{l'}))} \quad (1)$$

where $\gamma$ is a scaling hyper-parameter. We differentiate $\mathcal{T}_{i-1}^{L+1}$ from other existing threads by introducing a parameter $\mathbf{u}$ as the only input for LSTM.

**CATD-MATCH** (right in Fig.1): An alternative view of determining the thread tag of $m_i$ is to find

[2] https://tfhub.dev/google/universal-sentence-encoder/2

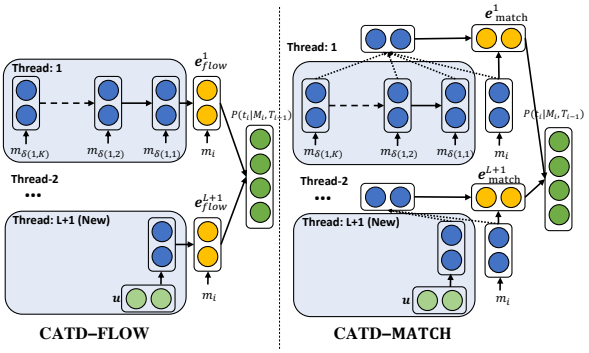


Figure 1: CATD Models: After being encoded by USE with user and time embeddings, ie. $\mathbf{enc}(\cdot)$, the last $K$ messages for each thread $\mathcal{T}^l_{i-1}$ in history are encoded by an LSTM. **CATD-FLOW** concatenates the current message $m_i$ as the final step of each LSTM, and gets the last output of LSTMs $\mathbf{e}^l_{\mathbf{flow}}$ to perform thread classification. **CATD-MATCH** runs LSTM on $\mathcal{T}^l_{i-1}$ and $m_i$ separately, performs attention to obtain the context embeddings and then gets the matching vector $\mathbf{e}^l_{\mathbf{match}}$ for thread classification. For a newly-generated thread, we use a parameter $\mathbf{u}$ as the only input for its LSTM.

a thread $\mathcal{T}^l_{i-1}$ semantically closest to $m_i$. Thus we independently encode each thread and $m_i$ with the parameter-shared LSTM (Only one LSTM step for $m_i$). In order to dynamically point to more related messages in the thread history, we use an one-way attention, which has been successfully adopted in many NLP tasks (Tan et al., 2016; Hermann et al., 2015). Specifically, given the sequence outputs of each thread's LSTM, $\{\mathbf{h}^l_{\delta(l,k)}\}^K_{k=1}$, we perform a weighted mean pooling to get a context embedding $\mathbf{e}^l_{cxt}$, attended by the one-step LSTM-encoded $m_i$, denoted as $\hat{\mathbf{h}}_i$.

$$\alpha^l_k = softmax(\mathbf{h}^l_{\delta(l,k)} \cdot \hat{\mathbf{h}}_i)_{k=[1,K]} \quad (2)$$

$$\mathbf{e}^l_{cxt} = \sum_k \alpha^l_k \mathbf{h}^l_{\delta(l,k)} \quad (3)$$

Next, we compute a matching vector $\mathbf{e}^l_{\mathbf{match}}$ below, which again computes dot-product $\mathbf{w}$ for classification. The function $\mathcal{N}(\mathbf{x})$ normalizes $\mathbf{x}$ by $l_2$ norm. $\otimes$ is element-wise multiplication.

$$\mathbf{e}^l_{\mathbf{match}} = \mathcal{N}(\mathbf{e}^l_{cxt}) \otimes \mathcal{N}(\hat{\mathbf{h}}_i) \quad (4)$$

**CATD-COMBINE:** Finally, we propose the dynamically-gated combination of FLOW and MATCH model, such that the $\mathbf{e}^l_{\mathbf{flow}}$ in Eq. 1 is replaced by a combination of the two models,

$$\mathbf{e}^l_{\mathbf{combine}} = (1-g^l)\mathbf{e}^l_{\mathbf{match}} + g^l \mathbf{e}^l_{\mathbf{flow}} \quad (5)$$

$$g^l = \text{sigmoid}\left(\left|\mathcal{N}(\mathbf{e}^l_{cxt}) - \mathcal{N}(\hat{\mathbf{h}}_i)\right| \cdot \mathbf{w}'\right) \quad (6)$$

where $\mathbf{w}'$ is a parameter vector. $g^l$ is determined by the distance between $\mathcal{N}(\mathbf{e}^l_{cxt})$ and $\mathcal{N}(\hat{\mathbf{h}}_i)$. We use this dynamic gate $g$ to linearly combine the two models. $g$ is computed based on the difference of the MATCH vector of context and the input message. Intuitively, if they are close, both FLOW and MATCH will be considered equally for prediction. Otherwise, the model dynamically computes the weights of MATCH and FLOW.

**Training Procedure:** Following (Jiang et al., 2018), apart from a new thread, we consider the candidate threads (**Active Threads**) in Eq. 1 only from those appearing in one hour time-frame before $m_i$. During training, we treat the messages of a channel as a single sequence, and optimize Eq. 1 with training examples, containing $m_i$ and its active threads. Though messages are sorted by time, the training examples are shuffled during training.

### 2.2 Thread Inference

During inference, we want to find the optimal thread labeling by maximizing:

$$\log P(T|M) = \sum_{i=1}^{N} \log P(t_i|M_i, T_{i-1}) \quad (7)$$

where $t_i$ are selected from active threads and the new thread. However, searching the entire space of $T$ is unfeasible. Hence, we resort to **Beam Search**, a generalized version of greedy search. It predicts sequentially from $m_1$ to $m_N$, while keeping $B$ states in the beam. For each $m_i$, each state in the beam is a candidate $T_{i-1}$. Each new state $T_i$ is ranked after labeling $t_i$ for $m_i$:

$$\max_{\mathcal{T}^l_{i-1}} P(T_i|M_i) = \max_{\mathcal{T}^l_{i-1}} [P(t_i = \mathcal{T}^l_{i-1}|M_i, T_{i-1}) \times P(T_{i-1}|M_{i-1})] \quad (8)$$

where $\mathcal{T}^l_{i-1}$ is selected from the active threads in the previous state and a new thread tag. The new states with scores lower than top-$B$ candidates are discarded. Similar to training, the active threads are also pruned by the "one-hour" constraint. However, they are not extracted from the ground-truth, but from previously-detected threads.

## 3 Experiments

**Datasets:** We conduct extensive experiments on three publicly available datasets from Reddit datasets. We strictly follow (Jiang et al., 2018) to construct our data. Comments under a post can be treated as messages in a single conversational thread, and we merge all comments in a

| | Gadgets | | | Iphones | | | Politics | | |
|---|---|---|---|---|---|---|---|---|---|
| | NMI | ARI | F1 | NMI | ARI | F1 | NMI | ARI | F1 |
| GTM | .662 | .319 | .344 | .685 | .323 | .343 | .798 | .032 | .032 |
| CISIR-SHCNN | .668 | .328 | .359 | .697 | .345 | .364 | .773 | .191 | .196 |
| CISIR-USE | .661 | .334 | .364 | .699 | .333 | .354 | .760 | .182 | .187 |
| CATD-FLOW | .688 | .374 | .400 | .740 | .410 | .424 | .826 | .420 | .423 |
| CATD-MATCH | **.703** | .385 | .404 | .740 | .428 | .441 | .831 | .427 | .430 |
| CATD-COMB. | .694 | **.395** | **.413** | **.750** | **.434** | **.445** | **.834** | **.461** | **.464** |

Table 2: CATD models are compared with baselines wrt. metrics of NMI, ARI and F1 for the three datasets

sub-reddit to construct a synthetic dataset of interleaved conversations. We take three sub-reddits to build three datasets, Gadgets, IPhones and Politics. [3] The data statistics and examples are shown in Appendix B.

**Reddit Dataset Improvement:** We use the same pre-processing method in (Jiang et al., 2018): we discard the messages which have less than 10 words or more than 100 words. Conversations less than 10 messages are also discarded. We guarantee that no more than 10 conversations happen at the same time. In their work, different message pairs of the same thread might be included in both train and test sets. Instead, we split the datasets on the thread level because in realistic settings, test threads should be completely unseen in train set. [4]

**Experimental Setup:** We use Adam (Kingma and Ba, 2015) to optimize the training objective (Eq. 1). During training, we fix $\gamma$ in Eq. 1 as 10. In inference, this value may influence the search quality. We set it as 20.0 by the validation accuracy on Politics. We set LSTM output dimensions to 400, the batch size to 10 and the beam size to 5 by default. We train 50 epochs and select the model with the best validation-set performance.

**Baseline:** (1) CISIR-SHCNN (Jiang et al., 2018): A recently proposed model based on CNN and ranking message pairs. (2) CISIR-USE: We replace CNN encoder in CISIR with a USE to test the effect of different sentence encoders. (3) GTM (Elsner and Charniak, 2008): A graph-theoretical model with chat and content specific features.

[3] We did not use the IRC dataset from (Jiang et al., 2018), because of its extremely small number of data, 39 threads and 491 messages.

[4] This dataset is released in https://github.com/SLAD-ml/thread_detection_data

| | Model Variations | NMI | ARI | F1 |
|---|---|---|---|---|
| A | COMBINE (share) | .832 | .420 | .422 |
| B | COMBINE (concat) | .828 | .446 | .448 |
| C | SPLIT | .824 | .417 | .420 |
| D | FLOW (K=5) | .813 | .395 | .398 |
| E | FLOW (K=10) | .823 | .414 | .417 |
| F | FLOW (K=20) | .826 | .420 | .423 |
| G | MATCH (K=5) | .820 | .399 | .402 |
| H | MATCH (K=10) | .823 | .405 | .408 |
| I | MATCH (K=20) | .831 | .427 | .430 |
| J | MATCH (K=20, bi-LSTM) | .832 | .428 | .430 |
| K | COMBINE (K=5) | .811 | .378 | .381 |
| L | COMBINE (K=10) | .822 | .403 | .405 |
| M | COMBINE (K=20, B=1) | .828 | .452 | .455 |
| N | COMBINE (K=20, B=5) | **.834** | **.461** | **.464** |
| O | COMBINE (K=20, B=10) | .833 | .431 | .433 |

Table 3: Analysis on Politics dataset

**Evaluation Metrics:** Normalized mutual information (NMI), Adjusted rand index (ARI) and F1 score, following (Jiang et al., 2018). F1 is computed based on all message pairs in a test set. Also, following their work, we assume the candidate threads of each message for our models and baselines are obtained from the ones which have messages in the previous hour. For examples, the CISIR-SHCNN models will take pairs only within the one-hour frame.

**Main Results:** Table 2 compares the CATD models and baselines on NMI, ARI and F1. CISIR models are generally better than non-deep-learning GTM. There is a clear gap between CISIR-USE and our proposed models, which proves our models' improvement is not due to the usage of USE but the new model structures. CATD models are significantly superior to all baselines and CATD-COMBINE generally performs best. Specifically, all baselines failed on Politics, probably because there are more threads in Politics than the other two datasets (see Appendix B), making disentanglement more difficult. But CATD models achieve better results because they encode ac-

tive threads in parallel, while considering longer history in each thread.

**Analysis** : In Table 3, we analyze our models on Politics, the largest dataset. First, we examine the effect of $K$. For all CATD models, with $K$ from 5 to 20 (D-F, G-I, K, L and N), and all metrics improve, showing the importance of the longer history in LSTM. Second, we adopt bidirectional LSTMs (J) for CATD-MATCH, without an obvious improvement, probably because most messages in the datasets can be fully comprehended only with previous history. This assumption is consistent with a mild improvement when we increase beam size from 1 (M) to 5 (N). We see a lower ARI with beam size as 10 (O), because of the incorrect candidates at lower ranking positions. Finally, the models are generally good when beam=1, enabling an "online" detection without knowing the future messages, which can not be directly fulfilled by most pairwise prior work.

| | Model Variations | NMI | ARI | F1 |
|---|---|---|---|---|
| A | COMBINE (share) | .699 | .403 | .412 |
| B | COMBINE (concat) | .675 | .366 | .396 |
| C | SPLIT | .692 | .359 | .382 |
| D | FLOW (K=5) | .676 | .368 | .398 |
| E | FLOW (K=10) | .690 | .354 | .386 |
| F | FLOW (K=20) | .688 | .374 | .400 |
| G | MATCH (K=5) | .667 | .366 | .396 |
| H | MATCH (K=10) | .674 | .368 | .399 |
| I | MATCH (K=20) | .703 | .385 | .404 |
| J | MATCH (K=20, bi-LSTM) | **.707** | **.405** | .412 |
| K | COMBINE (K=5) | .686 | .312 | .330 |
| L | COMBINE (K=10) | .687 | .369 | .399 |
| M | COMBINE (K=20, B=1) | .694 | .395 | .411 |
| N | COMBINE (K=20, B=5) | .694 | .395 | .413 |
| O | COMBINE (K=20, B=10) | .692 | .394 | **.414** |

Table 4: Analysis on Gadgets dataset

**Comparison with Model Variations:** In Table 3, we also shared the LSTM parameters for MATCH and FLOW models (A), with 4% drop on ARI. This is because we need two independent LSTMs to capture different linguistic features. Next, we combine FLOW and MATCH (B) by concatenating $\mathbf{e}^l_{\mathbf{flow}}$ and $\mathbf{e}^l_{\mathbf{match}}$, resulting in 1.5% drop on ARI, which proves the benefit of the gate in CATD-COMBINE. Also, we break the links between LSTM nodes and perform one-step LSTM on all the history messages (C), leading to over 4% drop on ARI. This reflects the necessity of a RNN encoding inter-messages information.

In Table 4 and 5, we show the analysis for for Gadgets and Iphones datasets similar to Politics dataset in Table 3. As compared to Politics, we observe that for Gadgets and Iphones, CATD-FLOW models have some fluctuations in performance when we increase $K$ from 5 to 20, which may be due to the limited capability of LSTMs for memorizing long-term history. This issue is more prevalent when the training data size is small.

| | Model Variations | NMI | ARI | F1 |
|---|---|---|---|---|
| A | COMBINE (share) | .748 | .415 | .427 |
| B | COMBINE (concat) | .745 | .419 | .433 |
| C | SPLIT | .749 | .418 | .433 |
| D | FLOW (K=5) | .742 | .411 | .425 |
| E | FLOW (K=10) | .743 | .434 | .446 |
| F | FLOW (K=20) | .740 | .410 | .424 |
| G | MATCH (K=5) | .744 | .385 | .398 |
| H | MATCH (K=10) | .744 | .409 | .420 |
| I | MATCH (K=20) | .740 | .428 | .441 |
| J | MATCH (K=20, bi-LSTM) | .739 | .430 | .445 |
| K | COMBINE (K=5) | .751 | .363 | .374 |
| L | COMBINE (K=10) | .751 | .419 | .429 |
| M | COMBINE (K=20, B=1) | .750 | .431 | .444 |
| N | COMBINE (K=20, B=5) | .750 | .434 | .445 |
| O | COMBINE (K=20, B=10) | **.750** | **.434** | **445** |

Table 5: Analysis on Iphones dataset

## 4 Conclusion

We propose context-aware thread detection models to perform thread detection for multi-party chat conversations which take into account threads' contextual information. These are integrated into an efficient beam search for inference. Our proposed method advances the state-of-the-art.

## References


James Allan. 2002. Introduction to topic detection and tracking. In *Topic detection and tracking*, pages 1–16.

Douglas Biber and Susan Conrad. 2019. *Register, genre, and style*. Cambridge University Press.

Brian Butler, Lee Sproull, Sara Kiesler, and Robert Kraut. 2002. Community effort in online groups: Who does the work and why. *Leadership at a distance: Research in technologically supported work*, 1:171–194.

Daniel Cer, Yinfei Yang, Sheng yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder.

Wenchao Du, Pascal Poupart, and Wei Xu. 2017. Discovering conversational dependencies between messages in dialogs. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Micha Elsner and Eugene Charniak. 2008. You talking to me? a corpus and algorithm for conversation disentanglement. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 834–842.

Micha Elsner and Eugene Charniak. 2010. Disentangling chat. *Computational Linguistics*, 36(3):389–409.

Micha Elsner and Eugene Charniak. 2011. Disentangling chat with local coherence models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1179–1189.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in neural information processing systems*, pages 1693–1701.

Jyun-Yu Jiang, Francine Chen, Yan-Ying Chen, and Wei Wang. 2018. Learning to disentangle interleaved conversational threads with a siamese hierarchical network and similarity ranking. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1812–1822.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations.*

Elijah Mayfield, David Adamson, and Carolyn Penstein Rosé. 2012. Hierarchical conversation structure prediction in multi-party chat. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 60–69. Association for Computational Linguistics.

Ameneh Shamekhi, Q Vera Liao, Dakuo Wang, Rachel KE Bellamy, and Thomas Erickson. 2018. Face value? In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 391. ACM.

Dou Shen, Qiang Yang, Jian-Tao Sun, and Zheng Chen. 2006. Thread detection in dynamic text message streams. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, pages 35–42, New York, NY, USA. ACM.

Ming Tan, Cicero Dos Santos, Bing Xiang, and Bowen Zhou. 2016. Improved representation learning for question answer matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 464–473.

Dakuo Wang, Youyang Hou, Lin Luo, and Yingxin Pan. 2016. Answerer engagement in an enterprise social question & answering system. *IConference 2016 Proceedings*.

Dakuo Wang, Haoyu Wang, Mo Yu, Zahra Ashktorab, and Ming Tan. 2019. Slack channels ecology in enterprises: How employees collaborate through group chat. *arXiv preprint arXiv:1906.01756*.

Lidan Wang and Douglas W Oard. 2009. Context-based message expansion for disentanglement of interleaved text conversations. In *Proceedings of human language technologies: The 2009 annual conference of the North American chapter of the association for computational linguistics*, pages 200–208. Association for Computational Linguistics.

Yi-Chia Wang, Mahesh Joshi, William W Cohen, and Carolyn Penstein Rosé. 2008. Recovering implicit thread structure in newsgroup style conversations. In *ICWSM*.

Yi-Chia Wang and Carolyn P Rosé. 2010. Making conversational structure explicit: identification of initiation-response pairs within online discussions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 673–676. Association for Computational Linguistics.

Amy Zhang, Bryan Culbertson, and Praveen Paritosh. 2017. Characterizing online discussion using coarse discourse sequences. *11th AAAI International Conference on Web and Social Media (ICWSM)*.

Amy X Zhang and Justin Cranshaw. 2018. Making sense of group chat through collaborative tagging and summarization. *Proceedings of the ACM on Human-Computer Interaction*, 2(CSCW):196.

# Author Index


Abburi, Harika, 1642
Abdou, Mostafa, 5838
Agarwal, Arvind, 1980
Agashe, Rajas, 5436
Aggarwal, Karan, 6432
Aggarwal, Samarth, 6262
Agrawal, Sweta, 1549
Aharonov, Ranit, 5591, 5625
Ahn, Emily, 1664
Aji, Alham Fikri, 3626
Ajjour, Yamen, 2922
Akhtar, Md Shad, 5647
Akimoto, Kosuke, 6225
Akkalyoncu Yilmaz, Zeynep, 3490
Alberti, Chris, 2131
Aldarmaki, Hanan, 3672
Allen, Carl, 5185
Almarwani, Nada, 3672
Alnajjar, Khalid, 5999
Alqahtani, Sawsan, 1442
Alshomary, Milad, 2922
Alsop, Anna, 2877
Altun, Yasemin, 5902
Amplayo, Reinald Kim, 5602
AN, Aixiu, 2888
An, Larry, 1255
An, Mingxiao, 4874
Ananiadou, Sophia, 3679, 4925
Anastasopoulos, Antonios, 984, 1192, 1417
Andor, Daniel, 5947
Andrews, Nicholas, 1684
Androutsopoulos, Ion, 6005
Anjum, Omer, 518
Anvesh Rao, Vijjini, 5658
Ao, Xiang, 3033, 6280
Apidianaki, Marianna, 6005
Arabshahi, Forough, 1129
Aralikatte, Rahul, 1229
Arase, Yuki, 5393
Archer, Amelia, 3632
Arivazhagan, Naveen, 3632
Arnold, Josh, 143
Arora, Rishi, 6071
Artzi, Yoav, 2119
Arvidsson, Martin, 6323
Asher, Nicholas, 2296
Assael, Yannis, 6368
Atari, Mohammad, 5753
Atkinson, David, 2911
Augenstein, Isabelle, 4685
Auli, Michael, 5360, 5696
Aw, Ai Ti, 942
Aw, AiTi, 2251
Awasthi, Abhijeet, 4260
Azab, Mahmoud, 1255

Badene, Sonia, 2296
Badjatiya, Pinkesh, 1642
Baevski, Alexei, 5360
Bagher Zadeh, AmirAli, 2046
Bai, Jiaxin, 5247
Bai, Kun, 1824
Bai, Shaojie, 4344
Bak, JinYeong, 1941
Bakhtin, Anton, 2463
Bakovic, Eric, 6176
Balakrishnan, Anusha, 1951
Balashankar, Ananth, 2338
Balasubramanian, Niranjan, 780
Balazevic, Ivana, 5185
Baldridge, Jason, 3687
Baldwin, Timothy, 1729
Balkir, Esma, 3591
Bansal, Mohit, 1317, 2495, 2553, 4474, 5100
Bansal, Srijan, 6336
Bao, Zuyi, 1028
Bapna, Ankur, 1538, 1565
Barbosa, Denilson, 749
Barke, Shraddha, 6176
Barker, Ken, 3884, 4864
Barrett, Maria, 6330
Barrón-Cedeño, Alberto, 5636
Barron, Jonathan, 2195
Barzilay, Regina, 3419, 5714
Basili, Roberto, 4037
Batra, Dhruv, 1449
Bedrax-Weiss, Tania, 2380

Beigi, Ghazaleh, 2360
Belding, Elizabeth, 4755
Belongie, Serge, 708
Beltagy, Iz, 3615, 3693
Ben Noach, Matan, 31
Bengio, Yoshua, 2796
Bentivogli, Luisa, 1368
Berant, Jonathan, 1161, 3659, 3810
Berg-Kirkpatrick, Taylor, 2195, 3603, 6025
Bergen, Leon, 6176
Bethard, Steven, 2578
Bethke, Anna, 4755
Bhagavatula, Chandra, 2391, 5043
Bhardwaj, Sangnie, 6262
Bhardwaj, Vikas, 1355
Bhat, Suma, 518
Bhatia, Rhea, 5753
Bhattacharyya, Pushpak, 5647
Bi, Bin, 2521
bi, tianchi, 856
Bi, Wei, 1866, 1898
Bian, Jiang, 337
Bian, Shuqing, 4810
Bilu, Yonatan, 5591
Bindel, David, 4991
Bing, Lidong, 198, 250, 1007, 1909, 3216, 4590, 4937, 5614
Bishop, Marcus, 1684
Bisk, Yonatan, 1494
Black, Alan W, 4947
Blanco, Eduardo, 663
Blasi, Damian, 5734
Blix, Hagen, 2877
Blunsom, Phil, 4303
Boggs, Thomas, 4731
Bogin, Ben, 3659
Bogoychev, Nikolay, 3626
Böhm, Florian, 3110
Bolukbasi, Tolga, 1167
Bordes, Antoine, 4186
Bordes, Patrick, 696
Bordia, Shikha, 2877
Bornea, Mihaela, 3884
Bosselut, Antoine, 4496, 5043, 6076
Boureau, Y-Lan, 1951, 5872
Bowman, Samuel R., 2877, 3342
Boyd-Graber, Jordan, 1243, 5918
Bozzon, Alessandro, 2349
Braud, Chloé, 4186
Breitfeller, Luke, 1664
Brennan, Jonathan, 5846
Brockett, Chris, 1814, 3304
Budzianowski, Paweł, 1273
Bui, Trung, 5953
Burachas, Giedrius, 5860
Busbridge, Daniel, 77
Buys, Jan, 3752
Byrne, Bill, 1273, 3356, 4516

Cai, Deng, 1866, 3799, 6362
Cai, Hengyi, 1793
Cai, Jie, 3438
Cai, Keke, 4601
Cai, Pengshan, 6155
Cai, Rui, 1018
Cai, Yi, 357, 3426
Cai, Zheng, 486
Camburu, Oana-Maria, 4303
Campos, Daniel, 5175
Cannaviccio, Matteo, 749
Cao, Lei, 1718
Cao, Meng, 3163
Cao, Wei, 337
Cao, Yixin, 261, 2723
Cao, Yu, 2877
Caragea, Cornelia, 2317, 6299
Carbonell, Jaime, 5164, 6025
Card, Dallas, 2185
Cardie, Claire, 5668
Carenini, Giuseppe, 2306, 3011, 3471
Carin, Lawrence, 3132
Carley, Kathleen, 4732, 5469
Carpenter, Ada, 2954
Carpuat, Marine, 1549, 5285
Carton, Samuel, 3968
Cases, Ignacio, 4485
Castro Ferreira, Thiago, 552
Caswell, Isaac, 1565
Cattle, Andrew, 6376
Cecchi, Guillermo, 4026
Cedilnik, Andy, 4516
Celikyilmaz, Asli, 1494
Cen, Yukuo, 1803
Chakrabarty, Tuhin, 2933
Chakraborty, Sunandan, 2338
Chan, Yee Seng, 6444
Chan, Zhangming, 1931, 3741, 4959
Chang, Baobao, 3388
Chang, Jinxin, 1920
Chang, Jonathan P., 4743
Chang, Kai-Wei, 1117, 1167, 1198, 3407, 4904, 5276
Chang, Serina, 5746

Chang, Shih-Fu, 313
Chang, Shiyu, 3566, 4094, 6456
Chang, Ting-Yun, 6064
Chang, Xiaojun, 1461
Changpinyo, Soravit, 1468
Chao, Wenhan, 3573
Charoenphakdee, Nontawat, 3993
Chattopadhyay, Prithvijit, 1449
Chaudhary, Aditi, 5164
Chaudhary, Vishrav, 6098
Chauhan, Dushyant Singh, 5647
Chaves Lima, Lucas, 4685
Chawla, Nitesh, 302
Che, Wanxiang, 133, 1586, 2078, 5721, 5883
Chechik, Gal, 1330, 3560
Chen, Bei, 3515, 5316
Chen, Boli, 466
Chen, Changyou, 3946, 5226
Chen, Danqi, 2851
Chen, Derek, 4463
Chen, Fuxiang, 2603
Chen, Haiqing, 4558
Chen, Hong-You, 3579
Chen, Hongshen, 1793
Chen, Huajie, 6362
Chen, Huajun, 4217
Chen, Jamin, 2829
CHEN, Jiajun, 931, 4197
Chen, Jiale, 963
Chen, Junfan, 326
Chen, Kehai, 1361
Chen, Lei, 6280
Chen, Long, 5144
Chen, Luyao, 1962
Chen, Ming, 3548
Chen, Mingda, 421, 649
Chen, Mingyang, 4217
Chen, Muhao, 1198, 5276
Chen, Nancy, 6165
Chen, Patrick H., 5257
Chen, Peng-Jen, 6098
Chen, Qiang, 4217
Chen, Qibin, 1803
Chen, Qin, 5033
Chen, Shizhe, 5133
Chen, Shuang, 3022
Chen, Tao, 1962
Chen, Ting, 5033
Chen, Tong, 2672
Chen, Weizhu, 638
Chen, Xinchi, 5415
Chen, Xinyue, 2829
Chen, Xiuying, 1931, 3741, 4959
Chen, Yanda, 4765
Chen, Yang, 421
Chen, Yi-Pei, 1507
Chen, Yubo, 738, 3830
Chen, Yufeng, 1051, 2652, 5569
Chen, Yun-Nung, 1061, 1855, 3370, 6064
Chen, Zhigang, 271
Chen, Zhipeng, 5883
Cheng, Gong, 5866
Cheng, Renhong, 4601, 5559
Cheng, Xiang, 2642
Cheng, Xueqi, 5766
Cheng, Yong, 3132
Cheng, Yu, 3304, 4323
Cheung, Alvin, 5426
Cheung, Jackie Chi Kit, 3163, 3382, 6019
Cheung, William K., 4047
Chhaya, Niyati, 154, 1642
Chinnappa, Dhivya, 663
Chintagunta, Bharath, 4537
Chiu, Elena Suet-Ying, 6155
Cho, Jaemin, 3121
Cho, Kyunghyun, 2402, 3342, 3700, 4385
Cho, Sungjun, 4991
Choi, Jinsoo, 2012
Choi, Seungtaek, 5890
Choi, Yejin, 1494, 2391, 3752, 4463, 5043
Choo, Jaegul, 2603
Choubey, Prafulla Kumar, 6343
Chouliaras, Georgios Christos, 6419
Christodoulopoulos, Christos, 2944
Christopoulou, Fenia, 4925
Chu, Chenhui, 1410
Chu, Zewei, 421, 649
Chua, Tat Seng, 942
Chua, Tat-Seng, 2723
Chua, Tat-seng, 261
Cifor, Marika, 2328
Cinarel, Ceyda, 3458
Clark, Christopher, 4069
Clark, Elizabeth, 5043
Clark, Peter, 2814, 4496, 5941, 6076
Clark, Stephen, 2284
Clarke, Christopher, 1311
Clarke, Nancy, 4526
Clavel, Chloé, 5539
Cognetta, Marco, 6425
Cohan, Arman, 3615, 3693
Cohen-Karlik, Edo, 5625

Cohen, Daniel, 3497
Cohen, Shay B., 1071, 2057, 4104
Cohen, William, 2380, 2567
Cohn, Trevor, 1488, 1729
Collins, Michael, 2131
Colombo, Pierre, 5539
Conia, Simone, 627
Copestake, Ann, 974
Correia, Gonçalo M., 2174
Côté, Marc-Alexandre, 2796
Cotterell, Ryan, 974, 2900, 5267, 5276, 5734
Cottrell, Garrison, 5988
Crabbé, Benoit, 1150
Cretu, Ana-Maria, 4303
Croce, Danilo, 4037
Croft, W. Bruce, 3497
Crook, Paul, 1951
Cui, Baiyun, 3548
Cui, Leyang, 4115
Cui, Yiming, 1586, 5883
Czapla, Piotr, 5702
Czarnowska, Paula, 974

Da San Martino, Giovanni, 5636
Dabre, Raj, 1410
Dadian, Stephane, 1255
Dagan, Ido, 3110
Dai, Hongliang, 6210
Dai, Wei, 6362
DAI, XIN-YU, 931, 4197
Dai, Xinyu, 5866
Dai, Zehui, 6362
Dai, Zeyu, 2976
d'Alché-Buc, Florence, 5539
Dalvi, Bhavana, 3693, 4496, 6076
Danescu-Niculescu-Mizil, Cristian, 4743
Dankers, Verna, 2218
Dankin, Lena, 5591
Das, Abhishek, 1449
Dasigi, Pradeep, 5925
Daumé III, Hal, 684, 4831
Dauphin, Yann, 5696
Davidson, Sam, 1513
Davis, Larry, 1481
Davison, Joe, 1173
Daza, Angel, 603
De Andrade, Anderson, 6293
de Lhoneux, Miryam, 2755
de Marneffe, Marie-Catherine, 6086
de Melo, Gerard, 6306
Dehghani, Morteza, 5753
Del Tredici, Marco, 4707
Delong, Natalie, 5753
Demberg, Vera, 5790
deng, xiang, 410
Deng, Yuntian, 1210
Derby, Steven, 5853
Deruyttere, Thierry, 2088
Desai, Shrey, 4718
Deutsch, Daniel, 3720
Devereux, Barry, 5853
Dhamecha, Tejas, 6071
Dhingra, Bhuwan, 2567
Di Fabio, Andrea, 627
Diab, Mona, 1285, 1442, 3672, 4526
Diesner, Jana, 1475, 2141
Dinan, Emily, 673, 4537
Ding, Jiwei, 2614
Ding, Ming, 1803
Ding, Ning, 347
Ding, Xiao, 2682, 4894
Ding, Xiaoan, 507
Ding, Yadong, 6362
Dinu, Liviu P., 2317
Divakaran, Ajay, 4622, 5860
Dixit, Shreya, 1962
Do, Quynh, 1455
Dodge, Jesse, 1179, 2185
Dognin, Charles, 4058
Dolan, Bill, 1814, 3304
Dong, Bin, 6232
Dong, Jin, 4506
Dong, Li, 4143
Dong, Tiansi, 4969
Dong, Wei, 5226
Dong, Wen, 3946
Dong, Xin, 6306
Dong, Yue, 6019
Dong, Zhen, 5405
Dou, Zi-Yi, 1192, 1417
Dredze, Mark, 833
Drozdov, Andrew, 1507
Du, Chunning, 456, 5489
Du, Donghong, 6210
Du, Jiachen, 5614
Du, Jinhua, 389
Du, Li, 2682
Du, Nan, 4979
Du, Wanyu, 6012
Du, Yupei, 6133
Duan, Junwen, 4894
Duan, Nan, 1618, 2442, 2485, 6059
Duan, Xiangyu, 3044

Dubey, Amit, 4516
Dubnov, Shlomo, 5216
Duckworth, Daniel, 4516
Dufter, Philipp, 1185
Duh, Kevin, 1382, 3786
Durmus, Esin, 5668
Durrett, Greg, 759, 2712, 3292
Dvijotham, Krishnamurthy, 4083
Dyer, Chris, 4083, 5846

Edunov, Sergey, 5360
Egawa, Ryo, 6274
Eger, Steffen, 563
Eisenschlos, Julian, 5702
Eisenstein, Jacob, 4238
Eisner, Jason, 2744, 6438
Ekbal, Asif, 5647
Elazar, Yanai, 6330
Elgohary, Ahmed, 5918
Elliott, Desmond, 6330
Elsahar, Hady, 2163
Emami, Ali, 3382
Er, Heyang, 1962, 5338
Erk, Katrin, 2712
Eshghi, Arash, 1741
Eskenazi, Maxine, 1752
Espinosa, Kurt Junshean, 3679
Essid, Slim, 5539
Ethayarajh, Kawin, 55, 3503

Fabbri, Alexander, 1962
Fabre, Murielle, 1150
Fan, Angela, 673, 4186
Fan, Bin, 2068
Fan, Chuang, 5614
Fan, Lisa, 6343
Fan, Lu, 4799
Fan, Miao, 1675
Fancellu, Federico, 2769
Fang, Le, 3946
Fang, Lei, 3515, 5316
Fano, Elena, 2755
Fasching, Joshua, 6444
Feldman, Joshua, 1173
Feng, Chao, 1675
Feng, Chong, 5206
Feng, Ling, 1718
Feng, Shi, 2153, 5066
Feng, Steven Y., 2701
Feng, Weijiang, 6155
Feng, Xiaocheng, 3022, 3143
Feng, Xinwei, 5896
Feng, Yang, 1527
Feng, Yansong, 240, 1291, 2632, 3851, 6155
Fergus, Rob, 2402
Fernández, Raquel, 2865, 4707
Ferreira Pinto Junior, Renato, 4654
Ferreira, William, 6350
Firat, Orhan, 1538, 1565
Fisch, Adam, 5714
Fischer, Frank, 6238
Florian, Radu, 399
Fodor, Brigi, 4526
Forbes, Maxwell, 708
Fraiberger, Samuel, 2338
Frank, Anette, 603, 2541
Freitas, André, 4313
Frermann, Lea, 2057
Friedman, Roni, 5625
Fu, Cong, 2672
Fu, Guohong, 997
Fu, Jie, 2796
Fu, Jinlan, 1040
Fu, Xuandi, 4799
Fu, Zhenxin, 4937
Fujita, Atsushi, 1410
Fujita, Katsuhide, 6274
Fung, Pascale, 121, 1297, 3065, 3541

Gai, Kun, 4959
Gallé, Matthias, 1375, 2163
Galley, Michel, 1814
Gallinari, patrick, 696
Galstyan, Aram, 4026
Gamon, Michael, 5002
Gan, Wee Chung, 5297
Gan, Zhe, 2141, 2510, 3304, 4323
Gangal, Varun, 1696
Ganguly, Niloy, 6092
Gao, Fei, 4375
Gao, Jianfeng, 1475, 1494, 1814, 2141, 2510, 3946
Gao, Jun, 1898
Gao, Mingfei, 1481
Gao, Shen, 3741
GAO, Sheng, 496
Gao, Tianyu, 6250
Gao, Wei, 198
Gao, Xiang, 1814
Gao, Yang, 563, 3076, 3110
Gao, Yifan, 3216
Gao, Yingbo, 1324
Gao, Yupeng, 6456
Gao, Zhe, 6187
Garbacea, Cristina, 3968

Garcia, Alexandre, 5539
Gardent, Claire, 3086, 3183, 4186
Gardner, Matt, 2153, 3659, 5307, 5925, 5941
Garg, Sahil, 4026
Garg, Sarthak, 4453
garg, shweta, 5911
Gaspers, Judith, 1455
Gauthier, Jon, 529
Ge, Suyu, 4884, 6389
Ge, yubing, 845
Geiger, Atticus, 4485
Gelbukh, Alexander, 154
Geng, Ruiying, 3904
Geng, Xiubo, 2442
Gera, Ariel, 5591
Geva, Mor, 1161
Ghazvininejad, Marjan, 6112
Ghosal, Deepanway, 154
Ghosh, Sabyasachi, 4260
Gildea, Daniel, 208
Gilroy, Sorcha, 2769
Gimpel, Kevin, 421, 507, 649
Glass, James, 4442
Glavaš, Goran, 2206, 4407
Gligorić, Kristina, 4831
Göksel, Kerem, 4129
Goldberg, Yoav, 31, 1161, 4175
Goldberger, Jacob, 1330, 3560
Gómez-Rodríguez, Carlos, 1500
Gonen, Hila, 4175, 5267
Gong, Heng, 3143
Gong, Hongyu, 518
Gong, Ming, 2485
Gong, Yeyun, 6059
Gontrum, Johannes, 2755
Gonzalez, Ana Valeria, 1229
Gonzalez, Elaine, 5753
Gooding, Sian, 4853
Goodrich, Ben, 4516
Gordon, Jonathan, 6052
Goto, Jun, 6317
Gowal, Sven, 4083
Goyal, Pawan, 6092
Goyal, Rasna, 4260
Gravano, Luis, 4611
Grave, Edouard, 974
Grenander, Matt, 6019
Gretz, Shai, 5625
Grissom II, Alvin, 6355
Grosu, Ioana, 2877
Grujicic, Dusan, 2088
Gu, Jia-Chen, 1845
Gu, Mao, 2109
Gu, Shuhao, 1527
Gu, Xiaotao, 271
Gu, Yuxian, 3376
Guan, Weili, 3821
Guan, Wenyv, 2370
Gugger, Sylvain, 5702
Gui, Lin, 3478, 5614
Gui, Min, 1222
Gui, Tao, 1040
Guo, Daya, 2442
Guo, Honglei, 5559
Guo, Hongyu, 326
Guo, Jiafeng, 5766
Guo, Jiang, 5714, 5721
Guo, Xiaoxiao, 6456
Gupta, Aditya, 759
Gupta, Arshit, 1285
Gupta, Manish, 1642
Gupta, Sonal, 1520
Gupta, Swapnil, 378
Gupta, Vivek, 3924
Gurevych, Iryna, 1093, 1607, 3110, 3183, 3982
Gururangan, Suchin, 2185
Guy, Ido, 3873
Guzmán, Francisco, 6098

Ha, Jung-Woo, 2603
Hadiwinoto, Christian, 5297
Hajishirzi, Hannaneh, 2851, 3121, 5784
Hale, John, 5846
Hale, Scott, 66
Hall Maudslay, Rowan, 5267
Hall, Keith, 5846
Hämäläinen, Mika, 5999
Hamilton, William L., 4506
Hammerla, Nils, 77
Han, Hojae, 2840, 5890
Han, Jiawei, 445, 5154
Han, Jizhong, 111
Han, Rujun, 434
Han, Simeng, 1007
Han, Wenjuan, 1423, 5728
Han, Xianpei, 292, 6232
Han, Xiaochuang, 4238
Han, Xu, 219, 3376, 5777, 6250
Han, Xudong, 1488
Han, Yo-Sub, 6425
Handler, Abram, 5969, 6355
Hansen, Casper, 4685
Hansen, Christian, 4685

Hao, Jie, 887, 1336
Hao, Yaru, 4143
Harding Graesser, Laura, 3700
Hartmann, Mareike, 3336
Hasan, Md Kamrul, 2046
Hashimoto, Chikara, 2988
Hashimoto, Tatsunori, 4227
Hauptmann, Alexander, 1461
Hay, Jamaal, 3914
Hayashi, Katsuhiko, 2422
He, Ben, 292
He, Bolei, 496
He, Di, 963, 4375, 5708
He, Guoxiu, 6187
He, Junxian, 3603
He, Luheng, 5947
He, Qing, 3033
He, Ruifang, 1920
He, Shexia, 5350
He, Shilin, 953
He, Shizhu, 367, 2431
He, Xiahui, 5033
He, Xiaohao, 1718
He, Yulan, 177, 282, 3478
He, Zhongjun, 856
Heafield, Kenneth, 3626
Hedderich, Michael A., 3554
Henderson, James, 4419
Herschcovich, Daniel, 1229
Herzig, Jonathan, 3810
Hessel, Jack, 2034
Hestness, Joel, 4293
Hewitt, John, 2733
Hidey, Christopher, 2933
Hill, Felix, 5838
Hill, Parker, 1311
Hirao, Tsutomu, 5797, 6408
Hiraoka, Takuya, 6225
Hirst, Graeme, 4654
Hockenmaier, Julia, 5112
Hoey, Jesse, 2701
Hofmann, Thomas, 3400
Hogan, Aidan, 718
Holtzman, Ari, 3752, 5043
Holvenstot, Chad, 4731
Hong, Yi-Te, 3579
Hong, Yu, 2251
Hoque, Mohammed (Ehsan), 2046
Horwood, Graham, 4731
Hospedales, Timothy, 5185
Hou, Lei, 2723, 3376, 4969
Houben, Geert-Jan, 2349
Hovy, Eduard, 1696, 3324, 4282, 5809
Howard, Jeremy, 5702
Hsieh, Cho-Jui, 5257
Hsu, Daniel, 4611
Hsu, Tsung-Yuan, 5933
Htut, Phu Mon, 2877
Hu, Changjian, 4547
Hu, Chi, 3585
Hu, Chuan, 5175
Hu, Guoping, 271, 1586, 5883
Hu, Junjie, 1388, 1417, 1475
Hu, Linmei, 3821
Hu, Mengting, 4601, 5559
Hu, Minghao, 1596
Hu, Songlin, 111
Hu, Wei, 2614
Hu, Wenpeng, 1909, 1931
Hu, Zhe, 3280
Hu, Zikun, 261
Hua, Xinyu, 591
Huai, Jinpeng, 3851
Huang, Binxuan, 4732, 5469
Huang, Fei, 4271
Huang, Guoping, 5689
Huang, Haoyang, 2485
Huang, Heyan, 3076
Huang, Jianqiang, 4874
Huang, Kuan-Hao, 5276
Huang, Liang, 1349, 1395
Huang, Lianzhe, 3444
Huang, Lifu, 2391
Huang, Luyang, 3280
Huang, Luyao, 3509
Huang, Minlie, 100, 3257, 4271
Huang, Po-Sen, 4083
Huang, Po-Yao, 1461
Huang, Qiuyuan, 1475, 2141
Huang, Rui, 1028
Huang, Ruihong, 2976, 6343
Huang, Shujian, 931, 4197
Huang, Siyu, 2001
Huang, Xiaolei, 6395
Huang, Xin, 466
Huang, Xinjing, 4558
Huang, Xuancheng, 6122
Huang, Xuanjing, 1040, 2692, 3509
Huang, Xuedong, 1261
Huang, Yongfeng, 4874, 4884, 6389
Huang, Yuyun, 389
Huang, Zhen, 1596

Huang, Zixian, 5866
Huber, Patrick, 2306
Humeau, Samuel, 673, 4537
Huo, Zhouyuan, 3132
Hurtado Bodell, Miriam, 6323
Hussain, Shehzeen, 5216
Hutchinson, Ben, 5740
Hwang, Alyssa, 2933
Hwang, Emily, 4831
Hwang, Seung-won, 2603, 2840, 5890
Hwu, Wen-Mei, 518

Iida, Ryu, 5816
Inui, Kentaro, 579, 1236, 3665
Ive, Julia, 3227
Iwakura, Tomoya, 6244
Iyer, Srinivasan, 5426, 5436
Iyyer, Mohit, 1507, 6355

J Kurisinkel, Litton, 6165
Jaakkola, Tommi, 4094
Jacovi, Michal, 5591, 5625
Jain, Alankar, 1083
Jain, Nitisha, 5771
Jain, Saachi, 673
Jana, Abhik, 6092
Jäschke, Robert, 6238
Jauhar, Sujay Kumar, 5002
Jeretic, Paloma, 2877
Jhamtani, Harsh, 6025
Ji, Feng, 4558
Ji, Heng, 261, 313, 6197
Ji, Houye, 4821
Ji, Yangfeng, 6012
Jia, Robin, 4129
Jia, Ruoyu, 4003
Jian, Ping, 3904
Jiang, Daxin, 2442, 2485
Jiang, Feng, 3022
Jiang, Jyun-Yu, 4904
Jiang, Meng, 302
Jiang, Ming, 1475, 2141
Jiang, Nanjiang, 6086
Jiang, Qingnan, 6280
Jiang, Shanshan, 6232
Jiang, Tianwen, 302
Jiang, Wenbin, 5896
Jiang, Xin, 3173
Jiang, Yichen, 4474
Jiang, Yong, 1423, 5728
Jiang, Youxuan, 1962
Jiang, Yufan, 3585
Jiang, Zhuoren, 6187
Jie, Zhanming, 3862
Jin, Beihong, 5090
Jin, Di, 3097
Jin, Hailong, 4969
Jin, Haolin, 4644
Jin, Hongxia, 1279
Jin, Qiao, 2567
Jin, Qin, 5133
Jin, Tao, 2001
Jin, Weike, 2109
Jin, Woojeong, 2672
Jin, Xiaolong, 5766
Jin, Yiping, 3993
Jin, Zhijing, 3097
Jing, Liping, 466
Jing, Yimin, 2452
Jo, Eunju, 2273
Jo, Hwiyeol, 3458
Johnson, Melvin, 3632
Joshi, Mandar, 5803
Joshi, Sachindra, 1980
Joshi, Vidur, 43
Joty, Shafiq, 1007, 2262, 2964, 6432
Jung, Taehee, 3324
Jurafsky, Dan, 4622
Jurgens, David, 1664
Jwalapuram, Prathyusha, 2964

K Sarma, Prathusha, 5549
Kadowaki, Kazuma, 5816
Kadras, Marcin, 5702
Kaeser-Chen, Christine, 708
Kamigaito, Hidetaka, 5797
Kandpal, Nikhil, 2153
Kang, Dongyeop, 1696, 1951, 3324, 5809
Kang, Gi-Cheon, 2024
Kang, Yangyang, 6187
Kann, Katharina, 3342
Kano, Ryuji, 3195
Kantor, Yoav, 5591
Karamanolakis, Giannis, 4611
Karamcheti, Siddharth, 673, 2402
Karttunen, Lauri, 4485
Kato, Tsuneaki, 3521
Kawahara, Daisuke, 2370
Ke, Pei, 4271
Keldibek, Amina, 6419
Keller, Frank, 1707
Kementchedjhieva, Yova, 1105, 3336, 6330
Kenkre, Sreyash, 378
Kennedy, Brendan, 5753

Keskar, Nitish Shirish, 540
Keung, Phillip, 1355
Khadanga, Swaraj, 6432
Khader, Hadi, 2119
Khadivi, Shahram, 866
Khalil, Talaat, 6419
Khapra, Mitesh M., 3314
Khashabi, Daniel, 3363
Khayrallah, Huda, 1382
Khot, Tushar, 2814
Kiela, Douwe, 673, 2402, 3700, 4385
Kiełczewski, Kornel, 6419
Kihara, Daisuke, 5953
Kim, Dong-Jin, 2012
Kim, Gyuwan, 5022
Kim, Kyu-Young, 4516
Kim, Sunghun, 2603
Kim, Taeuk, 3528
Kim, Yunsu, 866
King, Daniel, 3693
King, Irwin, 3216
Kiyono, Shun, 1236
Klakow, Dietrich, 579, 3554, 3762
Klein, Dan, 2195
Kloetzer, Julien, 5816
Knowles, Rebecca, 1382
Ko, Sang-Ki, 6425
Kobayashi, Naoki, 5797
Kochmar, Ekaterina, 4853
Kocijan, Vid, 4303
Koehn, Philipp, 1342, 1382, 6098, 6438
Kohli, Pushmeet, 4083
Kondratyuk, Dan, 2779
Korhonen, Anna, 1273, 2206, 2900, 4407
Kotlerman, Lili, 5591
Kotnis, Bhushan, 1
Koushanfar, Farinaz, 5216
Kovaleva, Olga, 4365
Koychev, Ivan, 2099
Kozareva, Zornitsa, 3894, 5012
Krahmer, Emiel, 552
Krause, Sebastian, 5054
Krishnamoorthi, Karthik, 4516, 5012
Krishnan, Radhika, 1642
Krone, Jason, 4526
Kross, Sean, 4831
Kruengkrai, Canasai, 6311
Kruk, Julia, 4622
Kryscinski, Wojciech, 540
Kuchhal, Dhruv, 4831
Kudugunta, Sneha, 1565
Kulmizev, Artur, 2755, 5838
Kumar, Harshit, 1980
Kumar, Sachin, 4153
Kumar, Sawan, 5911
Kummerfeld, Jonathan K., 1311
Kuncoro, Adhiguna, 5846
Kunkel, Rose, 6176
Kurohashi, Sadao, 5758
Kweon, In So, 2012

Labeau, Matthieu, 4104
Labutov, Igor, 1129, 4164
Ladhak, Faisal, 5668
Lahav, Dan, 5625
Lai, Chih-Te, 3579
Lai, Jianhuang, 4207, 4375
Lai, K. Robert, 3432
Lai, Kwunping, 250
Lai, Tuan, 5953
Lai, Vivian, 486
Lalor, John P., 4249
Lalwani, Garima, 1285
Lam, Albert Y.S., 4799
Lam, Wai, 250, 3799
Lample, Guillaume, 6098
Lamprier, Sylvain, 3246
Lan, Man, 6133
Lange, Lukas, 3554
Lapata, Mirella, 1018, 1707, 2057, 2769, 3730, 3774
Larson, Stefan, 1311
Lasecki, Walter, 1962
Laurenzano, Michael A., 1311
Lavee, Gal, 3873
Lavee, Tamar, 5591
Lawrence, Carolin, 1
Le Bras, Ronan, 2391, 4463
Le, Matthew, 5872
Le, Ran, 1909
Leach, Kevin, 1311
Lee, Andrew, 1311
Lee, Dongjun, 6045
Lee, Gang, 4979
Lee, Hung-Yi, 1061, 5960
Lee, Hung-yi, 1855, 5933
Lee, Jason, 4385
Lee, Jongyeong, 3993
Lee, Juho, 2840
Lee, Kenton, 5947
Lee, Kyong-Ho, 2273
Lee, Kyungjae, 2840
Lee, Lillian, 2034

Lee, Moontae, 4991
Lee, Sang-goo, 3528
Lee, Stefan, 5860
Lee, Sungjin, 1741, 1814
Lee, Taesung, 728
Lei, Deren, 6038
Lei, Wenqiang, 942
Lemon, Oliver, 1741
Leng, Jia, 3478
Lent, Heather, 1229
Lertvittayakumjorn, Piyawat, 5195
Leung, Ho-fung, 357
Levy, Omer, 5803, 6112
Levy, Roger, 529, 2888
Lewis, Martha, 2218
Lewis, Mike, 1520
Lewis, Patrick, 2463
Li, Aaron W., 2701
Li, Binhua, 3904
Li, Bohan, 3603
Li, Chaozhuo, 6286
Li, Chen, 1028
Li, Chengjiang, 2723
Li, Chenliang, 2521
Li, Chunyuan, 1494, 3946
Li, Dianqi, 3304
Li, Dongsheng, 1596
Li, Fei, 6155
Li, Guanlin, 5689
Li, Hao, 5478
Li, Jia, 1291
Li, Jiacheng, 188
Li, Jing, 3153, 4579, 4633
Li, Jingjing, 3216
Li, Juanzi, 2723, 3376, 4969, 5777
Li, Junhui, 1898, 5459
Li, Junjie, 3000
Li, Juntao, 1931, 4959
Li, Junyi Jessy, 4718
Li, Lei, 3388
Li, Miao, 5090
Li, Mingming, 111
Li, Ningyun, 1718
Li, Pan, 3237
Li, Peng, 2474, 5777, 6250
Li, Pengfei, 229
Li, Piji, 250, 3741, 4937
Li, Ping, 1675, 5236
Li, Qi, 229
Li, Qian, 1762
Li, Qimai, 4799
Li, Qiuchi, 4568
Li, Quanzhi, 2328
Li, Ruiping, 2642
Li, Shoushan, 5499, 5581
Li, Shuangyin, 2623
Li, Shunyao, 3388
Li, Shuyang, 5976
Li, Siqing, 4914
Li, Siyao, 6038
Li, Sujian, 2370, 3444, 4003, 5307
Li, Suyi, 1962
Li, Tao, 3924
Li, Tianrui, 6286
Li, Xian, 4282
Li, Xiang Lisa, 2744
Li, Xiao, 5866
Li, Xiaoli, 4821
Li, Xiaolin, 5144
Li, Xin, 3632, 4590, 6210
Li, Xiujun, 1494
Li, Yangming, 133, 2078
Li, Yingjie, 6299
Li, Yingming, 2001, 3548
Li, Yingya, 4664
Li, Yongbin, 3904
Li, Yuanpeng, 4293
Li, Zheng, 4590
Li, Zhenghua, 5382
Li, Zhifei, 1762
Li, Zhongyang, 2682, 4894
Li, Zhoujun, 3515, 3573
li, zhoujun, 1886, 5077
Li, Zhuohan, 5708
Li, Zifan, 1962
Li, Zihan, 1962
Li, Ziran, 347
Li, Zuchao, 5350
Liang, Di, 2692
Liang, Percy, 2733, 4129, 4227
Liang, Yaobo, 2485
Liang, Yingyu, 5549
Liang, Yun, 5518
Liang, Yunlong, 5569
Liang, Zhicheng, 4864
Liao, Jianxin, 456, 5489
Liao, Kuo, 4894
Liao, Ming, 4579
Liao, Xiangwen, 5896
Lien, Yen-Chieh, 3497
Lignos, Constantine, 3497
Lim, Jaeseo, 2024

Lin, Bill Yuchen, 2829
Lin, Chin-Yew, 3022
Lin, Fen, 219
LIN, Hongfei, 6402
Lin, Hongyu, 6232
Lin, Jimmy, 1204, 3490, 4431, 5370, 5823
Lin, Jun, 198
Lin, Junyang, 1803, 5508
Lin, Kevin, 5941
Lin, Leyu, 219
Lin, Sheng, 271
Lin, Shou-De, 3579
Lin, Xi Victoria, 1962, 5338
Lin, Xiang, 1007
Lin, Xiao, 4622
Lin, Xiaola, 6059
Lin, Xiaowen, 2590
Lin, Xinzhu, 5033
Lin, Yankai, 2474
Lin, Ying, 6197
Lin, Zehao, 4558
Lin, Zhaojiang, 121, 3541
Lin, Zheng, 2412
Lin, Zhouhan, 2796
Lin, Zi, 5708
Lin, Zibo, 347
Lin, Zizheng, 4675
Ling, Jeffrey, 2131
Ling, Zhen-Hua, 1845
Linmei, Hu, 4821
Linzen, Tal, 5831
Lioma, Christina, 4685
Lipton, Zachary C., 21, 1083
Liu, Bing, 1304, 6286
Liu, Cao, 2431
Liu, Chen, 6293
Liu, Chi-Liang, 5933
Liu, Everest, 4831
Liu, Fei, 563
Liu, Han, 4799
Liu, Haokun, 2877
Liu, Haoyan, 3515, 5316
Liu, Hongzhi, 2662, 5405
Liu, Huan, 2360
Liu, Jiacheng, 5112, 5154
Liu, Jian, 738
Liu, Jingjing, 2510, 4323
Liu, Jingming, 4003
Liu, Kang, 367, 738, 2431, 3830, 5896
Liu, Lemao, 5689
Liu, Linlin, 1007
Liu, Linqing, 1204, 5370
Liu, Liyuan, 3841, 5154
Liu, Ming, 5489
Liu, Mingtong, 2652
Liu, Nelson F., 5925
Liu, Qian, 3515, 5316
Liu, Qianying, 2370
Liu, Quan, 1845
Liu, Rui, 2412
Liu, Shengping, 367, 3830
Liu, Shujie, 770, 1834
Liu, Tie-Yan, 963, 4207, 4375, 5708
Liu, Ting, 133, 302, 1586, 2078, 2682, 3143, 4894, 5721, 5883
Liu, Xiao, 240
Liu, Xiaodong, 2510
Liu, Xiaojiang, 1824, 1866, 1898
Liu, Xiaopeng, 5206
Liu, Xiaozhong, 198, 2328, 5581, 6187
Liu, Xikai, 6402
Liu, Xudong, 5679
Liu, Yang, 791, 845, 3730, 6122
Liu, Yijia, 133, 5721
Liu, Yijin, 1051
Liu, Yingchi, 2328
Liu, Yingru, 780
Liu, Yinhan, 5360, 6112
Liu, Yinyin, 4755
Liu, Yuanxin, 2412
Liu, Yuchen, 3350
Liu, Zhengping, 2567
liu, zhibin, 1782
Liu, Zhiyuan, 219, 261, 347, 2474, 3376, 5777, 6250
Liu, Zihan, 1297, 3541
Liza, Farhana Ferdousi, 66
Lo, Kyle, 3615
Lofi, Christoph, 2349
Logan, Robert, 43
Long, Guodong, 2442
Lopez, Adam, 1105, 2769
Lorré, Jean-Pierre, 2296
LOU, Jian-Guang, 3515, 5316
Lou, Jian-Guang, 5405
Louis, Annie, 6019
Low, Daniel M., 5838
Lowell, David, 21
Lu, ChangTien, 5002
Lu, Chi-Jen, 3579
Lu, Chujie, 5144
Lu, Di, 313

Lu, Lihao, 5154
Lu, Wei, 3862, 4431, 5327, 5478, 6216
Lu, Xinghua, 2567
Lu, Yaojie, 845, 6232
lu, yichao, 1355
Lu, Zhichu, 1129
Luan, Huanbo, 791, 6122
Luan, Yi, 5784
Lubin, Jonah, 4622
Lukasiewicz, Thomas, 4303
Luo, Fuli, 3388
luo, jiebo, 845
Luo, Ling, 3033
Luo, Weihua, 3044
Luo, Yuan, 1267
Luo, Zhunchen, 5206
Lv, Shangwen, 111
Lv, Tengchao, 476
Lv, Xin, 3376
Lyu, Chunchuan, 1071, 5415
Lyu, Michael, 953
Lyu, Michael R., 3216
Lyu, Yajuan, 5896

Ma, Bing, 456
Ma, Dehong, 3444
Ma, Mingbo, 1349, 1395
Ma, Shuai, 770, 3851
Ma, Tengfei, 6071
Ma, Wentao, 5883
Ma, Xiaofei, 5878
Ma, Xiaojuan, 3957, 6376
Ma, Xuezhe, 1388, 4282
Ma, Xutai, 3786
Ma, Ye, 4843
Ma, Yukun, 5257
Mabona, Amandla, 2284
Madasu, Avinash, 5658
Madhyastha, Pranava, 3227
Madotto, Andrea, 121, 1297, 3065
Magnusson, Måns, 6323
Mahendran, Anish, 1311
Maheswaran, Arjun, 3269
Majumder, Bodhisattwa Prasad, 5976, 5988
Majumder, Navonil, 154
Makino, Kiminobu, 6317
Makino, Takuya, 6244
Malakasiotis, Prodromos, 6005
Malmi, Eric, 5054
Mandt, Stephan, 3400
Mandyam, Karishma, 1520
Manning, Christopher D., 2590
Mao, Hong-Ren, 5960
Mao, Huanru Henry, 5988
Mao, Kezhi, 229
Mao, Mingzhi, 2623
Mao, Ruibin, 5614
Mao, Wenji, 4787
Mao, Yongyi, 326, 5679
Mao, Yuning, 445
Marasovic, Ana, 5925
Marcheggiani, Diego, 4707
mars, Jason, 1311
Martelli, Federico, 3534
Martins, André F. T., 2174
Maru, Marco, 3534
Maszkiewicz, Lisa, 4831
Matthews, Nicholas, 3097
Mausam, Mausam, 6262
May, Jonathan, 313, 6052, 6395
Mayhew, Stephen, 6256
Mazurek, Michelle, 4831
McAuley, Julian, 188, 1876, 5976, 5988
McCallum, Andrew, 1507
McCann, Bryan, 540
McCarthy, Arya D., 2241
McGillivray, Barbara, 66
McKeown, Kathy, 2933, 4765, 5746
Mehr, Leo, 2590
Mehri, Shikib, 1752
Mehta, Kartik, 5911
Mehta, Maitrey, 3924
Mehta, Pratik, 3497
Mehta, Sanket Vaibhav, 6025
Mei, Qiaozhu, 3968
Meinhardt, Eric, 6176
Melamud, Oren, 3884
Meng, Dan, 2412
Meng, Fandong, 1051, 1527, 5569
Meng, Tao, 1117
Meng, Xiangbin, 5766
Mensah, Samuel, 5679
Mentch, Lucas, 3324
Merlo, Paola, 4419
Merullo, Jack, 6355
Mesbah, Sepideh, 2349
Mesquita, Filipe, 749
Meyer, Christian M., 563, 3110
Miao, Chunyan, 165
Mihalcea, Rada, 1255
Mihaylov, Todor, 2541
Miller, Alexander, 2463
Miller, Paul, 5853

Miller, Scott, 3497
Mimno, David, 2034, 4991
Min, Bonan, 1249, 6444
Min, Qingkai, 3652
Min, Sewon, 2851
Mirinjian, Arineh, 5753
Mirkin, Shachar, 5591
Mirylenka, Daniil, 5054
Mirza, Paramita, 749, 5771
Misawa, Shotaro, 3195
Mishra, Abhijit, 6144
Mishra, Ajay, 1442
Mita, Masato, 1236
Mitchell, Margaret, 5740
Mitchell, Tom, 1129, 4164
Mittal, Arpit, 2944, 3591
Miura, Yasuhide, 3195
Miwa, Makoto, 3679, 4925
Miyazaki, Taro, 6317
Mizumoto, Tomoya, 1236
Moens, Marie-Francine, 2088
Mohananey, Anhad, 2877
Mohankumar, Akash Kumar, 3314
Mohiuddin, Tasnim, 2262
Mohtarami, Mitra, 4442
Molino, Piero, 1304
Moniz, Joel Ruben Antony, 6268
Moon, Han Cheol, 2262
Moosavi, Nafise Sadat, 1607
Morales, Miraida, 4831
Morency, Louis-Philippe, 2046, 4344
Moreno, Francesc, 3451
Morio, Gaku, 6274
Mosallanezhad, Ahmadreza, 2360
Moshtaghi, Masud, 823
Mostafazadeh Davani, Aida, 5753
Motlicek, Petr, 5994
Mou, Lili, 3573
Mouallem, Marwa, 2119
Mueller, Aaron, 2241, 5831
Mueller, Jonas, 3097
Mueller, Thomas, 5902
Mukherjee, Animesh, 6336
Mukherjee, Subhabrata, 6413
Mullenbach, James, 6052
Murahari, Vishvak, 1449
Murawaki, Yugo, 5758
Muresan, Smaranda, 2933
Murugan, Srikala, 663
Muthuswamy, Karthik, 3153

Nagaev, Aleksander, 3936
Nagata, Masaaki, 5797, 6408
Nagata, Ryo, 3206
Nakamura, Kengo, 5797
Nakov, Preslav, 2099, 2964, 4442, 5636
Nallapati, Ramesh, 5878
Nallasamy, Udhyakumar, 4453
Naslidnyk, Masha, 3591
Nastase, Vivi, 1255
Natarajan, Premkumar, 3407
Navigli, Roberto, 88, 627, 3534
Nazir, Ambreen, 4644
Neekhara, Paarth, 5216
Neelakantan, Arvind, 4516
Negri, Matteo, 1368
Nema, Preksha, 3314
Nenkova, Ani, 1216
Neubig, Graham, 984, 1388, 1417, 3603, 4282, 5164
Neumann, Mark, 43
Ney, Hermann, 866, 1324
Ng, Hwee Tou, 5297
Ng, Patrick, 5878
Ng, Wilfred, 5247
Nguyen, Dong, 66
Nguyen, Khanh, 684
Ni, Jian, 399
Ni, Jianmo, 188, 1876, 5976
Nickel, Maximilian, 5872
Nicosia, Massimo, 5902
Niculae, Vlad, 2174
Nie, Liqiang, 3821
Nie, Yining, 2877
Nie, Yixin, 2553
Nie, Zaiqing, 2431
Niepert, Mathias, 1, 6225
Ning, Qiang, 434, 3363, 6203
Ninomiya, Takashi, 6244
Nishino, Masaaki, 6408
Nishino, Toru, 3195
Niu, Guocheng, 496
Niu, Muyao, 3438
Niu, Tong, 1317
Niu, Zheng-Yu, 1782
Nivre, Joakim, 1429, 2755
Norton, Thomas, 4947
Nugmanova, Aigul, 3936

O'Connor, Brendan, 5969, 6355
Oh, Alice, 1941
Oh, Byungkook, 2273
Oh, Jong-Hoon, 5816
Oh, Tae-Hyun, 2012

Ohkuma, Tomoko, 3195
Okamoto, Hiroki, 6317
Okumura, Manabu, 5797
Orbach, Matan, 5591
Oren, Yonatan, 4227
Osama, Muhammad, 6293
Ott, Myle, 6098
Ouchi, Hiroki, 3665
Ousidhoum, Nedjma, 4675
Ouyang, Yuanxin, 1140
Overwijk, Arnold, 5175

Padró, Lluís, 3451
Pal, Chris, 2796
Palfrey, Dave, 3591
Pallier, Christophe, 1150
Pan, Chunhong, 2068
Pan, Feiyang, 3033
Pan, Rong, 2623
Pan, Shimei, 728
Pan, Zhufeng, 1824
Pang, Bo, 1468, 1962
Papalampidi, Pinelopi, 1707
Papalexakis, Evangelos, 6376
Papasarantopoulos, Nikos, 2057
Paranjape, Bhargavi, 1083
Parida, Shantipriya, 5994
Parikh, Devi, 1449
Parikh, Pulkit, 1642
Park, Haeju, 5890
Park, Jong, 2954
Park, Jun-U, 6425
Park, Seong-Bae, 1436
Park, Sunghyun, 2840
Park, Youngja, 728
Parrish, Alicia, 2877
Parvez, Md Rizwan, 1167
Pasupat, Panupong, 1520
Patra, Barun, 6268
Patro, Jasabanta, 6336
Patton, Desmond, 4765
Paul, Michael J., 3471
Paul, Mithun, 3413
Paulik, Matthias, 4453
Pavlick, Ellie, 2230
Paz-Argaman, Tzuf, 6449
Peitz, Stephan, 4453
Peng, Hao, 1179, 3644
Peng, Minlong, 1040
Peng, Nanyun, 434, 1117, 3407, 6052, 6395
Peng, Wei, 2877
Peng, Yuxing, 1596
Peper, Joseph J., 1311
Perez, Ethan, 2402
Pergola, Gabriele, 3478
Perkins, Hugh, 4016
Peskov, Denis, 4526, 4831, 5918
Peters, Matthew E., 43
Pethe, Charuta, 1653
Petroni, Fabio, 2463
Petrov, Petre, 866
Petrov, Rostislav, 5636
Petrushkov, Pavel, 866
Peyrard, Maxime, 563
Pezzelle, Sandro, 2865
Phang, Jason, 2877
Piccinno, Francesco, 5902
Pineau, Joelle, 4506
Pino, Juan, 6098
Pinter, Yuval, 11
Piratla, Vihari, 4260
Pirtoaca, George Sebastian, 2531
Pitler, Emily, 5947
Piwowarski, Benjamin, 696, 3246
Poblete, Barbara, 718
Poerner, Nina, 1630
Polikarpova, Nadia, 6176
Ponti, Edoardo Maria, 2206, 2900
Popat, Kashyap, 6413
Poria, Soujanya, 154
Portillo Wightman, Gwenyth, 5753
Post, Matt, 910
Potdar, Saloni, 3566, 6456
Potts, Christopher, 3711, 4485
Prabhakaran, Vinodkumar, 5740
Prabhu, Ameya, 4058
Prag, Jonathan, 6368

Qi, Peng, 2590
Qi, Qi, 456, 5489
Qi, Tao, 4874, 4884, 6389
Qi, Weizhen, 6059
Qian, Dong, 4047
Qian, Jing, 4755
Qian, Kun, 1990
Qian, Lihua, 3173
Qian, Longhua, 5459
Qian, Peng, 2888
Qin, Bing, 302, 1586, 3022, 3143
Qin, Lianhui, 5043
Qin, Libo, 133, 2078
Qin, Pengda, 6038
QIN, Tao, 963, 2442, 4207, 4375, 5708
Qiu, Chen, 1229

Qiu, Delai, 5896
Qiu, Haoling, 6444
Qiu, Jiazuo, 5983
Qiu, Lin, 3173
Qiu, Xipeng, 3509
Qu, Meng, 2672
Qu, Yuzhong, 2614, 5866
Quan, Jun, 4547
Quan, Xiaojun, 4698

Rabinovich, Ella, 4776
Radev, Dragomir, 1962, 5338
Radinsky, Kira, 3873
Raghunathan, Aditi, 4129
Rahman, Wasifur, 2046
Rajagopal, Dheeraj, 5002
Ran, Qiu, 2474
Ranzato, Marc'Aurelio, 6098
Rao, Jinfeng, 1204, 5370
Rao, Sudha, 4831
Rao, Yuan, 4644
Rashkin, Hannah, 4463
Rasiwasia, Nikhil, 5911
Ravi, Sujith, 3894, 5012
Ravichander, Abhilasha, 4947
Ravindran, Balaraman, 3314
Ray, Arijit, 5860
Razniewski, Simon, 5771
Rebedea, Traian, 2531
Reddy, Vinay, 6071
Redmiles, Elissa, 4831
Rei, Marek, 1273, 2218
Reichart, Roi, 2206, 2900, 4407
Reimers, Nils, 3982
Reitter, David, 2131
Ren, Liliang, 1876
Ren, Shuo, 770
Ren, Xiang, 271, 445, 2672, 2829, 3841, 5753, 5777
Renduchintala, Adithya, 6438
Resnik, Philip, 1243
Ribeiro, Leonardo F. R., 3183
Riedel, Sebastian, 2463
Riesa, Jason, 3632
Rimell, Laura, 2284
Ringaard, Michael, 1229
Ringel, Dor, 3873
Roberto Filizzola Ortiz, Daniel, 3419
Rocktäschel, Tim, 673, 2463
Rogers, Anna, 4365
Romanov, Alexey, 4365
Rong, Wenge, 1140
Rosales-Méndez, Henry, 718
Rosenfeld, Alex, 4718
Rosenthal, Sara, 4864
Ross, Alexis, 2230
Rossini, Daniele, 4037
Roth, Dan, 3363, 3720, 3914, 6203, 6256
Rothe, Sascha, 5054
Roy, Arpita, 728
Rücklé, Andreas, 1607
Ruder, Sebastian, 974, 5702
Rumshisky, Anna, 4365
Ruseti, Stefan, 2531
Rush, Alexander, 1173, 1210

Sabharwal, Ashish, 2814
Sabir, Ahmed, 3451
Sadamasa, Kunihiko, 6225
Sadeh, Norman, 4947
Sagawa, Shiori, 4227
Saha, Swarnadeep, 6071
Saito, Jun, 5758
Sakaguchi, Keisuke, 6076
Salakhutdinov, Ruslan, 4344
Saligrama, Venkatesh, 1167
Sandholm, Anders, 1229
Sankar, Chinnadhurai, 4516
Sankaranarayanan, Karthik, 6144
Santra, Bishal, 6092
Santus, Enrico, 3097, 3419
Sap, Maarten, 4463
Sarawagi, Sunita, 4260
Savkov, Aleksandar, 77
Schluger, Jack, 2119
Schmidek, Jordan, 749
Schmidt, Florian, 3400
Schulte im Walde, Sabine, 4707
Schulz, Philip, 1488
Schuster, Tal, 3419
Schütze, Hinrich, 1185, 1630
Schwab, Michel, 6238
Schwartz, Roy, 43, 1179, 2185, 3644
Scialom, Thomas, 3246
Scozzafava, Federico, 3534
Sekine, Satoshi, 579, 3521
Selous, Charlotte, 4765
Sennrich, Rico, 877, 898, 1429, 4396
Seo, Minjoon, 3121
Seo, Seungmin, 2273
Seppi, Kevin, 3621
Sethares, William, 5549
Severyn, Aliaksei, 5054
Shafran, Izhak, 4979

Shah, Darsh, 3419
Shah, Pararth, 1951
Shah, Rushin, 1520
Shalyminov, Igor, 1741
Shang, Jingbo, 5154
Shang, Mingyue, 1909, 4937
Shao, Bo, 6059
Shao, Zhihong, 3257
Shapira, Ori, 3110
Sharma, Eva, 3280, 6343
Sharma, Piyush, 708, 1468
Sharma, Soumya, 6092
Sharp, Rebecca, 3413
Shaw, Peter, 5902
Sheikh, Zaid, 5164
Shen, April, 77
Shen, Dinghan, 3132, 5226
Shen, Longxiang, 2251
Shen, Tao, 2442
Shen, Xiaowei, 4601
Shen, Xiaoyu, 579, 3762
Shen, Yilin, 1279
Shen, Yulin, 5866
Sheng, Emily, 3407
Shi, Chuan, 3821, 4821
Shi, Jiaxin, 2723
Shi, Peng, 4431, 5370
Shi, Shuming, 887, 921, 953, 1336, 1403, 1866, 1898, 4937
Shi, Tianze, 1962, 5338
Shi, Wei, 5790
Shi, Weijia, 1198, 5276
Shi, Weiyan, 1990
Shi, Xiaofei, 813
Shi, Yuefeng, 3652
Shibaev, Viacheslav, 3936
Shim, Sungrok, 1962, 5338
Shimbo, Masashi, 2422
Shimorina, Anastasia, 3086
Shin, Cheolheon, 2273
Shin, Jamin, 121, 1297, 3541
Shoemark, Philippa, 66
Shou, Linjun, 2485
Shu, Lei, 1304
Shutova, Ekaterina, 2218
Si, Luo, 616, 2328, 5581, 6187
Sikka, Karan, 4622, 5860
Sil, Avirup, 313
Simonsen, Jakob Grue, 4685
Simpson, Edwin D., 1093
Singh, Maneesh, 4058
Singh, Munindar, 5528
Singh, Sameer, 43, 2153, 5307
Sinha, Koustuv, 4506
Sinno, Barea, 4718
Sips, Robert-Jan, 2349
Skiena, Steve, 1653
Slonim, Noam, 5591, 5625
Smith, Noah A., 43, 1179, 1494, 2185, 3644, 4153, 5925
Sneyd, Alison, 3484
Socher, Richard, 540, 1481, 1962, 5338
Sodhani, Shagun, 4506
Søgaard, Anders, 1105, 1229, 3336, 5838, 6330
Sommerschield, Thea, 6368
Song, Dawei, 4568
Song, Hyun-Je, 1436
Song, Kaisong, 198
Song, Linfeng, 208
Song, Yan, 3033, 5123, 5247
Song, Yang, 2662, 4810
Song, Yangqiu, 4675, 5123, 5247, 6210
Song, Yonghao, 1793
Soricut, Radu, 1468
Soto, Carlos, 3464
Soulier, Laure, 696
Specia, Lucia, 3227
Srikumar, Vivek, 3924
Srinivasan, Balaji Vasan, 3314
Srinivasan, Kumar Bhargav, 2911
Srivastava, Jaideep, 6432
Srivastava, Shashank, 4164
Srivatsan, Akshay, 2195
Stahlberg, Felix, 3356
Staiano, Jacopo, 3246
Stanforth, Robert, 4083
Stein, Benno, 2922
Stengel-Eskin, Elias, 910
Stepanjans, Armins, 4313
Stevens, Rock, 4831
Stevenson, Mark, 3484
Stevenson, Suzanne, 4776
Straka, Milan, 2779
Stratos, Karl, 421
Strötgen, Jannik, 6238
Strzyz, Michalina, 1500
Su, Hui, 579, 3762
SU, Jinsong, 845
su, jinsong, 208
Su, Jionglong, 4843
Su, Qi, 5508
Su, Qinliang, 4698, 5226

Su, Ruisi, 6343
Su, Tzu-ray, 910
Su, Yu, 5447
Su, Zhong, 4601, 5559
Subburathinam, Ananya, 313
Subramanian, Lakshminarayanan, 2338
Subramanian, Sanjay, 6203
Subramanian, Shivashankar, 1729
Sudhakar, Akhilesh, 3269
Sugiyama, Masashi, 3993
Suhr, Alane, 2119
Sui, Dianbo, 3830
Sui, Zhifang, 3388
Suleman, Kaheer, 3382
Sultani, Masih, 4776
Sumita, Eiichiro, 1361
Sun, Aixin, 3153
Sun, Changlong, 198, 5581, 6187
Sun, Chi, 3509
Sun, Haifeng, 456, 5489
Sun, Haitian, 2380
Sun, Huan, 410, 5447
Sun, Jian, 3904
Sun, Kai, 5679
Sun, Le, 292, 6232
Sun, Ling, 4644
Sun, Maosong, 219, 791, 5777, 6122, 6250
Sun, Ming-Ting, 3304
Sun, Mingming, 1675
Sun, Qingfeng, 476
Sun, Shengli, 476
Sun, Shizhao, 5405
Sun, Simeng, 1216
Sun, Siqi, 4323
SUN, Xu, 1618, 3388, 4431, 5508, 5518
Sung, Chul, 6071
Suntwal, Sandeep, 3413
Surdeanu, Mihai, 2578, 3413
Suzuki, Jun, 579, 1236, 3665
Szlam, Arthur, 673

Tafjord, Oyvind, 5941
Taitelbaum, Hagai, 1330, 3560
Takanobu, Ryuichi, 100
Takase, Sho, 6408
Takei, Yuka, 6317
Talukdar, Partha, 378
Tamura, Akihiro, 6244
Tan, Chenhao, 486, 2911
Tan, Chilie, 5144
Tan, Hao, 5100
Tan, Ming, 3566, 6456
Tan, Shulong, 5236
Tan, Xin, 1576
Tan, Xingwei, 3426
Tan, Xu, 963
Tan, Yi Chern, 1962
Tandon, Niket, 4496, 6076
Tang, Duyu, 1618, 2442
Tang, Gongbo, 1429
tang, hongyin, 5090
Tang, Jie, 1803
Tang, Jizhi, 2632
Tang, Lingjia, 1311
Tang, Raphael, 1204
Tang, Siliang, 271
Taniguchi, Tomoki, 3195
Tanveer, Md Iftekhar, 2046
Tao, Chongyang, 1291
Tar, Chris, 3687
Tater, Tarun, 6144
Tay, Yi, 5370
tebbifakhr, amirhossein, 1368
Temnikova, Irina, 2964
Teufel, Simone, 5267
Thompson, Brian, 1342, 1382
Thompson, Kate, 2296
Thorne, James, 2944
Tian, Fei, 5708
Tian, Jingjing, 445
Tian, Junfeng, 1222
Tikhonov, Alexey, 3936
Titov, Ivan, 877, 898, 1071, 3774, 4396, 5415
Tokala, Santosh, 6092
Toledo, Assaf, 5625
Toni, Francesca, 5195
Torisawa, Kentaro, 5816
Tou, Huaixiao, 5033
Tran, Linh, 4979
Tran, Quan Hung, 5953
Trichelair, Paul, 3382
Tripodi, Rocco, 88
Trischler, Adam, 2796, 3382
Tsai, Henry, 3632
Tsai, Yao-Hung Hubert, 4344
Tsarfaty, Reut, 6449
Tseng, Bo-Hsiang, 1273
Tsujii, Jun'ichi, 5393
Tsvetkov, Yulia, 1664, 4153
Tsygankova, Tatiana, 6256
Tu, Kewei, 1423, 5728
Tu, Zhaopeng, 887, 921, 931, 953, 1336, 1403, 1866

Tuan, Yi-Lin, 1855
Turchi, Marco, 1368
Turner, Richard, 1273
Tuzhilin, Alexander, 3237

Uban, Ana, 2317
Upadhyay, Bhargav, 3269
Urbanek, Jack, 673
Utiyama, Masao, 1361

Valle Torre, Manuel, 2349
van der Lee, Chris, 552
Van Durme, Benjamin, 910, 3786
Van Gool, Luc, 2088
van Miltenburg, Emiel, 552
van Schijndel, Marten, 5831
Vandenhende, Simon, 2088
Vania, Clara, 1105
Varma, Vasudeva, 1642
Venezian, Elad, 5625
Ver Steeg, Greg, 4026
Verga, Patrick, 1507
Versteegh, Maarten, 6419
Vilares, David, 1500
Vlachos, Andreas, 2284, 2944, 6350
Voita, Elena, 877, 4396
Voss, Clare, 313
Vulić, Ivan, 2206, 2900, 4407
Vyas, Yogarshi, 5285

Wachsmuth, Henning, 2922
Wadden, David, 5784
Wallace, Byron C., 21
Wallace, Eric, 2153, 5307
Wallach, Hanna, 5734
Wang, Bailin, 3774, 6216
Wang, Chao, 791
Wang, Chun, 456
Wang, Dakuo, 3566, 6456
Wang, Daling, 5066
Wang, Di, 165
Wang, Dongsheng, 4685
Wang, Gang, 3851
Wang, Ge, 5728
Wang, Guoyin, 3132
Wang, Haifeng, 856, 1782
Wang, Hao, 6286
Wang, Haoyu, 3566, 6456
Wang, Haozhou, 4419
Wang, Heng, 2623
Wang, Hongning, 2510
WANG, Houfeng, 3444
Wang, Huazheng, 2510
Wang, Jiancheng, 198, 5581
Wang, Jianyu, 4293
Wang, Jin, 3432
Wang, Jingjing, 5581
Wang, Jingyu, 456, 5489
Wang, Jinpeng, 3022
Wang, Jun, 4664
Wang, Liang, 4003
Wang, Lingfeng, 2068
Wang, Lingzhi, 4579
WANG, Liwei, 5708
Wang, Longbiao, 1920
Wang, Longyue, 921, 953, 1403
Wang, Lu, 591, 3280, 4633, 6343
Wang, Mingqiu, 4979
Wang, Mingxuan, 803
Wang, Prince Zizhuang, 4333
Wang, Qian, 3054
Wang, Rui, 282, 616, 1222, 1361
Wang, Ruifang, 1920
Wang, Shaonan, 3054
Wang, Sheng-Fu, 2877
Wang, Shijin, 1586, 5883
Wang, Shuo, 791
Wang, Songhe, 2553
Wang, Su, 2712
Wang, Wei, 2521, 4904
Wang, Weichao, 5066
Wang, Weikang, 1762
Wang, Weiping, 2412
Wang, Weiying, 5133
Wang, Weiyue, 1324
Wang, Wenbo, 3076
Wang, Wenlin, 3132
Wang, William Yang, 4333, 4354, 4755, 6038
Wang, Xiaozhi, 5777
Wang, Xin, 780, 1718, 2141
Wang, Xing, 887, 921, 953, 1336, 1403
Wang, Xuepeng, 3000
Wang, Xuewei, 1990
Wang, Yan, 1824, 1866
Wang, Yaushian, 1061
Wang, Yining, 3054, 3350
Wang, Yiren, 4207
Wang, Yisen, 1267
Wang, Yizhong, 5307
Wang, Yongcheng, 5133
Wang, Yongliang, 4959
Wang, Yu, 3597
Wang, Yuechen, 1618

Wang, Yunli, 3573
Wang, Yuxuan, 5721
Wang, Zheng, 240
Wang, Zhiguo, 208, 5878
Wang, Zhongqing, 5499
Wang, Zihan, 5154
Wang, Zihao, 250
Wang, Zijian, 2590, 3711
Wang, Ziqi, 5777
Wanvarie, Dittaya, 3993
Warstadt, Alex, 2877
Washio, Koki, 3521
Watanabe, Taiki, 6244
Watson, William, 2241
Webber, Bonnie, 4547
Wei, Furu, 2109, 4143
Wei, Jason, 6382
Wei, Penghui, 4787
Wei, Ying, 4590
Wei, Yu'ang, 5866
Wei, Zhongyu, 1040, 5033
Wei, Zihan, 1718
Weikum, Gerhard, 5771, 6413
Welbl, Johannes, 4083
Weld, Dan, 3693
Weld, Daniel, 5803
Weller, Orion, 3621
Wen, Haoyang, 133, 2078
Wen, Ji-Rong, 4810, 4914
Wen, Jiangtao, 3257
Wennberg, Ulme, 5784
West, Peter, 3752
Weston, Jason, 673, 1951, 2402, 4537
White, Marshall, 6343
Wiegreffe, Sarah, 11
Wilcox, Ethan, 2888
Williams, Adina, 5734
Wilson, Shomir, 4947
Winata, Genta Indra, 1297, 3541
Wintner, Shuly, 4153
Wolf-Sonkin, Lawrence, 5734
Wong, Kam-Fai, 4579, 4633
Wu, Chen, 2521
Wu, Chien-Sheng, 3065
Wu, Chuhan, 4874, 4884, 6389
Wu, Fangzhao, 4874, 4884, 6389
Wu, Fei, 271
Wu, Hao, 4249
Wu, Hua, 496, 856, 1782
Wu, Jiawei, 4354
Wu, Lianwei, 4644
Wu, Lijun, 4207, 4375
Wu, Ruidong, 219
Wu, Shijie, 833
wu, wei, 1291, 1886, 5077
Wu, Winston, 2241
Wu, Xiao-Ming, 4799
Wu, Xixin, 4579
Wu, Yike, 5559
Wu, Yu, 770, 1834, 3573
Wu, Yuanbin, 6133
Wu, Yunfang, 3394, 6032
Wu, Yuting, 240
Wu, Yuxiang, 2463
Wu, Zhonghai, 2662

Xenouleas, Stratos, 6005
Xia, Feng, 6402
Xia, Jiangnan, 2521
Xia, Qiaolin, 1494
Xia, Qingrong, 5382
Xia, Yingce, 963, 4207
Xiang, Bing, 5878
Xiang, Shinming, 2068
Xiang, Yuanxin, 942
Xiao, Jun, 2109, 5144
Xiao, Li, 5883
Xiao, Lin, 466
Xiao, Shiyuan, 1140
Xiao, Tong, 3585
Xiao, Wen, 3011
Xiao, Xinyan, 496
Xiao, Xinyu, 2068
Xiao, Yanghua, 813
Xie, Dongliang, 780
Xie, Jiateng, 5164
Xie, Jing Yi, 4654
Xie, Jun, 5508
Xie, Ruobing, 219
Xie, Xing, 4874, 4884, 6389
Xin, Ji, 5823
Xing, Linzi, 3471
Xiong, Caiming, 540, 1481, 1962, 5338
Xiong, Chenyan, 5175
Xiong, Deyi, 1576, 2452, 4547, 5983
xiong, hao, 856
Xiong, JinJun, 518
Xiong, Lee, 5175
Xiong, Wenhan, 4354
Xiong, Zhang, 1140
Xu, Can, 1886, 5077
Xu, Chi, 2262
Xu, Guandong, 357

Xu, Hengru, 496
Xu, Hu, 1304
Xu, Jiacheng, 3292
Xu, Jie, 326
Xu, Jinan, 1051, 2652, 5569
Xu, Jingfang, 6122
Xu, Jingjing, 1618, 5508, 5518
Xu, Jingyun, 357
Xu, Ke, 4143
Xu, Kun, 5247
Xu, Nan, 4787
Xu, Peng, 121, 1297, 3065
Xu, Qixin, 2614
Xu, Ruifeng, 3478, 5614, 6280
Xu, Tong, 5489
Xu, Wei, 337
Xu, Weiwen, 942
Xu, Wenfei, 3257
Xu, Yan, 1297
Xu, Yanfu, 2412
Xu, Yang, 4654
Xu, Zenan, 4698
Xu, Zhaozhuo, 5236
Xue, Eric, 1962, 5338

Yadav, Vikas, 2578
Yamada, Makoto, 4344
Yamshchikov, Ivan P., 3936
Yan, Claudia, 2119
Yan, Hanqi, 5518
Yan, Haoran, 5766
Yan, Hongyu, 5614
Yan, Lingyong, 292
Yan, Ming, 2521
Yan, Rui, 1291, 1909, 1931, 3741, 4937, 4959
Yan, Shiyan, 3968
Yan, Zhen, 2452
Yang, Changjiang, 4731
Yang, Cheng, 3821
Yang, Hongxia, 1803
Yang, Hsiu-Wei, 4431
Yang, Huiyun, 4197
Yang, Jian, 1886
Yang, Jianxin, 1140
Yang, Jie, 2349
Yang, Min, 3033, 5614, 6280
Yang, Pengcheng, 1618, 3388, 5508
Yang, Qian, 3132
Yang, Qiang, 4590
Yang, Tianchi, 4821
Yang, Ting, 1920
Yang, Wei, 1204, 3490, 5370
Yang, Weiwei, 1243
Yang, Wonsuk, 2954
Yang, Xiaopeng, 1931
Yang, Xiyuan, 271
Yang, Xuefeng, 229
Yang, Xuewen, 780
Yang, Yan, 6286
Yang, Yang, 177
Yang, Yi, 4016
Yang, Yikang, 4843
Yang, Yiming, 3603
Yang, Yinfei, 3687
Yang, Ze, 1886, 5077
Yang, Zhenglu, 1222
Yang, Zhengxin, 1527
Yang, Zhichao, 6155
Yang, Ziyi, 638
Yao, Yuan, 219
Yao, Ziyu, 5447
Yarowsky, David, 2241
Yasunaga, Michihiro, 1962
Yates, Andrew, 6413
Yatskar, Mark, 4069
Yavuz, Semih, 4516
Ye, Qinyuan, 3841
Yee, Kyra, 5696
Yeh, Leigh, 5753
Yeh, Luke, 6355
Yeh, Yi-Ting, 3370
Yeo, Jinyoung, 2840
Yeo, Yun Jie Serene, 3419
Yeung, Dit-Yan, 4675
Yih, Wen-tau, 4496, 5447
Yin, Dawei, 1793, 3000
Yin, Eddy Jing, 4914
Yin, Mingming, 3044
Yin, Wenpeng, 3914
Yin, Yongjing, 845
Yogatama, Dani, 4083
Yoo, Kang Min, 3528
Yoo, Shinjae, 3464
yoon, seungwon, 2954
Yordanov, Yordan, 4303
You, Zhenjun, 1909
Youssef, Adel, 4526
Yu, Bei, 4664
Yu, Changlong, 5247
Yu, Dian, 1513
Yu, Dong, 5247
Yu, Hong, 4249
yu, hong, 6155

Yu, Hongfei, 3044
Yu, Jingsong, 1834
Yu, Kai, 3637
Yu, Keyi, 1192
Yu, Liang-Chih, 3432
Yu, Mo, 208, 3566, 4094, 6456
Yu, Ning, 4731
Yu, Seunghak, 5636
Yu, Stanley, 2119
Yu, Tao, 1962, 5338
Yu, Xintong, 5123
Yu, Yang, 3566
Yu, Yaoliang, 5823
Yu, Yong, 3173
Yu, Zhou, 143, 1513, 1990
Yuan, Chunyuan, 111
Yuan, Xingdi, 2796

Zablocki, Eloi, 696
Zeng, Daojian, 367
Zeng, Jiali, 845
Zeng, Michael, 1261
Zeng, Min, 1267
Zeng, Qi, 1618, 5518
Zeng, Xiangrong, 367
Zeng, Xiangyu, 1279
Zeng, Xingshan, 4633
Zettlemoyer, Luke, 1520, 2851, 4069, 5360, 5426, 5436, 5803, 6112
Zhang, Biao, 898
Zhang, Bo, 1772
Zhang, Byoung-Tak, 2024
Zhang, Changshui, 5123
Zhang, Chen, 4568
Zhang, Cheng, 1793
Zhang, Chunliang, 3585
Zhang, Dongmei, 5316, 5405
Zhang, Dongyu, 6402
Zhang, Fubao, 2692
Zhang, Haisong, 4579
Zhang, Haotian, 3490
Zhang, Heting, 6402
Zhang, Hongming, 4675, 5123, 5247
Zhang, Huijun, 1718
Zhang, Iris, 2119
Zhang, Jiajun, 1762, 3054, 3350
Zhang, Jinchao, 1051, 1527, 5569
Zhang, Jinfeng, 887, 1336
Zhang, Kai, 1834
Zhang, Lei, 1475, 2141
Zhang, Li, 4601
Zhang, Longyin, 1576
Zhang, Luhao, 3821
Zhang, Maosen, 3841
Zhang, Meishan, 997
Zhang, Meng, 177
Zhang, Min, 3044, 5382, 5459, 5499, 5581
Zhang, Minghua, 3394, 6032
Zhang, Peng, 1285
Zhang, Pengchuan, 2141
Zhang, Qi, 1040, 2692
Zhang, Qiong, 198, 2328, 6187
Zhang, Richong, 326, 5679
Zhang, Rui, 1962, 5338
Zhang, Sheng, 3786
Zhang, Shiyue, 2495
Zhang, Tao, 2662, 4810
Zhang, Vincent, 1962
Zhang, Wei, 4217
Zhang, Weijia, 4698
Zhang, Weinan, 3173
Zhang, Wen, 4217
Zhang, Xiaodong, 3444
Zhang, Xiaoming, 1772
Zhang, Xiaotong, 4799
Zhang, Xuan, 1382
Zhang, Xuchao, 5002
Zhang, Xuejie, 3432
Zhang, Yang, 4094
Zhang, Yi, 4526
Zhang, Yichang, 1803
Zhang, Yifei, 5066
Zhang, Yin, 4558
Zhang, Yizhe, 1814, 3304
Zhang, Yu, 4590
Zhang, Yuan, 3687
Zhang, Yuanzhe, 5896
Zhang, Yue, 208, 616, 997, 3044, 3652, 4115
Zhang, Yujie, 2652
Zhang, Yuwen, 1962
Zhang, Zhe, 5528
Zhang, Zhiqiang, 4959
Zhang, Zhongfei, 2001, 3548
Zhao, Dongyan, 240, 1291, 1909, 1931, 2632, 3741, 4937, 4959
Zhao, Hai, 5350
Zhao, He, 5206
Zhao, Jieyu, 5276
Zhao, Jun, 367, 738, 2431, 3830, 5896
Zhao, Liang, 4293, 5518
Zhao, Lujun, 198
Zhao, Shiwan, 4601, 5559
Zhao, Tiejun, 5689

Zhao, Tong, 302
Zhao, Wayne Xin, 4810, 4914
Zhao, Wei, 563, 4003
Zhao, Xiangyu, 1920
Zhao, Xiaofang, 1793
Zhao, Xiaoxi, 1249
Zhao, Yang, 3762
Zhao, Zhenjie, 3957, 6376
Zhao, Zhou, 2109
Zhao, Zijian, 3637
Zhelezniak, Vitalii, 77
Zheng, Anqing, 5206
Zheng, Baigong, 1349, 1395
Zheng, Changmeng, 357
Zheng, Haitao, 347
Zheng, Renjie, 1349, 1395
Zheng, Shun, 337
Zheng, Xin, 3153
Zheng, Zaixiang, 931
Zhong, Jianyuan, 2046
Zhong, Peixiang, 165
Zhong, Ruiqi, 4765
Zhou, Ben, 3363
Zhou, Bin, 5316
Zhou, Chunting, 1388, 4282
ZHOU, Deyu, 177, 282
Zhou, Guodong, 1576, 1898, 2251, 5459, 5499, 5581
Zhou, Jie, 1051, 1527, 2474, 5569, 5777, 6250
Zhou, Kevin, 476
Zhou, Kun, 1834
Zhou, Lianqiang, 1824
Zhou, Lin, 5866
Zhou, Long, 3350
Zhou, Ming, 770, 1618, 2485
Zhou, Mingyang, 143
Zhou, Pei, 1198, 5276
Zhou, Wei, 111
Zhou, Wenjie, 3394, 6032
Zhou, Xiabing, 5499
Zhou, Yichao, 4904
Zhou, Yu, 3054
zhou, yu, 3478
Zhou, Yuxiang, 3076
Zhou, Zhixin, 5236
Zhu, Changxi, 3426
Zhu, Chenguang, 638, 1261
Zhu, Conghui, 5689
Zhu, Fuqing, 111
Zhu, Hanlin, 100
Zhu, Hao, 6250
Zhu, Jie, 5459
Zhu, Jingbo, 3585
Zhu, Jinhua, 4375
Zhu, Junnan, 3054
Zhu, Kenny, 1028
Zhu, Muhua, 5459
Zhu, Qiaoming, 2251
Zhu, Su, 3637
Zhu, Xiaodan, 1845, 3904
zhu, xiaoyan, 3257, 4271
Zhu, Yao, 2662
Zhuang, Yueting, 271, 2109
Ziegler, Zachary, 1210
Zlatkova, Dimitrina, 2099
Zong, Chengqing, 1762, 3000, 3054, 3350
Zong, Lu, 4843
Zou, Bowei, 2251
Zou, Kai, 6382
Zou, Yanyan, 4431, 5327
Zou, Yicheng, 1040